인라인 및 어시스턴트 코드 생성을 위한 Amazon Q Developer의 모범 사례

AWS 권장 가이드

AWS 권장 가이드: 인라인 및 어시스턴트 코드 생성을 위한 Amazon Q Developer의 모범 사례

Table of Contents

인라인 및 어시스턴트 코드 생성을 위한 Amazon Q Developer의 모범 사례

Amazon Web Services(기여자)

2024년 8월(문서 이력)

전통적으로 개발자는 코드를 작성하고 유지하기 위해 다양한 소스의 자체 전문 지식, 설명서 및 코드 조각에 의존했습니다. 이러한 방법은 업계에 좋은 영향을 미쳤지만 시간이 많이 걸리고 인적 오류가 발생하기 쉬워 비효율성과 잠재적 버그가 발생할 수 있습니다.

여기에서 Amazon Q Developer는 개발자의 여정을 개선하기 위해 를 진행합니다. Amazon Q Developer는 지능형 코드 AWS 생성 및 권장 사항을 제공하여 코드 개발 작업을 가속화하도록 설계된 강력한 생성형 AI 기반 어시스턴트입니다.

그러나 모든 신기술과 마찬가지로 문제가 있을 수 있습니다. 개발자가 직면할 수 있는 일반적인 장애물은 비현실적인 기대치, 온보딩 문제, 부정확한 코드 생성 문제 해결, Amazon Q 기능 사용입니다. 이 종합 가이드는 이러한 문제를 해결하여 특히 에 대한 실제 시나리오, 세부 모범 사례, 문제 해결 및 실제 코드 예제를 제공합니다.Python 그리고 Java가장 널리 사용되는 프로그래밍 언어 중 두 가지입니다.

이 안내서는 Amazon Q Developer를 사용하여 다음과 같은 코드 개발 작업을 수행하는 데 중점을 둡니다.

- 코드 완료 - 개발자 코드로 인라인 제안을 실시간으로 생성합니다.
- 코드 개선 및 조언 - 소프트웨어 개발에 대해 논의하고, 자연어로 새 코드를 생성하고, 기존 코드를 개선합니다.

목표

이 가이드의 목표는 Amazon Q Developer의 신규 또는 지속적 사용자인 개발자를 지원하여 일상적인 코딩 작업에서 서비스를 성공적으로 사용할 수 있도록 지원하는 것입니다. 개발 팀 관리자는 이 가이드를 읽으면서 혜택을 얻을 수도 있습니다.

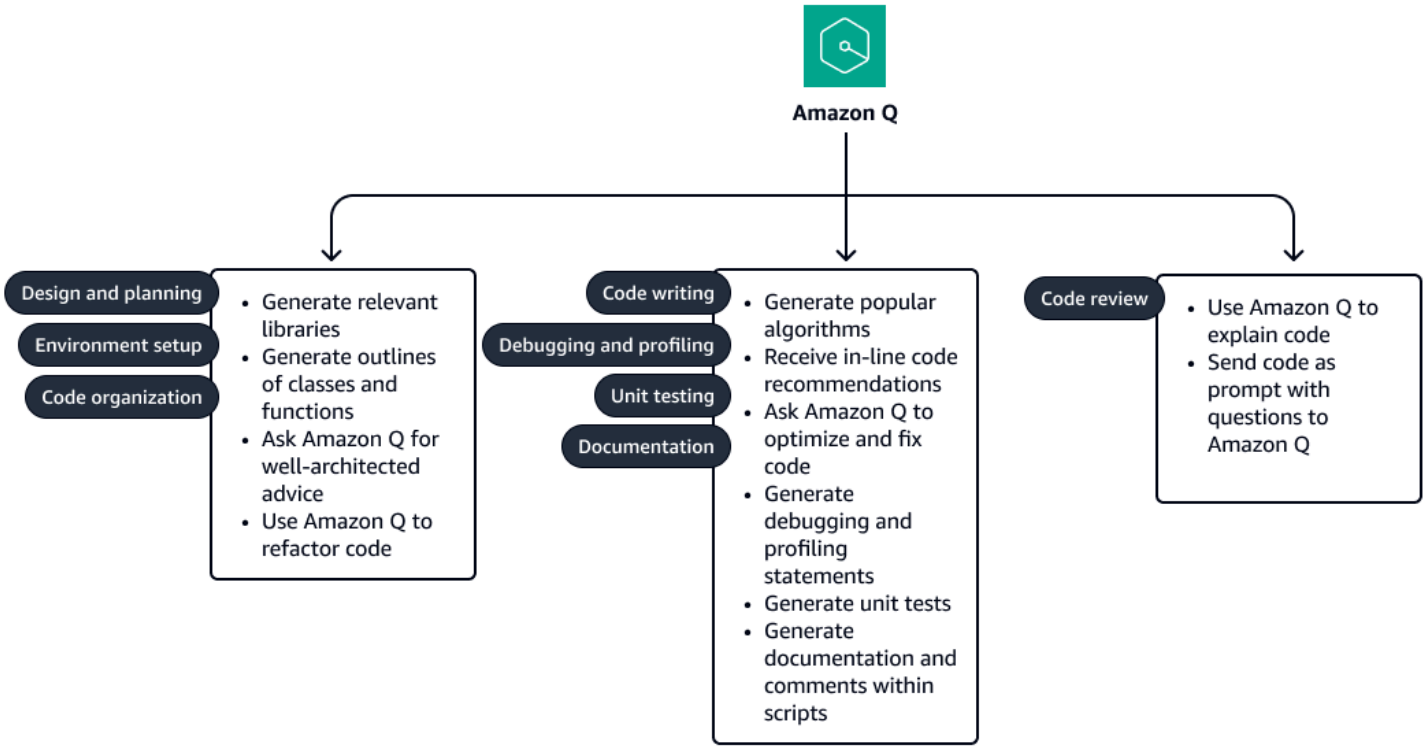
이 가이드에서는 Amazon Q Developer 사용에 대한 다음과 같은 인사이트를 제공합니다.

- 코드 개발을 위한 Amazon Q 개발자의 효과적인 사용 이해
  - Amazon Q Developer를 개발자의 워크플로에 통합하는 모범 사례를 제공합니다.

  - 성공적인 코드 생성에 대한 예제와 권장 사항이 포함된 step-by-step 지침을 제공합니다.
- Amazon Q Developer 사용에 대한 일반적인 문제를 완화하고 개발자의 명확성을 높입니다.
  - 개발자 기대치를 충족하고 코드 생성 정확도 및 성능과 관련된 장애물을 극복할 수 있는 전략과 통찰력을 제공합니다.
- 문제 해결 및 오류 처리 제공
  - 부정확한 결과 또는 예상치 못한 동작을 해결하기 위한 Amazon Q 개발자 코드 생성 문제 해결 지침을 개발자에게 제공합니다.
  - 와 관련된 실제 예제 및 시나리오 제공 Python 그리고 Java.
- 워크플로 및 생산성 최적화
  - Amazon Q Developer를 사용하여 코드 개발 워크플로를 최적화합니다.
  - 개발자 생산성을 높이기 위한 전략을 논의합니다.

개발자 워크플로에서 Amazon Q Developer 사용

개발자는 요구 사항 수집, 설계 및 계획, 코딩, 테스트, 코드 검토 및 배포 단계를 포함하는 표준 워크플로를 따릅니다. 이 섹션에서는 Amazon Q Developer 기능을 사용하여 주요 개발 단계를 최적화하는 방법에 중점을 둡니다.



이전 다이어그램은 Amazon Q Developer가 코드 개발 단계에서 다음과 같은 일반적인 작업을 가속화하고 간소화하는 방법을 보여줍니다.

- 설계 및 계획 | 환경 설정 | 코드 구성
  - 관련 라이브러리 생성
  - 클래스 및 함수 개요 생성
  - Amazon Q에 잘 설계된 조언 요청
  - Amazon Q를 사용하여 코드 리팩터링
- 코드 작성 | 디버깅 및 프로파일링 | 유닛 테스트 | 설명서
  - 인기 알고리즘 생성
  - 인라인 코드 권장 사항 수신
  - Amazon Q에 코드 최적화 및 수정 요청

  - 디버깅 및 프로파일링 문 생성
  - 단위 테스트 생성
  - 스크립트 내에서 설명서 및 의견 생성
- 코드 검토
  - Amazon Q에 코드 설명 요청
  - Amazon Q에 질문이 포함된 코드를 프롬프트로 전송

설계 및 계획

비즈니스 및 기술 요구 사항을 수집한 후 개발자는 새로운 코드베이스를 설계하거나 기존 코드베이스를 확장합니다. 이 단계에서 Amazon Q Developer는 개발자가 다음 작업을 수행하도록 지원할 수 있습니다.

- 잘 설계된 조언을 위해 관련 라이브러리와 클래스 및 함수 개요를 생성합니다.
- 엔지니어링, 호환성 및 아키텍처 설계 쿼리에 대한 지침을 제공합니다.

코딩

코딩 프로세스는 Amazon Q Developer를 사용하여 다음과 같은 방식으로 개발을 가속화합니다.

- 환경 설정 - 통합 개발 환경(IDE)(예: VS 코드 또는 IntelliJ) AWS Toolkit 에 를 설치합니다. IntelliJ 그런 다음 Amazon Q를 사용하여 라이브러리를 생성하거나 프로젝트 목표에 따라 설정 제안을 받습니다. 자세한 내용은 Amazon Q 개발자 온보딩 모범 사례를 참조하세요.
- 코드 조직 - 프로젝트 목표에 맞는 코드를 리팩터링하거나 Amazon Q에서 조직 권장 사항을 얻습니다.
- 코드 작성 - 인라인 제안을 사용하여 개발하는 동안 코드를 생성하거나 에서 Amazon Q 채팅 패널을 사용하여 Amazon Q에 코드를 생성하도록 요청합니다IDE. 자세한 내용은 Amazon Q 개발자 를 사용한 코드 생성 모범 사례를 참조하세요.
- 디버깅 및 프로파일링 - 프로파일링 명령을 생성하거나 수정 및 설명과 같은 Amazon Q 옵션을 사용하여 문제를 디버깅합니다.
- 단위 테스트 - 채팅 세션 중에 Amazon Q에 프롬프트로 코드를 제공하고 해당 단위 테스트 생성을 요청합니다. 자세한 내용은 Amazon Q 개발자의 코드 예제를 참조하세요.
- 설명서 - 인라인 제안을 사용하여 주석 및 문서 문자열을 생성하거나 설명 옵션을 사용하여 코드 선택에 대한 세부 요약을 생성합니다. 자세한 내용은 Amazon Q 개발자의 코드 예제를 참조하세요.

코드 검토

검토자는 개발 코드를 프로덕션으로 승격하기 전에 이해해야 합니다. 이 프로세스를 가속화하려면 Amazon Q 설명 및 최적화 옵션을 사용하거나 채팅 세션에서 사용자 지정 프롬프트 지침과 함께 코드 선택을 Amazon Q에 전송합니다. 자세한 내용은 채팅 예제를 참조하세요.

통합 및 배포

Amazon Q에 프로젝트의 아키텍처와 관련된 지속적인 통합, 전송 파이프라인 및 배포 모범 사례에 대한 지침을 문의하세요.

이러한 권장 사항을 사용하면 Amazon Q Developer 기능을 효과적으로 활용하여 워크플로를 최적화하고 전체 개발 수명 주기에서 생산성을 높이는 방법을 배울 수 있습니다.

Amazon Q Developer의 고급 기능

이 가이드는 실습 프로그래밍 작업에서 Amazon Q Developer를 사용하는 데 중점을 두지만 다음과 같은 고급 기능을 알고 있어야 합니다.

- Amazon Q Developer 코드 변환
- Amazon Q Developer 사용자 지정

Amazon Q Developer 코드 변환

코드 변환을 위한 Amazon Q Developer Agent는 코드를 수동으로 다시 작성할 필요 없이 파일의 코드 언어 버전을 업그레이드할 수 있습니다. 기존 코드 파일을 분석하고 새 버전의 언어를 사용하도록 자동으로 다시 작성하는 방식으로 작동합니다. 예를 들어 Amazon Q는와 같은 IDE에서 작업하는 경우 단일 모듈을 변환합니다Eclipse. Visual Studio 코드를 사용하는 경우 Amazon Q는 전체 프로젝트 또는 워크스페이스를 변환할 수 있습니다.

다음과 같은 일반적인 코드 업그레이드 작업을 수행하려는 경우 Amazon Q를 사용합니다.

- 언어 버전의 새 구문으로 작동하도록 코드를 업데이트합니다.
- 단위 테스트를 실행하여 성공적인 컴파일 및 실행을 검증합니다.
- 배포 문제를 확인하고 해결합니다.

Amazon Q는 개발자가 코드 기반을 업그레이드하기 위해 며칠에서 몇 달에 이르는 지루하고 반복적인 작업을 줄일 수 있습니다.

2024년 6월부터 Amazon Q Developer는 Java 코드 업그레이드를 지원하며 8개의 코드를 Java 11 또는 Java 17과 같은 최신 버전으로 변환할 수 있습니다.

Amazon Q Developer 사용자 지정

Amazon Q Developer는 사용자 지정 기능을 사용하여 회사의 자체 코드베이스를 기반으로 인라인 제안을 제공할 수 있습니다. 회사는 코드 리포지토리를 Amazon Simple Storage Service(Amazon S3) 또는 이전에 AWS CodeConnections에 제공합니다. AWS CodeStar 그런 다음 Amazon Q는 보안이 활성화된 사용자 지정 코드 리포지토리를 사용하여 해당 조직의 개발자와 관련된 코딩 패턴을 추천합니다.

Amazon Q Developer 사용자 지정을 사용할 때는 다음 사항에 유의하세요.

- 2024년 6월부터 Amazon Q Developer Customizations 기능은 미리 보기 모드입니다. 따라서 가용성 및 지원이 제한될 수 있습니다.
- 사용자 지정 인라인 코드 제안은 제공된 코드 리포지토리의 품질을 고려할 때만 정확합니다. 생성하는 각 사용자 지정에 대한 평가 점수를 검토하는 것이 좋습니다.
- 성능을 최적화하려면 모든 소스 파일이 10MB개 이상 포함하는 것이 좋습니다. 리포지토리가 메타데이터 파일(예: 구성 파일, 속성 파일 및 readme 파일)이 아닌 참조 가능한 소스 코드로 구성되어 있는지 확인합니다.

Amazon Q Developer 사용자 지정을 사용하면 다음과 같은 방법으로 시간을 절약할 수 있습니다.

- 자체 회사 독점 코드를 기반으로 하는 권장 사항을 사용합니다.
- 기존 코드 베이스의 재사용 가능성을 높입니다.
- 회사 전체에 일반화된 반복 가능한 패턴을 생성합니다.

Amazon Q Developer의 코딩 모범 사례

이 섹션에서는 Amazon Q Developer를 사용한 코딩 모범 사례를 설명합니다. 모범 사례에는 다음 범주가 포함됩니다.

- 온보딩 - 온보딩 시 방법 및 고려 사항
- 코드 생성 - 코드 생성을 성공적으로 사용하기 위한 지침
- 코드 권장 사항 - 코드 개선을 위한 기법

Amazon Q Developer 온보딩 모범 사례

Amazon Q Developer는 Visual Studio Code 및 JetBrains와 같은 인기 있는 IDEs를 통해 사용할 수 있는 강력한 생성형 AI 코딩 어시스턴트입니다. JetBrains 이 섹션에서는 Amazon Q Developer에 액세스하고 코딩 개발 환경에 온보딩하는 모범 사례에 중점을 둡니다.

Amazon Q Developer의 사전 조건

Amazon Q Developer는 AWS Toolkit for Visual Studio Code 및 AWS Toolkit for JetBrains (예: IntelliJ 및 PyCharm)의 일부로 사용할 수 있습니다. Visual Studio Code 및 JetBrains IDEs 경우 Amazon Q Developer는 Python, Java, JavaScript, TypeScript, C#, Go, Rust, PHP, Ruby, Kotlin, C, C++, Shell 스크립팅, SQL 및 Scala를 지원합니다.

Visual Studio 코드와 JetBrains IDE 모두에 AWS Toolkit 대한 설치에 대한 자세한 지침은 Amazon Q Developer 사용 설명서의 IDE에서 Amazon Q Developer 확장 또는 플러그인 설치를 참조하세요.

Amazon Q Developer 사용 모범 사례

Amazon Q Developer 사용 시 일반적인 모범 사례는 다음과 같습니다.

- 관련 컨텍스트를 제공하여 사용 중인 프로그래밍 언어, 프레임워크 및 도구와 같은 보다 정확한 응답을 얻을 수 있습니다. 복잡한 문제를 더 작은 구성 요소로 나눕니다.
- 프롬프트와 질문을 실험하고 반복합니다. 프로그래밍에는 종종 다양한 접근 방식을 시도하는 것이 포함됩니다.
- 코드 제안을 수락하기 전에 항상 검토하고 필요에 따라 편집하여 의도한 대로 정확하게 작동하는지 확인합니다.
- 사용자 지정 기능을 사용하여 Amazon Q Developer가 내부 라이브러리, APIs, 모범 사례 및 아키텍처 패턴을 인식하도록 하여 관련성이 더 높은 추천을 받을 수 있습니다.

Amazon Q Developer의 데이터 프라이버시 및 콘텐츠 사용

Amazon Q Developer를 사용하기로 결정할 때는 데이터와 콘텐츠가 어떻게 사용되는지 이해해야 합니다. 다음은 핵심 사항입니다.

- Amazon Q Developer Pro 사용자의 경우 코드 콘텐츠는 서비스 개선 또는 모델 훈련에 사용되지 않습니다.
- Amazon Q Developer 프리 티어 사용자의 경우 IDE 설정 또는 AWS Organizations 정책을 통해 서비스 개선에 콘텐츠를 사용하지 않도록 선택할 수 있습니다.
- 전송된 콘텐츠는 암호화되며 저장된 콘텐츠는 저장 시 암호화 및 액세스 제어로 보호됩니다. 자세한 내용은 Amazon Q Developer 사용 설명서의 Amazon Q Developer의 데이터 암호화를 참조하세요.

