사용자 가이드

아마존 CodeCatalyst

아마존 CodeCatalyst: 사용자 가이드

Table of Contents

아마존이란 CodeCatalyst 무엇입니까?

CodeCatalyst Amazon은 소프트웨어 개발 프로세스에 지속적인 통합 및 배포 방식을 도입하는 소프트웨어 개발 팀을 위한 통합 서비스입니다. CodeCatalyst 필요한 도구를 모두 한 곳에 모았습니다. CI/CD (지속적 통합/지속적 전달) 도구를 사용하여 작업을 계획하고, 코드에 대해 협업하고, 애플리케이션을 빌드, 테스트 및 배포할 수 있습니다. 공간에 연결하여 AWS 리소스를 프로젝트와 통합할 수도 있습니다. AWS 계정 CodeCatalyst 애플리케이션 라이프사이클의 모든 단계와 측면을 하나의 도구로 관리하면 소프트웨어를 빠르고 확실하게 제공할 수 있습니다.

에서는 회사 CodeCatalyst, 부서 또는 그룹을 대표하는 공간을 만든 다음 개발 팀과 작업을 지원하는 데 필요한 리소스가 포함된 프로젝트를 만듭니다. CodeCatalyst리소스는 공간 내에 있는 프로젝트 내에서 구조화됩니다. 팀이 빠르게 시작할 수 있도록 언어 또는 도구 기반 프로젝트 청사진을 CodeCatalyst 제공합니다. 프로젝트 블루프린트에서 프로젝트를 만들면 프로젝트에는 샘플 코드가 있는 소스 리포지토리, 빌드 스크립트, 배포 작업, 가상 서버 또는 서버리스 리소스 등과 같은 리소스가 함께 제공됩니다.

어떻게 할 수 있나요? CodeCatalyst

여러분과 개발팀은 작업 계획부터 애플리케이션 CodeCatalyst 배포에 이르기까지 소프트웨어 개발의 각 측면을 수행하는 데 사용할 수 있습니다. CodeCatalyst를 사용하여 다음을 수행할 수 있습니다.

- 코드 반복 및 공동 작업 — 소스 코드 리포지토리의 브랜치, 병합, 풀 리퀘스트, 주석을 사용하여 팀과 함께 코드 작업을 진행하세요. 리포지토리를 복제하거나 리포지토리에 대한 연결을 설정할 필요 없이 개발 환경을 만들어 코드를 빠르게 작업할 수 있습니다.
- 워크플로를 사용하여 애플리케이션 빌드, 테스트, 배포 — 애플리케이션의 지속적 통합 및 딜리버리를 처리할 수 있도록 빌드, 테스트, 배포 작업으로 워크플로를 구성합니다. 워크플로를 수동으로 시작하거나 코드 푸시, 풀 요청 생성 또는 종료와 같은 이벤트에 따라 자동으로 시작되도록 구성할 수 있습니다.
- 이슈 트래킹으로 팀 작업의 우선순위를 정하세요. 이슈를 사용하여 백로그를 만들고 보드를 통해 진행 중인 작업의 상태를 모니터링하세요. 팀이 작업할 수 있는 항목의 백로그를 정상적으로 만들고 유지하는 것은 소프트웨어 개발의 중요한 부분입니다.
- 모니터링 및 알림 설정 — 팀 활동 및 리소스 상태를 모니터링하고 중요한 변경 사항을 최신 상태로 유지하도록 알림을 구성합니다.

시작하려면 어떻게 해야 하나요 CodeCatalyst?

공간이 없거나 공간을 설정하고 관리하는 방법을 알아보려면 Amazon CodeCatalyst 관리자 안내서로 시작하는 것이 좋습니다.

프로젝트나 스페이스에서 처음 작업하는 경우 다음과 같이 시작하는 것이 좋습니다.

- 리뷰 CodeCatalyst 개념
- 스페이스 만들기
- 다음 단계에 따라 첫 번째 프로젝트 생성하기 튜토리얼: 모던 3계층 웹 애플리케이션 블루프린트로 프로젝트 생성

에 대해 자세히 알아보기 CodeCatalyst

이 사용 설명서의 CodeCatalyst 기능과 다음 리소스에 대해 자세히 알아볼 수 있습니다.

- AWS DevOps Amazon에 대한 블로그 기사 CodeCatalyst
- 아마존 CodeCatalyst API 레퍼런스 가이드
- Amazon CodeCatalyst 액션 개발 키트 개발자 가이드
- CodeCatalyst FAQ
- 추천사

CodeCatalyst 개념

Amazon에서의 협업 및 애플리케이션 개발 속도를 높이는 데 도움이 되는 주요 개념을 숙지하십시오 CodeCatalyst. 이러한 개념에는 소스 제어, 지속적 통합 및 지속적 전송 (CI/CD), 자동 릴리스 프로세스 모델링 및 구성에 사용되는 용어가 포함됩니다.

추가 개념 정보는 다음 항목을 참조하십시오.

- 소스 리포지토리 개념
- 워크플로우 개념

주제

- AWS 빌더 ID 스페이스의 CodeCatalyst
- ID 페더레이션을 지원하는 스페이스 CodeCatalyst
- 프로젝트
- 블루프린트
- 계정 연결
- VPC연결
- AWS 빌더 ID
- 사용자 프로필: CodeCatalyst
- 소스 리포지토리
- 커밋
- 개발 환경
- 워크플로
- 작업
- 문제
- 개인용 액세스 토큰 (PATs)
- 개인 연결
- 역할

AWS 빌더 ID 스페이스의 CodeCatalyst

스페이스 관리자는 구성원 페이지에서 개별 초대 이메일을 CodeCatalyst 보내 사용자를 초대합니다. 초대를 받았거나 등록하여 CodeCatalyst 자신만의 AWS 빌더 ID를 만든 사용자. 프로필은 AWS 빌더 ID에서 관리되며 사용자 설정의 사용자 설정에 사용자 이름 및 프로필 정보로 표시됩니다 CodeCatalyst.

ID 페더레이션을 지원하는 스페이스 CodeCatalyst

IAMIdentity Center 인스턴스의 SSO 사용자 및 그룹에 추가되어 ID 저장소에서 관리되며 Identity Center를 통해 IAM 스페이스로 초대되는 사용자. 스페이스 관리자는 CodeCatalyst 구성원 페이지를 동기화하여 최신 업데이트를 확인합니다. 사용자는 회사 IAM Identity Center 인스턴스에 설정된 SSO 로그인 포털을 사용하여 로그인합니다. ID 페더레이션을 지원하는 스페이스는 Identity Center 애플리케이션을 통해 ID 저장소 인스턴스에 연결되고 ID 저장소 ID에 매핑됩니다.

프로젝트

프로젝트는 개발 팀과 작업을 CodeCatalyst 지원하는 공동 노력을 나타냅니다. 프로젝트가 완성되면 사용자와 리소스를 추가, 업데이트 또는 제거하고, 프로젝트 대시보드를 사용자 지정하고, 팀 작업의 진행 상황을 모니터링할 수 있습니다. 스페이스 내에 여러 프로젝트를 포함할 수 있습니다.

프로젝트에 대한 자세한 내용은 을 참조하십시오프로젝트 관련 작업을 체계적으로 정리하세요 CodeCatalyst.

블루프린트

블루프린트는 콘솔에서 프로젝트를 생성하는 동시에 애플리케이션 지원 파일 및 종속성을 생성 및 확장하는 프로젝트 신시사이저입니다. CodeCatalyst 에서 선택한 블루프린트 중에서 프로젝트 유형을 선택하고, README 파일을 보고 CodeCatalyst, 생성될 프로젝트 리포지토리와 리소스를 미리 볼 수 있습니다. 프로젝트는 블루프린트에 지정된 기본 구성에서 생성됩니다. 정기적으로 프로젝트 블루프린트에 합성하여 소프트웨어 종속성과 같은 프로젝트 파일을 업데이트하고 리소스를 재생성합니다. 프로젝트는 Projen이라는 도구를 사용하여 최신 프로젝트 업데이트를 동기화하고 지원 파일을 생성하여 프로젝트를 합성합니다. 이러한 파일에는 응용 프로그램 유형 및 `eslint` 언어에 따라 `package.jsonMakefile`, 등이 포함될 수 있습니다. 프로젝트 CDK 블루프린트는 구성, AWS CloudFormation 템플릿, 템플릿과 같은 AWS 리소스를 지원하는 파일을 생성할 수 있습니다. AWS Serverless Application Model

프로젝트 블루프린트에 대한 자세한 내용은 을 참조하십시오. CodeCatalyst 청사진이 포함된 포괄적인 프로젝트 만들기

계정 연결

계정 연결은 스페이스를 내 CodeCatalyst 스페이스와 연결합니다. AWS 계정계정 연결이 설정되면 스페이스에서 AWS 계정 사용할 수 있습니다. 그런 다음 내 리소스에 액세스할 수 CodeCatalyst 있도록 IAM 역할을 추가할 수 있습니다 AWS 계정. CodeCatalyst 워크플로 작업에도 이러한 역할을 사용할 수 있습니다.

프로젝트 제한 계정 연결을 활성화하여 계정 연결에 액세스할 수 있는 프로젝트 및 리소스를 제한할 수 있습니다. 스페이스의 지정된 프로젝트만 액세스할 수 AWS 계정 있는 프로젝트 제한 계정 연결이 연결됩니다. 이렇게 하면 스페이스에 있는 팀이 프로젝트별로 통합 AWS 리소스에 AWS 계정 대한 사용을 제한할 수 있습니다. 예를 들어 특정 프로젝트의 배포 워크플로 및 VPC 연결에 사용되는 계정은 프로젝트 제한 계정 연결을 통해서만 사용할 수 있습니다. 자세한 내용은 프로젝트 제한 계정 연결 구성을 참조하십시오.

계정 연결에 대한 자세한 내용은 을 참조하십시오. 연결된 AWS 리소스에 대한 액세스 허용 AWS 계정

VPC연결

VPC연결은 워크플로가 a에 액세스하는 데 필요한 모든 구성을 포함하는 CodeCatalyst 리소스입니다 VPC. 스페이스 관리자는 스페이스 구성원을 대신하여 Amazon CodeCatalyst 콘솔에서 자신의 VPC 연결을 추가할 수 있습니다. VPC연결을 추가하면 스페이스 구성원은 워크플로 작업을 실행하고 네트워크 규칙을 준수하고 관련 VPC 리소스에 액세스할 수 있는 개발 환경을 만들 수 있습니다.

VPC연결에 대한 자세한 내용은 CodeCatalyst 관리자 안내서의 Amazon 가상 사설 클라우드 관리를 참조하십시오.

AWS 빌더 ID

AWS 빌더 ID는 다른 참여 애플리케이션에 가입하고 로그인하는 데 사용할 수 있는 개인 ID입니다. CodeCatalyst 이는 a와 동일하지 않습니다 AWS 계정. AWS 빌더 ID는 사용자 별칭 및 이메일 주소와 같은 메타데이터를 관리합니다. AWS 빌더 ID는 내의 모든 스페이스에서 CodeCatalyst 사용자를 지원하는 고유한 ID입니다. AWS 빌더 ID 프로필 접속에 대한 자세한 내용은 을 참조하십시오프로필 업데이트. AWS 빌더 ID에 대해 자세히 알아보려면 의 AWS 빌더 ID를 참조하십시오 AWS 일반 참조.

가입 및 로그인에 대한 자세한 내용은 을 참조하십시오설정 및 로그인 CodeCatalyst.

사용자 프로필: CodeCatalyst

의 아무 페이지에서나 로그인 이니셜 아래에 있는 드롭다운에서 프로필 옵션을 선택하여 CodeCatalyst 사용자 프로필에 액세스할 수 있습니다. CodeCatalyst 프로필 페이지에서 개인용 액세스 토큰 (PATs) 을 생성할 수 있지만 를 PATs 사용해서만 보거나 삭제할 수 있습니다. AWS CLI사용자 이름은 가입할 때 선택한 별칭입니다. 사용자 이름은 변경할 수 없습니다. 다른 CodeCatalyst 사용자의 프로필 페이지를 보려면 프로젝트의 멤버 탭으로 이동하여 적절한 사용자를 선택하세요.

CodeCatalyst 프로필을 확인한 다음 AWS 빌더 ID로 이동하면 AWS 빌더 ID에 액세스할 수 있습니다. AWS 빌더 ID 프로필 페이지로 리디렉션됩니다. 프로필의 전체 이름, 이메일 주소, 비밀번호는 빌더 ID로 관리되며 AWS 빌더 ID 페이지를 사용하여 해당 정보를 편집할 수 있습니다. AWS 가입할 때 이 정보를 입력했습니다. 로그인에 인증 애플리케이션을 MFA 사용하도록 설정할 준비가 되면 AWS 빌더 ID 페이지를 사용하게 됩니다. AWS 빌더 ID 프로필 보기에 대한 자세한 내용은 을 참조하십시오. 프로필 업데이트

가입 및 로그인에 대한 자세한 내용은 을 참조하십시오설정 및 로그인 CodeCatalyst.

소스 리포지토리

소스 리포지토리는 프로젝트의 코드와 파일을 안전하게 저장하는 곳입니다. 또한 파일의 버전 기록도 저장합니다. 기본적으로 소스 리포지토리는 CodeCatalyst 프로젝트의 다른 사용자와 공유됩니다. 프로젝트의 소스 리포지토리는 두 개 이상 있을 수 있습니다. 에서 프로젝트의 소스 리포지토리를 만들거나 CodeCatalyst, 설치된 확장 프로그램에서 해당 서비스를 지원하는 경우 다른 서비스에서 호스팅하는 기존 소스 리포지토리를 연결하도록 선택할 수 있습니다. 예를 들어 리포지토리 확장을 설치한 후 GitHub GitHub 리포지토리를 프로젝트에 연결할 수 있습니다. 자세한 내용은 프로젝트의 리포지토리에 소스 코드 저장 CodeCatalyst 및 빠른 시작: 확장 프로그램 설치, 공급자 연결, 리소스 연결 CodeCatalyst 단원을 참조하세요.

소스 리포지토리에는 CI/CD 워크플로의 속성 및 작업을 정의하는 구성 파일과 같은 CodeCatalyst 프로젝트의 구성 정보가 저장되는 곳이기도 합니다. 블루프린트를 사용하여 프로젝트를 생성하면 프로젝트 구성 정보가 저장된 소스 리포지토리가 생성됩니다. 빈 프로젝트를 생성하는 경우 워크플로와 같은 구성 정보가 필요한 리소스를 만들려면 먼저 소스 리포지토리를 만들어야 합니다.

소스 리포지토리 및 소스 컨트롤 작업에 도움이 될 수 있는 추가 개념은 을 참조하십시오. 소스 리포지토리 개념

커밋

커밋은 파일 또는 파일 세트에 대한 변경입니다. Amazon CodeCatalyst 콘솔에서 커밋은 변경 내용을 저장하고 소스 리포지토리에 푸시합니다. 커밋에는 변경한 사용자의 ID, 변경 시간 및 날짜, 커밋 제목, 변경에 대한 모든 메시지 등 변경 정보가 포함됩니다. 자세한 내용은 Amazon에서 커밋을 사용한 소스 코드의 변경 사항 이해 CodeCatalyst 단원을 참조하십시오.

의 소스 리포지토리 컨텍스트에서 CodeCatalyst 커밋은 리포지토리 내용에 대한 변경 사항의 스냅샷입니다. 사용자가 변경 내용을 커밋하고 푸시할 때마다 변경 내용을 커밋한 사람, 커밋 날짜 및 시간, 커밋의 일부로 변경한 내용 등의 정보가 CodeCatalyst 저장됩니다. 또한 커밋에 Git 태그를 추가하여 특정 커밋을 식별할 수 있습니다.

커밋에 대한 자세한 내용은 을 참조하십시오. Amazon에서 커밋을 사용한 소스 코드의 변경 사항 이해 CodeCatalyst

개발 환경

개발 환경은 프로젝트의 소스 리포지토리에 저장된 코드를 빠르게 작업하는 CodeCatalyst 데 사용할 수 있는 클라우드 기반 개발 환경입니다. Dev Environment에 포함된 프로젝트 도구와 애플리케이션 라이브러리는 프로젝트의 소스 저장소에 있는 devfile에 의해 정의됩니다. 소스 리포지토리에 devfile이 없는 경우 기본 devfile이 자동으로 적용됩니다. 기본 devfile에는 가장 자주 사용되는 프로그래밍 언어 및 프레임워크용 도구가 포함되어 있습니다. 기본적으로 개발 환경은 2코어 프로세서, 4GB, 16GiB의 RAM 영구 스토리지로 구성됩니다.

워크플로

워크플로는 지속적 통합 및 지속적 전달 (CI/CD) 시스템의 일부로 코드를 빌드, 테스트 및 배포하는 방법을 설명하는 자동화된 절차입니다. 워크플로는 워크플로 실행 중에 수행할 일련의 단계 또는 조치를 정의합니다. 또한 워크플로는 워크플로를 시작하게 하는 이벤트 또는 트리거를 정의합니다. 워크플로를 설정하려면 CodeCatalyst 콘솔의 시각적 또는 YAML 편집기를 사용하여 워크플로 정의 파일을 만듭니다.

> **Tip**
>
> 프로젝트에서 워크플로를 사용하는 방법을 간단히 살펴보려면 청사진을 사용하여 프로젝트를 만들어 보세요. 각 블루프린트는 검토, 실행, 실험할 수 있는 작동하는 워크플로를 배포합니다.

워크플로에 대한 자세한 내용은 워크플로를 통한 빌드, 테스트, 배포 섹션을 참조하세요.

작업

작업은 워크플로의 기본 구성 요소이며, 워크플로 실행 중에 수행할 논리적 작업 단위 또는 작업을 정의합니다. 일반적으로 워크플로에는 구성 방법에 따라 순차적으로 또는 병렬로 실행되는 여러 작업이 포함됩니다.

작업에 대한 자세한 내용은 을 참조하십시오워크플로우 작업 구성.

문제

이슈는 프로젝트와 관련된 작업을 추적하는 레코드입니다. 기능, 작업, 버그 또는 프로젝트와 관련된 기타 작업에 대한 이슈를 생성할 수 있습니다. 애자일 개발을 사용하는 경우 에픽 또는 사용자 스토리를 설명하는 이슈가 될 수도 있습니다.

문제에 대한 자세한 내용은 을 참조하십시오. 에서 문제가 있는 작업을 추적하고 정리하세요. CodeCatalyst

개인용 액세스 토큰 (PATs)

개인용 액세스 토큰 (PAT) 은 비밀번호와 비슷합니다. 이는 내 모든 공간 및 프로젝트에서 사용할 수 있도록 사용자 ID와 연결되어 CodeCatalyst 있습니다. 통합 개발 환경 (IDEs) 및 Git 기반 소스 리포지토리를 포함하는 CodeCatalyst 리소스에 액세스하는 PATs 데 사용합니다. PATs사용자를 CodeCatalyst 대표하고 사용자 설정에서 관리할 수 있습니다. 한 사용자가 둘 이상을 가질 수 PAT 있습니다. 개인용 액세스 토큰은 한 번만 표시됩니다. 로컬 컴퓨터에 안전하게 보관하는 것이 가장 좋습니다. 기본적으로 1년 후에 PATs 만료됩니다.

에 대한 자세한 내용은 PATs 을 참조하십시오개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여.

개인 연결

개인 연결은 사용자 CodeCatalyst ID와 외부 소스 제공업체 (예:) 간의 GitHub 승인입니다. 개인 연결을 사용하여 CodeCatalyst 사용자가 타사 소스 리포지토리를 추가할 수 있도록 허용합니다. 예를 들어 GitHub 저장소를 스페이스에 연결할 수 있습니다. CodeCatalyst 설치된 커넥터 응용 프로그램은 GitHub 계정 소유자가 지정한 저장소에서 사용할 수 있도록 계정에 설치됩니다. 특정 공급자 유형 (예:)

의 모든 스페이스에서 하나의 사용자 ID (CodeCatalyst 별칭) 에 대해 하나의 개인 연결을 생성할 수 있습니다. GitHub 개인 연결은 AWS 빌더 ID 또는 SSO 사용자와 연결됩니다.

자세한 내용은 개인적인 연결을 통해 GitHub 리소스에 접근하기 단원을 참조하십시오.

역할

역할은 프로젝트 또는 스페이스의 리소스에 대한 사용자의 액세스 권한과 사용자가 취할 수 있는 작업을 정의합니다. 사용자를 프로젝트에 초대할 때 해당 사용자의 역할을 선택합니다. 에는 스페이스 수준 역할과 프로젝트 수준 역할이 있습니다. CodeCatalyst 올바른 수준의 관리 역할을 가진 사용자는 할당된 역할을 변경할 수 있습니다. 예를 들어 프로젝트의 프로젝트 관리자 역할을 가진 사용자는 해당 프로젝트를 완전히 제어할 수 있으며 해당 프로젝트에서 사용자의 역할을 변경할 수 있습니다. 사용 가능한 역할 및 각 역할에 부여되는 권한에 대한 자세한 내용은 을 참조하십시오사용자 역할을 통한 액세스 권한 부여.

역할에 관한 자세한 내용은 사용자 역할을 통한 액세스 권한 부여 단원을 참조하세요.

설정 및 로그인 CodeCatalyst

다음과 같은 두 가지 유형의 공간을 설정할 수 있습니다. 하나는 AWS Builder ID 사용자를 지원하는 공간이고, 다른 하나는 Identity Center에서 SSO IAM 사용자와 그룹을 관리하는 ID 페더레이션을 지원하는 공간을 만드는 것입니다. CodeCatalyst AWS Builder ID 스페이스의 사용자는 AWS Builder CodeCatalyst ID로 로그인하고 ID 페더레이션을 위해 설정된 스페이스의 사용자는 스페이스와 연결된 회사의 SSO 포털을 CodeCatalyst 사용하여 로그인합니다.

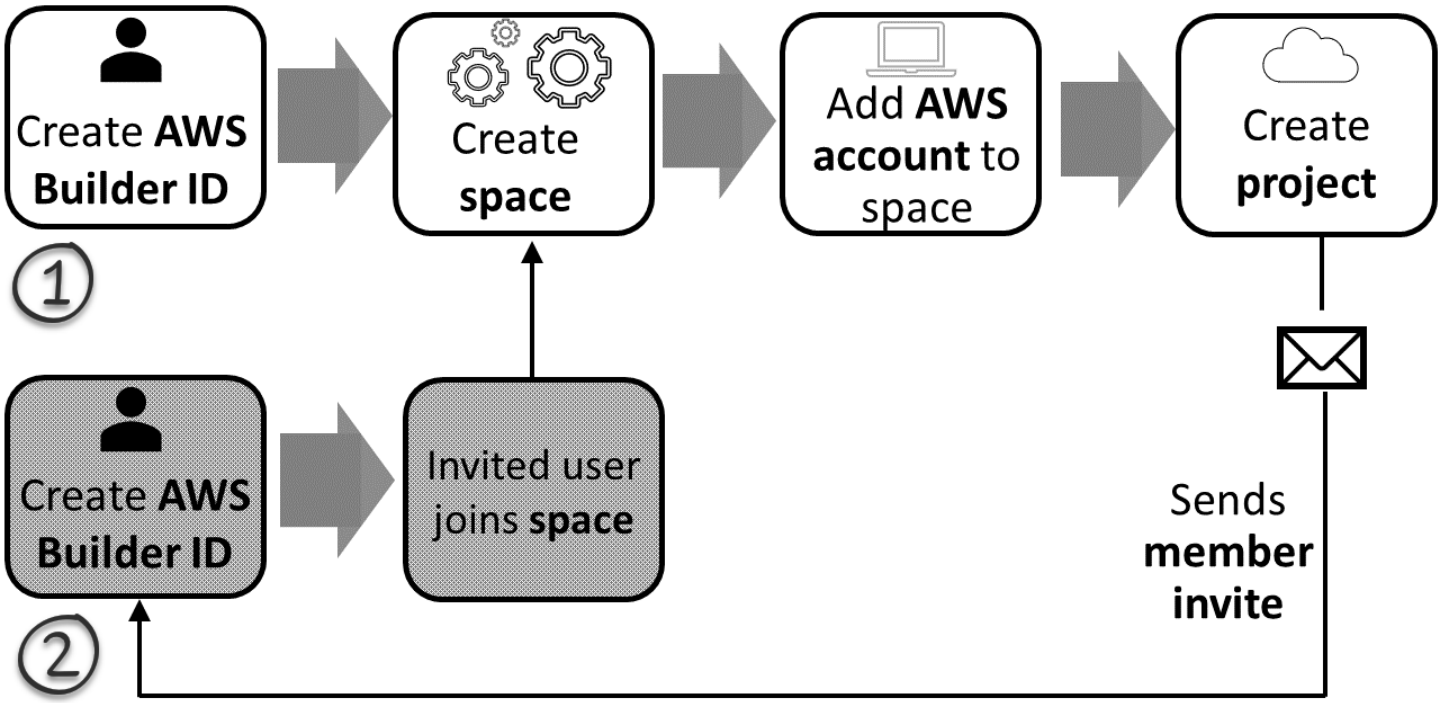
> **Note**
>
> CodeCatalyst 사용자 이름의 최소 길이는 3자에서 최대 100자입니다. 입력한 사용자 이름이 100자를 초과하면 잘립니다. 이로 인해 사용자 이름이 다른 100자 사용자 이름과 중복된 것처럼 보일 수 있습니다. 자세한 내용은 사용자 이름이 잘려서 새 사용자로 내 BID 스페이스에 액세스할 수 없거나 새 SSO 사용자로 추가할 수 없습니다. 단원을 참조하십시오.

AWS 빌더 ID 공간을 설정하고 관리하는 단계는 이 안내서에 나와 있습니다. CodeCatalyst AWS 빌더 ID 스페이스를 사용하려면 로그인할 때 사용하는 사용자 설정과 AWS 빌더 ID를 CodeCatalyst 사용하여 설정해야 합니다. CodeCatalyst

ID 페더레이션을 지원하는 공간을 설정하고 관리하는 단계는 CodeCatalyst 관리자 안내서에 나와 있습니다. ID 페더레이션을 위해 설정된 스페이스를 사용하려면 Amazon CodeCatalyst 관리자 안내서의 CodeCatalyst 스페이스 설정 및 관리를 참조하십시오.

이 섹션에서는 Amazon에서 AWS Builder ID CodeCatalyst 스페이스로 작업하도록 설정하는 두 가지 일반적인 경로를 제공합니다. 하나는 첫 번째 사용자로 스페이스와 프로젝트를 생성하는 것이고, 다른 하나는 기존 스페이스 또는 프로젝트에 대한 초대를 수락하는 것입니다. 이러한 설정 워크플로는 반드시 상당히 다릅니다. 다음 다이어그램은 두 등록 프로세스를 다음과 같이 보여줍니다.

1. 첫 번째 경우에는 회사, 팀 또는 그룹을 위한 공간을 만들어 설정하고 프로젝트를 먼저 만든 다음, 이러한 리소스에 다른 사람을 초대합니다. 청구를 위해 AWS 계정 반드시 를 제공해야 하며, 그래도 프리 티어를 기본으로 사용할 수 있습니다.
2. 두 번째 경우에는 프로젝트 초대를 CodeCatalyst 수락하여 참여하면 다른 사람이 이미 스페이스와 프로젝트를 만들어 준 것입니다. 하지만 다른 사람들과 함께 작업을 시작할 준비가 되려면 여전히 프로필을 구성하는 것이 좋습니다.



> **Tip**
>
> CodeCatalyst 스페이스를 사용하여 프로젝트와 리소스를 그룹화합니다. 처음 CodeCatalyst 등록하면 공간뿐만 아니라 프로젝트도 만들라는 메시지가 표시됩니다.

공간 및 프로젝트를 만들기 위해 등록하든 초대를 수락하기 위해 등록하든 관계없이 로그인하는 데 사용할 AWS 빌더 ID가 생성됩니다 CodeCatalyst. AWS 빌더 ID를 생성하려면 AWS 애플리케이션에 로그인할 때 사용하는 전체 이름, 암호, 이메일 주소를 입력합니다. 이 시점 CodeCatalyst 이후에는 이메일과 비밀번호를 사용하여 로그인합니다. 이 AWS 빌더 ID를 사용하여 빌더 ID 자격 증명을 사용하는 AWS 다른 애플리케이션에 로그인할 수도 있습니다.

AWS 빌더 ID 내부 CodeCatalyst 및 빌더 ID에서는 로그인 정보를 기반으로 프로필이 생성됩니다. 프로필에는 CodeCatalyst 프로젝트의 언어 및 알림 CodeCatalyst 설정에 대한 기본 설정이 포함됩니다.

> **Tip**
>
> Amazon CodeCatalyst 프로필에 가입하는 동안 문제가 발생하는 경우 해당 페이지에 제공된 단계를 따르십시오. 추가 지원이 필요한 경우 을 참조하십시오가입 문제.

주제

첫 번째 스페이스 및 개발 역할 만들기 (초대 없이 시작)

기존 스페이스나 프로젝트에 CodeCatalyst 초대하지 않고도 Amazon에 가입할 수 있습니다. 등록하면 AWS 빌더 ID를 생성한 후 공간과 프로젝트를 생성하게 됩니다. 공간 생성의 일환으로 청구 AWS 계정 목적으로 공간을 추가해야 합니다.

> **Tip**
>
> Amazon CodeCatalyst 프로필에 가입하는 동안 문제가 발생하는 경우 해당 페이지에 제공된 단계를 따르십시오. 추가 지원이 필요한 경우 을 참조하십시오가입 문제.

다음은 프로젝트나 스페이스에 CodeCatalyst 초대하지 않고 시작하는 사용자가 사용할 수 있는 한 가지 방법입니다.

Mary Major는 이 분야에 관심이 CodeCatalyst 있고 시도해 보기로 결정한 개발자입니다. 그녀는 CodeCatalyst 콘솔로 가서 가입하고 AWS 빌더 ID를 생성하는 옵션을 선택합니다. Mary는 AWS 빌더 ID를 생성하는 데 필요한 이메일 주소와 비밀번호를 제공합니다. 그녀는 자신의 AWS 빌더 ID를 사용하여 다른 애플리케이션에 로그인할 CodeCatalyst 수 있습니다. 별칭을 선택하라는 메시지가 표시되면 다른 프로젝트 멤버가 @mention Mary에 CodeCatalyst 사용할 CodeCatalyst 사용자 이름으로 표시합니다. `MaryMajor`

다음으로 Mary는 자동으로 스페이스를 만들라는 안내를 받습니다. 이 흐름의 일환으로 Mary는 첫 번째 프로젝트 빌드 및 배포에서 샘플 코드를 볼 수 있도록 자신이 만들고 AWS 계정 있는 공간에 a를 연결해 달라는 요청을 받습니다. 그녀는 이 정보를 추가하고 스페이스를 만들고, 새 스페이스에서 프로젝트에 사용할 수 있는 미리 보기 개발 역할을 만들 수 있는 옵션을 선택합니다. Mary는 프로젝트를 만들기로 선택한 다음 프로젝트의 청사진 목록을 확인합니다. 사용 가능한 청사진에 대한 정보를 검토한

후, 그녀는 첫 번째 프로젝트에 Modern 3-tier 웹 애플리케이션 청사진을 사용해 보기로 합니다. 그녀는 필수 필드를 채우고 프로젝트를 생성합니다. 프로젝트가 준비되자마자 그녀는 프로젝트 요약 페이지로 이동합니다. 이 페이지에는 최근 활동은 물론 프로젝트 코드와 해당 코드를 자동으로 빌드하고 배포하는 워크플로로 연결되는 링크가 포함되어 있습니다. 그녀는 배포된 샘플 웹 애플리케이션 보기를 포함하여 코드와 워크플로를 모두 살펴봅니다. 마음에 들었던 그녀는 동료 몇 명을 프로젝트에 초대해 탐색을 시작하기로 결심합니다. CodeCatalyst

시간이 나면 Mary는 다중 요소 인증 (MFA) CodeCatalyst 으로 로그인할 AWS 빌더 ID를 구성합니다. MFA를 구성하면 Mary는 CodeCatalyst 암호와 승인된 타사 인증 앱의 암호 또는 토큰을 조합하여 CodeCatalyst 로그인할 수 있습니다.

첫 번째 스페이스 및 IAM 역할 생성

다음 단계에 따라 Amazon CodeCatalyst 프로필에 가입하고, 스페이스를 생성하고, 스페이스에 대한 계정, 지원 역할 및 개발자 역할을 추가하십시오.

마지막 절차에서는 개발자 역할을 생성하고 추가합니다. 개발자 역할은 CodeCatalyst 워크플로가 AWS 리소스에 액세스할 수 있게 해주는 AWS IAM 역할입니다. 개발자 역할은 관리하는 데 사용되는 서비스 AWS 서비스 역할이며 로그인한 계정에서 생성됩니다. 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 역할에는 이름이 `CodeCatalystWorkflowDevelopmentRole-spaceName` 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오CodeCatalystWorkflowDevelopmentRole-**spaceName**서비스 역할 이해.

> **Note**
>
> 보안 모범 사례에 따라 AWS 스페이스의 리소스에 대한 액세스를 관리해야 하는 관리자 및 개발자에게만 관리 액세스 권한을 할당하십시오.

시작하기 전에 관리자 권한이 있는 계정의 AWS 계정 ID를 제공할 준비가 되어 있어야 합니다. 12자리 AWS 계정 ID를 준비하세요. ID 찾기에 대한 자세한 내용은 AWS 계정 AWS 계정 ID 및 별칭을 참조하십시오.

새 사용자 등록

1. CodeCatalyst 콘솔에서 시작하기 전에 를 열고 스페이스를 만들 때 사용하려는 AWS 계정 것과 동일한 계정으로 로그인했는지 확인하십시오. AWS Management Console

2. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
3. 시작 페이지에서 가입을 선택합니다. AWS Builder ID 만들기 페이지가 표시됩니다. AWS 빌더 ID는 로그인하기 위해 생성하는 ID입니다. a와 동일하지 않습니다 AWS 계정.
4. 이메일 주소에 연결하려는 이메일 주소를 입력합니다 CodeCatalyst. 다음을 선택합니다.
5. AWS 빌더 ID를 사용하는 애플리케이션에 표시할 이름과 성을 사용자 이름에 입력합니다. 공백은 허용됩니다. 이 이름은 AWS 빌더 ID 프로필 이름입니다 (예: Mary Major). 이름은 나중에 변경할 수 있습니다.

   다음을 선택합니다. 이메일 확인 페이지가 표시됩니다.
6. 지정한 이메일로 확인 코드가 전송됩니다. 인증 코드에 이 코드를 입력한 다음 확인을 선택합니다. 5분 후에도 코드를 받지 못하고 스팸 폴더나 정크 폴더에서 코드를 찾을 수 없는 경우 코드 재전송을 선택하세요.
7. 코드가 확인되면 암호 및 암호 확인의 요구 사항을 충족하는 암호를 입력합니다.

   AWS 고객 계약 및 AWS 서비스 약관에 대한 동의를 확인하는 확인란을 선택한 다음 AWS Builder ID 생성을 선택합니다.
8. CodeCatalyst 별칭 만들기 페이지에서 고유 사용자 식별자로 사용할 별칭을 입력합니다. CodeCatalyst 예를 들어, 공백 없이 단축된 이름을 선택하세요. MaryMajor 다른 CodeCatalyst 사용자들은 댓글과 풀 리퀘스트에서 이 정보를 사용하여 여러분을 @mention 로 여길 것입니다. CodeCatalyst 프로필에는 AWS 빌더 ID의 전체 이름과 CodeCatalyst 별칭이 모두 포함됩니다. CodeCatalyst 별칭은 나중에 변경할 수 없습니다.

   전체 이름과 별칭은 의 여러 영역에 표시됩니다. CodeCatalyst 예를 들어 활동 피드에 나열된 활동에 대한 프로필 이름이 표시되지만 프로젝트 멤버는 사용자의 별칭을 @mention you 이름으로 사용합니다.

   다음을 선택합니다. 페이지가 업데이트되어 CodeCatalyst 스페이스 만들기 섹션이 표시됩니다.
9. 스페이스 이름에 스페이스 이름을 입력합니다. 나중에 변경할 수 없습니다.

   > **Note**
   >
   > 스페이스 이름은 전체에서 고유해야 CodeCatalyst 합니다. 삭제된 스페이스의 이름은 재사용할 수 없습니다.
10. AWS 리전드롭다운 메뉴에서 공간 및 프로젝트 데이터를 저장할 지역을 선택합니다. 나중에 변경할 수 없습니다.

11. 다음을 선택합니다. 페이지를 추가하기 위한 페이지가 표시되도록 페이지가 AWS 계정업데이트됩니다. 이 계정은 스페이스의 결제 계정으로 사용됩니다.
12. AWS 계정 ID에 스페이스에 연결하려는 계정의 12자리 ID를 입력합니다.

    AWS 계정 확인 토큰에서 생성된 토큰 ID를 복사합니다. 토큰은 자동으로 복사되지만 AWS 연결 요청을 승인하는 동안 저장하는 것이 좋습니다.
13. 확인을 위해 [AWS 콘솔로 이동] 을 선택합니다.
14. 에서 Amazon CodeCatalyst 스페이스 확인 페이지가 열립니다 AWS Management Console. 아마존 CodeCatalyst 스페이스 페이지입니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.

    에서 AWS Management Console공간을 만들려는 AWS 리전 위치와 동일한 위치를 선택해야 합니다.

    페이지에 직접 액세스하려면 https://console.aws.amazon.com/codecatalyst/home/ AWS Management Console 에서 Amazon CodeCatalyst Spaces에 로그인하십시오.

    의 확인 토큰 AWS Management Console 필드는 에서 CodeCatalyst 생성된 토큰으로 자동으로 채워집니다.
15. (선택 사항) 승인된 유료 티어에서 유료 티어 승인 (Standard, Enterprise) 을 선택하여 결제 계정의 유료 티어를 활성화합니다.

    > **Note**
    >
    > 이렇게 해도 청구 등급이 유료 등급으로 업그레이드되지는 않습니다. 하지만 이렇게 하면 언제든지 숙소의 청구 등급을 변경할 수 AWS 계정 있도록 구성됩니다. CodeCatalyst 언제든지 유료 등급을 켤 수 있습니다. 이렇게 변경하지 않으면 공간은 프리 티어만 사용할 수 있습니다.
16. 스페이스 확인을 선택합니다.

    계정이 스페이스에 추가되었음을 알리는 계정 확인 성공 메시지가 표시됩니다.
17. Amazon CodeCatalyst 스페이스 확인 페이지에 그대로 남아 있어야 합니다. 다음 링크를 선택하십시오. 이 스페이스에 IAM 역할을 추가하려면 스페이스 세부 정보를 확인하십시오.

    CodeCatalyst 스페이스 세부 정보가 있는 연결 페이지가 에서 AWS Management Console열립니다. 아마존 CodeCatalyst 스페이스 페이지입니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.
18. CodeCatalyst 페이지로 돌아가서 다음을 선택합니다.

19. 스페이스가 생성되는 동안 상태 메시지가 표시됩니다. 스페이스가 생성되면 CodeCatalyst 다음 메시지가 표시됩니다. 스페이스가 준비되었습니다. 마지막 단계는 프로젝트를 만드는 것입니다. . 다음 중 하나를 수행할 수 있습니다.

    - 지금은 건너뛰기를 선택하세요.
    - 스페이스에 사용할 첫 번째 프로젝트 만들기를 선택합니다. 블루프린트로 프로젝트를 만드는 방법을 보여주는 튜토리얼은 을 참조하십시오. 튜토리얼: 모던 3계층 웹 애플리케이션 블루프린트로 프로젝트 생성

    > **Note**
    >
    > 권한 오류 또는 배너가 표시되면 페이지를 새로 고치고 페이지를 다시 확인해 봅니다.

생성 및 추가하기 CodeCatalyst CodeCatalystWorkflowDevelopmentRole-***spaceName***

1. CodeCatalyst 콘솔에서 시작하기 전에 를 AWS Management Console열고 AWS 계정 스페이스에 동일한 계정으로 로그인했는지 확인하십시오.
2. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 여십시오.
3. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.
4. 역할을 생성하려는 AWS 계정 위치의 링크를 선택합니다. AWS 계정 세부 정보 페이지가 표시됩니다.
5. 에서 역할 관리를 선택합니다 AWS Management Console.

   Amazon CodeCatalyst 스페이스에 IAM 역할 추가 페이지가 에서 AWS Management Console열립니다. 아마존 CodeCatalyst 스페이스 페이지입니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.
6. IAM에서 CodeCatalyst 개발 관리자 역할 생성을 선택합니다. 이 옵션은 개발 역할에 대한 권한 정책 및 신뢰 정책을 포함하는 서비스 역할을 생성합니다. 역할에는 이름이 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오CodeCatalystWorkflowDevelopmentRole-***spaceName***서비스 역할 이해.

> **Note**
>
> 이 역할은 개발자 계정에만 사용하는 것이 좋으며 `AdministratorAccess` AWS 관리형 정책을 사용하므로 이 계정에서 새 정책 및 리소스를 만들 수 있는 전체 액세스 권한을 AWS 계정부여합니다.

7. 개발 역할 생성을 선택합니다.
8. 연결 페이지의 사용 가능한 IAM 역할 아래에서 계정에 CodeCatalyst 추가된 IAM 역할 목록의 역할을 확인합니다. `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*
9. 숙소로 돌아가려면 Amazon으로 이동을 선택하십시오 CodeCatalyst.

만들고 추가하려면 CodeCatalyst AWSRoleForCodeCatalystSupport

1. CodeCatalyst 콘솔에서 시작하기 전에 를 AWS Management Console열고 AWS 계정 스페이스에 동일한 계정으로 로그인했는지 확인하십시오.
2. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.
3. 역할을 생성하려는 AWS 계정 위치의 링크를 선택합니다. AWS 계정 세부 정보 페이지가 표시됩니다.
4. 에서 역할 관리를 선택합니다 AWS Management Console.

   Amazon CodeCatalyst 스페이스에 IAM 역할 추가 페이지가 에서 AWS Management Console열립니다. 아마존 CodeCatalyst 스페이스 페이지입니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.
5. CodeCatalyst 스페이스 세부 정보에서 CodeCatalyst Support 역할 추가를 선택합니다. 이 옵션은 미리 보기 개발 역할에 대한 권한 정책 및 신뢰 정책을 포함하는 서비스 역할을 생성합니다. 역할에는 고유 식별자가 AWSRoleForCodeCatalystSupport추가된 이름이 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오AWSRoleForCodeCatalystSupport서비스 역할 이해.
6. CodeCatalyst Support용 역할 추가 페이지에서 기본값을 선택된 상태로 두고 역할 생성을 선택합니다.
7. 사용 가능한 IAM 역할에서 계정에 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 추가된 IAM 역할 목록에서 역할을 확인하십시오. CodeCatalyst
8. 숙소로 돌아가려면 Amazon으로 이동을 선택하십시오 CodeCatalyst.

AWS 빌더 ID를 생성하고, 첫 번째 스페이스를 만들고, 계정을 추가한 후 프로젝트를 생성할 수 있습니다. 자세한 정보는 프로젝트 생성을 참조하세요. 처음 사용하는 CodeCatalyst 경우 먼저 시작하는 것이 좋습니다튜토리얼: 모던 3계층 웹 애플리케이션 블루프린트로 프로젝트 생성.

초대 수락 및 AWS 빌더 ID 생성

프로젝트 또는 스페이스에 대한 초대를 수락하는 과정에서 CodeCatalyst Amazon에 가입할 수 있습니다. 초대를 수락하는 과정에서 AWS 빌더 ID를 생성하라는 메시지가 표시됩니다. AWS 빌더 ID를 사용하여 의 리소스에 액세스할 수 CodeCatalyst 있습니다.

> **Tip**
>
> 추가 도움이 필요한 경우 을 참조하십시오가입 문제.

다음은 프로젝트 또는 스페이스에 대한 초대를 받고 CodeCatalyst 시작하는 사용자가 사용할 수 있는 한 가지 방법입니다.

Saanvi Sarkar는 CodeCatalyst 프로젝트 관리자로 프로젝트에 참여하라는 초대를 받은 개발자입니다. Saanvi가 초대를 수락하면 로그인 페이지가 열립니다. CodeCatalyst 그녀는 가입을 선택하고 빌더 ID를 생성하는 데 필요한 이메일 주소와 비밀번호를 제공합니다. AWS Saanvi는 AWS 빌더 ID를 사용하여 다른 애플리케이션에 로그인할 CodeCatalyst 수 있습니다. 나중에 프로필을 편집하여 로그인 이메일 주소 또는 비밀번호를 변경할 수 있습니다. 별칭을 선택하라는 메시지가 표시되면 Saanvi는 표시할 `SaanviSarkar` CodeCatalyst 별칭으로 지정하고 다른 프로젝트 멤버가 @mention Saanvi에 CodeCatalyst 사용할 별칭으로 지정합니다. 가입하고 나면 Saanvi는 Builder ID 자격 증명을 사용하는 다른 애플리케이션에서도 로그인 자격 증명을 사용할 수 있게 됩니다. AWS

등록을 완료하면 Saanvi는 초대장에 지정된 CodeCatalyst 프로젝트와 스페이스에 자동으로 참여합니다. 또한 이 초대장에는 프로젝트와 공간에서 그녀가 맡은 역할에 대한 사전 정의된 권한이 제공됩니다. 프로젝트 설정에서 Saanvi의 별칭은 할당된 프로젝트 역할과 함께 멤버 목록에 표시됩니다. 에서 CodeCatalyst 소스 리포지토리를 사용하기 위해 Saanvi는 잠시 시간을 내어 개인용 액세스 토큰 (PAT)을 생성합니다. PAT는 소스를 변경하거나 인증 토큰이 CodeCatalyst 필요한 작업을 수행할 때 인증에 사용됩니다.

Saanvi가 프로젝트를 작업할 때 별칭은 해당 프로젝트의 작업 활동 로그에 나열됩니다. Saanvi가 발행한 이슈와 댓글에는 별칭이 표시되며, 다른 프로젝트 멤버들은 Saanvi의 답글을 통해 Saanvi의 별칭을 @mention 로 표시할 수 있습니다. 다른 프로젝트 멤버를 @mention 처리하기 위해 Saanvi는 프로필에서 해당 멤버의 별칭을 찾습니다. CodeCatalyst

시간이 나면 Saanvi는 AWS 빌더 ID를 구성하여 다단계 인증 (MFA) CodeCatalyst 으로 로그인합니다. MFA를 구성하면 Saanvi는 승인된 CodeCatalyst 타사 인증 앱의 CodeCatalyst 암호와 암호 또는 토큰을 조합하여 로그인할 수 있습니다.

초대 수락 및 빌더 ID 생성 AWS

CodeCatalystAmazon의 프로젝트 또는 스페이스에 초대를 받으면 notify@codecatalyst.aws 으로부터 초대를 수락하라는 이메일을 받게 됩니다. 이미 AWS 빌더 ID가 있고 로그인한 경우 초대 수락을 CodeCatalyst 선택하면 브라우저 탭에서 프로젝트 또는 스페이스가 자동으로 열립니다. 콘솔에 로그인하지 않았지만 AWS 빌더 ID가 있는 경우 로그인 페이지로 이동합니다. 자세한 정보는 AWS빌더 ID로 로그인을 참조하세요.

AWS 빌더 ID가 없는 경우 초대 수락을 선택하면 로그인 페이지로 이동하며, 여기서 AWS 빌더 ID 생성 옵션을 선택할 수 있습니다.

초대를 수락하고 AWS 빌더 ID를 만들려면

1. 초대 이메일에서 초대 수락을 선택합니다.
2. 로그인 페이지에서 가입하지 않았나요? 를 선택합니다. AWS 빌더 ID를 만드세요.

   > **Tip**
   >
   > AWS 빌더 ID는 로그인하기 위해 생성하는 ID입니다. a와 동일하지 않습니다 AWS 계정.

3. AWS 빌더 ID 만들기 페이지의 이메일 주소에 AWS 빌더 ID로 사용할 이메일 주소를 입력합니다.

   AWS 빌더 ID를 사용하는 애플리케이션에 표시할 이름과 성을 사용자 이름에 입력합니다. 공백은 허용됩니다. 이 이름은 AWS 빌더 ID 프로필 이름입니다 (예: Mary Major). 이름은 나중에 변경할 수 있습니다.

   다음을 선택합니다.

   지정한 이메일로 확인 코드가 전송됩니다. 인증 코드에 이 코드를 입력한 다음 확인을 선택합니다. 5분 후에도 코드를 받지 못하고 스팸 폴더나 정크 폴더에서 코드를 찾을 수 없는 경우 코드 재전송을 선택하세요.
4. 코드가 확인되면 암호 및 암호 확인의 요구 사항을 충족하는 암호를 입력합니다.
5. AWS 빌더 ID 생성을 선택합니다.
6. 별칭 만들기 페이지에서 고유 사용자 식별자로 사용할 별칭을 입력합니다. CodeCatalyst 예를 들어, 공백 없이 단축된 이름을 선택하세요. MaryMajor 다른 CodeCatalyst 사용자들은 댓글과 풀 리

퀘스트에서 이 정보를 사용하여 여러분을 @mention 로 여길 것입니다. CodeCatalyst 프로필에는 AWS 빌더 ID의 전체 이름과 CodeCatalyst 별칭이 모두 포함됩니다. CodeCatalyst 별칭은 변경할 수 없습니다.

전체 이름과 별칭은 의 여러 영역에 표시됩니다. CodeCatalyst 예를 들어 활동 피드에 나열된 활동의 프로필 이름이 표시되지만 프로젝트 멤버는 사용자의 별칭을 사용하여 @mention you 이름을 사용합니다.

별칭 생성을 선택합니다. 초대를 받은 프로젝트 또는 스페이스로 이동하게 됩니다.

AWS빌더 ID로 로그인

Amazon CodeCatalyst 프로필에 로그인하려면 다음 단계를 따르십시오.

> **Note**
>
> 다단계 인증 (MFA) 을 위한 디바이스를 아직 등록하지 않으셨나요? 보안을 강화하려면 Amazon에서 MFA를 구성하는 CodeCatalyst 것이 좋습니다. 자세한 정보는 다단계 인증에 사용할 장치를 등록하는 방법을 참조하세요.

AWS빌더 ID로 로그인하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 이메일 주소를 입력합니다. 나중에 로그인할 수 있도록 이메일 주소를 저장하려면 [내 이메일 주소 저장] 을 선택합니다. 계속을 선택합니다.
3. 비밀번호를 입력합니다. [Sign in]을 선택합니다. 비밀번호가 기억나지 않는 경우 다음 단계를 따르세요암호를 잊어버렸습니다.

신뢰할 수 있는 기기

로그인 페이지에서 '신뢰할 수 있는 디바이스' 옵션을 선택하면 Amazon은 향후 해당 디바이스에서의 모든 로그인을 승인된 CodeCatalyst 것으로 간주합니다. CodeCatalyst Amazon은 신뢰할 수 있는 디바이스를 사용하는 한 MFA 코드를 입력할 수 있는 옵션을 제공하지 않습니다. 새 브라우저에서 로그인하거나 디바이스에 알 수 없는 IP 주소를 발급받은 경우는 예외입니다.

SSO로 로그인

다음 단계에 따라 SSO를 사용하여 CodeCatalyst Amazon에 로그인하십시오.

대신 AWS 빌더 ID로 로그인하려면 을 참조하십시오AWS빌더 ID로 로그인.

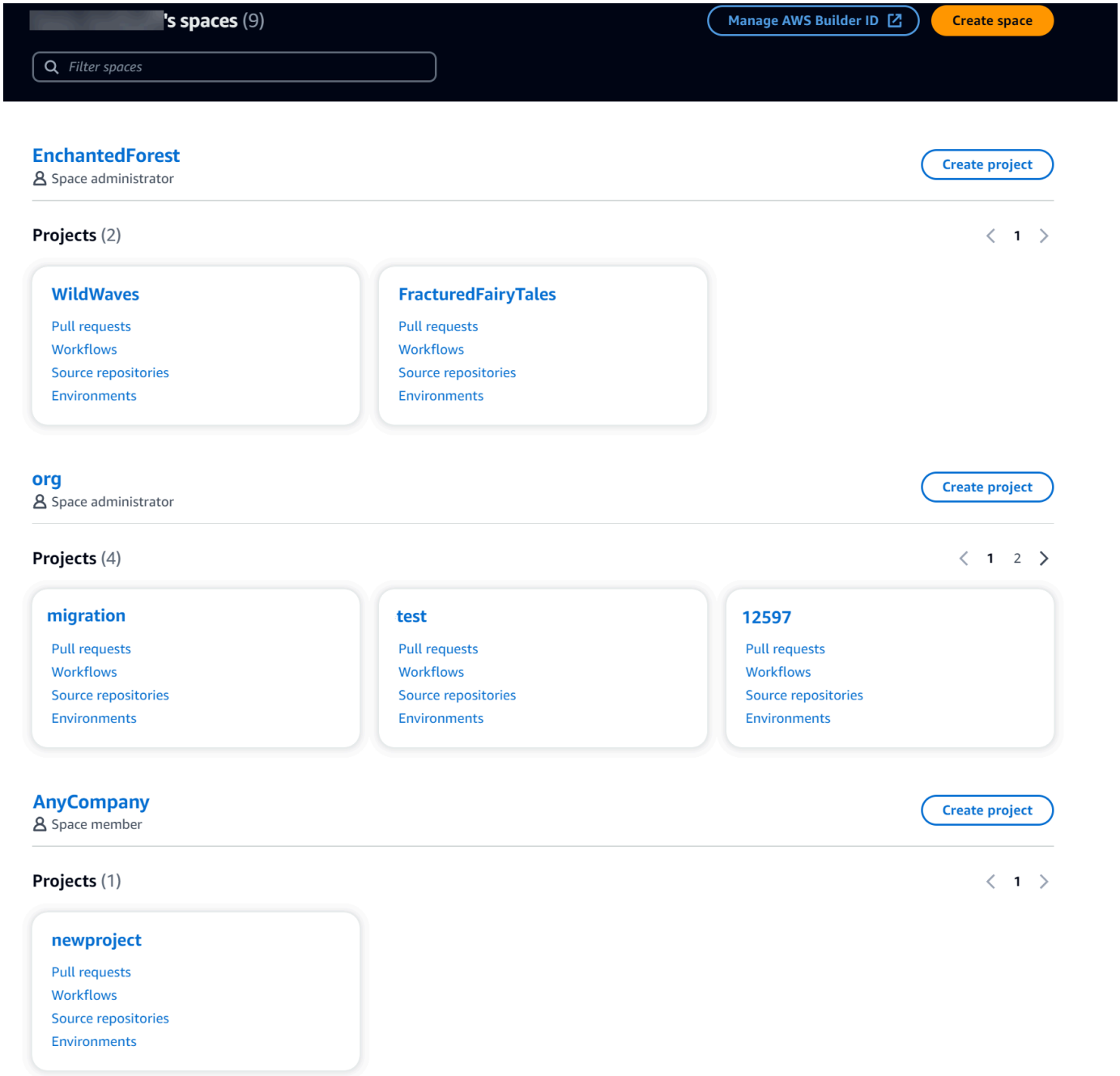
SSO로 로그인하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 로그인 옵션 선택에서 싱글 사인온 (SSO) 사용을 선택합니다.
3. AWSIdentity Center 애플리케이션 이름에 ID 페더레이션 관리자가 제공한 애플리케이션 이름을 입력합니다.
4. [IAM ID 센터로 계속] 을 선택합니다.

사용자의 모든 공간 및 프로젝트 보기

사용자 홈 페이지에서 스페이스 및 프로젝트 목록을 볼 수 있습니다. 사용자 홈 페이지에는 사용자가 속한 각 스페이스, 해당 스페이스의 사용자 역할 (예: 스페이스 관리자), 사용자가 멤버십을 보유한 각 스페이스의 프로젝트 목록이 표시됩니다.

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 브라우저에서 다음 주소를 입력합니다. https://codecatalyst.aws/home



3. 열려는 스페이스나 프로젝트를 선택합니다. 예상했던 스페이스나 프로젝트가 보이지 않는 경우 다른 사용자로 로그인해야 할 수 있습니다.

CodeCatalyst 프로필 보기 및 관리

CodeCatalyst Amazon에서 사용자 프로필을 보고 이메일 주소 및 CodeCatalyst 별칭과 같은 정보를 얻을 수 있습니다. 프로필과 AWS 빌더 ID를 업데이트할 수도 있습니다. 비밀번호를 잊은 경우 비밀번호 재설정을 요청할 수 있습니다.

CodeCatalyst 프로필 보기

가입 시 CodeCatalyst Amazon에 로그인하기 위한 자격 증명으로 사용되고 프로필에서 관리되는 정보를 제공합니다. 여기에는 이름, 닉네임, 로그인할 때 사용하는 이메일 주소가 포함됩니다. CodeCatalyst

> **Note**
>
> AWS 빌더 ID 닉네임은 별칭이 아닙니다. CodeCatalyst 가입 시 CodeCatalyst 별칭을 선택했습니다.

프로필을 보려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 오른쪽 상단에서 첫 이니셜이 있는 아이콘 옆의 화살표를 선택한 다음 내 설정을 선택합니다. CodeCatalyst 내 설정 페이지가 열립니다.
3. AWS 빌더 ID 이메일 주소 또는 비밀번호를 업데이트하거나 MFA를 설정하려면 AWS 빌더 ID 관리를 선택합니다. AWS 빌더 ID 페이지가 열립니다.

다른 사용자의 CodeCatalyst 프로필 보기

다른 사용자의 CodeCatalyst 프로필을 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 측면 탐색에서 프로젝트 설정을 선택합니다. 구성원 탭을 선택합니다. CodeCatalyst프로젝트의 구성원 목록을 확인하세요.
3. 검색하려는 멤버 이름 또는 @mention 를 선택합니다. 내 설정 페이지에는 사용자의 별칭, 이메일 주소, 전체 이름이 표시됩니다. 이 CodeCatalyst 별칭을 @mention 프로젝트 멤버에게 사용하세요.

> **Note**
>
> 사용자의 AWS 빌더 ID 닉네임은 사용자의 별칭이 아닙니다. CodeCatalyst 가입할 때 CodeCatalyst 별칭을 선택했습니다.

프로젝트에 있는 다른 사용자의 프로필을 보려면 목록에서 해당 사용자의 이름을 선택하세요.

프로필 업데이트

에서 CodeCatalyst 프로필은 Builder ID에서 관리하는 개인 정보와 AWS Builder ID에서 관리하는 설정으로 구성됩니다 CodeCatalyst.

- 프로필의 전체 이름, 이메일 주소 및 비밀번호는 AWS Builder ID로 관리됩니다. 가입할 때 이 정보를 입력했습니다. 애플리케이션 로그인에 인증 앱을 사용하도록 MFA를 설정하면 빌더 ID 페이지로 CodeCatalyst 이동합니다.AWS
- CodeCatalyst 개인 액세스 토큰 (PAT), CodeCatalyst 알림 및 언어 기본 설정에 대한 설정은 의 내 설정 페이지에서 관리됩니다. CodeCatalyst 자세한 정보는 개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여을 참조하세요.

> **Note**
>
> AWS 빌더 ID 전체 이름 (CodeCatalyst 표시 이름) 과 이름을 업데이트할 수 있습니다. 하지만 CodeCatalyst 별칭은 변경할 수 없습니다.

AWS 빌더 ID 또는 이메일 주소 업데이트

사용자 AWS Builder ID 또는 이메일 주소를 업데이트하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 오른쪽 상단에서 첫 이니셜이 있는 아이콘 옆의 화살표를 선택한 다음 내 설정을 선택합니다. CodeCatalyst내 설정 페이지가 열립니다.
3. 프로필 페이지에서 AWS 빌더 ID 관리를 선택합니다. AWS 빌더 ID 페이지가 열립니다.
4. 페이지 왼쪽에서 내 세부 정보를 선택합니다.
5. 프로필 정보에서 편집을 선택하여 이름이나 닉네임을 업데이트합니다. 닉네임을 지정하지 않은 경우 닉네임 필드에 전체 이름의 첫 번째 이름이 반영됩니다. 별칭이 아닙니다. CodeCatalyst

> **Note**
>
> 그러면 AWS 빌더 ID 전체 이름 및 이름이 업데이트됩니다. 이렇게 해도 CodeCatalyst 별칭은 업데이트되지 않습니다.

연락처 정보에서 편집을 선택하여 이메일 주소를 업데이트합니다.

> **Note**
>
> 이렇게 하면 로그인하는 데 사용할 이메일 주소가 CodeCatalyst 업데이트됩니다.

CodeCatalyst 비밀번호 변경

다음 지침에 따라 AWS 빌더 ID와 연결된 Amazon CodeCatalyst 비밀번호를 변경하십시오.

> **Note**
>
> SSO를 사용하여 로그인하는 경우 관리자에게 비밀번호 변경에 대해 문의하세요. CodeCatalyst

비밀번호를 CodeCatalyst 변경하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 오른쪽 상단에서 첫 이니셜이 있는 아이콘 옆의 화살표를 선택한 다음 사용자 프로필을 선택합니다. CodeCatalyst 내 설정 페이지가 열립니다.
3. 프로필 페이지에서 AWS 빌더 ID 관리를 선택합니다. AWS 빌더 ID 페이지가 열립니다.
4. 페이지 왼쪽에서 보안을 선택합니다.
5. 비밀번호 변경을 선택하고 지침을 따릅니다.

AWS CLI와 함께 사용하기 위한 설정 CodeCatalyst

Amazon CodeCatalyst 콘솔에서는 대부분의 일상 작업을 수행할 수 있습니다. 하지만 개발 환경, 개인용 액세스 토큰 또는 이벤트 로그인으로 작업할 AWS CLI 때는 이를 설정하고 구성해야 할 수도 있습니다. CodeCatalyst 에서 사용하려면 먼저 프로필을 AWS CLI 설치하고 구성해야 합니다 CodeCatalyst.

다음을 AWS CLI 설정하려면 CodeCatalyst

1. AWS CLI의 최신 버전을 설치합니다. 이미 AWS CLI 설치된 버전이 있는 경우 최신 버전이고 에 대한 CodeCatalyst 명령이 포함되어 있는지 확인하고 필요한 경우 업데이트하십시오. 명령이 포함된 버전이 설치되어 있는지 확인하려면 CodeCatalyst 명령 프롬프트를 열고 다음 명령을 실행합니다.

   ```
   aws codecatalyst help
   ```

   CodeCatalyst 명령 목록이 표시되면 지원하는 버전이 있는 CodeCatalyst 것입니다. 명령이 인식되지 않는 경우 버전을 최신 버전으로 업데이트하십시오. AWS CLI 자세한 내용은 AWS Command Line Interface 사용 설명서의 최신 버전 설치 또는 업데이트를 참조하십시오. AWS CLI
2. 프로필이 없거나 특정 용도에 맞게 명명된 프로필을 사용하려는 경우 aws configure 명령을 실행하여 프로필을 만드십시오 CodeCatalyst. 특별히 사용할 이름이 지정된 프로필을 만드는 것이 좋지만 기본 프로필을 사용할 수도 있습니다. CodeCatalyst 자세한 내용은 구성 기본 사항을 참조하십시오.
3. 프로필 `config` 파일을 편집하여 다음과 CodeCatalyst 같이 연결 대상 섹션을 추가합니다. `config` 파일은 `~/.aws/config`(Linux 또는 macOS) 또는 `C:\Users\`*`USERNAME`*`\.aws\config`(Windows)에 저장됩니다.

   ```
   [profile codecatalyst]
   region = us-west-2
   sso_session = codecatalyst

   [sso-session codecatalyst]
   sso_region = us-east-1
   sso_start_url = https://view.awsapps.com/start
   sso_registration_scopes = codecatalyst:read_write
   ```
4. 파일을 저장합니다.

5. CodeCatalyst 명령을 실행하기 전에 새 터미널 또는 명령 프롬프트를 열고 다음 명령을 실행하여 명령 실행을 위한 자격 증명을 요청하고 검색하십시오`aws codecatalyst`. 필요한 경우 프로필 `codecatalyst` 이름으로 바꾸십시오.

```
aws sso login --profile codecatalyst
```

codecatalyst명령의 예를 보려면 다음 항목을 참조하십시오.

- 개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여
- 이벤트 로깅을 사용하여 로깅된 이벤트에 액세스

시작하기 튜토리얼

CodeCatalyst Amazon은 프로젝트를 시작하는 데 도움이 되는 다양한 템플릿을 제공합니다. 빈 프로젝트로 시작하고 리소스를 추가할 수도 있습니다. 이 튜토리얼의 단계를 따라 작업할 수 있는 몇 가지 방법을 알아보세요. CodeCatalyst

처음 사용하는 CodeCatalyst 경우 먼저 시작하는 튜토리얼: 모던 3계층 웹 애플리케이션 블루프린트로 프로젝트 생성 것이 좋습니다.

> **Note**
>
> 이 튜토리얼을 따르려면 먼저 설정을 완료해야 합니다. 자세한 정보는 설정 및 로그인 CodeCatalyst을 참조하세요.

주제

- 튜토리얼: 모던 3계층 웹 애플리케이션 블루프린트로 프로젝트 생성
- 튜토리얼: 빈 프로젝트로 시작하고 리소스를 수동으로 추가하기
- 튜토리얼: CodeCatalyst 제너레이티브 AI 기능을 사용하여 개발 작업의 속도를 높이세요
- 튜토리얼: 컴포저블 PDK 블루프린트로 풀스택 애플리케이션 만들기

의 특정 기능 영역에 초점을 맞춘 추가 자습서는 CodeCatalyst 다음을 참조하십시오.

- 슬랙 알림 시작하기
- CodeCatalyst 소스 리포지토리 및 단일 페이지 애플리케이션 청사진 시작하기
- 워크플로우 시작하기
- 커스텀 블루프린트 시작하기
- Amazon CodeCatalyst 액션 개발자 가이드로 시작하기

자세한 자습서는 다음을 참조하십시오.

- 자습서: Amazon S3에 아티팩트 업로드
- 자습서: 서버리스 애플리케이션 배포
- 자습서: Amazon에 애플리케이션 배포 ECS

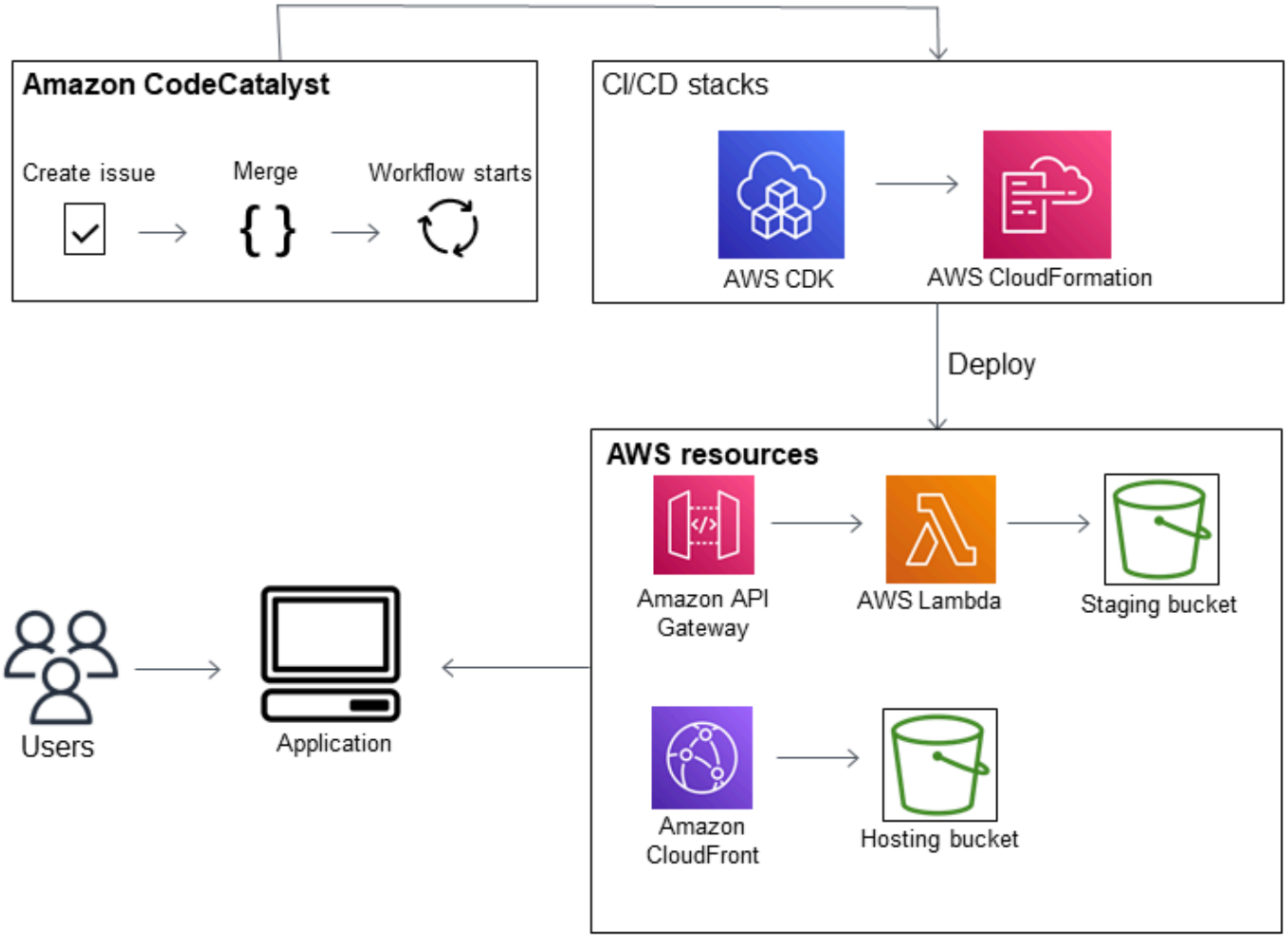
- 자습서: Amazon에 애플리케이션 배포 EKS
- 튜토리얼: 액션을 GitHub 사용한 린트 코드
- 튜토리얼: React 애플리케이션 생성 및 업데이트

튜토리얼: 모던 3계층 웹 애플리케이션 블루프린트로 프로젝트 생성

블루프린트가 포함된 프로젝트를 생성하여 소프트웨어 개발을 더 빠르게 시작할 수 있습니다. 블루프린트로 만든 프로젝트에는 코드를 관리하는 소스 리포지토리, 애플리케이션을 빌드하고 배포하기 위한 워크플로우 등 필요한 리소스가 포함되어 있습니다. 이 자습서에서는 Modern 3계층 웹 애플리케이션 블루프린트를 사용하여 Amazon에서 프로젝트를 생성하는 방법을 안내합니다. CodeCatalyst 이 자습서에는 배포된 샘플을 보고, 다른 사용자를 초대하여 작업하도록 하고, pull 요청이 병합될 AWS 계정 때 자동으로 빌드되어 연결된 리소스에 배포되는 pull 요청을 사용하여 코드를 변경하는 작업도 포함됩니다. 보고서, 활동 피드, 기타 도구를 사용하여 프로젝트를 생성하는 경우, 블루프린트는 AWS 계정 프로젝트와 관련된 AWS 리소스를 생성합니다. CodeCatalyst 블루프린트 파일을 사용하면 샘플 최신 애플리케이션을 빌드 및 테스트하고 의 인프라에 배포할 수 있습니다. AWS 클라우드

다음 그림은 의 CodeCatalyst 도구를 사용하여 이슈를 생성하여 변경 사항을 추적하고 병합하고 자동으로 빌드한 다음 인프라를 AWS CDK 허용하고 AWS CloudFormation 프로비저닝하는 작업을 실행하는 CodeCatalyst 프로젝트에서 워크플로를 시작하는 방법을 보여줍니다.

작업은 연결된 AWS 계정 리소스에서 리소스를 생성하고 API Gateway 엔드포인트가 있는 서버리스 AWS Lambda함수에 애플리케이션을 배포합니다. AWS Cloud Development Kit (AWS CDK) 작업은 하나 이상의 스택을 템플릿으로 변환하고 AWS CDK 스택을 AWS CloudFormation 템플릿에 배포합니다. AWS 계정스택의 리소스에는 동적 웹 콘텐츠를 배포하기 위한 Amazon CloudFront 리소스, 애플리케이션 데이터를 위한 Amazon DynamoDB 인스턴스, 배포된 애플리케이션을 지원하는 역할 및 정책이 포함됩니다.



모던 3계층 웹 애플리케이션 블루프린트로 프로젝트를 만들면 다음 리소스가 포함된 프로젝트가 생성됩니다.

프로젝트에서: CodeCatalyst

- 샘플 코드와 워크플로 YAML이 있는 소스 리포지토리
- 기본 브랜치가 변경될 때마다 샘플 코드를 빌드하고 배포하는 워크플로입니다.
- 작업을 계획하고 추적하는 데 사용할 수 있는 이슈 보드 및 백로그
- 샘플 코드에 자동 보고서가 포함된 테스트 보고서 세트

관련 항목 AWS 계정:

- 애플리케이션에 필요한 리소스를 생성하는 세 개의 AWS CloudFormation 스택.

이 자습서에서 AWS 또는 이 자습서의 CodeCatalyst 일부로 생성될 리소스에 대한 자세한 내용은 을 참조하십시오레퍼런스.

> **Note**
>
> 프로젝트에 포함되는 리소스와 샘플은 선택한 블루프린트에 따라 달라집니다. CodeCatalyst Amazon은 정의된 언어 또는 프레임워크와 관련된 리소스를 정의하는 여러 프로젝트 청사진을 제공합니다. 블루프린트에 대한 자세한 내용은 을 참조하십시오. CodeCatalyst 청사진이 포함된 포괄적인 프로젝트 만들기

주제

사전 조건

이 자습서에서 최신 응용 프로그램 프로젝트를 만들려면 다음과 설정 및 로그인 CodeCatalyst 같은 작업을 완료해야 합니다.

- 로그인하기 위한 AWS 빌더 ID가 있어야 CodeCatalyst 합니다.
- 스페이스에 속하고 해당 스페이스에서 스페이스 관리자 또는 파워 사용자 역할을 할당받아야 합니다. 자세한 내용은 스페이스 만들기, 사용자에게 공간 권한 부여, 스페이스 관리자 역할 단원을 참조하세요.

- AWS 계정 스페이스와 관련이 있고 가입 시 생성한 IAM 역할을 보유하세요. 예를 들어 가입 시 역할 정책이라는 역할 정책을 사용하여 서비스 역할을 생성하도록 선택할 수 있습니다. CodeCatalystWorkflowDevelopmentRole-***spaceName*** 역할에는 고유 식별자가 CodeCatalystWorkflowDevelopmentRole-*spaceName* 추가된 이름이 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오CodeCatalystWorkflowDevelopmentRole-***spaceName***서비스 역할 이해. 역할 생성 단계는 을 참조하십시오계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기.

1단계: 현대적인 3계층 웹 애플리케이션 프로젝트 만들기

프로젝트를 만든 후에는 코드를 개발 및 테스트하고, 개발 작업을 조정하고, 프로젝트 지표를 확인할 수 있습니다. 프로젝트에는 개발 도구와 리소스도 포함되어 있습니다.

이 자습서에서는 모던 3계층 웹 애플리케이션 블루프린트를 사용하여 대화형 애플리케이션을 만들어 보겠습니다. 프로젝트의 일부로 자동으로 생성되어 실행되는 워크플로우는 애플리케이션을 빌드하고 배포합니다. 워크플로는 스페이스에 대해 모든 역할 및 계정 정보를 구성한 후에만 정상적으로 실행됩니다. 워크플로가 성공적으로 실행되면 엔드포인트 URL로 이동하여 애플리케이션을 볼 수 있습니다.

블루프린트로 프로젝트를 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 프로젝트를 만들려는 스페이스로 이동합니다.
3. 프로젝트 만들기를 선택합니다.
4. '블루프린트로 시작하기'를 선택합니다.
5. 검색 창에 **modern**를 입력합니다.
6. 모던 3티어 웹 애플리케이션 블루프린트를 선택한 후 다음을 선택합니다.
7. 프로젝트 이름에 프로젝트 이름을 입력합니다. 예:

   MyExampleProject.

   > **Note**
   >
   > 이름은 스페이스에서 고유해야 합니다.

8. 계정에서 가입 시 추가한 AWS 계정 이름을 선택합니다. 블루프린트는 이 계정에 리소스를 설치합니다.

9. 배포 역할에서 가입 시 추가한 역할을 선택합니다. 예를 들어 `CodeCatalystWorkflowDevelopmentRole-spaceName`를 선택합니다.

   목록에 역할이 없는 경우 추가하세요. 역할을 추가하려면 Add IAM role (IAM role) 을 선택하고 역할을 자신의 AWS 계정역할에 추가합니다. 자세한 정보는 연결된 AWS 리소스에 대한 액세스 허용 AWS 계정을 참조하세요.
10. 컴퓨팅 플랫폼에서 Lambda를 선택합니다.
11. 프런트엔드 호스팅 옵션에서 Amplify 호스팅을 선택합니다. 에 대한 자세한 내용은 AWS Amplify호스팅이란 AWS Amplify 무엇입니까? 를 참조하십시오. AWS Amplify 사용 설명서에서
12. 배포 지역에 Mysfits 애플리케이션 및 지원 AWS 리전 리소스를 배포하기 위한 블루프린트를 구축할 지역의 지역 코드를 입력합니다. 지역 코드 목록은 의 지역 엔드포인트를 참조하십시오. AWS 일반 참조
13. 애플리케이션 이름에서 기본값을 로 유지합니다. `mysfitsstring`
14. (선택 사항) 프로젝트 미리 보기 생성에서 코드 보기를 선택하여 블루프린트가 설치할 소스 파일을 미리 봅니다. 워크플로우 보기를 선택하면 블루프린트가 설치할 CI/CD 워크플로 정의 파일을 미리 볼 수 있습니다. 미리보기는 선택에 따라 동적으로 업데이트됩니다.
15. 프로젝트 만들기를 선택합니다.

프로젝트 워크플로는 프로젝트를 만들자마자 시작됩니다. 코드 빌드 및 배포를 완료하는 데 시간이 조금 걸릴 수 있습니다. 그때까지는 다른 사람을 프로젝트에 초대하세요.

2단계: 프로젝트에 다른 사람 초대하기

이제 프로젝트를 설정했으니 함께 일할 다른 사람을 초대하세요.

프로젝트에 다른 사람 초대하기

1. 사용자를 초대하려는 프로젝트로 이동합니다.
2. 탐색 창에서 프로젝트 설정을 선택합니다.
3. 멤버 탭에서 초대를 선택합니다.
4. 프로젝트 사용자로 초대하려는 사람들의 이메일 주소를 입력합니다. 이메일 주소를 공백이나 쉼표로 구분하여 여러 개 입력할 수 있습니다. 프로젝트 구성원이 아닌 스페이스 구성원 중에서 선택할 수도 있습니다.
5. 사용자의 역할을 선택합니다.

   사용자 추가를 마치면 초대를 선택합니다.

3단계: 협업할 이슈 생성 및 작업 진행 상황 추적

CodeCatalyst 기능, 작업, 버그 및 문제가 있는 프로젝트와 관련된 기타 작업을 추적하는 데 도움이 됩니다. 이슈를 생성하여 필요한 작업과 아이디어를 추적할 수 있습니다. 기본적으로 이슈를 생성하면 백로그에 추가됩니다. 진행 중인 작업을 추적할 수 있는 게시판으로 이슈를 옮길 수 있습니다. 특정 프로젝트 멤버에게 이슈를 할당할 수도 있습니다.

프로젝트에 대한 이슈를 만들려면

1. 탐색 창에서 이슈를 선택합니다.
2. 이슈 생성을 선택합니다.
3. 이슈 제목에 이슈의 이름을 입력합니다. 문제에 대한 설명을 제공할 수도 있습니다. 이 예시에서는 다음을 사용합니다. **`make a change in the src/mysfit_data.json file.`**
4. 우선순위, 추정치, 상태, 라벨을 선택합니다. 담당자 아래에서 +Add me를 선택하여 자신에게 문제를 할당하세요.
5. 이슈 생성을 선택합니다. 이제 게시판에 이슈가 표시됩니다. 카드를 선택하여 이슈를 진행 중 열로 이동합니다.

자세한 정보는 에서 문제가 있는 작업을 추적하고 정리하세요. CodeCatalyst 을 참조하세요.

4단계: 소스 리포지토리 보기

블루프린트는 애플리케이션 또는 서비스를 정의하고 지원하는 파일이 포함된 소스 리포지토리를 설치합니다. 소스 리포지토리에서 주목할 만한 몇 가지 디렉터리 및 파일은 다음과 같습니다.

- .cloud9 디렉터리 — 개발 환경을 지원하는 파일이 들어 있습니다. AWS Cloud9
- .codecatalyst 디렉터리 - 블루프린트에 포함된 각 YAML 워크플로우에 대한 워크플로 정의 파일을 포함합니다.
- .idea 디렉터리 — 개발 환경을 위한 지원 파일이 들어 있습니다. JetBrains
- .vscode 디렉터리 — 비주얼 스튜디오 코드 개발 환경을 위한 지원 파일이 들어 있습니다.
- cdkStacks 디렉터리 - 의 인프라를 정의하는 AWS CDK 스택 파일이 들어 있습니다. AWS 클라우드
- src 디렉터리 - 애플리케이션 소스 코드가 들어 있습니다.
- 테스트 디렉터리 - 애플리케이션을 빌드하고 테스트할 때 실행되는 자동화된 CI/CD 워크플로의 일부로 실행되는 정수 및 단위 테스트용 파일을 포함합니다.
- 웹 디렉터리 - 프런트엔드 소스 코드가 들어 있습니다. 다른 파일에는 프로젝트에 대한 중요한 메타데이터가 포함된 `package.json` 파일, 웹 사이트 `index.html` 페이지, 린팅 코드용 파일, 루트

`.eslintrc.cjs` 파일 및 컴파일러 옵션을 지정하는 `tsconfig.json` 파일 등의 프로젝트 파일이 포함됩니다.

- `Dockerfile`파일 — 애플리케이션의 컨테이너를 설명합니다.
- `README.md`파일 — 프로젝트의 구성 정보가 들어 있습니다.

프로젝트의 소스 리포지토리로 이동하려면

1. 프로젝트로 이동하여 다음 중 하나를 수행하십시오.
   - 프로젝트 요약 페이지의 목록에서 원하는 리포지토리를 선택한 다음 리포지토리 보기를 선택합니다.
   - 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다. 소스 리포지토리의 목록에서 리포지토리 이름을 선택합니다. 필터 막대에 리포지토리 이름의 일부를 입력하여 리포지토리 목록을 필터링할 수 있습니다.
2. 리포지토리 홈 페이지에서 리포지토리의 콘텐츠와 관련 리소스에 대한 정보 (예: pull request 및 워크플로 수) 를 볼 수 있습니다. 기본적으로 기본 브랜치의 콘텐츠가 표시됩니다. 드롭다운 목록에서 다른 브랜치를 선택하여 보기를 변경할 수 있습니다.

5단계: 테스트 브랜치가 포함된 개발 환경 생성 및 빠른 코드 변경

개발 환경을 만들어 소스 리포지토리의 코드를 빠르게 작업할 수 있습니다. 이 자습서에서는 다음을 수행한다고 가정합니다.

- AWS Cloud9 개발 환경 만들기.
- 개발 환경을 만들 때 메인 브랜치 외부의 새 브랜치에서 작업하기 위한 옵션을 선택하세요.
- 이 새 브랜치의 이름을 `test` 사용하세요.

이후 단계에서는 개발 환경을 사용하여 코드를 변경하고 풀 리퀘스트를 생성할 것입니다.

새 브랜치를 사용하여 개발 환경을 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 생성하려는 스페이스로 이동합니다.
3. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택하고 소스 리포지토리를 선택한 다음 개발 환경을 만들려는 리포지토리를 선택합니다.
4. 리포지토리 홈 페이지에서 개발 환경 생성을 선택합니다.

5. 드롭다운 메뉴에서 지원되는 IDE를 선택합니다. 자세한 정보는 개발 환경을 위해 지원되는 통합 개발 환경을 참조하세요.
6. 복제할 리포지토리를 선택하고, 새 브랜치에서 작업을 선택하고, 브랜치 이름 필드에 브랜치 이름을 입력하고, 다음에서 브랜치 생성 드롭다운 메뉴에서 새 브랜치를 만들 브랜치를 선택합니다.
7. 원하는 경우 개발 환경의 별칭을 추가할 수 있습니다.
8. 선택적으로 개발 환경 구성 편집 버튼을 선택하여 개발 환경의 컴퓨팅, 스토리지 또는 제한 시간 구성을 편집합니다.
9. 생성을 선택합니다. 개발 환경이 생성되는 동안 개발 환경 상태 열에 시작 중이 표시되고, 개발 환경이 생성되면 상태 열에 실행 중이 표시됩니다. 선택한 IDE의 개발 환경과 함께 새 탭이 열립니다. 코드를 편집하고 변경 내용을 커밋 및 푸시할 수 있습니다.

이 섹션에서는 pull 요청이 병합될 AWS 계정 때 자동으로 빌드되어 연결된 리소스에 배포되는 pull 요청을 사용하여 코드를 변경하는 방식으로 생성된 샘플 애플리케이션을 사용하여 CodeCatalyst 작업해 봅니다.

파일을 변경하려면 **src/mysfit_data.json**

1. 프로젝트 개발 환경으로 이동합니다. 에서 AWS Cloud9측면 탐색 메뉴를 확장하여 파일을 찾아보십시오. 펼치고 `mysfitssrc`, 엽니다`src/mysfit_data.json`.
2. 파일에서 `"Age":` 필드 값을 6에서 12로 변경합니다. 줄은 다음과 같아야 합니다.

```
    {
        "Age": 12,
        "Description": "Twilight's personality sparkles like the night sky and is
 looking for a forever home with a Greek hero or God. While on the smaller side
 at 14 hands, he is quite adept at accepting riders and can fly to 15,000 feet.
 Twilight needs a large area to run around in and will need to be registered with
 the FAA if you plan to fly him above 500 feet. His favorite activities include
 playing with chimeras, going on epic adventures into battle, and playing with a
 large inflatable ball around the paddock. If you bring him home, he'll quickly
 become your favorite little Pegasus.",
        "GoodEvil": "Good",
        "LawChaos": "Lawful",
        "Name": "Twilight Glitter",
        "ProfileImageUri": "https://www.mythicalmysfits.com/images/
pegasus_hover.png",
        "Species": "Pegasus",
        "ThumbImageUri": "https://www.mythicalmysfits.com/images/pegasus_thumb.png"
```

```
    },
```

3. 파일을 저장합니다.
4. 명령을 사용하여 mysfits 저장소로 변경합니다. **`cd /projects/mysfits`**
5. git add, git commit, git push 명령을 사용하여 변경 내용을 추가하고 커밋하고 푸시할 수 있습니다.

```
git add .
git commit -m "make an example change"
git push
```

6단계: 최신 애플리케이션을 빌드하는 워크플로우 보기

최신 애플리케이션 프로젝트를 만든 후 워크플로를 포함하여 사용자를 대신하여 여러 리소스를 CodeCatalyst 생성합니다. 워크플로는.yaml 파일에 정의된 자동화된 프로시저로, 코드를 빌드, 테스트 및 배포하는 방법을 설명합니다.

이 자습서에서는 워크플로를 CodeCatalyst 만들고 프로젝트를 만들 때 워크플로를 자동으로 시작했습니다. (프로젝트를 만든 지 얼마나 되었는지에 따라 워크플로가 계속 실행 중일 수 있습니다.) 다음 절차를 사용하여 워크플로의 진행 상황을 확인하고, 생성된 로그와 테스트 보고서를 검토하고, 마지막으로 배포된 응용 프로그램의 URL을 탐색하십시오.

워크플로우 진행 상황을 확인하려면

1. CodeCatalyst 콘솔의 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.

   워크플로 목록이 나타납니다. 다음은 프로젝트를 만들 때 CodeCatalyst 블루프린트가 생성하고 시작한 워크플로입니다.
2. 워크플로 목록을 살펴보세요. 다음 네 가지가 보일 것입니다.

   - 맨 위에 있는 두 개의 워크플로는 이전에 만든 `test` 브랜치에 해당합니다5단계: 테스트 브랜치가 포함된 개발 환경 생성 및 빠른 코드 변경. 이러한 워크플로는 `main` 브랜치의 워크플로를 복제합니다. ApplicationDeploymentPipeline는 `main` 브랜치와 함께 사용하도록 구성되었으므로 활성화되지 않습니다. 풀 리퀘스트가 이루어지지 않아 OnPullRequest워크플로가 실행되지 않았습니다.
   - 맨 아래에 있는 두 개의 워크플로는 앞서 블루프린트를 실행할 때 만든 `main` 브랜치에 해당합니다. ApplicationDeploymentPipeline워크플로가 활성 상태이고 실행 중 (또는 완료) 상태입니다.

> **Note**
>
> Build @cdk_bootstrap 또는 DeployBackend오류와 함께 ApplicationDeploymentPipeline실행이 실패한다면, 이전에 모던 3계층 웹 애플리케이션을 실행했는데 기존 리소스가 남아 있어 현재 블루프린트와 충돌하기 때문일 수 있습니다. 이러한 이전 리소스를 삭제한 다음 워크플로를 다시 실행해야 합니다. 자세한 정보는 리소스 정리을 참조하세요.

3. 하단에서 `main` 브랜치와 관련된 ApplicationDeploymentPipeline워크플로를 선택합니다. 이 워크플로는 `main` 브랜치의 소스 코드를 사용하여 실행되었습니다.

   워크플로 다이어그램이 나타납니다. 다이어그램에는 각각 작업 또는 작업을 나타내는 여러 블록이 나와 있습니다. 대부분의 액션은 세로로 정렬되며 맨 위에 있는 액션이 아래 액션보다 먼저 정렬됩니다. 나란히 정렬된 액션은 병렬로 실행됩니다. 함께 그룹화된 액션이 모두 성공적으로 실행되어야 아래 액션을 시작할 수 있습니다.

   기본 블록은 다음과 같습니다.

   - WorkflowSource— 이 블록은 소스 리포지토리를 나타냅니다. 여기에는 다른 정보 중에서도 소스 리포지토리 이름 (mysfits) 과 워크플로우 실행을 자동으로 시작한 커밋이 표시됩니다. CodeCatalyst 프로젝트를 만들 때 이 커밋을 생성했습니다.
   - 빌드 — 이 블록은 다음 액션을 시작하려면 둘 다 성공적으로 완료해야 하는 두 액션을 그룹화한 것입니다.
   - DeployBackend— 이 블록은 애플리케이션의 백엔드 구성 요소를 클라우드에 배포하는 작업을 나타냅니다. AWS
   - 테스트 - 이 블록은 다음 작업을 시작하려면 둘 다 성공적으로 완료해야 하는 두 테스트 작업을 그룹화한 것입니다.
   - DeployFrontend— 이 블록은 애플리케이션의 프론트엔드 구성 요소를 클라우드에 배포하는 작업을 나타냅니다. AWS

4. 정의 탭 (상단 근처) 을 선택합니다. 워크플로우 정의 파일이 오른쪽에 나타납니다. 이 파일에는 다음과 같은 주목할 만한 섹션이 있습니다.

   - 맨 위에 있는 `Triggers` 섹션. 이 섹션은 코드가 소스 리포지토리의 `main` 브랜치에 푸시될 때마다 워크플로를 시작해야 함을 나타냅니다. 다른 브랜치 (예:`test`) 로 푸시해도 이 워크플로우는 시작되지 않습니다. 워크플로는 `main` 브랜치에 있는 파일을 사용하여 실행됩니다.

- `Actions`섹션, 아래`Triggers`. 이 섹션에서는 워크플로 다이어그램에 표시되는 작업을 정의합니다.

5. 상단 근처의 최신 상태 탭을 선택하고 워크플로 다이어그램에서 원하는 작업을 선택합니다.
6. 오른쪽에서 구성 탭을 선택하면 최근 실행 중에 작업에 사용된 구성 설정을 볼 수 있습니다. 각 구성 설정에는 워크플로 정의 파일에 일치하는 속성이 있습니다.
7. 콘솔을 열어 두고 다음 절차로 이동합니다.

빌드 로그와 테스트 보고서를 검토하려면

1. 최신 상태 탭을 선택합니다.
2. 워크플로 다이어그램에서 DeployFrontend작업을 선택합니다.
3. 작업이 완료될 때까지 기다리십시오. “진행 중” 아이콘
   ( )
   이 “성공” 아이콘
   ( )
   으로 바뀌는지 확인하십시오.
4. build_backend 작업을 선택합니다.
5. Logs 탭을 선택하고 두 개의 섹션을 펼쳐 이 단계에 대한 로그 메시지를 확인하세요. 백엔드 설정과 관련된 메시지를 볼 수 있습니다.
6. 보고서 탭을 선택한 다음 `backend-coverage.xml` 보고서를 선택합니다. CodeCatalyst 관련 보고서를 표시합니다. 보고서에는 실행된 코드 커버리지 테스트가 표시되며 테스트를 통해 성공적으로 검증된 코드 행의 비율 (예: 80%) 이 표시됩니다.

   테스트 보고서에 대한 자세한 내용은 을 참조하십시오. 워크플로를 사용한 테스트

   > **Tip**
   > 탐색 창에서 보고서를 선택하여 테스트 보고서를 볼 수도 있습니다.

7. CodeCatalyst 콘솔을 열어 두고 다음 절차로 이동합니다.

최신 응용 프로그램이 성공적으로 배포되었는지 확인합니다.

1. ApplicationDeploymentPipeline워크플로로 돌아가서 최근 실행의 Run- ***string*** 링크를 선택합니다.

2. 워크플로 다이어그램에서 DeployFrontend작업을 찾고 앱 보기 링크를 선택합니다. Mysfit 웹 사이트가 표시됩니다.

> **Note**
>
> DeployFrontend작업 내에서 앱 보기 링크가 보이지 않는 경우 실행 ID 링크를 선택했는지 확인하세요.

3. 트와일라잇 글리터라는 이름의 페가수스 마이스핏을 검색해 보세요. 나이의 가치를 기록해 두세요. 맞아요6. 코드를 변경하여 연령을 업데이트해야 합니다.

7단계: 다른 사람에게 변경 내용을 검토해 달라고 요청하세요.

이름이 지정된 `test` 브랜치에 변경 사항이 생겼으니 풀 리퀘스트를 생성하여 다른 사람에게 검토해 달라고 요청할 수 있습니다. 브랜치의 변경 내용을 `test` 브랜치에 병합하기 위한 풀 요청을 만들려면 다음 단계를 수행하십시오. `main`

풀 리퀘스트를 만들려면

1. 프로젝트로 이동합니다.
2. 다음 중 하나를 수행하십시오.
   - 탐색 창에서 코드를 선택하고 풀 요청을 선택한 다음 풀 요청 생성을 선택합니다.
   - 리포지토리 홈 페이지에서 추가를 선택한 다음 풀 리퀘스트 생성을 선택합니다.
   - 프로젝트 페이지에서 풀 리퀘스트 생성을 선택합니다.
3. 소스 리포지토리에서 지정된 소스 리포지토리가 커밋된 코드를 포함하는 리포지토리인지 확인합니다. 이 옵션은 리포지토리의 기본 페이지에서 풀 리퀘스트를 만들지 않은 경우에만 나타납니다.
4. 대상 브랜치에서 코드를 검토한 후 병합할 브랜치를 선택합니다.
5. 소스 브랜치에서 커밋된 코드가 들어 있는 브랜치를 선택합니다.
6. 풀 리퀘스트 제목에 다른 사용자가 검토해야 할 내용과 이유를 이해하는 데 도움이 되는 제목을 입력합니다.
7. (선택 사항) 풀 요청 설명에 문제 링크 또는 변경 내용 설명과 같은 정보를 입력합니다.

> **Tip**
>
> Pull Request에 포함된 변경 사항에 대한 설명을 CodeCatalyst 자동으로 생성하도록 나를 위한 설명 쓰기를 선택할 수 있습니다. 자동으로 생성된 설명을 풀 리퀘스트에 추가한 후 변경할 수 있습니다.
> 이 기능을 사용하려면 스페이스에 대해 제너레이티브 AI 기능을 활성화해야 하며 연결된 리포지토리의 풀 요청에는 사용할 수 없습니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.

8. (선택 사항) 이슈에서 이슈 연결을 선택한 다음 목록에서 이슈를 선택하거나 해당 ID를 입력합니다. 이슈의 연결을 해제하려면 연결 해제 아이콘을 선택합니다.
9. (선택 사항) 필수 검토자에서 필수 검토자 추가를 선택합니다. 프로젝트 멤버 목록에서 선택하여 추가합니다. 풀 리퀘스트를 대상 브랜치에 병합하려면 먼저 필수 검토자가 변경 사항을 승인해야 합니다.

> **Note**
>
> 검토자를 필수 검토자와 선택적 검토자로 모두 추가할 수는 없습니다. 자신을 리뷰어로 추가할 수 없습니다.

10. (선택 사항) 선택적 검토자에서 선택적 검토자 추가를 선택합니다. 프로젝트 멤버 목록에서 선택하여 추가합니다. 선택적 검토자는 풀 리퀘스트를 대상 브랜치에 병합하기 전에 변경 사항을 요구 사항으로 승인할 필요가 없습니다.
11. 브랜치 간의 차이점을 검토하세요. 풀 리퀘스트에 표시되는 차이는 소스 브랜치의 수정 버전과 풀 리퀘스트가 생성된 시점의 대상 브랜치의 헤드 커밋인 병합 베이스 사이의 변경 내용입니다. 변경 사항이 표시되지 않으면 브랜치가 동일하거나 소스와 대상 모두에 대해 동일한 브랜치를 선택했을 수 있습니다.
12. 풀 리퀘스트에 검토하려는 코드와 변경 사항이 포함되어 있다고 판단되면 [Create] 를 선택합니다.

> **Note**
>
> 풀 리퀘스트를 생성한 후 코멘트를 추가할 수 있습니다. 풀 리퀘스트나 파일의 개별 라인, 전체 풀 리퀘스트에 댓글을 추가할 수 있습니다. @ 기호 뒤에 파일 이름을 붙여 파일 등의 리소스에 링크를 추가할 수 있습니다.

풀 리퀘스트를 만들면 `test` 브랜치의 소스 파일을 사용하여 OnPullRequest워크플로가 시작됩니다. 검토자가 코드 변경을 승인하는 동안 워크플로를 선택하고 테스트 결과를 보면 결과를 확인할 수 있습니다.

변경 사항을 검토한 후 코드를 병합할 수 있습니다. 코드를 기본 브랜치에 병합하면 변경 내용을 빌드하고 배포하는 워크플로가 자동으로 시작됩니다.

콘솔에서 풀 리퀘스트를 병합하려면 CodeCatalyst

1. 최신 애플리케이션 프로젝트로 이동하세요.
2. 프로젝트 페이지의 오픈 풀 리퀘스트에서 병합하려는 풀 리퀘스트를 선택합니다. 풀 리퀘스트가 보이지 않는 경우 모두 보기를 선택한 다음 목록에서 선택하십시오. 병합을 선택합니다.
3. 풀 리퀘스트에 사용할 수 있는 병합 전략 중에서 선택하십시오. 선택적으로 풀 리퀘스트를 병합한 후 소스 브랜치를 삭제하는 옵션을 선택하거나 선택 취소한 다음 병합을 선택합니다.

> **Note**
>
> 병합 버튼이 활성화되지 않았거나 병합할 수 없음 레이블이 표시되는 경우, 한 명 이상의 필수 검토자가 아직 풀 요청을 승인하지 않았거나 콘솔에서 풀 요청을 병합할 수 없습니다. CodeCatalyst 풀 리퀘스트를 승인하지 않은 검토자는 풀 리퀘스트 세부 정보 영역의 개요에 있는 시계 아이콘으로 표시됩니다. 필요한 모든 검토자가 풀 요청을 승인했지만 병합 버튼이 여전히 활성화되지 않은 경우 병합 충돌이 발생할 수 있습니다. CodeCatalyst콘솔에서 대상 브랜치의 병합 충돌을 해결한 다음 풀 요청을 병합하거나, 충돌을 해결하고 로컬에서 병합한 다음 병합이 포함된 커밋을 푸시할 CodeCatalyst 수 있습니다. 자세한 내용은 풀 리퀘스트 병합 (Git) 및 Git 설명서를 참조하십시오.

브랜치의 변경 내용을 `test` 브랜치에 병합하면 변경 내용을 빌드하고 배포하는 ApplicationDeploymentPipeline워크플로가 자동으로 시작됩니다. **`main`**

병합된 커밋을 보려면 워크플로를 통해 실행하세요. ApplicationDeploymentPipeline

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 워크플로의 에서 ApplicationDeploymentPipeline최근 실행을 확장합니다. 병합 커밋으로 워크플로 실행이 시작된 것을 볼 수 있습니다. 선택적으로 선택하여 실행 진행 상황을 확인할 수도 있습니다.
3. 실행이 완료되면 이전에 방문한 URL을 다시 로드합니다. 페가수스를 보고 나이가 바뀌었는지 확인하세요.

8단계: 이슈 종료

문제가 해결되면 CodeCatalyst 콘솔에서 해당 문제를 종료할 수 있습니다.

프로젝트 이슈를 종료하려면

1. 프로젝트로 이동합니다.
2. 탐색 창에서 이슈를 선택합니다.
3. rag-and-drop 이슈를 완료 열에 추가합니다.

자세한 정보는 에서 문제가 있는 작업을 추적하고 정리하세요. CodeCatalyst 을 참조하세요.

리소스 정리

사용자 환경에서 이 자습서의 흔적을 CodeCatalyst 정리하고 AWS 제거하세요.

이 자습서에서 사용한 프로젝트를 계속 사용하거나 프로젝트 및 관련 리소스를 삭제할 수 있습니다.

> **Note**
>
> 이 프로젝트를 삭제하면 프로젝트에 있는 모든 멤버의 모든 리포지토리, 이슈, 아티팩트가 모두 삭제됩니다.

프로젝트를 삭제하려면 다음과 같이 하세요.

1. 프로젝트로 이동한 다음 프로젝트 설정을 선택합니다.
2. 일반 탭을 선택합니다.
3. 프로젝트 이름 아래에서 프로젝트 삭제를 선택합니다.

AWS CloudFormation 및 Amazon S3에서 리소스를 삭제하려면

1. CodeCatalyst스페이스에 추가한 것과 동일한 AWS Management Console 계정으로 로그인합니다.
2. AWS CloudFormation서비스로 이동하세요.
3. mysfits ***###*** 스택을 삭제합니다.
4. 개발-mysfits 문자열 스택을 ***#####.***
5. CDKToolkit 스택을 선택하십시오 (삭제하지는 마십시오). 리소스 탭을 선택합니다. StagingBucket링크를 선택하고 Amazon S3에서 버킷과 버킷 콘텐츠를 삭제합니다.

   > **Note**
   >
   > 이 버킷을 수동으로 삭제하지 않으면 모던 3계층 웹 애플리케이션 블루프린트를 다시 실행할 때 오류가 발생할 수 있습니다.

6. (선택 사항) CDKToolkit 스택을 삭제합니다.

레퍼런스

최신 3계층 웹 애플리케이션 블루프린트는 리소스를 클라우드의 CodeCatalyst 공간과 AWS 계정에 배포합니다. AWS 이러한 리소스는 다음과 같습니다.

- CodeCatalyst 숙소 내:
  - 다음 리소스가 포함된 CodeCatalyst 프로젝트:

  - 소스 리포지토리 — 이 리포지토리에는 'Mysfits' 웹 애플리케이션의 샘플 코드가 들어 있습니다.
  - 워크플로 — 이 워크플로는 기본 브랜치가 변경될 때마다 Mysfits 애플리케이션 코드를 빌드하고 배포합니다.
  - 이슈 보드 및 백로그 — 이 게시판과 백로그를 사용하여 작업을 계획하고 추적할 수 있습니다.
  - 테스트 보고서 세트 — 이 제품군에는 샘플 코드에 포함된 자동 보고서가 포함되어 있습니다.
- 관련 항목 AWS 계정:
  - CDKToolkit 스택 — 이 스택은 다음 리소스를 배포합니다.
    - Amazon S3 스테이징 버킷, 버킷 정책 및 버킷을 암호화하는 데 사용되는 AWS KMS 키.
    - 배포 작업을 위한 IAM 배포 역할.
    - AWS 스택의 리소스를 지원하는 IAM 역할 및 정책

    > **Note**
    >
    > CDKToolkit은 배포할 때마다 해체되거나 다시 생성되지 않습니다. 이 스택은 지원을 위해 각 계정에서 시작되는 스택입니다. AWS CDK
  - 개발-mysfits ### BackEnd 스택 — 이 스택은 다음과 같은 백엔드 리소스를 배포합니다.
    - 아마존 API 게이트웨이 엔드포인트.
    - AWS 스택의 리소스를 지원하는 IAM 역할 및 정책.
    - AWS Lambda 함수와 계층은 최신 애플리케이션을 위한 서버리스 컴퓨팅 플랫폼을 제공합니다.
    - 버킷 배포 및 Lambda 함수를 위한 IAM 정책 및 역할.
  - mysfit ### 스택 — 이 스택은 프런트엔드 애플리케이션을 배포합니다. AWS Amplify

다음 사항도 참조하세요.

이 자습서의 일부로 리소스가 생성되는 AWS 서비스에 대한 자세한 내용은 다음을 참조하십시오.

- Amazon S3 — 업계 최고의 확장성, 데이터 고가용성, 보안 및 성능을 제공하는 객체 스토리지 서비스에 프런트엔드 자산을 저장하는 서비스입니다. 자세한 내용은 Amazon S3 사용 설명서를 참조하십시오.
- Amazon API Gateway — 모든 규모에서 REST, HTTP 및 WebSocket API를 생성, 게시, 유지 관리, 모니터링 및 보호하는 서비스입니다. 자세한 내용은 API Gateway 개발자 안내서를 참조하십시오.
- Amplify — 프론트엔드 애플리케이션을 호스팅하는 서비스입니다. 자세한 내용은 AWS Amplify 호스팅 사용 설명서를 참조하십시오.

- AWS Cloud Development Kit (AWS CDK)— 코드로 클라우드 인프라를 정의하고 이를 통해 AWS CloudFormation프로비저닝하기 위한 프레임워크입니다. AWS CDK 여기에는 AWS CDK 앱 및 스택과 상호 작용하기 위한 명령줄 도구인 AWS CDK 툴킷이 포함됩니다. 자세한 정보는 AWS Cloud Development Kit (AWS CDK) 개발자 안내서를 참조하십시오.
- Amazon DynamoDB — 데이터 저장을 위한 완전 관리형 NoSQL 데이터베이스 서비스입니다. 자세한 내용은 Amazon DynamoDB 개발자 안내서를 참조하십시오.
- AWS Lambda— 서버를 프로비저닝하거나 관리하지 않고도 고가용성 컴퓨팅 인프라에서 코드를 호출하는 서비스입니다. 자세한 정보는 AWS Lambda 개발자 안내서를 참조하십시오.
- AWS IAM — 해당 리소스에 대한 액세스를 안전하게 제어하기 위한 서비스입니다. AWS 자세한 내용은 IAM 사용 설명서를 참조하세요.

튜토리얼: 빈 프로젝트로 시작하고 리소스를 수동으로 추가하기

프로젝트를 만들 때 빈 프로젝트 블루프린트를 선택하면 사전 정의된 리소스 없이 빈 프로젝트를 만들 수 있습니다. 빈 프로젝트를 만든 후 프로젝트 요구 사항에 따라 리소스를 만들고 추가할 수 있습니다. 블루프린트 없이 만든 프로젝트는 생성 시 비어 있기 때문에 이 옵션을 시작하려면 CodeCatalyst 리소스 생성 및 구성에 대한 더 많은 지식이 필요합니다.

주제

사전 조건

빈 프로젝트를 만들려면 스페이스 관리자 또는 Power 사용자 역할이 할당되어야 합니다. 처음 로그인하는 CodeCatalyst 경우 을 참조하십시오설정 및 로그인 CodeCatalyst.

빈 프로젝트 만들기

프로젝트를 만드는 것은 함께 작업할 수 있는 첫 번째 단계입니다. 소스 리포지토리 및 워크플로와 같은 자체 리소스를 만들려면 빈 프로젝트로 시작할 수 있습니다.

빈 프로젝트를 만들려면

1. 프로젝트를 생성하려는 스페이스로 이동합니다.
2. 스페이스 대시보드에서 프로젝트 생성을 선택합니다.
3. 처음부터 시작을 선택합니다.
4. 프로젝트에 이름 부여에서 프로젝트에 할당할 이름을 입력합니다. 이름은 스페이스 내에서 고유해야 합니다.
5. 프로젝트 만들기를 선택합니다.

이제 빈 프로젝트가 생겼으니 다음 단계는 소스 리포지토리를 만드는 것입니다.

소스 리포지토리 만들기

프로젝트 코드를 저장하고 협업할 소스 리포지토리를 만드세요. 프로젝트 멤버는 이 리포지토리를 로컬 컴퓨터에 복제하여 코드 작업을 할 수 있습니다. 또는 지원되는 서비스에 호스팅된 리포지토리를 연결하도록 선택할 수 있지만 이 자습서에서는 다루지 않습니다. 자세한 정보는 소스 리포지토리 연결을 참조하세요.

소스 리포지토리를 생성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다.
3. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
4. 리포지토리 추가를 선택하고 리포지토리 생성을 선택합니다.
5. 리포지토리 이름에 리포지토리 이름을 제공합니다. 이 안내서에서는 이름을 *codecatalyst-source-repository*사용하지만 다른 이름을 선택할 수 있습니다. 리포지토리 이름은 프로젝트에서 고유해야 합니다. 리포지토리 이름 요구 사항에 대한 자세한 내용은 을 참조하십시오의 소스 리포지토리 할당량 CodeCatalyst.
6. (선택 사항) 설명에 리포지토리의 용도를 프로젝트의 다른 사용자가 이해하는 데 도움이 되도록 리포지토리에 대한 설명을 추가합니다.
7. 리포지토리 만들기 (기본값) 를 선택합니다. 이 옵션은 기본 브랜치와 README.md 파일을 포함하는 리포지토리를 생성합니다. 빈 리포지토리와 달리 이 리포지토리는 생성되자마자 사용할 수 있습니다.
8. Default 브랜치에서는 다른 이름을 선택할 이유가 없는 한 이름을 main으로 그대로 두십시오. 이 가이드의 예시에서는 모두 기본 브랜치에 main이라는 이름을 사용합니다.

9. (선택 사항) 푸시하려는 코드 유형에 맞는 `.gitignore` 파일을 추가합니다.
10. 생성을 선택합니다.

> **Note**
>
> CodeCatalyst `README.md`파일을 만들 때 저장소에 파일을 추가합니다. CodeCatalyst또한 main이라는 기본 브랜치에 리포지토리에 대한 초기 커밋을 생성합니다. Readme.md 파일을 편집하거나 삭제할 수 있지만 기본 브랜치는 삭제할 수 없습니다.

개발 환경을 만들어 리포지토리에 코드를 빠르게 추가할 수 있습니다. 이 자습서에서는 를 사용하여 AWS Cloud9개발 환경을 만들고 개발 환경을 만들 때 기본 브랜치에서 브랜치를 만드는 옵션을 선택하는 것이 좋습니다. 이 브랜치의 이름을 **`test`** 사용하지만 원하는 경우 다른 브랜치 이름을 입력할 수 있습니다.

새 브랜치를 사용하여 개발 환경을 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 생성하려는 스페이스로 이동합니다.
3. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택하고 소스 리포지토리를 선택한 다음 개발 환경을 만들려는 리포지토리를 선택합니다.
4. 리포지토리 홈 페이지에서 개발 환경 생성을 선택합니다.
5. 드롭다운 메뉴에서 지원되는 IDE를 선택합니다. 자세한 정보는 개발 환경을 위해 지원되는 통합 개발 환경을 참조하세요.
6. 복제할 리포지토리를 선택하고, 새 브랜치에서 작업을 선택하고, 브랜치 이름 필드에 브랜치 이름을 입력하고, 다음에서 브랜치 생성 드롭다운 메뉴에서 새 브랜치를 만들 브랜치를 선택합니다.
7. 원하는 경우 개발 환경의 별칭을 추가할 수 있습니다.
8. 선택적으로 개발 환경 구성 편집 버튼을 선택하여 개발 환경의 컴퓨팅, 스토리지 또는 제한 시간 구성을 편집합니다.
9. 생성을 선택합니다. 개발 환경이 생성되는 동안 개발 환경 상태 열에 시작 중이 표시되고, 개발 환경이 생성되면 상태 열에 실행 중이 표시됩니다. 선택한 IDE의 개발 환경과 함께 새 탭이 열립니다. 코드를 편집하고 변경 내용을 커밋 및 푸시할 수 있습니다.

워크플로를 만들어 코드 변경 사항을 빌드, 테스트, 배포하세요.

에서는 CodeCatalyst 애플리케이션 또는 서비스의 빌드, 테스트 및 배포를 워크플로우로 구성합니다. 워크플로는 작업으로 구성되며 코드 푸시나 풀 요청 열기 또는 업데이트와 같은 지정된 소스 리포지토리 이벤트가 발생한 후 자동으로 실행되도록 구성할 수 있습니다. 워크플로에 대한 자세한 내용은 워크플로를 통한 빌드, 테스트, 배포 섹션을 참조하세요.

안내에 따라 첫 번째 워크플로우 시작하기 워크플로를 생성하십시오.

프로젝트에 누군가를 초대하세요

이제 사용자 지정 프로젝트를 설정했으니 함께 작업할 다른 사람을 초대하세요.

프로젝트에 다른 사람 초대하기

1. 사용자를 초대하려는 프로젝트로 이동합니다.
2. 탐색 창에서 프로젝트 설정을 선택합니다.
3. 멤버 탭에서 초대를 선택합니다.
4. 프로젝트 사용자로 초대하려는 사람들의 이메일 주소를 입력합니다. 이메일 주소를 공백이나 쉼표로 구분하여 여러 개 입력할 수 있습니다. 프로젝트 구성원이 아닌 스페이스 구성원 중에서 선택할 수도 있습니다.
5. 사용자의 역할을 선택합니다.

   사용자 추가를 완료하면 Invite (초대) 를 선택합니다.

공동 작업하고 작업을 추적할 이슈를 생성하세요.

CodeCatalyst 기능, 작업, 버그 및 문제가 있는 프로젝트와 관련된 기타 작업을 추적하는 데 도움이 됩니다. 이슈를 생성하여 필요한 작업과 아이디어를 추적할 수 있습니다. 기본적으로 이슈를 생성하면 백로그에 추가됩니다. 진행 중인 작업을 추적할 수 있는 게시판으로 이슈를 옮길 수 있습니다. 특정 프로젝트 멤버에게 이슈를 할당할 수도 있습니다.

프로젝트에 대한 이슈를 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.

   이슈를 만들려는 프로젝트를 탐색하고 있는지 확인하세요. 모든 프로젝트를 보려면 탐색 창에서 CodeCatalystAmazon을 선택하고 필요한 경우 모든 프로젝트 보기를 선택합니다. 생성하거나 이슈를 해결하려는 프로젝트를 선택합니다.

2. 탐색 창에서 [Track] 을 선택한 다음 [Backlog] 를 선택합니다.
3. 이슈 생성을 선택합니다.
4. 이슈 제목에 이슈의 이름을 입력합니다. 필요에 따라 문제에 대한 설명을 제공할 수 있습니다. 원하는 경우 문제의 상태, 우선 순위 및 예상치를 선택합니다. 프로젝트 멤버 목록에서 프로젝트 멤버에게 이슈를 할당할 수도 있습니다.

> **Tip**
>
> Amazon Q에 문제를 할당하여 Amazon Q에서 문제 해결을 시도하도록 할 수 있습니다. 시도가 성공하면 풀 리퀘스트가 생성되고 문제 상태가 검토 중으로 변경되므로 코드를 검토하고 테스트할 수 있습니다. 자세한 정보는 튜토리얼: CodeCatalyst 제너레이티브 AI 기능을 사용하여 개발 작업의 속도를 높이세요을 참조하세요.
> 이 기능을 사용하려면 공간에 제너레이티브 AI 기능을 활성화해야 합니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.

5. 저장을 선택합니다.

이슈를 생성한 후에는 프로젝트 멤버에게 할당하고, 예측하고, 칸반 보드에서 추적할 수 있습니다. 자세한 내용은 에서 문제가 있는 작업을 추적하고 정리하세요. CodeCatalyst 을(를) 참조하세요.

튜토리얼: CodeCatalyst 제너레이티브 AI 기능을 사용하여 개발 작업의 속도를 높이세요

CodeCatalyst Amazon에 제너레이티브 AI 기능이 활성화된 공간에 프로젝트와 소스 리포지토리가 있는 경우 이러한 기능을 사용하여 소프트웨어 개발 속도를 높일 수 있습니다. 개발자가 수행하는 데 걸리는 시간보다 해야 할 작업이 더 많은 경우가 많습니다. 변경 사항 검토를 위한 풀 리퀘스트를 만들 때 팀원에게 코드 변경 내용을 설명하는 데 시간을 할애하지 않는 경우가 많습니다. 다른 사용자가 변경 내용을 스스로 설명할 수 있을 것이라고 기대하기 때문입니다. 또한 풀 리퀘스트 생성자와 리뷰어는 풀 리퀘스트의 모든 코멘트를 자세히 찾아서 읽을 시간이 없습니다. 특히 풀 리퀘스트에 여러 개의 수정 사항이 있는 경우에는 더욱 그렇습니다. CodeCatalyst 소프트웨어 개발용 Amazon Q Developer Agent와 통합하여 팀 구성원이 작업을 더 빠르게 완료하고 업무의 가장 중요한 부분에 집중하는 데 걸리는 시간을 늘릴 수 있는 제너레이티브 AI 기능을 제공합니다.

Amazon Q Developer는 애플리케이션을 이해, 구축, 확장 및 운영하는 데 도움이 되는 생성형 AI 기반 대화형 도우미입니다. AWS 개발을 가속화하기 위해 Amazon Q를 지원하는 모델을 고품질 AWS 콘텐츠로 보강하여 보다 완전하고 실행 가능하며 참조할 수 있는 답변을 제공합니다. AWS자세한 내용은

Amazon Q 개발자란 무엇입니까? 를 참조하십시오. Amazon Q 개발자 사용 설명서에서 확인할 수 있습니다.

> **Note**
>
> Amazon Bedrock 제공: 자동 악용 탐지 AWS 기능을 구현합니다. 나를 위한 설명 작성, 콘텐츠 요약 생성, 작업 추천, Amazon Q를 사용하여 프로젝트에 기능 생성 또는 추가, Amazon Q Developer Agent의 소프트웨어 개발 기능에 대한 Amazon Q 이슈 할당 기능이 Amazon Bedrock에 구축되어 있기 때문에 사용자는 Amazon Bedrock에 구현된 제어 기능을 최대한 활용하여 안전, 보안 및 인공 지능 (AI) 의 책임 있는 사용을 강화할 수 있습니다.

이 튜토리얼에서는 의 제너레이티브 AI 기능을 사용하여 블루프린트로 프로젝트를 생성하고 기존 프로젝트에 블루프린트를 추가하는 방법을 알아봅니다. CodeCatalyst 또한 풀 리퀘스트를 만들 때 브랜치 간 변경 사항을 요약하는 방법과 풀 리퀘스트에 남긴 댓글을 요약하는 방법을 배우게 됩니다. 코드 변경 또는 개선에 대한 아이디어로 이슈를 생성하여 Amazon Q에 할당하는 방법도 배우게 됩니다. Amazon Q에 할당된 이슈 관련 작업의 일환으로 Amazon Q가 작업을 제안하도록 허용하는 방법과 문제 해결의 일환으로 생성한 작업을 할당하고 처리하는 방법을 배우게 됩니다.

주제

사전 조건

이 자습서의 CodeCatalyst 기능을 사용하려면 먼저 작업을 완료하고 다음 리소스에 액세스할 수 있어야 합니다.

- 로그인하기 위한 AWS 빌더 ID 또는 싱글 사인온 (SSO) ID가 있어야 합니다. CodeCatalyst
- 제너레이티브 AI 기능이 활성화된 공간에 있습니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.

- 해당 스페이스의 프로젝트에서 기여자 또는 프로젝트 관리자 역할을 맡을 수 있습니다.
- 제너레이티브 AI로 프로젝트를 만들지 않는 한, 기존 프로젝트에는 이를 위한 소스 리포지토리가 하나 이상 구성되어 있습니다. 연결된 리포지토리는 지원되지 않습니다.
- 제너레이티브 AI로 만든 초기 솔루션으로 이슈를 할당하는 경우 Jira Software 확장 프로그램으로 프로젝트를 구성할 수 없습니다. 이 기능에는 확장 프로그램이 지원되지 않습니다.

자세한 내용은 스페이스 만들기, 에서 문제가 있는 작업을 추적하고 정리하세요. CodeCatalyst , 확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst 및 사용자 역할을 통한 액세스 권한 부여 부분을 참조하세요.

이 튜토리얼은 Python을 사용한 모던 3계층 웹 애플리케이션 블루프린트를 사용하여 만든 프로젝트를 기반으로 합니다. 다른 블루프린트로 만든 프로젝트를 사용하는 경우에도 단계를 따를 수는 있지만 샘플 코드 및 언어와 같은 일부 세부 사항은 달라질 수 있습니다.

프로젝트를 생성하거나 기능을 추가할 때 Amazon Q를 사용하여 청사진을 선택합니다.

프로젝트 개발자는 새 프로젝트를 만들거나 기존 프로젝트에 구성 요소를 추가할 때 제너레이티브 AI 어시스턴트인 Amazon Q와 협업할 수 있습니다. 채팅과 유사한 인터페이스로 Amazon Q와 상호 작용하여 프로젝트에 대한 요구 사항을 Amazon Q에 제공할 수 있습니다. Amazon Q는 요구 사항에 따라 청사진을 제안하고 충족할 수 없는 요구 사항도 설명합니다. 공간에 사용자 지정 청사진이 있는 경우 Amazon Q는 학습하여 해당 청사진도 권장 사항에 포함합니다. 그런 다음 만족하면 Amazon Q의 제안을 진행할 수 있습니다. 그러면 요구 사항에 맞는 코드가 포함된 소스 리포지토리와 같은 필요한 리소스가 생성됩니다. Amazon Q는 또한 청사진으로는 충족할 수 없는 요구 사항에 대한 문제를 생성합니다. 사용 가능한 CodeCatalyst 블루프린트에 대한 자세한 내용은 을 참조하십시오. CodeCatalyst 청사진이 포함된 포괄적인 프로젝트 만들기 Amazon Q를 블루프린트와 함께 사용하는 방법에 대한 자세한 내용은 을 참조하십시오Amazon Q를 사용하여 프로젝트를 생성하거나 블루프린트로 기능을 추가하는 모범 사례.

Amazon Q를 사용하여 프로젝트를 생성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 블루프린트를 만들려는 스페이스로 이동합니다.
3. 스페이스 대시보드에서 Amazon Q로 만들기를 선택합니다.
4. Amazon Q 프롬프트 텍스트 입력 필드에 빌드하려는 프로젝트에 대한 간략한 설명을 작성하여 지침을 제공합니다. 예제: “`I want to create a project in Python that has`

a presentation layer responsible for how the data is presented, an application layer that contains the core logic and functionality of the application, and a data layer that manages the storage and retrieval of the data.”

(선택 사항) Try examples에서 블루프린트를 선택하여 미리 작성된 프롬프트를 사용할 수 있습니다. 예를 들어 React 앱을 선택하면 다음과 같은 프롬프트가 제공됩니다. “I want to create a project in Python that has a presentation layer responsible for how the data is presented, an application layer that contains the core logic and functionality of the application, and a data layer that manages the storage and retrieval of the data. I also want to add authentication and authorization mechanisms for security and allowable actions.”

5. [Send] 를 선택하여 Amazon Q에 지침을 제출하십시오. 제너레이티브 AI 어시스턴트는 제안 사항을 제공하고 청사진으로는 충족할 수 없는 요구 사항을 설명합니다. 예를 들어 Amazon Q는 기준에 따라 다음을 제안할 수 있습니다.

```
I recommend using the Modern three-tier web application blueprint based on
                    your requirements. Blueprints are dynamic and can always be
 updated and edited later.

Modern three-tier web application
By Amazon Web Services

This blueprint creates a Mythical Mysfits 3-tier web application with a modular
 presentation, application, and data layers.
The application leverages containers, infrastructure as code (IaC), continuous
 integration and continuous delivery (CI/CD),
and serverless code functions.

Version: 0.1.163

View details

The following requirements could not be met so I will create issues for you.
• Add authentication and authorization mechanisms for security and allowable
 actions.
```

6. (선택 사항) 제안된 청사진의 세부 정보를 자세히 보려면 세부 정보 보기를 선택합니다.
7. 다음 중 하나를 수행합니다.

a. 제안이 만족스러우면 예, 이 청사진을 사용하세요.

b. 프롬프트를 수정하려면 프롬프트 편집을 선택합니다.

c. 프롬프트를 완전히 지우려면 [Start over] 를 선택합니다.

8. 다음 중 하나를 수행합니다.

a. 제안된 블루프린트를 구성하려면 구성을 선택하세요. 나중에 블루프린트를 구성할 수도 있습니다.

b. 현재 블루프린트 구성을 수정하지 않으려면 스킵을 선택하세요.

9. 블루프린트를 구성하기로 선택한 경우, 프로젝트 리소스를 수정한 후 Continue를 선택하세요.

10. 메시지가 표시되면 프로젝트에 할당하려는 이름과 관련 리소스 이름을 입력합니다. 이름은 스페이스 내에서 고유해야 합니다.

11. Create project (프로젝트 만들기) 를 선택하여 블루프린트로 프로젝트를 생성합니다. Amazon Q는 블루프린트를 사용하여 리소스를 생성합니다. 예를 들어 단일 페이지 애플리케이션 블루프린트로 프로젝트를 생성하면 CI/CD용 관련 코드 및 워크플로를 위한 소스 리포지토리가 생성됩니다.

12. (선택 사항) 기본적으로 Amazon Q는 청사진으로는 충족되지 않는 요구 사항에 대한 문제도 생성합니다. 이슈를 생성하지 않으려는 항목을 선택할 수 있습니다. Amazon Q에서 이슈를 생성하도록 선택한 후 Amazon Q에도 이슈를 할당할 수 있습니다. 지정된 소스 리포지토리의 컨텍스트에서 문제를 분석하여 관련 소스 파일 및 코드의 요약을 제공합니다. 자세한 내용은 문제 찾기 및 보기, 이슈를 생성하여 Amazon Q에 할당합니다., Amazon Q에 할당된 이슈를 생성하고 처리하는 모범 사례 단원을 참조하세요.

Amazon Q로 프로젝트를 생성한 후에는 요구 사항에 따라 CodeCatalyst 청사진을 제안하므로 Amazon Q를 사용하여 새 구성 요소를 추가할 수도 있습니다.

Amazon Q를 사용하여 청사진을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.

2. CodeCatalyst 콘솔에서 블루프린트를 추가하려는 프로젝트로 이동합니다.

3. Amazon Q로 추가를 선택합니다.

4. Amazon Q 프롬프트 텍스트 입력 필드에 빌드하려는 프로젝트에 대한 간략한 설명을 작성하여 지침을 제공합니다. 예제: “`I want to create a project in Python that has a presentation layer responsible for how the data is presented, an application layer that contains the core logic and functionality of the`

application, and a data layer that manages the storage and retrieval of the data.”

(선택 사항) Try examples에서 블루프린트를 선택하여 미리 작성된 프롬프트를 사용할 수 있습니다. 예를 들어 React 앱을 선택하면 다음과 같은 프롬프트가 제공됩니다. “I want to create a project in Python that has a presentation layer responsible for how the data is presented, an application layer that contains the core logic and functionality of the application, and a data layer that manages the storage and retrieval of the data. I also want to add authentication and authorization mechanisms for security and allowable actions.”

5. [Send] 를 선택하여 Amazon Q에 지침을 제출하십시오. 제너레이티브 AI 어시스턴트는 제안 사항을 제공하고 청사진으로는 충족할 수 없는 요구 사항을 설명합니다. 예를 들어 Amazon Q는 기준에 따라 다음을 제안할 수 있습니다.

```
I recommend using the Single-page application blueprint based on your requirements.
 Blueprints are dynamic and can always be updated and edited later.

Single-page application
By Amazon Web Services

This blueprint creates a SPA (single-page application) using React, Vue, or Angular
 frameworks and deploys to AWS Amplify Hosting.

Version: 0.2.15
View details

The following requirements could not be met so I will create issues for you.
• The application should have reusable UI components
• The application should support for client-side routing
• The application may require server-side rendering for improved performance and
 SEO
```

6. (선택 사항) 제안된 청사진의 세부 정보를 자세히 보려면 세부 정보 보기를 선택합니다.
7. 다음 중 하나를 수행합니다.
   a. 제안이 만족스러우면 예, 이 청사진을 사용하세요.
   b. 프롬프트를 수정하려면 프롬프트 편집을 선택합니다.
   c. 프롬프트를 완전히 지우려면 [Start over] 를 선택합니다.

8. 다음 중 하나를 수행합니다.

    a. 제안된 블루프린트를 구성하려면 구성을 선택하세요. 나중에 블루프린트를 구성할 수도 있습니다.

    b. 현재 블루프린트 구성을 수정하지 않으려면 스킵을 선택하세요.

9. 블루프린트를 구성하기로 선택한 경우, 프로젝트 리소스를 수정한 후 Continue를 선택하세요.

10. 프로젝트에 추가를 선택하여 블루프린트가 있는 프로젝트에 리소스를 추가합니다. Amazon Q는 블루프린트를 사용하여 리소스를 생성합니다. 예를 들어 단일 페이지 애플리케이션 블루프린트로 프로젝트에 추가하면 CI/CD용 관련 코드 및 워크플로를 위한 소스 리포지토리가 생성됩니다.

11. (선택 사항) 기본적으로 Amazon Q는 청사진으로는 충족되지 않는 요구 사항에 대한 문제도 생성합니다. 이슈를 생성하지 않으려는 항목을 선택할 수 있습니다. Amazon Q에서 이슈를 생성하도록 선택한 후 Amazon Q에도 이슈를 할당할 수 있습니다. 지정된 소스 리포지토리의 컨텍스트에서 문제를 분석하여 관련 소스 파일 및 코드의 요약을 제공합니다. 자세한 내용은 이슈를 생성하여 Amazon Q에 할당합니다. 및 Amazon Q에 할당된 이슈를 생성하고 처리하는 모범 사례 단원을 참조하세요.

풀 리퀘스트를 생성할 때 브랜치 간 코드 변경 사항을 요약한 내용을 작성하세요.

풀 리퀘스트는 여러분과 다른 프로젝트 구성원이 한 브랜치에서 다른 브랜치로의 코드 변경 사항을 검토하고, 코멘트를 달고, 병합할 수 있는 주요 방법입니다. 풀 리퀘스트를 사용하면 코드 변경 사항을 공동으로 검토하여 사소한 변경이나 수정, 주요 기능 추가 또는 출시된 소프트웨어의 새 버전을 검토할 수 있습니다. 풀 리퀘스트 설명의 일부로 코드 변경과 변경의 의도를 요약하면 코드를 검토하는 다른 사용자에게 도움이 될 뿐만 아니라 시간 경과에 따른 코드 변경 내역을 이해하는 데도 도움이 됩니다. 그러나 개발자는 검토자가 검토 중인 내용이나 코드 변경의 의도가 무엇인지 이해할 수 있도록 충분한 세부 정보로 변경 내용을 설명하기보다는 코드에 의존하여 스스로 설명하거나 모호한 세부 정보를 제공하는 경우가 많습니다.

풀 요청을 생성할 때 나를 위한 설명 쓰기 기능을 사용하여 Amazon Q에서 풀 요청에 포함된 변경 사항에 대한 설명을 생성하도록 할 수 있습니다. 이 옵션을 선택하면 Amazon Q는 코드 변경 사항이 포함된 소스 브랜치와 이러한 변경 사항을 병합하려는 대상 브랜치 간의 차이를 분석합니다. 그런 다음 이러한 변경 사항에 대한 요약과 해당 변경의 의도와 영향에 대한 최상의 해석을 생성합니다.

> **Note**
>
> 이 기능은 Git 하위 모듈에서는 작동하지 않습니다. pull 요청의 일부인 Git 하위 모듈의 변경 사항은 요약되지 않습니다.
> 연결된 리포지토리의 pull 요청에는 이 기능을 사용할 수 없습니다.

어떤 풀 리퀘스트를 생성하든 이 기능을 사용해 볼 수 있지만, 이 자습서에서는 Python 기반 Modern 3 계층 웹 애플리케이션 블루프린트에서 만든 프로젝트에 포함된 코드를 간단히 변경하여 테스트해 보겠습니다.

> **Tip**
>
> 다른 블루프린트나 자체 코드로 만든 프로젝트를 사용하는 경우에도 이 자습서를 따를 수 있지만 이 자습서의 예제가 프로젝트의 코드와 일치하지 않을 수 있습니다. 아래 제안된 예제 대신 브랜치에서 프로젝트 코드를 간단히 변경한 다음 다음 단계와 같이 풀 요청을 생성하여 기능을 테스트해 보세요.

먼저 소스 리포지토리에 브랜치를 생성합니다. 그런 다음 콘솔의 텍스트 편집기를 사용하여 해당 브랜치에 있는 파일의 코드를 빠르게 변경할 수 있습니다. 그런 다음 풀 리퀘스트를 생성하고 나를 위한 설명 쓰기 기능을 사용하여 변경한 내용을 요약해 보겠습니다.

브랜치를 만들려면 (콘솔)

1. CodeCatalyst 콘솔에서 소스 리포지토리가 있는 프로젝트로 이동합니다.
2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
3. 브랜치를 만들려는 리포지토리를 선택합니다.
4. 리포지토리의 개요 페이지에서 추가를 선택한 다음 브랜치 생성을 선택합니다.
5. 브랜치 이름을 입력합니다.
6. 브랜치를 만들 브랜치를 선택한 다음 [Create] 를 선택합니다.

브랜치가 생성되면 간단히 변경하여 해당 브랜치의 파일을 편집하세요. 이 예시에서는 `test_endpoint.py` 파일을 편집하여 테스트 재시도 횟수를 5번에서 **3** 5번으로 변경합니다.

> **Tip**
>
> 또한 개발 환경을 생성하거나 사용하여 이 코드를 변경할 수도 있습니다. 자세한 내용은 Dev Environment 생성 단원을 참조하십시오.

콘솔에서 **test_endpoint.py** 파일을 편집하려면

1. **mysfits**소스 리포지토리의 개요 페이지에서 브랜치 드롭다운을 선택하고 이전 절차에서 만든 브랜치를 선택합니다.
2. 파일에서 편집하려는 파일을 탐색합니다. 예를 들어 test_endpoint.py 파일을 편집하려면 테스트를 확장하고 integ를 확장한 다음 선택하십시오test_endpoint.py.
3. 편집을 선택합니다.
4. 7행에서 모든 테스트를 재시도할 횟수를 다음과 같이 변경합니다.

   ```
   def test_list_all(retry=3):
   ```

   변경 후:

   ```
   def test_list_all(retry=5):
   ```

5. Commit을 선택하고 변경 내용을 브랜치에 커밋합니다.

이제 브랜치에 변경 사항이 적용되었으니 풀 리퀘스트를 생성할 수 있습니다.

변경 내용을 요약하여 풀 리퀘스트를 생성하세요.

1. 리포지토리의 개요 페이지에서 추가를 선택한 다음 풀 리퀘스트 생성을 선택합니다.
2. 대상 브랜치에서 코드를 검토한 후 병합할 브랜치를 선택합니다.

   > **Tip**
   >
   > 이 기능의 가장 간단한 데모를 보려면 이전 절차에서 브랜치를 만든 브랜치를 선택하십시오. 예를 들어 리포지토리의 기본 브랜치에서 브랜치를 만든 경우 해당 브랜치를 pull 요청의 대상 브랜치로 선택하십시오.

3. 소스 브랜치에서 방금 `test_endpoint.py` 파일에 커밋한 변경 내용이 들어 있는 브랜치를 선택합니다.
4. 풀 리퀘스트 제목에 다른 사용자가 검토해야 할 사항과 그 이유를 이해하는 데 도움이 되는 제목을 입력합니다.
5. 풀 리퀘스트 설명에서 나를 위한 설명 작성을 선택하면 Amazon Q에서 풀 리퀘스트에 포함된 변경 사항에 대한 설명을 생성하도록 할 수 있습니다.
6. 변경 사항 요약이 표시됩니다. 제안된 텍스트를 검토한 다음 수락을 선택하고 설명에 추가를 선택합니다.
7. 코드에서 변경한 내용을 더 잘 반영하도록 요약을 수정할 수도 있습니다. 이 풀 리퀘스트에 리뷰어를 추가하거나 이슈를 연결하도록 선택할 수도 있습니다. 원하는 추가 변경을 마쳤으면 [Create]를 선택합니다.

풀 리퀘스트에서 코드 변경에 남긴 코멘트의 요약을 생성하세요.

사용자는 풀 리퀘스트를 검토할 때 해당 풀 리퀘스트의 변경 사항에 대해 여러 개의 코멘트를 남기는 경우가 많습니다. 많은 리뷰어들의 의견이 많으면 피드백에서 공통된 주제를 고르기가 어려울 수 있으며, 모든 수정 버전에서 모든 댓글을 검토했는지 확인하기 어려울 수 있습니다. 코멘트 요약 생성 기능을 사용하면 Amazon Q가 풀 리퀘스트의 코드 변경에 대한 모든 코멘트를 분석하고 해당 코멘트에 대한 요약을 생성하도록 할 수 있습니다.

> **Note**
>
> 코멘트 요약은 일시적입니다. 풀 리퀘스트를 새로 고치면 요약이 사라집니다. 콘텐츠 요약에는 전체 풀 리퀘스트에 대한 코멘트가 포함되지 않고 풀 리퀘스트 수정 시 코드의 차이에 대해 남긴 코멘트만 포함됩니다.
> 이 기능은 Git 하위 모듈의 코드 변경에 대한 주석에는 작동하지 않습니다.
> 연결된 리포지토리의 pull 요청에는 이 기능을 사용할 수 없습니다.

풀 리퀘스트에서 코멘트 요약을 만들려면

1. 이전 절차에서 생성한 풀 요청으로 이동합니다.

> **Tip**
>
> 원하는 경우 프로젝트에서 모든 공개 풀 리퀘스트를 사용할 수 있습니다. 내비게이션 바에서 코드를 선택하고 풀 리퀘스트를 선택한 다음, 열려 있는 풀 리퀘스트를 선택합니다.

2. 풀 리퀘스트에 아직 댓글이 없는 경우 변경사항에서 풀 리퀘스트에 코멘트를 몇 개 추가하세요.
3. 개요에서 댓글 요약 만들기를 선택합니다. 완료되면 댓글 요약 섹션이 확장됩니다.
4. 풀 리퀘스트 수정 시 코드 변경에 대해 남긴 주석의 요약을 검토하고 이를 풀 리퀘스트의 주석과 비교하십시오.

이슈를 생성하여 Amazon Q에 할당합니다.

개발팀은 작업을 추적하고 관리하기 위해 문제를 생성하지만, 누가 문제를 해결해야 하는지 명확하지 않거나, 코드 베이스의 특정 부분에 대한 조사가 필요한 문제, 또는 기타 긴급한 작업을 먼저 처리해야 하기 때문에 문제가 남아 있는 경우가 있습니다. CodeCatalyst 소프트웨어 개발을 위한 Amazon Q 개발자 에이전트와의 통합이 포함됩니다. 제목과 설명을 기반으로 문제를 분석할 수 있는 Amazon Q라는 제너레이티브 AI 어시스턴트에 이슈를 할당할 수 있습니다. Amazon Q에 문제를 할당하면 평가할 수 있는 초안 솔루션을 만들려고 시도합니다. 이를 통해 사용자와 팀은 주의가 필요한 문제에 집중하고 작업을 최적화할 수 있으며, Amazon Q는 즉시 해결할 리소스가 없는 문제에 대한 솔루션을 개발합니다.

> **Tip**
>
> Amazon Q는 간단한 문제와 간단한 문제에서 가장 잘 작동합니다. 최상의 결과를 얻으려면 평이한 언어를 사용하여 수행하려는 작업을 명확하게 설명하십시오.

Amazon Q에 문제를 CodeCatalyst 할당하면 Amazon Q에서 문제를 해결하는 방법을 확인할 때까지 문제를 차단된 것으로 표시합니다. 계속하려면 먼저 다음 세 가지 질문에 답해야 합니다.

- 모든 단계를 확인하고 싶은지 아니면 피드백 없이 진행하기를 원하는지 여부. 각 단계를 확인하기로 선택한 경우 Amazon Q에서 생성한 접근 방식에 대한 피드백을 제공하여 필요한 경우 접근 방식을 반복할 수 있도록 회신할 수 있습니다. 이 옵션을 선택하면 Amazon Q에서 생성한 모든 풀 리퀘스트에 대해 사용자가 남긴 피드백을 검토할 수도 있습니다. 각 단계를 확인하지 않도록 선택하면 Amazon Q가 작업을 더 빨리 완료할 수 있지만 문제나 생성한 풀 요청에서 제공하는 피드백은 검토하지 않습니다.

- 작업의 일환으로 워크플로 파일을 업데이트하도록 허용할지 여부. 프로젝트에 풀 리퀘스트 이벤트에서 실행을 시작하도록 구성된 워크플로가 있을 수 있습니다. 그렇다면 Amazon Q에서 워크플로 생성 또는 업데이트를 포함하여 생성하는 모든 pull 요청이 pull 요청에 포함된 워크플로의 실행을 시작할 YAML 수 있습니다. 가장 좋은 방법은 생성한 pull 요청을 검토 및 승인하기 전에 프로젝트에 이러한 워크플로를 자동으로 실행하는 워크플로가 없다는 확신이 들지 않는 한 Amazon Q가 워크플로 파일에서 작업하도록 허용하지 않는 것입니다.
- 이슈 관련 작업을 Amazon Q 자체를 포함하여 사용자에게 개별적으로 할당할 수 있는 더 작은 단위로 세분화하는 작업을 생성하도록 제안할지 여부 Amazon Q에서 작업을 제안하고 생성하도록 허용하면 여러 사람이 문제의 개별 부분에 대해 작업할 수 있으므로 복잡한 문제에 대한 개발을 가속화할 수 있습니다. 또한 각 작업을 완료하는 데 필요한 작업이 해당 문제보다 간단하기 때문에 전체 작업을 이해하는 데 따르는 복잡성을 줄이는 데도 도움이 될 수 있습니다.
- 어떤 소스 리포지토리에서 작업하길 원하시나요? 프로젝트에 소스 리포지토리가 여러 개 있더라도 Amazon Q는 하나의 소스 리포지토리에 있는 코드에서만 작업할 수 있습니다. 연결된 리포지토리는 지원되지 않습니다.

선택을 하고 확인하면 Amazon Q는 해당 문제를 진행 중 상태로 전환하고, 동시에 문제 제목과 설명, 지정된 저장소의 코드를 기반으로 요청이 무엇인지 확인합니다. 그러면 작업 상태에 대한 업데이트를 제공하는 고정된 주석이 생성됩니다. Amazon Q는 데이터를 검토한 후 솔루션에 대한 잠재적 접근 방식을 공식화합니다. Amazon Q는 고정된 코멘트를 업데이트하고 모든 단계에서 문제에 대한 진행 상황을 설명하여 조치를 기록합니다. 고정된 댓글 및 답글과 달리 작업을 시간순으로 엄격하게 기록하지는 않습니다. 대신 고정된 댓글의 최상위 레벨에 작업과 관련된 가장 관련성이 높은 정보를 배치합니다. 해당 접근 방식과 저장소에 이미 있는 코드에 대한 분석을 기반으로 코드를 만들려고 시도합니다. 잠재적 솔루션을 성공적으로 생성하면 브랜치를 만들고 해당 브랜치에 코드를 커밋합니다. 그런 다음 해당 브랜치를 기본 브랜치와 병합하는 풀 리퀘스트를 생성합니다. Amazon Q가 작업을 완료하면 문제를 검토 중으로 옮겨 사용자와 팀이 평가할 수 있는 코드가 준비되었음을 알 수 있습니다.

> **Note**
>
> 이 기능은 미국 서부 (오레곤) 지역의 Issues를 통해서만 사용할 수 있습니다. Jira Software 확장 프로그램과 함께 Jira를 사용하도록 프로젝트를 구성한 경우에는 사용할 수 없습니다. 또한 보드 레이아웃을 사용자 지정한 경우 문제가 상태를 변경하지 않을 수 있습니다. 최상의 결과를 얻으려면 표준 보드 레이아웃이 있는 프로젝트에만 이 기능을 사용하십시오.
>
> 이 기능은 Git 하위 모듈에서는 작동하지 않습니다. 리포지토리에 포함된 Git 하위 모듈은 변경할 수 없습니다.

Amazon Q에 이슈를 할당한 후에는 이슈의 제목 또는 설명을 변경하거나 다른 사람에게 할당할 수 없습니다. 이슈에서 Amazon Q 할당을 취소하면 현재 단계가 완료되고 작업이 중지됩니다. 일단 할당이 취소되면 작업을 재개하거나 이슈에 재할당할 수 없습니다.
사용자가 작업 생성을 허용하도록 선택하면 Amazon Q에 문제를 할당하면 자동으로 검토 중 열로 이동될 수 있습니다. 하지만 검토 중인 문제에도 여전히 진행 중 상태와 같은 다른 상태에 있는 작업이 있을 수 있습니다.

자습서의 이 부분에서는 모던 3계층 웹 애플리케이션 블루프린트로 만든 프로젝트에 포함된 코드의 잠재적 기능을 기반으로 세 개의 이슈를 만들어 보겠습니다. 하나는 새로운 mysfit creature를 만들기 위한 것이고, 다른 하나는 정렬 기능을 추가하기 위한 것이고, 다른 하나는 이름이 지정된 브랜치를 포함하도록 워크플로를 업데이트하는 것입니다. **`test`**

Note

코드가 다른 프로젝트에서 작업하는 경우 해당 코드베이스와 관련된 제목과 설명으로 이슈를 만들어 보세요.

문제를 만들고 평가할 수 있는 솔루션을 생성하려면

1. 탐색 창에서 이슈를 선택하고 보드 뷰에 있는지 확인합니다.
2. 이슈 만들기를 선택합니다.
3. 수행하려는 작업을 평이한 언어로 설명하는 제목을 이슈에 붙이세요. 예를 들어, 이 문제의 경우 제목을 입력합니다**`Create another mysfit named Quokkapus`**. 설명에 다음 세부 정보를 제공하십시오.

```
Expand the table of mysfits to 13, and give the new mysfit the following
 characteristics:

Name: Quokkapus

Species: Quokka-Octopus hybrid

Good/Evil: Good

Lawful/Chaotic: Chaotic

Age: 216
```

```
Description: Australia is full of amazing marsupials, but there's nothing there
 quite like the Quokkapus.
She's always got a friendly smile on her face, especially when she's using her
 eight limbs to wrap you up
in a great big hug. She exists on a diet of code bugs and caffeine. If you've got
 some gnarly code that needsa
assistance, adopt Quokkapus and put her to work - she'll love it! Just make sure
 you leave enough room for
her to grow, and keep that coffee coming.
```

4. (선택 사항) 문제에 대한 mysfit의 미리보기 이미지 및 프로필 사진으로 사용할 이미지를 첨부합니다. 이렇게 하려면 사용하려는 이미지와 그 이유에 대한 세부 정보를 포함하도록 설명을 업데이트하세요. 예를 들어 설명에 다음을 추가할 수 있습니다. “mysfit을 사용하려면 웹 사이트에 이미지 파일을 배포해야 합니다. 이번 호에 첨부된 이러한 이미지를 작업의 일환으로 소스 리포지토리에 추가하고 이미지를 웹 사이트에 배포하십시오.”

> **Note**
>
> 이 자습서에서 상호 작용하는 동안 첨부된 이미지가 웹 사이트에 배포될 수도 있고 그렇지 않을 수도 있습니다. 웹 사이트에 이미지를 직접 추가한 다음 Amazon Q에서 pull 요청을 생성한 후 사용하려는 이미지를 가리키도록 코드를 업데이트하도록 댓글을 남길 수 있습니다.

다음 단계로 진행하기 전에 설명을 검토하고 필요할 수 있는 모든 세부 정보가 포함되어 있는지 확인하십시오.

5. 양수인에서 Amazon Q에 할당을 선택합니다.
6. 소스 리포지토리에서 프로젝트 코드가 들어 있는 소스 리포지토리를 선택합니다.
7. Required Amazon Q를 각 단계 후에 중지하고 필요한 경우 작업 선택기의 검토를 활성 상태로 기다립니다.

> **Note**
>
> 모든 단계에서 Amazon Q가 중지되도록 하는 옵션을 선택하면 문제 또는 생성된 작업에 대해 의견을 제시하여 Amazon Q에서 사용자의 의견에 따라 접근 방식을 최대 3회까지 변경하도록 할 수 있습니다. 작업을 검토할 수 있도록 모든 단계에서 Amazon Q가 중지되지 않도록 하는 옵션을 선택하면 Amazon Q가 피드백을 기다리지 않기 때문에 작업이 더 빨리 진행될 수 있지만 댓글을 남긴다고 해서 Amazon Q가 취하는 방향에 영향을 미칠 수는

없습니다. Amazon Q는 풀 리퀘스트에 남긴 댓글에도 응답하지 않습니다. 해당 옵션을 선택하면 Amazon Q는 해당 요청에 응답하지 않습니다.

8. Amazon Q에서 워크플로 파일을 수정하도록 허용 선택기를 비활성 상태로 둡니다.
9. Allow Amazon Q를 슬라이드하여 작업 선택기 생성을 활성 상태로 제안하십시오.
10. 이슈 생성을 선택합니다. 보기가 이슈 게시판으로 바뀝니다.
11. 이슈 생성을 선택하여 이번에는 제목이 있는 이슈를 하나 더 생성합니다**`Change the get_all_mysfits() API to return mysfits sorted by the Age attribute`**. 이 이슈를 Amazon Q에 할당하고 이슈를 생성하십시오.
12. 이슈 생성을 선택하여 다른 이슈를 생성합니다. 이번에는 제목이 있는 **`Update the OnPullRequest workflow to include a branch named test in its triggers`** 이슈입니다. 필요에 따라 설명에 있는 워크플로에 연결할 수 있습니다. 이 문제를 Amazon Q에 할당하되, 이번에는 Amazon Q에서 워크플로 파일을 수정할 수 있도록 허용 선택기가 활성 상태로 설정되어 있는지 확인하십시오. 이슈를 생성하여 이슈 게시판으로 돌아가십시오.

> **Tip**
>
> at 기호 (@) 를 입력하고 파일 이름을 입력하여 워크플로 파일을 비롯한 파일을 검색할 수 있습니다.

이슈를 생성하고 할당하면 이슈가 진행 중으로 이동합니다. Amazon Q는 고정 코멘트에 문제 내 진행 상황을 추적하는 코멘트를 추가합니다. 솔루션에 대한 접근 방식을 정의할 수 있는 경우 코드 베이스에 대한 분석을 포함하는 배경 섹션과 솔루션을 생성하기 위해 제안된 접근 방식을 자세히 설명하는 접근 방식 섹션으로 문제 설명을 업데이트합니다. Amazon Q는 문제에 설명된 문제에 대한 해결책을 찾는 데 성공하면 브랜치를 생성하고 제안된 솔루션을 구현하는 코드를 해당 브랜치에 변경합니다. 제안된 코드가 Amazon Q에서 알고 있는 오픈 소스 코드와 유사한 점을 포함하는 경우 사용자가 검토할 수 있도록 해당 코드에 대한 링크가 포함된 파일을 제공합니다. 코드가 준비되면 제안된 코드 변경 사항을 검토할 수 있도록 풀 요청을 생성하고 해당 풀 리퀘스트의 링크를 이슈에 추가한 다음 이슈를 In Review로 이동합니다.

> **Important**
>
> 풀 리퀘스트를 병합하기 전에 항상 풀 리퀘스트의 코드 변경 사항을 검토해야 합니다. Amazon Q에서 변경한 코드는 다른 코드 변경과 마찬가지로 병합된 코드를 제대로 검토하지 않고 병합 시 오류가 있는 경우 코드 베이스와 인프라 코드에 부정적인 영향을 미칠 수 있습니다.

Amazon Q에서 변경한 사항이 포함된 문제 및 연결된 풀 요청을 검토하려면

1. 이슈에서 Amazon Q에 할당된 진행 중인 이슈를 선택합니다. 의견을 검토하여 Amazon Q의 진행 상황을 모니터링하십시오. 있는 경우 배경을 검토하고 문제 설명에 접근 방법을 기록하십시오. Amazon Q에서 작업을 제안하도록 허용하기로 선택한 경우 제안된 작업을 검토하고 필요한 조치를 취하십시오. 예를 들어, Amazon Q에서 작업을 제안했는데 주문을 변경하거나 특정 사용자에게 작업을 할당하려면 작업 변경, 추가 또는 재정렬을 선택하고 필요한 업데이트를 수행하십시오. 이슈 보기를 마쳤으면 X를 선택하여 이슈 창을 닫습니다.

   > **Tip**
   >
   > 작업 진행 상황을 보려면 이슈의 작업 목록에서 작업을 선택하세요. 작업은 보드에 별도의 항목으로 표시되지 않으며 이슈를 통해서만 접근할 수 있습니다. 작업이 Amazon Q에 할당된 경우 작업을 열어 수행하려는 작업을 승인해야 합니다. 또한 연결된 풀 리퀘스트는 이슈에서 링크로 표시되지 않고 작업에만 표시되므로 작업을 열어야 합니다. 작업에서 이슈로 돌아가려면 이슈로 연결되는 링크를 선택하세요.

2. 이제 검토 중인 Amazon Q에 할당된 이슈를 선택하십시오. 배경을 검토하고 문제의 설명에 접근 방식이 기록되어 있습니다. 의견을 검토하여 해당 의견이 수행한 조치를 이해하십시오. 진행 상황, 취해야 할 조치, 의견 등 이 문제와 관련된 작업을 위해 생성된 모든 작업을 검토하세요. 풀 리퀘스트에서 오픈 라벨 옆에 있는 풀 리퀘스트 링크를 선택하여 코드를 검토하세요.

   > **Tip**
   >
   > 태스크에 대해 생성된 풀 리퀘스트는 태스크 뷰에서 연결된 풀 리퀘스트로만 나타납니다. 해당 문제에 대한 연결된 풀 리퀘스트로는 표시되지 않습니다.

3. 풀 리퀘스트에서 코드 변경사항을 검토하세요. 자세한 내용은 풀 리퀘스트 검토 단원을 참조하십시오. Amazon Q에서 제안된 코드를 변경하도록 하려면 풀 리퀘스트에 의견을 남기십시오. 최상의 결과를 얻으려면 Amazon Q에 댓글을 남길 때는 구체적으로 작성하세요.

   예를 들어, 생성된 pull 요청을 검토할 때 **`Create another mysfit named Quokkapus`** 설명에 오타가 있는 것을 확인할 수 있습니다. Amazon Q에 “" needs"와 “a” 사이에 공백을 추가하여 “needsa”의 오타를 수정하려면 설명을 변경하십시오.” 라는 댓글을 남길 수 있습니다. 또는 설명을 업데이트하고 설명이 포함되도록 수정된 전체 설명을 제공하라는 설명을 Amazon Q에 남길 수도 있습니다.

웹사이트에 새 mysfit 이미지를 업로드한 경우 Amazon Q에 댓글을 남기면 이미지에 대한 포인터와 새 mysfit에 사용할 썸네일로 mysfit을 업데이트할 수 있습니다.

> **Note**
>
> Amazon Q는 개별 의견에 응답하지 않습니다. 문제를 생성할 때 승인을 위해 모든 단계 후에 중지하는 기본 옵션을 선택한 경우에만 Amazon Q에서 댓글에 남긴 피드백을 풀 리퀘스트에 통합합니다.

4. (선택 사항) 귀하와 다른 프로젝트 사용자가 코드 변경을 원하는 모든 코멘트를 남긴 후, Create revision을 선택하여 Amazon Q가 코멘트에 요청한 변경 내용을 포함하는 풀 리퀘스트의 개정판을 생성하도록 합니다. Amazon Q는 변경 내용이 아닌 개요에서 수정 버전 생성 진행 상황을 보고합니다. 수정 버전 생성에 대한 Amazon Q의 최신 업데이트를 보려면 브라우저를 새로 고침해야 합니다.

> **Note**
>
> 이슈를 생성한 사용자만 풀 리퀘스트의 개정판을 생성할 수 있습니다. 풀 리퀘스트의 수정은 한 번만 요청할 수 있습니다. 수정본 만들기를 선택하기 전에 댓글과 관련된 모든 문제를 해결했고 댓글의 내용이 만족스러운지 확인하십시오.

5. 이 예제 프로젝트의 각 풀 리퀘스트에 대해 워크플로가 실행됩니다. 풀 요청을 병합하기 전에 워크플로가 성공적으로 실행되었는지 확인하세요. 또한 코드를 병합하기 전에 테스트할 추가 워크플로와 환경을 만들 수도 있습니다. 자세한 내용은 워크플로우 시작하기 단원을 참조하십시오.
6. 풀 리퀘스트의 최신 버전에 만족하면 Merge를 선택합니다.

이슈를 생성하고 Amazon Q에서 해당 이슈에 대한 작업을 추천하도록 하세요.

때때로 이슈에는 복잡하거나 긴 양의 작업이 포함될 수 있습니다. CodeCatalyst 소프트웨어 개발을 위한 Amazon Q 개발자 에이전트와의 통합이 포함됩니다. Amazon Q에 제목과 설명에 따라 문제를 분석하도록 요청하고 작업을 논리적으로 분류하여 개별 작업으로 나누도록 권장할 수 있습니다. 추천 작업 목록을 만들려고 시도한 다음 검토, 수정 및 생성 여부를 선택할 수 있습니다. 이렇게 하면 여러분과 팀이 보다 관리하기 쉬운 방법으로 작업의 개별 부분을 사용자에게 배정할 수 있어 더 빨리 완료할 수 있습니다.

이슈에 대한 권장 작업 목록을 만들고 검토하기

1. 탐색 창에서 이슈를 선택하고 보드 뷰에 있는지 확인합니다.
2. 이슈 만들기를 선택합니다.
3. 수행하려는 작업을 평이한 언어로 설명하는 제목을 이슈에 붙이세요. 예를 들어, 이 문제의 경우 제목을 입력합니다**`Change the get_all_mysfits() API to return mysfits sorted by the Good/Evil attribute`**. 설명에 다음 세부 정보를 제공하십시오.

   ```
   Update the API to allow sorting of mysfits by whether they are Good, Neutral, or
    Evil. Add a button on the website that allows users to quickly choose this sort
    and to exclude alignments that they don't want to see.
   ```

4. 설명을 검토하고 다음 단계로 진행하기 전에 필요할 수 있는 모든 세부 정보가 포함되어 있는지 확인하십시오.
5. 담당자에서 문제를 자신에게 할당하도록 선택합니다.
6. 이슈 생성을 선택합니다. 보기가 이슈 게시판으로 바뀝니다.
7. 방금 생성한 이슈를 선택하여 엽니다. 추천 작업을 선택합니다.
8. 문제의 코드가 들어 있는 소스 리포지토리를 선택합니다. 작업 추천 시작을 선택합니다.

대화 상자가 닫히고 Amazon Q가 문제의 복잡성 분석을 시작합니다. 문제가 복잡하면 작업을 별도의 순차적 작업으로 나누라는 제안이 나옵니다. 목록이 준비되면 권장 작업 보기를 선택합니다. 다른 작업을 추가하고, 권장 작업을 수정하고, 작업을 재정렬할 수 있습니다. 권장 사항에 동의하는 경우 작업 생성을 선택하면 작업이 생성됩니다. 그런 다음 해당 작업을 사용자에게 할당하여 작업을 수행하거나 Amazon Q 자체에 할당할 수 있습니다.

리소스 정리

이 자습서를 완료한 후에는 다음 작업을 수행하여 이 자습서에서 만든 리소스 중 더 이상 필요하지 않은 리소스를 정리해 보세요.

- 더 이상 해결되지 않는 문제는 Amazon Q의 할당을 취소하십시오. Amazon Q가 문제에 대한 작업을 완료했거나 해결책을 찾지 못한 경우, Amazon Q 할당을 취소하여 생성적 AI 기능의 최대 할당량에 도달하지 않도록 하십시오. 자세한 내용은 제너레이티브 AI 기능 관리 및 요금을 참조하십시오.
- 작업이 완료된 모든 문제를 완료로 옮기세요.
- 프로젝트가 더 이상 필요하지 않은 경우 프로젝트를 삭제하세요.

튜토리얼: 컴포저블 PDK 블루프린트로 풀스택 애플리케이션 만들기

CodeCatalyst Amazon은 프로젝트를 빠르게 시작하는 데 도움이 되는 다양한 청사진을 제공합니다. 블루프린트로 만든 프로젝트에는 소스 리포지토리, 샘플 소스 코드, CI/CD 워크플로, 빌드 및 테스트 보고서, 통합 문제 추적 도구 등 필요한 리소스가 포함됩니다. 하지만 가끔은 프로젝트를 점진적으로 구축하거나 블루프린트로 만든 기존 프로젝트에 기능을 추가하고 싶을 수도 있습니다. 블루프린트로 이 작업을 수행할 수도 있습니다. 이 튜토리얼에서는 토대를 마련하고 모든 프로젝트 코드를 단일 리포지토리에 저장할 수 있는 단일 블루프린트로 시작하는 방법을 보여줍니다. 이제 편의에 따라 초기 청사진 위에 다른 청사진을 추가하여 추가 리소스와 인프라를 유연하게 통합할 수 있습니다. 이 빌딩 블록 방법을 통해 여러 프로젝트의 특정 요구 사항을 해결할 수 있습니다.

이 자습서에서는 여러 AWS 프로젝트 개발 키트 (AWS PDK) 청사진을 함께 작성하여 React 웹 사이트, Smithy API 및 AWS에 배포할 지원 CDK 인프라로 구성된 애플리케이션을 만드는 방법을 보여줍니다. AWS PDK는 프로젝트를 관리하고 구축하기 위한 개발 도구와 함께 공통 패턴을 위한 구성 요소를 제공합니다. 자세한 내용은 AWS PDK GitHub 소스 리포지토리를 참조하십시오.

다음 PDK 블루프린트는 구성 가능한 방식으로 애플리케이션을 구축하는 데 서로 함께 사용할 수 있도록 설계되었습니다.

- Monorepo - 모노레포 내 프로젝트 간의 상호 종속성을 관리하는 루트 수준 프로젝트를 만듭니다. 이 프로젝트는 빌드 캐싱과 종속성 시각화도 제공합니다.
- 유형 안전 API - Smithy 또는 OpenAPI v3에서 정의할 수 있는 API를 생성하고, 유형 안전 방식으로 API를 구현하고 상호 작용할 수 있도록 빌드 타임 코드 생성을 관리합니다. API Gateway에 대한 API 배포를 관리하고 자동 입력 검증을 구성하는 CDK 구조를 제공합니다.
- Cloudscape React 웹 사이트 - Cognito Auth 및 생성된 API (선택 사항) 와 사전 통합되어 제공되는 Cloudscape를 사용하여 구축된 React 기반 웹 사이트를 생성합니다. 이 웹 사이트는 API를 안전하게 호출할 수 있는 기능을 제공합니다.
- 인프라 - 애플리케이션을 배포하는 데 필요한 모든 CDK 관련 인프라를 설정하는 프로젝트를 생성합니다. 또한 빌드할 때마다 CDK 코드를 기반으로 다이어그램을 생성하도록 사전 구성되어 제공됩니다.
- DevOps— AWS 프로젝트 개발 키트 (AWS PDK) 에 있는 구문과 호환되는 DevOps 워크플로를 생성합니다.

이 자습서에는 배포된 애플리케이션을 보고, 다른 사용자를 초대하여 작업하도록 하고, pull 요청이 병합되면 연결된 AWS 계정의 리소스에 자동으로 빌드되어 배포되는 pull 요청을 사용하여 코드를 변경하는 방법에 대한 단계도 포함되어 있습니다.

PDK 블루프린트로 구성된 프로젝트를 생성하면 프로젝트에 다음 리소스가 포함된 프로젝트가 생성됩니다. CodeCatalyst

- 모노레포로 구성된 소스 리포지토리.
- 정적 코드 분석 및 라이선스 검사를 실행하고 기본 브랜치가 변경될 때마다 샘플 코드를 빌드 및 배포하는 워크플로입니다. 코드를 변경할 때마다 아키텍처 다이어그램이 생성됩니다.
- 작업을 계획하고 추적하는 데 사용할 수 있는 이슈 보드 및 백로그입니다.
- 자동 보고서가 포함된 테스트 보고서 세트입니다.

주제

사전 조건

프로젝트를 만들고 업데이트하려면 다음과 설정 및 로그인 CodeCatalyst 같은 작업을 완료해야 합니다.

- 로그인할 AWS 빌더 ID가 있어야 CodeCatalyst 합니다.
- 스페이스에 속하고 해당 스페이스에서 스페이스 관리자 또는 파워 사용자 역할을 할당받아야 합니다. 자세한 내용은 스페이스 만들기, 사용자에게 공간 권한 부여, 스페이스 관리자 역할 단원을 참조하세요.
- 스페이스와 연결된 AWS 계정을 보유하고 가입 시 생성한 IAM 역할을 보유하십시오. 예를 들어 가입 시 CodeCatalystWorkflowDevelopmentRole- ***spaceName*** 역할 정책이라는 역할 정책을 사용하여 서비스 역할을 생성하도록 선택할 수 있습니다. 역할에는 고유 식별자가 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 추가된 이름이 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시

오CodeCatalystWorkflowDevelopmentRole-*spaceName*서비스 역할 이해. 역할 생성 단계는 을 참조하십시오계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-*spaceName*역할 만들기.

1단계: 모노레포 프로젝트 생성

먼저 PDK - Monorepo 블루프린트로 시작하여 추가 PDK 블루프린트를 추가할 수 있는 기반 역할을 하는 모노레포 코드베이스를 만드세요.

PDK를 사용하여 프로젝트를 만들려면 - 모노레포 블루프린트

1. https://codecatalyst.aws/ 에서 콘솔을 엽니다. CodeCatalyst
2. CodeCatalyst 콘솔에서 프로젝트를 만들려는 스페이스로 이동합니다.
3. 스페이스 대시보드에서 프로젝트 생성을 선택합니다.
4. '블루프린트로 시작하기'를 선택합니다.
5. PDK - Monorepo 블루프린트를 선택한 후 다음을 선택합니다.
6. 프로젝트 이름 지정에 프로젝트에 할당하려는 이름과 관련 리소스 이름을 입력합니다. 이름은 스페이스 내에서 고유해야 합니다.
7. 프로젝트 리소스에서 다음을 수행하십시오.
   a. 기본 프로그래밍 언어에서 프로젝트 코드를 개발하는 데 사용할 언어를 선택합니다. Java 또는 Python 중에서 TypeScript 선택할 수 있습니다.
   b. [코드 구성] 을 선택합니다.
   c. 소스 리포지토리 텍스트 입력 필드에 새 리포지토리를 생성할 소스 리포지토리의 이름을 입력하거나 기존 링크된 리포지토리에서 선택합니다. 기존 저장소는 비어 있어야 합니다. 자세한 정보는 소스 리포지토리 연결을 참조하세요.
   d. (선택 사항) Package Manager 드롭다운 메뉴에서 패키지 관리자를 선택합니다. 이는 기본 프로그래밍 언어로 선택한 TypeScript 경우에만 필요합니다.
8. (선택 사항) 선택한 프로젝트 매개 변수에 따라 생성될 코드를 미리 보려면 프로젝트 미리 보기 생성에서 코드 보기를 선택합니다.
9. (선택 사항) 블루프린트 카드에서 세부 정보 보기를 선택하면 블루프린트의 아키텍처 개요, 필요한 연결 및 권한, 블루프린트가 생성하는 리소스 종류 등 블루프린트에 대한 구체적인 세부 정보를 볼 수 있습니다.
10. 프로젝트 생성을 선택하여 모노레포 프로젝트를 생성합니다. 생성된 루트 레벨 프로젝트는 모노레포 내 프로젝트 간의 상호 종속성을 관리하고 빌드 캐싱 및 종속성 관리를 제공합니다.

프로젝트 블루프린트에 대한 자세한 내용은 을 참조하십시오. CodeCatalyst 청사진이 포함된 포괄적인 프로젝트 만들기

PDK - Monorepo 블루프린트는 프로젝트의 토대를 만들 뿐입니다. 블루프린트를 사용하여 실행 가능한 애플리케이션을 만들려면 Type Safe API, Cloudscape React 웹사이트, 인프라 등과 같은 다른 PDK 블루프린트를 추가해야 합니다. DevOps 다음 단계에서는 프로젝트에 Type Safe API를 추가할 것입니다.

2단계: 프로젝트에 타입 세이프 API 추가

PDK - 타입 세이프 API 블루프린트를 사용하면 Smithy 또는 OpenAI v3를 사용하여 API를 정의할 수 있습니다. API 정의에서 런타임 패키지를 생성하는데, 여기에는 API와 상호작용하는 클라이언트와 API 구현을 위한 서버 측 코드가 포함됩니다. 또한 블루프린트는 모든 API 작업에 대해 형식 안전성을 갖춘 CDK 구문을 생성합니다. 기존 PDK monorepo 프로젝트에 블루프린트를 추가하여 프로젝트에 API 기능을 추가할 수 있습니다.

PDK를 추가하려면 - 세이프 API 블루프린트를 입력하세요.

1. 모노레포 프로젝트의 탐색 패널에서 블루프린트를 선택한 다음 블루프린트 추가를 선택합니다.
2. PDK - Type Safe API 블루프린트를 선택하고 다음을 선택합니다.
3. 블루프린트 구성에서 블루프린트 파라미터를 구성합니다.
   - 모델 언어에서 API 모델이 정의되는 언어를 선택합니다.
   - 네임스페이스 텍스트 입력 필드에 API의 네임스페이스를 입력합니다.
   - API 이름 텍스트 입력 필드에 API 이름을 입력합니다.
   - CDK 언어에서 API를 배포할 CDK 인프라를 작성하는 데 사용할 선호 언어를 선택합니다.
   - 핸들러 언어 드롭다운 메뉴를 선택한 다음 API 작업에 사용할 핸들러를 구현하려는 언어를 선택합니다.
   - 문서 형식 드롭다운 메뉴를 선택한 다음 API 설명서 생성에 사용할 형식을 선택합니다.
4. 코드 변경 탭에서 제안된 변경 사항을 검토하십시오. 풀 리퀘스트에 표시된 차이는 풀 리퀘스트가 생성된 시점의 프로젝트 변경 내용을 보여줍니다.
5. 블루프린트 적용 시 제안된 변경 사항이 만족스러우면 블루프린트 추가를 선택합니다.

   풀 리퀘스트를 생성한 후 코멘트를 추가할 수 있습니다. 풀 리퀘스트나 파일의 개별 라인, 전체 풀 리퀘스트에 댓글을 추가할 수 있습니다. @기호와 파일 이름을 차례로 사용하여 파일 등의 리소스에 링크를 추가할 수 있습니다.

> **Note**
>
> 풀 리퀘스트가 승인되고 병합되기 전까지는 블루프린트가 적용되지 않습니다. 자세한 내용은 풀 리퀘스트 검토 및 풀 리퀘스트 병합 섹션을 참조하세요.

6. 상태 열에서 PDK에 대해 Pending pull request - Safe API 블루프린트 행을 입력한 다음, 공개 풀 요청의 링크를 선택합니다.
7. 병합을 선택하고 선호하는 병합 전략을 선택한 다음 병합을 선택하여 적용된 블루프린트의 변경 사항을 통합합니다.

   풀 리퀘스트가 병합되면 구성된 Type Safe API의 API 관련 소스 코드가 모두 들어 있는 monorepo 프로젝트 내에 새 `packages/apis/mypdkapi` 폴더가 생성됩니다.
8. 탐색 창에서 블루프린트를 선택하여 PDK - Type Safe API의 상태가 최신으로 표시되는지 확인합니다.

3단계: 프로젝트에 클라우드스케이프 React 웹 사이트 추가

PDK - 클라우드스케이프 리액트 웹사이트 블루프린트는 웹사이트를 생성합니다. 선택적 매개변수 (Type Safe API) 를 연결하여 다양한 API를 테스트하기 위한 대화형 API 탐색기와 함께 인증된 유형의 안전 클라이언트를 설정하도록 웹사이트를 자동으로 구성할 수 있습니다.

PDK 추가하기 - 클라우드스케이프 리액트 웹사이트 블루프린트

1. 모노레포 프로젝트의 탐색 창에서 블루프린트를 선택한 다음 블루프린트 추가를 선택합니다.
2. PDK - Cloudscape React 웹사이트 블루프린트를 선택한 후 다음을 선택합니다.
3. 블루프린트 구성에서 블루프린트 매개변수를 구성합니다.
   - 웹사이트 이름 텍스트 입력 필드에 웹사이트 이름을 입력합니다.
   - Type Safe API 드롭다운 메뉴를 선택한 다음 웹사이트에 통합하려는 API 블루프린트를 선택합니다. API를 전달하면 인증된 클라이언트가 설정되고 필요한 종속성, API 탐색기 및 기타 기능이 추가됩니다.
4. 코드 변경 탭에서 제안된 변경 사항을 검토하십시오. 풀 리퀘스트에 표시된 차이는 풀 리퀘스트가 생성된 시점의 프로젝트 변경 내용을 보여줍니다.
5. 블루프린트 적용 시 제안된 변경 사항이 만족스러우면 블루프린트 추가를 선택합니다.

풀 리퀘스트를 생성한 후 코멘트를 추가할 수 있습니다. 풀 리퀘스트나 파일의 개별 라인, 전체 풀 리퀘스트에 댓글을 추가할 수 있습니다. @기호와 파일 이름을 차례로 사용하여 파일 등의 리소스에 링크를 추가할 수 있습니다.

> **Note**
>
> 풀 리퀘스트가 승인되고 병합되기 전까지는 블루프린트가 적용되지 않습니다. 자세한 내용은 풀 리퀘스트 검토 및 풀 리퀘스트 병합 섹션을 참조하세요.

6. 상태 열에서 PDK - Cloudscape React 웹 사이트 블루프린트 행에 대해 대기 중인 풀 리퀘스트를 선택한 다음, 열려 있는 풀 리퀘스트의 링크를 선택합니다.
7. 병합을 선택하고 선호하는 병합 전략을 선택한 다음 병합을 선택하여 적용된 블루프린트의 변경 사항을 통합합니다.

   풀 리퀘스트가 병합되면 monorepo 프로젝트 내에 새 웹사이트의 모든 소스 코드가 들어 있는 새 `packages/websites/my-website-name` 폴더가 생성됩니다.
8. 탐색 창에서 블루프린트를 선택하여 PDK - Cloudscape React 웹 사이트의 상태가 최신으로 표시되는지 확인합니다.

다음으로 PDK - 인프라 청사진을 추가하여 웹 사이트를 AWS 클라우드에 배포하기 위한 인프라를 생성합니다.

4단계: 애플리케이션을 AWS 클라우드에 배포하기 위한 인프라 생성

PDK - 인프라 블루프린트는 모든 CDK 코드가 포함된 패키지를 설정하여 웹 사이트와 API를 배포합니다. 또한 기본적으로 다이어그램을 생성하고 프로토타이핑 백 팩을 준수할 수 있습니다.

PDK 추가하기 - 인프라 청사진

1. 모노레포 프로젝트의 탐색 창에서 블루프린트를 선택한 다음 블루프린트 추가를 선택합니다.
2. PDK - 인프라 블루프린트를 선택한 후 다음을 선택합니다.
3. 블루프린트 구성에서 블루프린트 매개변수를 구성합니다.

   - CDK 언어에서 인프라 개발에 사용할 언어를 선택합니다.
   - 스택 이름 텍스트 입력 필드에 블루프린트용으로 생성된 CloudFormation 스택의 이름을 입력합니다.

> **Note**
>
> DevOps 워크플로를 설정하는 다음 단계를 위해 이 스택 이름을 메모해 두세요.

- Type Safe API 드롭다운 메뉴를 선택한 다음 웹 사이트에 통합하려는 API 블루프린트를 선택합니다.
- Cloudscape React TS 웹 사이트 드롭다운 메뉴를 선택한 다음 인프라 내에 배포하려는 웹 사이트 블루프린트 (예: PDK - Cloudscape React 웹 사이트) 를 선택합니다.

4. 코드 변경 탭에서 제안된 변경 사항을 검토하세요. 풀 리퀘스트에 표시된 차이는 풀 리퀘스트가 생성된 시점의 프로젝트 변경 내용을 보여줍니다.
5. 블루프린트 적용 시 제안된 변경 사항이 만족스러우면 블루프린트 추가를 선택합니다.

   풀 리퀘스트를 생성한 후 코멘트를 추가할 수 있습니다. 풀 리퀘스트나 파일의 개별 라인, 전체 풀 리퀘스트에 댓글을 추가할 수 있습니다. @기호와 파일 이름을 차례로 사용하여 파일 등의 리소스에 링크를 추가할 수 있습니다.

   > **Note**
   >
   > 풀 리퀘스트가 승인되고 병합되기 전까지는 블루프린트가 적용되지 않습니다. 자세한 내용은 풀 리퀘스트 검토 및 풀 리퀘스트 병합 섹션을 참조하세요.

6. 상태 열에서 PDK - 인프라 블루프린트 행의 Pending pull request를 선택한 다음, 열려 있는 풀 리퀘스트의 링크를 선택합니다.
7. 병합을 선택하고 선호하는 병합 전략을 선택한 다음 병합을 선택하여 적용된 청사진의 변경 사항을 통합합니다.

   풀 리퀘스트가 병합되면 monorepo 프로젝트 내에 새 `packages/infra` 폴더가 생성되며, 이 폴더에는 프로젝트를 AWS 클라우드에 배포할 인프라가 포함되어 있습니다.
8. 탐색 창에서 블루프린트를 선택하여 PDK - 인프라 상태가 최신으로 표시되는지 확인합니다.

다음으로 PDK - DevOps 블루프린트를 추가하여 애플리케이션을 배포해 보겠습니다.

5단계: 프로젝트를 배포하기 위한 DevOps 워크플로를 설정합니다.

PDK - DevOps blueprint는 구성에 지정된 AWS 계정 및 역할을 사용하여 프로젝트를 빌드하고 배포하는 데 필요한 DevOps 워크플로를 생성합니다.

PDK 추가하기 - 블루프린트 DevOps

1. 모노레포 프로젝트의 탐색 창에서 블루프린트를 선택한 다음 블루프린트 추가를 선택합니다.
2. PDK - DevOps 블루프린트를 선택한 후 다음을 선택합니다.
3. 블루프린트 구성에서 블루프린트 파라미터를 구성합니다.

   - 현재 환경에서 부트스트랩 CDK를 선택합니다.
   - 스택 이름 텍스트 입력 필드에 배포하려는 CloudFormation 스택의 이름을 입력합니다. 이 이름은 PDK - 인프라 4단계: 애플리케이션을 AWS 클라우드에 배포하기 위한 인프라 생성 블루프린트에 구성된 스택 이름과 일치해야 합니다.
   - AWS 계정 연결 드롭다운 메뉴를 선택한 다음 리소스로 사용할 AWS 계정을 선택합니다. 자세한 정보는 AWS 계정 스페이스에 추가을 참조하세요.
   - 애플리케이션 배포에 사용할 역할 드롭다운 메뉴를 선택한 다음 프로젝트 애플리케이션을 배포하는 데 사용할 IAM 역할을 선택합니다.

     > **Note**
     >
     > IAM 역할을 생성할 때는 프로젝트 설정에 있는 현재 `SourceArn` `ProjectID` 역할로 제한하십시오. 자세한 정보는 CodeCatalystWorkflowDevelopmentRole-***spaceName***서비스 역할 이해을 참조하세요.

   - 지역 드롭다운 메뉴를 선택한 다음 monorepo 프로젝트를 배포할 지역을 선택합니다. 배포는 필수 AWS 서비스가 있는 지역에서만 가능합니다. 자세한 내용은 지역별 AWS 서비스를 참조하십시오.
4. 코드 변경 탭에서 제안된 변경 사항을 검토하십시오. 풀 리퀘스트에 표시된 차이는 풀 리퀘스트가 생성된 시점의 프로젝트 변경 내용을 보여줍니다.
5. 블루프린트 적용 시 제안된 변경 사항이 만족스러우면 블루프린트 추가를 선택합니다.

   풀 리퀘스트를 생성한 후 코멘트를 추가할 수 있습니다. 풀 리퀘스트나 파일의 개별 라인, 전체 풀 리퀘스트에 댓글을 추가할 수 있습니다. @기호와 파일 이름을 차례로 사용하여 파일 등의 리소스에 링크를 추가할 수 있습니다.

   > **Note**
   >
   > 풀 리퀘스트가 승인되고 병합되기 전까지는 블루프린트가 적용되지 않습니다. 자세한 내용은 풀 리퀘스트 검토 및 풀 리퀘스트 병합 섹션을 참조하세요.

6. 상태 열에서 PDK - 인프라 블루프린트 행의 Pending pull request를 선택한 다음, 열려 있는 풀 리퀘스트의 링크를 선택합니다.
7. 병합을 선택하고 선호하는 병합 전략을 선택한 다음 병합을 선택하여 적용된 청사진의 변경 사항을 통합합니다.

   풀 리퀘스트가 병합되면 모노레포 프로젝트 내에 새 `.codecatalyst/workflows` 폴더가 생성됩니다.
8. 탐색 창에서 블루프린트를 선택하여 PDK 상태가 최신으로 표시되는지 확인합니다. DevOps

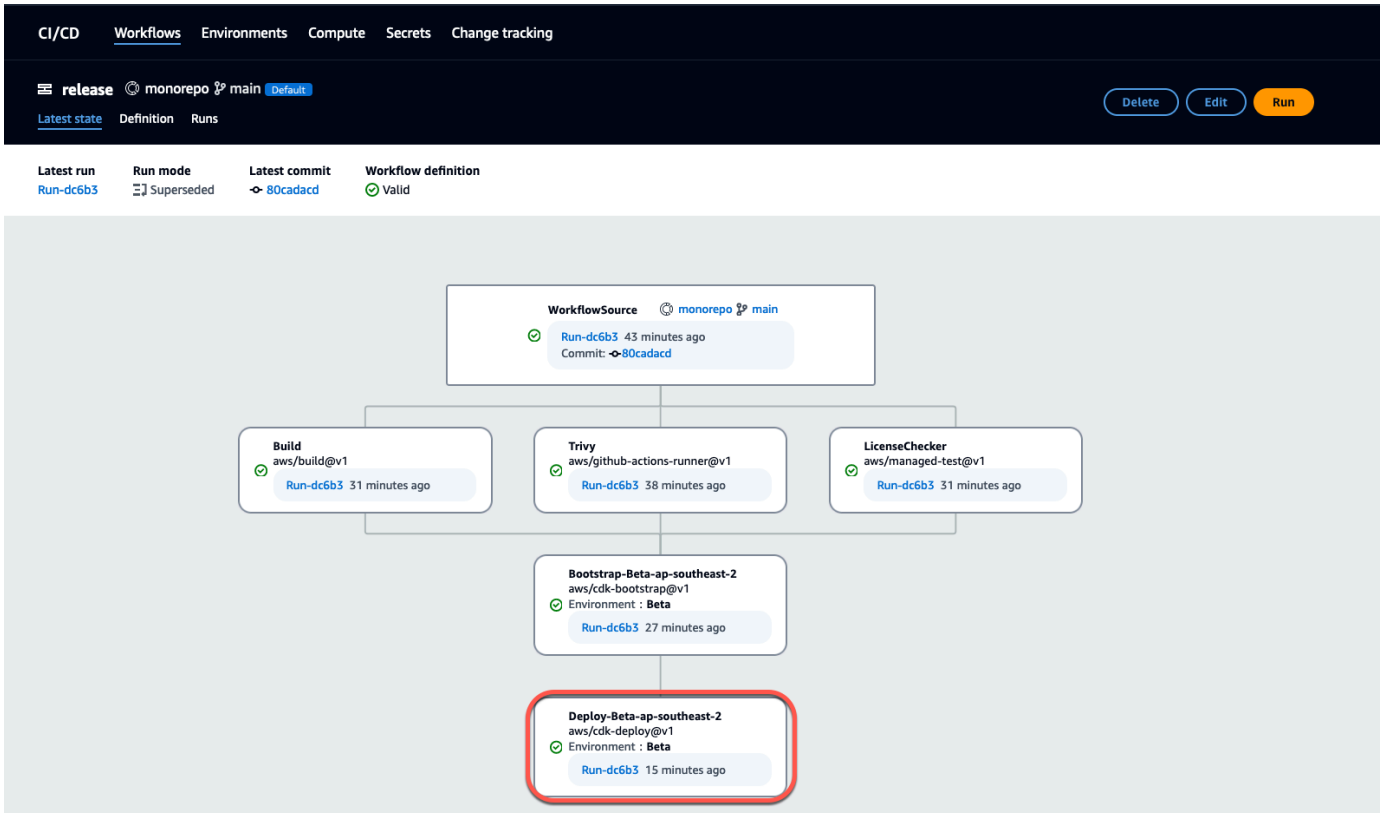
> **Note**
>
> PDK - DevOps 블루프린트와 PDK 블루프린트에 대한 모든 후속 변경 사항은 이 시점부터 상당히 느려질 것입니다. 향후 빌드와 배포를 반복할 수 있도록 백그라운드에서 잠금 파일이 생성되기 때문입니다. 지원되는 모든 언어의 모든 패키지에 대한 잠금 파일이 생성됩니다.

6단계: 출시 워크플로 확인 및 웹 사이트 보기

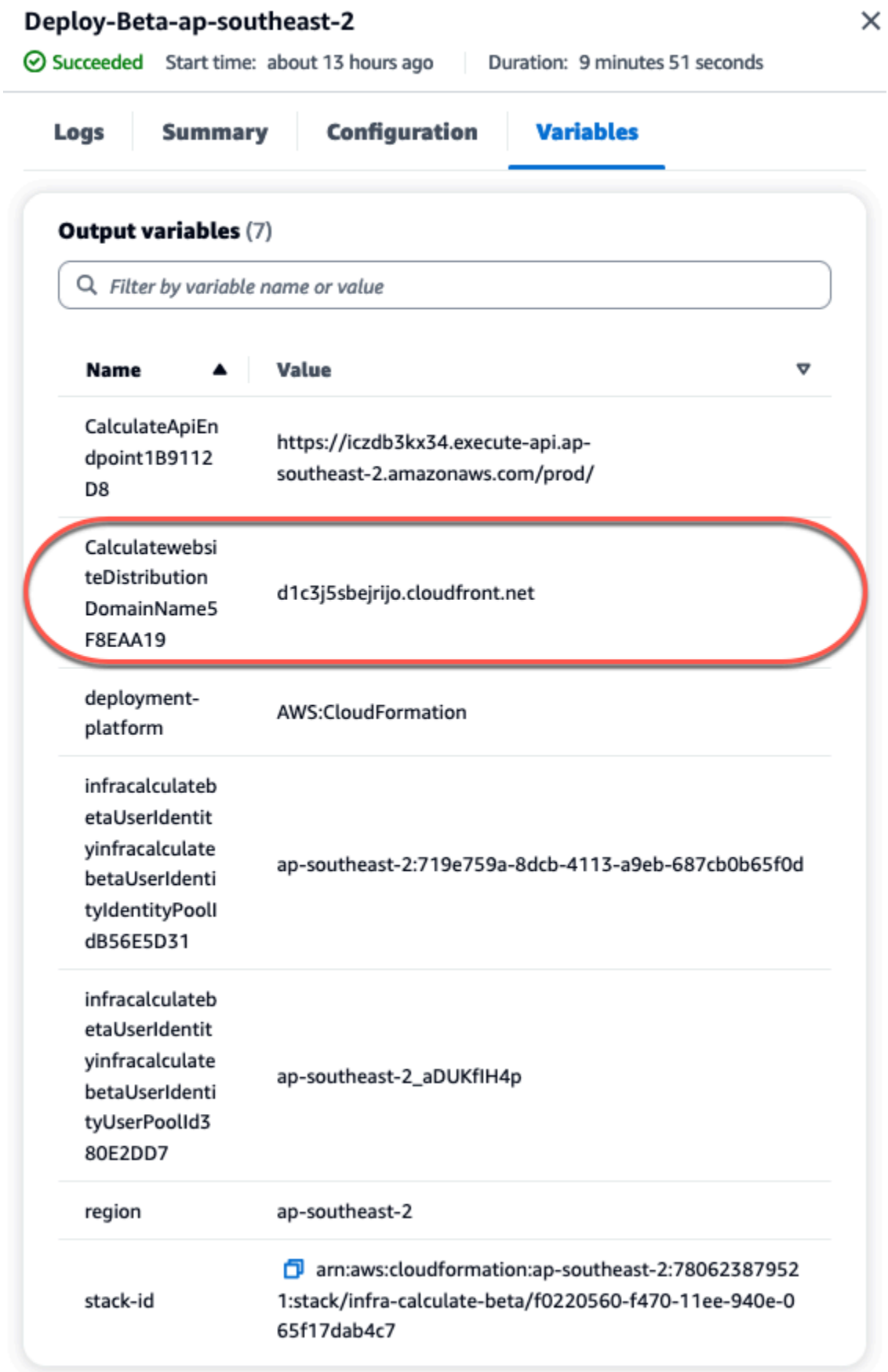
이전 단계를 완료했으면 릴리스 워크플로를 확인하여 프로젝트가 빌드되고 있는지 확인할 수 있습니다.

출시 워크플로를 확인하고 웹사이트를 보려면

1. monorepo 프로젝트의 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 릴리스 워크플로의 경우 최신 워크플로 실행을 선택하여 세부 정보를 확인하세요. 자세한 정보는 단일 실행의 상태 및 세부 정보 보기을 참조하세요.
3. 워크플로 실행이 성공적으로 완료되면 워크플로의 마지막 작업 (예: eta-ap-souteastDeploy-B-2) 을 선택한 다음 변수를 선택합니다.



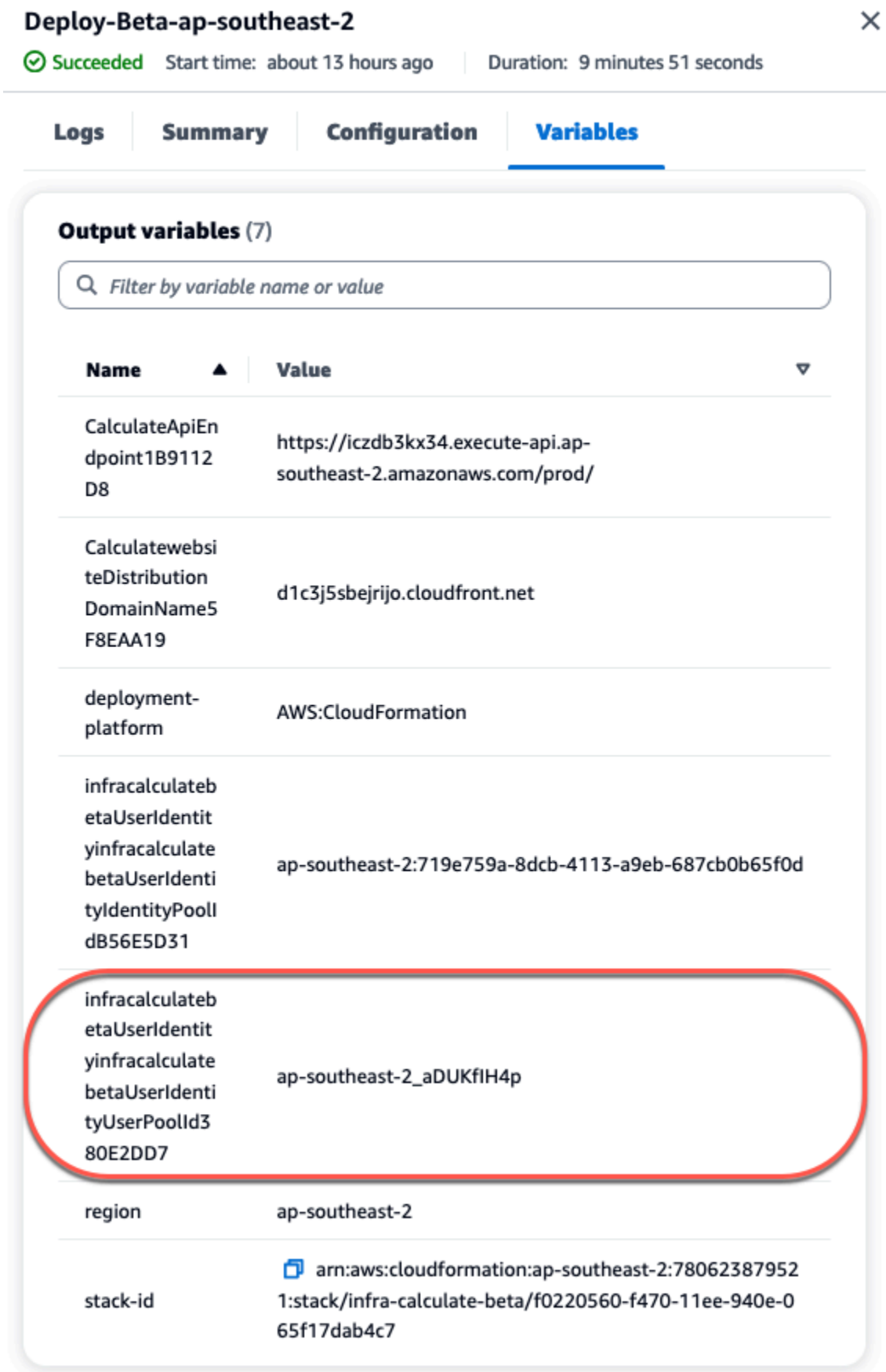
4. 변수 테이블에 있는 링크 (예: ***MyPDKAPI*** websiteDistributionDomain NameXxxxx) 를 복사하여 새 브라우저 창에 붙여넣으면 배포된 웹 사이트를 볼 수 있습니다.


Deploy-Beta-ap-southeast-2
Succeeded
Start time: about 13 hours ago
Duration: 9 minutes 51 seconds
Logs
Summary
Configuration
Variables
Output variables (7)
Filter by variable name or value
Name
Value
CalculateApiEndpoint1B9112D8
https://iczdb3kx34.execute-api.ap-southeast-2.amazonaws.com/prod/
CalculatewebsiteDistributionDomainName5F8EAA19
d1c3j5sbejrijo.cloudfront.net
deployment-platform
AWS:CloudFormation
infracalculatebetaUserIdentityinfracalculatebetaUserIdentityIdentityPooldB56E5D31
ap-southeast-2:719e759a-8dcb-4113-a9eb-687cb0b65f0d
infracalculatebetaUserIdentityinfracalculatebetaUserIdentityUserPoolId380E2DD7
ap-southeast-2_aDUKfIH4p
region
ap-southeast-2
stack-id
arn:aws:cloudformation:ap-southeast-2:780623879521:stack/infra-calculate-beta/f0220560-f470-11ee-940e-065f17dab4c7

웹사이트에 로그인하려면 Amazon Cognito 계정이 필요합니다. 기본적으로 사용자 풀은 자체 등록을 허용하도록 설정되지 않습니다.

a. AWS Cognito 콘솔로 이동합니다.

b. 사용자 풀 ***##### PDK# ### ### ## #### ### # ### #####. # DevOps ### ### ## ##### ## # #### (#: ### ##: XXXXX). betaUserIdentityinfra*** betaUserIdentity IdentityPoolId 자세한 내용은 사용자 풀 시작하기를 참조하십시오.


Deploy-Beta-ap-southeast-2
Succeeded
Start time: about 13 hours ago
Duration: 9 minutes 51 seconds
Logs
Summary
Configuration
Variables
Output variables (7)
Filter by variable name or value
Name
Value
CalculateApiEndpoint1B9112D8
https://iczdb3kx34.execute-api.ap-southeast-2.amazonaws.com/prod/
CalculatewebsiteDistributionDomainName5F8EAA19
d1c3j5sbejrijo.cloudfront.net
deployment-platform
AWS:CloudFormation
infracalculatebetaUserIdentityinfracalculatebetaUserIdentityIdentityPoolIdB56E5D31
ap-southeast-2:719e759a-8dcb-4113-a9eb-687cb0b65f0d
infracalculatebetaUserIdentityinfracalculatebetaUserIdentityUserPoolId380E2DD7
ap-southeast-2_aDUKfIH4p
region
ap-southeast-2
stack-id
arn:aws:cloudformation:ap-southeast-2:780623879521:stack/infra-calculate-beta/f0220560-f470-11ee-940e-065f17dab4c7

c. 사용자 생성을 선택합니다.

d. 사용자 정보 매개변수 구성:

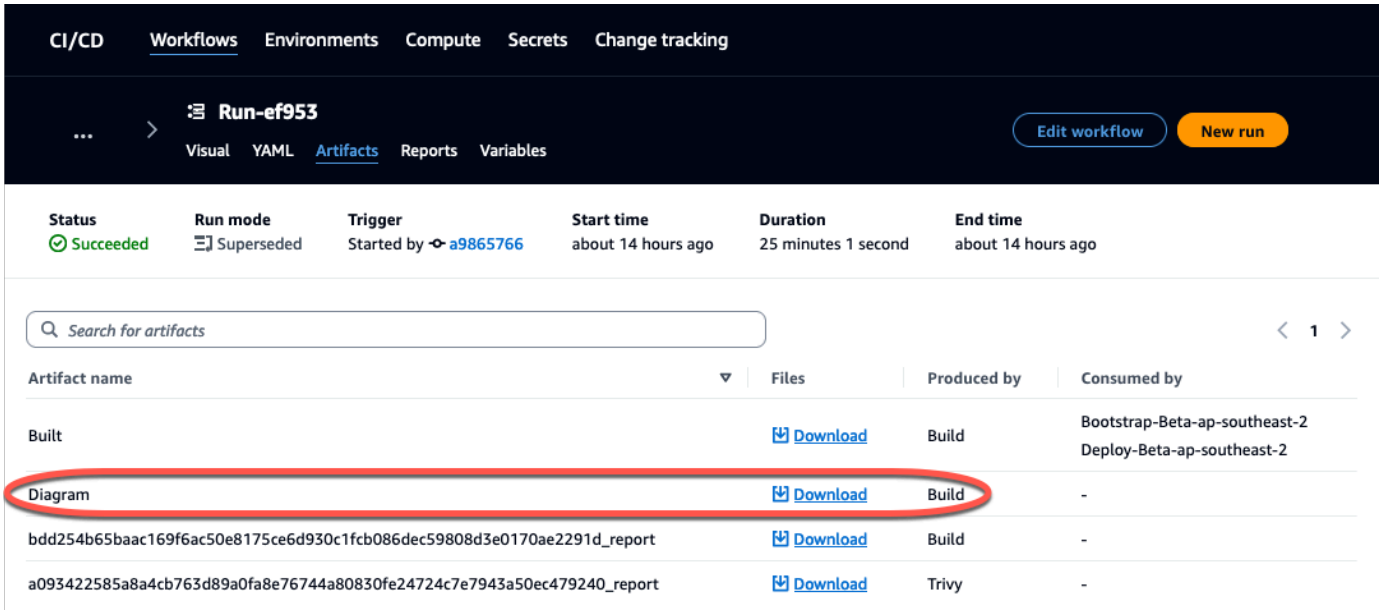
- 초대 메시지에서 이메일 초대 보내기를 선택합니다.
- 사용자 이름 텍스트 입력 필드에 사용자 이름을 입력합니다.
- 이메일 주소 텍스트 입력 필드에 사용자 이름을 입력합니다.
- 임시 비밀번호에서 비밀번호 생성을 선택합니다.

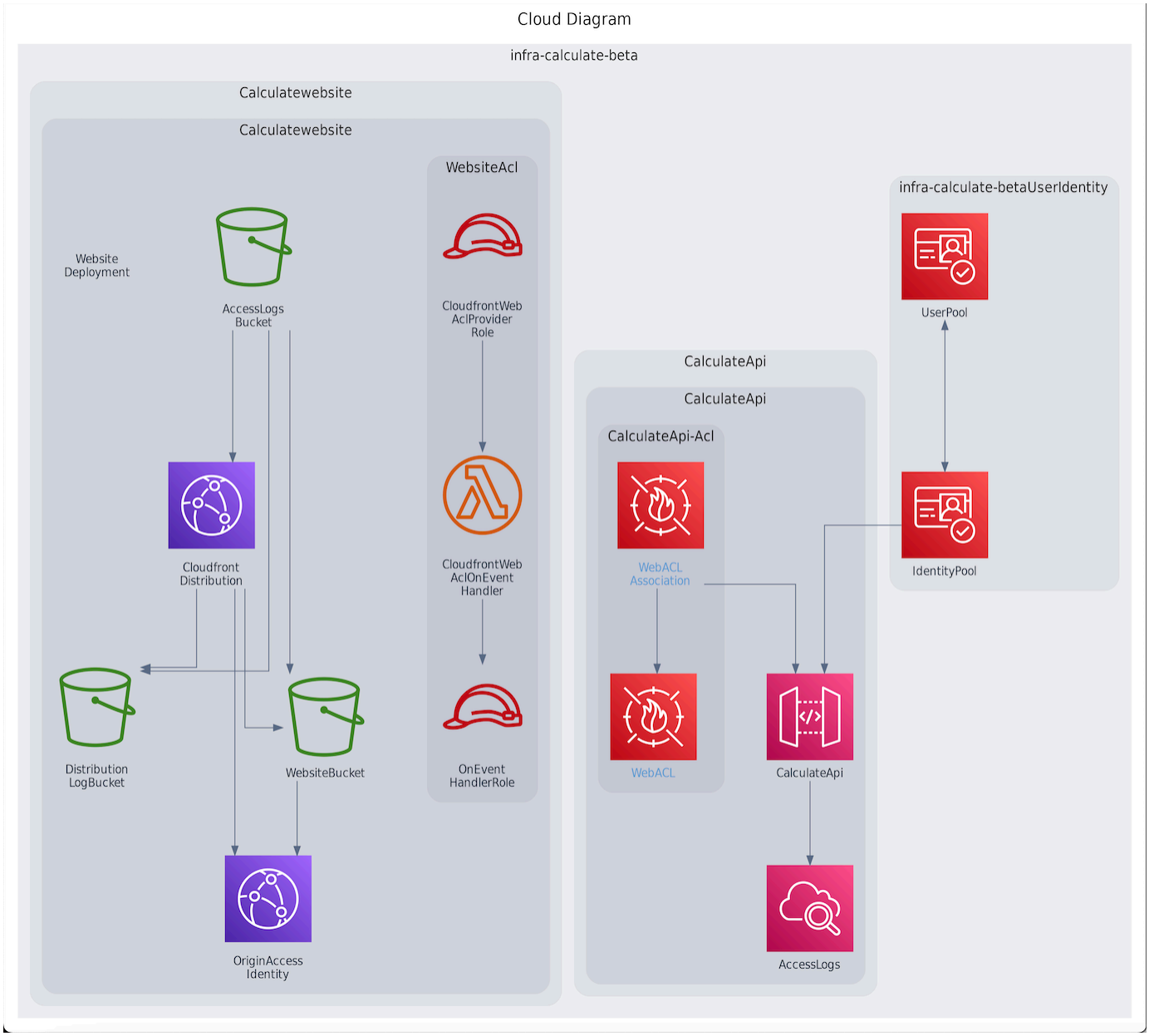
e. 사용자 생성을 선택합니다.

f. 사용자 정보 매개변수로 입력한 이메일 계정으로 이동하여 임시 비밀번호가 포함된 이메일을 엽니다. 비밀번호를 기록해 두십시오.

g. 배포된 웹사이트로 돌아가서 생성한 사용자 이름과 받은 임시 비밀번호를 입력한 다음 로그인을 선택합니다.

5. (선택 사항) 워크플로 실행이 성공적으로 완료된 후 생성된 다이어그램을 볼 수도 있습니다. 에서 CodeCatalyst아티팩트 탭을 선택하고 다이어그램 행에서 다운로드를 선택한 다음 다운로드한 파일을 엽니다.





PDK 프로젝트에서 협업하고 반복하세요

프로젝트를 설정한 후 소스 코드를 변경할 수 있습니다. 다른 스페이스 구성원을 초대하여 프로젝트 작업을 수행할 수도 있습니다. PDK 블루프린트를 사용하면 각 블루프린트의 구성을 완전히 제어하면서 필요할 때 필요한 것만 추가하여 반복적으로 애플리케이션을 빌드할 수 있습니다.

주제

- 1단계: 프로젝트에 구성원 초대
- 2단계: 공동 작업을 위한 이슈 생성 및 작업 추적

- 3단계: 소스 리포지토리 보기
- 4단계: 개발 환경 만들기 및 코드 변경
- 5단계: 코드 변경 사항 푸시 및 병합

1단계: 프로젝트에 구성원 초대

콘솔을 사용하여 프로젝트에 사용자를 초대할 수 있습니다. 스페이스 구성원을 초대하거나 스페이스 외부에서 이름을 추가할 수 있습니다.

사용자를 프로젝트에 초대하려면 프로젝트 관리자 또는 스페이스 관리자 역할로 로그인해야 합니다.

스페이스 관리자 역할을 가진 사용자는 이미 스페이스의 모든 프로젝트에 대한 암시적 액세스 권한을 가지고 있으므로 프로젝트에 초대할 필요가 없습니다.

스페이스 관리자 역할을 할당하지 않고 프로젝트에 사용자를 초대하면 해당 사용자는 프로젝트 아래의 프로젝트 멤버 테이블과 스페이스 아래의 프로젝트 멤버 테이블에 표시됩니다.

프로젝트 설정 탭에서 멤버를 프로젝트에 초대하려면

1. 프로젝트로 이동합니다.

   > **Tip**
   > 상단 내비게이션 바에서 보려는 프로젝트를 선택할 수 있습니다.

2. 탐색 창에서 프로젝트 설정을 선택합니다.
3. 구성원 탭을 선택합니다.
4. 프로젝트 멤버에서 새 멤버 초대를 선택합니다.
5. 새 구성원의 이메일 주소를 입력하고 이 구성원의 역할을 선택한 다음 초대를 선택합니다. 역할에 관한 자세한 내용은 사용자 역할을 통한 액세스 권한 부여 단원을 참조하세요.

프로젝트 개요 페이지에서 멤버를 프로젝트에 초대하려면

1. 프로젝트로 이동합니다.

   > **Tip**
   > 상단 내비게이션 바에서 보려는 프로젝트를 선택할 수 있습니다.

2. 멤버 + 버튼을 선택합니다.
3. 새 구성원의 이메일 주소를 입력하고 이 구성원의 역할을 선택한 다음 초대를 선택합니다. 역할에 관한 자세한 내용은 사용자 역할을 통한 액세스 권한 부여 단원을 참조하세요.

2단계: 공동 작업을 위한 이슈 생성 및 작업 추적

CodeCatalyst 기능, 작업, 버그 및 문제가 있는 프로젝트와 관련된 기타 작업을 추적하는 데 도움이 됩니다. 이슈를 생성하여 필요한 작업과 아이디어를 추적할 수 있습니다. 기본적으로 이슈를 생성하면 백로그에 추가됩니다. 진행 중인 작업을 추적할 수 있는 게시판으로 이슈를 옮길 수 있습니다. 특정 프로젝트 멤버에게 이슈를 할당할 수도 있습니다. 이 단계에서 이슈를 생성하여 PDK 프로젝트를 변경하세요.

이슈를 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 이슈를 만들려는 모노레포 프로젝트로 이동하세요.
3. 프로젝트 홈페이지에서 이슈 생성을 선택합니다. 또는 탐색 창에서 이슈를 선택합니다.
4. 이슈 생성을 선택합니다.

> **Note**
>
> 그리드 보기를 사용할 때 이슈를 인라인으로 추가할 수도 있습니다.

5. 이슈의 제목을 입력합니다.
6. (선택 사항) 설명을 입력합니다. 이 문제의 경우 다음 설명을 입력합니다`a change in the src/mysfit_data.json file.`. 마크다운을 사용하여 서식을 추가할 수 있습니다.
7. (선택 사항) 문제의 상태, 우선 순위, 추정을 선택합니다.
8. (선택 사항) 기존 라벨을 추가하거나 새 라벨을 만든 다음 + 라벨 추가를 선택하여 추가합니다.
   a. 기존 라벨을 추가하려면 목록에서 라벨을 선택합니다. 필드에 검색어를 입력하여 프로젝트에서 해당 용어가 포함된 모든 라벨을 검색할 수 있습니다.
   b. 새 라벨을 생성하여 추가하려면 생성하려는 라벨의 이름을 검색 필드에 입력하고 Enter 키를 누릅니다.
9. (선택 사항) + 담당자 추가를 선택하여 담당자를 추가합니다. + Add me를 선택하여 자신을 담당자로 빠르게 추가할 수 있습니다.

> **Tip**
>
> Amazon Q에 문제를 할당하여 Amazon Q에서 문제 해결을 시도하도록 할 수 있습니다. 자세한 정보는 튜토리얼: CodeCatalyst 제너레이티브 AI 기능을 사용하여 개발 작업의 속도를 높이세요을 참조하세요.
> 이 기능을 사용하려면 공간에 제너레이티브 AI 기능을 활성화해야 합니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.

10. (선택 사항) 기존 사용자 지정 필드를 추가하거나 새 사용자 지정 필드를 생성합니다. 이슈에는 사용자 지정 필드가 여러 개 있을 수 있습니다.

    a. 기존 사용자 지정 필드를 추가하려면 목록에서 사용자 지정 필드를 선택합니다. 필드에 검색어를 입력하여 프로젝트에서 해당 용어가 포함된 모든 사용자 지정 필드를 검색할 수 있습니다.

    b. 새 사용자 지정 필드를 만들어 추가하려면 만들려는 사용자 지정 필드의 이름을 검색 필드에 입력하고 Enter 키를 누릅니다. 그런 다음 만들려는 사용자 지정 필드 유형을 선택하고 값을 설정합니다.

11. 이슈 생성을 선택합니다. 오른쪽 하단에 알림이 표시됩니다. 이슈가 성공적으로 생성되면 이슈가 성공적으로 생성되었다는 확인 메시지가 나타납니다. 이슈가 성공적으로 생성되지 않은 경우 실패 이유가 포함된 오류 메시지가 나타납니다. 그런 다음 [Retry] 를 선택하여 문제를 편집하고 다시 생성하거나 [취소] 를 선택하여 문제를 삭제할 수 있습니다. 두 옵션 모두 알림을 무시합니다.

> **Note**
>
> 풀 리퀘스트를 생성할 때는 이슈에 연결할 수 없습니다. 하지만 생성 후 편집하여 풀 리퀘스트에 링크를 추가할 수 있습니다.

자세한 정보는 에서 문제가 있는 작업을 추적하고 정리하세요. CodeCatalyst 을 참조하세요.

3단계: 소스 리포지토리 보기

Amazon에서 프로젝트와 관련된 소스 리포지토리를 볼 수 있습니다. CodeCatalyst 의 CodeCatalyst 소스 리포지토리의 경우 리포지토리의 개요 페이지는 다음을 포함하여 해당 리포지토리의 정보 및 활동에 대한 간략한 개요를 제공합니다.

- 리포지토리에 대한 설명 (있는 경우)

- 리포지토리 내 브랜치 수
- 리포지토리에 대해 열려 있는 풀 리퀘스트 수
- 리포지토리의 관련 워크플로 수
- 기본 브랜치 또는 선택한 브랜치의 파일 및 폴더
- 표시된 브랜치에 마지막으로 커밋한 제목, 작성자, 날짜
- 마크다운으로 렌더링된 README.md 파일의 내용 (README.md 파일이 포함된 경우)

또한 이 페이지는 리포지토리의 커밋, 브랜치 및 풀 요청에 대한 링크와 개별 파일을 열고, 보고, 편집할 수 있는 빠른 방법을 제공합니다.

> **Note**
>
> 콘솔에서는 연결된 리포지토리에 대한 이 정보를 볼 수 없습니다. CodeCatalyst 연결된 리포지토리에 대한 정보를 보려면 리포지토리 목록에서 링크를 선택하여 해당 리포지토리를 호스팅하는 서비스에서 해당 리포지토리를 여십시오.

프로젝트의 소스 리포지토리로 이동하려면

1. 프로젝트로 이동하여 다음 중 하나를 수행하십시오.
   - 프로젝트 요약 페이지의 목록에서 원하는 리포지토리를 선택한 다음 리포지토리 보기를 선택합니다.
   - 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다. 소스 리포지토리의 목록에서 리포지토리 이름을 선택합니다. 필터 막대에 리포지토리 이름의 일부를 입력하여 리포지토리 목록을 필터링할 수 있습니다.
2. 리포지토리 홈 페이지에서 리포지토리의 콘텐츠와 관련 리소스에 대한 정보 (예: pull request 및 워크플로 수) 를 볼 수 있습니다. 기본적으로 기본 브랜치의 콘텐츠가 표시됩니다. 드롭다운 목록에서 다른 브랜치를 선택하여 보기를 변경할 수 있습니다.

> **Tip**
>
> 프로젝트 요약 페이지에서 프로젝트 코드 보기를 선택하여 프로젝트의 리포지토리로 빠르게 이동할 수도 있습니다.

4단계: 개발 환경 만들기 및 코드 변경

이 단계에서는 개발 환경을 만들고 코드를 변경한 다음 이를 기본 브랜치에 병합합니다. 이 자습서에서는 간단한 AWS PDK 프로젝트를 안내하지만, AWS PDK 리포지토리에서 제공하는 좀 더 복잡한 예제를 따라해볼 수도 있습니다. GitHub

새 브랜치를 사용하여 개발 환경을 만들려면

1. monorepo 프로젝트의 탐색 창에서 다음 중 하나를 수행하십시오.
   - 개요를 선택한 다음 내 개발 환경 섹션으로 이동합니다.
   - 코드를 선택한 다음 개발 환경을 선택합니다.
   - 코드를 선택하고 소스 리포지토리를 선택한 다음 개발 환경을 만들려는 모노레포 리포지토리를 선택합니다.
2. 드롭다운 메뉴에서 지원되는 IDE를 선택합니다. 자세한 정보는 개발 환경을 위해 지원되는 통합 개발 환경을 참조하세요.
3. 리포지토리 복제를 선택합니다.
4. 복제할 리포지토리를 선택하고, 새 브랜치에서 작업을 선택하고, 브랜치 이름 필드에 브랜치 이름을 입력하고, 다음에서 브랜치 생성 드롭다운 메뉴에서 새 브랜치를 만들 브랜치를 선택합니다.

   > **Note**
   >
   > 소스 리포지토리 페이지 또는 특정 소스 리포지토리에서 개발 환경을 만드는 경우 리포지토리를 선택할 필요가 없습니다. 개발 환경은 소스 리포지토리 페이지에서 선택한 소스 리포지토리에서 생성됩니다.

5. (선택 사항) 별칭 - 선택 사항에 개발 환경의 별칭을 입력합니다.
6. (선택 사항) 개발 환경 구성 편집 버튼을 선택하여 개발 환경의 컴퓨팅, 스토리지 또는 타임아웃 구성을 편집합니다.
7. (선택 사항) Amazon 가상 사설 클라우드 (Amazon VPC) - 선택 사항의 드롭다운 메뉴에서 개발 환경과 연결하려는 VPC 연결을 선택합니다.

   공간에 기본 VPC가 설정된 경우 개발 환경은 해당 VPC에 연결되어 실행됩니다. 다른 VPC 연결을 연결하여 이를 재정의할 수 있습니다. 또한 VPC로 연결된 개발 환경은 AWS Toolkit을 지원하지 않는다는 점에 유의하십시오.

> **Note**
>
> VPC 연결을 사용하여 개발 환경을 만들면 VPC 내에 새 네트워크 인터페이스가 생성됩니다. CodeCatalyst 연결된 VPC 역할을 사용하여 이 인터페이스와 상호 작용합니다. 또한 IPv4 CIDR 블록이 IP 주소 범위로 구성되지 않았는지 확인하십시오. `172.16.0.0/12`

8. 생성을 선택하세요. 개발 환경이 생성되는 동안 개발 환경 상태 열에 시작 중이 표시되고, 개발 환경이 생성되면 상태 열에 실행 중이 표시됩니다.

개발 환경이 실행된 후 pull 요청이 병합되면 연결된 AWS 계정의 리소스에 자동으로 빌드되고 배포되는 pull 요청으로 코드를 CodeCatalyst 변경하여 생성된 샘플 애플리케이션을 사용할 수 있습니다. 모노레포는 devfile을 제공하므로 필요한 모든 글로벌 종속 항목 및 런타임이 자동으로 표시됩니다.

프로젝트의 코드를 변경하려면

1. 개발 환경의 작동 중인 터미널에서 monorepo 프로젝트로 이동한 다음 다음 명령어를 실행하여 프로젝트 종속 항목을 설치합니다.

```
npx projen install
```

2. 예제 API 작업을 정의하는 `packages/apis/mypdkapi/model/src/main/smithy/operations/say-hello.smithy` 로 이동합니다. 이 자습서에서는 두 숫자를 더하는 간단한 `Calculate` 연산을 만들어 보겠습니다. 코드를 변경하여 입력과 출력을 포함하여 이 연산을 정의하십시오.

   예:

```
$version: "2"
namespace com.aws

@http(method: "POST", uri: "/calculate")
@handler(language: "typescript")
operation Calculate {
    input := {
        @required
        numberA: Integer
        @required
        numberB: Integer
    }
```

```
    output := {
        @required
        result: Integer
    }
}
```

이 @handler 특성은 Type Safe API에 이 작업을 내장된 AWS Lambda 핸들러로 구현하도록 지시합니다. TypeScript Type Safe API는 사용자가 구현할 수 있는 이 작업에 대한 스텁을 생성합니다. TypeScript @required특성이 추가되었는데, 이는 배포되는 API 게이트웨이에 의해 런타임에 적용된다는 의미입니다. 자세한 내용은 Smithy 설명서를 참조하십시오.

3. 코드 변경에 맞는 /say-hello.smithy 이름으로 파일 이름을 바꾸십시오 (예:). calculate.smithy

4. 로 이동하여 packages/apis/*mypdkapi*/model/src/main/smithy/main.smithy 코드를 변경하여 작업을 연결합니다. 에 정의된 Calculate 작업을 이 파일의 operations 필드에 / calculate.smithy 나열하여 표시할 수 있습니다.

   예:

```
$version: "2"
namespace com.aws

use aws.protocols#restJson1

/// A sample smithy api
@restJson1
service MyPDKApi {
    version: "1.0"
    operations: [Calculate]
    errors: [
      BadRequestError
      NotAuthorizedError
      InternalFailureError
    ]
}
```

5. 다음 명령을 실행하여 변경 내용을 빌드합니다.

```
npx projen build
```

> **Note**
>
> 선택적으로 `--parallel X` 플래그를 전달하여 여러 X 코어에 빌드를 배포할 수 있습니다.

@handler트레이트가 추가되었으므로 빌드가 완료된 후 다음 파일이 생성됩니다.

- /packages/apis/*mypdkapi*/handlers/typescript/src/*calculate.ts*
- /packages/apis/*mypdkapi*/handlers/typescript/test/*calculate.test.ts*

6. packages/apis/*mypdkapi*/handlers/typescript/src/*calculate.ts*로 이동하여 코드를 변경합니다. 이 파일은 API에 대해 호출되는 서버 핸들러입니다.

```
import {
  calculateHandler,
  CalculateChainedHandlerFunction,
  INTERCEPTORS,
  Response,
  LoggingInterceptor,
} from 'mypdkapi-typescript-runtime';

/**
 * Type-safe handler for the Calculate operation
 */
export const calculate: CalculateChainedHandlerFunction = async (request) => {
  LoggingInterceptor.getLogger(request).info('Start Calculate Operation');

  const { input } = request;

  return Response.success({
    result: input.body.numberA + input.body.numberB,
  });
};

/**
 * Entry point for the AWS Lambda handler for the Calculate operation.
 * The calculateHandler method wraps the type-safe handler and manages marshalling
 inputs and outputs
 */
export const handler = calculateHandler(...INTERCEPTORS, calculate);
```

7. /packages/apis/*mypdkapi*/handlers/typescript/test/*calculate.test.ts*파일을 탐색하고 코드를 변경하여 단위 테스트를 업데이트합니다.

   예:

```
import {
  CalculateChainedRequestInput,
  CalculateResponseContent,
} from 'mypdkapi-typescript-runtime';
import {
  calculate,
} from '../src/calculate';

// Common request arguments
const requestArguments = {
  chain: undefined as never,
  event: {} as any,
  context: {} as any,
  interceptorContext: {
    logger: {
      info: jest.fn(),
    },
  },
} satisfies Omit<CalculateChainedRequestInput, 'input'>;

describe('Calculate', () => {

  it('should return correct sum', async () => {
    const response = await calculate({
      ...requestArguments,
      input: {
        requestParameters: {},
        body: {
          numberA: 1,
          numberB: 2
        }
      },
    });

    expect(response.statusCode).toBe(200);
    expect((response.body as CalculateResponseContent).result).toEqual(3);
  });
```

```
});
```

8. /packages/infra/main/src/constructs/apis/*mypdkapi.ts*파일을 탐색하고 코드를 변경하여 CDK 인프라에 Calculate 작업을 위한 통합을 추가하세요. API 구조에는 통합 속성이 있으며, 이 속성을 사용하여 이전에 추가한 구현을 전달할 수 있습니다. Smithy 모델의 @handler 트레이트를 Calculate 작업에 사용하므로 미리 구성된 생성된 CalculateFunction CDK 구문을 사용하여 핸들러 구현을 가리킬 수 있습니다.

   예:

```
import { UserIdentity } from "@aws/pdk/identity";
import { Authorizers, Integrations } from "@aws/pdk/type-safe-api";
import { Stack } from "aws-cdk-lib";
import { Cors } from "aws-cdk-lib/aws-apigateway";
import {
  AccountPrincipal,
  AnyPrincipal,
  Effect,
  PolicyDocument,
  PolicyStatement,
} from "aws-cdk-lib/aws-iam";
import { Construct } from "constructs";
import { Api, CalculateFunction } from "calculateapi-typescript-infra";

/**
 * Api construct props.
 */
export interface CalculateApiProps {
  /**
   * Instance of the UserIdentity.
   */
  readonly userIdentity: UserIdentity;
}

/**
 * Infrastructure construct to deploy a Type Safe API.
 */
export class CalculateApi extends Construct {
  /**
   * API instance
   */
  public readonly api: Api;
```

```
 constructor(scope: Construct, id: string, props?: CalculateApiProps) {
   super(scope, id);

   this.api = new Api(this, id, {
     defaultAuthorizer: Authorizers.iam(),
     corsOptions: {
       allowOrigins: Cors.ALL_ORIGINS,
       allowMethods: Cors.ALL_METHODS,
     },
     integrations: {
       calculate: {
         integration: Integrations.lambda(new CalculateFunction(this,
"CalculateFunction"))
       }
     },
     policy: new PolicyDocument({
       statements: [
         // Here we grant any AWS credentials from the account that the prototype
is deployed in to call the api.
         // Machine to machine fine-grained access can be defined here using more
specific principals (eg roles or
         // users) and resources (ie which api paths may be invoked by which
principal) if required.
         // If doing so, the cognito identity pool authenticated role must still
be granted access for cognito users to
         // still be granted access to the API.
         new PolicyStatement({
           effect: Effect.ALLOW,
           principals: [new AccountPrincipal(Stack.of(this).account)],
           actions: ["execute-api:Invoke"],
           resources: ["execute-api:/*"],
         }),
         // Open up OPTIONS to allow browsers to make unauthenticated preflight
requests
         new PolicyStatement({
           effect: Effect.ALLOW,
           principals: [new AnyPrincipal()],
           actions: ["execute-api:Invoke"],
           resources: ["execute-api:/*/OPTIONS/*"],
         }),
       ],
     }),
   });
```

```
    // Grant authenticated users access to invoke the api
    props?.userIdentity.identityPool.authenticatedRole.addToPrincipalPolicy(
      new PolicyStatement({
        effect: Effect.ALLOW,
        actions: ["execute-api:Invoke"],
        resources: [this.api.api.arnForExecuteApi("*", "/*", "*")],
      }),
    );
  }
}
```

9. 다음 명령어를 실행하여 변경 내용을 빌드하세요.

```
npx projen build
```

프로젝트 빌드가 완료되면 업데이트된 생성된 다이어그램을 볼 수 있습니다. 이 다이어그램은 에서 찾을 수 /packages/infra/main/cdk.out/cdkgraph/diagram.png 있습니다. 다이어그램은 함수가 어떻게 추가되고 생성된 API에 연결되는지를 보여줍니다. CDK 코드가 수정되면 이 다이어그램도 업데이트됩니다.

이제 변경 내용을 저장소의 기본 브랜치에 푸시하고 병합하여 변경 내용을 배포할 수 있습니다.

5단계: 코드 변경 사항 푸시 및 병합

코드 변경 사항을 커밋하고 푸시한 다음 소스 리포지토리의 기본 브랜치에 병합할 수 있습니다.

변경 내용을 기능 브랜치에 푸시하려면

- 다음 명령어를 실행하여 변경 사항을 기능 브랜치에 커밋하고 푸시하세요.

```
git add .
```

```
git commit -m "my commit message"
```

```
git push
```

변경 사항을 푸시하면 기능 브랜치에 대한 새 워크플로가 실행되며 콘솔에서 이를 확인할 수 있습니다. CodeCatalyst 그런 다음 풀 요청을 생성하여 변경 내용을 소스 리포지토리의 기본 브랜치에 병합할 수

있습니다. 기능 브랜치를 메인 브랜치에 병합하면 릴리스 워크플로가 트리거됩니다. 풀 리퀘스트를 이슈에 연결할 수도 있습니다.

풀 리퀘스트를 생성하여 이슈에 연결하려면

1. 모노레포 프로젝트에서 다음 중 하나를 수행하세요.
   - 탐색 창에서 코드를 선택하고 풀 요청을 선택한 다음 풀 리퀘스트 생성을 선택합니다.
   - 리포지토리 홈 페이지에서 추가를 선택한 다음 풀 리퀘스트 생성을 선택합니다.
   - 프로젝트 페이지에서 풀 리퀘스트 생성을 선택합니다.
2. 소스 리포지토리에서 지정된 소스 리포지토리가 커밋된 코드를 포함하는 리포지토리인지 확인합니다. 이 옵션은 리포지토리의 기본 페이지에서 풀 리퀘스트를 만들지 않은 경우에만 나타납니다.
3. 대상 브랜치에서 코드를 검토한 후 병합할 메인 브랜치를 선택합니다.
4. 소스 브랜치에서 커밋된 코드가 포함된 기능 브랜치를 선택합니다.
5. 풀 리퀘스트 제목에 다른 사용자가 검토해야 할 내용과 이유를 이해하는 데 도움이 되는 제목을 입력합니다.
6. (선택 사항) 풀 요청 설명에 문제 링크 또는 변경 내용 설명과 같은 정보를 입력합니다.

> **Tip**
>
> Pull Request에 포함된 변경 사항에 대한 설명을 CodeCatalyst 자동으로 생성하도록 나를 위한 설명 쓰기를 선택할 수 있습니다. 자동으로 생성된 설명을 풀 리퀘스트에 추가한 후 변경할 수 있습니다.
> 이 기능을 사용하려면 공간에 제너레이티브 AI 기능을 활성화해야 합니다. 자세한 내용은 CodeCatalystAmazon의 제너레이티브 AI 기능 관리를 참조하십시오.

7. 이슈에서 이슈 연결을 선택한 다음 에서 2단계: 공동 작업을 위한 이슈 생성 및 작업 추적 생성한 이슈를 선택합니다. 이슈의 연결을 해제하려면 연결 해제 아이콘을 선택합니다.
8. (선택 사항) 필수 검토자에서 필수 검토자 추가를 선택합니다. 프로젝트 멤버 목록에서 선택하여 추가합니다. 풀 리퀘스트를 대상 브랜치에 병합하려면 먼저 필수 검토자가 변경 사항을 승인해야 합니다.

> **Note**
>
> 검토자를 필수 검토자와 선택적 검토자로 모두 추가할 수는 없습니다. 자신을 리뷰어로 추가할 수 없습니다.

9. (선택 사항) 선택적 검토자에서 선택적 검토자 추가를 선택합니다. 프로젝트 멤버 목록에서 선택하여 추가합니다. 선택적 검토자는 풀 리퀘스트를 대상 브랜치에 병합하기 전에 변경 사항을 요구 사항으로 승인할 필요가 없습니다.
10. 풀 리퀘스트는 리뷰어 또는 직접 검토하여 메인 브랜치에 병합해야 합니다. 자세한 정보는 풀 리퀘스트 병합을 참조하세요.

    변경 내용이 소스 리포지토리의 메인 브랜치에 병합되면 새 워크플로가 자동으로 실행됩니다.
11. 병합이 완료되면 이슈를 완료로 옮길 수 있습니다.

    a. 탐색 창에서 이슈를 선택합니다.
    b. 에서 2단계: 공동 작업을 위한 이슈 생성 및 작업 추적 생성한 이슈를 선택하고 상태 드롭다운을 선택한 다음 완료를 선택합니다.

릴리스 워크플로는 실행 성공 후 애플리케이션을 배포하므로 변경 사항을 확인할 수 있습니다.

릴리스 워크플로를 확인하고 웹 사이트를 보려면

1. monorepo 프로젝트의 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 릴리스 워크플로의 경우 최신 워크플로 실행을 선택하여 세부 정보를 확인하세요. 자세한 정보는 단일 실행의 상태 및 세부 정보 보기을 참조하세요.
3. 워크플로 실행이 성공적으로 완료되면 워크플로의 마지막 작업 (eta-ap-souteastDeploy-B-2) 을 선택한 다음 변수를 선택합니다.
4. ***MyPDKapi*** websiteDistributionDomain NameXxxxx 행의 링크를 복사하여 새 브라우저 창에 붙여넣으면 배포된 웹 사이트를 볼 수 있습니다.
5. 생성한 사용자 이름과 암호를 입력한 다음 로그인을 선택합니다. 6단계: 출시 워크플로 확인 및 웹 사이트 보기
6. (선택 사항) 애플리케이션의 변경 사항을 테스트합니다.

    a. POST 드롭다운 메뉴를 선택합니다.
    b. `numberA`및 `number B` 에 두 개의 값을 입력한 다음 실행을 선택합니다.
    c. 응답 본문에서 결과를 확인합니다.

시간이 지나면 PDK 블루프린트의 카탈로그 버전이 변경될 수 있습니다. 프로젝트의 청사진을 카탈로그 버전으로 변경하여 최신 변경 사항을 유지할 수 있습니다. 프로젝트의 블루프린트 버전을 변경하기 전에 코드 변경 사항과 영향을 받는 환경을 확인할 수 있습니다. 자세한 내용은 프로젝트의 블루프린트 버전 변경을(를) 참조하세요.

공백으로 리소스를 정리하세요 CodeCatalyst

본인, 회사, 부서 또는 그룹을 대표하는 공간을 만들고 개발팀이 프로젝트를 관리할 수 있는 공간을 마련하세요. Amazon에서 생성한 프로젝트, 구성원 및 관련 클라우드 리소스를 추가할 스페이스를 생성해야 CodeCatalyst 합니다.

> **Note**
>
> 스페이스 이름은 전체에서 고유해야 CodeCatalyst 합니다. 삭제된 스페이스의 이름은 재사용할 수 없습니다.

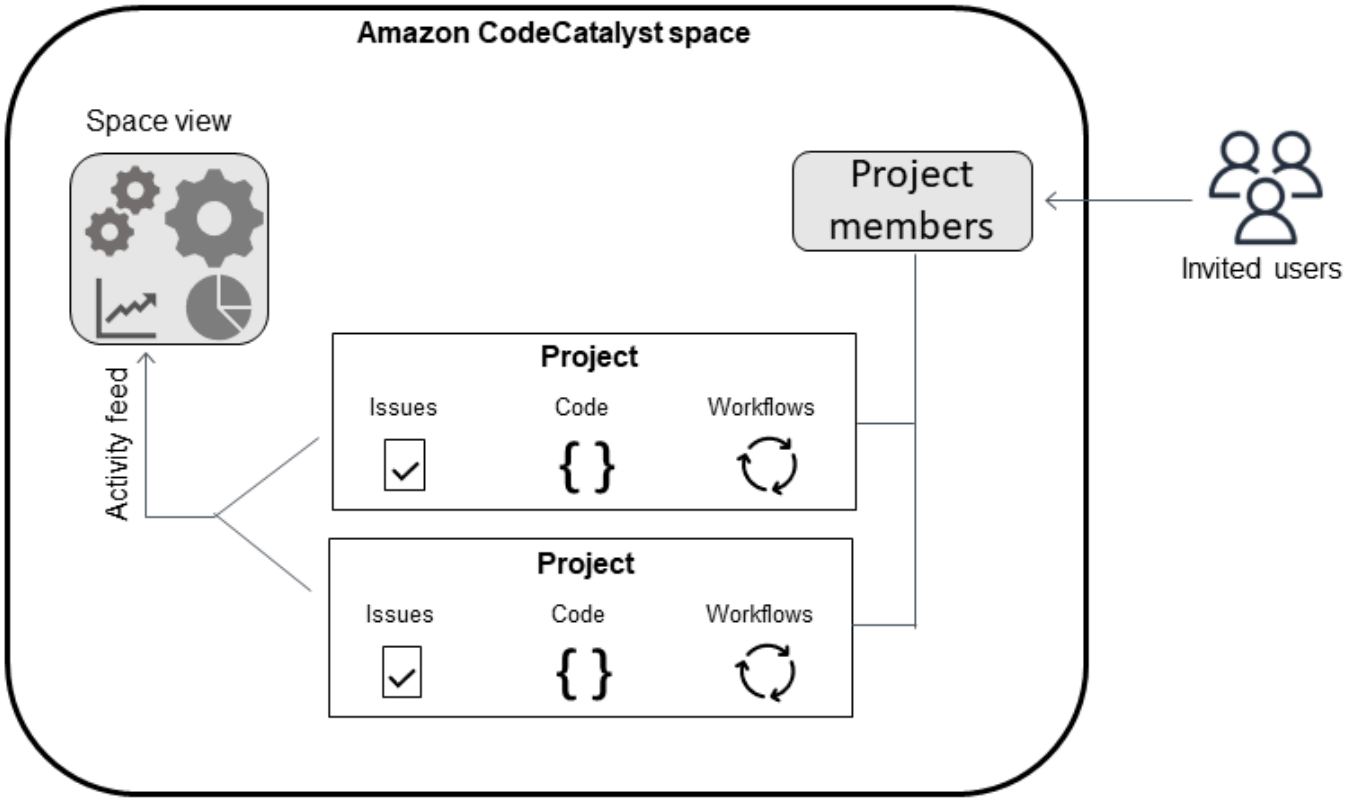
스페이스를 만들면 스페이스 관리자 역할이 자동으로 할당됩니다. 스페이스의 다른 사용자에게 이 역할을 추가할 수 있습니다.

스페이스 관리자 역할을 사용하면 다음과 같이 공간을 관리할 수 있습니다.

- 스페이스에 다른 스페이스 관리자를 추가합니다.
- 구성원 역할 및 권한 변경
- 스페이스 편집 또는 삭제
- 프로젝트를 생성하고 프로젝트에 구성원을 초대합니다.
- 스페이스의 모든 프로젝트 목록 보기
- 스페이스에 있는 모든 프로젝트의 활동 피드 보기

스페이스를 만들면 스페이스 관리자 역할과 스페이스 만들기의 일환으로 만든 프로젝트의 프로젝트 관리자 역할이라는 두 가지 역할이 스페이스에 자동으로 추가됩니다. 프로젝트 초대를 수락하면 추가 사용자가 자동으로 스페이스에 구성원으로 추가됩니다. 스페이스의 이 구성원 자격은 스페이스에 대한 어떠한 권한도 부여하지 않습니다. 스페이스에서 사용자가 수행할 수 있는 작업은 특정 프로젝트에서 사용자의 역할에 따라 결정됩니다.

역할에 관한 자세한 내용은 사용자 역할을 통한 액세스 권한 부여 단원을 참조하세요.



추가 계정에 대한 추가 고려 사항은 다음과 같습니다.

- AWS 계정 CodeCatalyst 스페이스에 추가된 항목은 해당 스페이스의 모든 프로젝트에서 사용할 수 있습니다.
- 각 환경은 여러 AWS 계정개를 지원할 수 있지만 작업에는 환경당 하나의 계정만 사용할 수 있습니다.
- 청구는 공간 수준에서 구성됩니다. 청구를 위해 여러 계정을 구성할 수 있지만 CodeCatalyst 스페이스에서 활성화할 수 있는 계정은 하나뿐입니다. An은 두 개 이상의 스페이스에 대한 결제 계정으로 사용할 AWS 계정 수 CodeCatalyst 있습니다. 스페이스의 AWS 계정 결제 계정으로 지정된 계정에는 CodeCatalyst 스페이스에 대한 다른 계정 연결과 할당량이 다릅니다. 자세한 내용은 에 대한 할당량 CodeCatalyst 단원을 참조하십시오.
- 연결을 만든 후 워크플로에서 환경을 통해 해당 AWS IAM IAM 역할에 액세스해야 하는 경우 연결에 역할을 추가해야 합니다. CodeCatalyst 환경 사용 방법에 대한 자세한 내용은 을 참조하십시오및 에 AWS 계정 배포 VPCs.

주제

- 스페이스 만들기

- 스페이스 편집
- 스페이스 삭제
- 스페이스 내 사용자 및 리소스의 활동 모니터링
- 연결된 AWS 리소스에 대한 액세스 허용 AWS 계정
- 연결된 계정의 IAM 역할 구성
- 사용자에게 공간 권한 부여
- 팀을 통한 공간 액세스 허용
- 시스템 리소스에 대한 공간 액세스 허용
- 스페이스의 개발 환경 관리
- 공간 할당량

스페이스 만들기

AWS 빌더 CodeCatalyst ID로 Amazon에 처음 가입할 때는 공간을 만들어야 합니다. 자세한 내용은 설정 및 로그인 CodeCatalyst 단원을 참조하십시오. 비즈니스 요구 사항에 맞게 추가 공간을 만들 수 있습니다.

> **Note**
>
> 스페이스 이름은 전체에서 고유해야 CodeCatalyst 합니다. 삭제된 스페이스의 이름은 재사용할 수 없습니다.

이 안내서의 정보는 AWS Builder ID 사용자를 CodeCatalyst 지원하는 스페이스를 만들기 위해 제공됩니다. ID 페더레이션을 지원하는 공간을 설정하고 관리하는 단계는 CodeCatalyst 관리자 안내서에 나와 있습니다. ID 페더레이션을 위해 설정된 스페이스를 사용하려면 Amazon CodeCatalyst 관리자 안내서의 CodeCatalyst 스페이스 설정 및 관리를 참조하십시오.

AWS Builder ID 사용자를 지원하는 추가 스페이스를 생성하려면 스페이스 관리자 역할을 할당받아야 합니다.

> **Note**
>
> 추가 공간을 생성할 때 프로젝트를 생성하라는 메시지가 표시되지 않습니다. 스페이스에서 프로젝트를 만드는 방법을 알아보려면 을 참조하십시오프로젝트 생성.

다른 공간을 만들려면

1. 에서 AWS Management Console CodeCatalyst 스페이스에 연결하려는 AWS 계정 것과 동일한 계정으로 로그인했는지 확인하세요.
2. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
3. 스페이스로 이동합니다.

   > **Tip**
   >
   > 두 개 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택하세요.

4. 스페이스 만들기를 선택합니다.
5. 스페이스 만들기 페이지의 스페이스 이름에 스페이스 이름을 입력합니다. 나중에 변경할 수 없습니다.

   > **Note**
   >
   > 스페이스 이름은 전체에서 고유해야 CodeCatalyst 합니다. 삭제된 스페이스의 이름은 재사용할 수 없습니다.

6. 에서 AWS 리전공간 및 프로젝트 데이터를 저장할 지역을 선택합니다. 나중에 변경할 수 없습니다.
7. AWS 계정 ID에 스페이스에 연결하려는 계정의 12자리 ID를 입력합니다.

   AWS 계정 확인 토큰에서 생성된 토큰 ID를 복사합니다. 토큰은 자동으로 복사되지만 AWS 연결 요청을 승인하는 동안 저장하는 것이 좋습니다.
8. 인증을 선택합니다. AWS
9. 에서 Amazon CodeCatalyst 스페이스 확인 페이지가 열립니다 AWS Management Console. 아마존 CodeCatalyst 스페이스 페이지입니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.

   에서 AWS Management Console공간을 만들려는 AWS 리전 위치와 동일한 위치를 선택해야 합니다.

   페이지에 직접 액세스하려면 AWS Management Console at https://console.aws.amazon.com/codecatalyst/ home/에서 Amazon CodeCatalyst Spaces에 로그인하십시오.

   확인 토큰은 확인 토큰에 자동으로 입력됩니다. 성공 배너에는 토큰이 유효한 토큰이라는 메시지가 표시됩니다.

10. 스페이스 확인을 선택합니다.

    계정이 스페이스에 추가되었음을 알리는 계정 확인 성공 메시지가 표시됩니다.

11. Amazon CodeCatalyst 스페이스 확인 페이지에 그대로 남아 있어야 합니다. 다음 링크를 선택하십시오. 이 스페이스에 IAM 역할을 추가하려면 스페이스 세부 정보를 확인하십시오.

    CodeCatalyst 스페이스 세부 정보 페이지가 에서 열립니다 AWS Management Console. 아마존 CodeCatalyst 스페이스 페이지입니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.

12. 사용 가능한 IAM 역할에서 CodeCatalyst IAM 역할 추가를 선택합니다.

    사용 가능한 IAM 역할 추가 CodeCatalyst 페이지가 표시됩니다.

13. 에서 CodeCatalyst 개발 관리자 역할 생성을 선택합니다IAM. 이 옵션은 개발 역할에 대한 권한 정책 및 신뢰 정책을 포함하는 서비스 역할을 생성합니다.

    개발자 역할은 CodeCatalyst 워크플로가 Amazon S3, Lambda 등과 같은 AWS 리소스에 액세스할 수 있도록 하는 AWS IAM 역할입니다. AWS CloudFormation역할에는 고유 식별자가 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 추가된 이름이 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오CodeCatalystWorkflowDevelopmentRole-**`spaceName`**서비스 역할 이해.

14. 개발 역할 생성을 선택합니다.

15. 연결 페이지의 사용 가능한 IAM 역할 아래에서 계정에 추가된 IAM 역할 목록의 개발자 역할을 확인합니다. CodeCatalyst

16. Amazon으로 이동을 선택합니다 CodeCatalyst.

17. 의 생성 페이지에서 스페이스 생성을 선택합니다. CodeCatalyst

스페이스 편집

사용자가 스페이스의 용도를 더 잘 이해할 수 있도록 스페이스 설명을 변경할 수 있습니다.

스페이스 세부 정보를 편집하려면 스페이스 관리자 역할이 있어야 합니다.

이 안내서의 정보는 AWS Builder ID 사용자를 CodeCatalyst 지원하는 스페이스를 편집하기 위해 제공됩니다. ID 페더레이션을 지원하는 스페이스를 설정하고 관리하는 단계에 대한 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 CodeCatalyst 스페이스 설정 및 관리를 참조하십시오.

스페이스 설명을 편집하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.

2. 스페이스로 이동합니다.

> **Tip**
>
> 두 개 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택하세요.

3. 스페이스 설정 탭에서 편집을 선택합니다. 스페이스 설명을 원하는 대로 변경한 다음 저장을 선택합니다.

스페이스 삭제

스페이스를 삭제하여 스페이스의 모든 리소스에 대한 액세스 권한을 제거할 수 있습니다. 스페이스를 삭제하려면 스페이스 관리자 역할이 있어야 합니다.

> **Note**
>
> 스페이스 삭제는 취소할 수 없습니다.

스페이스를 삭제한 후에는 모든 스페이스 구성원이 스페이스 리소스에 액세스할 수 없습니다. 스페이스 리소스에 대한 요금 청구도 중지되며 타사 소스 리포지토리에서 요청하는 모든 워크플로도 중지됩니다.

> **Note**
>
> 스페이스 이름은 전체에서 고유해야 합니다. CodeCatalyst 삭제된 스페이스의 이름은 재사용할 수 없습니다.

이 안내서의 정보는 AWS Builder ID 사용자를 CodeCatalyst 지원하는 스페이스를 삭제하기 위해 제공됩니다. ID 페더레이션을 지원하는 스페이스를 설정하고 관리하는 단계에 대한 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 CodeCatalyst 스페이스 설정 및 관리를 참조하십시오.

스페이스를 삭제하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다.

> **Tip**
>
> 둘 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택하세요.

3. 설정을 선택한 다음 삭제를 선택합니다.
4. **`delete`**입력하여 삭제를 확인합니다.
5. Delete(삭제)를 선택합니다.

> **Note**
>
> 둘 이상의 스페이스에 속해 있는 경우 스페이스 개요 페이지로 리디렉션됩니다. 한 스페이스에 속해 있는 경우 스페이스 생성 페이지로 리디렉션됩니다.

스페이스 내 사용자 및 리소스의 활동 모니터링

최근에 생성된 프로젝트와 상태 업데이트를 보려면 CodeCatalyst 콘솔을 사용하여 스페이스 리소스에 대한 업데이트를 보여주는 활동 피드를 볼 수 있습니다.

활동 피드에서 워크플로 실행 실패, 생성된 프로젝트 등의 지표를 볼 수 있습니다.

스페이스에서의 활동을 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.

> **Tip**
>
> 둘 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택합니다.

3. 활동을 선택합니다.
4. 활동에서 정보를 확인하세요.
5. 활동별로 필터링하려면 오른쪽 상단의 선택기를 선택합니다.
6. 스페이스의 모든 활동을 보려면 모든 활동 유형을 선택합니다.

연결된 AWS 리소스에 대한 액세스 허용 AWS 계정

Amazon AWS 계정 내 CodeCatalyst 스페이스의 리소스를 사용할 수 있습니다. 이렇게 하려면 내 공간과 공간 사이에 연결을 설정해야 CodeCatalyst 합니다. AWS 계정 이와 같은 연결을 만들면 스페이스 내의 프로젝트 및 워크플로가 CodeCatalyst 공간에 있는 리소스와 상호 작용할 수 있습니다 AWS 계정. CodeCatalyst 공간에 AWS 계정 사용하려는 각 연결마다 연결을 하나씩 만들어야 합니다.

연결을 만든 후 AWS IAM 역할을 연결하도록 선택할 수 있습니다.

주제

- AWS 계정 스페이스에 추가
- 계정 연결에 IAM 역할 추가
- 배포 환경에 계정 연결 및 IAM 역할 추가
- 계정 연결 보기
- 계정 연결 삭제 (in CodeCatalyst)
- 스페이스에 대한 결제 계정 구성

CodeCatalyst 스페이스에 계정을 AWS 계정 추가하여 승인된 계정을 사용하도록 설정할 수 있습니다. CodeCatalyst 공간을 추가하면 프로젝트 AWS 계정 워크플로에 AWS 계정 리소스 및 결제 구성에 대한 액세스 권한을 부여할 수 있습니다.

추가하면 이 계정을 사용할 수 있는 권한을 CodeCatalyst 부여하는 연결이 AWS 계정 생성됩니다. added를 AWS 계정 사용하여 다음 작업을 수행할 수 있습니다.

- CodeCatalyst 공간 청구를 설정합니다. Amazon CodeCatalyst 관리자 안내서의 청구 관리를 참조하십시오. 스페이스의 AWS 계정 결제 계정으로 지정된 계정에는 CodeCatalyst 스페이스에 대한 다른 계정 연결과 할당량이 다릅니다. 자세한 내용은 에 대한 할당량 CodeCatalyst 단원을 참조하십시오.
- 계정 CodeCatalyst AWS 서비스 내에서 AWS 리소스에 액세스하고 배포할 IAM 역할을 맡을 수 있습니다. 연결된 계정의 IAM 역할 구성을 참조하세요.

로 인증을 완료하면 계정 연결이 생성됩니다 AWS 계정. 연결이 생성되면 IAM 역할을 추가하여 워크플로와 프로젝트에서 사용할 연결을 추가로 구성합니다.

AWS 계정 및 스페이스의 관리자 권한 AWS Management Console 페이지에서 계정 연결을 구성하는 단계는 CodeCatalyst 관리자 안내서의 연결된 계정 관리를 참조하십시오. CodeCatalyst 특정 프로젝트에 제한되도록 계정 연결을 구성할 수 있습니다. 워크플로우 또는 VPC 연결은 프로젝트에 액세스할

수 AWS 계정 있는 사용자와만 연결할 수 있습니다. 자세한 내용은 프로젝트 제한 계정 연결 구성을 참조하십시오.

AWS 계정 스페이스에 추가

CodeCatalyst 콘솔과 AWS Management Console 를 사용하여 공간을 공간에 연결합니다 AWS 계정.

에서 CodeCatalyst 스페이스에 AWS 계정 를 추가하기 전에 다음 사전 요구 사항을 완료하십시오.

- 계정을 AWS 계정 생성하고 연결하려는 계정에서 AWS IAM 역할을 생성할 수 있는 권한을 획득하십시오.
- 역할에 대한 권한이 있는 IAM 정책을 포함하여 계정 연결에 연결할 역할을 하나 또는 여러 개 생성합니다. IAM
- 연결을 만들려는 CodeCatalyst 스페이스에서 스페이스 관리자 역할을 획득하십시오.

주제

1단계: 연결 요청 생성

CodeCatalyst 콘솔에서 연결 요청을 만들면 인증을 완료하는 데 사용할 수 있는 연결 토큰이 생성됩니다.

연결을 만들려는 스페이스에 스페이스 관리자 또는 파워 사용자 역할이 있어야 합니다. CodeCatalyst 또한 AWS 계정 추가하려는 항목에 대한 관리자 권한이 있어야 합니다.

연결 생성

1. 에서 AWS Management Console연결을 만들려는 계정과 동일한 계정으로 로그인했는지 확인하십시오.
2. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
3. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.

4. 추가를 선택합니다 AWS 계정.
5. AWS 계정 Amazon과 연결 CodeCatalyst 페이지의 AWS 계정 ID에 스페이스에 연결하려는 계정의 12자리 ID를 입력합니다. ID를 찾는 방법에 대한 자세한 내용은 AWS 계정 AWS 계정 ID 및 별칭을 참조하십시오.
6. Amazon CodeCatalyst 표시 이름에 계정의 참조 이름을 입력합니다.
7. (선택 사항) 연결 설명에 계정 및 역할 또는 역할이 적용될 프로젝트를 선택하는 데 도움이 되는 계정 설명을 입력합니다.
8. 연결 AWS 계정을 선택합니다.
9. 페이지는 성공 배너가 표시되는 AWS 계정 세부 정보 페이지로 돌아갑니다.

2단계: 계정 연결 요청 수락

CodeCatalyst 콘솔에서 연결 요청을 제출한 후에는 AWS 관리자와 함께 제공된 연결 토큰을 사용하여 연결 요청을 제출하여 연결 요청을 수락합니다. AWS 계정

계정에 대한 관리자 권한이 있고 연결을 생성할 때 AWS Management Console 사용한 계정으로 로그인했는지 확인하세요. AWS 계정

연결 요청을 승인하려면 (콘솔)

1. 에서 AWS Management Console연결을 만들려는 계정과 동일한 계정으로 로그인했는지 확인하십시오.
2. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
3. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.
4. AWS 계정 세부 정보 페이지의 에서 설정 완료를 선택합니다 AWS Management Console.
5. 에서 Amazon CodeCatalyst 스페이스 확인 페이지가 열립니다 AWS Management Console. 아마존 CodeCatalyst 스페이스 페이지입니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.

   페이지에 직접 액세스하려면 AWS Management Console at https://console.aws.amazon.com/codecatalyst/ home/에서 Amazon CodeCatalyst Spaces에 로그인하십시오.

   확인 토큰은 확인 토큰에 자동으로 입력됩니다. 성공 메시지에는 토큰이 유효한 토큰이라는 메시지가 표시됩니다.
6. (선택 사항) 승인된 유료 티어에서 유료 티어 승인 (Standard, Enterprise) 을 선택하여 결제 계정의 유료 티어를 활성화합니다.

> **Note**
>
> 이렇게 해도 청구 등급이 유료 등급으로 업그레이드되지는 않습니다. 하지만 이렇게 하면 언제든지 숙소의 청구 등급을 변경할 수 AWS 계정 있도록 구성됩니다. CodeCatalyst 언제든지 유료 등급을 켤 수 있습니다. 이렇게 변경하지 않으면 공간은 프리 티어만 사용할 수 있습니다.

7. 스페이스 확인을 선택합니다.

   계정이 스페이스에 추가되었음을 알리는 계정 확인 성공 메시지가 표시됩니다.

3단계: 승인된 연결 검토

연결이 승인되면 콘솔에서 연결에 추가한 IAM 역할과 함께 연결을 볼 수 있습니다.

승인된 연결을 검토하려면

1. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.
2. 계정 연결이 생성된 날짜와 함께 나열됩니다.
3. 계정 표시 이름을 선택합니다. AWS 계정 세부 정보 페이지가 표시됩니다.

4단계: 연결에 IAM 역할 추가

CodeCatalyst 배포 작업에 대해 구성된 IAM 역할을 사용하는 경우 배포 환경에 역할을 추가하세요. 자세한 내용은 계정 연결에 IAM 역할 추가 단원을 참조하십시오.

다음 단계: 계정 연결을 위한 추가 IAM 역할 생성

연결을 생성한 후 추가할 IAM 역할을 추가로 생성할 수 있습니다. 추가하는 IAM 역할은 워크플로에 따라 달라집니다. 예를 들어 CodeCatalyst 빌드 작업에는 CodeCatalyst 빌드 역할이 필요합니다.

계정을 연결하려면 생성한 역할에 대한 Amazon 리소스 이름 (ARN) 이 필요합니다. 여기에 설명된 대로 역할 또는 역할의 ARN 이름을 복사하십시오. IAM역할 사용에 대한 자세한 내용은 Amazon 리소스 이름 (ARN) 을 참조하십시오. ARNs

IAM역할에 액세스하려면 ARN

1. 에서 IAM 콘솔을 엽니다 https://console.aws.amazon.com/iam/.

2. 탐색 창에서 역할을 선택합니다.
3. 검색 상자에 추가하려는 역할의 이름을 입력합니다.
4. 목록에서 역할을 선택합니다.

   역할의 요약 페이지가 나타납니다.
5. 상단에서 역할 ARN 값을 복사합니다.

계정 연결에 IAM 역할 추가

계정 연결 생성 과정에는 CodeCatalyst 스페이스의 프로젝트에 사용할 IAM 역할 또는 역할을 추가하는 작업이 포함됩니다.

> **Note**
>
> 계정 연결이 있는 IAM 역할을 사용하려면 CodeCatalyst 서비스 주체를 사용하도록 신뢰 정책을 업데이트해야 합니다.

계정 연결에 IAM 역할 추가 (콘솔)

1. 에서 AWS Management Console관리하려는 계정과 동일한 계정으로 로그인했는지 확인하십시오.
2. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
3. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.
4. 계정 연결의 Amazon CodeCatalyst 표시 이름을 선택한 다음 역할 관리를 선택합니다 AWS Management Console.

   Amazon CodeCatalyst 스페이스에 IAM 역할 추가 페이지가 표시됩니다.
5. 다음 중 하나를 수행합니다.

   - 개발자 역할에 대한 권한 정책 및 신뢰 정책이 포함된 서비스 역할을 생성하려면 에서 CodeCatalyst 개발 관리자 역할 생성을 선택합니다IAM. 역할에는 고유 식별자가 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 추가된 이름이 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오CodeCatalystWorkflowDevelopmentRole-***spaceName***서비스 역할 이해.

     개발 역할 생성을 선택합니다.

- 에서 IAM 이미 생성한 역할을 추가하려면 기존 IAM 역할 추가를 선택합니다. 기존 IAM 역할 선택의 드롭다운 목록에서 역할을 선택합니다.

  [Add role]을 선택합니다.

  에서 페이지가 열립니다 AWS Management Console. 페이지에 액세스하려면 로그인해야 할 수 있습니다.

6. Amazon CodeCatalyst 스페이스 페이지 탐색 창에서 스페이스를 선택합니다.

   페이지에 직접 액세스하려면 AWS Management Console at https://console.aws.amazon.com/codecatalyst/ home/에서 Amazon CodeCatalyst Spaces에 로그인하십시오.

7. CodeCatalyst 스페이스에 추가된 계정을 선택합니다. 연결 페이지가 표시됩니다.
8. 연결 페이지의 사용 가능한 IAM 역할 아래에서 계정에 추가된 IAM 역할 목록을 확인합니다. CodeCatalyst [IAM역할 연결 대상] 을 선택합니다 CodeCatalyst.
9. 역할 연결 팝업의 IAM역할에 ARN CodeCatalyst 스페이스와 연결할 IAM 역할의 Amazon 리소스 이름 (ARN) 을 입력합니다.

   목적에서 계정 연결에서 역할을 어떻게 사용할지 설명하는 역할 용도를 선택합니다. 워크플로에서 작업을 실행하는 데 RUNNER 사용할 역할을 지정합니다. 다른 서비스에 액세스하는 데 사용할 역할을 SERVICE 지정하십시오.

   용도를 두 개 이상 지정할 수 있습니다.

   > **Note**
   >
   > ARN역할의 용도를 선택해야 합니다.

10. [IAM역할 연결] 을 선택합니다. IAM역할을 추가하려면 이 단계를 반복하세요.

배포 환경에 계정 연결 및 IAM 역할 추가

Amazon 등의 AWS 리소스 ECS 또는 배포용 AWS Lambda 리소스에 액세스하려면 CodeCatalyst 빌드 및 배포 작업에 해당 리소스에 액세스할 권한이 있는 IAM 역할이 필요합니다. 스페이스 관리자 또는 파워 사용자 역할을 사용하면 리소스가 생성되는 AWS 계정 위치에 CodeCatalyst 계정을 연결할 수 있습니다. 그런 다음 계정 연결에 IAM 역할을 추가합니다. 배포 작업의 경우 CodeCatalyst 환경에 IAM 역할을 추가해야 합니다.

배포 환경에서 사용할 IAM 역할을 프로젝트에 추가해야 합니다. 계정 연결에 역할을 추가해도 프로젝트 배포 환경에는 역할과 연결이 추가되지 않습니다. 배포 환경에 계정 연결 및 IAM 역할을 추가하려면 에 설명된 대로 계정 연결 및 역할을 생성해야 4단계: 연결에 IAM 역할 추가 합니다.

그런 다음 CodeCatalyst 콘솔의 환경 페이지를 사용하여 프로젝트의 배포 환경에 계정 연결 및 IAM 역할을 추가합니다.

> **Note**
>
> IAM역할이 필요한 CodeCatalyst 작업에 역할을 사용하는 경우에만 환경에 IAM 역할을 추가할 수 IAM 있습니다. 빌드 작업을 포함하여 IAM 역할이 필요한 모든 워크플로 작업은 환경을 사용해야 합니다. CodeCatalyst

배포 환경에 계정 연결 및 IAM 역할을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 계정 연결 및 IAM 역할을 추가하려는 배포 환경이 있는 프로젝트로 이동합니다.
3. CI/CD를 확장한 다음 환경을 선택합니다.
4. 환경을 선택하면 추가 탭이 표시됩니다.
5. AWS 계정 연결 탭을 선택합니다. 연결 이름 아래에 환경에 추가된 계정 (있는 경우) 이 나열됩니다.
6. 연결 AWS 계정을 선택합니다. AWS 계정 연결 대상 <environment_name>페이지가 표시됩니다.
7. 연결에서 추가하려는 IAM 역할이 포함된 계정 연결 이름을 선택합니다. 연결을 선택합니다.

계정 연결 보기

연결 목록을 보고 각 연결에 대한 세부 정보를 볼 수 있습니다.

스페이스 연결을 관리하려면 스페이스 관리자 또는 파워 사용자 역할이 있어야 합니다.

CodeCatalyst 스페이스의 모든 연결을 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 보려는 계정 연결이 있는 스페이스로 이동합니다.
3. AWS 계정 탭을 선택합니다.

4. AWS 계정에서 각 연결의 계정 ID 및 상태를 포함하여 스페이스의 계정 연결 목록을 볼 수 있습니다.

계정 연결 세부 정보를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.
3. Amazon CodeCatalyst 표시 이름에서 연결 이름을 선택합니다. 세부 정보 페이지에서 다른 세부 정보와 함께 연결과 관련된 IAM 역할 목록을 볼 수 있습니다.

계정 연결 삭제 (in CodeCatalyst)

더 이상 필요하지 않은 계정 연결은 삭제할 수 있습니다. 이 절차에서는 이전에 CodeCatalyst 스페이스에 추가한 계정 연결을 삭제하는 데 사용합니다. 이렇게 하면 스페이스에서 계정 연결이 삭제됩니다. 단, 계정이 스페이스의 결제 계정이 아닌 경우에 한합니다.

> **Important**
>
> 계정 연결이 삭제된 후에는 다시 연결할 수 없습니다. 새 계정 연결을 만든 다음 필요에 따라 IAM 역할과 환경을 연결하거나 결제를 설정해야 합니다.

CodeCatalyst 공간 사용량이 프리 티어를 초과하지 않는 경우에도 공간에 대한 결제 계정을 지정해야 합니다. 지정된 결제 계정인 계정의 공간을 제거하려면 먼저 해당 공간에 다른 계정을 추가해야 합니다. Amazon CodeCatalyst 관리자 안내서의 청구 관리를 참조하십시오.

> **Important**
>
> 다음 단계를 사용하여 계정을 제거할 수 있지만 권장하지는 않습니다. 의 워크플로를 지원하도록 계정을 설정할 수도 CodeCatalyst 있습니다.

스페이스의 계정 연결을 관리하려면 스페이스 관리자 또는 고급 사용자 역할이 있어야 합니다.

제거된 계정은 나중에 다시 추가할 수 있지만 계정과 스페이스 사이에 새 연결을 만들어야 합니다. 추가된 계정에 모든 IAM 역할을 다시 연결해야 합니다.

계정 연결을 삭제하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.
3. Amazon CodeCatalyst 표시 이름에서 제거하려는 계정 연결 옆의 선택기를 선택합니다.
4. 제거 AWS 계정를 선택합니다. 필드에 이름을 입력하여 삭제를 확인한 다음 제거를 선택합니다.

   성공 배너가 표시되고 연결 목록에서 계정 연결이 제거됩니다.

스페이스에 대한 결제 계정 구성

CodeCatalyst 공간 사용량이 프리 티어를 초과하지 않더라도 공간에 대한 결제 계정을 지정해야 합니다.

결제 계정을 구성하려면 CodeCatalyst 관리자 안내서의 청구를 참조하십시오. 스페이스의 AWS 계정 결제 계정으로 지정된 계정에는 CodeCatalyst 스페이스에 대한 다른 계정 연결과 할당량이 다릅니다. 자세한 내용은 에 대한 할당량 CodeCatalyst 단원을 참조하십시오.

CodeCatalyst 스페이스에 지정된 결제 계정인 계정을 제거하려면 먼저 새 결제 계정을 지정해야 합니다.

연결된 계정의 IAM 역할 구성

AWS Identity and Access Management (IAM) 에서 추가하려는 계정의 역할을 생성합니다 CodeCatalyst. 결제 계정을 추가하는 경우 역할을 생성할 필요가 없습니다.

에는 스페이스에 AWS 계정 추가하려는 역할을 만들 수 AWS 계정있는 권한이 있어야 합니다. IAM 참조 및 예제 정책을 포함하여 IAM 역할 및 정책에 대한 자세한 내용은 을 참조하십시오Identity 및 Access Management와 Amazon CodeCatalyst. 에서 CodeCatalyst 사용되는 신뢰 정책 및 서비스 주체에 대한 자세한 내용은 을 참조하십시오. CodeCatalyst 신뢰 모델 이해

에서 CodeCatalyst 스페이스에 계정 (및 해당하는 경우 역할) 을 추가하는 단계를 완료하려면 스페이스 관리자 역할로 로그인해야 합니다.

다음 방법 중 하나를 사용하여 계정 연결에 역할을 추가할 수 있습니다.

- 에 대한 권한 정책 및 신뢰 정책을 포함하는 서비스 역할을 만들려면 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할은 을 참조하십시오CodeCatalystWorkflowDevelopmentRole-***spaceName*** 역할.

- 역할을 만들고 정책을 추가하여 블루프린트에서 프로젝트를 생성하는 예제는 을 참조하십시오IAM 역할 생성 및 CodeCatalyst 신뢰 정책 사용.
- 역할을 만들 때 사용할 샘플 IAM 역할 정책 목록은 을 참조하십시오IAM 역할을 사용하여 프로젝트 AWS 리소스에 대한 액세스 권한 부여.
- 워크플로 작업에 대한 역할을 만드는 자세한 단계는 다음과 같이 해당 작업에 대한 워크플로 자습서를 참조하십시오.
  - 자습서: Amazon S3에 아티팩트 업로드
  - 자습서: 서버리스 애플리케이션 배포
  - 자습서: Amazon에 애플리케이션 배포 ECS
  - 튜토리얼: 액션을 GitHub 사용한 린트 코드

주제

- CodeCatalystWorkflowDevelopmentRole-spaceName 역할
- AWSRoleForCodeCatalystSupport 역할
- IAM역할 생성 및 CodeCatalyst 신뢰 정책 사용

CodeCatalystWorkflowDevelopmentRole-*spaceName* 역할

에서 개발자 역할을 원클릭 역할로 생성합니다. IAM 계정을 추가하려는 스페이스에 스페이스 관리자 또는 고급 사용자 역할이 있어야 합니다. 또한 AWS 계정 추가하려는 항목에 대한 관리자 권한이 있어야 합니다.

아래 절차를 시작하기 전에 CodeCatalyst 스페이스에 추가하려는 것과 동일한 계정으로 로그인해야 합니다. AWS Management Console 그렇지 않으면 콘솔에서 알 수 없는 계정 오류가 반환됩니다.

생성 및 추가하기 CodeCatalyst CodeCatalystWorkflowDevelopmentRole-*spaceName*

1. CodeCatalyst 콘솔에서 시작하기 전에 를 AWS Management Console열고 AWS 계정 스페이스에 동일한 계정으로 로그인했는지 확인하십시오.
2. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
3. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.
4. 역할을 생성하려는 AWS 계정 위치의 링크를 선택합니다. AWS 계정 세부 정보 페이지가 표시됩니다.
5. 에서 역할 관리를 선택합니다 AWS Management Console.

Amazon CodeCatalyst 스페이스에 IAM 역할 추가 페이지가 에서 열립니다 AWS Management Console. 아마존 CodeCatalyst 스페이스 페이지입니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.

6. 에서 CodeCatalyst 개발 관리자 역할 생성을 선택합니다IAM. 이 옵션은 개발 역할에 대한 권한 정책 및 신뢰 정책을 포함하는 서비스 역할을 생성합니다. 역할에는 이름이 `CodeCatalystWorkflowDevelopmentRole-spaceName` 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오CodeCatalystWorkflowDevelopmentRole-**spaceName**서비스 역할 이해.

   > **Note**
   >
   > 이 역할은 개발자 계정에만 사용하는 것이 좋으며 `AdministratorAccess` AWS 관리형 정책을 사용하므로 이 계정에서 새 정책 및 리소스를 만들 수 있는 전체 액세스 권한을 AWS 계정부여합니다.

7. 개발 역할 생성을 선택합니다.
8. 연결 페이지의 사용 가능한 IAM 역할 아래에서 계정에 추가된 `CodeCatalystWorkflowDevelopmentRole-spaceName` IAM 역할 목록의 역할을 확인합니다. CodeCatalyst
9. 숙소로 돌아가려면 Amazon으로 이동을 선택하십시오 CodeCatalyst.

AWSRoleForCodeCatalystSupport 역할

에서 원클릭 역할로 지원 역할을 생성합니다. IAM 계정을 추가하려는 스페이스에 스페이스 관리자 또는 고급 사용자 역할이 있어야 합니다. 또한 AWS 계정 추가하려는 항목에 대한 관리자 권한이 있어야 합니다.

아래 절차를 시작하기 전에 CodeCatalyst 스페이스에 추가하려는 것과 동일한 계정으로 로그인해야 합니다. AWS Management Console 그렇지 않으면 콘솔에서 알 수 없는 계정 오류가 반환됩니다.

생성 및 추가하기 CodeCatalyst AWSRoleForCodeCatalystSupport

1. CodeCatalyst 콘솔에서 시작하기 전에 를 AWS Management Console열고 AWS 계정 스페이스에 동일한 계정으로 로그인했는지 확인하십시오.
2. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.

3. 역할을 생성하려는 AWS 계정 위치의 링크를 선택합니다. AWS 계정 세부 정보 페이지가 표시됩니다.
4. 에서 역할 관리를 선택합니다 AWS Management Console.

   Amazon CodeCatalyst 스페이스에 IAM 역할 추가 페이지가 에서 열립니다 AWS Management Console. 아마존 CodeCatalyst 스페이스 페이지입니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.
5. CodeCatalyst 스페이스 세부 정보에서 CodeCatalyst Support 역할 추가를 선택합니다. 이 옵션은 미리 보기 개발 역할에 대한 권한 정책 및 신뢰 정책을 포함하는 서비스 역할을 생성합니다. 역할에는 고유 식별자가 AWSRoleForCodeCatalystSupport추가된 이름이 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오AWSRoleForCodeCatalystSupport서비스 역할 이해.
6. CodeCatalyst Support용 역할 추가 페이지에서 기본값을 선택된 상태로 두고 역할 생성을 선택합니다.
7. 사용 가능한 IAM 역할에서 계정에 추가된 IAM 역할 목록에서 역할을 확인하십시오. CodeCatalyst `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*
8. 숙소로 돌아가려면 Amazon으로 이동을 선택하십시오 CodeCatalyst.

IAM역할 생성 및 CodeCatalyst 신뢰 정책 사용

IAM AWS 계정 연결에 CodeCatalyst 사용할 역할은 여기에 제공된 신뢰 정책을 사용하도록 구성해야 합니다. 다음 단계를 사용하여 IAM 역할을 만들고 의 블루프린트에서 CodeCatalyst 프로젝트를 생성할 수 있는 정책을 첨부하십시오.

또는 역할에 대한 권한 정책 및 신뢰 정책을 포함하는 서비스 역할을 생성할 수 있습니다. `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 자세한 내용은 계정 연결에 IAM 역할 추가 단원을 참조하십시오.

1. 에 AWS Management Console 로그인하고 에서 IAM 콘솔을 엽니다 https://console.aws.amazon.com/iam/.
2. 역할을 선택한 다음 역할 생성을 선택합니다.
3. 사용자 지정 신뢰 정책을 선택합니다.
4. 사용자 지정 신뢰 정책 양식에서 다음 신뢰 정책을 붙여넣습니다.

```
"Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
             "Principal": {
                "Service": [
                    "codecatalyst-runner.amazonaws.com",
                    "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:codecatalyst:::space/spaceId/project/
*"
                }
            }
        }
    ]
```

5. Next(다음)를 선택합니다.
6. 권한 추가에서 이미 만든 사용자 지정 정책을 검색하여 선택합니다IAM.
7. Next(다음)를 선택합니다.
8. 역할 이름에 역할 이름을 입력합니다. 예를 들면 다음과 같습니다. `codecatalyst-project-role`
9. 역할 생성을 선택합니다.
10. Amazon 리소스 이름 (ARN) 역할을 복사합니다. 계정 연결 또는 환경에 역할을 추가할 때 이 정보를 제공해야 합니다.

사용자에게 공간 권한 부여

스페이스에 참여하는 사용자의 역할을 보거나, 추가, 제거 또는 변경하여 스페이스 구성원을 관리할 수 있습니다.

이 안내서의 정보는 AWS Builder ID 사용자를 CodeCatalyst 지원하는 스페이스에서 사용자를 초대하고 관리하기 위해 제공됩니다. ID 페더레이션을 지원하는 스페이스를 설정하고 관리하는 단계에 대한 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 CodeCatalyst 스페이스 설정 및 관리를 참조하십시오.

스페이스의 구성원 보기

표시 이름, 별칭, 스페이스에서 수행하는 역할에 대한 정보를 포함하여 스페이스에 있는 사용자를 볼 수 있습니다. 스페이스 구성원의 역할은 다음과 같이 세 가지입니다.

- 스페이스 관리자 - 이 역할에는 프로젝트 생성을 CodeCatalyst 비롯한 모든 권한이 있습니다. 스페이스의 모든 프로젝트에 액세스하는 등 스페이스의 모든 측면을 관리해야 하는 사용자에게만 이 역할을 할당하십시오.

  사용자를 먼저 제거하지 않으면 나중에 이 역할을 변경할 수 없습니다. 자세한 내용은 스페이스 관리자 역할 단원을 참조하십시오.
- 고급 사용자 — 이 역할은 Amazon CodeCatalyst 스페이스에서 두 번째로 강력한 역할이지만 스페이스의 프로젝트에 대한 액세스 권한은 없습니다. 스페이스에서 프로젝트를 생성하고 해당 스페이스의 사용자와 리소스를 관리하는 데 도움이 되어야 하는 사용자를 위해 설계되었습니다. 자세한 내용은 파워 유저 역할 단원을 참조하십시오.
- 제한된 액세스 — 스페이스의 프로젝트 초대를 수락하여 스페이스에 참여하는 사용자에게 기본적으로 이 역할이 할당됩니다. 프로젝트 멤버에게는 프로젝트 내 역할이 할당됩니다. 프로젝트 멤버 관리에 대한 자세한 내용은 을 참조하십시오사용자에게 프로젝트 권한 부여.

스페이스 관리자 테이블에는 스페이스 관리자 역할을 가진 사용자가 표시됩니다. 이러한 사용자는 스페이스의 모든 프로젝트에 자동으로 (암시적으로) 할당되고 프로젝트에서 역할이 없으므로 스페이스 구성원에 표시되지 않습니다.

스페이스 구성원 테이블에는 프로젝트에서 역할을 갖고 있지만 스페이스 관리자 역할은 없는 스페이스 내 모든 구성원이 표시됩니다.

사용자는 다음과 같이 사용자에게 스페이스 관리자 역할이 있는지 여부에 따라 표시됩니다. CodeCatalyst

- 스페이스 관리자 역할을 가진 사용자가 나중에 프로젝트 초대와 역할을 수락하면 스페이스 아래의 스페이스 멤버 테이블이나 프로젝트 아래의 프로젝트 멤버 테이블에 표시되지 않습니다. 이러한 사용자는 두 위치의 스페이스 관리자 테이블에 계속 표시됩니다. 각 프로젝트에서 스페이스 관리자 역할을 가진 모든 사용자는 해당 프로젝트의 프로젝트 스페이스 관리자 테이블에 표시됩니다.
- 프로젝트 역할을 사용하여 참여하라는 프로젝트 초대를 수락한 사용자는 제한된 액세스 역할을 가진 스페이스에 추가됩니다. 사용자 역할이 나중에 스페이스 관리자 역할로 변경되지만 스페이스 멤버 테이블에서 스페이스 관리자 테이블로 이동되는 경우 프로젝트에서 사용자는 프로젝트 멤버 테이블에서 스페이스 관리자 테이블로 이동합니다.

스페이스의 사용자 및 역할을 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다.

> **Tip**
> 둘 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택하세요.

3. 설정을 선택한 다음 구성원을 선택합니다.

   스페이스 구성원인 사용자는 스페이스 구성원 테이블에 표시됩니다.

> **Tip**
> 스페이스 관리자 역할을 가진 경우 자신이 직접 초대된 프로젝트를 볼 수 있습니다. 프로젝트의 프로젝트 설정으로 이동한 다음 내 프로젝트를 선택합니다.

   상태 열의 유효한 값은 다음과 같습니다.

   - 초대됨 - 초대를 CodeCatalyst 보냈지만 사용자가 아직 수락하거나 거부하지 않았습니다.
   - 회원 - 사용자가 초대를 수락했습니다.

스페이스에 사용자 직접 초대하기

사용자를 CodeCatalyst 스페이스에 직접 초대할 수 있습니다. 이는 스페이스 관리자 또는 고급 사용자 역할을 할당하여 공간 관리를 도와줄 사용자를 초대하려는 경우에 유용합니다. 이러한 역할 중 하나를 다른 사용자에게 할당하면 이러한 사용자를 프로젝트에 초대하지 않고도 더 많은 사람들에게 공간 관리 책임을 분산할 수 있습니다.

> **Note**
> 구성원을 초대하려면 스페이스 관리자 또는 고급 사용자 역할이 있어야 합니다.

스페이스 관리자 표에는 스페이스 관리자 역할을 가진 사용자가 표시됩니다. 이러한 사용자는 스페이스의 모든 프로젝트에 자동으로 (암시적으로) 할당되고 프로젝트에서 역할이 없으므로 스페이스 구성원 테이블에 표시되지 않습니다.

프로젝트 초대를 수락한 구성원은 기본적으로 스페이스에 추가됩니다. 프로젝트 멤버 테이블에는 스페이스에서 프로젝트에서 역할을 가진 모든 멤버가 표시됩니다.

초대를 수락하고 처음으로 로그인하는 방법에 대한 자세한 내용은 을 참조하십시오설정 및 로그인 CodeCatalyst.

스페이스에 사용자 초대하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다.
3. 설정을 선택한 다음 구성원을 선택합니다.
4. [Invite]를 선택합니다.
5. 스페이스에 초대하려는 사람의 이메일을 입력합니다. 역할에서 스페이스에서 해당 사용자에게 할당할 역할을 선택합니다.
6. 초대를 선택합니다.

스페이스 초대 취소

최근에 보낸 스페이스에 참여하라는 초대를 취소하고 싶지만 아직 수락되지 않은 경우 취소할 수 있습니다.

스페이스 초대를 관리하려면 스페이스 관리자 또는 고급 사용자 역할이 있어야 합니다.

스페이스 구성원 초대를 취소하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다.

> **Tip**
> 둘 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택하세요.

3. 설정을 선택한 다음 구성원을 선택합니다.
4. 구성원이 초대됨 상태인지 확인하십시오.

> **Note**
>
> 아직 수락되지 않은 초대만 취소할 수 있습니다.

5. 초대된 구성원이 있는 행 옆의 옵션을 선택한 다음 초대 취소를 선택합니다.
6. 확인 창이 표시됩니다. 초대 취소를 선택하여 확인합니다.

스페이스 구성원의 역할 변경

스페이스 구성원에 할당된 역할을 변경할 수 있습니다. 스페이스 내 사용자 역할을 변경하려면 스페이스 관리자 역할이 있어야 합니다.

스페이스 관리자 테이블에는 스페이스 관리자 역할을 가진 사용자가 표시됩니다. 이러한 사용자는 스페이스의 모든 프로젝트에 자동으로 (암시적으로) 할당되므로 스페이스 구성원 테이블에 표시되지 않습니다.

스페이스에서 사용자의 역할을 변경하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다.

> **Tip**
>
> 두 개 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택하세요.

3. 설정을 선택한 다음 구성원을 선택합니다.
4. 스페이스 구성원 테이블에서 역할을 변경하려는 사용자를 선택합니다. 역할 변경을 선택합니다.

스페이스 구성원 제거

스페이스 리소스에 액세스할 필요가 없는 경우 스페이스 구성원을 제거할 수 있습니다. 스페이스에서 구성원을 제거하려면 스페이스 관리자 역할이 있어야 합니다.

스페이스 관리자 테이블에는 스페이스 관리자 역할을 가진 사용자가 표시됩니다. 이러한 사용자는 스페이스의 모든 프로젝트에 자동으로 (암시적으로) 할당되고 프로젝트에서 역할이 없으므로 스페이스 구성원 테이블에 표시되지 않습니다. 이 표에서는 스페이스 구성원을 직접 제거할 수만 있습니다.

프로젝트 멤버 테이블에서 사용자를 제거하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다.

> **Tip**
>
> 두 개 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택하세요.

3. 설정을 선택한 다음 구성원을 선택합니다.
4. 프로젝트 멤버 테이블에서 사용자를 선택합니다. 제거를 선택합니다.

> **Note**
>
> 스페이스에서 구성원을 제거하면 해당 프로젝트의 리소스와 관련된 권한과 함께 스페이스의 모든 프로젝트에서 사용자가 제거됩니다.

스페이스 관리자 역할을 가진 사용자의 역할 제거 또는 변경

스페이스의 스페이스 관리자 역할을 가진 사용자의 역할을 제거하거나 변경할 수 있습니다.

스페이스 관리자 역할을 가진 사용자를 스페이스에서 제거하려면 스페이스 관리자 역할이 있어야 합니다. 스페이스 관리자 역할을 가진 사용자의 역할을 변경하면 기본적으로 해당 사용자가 스페이스 관리자 테이블에서 제거됩니다. 해당 사용자에게 스페이스의 어떤 프로젝트에서도 프로젝트 역할이 없는 경우 사용자에서 스페이스 관리자 역할을 제거하면 사용자가 스페이스에서 제거됩니다.

> **Note**
>
> 스페이스 관리자 역할을 가진 사용자는 자신을 제거할 수 없습니다. 스페이스 관리자 역할을 가진 다른 사용자에게 문의하세요.

스페이스 구성원 테이블에서 스페이스 관리자 역할을 가진 사용자를 제거하려면

> **Note**
>
> 프로젝트에 명시적으로 추가되지 않은 사용자의 경우 프로젝트 역할 (프로젝트 관리자 또는 기여자) 이 없습니다. 스페이스 관리자 역할이 사용자의 유일한 역할인 경우 사용자는 스페이스에서 완전히 제거됩니다.

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스 관리자 역할을 가진 사용자의 역할을 제거하거나 변경하려는 스페이스로 이동합니다.
3. 설정을 선택한 다음 구성원을 선택합니다.
4. 구성원 목록의 초대 상태를 확인하고, 목록에 스페이스로 보류 중인 승인되지 않은 초대가 없는지 확인합니다 (상태는 초대됨).

   > **Important**
   >
   > 스페이스 관리자 역할을 가진 사용자를 제거하기 전에 보류 중인 초대가 시작되지 않았는지 확인해야 합니다.

5. 구성원 탭을 선택합니다. 스페이스 관리자 테이블에서 사용자를 선택한 다음 제거를 선택합니다.

   구성원 제거 대화 상자에서 다음 중 하나를 수행하십시오.

   - 사용자의 스페이스 관리자 역할만 제거하는 옵션을 선택합니다. 제거를 선택합니다.

     > **Important**
     >
     > 사용자에게 다른 역할이 할당되지 않은 경우 스페이스 관리자에서 역할을 변경하면 사용자가 스페이스에서 제거됩니다.

   - 스페이스 관리자 역할을 가진 사용자를 스페이스 및 모든 프로젝트에서 제거하는 옵션을 선택합니다. 제거를 선택합니다.

6. 구성원 탭을 새로 고칩니다. 사용자가 프로젝트 역할을 통해 멤버십을 가졌던 모든 프로젝트의 프로젝트 멤버 목록에 자동으로 추가됩니다. 스페이스 관리자 역할이 사용자의 유일한 역할인 경우 사용자는 스페이스에서 완전히 제거됩니다.

팀을 통한 공간 액세스 허용

스페이스를 만든 후 팀을 추가할 수 있습니다. 팀을 사용하면 사용자를 그룹화하여 권한을 공유하고 프로젝트, 이슈 추적, 역할 및 리소스를 관리할 수 CodeCatalyst 있습니다.

팀을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

팀은 프로젝트/스페이스 수준에서도 관리됩니다. CodeCatalyst 스페이스/프로젝트의 팀에 대한 자세한 내용은 을 참조하십시오. 팀을 통한 공간 액세스 허용

주제

팀 만들기

팀은 스페이스에서 파워 사용자와 같은 역할 권한을 가질 수 있습니다. 또한 팀은 프로젝트에 대한 프로젝트 권한 (예: 프로젝트 관리자) 을 가질 수 있습니다. 팀은 프로젝트마다 다른 역할을 가진 많은 프로젝트에 연결될 수 있습니다. 팀 구성원이 AWS Builder ID 스페이스의 경우 개별 사용자이거나 ID 페더레이션을 지원하는 스페이스의 경우 SSO 그룹으로 구성된 팀을 관리할 수 있습니다.

스페이스 및 프로젝트 사용자의 멤버 페이지에서 사용자는 여러 역할을 가질 수 있습니다. 여러 역할을 가진 사용자는 역할을 여러 개 가지고 있을 때 표시기가 표시되며, 가장 많은 권한을 가진 역할부터 표시됩니다.

> **Note**
>
> 스페이스가 ID 페더레이션을 지원하는 경우 Identity Center에 IAM 이미 SSO 사용자 또는 SSO 그룹이 설정되어 있어야 합니다.

팀 구성원을 관리하는 방법은 사용자를 추가하고 제거하는 방법에 따라 달라집니다. 팀원을 관리하는 데는 두 가지 옵션이 있습니다.

- 사용자 직접 추가 — 사용자를 개별적으로 추가하거나 제거합니다. 예를 들어 AWS Builder ID 사용자 또는 IAM Identity Center에 이미 설정된 SSO 사용자를 선택하여 팀에 사용자를 추가합니다. AWS Builder ID 사용자 또는 SSO 사용자를 직접 추가하여 팀 구성원을 관리하도록 선택하면 SSO 그룹을 사용하는 옵션을 더 이상 사용할 수 없습니다.
- SSO그룹 사용 — IAM Identity Center에 이미 설정된 SSO 그룹을 통해 팀 구성원을 관리합니다. SSO그룹을 사용하여 팀 구성원을 관리하기로 선택하면 사용자를 직접 추가하는 옵션을 더 이상 사용할 수 없습니다.

팀을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

팀을 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다. 설정을 선택한 다음 팀을 선택합니다.
3. 팀 만들기를 선택합니다.
4. 팀 이름에 팀을 설명하는 이름을 입력합니다.

   > **Note**
   >
   > 팀 이름은 스페이스에서 고유해야 합니다.

   (선택 사항) 팀 설명에 팀에 대한 설명을 입력합니다.
5. 스페이스 역할에서 사용 가능한 스페이스 역할 목록에서 팀에 CodeCatalyst 할당하려는 역할을 선택합니다. 팀의 모든 구성원이 역할을 상속합니다.

   - 스페이스 관리자 - 자세한 내용은 을 참조하십시오. 스페이스 관리자 역할
   - 제한된 액세스 - 자세한 내용은 을 참조하십시오제한된 액세스 역할.
   - 고급 사용자 - 자세한 내용은 을 참조하십시오파워 유저 역할.
6. 팀 멤버십에서 다음 중 하나를 선택하여 팀에 구성원을 추가하는 방법을 선택합니다.

   - 사용자를 개별적으로 관리하려면 구성원 직접 추가를 선택합니다. 여기에는 스페이스에 AWS Builder ID 사용자를 추가하거나 ID 페더레이션을 지원하는 스페이스에 SSO 사용자를 추가하는 작업이 포함됩니다.

- IAMIdentity Center에서 이미 설정한 SSOSSO그룹을 선택하려면 그룹 사용을 선택합니다.

  SSO그룹에서 추가하려는 그룹 옆의 상자를 선택합니다. SSO그룹은 최대 5개까지 추가할 수 있습니다.

> **Note**
>
> 나중에 변경할 수 없습니다. AWS Builder ID 사용자 또는 SSO 사용자를 직접 추가하여 팀 구성원을 관리하도록 선택하면 SSO그룹 사용 옵션을 더 이상 사용할 수 없습니다. SSO 그룹을 사용하여 팀원을 관리하기로 선택하면 사용자를 직접 추가하는 옵션을 더 이상 사용할 수 없습니다.

7. 생성(Create)을 선택합니다.

> **Note**
>
> SSO그룹을 사용하기로 선택한 경우 팀을 만들 때 SSO 그룹 내 사용자는 삭제되지 않는다는 점에 유의하세요. 사용자가 목록에 표시되려면 CodeCatalyst 먼저 로그인해야 합니다.

팀 보기

CodeCatalyst에서는 팀의 프로젝트와 역할을 볼 수 있습니다. 멤버 페이지에서 프로젝트 역할과 사용자 목록을 볼 수 있습니다. SSO그룹 유형 팀의 경우 팀과 관련된 SSO 그룹 목록도 볼 수 있습니다.

팀을 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다. 설정을 선택한 다음 팀을 선택합니다.
3. 스페이스 역할에서 이 스페이스에 대해 팀에 할당된 역할을 확인합니다.
4. 프로젝트 역할 탭에서 팀이 구성원으로 추가된 스페이스의 각 프로젝트에 대해 팀에 할당된 CodeCatalyst 프로젝트 및 프로젝트 역할을 확인할 수 있습니다 (AWS Builder ID 스페이스만 해당).
5. 멤버 탭에서 팀에 배정된 멤버의 목록을 볼 수 있습니다.
6. SSO그룹 탭에서 팀에 할당된 SSO 그룹 목록을 볼 수 있습니다 (ID 페더레이션만 지원하는 스페이스의 경우).

팀에 스페이스 역할 부여

팀은 사용자를 그룹화하여 에서 CodeCatalyst 프로젝트에 대한 팀 액세스 권한을 부여하고 관리할 수 있는 방법입니다. 예를 들어 팀을 사용하면 팀에 사용자를 위한 공간을 관리할 수 있는 권한을 부여하여 사용자의 역할과 권한을 신속하게 관리할 수 있습니다.

팀은 스페이스에서 고급 사용자와 같은 역할 권한을 가질 수 있습니다. 팀의 스페이스 역할을 변경할 수 있지만 팀의 모든 구성원이 해당 권한을 상속한다는 점에 유의하세요.

팀을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

팀의 스페이스 역할 변경

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다. 설정을 선택한 다음 팀을 선택합니다.
3. [작업] 에서 스페이스 역할 변경을 선택합니다. 스페이스 역할을 다음 중 하나로 변경할 수 있습니다. 이렇게 하면 모든 팀 구성원의 역할이 변경됩니다.
   - 스페이스 관리자 - 자세한 내용은 을 참조하십시오스페이스 관리자 역할.
   - 제한된 액세스 - 자세한 내용은 을 참조하십시오제한된 액세스 역할.
   - 고급 사용자 - 자세한 내용은 을 참조하십시오파워 유저 역할.
4. 저장(Save)을 선택합니다.

스페이스 수준에서 팀에 프로젝트 역할 부여

의 CodeCatalyst 팀은 팀 구성원이 프로젝트에서 프로젝트 관리자와 같은 역할 권한을 가질 수 있다는 점에서 사용자와 비슷합니다. 역할 변경이 팀에 적용되고 팀의 모든 구성원이 해당 권한을 상속합니다. 팀에 자동으로 부여되는 역할을 각 프로젝트에 대해 하나씩 선택할 수 있습니다.

팀을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

프로젝트 역할 추가 또는 변경하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다. 설정을 선택한 다음 팀을 선택합니다.
3. 프로젝트 역할 탭을 선택합니다.

4. 역할을 변경하려면 이 목록에서 프로젝트 옆에 있는 선택기를 선택한 다음 역할 변경을 선택합니다. 역할을 추가하려면 프로젝트 역할 추가를 선택합니다. 프로젝트에서 추가하려는 프로젝트를 선택하고 역할에서 역할을 선택합니다. 사용 가능한 프로젝트 역할 중 하나를 선택합니다.

   - 프로젝트 관리자 - 자세한 내용은 을 참조하십시오프로젝트 관리자 역할.
   - 기여자 - 자세한 내용은 을 참조하십시오기여자 역할.
   - 리뷰어 - 자세한 내용은 을 참조하십시오. 리뷰어 역할
   - 읽기 전용 - 자세한 내용은 을 참조하십시오읽기 전용 역할.

5. 저장(Save)을 선택합니다.

프로젝트 역할을 제거하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다. 설정을 선택한 다음 팀을 선택합니다.
3. 프로젝트 역할 탭을 선택합니다.
4. 제거하려는 역할을 선택합니다.

> **Important**
>
> 팀에서 역할을 제거하면 팀 내 모든 사용자의 관련 권한이 제거됩니다.

5. 저장(Save)을 선택합니다.

팀에 사용자 직접 추가

팀에 팀원을 추가할 수 있습니다. 사용자를 추가하면 새 사용자는 팀의 모든 기존 역할로부터 권한을 상속받게 됩니다.

스페이스가 AWS Builder ID 사용자 지원 또는 ID 페더레이션을 위해 설정되었든 관계없이 사용자를 직접 추가하도록 공간을 설정할 수 있습니다.

> **Note**
>
> SSO그룹을 사용하여 팀 구성원을 관리하도록 공간을 설정한 경우 사용자 직접 추가를 사용하는 옵션을 사용할 수 없습니다. SSO그룹을 사용하려면 을 참조하십시오팀에 SSO 그룹 추가.

팀을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

사용자를 직접 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다. 설정을 선택한 다음 팀을 선택합니다.
3. 구성원 탭을 선택합니다.
4. 구성원 추가를 선택합니다.

> **Note**
>
> 팀에 추가되는 사용자는 이미 스페이스의 구성원이어야 합니다. 스페이스 구성원이 아닌 팀원을 추가하거나 초대할 수 없습니다.

5. 드롭다운 필드에서 사용자를 선택한 다음 저장을 선택합니다. AWS 빌더 ID 사용자 또는 IAM Identity Center에 이미 설정된 SSO 사용자를 선택합니다.

팀에서 직접 사용자 제거

팀에서 팀원을 제거할 수 있습니다. 사용자는 더 이상 모든 권한을 상속하지 않습니다. 나중에 사용자를 팀에 다시 추가할 수 있습니다.

> **Note**
>
> 팀 구성원을 제거하면 스페이스의 모든 프로젝트 및 리소스에서 해당 사용자에 대한 관련 권한이 제거됩니다.

팀을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

팀원을 제거하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다. 설정을 선택한 다음 팀을 선택합니다.
3. 구성원 탭을 선택합니다.
4. 제거하려는 사용자 옆의 선택기를 선택한 다음 제거를 선택합니다.
5. 입력 필드에 remove 를 입력한 다음 제거를 선택합니다.

팀에 SSO 그룹 추가

공간이 IAM Identity Center에서 관리되는 SSO 사용자 및 그룹이 있는 공간으로 구성된 경우 스페이스에 별도의 팀으로 참여할 SSO 그룹을 추가할 수 있습니다.

> **Note**
>
> AWS Builder ID 사용자 또는 SSO 사용자를 직접 추가하여 팀 구성원을 관리하도록 선택하면 SSO그룹 사용 옵션을 사용할 수 없습니다. 사용자를 직접 추가하려면 을 참조하십시오팀에 사용자 직접 추가.

팀을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

SSO그룹을 팀으로 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스 페이지에서 Teams를 선택합니다. SSO그룹 탭을 선택합니다.
3. 추가하려는 SSO 그룹을 선택합니다. SSO그룹을 5개까지 추가할 수 있습니다.

팀 삭제

더 이상 필요하지 않은 팀은 삭제할 수 있습니다.

> **Note**
>
> 팀을 삭제하면 스페이스의 모든 프로젝트 및 리소스에서 모든 팀 구성원의 관련 권한이 제거됩니다.

팀을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

팀 삭제

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다. 설정을 선택한 다음 팀을 선택합니다.
3. 작업에서 팀 삭제를 선택합니다. 이렇게 하면 팀 전체의 역할이 바뀝니다.

4. Delete(삭제)를 선택합니다.

시스템 리소스에 대한 공간 액세스 허용

컴퓨터 리소스는 프로젝트 또는 스페이스에 대한 권한이 부여된 특정 리소스입니다 CodeCatalyst. CodeCatalyst

> **Note**
>
> 머신 리소스라는 용어는 Amazon EC2 인스턴스와 같은 클라우드 인프라를 가리키는 것이 아니라 공간 또는 프로젝트에 대한 권한이 있는 블루프린트 또는 워크플로 리소스를 가리킵니다.

머신 리소스는 승인된 리소스가 액세스 시 CodeCatalyst SSO 획득한 사용자 ID를 나타냅니다. 머신 리소스는 블루프린트 및 워크플로와 같은 공간 내 리소스에 권한을 부여하는 데 사용됩니다. 스페이스의 컴퓨터 리소스를 볼 수 있으며 공간에 대한 컴퓨터 리소스를 활성화하거나 비활성화하도록 선택할 수 있습니다. 예를 들어 액세스를 관리하기 위해 컴퓨터 리소스를 사용하지 않도록 설정했다가 나중에 다시 사용하도록 설정하는 것이 좋습니다.

이러한 작업은 컴퓨터 리소스를 해지하거나 사용하지 않도록 설정해야 하는 경우 컴퓨터 리소스에 사용할 수 있습니다. 예를 들어 자격 증명이 손상되었을 것으로 의심되는 경우 컴퓨터 리소스를 사용하지 않도록 설정할 수 있습니다. 일반적으로 이러한 작업은 사용할 필요가 없습니다.

이 페이지를 보고 스페이스 수준에서 컴퓨터 리소스를 관리하려면 스페이스 관리자 역할이 있어야 합니다.

컴퓨터 리소스는 에서도 프로젝트 수준에서 CodeCatalyst 관리됩니다. 프로젝트 내 팀에 대해 자세히 알아보려면 을 참조하십시오시스템 리소스에 대한 공간 액세스 허용 .

주제

- 컴퓨터 리소스에 대한 공간 액세스 보기
- 컴퓨터 리소스에 대한 공간 액세스 비활성화
- 시스템 리소스에 대한 공간 액세스 활성화

컴퓨터 리소스에 대한 공간 액세스 보기

스페이스에서 사용 중인 컴퓨터 리소스 목록을 볼 수 있습니다.

컴퓨터 리소스를 관리하려면 스페이스 관리자 역할이 있어야 합니다.

컴퓨터 리소스를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동한 다음 설정을 선택합니다. 컴퓨터 리소스를 선택합니다.
3. 드롭다운에서 워크플로 작업을 선택하면 워크플로의 컴퓨터 리소스만 볼 수 있습니다. 블루프린트의 머신 리소스만 보려면 블루프린트를 선택합니다.

   필터 필드를 사용하여 이름을 필터링할 수도 있습니다.

컴퓨터 리소스에 대한 공간 액세스 비활성화

스페이스에서 사용 중인 컴퓨터 리소스를 비활성화하도록 선택할 수 있습니다.

> **Important**
>
> 컴퓨터 리소스를 비활성화하면 스페이스의 모든 관련 청사진 또는 워크플로에 대한 권한이 모두 제거됩니다.

컴퓨터 리소스를 관리하려면 스페이스 관리자 역할이 있어야 합니다.

컴퓨터 리소스를 사용하지 않도록 설정하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동한 다음 설정을 선택합니다. 컴퓨터 리소스를 선택합니다.
3. 다음 중 하나를 선택합니다.

   > **Important**
   >
   > 머신 리소스를 비활성화하면 스페이스의 모든 관련 블루프린트 또는 워크플로에 대한 모든 권한이 제거됩니다.

   - 개별적으로 비활성화하려면 비활성화하려는 하나 이상의 머신 리소스 옆에 있는 선택기를 선택합니다. 비활성화를 선택한 다음 이 리소스를 선택합니다.
   - 모든 리소스를 비활성화하려면 비활성화를 선택한 다음 모든 리소스를 선택합니다.

- 모든 워크플로 작업을 비활성화하려면 비활성화를 선택한 다음 모든 워크플로 작업을 선택합니다.
- 블루프린트를 모두 비활성화하려면 비활성화를 선택한 다음 모든 블루프린트를 선택합니다.

시스템 리소스에 대한 공간 액세스 활성화

스페이스에서 사용 중이지만 비활성화된 컴퓨터 리소스를 활성화하도록 선택할 수 있습니다.

컴퓨터 리소스를 관리하려면 스페이스 관리자 역할이 있어야 합니다.

컴퓨터 리소스를 활성화하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동한 다음 설정을 선택합니다. 컴퓨터 리소스를 선택합니다.
3. 다음 중 하나를 선택합니다.

   - 개별적으로 활성화하려면 활성화하려는 하나 이상의 컴퓨터 리소스 옆에 있는 선택기를 선택합니다. [활성화] 를 선택한 다음 [이 리소스] 를 선택합니다.
   - 모든 리소스를 활성화하려면 활성화를 선택한 다음 모든 리소스를 선택합니다.
   - 모든 워크플로 작업을 활성화하려면 활성화를 선택한 다음 모든 워크플로 작업을 선택합니다.
   - 모든 블루프린트를 활성화하려면 활성화를 선택한 다음 모든 블루프린트를 선택합니다.

스페이스의 개발 환경 관리

모든 개발 환경은 스페이스 내 프로젝트의 일부로 생성됩니다. 스페이스 구성원은 소스 리포지토리 수준에서 프로젝트 내에 자체 개발 환경을 만들 수 있습니다. 그러면 스페이스 관리자는 Amazon CodeCatalyst 콘솔을 사용하여 스페이스 구성원을 대신하여 개발 환경을 보고, 편집하고, 삭제하고, 중지할 수 있습니다. 간단히 말해, 공간 관리자는 공간 수준에서 개발 환경을 유지 관리합니다.

개발 환경 관리를 위한 고려 사항

- 설정에서 개발 환경 페이지를 보고 스페이스 수준에서 개발 환경을 관리하려면 스페이스 관리자 역할이 있어야 합니다.
- 스페이스 구성원은 자신의 계정을 통해 프로젝트에서 만든 개발 환경을 관리합니다. CodeCatalyst 스페이스 관리자로 개발 환경을 관리할 때는 스페이스 구성원을 대신하여 이러한 리소스를 유지 관리해야 합니다.

- 개발 환경은 기본적으로 특정 컴퓨팅 및 스토리지 구성을 사용합니다. 구성 업그레이드에 대한 청구 및 요금에 대한 자세한 내용은 Amazon CodeCatalyst 요금 페이지를 참조하십시오.

> **Important**
>
> Active Directory가 ID 공급자로 사용되는 공간의 사용자는 개발 환경을 사용할 수 없습니다. 자세한 내용은 SSO (Single Sign-On) 계정을 CodeCatalyst 사용하여 로그인한 상태에서는 개발 환경을 만들 수 없습니다. 단원을 참조하십시오.

인스턴스 실행 중지, 기본 컴퓨팅 구성, 컴퓨팅 업그레이드, 비용 발생, 제한 시간 구성 등 개발 환경에 대한 기타 고려 사항은 을 참조하십시오. 의 개발 환경을 사용하여 코드 작성 및 수정 CodeCatalyst

주제

- 해당 공간의 개발 환경 보기
- 공간에 맞는 개발 환경 편집하기
- 스페이스를 위한 개발 환경 중단하기
- 스페이스의 개발 환경 삭제

해당 공간의 개발 환경 보기

공간에 있는 모든 개발 환경의 유형, 상태 및 세부 정보를 볼 수 있습니다. 개발 환경 생성 및 실행에 대한 자세한 내용은 을 참조하십시오. Dev Environment 생성

이 페이지를 보고 스페이스 수준에서 개발 환경을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

스페이스의 개발 환경을 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.

   > **Tip**
   >
   > 둘 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택합니다.

3. 설정을 선택한 다음 개발 환경을 선택합니다.

이 페이지에는 해당 공간의 모든 개발 환경이 나열됩니다. 리소스 이름, 해당하는 경우 리소스 별칭, 유형, 기본 또는 구성된 컴퓨팅 및 스토리지 IDE, 각 개발 환경에 대해 구성된 제한 시간을 볼 수 있습니다.

공간에 맞는 개발 환경 편집하기

유휴 개발 환경의 실행을 중지하도록 구성된 제한 시간 (있는 경우) 과 같은 개발 환경의 구성을 편집할 수 있습니다. 개발 환경 편집에 대한 자세한 내용은 을 참조하십시오. 개발 환경 편집

이 페이지를 보고 스페이스 수준에서 개발 환경을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

스페이스의 개발 환경을 편집하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.

> **Tip**
>
> 둘 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택합니다.

3. 설정을 선택한 다음 개발 환경을 선택합니다.
4. 관리하려는 개발 환경 옆의 선택기를 선택합니다. 편집을 선택합니다.
5. 개발 환경의 컴퓨팅 또는 비활성 제한 시간을 원하는 대로 변경합니다.
6. 저장(Save)을 선택합니다.

스페이스를 위한 개발 환경 중단하기

개발 환경에 타임아웃이 있도록 구성된 경우 유휴 상태가 되기 전에 실행 중인 개발 환경을 중지할 수 있습니다. 그렇지 않으면 제한 시간이 경과된 개발 환경은 이미 중지됩니다. 개발 환경 중지에 대한 자세한 내용은 을 참조하십시오. 개발 환경 중지

이 페이지를 보고 스페이스 수준에서 개발 환경을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

스페이스의 개발 환경을 중지하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.

> **Tip**
>
> 둘 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택합니다.

3. 설정을 선택한 다음 개발 환경을 선택합니다.
4. 관리하려는 개발 환경 옆의 선택기를 선택합니다. 중지를 선택합니다.

스페이스의 개발 환경 삭제

더 이상 필요하지 않거나 더 이상 소유자가 없는 개발 환경을 삭제할 수 있습니다. 개발 환경 삭제 고려 사항에 대한 자세한 내용은 을 참조하십시오. Dev Environment 삭제

이 페이지를 보고 스페이스 수준에서 개발 환경을 관리하려면 스페이스 관리자 역할이 있어야 합니다.

스페이스의 개발 환경을 삭제하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.

> **Tip**
>
> 둘 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택합니다.

3. 설정을 선택한 다음 개발 환경을 선택합니다.
4. 관리하려는 개발 환경 옆의 선택기를 선택합니다. Delete(삭제)를 선택합니다. 확인하려면 를 입력한 다음 **`delete`** 삭제를 선택합니다.

공간 할당량

다음 표에는 Amazon의 공간 할당량 및 한도가 설명되어 있습니다. CodeCatalyst CodeCatalystAmazon의 할당량에 대한 자세한 내용은 을 참조하십시오. 에 대한 할당량 CodeCatalyst

스페이스의 최대 슬랙 채널 수	500
이메일 주소의 최대 초대 수	25

한 사용자에 대한 최대 초대 수	500
사용자당 최대 활성 공간 수 AWS 리전	5
사용자당 월별 지역별 최대 공간 생성 수	5
한 팀의 최대 SSO 그룹 수	5
스페이스의 최대 팀 수	100
한 팀의 최대 사용자 수	1000
스페이스 설명	스페이스 설명은 선택 사항입니다. 지정하는 경우 길이는 0~200자 사이여야 합니다. 문자, 숫자, 공백, 마침표, 밑줄, 쉼표, 대시 및 다음 특수 문자의 조합을 포함할 수 있습니다. `? & $ % + = / \ ; : \n \t \r`
스페이스 이름	스페이스 이름은 전체에서 고유해야 CodeCatalyst 합니다. 삭제된 스페이스의 이름은 재사용할 수 없습니다. 스페이스 이름의 길이는 3~63자 사이여야 합니다. 또한 영숫자로 시작해야 합니다. 스페이스 이름에는 문자, 숫자, 마침표, 밑줄 및 대시를 조합하여 사용할 수 있습니다. 다음 문자를 포함할 수 없습니다. `! ? @ # $ % ^ & * ( ) + = { } [ ] | 八 > < ~ ` ' " ; :`

프로젝트 관련 작업을 체계적으로 정리하세요 CodeCatalyst

CodeCatalyst Amazon의 프로젝트를 사용하여 개발팀이 공유된 지속적 통합/지속적 전달 (CI/CD) 워크플로 및 리포지토리로 개발 작업을 수행할 수 있는 협업 공간을 구축합니다. 프로젝트를 생성할 때 리소스를 추가, 업데이트 또는 제거할 수 있습니다. 또한 팀 작업의 진행 상황을 모니터링할 수 있습니다. 스페이스 내에 여러 프로젝트를 포함할 수 있습니다.

의 CodeCatalyst 스페이스는 프로젝트로 구성되어 있습니다. 스페이스 내의 모든 프로젝트를 볼 수 있지만 자신이 멤버로 속해 있는 프로젝트만 사용할 수 있습니다. 프로젝트를 만들면 프로젝트의 기본 역할이 생성되며, 프로젝트에 초대된 사용자에게 이 역할을 할당합니다.

- 기여자 역할과 같은 프로젝트 역할을 가진 프로젝트에 배정된 사람은 누구나 소스 리포지토리와 같은 프로젝트 리소스에 액세스할 수 있습니다.
- 스페이스 관리자, 프로젝트 관리자 또는 역할을 가진 사람은 누구나 프로젝트 참여 초대를 보낼 수 있습니다.
- 프로젝트 관리자 역할을 가진 사용자는 공유 리소스의 활동, 상태 및 기타 설정을 추적할 수 있습니다.
- 제한된 액세스 역할을 가진 사용자는 CI/CD 워크플로의 일부로 기능, 코드 수정 및 테스트에 대한 프로젝트 할당을 관리할 수 있습니다.

  워크플로는 애플리케이션을 CI/CD 파이프라인으로 빌드, 테스트, 릴리스 또는 업데이트하는 데 사용됩니다. 소스 아티팩트를 전송하고 작업하는 작업을 추가하여 워크플로를 조합할 수 있습니다. 작업을 실행하면 프로젝트 클라우드 리소스가 워크플로 작업에 온디맨드 컴퓨팅 기능을 제공하는 데 사용됩니다. 설정하려는 활동 및 출력을 기반으로 더 많은 CI/CD 워크플로를 구성할 수 있습니다. 예를 들어, 빌드 및 테스트 작업 전용 워크플로를 만들어 테스트 결과를 보고 버그를 수정하면서 배포 없이 워크플로를 완료할 수 있습니다. 그런 다음 다른 워크플로를 만들어 애플리케이션을 빌드하고 스테이징 환경에 배포할 수 있습니다.

프로젝트를 만들 때 블루프린트를 사용하여 샘플 코드를 포함하고 리소스를 생성하는 프로젝트를 만들거나 빈 프로젝트로 시작할 수 있습니다. 블루프린트를 사용하여 프로젝트를 생성하는 경우, 선택한 블루프린트에 따라 프로젝트에 추가할 리소스와 프로젝트 리소스를 추적하고 사용할 수 있도록 CodeCatalyst 생성 또는 구성하는 도구가 결정됩니다. 프로젝트를 만든 후 리소스를 수동으로 추가하거나 제거할 수 있습니다.

각 프로젝트는 프로젝트가 생성되거나 리소스가 수정되는 시기와 같은 사용자별 이벤트 목록으로 프로젝트 활동을 추적합니다. 프로젝트 활동은 공간 수준에서 모니터링 및 집계됩니다. 활동 데이터 작업에 대한 자세한 내용은 을 참조하십시오스페이스의 모든 프로젝트 보기.

프로젝트에서 AWS 리소스를 사용하는 경우 관리자 권한이 있는 CodeCatalyst 계정에 AWS 계정을 연결하여 프로젝트의 리소스를 통합할 수 있습니다.

프로젝트를 만든 후 소스 리포지토리, 이슈 및 기타 리소스를 프로젝트에 추가할 수 있습니다. 프로젝트를 만들려면 스페이스 관리자 역할이 있어야 합니다.

프로젝트 생성

CodeCatalyst 프로젝트를 사용하면 공유된 지속적 통합/지속적 전달 (CI/CD) 워크플로 및 리포지토리를 사용하여 개발 작업을 수행하고, 리소스를 관리하고, 문제를 추적하고, 사용자를 추가할 수 있습니다.

프로젝트를 생성하려면 먼저 스페이스 관리자 또는 고급 사용자 역할이 있어야 합니다.

주제

- Amazon에서 빈 프로젝트 생성 CodeCatalyst
- 연결된 타사 리포지토리를 사용하여 프로젝트 만들기
- 블루프린트로 프로젝트 생성
- Amazon Q를 사용하여 프로젝트를 생성하거나 블루프린트로 기능을 추가하는 모범 사례
- 프로젝트에 블루프린트를 사용하는 모범 사례
- 생성된 프로젝트에 리소스 및 작업 추가

Amazon에서 빈 프로젝트 생성 CodeCatalyst

리소스가 없는 빈 프로젝트를 만들고 나중에 원하는 리소스를 수동으로 추가할 수 있습니다.

프로젝트를 만들려면 먼저 스페이스 관리자 또는 고급 사용자 역할이 있어야 합니다.

빈 프로젝트를 만들려면

1. 프로젝트를 생성하려는 스페이스로 이동합니다.
2. 스페이스 대시보드에서 프로젝트 생성을 선택합니다.
3. 처음부터 시작을 선택합니다.

4. 프로젝트에 이름 부여에서 프로젝트에 할당할 이름을 입력합니다. 이름은 스페이스 내에서 고유해야 합니다.
5. 프로젝트 만들기를 선택합니다.

연결된 타사 리포지토리를 사용하여 프로젝트 만들기

프로젝트의 소스 코드를 선호하는 타사 제공업체에 보관하면서도 청사진, 라이프사이클 관리, 워크플로우 등과 같은 모든 CodeCatalyst 기능을 계속 사용할 수 있습니다. 이를 위해 GitHub 리포지토리, Bitbucket 리포지토리 또는 CodeCatalyst 프로젝트 리포지토리에 연결되는 새 GitLab 프로젝트를 만들 수 있습니다. 그런 다음 CodeCatalyst 프로젝트에서 링크된 소스 리포지토리를 사용할 수 있습니다.

CodeCatalyst 프로젝트를 만들려면 먼저 스페이스 관리자 또는 Power 사용자 역할이 있어야 합니다. 자세한 내용은 스페이스 만들기 및 스페이스에 사용자 직접 초대하기 단원을 참조하세요.

GitHub 계정의 소스 리포지토리로 CodeCatalyst 연결되는 링크를 포함하는 프로젝트를 만들려면 다음 세 가지 작업을 완료해야 합니다.

1. 리포지토리, GitHub Bitbucket 리포지토리 또는 리포지토리 확장을 설치합니다. GitLab 외부 사이트에 저장소에 연결하고 액세스 권한을 CodeCatalyst 제공하라는 메시지가 표시되며, 이 작업은 다음 단계의 일부로 수행됩니다.

> **Important**
>
> 스페이스에 GitHub 리포지토리, Bitbucket GitLab 리포지토리 또는 리포지토리 확장을 설치하려면 스페이스에서 CodeCatalyst 스페이스 관리자 역할을 가진 계정으로 로그인해야 합니다.

2. GitHub 계정이나 Bitbucket 작업 공간 또는 GitLab 사용자를 연결합니다 CodeCatalyst.

> **Important**
>
> GitHub 계정, Bitbucket 작업 공간, GitLab 사용자를 CodeCatalyst 공간에 연결하려면 타사 소스의 관리자이자 CodeCatalyst 스페이스 관리자여야 합니다.

> **Important**
>
> 리포지토리 확장을 설치한 후 연결하는 모든 리포지토리는 해당 코드를 인덱싱하여 CodeCatalyst 저장합니다. CodeCatalyst 그러면 코드를 검색할 수 있게 됩니다. CodeCatalyst 에서 CodeCatalyst 연결된 리포지토리를 사용할 때의 코드 데이터 보호를 더 잘 이해하려면 Amazon CodeCatalyst User Guide의 데이터 보호를 참조하십시오.

3. GitHub 리포지토리, Bitbucket 리포지토리 또는 CodeCatalyst GitLab 프로젝트 리포지토리에 연결된 프로젝트를 생성하십시오.

> **Important**
>
> 기여자로서 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 연결할 수 있지만, 타사 리포지토리는 스페이스 관리자 또는 프로젝트 관리자만 연결을 해제할 수 있습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

> **Important**
>
> CodeCatalyst 연결된 리포지토리의 기본 브랜치에서 변경 사항 감지를 지원하지 않습니다. 연결된 저장소의 기본 분기를 변경하려면 먼저 연결을 해제하고 기본 분기를 변경한 다음 다시 연결해야 합니다. CodeCatalyst 자세한 내용은 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, GitLab Jira 프로젝트 연결 CodeCatalyst 단원을 참조하십시오. 가장 좋은 방법은 리포지토리를 연결하기 전에 항상 최신 버전의 확장 프로그램을 사용하는 것입니다.

> **Note**
>
> - GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리는 스페이스의 한 CodeCatalyst 프로젝트에만 연결할 수 있습니다.
> - 비어 있거나 보관된 GitHub 리포지토리, Bitbucket 리포지토리 또는 프로젝트 리포지토리는 프로젝트와 함께 사용할 수 없습니다. GitLab CodeCatalyst

- 프로젝트의 리포지토리와 이름이 같은 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리는 연결할 수 없습니다. CodeCatalyst
- GitHub 리포지토리 확장은 GitHub 엔터프라이즈 서버 리포지토리와 호환되지 않습니다.
- Bitbucket 리포지토리 확장은 Bitbucket 데이터 센터 리포지토리와 호환되지 않습니다.
- 리포지토리 확장은 자체 관리형 GitLab 프로젝트 리포지토리와 호환되지 않습니다. GitLab
- 연결된 리포지토리에서는 나를 위한 설명 쓰기 또는 댓글 요약 기능을 사용할 수 없습니다. 이러한 기능은 풀 리퀘스트 인에서만 사용할 수 있습니다. CodeCatalyst

자세한 내용은 확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst 단원을 참조하십시오.

타사 확장 프로그램을 설치하려면

1. 프로젝트를 생성하려는 스페이스로 이동합니다.
2. 스페이스 대시보드에서 프로젝트 생성을 선택합니다.
3. 자체 코드 가져오기를 선택합니다.
4. 기존 리포지토리 연결에서 사용하려는 타사 GitHub 리포지토리 공급자에 따라 리포지토리, Bitbucket GitLab 리포지토리, 리포지토리를 선택합니다. 이전에 연결하지 않았다면 GitHub 계정, Bitbucket 작업 공간 또는 GitLab 계정을 연결하라는 메시지가 표시됩니다. 선택한 타사 확장 프로그램이 아직 설치되지 않은 경우 설치 메시지가 표시됩니다.
5. 메시지가 표시되면 설치를 선택합니다. 확장에 필요한 권한을 검토하고 계속하려면 설치를 다시 선택합니다.

타사 확장 프로그램을 설치한 후 다음 단계는 GitHub 계정, Bitbucket 작업 공간 또는 GitLab 사용자를 공간에 연결하는 것입니다 CodeCatalyst .

GitHub 계정, Bitbucket 작업 공간 또는 GitLab 사용자를 연결하려면 CodeCatalyst

구성하기로 선택한 타사 확장 프로그램에 따라 다음 중 하나를 수행하십시오.

- GitHub 리포지토리: 계정에 연결합니다. GitHub

  1. Connect GitHub 계정을 선택하여 외부 사이트로 이동합니다 GitHub.
  2. GitHub 자격 증명을 사용하여 GitHub 계정에 로그인한 다음 Amazon을 설치할 계정을 선택합니다 CodeCatalyst.

> **Tip**
>
> 이전에 GitHub 계정을 스페이스에 연결한 경우 재승인 메시지가 표시되지 않습니다. 대신 둘 이상의 스페이스에서 구성원 또는 공동 작업자인 경우 확장 프로그램을 설치할 위치를 묻는 대화 상자가 표시되고, 한 GitHub 스페이스에만 속하는 경우 Amazon CodeCatalyst 애플리케이션의 구성 페이지가 표시됩니다. GitHub 허용하려는 리포지토리 액세스를 허용하도록 애플리케이션을 구성한 다음 [Save] 를 선택합니다. 저장 버튼이 활성화되지 않은 경우 구성을 변경한 다음 다시 시도하십시오.

3. 현재 및 미래의 모든 리포지토리에 대한 액세스를 CodeCatalyst 허용할지, 아니면 사용할 특정 GitHub 리포지토리를 선택할지를 선택합니다. CodeCatalyst 기본 옵션은 향후 액세스할 GitHub 리포지토리를 포함하여 GitHub 계정의 모든 리포지토리를 포함하는 것입니다. CodeCatalyst
4. 부여된 권한을 검토한 다음 설치를 CodeCatalyst 선택합니다.

계정을 연결하면 연결된 GitHub 계정과 연결된 GitHub GitHub 리포지토리를 보고 관리할 수 있는 리포지토리 확장 세부 정보 페이지로 이동합니다. CodeCatalyst GitHub

- 비트버킷 리포지토리: Bitbucket 작업 공간에 연결합니다.

1. Bitbucket의 외부 사이트로 이동하려면 Connect Bitbucket 작업 영역을 선택합니다.
2. Bitbucket 자격 증명을 사용하여 Bitbucket 작업 공간에 로그인하고 부여된 권한을 검토하십시오. CodeCatalyst
3. 작업 영역 승인 드롭다운 메뉴에서 CodeCatalyst 액세스를 제공하려는 Bitbucket 작업 영역을 선택한 다음 액세스 허용을 선택합니다.

> **Tip**
>
> 이전에 Bitbucket 작업 영역을 공간에 연결한 경우 재승인 메시지가 표시되지 않습니다. 대신 둘 이상의 Bitbucket 작업 영역의 구성원 또는 공동 작업자인 경우 확장 프로그램을 설치할 위치를 묻는 대화 상자가 표시되고, 한 Bitbucket 작업 영역에만 속하는 경우 Amazon CodeCatalyst 애플리케이션의 구성 페이지가 표시됩니다. 허용하려는 작업 영역 액세스를 위해 애플리케이션을 구성한 다음 액세스 허용을 선택합니다. 액세스 허용 버튼이 활성화되지 않은 경우 구성을 변경한 다음 다시 시도하십시오.

Bitbucket 작업 영역을 연결하면 연결된 Bitbucket 작업 영역 및 연결된 Bitbucket 리포지토리를 보고 관리할 수 있는 Bitbucket 리포지토리 확장 세부 정보 페이지로 이동합니다. CodeCatalyst

- GitLab 리포지토리: 사용자에게 연결합니다. GitLab

  1. Connect GitLab user를 선택하여 외부 사이트로 이동합니다 GitLab.
  2. GitLab 자격 증명을 사용하여 GitLab 사용자에게 로그인하고 부여된 권한을 검토하십시오 CodeCatalyst.

     > **Tip**
     >
     > 이전에 GitLab 사용자를 스페이스에 연결한 경우 재승인 메시지가 표시되지 않습니다. 대신 콘솔로 다시 이동하게 됩니다. CodeCatalyst

  3. [AWS 커넥터 인증 대상] 을 선택합니다. GitLab

  사용자를 연결하면 연결된 GitLab 사용자 및 연결된 GitLab 프로젝트 GitLab 리포지토리를 보고 관리할 수 있는 리포지토리 확장 세부정보 페이지로 이동합니다. CodeCatalyst GitLab

타사 소스를 연결한 후 타사 리포지토리를 프로젝트에 연결할 수 있습니다. CodeCatalyst CodeCatalyst

프로젝트를 생성하려면

1. 프로젝트 만들기 페이지에서 연결한 GitHub 계정을 선택합니다.
2. 연결한 타사 저장소 공급자에 따라 GitHub 저장소, Bitbucket 저장소 또는 저장소 GitLab 저장소 드롭다운 메뉴를 선택하여 타사 저장소를 확인한 다음 프로젝트에 연결할 저장소를 선택합니다.
3. 프로젝트 이름 텍스트 입력 필드에 프로젝트에 할당하려는 이름을 입력합니다. 이름은 스페이스 내에서 고유해야 합니다.
4. 프로젝트 만들기를 선택합니다.

GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 리포지토리 확장을 설치하고, 리소스 공급자를 연결하고, 타사 리포지토리를 CodeCatalyst 프로젝트에 연결한 후 워크플로와 개발 환경에서 사용할 수 있습니다. CodeCatalyst 블루프린트에서 생성된 코드를 사용하여 연결된 GitHub 계정, Bitbucket 작업 공간 또는 사용자에 타사 리포지토리를 만들 수도 있습니다. GitLab Amazon Q Developer, 블루프

린트 등과 함께 연결된 리포지토리를 사용할 수도 있습니다. 자세한 내용은 타사 리포지토리 이벤트 이후 자동으로 워크플로 실행 및 Dev Environment 생성 단원을 참조하세요.

블루프린트로 프로젝트 생성

프로젝트 블루프린트로 모든 프로젝트 리소스와 샘플 코드를 프로비저닝할 수 있습니다. 블루프린트에 대한 자세한 내용은 를 참조하십시오. CodeCatalyst 청사진이 포함된 포괄적인 프로젝트 만들기

블루프린트로 프로젝트를 만들려면

1. CodeCatalyst 콘솔에서 프로젝트를 만들려는 스페이스로 이동합니다.
2. 스페이스 대시보드에서 프로젝트 생성을 선택합니다.
3. '블루프린트로 시작하기'를 선택합니다.

> **Tip**
>
> Amazon Q에서 청사진을 제안하도록 하기 위한 프로젝트 요구 사항을 Amazon Q에 제공하여 청사진을 추가할 수 있습니다. 자세한 내용은 프로젝트를 생성하거나 기능을 추가할 때 Amazon Q를 사용하여 청사진을 선택합니다. 및 Amazon Q를 사용하여 프로젝트를 생성하거나 블루프린트로 기능을 추가하는 모범 사례 단원을 참조하세요. 이 기능은 미국 서부 (오레곤) 지역에서만 사용할 수 있습니다.
> 이 기능을 사용하려면 해당 공간에 제너레이티브 AI 기능을 활성화해야 합니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.

4. CodeCatalyst 블루프린트 또는 스페이스 블루프린트 탭에서 블루프린트를 선택하고 다음을 선택합니다.
5. 프로젝트 이름 아래에 프로젝트에 할당하려는 이름과 관련 리소스 이름을 입력합니다. 이름은 스페이스 내에서 고유해야 합니다.
6. (선택 사항) 기본적으로 블루프린트로 만든 소스 코드는 CodeCatalyst 리포지토리에 저장됩니다. 또는 블루프린트의 소스 코드를 타사 리포지토리에 저장하도록 선택할 수 있습니다. 자세한 내용은 확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst 단원을 참조하십시오.

> **Important**
>
> CodeCatalyst 연결된 리포지토리의 기본 브랜치 변경 감지를 지원하지 않습니다. 연결된 저장소의 기본 분기를 변경하려면 먼저 연결을 해제하고 기본 분기를 변경한 다음 다시 연결해야 합니다. CodeCatalyst 자세한 내용은 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, GitLab Jira 프로젝트 연결 CodeCatalyst 단원을 참조하십시오.

가장 좋은 방법은 리포지토리를 연결하기 전에 항상 최신 버전의 확장 프로그램을 사용하는 것입니다.

사용하려는 타사 저장소 공급자에 따라 다음 중 하나를 수행하십시오.

- GitHub 리포지토리: 계정을 연결합니다. GitHub

  고급 드롭다운 메뉴를 선택하고 리포지토리 GitHub 공급자로 선택한 다음 블루프린트에서 만든 소스 코드를 저장할 GitHub 계정을 선택합니다.

  > **Note**
  >
  > GitHub 계정을 연결하는 경우 개인 연결을 만들어 ID와 ID 간에 ID 매핑을 설정해야 합니다. CodeCatalyst GitHub 자세한 내용은 개인 연결 및 개인적인 연결을 통해 GitHub 리소스에 접근하기 단원을 참조하세요.

- 비트버킷 리포지토리: Bitbucket 작업 공간을 연결합니다.

  고급 드롭다운 메뉴를 선택하고 Bitbucket을 리포지토리 공급자로 선택한 다음 블루프린트로 만든 소스 코드를 저장할 Bitbucket 작업 공간을 선택합니다.

- GitLab 리포지토리: 사용자를 연결합니다. GitLab

  고급 드롭다운 메뉴를 선택하고 리포지토리 GitLab 제공자로 선택한 다음 블루프린트에서 만든 소스 코드를 저장할 GitLab 사용자를 선택합니다.

7. 프로젝트 리소스에서 블루프린트 파라미터를 구성합니다. 블루프린트에 따라 소스 리포지토리 이름을 지정하는 옵션이 있을 수 있습니다.
8. (선택 사항) 선택한 프로젝트 매개 변수에 따라 업데이트된 정의 파일을 보려면 프로젝트 미리 보기 생성에서 코드 보기 또는 워크플로 보기를 선택합니다.
9. (선택 사항) 블루프린트 카드에서 세부 정보 보기를 선택하면 블루프린트의 아키텍처 개요, 필요한 연결 및 권한, 블루프린트가 생성하는 리소스 종류 등 블루프린트에 대한 구체적인 세부 정보를 볼 수 있습니다.
10. 프로젝트 만들기를 선택합니다.

Amazon Q를 사용하여 프로젝트를 생성하거나 블루프린트로 기능을 추가하는 모범 사례

프로젝트를 생성하거나 기존 프로젝트에 새 구성 요소를 추가하려는 경우 어떤 청사진을 사용해야 할지 또는 기능을 통합하는 방법을 잘 모를 수 있습니다. CodeCatalyst 프로젝트 요구 사항을 분석하고 요구 사항에 가장 적합한 청사진을 제안할 수 있는 Amazon Q라는 제너레이티브 AI 어시스턴트와의 통합이 포함됩니다.

Amazon Q를 사용하여 요구 사항에 따라 구성 요소를 생성하는 청사진으로 프로젝트를 생성하거나 Amazon Q를 사용하여 기존 프로젝트에 청사진을 추가할 수 있습니다. 예를 들어 웹 애플리케이션이나 최신 애플리케이션을 위한 리소스를 프로젝트에 추가하려면 요구 사항을 지정하면 권장 청사진과 함께 리소스가 추가됩니다. 나머지 구성 요소에 대한 이슈는 자동으로 생성할 수 있습니다.

Amazon Q는 또한 제안된 청사진으로는 해결할 수 없는 요구 사항 관련 문제를 생성합니다. 또한 이러한 문제를 Amazon Q에 할당할 수 있습니다. Amazon Q에 문제를 할당하면 평가할 수 있는 초안 솔루션을 만들려고 시도합니다. 이를 통해 사용자와 팀은 주의가 필요한 문제에 집중하고 작업을 최적화할 수 있으며, Amazon Q는 즉시 해결할 리소스가 없는 문제에 대한 솔루션을 개발합니다.

> **Note**
>
> Amazon Bedrock 제공: 자동 악용 탐지 AWS 기능을 구현합니다. 나를 위한 설명 작성, 콘텐츠 요약 생성, 작업 추천, Amazon Q를 사용하여 프로젝트에 기능 생성 또는 추가, Amazon Q Developer Agent의 소프트웨어 개발 기능에 대한 Amazon Q 이슈 할당 기능이 Amazon Bedrock에 구축되어 있기 때문에 사용자는 Amazon Bedrock에 구현된 제어 기능을 최대한 활용하여 안전, 보안 및 인공 지능 (AI) 의 책임 있는 사용을 강화할 수 있습니다.

다음은 Amazon Q를 사용하여 프로젝트를 생성하고 청사진을 추가하는 데 도움이 되는 몇 가지 모범 사례입니다.

> **Important**
>
> 제너레이티브 AI 기능은 미국 서부 (오레곤) 지역에서만 사용할 수 있습니다.

- Amazon Q에서 제공하는 기본 프롬프트를 사용하십시오. Amazon Q는 제공된 프롬프트에서 청사진을 선택하는 데 가장 적합합니다.

- Amazon Q에서 제안한 구성 옵션을 사용하여 청사진을 미리 볼 수 있습니다. 블루프린트를 선택하면 블루프린트로 생성될 샘플 코드와 리소스를 미리 볼 수 있습니다.
- Amazon Q에서 사용할 수 있는 스페이스를 사용하십시오. Amazon Q로 프로젝트를 생성하거나 Amazon Q를 사용하여 블루프린트가 있는 프로젝트에 기능을 추가하려면 제너레이티브 AI 기능이 활성화된 공간을 사용하십시오. 자세한 내용은 스페이스의 제너레이티브 AI 기능 활성화 또는 비활성화를 참조하십시오.
- Amazon Q에서 추천하는 블루프린트에 대한 자세한 정보를 확인하십시오. 특정 권장 블루프린트로 생성된 프로젝트 리소스의 종류, 샘플 코드, 구성 요소에 대해 자세히 알아보고 싶을 수도 있습니다. 에서 CodeCatalyst 사용 가능한 블루프린트에 대한 자세한 내용은 을 참조하십시오. CodeCatalyst 청사진이 포함된 포괄적인 프로젝트 만들기
- Amazon Q에서 문제를 해결하도록 허용하십시오. Amazon Q가 사용자를 대신하여 이슈를 생성하고, 해당 이슈를 할당하고, 추적하도록 허용하십시오. 자세한 내용은 튜토리얼: CodeCatalyst 제너레이티브 AI 기능을 사용하여 개발 작업의 속도를 높이세요 단원을 참조하십시오.
- 더 이상 해결되지 않은 문제에 대해 Amazon Q의 할당을 취소하십시오. 예제를 완료한 후 더 이상 해결되지 않는 문제에 대해 Amazon Q 할당을 취소하십시오. Amazon Q가 문제에 대한 작업을 완료했거나 해결책을 찾지 못한 경우, Amazon Q 할당을 취소하여 생성적 AI 기능의 최대 할당량에 도달하지 않도록 하십시오. 자세한 내용은 제너레이티브 AI 기능 관리 및 요금을 참조하십시오.
- Amazon Q 사용량 보기 사용자 수준에서 제너레이티브 AI 기능의 사용량을 볼 수 있습니다. 내 설정으로 이동하여 생성적 AI 할당량을 관리하고 빌더 ID 또는 Single Sign-On () ID별 사용량을 확인하세요. SSO 자세한 내용은 스페이스에서 제너레이티브 AI 기능 사용량 보기를 참조하세요.

> **Important**
>
> 의 제너레이티브 AI CodeCatalyst 기능에는 할당량이 적용됩니다. 자세한 내용은 Amazon Q 개발자 요금, 스페이스에 대한 생성적 AI 기능 활성화 또는 비활성화 및 청구를 참조하십시오.

프로젝트에 블루프린트를 사용하는 모범 사례

다음은 블루프린트로 프로젝트를 만들거나 블루프린트를 추가하는 데 도움이 되는 몇 가지 모범 사례입니다.

- 에서 제공하는 블루프린트를 사용하여 프로젝트를 만들거나 CodeCatalyst 프로젝트에 추가할 수 있습니다. 블루프린트를 사용하여 개발자를 위한 소스 코드와 리소스가 포함된 전체 프로젝트를 만들 수 있습니다. 예를 들어, 웹 애플리케이션 블루프린트는 애플리케이션 및 인프라 리소스를 생성하고 웹 애플리케이션을 배포합니다. 청사진을 사용하여 프로젝트를 만들거나 기존 프로젝트에 사용자

지정 청사진을 추가할 수 있습니다. 자세한 내용은 블루프린트로 프로젝트 만들기 단원을 참조하십시오. 에서 CodeCatalyst 블루프린트를 보고 블루프린트로 생성될 샘플 코드와 리소스를 미리 볼 수 있습니다.

- 조직에서 설계한 사용자 지정 청사진을 사용하세요. 사용자 지정 청사진을 사용하여 공간에 전체 프로젝트를 만들 수 있습니다. 조직에서 설계한 사용자 지정 청사진은 표준화 및 모범 사례를 제공할 수 있으며, 이는 새 프로젝트를 설정하는 데 드는 노력을 줄이는 데도 도움이 될 수 있습니다. 사용자 지정 청사진 작성자는 공간 전체에서 청사진을 사용하는 프로젝트에 대한 세부 정보를 볼 수 있습니다. 라이프사이클 관리를 통해 모든 프로젝트의 소프트웨어 개발 라이프사이클을 중앙에서 관리할 수 있으며, 블루프린트 사용자는 라이프사이클 관리를 활용하여 블루프린트의 업데이트된 옵션 또는 버전으로 코드베이스를 재생성할 수 있습니다. 자세한 내용은 블루프린트 작성자로서 라이프사이클 관리와 협력하기 단원을 참조하십시오.
- 프로젝트 계정에 개발자 역할 또는 적절한 IAM 역할을 추가하세요. 프로젝트 생성 단계를 완료하는 동안 또는 완료한 후에 스페이스에 연결된 IAM 역할을 선택하거나 생성하여 블루프린트 권한을 구성할 수 있습니다. AWS 계정

생성된 프로젝트에 리소스 및 작업 추가

프로젝트가 준비되면 리소스와 작업을 추가할 수 있습니다.

- 프로젝트로 만든 CI/CD 워크플로에 대한 자세한 내용은 을 참조하십시오. 워크플로우 시작하기
- Amazon S3 버킷에 빌드 아티팩트를 배포하는 새 프로젝트의 빌드 작업과 유사한 빌드 작업을 수행하려면 워크플로를 사용한 구축 및 자습서: Amazon S3에 아티팩트 업로드 을 참조하십시오.
- 빈 프로젝트로 시작하여 AWS CloudFormation 스택 배포를 통해 유사한 서버리스 애플리케이션을 배포하는 작업을 하려면 을 참조하십시오. 자습서: 서버리스 애플리케이션 배포
- 문제 계획 게시판을 추가하려면 을 참조하십시오. 에서 문제가 있는 작업을 추적하고 정리하세요. CodeCatalyst
- 프로젝트 개요, 프로젝트 상태, 최근 팀 활동, 배정된 작업을 보려면 을 참조하십시오프로젝트 목록 가져오기.
- 소스 코드를 보거나 풀 리퀘스트를 만들려면 을 참조하십시오소스 리포지토리를 사용하여 코드를 저장하고 공동 작업하십시오. CodeCatalyst .
- 워크플로 실행 성공 또는 실패에 대한 상태 알림을 보내는 알림을 설정하려면 을 참조하십시오에서 Slack 및 이메일 알림 보내기 CodeCatalyst.
- 프로젝트에 멤버를 초대하려면 을 참조하십시오사용자에게 프로젝트 권한 부여.
- 개발 환경을 설정하려면 을 참조하십시오의 개발 환경을 사용하여 코드 작성 및 수정 CodeCatalyst.

프로젝트 목록 가져오기

CodeCatalyst 스페이스에서 프로젝트 권한이 있는 각 프로젝트의 세부 정보를 볼 수 있습니다.

프로젝트를 보려면 프로젝트의 구성원이거나 스페이스의 스페이스 관리자 역할이 있어야 합니다.

아직 프로젝트를 만들지 않은 경우 을 참조하십시오프로젝트 생성. 프로젝트를 만들려는 스페이스의 스페이스 관리자 역할이 있어야 합니다.

- 프로젝트 개요에서 프로젝트 멤버, 소스 리포지토리, 워크플로 실행, 진행 중인 풀 리퀘스트, 프로젝트 개발 환경, 이슈를 볼 수 있습니다.
- 프로젝트 설정에서 프로젝트 세부 정보를 확인 및 관리하고, 프로젝트를 삭제하고, 프로젝트에 새 구성원을 초대하고, 프로젝트 구성원을 관리하고, 알림을 구성할 수 있습니다.

프로젝트 작업 및 개발 환경 보기

사용자에게 배정되거나 직접 생성한 미해결 이슈, 풀 리퀘스트, 프로젝트와 관련된 개발 환경 등의 프로젝트 작업의 요약을 보려면 콘솔을 사용하세요.

프로젝트를 보려면 프로젝트의 멤버이거나 스페이스에 대한 스페이스 관리자 역할이 있어야 합니다.

소스 리포지토리, 워크플로 실행, 이슈, 풀 리퀘스트, 개발 환경, 이슈를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 보려는 프로젝트가 있는 스페이스로 이동합니다. 프로젝트에서 프로젝트를 선택합니다.
3. 탐색 창에서 개요를 선택합니다.
4. 자신에게 배정되고 직접 생성한 프로젝트 작업을 볼 수 있습니다.
    - 멤버 보기 + 전체 목록 보기를 통해 프로젝트 멤버 목록을 볼 수 있습니다.
    - 리포지토리 카드를 보면 프로젝트와 관련된 소스 리포지토리를 볼 수 있습니다.
    - 워크플로 실행 카드를 보면 프로젝트와 관련된 워크플로를 볼 수 있습니다.
    - 오픈 풀 리퀘스트 카드를 보면 자신에게 할당되고 사용자가 생성한 풀 리퀘스트 외에도 코드 리포지토리 상태에 대한 요약을 볼 수 있습니다.
    - 내 개발 환경 카드를 보면 프로젝트와 관련된 개발 환경의 요약을 볼 수 있습니다.
    - 이슈 카드를 보면 배정된 작업 또는 생성한 작업의 요약을 볼 수 있습니다.

스페이스의 모든 프로젝트 보기

스페이스의 프로젝트 목록에서 권한이 있는 모든 프로젝트를 볼 수 있습니다.

사용자에게 배정되거나 직접 생성한 미해결 이슈, 풀 리퀘스트, 프로젝트와 관련된 개발 환경 등의 프로젝트 작업의 요약을 보려면 콘솔을 사용하세요.

프로젝트를 보려면 프로젝트의 멤버이거나 스페이스에 대한 스페이스 관리자 역할이 있어야 합니다.

소스 리포지토리, 워크플로 실행, 이슈, 풀 리퀘스트, 개발 환경, 이슈를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 보려는 프로젝트가 있는 스페이스로 이동합니다. 프로젝트에서 프로젝트를 선택합니다.
3. 탐색 창에서 프로젝트 설정을 선택합니다.
4. 프로젝트 이름, 경로, 프로젝트 ID, 설명을 볼 수 있습니다.

프로젝트 설정 보기

프로젝트 설정에서 프로젝트 멤버, 소스 리포지토리, 워크플로 실행, 공개 풀 리퀘스트, 프로젝트 개발 환경, 이슈를 볼 수 있습니다.

사용자에게 배정되거나 직접 생성한 미해결 이슈, 풀 리퀘스트, 프로젝트와 관련된 개발 환경 등의 프로젝트 작업 요약을 보려면 콘솔을 사용하세요.

소스 리포지토리, 워크플로 실행, 이슈, 풀 리퀘스트, 개발 환경, 이슈를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 보려는 프로젝트가 있는 스페이스로 이동합니다. 프로젝트에서 프로젝트를 선택합니다.
3. 탐색 창에서 프로젝트 설정을 선택합니다.
4. 프로젝트 이름, 경로, 프로젝트 ID, 설명을 볼 수 있습니다.

에서 다른 프로젝트로 변경 CodeCatalyst

다른 프로젝트로 변경하려면 콘솔을 사용하여 액세스 권한이 있는 프로젝트 목록에서 선택하십시오.

다른 프로젝트로 변경하기

1. CodeCatalyst 콘솔에서 상단의 프로젝트 선택기를 선택합니다.

2. 드롭다운을 펼치고 탐색하려는 프로젝트를 선택합니다.

프로젝트 삭제

프로젝트를 삭제하여 프로젝트 리소스에 대한 모든 액세스 권한을 제거할 수 있습니다. 프로젝트를 삭제하려면 스페이스 관리자 또는 프로젝트 관리자 역할이 있어야 합니다. 프로젝트를 삭제하면 프로젝트 구성원은 프로젝트 리소스에 액세스할 수 없으며 타사 소스 저장소에서 요청하는 모든 워크플로가 중지됩니다.

프로젝트를 삭제하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 보려는 프로젝트가 있는 스페이스로 이동합니다. 프로젝트에서 프로젝트를 선택합니다.
3. 탐색 창에서 프로젝트 설정을 선택합니다.
4. 프로젝트 삭제를 선택합니다.
5. 삭제를 **`delete`** 확인하려면 Enter를 누르십시오.
6. 프로젝트 삭제를 선택합니다.

사용자에게 프로젝트 권한 부여

Amazon CodeCatalyst 콘솔을 사용하여 프로젝트의 멤버를 관리할 수 있습니다. 사용자를 추가 또는 제거하고, 현재 구성원의 역할을 관리하고, 프로젝트 참여 초대를 보내고, 아직 수락되지 않은 초대를 취소할 수 있습니다.

스페이스 및 프로젝트 사용자의 구성원 페이지에서 사용자는 여러 역할을 가질 수 있습니다. 여러 역할을 가진 사용자는 역할을 여러 개 가지고 있을 때 표시기가 표시되며, 가장 많은 권한을 가진 역할부터 표시됩니다.

구성원 및 프로젝트 역할 목록 가져오기

프로젝트에 사용자를 추가할 때 다음과 같이 프로젝트 권한을 부여하는 역할을 할당합니다.

- 프로젝트 관리자 역할은 프로젝트의 모든 권한을 가집니다. 프로젝트 설정 편집, 프로젝트 권한 관리, 프로젝트 삭제 등 프로젝트의 모든 측면을 관리해야 하는 사용자에게만 이 역할을 할당하십시오. 자세한 내용은 프로젝트 관리자 역할 단원을 참조하십시오.

- 기여자 역할에는 프로젝트 작업에 필요한 권한이 있습니다. 프로젝트의 코드, 워크플로, 이슈 및 작업을 수행해야 하는 사용자에게 이 역할을 할당하세요. 자세한 내용은 기여자 역할 단원을 참조하십시오.
- 검토자 역할에는 검토 권한이 있습니다. 세부 정보는 리뷰어 역할을 참조하세요.
- 읽기 전용 역할에는 읽기 권한이 있습니다. 세부 정보는 읽기 전용 역할을 참조하세요.

스페이스 관리자 역할을 가진 사용자는 이미 스페이스의 모든 프로젝트에 대한 암시적 액세스 권한을 가지고 있으므로 프로젝트에 초대할 필요가 없습니다.

스페이스 관리자 역할을 할당하지 않고 프로젝트에 사용자를 초대하면 해당 사용자는 프로젝트 아래의 프로젝트 멤버 테이블과 스페이스 아래의 프로젝트 멤버 테이블에 표시됩니다.

스페이스의 사용자 및 역할을 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 보려는 프로젝트가 있는 스페이스로 이동합니다. 프로젝트에서 프로젝트를 선택합니다.
3. 탐색 창에서 프로젝트 설정을 선택합니다.
4. 구성원 탭을 선택합니다.

   프로젝트 멤버 테이블에는 프로젝트에서 역할을 가진 모든 멤버가 표시됩니다.

   > **Tip**
   >
   > 스페이스 관리자 역할이 있는 경우 자신이 직접 초대된 프로젝트를 볼 수 있습니다. 프로젝트의 프로젝트 설정으로 이동한 다음 내 프로젝트를 선택합니다.

   스페이스 관리자 표에는 스페이스 관리자 역할을 가진 사용자가 표시됩니다. 이러한 사용자는 스페이스의 모든 프로젝트에 자동으로 (단순하게) 할당되며 프로젝트에서의 역할은 없습니다.

   상태 열의 유효한 값은 다음과 같습니다.

   - 초대됨 - 초대를 CodeCatalyst 보냈지만 사용자가 아직 수락하거나 거부하지 않았습니다.
   - 회원 - 사용자가 초대를 수락했습니다.

주제

- 프로젝트에 사용자 초대하기

- 초대 취소
- 프로젝트에서 사용자 삭제하기
- 프로젝트 초대 수락 또는 거절

프로젝트에 사용자 초대하기

콘솔을 사용하여 프로젝트에 사용자를 초대할 수 있습니다. 스페이스 구성원을 초대하거나 스페이스 외부에서 이름을 추가할 수 있습니다.

사용자를 프로젝트에 초대하려면 프로젝트 관리자 또는 스페이스 관리자 역할로 로그인해야 합니다.

스페이스 관리자 역할을 가진 사용자는 이미 스페이스의 모든 프로젝트에 대한 암시적 액세스 권한을 가지고 있으므로 프로젝트에 초대할 필요가 없습니다.

스페이스 관리자 역할을 할당하지 않고 프로젝트에 사용자를 초대하면 해당 사용자는 프로젝트 아래의 프로젝트 멤버 테이블과 스페이스 아래의 프로젝트 멤버 테이블에 표시됩니다.

프로젝트 설정 탭에서 멤버를 프로젝트에 초대하려면

1. 프로젝트로 이동합니다.

   > **Tip**
   >
   > 상단 내비게이션 바에서 보려는 프로젝트를 선택할 수 있습니다.

2. 탐색 창에서 프로젝트 설정을 선택합니다.
3. 구성원 탭을 선택합니다.
4. 프로젝트 멤버에서 새 멤버 초대를 선택합니다.
5. 새 구성원의 이메일 주소를 입력하고 이 구성원의 역할을 선택한 다음 초대를 선택합니다. 역할에 관한 자세한 내용은 사용자 역할을 통한 액세스 권한 부여 단원을 참조하세요.

프로젝트 개요 페이지에서 멤버를 프로젝트에 초대하려면

1. 프로젝트로 이동합니다.

> **Tip**
>
> 상단 내비게이션 바에서 보려는 프로젝트를 선택할 수 있습니다.

2. 멤버 + 버튼을 선택합니다.
3. 새 구성원의 이메일 주소를 입력하고 이 구성원의 역할을 선택한 다음 초대를 선택합니다. 역할에 관한 자세한 내용은 사용자 역할을 통한 액세스 권한 부여 단원을 참조하세요.

초대 취소

최근에 초대를 보낸 경우 초대가 아직 수락되지 않았으면 취소할 수 있습니다.

프로젝트 초대를 관리하려면 프로젝트 관리자 또는 스페이스 관리자 역할이 있어야 합니다.

프로젝트 멤버 초대를 취소하려면

1. 취소하려는 초대를 보낸 프로젝트로 이동합니다.
2. 탐색 창에서 프로젝트 설정을 선택합니다.
3. 멤버 탭을 보고 멤버가 초대됨 상태인지 확인합니다.

> **Note**
>
> 아직 수락되지 않은 초대만 취소할 수 있습니다.

4. 초대된 구성원이 있는 행 옆의 옵션을 선택한 다음 초대 취소를 선택합니다.
5. 확인 창이 표시됩니다. 초대 취소를 선택하여 확인합니다.

프로젝트에서 사용자 삭제하기

콘솔을 사용하여 프로젝트에서 사용자를 제거할 수 있습니다.

프로젝트에서 사용자를 제거하려면 프로젝트 관리자 또는 스페이스 관리자 역할로 로그인해야 합니다.

> **Note**
>
> 스페이스 내의 모든 프로젝트에서 사용자를 제거하면 해당 사용자가 해당 스페이스에서 자동으로 제거됩니다.

프로젝트에서 사용자를 제거하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 보려는 프로젝트가 있는 스페이스로 이동합니다. 프로젝트에서 프로젝트를 선택합니다.
3. 탐색 창에서 프로젝트 설정을 선택합니다.
4. 구성원 탭을 선택합니다.
5. 제거하려는 프로필 옆의 선택기를 선택한 다음 제거를 선택합니다.
6. 사용자를 제거할 것인지 확인한 다음 제거를 선택합니다.

프로젝트 초대 수락 또는 거절

Amazon CodeCatalyst 프로젝트에 참여하라는 이메일 초대장을 받을 수 있습니다. 여러분은 초대를 수락하거나 거부할 수 있습니다.

초대를 수락하거나 거부하려면

1. 초대 이메일을 엽니다.
2. 이메일에서 프로젝트 링크를 선택합니다.
3. 수락 또는 거부를 선택합니다.

   거절을 선택하면 초대를 거부했음을 알리는 이메일이 프로젝트 관리 계정으로 전송됩니다.

팀을 통한 프로젝트 액세스 허용

프로젝트를 생성한 후 팀을 추가할 수 있습니다. 팀을 사용하면 사용자를 그룹화하여 이들이 권한을 공유하고 프로젝트, 이슈 추적, 역할 및 리소스를 프로젝트 및 스페이스 CodeCatalyst 구성원으로 관리할 수 있습니다.

프로젝트의 팀을 관리하려면 프로젝트 관리자 역할이 있어야 합니다.

팀은 의 스페이스 수준에서도 CodeCatalyst 관리됩니다. 스페이스의 팀에 대한 자세한 내용은 을 참조하십시오팀을 통한 공간 액세스 허용.

주제

- 프로젝트에 팀 추가
- 팀에 프로젝트 역할 부여
- 팀의 프로젝트 역할 제거

프로젝트에 팀 추가

팀 구성원이 프로젝트의 리소스에 액세스할 수 있는 팀을 관리할 수 있습니다.

스페이스 및 프로젝트 사용자의 멤버 페이지에서 사용자는 여러 역할을 가질 수 있습니다. 여러 역할을 가진 사용자는 역할을 여러 개 가지고 있을 때 표시기가 표시되며, 가장 많은 권한을 가진 역할부터 표시됩니다.

팀 추가하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다. 프로젝트 설정을 선택한 다음 팀을 선택합니다.
3. 팀 추가를 선택합니다.
4. Team에서 사용 가능한 팀 목록에서 팀을 선택합니다.
5. 프로젝트 역할에서 사용 가능한 프로젝트 역할 목록에서 역할을 선택합니다 CodeCatalyst.
   - 프로젝트 관리자 - 자세한 내용은 을 참조하십시오프로젝트 관리자 역할.
   - 기여자 — 자세한 내용은 을 참조하십시오기여자 역할.
   - 검토자 — 자세한 내용은 을 참조하십시오. 리뷰어 역할
   - 읽기 전용 — 자세한 내용은 을 참조하십시오읽기 전용 역할.
6. 팀 추가를 선택합니다.

팀에 프로젝트 역할 부여

팀은 스페이스에서 고급 사용자와 같은 역할 권한을 가질 수 있습니다. 팀의 스페이스 역할을 변경할 수 있지만 팀의 모든 구성원이 해당 권한을 상속한다는 점에 유의하세요.

프로젝트 역할 추가 또는 변경하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다. 프로젝트 설정을 선택한 다음 팀을 선택합니다.
3. 역할을 변경하려면 이 목록에서 팀 옆에 있는 선택기를 선택한 다음 역할 변경을 선택합니다. 역할을 추가하려면 프로젝트 역할 추가를 선택합니다. 프로젝트에서 추가하려는 프로젝트를 선택하고 역할에서 역할을 선택합니다. 사용 가능한 프로젝트 역할 중 하나를 선택합니다.
   - 프로젝트 관리자 - 자세한 내용은 을 참조하십시오프로젝트 관리자 역할.
   - 기여자 - 자세한 내용은 을 참조하십시오기여자 역할.
   - 리뷰어 - 자세한 내용은 을 참조하십시오. 리뷰어 역할
   - 읽기 전용 - 자세한 내용은 을 참조하십시오읽기 전용 역할.
4. 저장(Save)을 선택합니다.

팀의 프로젝트 역할 제거

에서 CodeCatalyst 팀의 프로젝트 역할을 볼 수 있습니다. 팀 구성원을 볼 수도 있습니다. 팀의 프로젝트 역할을 제거할 수 있습니다.

프로젝트 역할을 제거하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다. 프로젝트 설정을 선택한 다음 팀을 선택합니다.
3. 프로젝트 역할 탭을 선택합니다.
4. 제거하려는 역할을 선택합니다.

> **Important**
> 팀에서 역할을 제거하면 팀 내 모든 사용자의 관련 권한이 제거됩니다.

5. 저장(Save)을 선택합니다.

시스템 리소스에 대한 프로젝트 액세스 허용

컴퓨터 리소스는 프로젝트 또는 스페이스에 대한 권한이 부여된 특정 리소스입니다 CodeCatalyst. CodeCatalystthat

> **Note**
>
> 머신 리소스라는 용어는 EC2 인스턴스와 같은 클라우드 인프라를 가리키는 것이 아니라 공간이나 프로젝트에 대한 권한이 있는 블루프린트 또는 워크플로 리소스를 가리킵니다.

프로젝트에서 머신 리소스를 사용하는 예로는 블루프린트 리소스가 사용자를 대신하여 프로젝트에 액세스할 수 있도록 하는 경우를 들 수 있습니다.

머신 리소스는 승인된 리소스가 액세스 시 CodeCatalyst SSO 획득한 사용자 ID를 나타냅니다. 머신 리소스는 프로젝트의 리소스 (예: 청사진 및 워크플로우) 에 권한을 부여하는 데 사용됩니다. 프로젝트의 컴퓨터 리소스를 볼 수 있으며 프로젝트의 컴퓨터 리소스를 활성화하거나 비활성화하도록 선택할 수 있습니다. 예를 들어 액세스를 관리하기 위해 컴퓨터 리소스를 비활성화했다가 나중에 다시 활성화해야 할 수 있습니다.

이러한 작업은 컴퓨터 리소스를 해지하거나 사용하지 않도록 설정해야 하는 경우 컴퓨터 리소스에 사용할 수 있습니다. 예를 들어 자격 증명이 손상되었을 것으로 의심되는 경우 컴퓨터 리소스를 사용하지 않도록 설정할 수 있습니다. 일반적으로 이러한 작업은 사용할 필요가 없습니다.

이 페이지를 보고 프로젝트 수준에서 컴퓨터 리소스를 관리하려면 스페이스 관리자 역할 또는 프로젝트 관리자 역할이 있어야 합니다.

컴퓨터 리소스는 의 공간 수준에서도 CodeCatalyst 관리됩니다. 스페이스/프로젝트의 팀에 대해 자세히 알아보려면 을 참조하십시오. 시스템 리소스에 대한 공간 액세스 허용

주제

- 컴퓨터 리소스에 대한 프로젝트 액세스 보기
- 컴퓨터 리소스에 대한 프로젝트 액세스 비활성화
- 시스템 리소스에 대한 프로젝트 액세스 활성화

컴퓨터 리소스에 대한 프로젝트 액세스 보기

프로젝트에서 사용 중인 컴퓨터 리소스 목록을 볼 수 있습니다.

스페이스 관리자 역할 또는 프로젝트 관리자 역할이 있어야 합니다.

컴퓨터 리소스를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.

2. 프로젝트로 이동한 다음 프로젝트 설정을 선택합니다. 컴퓨터 리소스를 선택합니다.
3. 드롭다운에서 워크플로 작업을 선택하면 워크플로의 컴퓨터 리소스만 볼 수 있습니다. 블루프린트의 머신 리소스만 보려면 블루프린트를 선택합니다.

   필터 필드를 사용하여 이름을 필터링할 수도 있습니다.

컴퓨터 리소스에 대한 프로젝트 액세스 비활성화

프로젝트에서 사용 중인 컴퓨터 리소스를 비활성화하도록 선택할 수 있습니다.

> **Important**
>
> 머신 리소스를 비활성화하면 스페이스의 모든 관련 블루프린트 또는 워크플로에 대한 모든 권한이 제거됩니다.

스페이스 관리자 역할 또는 프로젝트 관리자 역할이 있어야 합니다.

컴퓨터 리소스를 사용하지 않도록 설정하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동한 다음 프로젝트 설정을 선택합니다. 컴퓨터 리소스를 선택합니다.
3. 다음 중 하나를 선택합니다.

   > **Important**
   >
   > 머신 리소스를 비활성화하면 스페이스의 모든 관련 블루프린트 또는 워크플로에 대한 모든 권한이 제거됩니다.

   - 개별적으로 비활성화하려면 비활성화하려는 하나 이상의 머신 리소스 옆에 있는 선택기를 선택합니다. 비활성화를 선택한 다음 이 리소스를 선택합니다.
   - 모든 리소스를 비활성화하려면 비활성화를 선택한 다음 모든 리소스를 선택합니다.
   - 모든 워크플로 작업을 비활성화하려면 비활성화를 선택한 다음 모든 워크플로 작업을 선택합니다.
   - 블루프린트를 모두 비활성화하려면 비활성화를 선택한 다음 모든 블루프린트를 선택합니다.

시스템 리소스에 대한 프로젝트 액세스 활성화

프로젝트에서 사용 중이지만 비활성화된 컴퓨터 리소스를 활성화하도록 선택할 수 있습니다.

스페이스 관리자 역할 또는 프로젝트 관리자 역할이 있어야 합니다.

컴퓨터 리소스를 활성화하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동한 다음 프로젝트 설정을 선택합니다. 컴퓨터 리소스를 선택합니다.
3. 다음 중 하나를 선택합니다.
   - 개별적으로 활성화하려면 활성화하려는 하나 이상의 컴퓨터 리소스 옆에 있는 선택기를 선택합니다. [활성화] 를 선택한 다음 [이 리소스] 를 선택합니다.
   - 모든 리소스를 활성화하려면 활성화를 선택한 다음 모든 리소스를 선택합니다.
   - 모든 워크플로 작업을 활성화하려면 활성화를 선택한 다음 모든 워크플로 작업을 선택합니다.
   - 모든 블루프린트를 활성화하려면 활성화를 선택한 다음 모든 블루프린트를 선택합니다.

프로젝트 할당량

다음 표에는 Amazon 프로젝트의 할당량 및 한도가 설명되어 있습니다. CodeCatalyst CodeCatalystAmazon의 할당량에 대한 자세한 내용은 을 참조하십시오. 에 대한 할당량 CodeCatalyst

스페이스당 최대 프로젝트 수	100
사용자가 속할 수 있는 최대 프로젝트 수	1,000
프로젝트에 속할 수 있는 최대 구성원 수	10,000개
프로젝트 이름	프로젝트 이름은 스페이스 내에서 고유해야 합니다. 이름은 3~63자 사이여야 합니다. 이름은 대/소문자를 구분합니다. 프로젝트 이름은 영숫자로 시작해야 합니다. 유효한 문자: A-Z, a-z, 0-9, 공백,., _ (밑줄) - (하이픈)

	프로젝트 이름에는 다음 문자를 사용할 수 없습니다. ! ? @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
제품 설명	프로젝트 설명은 최대 200자까지 입력할 수 있습니다. 유효한 문자: A-Z, a-z, 0-9, 공백 및., _ (밑줄) - (하이픈) 프로젝트 설명은 선택 사항입니다.

에서 알림 보내기 CodeCatalyst

에서 프로젝트 및 리소스를 모니터링하도록 알림을 설정할 수 CodeCatalyst 있습니다. 사용자는 자신이 멤버로 있는 모든 프로젝트에서 이메일을 받고 싶은 프로젝트 이벤트를 선택할 수 있습니다. Slack과 같은 팀 메시징 애플리케이션에서 CodeCatalyst 스페이스와 Slack 작업 영역 간의 액세스를 구성한 다음 해당 Slack 작업 영역에 있는 하나 이상의 채널로 프로젝트 알림을 전송하도록 구성하여 팀 전체에 알림을 보내도록 구성할 수도 있습니다. CodeCatalyst 스페이스와 Slack 작업 영역 간에 액세스를 구성하면 프로젝트 구성원이 자신의 Slack 구성원을 추가하여 연결된 Slack 작업 공간 및 채널의 CodeCatalyst 이벤트에 대해 직접 알림을 받을 수도 있습니다. IDs

> **Note**
>
> Slack으로 전송할 수 있는 프로젝트 이벤트 세트는 사용자가 이메일로 알림을 받도록 선택할 수 있는 이벤트 세트와 다릅니다.

주제

- 알림은 어떻게 작동합니까?
- 슬랙 알림 시작하기
- 에서 Slack 및 이메일 알림 보내기 CodeCatalyst

알림은 어떻게 작동합니까?

Slack과 같은 팀 메시징 애플리케이션에 알림을 제공하도록 프로젝트를 설정할 수 있습니다.

알림에 필요한 권한은 무엇인가요?

모든 프로젝트 멤버는 에서 채널의 알림 설정을 구성, 확인, 업데이트 또는 삭제할 수 CodeCatalyst 있습니다. 하지만 스페이스 관리자 역할을 가진 사용자만 Slack 작업 영역을 추가하거나 삭제할 수 있습니다. 모든 사용자는 자신이 속한 프로젝트에 대해 이메일을 받고 싶은 프로젝트 이벤트를 구성할 수 있습니다. CodeCatalyst

어떤 CodeCatalyst 이벤트에 대한 알림을 구성할 수 있나요?

워크플로 이벤트에 대한 알림을 하나 이상의 Slack 채널에 CodeCatalyst 전달하도록 구성할 수 있습니다. 프로젝트와 Slack 간에 알림이 구성되면 CodeCatalyst 프로젝트 사용자는 Slack 멤버 ID를 추가하여 Slack 채널에서 이벤트에 대한 다이렉트 메시지를 수신하도록 선택할 수 있습니다. CodeCatalyst Slack 회원 ID를 추가한 사용자는 프로젝트에 맞게 구성된 Slack 채널에서 자신의 ID가 직접 언급되어 관심 있는 이벤트에 대한 인식을 높이는 데 도움이 됩니다.

이메일을 받고 싶은 이벤트도 선택할 수 있습니다. 이러한 이메일은 AWS 빌더 ID에 구성된 이메일 주소로 전송됩니다.

알림은 어떻게 표시되나요?

하나 이상의 Slack 채널에 알림을 CodeCatalyst 전달하도록 구성할 수 있습니다. Slack 작업 공간에 액세스할 수 있는 권한을 CodeCatalyst 부여하려면 권한을 부여해야 합니다. 승인이 제공되면 구성한 Slack 채널에 알림을 전달할 CodeCatalyst 수 있습니다. 프로젝트 구성원이 Slack 멤버 ID를 추가하기로 선택하면 해당 프로젝트에 구성된 Slack 채널에서 CodeCatalyst 이벤트에 대한 멘션을 받을 수 있습니다.

알림을 설정하려면 어떻게 해야 하나요?

이메일 알림은 의 일부로 구성됩니다 CodeCatalyst. 프로젝트 사용자는 내 설정 페이지에서 이메일을 받고 싶은 이벤트를 선택할 수 있습니다.

프로젝트 리소스에 대해 Slack 알림을 설정하려면 다음과 같은 고급 작업을 완료해야 합니다.

알림을 설정하려면 (상위 수준 작업)

1. CodeCatalyst에서는 CodeCatalyst Slack과 같은 메시징 클라이언트 간의 연결을 설정합니다. Slack 작업 영역이 연결되면 해당 스페이스의 모든 프로젝트에서 사용할 수 있습니다.

> **Note**
>
> 스페이스 관리자 역할을 가진 사용자만 Slack 작업 영역을 추가하거나 삭제할 수 있습니다.

2. 내 CodeCatalyst 프로젝트에 팀이 알림을 받길 원하는 채널을 추가하세요.
3. CodeCatalyst에서는 워크플로 실행 실패와 같은 다양한 이벤트에 대한 알림을 켜고 알림을 보낼 채널을 지정합니다.

자세한 단계는 슬랙 알림 시작하기 단원을 참조하세요.

CodeCatalyst 스페이스와 Slack 사이에 알림이 구성되면 사용자는 자신의 Slack 멤버 ID를 추가하여 프로젝트에 구성된 Slack 채널의 CodeCatalyst 이벤트에 대한 다이렉트 메시지를 수신하도록 선택할 수 있습니다.

슬랙 알림 시작하기

프로젝트를 만든 후 팀에서 프로젝트 리소스를 모니터링하는 데 도움이 되는 Slack 알림을 설정할 수 있습니다.

다음 단계는 Slack 알림을 처음으로 설정하는 과정을 안내합니다. CodeCatalyst 알림을 이미 구성한 경우 을 참조하십시오에서 Slack 및 이메일 알림 보내기 CodeCatalyst.

> **Note**
>
> 알림 채널로 전송할 수 있는 프로젝트 이벤트 세트는 사용자가 이메일로 알림을 받도록 선택할 수 있는 이벤트 세트와 다릅니다. 자세한 정보는 에서 Slack 및 이메일 알림 보내기 CodeCatalyst을 참조하세요.

주제

- 4단계: 다음 단계

사전 요구 사항

시작하려면 다음이 필요합니다.

- CodeCatalyst 스페이스. CodeCatalyst 스페이스를 만들고 처음으로 로그인하는 방법에 대한 자세한 내용은 을 참조하십시오설정 및 로그인 CodeCatalyst.
- CodeCatalyst 프로젝트. 자세한 정보는 프로젝트 생성을 참조하세요.
- 프로젝트 관리자 또는 스페이스 관리자 역할을 가진 CodeCatalyst 계정. 자세한 정보는 사용자 역할을 통한 액세스 권한 부여을 참조하세요.
- 에서 액세스할 수 있는 Slack 계정 및 Slack 작업 영역 CodeCatalyst
- 알림을 보내는 Slack 채널입니다. CodeCatalyst 채널은 공개 또는 비공개일 수 있습니다.

1단계: Slack CodeCatalyst 워크스페이스에 연결

스페이스 관리자 역할을 가진 사용자만 Slack 작업 영역을 추가하거나 삭제할 수 있습니다. Slack 작업 영역을 추가하거나 삭제하면 스페이스의 모든 프로젝트에 영향을 줍니다. Slack 간의 연결을 설정하려면 Slack 작업 CodeCatalyst 영역과 함께 안전한 OAuth 인증 핸드셰이크를 수행하십시오. CodeCatalyst

다음 지침에 따라 Slack 워크스페이스에 CodeCatalyst 연결하세요.

> **Note**
>
> 이 작업은 각 Slack 작업 영역에 한 번만 수행하면 됩니다. 그런 다음 Slack 채널별로 알림을 설정할 수 있습니다.

Slack 작업 CodeCatalyst 공간에 연결하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다.
3. 탐색 창에서 프로젝트 설정을 선택합니다.
4. 알림 탭을 선택합니다.

5. 알림 구성을 선택합니다.
6. Slack 작업 영역에 연결을 선택합니다.
7. 대화 상자 내용을 읽은 다음 [Slack 작업 영역에 연결] 을 선택합니다.
8. AWSChatbot 메시지에서:

   a. 오른쪽 상단에서 채널이 포함된 Slack 작업 영역을 선택합니다.

   b. [Allow]를 선택합니다.

   CodeCatalyst 콘솔로 돌아갑니다.
9. 계속해서 2단계: Slack 채널을 다음에 추가 CodeCatalyst로 이동하십시오.

2단계: Slack 채널을 다음에 추가 CodeCatalyst

채널을 추가하려면 Slack 채널 ID가 필요합니다. CodeCatalyst

슬랙 채널 ID를 받으려면

1. 슬랙에 로그인하세요. 자세한 내용은 Slack에 로그인을 참조하십시오.
2. 알림을 보내려는 채널이 포함된 Slack 작업 영역으로 이동합니다. 자세한 내용은 Slack 작업 영역 간 전환 또는 추가 Slack 작업 영역에 로그인을 참조하십시오.
3. 탐색 창에서 알림을 보낼 채널의 컨텍스트 메뉴 (오른쪽 클릭) 를 열고 채널 세부 정보 열기를 선택합니다.

   채널 ID는 대화 상자 하단에 표시됩니다.
4. 채널 ID 값을 복사합니다. 다음 단계에서 이 정보를 사용할 것입니다.

방금 복사한 채널 ID를 사용하여 이제 Slack 채널을 연결할 CodeCatalyst 수 있습니다.

Slack 채널을 추가하려면 CodeCatalyst

1. 시작하기 전에 Slack 채널이 비공개인 경우 다음과 같이 채널에 AWS Chatbot 앱을 추가하세요.

   a. Slack 채널의 메시지 상자에 를 **`@aws`** 입력하고 대화 상자에서 aws app을 선택합니다.

   b. Enter(입력) 키를 누릅니다.

   Chatbot이 비공개 채널에 없음을 나타내는 AWS Slackbot 메시지가 나타납니다.

c. 사용자 초대를 선택하여 채널에 AWS Chatbot을 초대하세요.

2. CodeCatalyst 콘솔에서 다음을 선택합니다.
3. 채널 ID에 이전에 확보한 Slack 채널 ID를 붙여넣습니다.
4. 채널 이름에 이름을 입력합니다. Slack 채널 이름을 사용하는 것이 좋습니다.
5. 다음을 선택합니다.
6. 알림 이벤트 선택에서 알림을 받을 이벤트 유형을 선택합니다.
7. 마침을 클릭합니다.

3단계: Slack에서 CodeCatalyst Slack으로 알림 테스트

워크플로우 상태에 대한 알림을 보내도록 프로젝트를 구성한 후에는 Slack에서 알림을 볼 수 있습니다.

Slack에서 알림을 보려면

1. CodeCatalyst 프로젝트에서 워크플로를 수동으로 시작하여 워크플로 실행을 완료하고 실행이 완료되면 상태 알림을 받으세요.
2. Slack에서 알림을 받도록 설정한 채널을 확인하세요. 알림에는 각 워크플로 실행의 최신 상태와 실패 또는 성공 여부가 표시됩니다.

4단계: 다음 단계

공간에 맞게 Slack 작업 CodeCatalyst 공간을 구성한 후에는 기존 CodeCatalyst 프로젝트에 Slack 채널을 추가하고 프로젝트를 생성한 후 새 프로젝트에 추가할 수 있습니다. 또한 프로젝트 사용자에게 Slack 멤버 ID에 대한 개인 Slack 알림을 구성하고 이메일을 수신할 이벤트를 구성할 수 있음을 알릴 수 있습니다. 자세한 정보는 에서 Slack 및 이메일 알림 보내기 CodeCatalyst 단원을 참조하세요.

에서 Slack 및 이메일 알림 보내기 CodeCatalyst

프로젝트에서 발생하는 이벤트에 대한 알림을 CodeCatalyst 보내도록 구성할 수 있습니다. CodeCatalyst Slack 채널과 같은 메시징 클라이언트에 알림을 보낼 수 있습니다. Slack 채널로 메시지를 CodeCatalyst 보내면 팀 전체가 워크플로 장애와 같은 중요한 이벤트를 인지할 수 있습니다. 선택적으로 보내는 Slack 메시지에 CodeCatalyst @mention 님이 포함되도록 선택하여 해당하는 다이렉트 메시지 (DM) 를 받을 수 있습니다.

CodeCatalyst 이메일로 직접 알림을 보낼 수도 있습니다. 멤버로 속해 있는 모든 프로젝트의 이벤트에 대한 이메일 알림이 전송됩니다. 이러한 이메일은 AWS 빌더 ID에 구성된 이메일 주소로 발송됩니다.

> **Note**
>
> Slack 채널로 전송할 수 있는 이벤트는 이메일로 전송되는 이벤트와 다를 수 있습니다.

주제

이메일 알림 구성

멤버로 속해 있는 모든 프로젝트의 이벤트에 대한 이메일 알림을 받도록 선택할 수 있습니다. 이러한 이메일은 AWS 빌더 ID에 구성된 이메일 주소로 발송됩니다. 기본적으로 이메일을 보낼 수 있는 모든 프로젝트 이벤트에 대한 이메일을 받게 됩니다.

프로젝트 이벤트에 대한 이메일 알림을 구성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 상단 메뉴 바에서 프로필 배지를 선택한 다음 내 설정을 선택합니다. CodeCatalyst 내 설정 페이지가 열립니다.

   > **Tip**
   >
   > 프로젝트 또는 스페이스의 구성원 페이지로 이동한 다음 구성원 목록에서 이름을 선택하여 사용자 프로필을 찾을 수도 있습니다.

3. 이메일 알림의 목록에서 이메일 알림을 구성하려는 프로젝트를 찾은 다음 편집을 선택합니다.
4. 이메일을 수신하려는 이벤트를 선택한 다음 저장을 선택합니다.

Slack 채널에 알림 보내기

팀의 Slack 채널에 프로젝트 이벤트에 대한 알림을 CodeCatalyst 보내도록 구성할 수 있습니다. 이렇게 하면 워크플로 실행이 실패하는 경우와 같은 중요한 이벤트를 팀 전체가 인지하도록 할 수 있습니다.

> **Note**
>
> 프로젝트의 모든 구성원은 해당 프로젝트의 채널로 전송되는 알림을 관리할 수 있습니다. 하지만 스페이스 관리자 역할을 가진 사용자만 Slack 작업 영역을 추가하거나 삭제할 수 있습니다.

다음 지침에 따라 알림을 보낼 Slack 채널을 추가하세요.

알림을 위한 Slack 채널을 추가하려면

1. Slack 채널을 처음 추가하는 경우 대신 확인하세요. 슬랙 알림 시작하기

   첫 번째 채널을 설정한 후 이 절차로 돌아가 추가 채널을 설정하세요.
2. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
3. 프로젝트로 이동합니다.
4. 탐색 창에서 프로젝트 설정을 선택합니다.
5. 알림 탭을 선택합니다.
6. Add channel(채널 추가)을 선택합니다.
7. 작업 영역 선택을 선택한 다음 알림을 보내려는 채널이 포함된 Slack 작업 영역을 선택합니다.

   Slack 작업 영역이 목록에 없는 경우, 의 지침에 따라 추가할 수 있습니다. 슬랙 알림 시작하기
8. 추가하려는 Slack 채널이 비공개인 경우 채널 ID를 입력하기 전에 다음 단계를 완료하세요.

   a. Slack 채널의 메시지 상자에 를 **`@aws`** 입력하고 팝업에서 aws 앱을 선택합니다.

   b. Enter를 누릅니다.

      Chatbot이 비공개 채널에 없음을 나타내는 AWS Slackbot 메시지가 나타납니다.

   c. 사용자 초대를 선택하여 채널에 AWS Chatbot을 초대하세요.
9. CodeCatalyst의 채널 ID 필드에 Slack 채널 ID를 입력합니다. ID를 찾으려면 Slack으로 이동하여 탐색 창에서 채널을 마우스 오른쪽 버튼으로 클릭하고 채널 세부 정보 열기를 선택합니다.

채널 ID는 대화 상자 하단에 표시됩니다.

10. 채널 이름에 이름을 입력합니다. Slack 채널 이름을 사용하는 것이 좋습니다.
11. 알림 이벤트 선택에서 알림을 받을 이벤트 유형을 선택합니다.
12. 추가를 선택합니다.

슬랙 다이렉트 메시지 구성

CodeCatalyst 프로젝트가 Slack 채널에 알림을 보내도록 구성된 경우 이러한 알림을 다이렉트 메시지(DM) 로 전송할 수도 있습니다. 알림을 DM으로 직접 보내면 담당자가 맡은 프로젝트에서 발생하는 이벤트에 대한 인식을 높이는 데 도움이 됩니다. DM을 활성화하려면 Slack 멤버 ID를 추가해야 합니다. CodeCatalyst

Slack 다이렉트 메시지를 구성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 상단 메뉴 바에서 프로필 배지를 선택한 다음 내 설정을 선택합니다. CodeCatalyst 내 설정 페이지가 열립니다.

> **Tip**
>
> 프로젝트 또는 스페이스의 구성원 페이지로 이동한 다음 구성원 목록에서 이름을 선택하여 사용자 프로필을 찾을 수도 있습니다.

3. 개인용 슬랙 알림에서 Connect Slack ID를 선택한 다음 Slack 작업 영역에 연결을 선택합니다. 별도의 창이 열립니다.

> **Tip**
>
> 스페이스 관리자 역할을 가진 사용자가 스페이스에 Slack 작업 영역을 추가하지 않는 한 이 옵션을 구성할 수 CodeCatalyst 없습니다. 자세한 내용은 슬랙 알림 시작하기 및 Slack 채널에 알림 보내기 섹션을 참조하세요.

4. 권한 요청 창에서 작업 영역 이름이 공간에 구성된 Slack 작업 영역과 일치하는지 확인하십시오. CodeCatalyst 허용을 선택하여 작업 영역에 AWS Chatbot 대한 액세스를 허용하십시오. 창이 닫히고 Slack 작업 영역에 연결 상태가 연결됨으로 표시됩니다.

> **Tip**
>
> 연결 상태가 변경되지 않는 경우 Slack 작업 공간을 연결하는 중 오류가 발생했는지 확인하세요. 위로 스크롤해야 오류가 표시될 수 있습니다.

5. 개인 Slack 알림 수신을 중단하려면 연결된 Slack 작업 영역을 선택한 다음 Slack ID 연결 해제를 선택합니다.

알림 채널에 대한 알림 편집

알림이 전송되는 채널을 변경하고 특정 알림을 완전히 끌 수 있습니다.

알림을 수정하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다.
3. 탐색 창에서 프로젝트 설정을 선택합니다.
4. 알림 탭을 선택합니다.
5. 알림 수정을 선택합니다.
6. 다음 중 하나를 수행하십시오.
   - 특정 채널에 알림을 보내려면 드롭다운 목록에서 채널을 선택합니다.
   - 전 세계에서 알림을 끄려면 알림 옆의 토글을 선택합니다.
   - 특정 채널로의 알림 전송을 중지하려면 채널에서 X를 선택합니다.
7. 저장을 선택합니다.

채널 제거

CodeCatalystAmazon에서 Slack 채널을 제거할 수 있습니다. Slack 채널을 삭제하면 선택한 CodeCatalyst 프로젝트에 대한 알림이 더 이상 채널로 전송되지 않습니다.

채널 삭제하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다. 탐색 창에서 프로젝트 설정을 선택합니다.

3. 프로젝트 설정 페이지에서 알림 탭을 선택합니다.
4. 삭제하려는 채널 옆의 표시기를 선택한 다음 채널 삭제를 선택합니다. 확인 창에서 확인을 선택합니다.

CodeCatalyst 블루프린트로 프로젝트 설정

블루프린트는 프로젝트의 아키텍처 구성 요소를 나타내는 임의 코드 생성기입니다. CodeCatalyst 컴포넌트는 단일 파일의 워크플로우부터 샘플 코드가 포함된 전체 프로젝트에 이르기까지 모든 것으로 구성될 수 있습니다. 블루프린트는 임의의 옵션 세트를 사용하여 프로젝트로 전달되는 임의의 출력 코드 세트를 생성하는 데 사용합니다. 블루프린트가 최신 베스트 프랙티스나 새 옵션으로 업데이트되면 해당 블루프린트가 포함된 프로젝트에서 코드베이스의 관련 부분을 다시 생성할 수 있습니다.

Amazon CodeCatalyst 블루프린트를 사용하여 소스 리포지토리, 샘플 소스 코드, CI/CD 워크플로, 빌드 및 테스트 보고서, 통합된 문제 추적 도구가 포함된 전체 프로젝트를 생성할 수 있습니다. CodeCatalyst 블루프린트는 설정된 구성 파라미터를 기반으로 리소스와 소스 코드를 생성합니다. CodeCatalyst-managed 블루프린트를 사용하는 경우, 선택한 블루프린트에 따라 프로젝트에 추가할 리소스와 CodeCatalyst 생성 또는 구성하는 도구가 결정되므로 프로젝트 리소스를 추적하고 사용할 수 있습니다. 블루프린트 사용자는 블루프린트로 프로젝트를 만들거나 기존 프로젝트에 추가할 수 있습니다. CodeCatalyst 프로젝트에 여러 블루프린트를 추가할 수 있으며, 각 블루프린트를 독립 컴포넌트로 적용할 수 있습니다. 예를 들어, 웹 애플리케이션 블루프린트로 프로젝트를 생성한 다음 나중에 보안 블루프린트를 추가할 수 있습니다. 청사진 중 하나가 업데이트되면 수명 주기 관리를 통해 변경 사항이나 수정 사항을 프로젝트에 통합할 수 있습니다. 자세한 내용은 CodeCatalyst 청사진이 포함된 포괄적인 프로젝트 만들기 및 블루프린트 사용자로서 라이프사이클 관리 활용하기 단원을 참조하세요.

청사진 작성자는 CodeCatalyst 스페이스 구성원이 프로젝트 리소스를 사용할 수 있도록 사용자 지정 청사진을 만들고 게시할 수도 있습니다. 공간 프로젝트의 특정 요구 사항을 충족하도록 사용자 지정 청사진을 개발할 수 있습니다. 스페이스의 청사진 카탈로그에 사용자 지정 청사진을 추가한 후에는 청사진을 관리하고 계속 업데이트하여 공간 프로젝트가 최신 모범 사례에 따라 최신 상태를 유지할 수 있습니다. 자세한 내용은 프로젝트 사용자 지정 청사진 표준화 CodeCatalyst 단원을 참조하십시오. SDK청사진과 샘플 청사진을 보려면 오픈 소스 리포지토리를 참조하세요. GitHub

이미 표준화 및 모범 사례가 마련되어 있을 수 있습니다. 사용자 지정 블루프린트를 처음부터 만들고 개발하는 대신 소스 코드가 있는 기존 소스 리포지토리를 사용자 지정 블루프린트로 변환할 수 있습니다. 자세한 내용은 소스 리포지토리를 커스텀 블루프린트로 전환 단원을 참조하십시오.

주제

- 프로젝트의 블루프린트 버전 변경
- 프로젝트의 블루프린트에 대한 설명 편집하기
- 블루프린트 사용자로서 라이프사이클 관리 활용하기
- CodeCatalyst 청사진이 포함된 포괄적인 프로젝트 만들기
- 프로젝트 사용자 지정 청사진 표준화 CodeCatalyst
- 청사진 할당량 CodeCatalyst

블루프린트로 프로젝트 만들기

Amazon 청사진 카탈로그의 청사진 또는 사용자 지정 CodeCatalyst 청사진이 포함된 팀의 우주 카탈로그를 사용하여 프로젝트를 빠르게 생성할 수 있습니다. 블루프린트에 따라 특정 리소스로 프로젝트가 생성됩니다. 또한 새 프로젝트를 만들거나 기존 프로젝트에 구성 요소를 추가할 때 제너레이티브 AI 어시스턴트인 Amazon Q와 협업할 수 있습니다. 채팅과 유사한 인터페이스로 Amazon Q와 상호 작용하여 프로젝트에 대한 요구 사항을 Amazon Q에 제공할 수 있습니다. Amazon Q는 요구 사항에 따라 청사진을 제안하고 충족할 수 없는 요구 사항도 설명합니다. 그런 다음 만족하면 Amazon Q의 제안을 진행할 수 있습니다. 그러면 요구 사항에 맞는 코드가 포함된 소스 리포지토리와 같은 필요한 리소스가 생성됩니다. 자세한 내용은 블루프린트로 프로젝트 생성, Amazon Q를 사용하여 프로젝트를 생성하거나 블루프린트로 기능을 추가하는 모범 사례, CodeCatalyst 청사진이 포함된 포괄적인 프로젝트 만들기 단원을 참조하세요.

프로젝트를 생성한 후 CodeCatalyst 카탈로그 또는 스페이스 카탈로그에서 사용자 지정 청사진을 사용하여 CodeCatalyst 프로젝트에 청사진을 더 추가할 수 있습니다. 블루프린트는 아키텍처 구성 요소를 나타내므로 프로젝트에 여러 청사진을 함께 사용하여 팀의 모범 사례를 통합할 수 있습니다. 또한 진화하는 구성 요소의 최신 변경 사항을 반영하여 프로젝트를 최신 상태로 유지할 수 있습니다. 프로젝트에서 블루프린트를 사용하는 방법에 대해 자세히 알아보려면 을 참조하십시오. 블루프린트 사용자로서 라이프사이클 관리 활용하기

프로젝트에 블루프린트를 추가하여 리소스를 통합하기

프로젝트에 여러 청사진을 추가하여 기능적 구성 요소, 리소스 및 거버넌스를 통합할 수 있습니다. 프로젝트는 별도의 청사진에서 독립적으로 관리되는 다양한 요소를 지원할 수 있습니다. 프로젝트에 블루프린트를 추가하면 리소스를 수동으로 생성하고 소프트웨어 구성 요소를 작동시킬 필요가 줄어듭니다. 또한 요구 사항이 변화하더라도 프로젝트를 최신 상태로 유지할 수 있습니다. 프로젝트에 블루프린트를 추가하는 방법에 대해 자세히 알아보려면 을 참조하십시오. 블루프린트 사용자로서 라이프사이클 관리 활용하기

블루프린트의 세부 정보를 구성하는 동안 블루프린트의 소스 코드를 선호하는 타사 리포지토리에 저장하도록 선택할 수도 있습니다. 그러면 청사진을 관리하고 라이프사이클 관리 기능을 활용하여 프로젝트를 최신 상태로 유지할 수 있습니다. 자세한 내용은 확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst 및 블루프린트 사용자로서 라이프사이클 관리 활용하기 단원을 참조하세요.

Important

CodeCatalyst 프로젝트에 청사진을 추가하려면 스페이스에서 스페이스 관리자, 고급 사용자 또는 프로젝트 관리자 역할을 가진 계정으로 로그인해야 합니다.

Tip

프로젝트에 블루프린트를 추가한 후, 블루프린트의 최신 변경 사항에 대한 업데이트를 제공하도록 이메일과 Slack 알림을 구성할 수 있습니다. 자세한 내용은 에서 알림 보내기 CodeCatalyst 단원을 참조하십시오.

프로젝트에 블루프린트를 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 스페이스로 이동한 다음 블루프린트를 추가할 프로젝트를 선택합니다.
3. 탐색 창에서 블루프린트를 선택한 다음 블루프린트 추가를 선택합니다.

   Tip

   Amazon Q에서 청사진을 제안하도록 하기 위한 프로젝트 요구 사항을 Amazon Q에 제공하여 청사진을 추가할 수 있습니다. 자세한 내용은 프로젝트를 생성하거나 기능을 추가할 때 Amazon Q를 사용하여 청사진을 선택합니다. 및 Amazon Q를 사용하여 프로젝트를 생성하거나 블루프린트로 기능을 추가하는 모범 사례 단원을 참조하세요. 이 기능은 미국 서부 (오레곤) 지역에서만 사용할 수 있습니다.
   이 기능을 사용하려면 해당 공간에 제너레이티브 AI 기능을 활성화해야 합니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.
4. 블루프린트 탭에서 블루프린트를 선택하거나 스페이스 CodeCatalyst 블루프린트 탭에서 커스텀 블루프린트를 선택한 후 다음을 선택합니다.

5. 블루프린트 디테일 아래, 타겟 버전 드롭다운 메뉴에서 블루프린트 버전을 선택합니다. 최신 카탈로그 버전이 자동으로 선택됩니다.
6. (선택 사항) 기본적으로 블루프린트로 만든 소스 코드는 CodeCatalyst 저장소에 저장됩니다. 또는 블루프린트의 소스 코드를 타사 리포지토리에 저장하도록 선택할 수 있습니다. 자세한 내용은 확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst 단원을 참조하십시오.

   사용하려는 타사 저장소 제공자에 따라 다음 중 하나를 수행하십시오.

   - GitHub 리포지토리: 계정을 연결합니다. GitHub

     고급 드롭다운 메뉴를 선택하고 리포지토리 GitHub 공급자로 선택한 다음 블루프린트에서 만든 소스 코드를 저장할 GitHub 계정을 선택합니다.

     > **Note**
     >
     > GitHub 계정 연결을 사용하는 경우 개인 연결을 생성하여 ID와 ID 간에 ID 매핑을 설정해야 합니다. CodeCatalyst GitHub 자세한 내용은 개인 연결 및 개인적인 연결을 통해 GitHub 리소스에 접근하기 단원을 참조하세요.

   - 비트버킷 리포지토리: Bitbucket 작업 공간을 연결합니다.

     고급 드롭다운 메뉴를 선택하고 Bitbucket을 리포지토리 공급자로 선택한 다음 블루프린트로 만든 소스 코드를 저장할 Bitbucket 작업 공간을 선택합니다.

   - GitLab 리포지토리: 사용자를 연결합니다. GitLab

     고급 드롭다운 메뉴를 선택하고 리포지토리 GitLab 제공자로 선택한 다음 블루프린트에서 만든 소스 코드를 저장할 GitLab 사용자를 선택합니다.
7. 블루프린트 구성에서 블루프린트 파라미터를 구성합니다. 블루프린트에 따라 소스 리포지토리의 이름을 지정하는 옵션이 있을 수 있습니다.
8. 현재 블루프린트 버전과 업데이트된 버전 간의 차이점을 검토하세요. 풀 리퀘스트에 표시된 차이는 현재 버전과 풀 리퀘스트 생성 당시 원하는 버전인 최신 버전 간의 변경 사항을 보여줍니다. 변경 사항이 표시되지 않으면 버전이 동일하거나 현재 버전과 원하는 버전 모두에 대해 동일한 버전을 선택했을 수 있습니다.
9. 풀 리퀘스트에 검토하려는 코드와 변경 사항이 포함되어 있다고 생각되면 Add blueprint (블루프린트 추가) 를 선택합니다. 풀 리퀘스트를 생성한 후 코멘트를 추가할 수 있습니다. 풀 리퀘스트나 파일의 개별 라인, 전체 풀 리퀘스트에 댓글을 추가할 수 있습니다. @기호와 파일 이름을 차례로 사용하여 파일 등의 리소스에 링크를 추가할 수 있습니다.

> **Note**
>
> 풀 리퀘스트가 승인되고 병합되기 전까지는 블루프린트가 적용되지 않습니다. 자세한 내용은 풀 리퀘스트 검토 및 풀 리퀘스트 병합 단원을 참조하세요.

블루프린트 작성자는 새 프로젝트를 만들거나 기존 프로젝트에 추가하는 데 사용할 블루프린트가 없는 지정된 공간의 프로젝트에 커스텀 블루프린트를 추가할 수도 있습니다. 자세한 내용은 지정된 공간 및 프로젝트에 커스텀 블루프린트 게시 및 추가 단원을 참조하십시오.

블루프린트에 대한 업데이트를 더 이상 받고 싶지 않다면, 블루프린트를 프로젝트에서 분리하면 됩니다. 자세한 내용은 블루프린트를 프로젝트에서 분리하여 업데이트 중지 단원을 참조하십시오.

블루프린트를 프로젝트에서 분리하여 업데이트 중지

블루프린트에서 새 업데이트를 하고 싶지 않다면 프로젝트에서 블루프린트를 분리하면 됩니다. 블루프린트에서 프로젝트에 추가된 리소스 및 기능 소프트웨어 구성 요소는 프로젝트에 그대로 남아 있습니다.

> **Important**
>
> CodeCatalyst 프로젝트에서 블루프린트를 분리하려면 스페이스에서 스페이스 관리자, 고급 사용자 또는 프로젝트 관리자 역할을 가진 계정으로 로그인해야 합니다.

블루프린트를 프로젝트에서 분리하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 스페이스로 이동한 다음 블루프린트를 분리하려는 프로젝트를 선택합니다.
3. 탐색 창에서 [블루프린트(Blueprints)]를 선택합니다.
4. 연결 해제하려는 리소스가 있는 블루프린트를 선택하고, 액션 드롭다운 메뉴를 선택한 다음, 블루프린트 분리 (Disassociate Blueprint) 를 선택합니다.
5. 를 입력하여 연결 `confirm` 해제를 확인합니다.
6. 확인(Confirm)을 선택합니다.

프로젝트의 블루프린트 버전 변경

블루프린트로 프로젝트를 생성하거나 기존 프로젝트에 블루프린트를 추가한 경우, 블루프린트의 새 버전에 대한 알림을 받게 됩니다. 승인된 풀 리퀘스트를 통해 블루프린트 버전을 업데이트하기 전에 코드 변경 사항과 영향을 받는 환경을 확인할 수 있습니다. 라이프사이클 관리를 사용하면 프로젝트에 적용된 하나 이상의 블루프린트 버전을 변경할 수 있으므로 프로젝트의 다른 영역에 영향을 주지 않고 각 블루프린트 버전을 변경할 수 있습니다. 블루프린트 업데이트를 오버라이드할 수도 있습니다. 자세한 내용은 블루프린트 사용자로서 라이프사이클 관리 활용하기 단원을 참조하십시오.

Important

CodeCatalyst 프로젝트에 있는 블루프린트 버전을 변경하려면 스페이스에서 스페이스 관리자, 파워 사용자 또는 프로젝트 관리자 역할을 가진 계정으로 로그인해야 합니다.

Tip

프로젝트에 블루프린트를 추가한 후, 블루프린트의 최신 변경 사항에 대한 업데이트를 제공하도록 이메일과 Slack 알림을 구성할 수 있습니다. 자세한 내용은 에서 알림 보내기 CodeCatalyst 단원을 참조하십시오.

블루프린트를 최신 버전으로 업데이트하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 블루프린트 버전을 업데이트하려는 공간으로 이동합니다.
3. 스페이스 대시보드에서 업데이트하려는 블루프린트가 있는 프로젝트를 선택합니다.
4. 탐색 창에서 [블루프린트(Blueprints)]를 선택합니다.
5. 상태 열에서 카탈로그 버전 변경 링크 (예: 카탈로그 버전 0.3.109 변경) 를 선택합니다.
6. 작업 드롭다운 메뉴를 선택한 다음 버전 업데이트를 선택합니다. 최신 버전이 자동으로 선택됩니다.
7. (선택 사항) 블루프린트 구성에서 블루프린트 매개변수를 구성합니다.
8. (선택 사항) 코드 변경 탭에서 현재 블루프린트 버전과 업데이트된 버전 간의 차이점을 검토하세요. 풀 리퀘스트에 표시되는 차이는 현재 버전과 최신 버전 (풀 리퀘스트 생성 시 원하는 버전) 간의 변경입니다. 변경 사항이 표시되지 않으면 버전이 동일하거나 현재 버전과 원하는 버전 모두에 대해 동일한 버전을 선택했을 수 있습니다.

9. 풀 리퀘스트에 검토하려는 코드와 변경 사항이 포함되어 있다고 확인되면 업데이트 적용을 선택합니다. 풀 리퀘스트를 생성한 후 코멘트를 추가할 수 있습니다. 풀 리퀘스트나 파일의 개별 라인, 전체 풀 리퀘스트에 댓글을 추가할 수 있습니다. @기호와 파일 이름을 차례로 사용하여 파일 등의 리소스에 링크를 추가할 수 있습니다.

> **Note**
>
> 풀 리퀘스트가 승인되고 병합될 때까지 블루프린트는 업데이트되지 않습니다. 자세한 내용은 풀 리퀘스트 검토 및 풀 리퀘스트 병합 단원을 참조하세요.

> **Note**
>
> 블루프린트 버전 업데이트를 위한 기존 풀 리퀘스트가 열려 있는 경우, 새 풀 리퀘스트를 생성하기 전에 이전 풀 리퀘스트를 닫으세요. Update version (버전 업데이트) 를 선택하면 블루프린트에 대해 보류 중인 풀 리퀘스트 목록으로 이동합니다. 프로젝트 세팅의 블루프린트 탭과 프로젝트 요약 페이지에서도 보류 중인 풀 리퀘스트를 볼 수 있습니다. 자세한 내용은 풀 리퀘스트 보기 단원을 참조하십시오.

블루프린트 버전 변경하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 블루프린트 버전을 업데이트하려는 공간으로 이동합니다.
3. 스페이스 대시보드에서 업데이트하려는 블루프린트가 있는 프로젝트를 선택합니다.
4. 탐색 창에서 블루프린트를 선택한 다음 업데이트하려는 블루프린트의 라디오 버튼을 선택합니다.
5. 액션 드롭다운 메뉴를 선택한 다음 블루프린트 구성을 선택합니다.
6. 타겟 버전 드롭다운 메뉴에서 사용하려는 버전을 선택합니다. 최신 버전이 자동으로 선택됩니다.
7. (선택 사항) 블루프린트 구성에서 블루프린트 매개변수를 구성합니다.
8. (선택 사항) 코드 변경 탭에서 현재 블루프린트 버전과 업데이트된 버전 간의 차이점을 검토하세요. 풀 리퀘스트에 표시되는 차이는 현재 버전과 최신 버전 (풀 리퀘스트 생성 시 원하는 버전) 간의 변경입니다. 변경 사항이 표시되지 않으면 버전이 동일하거나 현재 버전과 원하는 버전 모두에 대해 동일한 버전을 선택했을 수 있습니다.
9. 풀 리퀘스트에 검토하려는 코드와 변경 사항이 포함되어 있다고 확인되면 업데이트 적용을 선택합니다. 풀 리퀘스트를 생성한 후 코멘트를 추가할 수 있습니다. 풀 리퀘스트나 파일의 개별 라인,

전체 풀 리퀘스트에 댓글을 추가할 수 있습니다. @기호와 파일 이름을 차례로 사용하여 파일 등의 리소스에 링크를 추가할 수 있습니다.

> **Note**
>
> 풀 리퀘스트가 승인되고 병합될 때까지 블루프린트는 업데이트되지 않습니다. 자세한 내용은 풀 리퀘스트 검토 및 풀 리퀘스트 병합 단원을 참조하세요.

> **Note**
>
> 블루프린트 버전 업데이트를 위한 기존 풀 리퀘스트가 열려 있는 경우, 새 풀 리퀘스트를 생성하기 전에 이전 풀 리퀘스트를 닫으세요. Update version (버전 업데이트) 를 선택하면 블루프린트에 대해 보류 중인 풀 리퀘스트 목록으로 이동합니다. 프로젝트 세팅의 블루프린트 탭과 프로젝트 요약 페이지에서도 보류 중인 풀 리퀘스트를 볼 수 있습니다. 자세한 내용은 풀 리퀘스트 보기 단원을 참조하십시오.

프로젝트의 블루프린트에 대한 설명 편집하기

프로젝트를 생성할 때 사용했거나 프로젝트가 생성된 후에 적용한 블루프린트의 설명을 편집할 수 있습니다. 블루프린트는 프로젝트에서 두 번 이상 사용할 수 있습니다. 프로젝트에서 블루프린트의 용도를 구분하기 위해 해당 블루프린트에 대한 설명을 사용할 수 있습니다. 설명을 사용하여 특정 블루프린트에서 추가하려는 구성 요소를 식별할 수도 있습니다.

> **Important**
>
> 스페이스에서 사용자 지정 청사진의 설명을 편집하려면 스페이스에서 스페이스 관리자, 고급 사용자 또는 프로젝트 관리자 역할을 가진 계정으로 로그인해야 합니다.

프로젝트에서 블루프린트 설명을 편집하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 스페이스로 이동한 다음 업데이트하려는 블루프린트 설정이 있는 프로젝트를 선택합니다.
3. 탐색 창에서 [블루프린트(Blueprints)]를 선택합니다.

4. 업데이트하려는 설명이 있는 블루프린트를 선택하고 액션 드롭다운 메뉴를 선택한 다음 설정을 선택합니다.
5. 블루프린트 설명 텍스트 입력 필드에 프로젝트의 블루프린트를 식별하는 설명을 입력합니다.
6. 저장(Save)을 선택합니다.

블루프린트 사용자로서 라이프사이클 관리 활용하기

수명 주기 관리란 블루프린트의 업데이트된 옵션 또는 버전으로 코드베이스를 재생성하는 기능입니다. 이를 통해 블루프린트 작성자는 특정 블루프린트가 포함된 모든 프로젝트의 소프트웨어 개발 라이프사이클을 중앙에서 관리할 수 있습니다. 예를 들어 웹 애플리케이션 블루프린트에 보안 수정 사항을 푸시하면 웹 애플리케이션 블루프린트를 포함하거나 웹 애플리케이션 블루프린트에서 생성한 모든 프로젝트가 해당 수정 사항을 자동으로 적용할 수 있습니다. 또한 동일한 관리 프레임워크를 통해 블루프린트 사용자는 블루프린트 선택 후 블루프린트 옵션을 변경할 수 있습니다.

주제

기존 프로젝트에 라이프사이클 관리 사용

블루프린트에서 생성한 프로젝트 또는 블루프린트와 연관되지 않은 기존 프로젝트에 라이프사이클 관리를 사용할 수 있습니다. 예를 들어, 블루프린트에서 생성되지 않은 표준 보안 관행 블루프린트를 five-year-old Java 애플리케이션에 추가할 수 있습니다. 블루프린트는 보안 스캔 워크플로우 및 기타 관련 코드를 생성합니다. 이제 블루프린트가 변경될 때마다 팀의 모범 사례에 따라 Java 애플리케이션의 코드베이스 부분이 자동으로 최신 상태로 유지됩니다.

프로젝트 내 여러 블루프린트에서 라이프사이클 관리 사용

청사진은 아키텍처 구성 요소를 나타내므로 동일한 프로젝트에서 여러 청사진을 함께 사용할 수 있는 경우가 많습니다. 예를 들어, 프로젝트는 회사 플랫폼 엔지니어가 구축한 중앙 웹 API 블루프린트와 앱 보안 팀이 구축한 릴리스 체크 블루프린트로 구성될 수 있습니다. 각 블루프린트는 독립적으로 업데이트할 수 있으며 과거에 적용된 병합 결의안도 기억할 것입니다.

> **Note**
>
> 임의의 아키텍처 구성 요소이기 때문에 모든 블루프린트가 함께 의미가 있거나 논리적으로 함께 작동하는 것은 아닙니다. 비록 여전히 서로 병합을 시도하긴 하지만 말입니다.

라이프사이클 풀 리퀘스트의 충돌 문제 해결하기

라이프사이클 풀 요청으로 인해 병합 충돌이 발생하는 경우가 있습니다. 이러한 문제는 수동으로 해결할 수 있습니다. 해상도는 후속 블루프린트 업데이트 시 기억됩니다.

라이프사이클 관리 변경 옵트아웃

사용자는 프로젝트에서 블루프린트를 삭제하여 블루프린트에 대한 모든 참조를 분리하고 라이프사이클 업데이트를 옵트아웃할 수 있습니다. 안전상의 이유로, 이렇게 해도 블루프린트에서 추가된 내용을 포함하여 프로젝트의 코드나 리소스가 제거되거나 영향을 주지는 않습니다. 자세한 내용은 블루프린트를 프로젝트에서 분리하여 업데이트 중지 단원을 참조하십시오.

프로젝트에서 블루프린트의 라이프사이클 관리 재정의하기

프로젝트의 특정 파일에 대한 블루프린트 업데이트를 오버라이드하려면 리포지토리에 소유권 파일을 포함하면 됩니다. GitLab의 코드 소유자 사양은 권장 지침입니다. 블루프린트는 항상 다른 모든 것보다 코드 소유자 파일을 존중하며 다음과 같은 샘플 파일을 생성할 수 있습니다.

```
new BlueprintOwnershipFile(sourceRepo, {
      resynthesis: {
        strategies: [
          {
            identifier: 'dont-override-sample-code',
            description: 'This strategy is applied accross all sample code. The
 blueprint will create sample code, but skip attempting to update it.',
            strategy: MergeStrategies.neverUpdate,
            globs: [
              '**/src/**',
              '**/css/**',
            ],
          },
        ],
      },
    });
```

그러면 다음과 같은 내용이 .ownership-file 포함된 a가 생성됩니다.

```
[dont-override-sample-code] @amazon-codecatalyst/blueprints.import-from-git
# This strategy is applied accross all sample code. The blueprint will create sample
 code, but skip attempting to update it.
# Internal merge strategy: neverUpdate
**/src/**
**/css/**
```

CodeCatalyst 청사진이 포함된 포괄적인 프로젝트 만들기

블루프린트를 사용하여 프로젝트를 생성하면 소스 리포지토리, 샘플 소스 코드, CI/CD 워크플로, 빌드 및 테스트 보고서, 통합된 문제 추적 도구가 포함된 전체 프로젝트를 CodeCatalyst 생성합니다. 프로젝트 블루프린트는 코드를 사용하여 다양한 유형의 애플리케이션 및 프레임워크에 대한 클라우드 인프라, 리소스 및 샘플 소스 아티팩트를 프로비저닝합니다.

자세한 내용은 프로젝트 생성 단원을 참조하십시오. 프로젝트를 생성하려면 스페이스 관리자여야 합니다.

주제

- 사용 가능한 블루프린트
- 프로젝트 청사진 정보 찾기

사용 가능한 블루프린트

블루프린트 이름	블루프린트 설명
ASP. NET코어 웹 API	이 청사진은 a를 생성합니다. NET6ASP. NET 코어 웹 API 애플리케이션. 블루프린트는 AWS 디플로이먼트 툴을 사용합니다. NETAmazon Elastic Container Service를 구성하거나 AWS Elastic Beanstalk 배포 대상으로 구성할 수 있는 옵션을 제공합니다. AWS App Runner
AWSGlue ETL	이 블루프린트는 AWS Glue AWS CDK, AWS Lambda 및 Amazon Athena를 사용하여 샘플 추출 변환 load (ETL) 참조 구현을 생성하여 쉼표

블루프린트 이름	블루프린트 설명
	로 구분된 값 () 을 Apache Parquet으로 변환합니다. CSVs
DevOps 배포 파이프라인	이 블루프린트는 참조 애플리케이션을 여러 단계에 AWS 걸쳐 배포하는 AWS 배포 파이프라인 참조 아키텍처를 사용하여 배포 파이프라인을 생성합니다.
Java: API AWS Fargate	이 블루프린트는 컨테이너화된 웹 서비스 프로젝트를 생성합니다. 이 프로젝트는 AWS CLICopilot을 사용하여 Amazon에서 Amazon DynamoDB를 기반으로 하는 컨테이너식 스프링 부트 자바 웹 서비스를 구축하고 배포합니다. ECS 이 프로젝트는 컨테이너식 앱을 서버리스 컴퓨팅의 Amazon ECS 클러스터에 AWS Fargate 배포합니다. 앱은 DynamoDB 테이블에 데이터를 저장합니다. 워크플로가 성공적으로 실행되면 샘플 웹 서비스는 Application Load Balancer를 통해 공개적으로 사용할 수 있습니다.
최신 3계층 웹 애플리케이션	이 블루프린트는 애플리케이션 계층 및 Vue 프론트엔드 프레임워크용 코드를 Python으로 생성하여 잘 설계된 3계층 최신 웹 애플리케이션을 빌드하고 배포합니다.
. NET서버리스 애플리케이션	이 블루프린트는 를 사용하여 AWS Lambda 함수를 생성합니다. NETCLILambda 도구. 블루프린트는 C# 또는 F# 선택을 포함하여 AWS Lambda 함수에 대한 옵션을 제공합니다.

블루프린트 이름	블루프린트 설명
Node.js, API AWS Fargate	이 블루프린트는 컨테이너화된 웹 서비스 프로젝트를 생성합니다. 이 프로젝트는 AWS CLICopilot을 사용하여 Amazon Elastic Container Service에서 컨테이너식 Express/Node.js 웹 서비스를 구축하고 배포합니다. 이 프로젝트는 컨테이너식 앱을 서버리스 컴퓨팅의 Amazon ECS 클러스터에 AWS Fargate 배포합니다. 워크플로가 성공적으로 실행되면 샘플 웹 서비스는 Application Load Balancer를 통해 공개적으로 사용할 수 있습니다.
서버리스 애플리케이션 모델 () SAM	이 블루프린트는 서버리스 애플리케이션 모델 (SAM) 을 사용하여 생성 및 배포하는 프로젝트를 생성합니다. API 프로그래밍 언어로 Java 또는 SDK Python을 선택할 SDK 수 있습니다. TypeScript
서버리스 이미지 핸들러	이 청사진은 이미지 품질을 저하시키지 않으면서 고속 이미지 처리를 위한 애플리케이션을 만듭니다.
서버리스 마이크로서비스 RESTful	이 블루프린트는 To REST API Do 서비스 AWS Lambda Amazon API Gateway 참조를 사용하는 것을 생성합니다. 프로그래밍 언어로 Java 또는 SDK Python을 선택할 SDK 수 있습니다. TypeScript
단일 페이지 애플리케이션	이 블루프린트는 React, Vue, Angular SPA 프레임워크를 사용하는 단일 페이지 애플리케이션 () 을 만듭니다. 호스팅의 경우 AWS Amplify 호스팅 또는 Amazon CloudFront Amazon S3에서 선택합니다.

블루프린트 이름	블루프린트 설명
정적 웹 사이트	이 블루프린트는 Hugo 또는 Jekyll 정적 사이트 생성기를 사용하여 정적 웹 사이트를 만듭니다. 정적 사이트 생성기는 텍스트 입력 파일 (예: Markdown) 을 사용하여 정적 웹 페이지를 생성합니다. 제품 페이지, 설명서, 블로그와 같이 거의 변경되지 않는 정보를 제공하는 콘텐츠에 적합합니다. 블루프린트는 AWS CDK 를 사용하여 정적 웹 페이지를 둘 중 하나 AWS Amplify 또는 Amazon CloudFront S3+에 배포합니다.
웹 애플리케이션을 실행하려면	이 청사진은 프론트엔드와 백엔드 구성 요소가 포함된 To Do 서버리스 웹 애플리케이션을 만듭니다. 프로그래밍 언어로 Java 또는 SDK Python을 선택할 SDK 수 있습니다. TypeScript
V ideo-on-demand 웹 서비스	이 청사진은 콘텐츠를 받아 들여 트랜스코딩하고 전송하는 기능을 제공하는 video-on-demand 서비스를 만듭니다. 블루프린트는 Amazon S3 AWS Lambda Amazon CloudWatch, 및 AWS Elemental MediaConvert를 사용합니다.
외부 블루프린트를 구독하세요	이 블루프린트는 가져온 각 패키지에 대한 워크플로를 생성합니다. 이 워크플로는 하루에 한 번 실행되어 새 버전의 패키지가 있는지 확인합니다NPM. 새 버전이 있는 경우 워크플로는 해당 버전을 CodeCatalyst 공간에 사용자 지정 청사진으로 추가하려고 합니다. 패키지를 찾을 수 없거나 블루프린트가 아닌 경우 작업이 실패합니다. 대상 패키지가 켜져 NPM 있어야 하고 패키지는 블루프린트여야 합니다. 스페이스는 커스텀 블루프린트를 지원하는 등급에 가입해야 합니다.

블루프린트 이름	블루프린트 설명
베드락 GenAI 챗봇	이 블루프린트는 Amazon Bedrock과 Anthropic의 Claude를 사용하여 생성형 AI 챗봇을 만듭니다. 이 블루프린트를 사용하면 데이터에 맞게 사용자 지정할 수 있는 안전한 로그인 보호 플레이그라운드를 구축하고 배포할 수 있습니다. LLM 자세한 내용은 베드락 GenAI 챗봇 문서를 참조하십시오.
AWS프로젝트 개발 키트 () 청사진 AWS PDK	이러한 PDK 블루프린트를 함께 구성하여 React 웹 사이트, Smithy API 및 이를 배포할 지원 CDK 인프라로 구성된 애플리케이션을 만들 수 있습니다. AWS 프로젝트를 관리하고 빌드하기 위한 개발 도구와 함께 공통 패턴을 위한 구성 요소를 AWS PDK 제공합니다. 자세한 내용은 AWSPDK GitHub 소스 리포지토리 및 을 참조하십시오튜토리얼: 컴포저블 PDK 블루프린트로 풀스택 애플리케이션 만들기.

프로젝트 청사진 정보 찾기

에서 여러 프로젝트 청사진을 사용할 수 있습니다. CodeCatalyst 각 청사진에는 요약 및 파일이 함께 제공됩니다. README 요약에는 블루프린트로 설치되는 리소스가 설명되고, README 파일은 블루프린트에 대해 자세히 설명하고 사용 방법에 대한 지침을 제공합니다.

프로젝트 사용자 지정 청사진 표준화 CodeCatalyst

사용자 지정 청사진을 사용하여 CodeCatalyst 공간 프로젝트의 개발 및 모범 사례를 표준화할 수 있습니다. 사용자 지정 청사진을 사용하여 워크플로 정의 및 애플리케이션 코드와 같은 CodeCatalyst 프로젝트의 다양한 측면을 정의할 수 있습니다. 커스텀 블루프린트를 사용하여 새 프로젝트를 만들거나 기존 프로젝트에 적용한 후에는 블루프린트에 대한 모든 변경 사항을 풀 리퀘스트 업데이트로 해당 프로젝트에 사용할 수 있습니다. 청사진 작성자는 공간 전체에서 청사진을 사용하는 프로젝트에 대한 세부 정보를 볼 수 있으므로 프로젝트 전반에 표준이 어떻게 적용되고 있는지 확인할 수 있습니다. 블루프린트의 라이프사이클 관리를 통해 모든 프로젝트의 소프트웨어 개발 라이프사이클을 중앙에서 관리할

수 있으므로, 해당 분야의 프로젝트가 최신 변경 또는 수정과 함께 모범 사례를 계속 따르도록 할 수 있습니다. 자세한 내용은 블루프린트 작성자로서 라이프사이클 관리와 협력하기 단원을 참조하십시오.

사용자 지정 블루프린트는 재합성을 통해 이전 프로젝트에 맞게 블루프린트 버전을 업데이트할 수 있는 기능을 제공합니다. 재합성은 업데이트된 버전으로 블루프린트 합성을 다시 실행하거나 수정 및 변경 사항을 기존 프로젝트에 통합하는 기능을 사용하여 블루프린트 합성을 다시 실행하는 프로세스입니다. 자세한 내용은 커스텀 블루프린트 컨셉 단원을 참조하십시오.

이미 표준화 및 모범 사례가 마련되어 있을 수 있습니다. 사용자 지정 블루프린트를 처음부터 만들고 개발하는 대신 소스 코드가 있는 기존 소스 리포지토리를 사용자 지정 블루프린트로 변환할 수 있습니다. 자세한 내용은 소스 리포지토리를 커스텀 블루프린트로 전환 단원을 참조하십시오.

SDK블루프린트와 샘플 블루프린트를 보려면 오픈 소스 리포지토리를 참조하세요. GitHub

주제

커스텀 블루프린트 컨셉

에서 커스텀 블루프린트로 작업할 때 알아두어야 할 몇 가지 개념과 용어를 소개합니다. CodeCatalyst

주제

블루프린트 프로젝트

블루프린트 프로젝트를 사용하면 블루프린트를 개발하여 공간에 게시할 수 있습니다. 소스 리포지토리는 프로젝트 생성 프로세스 중에 생성되며, 리포지토리 이름은 프로젝트 리소스 세부 정보를 입력할 때 선택한 이름입니다. 블루프린트 생성 프로세스 중에 워크플로 릴리스를 생성하기로 선택하면 블루프린트 빌더 블루프린트와 함께 블루프린트에 퍼블리싱 워크플로가 생성됩니다. 워크플로는 최신 버전을 자동으로 퍼블리시합니다.

우주 청사진

스페이스의 블루프린트 섹션으로 이동하면 스페이스 블루프린트 테이블에서 모든 블루프린트를 보고 관리할 수 있습니다. 블루프린트가 스페이스에 게시되고 나면 스페이스 청사진으로 사용할 수 있게 되어 스페이스의 청사진 카탈로그에서 추가하고 제거할 수 있습니다. 스페이스의 Blueprints 섹션에서 게시 권한을 관리하고 청사진을 삭제할 수도 있습니다. 자세한 내용은 커스텀 블루프린트의 세부 정보, 버전, 프로젝트 보기 단원을 참조하십시오.

우주 청사진 카탈로그

스페이스의 청사진 카탈로그에서 추가된 모든 사용자 지정 청사진을 볼 수 있습니다. 여기서 스페이스 구성원은 사용자 지정 청사진을 선택하여 새 프로젝트를 만들 수 있습니다. 이 카탈로그는 모든 스페이스 구성원이 사용할 수 있는 청사진이 이미 있는 CodeCatalyst 카탈로그와는 다릅니다. 자세한 내용은 CodeCatalyst 청사진이 포함된 포괄적인 프로젝트 만들기 단원을 참조하십시오.

종합

합성은 프로젝트의 소스 코드, 구성 및 리소스를 나타내는 CodeCatalyst 프로젝트 번들을 생성하는 프로세스입니다. 그런 다음 CodeCatalyst 배포 API 작업에서 번들을 사용하여 프로젝트에 배포합니다.

에서 프로젝트를 생성하지 않고도 프로젝트 생성을 에뮬레이션하는 사용자 지정 블루프린트를 개발하는 동안 프로세스를 로컬에서 실행할 수 있습니다. CodeCatalyst 다음 명령을 사용하여 합성을 수행할 수 있습니다.

```
yarn blueprint:synth             # fast mode
yarn blueprint:synth --cache     # wizard emulation mode
```

블루프린트는 해당 옵션이 병합된 상태에서 메인 `blueprint.ts` 클래스를 호출하는 것으로 저절로 시작됩니다. `defaults.json` 폴더 아래에 새 프로젝트 번들이 `synth/synth.[options-name]/proposed-bundle/` 생성됩니다. 출력에는 사용자가 설정한 옵션에 따라 커스텀 블루프린트가 생성하는 프로젝트 번들이 포함됩니다. 여기에는 사용자가 구성했을 수도 있는 부분 옵션도 포함됩니다.

재합성

재합성은 다양한 청사진 옵션 또는 기존 프로젝트의 청사진 버전을 사용하여 청사진을 재생성하는 과정입니다. 블루프린트 작성자는 커스텀 블루프린트 코드에서 커스텀 병합 전략을 정의할 수 있습니다. 또한 에서 소유권 경계를 `.ownership-file` 정의하여 코드베이스의 어느 부분에서 블루프린트를 업데이트할 수 있는지 지정할 수 있습니다. 사용자 지정 블루프린트에서 업데이트를 제안할 수 있지만 사용자 지정 블루프린트를 사용하는 프로젝트 개발자는 프로젝트의 소유권 경계를 결정할 수 있습니다. `.ownership-file` 로컬에서 재합성을 실행하고 커스텀 블루프린트를 게시하기 전에 테스트 및 업데이트할 수 있습니다. 다음 명령어를 사용하여 재합성을 수행하십시오.

```
yarn blueprint:resynth             # fast mode
yarn blueprint:resynth --cache     # wizard emulation mode
```

블루프린트는 해당 옵션이 병합된 상태에서 메인 `blueprint.ts` 클래스를 호출하는 것으로 저절로 시작됩니다. `defaults.json` 폴더 아래에 새 프로젝트 번들이 `synth/resynth.[options-name]/` 생성됩니다. 출력에는 사용자가 설정한 옵션에 따라 커스텀 블루프린트가 생성하는 프로젝트 번들이 포함됩니다. 여기에는 사용자가 구성했을 수도 있는 부분 옵션도 포함됩니다.

다음 콘텐츠는 합성 및 재합성 프로세스 후에 생성됩니다.

- 제안 번들 - 대상 블루프린트 버전에 새 옵션을 적용해 실행했을 때의 합성 결과입니다.
- 기존 번들 - 기존 프로젝트를 모방한 것입니다. 이 폴더에 아무것도 없으면 이 폴더와 동일한 출력으로 생성됩니다. `proposed-bundle`
- ancestor-bundle - 이전 버전, 이전 옵션 또는 조합을 사용하여 실행했을 때 블루프린트가 생성하는 것을 모방한 것입니다. 이 폴더에 아무것도 없으면 와 같은 출력으로 생성됩니다. `proposed-bundle`

- resolved-bundle - 번들은 항상 재생성되며,,,, 사이의 3방향 병합이 기본값입니다. proposed-bundle existing-bundle ancestor-bundle 이 번들은 재합성이 로컬에서 출력하는 결과를 에뮬레이션합니다.

블루프린트 출력 번들에 대한 자세한 내용은 을 참조하십시오. 재합성을 통한 파일 생성

부분 옵션

Options인터페이스 전체를 src/wizard-configuration/ 열거할 필요 없이 옵션 변형을 추가할 수 있으며 옵션은 파일 상단에 병합됩니다. defaults.json 이를 통해 특정 옵션에 맞게 테스트 사례를 조정할 수 있습니다.

예:

Options인터페이스:

```
{
  language: "Python" | "Java" | "Typescript",
  repositoryName: string
  ...
}
```

defaults.json 파일:

```
{
  language: "Python",
  repositoryName: "Myrepo"
  ...
}
```

추가 구성 테스트:

- ```
  #wizard-config-typescript-test.json
  {
    language: "Typescript",
  }
  ```
- ```
  #wizard-config-java-test.json
  {
    language: "Java",
  ```

```
}
```

프로젝트

Progen은 커스텀 블루프린트가 자체적으로 업데이트되고 일관성을 유지하는 데 사용하는 오픈 소스 도구입니다. 블루프린트가 Projen 패키지로 제공되는 이유는 이 프레임워크가 프로젝트를 빌드, 번들 및 퍼블리싱하는 기능을 제공하고 인터페이스를 사용하여 프로젝트의 구성 및 설정을 관리할 수 있기 때문입니다.

Projen을 사용하면 블루프린트를 만든 후에도 대규모로 업데이트할 수 있습니다. Projen 툴은 프로젝트 번들을 생성하는 블루프린트 합성을 뒷받침하는 기반 기술입니다. Projen은 프로젝트의 구성을 소유하므로 청사진 작성자인 여러분에게 영향을 미치지 않아야 합니다. 종속성을 추가한 후 `yarn projen` 를 실행하여 프로젝트 구성을 다시 생성하거나 파일에서 옵션을 변경할 수 있습니다. `projenrc.ts` Projen은 프로젝트를 합성하기 위한 커스텀 블루프린트의 기본 생성 도구이기도 합니다. 자세한 내용은 projen 페이지를 참조하십시오. GitHub Projen 사용에 대한 자세한 내용은 Projen 설명서를 참조하십시오.

커스텀 블루프린트 시작하기

블루프린트를 생성하는 과정에서 블루프린트를 구성하고 프로젝트 리소스의 미리보기를 생성할 수 있습니다. 각 사용자 지정 청사진은 CodeCatalyst 프로젝트에서 관리되며, 프로젝트에는 스페이스의 청사진 카탈로그에 게시하기 위한 워크플로가 기본적으로 포함되어 있습니다.

사용자 지정 청사진의 세부 정보를 구성하는 동안 청사진의 소스 코드를 타사 저장소에 저장하도록 선택할 수도 있습니다. 이 저장소에서는 사용자 지정 청사진을 관리하고 수명 주기 관리 기능을 활용하여 사용자 지정 청사진 수정 시 공간 프로젝트를 동기화된 상태로 유지할 수 있습니다. 자세한 내용은 확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst 및 블루프린트 작성자로서 라이프사이클 관리와 협력하기 섹션을 참조하세요.

표준화 및 모범 사례를 갖춘 소스 리포지토리가 이미 있는 경우 해당 소스 리포지토리를 사용자 지정 블루프린트로 변환할 수 있습니다. 자세한 정보는 소스 리포지토리를 커스텀 블루프린트로 전환을 참조하세요.

주제

- 사전 조건
- 1단계: 에서 사용자 지정 블루프린트 만들기 CodeCatalyst
- 2단계: 구성 요소가 포함된 사용자 지정 블루프린트 개발

- 3단계: 커스텀 블루프린트 미리 보기
- (선택 사항) 4단계: 커스텀 블루프린트 프리뷰 버전 퍼블리시

사전 조건

커스텀 블루프린트를 만들기 전에 다음 요구사항을 고려하세요.

- CodeCatalyst 공간은 엔터프라이즈 등급이어야 합니다. 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 청구 관리를 참조하십시오.
- 사용자 지정 블루프린트를 생성하려면 스페이스 관리자 또는 파워 사용자 역할이 있어야 합니다. 자세한 정보는 사용자 역할을 통한 액세스 권한 부여을 참조하세요.

1단계: 에서 사용자 지정 블루프린트 만들기 CodeCatalyst

스페이스 설정에서 사용자 지정 청사진을 만들면 저장소가 자동으로 생성됩니다. 저장소에는 스페이스의 청사진 카탈로그에 청사진을 게시하기 전에 청사진을 개발하는 데 필요한 모든 필수 리소스가 포함되어 있습니다.

사용자 지정 청사진을 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 사용자 지정 블루프린트를 만들려는 공간으로 이동합니다.
3. 스페이스 대시보드에서 설정 탭을 선택한 다음 블루프린트를 선택합니다.
4. 블루프린트 생성을 선택합니다.
5. 블루프린트 이름 지정에 프로젝트에 할당하려는 이름과 관련 리소스 이름을 입력합니다. 이름은 스페이스 내에서 고유해야 합니다.
6. (선택 사항) 기본적으로 블루프린트로 만든 소스 코드는 리포지토리에 저장됩니다. CodeCatalyst 또는 블루프린트의 소스 코드를 타사 리포지토리에 저장하도록 선택할 수 있습니다. 자세한 정보는 확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst을 참조하세요.

   사용하려는 서드 파티 리포지토리 제공자에 따라 다음 중 하나를 수행하십시오.

   - GitHub 리포지토리: 계정을 연결합니다. GitHub

     고급 드롭다운 메뉴를 선택하고 리포지토리 GitHub 공급자로 선택한 다음 블루프린트에서 만든 소스 코드를 저장할 GitHub 계정을 선택합니다.

> **Note**
>
> GitHub 계정 연결을 사용하는 경우 개인 연결을 생성하여 ID와 ID 간에 ID 매핑을 설정해야 합니다. CodeCatalyst GitHub 자세한 내용은 개인 연결 및 개인적인 연결을 통해 GitHub 리소스에 접근하기 섹션을 참조하세요.

- 비트버킷 리포지토리: Bitbucket 작업 공간을 연결합니다.

  고급 드롭다운 메뉴를 선택하고 Bitbucket을 리포지토리 공급자로 선택한 다음 블루프린트로 만든 소스 코드를 저장할 Bitbucket 작업 공간을 선택합니다.

- GitLab 리포지토리: 사용자를 연결합니다. GitLab

  고급 드롭다운 메뉴를 선택하고 리포지토리 GitLab 제공자로 선택한 다음 블루프린트에서 만든 소스 코드를 저장할 GitLab 사용자를 선택합니다.

7. 블루프린트 디테일 아래에서 다음과 같이 하세요.

   a. 블루프린트 디스플레이 이름 텍스트 입력 필드에 스페이스의 블루프린트 카탈로그에 표시될 이름을 입력합니다.

   b. 설명 텍스트 입력 필드에 커스텀 블루프린트에 대한 설명을 입력합니다.

   c. 저자 이름 텍스트 입력 필드에 사용자 지정 블루프린트의 저자 이름을 입력합니다.

   d. (선택 사항) 고급 설정을 선택합니다.

      i. + 추가를 선택하여 `package.json` 파일에 추가된 태그를 추가합니다.

      ii. 라이선스 드롭다운 메뉴를 선택한 다음 커스텀 블루프린트에 사용할 라이선스를 선택합니다.

      iii. 블루프린트 패키지 이름 텍스트 입력 필드에 블루프린트 패키지를 식별할 이름을 입력합니다.

      iv. 기본적으로 릴리스 워크플로는 블루프린트 빌더 (Blueprint Builder) 라는 프로젝트 내 퍼블리싱 블루프린트를 사용하여 생성됩니다. 릴리스 워크플로에서 퍼블리싱 권한이 활성화되었으므로, 워크플로는 변경사항을 푸시하면 스페이스에 최신 블루프린트 버전을 퍼블리시합니다. 워크플로우 생성을 끄려면 릴리스 워크플로 체크박스를 선택 취소하십시오.

8. (선택 사항) 블루프린트 프로젝트에는 스페이스의 블루프린트 카탈로그에 블루프린트를 게시할 수 있도록 지원하는 사전 정의된 코드가 함께 제공됩니다. 선택한 프로젝트 파라미터에 따라 업데

이트된 정의 파일을 보려면, 블루프린트 미리 보기 생성에서 코드 보기 또는 워크플로 보기를 선택합니다.

9. 블루프린트 생성을 선택합니다.

커스텀 블루프린트의 워크플로 생성을 끄지 않은 경우, 블루프린트가 생성되면 워크플로가 자동으로 실행되기 시작합니다. 워크플로 실행이 완료되면 기본적으로 사용자 지정 청사진을 스페이스의 청사진 카탈로그에 추가할 수 있습니다. 최신 블루프린트 버전을 스페이스에 자동으로 게시하지 않으려면 게시 권한을 끌 수 있습니다. 자세한 내용은 커스텀 블루프린트에 대한 퍼블리싱 권한 설정 및 워크플로 실행 섹션을 참조하세요.

라는 `blueprint-release` 퍼블리싱 워크플로는 블루프린트를 사용하여 생성되므로, 블루프린트는 프로젝트에서 적용된 블루프린트로 찾을 수 있습니다. 자세한 내용은 프로젝트에 블루프린트를 추가하여 리소스를 통합하기 및 블루프린트를 프로젝트에서 분리하여 업데이트 중지 섹션을 참조하세요.

2단계: 구성 요소가 포함된 사용자 지정 블루프린트 개발

블루프린트 마법사는 커스텀 블루프린트를 생성할 때 생성되며, 커스텀 블루프린트를 개발할 때 컴포넌트로 수정할 수 있습니다. `src/blueprints.js`및 `src/defaults.json` 파일을 업데이트하여 마법사를 수정할 수 있습니다.

> **Important**
>
> 외부 소스의 블루프린트 패키지를 사용하려는 경우 해당 패키지와 함께 발생할 수 있는 위험을 고려하세요. 공간에 추가하는 커스텀 블루프린트와 이를 통해 생성되는 코드에 대한 책임은 귀하에게 있습니다.

블루프린트 코드를 구성하기 전에 지원되는 통합 개발 환경 (IDE) 을 사용하여 CodeCatalyst 프로젝트에 개발 환경을 만드세요. 필수 도구 및 패키지를 사용하려면 개발 환경이 필요합니다.

개발 환경을 만들려면

1. 탐색 창에서 다음 중 하나를 수행하십시오.

   a. 개요를 선택한 다음 내 개발 환경 섹션으로 이동합니다.

   b. 코드를 선택한 다음 개발 환경을 선택합니다.

   c. 코드를 선택하고 소스 리포지토리를 선택한 다음 블루프린트를 만들 때 만든 리포지토리를 선택합니다.

2. 개발 환경 생성을 선택합니다.
3. 드롭다운 메뉴에서 지원되는 IDE를 선택합니다. 자세한 내용은 개발 환경에 지원되는 통합 개발 환경을 참조하십시오.
4. 기존 브랜치에서 작업을 선택하고 기존 브랜치 드롭다운 메뉴에서 생성한 기능 브랜치를 선택합니다.
5. (선택 사항) 별칭 - 선택적 텍스트 입력 필드에 개발 환경을 식별하는 별칭을 입력합니다.
6. 생성을 선택합니다. 개발 환경을 만드는 동안 개발 환경 상태 열에는 시작 중으로 표시되고, 상태 열에는 개발 환경이 생성되었을 때 실행 중으로 표시됩니다.

자세한 정보는 의 개발 환경을 사용하여 코드 작성 및 수정 CodeCatalyst을 참조하세요.

커스텀 블루프린트를 개발하려면

1. 작동하는 터미널에서 다음 `yarn` 명령어를 사용하여 종속성을 설치합니다.

   ```
   yarn
   ```

   필요한 도구와 패키지는 Yarn을 포함한 CodeCatalyst 개발 환경을 통해 사용할 수 있습니다. 개발자 환경 없이 커스텀 블루프린트를 작업하는 경우 먼저 Yarn을 시스템에 설치하세요. 자세한 내용은 Yarn의 설치 설명서를 참조하십시오.
2. 사용자 지정 블루프린트를 개발하여 원하는 대로 구성하세요. 컴포넌트를 추가하여 블루프린트 마법사를 수정할 수 있습니다. 자세한 내용은 프로젝트 요구 사항을 충족하는 맞춤형 청사진 개발, 프론트엔드 마법사로 블루프린트 기능 수정, 스페이스에 사용자 지정 청사진 게시 단원을 참조하세요.

3단계: 커스텀 블루프린트 미리 보기

커스텀 블루프린트를 설정하고 개발한 후, 블루프린트의 프리뷰 버전을 미리 보고 스페이스에 퍼블리시할 수 있습니다. 프리뷰 버전을 사용하면 새 프로젝트를 만드는 데 사용하거나 기존 프로젝트에 적용하기 전에 블루프린트가 원하는 것인지 확인할 수 있습니다.

커스텀 블루프린트를 미리 보려면

1. 작동 중인 터미널에서 다음 `yarn` 명령어를 사용하세요.

   ```
   yarn blueprint:preview
   ```

2. 제공된 `See this blueprint at:` 링크로 이동하여 커스텀 블루프린트를 미리 보세요.
3. 구성에 따라 텍스트를 포함한 UI가 예상대로 나타나는지 확인하세요. 커스텀 블루프린트를 변경하려면 `blueprint.ts` 파일을 편집하고 블루프린트를 재합성한 다음 프리뷰 버전을 다시 퍼블리시하면 됩니다. 자세한 정보는 재합성을 참조하세요.

(선택 사항) 4단계: 커스텀 블루프린트 프리뷰 버전 퍼블리시

사용자 지정 청사진을 스페이스의 청사진 카탈로그에 추가하려는 경우 사용자 지정 청사진의 미리보기 버전을 스페이스에 게시할 수 있습니다. 이렇게 하면 미리보기가 아닌 버전을 카탈로그에 추가하기 전에 사용자로 블루프린트를 볼 수 있습니다. 프리뷰 버전을 사용하면 실제 버전을 사용하지 않고도 퍼블리싱할 수 있습니다. 예를 들어, 버전을 작업하는 경우 미리 보기 `0.0.1` 버전을 게시하고 추가하여 두 번째 버전에 대한 새 업데이트를 게시하고 추가할 수 있습니다`0.0.2`.

커스텀 블루프린트의 프리뷰 버전을 퍼블리싱하려면

제공된 `Enable version` *`[version number]`* `at:` 링크로 이동하여 커스텀 블루프린트를 활성화하세요. 이 링크는 에서 `yarn` 3단계: 커스텀 블루프린트 미리 보기 명령을 실행할 때 제공됩니다.

사용자 지정 청사진을 만들고, 개발하고, 미리 보고, 게시한 후 최종 청사진 버전을 게시하고 스페이스의 청사진 카탈로그에 추가할 수 있습니다. 자세한 내용은 스페이스에 사용자 지정 청사진 게시 및 우주 청사진 카탈로그에 사용자 지정 청사진 추가 단원을 참조하세요.

튜토리얼: React 애플리케이션 생성 및 업데이트

블루프린트 작성자는 커스텀 블루프린트를 개발하여 스페이스의 블루프린트 카탈로그에 추가할 수 있습니다. 그러면 스페이스 구성원이 이 청사진을 사용하여 새 프로젝트를 만들거나 기존 프로젝트에 추가할 수 있습니다. 블루프린트를 계속 변경한 다음 풀 리퀘스트를 통해 업데이트로 사용할 수 있습니다.

이 튜토리얼에서는 블루프린트 작성자의 관점과 블루프린트 사용자의 관점에서의 안내를 제공합니다. 이 튜토리얼은 React 단일 페이지 웹 애플리케이션 블루프린트를 만드는 방법을 보여줍니다. 그런 다음 블루프린트를 사용하여 새 프로젝트를 만듭니다. 블루프린트가 변경사항으로 업데이트되면, 블루프린트에서 생성된 프로젝트는 풀 리퀘스트를 통해 해당 변경사항을 통합합니다.

주제

- 사전 조건
- 1단계: 사용자 지정 청사진 만들기
- 2단계: 출시 워크플로 보기

- 3단계: 카탈로그에 청사진 추가
- 4단계: 블루프린트로 프로젝트 만들기
- 5단계: 블루프린트 업데이트
- 6단계: 블루프린트에 게시된 카탈로그 버전을 새 버전으로 업데이트
- 7단계: 새 블루프린트 버전으로 프로젝트 업데이트
- 8단계: 프로젝트의 변경 내용 보기

사전 조건

커스텀 블루프린트를 생성하고 업데이트하려면 다음과 같은 작업을 완료해야 합니다. 설정 및 로그인 CodeCatalyst

- 로그인할 AWS CodeCatalyst 빌더 ID가 있어야 합니다.
- 스페이스에 속하고 해당 스페이스에서 스페이스 관리자 또는 파워 사용자 역할을 할당받아야 합니다. 자세한 내용은 스페이스 만들기, 사용자에게 공간 권한 부여, 스페이스 관리자 역할 단원을 참조하세요.

1단계: 사용자 지정 청사진 만들기

커스텀 블루프린트를 만들면 블루프린트 소스 코드, 개발 도구 및 리소스가 포함된 CodeCatalyst 프로젝트가 생성됩니다. 프로젝트는 블루프린트를 개발, 테스트 및 게시하는 곳입니다.

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 블루프린트를 만들려는 스페이스로 이동합니다.
3. 설정을 선택하여 스페이스 설정으로 이동합니다.
4. 스페이스 설정 탭에서 블루프린트를 선택하고 블루프린트 생성을 선택합니다.
5. 블루프린트 생성 마법사의 필드를 다음 값으로 업데이트합니다.
   - 블루프린트 이름에 를 입력합니다. `react-app-blueprint`
   - 블루프린트 디스플레이 이름에 를 입력합니다. `react-app-blueprint`
6. 선택적으로 View code 를 선택하여 블루프린트의 블루프린트 소스 코드를 미리 볼 수 있습니다. 마찬가지로 View workflow (워크플로 보기) 를 선택하여 블루프린트를 빌드하고 퍼블리시하는 프로젝트에서 생성될 워크플로를 미리 볼 수 있습니다.
7. 블루프린트 생성을 선택합니다.

8. 블루프린트가 생성되면 해당 블루프린트 프로젝트로 이동합니다. 이 프로젝트에는 블루프린트 소스 코드와 함께 블루프린트를 개발, 테스트, 퍼블리싱하는 데 필요한 도구 및 리소스가 포함되어 있습니다. 릴리스 워크플로가 생성되었고 블루프린트가 스페이스에 자동으로 게시되었습니다.
9. 블루프린트와 블루프린트 프로젝트가 생성되었으니, 다음 단계는 소스 코드를 업데이트하여 구성하는 것입니다. Dev Environments를 사용하여 브라우저에서 직접 소스 리포지토리를 열고 편집할 수 있습니다.

   탐색 창에서 코드를 선택한 다음 개발 환경을 선택합니다.
10. [개발 환경 만들기] 를 선택한 다음 AWS Cloud9 (브라우저에서) 를 선택합니다.
11. 기본 설정을 유지하고 [Create] 를 선택합니다.
12. AWS Cloud9 터미널에서 다음 명령어를 실행하여 블루프린트 프로젝트 디렉토리로 이동합니다.

```
cd react-app-blueprint
```

13. 블루프린트가 생성되면 `static-assets` 폴더가 자동으로 생성되고 채워집니다. 이 튜토리얼에서는 기본 폴더를 삭제하고 반응 앱 블루프린트를 위한 새 폴더를 생성합니다.

   다음 명령어를 실행하여 static-assets 폴더를 삭제합니다.

```
rm -r static-assets
```

   AWS Cloud9 Linux 기반 플랫폼을 기반으로 구축되었습니다. Windows 운영 체제를 사용하는 경우 다음 명령을 대신 사용할 수 있습니다.

```
rmdir /s /q static-assets
```

14. 이제 기본 폴더가 삭제되었으니 다음 명령어를 실행하여 react-app 블루프린트용 `static-assets` 폴더를 생성하세요.

```
npx create-react-app static-assets
```

   메시지가 표시되면 를 입력하여 진행하세요. y

   필요한 패키지가 들어 있는 `static-assets` 폴더에 새 반응 애플리케이션이 생성되었습니다. 변경사항을 원격 CodeCatalyst 소스 리포지토리로 푸시해야 합니다.
15. 최신 변경 사항이 적용되었는지 확인한 다음 다음 명령을 실행하여 변경 사항을 블루프린트의 CodeCatalyst 소스 리포지토리에 커밋하고 푸시하세요.

```
git pull
```

```
git add .
```

```
git commit -m "Add React app to static-assets"
```

```
git push
```

변경 사항이 블루프린트의 소스 리포지토리에 푸시되면 릴리스 워크플로가 자동으로 시작됩니다. 이 워크플로는 블루프린트 버전을 늘리고, 블루프린트를 빌드하고, 스페이스에 게시합니다. 다음 단계에서는 릴리스 워크플로 실행으로 이동하여 진행 상황을 살펴보겠습니다.

2단계: 출시 워크플로 보기

1. CodeCatalyst 콘솔의 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 블루프린트 출시 워크플로를 선택합니다.
3. 워크플로에 블루프린트를 빌드하고 게시하기 위한 작업이 있는 것을 볼 수 있습니다.
4. 최신 실행에서 워크플로 실행 링크를 선택하면 변경한 코드에 따른 실행을 볼 수 있습니다.
5. 실행이 완료되면 새 블루프린트 버전이 게시됩니다. 게시된 블루프린트 버전은 스페이스 설정에서 볼 수 있지만, 스페이스의 블루프린트 카탈로그에 추가되기 전까지는 프로젝트에 사용할 수 없습니다. 다음 단계에서는 카탈로그에 청사진을 추가해 보겠습니다.

3단계: 카탈로그에 청사진 추가

스페이스의 청사진 카탈로그에 청사진을 추가하면 해당 청사진을 스페이스의 모든 프로젝트에서 사용할 수 있습니다. 스페이스 구성원은 청사진을 사용하여 새 프로젝트를 만들거나 기존 프로젝트에 추가할 수 있습니다.

1. CodeCatalyst 콘솔에서 스페이스로 다시 이동합니다.
2. 설정을 선택한 다음 블루프린트를 선택합니다.
3. 선택한 react-app-blueprint다음 카탈로그에 추가를 선택합니다.
4. 저장을 선택합니다.

4단계: 블루프린트로 프로젝트 만들기

이제 블루프린트가 카탈로그에 추가되었으니 프로젝트에서 사용할 수 있습니다. 이 단계에서는 방금 만든 청사진을 사용하여 프로젝트를 생성합니다. 이후 단계에서는 청사진의 새 버전을 업데이트하고 게시하여 이 프로젝트를 업데이트합니다.

1. 프로젝트 탭을 선택한 다음 프로젝트 생성을 선택합니다.
2. 스페이스 블루프린트를 선택한 다음 선택하세요 react-app-blueprint.

   > **Note**
   >
   > 블루프린트가 선택되면 블루프린트 파일의 내용을 볼 수 있습니다. `README.md`

3. 다음을 선택합니다.
4.

   > **Note**
   >
   > 이 프로젝트 생성 마법사의 내용은 블루프린트에서 구성할 수 있습니다.

   블루프린트 사용자로 프로젝트 이름을 입력합니다. 이 자습서에서는 `react-app-project`을 입력합니다. 자세한 정보는 프로젝트 요구 사항을 충족하는 맞춤형 청사진 개발을 참조하세요.

다음으로 블루프린트를 업데이트하고 이 프로젝트를 업데이트하는 데 사용할 새 버전을 카탈로그에 추가합니다.

5단계: 블루프린트 업데이트

블루프린트를 사용하여 새 프로젝트를 만들거나 기존 프로젝트에 적용한 후에도 블루프린트 작성자로서 계속 업데이트할 수 있습니다. 이 단계에서는 블루프린트를 변경하고 새 버전을 스페이스에 자동으로 퍼블리시합니다. 그런 다음 새 버전을 카탈로그 버전으로 추가할 수 있습니다.

1. 에서 만든 react-app-blueprint프로젝트로 이동합니다튜토리얼: React 애플리케이션 생성 및 업데이트.
2. 에서 튜토리얼: React 애플리케이션 생성 및 업데이트 만든 개발 환경을 엽니다.

   a. 탐색 창에서 코드를 선택한 다음 개발 환경을 선택합니다.

   b. 표에서 개발 환경을 찾은 다음 [브라우저에서 AWS Cloud9 열기] 를 선택합니다.

3. 블루프린트 릴리스 워크플로가 실행되었을 때 파일을 업데이트하여 블루프린트 버전을 늘렸습니다. package.json 터미널에서 다음 명령을 실행하여 변경 내용을 가져오세요. AWS Cloud9

```
git pull
```

4. 다음 명령을 실행하여 static-assets 폴더로 이동합니다.

```
cd /projects/react-app-blueprint/static-assets
```

5. 다음 명령을 실행하여 static-assets 폴더에 hello-world.txt 파일을 생성합니다.

```
touch hello-world.txt
```

AWS Cloud9 Linux 기반 플랫폼을 기반으로 구축되었습니다. Windows 운영 체제를 사용하는 경우 다음 명령을 대신 사용할 수 있습니다.

```
echo > hello-world.text
```

6. 왼쪽 탐색에서 hello-world.txt 파일을 두 번 클릭하여 편집기에서 열고 다음 내용을 추가합니다.

```
Hello, world!
```

파일을 저장합니다.

7. 최신 변경사항이 있는지 확인한 다음 다음 명령어를 실행하여 변경사항을 블루프린트의 CodeCatalyst 소스 리포지토리에 커밋하고 푸시하세요.

```
git pull
```

```
git add .
```

```
git commit -m "prettier setup"
```

```
git push
```

변경 사항을 푸시하면서 릴리스 워크플로가 시작되었고, 새 버전의 블루프린트가 스페이스에 자동으로 게시됩니다.

6단계: 블루프린트에 게시된 카탈로그 버전을 새 버전으로 업데이트

청사진을 사용하여 새 프로젝트를 만들거나 기존 프로젝트에 적용한 후에도 청사진 작성자로서 청사진을 업데이트할 수 있습니다. 이 단계에서는 블루프린트를 변경하고 블루프린트의 카탈로그 버전을 변경합니다.

1. CodeCatalyst 콘솔에서 스페이스로 다시 이동합니다.
2. 설정을 선택한 다음 블루프린트를 선택합니다.
3. 선택한 react-app-blueprint다음 카탈로그 버전 관리를 선택합니다.
4. 새 버전을 선택한 다음 [Save] 를 선택합니다.

7단계: 새 블루프린트 버전으로 프로젝트 업데이트

이제 스페이스의 블루프린트 카탈로그에서 새 버전을 사용할 수 있습니다. 블루프린트 사용자는 에서 만든 프로젝트의 버전을 업데이트할 수 있습니다. 4단계: 블루프린트로 프로젝트 만들기 이렇게 하면 모범 사례를 충족하는 데 필요한 최신 변경 사항 및 수정 사항을 적용할 수 있습니다.

1. CodeCatalyst 콘솔에서 에서 만든 react-app-project프로젝트로 이동합니다4단계: 블루프린트로 프로젝트 만들기.
2. 탐색 창에서 [블루프린트(Blueprints)]를 선택합니다.
3. 정보 상자에서 블루프린트 업데이트를 선택합니다.
4. 오른쪽 코드 변경 패널에서 `hello-world.txt` 및 `package.json` 업데이트를 확인할 수 있습니다.
5. 업데이트 적용을 선택합니다.

업데이트 적용을 선택하면 업데이트된 블루프린트 버전의 변경 사항이 포함된 풀 리퀘스트가 프로젝트에 생성됩니다. 프로젝트를 업데이트하려면 풀 요청을 병합해야 합니다. 자세한 내용은 풀 리퀘스트 검토 및 풀 리퀘스트 병합 섹션을 참조하세요.

1. 블루프린트 테이블에서 블루프린트를 찾으세요. 상태 열에서 보류 중인 풀 리퀘스트를 선택한 다음, 오픈 풀 리퀘스트 링크를 선택합니다.
2. 풀 리퀘스트를 검토한 다음 병합을 선택합니다.
3. [빠른 전달 병합] 을 선택하여 기본값을 유지한 다음 [병합] 을 선택합니다.

8단계: 프로젝트의 변경 내용 보기

청사진에 대한 변경 사항은 이제 이후에 7단계: 새 블루프린트 버전으로 프로젝트 업데이트 프로젝트에서 사용할 수 있습니다. 블루프린트 사용자는 소스 리포지토리에서 변경사항을 볼 수 있습니다.

1. 탐색 창에서 소스 리포지토리를 선택한 다음 프로젝트 생성 시 만든 소스 리포지토리의 이름을 선택합니다.
2. 파일에서 생성된 파일을 볼 수 있습니다. `hello-world.txt` 5단계: 블루프린트 업데이트
3. 파일 내용을 `hello-world.txt` 보려면 를 선택합니다.

라이프사이클 관리를 통해 블루프린트 작성자는 특정 블루프린트가 포함된 모든 프로젝트의 소프트웨어 개발 라이프사이클을 중앙에서 관리할 수 있습니다. 이 튜토리얼에서 볼 수 있듯이, 블루프린트에 업데이트를 푸시한 다음, 블루프린트를 사용하여 새 프로젝트를 만들거나 기존 프로젝트에 적용한 프로젝트에 통합할 수 있습니다. 자세한 내용은 블루프린트 작성자로서 라이프사이클 관리와 협력하기을(를) 참조하세요.

소스 리포지토리를 커스텀 블루프린트로 전환

사용자 지정 청사진을 사용하면 한 공간에 있는 여러 프로젝트의 모범 사례 또는 새로운 옵션을 통합할 수 있습니다. CodeCatalyst 새로운 커스텀 블루프린트를 처음부터 만들고 개발할 수 있지만, 코드와 확립된 모범 사례가 있는 기존 CodeCatalyst 또는 타사 소스 리포지토리를 블루프린트 프로젝트로 변환할 수도 있습니다. 기존 리포지토리의 관련 아티팩트를 블루프린트 프로젝트로 복사하지 않아도 됩니다. 소스 리포지토리를 커스텀 블루프린트로 변환한 후, 다른 커스텀 블루프린트와 마찬가지로 블루프린트를 업데이트, 퍼블리싱, 추가할 수 있습니다.

소스 리포지토리를 커스텀 블루프린트로 변환한 후, 소스 리포지토리는 블루프린트 프로젝트로 재구성되고, 리포지토리의 콘텐츠는 리포지토리 내 `static-assets` 폴더로 이동되며, 블루프린트에 필요한 관련 자산이 리포지토리에 추가됩니다. 커스텀 블루프린트를 사용하여 프로젝트를 생성하거나 기존 프로젝트에 추가하면 변환된 소스 리포지토리에 저장된 워크플로 정의도 블루프린트에 의해 프로젝트에 추가됩니다.

> **Note**
>
> 소스 리포지토리를 커스텀 블루프린트로 변환할 때 환경 및 비밀과 같은 리소스는 포함되지 않습니다. 소스 리포지토리를 커스텀 블루프린트로 변환한 후에는 해당 리소스를 수동으로 복사하거나 추가해야 합니다.

> **Important**
>
> 소스 리포지토리를 사용자 지정 블루프린트로 변환하려면 스페이스의 프로젝트 관리자, 스페이스 관리자 또는 파워 사용자 역할이 있는 계정으로 로그인해야 합니다.

소스 리포지토리 목록에서 소스 리포지토리를 사용자 지정 블루프린트로 변환하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 스페이스로 이동한 다음 소스 리포지토리를 사용자 지정 블루프린트로 변환하려는 프로젝트를 선택합니다.
3. 탐색 창에서 코드를 선택하고 소스 리포지토리를 선택한 다음 커스텀 블루프린트로 변환하려는 소스 리포지토리의 라디오 버튼을 선택합니다.
4. 소스 리포지토리를 커스텀 블루프린트로 변환하려면 [블루프린트로 변환] 을 선택합니다.

콘솔의 소스 CodeCatalyst 리포지토리 페이지에서 리포지토리를 커스텀 블루프린트로 변환할 수도 있습니다. CodeCatalyst

소스 리포지토리 페이지에서 소스 리포지토리를 사용자 지정 블루프린트로 변환하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 스페이스로 이동한 다음 소스 리포지토리를 사용자 지정 블루프린트로 변환하려는 프로젝트를 선택합니다.
3. 탐색 창에서 코드를 선택하고 소스 리포지토리를 선택한 다음 커스텀 블루프린트로 변환하려는 CodeCatalyst 소스 리포지토리의 이름을 선택합니다.
4. 추가 드롭다운 메뉴를 선택한 다음 Convert to Blueprint를 선택하여 소스 리포지토리를 사용자 지정 블루프린트로 변환합니다.

소스 리포지토리가 커스텀 블루프린트로 변환되면 릴리스 워크플로가 자동으로 실행됩니다. 실행이 성공적으로 완료되면 사용자 지정 블루프린트가 스페이스의 사용자 지정 블루프린트 목록에 게시됩니다. 여기서 변환된 사용자 정의 청사진을 스페이스 카탈로그에 추가하여 새 프로젝트를 만들거나 기존 프로젝트에 추가할 수 있습니다. 자세한 내용은 스페이스에 사용자 지정 청사진 게시 및 우주 청사진 카탈로그에 사용자 지정 청사진 추가 섹션을 참조하세요.

> **Important**
>
> 스페이스의 청사진 카탈로그에 사용자 정의 청사진을 추가하려면 스페이스의 스페이스 관리자 또는 고급 사용자 역할을 가진 계정으로 로그인해야 합니다.

블루프린트 작성자로서 라이프사이클 관리와 협력하기

수명 주기 관리를 사용하면 공통된 단일 모범 사례 소스에서 많은 프로젝트를 동기화할 수 있습니다. 이를 통해 전체 소프트웨어 개발 수명 주기에 걸쳐 수정 사항 전파와 프로젝트 수의 유지 관리를 확장할 수 있습니다. 라이프사이클 관리는 내부 캠페인, 보안 수정, 감사, 런타임 업그레이드, 모범 사례 변경 및 기타 유지 관리 관행을 간소화합니다. 이러한 표준이 한 곳에서 정의되고 새 표준이 게시되면 중앙에서 자동으로 최신 상태로 유지되기 때문입니다.

청사진의 새 버전이 게시되면 해당 청사진을 포함하는 모든 프로젝트에 최신 버전으로 업데이트하라는 메시지가 표시됩니다. 블루프린트 작성자는 규정 준수를 위해 각 프로젝트에 포함된 특정 블루프린트의 버전도 볼 수 있습니다. 기존 소스 리포지토리에 충돌이 있는 경우 라이프사이클 관리는 pull 요청을 생성합니다. 개발 환경과 같은 다른 모든 리소스의 경우 모든 수명 주기 관리 업데이트는 엄격하게 새 리소스를 생성합니다. 사용자는 이러한 pull 요청을 병합하거나 병합하지 않을 수 있습니다. 보류 중인 풀 리퀘스트가 병합되면 프로젝트에 사용된 옵션을 포함한 블루프린트 버전이 업데이트됩니다. 블루프린트 사용자로서 라이프사이클 관리를 사용하는 방법에 대해 알아보려면 및 을 참조하십시오. 기존 프로젝트에 라이프사이클 관리 사용 프로젝트 내 여러 블루프린트에서 라이프사이클 관리 사용

주제

- 번들 출력 및 병합 충돌에 대한 라이프사이클 관리 테스트
- 병합 전략을 사용하여 번들 생성 및 파일 지정
- 프로젝트 세부 정보를 위한 컨텍스트 객체 액세스

번들 출력 및 병합 충돌에 대한 라이프사이클 관리 테스트

블루프린트의 라이프사이클 관리를 로컬에서 테스트하고 충돌 해결을 병합할 수 있습니다. `synth/`디렉터리 아래에 라이프사이클 업데이트의 다양한 단계를 나타내는 일련의 번들이 생성됩니다. 라이프사이클 관리를 테스트하려면 블루프린트에서 다음 yarn 명령을 실행할 수 있습니다. `yarn blueprint: resynth` 재합성 및 번들에 대한 자세한 내용은 및 을 참조하십시오. 재합성 재합성을 통한 파일 생성

병합 전략을 사용하여 번들 생성 및 파일 지정

주제

재합성을 통한 파일 생성

재합성은 블루프린트에서 생성된 소스 코드를 동일한 블루프린트에서 이전에 생성한 소스 코드와 병합하여 블루프린트에 대한 변경 사항을 기존 프로젝트에 전파할 수 있습니다. 함수에서 블루프린트 출력 번들 전반에서 병합이 `resynth()` 실행됩니다. 재합성은 먼저 블루프린트와 프로젝트 상태의 다양한 측면을 나타내는 세 개의 번들을 생성합니다. `yarn blueprint:resynth`명령을 사용하여 로컬에서 수동으로 실행할 수 있으며, 번들이 아직 없는 경우 번들이 생성됩니다. 번들을 수동으로 사용하여 작업하면 로컬에서 재합성 동작을 모의하고 테스트할 수 있습니다. 기본적으로 블루프린트는 저장소 전체에서만 재합성을 실행합니다. 보통 번들 중 일부만 소스 컨트롤 하에 `src/*` 있기 때문입니다.

- `existing-bundle`- 이 번들은 기존 프로젝트 상태를 나타냅니다. 이는 합성 컴퓨팅을 통해 인위적으로 구성한 것으로, 배포하려는 프로젝트의 내용 (있는 경우) 에 대한 청사진 컨텍스트를 제공합니다. 로컬에서 재합성을 실행할 때 이 위치에 무언가가 이미 존재한다면, 재설정되고 모의 객체로 간주될 것입니다. 그렇지 않으면 의 내용으로 설정됩니다. `ancestor-bundle`
- `ancestor-bundle`- 이전 옵션 및/또는 버전과 합성된 경우 블루프린트 출력을 나타내는 번들입니다. 이 블루프린트가 프로젝트에 처음 추가되는 것이라면, 상위 블루프린트는 존재하지 않으므로 블루프린트와 동일한 내용으로 설정됩니다. `existing-bundle` 로컬에서 이 번들이 이미 이 위치에 있으면 모의 제품으로 간주합니다.
- `proposed-bundle`- 이 번들은 블루프린트가 몇 가지 새로운 옵션 및/또는 버전으로 합성된 경우 블루프린트를 모방하는 번들입니다. 이 번들은 함수에서 생성되는 번들과 동일합니다. `synth()` 로컬에서는 이 번들이 항상 오버라이드됩니다.

각 번들은 재합성 단계에서 생성되며, 아래의 블루프린트 클래스에서 접근할 수 있습니다. `this.context.resynthesisPhase`

- `resolved-bundle`- 이것은 최종 번들로, 무엇이 패키징되어 프로젝트에 배포되는지를 표현한 CodeCatalyst 것입니다. 배포 메커니즘으로 전송되는 파일 및 diff를 볼 수 있습니다. 이것은 다른 세 번들 간의 병합을 해결하는 `resynth()` 함수의 출력입니다.

3방향 병합은 ancestor-bundle 와 proposed-bundle 의 차이를 구한 다음 에 더하여 를 생성하는 방식으로 적용됩니다. existing-bundle resolved-bundle 모든 병합 전략은 resolved-bundle 파일을 로 해결합니다. 재합성은 도중에 resynth() 블루프린트의 병합 전략을 통해 이러한 번들 범위를 해결하고 그 결과로부터 해결된 번들을 생성합니다.

병합 전략 사용

블루프린트 라이브러리에서 제공하는 병합 전략을 사용할 수 있습니다. 이 전략은 섹션에 언급된 파일의 파일 출력 및 충돌을 해결하는 방법을 제공합니다. 재합성을 통한 파일 생성

- alwaysUpdate- 제안된 파일을 항상 반영하는 전략.
- neverUpdate- 항상 기존 파일로 해석되는 전략.
- onlyAdd- 기존 파일이 아직 존재하지 않는 경우 제안된 파일로 변환하는 전략입니다. 그렇지 않으면 기존 파일로 해결됩니다.
- threeWayMerge- 기존, 제안된 파일 및 공통 상위 파일 간에 3방향 병합을 수행하는 전략입니다. 파일을 완전히 병합할 수 없는 경우 해결된 파일에 충돌 표시가 포함될 수 있습니다. 전략이 의미 있는 결과를 산출하려면 제공된 파일의 내용을 UTF-8 인코딩해야 합니다. 이 전략은 입력 파일이 바이너리인지 여부를 감지하려고 시도합니다. 전략이 바이너리 파일에서 병합 충돌을 감지하면 항상 제안된 파일을 반환합니다.
- preferProposed- 기존, 제안된 파일 및 공통 상위 파일 간에 3방향 병합을 수행하는 전략입니다. 이 전략은 각 충돌에서 제안된 파일의 측면을 선택하여 충돌을 해결합니다.
- preferExisting- 기존, 제안된 파일 및 공통 상위 파일 간에 3방향 병합을 수행하는 전략입니다. 이 전략은 각 충돌에서 기존 파일의 측면을 선택하여 충돌을 해결합니다.

병합 전략의 소스 코드를 보려면 오픈 소스 GitHub 리포지토리를 참조하십시오.

라이프사이클 관리 업데이트를 위한 파일 지정

재합성 과정에서 블루프린트는 변경 사항이 기존 소스 저장소에 병합되는 방식을 제어합니다. 하지만 블루프린트의 모든 파일에 업데이트를 푸시하고 싶지는 않을 수도 있습니다. 예를 들어 CSS 스타일시트와 같은 샘플 코드는 프로젝트 전용입니다. 다른 전략을 지정하지 않는 경우 3방향 병합 전략이 기본 옵션입니다. 블루프린트는 리포지토리 구조 자체에 병합 전략을 지정하여 소유한 파일과 소유하지 않은 파일을 지정할 수 있습니다. 블루프린트는 병합 전략을 업데이트할 수 있고, 재합성 중에 최신 전략을 사용할 수 있습니다.

```
const sourceRepo = new SourceRepository(this, {
      title: 'my-repo',
    });
```

```
    sourceRepo.setResynthStrategies([
      {
        identifier: 'dont-override-sample-code',
        description: 'This strategy is applied accross all sample code. The blueprint
 will create sample code, but skip attempting to update it.',
        strategy: MergeStrategies.neverUpdate,
        globs: [
          '**/src/**',
          '**/css/**',
        ],
      },
    ]);
```

병합 전략을 여러 개 지정할 수 있으며 마지막 전략이 우선합니다. 발견되지 않은 파일은 기본적으로 three-way-merge Git과 비슷합니다. `MergeStrategies`구문을 통해 제공되는 여러 병합 전략이 있지만 직접 작성할 수도 있습니다. 제공된 전략은 git merge 전략 드라이버를 준수합니다.

병합 전략 작성

제공된 빌드 병합 전략 중 하나를 사용하는 것 외에도 자신만의 전략을 작성할 수도 있습니다. 전략은 표준 전략 인터페이스를 준수해야 합니다. `existing-bundle`,`proposed-bundle`, 에서 파일 버전을 가져와서 하나의 확인된 파일로 병합하는 전략 함수를 작성해야 합니다. `ancestor-bundle` 예:

```
type StrategyFunction = (
   /**
   * file from the ancestor bundle (if it exists)
   */
    commonAncestorFile: ContextFile | undefined,
   /**
   * file from the existing bundle (if it exists)
   */
    existingFile: ContextFile | undefined,
   /**
   * file from the proposed bundle (if it exists)
   */
    proposedFile: ContextFile | undefined,
    options?: {})
    /**
    * Return: file you'd like in the resolved bundle
    * passing undefined will delete the file from the resolved bundle
    */
=> ContextFile | undefined;
```

파일이 존재하지 않는 경우 (정의되지 않은 경우) 해당 파일 경로는 특정 위치 번들에 존재하지 않는 것입니다.

예:

```
strategies: [
          {
            identifier: 'dont-override-sample-code',
            description: 'This strategy is applied across all sample code. The
 blueprint will create sample code, but skip attempting to update it.',
            strategy: (ancestor, existing, proposed) => {
                const resolvedfile = ...
                ...
                // do something
                ...
                return resolvedfile
            },
            globs: [
              '**/src/**',
              '**/css/**',
            ],
          },
        ],
```

프로젝트 세부 정보를 위한 컨텍스트 객체 액세스

블루프린트 작성자는 합성 중에 블루프린트 프로젝트의 컨텍스트에 액세스하여 프로젝트 소스 저장소의 공간, 프로젝트 이름 또는 기존 파일과 같은 정보를 얻을 수 있습니다. 블루프린트가 생성하는 재합성 단계와 같은 세부 정보도 얻을 수 있습니다. 예를 들어 컨텍스트에 액세스하여 조상 번들 또는 제안된 번들을 생성하기 위해 재합성하고 있는지 알 수 있습니다. 그러면 기존 코드 컨텍스트를 사용하여 저장소의 코드를 변환할 수 있습니다. 예를 들어, 고유한 재합성 전략을 작성하여 특정 코드 표준을 설정할 수 있습니다. 작은 청사진을 위해 전략을 `blueprint.ts` 파일에 추가하거나 전략을 위한 별도의 파일을 만들 수 있습니다.

다음 예제는 프로젝트 컨텍스트에서 파일을 찾고, 워크플로 빌더를 설정하고, 특정 파일에 대해 블루프린트가 제공하는 재합성 전략을 설정하는 방법을 보여줍니다.

```
const contextFiles = this.context.project.src.findAll({
      fileGlobs: ['**/package.json'],
    });
```

```
    // const workflows = this.context.project.src.findAll({
    //   fileGlobs: ['**/.codecatalyst/**.yaml'],
    // });

    const security = new WorkflowBuilder(this, {
      Name: 'security-workflow',
    });
    new Workflow(this, repo, security.getDefinition());
    repo.setResynthStrategies([
      {
        identifier: 'force-security',
        globs: ['**/.codecatalyst/security-workflow.yaml'],
        strategy: MergeStrategies.alwaysUpdate,
      },
    ]);


    for (const contextFile of contextFiles) {
      const packageObject = JSON.parse(contextFile.buffer.toString());
      new SourceFile(internalRepo, contextFile.path, JSON.stringify({
        ...packageObject,
      }, null, 2));
    }
  }
```

프로젝트 요구 사항을 충족하는 맞춤형 청사진 개발

사용자 지정 청사진을 게시하기 전에 특정 요구 사항을 충족하는 청사진을 개발할 수 있습니다. 미리 볼 때 프로젝트를 만들어 사용자 지정 청사진을 개발하고 청사진을 테스트할 수 있습니다. 특정 소스 코드, 계정 연결, 워크플로, 이슈 또는 에서 생성할 수 있는 기타 구성 요소와 같은 프로젝트 구성 요소를 포함하도록 사용자 지정 청사진을 개발할 수 있습니다. CodeCatalyst

> **Important**
>
> 외부 소스의 블루프린트 패키지를 사용하려는 경우 해당 패키지와 함께 발생할 수 있는 위험을 고려하세요. 공간에 추가하는 커스텀 블루프린트와 이를 통해 생성되는 코드에 대한 책임은 귀하에게 있습니다.

> **Important**
>
> 스페이스에서 사용자 지정 청사진을 개발하려면 스페이스에서 CodeCatalyst 스페이스 관리자 또는 고급 사용자 역할을 가진 계정으로 로그인해야 합니다.

사용자 지정 청사진을 개발하거나 업데이트하려면

1. 개발 환경을 재개하세요. 자세한 정보는 Dev Environment 재개을 참조하세요.

   개발 환경이 없는 경우 먼저 개발 환경을 만들어야 합니다. 자세한 정보는 Dev Environment 생성을 참조하세요.
2. 개발 환경에서 작동하는 터미널을 여십시오.
3. 블루프린트를 만들 때 릴리스 워크플로를 선택한 경우 최신 블루프린트 버전이 자동으로 게시됩니다. 변경사항을 가져와서 `package.json` 파일에 증분된 버전이 있는지 확인하세요. 다음 명령을 사용합니다.

   ```
   git pull
   ```
4. `src/blueprint.ts`파일에서 커스텀 블루프린트의 옵션을 편집하세요. CodeCatalyst 마법사는 `Options` 인터페이스를 동적으로 해석하여 선택 사용자 인터페이스 (UI) 를 생성합니다. 구성 요소와 지원되는 태그를 추가하여 사용자 지정 블루프린트를 개발할 수 있습니다. 자세한 내용은 프론트엔드 마법사로 블루프린트 기능 수정, 블루프린트에 환경 구성 요소 추가, 블루프린트에 영역 컴포넌트 추가, 블루프린트에 리포지토리 및 소스 코드 구성 요소 추가, 블루프린트에 워크플로우 구성 요소 추가 및 블루프린트에 개발 환경 구성 요소 추가 단원을 참조하세요.

   커스텀 블루프린트를 개발할 때 블루프린트 SDK와 샘플 블루프린트를 보고 추가 지원을 받을 수도 있습니다. 자세한 내용은 오픈 소스 리포지토리를 참조하십시오. GitHub

사용자 지정 블루프린트는 성공적인 합성의 결과로 프리뷰 번들을 제공합니다. 프로젝트 번들은 프로젝트의 소스 코드, 구성, 리소스를 나타내며, CodeCatalyst 배포 API 작업에서 프로젝트에 배포하는 데 사용됩니다. 커스텀 블루프린트를 계속 개발하려면 블루프린트 합성 프로세스를 다시 실행하세요. 자세한 내용은 커스텀 블루프린트 컨셉을(를) 참조하세요.

프론트엔드 마법사로 블루프린트 기능 수정

의 블루프린트 선택 마법사는 파일의 `Options` 인터페이스에 의해 자동 생성됩니다. CodeCatalyst `blueprint.ts` 프론트엔드 마법사는 JSDOC 스타일 주석과 태그를 `Options` 사용하여 블루프린트

의 수정 및 기능을 지원합니다. JSDOC 스타일 주석과 태그를 사용하여 작업을 수행할 수 있습니다. 예를 들어, 옵션 위에 표시된 텍스트를 선택하거나, 입력 유효성 검사와 같은 기능을 활성화하거나, 옵션을 축소 가능하게 만들 수 있습니다. 마법사는 해당 TypeScript 유형에서 생성된 AST (추상 구문 트리)를 인터페이스에서 해석하는 방식으로 작동합니다. `Options` 마법사는 가능한 한 가장 잘 설명된 유형으로 자동 구성합니다. 모든 유형이 지원되는 것은 아닙니다. 지원되는 다른 유형으로는 지역 선택기 및 환경 선택기가 있습니다.

다음은 블루프린트와 함께 JSDOC 주석과 태그를 사용하는 마법사의 예시입니다. `Options`

```
export interface Options {
  /**
   * What do you want to call your new blueprint?
   * @validationRegex /^[a-zA-Z0-9_]+$/
   * @validationMessage Must contain only upper and lowercase letters, numbers and
 underscores
   */
  blueprintName: string;

  /**
   * Add a description for your new blueprint.
   */
   description?: string;

   /**
    * Tags for your Blueprint:
    * @collapsed true
    */
  tags?: string[];
}
```

`Options`인터페이스의 각 옵션 표시 이름이 기본적으로 에 나타납니다. `camelCase` JSDOC 스타일 주석의 일반 텍스트는 마법사의 옵션 위에 텍스트로 표시됩니다.

주제

- 지원되는 태그
- 지원되는 TypeScript 유형
- 합성 중 사용자와 커뮤니케이션

지원되는 태그

프론트엔드 마법사의 Options 커스텀 블루프린트에서 지원되는 JSDOC 태그는 다음과 같습니다.

@inlinePolicy. /경로/to/정책/파일.json

- 필수 - 옵션은 유형이어야 합니다. Role
- 사용법 - 역할에 필요한 인라인 정책을 전달할 수 있습니다. policy.json경로는 소스 코드 아래에 있을 것으로 예상됩니다. 역할에 대한 사용자 지정 정책이 필요한 경우 이 태그를 사용하세요.
- 종속성 - blueprint-cli 0.1.12 그 이상
- 예시 - @inlinePolicy ./deployment-policy.json

```
environment: EnvironmentDefinition{
    awsAccountConnection: AccountConnection{
      /**
       * @inlinePolicy ./path/to/deployment-policy.json
       */
      cdkRole: Role[];
    };
   };
```

@trustPolicy. /경로/to/정책/파일.json

- 필수 - 옵션은 유형이어야 합니다. Role
- 사용법 - 역할에 필요한 신뢰 정책을 전달할 수 있습니다. policy.json경로는 소스 코드 아래에 있을 것으로 예상됩니다. 역할에 대한 사용자 지정 정책이 필요한 경우 이 태그를 사용하세요.
- 종속성 - blueprint-cli 0.1.12 그 이상
- 예시 - @trustPolicy ./trust-policy.json

```
environment: EnvironmentDefinition{
    awsAccountConnection: AccountConnection{
      /**
       * @trustPolicy ./path/to/trust-policy.json
       */
      cdkRole: Role[];
    };
   };
```

@validationRegex 정규식 표현식

- 필수 - 문자열이어야 하는 옵션.
- 사용법 - 지정된 정규식 표현식을 사용하여 옵션에 대한 입력 검증을 수행하고 표시합니다. `@validationMessage`
- 예 - `@validationRegex /^[a-zA-Z0-9_]+$/`
- 권장 사항 - 함께 사용하십시오`@validationMessage`. 검증 메시지는 기본적으로 비어 있습니다.

@validationMessage 문자열

- 사용량을 검토하려면 - **`@validationRegex`** 또는 기타 오류가 필요합니다.
- 사용법 - `@validation*` 실패 시 검증 메시지를 표시합니다.
- 예 -`@validationMessage Must contain only upper and lowercase letters, numbers, and underscores.`
- 권장 사항 - 함께 사용하십시오`@validationMessage`. 검증 메시지는 기본적으로 비어 있습니다.

@collapsed 부울 (선택 사항)

- 필요 - N/A
- 사용법 - 하위 옵션을 접을 수 있게 하는 부울. 축소된 주석이 있는 경우 기본값은 true입니다. 값을 로 설정하면 처음에는 열려 있던 접을 수 있는 섹션이 `@collapsed false` 만들어집니다.
- 예 - `@collapsed true`

@displayName 문자열

- 필요 - N/A
- 사용법 - 옵션 표시 이름을 변경합니다. 표시 이름에 CamelCase 이외의 형식을 사용할 수 있습니다.
- 예 - `@displayName Blueprint Name`

@displayName 문자열

- 필요 - N/A
- 사용법 - 옵션 표시 이름을 변경합니다. 표시 이름에 CamelCase 이외의 형식을 사용할 수 있습니다.
- 예 - `@displayName Blueprint Name`

@defaultEntropy 넘버

- 필수 - 문자열일 수 있는 옵션입니다.
- 사용법 - 지정된 길이의 무작위 영숫자 문자열을 옵션에 추가합니다.
- 예 - `@defaultEntropy 5`

@placeholder 문자열 (선택 사항)

- 필요 - N/A
- 사용법 - 기본 텍스트 필드 자리 표시자를 변경합니다.
- 예 - `@placeholder type project name here`

@textArea 번호 (선택 사항)

- 필요 - N/A
- 사용법 - 큰 텍스트 섹션의 경우 문자열 입력을 텍스트 영역 구성 요소로 변환합니다. 숫자를 추가하면 행 수가 정의됩니다. 기본값은 5개 행입니다.
- 예 - `@textArea 10`

@hidden 부울 (선택 사항)

- 필요 - N/A
- 사용법 - 유효성 검사에 실패하지 않는 한 사용자에게 파일을 숨깁니다. 기본값은 true입니다.
- 예 - `@hidden`

@button 부울 (선택 사항)

- 필요 - N/A
- 사용법 - 주석은 Boolean 속성에 있어야 합니다. 선택하면 true로 합성되는 버튼을 추가합니다. 토글이 아닙니다.
- 예시 - `buttonExample: boolean;`

```
/**
  * @button
  */
```

```
buttonExample: boolean;
```

@showName 부울 (선택 사항)

- 필요 - N/A
- 사용 - 계정 연결 유형에서만 사용할 수 있습니다. 숨겨진 이름 입력을 표시합니다. 기본값은 `default_environment`입니다.
- 예 - `@showName true`

```
/**
  * @showName true
  */
accountConnection: AccountConnection<{
    ...
}>;
```

@ showEnvironmentType 부울 (선택 사항)

- 필요 - N/A
- 사용 - 계정 연결 유형에서만 사용할 수 있습니다. 숨겨진 환경 유형 드롭다운 메뉴를 표시합니다. 모든 연결의 기본값은 입니다. `production` 옵션은 비프로덕션 또는 프로덕션입니다.
- 예 - `@showEnvironmentType true`

```
/**
  * @showEnvironmentType true
  */
accountConnection: AccountConnection<{
    ...
}>;
```

@forceDefault 부울 (선택 사항)

- 필요 - N/A
- 사용법 - 사용자가 이전에 사용한 값 대신 청사진 작성자가 제공한 기본값을 사용합니다.
- 예시 - `forceDeafultExample: any;`

```
/**
  * @forceDefault
  */
forceDeafultExample: any;
```

@requires 블루프린트 이름

- 필요 - 인터페이스에 주석을 달아줍니다. `Options`
- 사용법 - 현재 블루프린트의 요구 사항으로 프로젝트에 지정된 `blueprintName` 항목을 추가하라고 사용자에게 경고합니다.
- 예시 - `@requires '@amazon-codecatalyst/blueprints.blueprint-builder'`

```
/*
 * @requires '@amazon-codecatalyst/blueprints.blueprint-builder'
 */
export interface Options extends ParentOptions {
...
```

지원되는 TypeScript 유형

프론트엔드 마법사의 `Options` 커스텀 블루프린트는 다음 TypeScript 유형을 지원합니다.

숫자

- 필수 - 옵션은 유형이어야 합니다. `number`
- 사용법 - 숫자 입력 필드를 생성합니다.
- 예 - `age: number`

```
{
  age: number
  ...
}
```

String

- 필수 - 유형이어야 하는 옵션`string`.

- 사용법 - 문자열 입력 필드를 생성합니다.
- 예 - `name: string`

```
{
  age: string
  ...
}
```

문자열 목록

- 필수 - 유형이어야 하는 옵션boolean.
- 사용법 - 체크박스 생성.
- 예 - `isProduction: boolean`

```
{
  isProduction: boolean
  ...
}
```

라디오

- 필수 - 세 개 이하의 문자열을 조합할 수 있는 옵션입니다.
- 사용법 - 선택한 라디오를 생성합니다.

> **Note**
>
> 항목이 4개 이상인 경우 이 유형은 드롭다운으로 렌더링됩니다.

- 예 - `color: 'red' | 'blue' | 'green'`

```
{
  color: 'red' | 'blue' | 'green'
  ...
}
```

Dropdown

- 필수 - 4개 이상의 문자열을 합친 옵션입니다.
- 사용법 - 드롭다운 생성.
- 예 - `runtimes: 'nodejs' | 'python' | 'java' | 'dotnetcore' | 'ruby'`

```
{
  runtimes: 'nodejs' | 'python' | 'java' | 'dotnetcore' | 'ruby'
  ...
}
```

확장 가능한 섹션

- 필수 - 객체가 되기 위한 옵션입니다.
- 사용법 - 확장 가능한 섹션을 생성합니다. 개체의 옵션은 마법사의 확장 가능한 섹션 내에 중첩됩니다.
- 예 -

```
{
     expandableSectionTitle: {
         nestedString: string;
         nestedNumber: number;
     }
}
```

튜플

- 필수 - 옵션은 `Tuple` 유형이어야 합니다.
- 사용법 - 키-값 유료 입력을 생성합니다.
- 예 - `tuple: Tuple[string, string]>`

```
{
    tuple: Tuple[string, string]>;
    ...
}
```

튜플 리스트

- 필수 - 옵션은 다양한 유형의 Tuple 배열이어야 합니다.
- 사용법 - 튜플 목록 입력을 생성합니다.
- 예 - `tupleList: Tuple[string, string]>[]`

```
{
  tupleList: Tuple[string, string]>[];
  ...
}
```

Selector

- 필수 - 옵션은 유형이어야 Selector 합니다.
- 사용법 - 프로젝트에 적용된 소스 리포지토리 또는 블루프린트의 드롭다운을 생성합니다.
- 예시 - `sourceRepo: Selector<SourceRepository>`

```
{
    sourceRepo: Selector<SourceRepository>;
    sourceRepoOrAdd: Selector<SourceRepository | string>;
    blueprintInstantiation: Selector<BlueprintInstantiation>;
  ...
}
```

다중 선택

- 필수 - 옵션은 Selector 유형이어야 합니다.
- 사용법 - 다중 선택 입력을 생성합니다.
- 예 - `multiselect: MultiSelect['A' | 'B' | 'C' | 'D' | 'E']>`

```
{
  multiselect: MultiSelect['A' | 'B' | 'C' | 'D' | 'E']>;
  ...
}
```

합성 중 사용자와 커뮤니케이션

블루프린트 작성자는 검증 메시지 외에도 사용자와 다시 소통할 수 있습니다. 예를 들어 스페이스 구성원은 명확하지 않은 청사진을 생성하는 옵션 조합을 볼 수 있습니다. 커스텀 블루프린트는 합성을 호출하여 사용자에게 오류 메시지를 다시 전달하는 기능을 지원합니다. 기본 블루프린트는 명확한 오류 메시지가 예상되는 `throwSynthesisError(...)` 함수를 구현합니다. 다음을 사용하여 메시지를 호출할 수 있습니다.

```
//blueprint.ts
this.throwSynthesisError({
   name: BlueprintSynthesisErrorTypes.BlueprintSynthesisError,
   message: 'hello from the blueprint! This is a custom error communicated to the
 user.'
})
```

블루프린트에 환경 구성 요소 추가

커스텀 블루프린트 마법사는 마법사를 통해 표시되는 `Options` 인터페이스에서 동적으로 생성됩니다. 블루프린트는 노출된 유형에서 사용자 인터페이스 (UI) 컴포넌트 생성을 지원합니다.

Amazon CodeCatalyst 블루프린트 환경 구성 요소를 가져오려면

`blueprint.ts`파일에 다음을 추가합니다.

```
import {...} from '@amazon-codecatalyst/codecatalyst-environments'
```

주제

- 개발 환경 만들기
- 모의 인터페이스 예제

개발 환경 만들기

다음 예제는 애플리케이션을 클라우드에 배포하는 방법을 보여줍니다.

```
export interface Options extends ParentOptions {
        ...
        myNewEnvironment:  EnvironmentDefinition{
            thisIsMyFirstAccountConnection: AccountConnection{
                thisIsARole: Role['lambda', 's3', 'dynamo'];
```

```
            };
        };
    }
```

인터페이스는 단일 계정 연결 () 을 사용하여 새 환경 (myNewEnvironment) 을 요청하는 UI 구성 요소를 생성합니다 (thisIsMyFirstAccountConnection. 필요한 최소 역할 기능을 사용하여 계정 연결 (thisIsARole) 상의 ['lambda', 's3', 'dynamo'] 역할도 생성됩니다. 모든 사용자에게 계정 연결이 있는 것은 아니므로 사용자가 계정을 연결하지 않거나 계정과 역할을 연결하지 않는 경우를 확인해야 합니다. 역할에 주석을 달 수도 있습니다. @inlinePolicies 자세한 정보는 @inlinePolicy. /경로/to/정책/파일.json을 참조하세요.

환경 구성 요소에는 AND가 필요합니다. name environmentType 다음 코드는 필요한 최소 기본 세이프입니다.

```
{
  ...
  "myNewEnvironment": {
    "name": "myProductionEnvironment",
    "environmentType": "PRODUCTION"
  },
}
```

그러면 UI 구성요소가 다양한 필드를 입력하라는 메시지를 표시합니다. 필드를 채우면 블루프린트가 완전히 확장된 모양이 됩니다. 테스트 및 개발 목적으로 전체 모의 객체를 defaults.json 파일에 포함시키는 것이 유용할 수 있습니다.

모의 인터페이스 예제

간단한 모의 인터페이스

```
{
    ...
    "thisIsMyEnvironment": {
        "name": "myProductionEnvironment",
        "environmentType": "PRODUCTION",
        "thisIsMySecondAccountConnection": {
            "id": "12345678910",
            "name": "my-account-connection-name",
            "secondAdminRole": {
                "arn": "arn:aws:iam::12345678910:role/ConnectedQuokkaRole",
                "name": "ConnectedQuokkaRole",
```

```
                "capabilities": [
                    "lambda",
                    "s3",
                    "dynamo"
                ]
            }
        }
    }
}
```

복잡한 모의 인터페이스

```
export interface Options extends ParentOptions {
  /**
   * The name of an environment
   * @displayName This is a Environment Name
   * @collapsed
   */
  thisIsMyEnvironment: EnvironmentDefinition{
    /**
     * comments about the account that is being deployed into
     * @displayName This account connection has an overriden name
     * @collapsed
     */
    thisIsMyFirstAccountConnection: AccountConnection{
      /**
       * Blah blah some information about the role that I expect
       * e.g. here's a copy-pastable policy: [to a link]
       * @displayName This role has an overriden name
       */
      adminRole: Role['admin', 'lambda', 's3', 'cloudfront'];
      /**
       * Blah blah some information about the second role that I expect
       * e.g. here's a copy-pastable policy: [to a link]
       */
      lambdaRole: Role['lambda', 's3'];
    };
    /**
     * comments about the account that is being deployed into
     */
    thisIsMySecondAccountConnection: AccountConnection{
      /**
         * Blah blah some information about the role that I expect
```

```
         * e.g. here's a copy-pastable policy: [to a link]
         */
      secondAdminRole: Role['admin', 'lambda', 's3', 'cloudfront'];
      /**
         * Blah blah some information about the second role that I expect
         * e.g. here's a copy-pastable policy: [to a link]
         */
      secondLambdaRole: Role['lambda', 's3'];
    };
  };
}
```

완전한 모의 인터페이스

```
{
  ...
  "thisIsMyEnvironment": {
    "name": "my-production-environment",
    "environmentType": "PRODUCTION",
    "thisIsMySecondAccountConnection": {
      "id": "12345678910",
      "name": "my-connected-account",
      "secondAdminRole": {
        "name": "LambdaQuokkaRole",
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "admin",
          "lambda",
          "s3",
          "cloudfront"
        ]
      },
      "secondLambdaRole": {
        "name": "LambdaQuokkaRole",
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "lambda",
          "s3"
        ]
      }
    },
    "thisIsMyFirstAccountConnection": {
      "id": "12345678910",
```

```
      "name": "my-connected-account",
      "adminRole": {
        "name": "LambdaQuokkaRole",
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "admin",
          "lambda",
          "s3",
          "cloudfront"
        ]
      },
      "lambdaRole": {
        "name": "LambdaQuokkaRole",
        "arn": "arn:aws:iam::12345678910:role/LambdaQuokkaRole",
        "capabilities": [
          "lambda",
          "s3"
        ]
      }
    }
  },
}
```

블루프린트에 시크릿 컴포넌트 추가

CodeCatalyst 시크릿을 사용하여 워크플로우에서 참조할 수 있는 민감한 데이터를 저장할 수 있습니다. 사용자 지정 블루프린트에 비밀을 추가하고 워크플로우에서 참조할 수 있습니다. 자세한 정보는 비밀을 사용한 데이터 마스킹을 참조하세요.

Amazon CodeCatalyst 블루프린트를 가져오려면 지역 유형을 입력하십시오.

`blueprint.ts`파일에 다음을 추가합니다.

```
import { Secret, SecretDefinition } from '@amazon-codecatalyst/blueprint-
component.secrets'
```

주제

- 시크릿 생성
- 워크플로우에서 시크릿 참조

시크릿 생성

다음 예제는 사용자에게 비밀 값과 선택적 설명을 입력하라는 메시지를 표시하는 UI 구성요소를 만듭니다.

```
export interface Options extends ParentOptions {
    ...
    mySecret: SecretDefinition;
}


export class Blueprint extends ParentBlueprint {
  constructor(options_: Options) {
    new Secret(this, options.secret);
}
```

비밀 구성 요소에는 a가 필요합니다. name 다음 코드는 필요한 최소 기본 셰이프입니다.

```
{
    ...
    "secret": {
        "name": "secretName"
    },

}
```

워크플로우에서 시크릿 참조

다음 예제 블루프린트는 시크릿과 시크릿 값을 참조하는 워크플로를 생성합니다. 자세한 정보는 워크플로우에서 시크릿 참조을 참조하세요.

```
export interface Options extends ParentOptions {
    ...
/**
*
* @validationRegex /^\w+$/
*/
  username: string;


  password: SecretDefinition;
}
```

```
export class Blueprint extends ParentBlueprint {
  constructor(options_: Options) {
    const password = new Secret(this, options_.password);

    const workflowBuilder = new WorkflowBuilder(this, {
      Name: 'my_workflow',
    });


    workflowBuilder.addBuildAction({
      actionName: 'download_files',
      input: {
        Sources: ['WorkflowSource'],
      },
      output: {
        Artifacts: [{ Name: 'download', Files: ['file1'] }],
      },
      steps: [
        `curl -u ${options_.username}:${password.reference} https://example.com`,
      ],
    });

    new Workflow(
      this,
      repo,
      workflowBuilder.getDefinition(),
    );

}
```

에서 시크릿을 사용하는 방법에 대해 자세히 CodeCatalyst 알아보려면 을 참조하십시오비밀을 사용한 데이터 마스킹.

블루프린트에 영역 컴포넌트 추가

사용자 지정 블루프린트의 Options 인터페이스에 지역 유형을 추가하여 하나 이상의 AWS gion을 입력할 수 있는 블루프린트 마법사에서 구성 요소를 생성할 수 있습니다. gion 유형은 파일의 기본 블루프린트에서 가져올 수 있습니다. blueprint.ts 자세한 내용은 AWS 리전을 참조하십시오.

Amazon CodeCatalyst 블루프린트 지역 유형을 가져오려면

blueprint.ts파일에 다음을 추가합니다.

```
import { Region } from '@amazon-codecatalyst/blueprints.blueprint'
```

지역 유형 파라미터는 선택할 수 있는 AWS 지역 코드의 배열이거나 지원되는 모든 AWS 지역을 포함하는 * 데 사용할 수 있습니다.

주제

주석

Options인터페이스의 각 필드에 JSDoc 태그를 추가하여 마법사에서 필드가 나타나고 동작하는 방식을 사용자 정의할 수 있습니다. 지역 유형의 경우 다음 태그가 지원됩니다.

- @displayName주석을 사용하여 마법사에서 필드 레이블을 변경할 수 있습니다.

  예제: `@displayName AWS Region`
- @placeholder주석을 사용하여 선택/다중 선택 구성 요소의 자리 표시자를 변경할 수 있습니다.

  예제: `@placeholder Choose AWS Region`

지역 구성 요소 예제

지정된 목록에서 지역 선택

```
export interface Options extends ParentOptions {
    ...
  /**
   * @displayName Region
   */
  region: Region<['us-east-1', 'us-east-2', 'us-west-1', 'us-west-2']>;
}
```

지정된 목록에서 하나 이상의 지역 선택

```
export interface Options extends ParentOptions {
    ...
  /**
   * @displayName Regions
```

```
   */
  multiRegion: Region<['us-east-1', 'us-east-2', 'us-west-1', 'us-west-2']>[];
}
```

하나의 AWS 지역 선택

```
export interface Options extends ParentOptions {
    ...
  /**
   * @displayName Region
   */
  region: Region<['*']>;
}
```

지정된 목록에서 하나 이상의 지역 선택

```
export interface Options extends ParentOptions {
    ...
  /**
   * @displayName Regions
   */
  multiRegion: Region<['us-east-1', 'us-east-2', 'us-west-1', 'us-west-2']>[];
}
```

블루프린트에 리포지토리 및 소스 코드 구성 요소 추가

Amazon은 리포지토리를 CodeCatalyst 사용하여 코드를 저장합니다. 리포지토리는 이름을 입력으로 사용합니다. 소스 코드 파일, 워크플로, 관리형 개발 환경 (MDE) 과 같은 기타 구성 요소와 같은 대부분의 구성 요소는 리포지토리에 저장됩니다. 또한 소스 리포지토리 구성 요소는 파일 및 정적 자산을 관리하는 데 사용되는 구성 요소를 내보냅니다. 리포지토리에는 이름 제약이 있습니다. 자세한 정보는 소스 리포지토리를 사용하여 코드를 저장하고 공동 작업하십시오. CodeCatalyst 을 참조하세요.

```
const repository = new SourceRepository(this, {
  title: 'my-new-repository-title',
});
```

Amazon CodeCatalyst 블루프린트 리포지토리 및 소스 코드 구성 요소를 가져오려면

`blueprint.ts`파일에 다음을 추가합니다.

```
import {...} from '@caws-blueprint-component/caws-source-repositories'
```

주제

파일 추가

구문을 사용하여 저장소에 텍스트 파일을 쓸 수 SourceFile 있습니다. 작업은 가장 일반적인 사용 사례 중 하나이며 리포지토리, 파일 경로 및 텍스트 콘텐츠가 필요합니다. 파일 경로가 리포지토리 내에 없는 경우 구성 요소는 필요한 모든 폴더를 만듭니다.

```
new SourceFile(repository, `path/to/my/file/in/repo/file.txt`, 'my file contents');
```

> **Note**
>
> 동일한 리포지토리 내의 동일한 위치에 두 파일을 쓰는 경우 가장 최근의 구현이 이전 구현을 덮어씁니다. 이 기능을 사용하여 생성된 코드를 계층화할 수 있으며, 사용자 지정 블루프린트에서 생성했을 수 있는 코드를 확장하는 데 특히 유용합니다.

제네릭 파일 추가

리포지토리에 임의의 비트를 쓸 수 있습니다. 버퍼에서 읽고 File 구문을 사용할 수 있습니다.

```
new File(repository, `path/to/my/file/in/repo/file.img`, new Buffer(...));

new File(repository, `path/to/my/file/in/repo/new-img.img`, new StaticAsset('path/to/
image.png').content());
```

파일 복사

시작 코드를 복사하여 붙여넣은 다음 해당 기반 위에 추가 코드를 생성하여 생성된 코드로 시작할 수 있습니다. 코드를 static-assets 디렉터리 내에 배치한 다음 해당 코드를 StaticAsset 구문으로 대상으로 지정합니다. 이 경우 경로는 항상 static-assets 디렉터리의 루트에서 시작됩니다.

```
const starterCode = new StaticAsset('path/to/file/file.txt')
```

```
const starterCodeText = new StaticAsset('path/to/file/file.txt').toString()
const starterCodeRawContent = new StaticAsset('path/to/image/hello.png').content()

const starterCodePath = new StaticAsset('path/to/image/hello.png').path()
// starterCodePath is equal to 'path/to/image/hello.png'
```

의 서브클래스는 StaticAsset 입니다. SubstitutionAsset 서브클래스의 함수는 완전히 동일하지만 대신 파일에 콧수염 대체를 실행할 수 있습니다. 스타일 생성을 수행하는 copy-and-replace 데 유용할 수 있습니다.

정적 애셋 대체는 콧수염 템플릿 엔진을 사용하여 생성된 소스 리포지토리에 시드되는 정적 파일을 렌더링합니다. Mustache 템플릿 규칙은 렌더링 중에 적용됩니다. 즉, 기본적으로 모든 값이 HTML로 인코딩됩니다. 이스케이프 처리되지 않은 HTML을 렌더링하려면 트리플 머스타치 구문을 사용하십시오. {{{name}}} 자세한 내용은 콧수염 템플릿 규칙을 참조하십시오.

> **Note**
>
> 텍스트 해석이 불가능한 파일을 대체하여 실행하면 오류가 발생할 수 있습니다.

```
const starterCodeText = new SubstitionAsset('path/to/file/file.txt').subsitite({
  'my_variable': 'subbed value1',
  'another_variable': 'subbed value2'
})
```

여러 파일을 대상으로 지정

정적 애셋은 on StaticAsset 및 호출된 해당 서브클래스를 통한 글로브 타깃팅을 지원합니다. 이 함수는 경로findAll(...), 콘텐츠 등이 미리 로드된 정적 애셋 목록을 반환합니다. 목록을 구문으로 연결하여 내용을 File 복사하여 디렉터리에 붙여넣을 수 있습니다. static-assets

```
new File(repository, `path/to/my/file/in/repo/file.img`, new Buffer(...));

new File(repository, `path/to/my/file/in/repo/new-img.img`, new StaticAsset('path/to/
image.png').content());
```

새 리포지토리 생성 및 파일 추가

리포지토리 구성 요소를 사용하여 생성된 프로젝트에 새 리포지토리를 만들 수 있습니다. 그런 다음 생성된 저장소에 파일이나 워크플로를 추가할 수 있습니다.

```
import { SourceRepository } from '@amazon-codecatalyst/codecatalyst-source-
repositories';
...
const repository = new SourceRepository(this, { title: 'myRepo' });
```

다음 예제는 기존 리포지토리에 파일 및 워크플로를 추가하는 방법을 보여줍니다.

```
import { SourceFile } from '@amazon-codecatalyst/codecatalyst-source-repositories';
import { Workflow } from '@amazon-codecatalyst/codecatalyst-workflows';
...
new SourceFile(repository, 'README.md', 'This is the content of my readme');
new Workflow(this, repository, {/**...workflowDefinition...**/});
```

두 코드를 결합하면 소스 파일이 `README.md` 있고 루트에 CodeCatalyst 워크플로가 `myRepo` 있는 이름이 지정된 단일 리포지토리가 생성됩니다.

블루프린트에 워크플로우 구성 요소 추가

워크플로는 Amazon CodeCatalyst 프로젝트에서 트리거를 기반으로 작업을 실행하는 데 사용됩니다. 워크플로 구성 요소를 사용하여 워크플로 YAML 파일을 만들고 구성할 수 있습니다. 자세한 정보는 워크플로우 YAML 정의을 참조하세요.

Amazon CodeCatalyst 블루프린트 워크플로 구성 요소를 가져오려면

`blueprint.ts`파일에 다음을 추가합니다.

```
import { WorkflowBuilder, Workflow } from '@amazon-codecatalyst/codecatalyst-workflows'
```

주제

- 워크플로 구성 요소 예제
- 환경에 연결

워크플로 구성 요소 예제

WorkflowBuilder 구성 요소

클래스를 사용하여 워크플로 정의를 작성할 수 있습니다. 워크플로 구성 요소에 정의를 지정하여 저장소에서 렌더링할 수 있습니다.

```
import { WorkflowBuilder } from '@amazon-codecatalyst/codecatalyst-workflows'
```

```
const workflowBuilder = new WorkflowBuilder({} as Blueprint, {
  Name: 'my_workflow',
});

// trigger the workflow on pushes to branch 'main'
workflowBuilder.addBranchTrigger(['main']);

// add a build action
workflowBuilder.addBuildAction({
  // give the action a name
  actionName: 'build_and_do_some_other_stuff',

  // the action pulls from source code
  input: {
    Sources: ['WorkflowSource'],
  },

  // the output attempts to autodiscover test reports, but not in the node modules
  output: {
    AutoDiscoverReports: {
      Enabled: true,
      ReportNamePrefix: AutoDiscovered,
      IncludePaths: ['**/*'],
      ExcludePaths: ['*/node_modules/**/*'],
    },
  },
  // execute some arbitrary steps
  steps: [
    'npm install',
    'npm run myscript',
    'echo hello-world',
  ],
  // add an account connection to the workflow
  environment: convertToWorkflowEnvironment(myEnv),
});
```

워크플로우 Projen 구성 요소

다음 예제는 Projen 구성 요소를 사용하여 리포지토리에 워크플로 YAML을 작성하는 방법을 보여줍니다.

```
import { Workflow } from '@amazon-codecatalyst/codecatalyst-workflows'
```

```
...

const repo = new SourceRepository
const blueprint = this;
const workflowDef = workflowBuilder.getDefinition()

// creates a workflow.yaml at .aws/workflows/${workflowDef.name}.yaml
new Workflow(blueprint, repo, workflowDef);

// can also pass in any object and have it rendered as a yaml. This is unsafe and may
 not produce a valid workflow
new Workflow(blueprint, repo, {... some object ...});
```

환경에 연결

많은 워크플로를 AWS 계정 연결에서 실행해야 합니다. 워크플로는 계정 및 역할 이름 사양이 지정된 환경에 작업을 연결할 수 있도록 하여 이를 처리합니다.

```
import { convertToWorkflowEnvironment } from '@amazon-codecatalyst/codecatalyst-
workflows'


const myEnv = new Environment(...);

// can be passed into a workflow constructor
const workflowEnvironment = convertToWorkflowEnvironment(myEnv);


// add a build action
workflowBuilder.addBuildAction({
  ...
  // add an account connection to the workflow
  environment: convertToWorkflowEnvironment(myEnv),
});
```

블루프린트에 개발 환경 구성 요소 추가

관리형 개발 환경 (MDE) 은 MDE 워크스페이스를 만들고 구축하는 데 사용됩니다. CodeCatalyst 구성 요소가 파일을 생성합니다. `devfile.yaml` 자세한 내용은 Devfile 소개 및 개발 환경을 위한 리포지토리 devfile 편집 을 참조하십시오.

```
new Workspace(this, repository, SampleWorkspaces.default);
```

Amazon CodeCatalyst 블루프린트 워크스페이스 구성 요소를 가져오려면

`blueprint.ts`파일에 다음을 추가합니다.

```
import {...} from '@amazon-codecatalyst/codecatalyst-workspaces'
```

블루프린트에 이슈 컴포넌트 추가

에서는 기능 CodeCatalyst, 작업, 버그 및 프로젝트와 관련된 기타 작업을 모니터링할 수 있습니다. 각 작업은 이슈라는 별개의 기록에 보관됩니다. 각 이슈에는 설명, 담당자, 상태 및 기타 속성이 있을 수 있으며, 이러한 속성을 검색하고 그룹화하고 필터링할 수 있습니다. 기본 보기를 사용하여 문제를 보거나 사용자 지정 필터링, 정렬 또는 그룹화를 사용하여 자체 보기를 만들 수 있습니다. 문제와 관련된 개념에 대한 자세한 내용은 이슈 개념 및 내 이슈에 대한 할당량 CodeCatalyst 을 참조하십시오.

이슈 구성 요소는 이슈의 JSON 표현을 생성합니다. 구성 요소는 ID 필드와 이슈 정의를 입력으로 받습니다.

Amazon CodeCatalyst 블루프린트 이슈 구성 요소를 가져오려면

`blueprint.ts`파일에 다음을 추가합니다.

```
import {...} from '@amazon-codecatalyst/blueprint-component.issues'
```

주제

- 이슈 구성 요소 예제

이슈 구성 요소 예제

이슈 생성

```
import { Issue } from '@amazon-codecatalyst/blueprint-component.issues';
...
new Issue(this, 'myFirstIssue', {
  title: 'myFirstIssue',
  content: 'This is an example issue.',
});
```

우선순위가 높은 이슈 생성

```
import { Workflow } from '@amazon-codecatalyst/codecatalyst-workflows'
...
const repo = new SourceRepository
const blueprint = this;
const workflowDef = workflowBuilder.getDefinition()

// Creates a workflow.yaml at .aws/workflows/${workflowDef.name}.yaml
new Workflow(blueprint, repo, workflowDef);

// Can also pass in any object and have it rendered as a yaml. This is unsafe and may
 not produce a valid workflow
new Workflow(blueprint, repo, {... some object ...});
```

라벨로 우선순위가 낮은 이슈 생성

```
import { Issue } from '@amazon-codecatalyst/blueprint-component.issues';
...
new Issue(this, 'myThirdIssue', {
  title: 'myThirdIssue',
  content: 'This is an example of a low priority issue with a label.',
  priority: 'LOW',
  labels: ['exampleLabel'],
});
```

블루프린트 툴링 및 CLI로 작업하기

blueprint CLI는 사용자 지정 블루프린트를 관리하고 사용할 수 있는 도구를 제공합니다.

주제

- 블루프린트 툴링으로 작업하기
- 이미지 업로드 도구

블루프린트 툴링으로 작업하기

블루프린트 툴로 작업하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 재개하세요. 자세한 설명은 Dev Environment 재개 섹션을 참조하세요.

개발 환경이 없는 경우 먼저 개발 환경을 만들어야 합니다. 자세한 설명은 Dev Environment 생성 섹션을 참조하세요.

3. 작동하는 터미널에서 다음 명령을 실행하여 blueprint CLI를 설치합니다.

```
npm install -g @amazon-codecatalyst/blueprint-util.cli
```

4. `blueprint.ts`파일에서 사용하려는 도구를 다음 형식으로 가져옵니다.

```
import { <tooling-function-name> } from '@amazon-codecatalyst/blueprint-util.cli/
lib/<tooling-folder-name>/<tooling-file-name>;
```

> **Tip**
>
> 로 이동하여 사용하려는 툴링의 이름을 찾을 수 있습니다. CodeCatalyst blueprints GitHub repository

이미지 업로드 도구를 사용하려면 스크립트에 다음을 추가하세요.

```
import { uploadImagePublicly } from '@amazon-codecatalyst/blueprint-util.cli/lib/
image-upload-tool/upload-image-to-aws';
```

예제

- 게시 기능을 사용하려면 스크립트에 다음을 추가하세요.

```
import { publish } from '@amazon-codecatalyst/blueprint-util.cli/lib/publish/
publish';
```

- 이미지 업로드 도구를 사용하려면 스크립트에 다음을 추가하세요.

```
import { uploadImagePublicly } from '@amazon-codecatalyst/blueprint-util.cli/lib/
image-upload-tool/upload-image-to-aws';
```

5. 함수를 호출합니다.

예제:

- 퍼블리싱 함수를 사용하려면 스크립트에 다음을 추가하세요.

```
await publish(logger, config.publishEndpoint, {<your publishing options>});
```

- 이미지 업로드 도구를 사용하려면 스크립트에 다음을 추가하세요.

```
const {imageUrl, imageName} = await uploadImagePublicly(logger, 'path/to/
image'));
```

이미지 업로드 도구

이미지 업로드 도구를 사용하면 자신의 이미지를 AWS 계정의 S3 버킷에 업로드한 다음 해당 이미지를 공개적으로 배포할 수 CloudFront 있습니다. 이 도구는 로컬 스토리지의 이미지 경로 (및 선택적 버킷 이름) 를 입력으로 받아 공개적으로 사용 가능한 이미지의 URL을 반환합니다. 자세한 내용은 Amazon이란 무엇입니까 CloudFront? 를 참조하십시오. 그리고 아마존 S3란 무엇입니까?

이미지 업로드 도구를 사용하려면

1. 블루프린트 SDK와 샘플 블루프린트에 대한 액세스를 제공하는 오픈 소스 블루프린트 GitHub 리포지토리를 복제하세요. 작동 중인 터미널에서 다음 명령어를 실행합니다.

   ```
   git clone https://github.com/aws/codecatalyst-blueprints.git
   ```

2. 다음 명령을 실행하여 블루프린트 GitHub 리포지토리로 이동합니다.

   ```
   cd codecatalyst-blueprints
   ```

3. 다음 명령을 실행하여 종속성을 설치합니다.

   ```
   yarn && yarn build
   ```

4. 다음 명령을 실행하여 최신 블루프린트 CLI 버전이 설치되어 있는지 확인합니다.

   ```
   yarn upgrade @amazon-codecatalyst/blueprint-util.cli
   ```

5. 이미지를 업로드하려는 S3 버킷으로 AWS 계정에 로그인합니다. 자세한 내용은 AWS CLI 구성 및 AWS 명령줄 인터페이스를 통한 로그인을 참조하십시오.

6. CodeCatalyst 저장소의 루트에서 다음 명령을 실행하여 Blueprint CLI를 사용하여 디렉토리로 이동합니다.

```
cd packages/utils/blueprint-cli
```

7. 다음 명령을 실행하여 S3 버킷에 이미지를 업로드합니다.

```
yarn blueprint upload-image-public <./path/to/your/image>
      <optional:optional-bucket-name>
```

이미지 URL이 생성됩니다. 배포를 배포하는 데 시간이 걸리므로 URL을 즉시 사용할 수 없습니다. CloudFront 배포 상태를 확인하여 최신 배포 상태를 확인하세요. 자세한 내용은 배포판 작업을 참조하십시오.

스냅샷 테스트를 통한 인터페이스 변경 평가

블루프린트의 여러 구성에서 생성된 스냅샷 테스트가 지원됩니다.

블루프린트는 블루프린트 작성자가 제공한 구성에 대한 스냅샷 테스트를 지원합니다. 구성은 블루프린트 루트에 있는 defaults.json 파일 위에 병합되는 부분 오버라이드입니다. 스냅샷 테스트를 활성화하고 구성하면 빌드 및 테스트 프로세스가 지정된 구성을 합성하고 합성된 출력이 참조 스냅샷에서 변경되지 않았는지 확인합니다. 스냅샷 테스트 코드를 보려면 블루프린트 저장소를 참조하세요. CodeCatalyst GitHub

스냅샷 테스트를 활성화하려면

1. `.projenrc.ts`파일에서 스냅샷을 만들려는 파일로 입력 객체를 업데이트하십시오. ProjenBlueprint 예:

```
{
  ....
  blueprintSnapshotConfiguration: {
    snapshotGlobs: ['**', '!environments/**', '!aws-account-to-environment/**'],
  },
}
```

2. 블루프린트를 재합성하여 블루프린트 프로젝트에 TypeScript 파일을 생성합니다. 소스 파일은 Progen에서 유지 관리하고 재생성하므로 편집하지 마세요. 다음 명령을 사용합니다.

```
yarn projen
```

3. src/snapshot-configurations디렉토리로 이동하면 빈 개체가 있는 default-config.json 파일을 볼 수 있습니다. 파일을 업데이트하거나 하나 이상의 자체 테스트 구성으로 교체하십시오. 그런 다음 각 테스트 구성을 프로젝트 defaults.json 파일에 병합하고 합성하여 테스트할 때 스냅샷과 비교합니다. 다음 명령을 사용하여 테스트하십시오.

```
yarn test
```

테스트 명령을 처음 사용하면 다음 메시지가 표시됩니다Snapshot Summary › NN snapshots written from 1 test suite. 후속 테스트 실행은 합성된 출력이 스냅샷에서 변경되지 않았는지 확인하고 다음 메시지를 표시합니다. Snapshots: NN passed, NN total

의도적으로 다른 출력을 생성하도록 블루프린트를 변경한 경우, 다음 명령어를 실행하여 참조 스냅샷을 업데이트하세요.

```
yarn test:update
```

스냅샷은 합성된 출력이 실행 시마다 일정할 것으로 예상합니다. 블루프린트에서 다양한 파일을 생성하는 경우 해당 파일을 스냅샷 테스트에서 제외해야 합니다. ProjenBluerpint입력 blueprintSnapshotConfiguration 객체의 객체를 업데이트하여 snapshotGlobs 프로퍼티를 추가하세요. snapshotGlobs속성은 스냅샷에 포함하거나 제외할 파일을 결정하는 글로브 배열입니다.

> **Note**
>
> 기본 글로브 목록이 있습니다. 목록을 직접 지정하는 경우 기본 항목을 명시적으로 다시 가져와야 할 수 있습니다.

스페이스에 사용자 지정 청사진 게시

스페이스의 청사진 카탈로그에 사용자 정의 청사진을 만들려면 먼저 스페이스에 게시해야 합니다. 게시하기 전에 CodeCatalyst 콘솔에서 청사진을 볼 수도 있습니다. 블루프린트의 프리뷰 버전 또는 일반 버전을 게시할 수 있습니다.

> **Important**
>
> 외부 소스의 블루프린트 패키지를 사용하려는 경우, 해당 패키지에 수반될 수 있는 위험을 고려하세요. 공간에 추가하는 커스텀 블루프린트와 이를 통해 생성되는 코드에 대한 책임은 귀하에게 있습니다.

> **Important**
>
> 사용자 지정 청사진을 스페이스에 게시하려면 스페이스에서 CodeCatalyst 스페이스 관리자 또는 고급 사용자 역할을 가진 계정으로 로그인해야 합니다.

주제

사용자 지정 청사진의 미리 보기 및 게시

사용자 지정 청사진을 스페이스의 청사진 카탈로그에 추가하려는 경우 사용자 지정 청사진의 미리보기 버전을 스페이스에 게시할 수 있습니다. 이렇게 하면 미리보기가 아닌 버전을 카탈로그에 추가하기 전에 사용자로 블루프린트를 볼 수 있습니다. 프리뷰 버전을 사용하면 실제 버전을 사용하지 않고도 퍼블리싱할 수 있습니다. 예를 들어, 버전을 작업하는 경우 미리 보기 `0.0.1` 버전을 게시하고 추가하여 두 번째 버전에 대한 새 업데이트를 게시하고 추가할 수 있습니다`0.0.2`.

변경한 후에는 `package.json` 파일을 실행하여 커스텀 블루프린트 패키지를 다시 빌드하고 변경 내용을 미리 보세요.

커스텀 블루프린트의 프리뷰 버전을 보고 퍼블리시하려면

1. 개발 환경을 재개하세요. 자세한 내용은 Dev Environment 재개 단원을 참조하세요.
2. 개발 환경에서 작동하는 터미널을 엽니다.
3. (선택 사항) 프로젝트에 필요한 종속 항목을 아직 설치하지 않았다면 제대로 작동하는 터미널에서 설치하세요. 다음 명령을 사용합니다.

```
yarn
```

4. (선택 사항) `.projenrc.ts` 파일을 변경했다면, 블루프린트를 빌드하고 미리 보기 전에 프로젝트 구성을 다시 생성하세요. 다음 명령을 사용합니다.

   ```
   yarn projen
   ```

5. 다음 명령어를 사용하여 커스텀 블루프린트를 다시 빌드하고 미리 보세요. 다음 명령을 사용합니다.

   ```
   yarn blueprint:preview
   ```

   제공된 `See this blueprint at:` 링크로 이동하여 커스텀 블루프린트를 미리 보세요. 구성에 따라 텍스트를 포함한 UI가 예상대로 나타나는지 확인하세요. 커스텀 블루프린트를 변경하려면 `blueprint.ts` 파일을 편집하고 블루프린트를 재합성한 다음 프리뷰 버전을 다시 퍼블리시하면 됩니다. 자세한 내용은 재합성 단원을 참조하십시오.

6. (선택 사항) 커스텀 블루프린트의 프리뷰 버전을 퍼블리시한 다음 스페이스의 블루프린트 카탈로그에 추가할 수 있습니다. `Enable version` *`[preview version number]`* `at:`링크를 탐색하여 프리뷰 버전을 스페이스에 게시하세요.

에서 프로젝트를 만들지 않고도 프로젝트 생성을 에뮬레이션할 수 있습니다. CodeCatalyst 프로젝트를 합성하려면 다음 명령어를 사용하세요.

```
yarn blueprint:synth
```

폴더에 블루프린트가 생성됩니다. `synth/synth.`*`[options-name]`*`/proposed-bundle/` 자세한 내용은 종합 단원을 참조하십시오.

커스텀 블루프린트를 업데이트하는 경우, 대신 다음 명령어를 사용하여 프로젝트를 재합성하세요.

```
yarn blueprint:resynth
```

폴더에 블루프린트가 생성됩니다. `synth/synth.`*`[options-name]`*`/proposed-bundle/` 자세한 내용은 재합성 단원을 참조하십시오.

미리 보기 버전을 게시한 후 청사진을 추가하여 스페이스 구성원이 이를 사용하여 새 프로젝트를 만들거나 기존 프로젝트를 추가할 수 있습니다. 자세한 내용은 우주 청사진 카탈로그에 사용자 지정 청사진 추가 단원을 참조하십시오.

커스텀 블루프린트의 일반 버전 보기 및 게시

사용자 지정 청사진을 개발하고 미리 본 후 스페이스의 청사진 카탈로그에 추가하려는 새 버전을 보고 게시할 수 있습니다. 프로젝트 생성 시 생성되는 릴리스 워크플로는 푸시된 변경 사항을 자동으로 게시합니다. 블루프린트를 만들 때 워크플로우 생성을 끄면 블루프린트가 스페이스의 블루프린트 카탈로그에 자동으로 추가되지 않습니다. 커맨드를 실행한 후에도 사용자 정의 블루프린트를 스페이스에 퍼블리시할 수 있습니다. `yarn`

커스텀 블루프린트 보기 및 게시하기

1. 개발 환경을 재개하세요. 자세한 내용은 Dev Environment 재개 단원을 참조하세요.
2. 개발 환경에서 작동하는 터미널을 엽니다.
3. • 블루프린트를 만들 때 릴리스 워크플로 생성을 옵트아웃한 경우 다음 명령어를 사용하세요.

   ```
   yarn blueprint:release
   ```

   여전히 제공된 `See this blueprint at:` 링크로 이동하여 커스텀 블루프린트를 볼 수 있습니다.

   사용자 지정 청사진의 업데이트된 버전을 게시한 다음 스페이스의 청사진 카탈로그에 추가할 수 있습니다. `Enable version` *`[release version number]`* `at:`링크를 탐색하여 최신 버전을 스페이스에 게시하세요.

   • 블루프린트를 만들 때 릴리스 워크플로를 선택한 경우, 변경 사항이 푸시되면 최신 블루프린트 버전이 자동으로 게시됩니다. 다음 명령을 사용합니다.

   ```
   git add .
   ```

   ```
   git commit -m "commit message"
   ```

   ```
   git push
   ```

일반 버전을 게시한 후, 스페이스 구성원이 블루프린트를 사용하여 새 프로젝트를 만들거나 기존 프로젝트를 추가할 수 있도록 블루프린트를 추가할 수 있습니다. 자세한 내용은 우주 청사진 카탈로그에 사용자 지정 청사진 추가 단원을 참조하십시오.

지정된 공간 및 프로젝트에 커스텀 블루프린트 게시 및 추가

기본적으로 `blueprint:preview` 및 `blueprint:release` 명령은 청사진을 만든 CodeCatalyst 공간에 게시됩니다. Enterprise 스페이스가 여러 개 있는 경우 해당 스페이스에서 동일한 블루프린트를 미리 보고 게시할 수도 있습니다. 다른 공간의 기존 프로젝트에 청사진을 추가할 수도 있습니다.

지정된 공간에 사용자 정의 청사진을 게시하거나 추가하려면

1. 개발 환경을 재개하세요. 자세한 내용은 Dev Environment 재개 단원을 참조하십시오.
2. 개발 환경에서 작동하는 터미널을 엽니다.
3. (선택 사항) 프로젝트에 필요한 종속 항목을 아직 설치하지 않았다면 설치하세요. 다음 명령을 사용합니다.

   ```
   yarn
   ```

4. `--space`태그를 사용하여 미리 보기 또는 일반 버전을 지정된 공간에 게시할 수 있습니다. 예:

   - ```
     yarn blueprint:preview --space my-awesome-space # publishes under a "preview"
      version tag to 'my-awesome-space'
     ```

     출력 예제:

     ```
     Enable version 0.0.1-preview.0 at: https://codecatalyst.aws/spaces/my-awesome-
     space/blueprints
     Blueprint applied to [NEW]: https://codecatalyst.aws/spaces/my-awesome-space/
     blueprints/%40amazon-codecatalyst%2Fmyspace.my-blueprint/publishers/1524817d-
     a69b-4abe-89a0-0e4a9a6c53b2/versions/0.0.1-preview.0/projects/create
     ```

   - ```
     yarn blueprint:release --space my-awesome-space # publishes normal version to
      'my-awesome-space'
     ```

     출력 예제:

     ```
     Enable version 0.0.1 at: https://codecatalyst.aws/spaces/my-awesome-space/
     blueprints
     Blueprint applied to [NEW]: https://codecatalyst.aws/spaces/my-awesome-space/
     blueprints/%40amazon-codecatalyst%2Fmyspace.my-blueprint/publishers/1524817d-
     a69b-4abe-89a0-0e4a9a6c53b2/versions/0.0.1/projects/create
     ```

`--project`를 사용하여 사용자 지정 블루프린트의 미리 보기 버전을 지정된 공간에 있는 기존 프로젝트에 추가할 수 있습니다. 예:

```
yarn blueprint:preview --space my-awesome-space --project my-project # previews
 blueprint application to an existing project
```

출력 예제:

```
Enable version 0.0.1-preview.1 at: https://codecatalyst.aws/spaces/my-awesome-
space/blueprints
Blueprint applied to [my-project]: https://codecatalyst.aws/spaces/my-awesome-
space/projects/my-project/blueprints/%40amazon-codecatalyst%2FmySpace.my-blueprint/
publishers/1524817d-a69b-4abe-89a0-0e4a9a6c53b2/versions/0.0.1-preview.1/add
```

커스텀 블루프린트의 세부 정보, 버전, 프로젝트 보기

청사진의 세부 정보, 버전, 청사진을 사용하여 생성하거나 청사진을 추가한 프로젝트를 포함하여 스페이스에 게시된 사용자 지정 청사진을 볼 수 있습니다.

주제

- 스페이스의 사용자 지정 청사진 보기
- 커스텀 블루프린트로 생성하거나 추가한 프로젝트 보기

스페이스의 사용자 지정 청사진 보기

스페이스의 사용자 정의 청사진을 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 커스텀 블루프린트를 보려는 공간으로 이동합니다.
3. 스페이스 대시보드에서 설정 탭을 선택한 다음 블루프린트를 선택하여 스페이스 블루프린트를 확인합니다. 표에는 다음과 같은 세부 정보가 표시됩니다.

   - 이름 - 커스텀 블루프린트의 이름.
   - 카탈로그 상태 - 사용자 정의 청사진을 스페이스의 청사진 카탈로그에 게시할지 여부.
   - 최신 버전 - 사용자 지정 청사진의 최신 버전입니다.

- 최근 수정 날짜 - 우주 청사진이 마지막으로 업데이트된 날짜.

커스텀 블루프린트로 생성하거나 추가한 프로젝트 보기

커스텀 블루프린트로 생성하거나 추가한 프로젝트를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 커스텀 블루프린트를 보려는 공간으로 이동합니다.
3. 스페이스 대시보드에서 설정 탭을 선택한 다음 블루프린트를 선택합니다.
4. 스페이스 블루프린트 테이블에서 커스텀 블루프린트 이름을 선택하면 블루프린트를 사용하는 프로젝트와 블루프린트를 사용하지 않는 프로젝트를 볼 수 있습니다.

우주 청사진 카탈로그에 사용자 지정 청사진 추가

사용자 지정 청사진을 공간에 게시한 후 공간의 청사진 카탈로그에 추가할 수 있습니다. CodeCatalyst 스페이스의 청사진 카탈로그에 사용자 정의 청사진을 추가하면 모든 스페이스 구성원이 프로젝트를 만들거나 기존 프로젝트에 추가할 때 청사진을 사용할 수 있습니다. 스페이스의 청사진 카탈로그에 사용자 정의 청사진을 추가하기 전에 청사진의 게시 권한을 활성화해야 합니다. 워크플로 릴리스 생성을 선택한 경우 게시 권한이 기본적으로 활성화됩니다. 자세한 내용은 커스텀 블루프린트에 대한 퍼블리싱 권한 설정 및 스페이스에 사용자 지정 청사진 게시 섹션을 참조하세요.

> **Important**
>
> CodeCatalyst 스페이스의 청사진 카탈로그에 사용자 지정 청사진을 추가하려면 스페이스에서 스페이스 관리자 또는 고급 사용자 역할을 가진 계정으로 로그인해야 합니다.

스페이스의 청사진 카탈로그에 청사진을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 블루프린트는 소스 리포지토리의 기본 브랜치에서만 추가할 수 있습니다. 피처 브랜치에서 블루프린트를 개발한 경우, 피처 브랜치를 기본 브랜치의 변경 사항과 병합하세요. 모든 변경 사항을 기본 브랜치에 병합하려면 풀 리퀘스트를 생성하세요. 자세한 정보는 Amazon에서 풀 리퀘스트를 사용하여 코드 검토하기 CodeCatalyst을 참조하세요.
3. CodeCatalyst 콘솔에서 커스텀 블루프린트가 있는 스페이스 대시보드로 이동하세요.
4. 스페이스 대시보드에서 설정 탭을 선택한 다음 블루프린트를 선택합니다.

5. 추가하려는 블루프린트 이름을 선택한 다음 카탈로그에 추가를 선택합니다. 버전이 두 개 이상인 경우, 카탈로그 버전 드롭다운 메뉴에서 버전을 선택하세요.
6. 저장을 선택합니다.

우주 청사진 카탈로그에서 사용자 지정 청사진 제거

사용자 지정 청사진을 새 프로젝트를 생성하는 데 더 이상 사용하거나 기존 프로젝트에 적용하지 않으려면 스페이스의 청사진 카탈로그에서 사용자 지정 청사진을 제거할 수 있습니다.

> **Note**
>
> 스페이스의 청사진 카탈로그에서 사용자 지정 청사진을 제거해도 청사진에서 만든 프로젝트 또는 해당 청사진을 적용한 프로젝트에는 영향을 주지 않습니다. 블루프린트의 리소스는 프로젝트에서 제거되지 않습니다.

> **Important**
>
> CodeCatalyst 스페이스의 청사진 카탈로그에서 사용자 지정 청사진을 제거하려면 스페이스에서 스페이스 관리자 또는 고급 사용자 역할을 가진 계정으로 로그인해야 합니다.

우주 청사진 카탈로그에서 사용자 정의 청사진을 제거하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 커스텀 블루프린트가 있는 스페이스 대시보드로 이동하세요.
3. 스페이스 대시보드에서 설정 탭을 선택한 다음 블루프린트를 선택합니다.
4. 제거하려는 블루프린트 이름을 선택한 다음 카탈로그에서 블루프린트 제거를 선택합니다.

커스텀 블루프린트에 대한 퍼블리싱 권한 설정

기본적으로 프로젝트 생성 중에 워크플로 릴리스가 생성된 경우 사용자 지정 블루프린트 권한이 활성화됩니다. 게시 권한이 활성화되면 청사진을 스페이스에 게시할 수 있습니다. 블루프린트를 게시할 수 없도록 권한을 비활성화할 수 있습니다. 권한이 비활성화되면 블루프린트 생성 중에 생성된 릴리스 워크플로가 실행되지 않습니다. 블루프린트의 권한이 활성화되지 않으면 블루프린트의 새 변경사항을 게시할 수 없습니다.

> **Important**
>
> 커스텀 블루프린트 프로젝트의 퍼블리싱 권한을 활성화하거나 비활성화하려면 스페이스에서 스페이스 관리자 또는 파워 사용자 역할이 있는 계정으로 로그인해야 합니다.

블루프린트 프로젝트의 퍼블리싱 권한을 설정하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 사용자 지정 블루프린트의 퍼블리싱 권한을 관리하려는 스페이스로 이동합니다.
3. 스페이스 대시보드에서 설정 탭을 선택한 다음 블루프린트를 선택합니다.
4. 프로젝트 퍼블리싱 권한 탭을 선택하면 스페이스의 모든 블루프린트에 대한 퍼블리싱 권한을 볼 수 있습니다.
5. 관리하려는 블루프린트를 선택한 다음 활성화 또는 비활성화를 선택하여 게시 권한을 변경합니다. 권한을 활성화하는 경우 권한 변경 세부 정보를 검토한 다음 블루프린트 게시 활성화를 선택하여 변경을 확인합니다.

커스텀 블루프린트의 카탈로그 버전 변경

블루프린트 작성자는 스페이스의 블루프린트 카탈로그에 게시하려는 버전을 관리할 수 있습니다. 블루프린트의 카탈로그 버전을 변경해도 다른 블루프린트 버전을 사용하는 프로젝트에는 영향을 주지 않습니다.

> **Important**
>
> Amazon CodeCatalyst 스페이스의 청사진 카탈로그에 있는 사용자 지정 청사진의 카탈로그 버전을 변경하려면 스페이스에서 스페이스 관리자 또는 고급 사용자 역할을 가진 계정으로 로그인해야 합니다.

사용자 지정 블루프린트 버전을 관리하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 커스텀 블루프린트의 버전을 변경하려는 공간으로 이동합니다.
3. 스페이스 대시보드에서 설정 탭을 선택한 다음 블루프린트를 선택합니다.

4. 스페이스 블루프린트 테이블에서 관리하려는 커스텀 블루프린트의 라디오 버튼을 선택합니다.
5. [카탈로그 버전 만들기] 를 선택한 다음, [카탈로그 버전] 드롭다운 메뉴에서 버전을 선택합니다.
6. 저장을 선택합니다.

게시된 사용자 지정 블루프린트 또는 버전 삭제

Amazon CodeCatalyst 스페이스에서 사용자 지정 블루프린트 버전 또는 블루프린트 자체를 삭제하면 블루프린트 프로젝트 또는 블루프린트 버전의 리소스에 대한 모든 액세스 권한이 제거됩니다. 블루프린트 버전이나 블루프린트를 삭제하면 프로젝트 구성원은 프로젝트 리소스에 액세스할 수 없게 되며 타사 소스 리포지토리에서 요청하는 모든 워크플로가 중지됩니다.

> **Note**
>
> 블루프린트를 삭제해도 블루프린트가 적용된 프로젝트에는 영향을 주지 않습니다. 블루프린트의 리소스는 프로젝트에서 제거되지 않습니다.

블루프린트 버전이 스페이스의 청사진 카탈로그에 게시된 경우, 게시된 버전을 삭제하기 전에 카탈로그의 새 버전을 선택하십시오.

> **Important**
>
> 게시된 사용자 정의 청사진 또는 사용자 정의 청사진의 카탈로그 버전을 스페이스에서 삭제하려면 스페이스에서 스페이스 관리자 또는 고급 사용자 역할을 가진 계정으로 로그인해야 합니다.

커스텀 블루프린트의 카탈로그 버전을 삭제하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 커스텀 블루프린트의 카탈로그 버전을 삭제하려는 공간으로 이동합니다.
3. 스페이스 대시보드에서 설정 탭을 선택한 다음 블루프린트를 선택합니다.
4. 삭제하려는 카탈로그 버전이 있는 블루프린트의 이름을 선택합니다.
5. 삭제하려는 카탈로그 버전의 라디오 버튼을 선택한 다음 버전 삭제를 선택합니다.
6. 세부 정보를 검토한 다음 새 블루프린트 카탈로그 버전 선택 드롭다운 메뉴에서 다른 블루프린트 버전을 선택합니다.

7. delete를 입력하여 블루프린트 카탈로그 버전 삭제를 확인합니다.
8. 삭제를 선택합니다.

블루프린트 버전이 스페이스의 블루프린트 카탈로그에 없는 경우, 새 버전을 선택하지 않고도 버전을 삭제할 수 있습니다.

커스텀 블루프린트 버전 삭제하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 커스텀 블루프린트 버전을 삭제하려는 공간으로 이동합니다.
3. 스페이스 대시보드에서 설정 탭을 선택한 다음 블루프린트를 선택합니다.
4. 삭제하려는 버전의 블루프린트 이름을 선택합니다.
5. 삭제하려는 버전의 라디오 버튼을 선택한 다음 버전 삭제를 선택합니다.
6. delete를 입력하여 블루프린트 버전 삭제를 확인합니다.
7. 삭제를 선택합니다.

스페이스의 청사진 카탈로그에서 청사진을 삭제하면 청사진의 모든 버전이 삭제됩니다. 블루프린트를 사용하는 스페이스 프로젝트는 삭제의 영향을 받지 않습니다.

커스텀 블루프린트 버전 삭제하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 콘솔에서 커스텀 블루프린트를 삭제하려는 공간으로 이동합니다.
3. 스페이스 대시보드에서 설정 탭을 선택한 다음 블루프린트를 선택합니다.
4. 스페이스 블루프린트 테이블에서 삭제하려는 커스텀 블루프린트의 라디오 버튼을 선택한 다음 블루프린트 삭제를 선택합니다.
5. delete를 입력하여 커스텀 블루프린트 삭제를 확인합니다.
6. 삭제를 선택합니다.

종속성, 불일치 및 도구 처리

주제

- 종속성 추가
- 종속성 유형 불일치 처리

- yarn 및 npm 사용
- 툴링 및 컴포넌트 업그레이드

종속성 추가

블루프린트 작성자는 블루프린트에 패키지 (예:) 를 추가해야 할 수도 있습니다. `@amazon-codecatalyst/blueprint-component.environments` 해당 패키지로 `projen.ts` 파일을 업데이트한 다음 Projen으로 프로젝트 구성을 다시 생성해야 합니다. Projen은 각 블루프린트 코드베이스의 프로젝트 모델 역할을 하며, 모델이 구성 파일을 렌더링하는 방식을 변경하여 이전 버전과 호환되는 툴링 업데이트를 푸시할 수 있는 기능을 제공합니다. `package.json`파일은 Projen 모델이 부분적으로 소유하는 파일입니다. Projen은 package.json 파일에 포함된 종속성 버전을 인정하지만 다른 옵션은 모델에서 가져와야 합니다.

종속성 추가 및 파일 업데이트하기 **`projenrc.ts`**

1. `projen.ts`파일에서 deps 섹션으로 이동합니다.
2. 블루프린트에 사용하려는 종속성을 추가하세요.
3. 다음 명령어를 사용하여 프로젝트 구성을 재생성하세요.

```
yarn projen && yarn
```

종속성 유형 불일치 처리

Yarn 업데이트 후 리포지토리 매개변수와 관련하여 다음 오류가 발생할 수 있습니다.

```
Type 'SourceRepository' is missing the following properties from type
 'SourceRepository': synthesisSteps, addSynthesisStep
```

이 오류는 한 구성 요소가 다른 구성 요소의 새 버전에 의존하지만 의존하는 구성 요소가 이전 버전에 고정되어 있을 때 발생하는 종속성 불일치로 인한 것입니다. 모든 구성 요소가 동일한 버전을 사용하도록 설정하여 두 구성 요소 간에 버전이 동기화되도록 하면 오류를 해결할 수 있습니다. 버전을 어떻게 다루는지 확실치 않은 이상, 블루프린트가 제공하는 모든 패키지를 같은 최신 버전 (`0.0.x`) 으로 유지하는 것이 가장 좋습니다. 다음 예제는 모든 종속성이 동일한 버전에 의존하도록 `package.json` 파일을 구성하는 방법을 보여줍니다.

```
...
"@caws-blueprint-component/caws-environments": "^0.1.12345",
```

```
"@caws-blueprint-component/caws-source-repositories": "^0.1.12345",
"@caws-blueprint-component/caws-workflows": "^0.1.12345",
"@caws-blueprint-component/caws-workspaces": "^0.1.12345",
"@caws-blueprint-util/blueprint-utils": "^0.1.12345",
...
"@caws-blueprint/blueprints.blueprint": "*",
```

모든 종속성에 대한 버전을 구성한 후 다음 명령을 사용하십시오.

```
yarn install
```

yarn 및 npm 사용

블루프린트는 툴링에 Yarn을 사용합니다. npm과 Yarn을 사용하면 종속성 트리가 각각 해결되는 방식이 다르기 때문에 툴링 문제가 발생합니다. 이러한 문제를 피하려면 Yarn만 사용하는 것이 가장 좋습니다.

npm을 사용하여 실수로 종속성을 설치한 경우 생성된 `package-lock.json` 파일을 제거하고 필요한 종속 항목으로 `.projenrc.ts` 파일이 업데이트되었는지 확인할 수 있습니다. Projen을 사용하여 프로젝트 구성을 다시 생성합니다.

모델을 재생성하려면 다음을 사용하십시오.

```
yarn projen
```

.projenrc.ts 파일이 필요한 종속 항목으로 업데이트되었는지 확인한 후 다음 명령을 사용하십시오.

```
yarn
```

툴링 및 컴포넌트 업그레이드

경우에 따라 툴링과 부품을 업그레이드하여 새 기능을 사용할 수 있게 해야 할 수도 있습니다. 버전을 처리하는 방법이 확실하지 않으면 모든 구성 요소를 동일한 버전으로 유지하는 것이 좋습니다. 구성 요소 간에 버전이 동기화되므로 모든 구성 요소의 버전이 같으면 구성 요소 간에 적절한 종속성이 보장됩니다.

Yarn 작업 공간 모노레포 사용

다음 명령어를 사용하여 커스텀 블루프린트 리포지토리의 루트에서 유틸리티와 컴포넌트를 업그레이드하세요.

```
yarn upgrade @amazon-codecatalyst/*
```

모노레포를 사용하지 않는 경우 다음 명령어를 사용하세요.

```
yarn upgrade –pattern @amazon-codecatalyst/*
```

도구 및 구성 요소를 업그레이드하는 데 사용할 수 있는 기타 옵션:

- npm `@caws-blueprint-component/<some-component>` view를 사용하여 최신 버전을 다운로드하세요.
- package.json 파일에서 버전을 설정하고 다음 명령을 사용하여 수동으로 최신 버전으로 업그레이드하세요. `yarn` 모든 구성 요소와 유틸리티의 버전이 같아야 합니다.

기여하기

블루프린트 소프트웨어 개발 키트 (SDK) 는 여러분이 기여할 수 있는 오픈 소스 라이브러리입니다. 기여자로서 기여 가이드라인, 피드백, 결함을 고려해 보세요. 자세한 내용은 블루프린트 GitHub 저장소를 참조하십시오.

청사진 할당량 CodeCatalyst

다음 표에는 Amazon의 블루프린트 할당량 및 한도가 설명되어 있습니다. CodeCatalyst CodeCatalystAmazon의 할당량에 대한 자세한 내용은 을 참조하십시오. 에 대한 할당량 CodeCatalyst

프로젝트당 적용되는 최대 블루프린트 수 CodeCatalyst	100

소스 리포지토리를 사용하여 코드를 저장하고 공동 작업하십시오. CodeCatalyst

CodeCatalyst 소스 리포지토리는 Amazon에서 호스팅되는 Git 리포지토리입니다. CodeCatalyst 에서 CodeCatalyst 소스 리포지토리를 사용하여 프로젝트의 자산을 안전하게 저장, 버전 관리 및 관리할 수 있습니다.

CodeCatalyst 리포지토리의 에셋에는 다음이 포함될 수 있습니다.

- Documents
- 소스 코드
- 바이너리 파일

CodeCatalyst 또한 프로젝트의 소스 리포지토리를 사용하여 워크플로 구성 파일과 같은 프로젝트의 구성 정보를 저장합니다.

CodeCatalyst 프로젝트에 소스 리포지토리가 두 개 이상 있을 수 있습니다. 예를 들어 프런트 엔드 소스 코드, 백엔드 소스 코드, 유틸리티 및 설명서를 위한 별도의 소스 리포지토리를 원할 수 있습니다.

다음은 소스 리포지토리, 풀 리퀘스트 및 개발 환경의 코드로 작업할 때 사용할 수 있는 한 가지 워크플로입니다. CodeCatalyst

Mary Major는 블루프린트를 CodeCatalyst 사용하여 웹 애플리케이션 프로젝트를 생성합니다. 블루프린트는 샘플 코드가 포함된 소스 리포지토리를 생성합니다. 그녀는 친구인 리 후안, 산비 사카르, 호르헤 수자를 초대하여 함께 프로젝트를 진행해 보자고 한다. Li Juan은 소스 저장소의 샘플 코드를 보고 몇 가지 빠른 변경을 통해 코드에 테스트를 추가하기로 결정합니다. Li는 개발 환경을 만들고, AWS Cloud9 환경으로 선택하고IDE, 새 브랜치를 지정합니다.*test-code*. 개발 환경이 열립니다. Li는 코드를 빠르게 추가한 다음, 소스 리포지토리의 변경 내용이 포함된 브랜치를 커밋하고 푸시합니다. CodeCatalyst 그런 다음 Li는 풀 리퀘스트를 생성합니다. 풀 리퀘스트를 작성하는 과정에서 Li는 호르헤 수자와 사안비 사카르를 검토자로 추가하여 코드가 검토되도록 합니다.

코드를 검토하는 동안 Jorge Souza는 자신이 개발 중인 앱의 프로토타입이 들어 있는 자체 프로젝트 저장소가 GitHub 있다는 사실을 기억합니다. 그는 Mary Major에게 리포지토리를 프로젝트에 추가 소스 GitHub 리포지토리로 연결할 수 있는 확장 프로그램을 설치하고 구성해 달라고 요청합니다. Mary는 에서 리포지토리를 검토하고 GitHub Jorge와 협력하여 리포지토리를 프로젝트의 추가 소스 GitHub 리포지토리로 연결할 수 있도록 GitHub 확장을 구성합니다.

CodeCatalyst 소스 리포지토리는 Git의 표준 기능을 지원하며 기존 Git 기반 도구와 함께 작동합니다. Git 클라이언트 또는 통합 개발 환경 (PATs) 에서 소스 리포지토리를 복제하고 작업할 때 개인용 액세스 토큰 () 을 애플리케이션별 비밀번호로 만들어 사용할 수 있습니다. IDEs 이는 사용자 PATs ID와 연결됩니다. CodeCatalyst 자세한 내용은 개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여 단원을 참조하십시오.

CodeCatalyst 소스 리포지토리는 풀 리퀘스트를 지원합니다. 이는 한 브랜치에서 다른 브랜치로 코드 변경 사항을 병합하기 전에 사용자와 다른 프로젝트 구성원이 코드 변경 내용을 검토하고 이에 대한 의견을 제시할 수 있는 간단한 방법입니다. CodeCatalyst 콘솔에서 변경 내용을 확인하고 코드 라인에 댓글을 달 수 있습니다.

CodeCatalyst 소스 리포지토리의 브랜치로 푸시하면 워크플로에서 자동으로 실행되어 변경 내용을 빌드, 테스트 및 배포할 수 있습니다. 프로젝트 템플릿을 사용하여 프로젝트의 일부로 소스 리포지토리를 만든 경우 하나 이상의 워크플로가 프로젝트의 일부로 구성됩니다. 언제든지 리포지토리에 워크플로를 추가할 수 있습니다. 프로젝트의 워크플로에 대한 YAML 구성 파일은 해당 워크플로의 소스 작업에 구성된 소스 리포지토리에 저장됩니다. 자세한 내용은 워크플로우 시작하기 단원을 참조하십시오.

주제

소스 리포지토리 개념

CodeCatalyst 소스 리포지토리를 사용할 때 알아두어야 할 몇 가지 개념은 다음과 같습니다.

주제

- 개발 환경
- 개인용 액세스 토큰 () PATs
- 브랜치
- 기본 브랜치
- 커밋
- 풀 요청
- 개정
- 워크플로

프로젝트

프로젝트는 개발 팀과 작업을 CodeCatalyst 지원하는 공동 작업을 나타냅니다. 프로젝트가 완성되면 사용자와 리소스를 추가, 업데이트 또는 제거하고, 프로젝트 대시보드를 사용자 지정하고, 팀 작업의 진행 상황을 모니터링할 수 있습니다. 스페이스 내에 여러 프로젝트를 포함할 수 있습니다.

소스 리포지토리는 스페이스에서 생성하거나 링크하는 프로젝트에만 해당됩니다. 프로젝트 간에 저장소를 공유할 수 없으며 저장소를 스페이스에 있는 둘 이상의 프로젝트에 연결할 수 없습니다. 프로젝트에서 기여자 또는 프로젝트 관리자 역할을 가진 사용자는 해당 역할에 부여된 권한에 따라 해당 프로젝트와 관련된 소스 저장소와 상호 작용할 수 있습니다. 자세한 내용은 사용자 역할을 통한 액세스 권한 부여 단원을 참조하십시오.

소스 리포지토리

소스 리포지토리는 프로젝트의 코드와 파일을 안전하게 저장하는 곳입니다. 또한 파일의 버전 기록도 저장합니다. 기본적으로 소스 리포지토리는 CodeCatalyst 프로젝트의 다른 사용자와 공유됩니다. 프로젝트의 소스 리포지토리는 두 개 이상 있을 수 있습니다. 에서 프로젝트의 소스 리포지토리를 만들거나 CodeCatalyst, 설치된 확장 프로그램에서 해당 서비스를 지원하는 경우 다른 서비스에서 호스팅하는 기존 소스 리포지토리를 연결하도록 선택할 수 있습니다. 예를 들어 리포지토리 확장을 설치한 후 GitHub GitHub 리포지토리를 프로젝트에 연결할 수 있습니다. 자세한 내용은 프로젝트의 리포지토리에 소스 코드 저장 CodeCatalyst 및 빠른 시작: 확장 프로그램 설치, 공급자 연결, 리소스 연결 CodeCatalyst 단원을 참조하세요.

개발 환경

개발 환경은 프로젝트의 소스 리포지토리에 저장된 코드를 빠르게 작업하는 CodeCatalyst 데 사용할 수 있는 클라우드 기반 개발 환경입니다. Dev Environment에 포함된 프로젝트 도구와 애플리케이션 라

이브러리는 프로젝트의 소스 저장소에 있는 devfile에 의해 정의됩니다. 소스 리포지토리에 devfile이 없는 경우 기본 devfile이 자동으로 적용됩니다. 기본 devfile에는 가장 자주 사용되는 프로그래밍 언어 및 프레임워크용 도구가 포함되어 있습니다. 기본적으로 개발 환경은 2코어 프로세서, 4GB, 16GiB의 RAM 영구 스토리지로 구성됩니다.

소스 리포지토리의 기존 브랜치를 개발 환경에 복제하거나 개발 환경 생성의 일환으로 새 브랜치를 만들도록 선택할 수 있습니다.

개인용 액세스 토큰 () PATs

개인용 액세스 토큰 (PAT) 은 비밀번호와 비슷합니다. 이는 내 모든 공간 및 프로젝트에서 사용할 수 있도록 사용자 ID와 연결되어 CodeCatalyst 있습니다. 통합 개발 환경 (IDEs) 및 Git 기반 소스 리포지토리를 포함하는 CodeCatalyst 리소스에 액세스하는 PATs 데 사용합니다. PATs사용자를 CodeCatalyst 대표하고 사용자 설정에서 관리할 수 있습니다. 한 사용자가 둘 이상을 가질 수 PAT 있습니다. 개인용 액세스 토큰은 한 번만 표시됩니다. 로컬 컴퓨터에 안전하게 보관하는 것이 가장 좋습니다. 기본적으로 1년 후에 PATs 만료됩니다.

통합 개발 환경 (IDEs) 으로 작업하는 경우 PATs Git 암호와 동일합니다. Git PAT 리포지토리를 사용하도록 설정할 때 암호를 묻는 메시지가 표시되면 입력합니다. IDE Git 기반 IDE 리포지토리와 연결하는 방법에 대한 자세한 내용은 해당 설명서를 참조하십시오. IDE

브랜치

브랜치는 Git과 in의 커밋에 대한 포인터 또는 참조입니다. CodeCatalyst 브랜치를 사용하여 작업을 정리할 수 있습니다. 예를 들어 브랜치를 사용하면 다른 브랜치에 있는 파일에 영향을 주지 않고 새 버전이나 다른 버전의 파일에서 작업할 수 있습니다. 브랜치를 사용하여 새 기능을 개발하고 프로젝트의 특정 버전을 저장하는 등의 작업을 수행할 수 있습니다. 소스 리포지토리에는 브랜치가 하나 또는 여러 개 있을 수 있습니다. 템플릿을 사용하여 프로젝트를 만드는 경우 프로젝트용으로 만든 소스 리포지토리에는 main이라는 브랜치에 샘플 파일이 포함됩니다. 기본 브랜치는 리포지토리의 기본 브랜치입니다.

기본 브랜치

의 소스 CodeCatalyst 리포지토리에는 생성 방법에 관계없이 기본 브랜치가 있습니다. 템플릿을 사용하여 프로젝트를 만들기로 선택한 경우 해당 프로젝트용으로 만든 소스 리포지토리에는 샘플 코드, 워크플로 정의 및 기타 리소스 README 외에.md 파일이 포함됩니다. 템플릿을 사용하지 않고 소스 리포지토리를 만들면 첫 번째 README 커밋으로.md 파일이 추가되고 리포지토리를 만드는 과정에서 기본 브랜치가 자동으로 생성됩니다. 이 기본 브랜치의 이름은 main입니다. 이 기본 브랜치는 사용자가 리

포지토리를 복제할 때 로컬 리포지토리에서 기본 브랜치로 사용될 브랜치입니다. 기본 브랜치로 사용할 브랜치를 변경할 수 있습니다. 자세한 내용은 리포지토리의 기본 브랜치 관리 단원을 참조하십시오.

소스 리포지토리의 기본 브랜치는 삭제할 수 없습니다. 검색 결과에는 기본 브랜치의 결과만 포함됩니다.

커밋

커밋은 파일 또는 파일 세트에 대한 변경입니다. Amazon CodeCatalyst 콘솔에서 커밋은 변경 내용을 저장하고 소스 리포지토리에 푸시합니다. 커밋에는 변경한 사용자의 ID, 변경 시간 및 날짜, 커밋 제목, 변경에 대한 모든 메시지 등 변경 정보가 포함됩니다. 자세한 내용은 Amazon에서 커밋을 사용한 소스 코드의 변경 사항 이해 CodeCatalyst 단원을 참조하십시오.

의 소스 리포지토리 컨텍스트에서 CodeCatalyst 커밋은 리포지토리 컨텐트의 스냅샷과 해당 컨텐트에 대한 변경 내용을 보여주는 스냅샷입니다. 또한 커밋에 Git 태그를 추가하여 특정 커밋을 식별할 수 있습니다.

풀 요청

풀 리퀘스트는 여러분과 다른 사용자가 소스 리포지토리의 한 브랜치에서 다른 브랜치로의 코드 변경 내용을 검토하고, 코멘트를 달고, 병합하는 주요 방법입니다. 풀 요청을 사용하면 코드 변경 내용을 공동으로 검토하여 사소한 변경 사항이나 수정 사항, 주요 기능 추가 사항 또는 출시된 소프트웨어의 새 버전을 확인할 수 있습니다. 풀 리퀘스트에서 소스 브랜치와 대상 브랜치 간의 변경 사항 또는 해당 브랜치의 수정 버전 간 차이를 검토할 수 있습니다. 코드 변경의 개별 라인에 설명을 추가할 수 있을 뿐만 아니라 전체 풀 리퀘스트에 대한 설명을 추가할 수 있습니다.

> **Tip**
>
> 풀 리퀘스트를 생성하는 동안 표시되는 차이는 소스 브랜치의 팁과 대상 브랜치의 팁 간의 차이입니다. 풀 리퀘스트가 생성되면 선택한 풀 리퀘스트의 수정과 풀 리퀘스트를 생성할 때 대상 브랜치의 팁이었던 커밋 간의 차이가 표시됩니다. Git의 차이점과 병합 기준에 대한 자세한 내용은 Git git-merge-base설명서를 참조하십시오.

개정

개정은 풀 리퀘스트의 업데이트된 버전입니다. 풀 리퀘스트의 소스 브랜치로 푸시할 때마다 해당 푸시에 포함된 커밋의 변경 내용이 포함된 수정 버전이 생성됩니다. 원본 브랜치와 대상 브랜치 간의 차이

뿐만 아니라 풀 리퀘스트 수정 간의 차이점을 볼 수 있습니다. 자세한 내용은 Amazon에서 풀 리퀘스트를 사용하여 코드 검토하기 CodeCatalyst 단원을 참조하십시오.

워크플로

워크플로는 지속적 통합 및 지속적 전달 (CI/CD) 시스템의 일부로 코드를 빌드, 테스트 및 배포하는 방법을 설명하는 자동화된 절차입니다. 워크플로는 워크플로 실행 중에 수행할 일련의 단계 또는 조치를 정의합니다. 또한 워크플로는 워크플로를 시작하게 하는 이벤트 또는 트리거를 정의합니다. 워크플로를 설정하려면 CodeCatalyst 콘솔의 시각적 또는 YAML 편집기를 사용하여 워크플로 정의 파일을 만듭니다.

> **Tip**
>
> 프로젝트에서 워크플로를 사용하는 방법을 간단히 살펴보려면 청사진을 사용하여 프로젝트를 만들어 보세요. 각 블루프린트는 검토, 실행, 실험할 수 있는 작동하는 워크플로를 배포합니다.

또한 소스 리포지토리는 구성 파일 및 워크플로에 대한 기타 정보, 알림, 문제 및 프로젝트의 기타 구성 정보를 저장할 수 있습니다. 구성 파일은 구성 파일이 필요한 리소스를 만들거나 리포지토리를 워크플로의 소스 작업으로 지정할 때 소스 리포지토리에 작성되어 저장됩니다. 블루프린트에서 프로젝트를 생성하는 경우, 프로젝트의 일부로 소스 리포지토리에 구성 파일이 이미 저장되어 있을 것입니다. 이 구성 정보는 리포지토리의 기본 브랜치에 이름이 지정된 `.codecatalyst` 폴더에 저장됩니다. 기본 브랜치의 브랜치를 만들 때마다 해당 브랜치의 다른 모든 파일 및 폴더와 함께 이 폴더와 해당 구성의 복사본이 만들어집니다.

소스 리포지토리 작업을 위한 설정

로컬 시스템에서 CodeCatalyst Amazon의 소스 리포지토리를 사용하는 경우 Git을 단독으로 사용하거나 지원되는 통합 개발 환경 (IDE) 에서 사용하여 코드를 변경하고 코드를 푸시하고 가져올 수 있습니다. 가장 좋은 방법은 최신 버전의 Git 및 기타 소프트웨어를 사용하는 것입니다.

> **Note**
>
> 개발 환경을 사용하는 경우 Git을 설치할 필요가 없습니다. 최신 버전의 Git이 개발 환경에 포함되어 있습니다.

에 대한 버전 호환성 정보 CodeCatalyst

구성 요소	버전
Git	최신

Git 설치

Git 클라이언트 없이 소스 리포지토리의 파일, 커밋, 브랜치 및 기타 정보를 사용하려면 로컬 컴퓨터에 Git을 IDE 설치하세요.

Git 설치를 위해서는 Git 다운로드와 같은 웹 사이트를 권장합니다.

개인용 액세스 토큰 생성

소스 리포지토리를 로컬 시스템이나 원하는 IDE 시스템에 복제하려면 개인용 액세스 토큰 () PAT 을 만들어야 합니다.

개인용 액세스 토큰을 만들려면 () PAT

1. 상단 메뉴 바에서 프로필 배지를 선택한 다음 내 설정을 선택합니다.

   > **Tip**
   >
   > 프로젝트 또는 스페이스의 구성원 페이지로 이동한 다음 구성원 목록에서 이름을 선택하여 사용자 프로필을 찾을 수도 있습니다.

2. PAT이름에는 이름을 설명하는 이름을 입력합니다. PAT
3. 만료일에 기본 날짜를 그대로 두거나 달력 아이콘을 선택하여 사용자 지정 날짜를 선택합니다. 만료 날짜는 기본적으로 현재 날짜로부터 1년입니다.
4. 생성(Create)을 선택합니다.

   소스 리포지토리의 복제 리포지토리를 선택할 때도 이 토큰을 생성할 수 있습니다.
5. PAT암호를 안전한 위치에 저장합니다.

   > **Important**
   >
   > PAT비밀은 한 번만 표시됩니다. 창을 닫은 후에는 검색할 수 없습니다.

CodeCatalyst 소스 리포지토리 및 단일 페이지 애플리케이션 청사진 시작하기

이 자습서의 단계를 따라 Amazon에서 소스 리포지토리를 사용하는 방법을 알아보십시오. CodeCatalyst

CodeCatalyst Amazon에서 소스 리포지토리 작업을 시작하는 가장 빠른 방법은 템플릿을 사용하여 프로젝트를 생성하는 것입니다. 템플릿을 사용하여 프로젝트를 생성하면 샘플 코드가 포함된 소스 리포지토리를 비롯한 리소스가 자동으로 생성됩니다. 이 리포지토리와 코드 예제를 사용하여 다음 방법을 배울 수 있습니다.

- 프로젝트의 소스 리포지토리 보기 및 콘텐츠 찾아보기
- 코드 작업을 할 수 있는 새 브랜치로 개발 환경을 만드세요.
- 파일 변경, 변경 사항 커밋 및 푸시
- 풀 리퀘스트를 만들고 다른 프로젝트 멤버와 함께 코드 변경사항을 검토하세요.
- 프로젝트의 워크플로를 확인하고, 풀 요청의 소스 브랜치에서 변경 사항을 자동으로 빌드하고 테스트하십시오.
- 소스 브랜치의 변경사항을 대상 브랜치에 병합하고 풀 리퀘스트를 종료합니다.
- 병합된 변경 사항이 자동으로 빌드되고 배포되었는지 확인하세요.

이 튜토리얼을 최대한 활용하려면 풀 리퀘스트를 통해 함께 작업할 수 있도록 다른 사람들을 프로젝트에 초대하세요. 에서 이슈를 생성하여 풀 리퀘스트와 연결하거나 CodeCatalyst, 알림을 구성하고 관련 워크플로가 실행될 때 알림을 받는 등의 추가 기능을 살펴볼 수도 있습니다. 전체 탐색은 CodeCatalyst을 참조하십시오시작하기 튜토리얼.

블루프린트가 포함된 프로젝트 만들기

프로젝트를 만드는 것은 함께 작업할 수 있는 첫 번째 단계입니다. 블루프린트를 사용하여 프로젝트를 만들 수 있으며, 이렇게 하면 샘플 코드가 포함된 소스 리포지토리와 코드를 변경할 때 코드를 자동으로 빌드하고 배포하는 워크플로도 생성됩니다. 이 자습서에서는 단일 페이지 애플리케이션 블루프린트로 만든 프로젝트를 안내해 드리지만 소스 리포지토리가 있는 모든 프로젝트의 절차를 따를 수 있습니다. 프로젝트를 만들 때 IAM 역할을 선택하거나 IAM 역할이 없는 경우 역할을 추가하세요. 를 사용하는 것이 좋습니다. CodeCatalystWorkflowDevelopmentRole-*spaceName*이 프로젝트의 서비스 역할.