Amazon Q Developer를 사용한 코드 생성 모범 사례

Amazon Q Developer는 자동 코드 생성, 자동 완성 및 자연어 코드 제안을 제공합니다. 다음은 Amazon Q Developer 인라인 코딩 지원을 사용하는 모범 사례입니다.

- 응답의 정확도를 개선하는 데 도움이 되는 컨텍스트 제공

  기존 코드로 시작하거나, 라이브러리를 가져오거나, 클래스 및 함수를 생성하거나, 코드 스켈레톤을 설정합니다. 이 컨텍스트는 코드 생성 품질을 크게 개선하는 데 도움이 됩니다.
- 자연스럽게 코딩

  강력한 자동 완성 엔진과 같은 Amazon Q Developer 코드 생성을 사용합니다. 평소와 같이 코드를 지정하고 입력하거나 일시 중지할 때 Amazon Q가 제안을 제공하도록 합니다. 코드 생성을 사용할 수 없거나 코드 문제가 있는 경우 PC의 Alt+C 또는 MacOS의 Option+C를 입력하여 Amazon Q를 시작합니다. 인라인 제안을 사용하는 동안 수행할 수 있는 일반적인 작업에 대한 자세한 내용은 Amazon Q Developer 사용 설명서의 단축키 사용을 참조하세요.
- 스크립트의 목표와 관련된 가져오기 라이브러리 포함

  Amazon Q가 컨텍스트를 이해하고 그에 따라 코드를 생성하는 데 도움이 되는 관련 가져오기 라이브러리를 포함합니다. Amazon Q에 관련 가져오기 문을 제안하도록 요청할 수도 있습니다.
- 명확하고 집중적인 컨텍스트 유지

  스크립트를 특정 목표에 초점을 맞추고 고유한 기능을 관련 컨텍스트가 있는 별도의 스크립트로 모듈화합니다. 시끄럽거나 혼란스러운 컨텍스트를 피합니다.
- 프롬프트로 실험

다양한 프롬프트를 탐색하여 Amazon Q를 넛지하여 코드 생성에서 유용한 결과를 생성합니다. 예를 들어 다음 접근 방식을 시도해 보세요.

- 자연어 프롬프트에 표준 주석 블록을 사용합니다.
- 설명과 함께 스켈레톤을 생성하여 클래스와 함수를 채웁니다.
- 일반화 대신 세부 정보를 제공하여 프롬프트에 구체적이어야 합니다.

- Amazon Q Developer와 채팅하고 지원 요청

  Amazon Q Developer가 정확한 제안을 제공하지 않는 경우 IDE에서 Amazon Q Developer와 채팅하세요. 코드 조각 또는 전체 클래스 및 함수를 제공하여 컨텍스트를 시작할 수 있습니다. 자세한 내용은 Amazon Q Developer 사용 설명서의 코드에 대해 Amazon Q Developer와 채팅을 참조하세요.

Amazon Q Developer의 코드 권장 사항 모범 사례

Amazon Q Developer는 개발자 질문을 받고 코드를 평가하여 코드 생성 및 버그 수정부터 자연어를 사용한 지침까지 다양한 권장 사항을 제공할 수 있습니다. 다음은 Amazon Q에서 채팅을 사용하는 모범 사례입니다.

- 처음부터 코드 생성

  새 프로젝트의 경우 또는 일반 함수(예: Amazon S3에서 파일 복사)가 필요한 경우 Amazon Q Developer에 자연어 프롬프트를 사용하여 코드 예제를 생성하도록 요청합니다. Amazon Q는 추가 검증 및 조사를 위해 퍼블릭 리소스에 대한 관련 링크를 제공할 수 있습니다.

- 코딩 지식 및 오류 설명 찾기

  코딩 문제 또는 오류 메시지에 직면한 경우 코드 블록(해당하는 경우 오류 메시지 포함)과 질문을 Amazon Q Developer에 프롬프트로 제공합니다. 이 컨텍스트는 Amazon Q가 정확하고 관련성 있는 응답을 제공하는 데 도움이 됩니다.

- 기존 코드 개선

  알려진 오류를 수정하거나 코드를 최적화하려면(예: 복잡성을 줄이기 위해) 관련 코드 블록을 선택하고 요청과 함께 Amazon Q Developer로 전송합니다. 더 나은 결과를 얻으려면 프롬프트를 구체적으로 지정하십시오.

- 코드 기능 설명

  새 코드 리포지토리를 탐색할 때 코드 블록 또는 전체 스크립트를 선택하고 설명을 위해 Amazon Q Developer로 전송합니다. 보다 구체적인 설명을 위해 선택 크기를 줄입니다.

- 단위 테스트 생성

  코드 블록을 프롬프트로 전송한 후 Amazon Q Developer에 단위 테스트를 생성하도록 요청합니다. 이 접근 방식을 사용하면 코드 적용 범위 및 DevOps와 관련된 시간과 개발 비용을 절약할 수 있습니다.

- AWS 답변 찾기

  Amazon Q Developer는와 관련된 방대한 지식이 포함되어 있으므로 로 AWS 서비스 작업하는 개발자에게 유용한 리소스입니다 AWS. 특정 문제에 직면하든 AWS 서비스, 특정 오류 메시지가 발생하든 AWS, 새로운 오류 메시지를 배우려고 하든, AWS 서비스 Amazon Q는 종종 관련성이 높고 유용한 정보를 제공합니다.

  Amazon Q Developer가 제공하는 권장 사항을 항상 검토하세요. 그런 다음 필요한 편집을 수행하고 테스트를 실행하여 코드가 의도한 기능을 충족하는지 확인합니다.

Amazon Q Developer를 사용한 코드 예제

이 섹션에서는 Python 및 Java 언어에 중점을 두고 경험 및 코드 생성을 개선하는 방법에 대한 현실적인 예를 제공합니다. 인라인 예제와 함께 일부 시나리오에서는 Amazon Q Developer의 채팅 기능을 사용하여 프로그래밍하는 동안 코드 추천 프롬프트를 요청하는 방법을 보여줍니다.

자세한 정보는 다음의 주제를 참조하세요.

- Python 예제
- Java 예제
- 채팅 예제

Python 예제

다음 Python 예제에서는 Amazon Q Developer를 사용하여 인라인 코드 제안을 생성하는 방법을 보여줍니다. 이 예제에서는 자체 프로젝트에 대해 코딩 도우미를 트리거하는 방법을 보여줍니다. 다음 예제의 대부분은 다른 언어에 적용됩니다.

- 클래스 및 함수 생성
- 문서 코드
- 알고리즘 생성
- 단위 테스트 생성

클래스 및 함수 생성

전체 또는 부분 클래스 구현을 생성하려면 코드 설명을 사용합니다. 주석은 클래스의 의도를 설명합니다. 간단히 시작하고 필요한 경우 세부 정보를 추가합니다. 클래스를 생성한 후 해당 스켈레톤 내에서 함수를 계속 생성할 수 있습니다.

주석을 작성할 때는 언어의 주석 구조를 사용하는 것이 좋습니다. 에서 한 줄 주석의 Python경우 #를 사용합니다. 여러 줄 주석의 경우 ''' ''' 또는를 사용합니다""" """.

입력:

```
'''
DocumentReader class. has a name, children and optional parent.
Includes I/O functions to read and
```

```
write document
'''
```

출력:

```
class DocumentReader:
    def __init__(self, name, children, parent=None):
        self.name = name
        self.parent = parent
        self.children = children

    def read(self):
        """Reads current document and returns text"""
        pass

    def write(self, text):
        """Writes text to current document"""
        pass
```

이제 초기 클래스가 있으므로 추가 설명과 함께 추가 함수를 생성할 수 있습니다.

입력:

```
'''
function to read document from local directory path and print the first line of the
 document
'''
```

출력:

```
def read_document(self, path):
        """Reads document from path and prints first line"""
        import os
        if os.path.isfile(path):
            with open(path, 'r') as f:
                print(f.readline())
        else:
            print(f"File not found at {path}")
```

문서 코드

인라인 설명서에 의존하여 코드 문서의 시간을 단축합니다. Python 설명 나누기 추가''' '''.

입력:

```
class S3Utils:
    '''
    <place cursor here>
    '''
    def __init__(self):
        self.base_url = "https://s3"

    def generate_s3_url(self, bucket_name, region):
        return f"{self.base_url}.{region}.amazonaws.com/{bucket_name}"

# Example usage:
if __name__ == "__main__":
    s3_util = S3Utils()
    bucket_name = "my-example-bucket"
    region = "us-west-1"
    s3_url = s3_util.generate_s3_url(bucket_name, region)
    print(f"S3 URL for bucket '{bucket_name}' in region '{region}': {s3_url}")
```

출력:

```
class S3Utils:
    '''
    Class to generate S3 URL from bucket name and region
    '''
    def __init__(self):
        self.base_url = "https://s3"

    def generate_s3_url(self, bucket_name, region):
        return f"{self.base_url}.{region}.amazonaws.com/{bucket_name}"

# Example usage:
if __name__ == "__main__":
    s3_util = S3Utils()
    bucket_name = "my-example-bucket"
    region = "us-west-1"
    s3_url = s3_util.generate_s3_url(bucket_name, region)
    print(f"S3 URL for bucket '{bucket_name}' in region '{region}': {s3_url}")
```

알고리즘 생성

Amazon Q Developer는 특히 AWS SDKs. 스크립트에 원하는 함수를 지정하는 프롬프트를 추가합니다.

이 예제에서는 이름 값에 대한 사용자 ID 키 사전을 생성한 다음 버블 정렬 알고리즘을 사용하여 정렬 함수를 생성하는 방법을 보여줍니다.

입력:

```
class SortList:
    @staticmethod
    def get_user_id_name_map():
        user_id_name_map = {
            1: "Amit",
            2: "Rahul",
            3: "Vijay",
            4: "Raj",
            5: "Kumar"
        }
        return user_id_name_map


    '''
    Sort the list of user id's alphabetically using bubble sort
    '''
```

출력:

```
    @staticmethod
    def sort_user_id_name_map():
        user_id_name_map = SortList.get_user_id_name_map()
        list_of_user_ids = list(user_id_name_map.keys())
        list_of_user_names = list(user_id_name_map.values())
        print("Before sort: ", list_of_user_ids)
        for i in range(len(list_of_user_ids)):
            for j in range(len(list_of_user_ids) - i - 1):
                if list_of_user_names[j] > list_of_user_names[j + 1]:
                    list_of_user_ids[j], list_of_user_ids[j + 1] = list_of_user_ids[j +
 1], list_of_user_ids[j]
                    list_of_user_names[j], list_of_user_names[j + 1] =
 list_of_user_names[j + 1], list_of_user_names[j]
```

```
        print("After sort: ", list_of_user_ids)
```

또한 표준 절차에 대한 프롬프트를 생성할 수 있습니다. 이 예제에서 사용자는 사전을 Amazon DynamoDB 테이블에 저장하라는 프롬프트를 생성합니다. 스크립트 시작 부분에 사용자는 Amazon Q Developer에 추가 컨텍스트를 제공하는 데 도움이 되는 DynamoDB 라이브러리를 추가하라는 프롬프트를 추가합니다.

입력:

```
'''
Add libraries for Amazon Dynamo DB
'''
```

출력:

```
'''
Add libraries for Amazon Dynamo DB and JSON.

iMport necessary libraries for working with Amazon DynamoDB and JSON data.
'''

import boto3
import json
from botocore.exceptions import ClientError

def load_data_to_dynamodb(table_name, data_file):
    # Create a DynamoDB resource
    dynamodb = boto3.resource('dynamodb')

    # Get a reference to the table
    table = dynamodb.Table(table_name)

    # Load data from the JSON file
    with open(data_file, 'r') as file:
        data = json.load(file)

    # Iterate over the data and load it to the table
    for item in data:
        try:
            table.put_item(Item=item)
            print(f"Successfully loaded data: {item}")
        except ClientError as e:
```

```
            print(f"Error loading data: {e.response['Error']['Message']}")

# Example usage
load_data_to_dynamodb('myTable', 'data.json')
```

이제 사용자는 Amazon Q에 DynamoDB에 사전을 저장하는 데 도움을 요청할 수 있습니다.

입력:

```
'''
   function to store the above sorted dict to dynamodb table named users
'''
```

출력:

```
@staticmethod
    def store_user_id_name_map_in_dynamodb():
        user_id_name_map = SortList.get_user_id_name_map()
        list_of_user_ids = list(user_id_name_map.keys())
        list_of_user_names = list(user_id_name_map.values())
        dynamodb = boto3.resource('dynamodb')
        table = dynamodb.Table('users')
        for i in range(len(list_of_user_ids)):
            try:
                response = table.put_item(
                    Item={
                        'user_id': list_of_user_ids[i],
                        'user_name': list_of_user_names[i]
                    }
                )
            except ClientError as e:
                print(e.response['Error']['Message'])
```

단위 테스트 생성

Amazon Q Developer는 인라인 주석을 사용하여 단위 테스트를 생성하여 어설션 및 기타 테스트를 수행할 수 있습니다. 이 예제에서 사용자는 Amazon Q에 테스트 스크립트의 컨텍스트를 설정하기 위한 단위 테스트를 위한 해당 라이브러리를 생성하도록 요청합니다. 그런 다음 사용자는 Amazon Q에 함수에 대한 테스트 `sort_user_id_name_map` 함수를 작성하도록 요청합니다.

입력:

```
'''
Import all the required unit test libraries for python and then generate a unit test
 for sort_user_id_name_map function.
'''
```

출력:

```
import unittest
from unittest.mock import patch
from sort_user_id_name_map import sort_user_id_name_map

class TestSortUserIdNameMap(unittest.TestCase):
    @patch('sort_user_id_name_map.sort_user_id_name_map')
    def test_sort_user_id_name_map(self, mock_sort):
        mock_sort.return_value = {'id1': 'name1', 'id2': 'name2'}
        result = sort_user_id_name_map({'id2': 'name2', 'id1': 'name1'})
        self.assertEqual(result, {'id1': 'name1', 'id2': 'name2'})

if __name__ == '__main__':
    unittest.main()
```

또한 사용자와 Amazon Q Developer 간의 채팅에서 사용자의 코드 입력 프롬프트를 기반으로 단위 테스트를 제안하고 생성합니다. 자세한 내용은 채팅 예제를 참조하세요.

Java 예제

다음 Java 예제에서는 사용자가 인라인 코드 제안 생성을 시작하는 방법을 보여줍니다. 이 예제에서는 자체 프로젝트에 대해 코딩 도우미를 트리거하는 방법을 보여줍니다. 다음 예제의 대부분은 다른 언어에 적용됩니다.

- 클래스 및 함수 생성
- 문서 코드
- 알고리즘 생성
- 단위 테스트 생성

클래스 및 함수 생성

전체 또는 부분 클래스 구현을 생성하려면 코드 설명을 사용합니다. 주석은 클래스의 의도를 설명합니다. 간단히 시작하고 필요한 경우 세부 정보를 추가합니다. 클래스를 생성한 후 해당 스켈레톤 내에서 함수를 계속 생성할 수 있습니다.

주석을 작성할 때는 언어의 주석 구조를 사용하는 것이 좋습니다. 에서 Java이는 입니다`/** */`.

입력:

```
/**
 * DocumentReader class. has a name, children and optional parent.
 * Includes I/O functions to read and
 * write document
 */
```

출력:

```
/**
 * DocumentReader class. has a name, children and optional parent.
 * Includes I/O functions to read and
 * write document
 */
import java.util.ArrayList;
public class DocumentReader {
    private String name;
    private DocumentReader parent;
    private ArrayList<DocumentReader> children;

    /**
     * Constructor
     * @param name name of the document
     * @param parent parent of the document
     */
    public DocumentReader(String name, DocumentReader parent) {
        this.name = name;
        this.parent = parent;
        this.children = new ArrayList<>();
    }

    /**
     * Constructor
     * @param name name of the document
```

```
     */
    public DocumentReader(String name) {
        this(name, null);
    }
    /**
     * Getter for name
     * @return name of the document
     */
    public String getName() {
        return name;
    }

    /**
     * Getter for parent
     * @return parent of the document
     */
    public DocumentReader getParent() {
        return parent;
    }

    /**
     * Getter for children
     * @return children of the document
     */
    public ArrayList<DocumentReader> getChildren() {
        return children;
    }

    /**
     * Adds a child to the document
     * @param child child to be added
     */
    public void addChild(DocumentReader child) {
        children.add(child);
    }

    /**
     * Removes a child from the document
     * @param child child to be removed
     */
    public void removeChild(DocumentReader child) {
        children.remove(child);
    }
```

```
}
```

이제 초기 클래스가 있으므로 추가 설명과 함께 추가 함수를 생성할 수 있습니다.

입력:

```
/**
* function to read document from local directory path
* and print first line of document
*/
```

출력:

```
/**
 * function to read document from local directory path
 * and print first line of document
 */
public void readDocument() {
    try {
        BufferedReader reader = new BufferedReader(new FileReader(name));
        String line = reader.readLine();
        System.out.println(line);
        reader.close();
    } catch (IOException e) {
        System.out.println("Error reading document: " + e.getMessage());
    }
}
```

문서 코드

인라인 설명서에 의존하여 코드 문서의 시간을 단축합니다. Java 설명 나누기을 추가합니다/* */.

입력:

```
import java.util.function.Function;

public class S3Utils {
    /*
     * <place cursor here>
    */
    public static Function<String, String> generateS3Function(String bucketName, String
 region) {
```

```
        return (String key) -> {
            String url = generateS3Url(bucketName, key, region);
            return url;
        };
    }
}
```

인라인 생성이 발생하지 않는 경우 문서를 원하는 위치에 커서를 배치한 후 Amazon Q Developer 인라인 제안을 실행합니다.