이미 프로젝트가 있다면 다음으로 넘어가셔도 됩니다. 프로젝트의 리포지토리 보기

> **Note**
>
> 스페이스 관리자 또는 고급 사용자 역할을 가진 사용자만 에서 CodeCatalyst 프로젝트를 만들 수 있습니다. 이 역할이 없고 이 자습서에서 작업할 프로젝트가 필요한 경우, 이러한 역할 중 하나를 가진 사람에게 프로젝트를 만들어 달라고 요청하고 생성된 프로젝트에 사용자를 추가해 달라고 요청하세요. 자세한 내용은 사용자 역할을 통한 액세스 권한 부여 단원을 참조하십시오.

블루프린트로 프로젝트를 만들려면

1. CodeCatalyst 콘솔에서 프로젝트를 만들려는 스페이스로 이동합니다.
2. 스페이스 대시보드에서 프로젝트 생성을 선택합니다.
3. '블루프린트로 시작하기'를 선택합니다.

> **Tip**
>
> Amazon Q에서 청사진을 제안하도록 하기 위한 프로젝트 요구 사항을 Amazon Q에 제공하여 청사진을 추가할 수 있습니다. 자세한 내용은 프로젝트를 생성하거나 기능을 추가할 때 Amazon Q를 사용하여 청사진을 선택합니다. 및 Amazon Q를 사용하여 프로젝트를 생성하거나 블루프린트로 기능을 추가하는 모범 사례 단원을 참조하세요. 이 기능은 미국 서부 (오레곤) 지역에서만 사용할 수 있습니다.
> 이 기능을 사용하려면 해당 공간에 제너레이티브 AI 기능을 활성화해야 합니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.

4. CodeCatalyst 블루프린트 또는 스페이스 블루프린트 탭에서 블루프린트를 선택하고 다음을 선택합니다.
5. 프로젝트 이름 아래에 프로젝트에 할당하려는 이름과 관련 리소스 이름을 입력합니다. 이름은 스페이스 내에서 고유해야 합니다.
6. (선택 사항) 기본적으로 블루프린트로 만든 소스 코드는 CodeCatalyst 리포지토리에 저장됩니다. 또는 블루프린트의 소스 코드를 타사 리포지토리에 저장하도록 선택할 수 있습니다. 자세한 내용은 확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst 단원을 참조하십시오.

> **Important**
>
> CodeCatalyst 연결된 리포지토리의 기본 브랜치 변경 감지를 지원하지 않습니다. 연결된 저장소의 기본 분기를 변경하려면 먼저 연결을 해제하고 기본 분기를 변경한 다음 다시 연

결해야 합니다. CodeCatalyst 자세한 내용은 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, GitLab Jira 프로젝트 연결 CodeCatalyst 단원을 참조하십시오. 가장 좋은 방법은 리포지토리를 연결하기 전에 항상 최신 버전의 확장 프로그램을 사용하는 것입니다.

사용하려는 타사 저장소 공급자에 따라 다음 중 하나를 수행하십시오.

- GitHub 리포지토리: 계정을 연결합니다. GitHub

  고급 드롭다운 메뉴를 선택하고 리포지토리 GitHub 공급자로 선택한 다음 블루프린트에서 만든 소스 코드를 저장할 GitHub 계정을 선택합니다.

  > **Note**
  >
  > GitHub 계정을 연결하는 경우 개인 연결을 만들어 ID와 ID 간에 ID 매핑을 설정해야 합니다. CodeCatalyst GitHub 자세한 내용은 개인 연결 및 개인적인 연결을 통해 GitHub 리소스에 접근하기 단원을 참조하세요.

- 비트버킷 리포지토리: Bitbucket 작업 공간을 연결합니다.

  고급 드롭다운 메뉴를 선택하고 Bitbucket을 리포지토리 공급자로 선택한 다음 블루프린트로 만든 소스 코드를 저장할 Bitbucket 작업 공간을 선택합니다.

- GitLab 리포지토리: 사용자를 연결합니다. GitLab

  고급 드롭다운 메뉴를 선택하고 리포지토리 GitLab 제공자로 선택한 다음 블루프린트에서 만든 소스 코드를 저장할 GitLab 사용자를 선택합니다.

7. 프로젝트 리소스에서 블루프린트 파라미터를 구성합니다. 블루프린트에 따라 소스 리포지토리 이름을 지정하는 옵션이 있을 수 있습니다.
8. (선택 사항) 선택한 프로젝트 매개 변수에 따라 업데이트된 정의 파일을 보려면 프로젝트 미리 보기 생성에서 코드 보기 또는 워크플로 보기를 선택합니다.
9. (선택 사항) 블루프린트 카드에서 세부 정보 보기를 선택하면 블루프린트의 아키텍처 개요, 필요한 연결 및 권한, 블루프린트가 생성하는 리소스 종류 등 블루프린트에 대한 구체적인 세부 정보를 볼 수 있습니다.
10. 프로젝트 만들기를 선택합니다.

프로젝트를 생성하거나 프로젝트 초대를 수락하고 로그인 프로세스를 완료하면 바로 프로젝트 개요 페이지가 열립니다. 새 프로젝트의 프로젝트 개요 페이지에는 미해결 이슈나 풀 리퀘스트가 없습니다. 선택적으로 이슈를 생성하여 자신에게 배정할 수 있습니다. 프로젝트에 다른 사람을 초대할 수도 있습니다. 자세한 내용은 에서 이슈 생성 CodeCatalyst 및 프로젝트에 사용자 초대하기 단원을 참조하세요.

프로젝트의 리포지토리 보기

프로젝트 멤버는 프로젝트의 소스 리포지토리를 볼 수 있습니다. 추가 리포지토리를 만들도록 선택할 수도 있습니다. 스페이스 관리자 역할을 가진 사용자가 리포지토리, GitHub Bitbucket 리포지토리 또는 리포지토리 확장을 설치하고 구성한 경우 확장 프로그램에 구성된 GitHub 계정, GitLab Bitbucket 작업 영역 또는 사용자에 타사 리포지토리로 연결되는 링크를 추가할 수도 있습니다. GitLab 자세한 내용은 소스 리포지토리 생성 및 빠른 시작: 확장 프로그램 설치, 공급자 연결, 리소스 연결 CodeCatalyst 단원을 참조하세요.

> **Note**
>
> 단일 페이지 애플리케이션 블루프린트로 만든 프로젝트의 경우 샘플 코드가 들어 있는 소스 리포지토리의 기본 이름은 다음과 같습니다. ***spa-app***.

프로젝트의 소스 리포지토리로 이동하려면

1. 프로젝트로 이동하여 다음 중 하나를 수행하십시오.
   - 프로젝트 요약 페이지의 목록에서 원하는 리포지토리를 선택한 다음 리포지토리 보기를 선택합니다.
   - 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다. 소스 리포지토리의 목록에서 리포지토리 이름을 선택합니다. 필터 막대에 리포지토리 이름의 일부를 입력하여 리포지토리 목록을 필터링할 수 있습니다.
2. 리포지토리 홈 페이지에서 리포지토리의 콘텐츠와 관련 리소스에 대한 정보 (예: pull request 및 워크플로 수) 를 볼 수 있습니다. 기본적으로 기본 브랜치의 콘텐츠가 표시됩니다. 드롭다운 목록에서 다른 브랜치를 선택하여 보기를 변경할 수 있습니다.

리포지토리의 개요 페이지에는 이 리포지토리 및 해당 파일의 분기에 대해 구성된 워크플로우 및 풀 리퀘스트에 대한 정보가 포함되어 있습니다. 프로젝트를 방금 생성한 경우 코드를 빌드, 테스트 및 배포하기 위한 초기 워크플로는 완료하는 데 몇 분 정도 걸리므로 계속 실행됩니다. 관련 워크플로우 아래에서 숫자를 선택하여 관련 워크플로와 해당 상태를 볼 수 있지만 이렇게 하면 CI/CD의 워크플로 페이지가 열립니다. 이 자습서에서는 개요 페이지에 머물면서 리포지토리의 코드를 살펴보세요.

README.md파일 내용은 이 페이지의 리포지토리 파일 아래에 표시됩니다. 파일에는 기본 분기의 내용이 표시됩니다. 다른 브랜치가 있는 경우 해당 컨텐츠가 표시되도록 파일 보기를 변경할 수 있습니다. .codecatalyst폴더에는 워크플로 YAML 파일과 같은 프로젝트의 다른 부분에 사용되는 코드가 들어 있습니다.

폴더 내용을 보려면 폴더 이름 옆에 있는 화살표를 선택하여 폴더를 확장합니다. 예를 들어 src 옆에 있는 화살표를 선택하면 해당 폴더에 포함된 단일 페이지 웹 응용 프로그램의 파일을 볼 수 있습니다. 파일의 콘텐츠를 보려면 목록에서 선택합니다. 그러면 파일 보기가 열리고 여러 파일의 내용을 찾아볼 수 있습니다. 콘솔에서도 단일 파일을 편집할 수 있지만, 여러 파일을 편집하려면 개발 환경을 만들어야 합니다.

Dev Environment 생성

Amazon CodeCatalyst 콘솔의 소스 리포지토리에 파일을 추가하고 변경할 수 있습니다. 하지만 여러 파일 및 브랜치를 효과적으로 사용하려면 개발 환경을 사용하거나 리포지토리를 로컬 컴퓨터에 복제하는 것이 좋습니다. 이 자습서에서는 브랜치 이름이 지정된 AWS Cloud9 개발 환경을 만들어 보겠습니다. **develop** 다른 브랜치 이름을 선택할 수 있지만 브랜치에 **develop** 이름을 지정하면 이 튜토리얼의 뒷부분에서 pull 요청을 생성할 때 코드를 빌드하고 테스트하는 워크플로가 자동으로 실행됩니다.

> **Tip**
>
> 개발 환경을 사용하는 대신 또는 추가로 로컬에서 리포지토리를 복제하려는 경우 로컬 컴퓨터에 Git이 있는지 또는 Git이 IDE 포함되어 있는지 확인하세요. 자세한 내용은 소스 리포지토리 작업을 위한 설정 단원을 참조하십시오.

새 브랜치를 사용하여 개발 환경을 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 생성하려는 스페이스로 이동합니다.
3. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택하고 소스 리포지토리를 선택한 다음 개발 환경을 만들려는 리포지토리를 선택합니다.
4. 리포지토리 홈 페이지에서 개발 환경 생성을 선택합니다.
5. 드롭다운 IDE 메뉴에서 지원되는 항목을 선택합니다. 자세한 내용은 개발 환경을 위해 지원되는 통합 개발 환경 섹션을 참조하세요.
6. 복제할 리포지토리를 선택하고, 새 브랜치에서 작업을 선택하고, 브랜치 이름 필드에 브랜치 이름을 입력하고, 다음에서 브랜치 생성 드롭다운 메뉴에서 새 브랜치를 만들 브랜치를 선택합니다.

7. 원하는 경우 개발 환경의 별칭을 추가할 수 있습니다.
8. 선택적으로 개발 환경 구성 편집 버튼을 선택하여 개발 환경의 컴퓨팅, 스토리지 또는 제한 시간 구성을 편집합니다.
9. 생성(Create)을 선택합니다. 개발 환경이 생성되는 동안 개발 환경 상태 열에 시작 중이 표시되고, 개발 환경이 생성되면 상태 열에 실행 중이 표시됩니다. 원하는 개발 환경이 포함된 새 탭이 IDE 열립니다. 코드를 편집하고 변경 사항을 커밋 및 푸시할 수 있습니다.

개발 환경을 만든 후에는 파일을 편집하고, 변경 사항을 커밋하고, 변경 사항을 **test** 브랜치에 푸시할 수 있습니다. 이 자습서에서는 `src` 폴더 내 `App.tsx` 파일에 있는 <p> 태그 사이의 내용을 편집하여 웹 페이지에 표시되는 텍스트를 변경해 보세요. 변경 사항을 커밋하고 푸시한 다음 CodeCatalyst 탭으로 돌아가십시오.

AWS Cloud9 개발 환경에서 변경 사항을 적용하고 적용하려면

1. 에서 AWS Cloud9사이드 탐색 메뉴를 확장하여 파일을 찾아보십시오. `src`펼치고 엽니다`App.tsx`.
2. <p>태그 안의 텍스트를 변경하세요.
3. 파일을 저장한 다음 Git 메뉴를 사용하여 변경 사항을 커밋하고 푸시합니다. 또는 터미널 창에서 및 git push 명령을 사용하여 변경 내용을 커밋하고 푸시할 수도 있습니다. git commit

```
git commit -am "Making an example change"
git push
```

> **Tip**
>
> Git 명령을 성공적으로 실행하려면 터미널의 디렉터리를 Git 리포지토리 디렉터리로 변경해야 할 수 있습니다.

풀 요청 생성

풀 리퀘스트를 사용하면 코드 변경 사항을 공동으로 검토하여 사소한 변경 사항이나 수정 사항, 주요 기능 추가 사항 또는 출시된 소프트웨어의 새 버전을 확인할 수 있습니다. 이 자습서에서는 풀 리퀘스트를 만들어 변경 내용을 검토해 보겠습니다.*`test`* 브랜치를 메인 브랜치와 비교합니다. 템플릿으로 만든 프로젝트에서 풀 리퀘스트를 만들면 관련 워크플로 (있는 경우) 의 실행도 시작됩니다.

풀 리퀘스트를 만들려면

1. 프로젝트로 이동합니다.
2. 다음 중 하나를 수행합니다.
   - 탐색 창에서 코드를 선택하고 풀 요청을 선택한 다음 풀 요청 생성을 선택합니다.
   - 리포지토리 홈 페이지에서 추가를 선택한 다음 풀 리퀘스트 생성을 선택합니다.
   - 프로젝트 페이지에서 풀 리퀘스트 생성을 선택합니다.
3. 소스 리포지토리에서 지정된 소스 리포지토리가 커밋된 코드를 포함하는 리포지토리인지 확인합니다. 이 옵션은 리포지토리의 기본 페이지에서 풀 리퀘스트를 만들지 않은 경우에만 나타납니다.
4. 대상 브랜치에서 코드를 검토한 후 병합할 브랜치를 선택합니다.
5. 소스 브랜치에서 커밋된 코드가 들어 있는 브랜치를 선택합니다.
6. 풀 리퀘스트 제목에 다른 사용자가 검토해야 할 내용과 이유를 이해하는 데 도움이 되는 제목을 입력합니다.
7. (선택 사항) 풀 리퀘스트 설명에 문제 링크 또는 변경 내용 설명과 같은 정보를 입력합니다.

> **Tip**
>
> Pull Request에 포함된 변경 사항에 대한 설명을 CodeCatalyst 자동으로 생성하도록 나를 위한 설명 쓰기를 선택할 수 있습니다. 자동으로 생성된 설명을 풀 리퀘스트에 추가한 후 변경할 수 있습니다.
> 이 기능을 사용하려면 스페이스에 대해 제너레이티브 AI 기능을 활성화해야 하며 연결된 리포지토리의 풀 요청에는 사용할 수 없습니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.

8. (선택 사항) 이슈에서 이슈 연결을 선택한 다음 목록에서 이슈를 선택하거나 해당 ID를 입력합니다. 이슈의 연결을 해제하려면 연결 해제 아이콘을 선택합니다.
9. (선택 사항) 필수 검토자에서 필수 검토자 추가를 선택합니다. 프로젝트 멤버 목록에서 선택하여 추가합니다. 풀 리퀘스트를 대상 브랜치에 병합하려면 먼저 필수 검토자가 변경 사항을 승인해야 합니다.

> **Note**
>
> 검토자를 필수 검토자와 선택적 검토자로 모두 추가할 수는 없습니다. 자신을 리뷰어로 추가할 수 없습니다.

10. (선택 사항) 선택적 검토자에서 선택적 검토자 추가를 선택합니다. 프로젝트 멤버 목록에서 선택하여 추가합니다. 선택적 검토자는 풀 리퀘스트를 대상 브랜치에 병합하기 전에 변경 사항을 요구 사항으로 승인할 필요가 없습니다.
11. 브랜치 간의 차이점을 검토하세요. 풀 리퀘스트에 표시되는 차이는 소스 브랜치의 수정 버전과 풀 리퀘스트가 생성된 시점의 대상 브랜치의 헤드 커밋인 병합 베이스 사이의 변경 내용입니다. 변경 사항이 표시되지 않으면 브랜치가 동일하거나 소스와 대상 모두에 대해 동일한 브랜치를 선택했을 수 있습니다.
12. 풀 리퀘스트에 검토하려는 코드와 변경 사항이 포함되어 있다고 판단되면 [Create] 를 선택합니다.

> **Note**
>
> 풀 리퀘스트를 생성한 후 코멘트를 추가할 수 있습니다. 풀 리퀘스트나 파일의 개별 라인, 전체 풀 리퀘스트에 댓글을 추가할 수 있습니다. @ 기호 뒤에 파일 이름을 붙여 파일 등의 리소스에 링크를 추가할 수 있습니다.

개요를 선택한 다음 워크플로 실행 아래의 풀 요청 세부 정보 영역에 있는 정보를 검토하면 이 풀 요청을 만들 때 시작된 관련 워크플로에 대한 정보를 볼 수 있습니다. 워크플로 실행을 보려면 실행을 선택합니다.

> **Tip**
>
> 브랜치 이름을 이외의 **`develop`** 이름으로 지정한 경우 변경 사항을 빌드하고 테스트하는 워크플로가 자동으로 실행되지 않습니다. 이를 구성하려면 onPullRequestBuildAndTest워크플로의 YAML 파일을 편집하세요. 자세한 내용은 워크플로 생성 단원을 참조하십시오.

이 풀 리퀘스트에 댓글을 달고 다른 프로젝트 멤버에게 댓글을 달아달라고 요청할 수 있습니다. 선택적 검토자 또는 필수 검토자를 추가하거나 변경할 수도 있습니다. 리포지토리의 소스 브랜치를 더 변경하도록 선택하고 커밋된 변경 사항으로 인해 풀 리퀘스트가 어떻게 수정되는지 확인할 수 있습니다. 자세한 내용은, 풀 리퀘스트 검토 풀 리퀘스트 업데이트Amazon에서 풀 리퀘스트를 사용하여 코드 검토하기 CodeCatalyst, 및 워크플로 실행 상태 및 세부 정보 보기 을 참조하십시오.

풀 리퀘스트 병합

풀 리퀘스트를 검토하고 필요한 리뷰어로부터 승인을 받으면 콘솔에서 풀 리퀘스트의 소스 브랜치를 대상 브랜치에 병합할 수 있습니다. CodeCatalyst 또한 풀 리퀘스트를 병합하면 대상 브랜치와 관련된

모든 워크플로를 통해 변경 사항이 실행되기 시작합니다. 이 자습서에서는 테스트 브랜치를 메인에 병합하여 onPushToMainDeployPipeline워크플로 실행을 시작합니다.

풀 리퀘스트를 병합하려면 (콘솔)

1. 풀 요청에서 이전 단계에서 생성한 풀 요청을 선택합니다. 풀 리퀘스트에서 병합을 선택합니다.
2. 풀 리퀘스트에 사용할 수 있는 병합 전략 중에서 선택하십시오. 선택적으로 풀 리퀘스트를 병합한 후 소스 브랜치를 삭제하는 옵션을 선택하거나 선택 취소한 다음 병합을 선택합니다. 병합이 완료되면 풀 요청의 상태가 병합됨으로 변경되고 풀 요청의 기본 보기에 더 이상 표시되지 않습니다. 기본 보기에는 Open 상태의 풀 리퀘스트가 표시됩니다. 병합된 풀 리퀘스트를 계속 볼 수는 있지만 승인하거나 상태를 변경할 수는 없습니다.

> **Note**
>
> 병합 버튼이 활성화되지 않았거나 병합할 수 없음 레이블이 표시되는 경우 필수 검토자가 아직 풀 요청을 승인하지 않았거나 콘솔에서 풀 요청을 병합할 수 없는 것입니다. CodeCatalyst 풀 리퀘스트를 승인하지 않은 검토자는 풀 리퀘스트 세부 정보 영역의 개요에 있는 시계 아이콘으로 표시됩니다. 필요한 모든 검토자가 풀 요청을 승인했지만 병합 버튼이 여전히 활성화되지 않은 경우 병합 충돌이 발생하거나 스페이스의 스토리지 할당량을 초과했을 수 있습니다. 개발 환경에서 대상 브랜치의 병합 충돌을 해결하고, 변경 사항을 푸시한 다음 풀 요청을 병합하거나, 충돌을 해결하고 로컬에서 병합한 다음 병합이 포함된 커밋을 푸시할 수 있습니다. CodeCatalyst 자세한 내용은 풀 리퀘스트 병합 (Git) 및 Git 설명서를 참조하십시오.

배포된 코드 보기

이제 기본 브랜치에 있던 원래 배포된 코드를 확인하고, 자동으로 빌드, 테스트 및 배포된 후 병합된 변경 내용을 확인할 차례입니다. 이렇게 하려면 리포지토리의 개요 페이지로 돌아가서 관련 워크플로우 아이콘 옆의 번호를 선택하거나 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택하면 됩니다.

배포된 코드를 보려면

1. 워크플로의 에서 `onPushToMainDeployPipeline` 최근 실행을 확장합니다.

> **Note**
>
> 단일 페이지 애플리케이션 블루프린트로 만든 프로젝트의 워크플로우의 기본 이름입니다.

2. 가장 최근 실행은 **main** 브랜치에 대한 풀 리퀘스트 커밋을 병합하면서 시작된 것으로, 상태가 진행 중으로 표시될 가능성이 높습니다. 목록에서 성공적으로 완료된 실행을 선택하여 해당 실행의 세부 정보를 엽니다.
3. 변수를 선택합니다. 앱의 값을 URL 복사합니다. 배포된 단일 페이지 웹 애플리케이션을 URL 위한 것입니다. 새 브라우저 탭을 열고 값을 붙여넣으면 빌드되고 배포된 코드를 볼 수 있습니다. 탭은 열어 두세요.
4. 워크플로 실행 목록으로 돌아가서 가장 최근 실행이 완료될 때까지 기다립니다. 실행되면 열었던 탭으로 돌아가서 웹 애플리케이션을 보고 브라우저를 새로 고치십시오. 병합된 풀 리퀘스트에서 변경한 내용을 확인할 수 있을 것입니다.

리소스 정리

소스 리포지토리와 풀 리퀘스트를 사용해 본 후에는 필요하지 않은 리소스를 모두 삭제하고 싶을 수도 있습니다. 풀 리퀘스트는 삭제할 수 없지만 닫을 수는 있습니다. 생성한 브랜치는 모두 삭제할 수 있습니다.

소스 리포지토리나 프로젝트가 더 이상 필요하지 않은 경우 해당 리소스를 삭제할 수도 있습니다. 자세한 내용은 소스 리포지토리 삭제 및 프로젝트 삭제 단원을 참조하세요.

프로젝트의 리포지토리에 소스 코드 저장 CodeCatalyst

소스 리포지토리는 프로젝트의 코드와 파일을 안전하게 저장하는 곳입니다. 또한 최초 커밋부터 최신 변경 사항까지의 소스 기록도 저장합니다. 소스 리포지토리가 포함된 블루프린트를 선택하면 해당 리포지토리에는 구성 파일과 프로젝트의 워크플로우 및 알림에 대한 기타 정보도 포함됩니다. 이 구성 정보는 .codealyst라는 폴더에 저장됩니다.

프로젝트 생성의 일환으로 소스 리포지토리를 만드는 블루프린트로 프로젝트를 만들거나 기존 프로젝트에 소스 리포지토리를 만드는 방법으로 소스 리포지토리를 만들 수 있습니다. CodeCatalyst 프로젝트 사용자는 프로젝트용으로 만든 리포지토리를 자동으로 보고 사용할 수 있습니다. Bibucket에서 호스팅되는 Git 리포지토리를 GitHub, 또는 프로젝트에 GitLab 연결하도록 선택할 수도 있습니다. 이렇게 하면 프로젝트 사용자가 프로젝트의 리포지토리 목록에서 연결된 리포지토리를 보고 액세스할 수 있습니다.

> **Note**
>
> 리포지토리를 연결하려면 먼저 리포지토리를 호스팅하는 서비스의 확장 프로그램을 설치해야 합니다. 보관된 리포지토리는 연결할 수 없습니다. 빈 리포지토리를 연결할 수는 있지만 기본

브랜치를 만드는 초기 커밋으로 초기화하기 CodeCatalyst 전까지는 빈 리포지토리를 사용할 수 없습니다. 자세한 내용은 스페이스에 확장 프로그램 설치 단원을 참조하십시오.

기본적으로 소스 리포지토리는 Amazon CodeCatalyst 프로젝트의 다른 구성원과 공유됩니다. 프로젝트에 대한 추가 소스 리포지토리를 생성하거나 리포지토리를 프로젝트에 연결할 수 있습니다. 프로젝트의 모든 구성원은 프로젝트의 소스 리포지토리에 있는 파일 및 폴더를 보고, 추가하고, 편집하고, 삭제할 수 있습니다.

소스 리포지토리의 코드를 빠르게 작업하려면 지정된 리포지토리를 복제하고 해당 리포지토리로 브랜치하는 개발 환경을 만들어 개발 환경에 맞게 선택한 통합 개발 환경 (IDE) 에서 코드 작업을 수행할 수 있습니다. 로컬 컴퓨터의 소스 리포지토리를 복제하고 로컬 리포지토리와 원격 리포지토리 사이의 변경 내용을 가져와 푸시할 수 있습니다. CodeCatalyst 자격 증명 관리를 IDE 지원하는 한 원하는 IDE 대로 소스 리포지토리에 대한 액세스를 구성하여 소스 리포지토리를 사용할 수도 있습니다.

저장소 이름은 프로젝트 내에서 고유해야 합니다. CodeCatalyst

주제

소스 리포지토리 생성

CodeCatalystAmazon에서 블루프린트를 사용하여 프로젝트를 생성하면 소스 리포지토리가 자동으로 CodeCatalyst 생성됩니다. 해당 소스 리포지토리에는 샘플 코드뿐 아니라 사용자를 위해 만든 워크플로우 및 기타 리소스에 대한 구성 정보가 들어 있습니다. 에서 리포지토리를 시작하는 데 권장되는 방법입니다. CodeCatalyst 프로젝트용 리포지토리를 만들도록 선택할 수 있습니다. 이러한 리포지토리에는 언제든지 편집하거나 삭제할 수 README있는.md 파일이 포함됩니다. 소스 리포지토리를 생성할 때 선택한 옵션에 따라 리포지토리에 파일이 포함될 수도 있습니다. `.gitignore`

기존 Git 리포지토리를 CodeCatalyst 소스 리포지토리로 복제하려면 대신 빈 리포지토리를 만드는 것이 좋습니다. 이 리포지토리는 콘텐츠를 추가할 CodeCatalyst 때까지 에서 사용할 수 없으며, 몇 가지

간단한 Git 명령으로 이 작업을 수행할 수 있습니다. 또는 콘솔에서 직접 빈 리포지토리에 콘텐츠를 추가할 수도 있습니다. CodeCatalyst 또는 지원되는 Git 리포지토리 공급자에 소스 리포지토리를 연결할 수 있습니다. 자세한 내용은 소스 리포지토리 연결 단원을 참조하십시오.

소스 리포지토리를 생성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다.
3. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
4. 리포지토리 추가를 선택하고 리포지토리 생성을 선택합니다.
5. 리포지토리 이름에 리포지토리 이름을 제공합니다. 이 안내서에서는 다음을 사용합니다.*codecatalyst-source-repository*하지만 다른 이름을 선택할 수 있습니다. 리포지토리 이름은 프로젝트에서 고유해야 합니다. 리포지토리 이름 요구 사항에 대한 자세한 내용은 을 참조하십시오의 소스 리포지토리 할당량 CodeCatalyst.
6. (선택 사항) 설명에 리포지토리의 용도를 프로젝트의 다른 사용자가 이해하는 데 도움이 되도록 리포지토리에 대한 설명을 추가합니다.
7. 리포지토리 만들기 (기본값) 를 선택합니다. 이 옵션은 기본 README 브랜치와.md 파일이 포함된 리포지토리를 만듭니다. 빈 리포지토리와 달리 이 리포지토리는 생성되자마자 사용할 수 있습니다.
8. Default 브랜치에서는 다른 이름을 선택할 이유가 없는 한 이름을 main으로 그대로 두십시오. 이 가이드의 예시에서는 모두 기본 브랜치에 main이라는 이름을 사용합니다.
9. (선택 사항) 푸시하려는 코드 유형에 맞는 `.gitignore` 파일을 추가합니다.
10. 생성(Create)을 선택합니다.

> **Note**
>
> CodeCatalyst `README.md`파일을 만들 때 저장소에 파일을 추가합니다. CodeCatalyst또한 main이라는 기본 브랜치에 리포지토리에 대한 초기 커밋을 생성합니다. README.md 파일을 편집하거나 삭제할 수 있지만 기본 브랜치는 삭제할 수 없습니다.

빈 소스 리포지토리를 만들려면

1. CodeCatalyst 콘솔에서 빈 리포지토리를 만들려는 프로젝트로 이동합니다.

2. 프로젝트 요약 페이지의 소스 리포지토리에서 리포지토리 추가를 선택한 다음 리포지토리 만들기를 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다. 리포지토리 추가를 선택하고 리포지토리 생성을 선택합니다.
3. 리포지토리 이름에 리포지토리 이름을 제공합니다. 이 가이드에서는 다음을 사용합니다.*codecatalyst-source-repository*하지만 다른 이름을 선택할 수 있습니다. 리포지토리 이름은 프로젝트에서 고유해야 합니다. 리포지토리 이름 요구 사항에 대한 자세한 내용은 을 참조하십시오의 소스 리포지토리 할당량 CodeCatalyst.
4. (선택 사항) 설명에 리포지토리의 용도를 프로젝트의 다른 사용자가 이해하는 데 도움이 되도록 리포지토리에 대한 설명을 추가합니다.
5. [빈 리포지토리 만들기] 를 선택한 다음 [Create] 를 선택합니다.

기존 Git 리포지토리를 소스 리포지토리로 복제

기존 Git 리포지토리를 Amazon의 빈 소스 리포지토리로 복제할 수 있습니다. CodeCatalyst 이는 이전에 다른 Git 리포지토리 CodeCatalyst 공급자에서 호스팅된 코드를 빠르게 시작할 수 있는 방법입니다. 미러 복제본을 만든 다음 미러를 푸시하여 리포지토리의 내용을 복제할 수 있습니다. CodeCatalyst 또는 콘텐츠를 추가하려는 리포지토리의 로컬 리포지토리가 있는 경우 소스 리포지토리를 로컬 리포지토리에 다른 원격 리포지토리로 추가한 다음 빈 CodeCatalyst 소스 리포지토리로 푸시할 수 있습니다. CodeCatalyst 두 접근 방식 모두 똑같이 유효합니다. 미러 클론을 사용하면 브랜치를 매핑할 뿐만 아니라 모든 참조를 매핑할 수 있습니다. 에서 CodeCatalyst 리포지토리의 작업 복사본을 만드는 간단하고 깔끔한 방법입니다. 빈 CodeCatalyst 소스 리포지토리를 가리키는 로컬 리포지토리에 리모컨을 추가하면 리포지토리 내용이 추가되지만 로컬 리포지토리에서 CodeCatalyst 소스 리포지토리와 원본 Git 원격 리포지토리로 푸시할 수도 있습니다. CodeCatalyst 이는 코드를 다른 원격 리포지토리에 유지하려는 경우에 유용할 수 있지만, 다른 개발자가 원격 리포지토리 중 하나에만 코드를 커밋하면 충돌이 발생할 수 있습니다.

다음 절차에서는 기본 Git 명령을 사용하여 이 작업을 수행합니다. 복제를 포함하여 Git에서 여러 가지 방법으로 작업을 수행할 수 있습니다. 자세한 내용은 Git 설명서를 참조하세요.

> **Important**
>
> 콘텐츠를 복제하려면 CodeCatalyst 먼저 에서 빈 리포지토리를 만들어야 합니다. 또한 개인용 액세스 토큰이 있어야 합니다. 자세한 내용은 빈 소스 리포지토리를 만들려면 및 개인용 액세스 토큰 생성 단원을 참조하세요.

기존 Git **`git clone --mirror`** 리포지토리를 복제하는 데 사용하려면 CodeCatalyst

1. CodeCatalyst 콘솔에서 빈 리포지토리를 만든 프로젝트로 이동합니다.
2. 프로젝트 요약 페이지의 목록에서 빈 저장소를 선택한 다음 저장소 보기를 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다. 프로젝트의 소스 리포지토리 목록에서 빈 리포지토리의 이름을 선택합니다.
3. 빈 URL 저장소의 HTTPS 복제본을 복사합니다. 미러 클론을 푸시하려면 이 파일이 필요합니다. 예를 들어 MyExampleRepo ExampleCorp 스페이스에 있는 MyExampleProject 프로젝트의 소스 리포지토리 이름을 지정하고 사용자 이름이 인 경우 클론은 다음과 같이 URL 보일 수 있습니다. LiJuan

   ```
   https://LiJuan@git.us-west-2.codecatalyst.aws/
   v1/ExampleCorp/MyExampleProject/MyExampleRepo
   ```

4. 명령줄 또는 터미널 창에서 `git clone --mirror` 명령을 사용하여 복제하려는 Git 저장소의 미러 복제본을 생성합니다. CodeCatalyst 예를 들어 GitHub, 에서 codecatalyst-blueprints 리포지토리의 미러 클론을 생성하려면 다음 명령을 입력합니다.

   ```
   git clone --mirror https://github.com/aws/codecatalyst-blueprints.git
   ```

5. 디렉터리를 복제를 생성한 디렉터리로 변경합니다.

   ```
   cd codecatalyst-blueprints.git
   ```

6. git push명령을 실행하여 대상 CodeCatalyst 소스 리포지토리의 이름 URL 및 이름과 옵션을 지정합니다. --all (이 내용은 3단계에서 URL 복사한 것입니다.) 예:

   ```
   git push https://LiJuan@git.us-west-2.codecatalyst.aws/
   v1/ExampleCorp/MyExampleProject/MyExampleRepo --all
   ```

리모컨을 추가하고 로컬 리포지토리를 푸시하려면 CodeCatalyst

1. CodeCatalyst 콘솔에서 빈 리포지토리를 만든 프로젝트로 이동합니다.
2. 프로젝트 요약 페이지의 목록에서 빈 저장소를 선택한 다음 저장소 보기를 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다. 프로젝트의 소스 리포지토리 목록에서 빈 리포지토리의 이름을 선택합니다.

3. 빈 URL 저장소의 HTTPS 복제본을 복사합니다. 미러 클론을 푸시하려면 이 파일이 필요합니다. 예를 들어 MyExampleRepo ExampleCorp 스페이스에 있는 MyExampleProject 프로젝트의 소스 리포지토리 이름을 지정하고 사용자 이름이 인 경우 클론은 다음과 같이 URL 보일 수 있습니다. LiJuan

```
https://LiJuan@git.us-west-2.codecatalyst.aws/
v1/ExampleCorp/MyExampleProject/MyExampleRepo
```

4. 명령줄 또는 터미널 창에서 디렉터리를 푸시하려는 로컬 리포지토리로 변경합니다. CodeCatalyst

5. git remote -v 명령어를 실행하여 로컬 리포지토리의 기존 리모트를 확인합니다. 예를 들어 미국 동부 (오하이오) 지역에 있는 리포지토리의 로컬 AWS CodeCommit 리포지토리를 복제하는 경우 명령 출력은 다음과 같을 수 있습니다. **MyDemoRepo**

```
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (push)
```

   리포지토리를 계속 URL 사용하려면 리모컨을 복사하세요.

6. 다음 `git remote remove` 명령을 사용하여 URLs 가져오기용 CodeCommit 리포지토리를 제거하고 오리진용으로 푸시할 수 있습니다.

```
git remote remove origin
```

7. git remote add 명령을 사용하여 CodeCatalyst 소스 리포지토리를 로컬 리포지토리의 페치 및 푸시 URL 리모컨으로 추가합니다. 예:

```
git remote add origin https://LiJuan@git.us-west-2.codecatalyst.aws/
v1/ExampleCorp/MyExampleProject/MyExampleRepo
```

   이렇게 하면 CodeCommit 리포지토리 푸시가 CodeCatalyst 소스 URL 리포지토리로 URL 대체되지만 페치는 변경되지 않습니다. URL 따라서 git remote -v 명령을 다시 실행하면 이제 원격 리포지토리에서 코드를 가져오는 것을 볼 수 있지만 로컬 리포지토리에서 소스 리포지토리로 변경 사항을 푸시하도록 구성되었습니다. CodeCommit CodeCatalyst

```
origin  https://git-codecommit.us-east-2.amazonaws.com/v1/repos/MyDemoRepo (fetch)
origin  https://LiJuan@git.us-west-2.codecatalyst.aws/v1/ExampleCorp/
MyExampleProject/MyExampleRepo (push)
```

다음 명령을 사용하여 두 리포지토리에 모두 URL 푸시하려면 선택적으로 CodeCommit 원격 리포지토리를 다시 추가할 수 있습니다. `git remote set-url`

```
git remote set-url --add --push origin https://git-codecommit.us-
east-2.amazonaws.com/v1/repos/MyDemoRepo
```

8. `git push`명령을 실행하여 로컬 리포지토리를 구성된 모든 푸시 리모트에 푸시합니다. 또는 git push -u -origin 명령을 실행하여 로컬 리포지토리를 두 리포지토리에 푸시하는 --all 옵션을 지정합니다. 예:

```
git push -u -origin --all
```

> **Tip**
>
> Git 버전에 따라 --all이 로컬 저장소의 모든 브랜치를 빈 저장소로 푸시하는 데 작동하지 않을 수 있습니다. 각 브랜치를 체크아웃하고 별도로 푸시해야 할 수도 있습니다.

소스 리포지토리 연결

소스 리포지토리를 프로젝트에 연결할 때 해당 리포지토리를 호스팅하는 서비스용 CodeCatalyst 확장이 있는 리포지토리를 포함할 수 있습니다 (스페이스에 해당 확장이 설치된 경우). 스페이스 관리자 역할을 가진 사용자만 확장을 설치할 수 있습니다. 확장이 설치되면 해당 확장에서 액세스할 수 있도록 구성된 저장소에 연결할 수 있습니다. 자세한 내용은 스페이스에 확장 프로그램 설치 또는 팔로우를 참조하십시오. GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, GitLab Jira 프로젝트 연결 CodeCatalyst

> **Important**
>
> 리포지토리 확장을 설치한 후에는 연결하는 CodeCatalyst 모든 리포지토리의 코드가 인덱싱되고 저장됩니다. CodeCatalyst 이렇게 하면 코드를 검색할 수 있게 됩니다. CodeCatalyst에서 CodeCatalyst 연결된 리포지토리를 사용할 때의 코드 데이터 보호를 더 잘 이해하려면 Amazon CodeCatalyst User Guide의 데이터 보호를 참조하십시오.

스페이스의 한 프로젝트에만 리포지토리를 연결할 수 있습니다. 보관된 저장소는 링크할 수 없습니다. 빈 리포지토리를 연결할 수는 있지만 기본 브랜치를 만드는 초기 커밋으로 초기화하기 CodeCatalyst 전까지는 빈 리포지토리를 사용할 수 없습니다. 뿐만 아니라

- GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리는 스페이스의 한 CodeCatalyst 프로젝트에만 연결할 수 있습니다.
- 비어 있거나 보관된 GitHub 리포지토리, Bitbucket 리포지토리 또는 프로젝트 리포지토리는 프로젝트와 함께 사용할 수 없습니다. GitLab CodeCatalyst
- 프로젝트의 리포지토리와 이름이 같은 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리는 연결할 수 없습니다. CodeCatalyst
- GitHub 리포지토리 확장은 GitHub 엔터프라이즈 서버 리포지토리와 호환되지 않습니다.
- Bitbucket 리포지토리 확장은 Bitbucket 데이터 센터 리포지토리와 호환되지 않습니다.
- 리포지토리 확장은 자체 관리형 GitLab 프로젝트 리포지토리와 호환되지 않습니다. GitLab
- 연결된 리포지토리에서는 나를 위한 설명 쓰기 또는 댓글 요약 기능을 사용할 수 없습니다. 이러한 기능은 풀 리퀘스트 인에서만 사용할 수 있습니다. CodeCatalyst

기여자로서 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 연결할 수 있지만, 타사 리포지토리는 스페이스 관리자 또는 프로젝트 관리자로만 연결을 해제할 수 있습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

> **Important**
>
> CodeCatalyst 연결된 리포지토리의 기본 브랜치에서 변경 사항 감지를 지원하지 않습니다. 연결된 저장소의 기본 분기를 변경하려면 먼저 연결을 해제하고 기본 분기를 변경한 다음 다시 연결해야 합니다. CodeCatalyst 자세한 내용은 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, GitLab Jira 프로젝트 연결 CodeCatalyst 단원을 참조하십시오.
> 가장 좋은 방법은 리포지토리를 연결하기 전에 항상 최신 버전의 확장 프로그램을 사용하는 것입니다.

소스 리포지토리를 연결하려면

1. 리포지토리를 연결하려는 프로젝트로 이동합니다.

> **Note**
>
> 저장소를 연결하려면 먼저 스페이스 관리자 역할을 가진 사용자가 저장소를 호스팅하는 공급자의 확장 프로그램을 설치해야 합니다. 자세한 내용은 스페이스에 확장 프로그램 설치 단원을 참조하십시오.

2. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
3. 리포지토리 추가를 선택한 다음 리포지토리 연결을 선택합니다.
4. 리포지토리 제공자 드롭다운 메뉴에서 다음 타사 리포지토리 제공자 중 하나 (GitHub또는 Bitbucket) 를 선택합니다.
5. 연결하기로 선택한 타사 저장소 공급자에 따라 다음 중 하나를 수행하십시오.

   - GitHub 리포지토리: 리포지토리를 연결합니다. GitHub

     1. GitHub 계정 드롭다운 메뉴에서 연결하려는 저장소가 들어 있는 GitHub 계정을 선택합니다.
     2. GitHub 리포지토리 드롭다운 메뉴에서 프로젝트를 연결하려는 GitHub 계정을 선택합니다. CodeCatalyst
     3. (선택 사항) 리포지토리 목록에 GitHub 리포지토리가 없다면 Amazon CodeCatalyst 애플리케이션에서 리포지토리에 액세스할 수 있도록 구성되지 않은 것일 수 있습니다. GitHub 연결된 CodeCatalyst 계정에서 사용할 수 있는 GitHub 리포지토리를 구성할 수 있습니다.

        a. GitHub계정으로 이동하여 설정을 선택한 다음 애플리케이션을 선택합니다.

        b. 설치된 GitHub 앱 탭에서 Amazon CodeCatalyst 애플리케이션에 맞게 구성을 선택합니다.

        c. 다음 중 하나를 수행하여 연결하려는 GitHub 리포지토리에 대한 액세스를 구성하십시오. CodeCatalyst

           - 현재 및 미래의 모든 리포지토리에 대한 액세스를 제공하려면 모든 리포지토리를 선택합니다.
           - 특정 리포지토리에 대한 액세스를 제공하려면 리포지토리만 선택을 선택하고 리포지토리 선택 드롭다운을 선택한 다음 연결을 허용할 리포지토리를 선택합니다. CodeCatalyst

   - Bitbucket 리포지토리: Bitbucket 리포지토리를 연결합니다.

1. Bitbucket 작업 영역 드롭다운 메뉴에서 연결하려는 리포지토리가 포함된 Bitbucket 작업 영역을 선택합니다.
2. Bitbucket 리포지토리 드롭다운 메뉴에서 프로젝트를 연결하려는 Bitbucket 리포지토리를 선택합니다. CodeCatalyst

> **Tip**
>
> 리포지토리 이름이 회색으로 표시된 경우 해당 리포지토리가 이미 CodeCatalyst Amazon의 다른 프로젝트에 연결되어 있기 때문에 해당 리포지토리를 연결할 수 없습니다.

6. 연결을 선택합니다.

에서 CodeCatalyst 리포지토리, Bitbucket 리포지토리 또는 프로젝트 GitHub 리포지토리를 더 이상 사용하지 않으려면 프로젝트에서 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 연결 해제할 수 있습니다. CodeCatalyst 리포지토리의 연결이 해제되면 해당 리포지토리의 이벤트가 워크플로우 실행을 시작하지 않으므로 해당 리포지토리를 Dev Environments와 함께 CodeCatalyst 사용할 수 없습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

소스 리포지토리 보기

Amazon에서 프로젝트와 관련된 소스 리포지토리를 볼 수 있습니다. CodeCatalyst 의 CodeCatalyst 소스 리포지토리의 경우 리포지토리의 개요 페이지는 다음을 포함하여 해당 리포지토리의 정보 및 활동에 대한 간략한 개요를 제공합니다.

- 리포지토리에 대한 설명 (있는 경우)
- 리포지토리 내 브랜치 수
- 리포지토리에 대해 열려 있는 풀 리퀘스트 수
- 리포지토리의 관련 워크플로 수
- 기본 브랜치 또는 선택한 브랜치에 있는 파일 및 폴더
- 표시된 브랜치에 마지막으로 커밋한 제목, 작성자, 날짜
- 마크다운에서 README 렌더링된.md 파일의 내용 (README.md 파일이 포함된 경우)

또한 이 페이지는 리포지토리의 커밋, 브랜치 및 풀 요청에 대한 링크와 개별 파일을 열고, 보고, 편집할 수 있는 빠른 방법을 제공합니다.

> **Note**
>
> 콘솔에서는 연결된 리포지토리에 대한 이 정보를 볼 수 없습니다. CodeCatalyst 연결된 리포지토리에 대한 정보를 보려면 리포지토리 목록에서 링크를 선택하여 해당 리포지토리를 호스팅하는 서비스에서 해당 리포지토리를 여십시오.

프로젝트의 소스 리포지토리로 이동하려면

1. 프로젝트로 이동하여 다음 중 하나를 수행하십시오.
   - 프로젝트 요약 페이지의 목록에서 원하는 리포지토리를 선택한 다음 리포지토리 보기를 선택합니다.
   - 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다. 소스 리포지토리의 목록에서 리포지토리 이름을 선택합니다. 필터 막대에 리포지토리 이름의 일부를 입력하여 리포지토리 목록을 필터링할 수 있습니다.
2. 리포지토리 홈 페이지에서 리포지토리의 콘텐츠와 관련 리소스에 대한 정보 (예: pull request 및 워크플로 수) 를 볼 수 있습니다. 기본적으로 기본 브랜치의 콘텐츠가 표시됩니다. 드롭다운 목록에서 다른 브랜치를 선택하여 보기를 변경할 수 있습니다.

> **Tip**
>
> 프로젝트 요약 페이지에서 프로젝트 코드 보기를 선택하여 프로젝트의 리포지토리로 빠르게 이동할 수도 있습니다.

소스 리포지토리의 설정 편집

리포지토리 설명 편집, 기본 브랜치 선택, 브랜치 규칙 생성 및 관리, 풀 리퀘스트에 대한 승인 규칙 생성 및 관리 등 리포지토리 설정을 관리할 수 CodeCatalyst 있습니다. 이를 통해 프로젝트 구성원은 리포지토리가 어디에 사용되는지 이해하고 팀에서 사용하는 모범 사례와 프로세스를 적용할 수 있습니다.

> **Note**
>
> 소스 리포지토리의 이름은 편집할 수 없습니다.
> 에서는 링크된 저장소의 이름, 설명 또는 기타 정보를 편집할 수 없습니다 CodeCatalyst. 링크된 리포지토리에 대한 정보를 수정하려면 링크된 리포지토리를 호스팅하는 공급자에서 해당 정보를 편집해야 합니다. 자세한 내용은 링크된 리포지토리를 호스팅하는 서비스 설명서를 참조하십시오.

리포지토리의 설정을 편집하려면

1. CodeCatalyst 콘솔에서 설정을 편집하려는 소스 리포지토리가 포함된 프로젝트로 이동합니다.
2. 프로젝트 요약 페이지의 목록에서 원하는 리포지토리를 선택한 다음 리포지토리 보기를 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다.
3. 리포지토리의 개요 페이지에서 기타를 선택한 다음 설정 관리를 선택합니다.
4. 다음 중 한 개 이상을 수행할 수 있습니다.
   - 리포지토리의 설명을 편집한 다음 저장을 선택합니다.
   - 리포지토리의 기본 브랜치를 변경하려면 기본 브랜치에서 편집을 선택합니다. 자세한 내용은 리포지토리의 기본 브랜치 관리 단원을 참조하십시오.
   - 브랜치에서 특정 작업을 수행할 권한이 있는 프로젝트 역할에 대한 규칙을 추가, 제거 또는 변경하려면 브랜치 규칙에서 편집을 선택합니다. 자세한 내용은 브랜치 규칙을 사용하여 브랜치에 허용된 작업 관리 단원을 참조하십시오.
   - 브랜치에 풀 리퀘스트를 병합하기 위한 승인 규칙을 추가, 제거 또는 변경하려면 승인 규칙에서 편집을 선택합니다. 자세한 내용은 풀 리퀘스트를 승인 규칙과 병합하기 위한 요구 사항 관리 단원을 참조하십시오.

소스 리포지토리 복제

소스 리포지토리의 여러 파일, 브랜치 및 커밋으로 효과적으로 작업하려면 소스 리포지토리를 로컬 컴퓨터에 복제하고 Git 클라이언트 또는 통합 개발 환경 () IDE 을 사용하여 변경하십시오. 이슈 및 풀 리퀘스트와 같은 CodeCatalyst 기능을 사용하려면 변경 내용을 소스 리포지토리에 커밋하고 푸시하세요. 코드 작업을 위한 개발 환경을 만들도록 선택할 수도 있습니다. 개발 환경을 만들면 지정한 리포지토리와 브랜치가 개발 환경에 자동으로 복제됩니다.

> **Note**
>
> CodeCatalyst 콘솔에서 연결된 리포지토리를 복제하거나 해당 리포지토리에 대한 개발 환경을 생성할 수 없습니다. 연결된 리포지토리를 로컬에서 복제하려면 리포지토리 목록에서 링크를 선택하여 해당 리포지토리를 호스팅하는 서비스에서 해당 리포지토리를 연 다음 복제합니다. 자세한 내용은 연결된 리포지토리를 호스팅하는 서비스 설명서를 참조하십시오.

소스 리포지토리에서 개발 환경을 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
3. 코드 작업을 하려는 소스 리포지토리를 선택합니다.
4. 개발 환경 생성을 선택합니다.
5. 드롭다운 IDE 메뉴에서 지원되는 항목을 선택합니다. 자세한 내용은 개발 환경을 위해 지원되는 통합 개발 환경 섹션을 참조하세요.
6. 다음 중 하나를 수행합니다.
   - 기존 브랜치에서 작업을 선택한 다음 기존 브랜치 드롭다운 메뉴에서 브랜치를 선택합니다.
   - 새 브랜치에서 근무를 선택하고 브랜치 이름 필드에 브랜치 이름을 입력한 다음 드롭다운 메뉴에서 브랜치 생성 기반으로 새 브랜치를 만들 브랜치를 선택합니다.
7. 선택적으로 개발 환경의 이름을 추가하거나 구성을 편집할 수 있습니다.
8. 생성(Create)을 선택합니다.

소스 리포지토리를 복제하려면

1. 프로젝트로 이동합니다.
2. 프로젝트 요약 페이지의 목록에서 원하는 리포지토리를 선택한 다음 리포지토리 보기를 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 필터 막대에 리포지토리 이름의 일부를 입력하여 리포지토리 목록을 필터링할 수 있습니다.
3.
4. 리포지토리 복제를 선택합니다. 리포지토리의 URL 복제본을 복사합니다.

> **Note**
>
> 개인용 액세스 토큰 (PAT) 이 없는 경우 토큰 생성을 선택합니다. 토큰을 복사하여 안전한 위치에 저장합니다. Git 클라이언트 또는 통합 개발 환경 () IDE 에서 비밀번호를 입력하라는 메시지가 PAT 표시되면 이 옵션을 사용합니다.

5. 다음 중 하나를 수행합니다.

   - 리포지토리를 로컬 컴퓨터에 복제하려면 터미널 또는 명령줄을 열고 git clone 명령 URL 뒤에 복제본을 포함하여 명령을 실행합니다. 예:

     ```
     git clone https://LiJuan@git.us-west-2.codecatalyst.aws/
     v1/ExampleCorp/MyExampleProject/MyExampleRepo
     ```

     암호를 입력하라는 메시지가 표시되면 이전에 저장한 PAT 암호를 붙여넣습니다.

     > **Note**
     >
     > 운영 체제에서 자격 증명 관리를 제공하거나 자격 증명 관리 시스템을 설치한 경우 자격 증명 관리 시스템을 한 번만 제공하면 됩니다PAT. 그렇지 않은 경우 모든 Git PAT 작업에 대해 를 제공해야 할 수 있습니다. 가장 좋은 방법은 자격 증명 관리 시스템에 사용자 인증 정보가 안전하게 저장되도록 하는 것입니다. PAT 를 복제 URL 문자열의 PAT 일부로 포함하지 마십시오.

   - 를 사용하여 리포지토리를 복제하려면 해당 설명서를 따르십시오IDE. IDE Git 리포지토리를 복제하는 옵션을 선택하고 를 제공합니다. URL 암호를 입력하라는 메시지가 표시되면 암호를 입력합니다. PAT

소스 리포지토리 삭제

Amazon CodeCatalyst 프로젝트의 소스 리포지토리가 더 이상 필요하지 않은 경우 삭제할 수 있습니다. 소스 리포지토리를 삭제하면 리포지토리에 저장된 모든 프로젝트 정보도 삭제됩니다. 소스 리포지토리에 종속된 워크플로가 있는 경우 리포지토리를 삭제한 후 해당 워크플로가 프로젝트 워크플로 목록에서 삭제됩니다. 원본 리포지토리를 참조하는 이슈는 삭제되거나 변경되지 않지만, 리포지토리를 삭제하면 이슈에 추가된 소스 리포지토리 링크는 삭제되지 않습니다.

Important

소스 리포지토리 삭제는 취소할 수 없습니다. 소스 리포지토리를 삭제한 후에는 더 이상 소스 리포지토리를 복제하거나 데이터를 가져오거나 데이터를 푸시할 수 없습니다. 소스 리포지토리를 삭제해도 해당 리포지토리 (로컬 리포지토리) 의 로컬 사본은 삭제되지 않습니다. 로컬 리포지토리를 삭제하려면 로컬 컴퓨터의 디렉터리 및 파일 관리 도구를 사용하세요.

Note

CodeCatalyst 콘솔에서는 링크된 리포지토리를 삭제할 수 없습니다. 연결된 리포지토리를 삭제하려면 리포지토리 목록에서 링크를 선택하여 해당 리포지토리를 호스팅하는 서비스에서 해당 리포지토리를 연 다음 삭제하십시오. 자세한 내용은 링크된 리포지토리를 호스팅하는 서비스 설명서를 참조하십시오.
프로젝트에서 연결된 저장소를 제거하려면 을 참조하십시오에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst.

소스 리포지토리를 삭제하려면

1. 삭제하려는 소스 리포지토리가 포함된 프로젝트로 이동합니다.
2. 프로젝트 요약 페이지의 목록에서 원하는 리포지토리를 선택한 다음 리포지토리 보기를 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다.
3. 리포지토리 홈 페이지에서 추가를 선택한 다음 리포지토리 삭제를 선택합니다.
4. 브랜치, 풀 리퀘스트 및 관련 워크플로 정보를 검토하여 아직 사용 중이거나 작업이 완료되지 않은 리포지토리를 삭제하지 않도록 하십시오. 계속하려면 삭제를 입력한 다음 삭제를 선택합니다.

Amazon에서 브랜치를 사용하여 소스 코드 작업을 정리하세요. CodeCatalyst

Git에서 브랜치는 커밋에 대한 포인터 또는 참조입니다. 개발 시 작업을 체계적으로 정리할 수 있는 편리한 수단입니다. 브랜치를 사용하면 다른 브랜치의 작업에 영향을 주지 않으면서 파일의 새 버전이나 다른 버전에 대한 작업을 분리할 수 있습니다. 브랜치를 사용하여 새 기능을 개발하고 프로젝트의 특정 버전을 저장하는 등의 작업을 수행할 수 있습니다. 소스 리포지토리의 브랜치에 대한 규칙을 구성하여 브랜치의 특정 작업을 해당 프로젝트의 특정 역할로 제한할 수 있습니다.

Amazon의 소스 CodeCatalyst 리포지토리에는 생성 방법에 관계없이 콘텐츠와 기본 브랜치가 있습니다. 연결된 리포지토리에는 기본 브랜치나 콘텐츠가 없을 수 있지만, 이를 초기화하고 기본 브랜치를 생성하기 CodeCatalyst 전까지는 사용할 수 없습니다. 블루프린트를 사용하여 프로젝트를 생성할 때 README 는.md 파일, 샘플 코드, 워크플로 정의 및 기타 리소스가 포함된 해당 프로젝트의 소스 리포지토리를 CodeCatalyst 생성합니다. 블루프린트를 사용하지 않고 소스 리포지토리를 만들면 첫 번째 README 커밋으로.md 파일이 추가되고 기본 브랜치가 자동으로 생성됩니다. 이 기본 브랜치의 이름은 main입니다. 이 기본 브랜치는 사용자가 리포지토리를 복제할 때 로컬 리포지토리에서 기본 브랜치로 사용될 브랜치입니다.

> **Note**
>
> 기본 브랜치는 삭제할 수 없습니다. 소스 리포지토리용으로 만든 첫 번째 브랜치가 해당 리포지토리의 기본 브랜치입니다. 또한 검색에는 기본 브랜치의 결과만 표시됩니다. 다른 브랜치에서는 코드를 검색할 수 없습니다.

에서 리포지토리를 만들면 첫 번째 CodeCatalyst 커밋도 생성되는데, 이 README 커밋에서는.md 파일이 포함된 기본 브랜치가 만들어집니다. 기본 브랜치의 이름은 main입니다. 이 이름이 본 안내서의 예제에 사용된 기본 브랜치 이름입니다.

주제

- 브랜치 생성
- 리포지토리의 기본 브랜치 관리
- 브랜치 규칙을 사용하여 브랜치에 허용된 작업 관리
- 브랜치를 위한 Git 명령어
- 지점 및 세부 정보 보기
- 브랜치 삭제

브랜치 생성

CodeCatalyst 콘솔을 사용하여 CodeCatalyst 리포지토리에 브랜치를 만들 수 있습니다. 생성한 브랜치는 다음 번에 다른 사용자가 리포지토리에서 변경 내용을 가져올 때 볼 수 있습니다.

> **Tip**
>
> 코드 작업을 위한 개발 환경을 만드는 과정에서 브랜치를 만들 수도 있습니다. 자세한 내용은 Dev Environment 생성 단원을 참조하십시오.

Git을 사용하여 브랜치를 만들 수도 있습니다. 자세한 내용은 브랜치를 위한 일반적인 Git 명령 단원을 참조하십시오.

브랜치를 만들려면 (콘솔)

1. CodeCatalyst 콘솔에서 소스 리포지토리가 있는 프로젝트로 이동합니다.
2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
3. 브랜치를 만들려는 리포지토리를 선택합니다.
4. 리포지토리의 개요 페이지에서 추가를 선택한 다음 브랜치 생성을 선택합니다.
5. 브랜치 이름을 입력합니다.
6. 브랜치를 만들 브랜치를 선택한 다음 [Create] 를 선택합니다.

리포지토리의 기본 브랜치 관리

Amazon의 소스 리포지토리에서 기본 브랜치로 사용할 브랜치를 지정할 수 CodeCatalyst 있습니다. 의 모든 소스 CodeCatalyst 리포지토리에는 생성 방법에 관계없이 콘텐츠와 기본 브랜치가 있습니다. 블루프린트를 사용하여 프로젝트를 생성하는 경우, 해당 프로젝트용으로 만든 소스 리포지토리의 기본 브랜치는 main으로 지정됩니다. 기본 브랜치의 컨텐츠는 해당 리포지토리의 개요 페이지에 자동으로 표시됩니다.

> **Important**
>
> CodeCatalyst 연결된 리포지토리의 기본 브랜치 변경 감지를 지원하지 않습니다. 연결된 저장소의 기본 분기를 변경하려면 먼저 연결을 해제하고 기본 분기를 변경한 다음 다시 연결해야 합니다. CodeCatalyst 자세한 내용은 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, GitLab Jira 프로젝트 연결 CodeCatalyst 단원을 참조하십시오.
> 가장 좋은 방법은 리포지토리를 연결하기 전에 항상 최신 버전의 확장 프로그램을 사용하는 것입니다.

기본 브랜치는 소스 리포지토리의 다른 모든 브랜치와 약간 다르게 취급됩니다. 이름 옆에 Default라는 특수 레이블이 있습니다. 기본 브랜치는 사용자가 Git 클라이언트를 사용하여 로컬 컴퓨터에 리포지토리를 복제할 때 로컬 리포지토리 (리포지토리) 에서 기본 또는 기본 브랜치로 사용되는 브랜치입니다. 또한 워크플로 YAML 파일을 저장하고 이슈에 대한 정보를 저장하기 위한 워크플로를 만들 때 기본적으로 사용됩니다. 에서 검색을 사용하는 CodeCatalyst 경우 저장소의 기본 분기만 검색됩니다. 기본 브랜치는 프로젝트의 여러 측면에서 기본적으로 사용되므로 브랜치를 기본 브랜치로 지정한 경우 브랜치를 삭제할 수 없습니다. 하지만 다른 브랜치를 기본 브랜치로 사용하도록 선택할 수 있습니다. 이렇게 하면 이전의 기본 브랜치에 적용된 모든 브랜치 규칙이 기본 브랜치로 지정한 브랜치에 자동으로 적용됩니다.

> **Note**
>
> 프로젝트의 소스 리포지토리에 대한 기본 브랜치를 변경하려면 프로젝트 관리자 역할이 있어야 합니다. CodeCatalyst 연결된 리포지토리에는 적용되지 않습니다.

리포지토리의 기본 브랜치 보기 및 변경하기

1. 리포지토리가 있는 프로젝트로 이동합니다.
2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.

   기본 브랜치를 포함하여 설정을 보려는 리포지토리를 선택합니다.
3. 리포지토리의 개요 페이지에서 기타를 선택한 다음 설정 관리를 선택합니다.
4. 기본 브랜치에서 기본 브랜치로 지정된 브랜치의 이름이 이름 옆에 Default라는 레이블과 함께 표시됩니다. 브랜치의 브랜치 목록에서 브랜치 이름 옆에 이와 동일한 라벨이 나타납니다.
5. 기본 브랜치를 변경하려면 편집을 선택합니다.

   > **Note**
   >
   > 기본 브랜치를 변경하려면 프로젝트에 프로젝트 관리자 역할이 있어야 합니다.
6. 드롭다운 목록에서 기본 브랜치로 설정하려는 브랜치의 이름을 선택한 다음 저장을 선택합니다.

브랜치 규칙을 사용하여 브랜치에 허용된 작업 관리

브랜치를 만들면 해당 역할의 권한에 따라 해당 브랜치에 대해 특정 작업이 허용됩니다. 브랜치 규칙을 구성하여 특정 브랜치에 허용되는 작업을 변경할 수 있습니다. 브랜치 규칙은 프로젝트에서 사용자의 역할을 기반으로 합니다. 브랜치에 커밋을 푸시하는 등 사전 정의된 일부 작업을 프로젝트에서 특정 역할을 가진 사용자만 수행할 수 있도록 제한할 수 있습니다. 이렇게 하면 특정 작업을 수행할 수 있는 역할을 제한하여 프로젝트의 특정 브랜치를 보호할 수 있습니다. 예를 들어 프로젝트 관리자 역할을 가진 사용자만 해당 브랜치를 병합하거나 푸시할 수 있도록 브랜치 규칙을 구성하면 프로젝트에서 다른 역할을 가진 사용자는 해당 브랜치의 코드를 변경할 수 없습니다.

브랜치에 규칙을 만들 때 발생할 수 있는 모든 영향을 신중하게 고려해야 합니다. 예를 들어 브랜치에 대한 푸시를 프로젝트 관리자 역할을 가진 사용자로 제한하도록 선택하면 기여자 역할을 가진 사용자는 해당 브랜치에서 워크플로를 만들거나 편집할 수 없습니다. 워크플로가 해당 브랜치에 YAML 저장되고 해당 사용자는 변경 사항을 커밋하고 푸시할 수 없기 때문입니다. YAML 가장 좋은 방법은 브랜치 규칙을 만든 후 테스트하여 의도하지 않은 영향이 없는지 확인하는 것입니다. 브랜치 규칙을 풀 리퀘스트의 승인 규칙과 함께 사용할 수도 있습니다. 자세한 내용은 풀 리퀘스트를 승인 규칙과 병합하기 위한 요구 사항 관리 단원을 참조하십시오.

> **Note**
>
> 프로젝트의 소스 리포지토리에 대한 분기 규칙을 관리하려면 프로젝트 관리자 역할이 있어야 합니다. CodeCatalyst 연결된 리포지토리에는 분기 규칙을 만들 수 없습니다.
> 역할의 기본 권한보다 더 제한적인 분기 규칙만 만들 수 있습니다. 프로젝트에서 사용자 역할이 허용하는 것보다 더 관대한 분기 규칙은 만들 수 없습니다. 예를 들어 리뷰어 역할을 가진 사용자가 브랜치로 푸시하도록 허용하는 분기 규칙을 만들 수 없습니다.

소스 리포지토리의 기본 분기에 적용되는 분기 규칙은 다른 분기에 적용되는 분기 규칙과 약간 다르게 동작합니다. 기본 브랜치에 적용된 모든 규칙은 기본 브랜치로 지정한 모든 브랜치에 자동으로 적용됩니다. 이전에 기본 브랜치로 설정된 브랜치는 삭제에 대한 보호 기능이 더 이상 없다는 점을 제외하면 규칙이 계속 적용됩니다. 이 보호는 현재 기본 브랜치에만 적용됩니다.

분기 규칙에는 표준과 사용자 지정이라는 두 가지 상태가 있습니다. Standard는 브랜치에서 허용되는 작업이 브랜치 작업에 대한 사용자 역할에 대한 권한과 일치하는 작업임을 나타냅니다. CodeCatalyst 어떤 역할에 어떤 권한이 있는지 자세히 알아보려면 을 참조하십시오사용자 역할을 통한 액세스 권한 부여. 사용자 지정은 하나 이상의 분기 작업에 해당 작업을 수행할 수 있는 특정 역할 목록이 있는 작업이 프로젝트의 사용자 역할에 의해 부여된 기본 권한과 다르다는 것을 나타냅니다.

> **Note**
>
> 브랜치에 대해 하나 이상의 작업을 제한하는 분기 규칙을 만들면 프로젝트 관리자 역할을 가진 사용자만 해당 브랜치를 삭제할 수 있도록 브랜치 삭제 작업이 자동으로 설정됩니다.

다음 표에는 브랜치에서 이러한 작업을 수행할 수 있는 작업 및 역할의 기본 설정이 나열되어 있습니다.

브랜치 작업 및 역할

브랜치 액션	분기 규칙이 적용되지 않을 때 이 작업을 수행할 수 있는 역할
브랜치에 병합 (브랜치에 대한 풀 리퀘스트 병합 포함)	프로젝트 관리자, 컨트리뷰터
브랜치로 푸시	프로젝트 관리자, 컨트리뷰터
브랜치 삭제	프로젝트 관리자, 컨트리뷰터
브랜치 삭제 (기본 브랜치)	허용되지 않음

분기 규칙을 삭제할 수는 없지만 브랜치에서 이 작업을 수행할 수 있는 모든 역할의 작업을 허용하도록 업데이트하면 사실상 규칙이 제거됩니다.

> **Note**
>
> 프로젝트의 소스 리포지토리에 대한 분기 규칙을 구성하려면 프로젝트 관리자 역할이 있어야 합니다. CodeCatalyst 연결된 리포지토리에는 적용되지 않습니다. 연결된 리포지토리는 의 분기 규칙을 지원하지 않습니다. CodeCatalyst

리포지토리의 분기 규칙을 보고 편집하려면

1. 리포지토리가 있는 프로젝트로 이동합니다.
2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.

분기 규칙을 보려는 리포지토리를 선택합니다.

3. 리포지토리의 개요 페이지에서 Branch를 선택합니다.
4. 브랜치 규칙 열에서 리포지토리의 각 브랜치에 대한 규칙 상태를 확인합니다. 표준에 따르면 분기 작업에 대한 규칙은 원본 리포지토리에서 만든 모든 브랜치의 기본 규칙이며 프로젝트에서 해당 역할에 부여된 권한과 일치합니다. 사용자 지정은 하나 이상의 분기 작업에 해당 분기에 허용되는 하나 이상의 작업을 다른 역할 집합으로 제한하는 규칙이 있음을 나타냅니다.

   브랜치에 대한 브랜치 규칙의 세부 사항을 보려면 검토하려는 브랜치 옆의 Standard 또는 Custom 이라는 단어를 선택하십시오.
5. 분기 규칙을 만들거나 변경하려면 설정 관리를 선택합니다. 소스 리포지토리의 설정 페이지에 있는 분기 규칙에서 편집을 선택합니다.
6. Branch의 드롭다운 목록에서 규칙을 구성하려는 브랜치의 이름을 선택합니다. 허용된 각 작업 유형에 대해 드롭다운 목록에서 해당 작업을 수행하도록 허용하려는 역할을 선택한 다음 [Save] 를 선택합니다.

브랜치를 위한 Git 명령어

Git을 사용하여 컴퓨터 (로컬 리포지토리) 또는 개발 환경에 있는 소스 리포지토리의 복제본에서 브랜치를 생성, 관리 및 삭제한 다음 변경 내용을 CodeCatalyst 소스 리포지토리 (원격 리포지토리) 에 커밋하고 푸시할 수 있습니다. 예:

브랜치를 위한 일반적인 Git 명령

로컬 리포지토리의 모든 브랜치를 나열하며, 현재 브랜치 옆에 별표(*)를 표시합니다.	`git branch`
원격 리포지토리의 모든 기존 브랜치에 대한 정보를 로컬 리포지토리로 가져옵니다.	`git fetch`
로컬 리포지토리의 브랜치와 로컬 리포지토리의 원격 추적 브랜치를 모두 나열합니다.	`git branch -a`
로컬 리포지토리의 원격 추적 브랜치만 나열합니다.	`git branch -r`
지정된 브랜치 이름을 사용하여 로컬 리포지토리에 브랜치를 만듭니다. 이 브랜치는 변경 사항	`git branch branch-name`

을 커밋하고 푸시할 때까지 원격 리포지토리에 표시되지 않습니다.	
지정된 브랜치 이름을 사용하여 로컬 리포지토리에 브랜치를 만든 다음 브랜치로 전환합니다.	`git checkout -b branch-name`
지정된 브랜치 이름을 사용하여 로컬 리포지토리의 다른 브랜치로 전환합니다.	`git checkout other-branch-name`
로컬 리포지토리의 지정된 원격 리포지토리 닉네임과 지정된 브랜치 이름을 사용하여 로컬 리포지토리에서 원격 리포지토리로 브랜치를 푸시합니다. 또한 로컬 리포지토리의 브랜치에 대한 업스트림 추적 정보를 설정합니다.	`git push -u remote-name branch-name`
로컬 리포지토리의 다른 브랜치에서 변경한 내용을 로컬 리포지토리의 현재 브랜치에 병합합니다.	`git merge from-other-branch-name`
병합되지 않은 작업이 포함되어 있지 않는 한 로컬 리포지토리에서 브랜치를 삭제합니다.	`git branch -d branch-name`

로컬 리포지토리의 지정된 원격 리포지토리 닉네임과 지정된 브랜치 이름을 사용하여 원격 리포지토리의 브랜치를 삭제합니다. (콜론(:) 사용에 주의하세요.) 또는 명령의 `--delete` 일부로 지정할 수도 있습니다.	`git push remote-name :branch-name`
	
	`git push remote-name --delete branch-name`

자세한 내용은 Git 설명서를 참조하십시오.

지점 및 세부 정보 보기

Amazon CodeCatalyst 콘솔에서 파일 CodeCatalyst, 폴더 및 특정 브랜치에 대한 최신 커밋의 세부 정보를 포함하여 Amazon의 원격 브랜치에 대한 정보를 볼 수 있습니다. 또한 Git 명령과 로컬 운영 체제를 사용하여 원격 및 로컬 브랜치에 대한 이 정보를 볼 수 있습니다.

브랜치를 보려면 (콘솔)

1. CodeCatalyst 콘솔에서 브랜치를 보려는 소스 리포지토리가 포함된 프로젝트로 이동합니다. 코드를 선택하고 소스 리포지토리를 선택한 다음 소스 리포지토리를 선택합니다.
2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.

   브랜치를 보려는 리포지토리를 선택합니다.
3. 리포지토리의 기본 브랜치가 표시됩니다. 브랜치에 있는 파일 및 폴더 목록, 가장 최근 커밋에 대한 정보, README .md 파일 (브랜치에 있는 경우) 의 내용을 볼 수 있습니다. 다른 브랜치에 대한 정보를 보려면 리포지토리의 브랜치 드롭다운 목록에서 해당 브랜치를 선택합니다.
4. 리포지토리의 모든 브랜치를 보려면 모두 보기를 선택합니다. 브랜치 페이지에는 각 브랜치의 이름, 가장 최근 커밋, 규칙에 대한 정보가 표시됩니다.

Git과 운영 체제를 사용하여 브랜치와 세부 정보를 보는 방법에 대한 자세한 내용은 브랜치용 일반 Git 명령, Git 설명서 및 운영 체제 설명서를 참조하십시오.

브랜치 삭제

브랜치가 더 이상 필요하지 않으면 삭제할 수 있습니다. 예를 들어 기능이 변경된 브랜치를 기본 브랜치에 병합하고 해당 기능이 출시된 경우 변경 사항이 이미 기본 브랜치의 일부이므로 원래 기능 브랜치를 삭제하는 것이 좋습니다. 브랜치 수를 적게 유지하면 사용자가 작업하려는 변경 사항이 포함된 브랜치를 찾는 데 도움이 될 수 있습니다. 브랜치를 삭제하면 사용자가 변경 내용을 가져와 동기화할 때까지 해당 브랜치의 복사본이 로컬 컴퓨터의 리포지토리 복제본에 남아 있습니다.

브랜치를 삭제하려면 (콘솔)

1. 리포지토리가 있는 프로젝트로 이동합니다.
2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.

   브랜치를 삭제하려는 리포지토리를 선택합니다.
3. 리포지토리의 개요 페이지에서 브랜치 이름 옆의 드롭다운 선택기를 선택한 다음 모두 보기를 선택합니다.
4. 삭제하려는 브랜치를 선택한 다음 브랜치 삭제를 선택합니다.

> **Note**
>
> 리포지토리의 기본 브랜치는 삭제할 수 없습니다.

5. 확인 대화 상자가 표시됩니다. 리포지토리, 열려 있는 풀 리퀘스트 수, 브랜치와 관련된 워크플로우 수를 보여 줍니다.
6. 브랜치 삭제를 확인하려면 텍스트 상자에 delete를 입력한 다음 삭제를 선택합니다.

Git을 사용하여 브랜치를 삭제할 수도 있습니다. 자세한 내용은 브랜치를 위한 일반적인 Git 명령 단원을 참조하십시오.

Amazon에서 소스 코드 파일 관리 CodeCatalyst

CodeCatalystAmazon에서 파일은 버전이 관리되고 독립된 정보로, 파일이 저장되는 소스 리포지토리 및 브랜치의 사용자와 다른 사용자가 사용할 수 있습니다. 디렉터리 구조로 리포지토리 파일을 구성할 수 있습니다. CodeCatalyst파일에 커밋된 모든 변경 사항을 자동으로 추적합니다. 서로 다른 저장소 브랜치에 다양한 버전의 파일을 저장할 수 있습니다.

소스 리포지토리에서 여러 파일을 추가하거나 편집하려면 Git 클라이언트, 개발 환경 또는 통합 개발 환경 () 을 사용할 수 있습니다. IDE 단일 파일을 추가하거나 편집하려면 콘솔을 CodeCatalyst 사용할 수 있습니다.

주제

- 파일 생성 또는 추가
- 파일 보기
- 파일 변경 기록 보기
- 파일 편집
- 파일 이름 변경 또는 삭제

파일 생성 또는 추가

파일을 생성하고 소스 리포지토리에 추가하려면 Amazon CodeCatalyst 콘솔, 개발 환경, 연결된 통합 개발 환경 (IDE) 또는 Git 클라이언트를 사용할 수 있습니다. CodeCatalyst 콘솔에는 파일 생성을 위한 코드 편집기가 포함되어 있습니다. 이 편집기를 사용하면 저장소의 README 브랜치에서.md 파일과

같은 간단한 파일을 만들거나 편집할 수 있는 편리한 방법입니다. 둘 이상의 파일을 작업할 때는 개발 환경을 만들어 보세요.

소스 리포지토리에서 개발 환경을 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
3. 코드 작업을 하려는 소스 리포지토리를 선택합니다.
4. 개발 환경 생성을 선택합니다.
5. 드롭다운 IDE 메뉴에서 지원되는 항목을 선택합니다. 자세한 내용은 개발 환경을 위해 지원되는 통합 개발 환경 섹션을 참조하세요.
6. 다음 중 하나를 수행합니다.
   - 기존 브랜치에서 작업을 선택한 다음 기존 브랜치 드롭다운 메뉴에서 브랜치를 선택합니다.
   - 새 브랜치에서 근무를 선택하고 브랜치 이름 필드에 브랜치 이름을 입력한 다음 드롭다운 메뉴에서 브랜치 생성 기반으로 새 브랜치를 만들 브랜치를 선택합니다.
7. 선택적으로 개발 환경의 이름을 추가하거나 구성을 편집할 수 있습니다.
8. 생성(Create)을 선택합니다.

콘솔에서 CodeCatalyst 파일을 만들려면

1. 파일을 만들려는 프로젝트로 이동합니다. 리포지토리로 이동하는 방법에 대한 자세한 내용은 을 참조하십시오소스 리포지토리 보기.
2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.

   파일을 만들려는 리포지토리를 선택합니다.
3. (선택 사항) 기본 브랜치가 아닌 다른 브랜치에 파일을 만들려면 파일을 만들려는 브랜치를 선택합니다.
4. 파일 만들기를 선택합니다.
5. 파일 이름에 파일 이름을 입력합니다. 편집기에 파일 내용을 추가합니다.

> **Tip**
>
> 브랜치 루트의 하위 폴더나 하위 디렉터리에 파일을 만들려면 해당 구조를 파일 이름의 일부로 포함하십시오.

변경 내용이 만족스러우면 [Commit] 을 선택합니다.

6. 파일 이름에서 파일 이름을 검토하고 필요에 따라 변경합니다. Branch에서 사용 가능한 브랜치 목록에서 파일을 만들려는 브랜치를 선택할 수도 있습니다. 커밋 메시지에 필요에 따라 변경한 이유에 대한 간략하지만 유용한 설명을 입력합니다. 이 정보는 소스 저장소에 파일을 추가하는 커밋에 대한 기본 커밋 정보로 표시됩니다.
7. 커밋을 선택하여 파일을 커밋하고 소스 리포지토리에 푸시합니다.

파일을 로컬 컴퓨터에 복제하고 Git 클라이언트 또는 연결된 통합 개발 환경 IDE () 을 사용하여 파일 및 변경 내용을 푸시하여 소스 리포지토리에 파일을 추가할 수도 있습니다.

> **Note**
>
> Git 하위 모듈을 추가하려면 Git 클라이언트 또는 개발 환경을 사용하고 명령을 실행해야 합니다. git submodule add CodeCatalyst 콘솔에서 Git 하위 모듈을 추가하거나 볼 수 없으며 pull 요청에서 Git 하위 모듈의 차이점을 볼 수 없습니다. Git 서브모듈에 대한 자세한 내용은 Git 설명서를 참조하십시오.

Git 클라이언트 또는 연결된 통합 개발 환경을 사용하여 파일을 추가하려면 () IDE

1. 소스 리포지토리를 로컬 컴퓨터에 복제합니다. 자세한 내용은 소스 리포지토리 복제 단원을 참조하십시오.
2. 로컬 리포지토리에 파일을 생성하거나 로컬 리포지토리에 파일을 복사하세요.
3. 다음 중 하나를 수행하여 커밋을 만들고 푸시하세요.

   - Git 클라이언트를 사용하는 경우 터미널 또는 명령줄에서 추가하려는 파일 이름을 지정하여 git add 명령을 실행합니다. 또는 추가되거나 변경된 파일을 모두 추가하려면 git add 명령을 실행한 다음 단일 또는 이중 마침표를 실행하여 현재 디렉터리 수준의 모든 변경 내용을 포함할지 (단일 기간) 또는 현재 디렉터리와 모든 하위 디렉터리의 모든 변경 내용을 포함할지 (이중 기간) 지정합니다. 변경 내용을 커밋하려면 git commit -m 명령을 실행하고 커밋 메시지를 제공하십시오.

의 소스 리포지토리에 CodeCatalyst 변경 내용을 푸시하려면 를 실행하십시오git push. Git 명령에 대한 자세한 내용은 Git 설명서 및 을 참조하십시오. 브랜치를 위한 Git 명령어

- 개발 환경 또는 IDE an을 사용하는 경우 에서 파일을 만들고 파일을 추가한 다음 변경 사항을 커밋하고 푸시하십시오. IDE 자세한 내용은 IDE 설명서를 의 개발 환경을 사용하여 코드 작성 및 수정 CodeCatalyst 참조하거나 참조하십시오.

파일 보기

Amazon CodeCatalyst 콘솔에서 소스 리포지토리의 파일을 볼 수 있습니다. 기본 브랜치 및 기타 브랜치에서 파일을 볼 수 있습니다. 파일 내용은 보려는 브랜치에 따라 달라질 수 있습니다.

CodeCatalyst 콘솔에서 파일을 보려면

1. 파일을 보려는 프로젝트로 이동합니다. 자세한 내용은 소스 리포지토리 보기 단원을 참조하십시오.
2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.

   파일을 보려는 리포지토리를 선택합니다.
3. 기본 브랜치의 경우 파일 및 폴더 목록이 표시됩니다. 파일은 종이 아이콘으로 표시되고 폴더는 폴더 아이콘으로 표시됩니다.
4. 다음을 수행합니다.
   - 다른 브랜치에 있는 파일 및 폴더를 보려면 브랜치 목록에서 해당 브랜치를 선택합니다.
   - 폴더를 확장하려면 목록에서 폴더를 선택합니다.
5. 특정 파일의 내용을 보려면 목록에서 해당 파일을 선택합니다. 파일 내용이 브랜치에 표시됩니다. 다른 브랜치에 있는 파일의 내용을 보려면 브랜치 선택기에서 원하는 브랜치를 선택합니다.

> **Tip**
>
> 파일 콘텐츠를 볼 때 파일 보기에서 보려는 추가 파일을 선택할 수 있습니다. 파일을 편집하려면 편집을 선택합니다.

콘솔에서 여러 파일을 볼 수 있습니다. Git 클라이언트 또는 통합 개발 환경 IDE () 을 사용하여 로컬 컴퓨터에 클론한 파일을 볼 수도 있습니다. 자세한 내용은 Git 클라이언트 또는 설명서를 참조하십시오. IDE

> **Note**
>
> 콘솔에서는 Git 서브모듈을 볼 수 없습니다. CodeCatalyst Git 서브모듈에 대한 자세한 내용은 Git 설명서를 참조하십시오.

파일 변경 기록 보기

Amazon CodeCatalyst 콘솔에서 소스 리포지토리의 파일 변경 기록을 볼 수 있습니다. 이를 통해 파일 기록을 보기로 선택한 브랜치에 대한 다양한 커밋으로 인한 파일 변경 내용을 이해할 수 있습니다. 예를 들어 소스 리포지토리의 브랜치에서 **readme.md** 파일 변경 기록을 보면 해당 **main** 브랜치의 해당 파일에 대한 변경 내용이 포함된 커밋 목록이 표시됩니다.

> **Note**
>
> CodeCatalyst 콘솔에서는 링크된 저장소의 파일 기록을 볼 수 없습니다.

CodeCatalyst 콘솔에서 파일 기록을 보려면

1. 파일 기록을 보려는 프로젝트로 이동합니다. 자세한 내용은 소스 리포지토리 보기 단원을 참조하십시오.
2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
3. 파일 기록을 보려는 리포지토리를 선택합니다. 파일 기록을 보려는 브랜치를 선택한 다음 목록에서 파일을 선택합니다. 내역 보기를 선택합니다.
4. 지정된 브랜치에서 이 파일의 변경 내용을 포함한 커밋 목록을 검토하십시오. 특정 커밋에 포함된 변경 사항의 세부 정보를 보려면 목록에서 해당 커밋에 대한 커밋 메시지를 선택합니다. 해당 커밋과 상위 커밋 간의 차이점이 표시됩니다.
5. 다른 브랜치의 파일 변경 기록을 검토하려면 브랜치 선택기를 사용하여 해당 브랜치의 보기를 변경하고 파일 목록에서 파일을 선택한 다음 히스토리 보기를 선택합니다.

> **Note**
>
> 콘솔에서는 Git 서브모듈의 변경 기록을 볼 수 없습니다. CodeCatalyst Git 서브모듈에 대한 자세한 내용은 Git 설명서를 참조하십시오.

파일 편집

Amazon CodeCatalyst 콘솔에서 개별 파일을 편집할 수 있습니다. 여러 파일을 한 번에 편집하려면 개발 환경을 만들거나 리포지토리를 복제하고 Git 클라이언트 또는 통합 개발 환경 IDE () 을 사용하여 변경합니다. 자세한 내용은 의 개발 환경을 사용하여 코드 작성 및 수정 CodeCatalyst 또는 소스 리포지토리 복제을 참조하세요.

콘솔에서 파일을 편집하려면 CodeCatalyst

1. 파일을 편집하려는 프로젝트로 이동합니다. 리포지토리로 이동하는 방법에 대한 자세한 내용은 을 참조하십시오소스 리포지토리 보기.
2. 파일을 편집하려는 리포지토리를 선택합니다. 브랜치 보기를 선택한 다음 작업하려는 브랜치를 선택합니다. 해당 브랜치의 파일 및 폴더 목록에서 파일을 선택합니다.

   파일 내용이 표시됩니다.
3. 편집을 선택합니다.
4. 편집기에서 파일 내용을 편집한 다음 [Commit] 을 선택합니다. 변경 내용 커밋에서 커밋 메시지에 변경 사항에 대한 추가 정보를 추가할 수도 있습니다. 변경 사항에 만족하면 [커밋] 을 선택합니다.

파일 이름 변경 또는 삭제

개발 환경, 컴퓨터의 로컬 또는 통합 개발 환경 () 에서 파일 이름을 바꾸거나 파일을 삭제할 수 있습니다. IDE 파일 이름을 바꾸거나 파일을 삭제한 후에는 해당 변경 사항을 커밋하여 소스 리포지토리에 푸시하십시오. Amazon CodeCatalyst 콘솔에서는 파일의 이름을 바꾸거나 파일을 삭제할 수 없습니다.

Amazon에서 풀 리퀘스트를 사용하여 코드 검토하기 CodeCatalyst

풀 리퀘스트는 사용자와 다른 프로젝트 멤버가 한 브랜치에서 다른 브랜치로의 코드 변경 사항을 검토하고, 코멘트를 달고, 병합할 수 있는 기본 방법입니다. 풀 리퀘스트를 사용하면 코드 변경 사항을 공동으로 검토하여 사소한 변경이나 수정, 주요 기능 추가 또는 출시된 소프트웨어의 새 버전을 검토할 수 있습니다. 이슈를 사용하여 프로젝트 작업을 추적하는 경우 특정 이슈를 풀 리퀘스트에 연결하여 풀 리퀘스트의 코드 변경으로 해결되는 문제를 추적할 수 있습니다. 풀 리퀘스트를 생성, 업데이트, 코멘트, 병합 또는 종료하면 풀 리퀘스트 작성자는 물론 해당 풀 리퀘스트의 필수 또는 선택적 검토자에게도 이메일이 자동으로 전송됩니다.

> **Tip**
>
> 프로필의 일부로 이메일을 수신할 풀 리퀘스트 이벤트를 구성할 수 있습니다. 자세한 내용은 에서 Slack 및 이메일 알림 보내기 CodeCatalyst 단원을 참조하십시오.

풀 리퀘스트에는 소스 리포지토리에 두 개의 브랜치가 필요합니다. 하나는 검토하려는 코드가 들어 있는 소스 브랜치이고, 다른 하나는 검토된 코드를 병합하려는 대상 브랜치입니다. 소스 브랜치에는 대상 브랜치에 병합하려는 변경 내용이 포함된 커밋인 커밋이 들어 있습니다. AFTER 대상 브랜치에는 풀 리퀘스트 브랜치가 대상 브랜치에 병합되기 전의 코드 상태를 나타내는 BEFORE 커밋이 포함되어 있습니다.

> **Note**
>
> 풀 리퀘스트를 생성하는 동안 표시되는 차이는 소스 브랜치의 팁과 대상 브랜치의 팁의 차이입니다. 풀 리퀘스트를 생성하고 나면 선택한 풀 리퀘스트의 수정과 풀 리퀘스트를 생성할 때 대상 브랜치의 끝부분이었던 커밋 간의 차이가 표시됩니다. Git의 차이점 및 병합 기준에 대한 자세한 내용은 Git git-merge-base설명서를 참조하십시오.

특정 소스 리포지토리 및 브랜치에 대한 풀 리퀘스트가 생성되는 동안 프로젝트 작업의 일환으로 해당 리포지토리와 브랜치를 만들고, 보고, 검토하고, 종료할 수 있습니다. 풀 리퀘스트를 보고 작업하기 위해 소스 리포지토리를 볼 필요는 없습니다. 풀 리퀘스트를 생성하면 풀 리퀘스트 상태가 열기로 설정됩니다. 풀 리퀘스트는 CodeCatalyst 콘솔에서 병합하여 상태를 Merged로 변경하고, 닫으면 상태가 Closed로 변경될 때까지 열려 있는 상태로 유지됩니다.

코드를 검토한 후에는 여러 방법 중 하나로 풀 리퀘스트 상태를 변경할 수 있습니다.

- CodeCatalyst 콘솔에서 풀 리퀘스트를 병합하세요. 풀 리퀘스트의 소스 브랜치에 있는 코드는 대상 브랜치에 병합됩니다. 풀 리퀘스트 상태가 Merged로 변경됩니다. 다시 열기로 변경할 수 없습니다.
- 브랜치를 로컬로 병합하고 변경 사항을 푸시한 다음 CodeCatalyst 콘솔에서 풀 리퀘스트를 종료합니다.
- CodeCatalyst 콘솔을 사용하면 병합하지 않고 풀 리퀘스트를 종료할 수 있습니다. 그러면 상태가 Closed로 변경되고 소스 브랜치의 코드가 대상 브랜치에 병합되지 않습니다.

풀 요청을 생성하려면 먼저 다음을 수행해야 합니다.

- 검토하려는 코드 변경 사항을 커밋하고 브랜치 (소스 브랜치) 에 푸시합니다.

- 풀 리퀘스트를 생성할 때 실행되는 모든 워크플로에 대해 다른 사용자에게 알릴 수 있도록 프로젝트 알림을 설정하세요. (이 단계는 선택 사항이지만 권장됩니다.)

주제

풀 요청 생성

풀 요청을 생성하면 변경 내용을 다른 브랜치에 병합하기 전에 다른 사용자가 코드 변경을 보고 검토할 수 있습니다. 먼저 코드 변경을 위한 브랜치를 생성합니다. 이 브랜치를 풀 요청의 소스 브랜치라고도 합니다. 리포지토리에 변경 사항을 커밋하고 푸시한 후 원본 브랜치의 콘텐츠를 대상 브랜치의 콘텐츠와 비교하는 풀 요청을 생성할 수 있습니다.

Amazon CodeCatalyst 콘솔에서 특정 브랜치, 풀 리퀘스트 페이지 또는 프로젝트 개요에서 풀 요청을 생성할 수 있습니다. 특정 브랜치에서 풀 리퀘스트를 생성하면 풀 리퀘스트 생성 페이지에 리포지토리 이름과 소스 브랜치가 자동으로 제공됩니다. 풀 리퀘스트를 생성하면 풀 리퀘스트에 대한 업데이트와 풀 리퀘스트가 합쳐지거나 닫힐 때 관련 이메일이 자동으로 수신됩니다.

> **Note**
>
> 풀 리퀘스트를 생성하는 동안 표시되는 차이는 소스 브랜치의 끝과 대상 브랜치의 끝부분 간의 차이입니다. 풀 리퀘스트가 생성되면 선택한 풀 리퀘스트의 수정과 풀 리퀘스트를 생성할 때 대상 브랜치의 팁이었던 커밋 간의 차이가 표시됩니다. Git의 차이점과 병합 기준에 대한 자세한 내용은 Git git-merge-base설명서를 참조하십시오.

풀 요청을 생성할 때 나를 위한 설명 쓰기 기능을 사용하여 Amazon Q에서 풀 요청에 포함된 변경 사항에 대한 설명을 자동으로 생성하도록 할 수 있습니다. 이 옵션을 선택하면 Amazon Q는 코드 변경 사항이 포함된 소스 브랜치와 이러한 변경 사항을 병합하려는 대상 브랜치 간의 차이를 분석합니다. 그런 다음 변경 내용에 대한 요약과 해당 변경의 의도와 영향에 대한 최상의 해석을 생성합니다. 이 기능은

미국 서부 (오레곤) 지역에서만 CodeCatalyst 풀 리퀘스트에 사용할 수 있습니다. 연결된 리포지토리의 풀 리퀘스트에는 나를 위한 설명 쓰기 기능을 사용할 수 없습니다.

> **Note**
>
> > **Note**
> >
> > Amazon Bedrock 제공: 자동 악용 탐지 AWS 기능을 구현합니다. 나를 위한 설명 작성, 콘텐츠 요약 생성, 작업 추천, Amazon Q를 사용하여 프로젝트에 기능 생성 또는 추가, Amazon Q Developer Agent의 소프트웨어 개발 기능에 대한 Amazon Q 이슈 할당 기능이 Amazon Bedrock에 구축되어 있기 때문에 사용자는 Amazon Bedrock에 구현된 제어 기능을 최대한 활용하여 안전, 보안 및 인공 지능 (AI) 의 책임 있는 사용을 강화할 수 있습니다.

풀 리퀘스트를 생성하면 풀 리퀘스트를 생성할

1. 프로젝트로 이동합니다.
2. 다음 중 하나를 수행합니다.
   - 탐색 창에서 코드를 선택하고 풀 요청을 선택한 다음 풀 요청 생성을 선택합니다.
   - 리포지토리 홈 페이지에서 추가를 선택한 다음 풀 리퀘스트 생성을 선택합니다.
   - 프로젝트 페이지에서 풀 리퀘스트 생성을 선택합니다.
3. 소스 리포지토리에서 지정된 소스 리포지토리가 커밋된 코드를 포함하는 리포지토리인지 확인합니다. 이 옵션은 리포지토리의 기본 페이지에서 풀 리퀘스트를 만들지 않은 경우에만 나타납니다.
4. 대상 브랜치에서 코드를 검토한 후 병합할 브랜치를 선택합니다.
5. 소스 브랜치에서 커밋된 코드가 포함된 브랜치를 선택합니다.
6. 풀 리퀘스트 제목에 다른 사용자가 검토해야 할 내용과 이유를 이해하는 데 도움이 되는 제목을 입력합니다.
7. (선택 사항) 풀 요청 설명에 문제 링크 또는 변경 내용 설명과 같은 정보를 입력합니다.

> **Tip**
>
> Pull Request에 포함된 변경 사항에 대한 설명을 CodeCatalyst 자동으로 생성하도록 나를 위한 설명 쓰기를 선택할 수 있습니다. 자동으로 생성된 설명을 풀 리퀘스트에 추가한 후 변경할 수 있습니다.

이 기능을 사용하려면 스페이스에 대해 제너레이티브 AI 기능을 활성화해야 하며 연결된 리포지토리의 풀 요청에는 사용할 수 없습니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.

8. (선택 사항) 이슈에서 이슈 연결을 선택한 다음 목록에서 이슈를 선택하거나 해당 ID를 입력합니다. 이슈의 연결을 해제하려면 연결 해제 아이콘을 선택합니다.
9. (선택 사항) 필수 검토자에서 필수 검토자 추가를 선택합니다. 프로젝트 멤버 목록에서 선택하여 추가합니다. 풀 리퀘스트를 대상 브랜치에 병합하려면 먼저 필수 검토자가 변경 사항을 승인해야 합니다.

> **Note**
>
> 검토자를 필수 검토자와 선택적 검토자로 모두 추가할 수는 없습니다. 자신을 리뷰어로 추가할 수 없습니다.

10. (선택 사항) 선택적 검토자에서 선택적 검토자 추가를 선택합니다. 프로젝트 멤버 목록에서 선택하여 추가합니다. 선택적 검토자는 풀 리퀘스트를 대상 브랜치에 병합하기 전에 변경 사항을 요구 사항으로 승인할 필요가 없습니다.
11. 브랜치 간의 차이점을 검토하세요. 풀 리퀘스트에 표시되는 차이는 소스 브랜치의 수정 버전과 풀 리퀘스트가 생성된 시점의 대상 브랜치의 헤드 커밋인 병합 베이스 사이의 변경 내용입니다. 변경 사항이 표시되지 않으면 브랜치가 동일하거나 소스와 대상 모두에 대해 동일한 브랜치를 선택했을 수 있습니다.
12. 풀 리퀘스트에 검토하려는 코드와 변경 사항이 포함되어 있다고 판단되면 [Create] 를 선택합니다.

> **Note**
>
> 풀 리퀘스트를 생성한 후 코멘트를 추가할 수 있습니다. 풀 리퀘스트나 파일의 개별 라인, 전체 풀 리퀘스트에 댓글을 추가할 수 있습니다. @ 기호 뒤에 파일 이름을 붙여 파일 등의 리소스에 링크를 추가할 수 있습니다.

브랜치에서 풀 리퀘스트를 만들려면

1. 풀 리퀘스트를 만들려는 프로젝트로 이동합니다.
2. 탐색 창에서 소스 리포지토리를 선택한 다음 검토할 코드 변경 사항이 있는 브랜치가 포함된 리포지토리를 선택합니다.

3. 기본 브랜치 이름 옆의 드롭다운 화살표를 선택한 다음 목록에서 원하는 브랜치를 선택합니다. 리포지토리의 모든 브랜치를 보려면 모두 보기를 선택합니다.
4. 추가를 선택한 다음 풀 리퀘스트 생성을 선택합니다.
5. 리포지토리와 소스 브랜치가 미리 선택됩니다. 대상 브랜치에서 검토 후 코드를 병합할 브랜치를 선택합니다. 풀 리퀘스트 제목에 다른 프로젝트 사용자가 검토해야 하는 내용과 그 이유를 이해하는 데 도움이 되는 제목을 입력합니다. 필요에 따라 풀 리퀘스트 설명란에 관련 문제 링크를 붙여 넣거나 변경한 내용에 대한 설명을 추가하는 등 자세한 정보를 제공할 수 있습니다. CodeCatalyst

   > **Note**
   >
   > 풀 요청 생성 이벤트에 대해 실행되도록 구성된 워크플로는 풀 요청의 대상 브랜치가 워크플로에 지정된 브랜치 중 하나와 일치하는 경우 풀 요청이 생성된 후에 실행됩니다.

6. 브랜치 간의 차이점을 검토하세요. 변경 사항이 표시되지 않으면 브랜치가 동일하거나 소스와 대상 모두에 대해 동일한 브랜치를 선택했을 수 있습니다.
7. (선택 사항) 이슈에서 이슈 연결을 선택한 다음 목록에서 이슈를 선택하거나 해당 ID를 입력합니다. 이슈의 연결을 해제하려면 연결 해제 아이콘을 선택합니다.
8. (선택 사항) 필수 검토자에서 필수 검토자 추가를 선택합니다. 프로젝트 멤버 목록에서 선택하여 추가합니다. 풀 리퀘스트를 대상 브랜치에 병합하려면 먼저 필수 검토자가 변경 사항을 승인해야 합니다.

   > **Note**
   >
   > 필수 및 선택 사항으로 검토자를 추가할 수는 없습니다. 자신을 리뷰어로 추가할 수 없습니다.

9. (선택 사항) 선택적 검토자에서 선택적 검토자 추가를 선택합니다. 프로젝트 멤버 목록에서 선택하여 추가합니다. 선택적 검토자는 풀 리퀘스트를 대상 브랜치에 병합하기 전에 변경 사항을 승인할 필요가 없습니다.
10. 풀 요청에 검토하려는 변경 사항과 필수 검토자가 포함되어 있다고 판단되면 [Create] 를 선택합니다.

브랜치가 풀 요청의 대상 브랜치와 일치하는 곳에서 실행되도록 구성된 워크플로가 있는 경우, 풀 리퀘스트가 생성된 후 풀 리퀘스트 세부 정보 영역의 개요에 해당 워크플로우 실행에 대한 정보가 표시됩니다. 자세한 내용은 워크플로에 트리거 추가 단원을 참조하십시오.

풀 리퀘스트 보기

Amazon CodeCatalyst 콘솔에서 프로젝트에 대한 풀 리퀘스트를 볼 수 있습니다. 프로젝트 요약 페이지에는 프로젝트에 대해 진행 중인 모든 풀 리퀘스트가 표시됩니다. 상태와 상관없이 모든 풀 리퀘스트를 보려면 프로젝트의 풀 리퀘스트 페이지로 이동하세요. 풀 리퀘스트를 볼 때, 풀 리퀘스트의 변경 사항에 대한 모든 코멘트의 요약이 자동으로 생성되도록 선택할 수 있습니다.

Note

Note

Amazon Bedrock 제공: 자동 악용 탐지 AWS 기능을 구현합니다. 나를 위한 설명 작성, 콘텐츠 요약 생성, 작업 추천, Amazon Q를 사용하여 프로젝트에 기능 생성 또는 추가, Amazon Q Developer Agent의 소프트웨어 개발 기능에 대한 Amazon Q 이슈 할당 기능이 Amazon Bedrock에 구축되어 있기 때문에 사용자는 Amazon Bedrock에 구현된 제어 기능을 최대한 활용하여 안전, 보안 및 인공 지능 (AI) 의 책임 있는 사용을 강화할 수 있습니다.

미해결 풀 리퀘스트를 확인하려면

1. 풀 리퀘스트를 보려는 프로젝트로 이동합니다.
2. 프로젝트 페이지에는 풀 리퀘스트를 생성한 사람, 풀 리퀘스트의 브랜치가 들어 있는 리포지토리, 풀 리퀘스트가 생성된 날짜에 대한 정보를 포함한 오픈 풀 리퀘스트가 표시됩니다. 오픈 풀 리퀘스트 보기를 소스 리포지토리별로 필터링할 수 있습니다.
3. 모든 풀 리퀘스트를 보려면 모두 보기를 선택합니다. 선택기를 사용하여 옵션 중에서 선택할 수 있습니다. 예를 들어 모든 풀 리퀘스트를 보려면 모든 상태와 모든 작성자를 선택합니다.

   또는 탐색 창에서 코드를 선택한 다음 풀 요청을 선택한 다음 선택기를 사용하여 보기를 세분화할 수 있습니다.
4. 풀 리퀘스트 페이지에서 ID, 제목, 상태 등을 기준으로 풀 리퀘스트를 정렬할 수 있습니다. 풀 리퀘스트 페이지에 표시할 정보와 정보의 양을 사용자 지정하려면 톱니바퀴 아이콘을 선택하세요.
5. 특정 풀 리퀘스트를 보려면 목록에서 해당 풀 리퀘스트를 선택하십시오.
6. 이 풀 요청과 관련된 워크플로우 실행 상태 (있는 경우) 를 보려면 개요를 선택하고 워크플로우 실행 아래에 있는 풀 요청의 풀 요청 세부 정보 영역에 있는 정보를 검토하십시오.

워크플로가 풀 요청 생성 또는 수정 이벤트로 구성되어 있고 워크플로의 대상 분기 요구 사항이 풀 요청에 지정된 대상 브랜치와 일치하는 경우 워크플로가 실행됩니다. 자세한 내용은 워크플로에 트리거 추가 단원을 참조하십시오.

7. 연결된 이슈가 있는 경우, 개요를 선택하고 이슈 아래의 풀 리퀘스트 세부 정보에 있는 정보를 검토하십시오. 연결된 이슈를 보려면 목록에서 해당 ID를 선택하세요.
8. (선택 사항) 풀 리퀘스트 수정 시 코드 변경에 남긴 댓글의 요약을 만들려면 콘텐츠 요약 만들기를 선택합니다. 전체 풀 리퀘스트에 남긴 코멘트는 요약에 포함되지 않습니다.

> **Note**
>
> 이 기능을 사용하려면 해당 공간에 제너레이티브 AI 기능을 활성화해야 하고, 연결된 리포지토리에서는 사용할 수 없으며, 미국 서부 (오레곤) 지역에서만 사용할 수 있어야 합니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.

9. 풀 리퀘스트의 코드 변경 내용을 보려면 변경을 선택합니다. Files changed에서 풀 리퀘스트에서 변경 사항이 있는 파일 수와 풀 리퀘스트에서 해당 변경 사항에 대한 코멘트가 있는 파일을 빠르게 확인할 수 있습니다. 폴더 옆에 표시되는 코멘트 수는 해당 폴더 내 파일에 달린 코멘트 수를 나타냅니다. 폴더를 확장하면 폴더 내 각 파일에 대한 댓글 수를 볼 수 있습니다. 특정 코드 줄에 남겨진 주석도 볼 수 있습니다.

> **Note**
>
> 풀 리퀘스트의 모든 변경 사항을 콘솔에 표시할 수 있는 것은 아닙니다. 예를 들어 콘솔에서 Git 하위 모듈을 볼 수 없으므로 pull 요청에서 하위 모듈의 차이점을 볼 수 없습니다. 일부 차이가 너무 커서 표시할 수 없을 수도 있습니다. 자세한 내용은 의 소스 리포지토리 할당량 CodeCatalyst 및 파일 보기 단원을 참조하세요.

10. 이 풀 리퀘스트에 대한 품질 보고서를 보려면 보고서를 선택합니다.

> **Note**
>
> 풀 리퀘스트에 보고서를 표시하려면 보고서를 생성하도록 워크플로를 구성해야 합니다. 자세한 내용은 워크플로를 사용한 테스트 단원을 참조하십시오.

풀 리퀘스트를 승인 규칙과 병합하기 위한 요구 사항 관리

풀 리퀘스트를 생성할 때 개별 풀 리퀘스트에 필수 또는 선택적 검토자를 추가하도록 선택할 수 있습니다. 하지만 특정 대상 브랜치에 병합할 때 모든 풀 리퀘스트가 충족해야 하는 요구 사항을 생성할 수도 있습니다. 이러한 요구 사항을 승인 규칙이라고 합니다. 승인 규칙은 저장소의 브랜치에 대해 구성됩니다. 대상 브랜치에 승인 규칙이 구성된 풀 리퀘스트를 생성하는 경우, 풀 요청을 해당 브랜치에 병합하려면 먼저 필요한 검토자의 승인과 함께 해당 규칙에 대한 요구 사항도 충족해야 합니다. 승인 규칙을 생성하면 기본 브랜치와 같은 브랜치에 대한 병합에 대한 품질 표준을 유지하는 데 도움이 될 수 있습니다.

소스 리포지토리의 기본 분기에 적용되는 승인 규칙은 다른 분기에 적용되는 승인 규칙과 약간 다르게 작동합니다. 기본 브랜치에 적용된 모든 규칙은 기본 브랜치로 지정한 모든 브랜치에 자동으로 적용됩니다. 이전에 기본 브랜치로 설정된 브랜치에도 규칙이 계속 적용됩니다.

승인 규칙을 만들 때는 현재와 미래에 프로젝트 사용자가 해당 규칙을 어떻게 충족시킬지 고려해야 합니다. 예를 들어 프로젝트에 6명의 사용자가 있고 대상 브랜치에 병합하기 전에 5번의 승인을 받아야 하는 승인 규칙을 만들면 풀 요청을 생성한 사람을 제외한 모든 사람이 풀 요청을 승인해야 병합되는 규칙이 효과적으로 생성된 것입니다.

> **Note**
>
> 프로젝트에서 승인 규칙을 생성하고 관리하려면 프로젝트 관리자 역할이 있어야 합니다. CodeCatalyst 연결된 저장소에는 승인 규칙을 만들 수 없습니다.

승인 규칙을 삭제할 수는 없지만 승인이 0건만 요구되도록 업데이트할 수 있습니다. 이렇게 하면 사실상 규칙이 제거됩니다.

풀 리퀘스트의 대상 브랜치에 대한 승인 규칙을 보고 편집하려면

1. 리포지토리가 있는 프로젝트로 이동합니다.
2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.

   승인 규칙을 보려는 리포지토리를 선택합니다.
3. 리포지토리의 개요 페이지에서 Branch를 선택합니다.
4. 승인 규칙 열에서 보기를 선택하여 리포지토리의 각 분기에 대한 규칙의 상태를 확인합니다.

최소 승인 수에서 이 수는 풀 리퀘스트를 해당 브랜치에 병합하기 위해 필요한 승인 수에 해당합니다.

5. 승인 규칙을 생성하거나 변경하려면 설정 관리를 선택합니다. 소스 리포지토리의 설정 페이지에 있는 승인 규칙에서 편집을 선택합니다.

> **Note**
>
> 승인 규칙을 편집하려면 프로젝트 관리자 역할이 있어야 합니다.

6. Branch의 드롭다운 목록에서 승인 규칙을 구성하려는 지점의 이름을 선택합니다. 최소 승인 수에 숫자를 입력한 다음 저장을 선택합니다.

풀 리퀘스트 검토

Amazon CodeCatalyst 콘솔을 사용하여 풀 리퀘스트에 포함된 변경 사항을 공동으로 검토하고 이에 대해 의견을 제시할 수 있습니다. 소스 브랜치와 대상 브랜치의 차이 또는 풀 리퀘스트 수정 간의 차이에 따라 개별 코드 라인에 설명을 추가할 수 있습니다. 다른 사용자가 남긴 피드백을 빠르게 이해하는 데 도움이 되도록 풀 요청의 코드 변경 사항에 대해 남긴 주석을 요약하여 작성할 수 있습니다. 코드 작업을 위한 개발 환경을 만들도록 선택할 수도 있습니다.

> **Note**
>
> > **Note**
> >
> > Amazon Bedrock 제공: 자동 악용 탐지 AWS 기능을 구현합니다. 나를 위한 설명 작성, 콘텐츠 요약 생성, 작업 추천, Amazon Q를 사용하여 프로젝트에 기능 생성 또는 추가, Amazon Q Developer Agent의 소프트웨어 개발 기능에 대한 Amazon Q 이슈 할당 기능이 Amazon Bedrock에 구축되어 있기 때문에 사용자는 Amazon Bedrock에 구현된 제어 기능을 최대한 활용하여 안전, 보안 및 인공 지능 (AI) 의 책임 있는 사용을 강화할 수 있습니다.

> **Tip**
>
> 프로필의 일부로 이메일을 수신할 풀 리퀘스트 이벤트를 구성할 수 있습니다. 자세한 내용은 에서 Slack 및 이메일 알림 보내기 CodeCatalyst 단원을 참조하십시오.

풀 리퀘스트는 풀 리퀘스트의 수정과 풀 리퀘스트를 생성할 때 대상 브랜치의 팁이었던 커밋 간의 차이를 보여줍니다. 이를 머지 베이스라고 합니다. Git의 차이점 및 병합 기준에 대한 자세한 내용은 Git git-merge-base설명서를 참조하십시오.

> **Tip**
>
> 콘솔에서 작업할 때, 특히 풀 리퀘스트를 연 지 꽤 되었다면, 풀 리퀘스트를 검토하기 전에 풀 리퀘스트에 대한 최신 버전을 사용할 수 있도록 브라우저를 새로고침해 보세요.

CodeCatalyst 콘솔에서 풀 리퀘스트를 검토하려면

1. 프로젝트로 이동합니다.
2. 다음 중 하나를 수행하여 풀 리퀘스트로 이동하십시오.
   - 풀 리퀘스트가 프로젝트 페이지에 나열되어 있는 경우 목록에서 선택하십시오.
   - 풀 리퀘스트가 프로젝트 페이지에 나열되어 있지 않은 경우 모두 보기를 선택합니다. 필터와 정렬을 사용하여 풀 요청을 찾은 다음 목록에서 선택합니다.
   - 탐색 창에서 코드를 선택한 다음 풀 요청을 선택합니다.
3. 목록에서 검토하려는 풀 리퀘스트를 선택합니다. 필터 막대에 풀 요청 이름의 일부를 입력하여 풀 요청 목록을 필터링할 수 있습니다.
4. 개요에서 풀 요청의 이름과 제목을 검토할 수 있습니다. 풀 리퀘스트 자체에 남긴 댓글을 작성하고 볼 수 있습니다. 또한 워크플로 실행, 연결된 이슈, 검토자, 풀 리퀘스트 작성자, 실행 가능한 병합 전략에 대한 정보를 포함하여 풀 요청의 세부 정보를 볼 수 있습니다.

   > **Note**
   >
   > 특정 코드 줄에 남긴 코멘트는 변경사항에 표시됩니다.

5. (선택 사항) 전체 풀 리퀘스트에 적용되는 코멘트를 추가하려면 풀 리퀘스트에 대한 코멘트를 펼친 다음 코멘트 생성을 선택합니다.

6. (선택 사항) 이 풀 리퀘스트의 수정 사항에 대해 남긴 모든 댓글의 요약을 보려면 댓글 요약 만들기를 선택합니다.

> **Note**
>
> 이 기능을 사용하려면 해당 공간에 제너레이티브 AI 기능을 활성화해야 하며, 이 기능은 미국 서부 (오레곤) 지역에서만 사용할 수 있습니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.

7. 변경사항에서 대상 브랜치와 풀 리퀘스트의 최신 수정 버전 간의 차이점을 확인할 수 있습니다. 수정본이 두 개 이상인 경우, 버전 간의 차이를 비교하는 버전을 변경할 수 있습니다. 수정에 대한 자세한 내용은 을 참조하십시오. 개정

> **Tip**
>
> Files changed에서 풀 리퀘스트에서 변경 사항이 있는 파일 수와 풀 리퀘스트에서 이에 대한 코멘트가 있는 파일을 빠르게 확인할 수 있습니다. 폴더 옆에 표시되는 코멘트 수는 해당 폴더 내 파일에 달린 코멘트 수를 나타냅니다. 폴더를 확장하면 폴더 내 각 파일에 대한 댓글 수를 볼 수 있습니다.

8. 차이점이 표시되는 방식을 변경하려면 통합과 분할 중에서 선택하십시오.
9. 풀 리퀘스트의 라인에 코멘트를 추가하려면 코멘트를 달고 싶은 라인으로 이동하세요. 해당 라인에 표시되는 코멘트 아이콘을 선택하고 코멘트를 입력한 다음 저장을 선택합니다.
10. 풀 리퀘스트의 수정 간 변경 또는 출처 및 대상 분기 간의 변경 사항을 보려면 비교에서 사용 가능한 옵션 중에서 선택하십시오. 수정본의 라인에 대한 주석은 해당 수정본에 보존됩니다.
11. 풀 리퀘스트 트리거에 대한 코드 커버리지 보고서를 생성하도록 워크플로를 구성한 경우 관련 풀 요청에서 라인 및 브랜치 커버리지 결과를 볼 수 있습니다. 코드 커버리지 결과를 숨기려면 코드 커버리지 숨기기를 선택합니다. 자세한 내용은 코드 범위 보고서 단원을 참조하십시오.
12. 풀 리퀘스트의 코드를 변경하려면 풀 리퀘스트에서 개발 환경을 만들면 됩니다. 개발 환경 생성을 선택합니다. 선택적으로 개발 환경의 이름을 추가하거나 구성을 편집한 다음 [Create] 를 선택합니다.
13. 보고서에서 이 풀 리퀘스트의 품질 보고서를 볼 수 있습니다. 개정이 두 개 이상인 경우 두 개정의 차이를 비교하는 버전을 변경할 수 있습니다. 보고서를 이름, 상태, 워크플로, 작업 및 유형별로 필터링할 수 있습니다.

> **Note**
>
> 풀 리퀘스트에 보고서를 표시하려면 보고서를 생성하도록 워크플로를 구성해야 합니다. 자세한 내용은 작업의 품질 보고서 구성 단원을 참조하십시오.

14. 특정 보고서를 보려면 목록에서 해당 보고서를 선택하십시오. 자세한 내용은 워크플로를 사용한 테스트 단원을 참조하십시오.
15. 이 풀 리퀘스트의 검토자로 등재되어 있고 변경 사항을 승인하려면 최신 수정 버전을 보고 있는지 확인한 다음 승인을 선택합니다.

> **Note**
>
> 모든 필수 검토자가 풀 리퀘스트를 승인해야만 풀 리퀘스트를 병합할 수 있습니다.

풀 리퀘스트 업데이트

풀 리퀘스트를 업데이트하여 다른 프로젝트 멤버가 코드를 더 쉽게 검토할 수 있도록 할 수 있습니다. 풀 리퀘스트를 업데이트하여 리뷰어, 이슈에 대한 링크, 풀 리퀘스트의 제목 또는 설명을 변경할 수 있습니다. 예를 들어, 풀 리퀘스트의 필수 검토자를 변경하여 휴가 중인 사람을 삭제하고 다른 사람을 추가할 수 있습니다. 또한 공개 풀 요청의 소스 브랜치로 커밋을 푸시하여 추가 코드 변경으로 풀 리퀘스트를 업데이트할 수도 있습니다. 소스 리포지토리에 있는 풀 요청의 소스 브랜치로 푸시할 CodeCatalyst 때마다 수정 내용이 생성됩니다. 프로젝트 멤버는 풀 리퀘스트에서 수정 버전 간의 차이점을 확인할 수 있습니다.

풀 리퀘스트를 위해 리뷰어에게 업데이트하기

1. 풀 리퀘스트의 리뷰어를 업데이트하려는 프로젝트로 이동합니다.
2. 프로젝트 페이지의 풀 리퀘스트 열기에서 리뷰어를 업데이트하려는 풀 리퀘스트를 선택합니다. 또는 탐색 창에서 코드를 선택하고 풀 요청을 선택한 다음 업데이트하려는 풀 요청을 선택합니다.
3. (선택 사항) 개요의 풀 요청 세부 정보 영역에서 더하기 기호를 선택하여 필수 또는 선택적 검토자를 추가합니다. 선택적 또는 필수 검토자에서 제외하려면 검토자 옆에 있는 X를 선택합니다.
4. (선택 사항) 개요의 풀 리퀘스트 세부 정보 영역에서 이슈 연결을 선택하여 이슈를 풀 리퀘스트에 연결한 다음 목록에서 이슈를 선택하거나 해당 ID를 입력합니다. 이슈의 연결을 해제하려면 연결 해제하려는 이슈 옆의 연결 해제 아이콘을 선택하세요.

풀 리퀘스트의 소스 브랜치에 있는 파일 및 코드를 업데이트하려면

1. 여러 파일을 업데이트하려면 개발 환경을 만들거나 리포지토리와 소스 브랜치를 복제하고 Git 클라이언트 또는 통합 개발 환경 IDE () 을 사용하여 소스 브랜치의 파일을 변경합니다. 변경 내용을 소스 리포지토리의 소스 브랜치에 커밋하고 푸시하면 변경 내용이 포함된 pull 요청이 자동으로 업데이트됩니다. CodeCatalyst 자세한 내용은 소스 리포지토리 복제 및 Amazon에서 커밋을 사용한 소스 코드의 변경 사항 이해 CodeCatalyst 단원을 참조하세요.
2. 소스 브랜치의 개별 파일을 업데이트하려면 Git 클라이언트를 사용하거나 여러 파일의 IDE 경우처럼 사용할 수 있습니다. CodeCatalyst콘솔에서 직접 편집할 수도 있습니다. 자세한 내용은 파일 편집 단원을 참조하십시오.

풀 리퀘스트의 제목과 설명을 업데이트하려면

1. 풀 리퀘스트의 제목이나 설명을 업데이트하려는 프로젝트로 이동합니다.
2. 프로젝트 페이지에는 풀 리퀘스트를 생성한 사람, 풀 리퀘스트의 브랜치가 들어 있는 저장소, 풀 리퀘스트가 언제 생성되었는지에 대한 정보를 포함하여 진행 중인 풀 리퀘스트가 표시됩니다. 오픈 풀 리퀘스트 보기를 소스 리포지토리별로 필터링할 수 있습니다. 목록에서 변경하려는 풀 요청을 선택합니다.
3. 모든 풀 리퀘스트를 보려면 모두 보기를 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 풀 요청을 선택합니다. 필터 상자 또는 정렬 함수를 사용하여 변경하려는 풀 요청을 찾은 다음 선택합니다.
4. 개요에서 편집을 선택합니다.
5. 제목 또는 설명을 변경한 다음 저장을 선택합니다.

풀 리퀘스트 병합

코드를 검토하고 필요한 검토자가 모두 승인한 후에는 지원되는 병합 전략 (예: 빠른 전달) 을 사용하여 CodeCatalyst 콘솔에서 풀 리퀘스트를 병합할 수 있습니다. CodeCatalyst 콘솔에서 지원되는 모든 병합 전략을 모든 풀 요청에 대한 선택 항목으로 사용할 수 있는 것은 아닙니다. CodeCatalyst 병합을 평가하여 콘솔에서 사용할 수 있고 소스 브랜치를 대상 브랜치에 병합할 수 있는 병합 전략 중에서만 선택할 수 있도록 합니다. 로컬 컴퓨터 또는 개발 환경에서 git merge 명령을 실행하여 소스 브랜치를 대상 브랜치에 병합하여 pull 요청을 원하는 Git 병합 전략과 병합할 수도 있습니다. 그런 다음 대상 브랜치의 변경 내용을 의 소스 리포지토리로 푸시할 수 있습니다. CodeCatalyst

> **Note**
>
> 브랜치를 병합하고 Git에서 변경 사항을 푸시해도 pull 요청이 자동으로 종료되지는 않습니다.

프로젝트 관리자 역할이 있는 경우 승인 및 승인 규칙에 대한 모든 요구 사항을 아직 충족하지 않은 풀 요청을 병합하도록 선택할 수도 있습니다.

풀 리퀘스트 병합 (콘솔)

소스 브랜치와 대상 브랜치 간에 병합 충돌이 없고 필요한 모든 검토자가 풀 요청을 승인한 경우 CodeCatalyst 콘솔에서 풀 요청을 병합할 수 있습니다. 충돌이 있거나 병합을 완료할 수 없는 경우 병합 버튼이 비활성화되고 병합할 수 없음 레이블이 표시됩니다. 이 경우 필요한 승인자의 승인을 받고, 필요한 경우 충돌을 로컬에서 해결하고, 해당 변경 사항을 푸시해야 병합할 수 있습니다. 풀 리퀘스트를 병합하면 풀 리퀘스트 생성자와 필수 또는 선택적 검토자에게 자동으로 이메일이 발송됩니다. 풀 리퀘스트와 연결된 이슈의 상태는 자동으로 종료되거나 변경되지 않습니다.

> **Tip**
>
> 프로필의 일부로 이메일을 수신할 풀 리퀘스트 이벤트를 구성할 수 있습니다. 자세한 내용은 에서 Slack 및 이메일 알림 보내기 CodeCatalyst 단원을 참조하십시오.

풀 리퀘스트를 병합하려면

1. 풀 리퀘스트를 병합하려는 프로젝트로 이동합니다.
2. 프로젝트 페이지의 오픈 풀 리퀘스트에서 병합하려는 풀 리퀘스트를 선택합니다. 풀 리퀘스트가 보이지 않는 경우 모든 풀 리퀘스트 보기를 선택한 다음 목록에서 선택하십시오. 또는 탐색 창에서 코드를 선택하고 풀 요청을 선택한 다음 병합하려는 풀 요청을 선택합니다. 병합을 선택합니다.
3. 풀 리퀘스트에 사용할 수 있는 병합 전략 중에서 선택하십시오. 풀 리퀘스트를 병합한 후 소스 브랜치를 삭제하는 옵션을 선택하거나 선택 취소한 다음 병합을 선택할 수도 있습니다.

   > **Note**
   >
   > 병합 버튼이 비활성화되어 있거나 병합할 수 없음 레이블이 표시되는 경우 필수 검토자가 아직 풀 요청을 승인하지 않았거나 콘솔에서 풀 요청을 병합할 수 없는 것입니다. CodeCatalyst 풀 리퀘스트를 승인하지 않은 리뷰어는 개요의 풀 리퀘스트 세부 정보 영역에 시계 아이콘으로 표시됩니다. 필요한 모든 검토자가 풀 요청을 승인했지만 병합 버튼이

여전히 비활성 상태인 경우 병합 충돌이 발생할 수 있습니다. 밑줄이 그어진 병합할 수 없음 레이블을 선택하면 풀 요청을 병합할 수 없는 이유에 대한 자세한 내용을 볼 수 있습니다. 개발 환경 또는 CodeCatalyst 콘솔에서 대상 브랜치의 병합 충돌을 해결한 다음 pull 요청을 병합하거나 충돌을 해결하고 로컬에서 병합한 다음 병합이 포함된 커밋을 소스 브랜치에 푸시할 수 있습니다. CodeCatalyst 자세한 내용은 풀 리퀘스트 병합 (Git) 및 Git 설명서를 참조하십시오.

병합 요구 사항 재정의

프로젝트 관리자 역할이 있는 경우 필수 승인 및 승인 규칙에 대한 모든 요구 사항을 아직 충족하지 않은 풀 요청을 병합하도록 선택할 수 있습니다. 이를 풀 리퀘스트의 요구 사항 무효화라고 합니다. 필수 검토자가 없는 경우 또는 특정 풀 리퀘스트를 승인 규칙을 신속하게 충족할 수 없는 브랜치에 병합해야 하는 긴급한 필요가 발생하는 경우 이 방법을 선택할 수 있습니다.

풀 리퀘스트를 병합하려면

1. 요구 사항을 무시하고 병합하려는 풀 요청에서 병합 버튼 옆에 있는 드롭다운 화살표를 선택합니다. 승인 요건 재정의를 선택합니다.
2. 무시 이유에서 승인 규칙 및 필수 검토자 요구 사항을 충족하지 않고 이 pull 요청을 병합하는 이유를 자세히 입력하십시오. 이는 선택 사항이지만 적극 권장됩니다.
3. 선택적으로 병합 전략을 선택하거나 기본값을 그대로 사용할 수 있습니다. 자동 생성된 커밋 메시지를 더 자세한 내용으로 업데이트하도록 선택할 수도 있습니다.
4. 병합 시 소스 브랜치를 삭제하는 옵션을 선택하거나 선택 취소합니다. 풀 리퀘스트를 병합하기 위한 요구 사항을 재정의할 때는 다른 팀 구성원과 함께 결정을 검토할 수 있을 때까지 소스 브랜치를 보존하는 것이 좋습니다.
5. 병합을 선택합니다.

풀 리퀘스트 병합 (Git)

Git은 브랜치 병합 및 관리를 위한 다양한 옵션을 지원합니다. 다음 명령은 사용할 수 있는 몇 가지 옵션입니다. 자세한 내용은 Git 웹 사이트에서 제공되는 설명서를 참조하십시오. 변경 사항을 병합하고 푸시한 후에는 pull 요청을 수동으로 닫으십시오. 자세한 내용은 풀 리퀘스트 닫기 단원을 참조하십시오.

브랜치 병합을 위한 일반적인 Git 명령

로컬 리포지토리의 소스 브랜치에서 로컬 리포지토리의 대상 브랜치로 변경 내용을 병합합니다.	`git checkout destination-branch-name`
	
	`git merge source-branch-name`

빠른 병합을 지정하여 소스 브랜치를 대상 브랜치에 병합합니다. 이렇게 하면 브랜치가 병합되고 대상 브랜치 포인터가 소스 브랜치의 끝으로 이동합니다.	`git checkout destination-branch-name`
	
	`git merge --ff-only  source-branch-name`

스쿼시 병합을 지정하여 소스 브랜치를 대상 브랜치에 병합합니다. 이렇게 하면 소스 브랜치의 모든 커밋이 대상 브랜치의 단일 병합 커밋으로 합쳐집니다.	`git checkout destination-branch-name`
	
	`git merge --squash source-branch-name`

3방향 병합을 지정하여 소스 브랜치를 대상 브랜치에 병합합니다. 그러면 병합 커밋이 생성되고 소스 브랜치의 개별 커밋이 대상 브랜치에 추가됩니다.	`git checkout destination-branch-name`
	
	`git merge --no-ff source-branch-name`

로컬 리포지토리에서 소스 브랜치를 삭제합니다. 이는 대상 브랜치에 병합하고 변경 내용을 소스 리포지토리로 푸시한 후 로컬 리포지토리를 정리할 때 유용합니다.	`git branch -d source-branch-name`

로컬 리포지토리의 지정된 원격 리포지토리 닉네임을 사용하여 원격 리포지토리의 소스 브랜치 (소스 리포지토리 CodeCatalyst) 를 삭제합니다. (콜론(`:`) 사용에 주의하세요.) 또는 명령의 `--delete` 일부로 지정할 수도 있습니다.	`git push remote-name  :source-branch-name`
	
	`git push remote-name  --delete source-branch-name`

풀 리퀘스트 닫기

풀 리퀘스트를 종결로 표시할 수 있습니다. 이렇게 하면 풀 리퀘스트가 병합되지는 않지만, 조치가 필요한 풀 리퀘스트와 더 이상 관련이 없는 풀 리퀘스트를 판단하는 데 도움이 될 수 있습니다. 변경 내용을 더 이상 병합할 계획이 없거나 변경 내용이 다른 풀 리퀘스트에 병합된 경우에는 풀 리퀘스트를 종료하는 것이 좋습니다.

풀 리퀘스트를 종료하면 풀 리퀘스트 생성자와 필수 또는 선택적 검토자에게 자동으로 이메일이 발송됩니다. 풀 리퀘스트와 연결된 이슈의 상태는 자동으로 변경되지 않습니다.

> **Note**
>
> 풀 리퀘스트가 종료된 후에는 다시 열 수 없습니다.

풀 리퀘스트를 종료하려면

1. 풀 리퀘스트를 종료하려는 프로젝트로 이동합니다.
2. 프로젝트 페이지에는 진행 중인 풀 리퀘스트가 표시됩니다. 종료하려는 풀 리퀘스트를 선택합니다.
3. 닫기를 선택하세요.
4. 정보를 검토한 다음 풀 리퀘스트 종료를 선택합니다.

Amazon에서 커밋을 사용한 소스 코드의 변경 사항 이해 CodeCatalyst

커밋은 리포지토리의 콘텐츠 및 변경 사항에 대한 스냅샷입니다. 사용자가 브랜치에 변경 사항을 커밋하고 푸시할 때마다 해당 정보가 저장됩니다. Git 커밋 정보에는 커밋 작성자, 변경을 커밋한 사람, 날짜 및 시간, 변경 내용이 포함됩니다. Amazon CodeCatalyst 콘솔에서 파일을 만들거나 편집할 때 유사한 정보가 자동으로 포함되지만 작성자 이름은 CodeCatalyst 사용자 이름입니다. 또한 커밋에 Git 태그를 추가하여 특정 커밋을 식별할 수 있습니다.

CodeCatalystAmazon에서는 다음을 수행할 수 있습니다.

- 브랜치의 커밋 목록 보기.
- 부모 또는 상위 커밋과 비교한 커밋의 변경 내용을 포함하여 개별 커밋을 볼 수 있습니다.

파일 및 폴더도 볼 수 있습니다. 자세한 내용은 Amazon에서 소스 코드 파일 관리 CodeCatalyst 단원을 참조하십시오.

주제

- 브랜치에 대한 커밋 보기
- 커밋 표시 방식 변경 (CodeCatalyst콘솔)

브랜치에 대한 커밋 보기

콘솔에서 브랜치의 커밋을 검토하여 브랜치의 변경 기록을 볼 수 있습니다. CodeCatalyst 이를 통해 누가 언제 브랜치를 변경했는지 파악할 수 있습니다. 특정 커밋에서 변경된 내용을 검토할 수도 있습니다.

> **Tip**
>
> 특정 파일을 변경한 커밋 기록도 볼 수 있습니다. 자세한 정보는 파일 보기 섹션을 참조하세요.

Git 클라이언트를 사용하여 커밋을 볼 수도 있습니다. 자세한 내용은 Git 설명서를 참조하십시오.

커밋을 보려면 (콘솔)

1. 커밋을 보려는 소스 리포지토리가 들어 있는 프로젝트로 이동합니다.

2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.

   브랜치에 대한 커밋을 보려는 리포지토리를 선택합니다.

3. 브랜치에 대한 가장 최근 커밋에 대한 정보를 포함하여 리포지토리의 기본 브랜치가 표시됩니다. 커밋을 선택합니다. 또는 기타를 선택한 다음 커밋 보기를 선택합니다.
4. 다른 브랜치의 커밋을 보려면 브랜치 선택기를 선택한 다음 브랜치의 이름을 선택합니다.
5. 특정 커밋에 대한 세부 정보를 보려면 커밋 제목에서 제목을 선택하세요. 부모 커밋에 대한 정보와 부모 커밋을 지정된 커밋과 비교하여 코드에서 변경한 내용을 포함하여 커밋의 세부 정보가 표시됩니다.

> **Tip**
>
> 커밋에 상위 커밋이 두 개 이상 있는 경우 부모 커밋 ID 옆의 드롭다운 아이콘을 선택하여 정보를 보고 변경 내용을 표시할 상위 커밋을 선택할 수 있습니다.

커밋 표시 방식 변경 (CodeCatalyst콘솔)

커밋 뷰에 표시되는 정보를 변경할 수 있습니다. 작성자 및 커밋 ID와 같은 열을 숨기거나 표시하도록 선택할 수 있습니다.

커밋 표시 방식 변경하기 (콘솔)

1. 커밋을 보려는 소스 리포지토리가 포함된 프로젝트로 이동합니다.
2. 프로젝트의 소스 리포지토리 목록에서 리포지토리 이름을 선택합니다. 또는 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.

   커밋을 보는 방식을 변경하려는 리포지토리를 선택합니다.
3. 브랜치에 대한 가장 최근 커밋에 대한 정보를 포함하여 리포지토리의 기본 브랜치가 표시됩니다. 커밋을 선택합니다.
4. 기어 모양 아이콘을 선택합니다.
5. 기본 설정에서 표시할 커밋 수를 선택하고 커밋 작성자, 커밋 날짜 및 커밋 ID에 대한 정보를 표시할지 여부를 선택합니다.

   > **Note**
   >
   > 정보 표시에서 커밋 제목을 숨길 수는 없습니다.
6. 변경한 후에는 저장을 선택하여 저장하거나 취소를 선택하여 취소합니다.

의 소스 리포지토리 할당량 CodeCatalyst

다음 표에는 Amazon의 소스 리포지토리 할당량 및 한도가 설명되어 있습니다. CodeCatalyst CodeCatalystAmazon의 할당량에 대한 자세한 내용은 을 참조하십시오. 에 대한 할당량 CodeCatalyst

Resource	정보
브랜치 이름	허용되는 문자 조합은 길이가 1~256자이고 저장소 내에서 고유해야 합니다. 브랜치 이름에는 다음과 같은 제한이 있습니다. • 슬래시(/) 또는 마침표(.)로 시작하거나 끝날 수 없음 • 단일 문자 @로 구성될 수 없음 • 두 개 이상의 연속 마침표(..), 슬래시(//) 또는 다음 문자 조합은 포함할 수 없음: @{ • 공백 또는 다음 문자를 포함할 수 없음: ? ^ * [\ ~ : 브랜치 이름은 참조입니다. 브랜치 이름에 대한 대부분의 제한 사항은 Git 참조 표준에 근거합니다. 자세한 내용은 Git 내부 및 을 참조하십시오. git-check-ref-format
풀 리퀘스트에 대한 의견	풀 리퀘스트는 최대 1,000개까지 가능합니다.
커밋 메시지	최대 1024자까지 입력할 수 있습니다.

파일 경로	1~4,096자의 허용되는 문자 조합이면 됩니다. 파일 경로는 파일과 파일의 정확한 위치를 지정하는 명확한 이름이어야 합니다. 파일 경로는 깊이가 디렉터리 20개를 초과할 수 없습니다. 또한 파일 경로는 다음과 같아서는 안 됩니다.
	• 빈 문자열 포함
	• 상대적 파일 경로
	• 후행 슬래시나 백슬래시로 끝나는
	
	`/./`
	
	`/../`

Resource	정보
	// • 문자 조합 포함 파일 이름 및 경로는 정규화된 이름과 경로여야 합니다. 로컬 컴퓨터의 파일 이름과 경로는 해당 운영 체제에 대한 표준을 따라야 합니다. 리포지토리의 파일 경로를 지정할 때는 Amazon Linux 용 표준을 사용하십시오.
파일 크기	CodeCatalyst콘솔 사용 시 개별 파일의 경우 최대 6MB입니다.
콘솔에서 볼 수 있는 파일 크기 CodeCatalyst	CodeCatalyst콘솔 사용 시 개별 파일의 경우 최대 6MB입니다.
Git BLOB 크기	최대 2GB Note 단일 커밋의 모든 파일 수 또는 총 파일 크기에 대한 제한은 없습니다. 단, 메타데이터는 6MB를 초과할 수 없고 단일 BLOB은 2GB를 초과할 수 없습니다. 하지만 한 개의 큰 커밋보다는 작은 커밋을 여러 개 만드는 것이 좋습니다.

Resource	정보
커밋에 대한 메타데이터	커밋의 결합된 메타데이터는 최대 6MB입니다 (예: 작성자 정보, 날짜, 상위 커밋 목록 및 커밋 메시지의 조합). **Note** 단일 커밋의 모든 파일 수 또는 총 파일 크기에 대한 제한은 없습니다. 단, 데이터는 20MB를 초과할 수 없고 개별 파일은 6MB를 초과할 수 없으며 단일 BLOB은 2GB를 초과할 수 없습니다.
풀 리퀘스트에 연결할 수 있는 CodeCatalyst 이슈 수	50
풀 리퀘스트에 연결할 수 있는 Jira 이슈 수	50
스페이스에 있는 미해결 풀 리퀘스트 수	아마존 CodeCatalyst 공간 1개당 최대 1,000개
스페이스에 있는 총 풀 리퀘스트 수	아마존 CodeCatalyst 스페이스는 최대 10,000개입니다.
단일 푸시의 참조 수	생성, 삭제, 업데이트를 포함하여 최대 4,000개. 리포지토리의 최대 참조 수는 제한이 없습니다.
스페이스에 있는 리포지토리 수	아마존 CodeCatalyst 공간 1개당 최대 5,000개
리포지토리 설명	0~1,000자의 문자 조합이면 됩니다. 리포지토리 설명은 선택 사항입니다.

Resource	정보
리포지토리 이름	리포지토리 이름은 프로젝트에서 고유해야 합니다. 1~100자 사이의 문자, 숫자, 마침표, 밑줄 및 대시 조합을 포함할 수 있습니다. 이름은 대소문자를 구분하지 않습니다. 리포지토리 이름은.git로 끝날 수 없고, 공백을 포함할 수 없으며, 다음 문자를 포함할 수 없습니다. ! ? @ # $ % ^ & * ( ) + = { } [ ] | \ / > < ~ ` ' " ; :
리포지토리 크기	저장소 크기는 공간의 전체 저장소 한도에 영향을 받습니다. 자세한 내용은 요금 및 소스 리포지토리 관련 문제 해결 섹션을 참조하세요.
풀 리퀘스트의 리뷰어들	풀 리퀘스트의 총 리뷰어 수는 최대 100명 (선택 또는 필수) 입니다.
풀 리퀘스트에 대한 서면 요약	풀 리퀘스트의 최대 요약 작성 개수는 해당 공간의 청구 등급에 따라 다릅니다. 자세한 설명은 요금을 참조하세요.

의 개발 환경을 사용하여 코드 작성 및 수정 CodeCatalyst

개발 환경은 클라우드 기반 개발 환경입니다. CodeCatalystAmazon에서는 Dev Environments를 사용하여 프로젝트의 소스 리포지토리에 저장된 코드를 작업합니다. 개발 환경을 만들 때 다음과 같은 몇 가지 옵션을 사용할 수 있습니다.

- 지원되는 통합 개발 환경 () 으로 코드 작업을 수행할 CodeCatalyst 수 있는 프로젝트별 개발 환경을 만드십시오. IDE
- 빈 개발 환경을 만들고, 소스 리포지토리에서 코드를 복제하고, 지원되는 환경에서 해당 코드를 작업하세요. IDE
- 에서 개발 환경을 만들고 소스 리포지토리를 개발 환경에 복제합니다. IDE

devfile은 개발 환경을 표준화하는 개방형 표준 YAML 파일입니다. 즉, 이 파일은 개발 환경에 필요한 개발 도구를 코드화합니다. 따라서 개발 환경을 빠르게 설정하고, 프로젝트 간에 전환하고, 팀 구성원 간에 개발 환경 구성을 복제할 수 있습니다. 개발 환경은 특정 프로젝트의 코딩, 테스트 및 디버깅에 필요한 모든 도구를 구성하는 개발 파일을 사용하기 때문에 로컬 개발 환경을 만들고 유지 관리하는 데 드는 시간을 최소화합니다.

개발 환경에 포함된 프로젝트 도구와 애플리케이션 라이브러리는 프로젝트의 소스 저장소에 있는 devfile에 의해 정의됩니다. 소스 리포지토리에 devfile이 없는 경우 는 기본 devfile을 CodeCatalyst 자동으로 적용합니다. 이 기본 devfile에는 가장 자주 사용되는 프로그래밍 언어 및 프레임워크용 도구가 포함되어 있습니다. 블루프린트를 사용하여 프로젝트를 만든 경우 에서 자동으로 devfile을 생성합니다. CodeCatalyst 개발 파일에 대한 자세한 내용은 https://devfile.io 을 참조하십시오.

개발 환경을 만든 후에는 사용자만 액세스할 수 있습니다. 개발 환경에서는 지원되는 소스 리포지토리의 코드를 보고 작업할 수 있습니다. IDE

기본적으로 개발 환경은 2코어 프로세서, 4GB, 16GB의 RAM 영구 스토리지로 구성되어 있습니다. 스페이스 관리자 권한이 있는 경우 다양한 개발 환경 구성 옵션을 사용하도록 스페이스의 청구 등급을 변경하고 컴퓨팅 및 스토리지 한도를 관리할 수 있습니다.

주제

- Dev Environment 삭제
- 를 사용하여 개발 환경에 연결 SSH
- 개발 환경을 위한 개발 파일 구성
- 개발 VPC 환경에 연결 연결
- 의 개발 환경에 대한 할당량 CodeCatalyst

Dev Environment 생성

Dev Environment는 다양한 방법으로 만들 수 있습니다.

- 개요, 개발 환경 또는 CodeCatalyst 소스 리포지토리 페이지에서 소스 리포지토리 또는 링크된 소스 리포지토리를 CodeCatalyst 사용하여 개발 환경을 생성합니다.
- 개발 환경 페이지에서 소스 리포지토리에 연결되지 CodeCatalyst 않은 빈 개발 환경을 만드세요.
- 원하는 대로 개발 환경을 만들고 모든 소스 리포지토리를 개발 환경에 복제하세요IDE.

> **Important**
>
> Active Directory가 ID 공급자로 사용되는 공간의 사용자는 개발 환경을 사용할 수 없습니다. 자세한 내용은 SSO (Single Sign-On) 계정을 CodeCatalyst 사용하여 로그인한 상태에서는 개발 환경을 만들 수 없습니다. 단원을 참조하십시오.

리포지토리 브랜치당 하나의 개발 환경을 만들 수 있습니다. 프로젝트에는 여러 리포지토리가 존재할 수 있습니다. 생성한 개발 환경은 사용자 CodeCatalyst 계정으로만 관리할 수 있지만, 개발 환경을 열고 지원되는 모든 환경에서 작업할 수 있습니다. IDEs 에서 개발 환경을 사용하려면 이 AWS Toolkit 설치되어 있어야 합니다. IDE 자세한 내용은 개발 환경을 위해 지원되는 통합 개발 환경 단원을 참조하십시오. 기본적으로 개발 환경은 2코어 프로세서, 4GB, 16GB의 영구 RAM 스토리지로 만들어집니다.

> **Note**
>
> 소스 리포지토리와 연결된 개발 환경을 만든 경우 리소스 열에는 항상 이 개발 환경을 만들 때 지정한 분기가 표시됩니다. 이는 다른 브랜치를 생성하거나, 개발 환경 내에서 다른 브랜치로 전환하거나, 추가 리포지토리를 복제하는 경우에도 적용됩니다. 빈 개발 환경을 만든 경우 리소스 열은 비어 있습니다.

개발 환경을 위해 지원되는 통합 개발 환경

다음과 같은 지원되는 통합 개발 환경 () IDEs 과 함께 개발 환경을 사용할 수 있습니다.

- AWS Cloud9
- JetBrains IDEs
  - 인텔리J 얼티밋 IDEA
  - GoLand
  - PyCharm전문가용
- 비주얼 스튜디오 코드

에서 개발 환경 만들기 CodeCatalyst

에서 CodeCatalyst 개발 환경 작업을 시작하려면 AWS 빌더 ID 또는 를 사용하여 인증하고 로그인하세요. SSO

브랜치에서 개발 환경을 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 생성하려는 스페이스로 이동합니다.
3. 탐색 창에서 다음 중 하나를 수행하십시오.
   - 개요를 선택한 다음 내 개발 환경 섹션으로 이동합니다.
   - 코드를 선택한 다음 개발 환경을 선택합니다.
   - 코드를 선택하고 소스 리포지토리를 선택한 다음 개발 환경을 만들려는 리포지토리를 선택합니다.
4. 개발 환경 생성을 선택합니다.
5. 드롭다운 IDE 메뉴에서 지원되는 항목을 선택합니다. 자세한 내용은 개발 환경을 위해 지원되는 통합 개발 환경 섹션을 참조하세요.
6. 리포지토리 복제를 선택합니다.
7. 다음 중 하나를 수행합니다.
   a. 복제할 리포지토리를 선택하고, 기존 브랜치에서 작업을 선택한 다음 기존 브랜치 드롭다운 메뉴에서 브랜치를 선택합니다.

> **Note**
>
> 타사 리포지토리를 선택하는 경우 기존 브랜치에서 작업해야 합니다.

b. 복제할 리포지토리를 선택하고, 새 브랜치에서 작업을 선택하고, 브랜치 이름 필드에 브랜치 이름을 입력하고, 다음에서 브랜치 생성 드롭다운 메뉴에서 새 브랜치를 만들 브랜치를 선택합니다.

> **Note**
>
> 소스 리포지토리 페이지 또는 특정 소스 리포지토리에서 개발 환경을 만드는 경우 리포지토리를 선택할 필요가 없습니다. 개발 환경은 소스 리포지토리 페이지에서 선택한 소스 리포지토리에서 생성됩니다.

8. (선택 사항) 별칭 - 선택 사항에 개발 환경의 별칭을 입력합니다.
9. (선택 사항) 개발 환경 구성 편집 버튼을 선택하여 개발 환경의 컴퓨팅, 스토리지 또는 타임아웃 구성을 편집합니다.
10. (선택 사항) Amazon Virtual Private Cloud (AmazonVPC) - 선택 사항인 경우 드롭다운 메뉴에서 개발 환경에 연결할 VPC 연결을 선택합니다.

    공간에 기본값이 VPC 설정되어 있으면 개발 환경이 해당 공간에 연결되어 실행됩니다. VPC 다른 연결을 연결하여 이를 무시할 수 있습니다. VPC 또한 VPC -connected 개발 환경은 지원하지 않는다는 점에 유의하세요. AWS Toolkit

    사용하려는 VPC 연결이 목록에 없다면 프로젝트에서 허용되지 않는 AWS 계정 연결이 포함되어 있기 때문일 수 있습니다. 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 프로젝트 제한 계정 연결 구성을 참조하십시오.

    > **Note**
    >
    > VPC연결이 있는 개발 환경을 생성하면 내부에 새 네트워크 인터페이스가 생성됩니다. VPC CodeCatalyst 관련 VPC 역할을 사용하여 이 인터페이스와 상호 작용합니다. 또한 IPv4 CIDR 블록이 `172.16.0.0/12` IP 주소 범위로 구성되어 있지 않은지 확인하십시오.

11. 생성(Create)을 선택합니다. 개발 환경이 생성되는 동안 개발 환경 상태 열에 시작 중이 표시되고, 개발 환경이 생성되면 상태 열에 실행 중이 표시됩니다.

빈 개발 환경을 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 생성하려는 스페이스로 이동합니다.
3. 탐색 창에서 다음 중 하나를 수행하십시오.
   - 개요를 선택한 다음 내 개발 환경 섹션으로 이동합니다.
   - 코드를 선택한 다음 개발 환경을 선택합니다.
4. 개발 환경 생성을 선택합니다.
5. 드롭다운 IDE 메뉴에서 지원되는 항목을 선택합니다. 자세한 내용은 개발 환경을 위해 지원되는 통합 개발 환경 섹션을 참조하세요.
6. 빈 개발 환경 만들기를 선택합니다.
7. (선택 사항) 별칭 - 선택 사항에 개발 환경의 별칭을 입력합니다.
8. (선택 사항) 개발 환경 구성 편집 버튼을 선택하여 개발 환경의 컴퓨팅, 스토리지 또는 타임아웃 구성을 편집합니다.
9. (선택 사항) Amazon Virtual Private Cloud (AmazonVPC) - 선택 사항인 경우 드롭다운 메뉴에서 개발 환경에 연결할 VPC 연결을 선택합니다.

   공간에 기본값이 VPC 설정되어 있으면 개발 환경이 해당 공간에 연결되어 실행됩니다. VPC 다른 연결을 연결하여 이를 무시할 수 있습니다. VPC 또한 VPC -connected 개발 환경은 지원하지 않는다는 점에 유의하세요. AWS Toolkit

   사용하려는 VPC 연결이 목록에 없다면 프로젝트에서 허용되지 않는 AWS 계정 연결이 포함되어 있기 때문일 수 있습니다. 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 프로젝트 제한 계정 연결 구성을 참조하십시오.

   > **Note**
   >
   > VPC연결이 있는 개발 환경을 생성하면 내부에 새 네트워크 인터페이스가 생성됩니다. VPC CodeCatalyst 관련 VPC 역할을 사용하여 이 인터페이스와 상호 작용합니다. 또한 IPv4 CIDR 블록이 `172.16.0.0/12` IP 주소 범위로 구성되어 있지 않은지 확인하십시오.

10. 생성(Create)을 선택합니다. 개발 환경이 생성되는 동안 개발 환경 상태 열에 시작 중이 표시되고, 개발 환경이 생성되면 상태 열에 실행 중이 표시됩니다.

> **Note**
>
> 처음으로 개발 환경을 만들고 여는 데 1~2분 정도 걸릴 수 있습니다.

> **Note**
>
> 에서 개발 환경을 연 후 코드에 변경 내용을 커밋하고 푸시하기 전에 디렉터리를 소스 리포지토리로 변경해야 할 수 있습니다. IDE

개발 환경 만들기 IDE

Dev Environments를 사용하여 프로젝트의 소스 리포지토리에 저장된 코드를 빠르게 작업할 수 있습니다. Dev Environments는 지원되는 통합 개발 환경 () 을 통해 프로젝트별로 완벽하게 작동하는 클라우드 개발 환경에서 즉시 코딩을 시작할 수 있으므로 개발 속도를 높입니다. IDE

CodeCatalyst 에서 작업하는 방법에 대한 자세한 내용은 다음 IDE 설명서를 참조하십시오.

- CodeCatalyst 아마존의 경우 JetBrains IDEs
- VS CodeCatalyst 코드용 아마존
- CodeCatalyst 아마존의 경우 AWS Cloud9

개발 환경 중지

개발 환경의 `/projects` 디렉터리에는 소스 리포지토리에서 가져온 파일과 개발 환경을 구성하는 데 사용되는 devfile이 저장됩니다. 개발 환경 생성 시 비어 있는 `/home` 디렉토리에는 개발 환경을 사용하는 동안 생성한 파일이 저장됩니다. 개발 환경의 `/projects` 및 `/home` 디렉터리에 있는 모든 내용은 영구적으로 저장되므로 다른 개발 환경, 리포지토리 또는 프로젝트로 전환해야 하는 경우 개발 환경에서 작업을 중단할 수 있습니다.

> **Warning**
>
> 웹 브라우저, 원격 셸 등을 비롯한 모든 인스턴스가 연결 상태를 유지하더라도 개발 환경은 제한 시간이 초과되지 않습니다. IDEs 따라서 추가 비용이 발생하지 않도록 연결된 모든 인스턴스를 닫아야 합니다.

개발 환경을 만드는 동안 Timeout 필드에서 선택한 시간 동안 유휴 상태인 경우 개발 환경이 자동으로 중지됩니다. 유휴 상태가 되기 전에 개발 환경을 중지할 수 있습니다. 개발 환경을 만들 때 타임아웃 없음을 선택한 경우 개발 환경이 자동으로 중지되지 않습니다. 대신 계속 실행됩니다.

> **Warning**
>
> 삭제된 VPC 연결과 관련된 개발 환경을 중지하면 다시 시작할 수 없습니다.

개발 환경 페이지에서 개발 환경을 중지하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 중지하려는 프로젝트로 이동합니다.
3. 탐색 창에서 코드를 선택합니다.
4. 개발 환경을 선택합니다.
5. 중지하려는 개발 환경의 라디오 버튼을 선택합니다.
6. 작업 메뉴에서 중지를 선택합니다.

> **Note**
>
> 컴퓨팅 사용은 개발 환경이 실행되는 동안에만 청구되지만, 스토리지 사용에 대해서는 개발 환경이 존재하는 전체 기간 동안 요금이 청구됩니다. 사용하지 않을 때는 개발 환경을 중지하여 컴퓨팅 요금 청구를 중단하세요.

Dev Environment 재개

개발 환경의 `/projects` 디렉터리에는 소스 리포지토리에서 가져온 파일과 개발 환경을 구성하는 데 사용되는 devfile이 저장됩니다. 개발 환경 생성 시 비어 있는 `/home` 디렉토리에는 개발 환경을 사용하는 동안 생성한 파일이 저장됩니다. 개발 환경의 `/projects` 및 `/home` 디렉터리에 있는 모든 내용은 영구적으로 저장되므로 다른 개발 환경, 리포지토리 또는 프로젝트로 전환해야 하는 경우 개발 환경에서 작업을 중단하고 나중에 개발 환경에서 작업을 재개할 수 있습니다.

개발 환경을 만드는 동안 Timeout 필드에서 선택한 시간 동안 유휴 상태인 경우 개발 환경이 자동으로 중지됩니다. 개발 환경을 유휴 상태로 AWS Cloud9 만들려면 브라우저 탭을 닫아야 합니다.

> **Note**
>
> 개발 환경을 만들 때 사용한 브랜치를 삭제해도 개발 환경은 계속 사용 가능하고 실행됩니다. 브랜치를 삭제한 개발 환경에서 작업을 재개하려면 새 브랜치를 만들고 변경 내용을 푸시하세요.

개요 페이지에서 개발 환경을 재개하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 재개하려는 프로젝트로 이동한 다음 내 개발 환경 섹션으로 이동합니다.
3. () IDE 에서 재개를 선택합니다.
   - 의 JetBrains IDEs 경우 JetBrains JetBrains-gateway 링크를 열 애플리케이션을 선택하라는 메시지가 EAP 표시되면 Gateway-를 선택합니다. 메시지가 표시되면 [링크 열기] 를 선택하여 확인합니다.
   - VS IDE Code의 경우 VS Code 링크를 열 애플리케이션을 선택하라는 메시지가 표시되면 VS Code를 선택합니다. 링크 열기를 선택하여 확인합니다.

소스 리포지토리에서 개발 환경을 재개하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 재개하려는 프로젝트로 이동합니다.
3. 탐색 창에서 [Code] 를 선택합니다.
4. 소스 리포지토리를 선택합니다.
5. 재개하려는 개발 환경이 포함된 소스 리포지토리를 선택합니다.
6. 브랜치의 드롭다운 메뉴를 보려면 브랜치 이름을 선택한 다음 브랜치를 선택합니다.
7. [개발 환경 재개] 를 선택합니다.
   - 또는 이 사이트에서 JetBrains 게이트웨이와 -gateway 링크를 열 수 있도록 허용하시겠습니까? 라는 메시지가 표시되면 [링크 열기] 를 선택하여 확인합니다. JetBrains IDEs JetBrains .
   - VS IDE Code의 경우 이 사이트에서 Visual Studio Code를 사용하여 VS Code 링크를 열 수 있도록 허용하시겠습니까? 라는 메시지가 표시되면 링크 열기를 선택하여 확인합니다. .

개발 환경 페이지에서 개발 환경을 재개하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 재개하려는 프로젝트로 이동합니다.
3. 탐색 창에서 [Code] 를 선택합니다.
4. [개발 환경] 을 선택합니다.
5. IDE열에서 개발 환경의 Resume in (IDE) 을 선택합니다.
   - 의 경우 JetBrains IDEs, 이 사이트에서 게이트웨이를 통해 JetBrains -gateway 링크를 열 수 있도록 허용하시겠습니까? 라는 메시지가 표시되면 [링크 열기] 를 선택하여 확인합니다. JetBrains .
   - VS IDE Code의 경우 이 사이트에서 Visual Studio Code를 사용하여 VS Code 링크를 열 수 있도록 허용하시겠습니까? 라는 메시지가 표시되면 링크 열기를 선택하여 확인합니다. .

> **Note**
>
> Dev Environment 재개는 몇 분 정도 걸립니다.

개발 환경 편집

를 IDE 실행하는 동안 개발 환경을 편집할 수 있습니다. 컴퓨팅 또는 비활성 제한 시간을 수정하면 변경 내용을 저장한 후 개발 환경이 다시 시작됩니다.

개발 환경을 편집하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 편집하려는 프로젝트로 이동합니다.
3. 탐색 창에서 코드를 선택합니다.
4. 개발 환경을 선택합니다.
5. 편집하려는 개발 환경을 선택합니다.
6. 편집을 선택합니다.
7. 컴퓨팅 또는 비활성 타임아웃을 원하는 대로 변경합니다.
8. 저장(Save)을 선택합니다.

Dev Environment 삭제

개발 환경에 저장된 콘텐츠 작업을 마치면 개발 환경을 삭제할 수 있습니다. 새 콘텐츠 작업을 위한 새 개발 환경을 만드세요. 개발 환경을 삭제하면 지속된 콘텐츠가 영구적으로 삭제됩니다. 개발 환경을 삭제하기 전에 코드 변경 사항을 커밋하고 개발 환경의 원본 소스 리포지토리에 푸시해야 합니다. 개발 환경을 삭제하면 개발 환경에 대한 컴퓨팅 및 스토리지 청구가 중지됩니다.

개발 환경을 삭제한 후 스토리지 할당량이 업데이트되는 데 몇 분 정도 걸릴 수 있습니다. 스토리지 할당량에 도달한 경우 이 기간 동안에는 새 개발 환경을 만들 수 없습니다.

> **Important**
>
> 개발 환경 삭제는 취소할 수 없습니다. 개발 환경을 삭제한 후에는 더 이상 복구할 수 없습니다.

개발 환경을 삭제하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 개발 환경을 삭제하려는 프로젝트로 이동합니다.
3. 탐색 창에서 코드를 선택합니다.
4. 개발 환경을 선택합니다.
5. 삭제하려는 개발 환경을 선택합니다.
6. Delete(삭제)를 선택합니다.
7. **`delete`**를 입력하여 개발 환경 삭제를 확인합니다.
8. Delete(삭제)를 선택합니다.

> **Note**
>
> 스페이스에서 VPC 연결을 삭제하기 전에 해당 연결과 관련된 개발 환경을 제거해야 합니다. VPC
> 개발 환경을 삭제하더라도 에서 네트워크 인터페이스는 삭제되지 않을 수 있습니다. VPC 필요에 따라 리소스를 정리하세요. VPC연결된 개발 환경을 삭제할 때 오류가 발생하면 오래된 연결을 분리하고 사용되지 않음을 확인한 후 삭제해야 합니다.

를 사용하여 개발 환경에 연결 SSH

를 사용하여 SSH 개발 환경에 연결하여 포트 포워딩, 파일 업로드 및 다운로드, 기타 사용과 같은 작업을 제한 없이 수행할 수 있습니다. IDEs

> **Note**
>
> IDE탭이나 창을 닫은 후 오랜 시간 SSH 동안 계속 사용하려면 개발 환경의 제한 시간을 높게 설정하여 에서 비활성 상태로 인해 중단되지 않도록 하세요. IDE

사전 조건

- 다음 운영 체제 중 하나가 필요합니다.
  - Windows 10 이상 버전, 오픈 지원 SSH
  - 맥OS 및 배쉬 버전 3 이상
  - Linux (dpkg또는 `rpm` 패키지 관리자 포함yum) 및 Bash 버전 3 이상
- AWS CLI 버전 2.9.4 이상도 필요합니다.

를 사용하여 개발 환경에 연결하려면 SSH

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 를 사용하여 SSH 개발 환경에 연결하려는 프로젝트로 이동합니다.
3. 탐색 창에서 [Code] 를 선택합니다.
4. [개발 환경] 을 선택합니다.
5. 연결하려는 실행 중인 개발 환경을 선택합니다. SSH
6. Connect via 를 SSH 선택하고 원하는 운영 체제를 선택한 후 다음을 수행하십시오.
   - 아직 실행하지 않았다면 지정된 터미널에서 첫 번째 명령을 붙여넣고 실행하십시오. 이 명령은 스크립트를 다운로드하고 로컬 환경에서 다음 수정 사항을 실행하므로 다음을 사용하여 개발 환경에 연결할 수 있습니다. SSH
     - 를 위한 세션 관리자 플러그인을 설치합니다. AWS CLI
     - 로그인을 수행할 수 있도록 AWS Config 로컬을 수정하고 CodeCatalyst 프로필을 추가합니다. SSO 자세한 내용은 AWS CLI와 함께 사용하기 위한 설정 CodeCatalyst 단원을 참조하십시오.

    - 로컬 SSH 구성을 수정하고 를 사용하여 개발 환경에 연결하는 데 필요한 구성을 추가합니다. SSH
    - SSH클라이언트가 개발 환경에 연결하는 데 사용하는 스크립트를 ~/.aws/codecatalyst-dev-env 디렉터리에 추가합니다. 이 스크립트는 AWS Systems Manager Session Manager 플러그인을 CodeCatalyst StartDevEnvironmentSession API호출하고 이를 사용하여 개발 환경과의 AWS Systems Manager 세션을 설정합니다. 이 세션을 로컬 SSH 클라이언트가 원격 개발 환경에 안전하게 연결하는 데 사용합니다.
- 두 번째 명령어를 CodeCatalyst AWS SSO 사용하여 Amazon에 로그인합니다. 이 명령은 ~/.aws/codecatalyst-dev-env 디렉터리의 스크립트가 호출할 수 있도록 자격 증명을 요청하고 검색합니다. CodeCatalyst StartDevEnvironmentSession API 이 명령은 자격 증명이 만료될 때마다 실행해야 합니다. <destination>자격 증명이 만료되었거나 이 단계에 설명된 대로 SSO 로그인을 수행하지 않은 경우 모달 (ssh) 에서 마지막 명령을 실행할 때 오류가 발생합니다.
- 세 번째 명령을 SSH 사용하여 지정된 개발 환경에 연결합니다. 이 명령의 구조는 다음과 같습니다.

  ```
  ssh codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>
  ```

  또한 이 명령을 사용하여 포트 포워딩, 파일 업로드 및 다운로드와 같이 SSH 클라이언트가 허용하는 다른 작업을 수행할 수 있습니다.

  - 포트 포워딩:

    ```
    ssh -L <local-port>:127.0.0.1:<remote-port> codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>
    ```

  - 개발 환경의 홈 디렉터리에 파일 업로드:

    ```
    scp -O </path-to-local-file> codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>:</path-to-remote-file-or-directory>
    ```

개발 환경을 위한 개발 파일 구성

devfile은 팀 전체에 걸쳐 개발 개발 환경을 사용자 지정하는 데 도움이 되는 개방형 표준입니다. devfile은 필수 개발 도구를 코드화하는 YAML 파일입니다. devfile을 구성하면 필요한 프로젝트 도구 및 애플리케이션 라이브러리를 미리 결정할 수 있으며 Amazon은 이를 개발 환경에 자동으로 CodeCatalyst 설

치합니다. devfile은 생성된 리포지토리에만 적용되며 각 리포지토리에 대해 별도의 devfile을 생성할 수 있습니다. 개발 환경은 명령과 이벤트를 지원하며 기본 범용 devfile 이미지를 제공합니다.

빈 블루프린트를 사용하여 프로젝트를 생성하는 경우 수동으로 devfile을 만들 수 있습니다. 다른 블루프린트를 사용하여 프로젝트를 생성하는 경우, 자동으로 devfile이 CodeCatalyst 생성됩니다. Dev Environment의 `/projects` 디렉터리에는 소스 리포지토리와 devfile에서 가져온 파일이 저장됩니다. 개발 환경을 처음 만들 때 비어 있는 `/home` 디렉토리에는 개발 환경을 사용하는 동안 만든 파일이 저장됩니다. 개발 환경의 `/projects` 및 `/home` 디렉터리에 있는 모든 내용은 영구적으로 저장됩니다.

> **Note**
>
> devfile 또는 devfile 구성 요소 이름을 변경하는 경우에만 `/home` 폴더가 변경됩니다. devfile 또는 devfile 구성 요소 이름을 변경하면 `/home` 디렉터리의 내용이 바뀌고 이전 `/home` 디렉터리 데이터를 복구할 수 없습니다.

루트에 devfile이 포함되지 않은 소스 리포지토리가 있는 개발 환경을 만들거나 소스 리포지토리가 없는 개발 환경을 만드는 경우 기본 유니버설 devfile이 소스 리포지토리에 자동으로 적용됩니다. 모두 동일한 기본 유니버설 개발 파일 이미지가 사용됩니다. IDEs CodeCatalyst 현재 데브파일 버전 2.0.0을 지원합니다. devfile에 대한 자세한 내용은 Devfile 스키마 - 버전 2.0.0을 참조하십시오.

> **Note**
>
> Devfile에는 퍼블릭 컨테이너 이미지만 포함할 수 있습니다.

VPC-connected 개발 환경은 다음과 같은 devfile 이미지만 지원한다는 점에 유의하세요.

- 유니버설 이미지
- 비공개 Amazon ECR 이미지 (리포지토리가 동일한 지역에 있는 경우) VPC

주제

- 개발 환경을 위한 리포지토리 devfile 편집
- 에서 지원하는 Devfile 기능 CodeCatalyst
- 개발 환경을 위한 devfile 예제
- 복구 모드를 사용한 리포지토리 devfile 문제 해결
- 개발 환경을 위한 범용 devfile 이미지 지정하기

- 데브파일 명령
- 데브파일 이벤트
- 디브파일 컴포넌트

개발 환경을 위한 리포지토리 devfile 편집

다음 절차를 사용하여 개발 환경용 리포지토리 devfile을 편집하십시오.

에서 개발 환경을 위한 리포지토리 개발 파일 편집 CodeCatalyst

리포지토리 개발 파일을 편집하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. devfile을 편집하려는 소스 리포지토리가 들어 있는 프로젝트로 이동합니다.
3. 탐색 창에서 코드를 선택합니다.
4. 소스 리포지토리를 선택합니다.
5. 편집하려는 devfile이 들어 있는 소스 리포지토리를 선택합니다.
6. 파일 목록에서 `devfile.yaml` 파일을 선택합니다.
7. 편집을 선택합니다.
8. dev파일을 편집합니다.
9. 커밋을 선택하거나 팀원이 변경 사항을 검토하고 승인할 수 있도록 풀 리퀘스트를 생성하세요.

> **Note**
>
> devfile을 편집한 경우 변경사항을 적용하려면 devfile을 다시 시작해야 합니다. 를 실행하여 이 작업을 수행할 수 있습니다. `/aws/mde/mde start --location devfile.yaml` devfile을 시작하는 데 문제가 있으면 복구 모드로 전환됩니다. 하지만 VPC 연결된 개발 환경과 관련된 개발 파일을 편집하는 경우 변경 내용을 적용하려면 대신 개발 환경을 다시 시작해야 합니다.

를 실행하여 사용 중인 devfile을 검토할 수 있습니다. `/aws/mde/mde status` 위치 필드에는 환경 폴더를 기준으로 한 devfile 경로가 있습니다. `/projects`

```
{
```

```
            "status": "STABLE",
            "location": "devfile.yaml"
        }
```

기본 devfile을 소스 코드 리포지토리로 옮길 수도 있습니다. `/projects/devfile.yaml` devfile 위치를 업데이트하려면 다음 명령을 사용하십시오. `/aws/mde/mde start --location` *`repository-name`*`/devfile.yaml`

개발 환경의 리포지토리 개발 파일 편집 IDE

개발 환경의 구성을 변경하려면 devfile을 편집해야 합니다. 지원되는 IDE 환경에서 devfile을 편집한 다음 개발 환경을 업데이트하는 것이 좋지만, 에서 소스 리포지토리의 루트에서 devfile을 편집할 수도 있습니다. CodeCatalyst 지원되는 IDE 환경에서 개발 파일을 편집하는 경우 변경 사항을 소스 리포지토리에 커밋하고 푸시하거나 팀 구성원이 개발 파일 편집 내용을 검토하고 승인할 수 있도록 pull 요청을 만들어야 합니다.

- 개발 환경의 리포지토리 개발 파일 편집 AWS Cloud9
- VS Code에서 개발 환경을 위한 리포지토리 개발 파일 편집
- 개발 환경을 위한 리포지토리 개발 파일 편집 JetBrains

에서 지원하는 Devfile 기능 CodeCatalyst

CodeCatalyst 버전 2.0.0에서 다음과 같은 devfile 기능을 지원합니다. devfile에 대한 자세한 내용은 Devfile 스키마 - 버전 2.0.0을 참조하십시오.

기능	유형
`exec`	Command
`postStart`	Event
`container`	구성 요소
`args`	구성 요소 속성
`env`	구성 요소 속성
`mountSources`	구성 요소 속성

기능	유형
volumeMounts	구성 요소 속성

개발 환경을 위한 devfile 예제

다음은 간단한 devfile의 예시입니다.

```
schemaVersion: 2.0.0
metadata:
  name: al2
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      mountSources: true
      command: ['sleep', 'infinity']
  - name: dockerstore
commands:
  - id: setupscript
    exec:
      component: test
      commandLine: "chmod +x script.sh"
      workingDir: /projects/devfiles
  - id: executescript
    exec:
      component: test
      commandLine: "/projects/devfiles/script.sh"
  - id: yumupdate
    exec:
      component: test
      commandLine: "yum -y update --security"
events:
  postStart:
    - setupscript
    - executescript
    - yumupdate
```

Devfile 시작, 명령 및 이벤트 로그가 캡처되어 저장됩니다. /aws/mde/logs devfile 동작을 디버깅하려면 제대로 작동하는 devfile을 사용하여 개발 환경을 시작하고 로그에 액세스하세요.

복구 모드를 사용한 리포지토리 devfile 문제 해결

개발 파일을 시작하는 데 문제가 있는 경우 복구 모드로 전환되므로 환경에 계속 연결하여 devfile을 수정할 수 있습니다. 복구 모드에서는 실행 시 devfile 위치가 포함되지 `/aws/mde/mde status` 않습니다.

```
{
            "status": "STABLE"
        }
```

아래 로그에서 오류를 확인하고, devfile을 수정하고`/aws/mde/logs`, 다시 `/aws/mde/mde start` 실행해 볼 수 있습니다.

개발 환경을 위한 범용 devfile 이미지 지정하기

기본 유니버설 이미지에는 IDE에 사용할 수 있는 가장 일반적으로 사용되는 프로그래밍 언어와 관련 도구가 포함되어 있습니다. 지정된 이미지가 없는 경우 이 이미지가 제공되며 에서 유지 관리하는 도구가 포함됩니다 CodeCatalyst. CodeCatalyst 새 이미지 릴리스에 대한 알림을 계속 받으려면 을 참조하십시오SNS를 통한 유니버설 이미지 알림 구독.

> **Note**
>
> `public.ecr.aws/aws-mde/universal-image:latest`이미지가 `public.ecr.aws/aws-mde/universal-image:3.0` 이미지를 가져옵니다.

CodeCatalyst Amazon은 다음과 같은 devfile 이미지를 지원합니다.

이미지 버전	이미지 식별자
`Universal image 1.0`	`public.ecr.aws/aws-mde/universal-image:1.0`
`Universal image 2.0`	`public.ecr.aws/aws-mde/universal-image:2.0`
`Universal image 3.0`	`public.ecr.aws/aws-mde/universal-image:3.0`

> **Note**
>
> 를 사용하는 경우 PHP AWS Cloud9, Ruby 및 CSS로 업그레이드한 후에는 자동 완성 기능이 작동하지 않습니다. `universal-image:3.0`

주제

SNS를 통한 유니버설 이미지 알림 구독

CodeCatalyst 범용 이미지 알림 서비스를 제공합니다. 이를 사용하여 범용 이미지 업데이트가 CodeCatalyst 릴리스되면 알려주는 Amazon Simple Notification Service (SNS) 주제를 구독할 수 있습니다. SNS 주제에 대한 자세한 내용은 Amazon 단순 알림 서비스란 무엇입니까? 를 참조하십시오. .

새 유니버설 이미지가 출시될 때마다 구독자에게 알림을 보냅니다. 이 섹션에서는 CodeCatalyst 유니버설 이미지 업데이트를 구독하는 방법을 설명합니다.

샘플 메시지

```
{
    "Type": "Notification",
    "MessageId": "123456789",
    "TopicArn": "arn:aws:sns:us-east-1:1234657890:universal-image-updates",
    "Subject": "New Universal Image Release",
    "Message": {
        "v1": {
            "Message": "A new version of the Universal Image has been released. You are
 now able to launch new DevEnvironments using this image.",
            "image ": {
                "release_type": "MAJOR VERSION",
                "image_name": "universal-image",
                "image_version": "2.0",
                "image_uri": "public.ecr.aws/amazonlinux/universal-image:2.0"
            }
        }
    },
```

```
    "Timestamp": "2021-09-03T19:05:57.882Z",
    "UnsubscribeURL": "example url"
}
```

Amazon SNS 콘솔을 사용하여 CodeCatalyst 유니버설 이미지 업데이트를 구독하려면

1. Amazon SNS 콘솔을 열어 대시보드로 이동합니다.
2. 탐색 표시줄에서 원하는 항목을 선택합니다 AWS 리전.
3. 탐색 창에서 구독을 선택하고 나서 구독 생성을 선택합니다.
4. 주제 ARN에 를 입력합니다. `arn:aws:sns:us-east-1:089793673375:universal-image-updates`
5. 프로토콜에서 이메일을 선택합니다.
6. 엔드포인트에 이메일 주소를 입력합니다. 이 이메일 주소는 알림을 받는 데 사용됩니다.
7. 구독 생성을 선택합니다.
8. 제목이 “AWS 알림 - 구독 확인”인 확인 이메일을 받게 됩니다. 이메일을 열고 구독 확인을 선택합니다.

Amazon SNS 콘솔을 사용하여 CodeCatalyst 유니버설 이미지 업데이트 구독을 취소하려면

1. Amazon SNS 콘솔을 열어 대시보드로 이동합니다.
2. 탐색 표시줄에서 원하는 항목을 선택합니다 AWS 리전.
3. 탐색 창에서 구독을 선택한 다음 구독을 취소하려는 구독을 선택합니다.
4. 작업을 선택한 다음 구독 삭제를 선택합니다.
5. 삭제를 선택합니다.

유니버설 이미지 1.0 런타임 버전

다음 표에는 에서 사용할 수 있는 런타임이 나와 있습니다. `universal-image:1.0`

`universal-image:1.0`런타임 버전

실행 시간 이름	버전	특정 메이저 버전 및 최신 마이너 버전
aws cli	2.11	`aws-cli: 2.x`
도커 컴포즈	2.16	`docker-compose: 2.x`

실행 시간 이름	버전	특정 메이저 버전 및 최신 마이너 버전
dotnet	6.0	`dotnet: 6.x`
	7.0	`dotnet: 7.x`
golang	1.19	`golang: 1.x`
java	corretto11	`java: corretto11.x`
	corretto17	`java: corretto17.x`
nodejs	14.20	`nodejs: 14.x`
	16.19	`nodejs: 16.x`
OpenSSL	1.0	`openssl: 1.x`
	1.1	
php	7.2	`php: 7.x`
python	3.9	`python: 3.x`
	3.10	
ruby	3.1	`ruby: 3.x`
테라포밍	1.4	`terraform: 1.x`

유니버설 이미지 2.0 런타임 버전

다음 표에는 에서 사용할 수 있는 런타임이 나와 있습니다. `universal-image:2.0`

`universal-image:2.0`런타임 버전

실행 시간 이름	버전	특정 메이저 버전 및 최신 마이너 버전
aws cli	2.11	`aws-cli: 2.x`
도커 컴포즈	2.17	`docker-compose: 2.x`

실행 시간 이름	버전	특정 메이저 버전 및 최신 마이너 버전
dotnet	6.0	`dotnet: 6.x`
	7.0	`dotnet: 7.x`
golang	1.20	`golang: 1.x`
java	corretto11	`java: corretto11.x`
	corretto17	`java: corretto17.x`
nodejs	16.19	`nodejs: 16.x`
OpenSSL	1.0	`openssl: 1.x`
	1.1	
php	7.2	`php: 7.x`
python	3.9	`python: 3.x`
	3.10	
ruby	3.2	`ruby: 3.x`
테라포밍	1.4	`terraform: 1.x`

유니버설 이미지 3.0 런타임 버전

다음 표에는 에서 사용할 수 있는 런타임이 나와 있습니다. `universal-image:3.0`

`universal-image:3.0`런타임 버전

실행 시간 이름	버전	특정 메이저 버전 및 최신 마이너 버전
aws cli	2.11	`aws-cli: 2.x`
도커 컴포즈	2.17	`docker-compose: 2.x`
dotnet	6.0	`dotnet: 6.x`

실행 시간 이름	버전	특정 메이저 버전 및 최신 마이너 버전
	7.0	`dotnet: 7.x`
golang	1.21	`golang: 1.x`
java	corretto11	`java: corretto11.x`
	corretto17	`java: corretto17.x`
nodejs	18.17	`nodejs: 18.x`
	20.6	`nodejs: 20.x`
OpenSSL	3.0	`openssl: 3.x`
php	8.2	`php: 8.x`
python	3.9	`python: 3.x`
	3.11	
ruby	3.2	`ruby: 3.x`
테라포밍	1.5	`terraform: 1.x`

데브파일 명령

현재는 devfile의 CodeCatalyst `exec` 명령어만 지원합니다. 자세한 내용은 DevFile.io 설명서의 명령 추가를 참조하십시오.

다음 예제는 devfile에서 `exec` 명령을 지정하는 방법을 보여줍니다.

```
commands:
  - id: setupscript
    exec:
      component: test
      commandLine: "chmod +x script.sh"
      workingDir: /projects/devfiles
  - id: executescript
    exec:
```

```
      component: test
      commandLine: "./projects/devfiles/script.sh"
  - id: updateyum
    exec:
      component: test
      commandLine: "yum -y update --security"
```

개발 환경에 연결되면 터미널을 통해 정의된 명령을 실행할 수 있습니다.

```
/aws/mde/mde command <command-id>
/aws/mde/mde command executescript
```

장기 실행 명령의 경우 스트리밍 플래그를 `-s` 사용하여 명령 실행을 실시간으로 출력할 수 있습니다.

```
/aws/mde/mde -s command <command-id>
```

> **Note**
>
> `command-id`소문자여야 합니다.

에서 지원하는 실행 매개변수 CodeCatalyst

CodeCatalyst devfile 버전 `exec` 2.0.0에서 다음 파라미터를 지원합니다.

- `commandLine`
- `component`
- `id`
- `workingDir`

데브파일 이벤트

현재는 devfile의 CodeCatalyst `postStart` 이벤트만 지원합니다. 자세한 내용은 postStartObjectDevFile.io 설명서를 참조하십시오.

다음 예제는 devfile에 postStart 이벤트 바인딩을 추가하는 방법을 보여줍니다.

```
commands:
  - id: executescript
    exec:
      component: test
      commandLine: "./projects/devfiles/script.sh"
  - id: updateyum
    exec:
      component: test
      commandLine: "yum -y update --security"
events:
  postStart:
    - updateyum
    - executescript
```

시작 후 개발 환경은 지정된 postStart 명령을 정의된 순서대로 실행합니다. 명령이 실패해도 개발 환경은 계속 실행되며 실행 출력은 아래 로그에 저장됩니다. /aws/mde/logs

디브파일 컴포넌트

현재는 devfile의 container 구성 CodeCatalyst 요소만 지원합니다. 자세한 내용은 DevFile.io 설명서의 구성 요소 추가를 참조하십시오.

다음 예제는 devfile의 컨테이너에 시작 명령을 추가하는 방법을 보여줍니다.

```
components:
  - name: test
    container:
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      command: ['sleep', 'infinity']
```

> **Note**
>
> 컨테이너에 수명이 짧은 입력 명령이 있는 경우 컨테이너를 계속 command: ['sleep', 'infinity'] 실행하도록 포함해야 합니다.

CodeCatalyst 또한 컨테이너 구성 요소의 다음 속성도 지원합니다: args envmountSources,,volumeMounts.

개발 VPC 환경에 연결 연결

VPC연결은 워크플로에서 a에 액세스하는 데 필요한 모든 구성을 포함하는 CodeCatalyst 리소스입니다. VPC 스페이스 관리자는 스페이스 구성원을 대신하여 Amazon CodeCatalyst 콘솔에서 자신의 VPC 연결을 추가할 수 있습니다. VPC연결을 추가하면 스페이스 구성원은 워크플로 작업을 실행하고 네트워크 규칙을 준수하고 관련 VPC 리소스에 액세스할 수 있는 개발 환경을 만들 수 있습니다.

> **Important**
>
> VPC연결이 있는 개발 환경은 연결된 타사 소스 리포지토리를 지원하지 않습니다. CodeCatalyst

개발 환경 생성 시에만 개발 환경을 VPC 연결에 연결할 수 있습니다. 개발 환경을 만든 후에는 해당 VPC 연결을 변경할 수 없습니다. 다른 VPC 연결을 사용하려면 현재 개발 환경을 삭제하고 새 개발 환경을 만들어야 합니다.

> **Note**
>
> 개발 환경은 프로젝트에 액세스할 수 있는 AWS 계정의 VPC 연결에만 연결할 수 있습니다. 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 프로젝트 제한 계정 연결 구성을 참조하십시오.

참고로 개발 환경은 생성 시 여러 AWS 리소스와 서비스를 활용합니다. 즉, 개발 환경은 다음 AWS 서비스에 연결됩니다.

- 아마존 CodeCatalyst
- AWS SSM
- AWS KMS
- 아마존 ECR
- 아마존 CloudWatch
- 아마존 ECS

> **Note**
>
> AWS Toolkit 연결된 VPC 연결을 통한 개발 환경 생성을 지원하지 않습니다. 또한 IDE 다른 방법을 사용할 경우 로딩 시간이 5분 정도 걸릴 수 있다는 점도 참고하세요. AWS Cloud9

스페이스 수준에서 VPC 연결을 관리하려면 스페이스 관리자 역할 또는 고급 사용자 역할이 있어야 합니다. 에 대한 VPCs 자세한 내용은 CodeCatalyst 관리자 안내서의 Amazon VPCs 관리를 참조하십시오. CodeCatalyst

의 개발 환경에 대한 할당량 CodeCatalyst

다음 표에는 Amazon의 개발 환경에 대한 할당량 및 한도가 설명되어 있습니다. CodeCatalyst CodeCatalystAmazon의 할당량에 대한 자세한 내용은 을 참조하십시오. 에 대한 할당량 CodeCatalyst

월별 개발 환경 시간 수	개발 환경 시간은 공간의 전체 스토리지 한도에 영향을 받습니다. 자세한 내용은 요금 및 개발 환경 관련 문제 해결 섹션을 참조하세요.
공간당 개발 환경 스토리지의 양	개발 환경 스토리지는 공간의 전체 스토리지 한도의 영향을 받습니다. 자세한 내용은 요금 및 개발 환경 관련 문제 해결 섹션을 참조하세요.
개발 환경 컴퓨팅 용량	개발 환경 컴퓨팅은 공간의 전체 스토리지 한도의 영향을 받습니다. 자세한 내용은 요금 및 개발 환경 관련 문제 해결 섹션을 참조하세요.

에서 소프트웨어 패키지 게시 및 공유 CodeCatalyst

CodeCatalyst Amazon에는 개발 팀이 애플리케이션 개발에 사용되는 소프트웨어 패키지를 안전하게 저장하고 공유할 수 있는 완전 관리형 패키지 저장소 서비스가 포함되어 있습니다. 이러한 패키지는 패키지 리포지토리에 저장되며, 패키지 리포지토리는 에서 프로젝트 내에서 생성 및 구성됩니다. CodeCatalyst

단일 패키지 저장소에는 지원되는 모든 패키지 유형의 패키지를 저장할 수 있습니다. CodeCatalyst 지원되는 패키지 형식은 다음과 같습니다.

- npm
- Maven
- NuGet
- Python

패키지 리포지토리의 패키지를 검색하고 해당 리포지토리를 포함하는 프로젝트 멤버 간에 공유할 수 있습니다.

패키지를 리포지토리에 게시하고 리포지토리에서 패키지를 사용하려면 리포지토리 엔드포인트 (URL)를 사용하도록 패키지 관리자를 구성하십시오. 그런 다음 패키지 관리자를 사용하여 패키지를 리포지토리에 게시할 수 있습니다. Maven, Gradle, npm, yarn, nuget, dotnet, pip, twine과 같은 패키지 관리자를 사용할 수 있습니다.

패키지 리포지토리를 사용하도록 워크플로를 구성할 수도 있습니다. CodeCatalyst CodeCatalyst 워크플로에서 패키지를 사용하는 방법에 대한 자세한 내용은 을 참조하십시오. 패키지 리포지토리를 워크플로에 연결

한 패키지 저장소의 패키지를 업스트림 저장소로 추가하여 동일한 프로젝트의 다른 저장소에서 사용할 수 있도록 할 수 있습니다. 업스트림 리포지토리에서 사용할 수 있는 모든 패키지 버전을 다운스트림 리포지토리에서도 사용할 수 있습니다. 자세한 내용은 업스트림 리포지토리 구성 및 사용 단원을 참조하십시오.

게이트웨이라는 특별한 유형의 CodeCatalyst 리포지토리를 만들어 리포지토리에서 오픈 소스 패키지를 사용할 수 있도록 할 수 있습니다. 게이트웨이 리포지토리로 업스트리밍하면 npmjs.com 및 pypi.org와 같은 인기 있는 퍼블릭 리포지토리의 패키지를 사용하고 리포지토리에 자동으로 캐시할 수 있습니다. CodeCatalyst 자세한 내용은 공용 외부 리포지토리에 연결 단원을 참조하십시오.

주제

패키지 컨셉

다음은 패키지를 관리, 게시 또는 사용할 때 알아야 할 몇 가지 개념과 용어입니다 CodeCatalyst.

패키지

패키지는 소프트웨어를 설치하고 종속성을 해결하는 데 필요한 소프트웨어와 메타데이터를 모두 포함하는 번들입니다. CodeCatalyst npm 패키지 형식을 지원합니다.

패키지는 다음과 같이 구성됩니다.

- 이름 (예: 인기 있는 npm 패키지의 이름) `webpack`
- 선택적 네임스페이스 (예: in) `@types` `@types/node`
- 버전 세트 (예:,`1.0.0`,`1.0.1`) `1.0.2`
- 패키지 수준 메타데이터 (예: npm dist 태그)

패키지 네임스페이스

일부 패키지 형식은 패키지를 논리적 그룹으로 구성하고 이름 충돌을 방지하기 위해 계층적 패키지 이름을 지원합니다. 이름이 같은 패키지를 다른 네임스페이스에 저장할 수 있습니다. 예를 들어 npm 은 범위를 지원하며 npm 패키지는 범위와 `@types/node` 이름을 가집니다. `@types` node `@types` 범위에는 다른 패키지 이름이 많습니다. 에서 CodeCatalyst 범위 (“유형”) 는 패키지 네임스페이스라고 하고 이름 (“노드”) 은 패키지 이름이라고 합니다. Maven 패키지의 경우 패키지 네임스페이

스는 Maven groupID에 해당합니다. `org.apache.logging.log4j:log4j` Maven 패키지에는 `org.apache.logging.log4j` groupID(패키지 네임스페이스)와 `log4j` artifactID(패키지 이름)가 있습니다. Python과 같은 일부 패키지 형식은 npm scope 또는 Maven GroupId와 유사한 개념을 가진 계층 이름을 지원하지 않습니다. 패키지 이름을 그룹화할 방법이 없으면 이름 충돌을 피하기가 더 어려울 수 있습니다.

패키지 버전

패키지 버전은 패키지의 특정 버전(예: `@types/node@12.6.9`)을 식별합니다. 버전 번호 형식과 의미 체계는 패키지 형식에 따라 다릅니다. 예를 들어, npm 패키지 버전은 의미 체계 버전 관리 사양을 준수해야 합니다. 에서 CodeCatalyst 패키지 버전은 버전 식별자, package-version-level 메타데이터 및 자산 세트로 구성됩니다.

자산

자산은 npm 파일, Maven POM 또는 JAR 파일과 같은 패키지 버전과 연결된 개별 `.tgz` 파일에 저장되어 있는 개별 파일입니다. CodeCatalyst

패키지 리포지토리

CodeCatalyst 패키지 리포지토리는 패키지 세트를 포함하며, 패키지 버전에는 패키지 버전이 포함되며, 각 패키지는 자산 세트에 매핑됩니다. 패키지 리포지토리는 다중 언어를 사용하므로 단일 리포지토리에 지원되는 모든 유형의 패키지가 포함될 수 있습니다. 각 패키지 저장소는 (,), the, Maven CLI (`dotnet`)`nuget`, Python NuGet CLIs (및) 과 같은 도구를 사용하여 패키지를 가져오고 게시하기 위한 엔드포인트를 제공합니다. npm CLI `mvn` CLIs `pip` `twine` 각 스페이스에서 만들 수 있는 패키지 리포지토리 수를 CodeCatalyst 포함하여 패키지 할당량에 대한 자세한 내용은 을 참조하십시오. 패키지 할당량

업스트림으로 설정하여 패키지 저장소를 다른 패키지에 연결할 수 있습니다. 저장소가 업스트림으로 설정되면 업스트림의 모든 패키지는 물론 체인의 추가 업스트림 저장소도 사용할 수 있습니다. 자세한 내용은 업스트림 리포지토리 단원을 참조하십시오.

게이트웨이 리포지토리는 공식 외부 패키지 기관으로부터 패키지를 가져와 저장하는 특별한 유형의 패키지 리포지토리입니다. 자세한 내용은 게이트웨이 리포지토리 단원을 참조하십시오.

업스트림 리포지토리

를 CodeCatalyst 사용하여 두 패키지 리포지토리 간에 업스트림 관계를 생성할 수 있습니다. 패키지 리포지토리는 다른 패키지 리포지토리의 업스트림입니다. 패키지 리포지토리에 포함된 패키지 버전을

다운스트림 리포지토리의 패키지 리포지토리 엔드포인트에서 액세스할 수 있는 경우 패키지 리포지토리는 다른 패키지 리포지토리의 업스트림입니다. 업스트림 관계를 사용하면 클라이언트의 관점에서 두 패키지 리포지토리의 콘텐츠가 효과적으로 병합됩니다.

예를 들어 패키지 관리자가 리포지토리에 없는 패키지 버전을 요청하면 구성된 업스트림 리포지토리에서 패키지 버전을 검색합니다. CodeCatalyst 업스트림 리포지토리는 구성된 순서대로 검색되며, 패키지를 찾으면 검색이 중지됩니다. CodeCatalyst

게이트웨이 리포지토리

게이트웨이 리포지토리는 지원되는 외부의 공식 패키지 기관에 연결되는 특별한 유형의 패키지 리포지토리입니다. 게이트웨이 리포지토리를 업스트림 리포지토리로 추가하면 해당 공식 패키지 기관의 패키지를 사용할 수 있습니다. 다운스트림 리포지토리는 퍼블릭 리포지토리와 통신하지 않고 게이트웨이 리포지토리가 모든 것을 중재합니다. 이러한 방식으로 사용된 패키지는 게이트웨이 리포지토리와 원래 요청을 받은 다운스트림 리포지토리 모두에 저장됩니다.

게이트웨이 리포지토리는 미리 정의되어 있지만 사용하려면 각 프로젝트에서 만들어야 합니다. 다음 목록에는 생성할 수 있는 모든 게이트웨이 CodeCatalyst 리포지토리와 해당 리포지토리가 연결된 패키지 권한이 포함되어 있습니다.

- npm-public-registry-gatewaynpmjs.com의 npm 패키지를 제공합니다.
- maven-central-gateway메이븐 센트럴 리포지토리의 메이븐 패키지를 제공합니다.
- google-android-gateway구글 안드로이드의 Maven 패키지를 제공합니다.
- 커먼웨어 게이트웨이는 에서 Maven 패키지를 제공합니다. CommonsWare
- gradle-plugins-gatewayGradle 플러그인의 Maven 패키지를 제공합니다.
- nuget-gallery-gateway갤러리에서 NuGet NuGet 패키지를 제공합니다.
- pypi-gateway는 파이썬 패키지 인덱스에서 파이썬 패키지를 제공합니다.

패키지 리포지토리 구성 및 사용

CodeCatalyst에서는 패키지가 패키지 리포지토리 내에 저장되고 관리됩니다. CodeCatalyst (CodeCatalyst 또는 지원되는 모든 공용 패키지 저장소) 에 패키지를 게시하거나 패키지를 사용하려면 패키지 저장소를 만들고 패키지 관리자를 해당 저장소에 연결해야 합니다.

주제

- 패키지 리포지토리 생성

- 패키지 리포지토리에 연결
- 패키지 리포지토리 삭제

패키지 리포지토리 생성

에서 패키지 저장소를 만들려면 다음 단계를 수행하십시오 CodeCatalyst.

패키지 리포지토리를 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 패키지 저장소를 만들려는 프로젝트로 이동합니다.
3. 탐색 창에서 패키지를 선택합니다.
4. 패키지 저장소 페이지에서 패키지 저장소 만들기를 선택합니다.
5. Package 리포지토리 세부 정보 섹션에서 다음을 추가합니다.

   a. 리포지토리 이름. 프로젝트 또는 팀 이름, 리포지토리 사용 방식 등의 세부 정보가 포함된 설명이 포함된 이름을 사용해 보세요.

   b. (선택 사항) 설명. 리포지토리 설명은 프로젝트의 여러 팀에 리포지토리가 여러 개 있을 때 특히 유용합니다.
6. 업스트림 리포지토리 섹션에서 업스트림 리포지토리 선택을 선택하여 패키지 리포지토리를 통해 액세스하려는 패키지 리포지토리를 추가합니다. CodeCatalyst 게이트웨이 리포지토리를 추가하여 외부 패키지 리포지토리나 다른 리포지토리에 연결할 수 있습니다. CodeCatalyst

   - 패키지 리포지토리에서 패키지를 요청하면 업스트림 리포지토리가 이 목록에 표시된 순서대로 검색됩니다. 패키지를 찾으면 검색이 CodeCatalyst 중지됩니다. 업스트림 리포지토리의 순서를 변경하려면 목록에서 리포지토리를 끌어다 놓을 수 있습니다.
7. [Create] 를 선택하여 패키지 저장소를 생성합니다.

패키지 리포지토리에 연결

패키지를 게시하거나 패키지를 사용하려면 패키지 리포지토리 엔드포인트 정보 및 CodeCatalyst 자격 증명으로 패키지 관리자를 구성해야 합니다. CodeCatalyst 아직 리포지토리를 만들지 않은 경우 의 지침에 따라 리포지토리를 만들 수 패키지 리포지토리 생성 있습니다.

패키지 관리자를 CodeCatalyst 패키지 저장소에 연결하는 방법에 대한 지침은 다음 설명서를 참조하십시오.

- 그라들 그루비 구성 및 사용
- mvn 구성 및 사용
- 너겟 또는 닷넷 구성 및 사용 CLI
- npm 구성 및 사용
- pip 설정과 Python 패키지 설치
- 트와인 설정 및 Python 패키지 퍼블리싱

패키지 리포지토리 삭제

에서 패키지 저장소를 삭제하려면 다음 단계를 수행하십시오 CodeCatalyst.

패키지 리포지토리를 삭제하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 삭제하려는 패키지 저장소가 들어 있는 프로젝트로 이동합니다.
3. 탐색 창에서 패키지를 선택합니다.
4. Package 리포지토리 페이지에서 삭제하려는 리포지토리를 선택합니다.
5. Delete(삭제)를 선택합니다.
6. 패키지 리포지토리 삭제의 영향에 대해 제공된 정보를 검토하십시오.
7. 입력 `delete` 필드에 입력하고 삭제를 선택합니다.

업스트림 리포지토리 구성 및 사용

게이트웨이 리포지토리와 다른 CodeCatalyst 패키지 리포지토리를 모두 패키지 리포지토리에 업스트림으로 연결할 수 있습니다. 이를 통해 패키지 관리자 클라이언트는 단일 패키지 리포지토리 엔드포인트를 사용하여 둘 이상의 패키지 리포지토리에 포함된 패키지에 액세스할 수 있습니다. 업스트림 리포지토리를 사용할 때의 주요 이점은 다음과 같습니다.

- 여러 소스에서 가져오도록 단일 리포지토리 엔드포인트로 패키지 관리자를 구성하기만 하면 됩니다.
- 업스트림 리포지토리에서 사용된 패키지는 다운스트림 리포지토리에 저장되므로 업스트림 리포지토리에 예기치 않은 중단이 발생하거나 업스트림 리포지토리의 패키지가 삭제되더라도 패키지를 사용할 수 있습니다.

패키지 리포지토리를 생성할 때 업스트림 리포지토리를 추가할 수 있습니다. 콘솔에서 기존 패키지 리포지토리에 업스트림 리포지토리를 추가하거나 기존 패키지 리포지토리에서 업스트림 리포지토리를 제거할 수도 있습니다. CodeCatalyst

게이트웨이 리포지토리를 업스트림 리포지토리로 추가하면 패키지 리포지토리가 게이트웨이 리포지토리의 해당 공용 패키지 리포지토리에 연결됩니다. 지원되는 공개 패키지 리포지토리 목록은 을 참조하십시오. 지원되는 외부 패키지 리포지토리 및 게이트웨이 리포지토리

여러 리포지토리를 업스트림 리포지토리로 연결할 수 있습니다. 예를 들어 팀에서 이름이 and인 다른 저장소를 만들고 이미 사용 `project-repo` 중이며, 이 저장소에 업스트림 저장소로 npm-public-registry-gateway추가된 이 저장소를 공용 npm 저장소에 연결되어 `team-repo` 있다고 가정해 보겠습니다. `npmjs.com` 에 업스트림 `team-repo` 리포지토리로 추가할 수 있습니다. `project-repo` 이 경우,, `project-repo` `team-repo`npm-public-registry-gateway, 에서 패키지를 가져오는 `project-repo` 데 사용할 패키지 관리자를 구성하기만 하면 됩니다. `npmjs.com`

주제

- 업스트림 리포지토리 추가
- 업스트림 리포지토리의 검색 순서 편집
- 업스트림 리포지토리가 포함된 패키지 버전 요청
- 업스트림 리포지토리 제거

업스트림 리포지토리 추가

공용 패키지 리포지토리 또는 다른 CodeCatalyst 패키지 리포지토리를 다운스트림 리포지토리에 업스트림 리포지토리로 추가하면 다운스트림 리포지토리에 연결된 패키지 관리자가 업스트림 리포지토리의 모든 패키지를 사용할 수 있습니다.

업스트림 리포지토리를 추가하려면

1. 탐색 창에서 Packages(패키지)를 선택합니다.
2. 패키지 저장소 페이지에서 업스트림 저장소를 추가할 패키지 저장소를 선택합니다.
3. 패키지 리포지토리 이름 아래에서 업스트림을 선택하고 업스트림 리포지토리 선택을 선택합니다.
4. 업스트림 유형 선택에서 다음 중 하나를 선택합니다.
   - 게이트웨이 리포지토리

     사용 가능한 게이트웨이 리포지토리 목록에서 선택할 수 있습니다.

> **Note**
>
> Maven Central, npmjs.com 또는 Nuget Gallery와 같은 공용 외부 패키지 기관에 연결하려면 게이트웨이 리포지토리를 외부 리포지토리에서 가져온 패키지를 검색하고 저장하는 중간 리포지토리로 CodeCatalyst 사용합니다. 이렇게 하면 프로젝트의 모든 패키지 리포지토리가 게이트웨이 중간 리포지토리의 패키지를 사용하므로 시간과 데이터 전송이 절약됩니다. 자세한 내용은 공용 외부 리포지토리에 연결 단원을 참조하십시오.

- CodeCatalyst 리포지토리

  프로젝트에서 사용 가능한 CodeCatalyst 패키지 리포지토리 목록에서 선택할 수 있습니다.

5. 업스트림 리포지토리로 추가할 리포지토리를 모두 선택했으면 [Select] 를 선택한 다음 [Save] 를 선택합니다.

   업스트림 저장소의 검색 순서 변경에 대한 자세한 내용은 을 참조하십시오. 업스트림 리포지토리의 검색 순서 편집

업스트림 리포지토리를 추가한 경우 로컬 리포지토리에 연결된 패키지 관리자를 사용하여 업스트림 리포지토리에서 패키지를 가져올 수 있습니다. 패키지 관리자 구성을 업데이트할 필요가 없습니다. 업스트림 저장소에서 패키지 버전을 요청하는 방법에 대한 자세한 내용은 을 참조하십시오업스트림 리포지토리가 포함된 패키지 버전 요청.

업스트림 리포지토리의 검색 순서 편집

CodeCatalyst 구성된 검색 순서대로 업스트림 리포지토리를 검색합니다. 패키지를 찾으면 검색을 중단합니다. CodeCatalyst 업스트림 리포지토리에서 패키지를 검색하는 순서를 변경할 수 있습니다.

업스트림 저장소의 검색 순서 편집하기

1. 탐색 창에서 Packages(패키지)를 선택합니다.
2. 패키지 저장소 페이지에서 업스트림 저장소 검색 순서를 편집하려는 패키지 저장소를 선택합니다.
3. 패키지 저장소 이름 아래에서 업스트림을 선택합니다.
4. 업스트림 리포지토리 섹션에서 업스트림 리포지토리와 해당 검색 순서를 볼 수 있습니다. 검색 순서를 변경하려면 목록에서 리포지토리를 끌어다 놓습니다.
5. 업스트림 저장소의 검색 순서 편집을 마치면 [Save] 를 선택합니다.

업스트림 리포지토리가 포함된 패키지 버전 요청

다음 예제는 패키지 관리자가 업스트림 리포지토리가 있는 패키지 저장소에서 패키지를 요청할 때 발생할 수 CodeCatalyst 있는 시나리오를 보여줍니다.

이 예제의 경우 패키지 관리자 (예:) 는 여러 npm 업스트림 저장소가 downstream 있는 이름이 지정된 패키지 저장소에서 패키지 버전을 요청합니다. 패키지 요청 시 다음과 같은 상황이 발생할 수 있습니다.

- 요청된 패키지 버전이 downstream에 포함되어 있는 경우 클라이언트에 반환됩니다.
- 요청된 패키지 버전을 포함하지 downstream 않는 경우 업스트림 리포지토리에서 구성된 검색 순서대로 해당 패키지를 CodeCatalyst 검색합니다. downstream 패키지 버전을 찾으면 해당 패키지에 대한 참조가 downstream에 복사되고 패키지 버전이 클라이언트에 반환됩니다.
- downstream둘 다 해당 업스트림 리포지토리에 패키지 버전이 포함되어 있지 않은 경우 HTTP 404 응답이 클라이언트에 Not Found 반환됩니다.

하나의 리포지토리에 허용되는 최대 직접 업스트림 리포지토리 수는 10개입니다. 패키지 버전을 요청할 때 CodeCatalyst 검색할 수 있는 최대 저장소 수는 25개입니다.

업스트림 리포지토리의 패키지 보존

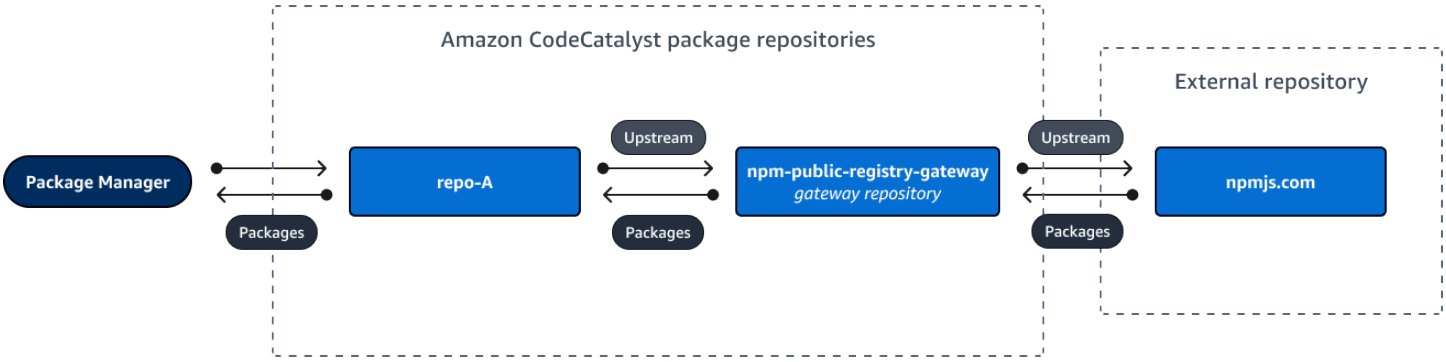
요청된 패키지 버전이 업스트림 저장소에 있는 경우 해당 버전에 대한 참조는 유지되며 요청한 저장소에서 항상 사용할 수 있습니다. 이렇게 하면 업스트림 저장소가 예기치 않게 중단되는 경우에도 패키지에 액세스할 수 있습니다. 유지되는 패키지 버전은 다음 사항에 영향을 받지 않습니다.

- 업스트림 리포지토리 삭제
- 업스트림 리포지토리와 다운스트림 리포지토리 간 연결 해제
- 업스트림 리포지토리에서 패키지 버전 삭제
- 업스트림 리포지토리의 패키지 버전 편집(예: 새 자산 추가)

업스트림 관계를 통해 패키지 가져오기

CodeCatalyst 업스트림 리포지토리라고 하는 여러 개의 연결된 리포지토리를 통해 패키지를 가져올 수 있습니다. CodeCatalyst 패키지 저장소에 게이트웨이 저장소에 대한 업스트림 연결이 있는 다른 CodeCatalyst 패키지 저장소에 대한 업스트림 연결이 있는 경우 업스트림 저장소에 없는 패키지에 대한 요청은 외부 저장소에서 복사됩니다. 예를 들어, 이라는 리포지토리가 게이트웨이 리포지토리

에 대한 업스트림 연결을 repo-A 가지고 있는 구성을 생각해 보십시오. npm-public-registry-gateway npm-public-registry-gateway공개 패키지 리포지토리인 https://npmjs.com 에 대한 업스트림 연결이 있습니다.



repo-A저장소를 사용하도록 구성된 경우 npm 를 실행하면 https://npmjs.com 에서 로 패키지 복사가 npm install 시작됩니다. npm-public-registry-gateway 설치된 버전도 함께 repo-A로 가져옵니다. 다음 예제에서는 lodash를 설치합니다.

```
$ npm config get registry
https://packages.region.codecatalyst.aws/npm/space-name/proj-name/repo-name/
$ npm install lodash
+ lodash@4.17.20
added 1 package from 2 contributors in 6.933s
```

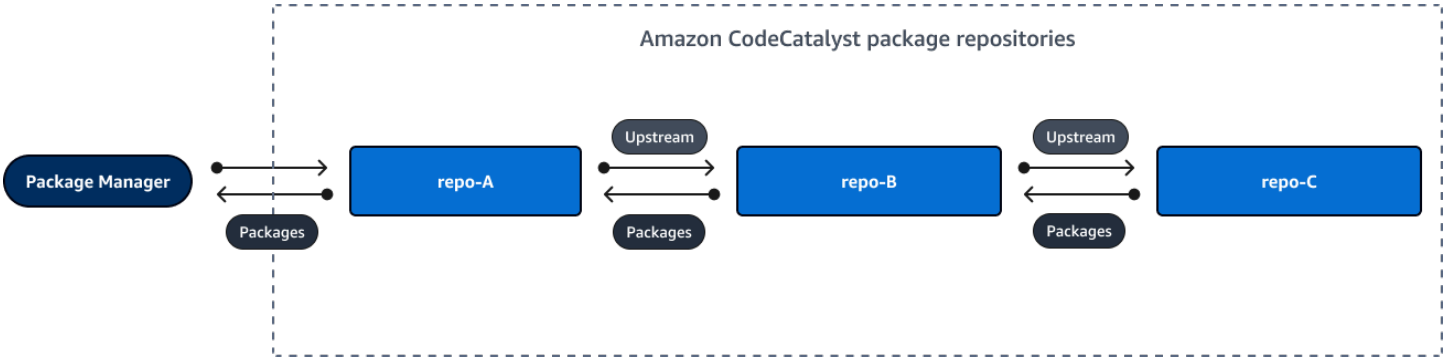
실행 npm install 후에는 최신 버전 (lodash 4.17.20) 만 repo-A 포함됩니다. 해당 버전이 가져온 버전이기 때문입니다. npm repo-A

https://npmjs.com 에 대한 외부 업스트림 연결이 npm-public-registry-gateway 있기 때문에 https://npmjs.com 에서 가져온 모든 패키지 버전이 에 저장됩니다. npm-public-registry-gateway 으로 연결되는 업스트림 연결이 있는 모든 다운스트림 저장소에서 이러한 패키지 버전을 가져올 수 있습니다. npm-public-registry-gateway

의 내용은 시간이 npm-public-registry-gateway 지남에 따라 https://npmjs.com 에서 가져온 모든 패키지 및 패키지 버전을 볼 수 있는 방법을 제공합니다.

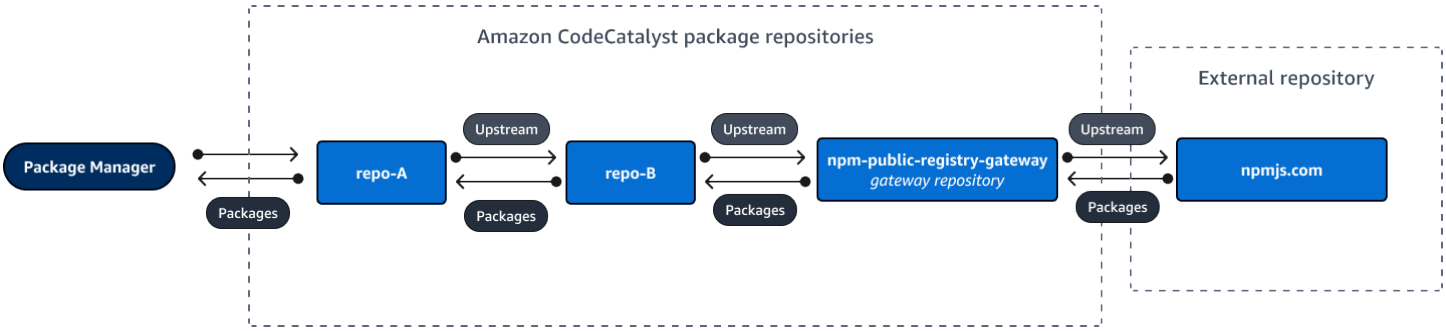
중간 리포지토리에 패키지 보존

CodeCatalyst 업스트림 리포지토리를 연결할 수 있습니다. 예를 들어 업스트림 리포지토리로 repo-B 둘 repo-A 수 있고 업스트림 repo-C 리포지토리로 가질 repo-B 수 있습니다. 이 구성을 통해 repo-A에서 repo-B 및 repo-C 패키지 버전을 사용할 수 있습니다.



패키지 관리자가 저장소에 repo-A 연결하고 저장소에서 패키지 버전을 가져오는 repo-C 경우 패키지 버전은 저장소에 보관되지 않습니다. repo-B 패키지 버전은 가장 멀리 떨어진 다운스트림 저장소 (이 예에서는) 에만 보관됩니다. repo-A 중간 저장소에는 보관되지 않습니다. 더 긴 체인의 경우에도 마찬가지입니다. 예를 들어,,,, 네 개의 리포지토리가 있고 repo-D 패키지 관리자가 연결되어 패키지 버전을 repo-A 가져온 경우 패키지 버전은 or에 유지되지만 or에는 유지되지 않습니다. repo-A repo-B repo-C repo-D repo-A repo-B repo-C

패키지 보존 동작은 공용 패키지 저장소에서 패키지 버전을 가져올 때와 비슷합니다. 단, 패키지 버전이 공용 저장소에 직접 업스트림으로 연결된 게이트웨이 저장소에 항상 보존된다는 점이 다릅니다. 예를 들어 업스트림 repo-A repo-B 리포지토리로 사용합니다. repo-B공용 리포지토리인 npmjs.com에 대한 업스트림 연결이 있는 업스트림 리포지토리로 있습니다. 아래 다이어그램을 참조하십시오. npm-public-registry-gateway



에 연결된 패키지 관리자가 특정 패키지 버전 (예: lodash 4.17.20) 을 **repo-A** 요청했는데 패키지 버전이 세 저장소 중 어디에도 없는 경우 npmjs.com에서 가져옵니다. lodash 4.17.20을 가져오면 가장 먼 다운스트림 저장소이고 공용 외부 저장소인 npmjs.com에 대한 업스트림 연결이 있기 때문에 이 저장소는 그대로 유지됩니다. repo-A npm-public-registry-gateway lodash 4.17.20은 중간 저장소이기 때문에 보관되지 않습니다. repo-B

업스트림 리포지토리 제거

업스트림 리포지토리의 패키지에 더 이상 액세스하지 않으려면 패키지 리포지토리에서 업스트림 리포지토리를 제거하면 됩니다.

> **Warning**
>
> 업스트림 리포지토리를 제거하면 업스트림 관계 체인이 끊어져 프로젝트 또는 빌드가 중단될 수 있습니다.

업스트림 리포지토리를 제거하려면

1. 탐색 창에서 Packages(패키지)를 선택합니다.
2. 패키지 저장소 페이지에서 업스트림 저장소를 제거하려는 패키지 저장소를 선택합니다.
3. 패키지 리포지토리 이름 아래에서 업스트림을 선택합니다.
4. 업스트림 리포지토리 편집 섹션에서 제거하려는 업스트림 리포지토리를 찾아 선택합니다.
   
5. 업스트림 리포지토리를 모두 제거했으면 [Save] 를 선택합니다.

공용 외부 리포지토리에 연결

해당 게이트웨이 리포지토리를 업스트림 리포지토리로 추가하여 CodeCatalyst 패키지 리포지토리를 지원되는 퍼블릭 외부 리포지토리에 연결할 수 있습니다. 게이트웨이 리포지토리는 외부 리포지토리에서 가져온 패키지를 검색하고 저장하는 중간 리포지토리 역할을 합니다. 이렇게 하면 프로젝트의 모든 패키지 리포지토리에서 게이트웨이 리포지토리의 저장된 패키지를 사용할 수 있으므로 시간과 데이터 전송이 절약됩니다.

게이트웨이 리포지토리를 사용하여 공용 리포지토리에 연결하려면

1. 탐색 창에서 Packages(패키지)를 선택합니다.
2. 패키지에서 게이트웨이 리포지토리 페이지를 선택합니다. 지원되는 게이트웨이 리포지토리 목록과 설명을 볼 수 있습니다.
3. 게이트웨이 리포지토리를 사용하려면 먼저 게이트웨이 리포지토리를 만들어야 합니다. 게이트웨이 리포지토리가 생성된 경우 생성된 날짜와 시간이 표시됩니다. 아직 생성하지 않은 경우 [Create] 를 선택하여 생성하십시오.

4. 게이트웨이 리포지토리의 패키지를 사용하려면 리포지토리에서 해당 패키지로의 업스트림 연결을 설정해야 합니다. CodeCatalyst Package 리포지토리를 선택하고 연결하려는 패키지 리포지토리를 선택합니다.
5. 공용 리포지토리에 연결하려면 업스트림을 선택하고 업스트림 리포지토리 선택을 선택합니다.
6. 게이트웨이 리포지토리를 선택하고 업스트림 리포지토리로 연결하려는 공용 리포지토리에 해당하는 게이트웨이 리포지토리를 선택합니다.
7. 업스트림 리포지토리로 추가할 게이트웨이 리포지토리를 모두 선택했으면 [Select] 를 선택합니다.
8. 업스트림 리포지토리 주문을 완료하면 [Save] 를 선택합니다.

업스트림 리포지토리에 대한 자세한 내용은 업스트림 리포지토리 구성 및 사용 섹션을 참조하세요.

게이트웨이 리포지토리를 업스트림 리포지토리로 추가한 경우 로컬 리포지토리에 연결된 패키지 관리자를 사용하여 해당 리포지토리에 해당하는 공개 외부 패키지 리포지토리에서 패키지를 가져올 수 있습니다. 패키지 관리자 구성을 업데이트할 필요가 없습니다. 이러한 방식으로 사용된 패키지는 게이트웨이 리포지토리와 로컬 패키지 리포지토리 모두에 저장됩니다. 업스트림 리포지토리에서 패키지 버전을 요청하는 방법에 대한 자세한 내용은 을 참조하십시오업스트림 리포지토리가 포함된 패키지 버전 요청.

지원되는 외부 패키지 리포지토리 및 게이트웨이 리포지토리

CodeCatalyst 게이트웨이 리포지토리를 사용하여 다음과 같은 공식 패키지 기관에 업스트림 연결을 추가할 수 있습니다.

리포지토리 패키지 유형	설명	게이트웨이 리포지토리 이름
npm	npm 공용 레지스트리	`npm-public-registry-gateway`
Python	Python Package Index	`pypi-gateway`
Maven	Maven Central	`maven-central-gateway`
Maven	Google Android Repository	`google-android-gateway`

리포지토리 패키지 유형	설명	게이트웨이 리포지토리 이름
Maven	CommonsWare	`commonsware-gateway`
Maven	Gradle 플러그인 저장소	`gradle-plugins-gateway`
NuGet	NuGet 갤러리	`nuget-gallery-gateway`

패키지 게시 및 수정

의 CodeCatalyst 패키지는 종속성을 해결하고 소프트웨어를 설치하는 데 필요한 소프트웨어 및 메타데이터 번들입니다. 에서 지원되는 패키지 형식 목록은 CodeCatalyst 을 참조하십시오에서 소프트웨어 패키지 게시 및 공유 CodeCatalyst. 이 섹션에서는 패키지 게시, 보기, 삭제 및 패키지 버전 상태 업데이트에 대한 정보를 제공합니다.

주제

- 패키지 저장소에 CodeCatalyst 패키지 게시
- 패키지 버전 세부 정보 보기
- 패키지 버전 삭제
- 패키지 버전 상태 업데이트
- 패키지 원본 제어 편집

패키지 저장소에 CodeCatalyst 패키지 게시

패키지 관리자 도구를 사용하여 지원되는 모든 패키지 유형의 버전을 CodeCatalyst 패키지 저장소에 게시할 수 있습니다. 패키지 버전을 게시하는 단계는 다음과 같습니다.

패키지 버전을 CodeCatalyst 패키지 저장소에 게시하려면

1. 아직 만들지 않았다면 패키지 저장소를 만드세요.
2. 패키지 관리자를 패키지 저장소에 연결합니다. npm 패키지 관리자를 CodeCatalyst 패키지 저장소에 연결하는 방법에 대한 지침은 을 참조하십시오npm 구성 및 사용.

3. 연결된 패키지 관리자를 사용하여 패키지 버전을 게시하십시오.

목차

리포지토리 게시 및 업스트림

CodeCatalyst에서는 연결 가능한 업스트림 리포지토리 또는 공용 리포지토리에 있는 패키지 버전을 게시할 수 없습니다. 예를 들어 npm 패키지를 패키지 리포지토리에 게시하려고 하는데 업스트림 리포지토리로 구성된 게이트웨이 리포지토리를 통해 `myrepo` npmjs.com에 `myrepo` 연결되어 있다고 가정해 보겠습니다. `lodash@1.0` 업스트림 리포지토리 또는 npmjs.com에 있는 경우 `lodash@1.0` 409 충돌 오류가 발생하여 게시 시도를 CodeCatalyst 거부합니다. `myrepo` 이렇게 하면 패키지와 이름 및 버전이 같은 패키지를 업스트림 저장소에 실수로 게시하여 예기치 않은 동작이 발생하는 것을 방지할 수 있습니다.

업스트림 리포지토리에 있는 패키지 이름의 다른 버전은 계속 게시할 수 있습니다. 예를 들어 업스트림 리포지토리에는 있지만 `lodash@1.1` 없는 경우 `lodash@1.0` 다운스트림 리포지토리에 `lodash@1.1` 게시할 수 있습니다.

프라이빗 패키지 및 퍼블릭 리포지토리

CodeCatalyst CodeCatalyst 리포지토리에 저장된 패키지를 npmjs.com 또는 Maven Central과 같은 공용 리포지토리에 게시하지 않습니다. CodeCatalyst 패키지를 공용 리포지토리에서 리포지토리로 가져오지만 패키지를 반대 방향으로 CodeCatalyst 이동하지는 않습니다. 리포지토리에 게시한 패키지는 비공개로 유지되며 CodeCatalyst 리포지토리가 속한 CodeCatalyst 프로젝트에서만 사용할 수 있습니다.

패키지 자산 덮어쓰기

이미 존재하며 콘텐츠가 다른 패키지 자산은 다시 게시할 수 없습니다. 예를 들어 에셋이 포함된 Maven 패키지를 이미 게시했다고 가정해 보겠습니다. JAR `mypackage-1.0.jar` 이 자산은 이전 자산과 새 자산의 체크섬이 동일한 경우에만 다시 게시할 수 있습니다. 새 콘텐츠와 함께 동일한 에셋을 다시 게시하려면 먼저 패키지 버전을 삭제하십시오. 콘텐츠가 다른 동일한 에셋 이름을 다시 게시하려고 하면 HTTP 409 충돌 오류가 발생합니다.

여러 자산 (Python 및 Maven) 을 지원하는 패키지 형식의 경우 필요한 권한이 있다고 가정하면 언제든지 기존 패키지 버전에 이름이 다른 새 자산을 추가할 수 있습니다. npm은 패키지 버전당 단일 NuGet 자산만 지원하므로 게시된 패키지 버전을 수정하려면 먼저 삭제해야 합니다.

이미 존재하는 자산(예:`mypackage-1.0.jar`)을 다시 게시하려고 할 때 게시된 자산과 새 자산의 내용이 동일하면, 작업은 멱등성이기 때문에 성공하게 됩니다.

패키지 버전 세부 정보 보기

CodeCatalyst 콘솔을 사용하여 특정 패키지 버전에 대한 세부 정보를 볼 수 있습니다.

패키지 버전 세부 정보를 보려면

1. 탐색 창에서 Packages(패키지)를 선택합니다.
2. 패키지 저장소 페이지에서 세부 정보를 보려는 패키지 버전이 들어 있는 저장소를 선택합니다.
3. 패키지 테이블에서 패키지 버전을 검색합니다. 검색 창을 사용하여 패키지 이름 및 형식별로 패키지를 필터링할 수 있습니다. 목록에서 패키지를 선택합니다.
4. 패키지 세부 정보 페이지에서 [Versions] 를 선택한 다음 보려는 버전을 선택합니다.

패키지 버전 삭제

CodeCatalyst 콘솔의 Package 버전 세부정보 페이지에서 패키지 버전을 삭제할 수 있습니다.

패키지 버전을 삭제하려면

1. 탐색 창에서 Packages(패키지)를 선택합니다.
2. 패키지 저장소 페이지에서 삭제하려는 패키지 버전이 들어 있는 저장소를 선택합니다.
3. 테이블에서 패키지를 검색하고 선택합니다.
4. 패키지 세부정보 페이지에서 버전을 선택하고 삭제하려는 버전을 선택합니다.
5. Package 버전 세부정보 페이지에서 버전 작업을 선택한 다음 삭제를 선택합니다.
6. 텍스트 필드에 delete를 입력하고 삭제를 선택합니다.

패키지 버전 상태 업데이트

의 모든 패키지 CodeCatalyst 버전에는 패키지 버전의 현재 상태 및 가용성을 설명하는 상태가 있습니다. CodeCatalyst콘솔에서 패키지 버전 상태를 변경할 수 있습니다. 패키지 버전의 가능한 상태 값 및 의미에 대한 자세한 내용은 을 참조하십시오패키지 버전 상태.

패키지 버전 상태를 업데이트하려면

1. 탐색 창에서 Packages(패키지)를 선택합니다.
2. 패키지 저장소 페이지에서 상태를 업데이트하려는 패키지 버전이 들어 있는 저장소를 선택합니다.
3. 테이블에서 패키지를 검색하고 선택합니다.
4. 패키지 세부 정보 페이지에서 버전을 선택한 다음 보려는 버전을 선택합니다.
5. 패키지 버전 세부정보 페이지에서 작업을 선택한 다음 목록 취소, 보관 또는 폐기를 선택합니다. 각 패키지 버전 상태에 대한 자세한 내용은 을 참조하십시오. 패키지 버전 상태
6. 텍스트 필드에 확인 텍스트를 입력한 다음 업데이트 중인 상태에 따라 목록 취소, 보관 또는 폐기를 선택합니다.

패키지 버전 상태

다음은 패키지 버전 상태에 사용할 수 있는 값입니다. 콘솔에서 패키지 버전 상태를 변경할 수 있습니다. 자세한 내용은 패키지 버전 상태 업데이트 단원을 참조하십시오.

- 게시됨: 패키지 버전이 성공적으로 게시되었으며 패키지 관리자가 요청할 수 있습니다. 패키지 버전은 패키지 관리자에게 반환되는 패키지 버전 목록 (예: 의 출력) 에 포함됩니다`npm view <package-name> versions`. 패키지 버전의 모든 자산은 리포지토리에서 사용할 수 있습니다.
- 미완료: 마지막 게시 시도가 완료되지 않았습니다. 현재는 Maven 패키지 버전만 미완료 상태일 수 있습니다. 이는 클라이언트가 패키지 버전용 에셋을 하나 이상 업로드하지만 해당 버전이 포함된 패키지의 `maven-metadata.xml` 파일은 게시하지 않는 경우 발생할 수 있습니다.
- 미등록: 패키지 버전 자산은 저장소에서 다운로드할 수 있지만 패키지 버전은 패키지 관리자에게 반환되는 버전 목록에 포함되지 않습니다. 예를 들어, npm 패키지의 경우 출력에는 패키지 버전이 포함되지 `npm view <package-name> versions` 않습니다. 즉, 사용 가능한 버전 목록에 버전이 표시되지 않기 때문에 npm 종속성 해결 로직에서는 패키지 버전을 선택하지 않습니다. 그러나 목록에 없는 패키지 버전이 `npm package-lock.json` 파일에서 이미 참조되어 있는 경우에도 해당 버전을 다운로드하여 설치할 수 있습니다 (예: 실행 시). `npm ci`
- 보관: 패키지 버전 자산을 다운로드할 수 없습니다. 패키지 버전은 패키지 관리자에 반환되는 버전 목록에 포함됩니다. 자산을 사용할 수 없으므로 클라이언트의 패키지 버전 사용이 차단됩니다. 애플리케이션 빌드가 Archived로 업데이트된 버전에 의존하는 경우 패키지 버전이 로컬에 캐시되지 않는 한 빌드는 실패합니다. 보관된 패키지 버전은 저장소에 계속 존재하므로 패키지 관리자나 빌드 도구를 사용하여 다시 게시할 수 없습니다. 하지만 콘솔에서 패키지 버전 상태를 미등록 또는 게시됨으로 다시 변경할 수 있습니다.

- 폐기: 패키지 버전이 목록에 표시되지 않고 저장소에서 자산을 다운로드할 수 없습니다. 삭제된 것과 보관된 것의 주요 차이점은 폐기 상태인 경우 패키지 버전의 자산이 에 의해 영구적으로 삭제된다는 것입니다. CodeCatalyst 따라서 패키지 버전을 폐기됨에서 보관됨, 미등록 또는 게시됨으로 전환할 수 없습니다. 자산이 삭제되었으므로 패키지 버전을 사용할 수 없습니다. 패키지 버전이 삭제됨으로 표시된 경우 패키지 자산 보관에 대한 요금은 청구되지 않습니다.

위 목록의 상태 외에도 패키지 버전을 삭제할 수 있습니다. 삭제된 패키지 버전은 저장소에 없으므로 패키지 관리자 또는 빌드 도구를 사용하여 해당 패키지 버전을 자유롭게 다시 게시할 수 있습니다.

패키지 이름, 패키지 버전 및 자산 이름 표준화

CodeCatalyst 패키지 이름, 패키지 버전 및 자산 이름을 저장하기 전에 정규화합니다. 즉, 패키지가 게시될 때 제공된 이름 또는 버전과 이름이 다를 CodeCatalyst 수 있습니다. 각 패키지 유형에서 이름 및 버전을 정규화하는 방법에 CodeCatalyst 대한 자세한 내용은 다음 설명서를 참조하십시오.

- Python 패키지 이름 정규화
- NuGet 패키지 이름, 버전 및 자산 이름 정규화

CodeCatalyst 다른 패키지 형식에서는 정규화를 수행하지 않습니다.

패키지 원본 제어 편집

Amazon에서는 패키지 버전을 직접 게시하거나 CodeCatalyst, 업스트림 리포지토리에서 가져오거나, 게이트웨이를 통해 외부 공개 리포지토리에서 패키지 버전을 수집하여 패키지 리포지토리에 추가할 수 있습니다. 공개 리포지토리에서 직접 게시하고 수집하여 패키지 버전을 추가하도록 허용하면 종속성 대체 공격에 취약합니다. 자세한 내용은 종속성 대체 공격 단원을 참조하십시오. 종속성 대체 공격으로부터 자신을 보호하려면 리포지토리의 패키지에 대한 패키지 오리진 제어를 구성하여 해당 패키지의 버전을 리포지토리에 추가할 수 있는 방법을 제한하십시오.

직접 게시와 같은 내부 소스와 공개 리포지토리와 같은 외부 소스에서 서로 다른 패키지의 새 버전을 가져오도록 패키지 오리진 제어를 구성하는 것을 고려해야 합니다. 기본적으로 패키지 오리진 제어는 패키지의 첫 번째 버전이 저장소에 추가되는 방식을 기반으로 구성됩니다.

패키지 원본 제어 설정

패키지 원본 제어를 사용하여 패키지 버전을 리포지토리에 추가하는 방법을 구성할 수 있습니다. 다음 목록에는 사용 가능한 패키지 원본 제어 설정 및 값이 포함되어 있습니다.

게시

이 설정은 패키지 관리자 또는 이와 유사한 도구를 사용하여 패키지 버전을 리포지토리에 직접 게시할 수 있는지를 구성합니다.

- ALLOW: 패키지 버전을 직접 게시할 수 있습니다.
- BLOCK: 패키지 버전은 직접 게시할 수 없습니다.

업스트림

이 설정은 패키지 관리자가 요청하는 경우 패키지 버전을 외부 또는 퍼블릭 리포지토리에서 수집하거나 업스트림 리포지토리에서 유지할 수 있는지를 구성합니다.

- ALLOW: 모든 패키지 버전은 업스트림 CodeCatalyst 리포지토리로 구성된 다른 리포지토리에서 보존하거나 외부 연결을 통해 퍼블릭 소스에서 수집할 수 있습니다.
- BLOCK: 패키지 버전은 업스트림 CodeCatalyst 리포지토리로 구성된 다른 리포지토리에서 보존하거나 외부 연결을 통해 퍼블릭 소스에서 인제스트할 수 없습니다.

기본 패키지 원본 제어 설정

패키지의 기본 패키지 오리진 제어는 해당 패키지의 첫 번째 버전이 패키지 저장소에 추가되는 방식을 기반으로 합니다.

- 패키지 관리자가 첫 번째 패키지 버전을 직접 게시하는 경우 설정은 Publish: ALLOW 및 Upstream: 이 됩니다. BLOCK
- 첫 번째 패키지 버전을 공개 소스에서 인제스트하는 경우 설정은 Publish: BLOCK 및 Upstream:이 됩니다. ALLOW

일반적인 패키지 액세스 제어 시나리오

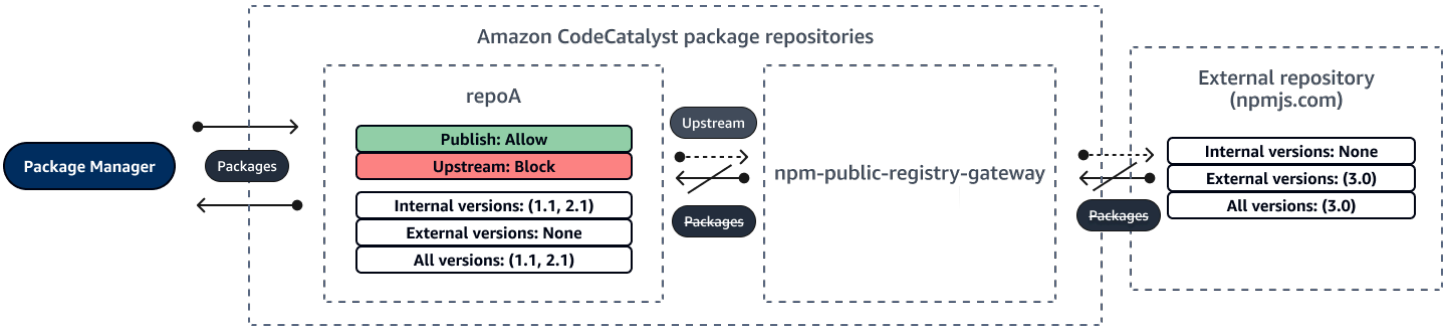
이 섹션에서는 패키지 버전이 패키지 저장소에 추가되는 몇 가지 일반적인 시나리오를 설명합니다. CodeCatalyst 첫 번째 패키지 버전이 추가되는 방식에 따라 새 패키지에 대한 Package 원본 제어 설정이 설정됩니다.

다음 시나리오에서는 유지 관리하는 패키지와 같은 내부 패키지가 패키지 관리자에서 저장소에 직접 게시됩니다. 외부 패키지는 게이트웨이 리포지토리 업스트림을 통해 리포지토리로 인제스트될 수 있는 공용 리포지토리에 있는 패키지입니다.

외부 패키지 버전은 기존 내부 패키지를 대상으로 게시됩니다.

이 시나리오에서는 내부 패키지인 packageA를 가정합니다. 팀에서 PackageA의 첫 번째 패키지 버전을 패키지 저장소에 게시합니다. CodeCatalyst 이 패키지의 첫 번째 패키지 버전이므로, 패키지 원본 제어 설정은 게시: 허용 및 업스트림: 차단으로 자동 설정됩니다. 패키지가 저장소에 게시되면 패키지 저장소에 연결된 공용 저장소에 같은 이름의 패키지가 게시됩니다. CodeCatalyst 이는 내부 패키지에 대한 종속성 대체 공격 시도일 수도 있고 우연의 일치일 수도 있습니다. 하지만 패키지 원본 제어는 새로운 외부 버전의 수집을 차단하여 잠재적 공격을 방어하도록 구성되어 있습니다.

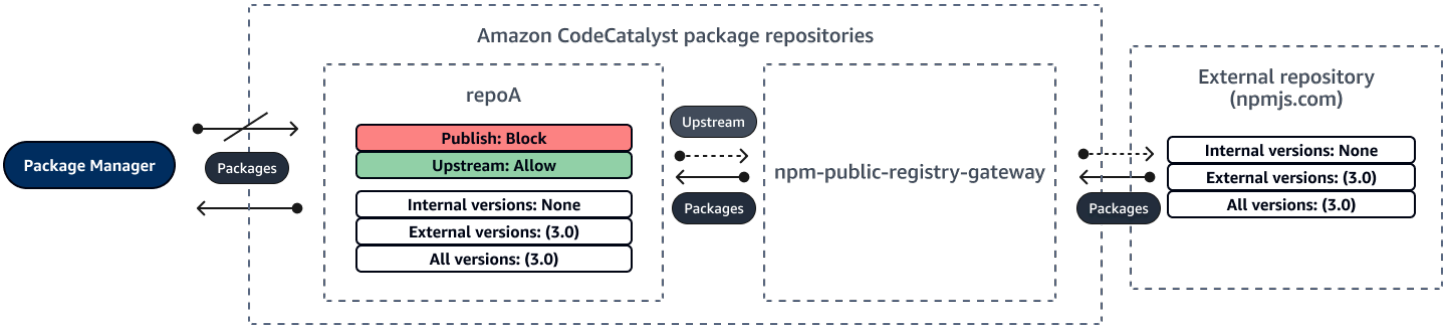
다음 이미지에서 RePOA는 저장소에 대한 업스트림 연결이 있는 CodeCatalyst 패키지 저장소입니다. `npm-public-registry-gateway` 리포지토리에 packageA 버전 1.1 및 2.1이 있지만 버전 3.0은 퍼블릭 리포지토리에 게시됩니다. 일반적으로 RePOA는 패키지 관리자가 패키지를 요청한 후 버전 3.0을 수집합니다. 패키지 통합이 차단으로 설정되어 있기 때문에 버전 3.0은 패키지 저장소에 인제스트되지 않으며 연결된 CodeCatalyst 패키지 관리자가 버전 3.0을 사용할 수 없습니다.



내부 패키지 버전은 기존 외부 패키지를 대상으로 게시됩니다.

이 시나리오에서는 PackageB 패키지가 저장소에 연결된 공용 저장소 외부에 존재합니다. 리포지토리에 연결된 패키지 관리자가 packageB를 요청하면 해당 패키지 버전이 퍼블릭 리포지토리에서 사용자의 리포지토리로 수집됩니다. 이 패키지는 저장소에 추가된 PackageB의 첫 번째 패키지 버전이므로 패키지 오리진 설정은 Publish: 및 Upstream:으로 구성됩니다. BLOCK ALLOW 나중에 패키지 이름이 동일한 버전을 리포지토리에 게시합니다. 공용 패키지를 모르면서 관련 없는 패키지를 같은 이름으로 게시하려고 하거나, 패치가 적용된 버전을 게시하려고 하거나, 이미 외부에 있는 정확한 패키지 버전을 직접 게시하려고 할 수 있습니다. CodeCatalyst 게시하려는 버전을 거부하지만 거부를 명시적으로 무시하고 필요한 경우 버전을 게시할 수 있습니다.

다음 이미지에서 RePOA는 저장소에 대한 업스트림 연결이 있는 CodeCatalyst 패키지 저장소입니다. `npm-public-registry-gateway` 패키지 저장소에는 공용 저장소에서 수집한 버전 3.0이 포함되어 있습니다. 버전 1.2를 패키지 저장소에 게시하고 싶습니다. 일반적으로 버전 1.2를 RepoA에 게시할 수 있지만 게시가 차단으로 설정되어 있기 때문에 버전 1.2는 게시할 수 없습니다.



기존 외부 패키지의 패치가 적용된 패키지 버전 게시

이 시나리오에서 PackageB 패키지는 패키지 저장소에 연결된 공용 저장소 외부에 존재합니다. 리포지토리에 연결된 패키지 관리자가 packageB를 요청하면 해당 패키지 버전이 퍼블릭 리포지토리에서 사용자의 리포지토리로 수집됩니다. 이 패키지는 저장소에 추가된 PackageB의 첫 번째 패키지 버전이므로 패키지 오리진 설정은 Publish: 및 Upstream:으로 구성됩니다. BLOCK ALLOW 팀에서 이 패키지의 패치가 적용된 패키지 버전을 저장소에 게시하기로 결정합니다. 패키지 버전을 직접 게시할 수 있도록 팀은 패키지 원본 제어 설정을 Publish: ALLOW 및 Upstream:으로 변경합니다. BLOCK 이제 이 패키지의 버전을 리포지토리에 직접 게시하고 퍼블릭 리포지토리에서 수집할 수 있습니다. 팀에서 패치된 패키지 버전을 게시한 후 팀은 패키지 오리진 설정을 Publish: BLOCK 및 Upstream:으로 되돌립니다. ALLOW

패키지 원본 제어 편집

패키지 오리진 제어는 패키지의 첫 번째 패키지 버전이 패키지 저장소에 추가되는 방식에 따라 자동으로 구성됩니다. 자세한 내용은 기본 패키지 원본 제어 설정 단원을 참조하십시오. 패키지 저장소의 패키지에 대한 패키지 오리진 제어를 추가하거나 편집하려면 다음 절차의 단계를 수행하십시오. CodeCatalyst

패키지 오리진 컨트롤 추가 또는 편집하기

1. 탐색 창에서 Packages(패키지)를 선택합니다.
2. 편집하려는 패키지가 들어 있는 패키지 저장소를 선택합니다.
3. 패키지 테이블에서 편집하려는 패키지를 검색하여 선택합니다.
4. 패키지 요약 페이지에서 Origin 제어를 선택합니다.
5. Origin 컨트롤에서 이 패키지에 설정하려는 패키지 오리진 제어를 선택합니다. 패키지 오리진 컨트롤 설정인 퍼블리시와 업스트림을 동시에 설정해야 합니다.

   - 패키지 버전을 직접 게시할 수 있도록 허용하려면 게시에서 허용을 선택합니다. 패키지 버전 게시를 차단하려면 차단을 선택합니다.

- 외부 리포지토리에서 패키지를 수집하고 업스트림 리포지토리에서 패키지를 가져오도록 허용하려면 업스트림 소스에서 허용을 선택합니다. 외부 및 업스트림 리포지토리에서의 패키지 버전 수집과 가져오기를 모두 차단하려면 차단을 선택합니다.

6. 저장(Save)을 선택합니다.

리포지토리 게시 및 업스트림

CodeCatalyst에서는 연결 가능한 업스트림 리포지토리 또는 퍼블릭 리포지토리에 있는 패키지 버전을 게시할 수 없습니다. 예를 들어 npm 패키지를 리포지토리에 게시하려고 하는데 npmjs.com에 대한 외부 연결이 있는 `lodash@1.0` 업스트림 리포지토리가 있다고 가정해 `myrepo` 보겠습니다. `myrepo` 다음 시나리오를 고려해 보세요.

1. 패키지 오리진 제어 설정은 게시: 및 업스트림:입니다. **`lodash`** ALLOW ALLOW 업스트림 리포지토리 또는 npmjs.com에 있는 경우 `lodash@1.0` 409 충돌 오류가 발생하여 게시 시도를 CodeCatalyst 거부합니다. `myrepo` `lodash@1.1` 같은 다른 버전은 게시할 수 있습니다.
2. 패키지 오리진 제어 설정은 Publish: 및 Upstream: 입니다. **`lodash`** ALLOW BLOCK 패키지 버전에 연결할 수 없으므로 아직 존재하지 않는 모든 버전을 저장소에 게시할 수 있습니다. `lodash`
3. 의 패키지 원본 제어 `lodash` 설정은 게시: BLOCK 및 업스트림:입니다. ALLOW 패키지 버전은 리포지토리에 직접 게시할 수 없습니다.

종속성 대체 공격

패키지 관리자는 재사용 가능한 코드를 패키징하고 공유하는 프로세스를 단순화합니다. 이러한 패키지는 애플리케이션에서 사용하기 위해 조직에서 개발한 프라이빗 패키지일 수도 있고, 조직 외부에서 개발되어 퍼블릭 패키지 리포지토리에서 배포하는 퍼블릭(대부분의 경우 오픈 소스) 패키지일 수도 있습니다. 패키지를 요청할 때 개발자는 패키지 관리자를 이용해 종속 항목의 새 버전을 가져옵니다. 종속성 혼동 공격이라고도 하는 종속성 대체 공격은 대부분의 패키지 관리자가 패키지의 합법적인 버전과 악성 버전을 구분할 방법이 없다는 사실을 악용합니다.

종속성 대체 공격은 소프트웨어 공급망 공격으로 알려진 공격의 일부에 속합니다. 소프트웨어 공급망 공격은 소프트웨어 공급망에 존재하는 취약성을 악용하는 공격입니다.

종속성 대체 공격은 내부적으로 개발된 패키지와 퍼블릭 리포지토리에서 가져온 패키지를 모두 사용하는 사용자라면 누구나 노릴 수 있습니다. 공격자는 내부 패키지 이름을 식별한 다음 같은 이름의 악성 코드를 전략적으로 퍼블릭 패키지 리포지토리에 배치합니다. 일반적으로 악성 코드는 버전 번호가 높은 패키지에 게시됩니다. 패키지 관리자는 악성 패키지가 패키지 최신 버전이라고 생각하기 때문에

이러한 퍼블릭 피드에서 악성 코드를 가져옵니다. 이로 인해 원하는 패키지와 악성 패키지 간에 “혼동” 또는 “대체”가 발생하여 코드가 손상될 수 있습니다.

종속성 대체 공격을 방지하기 위해 CodeCatalyst Amazon은 패키지 오리진 제어를 제공합니다. 패키지 원본 제어는 패키지를 리포지토리에 추가하는 방법을 제어하는 설정입니다. 컨트롤은 새 패키지의 첫 번째 패키지 버전을 리포지토리에 추가할 때 자동으로 구성됩니다. 제어를 통해 패키지 버전을 CodeCatalyst 리포지토리에 직접 게시하거나 퍼블릭 소스에서 수집할 수 없도록 보장하여 종속성 대체 공격으로부터 보호할 수 있습니다. 패키지 원본 제어 및 이를 변경하는 방법에 대한 자세한 내용은 패키지 원본 제어 편집 섹션을 참조하세요.

npm 사용

이 항목에서는 Node.js 패키지 관리자를 사용하여 사용할 npm 수 있는 방법을 설명합니다. CodeCatalyst

> **Note**
>
> CodeCatalyst 지원 `node v4.9.1` 및 이후 버전 `npm v5.0.0` 및 이후 버전

주제

- npm 구성 및 사용
- npm 태그 처리

npm 구성 및 사용

npm와 함께 CodeCatalyst 사용하려면 패키지 저장소에 연결하고 npm 인증을 위한 개인 액세스 토큰 (PAT) 을 제공해야 합니다. CodeCatalyst 콘솔에서 패키지 리포지토리에 npm 연결하는 방법을 확인할 수 있습니다.

목차

- npm을 다음과 같이 구성합니다. CodeCatalyst
- 패키지 저장소에서 npm 패키지 설치 CodeCatalyst
- npm.js에서 npm 패키지를 설치하는 방법: CodeCatalyst
- npm 패키지를 패키지 저장소에 게시 CodeCatalyst

npm을 다음과 같이 구성합니다. CodeCatalyst

다음 지침은 CodeCatalyst 패키지 저장소를 인증하고 연결하는 npm 방법을 설명합니다. npm에 대한 자세한 내용은 공식 npm 설명서를 참조하십시오.

패키지 저장소에 **npm** 연결하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다.
3. 탐색 창에서 Packages(패키지)를 선택합니다.
4. 목록에서 패키지 저장소를 선택합니다.
5. 리포지토리에 연결을 선택합니다.
6. 구성 세부 사항의 패키지 관리자 클라이언트에서 npm client를 선택합니다.
7. 운영 체제를 선택하여 해당 구성 단계를 확인하세요.
8. npm을 인증하려면 개인 액세스 토큰 (PAT) 이 필요합니다. CodeCatalyst 이미 토큰이 있다면 사용할 수 있습니다. 없는 경우 다음 단계를 사용하여 생성할 수 있습니다.

   a. (선택 사항): PAT 이름 및 만료일을 업데이트합니다.

   b. 토큰 생성을 선택합니다.

   c. 복사하여 안전한 PAT 장소에 보관하세요.

   > **Warning**
   >
   > 대화 상자를 닫은 후에는 PAT 다시 보거나 복사할 수 없습니다. 공격자가 자격 증명을 도용한 후 사용할 수 있는 시간을 최소화하려면 자격 증명은 수명이 짧아야 합니다.

9. 프로젝트의 루트 디렉터리에서 다음 명령을 실행하여 패키지 저장소로 npm을 구성하세요. 명령은 다음과 같은 작업을 수행합니다.

   - 프로젝트에 프로젝트 수준 `.npmrc` 파일이 없는 경우 프로젝트 수준 파일을 만드세요.

- 패키지 리포지토리 엔드포인트 정보를 프로젝트 `.npmrc` 수준 파일에 추가합니다.
- 사용자 `.npmrc` 수준 파일에 자격 증명 (PAT) 을 추가합니다.

다음 값을 바꾸십시오.

> **Note**
>
> 콘솔 지침에서 복사하는 경우 다음 명령의 값이 자동으로 업데이트되므로 변경할 필요가 없습니다.

- Replace *`username`* CodeCatalyst 사용자 이름과 함께.
- Replace *`PAT`* 당신과 함께 CodeCatalyst PAT.
- Replace *`space_name`* CodeCatalyst 스페이스 이름과 함께.
- Replace *`proj_name`* CodeCatalyst 프로젝트 이름과 함께.
- Replace *`repo_name`* CodeCatalyst 패키지 리포지토리 이름으로.

```
npm set registry=https://packages.region.codecatalyst.aws/npm/space-name/proj-
name/repo-name/ --location project
npm set //packages.region.codecatalyst.aws/npm/space-name/proj-name/repo-
name/:_authToken=username:PAT
```

npm 6 이하의 경우: `GET` 요청의 경우에도 npm이 항상 인증 토큰을 전달하도록 하려면 CodeCatalyst always-auth 구성 변수를 다음과 같이 설정하십시오. `npm config set`

```
npm set //packages.region.codecatalyst.aws/npm/space-name/proj-name/repo-
name/:always-auth=true --location project
```

패키지 저장소에서 npm 패키지 설치 CodeCatalyst

의 단계에 따라 npm을 리포지토리에 연결한 후 리포지토리에서 npm을 다음과 같이 구성합니다. CodeCatalyst npm 명령을 실행할 수 있습니다.

명령을 사용하여 CodeCatalyst 패키지 리포지토리 또는 업스트림 리포지토리 중 하나에 있는 npm 패키지를 설치할 수 있습니다. `npm install`

```
npm install lodash
```

npm.js에서 npm 패키지를 설치하는 방법: CodeCatalyst

npmjs.com에 연결된 게이트웨이 리포지토리에 대한 업스트림 연결로 리포지토리를 구성하여 CodeCatalyst 리포지토리를 통해 npmjs.com에서 npm 패키지를 설치할 수 있습니다. npm-public-registry-gateway npmjs에서 설치한 패키지는 게이트웨이 리포지토리와 가장 멀리 떨어진 다운스트림 패키지 리포지토리에 수집되어 저장됩니다.

npmjs에서 패키지를 설치하려면

1. 아직 구성하지 않았다면 의 단계에 따라 CodeCatalyst 패키지 npm 저장소로 구성하세요. npm을 다음과 같이 구성합니다. CodeCatalyst
2. 리포지토리가 게이트웨이 리포지토리를 업스트림 연결로 추가했는지 확인하세요. npm-public-registry-gateway 의 지침에 따라 리포지토리를 선택하면 어떤 업스트림 소스가 업스트림 npm-public-registry-gateway소스로 추가되거나 추가되는지 확인할 수 있습니다. 업스트림 리포지토리 추가 npm-public-registry-gateway
3. 명령을 사용하여 패키지를 설치합니다. `npm install`

   ```
   npm install package_name
   ```

업스트림 리포지토리에서 패키지를 요청하는 방법에 대한 자세한 내용은 을 참조하십시오. 업스트림 리포지토리가 포함된 패키지 버전 요청

npm 패키지를 패키지 저장소에 게시 CodeCatalyst

작업을 npm을 다음과 같이 구성합니다. CodeCatalyst 완료한 후 npm 명령을 실행할 수 있습니다.

`npm publish`명령을 사용하여 npm 패키지를 CodeCatalyst 패키지 저장소에 게시할 수 있습니다.

```
npm publish
```

npm 패키지를 만드는 방법에 대한 자세한 내용은 npm Docs에서 Node.js 모듈 만들기를 참조하십시오.

npm 명령 지원

다음 섹션에서는 지원되지 않는 특정 npm 명령을 나열하는 것 외에도 CodeCatalyst 패키지 리포지토리에서 지원하는 명령을 요약합니다.

주제

- 패키지 리포지토리와 상호 작용하는 지원되는 명령
- 지원되는 클라이언트 측 명령
- 지원되지 않는 명령

패키지 리포지토리와 상호 작용하는 지원되는 명령

이 섹션에는 npm 클라이언트가 구성된 레지스트리에 하나 이상의 요청을 보내는 npm 명령 (예:npm `config set registry`) 이 나열되어 있습니다. 이러한 명령은 CodeCatalyst 패키지 저장소에 대해 호출했을 때 제대로 작동하는 것으로 확인되었습니다.

Command	설명
bugs	패키지의 버그 추적기 URL 위치를 추측한 다음 패키지를 열려고 시도합니다.
ci	프로젝트를 새로 다시 설치합니다.
deprecate	패키지 버전을 더 이상 사용하지 않습니다.
dist-tag	패키지 배포 태그를 수정합니다.
docs	패키지 설명서의 위치를 추측한 다음 URL `--browser` config 매개 변수를 사용하여 문서를 열려고 시도합니다.
doctor	일련의 검사를 실행하여 npm 설치가 패키지를 관리할 수 있는지 확인합니다. JavaScript
install	패키지를 설치합니다.
install-ci-test	프로젝트를 새로 다시 설치하고 테스트를 실행합니다. 별칭: `npm cit`. 이 명령은 를 실행한 `npm ci` 다음 즉시 를 `npm test` 실행합니다.

Command	설명
install-test	패키지를 설치하고 테스트를 실행합니다. npm installan을 실행하고 바로 뒤에 an을 실행합니다npm test.
outdated	구성된 레지스트리를 검사하여 설치된 패키지가 오래되었는지 확인합니다.
ping	구성되거나 지정된 npm 레지스트리를 ping하고 인증을 확인합니다.
publish	패키지 버전을 레지스트리에 게시합니다.
update	패키지 URL 리포지토리의 위치를 추측한 다음 --browser config 매개 변수를 사용하여 패키지를 열려고 시도합니다.
view	패키지 메타데이터를 표시합니다. 메타데이터 속성을 인쇄하는 데에도 사용할 수 있습니다.

지원되는 클라이언트 측 명령

이러한 명령은 패키지 저장소와 직접 상호 작용할 필요가 CodeCatalyst 없으므로 명령을 지원하는 데 필요한 것이 없습니다.

Command	설명
bin (레거시)	npm bin 디렉토리를 표시합니다.
build	패키지를 빌드합니다.
cache	패키지 캐시를 조작합니다.
completion	모든 npm 명령에서 탭 완성을 활성화합니다.
config	사용자 및 글로벌 npmrc 파일의 내용을 업데이트합니다.

Command	설명
dedupe	로컬 패키지 트리를 검색하고 여러 종속 패키지에서 종속성을 더 효과적으로 공유할 수 있도록 트리 위로 종속성을 이동하여 구조를 단순화하려고 합니다.
edit	설치된 패키지를 편집합니다. 현재 작업 디렉터리에서 종속성을 선택하고 기본 편집기에서 패키지 디렉터리를 엽니다.
explore	설치된 패키지를 찾아봅니다. 지정된 설치 패키지의 디렉터리에 서브셸을 생성합니다. 명령이 지정되면 해당 명령은 서브셸에서 실행되며, 서브셸은 즉시 종료됩니다.
help	npm에 관한 도움말을 가져옵니다.
help-search	npm 도움말 설명서를 검색합니다.
init	`package.json` 파일을 생성합니다.
link	패키지 디렉터리를 심볼릭 링크합니다.
ls	설치된 패키지를 나열합니다.
pack	패키지에서 tarball을 생성합니다.
접두사	접두사를 표시합니다. 별도로 지정하지 않는 한 `-g` 이 디렉토리는 `package.json` 파일을 포함하는 가장 가까운 상위 디렉토리입니다.
prune	상위 패키지의 종속성 목록에 나열되지 않은 패키지를 제거합니다.
rebuild	일치하는 폴더에서 `npm build` 명령을 실행합니다.
restart	패키지의 중지, 재시작, 시작 스크립트와 관련 사전 스크립트 및 사후 스크립트를 실행합니다.

Command	설명
root	유효 node_modules 디렉토리를 표준 출력으로 출력합니다.
run-script	임의의 패키지 스크립트를 실행합니다.
shrinkwrap	게시할 종속 버전을 잠급니다.
uninstall	패키지를 제거합니다.

지원되지 않는 명령

CodeCatalyst 패키지 저장소에서는 이러한 npm 명령을 지원하지 않습니다.

Command	설명	참고
access	게시된 패키지에서 액세스 수준을 설정합니다.	CodeCatalyst 공개 npmjs 저장소와는 다른 권한 모델을 사용합니다.
adduser	레지스트리 사용자 계정을 추가합니다.	CodeCatalyst 공개 npmjs 저장소와는 다른 사용자 모델을 사용합니다.
audit	보안 감사를 실행합니다.	CodeCatalyst 는 현재 보안 취약성 데이터를 판매하지 않습니다.
hook	추가, 제거, 나열 및 업데이트를 포함하여 npm 후크를 관리합니다.	CodeCatalyst 현재 변경 알림 메커니즘을 지원하지 않습니다.
login	사용자를 인증합니다. npm adduser에 대한 별칭입니다.	CodeCatalyst 공개 npmjs 저장소와는 다른 인증 모델을 사용합니다. 자세한 내용은 npm 을 다음과 같이 구성합니다. CodeCatalyst 을 참조하세요.

Command	설명	참고
logout	레지스트리에서 로그아웃합니다.	CodeCatalyst 공개 npmjs 저장소와는 다른 인증 모델을 사용합니다. CodeCatalyst 저장소에서 로그아웃할 수 있는 방법은 없지만 인증 토큰은 구성 가능한 만료 시간이 지나면 만료됩니다. 기본 토큰 지속 시간은 12시간입니다.
owner	패키지 소유자를 관리합니다.	CodeCatalyst 공개 npmjs 저장소와는 다른 권한 모델을 사용합니다.
profile	레지스트리 프로필의 설정을 변경합니다.	CodeCatalyst 공개 npmjs 저장소와는 다른 사용자 모델을 사용합니다.
search	레지스트리에서 검색어와 일치하는 패키지를 검색합니다.	CodeCatalyst 명령을 `search` 지원하지 않습니다.
star	좋아하는 패키지를 표시합니다.	CodeCatalyst 현재 어떤 즐겨찾기 메커니즘도 지원하지 않습니다.
stars	즐겨찾기로 표시된 패키지를 조회합니다.	CodeCatalyst 현재 어떤 즐겨찾기 메커니즘도 지원하지 않습니다.
team	팀과 팀 멤버십을 관리합니다.	CodeCatalyst 공개 npmjs 저장소와는 다른 사용자 및 그룹 구성원 모델을 사용합니다.

Command	설명	참고
token	인증 토큰을 관리합니다.	CodeCatalyst 인증 토큰을 가져오는 데 다른 모델을 사용합니다. 자세한 내용은 npm을 다음과 같이 구성합니다. CodeCatalyst 을 참조하세요.
unpublish	레지스트리에서 패키지를 제거합니다.	CodeCatalyst npm 클라이언트를 사용하여 저장소에서 패키지 버전을 제거하는 것을 지원하지 않습니다. 콘솔에서 패키지를 삭제할 수 있습니다.
whoami	npm 사용자 이름을 표시합니다.	CodeCatalyst 공개 npmjs 저장소와는 다른 사용자 모델을 사용합니다.

npm 태그 처리

npm 레지스트리는 패키지 버전의 문자열 별칭인 태그를 지원합니다. 버전 번호를 사용하는 대신 태그를 사용하여 별칭을 제공할 수 있습니다. 예를 들어, 개발 스트림이 여러 개 있는 프로젝트에서 각 스트림마다 다른 태그 (예:, `stable` `betadev`,`canary`) 를 사용합니다. 자세한 내용은 npm Docs의 dist-tag를 참조하십시오.

기본적으로 npm은 `latest` 태그를 사용하여 패키지의 현재 버전을 식별합니다. `npm install` *`pkg`*(`@`*`version`* 또는 `@`*`tag`* 지정자가 없는)는 최신 태그를 설치합니다. 일반적으로 프로젝트는 안정적인 릴리스 버전의 경우 최신 태그만 사용합니다. 그 밖의 태그는 불안정한 버전이나 프리릴리스 버전에 사용됩니다.

npm 클라이언트로 태그 편집

세 가지 `npm dist-tag` 명령 (`add`,`rm`,`ls`) 은 기본 npm 레지스트리에서 작동하는 것과 동일한 방식으로 CodeCatalyst 패키지 저장소에서도 작동합니다.

npm 태그와 업스트림 리포지토리

npm요청 시 패키지에 대한 태그와 해당 패키지의 버전이 업스트림 저장소에도 존재하면 는 태그를 CodeCatalyst 병합한 다음 클라이언트에 반환합니다. 예를 들어, 이름이 지정된 R 저장소에는 이름이 지정된 업스트림 저장소가 있습니다. U 다음 표에는 두 리포지토리에 모두 web-helper 있는 이름이 지정된 패키지의 태그가 나와 있습니다.

리포지토리	패키지 이름	패키지 태그
R	web-helper	latest(버전 1.0.0의 별칭)
U	web-helper	alpha(버전 1.0.1의 별칭)

이 경우 npm 클라이언트가 R 리포지토리에서 web-helper 패키지 태그를 가져오면 최신 태그와 알파 태그를 모두 받습니다. 태그가 가리키는 버전은 변경되지 않습니다.

업스트림 리포지토리와 로컬 리포지토리의 동일한 패키지에 동일한 태그가 있는 경우 는 마지막으로 업데이트된 태그를 CodeCatalyst 사용합니다. 예를 들어, webhelper의 태그가 다음과 같이 수정되었다고 가정해 보겠습니다.

리포지토리	패키지 이름	패키지 태그	최종 업데이트 날짜
R	web-helper	latest(버전 1.0.0의 별칭)	2023년 1월 1일
U	web-helper	latest(버전 1.0.1의 별칭)	2023년 6월 1일

이 경우 npm 클라이언트가 *R* 저장소에서 패키지 web-helper의 태그를 가져오면 가장 최근에 업데이트된 버전이므로 최신 태그가 버전 1.0.1에 별칭을 붙입니다. 이렇게 하면 로컬 리포지토리에 아직 없는 업스트림 리포지토리의 새 패키지 버전을 실행하여 쉽게 사용할 수 있습니다. npm update

Maven 사용

Maven 리포지토리 형식은 Java, Kotlin, Scala, Clojure를 비롯한 다양한 언어에서 사용됩니다. Maven, Gradle, ScalaSBT, Apache Ivy, Leiningen을 비롯한 다양한 빌드 도구에서 지원됩니다.

다음 버전과의 호환성을 테스트하고 확인했습니다. CodeCatalyst

- 최신 Maven 버전: 3.6.3.
- 최신 Gradle 버전: 6.4.1. 버전 5.5.1도 테스트되었습니다.

주제

그라들 그루비 구성 및 사용

Gradle Groovy를 함께 CodeCatalyst 사용하려면 Gradle Groovy를 패키지 저장소에 연결하고 인증을 위한 개인 액세스 토큰 () 을 제공해야 합니다. PAT 콘솔에서 Gradle Groovy를 패키지 리포지토리에 연결하는 지침을 확인할 수 있습니다. CodeCatalyst

목차

에서 종속성 가져오기 CodeCatalyst

다음 지침은 패키지 리포지토리의 종속성을 가져오도록 Gradle Groovy를 구성하는 방법을 설명합니다. CodeCatalyst

Gradle Groovy를 사용하여 패키지 저장소에서 종속 항목을 가져오려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 콘솔을 엽니다. CodeCatalyst
2. 프로젝트로 이동합니다.

3. 탐색 창에서 Packages(패키지)를 선택합니다.
4. 패키지 리포지토리 목록에서 패키지 리포지토리를 선택합니다.
5. 리포지토리에 연결을 선택합니다.
6. 리포지토리에 연결 대화 상자의 패키지 관리자 클라이언트 목록에서 Gradle Groovy를 선택합니다.
7. Gradle Groovy를 인증하려면 개인 액세스 토큰 (PAT) 이 필요합니다. CodeCatalyst 이미 가지고 있다면 사용할 수 있습니다. 없다면 여기에서 새로 만들 수 있습니다.

   a. 토큰 생성을 선택합니다.

   b. 복사를 선택하여 PAT 복사합니다.

   > **Warning**
   >
   > 대화 상자를 닫은 후에는 PAT 다시 보거나 복사할 수 없습니다.

8. 액세스 자격 증명으로 gradle 속성 파일을 업데이트하세요. Replace *username* CodeCatalyst 사용자 이름으로 바꾸고 바꾸세요.*PAT* CodeCatalyst 개인 액세스 토큰으로 다음 용도로는 어떤 값이든 사용할 수 있습니다.*spaceUsername* 그리고 *spacePassword* 다음 단계에서 동일한 값을 사용하기만 하면 됩니다.

   ```
   spaceUsername=username
   spacePassword=PAT
   ```

9. Gradle CodeCatalyst 빌드에서 종속 항목을 가져오려면 `maven` 코드 스니펫을 복사하여 프로젝트 파일의 `repositories` 섹션에 추가하세요. `build.gradle` 다음 값을 바꾸세요. 다음 용도로는 어떤 값이든 사용할 수 있습니다.*spaceName* 다음 단계에서 동일한 값을 사용하기만 하면 됩니다.

   > **Note**
   >
   > 콘솔 지침에서 복사하는 경우 다음 값을 자동으로 업데이트해야 하며 변경해서는 안 됩니다.

   - Replace *space_name* CodeCatalyst스페이스 이름과 함께.
   - Replace *proj_name* CodeCatalyst프로젝트 이름과 함께
   - Replace *repo_name* CodeCatalyst패키지 리포지토리 이름과 함께

```
maven {
  name = 'spaceName'
  url = uri('https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/')
  credentials(PasswordCredentials)
}
```

10. (선택 사항) CodeCatalyst 패키지 리포지토리를 프로젝트 종속성의 유일한 소스로 사용하려면 파일에서 리포지토리의 다른 섹션을 모두 제거합니다. `build.gradle` 리포지토리가 두 개 이상인 경우, Gradle은 나열된 순서대로 각 리포지토리에서 종속성을 검색합니다.

에서 플러그인 가져오기 CodeCatalyst

기본적으로 Gradle은 퍼블릭 Gradle 플러그인 포털에서 플러그인을 확인합니다. 다음 단계는 패키지 저장소의 플러그인을 해결하도록 Gradle 프로젝트를 구성합니다. CodeCatalyst

Gradle을 사용하여 패키지 저장소에서 플러그인을 가져오려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다.
3. 탐색 창에서 Packages(패키지)를 선택합니다.
4. 패키지 리포지토리 목록에서 패키지 리포지토리를 선택합니다.
5. 리포지토리에 연결을 선택합니다.
6. 리포지토리에 연결 대화 상자의 패키지 관리자 클라이언트 목록에서 Gradle을 선택합니다.
7. Gradle을 인증하려면 개인 액세스 토큰 (PAT) 이 필요합니다. CodeCatalyst 이미 가지고 있다면 사용할 수 있습니다. 없다면 여기에서 새로 만들 수 있습니다.

    a. 토큰 생성을 선택합니다.

    b. 복사를 선택하여 PAT 복사합니다.

    > **Warning**
    >
    > 대화 상자를 닫은 후에는 PAT 다시 보거나 복사할 수 없습니다.

8. 액세스 자격 증명으로 gradle 속성 파일을 업데이트하세요. Replace *username* CodeCatalyst 사용자 이름으로 바꾸고 바꾸세요.*PAT* CodeCatalyst 개인 액세스 토큰으로 다음 용도로는 어떤 값

이든 사용할 수 있습니다.*spaceUsername* 그리고 *spacePassword* 다음 단계에서 동일한 값을 사용하기만 하면 됩니다.

```
spaceUsername=username
spacePassword=PAT
```

9. `settings.gradle` 파일에 `pluginManagement` 블록을 추가합니다. `pluginManagement` 블록은 `settings.gradle`에서 다른 문 앞에 나타나야 합니다. 다음 값을 바꾸십시오.

> **Note**
>
> 콘솔 지침에서 복사하는 경우 다음 값을 자동으로 업데이트해야 하며 변경해서는 안 됩니다.

- Replace *spaceName* 이전 단계에서 사용한 이름 값을 사용합니다.
- Replace *space_name* CodeCatalyst스페이스 이름과 함께
- Replace *proj_name* CodeCatalyst프로젝트 이름과 함께
- Replace *repo_name* CodeCatalyst패키지 리포지토리 이름과 함께

```
pluginManagement {
    repositories {
        maven {
            name = 'spaceName'
            url = uri('https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/')
            credentials(PasswordCredentials)
        }
    }
}
```

이렇게 하면 Gradle이 지정된 리포지토리의 플러그인을 확인할 수 있습니다. 일반적으로 필요한 Gradle 플러그인을 빌드에서 사용할 수 있으려면 리포지토리에 Gradle Plugin Portal (`gradle-plugins-store`) 으로 구성된 업스트림 연결이 있어야 합니다. 자세한 내용은 Gradle 설명서를 참조하세요.

외부 패키지 리포지토리를 통해 패키지를 가져오는 중 CodeCatalyst

게이트웨이 리포지토리를 나타내는 게이트웨이에 대한 업스트림 연결로 구성하여 리포지토리를 통해 공용 CodeCatalyst 리포지토리에서 Maven 패키지를 설치할 수 있습니다. 게이트웨이 리포지토리에서 설치한 패키지는 리포지토리에 인제스트되어 저장됩니다. CodeCatalyst

CodeCatalyst 다음과 같은 공개 Maven 패키지 리포지토리를 지원합니다.

- maven-central-gateway
- google-android-gateway
- gradle-plugins-gateway
- 커먼웨어 게이트웨이

공개 Maven 패키지 리포지토리에서 패키지를 설치하려면

1. 아직 구성하지 않았다면 또는 의 단계에 따라 CodeCatalyst 패키지 저장소로 Gradle을 구성하세요. 에서 종속성 가져오기 CodeCatalyst 에서 플러그인 가져오기 CodeCatalyst
2. 업스트림 연결로 설치하려는 게이트웨이 리포지토리가 저장소에 추가되었는지 확인하세요. 의 지침에 따라 업스트림으로 추가할 공개 패키지 저장소를 선택하면 이 작업을 수행할 수 있습니다. 업스트림 리포지토리 추가

업스트림 리포지토리에서 패키지를 요청하는 방법에 대한 자세한 내용은 을 참조하십시오. 업스트림 리포지토리가 포함된 패키지 버전 요청

패키지 게시 위치 CodeCatalyst

이 섹션에서는 Gradle Groovy로 빌드한 자바 라이브러리를 리포지토리에 게시하는 방법을 설명합니다. CodeCatalyst

Gradle Groovy를 사용하여 패키지 리포지토리에 패키지를 게시하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트 개요 페이지에서 패키지를 선택합니다.
3. 패키지 리포지토리 목록에서 패키지 리포지토리를 선택합니다.
4. 리포지토리에 연결을 선택합니다.
5. 리포지토리에 연결 대화 상자의 패키지 관리자 클라이언트 목록에서 Gradle Groovy를 선택합니다.

6. Gradle을 인증하려면 개인 액세스 토큰 (PAT) 이 필요합니다. CodeCatalyst 이미 가지고 있다면 사용할 수 있습니다. 없다면 여기에서 새로 만들 수 있습니다.

   a. 토큰 생성을 선택합니다.

   b. 복사를 선택하여 PAT 복사합니다.

   > **Warning**
   >
   > 대화 상자를 닫은 후에는 PAT 다시 보거나 복사할 수 없습니다.

7. 액세스 자격 증명으로 gradle 속성 파일을 업데이트하세요. Replace *username* CodeCatalyst 사용자 이름으로 바꾸고 바꾸세요.*PAT* CodeCatalyst 개인 액세스 토큰으로 다음 용도로는 어떤 값이든 사용할 수 있습니다.*spaceUsername* 그리고 *spacePassword* 다음 단계에서 동일한 값을 사용하기만 하면 됩니다.

```
spaceUsername=username
spacePassword=PAT
```

8. 프로젝트 `build.gradle` 파일 `plugins` 섹션에 `maven-publish` 플러그인을 추가합니다.

```
plugins {
    id 'java-library'
    id 'maven-publish'
}
```

9. 다음으로 프로젝트 `build.gradle` 파일에 `publishing` 섹션을 추가합니다. 다음 값을 바꿉니다.

   > **Note**
   >
   > 콘솔 지침에서 복사하는 경우 다음 값을 자동으로 업데이트해야 하며 변경해서는 안 됩니다.

   - Replace *space_name* CodeCatalyst스페이스 이름과 함께.
   - Replace *proj_name* CodeCatalyst프로젝트 이름과 함께
   - Replace *repo_name* CodeCatalyst패키지 리포지토리 이름과 함께

```
publishing {
    publications {
        mavenJava(MavenPublication) {
            groupId = 'group-id'
            artifactId = 'artifact-id'
            version = 'version'
            from components.java
        }
    }
    repositories {
        maven {
            name = 'spaceName'
            url = uri('https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/')
            credentials(PasswordCredentials)
        }
    }
}
```

maven-publish플러그인은 publishing 섹션에 version 지정된groupId,artifactId, 를 기반으로 POM 파일을 생성합니다.

10. build.gradle에 대한 이러한 변경이 완료되면 다음 명령을 실행하여 프로젝트를 빌드하여 리포지토리에 업로드합니다.

```
./gradlew publish
```

11. CodeCatalyst 콘솔에서 패키지 저장소로 이동하여 패키지가 성공적으로 게시되었는지 확인합니다. 패키지 저장소의 패키지 목록에서 해당 패키지를 확인할 수 있습니다.

자세한 내용은 Gradle 웹 사이트에서 이 주제를 참조하세요.

- Java 라이브러리 빌드
- 프로젝트를 모듈로 게시

IntelliJ에서 Gradle 빌드 실행하기 IDEA

IDEAIntelliJ에서 종속 항목을 가져오는 Gradle 빌드를 실행할 수 있습니다. CodeCatalyst Gradle 을 인증하려면 개인용 액세스 CodeCatalyst 토큰 () 을 사용해야 합니다. PAT 원하는 파일 `gradle.properties` 또는 별도의 CodeCatalyst PAT 파일에 저장할 수 있습니다.

방법 1: PAT 넣기 **`gradle.properties`**

파일을 사용하고 있지 않고 `gradle.properties` 파일 내용을 사용자 PAT 이름으로 덮어쓸 수 있는 경우 이 방법을 사용하십시오. 를 사용하는 `gradle.properties` 경우 파일 내용을 덮어쓰는 PAT 대신 이 방법을 수정하여 추가할 수 있습니다.

> **Note**
>
> 예제는 `GRADLE_USER_HOME`에 있는 `gradle.properties` 파일을 보여줍니다.

계정이 없는 PAT 경우 먼저 생성하십시오.

개인용 액세스 토큰을 만들려면 (PAT)

1. 상단 메뉴 바에서 프로필 배지를 선택한 다음 내 설정을 선택합니다.

   > **Tip**
   >
   > 프로젝트 또는 스페이스의 구성원 페이지로 이동한 다음 구성원 목록에서 이름을 선택하여 사용자 프로필을 찾을 수도 있습니다.

2. PAT이름에는 이름을 설명하는 이름을 입력합니다. PAT
3. 만료일에 기본 날짜를 그대로 두거나 달력 아이콘을 선택하여 사용자 지정 날짜를 선택합니다. 만료 날짜는 기본적으로 현재 날짜로부터 1년입니다.
4. 생성(Create)을 선택합니다.

   소스 리포지토리의 복제 리포지토리를 선택할 때도 이 토큰을 생성할 수 있습니다.

5. PAT암호를 안전한 위치에 저장합니다.

   > **Important**
   >
   > PAT비밀은 한 번만 표시됩니다. 창을 닫은 후에는 검색할 수 없습니다.

다음으로, 다음 스니펫으로 build.gradle 파일을 업데이트하세요.

```
repositories {
    maven {
        name = 'spaceName'
        url = uri('https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/')
        credentials(PasswordCredentials)
    }
}
```

방법 2: 별도의 PAT 파일에 넣기

gradle.properties 파일을 수정하지 않으려면 이 방법을 사용하세요.

파일이 없는 PAT 경우 먼저 생성하십시오.

개인용 액세스 토큰을 만들려면 (PAT)

1. 상단 메뉴 바에서 프로필 배지를 선택한 다음 내 설정을 선택합니다.

   > **Tip**
   >
   > 프로젝트 또는 스페이스의 구성원 페이지로 이동한 다음 구성원 목록에서 이름을 선택하여 사용자 프로필을 찾을 수도 있습니다.

2. PAT이름에는 이름을 설명하는 이름을 입력합니다. PAT
3. 만료일에 기본 날짜를 그대로 두거나 달력 아이콘을 선택하여 사용자 지정 날짜를 선택합니다. 만료 날짜는 기본적으로 현재 날짜로부터 1년입니다.
4. 생성(Create)을 선택합니다.

   소스 리포지토리의 복제 리포지토리를 선택할 때도 이 토큰을 생성할 수 있습니다.

5. PAT암호를 안전한 위치에 저장합니다.

   > **Important**
   >
   > PAT비밀은 한 번만 표시됩니다. 창을 닫은 후에는 검색할 수 없습니다.

별도의 PAT 파일에 저장하기

1. 다음 스니펫으로 build.gradle 파일을 업데이트하세요. Replace *space_name*, *proj_name*, 및 *repo_name* CodeCatalyst 사용자 이름, 스페이스 이름, 프로젝트 이름, 패키지 저장소 이름을 입력합니다.

```
def props = new Properties()
file("fileName").withInputStream { props.load(it) }

repositories {
        maven {
            name = 'spaceName'
            url = uri('https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/')
            credentials(PasswordCredentials)
        }
    }
}
```

2. PAT파일에 지정된 build.gradle 파일에 다음을 입력합니다.

```
echo "codecatalystArtifactsToken=PAT" > fileName
```

mvn 구성 및 사용

mvn명령을 사용하여 Maven 빌드를 실행합니다. 패키지 mvn 리포지토리를 사용하도록 구성하고 인증을 위한 개인 액세스 토큰 (PAT) 을 제공해야 합니다.

목차

에서 종속성 가져오기 CodeCatalyst

CodeCatalyst 리포지토리에서 종속성을 mvn 가져오도록 구성하려면 Maven 구성 파일을 settings.xml 편집하고 선택적으로 프로젝트의 Project Model Object () 파일을 편집해야 합니다.

POM 이 POM 파일에는 프로젝트에 대한 정보와 Maven이 프로젝트를 빌드하는 데 필요한 구성 정보 (예: 종속성, 빌드 디렉터리, 소스 디렉터리, 테스트 소스 디렉터리, 플러그인, 목표 등) 가 들어 있습니다.