출력:

```
import java.util.function.Function;

public class S3Utils {
    /*
     * Generate a function that returns the S3 URL for a given key
     * @param bucketName the name of the bucket
     * @param region the region of the bucket
     * @return the function that generates the S3 URL
    */
    public static Function<String, String> generateS3Function(String bucketName, String
 region) {
        return (String key) -> {
            String url = generateS3Url(bucketName, key, region);
            return url;
        };
    }
}
```

코드(클래스, 함수 또는 스크립트 요소)에 대한 더 간단한 설명을 보려면 설명서를 추가하려는 코드 섹션 앞에 전략적으로 배치된 문자(//)를 사용합니다.

입력:

```
//
public class S3Utils {
    /*
     * Generate a function that returns the S3 URL for a given key
     * @param bucketName the name of the bucket
     * @param region the region of the bucket
     * @return the function that generates the S3 URL
    */
```

```
    public static Function<String, String> generateS3Function(String bucketName, String
 region) {
        return (String key) -> {
            String url = generateS3Url(bucketName, key, region);
            return url;
        };
    }
}
```

출력:

```
// This class is used to generate a function that returns the S3 URL for a given key
public class S3Utils {
    /*
     * Generate a function that returns the S3 URL for a given key
     * @param bucketName the name of the bucket
     * @param region the region of the bucket
     * @return the function that generates the S3 URL
    */
    public static Function<String, String> generateS3Function(String bucketName, String
 region) {
        return (String key) -> {
            String url = generateS3Url(bucketName, key, region);
            return url;
        };
    }
}
```

알고리즘 생성

Amazon Q Developer는 특히 AWS SDKs. 스크립트에 원하는 함수를 지정하는 프롬프트를 추가합니다.

이 예제에서는 이름 값에 대한 사용자 ID 키의 해시 맵을 생성한 다음 버블 정렬 알고리즘을 사용하여 정렬 함수를 생성합니다.

입력:

```
import java.util.HashMap;
public class SortList {

    // generate key value pair hash map for
    // user id and name
```

```
    public static HashMap<Integer, String> getUserIdNameMap() {
        HashMap<Integer, String> userIdNameMap = new HashMap<Integer, String>();
        userIdNameMap.put(1, "Amit");
        userIdNameMap.put(2, "Rahul");
        userIdNameMap.put(3, "Vijay");
        userIdNameMap.put(4, "Raj");
        userIdNameMap.put(5, "Kumar");
        return userIdNameMap;
    }

    /*
    * Create function to sort UserIdName map by name alphabetically using bubble sort
    */


}
```

출력:

```
/*
* Create function to sort UserIdName map by name alphabetically using bubble sort
*/
public static HashMap<Integer, String> sortUserIdNameMap(HashMap<Integer, String>
 userIdNameMap) {
    int size = userIdNameMap.size();
    for (int i = 0; i < size; i++) {
        for (int j = i + 1; j < size; j++) {
            if (userIdNameMap.get(i).compareTo(userIdNameMap.get(j)) > 0) {
                String temp = userIdNameMap.get(i);
                userIdNameMap.put(i, userIdNameMap.get(j));
                userIdNameMap.put(j, temp);
            }
        }
    }
    return userIdNameMap;
}
```

또한 사용자는 표준 절차에 대한 프롬프트를 생성할 수 있습니다. 이 예제에서 사용자는 해시 맵을 Amazon DynamoDB 테이블에 저장하는 프롬프트를 생성합니다. 스크립트 시작 부분에 사용자는 Amazon Q Developer에 추가 컨텍스트를 제공하는 데 도움이 되는 DynamoDB Java 라이브러리를 추가하라는 프롬프트를 추가합니다.

입력:

```
// add libraries for Amazon DynamoDB
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;

     /*
     * Store sorted UserIdName map into Amazon DynamoDB table named users
     */
```

출력:

```
// add libraries for Amazon DynamoDB
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;

     /*
     * Store sorted UserIdName map into Amazon DynamoDB table named users
     */
    public static void storeSortedUserIdNameMap(HashMap<Integer, String> userIdNameMap)
 {
        // create Amazon DynamoDB client
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.defaultClient();

        // store sorted UserIdName map into Amazon DynamoDB table named users
        for (int userId : userIdNameMap.keySet()) {
            String name = userIdNameMap.get(userId);
            client.putItem("users", new HashMap<String, Object>() {{
                put("userId", userId);
                put("name", name);
            }});
        }
    }
```

단위 테스트 생성

인라인 설명을 통해 Amazon Q Developer는 단위 테스트를 생성하여 어설션 및 기타 JUnit 테스트를 수행할 수 있습니다. 이 예제에서 사용자는 먼저 Amazon Q에 JUnit용 해당 라이브러리를 생성하여 테스트 스크립트의 컨텍스트를 설정하도록 요청합니다. 그런 다음 사용자는 퍼블릭 클래스와 선언자 및 설명 문자열을 차례로 생성합니다.

입력:

```
// import libraries for java unit tests
import org.junit.Test;
import static org.junit.Assert.*;

public class UnitTests {

    @Test
    /*
     * Create unit test for getUserIdNameMap
     */
}
```

출력:

```
@Test
/*
* Create unit test for getUserIdNameMap
*/
public void testGetUserIdNameMap() {
    // create a new instance of the UserIdNameMap class
    UserIdNameMap userIdNameMap = new UserIdNameMap();

    // create a new instance of the User class
    User user = new User("John", "Doe");

    // add the user to the map
    userIdNameMap.addUser(user);

    // get the user from the map
    User retrievedUser = userIdNameMap.getUser("John");

    // assert that the retrieved user is the same as the user added to the map
    assertEquals(user, retrievedUser);
}
```

또한 사용자와 Amazon Q Developer 간의 채팅에서 사용자의 코드 입력 프롬프트를 기반으로 단위 테스트를 제안하고 생성합니다. 자세한 내용은 채팅 예제를 참조하세요.

채팅 예제

Amazon Q Developer 채팅은 개발자가 애플리케이션 코딩, 테스트 및 업그레이드부터 오류 진단, 보안 스캔 및 수정 수행, AWS 리소스 최적화에 이르기까지 다양한 작업을 수행할 수 있도록 지원합니다. 이 섹션에서는 Amazon Q와의 채팅이 다음과 같은 일반적인 프로그래밍 작업에서 개발자를 지원하는 방법을 설명합니다.

- 에 대한 질문 AWS 서비스
- 코드 생성
- 단위 테스트 생성
- 코드 설명

에 대한 질문 AWS 서비스

사용자는 올바른 서비스를 찾거나 모범 사례를 이해하는 AWS 서비스등 Amazon Q Developer에 대해 질문할 수 있습니다.

에 대한 채팅 AWS 서비스

사용자:

비용을 낮게 유지해야 하는 경우 워크로드를 실행하는 데 사용할 수 있는 최상의 컨테이너 서비스는 무엇입니까?

Amazon Q:

제공된 정보에 따르면 워크로드를 실행할 비용 효율적인 컨테이너 서비스를 찾고 있는 것 같습니다. 이 경우 Fargate에서 Amazon Elastic Container Service(Amazon ECS)를 고려하는 것이 좋습니다. [1]

Fargate를 사용하는 Amazon ECS는 기본 인프라를 관리할 필요 없이 컨테이너를 실행할 수 있는 컨테이너용 서버리스 컴퓨팅 엔진입니다. Fargate를 사용하면 서버를 프로비저닝하거나 관리할 필요가 없습니다. Fargate가 이를 대신 처리합니다. 이렇게 하면 유휴 서버 용량을 지불할 필요 없이 컨테이너가 실제로 사용하는 컴퓨팅 리소스에 대해서만 비용을 지불하므로 비용을 낮게 유지할 수 있습니다. [2]

Fargate에서 Amazon ECS를 사용하면 다음과 같은 몇 가지 주요 이점이 있습니다.

• 서버 또는 클러스터를 관리할 필요 없음 - Fargate가 인프라 프로비저닝 및 관리를 처리합니다.

• Pay-as-you-go 요금 - 유휴 용량에 대한 비용을 지불하지 않고도 컨테이너가 사용하는 컴퓨팅 리소스에 대해서만 비용을 지불합니다.

• 자동 조정 - Fargate는 필요에 따라 컨테이너를 자동으로 확장 및 축소할 수 있습니다.

• 로드 밸런싱, 로깅 및 모니터링과 AWS 서비스 같은 다른와 긴밀하게 통합됩니다.

또한 제공된 리소스는 ECS 서비스가 컨테이너 일정 및 배치 측면에서 유연성을 제공하므로 비용을 더욱 최적화하는 데 도움이 될 수 있다고 언급합니다.

특정 워크로드 요구 사항에 따라 비용을 더 잘 파악하려면 웹 사이트에서 Fargate AWS 를 사용하는 Amazon ECS의 요금 세부 정보를 검토하는 것이 좋습니다. 그리고 다른 질문이 있으면 언제든지 알려주세요!

소스

[1] 어떤 AWS 컨테이너 서비스가 가장 적합합니까?

aptgetops.tech > aws-container-services-which-one-is-right-for-you

[2] 비 HTTP 워크로드에 대한 Java 컨테이너 옵션

> 질문 > 43417016

에 대한 Amazon Q 채팅 종료 AWS 서비스

코드 생성

사용자는 Amazon Q Developer에 코드 조각 작성에 대한 지원을 요청할 수 있습니다.

코드 작성에 대한 채팅

사용자:

를 사용하여 모든 S3 버킷을 나열하는 방법의 예를 들어줄 수 있나요 AWS SDK for Python (Boto)?

Amazon Q:

다음은 AWS SDK for Python (Boto)[1]을 사용하여 모든 S3 버킷을 나열하는 방법의 예입니다.

```
import boto3
# Create an S3 client
s3 = boto3.client('s3')

# List all the buckets
response = s3.list_buckets()

# Print the name of each bucket
```

```
for bucket in response['Buckets']:
    print(bucket['Name'])
```

코드는 다음과 같이 작동합니다.

1. 먼저 Python용 SDK인 boto3 라이브러리를 가져옵니다.
2. 를 사용하여 S3 클라이언트를 생성합니다`boto3.client('s3')`.
3. S3 클라이언트에서 `list_buckets()` 메서드를 호출하면 계정의 모든 버킷 목록이 포함된 응답이 반환됩니다.
4. 그런 다음 응답의 `Buckets` 목록을 반복하고 각 버킷의 이름을 인쇄합니다.

이는 기본 예제이지만는 버킷에서 객체 생성, 삭제 및 관리, 버킷 정책 설정 등과 같은 Amazon S3 작업을 위한 더 많은 고급 기능을 AWS SDK for Python (Boto) 제공합니다. 더 많은 예제와 설명서는 AWS SDK for Python (Boto)) 설명서에서 확인할 수 있습니다. [2]

로 작업할 때는 코드에서 액세스 키를 하드코딩하는 대신 환경 변수에 액세스 키를 저장하는 AWS 등 자격 증명 및 민감한 정보를 처리하는 모범 사례를 따르는 것이 AWS 서비스중요합니다.

소스: [1] SDK for Python (Boto3) - AWS SDK 코드 예제 docs.aws.amazon.com > code-library > latest-1x > ... > python-s3.html을 사용하는 Amazon S3 예제

[2] AWS SDK - AWS SDK 코드 예제 docs.aws.amazon.com > code-library > latest-1x > ... > s3_example_s3_buckets_GettingStarted_sect...를 사용하여 Amazon S3 버킷 및 객체 시작하기

코드 작성에 대한 Amazon Q 채팅 종료

단위 테스트 생성

사용자는 선택한 코드를 Amazon Q Developer에 프롬프트로 전송할 수 있으며, 여기서 단위 테스트 생성과 같은 추가 지침이 적용됩니다. step-by-step 지침은 Amazon Q Developer 사용 설명서의 Amazon Q로 코드 전송을 참조하세요.

단위 테스트에 대한 채팅

사용자:

다음 함수에 대한 모의 데이터로 단위 테스트를 작성합니다.

```
// poll messages from Amazon SQS queue name
   public List<String> pollMessages(String queueName) {
       List<String> messages = new ArrayList<>();
```

```
        // poll messages from queue
        ReceiveMessageResult = sqsClient.receiveMessage(new ReceiveMessageRequest().
        for (Message : receiveMessageResult.getMessages()) {
            messages.add(message.getBody());
        }
        return messages;
        }
```

Amazon Q:

pollMessages 함수에 대한 모의 데이터로 단위 테스트를 작성하려면 Mockito와 같은 모의 프레임워크를 사용할 수 있습니다. 다음은 그 예입니다.

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.junit.jupiter.MockitoExtension;

import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;

@ExtendWith(MockitoExtension.class)
public class SQSUtilsTest {

    @Mock
    private AmazonSQS sqsClient;

    @InjectMocks
    private SQSUtils sqsUtils;

    @Test
    public void testPollMessages() {
        // Arrange
        String queueName = "test-queue";
        Message message1 = new Message().withBody("message1");
```

```
        Message message2 = new Message().withBody("message2");
        ReceiveMessageResult receiveMessageResult = new
 ReceiveMessageResult().withMessages(message1, message2);
        Mockito.when(sqsClient.receiveMessage(new
 ReceiveMessageRequest().withQueueUrl(queueName)))
                .thenReturn(receiveMessageResult);

        // Act
        List<String> messages = sqsUtils.pollMessages(queueName);

        // Assert
        assertEquals(Arrays.asList("message1", "message2"), messages);
    }
}
```

이 예제에서는 Mockito를 사용하여 AmazonSQS 클라이언트를 모의하고 receiveMessage 메서드에 대한 예상 동작을 설정하여 본문이 "message1" 및 "message2"인 모의 메시지 2개를 반환합니다. 그런 다음 pollMessages 메서드를 호출하고 반환된 메시지 목록이 예상 값과 일치한다고 어설션합니다.

핵심 사항은 다음과 같습니다.

1. @ExtendWith(MockitoExtension.class) 주석을 사용하여 JUnit와 Mockito 통합을 활성화합니다.
2. @Mock 주석을 사용하여 AmazonSQS 클라이언트의 모의 인스턴스를 생성합니다.
3. @InjectMocks 주석을 사용하여 모의 AmazonSQS 클라이언트를 SQSUtils 클래스에 주입합니다.
4. 테스트 방법에서는 Mockito.when().thenReturn()을 사용하여 receiveMessage 메서드의 예상 동작을 설정합니다.
5. pollMessages 메서드를 호출하고 반환된 메시지 목록이 예상 값과 일치한다고 주장합니다.

이 접근 방식을 사용하면 Amazon SQS 서비스와 실제로 상호 작용하지 않고도 pollMessages 메서드를 테스트할 수 있으므로 테스트가 더 빠르고 안정적일 수 있습니다.

단위 테스트에 대한 Amazon Q 채팅 종료

코드 설명

코드 검토 프로세스의 경우 개발자는 코드 선택을 Amazon Q Developer로 전송하여 코드를 더 잘 이해할 수 있습니다. 또한 Amazon Q가 제공하는 설명은 설명서 목적에도 도움이 될 수 있습니다. 다음 예제에서 Amazon Q는 함수로 핸들러 AWS Lambda 함수를 설명한 다음 문서스트링을 제안합니다.

코드 설명에 대한 채팅

사용자:

코드의 다음 부분을 설명합니다.

```
def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb')
    s3_client = boto3.client('s3')
    table = dynamodb.Table('DYNAMODB_TABLE')
    # Get s3 object from the event
    print(event)
    print(context)
    # get object from event
    bucket_key = event['Records'][0]['s3']['object']['key']
    object_key = unquote_plus(bucket_key)
    fullText = detect_text(object_key, bucket)

    response = s3_client.head_object(Bucket=bucket, Key=object_key)
    print(response)
    print(fullText)
    metadata = response['ResponseMetadata']
    metadata = json.dumps(metadata)

    now = datetime.now()
    current_time = now.strftime("%A %d %B, %Y %H:%M:%S")
    print("Current Time =", current_time + "\n\n")
    reason, outcome = detect_restaurant_closure(fullText, current_time)
    table.put_item(
        Item={
            'id': object_key,
            'creationTime': now.strftime("%Y-%m-%d %H:%M:%S"),
            'reason': reason,
            'text': fullText,
            'metadata': metadata
        }
    )
```

Amazon Q:

제공된 코드 블록은에 작성된 AWS Lambda 함수인 것처럼 보입니다Python. 이 코드의 주요 측면을 설명하겠습니다.

1. 함수의 이름은 lambda_handler이며 Lambda 함수의 진입점입니다.
2. boto3 라이브러리를 사용하여 AWS DynamoDB 및 S3 서비스와 상호 작용합니다.
3. 함수는 수신 이벤트 데이터에서 S3 객체 키와 버킷 이름을 검색합니다.
4. 그런 다음 detect_text 함수(제공된 코드에 표시되지 않음)를 호출하여 S3 객체의 전체 텍스트 콘텐츠를 추출합니다.
5. 또한이 함수는 head_object 메서드를 사용하여 S3 객체에 대한 메타데이터를 검색합니다.
6. 그런 다음 현재 시간을 추출하고 detect_restaurant_closure 함수(도시되지 않음)를 호출하여 추출된 텍스트를 기반으로 레스토랑 폐쇄의 이유와 결과를 결정합니다.
7. 마지막으로 함수는 DynamoDB 테이블에 다음 정보를 저장합니다.
   - ID로 사용되는 S3 객체 키
   - 생성 시간
   - 레스토랑 문을 닫은 이유와 결과
   - 추출된 S3 객체 텍스트

전반적으로이 Lambda 함수는 S3 객체를 처리하고, 텍스트 콘텐츠를 추출하고, 레스토랑 폐쇄 정보를 분석하고, 결과를 DynamoDB 테이블에 저장하는 더 큰 시스템의 일부인 것으로 보입니다. 이 코드의 기능을 완전히 이해하려면 detect_text 및 detect_restaurant_closure 함수의 특정 세부 정보가 필요합니다.

Amazon Q 응답 종료

사용자:이 설명을이 lambda_handler 함수의 문서 문자열에 요약합니다.

Amazon Q: lambda_handler 함수[1]에 대한 제안 문서 문자열은 다음과 같습니다.

```
def lambda_handler(event, context):
    """
    AWS Lambda function that processes an S3 object, extracts text content, analyzes
 it, and stores the results.
    Args:
        event (dict): The event data that triggered the Lambda function, containing
 information about the S3 object.
        context (object): The context object provided by AWS Lambda, containing
 information about the runtime environment.

    Returns:
        None
    """
```

```
    # function implementation details...
```

이 문서스트링은 함수의 용도, 예상 입력 파라미터 및 예상 반환 값(또는이 경우 부족)에 대한 개략적인 요약을 제공합니다. 이를 통해 사용자는 구현 세부 정보를 자세히 살펴보지 않고도 lambda_handler 함수가 수행하는 작업을 명확하게 이해할 수 있습니다.

소스: [1] C# - AWS Lambda https://docs.aws.amazon.com/lambda/latest/dg/csharp-handler.html - :~:text=에서 Lambda 함수 핸들러 정의Lambda 함수 핸들러는 , , 종료 또는 시간 초과입니다.

코드 설명에 대한 Amazon Q 채팅 종료

Amazon Q Developer의 코드 생성 시나리오 문제 해결

Amazon Q Developer를 사용할 때 코드 생성 및 해결이 부정확한 다음과 같은 일반적인 시나리오가 발생할 수 있습니다.

빈 코드 생성

코드를 개발하는 동안 다음과 같은 문제가 발생할 수 있습니다.

- Amazon Q는 제안을 제공하지 않습니다.
- 'Amazon Q의 제안 없음' 메시지가 IDE에 나타납니다.

첫 번째 생각은 Amazon Q가 제대로 작동하지 않는 것일 수 있습니다. 그러나 이러한 문제의 근본 원인은 일반적으로 스크립트의 컨텍스트 또는 IDE 내의 열린 프로젝트와 관련이 있습니다.

Amazon Q Developer가 제안을 자동으로 제공하지 않는 경우 다음 바로 가기를 사용하여 Amazon Q 인라인 제안을 수동으로 실행할 수 있습니다.

- PC - Alt+C
- MacOS - Option+C

자세한 내용은 Amazon Q Developer 사용 설명서의 단축키 사용을 참조하세요.

대부분의 시나리오에서 Amazon Q는 제안을 생성합니다. Amazon Q가 "Amazon Q의 제안 사항 없음" 메시지를 반환하면이 문제에 대해 다음과 같은 일반적인 해결 방법을 검토합니다.

- 부적절한 컨텍스트 - 해당 라이브러리가 있는지 확인합니다. 스크립트에 정의된 클래스 및 함수가 새 코드와 관련이 있는지 확인합니다.
- 모호한 프롬프트 - 프롬프트가 혼란스럽거나 명확하지 않은 경우 인라인 코드 제안을 하면 Amazon Q의 성능이 저하됩니다. 이러한 시나리오에서는 코드와 설명에 더 구체적으로 명시하여 적절한 응답을 유도합니다.

멈춘 경우 최대한 코딩을 계속합니다. 일반적으로 Amazon Q는 의도를 인식하여 제안을 제공합니다. 또 다른 옵션은 Amazon Q 채팅에서 프롬프트 지침이 포함된 코드 조각을 보내는 것입니다. 자세한 내용은 Amazon Q Developer 사용 설명서의 Amazon Q로 코드 전송을 참조하세요.

지속적인 설명

주석을 사용하여 프롬프트를 작성하고 Amazon Q로부터 일반 또는 반복 주석 줄만 받을 수 있습니다.

입력:

```
// Kafka test
```

결과:

```
// Kafka test
// Kafka test
// Kafka test
// Kafka test
```

일반적으로이 문제는 스크립트에 컨텍스트가 없거나 프롬프트 지침이 잘못되었기 때문입니다. 언어가 지정된 주석에는 표준 형식을 사용하는 것이 좋습니다. 이 예제에서 사용자는 설명 블록에를 사용하도록 프롬프트`/* */` 를 변경했으며,이 변경으로 Amazon Q의 성능이 향상되었습니다.

입력:

```
/*
 * Create a unit test to push a message to a Kafka topic
 */
```

결과:

이 결과 사용자는 여러 제안을 순차적으로 수락했습니다.

```
/*
 * Create a unit test to push a message to a Kafka topic
 */
import org.junit.Test;
import java.util.Properties;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;
import static org.junit.Assert.assertEquals;
```

```
// create function for a Kafka test
public class KafkaTest {
    @Test
    public void testKafka() {
        // create Kafka producer
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("key.serializer",
 "org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer",
 "org.apache.kafka.common.serialization.StringSerializer");
        KafkaProducer<String, String> producer = new KafkaProducer<>(props);
        // create Kafka topic
        String topic = "test";
        // send message to Kafka topic
        producer.send(new ProducerRecord<>(topic, "key", "value"));
        // close Kafka producer
        producer.close();
        // assert that message was sent successfully
        assertEquals("value", KafkaTest.consumer.poll().value());
    }
}
```

잘못된 인라인 코드 생성

경우에 따라 Amazon Q Developer 인라인 제안이 유용하지 않거나 프롬프트의 목표를 충족하지 못할 수 있습니다. 때때로 인라인 코드 제안은 간단한 Hello World 메서드와 유사한 일반 함수를 제공할 수 있습니다.

인라인 코드 제안이 도움이 되지 않는 경우 다음 모범 사례를 시도해 보세요.

- 스크립트의 컨텍스트를 검토합니다. 일반적으로 제공하는 컨텍스트가 많을수록 Amazon Q에서 더 나은 결과를 얻을 수 있습니다.
- 컨텍스트가 다르거나 노이즈가 있는 경우 코드를 모듈화합니다. 동일한 스크립트 내에서 유사한 함수를 그룹화합니다. 목표에 따라 서로 다른 클래스와 함수를 구분합니다.
- Amazon Q가 더 나은 추천을 제공할 수 있도록 클래스 및 함수 이름을 전술적으로 작성합니다.

이 예제에서 개발자는 하둡에 파일을 업로드하는 스크립트를 처음부터 작성하고 있습니다.

입력:

```
class  MyFunction{

    // create function to read data locally and upload to HDFS filesystem
```

결과:

```
class  MyFunction{

    // create function to read local data file path and upload to HDFS filesystem
    public void readAndUploadToHDFS(String inputPath, String outputPath) throws
 IOException {
        // read data from local file
        BufferedReader in = new BufferedReader(new FileReader(inputPath));

}
```

Amazon Q는 권장 사항을 제공했지만 개발자의 기대치를 충족하지 못했습니다. 성능을 개선하기 위해 개발자는 라이브러리를 가져오고 Hadoop 유틸리티를 나타내도록 클래스 이름을 변경합니다.

수정:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.FileReader;

// import applicable hadoop libraries for IO operations
import org.apache.hadoop.fs.Path;

class  HadoopUtilities{

    // create function to read local data file path and upload to HDFS filesystem
}
```

결과:

이 결과 개발자는 여러 제안을 순차적으로 수락했습니다.

```
class  HadoopUtilities{

    // create function to read local data file path and upload to HDFS filesystem
    public static void uploadToHDFS(String localFilePath, String hdfsFilePath) throws
 IOException {
```

```
        // create HDFS file path object
        Path hdfsPath = new Path(hdfsFilePath);
        // create local file path object
        Path localPath = new Path(localFilePath);
        // create hadoop file system object
        org.apache.hadoop.fs.FileSystem hdfs = org.apache.hadoop.fs.FileSystem.get(new
 org.apache.hadoop.conf.Configuration());
        // upload local file to HDFS
        hdfs.copyFromLocalFile(localPath, hdfsPath);
        // close HDFS file system
        hdfs.close();
        System.out.println("File uploaded successfully!");
}
```

동일한 스크립트에서 개발자는 Amazon Simple Queue Service(Amazon HadoopUtilities SQS) 주제를 읽는 주석 프롬프트를 추가합니다.Amazon SQS

입력:

```
// read message from SQS
```

결과:

```
// read message from SQS
    public static void readMessageFromSQS(String message) {
        System.out.println("Received message from SQS: " + message);
```

Amazon Q는 Java 메서드를 사용하여 Amazon SQS 메시지를 읽는 대신 문자열 메서드 인쇄를 제안합니다. 그러나이 제안은 개발자의 원래 목표를 충족하지 않습니다.

이 경우 코드를 다시 모듈화하는 것이 좋습니다. 스크립트는 하둡 메서드에 중점을 둡니다. 따라서 SQS 작업 전용 유틸리티 스크립트를 생성하는 것이 더 나은 방법일 수 있습니다.

SQS 유틸리티에 대해 별도의 클래스를 생성하면 HadoopUtilities 스크립트에서 사용할 수 있는 인라인 제안이 개선됩니다.

입력:

```
// import function for SQS
import com.amazonaws.services.sqs.AmazonSQS;
import java.util.ArrayList;
import java.util.List;
```

```
class SQSUtils {

    // instantiate SQS client
    private AmazonSQS sqsClient;

    // load SQS object
    public SQSUtils(AmazonSQS sqsClient) {
        this.sqsClient = sqsClient;
    }

    // poll message from Amazon SQS given queue name
}
```

결과:

```
// poll message from Amazon SQS given queue name
    public List<String> pollMessages(String queueName) {
        List<String> messages = new ArrayList<>();
        // poll messages from queue
        receiveMessageResult = sqsClient.receiveMessage(new
 ReceiveMessageRequest(queueName));
        for (Message message : receiveMessageResult.getMessages()) {
            messages.add(message.getBody());
        }
        return messages;
```

인라인 제안이 여전히 예상 결과를 제공하지 않는 경우 채팅 세션에서 Amazon Q Developer에 코드를 생성하도록 요청합니다.

코드 생성에 대한 채팅

사용자:

Amazon SQS 대기열에서 메시지를 읽는 Java 함수 생성

Amazon Q:

다음은 Amazon SQS 대기열에서 메시지를 읽는 Java 함수의 예입니다.

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.model.Message;
```

```
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
public class SQSMessageReader {
    private AmazonSQS sqsClient;
    private String queueUrl;

    public SQSMessageReader(AmazonSQS sqsClient, String queueUrl) {
        this.sqsClient = sqsClient;
        this.queueUrl = queueUrl;
    }

    public String readMessage() {
        ReceiveMessageRequest receiveMessageRequest = new
 ReceiveMessageRequest(queueUrl)
                .withMaxNumberOfMessages(1)
                .withWaitTimeSeconds(10);

        ReceiveMessageResult receiveMessageResult =
 sqsClient.receiveMessage(receiveMessageRequest);

        if (receiveMessageResult.getMessages().size() > 0) {
            Message message = receiveMessageResult.getMessages().get(0);
            String messageBody = message.getBody();
            String receiptHandle = message.getReceiptHandle();

            // Process the message
            System.out.println("Received message: " + messageBody);

            // Delete the message from the queue
            sqsClient.deleteMessage(queueUrl, receiptHandle);

            return messageBody;
        } else {
            return null;
        }
    }
```

이 코드는 AmazonSQS 클라이언트와 대기열 URL을 생성자 파라미터로 사용하는 SQSMessageReader 클래스를 생성합니다. readMessage() 메서드는 ReceiveMessageRequest를 사용하여 대기열에서 최대 1개의 메시지를 가져오고 최대 대기 시간은 10초입니다.

메시지가 수신되면 메시지 본문이 인쇄되고 DeleteMessageRequest를 사용하여 대기열에서 메시지가 삭제됩니다. 그러면 메시지 본문이 반환됩니다. 메시지가 수신되지 않으면 null이 반환됩니다.

코드 생성에 대한 Amazon Q 채팅 종료

채팅의 부적절한 결과

코드를 개발하는 동안 개발자는 Amazon Q에 공통 함수를 생성하거나, 권장 사항을 제공하거나, 코드를 설명하도록 요청할 수 있습니다. 경우에 따라 프롬프트(예: 질문 또는 코드 조각)가 주어지면 Amazon Q는 일반적인 피드백이나 기대치를 충족하지 않는 결과를 제공할 수 있습니다. 이러한 시나리오에서는 다음을 시도해 보세요.

- 다양한 프롬프트로 실험하여 Amazon Q에서 얻은 코드 생성 결과를 개선합니다.
- Amazon Q가 프롬프트 또는 질문에 대한 응답에 자주 제공하는 URLs을 따릅니다.
- Amazon Q의 향후 성능을 개선하는 데 도움이 되는 피드백을 제공합니다.

예를 들어 개발자는 검색 증강 생성(RAG) 솔루션을 생성하고 있으며 문장 간의 의미 관계를 고려하여 문서를 청크해야 합니다. 개발자는 다음과 같이 Amazon Q와 채팅합니다.

RAG 솔루션에 대한 채팅

사용자:

RAG 솔루션의 경우 의미 체계 청킹을 위한 코드 생성

Amazon Q:

LangChain 라이브러리를 Python 사용하여에서 검색 증강 생성(RAG) 솔루션에 대한 의미 체계 청킹을 구현하는 방법의 예는 다음과 같습니다.

```
from langchain.text_splitter import CharacterTextSplitter, TokenTextSplitter
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.vectorstores import Chroma

# Initialize the text splitter
text_splitter = TokenTextSplitter(
    chunk_size=500,
    chunk_overlap=50,
    length_function=len
)
# Initialize the embeddings model
embeddings = HuggingFaceEmbeddings()
```

```
# Create the vector store
vectorstore = Chroma(
    embedding_function=embeddings.embed_documents,
    persist_directory="./chroma-data"
)

# Chunk the text using semantic similarity
def semantic_chunk(text):
    chunks = text_splitter.split_text(text)
    embeddings = embeddings.embed_documents(chunks)
    vectorstore.add_texts(chunks, embeddings)
    return chunks
```

이 예제에서는 LangChainTokenTextSplitter의를 사용하여 입력 텍스트를 50자 겹치는 500자의 청크로 분할합니다. HuggingFaceEmbeddings 모델은 각 청크에 대한 임베딩을 생성하는 데 사용되며 Chroma 벡터 스토어는 청크와 해당 임베딩을 저장하는 데 사용됩니다.

semantic_chunk 함수는 입력 텍스트를 가져와 text_splitter를 사용하여 청크로 분할하고 임베딩 모델을 사용하여 각 청크에 대한 임베딩을 생성한 다음 청크와 임베딩을에 추가합니다vectorstore.

이 접근 방식을 사용하면 입력 텍스트를 의미 있는 청크로 분할할 수 있으며, 이는 RAG 솔루션에서 질문 답변 또는 요약과 같은 다양한 작업에 유용할 수 있습니다.

이는 기본 예제이며 특정 사용 사례 및 데이터를 기반으로 파라미터(예: chunk_size, chunk_overlap) 또는 텍스트 분할기 및 임베딩 모델 선택을 조정해야 할 수 있습니다.

RAG 솔루션에 대한 Amazon Q 채팅 종료

그러나이 접근 방식은 슬라이딩 윈도우 기술을 사용하여 문서를 청크하는 기본적인 방법만 제공합니다. 개발자는 임베딩 측정에 의존하는 고급 방법을 원합니다. 이러한 시나리오에서 개발자는 목표를 고려하여 프롬프트에 세부 정보를 추가해야 합니다. 개발자가 LangChain을 사용하여 알려진 AI21 텍스트 분할자 함수를 참조하면 다음 채팅과 같이 향상된 응답을 받습니다.

RAG 솔루션에 대한 채팅(개선된 프롬프트 사용)

사용자:

위의 프롬프트는 청크당 토큰 크기와 관련된 분할에만 관련이 있습니다. AI21 의미 체계 텍스트 분할자를 사용하고 싶습니다. langchain을 사용하여 A121 의미 체계 텍스트 분할을 사용하여 함수를 생성합니다.

Amazon Q:

LangChain과 함께 AI21 의미 체계 텍스트 분할기를 사용하여 의미 체계 청킹을 구현해 보겠습니다. 예는 다음과 같습니다. [1]

```
from langchain.text_splitter import AI21SemanticTextSplitter
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.vectorstores import Chroma

# Initialize the text splitter
text_splitter = AI21SemanticTextSplitter()

# Initialize the embeddings model
embeddings = HuggingFaceEmbeddings()

# Create the vector store
vectorstore = Chroma(
    embedding_function=embeddings.embed_documents,
    persist_directory="./chroma-data"
)