패키지 저장소에서 종속성을 가져오는 **mvn** 데 사용합니다. CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트 개요 페이지에서 패키지를 선택합니다.
3. 패키지 리포지토리 목록에서 패키지 리포지토리를 선택합니다.
4. 리포지토리에 연결을 선택합니다.
5. 리포지토리에 연결 대화 상자의 패키지 관리자 클라이언트 목록에서 mvn을 선택합니다.
6. mvn인증하려면 개인용 액세스 토큰 (PAT) 이 필요합니다. CodeCatalyst 이미 가지고 있다면 사용할 수 있습니다. 없다면 여기에서 새로 만들 수 있습니다.

   a. 토큰 생성을 선택합니다.

   b. 복사를 선택하여 PAT 복사합니다.

   > **Warning**
   >
   > 대화 상자를 닫은 후에는 PAT 다시 보거나 복사할 수 없습니다.

7. 리포지토리가 포함된 프로필을 `settings.xml` 파일에 추가합니다. 다음 값을 바꾸십시오.

   > **Note**
   >
   > 콘솔 지침에서 복사하는 경우 다음 값을 자동으로 업데이트해야 하며 변경해서는 안 됩니다.

   - Replace *space_name* CodeCatalyst스페이스 이름과 함께.
   - Replace *proj_name* CodeCatalyst프로젝트 이름과 함께
   - Replace *repo_name* CodeCatalyst패키지 리포지토리 이름과 함께

```
<profiles>
  <profile>
    <id>repo_name</id>
```

```
    <activation>
        <activeByDefault>true</activeByDefault>
    </activation>
    <repositories>
        <repository>
          <id>repo_name</id>
          <url>https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/</url>
        </repository>
    </repositories>
  </profile>
</profiles>
```

8. `settings.xml`파일의 서버 목록에 서버를 추가합니다. 다음 값을 바꾸십시오.

> **Note**
>
> 콘솔 지침에서 복사하는 경우 다음 값을 자동으로 업데이트해야 하며 변경해서는 안 됩니다.

- Replace *repo_name* CodeCatalyst패키지 리포지토리 이름과 함께.
- Replace *username* CodeCatalyst 사용자 이름으로
- Replace *PAT* 당신과 함께 CodeCatalystPAT.

```
<servers>
  <server>
    <id>repo_name</id>
    <username>username</username>
    <password>PAT</password>
  </server>
</servers>
```

9. (선택 사항) 모든 연결을 캡처하고 게이트웨이 리포지토리 대신 리포지토리로 라우팅하는 미러를 `settings.xml` 파일에 설정하십시오.

> **Note**
>
> 콘솔 지침에서 복사하는 경우 다음 값을 자동으로 업데이트해야 하며 변경해서는 안 됩니다.

- Replace *space_name* CodeCatalyst스페이스 이름과 함께.
- Replace *proj_name* CodeCatalyst프로젝트 이름과 함께
- Replace *repo_name* CodeCatalyst패키지 리포지토리 이름과 함께

```
<mirrors>
  <mirror>
    <id>repo_name</id>
    <name>repo_name</name>
    <url>https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/</url>
    <mirrorOf>*</mirrorOf>
  </mirror>
</mirrors>
```

> **Important**
>
> <id> 요소에서 어떤 값이든 사용할 수 있지만 그 값은 <server> 및 <repository> 요소에서 모두 동일해야 합니다. 이렇게 하면 요청에 지정된 자격 증명을 포함할 수 CodeCatalyst 있습니다.

이러한 구성을 변경한 후 프로젝트를 빌드할 수 있습니다.

```
mvn compile
```

를 통해 외부 패키지 리포지토리에서 패키지를 가져오는 중 CodeCatalyst

게이트웨이 리포지토리를 나타내는 게이트웨이에 대한 업스트림 연결로 구성하여 리포지토리를 통해 공용 CodeCatalyst 리포지토리에서 Maven 패키지를 설치할 수 있습니다. 게이트웨이 리포지토리에서 설치한 패키지는 리포지토리에 인제스트되어 저장됩니다. CodeCatalyst

현재, 다음과 같은 공개 Maven 패키지 리포지토리를 CodeCatalyst 지원합니다.

- maven-central-gateway
- google-android-gateway
- gradle-plugins-gateway
- 커먼웨어 게이트웨이

공개 Maven 패키지 리포지토리에서 패키지를 설치하려면

1. 아직 구성하지 않았다면 의 단계에 따라 CodeCatalyst 패키지 mvn 저장소로 구성하세요. 에서 종속성 가져오기 CodeCatalyst
2. 업스트림 연결로 설치하려는 게이트웨이 리포지토리가 저장소에 추가되었는지 확인하십시오. 어떤 업스트림 소스가 추가되었는지 확인하거나 게이트웨이 리포지토리를 업스트림 소스로 추가하려면 의 지침을 따르십시오. 업스트림 리포지토리 추가

업스트림 리포지토리에서 패키지를 요청하는 방법에 대한 자세한 내용은 을 참조하십시오. 업스트림 리포지토리가 포함된 패키지 버전 요청

패키지 게시 위치 CodeCatalyst

Maven 패키지를 CodeCatalyst 저장소에 게시하려면 POM 프로젝트도 `~/.m2/settings.xml` 편집해야 합니다. `mvn`

패키지를 패키지 **`mvn`** 저장소에 게시하는 데 CodeCatalyst 사용합니다.

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트 개요 페이지에서 패키지를 선택합니다.
3. 패키지 리포지토리 목록에서 패키지 리포지토리를 선택합니다.
4. 리포지토리에 연결을 선택합니다.
5. 리포지토리에 연결 대화 상자의 패키지 관리자 클라이언트 목록에서 mvn을 선택합니다.
6. mvn인증하려면 개인용 액세스 토큰 (PAT) 이 필요합니다. CodeCatalyst 이미 가지고 있다면 사용할 수 있습니다. 없다면 여기에서 새로 만들 수 있습니다.
   a. 토큰 생성을 선택합니다.
   b. 복사를 선택하여 PAT 복사합니다.

> **Warning**
>
> 대화 상자를 닫은 후에는 PAT 다시 보거나 복사할 수 없습니다.

7. 를 사용하여 로컬 컴퓨터의 환경 변수를 구성하십시오PAT. `setting.xml`파일에서 이 환경 변수를 사용하게 됩니다.

```
export CODECATALYST_ARTIFACTS_TOKEN=your_PAT
```

8. Maven이 HTTP 요청에서 토큰을 전달하도록 `CodeCatalyst_ARTIFACTS_TOKEN` 환경 변수에 대한 참조가 `settings.xml` 포함된 `<servers>` 섹션을 추가하십시오.

```
<settings>
...
    <servers>
        <server>
            <id>repo-name</id>
            <username>username</username>
            <password>${env.CodeCatalyst_ARTIFACTS_TOKEN}</password>
        </server>
    </servers>
...
</settings>
```

9. 프로젝트의 `pom.xml`에 `<distributionManagement>` 섹션을 추가합니다.

```
<project>
...
     <distributionManagement>
         <repository>
             <id>repo_name</id>
             <name>repo_name</name>
             <url>https://packages.region.codecatalyst.aws/
maven/space_name/proj_name/repo_name/</url>
         </repository>
     </distributionManagement>
...
</project>
```

이러한 구성을 변경한 후 프로젝트를 빌드하여 지정된 리포지토리에 게시할 수 있습니다.

```
mvn deploy
```

CodeCatalyst 콘솔에서 패키지 저장소로 이동하여 패키지가 성공적으로 게시되었는지 확인할 수 있습니다.

타사 패키지 게시

를 사용하여 타사 Maven 패키지를 CodeCatalyst 저장소에 게시할 수 있습니다. `mvn deploy:deploy-file` 이 방법은 패키지를 게시하고 파일만 갖고 싶어하고 패키지 소스 코드나 JAR POM 파일에는 액세스할 수 없는 사용자에게 유용할 수 있습니다.

이 `mvn deploy:deploy-file` 명령은 명령줄에 전달된 정보를 기반으로 POM 파일을 생성합니다.

파일이 없는 PAT 경우 먼저 생성하십시오.

개인용 액세스 토큰을 만들려면 (PAT)

1. 상단 메뉴 바에서 프로필 배지를 선택한 다음 내 설정을 선택합니다.

   > **Tip**
   >
   > 프로젝트 또는 스페이스의 구성원 페이지로 이동한 다음 구성원 목록에서 이름을 선택하여 사용자 프로필을 찾을 수도 있습니다.

2. PAT이름에는 이름을 설명하는 이름을 입력합니다. PAT
3. 만료일에 기본 날짜를 그대로 두거나 달력 아이콘을 선택하여 사용자 지정 날짜를 선택합니다. 만료 날짜는 기본적으로 현재 날짜로부터 1년입니다.
4. 생성(Create)을 선택합니다.

   소스 리포지토리의 복제 리포지토리를 선택할 때도 이 토큰을 생성할 수 있습니다.

5. PAT암호를 안전한 위치에 저장합니다.

   > **Important**
   >
   > PAT비밀은 한 번만 표시됩니다. 창을 닫은 후에는 검색할 수 없습니다.

타사 Maven 패키지를 게시하려면

1. 다음 콘텐츠가 포함된 `~/.m2/settings.xml` 파일을 생성합니다.

```
<settings>
    <servers>
        <server>
            <id>repo_name</id>
            <username>username</username>
            <password>PAT}</password>
        </server>
    </servers>
</settings>
```

2. `mvn deploy:deploy-file` 명령 실행:

```
mvn deploy:deploy-file -DgroupId=commons-cli          \
-DartifactId=commons-cli       \
-Dversion=1.4                  \
-Dfile=./commons-cli-1.4.jar   \
-Dpackaging=jar                \
-DrepositoryId=repo-name      \
-Durl=https://packages.region.codecatalyst.aws/maven/space-name/proj-name/repo-
name/
```

> **Note**
>
> 위 예제는 게시합니다. `commons-cli 1.4` , groupId ArtifactID, 버전 및 파일 인수를 수정하여 다른 것을 게시하십시오. JAR

이 지침은 Apache Maven 설명서에서 원격 리포지토리에 타사를 배포하는 방법에 대한 가이드의 예제를 기반으로 합니다. JARs

자세한 내용은 Apache Maven 프로젝트 웹 사이트에서 다음 항목을 참조하십시오.

- 여러 리포지토리 설정
- 설정 참조

- 배포 관리
- 프로파일

curl을 사용한 패키지 퍼블리싱

이 섹션에서는 HTTP 클라이언트를 사용하여 Maven 패키지를 CodeCatalyst 패키지 저장소에 curl 게시하는 방법을 보여줍니다. 현재 환경에 Maven 클라이언트가 없거나 설치하려는 경우 패키지를 게시하면 유용할 수 있습니다. curl

를 사용하여 Maven 패키지를 게시하려면 **curl**

1. curl인증에 사용할 환경 변수에 개인 액세스 토큰 (PAT) 을 저장해야 합니다. CodeCatalyst 이미 가지고 있다면 사용할 수 있습니다. 그렇지 않은 경우 새로 만들고 환경 변수를 구성할 수 있습니다.

   a. 의 단계에 PAT 따라 a를 생성하십시오개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여. 를 PAT 복사하여 환경 변수에 저장합니다.

   b. 로컬 컴퓨터의 명령줄에서 를 사용하여 환경 변수를 구성합니다PAT.

   ```
   export CodeCatalyst_ARTIFACTS_TOKEN=your_PAT
   ```

2. 다음 curl 명령을 사용하여 JAR CodeCatalyst 리포지토리에 게시합니다. Replace *username*, *space_name*, *proj_name*, 및 *repo_name* CodeCatalyst 사용자 이름, 스페이스 이름, 프로젝트 이름 및 패키지 저장소 이름을 입력합니다.

   ```
   curl --request PUT https://packages.region.codecatalyst.aws/maven/space-name/proj-name/repo-name/com/mycompany/app/my-app/1.0/my-app-1.0.jar \
        --user "username:CodeCatalyst_ARTIFACTS_TOKEN" --header "Content-Type:
    application/octet-stream" \
        --data-binary @target/path/to/my-app-1.0.jar
   ```

3. 다음 curl 명령을 사용하여 를 POM CodeCatalyst 저장소에 게시합니다. Replace *username*, *space_name*, *proj_name*, 및 *repo_name* CodeCatalyst 사용자 이름, 스페이스 이름, 프로젝트 이름 및 패키지 저장소 이름을 입력합니다.

   ```
   curl --request PUT https://packages.region.codecatalyst.aws/maven/space-name/proj-name/repo-name/com/mycompany/app/my-app/1.0/my-app-1.0.pom \
        --user "username:CodeCatalyst_ARTIFACTS_TOKEN" --header "Content-Type:
    application/octet-stream" \
   ```

```
    --data-binary @target/my-app-1.0.pom
```

4. 이때 Maven 패키지는 상태가 인 상태로 CodeCatalyst 저장소에 있게 됩니다. `Unfinished` 패키지를 사용하려면 패키지가 `Published` 상태에 있어야 합니다. 패키지에 `maven-metadata.xml` 파일을 업로드하거나 `Published` 콘솔에서 상태를 `Unfinished` 변경하여 패키지를 간에 이동할 수 있습니다. CodeCatalyst

   a. 옵션 1: 다음 `curl` 명령을 사용하여 패키지에 `maven-metadata.xml` 파일을 추가합니다. Replace *`username`*, *`space_name`*, *`proj_name`*, 및 *`repo_name`* CodeCatalyst 사용자 이름, 스페이스 이름, 프로젝트 이름, 패키지 저장소 이름을 입력합니다.

   ```
   curl --request PUT https://packages.region.codecatalyst.aws/maven/space-
   name/proj-name/repo-name/com/mycompany/app/my-app/maven-metadata.xml \
       --user "username:CodeCatalyst_ARTIFACTS_TOKEN" --header "Content-Type:
    application/octet-stream" \
       --data-binary @target/maven-metadata.xml
   ```

   다음은 `maven-metadata.xml` 파일 내용의 예입니다.

   ```
   <metadata modelVersion="1.1.0">
       <groupId>com.mycompany.app</groupId>
       <artifactId>my-app</artifactId>
       <versioning>
           <latest>1.0</latest>
           <release>1.0</release>
           <versions>
               <version>1.0</version>
           </versions>
           <lastUpdated>20200731090423</lastUpdated>
       </versioning>
   </metadata>
   ```

   b. 옵션 2: `Published` CodeCatalyst 콘솔에서 패키지 상태를 로 업데이트합니다. 패키지 버전 상태를 업데이트하는 방법에 대한 자세한 내용은 을 참조하십시오패키지 버전 상태 업데이트.

패키지 JAR 파일만 있는 경우 를 사용하여 `mvn` 소모품 패키지 버전을 CodeCatalyst 저장소에 게시할 수 있습니다. 이는 패키지의 소스 코드 또는 POM 에 액세스할 수 없는 경우에 유용할 수 있습니다. 세부 정보는 타사 패키지 게시를 참조하세요.

Maven 체크섬 및 스냅샷 사용

다음 섹션에서는 Maven 체크섬과 Maven 스냅샷을 사용하는 방법을 설명합니다. CodeCatalyst

Maven 체크섬 사용

Maven 패키지가 CodeCatalyst 패키지 저장소에 게시되면 패키지의 각 자산 또는 파일과 관련된 체크섬을 사용하여 업로드를 검증합니다. 자산의 예로는 jar, pom, war 파일 등이 있습니다. 각 에셋에 대해 Maven 패키지에는 자산 이름에 또는 같은 추가 확장자를 붙인 여러 개의 체크섬 파일이 포함되어 있습니다. `md5` `sha1` 예를 들어, `my-maven-package.jar`라는 이름이 지정된 파일의 체크섬 파일은 `my-maven-package.jar.md5` 및 `my-maven-package.jar.sh1`일 수 있습니다.

모든 Maven 패키지에는 파일도 포함되어 있습니다. `maven-metadata.xml` 게시가 성공하려면 이 파일을 업로드해야 합니다. 패키지 파일을 업로드하는 동안 체크섬 불일치가 감지되면 게시가 중지됩니다. 이로 인해 업로드되지 않을 수 `maven-metadata.xml` 있습니다. 이 경우 Maven 패키지의 상태는 로 `Unfinished` 설정됩니다. 이 상태의 패키지에 포함된 에셋은 다운로드할 수 없습니다.

Maven 패키지를 게시할 때 체크섬 불일치가 발생할 경우 다음 사항에 유의하십시오.

- 업로드하기 전에 `maven-metadata.xml` 체크섬 불일치가 발생하면 패키지 상태가 로 설정되지 않습니다. `Unfinished` 패키지가 보이지 않고 해당 자산을 사용할 수 없습니다. 이 경우 다음 중 하나를 시도한 다음 에셋을 다시 다운로드해 보십시오.
  - Maven 패키지를 게시하는 명령을 다시 실행합니다. 다운로드 중에 네트워크 문제로 인해 체크섬 파일이 손상된 경우 이 방법이 작동할 수 있습니다. 재시도 시 네트워크 문제가 해결되면 체크섬이 일치하여 다운로드가 성공적으로 완료됩니다.
  - Maven 패키지를 다시 게시해도 문제가 해결되지 않으면 패키지를 삭제한 다음 다시 게시하십시오.
- 업로드 후 `maven-metadata.xml` 체크섬 불일치가 발생하면 패키지 상태가 로 설정됩니다. `Published` 체크섬 불일치가 있는 에셋을 포함하여 패키지의 모든 에셋을 사용할 수 있습니다. 자산을 다운로드하면 에서 생성된 체크섬도 함께 CodeCatalyst 다운로드됩니다. 다운로드한 파일에 체크섬 불일치가 있는 경우 다운로드한 체크섬 파일이 패키지 게시 시 업로드된 체크섬과 일치하지 않을 수 있습니다.

Maven 스냅샷 사용

Maven 스냅샷은 최신 프로덕션 브랜치 코드를 참조하는 Maven 패키지의 특수 버전입니다. 이 스냅샷은 최종 릴리스 버전보다 앞서는 개발 버전입니다. 패키지 버전에 추가된 접미사로 Maven 패키지의 스냅샷 버전을 `SNAPSHOT` 식별할 수 있습니다. 예를 들어, 버전 `1.1`의 스냅샷은 `1.1-SNAPSHOT`입니다.

자세한 내용은 버전이란 무엇입니까? 를 참조하십시오. SNAPSHOT 아파치 메이븐 프로젝트 웹사이트에서.

CodeCatalyst Maven 스냅샷 게시 및 사용을 지원합니다. Maven 스냅샷을 리포지토리에 게시하거나, 직접 연결된 경우 업스트림 CodeCatalyst 리포지토리에 게시할 수 있습니다. 하지만 패키지 리포지토리와 해당 업스트림 리포지토리 중 하나의 스냅샷 버전은 지원되지 않습니다. 예를 들어 버전이 포함된 Maven 패키지를 패키지 저장소에 `1.2-SNAPSHOT` 업로드하는 경우 동일한 스냅샷 버전의 Maven 패키지를 업스트림 리포지토리 중 하나에 업로드하는 것은 지원되지 CodeCatalyst 않습니다. 이 시나리오에서는 예상치 못한 결과가 반환될 수 있습니다.

Maven 스냅샷이 게시되면 이전 버전은 빌드라는 새 버전에 보존됩니다. Maven 스냅샷이 게시될 때마다 새 빌드 버전이 생성됩니다. 스냅샷의 모든 이전 버전은 빌드 버전에서 유지 관리됩니다. Maven 스냅샷이 게시되면 상태가 로 `Published` 설정되고 이전 버전이 포함된 빌드의 상태가 로 설정됩니다. `Unlisted`

스냅샷을 요청하면 상태가 있는 버전이 `Published` 반환됩니다. 이 버전은 항상 Maven 스냅샷의 최신 버전입니다. 스냅샷의 특정 빌드를 요청할 수도 있습니다.

Maven 스냅샷의 모든 빌드 버전을 삭제하려면 CodeCatalyst 콘솔을 사용하십시오.

사용 NuGet

이 주제에서는 를 사용하여 `NuGet` 패키지를 사용하고 게시하는 방법을 설명합니다 CodeCatalyst.

> **Note**
>
> CodeCatalyst NuGet버전 4.8 이상을 지원합니다.

주제

- 비주얼 CodeCatalyst 스튜디오와 함께 사용
- 너겟 또는 닷넷 구성 및 사용 CLI
- NuGet 패키지 이름, 버전 및 자산 이름 정규화
- NuGet 적합성

비주얼 CodeCatalyst 스튜디오와 함께 사용

Visual Studio에서 CodeCatalyst 직접 패키지를 사용할 수 있습니다.

dotnet또는 등의 CLI 도구를 구성하고 NuGet 함께 사용하려면 nuget 을 참조하십시오너겟 또는 닷넷 구성 및 사용 CLI.

목차

다음과 같이 Visual Studio 구성 CodeCatalyst

Windows

다음을 사용하여 Visual Studio를 구성하려면 CodeCatalyst

1. 인증하려면 개인 액세스 토큰 (PAT) 이 필요합니다. CodeCatalyst 이미 가지고 있다면 사용할 수 있습니다. 그렇지 않은 경우, 안내에 개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여 따라 새로 만드세요.
2. nuget또는 dotnet 를 사용하여 패키지 리포지토리와 자격 증명을 구성합니다.

   dotnet

   Linux 및 macOS 사용자: Windows 이외의 플랫폼에서는 암호화가 지원되지 않으므로 다음 명령에 --store-password-in-clear-text 플래그를 추가해야 합니다. 이렇게 하면 구성 파일에 암호가 일반 텍스트로 저장된다는 점에 유의하세요.

   ```
   dotnet nuget add source https://packages.region.codecatalyst.aws/nuget/space-name/proj-name/repo-name/v3/index.json --name repo_name --password PAT --username user_name
   ```

   nuget

   ```
   nuget sources add -name repo_name -Source https://
   packages.region.codecatalyst.aws/nuget/space-name/proj-name/repo-name/v3/
   index.json -password PAT --username user_name
   ```

   출력 예제:

```
Package source with Name: repo_name added successfully.
```

3. 새 패키지 소스를 사용하도록 Visual Studio를 구성하십시오. Visual Studio에서 도구를 선택한 다음 옵션을 선택합니다.
4. 옵션 메뉴에서 NuGet Package Manager 섹션을 펼치고 패키지 소스를 선택합니다.
5. 사용 가능한 패키지 소스 목록에서 다음을 확인하십시오.*repo_name* 소스가 활성화되었습니다. NuGet 갤러리에 대한 업스트림 연결을 사용하여 패키지 저장소를 구성한 경우 nuget.org 소스를 비활성화하십시오.

macOS

다음과 같이 Visual Studio를 구성하려면 CodeCatalyst

1. 인증하려면 개인 액세스 토큰 (PAT) 이 필요합니다. CodeCatalyst 이미 가지고 있다면 사용할 수 있습니다. 그렇지 않은 경우, 안내에 개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여 따라 새로 만드세요.
2. 메뉴 막대에서 환경설정을 선택합니다.
3. NuGet섹션에서 소스를 선택합니다.
4. 추가를 선택하고 리포지토리 정보를 추가합니다.

    a. 이름에 CodeCatalyst 패키지 저장소 이름을 입력합니다.

    b. 위치에는 CodeCatalyst 패키지 리포지토리 엔드포인트를 입력합니다. 다음 스니펫은 예제 엔드포인트를 보여줍니다. Replace *space-name*, *proj-name*, 및 *repo-name* CodeCatalyst 스페이스 이름, 프로젝트 이름, 리포지토리 이름을 입력합니다.

    ```
    https://packages.region.codecatalyst.aws/nuget/space-name/proj-name/repo-name/
    ```

    c. 사용자 이름에는 유효한 값을 모두 입력합니다.

    d. 비밀번호에는 다음을 입력합니다PAT.
5. Add source(소스 추가)를 선택합니다.
6. NuGet갤러리에 대한 업스트림 연결을 사용하여 패키지 저장소를 구성한 경우 nuget.org 소스를 비활성화하십시오.

구성 후 Visual Studio는 CodeCatalyst 리포지토리, 모든 업스트림 리포지토리 NuGet또는.org (업스트림 소스로 구성한 경우) 의 패키지를 사용할 수 있습니다. Visual Studio에서 NuGet 패키지를 찾아 설치

하는 방법에 대한 자세한 내용은 NuGet 설명서의 NuGet 패키지 관리자를 사용하여 Visual Studio에서 패키지 설치 및 관리를 참조하십시오.

너겟 또는 닷넷 구성 및 사용 CLI

NuGet및 와 dotnet 같은 CLI 도구를 사용하여 패키지를 게시하고 사용할 수 CodeCatalyst 있습니다. 이 문서에서는 CLI 도구를 구성하고 도구를 사용하여 패키지를 게시하거나 사용하는 방법에 대한 정보를 제공합니다.

목차

다음을 사용하여 NuGet 구성하십시오. CodeCatalyst

를 구성하려면 NuGet 구성 NuGet 파일에 리포지토리 엔드포인트와 개인용 액세스 토큰을 추가하여 패키지 리포지토리에 대한 연결을 nuget 허용하거나 dotnet 허용하도록 하십시오. CodeCatalyst CodeCatalyst

CodeCatalyst 패키지 NuGet 리포지토리로 구성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트 개요 페이지에서 패키지를 선택합니다.
3. 패키지 리포지토리 목록에서 패키지 리포지토리를 선택합니다.
4. 리포지토리에 연결을 선택합니다.
5. 리포지토리에 연결 대화 상자의 패키지 관리자 클라이언트 목록에서 dotnet을 선택합니다 NuGet.
6. NuGet 인증하려면 개인용 액세스 토큰 (PAT) 이 필요합니다. CodeCatalyst 이미 가지고 있다면 사용할 수 있습니다. 없다면 여기에서 새로 만들 수 있습니다.
    a. 토큰 생성을 선택합니다.
    b. 복사를 선택하여 PAT 복사합니다.

> **Warning**
>
> 대화 상자를 닫은 후에는 PAT 다시 보거나 복사할 수 없습니다.

7. 리포지토리의 NuGet 엔드포인트를 `nuget` 구성하거나 `dotnet` 사용하도록 설정합니다 CodeCatalyst PAT. 다음 값을 바꾸십시오.

> **Note**
>
> 콘솔 지침에서 복사하는 경우 다음 값을 자동으로 업데이트해야 하며 변경해서는 안 됩니다.

- Replace *`username`* CodeCatalyst 사용자 이름과 함께
- Replace *`PAT`* 당신과 CodeCatalyst PAT 함께
- Replace *`space_name`* CodeCatalyst 스페이스 이름과 함께
- Replace *`proj_name`* CodeCatalyst 프로젝트 이름과 함께
- Replace *`repo_name`* CodeCatalyst 패키지 리포지토리 이름과 함께

a. `nuget`에서 `nuget sources add` 명령을 사용합니다.

```
nuget sources add -name "repo_name" -Source "https://
packages.region.codecatalyst.aws/nuget/space_name/proj_name/repo_name/v3/
index.json" -username "username" -password "PAT"
```

b. `dotnet`에서 `dotnet nuget add source` 명령을 사용합니다.

Linux 및 macOS 사용자: Windows 이외의 플랫폼에서는 암호화가 지원되지 않으므로 다음 명령에 `--store-password-in-clear-text` 플래그를 추가해야 합니다. 이렇게 하면 구성 파일에 암호가 일반 텍스트로 저장된다는 점에 유의하세요.

```
dotnet nuget add source "https://packages.region.codecatalyst.aws/
nuget/space_name/proj_name/repo_name/v3/index.json" -n "proj_name/repo_name" -u
 "username" -p "PAT" --store-password-in-clear-text
```

를 NuGet 구성한 후에는 CodeCatalyst 리포지토리 또는 해당 업스트림 리포지토리 중 하나에 저장된 NuGet 패키지를 사용하고 패키지를 리포지토리에 게시할 NuGet 수 있습니다. CodeCatalyst CodeCatalyst

NuGet 리포지토리의 패키지 소비 CodeCatalyst

NuGet 로 CodeCatalyst 구성한 후에는 CodeCatalyst 리포지토리 또는 해당 업스트림 리포지토리 중 하나에 저장된 NuGet 패키지를 사용할 수 있습니다.

nuget 또는 dotnet이 포함된 CodeCatalyst 리포지토리 또는 해당 업스트림 리포지토리 중 하나에서 패키지 버전을 사용하려면 다음 명령을 실행합니다. Replace *packageName* 사용하려는 패키지 이름과 함께 *packageSourceName* NuGet 구성 파일에 CodeCatalyst 패키지 저장소의 소스 이름을 포함하십시오. 이 이름은 저장소 이름이어야 합니다.

를 사용하여 패키지를 설치하려면 **dotnet**

```
dotnet add packageName --source packageSourceName
```

를 사용하여 패키지를 설치하려면 **nuget**

```
nuget install packageName --source packageSourceName
```

자세한 내용은 Microsoft 설명서의 nuget을 사용한 패키지 관리 CLI 또는 CLIdotnet을 사용하여 패키지 설치 및 관리를 참조하십시오.

.org부터 NuGet 패키지까지의 패키지 소비 NuGet CodeCatalyst

NuGet.org에 대한 업스트림 연결을 사용하여 리포지토리를 구성하면 CodeCatalyst 리포지토리를 통해.org의 NuGet 패키지를 사용할 수 있습니다. NuGet NuGet.org에서 사용한 패키지는 저장소에 인제스트되어 저장됩니다. CodeCatalyst

.org에서 패키지를 사용하려면 NuGet

1. 아직 구성하지 않았다면 의 단계에 따라 NuGet 패키지 저장소를 사용하여 CodeCatalyst 패키지 관리자를 구성하세요. 다음을 사용하여 NuGet 구성하십시오. CodeCatalyst
2. NuGet리포지토리가.org를 업스트림 연결로 추가했는지 확인하세요. 의 지침에 따라 스토어 리포지토리를 선택하여 어떤 업스트림 소스가 추가되었는지 확인하거나 NuGet.org를 업스트림 소스로 추가할 수 있습니다. 업스트림 리포지토리 추가 NuGet

패키지 게시 NuGet CodeCatalyst

NuGet 로 CodeCatalyst 구성한 후에는 nuget 또는 를 사용하여 패키지 버전을 리포지토리에 dotnet 게시할 수 있습니다. CodeCatalyst

패키지 버전을 CodeCatalyst 리포지토리로 푸시하려면 NuGet 구성 .nupkg 파일에 파일의 전체 경로와 CodeCatalyst 리포지토리의 소스 이름을 포함하여 다음 명령을 실행합니다.

를 사용하여 패키지를 게시하려면 **dotnet**

```
dotnet nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg --source packageSourceName
```

를 사용하여 패키지를 게시하려면 **nuget**

```
nuget push path/to/nupkg/SamplePackage.1.0.0.nupkg --source packageSourceName
```

NuGet 패키지 이름, 버전 및 자산 이름 정규화

CodeCatalyst 패키지 및 자산 이름과 패키지 버전을 저장하기 전에 정규화합니다. 즉, 패키지 또는 자산이 게시될 때 제공된 이름 또는 버전과 다를 CodeCatalyst 수 있습니다.

패키지 이름 정규화: 모든 문자를 소문자로 변환하여 NuGet 패키지 이름을 CodeCatalyst 정규화합니다.

패키지 버전 정규화: 와 동일한 패턴을 사용하여 NuGet 패키지 버전을 CodeCatalyst 정규화합니다. NuGet 다음 정보는 설명서의 정규화된 버전 번호에서 가져온 것입니다. NuGet

- 버전 번호에서 앞에 오는 0은 제거되었습니다.
  - 1.00은 1.0으로 간주합니다.
  - 1.01.1은 1.1.1으로 간주합니다.
  - 1.00.0.1은 1.0.0.1으로 간주합니다.
- 버전 번호의 네 번째 부분에 있는 0은 생략합니다.
  - 1.0.0.0은 1.0.0으로 간주합니다.
  - 1.0.01.0은 1.0.1으로 간주합니다.
- SemVer 2.0.0 빌드 메타데이터가 제거되었습니다.
  - 1.0.7+r3456은 1.0.7으로 간주합니다.

패키지 자산 이름 정규화: 정규화된 NuGet 패키지 이름 및 패키지 버전에서 패키지 자산 이름을 CodeCatalyst 구성합니다.

NuGet 적합성

이 안내서에는 CodeCatalyst 의 다양한 NuGet 도구 및 버전과의 호환성에 대한 정보가 포함되어 있습니다.

주제

- 일반 NuGet 호환성
- NuGet 명령줄 지원

일반 NuGet 호환성

CodeCatalyst NuGet 4.8 이상을 지원합니다.

CodeCatalyst 프로토콜의 NuGet HTTP V3만 지원합니다. 즉, 프로토콜의 V2를 사용하는 일부 CLI 명령은 지원되지 않습니다. 자세한 내용은 다음 너겟 명령 지원 섹션을 참조하십시오.

CodeCatalyst PowerShellGet 2.x는 지원하지 않습니다.

NuGet 명령줄 지원

CodeCatalyst NuGet (`nuget`) 및 을 지원합니다. NET코어 (`dotnet`) CLI 도구.

너겟 명령 지원

NuGet의 HTTP 프로토콜 CodeCatalyst V3만 지원하므로 다음 명령을 리소스에 CodeCatalyst 사용할 때는 작동하지 않습니다.

- `list`: 이 `nuget list` 명령은 지정된 소스의 패키지 목록을 표시합니다. 패키지 리포지토리의 CodeCatalyst 패키지 목록을 가져오려면 CodeCatalyst 콘솔에서 해당 리포지토리로 이동하십시오.

Python 사용

이 항목에서는 Python 패키지 관리자 및 `twine` Python 패키지 게시 유틸리티를 사용하는 `pip` 방법에 대해 설명합니다 CodeCatalyst.

주제

pip 설정과 Python 패키지 설치

pip와 함께 CodeCatalyst 사용하려면 패키지 저장소에 연결하고 pip 인증을 위한 개인 액세스 토큰을 제공해야 합니다. CodeCatalyst 콘솔에서 패키지 리포지토리에 pip 연결하는 방법을 확인할 수 있습니다. 인증하고 pip CodeCatalyst 연결한 후 pip 명령을 실행할 수 있습니다.

목차

pip를 CodeCatalyst 사용하여 Python 패키지 설치하기

다음 지침은 CodeCatalyst 패키지 리포지토리 또는 업스트림 리포지토리 중 하나에서 Python 패키지를 pip 설치하도록 구성하는 방법을 설명합니다.

패키지 리포지토리에서 Python 패키지를 구성하고 **pip** 설치하기 위해 사용하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트 개요 페이지에서 패키지를 선택합니다.
3. 패키지 리포지토리 목록에서 패키지 리포지토리를 선택합니다.
4. 리포지토리에 연결을 선택합니다.
5. 리포지토리에 연결 대화 상자의 패키지 관리자 클라이언트 목록에서 pip를 선택합니다.
6. pip를 인증하려면 개인 액세스 토큰 (PAT) 이 필요합니다. CodeCatalyst 이미 가지고 있다면 사용할 수 있습니다. 없다면 여기에서 새로 만들 수 있습니다.

   a. 토큰 생성을 선택합니다.

b. 복사를 선택하여 PAT 복사합니다.

> **Warning**
>
> 대화 상자를 닫은 후에는 PAT 다시 보거나 복사할 수 없습니다.

7. `pip config`명령을 사용하여 CodeCatalyst 레지스트리 URL 및 자격 증명을 설정합니다. 다음 값을 바꿉니다.

> **Note**
>
> 콘솔 지침에서 복사하는 경우 다음 값을 자동으로 업데이트해야 하며 변경해서는 안 됩니다.

- Replace *username* CodeCatalyst 사용자 이름과 함께
- Replace *PAT* 당신과 함께 CodeCatalyst PAT.
- Replace *space_name* CodeCatalyst 스페이스 이름과 함께
- Replace *proj_name* CodeCatalyst 프로젝트 이름과 함께
- Replace *repo_name* CodeCatalyst 패키지 리포지토리 이름과 함께

```
pip config set global.index-url https://username:PAT@https://
packages.region.codecatalyst.aws/pypi/space_name/proj_name/repo_name/simple/
```

8. 패키지가 리포지토리 또는 업스트림 리포지토리 중 하나에 있다고 할 경우, `pip install`을 사용하여 패키지를 설치할 수 있습니다. 예를 들어 다음 명령을 사용하여 `requests` 패키지를 설치할 수 있습니다.

```
pip install requests
```

이 `-i` 옵션을 사용하여 패키지 저장소 대신 https://pypi.org에서 패키지를 설치하는 것으로 일시적으로 되돌릴 수 있습니다 CodeCatalyst .

```
pip install -i https://pypi.org/simple requests
```

PyPI에서 PyPi까지 파이썬 패키지 사용하기 CodeCatalyst

PyPI에 대한 업스트림 연결을 사용하여 리포지토리를 구성하여 CodeCatalyst 리포지토리를 통해 Python 패키지 인덱스 (PyPi) 의 Python 패키지를 사용할 수 있습니다. PyPI에서 사용한 패키지는 저장소에 인제스트되어 저장됩니다. CodeCatalyst

PyPI에서 패키지를 사용하려면

1. 아직 구성하지 않았다면 의 단계에 따라 CodeCatalyst 패키지 저장소로 pip를 구성하세요. pip를 CodeCatalyst 사용하여 Python 패키지 설치하기
2. 리포지토리가 PyPI를 업스트림 소스로 추가했는지 확인하세요. 의 지침에 업스트림 리포지토리 추가 따라 PyPI 스토어 리포지토리를 선택하여 어떤 업스트림 소스가 추가되었는지 확인하거나 PyPI를 업스트림 소스로 추가할 수 있습니다.

업스트림 리포지토리에서 패키지를 요청하는 방법에 대한 자세한 내용은 을 참조하십시오. 업스트림 리포지토리가 포함된 패키지 버전 요청

pip 명령 지원

다음 섹션에는 지원되지 않는 특정 명령 외에도 CodeCatalyst 리포지토리에서 지원되는 pip 명령이 요약되어 있습니다.

주제

- 리포지토리와 상호 작용하는 지원되는 명령
- 지원되는 클라이언트 측 명령

리포지토리와 상호 작용하는 지원되는 명령

이 섹션에는 `pip` 클라이언트가 구성된 레지스트리에 하나 이상의 요청을 보내는 `pip` 명령이 나열되어 있습니다. 이러한 명령은 패키지 저장소에 대해 호출했을 때 제대로 작동하는 것으로 확인되었습니다. CodeCatalyst

Command	설명
install	패키지를 설치합니다.
download	패키지를 다운로드합니다.

CodeCatalyst `pip search`구현되지 않습니다. CodeCatalyst 패키지 리포지토리를 `pip` 구성한 경우 `pip search` 실행하면 PyPI에서 패키지를 검색하고 표시합니다.

지원되는 클라이언트 측 명령

이러한 명령은 리포지토리와 직접 상호 작용할 필요가 CodeCatalyst 없으므로 명령을 지원하기 위해 아무것도 할 필요가 없습니다.

Command	설명
uninstall	패키지를 제거합니다.
freeze	설치된 패키지를 요구 사항 형식으로 출력합니다.
list	설치된 패키지를 나열합니다.
show	설치된 패키지에 대한 정보를 표시합니다.
check	설치된 패키지에 호환되는 종속성이 있는지 확인하세요.
config	로컬 및 글로벌 구성을 관리합니다.
wheel	요구 사항에 맞게 휠을 빌드합니다.
hash	패키지 아카이브의 해시를 계산합니다.
completion	명령 완성을 돕습니다.
debug	디버깅에 유용한 정보를 표시합니다.
도움	명령에 대한 도움말을 표시합니다.

트와인 설정 및 Python 패키지 퍼블리싱

`twine`와 함께 CodeCatalyst 사용하려면 패키지 저장소에 연결하고 `twine` 인증을 위한 개인 액세스 토큰을 제공해야 합니다. CodeCatalyst 콘솔에서 패키지 리포지토리에 `twine` 연결하는 방법을 확인할 수 있습니다. 인증하고 `twine` CodeCatalyst 연결한 후 `twine` 명령을 실행할 수 있습니다.

Twine을 CodeCatalyst 사용하여 패키지를 게시합니다.

다음 지침은 CodeCatalyst 패키지 저장소를 인증하고 연결하는 `twine` 방법을 설명합니다.

패키지를 구성하고 사용하여 패키지 저장소에 패키지를 **`twine`** 게시하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트 개요 페이지에서 패키지를 선택합니다.
3. 패키지 리포지토리 목록에서 패키지 리포지토리를 선택합니다.
4. 리포지토리에 연결을 선택합니다.
5. 리포지토리에 연결 대화 상자의 패키지 관리자 클라이언트 목록에서 Twine을 선택합니다.
6. twine을 인증하려면 개인 액세스 토큰 (PAT) 이 필요합니다. CodeCatalyst 이미 가지고 있다면 사용할 수 있습니다. 없다면 여기에서 새로 만들 수 있습니다.

   a. 토큰 생성을 선택합니다.

   b. 복사를 선택하여 PAT 복사합니다.

   > **Warning**
   >
   > 대화 상자를 닫은 후에는 PAT 다시 보거나 복사할 수 없습니다.

7. `.pypirc`파일 또는 환경 변수를 사용하여 꼬기를 구성할 수 있습니다.

   a. **`.pypirc`**파일로 구성하기.

   선택한 `~/.pypirc` 편집기에서 엽니다.

   리포지토리 CodeCatalyst, 사용자 이름, 이전 단계에서 생성하고 복사한 인덱스 서버를 추가합니다. PAT 다음 값을 바꾸십시오.

   > **Note**
   >
   > 콘솔 지침에서 복사하는 경우 다음 값을 자동으로 업데이트해야 하며 변경해서는 안 됩니다.

   - Replace *`username`* CodeCatalyst 사용자 이름과 함께.
   - Replace *`PAT`* 당신과 함께 CodeCatalyst PAT.

- Replace *space_name* CodeCatalyst 스페이스 이름과 함께
- Replace *proj_name* CodeCatalyst 프로젝트 이름과 함께
- Replace *repo_name* CodeCatalyst 패키지 리포지토리 이름과 함께

```
[distutils]
index-servers = proj-name/repo-name

[proj-name/repo-name]
repository = https://packages.region.codecatalyst.aws/
pypi/space_name/proj_name/repo_name/
password = PAT
username = username
```

b. 환경 변수를 사용하여 구성하기.

다음 환경 변수를 설정합니다. TWINE_REPOSITORY_URL값을 업데이트하십시오.*space_name*, *proj_name*, 및 *repo_name* CodeCatalyst 스페이스, 프로젝트, 패키지 리포지토리 이름을 포함하세요.

```
export TWINE_USERNAME=username
```

```
export TWINE_PASSWORD=PAT
```

```
export TWINE_REPOSITORY_URL="https://packages.region.codecatalyst.aws/
pypi/space_name/proj_name/repo_name/"
```

8. `twine upload`명령을 사용하여 Python 배포판을 게시합니다.

Python 패키지 이름 정규화

CodeCatalyst 패키지 이름을 저장하기 전에 정규화합니다. 즉, 패키지 이름이 패키지가 게시될 때 제공된 이름과 다를 CodeCatalyst 수 있습니다.

Python 패키지의 경우, 정규화를 수행할 때 패키지 이름은 소문자와 `.`, `-` 및 `_` 문자의 모든 인스턴스가 단일 `-` 문자로 대체됩니다. 따라서 `pigeon_cli` 및 `pigeon.cli` 패키지 이름은 `pigeon-cli`과 같이 정규화되어 저장됩니다. 정규화되지 않은 이름은 pip 및 twine에서 사용할 수 있습니다. Python 패키지 이름 정규화에 대한 자세한 내용은 Python PEP설명서의 503을 참조하십시오.

Python 호환성

CodeCatalyst 는 `/simple/` API 지원하지 않지만 `Legacy` API 작업은 지원합니다. CodeCatalyst PyPI `XML-RPC` 또는 `JSON` API 작업을 지원하지 않습니다.

자세한 내용은 Python 패키징 기관의 GitHub 저장소에서 다음을 참조하십시오.

- 레거시 API
- XML-RPC API
- JSON API

패키지 할당량

다음 표에는 Amazon의 패키지 할당량 및 한도가 설명되어 있습니다. CodeCatalyst CodeCatalystAmazon의 할당량에 대한 자세한 내용은 을 참조하십시오. 에 대한 할당량 CodeCatalyst

Resource	기본 할당량
패키지 리포지토리	스페이스당 최대 1000개
다이렉트 업스트림 리포지토리	패키지 리포지토리당 최대 10개
업스트림 패키지 리포지토리를 검색했습니다.	요청된 패키지 버전당 최대 25개의 업스트림 리포지토리를 검색했습니다.
패키지 에셋 파일 크기	패키지 에셋당 최대 5GB
패키지 에셋	패키지 버전당 최대 150개

워크플로를 통한 빌드, 테스트, 배포

CodeCatalyst개발 환경에서 애플리케이션 코드를 작성하고 CodeCatalyst 소스 리포지토리로 푸시했으면 배포할 준비가 된 것입니다. 이 작업을 자동으로 수행하는 방법은 워크플로를 사용하는 것입니다.

워크플로는 CI/CD (지속적 통합 및 지속적 전달) 시스템의 일부로 코드를 빌드, 테스트 및 배포하는 방법을 설명하는 자동화된 절차입니다. 워크플로는 워크플로 실행 중에 수행할 일련의 단계 또는 조치를 정의합니다. 또한 워크플로는 워크플로를 시작하게 하는 이벤트 또는 트리거를 정의합니다. 워크플로를 설정하려면 CodeCatalyst 콘솔의 시각적 또는 YAML 편집기를 사용하여 워크플로 정의 파일을 만듭니다.

> **Tip**
>
> 프로젝트에서 워크플로를 사용하는 방법을 간단히 살펴보려면 청사진을 사용하여 프로젝트를 만들어 보세요. 각 블루프린트는 검토, 실행, 실험할 수 있는 작동하는 워크플로를 배포합니다.

워크플로우 정의 파일 정보

워크플로 정의 파일은 워크플로를 설명하는 YAML 파일입니다. 기본적으로 파일은 소스 리포지토리의 루트에 있는 `~/.codecatalyst/workflows/` 폴더에 저장됩니다. 파일 확장자는.yml 또는.yaml일 수 있으며 확장자는 소문자여야 합니다.

다음은 간단한 워크플로 정의 파일의 예입니다. 다음 표에서는 이 예제의 각 줄에 대해 설명합니다.

```
Name: MyWorkflow
SchemaVersion: 1.0
RunMode: QUEUED
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
```

```
    Steps:
      - Run: docker build -t MyApp:latest .
```

행	설명
`Name: MyWorkflow`	워크플로의 이름을 지정합니다. Name속성에 대한 자세한 내용은 을 참조하십시오최상위 속성.
`SchemaVersion: 1.0`	워크플로 스키마 버전을 지정합니다. SchemaVersion 속성에 대한 자세한 내용은 을 참조하십시오최상위 속성.
`RunMode: QUEUED`	다중 실행을 CodeCatalyst 처리하는 방법을 나타냅니다. 실행 모드에 대한 자세한 내용은 을 참조하십시오실행의 대기열 동작 구성.
`Triggers:`	워크플로 실행을 시작하는 로직을 지정합니다. 트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.

`- Type: PUSH`
`Branches:`
`- main`
	코드를 기본 소스 리포지토리의 main 브랜치에 푸시할 때마다 워크플로가 시작되어야 함을 나타냅니다. 워크플로 소스에 대한 자세한 내용은 을 참조하십시오소스 리포지토리를 워크플로에 연결.

`Actions:`	워크플로우 실행 중에 수행할 작업을 정의합니다. 이 예제에서 Actions 섹션은 이라는 단일 작업을 정의합니다Build. 액션에 대한 자세한 내용은 을 참조하십시오워크플로우 작업 구성.
`Build:`	Build액션의 속성을 정의합니다. 빌드 액션에 대한 자세한 내용은 을 참조하십시오워크플로를 사용한 구축.
`Identifier: aws/build@v1`	빌드 작업의 고유한 하드 코딩된 식별자를 지정합니다.

행	설명

`Inputs:`
`  Sources:`
`  - WorkflowSource`
	빌드 작업이 `WorkflowSource` 소스 저장소에서 처리를 완료하는 데 필요한 파일을 찾아야 함을 나타냅니다. 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

`Configuration:`	빌드 작업과 관련된 구성 속성을 포함합니다.

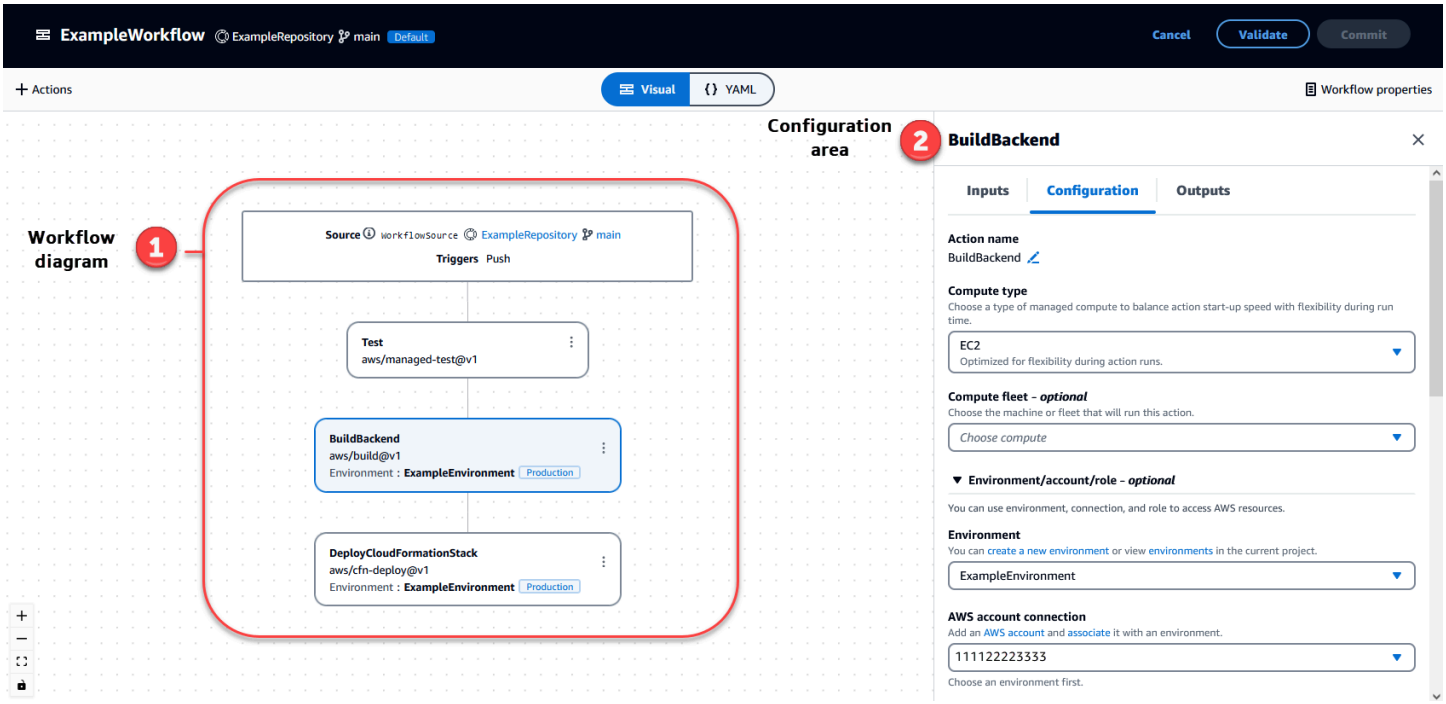
`Steps:`
`  - Run: docker build -t MyApp:lat`
`est .`
	라는 Docker 이미지를 `MyApp` 빌드하고 태그를 지정하도록 빌드 액션에 지시합니다. `latest`

워크플로 정의 파일에서 사용할 수 있는 모든 속성의 전체 목록은 를 참조하십시오. 워크플로우 YAML 정의

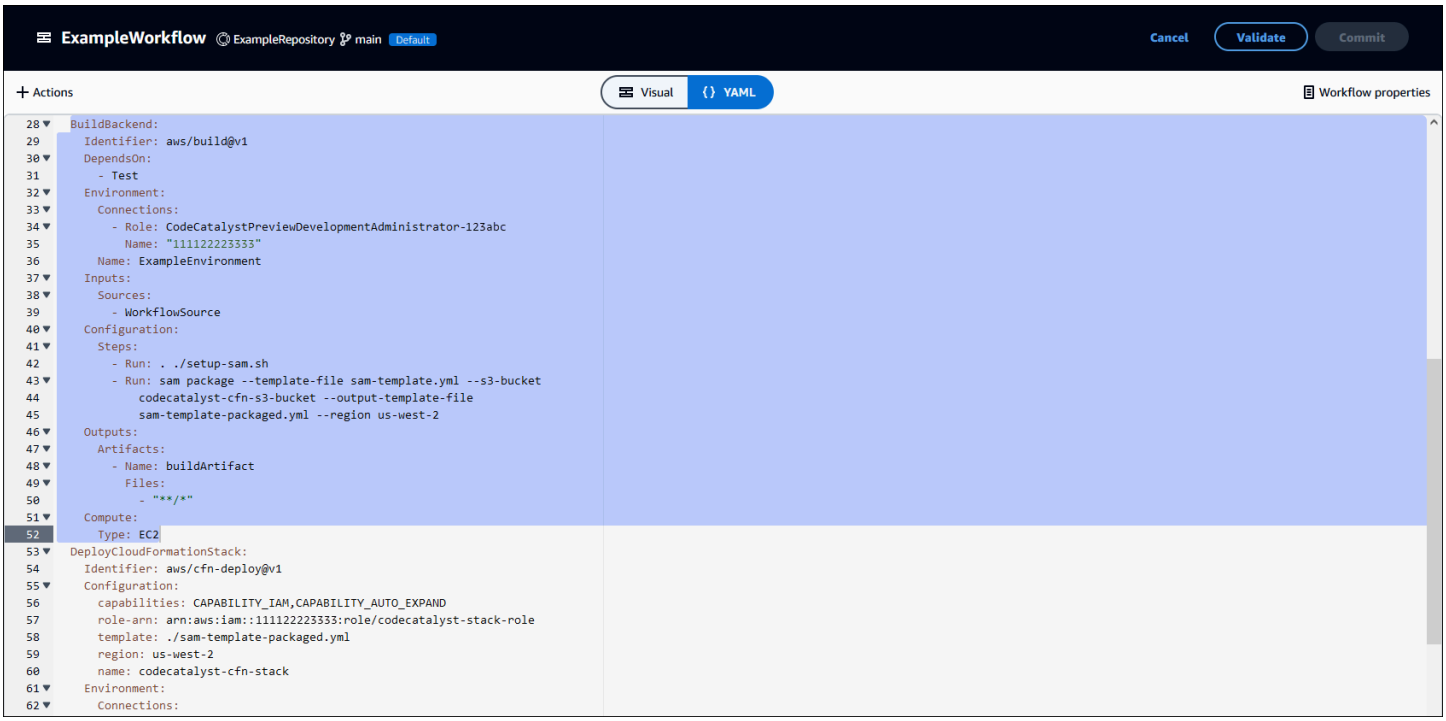
CodeCatalyst 콘솔의 비주얼 및 YAML 에디터 사용

워크플로 정의 파일을 만들고 편집하려면 원하는 편집기를 사용할 수 있지만 CodeCatalyst 콘솔의 시각적 편집기나 YAML 편집기를 사용하는 것이 좋습니다. 이러한 편집기는 YAML 속성 이름, 값, 중첩, 간격, 대소문자 등이 올바른지 확인하는 데 도움이 되는 유용한 파일 검증 기능을 제공합니다.

다음 이미지는 비주얼 에디터의 워크플로를 보여줍니다. 비주얼 에디터는 워크플로우 정의 파일을 만들고 구성할 수 있는 완전한 사용자 인터페이스를 제공합니다. 비주얼 에디터에는 워크플로의 주요 구성 요소를 보여주는 워크플로 다이어그램 (1) 과 구성 영역 (2) 이 있습니다.



다음 이미지에 표시된 YAML 편집기를 사용할 수도 있습니다. YAML편집기를 사용하여 큰 코드 블록(예: 자습서의 코드 블록) 을 붙여넣거나 시각적 편집기를 통해 제공되지 않는 고급 속성을 추가할 수 있습니다.

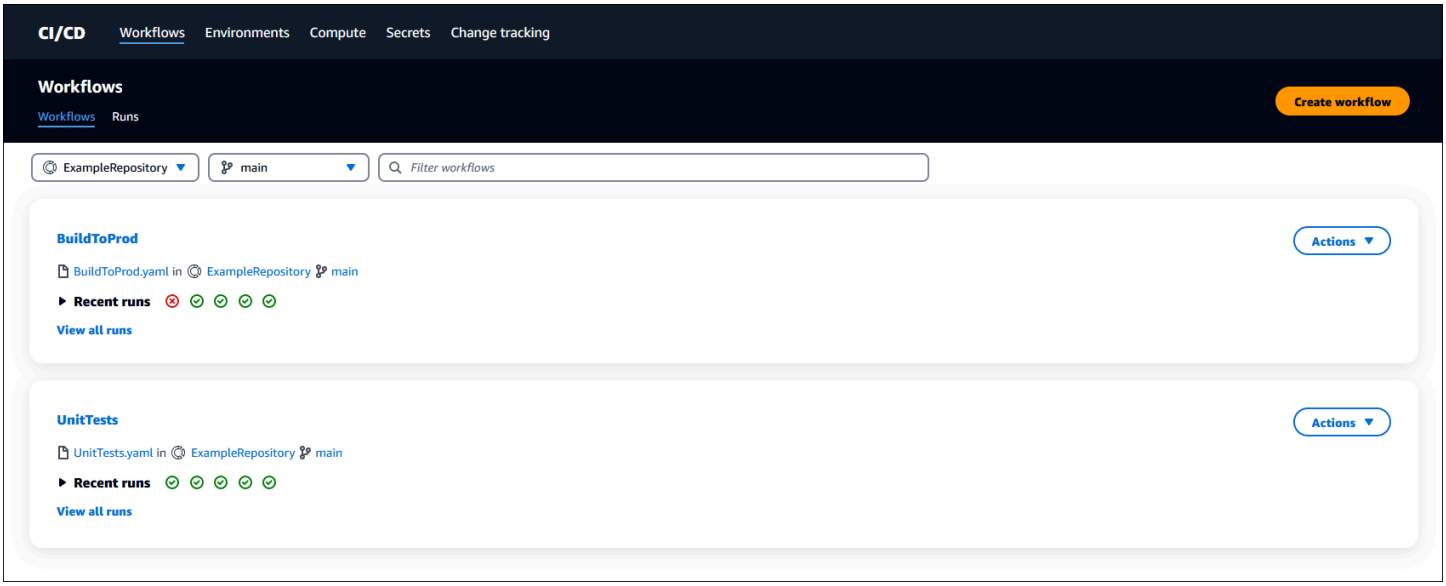


비주얼 편집기에서 편집기로 전환하여 구성이 기본 YAML 코드에 미치는 영향을 확인할 수 있습니다. YAML

워크플로우 검색

워크플로 요약 페이지에서 동일한 프로젝트에 설정한 다른 워크플로와 함께 워크플로를 볼 수 있습니다.

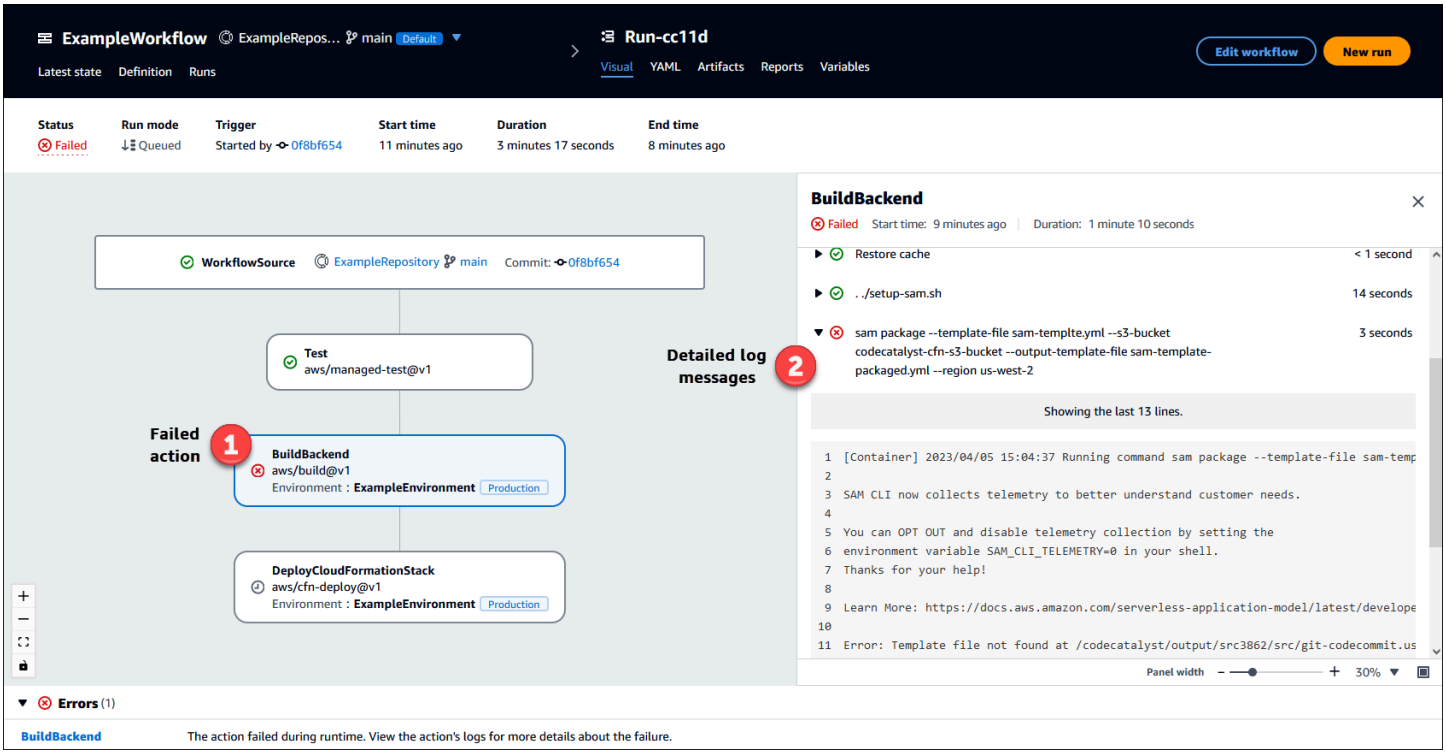
다음 이미지는 워크플로우 요약 페이지를 보여줍니다. 두 개의 워크플로로 채워져 있습니다: BuildToProd및 UnitTests. 둘 다 몇 번 실행된 것을 볼 수 있습니다. 최근 실행을 선택하여 실행 기록을 빠르게 보거나 워크플로 이름을 선택하여 워크플로의 YAML 코드 및 기타 세부 정보를 볼 수 있습니다.



워크플로 실행 세부 정보 보기

워크플로 요약 페이지에서 실행을 선택하여 워크플로 실행의 세부 정보를 볼 수 있습니다.

다음 이미지는 소스 커밋 시 자동으로 시작된 Run-CC11d라는 워크플로 실행의 세부 정보를 보여줍니다. 워크플로 다이어그램은 작업이 실패했음을 나타냅니다 (1). 로그 (2) 로 이동하여 자세한 로그 메시지를 보고 문제를 해결할 수 있습니다. 워크플로 실행에 대한 자세한 내용은 을 참조하십시오워크플로 실행.



다음 단계

워크플로우 개념에 대한 자세한 내용은 을 참조하십시오워크플로우 개념.

첫 워크플로를 만들려면 을 참조하십시오워크플로우 시작하기.

워크플로우 개념

워크플로를 사용하여 코드를 빌드, 테스트 또는 배포할 때 알아야 할 몇 가지 개념과 용어는 다음과 같습니다. CodeCatalyst

워크플로

워크플로는 CI/CD (지속적 통합 및 지속적 전달) 시스템의 일부로 코드를 빌드, 테스트 및 배포하는 방법을 설명하는 자동화된 절차입니다. 워크플로는 워크플로 실행 중에 수행할 일련의 단계 또는 조치를 정의합니다. 또한 워크플로는 워크플로를 시작하게 하는 이벤트 또는 트리거를 정의합니다. 워크플로를 설정하려면 CodeCatalyst 콘솔의 시각적 또는 YAML 편집기를 사용하여 워크플로 정의 파일을 만듭니다.

> **Tip**
>
> 프로젝트에서 워크플로를 사용하는 방법을 간단히 살펴보려면 청사진을 사용하여 프로젝트를 만들어 보세요. 각 블루프린트는 검토, 실행, 실험할 수 있는 작동하는 워크플로를 배포합니다.

워크플로 정의 파일

워크플로우 정의 파일은 워크플로를 설명하는 YAML 파일입니다. 기본적으로 파일은 소스 리포지토리의 루트에 있는 `~/.codecatalyst/workflows/` 폴더에 저장됩니다. 파일 확장자는.yml 또는.yaml 일 수 있으며 확장자는 소문자여야 합니다.

워크플로 정의 파일에 대한 자세한 내용은 을 참조하십시오. 워크플로우 YAML 정의

작업

작업은 워크플로의 기본 구성 요소이며, 워크플로 실행 중에 수행할 논리적 작업 단위 또는 작업을 정의합니다. 일반적으로 워크플로에는 구성 방법에 따라 순차적으로 또는 병렬로 실행되는 여러 작업이 포함됩니다.

작업에 대한 자세한 내용은 을 참조하십시오워크플로우 작업 구성.

액션 그룹

액션 그룹에는 하나 이상의 액션이 포함됩니다. 작업을 작업 그룹으로 그룹화하면 워크플로를 체계적으로 구성하고 여러 그룹 간의 종속성을 구성할 수 있습니다.

작업 그룹에 대한 자세한 내용은 을 참조하십시오. 작업을 작업 그룹으로 그룹화

아티팩트

아티팩트는 워크플로 작업의 출력이며, 일반적으로 폴더 또는 파일 아카이브로 구성됩니다. 아티팩트는 작업 간에 파일과 정보를 공유할 수 있게 해주므로 중요합니다.

아티팩트에 대한 자세한 내용은 작업 간 아티팩트 및 파일 공유 단원을 참조하십시오.

컴퓨팅

컴퓨트는 워크플로 작업을 CodeCatalyst 실행하기 위해 관리 및 유지 관리되는 컴퓨팅 엔진 (CPU, 메모리, 운영 체제) 을 말합니다.

컴퓨팅에 대한 자세한 내용은 을 참조하십시오컴퓨팅 및 런타임 이미지 구성.

환경

개발 CodeCatalyst 환경과 혼동하지 말아야 할 환경은 CodeCatalyst 워크플로가 연결되는 대상 AWS 계정 및 선택적 VPC Amazon을 정의합니다. 또한 환경은 워크플로가 대상 계정 내의 AWS 서비스와 리소스에 액세스하는 데 필요한 IAM역할을 정의합니다.

여러 환경을 설정하고 개발, 테스트, 스테이징, 프로덕션 등의 이름을 지정할 수 있습니다. 이러한 환경에 배포하는 경우 배포 관련 정보가 환경의 CodeCatalyst 배포 활동 및 배포 대상 탭에 표시됩니다.

환경에 대한 자세한 내용은 을 참조하십시오. 및 에 AWS 계정 배포 VPCs

게이츠

게이트는 특정 조건이 충족되지 않는 한 워크플로우 실행이 진행되지 않도록 하는 데 사용할 수 있는 워크플로 구성 요소입니다. 게이트의 예로는 사용자가 CodeCatalyst 콘솔에서 승인을 제출해야 워크플로 실행을 계속할 수 있는 승인 게이트가 있습니다.

워크플로의 작업 시퀀스 사이에 게이트를 추가하거나 소스 다운로드 직후에 실행되는 첫 번째 작업 앞에 게이트를 추가할 수 있습니다. 필요한 경우 마지막 작업 뒤에 게이트를 추가할 수도 있습니다.

게이트에 대한 자세한 내용은 을 참조하십시오워크플로 실행 게이팅.

보고서

보고서에는 워크플로 실행 중에 발생하는 테스트에 대한 세부 정보가 포함됩니다. 테스트 보고서, 코드 커버리지 보고서, 소프트웨어 구성 분석 보고서 및 정적 분석 보고서와 같은 보고서를 만들 수 있습니다. 보고서를 사용하면 워크플로 중에 문제를 해결하는 데 도움이 될 수 있습니다. 여러 워크플로의 보고서가 많은 경우 보고서를 사용하여 추세와 실패율을 확인하여 애플리케이션 및 배포 구성을 최적화할 수 있습니다.

보고서에 대한 자세한 내용은 품질 보고서 유형의 내용을 참조하세요.

실행

실행은 워크플로를 한 번 반복하는 것입니다. 실행 중에 워크플로 구성 파일에 정의된 작업을 CodeCatalyst 수행하고 관련 로그, 아티팩트 및 변수를 출력합니다.

실행에 대한 자세한 내용은 을 참조하십시오워크플로 실행.

소스

입력 소스라고도 하는 소스는 작업 수행에 필요한 파일을 얻기 위해 워크플로 작업이 연결되는 소스 저장소입니다. 예를 들어, 워크플로 작업을 소스 저장소에 연결하여 응용 프로그램을 빌드하기 위해 응용 프로그램 소스 파일을 가져올 수 있습니다.

소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

Variables

변수는 Amazon CodeCatalyst 워크플로에서 참조할 수 있는 정보가 포함된 키-값 쌍입니다. 워크플로가 실행되면 변수의 값 부분이 실제 값으로 대체됩니다.

변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

워크플로 트리거

워크플로 트리거 또는 단순한 트리거를 사용하면 코드 푸시와 같은 특정 이벤트가 발생할 때 자동으로 워크플로 실행을 시작할 수 있습니다. 소프트웨어 개발자가 CodeCatalyst 콘솔을 통해 수동으로 워크플로 실행을 시작하지 않아도 되도록 트리거를 구성하는 것이 좋습니다.

세 가지 유형의 트리거를 사용할 수 있습니다.

- 푸시 - 코드 푸시 트리거를 사용하면 커밋이 푸시될 때마다 워크플로가 실행됩니다.
- 풀 리퀘스트 — 풀 리퀘스트 트리거를 사용하면 풀 리퀘스트가 생성, 수정 또는 종료될 때마다 워크플로가 실행됩니다.
- 일정 — 일정 트리거를 사용하면 정의된 일정에 따라 워크플로가 실행됩니다. 소프트웨어 개발자가 다음 날 아침에 작업할 수 있도록 최신 빌드를 준비할 수 있도록 스케줄 트리거를 사용하여 야간 소프트웨어 빌드를 실행하는 것을 고려해 보세요.

푸시, 풀 리퀘스트 및 스케줄 트리거를 단독으로 사용하거나 동일한 워크플로우에서 조합하여 사용할 수 있습니다.

트리거는 선택 사항입니다. 구성하지 않은 경우 워크플로를 수동으로만 시작할 수 있습니다.

트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.

워크플로우 시작하기

이 자습서에서는 첫 번째 워크플로를 만들고 구성하는 방법을 알아봅니다.

> **Tip**
>
> 사전 구성된 워크플로로 시작하는 것을 선호하시나요? 에는 제대로 작동하는 워크플로블루프린트로 프로젝트 생성, 샘플 애플리케이션 및 기타 리소스가 포함된 프로젝트를 설정하는 방법에 대한 지침이 포함되어 있습니다.

주제

사전 조건

시작하기 전:

- CodeCatalyst 공간이 필요합니다. 자세한 내용은 스페이스 만들기 단원을 참조하십시오.
- CodeCatalyst 스페이스에는 다음과 같은 빈 프로젝트가 필요합니다.

  ```
  codecatalyst-project
  ```

  자세한 내용은 Amazon에서 빈 프로젝트 생성 CodeCatalyst 단원을 참조하십시오.
- 프로젝트에는 다음과 같은 CodeCatalyst 저장소가 필요합니다.

  ```
  codecatalyst-source-repository
  ```

  자세한 내용은 소스 리포지토리 생성 단원을 참조하십시오.

> **Note**
>
> 기존 프로젝트와 소스 리포지토리가 있는 경우 해당 리포지토리를 사용할 수 있지만 새 리포지토리를 만들면 이 튜토리얼의 마지막 부분에서 더 쉽게 정리할 수 있습니다.

1단계: 워크플로우 생성 및 구성

이 단계에서는 변경 시 소스 코드를 자동으로 빌드하고 테스트하는 워크플로를 만들고 구성합니다.

워크플로를 만들려면

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 워크플로 만들기를 선택합니다.

   워크플로 정의 파일이 CodeCatalyst 콘솔의 YAML 편집기에 나타납니다.

워크플로를 구성하려면

시각편집기 또는 편집기에서 워크플로를 YAML구성할 수 있습니다. 편집기부터 시작한 다음 비주얼 YAML 편집기로 전환해 보겠습니다.

1. 워크플로에 추가할 수 있는 워크플로 작업 목록을 보려면 + 작업을 선택합니다.
2. 빌드 작업에서 +를 선택하여 해당 작업을 워크플로 정의 파일에 추가합니다. YAML 이제 워크플로가 다음과 비슷해 보입니다.

```
Name: Workflow_fe47
SchemaVersion: "1.0"

# Optional - Set automatic triggers.
Triggers:
  - Type: Push
    Branches:
      - main

# Required - Define action configurations.
Actions:
  Build_f0:
    Identifier: aws/build@v1

    Inputs:
      Sources:
        - WorkflowSource # This specifies that the action requires this workflow as
 a source

    Outputs:
      AutoDiscoverReports:
```

```
        Enabled: true
        # Use as prefix for the report files
        ReportNamePrefix: rpt

    Configuration:
      Steps:
        - Run: echo "Hello, World!"
        - Run: echo "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>" >> report.xml
        - Run: echo "<testsuite tests=\"1\" name=\"TestAgentJunit\" >" >>
report.xml
        - Run: echo "<testcase classname=\"TestAgentJunit\" name=\"Dummy
            Test\"/></testsuite>" >> report.xml
```

워크플로는 WorkflowSource 소스 리포지토리의 파일을 Build_f0 작업을 실행하는 컴퓨팅 시스템에 복사하고, 로그에 Hello, World! 인쇄하고, 컴퓨팅 시스템에서 테스트 보고서를 검색하여 CodeCatalyst 콘솔의 보고서 페이지에 출력합니다.

3. 비주얼 에디터에서 워크플로 정의 파일을 보려면 Visual을 선택합니다. 비주얼 편집기의 필드를 사용하여 편집기에 표시되는 YAML 속성을 구성할 수 있습니다. YAML

2단계: 커밋과 함께 워크플로를 저장합니다.

이 단계에서는 변경 내용을 저장합니다. 워크플로는 저장소에 .yaml 파일로 저장되므로 변경 내용을 커밋과 함께 저장합니다.

워크플로 변경 내용을 적용하려면

1. (선택 사항) [Validate] 를 선택하여 워크플로의 YAML 코드가 유효한지 확인합니다.
2. 커밋을 선택합니다.
3. 워크플로 파일 이름에 워크플로 구성 파일의 이름 (예) 을 입력합니다**my-first-workflow**.
4. 커밋 메시지에 커밋을 식별하는 메시지 (예:) 를 입력합니다**create my-first-workflow.yaml**.
5. 리포지토리에서 워크플로를 저장할 리포지토리를 (codecatalyst-repository) 에서 선택합니다.
6. 브랜치 이름에서 워크플로를 저장할 브랜치 (main) 를 선택합니다.
7. 커밋을 선택합니다.

새 워크플로가 워크플로 목록에 표시됩니다. 표시되는 데 몇 분 정도 걸릴 수 있습니다.

워크플로는 커밋과 함께 저장되고 워크플로에 코드 푸시 트리거가 구성되어 있기 때문에 워크플로를 저장하면 워크플로가 자동으로 실행됩니다.

3단계: 실행 결과 보기

이 단계에서는 커밋에서 시작된 실행으로 이동하여 결과를 확인합니다.

실행 결과를 보려면

1. 워크플로의 이름을 선택합니다 (예:)`Workflow_fe47`.

   소스 리포지토리 (WorkflowSource) 및 빌드 작업 (예: Build_F0) 의 레이블을 보여주는 워크플로 다이어그램.
2. 워크플로 실행 다이어그램에서 빌드 작업 (예: Build_F0) 을 선택합니다.
3. 로그, 보고서, 구성 및 변수 탭의 내용을 검토하십시오. 이 탭은 빌드 작업의 결과를 보여줍니다.

   자세한 내용은 빌드 작업 결과 보기 단원을 참조하십시오.

(선택 사항) 4단계: 정리

이 단계에서는 이 튜토리얼에서 만든 리소스를 정리합니다.

리소스를 삭제하려면

- 이 자습서를 위해 새 프로젝트를 만든 경우 삭제하십시오. 지침은 프로젝트 삭제 단원을 참조하십시오. 프로젝트를 삭제하면 소스 리포지토리와 워크플로도 삭제됩니다.

워크플로를 사용한 구축

CodeCatalyst 워크플로를 사용하여 애플리케이션 및 기타 리소스를 빌드할 수 있습니다.

주제

- 애플리케이션을 빌드하려면 어떻게 해야 하나요?
- 빌드 작업의 이점
- 빌드 작업의 대안
- 빌드 액션 추가
- 빌드 작업 결과 보기

- 자습서: Amazon S3에 아티팩트 업로드
- 빌드 및 테스트 액션 YAML

애플리케이션을 빌드하려면 어떻게 해야 하나요?

에서 CodeCatalyst 애플리케이션이나 리소스를 빌드하려면 먼저 워크플로를 만든 다음 워크플로우 내에서 빌드 작업을 지정해야 합니다.

빌드 액션은 소스 코드를 컴파일하고, 단위 테스트를 실행하고, 배포할 준비가 된 아티팩트를 생성하는 워크플로 빌딩 블록입니다.

CodeCatalyst 콘솔의 시각적 편집기 또는 YAML 편집기를 사용하여 워크플로에 빌드 작업을 추가합니다.

애플리케이션 또는 리소스를 빌드하는 상위 단계는 다음과 같습니다.

애플리케이션을 빌드하려면 (고급 작업)

1. CodeCatalyst에서는 빌드하려는 애플리케이션의 소스 코드를 추가합니다. 자세한 내용은 프로젝트의 리포지토리에 소스 코드 저장 CodeCatalyst 단원을 참조하십시오.
2. 에서 CodeCatalyst 워크플로를 생성합니다. 워크플로는 애플리케이션을 빌드, 테스트 및 배포하는 방법을 정의하는 곳입니다. 자세한 내용은 워크플로우 시작하기 단원을 참조하십시오.
3. (선택 사항) 워크플로우에 워크플로를 자동으로 시작하게 하는 이벤트를 나타내는 트리거를 추가합니다. 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 단원을 참조하세요.
4. 워크플로우에서 애플리케이션 또는 리소스 소스 코드를 컴파일하고 패키징하는 빌드 작업을 추가합니다. 원하는 경우 이러한 목적으로 테스트 또는 배포 작업을 사용하지 않으려는 경우 빌드 작업으로 단위 테스트를 실행하고, 보고서를 생성하고, 애플리케이션을 배포하도록 할 수도 있습니다. 테스트 및 배포 작업에 대한 자세한 내용은 을 참조하십시오빌드 액션 추가.
5. (선택 사항) 워크플로우에서 테스트 작업과 배포 작업을 추가하여 애플리케이션 또는 리소스를 테스트하고 배포합니다. 사전 구성된 여러 작업 중에서 선택하여 ECS Amazon과 같은 다양한 대상에 애플리케이션을 배포할 수 있습니다. 자세한 정보는 워크플로를 사용한 테스트 및 워크플로를 통한 배포 섹션을 참조하세요.
6. 워크플로우는 수동 또는 트리거를 통해 자동으로 시작합니다. 워크플로는 빌드, 테스트 및 배포 작업을 순서대로 실행하여 애플리케이션과 리소스를 빌드, 테스트 및 대상에 배포합니다. 자세한 내용은 워크플로우 수동 실행 시작 단원을 참조하십시오.

빌드 작업의 이점

워크플로우 내에서 빌드 작업을 사용하면 다음과 같은 이점이 있습니다.

- 완전 관리형 — 빌드 작업을 수행하면 자체 빌드 서버를 설정, 패치, 업데이트, 관리할 필요가 없습니다.
- 온디맨드 — 빌드 작업은 빌드 요구 사항에 맞게 필요에 따라 확장됩니다. 사용한 빌드 시간만큼만 요금을 지불합니다. 자세한 내용은 컴퓨팅 및 런타임 이미지 구성 단원을 참조하십시오.
- 기본 제공 — 빌드 작업을 비롯한 모든 워크플로 작업을 실행하는 데 사용되는 사전 패키징된 런타임 환경 Docker 이미지가 CodeCatalyst 포함되어 있습니다. 이러한 이미지에는 응용 프로그램을 빌드하는 데 유용한 도구 (예: 및 Node.js) 가 AWS CLI 사전 구성되어 있습니다. 퍼블릭 또는 프라이빗 레지스트리에서 제공하는 빌드 이미지를 사용하도록 구성할 CodeCatalyst 수 있습니다. 자세한 내용은 런타임 환경 이미지 지정 단원을 참조하십시오.

빌드 작업의 대안

빌드 작업을 사용하여 애플리케이션을 배포하는 경우 배포 작업을 대신 사용해 보세요. CodeCatalyst 배포 작업은 빌드 작업을 사용하는 경우 수동으로 작성해야 하는 behind-the-scenes 구성을 수행합니다. 사용 가능한 배포 작업에 대한 자세한 내용은 을 참조하십시오배포 작업 목록.

를 사용하여 애플리케이션을 AWS CodeBuild 빌드할 수도 있습니다. 자세한 내용은 What is CodeBuild? 를 참조하십시오. .

빌드 액션 추가

CodeCatalyst 워크플로에 빌드 작업을 추가하려면 다음 절차를 따르세요.

Visual

비주얼 편집기를 사용하여 빌드 액션을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
3. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
4. 편집을 선택합니다.
5. Visual을 선택합니다.

6. 작업을 선택합니다.
7. [액션] 에서 [제작] 을 선택합니다.
8. 입력 및 구성 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오빌드 및 테스트 액션 YAML. 이 참조는 편집기와 시각적 편집기에 모두 나타나는 각 필드 (및 해당 YAML 속성 값) 에 대한 YAML 자세한 정보를 제공합니다.
9. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
10. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

YAML편집기를 사용하여 빌드 액션을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
3. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
4. 편집을 선택합니다.
5. 선택합니다 YAML.
6. 작업을 선택합니다.
7. [액션] 에서 [제작] 을 선택합니다.
8. 필요에 따라 YAML 코드의 속성을 수정하십시오. 사용 가능한 각 속성에 대한 설명은 에 나와 빌드 및 테스트 액션 YAML 있습니다.
9. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
10. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

액션 정의 빌드

빌드 액션은 워크플로 정의 파일 내의 YAML 속성 집합으로 정의됩니다. 이러한 속성에 대한 자세한 내용은 빌드 및 테스트 액션 YAML 을 참조하십시오워크플로우 YAML 정의.

빌드 작업 결과 보기

다음 지침을 사용하여 생성된 로그, 보고서, 변수를 비롯한 빌드 작업의 결과를 확인하세요.

빌드 작업의 결과를 보려면

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
3. 워크플로 다이어그램에서 빌드 작업의 이름 (예: Build) 을 선택합니다.
4. 빌드 실행 로그를 보려면 [Logs] 를 선택합니다. 다양한 빌드 단계의 로그가 표시됩니다. 필요에 따라 로그를 확장하거나 축소할 수 있습니다.
5. 빌드 작업으로 생성된 테스트 보고서를 보려면 [Reports] 를 선택하거나 탐색 창에서 [Reports] 를 선택합니다. 자세한 내용은 품질 보고서 유형 단원을 참조하십시오.
6. 빌드 작업에 사용된 구성을 보려면 구성을 선택합니다. 자세한 내용은 빌드 액션 추가 단원을 참조하십시오.
7. 빌드 작업에 사용되는 변수를 보려면 변수를 선택합니다. 자세한 내용은 워크플로우에서 변수 사용 단원을 참조하십시오.

자습서: Amazon S3에 아티팩트 업로드

이 자습서에서는 몇 가지 빌드 작업이 포함된 Amazon CodeCatalyst 워크플로를 사용하여 Amazon S3 버킷에 아티팩트를 업로드하는 방법을 알아봅니다. 이러한 작업은 워크플로가 시작될 때 순차적으로 실행됩니다. 첫 번째 빌드 작업은 파일 2개와 을 생성하여 빌드 아티팩트로 번들링합니다. `Hello.txt` `Goodbye.txt` 두 번째 빌드 작업은 아티팩트를 Amazon S3에 업로드합니다. 커밋을 소스 리포지토리로 푸시할 때마다 워크플로가 실행되도록 구성해야 합니다.

주제

- 사전 조건
- 1단계: AWS 역할 생성
- 2단계: Amazon S3 버킷 생성
- 3단계: 소스 리포지토리 생성
- 4단계: 워크플로우 만들기
- 5단계: 결과 확인
- 정리

사전 조건

시작하려면 다음이 필요합니다.

- 연결된 AWS 계정이 있는 CodeCatalyst 공간이 필요합니다. 자세한 내용은 스페이스 만들기 단원을 참조하십시오.
- 스페이스에 다음과 같은 빈 프로젝트가 필요합니다.

  ```
  codecatalyst-artifact-project
  ```

  처음부터 시작 옵션을 사용하여 이 프로젝트를 만들 수 있습니다.

  자세한 내용은 Amazon에서 빈 프로젝트 생성 CodeCatalyst 단원을 참조하십시오.
- 프로젝트에는 다음과 같은 CodeCatalyst 환경이 필요합니다.

  ```
  codecatalyst-artifact-environment
  ```

  이 환경을 다음과 같이 구성하십시오.
  - 원하는 유형 (예: 개발) 을 선택합니다.
  - AWS 계정을 여기에 연결하십시오.
  - 기본 IAM 역할의 경우 원하는 역할을 선택합니다. 나중에 다른 역할을 지정하게 됩니다.

  자세한 내용은 및 에 AWS 계정 배포 VPCs 단원을 참조하십시오.

1단계: AWS 역할 생성

이 단계에서는 나중에 워크플로의 빌드 작업에 할당할 AWS IAM 역할을 만듭니다. 이 역할은 CodeCatalyst 빌드 작업에 사용자 AWS 계정에 액세스하고 아티팩트가 저장될 Amazon S3에 쓸 수 있는 권한을 부여합니다. 이 역할을 빌드 역할이라고 합니다.

> **Note**
>
> 다른 자습서에서 만든 빌드 역할이 이미 있는 경우 이 자습서에도 사용할 수 있습니다. 다음 절차에 나와 있는 권한 및 신뢰 정책이 있는지 확인하기만 하면 됩니다.

IAM역할에 대한 자세한 내용은 AWS AWS Identity and Access Management 사용 설명서의 IAM 역할을 참조하십시오.

빌드 역할을 만들려면

1. 다음과 같이 역할에 대한 정책을 생성합니다.

   a. 에 AWS로그인하세요.

   b. 에서 IAM 콘솔을 엽니다 https://console.aws.amazon.com/iam/.

   c. 탐색 창에서 정책을 선택합니다.

   d. 정책 생성을 선택합니다.

   e. JSON탭을 선택합니다.

   f. 기존 코드를 삭제합니다.

   g. 다음 코드를 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름으로 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

   h. 다음: 태그를 선택합니다.

i. 다음: 검토를 선택합니다.

j. 이름에 다음을 입력합니다.

```
codecatalyst-s3-build-policy
```

k. 정책 생성을 선택합니다.

이제 권한 정책을 생성했습니다.

2. 다음과 같이 빌드 역할을 생성합니다.

a. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.

b. 사용자 지정 신뢰 정책을 선택합니다.

c. 기존 사용자 지정 신뢰 정책을 삭제합니다.

d. 다음 사용자 지정 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. Next(다음)를 선택합니다.

f. 권한 정책에서 해당 확인란을 `codecatalyst-s3-build-policy` 검색하여 선택합니다.

g. Next(다음)를 선택합니다.

h. 역할 이름에 다음을 입력합니다.

```
codecatalyst-s3-build-role
```

i. 역할 설명에 다음을 입력합니다.

```
CodeCatalyst build role
```

j. 역할 생성을 선택합니다.

이제 신뢰 정책 및 권한 정책을 사용하여 빌드 역할을 생성했습니다.

2단계: Amazon S3 버킷 생성

이 단계에서는 `Hello.txt` 및 `Goodbye.txt` 아티팩트가 업로드될 Amazon S3 버킷을 생성합니다.

Amazon S3 버킷을 생성하려면

1. 에서 Amazon S3 콘솔을 엽니다 https://console.aws.amazon.com/s3/.
2. 기본 창에서 버킷 생성을 선택합니다.
3. 버킷 이름에 다음을 입력합니다.

```
codecatalyst-artifact-bucket
```

4. AWS 지역에서 지역을 선택합니다. 이 자습서에서는 사용자가 미국 서부 (오레곤) us-west-2를 선택했다고 가정합니다. Amazon S3에서 지원하는 지역에 대한 자세한 내용은 의 Amazon 심플 스토리지 서비스 엔드포인트 및 할당량을 참조하십시오. AWS 일반 참조
5. 페이지 하단에서 버킷 생성을 선택합니다.
6. 방금 만든 버킷의 이름을 복사합니다. 예를 들면 다음과 같습니다.

```
codecatalyst-artifact-bucket
```

이제 미국 서부 (오레곤) **codecatalyst-artifact-bucket** us-west-2 지역에 버킷을 생성했습니다.

3단계: 소스 리포지토리 생성

이 단계에서는 에서 소스 리포지토리를 생성합니다 CodeCatalyst. 이 저장소는 자습서의 워크플로 정의 파일을 저장하는 데 사용됩니다.

소스 리포지토리에 대한 자세한 내용은 을 참조하십시오. 소스 리포지토리 생성

소스 리포지토리를 생성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다`codecatalyst-artifact-project`.
3. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
4. 리포지토리 추가를 선택하고 리포지토리 생성을 선택합니다.
5. 리포지토리 이름에 다음을 입력합니다.

```
codecatalyst-artifact-source-repository
```

6. 생성(Create)을 선택합니다.

라는 리포지토리가 생성되었습니다`codecatalyst-artifact-source-repository`.

4단계: 워크플로우 만들기

이 단계에서는 순차적으로 실행되는 다음과 같은 구성 블록으로 구성된 워크플로를 만듭니다.

- 트리거 - 이 트리거는 소스 리포지토리에 변경 내용을 푸시할 때 워크플로 실행을 자동으로 시작합니다. 트리거에 대한 자세한 내용은 을 참조하십시오트리거를 사용하여 자동으로 워크플로 실행 시작.
- 라는 빌드 작업 `GenerateFiles` - 트리거 시 `GenerateFiles` 작업은 파일 두 개를 만들고 이를 라는 출력 아티팩트로 패키징합니다. `Hello.txt` `Goodbye.txt` `codecatalystArtifact`
- 라고 하는 또 다른 빌드 `GenerateFiles` 작업 `Upload` — 작업이 완료되면 `Upload` 작업은 AWS CLI 명령을 `aws s3 sync` 실행하여 원본 리포지토리의 파일을 Amazon S3 버킷으로 업로드합니다. `codecatalystArtifact` CodeCatalyst컴퓨팅 플랫폼에 사전 설치 및 구성되어 AWS CLI 제공되므로 설치하거나 구성할 필요가 없습니다.

  CodeCatalyst 컴퓨팅 플랫폼에 사전 패키징된 소프트웨어에 대한 자세한 내용은 을 참조하십시오. 런타임 환경 이미지 지정 의 명령에 대한 자세한 내용은 `aws s3 sync` AWS CLI 명령 AWS CLI참조의 sync를 참조하십시오.

빌드 작업에 대한 자세한 내용은 을 참조하십시오워크플로를 사용한 구축.

워크플로 생성 방법

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 워크플로 만들기를 선택합니다.

3. YAML샘플 코드를 삭제합니다.
4. 다음 YAML 코드를 추가합니다.

> **Note**
>
> 다음 YAML 코드에서는 원하는 경우 해당 `Connections:` 섹션을 생략할 수 있습니다. 이 섹션을 생략하는 경우 환경의 기본 IAM 역할 필드에 지정된 역할에 에 설명된 권한 및 신뢰 정책이 포함되는지 확인해야 합니다. 1단계: AWS 역할 생성 기본 IAM 역할을 사용하는 환경을 설정하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

```
Name: codecatalyst-artifact-workflow
SchemaVersion: 1.0

Triggers:
  - Type: Push
    Branches:
      - main
Actions:
  GenerateFiles:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        # Create the output files.
        - Run: echo "Hello, World!" > "Hello.txt"
        - Run: echo "Goodbye!" > "Goodbye.txt"
    Outputs:
      Artifacts:
        - Name: codecatalystArtifact
          Files:
            - "**/*"
  Upload:
    Identifier: aws/build@v1
    DependsOn:
      - GenerateFiles
    Environment:
      Name: codecatalyst-artifact-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-s3-build-role
    Inputs:
```

```
    Artifacts:
      - codecatalystArtifact
  Configuration:
    Steps:
      # Upload the output artifact to the S3 bucket.
      - Run: aws s3 sync . s3://codecatalyst-artifact-bucket
```

위 코드에서 다음을 바꾸십시오.

- *codecatalyst-artifact-environment* 에서 만든 환경의 이름을 사용하세요사전 조건.
- *codecatalyst-account-connection* 에서 생성한 계정 연결의 이름을 사용하세요사전 조건.
- *codecatalyst-s3-build-role* 에서 만든 빌드 역할의 이름과 함께1단계: AWS 역할 생성.
- *codecatalyst-artifact-bucket* 에서 생성한 Amazon S3의 이름을 사용합니다2단계: Amazon S3 버킷 생성.

이 파일의 속성에 대한 자세한 내용은 를 참조하십시오빌드 및 테스트 액션 YAML.

5. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 YAML 코드가 유효한지 확인합니다.
6. 커밋을 선택합니다.
7. 워크플로 커밋 대화 상자에서 다음을 입력합니다.

   a. 워크플로 파일 이름의 경우 기본값인 을 그대로 유지합니다`codecatalyst-artifact-workflow`.

   b. 커밋 메시지에는 다음을 입력합니다.

   ```
   add initial workflow file
   ```

   c. 리포지토리의 경우 선택합니다 codecatalyst-artifact-source-repository.

   d. 브랜치 이름으로 main을 선택합니다.

   e. 커밋을 선택합니다.

이제 워크플로가 생성되었습니다. 워크플로 맨 위에 정의된 트리거로 인해 워크플로 실행이 자동으로 시작됩니다. 특히, `codecatalyst-artifact-workflow.yaml` 파일을 소스 리포지토리에 커밋 (및 푸시) 하면 트리거가 워크플로 실행을 시작했습니다.

실행 중인 워크플로를 보려면

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 방금 만든 워크플로를 선택합니다. `codecatalyst-artifact-workflow`
3. 첫 번째 빌드 작업 진행 상황을 GenerateFiles확인하도록 선택합니다.
4. 업로드를 선택하여 두 번째 빌드 작업 진행 상황을 확인하세요.
5. 업로드 작업이 완료되면 다음과 같이 하세요.
   - 워크플로가 성공적으로 실행되면 다음 절차로 이동합니다.
   - 워크플로 실행이 실패한 경우 [Logs] 를 선택하여 문제를 해결하십시오.

5단계: 결과 확인

워크플로가 실행되면 Amazon S3 서비스로 이동하여 다음을 살펴보십시오.*`codecatalyst-artifact-bucket`* 버킷. 이제 다음과 같은 파일 및 폴더가 포함되어야 합니다.

```
.
|— .aws/
|— .git/
|Goodbye.txt
|Hello.txt
|REAME.md
```

`Goodbye.txt`및 `Hello.txt` 파일은 `codecatalystArtifact` 아티팩트의 일부였기 때문에 업로드되었습니다. `.aws/`, `.git/`, 및 `README.md` 파일은 소스 저장소에 있었기 때문에 업로드되었습니다.

정리

CodeCatalyst 정리해서 이러한 서비스에 대한 요금이 부과되지 AWS 않도록 하세요.

안으로 청소하기 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. `codecatalyst-artifact-source-repository`소스 리포지토리를 삭제합니다.
3. `codecatalyst-artifact-workflow`워크플로를 삭제합니다.

에서 정리하려면 AWS

1. 다음과 같이 Amazon S3에서 정리합니다.
   a. 에서 Amazon S3 콘솔을 엽니다 https://console.aws.amazon.com/s3/.
   b. `codecatalyst-artifact-bucket`버킷에서 파일을 삭제합니다.
   c. `codecatalyst-artifact-bucket`버킷을 삭제합니다.
2. 다음과 IAM 같이 정리하세요.
   a. 에서 IAM 콘솔을 https://console.aws.amazon.com/iam/여십시오.
   b. `codecatalyst-s3-build-policy`를 삭제합니다.
   c. `codecatalyst-s3-build-role`를 삭제합니다.

빌드 및 테스트 액션 YAML

다음은 빌드 및 테스트 작업의 YAML 정의입니다. 두 액션의 YAML 속성이 매우 유사하기 때문에 두 액션에 대한 참조가 하나 있습니다.

이 작업 정의는 광범위한 워크플로 정의 파일 내에 섹션으로 존재합니다. 이 파일에 대한 자세한 내용은 워크플로우 YAML 정의을 참조합니다.

다음 코드에서 YAML 속성을 선택하면 해당 속성에 대한 설명이 표시됩니다.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

```
# The workflow definition starts here.
# See ### ## for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  action-name:
    Identifier: aws/build@v1 | aws/managed-test@v1
```

```
    DependsOn:
      - dependent-action-name-1
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Caching:
      FileCaching:
        key-name-1:
          Path: file1.txt
          RestoreKeys:
            - restore-key-1
    Inputs:
      Sources:
        - source-name-1
        - source-name-2
      Artifacts:
        - artifact-name
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
    Outputs:
      Artifacts:
        - Name: output-artifact-1
          Files:
            - build-output/artifact-1.jar
            - "build-output/build*"
        - Name: output-artifact-2
          Files:
            - build-output/artifact-2.1.jar
            - build-output/artifact-2.2.jar
      Variables:
        - variable-name-1
        - variable-name-2
      AutoDiscoverReports:
        Enabled: true | false
        ReportNamePrefix: AutoDiscovered
```

```
      IncludePaths:
        - "**/*"
      ExcludePaths:
        - node_modules/cdk/junit.xml
      SuccessCriteria:
        PassRate: percent
        LineCoverage: percent
        BranchCoverage: percent
        Vulnerabilities:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisBug:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisSecurity:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisQuality:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
        StaticAnalysisFinding:
          Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
          Number: whole-number
    Reports:
      report-name-1:
        Format: format
        IncludePaths:
          - "*.xml"
        ExcludePaths:
          - report2.xml
          - report3.xml
        SuccessCriteria:
          PassRate: percent
          LineCoverage: percent
          BranchCoverage: percent
          Vulnerabilities:
            Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
            Number: whole-number
          StaticAnalysisBug:
              Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
              Number: whole-number
          StaticAnalysisSecurity:
              Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
              Number: whole-number
```

```
            StaticAnalysisQuality:
                Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
                Number: whole-number
            StaticAnalysisFinding:
                Severity: CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL
                Number: whole-number
    Configuration:
      Container:
        Registry: registry
        Image: image
      Steps:
        - Run: "step 1"
        - Run: "step 2"
      Packages:
        NpmConfiguration:
          PackageRegistries:
            - PackagesRepository: package-repository
              Scopes:
                - "@scope"
        ExportAuthorizationToken: true | false
```

액션 이름

(필수)

액션의 이름을 지정합니다. 모든 작업 이름은 워크플로 내에서 고유해야 합니다. 액션 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 액션 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

해당 UI: 구성 탭/ 작업 이름

Identifier

(*action-name*/Identifier)

작업을 식별합니다. 버전을 변경하려는 경우가 아니면 이 속성을 변경하지 마십시오. 자세한 내용은 사용할 액션 버전 지정 단원을 참조하십시오.

빌드 `aws/build@v1` 작업에 사용합니다.

테스트 `aws/managed-test@v1` 작업에 사용합니다.

해당 UI: 워크플로 다이어그램/액션 이름/*aws/build@v1|aws/managed-test@v1* 레이블

DependsOn

(*action-name*/DependsOn)

(선택 사항)

이 액션을 실행하기 위해 성공적으로 실행되어야 하는 액션, 액션 그룹 또는 게이트를 지정하십시오.

'depends on' 기능에 대한 자세한 내용은 을 참조하십시오. 시퀀싱 액션

해당 UI: 입력 탭/ 기준 - 선택 사항

Compute

(*action-name*/Compute)

(선택 사항)

워크플로 작업을 실행하는 데 사용되는 컴퓨팅 엔진. 워크플로 수준 또는 작업 수준에서 컴퓨팅을 지정할 수 있지만 둘 다에서 지정할 수는 없습니다. 워크플로 수준에서 지정된 경우 컴퓨팅 구성은 워크플로에 정의된 모든 작업에 적용됩니다. 워크플로 수준에서는 동일한 인스턴스에서 여러 작업을 실행할 수도 있습니다. 자세한 내용은 작업 간 컴퓨팅 공유 단원을 참조하십시오.

해당 UI: 없음

Type

(*action-name*/Compute/유형)

(포함된 Compute 경우 필수)

컴퓨팅 엔진 유형. 다음 값 중 하나를 사용할 수 있습니다.

- EC2(비주얼 에디터) 또는 EC2 (YAML에디터)

  작업 실행 중 유연성을 위해 최적화되었습니다.
- Lambda (비주얼 에디터) 또는 (에디터) Lambda YAML

  작업 시작 속도를 최적화했습니다.

컴퓨팅 유형에 대한 자세한 정보는 컴퓨팅 유형을 참고하세요.

해당 UI: 구성 탭/ 컴퓨팅 유형

Fleet

(*action-name*/Compute/Fleet)

(선택 사항)

워크플로 또는 워크플로 작업을 실행할 시스템 또는 플릿을 지정합니다. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다. 온디맨드 플릿의 예:,. `Linux.x86-64.Large` `Linux.x86-64.XLarge` 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

생략된 경우 기본값은 `Fleet` 입니다. `Linux.x86-64.Large`

해당 UI: 구성 탭/ 컴퓨팅 플릿

Timeout

(*action-name*/Timeout)

(선택 사항)

작업이 CodeCatalyst 종료되기 전에 작업을 실행할 수 있는 시간을 분 (편집기) 또는 시간과 분 (비주얼 편집기) 단위로 지정합니다. YAML 최소값은 5분이고 최대값은 에 설명되어 의 워크플로우 할당량 CodeCatalyst 있습니다. 기본 타임아웃은 최대 타임아웃과 동일합니다.

해당 UI: 구성 탭/ 타임아웃 - 선택 사항

Environment

(*action-name*/Environment)

(선택 사항)

작업에 사용할 CodeCatalyst 환경을 지정합니다. 작업은 선택한 환경에 VPC 지정된 AWS 계정 및 선택적 Amazon에 연결됩니다. 작업은 환경에 지정된 기본 IAM 역할을 사용하여 Amazon에 연결하고 Amazon VPC 연결에 지정된 IAM 역할을 사용하여 Amazon에 연결합니다VPC. AWS 계정

> **Note**
>
> 기본 IAM 역할에 작업에 필요한 권한이 없는 경우 다른 역할을 사용하도록 작업을 구성할 수 있습니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.

환경에 대한 자세한 내용은 및 에 AWS 계정 배포 VPCs 및 을 참조하십시오환경 생성.

해당 UI: 구성 탭/ 환경

Name

(*action-name*/Environment/Name)

(선택 사항)

작업에 연결할 기존 환경의 이름을 지정합니다.

해당 UI: 구성 탭/ 환경

Connections

(*action-name*/Environment/Connections)

(선택 사항)

작업에 연결할 계정 연결을 지정합니다. 에서 최대 1개의 계정 연결을 지정할 수 `Environment` 있습니다.

계정 연결을 지정하지 않은 경우:

- 작업은 CodeCatalyst 콘솔 환경에 지정된 AWS 계정 연결 및 기본 IAM 역할을 사용합니다. 환경에 계정 연결 및 기본 IAM 역할을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.
- 기본 IAM 역할에는 작업에 필요한 정책 및 권한이 포함되어야 합니다. 이러한 정책 및 권한이 무엇인지 확인하려면 작업 YAML 정의 설명서에서 역할 속성에 대한 설명을 참조하십시오.

계정 연결에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정. 환경에 계정 연결을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

해당 UI: 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환

Name

(*action-name*/Environment/Connections/Name)

(포함된 경우 Connections 필수)

계정 연결 이름을 지정합니다.

해당 UI: 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환

Role

(*action-name*/Environment/Connections/Role)

(포함된 경우 Connections 필수)

Amazon S3 및 Amazon과 같은 AWS 서비스에 액세스하고 이를 운영하기 위해 이 작업이 사용하는 IAM 역할의 이름을 지정합니다ECR. 스페이스의 AWS 계정 연결에 이 역할이 추가되었는지 확인하십시오. 계정 연결에 IAM 역할을 추가하려면 을 참조하십시오계정 연결에 IAM 역할 추가.

IAM역할을 지정하지 않으면 CodeCatalyst 콘솔 환경에 나열된 기본 IAM 역할이 작업에 사용됩니다. 환경에서 기본 역할을 사용하는 경우 해당 역할에 다음 정책이 적용되는지 확인하십시오.

> **Note**
>
> 이 작업에 `CodeCatalystWorkflowDevelopmentRole-`*spaceName* 역할을 사용할 수 있습니다. 이에 대한 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기 섹션을 참조하세요. `CodeCatalystWorkflowDevelopmentRole-`*spaceName*역할에는 보안 위험을 초래할 수 있는 전체 액세스 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다.

> **Warning**
>
> 권한을 빌드 및 테스트 작업에 필요한 권한으로 제한하세요. 더 광범위한 권한이 있는 역할을 사용하면 보안 위험이 발생할 수 있습니다.

해당 UI: 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환

Caching

(*action-name*/Caching)

(선택 사항)

디스크에 있는 파일을 저장하고 후속 워크플로에서 해당 캐시에서 복원할 캐시를 지정할 수 있는 섹션이 실행됩니다.

파일 캐싱에 대한 자세한 내용은 을 참조하십시오. 워크플로우 실행 간 파일 캐싱

해당 UI: 구성 탭/ 파일 캐싱 - 선택 사항

FileCaching

(*action-name*/Caching/FileCaching)

(선택 사항)

캐시 시퀀스의 구성을 지정하는 섹션.

해당 UI: 구성 탭/파일 캐싱 - 선택 사항/ 캐시 추가

키 이름-1

(*action-name*/Caching/FileCaching/***key-name-1***)

(선택 사항)

기본 캐시 속성 이름을 지정합니다. 캐시 속성 이름은 워크플로우 내에서 고유해야 합니다. 각 작업에는 최대 5개의 항목이 포함될 수 `FileCaching` 있습니다.

해당 UI: 구성 탭/파일 캐싱 - 선택 사항/캐시 추가/키

Path

(*action-name*/Caching/FileCaching/*key-name-1*/Path)

(선택 사항)

캐시의 관련 경로를 지정합니다.

해당 UI: 구성 탭/파일 캐싱 - 선택 사항/캐시 추가/ 경로

RestoreKeys

(*action-name*/Caching/FileCaching/*key-name-1*/RestoreKeys)

(선택 사항)

기본 캐시 속성을 찾을 수 없을 때 폴백으로 사용할 복원 키를 지정합니다. 복원 키 이름은 워크플로우 내에서 고유해야 합니다. 각 캐시에는 최대 5개의 항목을 포함할 수 `RestoreKeys` 있습니다.

해당 UI: 구성 탭/파일 캐싱 - 선택 사항/캐시 추가/복원 키 - 선택 사항

Inputs

(*action-name*/Inputs)

(선택 사항)

이 `Inputs` 섹션에서는 워크플로우 실행 중에 작업에 필요한 데이터를 정의합니다.

> **Note**
>
> 빌드 작업 또는 테스트 작업당 최대 4개의 입력 (소스 1개와 아티팩트 3개) 이 허용됩니다. 변수는 이 총계에 포함되지 않습니다.

다른 입력 (예: 소스 및 아티팩트) 에 있는 파일을 참조해야 하는 경우 소스 입력은 기본 입력이고 아티팩트는 보조 입력입니다. 보조 입력의 파일에 대한 참조에는 기본 입력과 구분하기 위해 특수 접두사가 사용됩니다. 세부 정보는 예: 여러 아티팩트의 파일 참조을 참조하세요.

해당 UI: 입력 탭

Sources

(*action-name*/Inputs/Sources)

(선택 사항)

작업에 필요한 소스 리포지토리를 나타내는 레이블을 지정합니다. 현재 지원되는 유일한 레이블은 `WorkflowSource` 워크플로 정의 파일이 저장되는 소스 저장소를 나타내는 레이블입니다.

소스를 생략하는 경우 아래에 입력 아티팩트를 하나 이상 지정해야 합니다. *action-name*`/Inputs/Artifacts`

소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

해당 UI: 없음

Artifacts - input

(*action-name*/Inputs/Artifacts)

(선택 사항)

이 작업에 대한 입력으로 제공하려는 이전 작업의 아티팩트를 지정합니다. 이러한 아티팩트는 이전 작업에서 출력 아티팩트로 이미 정의되어 있어야 합니다.

입력 아티팩트를 지정하지 않는 경우 에서 하나 이상의 소스 리포지토리를 지정해야 합니다. *action-name*/Inputs/Sources

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

> **Note**
>
> 아티팩트 - 선택적 드롭다운 목록을 사용할 수 없거나 (비주얼 편집기) 를 검증할 때 오류가 발생하는 경우, 작업이 하나의 YAML 입력만 지원하기 때문일 수 있습니다. YAML 이런 경우에는 소스 입력을 제거해 보세요.

해당 UI: 입력 탭/ 아티팩트 - 선택 사항

Variables - input

(*action-name*/Inputs/Variables)

(선택 사항)

작업에 사용할 수 있도록 하려는 입력 변수를 정의하는 이름/값 쌍의 시퀀스를 지정합니다. 변수 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 변수 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

예제를 비롯한 변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

해당 UI: 입력 탭/ 변수 - 선택 사항

Outputs

(*action-name*/Outputs)

(선택 사항)

워크플로우 실행 중 작업에 의해 출력되는 데이터를 정의합니다.

해당 UI: 출력 탭

Artifacts - output

(*action-name*/Outputs/Artifacts)

(선택 사항)

액션으로 생성된 아티팩트의 이름을 지정합니다. 아티팩트 이름은 워크플로우 내에서 고유해야 하며 영숫자 (a-z, A-Z, 0-9) 와 밑줄 (_) 로 제한됩니다. 공백, 하이픈 (-) 및 기타 특수 문자는 사용할 수 없습니다. 출력 아티팩트 이름에 공백, 하이픈 및 기타 특수 문자를 사용할 때는 따옴표를 사용할 수 없습니다.

예를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 출력 탭/ 아티팩트

Name

(*action-name*/Outputs/Artifacts/Name)

(포함된 Artifacts - output 경우 필수)

액션으로 생성된 아티팩트의 이름을 지정합니다. 아티팩트 이름은 워크플로우 내에서 고유해야 하며 영숫자 (a-z, A-Z, 0-9) 와 밑줄 (_) 로 제한됩니다. 공백, 하이픈 (-) 및 기타 특수 문자는 사용할 수 없습니다. 출력 아티팩트 이름에 공백, 하이픈 및 기타 특수 문자를 사용할 때는 따옴표를 사용할 수 없습니다.

예를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 출력 탭/아티팩트/새 출력/빌드 아티팩트 이름

Files

(*action-name*/Outputs/Artifacts/Files)

Artifacts - output(포함된 경우 필수)

액션에 의해 출력되는 아티팩트에 CodeCatalyst 포함할 파일을 지정합니다. 이러한 파일은 워크플로 작업이 실행될 때 생성되며 소스 저장소에서도 사용할 수 있습니다. 파일 경로는 소스 리포지토리 또는 이전 작업의 아티팩트에 있을 수 있으며 소스 리포지토리 또는 아티팩트 루트를 기준으로 합니다. 글로브 패턴을 사용하여 경로를 지정할 수 있습니다. 예시:

- 빌드 위치 또는 소스 리포지토리 위치의 루트에 있는 단일 파일을 지정하려면 `my-file.jar`를 사용합니다..
- 하위 디렉터리에 단일 파일을 지정하려면 `directory/my-file.jar` 또는 `directory/subdirectory/my-file.jar`를 사용합니다.
- 모든 파일을 지정하려면 `"**/*"`를 사용합니다. ** glob 패턴은 임의의 수의 하위 디렉터리와 일치함을 나타냅니다.
- `directory`라는 디렉터리에 있는 모든 파일 및 디렉터리를 지정하려면 `"directory/**/*"`를 사용합니다. ** glob 패턴은 임의의 수의 하위 디렉터리와 일치함을 나타냅니다.
- `directory`라는 디렉터리의 모든 파일을 지정하되 해당 하위 디렉터리는 지정하지 않으려면 `"directory/*"`를 사용합니다.

> **Note**
>
> 파일 경로에 별표 (*) 또는 기타 특수 문자가 하나 이상 포함된 경우 경로를 큰따옴표 () 로 묶으십시오. "" 특수 문자에 대한 자세한 내용은 을 참조하십시오. 구문 지침 및 규칙

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오작업 간 아티팩트 및 파일 공유.

> **Note**
>
> 파일을 찾을 아티팩트나 소스를 나타내는 접두사를 파일 경로에 추가해야 할 수도 있습니다. 자세한 내용은 소스 리포지토리 파일 참조 및 아티팩트의 파일 참조 단원을 참조하세요.

해당 UI: 출력 탭/아티팩트/새 출력/빌드로 생성된 파일

Variables - output

(*`action-name`*/Outputs/Variables)

(선택 사항)

후속 작업에서 사용할 수 있도록 액션에서 내보낼 변수를 지정합니다.

예제를 비롯한 변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

해당 UI: 출력 탭/변수/ 변수 추가

변수 이름-1

(*action-name*/Outputs/Variables/*variable-name-1*)

(선택 사항)

액션으로 내보내려는 변수의 이름을 지정합니다. 이 변수는 동일한 액션의 `Inputs` 또는 `Steps` 섹션에 이미 정의되어 있어야 합니다.

예제를 비롯한 변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

해당 UI: 출력 탭/변수/변수 추가/이름

AutoDiscoverReports

(*action-name*/Outputs/AutoDiscoverReports)

(선택 사항)

자동 검색 기능의 구성을 정의합니다.

자동 CodeCatalyst 검색을 활성화하면 작업에 `Inputs` 전달된 모든 파일과 작업 자체에서 생성된 모든 파일을 검색하여 테스트, 코드 적용 범위 및 소프트웨어 구성 분석 (SCA) 보고서를 찾습니다. 발견된 각 보고서에 대해 보고서를 보고서로 CodeCatalyst 변환합니다. CodeCatalyst CodeCatalyst 보고서는 CodeCatalyst 서비스에 완전히 통합된 보고서이며 콘솔을 통해 보고 조작할 수 있습니다. CodeCatalyst

> **Note**
>
> 기본적으로 자동 검색 기능은 모든 파일을 검사합니다. 또는 속성을 사용하여 검사할 파일을 제한할 수 있습니다. IncludePaths ExcludePaths

해당 UI: 출력 탭/보고서/ 자동 검색 보고서

Enabled

(*action-name*/Outputs/AutoDiscoverReports/Enabled)

(선택 사항)

자동 검색 기능을 활성화하거나 비활성화합니다.

유효한 값은 `true` 또는 `false`입니다.

생략된 경우 기본값은 `Enabled` 입니다. `true`

해당 UI: 출력 탭/보고서/ 자동 검색 보고서

ReportNamePrefix

(*action-name*/Outputs/AutoDiscoverReports/ReportNamePrefix)

(포함되고 활성화된 경우 필수) AutoDiscoverReports

관련 보고서의 이름을 지정하려면 찾은 모든 보고서 CodeCatalyst 앞에 붙는 접두사를 지정하십시오. CodeCatalyst 예를 들어 접두사를 지정하고 두 개의 테스트 보고서를 CodeCatalyst 자동으로 검색하는 경우 관련 CodeCatalyst 보고서의 이름은 `TestSuiteOne.xml` 및 `TestSuiteTwo.xml` 로 지정됩니다. `AutoDiscovered` `AutoDiscoveredTestSuiteOne` `AutoDiscoveredTestSuiteTwo`

해당 UI: 출력 탭/보고서/ 접두사 이름

IncludePaths

(*action-name*/Outputs/AutoDiscoverReports/IncludePaths)

Or

(*action-name*/Outputs/Reports/*report-name-1*/IncludePaths)

(AutoDiscoverReports포함되고 활성화된 경우 또는 포함된 경우 필수) Reports

원시 보고서를 검색할 때 CodeCatalyst 포함할 파일 및 파일 경로를 지정합니다. 예를 들어`"/test/report/*"`, 지정하는 경우 는 작업에 사용된 전체 빌드 이미지를 CodeCatalyst 검색하여 `/test/report/*` 디렉터리를 찾습니다. 해당 디렉토리를 찾으면 해당 디렉토리에서 보고서를 찾습니다. CodeCatalyst

> **Note**
>
> 파일 경로에 별표 (*) 또는 기타 특수 문자가 하나 이상 포함된 경우 경로를 큰따옴표 () 로 묶으십시오. "" 특수 문자에 대한 자세한 내용은 을 참조하십시오. 구문 지침 및 규칙

이 속성을 생략하면 기본값은 입니다. 즉`"**/*"`, 모든 경로의 모든 파일이 검색에 포함됩니다.

> **Note**
>
> 수동으로 구성된 보고서의 경우 단일 `IncludePaths` 파일과 일치하는 글로브 패턴이어야 합니다.

해당 UI:

- 출력 탭/보고서/보고서 자동 검색/경로 포함/제외/경로 포함
- 출력 탭/보고서/보고서 수동 구성/*`report-name-1`*/경로 포함/제외/ 경로 포함

ExcludePaths

(*`action-name`*/Outputs/AutoDiscoverReports/ExcludePaths)

Or

(*`action-name`*/Outputs/Reports/*`report-name-1`*/ExcludePaths)

(선택 사항)

원시 보고서를 검색할 때 CodeCatalyst 제외할 파일 및 파일 경로를 지정합니다. 예를 들어`"/test/my-reports/**/*"`, 지정하는 경우 CodeCatalyst 는 `/test/my-reports/` 디렉터리에 있는 파일을 검색하지 않습니다. 디렉터리의 모든 파일을 무시하려면 `**/*` glob 패턴을 사용하십시오.

> **Note**
>
> 파일 경로에 별표 (*) 또는 기타 특수 문자가 하나 이상 포함된 경우 경로를 큰따옴표 () 로 묶으십시오. "" 특수 문자에 대한 자세한 내용은 을 참조하십시오. 구문 지침 및 규칙

해당 UI:

- 출력 탭/보고서/보고서 자동 검색/경로 포함/제외/경로 제외
- 출력 탭/보고서/보고서 수동 구성/*report-name-1*/경로 포함/제외/ 경로 제외

SuccessCriteria

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria)

(선택 사항)

테스트, 코드 커버리지, 소프트웨어 구성 분석 (SCA) 및 정적 분석 (SA) 보고서의 성공 기준을 지정합니다.

자세한 내용은 보고서의 성공 기준 구성 단원을 참조하십시오.

해당 UI: 출력 탭/보고서/성공 기준

PassRate

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/PassRate)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/PassRate)

(선택 사항)

관련 보고서를 통과한 것으로 표시되기 위해 통과해야 하는 테스트 CodeCatalyst 보고서의 테스트 비율을 지정합니다. 유효한 값에는 10진수가 포함됩니다. 예: `50`, `60.5`. 합격률 기준은 테스트 보고서에만 적용됩니다. 테스트 보고서에 대한 자세한 내용은 을 참조하십시오테스트 보고서.

해당 UI: 출력 탭/보고서/성공 기준/합격률

LineCoverage

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/LineCoverage)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/LineCoverage)

(선택 사항)

관련 보고서가 통과된 것으로 표시되기 위해 포함되어야 하는 코드 커버리지 보고서의 줄 비율을 지정합니다. CodeCatalyst 유효한 값에는 10진수가 포함됩니다. 예: `50`, `60.5`. 라인 커버리지 기준은 코드 커버리지 보고서에만 적용됩니다. 코드 커버리지 보고서에 대한 자세한 내용은 을 참조하십시오코드 범위 보고서.

해당 UI: 출력 탭/보고서/성공 기준/회선 커버리지

BranchCoverage

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/BranchCoverage)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/BranchCoverage)

(선택 사항)

관련 보고서가 통과된 것으로 표시되기 위해 포함되어야 하는 코드 커버리지 보고서의 분기 비율을 지정합니다. CodeCatalyst 유효한 값에는 10진수가 포함됩니다. 예: `50`, `60.5`. 브랜치 커버리지 기준은 코드 커버리지 보고서에만 적용됩니다. 코드 커버리지 보고서에 대한 자세한 내용은 을 참조하십시오코드 범위 보고서.

해당 UI: 출력 탭/보고서/성공 기준/브랜치 커버리지

Vulnerabilities

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/Vulnerabilities)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/Vulnerabilities)

(선택 사항)

관련 보고서가 통과된 것으로 표시되도록 SCA 보고서에 허용되는 최대 취약성 수와 심각도를 지정합니다. CodeCatalyst 취약성을 지정하려면 다음을 지정해야 합니다.

- 집계에 포함하려는 취약성의 최소 심각도. 가장 심각한 값부터 가장 심각하지 않은 값까지 유효한 값은,CRITICAL,, HIGH MEDIUMLOW, INFORMATIONAL 입니다.

  예를 들어HIGH, 선택하면 HIGH 및 CRITICAL 취약성이 집계됩니다.
- 허용하려는 지정된 심각도 내에서 허용되는 최대 취약성 수입니다. 이 수를 초과하면 CodeCatalyst 보고서가 실패로 표시됩니다. 유효한 값은 정수입니다.

취약성 기준은 SCA 보고서에만 적용됩니다. SCA보고서에 대한 자세한 내용은 을 참조하십시오소프트웨어 구성 분석 보고서.

최소 심각도를 지정하려면 Severity 속성을 사용하십시오. 최대 취약성 수를 지정하려면 Number 속성을 사용하십시오.

해당 UI: 출력 탭/보고서/성공 기준/취약성

StaticAnalysisBug

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/StaticAnalysisBug)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/StaticAnalysisBug)

(선택 사항)

관련 보고서가 통과된 것으로 표시되도록 SA 보고서에서 허용되는 최대 버그 수와 심각도를 지정하십시오. CodeCatalyst 버그를 지정하려면 다음을 지정해야 합니다.

- 카운트에 포함하려는 버그의 최소 심각도. 가장 심각한 값부터 가장 심각하지 않은 값까지 유효한 값은CRITICAL,,HIGH, MEDIUMLOW, INFORMATIONAL 입니다.

  예를 들어HIGH, 선택하면 HIGH CRITICAL 버그가 집계됩니다.
- 허용하려는 지정된 심각도의 최대 버그 수입니다. 이 숫자를 초과하면 CodeCatalyst 보고서가 실패로 표시됩니다. 유효한 값은 정수입니다.

버그 기준은 PyLint 및 ESLint SA 보고서에만 적용됩니다. SA 보고서에 대한 자세한 내용은 을 참조하십시오정적 분석 보고서.

최소 심각도를 지정하려면 Severity 속성을 사용하십시오. 최대 취약성 수를 지정하려면 Number 속성을 사용하십시오.

해당 UI: 출력 탭/보고서/성공 기준/버그

StaticAnalysisSecurity

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/StaticAnalysisSecurity)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/StaticAnalysisSecurity)

(선택 사항)

관련 보고서가 통과된 것으로 표시되도록 SA 보고서에서 허용되는 보안 취약성의 최대 수와 심각도를 지정하십시오. CodeCatalyst 보안 취약성을 지정하려면 다음을 지정해야 합니다.

- 카운트에 포함하려는 보안 취약성의 최소 심각도. 가장 심각한 값부터 가장 심각하지 않은 값까지 유효한 값은,`CRITICAL`,, `HIGH` `MEDIUMLOW`, `INFORMATIONAL` 입니다.

  예를 들어`HIGH`, 선택하면 `HIGH` `CRITICAL` 보안 취약성이 집계됩니다.
- 허용하려는 지정된 심각도의 최대 보안 취약성 수입니다. 이 수를 초과하면 CodeCatalyst 보고서가 실패로 표시됩니다. 유효한 값은 정수입니다.

보안 취약성 기준은 PyLint 및 ESLint SA 보고서에만 적용됩니다. SA 보고서에 대한 자세한 내용은 을 참조하십시오정적 분석 보고서.

최소 심각도를 지정하려면 `Severity` 속성을 사용하십시오. 최대 취약성 수를 지정하려면 `Number` 속성을 사용하십시오.

해당 UI: 출력 탭/보고서/성공 기준/보안 취약성

StaticAnalysisQuality

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/StaticAnalysisQuality)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/StaticAnalysisQuality)

(선택 사항)

관련 보고서가 통과된 것으로 표시되도록 SA 보고서에서 허용되는 품질 문제의 최대 수와 심각도를 지정하십시오. CodeCatalyst 품질 문제를 지정하려면 다음을 지정해야 합니다.

- 집계에 포함하려는 품질 문제의 최소 심각도. 가장 심각한 값부터 가장 심각하지 않은 값까지 유효한 값은CRITICAL,,HIGH, MEDIUMLOW, INFORMATIONAL 입니다.

  예를 들어HIGH, 선택하면 HIGH CRITICAL 품질 문제가 집계됩니다.
- 허용하려는 지정된 심각도의 품질 문제 최대 개수. 이 수를 초과하면 CodeCatalyst 보고서가 실패로 표시됩니다. 유효한 값은 정수입니다.

품질 문제 기준은 PyLint 및 ESLint SA 보고서에만 적용됩니다. SA 보고서에 대한 자세한 내용은 을 참조하십시오정적 분석 보고서.

최소 심각도를 지정하려면 Severity 속성을 사용하십시오. 최대 취약성 수를 지정하려면 Number 속성을 사용하십시오.

해당 UI: 출력 탭/보고서/성공 기준/품질 문제

StaticAnalysisFinding

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/StaticAnalysisFinding)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/StaticAnalysisFinding)

(선택 사항)

관련 보고서를 통과로 표시하기 위해 SA 보고서에서 허용되는 최대 결과 수와 심각도를 지정하십시오. CodeCatalyst 결과를 지정하려면 다음을 지정해야 합니다.

- 집계에 포함하려는 결과의 최소 심각도. 가장 심각한 값부터 가장 심각도가 낮은 값까지 유효한 값은CRITICAL,,HIGH, MEDIUMLOW, INFORMATIONAL 입니다.

  예를 들어 HIGHHIGH, 선택하면 CRITICAL 결과가 집계됩니다.
- 허용하려는 지정된 심각도의 최대 검색 결과 수입니다. 이 수를 초과하면 CodeCatalyst 보고서가 실패로 표시됩니다. 유효한 값은 정수입니다.

결과는 SARIF SA 보고서에만 적용됩니다. SA 보고서에 대한 자세한 내용은 을 참조하십시오정적 분석 보고서.

최소 심각도를 지정하려면 Severity 속성을 사용하십시오. 최대 취약성 수를 지정하려면 Number 속성을 사용하십시오.

해당 UI: 출력 탭/보고서/성공 기준/결과

Reports

(*action-name*/Outputs/Reports)

(선택 사항)

테스트 보고서의 구성을 지정하는 섹션.

해당 UI: 출력 탭/ 보고서

보고서 이름-1

(*action-name*/Outputs/Reports/보고서 이름-1)

(포함된 경우 필수) Reports

원시 보고서에서 생성될 CodeCatalyst 보고서에 부여하려는 이름.

해당 UI: 출력 탭/보고서/보고서 수동 구성/보고서 이름

Format

(*action-name*/Outputs/Reports/*report-name-1*/Format)

Reports(포함된 경우 필수)

보고서에 사용하는 파일 형식을 지정하십시오. 가능한 값은 다음과 같습니다.

- 테스트 보고서의 경우:
  - 오이의 경우 JSON Cucumber (비주얼 에디터) 또는 `CUCUMBERJSON` (YAML에디터) 를 지정하십시오.
  - 의 JUnit XML 경우 JUnit(비주얼 에디터) 또는 `JUNITXML` (YAML에디터) 를 지정하십시오.
  - 의 NUnit XML 경우 NUnit(비주얼 편집기) 또는 `NUNITXML` (YAML편집기) 를 지정하십시오.
  - NUnit3의 XML 경우 NUnit3(비주얼 에디터) 또는 `NUNIT3XML` (YAML에디터) 를 지정합니다.
  - 비주얼 TRX 스튜디오의 경우 Visual Studio TRX (비주얼 편집기) 또는 `VISUALSTUDIOTRX` (YAML편집기) 를 지정합니다.
  - TestNG의 경우 XML TestNG (비주얼 편집기) 또는 **`TESTNGXML`** (편집기) 를 지정하십시오. YAML

- 코드 커버리지 보고서의 경우:
  - Clover의 경우 Clover XML (비주얼 에디터) 또는 `CLOVERXML` (YAML에디터) 를 지정합니다.
  - Cobertura의 경우 Cobertura XML (비주얼 편집기) 또는 (편집기) 를 지정합니다. `COBERTURAXML` YAML
  - 의 경우 JaCoCo XML JaCoCo(비주얼 에디터) 또는 (에디터) 를 지정하십시오. `JACOCOXML` YAML
  - simplecov-json이 아닌 simplecov에서 SimpleCov JSON 생성한 경우 Simplecov (비주얼 편집기) 또는 (편집기) 로 지정하십시오. `SIMPLECOV` YAML
- 소프트웨어 구성 분석 () 보고서의 경우: SCA
  - 의 SARIF 경우 SARIF(비주얼 에디터) 또는 `SARIFSCA` (YAML에디터) 를 지정하십시오.

해당 UI: 출력 탭/보고서/보고서 수동 구성/보고서 추가/구성/*report-name-1*/보고서 유형 및 보고서 형식

Configuration

(*action-name*/Configuration)

(필수) 작업의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭

Container

(*action-name*/Configuration/Container)

(선택 사항)

작업이 처리를 완료하는 데 사용하는 Docker 이미지 또는 컨테이너를 지정합니다. 함께 CodeCatalyst 제공되는 활성 이미지 중 하나를 지정하거나 자체 이미지를 사용할 수 있습니다. 자체 이미지를 사용하기로 선택한 경우 AmazonECR, Docker Hub 또는 다른 레지스트리에 있을 수 있습니다. Docker 이미지를 지정하지 않으면 액션은 활성 이미지 중 하나를 처리에 사용합니다. 기본적으로 사용되는 활성 이미지에 대한 자세한 내용은 을 참조하십시오활성 이미지.

Docker 이미지를 직접 지정하는 방법에 대한 자세한 내용은 을 참조하십시오액션에 사용자 지정 런타임 환경 Docker 이미지 할당.

해당 UI: 런타임 환경 Docker 이미지 - 선택 사항

Registry

(*action-name*/Configuration/Container/Registry)

(포함된 `Container` 경우 필수)

이미지가 저장되는 레지스트리를 지정하십시오. 유효한 값으로는 다음이 포함됩니다.

- `CODECATALYST`(YAML편집자)

  이미지는 CodeCatalyst 레지스트리에 저장됩니다.
- Docker Hub (비주얼 에디터) 또는 `DockerHub` (YAML에디터)

  이미지는 Docker Hub 이미지 레지스트리에 저장됩니다.
- 기타 레지스트리 (비주얼 에디터) 또는 `Other` (YAML에디터)

  이미지는 사용자 지정 이미지 레지스트리에 저장됩니다. 공개적으로 사용 가능한 모든 레지스트리를 사용할 수 있습니다.
- Amazon Elastic 컨테이너 레지스트리 (비주얼 에디터) 또는 `ECR` (YAML에디터)

  이미지는 Amazon Elastic 컨테이너 레지스트리 이미지 리포지토리에 저장됩니다. Amazon ECR 리포지토리의 이미지를 사용하려면 이 작업을 수행하려면 Amazon에 대한 액세스 권한이 필요합니다 ECR. 이 액세스를 활성화하려면 다음 권한과 사용자 지정 신뢰 정책을 포함하는 IAM역할을 생성해야 합니다. (원하는 경우 권한과 정책을 포함하도록 기존 역할을 수정할 수 있습니다.)

  IAM역할의 역할 정책에 다음 권한이 포함되어야 합니다.

  - `ecr:BatchCheckLayerAvailability`
  - `ecr:BatchGetImage`
  - `ecr:GetAuthorizationToken`
  - `ecr:GetDownloadUrlForLayer`

  IAM역할에는 다음과 같은 사용자 지정 신뢰 정책이 포함되어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
```

```
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

IAM역할을 만드는 방법에 대한 자세한 내용은 사용 설명서의 사용자 지정 신뢰 정책을 사용하여 역할 만들기 (콘솔) 를 IAM참조하십시오.

역할을 만든 후에는 환경을 통해 작업에 할당해야 합니다. 자세한 내용은 환경을 액션과 연결하기 단원을 참조하십시오.

해당 UI: Amazon Elastic 컨테이너 레지스트리, Docker Hub 및 기타 레지스트리 옵션

Image

(*action-name*/Configuration/Container/Image)

(포함된 `Container` 경우 필수)

다음 중 하나를 지정하세요.

- `CODECATALYST`레지스트리를 사용하는 경우 이미지를 다음 활성 이미지 중 하나로 설정하십시오.
  - `CodeCatalystLinux_x86_64:2024_03`
  - `CodeCatalystLinux_x86_64:2022_11`
  - `CodeCatalystLinux_Arm64:2024_03`
  - `CodeCatalystLinux_Arm64:2022_11`
  - `CodeCatalystLinuxLambda_x86_64:2024_03`
  - `CodeCatalystLinuxLambda_x86_64:2022_11`
  - `CodeCatalystLinuxLambda_Arm64:2024_03`
  - `CodeCatalystLinuxLambda_Arm64:2022_11`
  - `CodeCatalystWindows_x86_64:2022_11`
- Docker Hub 레지스트리를 사용하는 경우 이미지를 Docker Hub 이미지 이름 및 선택적 태그로 설정합니다.

예제: `postgres:latest`

- Amazon ECR 레지스트리를 사용하는 경우 이미지를 Amazon ECR 레지스트리로 설정하십시오 URI.

  예제: `111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo`

- 사용자 지정 레지스트리를 사용하는 경우 이미지를 사용자 지정 레지스트리에서 예상하는 값으로 설정하십시오.

해당 UI: 런타임 환경 도커 이미지 (레지스트리가 있는 경우**CODECATALYST**), Docker Hub 이미지 (레지스트리가 Docker Hub인 경우), ECR이미지 URL (레지스트리가 Amazon Elastic Container 레지스트리인 경우), 이미지 URL (레지스트리가 기타 레지스트리인 경우).

Steps

(*action-name*/Configuration/Steps)

(필수)

빌드 도구를 설치, 구성 및 실행하기 위한 작업 중에 실행하려는 셸 명령을 지정합니다.

npm 프로젝트를 빌드하는 방법의 예는 다음과 같습니다.

```
Steps:
  - Run: npm install
  - Run: npm run build
```

파일 경로를 지정하는 방법의 예는 다음과 같습니다.

```
Steps:
  - Run: cd $ACTION_BUILD_SOURCE_PATH_WorkflowSource/app  && cat file2.txt
  - Run: cd $ACTION_BUILD_SOURCE_PATH_MyBuildArtifact/build-output/  && cat file.txt
```

파일 경로 지정에 대한 자세한 내용은 소스 리포지토리 파일 참조 및 을 참조하십시오아티팩트의 파일 참조.

해당 UI: 구성 탭/ 셸 명령

Packages

(*action-name*/Configuration/Packages)

(선택 사항)

작업이 종속성을 해결하는 데 사용하는 패키지 저장소를 지정할 수 있는 섹션입니다. 패키지를 사용하면 애플리케이션 개발에 사용되는 소프트웨어 패키지를 안전하게 저장하고 공유할 수 있습니다.

패키지에 대한 자세한 내용은 을 참조하십시오에서 소프트웨어 패키지 게시 및 공유 CodeCatalyst.

해당 UI: 구성 탭/ 패키지

NpmConfiguration

(*action-name*/Configuration/Packages/NpmConfiguration)

(포함된 Packages 경우 필수)

npm 패키지 형식의 구성을 정의하는 섹션입니다. 이 구성은 워크플로 실행 중 작업에 사용됩니다.

npm 패키지 구성에 대한 자세한 내용은 을 참조하십시오npm 사용.

해당 UI: 구성 탭/패키지/구성 추가/ npm

PackageRegistries

(*action-name*/Configuration/Packages/NpmConfiguration/PackageRegistries)

Packages(포함된 경우 필수)

패키지 리포지토리 시퀀스의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭/패키지/구성 추가/npm/ 패키지 리포지토리 추가

PackagesRepository

(*action-name*/Configuration/Packages/NpmConfiguration/PackageRegistries/PackagesRepository)

Packages(포함된 경우 필수)

작업에 사용할 CodeCatalyst 패키지 저장소의 이름을 지정합니다.

기본 리포지토리를 여러 개 지정하는 경우 마지막 리포지토리가 우선합니다.

패키지 리포지토리에 대한 자세한 내용은 을 참조하십시오. 패키지 리포지토리

해당 UI: 구성 탭/패키지/구성 추가/NPM/패키지 리포지토리 추가/패키지 리포지토리

Scopes

(*action-name*/Configuration/Packages/NpmConfiguration/PackageRegistries/Scopes)

(선택 사항)

패키지 레지스트리에서 정의하려는 범위 순서를 지정합니다. 범위를 정의할 때 지정된 패키지 저장소는 나열된 모든 범위의 레지스트리로 구성됩니다. npm 클라이언트를 통해 범위가 지정된 패키지를 요청하면 기본값 대신 해당 리포지토리를 사용합니다. 각 스코프 이름 앞에 “@”를 붙여야 합니다.

오버라이드 범위를 포함하면 마지막 리포지토리가 우선합니다.

를 생략하면 `Scopes` 지정된 패키지 저장소가 작업에 사용되는 모든 패키지의 기본 레지스트리로 구성됩니다.

범위에 대한 자세한 내용은 범위 지정 패키지를 참조하십시오패키지 네임스페이스.

해당 UI: 구성 탭/패키지/구성 추가/NPM/패키지 리포지토리 추가/범위 - 선택 사항

ExportAuthorizationToken

(*action-name*/Configuration/Packages/ExportAuthorizationToken)

(선택 사항)

내보내기 승인 토큰 기능을 활성화 또는 비활성화합니다. 활성화된 경우 내보낸 인증 토큰을 사용하여 패키지 관리자가 CodeCatalyst 패키지 리포지토리에서 인증하도록 수동으로 구성할 수 있습니다. 토큰을 작업에서 참조할 수 있는 환경 변수로 사용할 수 있습니다.

유효한 값은 `true` 또는 `false`입니다.

생략된 경우 기본값은 `ExportAuthorizationToken` 입니다. `false`

내보내기 인증 토큰에 대한 자세한 내용은 을 참조하십시오. 워크플로 작업에 인증 토큰 사용

해당 UI: 구성 탭/패키지/내보내기 인증 토큰

워크플로를 사용한 테스트

CodeCatalyst에서는 빌드 및 테스트와 같은 다양한 워크플로 작업의 일부로 테스트를 실행할 수 있습니다. 이러한 워크플로 작업은 모두 품질 보고서를 생성할 수 있습니다. 테스트 조치는 테스트, 코드 적용 범위, 소프트웨어 구성 분석 및 정적 분석 보고서를 생성하는 워크플로 작업입니다. 이러한 보고서는 CodeCatalyst 콘솔에 표시됩니다.

주제

품질 보고서 유형

Amazon CodeCatalyst 테스트 작업은 다음 유형의 품질 보고서를 지원합니다. 에서 이러한 보고서의 형식을 지정하는 방법에 대한 예는 YAML 을 참조하십시오품질 보고서 YAML 예제.

주제

테스트 보고서

에서는 CodeCatalyst 빌드 중에 실행되는 단위 테스트, 통합 테스트, 시스템 테스트를 구성할 수 있습니다. 그런 다음 테스트 결과가 포함된 보고서를 만들 CodeCatalyst 수 있습니다.

테스트 보고서를 사용하면 테스트 관련 문제를 해결하는 데 도움이 될 수 있습니다. 여러 빌드의 테스트 보고서가 많은 경우 테스트 보고서를 사용하여 실패율을 확인하여 빌드를 최적화할 수 있습니다.

다음과 같은 테스트 보고서 파일 형식을 사용할 수 있습니다.

- 오이 JSON (.json)
- JUnitXML(.xml)
- NUnitXML(.xml)
- NUnit3XML(.xml)
- TestNG XML (.xml)
- 비주얼 스튜디오 TRX (.trx, .xml)

코드 범위 보고서

에서 CodeCatalyst 테스트에 대한 코드 커버리지 보고서를 생성할 수 있습니다. CodeCatalyst 다음과 같은 코드 커버리지 메트릭을 제공합니다.

행 범위

테스트에서 다루는 명령문 수를 측정합니다. 명령문은 주석을 포함하지 않는 단일 명령입니다.

```
line coverage = (total lines covered)/(total number of lines)
```

분기 범위

`if`또는`case`문과 같은 제어 구조의 가능한 모든 분기 중에서 테스트가 다루는 분기 수를 측정합니다.

```
branch coverage = (total branches covered)/(total number of branches)
```

지원되는 코드 범위 보고서 파일 형식은 다음과 같습니다.

- JaCoCo XML(.xml)
- SimpleCov JSON(심플코브-json이 아닌 심플코브에서 생성됨, .json으로 생성됨)
- XML클로버 (버전 3, .xml)
- 커버리지 (.xml) XML
- LCOV(.info)

소프트웨어 구성 분석 보고서

CodeCatalyst에서는 소프트웨어 구성 분석 (SCA) 도구를 사용하여 애플리케이션 구성 요소를 분석하고 알려진 보안 취약성을 확인할 수 있습니다. 다양한 심각도의 취약성과 해결 방법을 자세히 설명하는

SARIF 보고서를 발견하고 분석할 수 있습니다. 가장 심각도가 높은 것부터 가장 심각하지 않은 것까지 유효한 심각도 값은,,,CRITICAL, HIGH 입니다. MEDIUM LOW INFORMATIONAL

지원되는 SCA 보고서 파일 형식은 다음과 같습니다.

- SARIF(.sarif, .json)

정적 분석 보고서

정적 분석 (SA) 보고서를 사용하여 소스 수준의 코드 결함을 식별할 수 있습니다. CodeCatalyst에서는 코드를 배포하기 전에 코드의 문제를 해결하는 데 도움이 되는 SA 보고서를 생성할 수 있습니다. 이러한 문제에는 버그, 보안 취약성, 품질 문제 및 기타 취약성이 포함됩니다. 가장 심각도가 높은 것부터 가장 심각하지 않은 것까지 유효한 심각도 값은,CRITICAL, HIGH MEDIUMLOW, 및 입니다. INFORMATIONAL

CodeCatalyst 다음과 같은 SA 메트릭을 제공합니다.

버그

소스 코드에서 발견될 수 있는 여러 버그를 식별합니다. 이러한 버그에는 메모리 안전 관련 문제가 포함될 수 있습니다. 다음은 버그의 예입니다.

```
// The while loop will inadvertently index into array x out-of-bounds
int x[64];
while (int n = 0; n <= 64; n++) {
  x[n] = 0;
}
```

보안 취약성

소스 코드에서 발견될 수 있는 여러 가지 보안 취약성을 식별합니다. 이러한 보안 취약성에는 비밀 토큰을 일반 텍스트로 저장하는 등의 문제가 포함될 수 있습니다.

품질 문제

소스 코드에서 발견될 수 있는 여러 가지 품질 문제를 식별합니다. 이러한 품질 문제에는 스타일 규칙과 관련된 문제가 포함될 수 있습니다. 다음은 품질 문제의 예입니다.

```
// The function name doesn't adhere to the style convention of camelCase
int SUBTRACT(int x, int y) {
```

```
  return x-y
}
```

기타 취약성

소스 코드에서 발견될 수 있는 여러 가지 기타 취약성을 식별합니다.

CodeCatalyst 다음과 같은 SA 보고서 파일 형식을 지원합니다.

- PyLint (.py)
- ESLint(.js, .jsx, .ts, .tsx)
- SARIF(.sarif, .json)

테스트 액션 추가

다음 절차를 사용하여 CodeCatalyst 워크플로에 테스트 작업을 추가합니다.

Visual

비주얼 편집기를 사용하여 테스트 동작을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
3. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
4. 편집을 선택합니다.
5. Visual을 선택합니다.
6. 작업을 선택합니다.
7. [액션] 에서 [테스트] 를 선택합니다.
8. 입력 및 구성 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오빌드 및 테스트 액션 YAML. 이 참조는 편집기와 시각적 편집기에 모두 나타나는 각 필드 (및 해당 YAML 속성 값) 에 대한 YAML 자세한 정보를 제공합니다.
9. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
10. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

YAML편집기를 사용하여 빌드 액션을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
3. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
4. 편집을 선택합니다.
5. 선택합니다 YAML.
6. 작업을 선택합니다.
7. [액션] 에서 [테스트] 를 선택합니다.
8. 필요에 따라 YAML 코드의 속성을 수정하십시오. 사용 가능한 각 속성에 대한 설명은 에 나와 빌드 및 테스트 액션 YAML 있습니다.
9. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
10. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

테스트 작업 정의

테스트 동작은 워크플로 정의 파일 내의 YAML 속성 집합으로 정의됩니다. 이러한 속성에 대한 자세한 내용은 빌드 및 테스트 액션 YAML 을 참조하십시오워크플로우 YAML 정의.

테스트 동작의 결과 보기

다음 지침을 사용하여 생성된 로그, 보고서, 변수를 포함한 테스트 작업의 결과를 확인하십시오.

테스트 조치의 결과를 보려면

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
3. 워크플로 다이어그램에서 테스트 작업의 이름 (예: Test) 을 선택합니다.
4. 작업으로 생성된 로그를 보려면 [Logs] 를 선택합니다. 다양한 작업 단계에 대한 로그가 표시됩니다. 필요에 따라 로그를 확장하거나 축소할 수 있습니다.

5. 테스트 동작으로 생성된 테스트 보고서를 보려면 [Reports] 를 선택하거나 탐색 창에서 [Reports] 를 선택합니다. 자세한 내용은 품질 보고서 유형 단원을 참조하십시오.
6. 테스트 작업에 사용된 구성을 보려면 구성을 선택합니다. 자세한 내용은 테스트 액션 추가 단원을 참조하십시오.
7. 테스트 작업에 사용된 변수를 보려면 변수를 선택합니다. 자세한 내용은 워크플로우에서 변수 사용 단원을 참조하십시오.

액션에서 실패한 테스트 건너뛰기

액션에 테스트 명령이 두 개 이상 있는 경우 이전 명령이 실패하더라도 액션의 후속 테스트 명령이 실행되도록 허용할 수 있습니다. 예를 들어 다음 명령에서는 `test1` 실패하더라도 항상 실행하는 `test2` 것이 좋습니다.

```
Steps:
- Run: npm install
- Run: npm run test1
- Run: npm run test2
```

일반적으로 단계에서 오류가 반환되면 Amazon은 워크플로 작업을 CodeCatalyst 중지하고 실패로 표시합니다. 오류 출력을 로 리디렉션하여 작업 단계가 계속 실행되도록 허용할 수 있습니다. `null` 명령에 `2>/dev/null` 추가하여 이 작업을 수행할 수 있습니다. 이렇게 수정하면 이전 예제는 다음과 같이 보일 것입니다.

```
Steps:
- Run: npm install
- Run: npm run test1 2>/dev/null
- Run: npm run test2
```

두 번째 코드 스니펫에서는 `npm install` 명령의 상태가 그대로 유지되지만 명령에서 반환되는 오류는 무시됩니다. `npm run test1` 그 결과 `npm run test2` 명령이 실행됩니다. 이렇게 하면 오류 발생 여부에 관계없이 두 보고서를 모두 한 번에 볼 수 있습니다.

다음과 통합하기 universal-test-runner

테스트 작업은 오픈 소스 명령줄 도구와 `universal-test-runner` 통합됩니다. `universal-test-runner`테스트 실행 프로토콜을 사용하여 지정된 프레임워크의 모든 언어에 대한 테스트를 실행합니다. `universal-test-runner`다음 프레임워크를 지원합니다.

- Gradle
- Jest
- Maven
- 파이테스트
- .NET

universal-test-runner테스트 작업을 위해 큐레이션된 이미지에만 설치됩니다. 사용자 지정 Docker Hub 또는 ECR Amazon을 사용하도록 테스트 작업을 구성하는 경우 고급 테스트 기능을 universal-test-runner 활성화하려면 수동으로 설치해야 합니다. 이렇게 하려면 이미지에 Node.js (14 이상) universal-test-runner 를 설치한 다음 shell 명령을 npm - Run: npm install -g @aws/universal-test-runner 사용하여 설치합니다. 셸 명령을 통해 컨테이너에 Node.js 를 설치하는 방법에 대한 자세한 내용은 노드 버전 관리자 설치 및 업데이트를 참조하십시오.

에 대한 universal-test-runner 자세한 내용은 universal-test-runner무엇입니까를 참조하십시오.

Visual

비주얼 universal-test-runner 에디터에서 사용하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
3. 워크플로의 이름을 선택합니다.
4. 편집을 선택합니다.
5. 비주얼을 선택합니다.
6. 작업을 선택합니다.
7. [액션] 에서 [테스트] 를 선택합니다.
8. 구성 탭에서 지원되는 프레임워크를 선택하여 샘플 코드를 업데이트하여 셸 명령 필드를 완성합니다. 예를 들어 지원되는 프레임워크를 사용하려면 다음과 비슷한 Run 명령을 사용합니다.

   ```
   - Run: run-tests <framework>
   ```

   원하는 프레임워크가 지원되지 않는 경우 사용자 지정 어댑터 또는 러너를 제공하는 것을 고려해 보십시오. 셸 명령 필드에 대한 설명은 을 참조하십시오Steps.
9. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로의 YAML 코드를 검증하십시오.

10. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

universal-test-runner YAML편집기에서 사용하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
3. 워크플로의 이름을 선택합니다.
4. 편집을 선택합니다.
5. 선택하세요 YAML.
6. 작업을 선택합니다.
7. [액션] 에서 [테스트] 를 선택합니다.
8. 필요에 따라 YAML 코드를 수정하십시오. 예를 들어 지원되는 프레임워크를 사용하려면 다음과 비슷한 Run 명령을 사용합니다.

```
Configuration:
  Steps:
    - Run: run-tests <framework>
```

원하는 프레임워크가 지원되지 않는 경우 사용자 지정 어댑터 또는 러너를 제공하는 것을 고려해 보십시오. Steps 속성에 대한 설명은 을 참조하십시오Steps.

9. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
10. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

작업의 품질 보고서 구성

이 섹션에서는 작업에서 품질 보고서를 구성하는 방법을 설명합니다.

주제

- 자동 검색 및 수동 보고서
- 보고서의 성공 기준 구성
- 품질 보고서 YAML 예제

자동 검색 및 수동 보고서

자동 검색이 활성화되면 작업에 전달된 모든 입력과 작업 자체에서 생성된 모든 파일을 CodeCatalyst 검색하여 테스트, 코드 적용 범위, 소프트웨어 구성 분석 (SCA) 및 SA (정적 분석) 보고서를 찾습니다. 에서 이러한 각 보고서를 보고 조작할 수 있습니다. CodeCatalyst

생성할 보고서를 수동으로 구성할 수도 있습니다. 생성하려는 보고서 유형과 파일 형식을 지정할 수 있습니다. 자세한 내용은 품질 보고서 유형 단원을 참조하십시오.

보고서의 성공 기준 구성

테스트, 코드 적용 범위, 소프트웨어 구성 분석 (SCA) 또는 정적 분석 (SA) 보고서의 성공 기준을 결정하는 값을 설정할 수 있습니다.

성공 기준은 보고서의 합격 또는 불합격 여부를 결정하는 임계값입니다. CodeCatalyst 먼저 테스트, 코드 커버리지 또는 SA 보고서가 될 수 있는 보고서를 생성한 다음 생성된 보고서에 성공 기준을 적용합니다. SCA 그런 다음 성공 기준의 충족 여부와 어느 정도까지 충족되었는지를 보여줍니다. 보고서가 지정된 성공 기준을 충족하지 않는 경우 성공 기준을 지정한 CodeCatalyst 작업이 실패합니다.

예를 들어 SCA 보고서의 성공 기준을 설정할 때 유효한 취약성 값 범위는 가장 심각한 것부터 가장 심각하지 않은 것까지 다양합니다. `CRITICAL` `HIGH` `MEDIUM` `LOW` `INFORMATIONAL` 심각도에 따라 취약성 하나를 검사하도록 기준을 설정한 경우 `HIGH` 심각도에 해당하는 취약성이 하나 이상 있거나 심각도가 없는 취약성이 하나 있지만 `HIGH` 심각도가 더 높은 취약성 (예: `HIGH` 심각도 수준의 취약성 하나)이 하나 이상 있으면 보고서가 실패합니다. `CRITICAL`

성공 기준을 지정하지 않는 경우:

- 원시 CodeCatalyst 보고서를 기반으로 생성된 보고서에는 성공 기준이 표시되지 않습니다.
- 성공 기준은 관련 워크플로 작업의 성공 또는 실패 여부를 결정하는 데 사용되지 않습니다.

Visual

성공 기준을 구성하려면

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 보고서를 생성하는 작업이 포함된 워크플로를 선택합니다. 성공 기준을 적용하려는 보고서입니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
3. 편집을 선택합니다.

4. Visual을 선택합니다.
5. 워크플로 다이어그램에서 CodeCatalyst 보고서를 생성하도록 구성한 작업을 선택합니다.
6. 출력 탭을 선택합니다.
7. 보고서 자동 검색 또는 보고서 수동 구성에서 성공 기준을 선택합니다.

   성공 기준이 표시됩니다. 이전 선택에 따라 다음 옵션 중 일부 또는 전체가 표시될 수 있습니다.

   합격률

   테스트 보고서에 있는 테스트 중 통과해야 관련 CodeCatalyst 보고서가 통과된 것으로 표시되는 테스트 비율의 비율을 지정합니다. 유효한 값에는 10진수가 포함됩니다. 예: `50`, `60.5`. 합격률 기준은 테스트 보고서에만 적용됩니다. 테스트 보고서에 대한 자세한 내용은 을 참조하십시오테스트 보고서.

   회선 커버리지

   관련 보고서가 통과된 것으로 표시되기 위해 포함되어야 하는 코드 커버리지 CodeCatalyst 보고서의 라인 비율을 지정합니다. 유효한 값에는 10진수가 포함됩니다. 예: `50`, `60.5`. 라인 커버리지 기준은 코드 커버리지 보고서에만 적용됩니다. 코드 커버리지 보고서에 대한 자세한 내용은 을 참조하십시오코드 범위 보고서.

   브랜치 커버리지

   코드 커버리지 보고서에서 관련 CodeCatalyst 보고서가 통과된 것으로 표시되기 위해 반드시 포함해야 하는 브랜치의 비율을 지정합니다. 유효한 값에는 10진수가 포함됩니다. 예: `50`, `60.5`. 브랜치 커버리지 기준은 코드 커버리지 보고서에만 적용됩니다. 코드 커버리지 보고서에 대한 자세한 내용은 을 참조하십시오코드 범위 보고서.

   취약성 () SCA

   관련 보고서가 통과된 것으로 표시되도록 SCA CodeCatalyst 보고서에 허용되는 최대 취약성 수와 심각도를 지정하십시오. 취약성을 지정하려면 다음을 지정해야 합니다.

   - 집계에 포함하려는 취약성의 최소 심각도. 가장 심각한 값부터 가장 심각하지 않은 값까지 유효한 값은,`CRITICAL`,, `HIGH` `MEDIUMLOW`, `INFORMATIONAL` 입니다.

     예를 들어`HIGH`, 선택하면 `HIGH` `CRITICAL` 취약성이 집계됩니다.
   - 허용하려는 지정된 심각도 내에서 허용되는 최대 취약성 수입니다. 이 수를 초과하면 CodeCatalyst 보고서가 실패로 표시됩니다. 유효한 값은 정수입니다.

취약성 기준은 SCA 보고서에만 적용됩니다. SCA보고서에 대한 자세한 내용은 을 참조하십시오소프트웨어 구성 분석 보고서.

버그

관련 CodeCatalyst 보고서가 통과된 것으로 표시되도록 SA 보고서에서 허용되는 최대 버그 수와 심각도를 지정하십시오. 버그를 지정하려면 다음을 지정해야 합니다.

- 카운트에 포함하려는 버그의 최소 심각도. 가장 심각한 값부터 가장 심각하지 않은 값까지 유효한 값은CRITICAL,,HIGH, MEDIUMLOW, INFORMATIONAL 입니다.

  예를 들어HIGH, 선택하면 HIGH CRITICAL 버그가 집계됩니다.
- 허용하려는 지정된 심각도의 최대 버그 수입니다. 이 숫자를 초과하면 CodeCatalyst 보고서가 실패로 표시됩니다. 유효한 값은 정수입니다.

버그 기준은 PyLint 및 ESLint SA 보고서에만 적용됩니다. SA 보고서에 대한 자세한 내용은 을 참조하십시오정적 분석 보고서.

보안 취약성

관련 CodeCatalyst 보고서가 통과된 것으로 표시되도록 SA 보고서에서 허용되는 보안 취약성의 최대 수와 심각도를 지정하십시오. 보안 취약성을 지정하려면 다음을 지정해야 합니다.

- 카운트에 포함하려는 보안 취약성의 최소 심각도. 가장 심각한 값부터 가장 심각하지 않은 값까지 유효한 값은,CRITICAL,, HIGH MEDIUMLOW, INFORMATIONAL 입니다.

  예를 들어HIGH, 선택하면 HIGH CRITICAL 보안 취약성이 집계됩니다.
- 허용하려는 지정된 심각도의 최대 보안 취약성 수입니다. 이 수를 초과하면 CodeCatalyst 보고서가 실패로 표시됩니다. 유효한 값은 정수입니다.

보안 취약성 기준은 PyLint 및 ESLint SA 보고서에만 적용됩니다. SA 보고서에 대한 자세한 내용은 을 참조하십시오정적 분석 보고서.

품질 문제

관련 CodeCatalyst 보고서를 통과로 표시하기 위해 SA 보고서에서 허용되는 품질 문제의 최대 수와 심각도를 지정하십시오. 품질 문제를 지정하려면 다음을 지정해야 합니다.

- 집계에 포함하려는 품질 문제의 최소 심각도. 가장 심각한 값부터 가장 심각하지 않은 값까지 유효한 값은CRITICAL,,HIGH, MEDIUMLOW, INFORMATIONAL 입니다.

  예를 들어HIGH, 선택하면 HIGH CRITICAL 품질 문제가 집계됩니다.
- 허용하려는 지정된 심각도의 품질 문제 최대 수입니다. 이 수를 초과하면 CodeCatalyst 보고서가 실패로 표시됩니다. 유효한 값은 정수입니다.

품질 문제 기준은 ESLint SA 보고서에만 적용됩니다 PyLint . SA 보고서에 대한 자세한 내용은 을 참조하십시오정적 분석 보고서.

8. 커밋을 선택합니다.
9. 워크플로를 실행하여 원시 보고서에 성공 기준을 CodeCatalyst 적용하고 성공 기준 정보가 포함된 관련 CodeCatalyst 보고서를 다시 생성하십시오. 자세한 내용은 워크플로우 수동 실행 시작 단원을 참조하십시오.

YAML

성공 기준을 구성하려면

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 보고서를 생성하는 작업이 포함된 워크플로를 선택합니다. 성공 기준을 적용하려는 보고서입니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
3. 편집을 선택합니다.
4. 선택합니다 YAML.
5. 워크플로 다이어그램에서 CodeCatalyst 보고서를 생성하도록 구성한 작업을 선택합니다.
6. 세부 정보 창에서 출력 탭을 선택합니다.
7. 액션, `AutoDiscoverReports` 섹션 또는 섹션에서,,, `PassRate` `LineCoverage` `BranchCoverage` `Vulnerabilities` `StaticAnalysisBug`StaticAnalysisSecurity, 및 SuccessCriteria속성과 함께 `StaticAnalysisQuality` 속성을 추가합니다. `Reports`

   각 속성에 대한 설명은 를 참조하십시오빌드 및 테스트 액션 YAML.
8. 커밋을 선택합니다.

9. 워크플로를 실행하여 원시 보고서에 성공 기준을 CodeCatalyst 적용하고 성공 기준 정보가 포함된 관련 CodeCatalyst 보고서를 재생성하십시오. 워크플로우 시작에 대한 자세한 내용은 을 참조하십시오워크플로우 수동 실행 시작.

품질 보고서 YAML 예제

다음 예제는 테스트 보고서, 코드 커버리지 보고서, 소프트웨어 구성 분석 보고서, 정적 분석 보고서 등 네 가지 보고서를 수동으로 구성하는 방법을 보여줍니다.

```
Reports:
  MyTestReport:
    Format: JUNITXML
    IncludePaths:
      - "*.xml"
    ExcludePaths:
      - report1.xml
      SuccessCriteria:
        PassRate: 90
  MyCoverageReport:
    Format: CLOVERXML
    IncludePaths:
      - output/coverage/jest/clover.xml
      SuccessCriteria:
        LineCoverage: 75
        BranchCoverage: 75
  MySCAReport:
    Format: SARIFSCA
    IncludePaths:
      - output/sca/reports.xml
      SuccessCriteria:
        Vulnerabilities:
          Number: 5
          Severity: HIGH
  MySAReport:
    Format: ESLINTJSON
    IncludePaths:
      - output/static/eslint.xml
      SuccessCriteria:
        StaticAnalysisBug:
          Number: 10
          Severity: MEDIUM
        StaticAnalysisSecurity:
```

```
          Number: 5
          Severity: CRITICAL
        StaticAnalysisQuality:
          Number: 0
          Severity: INFORMATIONAL
```

테스트 모범 사례

에서 제공하는 CodeCatalyst 테스트 기능을 사용할 때는 다음 모범 사례를 따르는 것이 좋습니다.

주제

- 자동 검색
- 성공 기준
- 경로 포함/제외

자동 검색

에서 CodeCatalyst 작업을 구성할 때 자동 검색을 통해 JUnit 테스트 보고서와 같은 다양한 도구의 결과를 자동으로 검색하고 해당 도구에서 관련 CodeCatalyst 보고서를 생성할 수 있습니다. 자동 검색을 통해 검색된 출력의 이름이나 경로가 변경되더라도 보고서가 계속 생성되도록 할 수 있습니다. 새 파일이 추가되면 CodeCatalyst 자동으로 파일을 검색하고 관련 보고서를 생성합니다. 그러나 자동 검색을 사용하는 경우 이 기능의 다음 측면 중 일부를 고려하는 것이 중요합니다.

- 작업에서 자동 검색을 활성화하면 동일한 유형의 자동 검색된 모든 보고서가 동일한 성공 기준을 공유합니다. 예를 들어, 자동 검색된 모든 테스트 보고서에 최소 합격률과 같은 공유 기준이 적용됩니다. 동일한 유형의 보고서에 대해 서로 다른 기준이 필요한 경우 이러한 각 보고서를 명시적으로 구성해야 합니다.
- 자동 검색을 통해 종속 관계에 따라 생성된 보고서를 찾을 수도 있으며, 성공 기준이 구성된 경우 이러한 보고서에 대한 작업이 실패할 수 있습니다. 이 문제는 제외 경로 구성을 업데이트하여 해결할 수 있습니다.
- 자동 검색은 런타임에 작업을 스캔하기 때문에 매번 동일한 보고서 목록을 생성한다고 보장할 수 없습니다. 특정 보고서가 항상 생성되도록 하려면 보고서를 명시적으로 구성해야 합니다. 예를 들어 빌드의 일부로 테스트 실행이 중지되면 테스트 프레임워크는 출력을 생성하지 않으므로 테스트 보고서가 생성되지 않아 작업이 성공할 수 있습니다. 특정 테스트에 따라 작업의 성공 여부가 결정되도록 하려면 해당 보고서를 명시적으로 구성해야 합니다.

> **Tip**
>
> 새 프로젝트 또는 기존 프로젝트를 시작할 때는 전체 프로젝트 디렉터리 (include**/*) 에 대해 자동 검색을 사용하세요. 그러면 하위 디렉터리에 있는 파일을 포함하여 프로젝트의 모든 파일에 대한 보고서가 생성됩니다.

자세한 내용은 작업의 품질 보고서 구성 단원을 참조하십시오.

성공 기준

성공 기준을 구성하여 보고서에 품질 임계값을 적용할 수 있습니다. 예를 들어, 회선 적용 범위가 80% 이고 다른 하나는 회선 적용 범위가 60% 인 두 개의 코드 적용 범위 보고서가 자동 검색된 경우 다음 옵션을 사용할 수 있습니다.

- 회선 커버리지의 자동 검색 성공 기준을 80% 로 설정합니다. 이렇게 하면 첫 번째 보고서는 통과하고 두 번째 보고서는 실패하여 전체 작업이 실패하게 됩니다. 워크플로의 차단을 해제하려면 두 번째 보고서의 라인 커버리지가 80% 를 초과할 때까지 프로젝트에 새 테스트를 추가하세요.
- 회선 커버리지의 자동 검색 성공 기준을 60% 로 설정합니다. 이렇게 하면 두 보고서가 모두 통과되어 작업이 성공하게 됩니다. 그런 다음 두 번째 보고서에서 코드 적용 범위를 확대할 수 있습니다. 하지만 이 방법을 사용하면 첫 번째 보고서의 적용 범위가 80% 미만으로 떨어지지 않는다고 보장할 수 없습니다.
- 시각적 편집기를 사용하거나 각 보고서에 대해 명시적인 YAML 섹션 및 경로를 추가하여 보고서 중 하나 또는 둘 다를 명시적으로 구성하십시오. 이렇게 하면 각 보고서에 대해 별도의 성공 기준과 사용자 지정 이름을 구성할 수 있습니다. 하지만 이 방법을 사용하면 보고서 경로가 변경될 경우 작업이 실패할 수 있습니다.

자세한 내용은 보고서의 성공 기준 구성 단원을 참조하십시오.

경로 포함/제외

작업 결과를 검토할 때 및 를 구성하여 CodeCatalyst 생성된 보고서 목록을 조정할 수 있습니다. `IncludePaths` `ExcludePaths`

- 보고서를 검색할 때 포함하려는 파일 및 파일 경로를 지정하는 CodeCatalyst 데 사용합니다`IncludePaths`. 예를 들어`"/test/report/*"`, 지정하는 경우 는 작업에 사용된 전체 빌드 이미지를 CodeCatalyst 검색하여 `/test/report/` 디렉토리를 찾습니다. 해당 디렉토리를 찾으면 해당 디렉토리에서 보고서를 찾습니다. CodeCatalyst

> **Note**
>
> 수동으로 구성된 보고서의 경우 단일 파일과 일치하는 글로브 `IncludePaths` 패턴이어야 합니다.

- 보고서를 검색할 때 제외하려는 파일 및 파일 경로를 지정하는 CodeCatalyst 데 사용합니다`ExcludePaths`. 예를 들어`"/test/reports/**/*"`, 지정하는 경우 CodeCatalyst 은 `/test/reports/` 디렉터리에 있는 파일을 검색하지 않습니다. 디렉터리의 모든 파일을 무시하려면 `**/*` glob 패턴을 사용하십시오.

다음은 가능한 글로브 패턴의 예입니다.

패턴	설명
`*.*`	현재 디렉터리에서 점이 있는 모든 개체 이름과 일치합니다.
`*.xml`	로 끝나는 현재 디렉터리의 모든 개체 이름과 일치합니다. `.xml`
`*.{xml,txt}`	`.xml`또는 로 끝나는 현재 디렉터리의 모든 개체 이름과 일치합니다. `.txt`
`**/*.xml`	로 끝나는 모든 디렉토리의 개체 이름을 일치시킵니다. `.xml`
`testFolder`	호출된 객체를 `testFolder` 일치시켜 파일로 취급합니다.
`testFolder/*`	하위 폴더의 한 수준에 있는 객체를 매칭합니다 (예: `testFolder` `testFolder/file.xml`
`testFolder/*/*`	하위 폴더의 두 수준에 있는 객체를 일치시킵니다 (예`testFolder` : `testFolder/reports Folder/file.xml`

패턴	설명
`testFolder/**`	`testFolder` 아래 파일뿐만 아니라 하위 폴더 `testFolder` 도 일치시킵니다(예: `testFolder/file.xml` 와 `testFolder/otherFolder/file.xml`).

CodeCatalyst 글로브 패턴을 다음과 같이 해석합니다.

- 슬래시 (/) 문자는 파일 경로의 디렉토리를 구분합니다.
- 별표(*) 문자는 폴더의 경계를 넘지 않고 0개 이상의 이름 구성 요소 문자와 해당합니다.
- 이중 별표(**)는 모든 디렉터리에서 이름 구성 요소의 0개 이상 문자와 일치합니다.

> **Note**
>
> `ExcludePaths`보다 우선합니다. `IncludePaths` 둘 다 `IncludePaths` 같은 폴더를 `ExcludePaths` 포함하는 경우 해당 폴더에서 보고서를 검색하지 않습니다.

지원되는 SARIF 속성

정적 분석 결과 교환 형식 (SARIF) 은 CodeCatalyst Amazon의 소프트웨어 구성 분석 (SCA) 및 정적 분석 보고서에서 사용할 수 있는 출력 파일 형식입니다. 다음 예제는 정적 분석 보고서에서 수동으로 구성하는 SARIF 방법을 보여줍니다.

```
Reports:
MySAReport:
Format: SARIFSA
IncludePaths:
    - output/sa_report.json
SuccessCriteria:
    StaticAnalysisFinding:
    Number: 25
    Severity: HIGH
```

CodeCatalyst 보고서에 분석 결과가 표시되는 방식을 최적화하는 데 사용할 수 있는 다음 SARIF 속성을 지원합니다.

주제

`sarifLog` 객체

명칭	필수	설명
`$schema`	예	버전 URI 2.1.0의 SARIF JSON 스키마입니다.
`version`	예	CodeCatalyst SARIF버전 2.1.0만 지원합니다.
`runs[]`	예	SARIF파일에는 하나 이상의 실행 배열이 포함되며, 각 실행은 분석 도구의 단일 실행을 나타냅니다.

`run` 객체

명칭	필수	설명
`tool.driver`	예	분석 도구를 설명하는 `toolComponent` 객체입니다.

명칭	필수	설명
`tool.name`	아니요	분석을 수행하는 데 사용되는 도구의 이름을 나타내는 속성입니다.
`results[]`	예	에 표시된 분석 도구의 결과 CodeCatalyst.

`toolComponent` 객체

명칭	필수	설명
`name`	예	분석 도구의 이름.
`properties.artifactScanned`	아니요	도구에서 분석한 아티팩트의 총 개수.
`rules[]`	예	규칙을 나타내는 `reportingDescriptor` 객체 배열. 분석 도구는 이러한 규칙을 기반으로 분석된 코드에서 문제를 찾습니다.

`reportingDescriptor` 객체

명칭	필수	설명
`id`	예	결과를 참조하는 데 사용되는 규칙의 고유 식별자입니다. 최대 길이: 1,024자
`name`	아니요	규칙의 표시 이름. 최대 길이: 1,024자

명칭	필수	설명
`shortDescription.text`	아니요	규칙에 대한 간략한 설명. 최대 길이: 3,000자
`fullDescription.text`	아니요	규칙에 대한 전체 설명. 최대 길이: 3,000자
`helpUri`	아니요	URI규칙의 기본 설명서의 절대 내용을 포함하도록 현지화할 수 있는 문자열입니다. 최대 길이: 3,000자
`properties.unscore`	아니요	스캔 결과에 점수를 매겼는지 여부를 나타내는 플래그입니다.
`properties.score.severity`	아니요	검색 결과의 심각도를 지정하는 고정된 문자열 세트. 최대 길이: 1,024자
`properties.cvssv3_baseSeverity`	아니요	일반 취약성 채점 시스템의 질적 심각도 등급 v3.1.
`properties.cvssv3_baseScore`	아니요	CVSSv3 기본 점수는 0.0에서 10.0 사이입니다.
`properties.cvssv2_severity`	아니요	CVSSv3 값을 사용할 수 없는 경우 v2 값을 CodeCatalyst 검색합니다. CVSS
`properties.cvssv2_score`	아니요	CVSSv2 기본 점수 범위는 0.0 - 10.0입니다.

명칭	필수	설명
properties.severity	아니요	결과의 심각도 수준을 지정하는 고정된 문자열 세트. 최대 길이: 1,024자
defaultConfiguration.level	아니요	규칙의 기본 심각도.

result 객체

명칭	필수	설명
ruleId	예	결과를 참조하는 데 사용되는 규칙의 고유 식별자입니다. 최대 길이: 1,024자
ruleIndex	예	도구 컴포넌트에 있는 관련 규칙의 인덱스. rules[]
message.text	예	결과를 설명하고 각 결과에 대한 메시지를 표시하는 메시지입니다. 최대 길이: 3,000자
rank	아니요	결과의 우선 순위 또는 중요도를 나타내는 0.0에서 100.0 (포함) 사이의 값입니다. 이 척도 값 0.0은 가장 낮은 우선 순위이고 100.0은 가장 높은 우선 순위입니다.
level	아니요	결과의 심각도. 최대 길이: 1,024자

명칭	필수	설명
`properties.unscore`	아니요	스캔 결과에 점수를 매겼는지 여부를 나타내는 플래그입니다.
`properties.score.severity`	아니요	검색 결과의 심각도를 지정하는 고정된 문자열 세트. 최대 길이: 1,024자
`properties.cvssv3_baseSeverity`	아니요	일반 취약성 채점 시스템의 질적 심각도 등급 v3.1.
`properties.cvssv3_baseScore`	아니요	CVSSv3 기본 점수는 0.0에서 10.0 사이입니다.
`properties.cvssv2_severity`	아니요	CVSSv3 값을 사용할 수 없는 경우 v2 값을 CodeCatalyst 검색합니다. CVSS
`properties.cvssv2_score`	아니요	CVSSv2 기본 점수 범위는 0.0 - 10.0입니다.
`properties.severity`	아니요	결과의 심각도 수준을 지정하는 고정된 문자열 세트. 최대 길이: 1,024자
`locations[]`	예	결과가 감지된 위치 세트. 지정된 모든 위치를 변경해야만 문제를 해결할 수 있는 경우가 아니면 한 위치만 포함해야 합니다. CodeCatalyst 위치 배열의 첫 번째 값을 사용하여 결과에 주석을 답니다. 최대 `location` 개체 수: 10개

명칭	필수	설명
`relatedLocations[]`	아니요	검색 결과에 포함된 추가 위치 참조 목록. 최대 `location` 개체 수: 50개
`fixes[]`	아니요	스캔 도구에서 제공하는 권장 사항을 나타내는 `fix` 개체 배열. CodeCatalyst `fixes`배열의 첫 번째 권장 사항을 사용합니다.

location 객체

명칭	필수	설명
`physicalLocation`	예	아티팩트와 지역을 식별합니다.
`logicalLocations[]`	아니요	아티팩트를 참조하지 않고 이름으로 설명되는 위치 세트.

physicalLocation 객체

명칭	필수	설명
`artifactLocation.uri`	예	아티팩트의 위치를 URI 나타냅니다. 일반적으로 리포지토리에 있거나 빌드 중에 생성된 파일입니다.
`fileLocation.uri`	아니요	폴백은 파일의 위치를 URI 나타냅니다. 빈 상태로 `artifactLocation.uri` 반환되는 경우에 사용됩니다.

명칭	필수	설명
region.startLine	예	영역에 있는 첫 번째 문자의 줄 번호입니다.
region.startColumn	예	영역에 있는 첫 번째 문자의 열 번호입니다.
region.endLine	예	영역의 마지막 문자의 줄 번호.
region.endColumn	예	영역 내 마지막 문자의 열 번호입니다.

logicalLocation 객체

명칭	필수	설명
fullyQualifiedName	아니요	결과 위치를 설명하는 추가 정보. 최대 길이: 1,024자

fix 객체

명칭	필수	설명
description.text	아니요	각 검색 결과에 대한 권장 사항을 표시하는 메시지입니다. 최대 길이: 3,000자
artifactChanges.[0].artifactLocation.uri	아니요	업데이트가 필요한 아티팩트의 위치를 URI 나타냅니다.

워크플로를 통한 배포

CodeCatalyst 워크플로를 사용하면 Amazon ECS AWS Lambda등과 같은 다양한 대상에 애플리케이션 및 기타 리소스를 배포할 수 있습니다.

애플리케이션을 배포하려면 어떻게 해야 합니까?

응용 프로그램 또는 리소스를 CodeCatalyst 배포하려면 먼저 워크플로를 만든 다음 워크플로우 내에 배포 작업을 지정합니다. 배포 작업은 배포할 대상, 배포할 위치, 배포 방법 (예: 파란색/녹색 체계 사용) 을 정의하는 워크플로우 구성 요소입니다. CodeCatalyst 콘솔의 시각적 편집기나 YAML 편집기를 사용하여 워크플로에 배포 작업을 추가합니다.

애플리케이션 또는 리소스를 배포하는 상위 단계는 다음과 같습니다.

응용 프로그램을 배포하려면 (상위 수준 작업)

1. CodeCatalyst 프로젝트에 배포하려는 애플리케이션의 소스 코드를 추가합니다. 자세한 내용은 프로젝트의 리포지토리에 소스 코드 저장 CodeCatalyst 단원을 참조하십시오.
2. CodeCatalyst 프로젝트에 배포하려는 대상 AWS 계정 및 선택적 Amazon Virtual Private Cloud (VPC) 를 정의하는 환경을 추가합니다. 자세한 내용은 및 에 AWS 계정 배포 VPCs 단원을 참조하십시오.
3. CodeCatalyst 프로젝트에서 워크플로를 생성합니다. 워크플로는 애플리케이션을 빌드, 테스트 및 배포하는 방법을 정의하는 곳입니다. 자세한 내용은 워크플로우 시작하기 단원을 참조하십시오.
4. 워크플로우에서 트리거, 빌드 작업, 테스트 작업 (선택 사항) 을 추가합니다. 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작, 빌드 액션 추가, 테스트 액션 추가 단원을 참조하세요.
5. 워크플로우에서 배포 작업을 추가합니다. 애플리케이션에 CodeCatalyst 제공되는 여러 배포 작업 중에서 ECS Amazon과 같은 다양한 대상에 대한 배포 작업을 선택할 수 있습니다. (빌드 작업이나 액션을 사용하여 애플리케이션을 배포할 수도 있습니다. GitHub 빌드 작업 및 GitHub 작업에 대한 자세한 내용은 을 참조하십시오액션 배포를 위한 대안.
6. 워크플로우는 수동으로 시작하거나 트리거를 통해 자동으로 시작합니다. 워크플로는 빌드, 테스트 및 배포 작업을 순서대로 실행하여 애플리케이션과 리소스를 대상에 배포합니다. 자세한 내용은 워크플로우 수동 실행 시작 단원을 참조하십시오.

배포 작업 목록

다음과 같은 배포 작업을 사용할 수 있습니다.

- AWS CloudFormation 스택 배포 - 이 작업을 수행하면 사용자가 제공한 AWS CloudFormationAWS Serverless Application Model 템플릿 또는 템플릿을 AWS 기반으로 CloudFormation 스택이 생성됩니다. 자세한 내용은 스택 배포 AWS CloudFormation 단원을 참조하십시오.
- Amazon에 배포 ECS — 이 작업은 제공하는 작업 정의 파일을 등록합니다. 자세한 내용은 워크플로를 ECS 사용하여 Amazon에 배포하기 단원을 참조하십시오.
- 쿠버네티스 클러스터에 배포 — 이 작업은 애플리케이션을 Amazon Elastic Kubernetes Service 클러스터에 배포합니다. 자세한 내용은 워크플로를 EKS 사용하여 Amazon에 배포하기 단원을 참조하십시오.
- AWS CDK 배포 — 이 작업은 앱을 에 배포합니다.AWS CDK AWS자세한 내용은 워크플로를 통한 AWS CDK 앱 배포 단원을 참조하십시오.

> **Note**
>
> 리소스를 배포할 수 있는 다른 CodeCatalyst 작업도 있지만 해당 배포 정보가 환경 페이지에 표시되지 않으므로 배포 작업으로 간주되지 않습니다. 환경 페이지 및 배포 보기에 대한 자세한 내용은 및 을 참조하십시오및 에 AWS 계정 배포 VPCs. 배포 정보 보기

배포 작업의 이점

워크플로우 내에서 배포 작업을 사용하면 다음과 같은 이점이 있습니다.

- 배포 기록 - 배포 기록을 보면 배포된 소프트웨어의 변경 사항을 관리하고 전달하는 데 도움이 됩니다.
- 추적성 — CodeCatalyst 콘솔을 통해 배포 상태를 추적하고 각 애플리케이션 수정 버전이 언제 어디서 배포되었는지 확인할 수 있습니다.
- 롤백 — 오류가 있는 경우 배포를 자동으로 롤백합니다. 배포 롤백을 활성화하도록 경보를 구성할 수도 있습니다.
- 모니터링 — 워크플로의 다양한 단계를 통해 배포가 진행되는 과정을 지켜보세요.
- 다른 CodeCatalyst 기능과의 통합 — 하나의 애플리케이션에서 소스 코드를 저장한 다음 빌드, 테스트 및 배포할 수 있습니다.

액션 배포를 위한 대안

배포 작업은 이전 섹션에 설명된 이점을 제공하므로 권장되기는 하지만 반드시 사용할 필요는 없습니다. 대신 다음 CodeCatalyst 작업을 사용할 수 있습니다.

- 빌드 액션.

  일반적으로 빌드 작업은 해당하는 배포 작업이 없는 대상에 배포하거나 배포 프로시저를 보다 세밀하게 제어하려는 경우에 사용합니다. 빌드 작업을 사용하여 리소스를 배포하는 방법에 대한 자세한 내용은 을 참조하십시오워크플로를 사용한 구축.

- GitHub 액션.

  CodeCatalyst 워크플로 내에서 GitHub 작업을 사용하여 (작업 대신) 애플리케이션과 리소스를 배포할 수 있습니다. CodeCatalyst CodeCatalyst 워크플로 내에서 GitHub 액션을 사용하는 방법에 대한 자세한 내용은 을 참조하십시오. 액션과 GitHub 통합

CodeCatalyst 워크플로를 사용하지 않으려는 경우 다음 AWS 서비스를 사용하여 애플리케이션을 배포할 수도 있습니다.

- AWS CodeDeploy — CodeDeploy무엇입니까를 참조하십시오.
- AWS CodeBuild 그리고 AWS CodePipeline — 이게 AWS CodeBuild뭔지 보세요. 그리고 이게 AWS CodePipeline뭐죠?
- AWS CloudFormation — 이게 AWS CloudFormation뭔지 보세요.

복잡한 엔터프라이즈 CodePipeline 배포에는 CodeDeploy CodeBuild,, 및 CloudFormation 서비스를 사용하십시오.

주제

- 워크플로를 ECS 사용하여 Amazon에 배포하기
- 워크플로를 EKS 사용하여 Amazon에 배포하기
- 스택 배포 AWS CloudFormation
- 워크플로를 통한 AWS CDK 앱 배포
- 워크플로를 사용하여 AWS CDK 앱 부트스트래핑하기
- 워크플로를 사용하여 Amazon S3에 파일 게시
- 및 에 AWS 계정 배포 VPCs

- 워크플로 URL 다이어그램에 앱 표시
- 배포 대상 제거
- 커밋별 배포 상태 추적
- 배포 로그 보기
- 배포 정보 보기

워크플로를 ECS 사용하여 Amazon에 배포하기

이 단원에서는 워크플로를 사용하여 컨테이너화된 애플리케이션을 Amazon Elastic Container Service 클러스터에 배포하는 CodeCatalyst 방법을 설명합니다. 이 작업을 수행하려면 워크플로에 Amazon에 배포 ECS 작업을 추가해야 합니다. 이 작업을 수행하면 제공하는 작업 정의 파일이 등록됩니다. 등록 시 Amazon 클러스터에서 실행되는 Amazon ECS 서비스에 의해 작업 정의가 인스턴스화됩니다. ECS "작업 정의를 인스턴스화"하는 것은 Amazon에 애플리케이션을 배포하는 것과 같습니다. ECS

이 작업을 사용하려면 Amazon ECS 클러스터, 서비스 및 작업 정의 파일이 준비되어 있어야 합니다.

Amazon에 대한 자세한 내용은 Amazon ECS Elastic 컨테이너 서비스 개발자 안내서를 참조하십시오.

> **Tip**
>
> Amazon에 배포 ECS 작업을 사용하는 방법을 보여주는 자습서는 을 참조하십시오자습서: Amazon에 애플리케이션 배포 ECS.

> **Tip**
>
> Amazon에 배포 ECS 작업의 실제 예제를 보려면 AWS Fargate블루프린트가 API포함된 Node.js AWS Fargate 또는 Java를 사용하여 프로젝트를 API 생성하십시오. 자세한 내용은 블루프린트로 프로젝트 생성 단원을 참조하십시오.

주제

- 자습서: Amazon에 애플리케이션 배포 ECS
- '아마존에 ECS 배포' 작업 추가
- '아마존에 ECS 배포' 변수
- '아마존에 ECS 배포' 조치 YAML

자습서: Amazon에 애플리케이션 배포 ECS

이 자습서에서는 워크플로, Amazon 및 기타 몇 가지 AWS 서비스를 사용하여 Amazon Elastic Container Service (AmazonECS) 에 서버리스 애플리케이션을 배포하는 방법을 알아봅니다. ECS 배포된 애플리케이션은 Apache 웹 서버 Docker 이미지를 기반으로 구축된 간단한 Hello World 웹 사이트입니다. 이 자습서에서는 클러스터 설정과 같은 필수 준비 작업을 안내한 다음 애플리케이션을 빌드하고 배포하기 위한 워크플로를 만드는 방법을 설명합니다.

> **Tip**
>
> 이 튜토리얼을 따라가는 대신 완전한 Amazon ECS 설정을 수행하는 블루프린트를 사용할 수 있습니다. API블루프린트와 함께 Node.js 또는 AWS Fargate블루프린트와 API 함께 AWS Fargate Java를 사용해야 합니다. 자세한 내용은 블루프린트로 프로젝트 생성 단원을 참조하십시오.

주제

사전 조건

시작하기 전:

- 연결된 AWS 계정이 있는 CodeCatalyst 공간이 필요합니다. 자세한 내용은 스페이스 만들기 단원을 참조하십시오.

- 스페이스에 다음과 같은 빈 프로젝트가 필요합니다.

```
codecatalyst-ecs-project
```

  처음부터 시작 옵션을 사용하여 이 프로젝트를 만들 수 있습니다.

  자세한 내용은 Amazon에서 빈 프로젝트 생성 CodeCatalyst 단원을 참조하십시오.
- 프로젝트에는 다음과 같은 CodeCatalyst 환경이 필요합니다.

```
codecatalyst-ecs-environment
```

  이 환경을 다음과 같이 구성하십시오.
  - 원하는 유형 (예: 비프로덕션) 을 선택합니다.
  - AWS 계정을 여기에 연결하십시오.
  - 기본 IAM 역할의 경우 원하는 역할을 선택합니다. 나중에 다른 역할을 지정하게 됩니다.

  자세한 내용은 및 에 AWS 계정 배포 VPCs 단원을 참조하십시오.

1단계: AWS 사용자 설정 및 AWS CloudShell

이 자습서의 첫 번째 단계는 에서 AWS IAM Identity Center사용자를 생성하고 이 사용자로 AWS CloudShell 인스턴스를 시작하는 것입니다. 이 자습서를 진행하는 동안 CloudShell 는 개발용 컴퓨터이며 AWS 리소스와 서비스를 구성하는 곳입니다. 자습서를 완료한 후 이 사용자를 삭제하십시오.

> **Note**
>
> 이 자습서에는 루트 사용자를 사용하지 마십시오. 별도의 사용자를 만들어야 합니다. 그렇지 않으면 나중에 AWS Command Line Interface (CLI) 에서 작업을 수행할 때 문제가 발생할 수 있습니다.

IAMIdentity Center 사용자에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서 및AWS CloudShell 사용 설명서를 참조하십시오. CloudShell

IAM아이덴티티 센터 사용자를 만들려면

1. 에 AWS Management Console 로그인하고 에서 AWS IAM Identity Center 콘솔을 엽니다 https://console.aws.amazon.com/singlesignon/.

> **Note**
>
> CodeCatalyst스페이스에 AWS 계정 연결된 계정을 사용하여 로그인해야 합니다. 스페이스로 이동하고 계정 탭을 선택하면 어떤 계정이 연결되어 있는지 확인할 수 있습니다. AWS 자세한 내용은 스페이스 만들기 단원을 참조하십시오.

2. 탐색 창에서 Users와 Add user를 차례대로 선택합니다.
3. 사용자 이름에 다음을 입력합니다.

```
CodeCatalystECSUser
```

4. 비밀번호에서 이 사용자와 공유할 수 있는 일회용 비밀번호 생성을 선택합니다.
5. 이메일 주소 및 이메일 주소 확인에 IAM Identity Center에 아직 없는 이메일 주소를 입력합니다.
6. 이름과 성에 다음을 입력합니다.

```
CodeCatalystECSUser
```

7. 표시 이름에 자동으로 생성된 이름을 유지합니다.

```
CodeCatalystECSUser CodeCatalystECSUser
```

8. Next(다음)를 선택합니다.
9. 그룹에 사용자 추가 페이지에서 다음을 선택합니다.
10. 사용자 검토 및 추가 페이지에서 정보를 검토하고 사용자 추가를 선택합니다.

    일회용 암호 대화 상자가 나타납니다.
11. 복사를 선택한 다음 AWS 액세스 포털 URL 및 일회용 비밀번호를 포함한 로그인 정보를 붙여넣습니다.
12. 닫기를 선택하세요.

권한 집합을 생성하려면

나중에 이 권한 세트를 할당할 `CodeCatalystECSUser` 것입니다.

1. 탐색 창에서 권한 세트를 선택한 다음 권한 세트 생성을 선택합니다.
2. 사전 정의된 권한 집합을 선택한 다음 선택합니다 AdministratorAccess. 이 정책은 모든 AWS 서비스사람에게 전체 권한을 제공합니다.

3. Next(다음)를 선택합니다.
4. 권한 세트 이름에 다음을 입력합니다.

```
CodeCatalystECSPermissionSet
```

5. Next(다음)를 선택합니다.
6. 검토 및 생성 페이지에서 정보를 검토하고 생성을 선택합니다.

권한 세트를 할당하려면 CodeCatalyst ECSUser

1. 탐색 창에서 선택한 다음 현재 로그인되어 AWS 계정 있는 권한 옆의 확인란을 선택합니다. AWS 계정
2. 사용자 또는 그룹 할당을 선택합니다.
3. 사용자 탭을 선택합니다.
4. `CodeCatalystECSUser` 옆의 확인란을 선택합니다.
5. Next(다음)를 선택합니다.
6. `CodeCatalystECSPermissionSet` 옆의 확인란을 선택합니다.
7. Next(다음)를 선택합니다.
8. 정보를 검토하고 제출을 선택합니다.

   이제 두 개를 하나로 `CodeCatalystECSUser` AWS 계정묶어 `CodeCatalystECSPermissionSet` 자신에게 할당했습니다.

다음과 같이 로그아웃하고 다시 로그인하려면 CodeCatalyst ECSUser

1. 로그아웃하기 전에 AWS 액세스 URL 포털과 사용자 이름, 일회용 비밀번호가 있는지 확인하세요. `CodeCatalystECSUser` 이전에 이 정보를 텍스트 편집기에 복사했어야 합니다.

   > **Note**
   >
   > 이 정보가 없는 경우 IAM Identity Center의 `CodeCatalystECSUser` 세부 정보 페이지로 이동하여 암호 재설정, 일회용 암호 생성 [...] 을 선택합니다. 그리고 암호를 다시 재설정하여 화면에 정보를 표시하십시오.

2. AWS로그아웃하세요.
3. AWS 액세스 URL 포털을 브라우저의 주소 표시줄에 붙여넣습니다.

4. 사용자 이름과 일회용 비밀번호로 로그인합니다. `CodeCatalystECSUser`
5. 새 비밀번호에 비밀번호를 입력하고 새 비밀번호 설정을 선택합니다.

   화면에 AWS 계정 상자가 나타납니다.
6. 선택한 AWS 계정다음 `CodeCatalystECSUser` 사용자에게 할당한 AWS 계정 이름과 권한 집합을 선택합니다.
7. `CodeCatalystECSPermissionSet` 옆에 있는 관리 콘솔을 선택합니다.

   가 AWS Management Console 나타납니다. 이제 적절한 `CodeCatalystECSUser` 권한으로 로그인했습니다.

AWS CloudShell 인스턴스를 시작하려면

1. `CodeCatalystECSUser`마찬가지로 상단 내비게이션 바에서 AWS 아이콘
   ()
   을 선택합니다.

   의 메인 페이지가 AWS Management Console 나타납니다.
2. 상단 내비게이션 바에서 AWS CloudShell 아이콘
   ()
   을 선택합니다.

   CloudShell 열립니다. CloudShell 환경이 만들어지는 동안 기다려 주세요.

> **Note**
>
> CloudShell 아이콘이 보이지 않는 경우 에서 지원하는 지역에 있는지 확인하세요 CloudShell. 이 자습서에서는 사용자가 미국 서부(오레곤) 리전에 있다고 가정합니다.

가 AWS CLI 설치되었는지 확인하려면

1. CloudShell 터미널에서 다음을 입력합니다.

```
aws --version
```

2. 버전이 나타나는지 확인합니다.

AWS CLI 는 이미 현재 사용자에 맞게 구성되어 있으므로 일반적으로 그렇듯이 AWS CLI 키와 자격 증명을 구성할 필요가 없습니다. `CodeCatalystECSUser`

2단계: Amazon에 플레이스홀더 애플리케이션 배포 ECS

이 섹션에서는 ECS Amazon에 플레이스홀더 애플리케이션을 수동으로 배포합니다. 이 플레이스홀더 애플리케이션은 워크플로에서 배포한 Hello World 애플리케이션으로 대체됩니다. 자리 표시자 응용 프로그램은 Apache 웹 서버입니다.

Amazon에 대한 자세한 내용은 Amazon ECS Elastic 컨테이너 서비스 개발자 안내서를 참조하십시오.

다음 일련의 절차를 완료하여 플레이스홀더 애플리케이션을 배포하십시오.

작업 실행 역할을 만들려면

이 역할은 ECS Amazon과 사용자를 대신하여 API 전화를 걸 수 있는 AWS Fargate (Fargate) 권한을 부여합니다.

1. 신뢰 정책 생성:

   a. 에서 AWS CloudShell다음 명령을 입력합니다.

   ```
   cat > codecatalyst-ecs-trust-policy.json
   ```

   CloudShell 터미널에 깜박이는 프롬프트가 나타납니다.

   b. 프롬프트에 다음 코드를 입력합니다.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
           "Service": "ecs-tasks.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

c. 마지막 중괄호 (}) 뒤에 커서를 놓습니다.

d. 버튼을 누른 **Enter** 다음 **Ctrl+d** 파일을 저장하고 cat을 종료합니다.

2. 작업 실행 역할 생성:

```
aws iam create-role \
      --role-name codecatalyst-ecs-task-execution-role \
      --assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
```

3. AWS 관리형 AmazonECSTaskExecutionRolePolicy 정책을 역할에 연결합니다.

```
aws iam attach-role-policy \
      --role-name codecatalyst-ecs-task-execution-role \
      --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy
```

4. 역할 세부 정보 표시:

```
aws iam get-role \
      --role-name codecatalyst-ecs-task-execution-role
```

5. 역할의 "Arn": 값 (예:) 을 기록해 둡니다arn:aws:iam::111122223333:role/codecatalyst-ecs-task-execution-role. 나중에 이 Amazon 리소스 이름 (ARN) 이 필요합니다.

Amazon ECS 클러스터를 만들려면

이 클러스터에는 Apache 플레이스홀더 애플리케이션이 포함되며 나중에는 Hello World 애플리케이션이 포함됩니다.

1. 에서와 CodeCatalystECSUser AWS CloudShell같이 빈 클러스터를 생성합니다.

```
aws ecs create-cluster --cluster-name codecatalyst-ecs-cluster
```

2. (선택 사항) 클러스터가 성공적으로 생성되었는지 확인합니다.

```
aws ecs list-clusters
```

codecatalyst-ecs-cluster클러스터가 목록에 나타나야 하는데, 이는 성공적으로 생성되었음을 나타냅니다. ARN

작업 정의 파일을 만들려면

작업 정의 파일은 가져온 Apache 2.4 웹 서버 Docker 이미지 (`httpd:2.4`) 를 실행하도록 지정합니다. DockerHub

1. 에서와 `CodeCatalystECSUser` AWS CloudShell같이 작업 정의 파일을 생성합니다.

```
cat > taskdef.json
```

2. 프롬프트에 다음 코드를 붙여넣습니다.

```
{
    "executionRoleArn": "arn:aws:iam::111122223333:role/codecatalyst-ecs-task-execution-role",
    "containerDefinitions": [
        {
            "name": "codecatalyst-ecs-container",
            "image": "httpd:2.4",
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "256",
    "family": "codecatalyst-ecs-task-def",
    "memory": "512",
    "networkMode": "awsvpc"
}
```

위 코드에서 다음을 대체하십시오.*arn:aws:iam::111122223333:role/codecatalyst-ecs-task-execution-role*

에서 ARN 작업 실행 역할을 만들려면 언급한 작업 실행 역할로 바꾸십시오.

3. 마지막 중괄호 (}) 뒤에 커서를 놓습니다.

4. 버튼을 누른 **Enter** 다음 **Ctrl+d** 파일을 저장하고 cat을 종료합니다.

Amazon에 작업 정의 파일을 등록하려면 ECS

1. 에서와 CodeCatalystECSUser AWS CloudShell같이 작업 정의를 등록하십시오.

```
aws ecs register-task-definition \
    --cli-input-json file://taskdef.json
```

2. (선택 사항) 작업 정의가 등록되었는지 확인합니다.

```
aws ecs list-task-definitions
```

codecatalyst-ecs-task-def작업 정의가 목록에 나타나야 합니다.

Amazon ECS 서비스를 만들려면

Amazon ECS 서비스는 Apache 플레이스홀더 애플리케이션의 작업 (및 관련 Docker 컨테이너) 을 실행하고 나중에는 Hello World 애플리케이션을 실행합니다.

1. 마찬가지로CodeCatalystECSUser, 아직 전환하지 않았다면 Amazon Elastic Container Service 콘솔로 전환하십시오.
2. 앞서 만든 클러스터를 선택하세요codecatalyst-ecs-cluster.
3. [서비스] 탭에서 [Create] 를 선택합니다.
4. 만들기 페이지에서 다음을 수행합니다.
   a. 다음에 나열된 설정을 제외한 모든 기본 설정을 유지합니다.
   b. 시작 유형에서 선택합니다 FARGATE.
   c. 작업 정의의 패밀리 드롭다운 목록에서 다음을 선택합니다.

      codecatalyst-ecs-task-def

   d. 서비스 이름에 다음을 입력합니다.

      codecatalyst-ecs-service

   e. 원하는 작업에 다음을 입력합니다.

```
3
```

이 자습서에서는 각 작업이 단일 Docker 컨테이너를 시작합니다.

f. 네트워킹 섹션을 확장하세요.

g. 원하는 VPC항목을 선택하십시오VPC.

h. 서브넷의 경우 원하는 서브넷을 선택합니다.

> **Note**
>
> 서브넷을 하나만 지정하십시오. 이 자습서에 필요한 것은 여기까지입니다.

> **Note**
>
> AND 서브넷이 없는 경우 해당 VPC 서브넷을 생성하십시오. Amazon VPC 사용 설명서의 서브넷 생성 및 서브넷 생성을 참조하십시오. VPC VPC

i. 보안 그룹의 경우 새 보안 그룹 생성을 선택하고 다음을 수행하십시오.

   i. 보안 그룹 이름에 다음을 입력합니다.

   ```
   codecatalyst-ecs-security-group
   ```

   ii. 보안 그룹 설명에 다음을 입력합니다.

   ```
   CodeCatalyst ECS security group
   ```

   iii. 규칙 추가를 선택합니다. 유형에서 선택하고 HTTP, 소스에 대해서는 Anywhere를 선택합니다.

j. 하단에서 [만들기] 를 선택합니다.

k. 서비스가 생성되는 동안 잠시 기다려 주세요. 몇 분 정도 소요될 수 있습니다.

5. 태스크 탭을 선택한 다음 새로 고침 버튼을 선택합니다. 세 작업 모두의 [마지막 상태] 열이 [실행 중] 으로 설정되어 있는지 확인하십시오.

(선택 사항) Apache 플레이스홀더 애플리케이션이 실행 중인지 확인하려면

1. 태스크 탭에서 세 가지 작업 중 하나를 선택합니다.
2. 퍼블릭 IP 필드에서 오픈 주소를 선택합니다.

   `It Works!`페이지가 나타납니다. 이는 Amazon ECS 서비스가 Apache 이미지를 사용하여 Docker 컨테이너를 시작하는 작업을 성공적으로 시작했음을 나타냅니다.

   자습서의 이 시점에서는 Amazon ECS 클러스터, 서비스, 작업 정의와 Apache 플레이스홀더 애플리케이션을 수동으로 배포했습니다. 이 모든 항목이 준비되었으므로 이제 Apache 자리 표시자 애플리케이션을 자습서의 Hello World 애플리케이션으로 대체할 워크플로를 만들 준비가 되었습니다.

3단계: Amazon ECR 이미지 리포지토리 생성

이 섹션에서는 Amazon Elastic 컨테이너 레지스트리 (AmazonECR) 에 프라이빗 이미지 리포지토리를 생성합니다. 이 리포지토리에는 이전에 배포한 Apache 플레이스홀더 이미지를 대체할 자습서의 Docker 이미지가 저장됩니다.

Amazon에 대한 자세한 내용은 Amazon ECR Elastic 컨테이너 레지스트리 사용 설명서를 참조하십시오.

Amazon에서 이미지 리포지토리를 만들려면 ECR

1. 에서와 ECR 같이 `CodeCatalystECSUser` Amazon에 AWS CloudShell빈 리포지토리를 생성하십시오.

   ```
   aws ecr create-repository --repository-name codecatalyst-ecs-image-repo
   ```

2. Amazon ECR 리포지토리의 세부 정보를 표시합니다.

   ```
   aws ecr describe-repositories \
         --repository-names codecatalyst-ecs-image-repo
   ```

3. `“repositoryUri”:`값을 기록해 둡니다 (예:)`111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo`.

   나중에 워크플로에 리포지토리를 추가할 때 필요합니다.

4단계: AWS 역할 생성

이 섹션에서는 CodeCatalyst 워크플로가 작동하는 데 필요한 AWS IAM 역할을 생성합니다. 이러한 역할은 다음과 같습니다.

- 빌드 역할 - (워크플로의) CodeCatalyst 빌드 작업에 AWS 계정에 액세스하고 Amazon ECR 및 Amazon에 글을 쓸 수 있는 권한을 EC2 부여합니다.
- 배포 역할 - 사용자 AWS 계정ECS, Amazon 및 기타 몇 가지 AWS 서비스에 액세스할 수 있는 CodeCatalyst Deploy to ECS Action (워크플로의) 권한을 부여합니다.

IAM역할에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM 역할을 참조하십시오.

> **Note**
>
> 시간을 절약하기 위해 앞서 나열한 두 역할 대신 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 역할이라는 단일 역할을 만들 수 있습니다. 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기 단원을 참조하십시오. `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*역할에는 보안 위험을 초래할 수 있는 매우 광범위한 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다. 이 자습서에서는 앞서 나열한 두 개의 역할을 생성한다고 가정합니다.

빌드 및 배포 역할을 만들려면 AWS Management Console 또는 를 AWS CLI사용할 수 있습니다.

AWS Management Console

빌드 및 배포 역할을 만들려면 다음 일련의 절차를 완료하세요.

빌드 역할을 만들려면

1. 다음과 같이 역할에 대한 정책을 생성합니다.
   a. 로그인 AWS.
   b. 에서 IAM 콘솔을 엽니다 https://console.aws.amazon.com/iam/.
   c. 탐색 창에서 정책을 선택합니다.
   d. 정책 생성을 선택합니다.

e. JSON탭을 선택합니다.

f. 기존 코드를 삭제합니다.

g. 다음 코드를 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고, 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

h. 다음: 태그를 선택합니다.

i. 다음: 검토를 선택합니다.

j. 이름에 다음을 입력합니다.

```
codecatalyst-ecs-build-policy
```

k. 정책 생성을 선택합니다.

이제 권한 정책을 생성했습니다.

2. 다음과 같이 빌드 역할을 생성합니다.

a. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.

b. 사용자 지정 신뢰 정책을 선택합니다.

c. 기존 사용자 지정 신뢰 정책을 삭제합니다.

d. 다음 사용자 지정 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. Next(다음)를 선택합니다.

f. 권한 정책에서 해당 확인란을 검색하여 선택합니다. `codecatalyst-ecs-build-policy`

g. Next(다음)를 선택합니다.

h. 역할 이름에 다음을 입력합니다.

```
codecatalyst-ecs-build-role
```

i. 역할 설명에 다음을 입력합니다.

```
CodeCatalyst ECS build role
```

j. 역할 생성을 선택합니다.

이제 권한 정책과 신뢰 정책이 포함된 빌드 역할을 만들었습니다.

3. 다음과 같이 빌드 역할을 ARN 확보하세요.

a. 탐색 창에서 역할을 선택합니다.

b. 검색 상자에 방금 생성한 역할의 이름을 입력합니다 (`codecatalyst-ecs-build-role`).

c. 목록에서 역할을 선택합니다.

역할의 요약 페이지가 나타납니다.

d. 상단에서 ARN값을 복사합니다. 나중에 필요합니다.

배포 역할을 만들려면

1. 다음과 같이 역할에 대한 정책을 생성합니다.

a. 로그인 AWS.

b. 에서 IAM 콘솔을 엽니다 https://console.aws.amazon.com/iam/.

c. 탐색 창에서 정책을 선택합니다.

d. 정책 생성(Create Policy)을 선택합니다.

e. JSON탭을 선택합니다.

f. 기존 코드를 삭제합니다.

g. 다음 코드를 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action":[
      "ecs:DescribeServices",
      "ecs:CreateTaskSet",
      "ecs:DeleteTaskSet",
      "ecs:ListClusters",
      "ecs:RegisterTaskDefinition",
      "ecs:UpdateServicePrimaryTaskSet",
      "ecs:UpdateService",
      "elasticloadbalancing:DescribeTargetGroups",
      "elasticloadbalancing:DescribeListeners",
      "elasticloadbalancing:ModifyListener",
      "elasticloadbalancing:DescribeRules",
      "elasticloadbalancing:ModifyRule",
      "lambda:InvokeFunction",
      "lambda:ListFunctions",
      "cloudwatch:DescribeAlarms",
      "sns:Publish",
```

```
      "sns:ListTopics",
      "s3:GetObject",
      "s3:GetObjectVersion",
      "codedeploy:CreateApplication",
      "codedeploy:CreateDeployment",
      "codedeploy:CreateDeploymentGroup",
      "codedeploy:GetApplication",
      "codedeploy:GetDeployment",
      "codedeploy:GetDeploymentGroup",
      "codedeploy:ListApplications",
      "codedeploy:ListDeploymentGroups",
      "codedeploy:ListDeployments",
      "codedeploy:StopDeployment",
      "codedeploy:GetDeploymentTarget",
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
   ],
   "Resource":"*",
   "Effect":"Allow"
},{"Action":[
      "iam:PassRole"
   ],
   "Effect":"Allow",
   "Resource":"*",
   "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
   }
}]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하십시오. 그런 다음 사용할 수 있게 되면 리소스 이름을 사용하여 정책의 범위를 좁힐 수 있습니다.
>
> ```
> "Resource": "*"
> ```

h. 다음: 태그를 선택합니다.

i. 다음: 검토를 선택합니다.

j. 이름에 다음을 입력합니다.

```
codecatalyst-ecs-deploy-policy
```

k. 정책 생성을 선택합니다.

이제 권한 정책을 생성했습니다.

2. 다음과 같이 배포 역할을 생성합니다.

a. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.

b. 사용자 지정 신뢰 정책을 선택합니다.

c. 기존 사용자 지정 신뢰 정책을 삭제합니다.

d. 다음 사용자 지정 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
```

```
    ]
}
```

e. Next(다음)를 선택합니다.

f. 권한 정책에서 해당 확인란을 검색하여 선택합니다. `codecatalyst-ecs-deploy-policy`

g. Next(다음)를 선택합니다.

h. 역할 이름에 다음을 입력합니다.

```
codecatalyst-ecs-deploy-role
```

i. 역할 설명에 다음을 입력합니다.

```
CodeCatalyst ECS deploy role
```

j. 역할 생성을 선택합니다.

이제 신뢰 정책을 사용하여 배포 역할을 생성했습니다.

3. 다음과 같이 배포 역할을 ARN 확보하십시오.

a. 탐색 창에서 역할을 선택합니다.

b. 검색 상자에 방금 생성한 역할의 이름을 입력합니다 (`codecatalyst-ecs-deploy-role`).

c. 목록에서 역할을 선택합니다.

역할의 요약 페이지가 나타납니다.

d. 상단에서 ARN값을 복사합니다. 나중에 필요합니다.

AWS CLI

빌드 및 배포 역할을 만들려면 다음 일련의 절차를 완료하세요.

두 역할 모두에 대한 신뢰 정책을 만들려면

에서와 `CodeCatalystECSUser` AWS CloudShell같이 신뢰 정책 파일을 생성하십시오.

1. 파일 생성:

```
cat > codecatalyst-ecs-trust-policy.json
```

2. 터미널 프롬프트에 다음 코드를 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

3. 마지막 중괄호 (}) 뒤에 커서를 놓습니다.
4. 버튼을 누른 **Enter** 다음 **Ctrl+d** 파일을 저장하고 cat을 종료합니다.

빌드 정책 및 빌드 역할을 만들려면

1. 빌드 정책 만들기:

   a. 에서 AWS CloudShell빌드 정책 파일을 생성하는 것처럼CodeCatalystECSUser:

   ```
   cat > codecatalyst-ecs-build-policy.json
   ```

   b. 프롬프트에 다음 코드를 입력합니다.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "ecr:*",
   ```

```
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

c. 마지막 중괄호 (}) 뒤에 커서를 놓습니다.

d. 버튼을 누른 **Enter** 다음 **Ctrl+d** 파일을 저장하고 cat을 종료합니다.

2. 빌드 정책을 AWS다음에 추가합니다.

```
aws iam create-policy \
    --policy-name codecatalyst-ecs-build-policy \
    --policy-document file://codecatalyst-ecs-build-policy.json
```

3. 명령 출력에서 `"arn":` 값 (예:) 을 기록해 둡니다`arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy`. ARN나중에 이 정보가 필요합니다.

4. 빌드 역할을 만들고 여기에 신뢰 정책을 연결합니다.

```
aws iam create-role \
      --role-name codecatalyst-ecs-build-role \
      --assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
```

5. 빌드 정책을 빌드 역할에 연결:

```
aws iam attach-role-policy \
      --role-name codecatalyst-ecs-build-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy
```

위치 *arn:aws:iam::111122223333:policy/codecatalyst-ecs-build-policy* 앞에서 언급한 빌드 정책으로 대체됩니다. ARN

6. 빌드 역할의 세부 정보 표시:

```
aws iam get-role \
      --role-name codecatalyst-ecs-build-role
```

7. 역할의 `"Arn":` 값 (예:) 을 기록해 둡니다`arn:aws:iam::111122223333:role/codecatalyst-ecs-build-role`. ARN나중에 필요합니다.

배포 정책 및 배포 역할을 만들려면

1. 배포 정책 생성:

   a. 에서 AWS CloudShell배포 정책 파일을 생성합니다.

   ```
   cat > codecatalyst-ecs-deploy-policy.json
   ```

   b. 프롬프트에 다음 코드를 입력합니다.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [{
       "Action":[
         "ecs:DescribeServices",
         "ecs:CreateTaskSet",
         "ecs:DeleteTaskSet",
         "ecs:ListClusters",
         "ecs:RegisterTaskDefinition",
         "ecs:UpdateServicePrimaryTaskSet",
         "ecs:UpdateService",
         "elasticloadbalancing:DescribeTargetGroups",
         "elasticloadbalancing:DescribeListeners",
         "elasticloadbalancing:ModifyListener",
         "elasticloadbalancing:DescribeRules",
         "elasticloadbalancing:ModifyRule",
         "lambda:InvokeFunction",
         "lambda:ListFunctions",
         "cloudwatch:DescribeAlarms",
         "sns:Publish",
         "sns:ListTopics",
         "s3:GetObject",
         "s3:GetObjectVersion",
         "codedeploy:CreateApplication",
         "codedeploy:CreateDeployment",
         "codedeploy:CreateDeploymentGroup",
         "codedeploy:GetApplication",
         "codedeploy:GetDeployment",
         "codedeploy:GetDeploymentGroup",
         "codedeploy:ListApplications",
         "codedeploy:ListDeploymentGroups",
         "codedeploy:ListDeployments",
         "codedeploy:StopDeployment",
   ```

```
      "codedeploy:GetDeploymentTarget",
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
   ],
   "Resource":"*",
   "Effect":"Allow"
},{"Action":[
      "iam:PassRole"
   ],
   "Effect":"Allow",
   "Resource":"*",
   "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
   }
}]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

c. 마지막 중괄호 () } 뒤에 커서를 놓습니다.

d. 버튼을 누른 **Enter** 다음 **Ctrl+d** 파일을 저장하고 cat을 종료합니다.

2. 배포 정책을 AWS다음에 추가합니다.

```
aws iam create-policy \
    --policy-name codecatalyst-ecs-deploy-policy \
    --policy-document file://codecatalyst-ecs-deploy-policy.json
```

3. 명령 출력에서 배포 정책의 "arn": 값 (예:) 을 기록해 둡니다arn:aws:iam::111122223333:policy/codecatalyst-ecs-deploy-policy. ARN나중에 이 정보가 필요합니다.
4. 배포 역할을 만들고 여기에 신뢰 정책을 연결합니다.

```
aws iam create-role \
      --role-name codecatalyst-ecs-deploy-role \
      --assume-role-policy-document file://codecatalyst-ecs-trust-policy.json
```

5. 배포 정책을 배포 역할에 연결합니다. 여기서 *arn:aws:iam::111122223333:policy/codecatalyst-ecs-deploy-policy* 앞에서 언급한 배포 정책으로 대체됩니다. ARN

```
aws iam attach-role-policy \
      --role-name codecatalyst-ecs-deploy-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-ecs-deploy-
policy
```

6. 배포 역할의 세부 정보를 표시합니다.

```
aws iam get-role \
      --role-name codecatalyst-ecs-deploy-role
```

7. 역할의 "Arn": 값 (예:) 을 기록해 둡니다arn:aws:iam::111122223333:role/codecatalyst-ecs-deploy-role. ARN나중에 필요합니다.

5단계: AWS 역할 추가 CodeCatalyst

이 단계에서는 스페이스의 CodeCatalyst 계정 연결에 빌드 역할 (codecatalyst-ecs-build-rolecodecatalyst-ecs-deploy-role) 과 배포 역할 () 을 추가합니다.

계정 연결에 빌드 및 배포 역할을 추가하려면

1. CodeCatalyst에서 스페이스로 이동합니다.
2. AWS 계정을 선택하세요. 계정 연결 목록이 나타납니다.
3. 빌드 및 배포 역할을 생성한 AWS 계정을 나타내는 계정 연결을 선택합니다.

4. 관리 콘솔에서 역할 AWS 관리를 선택합니다.

   Amazon CodeCatalyst 스페이스에 IAM 역할 추가 페이지가 나타납니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.

5. 에서 생성한 기존 역할 추가를 선택합니다IAM.

   드롭다운 목록이 나타납니다. 목록에는 `codecatalyst-runner.amazonaws.com` 및 `codecatalyst.amazonaws.com` 서비스 주체를 포함하는 신뢰 정책이 적용된 모든 IAM 역할이 표시됩니다.

6. 드롭다운 목록에서 역할을 선택하고 역할 `codecatalyst-ecs-build-role` 추가를 선택합니다.

   > **Note**
   >
   > 표시되는 `The security token included in the request is invalid` 경우 적절한 권한이 없기 때문일 수 있습니다. 이 문제를 해결하려면 CodeCatalyst 스페이스를 만들 때 사용한 AWS 계정으로 로그아웃하고 다시 로그인하세요. AWS

7. IAM역할 추가를 선택하고 에서 IAM 생성한 기존 역할 추가를 선택한 다음 드롭다운 목록에서 선택합니다`codecatalyst-ecs-deploy-role`. [Add role]을 선택합니다.

   이제 스페이스에 빌드 및 배포 역할을 추가했습니다.

8. Amazon CodeCatalyst 디스플레이 이름의 값을 복사합니다. 이 값은 나중에 워크플로를 만들 때 필요합니다.

6단계: 소스 리포지토리 만들기

이 단계에서는 에서 소스 리포지토리를 생성합니다 CodeCatalyst. 이 저장소에는 자습서의 소스 파일(예: 작업 정의 파일) 이 저장됩니다.

소스 리포지토리에 대한 자세한 내용은 을 참조하십시오. 소스 리포지토리 생성

소스 리포지토리를 생성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다`codecatalyst-ecs-project`.
3. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
4. 리포지토리 추가를 선택하고 리포지토리 생성을 선택합니다.

5. 리포지토리 이름에 다음을 입력합니다.

```
codecatalyst-ecs-source-repository
```

6. 생성(Create)을 선택합니다.

7단계: 소스 파일 추가

이 섹션에서는 Hello World 소스 파일을 리포지토리에 추가합니다. CodeCatalyst `codecatalyst-ecs-source-repository` 파일은 다음과 같이 구성됩니다.

- `index.html`파일 — 브라우저에 Hello World 메시지를 표시합니다.
- Dockerfile — Docker 이미지에 사용할 기본 이미지와 여기에 적용할 Docker 명령을 설명합니다.
- `taskdef.json`파일 - 클러스터에서 작업을 시작할 때 사용할 Docker 이미지를 정의합니다.

폴더 구조는 다음과 같습니다.

```
.
|— public-html
|   |— index.html
|— Dockerfile
|— taskdef.json
```

> **Note**
>
> 다음 지침은 CodeCatalyst 콘솔을 사용하여 파일을 추가하는 방법을 보여 주지만 원하는 경우 Git을 사용할 수 있습니다. 세부 정보는 소스 리포지토리 복제을 참조하세요.

주제

- index.html
- Dockerfile
- taskdef.json

index.html

`index.html`파일은 브라우저에 Hello World 메시지를 표시합니다.

index.html 파일을 추가하려면

1. CodeCatalyst 콘솔에서 소스 리포지토리로 이동합니다`codecatalyst-ecs-source-repository`.
2. 파일에서 파일 생성을 선택합니다.
3. 파일 이름에 다음을 입력합니다.

```
public-html/index.html
```

> **Important**
>
> 같은 이름의 폴더를 만들려면 `public-html/` 접두사를 포함해야 합니다. `index.html`이 폴더에 있을 것으로 예상됩니다.

4. 텍스트 상자에 다음 코드를 입력합니다.

```
<html>
  <head>
    <title>Hello World</title>
    <style>
      body {
      background-color: black;
      text-align: center;
      color: white;
      font-family: Arial, Helvetica, sans-serif;
      }
    </style>
  </head>
  <body>
    <h1>Hello World</h1>
  </body>
</html>
```

5. [커밋] 을 선택한 다음 [커밋] 을 다시 선택합니다.

   저장소의 `public-html` 폴더에 `index.html` 추가됩니다.

Dockerfile

Dockerfile은 사용할 기본 Docker 이미지와 여기에 적용할 Docker 명령을 설명합니다. Dockerfile에 대한 자세한 내용은 Dockerfile 참조를 참조하십시오.

여기에 지정된 Dockerfile은 Apache 2.4 기본 이미지 () 를 사용하도록 나타냅니다. `httpd` 또한 웹 페이지를 제공하는 Apache 서버의 폴더에 호출된 소스 파일을 `index.html` 복사하기 위한 지침도 포함되어 있습니다. Dockerfile의 `EXPOSE` 지침은 컨테이너가 포트 80에서 수신 중임을 Docker에 알립니다.

도커파일을 추가하려면

1. 소스 리포지토리에서 파일 생성을 선택합니다.
2. 파일 이름에 다음을 입력합니다.

   ```
   Dockerfile
   ```

   파일 확장자를 포함하지 마십시오.

   > **Important**
   >
   > Dockerfile은 리포지토리의 루트 폴더에 있어야 합니다. 워크플로의 `Docker build` 명령에서는 해당 파일이 있을 것으로 예상합니다.

3. 텍스트 상자에 다음 코드를 입력합니다.

   ```
   FROM httpd:2.4
   COPY ./public-html/index.html /usr/local/apache2/htdocs/index.html
   EXPOSE 80
   ```

4. [커밋] 을 선택한 다음 [커밋] 을 다시 선택합니다.

   Dockerfile이 리포지토리에 추가됩니다.

taskdef.json

이 단계에서 추가하는 `taskdef.json` 파일은 이미 지정한 파일과 동일하지만 다음과 같은 차이점이 있습니다. 2단계: Amazon에 플레이스홀더 애플리케이션 배포 ECS

여기의 작업 정의에서는 `image:` 필드 (`httpd:2.4`) 에 하드코딩된 Docker 이미지 이름을 지정하는 대신 두 개의 변수를 사용하여 이미지를 나타냅니다: 및. `$REPOSITORY_URI` `$IMAGE_TAG` 이러한 변수는 이후 단계에서 워크플로를 실행할 때 워크플로의 빌드 작업에서 생성된 실제 값으로 대체됩니다.

작업 정의 파라미터에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 작업 정의 파라미터를 참조하십시오.

taskdef.json 파일을 추가하려면

1. 소스 리포지토리에서 파일 생성을 선택합니다.
2. 파일 이름에 다음을 입력합니다.

taskdef.json

3. 텍스트 상자에 다음 코드를 입력합니다.

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-
execution-role",
    "containerDefinitions": [
        {
            "name": "codecatalyst-ecs-container",
            # The $REPOSITORY_URI and $IMAGE_TAG variables will be replaced
            # by the workflow at build time (see the build action in the
            # workflow)
            "image": $REPOSITORY_URI:$IMAGE_TAG,
            "essential": true,
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "networkMode": "awsvpc",
    "cpu": "256",
    "memory": "512",
```

```
    "family": "codecatalyst-ecs-task-def"
}
```

위 코드에서 다음을 대체하십시오.

arn:aws:iam::account_ID:role/codecatalyst-ecs-task-execution-role

에서 ARN 작업 실행 역할을 만들려면 언급한 작업 실행 역할로 바꾸십시오.

4. [커밋] 을 선택한 다음 [커밋] 을 다시 선택합니다.

   taskdef.json파일이 저장소에 추가됩니다.

8단계: 워크플로우 생성 및 실행

이 단계에서는 소스 파일을 가져와 Docker 이미지로 빌드한 다음 Amazon ECS 클러스터에 이미지를 배포하는 워크플로를 생성합니다. 이 배포는 기존 Apache 플레이스홀더 애플리케이션을 대체합니다.

워크플로는 순차적으로 실행되는 다음과 같은 구성 요소로 구성됩니다.

- 트리거 - 이 트리거는 소스 리포지토리에 변경 내용을 푸시할 때 워크플로가 자동으로 실행됩니다. 트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.
- 빌드 작업 (BuildBackend) — 트리거 시 작업은 Dockerfile을 사용하여 Docker 이미지를 빌드하고 Amazon에 이미지를 푸시합니다. ECR 또한 빌드 작업은 를 taskdef.json 올바른 image 필드 값으로 업데이트한 다음 이 파일의 출력 아티팩트를 생성합니다. 이 아티팩트는 다음 단계인 배포 작업의 입력으로 사용됩니다.

  빌드 작업에 대한 자세한 내용은 을 참조하십시오워크플로를 사용한 구축.
- 배포 작업 (DeployToECS) — 빌드 작업이 완료되면 배포 작업은 빌드 작업 (TaskDefArtifact) 에서 생성된 출력 아티팩트를 찾아 그 taskdef.json 내부를 찾아 Amazon ECS 서비스에 등록합니다. 그러면 서비스가 taskdef.json 파일의 지침에 따라 Amazon 클러스터 내에서 세 개의 Amazon ECS 작업과 관련 Hello World Docker 컨테이너를 실행합니다. ECS

워크플로 생성 방법

1. CodeCatalyst 콘솔의 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 워크플로 만들기를 선택합니다.
3. 소스 리포지토리의 경우 선택합니다codecatalyst-ecs-source-repository.

4. Branch의 경우 선택하십시오main.
5. 생성(Create)을 선택합니다.
6. YAML샘플 코드를 삭제합니다.
7. 다음 YAML 코드를 추가합니다.

> **Note**
>
> 다음 YAML 코드에서는 원하는 경우 Connections: 섹션을 생략할 수 있습니다. 이러한 섹션을 생략하는 경우 환경의 기본 역할 필드에 지정된 역할에 에서 설명한 두 IAM 역할의 권한 및 신뢰 정책이 포함되는지 확인해야 합니다. 5단계: AWS 역할 추가 CodeCatalyst 기본 IAM 역할을 사용하여 환경을 설정하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

```
Name: codecatalyst-ecs-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildBackend:
    Identifier: aws/build@v1
    Environment:
      Name: codecatalyst-ecs-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-ecs-build-role
    Inputs:
      Sources:
        - WorkflowSource
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-
image-repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
    Configuration:
      Steps:
```

```
        #pre_build:
        - Run: echo Logging in to Amazon ECR...
        - Run: aws --version
        - Run: aws ecr get-login-password --region us-west-2 | docker login --
username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
        #build:
        - Run: echo Build started on `date`
        - Run: echo Building the Docker image...
        - Run: docker build -t $REPOSITORY_URI:latest .
        - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
        #post_build:
        - Run: echo Build completed on `date`
        - Run: echo Pushing the Docker images...
        - Run: docker push $REPOSITORY_URI:latest
        - Run: docker push $REPOSITORY_URI:$IMAGE_TAG
        # Replace the variables in taskdef.json
        - Run: find taskdef.json -type f | xargs sed -i "s|\$REPOSITORY_URI|
$REPOSITORY_URI|g"
        - Run: find taskdef.json -type f | xargs sed -i "s|\$IMAGE_TAG|$IMAGE_TAG|
g"
        - Run: cat taskdef.json
        # The output artifact will be a zip file that contains a task definition
 file.
    Outputs:
      Artifacts:
        - Name: TaskDefArtifact
          Files:
            - taskdef.json
  DeployToECS:
    DependsOn:
      - BuildBackend
    Identifier: aws/ecs-deploy@v1
    Environment:
      Name: codecatalyst-ecs-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-ecs-deploy-role
    Inputs:
      Sources: []
      Artifacts:
        - TaskDefArtifact
    Configuration:
      region: us-west-2
      cluster: codecatalyst-ecs-cluster
```

```
      service: codecatalyst-ecs-service
      task-definition: taskdef.json
```

위 코드에서 다음을 바꾸십시오.

- 두 인스턴스 모두 *codecatalyst-ecs-environment* 만든 환경의 이름을 사용하세요사전 조건.
- 두 인스턴스 모두 *codecatalyst-account-connection* 계정 연결의 디스플레이 이름과 함께. 표시 이름은 숫자일 수 있습니다. 자세한 내용은 5단계: AWS 역할 추가 CodeCatalyst 단원을 참조하십시오.
- *codecatalyst-ecs-build-role* 에서 만든 빌드 역할의 이름을 사용합니다4단계: AWS 역할 생성.
- *111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo* (Value:속성에) 생성한 Amazon ECR 리포지토리의 정보를 포함합니다3단계: Amazon ECR 이미지 리포지토리 생성. URI
- *111122223333.dkr.ecr.us-west-2.amazonaws.com* (Run: aws ecr명령에서) 이미지 접미사 (/codecatalyst-ecs-image-repo) 가 없는 Amazon ECR 리포지토리를 사용합니다. URI
- *codecatalyst-ecs-deploy-role* 에서 생성한 배포 역할의 이름을 사용합니다. 4단계: AWS 역할 생성
- 의 두 인스턴스 모두 *us-west-2* AWS 지역 코드를 사용하세요. 지역 코드 목록은 의 지역 엔드포인트를 참조하십시오. AWS 일반 참조

> **Note**
>
> 빌드 및 배포 역할을 생성하지 않기로 결정했다면 다음을 대체하세요.*codecatalyst-ecs-build-role* 그리고 *codecatalyst-ecs-deploy-role* CodeCatalystWorkflowDevelopmentRole-*spaceName*역할 이름을 입력하세요. 이에 대한 자세한 내용은 4단계: AWS 역할 생성 섹션을 참조하세요.

> **Tip**
>
> 이전 워크플로 코드에 표시된 `find` 및 `sed` 명령을 사용하여 리포지토리와 이미지 이름을 업데이트하는 대신 Render Amazon ECS 작업 정의 작업을 이 용도로 사용할 수 있습니다. 자세한 내용은 Amazon ECS 작업 정의 수정 단원을 참조하십시오.

8. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 YAML 코드가 유효한지 확인하십시오.
9. 커밋을 선택합니다.
10. 워크플로 커밋 대화 상자에 다음을 입력합니다.

    a. 커밋 메시지의 경우 텍스트를 제거하고 다음을 입력합니다.

    ```
    Add first workflow
    ```

    b. 리포지토리의 경우 선택합니다`codecatalyst-ecs-source-repository`.

    c. 브랜치 이름으로 main을 선택합니다.

    d. 커밋을 선택합니다.

    이제 워크플로가 생성되었습니다. 워크플로 맨 위에 정의된 트리거로 인해 워크플로 실행이 자동으로 시작됩니다. 특히, `workflow.yaml` 파일을 소스 리포지토리에 커밋 (및 푸시) 하면 트리거가 워크플로 실행을 시작했습니다.

워크플로 실행 진행 상황을 보려면

1. CodeCatalyst 콘솔의 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 방금 만든 워크플로를 선택합니다. `codecatalyst-ecs-workflow`
3. 빌드 진행 상황을 BuildBackend확인하도록 선택합니다.
4. DeployToECS선택하여 배포 진행 상황을 확인하세요.

    실행 세부 정보 보기에 대한 자세한 내용은 을 참조하십시오워크플로 실행 상태 및 세부 정보 보기.

배포를 확인하려면

1. 에서 Amazon ECS 클래식 콘솔을 엽니다 https://console.aws.amazon.com/ecs/.

2. 클러스터를 선택하세요codecatalyst-ecs-cluster.
3. 작업 탭을 선택합니다.
4. 세 가지 작업 중 하나를 선택합니다.
5. 퍼블릭 IP 필드에서 공개 주소를 선택합니다.

   브라우저에 Amazon ECS 서비스가 애플리케이션을 성공적으로 배포했음을 나타내는 “Hello World” 페이지가 나타납니다.

9단계: 소스 파일 변경

이 섹션에서는 소스 리포지토리의 index.html 파일을 변경합니다. 이 변경으로 인해 워크플로우는 새 Docker 이미지를 빌드하고, 커밋 ID로 태그를 지정하고, Amazon으로 푸시하고ECR, ECS Amazon에 배포합니다.

index.html 변경하기

1. CodeCatalyst 콘솔의 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택한 다음 리포지토리를 선택합니다. codecatalyst-ecs-source-repository
2. [public-html]를 선택한 다음 [index.html]를 선택합니다.

   의 내용이 나타납니다. index.html
3. 편집을 선택합니다.
4. 14번째 줄에서 Hello World 텍스트를 로 변경합니다Tutorial complete!.
5. [커밋] 을 선택한 다음 [커밋] 을 다시 선택합니다.

   커밋하면 새 워크플로가 실행됩니다.
6. (선택 사항) 소스 리포지토리의 기본 페이지로 이동하여 커밋 보기를 선택한 다음 index.html 변경 사항의 커밋 ID를 기록해 둡니다.
7. 배포 진행 상황 보기:

   a. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.

   b. 최근 codecatalyst-ecs-workflow 실행을 보도록 선택합니다.

   c. 선택하고 BuildBackend워크플로우 실행 진행 DeployToECS상황을 확인하세요.
8. 다음과 같이 애플리케이션이 업데이트되었는지 확인하십시오.

   a. 에서 Amazon ECS 클래식 콘솔을 엽니다 https://console.aws.amazon.com/ecs/.

b. 클러스터를 선택하세요`codecatalyst-ecs-cluster`.

c. 작업 탭을 선택합니다.

d. 세 가지 작업 중 하나를 선택합니다.

e. 퍼블릭 IP 필드에서 공개 주소를 선택합니다.

`Tutorial complete!`페이지가 나타납니다.

9. (선택 사항) 에서 AWS Amazon ECR 콘솔로 전환하여 새 Docker 이미지에 6단계의 커밋 ID 태그가 지정되었는지 확인합니다.

정리

요금이 부과되지 않도록 이 자습서에서 사용되는 파일 및 서비스를 정리하세요.

AWS Management Console에서는 다음과 같은 순서로 정리하십시오.

1. ECSAmazon에서는 다음을 수행하십시오.
   a. 삭제`codecatalyst-ecs-service`.
   b. 삭제`codecatalyst-ecs-cluster`.
   c. `codecatalyst-ecs-task-definition` 등록을 취소합니다.
2. ECRAmazon에서는 삭제하십시오`codecatalyst-ecs-image-repo`.
3. EC2Amazon에서는 삭제하십시오`codecatalyst-ecs-security-group`.
4. IAMID 센터에서 삭제:
   a. `CodeCatalystECSUser`
   b. `CodeCatalystECSPermissionSet`

CodeCatalyst 콘솔에서 다음과 같이 정리합니다.

1. 삭제`codecatalyst-ecs-workflow`.
2. 삭제`codecatalyst-ecs-environment`.
3. 삭제`codecatalyst-ecs-source-repository`.
4. 삭제`codecatalyst-ecs-project`.

이 자습서에서는 CodeCatalyst 워크플로와 Amazon에 배포 ECS 작업을 사용하여 Amazon ECS 서비스에 애플리케이션을 배포하는 방법을 배웠습니다.

'아마존에 ECS 배포' 작업 추가

다음 지침에 따라 Amazon에 배포 ECS 작업을 워크플로에 추가하십시오.

Visual

시각 편집기를 사용하여 'Amazon에 ECS 배포' 작업을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. Amazon에 배포 ECS 작업을 검색하고 다음 중 하나를 수행하십시오.
   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - Amazon에 배포를 선택합니다ECS. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 액션의 소스 코드를 보려면 다운로드를 선택합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.
10. 입력 및 구성 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오'아마존에 ECS 배포' 조치 YAML. 이 참조는 편집기와 시각적 편집기에 모두 나타나는 각 필드 (및 해당 YAML 속성 값) 에 대한 자세한 정보를 제공합니다. YAML
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

편집기를 사용하여 'Amazon에 ECS 배포' 작업을 추가하려면 YAML

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. Amazon에 배포 ECS 작업을 검색하고 다음 중 하나를 수행하십시오.
   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - Amazon에 배포를 선택합니다ECS. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 액션의 소스 코드를 보려면 다운로드를 선택합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.
10. 필요에 따라 YAML 코드의 속성을 수정하십시오. 사용 가능한 각 속성에 대한 설명은 에 나와 '아마존에 ECS 배포' 조치 YAML 있습니다.
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

'아마존에 ECS 배포' 변수

Amazon에 배포 ECS 작업은 런타임에 다음 변수를 생성하고 설정합니다. 이러한 변수를 사전 정의된 변수라고 합니다.

워크플로우에서 이러한 변수를 참조하는 방법에 대한 자세한 내용은 을 참조하십시오. 사전 정의된 변수 사용

키	값
cluster	워크플로 실행 중에 배포된 Amazon ECS 클러스터의 이름. 예제: `codecatalyst-ecs-cluster`
배포 플랫폼	배포 플랫폼의 이름. 로 하드코딩했습니다. `AWS:ECS`
서비스	워크플로 실행 중에 배포된 Amazon ECS 서비스의 이름. 예제: `codecatalyst-ecs-service`
task-definition-arn	워크플로 실행 중에 등록된 작업 정의의 Amazon 리소스 이름 (ARN). 예제: `arn:aws:ecs:us-west-2:111122223333:task-definition/codecatalyst-task-def:8` 위 `:8` 예제의 내용은 등록된 수정 버전을 나타냅니다.
배포 URL	Amazon ECS 콘솔의 Events 탭으로 연결되는 링크입니다. 여기서 워크플로 실행과 관련된 Amazon ECS 배포의 세부 정보를 볼 수 있습니다. 예제: `https://console.aws.amazon.com/ecs/home?region=us-west-2#/clusters/codecatalyst-ecs-cluster/services/codecatalyst-ecs-service/events`

키	값
region	워크플로 실행 중에 AWS 리전 배포된 지역 코드입니다. 예제: `us-west-2`

'아마존에 ECS 배포' 조치 YAML

다음은 Amazon에 배포 ECS 작업의 YAML 정의입니다. 이 작업을 사용하는 방법을 알아보려면 을 참조하십시오워크플로를 ECS 사용하여 Amazon에 배포하기.

이 작업 정의는 광범위한 워크플로 정의 파일 내에 섹션으로 존재합니다. 이 파일에 대한 자세한 내용은 워크플로우 YAML 정의을 참조합니다.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

```
# The workflow definition starts here.
# See ### ## for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  ECSDeployAction_nn:
    Identifier: aws/ecs-deploy@v1
    DependsOn:
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
```

```
        - Name: account-connection-name
          Role: iam-role-name
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - task-definition-artifact
    Configuration:
      region: us-east-1
      cluster: ecs-cluster
      service: ecs-service
      task-definition: task-definition-path
      force-new-deployment: false|true
      codedeploy-appspec: app-spec-file-path
      codedeploy-application: application-name
      codedeploy-deployment-group: deployment-group-name
      codedeploy-deployment-description: deployment-description
```

ECSDeployAction

(필수)

액션의 이름을 지정합니다. 모든 작업 이름은 워크플로 내에서 고유해야 합니다. 액션 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 액션 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

기본값: `ECSDeployAction_nn`.

해당 UI: 구성 탭/ 작업 표시 이름

Identifier

(*ECSDeployAction*/Identifier)

(필수)

작업을 식별합니다. 버전을 변경하려는 경우가 아니면 이 속성을 변경하지 마십시오. 자세한 내용은 사용할 액션 버전 지정 단원을 참조하십시오.

기본값: `aws/ecs-deploy@v1`.

해당 UI: 워크플로 다이어그램/ ECSDeployAction _n/ aws/ecs-deploy @v1 레이블

DependsOn

(*ECSDeployAction*/DependsOn)

(선택 사항)

이 작업을 실행하기 위해 성공적으로 실행되어야 하는 작업, 작업 그룹 또는 게이트를 지정하십시오.

'종속 조건' 기능에 대한 자세한 내용은 을 참조하십시오. 시퀀싱 액션

해당 UI: 입력 탭/ 기준 - 선택 사항

Compute

(*ECSDeployAction*/Compute)

(선택 사항)

워크플로 작업을 실행하는 데 사용되는 컴퓨팅 엔진. 워크플로 수준 또는 작업 수준에서 컴퓨팅을 지정할 수 있지만 둘 다에서 지정할 수는 없습니다. 워크플로 수준에서 지정된 경우 컴퓨팅 구성은 워크플로에 정의된 모든 작업에 적용됩니다. 워크플로 수준에서는 동일한 인스턴스에서 여러 작업을 실행할 수도 있습니다. 자세한 내용은 작업 간 컴퓨팅 공유 단원을 참조하십시오.

해당 UI: 없음

Type

(*ECSDeployAction*/Compute/Type)

(포함된 Compute 경우 필수)

컴퓨팅 엔진 유형. 다음 값 중 하나를 사용할 수 있습니다.

- EC2(비주얼 에디터) 또는 EC2 (YAML에디터)

  작업 실행 중 유연성을 위해 최적화되었습니다.
- Lambda (비주얼 에디터) 또는 (에디터) Lambda YAML

  작업 시작 속도를 최적화했습니다.

컴퓨팅 유형에 대한 자세한 정보는 컴퓨팅 유형을 참고하세요.

해당 UI: 구성 탭/고급 - 선택 사항/ 컴퓨팅 유형

Fleet

(*ECSDeployAction*/Compute/Fleet)

(선택 사항)

워크플로 또는 워크플로 작업을 실행할 시스템 또는 플릿을 지정합니다. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다. 온디맨드 플릿의 예:,. `Linux.x86-64.Large` `Linux.x86-64.XLarge` 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

생략된 경우 기본값은 `Fleet` 입니다. `Linux.x86-64.Large`

해당 UI: 구성 탭/고급 - 선택 사항/ 컴퓨팅 플릿

Timeout

(*ECSDeployAction*/Timeout)

(선택 사항)

작업이 종료되기 전에 작업을 실행할 수 있는 시간을 분 (YAML편집기) 또는 시간과 분 (비주얼 편집기) 단위로 지정합니다. CodeCatalyst 최소값은 5분이고 최대값은 에 설명되어 의 워크플로우 할당량 CodeCatalyst 있습니다. 기본 타임아웃은 최대 타임아웃과 동일합니다.

해당 UI: 구성 탭/ 타임아웃 - 선택 사항

Environment

(*ECSDeployAction*/Environment)

(필수)

작업에 사용할 CodeCatalyst 환경을 지정합니다. 작업은 선택한 환경에 VPC 지정된 AWS 계정 및 선택적 Amazon에 연결됩니다. 작업은 환경에 지정된 기본 IAM 역할을 사용하여 Amazon에 연결하고 Amazon VPC 연결에 지정된 IAM 역할을 사용하여 Amazon에 연결합니다VPC. AWS 계정

> **Note**
>
> 기본 IAM 역할에 작업에 필요한 권한이 없는 경우 다른 역할을 사용하도록 작업을 구성할 수 있습니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.

환경에 대한 자세한 내용은 및 에 AWS 계정 배포 VPCs 및 을 참조하십시오환경 생성.

해당 UI: 구성 탭/ 환경

Name

(*ECSDeployAction*/Environment/Name)

(포함된 Environment 경우 필수)

작업에 연결할 기존 환경의 이름을 지정합니다.

해당 UI: 구성 탭/ 환경

Connections

(*ECSDeployAction*/Environment/Connections)

(새 버전의 액션에서는 선택 사항, 이전 버전에서는 필요)

액션과 연결할 계정 연결을 지정합니다. 에서 최대 1개의 계정 연결을 지정할 수 `Environment` 있습니다.

계정 연결을 지정하지 않은 경우:

- 작업은 CodeCatalyst 콘솔 환경에 지정된 AWS 계정 연결 및 기본 IAM 역할을 사용합니다. 환경에 계정 연결 및 기본 IAM 역할을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.
- 기본 IAM 역할에는 작업에 필요한 정책 및 권한이 포함되어야 합니다. 이러한 정책 및 권한이 무엇인지 확인하려면 작업 YAML 정의 설명서에서 역할 속성에 대한 설명을 참조하십시오.

계정 연결에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정. 환경에 계정 연결을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

해당 UI: 액션 버전에 따라 다음 중 하나

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환

- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Name

(*ECSDeployAction*/Environment/Connections/Name)

(포함된 경우 필수Connections)

계정 연결 이름을 지정합니다.

해당 UI: 액션 버전에 따라 다음 중 하나:

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Role

(*ECSDeployAction*/Environment/Connections/Role)

(포함된 경우 필수Connections)

Amazon에 배포 ECS 작업에서 액세스하는 데 사용하는 IAM 역할의 이름을 지정합니다 AWS. CodeCatalyst 스페이스에 역할을 추가했는지, 역할에 다음 정책이 포함되어 있는지 확인하십시오.

IAM역할을 지정하지 않으면 CodeCatalyst 콘솔 환경에 나열된 기본 IAM 역할이 작업에 사용됩니다. 환경에서 기본 역할을 사용하는 경우 해당 역할에 다음 정책이 적용되는지 확인하십시오.

- 다음 권한 정책:

> **Warning**
> 권한을 다음 정책에 표시된 권한으로 제한하십시오. 더 광범위한 권한이 있는 역할을 사용하면 보안 위험이 발생할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action":[
      "ecs:DescribeServices",
      "ecs:CreateTaskSet",
```

```
      "ecs:DeleteTaskSet",
      "ecs:ListClusters",
      "ecs:RegisterTaskDefinition",
      "ecs:UpdateServicePrimaryTaskSet",
      "ecs:UpdateService",
      "elasticloadbalancing:DescribeTargetGroups",
      "elasticloadbalancing:DescribeListeners",
      "elasticloadbalancing:ModifyListener",
      "elasticloadbalancing:DescribeRules",
      "elasticloadbalancing:ModifyRule",
      "lambda:InvokeFunction",
      "lambda:ListFunctions",
      "cloudwatch:DescribeAlarms",
      "sns:Publish",
      "sns:ListTopics",
      "s3:GetObject",
      "s3:GetObjectVersion",
      "codedeploy:CreateApplication",
      "codedeploy:CreateDeployment",
      "codedeploy:CreateDeploymentGroup",
      "codedeploy:GetApplication",
      "codedeploy:GetDeployment",
      "codedeploy:GetDeploymentGroup",
      "codedeploy:ListApplications",
      "codedeploy:ListDeploymentGroups",
      "codedeploy:ListDeployments",
      "codedeploy:StopDeployment",
      "codedeploy:GetDeploymentTarget",
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
   ],
   "Resource":"*",
   "Effect":"Allow"
},{"Action":[
      "iam:PassRole"
```

```
    ],
    "Effect":"Allow",
    "Resource":"*",
    "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
    }
}]
}
```

> **Note**
>
> 역할을 처음 사용할 때는 리소스 정책 설명에 다음 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

- 다음과 같은 사용자 지정 신뢰 정책:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

> **Note**
>
> 원하는 경우 이 작업에 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 역할을 사용할 수 있습니다. 이에 대한 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기 섹션을 참조하세요. `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*역할에는 보안 위험을 초래할 수 있는 전체 액세스 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다.

해당 UI: 액션 버전에 따라 다음 중 하나가 표시됩니다.

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 역할

Inputs

(*ECSDeployAction*/Inputs)

(선택 사항)

이 `Inputs` 섹션에서는 워크플로우 실행 중에 필요한 데이터를 정의합니다. `ECSDeployAction`

> **Note**
>
> Amazon에 배포 ECS 작업당 하나의 입력 (소스 또는 아티팩트) 만 허용됩니다.

해당 UI: 입력 탭

Sources

(*ECSDeployAction*/Inputs/Sources)

(작업 정의 파일이 소스 리포지토리에 저장되어 있는 경우 필수)

작업 정의 파일이 소스 리포지토리에 저장되어 있는 경우 해당 소스 리포지토리의 레이블을 지정하세요. 현재 지원되는 레이블은 뿐입니다`WorkflowSource`.

작업 정의 파일이 소스 리포지토리에 포함되어 있지 않은 경우 이 파일은 다른 작업에서 생성된 아티팩트에 있어야 합니다.

소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

해당 UI: 입력 탭/ 소스 - 선택 사항

Artifacts - input

(*ECSDeployAction*/Inputs/Artifacts)

(작업 정의 파일이 이전 작업의 출력 아티팩트에 저장되어 있는 경우 필수)

배포하려는 작업 정의 파일이 이전 작업에서 생성된 아티팩트에 포함되어 있는 경우 여기에 해당 아티팩트를 지정하세요. 작업 정의 파일이 아티팩트에 포함되어 있지 않은 경우 해당 파일은 소스 리포지토리에 있어야 합니다.

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 구성 탭/ 아티팩트 - 선택 사항

Configuration

(*ECSDeployAction*/Configuration)

(필수)

작업의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭

region

(Configuration/region)

(필수)

Amazon ECS 클러스터 및 서비스가 위치한 AWS 지역을 지정합니다. 지역 코드 목록은 의 지역 엔드포인트를 참조하십시오. AWS 일반 참조

해당 UI: 구성 탭/ 지역

cluster

(*ECSDeployAction*/Configuration/cluster)

(필수)

기존 Amazon ECS 클러스터의 이름을 지정합니다. Amazon에 배포 ECS 작업은 컨테이너화된 애플리케이션을 태스크로 이 클러스터에 배포합니다. Amazon ECS 클러스터에 대한 자세한 내용은 Amazon Elastic 컨테이너 서비스 개발자 안내서의 클러스터를 참조하십시오.

해당 UI: 구성 탭/ 클러스터

service

(*ECSDeployAction*/Configuration/service)

(필수)

작업 정의 파일을 인스턴스화할 기존 Amazon ECS 서비스의 이름을 지정합니다. 이 서비스는 필드에 지정된 클러스터 아래에 있어야 합니다. `cluster` Amazon ECS 서비스에 대한 자세한 내용은 Amazon Elastic 컨테이너 서비스 개발자 안내서의 Amazon ECS 서비스를 참조하십시오.

해당 UI: 구성 탭/ 서비스

task-definition

(*ECSDeployAction*/Configuration/task-definition)

(필수)

기존 작업 정의 파일의 경로를 지정합니다. 파일이 소스 리포지토리에 있는 경우 경로는 소스 리포지토리 루트 폴더를 기준으로 합니다. 파일이 이전 워크플로 작업의 아티팩트에 있는 경우 경로는 아티팩트 루트 폴더를 기준으로 합니다. 작업 정의 파일에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 작업 정의를 참조하십시오.

해당 UI: 구성 탭/ 작업 정의

force-new-deployment

(*ECSDeployAction*/Configuration/force-new-deployment)

(필수)

활성화된 경우 Amazon ECS 서비스는 서비스 정의를 변경하지 않고도 새 배포를 시작할 수 있습니다. 강제 배포를 수행하면 서비스가 현재 실행 중인 모든 작업을 중지하고 새 작업을 시작합니다. 새 배포를 강제하는 방법에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 서비스 업데이트를 참조하십시오.

기본값: `false`

해당 UI: 구성 탭/ 서비스를 강제로 새로 배포합니다.

codedeploy-appspec

(*ECSDeployAction*/Configuration/codedeploy-appspec)

(블루/그린 배포를 사용하도록 Amazon ECS 서비스를 구성한 경우 필수, 그렇지 않으면 생략)

기존 CodeDeploy 애플리케이션 사양 () 파일의 이름과 경로를 지정합니다. AppSpec 이 파일은 CodeCatalyst 소스 저장소의 루트에 있어야 합니다. AppSpec 파일에 대한 자세한 내용은 AWS CodeDeploy 사용 설명서의 CodeDeploy 응용 프로그램 사양 (AppSpec) 파일을 참조하십시오.

> **Note**
>
> 블루/그린 배포를 수행하도록 Amazon ECS 서비스를 구성한 경우에만 CodeDeploy 정보를 제공하십시오. 롤링 업데이트 배포 (기본값) 의 경우 정보를 생략하십시오. CodeDeploy Amazon ECS 배포에 대한 자세한 내용은 Amazon Elastic 컨테이너 서비스 개발자 안내서의 Amazon ECS 배포 유형을 참조하십시오.

> **Note**
>
> 비주얼 에디터에서는 CodeDeploy필드가 숨겨질 수 있습니다. 표시되도록 하려면 을 참조하십시오비주얼 에디터에서 CodeDeploy 필드가 누락된 이유는 무엇입니까?.

해당 UI: 구성 탭/ CodeDeploy AppSpec

codedeploy-application

(*ECSDeployAction*/Configuration/codedeploy-application)

(포함된 `codedeploy-appspec` 경우 필수)

기존 CodeDeploy 애플리케이션의 이름을 지정합니다. CodeDeploy 응용 프로그램에 대한 자세한 내용은 사용 설명서의 응용 프로그램AWS CodeDeploy 사용을 참조하십시오. CodeDeploy

해당 UI: 구성 탭/ 애플리케이션 CodeDeploy

codedeploy-deployment-group

(*ECSDeployAction*/Configuration/codedeploy-deployment-group)

(포함된 codedeploy-appspec 경우 필수)

기존 CodeDeploy 배포 그룹의 이름을 지정합니다. CodeDeploy 배포 그룹에 대한 자세한 내용은 AWS CodeDeploy 사용 설명서의 배포 그룹 작업을 참조하십시오. CodeDeploy

해당 UI: 구성 탭/ CodeDeploy 배포 그룹

codedeploy-deployment-description

(*ECSDeployAction*/Configuration/codedeploy-deployment-description)

(선택 사항)

이 작업으로 생성될 배포에 대한 설명을 지정합니다. 자세한 내용은 AWS CodeDeploy 사용 설명서의 배포 작업을 참조하십시오. CodeDeploy

해당 UI: 구성 탭/ 배포 설명 CodeDeploy

워크플로를 EKS 사용하여 Amazon에 배포하기

> **Tip**
>
> Deploy to Kubernetes 클러스터 작업을 사용하는 방법을 보여주는 자습서는 을 참조하십시오. 자습서: Amazon에 애플리케이션 배포 EKS

이 섹션에서는 워크플로를 사용하여 컨테이너화된 애플리케이션을 Kubernetes 클러스터에 배포하는 방법을 설명합니다. CodeCatalyst 이를 위해서는 Kubernetes 클러스터에 배포 작업을 워크플로에 추가해야 합니다. 이 작업은 하나 이상의 쿠버네티스 매니페스트 파일을 사용하여 Amazon Elastic Kubernetes Service () 에서 설정한 쿠버네티스 클러스터에 애플리케이션을 배포합니다EKS. 샘플 매니페스트는 in을 참조하십시오. deployment.yaml 자습서: Amazon에 애플리케이션 배포 EKS

쿠버네티스에 대한 자세한 내용은 쿠버네티스 설명서를 참조하십시오.

Amazon에 대한 자세한 내용은 EKS Amazon이란 무엇입니까EKS? 를 참조하십시오. Amazon EKS 사용 설명서에서 확인할 수 있습니다.

'Kubernetes 클러스터에 배포' 작업의 작동 방식

쿠버네티스에 배포 클러스터는 다음과 같이 작동합니다.

1. 런타임 시 작업은 작업이 실행되는 CodeCatalyst 컴퓨팅 머신에 Kubernetes `kubectl` 유틸리티를 설치합니다. 작업은 작업을 구성할 때 제공한 Amazon EKS 클러스터를 `kubectl` 가리키도록 구성됩니다. 다음에 `kubectl apply` 명령을 실행하려면 `kubectl` 유틸리티가 필요합니다.
2. 액션은 명령을 실행하며, `kubectl apply -f my-manifest.yaml` 명령은 다음 지침을 수행합니다.*my-manifest.yaml* 애플리케이션을 구성된 클러스터에 컨테이너 및 포드 세트로 배포합니다. 이 명령에 대한 자세한 내용은 쿠버네티스 참조 문서의 kubectl apply 항목을 참조하십시오.

주제

- 자습서: Amazon에 애플리케이션 배포 EKS
- '쿠버네티스 클러스터에 배포' 작업 추가
- '쿠버네티스 클러스터에 배포' 변수
- '쿠버네티스 클러스터에 배포' 작업 YAML

자습서: Amazon에 애플리케이션 배포 EKS

이 자습서에서는 Amazon EKS 워크플로, Amazon 및 기타 몇 가지 서비스를 사용하여 CodeCatalyst Amazon Elastic Kubernetes Service에 컨테이너화된 애플리케이션을 배포하는 방법을 알아봅니다. AWS 배포된 애플리케이션은 간단한 'Hello, World! '입니다. Apache 웹 서버 Docker 이미지를 기반으로 구축된 웹 사이트. 이 자습서에서는 개발 시스템 및 Amazon EKS 클러스터 설정과 같은 필수 준비 작업을 안내한 다음 애플리케이션을 구축하고 클러스터에 배포하기 위한 워크플로를 생성하는 방법을 설명합니다.

초기 배포가 완료되면 자습서에서는 애플리케이션 소스를 변경하라는 지침을 제공합니다. 이 변경으로 인해 새 Docker 이미지가 빌드되고 새 수정 정보와 함께 Docker 이미지 리포지토리로 푸시됩니다. 그러면 새 버전의 Docker 이미지가 EKS Amazon에 배포됩니다.

> **Tip**
>
> 이 튜토리얼을 따라가는 대신 완전한 Amazon EKS 설정을 수행하는 블루프린트를 사용할 수 있습니다. EKS앱 배포 블루프린트를 사용해야 합니다. 자세한 내용은 블루프린트로 프로젝트 생성 단원을 참조하십시오.

주제

- 사전 조건

사전 조건

이 자습서를 시작하기 전:

- 연결된 AWS 계정이 있는 Amazon CodeCatalyst 스페이스가 필요합니다. 자세한 내용은 스페이스 만들기 단원을 참조하십시오.
- 스페이스에 다음과 같은 빈 프로젝트가 필요합니다.

  ```
  codecatalyst-eks-project
  ```

  처음부터 시작 옵션을 사용하여 이 프로젝트를 만들 수 있습니다.

  자세한 내용은 Amazon에서 빈 프로젝트 생성 CodeCatalyst 단원을 참조하십시오.
- 프로젝트에는 다음과 같은 빈 CodeCatalyst 소스 리포지토리가 필요합니다.

  ```
  codecatalyst-eks-source-repository
  ```

  자세한 내용은 소스 리포지토리를 사용하여 코드를 저장하고 공동 작업하십시오. CodeCatalyst 단원을 참조하십시오.
- 프로젝트에는 다음과 같은 CodeCatalyst CI/CD 환경 (개발 환경 아님) 이 필요합니다.

  ```
  codecatalyst-eks-environment
  ```

  이 환경을 다음과 같이 구성하십시오.

- 원하는 유형 (예: 비프로덕션) 을 선택합니다.
- AWS 계정을 여기에 연결하십시오.
- 기본 IAM 역할의 경우 원하는 역할을 선택합니다. 나중에 다른 역할을 지정하게 됩니다.

자세한 내용은 및 에 AWS 계정 배포 VPCs 단원을 참조하십시오.

1단계: 개발 머신 설정

이 자습서의 첫 번째 단계는 이 자습서 전체에서 사용할 몇 가지 도구를 사용하여 개발 컴퓨터를 구성하는 것입니다. 이러한 도구는 다음과 같습니다.

- `eksctl`유틸리티 — 클러스터 생성용
- `kubectl`유틸리티 — 필수 조건 `eksctl`
- AWS CLI — 의 전제 조건이기도 합니다. `eksctl`

기존 개발 시스템이 있는 경우 이러한 도구를 설치하거나 클라우드 기반 CodeCatalyst 개발 환경을 사용할 수 있습니다. CodeCatalyst 개발 환경의 이점은 스핀업 및 제거가 쉽고 다른 CodeCatalyst 서비스와 통합되어 더 적은 단계로 이 자습서를 완료할 수 있다는 것입니다.

이 자습서에서는 CodeCatalyst 개발 환경을 사용한다고 가정합니다.

다음 지침은 CodeCatalyst 개발 환경을 시작하고 필요한 도구를 사용하여 구성하는 빠른 방법을 설명하지만 자세한 지침은 다음을 참조하십시오.

- Dev Environment 생성 단원을 참조하십시오.
- 아마존 EKS 사용자 가이드에서 kubectl 설치하기.
- Amazon EKS 사용 설명서의 eksctl 설치 또는 업그레이드
- AWS Command Line Interface 사용 AWS CLI설명서에 있는 최신 버전의 설치 또는 업데이트.

개발 환경을 시작하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다`codecatalyst-eks-project`.
3. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
4. 소스 리포지토리의 이름을 선택합니다`codecatalyst-eks-source-repository`.
5. 상단에서 개발 환경 생성을 선택한 다음 AWS Cloud9 (브라우저에서) 를 선택합니다.

6. 기존 브랜치 및 메인 브랜치에서의 작업이 선택되어 있는지 확인한 다음 [Create] 를 선택합니다.

   개발 환경이 새 브라우저 탭에서 시작되고 리포지토리 (codecatalyst-eks-source-repository) 가 새 브라우저 탭에 복제됩니다.

kubectl을 설치하고 구성하려면

1. 개발 환경 터미널에서 다음을 입력합니다.

   ```
   curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl
   ```

2. 다음을 입력합니다.

   ```
   chmod +x ./kubectl
   ```

3. 입력:

   ```
   mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
   ```

4. 입력:

   ```
   echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc
   ```

5. 입력:

   ```
   kubectl version --short --client
   ```

6. 버전이 나타나는지 확인하세요.

   이제 설치가 완료되었습니다kubectl.

eksctl을 설치하고 구성하려면

> **Note**
>
> eksctl대신 사용할 kubectl 수 있으므로 반드시 필요한 것은 아닙니다. 하지만 eksctl 클러스터 구성의 대부분을 자동화할 수 있다는 이점이 있으므로 이 자습서에서는 이 도구를 사용하는 것이 좋습니다.

1. 개발 환경 터미널에서 다음을 입력합니다.

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/
download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
```

2. 다음을 입력합니다.

```
sudo cp /tmp/eksctl /usr/bin
```

3. 입력:

```
eksctl version
```

4. 버전이 나타나는지 확인하세요.

   이제 설치가 완료되었습니다`eksctl`.

가 AWS CLI 설치되었는지 확인하려면

1. 개발 환경 터미널에서 다음을 입력합니다.

```
aws --version
```

2. 버전이 나타나는지 확인하여 가 설치되어 AWS CLI 있는지 확인합니다.

   나머지 절차를 완료하여 액세스에 필요한 AWS CLI 권한으로 를 AWS구성하십시오.

구성하려면 AWS CLI

액세스 키와 세션 AWS CLI 토큰으로 를 구성하여 AWS 서비스에 대한 액세스 권한을 부여해야 합니다. 다음 지침은 키와 토큰을 구성하는 빠른 방법을 제공하지만, 자세한 지침을 보려면 사용 AWS Command Line Interface 설명서의 AWS CLI구성을 참조하십시오.

1. 다음과 같이 IAM Identity Center 사용자를 생성합니다.

   a. 에서 AWS Management Console 로그인하고 AWS IAM Identity Center 콘솔을 엽니다 https://console.aws.amazon.com/singlesignon/.

      (이전에 IAM Identity Center에 로그인한 적이 없다면 활성화를 선택해야 할 수도 있습니다.)

> **Note**
>
> CodeCatalyst스페이스에 AWS 계정 연결된 서버를 사용하여 로그인해야 합니다. 스페이스로 이동하고 계정 탭을 선택하면 어떤 계정이 연결되어 있는지 확인할 수 있습니다. AWS 자세한 내용은 스페이스 만들기 단원을 참조하십시오.

b. 탐색 창에서 Users와 Add user를 차례대로 선택합니다.

c. 사용자 이름에 다음을 입력합니다.

```
codecatalyst-eks-user
```

d. 비밀번호에서 이 사용자와 공유할 수 있는 일회용 비밀번호 생성을 선택합니다.

e. 이메일 주소 및 이메일 주소 확인에 IAM Identity Center에 아직 없는 이메일 주소를 입력합니다.

f. 이름에 다음을 입력합니다.

```
codecatalyst-eks-user
```

g. 성에 다음을 입력합니다.

```
codecatalyst-eks-user
```

h. 표시 이름에 다음을 입력합니다.

```
codecatalyst-eks-user codecatalyst-eks-user
```

i. Next(다음)를 선택합니다.

j. 그룹에 사용자 추가 페이지에서 다음을 선택합니다.

k. 사용자 검토 및 추가 페이지에서 정보를 검토하고 사용자 추가를 선택합니다.

일회용 암호 대화 상자가 나타납니다.

l. 복사를 선택한 다음 로그인 정보를 텍스트 파일에 붙여넣습니다. 로그인 정보는 AWS 액세스 포털URL, 사용자 이름, 일회용 비밀번호로 구성됩니다.

m. 닫기를 선택하세요.

2. 다음과 같이 권한 집합을 생성합니다.

a. 탐색 창에서 권한 세트를 선택한 다음 권한 세트 생성을 선택합니다.

b. 사전 정의된 권한 집합을 선택한 다음 선택합니다 AdministratorAccess. 이 정책은 모든 AWS 서비스사람에게 전체 권한을 제공합니다.

c. Next(다음)를 선택합니다.

d. 권한 집합 이름에서 다음을 `AdministratorAccess` 제거하고 입력합니다.

```
codecatalyst-eks-permission-set
```

e. Next(다음)를 선택합니다.

f. 검토 및 생성 페이지에서 정보를 검토하고 생성을 선택합니다.

3. 다음과 같이 권한 세트를 에 `codecatalyst-eks-user` 할당합니다.

a. 탐색 창에서 선택한 다음 현재 로그인한 권한 옆의 AWS 계정 확인란을 선택합니다. AWS 계정

b. 사용자 또는 그룹 할당을 선택합니다.

c. 사용자 탭을 선택합니다.

d. `codecatalyst-eks-user` 옆의 확인란을 선택합니다.

e. Next(다음)를 선택합니다.

f. `codecatalyst-eks-permission-set` 옆의 확인란을 선택합니다.

g. Next(다음)를 선택합니다.

h. 정보를 검토하고 제출을 선택합니다.

이제 두 개를 하나로 `codecatalyst-eks-user` AWS 계정묶어 `codecatalyst-eks-permission-set` 자신에게 할당했습니다.

4. `codecatalyst-eks-user`Gavein의 액세스 키와 세션 토큰은 다음과 같습니다.

a. AWS 액세스 URL 포털과 사용자 이름 및 일회용 비밀번호가 있는지 확인하세요. `codecatalyst-eks-user` 이전에 이 정보를 텍스트 편집기에 복사했어야 합니다.

> **Note**
>
> 이 정보가 없는 경우 IAM Identity Center의 `codecatalyst-eks-user` 세부 정보 페이지로 이동하여 암호 재설정, 일회용 암호 생성 [...] 을 선택합니다. 그리고 암호를 다시 재설정하여 화면에 정보를 표시하십시오.

b. AWS로그아웃하세요.

c. AWS 액세스 URL 포털을 브라우저의 주소 표시줄에 붙여넣습니다.

d. 다음 방법으로 로그인하세요.

- 사용자명:

```
codecatalyst-eks-user
```

- 비밀번호:

one-time-password

e. 새 암호 설정에서 새 암호를 입력하고 새 암호 설정을 선택합니다.

화면에 AWS 계정 상자가 나타납니다.

f. 선택한 AWS 계정다음 `codecatalyst-eks-user` 사용자에게 할당한 AWS 계정 이름과 권한 집합을 선택합니다.

g. 다음으로 명령줄 또는 프로그래밍 방식 액세스를 선택합니다. `codecatalyst-eks-permission-set`

h. 페이지 중간에 있는 명령을 복사합니다. 다음과 비슷해 보입니다.

```
export AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
export AWS_SESSION_TOKEN="session-token"
```

... 어디서 *session-token* 긴 무작위 문자열입니다.

5. 다음과 같이 액세스 키와 세션 AWS CLI토큰을 에 추가합니다.

a. CodeCatalyst 개발 환경으로 돌아가십시오.

b. 터미널 프롬프트에서 복사한 명령을 붙여넣습니다. Enter를 누릅니다.

이제 액세스 AWS CLI 키와 세션 토큰을 사용하여 를 구성했습니다. 이제 를 AWS CLI 사용하여 이 자습서에 필요한 작업을 완료할 수 있습니다.

> **Important**
>
> 이 자습서를 진행하는 동안 언제든지 다음과 비슷한 메시지가 표시되는 경우:
> `Unable to locate credentials. You can configure credentials by running "aws configure".`
> 또는 다음과 같습니다.

`ExpiredToken: The security token included in the request is expired`
... AWS CLI 세션이 만료되었기 때문입니다. 이 경우에는 `aws configure` 명령을 실행하지 마십시오. 대신 로 `Obtain codecatalyst-eks-user's access key and session token` 시작하는 이 절차의 4단계에 있는 지침을 사용하여 세션을 새로 고치십시오.

2단계: Amazon EKS 클러스터 생성

이 섹션에서는 Amazon에서 클러스터를 생성합니다EKS. 아래 지침은 를 사용하여 클러스터를 빠르게 생성하는 방법을 `eksctl` 설명하지만 자세한 지침은 다음을 참조하십시오.

- 아마존 EKS 사용 설명서에서 eksctl 시작하기

  또는

- 콘솔 및 AWS CLIAmazon EKS 사용 설명서에서 시작하기 (이 항목에서는 클러스터 생성 `kubectl` 지침을 제공합니다)

> **Note**
>
> CodeCatalyst Amazon과의 통합은 프라이빗 클러스터를 지원하지 않습니다EKS.

시작하기 전에

개발 머신에서 다음 작업을 완료했는지 확인하십시오.

- `eksctl`유틸리티를 설치했습니다.
- `kubectl`유틸리티를 설치했습니다.
- 를 AWS CLI 설치하고 액세스 키와 세션 토큰을 사용하여 구성했습니다.

이러한 작업을 완료하는 방법에 대한 자세한 내용은 을 참조하십시오1단계: 개발 머신 설정.

클러스터 생성

> **Important**
>
> 클러스터가 제대로 구성되지 않으므로 Amazon EKS 서비스의 사용자 인터페이스를 사용하여 클러스터를 생성하지 마십시오. 다음 단계에 설명된 대로 `eksctl` 유틸리티를 사용하십시오.

1. 개발 환경으로 이동합니다.
2. 클러스터 및 노드 생성:

   ```
   eksctl create cluster --name codecatalyst-eks-cluster --region us-west-2
   ```

   위치:

   - *codecatalyst-eks-cluster* 클러스터에 부여하려는 이름으로 바꿉니다.
   - *us-west-2* 해당 지역으로 대체됩니다.

   10~20분 후 다음과 비슷한 메시지가 나타납니다.

   ```
   EKS cluster "codecatalyst-eks-cluster" in "us-west-2" region is ready
   ```

   > **Note**
   >
   > 클러스터를 AWS 생성하는 동안 여러 `waiting for CloudFormation stack` 메시지가 표시됩니다. 이는 예상된 동작입니다.

3. 클러스터가 성공적으로 생성되었는지 확인하세요.

   ```
   kubectl cluster-info
   ```

   클러스터 생성이 성공했음을 나타내는 다음과 비슷한 메시지가 표시됩니다.

   ```
   Kubernetes master is running at https://long-string.gr7.us-west-2.eks.amazonaws.com
   CoreDNS is running at https://long-string.gr7.us-west-2.eks.amazonaws.com/api/v1/
   namespaces/kube-system/services/kube-dns:dns/proxy
   ```

3단계: Amazon ECR 이미지 리포지토리 생성

이 섹션에서는 Amazon Elastic 컨테이너 레지스트리 (AmazonECR) 에 프라이빗 이미지 리포지토리를 생성합니다. 이 리포지토리는 자습서에 사용할 Docker 이미지를 저장합니다.

Amazon에 대한 자세한 내용은 Amazon ECR Elastic 컨테이너 레지스트리 사용 설명서를 참조하십시오.

Amazon에서 이미지 리포지토리를 만들려면 ECR

1. 개발 환경으로 이동하십시오.
2. Amazon에서 빈 리포지토리를 생성합니다ECR.

   ```
   aws ecr create-repository --repository-name codecatalyst-eks-image-repo
   ```

   Replace *codecatalyst-eks-image-repo* Amazon ECR 리포지토리에 부여하려는 이름을 사용합니다.

   이 자습서에서는 리포지토리 `codecatalyst-eks-image-repo` 이름을 지정했다고 가정합니다.
3. Amazon ECR 리포지토리의 세부 정보를 표시합니다.

   ```
   aws ecr describe-repositories \
         --repository-names codecatalyst-eks-image-repo
   ```
4. `"repositoryUri":`값을 기록해 둡니다 (예:)`111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-image-repo`.

   나중에 워크플로에 리포지토리를 추가할 때 필요합니다.

4단계: 소스 파일 추가

이 섹션에서는 소스 리포지토리 (`codecatalyst-eks-source-repository`) 에 애플리케이션 소스 파일을 추가합니다. 구성 요소는 다음과 같습니다.

- `index.html`파일 — 'Hello, World! '라는 메시지가 표시됩니다. 브라우저의 메시지.
- Dockerfile — Docker 이미지에 사용할 기본 이미지와 여기에 적용할 Docker 명령을 설명합니다.
- `deployment.yaml`파일 — 쿠버네티스 서비스 및 배포를 정의하는 쿠버네티스 매니페스트입니다.

폴더 구조는 다음과 같습니다.

```
|– codecatalyst-eks-source-repository
   |– Kubernetes
      |– deployment.yaml
   |– public-html
   |  |– index.html
   |– Dockerfile
```

주제

- index.html
- Dockerfile
- deployment.yaml

index.html

`index.html`파일에는 'Hello, World! '라는 문구가 표시됩니다. 브라우저의 메시지.

index.html 파일을 추가하려면

1. 개발 환경으로 이동합니다.
2. 에서 `codecatalyst-eks-source-repository` 라는 `public-html` 폴더를 생성합니다.
3. 에서 `/public-html` 다음 내용이 `index.html` 포함된 파일을 생성합니다.

```
<html>
  <head>
    <title>Hello World</title>
    <style>
      body {
      background-color: black;
      text-align: center;
      color: white;
      font-family: Arial, Helvetica, sans-serif;
      }
    </style>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
```

```
</html>
```

4. 터미널 프롬프트에서 다음을 입력합니다.

```
cd /projects/codecatalyst-eks-source-repository
```

5. 추가, 커밋, 푸시:

```
git add .
git commit -m "add public-html/index.html"
git push
```

저장소의 `public-html` 폴더에 `index.html` 추가됩니다.

Dockerfile

Dockerfile은 사용할 기본 Docker 이미지와 여기에 적용할 Docker 명령을 설명합니다. Dockerfile에 대한 자세한 내용은 Dockerfile 참조를 참조하십시오.

여기에 지정된 Dockerfile은 Apache 2.4 기본 이미지 () 를 사용하도록 나타냅니다. `httpd` 또한 웹 페이지를 제공하는 Apache 서버의 폴더에 호출된 소스 파일을 `index.html` 복사하기 위한 지침도 포함되어 있습니다. Dockerfile의 `EXPOSE` 지침은 컨테이너가 포트 80에서 수신 중임을 Docker에 알립니다.

도커파일을 추가하려면

1. 에서 `codecatalyst-eks-source-repository` 다음 내용이 `Dockerfile` 포함된 파일을 생성합니다.

```
FROM httpd:2.4
COPY ./public-html/index.html /usr/local/apache2/htdocs/index.html
EXPOSE 80
```

파일 확장자를 포함하지 마세요.

> **Important**
>
> Dockerfile은 리포지토리의 루트 폴더에 있어야 합니다. 워크플로의 `Docker build` 명령에서는 해당 파일이 있을 것으로 예상합니다.

2. 추가, 커밋, 푸시:

```
git add .
git commit -m "add Dockerfile"
git push
```

Dockerfile이 저장소에 추가됩니다.

deployment.yaml

이 섹션에서는 리포지토리에 `deployment.yaml` 파일을 추가합니다. `deployment.yaml`파일은 실행할 쿠버네티스 리소스 유형 또는 종류인 '서비스'와 '디플로이먼트'를 정의하는 쿠버네티스 매니페스트입니다.

- '서비스'는 Amazon에 로드 밸런서를 배포합니다. EC2 로드 밸런서는 'Hello, World! '페이지로 이동하는 데 사용할 수 있는 인터넷 공용 URL 및 표준 포트 (포트 80) 를 제공합니다. 애플리케이션을 배포합니다.
- '디플로이먼트'는 포드 3개를 배포하며, 각 포드에는 'Hello, World! '가 포함된 Docker 컨테이너가 포함됩니다. 애플리케이션을 배포합니다. 세 개의 포드는 클러스터를 생성할 때 생성된 노드에 배포됩니다.

이 튜토리얼의 매니페스트는 짧지만, 매니페스트에는 포드, 작업, 인그레스, 네트워크 정책 등 여러 Kubernetes 리소스 유형이 포함될 수 있습니다. 또한 배포가 복잡한 경우 매니페스트 파일을 여러 개 사용할 수 있습니다.

배포.yaml 파일을 추가하려면

1. 에서 라는 폴더를 `codecatalyst-eks-source-repository` 생성합니다. `Kubernetes`
2. 에서 `/Kubernetes` 다음 내용이 `deployment.yaml` 포함된 파일을 생성합니다.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
  labels:
    app: my-app
spec:
  type: LoadBalancer
  selector:
    app: my-app
```

```
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  labels:
    app: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: codecatalyst-eks-container
        # The $REPOSITORY_URI and $IMAGE_TAG placeholders will be replaced by
 actual values supplied by the build action in your workflow
        image: $REPOSITORY_URI:$IMAGE_TAG
        ports:
        - containerPort: 80
```

3. 추가, 커밋, 푸시:

```
git add .
git commit -m "add Kubernetes/deployment.yaml"
git push
```

deployment.yaml파일이 저장소의 라는 폴더에 추가됩니다Kubernetes.

이제 모든 소스 파일을 추가했습니다.

잠시 시간을 내어 작업을 다시 한 번 확인하고 모든 파일을 올바른 폴더에 배치했는지 확인하세요. 폴더 구조는 다음과 같습니다.

```
|– codecatalyst-eks-source-repository
   |– Kubernetes
      |– deployment.yaml
   |– public-html
   |  |– index.html
   |– Dockerfile
```

5단계: AWS 역할 생성

이 섹션에서는 CodeCatalyst 워크플로가 작동하는 데 필요한 AWS IAM 역할을 만듭니다. 이러한 역할은 다음과 같습니다.

- 빌드 역할 - (워크플로의) CodeCatalyst 빌드 작업에 AWS 계정에 액세스하고 Amazon ECR 및 Amazon에 글을 쓸 수 있는 권한을 EC2 부여합니다.
- 배포 역할 - Kubernetes CodeCatalyst 배포 클러스터 작업 (워크플로의) 에 사용자 AWS 계정과 Amazon에 액세스할 수 있는 권한을 부여합니다. EKS

IAM역할에 대한 자세한 내용은 사용 설명서의 IAM AWS Identity and Access Management 역할을 참조하십시오.

> **Note**
>
> 시간을 절약하기 위해 앞서 나열한 두 역할 대신 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 역할이라는 단일 역할을 만들 수 있습니다. 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기 단원을 참조하십시오. `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*역할에는 보안 위험을 초래할 수 있는 매우 광범위한 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다. 이 자습서에서는 앞서 나열한 두 개의 역할을 생성한다고 가정합니다.

빌드 및 배포 역할을 만들려면 다음 일련의 절차를 완료하세요.

1. 두 역할 모두에 대한 신뢰 정책을 만들려면

1. 개발 환경으로 이동하세요.

2. Cloud9-*long-string*디렉토리에서 다음 내용이 codecatalyst-eks-trust-policy.json 포함된 파일을 생성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. 빌드 역할에 대한 빌드 정책을 만들려면

- Cloud9-*long-string*디렉터리에 다음 내용이 codecatalyst-eks-build-policy.json 포함된 파일을 생성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

3. 배포 역할에 대한 배포 정책을 만들려면

- Cloud9-*long-string*디렉터리에 다음 내용이 codecatalyst-eks-deploy-policy.json 포함된 파일을 생성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

이제 개발 환경에 세 개의 정책 문서를 추가했습니다. 이제 디렉터리 구조가 다음과 같이 보입니다.