# Chunk the text using semantic similarity
def semantic_chunk(text):
    chunks = text_splitter.split_text(text)
    embeddings = embeddings.embed_documents(chunks)
    vectorstore.add_texts(chunks, embeddings)
    return chunks
```

이 예제에서는 LangChain의 AI21SemanticTextSplitter를 사용하여 입력 텍스트를 의미 있는 청크로 분할합니다. HuggingFaceEmbeddings 모델은 각 청크에 대한 임베딩을 생성하는 데 사용되며 Chroma 벡터 스토어는 청크와 해당 임베딩을 저장하는 데 사용됩니다.

semantic_chunk 함수는 입력 텍스트를 가져와 text_splitter를 사용하여 청크로 분할하고 임베딩 모델을 사용하여 각 청크에 대한 임베딩을 생성한 다음 청크와 해당 임베딩을에 추가합니다vectorstore.

이 접근 방식을 사용하면 입력 텍스트를 의미 있는 청크로 분할할 수 있으며, 이는 RAG 솔루션에서 질문 답변 또는 요약과 같은 다양한 작업에 유용할 수 있습니다.

특정 사용 사례 및 데이터에 따라 파라미터 또는 임베딩 모델 선택을 조정해야 할 수 있습니다.

소스 [1] LangChain에서 텍스트 분할자를 사용하는 방법 알아보기

soosweb3.hashnode.dev > the-ultimate-langchain-series-text-splitters

RAG 솔루션에 대한 Amazon Q 채팅 종료(개선된 프롬프트 사용)

채팅 결과에 따라 개발자는 함수를 사용하고 정보 소스로 이동할 수 있습니다.

일반적으로 고급 주제 및 질문은 Amazon Q Developer와 채팅할 때 프롬프트 내에서 더 구체적인 컨텍스트가 필요합니다. 채팅 결과가 정확하지 않다고 생각되면 축소 아이콘을 사용하여 Amazon Q 응답에 대한 피드백을 제공합니다. Amazon Q Developer는 피드백을 지속적으로 사용하여 향후 릴리스를 개선합니다. 긍정적인 결과를 가져온 상호 작용의 경우 썸업 아이콘을 사용하여 피드백을 제공하는 것이 유용합니다.

FAQs Amazon Q 개발자 정보

이 섹션에서는 코드 개발을 위한 Amazon Q 개발자 사용에 대해 자주 묻는 질문에 대한 답변을 제공합니다.

Amazon Q Developer란 무엇인가요?

Amazon Q Developer는 지능형 코드 생성 및 권장 사항을 제공하여 코드 개발 작업을 가속화하도록 설계된 강력한 생성형 AI 기반 서비스입니다. 2024년 4월 30일, Amazon은 Amazon Q Developer의 일부가 CodeWhisperer 되었습니다.

Amazon Q Developer에 액세스하려면 어떻게 해야 하나요?

Amazon Q Developer는 AWS Toolkits for Visual Studio Code 및 JetBrains IDEs, 예: IntelliJ 및 PyCharm. 시작하려면 최신 AWS Toolkit 버전 를 설치합니다.

Amazon Q Developer는 어떤 프로그래밍 언어를 지원하나요?

Visual Studio 코드 및 JetBrains IDEs, Amazon Q Developer에서 지원 Python, Java, JavaScript, TypeScript, C#,Go, Rust, PHP, Ruby, Kotlin, C, C++, Shell 스크립팅, SQL및 Scala. 이 가이드는 Python 그리고 Java 예를 들어, 개념은 지원되는 프로그래밍 언어에 적용됩니다.

더 나은 코드 생성을 위해 Amazon Q Developer에 컨텍스트를 제공하려면 어떻게 해야 하나요?

기존 코드부터 시작하거나, 관련 라이브러리를 가져오거나, 클래스 및 함수를 생성하거나, 코드 스켈레톤을 설정합니다. 자연어 프롬프트에는 표준 주석 블록을 사용합니다. 스크립트를 특정 목표에 집중하고, 고유한 기능을 관련 컨텍스트가 있는 별도의 스크립트로 모듈화합니다. 자세한 내용은 Amazon Q Developer 의 모범 코딩 사례를 참조하세요.

Amazon Q Developer를 사용한 인라인 코드 생성이 정확하지 않은 경우 어떻게 해야 하나요?

스크립트의 컨텍스트를 검토하고, 라이브러리가 있는지 확인하고, 클래스 및 함수가 새 코드와 관련이 있는지 확인합니다. 코드를 모듈화하고 목표에 따라 다양한 클래스와 함수를 구분합니다. 명확하고 구

체적인 프롬프트 또는 설명을 작성합니다. 코드의 정확성이 여전히 확실하지 않고 계속 진행할 수 없는 경우 Amazon Q와 채팅을 시작하고 지침과 함께 코드 조각을 전송합니다. 자세한 내용은 Amazon Q 개발자의 코드 생성 시나리오 문제 해결을 참조하세요.

코드 생성 및 문제 해결을 위해 Amazon Q Developer 채팅 기능을 사용하려면 어떻게 해야 하나요?

Amazon Q와 채팅하여 공통 함수를 생성하거나, 추천을 요청하거나, 코드를 설명합니다. 초기 응답이 만족스럽지 않은 경우 다른 프롬프트로 실험하고 제공된 를 따릅니다URLs. 또한 Amazon Q에 피드백을 제공하여 향후 채팅 성능을 개선합니다. 썸업 및 썸다운 아이콘을 사용하여 피드백을 제공합니다. 자세한 내용은 채팅 예제를 참조하세요.

Amazon Q Developer를 사용하는 모범 사례는 무엇입니까?

관련 컨텍스트를 제공하고, 프롬프트를 실험하고, 반복하고, 수락하기 전에 코드 제안을 검토하고, 사용자 지정 기능을 사용하고, 데이터 프라이버시 및 콘텐츠 사용 정책을 이해합니다. 자세한 내용은 Amazon Q 개발자를 사용한 코드 생성 모범 사례 및 Amazon Q 개발자를 사용한 코드 권장 사항 모범 사례를 참조하세요.

자체 코드를 기반으로 추천을 생성하도록 Amazon Q Developer를 사용자 지정할 수 있나요?

예, Amazon Q Developer의 고급 기능인 사용자 지정을 사용합니다. 사용자 지정을 통해 기업은 자체 코드 리포지토리를 제공하여 Amazon Q Developer가 인라인 코드 제안을 추천할 수 있습니다. 자세한 내용은 Amazon Q 개발자 및 리소스의 고급 기능을 참조하세요.

Amazon Q Developer 사용의 다음 단계

이 포괄적인 가이드에서 얻은 지식을 바탕으로 코딩 워크플로에서 Amazon Q Developer를 효과적으로 사용할 수 있습니다. 원하는 IDE(Visual Studio Code 또는 JetBrains) AWS Toolkit 에를 설치하고 Amazon Q Developer의 생성형 AI 기반 코드 생성 및 권장 사항을 살펴보세요.

Amazon Q Developer의 잠재력을 최대한 활용하는 가장 효과적인 방법은 코드를 직접 사용해 보는 것입니다. Amazon Q를 개발 수명 주기에 통합할 때 모범 사례, 문제 해결 및 실제 예제는이 가이드를 참조하세요.

또한 리소스에서 참조되는 AWS 블로그 및 개발자 안내서를 검토하여 최신 정보를 확인하세요. 이러한 리소스는 Amazon Q Developer 사용을 최적화하는 데 도움이 되는 최신 업데이트, 모범 사례 및 인사이트를 제공합니다.

이 가이드를 개선하고 개발자를 위한 귀중한 리소스를 유지하는 데 도움이 되는 피드백은 매우 중요합니다. 향후 버전에 대한 경험, 과제 및 제안을 공유합니다. 입력은 추가 예제, 문제 해결 시나리오 및 필요에 맞는 인사이트로 가이드를 개선하는 데 도움이 됩니다.

리소스

AWS 블로그

- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer Agent를 사용한 소프트웨어 개발 재구상
- Amazon Q의 문제 해결 예 5개
- 프라이빗 코드 기반으로 IDE에서 Amazon Q 개발자 사용자 지정
- 코드 변환을 위한 Amazon Q Developer 에이전트가 Java 업그레이드를 가속화하는 세 가지 방법
- 효율적인 코드 디버깅 및 유지 관리를 위한 Amazon Q 개발자 활용
- Amazon Q Developer로 애플리케이션 테스트

AWS 설명서

- Amazon Q 개발자 사용 설명서
- Amazon Q 개발자 코드 사용자 지정
- Amazon Q 개발자 코드 변환

AWS 워크숍

- Amazon Q 개발자 몰입의 날
- Amazon Q 개발자 워크숍 - Q-Words 앱 구축
- Amazon Q 개발자 워크숍 - 효과적인 프롬프트 생성

기여자

다음 개인이이 가이드에 기여했습니다.

- JoeKing, Senior Data Scientist, AWS
- Prateek Gupta, 팀 리드 – 선임 CAA, AWS
- Manohar Reddy Arranagu, DevOps Architect, AWS
- Soumik Roy, 클라우드 애플리케이션 아키텍트, AWS
- Sanket Shinde, 컨설턴트, AWS

문서 이력

아래 표에 이 가이드의 주요 변경 사항이 설명되어 있습니다. 향후 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하세요.

변경 사항	설명	날짜
최초 게시	—	2024년 8월 16일

AWS 권장 가이드 용어집

다음은 AWS 권장 가이드에서 제공하는 전략, 가이드 및 패턴에서 일반적으로 사용되는 용어입니다. 용어집 항목을 제안하려면 용어집 끝에 있는 피드백 제공 링크를 사용하십시오.

숫자

7가지 전략

애플리케이션을 클라우드로 이전하기 위한 7가지 일반적인 마이그레이션 전략 이러한 전략은 Gartner가 2011년에 파악한 5가지 전략을 기반으로 하며 다음으로 구성됩니다.

- 리팩터링/리아키텍트 - 클라우드 네이티브 기능을 최대한 활용하여 애플리케이션을 이동하고 해당 아키텍처를 수정함으로써 민첩성, 성능 및 확장성을 개선합니다. 여기에는 일반적으로 운영 체제와 데이터베이스 이식이 포함됩니다. 예: 온프레미스 Oracle 데이터베이스를 Amazon Aurora PostgreSQL 호환 버전으로 마이그레이션합니다.
- 리플랫포밍(리프트 앤드 리셰이프) - 애플리케이션을 클라우드로 이동하고 일정 수준의 최적화를 도입하여 클라우드 기능을 활용합니다. 예:에서 온프레미스 Oracle 데이터베이스를 Oracle용 Amazon Relational Database Service(RDS)로 마이그레이션합니다 AWS 클라우드.
- 재구매(드롭 앤드 숍) - 일반적으로 기존 라이선스에서 SaaS 모델로 전환하여 다른 제품으로 전환합니다. 예: 고객 관계 관리(CRM) 시스템을 Salesforce.com 마이그레이션합니다.
- 리호스팅(리프트 앤드 시프트) - 애플리케이션을 변경하지 않고 클라우드로 이동하여 클라우드 기능을 활용합니다. 예:의 EC2 인스턴스에서 온프레미스 Oracle 데이터베이스를 Oracle로 마이그레이션합니다 AWS 클라우드.
- 재배치(하이퍼바이저 수준의 리프트 앤 시프트) - 새 하드웨어를 구매하거나, 애플리케이션을 다시 작성하거나, 기존 운영을 수정하지 않고도 인프라를 클라우드로 이동합니다. 온프레미스 플랫폼에서 동일한 플랫폼의 클라우드 서비스로 서버를 마이그레이션합니다. 예: Microsoft Hyper-V 애플리케이션을 로 마이그레이션합니다 AWS.
- 유지(보관) - 소스 환경에 애플리케이션을 유지합니다. 대규모 리팩터링이 필요하고 해당 작업을 나중으로 연기하려는 애플리케이션과 비즈니스 차원에서 마이그레이션할 이유가 없어 유지하려는 레거시 애플리케이션이 여기에 포함될 수 있습니다.
- 사용 중지 - 소스 환경에서 더 이상 필요하지 않은 애플리케이션을 폐기하거나 제거합니다.

A

ABAC

속성 기반 액세스 제어를 참조하세요.

추상화된 서비스

관리형 서비스를 참조하세요.

ACID

원자성, 일관성, 격리, 내구성을 참조하세요.

능동-능동 마이그레이션

양방향 복제 도구 또는 이중 쓰기 작업을 사용하여 소스 데이터베이스와 대상 데이터베이스가 동기화된 상태로 유지되고, 두 데이터베이스 모두 마이그레이션 중 연결 애플리케이션의 트랜잭션을 처리하는 데이터베이스 마이그레이션 방법입니다. 이 방법은 일회성 전환이 필요한 대신 소규모의 제어된 배치로 마이그레이션을 지원합니다. 더 유연하지만 액티브-패시브 마이그레이션보다 더 많은 작업이 필요합니다.

능동-수동 마이그레이션

소스 데이터베이스와 대상 데이터베이스가 동기화된 상태로 유지되지만 데이터가 대상 데이터베이스에 복제되는 동안 소스 데이터베이스만 애플리케이션 연결의 트랜잭션을 처리하는 데이터베이스 마이그레이션 방법입니다. 대상 데이터베이스는 마이그레이션 중 어떤 트랜잭션도 허용하지 않습니다.

집계 함수

행 그룹에서 작동하고 그룹에 대한 단일 반환 값을 계산하는 SQL 함수입니다. 집계 함수의 예로는 `SUM` 및가 있습니다`MAX`.

AI

인공 지능을 참조하세요.

AIOps

인공 지능 작업을 참조하세요.

익명화

데이터세트에서 개인 정보를 영구적으로 삭제하는 프로세스입니다. 익명화는 개인 정보 보호에 도움이 될 수 있습니다. 익명화된 데이터는 더 이상 개인 데이터로 간주되지 않습니다.

안티 패턴

솔루션이 다른 솔루션보다 비생산적이거나 비효율적이거나 덜 효과적이어서 반복되는 문제에 자주 사용되는 솔루션입니다.

애플리케이션 제어

맬웨어로부터 시스템을 보호하기 위해 승인된 애플리케이션만 사용할 수 있는 보안 접근 방식입니다.

애플리케이션 포트폴리오

애플리케이션 구축 및 유지 관리 비용과 애플리케이션의 비즈니스 가치를 비롯하여 조직에서 사용하는 각 애플리케이션에 대한 세부 정보 모음입니다. 이 정보는 포트폴리오 검색 및 분석 프로세스의 핵심이며 마이그레이션, 현대화 및 최적화할 애플리케이션을 식별하고 우선순위를 정하는 데 도움이 됩니다.

인공 지능

컴퓨터 기술을 사용하여 학습, 문제 해결, 패턴 인식 등 일반적으로 인간과 관련된 인지 기능을 수행하는 것을 전문으로 하는 컴퓨터 과학 분야입니다. 자세한 내용은 What is Artificial Intelligence?를 참조하십시오.

인공 지능 운영(AIOps)

기계 학습 기법을 사용하여 운영 문제를 해결하고, 운영 인시던트 및 사용자 개입을 줄이고, 서비스 품질을 높이는 프로세스입니다. AWS 마이그레이션 전략에서 AIOps가 사용되는 방법에 대한 자세한 내용은 운영 통합 가이드를 참조하십시오.

비대칭 암호화

한 쌍의 키, 즉 암호화를 위한 퍼블릭 키와 복호화를 위한 프라이빗 키를 사용하는 암호화 알고리즘입니다. 퍼블릭 키는 복호화에 사용되지 않으므로 공유할 수 있지만 프라이빗 키에 대한 액세스는 엄격히 제한되어야 합니다.

원자성, 일관성, 격리성, 내구성(ACID)

오류, 정전 또는 기타 문제가 발생한 경우에도 데이터베이스의 데이터 유효성과 운영 신뢰성을 보장하는 소프트웨어 속성 세트입니다.

ABAC(속성 기반 액세스 제어)

부서, 직무, 팀 이름 등의 사용자 속성을 기반으로 세분화된 권한을 생성하는 방식입니다. 자세한 내용은 AWS Identity and Access Management (IAM) 설명서의 용 ABAC AWS를 참조하세요.

신뢰할 수 있는 데이터 소스

가장 신뢰할 수 있는 정보 소스로 간주되는 기본 버전의 데이터를 저장하는 위치입니다. 익명화, 편집 또는 가명화와 같은 데이터 처리 또는 수정의 목적으로 신뢰할 수 있는 데이터 소스의 데이터를 다른 위치로 복사할 수 있습니다.

가용 영역

다른 가용 영역의 장애로부터 격리 AWS 리전 되고 동일한 리전의 다른 가용 영역에 저렴하고 지연 시간이 짧은 네트워크 연결을 제공하는 내의 고유한 위치입니다.

AWS 클라우드 채택 프레임워크(AWS CAF)

조직이 클라우드로 성공적으로 전환 AWS 하기 위한 효율적이고 효과적인 계획을 개발하는 데 도움이 되는의 지침 및 모범 사례 프레임워크입니다. AWS CAF는 지침을 비즈니스, 사람, 거버넌스, 플랫폼, 보안 및 운영이라는 6가지 중점 영역으로 구성합니다. 비즈니스, 사람 및 거버넌스 관점은 비즈니스 기술과 프로세스에 초점을 맞추고, 플랫폼, 보안 및 운영 관점은 전문 기술과 프로세스에 중점을 둡니다. 예를 들어, 사람 관점은 인사(HR), 직원 배치 기능 및 인력 관리를 담당하는 이해관계자를 대상으로 합니다. 이러한 관점에서 AWS CAF는 성공적인 클라우드 채택을 위해 조직을 준비하는 데 도움이 되는 인력 개발, 교육 및 커뮤니케이션에 대한 지침을 제공합니다. 자세한 내용은 AWS CAF 웹 사이트와 AWS CAF 백서를 참조하십시오.

AWS 워크로드 검증 프레임워크(AWS WQF)

데이터베이스 마이그레이션 워크로드를 평가하고, 마이그레이션 전략을 권장하고, 작업 견적을 제공하는 도구입니다. AWS WQF는 AWS Schema Conversion Tool (AWS SCT)에 포함되어 있습니다. 데이터베이스 스키마 및 코드 객체, 애플리케이션 코드, 종속성 및 성능 특성을 분석하고 평가 보고서를 제공합니다.

B

잘못된 봇

개인 또는 조직을 방해하거나 해를 입히기 위한 봇입니다.

BCP

비즈니스 연속성 계획을 참조하세요.

동작 그래프

리소스 동작과 시간 경과에 따른 상호 작용에 대한 통합된 대화형 뷰입니다. Amazon Detective에서 동작 그래프를 사용하여 실패한 로그온 시도, 의심스러운 API 호출 및 유사한 작업을 검사할 수 있습니다. 자세한 내용은 Detective 설명서의 Data in a behavior graph를 참조하십시오.

빅 엔디안 시스템

가장 중요한 바이트를 먼저 저장하는 시스템입니다. Endianness도 참조하세요.

바이너리 분류

바이너리 결과(가능한 두 클래스 중 하나)를 예측하는 프로세스입니다. 예를 들어, ML 모델이 “이 이메일이 스팸인가요, 스팸이 아닌가요?”, '이 제품은 책인가요, 자동차인가요?' 등의 문제를 예측해야 할 수 있습니다.

블룸 필터

요소가 세트의 멤버인지 여부를 테스트하는 데 사용되는 메모리 효율성이 높은 확률론적 데이터 구조입니다.

블루/그린(Blue/Green) 배포

두 개의 별개이지만 동일한 환경을 생성하는 배포 전략입니다. 현재 애플리케이션 버전은 한 환경(파란색)에서 실행하고 새 애플리케이션 버전은 다른 환경(녹색)에서 실행합니다. 이 전략을 사용하면 영향을 최소화하면서 빠르게 롤백할 수 있습니다.

bot

인터넷을 통해 자동화된 작업을 실행하고 인적 활동 또는 상호 작용을 시뮬레이션하는 소프트웨어 애플리케이션입니다. 인터넷에서 정보를 인덱싱하는 웹 크롤러와 같은 일부 봇은 유용하거나 유용합니다. 잘못된 봇이라고 하는 일부 다른 봇은 개인 또는 조직을 방해하거나 해를 입히기 위한 것입니다.

봇넷

맬웨어에 감염되고 봇 셰이더 또는 봇 운영자라고 하는 단일 당사자에 의해 제어되는 봇 네트워크입니다. Botnet은 봇과 봇의 영향을 확장하는 가장 잘 알려진 메커니즘입니다.

브랜치

코드 리포지토리의 포함된 영역입니다. 리포지토리에 생성되는 첫 번째 브랜치가 기본 브랜치입니다. 기존 브랜치에서 새 브랜치를 생성한 다음 새 브랜치에서 기능을 개발하거나 버그를 수정할 수 있습니다. 기능을 구축하기 위해 생성하는 브랜치를 일반적으로 기능 브랜치라고 합니다. 기능을 출시할 준비가 되면 기능 브랜치를 기본 브랜치에 다시 병합합니다. 자세한 내용은 About branches(GitHub 설명서)를 참조하십시오.

브레이크 글래스 액세스

예외적인 상황에서 승인된 프로세스를 통해 사용자가 일반적으로 액세스할 권한이 없는에 액세스할 수 AWS 계정 있는 빠른 방법입니다. 자세한 내용은 Well-Architected 지침의 깨진 절차 구현 표시기를 AWS 참조하세요.

브라운필드 전략

사용자 환경의 기존 인프라 시스템 아키텍처에 브라운필드 전략을 채택할 때는 현재 시스템 및 인프라의 제약 조건을 중심으로 아키텍처를 설계합니다. 기존 인프라를 확장하는 경우 브라운필드 전략과 그린필드 전략을 혼합할 수 있습니다.

버퍼 캐시

가장 자주 액세스하는 데이터가 저장되는 메모리 영역입니다.

사업 역량

기업이 가치를 창출하기 위해 하는 일(예: 영업, 고객 서비스 또는 마케팅)입니다. 마이크로서비스 아키텍처 및 개발 결정은 비즈니스 역량에 따라 이루어질 수 있습니다. 자세한 내용은 백서의 AWS에서 컨테이너화된 마이크로서비스 실행의 비즈니스 역량 중심의 구성화 섹션을 참조하십시오.

비즈니스 연속성 계획(BCP)

대규모 마이그레이션과 같은 중단 이벤트가 운영에 미치는 잠재적 영향을 해결하고 비즈니스가 신속하게 운영을 재개할 수 있도록 지원하는 계획입니다.

C

CAF

AWS 클라우드 채택 프레임워크를 참조하세요.

canary 배포

최종 사용자에게 버전의 느린 증분 릴리스입니다. 확신이 드는 경우 새 버전을 배포하고 현재 버전을 완전히 교체합니다.

CCoE

Cloud Center of Excellence를 참조하세요.

CDC

변경 데이터 캡처를 참조하세요.

변경 데이터 캡처(CDC)

데이터베이스 테이블과 같은 데이터 소스의 변경 내용을 추적하고 변경 사항에 대한 메타데이터를 기록하는 프로세스입니다. 대상 시스템의 변경 내용을 감사하거나 복제하여 동기화를 유지하는 등의 다양한 용도로 CDC를 사용할 수 있습니다.

카오스 엔지니어링

시스템의 복원력을 테스트하기 위해 의도적으로 장애 또는 중단 이벤트를 도입합니다. AWS Fault Injection Service (AWS FIS)를 사용하여 AWS 워크로드에 스트레스를 주고 응답을 평가하는 실험을 수행할 수 있습니다.

CI/CD

지속적 통합 및 지속적 전달을 참조하세요.

분류

예측을 생성하는 데 도움이 되는 분류 프로세스입니다. 분류 문제에 대한 ML 모델은 이산 값을 예측합니다. 이산 값은 항상 서로 다릅니다. 예를 들어, 모델이 이미지에 자동차가 있는지 여부를 평가해야 할 수 있습니다.

클라이언트측 암호화

대상이 데이터를 AWS 서비스 수신하기 전에 로컬에서 데이터를 암호화합니다.

클라우드 혁신 센터(CCoE)

클라우드 모범 사례 개발, 리소스 동원, 마이그레이션 타임라인 설정, 대규모 혁신을 통한 조직 선도 등 조직 전체에서 클라우드 채택 노력을 추진하는 다분야 팀입니다. 자세한 내용은 AWS 클라우드 엔터프라이즈 전략 블로그의 CCoE 게시물을 참조하세요.

클라우드 컴퓨팅

원격 데이터 스토리지와 IoT 디바이스 관리에 일반적으로 사용되는 클라우드 기술 클라우드 컴퓨팅은 일반적으로 엣지 컴퓨팅 기술과 연결됩니다.

클라우드 운영 모델

IT 조직에서 하나 이상의 클라우드 환경을 구축, 성숙화 및 최적화하는 데 사용되는 운영 모델입니다. 자세한 내용은 클라우드 운영 모델 구축을 참조하십시오.

클라우드 채택 단계

조직이 로 마이그레이션할 때 일반적으로 거치는 4단계: AWS 클라우드

- 프로젝트 - 개념 증명 및 학습 목적으로 몇 가지 클라우드 관련 프로젝트 실행
- 기반 – 클라우드 채택 확장을 위한 기초 투자(예: 랜딩 존 생성, CCoE 정의, 운영 모델 구축)
- 마이그레이션 - 개별 애플리케이션 마이그레이션
- Re-invention - 제품 및 서비스 최적화와 클라우드 혁신

이러한 단계는 Stephen Orban이 블로그 게시물 The Journey Toward Cloud-First and the Stages of Adoption on the AWS 클라우드 Enterprise Strategy 블로그에서 정의했습니다. AWS 마이그레이션 전략과 어떤 관련이 있는지에 대한 자세한 내용은 마이그레이션 준비 가이드를 참조하세요.

CMDB

구성 관리 데이터베이스를 참조하세요.

코드 리포지토리

소스 코드와 설명서, 샘플, 스크립트 등의 기타 자산이 버전 관리 프로세스를 통해 저장되고 업데이트되는 위치입니다. 일반적인 클라우드 리포지토리에는 GitHub 또는가 포함됩니다Bitbucket Cloud. 코드의 각 버전을 브랜치라고 합니다. 마이크로서비스 구조에서 각 리포지토리는 단일 기능 전용입니다. 단일 CI/CD 파이프라인은 여러 리포지토리를 사용할 수 있습니다.

콜드 캐시

비어 있거나, 제대로 채워지지 않았거나, 오래되었거나 관련 없는 데이터를 포함하는 버퍼 캐시입니다. 주 메모리나 디스크에서 데이터베이스 인스턴스를 읽어야 하기 때문에 성능에 영향을 미치며, 이는 버퍼 캐시에서 읽는 것보다 느립니다.

콜드 데이터

거의 액세스되지 않고 일반적으로 과거 데이터인 데이터. 이런 종류의 데이터를 쿼리할 때는 일반적으로 느린 쿼리가 허용됩니다. 이 데이터를 성능이 낮고 비용이 저렴한 스토리지 계층 또는 클래스로 옮기면 비용을 절감할 수 있습니다.

컴퓨터 비전(CV)

기계 학습을 사용하여 디지털 이미지 및 비디오와 같은 시각적 형식에서 정보를 분석하고 추출하는 AI 필드입니다. 예를 들어 Amazon SageMaker AI는 CV에 대한 이미지 처리 알고리즘을 제공합니다.

구성 드리프트

워크로드의 경우 구성이 예상 상태에서 변경됩니다. 이로 인해 워크로드가 규정을 준수하지 않을 수 있으며 일반적으로 점진적이고 의도하지 않은 것입니다.

구성 관리 데이터베이스(CMDB)

하드웨어 및 소프트웨어 구성 요소와 해당 구성을 포함하여 데이터베이스와 해당 IT 환경에 대한 정보를 저장하고 관리하는 리포지토리입니다. 일반적으로 마이그레이션의 포트폴리오 검색 및 분석 단계에서 CMDB의 데이터를 사용합니다.

규정 준수 팩

규정 준수 및 보안 검사를 사용자 지정하기 위해 조합할 수 있는 AWS Config 규칙 및 수정 작업 모음입니다. YAML 템플릿을 사용하여 적합성 팩을 AWS 계정 및 리전 또는 조직 전체에 단일 엔터티로 배포할 수 있습니다. 자세한 내용은 AWS Config 설명서의 적합성 팩을 참조하세요.

지속적 통합 및 지속적 전달(CI/CD)

소프트웨어 릴리스 프로세스의 소스, 빌드, 테스트, 스테이징 및 프로덕션 단계를 자동화하는 프로세스입니다. CI/CD는 일반적으로 파이프라인으로 설명됩니다. CI/CD를 통해 프로세스를 자동화하고, 생산성을 높이고, 코드 품질을 개선하고, 더 빠르게 제공할 수 있습니다. 자세한 내용은 지속적 전달의 이점을 참조하십시오. CD는 지속적 배포를 의미하기도 합니다. 자세한 내용은 지속적 전달(Continuous Delivery)과 지속적인 개발을 참조하십시오.

CV

컴퓨터 비전을 참조하세요.

D

저장 데이터

스토리지에 있는 데이터와 같이 네트워크에 고정되어 있는 데이터입니다.

데이터 분류

중요도와 민감도를 기준으로 네트워크의 데이터를 식별하고 분류하는 프로세스입니다. 이 프로세스는 데이터에 대한 적절한 보호 및 보존 제어를 결정하는 데 도움이 되므로 사이버 보안 위험 관리 전략의 중요한 구성 요소입니다. 데이터 분류는 AWS Well-Architected Framework에서 보안 원칙의 구성 요소입니다. 자세한 내용은 데이터 분류를 참조하십시오.

데이터 드리프트

프로덕션 데이터와 ML 모델 학습에 사용된 데이터 간의 상당한 차이 또는 시간 경과에 따른 입력 데이터의 의미 있는 변화. 데이터 드리프트는 ML 모델 예측의 전반적인 품질, 정확성 및 공정성을 저하시킬 수 있습니다.

전송 중 데이터

네트워크를 통과하고 있는 데이터입니다. 네트워크 리소스 사이를 이동 중인 데이터를 예로 들 수 있습니다.

데이터 메시

중앙 집중식 관리 및 거버넌스를 통해 분산되고 분산된 데이터 소유권을 제공하는 아키텍처 프레임워크입니다.

데이터 최소화

꼭 필요한 데이터만 수집하고 처리하는 원칙입니다. 에서 데이터를 최소화하면 개인 정보 보호 위험, 비용 및 분석 탄소 발자국을 줄일 AWS 클라우드 수 있습니다.

데이터 경계

신뢰할 수 있는 자격 증명만 예상 네트워크에서 신뢰할 수 있는 리소스에 액세스하도록 하는 데 도움이 되는 AWS 환경의 예방 가드레일 세트입니다. 자세한 내용은 데이터 경계 구축을 참조하세요 AWS.

데이터 사전 처리

원시 데이터를 ML 모델이 쉽게 구문 분석할 수 있는 형식으로 변환하는 것입니다. 데이터를 사전 처리한다는 것은 특정 열이나 행을 제거하고 누락된 값, 일관성이 없는 값 또는 중복 값을 처리함을 의미할 수 있습니다.

데이터 출처

라이프사이클 전반에 걸쳐 데이터의 출처와 기록을 추적하는 프로세스(예: 데이터 생성, 전송, 저장 방법).

데이터 주체

데이터를 수집 및 처리하는 개인입니다.

데이터 웨어하우스

분석과 같은 비즈니스 인텔리전스를 지원하는 데이터 관리 시스템입니다. 데이터 웨어하우스에는 일반적으로 많은 양의 기록 데이터가 포함되며 일반적으로 쿼리 및 분석에 사용됩니다.

데이터 정의 언어(DDL)

데이터베이스에서 테이블 및 객체의 구조를 만들거나 수정하기 위한 명령문 또는 명령입니다.

데이터베이스 조작 언어(DML)

데이터베이스에서 정보를 수정(삽입, 업데이트 및 삭제)하기 위한 명령문 또는 명령입니다.

DDL

데이터베이스 정의 언어를 참조하세요.

딥 앙상블

예측을 위해 여러 딥 러닝 모델을 결합하는 것입니다. 딥 앙상블을 사용하여 더 정확한 예측을 얻거나 예측의 불확실성을 추정할 수 있습니다.

딥 러닝

여러 계층의 인공 신경망을 사용하여 입력 데이터와 관심 대상 변수 간의 매핑을 식별하는 ML 하위 분야입니다.

심층 방어

네트워크와 그 안의 데이터 기밀성, 무결성 및 가용성을 보호하기 위해 컴퓨터 네트워크 전체에 일련의 보안 메커니즘과 제어를 신중하게 계층화하는 정보 보안 접근 방식입니다. 이 전략을 채택하면 AWS Organizations 구조의 여러 계층에 여러 컨트롤을 AWS추가하여 리소스를 보호할 수 있습니다. 예를 들어, 심층 방어 접근 방식은 다단계 인증, 네트워크 세분화 및 암호화를 결합할 수 있습니다.

위임된 관리자

에서 AWS Organizations호환되는 서비스는 AWS 멤버 계정을 등록하여 조직의 계정을 관리하고 해당 서비스에 대한 권한을 관리할 수 있습니다. 이러한 계정을 해당 서비스의 위임된 관리자라고 합니다. 자세한 내용과 호환되는 서비스 목록은 AWS Organizations 설명서의 AWS Organizations와 함께 사용할 수 있는 AWS 서비스를 참조하십시오.

배포

대상 환경에서 애플리케이션, 새 기능 또는 코드 수정 사항을 사용할 수 있도록 하는 프로세스입니다. 배포에는 코드 베이스의 변경 사항을 구현한 다음 애플리케이션 환경에서 해당 코드베이스를 구축하고 실행하는 작업이 포함됩니다.

개발 환경

환경을 참조하세요.

탐지 제어

이벤트 발생 후 탐지, 기록 및 알림을 수행하도록 설계된 보안 제어입니다. 이러한 제어는 기존의 예방적 제어를 우회한 보안 이벤트를 알리는 2차 방어선입니다. 자세한 내용은 Implementing security controls on AWS의 Detective controls를 참조하십시오.

개발 가치 흐름 매핑 (DVSM)

소프트웨어 개발 라이프사이클에서 속도와 품질에 부정적인 영향을 미치는 제약 조건을 식별하고 우선 순위를 지정하는 데 사용되는 프로세스입니다. DVSM은 원래 린 제조 방식을 위해 설계된 가치 흐름 매핑 프로세스를 확장합니다. 소프트웨어 개발 프로세스를 통해 가치를 창출하고 이동하는 데 필요한 단계와 팀에 중점을 둡니다.

디지털 트윈

건물, 공장, 산업 장비 또는 생산 라인과 같은 실제 시스템을 가상으로 표현한 것입니다. 디지털 트윈은 예측 유지 보수, 원격 모니터링, 생산 최적화를 지원합니다.

차원 테이블

스타 스키마에서는 팩트 테이블의 정량적 데이터에 대한 데이터 속성을 포함하는 더 작은 테이블입니다. 차원 테이블 속성은 일반적으로 텍스트 필드 또는 텍스트처럼 동작하는 개별 숫자입니다. 이러한 속성은 일반적으로 쿼리 제약, 필터링 및 결과 집합 레이블 지정에 사용됩니다.

재해