```
|— Cloud9-long-string
```

```
|– .c9
|– codecatalyst-eks-source-repository
   |– Kubernetes
   |– public-html
   |– Dockerfile
codecatalyst-eks-build-policy.json
codecatalyst-eks-deploy-policy.json
codecatalyst-eks-trust-policy.json
```

4. 빌드 정책을 추가하려면 AWS

1. 개발 환경 터미널에서 다음을 입력합니다.

```
cd /projects
```

2. 다음을 입력합니다.

```
aws iam create-policy \
    --policy-name codecatalyst-eks-build-policy \
    --policy-document file://codecatalyst-eks-build-policy.json
```

3. Enter를 누릅니다.
4. 명령 출력에서 `"arn":` 값 (예:) 을 기록해 둡니다`arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy`. ARN나중에 필요합니다.

5. 배포 정책을 추가하려면 AWS

1. 다음을 입력합니다.

```
aws iam create-policy \
    --policy-name codecatalyst-eks-deploy-policy \
    --policy-document file://codecatalyst-eks-deploy-policy.json
```

2. Enter를 누릅니다.
3. 명령 출력에서 배포 정책 `"arn":` 값 (예:) 을 기록해 둡니다`arn:aws:iam::111122223333:policy/codecatalyst-eks-deploy-policy`. ARN나중에 이 정보가 필요합니다.

6. 빌드 역할을 만들려면

1. 다음을 입력합니다.

```
aws iam create-role \
      --role-name codecatalyst-eks-build-role \
      --assume-role-policy-document file://codecatalyst-eks-trust-policy.json
```

2. Enter를 누릅니다.
3. 입력:

```
aws iam attach-role-policy \
      --role-name codecatalyst-eks-build-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy
```

위치 *arn:aws:iam::111122223333:policy/codecatalyst-eks-build-policy* 앞에서 언급한 빌드 정책으로 대체됩니다. ARN

4. Enter를 누릅니다.
5. 터미널 프롬프트에서 다음을 입력합니다.

```
aws iam get-role \
      --role-name codecatalyst-eks-build-role
```

6. Enter를 누릅니다.
7. 역할 `"Arn":` 값 (예:) 을 기록해 둡니다`arn:aws:iam::111122223333:role/codecatalyst-eks-build-role`. ARN나중에 필요합니다.

7. 배포 역할을 만들려면

1. 다음을 입력합니다.

```
aws iam create-role \
      --role-name codecatalyst-eks-deploy-role \
      --assume-role-policy-document file://codecatalyst-eks-trust-policy.json
```

2. Enter를 누릅니다.
3. 입력:

```
aws iam attach-role-policy \
```

```
      --role-name codecatalyst-eks-deploy-role \
      --policy-arn arn:aws:iam::111122223333:policy/codecatalyst-eks-deploy-policy
```

위치 *arn:aws:iam::111122223333:policy/codecatalyst-eks-deploy-policy* 앞에서 언급한 배포 정책으로 대체됩니다. ARN

4. Enter를 누릅니다.
5. 다음을 입력합니다.

```
aws iam get-role \
      --role-name codecatalyst-eks-deploy-role
```

6. Enter를 누릅니다.
7. 역할 `"Arn":` 값 (예:) 을 기록해 둡니다`arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role`. ARN나중에 필요합니다.

이제 빌드 및 배포 역할을 생성하고 해당 역할을 기록했습니다ARNs.

6단계: AWS 역할 추가 CodeCatalyst

이 단계에서는 스페이스에 연결한 위치에 빌드 역할 (`codecatalyst-eks-build-role`codecatalyst-eks-deploy-role) 과 배포 역할 () 을 추가합니다. AWS 계정 이렇게 하면 워크플로에서 역할을 사용할 수 있습니다.

빌드 및 배포 역할을 사용자 계정에 추가하려면 AWS 계정

1. CodeCatalyst 콘솔에서 스페이스로 이동합니다.
2. 상단에서 설정을 선택합니다.
3. 탐색 창에서 AWS 계정을 선택합니다. 계정 목록이 나타납니다.
4. Amazon CodeCatalyst 디스플레이 이름 열에서 빌드 및 배포 역할을 생성한 AWS 계정 곳의 디스플레이 이름을 복사합니다. (숫자일 수 있습니다.) 이 값은 나중에 워크플로를 만들 때 필요합니다.
5. 표시 이름을 선택합니다.
6. 관리 콘솔에서 역할 AWS 관리를 선택합니다.

   Amazon CodeCatalyst 스페이스에 IAM 역할 추가 페이지가 나타납니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.
7. 에서 생성한 기존 역할 추가를 선택합니다IAM.

드롭다운 목록이 나타납니다. 목록에는 빌드 및 배포 역할과, `codecatalyst-runner.amazonaws.com` 및 `codecatalyst.amazonaws.com` 서비스 IAM 주체를 포함하는 신뢰 정책이 적용된 기타 모든 역할이 표시됩니다.

8. 드롭다운 목록에서 다음을 추가합니다.

   - `codecatalyst-eks-build-role`
   - `codecatalyst-eks-deploy-role`

   > **Note**
   >
   > 표시되는 `The security token included in the request is invalid` 경우 적절한 권한이 없기 때문일 수 있습니다. 이 문제를 해결하려면 CodeCatalyst 스페이스를 만들 때 사용한 AWS 계정으로 로그아웃하고 다시 로그인하세요. AWS

9. CodeCatalyst 콘솔로 돌아가서 페이지를 새로고침하세요.

   이제 빌드 및 배포 역할이 IAM역할 아래에 표시되어야 합니다.

   이제 CodeCatalyst 워크플로에서 이러한 역할을 사용할 수 있습니다.

7단계: 업데이트 ConfigMap

Kubernetes 파일에 생성한 배포 역할을 Kubernetes `ConfigMap` 파일에 추가하여 (워크플로의) Deploy to Kubernetes 클러스터 작업에 클러스터에 액세스하고 클러스터와 상호 작용할 수 있는 기능을 부여해야 합니다. 5단계: AWS 역할 생성 또는 를 사용하여 `eksctl` 이 작업을 수행할 수 있습니다. `kubectl`

eksctl을 사용하여 쿠버네티스 파일을 ConfigMap 구성하려면

- 개발 환경 터미널에서 다음을 입력합니다.

```
eksctl create iamidentitymapping --cluster codecatalyst-eks-cluster --
arn arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role --group
 system:masters --username codecatalyst-eks-deploy-role --region us-west-2
```

위치:

- *`codecatalyst-eks-cluster`* Amazon EKS 클러스터의 클러스터 이름으로 대체됩니다.

- *arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role* 에서 생성한 배포 ARN 역할의 역할로 대체됩니다5단계: AWS 역할 생성.
- *codecatalyst-eks-deploy-role* (옆--username) 은 에서 만든 배포 역할의 이름으로 바꿉니다5단계: AWS 역할 생성.

> **Note**
>
> 배포 역할을 만들지 않기로 결정했다면 다음을 대체하십시오.*codecatalyst-eks-deploy-role* CodeCatalystWorkflowDevelopmentRole-*spaceName*역할 이름과 함께. 이에 대한 자세한 내용은 5단계: AWS 역할 생성 섹션을 참조하세요.

- *us-west-2* 해당 지역으로 대체됩니다.

이 명령에 대한 자세한 내용은 IAM사용자 및 역할 관리를 참조하십시오.

다음과 비슷한 메시지가 나타납니다.

```
2023-06-09 00:58:29 [#]  checking arn arn:aws:iam::111122223333:role/codecatalyst-
eks-deploy-role against entries in the auth ConfigMap
2023-06-09 00:58:29 [#]  adding identity "arn:aws:iam::111122223333:role/
codecatalyst-eks-deploy-role" to auth ConfigMap
```

kubectl을 사용하여 쿠버네티스 파일을 ConfigMap 구성하려면

1. 개발 환경 터미널에서 다음을 입력합니다.

```
kubectl edit configmap -n kube-system aws-auth
```

ConfigMap 파일이 화면에 나타납니다.

2. 빨간색 기울임꼴로 텍스트 추가:

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file
 will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
```

```
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::111122223333:role/eksctl-codecatalyst-eks-cluster-n-
NodeInstanceRole-16BC456ME6YR5
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - system:masters
      rolearn: arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role
      username: codecatalyst-eks-deploy-role
  mapUsers: |
    []
kind: ConfigMap
metadata:
  creationTimestamp: "2023-06-08T19:04:39Z"
  managedFields:
  ...
```

위치:

- *arn:aws:iam::111122223333:role/codecatalyst-eks-deploy-role* 에서 5단계: AWS 역할 생성 생성한 배포 ARN 역할의 역할로 대체됩니다.
- *codecatalyst-eks-deploy-role* (옆username:) 은 에서 만든 배포 역할의 이름으로 바꿉니다5단계: AWS 역할 생성.

> **Note**
>
> 배포 역할을 만들지 않기로 결정했다면 다음을 대체하십시오.*codecatalyst-eks-deploy-role* CodeCatalystWorkflowDevelopmentRole-*spaceName*역할 이름과 함께. 이에 대한 자세한 내용은 5단계: AWS 역할 생성 섹션을 참조하세요.

자세한 내용은 Amazon 사용 EKS 설명서의 클러스터에 대한 IAM 보안 주체 액세스 활성화를 참조하십시오.

이제 쿠버네티스 클러스터에 배포 역할, 나아가 Amazon에 배포 EKS 작업 system:masters 권한을 부여했습니다.

8단계: 워크플로 생성 및 실행

이 단계에서는 소스 파일을 가져와 Docker 이미지로 빌드한 다음 Amazon 클러스터의 트리 포드에 이미지를 배포하는 워크플로를 생성합니다. EKS

워크플로는 순차적으로 실행되는 다음과 같은 구성 블록으로 구성됩니다.

- 트리거 - 이 트리거는 소스 리포지토리에 변경 내용을 푸시할 때 워크플로가 자동으로 실행됩니다. 트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.
- 빌드 작업 (BuildBackend) — 트리거 시 작업은 Dockerfile을 사용하여 Docker 이미지를 빌드하고 Amazon에 이미지를 푸시합니다. ECR 또한 빌드 작업은 deployment.yaml 파일의 $REPOSITORY_URI 및 $IMAGE_TAG 변수를 올바른 값으로 업데이트한 다음 이 파일과 폴더 내 다른 파일의 출력 아티팩트를 생성합니다. Kubernetes 이 자습서에서는 Kubernetes 폴더에 있는 유일한 deployment.yaml 파일이지만 더 많은 파일을 포함할 수 있습니다. 아티팩트는 다음 단계인 배포 작업의 입력으로 사용됩니다.

  빌드 작업에 대한 자세한 내용은 을 참조하십시오워크플로를 사용한 구축.
- 배포 작업 (DeployToEKS) — 빌드 작업이 완료되면 배포 작업은 빌드 작업 (Manifests) 에서 생성된 출력 아티팩트를 찾고 그 안에서 deployment.yaml 파일을 찾습니다. 그러면 액션은 deployment.yaml 파일의 지침에 따라 파드 3개를 실행하는데, 각 파드에는 'Hello, World! '하나가 하나씩 들어 있다. 도커 컨테이너 - Amazon 클러스터 내부. EKS

워크플로 생성 방법

1. 콘솔로 이동합니다. CodeCatalyst
2. 프로젝트 (codecatalyst-eks-project) 로 이동합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로 만들기를 선택합니다.
5. 소스 리포지토리의 경우 선택합니다codecatalyst-eks-source-repository.
6. Branch의 경우 선택하십시오main.
7. 생성(Create)을 선택합니다.
8. YAML샘플 코드를 삭제합니다.
9. 다음 YAML 코드를 추가하여 새 워크플로 정의 파일을 생성합니다.

> **Note**
>
> 워크플로 정의 파일에 대한 자세한 내용은 을 참조하십시오워크플로우 YAML 정의.

> **Note**
>
> 다음 YAML 코드에서는 원하는 경우 Connections: 섹션을 생략할 수 있습니다. 이러한 섹션을 생략하는 경우 환경의 기본 역할 필드에 지정된 역할에 에서 설명한 두 IAM 역할의 권한 및 신뢰 정책이 포함되는지 확인해야 합니다. 6단계: AWS 역할 추가 CodeCatalyst 기본 IAM 역할을 사용하여 환경을 설정하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

```
Name: codecatalyst-eks-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildBackend:
    Identifier: aws/build@v1
    Environment:
      Name: codecatalyst-eks-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-eks-build-role
    Inputs:
      Sources:
        - WorkflowSource
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-
image-repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
    Configuration:
      Steps:
```

```
        #pre_build:
        - Run: echo Logging in to Amazon ECR...
        - Run: aws --version
        - Run: aws ecr get-login-password --region us-west-2 | docker login --
username AWS --password-stdin 111122223333.dkr.ecr.us-west-2.amazonaws.com
        #build:
        - Run: echo Build started on `date`
        - Run: echo Building the Docker image...
        - Run: docker build -t $REPOSITORY_URI:latest .
        - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
        #post_build:
        - Run: echo Build completed on `date`
        - Run: echo Pushing the Docker images...
        - Run: docker push $REPOSITORY_URI:latest
        - Run: docker push $REPOSITORY_URI:$IMAGE_TAG
        # Replace the variables in deployment.yaml
        - Run: find Kubernetes/ -type f | xargs sed -i "s|\$REPOSITORY_URI|
$REPOSITORY_URI|g"
        - Run: find Kubernetes/ -type f | xargs sed -i "s|\$IMAGE_TAG|$IMAGE_TAG|g"
        - Run: cat Kubernetes/*
        # The output artifact will be a zip file that contains Kubernetes manifest
 files.
    Outputs:
      Artifacts:
        - Name: Manifests
          Files:
            - "Kubernetes/*"
  DeployToEKS:
    DependsOn:
      - BuildBackend
    Identifier: aws/kubernetes-deploy@v1
    Environment:
      Name: codecatalyst-eks-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-eks-deploy-role
    Inputs:
      Artifacts:
        - Manifests
    Configuration:
      Namespace: default
      Region: us-west-2
      Cluster: codecatalyst-eks-cluster
```

```
      Manifests: Kubernetes/
```

위 코드에서 다음을 바꾸십시오.

- 두 인스턴스 모두 *codecatalyst-eks-environment* 만든 환경의 이름을 사용하세요사전 조건.
- 두 인스턴스 모두 *codecatalyst-account-connection* 계정 연결의 디스플레이 이름과 함께. 표시 이름은 숫자일 수 있습니다. 자세한 내용은 6단계: AWS 역할 추가 CodeCatalyst 단원을 참조하십시오.
- *codecatalyst-eks-build-role* 에서 만든 빌드 역할의 이름을 사용합니다5단계: AWS 역할 생성.
- *111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-eks-image-repo* (`Value:`속성에) 생성한 Amazon ECR 리포지토리의 정보를 포함합니다3단계: Amazon ECR 이미지 리포지토리 생성. URI
- *111122223333.dkr.ecr.us-west-2.amazonaws.com* (`Run: aws ecr`명령에서) 이미지 접미사 (`/codecatalyst-eks-image-repo`) 가 없는 Amazon ECR 리포지토리를 사용합니다. URI
- *codecatalyst-eks-deploy-role* 에서 생성한 배포 역할의 이름을 사용합니다. 5단계: AWS 역할 생성
- 의 두 인스턴스 모두 *us-west-2* AWS 지역 코드를 사용하세요. 지역 코드 목록은 의 지역 엔드포인트를 참조하십시오. AWS 일반 참조

> **Note**
>
> 빌드 및 배포 역할을 생성하지 않기로 결정했다면 다음을 대체하세요.*codecatalyst-eks-build-role* 그리고 *codecatalyst-eks-deploy-role* `CodeCatalystWorkflowDevelopmentRole-`*spaceName*역할 이름을 입력하세요. 이에 대한 자세한 내용은 5단계: AWS 역할 생성 섹션을 참조하세요.

10. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 YAML 코드가 유효한지 확인합니다.
11. 커밋을 선택합니다.
12. 워크플로 커밋 대화 상자에 다음을 입력합니다.

    a. 커밋 메시지의 경우 텍스트를 제거하고 다음을 입력합니다.

```
Add first workflow
```

b. 리포지토리의 경우 선택합니다`codecatalyst-eks-source-repository`.

c. 브랜치 이름으로 main을 선택합니다.

d. 커밋을 선택합니다.

이제 워크플로가 생성되었습니다. 워크플로 맨 위에 정의된 트리거로 인해 워크플로 실행이 자동으로 시작됩니다. 특히 `workflow.yaml` 파일을 소스 리포지토리에 커밋 (및 푸시) 하면 트리거가 워크플로 실행을 시작했습니다.

워크플로 실행 진행 상황을 보려면

1. CodeCatalyst 콘솔의 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 방금 만든 워크플로를 선택합니다. `codecatalyst-eks-workflow`
3. 빌드 진행 상황을 BuildBackend확인하도록 선택합니다.
4. DeployToEKS선택하여 배포 진행 상황을 확인하세요.

   실행 세부 정보 보기에 대한 자세한 내용은 을 참조하십시오워크플로 실행 상태 및 세부 정보 보기.

배포를 확인하려면

1. 에서 Amazon EC2 콘솔을 엽니다 https://console.aws.amazon.com/ec2/.
2. 왼쪽 하단에서 로드 밸런서를 선택합니다.
3. Kubernetes 배포의 일부로 생성된 로드 밸런서를 선택합니다. 어떤 로드 밸런서를 선택해야 할지 잘 모르겠으면 Tags 탭에서 다음 태그를 찾아보세요.

   - `kubernetes.io/service-name`
   - `kubernetes.io/cluster/ekstutorialcluster`
4. 올바른 로드 밸런서를 선택한 상태에서 설명 탭을 선택합니다.
5. DNS이름 값을 복사하여 브라우저의 주소 표시줄에 붙여넣습니다.

   '헬로, 월드! ' 웹 페이지가 브라우저에 표시되어 애플리케이션을 성공적으로 배포했음을 나타냅니다.

9단계: 소스 파일 변경

이 섹션에서는 소스 리포지토리의 index.html 파일을 변경합니다. 이 변경으로 인해 워크플로우는 새 Docker 이미지를 빌드하고, 커밋 ID로 태그를 지정하고, Amazon으로 푸시하고ECR, ECS Amazon에 배포합니다.

index.html 변경하기

1. 개발 환경으로 이동하세요.
2. 터미널 프롬프트에서 소스 리포지토리로 변경합니다.

```
cd /projects/codecatalyst-eks-source-repository
```

3. 최신 워크플로 변경 내용 가져오기:

```
git pull
```

4. codecatalyst-eks-source-repository/public-html/index.html를 엽니다.
5. 14번째 줄에서 Hello, World! 텍스트를 로 변경합니다Tutorial complete!.
6. 추가, 커밋, 푸시:

```
git add .
git commit -m "update index.html title"
git push
```

   워크플로 실행이 자동으로 시작됩니다.
7. (선택 사항) 다음을 입력합니다.

```
git show HEAD
```

   index.html변경 내용을 위한 커밋 ID를 기록해 둡니다. 이 커밋 ID는 방금 시작한 워크플로 실행에서 배포될 Docker 이미지에 태그가 지정됩니다.
8. 배포 진행 상황 보기:

   a. CodeCatalyst 콘솔의 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
   b. 최근 codecatalyst-eks-workflow 실행을 보도록 선택합니다.
   c. 선택하고 BuildBackend워크플로우 실행 진행 DeployToEKS상황을 확인하세요.
9. 다음과 같이 애플리케이션이 업데이트되었는지 확인하십시오.

a. 에서 Amazon EC2 콘솔을 엽니다 https://console.aws.amazon.com/ec2/.

b. 왼쪽 하단에서 로드 밸런서를 선택합니다.

c. Kubernetes 배포의 일부로 생성된 로드 밸런서를 선택합니다.

d. DNS이름 값을 복사하여 브라우저의 주소 표시줄에 붙여넣습니다.

'튜토리얼 완료!' 브라우저에 웹 페이지가 나타나 애플리케이션의 새 버전을 성공적으로 배포했음을 나타냅니다.

10. (선택 사항) 에서 AWS Amazon ECR 콘솔로 전환하여 이 절차의 7단계에서 새 Docker 이미지에 커밋 ID 태그가 지정되었는지 확인합니다.

정리

이 자습서에서 사용하는 스토리지 및 컴퓨팅 리소스에 대해 불필요하게 요금이 청구되지 않도록 환경을 정리해야 합니다.

정리하려면

1. 클러스터 삭제:

   - 개발 환경 터미널에서 다음을 입력합니다.

     ```
     eksctl delete cluster --region=us-west-2 --name=codecatalyst-eks-cluster
     ```

     위치:

     - *us-west-2* 해당 지역으로 바꿉니다.
     - *codecatalyst-eks-cluster* 생성한 클러스터의 이름으로 대체됩니다.

     5~10분 후 AWS CloudFormation 스택, 노드 그룹 (AmazonEC2), 로드 밸런서를 포함하나 이에 국한되지 않는 클러스터 및 관련 리소스가 삭제됩니다.

> **Important**
>
> `eksctl delete cluster`명령이 작동하지 않는 경우 AWS 자격 증명 또는 자격 증명을 새로 고쳐야 할 수 있습니다. `kubectl` 어떤 자격 증명을 새로 고쳐야 할지 잘 모르겠으면 먼저 AWS 자격 증명을 새로 고치세요. AWS 자격 증명을 새로 고치려면 을 참조하십시

오“자격 증명을 찾을 수 없음” 및 "ExpiredToken" 오류를 해결하려면 어떻게 해야 하나요?. kubectl자격 증명을 새로 고치려면 을 참조하십시오“서버에 연결할 수 없음” 오류를 해결하려면 어떻게 해야 하나요?.

2. AWS 콘솔에서 다음과 같이 정리합니다.
    1. ECRAmazon에서는 삭제하십시오codecatalyst-eks-image-repo.
    2. IAMID 센터에서 삭제:
        a. codecatalyst-eks-user
        b. codecatalyst-eks-permission-set
    3. IAM에서 삭제:
        - codecatalyst-eks-build-role
        - codecatalyst-eks-deploy-role
        - codecatalyst-eks-build-policy
        - codecatalyst-eks-deploy-policy
3. CodeCatalyst 콘솔에서 다음과 같이 정리합니다.
    1. 삭제codecatalyst-eks-workflow.
    2. 삭제codecatalyst-eks-environment.
    3. 삭제codecatalyst-eks-source-repository.
    4. 개발 환경을 삭제하세요.
    5. 삭제codecatalyst-eks-project.

이 자습서에서는 CodeCatalyst 워크플로와 Kubernetes에 배포 클러스터 작업을 사용하여 Amazon EKS 서비스에 애플리케이션을 배포하는 방법을 배웠습니다.

'쿠버네티스 클러스터에 배포' 작업 추가

다음 지침에 따라 Kubernetes 클러스터에 배포 작업을 워크플로에 추가하십시오.

시작하기 전에

Kubernetes 클러스터에 배포 작업을 워크플로에 추가하기 전에 다음을 준비해야 합니다.

> **Tip**
>
> 이러한 사전 요구 사항을 빠르게 설정하려면 의 지침을 따르십시오. 자습서: Amazon에 애플리케이션 배포 EKS

- 아마존의 쿠버네티스 클러스터. EKS 클러스터에 대한 자세한 내용은 Amazon EKS 사용 설명서의 Amazon EKS 클러스터를 참조하십시오.
- 애플리케이션을 Docker 이미지로 어셈블하는 방법을 설명하는 Dockerfile이 하나 이상 있어야 합니다. Dockerfile에 대한 자세한 내용은 Dockerfile 참조를 참조하십시오.
- 쿠버네티스 매니페스트 파일이 하나 이상 있어야 합니다. 이 파일은 쿠버네티스 설명서에서 구성 파일 또는 구성이라고 합니다. 자세한 내용은 쿠버네티스 설명서의 리소스 관리를 참조하십시오.
- Amazon 클러스터에 액세스하고 Amazon 클러스터와 상호 작용할 수 있는 기능을 Kubernetes로 배포 클러스터 작업에 부여하는 IAM 역할입니다. EKS 자세한 내용은 '쿠버네티스 클러스터에 배포' 작업 YAML에서 Role 주제를 참조하십시오.

  이 역할을 생성한 후에는 다음 위치에 추가해야 합니다.

  - 쿠버네티스 파일 ConfigMap . ConfigMap 파일에 역할을 추가하는 방법을 알아보려면 Amazon EKS User Guide의 클러스터에 대한 IAM 보안 주체 액세스 활성화를 참조하십시오.
  - CodeCatalyst. IAM역할을 추가하는 방법을 CodeCatalyst 알아보려면 을 참조하십시오계정 연결에 IAM 역할 추가.
- CodeCatalyst 공간, 프로젝트, 환경. 공간과 환경이 모두 애플리케이션을 배포할 AWS 계정에 연결되어 있어야 합니다. 자세한 내용은 스페이스 만들기, Amazon에서 빈 프로젝트 생성 CodeCatalyst, 및 에 AWS 계정 배포 VPCs 단원을 참조하세요.
- 에서 지원하는 CodeCatalyst 소스 리포지토리. 리포지토리는 애플리케이션 소스 파일, Dockerfile 및 Kubernetes 매니페스트를 저장합니다. 자세한 내용은 소스 리포지토리를 사용하여 코드를 저장하고 공동 작업하십시오. CodeCatalyst 단원을 참조하십시오.

Visual

비주얼 에디터를 사용하여 'Kubernetes 클러스터에 배포' 작업을 추가하려면

1. https://codecatalyst.aws/ 에서 콘솔을 엽니다. CodeCatalyst
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.

4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. Kubernetes 클러스터에 배포 작업을 검색하고 다음 중 하나를 수행하십시오.

   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - Kubernetes 클러스터에 배포를 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 액션의 소스 코드를 보려면 다운로드를 선택합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.
10. 입력 및 구성 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오'쿠버네티스 클러스터에 배포' 작업 YAML. 이 참조는 편집기와 시각적 편집기에 모두 나타나는 각 필드 (및 해당 YAML 속성 값) 에 대한 자세한 정보를 제공합니다. YAML
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

편집기를 사용하여 'Kubernetes 클러스터에 배포' 작업을 추가하려면 YAML

1. https://codecatalyst.aws/ 에서 콘솔을 엽니다. CodeCatalyst
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.

6. 선택합니다 YAML.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. Kubernetes 클러스터에 배포 작업을 검색하고 다음 중 하나를 수행하십시오.

   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - Kubernetes 클러스터에 배포를 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 액션의 소스 코드를 보려면 다운로드를 선택합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.
10. 필요에 따라 YAML 코드의 속성을 수정하십시오. 사용 가능한 각 속성에 대한 설명은 에 나와 '쿠버네티스 클러스터에 배포' 작업 YAML 있습니다.
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

'쿠버네티스 클러스터에 배포' 변수

Kubernetes 클러스터에 배포 작업은 런타임에 다음 변수를 생성하고 설정합니다. 이러한 변수를 사전 정의된 변수라고 합니다.

워크플로우에서 이러한 변수를 참조하는 방법에 대한 자세한 내용은 을 참조하십시오. 사전 정의된 변수 사용

키	값
cluster	워크플로 실행 중에 배포된 Amazon EKS 클러스터의 Amazon.com 리소스 이름 (ARN). 예제: `arn:aws:eks:us-west-2:111122223333:cluster/codecatalyst-eks-cluster`

키	값
배포 플랫폼	배포 플랫폼의 이름. 로 하드코딩했습니다. `AWS:EKS`
metadata	예약. JSON-워크플로우 실행 중에 배포된 클러스터와 관련된 형식의 메타데이터
네임스페이스	클러스터가 배포된 Kubernetes 네임스페이스입니다. 예제: `default`
resources	예약. JSON워크플로 실행 중에 배포된 리소스와 관련된 -형식의 메타데이터
서버	와 같은 관리 도구를 사용하여 클러스터와 통신하는 데 사용할 수 있는 API 서버 엔드포인트의 이름입니다. `kubectl` API서비스 엔드포인트에 대한 자세한 내용은 Amazon EKS 사용 설명서의 Amazon EKS 클러스터 엔드포인트 액세스 제어를 참조하십시오. 예제: `https://random-string .gr7.us-west-2.eks.amazonaws.com`

'쿠버네티스 클러스터에 배포' 작업 YAML

다음은 쿠버네티스 디플로이 클러스터 액션의 YAML 정의입니다. 이 작업을 사용하는 방법을 알아보려면 을 참조하십시오. 워크플로를 EKS 사용하여 Amazon에 배포하기

이 작업 정의는 광범위한 워크플로 정의 파일 내에 섹션으로 존재합니다. 이 파일에 대한 자세한 내용은 워크플로우 YAML 정의을 참조합니다.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

```yaml
# The workflow definition starts here.
# See ### ## for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  DeployToKubernetesCluster_nn:
    Identifier: aws/kubernetes-deploy@v1
    DependsOn:
      - build-action
    Compute:
        - Type: EC2 | Lambda
        - Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: DeployToEKS
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - manifest-artifact
    Configuration:
      Namespace: namespace
      Region: us-east-1
      Cluster: eks-cluster
      Manifests: manifest-path
```

DeployToKubernetesCluster

(필수)

액션의 이름을 지정합니다. 모든 작업 이름은 워크플로 내에서 고유해야 합니다. 액션 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 액션 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

기본값: `DeployToKubernetesCluster_nn`.

해당 UI: 구성 탭/ 작업 표시 이름

Identifier

(*DeployToKubernetesCluster*/Identifier)

(필수)

작업을 식별합니다. 버전을 변경하려는 경우가 아니면 이 속성을 변경하지 마십시오. 자세한 내용은 사용할 액션 버전 지정 단원을 참조하십시오.

기본값: `aws/kubernetes-deploy@v1`.

해당 UI: 워크플로 다이어그램/ DeployToKubernetesCluster _n/ aws/kubernetes-deploy @v1 라벨

DependsOn

(*DeployToKubernetesCluster*/DependsOn)

(선택 사항)

이 작업을 실행하기 위해 성공적으로 실행되어야 하는 작업, 작업 그룹 또는 게이트를 지정하십시오.

'depends on' 기능에 대한 자세한 내용은 을 참조하십시오. 시퀀싱 액션

해당 UI: 입력 탭/ 기준 - 선택 사항

Compute

(*DeployToKubernetesCluster*/Compute)

(선택 사항)

워크플로 작업을 실행하는 데 사용되는 컴퓨팅 엔진. 워크플로 수준 또는 작업 수준에서 컴퓨팅을 지정할 수 있지만 둘 다에서 지정할 수는 없습니다. 워크플로 수준에서 지정된 경우 컴퓨팅 구성은 워크플로에 정의된 모든 작업에 적용됩니다. 워크플로 수준에서는 동일한 인스턴스에서 여러 작업을 실행할 수도 있습니다. 자세한 내용은 작업 간 컴퓨팅 공유 단원을 참조하십시오.

해당 UI: 없음

Type

(*DeployToKubernetesCluster*/Compute/Type)

(포함된 Compute 경우 필수)

컴퓨팅 엔진 유형. 다음 값 중 하나를 사용할 수 있습니다.

- EC2(비주얼 에디터) 또는 EC2 (YAML에디터)

  작업 실행 중 유연성을 위해 최적화되었습니다.
- Lambda (비주얼 에디터) 또는 (에디터) Lambda YAML

  작업 시작 속도를 최적화했습니다.

컴퓨팅 유형에 대한 자세한 정보는 컴퓨팅 유형을 참고하세요.

해당 UI: 구성 탭/고급 - 선택 사항/ 컴퓨팅 유형

Fleet

(*DeployToKubernetesCluster*/Compute/Fleet)

(선택 사항)

워크플로 또는 워크플로 작업을 실행할 시스템 또는 플릿을 지정합니다. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다. 온디맨드 플릿의 예:,. Linux.x86-64.Large Linux.x86-64.XLarge 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

생략된 경우 기본값은 Fleet 입니다. Linux.x86-64.Large

해당 UI: 구성 탭/고급 - 선택 사항/ 컴퓨팅 플릿

Timeout

(*DeployToKubernetesCluster*/Timeout)

(선택 사항)

작업이 종료되기 전에 작업을 실행할 수 있는 시간을 분 (YAML편집기) 또는 시간과 분 (비주얼 편집기) 단위로 지정합니다. CodeCatalyst 최소값은 5분이고 최대값은 에 설명되어 의 워크플로우 할당량 CodeCatalyst 있습니다. 기본 타임아웃은 최대 타임아웃과 동일합니다.

해당 UI: 구성 탭/ 타임아웃 - 선택 사항

Environment

(*DeployToKubernetesCluster*/Environment)

(필수)

작업에 사용할 CodeCatalyst 환경을 지정합니다. 작업은 선택한 환경에 VPC 지정된 AWS 계정 및 선택적 Amazon에 연결됩니다. 작업은 환경에 지정된 기본 IAM 역할을 사용하여 Amazon에 연결하고 Amazon VPC 연결에 지정된 IAM 역할을 사용하여 Amazon에 연결합니다VPC. AWS 계정

> **Note**
>
> 기본 IAM 역할에 작업에 필요한 권한이 없는 경우 다른 역할을 사용하도록 작업을 구성할 수 있습니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.

환경에 대한 자세한 내용은 및 에 AWS 계정 배포 VPCs 및 을 참조하십시오환경 생성.

해당 UI: 구성 탭/ 환경

Name

(*DeployToKubernetesCluster*/Environment/Name)

(포함된 Environment 경우 필수)

작업에 연결할 기존 환경의 이름을 지정합니다.

해당 UI: 구성 탭/ 환경

Connections

(*DeployToKubernetesCluster*/Environment/Connections)

(새 버전의 액션에서는 선택 사항, 이전 버전에서는 필요)

액션과 연결할 계정 연결을 지정합니다. 에서 최대 1개의 계정 연결을 지정할 수 `Environment` 있습니다.

계정 연결을 지정하지 않은 경우:

- 작업은 CodeCatalyst 콘솔 환경에 지정된 AWS 계정 연결 및 기본 IAM 역할을 사용합니다. 환경에 계정 연결 및 기본 IAM 역할을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.
- 기본 IAM 역할에는 작업에 필요한 정책 및 권한이 포함되어야 합니다. 이러한 정책 및 권한이 무엇인지 확인하려면 작업 YAML 정의 설명서에서 역할 속성에 대한 설명을 참조하십시오.

계정 연결에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정. 환경에 계정 연결을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

해당 UI: 액션 버전에 따라 다음 중 하나:

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Name

(*DeployToKubernetesCluster*/Environment/Connections/Name)

(선택 사항)

계정 연결 이름을 지정합니다.

해당 UI: 액션 버전에 따라 다음 중 하나:

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Role

(*DeployToKubernetesCluster*/Environment/Connections/Role)

(포함된 경우 필수Connections)

Deploy to Kubernetes 클러스터 작업이 액세스하는 데 사용하는 IAM 역할의 이름을 지정합니다. AWS CodeCatalyst 스페이스에 역할을 추가했는지, 역할에 다음 정책이 포함되어 있는지 확인하십시오.

IAM역할을 지정하지 않으면 CodeCatalyst 콘솔 환경에 나열된 기본 IAM 역할이 작업에 사용됩니다. 환경에서 기본 역할을 사용하는 경우 해당 역할에 다음 정책이 적용되는지 확인하십시오.

- 다음 권한 정책:

> **Warning**
>
> 권한을 다음 정책에 표시된 권한으로 제한하십시오. 더 광범위한 권한이 있는 역할을 사용하면 보안 위험이 발생할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 역할을 처음 사용할 때는 리소스 정책 설명에 다음 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

- 다음과 같은 사용자 지정 신뢰 정책:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
```

```
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

이 역할이 다음에 추가되었는지 확인하십시오.

- 계정 연결. 계정 연결에 IAM 역할을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오계정 연결에 IAM 역할 추가.
- 내 쿠버네티스 ConfigMap. 에 IAM 역할을 추가하는 방법에 대해 자세히 알아보려면 ConfigMap 설명서의 IAM사용자 및 역할 관리를 참조하십시오. `eksctl`

> **Tip**
>
> 계정 연결에 IAM 역할을 추가하는 방법에 자습서: Amazon에 애플리케이션 배포 EKS 대한 지침은 및 을 참조하십시오 ConfigMap.

> **Note**
>
> 원하는 경우 이 작업에 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 역할을 사용할 수 있습니다. 이에 대한 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기 섹션을 참조하세요. `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*역할에는 보안 위험을 초래할 수 있는 전체 액세스 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다.

해당 UI: 액션 버전에 따라 다음 중 하나가 표시됩니다.

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 역할

Inputs

(*DeployToKubernetesCluster*/Inputs)

(포함된 경우 필수Connections)

이 Inputs 섹션에서는 워크플로우 실행 중에 DeployToKubernetesCluster 필요한 데이터를 정의합니다.

> **Note**
>
> Amazon에 배포 EKS 작업당 하나의 입력 (소스 또는 아티팩트) 만 허용됩니다.

해당 UI: 입력 탭

Sources

(*DeployToKubernetesCluster*/Inputs/Sources)

(매니페스트 파일이 소스 저장소에 저장되어 있는 경우 필수)

Kubernetes 매니페스트 파일이 소스 리포지토리에 저장되어 있는 경우 해당 소스 리포지토리의 레이블을 지정하십시오. 현재 지원되는 레이블은 뿐입니다. WorkflowSource

매니페스트 파일이 소스 리포지토리에 포함되어 있지 않은 경우, 매니페스트 파일은 다른 작업으로 생성된 아티팩트에 있어야 합니다.

소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

해당 UI: 입력 탭/ 소스 - 선택 사항

Artifacts - input

(*DeployToKubernetesCluster*/Inputs/Artifacts)

(매니페스트 파일이 이전 작업의 출력 아티팩트에 저장되어 있는 경우 필수)

Kubernetes 매니페스트 파일 또는 파일이 이전 작업에서 생성된 아티팩트에 포함되어 있는 경우 여기에 해당 아티팩트를 지정하십시오. 매니페스트 파일이 아티팩트에 포함되어 있지 않은 경우 해당 파일은 소스 리포지토리에 있어야 합니다.

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 구성 탭/ 아티팩트 - 선택 사항

Configuration

(*DeployToKubernetesCluster*/Configuration)

(필수)

작업의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭

Namespace

(*DeployToKubernetesCluster*/Configuration/Namespace)

(선택 사항)

쿠버네티스 애플리케이션을 배포할 쿠버네티스 네임스페이스를 지정하십시오. 클러스터에서 네임스페이스를 사용하지 `default` 않는 경우에 사용하십시오. 네임스페이스에 대한 자세한 내용은 Kubernetes 설명서의 Kubernetes 네임스페이스를 사용하여 클러스터를 세분화하기 항목을 참조하십시오.

네임스페이스를 생략하면 값이 사용됩니다. `default`

해당 UI: 구성 탭/ 네임스페이스

Region

(*DeployToKubernetesCluster*/Configuration/Region)

(필수)

Amazon EKS 클러스터 및 서비스가 위치한 AWS 지역을 지정합니다. 지역 코드 목록은 의 지역 엔드포인트를 참조하십시오. AWS 일반 참조

해당 UI: 구성 탭/ 지역

Cluster

(*DeployToKubernetesCluster*/Configuration/Cluster)

(필수)

기존 Amazon EKS 클러스터의 이름을 지정합니다. Kubernetes 클러스터에 배포 작업은 컨테이너식 애플리케이션을 이 클러스터에 배포합니다. Amazon EKS 클러스터에 대한 자세한 내용은 Amazon EKS사용 설명서의 클러스터를 참조하십시오.

해당 UI: 구성 탭/ 클러스터

Manifests

(*DeployToKubernetesCluster*/Configuration/Manifests)

(필수)

YAML-formatted Kubernetes 매니페스트 파일의 경로를 지정하십시오. 이 파일은 Kubernetes 설명서에서 구성 파일, 구성 파일 또는 간단히 구성이라고 합니다.

매니페스트 파일을 여러 개 사용하는 경우 매니페스트 파일을 단일 폴더에 배치하고 해당 폴더를 참조하십시오. 매니페스트 파일은 Kubernetes에서 영숫자순으로 처리되므로 처리 순서를 제어하려면 파일 이름 앞에 숫자나 문자를 늘리는 접두사를 붙여야 합니다. 예:

```
00-namespace.yaml
```

```
01-deployment.yaml
```

매니페스트 파일이 소스 리포지토리에 있는 경우 경로는 소스 리포지토리 루트 폴더를 기준으로 합니다. 파일이 이전 워크플로 작업의 아티팩트에 있는 경우 경로는 아티팩트 루트 폴더를 기준으로 합니다.

예시:

```
Manifests/
```

```
deployment.yaml
```

```
my-deployment.yml
```

와일드카드 () 는 사용하지 마십시오. *

> **Note**
>
> 헬름 차트와 사용자 지정 파일은 지원되지 않습니다.

매니페스트 파일에 대한 자세한 내용은 Kubernetes 설명서의 리소스 구성 구성을 참조하십시오.

해당 UI: 구성 탭/ 매니페스트

스택 배포 AWS CloudFormation

이 섹션에서는 워크플로를 사용하여 AWS CloudFormation 스택을 배포하는 방법을 설명합니다. CodeCatalyst 이렇게 하려면 워크플로에 AWS CloudFormation 스택 배포 작업을 추가해야 합니다. 이 작업은 사용자가 제공한 템플릿을 AWS 기반으로 리소스 CloudFormation 스택을 배포합니다. 템플릿은 다음과 같을 수 있습니다.

- AWS CloudFormation 템플릿 — 자세한 내용은 AWS CloudFormation 템플릿 작업을 참조하십시오.
- AWS SAM 템플릿 — 자세한 내용은 AWS Serverless Application Model (AWS SAM) 사양을 참조하십시오.

> **Note**
>
> AWS SAM 템플릿을 사용하려면 먼저 `sam package` 작업을 사용하여 AWS SAM 애플리케이션을 패키징해야 합니다. Amazon CodeCatalyst 워크플로의 일부로 이 패키징을 자동으로 수행하는 방법을 보여주는 자습서는 을 참조하십시오자습서: 서버리스 애플리케이션 배포.

스택이 이미 있는 경우 작업을 실행한 다음 `ExecuteChangeSet` 작업을 실행합니다. CloudFormation `CreateChangeSet` 그러면 액션은 변경 내용이 배포될 때까지 기다린 후 결과에 따라 스스로를 성공 또는 실패로 표시합니다.

배포하려는 리소스가 포함된 또는 템플릿이 이미 있거나 AWS CloudFormation 또는 AWS SAM 템플릿이 있거나 AWS SAM 및 AWS Cloud Development Kit (AWS CDK)같은 도구를 사용하여 워크플로 빌드 작업의 일부로 자동으로 생성하려는 경우 AWS CloudFormation 스택 배포 작업을 사용하십시오.

작성할 수 있는 CloudFormation 템플릿이나 Deploy AWS CloudFormation Stack 작업에 사용할 AWS SAM 수 있는 템플릿 등 템플릿에는 제한이 없습니다.

> **Tip**
>
> Deploy AWS CloudFormation stack 작업을 사용하여 서버리스 애플리케이션을 배포하는 방법을 보여주는 자습서는 을 참조하십시오. 자습서: 서버리스 애플리케이션 배포

주제

- 자습서: 서버리스 애플리케이션 배포

- AWS CloudFormation '스택 배포' 작업 추가
- 롤백 구성
- '디플로이 AWS CloudFormation 스택' 변수
- AWS CloudFormation '스택 배포' 작업 YAML

자습서: 서버리스 애플리케이션 배포

이 자습서에서는 워크플로를 사용하여 서버리스 애플리케이션을 CloudFormation 스택으로 구축, 테스트 및 배포하는 방법을 알아봅니다.

이 자습서의 애플리케이션은 'Hello World' 메시지를 출력하는 간단한 웹 애플리케이션입니다. AWS Lambda 함수와 Amazon API Gateway로 구성되어 있으며, 확장인 AWS Serverless Application Model (AWS SAM) 를 사용하여 AWS CloudFormation빌드합니다.

주제

- 사전 조건
- 1단계: 소스 리포지토리 만들기
- 2단계: AWS 역할 생성
- 3단계: AWS 역할 추가 CodeCatalyst
- 4단계: Amazon S3 버킷 생성
- 5단계: 소스 파일 추가
- 6단계: 워크플로우 생성 및 실행
- 7단계: 변경
- 정리

사전 조건

시작하기 전:

- AWS 계정이 연결된 CodeCatalyst 공간이 필요합니다. 자세한 내용은 스페이스 만들기 단원을 참조하십시오.
- 스페이스에 다음과 같은 빈 프로젝트가 필요합니다.

```
codecatalyst-cfn-project
```

처음부터 시작 옵션을 사용하여 이 프로젝트를 만들 수 있습니다.

자세한 내용은 Amazon에서 빈 프로젝트 생성 CodeCatalyst 단원을 참조하십시오.

- 프로젝트에는 다음과 같은 CodeCatalyst 환경이 필요합니다.

```
codecatalyst-cfn-environment
```

이 환경을 다음과 같이 구성하십시오.

- 원하는 유형 (예: 비프로덕션) 을 선택합니다.
- AWS 계정을 여기에 연결하십시오.
- 기본 IAM 역할의 경우 원하는 역할을 선택합니다. 나중에 다른 역할을 지정하게 됩니다.

자세한 내용은 및 에 AWS 계정 배포 VPCs 단원을 참조하십시오.

1단계: 소스 리포지토리 만들기

이 단계에서는 에서 소스 리포지토리를 생성합니다 CodeCatalyst. 이 리포지토리는 Lambda 함수 파일과 같은 자습서의 소스 파일을 저장하는 데 사용됩니다.

소스 리포지토리에 대한 자세한 내용은 을 참조하십시오. 소스 리포지토리 생성

소스 리포지토리를 생성하려면

1. 의 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다. CodeCatalyst
2. 리포지토리 추가를 선택하고 리포지토리 생성을 선택합니다.
3. 리포지토리 이름에 다음을 입력합니다.

```
codecatalyst-cfn-source-repository
```

4. 생성(Create)을 선택합니다.

라는 리포지토리가 생성되었습니다`codecatalyst-cfn-source-repository`.

2단계: AWS 역할 생성

이 단계에서는 다음과 같은 AWS IAM 역할을 생성합니다.

- 배포 역할 - 서버리스 CodeCatalyst 애플리케이션을 배포할 AWS 계정 및 CloudFormation 서비스에 액세스할 수 있는 배포 AWS CloudFormation 스택 작업 권한을 부여합니다. AWS CloudFormation 스택 배포 작업은 워크플로의 일부입니다.
- 빌드 역할 — CodeCatalyst 빌드 작업에 AWS 계정에 액세스하고 서버리스 애플리케이션 패키지가 저장되는 Amazon S3에 쓸 수 있는 권한을 부여합니다. 빌드 작업은 워크플로의 일부입니다.
- 스택 역할 - 나중에 제공할 AWS SAM 템플릿에 지정된 리소스를 읽고 수정할 수 있는 CloudFormation 권한을 부여합니다. 또한 에 권한을 CloudWatch 부여합니다.

IAM역할에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM 역할을 참조하십시오.

> **Note**
>
> 시간을 절약하기 위해 앞서 나열한 세 가지 `CodeCatalystWorkflowDevelopmentRole-spaceName` 역할 대신 역할 이라는 단일 역할을 만들 수 있습니다. 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-**spaceName**역할 만들기 단원을 참조하십시오. `CodeCatalystWorkflowDevelopmentRole-spaceName`역할에는 보안 위험을 초래할 수 있는 매우 광범위한 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다. 이 자습서에서는 앞서 나열한 세 가지 역할을 생성한다고 가정합니다.

> **Note**
>
> Lambda 실행 역할도 필요하지만 5단계에서 워크플로를 실행할 때 파일이 자동으로 생성되므로 `sam-template.yml` 지금 생성할 필요가 없습니다.

배포 역할을 생성하려면

1. 다음과 같이 역할에 대한 정책을 생성합니다.
   a. 로그인하세요 AWS.
   b. 에서 IAM 콘솔을 엽니다 https://console.aws.amazon.com/iam/.
   c. 탐색 창에서 정책을 선택합니다.

d. 정책 생성을 선택합니다.

e. JSON탭을 선택합니다.

f. 기존 코드를 삭제합니다.

g. 다음 코드를 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "cloudformation:List*",
        "iam:PassRole"
    ],
    "Resource": "*",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름으로 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

h. 다음: 태그를 선택합니다.

i. 다음: 검토를 선택합니다.

j. 이름에 다음을 입력합니다.

```
codecatalyst-deploy-policy
```

k. 정책 생성을 선택합니다.

이제 권한 정책을 생성했습니다.

2. 다음과 같이 배포 역할을 생성합니다.

a. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.

b. 사용자 지정 신뢰 정책을 선택합니다.

c. 기존 사용자 지정 신뢰 정책을 삭제합니다.

d. 다음 사용자 지정 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. Next(다음)를 선택합니다.

f. 권한 정책에서 해당 확인란을 `codecatalyst-deploy-policy` 검색하여 선택합니다.

g. Next(다음)를 선택합니다.

h. 역할 이름에 다음을 입력합니다.

```
codecatalyst-deploy-role
```

i. 역할 설명에 다음을 입력합니다.

```
CodeCatalyst deploy role
```

j. 역할 생성을 선택합니다.

이제 신뢰 정책 및 권한 정책을 사용하여 배포 역할을 생성했습니다.

3. 다음과 같이 배포 역할을 ARN 확보하십시오.

   a. 탐색 창에서 역할을 선택합니다.

   b. 검색 상자에 방금 생성한 역할의 이름을 입력합니다 (`codecatalyst-deploy-role`).

   c. 목록에서 역할을 선택합니다.

      역할의 요약 페이지가 나타납니다.

   d. 상단에서 ARN값을 복사합니다.

   이제 적절한 권한이 있는 배포 역할을 생성하고 해당 역할을 획득했습니다ARN.

빌드 역할을 만들려면

1. 다음과 같이 역할에 대한 정책을 생성합니다.

   a. 로그인하세요 AWS.

   b. 에서 IAM 콘솔을 엽니다 https://console.aws.amazon.com/iam/.

   c. 탐색 창에서 정책을 선택합니다.

   d. 정책 생성을 선택합니다.

   e. JSON탭을 선택합니다.

   f. 기존 코드를 삭제합니다.

   g. 다음 코드를 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "s3:PutObject",
        "iam:PassRole"
    ],
    "Resource": "*",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름으로 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

h. 다음: 태그를 선택합니다.

i. 다음: 검토를 선택합니다.

j. 이름에 다음을 입력합니다.

```
codecatalyst-build-policy
```

k. 정책 생성을 선택합니다.

이제 권한 정책을 생성했습니다.

2. 다음과 같이 빌드 역할을 생성합니다.

a. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.

b. 사용자 지정 신뢰 정책을 선택합니다.

c. 기존 사용자 지정 신뢰 정책을 삭제합니다.

d. 다음 사용자 지정 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

```
}
```

e. Next(다음)를 선택합니다.

f. 권한 정책에서 해당 확인란을 codecatalyst-build-policy 검색하여 선택합니다.

g. Next(다음)를 선택합니다.

h. 역할 이름에 다음을 입력합니다.

```
codecatalyst-build-role
```

i. 역할 설명에 다음을 입력합니다.

```
CodeCatalyst build role
```

j. 역할 생성을 선택합니다.

이제 신뢰 정책 및 권한 정책을 사용하여 빌드 역할을 생성했습니다.

3. 다음과 같이 빌드 역할을 ARN 확보하세요.

a. 탐색 창에서 역할을 선택합니다.

b. 검색 상자에 방금 생성한 역할의 이름을 입력합니다 (codecatalyst-build-role).

c. 목록에서 역할을 선택합니다.

역할의 요약 페이지가 나타납니다.

d. 상단에서 ARN값을 복사합니다.

이제 적절한 권한이 있는 빌드 역할을 생성하고 해당 역할을 획득했습니다ARN.

스택 역할을 만들려면

1. 스택을 배포하려는 계정을 AWS 사용하여 로그인합니다.
2. 에서 IAM 콘솔을 엽니다 https://console.aws.amazon.com/iam/.
3. 다음과 같이 스택 역할을 생성합니다.

a. 탐색 창에서 역할을 선택합니다.

b. Create role(역할 생성)을 선택합니다.

c. AWS 서비스를 선택합니다.

d. 사용 사례 섹션의 드롭다운 목록에서 선택합니다. CloudFormation

e. CloudFormation라디오 버튼을 선택합니다.

f. 하단에서 다음을 선택합니다.

g. 검색 상자를 사용하여 다음 권한 정책을 찾은 다음 해당 확인란을 선택합니다.

> **Note**
> 정책을 검색했는데 표시되지 않는 경우 필터 지우기를 선택하고 다시 시도하세요.

- CloudWatchFullAccess
- AWS CloudFormationFullAccess
- IAMFullAccess
- AWS Lambda_ FullAccess
- 관리자 mazonAPIGateway
- 아마존 S3 FullAccess
- 아마존 EC2ContainerRegistryFullAccess

첫 번째 정책은 경보 발생 시 CloudWatch 액세스를 허용하여 스택 롤백을 활성화합니다.

나머지 정책으로는 AWS SAM 이 자습서에서 배포할 스택의 서비스와 리소스에 액세스할 수 있습니다. 자세한 내용은 AWS Serverless Application Model 개발자 안내서의 권한을 참조하십시오.

h. Next(다음)를 선택합니다.

i. 역할 이름에 다음을 입력합니다.

```
codecatalyst-stack-role
```

j. 역할 생성을 선택합니다.

4. 다음과 ARN 같이 스택 역할을 가져옵니다.

a. 탐색 창에서 역할을 선택합니다.

b. 검색 상자에 방금 생성한 역할의 이름을 입력합니다 (`codecatalyst-stack-role`).

c. 목록에서 역할을 선택합니다.

d. 요약 섹션에서 ARN값을 복사합니다. 나중에 필요합니다.

이제 적절한 권한이 있는 스택 역할을 생성하고 해당 스택 역할을 획득했습니다ARN.

3단계: AWS 역할 추가 CodeCatalyst

이 단계에서는 스페이스의 CodeCatalyst 계정 연결에 빌드 역할 (`codecatalyst-build-rolecodecatalyst-deploy-role`) 과 배포 역할 () 을 추가합니다.

> **Note**
>
> 연결에 스택 역할 (`codecatalyst-stack-role`) 을 추가할 필요가 없습니다. 이는 배포 역할 간에 CodeCatalyst 이미 연결이 설정된 후 배포 역할을 사용하지 않는 CodeCatalyst 사람이 스택 역할을 AWS 사용하기 때문입니다. CloudFormation 는 액세스 권한을 얻는 CodeCatalyst 데 스택 역할을 사용하지 않으므로 계정 연결에 연결할 필요가 없습니다. AWS

계정 연결에 빌드 및 배포 역할을 추가하려면

1. CodeCatalyst에서 스페이스로 이동합니다.
2. AWS 계정을 선택하세요. 계정 연결 목록이 나타납니다.
3. 빌드 및 배포 역할을 생성한 AWS 계정을 나타내는 계정 연결을 선택합니다.
4. 관리 콘솔에서 역할 AWS 관리를 선택합니다.

   Amazon CodeCatalyst 스페이스에 IAM 역할 추가 페이지가 나타납니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.
5. 에서 생성한 기존 역할 추가를 선택합니다IAM.

   드롭다운 목록이 나타납니다. 목록에는 `codecatalyst-runner.amazonaws.com` 및 `codecatalyst.amazonaws.com` 서비스 주체를 포함하는 신뢰 정책이 적용된 모든 IAM 역할이 표시됩니다.
6. 드롭다운 목록에서 선택하고 역할 `codecatalyst-build-role` 추가를 선택합니다.
7. IAM역할 추가를 선택하고 에서 IAM 만든 기존 역할 추가를 선택한 다음 드롭다운 목록에서 선택합니다`codecatalyst-deploy-role`. [Add role]을 선택합니다.

   이제 스페이스에 빌드 및 배포 역할을 추가했습니다.

8. Amazon CodeCatalyst 디스플레이 이름의 값을 복사합니다. 이 값은 나중에 워크플로를 만들 때 필요합니다.

4단계: Amazon S3 버킷 생성

이 단계에서는 서버리스 애플리케이션의 배포 패키지 .zip 파일을 저장하는 Amazon S3 버킷을 생성합니다.

Amazon S3 버킷을 생성하려면

1. 에서 Amazon S3 콘솔을 엽니다 https://console.aws.amazon.com/s3/.
2. 기본 창에서 버킷 생성을 선택합니다.
3. 버킷 이름에 다음을 입력합니다.

```
codecatalyst-cfn-s3-bucket
```

4. AWS 지역에서 지역을 선택합니다. 이 자습서에서는 사용자가 미국 서부 (오레곤) us-west-2를 선택했다고 가정합니다. Amazon S3에서 지원하는 지역에 대한 자세한 내용은 의 Amazon 심플 스토리지 서비스 엔드포인트 및 할당량을 참조하십시오. AWS 일반 참조
5. 페이지 하단에서 버킷 생성을 선택합니다.

이제 미국 서부 (오레곤) **codecatalyst-cfn-s3-bucket** us-west-2 지역에 버킷을 생성했습니다.

5단계: 소스 파일 추가

이 단계에서는 소스 리포지토리에 여러 애플리케이션 CodeCatalyst 소스 파일을 추가합니다. `hello-world`폴더에는 배포할 애플리케이션 파일이 들어 있습니다. `tests`폴더에는 단위 테스트가 들어 있습니다. 폴더 구조는 다음과 같습니다.

```
.
|- hello-world
|   |- tests
|      |- unit
|         |- test-handler.js
|   |- app.js
|- .npmignore
|- package.json
|- sam-template.yml
|- setup-sam.sh
```

.npmignore 파일

.npmignore파일은 npm이 애플리케이션 패키지에서 제외해야 하는 파일 및 폴더를 나타냅니다. 이 자습서에서 npm은 응용 프로그램의 일부가 아니므로 tests 폴더를 제외합니다.

.npmignore 파일을 추가하려면

1. https://codecatalyst.aws/ 에서 콘솔을 엽니다. CodeCatalyst
2. 프로젝트를 선택하고, codecatalyst-cfn-project
3. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
4. 소스 리포지토리 목록에서 리포지토리를 선택합니다. codecatalyst-cfn-source-repository
5. 파일에서 파일 생성을 선택합니다.
6. 파일 이름에 다음을 입력합니다.

```
.npmignore
```

7. 텍스트 상자에 다음 코드를 입력합니다.

```
tests/*
```

8. [커밋] 을 선택한 다음 [커밋] 을 다시 선택합니다.

   이제 리포지토리의 .npmignore 루트에 라는 파일이 생성되었습니다.

패키지.json 파일

이 package.json 파일에는 프로젝트 이름, 버전 번호, 설명, 종속성 및 애플리케이션과 상호 작용하고 실행하는 방법을 설명하는 기타 세부 정보와 같은 Node 프로젝트에 대한 중요한 메타데이터가 포함되어 있습니다.

이 자습서에는 종속성 목록과 스크립트가 포함되어 있습니다test. package.json 테스트 스크립트는 다음을 수행합니다.

- 테스트 스크립트는 mocha를 사용하여 지정된 단위 테스트를 실행하고 xunit 리포터를 사용하여 결과를 junit.xml 파일에 기록합니다. hello-world/tests/unit/
- 테스트 스크립트는 이스탄불 (nyc) 을 사용하여 클로버 리포터를 사용하여 코드 커버리지 보고서 (clover.xml) 를 생성합니다. 자세한 내용은 이스탄불 설명서의 대체 리포터 사용을 참조하십시오.

package.json 파일을 추가하려면

1. 리포지토리의 파일에서 파일 생성을 선택합니다.
2. 파일 이름에 다음을 입력합니다.

   package.json

3. 텍스트 상자에 다음 코드를 입력합니다.

```
{
  "name": "hello_world",
  "version": "1.0.0",
  "description": "hello world sample for NodeJS",
  "main": "app.js",
  "repository": "https://github.com/awslabs/aws-sam-cli/tree/develop/samcli/local/
init/templates/cookiecutter-aws-sam-hello-nodejs",
  "author": "SAM CLI",
  "license": "MIT",
  "dependencies": {
    "axios": "^0.21.1",
    "nyc": "^15.1.0"
  },
  "scripts": {
    "test": "nyc --reporter=clover mocha hello-world/tests/unit/ --reporter xunit
 --reporter-option output=junit.xml"
  },
  "devDependencies": {
    "aws-sdk": "^2.815.0",
    "chai": "^4.2.0",
    "mocha": "^8.2.1"
  }
}
```

4. [커밋] 을 선택한 다음 [커밋] 을 다시 선택합니다.

   이제 저장소의 `package.json` 루트에 라는 파일을 추가했습니다.

sam-template.yml 파일

이 `sam-template.yml` 파일에는 Lambda 함수와 API Gateway를 배포하고 함께 구성하기 위한 지침이 들어 있습니다. 이는 템플릿 사양을 따르며, 이는 AWS Serverless Application Model 템플릿 사양을 확장합니다. AWS CloudFormation

이 자습서에서는 일반 AWS SAM AWS CloudFormation 템플릿 대신 템플릿을 사용합니다. 유용한 AWS: :Serverless: :Function 리소스 유형을 AWS SAM 제공하기 때문입니다. 이 유형은 일반적으로 기본 구문을 사용하기 위해 작성해야 하는 많은 behind-the-scenes 구성을 수행합니다. CloudFormation 예를 들어, 는 Lambda 함수, Lambda 실행 역할 및 함수를 시작하는 이벤트 소스 매핑을 `AWS::Serverless::Function` 생성합니다. basic을 사용하여 작성하려면 이 모든 것을 코딩해야 합니다. CloudFormation

이 자습서에서는 미리 작성된 템플릿을 사용하지만 빌드 작업을 사용하여 워크플로의 일부로 템플릿을 생성할 수 있습니다. 자세한 내용은 스택 배포 AWS CloudFormation 단원을 참조하십시오.

sam-template.yml 파일을 추가하려면

1. 리포지토리의 파일에서 파일 생성을 선택합니다.
2. 파일 이름에 다음을 입력합니다.

   sam-template.yml

3. 텍스트 상자에 다음 코드를 입력합니다.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  serverless-api

  Sample SAM Template for serverless-api

# More info about Globals: https://github.com/awslabs/serverless-application-model/
blob/master/docs/globals.rst
Globals:
  Function:
    Timeout: 3

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # For details on this resource type,
 see https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: hello-world/
      Handler: app.lambdaHandler
      Runtime: nodejs12.x
      Events:
```

```
        HelloWorld:
          Type: Api # For details on this event source type, see
 https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#api
          Properties:
            Path: /hello
            Method: get

Outputs:
  # ServerlessRestApi is an implicit API created out of the events key under
 Serverless::Function
  # Find out about other implicit resources you can reference within AWS SAM at
  # https://github.com/awslabs/serverless-application-model/blob/master/docs/
internals/generated_resources.rst#api
  HelloWorldApi:
    Description: "API Gateway endpoint URL for the Hello World function"
    Value: !Sub "https://${ServerlessRestApi}.execute-api.
${AWS::Region}.amazonaws.com/Prod/hello/"
  HelloWorldFunction:
    Description: "Hello World Lambda function ARN"
    Value: !GetAtt HelloWorldFunction.Arn
  HelloWorldFunctionIamRole:
    Description: "Implicit Lambda execution role created for the Hello World
 function"
    Value: !GetAtt HelloWorldFunctionRole.Arn
```

4. [커밋] 을 선택한 다음 [커밋] 을 다시 선택합니다.

   이제 저장소의 루트 폴더 `sam-template.yml` 아래에 라는 파일을 추가했습니다.

setup-sam.sh 파일

이 `setup-sam.sh` 파일에는 AWS SAM CLI 유틸리티 다운로드 및 설치 지침이 들어 있습니다. 워크플로는 이 유틸리티를 사용하여 `hello-world` 소스를 패키징합니다.

setup-sam.sh 파일을 추가하려면

1. 리포지토리의 파일에서 파일 생성을 선택합니다.
2. 파일 이름에 다음을 입력합니다.

```
setup-sam.sh
```

3. 텍스트 상자에 다음 코드를 입력합니다.

```
#!/usr/bin/env bash
echo "Setting up sam"

yum install unzip -y

curl -LO https://github.com/aws/aws-sam-cli/releases/latest/download/aws-sam-cli-
linux-x86_64.zip
unzip -qq aws-sam-cli-linux-x86_64.zip -d sam-installation-directory

./sam-installation-directory/install; export AWS_DEFAULT_REGION=us-west-2
```

위 코드에서 다음을 대체하십시오.*us-west-2* 해당 AWS 지역으로.

4. 커밋을 선택한 다음 커밋을 다시 선택합니다.

이제 저장소의 `setup-sam.sh` 루트에 라는 파일을 추가했습니다.

app.js 파일

`app.js`에는 Lambda 함수 코드가 들어 있습니다. 이 자습서에서는 코드가 텍스트를 반환합니다. `hello world`

app.js 파일을 추가하려면

1. 리포지토리의 파일에서 파일 생성을 선택합니다.
2. 파일 이름에 다음을 입력합니다.

```
hello-world/app.js
```

3. 텍스트 상자에 다음 코드를 입력합니다.

```
// const axios = require('axios')
// const url = 'http://checkip.amazonaws.com/';
let response;

/**
 *
 * Event doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
lambda-proxy-integrations.html#api-gateway-simple-proxy-for-lambda-input-format
 * @param {Object} event - API Gateway Lambda Proxy Input Format
```

```
 *
 * Context doc: https://docs.aws.amazon.com/lambda/latest/dg/nodejs-prog-model-
context.html
 * @param {Object} context
 *
 * Return doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-
lambda-proxy-integrations.html
 * @returns {Object} object - API Gateway Lambda Proxy Output Format
 *
 */
exports.lambdaHandler = async (event, context) => {
    try {
        // const ret = await axios(url);
        response = {
            'statusCode': 200,
            'body': JSON.stringify({
                message: 'hello world',
                // location: ret.data.trim()
            })
        }
    } catch (err) {
        console.log(err);
        return err;
    }

    return response
};
```

4. [커밋] 을 선택한 다음 [커밋] 을 다시 선택합니다.

   라는 hello-world 폴더와 라는 파일이 생성되었습니다app.js.

test-handler.js 파일

이 test-handler.js 파일에는 Lambda 함수에 대한 단위 테스트가 포함되어 있습니다.

test-handler.js 파일을 추가하려면

1. 리포지토리의 파일에서 파일 생성을 선택합니다.
2. 파일 이름에 다음을 입력합니다.

   hello-world/tests/unit/test-handler.js

3. 텍스트 상자에 다음 코드를 입력합니다.

```
'use strict';

const app = require('../../app.js');
const chai = require('chai');
const expect = chai.expect;
var event, context;

describe('Tests index', function () {
    it('verifies successful response', async () => {
        const result = await app.lambdaHandler(event, context)

        expect(result).to.be.an('object');
        expect(result.statusCode).to.equal(200);
        expect(result.body).to.be.an('string');

        let response = JSON.parse(result.body);

        expect(response).to.be.an('object');
        expect(response.message).to.be.equal("hello world");
        // expect(response.location).to.be.an("string");
    });
});
```

4. [커밋] 을 선택한 다음 [커밋] 을 다시 선택합니다.

   이제 `hello-world/tests/unit` 폴더 `test-handler.js` 아래에 라는 파일을 추가했습니다.

이제 모든 소스 파일을 추가했습니다.

잠시 시간을 내어 작업을 다시 한 번 확인하고 모든 파일을 올바른 폴더에 배치했는지 확인하세요. 폴더 구조는 다음과 같습니다.

```
.
|— hello-world
|  |— tests
|     |— unit
|        |— test-handler.js
|  |— app.js
|— .npmignore
|— README.md
|— package.json
```

```
|— sam-template.yml
|— setup-sam.sh
```

6단계: 워크플로우 생성 및 실행

이 단계에서는 Lambda 소스 코드를 패키징하고 배포하는 워크플로를 생성합니다. 워크플로는 순차적으로 실행되는 다음과 같은 구성 요소로 구성됩니다.

- 트리거 - 이 트리거는 소스 리포지토리에 변경 내용을 푸시할 때 워크플로가 자동으로 실행됩니다. 트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.
- 테스트 작업 (Test) — 트리거 시 이 작업은 노드 패키지 관리자 (npm) 를 설치한 다음 명령을 실행합니다. `npm run test` 이 명령은 npm에게 파일에 정의된 `test` 스크립트를 실행하도록 지시합니다. `package.json` `test`스크립트는 차례로 단위 테스트를 실행하고 테스트 보고서 () 와 코드 커버리지 보고서 (`junit.xml`) 라는 두 개의 보고서를 생성합니다. `clover.xml` 자세한 내용은 패키지.json 파일 단원을 참조하십시오.

  그런 다음 테스트 동작은 XML 보고서를 CodeCatalyst 보고서로 변환하여 CodeCatalyst 콘솔의 테스트 동작의 보고서 탭 아래에 표시합니다.

  테스트 작업에 대한 자세한 내용은 을 참조하십시오워크플로를 사용한 테스트.
- 빌드 작업 (BuildBackend) — 테스트 작업이 완료되면 빌드 작업은 패키지를 다운로드하여 설치하고, `hello-world` 소스를 패키징하고 AWS SAM CLI, Lambda 서비스가 예상하는 Amazon S3 버킷에 패키지를 복사합니다. 또한 이 작업은 라는 새 AWS SAM 템플릿 파일을 `sam-template-packaged.yml` 출력하여 라는 출력 아티팩트에 배치합니다. `buildArtifact`

  빌드 작업에 대한 자세한 내용은 을 참조하십시오워크플로를 사용한 구축.
- 배포 작업 (DeployCloudFormationStack) — 빌드 작업이 완료되면 배포 작업은 빌드 작업 (`buildArtifact`) 에서 생성된 출력 아티팩트를 찾아 그 안에서 AWS SAM 템플릿을 찾은 다음 템플릿을 실행합니다. AWS SAM 템플릿은 서버리스 애플리케이션을 배포하는 스택을 생성합니다.

워크플로 생성 방법

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 워크플로 만들기를 선택합니다.
3. 소스 리포지토리의 경우 선택합니다`codecatalyst-cfn-source-repository`.
4. Branch의 경우 선택하십시오`main`.

5. 생성(Create)을 선택합니다.
6. YAML샘플 코드를 삭제합니다.
7. 다음 YAML 코드를 추가합니다.

> **Note**
>
> 다음 YAML 코드에서는 원하는 경우 `Connections:` 섹션을 생략할 수 있습니다. 이러한 섹션을 생략하는 경우 환경의 기본 역할 필드에 지정된 역할에 에서 설명한 두 IAM 역할의 권한 및 신뢰 정책이 포함되는지 확인해야 합니다. 2단계: AWS 역할 생성 기본 IAM 역할을 사용하여 환경을 설정하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

```
Name: codecatalyst-cfn-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  Test:
    Identifier: aws/managed-test@v1
    Inputs:
      Sources:
        - WorkflowSource
    Outputs:
      Reports:
        CoverageReport:
          Format: CLOVERXML
          IncludePaths:
            - "coverage/*"
        TestReport:
          Format: JUNITXML
          IncludePaths:
            - junit.xml
    Configuration:
      Steps:
        - Run: npm install
        - Run: npm run test
  BuildBackend:
    Identifier: aws/build@v1
```

```
    DependsOn:
      - Test
    Environment:
      Name: codecatalyst-cfn-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-build-role
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: . ./setup-sam.sh
        - Run: sam package --template-file sam-template.yml --s3-
bucket codecatalyst-cfn-s3-bucket --output-template-file sam-template-packaged.yml
 --region us-west-2
    Outputs:
      Artifacts:
        - Name: buildArtifact
          Files:
            - "**/*"
  DeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    DependsOn:
      - BuildBackend
    Environment:
      Name: codecatalyst-cfn-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-deploy-role
    Inputs:
      Artifacts:
        - buildArtifact
      Sources: []
    Configuration:
      name: codecatalyst-cfn-stack
      region: us-west-2
      role-arn: arn:aws:iam::111122223333:role/StackRole
      template: ./sam-template-packaged.yml
      capabilities: CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND
```

위 코드에서 다음을 바꾸십시오.

- 두 인스턴스 모두 *codecatalyst-cfn-environment* 사용자 환경의 이름과 함께.
- 두 인스턴스 모두 *codecatalyst-account-connection* 계정 연결의 디스플레이 이름과 함께. 표시 이름은 숫자일 수 있습니다. 자세한 내용은 3단계: AWS 역할 추가 CodeCatalyst 단원을 참조하십시오.
- *codecatalyst-build-role* 에서 만든 빌드 역할의 이름을 사용합니다2단계: AWS 역할 생성.
- *codecatalyst-cfn-s3-bucket* 에서 생성한 Amazon S3 버킷의 이름을 사용합니다4단계: Amazon S3 버킷 생성.
- 의 두 인스턴스 모두 *us-west-2* Amazon S3 버킷이 있는 지역 (첫 번째 인스턴스) 과 스택이 배포될 지역 (두 번째 인스턴스) 을 사용합니다. 이들 지역은 다를 수 있습니다. 이 자습서에서는 두 지역이 모두 로 us-west-2 설정되어 있다고 가정합니다. Amazon S3에서 지원하는 지역에 대한 자세한 내용과 자세한 내용은 의 서비스 엔드포인트 및 할당량을 참조하십시오. AWS CloudFormationAWS 일반 참조
- *codecatalyst-deploy-role* 에서 생성한 배포 역할의 이름과 함께. 2단계: AWS 역할 생성
- *codecatalyst-cfn-environment* 에서 만든 환경의 이름과 함께사전 조건.
- *arn:aws:iam::111122223333:role/StackRole* 에서 생성한 스택 역할의 Amazon 리소스 이름 (ARN) 을 사용합니다2단계: AWS 역할 생성.

> **Note**
>
> 빌드, 배포 및 스택 역할을 생성하지 않기로 결정했다면 다음을 대체하십시오.*codecatalyst-build-role*, *codecatalyst-deploy-role*, 및 *arn:aws:iam::111122223333:role/StackRole* 이름 또는 ARN CodeCatalystWorkflowDevelopmentRole-*spaceName* 역할의 이름을 입력하세요. 이에 대한 자세한 내용은 2단계: AWS 역할 생성 섹션을 참조하세요.

이전에 표시된 코드의 속성에 대한 자세한 내용은 를 참조하십시오AWS CloudFormation '스택 배포' 작업 YAML.

8. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 YAML 코드가 유효한지 확인합니다.
9. 커밋을 선택합니다.
10. 워크플로 커밋 대화 상자에서 다음을 입력합니다.
    a. 워크플로 파일 이름의 경우 기본값인 을 유지합니다codecatalyst-cfn-workflow.

b. 커밋 메시지에는 다음을 입력합니다.

```
add initial workflow file
```

c. 리포지토리의 경우 선택합니다 codecatalyst-cfn-source-repository.

d. 브랜치 이름으로 main을 선택합니다.

e. 커밋을 선택합니다.

이제 워크플로가 생성되었습니다. 워크플로 맨 위에 정의된 트리거로 인해 워크플로 실행이 자동으로 시작됩니다. 특히, `codecatalyst-cfn-workflow.yaml` 파일을 소스 리포지토리에 커밋(및 푸시) 하면 트리거가 워크플로 실행을 시작했습니다.

실행 중인 워크플로를 보려면

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 방금 만든 워크플로를 선택합니다. `codecatalyst-cfn-workflow`
3. 실행 탭을 선택합니다.
4. 실행 ID 열에서 실행 ID를 선택합니다.
5. 테스트를 선택하여 테스트 진행 상황을 확인합니다.
6. 빌드 진행 상황을 BuildBackend확인하도록 선택합니다.
7. DeployCloudFormationStack선택하여 배포 진행 상황을 확인하세요.

   실행 세부 정보 보기에 대한 자세한 내용은 을 참조하십시오워크플로 실행 상태 및 세부 정보 보기.
8. DeployCloudFormationStack작업이 완료되면 다음을 수행하십시오.

   - 워크플로가 성공적으로 실행되면 다음 절차로 이동합니다.
   - 테스트 또는 BuildBackend작업에서 워크플로 실행이 실패한 경우 로그를 선택하여 문제를 해결하십시오.
   - 작업에서 워크플로 실행이 DeployCloudFormationStack실패한 경우 배포 작업을 선택한 다음 요약 탭을 선택합니다. CloudFormation 이벤트 섹션으로 스크롤하여 자세한 오류 메시지를 확인합니다. 롤백이 발생한 경우 워크플로를 다시 AWS 실행하기 전에 AWS CloudFormation 콘솔을 통해 `codecatalyst-cfn-stack` 스택을 삭제하십시오.

배포를 확인하려면

1. 배포에 성공하면 상단 근처의 가로 메뉴 표시줄에서 Variables (7) 를 선택합니다. (오른쪽 창에서 변수를 선택하지 마십시오.)
2. HelloWorldApi다음으로 를 `https://` URL 브라우저에 붙여넣습니다.

   Lambda 함수의 hello world JSON 메시지가 표시되어 워크플로가 Lambda 함수 및 Gateway를 성공적으로 배포 및 구성했음을 나타냅니다. API

   > **Tip**
   >
   > 몇 가지 작은 구성으로 워크플로 URL 다이어그램에 이를 CodeCatalyst 표시할 수 있습니다. 자세한 내용은 워크플로 URL 다이어그램에 앱 표시 단원을 참조하십시오.

단위 테스트 결과 및 코드 커버리지를 확인하려면

1. 워크플로 다이어그램에서 [Test] 를 선택한 다음 [Reports] 를 선택합니다.
2. 단위 테스트 결과를 보거나 테스트 중인 파일의 코드 적용 범위 세부 정보 (이 경우에는 `app.js` 및) 를 볼 수 `test-handler.js` 있습니다. TestReportCoverageReport

배포된 리소스를 확인하려면

1. 에 AWS Management Console 로그인하고 API 게이트웨이 콘솔을 여십시오 https://console.aws.amazon.com/apigateway/.
2. AWS SAM 템플릿이 codecatalyst-cfn-stackAPI생성되었는지 확인해 보세요. API이름은 워크플로 정의 파일 (`codecatalyst-cfn-workflow.yaml`) 의 `Configuration/name` 값에서 따왔습니다.
3. 에서 AWS Lambda 콘솔을 엽니다 https://console.aws.amazon.com/lambda/.
4. 탐색 창에서 함수를 선택합니다.
5. Lambda 함수를 선택하십시오. `codecatalyst-cfn-stack-HelloWorldFunction-`*`string`*
6. API게이트웨이가 어떻게 함수의 트리거인지 확인할 수 있습니다. 이 통합은 AWS SAM `AWS::Serverless::Function` 리소스 유형에 따라 자동으로 구성되었습니다.

7단계: 변경

이 단계에서는 Lambda 소스 코드를 변경하고 커밋합니다. 이 커밋은 새 워크플로 실행을 시작합니다. 이 실행은 Lambda 콘솔에 지정된 기본 트래픽 이동 구성을 사용하는 청록색 체계로 새 Lambda 함수를 배포합니다.

Lambda 소스를 변경하려면

1. 에서 CodeCatalyst 프로젝트로 이동합니다.
2. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
3. 소스 리포지토리를 선택합니다`codecatalyst-cfn-source-repository`.
4. 애플리케이션 파일 변경:

    a. `hello-world` 폴더를 선택합니다.

    b. `app.js` 파일을 선택합니다.

    c. 편집을 선택합니다.

    d. 23번째 줄에서 `hello world` 로 **`Tutorial complete!`** 변경하십시오.

    e. [커밋] 을 선택한 다음 [커밋] 을 다시 선택합니다.

    커밋하면 워크플로우 실행이 시작됩니다. 이름 변경을 반영하도록 단위 테스트를 업데이트하지 않았으므로 이 실행은 실패합니다.

5. 단위 테스트 업데이트:

    a. `hello-world\tests\unit\test-handler.js`를 선택합니다.

    b. 편집을 선택합니다.

    c. 19번째 줄에서 `hello world` 으로 **`Tutorial complete!`** 변경하십시오.

    d. [커밋] 을 선택한 다음 [커밋] 을 다시 선택합니다.

    커밋하면 다른 워크플로가 실행됩니다. 이 실행은 성공할 것입니다.

6. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
7. 선택한 **`codecatalyst-cfn-workflow`** 다음 실행을 선택합니다.
8. 최근 실행의 실행 ID를 선택합니다. 아직 진행 중이어야 합니다.
9. Test BuildBackend, DeployCloudFormationStack를 선택하여 워크플로우 실행 진행 상황을 확인합니다.
10. 워크플로가 끝나면 상단 근처에 있는 변수 (7) 를 선택합니다.

11. 그런 다음 `https://` URL 브라우저에 붙여넣습니다. HelloWorldApi

    새 애플리케이션이 성공적으로 배포되었음을 알리는 `Tutorial complete!` 메시지가 브라우저에 나타납니다.

정리

요금이 부과되지 않도록 이 자습서에서 사용된 파일 및 서비스를 정리하세요.

CodeCatalyst 콘솔에서 정리하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 삭제`codecatalyst-cfn-workflow`.
3. 삭제`codecatalyst-cfn-environment`.
4. 삭제`codecatalyst-cfn-source-repository`.
5. 삭제`codecatalyst-cfn-project`.

에서 정리하려면 AWS Management Console

1. 다음과 CloudFormation 같이 청소하십시오.
   a. https://console.aws.amazon.com/cloudformation 에서 AWS CloudFormation 콘솔을 엽니다.
   b. `codecatalyst-cfn-stack`를 삭제합니다.

      스택을 삭제하면 API Gateway 및 Lambda 서비스에서 모든 튜토리얼 리소스가 제거됩니다.
2. 다음과 같이 Amazon S3에서 정리합니다.
   a. 에서 Amazon S3 콘솔을 엽니다 https://console.aws.amazon.com/s3/.
   b. `codecatalyst-cfn-s3-bucket`을 선택합니다.
   c. 버킷 콘텐츠를 삭제합니다.
   d. 버킷을 삭제합니다.
3. 다음과 IAM 같이 정리하십시오.
   a. 에서 IAM 콘솔을 https://console.aws.amazon.com/iam/여십시오.
   b. `codecatalyst-deploy-policy`를 삭제합니다.
   c. `codecatalyst-build-policy`를 삭제합니다.

d. `codecatalyst-stack-policy`를 삭제합니다.
e. `codecatalyst-deploy-role`를 삭제합니다.
f. `codecatalyst-build-role`를 삭제합니다.
g. `codecatalyst-stack-role`를 삭제합니다.

이 자습서에서는 CodeCatalyst 워크플로와 스택 배포 작업을 사용하여 서버리스 애플리케이션을 CloudFormation 스택으로 AWS CloudFormation 배포하는 방법을 배웠습니다.

AWS CloudFormation '스택 배포' 작업 추가

다음 지침에 따라 AWS CloudFormation 스택 배포 작업을 워크플로에 추가하십시오.

Visual

시각적 편집기를 사용하여 AWS CloudFormation '스택 배포' 작업을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. AWS CloudFormation 스택 배포 작업을 검색하고 다음 중 하나를 수행하십시오.

   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - [AWS CloudFormation 스택 배포] 를 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서는
     - (선택 사항) 액션의 소스 코드를 보려면 다운로드를 선택합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

10. 입력 및 구성 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오AWS CloudFormation '스택 배포' 작업 YAML. 이 참조는 편집기와 시각적 편집기에 모두 나타나는 각 필드 (및 해당 YAML 속성 값) 에 대한 자세한 정보를 제공합니다. YAML
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

편집기를 사용하여 AWS CloudFormation '스택 배포' 작업을 추가하려면 YAML

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. AWS CloudFormation 스택 배포 작업을 검색하고 다음 중 하나를 수행하십시오.
   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - [AWS CloudFormation 스택 배포] 를 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서는
     - (선택 사항) 액션의 소스 코드를 보려면 다운로드를 선택합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.
10. 필요에 따라 YAML 코드의 속성을 수정하십시오. 사용 가능한 각 속성에 대한 설명은 에 나와 AWS CloudFormation '스택 배포' 작업 YAML 있습니다.

11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

롤백 구성

기본적으로 AWS CloudFormation 스택 배포 작업이 실패하면 스택이 마지막으로 알려진 안정 AWS CloudFormation 상태로 롤백됩니다. 작업이 실패할 때뿐만 아니라 지정된 Amazon CloudWatch 경보가 발생할 때도 롤백이 발생하도록 동작을 변경할 수 있습니다. CloudWatch 경보에 대한 자세한 내용은 Amazon 사용 CloudWatch 설명서의 Amazon CloudWatch 경보 사용을 참조하십시오.

작업이 실패할 때 스택이 롤백되지 CloudFormation 않도록 기본 동작을 변경할 수도 있습니다.

다음 지침을 사용하여 롤백을 구성하십시오.

> **Note**
>
> 롤백을 수동으로 시작할 수는 없습니다.

Visual

시작하기 전 준비 사항

1. 제대로 작동하는 배포 AWS CloudFormation 스택 작업이 포함된 워크플로가 있는지 확인하십시오. 자세한 내용은 스택 배포 AWS CloudFormation 단원을 참조하십시오.
2. 스택 배포 작업의 AWS CloudFormation 스택 역할 - 선택 필드에 지정된 역할에 CloudWatchFullAccess권한을 포함해야 합니다. 적절한 권한으로 이 역할을 생성하는 방법에 대한 자세한 내용은 을 참조하십시오2단계: AWS 역할 생성.

AWS CloudFormation '스택 배포' 작업에 대한 롤백 경보를 구성하려면

1. https://codecatalyst.aws/ 에서 콘솔을 엽니다. CodeCatalyst
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.

5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. AWS CloudFormation 스택 배포 작업을 선택합니다.
8. 세부 정보 창에서 구성을 선택합니다.
9. 하단에서 고급을 확장합니다.
10. 모니터 알람에서 알람 ARNs 추가를 선택합니다.
11. 다음 필드에 정보를 입력합니다.

    - 알람 ARN

      롤백 트리거로 사용할 Amazon CloudWatch 경보의 Amazon 리소스 이름 (ARN) 을 지정합니다. 예: `arn:aws:cloudwatch::123456789012:alarm/MyAlarm`. 롤백 트리거는 최대 5개까지 지정할 수 있습니다.

      > **Note**
      >
      > CloudWatch 경보를 ARN 지정하는 경우 작업이 액세스할 수 있도록 추가 권한도 구성해야 합니다. CloudWatch 자세한 내용은 롤백 구성 단원을 참조하십시오.

    - 모니터링 시간

      지정된 알람을 CloudFormation 모니터링하는 시간을 0분에서 180분 사이로 지정합니다. 모든 스택 리소스가 배포된 후에 모니터링이 시작됩니다. 지정된 모니터링 시간 내에 경보가 발생하면 배포가 실패하고 전체 스택 작업이 CloudFormation 롤백됩니다.

      기본값: 0. CloudFormation 스택 리소스가 배포되는 동안에만 경보를 모니터링하고 이후는 모니터링하지 않습니다.

YAML

'스택 배포' 작업에 대한 롤백 트리거를 구성하려면 AWS CloudFormation

1. https://codecatalyst.aws/ 에서 콘솔을 엽니다. CodeCatalyst
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.

4. AWS CloudFormation 스택 배포 작업이 포함된 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. YAML코드에 `monitor-alarm-arns` 및 `monitor-timeout-in-minutes` 속성을 추가하여 롤백 트리거를 추가합니다. 각 속성에 대한 설명은 을 참조하십시오. AWS CloudFormation '스택 배포' 작업 YAML
8. AWS CloudFormation 스택 배포 작업의 `role-arn` 속성에 지정된 역할에 CloudWatchFullAccess권한을 포함해야 합니다. 적절한 권한으로 이 역할을 생성하는 방법에 대한 자세한 내용은 을 참조하십시오2단계: AWS 역할 생성.

Visual

'스택 배포' AWS CloudFormation 작업에 대한 롤백을 끄려면

1. https://codecatalyst.aws/ 에서 콘솔을 엽니다. CodeCatalyst
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. AWS CloudFormation 스택 배포 작업이 포함된 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. AWS CloudFormation 스택 배포 작업을 선택합니다.
8. 세부 정보 창에서 구성을 선택합니다.
9. 하단에서 고급을 확장합니다.
10. 롤백 비활성화를 설정합니다.

YAML

AWS CloudFormation '스택 배포' 작업에 대한 롤백을 끄려면

1. https://codecatalyst.aws/ 에서 콘솔을 엽니다. CodeCatalyst

2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. AWS CloudFormation 스택 배포 작업이 포함된 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. YAML코드에 `disable-rollback: 1` 속성을 추가하여 롤백을 중지합니다. 이 속성에 대한 설명은 을 참조하십시오AWS CloudFormation '스택 배포' 작업 YAML.

'디플로이 AWS CloudFormation 스택' 변수

Deploy AWS CloudFormation stack 작업은 런타임에 다음 변수를 생성하고 설정합니다. 이러한 변수를 사전 정의된 변수라고 합니다.

워크플로우에서 이러한 변수를 참조하는 방법에 대한 자세한 내용은 을 참조하십시오. 사전 정의된 변수 사용

키	값
배포 플랫폼	배포 플랫폼의 이름. 로 하드코딩했습니다. `AWS:CloudFormation`
region	워크플로 실행 중에 AWS 리전 배포된 지역 코드입니다. 예제: `us-west-2`
stack-id	배포된 스택의 Amazon 리소스 이름 (ARN) 예제: `arn:aws:cloudformation:us-west-2:111122223333:stack/codecatalyst-cfn-stack/6aad4380-100a-11ec-a10a-03b8a84d40df`

AWS CloudFormation '스택 배포' 작업 YAML

다음은 AWS CloudFormation 스택 배포 작업의 YAML 정의입니다. 이 작업을 사용하는 방법을 알아보려면 을 참조하십시오스택 배포 AWS CloudFormation.

이 작업 정의는 광범위한 워크플로 정의 파일 내에 섹션으로 존재합니다. 이 파일에 대한 자세한 내용은 워크플로우 YAML 정의을 참조합니다.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

```
# The workflow definition starts here.
# See ### ## for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  DeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    DependsOn:
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: DeployRole
    Inputs:
      Sources:
        - source-name-1
      Artifacts:
        - CloudFormation-artifact
    Configuration:
      name: stack-name
```

```
      region: us-west-2
      template: template-path
      role-arn: arn:aws:iam::123456789012:role/StackRole
      capabilities: CAPABILITY_IAM,CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND
      parameter-overrides: KeyOne=ValueOne,KeyTwo=ValueTwo | path-to-JSON-file
      no-execute-changeset: 1|0
      fail-on-empty-changeset: 1|0
      disable-rollback: 1|0
      termination-protection: 1|0
      timeout-in-minutes: minutes
      notification-arns: arn:aws:sns:us-east-1:123456789012:MyTopic,arn:aws:sns:us-
east-1:123456789012:MyOtherTopic
      monitor-alarm-arns: arn:aws:cloudwatch::123456789012:alarm/
MyAlarm,arn:aws:cloudwatch::123456789012:alarm/MyOtherAlarm
      monitor-timeout-in-minutes: minutes
      tags: '[{"Key":"MyKey1","Value":"MyValue1"},{"Key":"MyKey2","Value":"MyValue2"}]'
```

DeployCloudFormationStack

(필수)

액션의 이름을 지정합니다. 모든 작업 이름은 워크플로 내에서 고유해야 합니다. 액션 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 액션 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

기본값: `DeployCloudFormationStack_nn`.

해당 UI: 구성 탭/ 작업 표시 이름

Identifier

(*DeployCloudFormationStack*/Identifier)

(필수)

작업을 식별합니다. 버전을 변경하려는 경우가 아니면 이 속성을 변경하지 마십시오. 자세한 내용은 사용할 액션 버전 지정 단원을 참조하십시오.

기본값: `aws/cfn-deploy@v1`.

해당 UI: 워크플로 다이어그램/ DeployCloudFormationStack _n/ aws/cfn-deploy @v1 라벨

DependsOn

(*DeployCloudFormationStack*/DependsOn)

(선택 사항)

이 작업을 실행하기 위해 성공적으로 실행되어야 하는 작업, 작업 그룹 또는 게이트를 지정하십시오.

'종속 조건' 기능에 대한 자세한 내용은 을 참조하십시오. 시퀀싱 액션

해당 UI: 입력 탭/ 기준 - 선택 사항

Compute

(*DeployCloudFormationStack*/Compute)

(선택 사항)

워크플로 작업을 실행하는 데 사용되는 컴퓨팅 엔진. 워크플로 수준 또는 작업 수준에서 컴퓨팅을 지정할 수 있지만 둘 다에서 지정할 수는 없습니다. 워크플로 수준에서 지정된 경우 컴퓨팅 구성은 워크플로에 정의된 모든 작업에 적용됩니다. 워크플로 수준에서는 동일한 인스턴스에서 여러 작업을 실행할 수도 있습니다. 자세한 내용은 작업 간 컴퓨팅 공유 단원을 참조하십시오.

해당 UI: 없음

Type

(*DeployCloudFormationStack*/Compute/Type)

(포함된 Compute 경우 필수)

컴퓨팅 엔진 유형. 다음 값 중 하나를 사용할 수 있습니다.

- EC2(비주얼 에디터) 또는 EC2 (YAML에디터)

  작업 실행 중 유연성을 위해 최적화되었습니다.
- Lambda (비주얼 에디터) 또는 (에디터) Lambda YAML

  작업 시작 속도를 최적화했습니다.

컴퓨팅 유형에 대한 자세한 정보는 컴퓨팅 유형을 참고하세요.

해당 UI: 구성 탭/고급 - 선택 사항/ 컴퓨팅 유형

Fleet

(*DeployCloudFormationStack*/Compute/Fleet)

(선택 사항)

워크플로 또는 워크플로 작업을 실행할 시스템 또는 플릿을 지정합니다. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다. 온디맨드 플릿의 예:,. `Linux.x86-64.Large` `Linux.x86-64.XLarge` 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

생략된 경우 기본값은 `Fleet` 입니다. `Linux.x86-64.Large`

해당 UI: 구성 탭/고급 - 선택 사항/ 컴퓨팅 플릿

Timeout

(*`DeployCloudFormationStack`*/Timeout)

(선택 사항)

작업이 종료되기 전에 작업을 실행할 수 있는 시간을 분 (YAML편집기) 또는 시간과 분 (비주얼 편집기) 단위로 지정합니다. CodeCatalyst 최소값은 5분이고 최대값은 에 설명되어 의 워크플로우 할당량 CodeCatalyst 있습니다. 기본 타임아웃은 최대 타임아웃과 동일합니다.

해당 UI: 구성 탭/ 타임아웃 (분) - 선택 사항

Environment

(*`DeployCloudFormationStack`*/Environment)

(필수)

작업에 사용할 CodeCatalyst 환경을 지정합니다. 작업은 선택한 환경에 VPC 지정된 AWS 계정 및 선택적 Amazon에 연결됩니다. 작업은 환경에 지정된 기본 IAM 역할을 사용하여 Amazon에 연결하고 Amazon VPC 연결에 지정된 IAM 역할을 사용하여 Amazon에 연결합니다VPC. AWS 계정

> **Note**
>
> 기본 IAM 역할에 작업에 필요한 권한이 없는 경우 다른 역할을 사용하도록 작업을 구성할 수 있습니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.

환경에 대한 자세한 내용은 및 에 AWS 계정 배포 VPCs 및 을 참조하십시오환경 생성.

해당 UI: 구성 탭/ 환경

Name

(*DeployCloudFormationStack*/Environment/Name)

(포함된 Environment 경우 필수)

작업에 연결할 기존 환경의 이름을 지정합니다.

해당 UI: 구성 탭/ 환경

Connections

(*DeployCloudFormationStack*/Environment/Connections)

(액션의 최신 버전에서는 선택 사항, 이전 버전에서는 필수)

액션과 연결할 계정 연결을 지정합니다. 에서 최대 1개의 계정 연결을 지정할 수 `Environment` 있습니다.

계정 연결을 지정하지 않은 경우:

- 작업은 CodeCatalyst 콘솔 환경에 지정된 AWS 계정 연결 및 기본 IAM 역할을 사용합니다. 환경에 계정 연결 및 기본 IAM 역할을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.
- 기본 IAM 역할에는 작업에 필요한 정책 및 권한이 포함되어야 합니다. 이러한 정책 및 권한이 무엇인지 확인하려면 작업 YAML 정의 설명서에서 역할 속성에 대한 설명을 참조하십시오.

계정 연결에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정. 환경에 계정 연결을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

해당 UI: 액션 버전에 따라 다음 중 하나

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Name

(*DeployCloudFormationStack*/Environment/Connections/Name)

(포함된 경우 필수Connections)

계정 연결 이름을 지정합니다.

해당 UI: 액션 버전에 따라 다음 중 하나:

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Role

(*DeployCloudFormationStack*/Environment/Connections/Role)

(포함된 경우 필수Connections)

Deploy AWS CloudFormation stack 작업에서 AWS 액세스에 사용하는 IAM 역할의 이름과 AWS CloudFormation 서비스를 지정합니다. CodeCatalyst 스페이스에 역할을 추가했는지, 역할에 다음 정책이 포함되어 있는지 확인하세요.

IAM역할을 지정하지 않으면 CodeCatalyst 콘솔 환경에 나열된 기본 IAM 역할이 작업에 사용됩니다. 환경에서 기본 역할을 사용하는 경우 해당 역할에 다음 정책이 적용되는지 확인하십시오.

- 다음 권한 정책:

> **Warning**
>
> 권한을 다음 정책에 표시된 권한으로 제한하십시오. 더 광범위한 권한이 있는 역할을 사용하면 보안 위험이 발생할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
```

```
            "cloudformation:SetStackPolicy",
            "cloudformation:ValidateTemplate",
            "cloudformation:List*",
            "iam:PassRole"
        ],
        "Resource": "*",
        "Effect": "Allow"
    }]
    }
```

> **Note**
>
> 역할을 처음 사용할 때는 리소스 정책 설명에 다음 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

- 다음과 같은 사용자 지정 신뢰 정책:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

> **Note**
>
> 원하는 경우 이 작업에 CodeCatalystWorkflowDevelopmentRole-*spaceName* 역할을 사용할 수 있습니다. 이에 대한 자세한 내용은 계정 및 스페이스에 대한

CodeCatalystWorkflowDevelopmentRole-*spaceName*역할 만들기 섹션을 참조하세요. CodeCatalystWorkflowDevelopmentRole-*spaceName*역할에는 보안 위험을 초래할 수 있는 전체 액세스 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다.

해당 UI: 액션 버전에 따라 다음 중 하나가 표시됩니다.

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 역할

Inputs

(*DeployCloudFormationStack*/Inputs)

(선택 사항)

이 Inputs 섹션에서는 워크플로우 실행 중에 필요한 데이터를 정의합니다. DeployCloudFormationStack

> **Note**
>
> AWS CloudFormation 스택 배포 작업당 최대 4개의 입력 (소스 1개와 아티팩트 3개) 이 허용됩니다.

서로 다른 입력 (예: 소스 및 아티팩트) 에 있는 파일을 참조해야 하는 경우 소스 입력은 기본 입력이고 아티팩트는 보조 입력입니다. 보조 입력의 파일에 대한 참조에는 기본 입력과 구분하기 위해 특수 접두사가 사용됩니다. 세부 정보는 예: 여러 아티팩트의 파일 참조을 참조하세요.

해당 UI: 입력 탭

Sources

(*DeployCloudFormationStack*/Inputs/Sources)

(사용자 CloudFormation 또는 AWS SAM 템플릿이 소스 리포지토리에 저장되어 있는 경우 필수)

CloudFormation 또는 AWS SAM 템플릿이 소스 리포지토리에 저장되어 있는 경우 해당 소스 리포지토리의 레이블을 지정하십시오. 현재 지원되는 유일한 레이블은 입니다WorkflowSource.

CloudFormation 또는 AWS SAM 템플릿이 원본 리포지토리에 포함되어 있지 않은 경우 다른 작업에서 생성된 아티팩트나 Amazon S3 버킷에 있어야 합니다.

소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

해당 UI: 입력 탭/ 소스 - 선택 사항

Artifacts - input

(*DeployCloudFormationStack*/Inputs/Artifacts)

(사용자 CloudFormation 또는 AWS SAM 템플릿이 이전 작업의 출력 아티팩트에 저장되어 있는 경우 필수)

배포하려는 CloudFormation 또는 AWS SAM 템플릿이 이전 작업에서 생성된 아티팩트에 포함되어 있는 경우 여기에 해당 객체를 지정하십시오. CloudFormation 템플릿이 아티팩트에 포함되어 있지 않은 경우 템플릿은 원본 리포지토리 또는 Amazon S3 버킷에 있어야 합니다.

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 구성 탭/ 아티팩트 - 선택 사항

Configuration

(*DeployCloudFormationStack*/Configuration)

(필수)

작업의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭

name

(*DeployCloudFormationStack*/Configuration/name)

(필수)

CloudFormation 스택 배포 작업에서 생성하거나 업데이트하는 AWS CloudFormation 스택의 이름을 지정합니다.

해당 UI: 구성 탭/ 스택 이름

region

(*DeployCloudFormationStack*/Configuration/region)

(필수)

AWS 리전 스택을 배포할 대상을 지정합니다. 리전 코드 목록은 리전 엔드포인트를 참조하세요.

해당 UI: 구성 탭/ 스택 영역

template

(*DeployCloudFormationStack*/Configuration/template)

(필수)

CloudFormation 또는 AWS SAM 템플릿 파일의 이름과 경로를 지정합니다. 템플릿은 JSON 또는 YAML 형식일 수 있으며, 소스 리포지토리, 이전 작업의 아티팩트 또는 Amazon S3 버킷에 위치할 수 있습니다. 템플릿 파일이 소스 리포지토리 또는 아티팩트에 있는 경우 경로는 소스 또는 아티팩트 루트를 기준으로 합니다. 템플릿이 Amazon S3 버킷에 있는 경우 경로는 템플릿의 객체 URL 값입니다.

예시:

`./MyFolder/MyTemplate.json`

`MyFolder/MyTemplate.yml`

`https://MyBucket.s3.us-west-2.amazonaws.com/MyTemplate.yml`

> **Note**
>
> 템플릿의 파일 경로에 템플릿을 찾을 아티팩트 또는 소스를 나타내는 접두사를 추가해야 할 수도 있습니다. 자세한 내용은 소스 리포지토리 파일 참조 및 아티팩트의 파일 참조 단원을 참조하세요.

해당 UI: 구성 탭/ 템플릿

role-arn

(*DeployCloudFormationStack*/Configuration/role-arn)

(필수)

스택 역할의 Amazon 리소스 이름 (ARN) 을 지정합니다. CloudFormation 이 역할을 사용하여 스택의 리소스에 액세스하고 이를 수정합니다. 예: `arn:aws:iam::123456789012:role/StackRole`.

스택 역할에 다음이 포함되는지 확인하세요.

- 하나 이상의 권한 정책. 정책은 스택에 있는 리소스에 따라 달라집니다. 예를 들어 스택에 AWS Lambda 함수가 포함된 경우 Lambda에 대한 액세스 권한을 부여하는 권한을 추가해야 합니다. 에서 자습서: 서버리스 애플리케이션 배포 설명한 자습서를 따랐다면 일반적인 서버리스 애플리케이션 스택을 배포할 때 스택 역할에 필요한 권한을 스택 역할을 만들려면 나열하는 제목의 절차가 포함되어 있습니다.

> **Warning**
>
> CloudFormation 서비스가 스택의 리소스에 액세스하는 데 필요한 권한으로 권한을 제한하십시오. 더 광범위한 권한이 있는 역할을 사용하면 보안 위험이 발생할 수 있습니다.

- 다음 신뢰 정책:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudformation.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

선택적으로 이 역할을 계정 연결에 연결할 수 있습니다. IAM역할을 계정 연결과 연결하는 방법에 대한 자세한 내용은 을 참조하십시오계정 연결에 IAM 역할 추가. 스택 역할을 계정 연결과 연결하지 않으면 시각적 편집기의 스택 역할 드롭다운 목록에 스택 역할이 나타나지 않지만 편집기를 사용하여 `role-arn` 필드에 역할을 지정할 수는 ARN 있습니다. YAML

> **Note**
>
> 원하는 경우 이 작업에 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 역할을 사용할 수 있습니다. 이에 대한 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기 섹션을 참조하세요.

`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*역할에는 보안 위험을 초래할 수 있는 전체 액세스 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다.

해당 UI: 구성 탭/ 스택 역할 - 선택 사항

capabilities

(*`DeployCloudFormationStack`*/Configuration/capabilities)

(필수)

특정 스택을 생성하는 AWS CloudFormation 데 필요한 IAM 기능 목록을 지정합니다. 대부분의 경우 기본값인 `capabilities` 을 그대로 둘 수 있습니다. `CAPABILITY_IAM,CAPABILITY_NAMED_IAM,CAPABILITY_AUTO_EXPAND`

배포 AWS CloudFormation 스택 작업 `##[error] requires capabilities: [`*`capability-name`*`]` 로그에 표시되는 경우 문제 해결 방법에 대한 자세한 내용은 을 참조하십시오기능 오류를 수정하려면 어떻게 해야 합니까? IAM .

IAM기능에 대한 자세한 내용은 IAM사용 설명서의 AWS CloudFormation 템플릿의 IAM 리소스 승인을 참조하십시오.

해당 UI: 구성 탭/고급/기능

parameter-overrides

(*`DeployCloudFormationStack`*/Configuration/parameter-overrides)

(선택 사항)

기본값이 없거나 기본값이 아닌 값을 지정하려는 매개 변수를 사용자 AWS CloudFormation 또는 AWS SAM 템플릿에서 지정하십시오. 매개변수에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 매개변수를 참조하십시오.

이 `parameter-overrides` 속성은 다음을 허용합니다.

- 매개변수와 값이 들어 있는 JSON 파일입니다.
- 쉼표로 구분된 매개 변수 및 값 목록.

파일을 지정하려면 JSON

1. JSON파일이 다음 구문 중 하나를 사용하는지 확인하십시오.

```
{
  "Parameters": {
    "Param1": "Value1",
    "Param2": "Value2",
    ...
  }
}
```

또는...

```
[
  {
     "ParameterKey": "Param1",
     "ParameterValue": "Value1"
  },
  ...
]
```

(다른 구문도 있지만 작성 CodeCatalyst 당시에는 지원되지 않습니다.) JSON파일에 CloudFormation 매개 변수를 지정하는 방법에 대한 자세한 내용은 AWS CLI 명령 참조에서 지원되는 JSON 구문을 참조하십시오.

2. 다음 형식 중 하나를 사용하여 JSON 파일 경로를 지정합니다.

   - JSON파일이 이전 액션의 출력 아티팩트에 있는 경우 다음을 사용하십시오.

     `file:///artifacts/`*`current-action-name`*`/`*`output-artifact-name`*`/`*`path-to-json-file`*

     자세한 내용은 예제 1을 참조하십시오.

   - JSON파일이 소스 리포지토리에 있는 경우 다음을 사용하세요.

     `file:///sources/WorkflowSource/`*`path-to-json-file`*

     자세한 내용은 예제 2를 참조하십시오.

     예 1 - JSON 파일이 출력 아티팩트에 있습니다.

```
##My workflow YAML
...
Actions:
  MyBuildAction:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ParamArtifact
          Files:
            - params.json
    Configuration:
    ...
  MyDeployCFNStackAction:
    Identifier: aws/cfn-deploy@v1
    Configuration:
      parameter-overrides: file:///artifacts/MyDeployCFNStackAction/
ParamArtifact/params.json
```

예 2 - JSON 파일이 소스 리포지토리의 라는 폴더에 있습니다. `my/folder`

```
##My workflow YAML
...
Actions:
  MyDeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      parameter-overrides: file:///sources/WorkflowSource/my/folder/params.json
```

쉼표로 구분된 매개 변수 목록 사용하기

- 다음 형식을 사용하여 `parameter-overrides` 속성에 매개변수 이름-값 쌍을 추가합니다.

  param-1=*value-1*,*param-2*=*value-2*

  예를 들어 다음과 같은 템플릿을 가정해 보겠습니다. AWS CloudFormation

```
##My CloudFormation template

Description: My AWS CloudFormation template

Parameters:
  InstanceType:
    Description: Defines the Amazon EC2 compute for the production server.
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t2.small
      - t3.medium

Resources:
...
```

... parameter-overrides 속성을 다음과 같이 설정할 수 있습니다.

```
##My workflow YAML
...
Actions:
...
  DeployCloudFormationStack:
    Identifier: aws/cfn-deploy@v1
    Configuration:
      parameter-overrides: InstanceType=t3.medium,UseVPC=true
```

> **Note**
>
> 를 undefined 값으로 사용하여 해당 값을 사용하지 않고 매개 변수 이름을 지정할 수 있습니다. 예:
> parameter-overrides: MyParameter=undefined
> 결과적으로 스택 업데이트 중에 지정된 파라미터 이름에 기존 파라미터 값이 CloudFormation 사용됩니다.

해당 UI:

- 구성 탭/고급/파라미터 오버라이드
- 구성 탭/고급/파라미터 오버라이드/ 파일을 사용하여 오버라이드 지정
- 구성 탭/고급/파라미터 오버라이드/ 값 세트를 사용하여 오버라이드 지정

no-execute-changeset

(*DeployCloudFormationStack*/Configuration/no-execute-changeset)

(선택 사항)

변경 세트를 만든 다음 실행하기 전에 CodeCatalyst 중지할지 여부를 지정합니다. CloudFormation 이렇게 하면 CloudFormation 콘솔에서 변경 세트를 검토할 수 있습니다. 변경 세트가 괜찮다고 판단되면 이 옵션을 비활성화한 다음 워크플로를 다시 실행하여 중단하지 않고 변경 세트를 만들고 실행할 CodeCatalyst 수 있도록 하십시오. 기본값은 중단하지 않고 변경 세트를 만들고 실행하는 것입니다. 자세한 내용은 AWS CLI 명령 참조의 AWS CloudFormation 배포 매개 변수를 참조하십시오. 변경 세트 보기에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 변경 세트 보기를 참조하십시오.

해당 UI: 구성 탭/고급/변경 세트 실행 안 함

fail-on-empty-changeset

(*DeployCloudFormationStack*/Configuration/fail-on-empty-changeset)

(선택 사항)

변경 세트가 비어 있는 경우 AWS CloudFormation 스택 배포 작업을 CodeCatalyst 실패할지 여부를 지정합니다. CloudFormation (변경 세트가 비어 있으면 최근 배포 중에 스택에 적용된 변경 사항이 없음을 의미합니다.) 기본값은 변경 세트가 비어 있는 경우 작업이 진행되도록 허용하고 스택이 업데이트되지 않은 경우에도 `UPDATE_COMPLETE` 메시지를 반환하도록 하는 것입니다.

이 설정에 대한 자세한 내용은 AWS CLI 명령 참조의 AWS CloudFormation 배포 매개 변수를 참조하십시오. 변경 세트에 대한 자세한 내용은 사용 설명서의AWS CloudFormation 변경 세트를 사용한 스택 업데이트를 참조하십시오.

해당 UI: 구성 탭/고급/빈 변경 세트 실패

disable-rollback

(*DeployCloudFormationStack*/Configuration/disable-rollback)

(선택 사항)

스택 배포가 실패할 경우 스택 배포를 롤백할지 여부를 지정하십시오. CodeCatalyst 롤백은 스택을 마지막으로 알려진 안정 상태로 되돌립니다. 기본값은 롤백을 활성화하는 것입니다. 이 설정에 대한 자세한 내용은 AWS CLI 명령 참조의 AWS CloudFormation 배포 매개 변수를 참조하십시오.

AWS CloudFormation 스택 배포 작업이 롤백을 처리하는 방법에 대한 자세한 내용은 을 참조하십시오롤백 구성.

스택 롤백에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 스택 장애 옵션을 참조하십시오.

해당 UI: 구성 탭/고급/롤백 비활성화

termination-protection

(*DeployCloudFormationStack*/Configuration/termination-protection)

(선택 사항)

배포 스택에서 배포 중인 AWS CloudFormation 스택에 종료 보호를 추가할지 여부를 지정합니다. 종료 방지 기능이 활성화된 스택을 삭제하려고 시도하면 삭제가 실패하고 스택은 상태를 포함하여 변함없이 그대로 유지됩니다. 기본값은 종료 보호를 비활성화하는 것입니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 스택 삭제 방지를 참조하십시오.

해당 UI: 구성 탭/고급/종료 보호

timeout-in-minutes

(*DeployCloudFormationStack*/Configuration/timeout-in-minutes)

(선택 사항)

스택 생성 작업 시간을 초과하고 스택 상태를 로 설정하기 전에 CloudFormation 할당해야 하는 시간(분) 을 지정합니다. `CREATE_FAILED` 할당된 시간 내에 전체 스택을 생성할 CloudFormation 수 없는 경우 시간 초과로 인해 스택 생성이 실패하고 스택이 롤백됩니다.

기본적으로 스택 생성에는 시간 초과가 없습니다. 하지만 개인 리소스에는 해당 리소스가 구현하는 서비스의 특성에 따라 자체 시간 초과가 설정될 수 있습니다. 예를 들어 스택의 개별 리소스가 시간 초과될 경우, 스택 생성에 대해 지정한 시간 초과 시간이 아직 지나지 않았더라도 스택 생성이 시간 초과됩니다.

해당 UI: 구성 탭/고급/타임아웃 CloudFormation

notification-arns

(*DeployCloudFormationStack*/Configuration/notification-arns)

(선택 사항)

알림 메시지를 CodeCatalyst 보내려는 Amazon SNS 주제를 지정하십시오. ARN 예: `arn:aws:sns:us-east-1:111222333:MyTopic`. AWS CloudFormation 스택 배포 작업이 실행되면 CodeCatalyst 와 CloudFormation 협력하여 스택 생성 또는 업데이트 프로세스 중에 발생하는 AWS CloudFormation 이벤트당 알림 하나를 전송합니다. (이벤트는 AWS CloudFormation 콘솔의 스택에 대한 Events 탭에서 볼 수 있습니다.) 주제는 최대 5개까지 지정할 수 있습니다. 자세한 내용은 Amazon이란 무엇입니까SNS? 를 참조하십시오. .

해당 UI: 구성 탭/고급/알림 ARNs

monitor-alarm-arns

(*DeployCloudFormationStack*/Configuration/monitor-alarm-arns)

(선택 사항)

롤백 트리거로 사용할 Amazon CloudWatch 경보의 Amazon 리소스 이름 (ARN) 을 지정합니다. 예: `arn:aws:cloudwatch::123456789012:alarm/MyAlarm`. 롤백 트리거는 최대 5개까지 지정할 수 있습니다.

> **Note**
>
> CloudWatch 경보를 ARN 지정하는 경우 작업이 액세스할 수 있도록 추가 권한도 구성해야 합니다. CloudWatch 자세한 내용은 롤백 구성 단원을 참조하십시오.

해당 UI: 구성 탭/고급/모니터 알람 ARNs

monitor-timeout-in-minutes

(*DeployCloudFormationStack*/Configuration/monitor-timeout-in-minutes)

(선택 사항)

지정된 알람을 모니터링하는 시간을 0분에서 180분 사이로 지정합니다. CloudFormation 모든 스택 리소스가 배포된 후에 모니터링이 시작됩니다. 지정된 모니터링 시간 내에 경보가 발생하면 배포가 실패하고 전체 스택 작업이 CloudFormation 롤백됩니다.

기본값: 0. CloudFormation 스택 리소스가 배포되는 동안에만 경보를 모니터링하고 이후는 모니터링하지 않습니다.

해당 UI: 구성 탭/고급/모니터링 시간

tags

(*DeployCloudFormationStack*/Configuration/tags)

(선택 사항)

스택에 첨부할 태그를 지정하십시오. CloudFormation 태그는 비용 할당과 같은 목적으로 스택을 식별하는 데 사용할 수 있는 임의의 키-값 쌍입니다. 태그의 정의 및 사용 방법에 대한 자세한 내용은 Amazon EC2 User Guide의 리소스 태그 지정을 참조하십시오. 태그 지정에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 스택 옵션 설정을 참조하십시오. CloudFormation

키는 영숫자 문자나 공백을 사용할 수 있으며 최대 127자까지 사용할 수 있습니다. 값은 영숫자 문자나 공백을 포함할 수 있으며 최대 255자까지 포함할 수 있습니다.

각 스택에 대해 최대 50개의 고유 태그를 추가할 수 있습니다.

해당 UI: 구성 탭/고급/ 태그

워크플로를 통한 AWS CDK 앱 배포

이 섹션에서는 워크플로를 사용하여 AWS 계정에 AWS Cloud Development Kit (AWS CDK) 앱을 배포하는 방법을 설명합니다. 이 작업을 수행하려면 워크플로에 AWS CDK 배포 작업을 추가해야 합니다. AWS CDK 배포 작업은 앱을 합성하여 에 배포합니다. AWS Cloud Development Kit (AWS CDK) AWS 에 앱이 이미 있는 AWS경우 작업은 필요한 경우 앱을 업데이트합니다.

를 사용하여 앱을 작성하는 방법에 대한 일반적인 내용은 AWS CDKWhatis는 무엇입니까 AWS CDK?를 참조하십시오. AWS Cloud Development Kit (AWS CDK) 개발자 안내서에서

AWS CDK '배포' 작업 사용 시기

를 사용하여 앱을 개발했는데 이제 자동화된 지속적 통합 및 전달 (CI/CD) 워크플로의 일부로 앱을 자동으로 배포하려는 경우 이 작업을 사용하십시오. AWS CDK예를 들어, 누군가 AWS CDK 앱 소스와 관련된 풀 리퀘스트를 병합할 때마다 앱을 자동으로 배포하고 싶을 수 AWS CDK 있습니다.

AWS CDK '배포' 작업 작동 방식

AWS CDK 배포는 다음과 같이 작동합니다.

1. 런타임 시 액션의 버전 1.0.12 이하를 지정한 경우 액션은 최신 버전 CDK CLI (AWS CDK Tookit이라고도 함) 을 빌드 이미지에 다운로드합니다. CodeCatalyst

   버전 1.0.13 이상을 지정한 경우 작업은 의 특정 버전과 함께 번들로 제공되므로 다운로드가 수행되지 않습니다. CDK CLI
2. 액션은 를 CDK CLI 사용하여 명령을 실행합니다. `cdk deploy` 이 명령어는 AWS CDK 앱을 합성하여 에 배포합니다. AWS이 명령에 대한 자세한 내용은 개발자 안내서의 AWS CDK 툴킷 (cdk 명령) 항목을 참조하십시오.AWS Cloud Development Kit (AWS CDK)

CDKCLI“AWS CDK 배포” 작업에 사용되는 버전

다음 표는 AWS CDK 배포 작업의 여러 버전에서 기본적으로 사용되는 의 버전을 보여줍니다. CDK CLI

> **Note**
>
> 기본값을 재정의할 수 있을 수도 있습니다. 자세한 내용은 AWS CDK '배포' 작업 YAML의 CdkCliVersion 섹션을 참조하십시오.

AWS CDK '배포' 액션 버전	AWS CDK CLI버전
1.0.0 — 1.0.12	최신
1.0.13 이상	2.99.1

액션으로 배포할 수 있는 스택은 몇 개입니까?

AWS CDK 배포는 단일 스택만 배포할 수 있습니다. AWS CDK 앱이 여러 스택으로 구성된 경우 중첩된 스택으로 상위 스택을 만들고 이 작업을 사용하여 상위 스택을 배포해야 합니다.

주제

- 예: 앱 AWS CDK 배포
- AWS CDK '배포' 작업 추가
- AWS CDK '배포' 변수
- AWS CDK '배포' 작업 YAML

예: 앱 AWS CDK 배포

다음 예제 워크플로에는 AWS CDK 부트스트랩 작업과 함께 AWS CDK 배포 작업이 포함되어 있습니다. 워크플로는 순차적으로 실행되는 다음과 같은 구성 요소로 구성됩니다.

- 트리거 - 이 트리거는 소스 리포지토리에 변경 내용을 푸시할 때 워크플로가 자동으로 실행됩니다. 이 리포지토리에는 AWS CDK 앱이 포함되어 있습니다. 트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.
- AWS CDK 부트스트랩 액션 (CDKBootstrap) — 트리거 시 액션은 CDKToolkit 부트스트랩 스택을 에 배포합니다. AWSCDKToolkit스택이 환경에 이미 있는 경우 필요에 따라 업그레이드됩니다. 그렇지 않으면 아무 일도 일어나지 않고 작업이 성공한 것으로 표시됩니다.
- AWS CDK 배포 작업 (AWS CDK Deploy) — AWS CDK 부트스트랩 작업이 완료되면 AWS CDK 배포 작업은 AWS CDK 앱 코드를 템플릿에 합성하고 AWS CloudFormation 템플릿에 정의된 스택을 템플릿에 배포합니다. AWS

> **Note**
>
> 다음 워크플로 예제는 설명을 위한 것으로 추가 구성 없이는 작동하지 않습니다.

> **Note**
>
> 다음 YAML 코드에서는 원하는 경우 Connections: 섹션을 생략할 수 있습니다. 이러한 섹션을 생략하는 경우 환경의 기본 IAM 역할 필드에 지정된 역할에 AWS CDK 부트스트랩 및 배포 작업에 필요한 권한 및 신뢰 정책이 포함되어 있는지 확인해야 합니다.AWS CDK 기본 IAM 역할을 사용하여 환경을 설정하는 방법에 대한 자세한 내용은 을 참조하십시오. 환경 생성 AWS CDK 부트스트랩 및 AWS CDK 배포 작업에 필요한 권한 및 신뢰 정책에 대한 자세한 내용은 AWS CDK '부트스트랩' 액션 YAML 및 AWS CDK '배포' 작업 YAML 의 Role 속성 설명을 참조하십시오.

```
Name: codecatalyst-cdk-deploy-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
```

```
      - main
Actions:
  CDKBootstrap:
    Identifier: aws/cdk-bootstrap@v1
    Inputs:
      Sources:
        - WorkflowSource
    Environment:
      Name: codecatalyst-cdk-deploy-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-cdk-bootstrap-role
    Configuration:
      Region: us-west-2

  CDKDeploy:
    Identifier: aws/cdk-deploy@v1
    DependsOn:
      - CDKBootstrap
    Environment:
      Name: codecatalyst-cdk-deploy-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-cdk-deploy-role
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      StackName: my-app-stack
      Region: us-west-2
```

AWS CDK '배포' 작업 추가

다음 지침에 따라 워크플로에 AWS CDK 배포 작업을 추가하세요.

시작하기 전에

워크플로에 AWS CDK 배포 작업을 추가하기 전에 다음 작업을 완료하십시오.

1. AWS CDK 앱을 준비하세요. 에서 지원하는 모든 프로그래밍 언어로 AWS CDK v1 또는 v2를 사용하여 AWS CDK 앱을 작성할 수 있습니다. AWS CDK AWS CDK 앱 파일을 다음 위치에서 사용할 수 있는지 확인하세요.
   - CodeCatalyst 소스 리포지토리 또는

- 다른 워크플로 작업에 의해 생성된 CodeCatalyst 출력 아티팩트

2. 환경을 부트스트랩하세요. AWS 부트스트랩하려면 다음을 수행할 수 있습니다.

   - AWS Cloud Development Kit (AWS CDK) 개발자 안내서의 부트스트랩 방법에 설명된 방법 중 하나를 사용하십시오.
   - AWS CDK 부트스트랩 작업을 사용하십시오. 이 작업은 AWS CDK 배포와 동일한 워크플로에 추가하거나 다른 워크플로에 추가할 수 있습니다. AWS CDK 배포 작업을 실행하기 전에 부트스트랩 작업을 한 번 이상 실행해야 필요한 리소스가 제자리에 있는지 확인할 수 있습니다. AWS CDK 부트스트랩 작업에 대한 자세한 내용은 을 참조하십시오. 워크플로를 사용하여 AWS CDK 앱 부트스트래핑하기

     부트스트래핑에 대한 자세한 내용은 개발자 안내서의 부트스트래핑을 참조하십시오.AWS Cloud Development Kit (AWS CDK)

Visual

비주얼 편집기를 사용하여 'AWS CDK 배포' 작업을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. AWS CDK 배포 작업을 검색하고 다음 중 하나를 수행하십시오.

   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - AWS CDK 배포를 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 액션의 소스 코드를 보려면 다운로드를 선택합니다.

- 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

10. 입력 및 구성 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오AWS CDK '배포' 작업 YAML. 이 참조는 편집기와 시각적 편집기에 모두 나타나는 각 필드 (및 해당 YAML 속성 값) 에 대한 자세한 정보를 제공합니다. YAML
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

> **Note**
>
> 오류로 인해 AWS CDK 배포 작업이 실패하는 경우 `npm install` 오류 수정 방법에 대한 자세한 내용은 을 참조하십시오"npm install" 오류를 수정하려면 어떻게 해야 하나요?.

YAML

편집기를 사용하여 'AWS CDK 배포' 작업을 추가하려면 YAML

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. AWS CDK 배포 작업을 검색하고 다음 중 하나를 수행하십시오.

   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

- AWS CDK 배포를 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
  - (선택 사항) 액션의 소스 코드를 보려면 다운로드를 선택합니다.
  - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

10. 필요에 따라 YAML 코드의 속성을 수정하십시오. 사용 가능한 각 속성에 대한 설명은 에 나와 AWS CDK '배포' 작업 YAML 있습니다.
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

> **Note**
>
> 오류로 인해 AWS CDK 배포 작업이 실패하는 경우 `npm install` 오류 수정 방법에 대한 자세한 내용은 을 참조하십시오"npm install" 오류를 수정하려면 어떻게 해야 하나요?.

AWS CDK '배포' 변수

AWS CDK 배포 작업은 런타임에 다음 변수를 생성하고 설정합니다. 이러한 변수를 사전 정의된 변수라고 합니다.

워크플로우에서 이러한 변수를 참조하는 방법에 대한 자세한 내용은 을 참조하십시오. 사전 정의된 변수 사용

키	값
stack-id	워크플로 실행 중에 배포된 AWS CDK 애플리케이션 스택의 Amazon 리소스 이름 (ARN). 예제: `arn:aws:cloudformation:us-west-2:111122223333:stack/codecatalyst-cdk-app-stack/6aad4380-100a-11ec-a10a-03b8a84d40df`
배포 플랫폼	배포 플랫폼의 이름.

키	값
	로 하드코딩했습니다. `AWS:CloudFormation`
region	워크플로 실행 중에 AWS 리전 배포된 지역 코드입니다. 예제: `us-west-2`
SKIP-DEPLOYMENT	값이 0이면 워크플로 실행 중에 AWS CDK 애플리케이션 스택 배포를 `true` 건너뛰었음을 나타냅니다. 마지막 배포 이후 스택에 변경 사항이 없으면 스택 배포를 건너뛰게 됩니다. 이 변수는 값이 인 경우에만 생성됩니다. `true` 로 하드코딩됨. `true`
AWS CloudFormation variables	AWS CDK 배포 작업은 이전에 나열된 변수를 생성하는 것 외에도 CloudFormation출력 변수를 후속 워크플로 작업에 사용할 워크플로 변수로 노출합니다. 기본적으로 작업은 찾은 처음 4개 (또는 그 이하) 의 CloudFormation 변수만 노출합니다. 노출되는 변수를 확인하려면 AWS CDK 배포 작업을 한 번 실행한 다음 실행 세부 정보 페이지의 변수 탭을 살펴보세요. 변수 탭에 나열된 변수가 원하는 것과 다르면 `CfnOutputVariables` YAML 속성을 사용하여 다른 변수를 구성할 수 있습니다. 자세한 내용은 의 CfnOutputVariables 속성 설명을 참조하십시오AWS CDK '배포' 작업 YAML.

AWS CDK '배포' 작업 YAML

다음은 AWS CDK 배포 작업의 YAML 정의입니다. 이 작업을 사용하는 방법을 알아보려면 을 참조하십시오워크플로를 통한 AWS CDK 앱 배포.

이 작업 정의는 광범위한 워크플로 정의 파일 내에 섹션으로 존재합니다. 이 파일에 대한 자세한 내용은 워크플로우 YAML 정의을 참조합니다.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

```
# The workflow definition starts here.
# See ### ## for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  CDKDeploy_nn:
    Identifier: aws/cdk-deploy@v1
    DependsOn:
      - CDKBootstrap
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - artifact-name
    Outputs:
      Artifacts:
        - Name: cdk_artifact
          Files:
            - "cdk.out/**/*"
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
```

```
      StackName: my-cdk-stack
      Region: us-west-2
      Tags: '{"key1": "value1", "key2": "value2"}'
      Context: '{"key1": "value1", "key2": "value2"}'
      CdkCliVersion: version
      CdkRootPath: directory-containing-cdk.json-file
      CfnOutputVariables: '["CnfOutputKey1","CfnOutputKey2","CfnOutputKey3"]'
      CloudAssemblyRootPath: path-to-cdk.out
```

CDKDeploy

(필수)

액션의 이름을 지정합니다. 모든 작업 이름은 워크플로 내에서 고유해야 합니다. 액션 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 액션 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

기본값: CDKDeploy_nn.

해당 UI: 구성 탭/ 작업 이름

Identifier

(*CDKDeploy*/Identifier)

(필수)

작업을 식별합니다. 버전을 변경하려는 경우가 아니면 이 속성을 변경하지 마십시오. 자세한 내용은 사용할 액션 버전 지정 단원을 참조하십시오.

기본값: aws/cdk-deploy@v1.

해당 UI: 워크플로 다이어그램/ CDKDeploy _n/ aws/cdk-deploy @v1 라벨

DependsOn

(*CDKDeploy*/DependsOn)

(선택 사항)

배포 작업을 실행하기 위해 성공적으로 실행되어야 하는 작업 또는 작업 그룹을 지정합니다.AWS CDK 다음과 같이 DependsOn 속성에 AWS CDK 부트스트랩 작업을 지정하는 것이 좋습니다.

```
CDKDeploy:
  Identifier: aws/cdk-deploy@v1
  DependsOn:
    - CDKBootstrap
```

> **Note**
>
> 부트스트래핑은 앱을 배포하기 위한 필수 전제 조건입니다. AWS CDK 워크플로에 AWS CDK 부트스트랩 작업을 포함하지 않는 경우 배포 작업을 실행하기 전에 AWS CDK 부트스트랩 스택을 배포하는 다른 방법을 찾아야 합니다.AWS CDK 자세한 설명은 워크플로를 통한 AWS CDK 앱 배포에서 AWS CDK '배포' 작업 추가 섹션을 참조하십시오.

'종속 대상' 기능에 대한 자세한 내용은 을 참조하십시오. 시퀀싱 액션

해당 UI: 입력 탭/ 기준 - 선택 사항

Compute

(*CDKDeploy*/Compute)

(선택 사항)

워크플로 작업을 실행하는 데 사용되는 컴퓨팅 엔진. 워크플로 수준 또는 작업 수준에서 컴퓨팅을 지정할 수 있지만 둘 다에서 지정할 수는 없습니다. 워크플로 수준에서 지정된 경우 컴퓨팅 구성은 워크플로에 정의된 모든 작업에 적용됩니다. 워크플로 수준에서는 동일한 인스턴스에서 여러 작업을 실행할 수도 있습니다. 자세한 내용은 작업 간 컴퓨팅 공유 단원을 참조하십시오.

해당 UI: 없음

Type

(*CDKDeploy*/Compute/Type)

(포함된 Compute 경우 필수)

컴퓨팅 엔진 유형. 다음 값 중 하나를 사용할 수 있습니다.

- EC2(비주얼 에디터) 또는 EC2 (YAML에디터)

  작업 실행 중 유연성을 위해 최적화되었습니다.

- Lambda (비주얼 에디터) 또는 (에디터) Lambda YAML

  작업 시작 속도를 최적화했습니다.

컴퓨팅 유형에 대한 자세한 정보는 컴퓨팅 유형을 참고하세요.

해당 UI: 구성 탭/고급 - 선택 사항/ 컴퓨팅 유형

Fleet

(*CDKDeploy*/Compute/Fleet)

(선택 사항)

워크플로 또는 워크플로 작업을 실행할 시스템 또는 플릿을 지정합니다. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다. 온디맨드 플릿의 예:,. `Linux.x86-64.Large` `Linux.x86-64.XLarge` 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

생략된 경우 기본값은 `Fleet` 입니다. `Linux.x86-64.Large`

해당 UI: 구성 탭/고급 - 선택 사항/ 컴퓨팅 플릿

Timeout

(*CDKDeploy*/Timeout)

(필수)

작업이 종료되기 전에 작업을 실행할 수 있는 시간을 분 (YAML편집기) 또는 시간과 분 (비주얼 편집기) 단위로 지정합니다. CodeCatalyst 최소값은 5분이고 최대값은 에 설명되어 의 워크플로우 할당량 CodeCatalyst 있습니다. 기본 타임아웃은 최대 타임아웃과 동일합니다.

해당 UI: 구성 탭/ 타임아웃 - 선택 사항

Inputs

(*CDKDeploy*/Inputs)

(선택 사항)

이 `Inputs` 섹션에서는 워크플로우 실행 중에 `CDKDeploy` 필요한 데이터를 정의합니다.

> **Note**
>
> 각 AWS CDK 배포 작업에는 하나의 입력 (소스 또는 아티팩트) 만 허용됩니다.

해당 UI: 입력 탭

Sources

(*CDKDeploy*/Inputs/Sources)

(배포하려는 AWS CDK 앱이 소스 리포지토리에 저장되어 있는 경우 필수)

AWS CDK 앱이 소스 리포지토리에 저장되어 있는 경우 해당 소스 리포지토리의 레이블을 지정하십시오. AWS CDK 배포 작업은 배포 프로세스를 시작하기 전에 이 저장소에서 앱을 합성합니다. 현재 지원되는 유일한 레이블은 입니다`WorkflowSource`.

AWS CDK 앱이 소스 리포지토리에 포함되어 있지 않은 경우 다른 작업에 의해 생성된 아티팩트에 있어야 합니다.

소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

해당 UI: 입력 탭/ 소스 - 선택사항

Artifacts - input

(*CDKDeploy*/Inputs/Artifacts)

(배포하려는 AWS CDK 앱이 이전 작업의 출력 아티팩트에 저장되어 있는 경우 필수)

이전 작업에서 생성된 아티팩트에 AWS CDK 앱이 포함되어 있는 경우 여기에 해당 아티팩트를 지정하십시오. AWS CDK 배포 작업은 배포 프로세스를 시작하기 전에 지정된 아티팩트의 앱을 CloudFormation 템플릿으로 합성합니다. AWS CDK 앱이 아티팩트에 포함되어 있지 않은 경우 소스 저장소에 있어야 합니다.

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 입력 탭/ 아티팩트 - 선택 사항

Outputs

(*CDKDeploy*/Outputs)

(선택 사항)

워크플로우 실행 중 작업에 의해 출력되는 데이터를 정의합니다.

해당 UI: 출력 탭

Artifacts - output

(*CDKDeploy*/Outputs/Artifacts

(선택 사항)

액션으로 생성된 아티팩트를 지정합니다. 이러한 아티팩트를 다른 작업의 입력으로 참조할 수 있습니다.

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 출력 탭/ 아티팩트

Name

(*CDKDeploy*/Outputs/Artifacts/Name)

(포함된 Artifacts - output 경우 필수)

런타임 시 AWS CDK 배포 작업에 의해 합성되는 AWS CloudFormation 템플릿을 포함할 아티팩트의 이름을 지정합니다. 기본값은 `cdk_artifact`입니다. 아티팩트를 지정하지 않으면 액션은 템플릿을 합성하지만 아티팩트에 저장하지는 않습니다. 테스트 또는 문제 해결을 위해 합성된 템플릿을 아티팩트에 저장하여 기록을 보존하는 것을 고려해 보십시오.

해당 UI: 출력 탭/아티팩트/아티팩트 추가/빌드 아티팩트 이름

Files

(*CDKDeploy*/Outputs/Artifacts/Files)

Artifacts - output(포함된 경우 필수)

아티팩트에 포함할 파일을 지정합니다. AWS CDK 앱의 합성된 "cdk.out/**/*" AWS CloudFormation 템플릿을 포함하도록 지정해야 합니다.

> **Note**
>
> cdk.out합성된 파일이 저장되는 기본 디렉토리입니다. cdk.json파일 이외의 cdk.out 출력 디렉토리를 지정한 경우 대신 cdk.out 여기에 해당 디렉토리를 지정하십시오.

해당 UI: 출력 탭/아티팩트/아티팩트 추가/빌드로 생성된 파일

Environment

(*CDKDeploy*/Environment)

(필수)

작업에 사용할 환경을 지정합니다. CodeCatalyst 작업은 선택한 환경에 VPC 지정된 AWS 계정 및 선택적 Amazon에 연결됩니다. 작업은 환경에 지정된 기본 IAM 역할을 사용하여 Amazon에 연결하고 Amazon VPC 연결에 지정된 IAM 역할을 사용하여 Amazon에 연결합니다VPC. AWS 계정

> **Note**
>
> 기본 IAM 역할에 작업에 필요한 권한이 없는 경우 다른 역할을 사용하도록 작업을 구성할 수 있습니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.

환경에 대한 자세한 내용은 및 에 AWS 계정 배포 VPCs 및 을 참조하십시오환경 생성.

해당 UI: 구성 탭/ 환경

Name

(*CDKDeploy*/Environment/Name)

(포함된 Environment 경우 필수)

작업에 연결할 기존 환경의 이름을 지정합니다.

해당 UI: 구성 탭/ 환경

Connections

(*CDKDeploy*/Environment/Connections)

(액션의 최신 버전에서는 선택 사항, 이전 버전에서는 필수)

액션과 연결할 계정 연결을 지정합니다. 에서 최대 1개의 계정 연결을 지정할 수 `Environment` 있습니다.

계정 연결을 지정하지 않은 경우:

- 작업은 CodeCatalyst 콘솔 환경에 지정된 AWS 계정 연결 및 기본 IAM 역할을 사용합니다. 환경에 계정 연결 및 기본 IAM 역할을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.
- 기본 IAM 역할에는 작업에 필요한 정책 및 권한이 포함되어야 합니다. 이러한 정책 및 권한이 무엇인지 확인하려면 작업 YAML 정의 설명서에서 역할 속성에 대한 설명을 참조하십시오.

계정 연결에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정. 환경에 계정 연결을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

해당 UI: 액션 버전에 따라 다음 중 하나

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Name

(*CDKDeploy*/Environment/Connections/Name)

(포함된 경우 필수Connections)

계정 연결 이름을 지정합니다.

해당 UI: 액션 버전에 따라 다음 중 하나:

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Role

(*CDKDeploy*/Environment/Connections/Role)

(포함된 경우 필수Connections)

계정 연결 이름을 지정합니다.

AWS CDK 배포 작업에서 AWS CDK 애플리케이션 스택에 AWS 액세스하고 배포하는 데 사용하는 IAM 역할의 이름을 지정합니다. CodeCatalyst 스페이스에 역할을 추가했는지, 역할에 다음 정책이 포함되어 있는지 확인하십시오.

IAM역할을 지정하지 않으면 CodeCatalyst 콘솔 환경에 나열된 기본 IAM 역할이 작업에 사용됩니다. 환경에서 기본 역할을 사용하는 경우 해당 역할에 다음 정책이 적용되는지 확인하십시오.

- 다음 권한 정책:

> **Warning**
>
> 권한을 다음 정책에 표시된 권한으로 제한하십시오. 더 광범위한 권한이 있는 역할을 사용하면 보안 위험이 발생할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeChangeSet",
                "cloudformation:DescribeStacks",
                "cloudformation:ListStackResources"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::aws-account:role/cdk-*"
        }
    ]
}
```

- 다음과 같은 사용자 지정 신뢰 정책:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

> **Note**
>
> 원하는 경우 이 작업에 CodeCatalystWorkflowDevelopmentRole-*spaceName* 역할을 사용할 수 있습니다. 이에 대한 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기 섹션을 참조하세요. CodeCatalystWorkflowDevelopmentRole-*spaceName*역할에는 보안 위험을 초래할 수 있는 전체 액세스 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다.

해당 UI: 액션 버전에 따라 다음 중 하나가 표시됩니다.

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 역할

Configuration

(*CDKDeploy*/Configuration)

(필수)

작업의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭

StackName

(*CDKDeploy*/Configuration/StackName)

(필수)

AWS CDK 앱 디렉토리의 진입점 파일에 나타나는 AWS CDK 앱 스택의 `bin` 이름입니다. 다음 예제는 스택 이름이 강조 표시된 TypeScript 진입점 파일의 콘텐츠를 보여줍니다.*red italics*. 진입점 파일이 다른 언어로 되어 있으면 비슷하게 보일 것입니다.

```
import * as cdk from 'aws-cdk-lib';
import { CdkWorksopTypescriptStack } from '../lib/cdk_workshop_typescript-stack';

const app = new cdk.App();
new CdkWorkshopTypescriptStack(app, 'CdkWorkshopTypescriptStack');
```

스택을 하나만 지정할 수 있습니다.

> **Tip**
>
> 스택이 여러 개 있는 경우 중첩된 스택으로 부모 스택을 만들 수 있습니다. 그런 다음 이 작업에서 상위 스택을 지정하여 모든 스택을 배포할 수 있습니다.

해당 UI: 구성 탭/ 스택 이름

Region

(*CDKDeploy*/Configuration/Region)

(선택 사항)

AWS CDK 애플리케이션 AWS 리전 스택을 배포할 대상을 지정합니다. 리전 코드 목록은 리전 엔드포인트를 참조하세요.

지역을 지정하지 않으면 AWS CDK 코드에 지정된 지역에 배포 작업이 AWS CDK 배포됩니다. 자세한 내용은 AWS Cloud Development Kit (AWS CDK) 개발자 안내서의 환경을 참조하십시오.

해당 UI: 구성 탭/ 지역

Tags

(*CDKDeploy*/Configuration/Tags)

(선택 사항)

AWS CDK 애플리케이션 스택의 AWS 리소스에 적용할 태그를 지정합니다. 태그는 스택 자체뿐만 아니라 스택의 개별 리소스에도 적용됩니다. 태깅에 대한 자세한 내용은 AWS Cloud Development Kit (AWS CDK) 개발자 안내서의 태깅을 참조하십시오.

해당 UI: 구성 탭/고급 - 선택 사항/ 태그

Context

(*CDKDeploy*/Configuration/Context)

(선택 사항)

애플리케이션 스택과 연결할 컨텍스트를 키-값 쌍의 형태로 지정합니다. AWS CDK 컨텍스트에 대한 자세한 내용은 개발자 안내서의 런타임 컨텍스트를 참조하십시오.AWS Cloud Development Kit (AWS CDK)

해당 UI: 구성 탭/고급 - 선택적/ 컨텍스트

CdkCliVersion

(*CDKDeploy*/Configuration/CdkCliVersion)

(선택 사항)

이 속성은 AWS CDK 배포 작업의 버전 1.0.13 이상 및 부트스트랩 작업의 버전 1.0.8 이상에서 사용할 수 있습니다.AWS CDK

다음 중 하나를 지정하세요.

- 이 작업에 사용할 AWS Cloud Development Kit (AWS CDK) 명령줄 인터페이스 (CLI) 의 정식 버전 (AWS CDK 툴킷이라고도 함). 예: `2.102.1`. 응용 프로그램을 빌드하고 배포할 때 일관성과 안정성을 보장하려면 정식 버전을 지정하는 것이 좋습니다.

  Or

- `latest`. 의 최신 기능 및 수정 사항을 `latest` 활용하도록 지정하는 것을 고려해 보십시오 CDKCLI.

작업은 의 지정된 버전 (또는 최신 버전) 을 CodeCatalyst 빌드 이미지에 다운로드한 다음 이 버전을 사용하여 CDK 응용 프로그램을 배포하거나 AWS 환경을 부트스트랩하는 데 필요한 명령을 실행합니다. AWS CDK CLI

사용할 수 있는 지원 CDK CLI 버전 목록은 AWS CDK 버전을 참조하십시오.

이 속성을 생략하면 액션은 다음 항목 중 하나에 설명된 기본 AWS CDK CLI 버전을 사용합니다.

- CDKCLI“AWS CDK 배포” 작업에 사용되는 버전
- CDKCLI“부트스트랩” 작업에 사용되는 버전AWS CDK

해당 UI: 구성 탭/ 버전AWS CDK CLI

CdkRootPath

(*CDKDeploy*/Configuration/CdkRootPath)

(선택 사항)

AWS CDK 프로젝트 `cdk.json` 파일이 들어 있는 디렉토리의 경로입니다. AWS CDK 배포 작업은 이 폴더에서 실행되며 해당 작업으로 생성된 모든 출력이 이 디렉터리에 추가됩니다. 지정하지 않으면 AWS CDK 배포 작업은 `cdk.json` 파일이 AWS CDK 프로젝트의 루트에 있다고 가정합니다.

해당 UI: cdk.json이 있는 구성 탭/ 디렉터리

CfnOutputVariables

(*CDKDeploy*/Configuration/CfnOutputVariables)

(선택 사항)

AWS CDK 애플리케이션 코드에서 `CfnOutput` 워크플로 출력 변수로 노출하려는 구문을 지정합니다. 그런 다음 워크플로의 후속 작업에서 워크플로 출력 변수를 참조할 수 있습니다. 의 변수에 대한 자세한 내용은 CodeCatalyst 을 참조하십시오워크플로우에서 변수 사용.

예를 들어, AWS CDK 애플리케이션 코드가 다음과 같은 경우:

```
import { Duration, Stack, StackProps, CfnOutput, RemovalPolicy} from 'aws-cdk-lib';
import * as dynamodb from 'aws-cdk-lib/aws-dynamodb';
import * as s3 from 'aws-cdk-lib/aws-s3';
import { Construct } from 'constructs';
import * as cdk from 'aws-cdk-lib';
export class HelloCdkStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);
    const bucket = new s3.Bucket(this, 'my-bucket', {
      removalPolicy: RemovalPolicy.DESTROY,
    });
    new CfnOutput(this, 'bucketName', {
      value: bucket.bucketName,
      description: 'The name of the s3 bucket',
      exportName: 'myBucket',
    });
    const table = new dynamodb.Table(this, 'todos-table', {
      partitionKey: {name: 'todoId', type: dynamodb.AttributeType.NUMBER},
      billingMode: dynamodb.BillingMode.PAY_PER_REQUEST,
      removalPolicy: RemovalPolicy.DESTROY,
    })
    new CfnOutput(this, 'tableName', {
      value: table.tableName,
      description: 'The name of the dynamodb table',
      exportName: 'myDynamoDbTable',
    });
    ...
  }
}
```

... 그리고 `CfnOutputVariables` 재산은 다음과 같습니다.

```
Configuration:
  ...
  CfnOutputVariables: '["bucketName","tableName"]'
```

... 그러면 작업이 다음과 같은 워크플로 출력 변수를 생성합니다.

키	값
bucketName	`bucket.bucketName`

키	값
tableName	`table.tableName`

그러면 후속 작업에서 `bucketName` 및 `tableName` 변수를 참조할 수 있습니다. 후속 작업에서 워크플로 출력 변수를 참조하는 방법을 알아보려면 을 참조하십시오사전 정의된 변수 참조.

`CfnOutputVariables`속성에 `CfnOutput` 구문을 지정하지 않으면 액션은 찾은 처음 4개 (또는 그 이하) 의 CloudFormation 출력 변수를 워크플로 출력 변수로 노출합니다. 자세한 내용은 AWS CDK '배포' 변수 단원을 참조하십시오.

> **Tip**
>
> 작업이 생성하는 모든 CloudFormation 출력 변수 목록을 가져오려면 AWS CDK 배포 작업이 포함된 워크플로를 한 번 실행한 다음 작업의 로그 탭을 살펴보십시오. 로그에는 AWS CDK 앱과 관련된 모든 CloudFormation 출력 변수 목록이 포함됩니다. 모든 CloudFormation 변수가 무엇인지 알고 나면 `CfnOutputVariables` 속성을 사용하여 워크플로 출력 변수로 변환할 변수를 지정할 수 있습니다.

AWS CloudFormation 출력 변수에 대한 자세한 내용은 AWS Cloud Development Kit (AWS CDK) API 참조의 class CfnOutput (`CfnOutput`construct) 에서 사용할 수 있는 구문 설명서를 참조하십시오.

해당 UI: 구성 탭/ AWS CloudFormation 출력 변수

CloudAssemblyRootPath

(*CDKDeploy*/Configuration/CloudAssemblyRootPath)

(선택 사항)

`cdk synth`작업을 사용하여 이미 AWS CDK 앱 스택을 클라우드 어셈블리로 합성했다면 클라우드 어셈블리 디렉토리 (`cdk.out`) 의 루트 경로를 지정하세요. 지정된 클라우드 어셈블리 디렉터리에 있는 AWS CloudFormation 템플릿은 AWS CDK 배포 작업을 통해 **`cdk deploy --app`** 명령어를 AWS 계정 사용하여 사용자에게 배포됩니다. `--app`옵션이 있는 경우 `cdk synth` 작업이 수행되지 않습니다.

클라우드 어셈블리 디렉터리를 지정하지 않는 경우 AWS CDK 배포 작업은 `--app` 옵션 없이 `cdk deploy` 명령을 실행합니다. 이 `--app` 옵션이 없으면 `cdk deploy` 작업 시 AWS CDK 앱이 신디사이저 (`cdk synth`) 되고 배포됩니다. AWS 계정

런타임에 'AWS CDK 배포' 작업으로 합성을 수행할 수 있는데 기존의 합성된 클라우드 어셈블리를 지정하는 이유는 무엇입니까?

기존의 합성된 클라우드 어셈블리를 다음과 같이 지정할 수 있습니다.

- “AWS CDK 배포” 작업이 실행될 때마다 정확히 동일한 리소스 세트를 배포해야 합니다.

  클라우드 어셈블리를 지정하지 않으면 AWS CDK 배포 작업이 실행되는 시기에 따라 다른 파일을 합성하고 배포할 수 있습니다. 예를 들어 AWS CDK 배포 작업은 테스트 단계에서 한 세트의 종속성을 사용하여 클라우드 어셈블리를 합성하고 프로덕션 단계에서 다른 종속성 세트를 합성할 수 있습니다 (해당 종속성이 단계 간에 변경된 경우). 테스트된 항목과 배포된 항목 간에 정확한 패리티를 보장하려면 합성을 한 번 수행한 다음 Path to cloud assembly 디렉터리 필드 (비주얼 편집기) 또는 **`CloudAssemblyRootPath`** 속성 (YAML편집기) 을 사용하여 이미 합성된 클라우드 어셈블리를 지정하는 것이 좋습니다.
- 비표준 패키지 관리자 및 도구를 앱과 함께 사용하십시오. AWS CDK

  작업 중에 AWS CDK 배포 **`synth`** 작업은 npm 또는 pip와 같은 표준 도구를 사용하여 앱을 실행하려고 시도합니다. 액션이 이러한 도구를 사용하여 앱을 성공적으로 실행할 수 없는 경우 합성이 이루어지지 않고 작업이 실패합니다. 이 문제를 해결하려면 앱 `cdk.json` 파일에서 앱을 성공적으로 실행하는 데 필요한 정확한 명령을 지정한 다음 AWS CDK 배포 작업이 포함되지 않은 방법을 사용하여 앱을 합성할 수 있습니다. AWS CDK 클라우드 어셈블리가 생성되면 AWS CDK 배포 작업의 Path to cloud assembly 디렉터리 필드 (비주얼 편집기) 또는 `CloudAssemblyRootPath` 속성 (YAML편집기) 에서 지정할 수 있습니다.

앱 설치 및 실행 명령을 포함하도록 `cdk.json` 파일을 구성하는 방법에 대한 자세한 내용은 AWS CDK 앱 명령 지정을 참조하십시오.

`cdk deploy`및 *`cdk synth`* 명령과 *`--app`* 옵션에 대한 자세한 내용은 개발자 안내서의 스택 배포, 스택 합성 및 합성 건너뛰기를 참조하십시오.AWS Cloud Development Kit (AWS CDK)

클라우드 어셈블리에 대한 자세한 내용은 참조의 클라우드 어셈블리를 참조하십시오.AWS Cloud Development Kit (AWS CDK) API

해당 UI: 구성 탭/ 클라우드 어셈블리 디렉토리 경로

워크플로를 사용하여 AWS CDK 앱 부트스트래핑하기

이 섹션에서는 워크플로를 사용하여 AWS CDK 애플리케이션을 부트스트랩하는 방법을 설명합니다. CodeCatalyst 이 작업을 수행하려면 워크플로에 AWS CDK 부트스트랩 작업을 추가해야 합니다. AWS

CDK 부트스트랩 작업은 최신 템플릿을 사용하여 사용자 AWS 환경에 부트스트랩 스택을 프로비저닝합니다. 부트스트랩 스택이 이미 있는 경우 작업은 필요한 경우 해당 스택을 업데이트합니다. 앱을 배포하려면 부트스트랩 AWS 스택이 있어야 합니다. AWS CDK

부트스트래핑에 대한 자세한 내용은 개발자 안내서의 부트스트래핑을 참조하십시오.AWS Cloud Development Kit (AWS CDK)

'부트스트랩' 작업 사용 시기AWS CDK

AWS CDK 앱을 배포하는 워크플로가 있고 부트스트랩 스택을 동시에 배포 (필요한 경우 업데이트) 하려는 경우 이 작업을 사용하십시오. 이 경우 앱을 배포하는 워크플로와 동일한 워크플로에 AWS CDK 부트스트랩 작업을 추가합니다. AWS CDK

다음 중 하나에 해당하는 경우에는 이 작업을 사용하지 마세요.

- 이미 다른 메커니즘을 사용하여 부트스트랩 스택을 배포했고, 이를 그대로 유지하려고 합니다 (업데이트 없음).
- 부트스트랩 작업에서 지원되지 않는 사용자 지정 부트스트랩 템플릿을 사용하려고 합니다AWS CDK .

AWS CDK '부트스트랩' 작업 작동 방식

AWS CDK 부트스트랩은 다음과 같이 작동합니다.

1. 런타임 시 액션의 버전 1.0.7 이하를 지정한 경우 액션은 최신 버전 CDK CLI (AWS CDK Tookit이라고도 함) 을 빌드 이미지에 다운로드합니다. CodeCatalyst

   버전 1.0.8 이상을 지정한 경우 작업은 의 특정 버전과 함께 번들로 제공되므로 다운로드가 수행되지 않습니다. CDK CLI

2. 액션은 를 CDK CLI 사용하여 명령을 실행합니다. `cdk bootstrap` 이 명령은 개발자 안내서의 부트스트래핑 항목에 설명된 부트스트래핑 작업을 수행합니다.AWS Cloud Development Kit (AWS CDK)

CDKCLI“부트스트랩” 작업에 사용되는 버전AWS CDK

다음 표는 AWS CDK 부트스트랩 작업의 CDK CLI 여러 버전에서 기본적으로 사용되는 의 버전을 보여줍니다.

> **Note**
>
> 기본값을 재정의할 수 있습니다. 자세한 내용은 AWS CDK '부트스트랩' 액션 YAML의 CdkCliVersion 섹션을 참조하십시오.

AWS CDK '부트스트랩' 액션 버전	AWS CDK CLI버전
1.0.0 — 1.0.7	최신
1.0.8 이상	2.99.1

주제

- 예: 앱 부트스트래핑 AWS CDK
- AWS CDK '부트스트랩' 액션 추가
- AWS CDK '부트스트랩' 변수
- AWS CDK '부트스트랩' 액션 YAML

예: 앱 부트스트래핑 AWS CDK

부트스트랩 작업이 포함된 워크플로우는 예: 앱 AWS CDK 배포 의 워크플로를 통한 AWS CDK 앱 배포AWS CDK 를 참조하십시오.

AWS CDK '부트스트랩' 액션 추가

다음 지침에 따라 워크플로에 AWS CDK 부트스트랩 작업을 추가하십시오.

시작하기 전에

AWS CDK 부트스트랩 작업을 사용하려면 먼저 AWS CDK 앱을 준비해야 합니다. 부트스트랩 작업은 부트스트래핑 전에 AWS CDK 앱을 합성합니다. 에서 지원하는 모든 프로그래밍 언어로 앱을 작성할 수 있습니다. AWS CDK

AWS CDK 앱 파일을 다음 위치에서 사용할 수 있는지 확인하세요.

- CodeCatalyst 소스 리포지토리 또는

- 다른 워크플로 작업에 의해 생성된 CodeCatalyst 출력 아티팩트

Visual

비주얼 편집기를 사용하여 'AWS CDK 부트스트랩' 작업 추가하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. AWS CDK 부트스트랩 작업을 검색하고 다음 중 하나를 수행하십시오.
   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - AWS CDK 부트스트랩을 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서는
     - (선택 사항) 소스 보기를 선택하여 액션의 소스 코드를 확인합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.
10. 입력, 구성 및 출력 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오AWS CDK '부트스트랩' 액션 YAML. 이 참조는 편집기와 시각적 편집기에 모두 나타나는 각 필드 (및 해당 YAML 속성 값) 에 대한 자세한 정보를 제공합니다. YAML
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

> **Note**
>
> 오류로 인해 AWS CDK 부트스트랩 작업이 실패하는 경우 `npm install` 오류 수정 방법에 대한 자세한 내용은 을 참조하십시오"npm install" 오류를 수정하려면 어떻게 해야 하나요?.

YAML

편집기를 사용하여 'AWS CDK 부트스트랩' 작업을 추가하려면 YAML

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. AWS CDK 부트스트랩 작업을 검색하고 +를 선택하여 워크플로 다이어그램에 추가하고 해당 구성 창을 엽니다.
10. 필요에 따라 YAML 코드의 속성을 수정하십시오. 사용 가능한 각 속성에 대한 설명은 에 나와 AWS CDK '부트스트랩' 액션 YAML 있습니다.
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

> **Note**
>
> 오류로 인해 AWS CDK 부트스트랩 작업이 실패하는 경우 `npm install` 오류 수정 방법에 대한 자세한 내용은 을 참조하십시오"npm install" 오류를 수정하려면 어떻게 해야 하나요?.

AWS CDK '부트스트랩' 변수

AWS CDK 부트스트랩 작업은 런타임에 다음 변수를 생성하고 설정합니다. 이러한 변수를 사전 정의된 변수라고 합니다.

워크플로우에서 이러한 변수를 참조하는 방법에 대한 자세한 내용은 을 참조하십시오. 사전 정의된 변수 사용

키	값
배포 플랫폼	배포 플랫폼의 이름. 로 하드코딩했습니다. `AWS:CloudFormation`
region	워크플로 실행 중에 AWS CDK 부트스트랩 스택이 배포된 지역 코드입니다. AWS 리전 예제: `us-west-2`
stack-id	배포된 AWS CDK 부트스트랩 스택의 Amazon 리소스 이름 (ARN) 예제: `arn:aws:cloudformation:us-west-2:111122223333:stack/codecatalyst-cdk-bootstrap-stack/6aad4380-100a-11ec-a10a-03b8a84d40df`
SKIP-DEPLOYMENT	값이 0이면 워크플로 실행 중에 AWS CDK 부트스트랩 스택 배포를 `true` 건너뛰었음을 나타냅니다. 마지막 배포 이후 스택에 변경 사항이 없으면 스택 배포를 건너뛰게 됩니다. 이 변수는 해당 값이 인 경우에만 생성됩니다. `true` 로 하드코딩됨. `true`

AWS CDK '부트스트랩' 액션 YAML

다음은 부트스트랩 작업의 YAML AWS CDK 정의입니다. 이 작업을 사용하는 방법을 알아보려면 을 참조하십시오워크플로를 사용하여 AWS CDK 앱 부트스트래핑하기.

이 작업 정의는 광범위한 워크플로 정의 파일 내에 섹션으로 존재합니다. 이 파일에 대한 자세한 내용은 워크플로우 YAML 정의을 참조합니다.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

```
# The workflow definition starts here.
# See ### ## for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  CDKBootstrapAction_nn:
    Identifier: aws/cdk-bootstrap@v1
    DependsOn:
      - action-name
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - artifact-name
    Outputs:
      Artifacts:
        - Name: cdk_bootstrap_artifacts
          Files:
            - "cdk.out/**/*"
    Environment:
```

```
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
      Region: us-west-2
      CdkCliVersion: version
```

CDKBootstrapAction

(필수)

액션의 이름을 지정합니다. 모든 작업 이름은 워크플로 내에서 고유해야 합니다. 액션 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 액션 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

기본값: `CDKBootstrapAction_nn`.

해당 UI: 구성 탭/ 작업 표시 이름

Identifier

(*CDKBootstrapAction*/Identifier)

(필수)

작업을 식별합니다. 버전을 변경하려는 경우가 아니면 이 속성을 변경하지 마십시오. 자세한 내용은 사용할 액션 버전 지정 단원을 참조하십시오.

기본값: `aws/cdk-bootstrap@v1`.

해당 UI: 워크플로 다이어그램/ CDKBootstrapAction _nn/ aws/cdk-bootstrap @v1 라벨

DependsOn

(*CDKBootstrapAction*/DependsOn)

(선택 사항)

이 작업을 실행하기 위해 성공적으로 실행되어야 하는 작업, 작업 그룹 또는 게이트를 지정하십시오.

'종속 조건' 기능에 대한 자세한 내용은 을 참조하십시오. 시퀀싱 액션

해당 UI: 입력 탭/ 기준 - 선택 사항

Compute

(*CDKBootstrapAction*/Compute)

(선택 사항)

워크플로 작업을 실행하는 데 사용되는 컴퓨팅 엔진. 워크플로 수준 또는 작업 수준에서 컴퓨팅을 지정할 수 있지만 둘 다에서 지정할 수는 없습니다. 워크플로 수준에서 지정된 경우 컴퓨팅 구성은 워크플로에 정의된 모든 작업에 적용됩니다. 워크플로 수준에서는 동일한 인스턴스에서 여러 작업을 실행할 수도 있습니다. 자세한 내용은 작업 간 컴퓨팅 공유 단원을 참조하십시오.

해당 UI: 없음

Type

(*CDKBootstrapAction*/Compute/Type)

(포함된 Compute 경우 필수)

컴퓨팅 엔진 유형. 다음 값 중 하나를 사용할 수 있습니다.

- EC2(비주얼 에디터) 또는 EC2 (YAML에디터)

  작업 실행 중 유연성을 위해 최적화되었습니다.
- Lambda (비주얼 에디터) 또는 (에디터) Lambda YAML

  작업 시작 속도를 최적화했습니다.

컴퓨팅 유형에 대한 자세한 정보는 컴퓨팅 유형을 참고하세요.

해당 UI: 구성 탭/고급 - 선택 사항/ 컴퓨팅 유형

Fleet

(*CDKBootstrapAction*/Compute/Fleet)

(선택 사항)

워크플로 또는 워크플로 작업을 실행할 시스템 또는 플릿을 지정합니다. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다.

온디맨드 플릿의 예:,. `Linux.x86-64.Large` `Linux.x86-64.XLarge` 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

생략된 경우 기본값은 `Fleet` 입니다. `Linux.x86-64.Large`

해당 UI: 구성 탭/고급 - 선택 사항/ 컴퓨팅 플릿

Timeout

(*CDKBootstrapAction*/Timeout)

(필수)

작업이 종료되기 전에 작업을 실행할 수 있는 시간을 분 (YAML편집기) 또는 시간과 분 (비주얼 편집기) 단위로 지정합니다. CodeCatalyst 최소값은 5분이고 최대값은 에 설명되어 의 워크플로우 할당량 CodeCatalyst 있습니다. 기본 타임아웃은 최대 타임아웃과 동일합니다.

해당 UI: 구성 탭/ 타임아웃 - 선택 사항

Inputs

(*CDKBootstrapAction*/Inputs)

(선택 사항)

이 `Inputs` 섹션에서는 워크플로 실행 중에 AWS CDK 부트스트랩 작업에 필요한 데이터를 정의합니다.

해당 UI: 입력 탭

> **Note**
>
> 각 AWS CDK 부트스트랩 작업에는 하나의 입력 (소스 또는 아티팩트) 만 허용됩니다.

Sources

(*CDKBootstrapAction*/Inputs/Sources)

(AWS CDK 앱이 소스 리포지토리에 저장되어 있는 경우 필수)

AWS CDK 앱이 소스 리포지토리에 저장되어 있는 경우 해당 소스 리포지토리의 레이블을 지정하십시오. AWS CDK 부트스트랩 작업은 부트스트랩 프로세스를 시작하기 전에 이 저장소에서 앱을 합성합니다. 현재 지원되는 유일한 리포지토리 레이블은 입니다. `WorkflowSource`

AWS CDK 앱이 소스 리포지토리에 포함되어 있지 않은 경우 다른 작업에 의해 생성된 아티팩트에 있어야 합니다.

소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

해당 UI: 입력 탭/ 소스 - 선택사항

Artifacts - input

(*CDKBootstrapAction*/Inputs/Artifacts)

(AWS CDK 앱이 이전 작업의 출력 아티팩트에 저장되어 있는 경우 필수)

이전 작업에서 생성된 아티팩트에 AWS CDK 앱이 포함되어 있는 경우 여기에 해당 아티팩트를 지정하세요. AWS CDK 부트스트랩 작업은 부트스트랩 프로세스를 시작하기 전에 지정된 아티팩트의 앱을 CloudFormation 템플릿으로 합성합니다. AWS CDK 앱이 아티팩트에 포함되어 있지 않은 경우 소스 저장소에 있어야 합니다.

예제를 비롯한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 입력 탭/ 아티팩트 - 선택 사항

Outputs

(*CDKBootstrapAction*/Outputs)

(선택 사항)

워크플로우 실행 중 작업에 의해 출력되는 데이터를 정의합니다.

해당 UI: 출력 탭

Artifacts - output

(*CDKBootstrapAction*/Outputs/Artifacts)

(선택 사항)

액션으로 생성된 아티팩트를 지정합니다. 이러한 아티팩트를 다른 작업의 입력으로 참조할 수 있습니다.

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 출력 탭/ 아티팩트

Name

(*CDKBootstrapAction*/Outputs/Artifacts/Name)

(포함된 Artifacts - output 경우 필수)

런타임 시 AWS CDK 부트스트랩 작업에 의해 합성되는 AWS CloudFormation 템플릿을 포함할 아티팩트의 이름을 지정합니다. 기본값은 `cdk_bootstrap_artifacts`입니다. 아티팩트를 지정하지 않으면 액션은 템플릿을 합성하지만 아티팩트에 저장하지는 않습니다. 테스트 또는 문제 해결을 위해 합성된 템플릿을 아티팩트에 저장하여 기록을 보존하는 것을 고려해 보십시오.

해당 UI: 출력 탭/아티팩트/아티팩트 추가/빌드 아티팩트 이름

Files

(*CDKBootstrapAction*/Outputs/Artifacts/Files)

Artifacts - output(포함된 경우 필수)

아티팩트에 포함할 파일을 지정합니다. AWS CDK 앱의 합성된 `"cdk.out/**/*"` AWS CloudFormation 템플릿을 포함하도록 지정해야 합니다.

> **Note**
>
> `cdk.out`합성된 파일이 저장되는 기본 디렉토리입니다. `cdk.json`파일 이외의 `cdk.out` 출력 디렉토리를 지정한 경우 대신 `cdk.out` 여기에 해당 디렉토리를 지정하십시오.

해당 UI: 출력 탭/아티팩트/아티팩트 추가/빌드로 생성된 파일

Environment

(*CDKBootstrapAction*/Environment)

(필수)

작업에 사용할 환경을 지정합니다. CodeCatalyst 작업은 선택한 환경에 VPC 지정된 AWS 계정 및 선택적 Amazon에 연결됩니다. 작업은 환경에 지정된 기본 IAM 역할을 사용하여 Amazon에 연결하고 Amazon VPC 연결에 지정된 IAM 역할을 사용하여 Amazon에 연결합니다VPC. AWS 계정

> **Note**
>
> 기본 IAM 역할에 작업에 필요한 권한이 없는 경우 다른 역할을 사용하도록 작업을 구성할 수 있습니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.

환경에 대한 자세한 내용은 및 에 AWS 계정 배포 VPCs 및 을 참조하십시오환경 생성.

해당 UI: 구성 탭/ 환경

Name

(*CDKBootstrapAction*/Environment/Name)

(포함된 Environment 경우 필수)

작업에 연결할 기존 환경의 이름을 지정합니다.

해당 UI: 구성 탭/ 환경

Connections

(*CDKBootstrapAction*/Environment/Connections)

(새 버전의 액션에서는 선택 사항, 이전 버전에서는 필요)

작업에 연결할 계정 연결을 지정합니다. 에서 최대 1개의 계정 연결을 지정할 수 `Environment` 있습니다.

계정 연결을 지정하지 않은 경우:

- 작업은 CodeCatalyst 콘솔 환경에 지정된 AWS 계정 연결 및 기본 IAM 역할을 사용합니다. 환경에 계정 연결 및 기본 IAM 역할을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.
- 기본 IAM 역할에는 작업에 필요한 정책 및 권한이 포함되어야 합니다. 이러한 정책 및 권한이 무엇인지 확인하려면 작업 YAML 정의 설명서에서 역할 속성에 대한 설명을 참조하십시오.

계정 연결에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정. 환경에 계정 연결을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

해당 UI: 액션 버전에 따라 다음 중 하나

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Name

(*CDKBootstrapAction*/Environment/Connections/Name)

(포함된 경우 필수Connections)

계정 연결 이름을 지정합니다.

해당 UI: 액션 버전에 따라 다음 중 하나:

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Role

(*CDKBootstrapAction*/Environment/Connections/Role)

(포함된 경우 필수Connections)

AWS CDK 부트스트랩 작업이 부트스트랩 스택에 AWS 액세스하고 추가하는 데 사용하는 IAM 역할의 이름을 지정합니다. CodeCatalyst 스페이스에 역할을 추가했는지, 역할에 다음 정책이 포함되어 있는지 확인하십시오.

IAM역할을 지정하지 않으면 CodeCatalyst 콘솔 환경에 나열된 기본 IAM 역할이 작업에 사용됩니다. 환경에서 기본 역할을 사용하는 경우 해당 역할에 다음 정책이 적용되는지 확인하십시오.

> **Note**
>
> 다음 권한 정책에 표시된 권한은 작성 시 부트스트래핑을 수행하는 데 `cdk bootstrap` 명령이 필요로 하는 권한입니다. 부트스트랩 명령이 변경되면 이러한 권한이 변경될 수 있습니다. AWS CDK

> **Warning**
>
> 이 역할은 AWS CDK 부트스트랩 작업에만 사용하십시오. 이는 매우 관대하며 다른 작업과 함께 사용하면 보안 위험이 발생할 수 있습니다.

- 다음 권한 정책:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "ssm:GetParameterHistory",
                "ecr:PutImageScanningConfiguration",
                "cloudformation:*",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "ssm:GetParameters",
                "iam:PutRolePolicy",
                "ssm:GetParameter",
                "ssm:DeleteParameters",
                "ecr:DeleteRepository",
                "ssm:PutParameter",
                "ssm:DeleteParameter",
                "iam:PassRole",
                "ecr:SetRepositoryPolicy",
                "ssm:GetParametersByPath",
                "ecr:DescribeRepositories",
                "ecr:GetLifecyclePolicy"
            ],
            "Resource": [
                "arn:aws:ssm:aws-region:aws-account:parameter/cdk-bootstrap/*",
                "arn:aws:cloudformation:aws-region:aws-account:stack/CDKToolkit/*",
                "arn:aws:ecr:aws-region:aws-account:repository/cdk-*",
                "arn:aws:iam::aws-account:role/cdk-*"
            ]
        },
        {
```

```
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "cloudformation:RegisterType",
                "cloudformation:CreateUploadBucket",
                "cloudformation:ListExports",
                "cloudformation:DescribeStackDriftDetectionStatus",
                "cloudformation:SetTypeDefaultVersion",
                "cloudformation:RegisterPublisher",
                "cloudformation:ActivateType",
                "cloudformation:ListTypes",
                "cloudformation:DeactivateType",
                "cloudformation:SetTypeConfiguration",
                "cloudformation:DeregisterType",
                "cloudformation:ListTypeRegistrations",
                "cloudformation:EstimateTemplateCost",
                "cloudformation:DescribeAccountLimits",
                "cloudformation:BatchDescribeTypeConfigurations",
                "cloudformation:CreateStackSet",
                "cloudformation:ListStacks",
                "cloudformation:DescribeType",
                "cloudformation:ListImports",
                "s3:*",
                "cloudformation:PublishType",
                "ecr:CreateRepository",
                "cloudformation:DescribePublisher",
                "cloudformation:DescribeTypeRegistration",
                "cloudformation:TestType",
                "cloudformation:ValidateTemplate",
                "cloudformation:ListTypeVersions"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 역할을 처음 사용할 때는 리소스 정책 설명에 다음 와일드카드를 사용하고, 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.

```
"Resource": "*"
```

- 다음과 같은 사용자 지정 신뢰 정책:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

> **Note**
>
> 원하는 경우 이 작업에 CodeCatalystWorkflowDevelopmentRole-*spaceName* 역할을 사용할 수 있습니다. 이에 대한 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기 섹션을 참조하세요. CodeCatalystWorkflowDevelopmentRole-*spaceName*역할에는 보안 위험을 초래할 수 있는 전체 액세스 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다.

해당 UI: 액션 버전에 따라 다음 중 하나가 표시됩니다.

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 역할

Configuration

(*CDKBootstrapAction*/Configuration)

(필수)

작업의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭

Region

(*CDKBootstrapAction*/Configuration/Region)

(필수)

부트스트랩 AWS 리전 스택을 배포할 대상을 지정합니다. 이 지역은 AWS CDK 앱이 배포된 지역과 일치해야 합니다. 리전 코드 목록은 리전 엔드포인트를 참조하세요.

해당 UI: 구성 탭/ 지역

CdkCliVersion

(*CDKBootstrapAction*/Configuration/CdkCliVersion)

(선택 사항)

이 속성은 AWS CDK 배포 작업의 버전 1.0.13 이상 및 부트스트랩 작업의 버전 1.0.8 이상에서 사용할 수 있습니다.AWS CDK

다음 중 하나를 지정하세요.

- 이 작업에 사용할 AWS Cloud Development Kit (AWS CDK) 명령줄 인터페이스 (CLI) 의 정식 버전 (AWS CDK 툴킷이라고도 함). 예: `2.102.1`. 응용 프로그램을 빌드하고 배포할 때 일관성과 안정성을 보장하려면 정식 버전을 지정하는 것이 좋습니다.

  Or

- `latest`. 의 최신 기능 및 수정 사항을 `latest` 활용하도록 지정하는 것을 고려해 보십시오 CDKCLI.

작업은 의 지정된 버전 (또는 최신 버전) 을 CodeCatalyst 빌드 이미지에 다운로드한 다음 이 버전을 사용하여 CDK 응용 프로그램을 배포하거나 AWS 환경을 부트스트랩하는 데 필요한 명령을 실행합니다. AWS CDK CLI

사용할 수 있는 지원 CDK CLI 버전 목록은 AWS CDK 버전을 참조하십시오.

이 속성을 생략하면 액션은 다음 항목 중 하나에 설명된 기본 AWS CDK CLI 버전을 사용합니다.

- CDKCLI“AWS CDK 배포” 작업에 사용되는 버전
- CDKCLI“부트스트랩” 작업에 사용되는 버전AWS CDK

해당 UI: 구성 탭/ 버전AWS CDK CLI

워크플로를 사용하여 Amazon S3에 파일 게시

이 단원에서는 CodeCatalyst 워크플로를 사용하여 Amazon S3에 파일을 게시하는 방법을 설명합니다. 이 작업을 수행하려면 Amazon S3 게시 작업을 워크플로에 추가해야 합니다. Amazon S3 게시 작업은 원본 디렉터리의 파일을 Amazon S3 버킷으로 복사합니다. 소스 디렉터리는 다음 위치에 있을 수 있습니다.

- 소스 리포지토리 또는
- 다른 워크플로 작업에 의해 생성된 출력 아티팩트

'Amazon S3 퍼블리시' 작업을 사용해야 하는 경우

다음과 같은 경우 이 작업을 사용하십시오.

- Amazon S3에 저장하려는 파일을 생성하는 워크플로가 있습니다.

  예를 들어 Amazon S3에서 호스팅하려는 정적 웹 사이트를 구축하는 워크플로가 있을 수 있습니다. 이 경우 워크플로에는 사이트 HTML 및 지원 파일을 빌드하는 빌드 작업과 Amazon S3에 파일을 복사하는 Amazon S3 게시 작업이 포함됩니다.
- Amazon S3에 저장하려는 파일이 들어 있는 소스 리포지토리가 있습니다.

  예를 들어 Amazon S3에 야간에 보관하려는 애플리케이션 소스 파일이 있는 소스 리포지토리가 있을 수 있습니다.

주제

- 예: Amazon S3에 파일 게시
- '아마존 S3 퍼블리시' 작업 추가
- '아마존 S3 퍼블리시' 작업 YAML

예: Amazon S3에 파일 게시

다음 예제 워크플로에는 빌드 작업과 함께 Amazon S3 게시 작업이 포함되어 있습니다. 워크플로는 정적 설명서 웹 사이트를 구축한 다음 호스팅되는 Amazon S3에 게시합니다. 워크플로는 순차적으로 실행되는 다음과 같은 구성 요소로 구성됩니다.

- 트리거 - 이 트리거는 소스 리포지토리에 변경 내용을 푸시할 때 워크플로가 자동으로 실행됩니다. 트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.
- 빌드 작업 (BuildDocs) — 트리거 시 작업은 정적 문서 웹 사이트 (mkdocs build) 를 빌드하고 관련 HTML 파일 및 지원 메타데이터를 라는 MyDocsSite 아티팩트에 추가합니다. 빌드 작업에 대한 자세한 내용은 을 참조하십시오워크플로를 사용한 구축.
- Amazon S3 게시 작업 (PublishToS3) — 빌드 작업이 완료되면 MyDocsSite 아티팩트의 사이트를 Amazon S3에 복사하여 호스팅합니다.

> **Note**
>
> 다음 워크플로 예제는 설명을 위한 것으로 추가 구성 없이는 작동하지 않습니다.

> **Note**
>
> 다음 YAML 코드에서는 원하는 경우 이 Connections: 섹션을 생략할 수 있습니다. 이 섹션을 생략하는 경우 환경의 기본 역할 필드에 지정된 IAM 역할에 Amazon S3 게시 작업에 필요한 권한 및 신뢰 정책이 포함되어 있는지 확인해야 합니다. 기본 IAM 역할을 사용하여 환경을 설정하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성. Amazon S3 게시 작업에 필요한 권한 및 신뢰 정책에 대한 자세한 내용은 의 Role 속성 설명을 참조하십시오'아마존 S3 퍼블리시' 작업 YAML.

```
Name: codecatalyst-s3-publish-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
```

```
Actions:
  BuildDocs:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: echo BuildDocs started on `date`
        - Run: pip install --upgrade pip
        - Run: pip install mkdocs
        - Run: mkdocs build
        - Run: echo BuildDocs completed on `date`
    Outputs:
      Artifacts:
      - Name: MyDocsSite
        Files:
          - "site/**/*"

  PublishToS3:
    Identifier: aws/s3-publish@v1
    Environment:
      Name: codecatalyst-s3-publish-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-s3-publish-build-role
    Inputs:
      Sources:
        - WorkflowSource
      Artifacts:
        - MyDocsSite
    Configuration:
      DestinationBucketName: my-bucket
      SourcePath: /artifacts/PublishToS3/MyDocSite/site
      TargetPath: my/docs/site
```

'아마존 S3 퍼블리시' 작업 추가

다음 지침을 사용하여 Amazon S3 게시 작업을 워크플로에 추가하십시오.

Visual

비주얼 편집기를 사용하여 'Amazon S3 게시' 작업을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. Amazon S3 게시 작업을 검색하고 다음 중 하나를 수행하십시오.

   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - Amazon S3 게시를 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 소스 보기를 선택하여 액션의 소스 코드를 확인합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.
10. 입력, 구성 및 출력 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오'아마존 S3 퍼블리시' 작업 YAML. 이 참조는 편집기와 시각적 편집기에 모두 나타나는 각 필드 (및 해당 YAML 속성 값) 에 대한 YAML 자세한 정보를 제공합니다.
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

편집기를 사용하여 'Amazon S3 게시' 작업을 추가하려면 YAML

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.

2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. Amazon S3 게시 작업을 검색하고 다음 중 하나를 수행하십시오.
   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - Amazon S3 게시를 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 소스 보기를 선택하여 액션의 소스 코드를 확인합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.
10. 필요에 따라 YAML 코드의 속성을 수정하십시오. 사용 가능한 각 속성에 대한 설명은 에 나와 '아마존 S3 퍼블리시' 작업 YAML 있습니다.
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

'아마존 S3 퍼블리시' 작업 YAML

다음은 Amazon S3 게시 작업의 YAML 정의입니다. 이 작업을 사용하는 방법을 알아보려면 을 참조하십시오워크플로를 사용하여 Amazon S3에 파일 게시.

이 작업 정의는 광범위한 워크플로 정의 파일 내에 섹션으로 존재합니다. 이 파일에 대한 자세한 내용은 워크플로우 YAML 정의을 참조합니다.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

```yaml
# The workflow definition starts here.
# See ### ## for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  S3Publish_nn:
    Identifier: aws/s3-publish@v1
    DependsOn:
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      Sources:
        - source-name-1
      Artifacts:
        - artifact-name
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
      SourcePath: my/source
      DestinationBucketName: s3-bucket-name
      TargetPath: my/target
```

S3Publish

(필수)

액션의 이름을 지정합니다. 모든 작업 이름은 워크플로 내에서 고유해야 합니다. 액션 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 액션 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

기본값: `S3Publish_nn`.

해당 UI: 구성 탭/ 작업 이름

Identifier

(*S3Publish*/Identifier)

(필수)

조치를 식별합니다. 버전을 변경하려는 경우가 아니면 이 속성을 변경하지 마십시오. 자세한 내용은 사용할 액션 버전 지정 단원을 참조하십시오.

기본값: `aws/s3-publish@v1`.

해당 UI: 워크플로 다이어그램/ S3Publish _n/ aws/s3-publish @v1 레이블

DependsOn

(*S3Publish*/DependsOn)

(선택 사항)

이 액션을 실행하기 위해 성공적으로 실행되어야 하는 액션, 액션 그룹 또는 게이트를 지정하십시오.

'종속 조건' 기능에 대한 자세한 내용은 을 참조하십시오. 시퀀싱 액션

해당 UI: 입력 탭/ 기준 - 선택 사항

Compute

(*S3Publish*/Compute)

(선택 사항)

워크플로우 작업을 실행하는 데 사용되는 컴퓨팅 엔진. 워크플로 수준 또는 작업 수준에서 컴퓨팅을 지정할 수 있지만 둘 다에서 지정할 수는 없습니다. 워크플로 수준에서 지정된 경우 컴퓨팅 구성은 워크플로에 정의된 모든 작업에 적용됩니다. 워크플로 수준에서는 동일한 인스턴스에서 여러 작업을 실행할 수도 있습니다. 자세한 내용은 작업 간 컴퓨팅 공유 단원을 참조하십시오.

해당 UI: 없음

Type

(*S3 ####*/Compute/Type)

(포함된 Compute 경우 필수)

컴퓨팅 엔진 유형. 다음 값 중 하나를 사용할 수 있습니다.

- EC2(비주얼 에디터) 또는 EC2 (YAML에디터)

  작업 실행 중 유연성을 위해 최적화되었습니다.
- Lambda (비주얼 에디터) 또는 (에디터) Lambda YAML

  작업 시작 속도를 최적화했습니다.

컴퓨팅 유형에 대한 자세한 정보는 컴퓨팅 유형을 참고하세요.

해당 UI: 구성 탭/ 컴퓨팅 유형

Fleet

(*S3Publish*/Compute/Fleet)

(선택 사항)

워크플로 또는 워크플로 작업을 실행할 컴퓨터 또는 플릿을 지정하십시오. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다. 온디맨드 플릿의 예:,. `Linux.x86-64.Large` `Linux.x86-64.XLarge` 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

생략된 경우 기본값은 `Fleet` 입니다. `Linux.x86-64.Large`

해당 UI: 구성 탭/ 컴퓨팅 플릿

Timeout

(*S3Publish*/Timeout)

(필수)

작업이 CodeCatalyst 종료되기 전에 작업을 실행할 수 있는 시간을 분 (편집기) 또는 시간과 분 (비주얼 편집기) 단위로 지정합니다. YAML 최소값은 5분이고 최대값은 에 설명되어 의 워크플로우 할당량 CodeCatalyst 있습니다. 기본 타임아웃은 최대 타임아웃과 동일합니다.

해당 UI: 구성 탭/ 타임아웃 - 선택 사항

Inputs

(*S3Publish*/Inputs)

(선택 사항)

이 `Inputs` 섹션에서는 워크플로우 실행 중에 `S3Publish` 필요한 데이터를 정의합니다.

> **Note**
>
> 각 AWS CDK 배포 작업에는 최대 4개의 입력 (소스 1개와 아티팩트 3개) 이 허용됩니다. 변수는 이 총계에 포함되지 않습니다.

다른 입력 (예: 소스 및 아티팩트) 에 있는 파일을 참조해야 하는 경우 소스 입력은 기본 입력이고 아티팩트는 보조 입력입니다. 보조 입력의 파일에 대한 참조에는 기본 입력과 구분하기 위해 특수 접두사가 사용됩니다. 세부 정보는 예: 여러 아티팩트의 파일 참조을 참조하세요.

해당 UI: 입력 탭

Sources

(*S3 ####*/Inputs/Sources)

(Amazon S3에 게시하려는 파일이 원본 리포지토리에 저장되어 있는 경우 필수)

Amazon S3에 게시하려는 파일이 원본 리포지토리에 저장되어 있는 경우 해당 원본 리포지토리의 레이블을 지정하십시오. 현재 지원되는 유일한 레이블은 입니다`WorkflowSource`.

Amazon S3에 게시하려는 파일이 원본 리포지토리에 포함되어 있지 않은 경우 해당 파일은 다른 작업에서 생성된 아티팩트에 있어야 합니다.

소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

해당 UI: 입력 탭/ 소스 - 선택 사항

Artifacts - input

(*S3Publish*/Inputs/Artifacts)

(Amazon S3에 게시하려는 파일이 이전 작업의 출력 아티팩트에 저장되어 있는 경우 필수)

Amazon S3에 게시하려는 파일이 이전 작업에서 생성된 아티팩트에 포함되어 있는 경우 여기에 해당 아티팩트를 지정하십시오. 파일이 아티팩트에 포함되어 있지 않은 경우 해당 파일은 소스 리포지토리에 있어야 합니다.

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 구성 탭/ 아티팩트 - 선택 사항

Variables - input

(*S3Publish*/Inputs/Variables)

(선택 사항)

작업에 사용할 수 있도록 하려는 입력 변수를 정의하는 이름/값 쌍의 시퀀스를 지정하십시오. 변수 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 변수 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

예제를 비롯한 변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

해당 UI: 입력 탭/ 변수 - 선택 사항

Environment

(*S3Publish*/Environment)

(필수)

작업에 사용할 CodeCatalyst 환경을 지정하십시오. 작업은 선택한 환경에 VPC 지정된 AWS 계정 및 선택적 Amazon에 연결됩니다. 작업은 환경에 지정된 기본 IAM 역할을 사용하여 Amazon에 연결하고 Amazon VPC 연결에 지정된 IAM 역할을 사용하여 Amazon에 연결합니다VPC. AWS 계정

> **Note**
>
> 기본 IAM 역할에 작업에 필요한 권한이 없는 경우 다른 역할을 사용하도록 작업을 구성할 수 있습니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.

환경에 대한 자세한 내용은 및 에 AWS 계정 배포 VPCs 및 을 참조하십시오환경 생성.

해당 UI: 구성 탭/ 환경

Name

(*S3 ####*/Environment/Name)

(포함된 Environment 경우 필수)

작업에 연결할 기존 환경의 이름을 지정합니다.

해당 UI: 구성 탭/ 환경

Connections

(*S3 ####*/Environment/Connections)

(액션의 최신 버전에서는 선택 사항, 이전 버전에서는 필수)

작업에 연결할 계정 연결을 지정합니다. 에서 최대 1개의 계정 연결을 지정할 수 `Environment` 있습니다.

계정 연결을 지정하지 않은 경우:

- 작업은 CodeCatalyst 콘솔 환경에 지정된 AWS 계정 연결 및 기본 IAM 역할을 사용합니다. 환경에 계정 연결 및 기본 IAM 역할을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.
- 기본 IAM 역할에는 작업에 필요한 정책 및 권한이 포함되어야 합니다. 이러한 정책 및 권한이 무엇인지 확인하려면 작업 YAML 정의 설명서에서 역할 속성에 대한 설명을 참조하십시오.

계정 연결에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정. 환경에 계정 연결을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

해당 UI: 액션 버전에 따라 다음 중 하나:

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Name

(*S3 ####*/Environment/Connections/Name)

(포함된 Connections 경우 필수)

계정 연결 이름을 지정합니다.

해당 UI: 액션 버전에 따라 다음 중 하나:

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Role

(*S3 ####*/Environment/Connections/Role)

(포함된 Connections 경우 필수)

Amazon S3 게시 작업에서 Amazon S3에 AWS 액세스하고 파일을 복사하는 데 사용하는 IAM 역할의 이름을 지정합니다. CodeCatalyst 스페이스에 역할을 추가했는지, 역할에 다음 정책이 포함되어 있는지 확인하십시오.

IAM역할을 지정하지 않으면 CodeCatalyst 콘솔 환경에 나열된 기본 IAM 역할이 작업에 사용됩니다. 환경에서 기본 역할을 사용하는 경우 해당 역할에 다음 정책이 적용되는지 확인하십시오.

- 다음 권한 정책:

> ⚠ Warning
>
> 권한을 다음 정책에 표시된 권한으로 제한하십시오. 더 광범위한 권한이 있는 역할을 사용하면 보안 위험이 발생할 수 있습니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

- 다음과 같은 사용자 지정 신뢰 정책:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

> **Note**
>
> 원하는 경우 이 작업에 CodeCatalystWorkflowDevelopmentRole-*spaceName* 역할을 사용할 수 있습니다. 이에 대한 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기 섹션을 참조하세요.

`CodeCatalystWorkflowDevelopmentRole-`*`spaceName`*역할에는 보안 위험을 초래할 수 있는 전체 액세스 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다.

해당 UI: 액션 버전에 따라 다음 중 하나가 표시됩니다.

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 역할

Configuration

(*S3 ####*/Configuration)

(필수)

액션의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭

SourcePath

(*S3 ####*/Configuration/SourcePath)

(필수)

Amazon S3에 게시하려는 디렉터리 또는 파일의 이름과 경로를 지정합니다. 디렉터리 또는 파일은 소스 리포지토리 또는 이전 작업의 아티팩트에 있을 수 있으며, 소스 리포지토리나 아티팩트 루트를 기준으로 합니다.

예시:

지정하면 Amazon `/myFolder` S3에 의 콘텐츠가 `./myFolder/` 복사되고 기본 디렉터리 구조가 보존됩니다.

Amazon S3에만 *`./myFolder/myfile.txt`* `myfile.txt` 복사본을 지정합니다. (디렉터리 구조는 제거되었습니다.)

와일드카드는 사용할 수 없습니다.

> **Note**
>
> 디렉터리나 파일 경로에 해당 파일을 찾을 아티팩트 또는 소스를 나타내는 접두사를 추가해야 할 수도 있습니다. 자세한 내용은 소스 리포지토리 파일 참조 및 아티팩트의 파일 참조 단원을 참조하세요.

해당 UI: 구성 탭/ 소스 경로

DestinationBucketName

(*S3Publish*/Configuration/DestinationBucketName)

(필수)

파일을 게시하려는 Amazon S3 버킷의 이름을 지정합니다.

해당 UI: 구성 탭/ 대상 버킷 - 선택 사항

TargetPath

(*S3Publish*/Configuration/TargetPath)

(선택 사항)

Amazon S3에서 파일을 게시하려는 디렉터리의 이름과 경로를 지정합니다. 디렉터리가 없는 경우 디렉터리가 생성됩니다. 디렉터리 경로에는 버킷 이름이 포함되어서는 안 됩니다.

예시:

`myS3Folder`

`./myS3Folder/myS3Subfolder`

해당 UI: 구성 탭/ 대상 디렉터리 - 선택 사항

및 에 AWS 계정 배포 VPCs

CodeCatalyst 워크플로를 사용하면 AWS 클라우드의 AWS 계정%s 및 VPCs Amazon을 대상으로 애플리케이션 및 기타 리소스를 배포할 수 있습니다. 이러한 배포를 활성화하려면 환경을 CodeCatalyst 설정해야 합니다.

개발 CodeCatalyst 환경과 혼동하지 말아야 할 환경은 CodeCatalyst 워크플로가 연결되는 대상 AWS 계정 및 선택적 VPC Amazon을 정의합니다. 또한 환경은 워크플로가 대상 계정 내의 AWS 서비스와 리소스에 액세스하는 데 필요한 IAM역할을 정의합니다.

여러 환경을 설정하고 개발, 테스트, 스테이징, 프로덕션 등의 이름을 지정할 수 있습니다. 이러한 환경에 배포하는 경우 배포 관련 정보가 환경의 CodeCatalyst 배포 활동 및 배포 대상 탭에 표시됩니다.

환경을 시작하려면 어떻게 해야 하나요?

CodeCatalyst 환경을 추가하고 사용하는 상위 단계는 다음과 같습니다.

1. CodeCatalyst 스페이스에서 하나 이상의 AWS 계정을 연결합니다. 이 과정에서 워크플로가 내 리소스에 액세스하는 데 필요한 IAM 역할을 추가하세요 AWS 계정. 자세한 내용은 연결된 AWS 리소스에 대한 액세스 허용 AWS 계정 단원을 참조하십시오.
2. CodeCatalyst 프로젝트에서 1단계의 IAM 역할 및 역할 중 AWS 계정하나를 포함하는 환경을 만드세요. 자세한 내용은 환경 생성 단원을 참조하십시오.
3. CodeCatalyst 프로젝트의 워크플로우에 2단계에서 만든 환경을 가리키는 액션을 추가합니다. 자세한 내용은 워크플로에 작업 추가 단원을 참조하십시오.

   이제 환경을 구성했습니다. 이제 작업을 통해 환경의 AWS 계정 지정된 위치에 리소스를 배포할 수 있습니다.

> **Note**
>
> 환경에 VPC Amazon을 추가할 수도 있습니다. 자세한 내용은 CodeCatalyst 관리 가이드 및 의 스페이스에 대한 VPC 연결 추가를 참조하십시오a를 VPC 환경과 연결하기.

단일 워크플로 내에 여러 환경이 존재할 수 있습니까?

예. 워크플로에 여러 작업이 포함된 경우 각 작업에 환경을 할당할 수 있습니다. 예를 들어, 한 번에는 환경이 할당되고 다른 하나에는 `my-staging-enviroment` 환경이 할당되는 두 개의 배포 작업이 포함된 워크플로가 있을 수 `my-production-environment` 있습니다.

환경을 지원하는 워크플로 작업은 무엇입니까?

리소스를 AWS 클라우드에 배포하거나 다른 이유 (예: 모니터링 및 보고) 로 AWS 서비스와 통신하는 모든 워크플로 작업은 환경을 지원합니다.

배포 정보를 표시할 수 있도록 지원하는 작업은 무엇입니까? CodeCatalyst

환경을 지원하는 워크플로 작업 중에서 CodeCatalyst 콘솔의 배포 활동 및 배포 대상 페이지에 배포 정보가 표시되도록 지원하는 작업은 소수에 불과합니다.

다음 워크플로 작업은 해당 배포 정보를 표시하는 것을 지원합니다.

- AWS CloudFormation 스택 배포 — 자세한 내용은 을 참조하십시오. 스택 배포 AWS CloudFormation
- Amazon에 배포 ECS — 자세한 내용은 을 참조하십시오. 워크플로를 ECS 사용하여 Amazon에 배포하기
- Kubernetes 클러스터에 배포 — 자세한 내용은 을 참조하십시오. 워크플로를 EKS 사용하여 Amazon에 배포하기
- AWS CDK 배포 - 자세한 내용은 을 참조하십시오. 워크플로를 통한 AWS CDK 앱 배포

지원되는 리전

환경 페이지에는 모든 AWS 지역의 리소스가 표시될 수 있습니다.

환경은 필수인가요?

환경이 할당된 워크플로 작업에서 리소스를 AWS 클라우드에 배포하거나 다른 이유 (예: 모니터링 및 보고) 로 AWS 서비스와 통신하는 경우 환경은 필수입니다.

예를 들어 애플리케이션을 빌드하는 빌드 작업이 있지만 사용자 AWS 계정 또는 VPC Amazon과 통신할 필요가 없는 경우 작업에 환경을 할당할 필요가 없습니다. 하지만 빌드 작업에서 사용자의 AWS 계정 Amazon CloudWatch 서비스로 로그를 전송하는 경우에는 작업에 환경이 할당되어 있어야 합니다.

주제

- 환경 생성
- 환경을 액션과 연결하기
- a를 VPC 환경과 연결하기
- 환경과 연결하기 AWS 계정
- 액션의 IAM 역할 변경하기

환경 생성

다음 지침을 사용하여 나중에 워크플로 작업에 연결할 수 있는 환경을 만드십시오.

시작하기 전 준비 사항

다음이 필요합니다.

- CodeCatalyst 공백. 자세한 내용은 설정 및 로그인 CodeCatalyst 단원을 참조하십시오.
- CodeCatalyst 프로젝트. 자세한 내용은 블루프린트로 프로젝트 생성 단원을 참조하십시오.
- 워크플로 작업에서 액세스해야 하는 IAM 역할이 포함된 AWS 계정 연결 AWS. 계정 연결 생성에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정. 환경당 최대 1개의 계정 연결을 사용할 수 있습니다.

  > **Note**
  >
  > 계정 연결 없이 환경을 만들 수 있지만 나중에 다시 돌아와서 연결을 추가해야 합니다.

- 다음 CodeCatalyst 역할 중 하나:
  - 스페이스 관리자
  - 프로젝트 관리자
  - 기고자

  > **Note**
  >
  > 기여자 역할이 있는 경우 환경을 만들 수는 있지만 AWS 계정 연결에 연결할 수는 없습니다. 스페이스 관리자 또는 프로젝트 관리자 역할을 가진 사람에게 환경을 연결에 AWS 계정 연결하도록 요청해야 합니다.

  권한 및 역할에 대한 자세한 내용은 을 참조하십시오사용자에게 프로젝트 권한 부여.

환경을 생성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 환경을 선택합니다.
4. 환경 이름에 이름을 입력합니다 (예**`Production`**: 또는). **`Staging`**

5. 환경 유형에서 다음 중 하나를 선택합니다.

   - 비프로덕션 — 애플리케이션을 프로덕션으로 전환하기 전에 애플리케이션이 의도한 대로 작동하는지 테스트할 수 있는 환경입니다.
   - 프로덕션 — 공개적으로 사용할 수 있고 최종 애플리케이션을 호스팅하는 '라이브' 환경입니다.

     프로덕션을 선택하면 UI에서 환경과 관련된 모든 작업 옆에 프로덕션 배지가 나타납니다. 배지를 통해 어떤 액션이 프로덕션에 배포되고 있는지 빠르게 확인할 수 있습니다. 배지가 표시되는 것 외에는 프로덕션 환경과 비프로덕션 환경 간에 차이가 없습니다.

6. (선택 사항) 설명에 다음과 같은 설명을 입력합니다. **`Production environment for the hello-world app`**

7. AWS 계정 연결 - 선택 사항에서 이 환경에 연결할 AWS 계정 연결을 선택합니다. 이 환경에 할당된 워크플로 작업은 관련 환경에 연결할 수 AWS 계정있습니다. 에서 AWS 계정 연결을 만드는 방법에 대한 자세한 내용은 CodeCatalyst 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정.

   사용하려는 AWS 계정 연결이 목록에 없다면 프로젝트에서 허용되지 않기 때문일 수 있습니다. 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 프로젝트 제한 계정 연결 구성을 참조하십시오.

8. 기본 IAM 역할에서 이 IAM 환경에 연결할 역할을 선택합니다. 이 환경에 할당된 워크플로 작업은 이 역할을 상속하며, 이 IAM 역할을 사용하여 내 서비스 및 리소스에 연결할 수 있습니다. AWS 계정

   환경을 여러 작업에 할당해야 하고 이러한 작업에 여기에 지정된 기본 역할과 다른 IAM 역할이 필요한 경우 역할 전환 옵션을 사용하여 각 작업의 구성 탭에서 다른 역할을 지정할 수 있습니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.

   기본값으로 사용하려는 IAM 역할이 목록에 없는 경우 AWS 계정 연결에 아직 추가하지 않았기 때문일 수 있습니다. 계정 연결에 IAM 역할을 추가하려면 을 참조하십시오계정 연결에 IAM 역할 추가.

9. (선택 사항) 연결에서 VPC 이 환경에 연결할 VPC 연결을 선택합니다. VPC연결 생성에 대한 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 Amazon 가상 사설 클라우드 관리를 참조하십시오.

   사용하려는 VPC 연결이 목록에 없는 경우 프로젝트에서 허용되지 않는 AWS 계정 연결이 포함되어 있기 때문일 수 있습니다. 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 프로젝트 제한 계정 연결 구성을 참조하십시오.

10. 환경 생성을 선택합니다. CodeCatalyst 빈 환경을 만듭니다.

다음 단계

- 이제 환경을 만들었으니 워크플로 작업에 연결할 준비가 되었습니다. 자세한 내용은 환경을 액션과 연결하기 단원을 참조하십시오.

환경을 액션과 연결하기

환경을 지원되는 워크플로 작업과 연결하면 환경의 기본 IAM 역할 및 선택적 Amazon이 작업에 VPC 할당됩니다. AWS 계정그러면 작업을 해당 IAM 역할을 AWS 계정 사용하여 연결하고 배포할 수 있으며 선택적 Amazon에도 연결할 수 VPC 있습니다.

다음 지침을 사용하여 환경을 작업과 연결하십시오.

1단계: 환경을 워크플로 작업에 연결

다음 절차를 사용하여 환경을 워크플로 작업에 연결합니다.

Visual

비주얼 편집기를 사용하여 환경을 워크플로 작업에 연결하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로 다이어그램에서 환경에서 지원되는 작업을 선택합니다. 자세한 내용은 배포 정보를 표시할 수 있도록 지원하는 작업은 무엇입니까? CodeCatalyst 단원을 참조하십시오.
8. 구성 탭을 선택하고 다음과 같이 환경 필드에 정보를 지정합니다.

   환경

   작업에 사용할 CodeCatalyst 환경을 지정합니다. 작업은 선택한 환경에 VPC 지정된 AWS 계정 및 선택적 Amazon에 연결됩니다. 작업은 환경에 지정된 기본 IAM 역할을 사용하여

Amazon에 연결하고 Amazon VPC 연결에 지정된 IAM 역할을 사용하여 Amazon에 연결합니다VPC. AWS 계정

> **Note**
>
> 기본 IAM 역할에 작업에 필요한 권한이 없는 경우 다른 역할을 사용하도록 작업을 구성할 수 있습니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.

환경에 대한 자세한 내용은 및 에 AWS 계정 배포 VPCs 및 을 참조하십시오환경 생성.

9. (선택 사항) 작업과 관련된 IAM 역할을 변경합니다. 작업에 대한 잘못된 권한 집합이 포함되어 있는 경우 역할을 변경하는 것이 좋습니다.

   역할을 변경하려면:

   1. '내 내용' 보기 ***my-environment*** ? 상자를 클릭하고 세로 줄임표 아이콘 ()

      ⋮

      을 선택합니다.
   2. 다음 중 하나를 선택합니다.
      - 역할 전환. 이 작업에 사용되는 IAM 역할을 변경하려면 이 옵션을 선택합니다. 이 작업만 변경할 수 있습니다. 다른 작업은 관련 환경에 지정된 기본 IAM 역할을 계속 사용합니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.
      - 환경 편집. 환경에 나열된 기본 IAM 역할을 변경하려면 이 옵션을 선택합니다. 이 옵션을 선택하면 사용자 작업 및 동일한 환경과 관련된 다른 모든 작업이 새 기본 역할을 사용하기 시작합니다. IAM

      > **Important**
      >
      > 기본 역할을 업데이트할 때는 주의해야 합니다. IAM 역할 권한이 환경을 공유하는 모든 작업에 충분하지 않은 경우 역할을 변경하면 작업이 실패할 수 있습니다.

10. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
11. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

YAML편집기를 사용하여 환경을 워크플로 작업에 연결하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 환경에 연결하려는 워크플로 작업에 다음과 비슷한 코드를 추가합니다.

```
action-name:
  Environment:
    Name: environment-name
```

   자세한 내용은 작업 유형 항목을 참조하십시오. 이 항목에는 YAML 참조를 포함하여 각 작업에 대한 설명서로 연결되는 링크가 있습니다.
8. (선택 사항) 환경에 나열된 기본 IAM 역할과 다른 역할을 작업에 사용하려면 사용하려는 역할이 포함된 `Connections:` 섹션을 추가하십시오. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.
9. (선택 사항) 커밋하기 전에 워크플로 YAML 코드의 유효성을 검사하려면 [Validate] 를 선택합니다.
10. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

2단계: 배포 활동 페이지를 채웁니다.

환경을 워크플로 작업에 연결한 후 CodeCatalyst 콘솔의 환경 섹션에 있는 배포 활동 및 배포 대상 페이지를 배포 정보로 채울 수 있습니다. 다음 지침에 따라 이러한 페이지를 채우십시오.

> **Note**
>
> 일부 작업만 CodeCatalyst 콘솔에 배포 정보를 표시하는 것을 지원합니다. 자세한 내용은 배포 정보를 표시할 수 있도록 지원하는 작업은 무엇입니까? CodeCatalyst 단원을 참조하십시오.

에 배포 정보를 추가하려면 CodeCatalyst

1. 에서 1단계: 환경을 워크플로 작업에 연결 변경 내용을 커밋할 때 워크플로 실행이 자동으로 시작되지 않는 경우 다음과 같이 수동으로 실행을 시작하십시오.

   a. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.

   b. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.

   c. Run(실행)을 선택합니다.

   워크플로 실행 시 새 배포가 시작되며, 이로 CodeCatalyst 인해 배포 정보가 추가됩니다 CodeCatalyst.

2. CodeCatalyst 콘솔에 배포 활동이 추가되었는지 확인합니다.

   a. 탐색 창에서 CI/CD를 선택한 다음 환경을 선택합니다.

   b. 환경을 선택합니다 (예:). `Production`

   c. 배포 활동 탭을 선택하고 배포가 상태로 나타나는지 확인합니다 SUCCEEDED. 이는 워크플로 실행이 애플리케이션 리소스를 성공적으로 배포했음을 나타냅니다.

   d. 배포 대상 탭을 선택하고 애플리케이션 리소스가 나타나는지 확인합니다.

a를 VPC 환경과 연결하기

VPC연결이 있는 환경에서 작업을 구성하면 해당 작업은 네트워크 규칙 및 관련 환경에서 지정한 액세스 리소스에 따라 에 연결되어 실행됩니다. VPC VPC 하나 이상의 환경에서 동일한 VPC 연결을 사용할 수 있습니다.

다음 지침을 사용하여 VPC 연결을 환경에 연결하십시오.

VPC연결을 환경에 연결하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 환경을 선택합니다.
4. 환경을 선택합니다 (예:). `Production`
5. 환경 속성 탭을 선택합니다.

6. VPC연결 관리를 선택하고 원하는 VPC 연결을 선택한 다음 확인을 선택합니다. 그러면 선택한 VPC 연결이 이 환경과 연결됩니다.

> **Note**
>
> 사용하려는 VPC 연결이 목록에 없는 경우 프로젝트에 허용되지 않는 AWS 계정 연결이 포함되어 있기 때문일 수 있습니다. 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 프로젝트 제한 계정 연결 구성을 참조하십시오.

자세한 내용은 CodeCatalyst 관리자 안내서의 Amazon 가상 사설 클라우드 관리를 참조하십시오.

환경과 연결하기 AWS 계정

다음 지침에 따라 a를 AWS 계정 환경에 연결하십시오. 환경에 연결하면 환경에 할당된 워크플로 작업을 환경에 연결할 수 AWS 계정있습니다. AWS 계정

계정 연결에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정.

시작하기 전 준비 사항

다음이 필요합니다.

- 워크플로 작업에서 액세스해야 하는 IAM 역할이 포함된 AWS 계정 연결 AWS. 계정 연결 생성에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정. 환경당 최대 1개의 계정 연결을 사용할 수 있습니다.
- 다음 CodeCatalyst 역할 중 하나: 스페이스 관리자 또는 프로젝트 관리자 자세한 내용은 사용자에게 프로젝트 권한 부여 단원을 참조하십시오.

환경에 AWS 계정 연결하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 환경을 선택합니다.
4. 환경을 선택합니다 (예:). `Production`
5. 환경 편집을 선택합니다.
6. 환경 속성의AWS 계정 연결 - 옵션 드롭다운 목록에서 원하는 항목을 선택합니다 AWS 계정.

사용하려는 AWS 계정 연결이 목록에 없다면 프로젝트에서 허용되지 않기 때문일 수 있습니다. 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 프로젝트 제한 계정 연결 구성을 참조하십시오.

7. 기본 IAM 역할에서 이 IAM 환경에 연결할 역할을 선택합니다. 이 환경에 할당된 워크플로 작업은 이 역할을 상속하며, 이 IAM 역할을 사용하여 내 서비스 및 리소스에 연결할 수 있습니다. AWS 계정

   기본값으로 사용하려는 IAM 역할이 목록에 없다면 아직 AWS 계정 연결에 추가하지 않았기 때문일 수 있습니다. 계정 연결에 IAM 역할을 추가하려면 을 참조하십시오계정 연결에 IAM 역할 추가.

액션의 IAM 역할 변경하기

기본적으로 환경을 워크플로 작업에 연결하면 해당 작업은 환경에 지정된 기본 IAM 역할을 상속합니다. 작업에서 다른 역할을 사용하도록 이 동작을 변경할 수 있습니다. AWS 클라우드에서 작업을 운영하는 데 필요한 권한이 기본 IAM 역할에 없는 경우 작업에 다른 역할을 사용하는 것이 필요할 수 있습니다.

작업에 다른 IAM 역할을 할당하려면 시각적 편집기의 역할 전환 옵션이나 편집기의 `Connections:` 속성을 사용할 수 있습니다. YAML 새 역할은 환경에 지정된 기본 IAM 역할을 재정의하므로 기본 IAM 역할을 그대로 유지할 수 있습니다. 기본 역할을 사용하는 다른 작업이 있는 경우 기본 IAM 역할을 그대로 유지하는 것이 좋습니다.

다음 지침에 따라 해당 환경에서 지정된 역할과 다른 IAM 역할을 사용하도록 작업을 구성하십시오.

Visual

액션에 다른 IAM 역할 할당하기 (비주얼 에디터)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. IAM역할을 업데이트하려는 작업을 나타내는 상자를 선택합니다.
7. 구성 탭을 선택합니다.

8. What's in in (내용) 에서 ***my-environment*** ? 상자에서 세로 줄임표 아이콘 ()

   을 선택합니다.

9. 역할 전환을 선택합니다.

10. 역할 전환 대화 상자의 IAM역할 드롭다운 목록에서 작업에 사용할 IAM 역할을 선택합니다. 이 역할은 환경의 기본 IAM 역할보다 우선합니다. 사용하려는 역할이 목록에 없는 경우 스페이스에 추가했는지 확인하세요. 자세한 내용은 계정 연결에 IAM 역할 추가 단원을 참조하십시오.

    이제 선택한 역할이 What's in in (위치) 에 나타납니다.***my-environment***? 상자에 워크플로에 정의된 배지가 함께 들어 있습니다. 역할은 `Connections:` 섹션의 워크플로 정의 파일에도 표시됩니다.

11. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로의 YAML 코드를 검증합니다.

12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

작업에 다른 IAM 역할을 할당하려면 (YAML편집자)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 다른 IAM 역할을 사용하려는 워크플로 작업에서 다음과 비슷한 `Connections:` 섹션을 추가합니다.

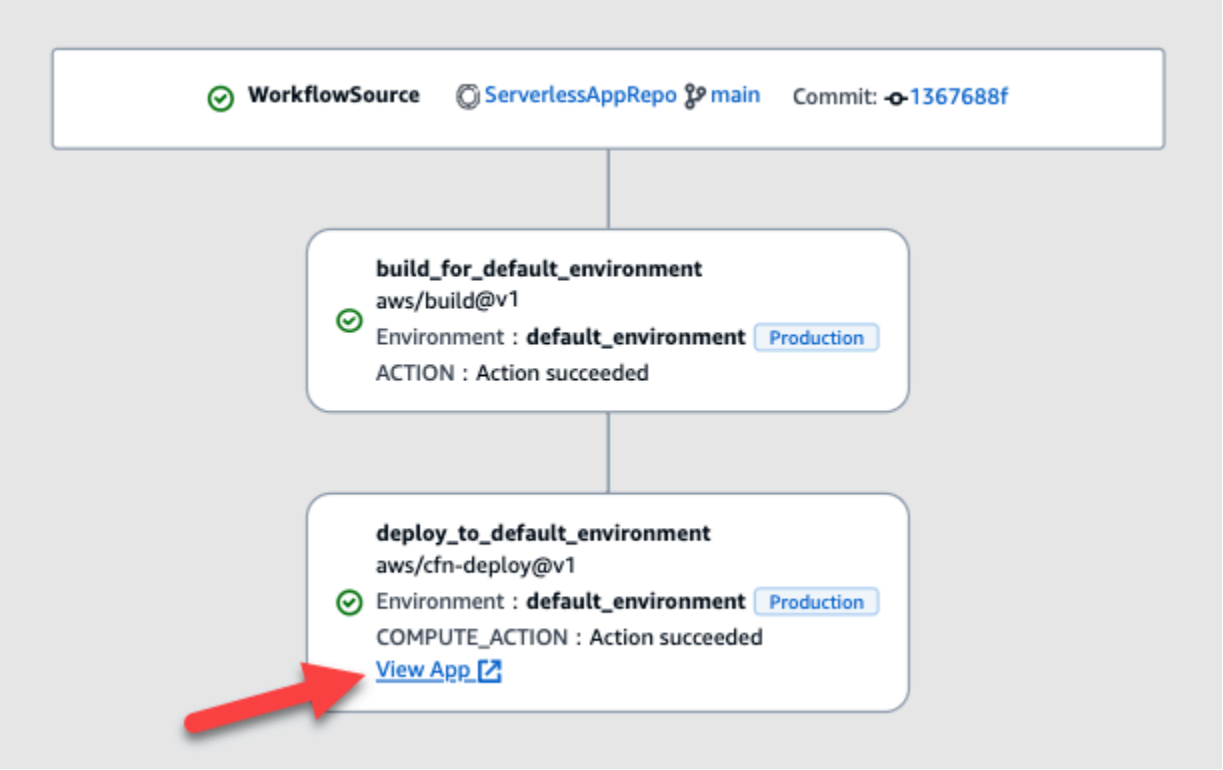
```
action-name:
  Environment:
    Name: environment-name
    Connections:
      - Name: account-connection-name
        Role: iam-role-name
```

위 코드에서 다음을 대체하십시오.*account-connection-name* IAM역할이 포함된 계정 연결 이름으로 바꾸고 *iam-role-name* 작업에 사용할 IAM 역할의 이름을 입력합니다. 이 역할은 환경의 기본 IAM 역할을 재정의합니다. 스페이스에 역할을 추가했는지 확인하세요. 자세한 내용은 계정 연결에 IAM 역할 추가 단원을 참조하십시오.

자세한 내용은 작업 유형 주제를 참조하십시오. 이 항목에는 YAML 참조를 포함하여 각 작업에 대한 설명서로 연결되는 링크가 있습니다.

워크플로 URL 다이어그램에 앱 표시

워크플로에서 애플리케이션을 배포하는 경우 애플리케이션을 클릭 URL 가능한 링크로 CodeCatalyst 표시하도록 Amazon을 구성할 수 있습니다. 이 링크는 CodeCatalyst 콘솔의 배포한 작업 내에 표시됩니다. 다음 워크플로 다이어그램은 작업 하단에 URL 나타나는 View 앱을 보여줍니다.



CodeCatalyst 콘솔에서 URL 클릭할 수 있게 하면 애플리케이션 배포를 빠르게 확인할 수 있습니다.

> **Note**
>
> URL이 앱은 Amazon에 배포 ECS 작업과 함께 지원되지 않습니다.

이 기능을 활성화하려면 또는 를 포함하는 appurl 이름을 가진 출력 변수를 작업에 추가하십시오endpointurl. 이름은 대시 (-), 밑줄 () 또는 공백 ( ) 을 포함하거나 포함하지 _ 않고 사용할 수 있습니다. 문자열은 대소문자를 구분하지 않습니다. 변수 값을 배포된 https URL 애플리케이션의 http or로 설정합니다.

> **Note**
>
> 또는 endpoint url 문자열을 포함하도록 기존 출력 변수를 업데이트하는 경우 새 변수 이름을 사용하도록 이 변수에 대한 모든 참조를 업데이트하십시오. app url

자세한 단계는 다음 절차 중 하나를 참조하십시오.

- "AWS CDK 배포" 작업에 앱을 URL 표시하려면
- AWS CloudFormation '스택 배포' 작업에 앱을 URL 표시하려면
- 다른 모든 작업에 앱을 URL 표시하려면

구성을 마쳤으면 다음 지침에 따라 예상대로 나타나는지 확인하십시오. URL

- 애플리케이션이 URL 추가되었는지 확인하려면

"AWS CDK 배포" 작업에 앱을 URL 표시하려면

1. AWS CDK 배포 작업을 사용하는 경우 AWS CDK 애플리케이션 코드에 CfnOutput 구문 (키-값 쌍) 을 추가하세요.

   - 키 이름은 대시 (-)endpointurl, 밑줄 () 또는 공백 () 을 포함하거나 포함하지 appurl 않아야 합니다. _ 문자열은 대소문자를 구분하지 않습니다.
   - 값은 배포된 https URL 애플리케이션의 http or이어야 합니다.

   예를 들어 AWS CDK 코드는 다음과 같을 수 있습니다.

```
import { Duration, Stack, StackProps, CfnOutput, RemovalPolicy} from 'aws-cdk-lib';
import * as dynamodb from 'aws-cdk-lib/aws-dynamodb';
import * as s3 from 'aws-cdk-lib/aws-s3';
import { Construct } from 'constructs';
import * as cdk from 'aws-cdk-lib';
export class HelloCdkStack extends Stack {
```

```
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);
    const bucket = new s3.Bucket(this, 'my-bucket', {
      removalPolicy: RemovalPolicy.DESTROY,
    });
    new CfnOutput(this, 'APP-URL', {
      value: https://mycompany.myapp.com,
      description: 'The URL of the deployed application',
      exportName: 'myApp',
    });
    ...
  }
}
```

`CfnOutput`구문에 대한 자세한 내용은 CfnOutputProps AWS Cloud Development Kit (AWS CDK) API참조의 인터페이스를 참조하십시오.

2. 코드를 저장하고 커밋하세요.
3. 애플리케이션이 URL 추가되었는지 확인하려면로 이동합니다.

AWS CloudFormation '스택 배포' 작업에 앱을 URL 표시하려면

1. Deploy AWS CloudFormation stack 작업을 사용하는 경우 CloudFormation 템플릿 또는 AWS SAM 템플릿의 `Outputs` 섹션에 다음과 같은 특징을 가진 출력을 추가하세요.

   - 키 (논리적 ID라고도 함) 는 결합 대시 ()`endpointurl`, 밑줄 () 또는 공백 (-) 을 포함하거나 포함하지 `appurl` 않아야 합니다. _ 문자열은 대소문자를 구분하지 않습니다.
   - 값은 배포된 `https` URL 애플리케이션의 `http` or이어야 합니다.

   예를 들어 CloudFormation 템플릿은 다음과 같을 수 있습니다.

```
"Outputs" : {
  "APP-URL" : {
    "Description" : "The URL of the deployed app",
    "Value" : "https://mycompany.myapp.com",
    "Export" : {
      "Name" : "My App"
    }
  }
}
```

CloudFormation 출력에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 출력을 참조하십시오.

2. 코드를 저장하고 커밋하세요.
3. 애플리케이션이 URL 추가되었는지 확인하려면로 이동합니다.

다른 모든 작업에 앱을 URL 표시하려면

빌드 작업 또는 작업과 같은 다른 GitHub 작업을 사용하여 애플리케이션을 배포하는 경우 다음을 수행하여 앱이 URL 표시되도록 하세요.

1. 워크플로 정의 파일에 있는 작업의 Inputs 또는 Steps 섹션에서 환경 변수를 정의합니다. 변수는 다음과 같은 특성을 가져야 합니다.

   - 는 대시 (-)endpointurl, 밑줄 () 또는 공백 ( ) 을 포함하거나 포함하지 name appurl 않아야 합니다. _ 문자열은 대소문자를 구분하지 않습니다.
   - 값은 배포된 https URL 애플리케이션의 http or이어야 합니다.

   예를 들어 빌드 작업은 다음과 같을 수 있습니다.

   ```
   Build-action:
     Identifier: aws/build@v1
     Inputs:
       Variables:
         - Name: APP-URL
           Value: https://mycompany.myapp.com
   ```

   ... 또는 다음과 같습니다.

   ```
   Actions:
     Build:
       Identifier: aws/build@v1
       Configuration:
         Steps:
           - Run: APP-URL=https://mycompany.myapp.com
   ```

   환경 변수 정의에 대한 자세한 내용은 을 참조하십시오변수 정의하기.

2. 변수 내보내기.

예를 들어 빌드 액션은 다음과 같을 수 있습니다.

```
Build-action:
  ...
  Outputs:
    Variables:
      - APP-URL
```

변수 내보내기에 대한 자세한 내용은 을 참조하십시오다른 액션에서 사용할 수 있도록 변수 내보내기.

3. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로의 YAML 코드를 검증하십시오.
4. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.
5. 애플리케이션이 URL 추가되었는지 확인하려면로 이동합니다.

애플리케이션이 URL 추가되었는지 확인하려면

- 워크플로 실행을 시작합니다 (자동으로 시작되지 않은 경우). 새로 실행할 때는 앱이 워크플로 다이어그램에 클릭 가능한 링크로 URL 표시되어야 합니다. 실행 시작에 대한 자세한 내용은 을 참조하십시오. 워크플로우 수동 실행 시작

배포 대상 제거

CodeCatalyst 콘솔의 배포 대상 페이지에서 Amazon ECS 클러스터 또는 AWS CloudFormation 스택과 같은 배포 대상을 제거할 수 있습니다.

> **Important**
>
> 배포 대상을 제거하면 CodeCatalyst 콘솔에서는 제거되지만 해당 대상을 호스팅하는 AWS 서비스에서 계속 사용할 수 있습니다 (아직 존재하는 경우).

배포 대상이 유효하지 않은 경우 배포 대상을 제거하는 것을 고려해 보십시오. CodeCatalyst 다음과 같은 경우 대상이 유효하지 않을 수 있습니다.

- 대상에 배포한 워크플로를 삭제했습니다.
- 배포하려는 스택 또는 클러스터를 변경했습니다.

- AWS 콘솔의 또는 Amazon ECS 서비스에서 스택 CloudFormation 또는 클러스터를 삭제했습니다.

배포 대상을 제거하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 환경을 선택합니다.
4. 제거하려는 배포 대상이 포함된 환경 이름을 선택합니다. 환경에 대한 자세한 내용은 을 참조하십시오및 에 AWS 계정 배포 VPCs.
5. 배포 대상 탭을 선택합니다.
6. 제거하려는 배포 대상 옆에 있는 라디오 버튼을 선택합니다.
7. 제거를 선택합니다.

   대상이 페이지에서 제거됩니다.

커밋별 배포 상태 추적

개발 수명 주기 중 언제든지 버그 수정, 새 기능 또는 기타 영향을 미치는 변경 사항과 같은 특정 커밋의 배포 상태를 파악하는 것이 중요합니다. 개발 팀에 배포 상태 추적 기능이 도움이 되는 다음과 같은 시나리오를 생각해 보세요.

- 개발자로서 버그를 해결하기 위해 수정했으니 팀의 배포 환경 전반에서 해당 버그의 배포 상태를 보고하고 싶을 것입니다.
- 릴리스 관리자는 배포된 커밋 목록을 보고 배포 상태를 추적하고 보고해야 합니다.

CodeCatalyst 개별 커밋 또는 변경 사항이 어디에 어떤 환경에 배포되었는지 한 눈에 확인할 수 있는 보기를 제공합니다. 이 뷰에는 다음이 포함됩니다.

- 커밋 목록.
- 커밋이 포함된 배포 상태.
- 커밋이 성공적으로 배포된 환경.
- CI/CD 워크플로의 커밋에 대해 실행되는 모든 테스트의 상태입니다.

다음 절차에서는 이 뷰로 이동하여 프로젝트의 변경 내용을 추적하는 방법을 자세히 설명합니다.

> **Note**
>
> 커밋별 배포 상태 추적은 CodeCatalyst 리포지토리에서만 지원됩니다. GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리에서는 이 기능을 사용할 수 없습니다.

커밋으로 배포 상태를 추적하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 추적 변경을 선택합니다.
4. 기본 창 상단에 있는 두 개의 드롭다운 목록에서 릴리스 상태를 보려는 커밋이 들어 있는 소스 리포지토리와 분기를 선택합니다.
5. 변경사항 보기를 선택합니다.

   커밋 목록이 나타납니다.

   각 커밋에 대해 다음을 확인할 수 있습니다.

   - ID, 작성자, 메시지, 커밋 시기와 같은 커밋 정보 자세한 내용은 소스 리포지토리를 사용하여 코드를 저장하고 공동 작업하십시오. CodeCatalyst 단원을 참조하십시오.
   - 각 환경에 대한 배포 상태. 자세한 내용은 및 에 AWS 계정 배포 VPCs 단원을 참조하십시오.
   - 테스트 및 코드 커버리지 결과. 자세한 내용은 워크플로를 사용한 테스트 단원을 참조하십시오.

   > **Note**
   >
   > 소프트웨어 구성 분석 (SCA) 결과는 표시되지 않습니다.

6. (선택 사항) 최신 배포, 세부 코드 적용 범위, 단위 테스트 정보 등 특정 커밋과 관련된 변경 사항에 대한 자세한 정보를 보려면 해당 커밋에 대한 세부 정보 보기를 선택합니다.

배포 로그 보기

특정 배포 작업과 관련된 로그를 보고 CodeCatalyst Amazon의 문제를 해결할 수 있습니다.

워크플로 또는 환경에서 시작하여 로그를 볼 수 있습니다.

워크플로에서 시작하는 배포 작업의 로그를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. [Run] 을 선택합니다.
6. 애플리케이션을 배포한 워크플로우 실행을 선택합니다.
7. 워크플로 다이어그램에서 로그를 보려는 작업을 선택합니다.
8. 로그 탭을 선택하고 섹션을 확장하여 로그 메시지를 표시합니다.
9. 더 많은 로그를 보려면 요약 탭을 선택한 다음 View in CloudFormation (사용 가능한 경우) 을 선택하여 그곳에서 더 많은 로그를 확인하세요. 에 로그인해야 할 수도 AWS있습니다.

환경에서 시작하는 배포 작업의 로그를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 환경을 선택합니다.
4. 애플리케이션이 배포된 환경을 선택합니다.
5. 배포 활동에서 워크플로 실행 ID 열을 찾아 스택을 배포한 워크플로 실행을 선택합니다.
6. 워크플로 다이어그램에서 로그를 보려는 작업을 선택합니다.
7. 로그 탭을 선택하고 섹션을 확장하여 로그 메시지를 표시합니다.
8. 더 많은 로그를 보려면 요약 탭을 선택한 다음 View in CloudFormation (사용 가능한 경우) 을 선택하여 그곳에서 더 많은 로그를 확인하세요. 에 로그인해야 할 수도 AWS있습니다.

배포 정보 보기

Amazon에서의 배포에 대한 다음 정보를 볼 수 있습니다 CodeCatalyst.

- 배포 상태, 시작 시간, 종료 시간, 기록, 이벤트 기간을 포함한 배포 활동
- 스택 이름 AWS 리전, 마지막 업데이트 시간, 관련 워크플로
- 커밋 및 풀 리퀘스트.

- 작업별 정보 (예: CloudFormation 이벤트 및 출력)

워크플로, 환경 또는 워크플로 작업에서 시작되는 배포 정보를 볼 수 있습니다.

워크플로우에서 시작하는 배포 정보를 보려면

- 애플리케이션을 배포한 워크플로 실행으로 이동합니다. 지침은 워크플로 실행 상태 및 세부 정보 보기 단원을 참조하십시오.

환경에서 시작하는 배포 정보를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 환경을 선택합니다.
4. 스택이 배포된 환경을 선택합니다 (예:). `Production`
5. 배포 활동을 선택하여 스택의 배포 기록, 배포 상태 (예: SUCCEEDED또는 FAILED) 및 기타 배포 관련 정보를 볼 수 있습니다.
6. 환경에 배포된 스택, 클러스터 또는 기타 대상에 대한 정보를 보려면 배포 대상을 선택합니다. 스택 이름, 지역, 공급자, 식별자와 같은 정보를 볼 수 있습니다.

작업에서 시작되는 배포 정보를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 워크플로 다이어그램에서 애플리케이션을 배포한 워크플로 작업을 선택합니다. 예를 들어, 선택할 수 DeployCloudFormationStack있습니다.
6. 오른쪽 창의 내용을 검토하여 작업별 배포 정보를 확인하십시오.

워크플로 생성

워크플로는 지속적 통합 및 지속적 전달 (CI/CD) 시스템의 일부로 코드를 빌드, 테스트 및 배포하는 방법을 설명하는 자동화된 절차입니다. 워크플로는 워크플로 실행 중에 수행할 일련의 단계 또는 조치를 정의합니다. 또한 워크플로는 워크플로를 시작하게 하는 이벤트 또는 트리거를 정의합니다. 워크플로를 설정하려면 CodeCatalyst 콘솔의 시각적 또는 YAML 편집기를 사용하여 워크플로 정의 파일을 만듭니다.

> **Tip**
>
> 프로젝트에서 워크플로를 사용하는 방법을 간단히 살펴보려면 청사진을 사용하여 프로젝트를 만들어 보세요. 각 블루프린트는 검토, 실행, 실험할 수 있는 작동하는 워크플로를 배포합니다.

다음 절차를 사용하여 워크플로를 만들 수 있습니다. CodeCatalyst 워크플로는 선택한 소스 저장소의 `~/.codecatalyst/workflows/` 폴더에 YAML 파일로 저장됩니다. 커밋할 때 워크플로 파일 이름 앞에 폴더 이름을 `~/.codecatalyst/workflows/` 붙여서 의 하위 폴더에 워크플로를 저장할 수도 있습니다. 자세한 내용은 다음 지침을 참조하십시오.

워크플로에 대한 자세한 내용은 워크플로를 통한 빌드, 테스트, 배포 섹션을 참조하세요.

Visual

비주얼 편집기를 사용하여 워크플로를 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로 만들기를 선택합니다.

   워크플로우 만들기 대화 상자가 나타납니다.
5. 소스 리포지토리 필드에서 워크플로 정의 파일이 보관될 소스 리포지토리를 선택합니다. 소스 리포지토리가 없는 경우 새로 만드십시오.
6. 브랜치 필드에서 워크플로 정의 파일이 위치할 브랜치를 선택합니다.
7. 생성(Create)을 선택합니다.

   Amazon은 리포지토리와 브랜치 정보를 메모리에 CodeCatalyst 저장하지만 워크플로는 아직 커밋되지 않았습니다.

8. Visual을 선택하십시오.
9. 워크플로우 구축:

   a. (선택 사항) 워크플로 다이어그램에서 소스 및 트리거 상자를 선택합니다. 트리거 창이 나타납니다. 트리거 추가를 선택하여 트리거를 추가합니다. 자세한 내용은 워크플로에 트리거 추가 단원을 참조하십시오.

   b. + 액션 (왼쪽 상단) 을 선택합니다. 액션 카탈로그가 나타납니다.

   c. 작업 내의 더하기 기호 (+) 를 선택하여 워크플로에 추가합니다. 오른쪽 창을 사용하여 동작을 구성합니다. 자세한 내용은 워크플로에 작업 추가 단원을 참조하십시오.

   d. (선택 사항) 워크플로우 속성 (오른쪽 상단) 을 선택합니다. 워크플로 속성 창이 나타납니다. 워크플로 이름, 실행 모드를 구성하고, 계산합니다. 자세한 내용은 실행의 대기열 동작 구성 및 컴퓨팅 및 런타임 이미지 구성 단원을 참조하세요.

10. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로의 YAML 코드를 검증합니다.
11. 커밋을 선택하고 워크플로 커밋 대화 상자에서 다음을 수행합니다.

    a. 워크플로 파일 이름의 경우 기본 이름을 그대로 두거나 사용자 이름을 입력합니다. 파일은 선택한 소스 리포지토리 및 브랜치의 `~/.codecatalyst/workflows/` 폴더에 저장됩니다. 파일 이름 앞에 폴더 또는 하위 폴더를 붙일 수 있습니다. 예시:

    - `my-workflow`(폴더 없음) 을 지정하면 파일이 다음과 같이 저장됩니다. `~/.codecatalyst/workflows/my-workflow.yaml`
    - 지정하면 파일이 다음과 같이 `folder/subfolder/my-workflow` 저장됩니다. `~/.codecatalyst/workflows/folder/subfolder/my-workflow.yaml`

    b. 커밋 메시지의 경우 기본 메시지를 그대로 두거나 직접 입력합니다.

    c. 리포지토리 및 브랜치의 경우 워크플로우 정의 파일의 소스 리포지토리와 브랜치를 선택합니다. 이러한 필드는 이전에 워크플로 만들기 대화 상자에서 지정한 리포지토리 및 브랜치로 설정해야 합니다. 원하는 경우 지금 리포지토리와 브랜치를 변경할 수 있습니다.

    > **Note**
    >
    > 워크플로 정의 파일을 커밋한 후에는 다른 리포지토리 또는 브랜치에 연결할 수 없으므로 신중하게 선택해야 합니다.

    d. 커밋을 선택하여 워크플로 정의 파일을 커밋합니다.

YAML

YAML편집기를 사용하여 워크플로를 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로 만들기를 선택합니다.

   워크플로우 만들기 대화 상자가 나타납니다.
5. 소스 리포지토리 필드에서 워크플로 정의 파일이 보관될 소스 리포지토리를 선택합니다. 소스 리포지토리가 없는 경우 새로 만드십시오.
6. 브랜치 필드에서 워크플로 정의 파일이 위치할 브랜치를 선택합니다.
7. 생성(Create)을 선택합니다.

   Amazon은 리포지토리와 브랜치 정보를 메모리에 CodeCatalyst 저장하지만 워크플로는 아직 커밋되지 않았습니다.
8. 선택하세요 YAML.
9. 워크플로우 구축:

   a. (선택 사항) YAML 코드에 트리거를 추가합니다. 자세한 내용은 워크플로에 트리거 추가 단원을 참조하십시오.

   b. + 액션 (왼쪽 상단) 을 선택합니다. 액션 카탈로그가 나타납니다.

   c. 작업 내의 더하기 기호 (+) 를 선택하여 워크플로에 추가합니다. 오른쪽 창을 사용하여 동작을 구성합니다. 자세한 내용은 워크플로에 작업 추가 단원을 참조하십시오.

   d. (선택 사항) 워크플로우 속성 (오른쪽 상단) 을 선택합니다. 워크플로 속성 창이 나타납니다. 워크플로 이름, 실행 모드 및 계산을 구성합니다. 자세한 내용은 실행의 대기열 동작 구성 및 컴퓨팅 및 런타임 이미지 구성 단원을 참조하세요.
10. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로의 YAML 코드를 검증합니다.
11. 커밋을 선택하고 워크플로 커밋 대화 상자에서 다음을 수행합니다.

    a. 워크플로 파일 이름의 경우 기본 이름을 그대로 두거나 사용자 이름을 입력합니다. 파일은 선택한 소스 리포지토리 및 브랜치의 `~/.codecatalyst/workflows/` 폴더에 저장됩니다. 파일 이름 앞에 폴더 또는 하위 폴더를 붙일 수 있습니다. 예시:

- `my-workflow`(폴더 없음) 을 지정하면 파일이 다음과 같이 저장됩니다. `~/.codecatalyst/workflows/my-workflow.yaml`
- 지정하면 파일이 다음과 같이 `folder/subfolder/my-workflow` 저장됩니다. `~/.codecatalyst/workflows/folder/subfolder/my-workflow.yaml`

b. 커밋 메시지의 경우 기본 메시지를 그대로 두거나 직접 입력합니다.

c. 리포지토리 및 브랜치의 경우 워크플로우 정의 파일의 소스 리포지토리와 브랜치를 선택합니다. 이러한 필드는 이전에 워크플로 만들기 대화 상자에서 지정한 리포지토리 및 브랜치로 설정해야 합니다. 원하는 경우 지금 리포지토리와 브랜치를 변경할 수 있습니다.

> **Note**
>
> 워크플로 정의 파일을 커밋한 후에는 다른 리포지토리 또는 브랜치에 연결할 수 없으므로 신중하게 선택해야 합니다.

d. 커밋을 선택하여 워크플로 정의 파일을 커밋합니다.

워크플로 실행

실행은 워크플로의 단일 반복입니다. 실행 중에 워크플로 구성 파일에 정의된 작업을 CodeCatalyst 수행하고 관련 로그, 아티팩트 및 변수를 출력합니다.

워크플로우 트리거를 통해 수동으로 시작하거나 자동으로 시작할 수 있습니다. 워크플로 트리거의 예로는 소프트웨어 개발자가 커밋을 메인 브랜치에 푸시하는 경우를 들 수 있습니다.

실수로 시작한 워크플로 처리 도중에 워크플로를 수동으로 중지할 수도 있습니다.

여러 워크플로 실행이 거의 동시에 시작되는 경우 이러한 실행을 대기열에 넣는 방식을 구성할 수 있습니다. 시작된 순서대로 실행이 차례로 대기열에 추가되는 기본 대기열 생성 동작을 사용하거나, 이후 실행이 이전 실행에서 대체 (또는 '인계') 되도록 하여 실행 속도를 높일 수 있습니다. 워크플로가 병렬로 실행되도록 설정하여 다른 실행이 대기하지 않도록 할 수도 있습니다.

워크플로 실행을 수동 또는 자동으로 시작한 후 실행 상태 및 기타 세부 정보를 볼 수 있습니다. 예를 들어, 언제 시작했는지, 누가 시작했는지, 아직 실행 중인지 확인할 수 있습니다.

주제

- 워크플로우 수동 실행 시작
- 트리거를 사용하여 자동으로 워크플로 실행 시작

- 수동 전용 트리거 구성
- 워크플로 실행 중지
- 워크플로 실행 게이팅
- 워크플로 실행 시 승인 필요
- 실행의 대기열 동작 구성
- 워크플로우 실행 간 파일 캐싱
- 워크플로 실행 상태 및 세부 정보 보기

워크플로우 수동 실행 시작

CodeCatalystAmazon에서는 CodeCatalyst 콘솔에서 워크플로를 수동으로 실행할 수 있습니다.

워크플로 실행에 대한 자세한 내용은 을 참조하십시오워크플로 실행.

> **Note**
>
> 트리거를 구성하여 워크플로 실행을 자동으로 시작할 수도 있습니다.

워크플로를 시작하려면 수동으로 실행하십시오.

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. Run(실행)을 선택합니다.

트리거를 사용하여 자동으로 워크플로 실행 시작

워크플로 트리거를 사용하여 Amazon CodeCatalyst 워크플로 실행을 자동으로 시작할 수 있습니다.

워크플로 트리거 또는 단순한 트리거를 사용하면 코드 푸시와 같은 특정 이벤트가 발생할 때 자동으로 워크플로 실행을 시작할 수 있습니다. 소프트웨어 개발자가 CodeCatalyst 콘솔을 통해 수동으로 워크플로 실행을 시작하지 않아도 되도록 트리거를 구성하는 것이 좋습니다.

세 가지 유형의 트리거를 사용할 수 있습니다.

- 푸시 - 코드 푸시 트리거를 사용하면 커밋이 푸시될 때마다 워크플로가 실행됩니다.
- 풀 리퀘스트 — 풀 리퀘스트 트리거를 사용하면 풀 리퀘스트가 생성, 수정 또는 종료될 때마다 워크플로가 실행됩니다.
- 일정 — 일정 트리거를 사용하면 정의된 일정에 따라 워크플로가 실행됩니다. 소프트웨어 개발자가 다음 날 아침에 작업할 수 있도록 최신 빌드를 준비할 수 있도록 스케줄 트리거를 사용하여 야간 소프트웨어 빌드를 실행하는 것을 고려해 보세요.

푸시, 풀 리퀘스트 및 스케줄 트리거를 단독으로 사용하거나 동일한 워크플로우에서 조합하여 사용할 수 있습니다.

트리거는 선택 사항이며, 구성하지 않은 경우 워크플로를 수동으로만 시작할 수 있습니다.

> **Tip**
>
> 트리거가 실제로 작동하는 모습을 보려면 블루프린트가 있는 프로젝트를 시작하세요. 대부분의 블루프린트에는 트리거가 있는 워크플로가 포함되어 있습니다. 블루프린트의 워크플로 정의 파일에서 `Trigger` 프로퍼티를 찾아보세요. 청사진에 대한 자세한 내용은 블루프린트로 프로젝트 생성을 참조하십시오.

주제

- 예: 워크플로의 트리거
- 트리거 및 브랜치에 대한 사용 지침
- 워크플로에 트리거 추가

예: 워크플로의 트리거

다음 예는 Amazon CodeCatalyst 워크플로 정의 파일에 다양한 유형의 트리거를 추가하는 방법을 보여줍니다.

트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.

주제

- 예: 간단한 코드 푸시 트리거
- 예: 간단한 'Push to main' 트리거

- 예: 간단한 풀 리퀘스트 트리거
- 예: 간단한 스케줄 트리거
- 예: 일정과 분기가 있는 트리거
- 예: 일정, 푸시, 브랜치가 있는 트리거
- 예: 풀과 브랜치가 있는 트리거
- 예: 풀, 브랜치, CLOSED '이벤트가 있는 트리거
- 예: 푸시, 브랜치, 파일이 있는 트리거
- 예: 수동 트리거
- 예: CI/CD 다중 워크플로 설정의 트리거

예: 간단한 코드 푸시 트리거

다음 예제는 소스 리포지토리의 브랜치에 코드가 푸시될 때마다 워크플로 실행을 시작하는 트리거를 보여줍니다.

이 트리거가 활성화되면 푸시하려는 브랜치 (즉, 대상 브랜치) 의 파일을 사용하여 워크플로 실행을 CodeCatalyst 시작합니다.

예를 들어 커밋을 `main` 푸시하면 워크플로 정의 파일 및 기타 소스 파일을 사용하여 워크플로 실행을 CodeCatalyst 시작합니다. `main`

또 다른 예로, 커밋을 에 `feature-branch-123` 푸시하면 워크플로 정의 파일 및 기타 소스 파일을 사용하여 워크플로 실행을 CodeCatalyst 시작합니다. `feature-branch-123`

```
Triggers:
  - Type: PUSH
```

> **Note**
>
> 로 푸시할 때만 워크플로 실행이 시작되도록 `main` 하려면 을 참조하십시오. 예: 간단한 'Push to main' 트리거

예: 간단한 'Push to main' 트리거

다음 예제는 소스 리포지토리의 브랜치에 코드가 푸시되고 `main` 브랜치에만 푸시될 때마다 워크플로 실행을 시작하는 트리거를 `main` 보여줍니다.

```
Triggers:
  - Type: PUSH
    Branches:
      - main
```

예: 간단한 풀 리퀘스트 트리거

다음 예제는 소스 리포지토리에서 풀 요청이 생성되거나 수정될 때마다 워크플로 실행을 시작하는 트리거를 보여줍니다.

이 트리거가 활성화되면 가져오는 브랜치 (즉, 소스 브랜치) 의 워크플로 정의 파일과 기타 소스 파일을 사용하여 워크플로 실행을 CodeCatalyst 시작합니다.

예를 들어 소스 브랜치가 `feature-123` 호출되고 대상 브랜치가 호출되는 pull 요청을 만들면 워크플로 정의 파일과 다른 소스 파일을 사용하여 워크플로 실행을 CodeCatalyst 시작합니다. `main` `feature-123`

```
Triggers:
  - Type: PULLREQUEST
    Events:
      - OPEN
      - REVISION
```

예: 간단한 스케줄 트리거

다음 예제는 매주 월요일부터 금요일까지 자정 (UTC+0) 에 워크플로 실행을 시작하는 트리거를 보여줍니다.

이 트리거가 활성화되면 이 트리거가 포함된 워크플로 정의 파일이 들어 있는 소스 리포지토리의 각 분기에 대해 단일 워크플로 실행이 CodeCatalyst 시작됩니다.

예를 들어, 소스 리포지토리에 세 개의 브랜치가 있고 각 브랜치에 다음 트리거가 포함된 워크플로 정의 파일이 들어 있는 경우, CodeCatalyst 의 파일을 사용하는 워크플로`main`, 의 파일을 사용하는 워크플로 `release-v1` 등 세 개의 워크플로가 실행됩니다`feature-123`. `main` `release-v1` `feature-123`

```
Triggers:
  - Type: SCHEDULE
    Expression: "0 0 ? * MON-FRI *"
```

Expression속성에 사용할 수 있는 크론 표현식의 더 많은 예는 을 참조하십시오Expression.

예: 일정과 분기가 있는 트리거

다음 예제는 매일 오후 6시 15분 (UTC+0) 에 워크플로 실행을 시작하는 트리거를 보여줍니다.

이 트리거가 CodeCatalyst 활성화되면 브랜치의 파일을 사용하여 워크플로 실행을 시작하고 로 시작하는 각 main 브랜치에 대해 추가 실행을 시작합니다. release-

예를 들어 소스 bugfix-2 리포지토리에main, release-v1bugfix-1, 라는 브랜치가 있는 경우 워크플로가 두 번 실행됩니다. 한 번은 의 main 파일을 사용하고 다른 하나는 의 파일을 사용합니다release-v1. CodeCatalyst bugfix-1및 bugfix-1 브랜치에 대한 워크플로 실행은 시작되지 않습니다.

```
Triggers:
  - Type: SCHEDULE
    Expression: "15 18 * * ? *"
    Branches:
      - main
      - release\-.*
```

Expression속성에 사용할 수 있는 크론 표현식의 더 많은 예는 을 참조하십시오Expression.

예: 일정, 푸시, 브랜치가 있는 트리거

다음 예제는 매일 자정 (UTC+0) 에, 그리고 브랜치에 코드가 푸시될 때마다 워크플로 실행을 시작하는 트리거를 보여줍니다. main

이 예제에서는 다음이 적용됩니다.

- 워크플로 실행은 매일 자정에 시작됩니다. 워크플로 실행은 main 브랜치의 워크플로 정의 파일 및 기타 소스 파일을 사용합니다.
- 또한 main 브랜치에 커밋을 푸시할 때마다 워크플로 실행이 시작됩니다. 워크플로 실행 시 대상 브랜치 (main) 의 워크플로 정의 파일 및 기타 소스 파일이 사용됩니다.

```
Triggers:
  - Type: SCHEDULE
    Expression: "0 0 * * ? *"
    Branches:
      - main
```

```
  - Type: PUSH
    Branches:
      - main
```

Expression속성에 사용할 수 있는 cron 표현식의 예제를 더 보려면 을 참조하십시오Expression.

예: 풀과 브랜치가 있는 트리거

다음 예제는 대상 브랜치가 있는 pull 요청을 열거나 수정할 때마다 워크플로 실행을 시작하는 트리거를 보여줍니다. main Triggers구성에 지정된 브랜치는 이지만 워크플로 실행에서는 워크플로 정의 파일과 소스 브랜치 (가져오려는 브랜치) 의 다른 소스 파일을 사용합니다. *main*

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main
    Events:
      - OPEN
      - REVISION
```

예: 풀, 브랜치, CLOSED '이벤트가 있는 트리거

다음 예제는 로 시작하는 브랜치에서 pull 요청이 종료될 때마다 워크플로 실행을 시작하는 트리거를 보여줍니다main.

이 예제에서는 다음이 적용됩니다.

- 로 시작하는 대상 브랜치로 풀 리퀘스트를 닫으면 워크플로 정의 파일과 (지금은 폐쇄된) 소스 브랜치의 다른 소스 파일을 사용하여 워크플로 실행이 자동으로 시작됩니다. main
- 풀 리퀘스트가 병합된 후 브랜치를 자동으로 삭제하도록 소스 리포지토리를 구성한 경우 이러한 브랜치는 CLOSED 상태가 될 기회가 전혀 없습니다. 즉, 병합된 브랜치는 풀 리퀘스트 CLOSED 트리거를 활성화하지 않습니다. 이 시나리오에서 CLOSED 트리거를 활성화하는 유일한 방법은 풀 리퀘스트를 병합하지 않고 종료하는 것입니다.

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main.*
    Events:
```

```
      - CLOSED
```

예: 푸시, 브랜치, 파일이 있는 트리거

다음 예제는 `main` 브랜치의 `filename.txt` 파일 또는 `src` 디렉터리에 있는 파일이 변경될 때마다 워크플로 실행을 시작하는 트리거를 보여줍니다.

이 트리거가 활성화되면 워크플로 정의 파일 및 `main` 브랜치의 다른 소스 파일을 사용하여 워크플로 실행을 CodeCatalyst 시작합니다.

```
Triggers:
  - Type: PUSH
    Branches:
      - main
    FilesChanged:
      - filename.txt
      - src\/.*
```

예: 수동 트리거

수동 트리거를 구성하려면 워크플로 정의 파일에서 `Triggers` 섹션을 생략하십시오. 이 섹션이 없으면 사용자는 CodeCatalyst 콘솔에서 실행 버튼을 선택하여 워크플로를 수동으로 시작해야 합니다. 자세한 내용은 워크플로우 수동 실행 시작 단원을 참조하십시오.

예: CI/CD 다중 워크플로 설정의 트리거

이 예제에서는 지속적 통합 (CI) 및 지속적 배포 (CD) 에 별도의 Amazon CodeCatalyst 워크플로를 사용하려는 경우 트리거를 설정하는 방법을 설명합니다.

이 시나리오에서는 두 가지 워크플로를 설정합니다.

- CI 워크플로 — 이 워크플로는 pull 요청이 생성되거나 수정될 때 애플리케이션을 빌드하고 테스트합니다.
- CD 워크플로 — 이 워크플로는 풀 리퀘스트가 병합될 때 애플리케이션을 빌드하고 배포합니다.

CI 워크플로의 정의 파일은 다음과 비슷합니다.

```
Triggers:
  - Type: PULLREQUEST
```

```
    Branches:
      - main
    Events:
      - OPEN
      - REVISION
Actions:
  BuildAction:
    instructions-for-building-the-app
  TestAction:
    instructions-for-test-the-app
```

Triggers코드는 소프트웨어 개발자가 기능 브랜치를 브랜치에 병합해 달라는 pull 요청을 생성 (또는 수정) 할 때마다 워크플로가 자동으로 실행되도록 지시합니다. main CodeCatalyst 소스 브랜치 (기능 브랜치) 의 소스 코드를 사용하여 워크플로 실행을 시작합니다.

CD 워크플로의 정의 파일은 다음과 비슷합니다.

```
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildAction:
    instructions-for-building-the-app
  DeployAction:
    instructions-for-deploying-the-app
```

Triggers코드는 병합이 main 발생할 때 워크플로가 자동으로 시작되도록 지시합니다. CodeCatalyst main브랜치의 소스 코드를 사용하여 워크플로 실행을 시작합니다.

트리거 및 브랜치에 대한 사용 지침

이 섹션에서는 브랜치를 포함하는 Amazon CodeCatalyst 트리거를 설정할 때의 몇 가지 주요 지침을 설명합니다.

트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.

- 지침 1: 푸시 및 풀 요청 트리거 모두에 대해 브랜치를 지정하려면 트리거 구성에서 대상 (또는 'to') 브랜치를 지정해야 합니다. 소스 (또는 'from') 브랜치를 지정하지 마십시오.

  다음 예시에서는 브랜치에서 푸시를 실행하여 워크플로를 main 활성화합니다.

```
Triggers:
  - Type: PUSH
    Branches:
      - main
```

다음 예시에서는 어떤 브랜치에서든 pull 요청을 받으면 워크플로가 main 활성화됩니다.

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main
    Events:
      - OPEN
      - REVISION
```

- 지침 2: 푸시 트리거의 경우 워크플로가 활성화되면 대상 브랜치의 워크플로 정의 파일과 소스 파일을 사용하여 워크플로가 실행됩니다.
- 지침 3: 풀 요청 트리거의 경우 워크플로가 활성화되면 워크플로는 소스 브랜치의 워크플로 정의 파일과 소스 파일을 사용하여 실행됩니다 (트리거 구성에서 대상 브랜치를 지정했더라도).
- 지침 4: 한 브랜치에서 정확히 동일한 트리거가 다른 브랜치에서는 실행되지 않을 수 있습니다.

  다음 푸시 트리거를 고려해 보세요.

```
Triggers:
  - Type: PUSH
    Branches:
      - main
```

  에 이 트리거가 포함된 워크플로 정의 파일이 main 존재하고 복제되면 워크플로가 의 파일을 사용하여 자동으로 시작되지 않습니다. 단test, test 워크플로를 수동으로 시작하여 파일을 사용하도록 할 수는 있습니다. test 지침 2를 검토하여 의 파일을 사용해도 워크플로가 자동으로 실행되지 않는 이유를 이해하십시오. test

  다음과 같은 풀 리퀘스트 트리거도 고려해 보세요.

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main
```

```
    Events:
      - OPEN
      - REVISION
```

에 이 트리거가 포함된 워크플로 정의 파일이 있는 `main` 경우 의 파일을 사용하여 워크플로가 실행되지 않습니다`main`. (하지만 에서 `test` 브랜치를 `main` 만들면 의 파일을 사용하여 워크플로가 실행됩니다`test`.) 지침 3을 검토하여 그 이유를 이해하세요.

워크플로에 트리거 추가

다음 지침에 따라 Amazon CodeCatalyst 워크플로에 푸시, 풀 또는 스케줄 트리거를 추가하십시오.

트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.

Visual

트리거를 추가하려면 (비주얼 에디터)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로 다이어그램에서 소스 및 트리거 상자를 선택합니다.
8. 구성 창에서 트리거 추가를 선택합니다.
9. 트리거 추가 대화 상자의 필드에 다음과 같이 정보를 입력합니다.

   트리거 유형

   트리거 유형을 지정합니다. 다음 값 중 하나를 사용할 수 있습니다.

   - 푸시 (비주얼 에디터) 또는 PUSH (YAML에디터)

     푸시 트리거는 변경 내용이 소스 저장소에 푸시될 때 워크플로 실행을 시작합니다. 워크플로 실행 시 푸시하려는 브랜치 (즉, 대상 브랜치) 의 파일이 사용됩니다.
   - 풀 리퀘스트 (비주얼 에디터) 또는 PULLREQUEST (YAML에디터)

풀 리퀘스트 트리거는 소스 리포지토리에서 풀 리퀘스트를 열거나 업데이트하거나 닫을 때 워크플로 실행을 시작합니다. 워크플로 실행에서는 가져오는 브랜치 (즉, 소스 브랜치) 의 파일을 사용합니다.

- 스케줄 (비주얼 에디터) 또는 SCHEDULE (YAML에디터)

  스케줄 트리거는 지정한 cron 표현식으로 정의된 일정에 따라 워크플로가 실행됩니다. 브랜치의 파일을 사용하여 소스 리포지토리의 각 브랜치에 대해 별도의 워크플로 실행이 시작됩니다. (트리거가 활성화되는 브랜치를 제한하려면 Branch 필드 (시각적 편집기) 또는 **Branches** 속성 (YAML편집기) 을 사용하십시오.)

  스케줄 트리거를 구성할 때는 다음 가이드라인을 따르십시오.

  - 워크플로우당 하나의 스케줄 트리거만 사용하십시오.
  - CodeCatalyst 스페이스에 여러 워크플로를 정의한 경우 동시에 시작하도록 10개 이하로 예약하는 것이 좋습니다.
  - 실행 간격을 충분히 두고 트리거의 cron 표현식을 구성해야 합니다. 자세한 내용은 Expression 단원을 참조하십시오.

예제는 예: 워크플로의 트리거 섹션을 참조하세요.

풀 리퀘스트용 이벤트

이 필드는 풀 요청 트리거 유형을 선택한 경우에만 나타납니다.

워크플로 실행을 시작할 풀 요청 이벤트의 유형을 지정합니다. 유효한 값은 다음과 같습니다.

- 풀 리퀘스트가 생성됨 (비주얼 에디터) 또는 OPEN (YAML에디터)

  풀 리퀘스트가 생성되면 워크플로 실행이 시작됩니다.

- 풀 리퀘스트가 종료됨 (비주얼 에디터) 또는 CLOSED (YAML에디터)

  풀 리퀘스트가 종료되면 워크플로 실행이 시작됩니다. CLOSED이벤트의 동작은 까다롭기 때문에 예제를 통해 가장 잘 이해할 수 있습니다. 자세한 내용은 예: 풀, 브랜치, CLOSED '이벤트가 있는 트리거 섹션을 참조하세요.

- 풀 리퀘스트 (비주얼 에디터) 또는 **REVISION** (YAML에디터) 가 새롭게 수정되었습니다.

  풀 리퀘스트에 대한 수정이 생성되면 워크플로 실행이 시작됩니다. 풀 리퀘스트가 생성되면 첫 번째 리비전이 생성됩니다. 그 이후에는 누군가가 풀 리퀘스트에 지정된 소스 브랜치에

새 커밋을 푸시할 때마다 새 리비전이 생성됩니다. 풀 리퀘스트 트리거에 REVISION 이벤트를 포함하면 OPEN 이벤트를 생략할 수 있습니다. 는 의 상위 REVISION 집합이기 때문입니다. OPEN

동일한 풀 리퀘스트 트리거에 여러 이벤트를 지정할 수 있습니다.

예제는 예: 워크플로의 트리거 섹션을 참조하세요.

Schedule

이 필드는 스케줄 트리거 유형을 선택한 경우에만 나타납니다.

예약된 워크플로를 실행하려는 시기를 설명하는 cron 표현식을 지정하십시오.

의 Cron 표현식은 각 필드를 공백으로 구분하는 다음과 같은 6개 필드 구문을 CodeCatalyst 사용합니다.

minutes hours days-of-month month days-of-week year

크론 표현식의 예

분	시간	한 달의 요일	월	요일	연도	의미
0	0	?	*	MON-FRI	*	매주 월요일부터 금요일까지 자정 (UTC+0)에 워크플로를 실행합니다.

분	시간	한 달의 요일	월	요일	연도	의미
0	2	*	*	?	*	매일 오전 2시 (UTC+0) 에 워크플로를 실행합니다.
15	22	*	*	?	*	매일 오후 10시 15분 (UTC+0) 에 워크플로를 실행합니다.
0/30	22-2	?	*	SAT-SUN	*	시작일 오후 10시부터 다음 날 오전 2시 (UTC+0) 사이에 토요일부터 일요일까지 30분마다 워크플로를 실행합니다.

분	시간	한 달의 요일	월	요일	연도	의미
45	13	L	*	?	2023-2027	2023년부터 2027년까지 매월 마지막 날 오후 1시 45분 (UTC+0)에 워크플로를 실행합니다.

에서 크론 표현식을 지정할 때는 다음 CodeCatalyst 지침을 준수해야 합니다.

- 트리거당 SCHEDULE 하나의 cron 표현식을 지정하십시오.
- 편집기에서 cron 표현식을 큰따옴표 () " 로 묶습니다. YAML
- 시간을 협정 세계시 () 로 지정합니다. UTC 다른 시간대는 지원되지 않습니다.
- 실행 간격을 최소 30분으로 설정하세요. 더 빠른 케이던스는 지원되지 않습니다.
- 다음을 지정하십시오.*days-of-month* 또는 *days-of-week* 필드를 둘 다 포함하지는 마세요. 필드 중 하나에 값이나 별표 (*) 를 지정하는 경우 다른 필드에는 물음표 (?) 를 사용해야 합니다. 별표는 '모두'를 의미하고 물음표는 '모두'를 의미합니다.

cron 표현식의 추가 예와 ? *L, 및 같은 와일드카드에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 Cron 표현식 참조를 참조하십시오. EventBridge 와 에서의 크론 표현식은 정확히 같은 방식으로 CodeCatalyst 작동합니다.

스케줄 트리거에 대한 예는 을 참조하십시오. 예: 워크플로의 트리거

브랜치 및 브랜치 패턴

(선택 사항)

워크플로우 실행 시작 시기를 알기 위해 트리거가 모니터링하는 소스 리포지토리의 브랜치를 지정하십시오. 정규식 패턴을 사용하여 브랜치 이름을 정의할 수 있습니다. 예를 들어 로 시작하는 모든 브랜치를 `main.*` 일치시키는 데 사용합니다. `main`

지정할 브랜치는 트리거 유형에 따라 다릅니다.

- 푸시 트리거의 경우 푸시하려는 브랜치, 즉 대상 브랜치를 지정하세요. 일치하는 브랜치의 파일을 사용하여 일치하는 브랜치당 한 번의 워크플로 실행이 시작됩니다.

  예: `main.*`, `mainline`
- 풀 리퀘스트 트리거의 경우 푸시하려는 브랜치, 즉 대상 브랜치를 지정하세요. 일치하는 브랜치가 아닌 소스 브랜치의 소스 파일과 워크플로 정의 파일을 사용하여 일치하는 브랜치당 한 번의 워크플로 실행이 시작됩니다.

  예:`main.*`,`mainline`, `v1\-.*` (로 시작하는 브랜치와 일치`v1-`)
- 스케줄 트리거의 경우 예약 실행에서 사용할 파일이 포함된 브랜치를 지정하십시오. 일치하는 브랜치의 워크플로 정의 파일과 소스 파일을 사용하여 일치하는 브랜치당 한 번의 워크플로 실행이 시작됩니다.

  예: `main.*`, `version\-1\.0`

> **Note**
>
> 브랜치를 지정하지 않으면 트리거는 소스 리포지토리의 모든 브랜치를 모니터링하고 다음 위치에 있는 워크플로 정의 파일 및 소스 파일을 사용하여 워크플로 실행을 시작합니다.
>
> - 푸시하려는 브랜치 (푸시 트리거용). 자세한 내용은 예: 간단한 코드 푸시 트리거 단원을 참조하십시오.
> - 가져오려는 브랜치 (풀 리퀘스트 트리거용). 자세한 내용은 예: 간단한 풀 리퀘스트 트리거 단원을 참조하십시오.
> - 모든 브랜치 (스케줄 트리거용). 소스 리포지토리의 브랜치당 한 번의 워크플로 실행이 시작됩니다. 자세한 내용은 예: 간단한 스케줄 트리거 단원을 참조하십시오.

브랜치 및 트리거에 대한 자세한 내용은 을 참조하십시오트리거 및 브랜치에 대한 사용 지침.

더 많은 예제는 예: 워크플로의 트리거를 참조합니다.

파일 변경됨

이 필드는 푸시 또는 풀 요청 트리거 유형을 선택한 경우에만 나타납니다.

워크플로우 실행 시작 시기를 알기 위해 트리거가 모니터링하는 소스 리포지토리의 파일 또는 폴더를 지정하십시오. 정규 표현식을 사용하여 파일 이름이나 경로를 일치시킬 수 있습니다.

예제는 예: 워크플로의 트리거 섹션을 참조하세요.

10. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사합니다.
11. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

트리거를 추가하려면 (YAML편집기)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 다음 예제를 가이드로 사용하여 `Triggers` 섹션과 기본 속성을 추가합니다. 자세한 내용은 워크플로우 YAML 정의의 Triggers를 참조하십시오.

   코드 푸시 트리거는 다음과 같을 수 있습니다.

   ```
   Triggers:
     - Type: PUSH
       Branches:
         - main
   ```

   풀 리퀘스트 트리거는 다음과 같을 수 있습니다.

```
Triggers:
  - Type: PULLREQUEST
    Branches:
      - main.*
    Events:
      - OPEN
      - REVISION
      - CLOSED
```

스케줄 트리거는 다음과 같을 수 있습니다.

```
Triggers:
  - Type: SCHEDULE
    Branches:
      - main.*
    # Run the workflow at 1:15 am (UTC+0) every Friday until the end of 2023
    Expression: "15 1 ? * FRI 2022-2023"
```

`Expression`속성에 사용할 수 있는 cron 표현식의 더 많은 예는 을 참조하십시오Expression.

푸시, 풀 요청, 스케줄 트리거에 대한 더 많은 예는 을 참조하십시오. 예: 워크플로의 트리거

8. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로의 YAML 코드를 검증하십시오.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

수동 전용 트리거 구성

팀이 콘솔의 실행 버튼을 사용하여 수동으로만 시작할 수 있도록 워크플로를 제한할 수 있습니다. CodeCatalyst 이 기능을 구성하려면 워크플로 정의 파일에서 해당 `Triggers` 섹션을 제거해야 합니다. `Triggers`섹션은 워크플로를 만들 때 기본적으로 포함되지만 섹션은 선택 사항이므로 제거할 수 있습니다.

워크플로를 수동으로만 시작할 수 있도록 워크플로 정의 파일에서 `Triggers` 섹션을 제거하려면 다음 지침을 따르십시오.

트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.

워크플로 실행에 대한 자세한 내용은 을 참조하십시오워크플로 실행.

Visual

'트리거' 섹션 삭제하기 (비주얼 에디터)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로우 다이어그램에서 소스 상자를 선택합니다.
8. 트리거에서 휴지통 아이콘을 선택하여 워크플로에서 `Triggers` 섹션을 제거합니다.
9. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사합니다.
10. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

'트리거' 섹션을 제거하려면 (YAML편집기)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 해당 `Triggers` 섹션을 찾아 삭제하세요.
8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드를 검증합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

워크플로 실행 중지

진행 중인 워크플로 실행을 중지하려면 다음 절차를 사용하십시오. 실수로 시작된 실행은 중지하는 것이 좋습니다.

워크플로 실행을 중지하면 진행 중인 작업이 완료될 때까지 CodeCatalyst 기다린 다음 콘솔에서 실행을 중지됨으로 표시합니다. CodeCatalyst 시작할 기회가 없었던 모든 작업은 시작되지 않고 중단된 것으로 표시됩니다.

> **Note**
>
> 실행이 대기열에 있는 경우 (즉, 진행 중인 작업이 없는 경우) 실행은 즉시 중지됩니다.

워크플로 실행에 대한 자세한 내용은 을 참조하십시오. 워크플로 실행

워크플로 실행을 중지하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로우에서 실행을 선택하고 목록에서 진행 중인 실행을 선택합니다.
5. 중지를 선택합니다.

워크플로 실행 게이팅

게이트는 특정 조건이 충족되지 않는 한 워크플로가 진행되지 않도록 하는 데 사용할 수 있는 워크플로 구성 요소입니다. 게이트의 예로는 사용자가 CodeCatalyst 콘솔에서 승인을 제출해야 워크플로 실행을 계속할 수 있는 승인 게이트가 있습니다.

워크플로의 작업 시퀀스 사이에 게이트를 추가하거나 소스 다운로드 직후에 실행되는 첫 번째 작업 앞에 게이트를 추가할 수 있습니다. 필요한 경우 마지막 작업 뒤에 게이트를 추가할 수도 있습니다.

워크플로 실행에 대한 자세한 내용은 을 참조하십시오워크플로 실행.

주제

- 게이트 유형

게이트 유형

현재 CodeCatalyst Amazon은 승인 게이트라는 한 가지 유형의 게이트를 지원합니다. 자세한 내용은 워크플로 실행 시 승인 필요 단원을 참조하십시오.

게이트가 다른 액션과 병렬로 실행되도록 설정할 수 있나요?

아니요. 게이츠는 액션 이전 또는 이후에만 실행할 수 있습니다. 자세한 내용은 시퀀싱 게이트 및 액션 단원을 참조하십시오.

게이트를 사용하여 워크플로 실행이 시작되지 않도록 할 수 있나요?

네, 자격 요건을 충족해야 합니다.

워크플로우 실행이 작업을 수행하지 못하도록 할 수 있습니다. 이는 시작되지 않도록 하는 것과 약간 다릅니다.

워크플로가 작업을 수행하지 못하게 하려면 워크플로의 첫 번째 작업 앞에 게이트를 추가하십시오. 이 시나리오에서는 워크플로 실행이 시작되어 소스 리포지토리 파일이 다운로드되지만 게이트가 잠금 해제될 때까지는 작업을 수행할 수 없습니다.

> **Note**
>
> 시작되었다가 게이트에 의해 차단되는 워크플로는 여전히 공간 할당량당 최대 동시 워크플로 실행 수 및 기타 할당량에 포함됩니다. 워크플로 할당량을 초과하지 않도록 하려면 게이트를 사용하는 대신 워크플로 트리거를 사용하여 워크플로를 조건부로 시작하는 것이 좋습니다. 또한 게이트 대신 풀 리퀘스트 승인 규칙을 사용하는 것도 고려해 보십시오. 할당량, 트리거, 풀 리퀘스트 승인 규칙에 대한 자세한 내용은의 워크플로우 할당량 CodeCatalyst, 트리거를 사용하여 자동으로 워크플로 실행 시작 및 을 참조하십시오. 풀 리퀘스트를 승인 규칙과 병합하기 위한 요구 사항 관리

게이트 제한

Gates에는 다음과 같은 제한 사항이 있습니다.

- Gates는 컴퓨팅 공유 기능과 함께 사용할 수 없습니다. 이 기능에 대한 자세한 내용은 작업 간 컴퓨팅 공유을 참조하세요.
- 액션 그룹 내에서는 게이트를 사용할 수 없습니다. 액션 그룹에 대한 자세한 내용은 을 참조하십시오작업을 작업 그룹으로 그룹화.

워크플로에 게이트 추가

CodeCatalystAmazon에서는 워크플로에 게이트를 추가하여 특정 조건이 충족되지 않는 한 워크플로가 진행되지 않도록 할 수 있습니다. 워크플로에 게이트를 추가하려면 다음 지침을 따르십시오.

게이트에 대한 자세한 내용은 을 참조하십시오워크플로 실행 게이팅.

게이트 추가 및 구성하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 왼쪽에서 Gates를 선택합니다.
8. 게이트 카탈로그에서 게이트를 검색한 다음 더하기 기호 (+) 를 선택하여 워크플로에 게이트를 추가합니다.
9. 게이트를 구성합니다. 비주얼 에디터를 사용하거나 에디터를 YAML사용하려면 Visual을 YAML 선택합니다. 자세한 지침은 다음을 참조하십시오.
    - '승인' 게이트 추가
10. (선택 사항) Validate를 선택하여 YAML 코드가 유효한지 확인합니다.
11. 커밋을 선택하여 변경 내용을 적용합니다.

시퀀싱 게이트 및 액션

Amazon에서는 워크플로 작업 CodeCatalyst, 작업 그룹 또는 게이트 이전 또는 이후에 실행되도록 게이트를 설정할 수 있습니다. 예를 들어, `Deploy` 작업 전에 실행되도록 `Approval` 게이트를 설정할 수 있습니다. 이 경우 `Deploy` 액션은 `Approval` 게이트에 따라 달라진다고 합니다.

게이트와 액션 간의 종속성을 설정하려면 게이트 또는 액션의 Dependson 속성을 구성하십시오. 지침은 액션 간 종속성 설정 단원을 참조하십시오. 참조된 지침은 워크플로 작업을 참조하지만 게이트에도 동일하게 적용됩니다.

게이트를 사용하여 Dependson 속성을 설정하는 방법에 대한 예는 을 참조하십시오예: '승인' 게이트.

게이트에 대한 자세한 내용은 을 참조하십시오워크플로 실행 게이팅.

워크플로 작업에 대한 자세한 내용은 을 참조하십시오워크플로우 작업 구성.

게이트 버전 지정하기

기본적으로 워크플로에 게이트를 추가하면 워크플로 정의 파일에 다음 형식을 사용하여 전체 버전이 CodeCatalyst 추가됩니다.

v*major*.*minor*.*patch*

예:

```
My-Gate:
  Identifier: aws/approval@v1
```

워크플로에서 게이트의 특정 메이저 또는 마이너 버전을 사용하도록 버전 길이를 늘릴 수 있습니다. 지침은 사용할 액션 버전 지정 단원을 참조하십시오. 참조된 항목은 워크플로 작업을 참조하지만 게이트에도 동일하게 적용됩니다.

게이트 인에 대한 자세한 내용은 CodeCatalyst 을 참조하십시오워크플로 실행 게이팅.

워크플로 실행 시 승인 필요

진행하기 전에 승인이 필요하도록 워크플로 실행을 구성할 수 있습니다. 이렇게 하려면 워크플로에 승인 게이트를 추가해야 합니다. 승인 게이트는 사용자 또는 사용자 집합이 콘솔에서 하나 이상의 승인을 제출할 때까지 워크플로가 진행되지 않도록 합니다. CodeCatalyst 승인이 모두 이루어지면 게이트가 '잠금 해제'되고 워크플로 실행을 재개할 수 있습니다.

워크플로우에 승인 게이트를 사용하면 개발팀, 운영팀, 경영진에게 변경 사항을 더 많은 사람들에게 배포하기 전에 검토할 기회를 제공할 수 있습니다.

워크플로 실행에 대한 자세한 내용은 을 참조하십시오워크플로 실행.

주제

승인 게이트를 잠금 해제하려면 어떻게 해야 합니까?

승인 게이트를 잠금 해제하려면 다음 조건을 모두 충족해야 합니다.

- 조건 1: 필요한 수의 승인을 제출해야 합니다. 필요한 승인 수는 구성할 수 있으며 각 사용자는 단일 승인을 제출할 수 있습니다.
- 조건 2: 게이트 타임아웃 전에 모든 승인을 제출해야 합니다. 게이트는 활성화된 지 14일이 지나면 제한 시간이 초과됩니다. 이 기간은 구성할 수 없습니다.
- 조건 3: 누구도 워크플로 실행을 거부해서는 안 됩니다. 한 번 거부하면 워크플로 실행이 실패합니다.
- 조건 4: (대체된 실행 모드를 사용하는 경우에만 적용됩니다.) 해당 실행을 이후 실행으로 대체해서는 안 됩니다. 자세한 내용은 대기열, 대체 및 병렬 실행 모드에서 워크플로우 승인은 어떻게 작동합니까? 단원을 참조하십시오.

조건이 하나라도 충족되지 않으면 워크플로를 CodeCatalyst 중지하고 실행 상태를 실패 (조건 1~3의 경우) 또는 대체됨 (조건 4의 경우) 으로 설정합니다.

'승인' 게이트 사용 시기

일반적으로 승인 게이트는 애플리케이션 및 기타 리소스를 프로덕션 서버 또는 품질 표준의 검증이 필요한 환경에 배포하는 워크플로우에서 사용합니다. 프로덕션 환경에 배포하기 전에 게이트를 설치하면 검토자가 새 소프트웨어 개정판을 대중에게 공개하기 전에 검증할 기회를 얻을 수 있습니다.

누가 승인을 제공할 수 있나요?

프로젝트의 멤버이자 기여자 또는 프로젝트 관리자 역할을 가진 모든 사용자가 승인을 제공할 수 있습니다. 프로젝트 스페이스에 속하는 스페이스 관리자 역할을 가진 사용자도 승인을 제공할 수 있습니다.

> **Note**
>
> 검토자 역할을 가진 사용자는 승인을 제공할 수 없습니다.

승인이 필요함을 사용자에게 알리려면 어떻게 해야 하나요?

승인이 필요함을 사용자에게 알리려면 다음을 수행해야 합니다.

- 그들에게 Slack 알림을 CodeCatalyst 보내도록 하세요. 자세한 내용은 승인 알림 구성 단원을 참조하십시오.
- 승인 및 거부 버튼이 있는 CodeCatalyst 콘솔의 페이지로 이동하여 승인자에게 보내는 이메일 또는 메시징 애플리케이션에 해당 페이지를 붙여넣습니다. URL 이 페이지로 이동하는 방법에 대한 자세한 내용은 을 참조하십시오. 워크플로 실행 승인 또는 거부

'승인' 게이트를 사용하여 워크플로 실행이 시작되지 않도록 할 수 있습니까?

예, 자격 조건이 있습니다. 자세한 내용은 게이트를 사용하여 워크플로 실행이 시작되지 않도록 할 수 있나요? 단원을 참조하십시오.

대기열, 대체 및 병렬 실행 모드에서 워크플로우 승인은 어떻게 작동합니까?

대기, 대체 또는 병렬 실행 모드를 사용하는 경우 승인 게이트는 작업과 유사한 방식으로 작동합니다. 대기 실행 모드에 대한 정보,대체 실행 모드에 대한 정보, 병렬 실행 모드에 대한 정보 섹션을 읽고 이러한 실행 모드에 익숙해지는 것이 좋습니다. 기본적인 사항을 이해했다면 이 섹션으로 돌아가 승인 게이트가 있을 때 이러한 실행 모드가 어떻게 작동하는지 알아보십시오.

승인 게이트가 있는 경우 실행은 다음과 같이 처리됩니다.

- 대기 실행 모드를 사용하는 경우, 현재 게이트에서 승인을 기다리고 있는 달리기 뒤에 달리기가 대기열에 추가됩니다. 게이트가 잠금 해제되면 (즉, 모든 승인이 승인된 경우) 대기열에 있는 다음 실행이 게이트로 진행되어 승인을 기다립니다. 이 프로세스는 게이트를 통해 대기중인 실행이 처리되면서 계속됩니다. one-by-one Figure 1이 프로세스를 보여 줍니다.
- 대체된 실행 모드를 사용하는 경우 동작은 대기 실행 모드의 동작과 동일합니다. 단, 게이트 대기열에 런이 쌓이는 대신 새 런이 이전 런을 대체 (인계) 한다는 점이 다릅니다. 대기열이 없으며, 게이트에서 현재 승인을 기다리고 있는 모든 실행은 취소되고 새 실행으로 대체됩니다. Figure 2이 프로세스를 보여 줍니다.
- 병렬 실행 모드를 사용하는 경우 실행은 병렬로 시작되며 대기열 없이 실행됩니다. 앞에 런이 없으므로 각 런은 게이트에서 즉시 처리됩니다. Figure 3이 과정을 보여줍니다.

그림 1: '대기 실행 모드' 및 승인 게이트

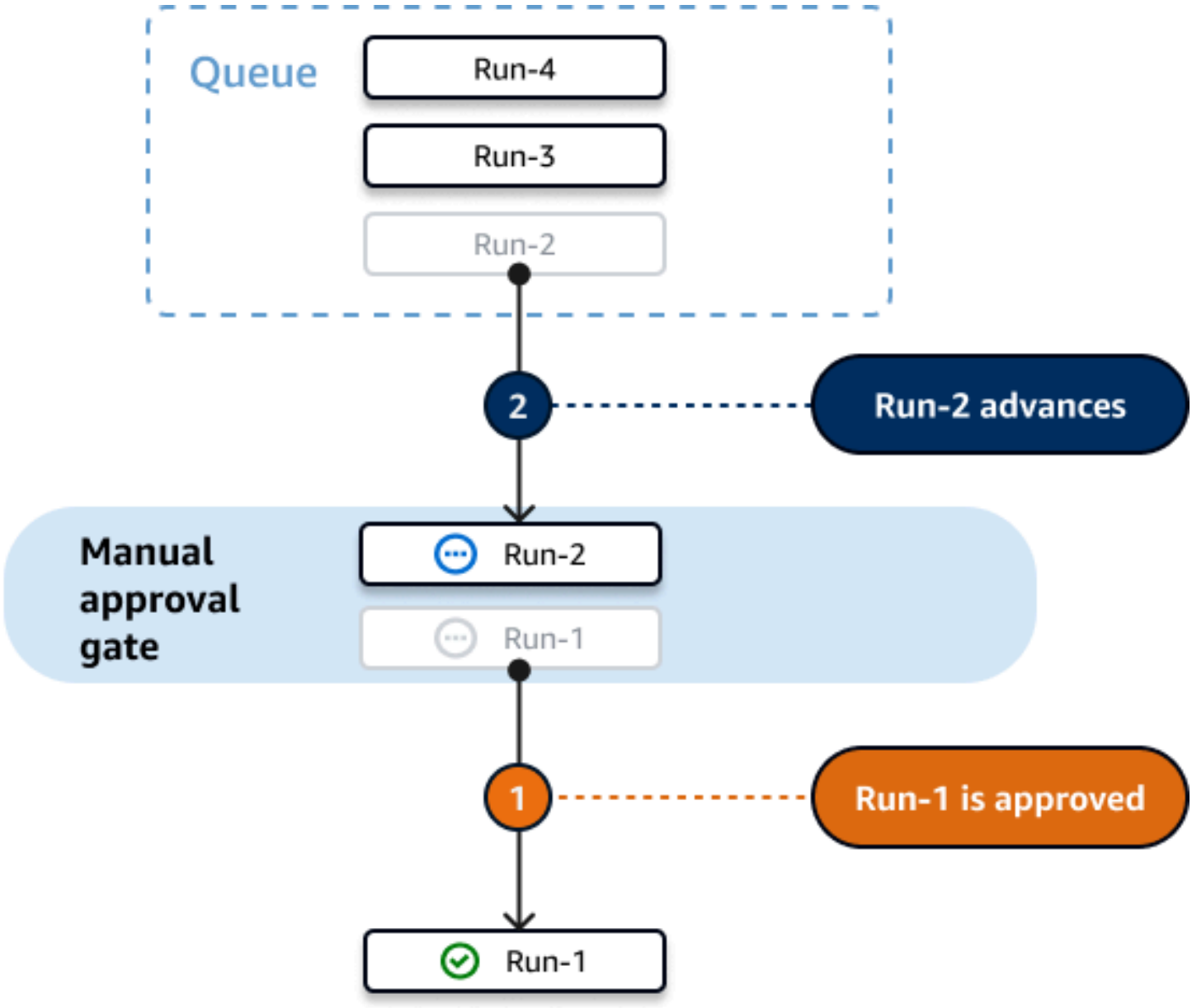


그림 2: '대체된 실행 모드' 및 승인 게이트

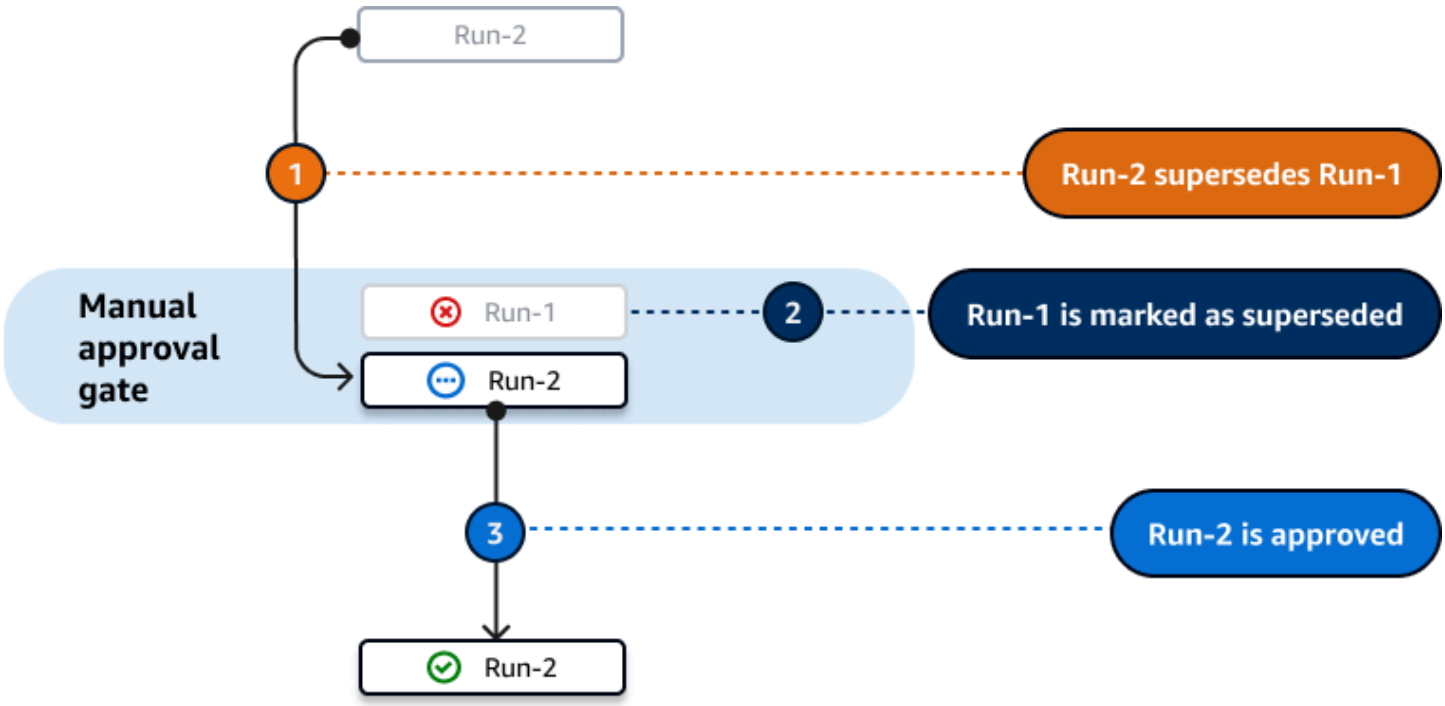
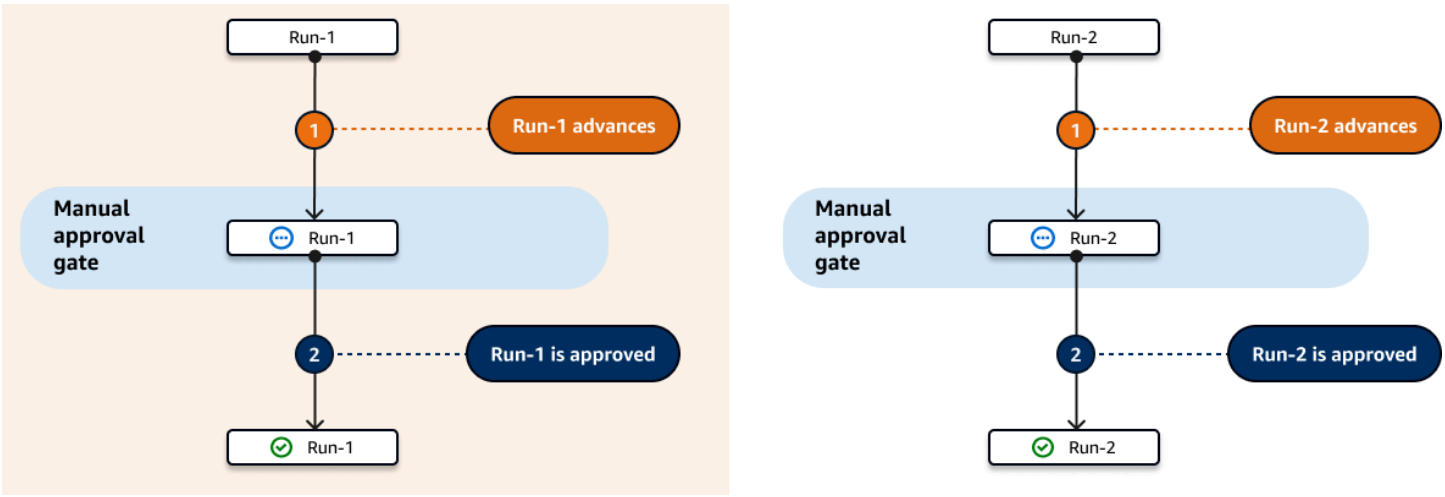


그림 3: '병렬 실행 모드' 및 승인 게이트



예: '승인' 게이트

다음 예제는Staging,, 라는 두 작업 Approval_01 사이에 승인 게이트 호출을 추가하는 방법을 보여줍니다Production. Staging액션이 먼저 실행되고, Approval_01 게이트가 두 번째로 실행되고, Production 액션이 마지막에 실행됩니다. Approval_01게이트가 잠금 해제된 경우에만 Production 액션이 실행됩니다. 이 DependsOn 속성은Staging,Approval_01, Production 단계가 순차적으로 실행되도록 합니다.

승인 게이트에 대한 자세한 내용은 을 참조하십시오. 워크플로 실행 시 승인 필요

```
Actions:
  Staging: # Deploy to a staging server
    Identifier: aws/ecs-deploy@v1
    Configuration:
    ...
  Approval_01:
    Identifier: aws/approval@v1
    DependsOn:
      - Staging
    Configuration:
      ApprovalsRequired: 2
  Production: # Deploy to a production server
    Identifier: aws/ecs-deploy@v1
    DependsOn:
      - Approval_01
    Configuration:
    ...
```

'승인' 게이트 추가

승인을 요구하도록 워크플로를 구성하려면 워크플로에 승인 게이트를 추가해야 합니다. 다음 지침에 따라 워크플로에 승인 게이트를 추가하십시오.

이 게이트에 대한 자세한 내용은 을 참조하십시오워크플로 실행 시 승인 필요.

Visual

워크플로에 '승인' 게이트 추가하기 (비주얼 에디터)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 왼쪽 상단에서 Gates를 선택합니다.
7. Gates 카탈로그의 승인에서 더하기 기호 (+) 를 선택합니다.
8. 입력을 선택하고 종속 대상 필드에서 다음을 수행하십시오.

이 게이트를 실행하기 위해 성공적으로 실행되어야 하는 액션, 액션 그룹 또는 게이트를 지정하십시오. 기본적으로 워크플로에 게이트를 추가하면 워크플로의 마지막 작업에 따라 게이트가 설정되도록 설정됩니다. 이 속성을 제거하면 게이트는 아무 것에도 종속되지 않고 다른 작업보다 먼저 실행됩니다.

> **Note**
>
> 게이트는 액션, 액션 그룹 또는 게이트 이전 또는 이후에 실행되도록 구성되어야 합니다. 다른 액션, 액션 그룹 및 게이트와 병렬로 실행되도록 설정할 수 없습니다.

종속 기능에 대한 자세한 내용은 을 참조하십시오시퀀싱 게이트 및 액션.

9. 구성 탭을 선택합니다.
10. 게이트 이름 필드에서 다음을 수행하십시오.

    게이트에 부여하려는 이름을 지정합니다. 모든 게이트 이름은 워크플로우 내에서 고유해야 합니다. 게이트 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-) 및 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 게이트 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.
11. (선택 사항) 승인 수 필드에서 다음을 수행하십시오.

    승인 게이트를 잠금 해제하는 데 필요한 최소 승인 수를 지정하십시오. 최소값은 1입니다. 최대값은 입니다. 2 생략할 경우 기본값은 입니다. 1

    > **Note**
    >
    > `ApprovalsRequired`속성을 생략하려면 워크플로 정의 파일에서 게이트 `Configuration` 섹션을 제거하십시오.
12. (선택 사항) 커밋하기 전에 워크플로우 YAML 코드를 검증하려면 [Validate] 를 선택합니다.
13. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

워크플로에 '승인' 게이트를 추가하려면 (YAML편집자)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.

3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 다음 예제를 가이드로 사용하여 `Approval` 섹션과 기본 속성을 추가합니다. 자세한 내용은 워크플로우 YAML 정의의 '승인' 게이트 YAML를 참조하십시오.