워크로드 또는 시스템이 기본 배포 위치에서 비즈니스 목표를 달성하지 못하게 방해하는 이벤트입니다. 이러한 이벤트는 자연재해, 기술적 오류, 의도하지 않은 구성 오류 또는 멀웨어 공격과 같은 사람의 행동으로 인한 결과일 수 있습니다.

재해 복구(DR)

재해로 인한 가동 중지 시간과 데이터 손실을 최소화하는 데 사용하는 전략 및 프로세스입니다. 자세한 내용은 AWS Well-Architected Framework의 Disaster Recovery of Workloads on AWS: Recovery in the Cloud를 참조하세요.

DML

데이터베이스 조작 언어를 참조하세요.

도메인 기반 설계

구성 요소를 각 구성 요소가 제공하는 진화하는 도메인 또는 핵심 비즈니스 목표에 연결하여 복잡한 소프트웨어 시스템을 개발하는 접근 방식입니다. 이 개념은 에릭 에반스에 의해 그의 저서인 도메인 기반 디자인: 소프트웨어 중심의 복잡성 해결(Boston: Addison-Wesley Professional, 2003)에서 소개되었습니다. Strangler Fig 패턴과 함께 도메인 기반 설계를 사용하는 방법에 대한 자세한 내용은 컨테이너 및 Amazon API Gateway를 사용하여 기존의 Microsoft ASP.NET(ASMX) 웹 서비스를 점진적으로 현대화하는 방법을 참조하십시오.

DR

재해 복구를 참조하세요.

드리프트 감지

기준 구성과의 편차 추적. 예를 들어 AWS CloudFormation 를 사용하여 시스템 리소스의 드리프트를 감지하거나를 사용하여 AWS Control Tower 거버넌스 요구 사항 준수에 영향을 미칠 수 있는 랜딩 존의 변경 사항을 감지할 수 있습니다.

DVSM

개발 값 스트림 매핑을 참조하세요.

E

EDA

탐색 데이터 분석을 참조하세요.

EDI

전자 데이터 교환을 참조하세요.

엣지 컴퓨팅

IoT 네트워크의 엣지에서 스마트 디바이스의 컴퓨팅 성능을 개선하는 기술 클라우드 컴퓨팅과 비교할 때 엣지 컴퓨팅은 통신 지연 시간을 줄이고 응답 시간을 개선할 수 있습니다.

전자 데이터 교환(EDI)

조직 간의 비즈니스 문서 자동 교환. 자세한 내용은 전자 데이터 교환이란 무엇입니까?를 참조하세요.

암호화

사람이 읽을 수 있는 일반 텍스트 데이터를 사이퍼텍스트로 변환하는 컴퓨팅 프로세스입니다.

암호화 키

암호화 알고리즘에 의해 생성되는 무작위 비트의 암호화 문자열입니다. 키의 길이는 다양할 수 있으며 각 키는 예측할 수 없고 고유하게 설계되었습니다.

엔디안

컴퓨터 메모리에 바이트가 저장되는 순서입니다. 빅 엔디안 시스템은 가장 중요한 바이트를 먼저 저장합니다. 리틀 엔디안 시스템은 가장 덜 중요한 바이트를 먼저 저장합니다.

엔드포인트

서비스 엔드포인트를 참조하세요.

엔드포인트 서비스

Virtual Private Cloud(VPC)에서 호스팅하여 다른 사용자와 공유할 수 있는 서비스입니다. 를 사용하여 엔드포인트 서비스를 생성하고 다른 AWS 계정 또는 AWS Identity and Access Management (IAM) 보안 주체에 권한을 AWS PrivateLink 부여할 수 있습니다. 이러한 계정 또는 보안 주체는 인터페이스 VPC 엔드포인트를 생성하여 엔드포인트 서비스에 비공개로 연결할 수 있습니다. 자세한 내용은 Amazon Virtual Private Cloud(VPC) 설명서의 엔드포인트 서비스 생성을 참조하십시오.

엔터프라이즈 리소스 계획(ERP)

엔터프라이즈의 주요 비즈니스 프로세스(예: 회계, MES, 프로젝트 관리)를 자동화하고 관리하는 시스템입니다.

봉투 암호화

암호화 키를 다른 암호화 키로 암호화하는 프로세스입니다. 자세한 내용은 AWS Key Management Service (AWS KMS) 설명서의 봉투 암호화를 참조하세요.

환경

실행 중인 애플리케이션의 인스턴스입니다. 다음은 클라우드 컴퓨팅의 일반적인 환경 유형입니다.

- 개발 환경 - 애플리케이션 유지 관리를 담당하는 핵심 팀만 사용할 수 있는 실행 중인 애플리케이션의 인스턴스입니다. 개발 환경은 변경 사항을 상위 환경으로 승격하기 전에 테스트하는 데 사용됩니다. 이러한 유형의 환경을 테스트 환경이라고도 합니다.
- 하위 환경 - 초기 빌드 및 테스트에 사용되는 환경을 비롯한 애플리케이션의 모든 개발 환경입니다.
- 프로덕션 환경 - 최종 사용자가 액세스할 수 있는 실행 중인 애플리케이션의 인스턴스입니다. CI/CD 파이프라인에서 프로덕션 환경이 마지막 배포 환경입니다.
- 상위 환경 - 핵심 개발 팀 이외의 사용자가 액세스할 수 있는 모든 환경입니다. 프로덕션 환경, 프로덕션 이전 환경 및 사용자 수용 테스트를 위한 환경이 여기에 포함될 수 있습니다.

에픽

애자일 방법론에서 작업을 구성하고 우선순위를 정하는 데 도움이 되는 기능적 범주입니다. 에픽은 요구 사항 및 구현 작업에 대한 개괄적인 설명을 제공합니다. 예를 들어, AWS CAF 보안 에픽에는 ID 및 액세스 관리, 탐지 제어, 인프라 보안, 데이터 보호 및 인시던트 대응이 포함됩니다. AWS 마이그레이션 전략의 에픽에 대한 자세한 내용은 프로그램 구현 가이드를 참조하십시오.

ERP

엔터프라이즈 리소스 계획을 참조하세요.

탐색 데이터 분석(EDA)

데이터 세트를 분석하여 주요 특성을 파악하는 프로세스입니다. 데이터를 수집 또는 집계한 다음 초기 조사를 수행하여 패턴을 찾고, 이상을 탐지하고, 가정을 확인합니다. EDA는 요약 통계를 계산하고 데이터 시각화를 생성하여 수행됩니다.

F

팩트 테이블

스타 스키마의 중앙 테이블입니다. 비즈니스 운영에 대한 정량적 데이터를 저장합니다. 일반적으로 팩트 테이블에는 측정값이 포함된 열과 차원 테이블에 대한 외래 키가 포함된 열의 두 가지 유형이 포함됩니다.

빠른 실패

자주 증분 테스트를 사용하여 개발 수명 주기를 줄이는 철학입니다. 애자일 접근 방식의 중요한 부분입니다.

장애 격리 경계

에서 장애의 영향을 제한하고 워크로드의 복원력을 개선하는 데 도움이 되는 가용 영역, AWS 리전 컨트롤 플레인 또는 데이터 플레인과 같은 AWS 클라우드경계입니다. 자세한 내용은 AWS 장애 격리 경계를 참조하세요.

기능 브랜치

브랜치를 참조하세요.

기능

예측에 사용하는 입력 데이터입니다. 예를 들어, 제조 환경에서 기능은 제조 라인에서 주기적으로 캡처되는 이미지일 수 있습니다.

기능 중요도

모델의 예측에 특성이 얼마나 중요한지를 나타냅니다. 이는 일반적으로 SHAP(Shapley Additive Descriptions) 및 통합 그레디언트와 같은 다양한 기법을 통해 계산할 수 있는 수치 점수로 표현됩니다. 자세한 내용은 기계 학습 모델 해석 가능성을 참조하세요 AWS.

기능 변환

추가 소스로 데이터를 보강하거나, 값을 조정하거나, 단일 데이터 필드에서 여러 정보 세트를 추출하는 등 ML 프로세스를 위해 데이터를 최적화하는 것입니다. 이를 통해 ML 모델이 데이터를 활용

할 수 있습니다. 예를 들어, 날짜 '2021-05-27 00:15:37'을 '2021년', '5월', '목', '15일'로 분류하면 학습 알고리즘이 다양한 데이터 구성 요소와 관련된 미묘한 패턴을 학습하는 데 도움이 됩니다.

몇 번의 프롬프트 표시

LLM에 유사한 작업을 수행하도록 요청하기 전에 작업과 원하는 출력을 보여주는 몇 가지 예제를 제공합니다. 이 기법은 컨텍스트 내 학습을 적용하여 모델이 프롬프트에 포함된 예제(샷)에서 학습합니다. 퓨샷 프롬프트는 특정 형식 지정, 추론 또는 도메인 지식이 필요한 작업에 효과적일 수 있습니다. 제로샷 프롬프트도 참조하세요.

FGAC

세분화된 액세스 제어를 참조하세요.

세분화된 액세스 제어(FGAC)

여러 조건을 사용하여 액세스 요청을 허용하거나 거부합니다.

플래시컷 마이그레이션

단계적 접근 방식을 사용하는 대신 변경 데이터 캡처를 통해 연속 데이터 복제를 사용하여 최대한 짧은 시간 내에 데이터를 마이그레이션하는 데이터베이스 마이그레이션 방법입니다. 목표는 가동 중지 시간을 최소화하는 것입니다.

FM

파운데이션 모델을 참조하세요.

파운데이션 모델(FM)

일반화 및 레이블 지정되지 않은 데이터의 대규모 데이터 세트에 대해 훈련된 대규모 딥 러닝 신경망입니다. FMs은 언어 이해, 텍스트 및 이미지 생성, 자연어 대화와 같은 다양한 일반 작업을 수행할 수 있습니다. 자세한 내용은 파운데이션 모델이란 무엇입니까?를 참조하세요.

G

생성형 AI

대량의 데이터에 대해 훈련되었으며 간단한 텍스트 프롬프트를 사용하여 이미지, 비디오, 텍스트 및 오디오와 같은 새로운 콘텐츠 및 아티팩트를 생성할 수 있는 AI 모델의 하위 집합입니다. 자세한 내용은 생성형 AI란 무엇입니까?를 참조하세요.

지리적 차단

지리적 제한을 참조하세요.

지리적 제한(지리적 차단)

Amazon CloudFront에서 특정 국가의 사용자가 콘텐츠 배포에 액세스하지 못하도록 하는 옵션입니다. 허용 목록 또는 차단 목록을 사용하여 승인된 국가와 차단된 국가를 지정할 수 있습니다. 자세한 내용은 CloudFront 설명서의 콘텐츠의 지리적 배포 제한을 참조하십시오.

Gitflow 워크플로

하위 환경과 상위 환경이 소스 코드 리포지토리의 서로 다른 브랜치를 사용하는 방식입니다. Gitflow 워크플로는 레거시로 간주되며 트렁크 기반 워크플로는 현대적이고 선호하는 접근 방식입니다.

골든 이미지

시스템 또는 소프트웨어의 새 인스턴스를 배포하기 위한 템플릿으로 사용되는 시스템 또는 소프트웨어의 스냅샷입니다. 예를 들어 제조업에서는 골든 이미지를 사용하여 여러 디바이스에 소프트웨어를 프로비저닝할 수 있으며 디바이스 제조 작업의 속도, 확장성 및 생산성을 개선하는 데 도움이 됩니다.

브라운필드 전략

새로운 환경에서 기존 인프라의 부재 시스템 아키텍처에 대한 그린필드 전략을 채택할 때 브라운필드라고도 하는 기존 인프라와의 호환성 제한 없이 모든 새로운 기술을 선택할 수 있습니다. 기존 인프라를 확장하는 경우 브라운필드 전략과 그린필드 전략을 혼합할 수 있습니다.

가드레일

조직 단위(OU) 전체에서 리소스, 정책 및 규정 준수를 관리하는 데 도움이 되는 중요 규칙입니다. 예방 가드레일은 규정 준수 표준에 부합하도록 정책을 시행하며, 서비스 제어 정책과 IAM 권한 경계를 사용하여 구현됩니다. 탐지 가드레일은 정책 위반 및 규정 준수 문제를 감지하고 해결을 위한 알림을 생성하며, 이는 AWS Config Amazon GuardDuty AWS Security Hub, , AWS Trusted Advisor Amazon Inspector 및 사용자 지정 AWS Lambda 검사를 사용하여 구현됩니다.

H

HA

고가용성을 참조하세요.

이기종 데이터베이스 마이그레이션

다른 데이터베이스 엔진을 사용하는 대상 데이터베이스로 소스 데이터베이스 마이그레이션(예: Oracle에서 Amazon Aurora로) 이기종 마이그레이션은 일반적으로 리아키텍트 작업의 일부이며 스

키마를 변환하는 것은 복잡한 작업일 수 있습니다.AWS 는 스키마 변환에 도움이 되는 AWS SCT를 제공합니다.

높은 가용성(HA)

문제나 재해 발생 시 개입 없이 지속적으로 운영할 수 있는 워크로드의 능력. HA 시스템은 자동으로 장애 조치되고, 지속적으로 고품질 성능을 제공하고, 성능에 미치는 영향을 최소화하면서 다양한 부하와 장애를 처리하도록 설계되었습니다.

히스토리언 현대화

제조 산업의 요구 사항을 더 잘 충족하도록 운영 기술(OT) 시스템을 현대화하고 업그레이드하는 데 사용되는 접근 방식입니다. 히스토리언은 공장의 다양한 출처에서 데이터를 수집하고 저장하는 데 사용되는 일종의 데이터베이스입니다.

홀드아웃 데이터

기계 학습 모델을 훈련하는 데 사용되는 데이터 세트에서 보류된 레이블이 지정된 기록 데이터의 일부입니다. 홀드아웃 데이터를 사용하여 모델 예측을 홀드아웃 데이터와 비교하여 모델 성능을 평가할 수 있습니다.

동종 데이터베이스 마이그레이션

동일한 데이터베이스 엔진을 공유하는 대상 데이터베이스로 소스 데이터베이스 마이그레이션(예: Microsoft SQL Server에서 Amazon RDS for SQL Server로) 동종 마이그레이션은 일반적으로 리호스팅 또는 리플랫포밍 작업의 일부입니다. 네이티브 데이터베이스 유틸리티를 사용하여 스키마를 마이그레이션할 수 있습니다.

핫 데이터

자주 액세스하는 데이터(예: 실시간 데이터 또는 최근 번역 데이터). 일반적으로 이 데이터에는 빠른 쿼리 응답을 제공하기 위한 고성능 스토리지 계층 또는 클래스가 필요합니다.

핫픽스

프로덕션 환경의 중요한 문제를 해결하기 위한 긴급 수정입니다. 핫픽스는 긴급하기 때문에 일반적인 DevOps 릴리스 워크플로 외부에서 실행됩니다.

하이퍼케어 기간

전환 직후 마이그레이션 팀이 문제를 해결하기 위해 클라우드에서 마이그레이션된 애플리케이션을 관리하고 모니터링하는 기간입니다. 일반적으로 이 기간은 1~4일입니다. 하이퍼케어 기간이 끝나면 마이그레이션 팀은 일반적으로 애플리케이션에 대한 책임을 클라우드 운영 팀에 넘깁니다.

정보

IaC

코드형 인프라를 참조하세요.

자격 증명 기반 정책

AWS 클라우드 환경 내에서 권한을 정의하는 하나 이상의 IAM 보안 주체에 연결된 정책입니다.

유휴 애플리케이션

90일 동안 평균 CPU 및 메모리 사용량이 5~20%인 애플리케이션입니다. 마이그레이션 프로젝트에서는 이러한 애플리케이션을 사용 중지하거나 온프레미스에 유지하는 것이 일반적입니다.

IIoT

산업용 사물 인터넷을 참조하십시오.

변경 불가능한 인프라

기존 인프라를 업데이트, 패치 또는 수정하는 대신 프로덕션 워크로드를 위한 새 인프라를 배포하는 모델입니다. 변경 불가능한 인프라는 변경 가능한 인프라보다 본질적으로 더 일관되고 안정적이며 예측 가능합니다. 자세한 내용은 AWS Well-Architected Framework의 변경 불가능한 인프라를 사용한 배포 모범 사례를 참조하세요.

인바운드(수신) VPC

AWS 다중 계정 아키텍처에서 애플리케이션 외부에서 네트워크 연결을 수락, 검사 및 라우팅하는 VPC입니다. AWS Security Reference Architecture에서는 애플리케이션과 더 넓은 인터넷 간의 양방향 인터페이스를 보호하기 위해 인바운드, 아웃바운드 및 검사 VPC로 네트워크 계정을 설정할 것을 권장합니다.

증분 마이그레이션

한 번에 전체 전환을 수행하는 대신 애플리케이션을 조금씩 마이그레이션하는 전환 전략입니다. 예를 들어, 처음에는 소수의 마이크로서비스나 사용자만 새 시스템으로 이동할 수 있습니다. 모든 것이 제대로 작동하는지 확인한 후에는 레거시 시스템을 폐기할 수 있을 때까지 추가 마이크로서비스 또는 사용자를 점진적으로 이동할 수 있습니다. 이 전략을 사용하면 대규모 마이그레이션과 관련된 위험을 줄일 수 있습니다.

Industry 4.0

연결성, 실시간 데이터, 자동화, 분석 및 AI/ML의 발전을 통한 제조 프로세스의 현대화를 참조하기 위해 2016년에 Klaus Schwab에서 도입한 용어입니다.

인프라

애플리케이션의 환경 내에 포함된 모든 리소스와 자산입니다.

코드형 인프라(IaC)

구성 파일 세트를 통해 애플리케이션의 인프라를 프로비저닝하고 관리하는 프로세스입니다. IaC는 새로운 환경의 반복 가능성, 신뢰성 및 일관성을 위해 인프라 관리를 중앙 집중화하고, 리소스를 표준화하고, 빠르게 확장할 수 있도록 설계되었습니다.

산업용 사물 인터넷(IIoT)

제조, 에너지, 자동차, 의료, 생명과학, 농업 등의 산업 부문에서 인터넷에 연결된 센서 및 디바이스의 사용 자세한 내용은 산업용 사물 인터넷(IoT) 디지털 트랜스포메이션 전략 구축을 참조하십시오.

검사 VPC