```
Actions:
  MyApproval_01:
    Identifier: aws/approval@v1
    DependsOn:
      - PreviousAction
    Configuration:
      ApprovalsRequired: 2
```

   다른 예제는 예: '승인' 게이트를 참조하세요.
8. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

승인 알림 구성

워크플로우 실행에 승인이 필요함을 사용자에게 알리는 알림을 Slack 채널로 CodeCatalyst 보낼 수 있습니다. 사용자는 알림을 보고 그 안에 있는 링크를 클릭합니다. 링크를 클릭하면 워크플로를 CodeCatalyst 승인하거나 거부할 수 있는 승인 페이지로 이동합니다.

워크플로가 승인, 거부되었거나 승인 요청이 만료되었음을 사용자에게 알리도록 알림을 구성할 수도 있습니다.

다음 지침에 따라 Slack 알림을 설정하세요.

시작하기 전 준비 사항

워크플로에 승인 게이트를 추가했는지 확인하세요. 자세한 내용은 '승인' 게이트 추가 단원을 참조하십시오.

Slack 채널에 워크플로 승인 알림을 보내려면

1. CodeCatalyst Slack으로 구성하세요. 자세한 내용은 슬랙 알림 시작하기 단원을 참조하십시오.
2. 승인이 필요한 워크플로가 포함된 CodeCatalyst 프로젝트에서 알림이 아직 활성화되지 않은 경우 알림을 활성화하세요. 알림을 활성화하려면:

   a. 프로젝트로 이동한 다음 탐색 창에서 프로젝트 설정을 선택합니다.
   b. 상단에서 알림을 선택합니다.
   c. 알림 이벤트에서 알림 수정을 선택합니다.
   d. 워크플로우 승인 보류를 켜고 알림을 보낼 Slack 채널을 선택합니다. CodeCatalyst
   e. (선택 사항) 승인, 거부 및 만료된 승인에 대해 사람들에게 알리려면 추가 알림을 켜세요. 워크플로 실행 승인, 워크플로 실행 거부, 워크플로 승인 대체 및 워크플로 승인 시간 초과를 켤 수 있습니다. 각 알림 옆에서 알림을 보낼 Slack 채널을 선택합니다. CodeCatalyst
   f. 저장(Save)을 선택합니다.

워크플로 실행 승인 또는 거부

승인 게이트가 포함된 워크플로 실행은 승인하거나 거부해야 합니다. 사용자는 다음과 같이 승인 또는 거부를 제공할 수 있습니다.

- 콘솔 CodeCatalyst
- 팀원이 제공한 링크
- 자동화된 슬랙 알림

사용자가 승인 또는 거부를 제공한 후에는 이 결정을 취소할 수 없습니다.

> **Note**
>
> 특정 사용자만 워크플로 실행을 승인하거나 거부할 수 있습니다. 자세한 내용은 누가 승인을 제공할 수 있나요? 단원을 참조하십시오.

시작하기 전 준비 사항

워크플로에 승인 게이트를 추가했는지 확인하세요. 자세한 내용은 '승인' 게이트 추가 단원을 참조하십시오.

워크플로를 승인 또는 거부하려면 콘솔에서 시작하여 실행하십시오. CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 워크플로 다이어그램에서 승인 게이트를 나타내는 상자를 선택합니다.

   사이드 패널이 나타납니다.

   > **Note**
   >
   > 이때 원하는 경우 이 페이지의 내용을 다른 승인자에게 보낼 수 있습니다. URL

6. 결정 검토에서 승인 또는 거부를 선택합니다.
7. (선택 사항) 코멘트 - 선택 사항에 워크플로우 실행을 승인 또는 거부한 이유를 나타내는 설명을 입력합니다.
8. 제출을 선택합니다.

팀 구성원이 제공한 링크를 사용하여 워크플로 실행을 승인 또는 거부하려면

1. 팀원이 보낸 링크를 선택합니다. (팀원에게 이전 절차를 읽고 링크를 가져오도록 할 수 있습니다.)
2. 로그인하라는 메시지가 CodeCatalyst 표시되면 로그인하세요.

   워크플로 실행 승인 페이지로 리디렉션됩니다.
3. 결정 검토에서 승인 또는 거부를 선택합니다.
4. (선택 사항) 코멘트 - 선택 사항에 워크플로우 실행을 승인 또는 거부한 이유를 나타내는 설명을 입력합니다.
5. 제출을 선택합니다.

자동화된 Slack 알림을 시작으로 워크플로 실행을 승인 또는 거부하려면

1. Slack 알림이 설정되어 있는지 확인하세요. 승인 알림 구성을 참조하세요.
2. Slack의 경우 승인 알림이 전송된 채널의 승인 알림에 있는 링크를 선택합니다.
3. 로그인하라는 메시지가 CodeCatalyst 표시되면 로그인하세요.

워크플로 실행 페이지로 리디렉션됩니다.

4. 워크플로 다이어그램에서 승인 게이트를 선택합니다.
5. 결정 검토에서 승인 또는 거부를 선택합니다.
6. (선택 사항) 코멘트 - 선택 사항에 워크플로우 실행을 승인 또는 거부한 이유를 나타내는 설명을 입력합니다.
7. 제출을 선택합니다.

'승인' 게이트 YAML

승인 게이트의 YAML 정의는 다음과 같습니다. 이 게이트를 사용하는 방법에 대한 자세한 내용은 을 참조하십시오워크플로 실행 시 승인 필요.

이 액션 정의는 광범위한 워크플로 정의 파일 내에 섹션으로 존재합니다. 이 파일에 대한 자세한 내용은 워크플로우 YAML 정의을 참조합니다.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

```
# The workflow definition starts here.
# See ### ## for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The 'Approval' gate definition starts here.
  Approval:
    Identifier: aws/approval@v1
    DependsOn:
      - another-action
    Configuration:
      ApprovalsRequired: number
```

Approval

(필수)

게이트에 부여하려는 이름을 지정합니다. 모든 게이트 이름은 워크플로우 내에서 고유해야 합니다. 게이트 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-) 및 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 게이트 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

기본값: `Approval_nn`.

해당 UI: 구성 탭/ 게이트 이름

Identifier

(##/Identifier)

(필수)

게이트를 식별합니다. 승인 게이트는 버전을 `1.0.0` 지원합니다. 버전 길이를 줄이려는 경우가 아니면 이 속성을 변경하지 마십시오. 자세한 내용은 사용할 액션 버전 지정 단원을 참조하십시오.

기본값: `aws/approval@v1`.

해당 UI: 워크플로 다이어그램/ Approval _n/ aws/approval @v1 라벨

DependsOn

(##/DependsOn)

(선택 사항)

이 게이트를 실행하기 위해 성공적으로 실행되어야 하는 액션, 액션 그룹 또는 게이트를 지정하십시오. 기본적으로 워크플로에 게이트를 추가하면 워크플로의 마지막 작업에 따라 게이트가 설정되도록 설정됩니다. 이 속성을 제거하면 게이트는 아무 것에도 종속되지 않고 다른 작업보다 먼저 실행됩니다.

> **Note**
>
> 게이트는 액션, 액션 그룹 또는 게이트 이전 또는 이후에 실행되도록 구성되어야 합니다. 다른 액션, 액션 그룹 및 게이트와 병렬로 실행되도록 설정할 수 없습니다.

종속 기능에 대한 자세한 내용은 을 참조하십시오시퀀싱 게이트 및 액션.

해당 UI: 입력 탭/에 따라 다름

Configuration

(##/Configuration)

(선택 사항)

게이트의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭

ApprovalsRequired

(##/Configuration/ApprovalsRequired)

(선택 사항)

승인 게이트 잠금을 해제하는 데 필요한 최소 승인 수를 지정합니다. 최소값은 1입니다. 최대값은 입니다. 2 생략할 경우 기본값은 입니다. 1

> **Note**
>
> `ApprovalsRequired`속성을 생략하려면 워크플로 정의 파일에서 게이트 `Configuration` 섹션을 제거하십시오.

해당 UI: 구성 탭/ 승인 횟수

실행의 대기열 동작 구성

CodeCatalystAmazon에서는 기본적으로 여러 워크플로가 동시에 실행되면 시작된 순서대로 워크플로를 CodeCatalyst 대기시키고 하나씩 처리합니다. 실행 모드를 지정하여 이 기본 동작을 변경할 수 있습니다. 몇 가지 실행 모드가 있습니다.

- (기본값) 대기 실행 모드 - CodeCatalyst 프로세스가 하나씩 실행됩니다.
- 대체된 실행 모드 - CodeCatalyst 프로세스가 하나씩 실행되며 새 실행이 이전 실행보다 우선합니다.
- 병렬 실행 모드 — CodeCatalyst 프로세스가 병렬로 실행

워크플로 실행에 대한 자세한 내용은 을 참조하십시오워크플로 실행.

주제

대기 실행 모드에 대한 정보

대기 실행 모드에서는 실행이 연속으로 진행되며 대기 중인 달리기가 대기열을 형성합니다.

대기열은 작업 및 작업 그룹으로 이어지는 진입점에 형성되므로 동일한 워크플로우 내에 여러 대기열을 둘 수 있습니다 (참조). Figure 1 대기열에 있는 실행이 작업에 들어가면 해당 작업이 잠기고 다른 실행은 들어갈 수 없습니다. 실행이 끝나고 액션을 종료하면 액션의 잠금이 해제되어 다음 실행을 위한 준비가 됩니다.

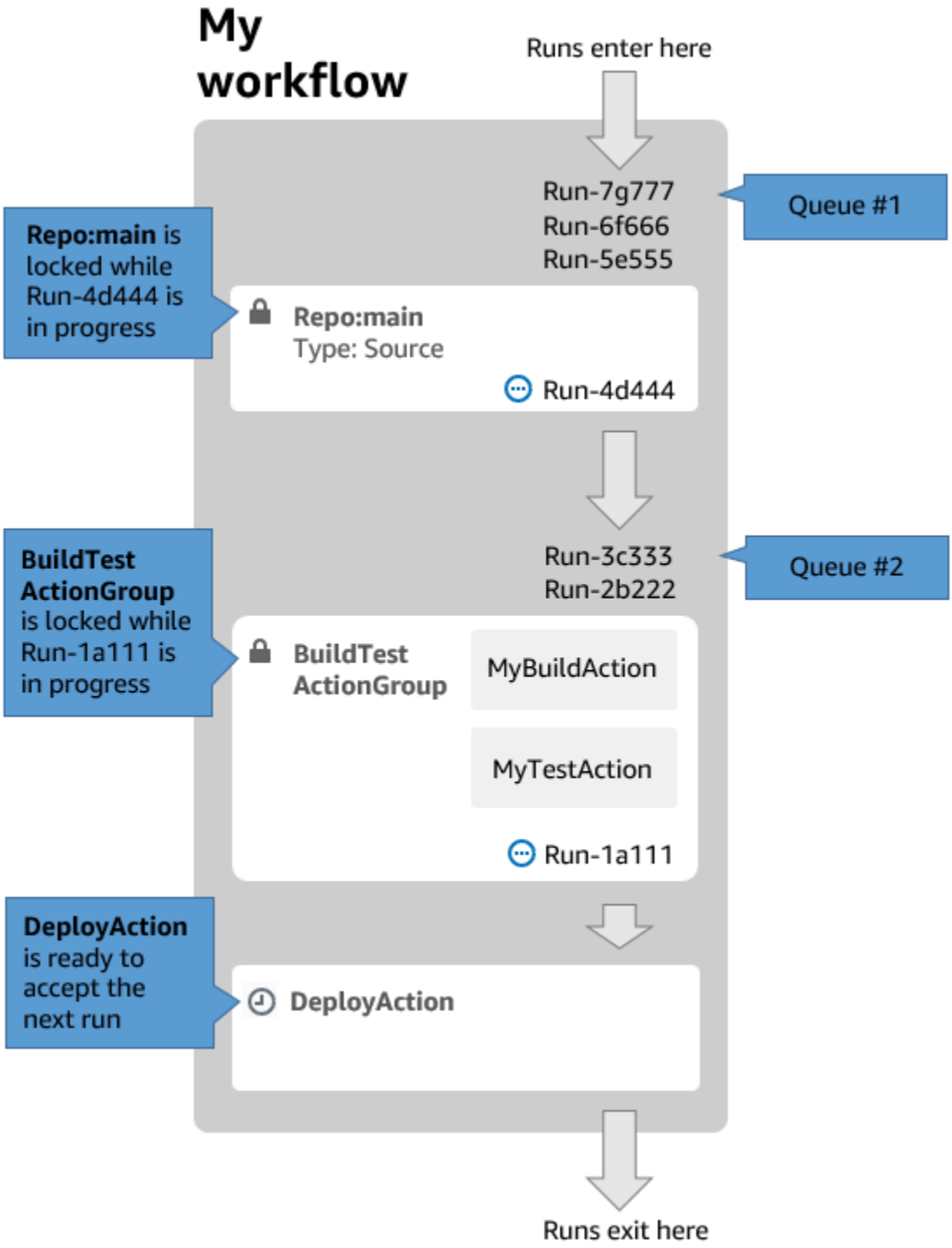
Figure 1대기 실행 모드로 구성된 워크플로를 보여 줍니다. 표시되는 정보는 다음과 같습니다.

- Seven 실행은 워크플로를 통해 순조롭게 진행됩니다.
- 대기열 2개: 하나는 입력 소스 항목 외부 (repo:Main) 이고 다른 하나는 작업 항목 외부에 있습니다. BuildTestActionGroup
- 잠긴 블록 두 개: 입력 소스 (리포지:메인) 와 BuildTestActionGroup

워크플로가 실행되고 처리가 완료될 때 상황이 어떻게 진행되는지는 다음과 같습니다.

- Run-4d444가 소스 리포지토리의 복제를 마치면 입력 소스를 종료하고 Run-3C333 뒤의 대기열에 합류합니다. 그러면 RUN-5E555가 입력 소스로 들어갑니다.
- Run-1A111이 빌드 및 테스트를 마치면 작업을 종료하고 작업을 시작합니다. BuildTestActionGroupDeployAction 그러면 Run-2B222가 액션에 들어갑니다. BuildTestActionGroup

그림 1: '대기 실행 모드'로 구성된 워크플로



다음과 같은 경우 대기 실행 모드를 사용하십시오.

- 기능과 실행 간의 one-to-one 관계를 유지하려는 경우 대체 모드를 사용할 때 이러한 기능을 그룹화할 수 있습니다. 예를 들어 커밋 1에서 기능 1을 병합하면 1이 시작되고 커밋 2에서 기능 2를 병합하면 2가 시작되는 식입니다. 대기열 모드 대신 대체 모드를 사용하면 실행 시 기능 (및 커밋) 이 그룹화되어 다른 기능을 대체합니다.

- 병렬 모드를 사용할 때 발생할 수 있는 경쟁 상황과 예상치 못한 문제를 피하는 것이 좋습니다. 예를 들어 두 소프트웨어 개발자인 Wang과 Saanvi가 거의 동시에 워크플로를 실행하여 Amazon ECS 클러스터에 배포하는 경우 Wang의 실행은 클러스터에서 통합 테스트를 시작하고 Saanvi의 실행은 클러스터에 새 애플리케이션 코드를 배포하여 Wang의 테스트가 실패하거나 잘못된 코드를 테스트할 수 있습니다. 또 다른 예로, 잠금 메커니즘이 없는 대상이 있을 수 있는데, 이 경우 두 실행이 예상치 못한 방식으로 서로의 변경 내용을 덮어쓸 수 있습니다.
- 실행을 처리하는 데 CodeCatalyst 사용되는 컴퓨팅 리소스의 부하를 제한하려고 합니다. 예를 들어 워크플로에 세 개의 작업이 있는 경우 동시에 최대 세 개의 실행이 발생할 수 있습니다. 한 번에 발생할 수 있는 실행 수를 제한하면 실행 처리량을 더 예측할 수 있습니다.
- 워크플로를 통해 타사 서비스에 보내는 요청 수를 제한하고 싶습니다. 예를 들어 워크플로에 Docker Hub에서 이미지를 가져오는 지침이 포함된 빌드 작업이 있을 수 있습니다. Docker Hub는 계정당 시간당 특정 수로 만들 수 있는 풀 요청 수를 제한하며, 한도를 초과할 경우 계정이 잠깁니다. 대기 실행 모드를 사용하여 실행 처리량을 줄이면 시간당 Docker Hub에 생성되는 요청 수가 줄어들어 잠김 및 그로 인한 빌드 및 실행 실패 가능성이 제한됩니다.

최대 대기열 크기: 50

최대 대기열 크기에 대한 참고 사항:

- 최대 대기열 크기는 워크플로의 모든 대기열에서 허용되는 최대 실행 수를 나타냅니다.
- 대기열이 50회 이상 실행되면 51번째 및 그 CodeCatalyst 이후의 실행이 삭제됩니다.

실패 동작:

액션으로 처리되는 동안 실행이 응답하지 않는 경우, 실행 시간이 초과될 때까지 실행 뒤에 있는 실행은 대기열에 보관됩니다. 한 시간이 지나면 액션이 타임아웃됩니다.

작업 내에서 실행이 실패하는 경우 해당 작업 뒤에 대기중인 첫 번째 실행이 계속 진행될 수 있습니다.

대체 실행 모드에 대한 정보

대체 달리기 모드는 다음 점을 제외하면 대기 실행 모드와 동일합니다.

- 대기 중인 실행이 대기열에 있는 다른 실행을 따라잡으면 이후 실행이 이전 실행을 대체 (대체) 하고 이전 실행이 취소되고 '대체됨'으로 표시됩니다.
- 첫 번째 bullet에서 설명한 동작의 결과로 대체된 실행 모드를 사용하는 경우 대기열에 하나의 실행만 포함할 수 있습니다.

의 워크플로를 Figure 1 지침으로 삼아 이 워크플로에 대체 실행 모드를 적용하면 다음과 같은 결과가 발생합니다.

- Run-7g777은 해당 대기열의 다른 두 실행을 대체하며 대기열 #1 에 남아 있는 유일한 실행이 됩니다. Run-6F666 및 Run-5E555는 취소됩니다.
- Run-3C333은 Run-2B222를 대체하며 대기열 #2 내에 남아 있는 유일한 실행입니다. Run-2B222는 취소될 것입니다.

원하는 경우 대체된 실행 모드를 사용하세요.

- 대기열 모드보다 처리량이 더 좋습니다.
- 대기 모드보다 타사 서비스에 대한 요청 수가 훨씬 적습니다. 이는 타사 서비스에 Docker Hub와 같은 속도 제한이 있는 경우에 유리합니다.

병렬 실행 모드에 대한 정보

병렬 실행 모드에서는 실행이 서로 독립적이므로 시작하기 전에 다른 실행이 완료될 때까지 기다리지 않아도 됩니다. 대기열이 없으며 워크플로우 내의 작업이 완료되는 데 걸리는 속도에 의해서만 실행 처리량이 제한됩니다.

각 사용자가 고유한 기능 브랜치를 갖고 다른 사용자와 공유하지 않는 대상에 배포하는 개발 환경에서 병렬 실행 모드를 사용하십시오.

> **Important**
>
> 프로덕션 환경의 Lambda 함수와 같이 여러 사용자가 배포할 수 있는 공유 대상이 있는 경우 경쟁 조건이 발생할 수 있으므로 병렬 모드를 사용하지 마십시오. Parallel 워크플로우 실행에서 공유 리소스를 동시에 변경하려고 시도하면 경쟁 상태가 발생하여 예기치 않은 결과가 발생합니다.

최대 병렬 실행 수: CodeCatalyst 공간당 1000개

실행 모드 구성

실행 모드를 대기열, 대체 또는 병렬로 설정할 수 있습니다. 기본값은 대기열입니다.

실행 모드를 대기열에 추가됨 또는 대체됨에서 CodeCatalyst 병렬로 변경하면 대기열에 있는 실행은 취소되며, 작업에 의해 현재 처리 중인 실행은 취소하기 전에 완료되도록 합니다.

실행 모드를 병렬에서 대기 또는 대체됨으로 변경하면 현재 실행 중인 CodeCatalyst 모든 병렬 실행이 완료됩니다. 실행 모드를 대기 또는 대체 모드로 변경한 후 시작하는 모든 실행은 새 모드를 사용합니다.

Visual

비주얼 에디터를 사용하여 실행 모드를 변경하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 오른쪽 상단에서 워크플로 속성을 선택합니다.
7. 고급을 확장하고 실행 모드에서 다음 중 하나를 선택합니다.
   a. 대기 중 — 참조 대기 실행 모드에 대한 정보
   b. 대체됨 — 참조 대체 실행 모드에 대한 정보
   c. 병렬 — 참조 병렬 실행 모드에 대한 정보
8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드를 검증하십시오.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

YAML편집기를 사용하여 실행 모드를 변경하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.

7. 다음과 같이 RunMode 속성을 추가합니다.

```
Name: Workflow_6d39
SchemaVersion: "1.0"
RunMode: QUEUED|SUPERSEDED|PARALLEL
```

자세한 내용은 최상위 속성 섹션의 RunMode 속성 설명을 참조하십시오워크플로우 YAML 정의.

8. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로의 YAML 코드를 검증하십시오.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

워크플로우 실행 간 파일 캐싱

파일 캐싱이 활성화되면 빌드 및 테스트 작업에서 디스크에 있는 파일을 캐시에 저장하고 후속 워크플로 실행 시 해당 캐시에서 파일을 복원합니다. 캐싱은 실행 사이에 변경되지 않은 종속성을 빌드하거나 다운로드할 때 발생하는 지연 시간을 줄여줍니다. CodeCatalyst 또한 일부 필수 종속성이 포함된 부분 캐시를 복원하는 데 사용할 수 있는 대체 캐시를 지원합니다. 이렇게 하면 캐시 누락이 지연 시간에 미치는 영향을 줄이는 데 도움이 됩니다.

> **Note**
>
> 파일 캐싱은 Amazon CodeCatalyst 빌드 및 테스트 작업에서만 사용할 수 있으며 EC2컴퓨팅 유형을 사용하도록 구성된 경우에만 사용할 수 있습니다.

주제

- 파일 캐싱에 대한 정보
- 캐시 생성
- 파일 캐싱 제약 조건

파일 캐싱에 대한 정보

파일 캐싱을 사용하면 데이터를 여러 캐시로 구성할 수 있으며, 각 캐시는 속성 아래에서 참조됩니다. FileCaching 각 캐시는 지정된 경로로 지정된 디렉토리를 저장합니다. 지정된 디렉토리는 향후 워크플로우 실행 시 복원됩니다. 다음은 이름이 및 인 여러 캐시를 사용하는 캐싱의 예제 YAML 스니펫입니다. cacheKey1 cacheKey2

```
Actions:
  BuildMyNpmApp:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: npm install
        - Run: npm run test
    Caching:
      FileCaching:
        cacheKey1:
          Path: file1.txt
          RestoreKeys:
             - restoreKey1
        cacheKey2:
          Path: /root/repository
          RestoreKeys:
             - restoreKey2
             - restoreKey3
```

> **Note**
>
> CodeCatalyst 로컬 캐시와 원격 캐시로 구성된 다중 계층 캐싱을 사용합니다. 프로비저닝된 플릿 또는 온디맨드 시스템에서 로컬 캐시에서 캐시 누락이 발생하는 경우 원격 캐시에서 종속성이 복원됩니다. 따라서 일부 작업 실행에서는 원격 캐시 다운로드로 인해 지연이 발생할 수 있습니다.

CodeCatalyst 캐시 액세스 제한을 적용하여 한 워크플로의 작업으로 다른 워크플로의 캐시를 수정할 수 없도록 합니다. 이렇게 하면 빌드나 배포에 영향을 주는 잘못된 데이터를 푸시할 수 있는 다른 워크플로로부터 각 워크플로를 보호할 수 있습니다. 모든 워크플로 및 브랜치 페어링에 캐시를 격리하는 캐시 범위에는 제한이 적용됩니다. 예를 들어 `workflow-A` in 브랜치의 파일 캐시는 형제 브랜치의 파일 `feature-A` 캐시와 다릅니다. `workflow-A` `feature-B`

워크플로에서 지정된 파일 캐시를 찾았지만 찾을 수 없는 경우 캐시 누락이 발생합니다. 이는 새 브랜치가 생성되거나 새 캐시가 참조되었는데 아직 생성되지 않은 경우 등 여러 가지 이유로 발생할 수 있습니다. 캐시가 만료되는 경우에도 발생할 수 있으며, 이 만료는 기본적으로 마지막으로 사용된 지 14일 후에 발생합니다. 캐시 누락을 줄이고 캐시 적중률을 높이기 위해 폴백 캐시를 CodeCatalyst 지원합

니다. 대체 캐시는 대체 캐시이며 이전 버전의 캐시일 수 있는 부분 캐시를 복원할 수 있는 기회를 제공합니다. 먼저 속성 이름과 일치하는 항목을 검색하여 캐시를 복원하고, 찾을 수 없는 `FileCaching` 경우 평가합니다. `RestoreKeys` 속성 이름과 모든 항목에 캐시 누락이 있는 경우`RestoreKeys`, 캐싱이 최선의 방법이며 보장되지 않기 때문에 워크플로는 계속 실행됩니다.

캐시 생성

다음 지침에 따라 워크플로에 캐시를 추가할 수 있습니다.

Visual

비주얼 편집기를 사용하여 캐시를 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로 다이어그램에서 캐시를 추가할 작업을 선택합니다.
8. Configuration(구성)을 선택합니다.
9. 파일 캐싱 - 선택 사항에서 캐시 추가를 선택하고 다음과 같이 필드에 정보를 입력합니다.

   Key(키)

   기본 캐시 속성 이름을 지정합니다. 캐시 속성 이름은 워크플로우 내에서 고유해야 합니다. 각 작업에는 최대 5개의 항목이 포함될 수 `FileCaching` 있습니다.

   경로

   캐시의 관련 경로를 지정합니다.

   복원 키 - 선택 사항

   기본 캐시 속성을 찾을 수 없을 때 폴백으로 사용할 복원 키를 지정합니다. 복원 키 이름은 워크플로우 내에서 고유해야 합니다. 각 캐시에는 최대 5개의 항목을 포함할 수 `RestoreKeys` 있습니다.

10. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
11. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

YAML편집기를 사용하여 캐시를 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 워크플로 작업에 다음과 비슷한 코드를 추가합니다.

```
action-name:
  Configuration:
    Steps: ...
  Caching:
    FileCaching:
      key-name:
        Path: file-path
        # # Specify any additional fallback caches
        # RestoreKeys:
        #   - restore-key
```

8. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

파일 캐싱 제약 조건

속성 이름 및 RestoreKeys 속성에 대한 제약 조건은 다음과 같습니다.

- 이름은 워크플로우 내에서 고유해야 합니다.

- 이름은 영숫자 (A-Z, a-z, 0-9), 하이픈 (-) 및 밑줄 (_) 로 제한됩니다.
- 이름은 최대 180자까지 입력할 수 있습니다.
- 각 액션에는 최대 5개의 캐시가 있을 수 있습니다. `FileCaching`
- 각 캐시는 최대 5개의 항목을 포함할 수 있습니다. `RestoreKeys`

경로에 대한 제약 조건은 다음과 같습니다.

- 별표 (*) 는 허용되지 않습니다.
- 경로는 최대 255자까지 입력할 수 있습니다.

워크플로 실행 상태 및 세부 정보 보기

CodeCatalystAmazon에서는 단일 워크플로 실행 또는 여러 실행의 상태 및 세부 정보를 동시에 볼 수 있습니다.

가능한 실행 상태 목록은 을 참조하십시오워크플로 실행 상태.

> **Note**
>
> 워크플로우 실행 상태와는 다른 워크플로우 상태도 볼 수 있습니다. 자세한 내용은 워크플로우 상태 보기 단원을 참조하십시오.

워크플로 실행에 대한 자세한 내용은 을 참조하십시오워크플로 실행.

주제

- 단일 실행의 상태 및 세부 정보 보기
- 프로젝트 내 모든 실행의 상태 및 세부 정보 보기
- 특정 워크플로의 모든 실행 상태 및 세부 정보 보기
- 워크플로 다이어그램에서 워크플로의 실행 보기

단일 실행의 상태 및 세부 정보 보기

단일 워크플로 실행의 상태 및 세부 정보를 보고 성공 여부를 확인하거나, 언제 완료되었는지 확인하거나, 누가 또는 누가 시작했는지 확인하는 것이 좋습니다.

단일 실행의 상태 및 세부 정보를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 워크플로 이름 아래에서 실행을 선택합니다.
6. 실행 기록의 실행 ID 열에서 실행을 선택합니다. 예: Run-95a4d.
7. 실행 이름 아래에서 다음 중 하나를 수행하십시오.

   - 워크플로 실행의 동작과 상태를 보여주는 워크플로 다이어그램을 시각적으로 볼 수 있습니다(참조워크플로 실행 상태). 이 뷰에는 실행 중에 사용된 소스 리포지토리와 브랜치도 표시됩니다.

     워크플로 다이어그램에서 작업을 선택하면 실행 중에 해당 작업에 의해 생성된 로그, 보고서, 출력 등의 세부 정보를 볼 수 있습니다. 표시되는 정보는 선택한 작업 유형에 따라 달라집니다. 빌드 또는 배포 로그를 보는 방법에 대한 자세한 내용은 빌드 작업 결과 보기 또는 을 참조하십시오배포 로그 보기.
   - YAML실행에 사용된 워크플로 정의 파일을 보려면
   - 워크플로우 실행으로 생성된 아티팩트를 확인할 수 있는 아티팩트입니다. 아티팩트에 대한 자세한 내용은 작업 간 아티팩트 및 파일 공유 단원을 참조하십시오.
   - 워크플로우에서 생성된 테스트 보고서 및 기타 유형의 보고서를 볼 수 있는 보고서가 실행됩니다. 보고서에 대한 자세한 내용은 품질 보고서 유형의 내용을 참조하세요.
   - 워크플로 실행에서 생성된 출력 변수를 확인하기 위한 변수입니다. 변수에 대한 자세한 내용은을 참조하십시오워크플로우에서 변수 사용.

> **Note**
>
> 실행의 상위 워크플로가 삭제된 경우 해당 사실을 나타내는 메시지가 실행 세부 정보 페이지 상단에 나타납니다.

프로젝트 내 모든 실행의 상태 및 세부 정보 보기

프로젝트 내 모든 워크플로 실행의 상태와 세부 정보를 보고, 프로젝트에서 진행 중인 워크플로 활동의 양을 파악하고, 워크플로의 전반적인 상태를 파악하는 것이 좋습니다.

프로젝트 내 모든 실행의 상태 및 세부 정보를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로우에서 실행을 선택합니다.

   모든 브랜치, 프로젝트 내 모든 리포지토리의 모든 워크플로에 대한 모든 실행이 표시됩니다.

   페이지에는 다음과 같은 열이 있습니다.

   - 실행 ID — 실행의 고유 식별자입니다. 런에 대한 자세한 정보를 보려면 런 ID 링크를 선택하세요.
   - 상태 - 워크플로우 실행의 처리 상태입니다. 실행 상태에 대한 자세한 내용은 을 참조하십시오워크플로 실행 상태.
   - 트리거 — 워크플로 실행을 시작한 사람, 커밋, 풀 리퀘스트 (PR) 또는 일정입니다. 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 단원을 참조하십시오.
   - 워크플로 — 실행이 시작된 워크플로의 이름, 워크플로 정의 파일이 있는 소스 리포지토리 및 브랜치의 이름입니다. 이 정보를 보려면 열 너비를 확장해야 할 수도 있습니다.

     > **Note**
     >
     > 이 열을 사용할 수 없음으로 설정된 경우 일반적으로 관련 워크플로가 삭제되거나 이동되었기 때문입니다.

   - 시작 시간 - 워크플로 실행이 시작된 시간입니다.
   - 기간 - 워크플로를 실행하여 처리하는 데 걸린 시간입니다. 지속 시간이 너무 길거나 짧으면 문제가 발생할 수 있습니다.
   - 종료 시간 - 워크플로우 실행이 종료된 시간입니다.

특정 워크플로의 모든 실행 상태 및 세부 정보 보기

특정 워크플로와 관련된 모든 실행의 상태 및 세부 정보를 보고 워크플로 내에서 병목 현상을 일으키는 실행이 있는지 확인하거나 현재 진행 중이거나 완료된 실행을 확인할 수 있습니다.

특정 워크플로의 모든 실행 상태 및 세부 정보를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 워크플로 이름 아래에서 실행을 선택합니다.

   선택한 워크플로와 관련된 실행이 표시됩니다.

   페이지는 두 섹션으로 구분됩니다.

   - 활성 실행 — 진행 중인 실행을 표시합니다. 이러한 실행은 다음 상태 중 하나일 것입니다: 진행 중.
   - 실행 기록 - 완료된 (즉, 진행 중이 아닌) 실행이 표시됩니다.

     실행 상태에 대한 자세한 내용은 을 참조하십시오워크플로 실행 상태.

워크플로 다이어그램에서 워크플로의 실행 보기

워크플로를 통해 함께 진행되는 워크플로의 모든 실행 상태를 볼 수 있습니다. 실행은 목록 보기가 아닌 워크플로 다이어그램에 표시됩니다. 이를 통해 어떤 실행이 어떤 작업에 의해 처리되고 있는지, 어떤 실행이 대기열에 대기 중인지 시각적으로 확인할 수 있습니다.

워크플로를 통해 함께 진행되는 여러 실행의 상태를 보려면

> **Note**
>
> 이 절차는 워크플로가 대기 또는 대체 실행 모드를 사용하는 경우에만 적용됩니다. 자세한 내용은 실행의 대기열 동작 구성 단원을 참조하십시오.

1. https://codecatalyst.aws/ 에서 콘솔을 엽니다. CodeCatalyst

2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.

> **Note**
>
> 실행 페이지가 아니라 워크플로 페이지를 보고 있는지 확인하세요.

5. 왼쪽 상단에서 최신 상태 탭을 선택합니다.

   워크플로 다이어그램이 나타납니다.
6. 워크플로 다이어그램을 검토하십시오. 다이어그램에는 워크플로우 내에서 현재 진행 중인 모든 실행과 완료된 최신 실행이 표시됩니다. 좀 더 구체적으로 설명하자면:

   - Sources 앞에 맨 위에 나타나는 실행은 대기열에 추가되어 시작을 기다리고 있습니다.
   - 액션 사이에 나타나는 런은 대기열에 추가되어 다음 액션에 의해 처리되기를 기다립니다.
   - 작업 내에 나타나는 실행은 1. 현재 작업에 의해 처리 중이거나, 2. 작업에 의해 처리가 완료되었거나, 3. 작업에 의해 처리되지 않은 경우 (일반적으로 이전 작업이 실패했기 때문) 입니다.

워크플로우 작업 구성

작업은 워크플로의 기본 구성 요소이며, 워크플로 실행 중에 수행할 논리적 작업 단위 또는 작업을 정의합니다. 일반적으로 워크플로에는 구성 방법에 따라 순차적으로 또는 병렬로 실행되는 여러 작업이 포함됩니다.

주제

- 사용 가능한 액션 버전 목록
- 액션의 소스 코드 보기
- 액션과 GitHub 통합

작업 유형

Amazon CodeCatalyst 워크플로 내에서 다음과 같은 유형의 작업을 사용할 수 있습니다.

작업 유형

- CodeCatalyst 조치
- CodeCatalyst 랩 작업
- GitHub 액션
- 타사 작업

CodeCatalyst 조치

CodeCatalyst 액션은 CodeCatalyst 개발팀에서 작성, 유지 관리 및 전적으로 지원하는 액션입니다.

응용 프로그램을 빌드, 테스트 및 배포하는 CodeCatalyst 작업은 물론 함수 호출과 같은 기타 작업을 수행하는 작업도 있습니다. AWS Lambda

다음과 같은 작업을 수행할 수 있습니다. CodeCatalyst

- 빌드

  이 액션은 아티팩트를 빌드하고 Docker 컨테이너에서 유닛 테스트를 실행합니다. 자세한 내용은 빌드 액션 추가 단원을 참조하십시오.
- 테스트

  이 작업은 애플리케이션 또는 아티팩트에 대한 통합 및 시스템 테스트를 실행합니다. 자세한 내용은 테스트 액션 추가 단원을 참조하십시오.
- 아마존 S3 퍼블리시

  이 작업은 애플리케이션 아티팩트를 Amazon S3 버킷에 복사합니다. 자세한 내용은 워크플로를 사용하여 Amazon S3에 파일 게시 단원을 참조하십시오.
- AWS CDK 부트스트랩

이 작업은 CDK 앱을 배포하는 데 AWS CDK 필요한 리소스를 프로비저닝합니다. 자세한 내용은 워크플로를 사용하여 AWS CDK 앱 부트스트래핑하기 단원을 참조하십시오.

- AWS CDK 배포

  이 작업은 앱을 합성하고 배포합니다. AWS Cloud Development Kit (AWS CDK) 자세한 내용은 워크플로를 통한 AWS CDK 앱 배포 단원을 참조하십시오.

- AWS Lambda 호출

  이 액션은 함수를 호출합니다. AWS Lambda 자세한 내용은 워크플로를 사용하여 Lambda 함수 호출 단원을 참조하십시오.

- GitHub 액션

  이 작업은 CodeCatalyst 워크플로우 내에서 GitHub 작업을 실행할 수 있는 작업입니다. CodeCatalyst 자세한 내용은 워크플로를 사용하여 Lambda 함수 호출 단원을 참조하십시오.

- AWS CloudFormation 스택 배포

  이 액션은 AWS CloudFormation 스택을 배포합니다. 자세한 내용은 스택 배포 AWS CloudFormation 단원을 참조하십시오.

- 아마존에 배포 ECS

  이 작업은 Amazon ECS 작업 정의를 등록하고 Amazon 서비스에 배포합니다. ECS 자세한 내용은 워크플로를 ECS 사용하여 Amazon에 배포하기 단원을 참조하십시오.

- 쿠버네티스 클러스터에 배포

  이 작업은 애플리케이션을 Kubernetes 클러스터에 배포합니다. 자세한 내용은 워크플로를 EKS 사용하여 Amazon에 배포하기 단원을 참조하십시오.

- 렌더 아마존 ECS 태스크 정의

  이 작업은 컨테이너 이미지를 URI Amazon ECS 작업 정의 JSON 파일에 삽입하여 새 작업 정의 파일을 생성합니다. 자세한 내용은 Amazon ECS 작업 정의 수정 단원을 참조하십시오.

CodeCatalyst 작업에 대한 설명서는 이 가이드와 각 작업의 추가 정보에서 확인할 수 있습니다.

사용 가능한 CodeCatalyst 작업 및 워크플로에 작업을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오워크플로에 작업 추가.

CodeCatalyst 랩 작업

CodeCatalyst 랩 액션은 실험적 애플리케이션을 위한 시험장인 Amazon CodeCatalyst Labs의 일부인 액션입니다. CodeCatalyst Labs 액션은 서비스와의 AWS 통합을 보여주기 위해 개발되었습니다.

다음과 같은 CodeCatalyst 랩 작업을 사용할 수 있습니다.

- AWS Amplify 호스팅에 배포

  이 작업은 Amplify 호스팅에 응용 프로그램을 배포합니다.
- 에 배포하십시오. AWS App Runner

  이 작업은 소스 이미지 저장소의 최신 이미지를 App Runner에 배포합니다.
- 아마존 CloudFront 및 아마존 S3에 배포

  이 작업은 Amazon S3에 애플리케이션을 CloudFront 배포합니다.
- 를 사용하여 배포하십시오. AWS SAM

  이 작업은 AWS Serverless Application Model ()AWS SAM를 사용하여 서버리스 애플리케이션을 배포합니다.
- 아마존 CloudFront 캐시 무효화

  이 작업을 수행하면 지정된 경로 세트의 CloudFront 캐시가 무효화됩니다.
- 발신 웹후크

  이 작업을 통해 사용자는 요청을 사용하여 워크플로우 내의 메시지를 임의의 웹 서버로 보낼 수 있습니다. HTTPS
- 게시 대상 AWS CodeArtifact

  이 작업은 패키지를 CodeArtifact 저장소에 게시합니다.
- 아마존에 게시 SNS

  이 작업을 통해 사용자는 주제를 생성하거나, 주제를 게시하거나, 주제를 SNS 구독하여 Amazon과 통합할 수 있습니다.
- 아마존으로 푸시 ECR

  이 작업은 Docker 이미지를 빌드하여 Amazon Elastic 컨테이너 레지스트리 ECR (Amazon) 리포지토리에 게시합니다.
- 아마존 CodeGuru 시큐리티로 스캔하기

이 작업은 구성된 코드 경로의 zip 아카이브를 생성하고 CodeGuru 보안을 사용하여 코드 스캔을 실행합니다.

- 테라폼 커뮤니티 에디션

  이 액션은 Terraform 커뮤니티 에디션 및 운영을 실행합니다. `plan` `apply`

CodeCatalyst 랩 작업에 대한 문서는 각 작업의 추가 정보에서 확인할 수 있습니다.

워크플로에 CodeCatalyst Labs 작업을 추가하고 해당 Readme를 보는 방법에 대한 자세한 내용은 을 참조하십시오. 워크플로에 작업 추가

GitHub 액션

GitHub 액션은 GitHub 워크플로우와 함께 사용하도록 개발되었다는 점을 제외하면 CodeCatalyst 액션과 매우 비슷합니다. GitHub 액션에 대한 자세한 내용은 GitHub 액션 설명서를 참조하십시오.

CodeCatalyst 워크플로우에서 네이티브 GitHub 액션과 함께 CodeCatalyst 액션을 사용할 수 있습니다.

편의를 위해 CodeCatalyst 콘솔에서는 몇 가지 인기 있는 GitHub 액션에 액세스할 수 있습니다. GitHub Marketplace에 나열된 모든 GitHub 작업을 사용할 수도 있습니다 (몇 가지 제한 사항 적용).

액션 설명서는 각 GitHub 액션의 readme에서 확인할 수 있습니다.

자세한 내용은 액션과 GitHub 통합 단원을 참조하십시오.

타사 작업

타사 작업은 타사 공급업체가 작성하여 콘솔에서 사용할 수 있는 작업입니다. CodeCatalyst 타사 작업의 예로는 Mend와 Sonar가 각각 작성한 Mend SCA 및 SonarCloud Scan 작업이 있습니다.

타사 작업에 대한 설명서는 각 작업의 추가 정보에서 확인할 수 있습니다. 타사 공급업체에서 추가 설명서를 제공할 수도 있습니다.

워크플로에 타사 작업을 추가하고 해당 Readme를 보는 방법에 대한 자세한 내용은 을 참조하십시오워크플로에 작업 추가.

워크플로에 작업 추가

다음 지침에 따라 워크플로에 작업을 추가한 다음 구성하십시오.

작업 추가 및 구성하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 왼쪽 상단에서 + Actions를 선택하면 Actions 카탈로그가 나타납니다.
7. 드롭다운 목록에서 다음 중 하나를 수행하십시오.
   - Amazon CodeCatalyst, CodeCatalyst Labs 또는 타사 작업을 CodeCatalyst 보려면 Amazon을 선택하십시오.
     - CodeCatalyst 액션에는 바이 AWS 라벨이 있습니다.
     - CodeCatalyst 랩 작업에는 By CodeCatalyst Labs 라벨이 있습니다.
     - 타사 작업에는 사용자 지정이 있습니다.***vendor***라벨, 위치 *vendor* 타사 공급업체의 이름입니다.
   - 선별된 GitHub 작업 목록을 GitHub보려면 선택하십시오.
8. 액션 카탈로그에서 액션을 검색한 후 다음 중 하나를 수행하십시오.
   - 더하기 기호 (+) 를 선택하여 워크플로에 동작을 추가합니다.
   - 액션의 이름을 선택하면 해당 액션의 추가 정보를 볼 수 있습니다.
9. 액션을 구성합니다. 비주얼 편집기를 사용하거나 편집기를 YAML사용하려면 Visual을 YAML 선택합니다. 자세한 지침은 다음 링크를 참조하십시오.

   CodeCatalyst작업 추가에 대한 지침은 다음을 참조하십시오.

   - 빌드 액션 추가
   - 테스트 액션 추가
   - '아마존에 ECS 배포' 작업 추가
   - '쿠버네티스 클러스터에 배포' 작업 추가
   - AWS CloudFormation '스택 배포' 작업 추가
   - AWS CDK '배포' 작업 추가
   - AWS CDK '부트스트랩' 액션 추가

- '아마존 S3 퍼블리시' 작업 추가
- AWS Lambda '호출' 작업 추가
- 'Amazon ECS 작업 정의 렌더링' 작업 추가

CodeCatalyst Labs 작업 추가에 대한 지침은 다음을 참조하십시오.

- 액션의 추가 정보입니다. 작업 카탈로그에서 작업 이름을 선택하면 추가 정보를 찾을 수 있습니다.

GitHub 액션 추가에 대한 지침은 다음을 참조하십시오.

- 액션과 GitHub 통합

타사 작업 추가에 대한 지침은 다음을 참조하십시오.

- 해당 액션의 추가 정보입니다. 작업 카탈로그에서 작업 이름을 선택하면 추가 정보를 찾을 수 있습니다.

10. (선택 사항) Validate를 선택하여 YAML 코드가 유효한지 확인합니다.
11. 커밋을 선택하여 변경 내용을 적용합니다.

워크플로에서 작업 제거

워크플로에서 동작을 제거하려면 다음 지침을 따르십시오.

Visual

시각 편집기를 사용하여 동작을 제거하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.

7. 워크플로 다이어그램의 제거하려는 작업에서 세로 줄임표 아이콘을 선택하고 제거를 선택합니다.
8. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로우 YAML 코드를 검증합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

YAML편집기를 사용하여 액션을 제거하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 제거하려는 동작이 YAML 포함된 섹션을 찾으십시오.

   섹션을 선택하고 키보드의 삭제 키를 누릅니다.
8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드를 검증합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

사용자 지정 작업 개발

CodeCatalyst 액션 개발 키트 (ADK) 를 사용하여 워크플로에 사용할 사용자 지정 액션을 개발할 수 있습니다. 그런 다음 액션을 CodeCatalyst 액션 카탈로그에 게시하여 다른 CodeCatalyst 사용자가 워크플로에서 보고 사용할 수 있도록 할 수 있습니다.

작업 개발, 테스트 및 게시 (상위 수준 작업)

1. 작업을 개발하는 데 필요한 필수 도구 및 패키지를 설치합니다.
2. 액션 코드를 저장할 CodeCatalyst 저장소를 만드세요.
3. 액션을 초기화하세요. 이렇게 하면 자체 코드로 업데이트할 수 있는 작업 정의 파일 (`action.yml`) 을 포함하여 작업에 필요한 소스 파일이 정리됩니다.

4. 액션 코드를 부트스트랩하여 액션 프로젝트를 빌드, 테스트 및 릴리스하는 데 필요한 도구와 라이브러리를 확보하세요.
5. 로컬 컴퓨터에서 액션을 빌드하고 변경 내용을 CodeCatalyst 저장소에 푸시하세요.
6. 로컬에서 단위 테스트로 작업을 테스트하고 에서 ADK CodeCatalyst 생성된 워크플로를 실행하십시오.
7. CodeCatalyst 콘솔에서 Publish 버튼을 선택하여 CodeCatalyst 액션을 액션 카탈로그에 게시합니다.

자세한 단계는 Amazon CodeCatalyst Action 개발 키트 개발자 안내서를 참조하십시오.

작업을 작업 그룹으로 그룹화

작업 그룹에는 하나 이상의 작업이 포함됩니다. 작업을 작업 그룹으로 그룹화하면 워크플로를 체계적으로 구성하고 여러 그룹 간의 종속성을 구성할 수 있습니다.

> **Note**
>
> 다른 작업 그룹이나 작업 내에 작업 그룹을 중첩할 수 없습니다.

주제

액션 그룹 정의

다음 지침을 사용하여 CodeCatalyst 작업 그룹을 정의하십시오.

Visual

사용할 수 없습니다. YAML지침을 YAML 보려면 선택하십시오.

YAML

그룹 정의

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.

3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. Actions에서 다음과 비슷한 코드를 추가합니다.

```
Actions:
  action-group-name:
    Actions:
      action-1:
        Identifier: aws/build@v1
        Configuration:
          ...
      action-2:
        Identifier: aws/build@v1
        Configuration:
          ...
```

   다른 예제는 예: 두 개의 액션 그룹 정의를 참조하세요. 자세한 내용은 작업 의 action-group-name 속성 설명을 참조하십시오워크플로우 YAML 정의.
8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

예: 두 개의 액션 그룹 정의

다음 예는 두 개의 Amazon CodeCatalyst 작업 그룹인 BuildAndTest 및 을 정의하는 방법을 보여줍니다Deploy. BuildAndTest그룹에는 두 개의 작업 (Build및Test) 이 포함되고 Deploy 그룹에는 두 개의 작업 (DeployCloudFormationStack및DeployToECS) 도 포함됩니다.

```
Actions:
  BuildAndTest: # Action group 1
    Actions:
      Build:
        Identifier: aws/build@v1
        Configuration:
          ...
```

```
      Test:
        Identifier: aws/managed-test@v1
        Configuration:
  Deploy: #Action group 2
    Actions:
      DeployCloudFormationStack:
        Identifier: aws/cfn-deploy@v1
        Configuration:
          ...
      DeployToECS:
        Identifier: aws/ecs-deploy@v1
        Configuration:
          ...
```

시퀀싱 액션

기본적으로 워크플로에 작업을 추가하면 비주얼 편집기에 작업이 나란히 추가됩니다. 즉, 워크플로 실행을 시작하면 작업이 병렬로 실행됩니다. 작업을 순차적으로 실행하고 시각적 편집기에 세로로 표시되도록 하려면 작업 간에 종속성을 설정해야 합니다. 예를 들어 빌드 작업 후에 테스트 `Test` 작업이 실행되도록 작업에 종속되도록 `Build` 작업을 설정할 수 있습니다.

액션과 액션 그룹 간의 종속성을 설정할 수 있습니다. 또한 시작 시 하나의 작업이 다른 여러 작업에 one-to-many 종속되도록 종속성을 구성할 수 있습니다. 의 지침을 액션 간 종속성 설정 참조하여 종속성 설정이 워크플로의 구문을 준수하는지 확인하십시오. YAML

주제

- 작업 간 종속성을 구성하는 방법의 예
- 액션 간 종속성 설정

작업 간 종속성을 구성하는 방법의 예

다음 예제는 워크플로 정의 파일에서 작업과 그룹 간의 종속성을 구성하는 방법을 보여줍니다.

주제

- 예: 단순 종속성 구성
- 예: 작업에 종속되도록 작업 그룹 구성
- 예: 다른 작업 그룹에 종속되도록 작업 그룹 구성
- 예: 여러 작업에 종속되도록 작업 그룹 구성

예: 단순 종속성 구성

다음 예제는 DependsOn 속성을 사용하여 작업에 종속되도록 Test Build 작업을 구성하는 방법을 보여줍니다.

```
Actions:
  Build:
    Identifier: aws/build@v1
    Configuration:
      ...
  Test:
    DependsOn:
      - Build
    Identifier: aws/managed-test@v1
     Configuration:
       ...
```

예: 작업에 종속되도록 작업 그룹 구성

다음 예제는 작업에 종속되도록 DeployGroup 작업 그룹을 구성하는 방법을 보여줍니다. FirstAction 액션 그룹과 액션 그룹은 같은 레벨에 있다는 것을 알 수 있습니다.

```
Actions:
  FirstAction: #An action outside an action group
    Identifier: aws/github-actions-runner@v1
    Configuration:
      ...
  DeployGroup: #An action group containing two actions
    DependsOn:
      - FirstAction
    Actions:
      DeployAction1:
      ...
      DeployAction2:
      ...
```

예: 다른 작업 그룹에 종속되도록 작업 그룹 구성

다음 예제는 DeployGroup 작업 그룹에 종속되도록 작업 그룹을 구성하는 방법을 보여줍니다. BuildAndTestGroup 액션 그룹이 같은 레벨에 있다는 것을 알 수 있습니다.

```
Actions:
```

```
  BuildAndTestGroup: # Action group 1
    Actions:
      BuildAction:
      ...
      TestAction:
      ...
  DeployGroup: #Action group 2
    DependsOn:
      - BuildAndTestGroup
    Actions:
      DeployAction1:
      ...
      DeployAction2:
      ...
```

예: 여러 작업에 종속되도록 작업 그룹 구성

다음 예에서는 작업, DeployGroup 작업, FirstAction 작업 그룹에 따라 SecondAction BuildAndTestGroup 작업 그룹을 구성하는 방법을 보여 줍니다. DeployGroup이 레벨 은FirstAction,SecondAction, 와 같다는 것을 알 수 BuildAndTestGroup 있습니다.

```
Actions:
  FirstAction: #An action outside an action group
    ...
  SecondAction: #Another action
    ...
  BuildAndTestGroup: #Action group 1
    Actions:
      Build:
      ...
      Test:
      ...
  DeployGroup: #Action group 2
    DependsOn:
      - FirstAction
      - SecondAction
      - BuildAndTestGroup
    Actions:
      DeployAction1:
      ...
      DeployAction2:
      ...
```

액션 간 종속성 설정

다음 지침을 사용하여 워크플로의 작업 간 종속성을 설정하십시오.

종속성을 구성할 때는 다음 지침을 따르십시오.

- 작업이 그룹 내에 있는 경우 해당 작업은 동일한 그룹 내의 다른 작업에만 종속될 수 있습니다.
- 작업 및 작업 그룹은 YAML 계층 내 동일한 수준에 있는 다른 작업 및 작업 그룹에 종속될 수 있지만 다른 수준에서는 사용할 수 없습니다.

Visual

비주얼 편집기를 사용하여 종속성을 설정하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로 다이어그램에서 다른 작업에 종속될 작업을 선택합니다.
8. 입력 탭을 선택합니다.
9. 종속 - 선택 사항에서 다음을 수행하십시오.

   이 액션을 실행하기 위해 성공적으로 실행되어야 하는 액션, 액션 그룹 또는 게이트를 지정하십시오.

   'depends on' 기능에 대한 자세한 내용은 을 참조하십시오. 시퀀싱 액션
10. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사하십시오.
11. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

편집기를 사용하여 종속성을 설정하려면 YAML

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 다른 액션에 따라 달라지는 액션에 다음과 비슷한 코드를 추가하세요.

```
action-name:
  DependsOn:
    - action-1
```

더 많은 예제는 작업 간 종속성을 구성하는 방법의 예를 참조합니다. 일반 지침은 을 참조하십시오액션 간 종속성 설정. 자세한 내용은 작업에 대한 DependsOn 속성에 워크플로우 YAML 정의 대한 설명을 참조하십시오.

8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사하십시오.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

작업 간 아티팩트 및 파일 공유

아티팩트는 워크플로 작업의 출력이며, 일반적으로 폴더 또는 파일 아카이브로 구성됩니다. 아티팩트는 작업 간에 파일과 정보를 공유할 수 있게 해주므로 중요합니다.

예를 들어 파일을 생성하는 빌드 작업이 있는데 배포 작업에서 해당 `sam-template.yml` 파일을 사용하려는 경우를 예로 들 수 있습니다. 이 시나리오에서는 아티팩트를 사용하여 빌드 작업이 배포 작업과 `sam-template.yml` 파일을 공유하도록 허용할 수 있습니다. 코드는 다음과 같을 수 있습니다.

```
Actions:
  BuildAction:
    Identifier: aws/build@v1
    Steps:
```

```
      - Run: sam package --output-template-file sam-template.yml
    Outputs:
      Artifacts:
        - Name: MYARTIFACT
          Files:
            - sam-template.yml
  DeployAction:
    Identifier: aws/cfn-deploy@v1
    Inputs:
      Artifacts:
        - MYARTIFACT
    Configuration:
      template: sam-template.yml
```

이전 코드에서는 build action (BuildAction) 이 sam-template.yml 파일을 생성한 다음 이라는 MYARTIFACT 출력 아티팩트에 추가합니다. 후속 배포 작업 (DeployAction) 은 MYARTIFACT 입력으로 지정하여 sam-template.yml 파일에 대한 액세스 권한을 부여합니다.

주제

- 아티팩트를 출력 및 입력으로 지정하지 않고도 아티팩트를 공유할 수 있나요?
- 워크플로 간에 아티팩트를 공유할 수 있습니까?
- 아티팩트의 예
- 출력 아티팩트 정의
- 입력 아티팩트 정의
- 아티팩트의 파일 참조
- 아티팩트 다운로드

아티팩트를 출력 및 입력으로 지정하지 않고도 아티팩트를 공유할 수 있나요?

예. 액션 코드의 Outputs 및 Inputs 섹션에 아티팩트를 지정하지 않고도 액션 간에 아티팩트를 공유할 수 있습니다. YAML 이렇게 하려면 컴퓨팅 공유를 켜야 합니다. 컴퓨팅 공유 및 켜져 있을 때 아티팩트를 지정하는 방법에 대한 자세한 내용은 을 참조하십시오작업 간 컴퓨팅 공유.

> **Note**
>
> 컴퓨팅 공유 기능을 사용하면 `Outputs` 및 `Inputs` 섹션이 필요 없어 워크플로의 YAML 코드를 간소화할 수 있지만, 이 기능을 켜기 전에 알아두어야 할 제한 사항이 있습니다. 이러한 제한에 대한 자세한 내용은 을 참조하십시오컴퓨팅 공유 고려 사항.

워크플로 간에 아티팩트를 공유할 수 있습니까?

아니요. 서로 다른 워크플로 간에는 아티팩트를 공유할 수 없지만 동일한 워크플로 내의 작업 간에는 아티팩트를 공유할 수 있습니다.

아티팩트의 예

다음 예는 Amazon CodeCatalyst 워크플로 정의 파일에서 아티팩트를 출력, 입력 및 참조하는 방법을 보여줍니다.

주제

- 예: 아티팩트 출력
- 예: 다른 작업에서 생성된 아티팩트 입력
- 예: 여러 아티팩트의 파일 참조
- 예: 단일 아티팩트의 파일 참조
- 예: a가 있을 때 아티팩트에 있는 파일 참조 WorkflowSource
- 예: 작업 그룹이 있을 때 아티팩트에 있는 파일 참조

예: 아티팩트 출력

다음 예제는 두 개의.jar 파일이 포함된 아티팩트를 출력하는 방법을 보여줍니다.

```
Actions:
  Build:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ARTIFACT1
          Files:
            - build-output/file1.jar
```

```
            - build-output/file2.jar
```

예: 다른 작업에서 생성된 아티팩트 입력

다음 예제는 ARTIFACT4 호출된 아티팩트를 출력하고 입력하는 방법을 보여줍니다. BuildActionA BuildActionB

```
Actions:
  BuildActionA:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ARTIFACT4
          Files:
            - build-output/file1.jar
            - build-output/file2.jar
  BuildActionB:
    Identifier: aws/build@v1
    Inputs:
      Artifacts:
        - ARTIFACT4
    Configuration:
```

예: 여러 아티팩트의 파일 참조

다음 예제에서는 이름이 ART5 및 in인 두 아티팩트를 출력한 다음 (under) ART6 에서 BuildActionC (아티팩트 내) 및 file5.txt (아티팩트 내ART5) 라는 이름의 두 파일을 참조하는 방법을 보여줍니다. file6.txt ART6 BuildActionD Steps

> **Note**
>
> 파일 참조에 대한 자세한 내용은 을 참조하십시오. 아티팩트의 파일 참조

> **Note**
>
> 이 예제에서는 사용 중인 $CATALYST_SOURCE_DIR_ART5 접두사를 보여 주지만 생략할 수도 있습니다. 이는 기본 *ART5* 입력이기 때문입니다. 기본 입력에 대한 자세한 내용은 을 참조하십시오아티팩트의 파일 참조.

```
Actions:
  BuildActionC:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ART5
          Files:
            - build-output/file5.txt
        - Name: ART6
          Files:
            - build-output/file6.txt
  BuildActionD:
    Identifier: aws/build@v1
    Inputs:
      Artifacts:
        - ART5
        - ART6
    Configuration:
      Steps:
        - run: cd $CATALYST_SOURCE_DIR_ART5/build-output && cat file5.txt
        - run: cd $CATALYST_SOURCE_DIR_ART6/build-output && cat file6.txt
```

예: 단일 아티팩트의 파일 참조

다음 예제에서는 ART7 에서 이름이 지정된 하나의 아티팩트를 출력한 다음 (BuildActionEunder) 에서 참조 file7.txt (아티팩트 내에서ART7) 하는 방법을 보여줍니다. BuildActionF Steps

참조에 다음이 필요하지 않다는 점에 주목하세요. $CATALYST_SOURCE_DIR_ *artifact-name* build-output디렉터리 앞에 접두사를 붙이세요. 에서와 같이 예: 여러 아티팩트의 파일 참조 지정된 항목이 하나뿐이기 때문입니다. Inputs

> **Note**
>
> 파일 참조에 대한 자세한 내용은 을 참조하십시오아티팩트의 파일 참조.

```
Actions:
  BuildActionE:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
```

```
        - Name: ART7
          Files:
            - build-output/file7.txt
  BuildActionF:
    Identifier: aws/build@v1
    Inputs:
      Artifacts:
        - ART7
    Configuration:
      Steps:
        - run: cd build-output && cat file7.txt
```

예: a가 있을 때 아티팩트에 있는 파일 참조 WorkflowSource

다음 예제에서는 ART8 에서 이름이 지정된 하나의 아티팩트를 출력한 다음 (BuildActionGunder) 에서 참조 file8.txt (아티팩트 내에서ART8) 하는 방법을 보여줍니다. BuildActionH Steps

참조에 다음이 얼마나 필요한지 주목하세요. $CATALYST_SOURCE_DIR_ *artifact-name* 접두사, 에서와 같습니다. 예: 여러 아티팩트의 파일 참조 이는 Inputs (소스 및 아티팩트) 에 여러 항목이 지정되어 있기 때문에 파일을 찾을 위치를 나타내는 접두사가 필요하기 때문입니다.

> **Note**
>
> 파일 참조에 대한 자세한 내용은 을 참조하십시오. 아티팩트의 파일 참조

```
Actions:
  BuildActionG:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ART8
          Files:
            - build-output/file8.txt
  BuildActionH:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
      Artifacts:
        - ART8
    Configuration:
```

```
      Steps:
        - run: cd $CATALYST_SOURCE_DIR_ART8/build-output && cat file8.txt
```

예: 작업 그룹이 있을 때 아티팩트에 있는 파일 참조

다음 예제에서는 ART9 in ActionGroup1ActionI, 이라는 이름의 아티팩트를 출력한 다음 참조 file9.txt (ART9아티팩트) 를 출력하는 방법을 보여 줍니다. ActionJ

파일 참조에 대한 자세한 내용은 을 참조하십시오. 아티팩트의 파일 참조

```
Actions:
  ActionGroup1:
    Actions:
      ActionI:
        Identifier: aws/build@v1
        Outputs:
          Artifacts:
            - Name: ART9
              Files:
                - build-output/file9.yml
      ActionJ:
        Identifier: aws/cfn-deploy@v1
        Inputs:
          Sources:
            - WorkflowSource
          Artifacts:
            - ART9
        Configuration:
          template: /artifacts/ActionGroup1@ActionJ/ART9/build-output/file9.yml
```

출력 아티팩트 정의

다음 지침을 사용하여 Amazon CodeCatalyst 작업에서 출력하려는 아티팩트를 정의하십시오. 그러면 이 아티팩트를 다른 작업에서 사용할 수 있게 됩니다.

> **Note**
>
> 모든 액션이 출력 아티팩트를 지원하는 것은 아닙니다. 액션이 해당 기능을 지원하는지 확인하려면 다음에 나오는 시각적 편집기 지침을 살펴보고 액션에 출력 탭의 출력 아티팩트 버튼이 포함되어 있는지 확인하세요. '예'인 경우 출력 아티팩트가 지원됩니다.

Visual

비주얼 에디터를 사용하여 출력 아티팩트를 정의하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로 다이어그램에서 아티팩트를 생성할 작업을 선택합니다.
8. 출력 탭을 선택합니다.
9. 아티팩트에서 아티팩트 추가를 선택합니다.
10. 객체 추가를 선택하고 다음과 같이 필드에 정보를 입력합니다.

    빌드 아티팩트 이름

    액션으로 생성된 아티팩트의 이름을 지정합니다. 아티팩트 이름은 워크플로우 내에서 고유해야 하며 영숫자 (a-z, A-Z, 0-9) 와 밑줄 (_) 로 제한됩니다. 공백, 하이픈 (-) 및 기타 특수 문자는 사용할 수 없습니다. 출력 아티팩트 이름에 공백, 하이픈 및 기타 특수 문자를 사용할 때는 따옴표를 사용할 수 없습니다.

    예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

    빌드로 생성된 파일

    액션에 의해 출력되는 아티팩트에 CodeCatalyst 포함되는 파일을 지정합니다. 이러한 파일은 워크플로 작업이 실행될 때 생성되며 소스 저장소에서도 사용할 수 있습니다. 파일 경로는 소스 리포지토리 또는 이전 작업의 아티팩트에 있을 수 있으며 소스 리포지토리 또는 아티팩트 루트를 기준으로 합니다. 글로브 패턴을 사용하여 경로를 지정할 수 있습니다. 예시:

    - 빌드 위치 또는 소스 리포지토리 위치의 루트에 있는 단일 파일을 지정하려면 `my-file.jar`를 사용합니다..
    - 하위 디렉터리에 단일 파일을 지정하려면 `directory/my-file.jar` 또는 `directory/subdirectory/my-file.jar`를 사용합니다.

- 모든 파일을 지정하려면 "**/*"를 사용합니다. ** glob 패턴은 임의의 수의 하위 디렉터리와 일치함을 나타냅니다.
- directory라는 디렉터리에 있는 모든 파일 및 디렉터리를 지정하려면 "directory/**/*"를 사용합니다. ** glob 패턴은 임의의 수의 하위 디렉터리와 일치함을 나타냅니다.
- directory라는 디렉터리의 모든 파일을 지정하되 해당 하위 디렉터리는 지정하지 않으려면 "directory/*"를 사용합니다.

> **Note**
>
> 파일 경로에 별표 (*) 또는 기타 특수 문자가 하나 이상 포함된 경우 경로를 큰따옴표 () 로 묶으십시오. "" 특수 문자에 대한 자세한 내용은 을 참조하십시오. 구문 지침 및 규칙

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오작업 간 아티팩트 및 파일 공유.

> **Note**
>
> 파일을 찾을 아티팩트나 소스를 나타내는 접두사를 파일 경로에 추가해야 할 수도 있습니다. 자세한 내용은 소스 리포지토리 파일 참조 및 아티팩트의 파일 참조 단원을 참조하세요.

11. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

편집기를 사용하여 출력 아티팩트를 정의하려면 YAML

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.

4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 워크플로 작업에 다음과 비슷한 코드를 추가합니다.

```
action-name:
  Outputs:
    Artifacts:
      - Name: artifact-name
        Files:
          - file-path-1
          - file-path-2
```

   더 많은 예제는 아티팩트의 예를 참조합니다. 자세한 내용은 해당 작업에 워크플로우 YAML 정의 대한 을 참조하십시오.
8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사하십시오.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

입력 아티팩트 정의

다른 Amazon CodeCatalyst 작업에서 생성된 아티팩트를 사용하려면 해당 객체를 현재 작업에 대한 입력으로 지정해야 합니다. 작업에 따라 여러 아티팩트를 입력으로 지정할 수 있습니다. 자세한 내용은 해당 작업에 대한 을 참조하십시오. 워크플로우 YAML 정의

> **Note**
>
> 다른 워크플로의 아티팩트는 참조할 수 없습니다.

다음 지침에 따라 다른 액션의 아티팩트를 현재 작업에 대한 입력으로 지정하십시오.

전제 조건

시작하기 전에 다른 액션의 아티팩트가 출력되었는지 확인하십시오. 자세한 내용은 출력 아티팩트 정의 단원을 참조하십시오. 아티팩트를 출력하면 다른 작업에서 사용할 수 있습니다.

Visual

아티팩트를 액션에 대한 입력으로 지정하기 (비주얼 에디터)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로우 다이어그램에서 아티팩트를 입력으로 지정할 작업을 선택합니다.
8. 입력을 선택합니다.
9. 아티팩트 - 선택 사항에서 다음을 수행하십시오.

   이 작업에 대한 입력으로 제공하려는 이전 작업의 아티팩트를 지정합니다. 이러한 아티팩트는 이전 작업에서 출력 아티팩트로 이미 정의되어 있어야 합니다.

   입력 아티팩트를 지정하지 않는 경우 에서 하나 이상의 소스 리포지토리를 지정해야 합니다. *action-name*/Inputs/Sources

   예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

   > **Note**
   >
   > 아티팩트 - 선택적 드롭다운 목록을 사용할 수 없거나 (비주얼 편집기) 를 검증할 때 오류가 발생하는 경우, 작업이 하나의 YAML 입력만 지원하기 때문일 수 있습니다. YAML 이런 경우에는 소스 입력을 제거해 보세요.

10. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드를 검증하십시오.
11. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

아티팩트를 작업에 대한 입력으로 지정하려면 (YAML편집기)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 아티팩트를 입력으로 지정하려는 액션에 다음과 비슷한 코드를 추가합니다.

```
action-name:
  Inputs:
    Artifacts:
      - artifact-name
```

   더 많은 예제는 아티팩트의 예를 참조합니다.
8. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

아티팩트의 파일 참조

아티팩트 내에 있는 파일이 있고 Amazon CodeCatalyst 워크플로 작업 중 하나에서 이 파일을 참조해야 하는 경우 다음 절차를 완료하십시오.

> **Note**
>
> 소스 리포지토리 파일 참조 섹션도 참조하십시오.

Visual

사용할 수 없습니다. YAML지침을 YAML 보려면 선택하십시오.

YAML

아티팩트의 파일을 참조하려면 (YAML편집기)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 파일을 참조하려는 액션에 다음과 비슷한 코드를 추가합니다.

```
Actions:
  My-action:
    Inputs:
      Sources:
        - WorkflowSource
      Artifacts:
        - artifact-name
    Configuration:
      template: artifact-path/path/to/file.yml
```

이전 코드에서 다음을 바꾸십시오.

- *artifact-name* 아티팩트의 이름으로.
- *artifact-path* 다음 표의 값을 사용합니다.

참조를 추가하는 경우...	Replace ***artifact-path*** 와 함께...
빌드 액션 또는 테스트 액션	$CATALYST_SOURCE_DIR_ *artifact-name* /
기타 모든 액션	$CATALYST_SOURCE_DIR_ *artifact-name* / 또는

참조를 추가하는 경우...	Replace ***artifact-path*** 와 함께...
	/artifacts/ *current-action-name* /*artifact-name* / 또는 현재 액션이 액션 그룹 내에 있는 경우: /artifacts/ *current-action-group@current-action-name* /*artifact-name* /

예제는 아티팩트의 예 섹션을 참조하세요.

> **Note**
>
> 생략할 수 있습니다.*artifact-path* 다음과 같은 경우에는 아티팩트 루트 디렉터리를 기준으로 파일 경로를 지정하기만 하면 됩니다.
>
> - 참조를 포함하는 액션에는 아래에 항목이 하나만 포함됩니다 `Inputs` (예: 입력 아티팩트는 하나이고 소스는 포함되지 않음).
> - 참조하려는 파일은 기본 입력에 있습니다. 기본 입력은 a이거나 나열된 첫 번째 입력 아티팩트 (없는 경우) 입니다. `WorkflowSource` `WorkflowSource`

8. (선택 사항) 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사하려면 [Validate] 를 선택합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

아티팩트 다운로드

문제 해결을 위해 Amazon CodeCatalyst 워크플로 작업에서 생성된 아티팩트를 다운로드하고 검사할 수 있습니다. 다운로드할 수 있는 아티팩트에는 두 가지 유형이 있습니다.

- 소스 아티팩트 — 실행 시작 당시 존재했던 소스 리포지토리 컨텐츠의 스냅샷이 들어 있는 아티팩트입니다.
- 워크플로 아티팩트 — 워크플로 구성 파일의 `Outputs` 속성에 정의된 아티팩트입니다.

워크플로우에서 출력한 아티팩트를 다운로드하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 워크플로 이름 아래에서 실행을 선택합니다.
6. 실행 기록의 실행 ID 열에서 실행을 선택합니다. 예: `Run-95a4d`.
7. 실행 이름 아래에서 아티팩트를 선택합니다.
8. 아티팩트 옆의 다운로드를 선택합니다. 아카이브 파일이 다운로드됩니다. 파일 이름은 7개의 임의의 문자로 구성되어 있습니다.
9. 선택한 아카이브 추출 유틸리티를 사용하여 아카이브를 추출합니다.

사용할 액션 버전 지정

기본적으로 워크플로에 작업을 추가하면 Amazon은 다음 형식을 사용하여 전체 버전을 워크플로 정의 파일에 CodeCatalyst 추가합니다.

v*major*.*minor*.*patch*

예:

```
My-Build-Action:
  Identifier: aws/build@v1.0.0
```

워크플로에서 항상 작업의 최신 마이너 버전 또는 패치 버전을 사용하도록 `Identifier` 속성의 전체 버전을 줄일 수 있습니다.

예를 들어, 다음과 같이 지정하는 경우

```
My-CloudFormation-Action:
  Identifier: aws/cfn-deploy@v1.0
```

... 그리고 최신 패치 버전이 `1.0.4` 인 경우 액션이 사용됩니다`1.0.4`. 예를 들어 `1.0.5` 이후 버전이 릴리스되면 액션이 사용됩니다`1.0.5`. 예를 `1.1.0` 들어 마이너 버전이 출시되더라도 액션은 계속 사용됩니다`1.0.5`.

버전 지정에 대한 자세한 지침은 다음 항목 중 하나를 참조하십시오.

다음 지침에 따라 워크플로에서 사용할 작업의 버전을 지정하십시오. 최신 메이저 또는 마이너 버전 또는 특정 패치 버전을 지정할 수 있습니다.

액션의 최신 마이너 버전 또는 패치 버전을 사용하는 것이 좋습니다.

Visual

사용할 수 없습니다. YAML지침을 YAML 보려면 선택하십시오.

YAML

액션의 최신 버전 또는 특정 패치 버전을 사용하도록 워크플로를 구성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 버전을 편집하려는 액션을 찾습니다.
8. 액션의 `Identifier` 속성을 찾아 버전을 다음 중 하나로 설정합니다.

   - 액션 식별자 @v*major* — 이 구문을 사용하면 워크플로에서 특정 메이저 버전을 사용하도록 하고 최신 마이너 버전과 패치 버전이 자동으로 선택되도록 할 수 있습니다.
   - 액션 식별자 @v*major*.*minor* — 이 구문을 사용하면 워크플로에서 특정 마이너 버전을 사용하도록 하고 최신 패치 버전이 자동으로 선택되도록 할 수 있습니다.
   - 액션 식별자 @v*major*.*minor*.*patch* — 워크플로에서 특정 패치 버전을 사용하도록 하려면 이 구문을 사용하십시오.

> **Note**
>
> 어떤 버전을 사용할 수 있는지 확실하지 않은 경우 을 참조하십시오사용 가능한 액션 버전 목록.

> **Note**
>
> 메이저 버전은 생략할 수 없습니다.

9. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로의 YAML 코드를 검증하십시오.
10. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

사용 가능한 액션 버전 목록

다음 지침에 따라 워크플로에서 사용할 수 있는 작업 버전을 결정하십시오.

Visual

사용할 수 있는 작업 버전을 결정하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 보려는 버전의 액션을 찾으세요.
   a. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
   b. 워크플로우의 이름을 선택하거나 새로 만들 수 있습니다. 워크플로우 만들기에 대한 자세한 내용은 을 참조하십시오워크플로 생성.
   c. 편집을 선택합니다.
   d. 왼쪽 상단에서 + Actions를 선택하여 작업 카탈로그를 엽니다.
   e. 드롭다운 목록에서 CodeCatalyst 보려는 Amazon CodeCatalyst, CodeCatalyst Labs 및 타사 작업을 선택하거나 큐레이션된 GitHub 작업을 GitHub보도록 선택합니다.
   f. 작업을 검색하고 해당 이름을 선택합니다. 더하기 기호 (+) 는 선택하지 마십시오.

   작업에 대한 세부 정보가 표시됩니다.
4. 작업 세부 정보 대화 상자의 오른쪽 상단에서 버전 드롭다운 목록을 선택하여 사용 가능한 작업 버전 목록을 확인합니다.

YAML

사용할 수 없습니다. 비주얼 에디터 지침을 보려면 '비주얼'을 선택하세요.

액션의 소스 코드 보기

작업의 소스 코드를 보고 위험한 코드, 보안 취약성 또는 기타 결함이 포함되어 있지 않은지 확인할 수 있습니다.

다음 지침을 사용하여 A CodeCatalyst, CodeCatalyst Labs 또는 타사 작업의 소스 코드를 확인하십시오.

> **Note**
>
> GitHub액션의 소스 코드를 보려면 GitHub Marketplace의 액션 페이지로 이동하세요. 페이지에는 액션의 소스 코드를 찾을 수 있는 액션 저장소로 연결되는 링크가 포함되어 있습니다.

> **Note**
>
> 빌드, 테스트, CodeCatalyst GitHub 액션 등의 액션의 소스 코드는 볼 수 없습니다.

> **Note**
>
> AWS 액션 또는 타사 액션의 액션 코드를 지원하거나 보장하지 않습니다. GitHub

액션의 소스 코드를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 코드를 보려는 액션을 찾으세요.
    a. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
    b. 워크플로우의 이름을 선택하거나 새로 만들 수 있습니다. 워크플로우 만들기에 대한 자세한 내용은 을 참조하십시오워크플로 생성.
    c. 편집을 선택합니다.
    d. 왼쪽 상단에서 + Actions를 선택하여 작업 카탈로그를 엽니다.
    e. 드롭다운 목록에서 CodeCatalyst 보려는 Amazon CodeCatalyst, CodeCatalyst Labs 및 타사 작업을 선택합니다.

f. 작업을 검색하고 해당 이름을 선택합니다. 더하기 기호 (+) 는 선택하지 마십시오.

작업에 대한 세부 정보가 표시됩니다.

4. 작업 세부 정보 대화 상자의 아래쪽에서 다운로드를 선택합니다.

작업의 소스 코드가 있는 Amazon S3 버킷을 보여주는 페이지가 나타납니다. Amazon S3에 대한 자세한 내용은 Amazon S3란 무엇입니까? 를 참조하십시오. Amazon 심플 스토리지 서비스 사용 설명서에서 확인할 수 있습니다.

5. 코드를 검사하여 품질 및 보안에 대한 기대치를 충족하는지 확인하십시오.

액션과 GitHub 통합

GitHub 액션은 GitHub 워크플로우와 함께 사용하도록 개발되었다는 점을 제외하면 CodeCatalyst 액션과 매우 비슷합니다. GitHub 액션에 대한 자세한 내용은 GitHub 액션 설명서를 참조하십시오.

CodeCatalyst 워크플로우에서 네이티브 GitHub 액션과 함께 CodeCatalyst 액션을 사용할 수 있습니다.

CodeCatalyst 워크플로에 GitHub 액션을 추가하는 방법에는 두 가지가 있습니다.

- CodeCatalyst 콘솔의 선별된 목록에서 GitHub 액션을 선택할 수 있습니다. 몇 가지 인기 있는 GitHub 액션을 사용할 수 있습니다. 자세한 내용은 큐레이션된 액션 GitHub 추가 단원을 참조하십시오.
- 사용하려는 GitHub 액션을 CodeCatalyst 콘솔에서 사용할 수 없는 경우 GitHub Actions 액션을 사용하여 액션을 추가할 수 있습니다.

  GitHub 액션은 액션을 래핑하여 CodeCatalyst 워크플로와 호환되도록 하는 GitHub 액션입니다. CodeCatalyst

  다음은 슈퍼 GitHub 린터 액션을 래핑하는 GitHub 액션 액션의 예시입니다.

```
Actions:
  GitHubAction:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
        - name: Lint Code Base
          uses: github/super-linter@v4
          env:
```

```
            VALIDATE_ALL_CODEBASE: "true"
            DEFAULT_BRANCH: main
```

이전 코드에서는 액션 CodeCatalyst GitHub 액션 (으로 `aws/github-actions-runner@v1` 식별됨) 이 Super-Linter 액션 (으로 식별`github/super-linter@v4`) 을 래핑하여 워크플로우에서 작동하도록 했습니다. CodeCatalyst

자세한 내용은 GitHub '액션' 액션 추가 단원을 참조하십시오.

이전 GitHub 예제와 같이 큐레이션된 액션과 큐레이션되지 않은 액션을 모두 GitHub Actions 액션 () `aws/github-actions-runner@v1` 안에 래핑해야 합니다. 액션이 제대로 작동하려면 래퍼가 필요합니다.

주제

- GitHub 액션은 액션과 어떻게 다릅니까 CodeCatalyst ?
- GitHub 액션이 워크플로의 다른 CodeCatalyst 액션과 상호 작용할 수 있나요?
- 어떤 GitHub 액션을 사용할 수 있나요?
- 의 GitHub 액션 제한 CodeCatalyst
- GitHub 액션 (상위 단계) 을 추가하려면 어떻게 해야 하나요?
- GitHub 액션이 GitHub 실행되나요?
- GitHub 워크플로도 사용할 수 있나요?
- 튜토리얼: 액션을 GitHub 사용한 린트 코드
- GitHub '액션' 액션 추가
- 큐레이션된 액션 GitHub 추가
- GitHub 출력 매개변수 내보내기
- GitHub 출력 매개변수 참조
- 'GitHub 액션' 액션 YAML

GitHub 액션은 액션과 어떻게 다릅니까 CodeCatalyst ?

GitHub CodeCatalyst 워크플로 내에서 사용되는 작업은 해당 작업과 동일한 수준의 액세스 및 통합 AWS 및 CodeCatalyst 기능 (예: 환경 및 문제) 을 갖지 않습니다. CodeCatalyst

GitHub 액션이 워크플로의 다른 CodeCatalyst 액션과 상호 작용할 수 있나요?

예. 예를 들어 GitHub 액션은 다른 CodeCatalyst 액션에서 생성된 변수를 입력으로 사용할 수 있으며 출력 파라미터 및 아티팩트를 CodeCatalyst 액션과 공유할 수도 있습니다. 자세한 내용은 GitHub 출력 매개변수 내보내기 및 GitHub 출력 매개변수 참조 단원을 참조하세요.

어떤 GitHub 액션을 사용할 수 있나요?

CodeCatalyst 콘솔을 통해 제공되는 모든 GitHub 작업과 GitHubMarketplace에서 제공되는 모든 GitHub 작업을 사용할 수 있습니다. Marketplace의 GitHub 액션을 사용하기로 결정한 경우 다음 제한 사항을 염두에 두십시오.

의 GitHub 액션 제한 CodeCatalyst

- GitHub CodeCatalyst Lambda 컴퓨팅 유형에는 작업을 사용할 수 없습니다.

- GitHub 작업은 이전 도구가 포함된 2022년 11월 런타임 환경 Docker 이미지에서 실행됩니다. 이미지 및 도구에 대한 자세한 내용은 을 참조하십시오. 런타임 환경 이미지 지정

- GitHub 내부적으로 `github`컨텍스트에 의존하거나 GitHub 참조별 리소스를 참조하는 작업은 작동하지 않습니다. CodeCatalyst 예를 들어, 다음 작업은 에서는 작동하지 않습니다. CodeCatalyst
  - GitHub 리소스를 추가, 변경 또는 업데이트하려는 작업 풀 리퀘스트를 업데이트하거나 이슈를 생성하는 작업을 예로 들 수 GitHub 있습니다.
  - 거의 모든 작업이 https://github.com/actions 에 나열되어 있습니다.
- GitHub Docker 컨테이너 작업인 작업은 작동하지만 기본 Docker 사용자 (루트) 가 실행해야 합니다. 1001 사용자 권한으로 작업을 실행하지 마십시오. (글을 쓰는 시점에서 사용자 1001은 작업을 하고 있지만 GitHub 내부에서는 작동하지 않습니다.) CodeCatalyst 자세한 내용은 액션에 대한 Dockerfile 지원의 USER항목을 참조하십시오. GitHub

CodeCatalyst 콘솔을 통해 사용할 수 있는 GitHub 작업 목록은 을 참조하십시오. 큐레이션된 액션 GitHub 추가

GitHub 액션 (상위 단계) 을 추가하려면 어떻게 해야 하나요?

CodeCatalyst 워크플로에 GitHub 액션을 추가하는 상위 단계는 다음과 같습니다.

1. CodeCatalyst 프로젝트에서 워크플로를 만듭니다. 워크플로는 애플리케이션을 빌드, 테스트 및 배포하는 방법을 정의하는 곳입니다. 자세한 내용은 워크플로우 시작하기 단원을 참조하십시오.

2. 워크플로우에서 GitHub 큐레이션된 작업을 추가하거나 Actions를 GitHub 추가합니다.
3. 다음 중 하나를 수행합니다.
   - 큐레이션된 작업을 추가하기로 선택한 경우 해당 작업을 구성하십시오. 자세한 내용은 큐레이션된 액션 GitHub 추가 단원을 참조하십시오.
   - 큐레이션되지 않은 액션을 추가하기로 선택한 경우 액션 GitHub액션 내에 액션의 코드를 붙여넣습니다. GitHub YAML GitHubMarketplace에서 선택한 GitHub 액션의 세부정보 페이지에서 이 코드를 찾을 수 있습니다. 제대로 작동하려면 코드를 약간 수정해야 할 수 CodeCatalyst 있습니다. 자세한 내용은 GitHub '액션' 액션 추가 단원을 참조하십시오.
4. (선택 사항) 워크플로 내에서 빌드 및 테스트 작업과 같은 다른 작업을 추가합니다. 자세한 내용은 워크플로를 통한 빌드, 테스트, 배포 단원을 참조하십시오.
5. 워크플로우는 수동 또는 트리거를 통해 자동으로 시작합니다. 워크플로는 GitHub 작업 및 워크플로의 다른 모든 작업을 실행합니다. 자세한 내용은 워크플로우 수동 실행 시작 단원을 참조하십시오.

자세한 단계는 다음을 참조하십시오.

- 큐레이션된 액션 GitHub 추가.
- GitHub '액션' 액션 추가.

GitHub 액션이 GitHub 실행되나요?

아니요. CodeCatalyst의 빌드 머신을 사용하여 GitHub 액션이 실행됩니다. CodeCatalyst

GitHub 워크플로도 사용할 수 있나요?

아니요.

튜토리얼: 액션을 GitHub 사용한 린트 코드

이 자습서에서는 Amazon 워크플로에 슈퍼 린터 GitHub 액션을 추가합니다. CodeCatalyst Super-Linter 작업은 코드를 검사하고 코드에 오류가 있는 영역, 형식 지정 문제, 의심스러운 구문을 찾아 결과를 콘솔에 출력합니다. CodeCatalyst 워크플로에 린터를 추가한 후 워크플로를 실행하여 샘플 Node.js 응용 프로그램 () 을 린트합니다. `app.js` 그런 다음 보고된 문제를 수정하고 워크플로를 다시 실행하여 수정 사항이 제대로 적용되었는지 확인합니다.

> **Tip**
>
> Super-Linter를 사용하여 템플릿과 같은 YAML 파일을 린트하는 것을 고려해 보십시오.AWS CloudFormation

주제

사전 조건

시작하기 전에 필요한 사항은 다음과 같습니다.

- 연결된 CodeCatalyst 공간 AWS 계정. 자세한 내용은 스페이스 만들기 단원을 참조하십시오.
- CodeCatalyst 스페이스에 있는 빈 프로젝트가 `codecatalyst-linter-project` 호출되었습니다. 처음부터 시작 옵션을 선택하여 이 프로젝트를 생성합니다.

  자세한 내용은 Amazon에서 빈 프로젝트 생성 CodeCatalyst 단원을 참조하십시오.

1단계: 소스 리포지토리 만들기

이 단계에서는 에서 소스 리포지토리를 생성합니다 CodeCatalyst. 이 자습서에서는 이 리포지토리를 사용하여 샘플 애플리케이션 소스 파일을 저장합니다. `app.js`

소스 리포지토리에 대한 자세한 내용은 을 참조하십시오. 소스 리포지토리 생성

소스 리포지토리를 생성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다`codecatalyst-linter-project`.

3. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
4. 리포지토리 추가를 선택하고 리포지토리 생성을 선택합니다.
5. 리포지토리 이름에 다음을 입력합니다.

```
codecatalyst-linter-source-repository
```

6. 생성(Create)을 선택합니다.

2단계: app.js 파일 추가

이 단계에서는 소스 리포지토리에 app.js 파일을 추가합니다. app.js여기에는 린터가 발견할 수 있는 몇 가지 실수가 있는 함수 코드가 들어 있습니다.

app.js 파일을 추가하려면

1. CodeCatalyst 콘솔에서 프로젝트 () 를 선택합니다codecatalyst-linter-project.
2. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
3. 소스 리포지토리 목록에서 리포지토리를 선택합니다. codecatalyst-linter-source-repository
4. 파일에서 파일 생성을 선택합니다.
5. 텍스트 상자에 다음 코드를 입력합니다.

```
// const axios = require('axios')
// const url = 'http://checkip.amazonaws.com/';
let response;
/**
 *
 * Event doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-lambda-proxy-integrations.html#api-gateway-simple-proxy-for-lambda-input-format
 * @param {Object} event - API Gateway Lambda Proxy Input Format
 *
 * Context doc: https://docs.aws.amazon.com/lambda/latest/dg/nodejs-prog-model-context.html
 * @param {Object} context
 *
 * Return doc: https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-lambda-proxy-integrations.html
 * @returns {Object} object - API Gateway Lambda Proxy Output Format
 *
 */
```

```
exports.lambdaHandler = async (event, context) => {
  try {
    // const ret = await axios(url);
    response = {
      statusCode: 200,
      'body': JSON.stringify({
        message: 'hello world'
        // location: ret.data.trim()
      })
    }
  } catch (err) {
    console.log(err)
    return err
  }

    return response
}
```

6. 파일 이름에 를 입력합니다`app.js`. 다른 기본 옵션은 그대로 유지합니다.
7. 커밋을 선택합니다.

   라는 파일이 생성되었습니다`app.js`.

3단계: 수퍼-린터 작업을 실행하는 워크플로우 만들기

이 단계에서는 코드를 소스 리포지토리로 푸시할 때 Super-Linter 작업을 실행하는 워크플로를 만듭니다. 워크플로는 파일에 정의하는 다음과 같은 구성 요소로 구성됩니다. YAML

- 트리거 - 이 트리거는 소스 리포지토리에 변경 내용을 푸시할 때 워크플로 실행을 자동으로 시작합니다. 트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.
- GitHub 'Actions' 작업 — 트리거 시 Actions 작업은 Super-Linter GitHub 작업을 실행하며, Super-Linter 작업은 차례로 소스 저장소의 모든 파일을 검사합니다. 린터가 문제를 발견하면 워크플로 작업이 실패합니다.

Super-Linter 작업을 실행하는 워크플로를 만들려면

1. CodeCatalyst 콘솔에서 프로젝트를 선택합니다. `codecatalyst-linter-project`
2. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
3. 워크플로 만들기를 선택합니다.

4. 소스 리포지토리의 경우 선택합니다`codecatalyst-linter-source-repository`.
5. Branch의 경우 선택하십시오`main`.
6. 생성(Create)을 선택합니다.
7. YAML샘플 코드를 삭제합니다.
8. YAML다음을 추가하세요.

```
Name: codecatalyst-linter-workflow
SchemaVersion: "1.0"
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  SuperLinterAction:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
        github-action-code
```

위 코드에서 다음을 대체하십시오.*github-action-code* 이 절차의 다음 단계에 설명된 대로 Super-Linter 액션 코드를 사용하십시오.

9. 마켓플레이스의 슈퍼-린터 페이지로 이동하세요. GitHub
10. `steps:`(소문자) 에서 코드를 찾아 (대문자) 의 CodeCatalyst 워크플로우에 붙여넣습니다. `Steps:`

    다음 코드와 같이 CodeCatalyst 표준을 준수하도록 GitHub 액션 코드를 조정하십시오.

    이제 CodeCatalyst 워크플로는 다음과 같습니다.

```
Name: codecatalyst-linter-workflow
SchemaVersion: "1.0"
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  SuperLinterAction:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
```

```
      - name: Lint Code Base
        uses: github/super-linter@v4
        env:
          VALIDATE_ALL_CODEBASE: "true"
          DEFAULT_BRANCH: main
```

11. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 YAML 코드가 유효한지 확인합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 `codecatalyst-linter-source-repository` 리포지토리를 선택하고 [커밋] 을 다시 선택합니다.

    이제 워크플로가 생성되었습니다. 워크플로 맨 위에 정의된 트리거로 인해 워크플로 실행이 자동으로 시작됩니다.

실행 중인 워크플로를 보려면

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 방금 만든 워크플로를 선택합니다. `codecatalyst-linter-workflow`
3. 워크플로 다이어그램에서 선택합니다 SuperLinterAction.
4. 작업이 실패할 때까지 기다리세요. 린터가 코드에서 문제를 발견했기 때문에 이 오류가 발생할 것으로 예상됩니다.
5. CodeCatalyst 콘솔을 열어 두고 로 이동하십시오. 4단계: 슈퍼 린터가 발견한 문제 해결

4단계: 슈퍼 린터가 발견한 문제 해결

Super-Linter는 소스 리포지토리에 포함된 `README.md` 파일뿐만 아니라 `app.js` 코드에서도 문제를 발견했을 것입니다.

린터가 발견한 문제를 수정하기 위해

1. CodeCatalyst 콘솔에서 로그 탭을 선택한 다음 린트 코드 베이스를 선택합니다.

   Super-Linter 작업에서 생성된 로그가 표시됩니다.
2. Super-Linter 로그에서 90줄 근처까지 아래로 스크롤하면 문제의 시작 부분을 확인할 수 있습니다. 다음과 비슷해 보입니다.

```
/github/workspace/hello-world/app.js:3:13: Extra semicolon.
/github/workspace/hello-world/app.js:9:92: Trailing spaces not allowed.
/github/workspace/hello-world/app.js:21:7: Unnecessarily quoted property 'body'
 found.
```

```
/github/workspace/hello-world/app.js:31:1: Expected indentation of 2 spaces but
 found 4.
/github/workspace/hello-world/app.js:32:2: Newline required at end of file but not
 found.
```

3. 소스 리포지토리에서 app.js 수정하고 README.md 변경 사항을 적용하십시오.

> **Tip**
>
> 를 README.md markdown 수정하려면 다음과 같이 코드 블록에 추가하십시오.
>
> ```
> ```markdown
> Setup examples:
> ...
> ```
> ```

변경하면 다른 워크플로가 자동으로 실행됩니다. 워크플로가 완료될 때까지 기다리세요. 모든 문제를 해결했으면 워크플로가 성공적으로 완료될 것입니다.

정리

CodeCatalyst 정리하여 환경에서 이 자습서의 흔적을 제거하세요.

에서 정리하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 여세요.
2. 삭제codecatalyst-linter-source-repository.
3. 삭제codecatalyst-linter-workflow.

이 자습서에서는 코드를 린트하기 위해 CodeCatalyst 워크플로에 Super-Linter GitHub Action을 추가하는 방법을 배웠습니다.

GitHub '액션' 액션 추가

액션은 GitHub 액션을 래핑하여 CodeCatalyst 워크플로우와 호환되도록 하는 GitHub 액션입니다. CodeCatalyst

자세한 내용은 액션과 GitHub 통합 단원을 참조하십시오.

워크플로에 GitHub 액션 액션을 추가하려면 다음 단계를 따르십시오.

> **Tip**
>
> GitHub Actions 작업을 사용하는 방법을 보여주는 자습서는 을 참조하십시오튜토리얼: 액션을 GitHub 사용한 린트 코드.

Visual

비주얼 편집기를 사용하여 GitHub '액션' 액션을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 선택합니다. GitHub
9. GitHub Actions 작업을 검색하고 다음 중 하나를 수행하십시오.

   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - [GitHub Actions] 를 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 소스 보기를 선택하여 액션의 소스 코드를 확인합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.
10. 입력 및 구성 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오'GitHub 액션' 액션 YAML. 이 참조는 편집기와 시각적 편집기에 모두 나타나는 각 필드 (및 해당 YAML 속성 값) 에 대한 자세한 정보를 제공합니다. YAML
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.

12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

편집기를 사용하여 'GitHub 액션' 액션을 추가하려면 YAML

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 선택합니다. GitHub
9. GitHub Actions 작업을 검색하고 다음 중 하나를 수행하십시오.

   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - [GitHub Actions] 를 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 소스 보기를 선택하여 액션의 소스 코드를 확인합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.
10. 필요에 따라 YAML 코드의 속성을 수정하십시오. 사용 가능한 각 속성에 대한 설명은 에 나와 'GitHub 액션' 액션 YAML 있습니다.
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

GitHub '액션' 액션 정의

GitHub Actions는 워크플로 정의 파일 내의 YAML 속성 집합으로 정의됩니다. 이러한 속성에 대한 자세한 내용은 'GitHub 액션' 액션 YAML 을 참조하십시오워크플로우 YAML 정의.

큐레이션된 액션 GitHub 추가

GitHub 큐레이션된 GitHub 액션은 CodeCatalyst 콘솔에서 사용할 수 있는 액션이며, CodeCatalyst 워크플로우 내에서 GitHub 액션을 사용하는 방법을 보여주는 예입니다.

큐레이션된 GitHub 액션은 식별자로 식별되는 CodeCatalyst -authored GitHub Actions 액션으로 래핑됩니다. `aws/github-actions-runner@v1` 예를 들어 GitHub 큐레이션된 액션은 다음과 같습니다. TruffleHog OSS

```
Actions:
  TruffleHogOSS_e8:
    Identifier: aws/github-actions-runner@v1
    Inputs:
      Sources:
        - WorkflowSource # This specifies that the action requires this Workflow as a
 source
    Configuration:
      Steps:
        - uses: trufflesecurity/trufflehog@v3.16.0
          with:
            path: ' ' # Required; description: Repository path
            base: ' ' # Required; description: Start scanning from here (usually main
 branch).
            head: ' ' # Optional; description: Scan commits until here (usually dev
 branch).
            extra_args: ' ' # Optional; description: Extra args to be passed to the
 trufflehog cli.
```

이전 코드에서는 액션 CodeCatalyst GitHub 액션 (으로 식별`aws/github-actions-runner@v1`) 이 TruffleHog OSS 액션을 래핑하여 `trufflesecurity/trufflehog@v3.16.0` 워크플로우에서 작동하도록 했습니다. CodeCatalyst

이 액션을 구성하려면 의 빈 문자열을 사용자 고유의 값으로 바꾸면 됩니다. `with:` 예:

```
Actions:
  TruffleHogOSS_e8:
    Identifier: aws/github-actions-runner@v1
    Inputs:
      Sources:
        - WorkflowSource # This specifies that the action requires this Workflow as a
 source
    Configuration:
```

```
      Steps:
        - uses: trufflesecurity/trufflehog@v3.16.0
          with:
            path: ./
            base: main # Required; description: Start scanning from here (usually main
branch).
            head: HEAD # Optional; description: Scan commits until here (usually dev
branch).
            extra_args: '--debug --only-verified' # Optional; description: Extra args
to be passed to the trufflehog cli.
```

선별된 GitHub 동작을 워크플로에 추가하려면 다음 절차를 사용하십시오. CodeCatalyst 워크플로에서의 GitHub 작업 사용에 대한 일반 정보는 을 참조하십시오액션과 GitHub 통합.

> **Note**
>
> 큐레이트된 GitHub 액션 목록에 액션이 보이지 않더라도 GitHub 액션 액션을 사용하여 워크플로에 액션을 추가할 수 있습니다. 자세한 내용은 GitHub '액션' 액션 추가 단원을 참조하십시오.

Visual

비주얼 편집기를 사용하여 GitHub 큐레이션된 액션 추가하기

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 선택합니다. GitHub
9. GitHub 작업을 찾아보거나 검색하고 다음 중 하나를 수행하십시오.
   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

Or

- GitHub 액션의 이름을 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서는
  - (선택 사항) 소스 보기를 선택하여 액션의 소스 코드를 확인합니다.
  - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

10. 입력, 구성 및 출력 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오'GitHub 액션' 액션 YAML. 이 참조는 GitHub작업 작업에 사용할 수 있는 각 필드 (및 해당 YAML 속성 값) 에 대한 자세한 정보를 제공합니다. 이러한 정보는 시각적 YAML 편집기와 시각적 편집기에 모두 표시됩니다.

    GitHub큐레이션된 작업에 사용할 수 있는 구성 옵션에 대한 자세한 내용은 해당 설명서를 참조하십시오.

11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사하십시오.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

편집기를 사용하여 GitHub 큐레이션된 동작을 추가하려면 YAML

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 선택합니다. GitHub
9. GitHub 작업을 찾아보거나 검색하고 다음 중 하나를 수행하십시오.

    - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

Or

- GitHub 액션의 이름을 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서는
  - (선택 사항) 소스 보기를 선택하여 액션의 소스 코드를 확인합니다.
  - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

10. 필요에 따라 YAML 코드의 속성을 수정하십시오. GitHub Actions 작업에 사용할 수 있는 각 속성에 대한 설명은 에 나와 'GitHub 액션' 액션 YAML 있습니다.

    GitHub큐레이션된 작업에 사용할 수 있는 구성 옵션에 대한 자세한 내용은 해당 설명서를 참조하십시오.

11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사하십시오.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

GitHub 출력 매개변수 내보내기

CodeCatalyst 워크플로에서 GitHub 출력 매개변수를 사용할 수 있습니다.

> **Note**
>
> 출력 파라미터의 또 다른 단어는 변수입니다. 설명서에서 출력 매개변수라는 용어를 사용하기 때문에 GitHub 이 용어도 사용하겠습니다.

다음 지침에 따라 GitHub 액션에서 GitHub 출력 매개변수를 내보내 다른 CodeCatalyst 워크플로 작업에서 사용할 수 있도록 하십시오.

GitHub 출력 매개 변수를 내보내려면

1. 워크플로를 열고 편집을 선택합니다. 자세한 내용은 워크플로 생성 단원을 참조하십시오.
2. 내보내려는 출력 매개 변수를 생성하는 GitHub Actions 작업에서 다음과 같은 기본 `Variables` 속성이 있는 `Outputs` 섹션을 추가합니다.

```
Actions:
  MyGitHubAction:
```

```
    Identifier: aws/github-actions-runner@v1
    Outputs:
      Variables:
        - 'step-id_output-name'
```

바꾸기:

- *step-id* GitHub 액션 `steps` 섹션의 `id:` 속성 값으로
- *output-name* GitHub 출력 파라미터의 이름과 함께.

예

다음 예제는 라는 GitHub 출력 매개 변수를 내보내는 방법을 보여줍니다`SELECTEDCOLOR`.

```
Actions:
  MyGitHubAction:
    Identifier: aws/github-actions-runner@v1
    Outputs:
      Variables:
        - 'random-color-generator_SELECTEDCOLOR'
    Configuration:
      Steps:
        - name: Set selected color
          run: echo "SELECTEDCOLOR=green" >> $GITHUB_OUTPUT
          id: random-color-generator
```

GitHub 출력 매개변수 참조

GitHub 출력 매개 변수를 참조하려면 다음 지침을 따르십시오.

GitHub 출력 매개변수를 참조하려면

1. GitHub 출력 매개변수 내보내기의 단계를 수행하세요.

   이제 GitHub 출력 매개변수를 다른 작업에 사용할 수 있습니다.
2. 출력 매개변수 `Variables` 값을 기록해 둡니다. 여기에는 밑줄 (_) 이 포함됩니다.
3. 다음 구문을 사용하여 출력 매개변수를 참조하십시오.

```
${action-name.output-name}
```

바꾸기:

- *action-name* 출력 매개변수를 생성하는 CodeCatalyst GitHub Action의 이름을 사용합니다 (GitHub 액션의 name 또는 사용 안 id 함).
- *output-name* 앞서 언급한 출력 매개 변수 Variables 값을 사용합니다.

예

```
BuildActionB:
  Identifier: aws/build@v1
  Configuration:
    Steps:
      - Run: echo ${MyGitHubAction.random-color-generator_SELECTEDCOLOR}
```

컨텍스트를 사용한 예제

다음 예제는 SELECTEDCOLOR 변수를 에서 GitHubActionA 설정하고 출력한 다음 참조하는 방법을 보여줍니다BuildActionB.

```
Actions:
  GitHubActionA:
    Identifier: aws/github-actions-runner@v1
    Configuration:
      Steps:
        - name: Set selected color
          run: echo "SELECTEDCOLOR=green" >> $GITHUB_OUTPUT
          id: random-color-generator
    Outputs:
      Variables:
      - 'random-color-generator_SELECTEDCOLOR'

   BuildActionB:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: echo ${GitHubActionA.random-color-generator_SELECTEDCOLOR}
```

'GitHub 액션' 액션 YAML

GitHub액션 액션의 YAML 정의는 다음과 같습니다.

이 작업 정의는 광범위한 워크플로 정의 파일 내에 섹션으로 존재합니다. 이 파일에 대한 자세한 내용은 워크플로우 YAML 정의을 참조합니다.

다음 코드에서 YAML 속성을 선택하면 해당 속성에 대한 설명이 표시됩니다.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

```
# The workflow definition starts here.
# See ### ## for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  action-name:
    Identifier:  aws/github-actions-runner@v1
    DependsOn:
      - dependent-action-name-1
    Compute:
      Fleet: fleet-name
    Timeout: timeout-minutes
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Inputs:
      Sources:
        - source-name-1
        - source-name-2
      Artifacts:
        - artifact-name
      Variables:
        - Name: variable-name-1
```

```
        Value: variable-value-1
      - Name: variable-name-2
        Value: variable-value-2
  Outputs:
    Artifacts:
      - Name: output-artifact-1
        Files:
          - github-output/artifact-1.jar
          - "github-output/build*"
      - Name: output-artifact-2
        Files:
          - github-output/artifact-2.1.jar
          - github-output/artifact-2.2.jar
    Variables:
      - variable-name-1
      - variable-name-2
    AutoDiscoverReports:
      Enabled: true | false
      ReportNamePrefix: AutoDiscovered
      IncludePaths:
        - "**/*"
      ExcludePaths:
        - node_modules/cdk/junit.xml
      SuccessCriteria:
        PassRate: percent
        LineCoverage: percent
        BranchCoverage: percent
        Vulnerabilities:
          Severity: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
          Number: whole-number
    Reports:
      report-name-1:
        Format: format
        IncludePaths:
          - "*.xml"
        ExcludePaths:
          - report2.xml
          - report3.xml
        SuccessCriteria:
          PassRate: percent
          LineCoverage: percent
          BranchCoverage: percent
          Vulnerabilities:
            Severity: CRITICAL|HIGH|MEDIUM|LOW|INFORMATIONAL
```

```
            Number: whole-number
    Configuration
      Steps:
        - github-actions-code
```

액션 이름

(필수)

액션의 이름을 지정합니다. 모든 작업 이름은 워크플로 내에서 고유해야 합니다. 액션 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 액션 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

해당 UI: 구성 탭/*action-name*

Identifier

(*action-name*/Identifier)

작업을 식별합니다. 버전을 변경하려는 경우가 아니면 이 속성을 변경하지 마십시오. 자세한 내용은 사용할 액션 버전 지정 단원을 참조하십시오.

aws/github-actions-runner@v1GitHub액션에 사용 작업.

해당 UI: 워크플로 다이어그램/*action-name*/AWS/ @v1 라벨 github-actions-runner

DependsOn

(*action-name*/DependsOn)

(선택 사항)

이 액션을 실행하기 위해 성공적으로 실행되어야 하는 액션, 액션 그룹 또는 게이트를 지정하십시오.

'종속 조건' 기능에 대한 자세한 내용은 을 참조하십시오. 시퀀싱 액션

해당 UI: 입력 탭/ 기준 - 선택 사항

Compute

(*action-name*/Compute)

(선택 사항)

워크플로 작업을 실행하는 데 사용되는 컴퓨팅 엔진. 워크플로 수준 또는 작업 수준에서 컴퓨팅을 지정할 수 있지만 둘 다에서 지정할 수는 없습니다. 워크플로 수준에서 지정된 경우 컴퓨팅 구성은 워크플로에 정의된 모든 작업에 적용됩니다. 워크플로 수준에서는 동일한 인스턴스에서 여러 작업을 실행할 수도 있습니다. 자세한 내용은 작업 간 컴퓨팅 공유 단원을 참조하십시오.

해당 UI: 없음

Fleet

(*action-name*/Compute/Fleet)

(선택 사항)

워크플로 또는 워크플로 작업을 실행할 시스템 또는 플릿을 지정합니다. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다. 온디맨드 플릿의 예:,. `Linux.x86-64.Large` `Linux.x86-64.XLarge` 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

생략된 경우 기본값은 `Fleet` 입니다. `Linux.x86-64.Large`

해당 UI: 구성 탭/ 컴퓨팅 플릿 - 선택 사항

Timeout

(*action-name*/Timeout)

(선택 사항)

작업이 CodeCatalyst 종료되기 전에 작업을 실행할 수 있는 시간을 분 (편집기) 또는 시간과 분 (비주얼 편집기) 단위로 지정합니다. YAML 최소값은 5분이고 최대값은 에 설명되어 의 워크플로우 할당량 CodeCatalyst 있습니다. 기본 타임아웃은 최대 타임아웃과 동일합니다.

해당 UI: 구성 탭/ 타임아웃 - 선택 사항

Environment

(*action-name*/Environment)

(선택 사항)

작업에 사용할 CodeCatalyst 환경을 지정합니다. 작업은 선택한 환경에 VPC 지정된 AWS 계정 및 선택적 Amazon에 연결됩니다. 작업은 환경에 지정된 기본 IAM 역할을 사용하여 Amazon에 연결하고 Amazon VPC 연결에 지정된 IAM 역할을 사용하여 Amazon에 연결합니다VPC. AWS 계정

> **Note**
>
> 기본 IAM 역할에 작업에 필요한 권한이 없는 경우 다른 역할을 사용하도록 작업을 구성할 수 있습니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.

환경에 대한 자세한 내용은 및 에 AWS 계정 배포 VPCs 및 을 참조하십시오환경 생성.

해당 UI: 구성 탭/ 환경

Name

(*action-name*/Environment/Name)

(포함된 Environment 경우 필수)

작업에 연결할 기존 환경의 이름을 지정합니다.

해당 UI: 구성 탭/ 환경

Connections

(*action-name*/Environment/Connections)

(선택 사항)

작업에 연결할 계정 연결을 지정합니다. 에서 최대 1개의 계정 연결을 지정할 수 `Environment` 있습니다.

계정 연결을 지정하지 않은 경우:

- 작업은 CodeCatalyst 콘솔 환경에 지정된 AWS 계정 연결 및 기본 IAM 역할을 사용합니다. 환경에 계정 연결 및 기본 IAM 역할을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.
- 기본 IAM 역할에는 작업에 필요한 정책 및 권한이 포함되어야 합니다. 이러한 정책 및 권한이 무엇인지 확인하려면 작업 YAML 정의 설명서에서 역할 속성에 대한 설명을 참조하십시오.

계정 연결에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정. 환경에 계정 연결을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

해당 UI: 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환

Name

(*action-name*/Environment/Connections/Name)

(포함된 경우 Connections 필수)

계정 연결 이름을 지정합니다.

해당 UI: 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환

Role

(*action-name*/Environment/Connections/Role)

(포함된 경우 Connections 필수)

Amazon S3 및 Amazon과 같은 AWS 서비스에 액세스하고 이를 운영하기 위해 이 작업이 사용하는 IAM 역할의 이름을 지정합니다ECR. 스페이스의 AWS 계정 연결에 이 역할이 추가되었는지 확인하십시오. 계정 연결에 IAM 역할을 추가하려면 을 참조하십시오계정 연결에 IAM 역할 추가.

IAM역할을 지정하지 않으면 CodeCatalyst 콘솔 환경에 나열된 기본 IAM 역할이 작업에 사용됩니다. 환경에서 기본 역할을 사용하는 경우 해당 역할에 다음 정책이 적용되는지 확인하십시오.

> **Note**
>
> 이 작업에 `CodeCatalystWorkflowDevelopmentRole-`*spaceName* 역할을 사용할 수 있습니다. 이에 대한 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기 섹션을 참조하세요. `CodeCatalystWorkflowDevelopmentRole-`*spaceName*역할에는 보안 위험을 초래할 수 있는 전체 액세스 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다.

> **Warning**
>
> 권한을 GitHub 작업 작업에 필요한 권한으로 제한하십시오. 더 광범위한 권한이 있는 역할을 사용하면 보안 위험이 발생할 수 있습니다.

해당 UI: 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환

Inputs

(*action-name*/Inputs)

(선택 사항)

이 `Inputs` 섹션은 워크플로 실행 중에 작업에 필요한 데이터를 정의합니다.

> **Note**
>
> GitHub Actions 작업당 최대 4개의 입력 (소스 1개와 아티팩트 3개) 이 허용됩니다. 변수는 이 총계에 포함되지 않습니다.

다른 입력 (예: 소스 및 아티팩트) 에 있는 파일을 참조해야 하는 경우 소스 입력은 기본 입력이고 아티팩트는 보조 입력입니다. 보조 입력의 파일에 대한 참조에는 기본 입력과 구분하기 위해 특수 접두사가 사용됩니다. 세부 정보는 예: 여러 아티팩트의 파일 참조을 참조하세요.

해당 UI: 입력 탭

Sources

(*action-name*/Inputs/Sources)

(선택 사항)

작업에 필요한 소스 리포지토리를 나타내는 레이블을 지정합니다. 현재 지원되는 유일한 레이블은 `WorkflowSource` 워크플로 정의 파일이 저장되는 소스 저장소를 나타내는 레이블입니다.

소스를 생략하는 경우 아래에 입력 아티팩트를 하나 이상 지정해야 합니다. *action-name*`/Inputs/Artifacts`

소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

해당 UI: 입력 탭/ 소스 - 선택 사항

Artifacts - input

(*action-name*/Inputs/Artifacts)

(선택 사항)

이 작업에 대한 입력으로 제공하려는 이전 작업의 아티팩트를 지정합니다. 이러한 아티팩트는 이전 작업에서 출력 아티팩트로 이미 정의되어 있어야 합니다.

입력 아티팩트를 지정하지 않는 경우 에서 하나 이상의 소스 리포지토리를 지정해야 합니다. *action-name*/Inputs/Sources

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

> **Note**
>
> 아티팩트 - 선택적 드롭다운 목록을 사용할 수 없거나 (비주얼 편집기) 를 검증할 때 오류가 발생하는 경우, 작업이 하나의 YAML 입력만 지원하기 때문일 수 있습니다. YAML 이런 경우에는 소스 입력을 제거해 보세요.

해당 UI: 입력 탭/ 아티팩트 - 선택 사항

Variables - input

(*action-name*/Inputs/Variables)

(선택 사항)

작업에 사용할 수 있도록 하려는 입력 변수를 정의하는 이름/값 쌍의 시퀀스를 지정합니다. 변수 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 변수 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

예제를 비롯한 변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

해당 UI: 입력 탭/ 변수 - 선택 사항

Outputs

(*action-name*/Outputs)

(선택 사항)

워크플로우 실행 중 작업에 의해 출력되는 데이터를 정의합니다.

해당 UI: 출력 탭

Artifacts - output

(*action-name*/Outputs/Artifacts)

(선택 사항)

액션으로 생성된 아티팩트의 이름을 지정합니다. 아티팩트 이름은 워크플로우 내에서 고유해야 하며 영숫자 (a-z, A-Z, 0-9) 와 밑줄 (_) 로 제한됩니다. 공백, 하이픈 (-) 및 기타 특수 문자는 사용할 수 없습니다. 출력 아티팩트 이름에 공백, 하이픈 및 기타 특수 문자를 사용할 때는 따옴표를 사용할 수 없습니다.

예를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 출력 탭/ 아티팩트

Name

(*action-name*/Outputs/Artifacts/Name)

(포함된 Artifacts - output 경우 필수)

액션으로 생성된 아티팩트의 이름을 지정합니다. 아티팩트 이름은 워크플로우 내에서 고유해야 하며 영숫자 (a-z, A-Z, 0-9) 와 밑줄 (_) 로 제한됩니다. 공백, 하이픈 (-) 및 기타 특수 문자는 사용할 수 없습니다. 출력 아티팩트 이름에 공백, 하이픈 및 기타 특수 문자를 사용할 때는 따옴표를 사용할 수 없습니다.

예를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 출력 탭/아티팩트/아티팩트 추가/빌드 아티팩트 이름

Files

(*action-name*/Outputs/Artifacts/Files)

Artifacts - output(포함된 경우 필수)

액션에 의해 출력되는 아티팩트에 CodeCatalyst 포함할 파일을 지정합니다. 이러한 파일은 워크플로 작업이 실행될 때 생성되며 소스 저장소에서도 사용할 수 있습니다. 파일 경로는 소스 리포지토리 또는 이전 작업의 아티팩트에 있을 수 있으며 소스 리포지토리 또는 아티팩트 루트를 기준으로 합니다. 글로브 패턴을 사용하여 경로를 지정할 수 있습니다. 예시:

- 빌드 위치 또는 소스 리포지토리 위치의 루트에 있는 단일 파일을 지정하려면 `my-file.jar`를 사용합니다..
- 하위 디렉터리에 단일 파일을 지정하려면 `directory/my-file.jar` 또는 `directory/subdirectory/my-file.jar`를 사용합니다.
- 모든 파일을 지정하려면 `"**/*"`를 사용합니다. ** glob 패턴은 임의의 수의 하위 디렉터리와 일치함을 나타냅니다.

- `directory`라는 디렉터리에 있는 모든 파일 및 디렉터리를 지정하려면 `"directory/**/*"`를 사용합니다. ** glob 패턴은 임의의 수의 하위 디렉터리와 일치함을 나타냅니다.
- `directory`라는 디렉터리의 모든 파일을 지정하되 해당 하위 디렉터리는 지정하지 않으려면 `"directory/*"`를 사용합니다.

> **Note**
>
> 파일 경로에 별표 (*) 또는 기타 특수 문자가 하나 이상 포함된 경우 경로를 큰따옴표 () 로 묶으십시오. "" 특수 문자에 대한 자세한 내용은 을 참조하십시오. 구문 지침 및 규칙

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오작업 간 아티팩트 및 파일 공유.

> **Note**
>
> 파일을 찾을 아티팩트나 소스를 나타내는 접두사를 파일 경로에 추가해야 할 수도 있습니다. 자세한 내용은 소스 리포지토리 파일 참조 및 아티팩트의 파일 참조 단원을 참조하세요.

해당 UI: 출력 탭/아티팩트/아티팩트 추가/빌드로 생성된 파일

Variables - output

(*action-name*/Outputs/Variables)

(선택 사항)

후속 작업에서 사용할 수 있도록 액션에서 내보낼 변수를 지정합니다.

예제를 비롯한 변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

해당 UI: 출력 탭/변수/ 변수 추가

변수 이름-1

(*action-name*/Outputs/Variables변수 이름-1)

(선택 사항)

액션으로 내보내려는 변수의 이름을 지정합니다. 이 변수는 동일한 액션의 `Inputs` 또는 `Steps` 섹션에 이미 정의되어 있어야 합니다.

예제를 비롯한 변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

해당 UI: 출력 탭/변수/변수 추가/이름

AutoDiscoverReports

(*action-name*/Outputs/AutoDiscoverReports)

(선택 사항)

자동 검색 기능의 구성을 정의합니다.

자동 CodeCatalyst 검색을 활성화하면 작업에 `Inputs` 전달된 모든 파일과 작업 자체에서 생성된 모든 파일을 검색하여 테스트, 코드 적용 범위 및 소프트웨어 구성 분석 (SCA) 보고서를 찾습니다. 발견된 각 보고서에 대해 보고서를 보고서로 CodeCatalyst 변환합니다. CodeCatalyst CodeCatalyst 보고서는 CodeCatalyst 서비스에 완전히 통합된 보고서이며 콘솔을 통해 보고 조작할 수 있습니다. CodeCatalyst

> **Note**
>
> 기본적으로 자동 검색 기능은 모든 파일을 검사합니다. 또는 속성을 사용하여 검사할 파일을 제한할 수 있습니다. IncludePaths ExcludePaths

해당 UI: 없음

Enabled

(*action-name*/Outputs/AutoDiscoverReports/Enabled)

(선택 사항)

자동 검색 기능을 활성화하거나 비활성화합니다.

유효한 값은 `true` 또는 `false`입니다.

생략된 경우 기본값은 `Enabled` 입니다. `true`

해당 UI: 출력 탭/보고서/ 보고서 자동 검색

ReportNamePrefix

(*action-name*/Outputs/AutoDiscoverReports/ReportNamePrefix)

(포함되고 활성화된 경우 AutoDiscoverReports 필수)

관련 보고서의 이름을 지정하려면 찾은 모든 보고서 CodeCatalyst 앞에 붙는 접두사를 지정하십시오. CodeCatalyst 예를 들어 접두사를 지정하고 두 개의 테스트 보고서를 CodeCatalyst 자동으로 검색하는 경우 관련 CodeCatalyst 보고서의 이름은 `TestSuiteOne.xml` 및 `TestSuiteTwo.xml` 로 지정됩니다. `AutoDiscovered` `AutoDiscoveredTestSuiteOne` `AutoDiscoveredTestSuiteTwo`

해당 UI: 출력 탭/보고서/보고서 자동 검색/보고서 접두사

IncludePaths

(*action-name*/Outputs/AutoDiscoverReports/IncludePaths)

Or

(*action-name*/Outputs/Reports/*report-name-1*/IncludePaths)

(포함 및 활성화된 경우 또는 포함된 AutoDiscoverReports 경우 필수) Reports

원시 보고서를 검색할 때 CodeCatalyst 포함할 파일 및 파일 경로를 지정하십시오. 예를 들어`"/test/report/*"`, 지정하는 경우 는 작업에 사용된 전체 빌드 이미지를 CodeCatalyst 검색하여 `/test/report/*` 디렉터리를 찾습니다. 해당 디렉토리를 찾으면 해당 디렉토리에서 보고서를 찾습니다. CodeCatalyst

> **Note**
>
> 파일 경로에 별표 (*) 또는 기타 특수 문자가 하나 이상 포함된 경우 경로를 큰따옴표 () 로 묶으십시오. "" 특수 문자에 대한 자세한 내용은 을 참조하십시오. 구문 지침 및 규칙

이 속성을 생략하면 기본값은 입니다. 즉`"**/*"`, 모든 경로의 모든 파일이 검색에 포함됩니다.

> **Note**
>
> 수동으로 구성된 보고서의 경우 단일 `IncludePaths` 파일과 일치하는 글로브 패턴이어야 합니다.

해당 UI:

- 출력 탭/보고서/보고서 자동 검색/'경로 포함/제외'/ 경로 포함

- 출력 탭/보고서/보고서 수동 구성/*report-name-1*/'경로 포함/제외'/ 경로 포함

ExcludePaths

(*action-name*/Outputs/AutoDiscoverReports/ExcludePaths)

Or

(*action-name*/Outputs/Reports/*report-name-1*/ExcludePaths)

(선택 사항)

원시 보고서를 검색할 때 제외할 파일 및 파일 경로를 지정하십시오. CodeCatalyst 예를 들어"`/test/my-reports/**/*`", 지정하는 경우 CodeCatalyst 는 `/test/my-reports/` 디렉터리에 있는 파일을 검색하지 않습니다. 디렉터리의 모든 파일을 무시하려면 `**/*` glob 패턴을 사용하십시오.

> **Note**
>
> 파일 경로에 별표 (*) 또는 기타 특수 문자가 하나 이상 포함된 경우 경로를 큰따옴표 () 로 묶으십시오. "" 특수 문자에 대한 자세한 내용은 을 참조하십시오. 구문 지침 및 규칙

해당 UI:

- 출력 탭/보고서/보고서 자동 검색/'경로 포함/제외'/ 경로 제외
- 출력 탭/보고서/보고서 수동 구성/*report-name-1*/'경로 포함/제외'/ 경로 제외

SuccessCriteria

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria)

(선택 사항)

테스트, 코드 적용 범위, 소프트웨어 구성 분석 (SCA) 및 정적 분석 (SA) 보고서의 성공 기준을 지정합니다.

자세한 내용은 보고서의 성공 기준 구성 단원을 참조하십시오.

해당 UI:

- 출력 탭/보고서/보고서 자동 검색/성공 기준
- 출력 탭/보고서/보고서 수동 구성/*report-name-1*/성공 기준

PassRate

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/PassRate)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/PassRate)

(선택 사항)

테스트 보고서의 테스트 중 통과해야 관련 CodeCatalyst 보고서가 통과된 것으로 표시되는 테스트 비율의 비율을 지정합니다. 유효한 값에는 10진수가 포함됩니다. 예: `50`, `60.5`. 합격률 기준은 테스트 보고서에만 적용됩니다. 테스트 보고서에 대한 자세한 내용은 을 참조하십시오테스트 보고서.

해당 UI:

- 출력 탭/보고서/보고서 자동 검색/성공 기준/합격률
- 출력 탭/보고서/보고서 수동 구성/*report-name-1*/성공 기준/ 합격률

LineCoverage

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/LineCoverage)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/LineCoverage)

(선택 사항)

관련 보고서가 통과로 표시되기 위해 포함되어야 하는 코드 커버리지 CodeCatalyst 보고서의 줄 비율을 지정합니다. 유효한 값에는 10진수가 포함됩니다. 예: `50`, `60.5`. 라인 커버리지 기준은 코드 커버리지 보고서에만 적용됩니다. 코드 커버리지 보고서에 대한 자세한 내용은 을 참조하십시오코드 범위 보고서.

해당 UI:

- 출력 탭/보고서/보고서 자동 검색/성공 기준/회선 커버리지
- 출력 탭/보고서/보고서 수동 구성/*report-name-1*/성공 기준/ 회선 커버리지

BranchCoverage

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/BranchCoverage)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/BranchCoverage)

(선택 사항)

코드 커버리지 보고서에서 관련 CodeCatalyst 보고서를 통과로 표시하기 위해 반드시 포함해야 하는 브랜치의 비율을 지정하십시오. 유효한 값에는 10진수가 포함됩니다. 예: `50`, `60.5`. 브랜치 커버리지 기준은 코드 커버리지 보고서에만 적용됩니다. 코드 커버리지 보고서에 대한 자세한 내용은 을 참조하십시오코드 범위 보고서.

해당 UI:

- 출력 탭/보고서/보고서 자동 검색/성공 기준/지사 커버리지
- 출력 탭/보고서/보고서 수동 구성/*report-name-1*/성공 기준/ 지사 커버리지

Vulnerabilities

(*action-name*/Outputs/AutoDiscoverReports/SuccessCriteria/Vulnerabilities)

Or

(*action-name*/Outputs/Reports/*report-name-1*/SuccessCriteria/Vulnerabilities)

(선택 사항)

관련 보고서가 통과된 것으로 표시되도록 SCA CodeCatalyst 보고서에 허용되는 최대 취약성 수와 심각도를 지정하십시오. 취약성을 지정하려면 다음을 지정해야 합니다.

- 집계에 포함하려는 취약성의 최소 심각도. 가장 심각한 값부터 가장 심각하지 않은 값까지 유효한 값은,`CRITICAL`,, `HIGH` `MEDIUMLOW`, `INFORMATIONAL` 입니다.

예를 들어HIGH, 선택하면 HIGH CRITICAL 취약성이 집계됩니다.

- 허용하려는 지정된 심각도 내에서 허용되는 최대 취약성 수입니다. 이 수를 초과하면 CodeCatalyst 보고서가 실패로 표시됩니다. 유효한 값은 정수입니다.

취약성 기준은 SCA 보고서에만 적용됩니다. SCA보고서에 대한 자세한 내용은 을 참조하십시오소프트웨어 구성 분석 보고서.

최소 심각도를 지정하려면 Severity 속성을 사용하십시오. 최대 취약성 수를 지정하려면 Number 속성을 사용하십시오.

SCA보고서에 대한 자세한 내용은 을 참조하십시오품질 보고서 유형.

해당 UI:

- 출력 탭/보고서/보고서 자동 검색/성공 기준/취약성
- 출력 탭/보고서/보고서 수동 구성/*report-name-1*/성공 기준/ 취약성

Reports

(*action-name*/Outputs/Reports)

(선택 사항)

테스트 보고서의 구성을 지정하는 섹션.

해당 UI: 출력 탭/ 보고서

보고서 이름-1

(*action-name*/Outputs/Reports/보고서 이름-1)

(포함된 경우 필수) Reports

원시 보고서에서 생성될 CodeCatalyst 보고서에 부여하려는 이름.

해당 UI: 출력 탭/보고서/보고서 수동 구성/보고서 이름

Format

(*action-name*/Outputs/Reports/*report-name-1*/Format)

Reports(포함된 경우 필수)

보고서에 사용하는 파일 형식을 지정하십시오. 가능한 값은 다음과 같습니다.

- 테스트 보고서의 경우:
  - 오이의 경우 JSON Cucumber (비주얼 에디터) 또는 CUCUMBERJSON (YAML에디터) 를 지정하십시오.
  - 의 JUnit XML 경우 JUnit(비주얼 편집기) 또는 JUNITXML (YAML편집기) 를 지정합니다.
  - NUnitXML경우 NUnit(비주얼 편집기) 또는 NUNITXML (YAML편집기) 를 지정하십시오.
  - NUnit3의 XML 경우 NUnit3(비주얼 편집기) 또는 NUNIT3XML (YAML편집기) 를 지정합니다.
  - 비주얼 TRX 스튜디오의 경우 Visual Studio TRX (비주얼 편집기) 또는 VISUALSTUDIOTRX (YAML편집기) 를 지정합니다.
  - TestNG의 경우 XML TestNG (비주얼 편집기) 또는 **TESTNGXML** (편집기) 를 지정하십시오. YAML
- 코드 커버리지 보고서의 경우:
  - Clover의 경우 Clover XML (비주얼 에디터) 또는 CLOVERXML (YAML에디터) 를 지정하십시오.
  - Cobertura의 경우 Cobertura XML (비주얼 편집기) 또는 (편집기) 를 지정합니다. COBERTURAXML YAML
  - 의 JaCoCo XML 경우 JaCoCo(비주얼 편집기) 또는 (편집기) 를 지정하십시오. JACOCOXML YAML
  - simplecov-json이 아닌 simplecov에서 SimpleCov JSON 생성한 경우 Simplecov (비주얼 편집기) 또는 (편집기) 로 지정하십시오. SIMPLECOV YAML
- 소프트웨어 구성 분석 () 보고서의 경우: SCA
  - 의 SARIF 경우 SARIF(비주얼 에디터) 또는 SARIFSCA (YAML에디터) 를 지정하십시오.

해당 UI: 출력 탭/보고서/보고서 수동 구성/보고서 추가/*report-name-1*/보고서 유형 및 보고서 형식

Configuration

(*action-name*/Configuration)

(필수) 작업의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭

Steps

(*action-name*/Configuration/Steps)

(필수)

GitHub Marketplace의 GitHub 액션 세부정보 페이지에 표시되는 대로 액션 코드를 지정하십시오. 다음 가이드라인에 따라 코드를 추가하세요.

1. GitHub 작업 섹션의 코드를 CodeCatalyst 워크플로의 `steps: Steps:` 섹션에 붙여넣습니다. 코드는 대시 (-) 로 시작하며 다음과 비슷합니다.

   GitHub 붙여넣을 코드:

   ```
   - name: Lint Code Base
     uses: github/super-linter@v4
     env:
       VALIDATE_ALL_CODEBASE: false
       DEFAULT_BRANCH: master
       GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
   ```

2. 방금 붙여넣은 코드를 검토하고 표준에 맞도록 필요에 따라 수정하십시오. CodeCatalyst 예를 들어, 위의 코드 블록을 사용하여 에서 코드를 제거할 수 있습니다.*red italics*그리고 굵은 글씨로 코드를 추가합니다.

   CodeCatalyst 워크플로 yaml:

   ```
   Steps:
      - name: Lint Code Base
        uses: github/super-linter@v4
        env:
          VALIDATE_ALL_CODEBASE: false
          DEFAULT_BRANCH: mastermain
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
   ```

3. GitHub Action에 포함되어 있지만 `steps:` 섹션 내에는 없는 추가 코드가 필요하면 CodeCatalyst - equirent 코드를 사용하여 CodeCatalyst 워크플로에 추가하세요. 를 워크플로우 YAML 정의 검토하여 GitHub 코드를 포팅하는 CodeCatalyst 방법에 대한 통찰력을 얻을 수 있습니다. 자세한 마이그레이션 단계는 이 가이드의 범위를 벗어납니다.

다음은 GitHub Actions 작업에서 파일 경로를 지정하는 방법의 예입니다.

```
Steps:
  - name: Lint Code Base
    uses: github/super-linter@v4
```

```
    ...
  - run: cd /sources/WorkflowSource/MyFolder/  && cat file.txt
  - run: cd /artifacts/MyGitHubAction/MyArtifact/MyFolder/  && cat file2.txt
```

파일 경로 지정에 대한 자세한 내용은 소스 리포지토리 파일 참조 및 을 참조하십시오아티팩트의 파일 참조.

해당 UI: 구성 탭/ 작업 GitHub YAML

컴퓨팅 및 런타임 이미지 구성

CodeCatalyst 워크플로우에서 워크플로 작업을 실행하는 데 CodeCatalyst 사용하는 컴퓨팅 및 런타임 환경 이미지를 지정할 수 있습니다.

컴퓨트는 워크플로 작업을 CodeCatalyst 실행하기 위해 관리 및 유지 관리되는 컴퓨팅 엔진 (CPU, 메모리, 운영 체제) 을 말합니다.

> **Note**
>
> 컴퓨팅이 워크플로의 속성으로 정의된 경우 해당 워크플로에 있는 작업의 속성으로 정의할 수 없습니다. 마찬가지로 컴퓨팅이 작업의 속성으로 정의된 경우 워크플로에서 정의할 수 없습니다.

런타임 환경 이미지는 워크플로 작업을 CodeCatalyst 실행하는 Docker 컨테이너입니다. Docker 컨테이너는 선택한 컴퓨팅 플랫폼에서 실행되며, 운영 체제와 워크플로 작업에 필요할 수 있는 추가 도구 (예:, Node.js AWS CLI, .tar) 를 포함합니다.

주제

- 작업 간 컴퓨팅 공유
- 런타임 환경 이미지 지정

컴퓨팅 유형

CodeCatalyst 다음과 같은 컴퓨팅 유형을 제공합니다.

- 아마존 EC2
- AWS Lambda

EC2Amazon은 작업 실행 중에 최적화된 유연성을 제공하고 Lambda는 최적화된 작업 시작 속도를 제공합니다. Lambda는 시작 지연 시간이 짧아 더 빠른 워크플로 작업 실행을 지원합니다. Lambda를 사용하면 일반적인 런타임으로 서버리스 애플리케이션을 구축, 테스트 및 배포할 수 있는 기본 워크플로를 실행할 수 있습니다. 이러한 런타임에는 Node.js, Python, Java 등이 포함됩니다. NET, 그리고 Go. 그러나 Lambda가 지원하지 않는 몇 가지 사용 사례가 있으며, 이러한 사용 사례가 영향을 미칠 경우 Amazon 컴퓨팅 유형을 사용하십시오. EC2

- Lambda는 지정된 레지스트리의 런타임 환경 이미지를 지원하지 않습니다.
- Lambda는 루트 권한이 필요한 도구를 지원하지 않습니다. `yum`또는 `rpm` 같은 도구의 경우 Amazon EC2 컴퓨팅 유형 또는 루트 권한이 필요하지 않은 기타 도구를 사용하십시오.
- Lambda는 Docker 빌드 또는 실행을 지원하지 않습니다. Docker 이미지를 사용하는 다음 작업은 지원되지 않습니다: AWS CloudFormation 스택 배포, Amazon에 배포ECS, Amazon S3 게시, AWS CDK 부트스트랩, AWS CDK 배포, AWS Lambda 호출 및 작업. GitHub CodeCatalyst GitHub 액션 작업 내에서 실행되는 GitHub Docker 기반 액션도 Lambda 컴퓨팅에서 지원되지 않습니다. Podman과 같이 루트 권한이 필요하지 않은 대안을 사용할 수 있습니다.
- Lambda는 외부 파일 쓰기를 지원하지 않습니다. `/tmp` 워크플로 작업을 구성할 때 도구를 재구성하여 설치하거나 쓸 수 있습니다. `/tmp` 설치할 빌드 작업이 있는 경우 설치하도록 구성해야 합니다`npm`. `/tmp`
- Lambda는 15분 이상의 런타임을 지원하지 않습니다.

컴퓨팅 플릿

CodeCatalyst 다음과 같은 컴퓨팅 플릿을 제공합니다.

- 온디맨드 플릿

- 프로비저닝된 플릿

온디맨드 플릿을 사용하면 워크플로 작업이 시작되면 워크플로가 필요한 리소스를 프로비저닝합니다. 작업이 완료되면 시스템이 폐기됩니다. 액션을 실행한 시간 (분) 만큼만 비용을 지불하면 됩니다. 온디맨드 플릿은 완전 관리형이며, 수요 급증을 처리할 수 있는 자동 규모 조정 기능이 포함되어 있습니다.

CodeCatalyst 또한 EC2 Amazon에서 제공하고 유지 관리하는 시스템을 포함하는 프로비저닝된 플릿을 제공합니다. CodeCatalyst 프로비저닝된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비저닝된 플릿을 사용하면 머신이 항상 가동되므로 프로비저닝하는 동안 비용이 발생합니다.

플릿을 생성, 업데이트 또는 삭제하려면 스페이스 관리자 역할 또는 프로젝트 관리자 역할이 있어야 합니다.

온디맨드 플릿 속성

CodeCatalyst 다음과 같은 온디맨드 플릿을 제공합니다.

명칭	운영 체제	아키텍처	vCPUs	메모리 (GiB)	디스크 공간	지원되는 컴퓨팅 유형
`Linux.Arm64.Large`	Amazon Linux 2	Arm64	2	4	64GB	아마존 EC2
					10GB	Lambda
`Linux.Arm64.XLarge`	Amazon Linux 2	Arm64	4	8	128GB	아마존 EC2
					10GB	Lambda
`Linux.Arm64.2XLarge`	Amazon Linux 2	Arm64	8	16	128GB	아마존 EC2
`Linux.x86-64.Large`	Amazon Linux 2	x86-64	2	4	64GB	아마존 EC2

명칭	운영 체제	아키텍처	vCPUs	메모리 (GiB)	디스크 공간	지원되는 컴퓨팅 유형
					10GB	Lambda
`Linux.x86-64.XLarge`	Amazon Linux 2	x86-64	4	8	128GB	아마존 EC2
					10GB	Lambda
`Linux.x86-64.2XLarge`	Amazon Linux 2	x86-64	8	16	128GB	아마존 EC2

> **Note**
>
> 온디맨드 플릿의 사양은 청구 등급에 따라 달라집니다. 자세한 내용은 요금을 참조하세요.

플릿을 선택하지 않은 경우 를 사용합니다. CodeCatalyst `Linux.x86-64.Large`

프로비저닝된 플릿 속성

프로비저닝된 플릿에는 다음과 같은 속성이 포함됩니다.

운영 체제

운영 체제입니다. 사용할 수 있는 운영 체제는 다음과 같습니다.

- Amazon Linux 2
- Windows Server 2022

> **Note**
>
> Windows 플릿은 빌드 작업에서만 지원됩니다. 다른 액션은 현재 Windows를 지원하지 않습니다.

아키텍처

프로세서 아키텍처. 사용할 수 있는 아키텍처는 다음과 같습니다.

- x86_64
- Arm64

컴퓨터 유형

각 인스턴스의 머신 유형. 사용할 수 있는 유형의 로그는 다음과 같습니다.

vCPUs	메모리(GiB)	디스크 공간	운영 체제
2	4	64GB	Amazon Linux 2
4	8	128GB	Amazon Linux 2
			Windows Server 2022
8	16	128GB	Amazon Linux 2
			Windows Server 2022

Capacity

플릿에 할당된 초기 시스템 수로, 병렬로 실행할 수 있는 작업 수를 정의합니다.

규모 조정 모드

작업 수가 플릿 용량을 초과할 때의 동작을 정의합니다.

온디맨드 방식으로 추가 용량 프로비저닝

필요에 따라 새 작업이 실행되면 자동으로 확장되고 작업이 완료되면 기본 용량까지 축소하는 추가 시스템이 설정됩니다. 실행 중인 각 머신에 대해 분 단위로 비용을 지불해야 하므로 추가 비용이 발생할 수 있습니다.

추가 플릿 용량을 사용할 수 있을 때까지 대기

작업 실행은 머신을 사용할 수 있을 때까지 대기열에 배치됩니다. 이렇게 하면 추가 머신이 할당되지 않으므로 추가 비용이 제한됩니다.

프로비저닝된 플릿 생성

다음 지침에 따라 프로비저닝된 플릿을 생성하십시오.

> **Note**
>
> 프로비저닝된 플릿은 2주 동안 활동이 없으면 비활성화됩니다. 다시 사용하면 자동으로 다시 활성화되지만 이렇게 다시 활성화되면 대기 시간이 발생할 수 있습니다.

프로비저닝된 플릿을 만들려면

1. 탐색 창에서 CI/CD를 선택한 다음 Compute를 선택합니다.
2. 프로비저닝된 플릿 생성을 선택합니다.
3. 프로비저닝된 플릿 이름 텍스트 필드에 플릿 이름을 입력합니다.
4. 운영 체제 드롭다운 메뉴에서 운영 체제를 선택합니다.
5. 머신 유형 드롭다운 메뉴에서 머신의 머신 유형을 선택합니다.
6. 용량 텍스트 필드에 플릿의 최대 시스템 수를 입력합니다.
7. 규모 조정 모드 드롭다운 메뉴에서 원하는 오버플로 동작을 선택합니다. 필드에 대한 자세한 내용은 프로비저닝된 플릿 속성 섹션을 참조하세요.
8. 생성(Create)을 선택합니다.

프로비저닝된 플릿을 생성하고 나면 작업에 할당할 준비가 된 것입니다. 자세한 내용은 작업에 플릿 또는 컴퓨팅 할당 단원을 참조하십시오.

프로비저닝된 플릿 편집

다음 지침을 사용하여 프로비저닝된 플릿을 편집하십시오.

> **Note**
>
> 프로비저닝된 플릿은 2주 동안 활동이 없으면 비활성화됩니다. 다시 사용하면 자동으로 다시 활성화되지만 이렇게 다시 활성화되면 대기 시간이 발생할 수 있습니다.

프로비저닝된 플릿을 편집하려면

1. 탐색 창에서 CI/CD를 선택한 다음 Compute를 선택합니다.
2. 프로비저닝된 플릿 목록에서 편집하려는 플릿을 선택합니다.
3. 편집을 선택합니다.
4. 용량 텍스트 필드에 플릿의 최대 시스템 수를 입력합니다.
5. 규모 조정 모드 드롭다운 메뉴에서 원하는 오버플로 동작을 선택합니다. 필드에 대한 자세한 내용은 프로비저닝된 플릿 속성 섹션을 참조하세요.
6. 저장(Save)을 선택합니다.

프로비저닝된 플릿 삭제

프로비저닝된 플릿을 삭제하려면 다음 지침을 따르십시오.

프로비저닝된 플릿을 삭제하려면

> **Warning**
>
> 프로비저닝된 플릿을 삭제하기 전에 작업 코드에서 `Fleet` 속성을 삭제하여 모든 작업에서 해당 플릿을 삭제하십시오. YAML 프로비저닝된 플릿을 삭제한 후에도 계속 참조하는 작업은 다음 번에 해당 작업이 실행될 때 실패합니다.

1. 탐색 창에서 CI/CD를 선택한 다음 Compute를 선택합니다.
2. 프로비저닝된 플릿 목록에서 삭제하려는 플릿을 선택합니다.
3. Delete(삭제)를 선택합니다.
4. 삭제를 **`delete`** 확인하려면 입력하십시오.
5. Delete(삭제)를 선택합니다.

작업에 플릿 또는 컴퓨팅 할당

기본적으로 워크플로 작업은 Amazon EC2 컴퓨팅 유형의 `Linux.x86-64.Large` 온디맨드 플릿을 사용합니다. 프로비저닝된 플릿을 대신 사용하거나 다른 온디맨드 플릿을 사용하려면 (예`Linux.x86-64.2XLarge`:) 다음 지침을 사용하십시오.

Visual

시작하기 전 준비 사항

- 프로비저닝된 플릿을 할당하려면 먼저 프로비저닝된 플릿을 생성해야 합니다. 자세한 내용은 프로비저닝된 플릿 생성 단원을 참조하십시오.

프로비저닝된 플릿 또는 다른 플릿 유형을 작업에 할당하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로 다이어그램에서 프로비저닝된 플릿 또는 새 플릿 유형을 할당할 작업을 선택합니다.
8. 구성 탭을 선택합니다.
9. Compute 플릿에서 다음을 수행하십시오.

   워크플로 또는 워크플로 작업을 실행할 컴퓨터 또는 플릿을 지정합니다. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다. 온디맨드 플릿의 예:,. `Linux.x86-64.Large` `Linux.x86-64.XLarge` 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

   프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

   생략된 경우 기본값은 `Fleet` 입니다. `Linux.x86-64.Large`
10. (선택 사항) 커밋하기 전에 워크플로우 YAML 코드를 검증하려면 [Validate] 를 선택합니다.
11. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

시작하기 전 준비 사항

- 프로비저닝된 플릿을 할당하려면 먼저 프로비저닝된 플릿을 생성해야 합니다. 자세한 내용은 프로비저닝된 플릿 생성 단원을 참조하십시오.

프로비저닝된 플릿 또는 다른 플릿 유형을 작업에 할당하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 프로비저닝된 플릿 또는 새 플릿 유형을 할당하려는 작업을 찾으십시오.
8. 액션에서 `Compute` 속성을 추가하고 플릿 이름 또는 온디맨드 플릿 유형으로 설정합니다`Fleet`. 자세한 내용은 작업에 대한 `Fleet` 속성의 설명을 참조하십시오. 빌드 및 테스트 액션 YAML
9. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사하십시오.
10. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

작업 간 컴퓨팅 공유

기본적으로 워크플로의 작업은 플릿의 개별 인스턴스에서 실행됩니다. 이 동작은 입력 상태를 격리하고 예측할 수 있는 액션을 제공합니다. 기본 동작을 사용하려면 작업 간에 파일 및 변수와 같은 컨텍스트를 공유하도록 명시적으로 구성해야 합니다.

컴퓨팅 공유는 동일한 인스턴스의 워크플로우 내에서 모든 작업을 실행할 수 있는 기능입니다. 컴퓨팅 공유를 사용하면 인스턴스를 프로비저닝하는 데 소요되는 시간이 줄어들어 워크플로 런타임이 빨라질 수 있습니다. 또한 추가 워크플로를 구성하지 않고도 작업 간에 파일 (아티팩트) 을 공유할 수 있습니다.

컴퓨팅 공유를 사용하여 워크플로를 실행하면 기본 또는 지정된 플릿의 인스턴스가 해당 워크플로의 모든 작업 기간 동안 예약됩니다. 워크플로 실행이 완료되면 인스턴스 예약이 해제됩니다.

주제

공유 컴퓨팅에서 여러 작업 실행

워크플로 YAML 수준에서 정의의 `Compute` 속성을 사용하여 작업의 플릿 및 컴퓨팅 공유 속성을 모두 지정할 수 있습니다. 에서 시각적 편집기를 사용하여 컴퓨팅 속성을 구성할 수도 CodeCatalyst 있습니다. 플릿을 지정하려면 기존 플릿의 이름을 설정하고 컴퓨팅 유형을 로 EC2설정한 다음 컴퓨팅 공유를 켜십시오.

> **Note**
>
> 컴퓨팅 공유는 컴퓨팅 유형이 로 EC2설정된 경우에만 지원되며 Windows Server 2022 운영 체제에서는 지원되지 않습니다. 컴퓨팅 플릿, 컴퓨팅 유형 및 속성에 대한 자세한 내용은 을 참조하십시오컴퓨팅 및 런타임 이미지 구성.

> **Note**
>
> 프리 티어를 사용 중이고 워크플로 YAML 정의에서 `Linux.x86-64.XLarge` 또는 `Linux.x86-64.2XLarge` 플릿을 수동으로 지정하는 경우에도 작업은 기본 플릿(`Linux.x86-64.Large`) 에서 계속 실행됩니다. 컴퓨팅 가용성 및 요금에 대한 자세한 내용은 티어 옵션 표를 참조하십시오.

컴퓨팅 공유를 켜면 워크플로 소스가 들어 있는 폴더가 작업 전체에 자동으로 복사됩니다. 워크플로 정의 (YAML파일) 전체에서 출력 아티팩트를 구성하고 이를 입력 아티팩트로 참조할 필요가 없습니다. 워크플로 작성자는 컴퓨팅 공유를 사용하지 않는 것과 마찬가지로 입력과 출력을 사용하여 환경 변수를 연결해야 합니다. 워크플로 소스 외부의 작업 간에 폴더를 공유하려면 파일 캐싱을 고려해 보세요. 자세한 내용은 작업 간 아티팩트 및 파일 공유 및 워크플로우 실행 간 파일 캐싱 단원을 참조하세요.

워크플로 정의 파일이 있는 소스 저장소는 레이블로 식별됩니다. `WorkflowSource` 컴퓨팅 공유를 사용하는 동안 워크플로 원본은 이를 참조하는 첫 번째 작업에서 다운로드되며, 실행 중인 워크플로의 후속 작업에 자동으로 제공됩니다. 파일 추가, 수정 또는 제거와 같은 작업을 통해 워크플로 소스가 포함된 폴더를 변경한 내용은 워크플로의 후속 작업에서도 확인할 수 있습니다. 컴퓨팅 공유를 사용하지 않는 것처럼 모든 워크플로 작업에서 워크플로 소스 폴더에 있는 파일을 참조할 수 있습니다. 자세한 내용은 소스 리포지토리 파일 참조 단원을 참조하십시오.

> **Note**
>
> 컴퓨팅 공유 워크플로는 엄격한 작업 순서를 지정해야 하므로 병렬 작업을 설정할 수 없습니다. 시퀀스의 모든 작업에서 출력 아티팩트를 구성할 수 있지만 입력 아티팩트는 지원되지 않습니다.

컴퓨팅 공유 고려 사항

컴퓨팅 공유로 워크플로를 실행하여 워크플로 실행을 가속화하고 동일한 인스턴스를 사용하는 워크플로의 작업 간에 컨텍스트를 공유할 수 있습니다. 다음 사항을 고려하여 컴퓨팅 공유를 사용하는 것이 시나리오에 적합한지 결정하십시오.

	컴퓨팅 공유	컴퓨팅 공유 없음
컴퓨팅 유형	아마존 EC2	아마존EC2, AWS 람다
인스턴스 프로비저닝	작업은 동일한 인스턴스에서 실행됩니다.	작업은 별도의 인스턴스에서 실행됩니다.
운영 체제	Amazon Linux 2	아마존 리눅스 2, 윈도우 서버 2022 (빌드 액션 전용)
참조 파일	`$CATALYST_SOURCE_DIR_WorkflowSource` , `/sources/WorkflowSource/`	`$CATALYST_SOURCE_DIR_WorkflowSource` , `/sources/WorkflowSource/`
워크플루 구조	액션은 순차적으로만 실행할 수 있습니다.	작업은 병렬로 실행될 수 있습니다.

	컴퓨팅 공유	컴퓨팅 공유 없음
워크플로우 작업 전반의 데이터 액세스	캐시된 워크플로 소스 액세스 () `WorkflowSource`	공유 아티팩트의 출력에 액세스 (추가 구성 필요)

컴퓨팅 공유 켜기

워크플로우에 컴퓨팅 공유를 켜려면 다음 지침을 따르십시오.

Visual

비주얼 에디터를 사용하여 컴퓨팅 공유를 켜려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다.
5. 편집을 선택합니다.
6. 비주얼을 선택합니다.
7. 워크플로우 속성을 선택합니다.
8. 컴퓨팅 유형 드롭다운 메뉴에서 선택합니다 EC2.
9. (선택 사항) Compute 플릿 - 옵션 드롭다운 메뉴에서 워크플로 작업을 실행하는 데 사용할 플릿을 선택합니다. 온디맨드 플릿을 선택하거나 프로비저닝된 플릿을 생성하여 선택할 수 있습니다. 자세한 내용은 프로비저닝된 플릿 생성 및 작업에 플릿 또는 컴퓨팅 할당 섹션을 참조하세요.
10. 토글을 전환하여 컴퓨팅 공유를 켜면 워크플로의 작업이 동일한 플릿에서 실행되도록 할 수 있습니다.
11. (선택 사항) 워크플로의 실행 모드를 선택합니다. 자세한 내용은 실행의 대기열 동작 구성 단원을 참조하십시오.
12. [Commit] 을 선택하고 커밋 메시지를 입력한 다음 [Commit] 을 다시 선택합니다.

YAML

YAML편집기를 사용하여 컴퓨팅 공유를 켜려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다.
5. 편집을 선택합니다.
6. 선택하세요 YAML.
7. 컴퓨팅 공유를 켜고 SharedInstance 필드를 로 TRUE 설정하거나 Type 종료하도록 설정합니다EC2. 워크플로 작업을 실행하는 데 사용할 컴퓨팅 플릿으로 설정합니다Fleet. 온디맨드 플릿을 선택하거나 프로비저닝된 플릿을 생성하여 선택할 수 있습니다. 자세한 내용은 프로비저닝된 플릿 생성 및 작업에 플릿 또는 컴퓨팅 할당 섹션을 참조하세요.

   워크플로우에 YAML 다음과 비슷한 코드를 추가합니다.

```
Name: MyWorkflow
SchemaVersion: "1.0"
Compute: # Define compute configuration.
  Type: EC2
  Fleet: MyFleet # Optionally, choose an on-demand or provisioned fleet.
  SharedInstance: true # Turn on compute sharing. Default is False.
Actions:
  BuildFirst:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: ...
        ...
```

8. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

예

주제

- 예: 아마존 S3 퍼블리시

예: 아마존 S3 퍼블리시

다음 워크플로 예제는 Amazon Amazon S3 Publish 작업을 두 가지 방법, 즉 입력 아티팩트를 사용한 다음 컴퓨팅 공유를 사용하는 방법을 보여줍니다. 컴퓨팅 공유를 사용하면 캐시된 객체에 액세스할 수 있으므로 입력 아티팩트가 필요하지 않습니다. `WorkflowSource` 또한 Build 작업의 출력 아티팩트는 더 이상 필요하지 않습니다. S3 Publish 작업은 명시적 `DependsOn` 속성을 사용하여 순차적 작업을 유지하도록 구성되어 있습니다. S3 Publish 작업을 실행하려면 빌드 작업이 성공적으로 실행되어야 합니다.

- 컴퓨팅 공유를 사용하지 않는 경우 입력 아티팩트를 사용하고 출력을 후속 작업과 공유해야 합니다.

```
Name: S3PublishUsingInputArtifact
SchemaVersion: "1.0"
Actions:
  Build:
    Identifier: aws/build@v1
    Outputs:
      Artifacts:
        - Name: ArtifactToPublish
          Files: [output.zip]
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: ./build.sh # Build script that generates output.zip
  PublishToS3:
    Identifier: aws/s3-publish@v1
    Inputs:
      Artifacts:
      - ArtifactToPublish
    Environment:
      Connections:
        - Role: codecatalyst-deployment-role
          Name: dev-deployment-role
```

```
      Name: dev-connection
    Configuration:
      SourcePath: output.zip
      DestinationBucketName: dev-bucket
```

- 로 SharedInstance 설정하여 TRUE 컴퓨팅 공유를 사용하는 경우 단일 워크플로 소스를 지정하여 동일한 인스턴스에서 여러 작업을 실행하고 아티팩트를 공유할 수 있습니다. 입력 아티팩트는 필수가 아니며 지정할 수도 없습니다.

```
Name: S3PublishUsingComputeSharing
SchemaVersion: "1.0"
Compute:
  Type: EC2
  Fleet: dev-fleet
  SharedInstance: TRUE
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: ./build.sh # Build script that generates output.zip
  PublishToS3:
    Identifier: aws/s3-publish@v1
    DependsOn:
      - Build
    Environment:
      Connections:
        - Role: codecatalyst-deployment-role
          Name: dev-deployment-role
      Name: dev-connection
    Configuration:
      SourcePath: output.zip
      DestinationBucketName: dev-bucket
```

런타임 환경 이미지 지정

런타임 환경 이미지는 워크플로 작업을 CodeCatalyst 실행하는 Docker 컨테이너입니다. Docker 컨테이너는 선택한 컴퓨팅 플랫폼에서 실행되며, 운영 체제와 워크플로 작업에 필요할 수 있는 추가 도구(예:, Node.js AWS CLI, .tar)를 포함합니다.

기본적으로 워크플로 작업은 에서 제공하고 유지 관리하는 활성 이미지 중 하나에서 실행됩니다. CodeCatalyst 빌드 및 테스트 작업만 사용자 지정 이미지를 지원합니다. 자세한 내용은 액션에 사용자 지정 런타임 환경 Docker 이미지 할당 단원을 참조하십시오.

주제

활성 이미지

액티브 이미지는 사전 설치된 도구가 완벽하게 지원되고 이를 포함하는 런타임 환경 이미지입니다. CodeCatalyst 현재 두 세트의 활성 이미지가 있습니다. 하나는 2024년 3월에 릴리스되고 다른 하나는 2022년 11월에 릴리스됩니다.

액션에서 2024년 3월 이미지를 사용할지 2022년 11월 이미지를 사용할지는 액션에 따라 달라집니다.

- 2024년 3월 26일 또는 그 이후에 워크플로에 추가되는 빌드 및 테스트 액션에는 2024년 3월 이미지를 명시적으로 지정하는 `Container` 섹션이 YAML 정의에 포함됩니다. `Container`섹션을 제거하여 2022년 11월 이미지로 되돌릴 수도 있습니다.
- 2024년 3월 26일 이전에 워크플로에 추가된 빌드 및 테스트 작업은 YAML 정의에 `Container` 섹션이 포함되지 않으므로 2022년 11월 이미지가 사용됩니다. 2022년 11월 이미지는 유지하거나 업그레이드할 수 있습니다. 이미지를 업그레이드하려면 비주얼 편집기에서 작업을 열고 구성 탭을 선택한 다음 런타임 환경 docker 이미지 드롭다운 목록에서 2024년 3월 이미지를 선택합니다. 이렇게 선택하면 적절한 2024년 3월 이미지로 채워진 작업 YAML 정의에 `Container` 섹션이 추가됩니다.
- 다른 모든 작업은 워크플로에 추가된 시기에 관계없이 2022년 11월 이미지를 사용합니다. 2024년 3월 이미지를 사용하도록 이러한 작업을 업그레이드하는 것은 현재 불가능합니다.

주제

- 2024년 3월 이미지
- 2022년 11월 이미지

2024년 3월 이미지

2024년 3월 이미지는 에서 제공한 최신 이미지입니다. CodeCatalyst 컴퓨팅 유형/플릿 조합당 2024년 3월 이미지 1개가 있습니다.

다음 표는 2024년 3월 각 이미지에 설치된 도구를 보여줍니다.

2024년 3월 이미지 도구

도구	CodeCatalyst EC2리눅스용 아마존 x86_64 - **CodeCatalystLinux_x86_64:2024_03**	CodeCatalyst 리눅스 x86_64용 람다 - **CodeCatalystLinuxLambda_x86_64:2024_03**	CodeCatalyst 리눅스 EC2 Arm64용 아마존 - **CodeCatalystLinux_Arm64:2024_03**	CodeCatalyst 리눅스 ARM64용 람다 - **CodeCatalystLinuxLambda_Arm64:2024_**
AWS CLI	2.15.17	2.15.17	2.15.17	2.15.17
AWS 부파일럿 CLI	1.32.1	1.32.1	1.32.1	1.32.1
Docker	24.0.9	N/A	24.0.9	N/A
Docker Compose	2.23.3	N/A	2.23.3	N/A
Git	2.43.0	2.43.0	2.43.0	2.43.0
Go	1.21.5	1.21.5	1.21.5	1.21.5
Gradle	8.5	8.5	8.5	8.5
Java	17로 수정	17로 수정	17로 수정	17로 수정
Maven	3.9.6	3.9.6	3.9.6	3.9.6
Node.js	18.19.0	18.19.0	18.19.0	18.19.0
npm	10.2.3	10.2.3	10.2.3	10.2.3

도구	CodeCatalyst EC2리눅스용 아마존 x86_64 - **CodeCatalystLinux_x86_64:2024_03**	CodeCatalyst 리눅스 x86_64용 람다 - **CodeCatalystLinuxLambda_x86_64:2024_03**	CodeCatalyst 리눅스 EC2 Arm64용 아마존 - **CodeCatalystLinux_Arm64:2024_03**	CodeCatalyst 리눅스 ARM64용 람다 - **CodeCatalystLinuxLambda_Arm64:2024_**
Python	3.9.18	3.9.18	3.9.18	3.9.18
Python3	3.11.6	3.11.6	3.11.6	3.11.6
pip	22.3.1	22.3.1	22.3.1	22.3.1
.NET	8.0.100	8.0.100	8.0.100	8.0.100

2022년 11월 이미지

컴퓨팅 유형/플릿 조합당 2022년 11월 이미지 1개가 있습니다. 프로비저닝된 플릿을 구성한 경우 빌드 작업과 함께 2022년 11월 Windows 이미지도 사용할 수 있습니다.

다음 표는 2022년 11월 각 이미지에 설치된 도구를 보여줍니다.

2022년 11월 이미지 도구

도구	CodeCatalyst EC2리눅스용 아마존 x86_64 - **CodeCatalystLinux_x86_64:2022_11**	CodeCatalyst 리눅스 x86_64용 람다 - **CodeCatalystLinuxLambda_x86_64:2022_11**	CodeCatalyst 리눅스 EC2 Arm64용 아마존 - **CodeCatalystLinux_Arm64:2022_11**	CodeCatalyst 리눅스 ARM64용 람다 - **CodeCatalystLinuxLambda_Arm64:2022_**
AWS CLI	2.15.17	2.15.17	2.15.17	2.15.17
AWS 부파일럿 CLI	0.6.0	0.6.0	N/A	N/A
Docker	23.01	N/A	23.0.1	N/A

도구	CodeCatalyst EC2리눅스용 아마존 x86_64 - **CodeCatalystLinux_x86_64:2022_11**	CodeCatalyst 리눅스 x86_64용 람다 - **CodeCatalystLinuxLambda_x86_64:2022_11**	CodeCatalyst 리눅스 EC2 Arm64용 아마존 - **CodeCatalystLinux_Arm64:2022_11**	CodeCatalyst 리눅스 ARM용 람다 - **CodeCatalystLinuxLambda_Arm64:2022_**
Docker Compose	2.16.0	N/A	2.16.0	N/A
Git	2.40.0	2.40.0	2.39.2	2.39.2
Go	1.20.2	1.20.2	1.20.1	1.20.1
Gradle	8.0.2	8.0.2	8.0.1	8.0.1
Java	17로 수정	17로 수정	17로 수정	17로 수정
Maven	3.9.4	3.9.4	3.9.0	3.9.0
Node.js	16.20.2	16.20.2	16.19.1	16.14.2
npm	8.19.4	8.19.4	8.19.3	8.5.0
Python	3.9.15	2.7.18	3.11.2	2.7.18
Python3	N/A	3.9.15	N/A	3.11.2
pip	22.2.2	22.2.2	23.0.1	23.0.1
.NET	6.0.407	6.0.407	6.0.406	6.0.406

활성 이미지에 필요한 도구가 포함되어 있지 않으면 어떻게 되나요?

에서 제공하는 CodeCatalyst 활성 이미지에 필요한 도구가 없는 경우 다음 두 가지 옵션을 사용할 수 있습니다.

- 필요한 도구가 포함된 사용자 지정 런타임 환경 Docker 이미지를 제공할 수 있습니다. 자세한 내용은 액션에 사용자 지정 런타임 환경 Docker 이미지 할당 단원을 참조하십시오.

> **Note**
>
> 사용자 지정 런타임 환경 Docker 이미지를 제공하려면 사용자 지정 이미지에 Git이 설치되어 있어야 합니다.

- 워크플로의 빌드 또는 테스트 작업에서 필요한 도구를 설치하도록 할 수 있습니다.

  예를 들어 빌드 또는 테스트 작업 YAML 코드의 `Steps` 섹션에 다음 지침을 포함할 수 있습니다.

```
Configuration:
  Steps:
    - Run: ./setup-script
```

  The *`setup-script`* 그러면 다음 스크립트를 실행하여 Node 패키지 관리자 (npm) 를 설치합니다.

```
#!/usr/bin/env bash
echo "Setting up environment"

touch ~/.bashrc
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
source ~/.bashrc
nvm install v16.1.0
source ~/.bashrc
```

  빌드 작업에 대한 자세한 내용은 YAML 을 참조하십시오빌드 및 테스트 액션 YAML.

액션에 사용자 지정 런타임 환경 Docker 이미지 할당

에서 제공하는 Active 이미지를 사용하지 않으려면 사용자 지정 런타임 환경 Docker 이미지를 제공할 수 있습니다. CodeCatalyst 사용자 지정 이미지를 제공하려면 이미지에 Git이 설치되어 있어야 합니다. 이미지는 Docker Hub, Amazon Elastic 컨테이너 레지스트리 또는 모든 공개 리포지토리에 있을 수 있습니다.

사용자 지정 Docker 이미지를 생성하는 방법을 알아보려면 Docker 설명서의 애플리케이션 컨테이너화를 참조하십시오.

다음 지침을 사용하여 사용자 지정 런타임 환경 Docker 이미지를 작업에 할당하세요. 이미지를 지정한 후 작업이 시작되면 컴퓨팅 플랫폼에 이미지를 CodeCatalyst 배포합니다.

> **Note**
>
> AWS CloudFormation 스택 배포, 배포 대상, 작업 등의 작업은 사용자 지정 런타임 환경 Docker 이미지를 지원하지 않습니다. 사용자 지정 런타임 환경 Docker 이미지도 Lambda 컴퓨팅 유형을 지원하지 않습니다. ECS GitHub

Visual

비주얼 편집기를 사용하여 사용자 지정 런타임 환경 Docker 이미지를 할당하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
3. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
4. 편집을 선택합니다.
5. Visual을 선택합니다.
6. 워크플로 다이어그램에서 사용자 지정 런타임 환경 Docker 이미지를 사용할 작업을 선택합니다.
7. 구성 탭을 선택합니다.
8. 하단 근처에 다음 필드를 입력합니다.

   런타임 환경 Docker 이미지 - 선택 사항

   이미지가 저장되는 레지스트리를 지정합니다. 유효한 값으로는 다음이 포함됩니다.

   - `CODECATALYST`(YAML편집자)

     이미지는 CodeCatalyst 레지스트리에 저장됩니다.
   - Docker Hub (비주얼 에디터) 또는 `DockerHub` (YAML에디터)

     이미지는 Docker Hub 이미지 레지스트리에 저장됩니다.
   - 기타 레지스트리 (비주얼 에디터) 또는 `Other` (YAML에디터)

     이미지는 사용자 지정 이미지 레지스트리에 저장됩니다. 공개적으로 사용 가능한 모든 레지스트리를 사용할 수 있습니다.
   - Amazon Elastic 컨테이너 레지스트리 (비주얼 에디터) 또는 `ECR` (YAML에디터)

이미지는 Amazon Elastic 컨테이너 레지스트리 이미지 리포지토리에 저장됩니다. Amazon ECR 리포지토리의 이미지를 사용하려면 이 작업을 수행하려면 Amazon에 대한 액세스 권한이 필요합니다ECR. 이 액세스를 활성화하려면 다음 권한과 사용자 지정 신뢰 정책을 포함하는 IAM역할을 생성해야 합니다. (원하는 경우 권한과 정책을 포함하도록 기존 역할을 수정할 수 있습니다.)

IAM역할의 역할 정책에 다음 권한이 포함되어야 합니다.

- `ecr:BatchCheckLayerAvailability`
- `ecr:BatchGetImage`
- `ecr:GetAuthorizationToken`
- `ecr:GetDownloadUrlForLayer`

IAM역할에는 다음과 같은 사용자 지정 신뢰 정책이 포함되어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

IAM역할 만들기에 대한 자세한 내용은 사용 설명서의 사용자 지정 신뢰 정책을 사용하여 역할 만들기 (콘솔) 를 IAM참조하십시오.

역할을 만든 후에는 환경을 통해 작업에 할당해야 합니다. 자세한 내용은 환경을 액션과 연결하기 단원을 참조하십시오.

ECR이미지 URL, Docker Hub 이미지 또는 이미지 URL

다음 중 하나를 지정하세요.

- CODECATALYST레지스트리를 사용하는 경우 이미지를 다음 활성 이미지 중 하나로 설정합니다.
  - CodeCatalystLinux_x86_64:2024_03
  - CodeCatalystLinux_x86_64:2022_11
  - CodeCatalystLinux_Arm64:2024_03
  - CodeCatalystLinux_Arm64:2022_11
  - CodeCatalystLinuxLambda_x86_64:2024_03
  - CodeCatalystLinuxLambda_x86_64:2022_11
  - CodeCatalystLinuxLambda_Arm64:2024_03
  - CodeCatalystLinuxLambda_Arm64:2022_11
  - CodeCatalystWindows_x86_64:2022_11
- Docker Hub 레지스트리를 사용하는 경우 이미지를 Docker Hub 이미지 이름 및 선택적 태그로 설정합니다.

  예제: postgres:latest
- Amazon ECR 레지스트리를 사용하는 경우 이미지를 Amazon ECR 레지스트리로 설정하십시오URI.

  예제: 111122223333.dkr.ecr.us-west-2.amazonaws.com/codecatalyst-ecs-image-repo
- 사용자 지정 레지스트리를 사용하는 경우 이미지를 사용자 지정 레지스트리에서 예상하는 값으로 설정하십시오.

9. (선택 사항) 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사하려면 [Validate] 를 선택합니다.
10. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

편집기를 사용하여 사용자 지정 런타임 환경 Docker 이미지를 할당하려면 YAML

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.

2. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
3. 편집을 선택합니다.
4. 선택합니다 YAML.
5. 런타임 환경 Docker 이미지를 할당하려는 작업을 찾습니다.
6. 액션에서 `Container` 섹션과 기본 `Registry` 및 `Image` 속성을 추가합니다. 자세한 내용은 작업에 대한`Container`, `Registry` 및 `Image` 속성에 작업 대한 설명을 참조하십시오.
7. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사하십시오.
8. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

예

다음 예제는 워크플로 정의 파일의 작업에 사용자 지정 런타임 환경 Docker 이미지를 할당하는 방법을 보여줍니다.

주제

- 예: 사용자 지정 런타임 환경 Docker 이미지를 사용하여 Amazon에서 Node.js 18에 대한 지원 추가 ECR
- 예: 사용자 지정 런타임 환경 Docker 이미지를 사용하여 Docker Hub에 Node.js 18에 대한 지원 추가

예: 사용자 지정 런타임 환경 Docker 이미지를 사용하여 Amazon에서 Node.js 18에 대한 지원 추가 ECR

다음 예제는 사용자 지정 런타임 환경 Docker 이미지를 사용하여 ECRAmazon에서 Node.js 18에 대한 지원을 추가하는 방법을 보여줍니다.

```
Configuration:
  Container:
    Registry: ECR
    Image: public.ecr.aws/amazonlinux/amazonlinux:2023
```

예: 사용자 지정 런타임 환경 Docker 이미지를 사용하여 Docker Hub에 Node.js 18에 대한 지원 추가

다음 예제는 사용자 지정 런타임 환경 Docker 이미지를 사용하여 Docker Hub와 함께 Node.js 18에 대한 지원을 추가하는 방법을 보여줍니다.

```
Configuration:
  Container:
    Registry: DockerHub
    Image: node:18.18.2
```

소스 리포지토리를 워크플로에 연결

입력 소스라고도 하는 소스는 작업 수행에 필요한 파일을 얻기 위해 워크플로 작업이 연결되는 소스 저장소입니다. 예를 들어, 워크플로 작업을 소스 저장소에 연결하여 응용 프로그램을 빌드하기 위해 응용 프로그램 소스 파일을 가져올 수 있습니다.

CodeCatalyst 워크플로는 다음 소스를 지원합니다.

- CodeCatalyst 소스 리포지토리 — 자세한 내용은 을 참조하십시오. 소스 리포지토리를 사용하여 코드를 저장하고 공동 작업하십시오. CodeCatalyst
- GitHub 리포지토리, Bitbucket 리포지토리 및 GitLab 프로젝트 리포지토리 — 자세한 내용은 을 참조하십시오. 확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst

주제

- 워크플로 파일의 소스 저장소 지정
- 워크플로 작업의 소스 저장소 지정
- 소스 리포지토리 파일 참조
- 'BranchName' 및 'CommitId' 변수

워크플로 파일의 소스 저장소 지정

다음 지침에 따라 워크플로 정의 파일을 저장할 CodeCatalyst 소스 리포지토리를 지정하십시오. GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 지정하려면 대신 확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst 을 참조하십시오.

워크플로 정의 파일이 있는 소스 리포지토리는 레이블로 식별됩니다. `WorkflowSource`

> **Note**
>
> 워크플로 정의 파일을 처음 커밋할 때 워크플로 정의 파일이 있는 소스 저장소를 지정합니다. 이 커밋 후에는 리포지토리와 워크플로 정의 파일이 영구적으로 연결됩니다. 최초 커밋 후 리포지토리를 변경하는 유일한 방법은 다른 리포지토리에서 워크플로를 다시 만드는 것입니다.

워크플로 정의 파일을 저장할 소스 리포지토리를 지정하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로 만들기를 선택하고 워크플로를 생성합니다. 자세한 내용은 워크플로 생성 단원을 참조하십시오.

   워크플로 생성 프로세스 중에 워크플로 정의 파일을 저장할 CodeCatalyst 저장소, 분기 및 폴더를 지정할 수 있습니다.

워크플로 작업의 소스 저장소 지정

다음 지침에 따라 워크플로 작업에 사용할 소스 리포지토리를 지정하십시오. 시작 시 작업은 구성된 소스 리포지토리의 파일을 아티팩트로 번들로 묶고 작업이 실행 중인 런타임 환경 Docker 이미지에 아티팩트를 다운로드한 다음 다운로드한 파일을 사용하여 처리를 완료합니다.

> **Note**
>
> 현재 워크플로 작업 내에서는 소스 리포지토리를 하나만 지정할 수 있습니다. 소스 리포지토리는 워크플로 정의 파일이 있는 소스 리포지토리 (`.codecatalyst/workflows/`디렉터리 또는 하위 디렉터리 중 하나) 입니다. 이 소스 저장소는 레이블로 표시됩니다. `WorkflowSource`

Visual

작업에서 사용할 소스 리포지토리를 지정하려면 (비주얼 에디터)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.

3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로 다이어그램에서 소스를 지정하려는 작업을 선택합니다.
8. 입력을 선택합니다.
9. 소스 - 선택 사항에서 다음을 수행하십시오.

   작업에 필요한 소스 리포지토리를 나타내는 레이블을 지정합니다. 현재 지원되는 유일한 레이블은 `WorkflowSource` 워크플로 정의 파일이 저장되는 소스 저장소를 나타내는 레이블입니다.

   소스를 생략하는 경우 아래에 입력 아티팩트를 하나 이상 지정해야 합니다. *`action-name`*`/Inputs/Artifacts`

   소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.
10. (선택 사항) 커밋하기 전에 워크플로우 YAML 코드를 검증하려면 [Validate] 를 선택합니다.
11. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

작업에서 사용할 소스 리포지토리를 지정하려면 (YAML편집기)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 액션에 다음과 비슷한 코드를 추가합니다.

```
action-name:
 Inputs:
```

```
    Sources:
      - WorkflowSource
```

자세한 내용은 작업에 대한 Sources 속성 설명을 참조하십시오. 워크플로우 YAML 정의

8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

소스 리포지토리 파일 참조

소스 리포지토리에 있는 파일이 있고 워크플로우 작업 중 하나에서 해당 파일을 참조해야 하는 경우 다음 절차를 완료하십시오.

> **Note**
>
> 아티팩트의 파일 참조 섹션도 참조하십시오.

소스 리포지토리에 저장된 파일을 참조하려면

- 파일을 참조하려는 액션에 다음과 비슷한 코드를 추가합니다.

```
Actions:
  My-action:
    Inputs:
      Sources:
        - WorkflowSource
      Configuration:
        Steps:
        - run: cd my-app && cat file1.jar
```

이전 코드에서 액션은 WorkflowSource 소스 리포지토리의 루트에 있는 my-app 디렉터리를 찾아 file1.jar 파일을 찾아 표시합니다.

'BranchName' 및 'CommitId' 변수

워크플로가 실행될 때 CodeCatalyst 소스에서 CommitId 변수를 BranchName 생성하고 설정합니다. 이를 사전 정의된 변수라고 합니다. 이러한 변수에 대한 자세한 내용은 다음 표를 참조하십시오.

워크플로에서 이러한 변수를 참조하는 방법에 대한 자세한 내용은 을 참조하십시오사전 정의된 변수 사용.

키	값
CommitId	워크플로 실행 시작 시점의 저장소 상태를 나타내는 커밋 ID입니다. 예제: `example3819261db00a3ab59468c8b` 또한 예: 미리 정의된 CommitId ""변수 참조 섹션도 참조하세요.
BranchName	워크플로 실행이 시작된 분기의 이름. 예: `main`, `feature/branch` , `test-LiJuan` 또한 예: "BranchName" 사전 정의된 변수 참조 섹션도 참조하세요.

패키지 리포지토리를 워크플로에 연결

패키지는 소프트웨어를 설치하고 종속성을 해결하는 데 필요한 소프트웨어와 메타데이터를 모두 포함하는 번들입니다. CodeCatalyst npm 패키지 형식을 지원합니다.

패키지는 다음과 같이 구성됩니다.

- 이름 (예: 인기 있는 npm 패키지의 이름) `webpack`
- 선택적 네임스페이스 (예: in) `@types` `@types/node`
- 버전 세트 (예:,`1.0.0`,`1.0.1`) `1.0.2`
- 패키지 수준 메타데이터 (예: npm dist 태그)

CodeCatalyst에서는 워크플로의 패키지 리포지토리에 패키지를 게시하고 패키지를 사용할 수 있습니다. CodeCatalyst CodeCatalyst패키지 리포지토리를 사용하여 빌드 또는 테스트 액션을 구성하여 액

션의 npm 클라이언트가 지정된 리포지토리에서 패키지를 푸시하고 풀도록 자동으로 구성할 수 있습니다.

패키지에 대한 자세한 내용은 을 참조하십시오에서 소프트웨어 패키지 게시 및 공유 CodeCatalyst.

> **Note**
>
> 현재 빌드 및 테스트 작업은 CodeCatalyst 패키지 리포지토리를 지원합니다.

주제

- 자습서: 패키지 저장소에서 가져오기
- 워크플로우에서 CodeCatalyst 패키지 리포지토리 지정
- 워크플로 작업에 인증 토큰 사용
- 예: 워크플로의 패키지 리포지토리

자습서: 패키지 저장소에서 가져오기

이 자습서에서는 패키지 저장소에서 종속성을 가져오는 애플리케이션을 실행하는 워크플로를 만드는 방법을 알아봅니다. CodeCatalyst 애플리케이션은 로그에 'Hello World' 메시지를 인쇄하는 간단한 Node.js 앱입니다. CodeCatalyst 애플리케이션에는 lodash npm 패키지라는 단일 종속성이 있습니다. `lodash`패키지는 문자열을 로 변환하는 데 사용됩니다. `hello-world` `Hello World` 이 패키지의 버전 4.17.20을 사용합니다.

응용 프로그램 및 워크플로를 설정한 후에는 공용 외부 레지스트리 (npmjs.com) `lodash` 에서 CodeCatalyst 패키지 저장소로 추가 버전을 가져오는 것을 CodeCatalyst 차단하도록 구성합니다. 그런 다음 의 `lodash` 추가 버전이 성공적으로 차단되었는지 테스트합니다.

이 자습서를 마치면 패키지를 가져오기 위해 워크플로가 내부 및 외부의 CodeCatalyst 패키지 리포지토리와 상호 작용하는 방식을 잘 이해할 수 있을 것입니다. 또한 npm, 패키지 리포지토리, 워크플로, 애플리케이션 파일 사이에서 발생하는 behind-the-scenes 상호 작용도 이해해야 합니다. `package.json`

주제

- 사전 조건
- 1단계: 소스 리포지토리 만들기
- 2단계: CodeCatalyst 및 게이트웨이 패키지 리포지토리 생성

사전 조건

시작하기 전:

- CodeCatalyst 스페이스가 필요합니다. 자세한 내용은 스페이스 만들기 단원을 참조하십시오.
- CodeCatalyst 스페이스에는 다음과 같은 빈 프로젝트가 필요합니다.

  ```
  codecatalyst-package-project
  ```

  처음부터 시작 옵션을 사용하여 이 프로젝트를 만들 수 있습니다.

  자세한 내용은 Amazon에서 빈 프로젝트 생성 CodeCatalyst 단원을 참조하십시오.

1단계: 소스 리포지토리 만들기

이 단계에서는 에서 소스 리포지토리를 생성합니다 CodeCatalyst. 이 저장소에는 자습서의 소스 파일(예: `index.js` 및 `package.json` 파일) 이 저장됩니다.

소스 리포지토리에 대한 자세한 내용은 을 참조하십시오. 소스 리포지토리 생성

소스 리포지토리를 생성하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트로 이동합니다`codecatalyst-package-project`.
3. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
4. 리포지토리 추가를 선택하고 리포지토리 생성을 선택합니다.
5. 리포지토리 이름에 다음을 입력합니다.

   ```
   hello-world-app
   ```

6. 생성(Create)을 선택합니다.

2단계: CodeCatalyst 및 게이트웨이 패키지 리포지토리 생성

이 단계에서는 프로젝트에 패키지 리포지토리를 만들고 이 리포지토리를 역시 CodeCatalyst 프로젝트에 있는 게이트웨이 리포지토리에 연결합니다. CodeCatalyst 나중에 npmjs.com에서 두 `lodash` 리포지토리로 튜토리얼의 종속 항목을 가져옵니다.

게이트웨이 리포지토리는 패키지 리포지토리를 퍼블릭 npmjs.com에 연결하는 '접착제'입니다. CodeCatalyst

패키지 리포지토리에 대한 자세한 내용은 을 참조하십시오. 에서 소프트웨어 패키지 게시 및 공유 CodeCatalyst

> **Note**
>
> 이 자습서에서는 CodeCatalyst 패키지 리포지토리와 게이트웨이 리포지토리라는 용어를 사용하여 다음 CodeCatalyst 절차에서 만드는 두 리포지토리를 가리킵니다.

CodeCatalyst 패키지 및 게이트웨이 리포지토리를 만들려면

1. 탐색 창에서 Packages(패키지)를 선택합니다.
2. 패키지 리포지토리 생성을 선택합니다.
3. 리포지토리 이름에 다음을 입력합니다.

   ```
   codecatalyst-package-repository
   ```

4. + 업스트림 리포지토리 선택을 선택합니다.
5. 게이트웨이 리포지토리를 선택합니다.
6. npm-public-registry-gateway상자에서 [Create] 를 선택합니다.
7. 선택을 선택하세요.
8. 생성(Create)을 선택합니다.

   CodeCatalyst 게이트웨이 리포지토리에 `codecatalyst-package-repository` 연결된 패키지 리포지토리를 만듭니다. 게이트웨이 리포지토리는 npmjs.com 레지스트리에 연결됩니다.

3단계: '헬로 월드' 애플리케이션 만들기

이 단계에서는 'Hello World' Node.js 애플리케이션을 만들고 해당 종속성 () `lodash` 을 게이트웨이 및 패키지 리포지토리로 가져옵니다. CodeCatalyst

애플리케이션을 생성하려면 Node.js 및 관련 클라이언트가 설치된 개발 시스템이 필요합니다. `npm`

이 자습서에서는 개발 환경을 CodeCatalyst 개발 컴퓨터로 사용한다고 가정합니다. CodeCatalyst 개발 환경을 사용할 필요는 없지만 깔끔한 작업 환경을 제공하고 Node.js 및 `npm` 사전 설치되어 있으며 자습서를 마치면 쉽게 삭제할 수 있으므로 이 환경을 사용하는 것이 좋습니다. CodeCatalyst 개발 환경에 대한 자세한 내용은 을 참조하십시오. Dev Environment 생성

다음 지침에 따라 CodeCatalyst 개발 환경을 시작하고 이를 사용하여 'Hello World' 애플리케이션을 만드십시오.

개발 환경을 시작하려면 CodeCatalyst

1. 탐색 창에서 코드를 선택한 다음 개발 환경을 선택합니다.
2. 상단에서 개발 환경 만들기를 선택한 다음 AWS Cloud9 (브라우저에서) 를 선택합니다.
3. 리포지토리가 로 설정되어 `hello-world-app` 있고 기존 브랜치가 로 `main` 설정되어 있는지 확인하십시오. 생성(Create)을 선택합니다.

   개발 환경이 새 브라우저 탭에서 시작되고 리포지토리 (`hello-world-app`) 가 새 브라우저 탭에 복제됩니다.
4. 두 CodeCatalyst 브라우저 탭을 모두 열어 두고 다음 절차로 이동하십시오.

'헬로 월드' Node.js 애플리케이션을 만들려면

1. 개발 환경으로 이동하세요.
2. 터미널 프롬프트에서 `hello-world-app` 소스 리포지토리 루트 디렉터리로 변경합니다.

   ```
   cd hello-world-app
   ```
3. Node.js 프로젝트 초기화:

   ```
   npm init -y
   ```

   초기화하면 의 루트 디렉터리에 `package.json` 파일이 생성됩니다. `hello-world-app`
4. 개발 환경의 npm 클라이언트를 CodeCatalyst 패키지 저장소에 연결합니다.

1. 콘솔로 전환하세요. CodeCatalyst
2. 탐색 창에서 Packages(패키지)를 선택합니다.
3. `codecatalyst-package-repository`를 선택합니다.
4. 리포지토리에 연결을 선택합니다.
5. 토큰 생성을 선택합니다. 개인용 액세스 토큰 (PAT) 이 자동으로 생성됩니다.
6. 복사를 선택하여 명령을 복사합니다.
7. 개발 환경으로 전환하세요.
8. `hello-world-app`디렉터리에 있는지 확인하세요.
9. 명령을 붙여넣습니다. 다음과 비슷해 보입니다.

```
npm set registry=https://packages.us-west-2.codecatalyst.aws/npm/ExampleCompany/
codecatalyst-package-project/codecatalyst-package-repository/ --location project
npm set //packages.us-west-2.codecatalyst.aws/npm/ExampleCompany/codecatalyst-
package-project/hello-world-app/:_authToken=username:token-secret
```

5. `lodash`버전 4.17.20 가져오기:

```
npm install lodash@v4.17.20 --save --save-exact
```

npm은 다음 위치에서 다음 순서로 `lodash` 버전 4.17.20을 찾습니다.

- 개발 환경에서. 여기서는 찾을 수 없습니다.
- CodeCatalyst 패키지 저장소에서 여기서는 찾을 수 없습니다.
- 게이트웨이 리포지토리에서 여기서는 찾을 수 없습니다.
- npmjs.com에서 여기서 찾을 수 있습니다.

npm은 게이트웨이 리포지토리, CodeCatalyst 패키지 리포지토리, 개발 `lodash` 환경으로 가져옵니다.

> **Note**
>
> 4단계에서 npm 클라이언트를 CodeCatalyst 패키지 리포지토리에 연결하지 않았다면 npm은 npmjs.com에서 `lodash` 직접 가져와서 패키지를 어느 리포지토리로도 가져오지 않았을 것입니다.

또한 npm은 package.json 파일을 종속 항목으로 업데이트하고 모든 lodash 종속성을 포함하는 디렉토리를 만듭니다. node_modules lodash

6. 개발 환경으로 lodash 성공적으로 가져온 테스트입니다. 다음을 입력합니다.

```
npm list
```

가져오기에 성공했음을 나타내는 다음 메시지가 나타납니다.

```
`-- lodash@4.17.20
```

7. (선택 사항) 줄을 hello-world-app/package.json 열고 안에 들어 있는지 확인합니다. ***red bold***추가됨:

```
{
  "name": "hello-world-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  dependencies": {
    "lodash": "4.17.20"
  }
}
```

8. 에서 /hello-world-app 다음 index.js 내용으로 라는 파일을 생성합니다.

> **Tip**
>
> 개발 환경의 측면 탐색을 사용하여 이 파일을 만들 수 있습니다.

```
// Importing lodash library
const _ = require('lodash');
```

```
// Input string
const inputString = 'hello-world';

// Transforming the string using lodash
const transformedString = _.startCase(inputString.replace('-', ' '));

// Outputting the transformed string to the console
console.log(transformedString);
```

'lodash'를 게이트웨이 및 패키지 리포지토리로 가져왔는지 테스트하려면 CodeCatalyst

1. 콘솔로 전환하세요. CodeCatalyst
2. 탐색 창에서 Packages(패키지)를 선택합니다.
3. 선택하세요 npm-public-registry-gateway.
4. `lodash`가 표시되는지 확인하세요. 최신 버전 열에 표시됩니다`4.17.20`.
5. 에 대해 이 절차를 반복합니다`codecatalyst-package-repository`. 가져온 패키지를 보려면 브라우저 창을 새로 고쳐야 할 수 있습니다.

개발 환경에서 'Hello World'를 테스트하려면

1. 개발 환경으로 전환하세요.
2. 아직 `hello-world-app` 디렉터리에 있는지 확인한 다음 애플리케이션을 실행하세요.

   ```
   node index.js
   ```

   `Hello World`메시지가 나타납니다. Node.js 는 이전 단계에서 개발 환경에 다운로드한 `lodash` 패키지를 사용하여 애플리케이션을 실행했습니다.

'node_modules' 디렉토리를 무시하고 '헬로 월드'를 커밋하려면

1. `node_modules`디렉터리는 무시하세요. 다음을 입력합니다.

   ```
   echo "node_modules/" >> .gitignore
   ```

   이 디렉터리를 커밋하지 않는 것이 좋습니다. 또한 이 디렉터리를 커밋하면 이 자습서의 이후 단계에 방해가 됩니다.

2. 추가, 커밋, 푸시:

```
git add .
git commit -m "add the Hello World application"
git push
```

'Hello World' 애플리케이션 및 프로젝트 파일이 소스 저장소에 추가됩니다.

4단계: '헬로 월드'를 실행하는 워크플로우 만들기

이 단계에서는 종속성을 사용하여 'Hello World' 애플리케이션을 실행하는 워크플로를 만듭니다. `lodash` 워크플로에는 라는 단일 작업 또는 작업이 포함됩니다. `RunHelloWorldApp` `RunHelloWorldApp`액션에는 다음과 같은 주목할 만한 명령과 섹션이 포함됩니다.

- **`Packages`**

  이 섹션은 작업을 실행할 `npm install` 때 연결해야 하는 CodeCatalyst 패키지 저장소의 이름을 나타냅니다.

- **`- Run: npm install`**

  이 명령은 npm에게 파일에 지정된 종속성을 설치하도록 지시합니다. `package.json` `package.json`파일에 지정된 유일한 종속성은`lodash`. npm은 다음 `lodash` 위치에서 찾습니다.

  - Docker 이미지에서 작업을 실행 중입니다. 여기서는 찾을 수 없습니다.
  - CodeCatalyst 패키지 저장소에서 여기에서 찾을 수 있습니다.

  npm은 찾은 `lodash` 후 작업을 실행하는 Docker 이미지로 가져옵니다.

- **`- Run: npm list`**

  이 명령은 액션을 실행하는 Docker 이미지에 `lodash` 다운로드된 의 버전을 출력합니다.

- **`- Run: node index.js`**

  이 명령은 파일에 지정된 종속성을 사용하여 'Hello World' 애플리케이션을 실행합니다. `package.json`

워크플로 상단 부근의 `RunHelloWorldApp` `aws/build@v1` 식별자로 표시된 대로 액션은 빌드 액션이라는 점을 알 수 있습니다. 빌드 작업에 대한 자세한 내용은 을 참조하십시오워크플로를 사용한 구축.

다음 지침에 따라 CodeCatalyst 패키지 저장소에서 lodash 종속성을 가져온 다음 'Hello World' 애플리케이션을 실행하는 워크플로를 만드십시오.

워크플로 생성 방법

1. 콘솔로 전환하세요. CodeCatalyst
2. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
3. 워크플로 만들기를 선택합니다.
4. 소스 리포지토리의 경우 선택합니다hello-world-app.
5. Branch의 경우 선택하십시오main.

   워크플로 정의 파일은 선택한 소스 리포지토리와 브랜치에 생성됩니다.
6. 생성(Create)을 선택합니다.
7. 상단 YAML근처에서 선택하세요.
8. YAML샘플 코드를 삭제합니다.
9. 다음 YAML 코드를 추가합니다.

```
Name: codecatalyst-package-workflow
SchemaVersion: "1.0"

# Required - Define action configurations.
Actions:
  RunHelloWorldApp:
    # Identifies the action. Do not modify this value.
    Identifier: aws/build@v1
    Compute:
      Type: Lambda
    Inputs:
      Sources:
        - WorkflowSource # This specifies your source repository.
    Configuration:
      Steps:
        - Run: npm install
        - Run: npm list
        - Run: node index.js
      Container: # This specifies the Docker image that runs the action.
        Registry: CODECATALYST
        Image: CodeCatalystLinuxLambda_x86_64:2024_03
    Packages:
      NpmConfiguration:
```

```
        PackageRegistries:
          - PackagesRepository: codecatalyst-package-repository
```

위 코드에서 다음을 대체하십시오.*codecatalyst-package-repository* 에서 2단계: CodeCatalyst 및 게이트웨이 패키지 리포지토리 생성 만든 CodeCatalyst 패키지 리포지토리의 이름으로

이 파일의 속성에 대한 자세한 내용은 를 참조하십시오빌드 및 테스트 액션 YAML.

10. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 YAML 코드가 유효한지 확인합니다.
11. 커밋을 선택합니다.
12. 워크플로 커밋 대화 상자에서 다음을 입력합니다.

    a. 워크플로 파일 이름의 경우 기본값인 을 유지합니다`codecatalyst-package-workflow`.

    b. 커밋 메시지에는 다음을 입력합니다.

    ```
    add initial workflow file
    ```

    c. 리포지토리의 경우 선택합니다 hello-world-app.

    d. 브랜치 이름으로 main을 선택합니다.

    e. 커밋을 선택합니다.

    이제 워크플로가 생성되었습니다.

워크플로를 실행하려면

1. 방금 만든 워크플로 옆의 (`codecatalyst-package-workflow`) 작업을 선택한 다음 실행을 선택합니다.

   워크플로 실행이 시작됩니다.

2. 오른쪽 상단의 녹색 알림에서 실행 링크를 선택합니다. 링크는 다음과 비슷해 보입니다`View Run-1234`.

   실행을 시작한 사람과 RunHelloWorldApp작업을 보여주는 워크플로 다이어그램이 나타납니다.

3. RunHelloWorldApp작업 상자를 선택하여 작업 진행 상황을 확인합니다.
4. 실행이 끝나면 으로 이동하십시오. 5단계: 워크플로우 확인

5단계: 워크플로우 확인

이 단계에서는 워크플로가 종속 항목과 함께 'Hello World' 애플리케이션을 성공적으로 실행했는지 확인합니다. lodash

'Hello World' 애플리케이션이 해당 종속성을 사용하여 실행되었는지 확인하려면

1. 워크플로 다이어그램에서 상자를 선택합니다. RunHelloWorldApp

   로그 메시지 목록이 나타납니다.
2. node index.js로그 메시지를 확장하세요.

   다음과 같은 메시지가 나타납니다.

   ```
   [Container] 2024/04/24 21:15:41.545650 Running command node index.js
   Hello World
   ```

   Hello Word(대신hello-world) 이 나타나면 lodash 종속성이 성공적으로 사용되었음을 나타냅니다.
3. npm list로그를 확장하세요.

   다음과 비슷한 메시지가 나타납니다.

   ```
   ### lodash@4.17.20
   ```

   이 메시지는 워크플로 작업을 실행하는 Docker 이미지에 lodash 버전 4.17.20이 다운로드되었음을 나타냅니다.

6단계: npmjs.com에서 가져오기를 차단합니다.

게이트웨이 및 CodeCatalyst 패키지 리포지토리에 lodash 버전 4.17.20이 추가되었으므로 다른 버전의 가져오기를 차단할 수 있습니다. 차단하면 악성 코드가 포함되어 있을 수 있는 최신 버전 (또는 이전 버전) 을 실수로 가져오는 것을 방지할 수 lodash 있습니다. 자세한 내용은 패키지 원본 제어 편집 및 종속성 대체 공격 단원을 참조하세요.

다음 지침에 따라 게이트웨이 lodash 저장소로의 가져오기를 차단하십시오. 게이트웨이에서 패키지를 차단하면 다운스트림 위치에서도 차단됩니다.

게이트웨이 리포지토리로 가져오는 것을 차단하려면

1. 탐색 창에서 Packages(패키지)를 선택합니다.
2. 선택하세요 npm-publish-registry-gateway.
3. `lodash`를 선택합니다.
4. 상단에서 Origin 컨트롤을 선택합니다.
5. 업스트림에서 차단을 선택합니다.
6. 저장(Save)을 선택합니다.

   이제 npmjs.com에서 게이트웨이 리포지토리 (및 다운스트림 리포지토리 및 컴퓨터) 로 가져오기를 차단했습니다.

7단계: 차단 기능 테스트

이 섹션에서는 설정한 6단계: npmjs.com에서 가져오기를 차단합니다. 차단이 제대로 작동하는지 확인합니다. 먼저 게이트웨이 리포지토리에서 사용할 수 있는 버전 (4.17.2 0) **`lodash`** 대신 버전 4.17.2 1을 요청하도록 'Hello World'를 구성합니다. 그런 다음 애플리케이션이 nmpjs.com에서 버전 4.17.21을 가져올 수 없는지 확인합니다. 이는 차단이 성공했음을 나타냅니다. 최종 테스트로 게이트웨이 저장소로의 가져오기 차단을 해제하고 애플리케이션이 버전 4.17.21을 성공적으로 가져올 수 있는지 확인합니다. `lodash`

다음 절차를 사용하여 차단 기능을 테스트하십시오.

시작하기 전 준비 사항

1. 개발 환경으로 전환하세요.
2. 이전에 CodeCatalyst 콘솔을 사용하여 만든 `codecatalyst-package-workflow.yaml` 파일을 가져오세요.

```
git pull
```

'헬로 월드'가 'lodash' 버전 4.17.21을 요청하도록 구성하려면

1. `/hello-world-app/package.json`를 엽니다.
2. 다음과 같이 버전을 4.17.21로 변경합니다. `lodash` ***red bold***:

```
{
```

```
  "name": "hello-world-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "lodash": "4.17.21"
  }
}
```

이제 `package.json` 파일의 버전 (4.17.21) 과 게이트웨이 및 CodeCatalyst 패키지 리포지토리의 버전 (4.17.20) 간에 불일치가 발생했습니다.

3. 추가, 커밋, 푸시:

```
git add .
git commit -m "update package.json to use lodash 4.17.21"
git push
```

'헬로 월드'가 'lodash' 버전 4.17.21을 가져올 수 없는지 테스트하려면

1. 버전이 일치하지 않는 워크플로를 실행하십시오.

   1. CodeCatalyst 콘솔로 전환하세요.
   2. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
   3. 옆에서 **codecatalyst-package-workflow** [작업] 을 선택한 다음 [실행] 을 선택합니다.

      npm은 종속성을 살펴본 결과 'Hello World'에 버전 `lodash` 4.17.21이 필요함을 확인합니다. npm은 다음 위치에서 다음과 같은 순서로 종속성을 찾습니다. `package.json`

      - Docker 이미지에서 작업을 실행하고 있습니다. 여기서는 찾을 수 없습니다.
      - CodeCatalyst 패키지 저장소에서 여기서는 찾을 수 없습니다.
      - 게이트웨이 리포지토리에서 여기서는 찾을 수 없습니다.
      - npmjs.com에서 여기서 찾을 수 있습니다.

npm이 npmjs.com에서 버전 4.17.21을 찾은 후 게이트웨이 저장소로 가져오려고 하지만 가져오기를 차단하도록 게이트웨이를 설정했기 때문에 가져오기가 수행되지 않습니다. lodash

가져오기가 수행되지 않기 때문에 워크플로가 실패합니다.

2. 워크플로가 실패했는지 확인:

   1. 오른쪽 상단의 녹색 알림에서 실행 링크를 선택합니다. 링크는 다음과 비슷해 보입니다View Run-2345.
   2. 워크플로 다이어그램에서 RunHelloWorldApp상자를 선택합니다.
   3. npm install로그 메시지를 확장합니다.

      다음과 같은 메시지가 나타납니다.

      ```
      [Container] 2024/04/25 17:20:34.995591 Running command npm install
      npm ERR! code ETARGET
      npm ERR! notarget No matching version found for lodash@4.17.21.
      npm ERR! notarget In most cases you or one of your dependencies are requesting
      npm ERR! notarget a package version that doesn't exist.

      npm ERR! A complete log of this run can be found in: /tmp/.npm/
      _logs/2024-05-08T22_03_26_493Z-debug-0.log
      ```

      이 오류는 버전 4.17.21을 찾을 수 없음을 나타냅니다. 차단했기 때문에 이 오류가 발생할 수 있습니다.

npmjs.com에서 가져오기 차단을 해제하려면

1. 탐색 창에서 Packages(패키지)를 선택합니다.
2. npm-publish-registry-gateway선택하세요.
3. lodash를 선택합니다.
4. 상단에서 Origin 컨트롤을 선택합니다.
5. 업스트림에서 허용을 선택합니다.
6. 저장(Save)을 선택합니다.

   이제 의 가져오기 차단을 해제했습니다. lodash

   이제 워크플로에서 버전 4.17.21을 가져올 수 있습니다. lodash

npmjs.com에서 가져오기가 차단 해제되었는지 테스트하려면

1. 워크플로를 다시 실행하세요. 이제 4.17.21의 임포트가 제대로 작동할 것이므로 이번에는 워크플로가 성공해야 합니다. 워크플로를 다시 실행하려면:

   1. CI/CD를 선택한 다음 워크플로를 선택합니다.
   2. `codecatalyst-package-workflow`다음으로 작업을 선택하고 실행을 선택합니다.
   3. 오른쪽 상단의 녹색 알림에서 실행 링크를 선택합니다. 링크는 다음과 비슷해 보입니다`View Run-3456`.

      실행을 시작한 사람과 RunHelloWorldApp작업을 보여주는 워크플로 다이어그램이 나타납니다.
   4. RunHelloWorldApp작업 상자를 선택하여 작업 진행 상황을 확인합니다.
   5. `npm list`로그 메시지를 확장하고 다음과 비슷한 메시지가 나타나는지 확인합니다.

      ```
      ### lodash@4.17.21
      ```

      이 메시지는 `lodash` 버전 4.17.21이 다운로드되었음을 나타냅니다.
2. 버전 4.17.21을 사용자 및 게이트웨이 리포지토리로 가져왔는지 확인하십시오. CodeCatalyst

   1. 탐색 창에서 Packages(패키지)를 선택합니다.
   2. 선택합니다. npm-public-registry-gateway
   3. 버전을 `lodash` 찾아서 확인하십시오`4.17.21`.

      > **Note**
      >
      > 버전 4.17.20은 이 페이지에 나열되어 있지 않지만 상단에서 버전을 선택한 다음 선택하여 `lodash` 찾을 수 있습니다.
   4. 이 단계를 반복하여 버전 4.17.21을 (로) 가져왔는지 확인하십시오. `codecatalyst-package-repository`

정리

요금이 부과되지 않도록 이 자습서에서 사용된 파일 및 서비스를 정리하세요.

패키지 튜토리얼을 정리하려면

1. `codecatalyst-package-project`다음을 삭제하십시오.

   a. 아직 `codecatalyst-package-project` 프로젝트에 참여하지 않았다면 CodeCatalyst 콘솔에서 해당 프로젝트로 이동하세요.
   b. 탐색 창에서 프로젝트 설정을 선택합니다.
   c. 프로젝트 삭제를 선택하고 를 입력한 다음 **`delete`** 프로젝트 삭제를 선택합니다.

      CodeCatalyst 소스, 게이트웨이, CodeCatalyst 패키지 리포지토리를 포함한 모든 프로젝트 리소스를 삭제합니다. 개발 환경도 삭제됩니다.

2. PAT토큰 삭제:

   a. 오른쪽에서 사용자 이름을 선택한 다음 내 설정을 선택합니다.
   b. 개인용 액세스 토큰에서 이 자습서에서 생성한 토큰을 선택하고 삭제를 선택합니다.

이 자습서에서는 CodeCatalyst 패키지 저장소에서 종속성을 가져오는 애플리케이션을 실행하는 워크플로를 만드는 방법을 배웠습니다. 또한 패키지가 게이트웨이 및 패키지 리포지토리에 들어오는 것을 차단 및 차단 해제하는 방법도 배웠습니다. CodeCatalyst

워크플로우에서 CodeCatalyst 패키지 리포지토리 지정

CodeCatalyst에서는 워크플로의 빌드 및 테스트 작업에 CodeCatalyst 패키지 저장소를 추가할 수 있습니다. 패키지 저장소는 npm과 같은 패키지 형식으로 구성해야 합니다. 선택한 패키지 리포지토리에 대한 일련의 범위를 포함하도록 선택할 수도 있습니다.

다음 지침에 따라 워크플로 작업에 사용할 패키지 구성을 지정하십시오.

Visual

작업에서 사용할 패키지 구성을 지정하려면 (비주얼 에디터)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.

6. Visual을 선택합니다.
7. 워크플로 다이어그램에서 패키지 리포지토리로 구성하는 데 사용할 빌드 또는 테스트 작업을 선택합니다.
8. 패키지를 선택합니다.
9. 구성 추가 드롭다운 메뉴에서 워크플로 작업에 사용할 패키지 구성을 선택합니다.
10. 패키지 리포지토리 추가를 선택합니다.
11. 패키지 저장소 드롭다운 메뉴에서 작업에 사용할 CodeCatalyst 패키지 저장소의 이름을 지정합니다.

    패키지 리포지토리에 대한 자세한 내용은 을 참조하십시오. 패키지 리포지토리
12. (선택 사항) Scopes - 선택 사항에서 패키지 레지스트리에 정의하려는 범위 시퀀스를 지정합니다.

    범위를 정의할 때 지정된 패키지 저장소는 나열된 모든 범위의 레지스트리로 구성됩니다. npm 클라이언트를 통해 범위가 지정된 패키지를 요청하면 기본값 대신 해당 리포지토리를 사용합니다. 각 스코프 이름 앞에 “@”를 붙여야 합니다.

    를 생략하면 Scopes 지정된 패키지 저장소가 작업에 사용되는 모든 패키지의 기본 레지스트리로 구성됩니다.

    범위에 대한 자세한 내용은 범위 지정 패키지를 참조하십시오패키지 네임스페이스.
13. 추가를 선택합니다.
14. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로의 YAML 코드를 검증하십시오.
15. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

작업에 사용할 패키지 구성을 지정하려면 (YAML편집자)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.

6. 선택합니다 YAML.
7. 빌드 또는 테스트 작업에 다음과 비슷한 코드를 추가합니다.

```
action-name:
 Configuration:
    Packages:
        NpmConfiguration:
          PackageRegistries:
            - PackagesRepository: package-repository
              Scopes:
                - "@scope"
```

자세한 빌드 및 테스트 액션 YAML 내용은 해당 작업의 Packages 속성 설명을 참조하십시오.

8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

워크플로 작업에 인증 토큰 사용

워크플로 작업에서 제공하는 토큰을 사용하여 패키지 리포지토리로 CodeCatalyst 인증하도록 패키지 관리자를 수동으로 구성할 수 있습니다. CodeCatalyst 이 토큰을 작업 시 참조할 수 있는 환경 변수로 사용할 수 있도록 합니다.

환경 변수	값
CATALYST_MACHINE_RESOURCE_NAME	인증 토큰의 사용자 ID.
CATALYST_PACKAGES_AUTHORIZATION_TOKEN	인증 토큰의 값입니다.

> **Note**
>
> 단, 이러한 환경 변수는 인증 토큰을 내보내도록 작업을 구성한 경우에만 입력된다는 점에 유의하십시오.

워크플로 작업에 권한 부여 토큰을 사용하려면 다음 지침을 따르십시오.

Visual

내보낸 인증 토큰을 작업과 함께 사용하려면 (비주얼 에디터)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로 다이어그램에서 패키지 리포지토리로 구성하는 데 사용할 빌드 또는 테스트 작업을 선택합니다.
8. 패키지를 선택합니다.
9. 승인 토큰 내보내기를 켜십시오.

YAML

내보낸 인증 토큰을 작업과 함께 사용하려면 (YAML편집자)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 빌드 또는 테스트 작업에 다음과 비슷한 코드를 추가합니다.

```
Actions:
  action-name:
    Packages:
      ExportAuthorizationToken: true
```

Steps섹션의 $CATALYST_MACHINE_RESOURCE_NAME 및 $CATALYST_PACKAGES_AUTHORIZATION_TOKEN 환경 변수를 참조할 수 있습니다YAML. 자세한 정보는 예: 인증을 pip 위해 수동으로 구성 CodeCatalyst 단원을 참조하세요.

8. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로의 YAML 코드를 검증하십시오.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

예: 워크플로의 패키지 리포지토리

다음 예제는 워크플로 정의 파일에서 패키지를 참조하는 방법을 보여줍니다.

주제

- 예: 를 사용하여 패키지 정의하기 NpmConfiguration
- 예: 기본 레지스트리 재정의
- 예: 패키지 레지스트리의 범위 재정의
- 예: 인증을 pip 위해 수동으로 구성 CodeCatalyst

예: 를 사용하여 패키지 정의하기 **NpmConfiguration**

다음 예제는 워크플로 정의 파일에서 패키지를 정의하는 방법을 보여줍니다. NpmConfiguration

```
Actions:
  Build:
  Identifier: aws/build-beta@v1
  Configuration:
    Packages:
        NpmConfiguration:
          PackageRegistries:
            - PackagesRepository: main-repo
            - PackagesRepository: scoped-repo
              Scopes:
                - "@scope1"
```

이 예제는 npm 클라이언트를 다음과 같이 구성합니다.

```
default: main-repo
@scope1: scoped-repo
```

이 예제에는 리포지토리가 두 개 정의되어 있습니다. 기본 레지스트리는 범위 없이 정의된 main-repo 대로 설정됩니다. @scope1범위는 PackageRegistries for에서 구성됩니다scoped-repo.

예: 기본 레지스트리 재정의

다음 예제는 기본 레지스트리를 재정의하는 방법을 보여줍니다.

```
NpmConfiguration:
  PackageRegistries:
    - PackagesRepository: my-repo-1
    - PackagesRepository: my-repo-2
    - PackagesRepository: my-repo-3
```

이 예제는 npm 클라이언트를 다음과 같이 구성합니다.

```
default: my-repo-3
```

기본 리포지토리를 여러 개 지정하면 마지막 리포지토리가 우선합니다. 이 예제에서 마지막으로 나열된 저장소는 입니다. 즉my-repo-3, npm이 연결할 저장소입니다. my-repo-3 이는 리포지토리 및 를 재정의합니다. my-repo-1 my-repo-2

예: 패키지 레지스트리의 범위 재정의

다음 예제는 패키지 레지스트리의 범위를 재정의하는 방법을 보여줍니다.

```
NpmConfiguration:
  PackageRegistries:
    - PackagesRepository: my-default-repo
    - PackagesRepository: my-repo-1
      Scopes:
        - "@scope1"
        - "@scope2"
    - PackagesRepository: my-repo-2
      Scopes:
        - "@scope2"
```

이 예제는 npm 클라이언트를 다음과 같이 구성합니다.

```
default: my-default-repo
@scope1: my-repo-1
@scope2: my-repo-2
```

오버라이드 범위를 포함하면 마지막 리포지토리가 우선합니다. 이 예제에서 해당 범위를 `@scope2` 마지막으로 구성한 시간은 `PackageRegistries` for입니다. `my-repo-2` 이렇게 하면 `@scope2` 구성된 범위가 재정의됩니다. `my-repo-1`

예: 인증을 **`pip`** 위해 수동으로 구성 CodeCatalyst

다음 예제는 빌드 작업에서 CodeCatalyst 권한 부여 환경 변수를 참조하는 방법을 보여줍니다.

```
Actions:
  Build:
    Identifier: aws/build@v1.0.0
    Configuration:
      Steps:
        - Run: pip config set global.index-url https://$CATALYST_MACHINE_RESOURCE_NAME:
$CATALYST_PACKAGES_AUTHORIZATION_TOKEN@codecatalyst.aws/pypi/my-space/my-project/my-
repo/simple/
    Packages:
      ExportAuthorizationToken: true
```

워크플로를 사용하여 Lambda 함수 호출

이 섹션에서는 워크플로를 사용하여 AWS Lambda 함수를 호출하는 방법을 설명합니다. CodeCatalyst 이 작업을 수행하려면 워크플로에 AWS Lambda 호출 작업을 추가해야 합니다. AWS Lambda 호출 작업은 지정한 Lambda 함수를 호출합니다.

함수 호출 외에도 AWS Lambda 호출 작업은 Lambda 함수에서 수신한 응답 페이로드의 각 최상위 키를 워크플로 출력 변수로 변환합니다. 그러면 후속 워크플로 작업에서 이러한 변수를 참조할 수 있습니다. 모든 최상위 키를 변수로 변환하지 않으려면 필터를 사용하여 정확한 키를 지정할 수 있습니다. 자세한 내용은 의 ResponseFilters AWS Lambda '호출' 액션 YAML 속성 설명을 참조하십시오.

이 작업을 사용하는 경우

Lambda 함수에 캡슐화되고 Lambda 함수에 의해 수행되는 기능을 워크플로에 추가하려는 경우 이 작업을 사용하십시오.

예를 들어, 애플리케이션 빌드를 시작하기 전에 워크플로에서 Slack 채널에 `Build started` 알림을 보내도록 할 수 있습니다. 이 경우 워크플로에는 Lambda를 AWS Lambda 호출하여 Slack 알림을 보내는 호출 작업과 애플리케이션을 빌드하기 위한 빌드 작업이 포함됩니다.

또 다른 예로, 애플리케이션을 배포하기 전에 워크플로에서 애플리케이션에 대한 취약성 스캔을 수행하기를 원할 수 있습니다. 이 경우 빌드 작업을 사용하여 애플리케이션을 빌드하고, AWS Lambda 호

출 작업을 사용하여 Lambda를 호출하여 취약성을 검색하고, 배포 작업을 사용하여 스캔한 애플리케이션을 배포합니다.

주제

예: Lambda 함수 호출

다음 예제 워크플로는 배포 작업과 함께 AWS Lambda 호출 작업을 포함합니다. 워크플로는 배포가 시작되었음을 알리는 Slack 알림을 보낸 다음 템플릿을 AWS 사용하여 애플리케이션을 배포합니다. AWS CloudFormation 워크플로는 순차적으로 실행되는 다음과 같은 구성 요소로 구성됩니다.

- 트리거 - 이 트리거는 소스 리포지토리에 변경 내용을 푸시할 때 워크플로가 자동으로 실행됩니다. 트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.
- AWS Lambda 호출 작업 (`LambdaNotify`) — 트리거 시 이 작업은 지정된 AWS 계정 및 리전 (, 및) 에서 `Notify-Start` Lambda 함수를 호출합니다. `my-aws-account` `us-west-2` 호출 시 Lambda 함수는 배포가 시작되었음을 알리는 Slack 알림을 보냅니다.
- AWS CloudFormation 스택 배포 작업 (`Deploy`) — AWS Lambda 호출 작업이 완료되면 스택 배포 작업은 템플릿 (`cfn-template.yml`) 을 실행하여 애플리케이션 AWS CloudFormation 스택을 배포합니다. AWS CloudFormation 스택 배포 작업에 대한 자세한 내용은 을 참조하십시오스택 배포 AWS CloudFormation.

> **Note**
>
> 다음 워크플로 예제는 설명을 위한 것으로, 추가 구성 없이는 작동하지 않습니다.

> **Note**
>
> 다음 YAML 코드에서는 원하는 경우 `Connections:` 섹션을 생략할 수 있습니다. 이러한 섹션을 생략하는 경우 환경의 기본 역할 필드에 지정된 IAM 역할에AWS Lambda 호출 및 배포

AWS CloudFormation 스택 작업에 필요한 권한 및 신뢰 정책이 포함되어 있는지 확인해야 합니다. 기본 IAM 역할을 사용하여 환경을 설정하는 방법에 대한 자세한 내용은 을 참조하십시오. 환경 생성 AWS Lambda 호출 및 배포 AWS CloudFormation 스택 작업에 필요한 권한 및 신뢰 정책에 대한 자세한 내용은 AWS Lambda '호출' 액션 YAML 및 AWS CloudFormation '스택 배포' 작업 YAML 의 `Role` 속성 설명을 참조하십시오.

```
Name: codecatalyst-lamda-invoke-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  LambdaNotify:
    Identifier: aws/lambda-invoke@v1
    Environment:
      Name: my-production-environment
      Connections:
        - Name: my-aws-account
          Role: codecatalyst-lambda-invoke-role
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Function: Notify-Start
      AWSRegion: us-west-2

  Deploy:
    Identifier: aws/cfn-deploy@v1
    Environment:
      Name: my-production-environment
      Connections:
        - Name: my-aws-account
          Role: codecatalyst-deploy-role
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      name: my-application-stack
      region: us-west-2
```

```
      role-arn: arn:aws:iam::111122223333:role/StackRole
      template: ./cfn-template.yml
      capabilities: CAPABILITY_IAM,CAPABILITY_AUTO_EXPAND
```

AWS Lambda '호출' 작업 추가

워크플로에 AWS Lambda 호출 작업을 추가하려면 다음 지침을 따르십시오.

전제 조건

시작하기 전에 AWS Lambda 함수 및 관련 Lambda 실행 역할이 에서 준비되고 사용 가능한지 확인하십시오. AWS자세한 내용은 개발자 안내서의 Lambda 실행 역할 주제를 참조하십시오.AWS Lambda

Visual

비주얼 에디터를AWS Lambda 사용하여 '호출' 작업을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. AWS Lambda 호출 작업을 검색하고 다음 중 하나를 수행하십시오.

   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - AWS Lambda 호출을 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 소스 보기를 선택하여 액션의 소스 코드를 확인합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

10. 입력, 구성 및 출력 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오AWS Lambda '호출' 액션 YAML. 이 참조는 편집기와 시각적 편집기에 모두 나타나는 각 필드 (및 해당 YAML 속성 값) 에 대한 자세한 정보를 제공합니다. YAML
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

편집기를 사용하여 'AWS Lambda 호출' 작업을 추가하려면 YAML

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. AWS Lambda 호출 작업을 검색하고 다음 중 하나를 수행하십시오.
   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - AWS Lambda 호출을 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 소스 보기를 선택하여 액션의 소스 코드를 확인합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.
10. 필요에 따라 YAML 코드의 속성을 수정하십시오. 사용 가능한 각 속성에 대한 설명은 에 나와 AWS Lambda '호출' 액션 YAML 있습니다.

11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

AWS Lambda '호출' 변수

기본적으로 AWS Lambda 호출 작업은 Lambda 응답 페이로드의 최상위 키당 하나의 변수를 생성합니다.

예를 들어, 응답 페이로드가 다음과 같은 경우:

```
responsePayload = {
  "name": "Saanvi",
  "location": "Seattle",
  "department": {
    "company": "Amazon",
    "team": "AWS"
  }
}
```

... 그러면 액션은 다음과 같은 변수를 생성합니다.

키	값
name	Saanvi
location	시애틀
department	{"회사": “아마존”, “팀”: "AWS“}

> **Note**
>
> `ResponseFilters`YAML속성을 사용하여 생성되는 변수를 변경할 수 있습니다. 자세한 내용은 AWS Lambda '호출' 액션 YAML의 ResponseFilters를 참조하십시오.

런타임에 'AWS Lambda invoke' 작업을 통해 생성되고 설정된 변수를 사전 정의된 변수라고 합니다.

워크플로에서 이러한 변수를 참조하는 방법에 대한 자세한 내용은 을 참조하십시오. 사전 정의된 변수 사용

AWS Lambda '호출' 액션 YAML

다음은 AWS Lambda 호출 작업의 YAML 정의입니다. 이 작업을 사용하는 방법을 알아보려면 을 참조하십시오워크플로를 사용하여 Lambda 함수 호출.

이 작업 정의는 광범위한 워크플로 정의 파일 내에 섹션으로 존재합니다. 이 파일에 대한 자세한 내용은 워크플로우 YAML 정의을 참조합니다.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

```
# The workflow definition starts here.
# See ### ## for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  LambdaInvoke_nn:
    Identifier: aws/lambda-invoke@v1
    DependsOn:
      - dependent-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
        - request-payload
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
```

```
        - Name: variable-name-2
          Value: variable-value-2
    Environment:
      Name: environment-name
      Connections:
        - Name: account-connection-name
          Role: iam-role-name
    Configuration:
      Function: my-function|function-arn
      AWSRegion: us-west-2
      # Specify RequestPayload or RequestPayloadFile, but not both.
      RequestPayload: '{"firstname": "Li", lastname: "Jean", "company": "ExampleCo",
"team": "Development"}'
      RequestPayloadFile: my/request-payload.json
      ContinueOnError: true|false
      LogType: Tail|None
      ResponseFilters: '{"name": ".name", "company": ".department.company"}'
    Outputs:
      Artifacts:
        - Name: lambda_artifacts
          Files:
            - "lambda-response.json"
```

LambdaInvoke

(필수)

액션의 이름을 지정합니다. 모든 작업 이름은 워크플로 내에서 고유해야 합니다. 액션 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 액션 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

기본값: `Lambda_Invoke_Action_Workflow_nn`.

해당 UI: 구성 탭/ 작업 이름

Identifier

(*LambdaInvoke*/Identifier)

(필수)

작업을 식별합니다. 버전을 변경하려는 경우가 아니면 이 속성을 변경하지 마십시오. 자세한 내용은 사용할 액션 버전 지정 단원을 참조하십시오.

기본값: `aws/lambda-invoke@v1`.

해당 UI: 워크플로 다이어그램/ LambdaInvoke _nn/ aws/lambda-invoke @v1 레이블

DependsOn

(*LambdaInvoke*/DependsOn)

(선택 사항)

이 작업을 실행하기 위해 성공적으로 실행되어야 하는 작업, 작업 그룹 또는 게이트를 지정하십시오.

'종속 조건' 기능에 대한 자세한 내용은 을 참조하십시오. 시퀀싱 액션

해당 UI: 입력 탭/ 기준 - 선택 사항

Compute

(*LambdaInvoke*/Compute)

(선택 사항)

워크플로 작업을 실행하는 데 사용되는 컴퓨팅 엔진. 워크플로 수준 또는 작업 수준에서 컴퓨팅을 지정할 수 있지만 둘 다에서 지정할 수는 없습니다. 워크플로 수준에서 지정된 경우 컴퓨팅 구성은 워크플로에 정의된 모든 작업에 적용됩니다. 워크플로 수준에서는 동일한 인스턴스에서 여러 작업을 실행할 수도 있습니다. 자세한 내용은 작업 간 컴퓨팅 공유 단원을 참조하십시오.

해당 UI: 없음

Type

(*LambdaInvoke*/Compute/Type)

(포함된 Compute 경우 필수)

컴퓨팅 엔진 유형. 다음 값 중 하나를 사용할 수 있습니다.

- EC2(비주얼 에디터) 또는 `EC2` (YAML에디터)

  작업 실행 중 유연성을 위해 최적화되었습니다.
- Lambda (비주얼 에디터) 또는 (에디터) `Lambda` YAML

  작업 시작 속도를 최적화했습니다.

컴퓨팅 유형에 대한 자세한 정보는 컴퓨팅 유형을 참고하세요.

해당 UI: 구성 탭/ 컴퓨팅 유형

Fleet

(*LambdaInvoke*/Compute/Fleet)

(선택 사항)

워크플로 또는 워크플로 작업을 실행할 시스템 또는 플릿을 지정합니다. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다. 온디맨드 플릿의 예:,. `Linux.x86-64.Large` `Linux.x86-64.XLarge` 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

생략된 경우 기본값은 `Fleet` 입니다. `Linux.x86-64.Large`

해당 UI: 구성 탭/ 컴퓨팅 플릿

Timeout

(*LambdaInvoke*/Timeout)

(필수)

작업이 CodeCatalyst 종료되기 전에 작업을 실행할 수 있는 시간을 분 (편집기) 또는 시간과 분 (비주얼 편집기) 단위로 지정합니다. YAML 최소값은 5분이고 최대값은 에 설명되어 의 워크플로우 할당량 CodeCatalyst 있습니다. 기본 타임아웃은 최대 타임아웃과 동일합니다.

해당 UI: 구성 탭/ 타임아웃 - 선택 사항

Inputs

(*LambdaInvoke*/Inputs)

(필수)

이 `Inputs` 섹션에서는 워크플로 실행 중에 AWS Lambda 호출 작업에 필요한 데이터를 정의합니다.

> **Note**
>
> AWS Lambda 호출 작업당 하나의 입력 (소스 또는 아티팩트) 만 허용됩니다. 변수는 이 총계에 포함되지 않습니다.

해당 UI: 입력 탭

Sources

(*LambdaInvoke*/Inputs/Sources)

(제공된 RequestPayloadFile경우 필수)

요청 페이로드 JSON 파일을 AWS Lambda 호출 작업에 전달하고 이 페이로드 파일이 소스 리포지토리에 저장되어 있는 경우 해당 소스 리포지토리의 레이블을 지정하십시오. 현재 지원되는 레이블은 뿐입니다. `WorkflowSource`

요청 페이로드 파일이 소스 리포지토리에 포함되어 있지 않은 경우 해당 파일은 다른 작업에서 생성된 아티팩트에 있어야 합니다.

페이로드 파일에 대한 자세한 내용은 을 참조하십시오. RequestPayloadFile

> **Note**
>
> 페이로드 파일을 지정하는 대신 속성을 사용하여 페이로드 JSON 코드를 작업에 직접 추가할 수 있습니다. `RequestPayload` 자세한 내용은 RequestPayload 단원을 참조하십시오.

소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

해당 UI: 입력 탭/ 소스 - 선택 사항

Artifacts - input

(*LambdaInvoke*/Inputs/Artifacts)

(제공된 RequestPayloadFile경우 필수)

요청 페이로드 JSON 파일을 AWS Lambda 호출 작업에 전달하고 이 페이로드 파일이 이전 작업의 출력 아티팩트에 포함되어 있는 경우 여기에 해당 아티팩트를 지정하십시오.

페이로드 파일에 대한 자세한 내용은 을 참조하십시오. RequestPayloadFile

> **Note**
>
> 페이로드 파일을 지정하는 대신 속성을 사용하여 페이로드 JSON 코드를 작업에 직접 추가할 수 있습니다. `RequestPayload` 자세한 내용은 RequestPayload 단원을 참조하십시오.

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 구성 탭/ 아티팩트 - 선택 사항

Variables - input

(*LambdaInvoke*/Inputs/Variables)

(선택 사항)

작업에 사용할 수 있도록 하려는 입력 변수를 정의하는 이름/값 쌍의 시퀀스를 지정합니다. 변수 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 변수 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

예제를 비롯한 변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

해당 UI: 입력 탭/ 변수 - 선택 사항

Environment

(*LambdaInvoke*/Environment)

(필수)

작업에 사용할 CodeCatalyst 환경을 지정합니다. 작업은 선택한 환경에 VPC 지정된 AWS 계정 및 선택적 Amazon에 연결됩니다. 작업은 환경에 지정된 기본 IAM 역할을 사용하여 Amazon에 연결하고 Amazon VPC 연결에 지정된 IAM 역할을 사용하여 Amazon에 연결합니다VPC. AWS 계정

> **Note**
>
> 기본 IAM 역할에 작업에 필요한 권한이 없는 경우 다른 역할을 사용하도록 작업을 구성할 수 있습니다. 자세한 내용은 액션의 IAM 역할 변경하기 단원을 참조하십시오.

환경에 대한 자세한 내용은 및 에 AWS 계정 배포 VPCs 및 을 참조하십시오환경 생성.

해당 UI: 구성 탭/ 환경

Name

(*LambdaInvoke*/Environment/Name)

(포함된 Environment 경우 필수)

작업에 연결할 기존 환경의 이름을 지정합니다.

해당 UI: 구성 탭/ 환경

Connections

(*LambdaInvoke*/Environment/Connections)

(액션의 최신 버전에서는 선택 사항, 이전 버전에서는 필수)

작업에 연결할 계정 연결을 지정합니다. 에서 최대 1개의 계정 연결을 지정할 수 `Environment` 있습니다.

계정 연결을 지정하지 않은 경우:

- 작업은 CodeCatalyst 콘솔 환경에 지정된 AWS 계정 연결 및 기본 IAM 역할을 사용합니다. 환경에 계정 연결 및 기본 IAM 역할을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.
- 기본 IAM 역할에는 작업에 필요한 정책 및 권한이 포함되어야 합니다. 이러한 정책 및 권한이 무엇인지 확인하려면 작업 YAML 정의 설명서에서 역할 속성에 대한 설명을 참조하십시오.

계정 연결에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정. 환경에 계정 연결을 추가하는 방법에 대한 자세한 내용은 을 참조하십시오환경 생성.

해당 UI: 액션 버전에 따라 다음 중 하나:

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Name

(*LambdaInvoke*/Environment/Connections/Name)

(포함된 경우 필수Connections)

계정 연결 이름을 지정합니다.

해당 UI: 액션 버전에 따라 다음 중 하나:

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 계정 연결AWS

Role

(*LambdaInvoke*/Environment/Connections/Role)

(포함된 경우 필수Connections)

호출 작업이 Lambda 함수에 AWS 액세스하고 AWS Lambda 호출하는 데 사용하는 IAM 역할 이름을 지정합니다. CodeCatalyst 스페이스에 역할을 추가했는지, 역할에 다음 정책이 포함되어 있는지 확인하십시오.

IAM역할을 지정하지 않으면 CodeCatalyst 콘솔 환경에 나열된 기본 IAM 역할이 작업에 사용됩니다. 환경에서 기본 역할을 사용하는 경우 해당 역할에 다음 정책이 적용되는지 확인하십시오.

- 다음 권한 정책:

> **Warning**
>
> 권한을 다음 정책에 표시된 권한으로 제한하십시오. 더 광범위한 권한이 있는 역할을 사용하면 보안 위험이 발생할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "arn:aws:lambda:aws-region:aws-account:function:function-
name"
        }
    ]
```

```
}
```

- 다음과 같은 사용자 지정 신뢰 정책:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

> **Note**
>
> 원하는 경우 이 작업에 CodeCatalystWorkflowDevelopmentRole-*spaceName* 역할을 사용할 수 있습니다. 이에 대한 자세한 내용은 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기 섹션을 참조하세요. CodeCatalystWorkflowDevelopmentRole-*spaceName*역할에는 보안 위험을 초래할 수 있는 전체 액세스 권한이 있다는 점을 이해하세요. 보안이 덜 우려되는 자습서 및 시나리오에서만 이 역할을 사용하는 것이 좋습니다.

해당 UI: 액션 버전에 따라 다음 중 하나가 표시됩니다.

- (최신 버전) 구성 탭/환경/내용 *my-environment*? /쓰리 닷 메뉴/ 역할 전환
- (이전 버전) 구성 탭/'환경/계정/역할'/ 역할

Configuration

(*LambdaInvoke*/Configuration)

(필수)

작업의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭

Function

(*LambdaInvoke*/Configuration/Function)

(필수)

이 작업이 호출할 AWS Lambda 함수를 지정합니다. 함수 이름 또는 Amazon 리소스 이름 (ARN) 을 지정할 수 있습니다. 이름 또는 Lambda ARN 콘솔에서 찾을 수 있습니다.

> **Note**
>
> Lambda 함수가 있는 AWS 계정은 에 지정된 계정과 다를 수 있습니다. `Connections:`

해당 UI: 구성 탭/ 함수

AWSRegion

(*LambdaInvoke*/Configuration/AWSRegion)

(필수)

AWS Lambda 함수가 있는 AWS 지역을 지정합니다. 지역 코드 목록은 의 지역별 엔드포인트를 참조하십시오. AWS 일반 참조

해당 UI: 구성 탭/ 대상 버킷 - 선택 사항

RequestPayload

(*LambdaInvoke*/Configuration/RequestPayload)

(선택 사항)

요청 페이로드를 AWS Lambda 호출 작업에 전달하려면 여기에 요청 페이로드를 형식으로 지정하십시오. JSON

예제 요청 페이로드:

```
'{ "key": "value" }'
```

요청 페이로드를 Lambda 함수로 전달하지 않으려면 이 속성을 생략하십시오.

> **Note**
>
> `RequestPayload` 또는 `RequestPayloadFile`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

요청 페이로드에 대한 자세한 내용은 참조의 Invoke 주제를 참조하십시오.AWS Lambda API

해당 UI: 구성 탭/ 요청 페이로드 - 선택 사항

RequestPayloadFile

(*LambdaInvoke*/Configuration/RequestPayloadFile)

(선택 사항)

요청 페이로드를 AWS Lambda invoke 작업에 전달하려면 여기에 이 요청 페이로드 파일의 경로를 지정하십시오. 파일은 형식이 맞아야 합니다. JSON

요청 페이로드 파일은 소스 리포지토리 또는 이전 작업의 아티팩트에 있을 수 있습니다. 파일 경로는 소스 리포지토리 또는 아티팩트 루트를 기준으로 합니다.

요청 페이로드를 Lambda 함수로 전달하지 않으려면 이 속성을 생략하십시오.

> **Note**
>
> `RequestPayload` 또는 `RequestPayloadFile`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

요청 페이로드 파일에 대한 자세한 내용은 참조의 Invoke 주제를 참조하십시오.AWS Lambda API

해당 UI: 구성 탭/ 요청 페이로드 파일 - 선택 사항

ContinueOnError

(*LambdaInvoke*/Configuration/RequestPayloadFile)

(선택 사항)

AWS Lambda 호출된 함수가 실패하더라도 호출 작업을 성공한 것으로 표시할지 여부를 지정합니다. AWS Lambda Lambda 장애에도 불구하고 워크플로의 후속 작업을 시작할 수 `true` 있도록 이 속성을 로 설정하는 것을 고려해 보십시오.

기본값은 Lambda 함수가 실패할 경우 작업이 실패하는 것입니다 (비주얼 `false` 에디터나 편집기에서는 “off”). YAML

해당 UI: 구성 탭/ 오류 발생 시 계속

LogType

(*LambdaInvoke*/Configuration/LogType)

(선택 사항)

호출 후 Lambda 함수의 응답에 오류 로그를 포함할지 여부를 지정합니다. 콘솔의 Lambda 호출 작업의 로그 탭에서 이러한 로그를 볼 수 있습니다. CodeCatalyst 가능한 값은 다음과 같습니다.

- `Tail`— 반환 로그
- `None`— 로그 반환 금지

기본값은 Tail입니다.

로그 유형에 대한 자세한 내용은 AWS Lambda API참조의 Invoke 항목을 참조하십시오.

로그 보기에 대한 자세한 내용은 워크플로 실행 상태 및 세부 정보 보기 단원을 참조하세요.

해당 UI: 구성 탭/ 로그 유형

ResponseFilters

(*LambdaInvoke*/Configuration/ResponseFilters)

(선택 사항)

Lambda 응답 페이로드에서 출력 변수로 변환하려는 키를 지정합니다. 그런 다음 워크플로의 후속 작업에서 출력 변수를 참조할 수 있습니다. 의 변수에 대한 자세한 내용은 CodeCatalyst 을 참조하십시오워크플로우에서 변수 사용.

예를 들어 응답 페이로드가 다음과 같은 경우:

```
responsePayload = {
  "name": "Saanvi",
  "location": "Seattle",
  "department": {
    "company": "Amazon",
    "team": "AWS"
  }
}
```

... 응답 필터는 다음과 같습니다.

```
Configuration:
  ...
  ResponseFilters: '{"name": ".name", "company": ".department.company"}'
```

... 그러면 액션은 다음과 같은 출력 변수를 생성합니다.

키	값
name	Saanvi
company	Amazon

그러면 후속 작업에서 name 및 company 변수를 참조할 수 있습니다.

에서 ResponseFilters 키를 지정하지 않는 경우 작업은 Lambda 응답의 각 최상위 키를 출력 변수로 변환합니다. 자세한 내용은 AWS Lambda '호출' 변수 단원을 참조하십시오.

응답 필터를 사용하여 생성된 출력 변수를 실제로 사용하려는 변수로만 제한하는 것을 고려해 보십시오.

해당 UI: 구성 탭/ 응답 필터 - 선택 사항

Outputs

(*LambdaInvoke*/Outputs)

(선택 사항)

워크플로우 실행 중 작업에 의해 출력되는 데이터를 정의합니다.

해당 UI: 출력 탭

Artifacts

(*LambdaInvoke*/Outputs/Artifacts)

(선택 사항)

액션으로 생성된 아티팩트를 지정합니다. 이러한 아티팩트를 다른 작업의 입력으로 참조할 수 있습니다.

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 출력 탭/아티팩트/빌드 아티팩트 이름

Name

(*LambdaInvoke*/Outputs/Artifacts/Name)

(선택 사항)

Lambda 함수에서 반환되는 Lambda 응답 페이로드를 포함할 아티팩트의 이름을 지정합니다. 기본값은 `lambda_artifacts`입니다. 아티팩트를 지정하지 않는 경우 Lambda 응답 페이로드는 콘솔의 작업에 대한 로그 탭에서 사용할 수 있는 작업 로그에서 볼 수 있습니다. CodeCatalyst 로그 보기에 대한 자세한 내용은 워크플로 실행 상태 및 세부 정보 보기 단원을 참조하세요.

해당 UI: 출력 탭/아티팩트/빌드 아티팩트 이름

Files

(*LambdaInvoke*/Outputs/Artifacts/Files)

(선택 사항)

아티팩트에 포함할 파일을 지정합니다. Lambda 응답 페이로드 파일이 `lambda-response.json` 포함되도록 지정해야 합니다.

해당 UI: 출력 탭/아티팩트/빌드로 생성된 파일

Amazon ECS 작업 정의 수정

이 단원에서는 CodeCatalyst 워크플로를 사용하여 Amazon Elastic Container Service (AmazonECS) 작업 정의 파일의 `image` 필드를 업데이트하는 방법을 설명합니다. 이 작업을 수행하려면 워크플로에

Render Amazon ECS 작업 정의 작업을 추가해야 합니다. 이 작업을 수행하면 작업 정의 파일의 이미지 필드가 런타임 시 워크플로에서 제공하는 Docker 이미지 이름으로 업데이트됩니다.

> **Note**
>
> 이 작업을 사용하여 작업 정의 `environment` 필드를 환경 변수로 업데이트할 수도 있습니다.

이 작업을 사용하는 경우

커밋 ID 또는 타임스탬프와 같은 동적 콘텐츠로 Docker 이미지를 빌드하고 태그를 지정하는 워크플로가 있는 경우 이 방법을 사용하십시오.

작업 정의 파일에 항상 동일하게 유지되는 이미지 값이 포함되어 있는 경우에는 이 작업을 사용하지 마세요. 이 경우 작업 정의 파일에 이미지 이름을 수동으로 입력할 수 있습니다.

'Amazon ECS 작업 정의 렌더링' 작업 작동 방식

워크플로의 빌드 및 Amazon에 배포 작업과 함께 Amazon ECS 작업 정의 렌더링 ECS 작업을 사용해야 합니다. 이러한 작업을 함께 수행하면 다음과 같이 작동합니다.

1. 빌드 작업은 Docker 이미지를 빌드하고 이름, 커밋 ID, 타임스탬프 또는 기타 동적 콘텐츠로 태그를 지정합니다. 예를 들어 빌드 작업은 다음과 같을 수 있습니다.

```
MyECSWorkflow
  Actions:
    BuildAction:
      Identifier: aws/build@v1
      ...
      Configuration:
        Steps:
        # Build, tag, and push the Docker image...
          - Run: docker build -t MyDockerImage:${WorkflowSource.CommitId} .
          ...
```

   위 코드에서 `docker build -t` 디렉티브는 Docker 이미지를 빌드하고 작업 런타임 시 커밋 ID로 태그를 지정하도록 지시합니다. 생성된 이미지 이름은 다음과 같을 수 있습니다.

   `MyDockerImage:a37bd7e`

2. Render Amazon ECS 작업 정의 작업은 동적으로 생성된 이미지 이름MyDockerImage:a37bd7e, 을 작업 정의 파일에 다음과 같이 추가합니다.

```
{
    "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-
execution-role",
    "containerDefinitions": [
        {
            "name": "codecatalyst-ecs-container",
            "image":  MyDockerImage:a37bd7e,
            "essential": true,
            ...
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 80
                }
            ]
        }
    ],
...
}
```

선택적으로 Render Amazon ECS 작업 정의 작업에서 다음과 같이 작업 정의에 환경 변수를 추가하도록 할 수도 있습니다.

```
{
  "executionRoleArn": "arn:aws:iam::account_ID:role/codecatalyst-ecs-task-execution-
role",
  "containerDefinitions": [
    {
      "name": "codecatalyst-ecs-container",
      "image":  MyDockerImage:a37bd7e,
      ...
      "environment": [
        {
          name": "ECS_LOGLEVEL",
          value": "info"
        }
      ]
    }
```

```
  ],
...
}
```

환경 변수에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 환경 변수 지정을 참조하십시오.

3. Amazon에 배포 ECS 작업은 업데이트된 작업 정의 파일을 ECS Amazon에 등록합니다. 업데이트된 작업 정의 파일을 등록하면 새 이미지가 `MyDockerImage:a37bd7e` ECS Amazon에 배포됩니다.

주제

- 예: Amazon ECS 태스크데프 수정
- 'Amazon ECS 작업 정의 렌더링' 작업 추가
- 업데이트된 작업 정의 파일 보기
- '렌더 아마존 ECS 태스크 정의' 변수
- 'Amazon ECS 작업 정의 렌더링' 작업 YAML

예: Amazon ECS 태스크데프 수정

다음은 빌드 및 배포 작업과 함께 Render Amazon ECS 작업 정의 작업을 포함하는 전체 워크플로의 예입니다. 워크플로의 목적은 Docker 이미지를 구축하여 Amazon ECS 클러스터에 배포하는 것입니다. 워크플로는 순차적으로 실행되는 다음과 같은 구성 블록으로 구성됩니다.

- 트리거 - 이 트리거는 소스 리포지토리에 변경 내용을 푸시할 때 워크플로가 자동으로 실행됩니다. 트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.
- 빌드 작업 (`BuildDocker`) — 트리거 시 작업은 Dockerfile을 사용하여 Docker 이미지를 빌드하고, 커밋 ID로 태그를 지정하고, 이미지를 Amazon에 푸시합니다. ECR 빌드 작업에 대한 자세한 내용은 을 참조하십시오. 워크플로를 사용한 구축
- Render Amazon ECS 작업 정의 작업 (`RenderTaskDef`) — 빌드 작업이 완료되면 이 작업은 소스 리포지토리의 루트에 `taskdef.json` 있는 기존 작업을 올바른 커밋 ID가 포함된 `image` 필드 값으로 업데이트합니다. 업데이트된 파일을 새 파일 이름 (`task-definition-`*`random-string`*`.json`) 으로 저장한 다음 이 파일이 포함된 출력 아티팩트를 생성합니다. 또한 렌더링 액션은 라는 `task-definition` 변수를 생성하여 새 작업 정의 파일의 이름으로 설정합니다. 아티팩트와 변수는 다음 단계인 배포 작업에 사용됩니다.

- Amazon에 배포 ECS 작업 (DeployToECS) — Amazon ECS 작업 정의 렌더링 작업을 완료하면 ECSAmazon에 배포 작업은 렌더링 작업 (TaskDefArtifact) 에서 생성된 출력 아티팩트를 찾아 그 안에서 task-definition-*random-string*.json 파일을 찾아 Amazon ECS 서비스에 등록합니다. 그러면 Amazon ECS 서비스가 task-definition-*random-string*.json 파일의 지침에 따라 Amazon 클러스터 내에서 Amazon ECS 작업 및 관련 Docker 이미지 컨테이너를 실행합니다. ECS

```
Name: codecatalyst-ecs-workflow
SchemaVersion: 1.0

Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
  BuildDocker:
    Identifier: aws/build@v1
    Environment:
      Name: codecatalyst-ecs-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-ecs-build-role
    Inputs:
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-
repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
    Configuration:
      Steps:
        #pre_build:
        - Run: echo Logging in to Amazon ECR...
        - Run: aws --version
        - Run: aws ecr get-login-password --region us-east-2 | docker login --username
 AWS --password-stdin 111122223333.dkr.ecr.us-east-2.amazonaws.com
        #build:
        - Run: echo Build started on `date`
        - Run: echo Building the Docker image...
        - Run: docker build -t $REPOSITORY_URI:latest .
        - Run: docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
```

```
        #post_build:
        - Run: echo Build completed on `date`
        - Run: echo Pushing the Docker images...
        - Run: docker push $REPOSITORY_URI:latest
        - Run: docker push $REPOSITORY_URI:$IMAGE_TAG

  RenderTaskDef:
    DependsOn:
      - BuildDocker
    Identifier: aws/ecs-render-task-definition@v1
    Inputs:
      Variables:
        - Name: REPOSITORY_URI
          Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-
repo
        - Name: IMAGE_TAG
          Value: ${WorkflowSource.CommitId}
    Configuration:
      task-definition: taskdef.json
      container-definition-name: codecatalyst-ecs-container
      image: $REPOSITORY_URI:$IMAGE_TAG
    # The output artifact contains the updated task definition file.
    # The new file is prefixed with 'task-definition'.
    # The output variable is set to the name of the updated task definition file.
    Outputs:
      Artifacts:
        - Name: TaskDefArtifact
          Files:
            - "task-definition*"
      Variables:
        - task-definition

  DeployToECS:
    Identifier: aws/ecs-deploy@v1
    Environment:
      Name: codecatalyst-ecs-environment
      Connections:
        - Name: codecatalyst-account-connection
          Role: codecatalyst-ecs-deploy-role
    #Input artifact contains the updated task definition file.
    Inputs:
      Sources: []
      Artifacts:
        - TaskDefArtifact
```

```
    Configuration:
      region: us-east-2
      cluster: codecatalyst-ecs-cluster
      service: codecatalyst-ecs-service
      task-definition: ${RenderTaskDef.task-definition}
```

'Amazon ECS 작업 정의 렌더링' 작업 추가

워크플로에 Render Amazon ECS 작업 정의 작업을 추가하려면 다음 지침을 따르십시오.

전제 조건

시작하기 전에 Docker 이미지를 동적으로 생성하는 빌드 작업이 포함된 워크플로가 있는지 확인하세요. 자세한 내용은 위의 예제 워크플로를 참조하십시오.

Visual

비주얼 편집기를 사용하여 'Render Amazon ECS 작업 정의' 작업을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. Render Amazon ECS 작업 정의 작업을 검색하고 다음 중 하나를 수행하십시오.
   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - Amazon ECS 작업 정의 렌더링을 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 소스 보기를 선택하여 액션의 소스 코드를 확인합니다.

- 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

10. 입력 및 구성 탭에서 필요에 따라 필드를 작성합니다. 각 필드에 대한 설명은 를 참조하십시오'Amazon ECS 작업 정의 렌더링' 작업 YAML. 이 참조는 편집기와 시각적 편집기에 모두 나타나는 각 필드 (및 해당 YAML 속성 값) 에 대한 자세한 정보를 제공합니다. YAML
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

편집기를 사용하여 'Render Amazon ECS 작업 정의' 작업을 추가하려면 YAML

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 왼쪽 상단에서 + Actions를 선택하여 액션 카탈로그를 엽니다.
8. 드롭다운 목록에서 Amazon을 선택합니다 CodeCatalyst.
9. Render Amazon ECS 작업 정의 작업을 검색하고 다음 중 하나를 수행하십시오.

   - 더하기 기호 (+) 를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

     Or

   - Amazon ECS 작업 정의 렌더링을 선택합니다. 작업 세부 정보 대화 상자가 나타납니다. 이 대화 상자에서:
     - (선택 사항) 소스 보기를 선택하여 액션의 소스 코드를 확인합니다.
     - 워크플로에 추가를 선택하여 워크플로 다이어그램에 작업을 추가하고 해당 구성 창을 엽니다.

10. 필요에 따라 YAML 코드의 속성을 수정하십시오. 사용 가능한 각 속성에 대한 설명은 에 나와 'Amazon ECS 작업 정의 렌더링' 작업 YAML 있습니다.
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

다음 단계

렌더링 작업을 추가한 후 의 지침에 따라 Amazon에 배포 ECS 작업을 워크플로에 추가합니다워크플로를 ECS 사용하여 Amazon에 배포하기. 배포 작업을 추가하는 동안 다음을 수행하십시오.

1. 배포 작업의 입력 탭에 있는 아티팩트 - 선택 사항에서 렌더링 작업으로 생성된 아티팩트를 선택합니다. 여기에는 업데이트된 작업 정의 파일이 들어 있습니다.

   아티팩트에 대한 자세한 내용은 작업 간 아티팩트 및 파일 공유 단원을 참조하십시오.
2. 배포 작업의 구성 탭에 `${action-name.task-definition}` 있는 작업 정의 필드에 다음 작업 변수를 지정합니다.*action-name* 는 렌더링 액션의 이름입니다 (예:)`RenderTaskDef`. 렌더 액션은 이 변수를 작업 정의 파일의 새 이름으로 설정합니다.

   변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

   배포 작업을 구성하는 방법에 대한 자세한 내용은 위의 예제 워크플로를 참조하십시오.

업데이트된 작업 정의 파일 보기

업데이트된 작업 정의 파일의 이름과 내용을 볼 수 있습니다.

Render Amazon 작업 정의 작업에서 처리된 후 업데이트된 ECS 작업 정의 파일의 이름을 보려면

1. 완료된 렌더링 작업이 포함된 실행을 찾으십시오.
   a. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
   b. 프로젝트를 선택합니다.
   c. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
   d. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
   e. 완료된 렌더링 작업이 포함된 실행을 선택합니다.

2. 워크플로 다이어그램에서 렌더링 작업을 선택합니다.
3. [출력] 을 선택합니다.
4. 변수를 선택합니다.
5. 작업 정의 파일 이름이 표시됩니다. 과 비슷해 보입니다`task-definition--259-0a2r7gxlTF5X-.json`.

업데이트된 작업 정의 파일의 내용을 보려면

1. 완료된 렌더링 작업이 포함된 실행을 찾으십시오.
   a. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
   b. 프로젝트를 선택합니다.
   c. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
   d. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
   e. 완료된 렌더링 작업이 포함된 실행을 선택합니다.
2. 워크플로 실행의 상단에 있는 Visual and YAML옆에서 워크플로 출력을 선택합니다.
3. 아티팩트 섹션에서 업데이트된 작업 정의 파일이 포함된 아티팩트 옆의 다운로드를 선택합니다. 이 아티팩트에는 렌더링 작업의 이름으로 Produced by 열이 설정됩니다.
4. .zip 파일을 열어 작업 정의.json 파일을 확인합니다.

'렌더 아마존 ECS 태스크 정의' 변수

Render Amazon ECS 작업 정의 작업은 런타임에 다음 변수를 생성하고 설정합니다. 이러한 변수를 사전 정의된 변수라고 합니다.

워크플로우에서 이러한 변수를 참조하는 방법에 대한 자세한 내용은 을 참조하십시오. 사전 정의된 변수 사용

키	값
작업 정의	Render Amazon 작업 정의 작업에 의해 업데이트된 ECS 작업 정의 파일에 지정된 이름입니다. 이름은 `task-definition-` *`random-string`* `.json` 형식입니다.

키	값
	예제: `task-definition--259-0a2r7g xlTF5Xr.json`

'Amazon ECS 작업 정의 렌더링' 작업 YAML

다음은 Render Amazon ECS 작업 YAML 정의 작업의 정의입니다. 이 작업의 사용 방법을 알아보려면 을 참조하십시오Amazon ECS 작업 정의 수정.

이 작업 정의는 광범위한 워크플로 정의 파일 내에 섹션으로 존재합니다. 이 파일에 대한 자세한 내용은 워크플로우 YAML 정의을 참조합니다.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

```
# The workflow definition starts here.
# See ### ## for details.

Name: MyWorkflow
SchemaVersion: 1.0
Actions:

# The action definition starts here.
  ECSRenderTaskDefinition_nn:
    Identifier: aws/ecs-render-task-definition@v1
    DependsOn:
      - build-action
    Compute:
      Type: EC2 | Lambda
      Fleet: fleet-name
    Timeout: timeout-minutes
    Inputs:
      # Specify a source or an artifact, but not both.
      Sources:
        - source-name-1
      Artifacts:
```

```
        - task-definition-artifact
      Variables:
        - Name: variable-name-1
          Value: variable-value-1
        - Name: variable-name-2
          Value: variable-value-2
    Configuration
      task-definition: task-definition-path
      container-definition-name: container-definition-name
      image: docker-image-name
      environment-variables:
        - variable-name-1=variable-value-1
        - variable-name-2=variable-value-2
    Outputs:
      Artifacts:
        - Name: TaskDefArtifact
          Files: "task-definition*"
      Variables:
        - task-definition
```

ECSRenderTaskDefinition

(필수)

액션의 이름을 지정합니다. 모든 작업 이름은 워크플로 내에서 고유해야 합니다. 액션 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 액션 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

기본값: `ECSRenderTaskDefinition_nn`.

해당 UI: 구성 탭/ 작업 이름

Identifier

(*ECSRenderTaskDefinition*/Identifier)

(필수)

작업을 식별합니다. 버전을 변경하려는 경우가 아니면 이 속성을 변경하지 마십시오. 자세한 내용은 사용할 액션 버전 지정 단원을 참조하십시오.

기본값: `aws/ecs-render-task-definition@v1`.

해당 UI: 워크플로 다이어그램/ ECSRenderTaskDefinition _n/ aws/ @v1 라벨 ecs-render-task-definition

DependsOn

(*ECSRenderTaskDefinition*/DependsOn)

(선택 사항)

이 작업을 실행하기 위해 성공적으로 실행되어야 하는 작업, 작업 그룹 또는 게이트를 지정하십시오.

'종속 조건' 기능에 대한 자세한 내용은 을 참조하십시오. 시퀀싱 액션

해당 UI: 입력 탭/ 기준 - 선택 사항

Compute

(*ECSRenderTaskDefinition*/Compute)

(선택 사항)

워크플로 작업을 실행하는 데 사용되는 컴퓨팅 엔진. 워크플로 수준 또는 작업 수준에서 컴퓨팅을 지정할 수 있지만 둘 다에서 지정할 수는 없습니다. 워크플로 수준에서 지정된 경우 컴퓨팅 구성은 워크플로에 정의된 모든 작업에 적용됩니다. 워크플로 수준에서는 동일한 인스턴스에서 여러 작업을 실행할 수도 있습니다. 자세한 내용은 작업 간 컴퓨팅 공유 단원을 참조하십시오.

해당 UI: 없음

Type

(*ECSRenderTaskDefinition*/Compute/Type)

(포함된 Compute 경우 필수)

컴퓨팅 엔진 유형. 다음 값 중 하나를 사용할 수 있습니다.

- EC2(비주얼 에디터) 또는 EC2 (YAML에디터)

  작업 실행 중 유연성을 위해 최적화되었습니다.
- Lambda (비주얼 에디터) 또는 (에디터) Lambda YAML

작업 시작 속도를 최적화했습니다.

컴퓨팅 유형에 대한 자세한 정보는 컴퓨팅 유형을 참고하세요.

해당 UI: 구성 탭/ 컴퓨팅 유형

Fleet

(*ECSRenderTaskDefinition*/Compute/Fleet)

(선택 사항)

워크플로 또는 워크플로 작업을 실행할 시스템 또는 플릿을 지정합니다. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다. 온디맨드 플릿의 예:,. `Linux.x86-64.Large` `Linux.x86-64.XLarge` 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

생략된 경우 기본값은 `Fleet` 입니다. `Linux.x86-64.Large`

해당 UI: 구성 탭/ 컴퓨팅 플릿

Timeout

(*ECSRenderTaskDefinition*/Timeout)

(선택 사항)

작업이 CodeCatalyst 종료되기 전에 작업을 실행할 수 있는 시간을 분 (편집기) 또는 시간과 분 (비주얼 편집기) 단위로 지정합니다. YAML 최소값은 5분이고 최대값은 에 설명되어 의 워크플로우 할당량 CodeCatalyst 있습니다. 기본 타임아웃은 최대 타임아웃과 동일합니다.

해당 UI: 구성 탭/ 타임아웃 - 선택 사항

Inputs

(*ECSRenderTaskDefinition*/Inputs)

(선택 사항)

이 Inputs 섹션에서는 워크플로우 실행 중에 ECSRenderTaskDefinition 필요한 데이터를 정의합니다.

> **Note**
>
> Render Amazon ECS 작업 정의 작업당 하나의 입력 (소스 또는 아티팩트) 만 허용됩니다. 변수는 이 총계에 포함되지 않습니다.

해당 UI: 입력 탭

Sources

(*ECSRenderTaskDefinition*/Inputs/Sources)

(작업 정의 파일이 소스 리포지토리에 저장되어 있는 경우 필수)

작업 정의 파일이 소스 리포지토리에 저장되어 있는 경우 해당 소스 리포지토리의 레이블을 지정하세요. 현재 지원되는 레이블은 뿐입니다WorkflowSource.

작업 정의 파일이 소스 리포지토리에 포함되어 있지 않은 경우 이 파일은 다른 작업에서 생성된 아티팩트에 있어야 합니다.

소스에 대한 자세한 내용은 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

해당 UI: 입력 탭/ 소스 - 선택 사항

Artifacts - input

(*ECSRenderTaskDefinition*/Inputs/Artifacts)

(작업 정의 파일이 이전 작업의 출력 아티팩트에 저장되어 있는 경우 필수)

배포하려는 작업 정의 파일이 이전 작업에서 생성된 아티팩트에 포함되어 있는 경우 여기에 해당 아티팩트를 지정하세요. 작업 정의 파일이 아티팩트에 포함되어 있지 않은 경우 해당 파일은 소스 리포지토리에 있어야 합니다.

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 구성 탭/ 아티팩트 - 선택 사항

Variables - input

(*ECSRenderTaskDefinition*/Inputs/Variables)

(필수)

작업에 사용할 수 있도록 하려는 입력 변수를 정의하는 이름/값 쌍의 시퀀스를 지정합니다. 변수 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 변수 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

예제를 비롯한 변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

해당 UI: 입력 탭/ 변수 - 선택 사항

Configuration

(*ECSRenderTaskDefinition*/Configuration)

(필수)

작업의 구성 속성을 정의할 수 있는 섹션입니다.

해당 UI: 구성 탭

task-definition

(*ECSRenderTaskDefinition*/Configuration/task-definition)

(필수)

기존 작업 정의 파일의 경로를 지정합니다. 파일이 소스 리포지토리에 있는 경우 경로는 소스 리포지토리 루트 폴더를 기준으로 합니다. 파일이 이전 워크플로 작업의 아티팩트에 있는 경우 경로는 아티팩트 루트 폴더를 기준으로 합니다. 작업 정의 파일에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 작업 정의를 참조하십시오.

해당 UI: 구성 탭/ 작업 정의

container-definition-name

(*ECSRenderTaskDefinition*/Configuration/container-definition-name)

(필수)

Docker 이미지가 실행될 컨테이너의 이름을 지정합니다. 이 이름은 작업 정의 파일 의`containerDefinitions`, `name` 필드에서 찾을 수 있습니다. 자세한 내용은 Amazon Elastic 컨테이너 서비스 개발자 안내서의 이름을 참조하십시오.

해당 UI: 구성 탭/ 컨테이너 이름

image

(*ECSRenderTaskDefinition*/Configuration/image)

(필수)

Render Amazon ECS 작업 정의 작업에서 작업 정의 파일에 추가하려는 Docker 이미지의 이름을 지정합니다. 액션은 이 이름을 작업 정의 파일의`containerDefinitions`, `image` 필드에 추가합니다. `image`필드에 값이 이미 있는 경우 액션은 해당 값을 덮어씁니다. 이미지 이름에 변수를 포함할 수 있습니다.

예시:

을 `MyDockerImage:${WorkflowSource.CommitId}` 지정하면 작업이 작업 정의 파일에 `MyDockerImage:`*`commit-id`* 추가됩니다. 여기서 *`commit-id`* 워크플로에서 런타임 시 생성되는 커밋 ID입니다.

`my-ecr-repo/image-repo:$(date +%m-%d-%y-%H-%m-%s)`지정하면 액션이 추가됩니다.*`my-ecr-repo`*/이미지-레포:*`date +%m-%d-%y-%H-%m-%s`* 작업 정의 파일로, 여기서 *`my-ecr-repo`* 는 Amazon Elastic 컨테이너 레지스트리 (ECR) 에 URI 속하며 *`date +%m-%d-%y-%H-%m-%s`* 워크플로에서 런타임 시 `month-day-year-hour-minute-second` 생성되는 형식의 타임스탬프입니다.

이 `image` 필드에 대한 자세한 내용은 Amazon Elastic 컨테이너 서비스 개발자 안내서의 이미지를 참조하십시오. 변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

해당 UI: 구성 탭/ 이미지 이름

environment-variables

(*ECSRenderTaskDefinition*/Configuration/environment-variables)

(필수)

Render Amazon ECS 작업 정의 작업에서 작업 정의 파일에 추가하려는 환경 변수를 지정합니다. 이 작업은 작업 정의 파일의`containerDefinitions`, `environment` 필드에 변수를 추가합니다. 파일에 변수가 이미 있는 경우 액션은 기존 변수 값을 덮어쓰고 새 변수를 추가합니다. Amazon ECS 환경 변수에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 환경 변수 지정을 참조하십시오.

해당 UI: 구성 탭/ 환경 변수 - 선택 사항

Outputs

(*ECSRenderTaskDefinition*/Outputs)

(필수)

워크플로우 실행 중 작업에 의해 출력되는 데이터를 정의합니다.

해당 UI: 출력 탭

Artifacts

(*ECSRenderTaskDefinition*/Outputs/Artifacts)

(필수)

액션으로 생성된 아티팩트를 지정합니다. 이러한 아티팩트를 다른 작업의 입력으로 참조할 수 있습니다.

예제를 포함한 아티팩트에 대한 자세한 내용은 을 참조하십시오. 작업 간 아티팩트 및 파일 공유

해당 UI: 출력 탭/ 아티팩트

Name

(*ECSRenderTaskDefinition*/Outputs/Artifacts/Name)

(필수)

업데이트된 작업 정의 파일을 포함할 아티팩트의 이름을 지정합니다. 기본값은 `MyTaskDefinitionArtifact`입니다. 그런 다음 이 아티팩트를 Amazon에 배포 ECS 작업에 대한 입력으로 지정해야 합니다. 이 아티팩트를 Amazon에 배포 ECS 작업에 입력으로 추가하는 방법을 이해하려면 을 참조하십시오예: Amazon ECS 태스크데프 수정.

해당 UI: 출력 탭/아티팩트/이름

Files

(*ECSRenderTaskDefinition*/Outputs/Artifacts/Files)

(필수)

아티팩트에 포함할 파일을 지정합니다. 로 `task-definition-` 시작하는 업데이트된 작업 정의 파일이 `task-definition-*` 포함되도록 지정해야 합니다.

해당 UI: 출력 탭/아티팩트/파일

Variables

(*ECSRenderTaskDefinition*/Outputs/Variables)

(필수)

렌더링 액션에서 설정할 변수 이름을 지정합니다. 렌더링 액션은 이 변수 값을 업데이트된 작업 정의 파일의 이름 (예:`task-definition-`*`random-string`*`.json`) 으로 설정합니다. 그런 다음 ECSAmazon으로 배포 작업의 작업 정의 (시각적 편집기) 또는 `task-definition` (yaml 편집기) 속성에 이 변수를 지정해야 합니다. 이 변수를 Amazon에 배포 ECS 작업에 추가하는 방법을 이해하려면 을 참조하십시오예: Amazon ECS 태스크데프 수정.

기본값: `task-definition`

해당 UI: 출력 탭/변수/이름 필드

워크플로우에서 변수 사용

변수는 Amazon CodeCatalyst 워크플로에서 참조할 수 있는 정보가 포함된 키-값 쌍입니다. 워크플로가 실행되면 변수의 값 부분이 실제 값으로 대체됩니다.

워크플로에서 사용할 수 있는 변수에는 두 가지 유형이 있습니다.

- 사용자 정의 변수 — 사용자가 정의하는 키-값 쌍입니다.
- 사전 정의된 변수 - 워크플로우에서 자동으로 생성되는 키-값 쌍입니다. 직접 정의할 필요가 없습니다.

워크플로에 대한 자세한 내용은 워크플로를 통한 빌드, 테스트, 배포 섹션을 참조하세요.

> **Note**
>
> CodeCatalyst 변수처럼 동작하고 다른 작업에서 참조할 수 있는 GitHub 출력 매개변수도 지원합니다. 자세한 내용은 GitHub 출력 매개변수 내보내기 및 GitHub 출력 매개변수 참조 섹션을 참조하세요.

주제

- 사용자 정의 변수 사용
- 사전 정의된 변수 사용

사용자 정의 변수 사용

사용자 정의 변수는 사용자가 정의하는 키-값 쌍입니다. 다음과 같은 두 가지 유형이 있습니다.

- 일반 텍스트 변수 또는 단순 변수 — 워크플로 정의 파일 내에서 일반 텍스트로 정의하는 키-값 쌍입니다.
- 비밀 — Amazon CodeCatalyst 콘솔의 개별 비밀 페이지에서 정의하는 키-값 쌍입니다. 키 (이름) 는 공개 레이블이고, 값에는 비공개로 유지하려는 정보가 들어 있습니다. 워크플로 정의 파일에서만 키를 지정할 수 있습니다. 워크플로 정의 파일의 암호 및 기타 민감한 정보 대신 암호를 사용하십시오.

> **Note**
>
> 간결하게 설명하기 위해 이 가이드에서는 변수라는 용어를 일반 텍스트 변수를 의미하는 것으로 사용합니다.

변수에 대한 자세한 내용은 을 참조하십시오. 워크플로우에서 변수 사용

주제

- 변수의 예
- 변수 정의하기
- 시크릿 정의하기
- 다른 액션에서 사용할 수 있도록 변수 내보내기

- 변수를 정의하는 액션에서 변수 참조
- 다른 액션의 변수 출력 참조
- 시크릿 참조하기

변수의 예

다음 예제는 워크플로 정의 파일에서 변수를 정의하고 참조하는 방법을 보여줍니다.

변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

예

- 예: Inputs 속성을 사용하여 변수 정의하기
- 예: Steps 속성을 사용하여 변수 정의하기
- 예: Outputs 속성을 사용하여 변수 내보내기
- 예: 동일한 액션에 정의된 변수 참조
- 예: 다른 액션에 정의된 변수 참조
- 예: 시크릿 참조

예: Inputs 속성을 사용하여 변수 정의하기

다음 예제는 `Inputs` 섹션에서 두 개의 변수 `VAR1` 및 `VAR2` 를 정의하는 방법을 보여줍니다.

```
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Variables:
      - Name: VAR1
        Value: "My variable 1"
      - Name: VAR2
        Value: "My variable 2"
```

예: Steps 속성을 사용하여 변수 정의하기

다음 예제는 `Steps` 섹션에서 `DATE` 변수를 명시적으로 정의하는 방법을 보여줍니다.

```
Actions:
```

```
  Build:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: DATE=$(date +%m-%d-%y)
```

예: Outputs 속성을 사용하여 변수 내보내기

다음 예제는 Outputs 섹션을 사용하여 두 개의 변수를 정의하고 내보내는 방법을 보여줍니다. REPOSITORY-URI TIMESTAMP

```
Actions:
  Build:
    Identifier: aws/build@v1
    Inputs:
      Variables:
        - Name: REPOSITORY-URI
          Value: 111122223333.dkr.ecr.us-east-2.amazonaws.com/codecatalyst-ecs-image-
repo
    Configuration:
      Steps:
        - Run: TIMESTAMP=$(date +%m-%d-%y-%H-%m-%s)
    Outputs:
      Variables:
        - REPOSITORY-URI
        - TIMESTAMP
```

예: 동일한 액션에 정의된 변수 참조

다음 예제에서는 에서 VAR1 MyBuildAction 변수를 지정한 다음 를 사용하여 $VAR1 동일한 액션에서 해당 변수를 참조하는 방법을 보여줍니다.

```
Actions:
  MyBuildAction:
    Identifier: aws/build@v1
    Inputs:
      Variables:
        - Name: VAR1
          Value: my-value
    Configuration:
      Steps:
        - Run: $VAR1
```

예: 다른 액션에 정의된 변수 참조

다음 예제는 에서 TIMESTAMP 변수를 지정하고BuildActionA, Outputs 속성을 사용하여 변수를 내보낸 다음, 사용 시 변수를 참조하는 방법을 보여줍니다. BuildActionB ${BuildActionA.TIMESTAMP}

```
Actions:
  BuildActionA:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: TIMESTAMP=$(date +%m-%d-%y-%H-%m-%s)
    Outputs:
      Variables:
        - TIMESTAMP
  BuildActionB:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: docker build -t my-ecr-repo/image-repo:latest .
        - Run: docker tag my-ecr-repo/image-repo:${BuildActionA.TIMESTAMP}

        # Specifying just '$TIMESTAMP' here will not work
        # because TIMESTAMP is not a variable
        # in the BuildActionB action.
```

예: 시크릿 참조

다음 예제는 my-password 비밀을 참조하는 방법을 보여줍니다. 비밀의 키입니다. my-password 이 암호의 키와 해당 암호 값은 워크플로 정의 파일에서 사용하기 전에 CodeCatalyst 콘솔의 비밀 페이지에서 지정해야 합니다. 자세한 내용은 비밀을 사용한 데이터 마스킹 단원을 참조하십시오.

```
Actions:
  BuildActionA:
    Identifier: aws/build@v1
    Configuration:
      Steps:
        - Run: curl -u LiJuan:${Secrets.my-password} https://example.com
```

변수 정의하기

다음과 같은 두 가지 방법으로 변수를 정의할 수 있습니다.

- 워크플로 작업 Inputs 섹션에서 — '입력' 섹션의 변수 정의하기를 참조하십시오.
- 워크플로 작업 Steps 섹션에서 — '단계' 섹션의 변수 정의하기를 참조하십시오.

> **Note**
>
> 이 Steps 메서드는 CodeCatalyst 빌드, 테스트 및 GitHub 액션 작업에만 사용할 수 있습니다. 이는 Steps 섹션을 포함하는 유일한 작업이기 때문입니다.

예제는 변수의 예 섹션을 참조하세요.

변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

Visual

'입력' 섹션에서 변수 정의하기 (비주얼 에디터)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로 다이어그램에서 변수를 설정하려는 작업을 선택합니다.
8. 입력을 선택합니다.
9. 변수 - 선택 사항에서 변수 추가를 선택하고 다음을 수행합니다.

   작업에 사용할 수 있도록 하려는 입력 변수를 정의하는 이름/값 쌍의 시퀀스를 지정합니다. 변수 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-), 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 변수 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

   예제를 비롯한 변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.
10. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
11. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

'입력' 섹션에서 변수를 정의하려면 (YAML편집기)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 워크플로 작업에 다음과 비슷한 코드를 추가합니다.

```
action-name:
  Inputs:
    Variables:
      - Name: variable-name
        Value: variable-value
```

   더 많은 예제는 변수의 예를 참조합니다. 자세한 내용은 해당 작업에 워크플로우 YAML 정의 대한 을 참조하십시오.
8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사하십시오.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

Visual

'단계' 섹션에서 변수를 정의하려면 (비주얼 에디터)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.

7. 워크플로 다이어그램에서 변수를 설정하려는 작업을 선택합니다.
8. Configuration(구성)을 선택합니다.
9. 사용 가능한 셸 명령 또는 GitHubYAML액션에서 액션의 `Steps` 변수를 명시적 또는 묵시적으로 정의합니다.

   - 변수를 명시적으로 정의하려면 bash 명령어를 사용하여 섹션에 직접 포함시키십시오. `Steps`
   - 변수를 암시적으로 정의하려면 작업 섹션에서 참조하는 파일에 변수를 지정하세요. `Steps`

     예제는 변수의 예 섹션을 참조하세요. 자세한 내용은 해당 액션을 참조하십시오워크플로우 YAML 정의.
10. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
11. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

'단계' 섹션에서 변수를 정의하려면 (YAML편집기)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 워크플로 작업에서 작업 `Steps` 섹션에 변수를 명시적으로 또는 암시적으로 정의합니다.

   - 변수를 명시적으로 정의하려면 섹션의 bash 명령어에 변수를 직접 포함시키십시오. `Steps`
   - 변수를 암시적으로 정의하려면 작업 섹션에서 참조하는 파일에 변수를 지정하세요. `Steps`

     예제는 변수의 예 섹션을 참조하세요. 자세한 내용은 해당 액션을 참조하십시오워크플로우 YAML 정의.
8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.

9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

시크릿 정의하기

CodeCatalyst콘솔의 시크릿 페이지에서 시크릿을 정의합니다. 자세한 내용은 비밀을 사용한 데이터 마스킹 단원을 참조하십시오.

예를 들어 다음과 같은 시크릿을 정의할 수 있습니다.

- 이름 (키): **my-password**
- 값: **^*H3#!b9**

암호가 정의되면 워크플로 정의 파일에 암호의 키 (**my-password**) 를 지정할 수 있습니다. 이렇게 하는 방법의 예는 예: 시크릿 참조 섹션을 참조하세요.

다른 액션에서 사용할 수 있도록 변수 내보내기

다음 지침에 따라 액션에서 변수를 내보내 다른 액션에서 해당 변수를 참조할 수 있습니다.

변수를 내보내기 전에 다음 사항을 참고하세요.

- 변수가 정의된 액션 내에서만 변수를 참조해야 하는 경우에는 변수를 내보내지 않아도 됩니다.
- 모든 액션이 변수 내보내기를 지원하는 것은 아닙니다. 액션이 이 기능을 지원하는지 확인하려면 다음에 나오는 시각적 편집기 지침을 살펴보고 액션에 출력 탭의 변수 버튼이 포함되어 있는지 확인하세요. 그렇다면 변수 내보내기가 지원됩니다.
- GitHub 액션에서 변수를 내보내려면 을 참조하십시오 GitHub 출력 매개변수 내보내기.

변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

전제 조건

내보내려는 변수를 정의했는지 확인하십시오. 자세한 내용은 변수 정의하기 단원을 참조하십시오.

Visual

변수를 내보내려면 (비주얼 에디터)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.

3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.
7. 워크플로 다이어그램에서 변수를 내보낼 작업을 선택합니다.
8. 출력을 선택합니다.
9. 변수 - 선택 사항에서 변수 추가를 선택하고 다음을 수행합니다.

   액션으로 내보내려는 변수 이름을 지정합니다. 이 변수는 동일한 액션의 `Inputs` 또는 `Steps` 섹션에 이미 정의되어 있어야 합니다.
10. (선택 사항) 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사하려면 [Validate] 를 선택합니다.
11. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

YAML

변수를 내보내려면 (YAML편집기)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 변수를 내보낼 액션에 다음과 비슷한 코드를 추가합니다.

```
action-name:
  Outputs:
    Variables:
      - Name: variable-name
```

더 많은 예제는 변수의 예를 참조합니다.

8. (선택 사항) 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사하려면 [Validate] 를 선택합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

변수를 정의하는 액션에서 변수 참조

다음 지침에 따라 변수를 정의하는 액션에서 변수를 참조하세요.

> **Note**
>
> GitHub 액션으로 생성된 변수를 참조하려면 을 참조하십시오 GitHub 출력 매개변수 참조.

변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

전제 조건

참조하려는 변수를 정의했는지 확인하십시오. 자세한 내용은 변수 정의하기 단원을 참조하십시오.

Visual

사용할 수 없습니다. YAML지침을 YAML 보려면 선택하십시오.

YAML

변수를 정의하는 동작에서 변수를 참조하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. 참조하려는 변수를 정의하는 CodeCatalyst 액션에서 다음 bash 구문을 사용하여 변수를 추가합니다.

```
$variable-name
```

예:

```
MyAction:
    Configuration:
      Steps:
        - Run: $variable-name
```

더 많은 예제는 변수의 예를 참조합니다. 자세한 내용은 의 워크플로우 YAML 정의 작업에 대한 참조 정보를 참조하십시오.

8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

다른 액션의 변수 출력 참조

다음 지침을 사용하여 다른 작업에 의해 출력된 변수를 참조할 수 있습니다.

> **Note**
>
> GitHub 액션의 변수 출력을 참조하려면 을 참조하십시오 GitHub 출력 매개변수 참조.

변수에 대한 자세한 내용은 을 참조하십시오워크플로우에서 변수 사용.

전제 조건

참조하려는 변수를 내보냈는지 확인하십시오. 자세한 내용은 다른 액션에서 사용할 수 있도록 변수 내보내기 단원을 참조하십시오.

Visual

사용할 수 없습니다. YAML지침을 YAML 보려면 선택하십시오.

YAML

다른 작업의 변수 출력을 참조하려면 (YAML편집기)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.

3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. CodeCatalyst 액션에서 다음 구문을 사용하여 변수에 대한 참조를 추가합니다.

```
${action-group-name.action-name.variable-name}
```

바꾸기:

- *action-group-name* 변수를 출력하는 액션을 포함하는 액션 그룹의 이름을 입력합니다.

> **Note**
>
> 생략할 수 있습니다.*action-group-name* 작업 그룹이 없는 경우 또는 동일한 작업 그룹의 작업에 의해 변수가 생성된 경우

- *action-name* 변수를 출력하는 액션의 이름과 함께.
- *variable-name* 변수 이름과 함께.

예:

```
MySecondAction:
    Configuration:
      Steps:
        - Run: ${MyFirstAction.TIMESTAMP}
```

더 많은 예제는 변수의 예를 참조합니다. 자세한 내용은 해당 작업에 워크플로우 YAML 정의 대한 을 참조하십시오.

8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사하십시오.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

시크릿 참조하기

워크플로 정의 파일에서 암호를 참조하는 방법에 대한 지침은 을 참조하십시오. 시크릿 사용

예시는 예: 시크릿 참조에서 확인하십시오.

사전 정의된 변수 사용

사전 정의된 변수는 워크플로에서 자동으로 생성되며 워크플로 작업에 사용할 수 있는 키-값 쌍입니다.

변수에 대한 자세한 내용은 을 참조하십시오. 워크플로우에서 변수 사용

주제

- 사전 정의된 변수 참조의 예
- 사전 정의된 변수 참조
- 워크플로에서 내보내는 사전 정의된 변수 결정
- 사전 정의된 변수 목록

사전 정의된 변수 참조의 예

다음 예제는 워크플로 정의 파일에서 사전 정의된 변수를 참조하는 방법을 보여줍니다.

사전 정의된 변수에 대한 자세한 내용은 을 참조하십시오. 사전 정의된 변수 사용

예

- 예: 미리 정의된 CommitId ""변수 참조
- 예: "BranchName" 사전 정의된 변수 참조

예: 미리 정의된 CommitId ""변수 참조

다음 예제는 작업에서 `CommitId` 사전 정의된 변수를 참조하는 방법을 보여줍니다. `MyBuildAction`에서 `CommitId` 변수를 자동으로 출력합니다. CodeCatalyst 자세한 내용은 사전 정의된 변수 목록 단원을 참조하십시오.

이 예제에서는 빌드 작업에 사용되는 변수를 보여 주지만 모든 작업에 사용할 `CommitId` 수 있습니다.

```
MyBuildAction:
    Identifier: aws/build@v1
```

```
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
      #Build Docker image and tag it with a commit ID
        - Run: docker build -t image-repo/my-docker-image:latest .
        - Run: docker tag image-repo/my-docker-image:${WorkflowSource.CommitId}
```

예: "BranchName" 사전 정의된 변수 참조

다음 예제는 작업에서 BranchName 사전 정의된 변수를 참조하는 방법을 보여줍니다. CDKDeploy 에서 BranchName 변수를 자동으로 출력합니다. CodeCatalyst 자세한 내용은 사전 정의된 변수 목록 단원을 참조하십시오.

이 예제에서는 AWS CDK 배포 작업에 사용되는 변수를 보여 주지만 모든 작업에 사용할 BranchName 수 있습니다.

```
CDKDeploy:
    Identifier: aws/cdk-deploy@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      StackName: app-stack-${WorkflowSource.BranchName}
```

사전 정의된 변수 참조

Amazon CodeCatalyst 워크플로 내의 모든 작업에서 사전 정의된 변수를 참조할 수 있습니다.

다음 지침을 사용하여 워크플로의 사전 정의된 변수를 참조하십시오.

사전 정의된 변수에 대한 자세한 내용은 을 참조하십시오. 사전 정의된 변수 사용

전제 조건

참조하려는 사전 정의된 변수의 이름 (예:) 을 결정합니다. CommitId 자세한 내용은 워크플로에서 내보내는 사전 정의된 변수 결정 단원을 참조하십시오.

Visual

사용할 수 없습니다. YAML지침을 YAML 보려면 선택하십시오.

YAML

사전 정의된 변수를 참조하려면 (YAML편집기)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. CodeCatalyst 액션에서 다음 구문을 사용하여 사전 정의된 변수 참조를 추가합니다.

```
${action-group-name.action-name-or-WorkflowSource.variable-name}
```

바꾸기:

- *action-group-name* 액션 그룹 이름과 함께.

  > **Note**
  >
  > 생략할 수 있습니다.*action-group-name* 작업 그룹이 없는 경우 또는 동일한 작업 그룹의 작업에 의해 변수가 생성된 경우

- *action-name-or-WorkflowSource* 다음으로 바꿉니다.

  변수를 출력하는 액션의 이름.

  또는

  `WorkflowSource`, 변수가 `BranchName` 또는 `CommitId` 변수인 경우

- *variable-name* 변수 이름과 함께.

예:

```
MySecondAction:
    Configuration:
      Steps:
```

```
        - Run: echo ${MyFirstECSAction.cluster}
```

또 다른 예시:

```
MySecondAction:
    Configuration:
      Steps:
        - Run: echo ${WorkflowSource.CommitId}
```

더 많은 예제는 사전 정의된 변수 참조의 예를 참조합니다. 자세한 내용은 해당 작업에 워크플로우 YAML 정의 대한 을 참조하십시오.

8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사하십시오.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

워크플로에서 내보내는 사전 정의된 변수 결정

다음 절차를 사용하여 워크플로가 실행될 때 내보내는 사전 정의된 변수를 결정하십시오. 그런 다음 동일한 워크플로 내에서 이러한 변수를 참조할 수 있습니다.

사전 정의된 변수에 대한 자세한 내용은 을 참조하십시오사전 정의된 변수 사용.

워크플로에서 내보내는 사전 정의된 변수를 확인하려면

- 다음 중 하나를 수행합니다.
  - 워크플로를 한 번 실행합니다. 실행이 완료되면 워크플로에서 생성된 변수가 실행 세부 정보 페이지의 변수 탭에 표시됩니다. 자세한 내용은 워크플로 실행 상태 및 세부 정보 보기 단원을 참조하십시오.
  - 를 참조하십시오. 사전 정의된 변수 목록 이 참조에는 사전 정의된 각 변수의 변수 이름 (키) 과 값이 나열되어 있습니다.

> **Note**
>
> 워크플로 변수의 최대 총 크기는 에 나열되어 있습니다. 의 워크플로우 할당량 CodeCatalyst 전체 크기가 최대값을 초과하는 경우 최대값에 도달한 후에 발생하는 작업이 실패할 수 있습니다.

사전 정의된 변수 목록

워크플로 실행의 일부로 CodeCatalyst 작업에 의해 자동으로 생성되는 사전 정의된 변수를 보려면 다음 섹션을 참조하십시오.

사전 정의된 변수에 대한 자세한 내용은 을 참조하십시오. 사전 정의된 변수 사용

> **Note**
>
> 이 목록에는 CodeCatalyst 소스 및 액션에서 내보낸 사전 정의된 변수만 포함됩니다. CodeCatalyst Action이나 CodeCatalyst Labs 액션과 같은 다른 유형의 GitHub 액션을 사용하는 경우 대신 참조하십시오. 워크플로에서 내보내는 사전 정의된 변수 결정

목록

> **Note**
>
> 모든 CodeCatalyst 액션이 사전 정의된 변수를 생성하는 것은 아닙니다. 액션이 목록에 없으면 변수가 생성되지 않습니다.

- 'BranchName' 및 'CommitId' 변수
- '디플로이 AWS CloudFormation 스택' 변수
- '아마존에 ECS 배포' 변수
- '쿠버네티스 클러스터에 배포' 변수
- AWS CDK '배포' 변수
- AWS CDK '부트스트랩' 변수
- AWS Lambda '호출' 변수
- '렌더 아마존 ECS 태스크 정의' 변수

비밀을 사용한 데이터 마스킹

워크플로에 인증 자격 증명과 같은 민감한 데이터를 사용해야 하는 경우가 있을 수 있습니다. 암호가 포함된 리포지토리에 액세스할 수 있는 모든 사용자가 해당 값을 볼 수 있으므로 리포지토리 어디에나

이러한 값을 일반 텍스트로 저장하는 것은 피해야 합니다. 마찬가지로 이러한 값은 저장소에서 파일로 표시되므로 워크플로 정의에 직접 사용해서는 안 됩니다. 를 사용하면 프로젝트에 암호를 추가한 다음 워크플로 정의 파일에서 암호를 참조하여 이러한 값을 보호할 수 있습니다. CodeCatalyst 액션당 최대 5개의 시크릿을 보유할 수 있다는 점에 유의하세요.

> **Note**
>
> 암호는 워크플로 정의 파일의 암호와 민감한 정보를 대체하는 데만 사용할 수 있습니다.

주제

시크릿 만들기

시크릿을 만들려면 다음 절차를 사용하십시오. 암호에는 보이지 않게 숨기려는 민감한 정보가 들어 있습니다.

> **Note**
>
> 보안 암호는 작업에 표시되며 파일에 기록될 때 가려지지 않습니다.

보안 암호 생성

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 시크릿을 선택합니다.
3. 암호 생성을 선택합니다.
4. 다음 정보를 입력합니다.

   이름

   시크릿 이름을 입력합니다.

값

비밀의 값을 입력합니다. 보이지 않게 숨기고 싶은 민감한 정보입니다. 기본적으로 이 값은 표시되지 않습니다. 값을 표시하려면 값 보기를 선택합니다.

설명

(선택 사항) 비밀에 대한 설명을 입력합니다.

5. 생성(Create)을 선택합니다.

시크릿 편집

암호를 편집하려면 다음 절차를 사용하십시오.

시크릿을 편집하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 시크릿을 선택합니다.
3. 시크릿 목록에서 편집하려는 시크릿을 선택합니다.
4. 편집을 선택합니다.
5. 다음 속성을 편집하십시오.

값

비밀의 값을 입력합니다. 이 값은 보이지 않게 숨기려는 값입니다. 기본적으로 값은 표시되지 않습니다.

설명

(선택 사항) 비밀에 대한 설명을 입력합니다.

6. 저장(Save)을 선택합니다.

시크릿 사용

워크플로 작업에서 암호를 사용하려면 암호의 참조 식별자를 얻고 워크플로 작업에서 해당 식별자를 사용해야 합니다.

주제

- 비밀의 식별자 획득

- 워크플로우에서 시크릿 참조

비밀의 식별자 획득

다음 절차를 사용하여 비밀의 참조 식별자를 얻을 수 있습니다. 워크플로에 이 식별자를 추가할 것입니다.

비밀의 참조 식별자를 얻으려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 시크릿을 선택합니다.
3. 암호 목록에서 사용하려는 암호를 찾으십시오.
4. 참조 ID 열에 비밀의 식별자를 복사합니다. 참조 ID의 구문은 다음과 같습니다.

```
${Secrets.<name>}
```

워크플로우에서 시크릿 참조

워크플로우에서 비밀을 참조하려면 다음 절차를 사용하십시오.

시크릿을 참조하려면

1. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
2. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
3. 편집을 선택합니다.
4. 선택합니다 YAML.
5. 비밀의 YAML 식별자를 사용하도록 수정하십시오. 예를 들어, 암호로 저장된 사용자 이름과 암호를 `curl` 명령과 함께 사용하려면 다음과 비슷한 Run 명령을 사용합니다.

```
- Run: curl -u <username-secret-identifier>:<password-secret-identifier> https://
example.com
```

6. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사합니다.
7. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

시크릿 삭제

다음 절차를 사용하여 비밀과 비밀 참조 식별자를 삭제하십시오.

> **Note**
>
> 암호를 삭제하기 전에 모든 워크플로 작업에서 암호의 참조 식별자를 제거하는 것이 좋습니다. 참조 식별자를 삭제하지 않고 암호를 삭제하면 다음 번에 작업을 실행할 때 작업이 실패합니다.

워크플로에서 비밀의 참조 식별자를 삭제하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
3. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
4. 편집을 선택합니다.
5. 선택합니다 YAML.
6. 워크플로에서 다음 문자열을 검색합니다.

   ```
   ${Secrets.
   ```

   이렇게 하면 모든 비밀의 참조 식별자가 모두 검색됩니다.
7. 선택한 비밀의 참조 식별자를 삭제하거나 일반 텍스트 값으로 대체합니다.
8. (선택 사항) [Validate] 를 선택하여 커밋하기 전에 워크플로우 YAML 코드의 유효성을 검사합니다.
9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.

보안 암호를 삭제하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 CI/CD를 선택한 다음 시크릿을 선택합니다.
3. 시크릿 목록에서 삭제하려는 시크릿을 선택합니다.
4. Delete(삭제)를 선택합니다.
5. 삭제를 **`delete`** 확인하려면 Enter를 누르십시오.

6. Delete(삭제)를 선택합니다.

워크플로우 상태 보기

워크플로의 상태를 보고 해결해야 하는 워크플로 구성 문제가 있는지 확인하거나 시작에 실패한 실행 문제를 해결할 수 있습니다. CodeCatalyst워크플로의 기본 워크플로 정의 파일을 만들거나 업데이트할 때마다 워크플로 상태를 평가합니다.

> **Note**
>
> 워크플로 상태와는 다른 워크플로의 실행 상태도 볼 수 있습니다. 자세한 내용은 워크플로 실행 상태 및 세부 정보 보기 단원을 참조하십시오.

가능한 워크플로 상태 목록은 을 참조하십시오워크플로우 상태 CodeCatalyst.

워크플로의 상태를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.

   상태는 워크플로와 함께 목록에 표시됩니다.
5. (선택 사항) 워크플로의 이름을 선택하고 워크플로 정의 필드를 찾습니다. 워크플로우 상태가 표시됩니다.

의 워크플로우 할당량 CodeCatalyst

다음 표에는 Amazon의 워크플로우 할당량 및 한도가 설명되어 있습니다. CodeCatalyst

CodeCatalystAmazon의 할당량에 대한 자세한 내용은 을 참조하십시오. 에 대한 할당량 CodeCatalyst

스페이스당 최대 워크플로 수	800
최대 워크플로 정의 파일 크기	256KB

단일 소스 이벤트에서 처리되는 최대 워크플로 파일 수	50
단일 소스 이벤트에서 처리되는 최대 파일 수	4,000
스페이스당 최대 활성 플릿 수	10
플릿당 최대 활성 컴퓨팅 인스턴스 수	20
작업당 최대 입력 아티팩트 수	10
액션당 최대 출력 아티팩트 수	10
단일 액션의 출력 변수의 최대 총 크기	120킬로바이트
출력 변수 값의 최대 길이	값을 내보내는 작업에 따라 500자 이상 **Note** 값이 작업 한도를 초과할 경우 값이 잘릴 수 있습니다.
워크플로우 실행 중에 생성된 아티팩트를 보관할 수 있는 최대 기간 (일)	30
작업당 최대 보고서 수	50
테스트 보고서당 최대 테스트 케이스 수	20,000건
코드 커버리지 보고서당 최대 파일 수	20,000건
보고서당 최대 소프트웨어 구성 분석 결과 수	20,000건
정적 분석 보고서당 최대 파일 수	20,000건
스페이스당 최대 동시 워크플로 실행 수	100
워크플로당 최대 작업 수	50
워크플로우당 동시에 실행되는 최대 작업 수	50

스페이스당 동시에 실행되는 최대 작업 수	200
작업을 실행할 수 있는 최대 시간	빌드 및 테스트 작업의 제한 시간은 8시간입니다. 다른 모든 작업의 제한 시간은 1시간입니다.
스페이스와 관련된 최대 환경 AWS 계정 수	5,000
액션당 최대 시크릿 수	5
스페이스당 최대 비밀 수	500,000

워크플로 실행 상태

워크플로 실행은 다음 상태 중 하나일 수 있습니다.

- 성공 — 워크플로 실행이 성공적으로 처리되었습니다.
- 실패 - 워크플로 실행에서 하나 이상의 작업이 실패했습니다.
- 진행 중 - 워크플로 실행이 현재 처리 중입니다.
- 중지됨 - 워크플로가 진행 중인 동안 한 사용자가 워크플로 실행을 중지했습니다.
- 중지 - 워크플로 실행이 현재 중지되고 있습니다.
- 취소됨 - 실행 진행 중에 관련 워크플로가 삭제되거나 CodeCatalyst 업데이트되어 워크플로 실행이 취소되었습니다.
- 대체됨 - 대체된 실행 모드를 구성한 경우에만 발생합니다. 이후 워크플로 실행이 이를 CodeCatalyst 대체했기 때문에 워크플로 실행이 취소되었습니다.

워크플로우 상태 CodeCatalyst

워크플로는 다음 상태 중 하나를 가질 수 있습니다.

- 유효 - 워크플로를 실행할 수 있으며 트리거를 통해 활성화할 수 있습니다.

  워크플로를 유효한 것으로 표시하려면 다음 조건을 모두 충족해야 합니다.

  - 워크플로 정의 파일은 유효해야 합니다.

  - 워크플로에는 트리거, 푸시 트리거 또는 현재 브랜치의 파일을 사용하여 실행되는 푸시 트리거가 없어야 합니다. 자세한 내용은 트리거 및 브랜치에 대한 사용 지침 단원을 참조하십시오.
- 유효하지 않음 — 워크플로의 정의 파일이 유효하지 않습니다. 워크플로는 수동으로 실행하거나 트리거를 통해 자동으로 실행할 수 없습니다. 유효하지 않은 워크플로는 워크플로 정의에 다음과 같이 표시됩니다.*n* CodeCatalyst 콘솔의 오류 메시지 (또는 이와 유사한 메시지).

  워크플로가 유효하지 않은 것으로 표시되려면 다음 조건이 충족되어야 합니다.

  - 워크플로 정의 파일을 잘못 구성해야 합니다.

    잘못 구성된 워크플로 정의 파일을 수정하려면 을 참조하십시오. "워크플로 정의에 다음과 같은 문제가 있는 문제를 해결하려면 어떻게 해야 합니까?*n* 오류" 오류를 해결합니까?
- 비활성 - 워크플로 정의는 유효하지만 수동으로 실행하거나 트리거를 통해 자동으로 실행할 수 없습니다.

  워크플로를 비활성으로 표시하려면 다음 조건을 모두 충족해야 합니다.

  - 워크플로 정의 파일은 유효해야 합니다.
  - 워크플로 정의 파일에는 현재 워크플로 정의 파일이 있는 브랜치와 다른 브랜치를 지정하는 푸시 트리거가 포함되어야 합니다. 자세한 내용은 트리거 및 브랜치에 대한 사용 지침 단원을 참조하십시오.

    워크플로를 비활성에서 활성으로 전환하려면 을 참조하십시오"워크플로가 비활성 상태입니다." 라는 메시지를 수정하려면 어떻게 해야 합니까?.

    > **Note**
    >
    > 비활성 상태인 워크플로가 많은 경우 보기에서 필터링할 수 있습니다. 비활성 워크플로를 필터링하려면 워크플로 페이지 상단의 워크플로 필터링 필드를 선택하고 상태를 선택한 다음 상태를 선택합니다. = 같지 않음, 선택하세요 INACTIVE.

> **Note**
>
> 워크플로에서 나중에 제거할 리소스 (예: 패키지 저장소) 를 지정하는 경우 이 변경 내용을 CodeCatalyst 감지하지 못하고 계속 워크플로를 유효한 것으로 표시합니다. 이러한 유형의 문제는 워크플로가 실행될 때 발견될 수 있습니다.

워크플로우 YAML 정의

다음은 워크플로 정의 파일에 대한 참조 문서입니다.

워크플로 정의 파일은 워크플로를 설명하는 YAML 파일입니다. 기본적으로 파일은 소스 리포지토리의 루트에 있는 `~/.codecatalyst/workflows/` 폴더에 저장됩니다. 파일 확장자는.yml 또는.yaml일 수 있으며 확장자는 소문자여야 합니다.

워크플로 정의 파일을 만들고 편집하려면 vim과 같은 편집기를 사용하거나 콘솔의 시각적 편집기 또는 편집기를 사용할 수 있습니다. CodeCatalyst YAML 자세한 내용은 CodeCatalyst 콘솔의 비주얼 및 YAML 에디터 사용 단원을 참조하십시오.

> **Note**
>
> 다음에 나오는 대부분의 YAML 속성에는 비주얼 편집기에 해당하는 UI 요소가 있습니다. UI 요소를 검색하려면 Ctrl+F를 사용합니다. 요소가 관련 YAML 속성과 함께 나열됩니다.

주제

워크플로 정의 파일의 예

다음은 간단한 워크플로 정의 파일의 예입니다. 여기에는 몇 가지 최상위 속성, `Triggers` 섹션 및 두 개의 작업 (및) 이 있는 `Actions` 섹션이 포함됩니다. `Build` `Test` 자세한 내용은 워크플로우 정의 파일 정보 단원을 참조하십시오.

```
Name: MyWorkflow
SchemaVersion: 1.0
RunMode: QUEUED
Triggers:
  - Type: PUSH
    Branches:
      - main
Actions:
```

```
  Build:
    Identifier: aws/build@v1
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: docker build -t MyApp:latest .
  Test:
    Identifier: aws/managed-test@v1
    DependsOn:
      - Build
    Inputs:
      Sources:
        - WorkflowSource
    Configuration:
      Steps:
        - Run: npm install
        - Run: npm run test
```

구문 지침 및 규칙

이 섹션에서는 워크플로 정의 파일의 구문 규칙과 이 참조 문서에 사용되는 이름 지정 규칙에 대해 설명합니다.

YAML구문 가이드라인

워크플로 정의 파일은 YAML1.1 사양에 따라 작성되므로 해당 사양에서 허용되는 모든 항목이 워크플로우에서도 허용됩니다YAML. YAML 처음 사용하는 경우 올바른 YAML 코드를 제공하는지 확인할 수 있는 몇 가지 간단한 지침이 있습니다. YAML

- 대소문자 구분: 워크플로 정의 파일은 대소문자를 구분하므로 이 설명서에 표시된 대소문자를 사용해야 합니다.
- 특수 문자: 속성 값 앞에는,,,, &,,,, <, >{,, } [ ] * # ? | -, 및 등의 특수 문자가 포함된 따옴표나 큰 따옴표를 사용하는 것이 좋습니다. = ! % @ : ` ,

  따옴표를 포함하지 않으면 이전에 나열된 특수 문자가 예상치 못한 방식으로 해석될 수 있습니다.
- 속성 이름: 속성 이름 (속성 값과 반대) 은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-) 및 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 속성 이름에 특수 문자와 공백을 사용할 때는 따옴표나 큰따옴표를 사용할 수 없습니다.

허용되지 않음:

`'My#Build@action'`

`My#Build@action`

`My Build Action`

허용:

`My-Build-Action_1`

- 이스케이프 코드: 속성 가치에 이스케이프 코드 (예: \n 또는\t) 가 포함된 경우 다음 가이드라인을 따르세요.
  - 이스케이프 코드를 문자열로 반환하려면 작은따옴표를 사용하십시오. 예를 들어`'my string \n my string'`, 는 문자열을 `my string \n my string` 반환합니다.
  - 큰따옴표를 사용하여 이스케이프 코드를 파싱합니다. 예를 들어`"my string \n my new line"`, 는 다음을 반환합니다.

    ```
    my string
    my new line
    ```
- 댓글: 댓글 앞에 를 붙입니다. #

  예제:

  ```
  Name: MyWorkflow
  # This is a comment.
  SchemaVersion: 1.0
  ```
- 트리플 대시 (`---`): `---` 코드에 사용하지 마세요. YAML CodeCatalyst 뒤에 오는 모든 항목을 무시합니다. `---`

이름 지정 규칙

이 가이드에서는 속성 및 섹션이라는 용어를 사용하여 워크플로 정의 파일의 기본 항목을 참조합니다.

- 속성은 콜론 (`:`) 이 포함된 모든 항목입니다. 예를 들어, 다음 코드 스니펫에서는 다음이 모두 속성입니다.`Name`,,, `SchemaVersion` `RunModeTriggers`, `Type` 및. `Branches`
- 섹션은 하위 속성이 있는 모든 속성입니다. 다음 코드 스니펫에는 섹션이 하나 있습니다. `Triggers`

> **Note**
>
> 이 가이드에서 '섹션'은 상황에 따라 '속성'이라고도 하며, 그 반대의 경우도 마찬가지입니다.

```
Name: MyWorkflow
SchemaVersion: 1.0
RunMode: QUEUED
Triggers:
  - Type: PUSH
    Branches:
      - main
```

최상위 속성

다음은 워크플로 정의 파일의 최상위 속성에 대한 참조 문서입니다.

```
# Name
Name: workflow-name

# Schema version
SchemaVersion: 1.0

# Run mode
RunMode: QUEUED|SUPERSEDED|PARALLEL

# Compute
Compute:
...

# Triggers
Triggers:
...

# Actions
Actions:
...
```

Name

(필수)

워크플로의 이름입니다. 워크플로 이름은 워크플로 목록에 표시되며 알림 및 로그에 언급됩니다. 워크플로 이름과 워크플로 정의 파일 이름이 일치할 수도 있고 이름을 다르게 지정할 수도 있습니다. 워크플로 이름은 고유하지 않아도 됩니다. 워크플로 이름은 영숫자 (a-z, A-Z, 0-9), 하이픈 (-) 및 밑줄 (_) 로 제한됩니다. 공백은 허용되지 않습니다. 워크플로 이름에 특수 문자와 공백을 사용할 때는 따옴표를 사용할 수 없습니다.

해당 UI: 비주얼 에디터/워크플로 속성/워크플로 이름

SchemaVersion

(필수)

워크플로 정의의 스키마 버전. 현재 유일한 유효 값은 `1.0`입니다.

해당 UI: 없음

RunMode

(선택 사항)

다중 실행을 CodeCatalyst 처리하는 방법 다음 값 중 하나를 사용할 수 있습니다.

- `QUEUED`— 여러 실행이 대기열에 추가되어 차례로 실행됩니다. 대기열에 최대 50개의 런을 넣을 수 있습니다.
- `SUPERSEDED`— 여러 실행이 대기열에 추가되어 차례로 실행됩니다. 대기열에는 한 번만 실행할 수 있으므로 두 실행이 같은 대기열에 있는 경우 이후 실행이 이전 실행을 대체 (인계) 하여 이전 실행이 취소됩니다.
- `PARALLEL`— 여러 실행이 동시에 발생합니다.

이 속성을 생략할 경우 기본값은 입니다. `QUEUED`

자세한 내용은 실행의 대기열 동작 구성 단원을 참조하십시오.

해당 UI: 비주얼 에디터/워크플로 속성/고급/실행 모드

Compute

(선택 사항)

워크플로 작업을 실행하는 데 사용되는 컴퓨팅 엔진입니다. 워크플로 수준 또는 작업 수준에서 컴퓨팅을 지정할 수 있지만 둘 다에서 지정할 수는 없습니다. 워크플로 수준에서 지정된 경우 컴퓨팅 구성은 워크플로에 정의된 모든 작업에 적용됩니다. 워크플로 수준에서는 동일한 인스턴스에서 여러 작업을 실행할 수도 있습니다. 자세한 내용은 작업 간 컴퓨팅 공유 단원을 참조하십시오.

컴퓨팅에 대한 자세한 내용은 을 참조하십시오컴퓨팅 및 런타임 이미지 구성.

해당 UI: 없음

```
Name: MyWorkflow
SchemaVersion: 1.0
...
Compute:
  Type: EC2 | Lambda
  Fleet: fleet-name
  SharedInstance: true | false
```

Type

(Compute/Type)

(Compute설정된 경우 필수)

컴퓨팅 엔진 유형. 다음 값 중 하나를 사용할 수 있습니다.

- EC2(비주얼 에디터) 또는 EC2 (YAML에디터)

  작업 실행 중 유연성을 위해 최적화되었습니다.
- Lambda (비주얼 에디터) 또는 (에디터) Lambda YAML

  작업 시작 속도를 최적화했습니다.

컴퓨팅 유형에 대한 자세한 정보는 컴퓨팅 유형을 참고하세요.

해당 UI: 비주얼 에디터/워크플로 속성/고급/컴퓨팅 유형

Fleet

(Compute/Fleet)

(선택 사항)

워크플로 또는 워크플로 작업을 실행할 컴퓨터 또는 플릿을 지정합니다. 온디맨드 플릿을 사용하면 작업이 시작되면 워크플로에서 필요한 리소스를 프로비저닝하고 작업이 완료되면 시스템이 폐기됩니다. 온디맨드 플릿의 예:,. `Linux.x86-64.Large` `Linux.x86-64.XLarge` 온디맨드 플릿에 대한 자세한 내용은 을 참조하십시오. 온디맨드 플릿 속성

프로비전된 플릿을 사용하면 워크플로 작업을 실행할 전용 컴퓨터 세트를 구성합니다. 이러한 시스템은 유휴 상태로 유지되므로 작업을 즉시 처리할 수 있습니다. 프로비전된 플릿에 대한 자세한 내용은 을 참조하십시오. 프로비저닝된 플릿 속성

생략된 경우 기본값은 `Fleet` 입니다. `Linux.x86-64.Large`

컴퓨팅 플릿에 대한 자세한 내용은 을 참조하십시오. 컴퓨팅 플릿

해당 UI: 비주얼 에디터/워크플로 속성/고급/컴퓨트 플릿

SharedInstance

(Compute/SharedInstance)

(선택 사항)

작업에 대한 컴퓨팅 공유 기능을 지정하십시오. 컴퓨팅 공유를 사용하면 워크플로의 작업이 동일한 인스턴스 (런타임 환경 이미지) 에서 실행됩니다. 다음 값 중 하나를 사용할 수 있습니다.

- `TRUE`런타임 환경 이미지가 워크플로 작업 간에 공유됨을 의미합니다.
- `FALSE`워크플로의 각 작업에 대해 별도의 런타임 환경 이미지가 시작되어 사용되므로 추가 구성 없이는 아티팩트 및 변수와 같은 리소스를 공유할 수 없습니다.

컴퓨팅 공유에 대한 자세한 내용은 을 참조하십시오작업 간 컴퓨팅 공유.

해당 UI: 없음

Triggers

(선택 사항)

이 워크플로에 대한 하나 이상의 트리거 시퀀스. 트리거가 지정되지 않은 경우 워크플로를 수동으로 시작해야 합니다.

트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.

해당 UI: 비주얼 에디터/워크플로 다이어그램/트리거

```
Name: MyWorkflow
SchemaVersion: 1.0
...
Triggers:
  - Type: PUSH
    Branches:
      - branch-name
    FilesChanged:
      - folder1/file
      - folder2/

  - Type: PULLREQUEST
    Events:
      - OPEN
      - CLOSED
      - REVISION
    Branches:
      - branch-name
    FilesChanged:
      - file1.txt

  - Type: SCHEDULE
    # Run the workflow at 10:15 am (UTC+0) every Saturday
    Expression: "15 10 ? * 7 *"
    Branches:
      - branch-name
```

Type

(Triggers/Type)

(설정된 경우 필수) `Triggers`

트리거 유형을 지정합니다. 다음 값 중 하나를 사용할 수 있습니다.

- 푸시 (비주얼 에디터) 또는 PUSH (YAML에디터)

  푸시 트리거는 변경 내용이 소스 저장소에 푸시될 때 워크플로 실행을 시작합니다. 워크플로 실행 시 푸시하려는 브랜치 (즉, 대상 브랜치) 의 파일이 사용됩니다.
- 풀 리퀘스트 (비주얼 에디터) 또는 PULLREQUEST (YAML에디터)

풀 리퀘스트 트리거는 소스 리포지토리에서 풀 리퀘스트를 열거나 업데이트하거나 닫을 때 워크플로 실행을 시작합니다. 워크플로 실행에서는 가져오는 브랜치 (즉, 소스 브랜치) 의 파일을 사용합니다.

- 스케줄 (비주얼 에디터) 또는 SCHEDULE (YAML에디터)

  스케줄 트리거는 지정한 cron 표현식으로 정의된 일정에 따라 워크플로가 실행됩니다. 브랜치의 파일을 사용하여 소스 리포지토리의 각 브랜치에 대해 별도의 워크플로 실행이 시작됩니다. (트리거가 활성화되는 브랜치를 제한하려면 Branch 필드 (시각적 편집기) 또는 **Branches** 속성 (YAML편집기) 을 사용하십시오.)

  스케줄 트리거를 구성할 때는 다음 가이드라인을 따르십시오.

  - 워크플로우당 하나의 스케줄 트리거만 사용하십시오.
  - CodeCatalyst 스페이스에 여러 워크플로를 정의한 경우 동시에 시작하도록 10개 이하로 예약하는 것이 좋습니다.
  - 실행 간격을 충분히 두고 트리거의 cron 표현식을 구성해야 합니다. 자세한 내용은 Expression 단원을 참조하십시오.

예제는 예: 워크플로의 트리거 섹션을 참조하세요.

해당 UI: 비주얼 에디터/워크플로 다이어그램/트리거/트리거 유형

Events

(Triggers/Events)

(트리거가 로 설정된 경우 필수) Type PULLREQUEST

워크플로 실행을 시작할 풀 요청 이벤트의 유형을 지정합니다. 유효한 값은 다음과 같습니다.

- 풀 리퀘스트가 생성됨 (비주얼 에디터) 또는 OPEN (YAML에디터)

  풀 리퀘스트가 생성되면 워크플로 실행이 시작됩니다.
- 풀 리퀘스트가 종료됨 (비주얼 에디터) 또는 CLOSED (YAML에디터)

  풀 리퀘스트가 종료되면 워크플로 실행이 시작됩니다. CLOSED이벤트의 동작은 까다롭기 때문에 예제를 통해 가장 잘 이해할 수 있습니다. 자세한 내용은 예: 풀, 브랜치, CLOSED '이벤트가 있는 트리거 섹션을 참조하세요.
- 풀 리퀘스트 (비주얼 에디터) 또는 **REVISION** (YAML에디터) 가 새롭게 수정되었습니다.

풀 리퀘스트에 대한 수정이 생성되면 워크플로 실행이 시작됩니다. 풀 리퀘스트가 생성되면 첫 번째 리비전이 생성됩니다. 그 이후에는 누군가가 풀 리퀘스트에 지정된 소스 브랜치에 새 커밋을 푸시할 때마다 새 리비전이 생성됩니다. 풀 리퀘스트 트리거에 REVISION 이벤트를 포함하면 OPEN 이벤트를 생략할 수 있습니다. 는 의 상위 REVISION 집합이기 때문입니다. OPEN

동일한 풀 리퀘스트 트리거에 여러 이벤트를 지정할 수 있습니다.

예제는 예: 워크플로의 트리거 섹션을 참조하세요.

해당 UI: 비주얼 에디터/워크플로 다이어그램/트리거/풀 리퀘스트용 이벤트

Branches

(Triggers/Branches)

(선택 사항)

워크플로우 실행 시작 시기를 알기 위해 트리거가 모니터링하는 소스 리포지토리의 브랜치를 지정하세요. 정규식 패턴을 사용하여 브랜치 이름을 정의할 수 있습니다. 예를 들어 로 시작하는 모든 브랜치를 main.* 일치시키는 데 사용합니다. main

지정할 브랜치는 트리거 유형에 따라 다릅니다.

- 푸시 트리거의 경우 푸시하려는 브랜치, 즉 대상 브랜치를 지정하세요. 일치하는 브랜치의 파일을 사용하여 일치하는 브랜치당 한 번의 워크플로 실행이 시작됩니다.

  예: main.*, mainline
- 풀 리퀘스트 트리거의 경우 푸시하려는 브랜치, 즉 대상 브랜치를 지정하세요. 일치하는 브랜치가 아닌 소스 브랜치의 소스 파일과 워크플로 정의 파일을 사용하여 일치하는 브랜치당 한 번의 워크플로 실행이 시작됩니다.

  예:main.*,mainline, v1\-.* (로 시작하는 브랜치와 일치v1-)
- 스케줄 트리거의 경우 예약 실행에서 사용할 파일이 포함된 브랜치를 지정하십시오. 일치하는 브랜치의 워크플로 정의 파일과 소스 파일을 사용하여 일치하는 브랜치당 한 번의 워크플로 실행이 시작됩니다.

  예: main.*, version\-1\.0

> **Note**
>
> 브랜치를 지정하지 않으면 트리거는 소스 리포지토리의 모든 브랜치를 모니터링하고 다음 위치에 있는 워크플로 정의 파일 및 소스 파일을 사용하여 워크플로 실행을 시작합니다.
>
> - 푸시하려는 브랜치 (푸시 트리거용). 자세한 내용은 예: 간단한 코드 푸시 트리거 단원을 참조하십시오.
> - 가져오려는 브랜치 (풀 리퀘스트 트리거용). 자세한 내용은 예: 간단한 풀 리퀘스트 트리거 단원을 참조하십시오.
> - 모든 브랜치 (스케줄 트리거용) 소스 리포지토리의 브랜치당 한 번의 워크플로 실행이 시작됩니다. 자세한 내용은 예: 간단한 스케줄 트리거 단원을 참조하십시오.

브랜치 및 트리거에 대한 자세한 내용은 을 참조하십시오트리거 및 브랜치에 대한 사용 지침.

더 많은 예제는 예: 워크플로의 트리거를 참조합니다.

해당 UI: 비주얼 에디터/워크플로 다이어그램/트리거/브랜치

FilesChanged

(Triggers/FilesChanged)

(트리거가 또는 로 설정된 경우 선택 사항입니다. Type PUSH PULLREQUEST Type트리거가 로 설정된 경우 지원되지 않습니다SCHEDULE.

워크플로우 실행 시작 시기를 알기 위해 트리거가 모니터링하는 소스 저장소의 파일 또는 폴더를 지정하십시오. 정규 표현식을 사용하여 파일 이름이나 경로를 일치시킬 수 있습니다.

예제는 예: 워크플로의 트리거 섹션을 참조하세요.

해당 UI: 비주얼 에디터/워크플로 다이어그램/트리거/파일 변경됨

Expression

(Triggers/Expression)

(트리거가 로 설정된 경우 필수) Type SCHEDULE

예약된 워크플로를 실행하려는 시기를 설명하는 cron 표현식을 지정하십시오.

의 크론 표현식은 각 필드를 공백으로 구분하는 다음과 같은 6개 필드 구문을 CodeCatalyst 사용합니다.

minutes hours days-of-month month days-of-week year

크론 표현식의 예

분	시간	한 달의 요일	월	요일	연도	의미
0	0	?	*	MON-FRI	*	매주 월요일부터 금요일까지 자정 (UTC+0) 에 워크플로를 실행합니다.
0	2	*	*	?	*	매일 오전 2시 (UTC+0) 에 워크플로를 실행합니다.
15	22	*	*	?	*	매일 오후 10시 15분 (UTC+0) 에 워크플로를 실행합니다.
0/30	22-2	?	*	SAT-SUN	*	토요일부터 일요일까지 시작일 오후 10시부터 다음 날 오전 2시 (UTC+0) 사

분	시간	한 달의 요일	월	요일	연도	의미
						이에 30분마다 워크플로를 실행합니다.
45	13	L	*	?	2023-2027	2023년부터 2027년까지 매월 마지막 날 오후 1시 45분 (UTC +0) 에 워크플로를 실행합니다.

에서 크론 표현식을 지정할 때는 다음 CodeCatalyst 지침을 준수해야 합니다.

- 트리거당 SCHEDULE 하나의 cron 표현식을 지정하십시오.
- 편집기에서 cron 표현식을 큰따옴표 () " 로 묶습니다. YAML
- 시간을 협정 세계시 () 로 지정합니다. UTC 다른 시간대는 지원되지 않습니다.
- 실행 간격을 최소 30분으로 설정하세요. 더 빠른 케이던스는 지원되지 않습니다.
- 다음을 지정하십시오.*days-of-month* 또는 *days-of-week* 필드를 모두 포함하되 둘 다 할 수는 없습니다. 필드 중 하나에 값이나 별표 (*) 를 지정하는 경우 다른 필드에는 물음표 (?) 를 사용해야 합니다. 별표는 '모두'를 의미하고 물음표는 '모두'를 의미합니다.

cron 표현식의 추가 예와 ? *L, 및 같은 와일드카드에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 Cron 표현식 참조를 참조하십시오. EventBridge 와 에서의 크론 표현식은 정확히 같은 방식으로 CodeCatalyst 작동합니다.

스케줄 트리거에 대한 예는 을 참조하십시오. 예: 워크플로의 트리거

해당 UI: 비주얼 에디터/워크플로 다이어그램/트리거/스케줄

작업

이 워크플로에 대한 하나 이상의 작업 시퀀스. CodeCatalyst 다양한 유형의 기능을 제공하는 빌드 및 테스트 작업과 같은 여러 작업 유형을 지원합니다. 각 작업 유형에는 다음이 포함됩니다.

- 액션의 고유한 하드 코딩된 ID를 나타내는 `Identifier` 속성입니다. 예를 들어, 빌드 작업을 `aws/build@v1` 식별합니다.
- 해당 작업과 관련된 속성을 포함하는 `Configuration` 섹션.

각 작업 유형에 대한 자세한 내용은 을 참조하십시오작업 유형. 작업 유형항목에는 각 작업에 대한 설명서로 연결되는 링크가 있습니다.

다음은 워크플로 정의 파일에 있는 작업 및 작업 그룹에 대한 YAML 참조입니다.

```
Name: MyWorkflow
SchemaVersion: 1.0
...
Actions:
  action-or-gate-name:
    Identifier: identifier
    Configuration:
    ...
  #Action groups
  action-group-name:
    Actions:
      ...
```

action-or-gate-name

(Actions/*action-or-gate-name*)

(필수)

Replace *action-name* 액션에 부여하려는 이름을 사용합니다. 작업 이름은 워크플로 내에서 고유해야 하며 영숫자, 하이픈, 밑줄만 포함해야 합니다. 구문 규칙에 대한 자세한 내용은 을 참조하십시오. YAML구문 가이드라인

제한을 비롯한 작업의 이름 지정 방법에 대한 자세한 내용은 를 참조하십시오. action-or-gate-name

해당 UI: 비주얼 에디터/*action-name*/구성 탭/ 작업 이름 또는 작업 표시 이름

action-group-name

(Actions/*action-group-name*)

(선택 사항)

작업 그룹에는 하나 이상의 작업이 포함됩니다. 작업을 작업 그룹으로 그룹화하면 워크플로를 체계적으로 구성하고 여러 그룹 간의 종속성을 구성할 수 있습니다.

Replace *action-group-name* 작업 그룹에 지정하려는 이름을 사용합니다. 작업 그룹 이름은 워크플로 내에서 고유해야 하며 영숫자, 하이픈, 밑줄만 포함해야 합니다. 구문 규칙에 대한 자세한 내용은 을 참조하십시오. YAML구문 가이드라인

작업 그룹에 대한 자세한 내용은 을 참조하십시오작업을 작업 그룹으로 그룹화.

해당 UI: 없음

에서 문제가 있는 작업을 추적하고 정리하세요. CodeCatalyst

에서는 CodeCatalyst 프로젝트와 관련된 기능, 버그 및 기타 작업을 모니터링할 수 있습니다. 각 작업은 이슈라는 별개의 기록에 보관됩니다. 작업 체크리스트를 추가하여 문제를 더 작은 목표로 나눌 수 있습니다. 각 이슈에는 설명, 담당자, 상태 및 기타 속성이 있을 수 있으며, 이러한 속성을 검색하고, 그룹화하고, 필터링할 수 있습니다. 기본 보기를 사용하여 문제를 보거나 사용자 지정 필터링, 정렬 또는 그룹화를 사용하여 자체 보기를 만들 수 있습니다. 문제와 관련된 개념에 대한 자세한 내용은 을 참조하십시오이슈 개념. 첫 번째 호를 만드는 방법을 알아보려면 을 참조하십시오에서 이슈 생성 CodeCatalyst.

이슈를 사용하는 팀이 사용할 수 있는 한 가지 워크플로는 다음과 같습니다.

Jorge Souza는 프로젝트를 진행하는 개발자입니다. 그와 그의 동료 프로젝트 멤버인 리 후안, 마테오 잭슨, 왕 시울란은 협업을 통해 어떤 작업을 해야 할지 결정합니다. 그와 동료 개발자들은 매일 왕 시울란이 이끄는 동기화 회의를 개최합니다. 팀원들이 보드에 대해 보는 관점 중 하나로 이동해 보드를 끌어올립니다. 뷰를 만들면 사용자와 팀이 필터, 그룹화, 이슈 정렬을 저장하여 지정된 기준에 맞는 이슈를 쉽게 볼 수 있습니다. 뷰에는 담당자별로 그룹화되고 우선 순위별로 정렬된 문제가 포함되어 있어 각 개발자의 가장 중요한 문제와 문제 상태를 보여줍니다. 완료해야 할 작업이 배정되면 호르헤는 각 작업에 대한 이슈를 생성하여 작업을 계획합니다. 이슈를 생성할 때 Jorge는 적절한 상태, 우선 순위 및 작업 추정 작업을 선택할 수 있습니다. 규모가 큰 문제의 경우 Jorge는 이슈에 작업을 추가하여 작업을 더 작은 목표로 나눕니다. 호르헤는 즉시 시작할 계획이 없기 때문에 백로그와 같은 초안 상태로 이슈를 생성합니다. 초안 상태의 이슈는 초안 보기에 표시되며, 여기서 계획을 세우고 우선 순위를 지정할 수 있습니다. 작업을 시작할 준비가 되면 Jorge는 해당 이슈의 상태를 다른 카테고리 (시작되지 않음, 시작됨 또는 완료됨) 의 상태로 업데이트하여 해당 이슈를 이사회로 옮깁니다. 각 작업이 진행됨에 따라 팀은 제목, 상태, 담당자, 레이블, 우선 순위 및 추정 기준으로 필터링하여 지정된 매개변수와 일치하는 특정 문제 또는 유사한 문제를 찾을 수 있습니다. Jorge와 그의 팀은 게시판을 사용하여 각 이슈에 대해 완료된 작업 수를 확인하고 작업이 완료될 때까지 각 문제를 한 상태에서 다음 상태로 드래그하여 day-to-day 진행 상황을 추적할 수 있습니다. 프로젝트가 진행됨에 따라 완료된 호는 완료됨 상태로 누적됩니다. Wang Xiulan은 개발자가 현재 및 향후 작업과 관련된 문제에 집중할 수 있도록 빠른 보관 버튼을 사용하여 문서를 보관함으로써 보기에서 제거하기로 결정합니다.

프로젝트를 진행하는 개발자들은 작업을 계획할 때 정렬 기준 및 그룹화 기준을 선택하여 백로그에서 게시판으로 옮기고 싶은 이슈를 찾습니다. 우선 순위가 가장 높은 고객 요청을 기준으로 보드에 이슈를 추가하여 고객 요청 레이블을 기준으로 보드를 그룹화하고 우선 순위별로 정렬할 수도 있습니다. 또한 예상치를 기준으로 정렬하여 달성할 수 있는 양의 작업을 수행하고 있는지 확인할 수도 있습니다. 프로젝트 매니저인 Saanvi Sarkar는 백로그를 정기적으로 검토하고 정리하여 프로젝트 성공에 대한 각 문제의 중요성을 우선 순위에 정확하게 반영하도록 합니다.

주제

이슈 개념

이슈를 생성하면 프로젝트 내에서 수행 중인 작업을 빠르고 효율적으로 추적할 수 있습니다. 이슈를 사용하여 일상적인 동기화 미팅에서 업무에 대해 논의하고, 업무의 우선순위를 정하는 등의 작업을 할 수 있습니다.

이 페이지에는 이슈를 효과적으로 사용하는 데 도움이 되는 개념 목록이 포함되어 있습니다.
CodeCatalyst

진행 중인 이슈

활성 이슈는 초안 상태이거나 보관되지 않은 모든 이슈입니다. 즉, 활성 이슈는 상태가 시작되지 않음, 시작됨, 완료됨과 같은 상태 카테고리에 속하는 이슈입니다. 상태 및 상태 범주에 대한 자세한 내용은 을 참조하십시오상태 및 상태 카테고리.

기본 활성 이슈 보기에서 프로젝트의 모든 활성 이슈를 볼 수 있습니다.

보관된 이슈

보관된 이슈는 더 이상 프로젝트와 관련이 없는 이슈입니다. 예를 들어 이슈가 완료되어 완료 열에 더 이상 표시되지 않아도 되는 경우 또는 오류로 생성된 이슈를 보관할 수 있습니다. 보관된 이슈는 필요한 경우 보관을 취소할 수 있습니다.

담당자

담당자는 문제가 배정된 사람입니다. 검색할 때 목록에 나타나지 않는 사람은 프로젝트에 추가되지 않은 것입니다. 이들을 추가하려면 을 참조하십시오프로젝트에 사용자 초대하기. 여러 담당자가 이슈에 참여할 수 있도록 하려면 을 참조하십시오. 여러 담당자 활성화 또는 비활성화 담당자가 여러 명인 이슈는 각각 담당자 중 한 명을 나타내는 서로 다른 색상의 아바타와 함께 보드에 표시됩니다.

사용자 지정 필드

사용자 지정 필드를 사용하면 프로젝트 내에서 문제를 추적하고 유지 관리하는 데 필요한 사항에 따라 이슈의 다양한 속성을 사용자 지정할 수 있습니다. 예를 들어 로드맵용 필드, 특정 마감일 또는 요청자 필드를 추가할 수 있습니다.

추정치

애자일 개발에서는 추정치를 스토리포인트라고 합니다. 문제 추정치를 사용하여 문제의 모호성 및 복잡성 외에도 필요한 작업량을 나타낼 수 있습니다. 위험도가 크고 난이도가 높으며 미지수가 있는 문제에는 더 높은 추정치를 사용하는 것을 고려해 보세요.

추정 유형 및 구성 방법에 대한 자세한 내용은 을 참조하십시오. 이슈 작업량 추정 구성

문제

이슈는 프로젝트와 관련된 작업을 추적하는 레코드입니다. 기능, 작업, 버그 또는 프로젝트와 관련된 기타 작업에 대한 이슈를 생성할 수 있습니다. 애자일 개발을 사용하는 경우 에픽 또는 사용자 스토리를 설명하는 이슈가 될 수도 있습니다.

레이블

레이블은 이슈를 그룹화, 정렬 및 필터링하는 데 사용됩니다. 새 레이블 이름을 입력하거나 채워진 목록에서 레이블 중 하나를 선택할 수 있습니다. 이 목록은 프로젝트에서 최근에 사용한 라벨로 구성되어 있습니다. 이슈에는 여러 라벨이 있을 수 있으며, 이슈에서 라벨을 제거할 수 있습니다. 라벨을 사용자 지정하려면 을 참조하십시오라벨로 작업 분류하기.

우선 순위

우선순위는 문제의 중요도 수준을 나타냅니다. 낮음, 보통, 높음, 우선 순위 없음의 네 가지 옵션이 있습니다.

상태 및 상태 카테고리

상태는 문제의 현재 상태이며, 시작부터 완료까지 수명 주기 전반에 걸쳐 문제의 진행 상황을 빠르게 확인하는 데 사용됩니다. 모든 이슈에는 상태가 있어야 하며 각 상태는 상태 카테고리에 속해야 합니다. 상태 카테고리는 상태를 정리하고 기본 이슈 보기를 채우는 데 사용됩니다.

에는 다섯 개의 기본 상태와 네 개의 상태 범주가 있습니다. CodeCatalyst 다른 상태는 만들 수 있지만 다른 상태 범주는 만들 수 없습니다. 다음 목록에는 백로그 (초안), 할 일 (시작되지 않음), 진행 중 (시작됨), 검토 중 (시작됨), 완료 (완료) 등의 기본 상태와 해당 상태 범주가 괄호로 묶여 있습니다.

상태 작업에 대한 자세한 내용은 을 참조하십시오. 사용자 지정 상태로 작업 추적

Tasks

이슈에 작업을 추가하여 해당 이슈의 작업을 더 세분화하고 정리할 수 있습니다. 이슈 생성 시 이슈에 작업을 추가하거나 기존 이슈에 작업을 추가할 수 있습니다. 이슈를 볼 때 작업을 재정렬하거나, 제거하거나, 완료로 표시할 수 있습니다.

보기

CodeCatalyst 프로젝트의 이슈는 뷰에 표시됩니다. 보기는 이슈를 목록 형식으로 표시하는 그리드 뷰이거나 이슈를 이슈 상태별로 정리된 열에 타일로 표시하는 보드 뷰일 수 있습니다. 네 가지 기본 보기가 있으며 사용자 지정 그룹화, 필터링, 정렬을 사용하여 사용자 지정 보기를 만들 수 있습니다. 다음 목록에는 네 가지 기본 보기에 대한 세부 정보가 포함되어 있습니다.

- 초안 보기는 현재 작업 중이 아닌 문제를 보여주는 그리드 뷰입니다. 초안 상태 범주의 상태로 생성된 모든 문제가 이 보기에 표시됩니다. 팀은 이 보기를 사용하여 아직 정의 중이거나 할당 및 작업 대기 중인 문제를 확인할 수 있습니다.
- 활성 이슈 뷰는 현재 작업 중인 모든 이슈를 게시판 뷰로 볼 수 있습니다. 상태가 [시작되지 않음], [시작됨] 또는 [완료됨] 범주에 있는 모든 문제가 이 보기에 표시됩니다.
- 모든 이슈 뷰는 초안과 진행 중인 이슈를 포함하여 프로젝트의 모든 이슈를 표시하는 그리드 뷰입니다.
- 보관된 보기에는 보관된 모든 이슈가 표시됩니다.

이슈 관련 작업 추적

이슈를 사용하여 프로젝트 작업을 계획하고 추적할 수 있습니다. 각 호는 서로 다른 기록으로 보관되는 작업의 일부입니다. 이슈를 태스크로 분류하여 해당 이슈의 작업을 더 체계적으로 구성하고 추적할 수 있습니다. 또한 이슈 간 링크를 만들어 관련 업무를 추적하고, 라벨을 추가하여 업무를 체계화하고 분류하고, 이슈를 그룹화하고, 업무에 우선 순위를 할당하고, 업무가 차단되었는지 여부를 표시할 수 있습니다.

특정 이슈나 일련의 이슈에 대해 작업할 준비가 되면 작업을 예측하여 사용자에게 배정하고 코멘트를 추가하여 다른 사람들이 작업과 진행 상황을 이해할 수 있도록 할 수 있습니다. 이슈에 포함된 정보를 다른 형식으로 가져오는 데 도움이 되도록 이슈를 내보낼 수도 있습니다.

에서 이슈 생성 CodeCatalyst

개발팀은 작업을 추적하고 관리하는 데 도움이 되는 문제를 생성합니다. 필요에 따라 프로젝트 내에서 이슈를 생성할 수 있습니다. 예를 들어 코드의 변수 업데이트를 추적하는 이슈를 만들 수 있습니다. 프로젝트의 다른 사용자에게 이슈를 할당하고, 레이블을 사용하여 작업을 추적하는 등의 작업을 수행할 수 있습니다.

다음 지침에 따라 에서 이슈를 생성하세요 CodeCatalyst.

이슈를 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 이슈를 생성하려는 프로젝트로 이동합니다.
3. 프로젝트 홈페이지에서 이슈 만들기를 선택합니다. 또는 탐색 창에서 이슈를 선택합니다.
4. 이슈 생성을 선택합니다.

> **Note**
>
> 그리드 보기를 사용할 때 이슈를 인라인으로 추가할 수도 있습니다.

5. 이슈의 제목을 입력합니다.
6. (선택 사항) 설명을 입력합니다. 마크다운을 사용하여 서식을 추가할 수 있습니다.
7. (선택 사항) 문제의 상태, 우선 순위 및 추정을 선택합니다.

> **Note**
>
> 프로젝트의 예상 설정이 예상치 숨기기로 설정된 경우 예상 필드가 표시되지 않습니다.

8. (선택 사항) 이슈에 작업을 추가합니다. 작업을 사용하여 이슈 작업을 더 작은 목표로 세분화할 수 있습니다. 작업을 추가하려면 + 작업 추가를 선택합니다. 그런 다음 텍스트 필드에 작업 이름을 입력하고 Enter 키를 누릅니다. 작업을 추가한 후 확인란을 선택하여 작업을 완료로 표시하거나, 확인란 왼쪽에서 작업을 선택하고 드래그하여 작업의 순서를 변경할 수 있습니다.
9. (선택 사항) 기존 레이블을 추가하거나 새 레이블을 만든 다음 + 레이블 추가를 선택하여 추가합니다.

   a. 기존 라벨을 추가하려면 목록에서 라벨을 선택합니다. 필드에 검색어를 입력하여 프로젝트에서 해당 용어가 포함된 모든 라벨을 검색할 수 있습니다.

b. 새 레이블을 생성하여 추가하려면 만들려는 레이블의 이름을 검색 필드에 입력하고 Enter 키를 누릅니다.

10. (선택 사항) + 담당자 추가를 선택하여 담당자를 추가합니다. + Add me를 선택하여 자신을 담당자로 빠르게 추가할 수 있습니다.

> **Tip**
>
> Amazon Q에 문제를 할당하여 Amazon Q에서 문제 해결을 시도하도록 할 수 있습니다. 자세한 내용은 튜토리얼: CodeCatalyst 제너레이티브 AI 기능을 사용하여 개발 작업의 속도를 높이세요 단원을 참조하십시오. 이 기능은 미국 서부 (오레곤) 지역에서만 사용할 수 있습니다.
> 이 기능을 사용하려면 해당 공간에 제너레이티브 AI 기능을 활성화해야 합니다. 자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오.

11. (선택 사항) 기존 사용자 지정 필드를 추가하거나 새 사용자 지정 필드를 생성합니다. 이슈에는 사용자 지정 필드가 여러 개 있을 수 있습니다.

    a. 기존 사용자 지정 필드를 추가하려면 목록에서 사용자 지정 필드를 선택합니다. 필드에 검색어를 입력하여 프로젝트에서 해당 용어가 포함된 모든 사용자 지정 필드를 검색할 수 있습니다.

    b. 새 사용자 지정 필드를 만들어 추가하려면 만들려는 사용자 지정 필드의 이름을 검색 필드에 입력하고 Enter 키를 누릅니다. 그런 다음 만들려는 사용자 지정 필드 유형을 선택하고 값을 설정합니다.

12. 이슈 생성을 선택합니다. 오른쪽 하단에 알림이 표시됩니다. 이슈가 성공적으로 생성되면 이슈가 성공적으로 생성되었다는 확인 메시지가 나타납니다. 이슈가 성공적으로 생성되지 않은 경우 실패 이유가 포함된 오류 메시지가 나타납니다. 그런 다음 [Retry] 를 선택하여 문제를 편집하고 다시 생성하거나 [취소] 를 선택하여 문제를 삭제할 수 있습니다. 두 옵션 모두 알림을 무시합니다.

> **Note**
>
> 풀 리퀘스트를 생성할 때는 이슈에 연결할 수 없습니다. 하지만 생성 후 편집하여 풀 리퀘스트에 링크를 추가할 수 있습니다.

Amazon Q에 할당된 이슈를 생성하고 처리하는 모범 사례

이슈를 생성할 때 일부 이슈가 남아 있는 경우가 있습니다. 원인은 복잡하고 다양할 수 있습니다. 누가 이 문제를 해결해야 하는지 명확하지 않기 때문인 경우도 있습니다. 코드 베이스의 특정 부분에 대한 조사나 전문 지식이 필요한 문제도 있고, 작업에 가장 적합한 후보자가 다른 문제로 바쁜 경우도 있습니다. 다른 긴급한 작업을 먼저 처리해야 하는 경우가 종종 있습니다. 이러한 원인 중 일부 또는 전부로 인해 해결되지 않은 문제가 발생할 수 있습니다. CodeCatalyst 제목과 설명을 기반으로 문제를 분석할 수 있는 Amazon Q라는 제너레이티브 AI 어시스턴트와의 통합이 포함됩니다. Amazon Q에 문제를 할당하면 평가할 수 있는 초안 솔루션을 만들려고 시도합니다. 이를 통해 사용자와 팀은 주의가 필요한 문제에 집중하고 작업을 최적화할 수 있으며, Amazon Q는 즉시 해결할 리소스가 없는 문제에 대한 솔루션을 개발합니다.

> **Note**
>
> > **Note**
> >
> > Amazon Bedrock 제공: 자동 악용 탐지 AWS 기능을 구현합니다. 나를 위한 설명 작성, 콘텐츠 요약 생성, 작업 추천, Amazon Q를 사용하여 프로젝트에 기능 생성 또는 추가, Amazon Q Developer Agent의 소프트웨어 개발 기능에 대한 Amazon Q 이슈 할당 기능이 Amazon Bedrock에 구축되어 있기 때문에 사용자는 Amazon Bedrock에 구현된 제어 기능을 최대한 활용하여 안전, 보안 및 인공 지능 (AI) 의 책임 있는 사용을 강화할 수 있습니다.

Amazon Q는 간단한 문제와 간단한 문제에서 가장 잘 작동합니다. 최상의 결과를 얻으려면 평이한 언어를 사용하여 수행하려는 작업을 명확하게 설명하십시오. 다음은 Amazon Q가 처리하도록 최적화된 문제를 생성하는 데 도움이 되는 몇 가지 모범 사례입니다.

> **Important**
>
> 제너레이티브 AI 기능은 미국 서부 (오레곤) 지역에서만 사용할 수 있습니다.

- 단순하게 사용하세요. Amazon Q는 문제의 제목과 설명에서 설명할 수 있는 간단한 코드 변경 및 수정을 사용하는 것이 가장 좋습니다. 제목이 모호하거나 지나치게 화려하거나 모순되는 설명으로 문제를 지정하지 마십시오.

- 구체적으로 작성하세요. 문제 해결에 필요한 정확한 변경 사항에 대해 더 많은 정보를 제공할수록 Amazon Q에서 문제를 해결하는 솔루션을 만들 가능성이 높아집니다. 가능하면 변경하려는 이름, APIs 업데이트하려는 방법, 변경이 필요한 테스트 및 생각할 수 있는 기타 세부 정보와 같은 구체적인 세부 정보를 포함하십시오.
- Amazon Q에 문제를 할당하기 전에 문제의 제목과 설명에 모든 세부 정보가 포함되어 있는지 확인하십시오. Amazon Q에 이슈를 할당한 후에는 이슈의 제목이나 설명을 변경할 수 없으므로 Amazon Q에 이슈를 할당하기 전에 이슈에 필요한 모든 정보가 있는지 확인하십시오.
- 단일 소스 리포지토리에서 코드 변경이 필요한 이슈만 배정하십시오. Amazon Q는 의 단일 소스 리포지토리에 있는 코드에서만 작동할 수 CodeCatalyst 있습니다. 연결된 리포지토리는 지원되지 않습니다. Amazon Q에 이슈를 할당하기 전에 단일 소스 리포지토리의 변경만 필요한 문제인지 확인하십시오.
- Amazon Q에서 제안한 기본값을 사용하여 각 단계를 승인하십시오. 기본적으로 Amazon Q는 수행하는 각 단계에 대해 사용자의 승인을 요구합니다. 이를 통해 이슈뿐만 아니라 해당 이슈가 생성한 모든 풀 리퀘스트에 대한 코멘트를 통해 Amazon Q와 상호 작용할 수 있습니다. 이를 통해 Amazon Q에서 보다 인터랙티브한 경험을 제공하므로 접근 방식을 조정하고 문제 해결을 위해 생성하는 코드를 구체화하는 데 도움이 됩니다.

  > **Note**
  >
  > Amazon Q는 이슈 또는 풀 리퀘스트에 대한 개별 의견에 응답하지 않지만 접근 방식을 재고하거나 수정 사항을 생성하라는 요청을 받으면 검토합니다.

- Amazon Q에서 제안한 접근 방식을 항상 주의 깊게 검토하십시오. 접근 방식을 승인하면 Amazon Q에서 해당 접근 방식을 기반으로 코드를 생성하는 작업을 시작합니다. Amazon Q에 진행을 요청하기 전에 접근 방식이 올바르고 예상한 모든 세부 정보가 포함되어 있는지 확인하십시오.
- 검토를 거치기 전에 배포할 수 있는 기존 워크플로가 없는 경우에만 Amazon Q가 워크플로에서 작업하도록 허용하십시오. 프로젝트에 풀 리퀘스트 이벤트에서 실행을 시작하도록 구성된 워크플로가 있을 수 있습니다. 그렇다면 Amazon Q에서 워크플로 생성 또는 업데이트를 포함하여 생성하는 모든 pull 요청이 pull 요청에 포함된 워크플로의 실행을 시작할 YAML 수 있습니다. 가장 좋은 방법은 생성한 pull 요청을 검토 및 승인하기 전에 프로젝트에 이러한 워크플로를 자동으로 실행하는 워크플로가 없다는 확신이 들지 않는 한 Amazon Q가 워크플로 파일에서 작업하도록 허용하지 않는 것입니다.

자세한 내용은 제너레이티브 AI 기능 관리를 참조하십시오튜토리얼: CodeCatalyst 제너레이티브 AI 기능을 사용하여 개발 작업의 속도를 높이세요.

문제 추정

애자일 개발에서는 추정치를 스토리포인트라고 합니다. 문제 추정치를 사용하여 문제의 모호성 및 복잡성 외에도 필요한 작업량을 나타낼 수 있습니다. 위험도가 크고 난이도가 높으며 미지수가 있는 문제에는 더 높은 추정치를 사용하는 것을 고려해 보세요.

문제 추정을 시작하기 전에 먼저 프로젝트에 사용할 추정치 유형을 선택해야 합니다. 기본적으로 두 가지 유형 중에서 선택할 수 있습니다. 티셔츠 사이즈나 피보나치 시퀀싱을 효과적으로 사용하려면 팀에서 각 사이즈가 무엇을 나타내는지 조정해야 합니다. 각 추정치가 무엇을 의미하는지 함께 결정한 다음 해당 추정치를 각 이슈에 적용하기 시작하세요. 정기적으로 검토해 보세요.

다음 단계에 따라 에서 문제에 대한 작업량 추정 설정을 구성하십시오. CodeCatalyst

문제에 대한 작업량 추정을 구성하려면

1. 탐색 창에서 Issues를 선택합니다.
2. 활성 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 설정을 선택합니다.
3. 기본 설정 섹션의 추정치에서 추정 값이 표시되는 방식을 선택합니다. 사용 가능한 추정치 유형으로는 티셔츠 사이즈, 피보나치 시퀀싱, 추정치 숨기기 등이 있습니다. 프로젝트의 예상 설정이 추정치 숨기기로 설정된 경우 프로젝트의 이슈에 예상 필드가 표시되지 않습니다.

   추정 유형이 업데이트되면 데이터가 손실되지 않으며 모든 발행물의 추정 값이 자동으로 변환됩니다. 전환 매핑은 다음 표에 나와 있습니다.

티셔츠 사이즈	피보나치 수열
XS	1
XS	2
S	3
M	5
L	8
특대	13

이슈에 대한 예상치를 추가하거나 변경하려면 이슈를 편집할 수 있습니다.

에서 이슈에 대한 편집 및 공동 작업 CodeCatalyst

목차

이슈 편집

다음 단계에 따라 문제의 제목, 설명, 상태, 담당자, 우선순위, 추정치 또는 라벨을 수정하세요.

문제를 편집하려면

1. 편집하려는 이슈를 선택하여 이슈 세부 정보를 확인하세요. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
2. 이슈 제목을 편집하려면 제목을 선택하고 새 제목을 입력한 다음 Enter 키를 누릅니다.
3. 설명을 편집하려면 설명을 선택하고 새 설명을 입력한 다음 Enter 키를 누릅니다. 마크다운을 사용하여 서식을 추가할 수 있습니다.
4. 태스크에서 이슈에 대한 태스크를 보고 관리할 수 있습니다. 작업이 없는 경우 Amazon Q에서 문제를 분석하고 해당 이슈의 작업을 사용자에게 각각 할당할 수 있는 개별 항목으로 분류할 수 있는 작업을 추천하도록 할 수 있습니다. 자세한 내용은 이슈 관련 작업 관리 단원을 참조하십시오.
5. 상태, 예상 또는 우선 순위를 편집하려면 해당 드롭다운 메뉴에서 옵션을 선택합니다.
6. 라벨에서 기존 라벨을 추가하거나, 새 라벨을 만들거나, 라벨을 제거할 수 있습니다.

    a. 기존 라벨을 추가하려면 + 라벨 추가를 선택하고 목록에서 라벨을 선택합니다. 필드에 검색어를 입력하여 프로젝트에서 해당 용어가 포함된 모든 라벨을 검색할 수 있습니다.

    b. 새 레이블을 만들어 추가하려면 + 레이블 추가를 선택하고 검색 필드에 만들려는 레이블의 이름을 입력한 다음 Enter 키를 누릅니다.

c. 라벨을 제거하려면 제거하려는 라벨 옆의 X 아이콘을 선택합니다. 모든 이슈에서 라벨을 제거하면 해당 라벨이 이슈 설정의 라벨 섹션에 있는 미사용 라벨 섹션에 표시됩니다. 필터를 사용하거나 이슈에 라벨을 추가할 때 라벨 목록 끝에 사용하지 않은 라벨이 표시됩니다. 이슈 설정에서 모든 라벨 (사용 및 미사용) 과 해당 라벨이 있는 이슈의 개요를 확인할 수 있습니다.

7. 이슈를 할당하려면 담당자 섹션에서 + 담당자 추가를 선택한 다음 목록에서 담당자를 검색하여 선택합니다. + Add me를 선택하여 자신을 담당자로 빠르게 추가할 수 있습니다.
8. 첨부 파일에서 첨부 파일을 추가, 다운로드 또는 제거할 수 있습니다. 자세한 내용은 첨부 파일 작업 단원을 참조하십시오.
9. 풀 리퀘스트를 연결하려면 풀 리퀘스트 연결을 선택한 다음 목록에서 풀 리퀘스트를 선택하거나 해당 URL 또는 ID를 입력합니다. 풀 리퀘스트의 연결을 해제하려면 연결 해제 아이콘을 선택합니다.

> **Tip**
>
> 풀 리퀘스트 링크를 이슈에 추가한 후 연결된 풀 리퀘스트 목록에서 해당 ID를 선택하면 해당 링크를 빠르게 탐색할 수 있습니다. 풀 리퀘스트를 사용하여 이슈 게시판과 다른 프로젝트에 있는 풀 리퀘스트를 연결할 수 있지만, 해당 프로젝트의 멤버인 사용자만 해당 풀 리퀘스트를 보거나 탐색할 수 있습니다. URL

10. (선택 사항) 기존 사용자 지정 필드를 추가 및 설정하거나, 새 사용자 지정 필드를 만들거나, 사용자 지정 필드를 제거합니다. 이슈에는 사용자 지정 필드가 여러 개 있을 수 있습니다.

    a. 기존 사용자 지정 필드를 추가하려면 목록에서 사용자 지정 필드를 선택합니다. 필드에 검색어를 입력하여 프로젝트에서 해당 용어가 포함된 모든 사용자 지정 필드를 검색할 수 있습니다.

    b. 새 사용자 지정 필드를 만들어 추가하려면 만들려는 사용자 지정 필드의 이름을 검색 필드에 입력하고 Enter 키를 누릅니다. 그런 다음 만들려는 사용자 지정 필드 유형을 선택하고 값을 설정합니다.

    c. 사용자 지정 필드를 제거하려면 제거하려는 사용자 지정 필드 옆에 있는 X 아이콘을 선택합니다. 모든 이슈에서 사용자 지정 필드를 제거하면 사용자 지정 필드가 삭제되고 필터링할 때 더 이상 표시되지 않습니다.

첨부 파일 작업

이슈에 첨부 파일을 추가하여 관련 파일에 쉽게 접근할 수 CodeCatalyst 있도록 할 수 있습니다. 다음 절차를 사용하여 이슈의 첨부 파일을 관리하세요.

이슈에 추가된 첨부 파일 크기는 스페이스의 스토리지 할당량에 포함됩니다. 프로젝트의 첨부 파일 보기 및 관리에 대한 자세한 내용은 을 참조하십시오. 첨부 파일 보기 및 관리

> **Important**
>
> CodeCatalyst아마존은 이슈의 첨부 파일을 스캔하거나 분석하지 않습니다. 모든 사용자는 악성 코드나 콘텐츠를 포함할 가능성이 있는 문제에 첨부 파일을 추가할 수 있습니다. 첨부 파일 관리 및 악성 코드, 콘텐츠 또는 바이러스로부터 보호하는 방법에 관한 모범 사례를 사용자가 알고 있는지 확인하세요.

첨부 파일 추가, 다운로드 또는 제거하기

1. 첨부 파일을 관리하려는 문제를 선택합니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
2. 첨부 파일을 추가하려면 파일 업로드를 선택합니다. 운영 체제의 파일 탐색기에서 파일을 찾아 선택합니다. 파일을 첨부 파일로 추가하려면 [열기] 를 선택합니다. 최대 첨부 파일 크기 등의 할당량 정보는 을 참조하십시오내 이슈에 대한 할당량 CodeCatalyst.

   첨부 파일 이름 및 콘텐츠 유형에 대한 다음과 같은 제한 사항을 참고하십시오.

   - 다음 문자는 파일 이름에 사용할 수 없습니다.
     - 제어 문자: `0x00-0x1f` 및 `0x80-0x9f`
     - 예약 문자:/, ?<,>,\,:,*,|, 및 "
     - Unix 예약 파일 이름: 및 `. ..`
     - 후행 마침표 및 공백
     - 윈도우 예약 파일 이름: `CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, and LPT9`
   - 첨부 파일의 콘텐츠 유형은 다음 미디어 유형 패턴을 준수해야 합니다.

     ```
     media-type = type "/" [tree "."] subtype ["+" suffix]* [";" parameter];
     ```

     예: `text/html; charset=UTF-8`.
3. 첨부 파일을 다운로드하려면 다운로드하려는 첨부 파일 옆의 줄임표 메뉴를 선택하고 다운로드를 선택합니다.

4. 첨부 파일을 복사하려면 복사하려는 첨부 파일 옆의 줄임표 메뉴를 선택하고 복사를 선택합니다. URL URL URL
5. 첨부 파일을 제거하려면 제거하려는 첨부 파일 옆의 줄임표 메뉴를 선택하고 삭제를 선택합니다.

첨부 파일 보기 및 관리

이슈 설정에서 프로젝트의 이슈에 추가된 모든 첨부 파일이 있는 표를 볼 수 있습니다. 이 표에는 콘텐츠 유형, 추가된 시기, 추가된 문제, 상태, 파일 크기 등의 정보를 포함하여 각 첨부 파일의 세부 정보가 포함되어 있습니다.

이 표를 사용하면 완료되었거나 보관된 문제에 대한 대용량 첨부 파일을 쉽게 식별하여 제거하여 공간을 확보할 수 있습니다.

> **Important**
>
> CodeCatalyst아마존은 이슈의 첨부 파일을 스캔하거나 분석하지 않습니다. 어떤 사용자라도 악성 코드나 콘텐츠가 포함되어 있을 수 있는 문제에 첨부 파일을 추가할 수 있습니다. 첨부 파일 관리 및 악성 코드, 콘텐츠 또는 바이러스로부터 보호하는 방법에 관한 모범 사례를 사용자가 알고 있는지 확인하세요.

프로젝트의 모든 이슈 첨부 파일을 보고 관리하려면

1. 탐색 창에서 이슈를 선택합니다.
2. 줄임표 아이콘을 선택하고 설정을 선택합니다.
3. 첨부 파일 탭을 선택합니다.

이슈 관련 작업 관리

이슈에 작업을 추가하여 해당 이슈의 작업을 더 세분화하고, 구성하고, 추적할 수 있습니다. 직접 작업을 생성하거나 Amazon Q를 사용하여 문제 분석 및 복잡성을 기반으로 작업을 추천할 수 있습니다.

Amazon Q Developer는 애플리케이션을 이해, 구축, 확장 및 운영하는 데 도움이 되는 생성형 AI 기반 대화형 도우미입니다. AWS 개발을 가속화하기 위해 Amazon Q를 지원하는 모델을 고품질 AWS 콘텐츠로 보강하여 보다 완전하고 실행 가능하며 참조할 수 있는 답변을 제공합니다. AWS자세한 내용은 Amazon Q 개발자란 무엇입니까? 를 참조하십시오. Amazon Q 개발자 사용 설명서에서 확인할 수 있습니다.

이슈에 대한 태스크를 관리하려면

1. 작업을 관리하려는 이슈를 선택합니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
2. 태스크에서 이슈에 대한 작업을 보고 관리할 수 있습니다.

    1. 작업을 추가하려면 텍스트 필드에 작업 이름을 입력하고 Enter 키를 누릅니다.
    2. 문제에 대한 작업이 없는 경우 Amazon Q에서 문제를 분석하고 문제 제목, 설명, 문제의 복잡성 분석 및 리포지토리 코드를 기반으로 작업을 생성하도록 선택하고 권장 작업을 선택할 수 있습니다. 문제의 코드가 들어 있는 소스 리포지토리를 지정해야 합니다. 작업 추천 시작을 선택하여 작업 권장 사항 분석을 시작합니다. 해당 대화 상자가 닫힙니다. 권장 사항이 완료되면 권장 작업 보기를 선택하여 작업을 검토하고 작업 생성을 선택하기 전에 목록에 작업을 삭제 또는 추가하거나 권장 작업을 다시 정렬하는 등 필요한 조치를 취하십시오.

        작업을 생성한 후에는 수동으로 생성한 작업과 동일한 방식으로 사용자에게 작업을 할당하고 작업할 수 있습니다.
    3. 작업을 완료로 표시하려면 해당 작업의 확인란을 선택합니다.
    4. 작업의 세부 정보를 보거나 업데이트하려면 목록에서 해당 작업을 선택합니다.
    5. 작업을 재정렬하려면 작업을 선택하고 확인란 왼쪽으로 드래그하십시오.
    6. 작업을 제거하려면 작업의 줄임표 메뉴를 선택하고 제거를 선택합니다.

이슈를 다른 이슈에 연결

이슈가 다른 이슈의 업무와 관련된 경우 해당 이슈를 연결할 수 있습니다. 이는 작업 항목 간의 종속성을 이해하고 추적하는 데 도움이 되는 빠른 방법입니다.

이슈를 연결할 때 선택할 수 있는 네 가지 상태는 다음과 같습니다.

- 관련 대상: 이 상태를 사용하면 연결된 이슈와 관련된 작업을 표시할 수 있습니다.
- 차단자: 이 상태를 사용하면 문제가 연결된 문제로 차단되었음을 표시할 수 있습니다.
- 차단 대상: 이 상태를 사용하면 문제가 연결된 문제의 진행을 차단하고 있음을 나타냅니다.
- 중복: 이 상태를 사용하면 문제가 중복되어 다른 이슈에서 캡처된 작업임을 나타냅니다.

링크를 만든 후 링크 상태를 변경할 수 있습니다. 링크와 해당 링크 상태는 링크를 생성한 이슈와 링크된 이슈 모두에서 링크된 이슈에 표시됩니다. 링크를 생성한 후 링크를 제거할 수도 있습니다.

이슈를 다른 이슈나 작업에 연결하려면

1. 다른 이슈에 연결하려는 이슈를 엽니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
2. 문제 연결을 선택합니다.
3. Mark as에서 링크의 상태를 선택합니다.

   이슈 #에 이슈의 번호를 입력하거나 드롭다운 목록에서 이슈를 선택합니다.
4. 다른 링크를 추가하려면 연결된 이슈 추가를 선택합니다.
5. 완료하면 업데이트를 선택하여 문제 및 연결된 관계에 연결된 모든 문제를 업데이트합니다.

문제를 차단됨 또는 차단되지 않음으로 표시

어떤 문제로 인해 문제를 해결할 수 없는 경우 해당 문제를 차단된 것으로 표시하는 것이 좋습니다. 예를 들어 아직 병합되지 않은 코드베이스의 다른 부분을 변경하여 문제가 발생한 경우 문제가 차단될 수 있습니다.

이슈를 차단됨으로 표시하면 이슈에 빨간색 차단됨 레이블이 CodeCatalyst 추가되어 백로그나 아카이브 또는 보드에서 눈에 잘 띄게 됩니다.

외부 상황이 해결되면 문제의 차단을 해제할 수 있습니다.

문제를 차단된 것으로 표시하려면

1. 차단된 것으로 표시하려는 문제를 엽니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
2. 작업을 선택한 다음 차단됨으로 표시를 선택합니다.

문제 차단을 해제하려면

1. 차단을 해제하려는 이슈를 엽니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
2. 작업을 선택한 다음 차단되지 않은 상태로 표시를 선택합니다.

댓글 추가, 편집 또는 삭제

이슈에 댓글을 남길 수 있습니다. 댓글에서 다른 스페이스 구성원, 스페이스의 다른 프로젝트, 관련 문제 및 코드에 태그를 지정할 수 있습니다.

이슈에 코멘트를 추가하려면

1. 프로젝트로 이동합니다.
2. 내비게이션 바에서 이슈를 선택합니다.
3. 댓글을 추가하려는 이슈를 선택합니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
4. 코멘트 필드에 코멘트를 입력합니다. 마크다운을 사용하여 서식을 추가할 수 있습니다.
5. 전송를 선택합니다.

설명을 편집하려면

이슈에 대해 작성한 코멘트를 편집할 수 있습니다. 본인이 작성한 댓글만 수정할 수 있습니다.

1. 프로젝트로 이동합니다.
2. 내비게이션 바에서 이슈를 선택합니다.
3. 댓글을 수정하려는 이슈를 선택합니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
4. 댓글을 수정하려면 수정하려는 댓글을 찾으세요.

> **Tip**
>
> 댓글을 가장 오래된 것부터 가장 최근에 나온 것부터 정렬할 수 있습니다. 댓글은 한 번에 10개씩 로드됩니다.

5. 줄임표 아이콘을 선택한 다음 편집을 선택합니다.
6. 댓글을 편집합니다. 마크다운을 사용하여 서식을 추가할 수 있습니다.
7. 저장(Save)을 선택합니다. 이제 댓글이 업데이트되었습니다.

설명을 삭제하려면

이슈에 대해 작성한 댓글을 삭제할 수 있습니다. 자신이 작성한 댓글만 삭제할 수 있습니다. 댓글이 삭제되면 사용자 이름이 표시되지만 원본 댓글 텍스트 대신 이 댓글이 삭제되었습니다. 라는 문구가 표시됩니다.

1. 프로젝트로 이동합니다.
2. 내비게이션 바에서 이슈를 선택합니다.
3. 댓글을 삭제하려는 이슈를 선택합니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
4. 줄임표 아이콘을 선택하고 삭제를 선택한 다음 확인을 선택합니다.

댓글에 멘션 사용

스페이스 구성원, 스페이스의 다른 프로젝트, 관련 문제 및 코멘트에 코드를 언급할 수 있습니다. 이렇게 하면 언급한 사용자 또는 리소스로 연결되는 빠른 링크가 만들어집니다.

댓글로 @mention 님께

1. 프로젝트로 이동합니다.
2. 내비게이션 바에서 이슈를 선택합니다.
3. 편집하려는 이슈를 선택하여 이슈 세부 정보를 확인하세요. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
4. 댓글 추가 입력란을 선택합니다.
5. 다른 사용자를 @*user_name* 멘션하려면 입력합니다.
6. 프로젝트를 @*project_name* 언급하려면 입력하세요.
7. 다른 문제를 @*issue_name* 입력하거나 @*issue_number* 언급하세요.
8. 소스 리포지토리의 특정 파일이나 코드를 @*file_name* 언급하려면 입력합니다.

> **Note**
>
> 입력하는 대로 사용자와 일치하는 용어가 포함된 상위 5개 항목 (사용자, 소스 리포지토리, 프로젝트 등) 의 목록이 @mention 채워집니다.

9. 언급하려는 원하는 항목을 선택합니다. 항목의 위치를 보여주는 경로가 댓글 입력란에 채워집니다.

10. 댓글을 완성하고 [Send] 를 선택합니다.

문제 찾기 및 보기

다음 섹션에서는 CodeCatalyst 프로젝트 내에서 이슈를 효과적으로 검색하고 보는 방법을 설명합니다.

이슈 검색

특정 매개변수를 검색하여 문제를 찾을 수 있습니다. 검색 세분화에 대한 자세한 내용은 을 참조하십시오에서 코드, 이슈, 프로젝트, 사용자 검색 CodeCatalyst.

문제를 검색하려면

1. 프로젝트로 이동합니다.
2. 검색 창을 사용하여 이슈나 이슈와 관련된 정보를 검색하세요. 쿼리 매개변수를 사용하여 검색을 세분화할 수 있습니다. 자세한 내용은 에서 코드, 이슈, 프로젝트, 사용자 검색 CodeCatalyst 단원을 참조하십시오.

정렬 문제

기본적으로 의 이슈는 수동 순서에 따라 CodeCatalyst 정렬됩니다. 수동 주문에는 사용자가 이동한 순서대로 문제가 표시됩니다. 수동 순서로 정렬한 이슈를 드래그 앤 드롭하여 순서를 변경할 수 있습니다. 이 정렬 옵션은 문제 백로그를 정리하고 문제의 우선 순위를 지정할 때 유용합니다.

다음 표는 그리드 보기와 보드 보기에서 이슈를 정렬하는 방법을 보여줍니다.

그리드 뷰 정렬 옵션	보드 뷰 정렬 옵션
수동 주문	수동 주문
최종 업데이트 날짜	최종 업데이트 날짜
우선 순위	우선 순위
견적	추정치
Title	Title

그리드 뷰 정렬 옵션	보드 뷰 정렬 옵션
ID	
상태 표시기	
차단됨	
사용자 지정 필드	

다음 절차를 사용하여 문제 정렬 방식을 변경하십시오.

이슈를 정렬하려면

1. 프로젝트로 이동합니다.
2. 탐색 창에서 이슈를 선택합니다. 기본 보기는 보드입니다.
3. (선택사항) 활성 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 다른 이슈 보기로 이동합니다.
4. 그리드 보기를 정렬하려면 다음 두 가지 옵션이 있습니다.
    a. 정렬 기준으로 사용할 필드의 헤더를 선택합니다. 헤더를 선택하면 오름차순과 내림차순이 번갈아 나타납니다.
    b. 정렬 기준 드롭다운 메뉴를 선택하고 정렬 기준으로 사용할 매개변수를 선택합니다. 이슈는 오름차순으로 정렬됩니다.
5. 보드 보기를 정렬하려면 정렬 기준 드롭다운 메뉴를 선택하고 정렬 기준으로 사용할 매개변수를 선택합니다. 이슈는 오름차순으로 정렬됩니다.

그룹화 문제

그룹화는 담당자, 레이블, 우선 순위와 같은 여러 매개 변수별로 게시판의 이슈를 정리하는 데 사용됩니다.

이슈를 그룹화하려면

1. 프로젝트로 이동합니다.
2. 탐색 창에서 이슈를 선택합니다. 기본 보기는 보드입니다.

3. (선택사항) 활성 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 다른 이슈 보기로 이동합니다.
4. 그룹을 선택합니다.
5. 그룹화 기준에서 그룹화 기준으로 사용할 매개변수를 선택합니다.
   - 담당자 또는 우선순위를 선택하는 경우 그룹 순서를 선택합니다.
   - 라벨을 선택하는 경우 라벨을 선택한 다음 그룹 주문을 선택합니다.
6. (선택 사항) 현재 문제가 할당되지 않은 그룹을 표시하거나 숨기려면 빈 그룹 표시 토글을 선택합니다.
7. 선택에 따라 보기가 업데이트됩니다. 구성된 매개변수와 일치하는 그룹에만 문제가 나타납니다.

필터링 문제

필터링을 사용하여 지정된 이름, 우선순위, 레이블, 사용자 지정 필드 또는 담당자가 포함된 문제를 찾을 수 있습니다.

문제를 필터링하려면

1. 프로젝트로 이동합니다.
2. 탐색 창에서 이슈를 선택합니다.
3. (선택 사항) 진행 중인 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 다른 이슈 보기로 이동합니다.

> **Note**
>
> 이슈 이름 또는 설명의 문자열을 기준으로 필터링하려면 이슈 검색창에 문자열을 입력합니다.

4. 필터를 선택한 다음 + 필터 추가를 선택합니다.
5. 필터링할 매개변수를 선택합니다. 여러 필터 및 매개변수를 선택할 수 있습니다. 및/또는 를 선택하여 모든 필터 또는 개별 필터와 일치하는 이슈를 표시하도록 필터를 구성할 수 있습니다. 뷰가 업데이트되어 필터와 일치하는 문제가 표시됩니다.

이슈 진행

모든 이슈에는 라이프사이클이 있습니다. CodeCatalyst에서는 이슈가 일반적으로 백로그의 초안으로 작성됩니다. 해당 문제에 대한 작업이 시작되면 다른 상태 범주로 이동하고 완료될 때까지 다양한 상태를 거쳐 보관됩니다. 다음과 같은 방법으로 수명 주기 전반에 걸쳐 이슈를 이동하거나 진행할 수 있습니다.

- 백로그와 게시판 간에 이슈를 이동할 수 있습니다.
- 진행 중인 이슈는 다양한 완료 단계를 거쳐 이동할 수 있습니다.
- 완료된 이슈를 보관할 수 있습니다.

백로그와 게시판 간 이슈 이동

이슈에 대한 작업을 시작하면 백로그에서 보드로 이슈를 옮길 수 있습니다. 작업이 연기된 경우 이슈를 백로그로 다시 옮길 수도 있습니다.

백로그와 게시판 간에 이슈를 옮기려면

1. 프로젝트로 이동합니다.
2. 탐색 창에서 이슈를 선택합니다. 기본 보기는 보드입니다.
3. 게시판에서 백로그로 이슈를 옮기는 방법:
    a. 옮기려는 이슈를 선택합니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
    b. 드롭다운 상태 메뉴에서 백로그를 선택합니다.
4. 백로그에서 보드로 이슈를 옮기는 방법:
    a. 백로그로 이동하려면 게시판을 선택하고 백로그를 선택합니다.
    b. 옮기려는 이슈를 선택합니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
    c. 보드에 추가를 선택하거나 백로그가 아닌 상태를 선택합니다.

보드의 라이프사이클 단계를 통해 문제를 진행하세요.

완료될 때까지 보드 내에서 이슈를 다른 상태로 이동할 수 있습니다.

게시판 내에서 이슈를 옮기려면

1. 탐색 창에서 이슈를 선택합니다. 기본 보기는 보드입니다.
2. 다음 중 하나를 수행합니다.
   - 이슈를 다른 상태로 드래그 앤 드롭합니다.
   - 이슈를 선택한 다음 상태 드롭다운 메뉴에서 상태를 선택합니다.
   - 문제를 선택한 다음 다음으로 이동을 선택합니다.***next-status***.

문제 보관에 대한 자세한 내용은 을 참조하십시오이슈 보관.

그룹 간 이동 문제

모든 이슈 및 보드 보기에서 다양한 매개변수별로 이슈를 그룹화할 수 있습니다. 이슈를 그룹화하면 한 그룹에서 다른 그룹으로 이슈를 이동할 수 있습니다. 한 그룹에서 다른 그룹으로 이슈를 이동하면 이슈가 그룹화된 필드가 대상 그룹과 일치하도록 자동으로 편집됩니다.

예를 들어, 사용 중인 회사 중 왕 시울란과 CodeCatalyst 사안비 사카르라는 두 사람에게 이슈가 배정되어 있다고 가정해 보겠습니다. 이사회는 다음과 같이 그룹화되어 있으며 `Assignee` 담당자당 하나씩 총 두 개의 그룹이 있습니다. Wang Xiulan 그룹에서 Saanvi Sarkar 그룹으로 이슈를 옮기면 해당 이슈의 담당자가 Saanvi Sarkar로 업데이트됩니다.

이슈 보관

> **Note**
>
> 이슈는 프로젝트 내에서 삭제되지 않고 보관됩니다. 이슈를 삭제하려면 프로젝트를 삭제해야 합니다.

프로젝트에 더 이상 필요하지 않은 이슈를 보관할 수 있습니다. 이슈를 보관하면 보관된 이슈를 필터링하는 모든 뷰에서 이슈를 CodeCatalyst 제거합니다. 보관된 이슈는 보관된 이슈 기본 보기에서 볼 수 있으며, 필요한 경우 보관을 취소할 수 있습니다.

다음과 같은 경우 이슈를 보관합니다.

- 이슈를 완료했으므로 완료 열에 해당 이슈가 더 이상 필요하지 않습니다.
- 작업할 계획이 없습니다.

- 실수로 만들었네요.
- 활성 이슈 수가 최대치에 도달했습니다.

이슈를 보관하려면

1. 보관하려는 이슈를 엽니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오문제 찾기 및 보기.
2. 작업을 선택한 다음 보관함으로 이동을 선택합니다.
3. (선택 사항) 완료 상태인 여러 이슈를 빠르게 보관하려면 보드의 모든 완료 상태 상단에 있는 세로 줄임표를 선택하고 이슈 보관을 선택합니다.

이슈 보관을 취소하려면

1. 보관을 취소하려는 이슈를 엽니다. 이슈 보기 전환기 드롭다운 메뉴에서 보관된 이슈 보기를 열어 보관된 이슈 목록을 볼 수 있습니다. 문제를 찾는 데 도움이 필요하면 을 참조하십시오. 문제 찾기 및 보기
2. 보관 취소를 선택합니다.

수출 문제

현재 뷰의 이슈를.xlsx 파일로 내보낼 수 있습니다. 이슈를 내보내려면 다음 단계를 수행하세요.

이슈를 내보내려면

1. 프로젝트로 이동합니다.
2. 내비게이션 바에서 이슈를 선택합니다.
3. 활성 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 내보내려는 이슈가 포함된 뷰로 이동합니다. 뷰에 표시된 이슈만 내보내집니다.
4. 줄임표 메뉴를 선택하고 Excel로 내보내기를 선택합니다.
5. .xlsx 파일이 다운로드됩니다. 기본적으로 프로젝트 이름과 내보내기가 완료된 날짜가 제목으로 지정됩니다.

백로그, 라벨, 보드로 작업 정리하기

모든 팀이 같은 방식으로 업무를 수행하는 것은 아닙니다. 문제가 표시되는 방식과 Amazon에서 할당할 수 있는 방식을 CodeCatalyst 구성하여 작업 중인 항목과 해당 작업의 상태를 정확하게 파악할 수

있습니다. 사용자가 모두 동일한 추정 방법을 사용하도록 문제에 사용할 수 있는 추정 방법을 선택할 수 있습니다. 작업 보기를 필터링하는 데 사용할 수 있는 사용자 지정 레이블 및 상태를 만들 수 있습니다. 팀의 작업 방식에 따라 한 이슈에 대해 여러 담당자를 허용할지, 아니면 단일 사용자에게만 이슈를 할당하도록 허용할지 구성할 수 있습니다. 또한 이슈에 대한 사용자 지정 보기를 생성하여 본인 또는 팀과 가장 관련성이 높은 정보를 표시하는 방식으로 업무가 표시되도록 할 수 있습니다.

라벨로 작업 분류하기

이슈에 맞게 라벨을 사용자 지정할 수 있습니다. 여기에는 레이블 편집 및 색상 변경이 포함됩니다. 레이블은 작업을 분류하고 구성하는 데 도움이 될 수 있습니다. 예를 들어, 소프트웨어의 특정 부분이나 다양한 그룹 또는 팀을 위한 레이블을 만들 수 있습니다.

주제

라벨 생성

CodeCatalyst에서는 새 이슈를 생성하거나 기존 이슈를 편집할 때 라벨을 추가하여 라벨을 생성합니다. 자세한 내용은 에서 이슈 생성 CodeCatalyst 및 에서 이슈에 대한 편집 및 공동 작업 CodeCatalyst 단원을 참조하세요.

라벨 편집

다음 절차를 사용하여 기존 레이블의 이름 또는 색상을 변경합니다.

레이블 편집하기

1. 탐색 창에서 이슈를 선택합니다.
2. 활성 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 설정을 선택합니다.
3. 레이블 타일에는 프로젝트에 사용된 레이블 목록이 있습니다. 편집하려는 레이블 옆의 편집 아이콘을 선택합니다. 다음 중 한 개 이상을 수행할 수 있습니다.
   a. 레이블 이름을 편집합니다.
   b. 색상을 변경하려면 색상표를 선택합니다. 피커를 사용하여 새 색상을 선택합니다.
4. 라벨 변경 내용을 저장하려면 체크 표시 아이콘을 선택합니다.

5. 이제 사용 가능한 레이블 목록에 변경된 레이블이 표시됩니다. 또한 해당 레이블을 사용하는 문제의 수를 확인할 수 있습니다.

> **Note**
>
> 각 라벨 옆에 표시된 번호를 선택하여 모든 이슈 페이지로 이동하여 해당 라벨이 포함된 모든 이슈를 확인할 수 있습니다.

라벨 삭제

현재 에서는 이슈 라벨을 삭제할 수 없습니다 CodeCatalyst. 모든 이슈에서 라벨을 제거하면 해당 라벨이 이슈 설정의 라벨 섹션에 있는 미사용 라벨 섹션에 표시됩니다. 필터를 사용하거나 이슈에 라벨을 추가할 때 라벨 목록 끝에 사용하지 않은 라벨이 표시됩니다. 이슈 설정에서 모든 라벨 (사용 및 미사용) 과 해당 라벨이 있는 이슈의 개요를 확인할 수 있습니다.

사용자 지정 필드로 작업 구성하기

사용자 지정 필드를 생성하여 프로젝트의 작업을 정리하고 확인할 수 있습니다. 사용자 지정 필드는 필터의 사용 가능한 필터 목록에 추가되므로 사용자 지정 필드별로 이슈를 필터링할 수 있습니다. 사용자 지정 필드는 이름과 값 쌍입니다. 사용자 지정 필드의 이름을 기준으로 필터링한 다음 해당 사용자 지정 필드의 값을 기준으로 필터링합니다.

이슈에는 사용자 지정 필드가 여러 개 있을 수 있습니다.

사용자 지정 필드 만들기

CodeCatalyst에서는 이슈를 생성하거나 기존 이슈를 편집할 때 사용자 정의 필드를 추가하여 사용자 지정 필드를 생성합니다. 자세한 내용은 에서 이슈 생성 CodeCatalyst 및 에서 이슈에 대한 편집 및 공동 작업 CodeCatalyst 단원을 참조하세요.

사용자 지정 필드 삭제

사용자 지정 필드를 삭제하려면 추가된 각 이슈에서 사용자 지정 필드를 제거해야 합니다. 사용자 지정 필드를 삭제하면 필터에 사용자 지정 필드가 더 이상 표시되지 않습니다. 필터를 사용하여 사용자 지정 필드와 관련된 모든 문제를 확인하고 문제를 편집하여 제거할 수 있습니다. 자세한 내용은 문제 찾기 및 보기 및 이슈 편집 섹션을 참조하세요.

사용자 지정 상태로 작업 추적

보드에 사용자 지정 상태를 추가할 수 있습니다. 각 사용자 지정 상태는 초안, 시작되지 않음, 시작됨 또는 완료됨 범주 중 하나에 속해야 합니다. 상태 범주는 상태를 구성하고 기본 보기를 채우는 데 도움이 됩니다. 상태 및 상태 카테고리에 대한 자세한 내용은 을 참조하십시오상태 및 상태 카테고리. 뷰에 대한 자세한 내용은 을 참조하십시오. 문제 찾기 및 보기

상태를 만들려면

1. 탐색 창에서 Issues를 선택합니다.
2. 활성 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 설정을 선택합니다.
3. 상태에서 상태를 표시하려는 카테고리 옆에 있는 더하기 아이콘을 선택합니다.
4. 상태 이름을 지정한 다음 체크 표시 아이콘을 선택합니다.

> **Note**
>
> X 아이콘을 선택하여 상태 추가를 취소합니다.

이제 사용자 지정 상태가 보드에 표시되며 이슈 생성 시 옵션으로 표시됩니다.

상태를 편집하려면

1. 탐색 창에서 이슈를 선택합니다.
2. 활성 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 설정을 선택합니다.
3. 상태에서 편집하거나 변경하려는 상태 옆의 수정 아이콘을 선택합니다.
4. 상태를 편집한 다음 체크 표시 아이콘을 선택합니다.

이제 편집된 상태가 보드에 표시됩니다.

상태를 이동하려면

1. 탐색 창에서 Issues를 선택합니다.
2. 줄임표 아이콘을 선택하고 설정을 선택합니다.
3. 상태에서 이동하려는 상태를 선택합니다.
4. 상태를 원하는 위치로 드래그 앤 드롭합니다.

> **Note**
>
> 상태는 지정된 범주 내에서만 이동할 수 있습니다.

이제 보드의 상태가 재정렬됩니다.

상태를 비활성화하려면

1. 탐색 창에서 Issues를 선택합니다.
2. 활성 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 설정을 선택합니다.
3. 상태에서 비활성화하려는 상태를 선택합니다.
4. 비활성화하려는 상태에서 상태의 토글을 선택합니다. 이제 상태가 회색으로 표시됩니다.

> **Note**
>
> 비활성화된 상태는 모든 이슈가 삭제될 때까지 보드에 표시됩니다. 비활성화된 상태에는 이슈를 추가할 수 없습니다.

5. 비활성화된 상태를 다시 활성화하려면 상태의 토글을 선택합니다. 상태는 더 이상 회색으로 표시되지 않습니다.

> **Note**
>
> 각 카테고리에는 하나 이상의 활성 상태가 있어야 합니다. 카테고리에 상태가 하나뿐인 경우 해당 상태를 비활성화할 수 없습니다.

이슈 작업량 추정 구성

다음 단계에 따라 에서 문제에 대한 작업량 추정 설정을 구성하십시오. CodeCatalyst

문제에 대한 작업량 추정을 구성하려면

1. 탐색 창에서 Issues를 선택합니다.
2. 활성 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 설정을 선택합니다.

3. 기본 설정 섹션의 추정치에서 추정 값이 표시되는 방식을 선택합니다. 사용 가능한 추정치 유형으로는 티셔츠 사이즈, 피보나치 시퀀싱, 추정치 숨기기 등이 있습니다. 추정 유형이 업데이트되면 데이터가 손실되지 않으며 모든 문제의 추정 값이 자동으로 변환됩니다. 전환 매핑은 다음 표에 나와 있습니다.

티셔츠 사이즈	피보나치 수열
XS	1
XS	2
S	3
M	5
L	8
특대	13

여러 담당자 활성화 또는 비활성화

다음 단계에 따라 문제의 여러 담당자에 대한 설정을 구성하십시오. CodeCatalyst

여러 담당자를 활성화 또는 비활성화하려면

1. 탐색 창에서 이슈를 선택합니다.
2. 활성 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 설정을 선택합니다.
3. 기본 설정 섹션의 담당자에서 지표를 전환하여 동일한 이슈에 여러 담당자를 배정할 수 있도록 하세요. 이슈에는 최대 10명의 담당자가 있을 수 있습니다. 이 옵션을 활성화하지 않으면 한 명의 담당자만 이슈에 배정할 수 있습니다.

이슈 뷰 생성

뷰를 생성하여 특정 필터 세트와 일치하는 이슈를 빠르게 볼 수 있습니다. 이렇게 하면 시간을 절약하고 이전에 필터링, 그룹화 또는 정렬한 문제를 빠르게 볼 수 있습니다.

이슈 뷰를 만들려면

1. 탐색 창에서 이슈를 선택합니다.
2. (선택 사항) 사용 사례에 따라 기존 뷰에서 뷰를 만들 수도 있습니다. 다른 보기로 이동하려면 활성 이슈를 선택하여 이슈 보기 전환기 드롭다운 메뉴를 열고 보기를 선택합니다.
3. (선택 사항) 보기를 만들기 전에 필터, 그룹화, 정렬을 구성하세요. 뷰를 만들 때 추가할 수 있지만 이전에 추가했다면 뷰를 만들기 전에 뷰에 표시된 내용을 미리 볼 수 있습니다.
4. 헤더 바에서 이슈 뷰 전환기 드롭다운 메뉴를 엽니다. 상태를 기반으로 한 열로 이슈를 볼 수 있는 보드 보기를 만들려면 보드 열에서 +를 선택하세요. 이슈를 목록으로 볼 수 있는 그리드 뷰를 만들려면 그리드 열에서 +를 선택하세요. 생각이 바뀌면 보기를 만들기 전에 보기 유형을 변경할 수 있습니다.
5. 뷰 만들기 대화 상자에서 뷰의 이름을 입력합니다.
6. 필터, 이슈 그룹화 기준, 이슈 정렬 기준 필드는 현재 뷰의 설정에 따라 채워집니다. 필요한 경우 업데이트하십시오.
7. 보기 만들기를 선택하여 보기를 만들고 해당 보기로 전환합니다.

내 이슈에 대한 할당량 CodeCatalyst

다음 표에는 Amazon 문제에 대한 할당량 및 한도가 설명되어 있습니다. CodeCatalyst Amazon의 할당량에 대한 자세한 내용은 CodeCatalyst 다음을 참조하십시오. 에 대한 할당량 CodeCatalyst

Resource	기본 할당량
진행 중인 이슈	프로젝트당 최대 1,000개
첨부 파일 크기	첨부 파일당 최대 500MB입니다. 최대 첨부 파일 저장 용량은 전체 용량 한도에 영향을 받습니다. 자세한 내용은 요금을 참조하세요.
총 발행물 수 (활성 및 보관된 문제)	프로젝트당 최대 100,000개
저장된 조회수	프로젝트당 저장된 이슈 뷰는 최대 50개입니다.
이슈에 연결할 수 있는 풀 리퀘스트 수	이슈당 최대 50개의 풀 리퀘스트.

Resource	기본 할당량
상태 (프로젝트당)	프로젝트당 최대 50개
상태 (호당)	발행당 최대 50개.
라벨 (프로젝트당)	프로젝트당 최대 200개
라벨 (호당)	호당 최대 50개.
사용자 지정 필드 (문제당)	문제당 최대 50개
양수인	문제당 최대 10명
설명	호당 최대 1,000개까지 가능합니다.
Tasks	호당 최대 100개까지 가능합니다.

에서 ID, 권한 및 액세스 구성 CodeCatalyst

CodeCatalyst Amazon에 처음 로그인할 때 AWS 빌더 ID를 생성합니다. AWS 에는 빌더가 존재하지 IDs 않습니다 AWS Identity and Access Management. 처음 로그인할 때 선택한 사용자 이름은 ID를 위한 고유한 사용자 ID가 됩니다.

로그인할 CodeCatalyst 때는 다음 두 가지 방법 중 하나로 처음 로그인할 수 있습니다.

- 스페이스 만들기의 일환으로
- 에 있는 프로젝트 또는 스페이스에 CodeCatalyst 대한 초대를 수락하는 과정의 일환으로

ID와 관련된 역할 또는 역할에 따라 수행할 수 있는 작업이 결정됩니다 CodeCatalyst. 프로젝트 관리자 및 기여자와 같은 프로젝트 역할은 프로젝트별로 다르므로 한 프로젝트에서는 한 역할을, 다른 프로젝트에서는 다른 역할을 가질 수 있습니다. 스페이스를 만들면 스페이스 관리자 역할이 CodeCatalyst 자동으로 할당됩니다. 사용자가 프로젝트 초대를 수락하면 해당 ID를 CodeCatalyst 스페이스에 추가하고 사용자에게 제한된 액세스 역할을 할당합니다. 사용자를 프로젝트에 초대할 때는 프로젝트에서 사용자에게 부여할 역할을 선택합니다. 이 역할에 따라 프로젝트 내에서 수행할 수 있는 작업과 수행할 수 없는 작업이 결정됩니다. 프로젝트에서 작업하는 대부분의 사용자는 작업을 수행하는 데 기여자 역할만 있으면 됩니다. 자세한 내용은 사용자 역할을 통한 액세스 권한 부여 단원을 참조하십시오.

프로젝트 사용자는 프로젝트 역할 외에도 Git 클라이언트 또는 통합 개발 환경 (PAT) 을 사용할 때 프로젝트의 소스 리포지토리에 액세스하기 위한 개인 액세스 토큰 () 이 필요합니다. IDEs 프로젝트 구성원은 이를 타사 애플리케이션에서 PAT 자신의 ID와 관련된 애플리케이션별 비밀번호로 사용할 수 있습니다. CodeCatalyst 예를 들어 소스 리포지토리를 로컬 컴퓨터에 복제할 때는 CodeCatalyst 사용자 이름과 함께 PAT 이름을 입력해야 합니다.

워크플로에 작업을 배포할 때 서비스 역할을 사용하여 AWS CloudFormation 스택 CodeCatalyst 및 AWS 리소스에 액세스하는 등의 작업을 수행하여 리소스 간 액세스를 구성할 수 있습니다. 실행할 프로젝트 템플릿에 포함된 워크플로 작업에 대해 AWS 리소스 간 CodeCatalyst 액세스를 구성해야 합니다.

주제

- 사용자 역할을 통한 액세스 권한 부여
- 개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여
- 개인적인 연결을 통해 GitHub 리소스에 접근하기
- 다단계 인증으로 로그인하도록 AWS 빌더 ID 구성 () MFA

- 아마존의 보안 CodeCatalyst
- 로깅을 사용하여 이벤트 및 API 통화 모니터링
- ID, 권한 및 액세스에 대한 할당량 CodeCatalyst
- 문제 해결

사용자 역할을 통한 액세스 권한 부여

CodeCatalystAmazon에서는 프로젝트 수준과 스페이스 수준 모두에서 사용자에게 역할을 할당할 수 있습니다. 프로젝트에서 역할은 사용자가 해당 프로젝트의 리소스로 프로젝트에서 수행할 수 있는 작업을 지정합니다. 사용자는 프로젝트에 참여할 때 스페이스의 멤버십을 얻습니다. 사용자를 스페이스 관리자로 추가 또는 제거할 수 있습니다. 스페이스 관리자 역할은 모든 역할 중에서 가장 광범위한 권한을 가집니다. CodeCatalyst 가장 좋은 방법은 작업을 수행하는 데 필요한 가장 좁은 권한을 사용자에게 할당하는 것입니다.

스페이스의 사용자에게 역할을 할당할 수 있습니다. 사용자가 멤버인 프로젝트의 사용자에게 역할을 할당할 수도 있습니다. 각 사용자는 프로젝트 또는 스페이스에서 하나의 역할만 가질 수 있지만 사용자는 각 프로젝트 및 스페이스에서 서로 다른 역할을 가질 수 있습니다. 예를 들어 사용자는 한 프로젝트에서는 프로젝트 관리자 역할을, 다른 프로젝트에서는 기여자 역할을 가질 수 있습니다.

주제

- 공간 및 프로젝트의 사용자 역할 이해
- 각 역할에 사용할 수 있는 권한 보기
- 사용자 역할 보기 및 변경

공간 및 프로젝트의 사용자 역할 이해

스페이스에 사용할 수 있는 역할은 세 가지입니다.

- 스페이스 관리자
- 파워 유저
- 제한된 액세스

프로젝트 초대를 수락한 사용자에게는 프로젝트가 포함된 스페이스에서 제한된 액세스 역할이 자동으로 할당됩니다.

프로젝트에서 멤버가 사용할 수 있는 역할은 네 가지입니다.

- 프로젝트 관리자
- 기고자
- 리뷰어
- 읽기 전용

프로젝트에 사용자를 추가하면 사용자에게 제한된 액세스 역할이 CodeCatalyst 자동으로 부여됩니다. 모든 프로젝트에서 사용자를 제거하면 해당 사용자의 제한된 액세스 역할이 CodeCatalyst 자동으로 제거됩니다.

스페이스 관리자 역할

스페이스 관리자 역할은 에서 가장 강력한 CodeCatalyst 역할입니다. 스페이스 관리자 역할은 모든 권한을 포함하므로 스페이스의 모든 측면을 관리해야 하는 사용자에게만 스페이스 관리자 역할을 할당하십시오. CodeCatalyst 스페이스 관리자 역할을 가진 사용자는 스페이스 관리자 역할에서 다른 사용자를 추가하거나 제거하고 스페이스를 삭제할 수 있는 유일한 사용자입니다.

스페이스를 만들면 스페이스 관리자 역할이 CodeCatalyst 자동으로 할당됩니다. 가장 좋은 방법은 원래 스페이스 생성자를 사용할 수 없는 경우에 대비하여 이 역할을 수행할 수 있는 다른 사용자 한 명 이상에게 이 역할을 추가하는 것입니다.

파워 유저 역할

파워 사용자 역할은 CodeCatalyst 스페이스에서 두 번째로 강력한 역할이지만 스페이스의 프로젝트에 대한 액세스 권한은 없습니다. 스페이스에서 프로젝트를 생성하고 스페이스의 사용자 및 리소스를 관리하는 데 도움이 되어야 하는 사용자를 위해 설계되었습니다. 프로젝트를 만들고 업무의 일환으로 스페이스의 사용자를 관리할 수 있어야 하는 팀 리더 또는 관리자인 사용자에게 고급 사용자 역할을 할당하십시오.

제한된 액세스 역할

제한된 액세스 액세스 역할은 CodeCatalyst 스페이스에서 대부분의 사용자가 맡게 되는 역할입니다. 사용자가 스페이스의 프로젝트 초대를 수락할 때 사용자에게 자동으로 할당되는 역할입니다. 프로젝트가 포함된 공간 내에서 작업하는 데 필요한 제한된 권한을 제공합니다. 스페이스에 직접 초대한 사용자에게 제한된 액세스 역할을 할당하십시오. 단, 업무상 스페이스의 특정 부분을 관리해야 하는 경우는 예외입니다.

프로젝트 관리자 역할

프로젝트 관리자 역할은 프로젝트에서 가장 강력한 역할입니다. CodeCatalyst 프로젝트 설정 편집, 프로젝트 권한 관리, 프로젝트 삭제 등 프로젝트의 모든 측면을 관리해야 하는 사용자에게만 이 역할을 할당하세요.

프로젝트 역할에는 스페이스 수준에서 어떠한 권한도 없습니다. 따라서 프로젝트 관리자 역할을 가진 사용자는 추가 프로젝트를 만들 수 없습니다. 스페이스 관리자 또는 고급 사용자 역할을 가진 사용자만 프로젝트를 만들 수 있습니다.

> **Note**
>
> 스페이스 관리자 역할에는 모든 권한이 CodeCatalyst 있습니다.

기여자 역할

기여자 역할은 프로젝트에 참여하는 대다수 구성원을 대상으로 합니다. CodeCatalyst 프로젝트의 코드, 워크플로, 이슈 및 작업을 수행할 수 있어야 하는 사용자에게 이 역할을 할당하세요.

리뷰어 역할

리뷰어 역할은 풀 리퀘스트 및 이슈와 같은 프로젝트 내 리소스와 상호 작용할 수 있어야 하지만 코드를 생성 및 병합하거나 워크플로를 생성하거나 프로젝트에서 워크플로 실행을 시작 또는 중지할 수는 없어야 하는 사용자를 대상으로 합니다. CodeCatalyst 풀 리퀘스트를 승인하고 코멘트를 작성하고, 이슈를 생성, 업데이트, 해결 및 코멘트하고, 프로젝트의 코드와 워크플로를 볼 수 있어야 하는 사용자에게 리뷰어 역할을 할당하세요.

읽기 전용 역할

읽기 전용 역할은 리소스 및 리소스의 상태를 볼 필요가 있지만 리소스와 상호 작용하거나 프로젝트에 직접 기여하지 않는 사용자를 위한 것입니다. 이 역할을 가진 사용자는 에서 CodeCatalyst 리소스를 만들 수 없지만 리포지토리를 복제하고 이슈의 첨부 파일을 로컬 컴퓨터에 다운로드하는 등 리소스를 보고 복사할 수는 있습니다. 리소스와 프로젝트 상태를 확인해야 하지만 프로젝트와 직접 상호 작용하지는 않는 사용자에게 읽기 전용 역할을 할당하십시오.

각 역할에 사용할 수 있는 권한 보기

다음 표에는 각 CodeCatalyst 역할에 사용할 수 있는 권한이 나와 있습니다. 링크를 사용하여 적절한 권한 세트로 이동할 수 있습니다.

- Space permissions
- Extensions permissions
- Project permissions
- Source repository permissions
- Dev Environment permissions
- Package repository and package permissions
- Workflow permissions
- Issues permissions
- Blueprint permissions
- Notifications permissions
- Search permissions

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
스페이스 권한							
공간 만들기	✓	✗	✗	✗	✗	✗	✗
스페이스 청구 세부 정보 편집	✓	✗	✗	✗	✗	✗	✗
싱글 사인온 설정 및 활성화	✓	✗	✗	✗	✗	✗	✗
싱글 사인온 삭제	✓	✗	✗	✗	✗	✗	✗
공간에 제너레이티	✓	✗	✗	✗	✗	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
브 AI 기능 활성화							
공간의 제너레이티브 AI 기능 비활성화	✓	✗	✗	✗	✗	✗	✗
스페이스 삭제	✓	✗	✗	✗	✗	✗	✗
스페이스 관리자 역할에 다른 사용자 추가	✓	✗	✗	✗	✗	✗	✗
스페이스 관리자 역할에서 다른 사용자 제거	✓	✗	✗	✗	✗	✗	✗
팀 생성	✓	✗	✗	✗	✗	✗	✗
팀 삭제	✓	✗	✗	✗	✗	✗	✗
팀 업데이트	✓	✗	✗	✗	✗	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
스페이스의 컴퓨터 리소스를 비활성화합니다.	✓	✗	✗	✗	✗	✗	✗
스페이스의 컴퓨터 리소스를 활성화합니다.	✓	✗	✗	✗	✗	✗	✗
프로젝트 생성	✓	✓	✗	✗	✗	✗	✗
AWS계정 연결을 스페이스에 연결	✓	✓	✗	✗	✗	✗	✗
AWS계정 연결 업데이트	✓	✓	✗	✗	✗	✗	✗
스페이스에서 AWS 계정 연결 끊기	✓	✓	✗	✗	✗	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
AWS계정 연결을 삭제하고 스페이스에서 제거합니다.	✓	✓	✗	✗	✗	✗	✗
스페이스에서 프로젝트 제한 계정 연결 활성화[1]	✓	✓	✗	✗	✗	✗	✗
스페이스에서 프로젝트 제한 계정 연결을 비활성화하세요[2]	✓	✓	✗	✗	✗	✗	✗
스페이스에 다른 사람 초대하기	✓	✓	✗	✗	✗	✗	✗
VPC연결 만들기	✓	✓	✗	✗	✗	✗	✗
VPC연결 편집	✓	✓	✗	✗	✗	✗	✗
VPC연결 삭제	✓	✓	✗	✗	✗	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
스페이스 내 활동 로그 보기	✓	✓	✗	✗	✗	✗	✗
AWS계정 연결 보기	✓	✓	✓	✓	✓	✓	✓
에 대한 인시던트 보기 CodeCatalyst	✓	✓	✓	✓	✓	✓	✓
스페이스 보기	✓	✓	✓	✓	✓	✓	✓
팀 보기	✓	✓	✓	✓	✓	✓	✓
VPC연결 보기	✓	✓	✓	✓	✓	✓	✓

[1] 고급 사용자 역할을 사용하면 계정에 대한 프로젝트 제한을 활성화할 수 있지만, 구성원인 프로젝트에 대한 액세스 권한만 구성할 수 있습니다.

[2] 고급 사용자 역할을 사용하면 계정에 대한 프로젝트 제한을 비활성화할 수 있지만 구성원인 프로젝트에 대해서만 액세스를 구성할 수 있습니다.

익스텐션 권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
확장 프로그램 설치	✓	✓	✗	✗	✗	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
확장 업데이트	✓	✓	✗	✗	✗	✗	✗
확장 프로그램 삭제	✓	✓	✗	✗	✗	✗	✗
GitHub 계정 연결	✓	✓	✗	✗	✗	✗	✗
계정 연결 끊기 GitHub	✓	✓	✗	✗	✗	✗	✗
Jira 사이트 연결	✓	✓	✗	✗	✗	✗	✗
Jira 사이트 연결 끊기	✓	✓	✗	✗	✗	✗	✗
설치된 확장 프로그램의 구성 세부 정보 보기	✓	✓	✗	✗	✗	✗	✗
확장 보기	✓	✓	✓	✓	✓	✓	✓

프로젝트 권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
프로젝트 설정 편집	✓	✗	✗	✓	✗	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
프로젝트의 컴퓨터 리소스를 비활성화합니다.							
프로젝트의 컴퓨터 리소스를 활성화합니다.							
프로젝트 삭제							
프로젝트에 사용자 초대							
프로젝트 내 사용자 역할 변경							
프로젝트에서 사용자 제거							
프로젝트에 팀 추가							
프로젝트에서 팀 삭제							

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
팀의 프로젝트 역할 변경							
프로젝트 보기							
프로젝트 활동 보기							
프로젝트 내 팀 보기							
청사진 보기							

소스 리포지토리 권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
리포지토리 생성							
리포지토리 연결							
리포지토리 연결 해제							
리포지토리 삭제							

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
리포지토리 설정 편집	✓	✗	✗	✓	✓	✗	✗
리포지토리 보기	✓	✗	✗	✓	✓	✓	✓
리포지토리 설정 보기	✓	✗	✗	✓	✓	✓	✓
클론 리포지토리	✓	✗	✗	✓	✓	✓	✓
브랜치 생성	✓	✗	✗	✓	✓	✗	✗
브랜치 규칙 생성	✓	✗	✗	✓	✗	✗	✗
기본 브랜치 변경	✓	✗	✗	✓	✗	✗	✗
브랜치 삭제	✓	✗	✗	✓	✓	✗	✗
브랜치 병합	✓	✗	✗	✓	✓	✗	✗
브랜치 규칙 업데이트	✓	✗	✗	✓	✗	✗	✗
브랜치 보기	✓	✗	✗	✓	✓	✓	✓

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
브랜치 규칙 보기							
폴더 생성							
폴더 삭제							
폴더 수정							
폴더 보기							
파일 생성							
파일 삭제							
파일 편집							
파일 보기							
커밋 생성 및 푸시							
커밋 보기							
풀 리퀘스트 생성							

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
풀 리퀘스트에 대한 승인 규칙 생성	✓	✗	✗	✓	✗	✗	✗
풀 리퀘스트의 병합 요구 사항 재정의	✓	✗	✗	✓	✗	✗	✗
풀 리퀘스트 업데이트	✓	✗	✗	✓	✓	✗	✗
풀 리퀘스트의 승인 규칙 업데이트	✓	✗	✗	✓	✗	✗	✗
풀 요청 보기	✓	✗	✗	✓	✓	✓	✓
풀 리퀘스트의 승인 규칙 보기	✓	✗	✗	✓	✓	✓	✓
풀 리퀘스트 닫기	✓	✗	✗	✓	✓	✗	✗
풀 리퀘스트 승인	✓	✗	✗	✓	✓	✓	✗
풀 리퀘스트에 댓글 달기	✓	✗	✗	✓	✓	✓	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
풀 리퀘스트에 대한 코멘트에서 Amazon Q와 상호 작용하세요	✓	✗	✗	✓	✓	✗	✗
Amazon Q에서 생성한 풀 리퀘스트에 대한 수정 버전 생성	✓	✗	✗	✓	✓	✗	✗
이슈를 풀 리퀘스트에 연결	✓	✗	✗	✓	✓	✗	✗
풀 리퀘스트에서 이슈 연결 해제	✓	✗	✗	✓	✓	✗	✗
개발 환경 권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
나만의 개발 환경 만들기	✓	✗	✗	✓	✓	✓	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
자체 개발 환경을 중단하세요							
다른 사용자가 만든 개발 환경을 중지하세요							
자체 개발 환경을 다시 시작하세요.							
자체 개발 환경 보기							
다른 사용자가 만든 개발 환경 보기							
자체 개발 환경 편집							
다른 사용자가 만든 개발 환경 편집							
자체 개발 환경 삭제							

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
다른 사용자가 만든 개발 환경 삭제	✓	✓	✗	✗	✗	✗	✗
개발 환경을 위한 개발 파일 만들기	✓	✗	✗	✓	✓	✗	✗
개발 환경을 위한 개발 파일 편집	✓	✗	✗	✓	✓	✗	✗
개발 환경용 개발 파일 삭제	✓	✗	✗	✓	✓	✗	✗
개발 환경용 개발 파일 보기	✓	✗	✗	✓	✓	✓	✓
패키지 리포지토리 및 패키지 권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
패키지 리포지토리 생성	✓	✗	✗	✓	✗	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
패키지 리포지토리 보기							
패키지 리포지토리 편집							
패키지 리포지토리 삭제							
게이트웨이 패키지 리포지토리 생성							
게이트웨이 패키지 리포지토리 보기							
게이트웨이 패키지 리포지토리 삭제							
업스트림 패키지 리포지토리 추가							

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
업스트림 리포지토리의 검색 순서 편집	✓	✗	✗	✓	✗	✗	✗
업스트림 패키지 리포지토리 제거	✓	✗	✗	✓	✗	✗	✗
패키지 리포지토리에 연결	✓	✗	✗	✓	✓	✓	✓
패키지 리포지토리에서 패키지 읽기	✓	✗	✗	✓	✓	✓	✓
패키지 저장소에 패키지 게시	✓	✗	✗	✓	✓	✗	✗
업스트림 리포지토리에서 패키지 읽기 및 보관	✓	✗	✗	✓	✓	✓	✓
패키지 보기	✓	✗	✗	✓	✓	✓	✓
패키지 버전 보기	✓	✗	✗	✓	✓	✓	✓

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
패키지 버전 자산 보기	✓	✗	✗	✓	✓	✓	✓
패키지 버전 종속성 목록	✓	✗	✗	✓	✓	✓	✓
패키지 버전 상태 업데이트	✓	✗	✗	✓	✗	✗	✗
패키지 오리진 구성 업데이트	✓	✗	✗	✓	✗	✗	✗
패키지 버전 삭제	✓	✗	✗	✓	✗	✗	✗
워크플로 권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
워크플로우 생성	✓	✗	✗	✓	✓	✗	✗
워크플로 업데이트	✓	✗	✗	✓	✓	✗	✗
워크플로우 삭제	✓	✗	✗	✓	✓	✗	✗
워크플로우 시작	✓	✗	✗	✓	✓	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
워크플로우 중지							
워크플로 시크릿 생성							
워크플로 비밀 업데이트							
워크플로 비밀 삭제							
환경 생성							
환경 삭제							
플릿 생성							
플릿 업데이트							
플릿 삭제							
다른 계정의 워크플로 리소스 관리							

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
AWS 계정 연결을 환경에 연결							
기본 IAM 역할을 환경에 연결							
VPC연결을 환경과 연결							
환경과의 VPC 연결 끊기							
VPC연결된 환경을 워크플로와 연결							
VPC연결된 환경을 워크플로우와 분리하십시오.							
워크플로 실행 승인							
워크플로우에서 커밋 추적							

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
환경 보기	✓	✗	✗	✓	✓	✓	✓
빌드 작업 로그 보기	✓	✗	✗	✓	✓	✓	✓
플릿 보기	✓	✗	✗	✓	✓	✓	✓
테스트 작업 로그 보기	✓	✗	✗	✓	✓	✓	✓
워크플로우 보기	✓	✗	✗	✓	✓	✓	✓
워크플로우 실행 보기	✓	✗	✗	✓	✓	✓	✓
워크플로 실행 결과 보기	✓	✗	✗	✓	✓	✓	✓
워크플로 시크릿 보기	✓	✗	✗	✓	✓	✓	✓
권한 발급	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
이슈 생성	✓	✗	✗	✓	✓	✓	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
업데이트 문제	✓	✗	✗	✓	✓	✓	✗
이슈 보기	✓	✗	✗	✓	✓	✓	✓
작업 생성	✓	✗	✗	✓	✓	✓	✗
업데이트 작업	✓	✗	✗	✓	✓	✓	✗
작업 보기	✓	✗	✗	✓	✓	✓	✓
이슈 아카이브	✓	✗	✗	✓	✓	✓	✗
Amazon Q에 이슈 할당하기	✓	✗	✗	✓	✓	✗	✗
문제에 대한 의견을 통해 Amazon Q와 상호 작용하십시오.	✓	✗	✗	✓	✓	✗	✗
이슈에서 Amazon Q 할당 취소	✓	✗	✗	✓	✓	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
Amazon Q 관련 문제에 대한 추천 작업	✓	✗	✗	✓	✓	✗	✗
Amazon Q에서 추천하는 작업 생성	✓	✗	✗	✓	✓	✗	✗
다른 사용자가 생성한 업데이트 문제	✓	✗	✗	✓	✓	✓	✗
이슈에 대한 의견 보기	✓	✗	✗	✓	✓	✓	✓
이슈에 댓글 작성	✓	✗	✗	✓	✓	✗	✗
문제에 대한 의견 업데이트	✓	✗	✗	✓	✓	✓	✗
라벨 생성	✓	✗	✗	✓	✓	✗	✗
라벨 업데이트	✓	✗	✗	✓	✓	✗	✗
라벨 보기	✓	✗	✗	✓	✓	✓	✓

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
이슈에 라벨 추가	✓	✗	✗	✓	✓	✓	✗
이슈에서 라벨 삭제	✓	✗	✗	✓	✓	✓	✗
이슈에 대한 사용자 지정 상태 만들기	✓	✗	✗	✓	✓	✗	✗
사용자 지정 상태 업데이트	✓	✗	✗	✓	✓	✗	✗
사용자 지정 상태 보기	✓	✗	✗	✓	✓	✓	✓
사용자 지정 상태 이동	✓	✗	✗	✓	✓	✓	✗
사용자 지정 상태 비활성화	✓	✗	✗	✓	✓	✓	✗
이슈에 첨부 파일 추가	✓	✗	✗	✓	✓	✓	✗
이슈 첨부 파일 보기	✓	✗	✗	✓	✓	✓	✓

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
이슈에서 첨부 파일 제거	✓	✗	✗	✓	✓	✓	✗
이슈를 다른 이슈에 연결	✓	✗	✗	✓	✓	✓	✗
이슈를 다른 이슈와 연결 해제하기	✓	✗	✗	✓	✓	✗	✗
이슈 링크 업데이트	✓	✗	✗	✓	✓	✓	✗
이슈에 대한 링크 보기	✓	✗	✗	✓	✓	✓	✓
풀 리퀘스트를 이슈에 연결	✓	✗	✗	✓	✓	✗	✗
풀 리퀘스트와 이슈의 연결 해제	✓	✗	✗	✓	✓	✗	✗
Jira 프로젝트 연결	✓	✗	✗	✓	✗	✗	✗
Jira 프로젝트 연결 해제	✓	✗	✗	✓	✗	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
블루프린트 권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
커스텀 블루프린트 프로젝트 생성	✓	✓	✗	✗	✗	✗	✗
프리뷰 커스텀 블루프린트 퍼블리시	✓	✓	✗	✗	✗	✗	✗
커스텀 블루프린트 퍼블리시	✓	✓	✗	✗	✗	✗	✗
우주 청사진 카탈로그에 사용자 지정 청사진 추가	✓	✓	✗	✗	✗	✗	✗
우주 청사진 카탈로그에서 사용자 지정 청사진 제거	✓	✓	✗	✗	✗	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
사용자 지정 청사진의 게시 권한 관리	✓	✓	✗	✗	✗	✗	✗
커스텀 블루프린트의 카탈로그 버전 관리	✓	✓	✗	✗	✗	✗	✗
커스텀 블루프린트 업데이트	✓	✓	✗	✗	✗	✗	✗
커스텀 블루프린트 버전 삭제	✓	✓	✗	✗	✗	✗	✗
커스텀 블루프린트 삭제	✓	✓	✗	✗	✗	✗	✗
소스 리포지토리를 커스텀 블루프린트로 변환	✓	✓	✗	✓	✗	✗	✗
프로젝트에 커스텀 블루프린트 추가	✓	✓	✗	✓	✗	✗	✗

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
프로젝트에서 커스텀 블루프린트를 분리하세요.							
적용된 커스텀 블루프린트의 버전 업데이트							
커스텀 블루프린트의 설정 편집							
게시된 커스텀 블루프린트 보기							

알림 권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
알림 채널 구성							
알림 채널 제거							
알림 설정 편집							

권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
알림 설정 보기							
CodeCatalyst 사고에 대한 알림을 자동으로 수신합니다.							
연결된 이메일 계정에 대한 이메일 알림을 구성하세요.							

검색 권한	스페이스 관리자 역할	파워 유저 역할	제한된 액세스 역할	프로젝트 관리자 역할	컨트리뷰터 역할	리뷰어 역할	읽기 전용 역할
프로젝트 내부 검색							
스페이스 전체 검색							

사용자 역할 보기 및 변경

사용자에게 할당된 역할을 볼 수 있습니다. 이렇게 하면 프로젝트에서 사용자가 취할 수 있는 작업을 이해하는 데 도움이 됩니다. 추가 권한이 필요한 경우 역할을 변경할 수도 있습니다.

프로젝트에서 사용자의 역할을 보려면

1. 각 프로젝트 멤버와 관련된 역할을 보려는 프로젝트로 이동합니다.

> **Tip**
> 상단 내비게이션 바에서 보려는 프로젝트를 선택할 수 있습니다.

2. 탐색 창에서 프로젝트 설정을 선택합니다.
3. 멤버 탭에서 각 프로젝트 멤버의 역할이 역할에 표시됩니다.

프로젝트에서 사용자 역할 변경하기

1. 프로젝트 멤버와 관련된 역할을 변경하려는 프로젝트로 이동합니다.

> **Tip**
> 상단 내비게이션 바에서 보려는 프로젝트를 선택할 수 있습니다.

2. 탐색 창에서 프로젝트 설정을 선택합니다.
3. 멤버 탭의 프로젝트 멤버에서 역할을 변경하려는 사용자를 선택합니다. 작업을 선택한 다음 역할 편집을 선택합니다.
4. 역할에서 프로젝트 역할을 선택한 다음 확인을 선택합니다.

스페이스 내 역할 보기 및 변경

에서 프로젝트 초대를 수락한 모든 사용자는 프로젝트 스페이스의 CodeCatalyst 구성원이 됩니다. 스페이스 구성원 목록을 볼 수 있습니다. 사용자 역할을 제한된 액세스에서 스페이스 관리자로 변경하여 스페이스와 리소스를 더 잘 관리할 수 있습니다. 스페이스 관리자 역할은 사용자가 프로젝트를 만들 수 있는 유일한 역할입니다. CodeCatalyst

> **Warning**
> 스페이스 관리자 역할은 에서 가장 강력한 CodeCatalyst 역할입니다. 이 역할을 가진 사용자는 스페이스 삭제를 CodeCatalyst 비롯한 모든 작업을 수행할 수 있습니다. 공간에 대해 이 수준의 액세스 권한이 필요한 사용자에게만 이 역할을 할당하세요. 자세한 내용은 스페이스 관리자 역할 단원을 참조하십시오.

스페이스에서 사용자 역할을 변경하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 스페이스로 이동합니다.

> **Tip**
>
> 둘 이상의 스페이스에 속해 있는 경우 상단 탐색 표시줄에서 보려는 스페이스를 선택할 수 있습니다.

3. 구성원 탭을 선택합니다.
4. 역할을 변경하려는 사용자를 선택한 다음 역할 변경을 선택합니다.
5. 역할 변경에서 할당하려는 역할을 선택한 다음 확인을 선택합니다.

개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여

Git 클라이언트 또는 통합 개발 환경 (IDE) 이 설치된 로컬 컴퓨터에서 소스 리포지토리와 같은 일부 CodeCatalyst 리소스에 액세스하려면 애플리케이션별 비밀번호를 입력해야 합니다. 이 용도로 사용할 개인용 액세스 토큰 (PAT) 을 만들 수 있습니다. PATs생성한 항목은 의 모든 스페이스 및 프로젝트에서 사용자 ID와 연결됩니다 CodeCatalyst. PAT CodeCatalystID에 대해 두 개 이상을 만들 수 있습니다.

생성한 이름과 만료 날짜를 볼 수 있으며 더 이상 필요하지 않은 이름은 삭제할 수 있습니다. PATs 암호를 만들 당시에만 PAT 암호를 복사할 수 있습니다.

> **Note**
>
> 기본적으로 1년 PATs 후에 만료됩니다.

생성 중 PATs

PATs의 사용자 ID와 연결되어 있습니다 CodeCatalyst. 암호를 만들 당시에만 PAT 암호를 복사할 수 있습니다.

생성 중 PATs (콘솔)

콘솔을 사용하여 PATs 에서 만들 수 CodeCatalyst 있습니다.

개인용 액세스 토큰을 만들려면 (콘솔)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 상단 메뉴 바에서 프로필 배지를 선택한 다음 내 설정을 선택합니다. CodeCatalyst내 설정 페이지가 열립니다.

> **Tip**
>
> 프로젝트 또는 스페이스의 구성원 페이지로 이동한 다음 구성원 목록에서 이름을 선택하여 사용자 프로필을 찾을 수도 있습니다.

3. 개인용 액세스 토큰에서 생성을 선택합니다.

   만들기 PAT 페이지가 표시됩니다.
4. PAT이름에 이름을 설명하는 이름을 입력합니다. PAT
5. 만료 날짜에 기본 날짜를 유지하거나 달력 아이콘을 선택하여 사용자 지정 날짜를 선택합니다. 만료 날짜는 기본적으로 현재 날짜로부터 1년 후입니다.
6. 생성(Create)을 선택합니다.

> **Tip**
>
> 소스 리포지토리의 복제 리포지토리를 선택할 때도 이 토큰을 생성할 수 있습니다.

7. PAT암호를 복사하려면 [복사] 를 선택합니다. 검색할 수 있는 위치에 PAT 암호를 저장하십시오.

> **Important**
>
> PAT비밀은 한 번만 표시됩니다. 창을 닫은 후에는 검색할 수 없습니다. 안전한 위치에 PAT 암호를 저장하지 않은 경우 다른 암호를 만들 수 있습니다.

.

생성 중 PATs (CLI)

를 CLI 사용하여 PATs 에서 만들 수 CodeCatalyst 있습니다.

개인용 액세스 토큰을 만들려면 (AWS CLI)

1. 터미널 또는 명령줄에서 다음과 같이 create-access-token 명령을 실행합니다.

```
aws codecatalyst create-access-token
```

성공하면 명령은 다음 예제와 PAT 같이 생성된 항목에 대한 정보를 반환합니다.

```
{
    "secret": "value",
    "name": "marymajor-22222EXAMPLE",
    "expiresTime": "2024-02-04T01:56:04.402000+00:00"
}
```

2.

암호를 만들 때 한 번만 PAT 암호를 볼 수 있습니다. PAT PAT암호를 분실했거나 안전하게 저장되지 않을까 걱정되는 경우 다른 암호를 만들 수 있습니다.

를 사용하여 사용자 PATs 계정과 관련된 정보를 볼 수 있습니다. AWS CLI에 대한 정보만 볼 수 있고 PAT 암호 자체의 가치에 대한 정보는 볼 수 없습니다. PAT

> **Note**
>
> 최신 버전을 AWS CLI 사용하여 작업하고 있는지 확인하십시오 CodeCatalyst. 이전 버전에는 CodeCatalyst 명령이 포함되어 있지 않을 수 있습니다. 에서 사용할 수 있으려면 먼저 AWS CLI 프로필을 구성해야 CodeCatalyst 합니다. 자세한 내용은 AWS CLI와 함께 사용하기 위한 설정 CodeCatalyst 단원을 참조하십시오.

보기 PATs

PATs에서 볼 수 CodeCatalyst 있습니다. 목록에는 사용자 ID와 관련된 모든 항목이 표시됩니다. PATs 내 모든 스페이스 및 프로젝트의 사용자 PAT 프로필은 내 사용자 프로필과 연결되어 CodeCatalyst 있습니다. 만료된 항목은 만료 후 삭제되므로 표시되지 PATs 않습니다.

보기 PATs (콘솔)

콘솔을 사용하여 에서 사용자 PATs ID와 관련된 내용을 볼 수 CodeCatalyst 있습니다.

개인용 액세스 토큰을 보려면 (콘솔)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 상단 메뉴 바에서 프로필 배지를 선택한 다음 내 설정을 선택합니다. CodeCatalyst 내 설정 페이지가 열립니다.

> **Tip**
>
> 프로젝트 또는 스페이스의 구성원 페이지로 이동한 다음 구성원 목록에서 이름을 선택하여 사용자 프로필을 찾을 수도 있습니다.

3. 개인용 액세스 토큰에서 현재 사용 중인 계정의 이름과 만료일을 확인하세요PATs.

보기 PATs (CLI)

에서 를 CLI 사용하여 사용자 PATs ID와 관련된 보기를 볼 수 CodeCatalyst 있습니다.

개인용 액세스 토큰을 보려면 (AWS CLI)

- 터미널 또는 명령줄에서 다음과 같이 list-access-tokens 명령을 실행합니다.

```
aws codecatalyst list-access-tokens
```

성공하면 명령은 다음 예와 같이 사용자 PATs 계정과 관련된 정보를 반환합니다.

```
{
    "items": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "name": "marymajor-22222EXAMPLE",
            "expiresTime": "2024-02-04T01:56:04.402000+00:00"
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "name": "marymajor-11111EXAMPLE",
            "expiresTime": "2023-03-12T01:58:40.694000+00:00"
        }
    ]
}
```

삭제 PATs

에서 사용자 PATs ID와 관련된 내용을 삭제할 수 CodeCatalyst 있습니다.

삭제 PATs (콘솔)

콘솔을 사용하여 삭제할 수 PATs 있습니다 CodeCatalyst.

개인용 액세스 토큰을 삭제하려면 (콘솔)

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 상단 메뉴 바에서 프로필 배지를 선택한 다음 내 설정을 선택합니다. CodeCatalyst 내 설정 페이지가 열립니다.

   > **Tip**
   >
   > 프로젝트 또는 스페이스의 구성원 페이지로 이동한 다음 구성원 목록에서 이름을 선택하여 사용자 프로필을 찾을 수도 있습니다.

3. 개인용 액세스 토큰에서 PAT 삭제하려는 액세스 토큰 옆의 선택기를 선택한 다음 삭제를 선택합니다.

   삭제 시PAT:<name>? 페이지에서 삭제를 확인하려면 텍스트 필드에 삭제를 입력합니다. Delete(삭제)를 선택합니다.

삭제 PATs (CLI)

를 사용하여 사용자 ID와 PAT 관련된 정보를 삭제할 수 AWS CLI있습니다. 이렇게 하려면 의 ID를 입력해야 하며PAT, delete-access-token 명령을 사용하여 해당 ID를 확인할 수 있습니다.

> **Note**
>
> AWS CLI 작업하려면 의 최신 버전을 사용하고 있는지 확인하십시오 CodeCatalyst. 이전 버전에는 CodeCatalyst 명령이 포함되어 있지 않을 수 있습니다. AWS CLI with 사용에 대한 자세한 내용은 CodeCatalyst 을 참조하십시오AWS CLI와 함께 사용하기 위한 설정 CodeCatalyst.

개인용 액세스 토큰을 삭제하려면 (AWS CLI)

- 터미널 또는 명령줄에서 PAT 삭제하려는 ID와 함께 delete-access-token 명령을 실행합니다. 예를 들어, 다음 명령을 실행하여 ID가 다음과 같은 PAT a를 삭제합니다.*123EXAMPLE*.

  ```
  aws codecatalyst delete-access-token --id a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb
  ```

  성공하면 이 명령은 응답을 반환하지 않습니다.

개인적인 연결을 통해 GitHub 리소스에 접근하기

개인 연결을 사용하여 타사 GitHub 리소스를 승인하고 연결할 수 있습니다. CodeCatalyst 예를 들어 개인 연결을 사용하여 GitHub 계정에 대한 CodeCatalyst 액세스를 승인하고 프로젝트 또는 청사진의 소스로 사용할 리포지토리를 만들 수 있습니다. 연결은 사용자 CodeCatalyst ID에 매핑되며 이를 사용하여 하나 이상의 소스 리포지토리에 연결할 수 있습니다. 생성한 연결은 내의 모든 스페이스 및 프로젝트에서 사용자 ID와 연결됩니다. CodeCatalyst

> **Note**
>
> 액세스 권한이 있는 GitHub 조직의 청사진을 사용하여 개인적인 관계를 관리할 수 있습니다.

모든 스페이스에서 공급자 유형별로 하나의 사용자 ID (CodeCatalyst 별칭) 에 대해 하나의 개인 연결을 생성할 수 있습니다.

에서 CodeCatalyst 개인 연결을 사용하여 프로젝트용 GitHub 리포지토리를 만들고, 블루프린트의 GitHub 소스 리포지토리를 선택하고, 리포지토리의 풀 리퀘스트를 관리할 수 GitHub 있습니다. CodeCatalyst

> **Note**
>
> 블루프린트와 리포지토리를 연결하는 데 개인 연결을 사용하는 것은 GitHub 리포지토리를 CodeCatalyst 연결할 때 확장을 사용하는 것과 다릅니다. GitHub 확장에 대한 자세한 내용은 을 참조하십시오. 확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst

개인 연결 만들기

콘솔을 사용하여 에서 사용자 ID와 관련된 개인용 연결을 만들 수 CodeCatalyst 있습니다.

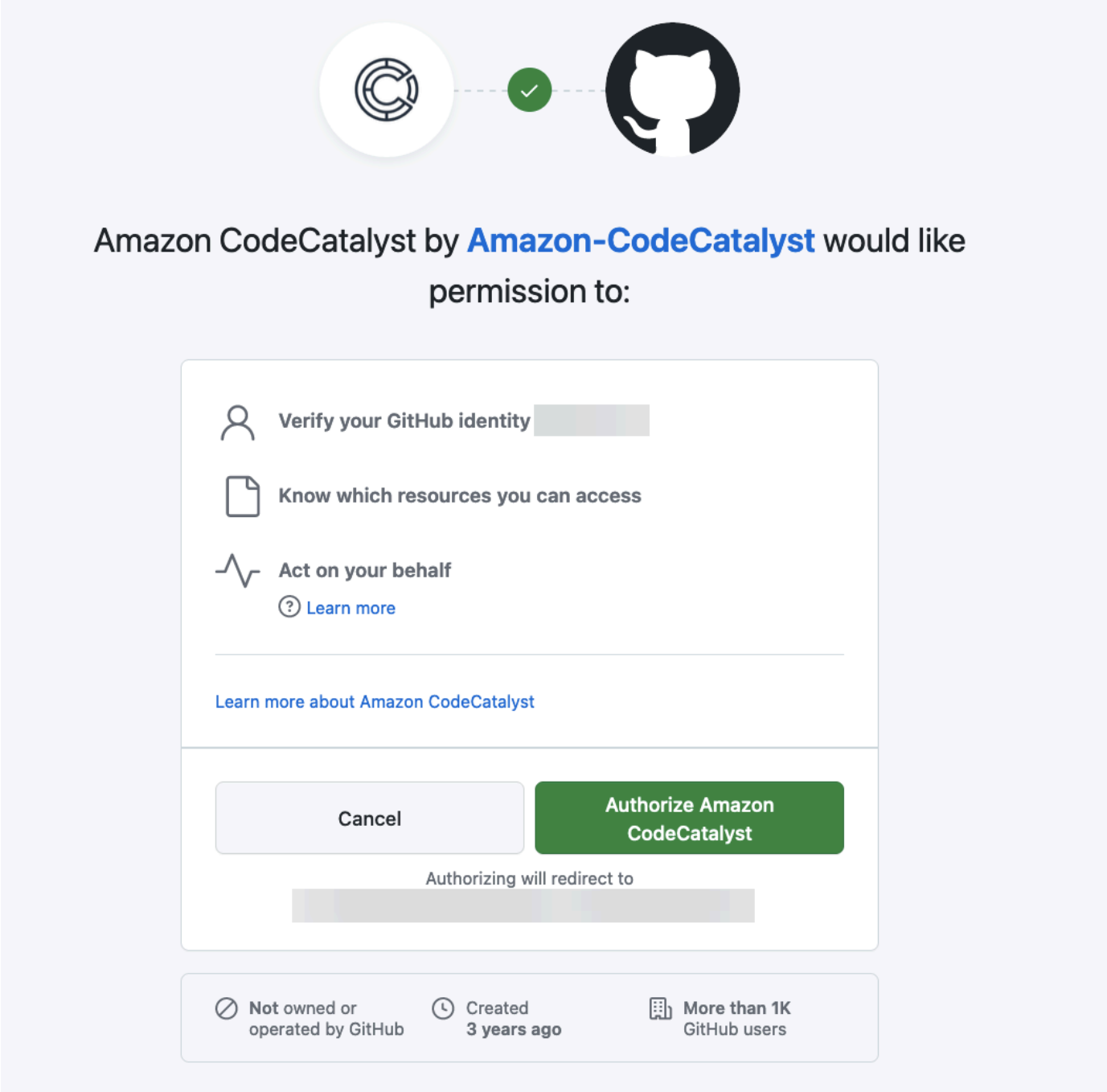
개인용 연결을 만들려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 상단 메뉴 바에서 프로필 배지를 선택한 다음 내 설정을 선택합니다. CodeCatalyst 내 설정 페이지가 열립니다.

   > **Tip**
   >
   > 프로젝트 또는 스페이스의 구성원 페이지로 이동한 다음 구성원 목록에서 이름을 선택하여 사용자 프로필을 찾을 수도 있습니다.

3. 개인 연결에서 만들기를 선택합니다.

   연결 만들기 페이지가 표시됩니다.
4. 생성(Create)을 선택합니다. 연결 만들기 페이지가 표시됩니다.
5. 연결 생성 페이지의 공급자에서 선택합니다 GitHub. 연결 이름에 연결 이름을 입력합니다. 생성(Create)을 선택합니다.
6. 메시지가 표시되면 GitHub 계정에 로그인합니다.
7. 연결 확인 페이지에서 수락을 선택합니다.
8. 설치 확인 페이지에서 인증 버튼을 선택하여 커넥터 응용 프로그램 설치를 확인합니다.



개인 연결 삭제

에서 사용자 ID와 관련된 개인용 연결을 삭제할 수 CodeCatalyst 있습니다.

> **Note**
>
> 에서 개인용 연결을 삭제해도 GitHub 계정에서 애플리케이션이 제거되지는 CodeCatalyst 않습니다. 새 개인용 연결을 생성하면 앱 설치를 사용할 수 있습니다. 에서 GitHub 애플리케이션을 제거하려면 애플리케이션을 해지하고 나중에 다시 설치하면 됩니다.

에서 개인 연결을 삭제하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 상단 메뉴 바에서 프로필 배지를 선택한 다음 내 설정을 선택합니다. CodeCatalyst 내 설정 페이지가 열립니다.

   > **Tip**
   >
   > 프로젝트 또는 스페이스의 구성원 페이지로 이동한 다음 구성원 목록에서 이름을 선택하여 사용자 프로필을 찾을 수도 있습니다.

3. 개인용 연결에서 삭제하려는 연결 옆의 선택기를 선택한 다음 삭제를 선택합니다.

   연결 삭제 시:<name>? 페이지에서 삭제를 확인하려면 텍스트 필드에 삭제를 입력합니다. Delete(삭제)를 선택합니다.

1. GitHub 로그인하고 설치된 앱의 계정 설정으로 이동합니다. 프로필 아이콘을 선택하고 설정을 선택한 다음 애플리케이션을 선택합니다.
2. 인증된 GitHub 앱 탭의 승인된 애플리케이션 목록에서 설치된 앱을 확인합니다 CodeCatalyst. 설치를 취소하려면 취소를 선택합니다.

다단계 인증으로 로그인하도록 AWS 빌더 ID 구성 () MFA

AWS 빌더 ID 프로필을 개인용으로 만들었든 업무용으로 만들었든 관계없이 멀티 팩터 인증 (MFA) 을 또 다른 보안 계층으로 구성하는 것이 좋습니다. 스페이스의 구성원인 MFA 경우 구성하여 다른 사용자와 프로젝트 공동 작업을 수행하는 것이 특히 좋습니다. 한 프로젝트에 두 명 이상이 액세스할 수 있기 때문에 보안 침해 가능성이 더 커집니다.

MFA활성화한 후에는 이메일과 비밀번호를 CodeCatalyst 사용하여 Amazon에 로그인해야 합니다. 로그인의 이 부분은 알고 있는 것을 사용하는 첫 번째 요소입니다. 그런 다음 코드나 보안 키를 사용하여

로그인합니다. 이것이 두 번째 요소인데, 여러분이 가지고 있는 것입니다. 두 번째 요소는 모바일 장치에서 생성하거나 컴퓨터에 연결된 보안 키를 탭하거나 눌러 생성되는 인증 코드일 수 있습니다. 이러한 여러 요소를 종합하면 무단 액세스를 방지하여 보안을 강화할 수 있습니다.

다단계 인증에 사용할 장치를 등록하는 방법

내 프로필 > 다단계 인증에서 다음 절차를 사용하여 다단계 인증을 위해 새 장치를 등록합니다 (). MFA

> **Note**
>
> 이 절차의 단계를 시작하기 전에 먼저 적절한 인증 앱을 장치에 다운로드하는 것이 좋습니다. MFA장치에 사용할 수 있는 앱 목록은 을 참조하십시오. 인증 애플리케이션

에서 사용할 장치를 등록하려면 MFA

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 오른쪽 상단에서 첫 번째 이니셜이 있는 아이콘 옆의 화살표를 선택한 다음 사용자 프로필을 선택합니다. CodeCatalyst프로필 페이지가 열립니다.
3. 프로필 페이지에서 프로필 및 보안 관리를 선택합니다. AWS 빌더 ID 프로필 페이지가 열립니다.
4. 페이지 왼쪽에서 보안을 선택합니다.
5. 다단계 인증 페이지에서 장치 등록을 선택합니다.
6. MFA장치 등록 페이지에서 다음 MFA 장치 유형 중 하나를 선택하고 지침을 따르십시오.

   - 보안 키 또는 내장된 인증자

     1. 사용자 보안 키 등록 페이지에서 브라우저 또는 플랫폼에 나와 있는 지침을 따릅니다.

        > **Note**
        >
        > 이 환경은 운영 체제 및 브라우저에 따라 다르므로 브라우저 또는 플랫폼에서 표시되는 지침을 따르십시오. 장치가 성공적으로 등록되고 나면 새로 등록된 장치에 친숙한 표시 이름을 연결할 수 있는 옵션이 제공됩니다. 이를 변경하려면 이름 바꾸기를 선택하고 새 이름을 입력한 다음 저장을 선택합니다.

   - 인증 앱

1. 인증 앱 설정 페이지에서 QR 코드 그래픽을 비롯한 새 MFA 장치의 구성 정보를 확인할 수 있습니다. 그래픽은 QR 코드를 지원하지 않는 디바이스 상에서 수동 입력할 수 있는 보안 구성 키를 표시한 것입니다.
2. 물리적 MFA 디바이스를 사용하여 다음을 수행하십시오.
   a. 호환되는 MFA 인증 앱을 엽니다. MFA디바이스에서 사용할 수 있는 테스트된 앱 목록은 을 참조하십시오. 테스트를 거친 인증 앱 MFA앱이 여러 기기를 지원하는 경우 옵션을 선택하여 새 MFA 기기를 생성하십시오.
   b. MFA앱이 QR 코드를 지원하는지 확인한 다음 인증 앱 설정 페이지에서 다음 중 하나를 수행하십시오.
      i. QR 코드 표시를 선택한 다음 해당 앱을 사용하여 QR 코드를 스캔합니다. 예를 들어, 카메라 모양의 아이콘을 선택하거나 코드 스캔와 비슷한 옵션을 선택합니다. 그런 다음 디바이스의 카메라를 사용하여 해당 코드를 스캔합니다.
      ii. '비밀 키 보기'를 선택한 다음 앱에 해당 비밀 키를 입력합니다. MFA

> **Important**
>
> AWS 빌더 ID를 사용하도록 MFA 기기를 구성할 때는 QR 코드 또는 비밀 키의 사본을 안전한 장소에 저장하세요. 이렇게 하면 휴대폰을 분실하거나 MFA 인증 앱을 다시 설치해야 하는 경우에 도움이 될 수 있습니다. 둘 중 하나라도 발생하는 경우 동일한 구성을 사용하도록 앱을 재구성할 수 있습니다. MFA

3. 인증 앱 설정 페이지의 인증 코드에서 현재 물리적 장치에 표시되는 일회용 비밀번호를 입력합니다. MFA

> **Important**
>
> 코드를 생성한 후 즉시 요청을 제출하세요. 코드를 생성한 다음 요청을 제출하는 데 너무 오래 기다리면 MFA 기기가 AWS 빌더 ID 프로필과 성공적으로 연결되지만 MFA 기기가 동기화되지 않습니다. 이는 시간 기반 일회용 암호 (TOTP) 가 잠시 후 만료되기 때문에 발생합니다. 이 경우, 디바이스를 재동기화할 수 있습니다.

4. 할당MFA을 선택합니다. 이제 MFA 디바이스에서 일회용 암호 생성을 시작할 수 있으며 이제 사용할 준비가 되었습니다.

인증 애플리케이션

인증 앱은 일회용 암호 (OTP) 기반 타사 인증 앱입니다. 사용자는 모바일 장치 또는 태블릿에 설치된 인증 애플리케이션을 인증된 장치로 사용할 수 있습니다. MFA 타사 인증 애플리케이션은 6자리 인증 코드를 생성할 수 있는 표준 기반 TOTP (시간 기반 일회용 암호) 알고리즘인 RFC 6238을 준수해야 합니다.

인증 앱을 입력하라는 메시지가 표시되면 사용자는 MFA 제공된 입력란에 인증 앱의 유효한 코드를 입력해야 합니다. 사용자에게 할당된 각 MFA 장치는 고유해야 합니다. 특정 사용자에 대해 두 개의 인증 앱을 등록할 수 있습니다.

테스트를 거친 인증 앱

모든 TOTP 호환 애플리케이션이 IAM Identity MFA Center와 호환되지만 다음 표에는 선택할 수 있는 잘 알려진 타사 인증 앱이 나열되어 있습니다.

운영 체제	테스트를 거친 인증 앱
Android	인증, 듀오 모바일, 인증기, 마이크로소프트 LastPass 인증기, 구글 인증기
iOS	인증, 듀오 모바일, 인증기, 마이크로소프트 LastPass 인증기, 구글 인증기

디바이스 변경 MFA

MFA장치를 등록한 후 이름을 변경하거나 삭제할 수 있습니다. 보안을 강화하기 위해 항상 하나 이상의 MFA 장치를 활성화해 두는 것이 좋습니다. 최대 5대의 장치를 등록할 수 있습니다. 추가 방법을 알아보려면 을 참조하십시오다단계 인증에 사용할 장치를 등록하는 방법.

장치 이름 변경 MFA

장치 이름을 변경하려면 MFA

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 오른쪽 상단에서 첫 번째 이니셜이 있는 아이콘 옆의 화살표를 선택한 다음 사용자 프로필을 선택합니다. CodeCatalyst프로필 페이지가 열립니다.
3. 프로필 페이지에서 프로필 및 보안 관리를 선택합니다. AWS 빌더 ID 프로필 페이지가 열립니다.

4. 페이지 왼쪽에서 다단계 인증을 선택합니다. 페이지에 도착하면 이름 바꾸기가 회색으로 표시된 것을 볼 수 있습니다.
5. 변경하려는 MFA 장치를 선택합니다. 이름 바꾸기를 선택합니다. 그러면 모달이 나타납니다.
6. 열리는 프롬프트에서 MFA장치 이름에 새 이름을 입력한 다음 이름 바꾸기를 선택합니다. 이름이 바뀐 장치는 다단계 인증 장치 () 아래에 표시됩니다. MFA

장치 삭제 MFA

MFA장치를 삭제하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 오른쪽 상단에서 첫 번째 이니셜이 있는 아이콘 옆의 화살표를 선택한 다음 사용자 프로필을 선택합니다. CodeCatalyst프로필 페이지가 열립니다.
3. 프로필 페이지에서 프로필 및 보안 관리를 선택합니다. AWS 빌더 ID 프로필 페이지가 열립니다.
4. 페이지 왼쪽에서 다단계 인증을 선택합니다. 페이지에 도착하면 삭제가 회색으로 표시된 것을 볼 수 있습니다.
5. 변경하려는 MFA 장치를 선택합니다. Delete(삭제)를 선택합니다. MFA장치 삭제? 라는 모달이 나타납니다. . 지침에 따라 기기를 삭제하세요.
6. Delete(삭제)를 선택합니다. 삭제된 장치는 더 이상 다단계 인증 장치 (MFA) 아래에 표시되지 않습니다.

아마존의 보안 CodeCatalyst

클라우드 AWS 보안이 최우선 과제입니다. AWS 고객은 가장 보안에 민감한 공간의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 두 사람 사이의 AWS 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다 AWS 클라우드. AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 프로그램별 범위 내AWS 서비스 규정 준수 참조하십시오. CodeCatalyst
- 클라우드에서의 보안 - 귀하의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 여러분은 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다

이 설명서는 Amazon을 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 CodeCatalyst 됩니다. 보안 및 규정 준수 목표를 CodeCatalyst 충족하도록 구성하는 방법을 보여줍니다. 또한 CodeCatalyst 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

내용

아마존에서의 데이터 보호 CodeCatalyst

이 모델에 설명된 대로 CodeCatalyst 은 (는) 서비스의 글로벌 인프라를 보호할 책임이 있습니다. 이 인프라에서 호스팅되는 콘텐츠에 대한 제어를 유지하는 것은 사용자의 책임입니다. 이 공동 책임 모델은 의 데이터 보호에 적용됩니다 CodeCatalyst.

데이터 보호를 위해 계정 자격 증명을 보호하고 로그인할 때 다단계 인증을 설정하는 것이 좋습니다. 자세한 정보는 다단계 인증으로 로그인하도록 AWS 빌더 ID 구성 () MFA을 참조하세요.

태그 또는 이름 필드와 같은 자유 형식 필드에 고객의 이메일 주소와 같은 기밀 또는 민감한 정보를 입력하지 마십시오. 여기에는 리소스 이름 및 연결된 식별자 외에 사용자가 입력하는 기타 모든 식별자가 포함됩니다. CodeCatalyst AWS 계정 예를 들어 기밀 또는 민감한 정보를 공간, 프로젝트 또는 배포 플릿 이름의 일부로 입력하지 마십시오. 이름에 사용되는 태그, 이름 또는 자유 형식 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용되거나 URL 경로에 포함될 수 있습니다. 이는 콘솔, APIAWS CLI, CodeCatalyst 액션 개발 키트 또는 모든 AWS SDK 사용에 적용됩니다.

외부 서버에 URL을 제공하는 경우 해당 서버에 대한 요청을 검증하기 위해 URL에 보안 자격 증명 정보를 포함하지 않는 것이 좋습니다.

CodeCatalyst 원본 리포지토리는 저장 시 자동으로 암호화됩니다. 고객 조치는 필요하지 않습니다. CodeCatalyst 또한 HTTPS 프로토콜을 사용하여 전송 중인 리포지토리 데이터를 암호화합니다.

CodeCatalyst MFA를 지원합니다. 자세한 정보는 다단계 인증으로 로그인하도록 AWS 빌더 ID 구성 () MFA을 참조하세요.

데이터 암호화

CodeCatalyst 서비스 내에서 데이터를 안전하게 저장하고 전송합니다. 모든 데이터는 전송 및 보관 중에 암호화됩니다. 서비스에 대한 모든 메타데이터를 포함하여 서비스에서 생성 또는 저장되는 모든 데이터는 기본적으로 서비스에 저장되고 암호화됩니다.

> **Note**
>
> 이슈에 대한 정보는 서비스 내에 안전하게 저장되지만 미해결 이슈에 대한 정보는 사용자가 이슈 보드, 백로그 및 개별 이슈를 본 브라우저의 로컬 캐시에 저장됩니다. 최적의 보안을 위해 브라우저 캐시를 삭제하여 이 정보를 제거해야 합니다.

에 CodeCatalyst 연결된 AWS 계정 저장소나 연결된 저장소에 대한 계정 연결 등 연결된 리소스를 사용하는 경우 연결된 리소스에서 CodeCatalyst 전송되는 데이터는 암호화되지만 연결된 리소스의 데이터 처리는 연결된 서비스에서 관리합니다. GitHub 자세한 내용은 연결된 서비스 및 설명서를 참조하십시오Amazon에서의 워크플로 작업에 대한 모범 사례 CodeCatalyst.

키 관리

CodeCatalyst 키 관리를 지원하지 않습니다.

인터네트워크 트래픽 개인 정보 보호

에서 CodeCatalyst 스페이스를 생성할 때 해당 공간에 대한 데이터와 리소스를 저장할 AWS 리전 위치를 선택합니다. 프로젝트 데이터와 메타데이터는 절대 그 자리를 벗어나지 않습니다AWS 리전. 하지만 내부 CodeCatalyst 탐색을 지원하기 위해 제한된 공간, 프로젝트 및 사용자 메타데이터가 파티션 전체에 AWS 리전 복제됩니다. 해당 파티션 AWS 리전 외부에는 복제되지 않습니다. 예를 들어 공간을 만들 AWS 리전 때 미국 서부 (오레곤) 를 선택하면 데이터가 중국 지역 또는 지역에 복제되지 않습니다. AWS GovCloud (US) 자세한 내용은 관리 AWS 리전, AWS글로벌 인프라 및 AWS서비스 엔드포인트를 참조하십시오.

파티션 AWS 리전 내부에 복제되는 데이터에는 다음이 포함됩니다.

- 공간 이름의 고유성을 보장하기 위해 공간 이름을 나타내는 암호화된 해시 값입니다. 이 값은 사람이 읽을 수 없으며 공백의 실제 이름을 노출하지 않습니다.

- 스페이스의 고유 ID
- 스페이스 간 탐색을 지원하는 스페이스의 메타데이터
- AWS 리전스페이스가 위치한 위치
- 스페이스에 있는 모든 프로젝트의 고유 ID
- 스페이스 또는 프로젝트에서 사용자의 역할을 나타내는 역할 ID
- 가입 시 다음을 CodeCatalyst 포함한 등록 프로세스에 대한 데이터 및 메타데이터
  - 의 고유 ID AWS Builder ID
  - 해당 사용자의 표시 이름 AWS Builder ID
  - 자신의 계정에 있는 사용자의 별칭 AWS Builder ID
  - 사용자가 가입할 때 사용한 이메일 주소 AWS Builder ID
  - 가입 프로세스 진행 상황
  - 가입 프로세스의 일환으로 스페이스를 생성하는 경우, 스페이스의 결제 계정으로 사용되는 AWS 계정 ID

스페이스 이름은 전체적으로 CodeCatalyst 고유합니다. 스페이스 이름에 민감한 데이터를 포함하지 않도록 주의하십시오.

연결된 리소스 및 연결된 계정 (예: GitHub 저장소 연결) 으로 작업할 때는 소스 및 대상 위치를 각 위치에서 지원하는 최고 수준의 보안으로 구성하는 것이 좋습니다. AWS 계정 CodeCatalyst 전송 계층 보안 (TLS) 1.2를 사용하여 AWS 계정AWS 리전, 및 가용 영역 간의 연결을 보호합니다.

Identity 및 Access Management와 Amazon CodeCatalyst

CodeCatalystAmazon에서는 스페이스와 프로젝트에 로그인하고 액세스하기 위해 AWS Builder ID를 생성하고 사용합니다. AWS 빌더 ID는 IAM AWS Identity and Access Management (ID) 이 아니며 IAM 내에 존재하지 않습니다. AWS 계정 CodeCatalyst 하지만 요금 청구를 위해 공간을 확인하고 해당 공간에 리소스를 생성하고 사용하기 위해 연결되면 IAM과 통합됩니다. AWS 계정 AWS 계정

AWS Identity and Access Management (IAM) 은 관리자가 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와줍니다. AWS IAM 관리자는 누가 리소스를 사용하도록 인증되고(로그인 됨) 권한이 부여되는지(권한 있음)를 제어합니다. IAM은 추가 AWS 서비스 비용 없이 사용할 수 있습니다.

CodeCatalystAmazon에서 스페이스를 생성할 때는 를 공간 결제 계정으로 연결해야 합니다. AWS 계정 CodeCatalyst 공간을 AWS 계정 확인하려면 에서 관리자 권한이 있거나 권한이 있어야 합니다. 연

결된 AWS 계정공간에서 리소스를 생성하고 액세스하는 데 사용할 CodeCatalyst 수 있는 IAM 역할을 추가할 수도 있습니다. 이를 서비스 역할이라고 합니다. 둘 AWS 계정 이상에 대한 연결을 생성하고 해당 계정 CodeCatalyst 각각에 대한 서비스 역할을 생성할 수 있습니다.

> **Note**
>
> 청구는 AWS 계정 지정된 청구 계정에서 CodeCatalyst 이루어집니다. 하지만 해당 CodeCatalyst 서비스 역할 AWS 계정 또는 기타 연결된 AWS 계정서비스 역할을 생성하면 해당 CodeCatalyst 서비스 역할로 생성되고 사용된 리소스에 연결된 AWS 계정리소스에 요금이 청구됩니다. 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 청구 관리를 참조하십시오.

주제

IAM의 자격 증명 기반 정책

자격 증명 기반 정책은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 해당 ID는 사용자, 사용자 그룹 또는 역할일 수 있습니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업와 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 자격 증명 기반 정책에서는 보안 주체가 연결된 사용자 또는

역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

CodeCatalyst 자격 증명 기반 정책 예시

CodeCatalyst ID 기반 정책의 예를 보려면 을 참조하십시오. 연결에 대한 ID 기반 정책 예제 CodeCatalyst

IAM에서의 정책 조치

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 주체가 어떤 리소스, 어떤 조건에서 어떤 작업을 수행할 수 있는지입니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 정책 조치는 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "prefix:action1",
      "prefix:action2"
         ]
```

IAM의 정책 리소스

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 주체가 어떤 리소스, 어떤 조건에서 어떤 작업을 수행할 수 있는지입니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 타입을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

IAM의 정책 조건 키

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 주체가 어떤 리소스, 어떤 조건에서 어떤 작업을 수행할 수 있는지입니다.

Condition 요소(또는 Condition 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. Condition 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 Condition요소를 지정하거나 단일 Condition요소에서 여러 키를 지정하는 경우 AWS 는 논리적 AND태스크를 사용하여 평가합니다. 단일 조건 키의 여러 값을 지정하는 경우 AWS 는 논리적 OR태스크를 사용하여 조건을 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

연결에 대한 ID 기반 정책 예제 CodeCatalyst

에서 CodeCatalyst, AWS 계정 공간 청구를 관리하고 프로젝트 워크플로의 리소스에 액세스하려면 필요합니다. 계정 연결은 공간 추가를 AWS 계정 승인하는 데 사용됩니다. ID 기반 정책은 커넥티드 환경에서 사용됩니다. AWS 계정

기본적으로 사용자와 역할에는 리소스를 만들거나 수정할 권한이 없습니다. CodeCatalyst 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. IAM 관리자는 리소스에서 작업을 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 사용자에 이러한 정책을 연결해야 합니다.

다음 예제 IAM 정책은 계정 연결과 관련된 작업에 대한 권한을 부여합니다. 이를 사용하여 계정 연결에 대한 액세스를 제한할 수 CodeCatalyst 있습니다.

예 1: 사용자가 한 번에 연결 요청을 수락하도록 허용 AWS 리전

다음 권한 정책은 사용자가 CodeCatalyst 및 사이의 연결 요청을 보고 수락하는 것만 허용합니다 AWS 계정. 또한 정책에서는 us-west-2 지역에서의 작업만 허용하고 다른 지역의 작업은 허용하지 않는 조건을 사용합니다. AWS 리전요청을 보고 승인하려면 사용자는 요청에 지정된 AWS Management Console 계정과 동일한 계정으로 로그인합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:AcceptConnection",
        "codecatalyst:GetPendingConnection"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-west-2"
        }
      }
    }
  ]
}
```

예 2: 콘솔에서 단일 연결을 관리할 수 있도록 허용 AWS 리전

다음 권한 정책을 통해 사용자는 단일 지역 간 CodeCatalyst 및 지역 AWS 계정 내 연결을 관리할 수 있습니다. 이 정책에서는 us-west-2 지역에서의 작업만 허용하고 다른 지역의 작업은 허용하지 않는 조건을 사용합니다. AWS 리전연결을 생성한 후 에서 옵션을 선택하여 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할을 생성할 수 있습니다. AWS Management Console예제 정책의 `iam:PassRole` 작업 조건에는 에 대한 서비스 주체가 포함됩니다. CodeCatalyst 해당 액세스 권한을 가진 역할만 에서 생성됩니다. AWS Management Console

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codecatalyst:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-west-2"
                }
            }
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "codecatalyst.amazonaws.com",
                        "codecatalyst-runner.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

예 3: 연결 관리 거부

다음 권한 정책은 사용자가 및 사이의 CodeCatalyst 연결을 관리할 수 있는 권한을 거부합니다. AWS 계정

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "codecatalyst:*"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

태그를 사용하여 계정 연결 리소스에 대한 액세스 제어

태그는 리소스에 첨부하거나 요청을 통해 태그 지정을 지원하는 서비스에 전달할 수 있습니다. 정책의 리소스에는 태그가 있을 수 있으며 정책의 일부 작업에는 태그가 포함될 수 있습니다. 태깅 조건 키에는 `aws:RequestTag` 및 `aws:ResourceTag` 조건 키가 포함됩니다. IAM 정책을 생성하면 태그 조건 키를 사용하여 다음을 제어할 수 있습니다.

- 연결 리소스에 이미 있는 태그를 기반으로 연결 리소스에서 작업을 수행할 수 있는 사용자
- 작업의 요청에서 전달될 수 있는 태그
- 요청에서 특정 키를 사용할 수 있는지 여부를 제어합니다.

다음 예제는 CodeCatalyst 계정 연결 사용자를 위한 정책에서 태그 조건을 지정하는 방법을 보여줍니다. 조건 키에 대한 자세한 내용은 IAM의 정책 조건 키 섹션을 참조하세요.

예 1: 요청의 태그를 기반으로 한 작업 허용

다음 정책은 사용자에게 계정 연결을 승인할 권한을 부여합니다.

이와 관련하여 정책은 요청이 `ProjectA` 값이 포함된 `Project` 태그를 지정하는 경우 `AcceptConnection` 및 `TagResource` 작업을 허용합니다. `aws:RequestTag` 조건 키는 IAM 요청에서 전달할 수 있는 태그를 제어하는 데 사용됩니다. `aws:TagKeys` 조건은 태그 키의 대/소문자를 구분합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:AcceptConnection",
        "codecatalyst:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Project": "ProjectA"
        },
```

```
        "ForAllValues:StringEquals": {
          "aws:TagKeys": ["Project"]
        }
      }
    }
  ]
}
```

예 2: 리소스 태그에 기반한 작업 허용

다음 정책은 사용자에게 계정 연결 리소스에 대한 작업을 수행하고 해당 리소스에 대한 정보를 얻을 수 있는 권한을 부여합니다.

이를 위해 연결에 값이 지정된 `Project` 태그가 있는 경우 특정 작업이 허용됩니다`ProjectA`. `aws:ResourceTag` 조건 키는 IAM 요청에서 전달할 수 있는 태그를 제어하는 데 사용됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecatalyst:GetConnection",
        "codecatalyst:DeleteConnection",
        "codecatalyst:AssociateIamRoleToConnection",
        "codecatalyst:DisassociateIamRoleFromConnection",
        "codecatalyst:ListIamRolesForConnection",
        "codecatalyst:PutBillingAuthorization"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "ProjectA"
        }
      }
    }
  ]
}
```

CodeCatalyst 권한 참조

이 섹션에서는 연결된 계정 연결 리소스와 함께 사용되는 작업에 대한 AWS 계정 권한 참조를 제공합니다 CodeCatalyst. 다음 섹션에서는 계정 연결과 관련된 권한 전용 작업에 대해 설명합니다.

계정 연결에 필요한 권한

계정 연결 작업에 필요한 권한은 다음과 같습니다.

CodeCatalyst 계정 연결에 대한 권한	필요한 권한	리소스
AcceptConnection	이 계정을 CodeCatalyst 스페이스에 연결하라는 요청을 수락하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	정책 Resource 요소에는 와일드카드(*)만 지원됩니다.
AssociateIamRoleToConnection	IAM 역할을 계정 연결에 연결하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
DeleteConnection	계정 연결을 삭제하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
DisassociateIamRoleFromConnection	계정 연결에서 IAM 역할을 분리하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
GetBillingAuthorization	계정 연결에 대한 결제 승인을 설명하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
GetConnection	계정을 연결하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*

CodeCatalyst 계정 연결에 대한 권한	필요한 권한	리소스
GetPendingConnection	이 계정을 CodeCatalyst 스페이스에 연결해 달라는 보류 중인 요청을 받는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	정책 Resource 요소에는 와일드카드(*)만 지원됩니다.
ListConnections	보류 중이 아닌 계정 연결을 나열하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	정책 Resource 요소에는 와일드카드(*)만 지원됩니다.
ListIamRolesForConnection	계정 연결과 관련된 IAM 역할을 나열하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
ListTagsForResource	계정 연결과 관련된 태그를 나열하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
PutBillingAuthorization	계정 연결을 위한 결제 승인을 생성하거나 업데이트하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
RejectConnection	이 계정을 CodeCatalyst 스페이스에 연결하라는 요청을 거부하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	정책 Resource 요소에는 와일드카드(*)만 지원됩니다.

CodeCatalyst 계정 연결에 대한 권한	필요한 권한	리소스
TagResource	계정 연결과 관련된 태그를 만들거나 편집하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*
UntagResource	계정 연결과 관련된 태그를 제거하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/connections/ *connection_ID*

IAM ID 센터 애플리케이션에 필요한 권한

IAM ID 센터 애플리케이션을 사용하려면 다음 권한이 필요합니다.

CodeCatalyst IAM ID 센터 애플리케이션에 대한 권한	필요한 권한	리소스
AssociateIdentityCenterApplicationToSpace	IAM ID 센터 애플리케이션을 스페이스와 연결하는 데 필요합니다. CodeCatalyst IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
AssociateIdentityToIdentityCenterApplication	스페이스의 IAM ID 센터 애플리케이션과 ID를 연결하는 데 필요합니다. CodeCatalyst IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecatalyst:region: *account_ID* :/identity-center-applications/ *identity-center-application_ID*
BatchAssociateIdentitiesToIdentityCenterApplication	스페이스의 IAM ID 센터 애플리케이션과 여러 ID를 연결하는 데 필요합니다. CodeCatal	arn:aws:codecatalyst:region: *account_ID* :/identity-

CodeCatalyst IAM ID 센터 애플리케이션에 대한 권한	필요한 권한	리소스
	yst IAM 정책 권한일 뿐, API 작업이 아닙니다.	center-applicati ons/ *identity-center-application_ID*
BatchDisassociateIdentities FromIdentityCenterApplication	스페이스의 IAM Identity Center 애플리케이션에서 여러 ID를 분리하는 데 필요합니다. CodeCatalyst IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecataly st:region: *account_I D* :/identity- center-applicati ons/ *identity-center-application_ID*
CreateIdentityCenterApplica tion	IAM ID 센터 애플리케이션을 생성하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecataly st:region: *account_I D* :/identity- center-applicati ons/ *identity-center-application_ID*
CreateSpaceAdminRo leAssignment	지정된 CodeCatalyst 공간 및 IAM Identity Center 애플리케이션에 대한 관리자 역할 할당을 생성하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecataly st:region: *account_I D* :/identity- center-applicati ons/ *identity-center-application_ID*
DeleteIdentityCenterApplica tion	IAM ID 센터 애플리케이션을 삭제하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	arn:aws:codecataly st:region: *account_I D* :/identity- center-applicati ons/ *identity-center-application_ID*

CodeCatalyst IAM ID 센터 애플리케이션에 대한 권한	필요한 권한	리소스
DisassociateIdentityCenterApplicationFromSpace	스페이스에서 IAM ID 센터 애플리케이션을 분리하는 데 필요합니다. CodeCatalyst IAM 정책 권한일 뿐, API 작업이 아닙니다.	`arn:aws:codecatalyst:region:` *account_ID* `:/identity-center-applications/` *identity-center-application_ID*
DisassociateIdentityFromIdentityCenterApplication	스페이스의 IAM ID 센터 애플리케이션에서 ID 연결을 끊는 데 필요합니다. CodeCatalyst IAM 정책 권한일 뿐, API 작업이 아닙니다.	`arn:aws:codecatalyst:region:` *account_ID* `:/identity-center-applications/` *identity-center-application_ID*
GetIdentityCenterApplication	IAM ID 센터 애플리케이션에 대한 정보를 가져오는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	`arn:aws:codecatalyst:region:` *account_ID* `:/identity-center-applications/` *identity-center-application_ID*
ListIdentityCenterApplications	계정의 모든 IAM ID 센터 애플리케이션 목록을 보는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	정책 `Resource` 요소에는 와일드카드(*)만 지원됩니다.
ListIdentityCenterApplicationsForSpace	공간별 CodeCatalyst IAM ID 센터 애플리케이션 목록을 보는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	`arn:aws:codecatalyst:region:` *account_ID* `:/identity-center-applications/` *identity-center-application_ID*

CodeCatalyst IAM ID 센터 애플리케이션에 대한 권한	필요한 권한	리소스
ListSpacesForIdentityCenterApplication	IAM ID 센터 애플리케이션별로 CodeCatalyst 공간 목록을 보는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	`arn:aws:codecatalyst:region:` *`account_ID`* `:/identity-center-applications/` *`identity-center-application_ID`*
SynchronizeIdentityCenterApplication	IAM ID 센터 애플리케이션을 지원 ID 스토어와 동기화하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	`arn:aws:codecatalyst:region:` *`account_ID`* `:/identity-center-applications/` *`identity-center-application_ID`*
UpdateIdentityCenterApplication	IAM ID 센터 애플리케이션을 업데이트하는 데 필요합니다. IAM 정책 권한일 뿐, API 작업이 아닙니다.	`arn:aws:codecatalyst:region:` *`account_ID`* `:/identity-center-applications/` *`identity-center-application_ID`*

CodeCatalyst에 서비스 연결 역할 사용

CodeCatalyst Amazon은 AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 직접 연결되는 고유한 유형의 IAM 역할입니다. CodeCatalyst 서비스 연결 역할은 사전 정의되며 서비스가 사용자를 CodeCatalyst 대신하여 다른 서비스를 호출하는 데 필요한 모든 권한을 포함합니다. AWS

서비스에 연결된 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 설정이 CodeCatalyst 더 쉬워집니다. CodeCatalyst 서비스 연결 역할의 권한을 정의하며, 달리 정의되지 않는 한 해당 역할만 CodeCatalyst 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 CodeCatalyst 리소스에 대한 액세스 권한을 실수로 제거할 수 없으므로 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할 열에 예가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

에 대한 서비스 연결 역할 권한 CodeCatalyst

CodeCatalyst 라는 서비스 연결 역할을 사용합니다. AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization— 사용자를 대신하여 애플리케이션 인스턴스 프로필과 관련 디렉터리 사용자 및 그룹에 대한 Amazon의 CodeCatalyst 읽기 전용 액세스를 허용합니다.

AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `codecatalyst.amazonaws.com`

이름이 지정된 역할 권한 정책을 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy 사용하면 지정된 리소스에서 다음 작업을 CodeCatalyst 완료할 수 있습니다.

- `View application instance profiles and associated directory users and groups`조치: `CodeCatalyst spaces that support identity federation and SSO users and groups`

사용자, 그룹 또는 역할이 서비스 연결 역할을 생성, 편집 또는 삭제할 수 있도록 사용 권한을 구성해야 합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하십시오.

에 대한 서비스 연결 역할 생성 CodeCatalyst

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management ConsoleAWS CLI, 또는 AWS API에서 스페이스를 생성하면 서비스 연결 CodeCatalyst 역할이 자동으로 생성됩니다.

> **Important**
>
> 이러한 서비스 연결 역할은 해당 역할이 지원하는 기능을 사용하는 다른 서비스에서 작업을 완료했을 경우 계정에 나타날 수 있습니다. 또한 CodeCatalyst 서비스 연결 역할을 지원하기 시

작한 2023년 11월 17일 이전에 서비스를 사용하고 있었다면 계정에 역할을 CodeCatalyst 생성하십시오AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization. 자세한 내용은 내 AWS 계정에 표시되는 새 역할을 참조하십시오.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 스페이스를 만들면 서비스 연결 역할이 다시 CodeCatalyst 생성됩니다.

또한 IAM 콘솔을 사용하여 View 애플리케이션 인스턴스 프로필과 관련 디렉터리 사용자 및 그룹 사용 사례를 사용하여 서비스 연결 역할을 생성할 수 있습니다. AWS CLI 또는 AWS API에서 `codecatalyst.amazonaws.com` 서비스 이름의 서비스 연결 역할을 생성합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 생성을 참조하십시오. 이 서비스 연결 역할을 삭제하면 동일한 프로세스를 사용하여 역할을 다시 생성할 수 있습니다.

에 대한 서비스 연결 역할 편집 CodeCatalyst

CodeCatalyst AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization서비스 연결 역할을 편집할 수 없습니다. 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하십시오.

에 대한 서비스 연결 역할 삭제 CodeCatalyst

AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 역할은 수동으로 삭제할 필요가 없습니다. AWS Management ConsoleAWS CLI, 또는 AWS API에서 스페이스를 삭제하면 리소스가 CodeCatalyst 정리되고 서비스 연결 역할이 자동으로 삭제됩니다.

또한 IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 서비스 연결 역할을 수동으로 삭제할 수 있습니다. 단, 서비스 연결 역할에 대한 리소스를 먼저 정리해야 수동으로 삭제할 수 있습니다.

> **Note**
>
> CodeCatalyst 서비스가 역할을 사용하고 있을 때 리소스를 삭제하려고 하면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하십시오.

에서 사용하는 CodeCatalyst 리소스를 삭제하려면 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization

- 스페이스를 삭제합니다.

IAM을 사용하여 수동으로 서비스 링크 역할을 삭제하려면

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 서비스 연결 역할을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하세요.

CodeCatalyst 서비스 연결 역할이 지원되는 지역

CodeCatalyst 서비스를 사용할 수 있는 모든 지역에서 서비스 연결 역할을 사용할 수 있습니다. 자세한 내용은 AWS 리전 및 엔드포인트 단원을 참조하십시오.

CodeCatalyst 서비스를 사용할 수 있는 모든 지역에서 서비스 연결 역할을 사용할 수 있는 것은 아닙니다. 다음 리전에서 AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronization 역할을 사용할 수 있습니다.

지역명	리전 자격 증명	에서 지원 CodeCatalyst
미국 동부(버지니아 북부)	us-east-1	아니요
미국 동부(오하이오)	us-east-2	아니요
미국 서부(캘리포니아 북부)	us-west-1	아니요
미국 서부(오레곤)	us-west-2	예
아프리카(케이프타운)	af-south-1	아니요
아시아 태평양(홍콩)	ap-east-1	아니요
아시아 태평양(자카르타)	ap-southeast-3	아니요
아시아 태평양(뭄바이)	ap-south-1	아니요
아시아 태평양(오사카)	ap-northeast-3	아니요
아시아 태평양(서울)	ap-northeast-2	아니요
아시아 태평양(싱가포르)	ap-southeast-1	아니요
아시아 태평양(시드니)	ap-southeast-2	아니요
아시아 태평양(도쿄)	ap-northeast-1	아니요

지역명	리전 자격 증명	에서 지원 CodeCatalyst
캐나다(중부)	ca-central-1	아니요
유럽(프랑크푸르트)	eu-central-1	아니요
유럽(아일랜드)	eu-west-1	예
유럽(런던)	eu-west-2	아니요
유럽(밀라노)	eu-south-1	아니요
유럽(파리)	eu-west-3	아니요
유럽(스톡홀름)	eu-north-1	아니요
중동(바레인)	me-south-1	아니요
중동(UAE)	me-central-1	아니요
남아메리카(상파울루)	sa-east-1	아니요
AWS GovCloud (미국 동부)	us-gov-east-1	아니요
AWS GovCloud (미국 서부)	us-gov-west-1	아니요

AWS아마존 관리형 정책 CodeCatalyst

AWS 관리형 정책은 AWS에 의해 생성되고 관리되는 독립 실행형 정책입니다. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에서 정의한 권한은 변경할 수 없습니다. AWS에서 AWS 관리형 정책에 정의된 권한을 업데이트할 경우 정책이 연결되어 있는 모든 보안 주체 엔터티(사용자, 그룹 및 역할)에도 업데

이트가 적용됩니다. 새로운 AWS 서비스를 시작하거나 새로운 API 작업을 기존 서비스에 이용하는 경우 AWS가 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: AmazonCodeCatalystSupportAccess

이 정책은 모든 스페이스 관리자와 스페이스 구성원이 스페이스 결제 계정과 관련된 Business 또는 Enterprise 프리미엄 지원 플랜을 이용할 수 있는 권한을 부여하는 정책입니다. 이러한 권한을 통해 스페이스 관리자와 구성원은 권한 정책 내에서 CodeCatalyst 사용 권한이 있는 리소스에 대한 프리미엄 지원 플랜을 활용할 수 있습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `support`— 사용자가 AWS Support 사례를 검색, 생성 및 해결할 수 있는 권한을 부여합니다. 또한 커뮤니케이션, 심각도 수준, 첨부 파일 및 관련 지원 사례 세부 정보를 설명할 권한을 부여합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "support:DescribeAttachment",
        "support:DescribeCaseAttributes",
        "support:DescribeCases",
        "support:DescribeCommunications",
        "support:DescribeIssueTypes",
        "support:DescribeServices",
        "support:DescribeSeverityLevels",
        "support:DescribeSupportLevel",
```

```
        "support:SearchForCases",
        "support:AddAttachmentsToSet",
        "support:AddCommunicationToCase",
        "support:CreateCase",
        "support:InitiateCallForCase",
        "support:InitiateChatForCase",
        "support:PutCaseAttributes",
        "support:RateCaseCommunication",
        "support:ResolveCase"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS 관리형 정책: AmazonCodeCatalystFullAccess

이 정책은 Amazon CodeCatalyst Spaces 페이지의 CodeCatalyst 스페이스 및 연결된 계정을 관리할 권한을 부여하는 AWS Management Console 정책입니다. 이 애플리케이션은 내 공간에 AWS 계정 연결되도록 구성하는 데 사용됩니다 CodeCatalyst.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `codecatalyst`— 의 Amazon CodeCatalyst Spaces 페이지에 전체 권한을 부여합니다AWS Management Console.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CodeCatalystResourceAccess"
            "Effect": "Allow",
            "Action": [
                "codecatalyst:*",
                "iam:ListRoles"
            ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "CodeCatalystAssociateIAMRole"
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "codecatalyst.amazonaws.com",
                        "codecatalyst-runner.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

AWS 관리형 정책: AmazonCodeCatalystReadOnlyAccess

이는 Amazon CodeCatalyst Spaces 페이지의 스페이스 및 연결된 계정에 대한 정보를 보고 나열할 권한을 부여하는 AWS Management Console 정책입니다. 이 애플리케이션은 내 공간에 AWS 계정 연결되도록 구성하는 데 사용됩니다 CodeCatalyst.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `codecatalyst`— 의 Amazon CodeCatalyst Spaces 페이지에 읽기 전용 권한을 부여합니다AWS Management Console.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "codecatalyst:Get*",
                "codecatalyst:List*",
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 관리형 정책:
AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy

AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy; 를 IAM 엔티티에 첨부할 수 없습니다. 이 정책은 사용자를 CodeCatalyst 대신하여 작업을 수행할 수 있는 서비스 연결 역할에 연결됩니다. 자세한 설명은 CodeCatalyst에 서비스 연결 역할 사용 섹션을 참조하세요.

이 정책을 통해 고객은 에서 공간을 관리할 때 애플리케이션 인스턴스 프로필과 관련 디렉터리 사용자 및 그룹을 볼 수 있습니다. CodeCatalyst 고객은 ID 페더레이션과 SSO 사용자 및 그룹을 지원하는 공간을 관리할 때 이러한 리소스를 볼 수 있습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- sso— 사용자가 IAM Identity Center에서 관리되는 애플리케이션 인스턴스 프로필을 내 관련 스페이스에 대해 볼 수 있는 권한을 부여합니다. CodeCatalyst

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid":
 "AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy",
   "Effect": "Allow",
   "Action": [
    "sso:ListInstances",
    "sso:ListApplications",
```

```
    "sso:ListApplicationAssignments",
    "sso:DescribeInstance",
    "sso:DescribeApplication"
   ],
   "Resource": "*"
  }
 ]
}
```

AWS 관리형 정책으로 CodeCatalyst 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 CodeCatalyst 이후의 AWS 관리형 정책 업데이트에 대한 세부 정보를 볼 수 있습니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 CodeCatalyst 문서 기록 페이지에서 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
AmazonCodeCatalystServiceRoleForIdentityCenterApplicationSynchronizationPolicy - 새 정책	CodeCatalyst 정책을 추가했습니다. CodeCatalyst 사용자에게 애플리케이션 인스턴스 프로필과 관련 디렉터리 사용자 및 그룹을 볼 수 있는 권한을 부여합니다.	2023년 11월 17일
AmazonCodeCatalystSupportAccess - 새 정책	CodeCatalyst 정책을 추가했습니다. CodeCatalyst 사용자가 지원 사례를 검색, 작성 및 해결하고 관련 커뮤니케이션 및 세부 정보를 볼 수 있는 권한을 부여합니다.	2023년 4월 20일
AmazonCodeCatalystFullAccess - 새 정책	CodeCatalyst 정책을 추가했습니다.	2023년 4월 20일

변경 사항	설명	날짜
	에 대한 전체 액세스 권한을 CodeCatalyst 부여합니다.	
AmazonCodeCatalyst ReadOnlyAccess - 새 정책	CodeCatalyst 정책을 추가했습니다. 에 대한 읽기 전용 액세스 권한을 CodeCatalyst 부여합니다.	2023년 4월 20일
CodeCatalyst 변경 내용 추적 시작	CodeCatalyst AWS관리형 정책의 변경 사항 추적을 시작했습니다.	2023년 4월 20일

IAM 역할을 사용하여 프로젝트 AWS 리소스에 대한 액세스 권한 부여

CodeCatalyst AWS CodeCatalyst 스페이스에 연결하여 AWS 계정 리소스에 액세스할 수 있습니다. 그런 다음 다음 서비스 역할을 생성하고 계정을 연결할 때 이들을 연결할 수 있습니다.

JSON 정책에서 사용하는 요소에 대한 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

- CodeCatalyst 프로젝트 및 AWS 계정 워크플로용 리소스에 액세스하려면 먼저 사용자를 대신하여 해당 리소스에 대한 CodeCatalyst 액세스 권한을 부여해야 합니다. 이렇게 하려면 스페이스의 사용자 및 프로젝트를 CodeCatalyst 대신할 수 AWS 계정 있는 서비스 역할을 커넥터에 생성해야 합니다. 서비스 역할을 생성하여 사용하도록 선택하거나 사용자 지정 CodeCatalystWorkflowDevelopmentRole-*spaceName*서비스 역할을 생성하고 이러한 IAM 정책 및 역할을 수동으로 구성할 수 있습니다. 가장 좋은 방법은 이러한 역할에 필요한 최소한의 권한을 할당하는 것입니다.

  > **Note**
  >
  > 사용자 지정된 서비스 역할의 경우 CodeCatalyst 서비스 주체가 필요합니다. CodeCatalyst 서비스 주체 및 신뢰 모델에 대한 자세한 내용은 을 참조하십시오 CodeCatalyst 신뢰 모델 이해.

- 연결을 AWS 계정통해 스페이스에 대한 지원을 관리하려면 CodeCatalyst 사용자가 지원에 액세스할 수 있는 AWSRoleForCodeCatalystSupport서비스 역할을 만들고 사용하도록 선택할 수 있습니다. CodeCatalyst 스페이스 지원에 대한 자세한 내용은 을 참조하십시오AWS Support아마존용 CodeCatalyst.

CodeCatalystWorkflowDevelopmentRole-*spaceName*서비스 역할 이해

연결된 AWS 계정공간에서 리소스를 생성하고 액세스하는 데 사용할 CodeCatalyst 수 있는 IAM 역할을 스페이스에 추가할 수 있습니다. 이를 서비스 역할이라고 합니다. 서비스 역할을 만드는 가장 간단한 방법은 공간을 만들 때 서비스 역할을 추가하고 해당 역할에 대한 CodeCatalystWorkflowDevelopmentRole-*spaceName*옵션을 선택하는 것입니다. 이렇게 하면 `AdministratorAccess` 연결된 서비스 역할이 만들어질 뿐만 아니라 스페이스의 프로젝트에서 사용자를 대신하여 역할을 맡을 수 CodeCatalyst 있는 신뢰 정책도 만들어집니다. 서비스 역할의 범위는 개별 프로젝트가 아니라 스페이스로 제한됩니다. 이 역할을 생성하려면 계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-*spaceName*역할 만들기 단원을 참조하세요. 각 계정의 각 스페이스에 대해 하나의 역할만 만들 수 있습니다.

> **Note**
>
> 이 역할은 개발 계정에만 사용하는 것이 좋으며 `AdministratorAccess` AWS 관리형 정책을 사용하므로 이 계정에서 새 정책 및 리소스를 만들 수 있는 전체 액세스 권한을 AWS 계정부여합니다.

CodeCatalystWorkflowDevelopmentRole-*spaceName*역할에 첨부된 정책은 스페이스에서 블루프린트를 사용하여 만든 프로젝트에 사용할 수 있도록 설계되었습니다. 이를 통해 해당 프로젝트의 사용자는 커넥티드 AWS 계정환경의 리소스를 사용하여 코드를 개발, 빌드, 테스트 및 배포할 수 있습니다. 자세한 내용은 AWS 서비스 역할 만들기를 참조하십시오.

CodeCatalystWorkflowDevelopmentRole-*spaceName*역할에 연결된 정책은 의 `AdministratorAccess` 관리형 정책입니다 AWS. 이 정책은 모든 AWS 작업과 리소스에 대한 전체 액세스 권한을 부여하는 정책입니다. IAM 콘솔에서 JSON 정책 문서를 보려면 을 참조하십시오. AdministratorAccess

다음 신뢰 정책은 역할을 CodeCatalyst 맡을 CodeCatalystWorkflowDevelopmentRole-*spaceName*수 있도록 허용합니다. CodeCatalyst 신뢰 모델에 대한 자세한 내용은 을 참조하십시오 CodeCatalyst 신뢰 모델 이해.

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
             "Principal": {
                "Service": [
                    "codecatalyst-runner.amazonaws.com",
                    "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:codecatalyst:::space/spaceId/project/*"
                }
            }
        }
    ]
```

계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-*spaceName*역할 만들기

다음 단계에 따라 스페이스의 워크플로에 사용할 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 역할을 만드세요. 프로젝트에서 사용할 IAM 역할을 할당하려는 각 계정에 대해 개발자 역할과 같은 역할을 스페이스에 추가해야 합니다.

시작하기 전에 관리자 권한이 AWS 계정 있거나 관리자와 협력할 수 있어야 합니다. 에서 CodeCatalyst IAM 역할을 사용하는 방법 AWS 계정 및 IAM 역할에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정.

생성 및 추가하기 CodeCatalyst CodeCatalystWorkflowDevelopmentRole-*spaceName*

1. CodeCatalyst 콘솔에서 시작하기 전에 를 AWS Management Console열고 AWS 계정 스페이스에 동일한 계정으로 로그인했는지 확인하십시오.
2. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 여십시오.
3. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.
4. 역할을 생성하려는 AWS 계정 위치의 링크를 선택합니다. AWS 계정 세부 정보 페이지가 표시됩니다.
5. 에서 역할 관리를 선택합니다 AWS Management Console.

Amazon CodeCatalyst 스페이스에 IAM 역할 추가 페이지가 에서 AWS Management Console열립니다. 아마존 CodeCatalyst 스페이스 페이지입니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.

6. IAM에서 CodeCatalyst 개발 관리자 역할 생성을 선택합니다. 이 옵션은 개발 역할에 대한 권한 정책 및 신뢰 정책을 포함하는 서비스 역할을 생성합니다. 역할에는 이름이 `CodeCatalystWorkflowDevelopmentRole-spaceName` 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오CodeCatalystWorkflowDevelopmentRole-**spaceName**서비스 역할 이해.

   > **Note**
   >
   > 이 역할은 개발자 계정에만 사용하는 것이 좋으며 `AdministratorAccess` AWS 관리형 정책을 사용하므로 이 계정에서 새 정책 및 리소스를 만들 수 있는 전체 액세스 권한을 AWS 계정부여합니다.

7. 개발 역할 생성을 선택합니다.
8. 연결 페이지의 사용 가능한 IAM 역할 아래에서 계정에 CodeCatalyst 추가된 IAM 역할 목록의 역할을 확인합니다. `CodeCatalystWorkflowDevelopmentRole-spaceName`
9. 숙소로 돌아가려면 Amazon으로 이동을 선택하십시오 CodeCatalyst.

AWSRoleForCodeCatalystSupport서비스 역할 이해

스페이스의 CodeCatalyst 사용자가 지원 사례를 생성하고 액세스하는 데 사용할 수 있는 스페이스의 IAM 역할을 추가할 수 있습니다. 이를 지원용 서비스 역할이라고 합니다. 지원용 서비스 역할을 생성하는 가장 간단한 방법은 공간을 생성할 때 하나를 추가하고 해당 역할에 대한 `AWSRoleForCodeCatalystSupport` 옵션을 선택하는 것입니다. 이렇게 하면 정책과 역할이 만들어질 뿐만 아니라 스페이스의 프로젝트에서 사용자를 대신하여 역할을 CodeCatalyst 맡을 수 있는 신뢰 정책도 만들어집니다. 서비스 역할의 범위는 개별 프로젝트가 아니라 스페이스로 제한됩니다. 이 역할을 생성하려면 계정 및 스페이스에 대한 AWSRoleForCodeCatalystSupport역할 생성 단원을 참조하세요.

`AWSRoleForCodeCatalystSupport`역할에 연결된 정책은 지원 권한에 대한 액세스를 제공하는 관리형 정책입니다. 자세한 정보는 AWS 관리형 정책: AmazonCodeCatalystSupportAccess을 참조하세요.

정책의 신뢰 역할을 통해 역할을 CodeCatalyst 수임할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "codecatalyst.amazonaws.com",
                    "codecatalyst-runner.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

계정 및 스페이스에 대한 AWSRoleForCodeCatalystSupport역할 생성

다음 단계에 따라 스페이스의 지원 사례에 사용할 `AWSRoleForCodeCatalystSupport` 역할을 만드십시오. 스페이스의 지정된 결제 계정에 역할을 추가해야 합니다.

시작하기 전에 관리자 권한이 AWS 계정 있거나 관리자와 함께 작업할 수 있어야 합니다. 에서 CodeCatalyst IAM 역할을 사용하는 방법 AWS 계정 및 IAM 역할에 대한 자세한 내용은 을 참조하십시오연결된 AWS 리소스에 대한 액세스 허용 AWS 계정.

생성 및 추가하기 CodeCatalyst AWSRoleForCodeCatalystSupport

1. CodeCatalyst 콘솔에서 시작하기 전에 를 AWS Management Console열고 AWS 계정 스페이스에 동일한 계정으로 로그인했는지 확인하십시오.
2. CodeCatalyst 스페이스로 이동하세요. 설정을 선택한 다음 AWS 계정를 선택합니다.
3. 역할을 생성하려는 AWS 계정 위치의 링크를 선택합니다. AWS 계정 세부 정보 페이지가 표시됩니다.
4. 에서 역할 관리를 선택합니다 AWS Management Console.

   Amazon CodeCatalyst 스페이스에 IAM 역할 추가 페이지가 에서 AWS Management Console열립니다. 아마존 CodeCatalyst 스페이스 페이지입니다. 페이지에 액세스하려면 로그인해야 할 수 있습니다.
5. CodeCatalyst 스페이스 세부 정보에서 CodeCatalyst Support 역할 추가를 선택합니다. 이 옵션은 미리 보기 개발 역할에 대한 권한 정책 및 신뢰 정책을 포함하는 서비스 역할을 생성합니다. 역

할에는 고유 식별자가 AWSRoleForCodeCatalystSupport추가된 이름이 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오AWSRoleForCodeCatalystSupport서비스 역할 이해.

6. CodeCatalyst Support용 역할 추가 페이지에서 기본값을 선택된 상태로 두고 역할 생성을 선택합니다.
7. 사용 가능한 IAM 역할에서 계정에 `CodeCatalystWorkflowDevelopmentRole-spaceName` 추가된 IAM 역할 목록에서 역할을 확인하십시오. CodeCatalyst
8. 숙소로 돌아가려면 Amazon으로 이동을 선택하십시오 CodeCatalyst.

워크플로우 작업에 대한 IAM 역할 구성 CodeCatalyst

이 섹션에서는 계정에 사용하기 위해 생성할 수 있는 IAM 역할 및 정책을 자세히 설명합니다. CodeCatalyst 예제 역할 생성에 대한 지침은 을 참조하십시오워크플로 작업을 위한 역할을 수동으로 생성. IAM 역할을 생성한 후 역할 ARN을 복사하여 계정 연결에 IAM 역할을 추가하고 프로젝트 환경에 연결합니다. 자세한 내용은 계정 연결에 IAM 역할 추가 섹션을 참조하세요.

CodeCatalyst Amazon S3 액세스를 위한 빌드 역할

CodeCatalyst 워크플로 빌드 작업의 경우 기본 CodeCatalystWorkflowDevelopmentRole-***spaceName***서비스 역할을 사용하거나 CodeCatalystBuildRoleforS3Access라는 IAM 역할을 생성할 수 있습니다. 이 역할은 내 리소스에서 AWS CloudFormation 작업을 CodeCatalyst 실행해야 하는 권한 범위가 지정된 정책을 사용합니다. AWS 계정

이 역할은 다음 작업을 수행할 수 있는 권한을 부여합니다.

- Amazon S3 버킷에 쓰기.
- 를 통한 자원 구축을 지원합니다 AWS CloudFormation. 이를 위해서는 Amazon S3 액세스가 필요합니다.

이 역할은 다음 정책을 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "s3:PutObject",
        "iam:PassRole"
    ],
    "Resource": "resource_ARN",
```

```
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 빌드 역할: AWS CloudFormation

CodeCatalyst 워크플로 빌드 작업의 경우 기본 CodeCatalystWorkflowDevelopmentRole-***spaceName***서비스 역할을 사용하거나 필요한 권한을 가진 IAM 역할을 생성할 수 있습니다. 이 역할은 내 AWS CloudFormation 리소스에서 작업을 CodeCatalyst 실행해야 하는 권한 범위가 지정된 정책을 사용합니다. AWS 계정

이 역할은 다음 작업을 수행할 수 있는 권한을 부여합니다.

- 를 통한 자원 구축을 지원합니다 AWS CloudFormation. 이는 Amazon S3 액세스를 위한 CodeCatalyst 빌드 역할 및 CodeCatalyst 배포 역할과 함께 필요합니다 AWS CloudFormation.

이 역할에는 다음과 같은 AWS 관리형 정책을 연결해야 합니다.

- AWSCloudFormationFullAccess
- IAM FullAccess
- 아마존 3 FullAccess
- 아마존 API GatewayAdministrator
- AWSLambdaFullAccess

CodeCatalyst CDK를 위한 빌드 역할

모던 3계층 웹 애플리케이션과 같이 CDK 빌드 작업을 실행하는 CodeCatalyst 워크플로의 경우 기본 CodeCatalystWorkflowDevelopmentRole-***spaceName***서비스 역할을 사용하거나 필요한 권한이 있는 IAM 역할을 생성할 수 있습니다. 이 역할은 범위 지정 권한이 있는 정책을 사용하며, 이 정책은 해당 리

소스에 대해 부트스트랩하고 CDK 빌드 명령을 CodeCatalyst 실행해야 합니다. AWS CloudFormation AWS 계정

이 역할은 다음 작업을 수행할 수 있는 권한을 부여합니다.

- Amazon S3 버킷에 쓰기.
- CDK 구조 및 AWS CloudFormation 리소스 스택 구축을 지원합니다. 이를 위해서는 아티팩트 스토리지를 위한 Amazon S3, 이미지 리포지토리 지원을 위한 Amazon ECR, 가상 인스턴스에 대한 시스템 거버넌스 및 모니터링을 위한 SSM에 액세스해야 합니다.

이 역할은 다음 정책을 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:*",
                "ecr:*",
                "ssm:*",
                "s3:*",
                "iam:PassRole",
                "iam:GetRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 역할 배포 대상: AWS CloudFormation

를 사용하는 CodeCatalyst AWS CloudFormation워크플로 배포 작업의 경우 기본 CodeCatalystWorkflowDevelopmentRole-*spaceName*서비스 역할을 사용하거나, 의 AWS CloudFormation 리소스에서 작업을 CodeCatalyst 실행해야 하는 범위 지정 권한이 있는 정책을 사용할 수 있습니다 AWS 계정.

이 역할은 다음을 수행할 수 있는 권한을 부여합니다.

- 함수를 CodeCatalyst 호출하여 블루/그린 배포를 수행할 수 있습니다. AWS CloudFormation
- CodeCatalyst 에서 스택과 변경 세트를 생성하고 업데이트할 수 있습니다. AWS CloudFormation

이 역할은 다음 정책을 사용합니다.

```
{"Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
        "cloudformation:ValidateTemplate",
        "cloudformation:List*",
        "iam:PassRole"
    ],
    "Resource": "resource_ARN",
    "Effect": "Allow"
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

CodeCatalyst Amazon EC2의 배포 역할

CodeCatalyst 워크플로 배포 작업은 필요한 권한이 있는 IAM 역할을 사용합니다. 이 역할은 사용자의 Amazon EC2 리소스에서 작업을 CodeCatalyst 실행해야 하는 권한 범위가 지정된 정책을 사용합니다. AWS 계정CodeCatalystWorkflowDevelopmentRole-***spaceName***역할의 기본 정책에는 Amazon EC2 또는 Amazon EC2 Auto Scaling에 대한 권한이 포함되지 않습니다.

이 역할은 다음을 수행할 수 있는 권한을 부여합니다.

- Amazon EC2 배포를 생성합니다.
- 인스턴스의 태그를 읽거나 Auto Scaling 그룹 이름으로 Amazon EC2 인스턴스를 식별합니다.
- Amazon EC2 Auto Scaling 그룹, 수명 주기 후크 및 크기 조정 정책을 읽고, 생성하고, 업데이트하고, 삭제합니다.
- Amazon SNS 주제로 정보를 게시합니다.
- CloudWatch 경보에 대한 정보를 검색합니다.
- Elastic Load Balancing을 읽고 업데이트합니다.

이 역할은 다음 정책을 사용합니다.

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
 "autoscaling:CompleteLifecycleAction",
"autoscaling:DeleteLifecycleHook",
"autoscaling:DescribeAutoScalingGroups",
"autoscaling:DescribeLifecycleHooks",
"autoscaling:PutLifecycleHook",
"autoscaling:RecordLifecycleActionHeartbeat",
"autoscaling:CreateAutoScalingGroup",
"autoscaling:UpdateAutoScalingGroup",
"autoscaling:EnableMetricsCollection",
"autoscaling:DescribePolicies",
"autoscaling:DescribeScheduledActions",
"autoscaling:DescribeNotificationConfigurations",
"autoscaling:SuspendProcesses",
"autoscaling:ResumeProcesses",
"autoscaling:AttachLoadBalancers",
```

```
"autoscaling:AttachLoadBalancerTargetGroups",
"autoscaling:PutScalingPolicy",
"autoscaling:PutScheduledUpdateGroupAction",
"autoscaling:PutNotificationConfiguration",
"autoscaling:PutWarmPool",
"autoscaling:DescribeScalingActivities",
"autoscaling:DeleteAutoScalingGroup",
"ec2:DescribeInstances",
"ec2:DescribeInstanceStatus",
"ec2:TerminateInstances",
"tag:GetResources",
"sns:Publish",
"cloudwatch:DescribeAlarms",
"cloudwatch:PutMetricAlarm",
"elasticloadbalancing:DescribeLoadBalancers",
"elasticloadbalancing:DescribeInstanceHealth",
"elasticloadbalancing:RegisterInstancesWithLoadBalancer",
"elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
"elasticloadbalancing:DescribeTargetGroups",
"elasticloadbalancing:DescribeTargetHealth",
"elasticloadbalancing:RegisterTargets",
"elasticloadbalancing:DeregisterTargets"
],
"Resource": "resource_ARN"
      }
   ]
}
```

Note

이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.

```
"Resource": "*"
```

CodeCatalyst Amazon ECS의 배포 역할

CodeCatalyst 워크플로 작업의 경우 필요한 권한을 가진 IAM 역할을 생성할 수 있습니다. 기본 CodeCatalystWorkflowDevelopmentRole-*spaceName*서비스 역할을 사용하거나 Lambda CodeCatalyst 배포에 사용할 배포 작업에 대한 IAM 역할을 생성할 수 있습니다. 이 역할은 사용자의

Amazon ECS 리소스에서 작업을 CodeCatalyst 실행해야 하는 권한 범위가 지정된 정책을 사용합니다. AWS 계정

이 역할은 다음을 수행할 수 있는 권한을 부여합니다.

- 연결에 지정된 계정으로 CodeCatalyst 사용자를 대신하여 롤링 Amazon ECS 배포를 시작합니다. CodeCatalyst
- Amazon ECS 작업 세트를 읽고, 업데이트하고, 삭제합니다.
- Elastic Load Balancing 대상그룹, 리스너 및 규칙을 업데이트합니다.
- Lambda 함수를 호출합니다.
- Amazon S3 버킷에 있는 개정 파일에 액세스합니다.
- 경보에 대한 정보를 검색합니다. CloudWatch
- Amazon SNS 주제로 정보를 게시합니다.

이 역할은 다음 정책을 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action":[
      "ecs:DescribeServices",
      "ecs:CreateTaskSet",
      "ecs:DeleteTaskSet",
      "ecs:ListClusters",
      "ecs:RegisterTaskDefinition",
      "ecs:UpdateServicePrimaryTaskSet",
      "ecs:UpdateService",
      "elasticloadbalancing:DescribeTargetGroups",
      "elasticloadbalancing:DescribeListeners",
      "elasticloadbalancing:ModifyListener",
      "elasticloadbalancing:DescribeRules",
      "elasticloadbalancing:ModifyRule",
      "lambda:InvokeFunction",
      "lambda:ListFunctions",
      "cloudwatch:DescribeAlarms",
      "sns:Publish",
      "sns:ListTopics",
      "s3:GetObject",
      "s3:GetObjectVersion",
      "codedeploy:CreateApplication",
```

```
      "codedeploy:CreateDeployment",
      "codedeploy:CreateDeploymentGroup",
      "codedeploy:GetApplication",
      "codedeploy:GetDeployment",
      "codedeploy:GetDeploymentGroup",
      "codedeploy:ListApplications",
      "codedeploy:ListDeploymentGroups",
      "codedeploy:ListDeployments",
      "codedeploy:StopDeployment",
      "codedeploy:GetDeploymentTarget",
      "codedeploy:ListDeploymentTargets",
      "codedeploy:GetDeploymentConfig",
      "codedeploy:GetApplicationRevision",
      "codedeploy:RegisterApplicationRevision",
      "codedeploy:BatchGetApplicationRevisions",
      "codedeploy:BatchGetDeploymentGroups",
      "codedeploy:BatchGetDeployments",
      "codedeploy:BatchGetApplications",
      "codedeploy:ListApplicationRevisions",
      "codedeploy:ListDeploymentConfigs",
      "codedeploy:ContinueDeployment"
   ],
   "Resource":"*",
   "Effect":"Allow"
},{"Action":[
      "iam:PassRole"
   ],
   "Effect":"Allow",
   "Resource":"*",
   "Condition":{"StringLike":{"iam:PassedToService":[
            "ecs-tasks.amazonaws.com",
            "codedeploy.amazonaws.com"
         ]
      }
   }
}]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.

```
"Resource": "*"
```

CodeCatalyst Lambda에 대한 배포 역할

CodeCatalyst 워크플로 작업의 경우 필요한 권한을 가진 IAM 역할을 생성할 수 있습니다. 기본 CodeCatalystWorkflowDevelopmentRole-*spaceName*서비스 역할을 사용하거나 Lambda CodeCatalyst 배포에 사용할 배포 작업에 대한 IAM 역할을 생성할 수 있습니다. 이 역할은 사용자의 Lambda 리소스에서 작업을 CodeCatalyst 실행해야 하는 범위 지정 권한이 있는 정책을 사용합니다. AWS 계정

이 역할은 다음을 수행할 수 있는 권한을 부여합니다.

- Lambda 함수 및 별칭을 읽고, 업데이트하고, 호출합니다.
- Amazon S3 버킷에 있는 개정 파일에 액세스합니다.
- 이벤트 경보에 대한 정보를 검색합니다. CloudWatch
- Amazon SNS 주제로 정보를 게시합니다.

이 역할은 다음 정책을 사용합니다.

```
*{*
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "lambda:UpdateAlias",
                "lambda:GetAlias",
                "lambda:GetProvisionedConcurrencyConfig",
                "sns:Publish"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
```

```
            ],
            "Resource": "arn:aws:s3:::/CodeDeploy/",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:::function:CodeDeployHook_*",
            "Effect": "Allow"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

CodeCatalyst Lambda에 대한 배포 역할

CodeCatalyst 워크플로 작업의 경우 기본 CodeCatalystWorkflowDevelopmentRole-***spaceName***서비스 역할을 사용하거나 필요한 권한이 있는 IAM 역할을 생성할 수 있습니다. 이 역할은 사용자의 Lambda 리소스에서 작업을 CodeCatalyst 실행해야 하는 범위 지정 권한이 있는 정책을 사용합니다. AWS 계정

이 역할은 다음을 수행할 수 있는 권한을 부여합니다.

- Lambda 함수 및 별칭을 읽고, 업데이트하고, 호출합니다.
- Amazon S3 버킷에 있는 개정 파일에 액세스합니다.
- 경보에 대한 정보를 검색합니다. CloudWatch
- Amazon SNS 주제로 정보를 게시합니다.

이 역할은 다음 정책을 사용합니다.

```
*{*
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "lambda:UpdateAlias",
                "lambda:GetAlias",
                "lambda:GetProvisionedConcurrencyConfig",
                "sns:Publish"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::/CodeDeploy/",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
```

```
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:::function:CodeDeployHook_*",
            "Effect": "Allow"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

CodeCatalyst 역할 배포 대상: AWS SAM

CodeCatalyst 워크플로 작업의 경우 기본 CodeCatalystWorkflowDevelopmentRole-***spaceName***서비스 역할을 사용하거나 필요한 권한을 가진 IAM 역할을 생성할 수 있습니다. 이 역할은 작업을 CodeCatalyst 실행해야 하는 권한 범위가 지정된 AWS SAM 정책과 해당 내 AWS CloudFormation 리소스를 사용합니다. AWS 계정

이 역할은 다음 작업을 수행할 수 있는 권한을 부여합니다.

- Lambda 함수를 CodeCatalyst 호출하여 서버리스 및 CLI 애플리케이션 배포를 수행할 수 있습니다. AWS SAM
- 에서 스택과 변경 CodeCatalyst 세트를 생성하고 업데이트할 수 있습니다. AWS CloudFormation

이 역할은 다음 정책을 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "iam:PassRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:TagRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "cloudformation:*",
                "lambda:*",
                "apigateway:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

CodeCatalyst Amazon EC2의 읽기 전용 역할

CodeCatalyst 워크플로 작업의 경우 필요한 권한을 가진 IAM 역할을 생성할 수 있습니다. 이 역할은 사용자의 Amazon EC2 리소스에서 작업을 CodeCatalyst 실행해야 하는 권한 범위가 지정된 정책을 사용합니다. AWS 계정CodeCatalystWorkflowDevelopmentRole-*spaceName*서비스 역할에는 Amazon EC2에 대한 권한이나 Amazon에 대해 설명된 작업은 포함되지 않습니다. CloudWatch

이 역할은 다음을 수행할 수 있는 권한을 부여합니다.

- Amazon EC2 인스턴스의 상태를 가져옵니다.
- Amazon EC2 인스턴스에 대한 CloudWatch 메트릭을 가져옵니다.

이 역할은 다음 정책을 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe",
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe",
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:Describe"
            ],
            "Resource": "resource_ARN"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:Describe",
            "Resource": "resource_ARN"
        }
    ]
}
```

Note

이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.

```
"Resource": "*"
```

CodeCatalyst Amazon ECS의 읽기 전용 역할

CodeCatalyst 워크플로 작업의 경우 필요한 권한을 가진 IAM 역할을 생성할 수 있습니다. 이 역할은 사용자의 Amazon ECS 리소스에서 작업을 CodeCatalyst 실행해야 하는 권한 범위가 지정된 정책을 사용합니다. AWS 계정

이 역할은 다음을 수행할 수 있는 권한을 부여합니다.

- Amazon ECS 태스크 세트를 읽어보십시오.
- CloudWatch 경보에 대한 정보를 검색합니다.

이 역할은 다음 정책을 사용합니다.

```
*{*
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ecs:DescribeServices",
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
       {
            "Action": [
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeRules"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
       {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
```

```
                }
            },
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam:::role/ecsTaskExecutionRole",
                "arn:aws:iam:::role/ECSTaskExecution"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

CodeCatalyst Lambda의 읽기 전용 역할

CodeCatalyst 워크플로 작업의 경우 필요한 권한을 가진 IAM 역할을 생성할 수 있습니다. 이 역할은 사용자의 Lambda 리소스에서 작업을 CodeCatalyst 실행해야 하는 범위 지정 권한이 있는 정책을 사용합니다. AWS 계정

이 역할은 다음에 대한 권한을 부여합니다.

- Lambda 함수 및 별칭을 읽을 수 있습니다.

- Amazon S3 버킷에 있는 개정 파일에 액세스합니다.
- 경보에 대한 정보를 검색합니다. CloudWatch

이 역할은 다음과 같은 정책을 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "lambda:GetAlias",
                "lambda:GetProvisionedConcurrencyConfig"
            ],
            "Resource": "resource_ARN",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::/CodeDeploy/",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            },
            "Effect": "Allow"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고, 사용할 수 있게 되면 리소스 이름으로 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

워크플로 작업을 위한 역할을 수동으로 생성

CodeCatalyst 워크플로 작업은 사용자가 생성한 IAM 역할, 즉 빌드 역할, 배포 역할, 스택 역할을 사용합니다.

다음 단계에 따라 IAM에서 이러한 역할을 생성하세요.

배포 역할을 만들려면

1. 다음과 같이 역할에 대한 정책을 생성합니다.

   a. 로그인 AWS.
   b. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
   c. 탐색 창에서 정책을 선택합니다.
   d. 정책 생성을 선택합니다.
   e. JSON 탭을 선택합니다.
   f. 기존 코드를 삭제합니다.
   g. 다음 코드를 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:Describe*",
        "cloudformation:UpdateStack",
        "cloudformation:CreateChangeSet",
        "cloudformation:DeleteChangeSet",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:SetStackPolicy",
```

```
            "cloudformation:ValidateTemplate",
            "cloudformation:List*",
            "iam:PassRole"
        ],
        "Resource": "*",
        "Effect": "Allow"
}]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고, 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

h. 다음: 태그를 선택합니다.

i. 다음: 검토를 선택합니다.

j. 이름에 다음을 입력합니다.

```
codecatalyst-deploy-policy
```

k. 정책 생성(Create policy)을 선택합니다.

이제 권한 정책을 생성했습니다.

2. 다음과 같이 배포 역할을 생성합니다.

a. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.

b. 사용자 지정 신뢰 정책을 선택합니다.

c. 기존 사용자 지정 신뢰 정책을 삭제합니다.

d. 다음 사용자 지정 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
```

```
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 다음을 선택합니다.

f. 권한 정책에서 해당 확인란을 codecatalyst-deploy-policy 검색하여 선택합니다.

g. 다음을 선택합니다.

h. 역할 이름에 다음을 입력합니다.

```
codecatalyst-deploy-role
```

i. 역할 설명에 다음을 입력합니다.

```
CodeCatalyst deploy role
```

j. 역할 생성을 선택합니다.

이제 신뢰 정책 및 권한 정책을 사용하여 배포 역할을 생성했습니다.

3. 다음과 같이 배포 역할 ARN을 확보하십시오.

a. 탐색 창에서 역할을 선택합니다.

b. 검색 상자에 방금 생성한 역할의 이름을 입력합니다 (codecatalyst-deploy-role).

c. 목록에서 역할을 선택합니다.

역할의 요약 페이지가 나타납니다.

d. 상단에서 ARN 값을 복사합니다.

이제 적절한 권한을 가진 배포 역할을 생성하고 해당 ARN을 획득했습니다.

빌드 역할을 만들려면

1. 다음과 같이 역할에 대한 정책을 생성합니다.

   a. 로그인 AWS.

   b. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

   c. 탐색 창에서 정책을 선택합니다.

   d. 정책 생성을 선택합니다.

   e. JSON 탭을 선택합니다.

   f. 기존 코드를 삭제합니다.

   g. 다음 코드를 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
    "Action": [
        "s3:PutObject",
        "iam:PassRole"
    ],
    "Resource": "*",
    "Effect": "Allow"
}]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고, 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.
>
> ```
> "Resource": "*"
> ```

   h. 다음: 태그를 선택합니다.

   i. 다음: 검토를 선택합니다.

   j. 이름에 다음을 입력합니다.

```
codecatalyst-build-policy
```

k. 정책 생성(Create policy)을 선택합니다.

이제 권한 정책을 생성했습니다.

2. 다음과 같이 빌드 역할을 생성합니다.

a. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.

b. 사용자 지정 신뢰 정책을 선택합니다.

c. 기존 사용자 지정 신뢰 정책을 삭제합니다.

d. 다음 사용자 지정 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 다음을 선택합니다.

f. 권한 정책에서 해당 확인란을 codecatalyst-build-policy 검색하여 선택합니다.

g. 다음을 선택합니다.

h. 역할 이름에 다음을 입력합니다.

```
codecatalyst-build-role
```

i. 역할 설명에 다음을 입력합니다.

```
CodeCatalyst build role
```

j. 역할 생성을 선택합니다.

이제 신뢰 정책 및 권한 정책을 사용하여 빌드 역할을 만들었습니다.

3. 다음과 같이 빌드 역할 ARN을 가져옵니다.

   a. 탐색 창에서 역할을 선택합니다.

   b. 검색 상자에 방금 생성한 역할의 이름을 입력합니다 (`codecatalyst-build-role`).

   c. 목록에서 역할을 선택합니다.

      역할의 요약 페이지가 나타납니다.

   d. 상단에서 ARN 값을 복사합니다.

   이제 적절한 권한이 있는 빌드 역할을 생성하고 해당 ARN을 획득했습니다.

스택 역할을 만들려면

> **Note**
>
> 스택 역할을 생성할 필요는 없지만 보안상의 이유로 그렇게 하는 것이 좋습니다. 스택 역할을 생성하지 않는 경우 이 절차에서 자세히 설명하는 권한 정책을 배포 역할에 추가해야 합니다.

1. 스택을 배포하려는 계정을 AWS 사용하여 로그인합니다.
2. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
3. 탐색 창에서 [Roles] 를 선택한 다음 [Create role] 을 선택합니다.
4. 상단에서 AWS 서비스를 선택합니다.
5. 서비스 목록에서 선택합니다 CloudFormation.
6. 다음: 권한을 선택합니다.
7. 검색 상자에 스택의 리소스에 액세스하는 데 필요한 정책을 추가합니다. 예를 들어 스택에 AWS Lambda 함수가 포함된 경우 Lambda에 대한 액세스 권한을 부여하는 정책을 추가해야 합니다.

> **Tip**
>
> 어떤 정책을 추가해야 할지 확실하지 않은 경우 지금은 해당 정책을 생략할 수 있습니다. 작업을 테스트할 때 적절한 권한이 없는 경우 추가해야 할 권한을 보여주는 오류가 AWS CloudFormation 발생합니다.

8. 다음: 태그를 선택합니다.
9. 다음: 검토를 선택합니다.
10. 역할 이름에 다음을 입력합니다.

```
codecatalyst-stack-role
```

11. 역할 생성을 선택합니다.
12. 스택 역할의 ARN을 가져오려면 다음과 같이 하십시오.
    a. 탐색 창에서 역할을 선택합니다.
    b. 검색 상자에 방금 생성한 역할의 이름 (`codecatalyst-stack-role`) 을 입력합니다.
    c. 목록에서 역할을 선택합니다.
    d. 요약 페이지에서 역할 ARN 값을 복사합니다.

IAM에서 정책 및 역할을 생성하는 AWS CloudFormation 데 사용

AWS CloudFormation 템플릿을 생성하고 사용하여 CodeCatalyst 프로젝트 및 워크플로의 리소스에 액세스하는 데 필요한 정책과 역할을 생성할 수 있습니다. AWS 계정 AWS CloudFormation 리소스를 모델링하고 설정하여 AWS 리소스를 관리하는 데 소요되는 시간을 줄이고 실행 중인 애플리케이션에 더 많은 시간을 할애할 수 있도록 도와주는 AWS서비스입니다. 역할을 여러 AWS 계정개로 생성하려는 경우 템플릿을 만들면 이 작업을 더 빠르게 수행할 수 있습니다.

다음 예제 템플릿은 배포 작업 역할 및 정책을 생성합니다.

```
Parameters:
  CodeCatalystAccountId:
    Type: String
    Description: Account ID from the connections page
  ExternalId:
    Type: String
    Description: External ID from the connections page
```

```
Resources:
  CrossAccountRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              AWS:
                - !Ref CodeCatalystAccountId
            Action:
              - 'sts:AssumeRole'
            Condition:
              StringEquals:
                sts:ExternalId: !Ref ExternalId
      Path: /
      Policies:
        - PolicyName: CodeCatalyst-CloudFormation-action-policy
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - 'cloudformation:CreateStack'
                  - 'cloudformation:DeleteStack'
                  - 'cloudformation:Describe*'
                  - 'cloudformation:UpdateStack'
                  - 'cloudformation:CreateChangeSet'
                  - 'cloudformation:DeleteChangeSet'
                  - 'cloudformation:ExecuteChangeSet'
                  - 'cloudformation:SetStackPolicy'
                  - 'cloudformation:ValidateTemplate'
                  - 'cloudformation:List*'
                  - 'iam:PassRole'
                Resource: '*'
```

웹 애플리케이션 블루프린트의 역할을 수동으로 생성

CodeCatalyst 웹 애플리케이션 블루프린트는 사용자가 생성한 IAM 역할, 즉 CDK용 빌드 역할, 배포 역할, 스택 역할을 사용합니다.

다음 단계에 따라 IAM에서 역할을 생성하세요.

빌드 역할을 만들려면

1. 다음과 같이 역할에 대한 정책을 생성합니다.

   a. 로그인 AWS.

   b. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

   c. 탐색 창에서 정책을 선택합니다.

   d. 정책 생성을 선택하세요.

   e. JSON 탭을 선택합니다.

   f. 기존 코드를 삭제합니다.

   g. 다음 코드를 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:*",
                "ecr:*",
                "ssm:*",
                "s3:*",
                "iam:PassRole",
                "iam:GetRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고, 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.

```
        "Resource": "*"
```

h. 다음: 태그를 선택합니다.

i. 다음: 검토를 선택합니다.

j. 이름에 다음을 입력합니다.

```
codecatalyst-webapp-build-policy
```

k. 정책 생성(Create policy)을 선택합니다.

이제 권한 정책을 생성했습니다.

2. 다음과 같이 빌드 역할을 생성합니다.

a. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.

b. 사용자 지정 신뢰 정책을 선택합니다.

c. 기존 사용자 지정 신뢰 정책을 삭제합니다.

d. 다음 사용자 지정 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 다음을 선택합니다.

f. 권한 정책을 빌드 역할에 연결합니다. 권한 추가 페이지의 권한 정책 섹션에서 해당 확인란을 `codecatalyst-webapp-build-policy` 검색하여 선택합니다.

g. 다음을 선택합니다.

h. 역할 이름에 다음을 입력합니다.

```
codecatalyst-webapp-build-role
```

i. 역할 설명에 다음을 입력합니다.

```
CodeCatalyst Web app build role
```

j. 역할 생성을 선택합니다.

이제 신뢰 정책 및 권한 정책을 사용하여 빌드 역할을 만들었습니다.

3. 다음과 같이 권한 정책을 빌드 역할에 연결합니다.

a. 탐색 창에서 역할을 선택한 다음 `codecatalyst-webapp-build-role` 검색합니다.

b. 세부 정보를 `codecatalyst-webapp-build-role` 표시하도록 선택합니다.

c. [권한] 탭에서 [권한 추가] 를 선택한 다음 [정책 연결] 을 선택합니다.

d. 정책을 검색하고 `codecatalyst-webapp-build-policy` 해당 확인란을 선택한 다음 정책 연결을 선택합니다.

이제 권한 정책을 빌드 역할에 연결했습니다. 이제 빌드 역할에는 권한 정책과 신뢰 정책이라는 두 가지 정책이 있습니다.

4. 다음과 같이 빌드 역할 ARN을 가져옵니다.

a. 탐색 창에서 역할을 선택합니다.

b. 검색 상자에 방금 생성한 역할의 이름을 입력합니다 (`codecatalyst-webapp-build-role`).

c. 목록에서 역할을 선택합니다.

역할의 요약 페이지가 나타납니다.

d. 상단에서 ARN 값을 복사합니다.

이제 적절한 권한이 있는 빌드 역할을 생성하고 해당 ARN을 획득했습니다.

SAM 블루프린트의 역할을 수동으로 생성

CodeCatalyst SAM 블루프린트는 사용자가 생성한 IAM 역할을 사용합니다. 이 역할을 SAM의 빌드 CloudFormation 역할과 배포 역할이라고 합니다.

다음 단계에 따라 IAM에서 역할을 생성하세요.

에 대한 빌드 역할을 만들려면 CloudFormation

1. 다음과 같이 역할에 대한 정책을 생성합니다.

   a. 로그인 AWS.

   b. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

   c. 탐색 창에서 정책을 선택합니다.

   d. 정책 생성을 선택하세요.

   e. JSON 탭을 선택합니다.

   f. 기존 코드를 삭제합니다.

   g. 다음 코드를 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudformation:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고, 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.

```
        "Resource": "*"
```

h. 다음: 태그를 선택합니다.

i. 다음: 검토를 선택합니다.

j. 이름에 다음을 입력합니다.

```
codecatalyst-SAM-build-policy
```

k. 정책 생성(Create policy)을 선택합니다.

이제 권한 정책을 생성했습니다.

2. 다음과 같이 빌드 역할을 생성합니다.

a. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.

b. 사용자 지정 신뢰 정책을 선택합니다.

c. 기존 사용자 지정 신뢰 정책을 삭제합니다.

d. 다음 사용자 지정 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 다음을 선택합니다.

f. 권한 정책을 빌드 역할에 연결합니다. 권한 추가 페이지의 권한 정책 섹션에서 해당 확인란을 `codecatalyst-SAM-build-policy` 검색하여 선택합니다.

g. 다음을 선택합니다.

h. 역할 이름에 다음을 입력합니다.

```
codecatalyst-SAM-build-role
```

i. 역할 설명에 다음을 입력합니다.

```
CodeCatalyst SAM build role
```

j. 역할 생성을 선택합니다.

이제 신뢰 정책 및 권한 정책을 사용하여 빌드 역할을 만들었습니다.

3. 다음과 같이 권한 정책을 빌드 역할에 연결합니다.

a. 탐색 창에서 역할을 선택한 다음 `codecatalyst-SAM-build-role` 검색합니다.

b. 세부 정보를 `codecatalyst-SAM-build-role` 표시하도록 선택합니다.

c. [권한] 탭에서 [권한 추가] 를 선택한 다음 [정책 연결] 을 선택합니다.

d. 정책을 검색하고 `codecatalyst-SAM-build-policy` 해당 확인란을 선택한 다음 정책 연결을 선택합니다.

이제 권한 정책을 빌드 역할에 연결했습니다. 이제 빌드 역할에는 권한 정책과 신뢰 정책이라는 두 가지 정책이 있습니다.

4. 다음과 같이 빌드 역할 ARN을 가져옵니다.

a. 탐색 창에서 역할을 선택합니다.

b. 검색 상자에 방금 생성한 역할의 이름을 입력합니다 (`codecatalyst-SAM-build-role`).

c. 목록에서 역할을 선택합니다.

역할의 요약 페이지가 나타납니다.

d. 상단에서 ARN 값을 복사합니다.

이제 적절한 권한이 있는 빌드 역할을 생성하고 해당 ARN을 획득했습니다.

SAM용 배포 역할을 만들려면

1. 다음과 같이 역할에 대한 정책을 생성합니다.

a. 로그인 AWS.

b. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

c. 탐색 창에서 정책을 선택합니다.

d. 정책 생성을 선택하세요.

e. JSON 탭을 선택합니다.

f. 기존 코드를 삭제합니다.

g. 다음 코드를 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "iam:PassRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:TagRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "cloudformation:*",
                "lambda:*",
                "apigateway:*"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 이 역할을 사용하여 워크플로 작업을 처음 실행할 때는 리소스 정책 설명에 와일드카드를 사용하고, 사용 가능한 상태가 되면 리소스 이름을 지정하여 정책의 범위를 좁히십시오.

```
        "Resource": "*"
```

h. 다음: 태그를 선택합니다.

i. 다음: 검토를 선택합니다.

j. 이름에 다음을 입력합니다.

```
codecatalyst-SAM-deploy-policy
```

k. 정책 생성(Create policy)을 선택합니다.

이제 권한 정책을 생성했습니다.

2. 다음과 같이 빌드 역할을 생성합니다.

a. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.

b. 사용자 지정 신뢰 정책을 선택합니다.

c. 기존 사용자 지정 신뢰 정책을 삭제합니다.

d. 다음 사용자 지정 신뢰 정책을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service":  [
                   "codecatalyst-runner.amazonaws.com",
                   "codecatalyst.amazonaws.com"
                 ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

e. 다음을 선택합니다.

f. 권한 정책을 빌드 역할에 연결합니다. 권한 추가 페이지의 권한 정책 섹션에서 해당 확인란을 `codecatalyst-SAM-deploy-policy` 검색하여 선택합니다.

g. 다음을 선택합니다.

h. 역할 이름에 다음을 입력합니다.

```
codecatalyst-SAM-deploy-role
```

i. 역할 설명에 다음을 입력합니다.

```
CodeCatalyst SAM deploy role
```

j. 역할 생성을 선택합니다.

이제 신뢰 정책 및 권한 정책을 사용하여 빌드 역할을 만들었습니다.

3. 다음과 같이 권한 정책을 빌드 역할에 연결합니다.

a. 탐색 창에서 역할을 선택한 다음 `codecatalyst-SAM-deploy-role` 검색합니다.

b. 세부 정보를 `codecatalyst-SAM-deploy-role` 표시하도록 선택합니다.

c. [권한] 탭에서 [권한 추가] 를 선택한 다음 [정책 연결] 을 선택합니다.

d. 정책을 검색하고 `codecatalyst-SAM-deploy-policy` 해당 확인란을 선택한 다음 정책 연결을 선택합니다.

이제 권한 정책을 빌드 역할에 연결했습니다. 이제 빌드 역할에는 권한 정책과 신뢰 정책이라는 두 가지 정책이 있습니다.

4. 다음과 같이 빌드 역할 ARN을 가져옵니다.

a. 탐색 창에서 역할을 선택합니다.

b. 검색 상자에 방금 생성한 역할의 이름을 입력합니다 (`codecatalyst-SAM-deploy-role`).

c. 목록에서 역할을 선택합니다.

역할의 요약 페이지가 나타납니다.

d. 상단에서 ARN 값을 복사합니다.

이제 적절한 권한이 있는 빌드 역할을 생성하고 해당 ARN을 획득했습니다.

Amazon에 대한 규정 준수 검증 CodeCatalyst

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면AWS 서비스 규정 준수 프로그램의AWS 서비스 범위별, 규정 참조하여 관심 있는 규정 준수 프로그램을 선택하십시오. 일반 정보는 AWS 규정 준수 프로그램AWS 보증 프로그램 규정AWS 참조하십시오.

를 사용하여 AWS Artifact타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- 보안 및 규정 준수 퀵 스타트 가이드 - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 HIPAA 적격 애플리케이션을 만드는 AWS 데 사용할 수 있는 방법을 설명합니다.

  > **Note**
  >
  > 모든 AWS 서비스 사람이 HIPAA 자격을 갖춘 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하십시오.

- AWS 규정 준수 리소스AWS — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (미국 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (PCI), 국제 표준화기구 (ISO) 등) 에서 보안 제어에 대한 지침을 매핑합니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 — 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하십시오.
- Amazon GuardDuty — 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정모니터링하여 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수

프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.

- AWS Audit Manager— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

아마존의 레질리언스 CodeCatalyst

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크를 통해 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 정보는 AWS 글로벌 인프라를 참조하세요. 어떤 CodeCatalyst 데이터에 AWS 리전 복제되는지에 대한 자세한 내용은 을 참조하십시오. 아마존에서의 데이터 보호 CodeCatalyst

아마존의 인프라 보안 CodeCatalyst

CodeCatalyst Amazon은 관리형 서비스로서 AWS 글로벌 네트워크 보안의 보호를 받습니다. AWS 보안 서비스와 AWS의 인프라 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안에 대한 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS게시된 API 호출을 사용하여 네트워크를 CodeCatalyst 통해 액세스합니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

Amazon의 구성 및 취약성 분석 CodeCatalyst

구성 및 IT 제어는 AWS와 고객 간의 공동 책임입니다. 자세한 내용은 AWS 공동 책임 모델을 참조하세요.

Amazon의 사용자 데이터 및 개인정보 보호 CodeCatalyst

CodeCatalyst Amazon은 사용자의 개인 정보를 중요하게 생각하며 사용자 정보의 보안을 최우선으로 생각합니다. AWS개인정보 취급방침에서 아마존이 귀하의 정보를 처리하는 방법에 대해 자세히 알아볼 수 있습니다.

데이터를 요청하고 보려면 에서 데이터 요청을 참조하십시오AWS 일반 참조.

AWS빌더 ID 프로필 삭제

프로필 삭제는 되돌릴 수 없는 영구적인 조치입니다. 삭제를 선택하면 바로 삭제 프로세스가 시작됩니다. CodeCatalystAmazon은 프로필 및 모든 관련 개인 정보를 삭제하기 시작합니다. 이 프로세스를 완료하는 데 최대 90일이 걸릴 수 있습니다.

프로필이 삭제되면 Amazon에 있는 데이터에 액세스하거나 데이터를 복구할 수 없습니다 CodeCatalyst. 여기에는 개인용 액세스 토큰, 역할, 사용자 멤버십 및 귀하가 유일한 회원인 모든 Amazon CodeCatalyst 스페이스가 포함됩니다. 더 이상 Amazon에 로그인할 수 없습니다 CodeCatalyst.

AWS빌더 ID 프로필을 삭제하는 방법에 대한 자세한 내용은 에서 AWS빌더 ID 삭제를 참조하십시오 AWS 일반 참조.

Amazon에서의 워크플로 작업에 대한 모범 사례 CodeCatalyst

에서 워크플로를 개발할 때 고려해야 할 보안 모범 사례가 많이 CodeCatalyst 있습니다. 다음은 일반적인 지침이며 완전한 보안 솔루션을 의미하지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용하세요.

주제

- 민감한 정보
- 라이선스 조건
- 신뢰할 수 없는 코드
- GitHub 조치

민감한 정보

YAML에 민감한 정보를 포함시키지 마세요. YAML에 자격 증명, 키 또는 토큰을 내장하는 대신 시크릿을 사용하는 것이 좋습니다. CodeCatalyst 시크릿은 YAML 내에서 민감한 정보를 저장하고 참조할 수 있는 간편한 방법을 제공합니다.

라이선스 조건

사용하기로 선택한 작업의 라이선스 조건에 주의를 기울이세요.

신뢰할 수 없는 코드

액션은 일반적으로 프로젝트, 스페이스 또는 더 넓은 커뮤니티에서 공유할 수 있는 독립된 단일 목적 모듈입니다. 다른 사람의 코드를 사용하면 편리함과 효율성이 크게 향상될 수 있지만 새로운 위협 요소가 발생할 수도 있습니다. 다음 섹션을 검토하여 CI/CD 워크플로의 보안을 유지하기 위한 모범 사례를 따르고 있는지 확인하십시오.

GitHub 조치

GitHub 액션은 커뮤니티에서 구축하고 관리하는 오픈 소스입니다. 우리는 공동 책임 모델을 따르며 GitHub Actions 소스 코드를 귀하가 책임져야 하는 고객 데이터로 간주합니다. GitHub 작업에는 비밀, 리포지토리 토큰, 소스 코드, 계정 링크 및 컴퓨팅 시간에 대한 액세스 권한을 부여할 수 있습니다. 실행하려는 GitHub 작업의 신뢰성과 보안에 확신이 있는지 확인하세요.

작업에 대한 보다 구체적인 지침 및 보안 모범 사례: GitHub

- 보안 강화
- 자체 요청 방지
- 신뢰할 수 없는 입력
- 빌딩 블록을 신뢰하는 방법

CodeCatalyst 신뢰 모델 이해

Amazon CodeCatalyst 신뢰 모델을 사용하면 연결된 CodeCatalyst 환경에서 서비스 역할을 맡을 수 AWS 계정있습니다. 모델은 IAM 역할, CodeCatalyst 서비스 주체 및 공간을 연결합니다. CodeCatalyst 신뢰 정책은 `aws:SourceArn` 조건 키를 사용하여 조건 키에 지정된 CodeCatalyst 공간에 권한을 부여합니다. 이 조건 키에 대한 자세한 내용은 IAM 사용 설명서의 SourceArn aws:를 참조하십시오.

신뢰 정책은 신뢰하는 보안 주체가 역할을 맡도록 정의하는 JSON 정책 문서입니다. 역할 신뢰 정책은 IAM의 역할에 연결된 필수 리소스 기반 정책입니다. 자세한 내용은 IAM 사용 설명서의 용어 및 개념을 참조하십시오. 의 서비스 주체에 대한 자세한 내용은 을 참조하십시오. CodeCatalyst 서비스 주체: CodeCatalyst

다음 신뢰 정책에서는 `Principal` 요소에 나열된 서비스 주체에 리소스 기반 정책의 권한이 부여되며, `Condition` 차단은 범위가 축소된 리소스에 대한 액세스를 제한하는 데 사용됩니다.

```
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
             "Principal": {
                "Service": [
                    "codecatalyst-runner.amazonaws.com",
                    "codecatalyst.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:codecatalyst:::space/spaceId/project/*"
                }
            }
        }
    ]
```

신뢰 정책에서 CodeCatalyst 서비스 주체는 스페이스 ID에 대한 Amazon 리소스 이름 (ARN) 이 포함된 `aws:SourceArn` 조건 키를 통해 액세스 권한을 부여받습니다. CodeCatalyst ARN은 다음 형식을 사용합니다.

```
arn:aws:codecatalyst:::space/spaceId/project/*
```

> **Important**
>
> 스페이스 ID는 조건 키 (예`aws:SourceArn`:) 에만 사용하십시오. IAM 정책문의 스페이스 ID를 리소스 ARN으로 사용하지 마십시오.

정책에서 권한의 범위를 최대한 좁히는 것이 가장 좋습니다.

- `aws:SourceArn`조건 키에 와일드카드 (*) 를 사용하여 `project/*` 스페이스의 모든 프로젝트를 지정할 수 있습니다.
- 를 사용하여 스페이스의 특정 프로젝트에 대한 `aws:SourceArn` 조건 키에 리소스 수준 권한을 지정할 수 있습니다. `project/projectId`

서비스 주체: CodeCatalyst

리소스 기반 JSON 정책의 `Principal` 요소를 사용하여 리소스에 대한 액세스를 허용하거나 거부할 보안 주체를 지정합니다. 신뢰 정책에서 지정할 수 있는 보안 주체에는 사용자, 역할, 계정 및 서비스가 포함됩니다. ID 기반 정책에서는 `Principal` 요소를 사용할 수 없습니다. 마찬가지로 그룹은 인증이 아니라 권한과 관련이 있고 보안 주체는 인증된 IAM 개체이기 때문에 사용자 그룹을 정책 (예: 리소스 기반 정책) 의 보안 주체로 식별할 수 없습니다.

신뢰 정책에서는 리소스 기반 정책의 `Principal` 요소나 보안 주체를 지원하는 조건 키를 지정할 AWS 서비스 수 있습니다. 서비스 주체는 서비스에 의해 정의됩니다. 다음에 대해 정의된 서비스 주체는 다음과 같습니다. CodeCatalyst

- codecatalyst.amazonaws.com - 이 서비스 주체는 액세스 권한을 부여하는 역할에 사용됩니다. CodeCatalyst AWS
- codecatalyst-runner.amazonaws.com - 이 서비스 주체는 워크플로 배포 시 리소스에 대한 액세스 권한을 부여하는 역할에 사용됩니다. CodeCatalyst AWS CodeCatalyst

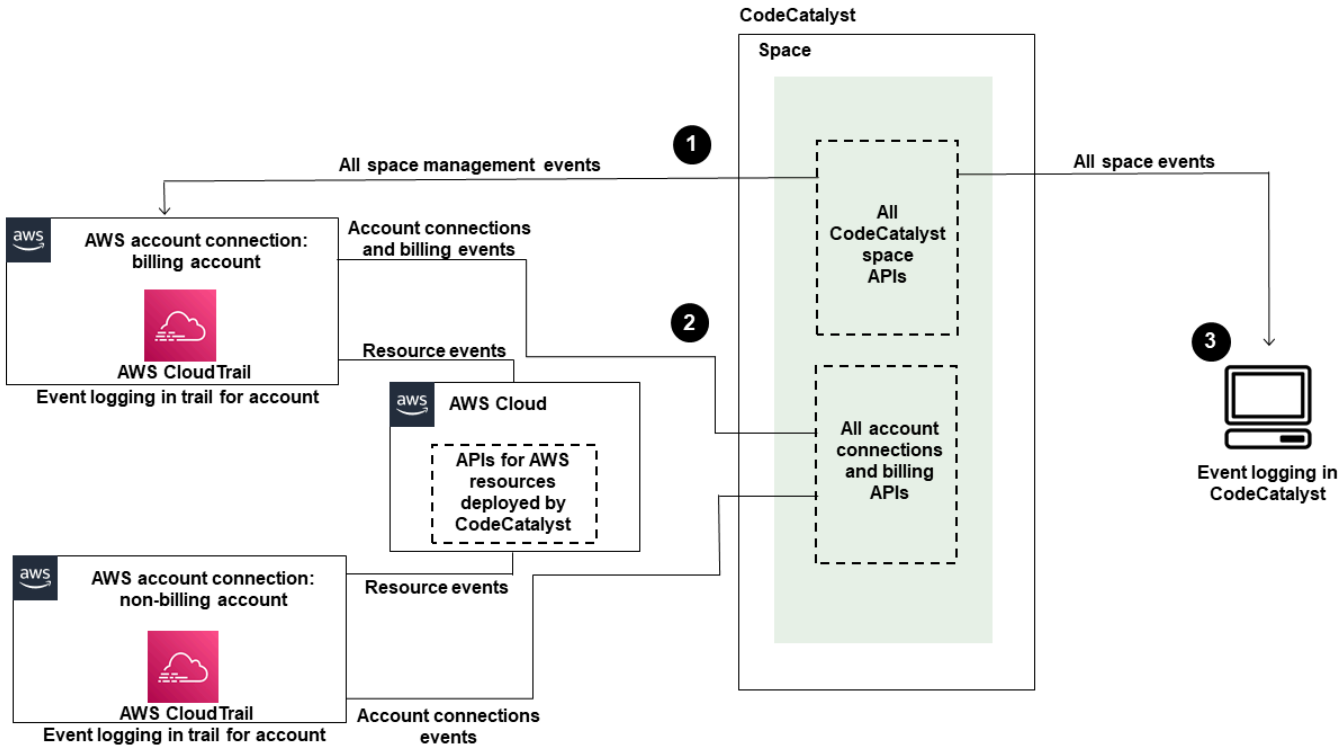
자세한 내용은 IAM 사용 설명서의 AWS JSON 정책 요소: 보안 주체를 참조하세요.

로깅을 사용하여 이벤트 및 API 통화 모니터링

CodeCatalystAmazon에서는 공간에 대한 관리 이벤트가 공간 결제 계정에 의해 AWS CloudTrail 수집되고 해당 트레일에 기록됩니다. CloudTrail 로깅은 CodeCatalyst 이벤트 로깅을 관리하는 기본 방법이고, 보조 방법은 이벤트 로그인을 보는 CodeCatalyst 것입니다.

계정의 이벤트는 에 대해 설정된 트레일 및 지정된 버킷으로 AWS 계정기록됩니다.

다음 다이어그램은 공간에 대한 모든 관리 이벤트가 결제 계정에 CloudTrail 로그인되고 계정 연결/청구 이벤트 및 AWS 리소스 이벤트는 해당 계정에 CloudTrail 로그인되는 방식을 보여줍니다.



다이어그램은 다음 단계들을 보여줍니다.

1. 스페이스가 생성되면 AWS 계정 이 스페이스에 연결되고 결제 계정으로 지정됩니다. 사용된 트레일은 스페이스 이벤트가 기록되는 결제 CloudTrail 계정용으로 생성된 트레일입니다. CloudTrail 스페이스에 의해 또는 CodeCatalyst 스페이스를 대신하여 이루어진 API 호출 및 관련 이벤트를 캡처하고 지정한 S3 버킷에 로그 파일을 전송합니다. 결제 계정이 다른 계정으로 변경되면 AWS 해당 계정의 트레일 및 버킷에 스페이스 이벤트가 기록됩니다. 로그인한 CodeCatalyst 관리 이벤트에 대한 자세한 내용은 CloudTrail 을 참조하십시오CodeCatalyst 정보: CloudTrail.
2. 결제 계정을 포함하여 스페이스에 연결된 다른 계정은 계정 연결 및 결제 이벤트에 대한 일부 이벤트를 기록합니다. CodeCatalyst 해당 계정에 배포된 AWS 리소스에 대한 계정 이벤트를 생성하는 워크플로도 의 트레일 및 버킷에 기록됩니다. AWS 계정 CloudTrail 스페이스에 의해 또는 CodeCatalyst 스페이스를 대신하여 이루어진 API 호출 및 관련 이벤트를 캡처하고 지정한 S3 버킷에 로그 파일을 전송합니다. 로깅하는 CodeCatalyst 관리 이벤트에 대한 자세한 내용은 CloudTrail 을 참조하십시오이벤트 로깅을 사용하여 로깅된 이벤트에 액세스.
3. 또한 를 사용하여 list-event-logs명령을 사용하여 스페이스의 특정 시간 내에 스페이스에서 발생한 CodeCatalyst 작업을 모니터링할 수 AWS CLI있습니다. 자세한 내용은 Amazon CodeCatalyst API 참조 안내서를 참조하십시오. 스페이스 내 CodeCatalyst 작업에 대한 이벤트 목록을 호출하려면 스

페이스 관리자 역할이 있어야 합니다. 자세한 내용은 이벤트 로깅을 사용하여 로깅된 이벤트에 액세스 단원을 참조하십시오.

> **Note**
>
> `ListEventLogs`지정된 공간에서 지난 30일 동안의 이벤트를 보장합니다. 또한 이벤트 기록을 보거나 지난 90일 동안의 이벤트 기록을 생성 및 유지 관리하는 트레일을 생성하여 CodeCatalyst AWS CloudTrail 콘솔에서 지난 90일 동안의 관리 이벤트 목록을 보고 검색할 수 있습니다. 자세한 내용은 CloudTrail 이벤트 기록 작업 및 CloudTrail 트레일 작업을 참조하십시오.

> **Note**
>
> AWS CodeCatalyst 워크플로의 연결된 계정에 배포된 리소스는 CodeCatalyst 공간 CloudTrail 로깅의 일부로 기록되지 않습니다. 예를 들어 CodeCatalyst 리소스에는 공간 또는 프로젝트가 포함됩니다. AWS 리소스에는 Amazon ECS 서비스 또는 Lambda 함수가 포함됩니다. 리소스가 AWS 계정 배포되는 각 위치에 대해 개별적으로 CloudTrail 로깅을 구성해야 합니다.

다음은 이벤트 모니터링에 사용할 수 있는 한 가지 CodeCatalyst 흐름입니다.

Mary Major는 스페이스의 CodeCatalyst 스페이스 관리자이며 로그인한 스페이스의 스페이스 수준 및 프로젝트 수준 리소스에 CodeCatalyst 대한 모든 관리 이벤트를 조회합니다. CloudTrail CodeCatalyst 정보: CloudTrail예를 들어 로그인한 이벤트를 참조하십시오. CloudTrail

Dev Environments와 같이 생성된 리소스의 경우 Mary는 스페이스의 결제 계정에서 이벤트 기록을 보고 프로젝트 구성원이 개발 환경을 만든 이벤트를 조사합니다. CodeCatalyst CodeCatalyst 이벤트는 개발 환경을 만든 사용자의 AWS 빌더 ID에 대한 IAM ID 저장소 ID 유형과 자격 증명을 제공합니다. 서버리스 배포용 Lambda 함수와 같이 워크플로에 CodeCatalyst 의해 배포될 AWS 때 생성된 리소스의 경우 소유자는 워크플로 배포 작업에 대한 AWS 계정 별도의 (연결된 CodeCatalyst 계정이기도 함) 과 연결된 트레일의 이벤트 기록을 볼 수 있습니다. AWS 계정

더 자세히 조사하기 위해 Mary는 의 list-event-logs명령을 사용하여 CodeCatalyst APIs 스페이스에 있는 모든 사람의 이벤트를 볼 수도 있습니다. AWS CLI

주제

- AWS 계정AWS CloudTrail 로깅을 사용한 API 호출 모니터링

- 이벤트 로깅을 사용하여 로깅된 이벤트에 액세스

AWS 계정AWS CloudTrail 로깅을 사용한 API 호출 모니터링

CodeCatalyst Amazon은 사용자 AWS CloudTrail, 역할 또는 담당자가 수행한 작업에 대한 기록을 제공하는 서비스와 통합되어 AWS 서비스있습니다. CloudTrail Connected AWS 계정 as 이벤트를 대신하여 이루어진 CodeCatalyst API 호출을 캡처합니다. 트레일을 생성하면 에 대한 이벤트를 포함하여 S3 버킷에 CloudTrail 이벤트를 지속적으로 전송할 수 CodeCatalyst 있습니다. 트레일을 구성하지 않아도 CloudTrail 콘솔의 이벤트 기록에서 가장 최근 이벤트를 계속 볼 수 있습니다.

CodeCatalyst 다음 작업을 CloudTrail 로그 파일에 이벤트로 기록할 수 있습니다.

- CodeCatalyst 스페이스에 대한 관리 이벤트는 스페이스에 AWS 계정 대해 지정된 결제 계정에 기록됩니다. 자세한 정보는 CodeCatalyst 스페이스 이벤트을 참조하세요.

  > **Note**
  >
  > CodeCatalyst 공간에 대한 데이터 이벤트는 에 설명된 대로 CLI를 사용하여 액세스할 수 있습니다. 이벤트 로깅을 사용하여 로깅된 이벤트에 액세스

- 커넥티드 환경에서 발생하는 CodeCatalyst 워크플로 작업에 사용되는 리소스의 AWS 계정 이벤트는 해당 이벤트로 로깅됩니다. AWS 계정자세한 정보는 CodeCatalyst 계정 연결 및 결제 이벤트을 참조하세요.

> **Important**
>
> 스페이스에 여러 계정을 연결할 수 있지만 CodeCatalyst 스페이스 및 프로젝트의 이벤트 CloudTrail 로깅은 결제 계정에만 적용됩니다.

스페이스 결제 계정은 AWS 프리 티어 이외의 CodeCatalyst 리소스에 대해 요금이 청구되는 사용자 AWS 계정 계정입니다. 스페이스에 여러 계정을 연결할 수 있지만, 하나의 계정만 지정된 결제 계정이 될 수 있습니다. 공간에 대한 결제 계정 또는 추가 연결 계정에는 워크플로에서 Amazon ECS 클러스터 또는 S3 버킷과 같은 AWS 리소스 및 인프라를 배포하는 데 사용되는 IAM 역할이 있을 수 있습니다. CodeCatalyst 워크플로 YAML을 사용하여 배포한 대상을 식별할 수 있습니다 AWS 계정 .

> **Note**
>
> AWS CodeCatalyst 워크플로의 연결된 계정에 배포된 리소스는 CodeCatalyst 공간 CloudTrail 로깅의 일부로 로깅되지 않습니다. 예를 들어 CodeCatalyst 리소스에는 공간 또는 프로젝트가 포함됩니다. AWS 리소스에는 Amazon ECS 서비스 또는 Lambda 함수가 포함됩니다. CloudTrail 리소스가 배포되는 각 AWS 계정 위치에 대해 로깅을 별도로 구성해야 합니다.

CodeCatalyst 연결된 계정 로그인에는 다음 고려 사항이 포함됩니다.

- CloudTrail 이벤트 액세스는 로그인이 아닌 연결된 계정의 IAM을 통해 관리됩니다. CodeCatalyst
- GitHub 리포지토리 연결과 같은 타사 연결을 사용하면 타사 리소스 이름이 CloudTrail 로그에 기록됩니다.

> **Note**
>
> CloudTrail CodeCatalyst 이벤트 로깅은 스페이스 수준에서 이루어지며 프로젝트 경계별로 이벤트를 분리하지 않습니다.

에 대한 CloudTrail 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

> **Note**
>
> 이 섹션에서는 로그인된 CodeCatalyst 스페이스에 로그인되어 있고 연결된 모든 이벤트에 대한 CloudTrail 로깅에 대해 설명합니다 CodeCatalyst. AWS 계정 또한 CodeCatalyst 스페이스에 기록된 모든 이벤트를 검토하려면 AWS CLI 및 aws codecatalyst list-event-logs 명령을 사용할 수도 있습니다. 자세한 정보는 이벤트 로깅을 사용하여 로깅된 이벤트에 액세스을 참조하세요.

CodeCatalyst 스페이스 이벤트

스페이스 수준 및 프로젝트 수준 리소스를 관리하기 CodeCatalyst 위한 작업은 스페이스의 결제 계정에 로그인됩니다. CodeCatalyst 스페이스 CloudTrail 로깅의 경우 이벤트는 다음 고려 사항에 따라 기록됩니다.

- CloudTrail 이벤트는 전체 공간에 적용되며 단일 프로젝트로 범위가 지정되지 않습니다.

- CodeCatalyst 스페이스에 연결하면 계정 연결을 위해 기록 가능한 이벤트가 해당 스페이스에 기록됩니다. AWS 계정 AWS 계정이 연결을 활성화한 후에는 비활성화할 수 없습니다.
- AWS 계정 스페이스에 연결하고 CodeCatalyst 스페이스를 해당 스페이스의 결제 계정으로 지정하면 해당 스페이스에 이벤트가 로그인됩니다. AWS 계정이 연결을 활성화한 후에는 비활성화할 수 없습니다.

  스페이스 수준 및 프로젝트 수준 리소스에 대한 이벤트는 결제 계정에만 기록됩니다. CloudTrail 대상 계정을 변경하려면 에서 결제 계정을 업데이트하세요. CodeCatalyst 다음 월별 결제 주기가 시작되면 의 새 결제 계정에 변경 내용이 적용됩니다 CodeCatalyst. 그런 다음 CloudTrail 대상 계정이 업데이트됩니다.

다음은 스페이스 수준 및 프로젝트 수준 리소스를 관리하기 CodeCatalyst 위한 작업과 관련된 이벤트의 예입니다. AWS 다음 API는 SDK 및 CLI를 통해 릴리스됩니다. 이벤트는 AWS 계정 지정된 공간에 대한 결제 계정으로 로그인됩니다. CodeCatalyst

- CreateDevEnvironment
- CreateProject
- DeleteDevEnvironment
- GetDevEnvironment
- GetProject
- GetSpace
- GetSubscription
- ListDevEnvironments
- ListDevEnvironmentSessions
- ListEventLogs
- ListProjects
- ListSourceRepositories
- StartDevEnvironment
- StartDevEnvironmentSession
- StopDevEnvironment
- StopDevEnvironmentSession
- UpdateDevEnvironment

CodeCatalyst 계정 연결 및 결제 이벤트

다음은 계정 연결 또는 청구 관련 작업과 관련된 이벤트의 예입니다. AWS CodeCatalyst

- `AcceptConnection`
- `AssociateIAMRoletoConnection`
- `DeleteConnection`
- `DissassociateIAMRolefromConnection`
- `GetBillingAuthorization`
- `GetConnection`
- `GetPendingConnection`
- `ListConnections`
- `ListIAMRolesforConnection`
- `PutBillingAuthorization`
- `RejectConnection`

CodeCatalyst 정보: CloudTrail

CloudTrail 해당 계정을 만들 AWS 계정 때 활성화됩니다. AWS 계정 CodeCatalyst 스페이스에 연결하면 CloudTrail 로그인한 해당 스페이스에 대한 이벤트가 해당 AWS 계정에 AWS 계정 로그인됩니다. 로그인 가능한 이벤트는 해당 계정에서 CodeCatalyst 기록 가능한 다른 CloudTrail 이벤트와 함께 연결된 계정의 CloudTrail 로그와 CloudTrail 콘솔의 이벤트 기록에 AWS 이벤트로 기록됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 신원 정보를 이용하면 다음을 쉽게 알아볼 수 있습니다.

- 빌더 ID를 가진 사용자가 요청했는지 여부. AWS
- 요청이 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명으로 이루어졌는지 여부.
- 역할 또는 연동 사용자를 위한 임시 보안 인증으로 요청을 생성하였는지.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail 사용자 ID 요소를 참조하십시오.

이벤트 액세스 CloudTrail

내 CodeCatalyst 활동에 대한 이벤트를 AWS 계정포함하여 내 이벤트의 지속적인 기록을 보려면 AWS 계정트레일을 만드십시오. 트레일을 사용하면 CloudTrail S3 버킷에 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 트레일은 AWS 파티션에 있는 모든 지역의 이벤트를 기록하고 지정한 S3 버킷으로 로그 파일을 전달합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 이에 따라 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하십시오.

- 추적 생성 개요
- CloudTrail 지원되는 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 지역에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

트레일은 지정한 S3 버킷에 이벤트를 로그 파일로 전송할 수 있는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함되어 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜 및 시간, 요청 매개 변수 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 공개 API 호출의 정렬된 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

예제: 의 CodeCatalyst 계정 연결 이벤트 AWS

다음 예제는 `ListConnections` 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다. 스페이스에 연결된 계정의 경우, `ListConnections` 는 AWS 계정 해당 스페이스에 CodeCatalyst 대한 모든 계정 연결을 보는 데 사용됩니다. AWS 계정이벤트는 AWS 계정 지정된 로그인으로 로그인되며 의 값은 작업에 사용된 역할의 Amazon 리소스 이름 (ARN) 이 됩니다. `accountId` `arn`

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "role-ARN",
        "accountId": "account-ID",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAI44QH8DHBEXAMPLE",
                "arn": "role-ARN",
```

```
                "accountId": "account-ID",
                "userName": "user-name"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-09-06T15:04:31Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-09-06T15:08:43Z",
    "eventSource": "account-ID",
    "eventName": "ListConnections",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.168.0.1",
    "userAgent": "aws-cli/1.18.147 Python/2.7.18 Linux/5.4.207-126.363.amzn2int.x86_64
 botocore/1.18.6",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 ",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 ",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "account-ID",
    "eventCategory": "Management"
}
```


예제: CodeCatalyst 프로젝트 리소스 이벤트: AWS

다음 예제는 `CreateDevEnvironment` 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다. 스페이스에 연결되고 스페이스에 지정된 결제 계정이 스페이스의 프로젝트 수준 이벤트 (예: 개발 환경 생성)에 사용됩니다. AWS 계정

`accountId`필드 `userIdentity` 아래에는 모든 AWS 빌더 ID ID의 자격 증명 풀을 호스팅하는 IAM Identity Center 계정 ID (432677196278) 가 있습니다. 이 계정 ID에는 이벤트 CodeCatalyst 사용자에 대한 다음 정보가 들어 있습니다.

- `type`필드는 요청의 IAM 엔티티 유형을 나타냅니다. 공간 및 프로젝트 리소스 CodeCatalyst 이벤트의 경우 이 값은 입니다`IdentityCenterUser`. 이 `accountId` 필드는 자격 증명을 얻는 데 사용된 엔티티를 소유한 계정을 지정합니다.

- 이 `userId` 필드에는 사용자의 AWS 빌더 ID 식별자가 포함됩니다.
- 이 `identityStoreArn` 필드에는 ID 저장소 계정 및 사용자에 대한 역할 ARN이 포함됩니다.

이 `recipientAccountId` 필드에는 공간 결제 계정의 계정 ID가 포함되며, 예제 값은 111122223333 입니다.

자세한 내용은 CloudTrail 사용자 ID 요소를 참조하십시오.


```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "IdentityCenterUser",
  "accountId": "432677196278",
  "onBehalfOf": {
   "userId": "user-ID",
   "identityStoreArn": "arn:aws:identitystore::432677196278:identitystore/d-9067642ac7"
  },
  "credentialId": "ABCDefGhiJKLMn11Lmn_1AbCDEFgHijk-AaBCdEFGHIjKLmnOPqrs11abEXAMPLE"
 },
 "eventTime": "2023-05-18T17:10:50Z",
 "eventSource": "codecatalyst.amazonaws.com",
 "eventName": "CreateDevEnvironment",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "192.168.0.1",
 "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101
Firefox/102.0",
 "requestParameters": {
  "spaceName": "MySpace",
  "projectName": "MyProject",
  "ides": [{
   "runtime": "public.ecr.aws/q6e8p2q0/cloud9-ide-runtime:2.5.1",
   "name": "Cloud9"
  }],
  "instanceType": "dev.standard1.small",
  "inactivityTimeoutMinutes": 15,
  "persistentStorage": {
   "sizeInGiB": 16
  }
 },
 "responseElements": {
  "spaceName": "MySpace",
  "projectName": "MyProject",
```

```
  "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 "
 },
 "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
 "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
 "readOnly": false,
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "111122223333",
 "sharedEventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
 "eventCategory": "Management"
}
```

> **Note**
>
> 경우에 따라 사용자 에이전트가 알려지지 않을 수 있습니다. 이 경우, CodeCatalyst 는 CloudTrail 이벤트의 Unknown `userAgent` 필드에 값을 제공합니다.

이벤트 트레일 CodeCatalyst 쿼리

Amazon Athena의 쿼리 테이블을 사용하여 CloudTrail 로그에 대한 쿼리를 생성하고 관리할 수 있습니다. 쿼리 생성에 대한 자세한 내용은 Amazon Athena AWS CloudTrail 사용 설명서의 로그 쿼리를 참조하십시오.

이벤트 로깅을 사용하여 로깅된 이벤트에 액세스

사용자가 Amazon에서 작업을 CodeCatalyst 수행하면 이러한 작업이 이벤트로 기록됩니다. 를 AWS CLI 사용하여 특정 기간의 공간 내 이벤트 로그를 볼 수 있습니다. 이러한 이벤트를 보고 작업 날짜 및 시간, 작업을 수행한 사용자 이름, 사용자가 요청한 IP 주소 등 스페이스에서 수행된 작업을 검토할 수 있습니다.

> **Note**
>
> CodeCatalyst 스페이스에 대한 관리 이벤트는 연결된 결제 계정에 CloudTrail 로그인됩니다. 로그인한 CodeCatalyst 관리 이벤트에 대한 자세한 내용은 CloudTrail 을 참조하십시오CodeCatalyst 정보: CloudTrail.

스페이스의 이벤트 로그를 보려면 프로파일을 AWS CLI 사용하여 를 설치하고 구성해야 하며 스페이스에 대한 CodeCatalyst 스페이스 관리자 역할이 있어야 합니다. 자세한 내용은 AWS CLI와 함께 사용하기 위한 설정 CodeCatalyst 및 스페이스 관리자 역할 섹션을 참조하세요.

> **Note**
>
> 연결된 계정을 대신하여 발생하는 이벤트에 대한 로깅을 보거나 연결된 AWS 계정결제 CodeCatalyst 계정에서 스페이스 또는 프로젝트 리소스에 대한 이벤트 로깅을 보려면 사용할 수 있습니다 AWS CloudTrail. 자세한 정보는 AWS 계정AWS CloudTrail 로깅을 사용한 API 호출 모니터링을 참조하세요.

1. 터미널 또는 명령줄을 열고 다음을 지정하여 aws codecatalyst list-event-logs 명령을 실행합니다.

   - **`--space-name`**옵션이 있는 스페이스의 이름.
   - RFC 3339에 지정된 협정 세계시 (UTC) 타임스탬프 형식으로 이벤트 검토를 시작하려는 날짜 및 시간 (옵션 포함) **`--start-time`**
   - 이벤트 검토를 중단하려는 날짜 및 시간 (RFC 3339에 지정된 UTC) 타임스탬프 형식 (옵션 포함) **`--end-time`**
   - (선택 사항) 단일 응답에서 반환되는 최대 결과 수 (옵션 포함) **`--max-results`** 결과 수가 지정한 수보다 큰 경우 응답에는 다음 결과를 반환하는 데 사용할 수 있는 `nextToken` 요소가 포함됩니다.
   - (선택 사항) **`--event-name`** 옵션을 사용하여 결과를 반환하려는 특정 이벤트 유형으로 제한합니다.

   이 예제는 *ExampleCorp2022-11-30## 2022-12-01* 기간까지 이름이 지정된 공간에 기록된 이벤트를 반환하며, 응답에는 최대 *2##* 이벤트가 반환됩니다.

   ```
   aws codecatalyst list-event-logs --space-name ExampleCorp --start-time 2022-11-30
    --end-time 2022-12-01 --event-name list-event-logs --max-results 2
   ```

2. 이 기간에 이벤트가 발생한 경우 명령은 다음과 비슷한 결과를 반환합니다.

   ```
   {
       "nextToken": "EXAMPLE",
       "items": [
           {
               "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
   ```

```
            "eventName": "listEventLogs",
            "eventType": "AwsApiCall",
            "eventCategory": "MANAGEMENT",
            "eventSource": "manage",
            "eventTime": "2022-12-01T22:47:24.605000+00:00",
            "operationType": "READONLY",
            "userIdentity": {
                "userType": "USER",
                "principalId": "a1b2c3d4e5-678fgh90-1a2b-3c4d-e5f6-EXAMPLE11111"
                "userName": "MaryMajor"
            },
            "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "requestPayload": {
                "contentType": "application/json",
                "data": "{\"spaceName\":\"ExampleCorp\",\"startTime\":
\"2022-12-01T00:00:00Z\",\"endTime\":\"2022-12-10T00:00:00Z\",\"maxResults\":
\"2\"}"
            },
            "sourceIpAddress": "127.0.0.1",
            "userAgent": "aws-cli/2.9.0 Python/3.9.11 Darwin/21.3.0 exe/x86_64
 prompt/off command/codecatalyst.list-event-logs"
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "eventName": "createProject",
            "eventType": "AwsApiCall",
            "eventCategory": "MANAGEMENT",
            "eventSource": "manage",
            "eventTime": "2022-12-01T09:15:32.068000+00:00",
            "operationType": "MUTATION",
            "userIdentity": {
                "userType": "USER",
                "principalId": "a1b2c3d4e5-678fgh90-1a2b-3c4d-e5f6-EXAMPLE11111",
                "userName": "MaryMajor"
            },
            "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
            "requestPayload": {
                "contentType": "application/json",
                "data": "{\"spaceName\":\"ExampleCorp\",\"name\":\"MyFirstProject
\",\"displayName\":\"MyFirstProject\"}"
            },
            "responsePayload": {
                "contentType": "application/json",
```

```
                "data": "{\"spaceName\":\"ExampleCorp\",\"name\":\"MyFirstProject
\",\"displayName\":\"MyFirstProject\",\"id\":\"a1b2c3d4-5678-90ab-cdef-
EXAMPLE4444\"}"
            },
            "sourceIpAddress": "192.0.2.23",
            "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:102.0)
 Gecko/20100101 Firefox/102.0"
        }
    ]
}
```


3. --next-token옵션과 반환된 토큰의 값을 사용하여 list-event-logs 명령을 다시 실행하여 요청과 일치하는 다음 로그 이벤트 세트를 검색하십시오.

ID, 권한 및 액세스에 대한 할당량 CodeCatalyst

다음 표에는 Amazon의 ID, 권한 및 액세스에 대한 할당량 및 한도가 설명되어 있습니다. CodeCatalyst CodeCatalystAmazon의 할당량에 대한 자세한 내용은 을 참조하십시오. 에 대한 할당량 CodeCatalyst

Resource	정보
별칭: CodeCatalyst	허용되는 모든 문자 조합은 길이가 3~100자 사이이며 문자로 시작해야 합니다. 유효한 문자: A-Z, a-z, 0-9. 별칭은 다음을 수행할 수 없습니다. • 3자 미만 포함 • 공백 또는 다음 문자를 포함할 수 없음: ? ^ * [\ ~ :
사용자가 하루에 보내는 최대 초대장 수	500
이메일 주소로 보내는 일일 최대 초대장 수	25
사용자당 최대 개인용 액세스 토큰 수 (PAT)	100
모든 스페이스의 각 사용자 ID (CodeCatalyst별칭) 에 대한 최대 개인 연결 수 (제공자 유형별)	1

Resource	정보
비밀번호: CodeCatalyst	길이가 8~64자 사이인 모든 허용 문자 조합 유효한 문자: A-Z, a-z, 0-9. 비밀번호에는 다음과 같은 영숫자 이외의 문자가 포함될 수 있습니다. (~ ! @ # $ % ^ & * _ - + = ` | \ { } [ ] : ; " ' < > , . ? /)
PAT이름은 다음과 같습니다. CodeCatalyst	1~100자 사이의 모든 허용 문자 조합
프로젝트 멤버 초대가 만료될 때까지의 시간	24시간 후 만료
스페이스 구성원 초대가 만료될 때까지의 시간	24시간 후 만료
이메일 주소 확인이 만료되기까지 남은 시간	전송 후 10분 후 만료

문제 해결

이 섹션은 Amazon CodeCatalyst 프로필에 액세스하는 동안 발생할 수 있는 몇 가지 일반적인 문제를 해결하는 데 도움이 될 수 있습니다.

가입 문제

가입하는 동안 몇 가지 문제가 발생할 수 있습니다. 몇 가지 해결책이 있습니다.

내 이메일 주소는 이미 사용 중입니다.

입력한 이메일이 이미 사용 중이고 이를 자신의 것으로 알고 있다면 이미 당사 프로필이 있을 수 있습니다. 기존 ID로 로그인하세요. 기존 이메일을 소유하고 있지 않다면 사용하지 않는 다른 이메일로 가입하세요.

이메일 인증을 완료할 수 없습니다

확인 이메일을 받지 못한 경우

1. 스팸, 정크, 삭제된 항목 폴더를 확인하세요.

> **Note**
>
> 이 확인 이메일은 주소 no-reply@signin.aws 또는 중 하나에서 no-reply@login.awsapps.com 발송됩니다. 이러한 발신자 이메일 주소에서 오는 이메일을 수신하고 이를 정크 또는 스팸으로 처리하지 않도록 메일 시스템을 구성하는 것이 좋습니다.

2. 5분 정도 기다린 후 받은 편지함을 새로 고치세요. 스팸, 정크, 지운 편지함 폴더를 다시 확인하세요.
3. 여전히 확인 이메일이 보이지 않으면 코드 재전송을 선택하세요. 해당 페이지를 이미 종료한 경우 Amazon CodeCatalyst 가입 워크플로를 다시 시작하십시오.

비밀번호가 최소 요구 사항을 충족하지 않습니다.

보안을 위해 비밀번호는 8~20자 (대문자와 소문자, 숫자 모두) 를 포함해야 합니다.

로그인 문제

암호를 잊어버렸습니다

암호를 잊어버렸습니다 단원의 단계를 따르세요.

암호가 작동하지 않습니다

비밀번호를 설정하거나 변경할 때마다 다음 요구 사항을 준수해야 합니다.

- 암호는 대/소문자를 구분합니다.
- 암호는 대문자와 소문자, 숫자를 모두 포함하는 8~64자 길이여야 하며 영숫자가 아닌 문자를 하나 이상 포함해야 합니다.
- 마지막 세 개의 비밀번호는 다시 사용할 수 없습니다. .

활성화할 수 없어요 MFA

MFA활성화하려면 의 단계에 따라 프로필에 하나 이상의 MFA 장치를 추가하세요다단계 인증으로 로그인하도록 AWS 빌더 ID 구성 () MFA.

MFA장치를 추가할 수 없어요

다른 MFA 장치를 추가할 수 없는 경우 등록할 수 있는 장치 수 한도에 MFA 도달했을 수 있습니다. 새 장치를 추가하기 전에 기존 MFA 장치를 제거해야 할 수 있습니다.

MFA장치를 제거할 수 없습니다.

MFA비활성화하려면 의 단계에 따라 MFA 기기 제거를 진행하세요장치 삭제 MFA. 하지만 MFA 활성화된 상태를 유지하려면 기존 MFA 장치를 삭제하기 전에 다른 MFA 장치를 추가해야 합니다. 다른 MFA 장치 추가에 대한 자세한 내용은 을 참조하십시오다단계 인증에 사용할 장치를 등록하는 방법.

로그아웃 문제

로그아웃할 위치를 찾을 수 없어요

페이지 오른쪽 상단에서 로그아웃을 선택합니다.

로그아웃을 해도 완전히 로그아웃되지 않습니다.

시스템은 즉시 로그아웃하도록 설계되어 있지만 완전 로그아웃에는 최대 1시간이 걸릴 수 있습니다.

실패한 워크플로우로 인해 역할이 존재하지 않습니다. 라는 오류 메시지가 나타납니다.

문제: 웹 애플리케이션 또는 서버리스 블루프린트에서 프로젝트를 생성한 후 워크플로가 실패하고 다음 오류가 발생합니다.

CLIENT_ERROR: 역할이 존재하지 않습니다.

가능한 해결 방법: 워크플로를 실행할 권한이 있는 IAM 역할을 구성하고 워크플로에 IAM 역할을 추가한 후에도 계정 연결에 IAM 역할을 추가해야 할 수 있으므로 워크플로가 여전히 실패합니다. YAML 에 설명된 대로 스페이스의 계정 연결에 IAM 역할을 추가합니다계정 연결에 IAM 역할 추가.

실패한 워크플로로 인해 역할 오류가 발생합니다.

문제: 웹 애플리케이션 또는 서버리스 블루프린트에서 프로젝트를 생성한 후 워크플로가 실패하고 다음 오류가 발생합니다.

CLIENT_ERROR: 역할이 제대로 설정되지 않았거나 존재하지 않음

가능한 해결 방법: 프로젝트를 만든 스페이스에 연결을 설정하거나 계정 AWS 계정 연결 요청을 완료해야 할 수 있습니다. 스페이스가 이미 활성 AWS 계정 상태인 경우 워크플로 작업을 실행할 권한이 있는 IAM 역할을 만들고 추가하세요. 에 설명된 대로 계정 연결에 IAM 역할을 추가합니다계정 연결에 IAM 역할 추가.

가능한 해결 방법: 연결을 지정하지 않고 프로젝트를 만든 경우 계정 연결을 배포 환경에 연결해야 합니다. 스페이스에 이미 활성 AWS 계정 연결 및 IAM 역할이 추가된 경우 에 설명된 대로 IAM 역할이 포함된 계정 연결을 배포 환경에 추가해야 배포 환경에 계정 연결 및 IAM 역할 추가 합니다.

프로젝트 워크플로에서 IAM 역할을 업데이트해야 합니다.

AWS 계정 연결이 완전히 설정되고 IAM 역할이 생성되어 계정 연결에 추가된 경우 프로젝트 워크플로에서 IAM 역할을 업데이트할 수 있습니다.

1. CI/CD 옵션을 선택하고 워크플로를 선택합니다. YAML 버튼을 선택합니다.
2. 편집을 선택합니다.
3. `ActionRoleArn:`필드에서 역할을 IAM 업데이트된 IAM ARN 역할로 교체합니다. ARN 검증을 선택합니다.
4. 커밋을 선택합니다.

   메인라인 브랜치에 있는 경우 워크플로가 자동으로 시작됩니다. 그렇지 않으면 워크플로를 다시 실행하려면 [Run] 을 선택합니다.

개인용 연결을 만든 후 내 GitHub 계정에 대한 검토 요청이 들어왔습니다.

개인용 연결을 생성하면 CodeCatalyst 앱이 사용자 GitHub 계정에 GitHub 앱으로 설치됩니다. GitHub 업데이트된 읽기 또는 쓰기 권한이 필요한 특정 리소스가 CodeCatalyst 있는 경우 GitHub 계정에 액세스하여 설치된 앱의 권한을 업데이트해야 할 수 있습니다.

1. GitHub 로그인하고 설치된 앱의 계정 설정으로 이동합니다. 프로필 아이콘을 선택하고 설정을 선택한 다음 애플리케이션을 선택합니다.
2. 설치된 GitHub 앱 탭의 설치된 애플리케이션 목록에서 설치된 앱을 확인합니다 CodeCatalyst. 검토할 권한이 있는 경우 검토 요청 링크가 표시됩니다.
3. 링크를 선택한 다음 메시지가 표시되면 자격 증명을 확인합니다. 자격 증명을 입력하고 확인을 선택합니다.
4. 새 권한을 수락하고 권한을 적용할 저장소를 지정한 다음 [Save] 를 선택합니다.

지원 양식을 작성하려면 어떻게 해야 하나요?

Amazon으로 CodeCatalyst 이동하거나 Support Feedback 양식을 작성할 수 있습니다. 정보 요청 섹션의 도움을 받을 수 있는 방법 아래에 귀하가 Amazon CodeCatalyst 고객이라는 내용을 포함하십시오. 문제를 가장 효율적으로 해결할 수 있도록 최대한 자세하게 설명해 주세요.

확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst

CodeCatalyst Amazon에는 기능을 추가하고 외부 제품과 통합하는 데 도움이 되는 확장 프로그램이 포함되어 CodeCatalyst 있습니다. CodeCatalyst 카탈로그의 확장 프로그램을 사용하여 팀은 에서 경험을 사용자 지정할 수 CodeCatalyst 있습니다.

주제

사용 가능한 타사 확장 프로그램

리소스를 통합하기로 선택한 확장 프로그램에 따라 CodeCatalyst 프로젝트에 특정 기능을 추가할 수 있습니다.

GitHub 리포지토리 통합 CodeCatalyst

GitHub 개발자가 코드를 저장하고 관리하는 데 도움이 되는 클라우드 기반 서비스입니다. GitHub 리포지토리 확장을 사용하면 Amazon 프로젝트에서 연결된 GitHub 리포지토리를 사용할 수 있습니다. CodeCatalyst 새 프로젝트를 생성할 때 GitHub 리포지토리를 연결할 수도 있습니다. CodeCatalyst 자세한 내용은 연결된 타사 리포지토리를 사용하여 프로젝트 만들기 단원을 참조하십시오.

> **Note**
>
> - 프로젝트에는 비어 있거나 보관된 GitHub 리포지토리를 사용할 수 없습니다. CodeCatalyst
> - GitHub 리포지토리 확장은 Enterprise Server 리포지토리와 호환되지 않습니다. GitHub

GitHub 리포지토리 확장을 설치하고 구성하면 다음을 수행할 수 있습니다.

- 다음 소스 GitHub 리포지토리 목록에서 리포지토리를 확인하세요. CodeCatalyst
- 리포지토리의 워크플로 정의 파일 저장 및 관리 GitHub
- 연결된 GitHub 리포지토리에 저장된 파일을 개발 환경에서 생성, 읽기, 업데이트 및 삭제합니다. CodeCatalyst
- 연결된 리포지토리의 파일을 다음 위치에 저장하고 인덱싱합니다. GitHub CodeCatalyst
- 연결된 계정의 기존 리포지토리로 CodeCatalyst 프로젝트 생성 GitHub
- 블루프린트로 프로젝트를 생성하거나 블루프린트를 추가할 때 블루프린트에서 생성된 코드로 GitHub 리포지토리를 생성하세요.
- 연결된 리포지토리에 코드를 푸시하거나 연결된 GitHub 리포지토리에서 pull 요청을 생성, 수정 또는 닫으면 CodeCatalyst 워크플로가 자동으로 실행됩니다. GitHub
- 워크플로우에서 링크된 GitHub 리포지토리 소스 파일 사용 CodeCatalyst
- CodeCatalyst 워크플로우에서 GitHub 작업 읽기 및 실행
- CodeCatalyst 워크플로 실행 상태를 연결된 GitHub 리포지토리로 보내고 커밋 상태에 따라 GitHub 풀 리퀘스트 병합을 차단합니다.

Bitbucket 리포지토리를 다음과 같이 통합하기 CodeCatalyst

Bitbucket은 개발자가 코드를 저장하고 관리하는 데 도움이 되는 클라우드 기반 서비스입니다. Bitbucket 리포지토리 확장 프로그램을 사용하면 Amazon 프로젝트에서 연결된 Bitbucket 리포지토리

를 사용할 수 있습니다. CodeCatalyst 새 프로젝트를 만들 때 Bitbucket 리포지토리를 연결할 수도 있습니다. CodeCatalyst 자세한 내용은 연결된 타사 리포지토리를 사용하여 프로젝트 만들기 단원을 참조하십시오.

> **Note**
>
> - 비어 있거나 보관된 Bitbucket 리포지토리는 프로젝트와 함께 사용할 수 없습니다. CodeCatalyst
> - Bitbucket 리포지토리 확장은 Bitbucket 데이터 센터 리포지토리와 호환되지 않습니다.

Bitbucket 리포지토리 확장을 설치하고 구성하면 다음을 수행할 수 있습니다.

- 의 소스 리포지토리 목록에서 Bitbucket 리포지토리를 볼 수 있습니다. CodeCatalyst
- Bitbucket 리포지토리에 워크플로 정의 파일을 저장하고 관리합니다.
- 개발 환경에서 연결된 Bitbucket 리포지토리에 저장된 파일을 만들고, 읽고, 업데이트하고, 삭제합니다. CodeCatalyst
- 연결된 Bitbucket 계정의 기존 리포지토리를 사용하여 CodeCatalyst 프로젝트를 생성합니다.
- 연결된 Bitbucket 리포지토리의 파일을 다음 위치에 저장하고 인덱싱합니다. CodeCatalyst
- 블루프린트로 프로젝트를 만들거나 블루프린트를 추가할 때 블루프린트에서 생성된 코드로 Bitbucket 리포지토리를 만드세요.
- 연결된 Bitbucket 리포지토리에 코드를 푸시하거나 연결된 Bitbucket 리포지토리에서 pull 요청을 생성, 수정 또는 닫으면 CodeCatalyst 워크플로가 자동으로 실행됩니다.
- 연결된 Bitbucket 리포지토리 소스 파일을 워크플로우에 사용하십시오. CodeCatalyst
- CodeCatalyst 워크플로 실행 상태를 연결된 Bitbucket 리포지토리로 보내고 커밋 상태에 따라 Bitbucket 풀 리퀘스트 병합을 차단합니다.

GitLab 리포지토리 통합 CodeCatalyst

GitLab 개발자가 코드를 저장하고 관리하는 데 도움이 되는 클라우드 기반 서비스입니다. GitLab 리포지토리 확장 프로그램을 사용하면 Amazon GitLab 프로젝트에서 연결된 프로젝트 리포지토리를 사용할 수 있습니다. CodeCatalyst 새 프로젝트를 생성할 때 GitLab 프로젝트 리포지토리를 연결할 수도 있습니다. CodeCatalyst 자세한 내용은 연결된 타사 리포지토리를 사용하여 프로젝트 만들기 단원을 참조하십시오.

> **Note**
>
> - 비어 있거나 보관된 GitLab 프로젝트 리포지토리는 프로젝트와 함께 사용할 수 없습니다. CodeCatalyst
> - GitLab 리포지토리 확장은 자체 관리형 리포지토리와 호환되지 않습니다. GitLab

GitLab 리포지토리 확장을 설치하고 구성하면 다음을 수행할 수 있습니다.

- 의 소스 리포지토리 목록에서 GitLab 프로젝트 리포지토리를 확인하세요. CodeCatalyst
- 프로젝트 리포지토리에 워크플로 정의 파일을 저장하고 관리합니다. GitLab
- Dev Environments에서 연결된 GitLab 프로젝트 리포지토리에 저장된 파일을 만들고, 읽고, 업데이트하고, 삭제합니다. CodeCatalyst
- 연결된 사용자의 기존 리포지토리로 CodeCatalyst 프로젝트 생성 GitLab
- 연결된 GitLab 프로젝트 리포지토리의 파일을 다음 위치에 저장하고 색인을 생성합니다. CodeCatalyst
- 블루프린트로 GitLab 프로젝트를 만들거나 블루프린트를 추가할 때 블루프린트에서 생성된 코드로 프로젝트 리포지토리를 만드세요.
- 연결된 프로젝트 리포지토리에 코드를 푸시하거나 연결된 GitLab 프로젝트 리포지토리에서 pull 요청을 생성, 수정 또는 닫으면 CodeCatalyst 워크플로가 자동으로 실행됩니다. GitLab
- 연결된 GitLab 프로젝트 리포지토리 소스 파일을 워크플로우에 사용 CodeCatalyst
- CodeCatalyst 워크플로 실행 상태를 연결된 GitLab 프로젝트 리포지토리로 보내고 커밋 상태에 따라 GitLab 병합 요청을 차단합니다.

Jira 이슈 통합 CodeCatalyst

Jira는 애자일 개발팀이 작업을 계획, 할당, 추적, 보고, 관리하는 데 도움이 되는 소프트웨어 애플리케이션입니다. Jira 소프트웨어 확장 프로그램을 사용하면 Amazon CodeCatalyst 프로젝트에서 Jira 프로젝트를 사용할 수 있습니다.

> **Note**
>
> CodeCatalyst Jira 소프트웨어 클라우드와만 호환됩니다.

Amazon CodeCatalyst 프로젝트용 Jira Software 확장 프로그램을 설치하고 구성하면 다음을 수행할 수 있습니다.

- Jira 프로젝트를 프로젝트에 CodeCatalyst 연결하여 Jira 프로젝트에 액세스할 수 있습니다. CodeCatalyst
- 풀 리퀘스트로 Jira 이슈 업데이트하기 CodeCatalyst
- Jira 이슈에서 연결된 CodeCatalyst 풀 리퀘스트의 상태 및 워크플로 실행 보기

익스텐션 컨셉

다음은 에서 확장 프로그램을 사용할 때 알아야 할 몇 가지 개념과 용어입니다 CodeCatalyst.

확장 프로그램

확장 프로그램은 CodeCatalyst 스페이스에 설치하여 프로젝트에 새 기능을 추가하고 외부 서비스와 통합할 수 있는 애드온입니다 CodeCatalyst. 확장 프로그램은 카탈로그에서 찾아보고 설치할 수 있습니다. CodeCatalyst

CodeCatalyst 카탈로그

CodeCatalyst 카탈로그는 에서 사용할 수 있는 모든 확장 프로그램의 중앙 목록입니다 CodeCatalyst. CodeCatalyst 카탈로그를 탐색하여 소스, 워크플로 CodeCatalyst 등과 같은 영역에서 팀의 경험을 개선할 수 있는 확장 프로그램을 찾을 수 있습니다.

연결 및 연결

사용하거나 관리하려는 타사 리소스에 따라 GitHub 계정, Bitbucket 작업 공간 또는 Jira 프로젝트를 연결해야 합니다. 그런 다음 GitHub 리포지토리, Bitbucket 리포지토리 또는 Jira 프로젝트를 프로젝트에 연결해야 합니다. CodeCatalyst

- GitHub 리포지토리: GitHub 계정을 연결한 다음 리포지토리를 연결합니다. GitHub
- 비트버킷 리포지토리: Bitbucket 작업 공간을 연결한 다음 Bitbucket 리포지토리를 연결합니다.
- GitLab 리포지토리: GitLab 사용자를 연결한 다음 GitLab 프로젝트 리포지토리를 연결합니다.
- Jira 소프트웨어: Jira 사이트를 연결한 다음 Jira 프로젝트를 연결합니다.

빠른 시작: 확장 프로그램 설치, 공급자 연결, 리소스 연결 CodeCatalyst

이 자습서에서는 다음 세 가지 작업을 단계별로 안내합니다.

1. 리포지토리, GitHub Bitbucket 리포지토리, 리포지토리 또는 Jira Software 확장을 설치합니다. GitLab 외부 사이트에 연결하여 타사 리소스에 대한 액세스 권한을 CodeCatalyst 제공하라는 메시지가 표시되며, 이 작업은 다음 단계의 일부로 수행됩니다.

> **Important**
>
> 스페이스에 GitHub 리포지토리, Bitbucket 리포지토리, GitLab 리포지토리 또는 Jira Software 확장 프로그램을 설치하려면 스페이스에서 CodeCatalyst 스페이스 관리자 역할을 가진 계정으로 로그인해야 합니다.

2. GitHub 계정, Bitbucket 작업 영역, GitLab 사용자 또는 Jira 사이트를 연결합니다. CodeCatalyst

> **Important**
>
> GitHub 계정, Bitbucket 작업 영역, GitLab 사용자 또는 Jira 사이트를 스페이스에 연결하려면 타사 소스의 관리자이자 CodeCatalyst 스페이스 관리자여야 합니다. CodeCatalyst

> **Important**
>
> 리포지토리 확장을 설치한 후 연결하는 모든 리포지토리는 해당 코드를 인덱싱하여 CodeCatalyst 저장합니다. CodeCatalyst 그러면 코드를 검색할 수 있게 됩니다. CodeCatalyst 에서 CodeCatalyst 연결된 리포지토리를 사용할 때의 코드 데이터 보호를 더 잘 이해하려면 Amazon CodeCatalyst User Guide의 데이터 보호를 참조하십시오.

> **Note**
>
> GitHub 계정 연결을 사용하는 경우 개인 연결을 생성하여 자격 증명과 ID 간의 ID 매핑을 설정해야 합니다. CodeCatalyst GitHub 자세한 내용은 개인 연결 및 개인적인 연결을 통해 GitHub 리소스에 접근하기 단원을 참조하세요.

3. GitHub 리포지토리, Bitbucket 리포지토리, GitLab 프로젝트 리포지토리 또는 Jira 프로젝트를 프로젝트에 연결합니다. CodeCatalyst

> **Important**
>
> - 기여자로서 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 연결할 수 있지만, 타사 리포지토리는 스페이스 관리자 또는 프로젝트 관리자만 연결을 해제할 수 있습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.
> - Jira 프로젝트를 프로젝트에 연결하려면 CodeCatalyst Space 관리자 또는 CodeCatalyst 프로젝트 관리자여야 합니다. CodeCatalyst

> **Important**
>
> CodeCatalyst 연결된 리포지토리의 기본 브랜치에서 변경 사항 감지를 지원하지 않습니다. 연결된 저장소의 기본 분기를 변경하려면 먼저 연결을 해제하고 기본 분기를 변경한 다음 다시 연결해야 합니다. CodeCatalyst 자세한 내용은 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, GitLab Jira 프로젝트 연결 CodeCatalyst 단원을 참조하십시오. 가장 좋은 방법은 리포지토리를 연결하기 전에 항상 최신 버전의 확장 프로그램을 사용하는 것입니다.

> **Note**
>
> - GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리는 스페이스의 한 CodeCatalyst 프로젝트에만 연결할 수 있습니다.
> - 비어 있거나 보관된 GitHub 리포지토리, Bitbucket 리포지토리 또는 프로젝트 리포지토리는 프로젝트와 함께 사용할 수 없습니다. GitLab CodeCatalyst
> - 프로젝트의 리포지토리와 이름이 같은 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리는 연결할 수 없습니다. CodeCatalyst
> - GitHub 리포지토리 확장은 GitHub 엔터프라이즈 서버 리포지토리와 호환되지 않습니다.
> - Bitbucket 리포지토리 확장은 Bitbucket 데이터 센터 리포지토리와 호환되지 않습니다.
> - 리포지토리 확장은 자체 관리형 GitLab 프로젝트 리포지토리와 호환되지 않습니다. GitLab

- 연결된 리포지토리에서는 나를 위한 설명 쓰기 또는 댓글 요약 기능을 사용할 수 없습니다. 이러한 기능은 풀 리퀘스트 인에서만 사용할 수 있습니다. CodeCatalyst
- CodeCatalyst 프로젝트는 하나의 Jira 프로젝트에만 연결될 수 있습니다. Jira 프로젝트를 여러 CodeCatalyst 프로젝트에 연결할 수 있습니다.

또한 새 프로젝트를 만들 때 GitHub 리포지토리, Bitbucket 리포지토리, GitLab 리포지토리 확장을 설치하고, GitHub 계정, Bitbucket 작업 공간 또는 GitLab 사용자에 연결하고, 타사 리포지토리를 연결할 수 있습니다. CodeCatalyst 자세한 내용은 연결된 타사 리포지토리를 사용하여 프로젝트 만들기 단원을 참조하십시오.

주제

1단계: 카탈로그에서 타사 확장 프로그램 설치 CodeCatalyst

에서 타사 리소스를 사용하기 위한 첫 번째 CodeCatalyst 단계는 카탈로그에서 GitHub 리포지토리 확장을 설치하는 것입니다. CodeCatalyst 확장을 설치하려면 다음 단계를 수행하여 사용하려는 타사 리소스의 확장을 선택합니다. GitHub 리포지토리, Bitbucket 리포지토리 및 리포지토리를 통해 에서 GitLab 리포지토리, Bitbucket 리포지토리 또는 프로젝트 GitHub 리포지토리를 사용할 수 있습니다. GitLab CodeCatalyst Jira 소프트웨어를 사용하면 CodeCatalyst Jira 이슈를 에서 관리할 수 있습니다.

카탈로그에서 확장 프로그램을 설치하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.
3. 상단 메뉴에서 CodeCatalyst CodeCatalyst 카탈로그 아이콘을

   선택하여 카탈로그로 이동합니다. 리포지토리, GitHub Bitbucket 리포지토리, 리포지토리 또는 Jira Software를 검색할 수 있습니다. GitLab 카테고리를 기준으로 확장을 필터링할 수도 있습니다.

4. (선택 사항) 확장 프로그램에 부여될 권한 등 확장 프로그램에 대한 자세한 내용을 보려면 확장 프로그램 이름을 선택합니다.
5. 설치를 선택합니다. 확장에 필요한 권한을 검토하고 계속하려면 설치를 다시 선택합니다.

확장 프로그램을 설치하면 확장 프로그램 세부 정보 페이지로 이동합니다. 설치한 확장 프로그램에 따라 연결된 공급자 및 연결된 리소스를 보고 관리할 수 있습니다.

2단계: 타사 제공업체를 CodeCatalyst 공간에 연결

리포지토리, GitHub Bitbucket 리포지토리, 리포지토리 또는 Jira Software 확장을 설치한 후 다음 단계는 GitHub 계정, Bitbucket 작업 영역, GitLab 프로젝트 GitLab 리포지토리 또는 Jira 사이트를 스페이스에 연결하는 것입니다. CodeCatalyst

계정, Bitbucket 작업 공간 또는 Jira 사이트를 연결하려면 GitHub CodeCatalyst

- 설치한 타사 확장 프로그램에 따라 다음 중 하나를 수행하십시오.
  - GitHub 리포지토리: 계정에 연결합니다. GitHub
    1. 연결된 GitHub 계정 탭에서 Connect account (GitHub 계정 연결) 를 선택하여 외부 사이트로 이동합니다 GitHub.
    2. GitHub 자격 증명을 사용하여 GitHub 계정에 로그인한 다음 Amazon을 설치할 계정을 선택합니다 CodeCatalyst.

       > **Tip**
       >
       > 이전에 GitHub 계정을 스페이스에 연결한 경우 재승인 메시지가 표시되지 않습니다. 대신 둘 이상의 조직의 구성원 또는 공동 작업자인 경우 확장 프로그램을 설치할 위치를 묻는 대화 상자가 표시되고, 한 GitHub 조직에만 속해 있는 경우 Amazon CodeCatalyst 애플리케이션의 구성 페이지가 표시됩니다. GitHub 허용하려는 리포지토리 액세스를 허용하도록 애플리케이션을 구성한 다음 [Save] 를 선택합니다. 저장 버튼이 활성화되지 않은 경우 구성을 변경한 다음 다시 시도하십시오.

    3. 현재 및 미래의 모든 리포지토리에 대한 액세스를 CodeCatalyst 허용할지, 아니면 사용할 특정 GitHub 리포지토리를 선택할지를 선택합니다. CodeCatalyst 기본 옵션은 향후 액세스할 GitHub 리포지토리를 포함하여 GitHub 계정의 모든 리포지토리를 포함하는 것입니다. CodeCatalyst
    4. 부여된 권한을 검토한 다음 설치를 CodeCatalyst 선택합니다.

계정을 연결하면 연결된 GitHub 계정과 연결된 GitHub GitHub 리포지토리를 보고 관리할 수 있는 리포지토리 확장 세부 정보 페이지로 이동합니다. CodeCatalyst GitHub

- 비트버킷 리포지토리: Bitbucket 작업 공간에 연결합니다.

  1. 연결된 비트버킷 작업 영역 탭에서 Bitbucket 연결 작업 영역을 선택하여 Bitbucket의 외부 사이트로 이동합니다.
  2. Bitbucket 자격 증명을 사용하여 Bitbucket 작업 공간에 로그인하고 부여된 권한을 검토하십시오. CodeCatalyst
  3. 작업 영역 승인 드롭다운 메뉴에서 CodeCatalyst 액세스를 제공하려는 Bitbucket 작업 영역을 선택한 다음 액세스 허용을 선택합니다.

     > **Tip**
     >
     > 이전에 Bitbucket 작업 영역을 공간에 연결한 경우 재승인 메시지가 표시되지 않습니다. 대신 둘 이상의 Bitbucket 작업 영역의 구성원 또는 공동 작업자인 경우 확장 프로그램을 설치할 위치를 묻는 대화 상자가 표시되고, 한 Bitbucket 작업 영역에만 속하는 경우 Amazon CodeCatalyst 애플리케이션의 구성 페이지가 표시됩니다. 허용하려는 작업 영역 액세스를 위해 애플리케이션을 구성한 다음 액세스 허용을 선택합니다. 액세스 허용 버튼이 활성화되지 않은 경우 구성을 변경한 다음 다시 시도하십시오.

  Bitbucket 작업 영역을 연결하면 연결된 Bitbucket 작업 영역 및 연결된 Bitbucket 리포지토리를 보고 관리할 수 있는 Bitbucket 리포지토리 확장 세부 정보 페이지로 이동합니다. CodeCatalyst

- GitLab 리포지토리: 사용자에게 연결합니다. GitLab

  1. Connect GitLab user를 선택하여 외부 사이트로 이동합니다 GitLab.
  2. GitLab 자격 증명을 사용하여 GitLab 사용자에게 로그인하고 부여된 권한을 검토하십시오 CodeCatalyst.

     > **Tip**
     >
     > 이전에 GitLab 사용자를 스페이스에 연결한 경우 재인증 메시지가 표시되지 않습니다. 대신 콘솔로 다시 이동하게 됩니다. CodeCatalyst

  3. [AWS 커넥터 인증 대상] 을 선택합니다. GitLab

사용자를 연결하면 연결된 GitLab 사용자 및 연결된 GitLab 프로젝트 GitLab 리포지토리를 보고 관리할 수 있는 리포지토리 확장 세부정보 페이지로 이동합니다. CodeCatalyst GitLab

- Jira 소프트웨어: Jira 사이트를 연결합니다.

  1. 연결된 Jira 사이트 탭에서 Connect Jira 사이트를 선택하여 Atlassian Marketplace의 외부 사이트로 이동합니다.
  2. 지금 다운로드를 선택하여 Jira CodeCatalyst 사이트에 설치를 시작하세요.

     > **Note**
     >
     > 이전에 Jira CodeCatalyst 사이트에 설치한 경우 알림을 받게 됩니다. 시작하기를 선택하여 마지막 단계로 이동합니다.

  3. 역할에 따라 다음 중 하나를 수행하십시오.

     1. Jira 사이트 관리자인 경우 사이트 드롭다운 메뉴에서 CodeCatalyst 애플리케이션을 설치할 Jira 사이트를 선택한 다음 앱 설치를 선택합니다.

        > **Note**
        >
        > Jira 사이트가 하나 있는 경우 이 단계는 표시되지 않고 자동으로 다음 단계로 이동합니다.

     2. a. Jira 관리자가 아닌 경우 사이트 드롭다운 메뉴에서 CodeCatalyst 애플리케이션을 설치할 Jira 사이트를 선택한 다음 앱 요청을 선택합니다. Jira 앱 설치에 대한 자세한 내용은 누가 앱을 설치할 수 있나요? 를 참조하십시오. .

        b. 설치가 CodeCatalyst 필요한 이유를 입력 텍스트 필드에 입력하거나 기본 텍스트를 유지한 다음 요청 제출을 선택합니다.

  4. 애플리케이션이 CodeCatalyst 설치될 때 수행한 작업을 검토한 다음 Get it now를 선택합니다.
  5. 애플리케이션이 설치된 후 Return to를 선택하여 다음으로 CodeCatalyst CodeCatalyst 돌아가십시오.

Jira 사이트를 연결한 후에는 Jira Software 확장 세부정보 페이지의 Connected Jira 사이트 탭에서 연결된 사이트를 볼 수 있습니다. CodeCatalyst

3단계: 타사 리소스를 프로젝트에 연결 CodeCatalyst

리포지토리, GitHub Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 사용하거나 Jira 이슈를 관리하기 위한 세 번째이자 마지막 단계는 사용하려는 CodeCatalyst 프로젝트에 리포지토리를 연결하는 것입니다. CodeCatalyst

확장 세부정보 페이지에서 GitHub 리포지토리, Bitbucket 리포지토리, GitLab 프로젝트 리포지토리 또는 Jira 프로젝트를 프로젝트에 연결하려면 CodeCatalyst

- 설치한 타사 확장 프로그램과 연결한 제공자에 따라 다음 중 하나를 수행하십시오.
  - GitHub 리포지토리: 리포지토리를 연결합니다. GitHub
    1. 연결된 GitHub 리포지토리 탭에서 리포지토리 연결을 선택합니다. GitHub
    2. GitHub 계정 드롭다운에서 연결하려는 저장소가 포함된 GitHub 계정을 선택합니다.
    3. GitHub 리포지토리 드롭다운에서 프로젝트에 연결하려는 리포지토리를 선택합니다. CodeCatalyst

       > **Tip**
       >
       > 저장소 이름이 회색으로 표시된 경우 해당 저장소는 스페이스의 다른 프로젝트에 이미 링크되어 있으므로 해당 저장소를 링크할 수 없습니다.

    4. (선택 사항) 리포지토리 목록에 GitHub 리포지토리가 없다면 Amazon CodeCatalyst 애플리케이션에서 리포지토리에 액세스할 수 있도록 구성되지 않은 것일 수 있습니다. GitHub 연결된 CodeCatalyst 계정에서 사용할 수 있는 GitHub 리포지토리를 구성할 수 있습니다.
       a. GitHub계정으로 이동하여 설정을 선택한 다음 애플리케이션을 선택합니다.
       b. 설치된 GitHub 앱 탭에서 Amazon CodeCatalyst 애플리케이션에 맞게 구성을 선택합니다.
       c. 다음 중 하나를 수행하여 연결하려는 GitHub 리포지토리에 대한 액세스를 구성하십시오. CodeCatalyst
          - 현재 및 미래의 모든 리포지토리에 대한 액세스를 제공하려면 모든 리포지토리를 선택합니다.
          - 특정 리포지토리에 대한 액세스를 제공하려면 리포지토리만 선택을 선택하고 리포지토리 선택 드롭다운을 선택한 다음 연결을 허용할 리포지토리를 선택합니다. CodeCatalyst

5. CodeCatalyst 프로젝트 드롭다운 메뉴에서 리포지토리를 연결하려는 CodeCatalyst 프로젝트를 선택합니다. GitHub
6. 연결을 선택합니다.

에서 GitHub CodeCatalyst 리포지토리를 더 이상 사용하지 않으려면 프로젝트에서 리포지토리를 연결 해제할 수 있습니다. CodeCatalyst 리포지토리의 연결이 해제되면 해당 리포지토리의 이벤트가 워크플로를 시작하지 않으므로 해당 리포지토리를 CodeCatalyst Dev Environments와 함께 사용할 수 없습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

- 비트버킷 리포지토리: Bitbucket 리포지토리를 연결합니다.

1. 연결된 비트버킷 리포지토리 탭에서 Bitbucket 리포지토리 연결을 선택합니다.
2. Bitbucket 작업 영역 드롭다운에서 연결하려는 리포지토리가 포함된 Bitbucket 작업 영역을 선택합니다.
3. Bitbucket 리포지토리 드롭다운에서 프로젝트에 연결하려는 리포지토리를 선택합니다. CodeCatalyst

> **Tip**
>
> 리포지토리 이름이 회색으로 표시된 경우 해당 리포지토리가 스페이스의 다른 프로젝트에 이미 연결되어 있기 때문에 해당 리포지토리를 연결할 수 없습니다.

4. CodeCatalyst 프로젝트 드롭다운 메뉴에서 Bitbucket 리포지토리를 연결할 CodeCatalyst 프로젝트를 선택합니다.
5. 연결을 선택합니다.

에서 CodeCatalyst Bitbucket 리포지토리를 더 이상 사용하지 않으려면 프로젝트에서 Bitbucket 리포지토리를 연결 해제할 수 있습니다. CodeCatalyst 리포지토리의 연결이 해제되면 해당 리포지토리의 이벤트가 워크플로우 실행을 시작하지 않으므로 해당 리포지토리를 Dev Environments와 함께 CodeCatalyst 사용할 수 없습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

- GitLab 리포지토리: 프로젝트 리포지토리를 연결합니다. GitLab

1. 연결된 GitLab 프로젝트 리포지토리 탭에서 프로젝트 리포지토리 연결을 선택합니다. GitLab

2. GitLab 사용자 드롭다운에서 연결하려는 저장소가 들어 있는 GitLab 사용자를 선택합니다.
3. GitLab 프로젝트 리포지토리 드롭다운에서 프로젝트에 연결하려는 리포지토리를 선택합니다. CodeCatalyst

> **Tip**
>
> 저장소 이름이 회색으로 표시된 경우 해당 저장소는 스페이스의 다른 프로젝트에 이미 연결되어 있으므로 해당 저장소를 연결할 수 없습니다.

4. CodeCatalyst 프로젝트 드롭다운 메뉴에서 CodeCatalyst 프로젝트 저장소를 연결하려는 GitLab 프로젝트를 선택합니다.
5. 연결을 선택합니다.

에서 CodeCatalyst 프로젝트 리포지토리를 더 이상 사용하지 않으려면 GitLab 프로젝트에서 프로젝트 리포지토리를 연결 해제할 수 있습니다. CodeCatalyst 프로젝트 리포지토리의 연결이 해제되면 해당 프로젝트 리포지토리의 이벤트가 워크플로우 실행을 시작하지 않으므로 해당 프로젝트 리포지토리를 CodeCatalyst Dev Environments와 함께 사용할 수 없습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

- Jira 소프트웨어: Jira 프로젝트를 연결합니다.

1. 연결된 Jira 프로젝트 탭에서 Jira 프로젝트 연결을 선택합니다.
2. Jira 사이트 드롭다운 메뉴에서 연결하려는 프로젝트가 포함된 Jira 사이트를 선택합니다.
3. Jira 프로젝트 드롭다운 메뉴에서 프로젝트에 연결하려는 프로젝트를 선택합니다. CodeCatalyst
4. CodeCatalyst 프로젝트 드롭다운 메뉴에서 Jira CodeCatalyst 프로젝트에 연결하려는 프로젝트를 선택합니다.
5. 연결을 선택합니다.

Jira 프로젝트가 프로젝트에 연결되면 CodeCatalyst 이슈에 대한 액세스가 완전히 비활성화되고 CodeCatalyst 탐색 창의 이슈가 Jira CodeCatalyst 프로젝트로 연결되는 Jira 이슈 항목으로 대체됩니다.

에서 CodeCatalyst 더 이상 Jira 프로젝트를 사용하지 않으려면 프로젝트에서 Jira 프로젝트를 연결 해제할 수 있습니다. CodeCatalyst Jira 프로젝트가 연결 해제되면 프로젝트에서 Jira 이슈

를 사용할 수 없게 되며 Issues는 CodeCatalyst 다시 CodeCatalyst 이슈 제공자가 됩니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 Code의 소스 리포지토리에서 프로젝트에 연결할 수도 있습니다. 자세한 내용은 연결된 타사 제공업체의 리소스 연결 단원을 참조하십시오.

다음 단계

리포지토리, GitHub Bitbucket 리포지토리 또는 GitLab 리포지토리 확장을 설치하고, 리소스 공급자를 연결하고, 타사 리포지토리를 프로젝트에 연결한 후 워크플로와 개발 환경에서 사용할 수 있습니다 CodeCatalyst . CodeCatalyst 블루프린트에서 생성된 코드를 사용하여 연결된 GitHub 계정, Bitbucket 작업 공간 또는 사용자에 타사 리포지토리를 만들 수도 있습니다. GitLab 자세한 내용은 타사 리포지토리 이벤트 이후 자동으로 워크플로 실행 및 Dev Environment 생성 단원을 참조하세요.

Jira Software 확장 프로그램을 설치하고, Jira 사이트를 연결하고, Jira 프로젝트를 프로젝트에 연결하고, 풀 리퀘스트를 연결하면 의 업데이트가 Jira CodeCatalyst 프로젝트에 반영됩니다. CodeCatalyst 풀 리퀘스트를 Jira 이슈에 연결하는 방법에 대한 자세한 내용은 을 참조하십시오. Jira 이슈를 CodeCatalyst 풀 리퀘스트에 연결 Jira에서 CodeCatalyst 이벤트를 보는 방법에 대한 자세한 내용은 을 참조하십시오. Jira 이슈 CodeCatalyst 이벤트 보기

스페이스에 확장 프로그램 설치

스페이스의 프로젝트에 기능을 추가하는 확장 프로그램을 CodeCatalyst 스페이스에 설치할 수 있습니다. 카탈로그 아이콘을 선택하여 CodeCatalyst 카탈로그를 볼 수

있습니다. 확장 및 기능에 대한 자세한 내용은 을 참조하십시오사용 가능한 타사 확장 프로그램.

> **Important**
>
> 확장 프로그램을 설치하려면 스페이스에서 스페이스 관리자 역할을 가진 계정으로 로그인해야 합니다.

> **Important**
>
> 저장소 확장을 설치한 후에는 연결하는 CodeCatalyst 모든 저장소의 코드가 색인화되고 저장됩니다. CodeCatalyst 그러면 코드를 검색할 수 있게 됩니다. CodeCatalyst 에서 CodeCatalyst 연결된 리포지토리를 사용할 때의 코드 데이터 보호를 더 잘 이해하려면 Amazon CodeCatalyst User Guide의 데이터 보호를 참조하십시오.

카탈로그에서 확장 프로그램을 설치하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.
3. 상단 메뉴에서 CodeCatalyst 카탈로그 아이콘을

   선택하여 카탈로그로 이동합니다. 리포지토리, GitHub Bitbucket 리포지토리, 리포지토리 또는 Jira Software를 검색할 수 있습니다. GitLab 카테고리를 기준으로 확장을 필터링할 수도 있습니다.
4. (선택 사항) 확장 프로그램의 이름을 선택하면 확장 프로그램이 갖게 될 권한 등 확장 프로그램에 대한 자세한 내용을 볼 수 있습니다.
5. 설치를 선택합니다. 확장에 필요한 권한을 검토하고 계속하려면 설치를 다시 선택합니다.

확장 프로그램을 설치하면 설치된 확장 프로그램의 세부 정보 페이지가 표시됩니다. 확장 프로그램에 대한 자세한 내용은 탭을 찾아보십시오. 필요한 경우 세부 정보 페이지에서 확장 프로그램의 추가 구성을 수행할 수도 있습니다.

스페이스에서 확장 프로그램 제거

스페이스에 이전에 설치된 확장 프로그램을 제거할 수 있습니다 CodeCatalyst . 확장을 제거하면 CodeCatalyst 스페이스 또는 프로젝트에서 해당 확장과 관련된 리소스가 제거될 수 있습니다.

> **Important**
>
> 확장 프로그램을 제거하려면 스페이스에서 스페이스 관리자 역할을 가진 계정으로 로그인해야 합니다.

스페이스에서 확장 프로그램을 제거하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.
3. 스페이스에 설치된 확장 프로그램 목록을 보려면 다음 중 하나를 수행하십시오.
   a. 설정을 선택한 다음 설치된 확장을 선택합니다.
   b. 상단 메뉴에서 카탈로그 아이콘을

      선택합니다.
4. 제거하려는 확장 프로그램에서 구성을 선택합니다.
5. 확장 세부 정보 페이지에서 제거를 선택합니다.
6. 확장 프로그램 제거 대화 상자의 정보를 검토하십시오. 지침을 따른 다음 제거를 선택하여 확장 프로그램을 제거합니다.

GitHub 계정, Bitbucket 작업 영역, GitLab 사용자, Jira 사이트 연결 CodeCatalyst

에서 GitHub CodeCatalyst 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 사용하거나 Jira 프로젝트를 관리하려면 먼저 타사 소스를 스페이스에 연결해야 합니다. CodeCatalyst 확장 및 기능에 대한 자세한 내용은 을 참조하십시오. 사용 가능한 타사 확장 프로그램

Important

GitHub 계정, Bitbucket 작업 영역, GitLab 사용자 또는 Jira 사이트를 스페이스에 연결하려면 타사 소스의 관리자이자 CodeCatalyst 스페이스 관리자여야 합니다. CodeCatalyst

Note

GitHub 계정 연결을 사용하는 경우 개인 연결을 만들어 ID와 ID 간에 ID 매핑을 설정해야 합니다. CodeCatalyst GitHub 자세한 내용은 개인 연결 및 개인적인 연결을 통해 GitHub 리소스에 접근하기 단원을 참조하세요.

GitHub 계정, Bitbucket 작업 영역, GitLab 사용자 또는 Jira 사이트를 연결하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.
3. 스페이스에 설치된 확장 프로그램 목록을 보려면 다음 중 하나를 수행하십시오.

   a. [설정] 을 선택한 다음 [설치된 확장 프로그램] 을 선택합니다.

   b. 상단 메뉴에서 카탈로그 아이콘을

   선택합니다.
4. 구성하려는 확장 (리포지토리, GitHub Bitbucket 리포지토리, 리포지토리 또는 Jira Software) 중 하나에 대해 구성을 선택합니다. GitLab
5. 구성하기로 선택한 타사 확장 프로그램에 따라 다음 중 하나를 수행하십시오.

   - GitHub 리포지토리: 계정에 연결합니다. GitHub

     1. 연결된 GitHub 계정 탭에서 Connect account (GitHub 계정 연결) 를 선택하여 외부 사이트로 이동합니다 GitHub.
     2. GitHub 자격 증명을 사용하여 GitHub 계정에 로그인한 다음 Amazon을 설치할 계정을 선택합니다 CodeCatalyst.

        > **Tip**
        >
        > 이전에 GitHub 계정을 스페이스에 연결한 경우 재승인 메시지가 표시되지 않습니다. 대신 둘 이상의 스페이스에서 구성원 또는 공동 작업자인 경우 확장 프로그램을 설치할 위치를 묻는 대화 상자가 표시되고, 한 GitHub 스페이스에만 속하는 경우 Amazon CodeCatalyst 애플리케이션의 구성 페이지가 표시됩니다. GitHub 허용하려는 리포지토리 액세스를 허용하도록 애플리케이션을 구성한 다음 [Save] 를 선택합니다. 저장 버튼이 활성화되지 않은 경우 구성을 변경한 다음 다시 시도하십시오.

     3. 현재 및 미래의 모든 리포지토리에 대한 액세스를 CodeCatalyst 허용할지, 아니면 사용할 특정 GitHub 리포지토리를 선택할지를 선택합니다. CodeCatalyst 기본 옵션은 향후 액세스할 GitHub 리포지토리를 포함하여 GitHub 계정의 모든 리포지토리를 포함하는 것입니다. CodeCatalyst
     4. 부여된 권한을 검토한 다음 설치를 CodeCatalyst 선택합니다.

계정을 연결하면 연결된 GitHub 계정과 연결된 GitHub GitHub 리포지토리를 보고 관리할 수 있는 리포지토리 확장 세부 정보 페이지로 이동합니다. CodeCatalyst GitHub

- 비트버킷 리포지토리: Bitbucket 작업 공간에 연결합니다.

  1. 연결된 비트버킷 작업 영역 탭에서 Bitbucket 연결 작업 영역을 선택하여 Bitbucket의 외부 사이트로 이동합니다.
  2. Bitbucket 자격 증명을 사용하여 Bitbucket 작업 공간에 로그인하고 부여된 권한을 검토하십시오. CodeCatalyst
  3. 작업 영역 승인 드롭다운 메뉴에서 CodeCatalyst 액세스를 제공하려는 Bitbucket 작업 영역을 선택한 다음 액세스 허용을 선택합니다.

     > **Tip**
     >
     > 이전에 Bitbucket 작업 영역을 공간에 연결한 경우 재승인 메시지가 표시되지 않습니다. 대신 둘 이상의 Bitbucket 작업 영역의 구성원 또는 공동 작업자인 경우 확장 프로그램을 설치할 위치를 묻는 대화 상자가 표시되고, 한 Bitbucket 작업 영역에만 속하는 경우 Amazon CodeCatalyst 애플리케이션의 구성 페이지가 표시됩니다. 허용하려는 작업 영역 액세스를 위해 애플리케이션을 구성한 다음 액세스 허용을 선택합니다. 액세스 허용 버튼이 활성화되지 않은 경우 구성을 변경한 다음 다시 시도하십시오.

  Bitbucket 작업 영역을 연결하면 연결된 Bitbucket 작업 영역 및 연결된 Bitbucket 리포지토리를 보고 관리할 수 있는 Bitbucket 리포지토리 확장 세부 정보 페이지로 이동합니다. CodeCatalyst

- GitLab 리포지토리: 사용자에게 연결합니다. GitLab

  1. Connect GitLab user를 선택하여 외부 사이트로 이동합니다 GitLab.
  2. GitLab 자격 증명을 사용하여 GitLab 사용자에게 로그인하고 부여된 권한을 검토하십시오 CodeCatalyst.

     > **Tip**
     >
     > 이전에 GitLab 사용자를 스페이스에 연결한 경우 재승인 메시지가 표시되지 않습니다. 대신 콘솔로 다시 이동하게 됩니다. CodeCatalyst

  3. [AWS 커넥터 인증 대상] 을 선택합니다. GitLab

사용자를 연결하면 연결된 GitLab 사용자 및 연결된 GitLab 프로젝트 GitLab 리포지토리를 보고 관리할 수 있는 리포지토리 확장 세부정보 페이지로 이동합니다. CodeCatalyst GitLab

- Jira 소프트웨어: Jira 사이트를 연결합니다.

  1. 연결된 Jira 사이트 탭에서 Connect Jira 사이트를 선택하여 Atlassian Marketplace의 외부 사이트로 이동합니다.
  2. 지금 다운로드를 선택하여 Jira CodeCatalyst 사이트에 설치를 시작하세요.

     > **Note**
     >
     > 이전에 Jira CodeCatalyst 사이트에 설치한 경우 알림을 받게 됩니다. 시작하기를 선택하여 마지막 단계로 이동합니다.

  3. 역할에 따라 다음 중 하나를 수행하십시오.

     1. Jira 사이트 관리자인 경우 사이트 드롭다운 메뉴에서 CodeCatalyst 애플리케이션을 설치할 Jira 사이트를 선택한 다음 앱 설치를 선택합니다.

        > **Note**
        >
        > Jira 사이트가 하나 있는 경우 이 단계는 표시되지 않고 자동으로 다음 단계로 이동합니다.

     2. a. Jira 관리자가 아닌 경우 사이트 드롭다운 메뉴에서 CodeCatalyst 애플리케이션을 설치할 Jira 사이트를 선택한 다음 앱 요청을 선택합니다. Jira 앱 설치에 대한 자세한 내용은 누가 앱을 설치할 수 있나요? 를 참조하십시오. .

        b. 설치가 CodeCatalyst 필요한 이유를 입력 텍스트 필드에 입력하거나 기본 텍스트를 유지한 다음 요청 제출을 선택합니다.

  4. 애플리케이션이 CodeCatalyst 설치될 때 수행한 작업을 검토한 다음 Get it now를 선택합니다.
  5. 애플리케이션이 설치된 후 Return to를 선택하여 다음으로 CodeCatalyst CodeCatalyst 돌아가십시오.

Jira 사이트를 연결한 후에는 Jira Software 확장 세부정보 페이지의 Connected Jira 사이트 탭에서 연결된 사이트를 볼 수 있습니다. CodeCatalyst

리포지토리, GitHub Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 더 이상 사용하지 않거나 Jira 이슈를 관리하고 싶지 않은 경우 타사 소스의 연결을 끊을 수 있습니다. CodeCatalyst GitHub 계정, Bitbucket 작업 영역 또는 GitLab 사용자의 연결이 끊기면 타사 리포지토리의 이벤트가 워크플로우 실행을 시작하지 않으므로 해당 리포지토리를 Dev Environments와 함께 사용할 수 없습니다. CodeCatalyst Jira 사이트 연결이 끊어지면 사이트 프로젝트의 Jira 이슈를 프로젝트에서 사용할 수 없게 되며 Issues는 다시 이슈 제공자가 CodeCatalyst 됩니다. CodeCatalyst 자세한 내용은 GitHub 계정, Bitbucket 작업 영역, GitLab 사용자, Jira 사이트 연결 끊기 CodeCatalyst 단원을 참조하십시오.

GitHub 계정, Bitbucket 작업 영역, GitLab 사용자, Jira 사이트 연결 끊기 CodeCatalyst

더 이상 리포지토리, GitHub Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 사용하지 않거나 에서 Jira 이슈를 관리하고 싶지 않다면 타사 소스의 연결을 끊을 수 있습니다. CodeCatalyst GitHub 계정, Bitbucket 작업 영역 또는 GitLab 사용자의 연결이 끊기면 리포지토리의 이벤트가 워크플로우 실행을 시작하지 않으므로 해당 리포지토리를 Dev Environments와 함께 사용할 수 없습니다. CodeCatalyst Jira 사이트 연결이 끊어지면 사이트 프로젝트의 Jira 이슈를 프로젝트에서 사용할 수 없게 되며 Issues는 다시 이슈 제공자가 CodeCatalyst 됩니다. CodeCatalyst

> **Note**
>
> - 계정 연결을 끊으려면 먼저 GitHub 해당 계정에서 연결된 GitHub 모든 리포지토리의 연결을 해제해야 합니다.
> - Bitbucket 작업 영역의 연결을 끊으려면 먼저 해당 작업 영역에서 연결된 모든 Bitbucket 리포지토리의 연결을 해제해야 합니다.
> - GitLab 사용자 연결을 끊으려면 먼저 해당 작업 영역에서 연결된 모든 프로젝트 리포지토리의 연결을 해제해야 합니다. GitLab
> - Jira 사이트의 연결을 끊으려면 먼저 해당 계정에서 연결된 모든 Jira 프로젝트의 연결을 해제해야 합니다.
>
> 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

GitHub 프로젝트, Bitbucket 작업 영역, 사용자 또는 Jira 사이트의 연결을 끊으려면 GitLab

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.

2. CodeCatalyst 스페이스로 이동하세요.
3. 스페이스에 설치된 확장 프로그램 목록을 보려면 다음 중 하나를 수행하십시오.

   a. [설정] 을 선택한 다음 [설치된 확장 프로그램] 을 선택합니다.

   b. 상단 메뉴에서 카탈로그 아이콘을

      선택합니다.
4. 구성하려는 확장 (리포지토리, GitHub Bitbucket 리포지토리, 리포지토리 또는 Jira Software) 중 하나에 대해 구성을 선택합니다. GitLab
5. 구성하기로 선택한 타사 확장 프로그램에 따라 다음 중 하나를 수행하십시오.

   - GitHub 리포지토리: 계정과의 연결을 끊습니다. GitHub

     연결된 GitHub 계정 탭에서 연결을 끊을 계정을 선택한 다음 GitHub 계정 연결 해제를 선택합니다. GitHub
   - Bitbucket 리포지토리: Bitbucket 작업 공간과의 연결을 끊습니다.

     연결된 Bitbucket 작업 영역 탭에서 연결을 끊을 Bitbucket 작업 영역을 선택한 다음 Bitbucket 작업 영역 연결 해제를 선택합니다.
   - GitLab 리포지토리: 사용자와의 연결을 끊습니다. GitLab

     연결된 사용자 탭에서 연결을 끊을 GitLab 사용자를 선택한 다음 GitLab 사용자 연결 해제를 선택합니다. GitLab
   - Jira 소프트웨어: Jira 사이트와의 연결을 끊습니다.

     연결된 Jira 사이트 탭에서 연결을 끊으려는 Jira 사이트를 선택한 다음 Jira 사이트 연결 해제를 선택합니다.
6. 연결 끊기 대화 상자에서 계정 연결 해제의 영향을 검토하세요.
7. 텍스트 입력 필드에 연결 해제를 입력한 다음 연결 해제를 선택합니다.

GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, GitLab Jira 프로젝트 연결 CodeCatalyst

GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 사용하거나 Jira 프로젝트를 관리하려면 먼저 리포지토리 또는 프로젝트가 속한 타사 소스를 스페이스에 연결해야 합니

다. CodeCatalyst 자세한 내용은 GitHub 계정, Bitbucket 작업 영역, GitLab 사용자, Jira 사이트 연결 CodeCatalyst 단원을 참조하십시오.

연결된 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 워크플로에 사용할 수 있습니다. 워크플로우에서는 연결된 리포지토리의 이벤트가 워크플로 구성에 따라 코드를 빌드, 테스트 또는 배포할 수 있는 워크플로를 시작합니다. 연결된 리포지토리 GitHub 또는 Bitbucket 리포지토리를 사용하는 워크플로의 워크플로 구성 파일은 연결된 리포지토리에 저장됩니다. 연결된 리포지토리를 Dev Environments와 함께 사용하여 연결된 리포지토리에서 파일을 생성, 업데이트 및 삭제할 수도 있습니다. GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 리포지토리 확장의 세부 정보 페이지 또는 CodeCatalyst 프로젝트 GitHub 자체의 코드에 있는 소스 리포지토리 보기에서 리포지토리, Bitbucket 리포지토리 또는 프로젝트 GitLab 리포지토리를 프로젝트에 연결할 수 있습니다.

Important

기여자로서 GitHub 또는 Bitbucket 리포지토리를 연결할 수 있지만, 타사 리포지토리는 스페이스 관리자 또는 프로젝트 관리자로만 연결을 해제할 수 있습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

Important

리포지토리 확장을 설치한 후 연결하는 모든 리포지토리는 해당 코드를 CodeCatalyst 인덱싱하여 저장합니다. CodeCatalyst 그러면 코드를 검색할 수 있게 됩니다. CodeCatalyst 에서 CodeCatalyst 연결된 리포지토리를 사용할 때의 코드 데이터 보호를 더 잘 이해하려면 Amazon CodeCatalyst User Guide의 데이터 보호를 참조하십시오.

Important

CodeCatalyst 연결된 리포지토리의 기본 브랜치 변경 감지를 지원하지 않습니다. 연결된 저장소의 기본 분기를 변경하려면 먼저 연결을 해제하고 기본 분기를 변경한 다음 다시 연결해야 합니다. CodeCatalyst
가장 좋은 방법은 리포지토리를 연결하기 전에 항상 최신 버전의 확장 프로그램을 사용하는 것입니다.

연결된 Jira 프로젝트를 사용하여 이슈를 관리하고 CodeCatalyst 풀 리퀘스트를 Jira 이슈에 연결할 수 있습니다. 풀 리퀘스트의 요약 상태와 관련 CodeCatalyst 워크플로 이벤트의 상태는 Jira 이슈에 반영됩니다.

Important

Jira 프로젝트를 프로젝트에 연결하려면 CodeCatalyst Space 관리자 또는 CodeCatalyst CodeCatalyst 프로젝트 관리자여야 합니다.

Note

- GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리는 스페이스의 한 CodeCatalyst 프로젝트에만 연결할 수 있습니다.
- 비어 있거나 보관된 GitHub 리포지토리, Bitbucket 리포지토리 또는 프로젝트 리포지토리는 프로젝트와 함께 사용할 수 없습니다. GitLab CodeCatalyst
- GitHub 리포지토리, Bitbucket 리포지토리 또는 프로젝트의 리포지토리와 이름이 같은 GitLab 리포지토리는 연결할 수 없습니다. CodeCatalyst
- GitHub 리포지토리 확장은 GitHub 엔터프라이즈 서버 리포지토리와 호환되지 않습니다.
- Bitbucket 리포지토리 확장은 Bitbucket 데이터 센터 리포지토리와 호환되지 않습니다.
- 리포지토리 확장은 자체 관리형 GitLab 프로젝트 리포지토리와 호환되지 않습니다. GitLab
- 연결된 리포지토리에서는 나를 위한 설명 쓰기 또는 댓글 요약 기능을 사용할 수 없습니다. 이러한 기능은 풀 리퀘스트 인에서만 사용할 수 있습니다. CodeCatalyst
- CodeCatalyst 프로젝트는 하나의 Jira 프로젝트에만 연결될 수 있습니다. Jira 프로젝트를 여러 CodeCatalyst 프로젝트에 연결할 수 있습니다.

주제

- 연결된 타사 제공업체의 리소스 연결
- CodeCatalyst 프로젝트 생성 중에 타사 리포지토리 연결

연결된 타사 제공업체의 리소스 연결

확장 세부정보 페이지에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리 또는 Jira CodeCatalyst 프로젝트를 프로젝트에 연결하려면 GitLab

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.
3. 스페이스에 설치된 확장 프로그램 목록을 보려면 다음 중 하나를 수행하십시오.

   a. 설정을 선택한 다음 설치된 확장을 선택합니다.

   b. 상단 메뉴에서 카탈로그 아이콘을

   선택합니다.
4. 리포지토리, GitHub Bitbucket 리포지토리, 리포지토리 또는 Jira Software와 같은 확장 중 하나에 대해 구성을 선택합니다. GitLab
5. 구성하기로 선택한 타사 확장 프로그램에 따라 다음 중 하나를 수행하십시오.

   - GitHub 리포지토리: 리포지토리를 연결합니다. GitHub

     1. 연결된 GitHub 리포지토리 탭에서 리포지토리 연결을 선택합니다. GitHub
     2. GitHub 계정 드롭다운에서 연결하려는 저장소가 포함된 GitHub 계정을 선택합니다.
     3. GitHub 리포지토리 드롭다운에서 프로젝트에 연결하려는 리포지토리를 선택합니다. CodeCatalyst

        > **Tip**
        > 저장소 이름이 회색으로 표시된 경우 해당 저장소는 스페이스의 다른 프로젝트에 이미 링크되어 있으므로 해당 저장소를 링크할 수 없습니다.

     4. (선택 사항) 리포지토리 목록에 GitHub 리포지토리가 없다면 Amazon CodeCatalyst 애플리케이션에서 리포지토리에 액세스할 수 있도록 구성되지 않은 것일 수 있습니다. GitHub 연결된 CodeCatalyst 계정에서 사용할 수 있는 GitHub 리포지토리를 구성할 수 있습니다.

        a. GitHub계정으로 이동하여 설정을 선택한 다음 애플리케이션을 선택합니다.

        b. 설치된 GitHub 앱 탭에서 Amazon CodeCatalyst 애플리케이션에 맞게 구성을 선택합니다.

c. 다음 중 하나를 수행하여 연결하려는 GitHub 리포지토리에 대한 액세스를 구성하십시오. CodeCatalyst

- 현재 및 미래의 모든 리포지토리에 대한 액세스를 제공하려면 모든 리포지토리를 선택합니다.
- 특정 리포지토리에 대한 액세스를 제공하려면 리포지토리만 선택을 선택하고 리포지토리 선택 드롭다운을 선택한 다음 연결을 허용할 리포지토리를 선택합니다. CodeCatalyst

5. CodeCatalyst 프로젝트 드롭다운 메뉴에서 리포지토리를 연결하려는 CodeCatalyst 프로젝트를 선택합니다. GitHub
6. 연결을 선택합니다.

에서 GitHub CodeCatalyst 리포지토리를 더 이상 사용하지 않으려면 프로젝트에서 리포지토리를 연결 해제할 수 있습니다. CodeCatalyst 리포지토리의 연결이 해제되면 해당 리포지토리의 이벤트가 워크플로를 시작하지 않으므로 해당 리포지토리를 CodeCatalyst Dev Environments와 함께 사용할 수 없습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

- 비트버킷 리포지토리: Bitbucket 리포지토리를 연결합니다.

1. 연결된 비트버킷 리포지토리 탭에서 Bitbucket 리포지토리 연결을 선택합니다.
2. Bitbucket 작업 영역 드롭다운에서 연결하려는 리포지토리가 포함된 Bitbucket 작업 영역을 선택합니다.
3. Bitbucket 리포지토리 드롭다운에서 프로젝트에 연결하려는 리포지토리를 선택합니다. CodeCatalyst

> **Tip**
>
> 리포지토리 이름이 회색으로 표시된 경우 해당 리포지토리가 스페이스의 다른 프로젝트에 이미 연결되어 있기 때문에 해당 리포지토리를 연결할 수 없습니다.

4. CodeCatalyst 프로젝트 드롭다운 메뉴에서 Bitbucket 리포지토리를 연결할 CodeCatalyst 프로젝트를 선택합니다.
5. 연결을 선택합니다.

에서 CodeCatalyst Bitbucket 리포지토리를 더 이상 사용하지 않으려면 프로젝트에서 Bitbucket 리포지토리를 연결 해제할 수 있습니다. CodeCatalyst 리포지토리의 연결이 해제되면 해

당 리포지토리의 이벤트가 워크플로우 실행을 시작하지 않으므로 해당 리포지토리를 Dev Environments와 함께 CodeCatalyst 사용할 수 없습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

- GitLab 리포지토리: 프로젝트 리포지토리를 연결합니다. GitLab

  1. 연결된 GitLab 프로젝트 리포지토리 탭에서 프로젝트 리포지토리 연결을 선택합니다. GitLab
  2. GitLab 사용자 드롭다운에서 연결하려는 프로젝트 저장소가 들어 있는 GitLab 사용자를 선택합니다.
  3. GitLab 프로젝트 리포지토리 드롭다운에서 프로젝트에 연결하려는 리포지토리를 선택합니다. CodeCatalyst

     > **Tip**
     >
     > 저장소 이름이 회색으로 표시된 경우 해당 저장소는 스페이스의 다른 프로젝트에 이미 연결되어 있으므로 해당 저장소를 연결할 수 없습니다.

  4. CodeCatalyst 프로젝트 드롭다운 메뉴에서 CodeCatalyst 프로젝트 저장소를 연결할 GitLab 프로젝트를 선택합니다.
  5. 연결을 선택합니다.

  에서 CodeCatalyst 프로젝트 리포지토리를 더 이상 사용하지 않으려면 GitLab 프로젝트에서 프로젝트 리포지토리를 연결 해제할 수 있습니다. CodeCatalyst 프로젝트 리포지토리의 연결이 해제되면 해당 프로젝트 리포지토리의 이벤트가 워크플로우 실행을 시작하지 않으므로 해당 프로젝트 리포지토리를 CodeCatalyst Dev Environments와 함께 사용할 수 없습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

- Jira 소프트웨어: Jira 프로젝트를 연결합니다.

  1. 연결된 Jira 프로젝트 탭에서 Jira 프로젝트 연결을 선택합니다.
  2. Jira 사이트 드롭다운 메뉴에서 연결하려는 프로젝트가 포함된 Jira 사이트를 선택합니다.
  3. Jira 프로젝트 드롭다운 메뉴에서 프로젝트에 연결하려는 프로젝트를 선택합니다. CodeCatalyst
  4. CodeCatalyst 프로젝트 드롭다운 메뉴에서 Jira CodeCatalyst 프로젝트에 연결하려는 프로젝트를 선택합니다.

5. 연결을 선택합니다.

Jira 프로젝트가 프로젝트에 연결되면 CodeCatalyst 이슈에 대한 액세스가 완전히 비활성화되고 CodeCatalyst 탐색 창의 이슈가 Jira CodeCatalyst 프로젝트로 연결되는 Jira 이슈 항목으로 대체됩니다.

에서 CodeCatalyst 더 이상 Jira 프로젝트를 사용하지 않으려면 프로젝트에서 Jira 프로젝트를 연결 해제할 수 있습니다. CodeCatalyst Jira 프로젝트가 연결 해제되면 프로젝트에서 Jira 이슈를 사용할 수 없게 되며 Issues는 CodeCatalyst 다시 CodeCatalyst 이슈 제공자가 됩니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

프로젝트의 소스 GitHub 리포지토리 페이지에서 리포지토리, Bitbucket 리포지토리 또는 GitLab CodeCatalyst 프로젝트 리포지토리를 프로젝트에 연결하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 프로젝트로 이동합니다.
3. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
4. 리포지토리 추가를 선택한 다음 리포지토리 연결을 선택합니다.
5. 리포지토리 제공자 드롭다운 메뉴에서 다음 타사 리포지토리 제공자 중 하나를 선택합니다. GitHub, Bitbucket,. GitLab
6. 연결하기로 선택한 타사 저장소 공급자에 따라 다음 중 하나를 수행하십시오.

   - GitHub 리포지토리: 리포지토리를 연결합니다. GitHub

     1. GitHub 계정 드롭다운 메뉴에서 연결하려는 저장소가 들어 있는 GitHub 계정을 선택합니다.
     2. GitHub 리포지토리 드롭다운 메뉴에서 프로젝트를 CodeCatalyst 연결하려는 GitHub 리포지토리를 선택합니다.

        > **Tip**
        >
        > 리포지토리 이름이 회색으로 표시된 경우 해당 리포지토리가 이미 CodeCatalyst Amazon의 다른 프로젝트에 연결되어 있기 때문에 해당 리포지토리를 연결할 수 없습니다.

3. (선택 사항) 리포지토리 목록에 GitHub 리포지토리가 없다면 Amazon CodeCatalyst 애플리케이션에서 리포지토리에 액세스할 수 있도록 구성되지 않은 것일 수 있습니다. GitHub 연결된 CodeCatalyst 계정에서 사용할 수 있는 GitHub 리포지토리를 구성할 수 있습니다.

   a. GitHub계정으로 이동하여 설정을 선택한 다음 애플리케이션을 선택합니다.

   b. 설치된 GitHub 앱 탭에서 Amazon CodeCatalyst 애플리케이션에 맞게 구성을 선택합니다.

   c. 다음 중 하나를 수행하여 연결하려는 GitHub 리포지토리에 대한 액세스를 구성하십시오. CodeCatalyst

      - 현재 및 미래의 모든 리포지토리에 대한 액세스를 제공하려면 모든 리포지토리를 선택합니다.
      - 특정 리포지토리에 대한 액세스를 제공하려면 리포지토리만 선택을 선택하고 리포지토리 선택 드롭다운을 선택한 다음 연결을 허용할 리포지토리를 선택합니다. CodeCatalyst

- Bitbucket 리포지토리: Bitbucket 리포지토리를 연결합니다.

  1. Bitbucket 작업 영역 드롭다운 메뉴에서 연결하려는 리포지토리가 포함된 Bitbucket 작업 영역을 선택합니다.
  2. Bitbucket 리포지토리 드롭다운 메뉴에서 프로젝트를 연결하려는 Bitbucket 리포지토리를 선택합니다. CodeCatalyst

  > **Tip**
  >
  > 리포지토리 이름이 회색으로 표시된 경우 해당 리포지토리가 이미 CodeCatalyst Amazon의 다른 프로젝트에 연결되어 있기 때문에 해당 리포지토리를 연결할 수 없습니다.

- GitLab 리포지토리: 프로젝트 리포지토리를 연결합니다. GitLab

  1. GitLab 사용자 드롭다운 메뉴에서 연결하려는 프로젝트 저장소가 들어 있는 GitLab 사용자를 선택합니다.
  2. GitLab 프로젝트 리포지토리 드롭다운 메뉴에서 프로젝트를 CodeCatalyst 연결하려는 GitLab 프로젝트 리포지토리를 선택합니다.

> **Tip**
>
> 프로젝트 리포지토리 이름이 회색으로 표시되면 해당 프로젝트 리포지토리가 이미 CodeCatalyst Amazon의 다른 프로젝트에 연결되어 있기 때문에 해당 프로젝트 리포지토리를 연결할 수 없습니다.

7. 연결을 선택합니다.

에서 CodeCatalyst 리포지토리, Bitbucket 리포지토리 또는 프로젝트 GitHub 리포지토리를 더 이상 사용하지 않으려면 프로젝트에서 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 연결 해제할 수 있습니다. CodeCatalyst 리포지토리의 연결이 해제되면 해당 리포지토리의 이벤트가 워크플로우 실행을 시작하지 않으므로 해당 리포지토리를 Dev Environments와 함께 CodeCatalyst 사용할 수 없습니다. 자세한 내용은 에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst 단원을 참조하십시오.

GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 프로젝트에 연결한 후 CodeCatalyst 워크플로 및 개발 환경에서 사용할 수 있습니다. CodeCatalyst Amazon Q Developer, 블루프린트 등과 함께 연결된 리포지토리를 사용할 수도 있습니다. 자세한 내용은 타사 리포지토리 이벤트 이후 자동으로 워크플로 실행 및 Dev Environment 생성 단원을 참조하세요.

Jira 프로젝트를 CodeCatalyst 프로젝트에 연결하고 풀 리퀘스트를 연결하면 의 업데이트가 Jira 프로젝트에 CodeCatalyst 반영됩니다. 풀 리퀘스트를 Jira 이슈에 연결하는 방법에 대한 자세한 내용은 을 참조하십시오. Jira 이슈를 CodeCatalyst 풀 리퀘스트에 연결 Jira에서 CodeCatalyst 이벤트를 보는 방법에 대한 자세한 내용은 을 참조하십시오. Jira 이슈 CodeCatalyst 이벤트 보기

CodeCatalyst 프로젝트 생성 중에 타사 리포지토리 연결

새 프로젝트를 만들 때 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 새 CodeCatalyst 프로젝트에 연결할 수 있습니다. CodeCatalyst 자세한 내용은 연결된 타사 리포지토리를 사용하여 프로젝트 만들기 단원을 참조하십시오.

에서 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, Jira GitLab 프로젝트 연결 해제 CodeCatalyst

에서 더 이상 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 사용하지 않거나 Jira 프로젝트를 관리하고 싶지 않은 경우 리포지토리 또는 프로젝트를 프로젝트에서 CodeCatalyst 연결 해제할 수 있습니다. CodeCatalyst

GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리의 연결을 해제해도 리포지토리가 삭제되거나 변경되지 않습니다. 연결된 리포지토리에 저장된 워크플로 구성 파일은 삭제되지 않습니다. 하지만 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리의 연결을 해제하면 해당 리포지토리의 이벤트가 더 이상 워크플로 실행을 시작하지 않으며 Dev Environments와 함께 리포지토리를 사용할 수 없습니다. GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 리포지토리 확장의 세부 정보 페이지나 CodeCatalyst 프로젝트 자체의 Code에 있는 소스 GitHub 리포지토리 보기에서 GitLab 리포지토리, Bitbucket 리포지토리 또는 프로젝트 리포지토리를 프로젝트에서 연결 해제할 수 있습니다.

Jira 프로젝트를 연결 해제해도 계획 항목이나 개발 정보를 포함한 프로젝트가 삭제되거나 변경되지는 않습니다. 하지만 Jira 프로젝트의 연결을 해제하면 프로젝트의 Jira 이슈를 더 이상 프로젝트에 연결할 수 없게 되며 Issues는 다시 CodeCatalyst 이슈 제공자가 됩니다. CodeCatalyst

> **Important**
>
> GitHub 리포지토리, Bitbucket 리포지토리 또는 Gitlab 프로젝트 리포지토리를 프로젝트에서 연결 해제하려면 스페이스 관리자 또는 CodeCatalyst 프로젝트 관리자여야 합니다.

확장 세부정보 페이지에서 프로젝트의 GitHub 리포지토리, Bitbucket 리포지토리, GitLab 프로젝트 리포지토리 또는 Jira 프로젝트의 연결을 해제하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.
3. 스페이스에 설치된 확장 프로그램 목록을 보려면 다음 중 하나를 수행하십시오.
   a. 설정을 선택한 다음 설치된 확장을 선택합니다.
   b. 상단 메뉴에서 카탈로그 아이콘을

      선택합니다.

4. 구성하려는 확장 (리포지토리, GitHub Bitbucket 리포지토리, 리포지토리 또는 Jira Software) 중 하나에 대해 구성을 선택합니다. GitLab
5. 구성하기로 선택한 타사 확장 프로그램에 따라 다음 중 하나를 수행하십시오.

   - GitHub 리포지토리: 리포지토리 연결을 해제합니다. GitHub

     GitHub 리포지토리 탭에서 연결을 해제하려는 리포지토리를 선택한 다음 GitHub 리포지토리 연결 해제를 선택합니다. GitHub
   - Bitbucket 리포지토리: Bitbucket 리포지토리의 연결을 해제합니다.

     Bitbucket 리포지토리 탭에서 연결을 해제하려는 Bitbucket 리포지토리를 선택한 다음 Bitbucket 리포지토리 연결 해제를 선택합니다.
   - GitLab 리포지토리: 프로젝트 리포지토리의 연결을 해제합니다. GitLab

     GitLab 프로젝트 리포지토리 탭에서 연결 해제하려는 프로젝트 리포지토리를 선택한 다음 GitLab 프로젝트 리포지토리 연결 해제를 선택합니다. GitLab
   - Jira 소프트웨어: Jira 프로젝트의 연결을 해제합니다.

     Jira 프로젝트 탭에서 연결 해제하려는 Jira 프로젝트를 선택한 다음 Jira 프로젝트 연결 해제를 선택합니다.
6. 연결 해제 대화 상자에서 리포지토리 연결 해제의 영향을 검토하십시오.
7. 텍스트 입력 필드에 연결 해제를 입력하고 연결 해제를 선택합니다.

프로젝트의 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 소스 리포지토리 페이지에서 CodeCatalyst 연결 해제하려면

1. https://codecatalyst.aws/ 에서 콘솔을 엽니다. CodeCatalyst
2. CodeCatalyst 프로젝트로 이동합니다.
3. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.
4. 연결을 해제하려는 저장소의 라디오 버튼을 선택한 다음 저장소 연결 해제를 선택합니다.
5. 대화 상자의 정보를 검토하십시오. 지침을 따른 다음 연결 해제를 선택하여 리포지토리 연결을 해제합니다.

타사 리포지토리 보기 및 Jira 이슈 검색 CodeCatalyst

리포지토리, GitHub Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 연결한 후 해당 리포지토리를 확인하여 리소스를 확인하고 구성할 수 있습니다. CodeCatalyst 에서 연결된 Jira 이슈를 검색할 수도 있습니다. CodeCatalyst

주제

에서 타사 리포지토리 보기 CodeCatalyst

GitLab 프로젝트의 소스 리포지토리 목록 또는 GitHub 리포지토리, Bitbucket 리포지토리 또는 리포지토리 확장 세부 정보 페이지에서 연결된 리포지토리, Bitbucket 리포지토리 또는 프로젝트 리포지토리를 볼 수 있습니다. GitHub GitLab 리포지토리 목록에서 CodeCatalyst 리포지토리를 선택해도 열리지는 않습니다. 대신 링크된 리포지토리의 코드를 보고 작업할 수 있는 타사 리포지토리 공급자에서 열립니다.

링크된 GitHub 리포지토리, Bitbucket 리포지토리 또는 프로젝트 리포지토리를 보려면 GitLab CodeCatalyst

1. CodeCatalyst https://codecatalyst.aws/ 에서 콘솔을 엽니다.
2. CodeCatalyst 프로젝트로 이동합니다.
3. 탐색 창에서 코드를 선택한 다음 소스 리포지토리를 선택합니다.

확장 세부 정보 페이지에서 연결된 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 보려면

1. CodeCatalyst https://codecatalyst.aws/ 에서 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동한 다음 설치된 확장 프로그램 탭을 선택합니다.
3. 보려는 타사 리포지토리에 따라 다음 중 하나를 수행하십시오.

    - GitHub 저장소에서 구성을 선택한 다음 연결된 저장소를 선택하면 스페이스의 프로젝트에 연결된 모든 GitHub GitHub 저장소를 볼 수 있습니다. CodeCatalyst CodeCatalyst

- Bitbucket 리포지토리에서 구성을 선택한 다음 연결된 Bitbucket 리포지토리를 선택하면 스페이스의 프로젝트에 연결된 모든 Bitbucket 리포지토리를 볼 수 있습니다. CodeCatalyst CodeCatalyst
- GitLab 리포지토리에서 구성을 선택한 다음 연결된 프로젝트 리포지토리를 선택하면 스페이스의 GitLab 프로젝트에 연결된 모든 프로젝트 리포지토리를 볼 수 있습니다. GitLab CodeCatalyst CodeCatalyst

프로젝트에 연결된 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리가 목록에 표시됩니다. CodeCatalyst GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 선택하여 타사 리포지토리 공급자에서 파일을 보고 편집할 수 있습니다.

> **Note**
>
> 워크플로가 소스 액션에서 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 사용하는 경우 시각적 편집기나 YAML 편집기에서 워크플로우를 CodeCatalyst 변경하면 자동으로 커밋되어 타사 리포지토리에 푸시됩니다. YAML

에서 Jira 이슈 검색 CodeCatalyst

Jira 프로젝트를 연결한 후 CodeCatalyst 글로벌 검색 창을 사용하여 연결된 Jira 프로젝트에서 이슈를 검색할 수 있습니다. 풀 리퀘스트에서 이슈에 CodeCatalyst 연결하면서 Jira 이슈를 검색할 수도 있습니다. Jira 이슈를 CodeCatalyst 풀 리퀘스트에 연결하는 방법에 대한 자세한 내용은 을 참조하십시오. Jira 이슈를 CodeCatalyst 풀 리퀘스트에 연결

연결된 Jira 프로젝트에서 Jira 이슈를 검색하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 프로젝트로 이동합니다.
3. 글로벌 검색 바에서 연결된 Jira 프로젝트에서 풀 리퀘스트에 연결하려는 이슈나 Jira 이슈를 검색하세요.

타사 리포지토리 이벤트 이후 자동으로 워크플로 실행

연결된 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 워크플로의 소스로 사용할 수 있습니다. 이 경우 연결된 리포지토리, Bitbucket 리포지토리 또는 프로젝트 GitHub 리포지토리에서 지정된 브랜치를 변경하면 워크플로가 자동으로 실행됩니다. GitLab

워크플로는 지속적 통합 및 지속적 전달 (CI/CD) 시스템의 일부로 코드를 빌드, 테스트 및 배포하는 방법을 설명하는 자동화된 절차입니다. 워크플로는 워크플로 실행 중에 수행할 일련의 단계 또는 조치를 정의합니다. 또한 워크플로는 워크플로를 시작하게 하는 이벤트 또는 트리거를 정의합니다. 워크플로를 설정하려면 CodeCatalyst 콘솔의 시각적 또는 YAML 편집기를 사용하여 워크플로 정의 파일을 만듭니다.

> **Tip**
>
> 프로젝트에서 워크플로를 사용하는 방법을 간단히 살펴보려면 청사진을 사용하여 프로젝트를 만들어 보세요. 각 블루프린트는 검토, 실행, 실험할 수 있는 작동하는 워크플로를 배포합니다.

연결된 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리를 사용하도록 워크플로를 구성하면 워크플로 구성 파일은 해당 리포지토리, Bitbucket 리포지토리 또는 프로젝트 GitHub 리포지토리에 저장됩니다. GitLab 워크플로 구성은 워크플로 이름, 트리거, 리소스, 아티팩트 및 작업을 정의하는 YAML 파일입니다. 워크플로 구성 파일에 대한 자세한 내용은 을 참조하십시오. 워크플로우 YAML 정의

워크플로 구성 파일은 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리의 `./codecatalyst/workflows/` 디렉터리에 있어야 합니다.

워크플로 편집기를 사용하여 워크플로를 만들고 구성할 수 있습니다. 자세한 내용은 워크플로우 시작하기 및 소스 리포지토리를 워크플로에 연결 단원을 참조하십시오.

트리거를 추가하여 워크플로 실행을 시작합니다.

코드가 사용자 GitHub 또는 Bitbucket 저장소의 지정된 브랜치로 푸시될 때 자동으로 실행을 시작하도록 CodeCatalyst 워크플로를 구성할 수 있습니다. 워크플로 실행을 자동으로 시작하려면 워크플로 구성 파일의 `Triggers` 섹션에 트리거를 추가하십시오.

예: 간단한 코드 푸시 트리거

다음 예제는 소스 리포지토리의 브랜치에 코드가 푸시될 때마다 워크플로 실행을 시작하는 트리거를 보여줍니다.

```
Triggers:
  - Type: PUSH
```

예: 간단한 풀 리퀘스트 트리거

다음 예제는 소스 리포지토리의 브랜치에 대해 풀 요청이 생성될 때마다 워크플로 실행을 시작하는 트리거를 보여줍니다.

```
Triggers:
  - Type: PULLREQUEST
    Events:
      - OPEN
```

자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 단원을 참조하십시오.

타사 리포지토리 공급자와의 IP 액세스 제한

규칙 또는 구성을 설정하여 IP 주소를 기반으로 GitHub 리포지토리, Bitbucket 리포지토리 또는 GitLab 프로젝트 리포지토리에 대한 액세스를 제한할 수 있습니다. 타사 공급자의 설정 또는 액세스 제어 기능을 통해 이 작업을 수행할 수 있습니다.

사용 중인 타사 저장소 공급자에 따라 다음 중 하나를 참조하십시오.

- Amazon CodeCatalyst GitHub 리포지토리 확장은 GitHub 엔터프라이즈 클라우드 IP 액세스 제한과 호환됩니다. 특정 IP 주소에 대한 액세스를 제한하도록 GitHub Enterprise Cloud 조직을 구성하는 경우 GitHub 앱이 허용 목록을 구성하도록 허용하여 IP 주소를 자동으로 CodeCatalyst 등록할 수도 있습니다. GitHub 또는 CodeCatalyst IP 주소를 수동으로 추가할 수도 있습니다.
- Amazon CodeCatalyst Bitbucket 리포지토리 확장 프로그램은 비트버킷 클라우드 프리미엄 액세스 제한과 호환됩니다. 특정 IP 주소에 대한 액세스를 제한하도록 Bitbucket Cloud Premium 작업 공간을 구성할 때 IP 주소 집합에 대한 IP 주소 또는 네트워크 블록을 허용 목록에 추가할 수도 있습니다.
- Amazon CodeCatalyst GitLab 리포지토리 확장은 GitLab IP 주소 제한과 호환됩니다. 특정 IP 주소에 대한 액세스를 제한하도록 GitLab Premium 또는 Ultimate 그룹을 구성하는 경우 IP 주소 집합에 대한 IP 주소 또는 네트워크 블록을 허용 목록에 추가할 수도 있습니다.

CodeCatalyst IP 주소가 타사 리포지토리의 허용 목록에 없는 경우 Amazon CodeCatalyst 앱은 타사 리포지토리에 액세스할 수 없습니다. 자세한 내용은 타사 리포지토리 확장에서 사용하는 IP 주소 단원을 참조하십시오.

타사 리포지토리 확장에서 사용하는 IP 주소

타사 확장 프로그램이 타사 리소스에 액세스할 때 사용하는 IP 주소는 다음과 같습니다.

- GitHub 리포지토리:

```
us-west-2
  52.32.242.246
  54.148.176.49
  35.164.118.94
eu-west-1
  34.241.64.10
  34.246.255.80
  3.248.38.7
```

- 비트버킷 리포지토리 및 리포지토리: GitLab

```
us-west-2
  35.160.210.199
  54.71.206.108
  54.71.36.205
eu-west-1
  34.242.64.82
  52.18.37.201
  54.77.75.62
```

워크플로우 실패 시 타사 병합 차단

GitHub 또는 Bitbucket 리포지토리를 연결한 후 풀 CodeCatalyst 요청을 위한 CodeCatalyst 워크플로를 추가할 수 있습니다. 마찬가지로 GitLab 프로젝트 리포지토리를 연결한 후 병합 요청에 CodeCatalyst 워크플로를 CodeCatalyst 추가할 수 있습니다. 특정 커밋에서 하나 이상의 워크플로가 실행될 수 있으며, 각 워크플로의 실행 CodeCatalyst 상태도 GitHub, Bitbucket 또는 GitLab 커밋 상태의 일부로 반영됩니다. 새 커밋이 푸시되면 Bitbucket 또는 해당 새 커밋에 GitHub GitLab 대해 새 워크플로 실행 상태가 반영됩니다. 커밋을 위해 워크플로를 다시 실행하면 새 워크플로 실행 상태가 해당 커밋 및 워크플로의 이전 상태보다 우선합니다.

GitHub 또는 Bitbucket에서 브랜치 보호 규칙을 설정하여 풀 요청 병합을 차단하거나, 가장 최근 커밋이 워크플로 실행 실패 상태인 경우 병합 요청을 GitLab 차단하도록 In을 설정할 수 있습니다. 브랜치 보호 규칙을 사용하는 경우, 최신 커밋의 상태는 Bitbucket 또는 에서 GitHub 풀 요청을 병합하는 기능

에 영향을 줍니다. GitLab 워크플로에 대한 자세한 내용은 워크플로 실행 및 을 참조하십시오. 트리거를 사용하여 자동으로 워크플로 실행 시작

사용 중인 타사 저장소 공급자에 따라 다음을 참조하십시오.

- GitHub 리포지토리: GitHub 의 설명서 상태 확인 정보 및 보호된 브랜치에 대한 정보.
- Bitbucket 리포지토리: Bitbucket Cloud의 브랜치 권한 사용 및 브랜치 권한을 통한 제어에 대한 Bitbucket 설명서.
- GitLab 리포지토리: 자동 병합 및 보호 브랜치에 대한 GitLab 설명서입니다.

Jira 이슈를 CodeCatalyst 풀 리퀘스트에 연결

CodeCatalyst 소스 리포지토리에서 생성된 풀 요청을 Jira 이슈에 연결할 수 있습니다. Jira 이슈를 연결하면 이슈가 pull 요청의 속성으로 표시됩니다. 결과적으로 풀 리퀘스트 이벤트, 워크플로 이벤트, 배포 이벤트가 Jira로 전송되고 Jira 이슈에 추가됩니다. 풀 리퀘스트는 하나 이상의 Jira 이슈에 연결될 수 있습니다. CodeCatalyst 소스 리포지토리에 있는 풀 리퀘스트만 연결할 수 있으며, 타사 리포지토리와 같은 GitHub 풀 리퀘스트에는 연결할 수 없습니다. Jira 이슈를 풀 리퀘스트에 연결하려면 먼저 Jira 프로젝트를 프로젝트에 연결해야 합니다. CodeCatalyst Jira 프로젝트를 프로젝트에 연결하는 방법에 대한 자세한 내용은 을 CodeCatalyst 참조하십시오. GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, GitLab Jira 프로젝트 연결 CodeCatalyst

> **Note**
>
> CodeCatalyst 프로젝트에 브랜치가 두 개 있는 소스 리포지토리가 없으면 pull 요청을 생성할 수 없습니다. 풀 리퀘스트에 대한 자세한 내용은 풀 리퀘스트를 사용하여 작업하기를 참조하십시오 CodeCatalyst.

Jira 이슈를 CodeCatalyst 풀 리퀘스트에 연결하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 프로젝트로 이동합니다.
3. 탐색 창에서 코드를 선택한 다음 풀 요청을 선택합니다.
4. 풀 리퀘스트 생성을 선택하여 풀 리퀘스트 세부 정보를 입력합니다.
5. 소스 리포지토리 드롭다운 메뉴에서 풀 요청을 연결하려는 소스 리포지토리를 선택합니다.
6. 소스 브랜치 드롭다운 메뉴에서 검토하려는 변경 사항이 포함된 브랜치를 선택합니다.

7. 대상 브랜치 드롭다운 메뉴에서 검토한 변경 사항을 병합하려는 브랜치를 선택합니다.
8. 풀 리퀘스트 제목 텍스트 입력 필드에 풀 리퀘스트의 제목을 입력합니다.
9. Jira 이슈에 대한 이슈 연결 - 선택 필드를 선택하고 드롭다운을 선택한 다음 연결된 Jira 프로젝트에서 추가하려는 Jira 이슈를 검색합니다.
10. 풀 리퀘스트에 추가하려는 Jira 이슈를 선택합니다.
11. [Create] 를 선택하여 풀 리퀘스트를 생성합니다.

Jira 이슈를 풀 리퀘스트에 연결하면 CodeCatalyst 풀 리퀘스트의 요약을 사용할 수 있습니다. 요약에는 워크플로 실행, 연결된 이슈, 필수 검토자, 선택적 검토자, 작성자가 포함됩니다.

> **Note**
>
> Jira 이슈와 관련된 담당자 및 작성자 정보는 에서 사용할 수 없습니다. CodeCatalyst

풀 리퀘스트를 연결하면 동기화된 프로젝트와 Jira CodeCatalyst 프로젝트를 통해 업데이트를 Jira 프로젝트에 CodeCatalyst 반영할 수 있습니다. 연결된 풀 리퀘스트의 상태 및 풀 요청과 관련된 모든 워크플로 이벤트는 Jira에서 볼 때 Jira 이슈에 표시됩니다. Jira에서 CodeCatalyst 이벤트를 보는 방법에 대한 자세한 내용은 을 참조하십시오. Jira 이슈 CodeCatalyst 이벤트 보기

Jira 이슈 CodeCatalyst 이벤트 보기

CodeCatalyst 프로젝트와 Jira 프로젝트가 연결되어 있는 경우 풀 리퀘스트의 요약 상태와 관련 CodeCatalyst 워크플로 이벤트의 상태가 Jira 이슈에 반영됩니다. 예를 들어 풀 리퀘스트를 종료하거나 병합하면 상태 업데이트가 Jira 이슈에 반영됩니다. CodeCatalyst CodeCatalyst CodeCatalyst 풀 리퀘스트와 관련된 워크플로 CI/CD 이벤트는 동기화되므로 워크플로 실행이 성공하면 Jira 이슈에도 전송됩니다.

Jira 이슈의 CodeCatalyst 이벤트를 보려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 프로젝트로 이동합니다.
3. CodeCatalyst 탐색 창에서 코드를 선택하고 풀 요청을 선택한 다음 Jira 프로젝트에서 보려는 Jira 이슈가 포함된 풀 리퀘스트를 선택합니다.
4. 추가 정보 패널에서 Jira 프로젝트에서 보려는 Jira 이슈를 선택합니다.

5. Jira 프로젝트의 세부 정보 패널에서 개발로 나열된 풀 요청을 선택하면 풀 요청의 세부 정보를 볼 수 있습니다.
6. (선택 사항) 최신 빌드를 보려면 Builds 탭을 선택합니다.
7. (선택 사항) 개발 상태를 보려면 배포 탭을 선택합니다.

에서 코드, 이슈, 프로젝트, 사용자 검색 CodeCatalyst

의 검색 창이나 전용 검색 결과 창을 사용하여 코드, 이슈, 프로젝트, 사용자를 검색하세요 CodeCatalyst. CodeCatalyst

검색 창에 이름, 설명, 상태 등의 쿼리를 입력하여 스페이스 및 프로젝트 전반의 리소스를 찾을 수 있습니다. 검색 쿼리 언어를 사용하여 검색 쿼리를 구체화할 수도 있습니다.

주제

- 검색 쿼리 구체화하기
- 검색 작업 시 고려 사항
- 검색 가능한 필드 참조

검색하려면

1. 상단 내비게이션 바의 검색 창에 검색어를 입력합니다.
2. (선택 사항) CodeCatalyst 의 검색어 언어를 사용하여 검색 쿼리를 세분화합니다. 자세한 정보는 검색 쿼리 구체화하기을 참조하세요.
3. 다음 중 하나를 수행하십시오.
   - 현재 참여하고 있는 프로젝트 내에서 리소스를 검색하려면 이 프로젝트를 선택합니다.
   - 현재 있는 스페이스의 모든 프로젝트에서 리소스를 검색하려면 이 스페이스를 선택합니다.
4. 다음 중 하나를 수행하여 전용 검색 결과 창에서 검색 결과를 확인하십시오.
   - 빠른 검색 결과 창 하단에서 projt-name | space-name으로 모든 결과 보기를 선택하여 모든 검색 결과를 확인합니다.
   - 모든 검색 결과를 보려면 Enter를 누르십시오.

> **Tip**
>
> 풀 리퀘스트 댓글이나 설명이나 이슈 댓글이나 설명에서 다른 프로젝트 사용자를 언급하세요. 이때 @ 기호 다음에 표시 이름이나 사용자 이름을 붙입니다. @ 기호 다음에 이슈 또는 코드 파일 이름을 사용하여 이슈 또는 코드 파일과 같은 리소스에 연결할 수도 있습니다.

검색 쿼리 구체화하기

검색 후에도 원하는 내용을 찾을 수 없는 경우 CodeCatalyst 의 특수 검색어 언어를 사용하여 검색을 세분화할 수 있습니다. 개별 필드에는 문자 제한이 없지만 전체 쿼리는 1,024자로 제한됩니다.

주제

유형별 상세 검색

검색 범위를 특정 유형의 정보로 구체화하려면 *type:result-type* 검색에 포함시키십시오. 여기서 *## ###,, ##* 입니다code. issue project user

예:

- type:code AND java— “java”가 포함된 코드 관련 필드에 코드 결과를 표시합니다.

  자세한 정보는 코드 필드을 참조하세요.
- type:issue AND Bug— “버그”가 포함된 이슈 관련 필드에 이슈 결과를 표시합니다.

  자세한 정보는 이슈 필드을 참조하세요.
- type:user AND MaryMajor— “" 가 포함된 사용자 관련 필드에 사용자 결과를 표시합니다. MaryMajor

  자세한 정보는 사용자 필드을 참조하세요.
- type:project AND Datafeeder— 'Datafeeder'가 포함된 프로젝트 결과를 표시합니다.

  자세한 정보는 프로젝트 필드을 참조하세요.

분야별 세분화

검색 범위를 특정 필드로 구체화하려면 *## ###,,,* 등인 검색에 *field-name:query* 포함하세요. *## #* 검색 대상 텍스트입니다. title username project description 필드 목록은 을 참조하십시오. 검색 가능한 필드 참조 괄호를 사용하여 여러 쿼리를 검색할 수 있습니다.

예:

- `title:bug`— 제목에 “버그”가 포함된 결과를 표시합니다.
- `username:John`— 사용자 이름에 “John”이 포함된 결과를 표시합니다.
- `project:DataFeeder`— 프로젝트 “DataFeeder" 에 결과를 표시합니다. 쿼리는 대소문자를 구분하지 않습니다.
- `description:overview`— 설명에 “개요”가 포함된 결과를 표시합니다.

부울 연산자를 사용한 구체화

검색 구문에 제약 조건을 지정하려면 부울 연산자AND, OR 및 를 사용할 수 있습니다. NOT 여러 구문을 나열하는 경우 기본적으로 해당 구문을 와 CodeCatalyst OR 결합합니다. 괄호를 사용하여 검색어를 그룹화할 수 있습니다.

- `exception AND type:code`— “예외”에 대한 코드 결과만 표시합니다.
- `path:README.md AND repo:ServerlessAPI`— 리포지토리 이름이 “서버리스API”인 “Readme.md”를 포함하는 경로에 대한 결과를 표시합니다.
- `buildspec.yml AND (repo:ServerlessAPI OR ServerlessWebApp)`— 리포지토리가 “서버리스 API” 또는 “" 인 경우 “buildspec.yml”에 대한 결과를 표시합니다. ServerlessWebApp
- `path:java NOT (path:py OR path:ts)`— 경로에 “java”는 포함되지만 “py” 또는 “ts”는 포함되지 않은 결과를 표시합니다.

프로젝트별 정제

검색 범위를 특정 프로젝트로 구체화하려면 *`project:name AND query`* 검색에 포함시키십시오. 여기서 *`###`* 검색 대상 프로젝트이고 *`###`* 검색 대상 콘텐츠입니다.

- `project:name AND query`— 경로에 쿼리와 프로젝트 이름이 포함된 결과를 표시합니다.

검색 작업 시 고려 사항

지연된 콘텐츠 업데이트 — 이름 변경이나 이슈 재배정과 같은 콘텐츠 업데이트가 검색 결과에 반영되는 데 몇 분 정도 걸릴 수 있습니다. 코드베이스 마이그레이션과 같은 대규모 업데이트는 검색결과에 표시되는 데 시간이 더 오래 걸릴 수 있습니다.

이스케이프 특수 문자 — 다음과 같은 특수 문자는 검색 쿼리에서 특별히 고려해야 합니다. `+ - & && || ! ( ) { } [ ] ^ " ~ * ? : \` 특수 문자는 쿼리에 영향을 주지 않으므로 제거하거나 이스케이프 처리해야 합니다. 문자를 이스케이프 처리하려면 문자 앞에 백슬래시 (\) 를 추가합니다. 예를 들어 [기능] 검색 쿼리는 기능 또는\ [기능\] 이어야 합니다.

검색 범위 좁히기 — 검색에서는 대소문자를 구분하지 않습니다. 모두 소문자로 검색하면 대소문자 변경 시 쿼리에서 단어가 분리되는 것을 방지할 수 있습니다. 예를 들어, 및만 쿼리하려면 또는 만 `MyService` 포함하는 결과가 나오지 않도록 쿼리를 `myservice` 사용해 보세요. `MyService my service`

검색은 기본적으로 OR별 연결을 사용하여 단어와 단어 일부를 결합합니다. 예를 들어, 와 를 모두 포함하는 결과를 `new function` 반환하고 또는 만 포함된 결과를 `new function` 반환할 수 있습니다. `new function` 후자를 피하려면 여러 단어를 다음과 같이 조합하십시오AND. 예를 들어, 검색할 수 있습니다`new AND function`.

기본 브랜치 - 검색은 소스 리포지토리의 기본 브랜치에서 최근에 커밋한 코드 결과만 반환합니다. 다른 브랜치나 커밋에서 코드를 찾으려면 리포지토리를 로컬에서 복제하거나, 개발 환경에서 브랜치를 열거나, UI에서 브랜치와 세부 정보를 보는 것을 고려해 보세요. CodeCatalyst 기본 브랜치를 변경하면 검색으로 검색할 수 있는 파일도 업데이트됩니다. 자세한 정보는 리포지토리의 기본 브랜치 관리을 참조하세요.

> **Important**
>
> CodeCatalyst 연결된 리포지토리의 기본 브랜치 변경 감지를 지원하지 않습니다. 연결된 저장소의 기본 분기를 변경하려면 먼저 연결을 해제하고 기본 분기를 변경한 다음 다시 연결해야 합니다. CodeCatalyst 자세한 정보는 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, GitLab Jira 프로젝트 연결 CodeCatalyst을 참조하세요.
> 가장 좋은 방법은 리포지토리를 연결하기 전에 항상 최신 버전의 확장 프로그램을 사용하는 것입니다.

검색 가능한 필드 참조

CodeCatalyst 검색 쿼리를 입력하면 다음 필드를 검색합니다. 별칭은 고급 쿼리 언어로 필드를 참조하는 데 사용할 수 있는 또 다른 이름입니다.

코드 필드

필드	별칭	설명
브랜치/이름	분기	코드 파일이 있는 지점의 이름
code	N/A	검색과 일치하는 소스 코드의 일부를 나타내는 코드 스니펫 형태의 코드 내용에 대한 정보입니다.
CommitId	N/A	반환된 코드 파일이 마지막으로 업데이트된 커밋의 커밋 ID입니다. 에서 지정한 브랜치 이름 끝에 있는 커밋 ID일 수도 있고 아닐 수도 있습니다. `branchName`
커밋 메시지	N/A	코드 파일이 마지막으로 업데이트된 커밋의 커밋 메시지입니다. 에서 지정한 브랜치 이름 끝에 있는 커밋 메시지일 수도 있고 아닐 수도 있습니다. `branchName` 커밋 메시지가 제공되지 않은 경우 이 값은 빈 문자열이 됩니다.
filePath	경로	이 코드 파일의 파일 경로입니다.
lastUpdatedBy	N/A	CodeCatalyst 코드 파일을 마지막으로 업데이트한 사용자 사용자 이름을 사용할 수 없는 경우 이 값은 Git 구성 파일에 구성된 사용자의 이메일 주소가 됩니다.

필드	별칭	설명
lastUpdatedBy아이디	N/A	코드 파일을 마지막으로 업데이트한 사용자의 시스템 생성 고유 ID. 사용자 ID를 사용할 수 없는 경우 이 값은 사용자의 이메일 주소일 수 있습니다.
lastUpdatedTime	N/A	코드 파일이 포함된 커밋으로 검색 데이터가 마지막으로 업데이트된 시간 (협정 세계시 (UTC) 타임스탬프).
projectId	N/A	시스템에서 생성한 프로젝트의 고유 ID.
projectName	프로젝트 이름, 프로젝트	코드 파일이 커밋된 소스 리포지토리가 포함된 프로젝트의 표시 이름.
리포지토리 ID	리포 ID	시스템에서 생성한 소스 리포지토리의 고유 ID.
repositoryName	리포지토리, 리포지토리	코드 파일이 커밋된 소스 리포지토리의 표시 이름.

이슈 필드

필드	별칭	설명
담당자 ID	담당자 ID	이슈에 배정된 사용자의 시스템에서 생성한 고유 ID.
양수인	양수인	이슈에 배정된 사용자의 사용자 이름
생성자	N/A	이슈를 생성한 사용자의 디스플레이 이름.

필드	별칭	설명
createdById	N/A	이슈를 생성한 사용자의 시스템 생성 고유 ID.
생성 시간	N/A	이슈가 생성된 시간 (협정 세계시 (UTC) 타임스탬프)
설명	N/A	문제에 대한 설명.
보관되었습니다.	archived	이슈를 보관된 상태로 생성할지 여부를 나타내는 부울 값입니다.
차단됨	차단됨	문제가 차단된 것으로 표시되었는지 여부를 나타내는 부울 값입니다.
레이블 ID	라벨 ID	시스템에서 생성한 문제 라벨의 고유 ID.
lastUpdatedBy	N/A	문제를 마지막으로 업데이트한 사용자의 이름을 표시합니다.
lastUpdatedBy아이디	N/A	시스템에서 생성한 문제를 마지막으로 업데이트한 사용자의 고유 ID입니다.
lastUpdatedTime	N/A	문제가 마지막으로 업데이트된 시간 (협정 세계시 (UTC) 타임스탬프 기준).
우선순위	N/A	문제의 우선 순위 (문제가 할당된 경우).
projectId	N/A	시스템에서 생성한 프로젝트의 고유 ID.

필드	별칭	설명
projectName	프로젝트 이름, 프로젝트	이 문제를 찾을 수 있는 프로젝트.
쇼트 ID	N/A	문제의 단축된 자동 증가 식별자입니다.
status	N/A	문제가 기내 백로그에 있는지 또는 컬럼에 있는지 여부를 나타내는 문제 상태.
상태 ID	N/A	상태의 시스템 식별자입니다.
title	N/A	문제 제목.

프로젝트 필드

필드	별칭	설명
설명	N/A	프로젝트 설명.
lastUpdatedTime	N/A	프로젝트 메타데이터가 마지막으로 업데이트된 시간 (협정 세계시 (UTC) 타임스탬프 기준).
projectName	project	스페이스에 있는 프로젝트의 이름.
프로젝트 경로	N/A	프로젝트 생성 중에 정의된 URL 라우팅 가능한 프로젝트 이름. 프로젝트 이름이 필요한 URL에 사용됩니다.

사용자 필드

필드	별칭	설명
displayName	N/A	에서 사용자가 사용하는 이름 CodeCatalyst. 표시 이름은 고유하지 않습니다.
이메일	N/A	사용자의 이메일 주소.
lastUpdatedTime	N/A	사용자 메타데이터가 마지막으로 업데이트된 시간 (협정 세계시 (UTC) 타임스탬프).
사용자 이름	사용자 이름	사용자가 가입할 때 선택한 사용자 이름. CodeCatalyst 표시 이름과 달리 사용자 이름은 변경할 수 없습니다.

아마존 문제 해결 CodeCatalyst

다음 정보는 에서 흔히 발생하는 문제를 해결하는 데 도움이 될 수 있습니다. CodeCatalyst Amazon CodeCatalyst 건강 보고서를 사용하여 경험에 영향을 줄 수 있는 서비스 문제가 있는지 확인할 수도 있습니다.

주제

일반 액세스 문제 해결

암호를 잊어버렸습니다

문제: AWS 빌더 ID와 CodeCatalyst Amazon에 사용하는 비밀번호를 잊어버렸습니다.

가능한 해결 방법: 이 문제를 해결하는 가장 쉬운 방법은 비밀번호를 재설정하는 것입니다.

1. CodeCatalystAmazon을 열고 이메일 주소를 입력합니다. 그런 다음 계속을 선택합니다.
2. Forgot Password?를 선택합니다.

3. 비밀번호를 변경할 수 있는 링크가 포함된 이메일을 보내드립니다. 받은 편지함에 이메일이 보이지 않으면 스팸 폴더를 확인하세요.

CodeCatalyst Amazon의 일부 또는 전체를 사용할 수 없음

문제: 콘솔로 이동하거나 CodeCatalyst 콘솔로 연결되는 링크를 따라갔는데 오류가 표시됩니다.

가능한 해결 방법: 이 문제가 발생하는 가장 일반적인 이유는 초대를 받지 않은 프로젝트 또는 스페이스의 링크를 클릭했거나 서비스에 일반적인 가용성 문제가 있기 때문입니다. Health 보고서를 확인하여 서비스에 알려진 문제가 있는지 확인하세요. 그렇지 않은 경우 프로젝트 또는 스페이스에 초대한 사람에게 연락하여 다른 초대를 요청하세요. 아직 어떤 프로젝트나 스페이스에 초대받지 못했다면 등록하여 자신만의 스페이스와 프로젝트를 만들 수 있습니다.

에서는 프로젝트를 생성할 수 없습니다. CodeCatalyst

문제: 프로젝트를 만들고 싶은데 프로젝트 만들기 버튼이 사용할 수 없는 것으로 표시되거나 오류 메시지가 나타납니다.

가능한 해결 방법: 이 문제가 발생하는 가장 일반적인 이유는 스페이스 관리자 역할이 없는 AWS 빌더 ID로 콘솔에 로그인했기 때문입니다. 스페이스에서 프로젝트를 만들려면 이 역할이 있어야 합니다.

이 역할을 가지고 있는데 버튼이 사용 가능한 것으로 표시되지 않는 경우 서비스에 일시적인 문제가 있을 수 있습니다. 브라우저를 새로 고치고 다시 시도하세요.

지원 문제 해결

AWS Support Amazon에 액세스할 때 오류가 발생합니다. CodeCatalyst

문제: AWS Support Amazon용 CodeCatalyst 옵션을 선택하면 다음 오류 메시지가 나타납니다.

```
Unable to assume role

To access support cases, you must add the role
AWSRoleForCodeCatalystSupport to the AWS ## that is the billing account for
the space.
```

가능한 해결 방법: 스페이스의 AWS 계정 결제 계정인 필수 역할을 추가합니다. 스페이스의 결제 계정으로 지정된 계정은 `AWSRoleForCodeCatalystSupport` 역할 및 `AmazonCodeCatalystSupportAccess` 관리형 정책을 사용합니다. 자세한 내용은 계정 및 스페이스에 대한 AWSRoleForCodeCatalystSupport역할 생성 단원을 참조하십시오.

> **Note**
>
> AWS 빌더 ID는 인증된 별칭과 권한 기반 리소스에 대한 지원만 받을 수 있습니다. CodeCatalyst 계정 및 청구 지원은 스페이스에 있는 모든 사용자에게 제공됩니다. 하지만 빌더는 권한이 있는 리소스 및 정보에 대한 지원만 받을 수 있습니다. CodeCatalyst

제 숙소에 대한 기술 지원 사례를 만들 수 없습니다.

문제: 내 공간에 대한 기술 지원 사례를 생성할 수 없습니다.

수정 사항: 스페이스의 사용자가 기술 지원 사례를 생성하려면 스페이스 결제 계정에 Business Support 또는 Enterprise Support 플랜을 추가해야 합니다. 스페이스 관리자에게 스페이스 빌링 계정에 AWS Support 플랜을 추가해 달라고 요청하거나 https://repost.aws/ 을 방문하여 AWS 커뮤니티에 문의하세요.

지원 사례를 위한 제 계정이 제 공간에 더 이상 연결되어 있지 않습니다. CodeCatalyst

문제: 지원 사례를 위한 내 계정이 내 스페이스에 더 이상 연결되어 있지 않습니다 CodeCatalyst.

해결 방법: 스페이스 관리자 역할을 가진 사용자가 스페이스 결제 계정을 전환하면 AWS Support 플랜 및 모든 관련 사례의 스페이스 연결이 끊깁니다. 이전 스페이스 결제 계정과 관련된 AWS Support 사례는 더 이상 AWS Support for Amazon에서 볼 수 없습니다 CodeCatalyst. 해당 결제 계정의 루트 사용자는 에서 오래된 사례를 보고 해결할 수 AWS Management Console 있으며 다른 사용자가 이전 사례를 보고 해결할 수 있는 IAM 권한을 설정할 수 있습니다. AWS Support 를 통해 기존 Space 결제 계정에 대한 CodeCatalyst 기술 지원을 계속 받을 수는 없지만 AWS Support 플랜이 취소되기 전까지는 다른 서비스에 대한 기술 지원을 받을 수 있습니다. AWS Management Console

자세한 내용은 AWS Support 사용 설명서의 케이스 업데이트, 해결 및 재개를 참조하십시오.

AWS 서비스 Amazon에서AWS Support 다른 지원 케이스에 대한 지원 케이스를 열 수 없습니다. CodeCatalyst

문제: AWS 서비스 for에서 AWS Support 다른 지원 케이스에 대한 지원 케이스를 열 수 없습니다 CodeCatalyst.

가능한 해결 방법: for에서만 CodeCatalyst 지원 사례를 열 수 CodeCatalyst 있습니다. AWS Support 다른 서비스 AWS, Amazon 또는 기타 타사 서비스에 배포한 서비스 또는 리소스에 대한 지원이 필

요한 경우, AWS Management Console 또는 타사 서비스 지원 채널을 통해 사례를 생성해야 합니다. CodeCatalyst 자세한 내용은 AWS Support 사용 설명서의 지원 사례 및 사례 관리 생성을 참조하십시오.

CodeCatalyst Amazon의 일부 또는 전체를 사용할 수 없음

문제: 콘솔로 이동하거나 CodeCatalyst 콘솔로 연결되는 링크를 따라갔는데 오류가 표시됩니다.

가능한 해결 방법: 이 문제가 발생하는 가장 일반적인 이유는 초대를 받지 않은 프로젝트 또는 스페이스의 링크를 클릭했거나 서비스에 일반적인 가용성 문제가 있기 때문입니다. Health 보고서를 확인하여 서비스에 알려진 문제가 있는지 확인하세요. 그렇지 않은 경우 프로젝트 또는 스페이스에 초대한 사람에게 연락하여 다른 초대를 요청하세요. 아직 어떤 프로젝트나 스페이스에 초대받지 못했다면 등록하여 자신만의 스페이스와 프로젝트를 만들 수 있습니다.

에서는 프로젝트를 생성할 수 없습니다. CodeCatalyst

문제: 프로젝트를 만들고 싶은데 프로젝트 만들기 버튼이 사용할 수 없는 것으로 표시되거나 오류 메시지가 나타납니다.

가능한 해결 방법: 이 문제가 발생하는 가장 일반적인 이유는 스페이스 관리자 역할이 없는 AWS 빌더 ID로 콘솔에 로그인했기 때문입니다. 스페이스에서 프로젝트를 만들려면 이 역할이 있어야 합니다.

이 역할을 가지고 있는데 버튼이 사용 가능한 것으로 표시되지 않는 경우 서비스에 일시적인 문제가 있을 수 있습니다. 브라우저를 새로 고치고 다시 시도하세요.

사용자 이름이 잘려서 새 사용자로 내 BID 스페이스에 액세스할 수 없거나 새 SSO 사용자로 추가할 수 없습니다.

문제: 사용자 이름을 100자 이하로 CodeCatalyst 잘라서 일부 사용자 이름이 동일하게 표시될 수 있습니다. 새 사용자가 스페이스에 액세스하는 경우 CodeCatalyst 스페이스 유형에 따라 다음과 같은 문제가 발생합니다.

- 로그인하는 데 사용하려는 AWS 빌더 ID가 CodeCatalyst 있습니다. 스페이스에 로그인하려고 하면 사용자 이름이 유효하지 않다는 메시지가 나타납니다.
- 저는 ID 페더레이션을 지원하는 CodeCatalyst 스페이스의 페더레이션 ID 관리자입니다. IAMIdentity Center의 사용자 및 그룹에 새 SSO 사용자를 추가할 때 해당 사용자가 유효하지 않다는 메시지가 나타납니다.

가능한 해결 방법: 잘린 사용자 이름을 가진 사용자로 스페이스에 처음 CodeCatalyst 로그인하거나 스페이스에 추가된 SSO 사용자가 성공합니다. AWS 빌더 ID로 가입하거나 그 이후에 IAM Identity Center에 추가된 사용자는 이름이 중복된 것으로 표시되므로 로그인할 수 없습니다. 공간 유형에 따라 다음 중 하나를 수행하십시오.

- AWS Builder ID 스페이스에 로그인하려면 다른 사용자 이름으로 가입하세요.
- IAMIdentity Center에 새 사용자를 추가하려면 다른 사용자 이름을 가진 사용자를 추가하십시오.

> **Note**
>
> 사용자 이름이 잘린 것처럼 보이더라도 이름이 잘린 사용자 이름의 영향을 받지 않는 방식으로 ID에 CodeCatalyst 매핑됩니다. 그러나 잘린 사용자 이름과 동일한 사용자 이름을 만들면 동일한 스페이스 또는 IAM Identity Center 응용 프로그램과 연결된 다른 사용자가 잘린 사용자 이름에 이미 CodeCatalyst 가입한 경우 해당 사용자 이름을 사용할 수 없습니다.

로 피드백을 제출하고 싶습니다. CodeCatalyst

문제: 에서 버그를 발견했는데 피드백을 CodeCatalyst 제출하고 싶습니다.

가능한 해결 방법: 에서 직접 피드백을 제출할 수 CodeCatalyst 있습니다.

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 탐색 창에서 피드백 제공을 선택합니다.
3. 드롭다운 메뉴에서 피드백 유형을 선택하고 피드백을 입력합니다.

소스 리포지토리 관련 문제 해결

다음 정보는 의 소스 리포지토리와 관련된 일반적인 문제를 해결하는 데 도움이 될 수 있습니다. CodeCatalyst

주제

- 내 공간이 최대 저장 공간에 도달했는데 경고 또는 오류가 표시됩니다.
- Amazon CodeCatalyst 소스 리포지토리로 복제하거나 푸시하려고 할 때 오류가 발생합니다.
- Amazon CodeCatalyst 소스 리포지토리로 커밋 또는 푸시하려고 할 때 오류가 발생합니다.
- 프로젝트를 위한 소스 리포지토리가 필요해요.

- 내 소스 리포지토리는 새 저장소인데 커밋이 들어 있습니다.
- 다른 브랜치를 기본 브랜치로 사용하고 싶습니다.
- 풀 리퀘스트 활동에 대한 이메일을 받고 있습니다.
- 개인 액세스 토큰 (PAT) 을 잊어버렸습니다.
- 풀 리퀘스트에는 예상한 변경 사항이 표시되지 않습니다.
- 풀 리퀘스트는 병합 불가 상태로 표시됩니다.

내 공간이 최대 저장 공간에 도달했는데 경고 또는 오류가 표시됩니다.

문제: 에 있는 하나 이상의 소스 리포지토리에 코드를 커밋하려고 하는데 오류가 표시됩니다. CodeCatalyst 콘솔의 소스 저장소 페이지에 공간 저장 한도에 도달했다는 메시지가 표시됩니다.

가능한 해결 방법: 프로젝트 또는 공간에서의 역할에 따라 하나 이상의 소스 리포지토리의 크기를 줄이거나, 사용하지 않는 소스 리포지토리를 삭제하거나, 청구 티어를 스토리지가 더 많은 리포지토리로 변경할 수 있습니다.

- 프로젝트에서 소스 리포지토리의 크기를 줄이려면 사용하지 않는 브랜치를 삭제하면 됩니다. 자세한 내용은 브랜치 삭제 및 기여자 역할 섹션을 참조하세요.
- 공간의 전체 스토리지를 줄이려면 사용하지 않는 소스 리포지토리를 삭제하면 됩니다. 자세한 내용은 소스 리포지토리 삭제 및 프로젝트 관리자 역할 섹션을 참조하세요.
- 공간에 사용할 수 있는 저장 용량을 늘리려면 청구 등급을 저장 용량이 더 많은 등급으로 변경할 수 있습니다. 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 CodeCatalyst 청구 등급 변경을 참조하십시오.

Amazon CodeCatalyst 소스 리포지토리로 복제하거나 푸시하려고 할 때 오류가 발생합니다.

문제: 소스 리포지토리를 로컬 컴퓨터나 통합 개발 환경 (IDE) 에 복제하려고 하면 권한 오류가 발생합니다.

가능한 해결 방법: AWS 빌더 ID용 개인 액세스 토큰 (PAT) 이 없거나, PAT로 자격 증명 관리 시스템을 구성하지 않았거나, PAT가 만료되었을 수 있습니다. 다음 해결 방법 중 하나 이상을 시도해 보세요.

- 개인용 액세스 토큰 (PAT) 을 만드세요. 자세한 정보는 개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여을 참조하세요.

- 소스 리포지토리가 포함된 프로젝트에 대한 초대를 수락했고 여전히 해당 프로젝트의 멤버인지 확인하세요. 프로젝트의 정회원이 아닌 경우 소스 리포지토리를 복제할 수 없습니다. 콘솔에 로그인하여 소스 리포지토리를 복제하려는 스페이스 및 프로젝트로 이동해 보세요. 스페이스의 프로젝트 목록에서 해당 프로젝트를 볼 수 없다면 해당 프로젝트의 구성원이 아니거나 프로젝트에 대한 초대를 수락하지 않은 것입니다. 자세한 정보는 초대 수락 및 AWS 빌더 ID 생성을 참조하세요.
- 복제 명령의 형식이 올바르고 AWS 빌더 ID가 포함되어 있는지 확인하세요. 예:

```
https://LiJuan@git.us-west-2.codecatalyst.aws/
v1/ExampleCorp/MyExampleProject/MyExampleRepo
```

- AWS CLI 를 사용하여 AWS 빌더 ID와 연결된 PAT가 있고 만료되지 않았는지 확인하세요. PAT가 없거나 PAT가 만료된 경우 새로 만드세요. 자세한 정보는 개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여을 참조하세요.
- 로컬 리포지토리나 IDE에 코드를 복제하는 대신 소스 리포지토리의 코드를 사용할 수 있는 개발 환경을 만들어 보세요. 자세한 정보는 Dev Environment 생성을 참조하세요.

Amazon CodeCatalyst 소스 리포지토리로 커밋 또는 푸시하려고 할 때 오류가 발생합니다.

문제: 소스 리포지토리로 푸시하려고 하면 권한 오류가 발생합니다.

가능한 해결 방법: 프로젝트에 코드 변경 사항을 커밋하고 푸시할 수 있는 역할이 프로젝트에 없을 수도 있습니다. 변경 내용을 소스 리포지토리로 푸시하려는 프로젝트에서 맡은 역할을 확인하세요. 자세한 내용은 구성원 및 프로젝트 역할 목록 가져오기 및 사용자 역할을 통한 액세스 권한 부여 섹션을 참조하세요.

변경 사항을 커밋하고 푸시할 수 있는 역할이 있는 경우 변경 내용을 커밋하려는 분기에 코드 변경 내용을 해당 분기로 푸시하지 못하도록 하는 분기 규칙이 구성되어 있을 수 있습니다. 대신 브랜치를 만든 다음 해당 브랜치에 코드를 푸시해 보세요. 자세한 정보는 브랜치 규칙을 사용하여 브랜치에 허용된 작업 관리을 참조하세요.

프로젝트를 위한 소스 리포지토리가 필요해요.

문제: 프로젝트에 소스 리포지토리가 없거나 프로젝트에 다른 소스 리포지토리가 필요합니다.

가능한 해결 방법: 일부 프로젝트는 리소스 없이 생성됩니다. 프로젝트의 멤버인 경우 에서 해당 프로젝트의 소스 리포지토리를 만들 수 있습니다. CodeCatalyst 스페이스 관리자 역할을 가진 사람이

GitHub 저장소를 설치하고 GitHub 계정에 연결한 경우, 프로젝트 관리자 역할이 있는 경우 사용 가능한 GitHub 저장소를 연결하여 프로젝트에 추가할 수 있습니다. 자세한 내용은 소스 리포지토리 만들기 및 소스 리포지토리 연결을 참조하십시오.

내 소스 리포지토리는 새 저장소인데 커밋이 들어 있습니다.

문제: 방금 소스 리포지토리를 만들었습니다. 비어 있어야 하는데 커밋, 브랜치, `README.md` 파일이 들어 있습니다.

가능한 해결 방법: 이는 예상된 동작입니다. 의 모든 소스 CodeCatalyst 리포지토리에는 기본 브랜치를 샘플 코드 (샘플 코드가 포함된 블루프린트를 사용하여 프로젝트용으로 만든 경우) 또는 리포지토리 README 파일용 템플릿 마크다운 파일로 설정하는 초기 커밋이 포함되어 있습니다. `main` 콘솔과 Git 클라이언트에서 추가 브랜치를 생성할 수 있습니다. 콘솔에서 파일을 생성 및 편집하고, 개발 환경 및 Git 클라이언트에서 파일을 삭제할 수 있습니다.

다른 브랜치를 기본 브랜치로 사용하고 싶습니다.

문제: 소스 리포지토리에 이름이 지정된 `main` 기본 브랜치가 있지만 다른 브랜치를 기본 브랜치로 사용하고 싶습니다.

가능한 해결 방법: 의 소스 리포지토리에서 기본 브랜치를 변경하거나 삭제할 수 없습니다. CodeCatalyst 추가 브랜치를 만들고 워크플로의 소스 작업에 해당 브랜치를 사용할 수 있습니다. GitHub 리포지토리를 연결하여 프로젝트의 리포지토리로 사용할 수도 있습니다.

풀 리퀘스트 활동에 대한 이메일을 받고 있습니다.

문제: 풀 리퀘스트 활동에 대한 이메일 알림을 등록하거나 구성하지 않았는데 어쨌든 수신되고 있습니다.

가능한 해결 방법: 풀 리퀘스트 활동에 대한 이메일 알림이 자동으로 전송됩니다. 자세한 정보는 Amazon에서 풀 리퀘스트를 사용하여 코드 검토하기 CodeCatalyst을 참조하세요.

개인 액세스 토큰 (PAT) 을 잊어버렸습니다.

문제: 소스 리포지토리의 코드를 복제, 푸시 및 가져오기 위해 PAT를 사용해 왔는데 토큰의 가치를 잃어 콘솔에서 찾을 수 없습니다. CodeCatalyst

가능한 해결 방법: 이 문제를 해결하는 가장 빠른 방법은 다른 PAT를 만들고 이 새 PAT를 사용하도록 자격 증명 관리자 또는 IDE를 구성하는 것입니다. PAT 생성 시에만 PAT 값이 표시됩니다. 이 값을 잃

으면 다시 가져올 수 없습니다. 자세한 정보는 개인용 액세스 토큰으로 사용자에게 리포지토리 액세스 권한 부여을 참조하세요.

풀 리퀘스트에는 예상한 변경 사항이 표시되지 않습니다.

문제: 풀 리퀘스트를 생성했지만 소스 브랜치와 대상 브랜치 간에 예상한 변경 사항이 보이지 않습니다.

가능한 해결 방법: 이 문제는 여러 가지 문제로 인해 발생할 수 있습니다. 다음 해결 방법 중 하나 이상을 시도해 보십시오.

- 이전 버전 사이의 변경 내용을 검토 중이거나 최신 변경 내용을 보지 않을 수 있습니다. 브라우저를 새로 고치고 보려는 수정 버전 간의 비교를 선택했는지 확인하세요.
- 풀 리퀘스트의 모든 변경 사항을 콘솔에 표시할 수 있는 것은 아닙니다. 예를 들어 콘솔에서 Git 하위 모듈을 볼 수 없으므로 pull 요청에서 하위 모듈의 차이점을 볼 수 없습니다. 일부 차이가 너무 커서 표시할 수 없을 수도 있습니다. 자세한 내용은 의 소스 리포지토리 할당량 CodeCatalyst 및 파일 보기 섹션을 참조하세요.
- 풀 리퀘스트는 병합 기반과 사용자가 선택한 수정 버전 간의 차이를 표시합니다. 풀 리퀘스트를 생성하면 소스 브랜치의 끝과 대상 브랜치의 끝 부분 간의 차이가 표시됩니다. 풀 리퀘스트가 생성되면 수정 버전과 병합 기준 간의 차이가 표시됩니다. 병합 기반은 수정 버전이 생성될 때 대상 브랜치의 끝부분이었던 커밋입니다. 수정 버전 간에 병합 기준이 변경될 수 있습니다. Git의 차이점과 병합 기준에 대한 자세한 내용은 Git git-merge-base설명서를 참조하십시오.

풀 리퀘스트는 병합 불가 상태로 표시됩니다.

문제: 풀 리퀘스트를 병합하고 싶은데 상태가 병합 불가능으로 표시됩니다.

가능한 해결 방법: 이 문제는 하나 이상의 문제로 인해 발생할 수 있습니다.

- 풀 리퀘스트에 필요한 모든 리뷰어가 풀 리퀘스트를 승인해야만 풀 리퀘스트를 병합할 수 있습니다. 이름 옆에 시계 아이콘이 있는 모든 검토자의 필수 검토자 목록을 검토하십시오. 시계 아이콘은 리뷰어가 풀 리퀘스트를 승인하지 않았음을 나타냅니다.

> **Note**
>
> 풀 리퀘스트를 승인하기 전에 필수 리뷰어가 프로젝트에서 제거된 경우에는 풀 리퀘스트를 병합할 수 없습니다. 풀 리퀘스트를 닫고 새 풀 리퀘스트를 생성하세요.

- 소스 브랜치와 대상 브랜치 사이에 병합 충돌이 있을 수 있습니다. CodeCatalyst 가능한 모든 Git 병합 전략 및 옵션을 지원하지는 않습니다. 개발 환경에서 브랜치의 병합 충돌을 평가하거나 리포지토리를 복제하고 IDE 또는 Git 도구를 사용하여 병합 충돌을 찾아 해결할 수 있습니다. 자세한 내용은 풀 리퀘스트 병합을(를) 참조하세요.

프로젝트 및 청사진 문제 해결

이 섹션은 Amazon에서 프로젝트 및 청사진을 작업할 때 발생할 수 있는 몇 가지 일반적인 문제를 해결하는 데 도움이 될 수 있습니다. CodeCatalyst

아파치-메이븐-3.8.6에 대한 종속성이 누락된 AWS Fargate 블루프린트가 있는 자바 API

문제: AWS Fargate 블루프린트가 포함된 Java API에서 만든 프로젝트의 경우 워크플로가 실패하고 종속성 누락 오류가 발생합니다. `apache-maven-3.8.6` 다음 예제와 비슷한 결과가 나오면서 워크플로가 실패합니다.

```
Step 8/25 : RUN wget https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-
maven-3.8.6-bin.tar.gz -P /tmp
---> Running in 1851ce6f4d1b
[91m--2023-03-10 01:24:55--  https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/
apache-maven-3.8.6-bin.tar.gz
[0m[91mResolving dlcdn.apache.org (dlcdn.apache.org)...
[0m[91m151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443...
[0m[91mconnected.
[0m[91mHTTP request sent, awaiting response... [0m[91m404 Not Found
2023-03-10 01:24:55 ERROR 404: Not Found.
[0mThe command '/bin/sh -c wget https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/
apache-maven-3.8.6-bin.tar.gz -P /tmp' returned a non-zero code: 8
[Container] 2023/03/10 01:24:55 Command failed with exit status 8
```

솔루션: 다음 단계를 사용하여 블루프린트 Dockerfile을 업데이트하십시오.

1. 검색 창에서 를 입력하여 블루프린트가 `apache-maven-3.8.6` 있는 Java API로 만든 프로젝트 내의 dockerfile을 찾습니다. AWS Fargate
2. Dockerfile (`/static-assets/app/Dockerfile`) 을 업데이트하여 기본 `maven:3.9.0-amazoncorretto-11` 이미지로 사용하고 패키지에 대한 종속성을 제거합니다. `apache-maven-3.8.6`

3. (권장) 또한 Maven 힙 크기를 6GB로 업데이트하는 것이 좋습니다.

다음은 Dockerfile의 예시입니다.

```
FROM maven:3.9.0-amazoncorretto-11 AS builder

COPY ./pom.xml ./pom.xml
COPY src ./src/

ENV MAVEN_OPTS='-Xmx6g'

RUN mvn -Dmaven.test.skip=true clean package

FROM amazoncorretto:11-alpine

COPY –from=builder target/CustomerService-0.0.1.jar CustomerService-0.0.1.jar
EXPOSE 80
CMD ["java","-jar","-Dspring.profiles.active=prod","/CustomerService-0.0.1.jar", "–
server.port=80"]
```

Amazon의 권한 오류로 OnPullRequest인해 최신 3계층 웹 애플리케이션 블루프린트 워크플로가 실패함 CodeGuru

문제: 프로젝트에 워크플로를 실행하려고 하면 워크플로가 실행되지 않고 다음 메시지가 표시됩니다.

```
Failed at codeguru_codereview: The action failed during runtime. View the action's logs
 for more details.
```

해결 방법: 이 작업 실패의 한 가지 가능한 원인은 IAM 역할 정책의 권한 누락 때문일 수 있습니다. AWS 계정 즉, 연결된 사용자가 CodeCatalyst 사용하는 서비스 역할 버전에 codeguru_codereview 작업을 성공적으로 실행하는 데 필요한 권한이 누락되었기 때문일 수 있습니다. 이 문제를 해결하려면 서비스 역할을 필수 권한으로 업데이트하거나 워크플로에 사용되는 서비스 역할을 Amazon CodeGuru 및 Amazon CodeGuru Reviewer에 필요한 권한이 있는 서비스 역할로 변경해야 합니다. 워크플로가 성공적으로 실행되도록 하려면 다음 단계를 사용하여 역할을 찾고 역할 정책 권한을 업데이트하십시오.

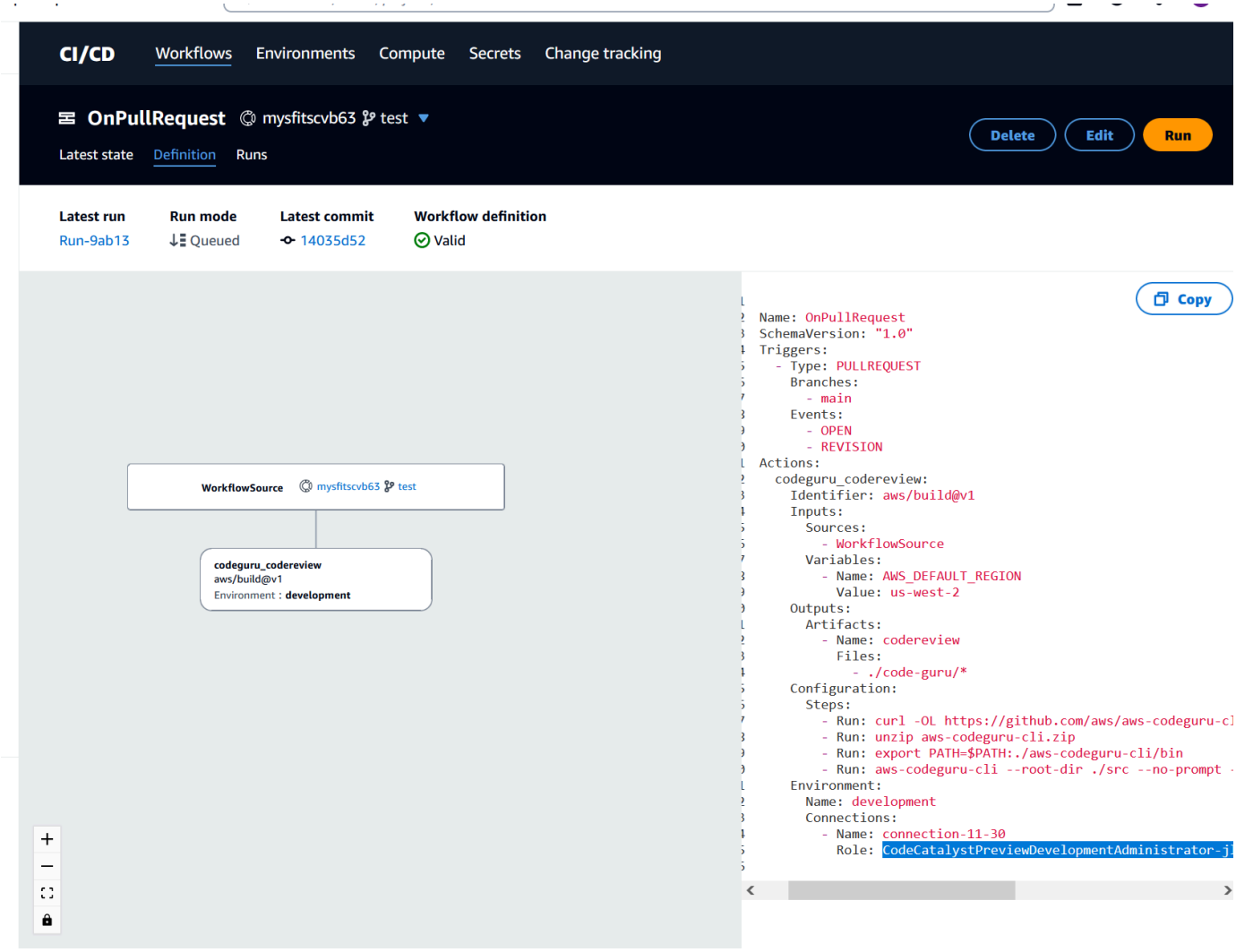
> **Note**
>
> 이 단계는 다음 워크플로에 적용됩니다 CodeCatalyst.
>
> - OnPullRequest워크플로는 최신 3계층 웹 애플리케이션 청사진을 사용하여 만든 프로젝트에 제공됩니다. CodeCatalyst
> - Amazon CodeGuru 또는 Amazon CodeGuru Reviewer에 액세스하는 작업을 CodeCatalyst 통해 프로젝트에 워크플로가 추가되었습니다.

각 프로젝트에는 프로젝트에 AWS 계정 연결된 사용자가 제공하는 역할과 환경을 사용하는 작업이 포함된 워크플로가 포함되어 있습니다. CodeCatalyst 작업 및 지정된 정책이 포함된 워크플로는 /.codecatalyst/workflows 디렉토리의 소스 리포지토리에 저장됩니다. 기존 워크플로에 새 역할 ID를 추가하지 않는 한 워크플로 YAML을 수정할 필요는 없습니다. YAML 템플릿 요소 및 형식에 대한 자세한 내용은 을 참조하십시오. 워크플로우 YAML 정의

다음은 역할 정책을 편집하고 워크플로 YAML을 확인하기 위해 따라야 하는 고급 단계입니다.

워크플로 YAML에서 역할 이름을 참조하고 정책을 업데이트하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요. 프로젝트로 이동하세요.
3. CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 제목이 지정된 워크플로를 선택합니다. OnPullRequest 정의 탭을 선택합니다.
5. 워크플로 YAML의 codeguru_codereview 작업 아래 `Role:` 필드에 역할 이름을 기록해 둡니다. 이 역할은 IAM에서 수정하려는 정책의 역할입니다. 다음 예는 역할 이름을 보여줍니다.



6. 다음 중 하나를 수행하세요.

- (권장) 프로젝트에 연결된 서비스 역할을 Amazon CodeGuru 및 Amazon CodeGuru Reviewer에 필요한 권한으로 업데이트하십시오. 역할에는 고유 식별자가 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 추가된 이름이 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오CodeCatalystWorkflowDevelopmentRole-***spaceName***서비스 역할 이해. 다음 단계로 진행하여 IAM에서 정책을 업데이트하십시오.

  > **Note**
  >
  > 역할 및 정책을 AWS 계정 가진 AWS 관리자 액세스 권한이 있어야 합니다.

- 워크플로에 사용되는 서비스 역할을 Amazon CodeGuru 및 Amazon CodeGuru Reviewer에 필요한 권한이 있는 역할로 변경하거나 필요한 권한이 있는 새 역할을 생성합니다.

7. AWS Management Console에 로그인하여 https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

   IAM 콘솔에서 5단계의 역할 (예:) 을 찾습니다. `CodeCatalystPreviewDevelopmentRole`

8. 5단계의 역할에서 `codeguru-reviewer:*` 및 권한을 포함하도록 `codeguru:*` 권한 정책을 변경합니다. 이러한 권한을 추가한 후에는 권한 정책이 다음과 비슷해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudformation:*",
                "lambda:*",
                "apigateway:*",
                "ecr:*",
                "ecs:*",
                "ssm:*",
                "codedeploy:*",
                "s3:*",
                "iam:DeleteRole",
                "iam:UpdateRole",
                "iam:Get*",
                "iam:TagRole",
                "iam:PassRole",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
                "iam:PutRolePolicy",
                "iam:CreatePolicy",
                "iam:DeletePolicy",
                "iam:CreatePolicyVersion",
                "iam:DeletePolicyVersion",
                "iam:PutRolePermissionsBoundary",
                "iam:DeleteRolePermissionsBoundary",
                "sts:AssumeRole",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeRules",
                "elasticloadbalancing:ModifyRule",
                "cloudwatch:DescribeAlarms",
                "sns:Publish",
```

```
                "sns:ListTopics",
                "codeguru-reviewer:*",
                "codeguru:*"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

9. 정책을 수정한 후에는 다시 CodeCatalyst 돌아가서 워크플로를 다시 실행하십시오.

아직도 문제 해결을 찾고 계신가요?

Amazon으로 CodeCatalyst 이동하거나 Support Feedback 양식을 작성할 수 있습니다. 정보 요청 섹션의 도움을 받을 수 있는 방법 아래에 아마존 CodeCatalyst 고객임을 기재하십시오. 문제를 가장 효율적으로 해결할 수 있도록 최대한 자세히 설명해 주십시오.

개발 환경 관련 문제 해결

개발 환경과 관련된 문제를 해결하려면 다음 섹션을 참조하십시오. 개발 환경에 대한 자세한 내용은 을 참조하십시오. 의 개발 환경을 사용하여 코드 작성 및 수정 CodeCatalyst

주제

- 할당량 문제 때문에 개발 환경 생성에 실패했습니다.
- 개발 환경의 변경 내용을 리포지토리의 특정 브랜치로 푸시할 수 없습니다.
- 개발 환경이 다시 시작되지 않았습니다.
- 내 개발 환경 연결이 끊어졌습니다.
- VPC연결된 개발 환경이 실패했습니다.
- 내 프로젝트가 어느 디렉터리에 있는지 찾을 수 없어요
- 를 통해 내 개발 환경에 연결할 수 없습니다. SSH
- 로컬 SSH 구성이 없어서 개발 환경에 연결할 SSH 수 없습니다.
- 내 프로필에 문제가 SSH 생겨서 개발 환경에 연결할 수 없습니다. AWS Configcodecatalyst
- SSO (Single Sign-On) 계정을 CodeCatalyst 사용하여 로그인한 상태에서는 개발 환경을 만들 수 없습니다.
- IDE 관련 문제 해결

- 개발 파일 관련 문제 해결

할당량 문제 때문에 개발 환경 생성에 실패했습니다.

문제: 에서 개발 환경을 만들고 CodeCatalyst 싶은데 오류가 나타납니다. 콘솔의 개발 환경 페이지에 해당 공간의 스토리지 한도에 도달했다는 메시지가 표시됩니다.

가능한 해결 방법: 프로젝트 또는 스페이스에서의 역할에 따라 자체 개발 환경을 하나 이상 삭제하거나, 스페이스 관리자 역할이 있는 경우 다른 사용자가 만든 사용하지 않는 개발 환경을 삭제할 수 있습니다. 또한 청구 등급을 스토리지가 더 많은 계층으로 변경할 수도 있습니다.

- 보관 한도를 확인하려면 Amazon CodeCatalyst 스페이스의 Billing 탭에서 사용 할당량이 최대 허용량에 도달했는지 확인하십시오. 할당량이 최대치에 도달한 경우 스페이스 관리자 역할을 가진 사람에게 문의하여 불필요한 개발 환경을 제거하거나 청구 계층 변경을 고려하세요.
- 생성한 개발 환경 중 더 이상 필요하지 않은 개발 환경을 제거하려면 을 참조하십시오. Dev Environment 삭제

문제가 계속되고 오류가 발생하는 경우 개발 환경을 만들 수 있는 CodeCatalyst 역할이 있는지 확인하세요. IDE 스페이스 관리자 역할, 프로젝트 관리자 역할 및 기여자 역할 모두 개발 환경을 만들 수 있는 권한을 가집니다. 자세한 내용은 사용자 역할을 통한 액세스 권한 부여 단원을 참조하십시오.

개발 환경의 변경 내용을 리포지토리의 특정 브랜치로 푸시할 수 없습니다.

문제: 개발 환경의 코드 변경 사항을 커밋하고 소스 리포지토리의 브랜치에 푸시하고 싶은데 오류가 발생했습니다.

가능한 해결 방법: 프로젝트 또는 스페이스에서의 역할에 따라 코드를 프로젝트의 소스 리포지토리로 푸시할 수 있는 권한이 없을 수 있습니다. 스페이스 관리자 역할, 프로젝트 관리자 역할 및 기여자 역할 모두 코드를 프로젝트의 저장소로 푸시할 수 있는 권한을 가집니다.

기여자 역할이 있지만 특정 브랜치에 코드를 푸시할 수 없는 경우, 해당 역할을 가진 사용자가 특정 브랜치로 코드를 푸시하지 못하도록 하는 브랜치 규칙이 특정 브랜치에 구성되어 있을 수 있습니다. 변경 내용을 다른 브랜치에 푸시하거나 브랜치를 만든 다음 코드를 해당 브랜치에 푸시해 보세요. 자세한 내용은 브랜치 규칙을 사용하여 브랜치에 허용된 작업 관리 단원을 참조하십시오.

개발 환경이 다시 시작되지 않았습니다.

문제: 개발 환경을 중지한 후 다시 시작하지 않았습니다.

가능한 해결 방법: 문제를 해결하려면 Amazon CodeCatalyst 스페이스의 Billing 탭에서 사용 할당량이 최대 한도에 도달했는지 확인하십시오. 할당량이 최대 한도에 도달한 경우 스페이스 관리자에게 문의하여 청구 등급을 높이십시오.

내 개발 환경 연결이 끊어졌습니다.

문제: 사용 중에 개발 환경 연결이 끊겼습니다.

가능한 해결 방법: 문제를 해결하려면 인터넷 연결을 확인하세요. 인터넷에 연결되어 있지 않은 경우, 연결하고 개발 환경에서 작업을 다시 시작하세요.

VPC연결된 개발 환경이 실패했습니다.

문제: 내 개발 환경에 VPC 연결을 연결했는데 오류가 발생했습니다.

가능한 해결 방법: Docker는 브리지 네트워크라는 링크 계층 장치를 사용하여 동일한 브리지 네트워크에 연결된 컨테이너가 통신할 수 있도록 합니다. 기본 브리지는 일반적으로 `172.17.0.0/16` 서브넷을 컨테이너 네트워킹에 사용합니다. 환경 인스턴스의 VPC 서브넷이 에서 이미 사용하고 있는 것과 동일한 주소 범위를 사용하는 경우 IP 주소 충돌이 발생할 수 있습니다. Docker 동일한 주소 블록을 Docker 사용하는 VPC Amazon으로 인해 발생하는 IP IPv4 CIDR 주소 충돌을 해결하려면 다른 CIDR 블록을 `172.17.0.0/16` 구성하십시오.

> **Note**
>
> 기존 VPC 또는 서브넷의 IP 주소 범위는 변경할 수 없습니다.

내 프로젝트가 어느 디렉터리에 있는지 찾을 수 없어요

문제: 내 프로젝트가 어느 디렉터리에 있는지 찾을 수 없어요.

가능한 해결 방법: 프로젝트를 찾으려면 디렉터리를 로 변경하세요`/projects`. 이 디렉토리에서 프로젝트를 찾을 수 있습니다.

를 통해 내 개발 환경에 연결할 수 없습니다. SSH

를 통해 SSH 개발 환경 연결 문제를 해결하려면 `-vvv` 옵션을 `ssh` 사용하여 명령을 실행하여 문제 해결 방법에 대한 추가 정보를 표시할 수 있습니다.

```
ssh -vvv codecatalyst-dev-env=<space-name>=<project-name>=<dev-environment-id>
```

로컬 SSH 구성이 없어서 개발 환경에 연결할 SSH 수 없습니다.

로컬 SSH config (`~/.ssh/config`) 가 없거나 `Host codecatalyst-dev-env*` 섹션 내용이 최신 상태가 아닌 경우 를 통해 개발 환경에 연결할 수 없습니다. SSH 이 문제를 해결하려면 `Host codecatalyst-dev-env*` 섹션을 삭제하고 SSH액세스 모달에서 첫 번째 명령을 다시 실행하세요. 자세한 내용은 를 사용하여 개발 환경에 연결 SSH 단원을 참조하십시오.

내 프로필에 문제가 SSH 생겨서 개발 환경에 연결할 수 없습니다. AWS Config**codecatalyst**

`codecatalyst`프로필의 AWS Config (`~/.aws/config`) 가 에 설명된 것과 일치하는지 확인하십시오AWS CLI와 함께 사용하기 위한 설정 CodeCatalyst. 그렇지 않은 경우 프로필을 `codecatalyst` 삭제하고 SSH액세스 모달에서 첫 번째 명령을 다시 실행하십시오. 자세한 내용은 를 사용하여 개발 환경에 연결 SSH 단원을 참조하십시오.

SSO (Single Sign-On) 계정을 CodeCatalyst 사용하여 로그인한 상태에서는 개발 환경을 만들 수 없습니다.

문제: CodeCatalyst 콘솔에 SSO 사용자로 로그인했을 때 스페이스에 개발 환경을 만들기로 선택하면 알 수 없는 예외 오류가 발생합니다. 개발 환경을 만들고 액세스 권한 (예:) 을 선택하면 다음과 AWS Cloud9비슷한 문제가 발생합니다. IDE

- CodeCatalyst 콘솔의 개발 환경 페이지에는 목록에 있는 개발 환경이 상태와 함께 표시됩니다. `FAILED`
- 다음과 유사한 오류 메시지가 표시됩니다.

  `An unknown exception happened`

  `We encountered an unknown exception when launching your Dev Environment. Mention your Dev Environment id` *`error_message_ID`* `if you want to report or need any help.`

수정 방법:

Active Directory가 ID 공급자로 사용되는 공간에 있는 사용자는 개발 환경을 사용할 수 없습니다. 스페이스 관리자는 Identity Center와 같은 IAM 대체 ID 공급자를 사용하여 개발 환경에 액세스할 수 있습니다. ID 페더레이션을 지원하는 공간을 계획하는 방법에 대한 자세한 내용은 CodeCatalyst 관리자 안내서의 ID 페더레이션을 지원하는 공간 계획을 참조하십시오.

IDE 관련 문제 해결

에서 IDE와 관련된 문제를 해결하려면 다음 섹션을 참조하십시오. CodeCatalyst IDE에 대한 자세한 내용은 을 참조하십시오. 개발 환경 만들기 IDE

주제

의 런타임 이미지 버전이 일치하지 않습니다. AWS Cloud9

AWS Cloud9다른 버전의 프론트엔드 에셋과 백엔드 런타임 이미지를 사용하고 있습니다. 다른 버전을 사용하면 Git 확장 프로그램이 제대로 작동하지 않을 AWS Toolkit 수 있습니다. 문제를 해결하려면 개발 환경 대시보드로 이동하여 개발 환경을 중지한 다음 다시 시작하세요. API를 사용하여 문제를 해결하려면 `UpdateDevEnvironment` API를 사용하여 런타임을 업데이트하세요. 자세한 내용은 Amazon CodeCatalyst API 참조를 참조하십시오 UpdateDevEnvironment.

내 **`/projects/projects`** 파일에 액세스할 수 없습니다. AWS Cloud9

AWS Cloud9편집기가 디렉터리에 있는 파일에 접근할 수 없습니다`/projects/projects`. 문제를 해결하려면 AWS Cloud9 터미널을 사용하여 파일에 액세스하거나 파일을 다른 디렉터리로 이동하십시오.

사용자 지정 devfile을 AWS Cloud9 사용하여 개발 환경을 시작할 수 없습니다.

devfile 이미지가 호환되지 않을 수 있습니다. AWS Cloud9 문제를 해결하려면 리포지토리와 해당 개발 환경에서 devfile을 검토하고 새 파일을 만들어 계속 진행하세요.

에서 문제가 생겼습니다. AWS Cloud9

다른 문제의 경우 AWS Cloud9사용 설명서의 문제 해결 섹션을 확인하십시오.

에서 을 JetBrains (를) 통해 내 개발 환경에 연결할 수 없습니다. CodeCatalyst

문제를 해결하려면 최신 버전만 JetBrains 설치되어 있는지 확인하세요. 여러 버전이 있는 경우 이전 버전을 제거하고 IDE와 브라우저를 닫아 프로토콜 핸들러를 다시 등록하십시오. 그런 다음 프로토콜 핸들러를 다시 JetBrains 열고 등록합니다.

AWS ToolkitIDE용으로 설치할 수 없습니다.

VS Code에서 이 문제를 해결하려면 AWS Toolkit for Visual Studio Code 에서 수동으로 설치하세요 GitHub.

에서 이 문제를 해결하려면 AWS Toolkit for JetBrains 에서 수동으로 GitHub설치하세요. JetBrains

IDE에서는 개발 환경을 시작할 수 없습니다.

VS Code에서 이 문제를 해결하려면 최신 버전의 VS Code가 AWS Toolkit for Visual Studio Code 설치되어 있는지 확인하세요. 최신 버전이 없는 경우 개발 환경을 업데이트하고 실행하세요. 자세한 내용은 Amazon CodeCatalyst for VS Code를 참조하십시오.

이 문제를 해결하려면 최신 버전이 AWS Toolkit for JetBrains 설치되어 있는지 확인하십시오. JetBrains JetBrains 최신 버전이 없는 경우 개발 환경을 업데이트하고 실행하세요. 자세한 내용은 CodeCatalyst Amazon을 참조하십시오 JetBrains.

개발 파일 관련 문제 해결

에서 개발파일과 관련된 문제를 해결하려면 다음 섹션을 참조하십시오. CodeCatalyst devfile에 대한 자세한 내용은 을 참조하십시오. 개발 환경을 위한 개발 파일 구성

주제

- 사용자 지정 devfile에 사용자 지정 이미지를 구현했는데도 내 개발 환경은 기본 범용 devfile을 사용하고 있습니다.
- 내 프로젝트가 기본 유니버설 devfile을 사용하는 개발 환경에서 빌드되지 않습니다.
- 개발 환경을 위한 리포지토리 devfile을 옮기고 싶습니다.
- 개발 파일을 시작하는 데 문제가 있습니다.
- 개발 파일 상태를 확인하는 방법을 잘 모르겠습니다.
- 내 devfile이 최신 이미지에 제공된 도구와 호환되지 않습니다.

사용자 지정 devfile에 사용자 지정 이미지를 구현했는데도 내 개발 환경은 기본 범용 devfile을 사용하고 있습니다.

사용자 지정 devfile을 사용하는 개발 환경을 시작하는 동안 오류가 CodeCatalyst 발생하면 개발 환경은 기본 범용 devfile을 기본값으로 사용합니다. 문제를 해결하려면 아래 로그에서 정확한 오류를 확인할 수 있습니다. `/aws/mde/logs/devfile.log` 로그에서 `postStart` 실행이 성공했는지 확인할 수도 있습니다`/aws/mde/logs/devfileCommand.log`.

내 프로젝트가 기본 유니버설 devfile을 사용하는 개발 환경에서 빌드되지 않습니다.

문제를 해결하려면 커스텀 devfile을 사용하고 있지 않은지 확인하세요. 사용자 지정 devfile을 사용하지 않는 경우 프로젝트의 소스 저장소에 있는 `devfile.yaml` 파일을 보고 오류를 찾아 수정하십시오.

개발 환경을 위한 리포지토리 devfile을 옮기고 싶습니다.

기본 devfile을 소스 코드 리포지토리로 옮길 수 있습니다. `/projects/devfile.yaml` devfile 위치를 업데이트하려면 다음 명령을 사용하십시오. `/aws/mde/mde start --location` *`repository-name`*`/devfile.yaml`

개발 파일을 시작하는 데 문제가 있습니다.

개발 파일을 시작하는 데 문제가 있는 경우 복구 모드로 전환되므로 환경에 계속 연결하여 devfile을 수정할 수 있습니다. 복구 모드에서는 실행 시 devfile 위치가 포함되지 `/aws/mde/mde status` 않습니다.

```
{
    "status": "STABLE"
}
```

아래 로그에서 오류를 확인하고, devfile을 수정하고`/aws/mde/logs`, 다시 `/aws/mde/mde start` 실행해 볼 수 있습니다.

개발 파일 상태를 확인하는 방법을 잘 모르겠습니다.

를 실행하여 devfile 상태를 확인할 수 있습니다. `/aws/mde/mde status` 이 명령을 실행하면 다음 중 하나가 표시될 수 있습니다.

- `{"status": "STABLE", "location": "devfile.yaml" }`

  이는 devfile이 정확하다는 것을 나타냅니다.

- `{"status": "STABLE" }`

  이는 devfile을 시작할 수 없고 복구 모드로 들어갔음을 나타냅니다.

의 로그에서 `/aws/mde/logs/devfile.log` 정확한 오류를 확인할 수 있습니다.

로그에서 `postStart` 실행이 성공했는지 확인할 수도 있습니다`/aws/mde/logs/devfileCommand.log`.

자세히 알아보려면 개발 환경을 위한 범용 devfile 이미지 지정하기의 내용을 참조하세요.

내 devfile이 최신 이미지에 제공된 도구와 호환되지 않습니다.

개발 환경에서 `devfile` 또는 `latest` 도구에 특정 프로젝트에 필요한 도구가 없는 경우 실패할 `devfile postStart` 수 있습니다. 문제를 해결하려면 다음과 같이 하세요.

1. devfile로 이동합니다.
2. devfile에서 대신 세분화된 이미지 버전으로 업데이트하세요. `latest` 다음과 비슷해 보일 수 있습니다.

   ```
   components:
     - container:
         image: public.ecr.aws/amazonlinux/universal-image:1.0
   ```
3. 업데이트된 devfile을 사용하여 새 개발 환경을 만드세요.

워크플로우 관련 문제 해결

CodeCatalystAmazon의 워크플로와 관련된 문제를 해결하려면 다음 섹션을 참조하십시오. 워크플로에 대한 자세한 내용은 워크플로를 통한 빌드, 테스트, 배포 섹션을 참조하세요.

주제

- “워크플로가 비활성 상태입니다.” 라는 메시지를 수정하려면 어떻게 해야 합니까?
- “워크플로 정의에 다음과 같은 문제가 있는 문제를 해결하려면 어떻게 해야 합니까?n 오류” 오류를 해결합니까?
- “자격 증명을 찾을 수 없음” 및 "ExpiredToken" 오류를 해결하려면 어떻게 해야 하나요?
- “서버에 연결할 수 없음” 오류를 해결하려면 어떻게 해야 하나요?
- 비주얼 에디터에서 CodeDeploy 필드가 누락된 이유는 무엇입니까?

“워크플로가 비활성 상태입니다.” 라는 메시지를 수정하려면 어떻게 해야 합니까?

문제: CodeCatalyst 콘솔의 CI/CD, 워크플로우에서 워크플로가 다음 메시지와 함께 나타납니다.

```
Workflow is inactive.
```

이 메시지는 워크플로 정의 파일에 현재 사용 중인 브랜치에 적용되지 않는 트리거가 포함되어 있음을 나타냅니다. 예를 들어 워크플로 정의 파일에는 브랜치를 참조하는 PUSH 트리거가 포함되어 있지만 사용자는 기능 `main` 브랜치를 사용하고 있을 수 있습니다. 기능 브랜치에서 변경한 내용은 해당 브랜치에 적용되지 않고 워크플로 실행을 시작하지도 않으므로 브랜치에서 `main` 워크플로를 CodeCatalyst 해제하고 로 표시합니다. `main` `Inactive`

수정 방법:

피처 브랜치에서 워크플로를 시작하려면 다음과 같이 하면 됩니다.

- 기능 브랜치의 워크플로 정의 파일에서 `Triggers` 섹션의 `Branches` 속성을 제거하여 다음과 같이 표시되도록 합니다.

```
Triggers:
  - Type: PUSH
```

이 구성을 사용하면 기능 브랜치를 포함한 모든 브랜치로 푸시할 때 트리거가 활성화됩니다. 트리거가 CodeCatalyst 활성화되면 푸시하려는 브랜치의 워크플로 정의 파일과 소스 파일을 사용하여 워크플로 실행을 시작합니다.

- 기능 브랜치의 워크플로 정의 파일에서 `Triggers` 섹션을 제거하고 워크플로를 수동으로 실행합니다.
- 기능 브랜치의 워크플로 정의 파일에서 다른 브랜치 (예`main`:) 가 아닌 기능 브랜치를 참조하도록 `PUSH` 섹션을 변경하십시오.

> **Important**
>
> 변경 내용을 `main` 브랜치에 다시 병합하지 않으려는 경우에는 이러한 변경 내용을 커밋하지 않도록 주의하십시오.

워크플로 정의 파일 편집에 대한 자세한 내용은 을 참조하십시오워크플로 생성.

트리거에 대한 자세한 내용은 트리거를 사용하여 자동으로 워크플로 실행 시작 주제를 참조하십시오.

“워크플로 정의에 다음과 같은 문제가 있는 문제를 해결하려면 어떻게 해야 합니까?*n* 오류” 오류를 해결합니까?

문제: 다음과 같은 오류 메시지가 표시됩니다.

오류 1:

CI/CD, 워크플로 페이지의 워크플로 이름 아래에 다음이 표시됩니다.

`Workflow definition has` *`n`* `errors`

오류 2:

워크플로를 편집하는 동안 검증 버튼을 선택하면 CodeCatalyst 콘솔 상단에 다음 메시지가 나타납니다.

`The workflow definition has errors. Fix the errors and choose Validate to verify your changes.`

오류 3:

워크플로의 세부 정보 페이지로 이동한 후 워크플로우 정의 필드에 다음 오류가 표시됩니다.

n errors

수정 방법:

- CI/CD를 선택하고 워크플로를 선택한 다음 오류가 있는 워크플로의 이름을 선택합니다. 상단 근처의 워크플로 정의 필드에서 오류 링크를 선택합니다. 오류에 대한 세부 정보가 페이지 하단에 표시됩니다. 오류의 문제 해결 팁을 따라 문제를 해결하십시오.
- 워크플로 정의 파일이 파일인지 확인하십시오. YAML
- 워크플로 정의 파일의 YAML 속성이 올바른 수준에 중첩되어 있는지 확인하십시오. 워크플로우 정의 파일에서 속성을 중첩하는 방법을 보려면 를 참조하거나 에서 링크된 작업 설명서를 참조하십시오. 워크플로우 YAML 정의 워크플로에 작업 추가
- 별표 (*) 및 기타 특수 문자가 제대로 이스케이프 처리되었는지 확인하십시오. 이를 피하려면 작은따옴표나 큰따옴표를 추가하세요. 예:

```
Outputs:
  Artifacts:
    - Name: myartifact
      Files:
        - "**/*"
```

  워크플로 정의 파일의 특수 문자에 대한 자세한 내용은 을 참조하십시오구문 지침 및 규칙.
- 워크플로 정의 파일의 YAML 속성이 올바른 대소문자를 사용하는지 확인하십시오. 케이스 규칙에 대한 자세한 내용은 을 참조하십시오구문 지침 및 규칙. 각 속성의 올바른 대/소문자를 확인하려면 를 참조하거나 링크된 작업 설명서를 참조하십시오워크플로에 작업 추가. 워크플로우 YAML 정의
- 워크플로우 정의 파일에 SchemaVersion 속성이 있고 올바른 버전으로 설정되어 있는지 확인하십시오. 자세한 내용은 SchemaVersion 단원을 참조하십시오.
- 워크플로 정의 파일의 Triggers 섹션에 필요한 모든 속성이 포함되어 있는지 확인하십시오. 필수 속성을 결정하려면 시각적 편집기에서 트리거를 선택하고 정보가 누락된 필드를 찾거나 에서 트리거 참조 설명서를 참조하십시오Triggers.
- 워크플로 정의 파일의 DependsOn 속성이 제대로 구성되어 있고 순환 종속성을 유발하지 않는지 확인하세요. 자세한 내용은 시퀀싱 액션 단원을 참조하십시오.
- 워크플로 정의 파일의 Actions 섹션에 하나 이상의 작업이 포함되어 있는지 확인하십시오. 자세한 내용은 작업 단원을 참조하십시오.
- 각 작업에 필요한 속성이 모두 포함되어 있는지 확인하십시오. 필수 속성을 결정하려면 비주얼 편집기에서 액션을 선택하고 정보가 누락된 필드를 찾거나 에서 링크된 액션 설명서를 참조하십시오워크플로에 작업 추가.

- 모든 입력 아티팩트에 해당하는 출력 아티팩트가 있는지 확인하십시오. 자세한 내용은 출력 아티팩트 정의 단원을 참조하십시오.
- 한 액션에 정의된 변수를 다른 액션에서 사용할 수 있도록 익스포트해야 합니다. 자세한 내용은 다른 액션에서 사용할 수 있도록 변수 내보내기 단원을 참조하십시오.

“자격 증명을 찾을 수 없음” 및 "ExpiredToken" 오류를 해결하려면 어떻게 해야 하나요?

문제: 작업을 진행하는 자습서: Amazon에 애플리케이션 배포 EKS 동안 개발 컴퓨터의 터미널 창에 다음 오류 메시지 중 하나 또는 둘 모두가 표시됩니다.

```
Unable to locate credentials. You can configure credentials by running "aws configure".
```

```
ExpiredToken: The security token included in the request is expired
```

수정 방법:

이러한 오류는 AWS 서비스에 액세스하는 데 사용하는 자격 증명이 만료되었음을 나타냅니다. 이 경우에는 `aws configure` 명령을 실행하지 마십시오. 대신 다음 지침에 따라 AWS 액세스 키와 세션 토큰을 새로 고치십시오.

AWS 액세스 키와 세션 토큰을 새로 고치려면

1. Amazon EKS 튜토리얼 완료 (`codecatalyst-eks-user`) 에서 사용 중인 사용자의 AWS 액세스 포털URL, 사용자 이름 및 암호가 있는지 확인하십시오. 자습서를 1단계: 개발 머신 설정 완료할 때 이러한 항목을 구성했어야 합니다.

   > **Note**
   >
   > 이 정보가 없는 경우 IAM Identity Center의 `codecatalyst-eks-user` 세부 정보 페이지로 이동하여 암호 재설정, 일회용 암호 생성 [...] 을 선택합니다. 그리고 암호를 다시 재설정하여 화면에 정보를 표시하십시오.

2. 다음 중 하나를 수행합니다.

   - AWS 액세스 URL 포털을 브라우저의 주소 표시줄에 붙여넣습니다.

     Or

- AWS 액세스 포털 페이지가 이미 로드되어 있다면 새로 고침하세요.

3. 아직 로그인하지 않았다면 codecatalyst-eks-user 해당 사용자 이름과 비밀번호로 로그인하세요.
4. 선택한 AWS 계정다음 codecatalyst-eks-user 사용자에게 할당한 AWS 계정 이름과 권한 집합을 선택합니다.
5. 권한 집합 이름 (codecatalyst-eks-permission-set) 옆에서 명령줄 또는 프로그래밍 방식 액세스를 선택합니다.
6. 페이지 중간에 있는 명령을 복사합니다. 다음과 비슷해 보입니다.

```
export AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
export AWS_SESSION_TOKEN="session-token"
```

... 어디서 *session-token* 길고 임의의 문자열입니다.

7. 개발 컴퓨터의 터미널 프롬프트에 명령을 붙여넣고 Enter 키를 누릅니다.

새 키와 세션 토큰이 로드됩니다.

이제 자격 증명을 새로 고쳤습니다. 이제 AWS CLIeksctl, 및 kubectl 명령이 작동할 것입니다.

“서버에 연결할 수 없음” 오류를 해결하려면 어떻게 해야 하나요?

문제: 에서 자습서: Amazon에 애플리케이션 배포 EKS 설명한 자습서를 진행하는 동안 개발 컴퓨터의 터미널 창에 다음과 비슷한 오류 메시지가 표시됩니다.

Unable to connect to the server: dial tcp: lookup *long-string*.gr7.us-west-2.eks.amazonaws.com on *1.2.3.4:5*: no such host

수정 방법:

이 오류는 일반적으로 kubectl 유틸리티가 Amazon EKS 클러스터에 연결하는 데 사용하는 자격 증명이 만료되었음을 나타냅니다. 문제를 해결하려면 터미널 프롬프트에 다음 명령을 입력하여 자격 증명을 새로 고치십시오.

```
aws eks update-kubeconfig --name codecatalyst-eks-cluster --region us-west-2
```

위치:

- *codecatalyst-eks-cluster* Amazon EKS 클러스터 이름으로 대체됩니다.
- *us-west-2* 클러스터가 배포된 AWS 지역으로 대체됩니다.

비주얼 에디터에서 CodeDeploy 필드가 누락된 이유는 무엇입니까?

문제: Amazon에 배포 ECS 작업을 사용하고 있는데 워크플로의 시각적 편집기와 같은 CodeDeploy CodeDeploy AppSpec필드가 표시되지 않습니다. 이 문제는 ECS 서비스 필드에 지정한 Amazon 서비스가 블루/그린 배포를 수행하도록 구성되지 않았기 때문에 발생할 수 있습니다.

수정 방법:

- Amazon에 배포 ECS 작업의 구성 탭에서 다른 Amazon ECS 서비스를 선택합니다. 자세한 내용은 워크플로를 ECS 사용하여 Amazon에 배포하기 단원을 참조하십시오.
- 블루/그린 배포를 수행하도록 선택한 Amazon ECS 서비스를 구성합니다. 블루/그린 배포 구성에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 블루/그린 배포를 참조하십시오. CodeDeploy

기능 오류를 수정하려면 어떻게 해야 합니까? IAM

문제: AWS CloudFormation 스택 배포 작업을 사용하고 있는데 배포 스택 작업의 **##[error] requires capabilities: [*capability-name*]** 로그에서 확인할 수 있습니다. AWS CloudFormation

가능한 해결 방법: 다음 절차를 완료하여 워크플로 정의 파일에 기능을 추가하십시오. IAM기능에 대한 자세한 내용은 IAM사용 설명서의 AWS CloudFormation 템플릿의 IAM 리소스 승인을 참조하십시오.

Visual

비주얼 편집기를 사용하여 IAM 기능을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. Visual을 선택합니다.

7. 워크플로 다이어그램에서 AWS CloudFormation 스택 배포 작업을 선택합니다.
8. 구성 탭을 선택합니다.
9. 하단에서 고급 - 선택 사항을 선택합니다.
10. 기능 드롭다운 목록에서 오류 메시지에 언급된 기능 옆의 확인란을 선택합니다. 목록에서 기능을 사용할 수 없는 경우 YAML 편집기를 사용하여 추가하십시오.
11. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.
12. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.
13. 새 워크플로 실행이 자동으로 시작되지 않는 경우 워크플로를 수동으로 실행하여 변경 사항으로 오류가 해결되는지 확인하십시오. 워크플로를 수동으로 실행하는 방법에 대한 자세한 내용은 을 참조하십시오워크플로우 수동 실행 시작.

YAML

YAML편집기를 사용하여 IAM 기능을 추가하려면

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. 프로젝트를 선택합니다.
3. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.
4. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.
5. 편집을 선택합니다.
6. 선택합니다 YAML.
7. AWS CloudFormation 스택 배포 작업에서 다음과 같은 `capabilities` 속성을 추가합니다.

```
DeployCloudFormationStack:
  Configuration:
    capabilities: capability-name
```

Replace *capability-name* 오류 메시지에 표시된 IAM 기능 이름과 함께 여러 기능을 나열하려면 공백 없이 쉼표를 사용합니다. 자세한 내용은 의 `capabilities` AWS CloudFormation 'Deploy stack' 작업 YAML 속성 설명을 참조하십시오.

8. (선택 사항) 커밋하기 전에 [Validate] 를 선택하여 워크플로우 YAML 코드의 유효성을 검사합니다.

9. [커밋] 을 선택하고 커밋 메시지를 입력한 다음 [커밋] 을 다시 선택합니다.
10. 새 워크플로 실행이 자동으로 시작되지 않는 경우 워크플로를 수동으로 실행하여 변경 사항으로 오류가 해결되는지 확인하십시오. 워크플로를 수동으로 실행하는 방법에 대한 자세한 내용은 을 참조하십시오워크플로우 수동 실행 시작.

“npm install” 오류를 수정하려면 어떻게 해야 하나요?

문제: AWS CDK 배포 작업 또는 AWS CDK 부트스트랩 작업이 오류와 함께 실패합니다. `npm install` 이 오류는 작업에서 액세스할 수 없는 프라이빗 노드 패키지 관리자 (npm) 레지스트리에 AWS CDK 앱 종속성을 저장하고 있기 때문에 발생할 수 있습니다.

가능한 해결 방법: 다음 지침에 따라 추가 레지스트리 및 인증 정보로 AWS CDK 앱 `cdk.json` 파일을 업데이트하십시오.

시작하기 전 준비 사항

1. 인증 정보를 위한 시크릿을 생성하세요. 이러한 비밀은 일반 텍스트로 입력하는 대신 `cdk.json` 파일에서 참조해야 합니다. 시크릿을 만들려면:
   a. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
   b. 프로젝트를 선택합니다.
   c. 탐색 창에서 CI/CD를 선택한 다음 시크릿을 선택합니다.
   d. 다음 속성을 사용하여 두 개의 시크릿을 생성하십시오.

첫 번째 비밀	두 번째 비밀
이름: `npmUsername` 값: *`npm-username`* , 어디서 *`npm-username`* 프라이빗 npm 레지스트리를 인증하는 데 사용되는 사용자 이름입니다. (선택 사항) 설명: `The username used to authenticate to the private npm registry.`	이름: `npmAuthToken` 값: *`npm-auth-token`* , 어디서 *`npm-auth-token`* 프라이빗 npm 레지스트리를 인증하는 데 사용되는 액세스 토큰입니다. npm 액세스 토큰에 대한 자세한 내용은 npm 설명서의 액세스 토큰 정보를 참조하십시오.

첫 번째 비밀	두 번째 비밀
	(선택 사항) 설명: `The access token used to authenticate to the private npm registry.`

비밀에 대한 자세한 내용은 을 참조하십시오비밀을 사용한 데이터 마스킹.

2. 시크릿을 AWS CDK 액션에 환경 변수로 추가하십시오. 액션은 실행 시 변수를 실제 값으로 대체합니다. 시크릿 추가하기:

   a. 탐색 창에서 CI/CD를 선택한 다음 워크플로를 선택합니다.

   b. 워크플로의 이름을 선택합니다. 워크플로가 정의된 소스 리포지토리 또는 브랜치 이름을 기준으로 필터링하거나 워크플로 이름 또는 상태별로 필터링할 수 있습니다.

   c. 편집을 선택합니다.

   d. Visual을 선택합니다.

   e. 워크플로 다이어그램에서 AWS CDK 작업을 선택합니다.

   f. 입력 탭을 선택합니다.

   g. 다음 속성을 가진 두 변수를 추가합니다.

첫 번째 변수	두 번째 변수
이름: `NPMUSER`	이름: `NPMTOKEN`
값: `${Secrets.npmUsername}`	값: `${Secrets.npmAuthToken}`

이제 비밀에 대한 참조를 포함하는 변수가 두 개 생겼습니다.

워크플로 정의 파일 YAML 코드는 다음과 비슷해야 합니다.

> **Note**
>
> 다음 코드 샘플은 AWS CDK 부트스트랩 작업에서 가져온 것으로, AWS CDK 배포 작업도 비슷합니다.

```
Name: CDK_Bootstrap_Action
SchemaVersion: 1.0
Actions:
  CDKBootstrapAction:
    Identifier: aws/cdk-bootstrap@v1
    Inputs:
      Variables:
        - Name: NPMUSER
          Value: ${Secrets.npmUsername}
        - Name: NPMTOKEN
          Value: ${Secrets.npmAuthToken}
      Sources:
        - WorkflowSource
    Environment:
      Name: Dev2
      Connections:
        - Name: account-connection
          Role: codecatalystAdmin
    Configuration:
      Parameters:
        Region: "us-east-2"
```

이제 cdk.json 파일에서 NPMUSER 및 NPMTOKEN 변수를 사용할 준비가 되었습니다. 다음 절차로 이동합니다.

cdk.json 파일을 업데이트하려면

1. AWS CDK 프로젝트의 루트 디렉터리로 변경한 다음 파일을 엽니다. cdk.json
2. "app":속성을 찾아 다음과 같은 코드를 포함하도록 변경합니다.*red italics*:

> **Note**
>
> 다음 샘플 코드는 TypeScript 프로젝트에서 가져온 것입니다. JavaScript 프로젝트를 사용하는 경우 코드는 동일하지는 않지만 비슷해 보일 것입니다.

```
{
```

```
  "app": "npm set registry=https://your-registry/folder/CDK-package/ --
userconfig .npmrc && npm set //your-registry/folder/CDK-package/:always-auth=true
 --userconfig .npmrc && npm set //your-registry/folder/CDK-package/:_authToken=
\"${NPMUSER}\":\"${NPMTOKEN}\" && npm install  && npx ts-node --prefer-ts-exts bin/
hello-cdk.ts|js",
  "watch": {
    "include": [
      "**"
    ],
    "exclude": [
      "README.md",
      "cdk*.json",
      "**/*.d.ts",
      "**/*.js",
      "tsconfig.json",
      "package*.json",
...
```

3. 에서 강조 표시된 코드에서 *red italics*, 다음을 대체하십시오.

   - *your-registry/folder/CDK-package/* 사설 레지스트리의 AWS CDK 프로젝트 종속성 경로를 사용하십시오.
   - *hello-cdk.ts|.js* 진입점 파일 이름과 함께. 사용 중인 언어에 따라 `.ts` (TypeScript) 또는 `.js` (JavaScript) 파일일 수 있습니다.

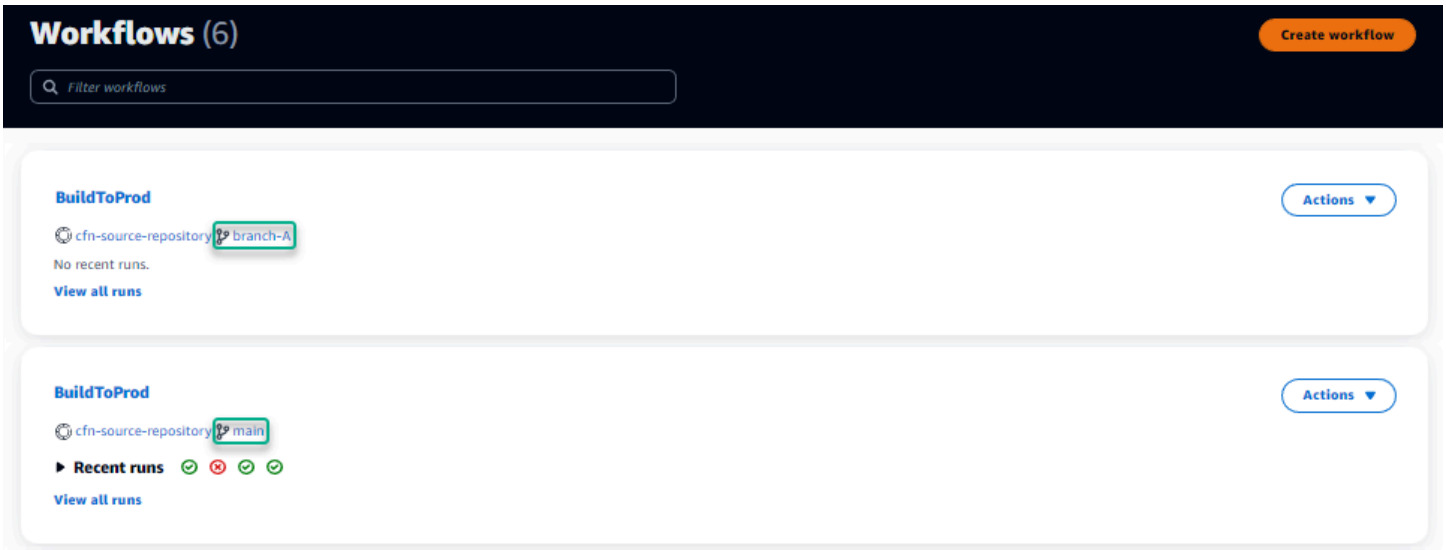
   > **Note**
   >
   > 액션은 다음을 대체합니다.*NPMUSER* 그리고 *NPMTOKEN* 변수는 Secrets에서 지정한 npm 사용자 이름과 액세스 토큰으로 구성됩니다.

4. `cdk.json` 파일을 저장합니다.
5. 작업을 수동으로 다시 실행하여 변경 사항으로 오류가 해결되는지 확인합니다. 액션을 수동으로 실행하는 방법에 대한 자세한 내용은 을 참조하십시오워크플로우 수동 실행 시작.

여러 워크플로의 이름이 같은 이유는 무엇입니까?

워크플로는 지점별, 리포지토리별로 저장됩니다. 두 개의 워크플로가 서로 다른 브랜치에 있는 경우 동일한 이름을 가질 수 있습니다. 워크플로 페이지에서 브랜치 이름을 보면 같은 이름의 워크플로를

구분할 수 있습니다. 자세한 내용은 Amazon에서 브랜치를 사용하여 소스 코드 작업을 정리하세요. CodeCatalyst 단원을 참조하십시오.



워크플로 정의 파일을 다른 폴더에 저장할 수 있나요?

아니요. 모든 워크플로 정의 파일은 폴더 또는 해당 `.codecatalyst/workflows` 폴더의 하위 폴더에 저장해야 합니다. 여러 논리적 프로젝트가 포함된 mono repo를 사용하는 경우 모든 워크플로 정의 파일을 `.codecatalyst/workflows` 폴더 또는 하위 폴더 중 하나에 배치한 다음 트리거 내의 Files changed 필드 (시각적 편집기) 또는 `FilesChanged` 속성 (YAML편집기) 을 사용하여 지정된 프로젝트 경로에서 워크플로를 자동으로 트리거합니다. 자세한 내용은 워크플로에 트리거 추가 및 예: 푸시, 브랜치, 파일이 있는 트리거 단원을 참조하세요.

워크플로에 순서대로 작업을 추가하려면 어떻게 해야 하나요?

기본적으로 작업을 워크플로에 추가하면 종속성이 없으며 다른 작업과 병렬로 실행됩니다.

작업을 순서대로 정렬하려는 경우 필드를 설정하여 다른 작업에 대한 종속성을 설정할 수 있습니다. `DependsOn` 다른 액션의 출력인 아티팩트 또는 변수를 사용하도록 액션을 구성할 수도 있습니다. 자세한 내용은 시퀀싱 액션 단원을 참조하십시오.

워크플로가 검증에 성공했지만 런타임에 실패하는 이유는 무엇입니까?

`Validate`버튼을 사용하여 워크플로를 검증했지만 워크플로가 실패했다면 검증기의 제한 때문일 수 있습니다.

워크플로 구성의 비밀, 환경 또는 플릿과 같은 CodeCatalyst 리소스와 관련된 오류는 커밋 중에 등록되지 않습니다. 유효하지 않은 참조를 사용하는 경우 워크플로 실행 시에만 오류가 식별됩니다. 마찬가지

로, 필수 필드 누락이나 작업 속성의 오타와 같은 작업 구성에 오류가 있는 경우 워크플로가 실행될 때만 식별됩니다. 자세한 내용은 워크플로 생성 단원을 참조하십시오.

자동 검색을 수행해도 내 작업에 대한 보고서가 검색되지 않습니다.

문제: 테스트를 실행하는 작업에 대해 자동 검색을 구성했지만 에서 보고서가 검색되지 않았습니다. CodeCatalyst

가능한 해결 방법: 이 문제는 여러 가지 문제로 인해 발생할 수 있습니다. 다음 해결 방법 중 하나 이상을 시도해 보십시오.

- 테스트를 실행하는 데 사용된 도구가 CodeCatalyst 이해할 수 있는 형식 중 하나로 출력을 생성하는지 확인하십시오. 예를 들어, 테스트 및 코드 커버리지 보고서를 검색할 수 있게 `pytest` CodeCatalyst 하려면 다음 인수를 포함하세요.

  ```
  --junitxml=test_results.xml --cov-report xml:test_coverage.xml
  ```

  자세한 내용은 품질 보고서 유형 단원을 참조하십시오.
- 출력의 파일 확장자가 선택한 형식과 일치하는지 확인하십시오. 예를 들어 결과를 `JUnitXML` 형식으로 `pytest` 생성하도록 구성하는 경우 파일 확장자가 인지 확인하십시오`.xml`. 자세한 내용은 품질 보고서 유형 단원을 참조하십시오.
- 특정 폴더를 일부러 제외하지 않는 한 전체 파일 시스템 (`**/*`) 을 포함하도록 구성해야 합니다. `IncludePaths` 마찬가지로, 보고서가 위치할 것으로 예상되는 디렉토리를 `ExcludePaths` 제외하지 않도록 하세요.
- 특정 출력 파일을 사용하도록 보고서를 수동으로 구성한 경우 자동 검색에서 제외됩니다. 자세한 내용은 품질 보고서 YAML 예제 단원을 참조하십시오.
- 출력이 생성되기 전에 작업이 실패했기 때문에 자동 검색에서 보고서를 찾지 못할 수 있습니다. 예를 들어 단위 테스트를 실행하기 전에 빌드가 실패했을 수 있습니다.

성공 기준을 구성한 후 자동 검색된 보고서에서 내 작업이 실패합니다.

문제: 자동 검색을 활성화하고 성공 기준을 구성하면 일부 보고서가 성공 기준을 충족하지 않아 작업이 실패합니다.

가능한 해결 방법: 이 문제를 해결하려면 다음 해결 방법 중 하나 이상을 시도해 보십시오.

- 관심이 없는 보고서를 `IncludePaths` 수정하거나 `ExcludePaths` 제외하십시오.

- 모든 보고서가 통과되도록 성공 기준을 업데이트하십시오. 예를 들어 회선 커버리지가 50% 이고 다른 하나는 70% 인 보고서가 두 개 발견된 경우 최소 회선 커버리지를 50% 로 조정하십시오. 자세한 내용은 성공 기준 단원을 참조하세요.
- 실패한 보고서를 수동으로 구성된 보고서로 전환하십시오. 이렇게 하면 특정 보고서에 대해 다양한 성공 기준을 구성할 수 있습니다. 자세한 내용은 보고서의 성공 기준 구성 단원을 참조하십시오.

자동 검색을 통해 원하지 않는 보고서가 생성됩니다.

문제: 자동 검색을 활성화하면 원하지 않는 보고서가 생성됩니다. 예를 들어, CodeCatalyst 는 내 애플리케이션의 종속 항목에 저장된 파일에 대한 코드 커버리지 보고서를 생성합니다. `node_modules`

가능한 해결 방법: 원하지 않는 파일을 제외하도록 `ExcludePaths` 구성을 조정할 수 있습니다. 예를 `node_modules` 들어 제외하려면 추가하십시오`node_modules/**/*`. 자세한 내용은 경로 포함/제외 단원을 참조하십시오.

자동 검색은 단일 테스트 프레임워크에 대해 여러 개의 작은 보고서를 생성합니다.

문제: 특정 테스트 및 코드 커버리지 보고 프레임워크를 사용할 때 자동 검색으로 인해 많은 수의 보고서가 생성되는 것을 발견했습니다. 예를 들어 Maven Surefire 플러그인을 사용하는 경우 자동 검색은 각 테스트 클래스에 대해 다른 보고서를 생성합니다.

가능한 해결 방법: 프레임워크가 출력을 단일 파일로 집계할 수 있습니다. 예를 들어 Maven Surefire 플러그인을 사용하는 경우 파일을 수동으로 `npx junit-merge` 집계하는 데 사용할 수 있습니다. 전체 표현식은 다음과 같을 수 있습니다.

```
mvn test; cd test-package-path/surefire-reports && npx junit-merge -d ./ && rm
 *Test.xml
```

CI/CD에 나열된 워크플로가 소스 리포지토리의 워크플로와 일치하지 않습니다.

문제: CI/CD, 워크플로 페이지에 표시된 워크플로가 소스 리포지토리의 `~/.codecatalyst/workflows/` 폴더에 있는 워크플로와 일치하지 않습니다. 다음과 같은 불일치가 나타날 수 있습니다.

- 워크플로가 워크플로 페이지에 나타나지만 소스 저장소에는 해당 워크플로 정의 파일이 없습니다.
- 워크플로 정의 파일은 소스 저장소에 있지만 해당 워크플로우는 워크플로 페이지에 나타나지 않습니다.

- 워크플로는 소스 리포지토리와 워크플로 페이지에 모두 존재하지만 둘은 다릅니다.

이 문제는 워크플로 페이지를 새로 고칠 시간이 없거나 워크플로 할당량을 초과한 경우 발생할 수 있습니다.

수정 방법:

- 대기합니다. 일반적으로 원본으로 커밋한 후 2~3초 정도 기다려야 워크플로 페이지에 변경 내용이 표시됩니다.
- 워크플로 할당량을 초과한 경우 다음 중 하나를 수행하십시오.

> **Note**
>
> 워크플로 할당량을 초과했는지 확인하려면 소스 리포지토리 또는 워크플로 페이지에 있는 워크플로와 비교하여 문서화된 할당량을 검토하고 의 워크플로우 할당량 CodeCatalyst 검토하십시오. 할당량을 초과했다는 오류 메시지는 없으므로 직접 조사해야 합니다.

  - 공간 할당량당 최대 워크플로 수를 초과한 경우 일부 워크플로를 삭제한 다음 워크플로 정의 파일에 대해 테스트 커밋을 수행하십시오. 테스트 커밋의 예로는 파일에 공간을 추가하는 경우를 들 수 있습니다.
  - 최대 워크플로 정의 파일 크기 할당량을 초과한 경우 워크플로 정의 파일을 변경하여 길이를 줄이십시오.
  - 단일 소스 이벤트 할당량에서 처리되는 워크플로 파일의 최대 수를 초과한 경우 테스트 커밋을 여러 번 수행하십시오. 각 커밋의 최대 워크플로 수보다 적은 수를 수정하십시오.
  - 유료 티어 청구를 활성화하여 워크플로 할당량을 늘리세요. 자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 청구 관리를 참조하십시오.

워크플로를 생성하거나 업데이트할 수 없습니다.

문제: 워크플로를 만들거나 업데이트하고 싶은데 변경 내용을 적용하려고 하면 오류가 나타납니다.

가능한 해결 방법: 프로젝트 또는 스페이스에서 맡은 역할에 따라 코드를 프로젝트의 소스 리포지토리로 푸시할 수 있는 권한이 없을 수 있습니다. 워크플로용 YAML 파일은 리포지토리에 저장됩니다. 자세한 내용은 워크플로 정의 파일 단원을 참조하십시오. 스페이스 관리자 역할, 프로젝트 관리자 역할 및 기여자 역할 모두 코드를 커밋하고 프로젝트의 저장소에 푸시할 수 있는 권한을 가집니다.

기여자 역할이 있지만 특정 YAML 브랜치에서 워크플로를 만들거나 변경 내용을 커밋할 수 없는 경우, 해당 분기에 대해 해당 역할을 가진 사용자가 특정 브랜치로 코드를 푸시하지 못하도록 하는 분기 규칙이 구성되어 있을 수 있습니다. 다른 브랜치에서 워크플로를 만들거나 변경 내용을 다른 브랜치에 커밋해 보세요. 자세한 내용은 브랜치 규칙을 사용하여 브랜치에 허용된 작업 관리 단원을 참조하십시오.

문제 관련 문제 해결

다음 정보는 에서 발생하는 일반적인 문제를 해결하는 데 도움이 될 수 있습니다. CodeCatalyst

주제

- 내 문제의 담당자를 선택할 수 없어요

내 문제의 담당자를 선택할 수 없어요

문제: 이슈를 생성할 때 담당자 목록이 비어 있습니다.

가능한 해결 방법: 담당자 목록은 프로젝트의 구성원으로 등록된 CodeCatalyst 사용자와 직접 연결됩니다. 사용자 프로필 액세스가 제대로 작동하는지 확인하려면 프로필 아이콘을 선택한 다음 사용자 프로필을 선택합니다. 사용자 프로필 정보가 입력되지 않는 경우 상태 보고서에서 문제가 있는지 확인하세요. 정보가 모두 채워지면 서비스 티켓을 제출하세요.

검색 관련 문제 해결 CodeCatalyst

검색과 관련된 문제를 해결하려면 다음 섹션을 참조하십시오. CodeCatalyst 워크플로에 대한 자세한 내용은 에서 코드, 이슈, 프로젝트, 사용자 검색 CodeCatalyst 섹션을 참조하세요.

주제

- 내 프로젝트에서 사용자를 찾을 수 없어요
- 내 프로젝트 또는 스페이스에서 원하는 내용을 찾을 수 없습니다.
- 페이지를 탐색할 때 검색 결과 수가 계속 변경됩니다.
- 검색 쿼리가 완료되지 않아요

내 프로젝트에서 사용자를 찾을 수 없어요

문제: 사용자의 세부 정보를 보려고 할 때 프로젝트에 해당 정보가 표시되지 않습니다.

가능한 해결 방법: 현재 검색은 프로젝트 내에서 사용자를 검색하는 기능을 지원하지 않습니다. 스페이스에 액세스할 수 있는 사용자를 검색하려면 이 스페이스 인으로 전환하거나 고급 쿼리 언어를 사용하여 지정했을 수 있는 프로젝트 필터를 삭제하세요. QuickSearch

내 프로젝트 또는 스페이스에서 원하는 내용을 찾을 수 없습니다.

문제: 특정 정보를 검색하려고 할 때 결과가 표시되지 않습니다.

가능한 해결 방법: 콘텐츠 업데이트가 검색결과에 업데이트되는 데 몇 초 정도 걸릴 수 있습니다. 대규모 업데이트는 몇 분 정도 걸릴 수 있습니다.

최근에 업데이트되지 않은 리소스의 경우 검색을 구체화해야 할 수 있습니다. 키워드를 더 추가하거나 고급 쿼리 언어를 사용하여 검색 범위를 좁힐 수 있습니다. 쿼리 세분화에 대한 자세한 내용은 을 참조하십시오. 검색 쿼리 구체화하기

페이지를 탐색할 때 검색 결과 수가 계속 변경됩니다.

문제: 다음 페이지로 이동하면 검색 결과 수가 변경되는 것처럼 보이기 때문에 총 결과 수가 몇 개인지 명확하지 않습니다.

가능한 해결 방법: 검색 결과 페이지를 탐색할 때 쿼리와 일치하는 검색 결과 수가 변경될 수 있습니다. 페이지를 탐색하면서 발견된 일치 항목의 수를 더 정확하게 반영하도록 결과 수가 업데이트될 수 있습니다.

결과를 탐색할 때 “테스트”에 대한 결과 없음 메시지가 표시될 수 있습니다. 나머지 결과에 액세스할 수 없는 경우 메시지를 받게 됩니다.

검색 쿼리가 완료되지 않아요

문제: 검색어 결과가 표시되지 않고 너무 오래 걸리는 것 같습니다.

가능한 해결 방법: 프로그래밍 방식으로 또는 팀 활동이 많아서 스페이스에서 동시에 많은 검색을 수행하는 경우 검색이 완료되지 않을 수 있습니다. 프로그래밍 방식의 검색을 실행하는 경우 검색을 일시 중지하거나 줄이세요. 그렇지 않으면 몇 초 후에 다시 시도하세요.

확장 프로그램 관련 문제 해결

의 확장 프로그램과 관련된 문제를 해결하려면 다음 섹션을 참조하십시오. CodeCatalyst 확장에 대한 자세한 내용은 을 참조하십시오확장 기능을 사용하여 프로젝트에 기능 추가 CodeCatalyst.

주제

연결된 타사 리포지토리의 변경 내용을 볼 수 없거나 변경 결과를 검색할 수 없습니다.

문제: 타사 저장소의 변경 내용이 에 표시되지 않습니다. CodeCatalyst

가능한 해결 방법: CodeCatalyst 현재 연결된 리포지토리의 기본 브랜치 변경 감지를 지원하지 않습니다. 연결된 저장소의 기본 분기를 변경하려면 먼저 연결을 해제하고 기본 분기를 변경한 다음 다시 연결해야 합니다. CodeCatalyst 자세한 내용은 GitHub 리포지토리, Bitbucket 리포지토리, 프로젝트 리포지토리, GitLab Jira 프로젝트 연결 CodeCatalyst을(를) 참조하세요.

스페이스와 관련된 계정 관련 문제 해결

CodeCatalyst에서는 스페이스에 를 AWS 계정 추가하여 리소스에 대한 권한을 부여하고 청구 목적으로 사용할 수 있습니다. 다음 정보는 에서 관련 계정과 관련된 일반적인 문제를 해결하는 데 도움이 될 수 있습니다 CodeCatalyst.

주제

AWS 계정 연결 요청에 잘못된 토큰 오류가 발생했습니다.

문제: 연결 토큰으로 연결 요청을 만들 때 페이지에서 토큰을 수락하지 않고 토큰이 유효하지 않다는 오류 메시지가 표시됩니다.

가능한 해결 방법: 스페이스에 추가하려는 계정 ID를 제공해야 합니다. 계정에 대한 관리자 권한이 AWS 계정 있거나 관리자와 협력하여 계정을 추가할 수 있어야 합니다.

계정을 확인하도록 선택하면 에서 새 브라우저 창이 열립니다 AWS Management Console. 콘솔 측에서 로그인하려면 동일한 계정이 필요합니다. 다음을 확인한 후 다시 시도하세요.

- 공간에 추가하려는 AWS 계정 것과 동일한 AWS Management Console 계정으로 로그인했습니다.
- 지역이 해당 공간에 맞는 지역으로 설정된 AWS Management Console 상태에서 에 로그인했습니다.
- 청구 페이지에서 도착하여 공간을 위한 지정된 결제 계정으로 추가하려는 경우 해당 계정이 다른 공간 또는 스페이스에 대한 결제 계정 할당량에 도달하지 않았는지 확인하세요. AWS 계정

구성된 계정, 환경 또는 IAM 역할에 대한 오류가 발생하여 Amazon CodeCatalyst 프로젝트 워크플로가 실패합니다.

문제: 워크플로가 실행되고 스페이스와 관련된 구성된 계정 또는 IAM 역할을 찾지 못하면 워크플로에서 역할, 연결 및 환경 필드를 수동으로 입력해야 YAML 합니다. 실패한 워크플로 작업을 확인하고 오류 메시지가 다음과 같은지 확인하십시오.

- 환경과 관련된 연결에는 역할을 사용할 수 없습니다.
- 작업이 성공하지 못했습니다. 상태:FAILED; 계정 연결 또는 환경에 대해 제공된 값이 유효하지 않습니다. 연결이 공간과 연결되어 있고 환경이 프로젝트와 연결되어 있는지 확인하십시오.
- 작업이 성공하지 못했습니다. 상태:FAILED; 제공된 IAM 역할 값이 유효하지 않습니다. 이름이 존재하고, IAM 역할이 계정 연결에 추가되고, 연결이 이미 Amazon CodeCatalyst 스페이스와 연결되어 있는지 확인합니다.

가능한 해결 방법: 워크플로 YAML 필드의 환경, 연결 및 역할 값이 정확한지 확인하십시오. 환경이 필요한 CodeCatalyst 워크플로 작업은 AWS 리소스를 실행하거나 AWS 리소스 스택을 생성하는 빌드 또는 배포 작업입니다.

실패한 워크플로 작업 블록을 선택한 다음 Visual을 선택합니다. 구성 탭을 선택합니다. 환경, 연결 이름 및 역할 이름 필드가 채워지지 않은 경우 워크플로를 수동으로 업데이트해야 합니다. 다음 단계를 사용하여 YAML 워크플로를 편집하십시오.

- `/.codecatalyst`디렉터리를 확장한 다음 `/workflows` 디렉터리를 확장합니다. 워크플로우 YAML 파일을 엽니다. 워크플로에 맞게 구성한 내용에 IAM 역할 및 계정 정보가 지정되어 있는지 확인하십시오. YAML 예제:

```
Actions:
```

```
cdk_bootstrap:
  Identifier: action-@v1
  Inputs:
    Sources:
      - WorkflowSource
  Environment:
    Name: Staging
    Connections:
      - Name: account-connection
        Role: build-role
```

환경, 연결 및 역할 속성은 AWS 리소스가 포함된 CodeCatalyst 워크플로 빌드 및 배포 작업을 실행하는 데 필요합니다. 예제는 환경, 연결, 역할에 대한 CodeCatalyst 빌드 작업 참조 YAML 파라미터를 참조하세요.

- 스페이스에 계정이 추가되었는지 확인하고 계정에 적절한 역할 또는 IAM 역할이 계정에 추가되었는지 확인하세요. 스페이스 관리자 역할이 있는 경우 계정을 조정하거나 추가할 수 있습니다. 자세한 내용은 연결된 AWS 리소스에 대한 액세스 허용 AWS 계정 단원을 참조하십시오.

프로젝트를 생성하려면 관련 계정, 역할, 환경이 필요합니다.

문제: 프로젝트 생성 옵션에서 내 프로젝트에 내 스페이스에서 사용할 수 있는 추가 계정이 없거나, 프로젝트에서 사용하려면 스페이스에 다른 계정을 추가해야 합니다.

가능한 해결 방법: 스페이스 관리자 역할이 있는 경우 스페이스의 경우 프로젝트에 추가할 권한이 AWS 계정 있는 사용자를 추가할 수 있습니다. 또한 AWS 관리자 권한이 있거나 관리자와 함께 작업할 수 있는 공간이 있어야 합니다. AWS 계정

프로젝트 생성 화면에서 계정과 역할을 사용할 수 있게 하려면 먼저 계정과 역할을 추가해야 합니다. 자세한 내용은 연결된 AWS 리소스에 대한 액세스 허용 AWS 계정 단원을 참조하십시오.

라는 역할 정책을 사용하여 서비스 역할을 생성하도록 선택할 수 있습니다. CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 정책. 역할에는 고유 식별자가 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 추가된 이름이 지정됩니다. 역할 및 역할 정책에 대한 자세한 내용은 을 참조하십시오CodeCatalystWorkflowDevelopmentRole-***spaceName***서비스 역할 이해. 역할 생성 단계는 을 참조하십시오계정 및 스페이스에 대한 CodeCatalystWorkflowDevelopmentRole-***spaceName***역할 만들기. 역할은 계정에 추가되며 의 프로젝트 생성 페이지에서 사용할 수 CodeCatalyst 있습니다.

Amazon CodeCatalyst Spaces 페이지에 액세스할 수 없습니다. AWS Management Console

문제: 내 CodeCatalyst 스페이스에 계정을 추가하거나 내 계정에 역할을 추가하기 위해 의 AWS Amazon CodeCatalyst 페이지에 액세스하려고 하면 권한 오류가 발생합니다. AWS Management Console

수정 방법:

스페이스 관리자 역할이 있는 경우 스페이스의 경우 프로젝트에 AWS 계정 추가할 수 있는 권한을 추가할 수 있습니다. 또한 AWS 관리자 권한이 있거나 관리자와 함께 작업할 수 있는 공간이 있어야 합니다. AWS 계정 먼저 관리하려는 AWS Management Console 계정과 동일한 계정으로 로그인했는지 확인해야 합니다. 에 로그인한 후 콘솔을 AWS Management Console열고 다시 시도할 수 있습니다.

AWS Management Console 앳 https://us-west-2.console.aws.amazon.com/codecatalyst/홈에서 Amazon CodeCatalyst 페이지를 여시겠습니까? 지역=미국-서부-2#/.

다른 계정을 결제 계정으로 사용하고 싶어요

문제: CodeCatalyst 로그인을 설정할 때 공간을 설정하고 승인된 사람과 연결하는 몇 가지 단계를 AWS 계정완료했습니다. 이제 청구를 위해 다른 계정을 승인하고 싶습니다.

가능한 해결 방법: 스페이스의 경우 스페이스 관리자 역할이 있는 경우 결제 계정을 승인할 수 있습니다. 또한 AWS 관리자 권한이 있거나 관리자와 협력할 수 있는 AWS 계정 곳이 있어야 합니다.

자세한 내용은 Amazon CodeCatalyst 관리자 안내서의 청구 관리를 참조하십시오.

연결 이름 오류로 인해 프로젝트 워크플로가 실패합니다.

문제: 프로젝트를 만든 다음 프로젝트 워크플로를 실행하면 워크플로가 실패하고 다음과 같이 연결 이름이 유효하지 않다는 오류 메시지가 표시됩니다.

실패: 연결 이름이 유효하지 않습니다. <action_name>

가능한 해결 방법: 스페이스에 추가하려는 계정 ID를 제공하고 계정이 프로젝트 제한 계정 연결에 사용할 수 있도록 설정되어 있지 않은지 확인하세요. 계정이 프로젝트 제한 계정 연결을 사용할 수 있도록 설정된 경우 새 프로젝트에 대한 액세스를 허용하여 계정 연결을 업데이트해야 할 수 있습니다. 자세한 내용은 프로젝트 제한 계정 연결 구성을 참조하십시오.

CodeCatalyst Amazon과 AWS SDK 간의 문제 해결 또는 AWS CLI

다음 정보는 CodeCatalyst 및 AWS CLI 또는 SDK를 사용할 때 흔히 발생하는 문제를 해결하는 데 도움이 될 수 있습니다. AWS

주제

명령줄이나 터미널에 입력할 aws codecatalyst 때 잘못된 선택이라는 오류 메시지가 나타납니다.

문제: AWS CLI 와 함께 CodeCatalyst 사용하려고 할 때 하나 이상의 aws codecatalyst 명령이 유효한 것으로 인식되지 않습니다.

해결 방법: 이 문제의 가장 일반적인 원인은 최신 서비스 및 명령에 대한 최신 업데이트가 포함되지 AWS CLI 않은 버전을 사용하고 있기 때문입니다. 의 설치를 AWS CLI 업데이트한 다음 다시 시도하십시오. 자세한 내용은 AWS CLI와 함께 사용하기 위한 설정 CodeCatalyst 섹션을 참조하세요.

aws codecatalyst명령을 실행할 때 자격 증명 오류가 발생합니다.

문제: AWS CLI 와 함께 CodeCatalyst 사용하려고 하면 `You can configure credentials by running "aws configure".` 또는 라는 메시지가 나타납니다`Unable to locate authorization token.`

해결 방법: CodeCatalyst 명령과 함께 작동하도록 AWS CLI 프로필을 구성해야 합니다. 자세한 내용은 AWS CLI와 함께 사용하기 위한 설정 CodeCatalyst을 참조하세요.

상태 보고서를 통한 현재 서비스 상태 CodeCatalyst 파악

Amazon CodeCatalyst Health Report는 광범위한 영향을 미치는 서비스의 리소스 성능 및 가용성과 관련된 up-to-the-minute 알림 목록을 사용자에게 집계하여 제공하는 공개 대시보드입니다. CodeCatalyst 어떤 리소스에 문제가 있고 애플리케이션에 영향을 미칠 수 있는지 확인할 수 있습니다. CodeCatalyst 이를 통해 시스템 전반에서 운영 중단 및 기타 리소스 가동 중지 시간을 추적할 수 있습니다. 사고가 발생하면 상태 보고서 아이콘에 파란색 표시기가 나타납니다. 또한 프로젝트에서 스페이스 관리자 역할을 가진 모든 사용자에게 경고 및 이메일 알림을 CodeCatalyst 자동으로 전송하여 사고의 세부 정보와 기록을 거의 실시간으로 제공합니다.

대시보드는 모든 활성 이벤트 목록과 지난 30일 동안 발생한 최대 100개의 이전 사고 기록을 제공합니다. 인시던트가 업데이트된 날짜를 기준으로 인시던트 목록을 구성할 수 있습니다. 또한 인시던트 목록을 새로 고쳐 최신 업데이트를 받을 수 있습니다.

CodeCatalyst 상태 보고서를 사용할 수 있는 워크플로는 다음과 같습니다.

마테오 잭슨은 스페이스 관리자 권한을 가진 버딩 스페이스의 개발자입니다. 풀 리퀘스트를 만들려고 하면 계속 오류 메시지가 뜹니다. 그는 이메일을 확인하다가 자신의 공간에 영향을 미친 시스템 문제에 대한 자세한 기록을 CodeCatalyst 제공하면서 자동 생성된 시스템 사고 이메일을 받았다는 사실을 알게 됩니다. 업데이트 보기를 선택하면 시스템에서 보고된 모든 사고를 볼 수 있는 CodeCatalyst 상태 보고서로 이동합니다. 그는 목록에서 해당 사건을 선택하여 자세한 정보를 찾아냅니다. 분할 화면이 열리고 마지막 업데이트의 타임스탬프, 기록, 영향을 받은 기능, 시작 시간 및 인시던트의 현재 상태가 표시됩니다. 또한 문제가 진행 중이라는 사실도 알 수 있지만 서비스 팀이 문제를 해결하기 시작했습니다. 사고 기록이나 상태가 업데이트될 때마다 그는 이메일을 받습니다. 이메일에 액세스할 수 없는 경우 상단 패널의 종 모양 아이콘을 선택하여 CodeCatalyst 건강 보고서로 이동할 수 있습니다.

CodeCatalyst 건강 보고서 개념

다음 개념을 익히면 CodeCatalyst 상태 보고서를 이해하고 이를 통해 애플리케이션, 서비스 및 리소스의 상태를 추적하는 방법을 이해하는 데 도움이 됩니다.

인시던트

인시던트는 내부 애플리케이션과 리소스에 영향을 미치는 시스템 이벤트입니다 CodeCatalyst. 인시던트를 선택하여 이벤트가 시작된 시간, 서비스 팀에서 해결 작업을 진행 중인지 등 이벤트의 자세한 기록을 볼 수 있습니다.

상태 표시기

상태는 사고의 실시간 상태입니다. 진행 중 또는 해결됨으로 표시됩니다.

영향을 받는 기능

영향을 받는 기능은 사고의 영향을 받는 리소스 또는 애플리케이션입니다. 단일 인시던트는 풀 리퀘스트, 이슈, 워크플로, 테스트, 배포, 소스 등 시스템의 여러 영역에 영향을 미칠 수 있습니다.

에 업데이트됨

업데이트 날짜는 해당 인시던트에 대한 마지막 업데이트의 타임스탬프를 제공합니다.

AWS Support아마존용 CodeCatalyst

스페이스를 생성할 때는 를 AWS 계정 연결하고 스페이스의 결제 계정으로 지정해야 합니다. 결제 계정으로 AWS 계정 지정한 계정에서 Amazon AWS Support CodeCatalyst 요금제에 액세스할 수도 있습니다. 지원이 필요한 경우 지정된 AWS 계정 지원 사례를 바탕으로 지원 사례를 생성할 수 있습니다.

CodeCatalyst 스페이스의 사용자는 AWS Support for Amazon CodeCatalyst 페이지를 사용하여 지원 사례를 관리합니다. CodeCatalyst 비즈니스 지원 또는 Enterprise Support와 같은 AWS Support 플랜으로 업그레이드하여 에서 CodeCatalyst 기술 지원 사례를 생성하고 관리할 수 CodeCatalyst 있습니다. 지원 사례에 대한 지원은 전화, 웹 또는 채팅을 통해 제공됩니다.

CodeCatalyst 서비스 및 리소스와 관련된 사례만 AWS Support for Amazon을 통해 지원할 수 CodeCatalyst 있습니다. CodeCatalyst 리소스에는 내부 CodeCatalyst 및 사용자에 의해 배포된 리소스가 포함되지만 다른 서비스 AWS 또는 타사 서비스를 위해 배포된 리소스는 여기에 포함되지 않습니다. CodeCatalyst 다른 서비스에 대한 지원이 필요한 경우 를 통해 해당 AWS 서비스를 열어야 합니다 AWS Management Console.

지원 플랜을 변경하려면 지원 플랜 변경을 참조하십시오.

> **Note**
>
> Developer Support 플랜은 프로덕션 환경을 위해 설계되지 않았습니다. 스페이스 결제 계정에 Developer Support 플랜이 있는 경우 이 플랜은 해당 플랜의 모든 스페이스 관리자와 스페이스 구성원에게 계단식으로 적용되지 않습니다. AWS Support CodeCatalyst

아마존 AWS Support 청구 CodeCatalyst

에서 CodeCatalyst 스페이스를 생성하면 해당 스페이스의 사용자가 AWS Support Amazon용 지원 사례를 생성하고 관리할 수 CodeCatalyst 있습니다. 두 가지 유형의 고객 사례를 생성할 수 있습니다.

- 계정 및 청구 지원 사례는 스페이스의 모든 CodeCatalyst 사용자가 사용할 수 있습니다. 에서 사용자의 권한에 따라 청구 및 계정 질문에 대한 도움을 받을 수 CodeCatalyst 있습니다.
- 기술 지원 사례를 통해 기술 지원 엔지니어와 연결하여 서비스 관련 기술 문제 및 타사 응용 프로그램 확장에 대한 도움을 받을 수 있습니다. 기본 지원 플랜에 가입한 경우 기술 지원 사례를 생성할 수 없습니다.

공간 결제 계정으로 AWS 계정 지정된 AWS Support CodeCatalyst 사용자에게는 기술 사례에 사용할 공간에 대한 비즈니스 지원 또는 Enterprise Support 플랜이 있어야 합니다.

> **Note**
>
> 비즈니스 지원 또는 Enterprise Support 플랜이 없는 계정의 공간을 AWS Support AWS Support Amazon용으로 사용하는 경우에도 계정 및 청구 사례에 대해서는 CodeCatalyst CodeCatalyst Amazon에서 계속 사용할 수 있습니다.

기술 지원을 받으려면 CodeCatalyst 콘솔을 통해 모든 케이스를 열어야 합니다. CodeCatalyst AWS Support에서 에 대한 기술 지원 사례를 생성할 수 없습니다AWS Management Console.

> **Note**
>
> Amazon에서는 서비스 한도 증가 AWS Support 요청을 받을 수 없습니다 CodeCatalyst. 이러한 요청은 의 스페이스 결제 계정에 대한 루트 사용자만 제출할 수 AWS Support Center Console 있습니다.

AWS Supportfor CodeCatalyst Amazon은 다음과 같은 AWS Support 사항을 고려하여 동일한 지원 계약을 체결했습니다.

- 의 지원 사례에는 심각도 선택에 자세히 설명된 대로 심각도 목록 CodeCatalyst, 응답 시간 및 SLA가 AWS Support 적용됩니다. AWS Support
- 스페이스 관리자와 스페이스 구성원은 Slack의 AWS Support API, AWS SDK 또는 AWS Support 앱을 사용하여 사례를 만들 수 없습니다. CodeCatalyst CodeCatalyst 지원 사례는 다음에서만 제출할 수 있습니다. CodeCatalyst

> **Note**
>
> CodeCatalyst AWS사고 탐지 및 대응과 AWS Trusted Advisor 완전히 통합되지 않았습니다. 통합 방식을 CodeCatalyst 검증하여 비즈니스 관행이 현재 통합과 일치하는지 확인하십시오.

지원을 요청하려는 스페이스의 사용자여야 합니다.

> **Note**
>
> 숙소에 빌더가 두 명 이상 있는 경우 비즈니스 지원 또는 Enterprise Support 플랜을 구매하는 것이 좋습니다. 이 플랜은 최대 5,000명의 건축업자를 위한 공간에 대한 기술 지원을 제공합니다.

공간의 결제 계정으로 AWS 계정 지정된 계정에서는 `AWSRoleForCodeCatalystSupport` 역할 및 AmazonCodeCatalystSupportAccess관리형 정책을 사용합니다. 이렇게 하면 스페이스의 CodeCatalyst 사용자가 AWS Support for Amazon CodeCatalyst 페이지에 액세스할 수 있습니다. 이 역할 및 정책에 대한 자세한 내용은 을 참조하십시오 AmazonCodeCatalystSupportAccess. 청구에 대한 기타 고려 사항은 Amazon CodeCatalyst 관리자 안내서의 청구 관리를 참조하십시오.

빌더가 다음과 같은 지원 사례를 만들 때 가능한 흐름은 CodeCatalyst 다음과 같습니다.

마테오 잭슨은 에서 프로젝트를 진행하고 있는 개발자입니다. CodeCatalyst Amazon에서 AWS 계정 청구를 관리하는 회사에 CodeCatalyst 가입하고 Business Support 플랜으로 AWS Support 업그레이드하면 해당 공간의 모든 빌더가 기술 지원 사례를 생성할 수 있습니다. Mateo는 프로젝트에서 실패한 워크플로에 대한 기술 지원 사례를 제출합니다. Mateo는 AWS Support for Amazon CodeCatalyst 페이지를 사용하여 양식을 작성하고 사례를 생성하여 요청에 워크플로 ID 및 기타 세부 정보를 제공합니다. 사례는 사례 ID로 생성되며 청구 계정으로 AWS 계정 지정되고 공간에 대한 지원 계획과 연결된 계정 ID를 포함합니다.

모든 빌더가 for에서 AWS Support 지원 사례를 생성할 수 있지만 생성된 각 사례에 대해 비용이 청구되지는 않습니다. CodeCatalyst 스페이스 빌링 계정에서 구입한 AWS Support 프리미엄 플랜을 기반으로 케이스와 연락처를 거의 무제한으로 개설할 수 있습니다.

> **Note**
>
> 스페이스 결제 계정은 CodeCatalyst 사용자 및 리소스에 대해 AWS 계정 청구되는 계정입니다. 추가 AWS 계정 서비스에 배포한 경우 를 AWS Support 통해 다른 서비스에 배포된 리소스에 AWS Management Console 대한 지원을 요청하십시오.
> 워크플로에서 배포한 AWS 계정 대상을 식별할 수 있습니다.

아마존을 AWS Support 위한 공간 설정하기 CodeCatalyst

AWS Supportfor Amazon은 AWS Support API 통합의 일환으로 지원 사례를 CodeCatalyst 관리합니다 CodeCatalyst.

`AWSRoleForCodeCatalystSupport`역할은 스페이스의 지원 사례에 사용되는 서비스 역할입니다. 스페이스의 지정된 결제 계정에 역할을 추가해야 합니다. 자세한 내용이나 역할 생성에 대한 자세한 내용은 을 참조하십시오계정 및 스페이스에 대한 AWSRoleForCodeCatalystSupport역할 생성.

> **Note**
>
> 2023년 4월 20일 이전에 생성된 스페이스의 경우 Support for 가 스페이스에서 CodeCatalyst 작동하도록 하려면 역할을 만들어야 합니다. 2023년 4월 20일 이후에 스페이스를 만드는 경우 공간 생성 중에 의 청구 세부 정보 페이지에서 또는 의 CodeCatalyst 지원 배너 링크를 클릭하여 역할을 생성할 수 CodeCatalyst 있습니다.

공간에 대한 지원을 설정하려면

1. CodeCatalyst 스페이스를 만들 때 결제 계정을 연결하라는 안내가 표시됩니다. 공간에 지정된 결제 계정을 통해 요금이 청구됩니다. AWS 스페이스 만들기에 대한 자세한 내용은 을 참조하십시오첫 번째 스페이스 및 개발 역할 만들기 (초대 없이 시작).
2. CodeCatalyst 스페이스를 만들 때 CodeCatalyst 사용자가 지원에 액세스할 수 있는 `AWSRoleForCodeCatalystSupport` 서비스 역할을 만들 수 있는 옵션이 제공됩니다. 역할은 관리형 정책을 사용합니다`AmazonCodeCatalystSupportAccess`. 스페이스의 결제 계정으로 AWS 계정 지정된 계정에 역할을 추가해야 합니다. 이 역할 생성에 대한 자세한 내용은 계정 및 스페이스에 대한 AWSRoleForCodeCatalystSupport역할 생성 섹션을 참조하십시오.
3. 스페이스에 지정된 결제 계정의 경우 스페이스 관리자는 에 대한 비즈니스 지원 또는 Enterprise Support 플랜을 구입하는 것이 좋습니다AWS 계정. 스페이스의 모든 구성원은 Amazon의 지원 사례를 관리할 수 있으며 CodeCatalyst, 통합 AWS Support 완료 시 구매한 AWS Support 플랜에 따라 지원 채널이 조정됩니다.
4. 에서 지원 사례를 생성하고 관리하려면 을 CodeCatalyst 참조하십시오. 에서 CodeCatalyst 지원 사례 만들기 CodeCatalyst

CodeCatalyst 에 대한 지원 액세스 AWS Management Console

스페이스에 대한 지원 활성화 결제 계정 연결이 끊기면 이전 스페이스 결제 계정 및 관련 지원 플랜과 관련된 AWS Support 사례는 더 이상 AWS Support for CodeCatalyst Amazon에서 표시되지 않습니다. 해당 결제 계정의 루트 사용자는 에서 이전 사례를 보고 해결할 수 AWS Management Console 있으며 다른 사용자가 이전 사례를 보고 해결할 수 있도록 IAM 권한을 설정할 수 있습니다. AWS Support 다른 모든 지원 플랜에서 제공하는 혜택을 계속 AWS 서비스 활용하고 이전에 해결되지 않은 CodeCatalyst 지원 사례를 모두 완료할 수 있습니다. AWS Management Console

자세한 내용은 사용 설명서의 케이스 업데이트, 해결 및 재개를 참조하십시오. AWS Support

에 대한 일반적인 사용 방법 정보에 대한 지원 케이스도 에서 열 CodeCatalyst 수 있지만 이 채널을 통해 기술 지원을 받을 수는 없습니다. AWS Management Console CodeCatalyst 자세한 내용은 사용 AWS Support설명서의 지원 사례 및 사례 관리 만들기를 참조하십시오.

사용자가 지원 사례를 해결하는 데 사용할 수 있는 흐름은 다음과 같습니다. CodeCatalyst AWS Management Console

모든 빌더가 AWS Support Amazon용 지원 사례를 생성할 수 있지만 CodeCatalyst, 지원 요청은 해당 공간의 결제 계정으로 지정된 계정에서 청구됩니다. Mateo Jackson은 프로젝트 개발자로, 프로젝트에서 CodeCatalyst 실패한 워크플로에 대한 기술 지원 케이스를 신청했습니다. 하지만 Amazon에 CodeCatalyst 가입하고 Business Support 플랜을 구매한 공간의 결제 계정은 스페이스에서 연결이 끊겼습니다. AWS Support Mateo가 접수된 최신 커뮤니케이션을 보고 접수된 사례를 해결할 수 있는 유일한 방법은 AWS Support 센터의 사례 ID를 관리하는 것입니다. CodeCatalyst AWS Management Console 이를 위해 Mateo는 지원 케이스에 연결된 이전 스페이스 빌링 계정의 루트 사용자로부터 IAM 권한을 받고 콘솔을 통해 케이스를 해결합니다. AWS Support

> **Important**
>
> 스페이스에 지정된 결제 계정을 변경해도 해당 월말까지만 AWS Support 요금제를 이용할 수 있습니다. AWS Management Console AWS Support에서 이전에 만든 지원 사례를 계속 이용하려면 업데이트된 결제 계정에서 CodeCatalyst 다시 구매해야 합니다. Amazon을 통해 AWS Support 지원 사례에 액세스하는 데 영향을 미치지 않도록 스페이스 결제 계정을 변경하기 위해 모든 지원 사례를 해결할 때까지 기다리는 것이 좋습니다 CodeCatalyst.

에서 CodeCatalyst 지원 사례 만들기 CodeCatalyst

AWS SupportAmazon용 CodeCatalyst 페이지에서 지원 사례를 생성할 수 있습니다.

AWS빌더 ID는 인증된 별칭과 권한을 기반으로 한 리소스에 대한 지원만 받을 수 있습니다. 계정 및 청구 옵션은 모든 스페이스 관리자와 스페이스 구성원이 사용할 수 있습니다. 하지만 사용자는 액세스 권한이 있는 리소스에 대한 지원만 받을 수 CodeCatalyst 있으며 계정 청구 관리와 관련된 지원은 받을 수 없습니다.

스페이스의 양식 CodeCatalyst 페이지를 사용하여 CodeCatalyst 리소스에 대한 계정 및 청구 사례 또는 기술 지원 사례를 만들 수 있습니다. AWS Support

> **Note**
>
> CodeCatalyst 서비스 및 리소스와 관련된 사례만 AWS Support for Amazon을 통해 지원할 수 CodeCatalyst 있습니다. CodeCatalyst 리소스에는 내부 CodeCatalyst 및 사용자에 의해 배포된 리소스가 포함되지만 다른 서비스 AWS 또는 타사 서비스를 위해 배포된 리소스는 여기에 포함되지 않습니다. CodeCatalyst 다른 서비스에 대한 지원이 필요한 경우 를 통해 해당 AWS 서비스를 열어야 합니다AWS Management Console.

에서 지원 사례를 만들려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.

   > **Tip**
   >
   > 둘 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택합니다.

3. 페이지 상단에서 ? 를 선택합니다. 아이콘을 선택한 다음 Support를 선택합니다.
4. 사례 생성(Create case)을 선택하세요.
5. 다음 옵션 중 하나를 선택합니다.

   - 계정 및 결제(Account and billing)
   - 기술(Technical)

   > **Note**
   >
   > CodeCatalystAmazon의 AWS Support 경우 비즈니스 지원 또는 Enterprise Support 플랜이 스페이스 결제 계정에 추가되면 모든 스페이스 관리자와 스페이스 구성원이

CodeCatalyst 기술 사례 지원을 이용할 수 있습니다. 문제 해결 정보는 제 숙소에 대한 기술 지원 사례를 만들 수 없습니다.를 참조하세요.
AWS Support플랜은 여러 공간에 걸쳐 적용되지 않습니다. 여러 스페이스의 구성원인 경우 스페이스 관리자가 모든 스페이스에 대한 AWS Support 프리미엄 플랜을 구매해야 모든 스페이스에 대한 기술 지원을 받을 수 있습니다.

6. 서비스(Service), 범주(Category), 심각도(Severity)를 선택합니다. 심각도 선택에 대한 자세한 내용은 심각도 선택을 참조하십시오.
   - 일반 지침
   - 시스템 손상
   - 프로덕션 시스템 손상
   - 프로덕션 시스템 중단
   - 비즈니스 크리티컬 시스템 중단
7. Next step: Additional information(다음 단계: 추가 정보)을 선택합니다
8. 추가 정보(Additional information) 페이지의 제목(Subject)에 해당 문제에 대한 제목을 입력합니다.
9. 설명(Description)에서 프롬프트를 따라 다음과 같이 사례를 설명합니다:
   - 워크플로 ID CodeCatalyst, 로그 또는 스크린샷과 같은 관련 문제 해결 정보
   - 수신한 오류 메시지
   - 수행한 문제 해결 단계

> **Note**
>
> 케이스 서신에는 자격 증명, 신용 카드, 서명된 URL 또는 개인 식별 정보와 같은 민감한 정보를 공유하지 마십시오.

10. (선택 사항) 파일 첨부(Attach files)를 선택하여 오류 로그 또는 스크린샷과 같은 관련 파일을 사례에 추가합니다. 최대 3개의 파일을 첨부할 수 있습니다. 각 파일의 크기는 최대 5MB까지입니다.
11. 스페이스 이름에는 스페이스 이름이 표시됩니다.
12. 빌더 이름에는 AWS 빌더 ID와 연결된 전체 이름이 자동으로 채워집니다.
13. (선택 사항) 프로젝트 이름에서 프로젝트를 선택합니다 (해당하는 경우).

> **Note**
>
> 권한이 있는 프로젝트만 표시됩니다. 다른 프로젝트에 액세스해야 하는 경우 지원 사례를 작성하기 전에 프로젝트 관리자에게 액세스 권한을 요청하세요.

14. 다음 단계: 문의하기를 선택하세요.
15. 선호 연락처 언어에서 기본값을 선택합니다. 현재는 영어만 사용할 수 있습니다.
16. 연락 방법으로 웹, 전화 또는 채팅 옵션을 선택합니다.
17. 사례 세부 정보를 검토한 다음 제출을 선택합니다. 사례 ID 번호와 요약이 표시됩니다.

    지원 사례는 스페이스 수준에서 생성되며 지원 사례에 정의된 스페이스 및 프로젝트 (선택한 경우)에 대한 액세스 권한이 있는 모든 구성원이 볼 수 있습니다. 현재로서는 개별 사용자의 지원 사례를 생략할 수 있는 방법이 없습니다.

에서 지원 사례 해결 CodeCatalyst

AWS SupportAmazon용 CodeCatalyst 페이지에서 미해결 지원 사례를 해결할 수 있습니다.

지원 사례를 해결하려는 스페이스에 스페이스 관리자 또는 스페이스 구성원 역할이 있어야 합니다. 스페이스 관리자 역할이 없거나 사례를 만들 때 프로젝트를 선택한 경우 사례를 확인하고 해결하려면 프로젝트 멤버십도 있어야 합니다.

미해결 지원 사례를 해결하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.

> **Tip**
>
> 둘 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택합니다.

3. 페이지 상단에서? 를 선택합니다. 아이콘을 선택한 다음 AWS SupportAmazon을 선택합니다 CodeCatalyst.
4. 관리하려는 지원 사례의 링크를 선택합니다. Resolve case(사례 해결)를 선택합니다.

에서 지원 사례 재개 CodeCatalyst

AWS SupportAmazon용 CodeCatalyst 페이지에서 해결된 지원 사례 다시 열기 기능을 사용할 수 있습니다.

> **Note**
>
> 문제가 해결된 날로부터 최대 14일 이내에 지원 사례를 다시 열 수 있습니다. 그러나 14일 이상 비활성 상태인 사례는 다시 열 수 없습니다. 케이스를 다시 열 수 없는 경우 새 케이스를 열고 이전 케이스 ID를 참조로 포함하십시오.
> 현재 문제와는 다른 정보가 있는 기존 사례를 다시 열면 지원 상담원이 새 사례 생성을 사용자에게 요청할 수 있습니다.

에서 지원 사례를 재개하려면 CodeCatalyst

1. https://codecatalyst.aws/ 에서 CodeCatalyst 콘솔을 엽니다.
2. CodeCatalyst 스페이스로 이동하세요.

   > **Tip**
   >
   > 둘 이상의 스페이스에 속해 있는 경우 상단 내비게이션 바에서 스페이스를 선택합니다.

3. 페이지 상단에서? 를 선택합니다. 아이콘을 클릭한 다음 AWS Support 대상을 선택합니다 CodeCatalyst.
4. 관리하려는 지원 사례의 링크를 선택합니다. Reopen(다시 열기)을 선택합니다. 확인 화면에서 [확인] 을 선택한 다음 [Submit] 을 선택합니다.
5. 설명에 동일한 문제에 대한 최신 정보를 입력합니다. 케이스 서신에는 자격 증명, 신용 카드, 서명된 URL 또는 개인 식별 정보와 같은 민감한 정보를 공유하지 마십시오.

에 대한 할당량 CodeCatalyst

다음 표에는 Amazon의 할당량 및 한도가 설명되어 있습니다. CodeCatalyst 다음 CodeCatalyst 주제에서 특정 측면에 대한 추가 정보를 찾을 수 있습니다.

- 의 소스 리포지토리 할당량 CodeCatalyst
- ID, 권한 및 액세스에 대한 할당량 CodeCatalyst
- 의 워크플로우 할당량 CodeCatalyst
- 의 개발 환경에 대한 할당량 CodeCatalyst
- 프로젝트 할당량
- 청사진 할당량 CodeCatalyst
- 공간 할당량
- 내 이슈에 대한 할당량 CodeCatalyst

계정의 최대 공간 수	5
한 달에 사용자가 만들 수 있는 최대 스페이스 수	5
AWS 계정 스페이스의 최소 개수	1
스페이스의 최대 계정 연결 수	5,000
AWS 계정 스페이스의 결제 계정으로 사용할 수 있는 최대 개수	1
하나를 결제 계정으로 지정할 AWS 계정 수 있는 프리 티어의 최대 공간 수	3
유료 등급에서 단일 공간을 결제 계정으로 지정할 AWS 계정 수 있는 최대 공간 수	15
스페이스의 최대 VPC 연결 수	100
스페이스의 최대 프로젝트 수	100

사용자가 속할 수 있는 최대 프로젝트 수	1,000
공간 설명	스페이스 설명은 선택 사항입니다. 지정하는 경우 길이는 0~200자 사이여야 합니다. 문자, 숫자, 공백, 마침표, 밑줄, 쉼표, 대시 및 다음 특수 문자의 조합을 포함할 수 있습니다. `? & $ % + = / \ ; : \n \t \r`
스페이스 이름	스페이스 이름은 전체에서 고유해야 CodeCatalyst 합니다. 삭제된 스페이스의 이름은 재사용할 수 없습니다. 스페이스 이름의 길이는 3~63자 사이여야 합니다. 또한 영숫자로 시작해야 합니다. 스페이스 이름에는 문자, 숫자, 마침표, 밑줄 및 대시를 조합하여 사용할 수 있습니다. 다음 문자를 포함할 수 없습니다. `! ? @ # $ % ^ & * ( ) + = { } [ ] | 八 > < ~ ` ' " ; :`

아마존의 문서 기록 CodeCatalyst

다음 표에는 전체 설명서의 설명서 기록 및 업데이트 내용이 설명되어 CodeCatalyst 있습니다.

변경 사항	설명	날짜
업데이트된 콘텐츠: 사용자 이름이 잘린 문제 해결	특정 경우에 잘릴 수 있는 사용자 이름 문제 해결에 대한 정보를 포함하도록 문제 해결 항목을 업데이트했습니다.	2024년 7월 31일
업데이트 내용: 환경 생성	CodeCatalyst 역할을 언급하도록 환경 생성 섹션을 업데이트했습니다.	2024년 7월 25일
업데이트 내용: 작업 간 아티팩트 및 파일 공유	액션 그룹에 대한 정보를 포함하도록 아티팩트의 파일 참조 하위 주제를 업데이트했습니다. 또한 예: 작업 그룹이 있을 때 아티팩트에 있는 파일 참조 하위 주제가 추가되었습니다.	2024년 7월 18일
업데이트 내용: 워크플로를 통한 빌드, 테스트, 배포	워크플로 정의 `.yml` 또는 `.yaml` 파일 확장자가 소문자여야 함을 나타내도록 설명서가 업데이트되었습니다.	2024년 7월 15일
콘텐츠 업데이트: 스페이스에 개발 환경을 만들 때 SSO 사용자를 위한 문제 해결 추가 CodeCatalyst	SSO사용자가 개발 환경을 만들 때 ID 페더레이션을 사용하여 스페이스에 로그인할 때 발생하는 문제에 대한 정보를 포함하도록 문제 해결 항목을 추가했습니다.	2024년 7월 12일
업데이트 내용: 워크플로 생성	워크플로우 정의 파일을 의 하위 디렉토리에 저장할 수 있음을 나타내도록 항목을 업데	2024년 7월 12일

	이트했습니다. `~/.codecatalyst/workflows/`	
업데이트 내용: 워크플로우 상태 CodeCatalyst	콘솔에서 비활성 워크플로를 숨기는 방법에 대한 지침이 CodeCatalyst 추가되었습니다.	2024년 7월 12일
새 콘텐츠: 수동 전용 트리거 구성	수동 전용 트리거 구성주제를 추가했습니다.	2024년 6월 26일
업데이트 내용: 환경 생성	CI/CD 환경을 만들 때 기본 IAM 역할을 지정할 수 있는 새로운 기능을 반영하도록 환경 생성 섹션 및 기타 여러 섹션을 업데이트했습니다. 이 역할은 환경과 관련된 모든 워크플로 작업에 할당됩니다. 더 이상 작업에 직접 IAM 역할을 할당할 필요가 없습니다.	2024년 6월 21일
새 콘텐츠: 자습서: 패키지 저장소에서 가져오기	CodeCatalyst 패키지 저장소에서 종속성을 가져오도록 워크플로를 구성하는 방법을 설명하는 자습서가 추가되었습니다.	2024년 6월 20일
콘텐츠 업데이트: 블루프린트가 포함된 GitLab 프로젝트 리포지토리 사용	또는 커스텀 블루프린트를 생성할 때 GitLab 블루프린트로 프로젝트 생성 프로젝트 리포지토리를 생성할 수 있는 기능에 대한 설명서가 업데이트되었습니다. 프로젝트에 블루프린트를 추가하여 리소스를 통합하기	2024년 6월 19일

콘텐츠 업데이트: 튜토리얼: 제너레이티브 AI 기능 사용	프로젝트를 생성하거나 기존 프로젝트에 청사진을 추가할 때 Amazon Q를 사용하여 청사진을 선택할 수 있는 기능을 반영하도록 자습서를 업데이트했습니다.	2024년 6월 18일
콘텐츠 업데이트: 소스 리포지토리 생성	빈 리포지토리 생성에 대한 정보를 포함하도록 설명서를 업데이트했습니다.	2024년 6월 18일
새 콘텐츠: Amazon CodeCatalyst Q로 프로젝트 생성	Amazon Q에서 프로젝트를 생성하고 청사진을 추가하는 방법에 대한 Amazon Q를 사용하여 프로젝트를 생성하거나 블루프린트로 기능을 추가하는 모범 사례 지침과 프로젝트에 블루프린트를 사용하는 모범 사례 함께 프로젝트 생성 제목에 새 섹션을 추가했습니다.	2024년 6월 18일
새 콘텐츠: 기존 Git 리포지토리를 소스 리포지토리로 복제	에 Git을 사용하여 미러 클론 또는 로컬 리포지토리를 빈 소스 리포지토리로 푸시하는 방법에 대한 새 항목이 추가되었습니다. CodeCatalyst	2024년 6월 18일
새 콘텐츠: Maven NuGet, 및 Python 패키지 유형에 대한 지원	에 Maven NuGet, Python 패키지 사용에 대한 설명서가 추가되었습니다. CodeCatalyst	2024년 6월 17일

스페이스 및 계정 연결 업데이트	이제 AWS 계정 지정된 결제 계정을 여러 CodeCatalyst 스페이스에 연결할 수 있습니다. ID 페더레이션을 위해 설정된 스페이스의 경우 하나의 Identity Center 인스턴스를 여러 스페이스에 연결할 수 있습니다. 연결된 AWS 리소스에 대한 액세스 허용 AWS 계정, 스페이스와 관련된 계정 관련 문제 해결 및 에 대한 할당량 CodeCatalyst 단원을 참조하세요.	2024년 6월 13일
업데이트된 콘텐츠: 역할 관련 작업	Amazon Q를 사용하여 문제에 대한 작업을 추천하고 생성할 수 있는 권한을 포함하도록 역할 설명서를 업데이트했습니다.	2024년 6월 13일
콘텐츠 업데이트: 역할 다루기	이슈 간 링크를 생성, 업데이트, 제거할 수 있는 권한을 포함하도록 역할 설명서를 업데이트했습니다.	2024년 6월 13일
콘텐츠 업데이트: 튜토리얼: 제너레이티브 AI 기능 사용	Amazon Q를 사용하여 문제에 대한 작업을 추천하는 방법에 대한 정보를 포함하도록 자습서를 업데이트했습니다.	2024년 6월 13일
콘텐츠 업데이트: 이슈 관련 작업 관리	Amazon Q를 사용하여 문제에 대한 작업을 추천하는 방법에 대한 정보가 추가되었습니다 CodeCatalyst.	2024년 6월 13일
새 콘텐츠: 이슈를 다른 이슈에 연결	에 이슈를 다른 이슈와 연결하는 방법에 대한 새 주제를 추가했습니다 CodeCatalyst.	2024년 6월 13일

업데이트 내용: AWS CDK '배포' 작업 YAML	`Region`속성에 대한 설명을 수정했습니다.	2024년 6월 12일
콘텐츠 업데이트: 블루프린트가 있는 타사 리포지토리 사용	GitHub 리포지토리를 생성할 때 블루프린트로 프로젝트 생성 또는 사용자 지정 블루프린트를 생성할 수 있는 기능에 대한 설명서가 업데이트되었습니다. 프로젝트에 블루프린트를 추가하여 리소스를 통합하기	2024년 6월 6일
새 콘텐츠: 개인 연결 작업을 위한 단계 추가	개인용 연결 생성 및 삭제 단계가 추가되었습니다. 개인 연결을 통해 CodeCatalyst Amazon의 프로젝트 및 청사진에 대한 GitHub 리소스를 관리할 수 있습니다.	2024년 6월 6일
새 콘텐츠: Bitbucket 리포지토리 확장	에서 Bitbucket 리포지토리 확장을 사용하기 위한 새 콘텐츠를 추가했습니다. CodeCatalyst	2024년 6월 5일
새 콘텐츠: 작업 유형	SonarCloud Scan 액션을 언급하도록 타사 작업 주제를 업데이트했습니다.	2024년 5월 29일
업데이트 내용: AWS CDK '배포' 작업 YAML	`CdkRootRootPath` 예제를 수정했습니다.	2024년 5월 28일
업데이트 내용	주제 제목을 업데이트하고 콘텐츠를 재구성하여 가독성과 발견성을 개선했습니다. 이러한 변경 사항에 대한 피드백을 제공하려면 이 피드백 제공 링크를 사용하세요.	2024년 5월 17일

새 콘텐츠: 파일 변경 기록 보기	소스 저장소의 파일 변경 기록을 볼 수 있는 새로운 기능을 반영하도록 설명서가 업데이트되었습니다.	2024년 5월 1일
콘텐츠 업데이트: 튜토리얼: 제너레이티브 AI 기능 사용	Amazon Q Developer와의 통합을 반영하도록 자습서를 업데이트했습니다.	2024년 4월 29일
콘텐츠 업데이트: 튜토리얼: 제너레이티브 AI 기능 사용	Amazon Q에서 복잡한 문제를 분석하고, 작업을 제안 및 생성하고, 문제가 발생한 작업을 수행할 수 있도록 허용하는 기능을 반영하도록 자습서를 업데이트했습니다.	2024년 4월 22일
새 콘텐츠: 워크플로 실행 게이팅	워크플로 실행 게이팅워크플로 실행 시 승인 필요, 및 워크플로우 승인과 관련된 기타 여러 항목이 추가되었습니다.	2024년 4월 22일
새 콘텐츠: 튜토리얼: 컴포저블 블루프린트로 풀스택 애플리케이션 만들기 PDK	Amazon AWS CodeCatalyst 프로젝트에 프로젝트 개발 키트 (AWSPDK) 블루프린트를 사용하기 위한 새 자습서가 추가되었습니다.	2024년 4월 9일
새 콘텐츠: 작업을 사용하여 문제를 더 작은 목표로 세분화합니다.	이슈가 되는 작업의 시작을 지원하는 콘텐츠를 추가했습니다. 이슈에 작업을 추가하여 해당 이슈의 작업을 더 세분화하고, 구성하고, 추적할 수 있습니다.	2024년 4월 4일

업데이트 내용: 작업 간 아티팩트 및 파일 공유	다음과 같은 작업 간 아티팩트 및 파일 공유 두 개의 새 하위 주제를 포함하도록 주제를 업데이트했습니다. 아티팩트를 출력 및 입력으로 지정하지 않고도 아티팩트를 공유할 수 있나요? 워크플로 간에 아티팩트를 공유할 수 있습니까?	2024년 4월 2일
업데이트 내용: 의 GitHub 액션 제한 CodeCatalyst	GitHubActions가 이전 런타임 환경 Docker 이미지에서 실행됨을 나타내도록 의 GitHub 액션 제한 CodeCatalyst 주제를 업데이트했습니다.	2024년 4월 2일
새 콘텐츠: AWS CDK '배포' 작업 YAML	에 새 `CloudAssemblyRootPath` 속성을 추가했습니다AWS CDK '배포' 작업 YAML.	2024년 4월 1일
업데이트 내용: 런타임 환경 이미지 지정	새로운 2024년 3월 런타임 환경 이미지에 대한 정보를 포함하도록 런타임 환경 이미지 지정 항목을 업데이트했습니다.	2024년 3월 26일
콘텐츠 업데이트: 역할 관련 작업	역할 권한 정보를 단일 테이블로 통합했습니다. 표는 새 각 역할에 사용할 수 있는 권한 보기 주제에 있습니다.	2024년 3월 18일

새 콘텐츠: 사용자의 모든 공간 및 프로젝트 보기	로그인한 사용자의 각 CodeCatalyst 스페이스 또는 프로젝트를 보여주는 사용자 홈 페이지의 목록 보기에 대한 정보가 추가되었습니다. CodeCatalyst 사용자의 모든 공간 및 프로젝트 보기을 참조하세요.	2024년 3월 18일
새 콘텐츠: 예: 풀과 브랜치가 있는 트리거	풀 리퀘스트 트리거 예시가 추가되었습니다. 트리거를 사용하여 자동으로 워크플로 실행 시작주제 전체를 약간 수정했습니다.	2024년 3월 11일
콘텐츠 업데이트: 역할 다루기	환경을 만들고, 삭제하고, 볼 수 있는 권한을 포함하도록 역할 설명서를 업데이트했습니다.	2024년 3월 4일
콘텐츠 업데이트: 튜토리얼: 제너레이티브 AI 기능 사용	Amazon Q에 이슈를 생성하고 할당할 때의 변경 사항을 반영하도록 자습서를 업데이트했습니다.	2024년 3월 4일
새 콘텐츠: 이슈 구성 요소	커스텀 블루프린트 개발자로서 이슈 컴포넌트를 다루는 방법에 대한 새 콘텐츠를 추가했습니다.	2024년 2월 27일
업데이트 내용: 작업 유형	CodeCatalyst 랩 액션 목록을 포함하도록 CodeCatalyst 랩 작업 주제를 업데이트했습니다.	2024년 2월 21일
콘텐츠 업데이트: 풀 리퀘스트 다루기	승인 규칙 및 풀 리퀘스트 병합을 위한 우선 요구 사항이 포함된 새로운 기능을 반영하도록 설명서를 업데이트했습니다.	2024년 2월 15일

콘텐츠 업데이트: 풀 리퀘스트 병합	필수 검토자의 승인을 아직 받지 않았거나 승인 규칙을 충족하지 않은 풀 요청을 병합하기 위해 병합 요구 사항을 재정의하는 방법에 대한 정보가 포함되도록 풀 요청에 대한 문서를 추가했습니다.	2024년 2월 15일
새 콘텐츠: 승인 규칙 관리	승인 규칙 생성 및 관리에 대한 정보가 포함되도록 풀 리퀘스트 문서를 추가했습니다.	2024년 2월 15일
콘텐츠 업데이트: 역할 관련 작업	승인 규칙 및 풀 리퀘스트 작업을 위한 권한을 포함하도록 역할 설명서를 업데이트했습니다.	2024년 2월 14일
업데이트 내용: “워크플로 정의에 다음과 같은 문제가 있는 문제를 해결하려면 어떻게 해야 합니까?*n* 오류” 오류를 해결합니까?	추가 문제 해결 팁을 포함하도록 “워크플로 정의에 다음과 같은 문제가 있는 문제를 해결하려면 어떻게 해야 합니까?*n* 오류” 오류를 해결합니까? 섹션을 업데이트했습니다.	2024년 2월 9일
새 콘텐츠: 워크플로우 상태 보기	워크플로 상태를 설명하는 섹션이 추가되었습니다.	2024년 2월 9일
새 콘텐츠: 워크플로우 상태 보기	워크플로 상태를 설명하는 섹션이 추가되었습니다.	2024년 2월 9일
콘텐츠 업데이트: 의 워크플로우 할당량 CodeCatalyst	워크플로당 최대 작업 수 및 AWS 계정 스페이스당 할당량과 관련된 최대 환경 수로 의 워크플로우 할당량 CodeCatalyst 항목을 업데이트했습니다.	2024년 2월 7일

업데이트 내용: 환경 생성	환경당 최대 1개의 계정 연결을 사용할 수 있도록 환경 생성 섹션을 업데이트했습니다.	2024년 1월 31일
새 콘텐츠: 커스텀 블루프린트 리포지토리 GitHub	공개적으로 사용할 수 있는 GitHub 리포지토리에 새 콘텐츠를 추가했습니다.	2024년 1월 10일
콘텐츠 업데이트: npm 구성 방법 CodeCatalyst	npm과 함께 사용하기 위한 일반 구성 지침을 업데이트하고 CodeCatalyst 옵션에 대한 명확성을 추가했습니다. `always-auth=true`	2024년 1월 5일
콘텐츠 업데이트: 풀 리퀘스트 다루기	제너레이티브 AI 기능이 포함된 새로운 기능을 반영하도록 설명서를 업데이트했습니다 CodeCatalyst.	2023년 11월 28일
콘텐츠 업데이트: 이슈 생성	제너레이티브 AI 기능이 포함된 새로운 기능을 반영하도록 설명서를 업데이트했습니다 CodeCatalyst.	2023년 11월 28일
새 콘텐츠: 튜토리얼: 제너레이티브 AI 기능 사용	CodeCatalystAmazon에서 제너레이티브 AI 기능을 사용하기 위한 자습서가 추가되었습니다.	2023년 11월 28일
새 콘텐츠: 사용자 지정 청사진 및 라이프사이클 관리	CodeCatalystAmazon에서 사용자 지정 청사진 및 수명 주기 관리 기능을 사용하기 위한 새 콘텐츠를 추가했습니다.	2023년 11월 27일
업데이트 내용: 튜토리얼: 모던 3계층 웹 애플리케이션 블루프린트로 프로젝트 생성	수정 및 문제 해결 정보로 자습서를 업데이트했습니다.	2023년 11월 22일

업데이트 내용: 트리거를 사용하여 자동으로 워크플로 실행 시작	풀 리퀘스트 트리거와 관련된 몇 가지 예제와 설명을 수정했습니다. 트리거 및 브랜치에 대한 사용 지침섹션이 추가되었습니다.	2023년 11월 22일
새 콘텐츠: SSO로 로그인	Single Sign-On (SSO) 으로 로그인하는 방법에 대한 정보와 ID 페더레이션을 지원하는 CodeCatalyst 공간 설정 및 관리에 대한 정보 링크가 추가되었습니다. 설정 및 로그인 CodeCatalyst 및 SSO로 로그인 단원을 참조하세요.	2023년 11월 17일
콘텐츠 업데이트: 역할 관련 작업	팀, VPC 연결, Single Sign-On 및 컴퓨터 리소스와 작업할 수 있는 권한을 포함하도록 역할 설명서를 업데이트했습니다.	2023년 11월 16일
콘텐츠 업데이트: 풀 리퀘스트 다루기	풀 리퀘스트의 변경 사항이 표시되는 방식의 변경 사항을 반영하도록 설명서를 업데이트했습니다.	2023년 11월 16일
콘텐츠 업데이트: 할당량 CodeCatalyst	공간 할당량의 최대 VPC 연결 수로 에 대한 할당량 CodeCatalyst 주제를 업데이트했습니다.	2023년 11월 16일
새 콘텐츠: 스페이스 및 CodeCatalyst 프로젝트 팀 관리	스페이스가 있는 팀을 사용하는 방법에 대한 정보가 추가되었습니다. 팀을 통한 공간 액세스 허용 및 팀을 통한 프로젝트 액세스 허용 단원을 참조하세요.	2023년 11월 16일

새 콘텐츠: 공간 내 청사진 및 워크플로를 위한 시스템 리소스 관리	스페이스와 함께 머신 리소스를 사용하는 방법에 대한 정보가 추가되었습니다. 시스템 리소스에 대한 공간 액세스 허용을 참조하세요.	2023년 11월 16일
새 콘텐츠: 프로젝트의 블루프린트와 워크플로를 위한 머신 리소스 관리 CodeCatalyst	CodeCatalyst 프로젝트에 머신 리소스를 사용하는 방법에 대한 정보가 추가되었습니다. 시스템 리소스에 대한 프로젝트 액세스 허용을 참조하세요.	2023년 11월 16일
새 콘텐츠: 환경과 VPC 연결 연결	워크플로에서 사용할 수 있는 환경과 VPC 연결을 연결하기 위한 설명서가 추가되었습니다.	2023년 11월 16일
새 콘텐츠: 개발 환경에 VPC 연결 연결	연결을 통한 개발 환경 사용에 대한 설명서가 추가되었습니다. VPC	2023년 11월 16일
새 콘텐츠	Amazon CodeCatalyst 관리자 안내서의 최초 간행물.	2023년 11월 16일
새 콘텐츠: AWS CDK '배포' 작업 YAML	AWS CDK '배포' 작업 YAML및에 새 `CdkCliVersion` 속성을 추가했습니다AWS CDK '부트스트랩' 액션 YAML.	2023년 11월 14일
콘텐츠 업데이트: 역할 다루기	분기 규칙 작업을 위한 권한을 포함하도록 역할 설명서를 업데이트했습니다.	2023년 11월 13일
콘텐츠 업데이트: 소스 리포지토리, 워크플로, 개발 환경 관련 문제 해결	분기 규칙 사용에 대한 정보를 포함하도록 문제 해결 항목을 업데이트했습니다.	2023년 11월 13일

업데이트 내용: 빌드 및 테스트 액션 YAML	`Environment` 속성에 대한 설명서가 업데이트되었습니다. 이제 이 필드는 빌드 및 테스트 작업을 위한 선택적 필드입니다.	2023년 11월 13일
새 콘텐츠: 브랜치 규칙 관리	소스 리포지토리의 브랜치에 대한 모든 규칙을 확인하고 브랜치 규칙을 생성 및 관리하는 방법에 대한 정보가 포함된 브랜치에 대한 설명서가 추가되었습니다.	2023년 11월 13일
콘텐츠 업데이트: 풀 리퀘스트 다루기	풀 리퀘스트에 대한 정보가 표시되는 방식의 변경 사항을 반영하도록 설명서를 업데이트했습니다.	2023년 11월 10일
업데이트 내용: 워크플로우 실행 간 파일 캐싱	파일 캐싱 제한을 포함하도록 설명서를 업데이트했습니다.	2023년 11월 10일
업데이트 내용: 자습서: Amazon에 애플리케이션 배포 EKS	EKS앱 배포 청사진을 언급하도록 설명서를 업데이트했습니다.	2023년 11월 9일
새 콘텐츠: 패키지 내부 CodeCatalyst	에서 패키지 사용에 대한 설명서가 추가되었습니다 CodeCatalyst.	2023년 11월 1일
신규 및 업데이트된 콘텐츠: 역할 다루기	고급 사용자, 제한된 액세스 권한, 검토자, 읽기 전용의 CodeCatalyst 네 가지 새 역할에 대한 설명서가 업데이트되었습니다.	2023년 11월 1일

업데이트 내용: GitHub 출력 매개변수 내보내기	`set-output` 명령 대신 `GITHUB_OUTPUT` 환경 파일을 사용하도록 예제를 업데이트했습니다. 출력 매개 변수를 설정하는 권장 방법은 환경 파일을 사용하는 것입니다. GitHub	2023년 10월 24일
새 콘텐츠: 트리거를 사용하여 자동으로 워크플로 실행 시작	스케줄 트리거에 대한 설명서가 추가되었습니다.	2023년 10월 16일
업데이트 내용: '쿠버네티스 클러스터에 배포' 작업 YAML	'쿠버네티스 클러스터에 배포' 작업 YAML및 자습서: Amazon에 애플리케이션 배포 EKS 항목에 `CodeCatalystWorkflowDevelopmentRole-`*`spaceName`* 역할 사용에 대한 정보가 추가되었습니다.	2023년 9월 22일
업데이트된 콘텐츠: 새 역할 이름 및 정책 CodeCatalystWorkflowDevelopmentRole-***spaceName*** 역할	개발자 역할 이름 변경에 대한 단계 및 역할 설명을 다음과 같이 업데이트했습니다. CodeCatalystWorkflowDevelopmentRole-***spaceName*** . 이제 개발자 역할은 `AdministratorAccess` AWS 관리형 정책을 사용합니다. CodeCatalystWorkflowDevelopmentRole-***spaceName*** 서비스 역할 이해 및 첫 번째 스페이스 및 개발 역할 만들기 (초대 없이 시작) 단원을 참조하세요.	2023년 9월 20일

업데이트 내용: 워크플로우에서 변수 사용	사용자 정의 변수와 사전 정의된 변수라는 두 가지 새로운 개념을 도입했습니다. 이러한 개념을 통해 워크플로우에서 변수 사용 섹션을 더 쉽게 읽고 이해할 수 있을 것입니다.	2023년 9월 19일
업데이트된 콘텐츠: 커밋 다루기	표시된 정보의 변경 사항을 반영하고 여러 부모가 있는 커밋을 보는 방법에 대한 세부 정보를 제공하도록 설명서를 업데이트했습니다.	2023년 9월 7일
새 콘텐츠: 트리거를 사용하여 자동으로 워크플로 실행 시작	트리거를 사용하여 자동으로 워크플로 실행 시작주제에 다음 예제를 추가했습니다. 예: 간단한 'Push to main' 트리거	2023년 9월 6일
콘텐츠 업데이트: 풀 리퀘스트 다루기	풀 리퀘스트 생성 시 소스 브랜치 및 대상 브랜치의 표시 순서 변경을 반영하도록 설명서를 업데이트했습니다.	2023년 8월 30일
새 콘텐츠: 기본 브랜치 보기 및 변경	소스 리포지토리의 기본 브랜치 보기 및 변경에 대한 정보가 포함되도록 브랜치에 대한 문서를 추가했습니다.	2023년 8월 30일
업데이트 내용: '쿠버네티스 클러스터에 배포' 작업 YAML	의 Manifests 속성 설명에 헬름과 Kustomize에 대한 메모를 추가했습니다. '쿠버네티스 클러스터에 배포' 작업 YAML	2023년 8월 15일
새 콘텐츠: 이슈 첨부 파일 관리	이슈에 대한 첨부 파일 작업 및 관리를 위한 설명서가 추가되었습니다.	2023년 8월 15일

업데이트 내용: 트리거를 사용하여 자동으로 워크플로 실행 시작	워크플로 트리거와 관련된 문서를 개선 및 확장했습니다.	2023년 8월 11일
새 콘텐츠: 역할 권한 문제 해결	Amazon에 액세스해야 하는 워크플로를 실행하기 위한 역할 권한 업데이트에 대한 정보가 추가되었습니다 CodeGuru. Amazon의 권한 오류로 OnPullRequest인해 최신 3계층 웹 애플리케이션 블루프린트 워크플로가 실패함 CodeGuru을 참조하세요.	2023년 8월 11일
새 콘텐츠: "워크플로가 비활성 상태입니다." 라는 메시지를 수정하려면 어떻게 해야 합니까?	다음 문제 해결 항목이 추가되었습니다. "워크플로가 비활성 상태입니다." 라는 메시지를 수정하려면 어떻게 해야 합니까?	2023년 8월 11일
새 콘텐츠: 워크플로를 EKS 사용하여 Amazon에 배포하기	Kubernetes 클러스터에 배포 작업에 대한 설명서가 추가되었습니다. 자세한 내용은 워크플로를 EKS 사용하여 Amazon에 배포하기 및 자습서: Amazon에 애플리케이션 배포 EKS 단원을 참조하세요.	2023년 7월 27일
스페이스에 대한 관리 이벤트가 기록되는 방식에 대한 업데이트 CodeCatalyst	CodeCatalyst스페이스에서 특정 작업에 대한 관리 이벤트가 기록되는 방식에 대한 정보가 추가되었습니다 AWS CloudTrail. list-event-logs 명령으로 스페이스의 모든 이벤트를 볼 수 있는 방법에 대한 정보가 추가되었습니다. 로깅을 사용하여 이벤트 및 API 통화 모니터링을 참조하세요.	2023년 7월 20일

업데이트 내용: 트리거를 사용하여 자동으로 워크플로 실행 시작	풀 리퀘스트 트리거가 이제 GitHub 소스 리포지토리에서 지원된다는 것을 나타내도록 설명서가 업데이트되었습니다. 이전에는 풀 리퀘스트 트리거가 소스 리포지토리에서만 지원되었습니다. CodeCatalyst	2023년 7월 14일
업데이트 내용: 의 워크플로우 할당량 CodeCatalyst	작업이 할당량을 실행할 수 있는 최대 시간으로 의 워크플로우 할당량 CodeCatalyst 주제를 업데이트했습니다.	2023년 6월 27일
업데이트 내용: 워크플로우 YAML 정의	`Compute`코드 블록의 서식 오류를 수정했습니다.	2023년 6월 27일
콘텐츠 업데이트: 데이터 보호	데이터 복제에 대한 추가 정보를 포함하도록 설명서를 업데이트했습니다.	2023년 6월 26일
새 콘텐츠: 사용할 액션 버전 지정	사용할 액션 버전 지정주제를 추가했습니다.	2023년 6월 21일
업데이트 내용: 및 에 AWS 계정 배포 VPCs	배포 정보를 표시할 수 있도록 지원하는 작업은 무엇입니까? CodeCatalyst 섹션을 명확히 했습니다.	2023년 6월 14일
콘텐츠 업데이트: 문제 문서 재구성	전체 문서 세트 및 사용자 흐름에 더 잘 맞도록 대부분의 문제 문서를 재구성했습니다.	2023년 5월 31일
콘텐츠 업데이트: 이슈 뷰 전환기	업데이트된 이슈 뷰 스위처에 맞춰 다양한 사용자 플로우를 업데이트했습니다.	2023년 5월 31일

콘텐츠 업데이트: 알림 관리	개인 Slack 알림 구성에 대한 정보를 포함하도록 알림 설명서를 업데이트했습니다.	2023년 5월 30일
콘텐츠 업데이트: 알림 관리	개인 Slack 알림 구성에 대한 정보를 포함하도록 알림 설명서를 업데이트했습니다.	2023년 5월 30일
새 콘텐츠: CodeCatalyst 신뢰 모델	연결된 환경에서 서비스 역할을 CodeCatalyst 맡을 수 있는 신뢰 모델에 대한 정보가 포함된 새 항목이 추가되었습니다 AWS 계정. 에 대해 정의된 서비스 주체에 대한 새 섹션이 추가되었습니다. CodeCatalyst CodeCatalyst 신뢰 모델 이해을 참조하세요.	2023년 5월 20일
업데이트 내용: 소스 리포지토리를 워크플로에 연결	의 지침을 단순화했습니다. 소스 리포지토리 파일 참조	2023년 5월 10일
업데이트 내용: 의 워크플로우 할당량 CodeCatalyst	출력 변수 값 할당량의 최대 길이로 의 워크플로우 할당량 CodeCatalyst 주제를 업데이트했습니다.	2023년 5월 10일
새 콘텐츠: 작업 간 아티팩트 및 파일 공유	두 가지 예시가 추가되었습니다: 예: 단일 아티팩트의 파일 참조 및예: a가 있을 때 아티팩트에 있는 파일 참조 WorkflowSource .	2023년 5월 10일
콘텐츠 업데이트: 풀 리퀘스트 다루기	풀 리퀘스트 이벤트에 대한 이메일 환경설정을 구성하는 방법에 대한 정보를 포함하도록 풀 리퀘스트 설명서를 업데이트했습니다.	2023년 4월 21일

콘텐츠 업데이트: 알림 관리	풀 리퀘스트 이벤트의 이메일 기본 설정을 구성하는 방법에 대한 정보를 포함하도록 알림 설명서를 업데이트했습니다.	2023년 4월 21일
관리형 정책으로 업데이트	AWS 관리형 정책: AmazonCodeCatalystFullAccessAWS 관리형 정책: AmazonCodeCatalystReadOnlyAccess, 및 AWS 관리형 정책: AmazonCodeCatalystSupportAccess 관리형 정책을 추가했습니다. AWS 관리형 정책으로 CodeCatalyst 업데이트을 참조하세요.	2023년 4월 20일
새 콘텐츠: 배포 대상 제거	배포 대상 제거주제를 추가했습니다.	2023년 4월 20일
새 콘텐츠: 작업 유형	CodeCatalyst 조치주제를 추가했습니다.	2023년 4월 20일
스페이스에서 스페이스 관리자 역할을 가진 사용자를 관리하기 위한 업데이트	스페이스에서 스페이스 관리자 역할을 가진 사용자의 역할을 제거하거나 변경하는 방법에 대한 정보가 추가되었습니다. 스페이스 관리자 역할을 가진 사용자의 역할 제거 또는 변경을 참조하세요.	2023년 4월 19일
개발 환경 관리를 위한 업데이트	Space 관리자로 개발 환경을 관리하는 방법에 대한 정보가 추가되었습니다. 스페이스의 개발 환경 관리을 참조하세요.	2023년 4월 19일
업데이트된 콘텐츠: 문제 찾기 및 보기	문제 찾기 및 보기주제 및 하위 주제를 재구성했습니다.	2023년 4월 19일

콘텐츠 업데이트: 연결된 GitLab 프로젝트 저장소를 CodeCatalyst 사용하여 프로젝트 만들기	연결된 타사 리포지토리를 사용하여 프로젝트 만들기섹션에 GitLab 통합을 프로젝트 생성 포함하도록 업데이트되었습니다.	2023년 4월 19일
업데이트 내용: 컴퓨팅 및 런타임 이미지 구성	아마존 리눅스 2에 Arm64 아키텍처에 대한 지원이 추가되었습니다.	2023년 4월 19일
새 콘텐츠: 그룹 내 이동 문제	게시판 및 모든 이슈 보기에 그룹 내 이슈 이동에 대한 문서를 추가했습니다.	2023년 4월 19일
업데이트 내용: 의 워크플로우 할당량 CodeCatalyst	할당량이 누락된 의 워크플로우 할당량 CodeCatalyst 주제를 업데이트하고 단일 작업의 출력 변수 할당량의 최대 총 크기를 2KB에서 120KB (기존) 로 업데이트했습니다.	2023년 4월 18일
새 콘텐츠: 액션의 소스 코드 보기	액션의 소스 코드 보기주제를 추가했습니다.	2023년 4월 18일
새 콘텐츠: 워크플로 실행 중지	워크플로 실행 중지주제를 추가했습니다.	2023년 4월 10일
새 콘텐츠: AWS Amazon과 Amazon 간의 계정 연결을 위해 리소스에 태그를 지정하는 섹션 추가 CodeCatalyst	계정 연결 리소스에 태그를 지정하고 연결 리소스에 대한 IAM 정책을 관리하기 위한 정보가 추가되었습니다. 태그를 사용하여 계정 연결 리소스에 대한 액세스 제어 및 CodeCatalyst 권한 참조 단원을 참조하세요.	2023년 4월 6일
새 콘텐츠: 작업 유형	작업 유형주제를 추가했습니다.	2023년 4월 6일

업데이트 내용: AWS CloudFormation '스택 배포' 작업 YAML	`parameter-overrides` 속성 설명을 업데이트했습니다. 이제 JSON 파일을 지원합니다.	2023년 4월 5일
새 콘텐츠: 연결된 GitHub 저장소를 CodeCatalyst 사용하여 프로젝트 만들기	GitHub 저장소에 연결되는 프로젝트를 생성하기 위한 지침이 연결된 타사 리포지토리를 사용하여 프로젝트 만들기 포함된 프로젝트 생성 제목의 새 섹션을 추가했습니다.	2023년 4월 5일
콘텐츠 업데이트: 알림 다루기	프로젝트 이벤트에 대한 이메일 구성에 대한 정보를 포함하도록 알림 설명서를 업데이트했습니다.	2023년 3월 31일
새 콘텐츠	Amazon CodeCatalyst 액션 개발 키트 가이드의 첫 출판.	2023년 3월 31일
콘텐츠 업데이트: Amazon의 스페이스 섹션을 재구성했습니다. CodeCatalyst	랜딩 페이지를 제거하고 주제를 통합하여 스페이스 섹션을 업데이트했습니다.	2023년 3월 29일
업데이트 내용: 자습서: Amazon에 애플리케이션 배포 ECS	AWS IAM Identity Center 대신 AWS Identity and Access Management사용자를 생성하는 방법을 1단계: AWS 사용자 설정 및 AWS CloudShell 설명하도록 변경되었습니다. IAM사용자 생성은 더 이상 권장되지 않습니다.	2023년 3월 23일
업데이트된 콘텐츠: 역할 관련 작업	이슈를 풀 리퀘스트에 연결할 수 있는 권한을 포함하도록 스페이스 관리자, 프로젝트 관리자, 기여자 역할에 대한 설명서가 업데이트되었습니다.	2023년 3월 13일

콘텐츠 업데이트: 풀 리퀘스트 다루기	풀 리퀘스트에 문제를 연결하는 방법에 대한 정보를 포함하도록 풀 리퀘스트 문서를 업데이트했습니다.	2023년 3월 13일
콘텐츠 업데이트: 이슈 관련 작업	문제를 풀 리퀘스트에 연결하는 방법에 대한 정보를 포함하도록 이슈 설명서를 업데이트했습니다.	2023년 3월 13일
새 콘텐츠: 프로젝트 내 모든 실행의 상태 및 세부 정보 보기	새로운 통합 워크플로 실행 페이지를 설명하는 섹션이 추가되었습니다.	2023년 3월 8일
새 콘텐츠: “워크플로 정의에 다음과 같은 문제가 있는 문제를 해결하려면 어떻게 해야 합니까?*n* 오류” 오류를 해결합니까?	“워크플로 정의에 오류가 있습니다.” 오류를 해결하는 방법에 대한 섹션이 추가되었습니다.	2023년 3월 7일
업데이트 내용: 워크플로 생성	새 UI를 반영하도록 지침을 업데이트했습니다.	2023년 3월 3일
새 콘텐츠: 다음과 통합하기 universal-test-runner	다음과 통합하기 universal-test-runner주제를 추가했습니다.	2023년 3월 3일
업데이트 내용: 워크플로를 통한 빌드, 테스트, 배포	워크플로 요약 페이지에 새 소스 리포지토리, 브랜치 및 워크플로 이름 필터를 반영하도록 다양한 섹션이 업데이트되었습니다.	2023년 3월 2일
새 콘텐츠: 커밋별 배포 상태 추적	커밋별 코드 품질 및 배포 상태 보기에 대한 섹션이 추가되었습니다.	2023년 2월 27일

새 콘텐츠: 'BranchName' 및 'CommitId' 변수	`BranchName` 사전 정의된 새 변수를 추가했습니다.	2023년 2월 16일
업데이트된 콘텐츠: Amazon에서 스페이스 구성원 관리 CodeCatalyst	에서 사용자에게 할당된 역할을 기반으로 하는 두 개의 새 테이블에서 구성원 역할 변경, 구성원 초대, 구성원 제거에 대한 정보가 업데이트되었습니다 CodeCatalyst.	2023년 2월 15일
콘텐츠 업데이트: Amazon CodeCatalyst 콘솔에 PAT 관리 단계 추가	PATs콘솔에서 조회, 생성 및 삭제하기 위한 단계가 추가되었습니다.	2023년 2월 15일
업데이트 내용: 런타임 환경 이미지 지정	기본 이미지 도구 버전 테이블에 더 많은 도구를 추가했습니다.	2023년 1월 10일
업데이트 내용: 작업 간 아티팩트 및 파일 공유	아티팩트 경로를 수정했습니다.	2023년 1월 3일
업데이트 내용: 'GitHub 액션' 액션 YAML	섹션의 코드 스니펫을 수정했습니다. `Steps`	2023년 1월 3일
업데이트 내용: 소스 리포지토리를 워크플로에 연결	소스 경로를 수정했습니다.	2023년 1월 3일
콘텐츠 업데이트: 풀 리퀘스트 업데이트	풀 리퀘스트의 필수 또는 선택적 검토자 업데이트에 대한 정보를 포함하도록 설명서를 업데이트했습니다.	2022년 12월 23일
새 콘텐츠: 워크플로우 실행 간 파일 캐싱	워크플로우에 파일 캐싱을 위한 페이지가 추가되었습니다.	2022년 12월 20일
콘텐츠 업데이트: 풀 리퀘스트 작업	알림에 대한 정보를 포함하도록 풀 리퀘스트 설명서를 업데이트했습니다.	2022년 12월 16일

새 콘텐츠: AWS CDK '배포' 작업 YAML	새 `CdkRootPath` 속성을 추가했습니다.	2022년 12월 16일
새 콘텐츠: 작업 간 컴퓨팅 공유	작업 간 컴퓨팅 공유주제를 추가했습니다.	2022년 12월 14일
업데이트 내용: 작업 간 아티팩트 및 파일 공유	입력 아티팩트를 지정하는 방법을 보여주는 예제가 수정되었습니다.	2022년 12월 13일
새 콘텐츠: 'GitHub 액션' 액션 YAML	GitHub Actions 작업을 위한 전용 참조 페이지를 추가했습니다.	2022년 12월 13일
콘텐츠 업데이트: 프로젝트 할당량 CodeCatalyst	한 공간에 최대 100개의 프로젝트가 포함되도록 설명서가 업데이트되었습니다.	2022년 12월 2일
새 콘텐츠	Amazon CodeCatalyst 사용 설명서의 최초 출판.	2022년 12월 1일
새 콘텐츠: ???	타사 확장 기능을 사용할 때 사용자가 겪을 수 있는 문제에 대한 문제 해결 항목이 추가되었습니다.	2022년 6월 18일

AWS 용어집

최신 AWS 용어는 참조의 AWS 용어집을 참조하십시오.AWS 용어집

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.