AWS 다중 계정 아키텍처에서는 VPC(동일하거나 다른 AWS 리전), 인터넷 및 온프레미스 네트워크 간의 네트워크 트래픽 검사를 관리하는 중앙 집중식 VPCs입니다. AWS Security Reference Architecture에서는 애플리케이션과 더 넓은 인터넷 간의 양방향 인터페이스를 보호하기 위해 인바운드, 아웃바운드 및 검사 VPC로 네트워크 계정을 설정할 것을 권장합니다.

사물 인터넷(IoT)

인터넷이나 로컬 통신 네트워크를 통해 다른 디바이스 및 시스템과 통신하는 센서 또는 프로세서가 내장된 연결된 물리적 객체의 네트워크 자세한 내용은 IoT란?을 참조하십시오.

해석력

모델의 예측이 입력에 따라 어떻게 달라지는지를 사람이 이해할 수 있는 정도를 설명하는 기계 학습 모델의 특성입니다. 자세한 내용은 기계 학습 모델 해석 가능성을 참조하세요 AWS.

IoT

사물 인터넷을 참조하세요.

IT 정보 라이브러리(TIL)

IT 서비스를 제공하고 이러한 서비스를 비즈니스 요구 사항에 맞게 조정하기 위한 일련의 모범 사례 ITIL은 ITSM의 기반을 제공합니다.

IT 서비스 관리(TSM)

조직의 IT 서비스 설계, 구현, 관리 및 지원과 관련된 활동 클라우드 운영을 ITSM 도구와 통합하는 방법에 대한 자세한 내용은 운영 통합 가이드를 참조하십시오.

ITIL

IT 정보 라이브러리를 참조하세요.

ITSM

IT 서비스 관리를 참조하세요.

L

레이블 기반 액세스 제어(LBAC)

사용자 및 데이터 자체에 각각 보안 레이블 값을 명시적으로 할당하는 필수 액세스 제어(MAC)를 구현한 것입니다. 사용자 보안 레이블과 데이터 보안 레이블 간의 교차 부분에 따라 사용자가 볼 수 있는 행과 열이 결정됩니다.

랜딩 존

랜딩 존은 확장 가능하고 안전한 잘 설계된 다중 계정 AWS 환경입니다. 조직은 여기에서부터 보안 및 인프라 환경에 대한 확신을 가지고 워크로드와 애플리케이션을 신속하게 시작하고 배포할 수 있습니다. 랜딩 존에 대한 자세한 내용은 안전하고 확장 가능한 다중 계정 AWS 환경 설정을 참조하십시오.

대규모 언어 모델(LLM)

방대한 양의 데이터를 기반으로 사전 훈련된 딥 러닝 AI 모델입니다. LLM은 질문 답변, 문서 요약, 텍스트를 다른 언어로 번역, 문장 완성과 같은 여러 작업을 수행할 수 있습니다. 자세한 내용은 LLMs 참조하십시오.

대규모 마이그레이션

300대 이상의 서버 마이그레이션입니다.

LBAC

레이블 기반 액세스 제어를 참조하세요.

최소 권한

작업을 수행하는 데 필요한 최소 권한을 부여하는 보안 모범 사례입니다. 자세한 내용은 IAM 설명서의 최소 권한 적용을 참조하십시오.

리프트 앤드 시프트

7R을 참조하세요.

리틀 엔디안 시스템

가장 덜 중요한 바이트를 먼저 저장하는 시스템입니다. Endianness도 참조하세요.

LLM

대규모 언어 모델을 참조하세요.

하위 환경

환경을 참조하세요.

M

기계 학습(ML)

패턴 인식 및 학습에 알고리즘과 기법을 사용하는 인공 지능의 한 유형입니다. ML은 사물 인터넷(IoT) 데이터와 같은 기록된 데이터를 분석하고 학습하여 패턴을 기반으로 통계 모델을 생성합니다. 자세한 내용은 기계 학습을 참조하십시오.

기본 브랜치

브랜치를 참조하세요.

맬웨어

컴퓨터 보안 또는 개인 정보 보호를 손상하도록 설계된 소프트웨어입니다. 맬웨어는 컴퓨터 시스템을 방해하거나, 민감한 정보를 유출하거나, 무단 액세스를 가져올 수 있습니다. 맬웨어의 예로는 바이러스, 웜, 랜섬웨어, 트로이 목마, 스파이웨어, 키로거 등이 있습니다.

관리형 서비스

AWS 서비스 는 인프라 계층, 운영 체제 및 플랫폼을 AWS 운영하며 사용자는 엔드포인트에 액세스하여 데이터를 저장하고 검색합니다. Amazon Simple Storage Service(Amazon S3) 및 Amazon DynamoDB는 관리형 서비스의 예입니다. 이를 추상화된 서비스라고도 합니다.

제조 실행 시스템(MES)

원재료를 작업 현장의 완성된 제품으로 변환하는 생산 프로세스를 추적, 모니터링, 문서화 및 제어하기 위한 소프트웨어 시스템입니다.

MAP

마이그레이션 가속화 프로그램을 참조하세요.

메커니즘

도구를 생성하고 도구 채택을 유도한 다음 결과를 검사하여 조정하는 전체 프로세스입니다. 메커니즘은 작동 시 자체를 강화하고 개선하는 주기입니다. 자세한 내용은 AWS Well-Architected Framework의 메커니즘 구축을 참조하세요.

멤버 계정

조직의 일부인 관리 계정을 AWS 계정 제외한 모든 계정. AWS Organizations하나의 계정은 한 번에 하나의 조직 멤버만 될 수 있습니다.

MES

제조 실행 시스템을 참조하세요.

메시지 대기열 원격 측정 전송(MQTT)

리소스가 제한된 IoT 디바이스에 대한 게시/구독 패턴을 기반으로 하는 경량 M2M(machine-to-machine) 통신 프로토콜입니다.

마이크로서비스

잘 정의된 API를 통해 통신하고 일반적으로 소규모 자체 팀이 소유하는 소규모 독립 서비스입니다. 예를 들어, 보험 시스템에는 영업, 마케팅 등의 비즈니스 역량이나 구매, 청구, 분석 등의 하위 영역에 매핑되는 마이크로 서비스가 포함될 수 있습니다. 마이크로서비스의 이점으로 민첩성, 유연한 확장, 손쉬운 배포, 재사용 가능한 코드, 복원력 등이 있습니다. 자세한 내용은 AWS 서버리스 서비스를 사용하여 마이크로서비스 통합을 참조하세요.

마이크로서비스 아키텍처

각 애플리케이션 프로세스를 마이크로서비스로 실행하는 독립 구성 요소를 사용하여 애플리케이션을 구축하는 접근 방식입니다. 이러한 마이크로서비스는 경량 API를 사용하여 잘 정의된 인터페이스를 통해 통신합니다. 애플리케이션의 특정 기능에 대한 수요에 맞게 이 아키텍처의 각 마이크로서비스를 업데이트, 배포 및 조정할 수 있습니다. 자세한 내용은 에서 마이크로서비스 구현을 참조하세요 AWS.

Migration Acceleration Program(MAP)

조직이 클라우드로 전환하기 위한 강력한 운영 기반을 구축하고 초기 마이그레이션 비용을 상쇄하는 데 도움이 되는 컨설팅 지원, 교육 및 서비스를 제공하는 AWS 프로그램입니다. MAP에는 레거시 마이그레이션을 체계적인 방식으로 실행하기 위한 마이그레이션 방법론과 일반적인 마이그레이션 시나리오를 자동화하고 가속화하는 도구 세트가 포함되어 있습니다.

대규모 마이그레이션

애플리케이션 포트폴리오의 대다수를 웨이브를 통해 클라우드로 이동하는 프로세스로, 각 웨이브에서 더 많은 애플리케이션이 더 빠른 속도로 이동합니다. 이 단계에서는 이전 단계에서 배운 모범 사례와 교훈을 사용하여 팀, 도구 및 프로세스의 마이그레이션 팩토리를 구현하여 자동화 및 민첩한 제공을 통해 워크로드 마이그레이션을 간소화합니다. 이것은 AWS 마이그레이션 전략의 세 번째 단계입니다.

마이그레이션 팩토리

자동화되고 민첩한 접근 방식을 통해 워크로드 마이그레이션을 간소화하는 다기능 팀입니다. 마이그레이션 팩토리 팀에는 일반적으로 스프린트에서 일하는 운영, 비즈니스 분석가 및 소유자, 마이그레이션 엔지니어, 개발자, DevOps 전문가가 포함됩니다. 엔터프라이즈 애플리케이션 포트폴리오의 20~50%는 공장 접근 방식으로 최적화할 수 있는 반복되는 패턴으로 구성되어 있습니다. 자세한 내용은 이 콘텐츠 세트의 클라우드 마이그레이션 팩토리 가이드와 마이그레이션 팩토리에 대한 설명을 참조하십시오.

마이그레이션 메타데이터

마이그레이션을 완료하는 데 필요한 애플리케이션 및 서버에 대한 정보 각 마이그레이션 패턴에는 서로 다른 마이그레이션 메타데이터 세트가 필요합니다. 마이그레이션 메타데이터의 예로는 대상 서브넷, 보안 그룹 및 AWS 계정이 있습니다.

마이그레이션 패턴

사용되는 마이그레이션 전략, 마이그레이션 대상, 마이그레이션 애플리케이션 또는 서비스를 자세히 설명하는 반복 가능한 마이그레이션 작업입니다. 예: AWS Application Migration Service를 사용하여 Amazon EC2로 마이그레이션을 리호스팅합니다.

Migration Portfolio Assessment(MPA)

로 마이그레이션하기 위한 비즈니스 사례를 검증하기 위한 정보를 제공하는 온라인 도구입니다 AWS 클라우드. MPA는 상세한 포트폴리오 평가(서버 적정 규모 조정, 가격 책정, TCO 비교, 마이그레이션 비용 분석)와 마이그레이션 계획(애플리케이션 데이터 분석 및 데이터 수집, 애플리케이션 그룹화, 마이그레이션 우선순위 지정, 웨이브 계획)을 제공합니다. MPA 도구(로그인 필요)는 모든 AWS 컨설턴트와 APN 파트너 컨설턴트가 무료로 사용할 수 있습니다.

마이그레이션 준비 상태 평가(MRA)

AWS CAF를 사용하여 조직의 클라우드 준비 상태에 대한 인사이트를 얻고, 강점과 약점을 식별하고, 식별된 격차를 해소하기 위한 실행 계획을 수립하는 프로세스입니다. 자세한 내용은 마이그레이션 준비 가이드를 참조하십시오. MRA는 AWS 마이그레이션 전략의 첫 번째 단계입니다.

마이그레이션 전략

워크로드를 로 마이그레이션하는 데 사용되는 접근 방식입니다 AWS 클라우드. 자세한 내용은이 용어집의 7R 항목을 참조하고 대규모 마이그레이션을 가속화하기 위해 조직 동원을 참조하세요.

ML

기계 학습을 참조하세요.

현대화

비용을 절감하고 효율성을 높이고 혁신을 활용하기 위해 구식(레거시 또는 모놀리식) 애플리케이션과 해당 인프라를 클라우드의 민첩하고 탄력적이고 가용성이 높은 시스템으로 전환하는 것입니다. 자세한 내용은 의 애플리케이션 현대화 전략을 참조하세요 AWS 클라우드.

현대화 준비 상태 평가

조직 애플리케이션의 현대화 준비 상태를 파악하고, 이점, 위험 및 종속성을 식별하고, 조직이 해당 애플리케이션의 향후 상태를 얼마나 잘 지원할 수 있는지를 확인하는 데 도움이 되는 평가입니다. 평가 결과는 대상 아키텍처의 청사진, 현대화 프로세스의 개발 단계와 마일스톤을 자세히 설명하는 로드맵 및 파악된 격차를 해소하기 위한 실행 계획입니다. 자세한 내용은 의 애플리케이션에 대한 현대화 준비 상태 평가를 참조하세요 AWS 클라우드.

모놀리식 애플리케이션(모놀리식 유형)

긴밀하게 연결된 프로세스를 사용하여 단일 서비스로 실행되는 애플리케이션입니다. 모놀리식 애플리케이션에는 몇 가지 단점이 있습니다. 한 애플리케이션 기능에 대한 수요가 급증하면 전체 아키텍처 규모를 조정해야 합니다. 코드 베이스가 커지면 모놀리식 애플리케이션의 기능을 추가하거나 개선하는 것도 더 복잡해집니다. 이러한 문제를 해결하기 위해 마이크로서비스 아키텍처를 사용할 수 있습니다. 자세한 내용은 마이크로서비스로 모놀리식 유형 분해를 참조하십시오.

MPA

마이그레이션 포트폴리오 평가를 참조하세요.

MQTT

메시지 대기열 원격 측정 전송을 참조하세요.

멀티클래스 분류

여러 클래스에 대한 예측(2개 이상의 결과 중 하나 예측)을 생성하는 데 도움이 되는 프로세스입니다. 예를 들어, ML 모델이 '이 제품은 책인가요, 자동차인가요, 휴대폰인가요?' 또는 '이 고객이 가장 관심을 갖는 제품 범주는 무엇인가요?'라고 물을 수 있습니다.

변경 가능한 인프라

프로덕션 워크로드를 위해 기존 인프라를 업데이트하고 수정하는 모델입니다. 일관성, 신뢰성 및 예측 가능성을 높이기 위해 AWS Well-Architected Framework는 변경 불가능한 인프라를 모범 사례로 사용할 것을 권장합니다.

O

OAC

오리진 액세스 제어를 참조하세요.

OAI

오리진 액세스 ID를 참조하세요.

OCM

조직 변경 관리를 참조하세요.

오프라인 마이그레이션

마이그레이션 프로세스 중 소스 워크로드가 중단되는 마이그레이션 방법입니다. 이 방법은 가동 중지 증가를 수반하며 일반적으로 작고 중요하지 않은 워크로드에 사용됩니다.

OI

작업 통합을 참조하세요.

OLA

운영 수준 계약을 참조하세요.

온라인 마이그레이션

소스 워크로드를 오프라인 상태로 전환하지 않고 대상 시스템에 복사하는 마이그레이션 방법입니다. 워크로드에 연결된 애플리케이션은 마이그레이션 중에도 계속 작동할 수 있습니다. 이 방법은 가동 중지 차단 또는 최소화를 수반하며 일반적으로 중요한 프로덕션 워크로드에 사용됩니다.

OPC-UA

Open Process Communications - Unified Architecture를 참조하세요.

Open Process Communications - 통합 아키텍처(OPC-UA)

산업 자동화를 위한 M2M(Machinemachine-to-machine) 통신 프로토콜입니다. OPC-UA는 데이터 암호화, 인증 및 권한 부여 체계와 상호 운용성 표준을 제공합니다.

운영 수준 협약(OLA)

서비스 수준에 관한 계약(SLA)을 지원하기 위해 직무 IT 그룹이 서로에게 제공하기로 약속한 내용을 명확히 하는 계약입니다.

운영 준비 상태 검토(ORR)

인시던트 및 가능한 장애의 범위를 이해, 평가, 예방 또는 줄이는 데 도움이 되는 질문 체크리스트 및 관련 모범 사례입니다. 자세한 내용은 AWS Well-Architected Framework의 운영 준비 검토(ORR)를 참조하세요.

운영 기술(OT)

물리적 환경과 함께 작동하여 산업 운영, 장비 및 인프라를 제어하는 하드웨어 및 소프트웨어 시스템입니다. 제조에서 OT 및 정보 기술(IT) 시스템의 통합은 Industry 4.0 혁신의 핵심 초점입니다.

운영 통합(OI)

클라우드에서 운영을 현대화하는 프로세스로 준비 계획, 자동화 및 통합을 수반합니다. 자세한 내용은 운영 통합 가이드를 참조하십시오.

조직 트레일

조직 AWS 계정 내 모든에 대한 모든 이벤트를 로깅 AWS CloudTrail 하는에서 생성된 추적입니다 AWS Organizations. 이 트레일은 조직에 속한 각 AWS 계정 에 생성되고 각 계정의 활동을 추적합니다. 자세한 내용은 CloudTrail 설명서의 Creating a trail for an organization을 참조하십시오.

조직 변경 관리(OCM)

사람, 문화 및 리더십 관점에서 중대하고 파괴적인 비즈니스 혁신을 관리하기 위한 프레임워크입니다. OCM은 변화 채택을 가속화하고, 과도기적 문제를 해결하고, 문화 및 조직적 변화를 주도함으로써 조직이 새로운 시스템 및 전략을 준비하고 전환할 수 있도록 지원합니다. AWS 마이그레이션 전략에서는 클라우드 채택 프로젝트에 필요한 변경 속도 때문에이 프레임워크를 인력 가속화라고 합니다. 자세한 내용은 사용 가이드를 참조하십시오.

오리진 액세스 제어(OAC)

CloudFront에서 Amazon Simple Storage Service(S3) 콘텐츠를 보호하기 위해 액세스를 제한하는 고급 옵션입니다. OAC는 AWS KMS (SSE-KMS)를 사용한 모든 서버 측 암호화 AWS 리전와 S3 버킷에 대한 동적 PUT 및 DELETE 요청에서 모든 S3 버킷을 지원합니다.

오리진 액세스 ID(OAI)

CloudFront에서 Amazon S3 콘텐츠를 보호하기 위해 액세스를 제한하는 옵션입니다. OAI를 사용하면 CloudFront는 Amazon S3가 인증할 수 있는 보안 주체를 생성합니다. 인증된 보안 주체는 특

정 CloudFront 배포를 통해서만 S3 버킷의 콘텐츠에 액세스할 수 있습니다. 더 세분화되고 향상된 액세스 제어를 제공하는 OAC도 참조하십시오.

ORR

운영 준비 상태 검토를 참조하세요.

OT

운영 기술을 참조하세요.

아웃바운드(송신) VPC

AWS 다중 계정 아키텍처에서 애플리케이션 내에서 시작된 네트워크 연결을 처리하는 VPC입니다. AWS Security Reference Architecture에서는 애플리케이션과 더 넓은 인터넷 간의 양방향 인터페이스를 보호하기 위해 인바운드, 아웃바운드 및 검사 VPC로 네트워크 계정을 설정할 것을 권장합니다.

P

권한 경계

사용자나 역할이 가질 수 있는 최대 권한을 설정하기 위해 IAM 보안 주체에 연결되는 IAM 관리 정책입니다. 자세한 내용은 IAM 설명서의 권한 경계를 참조하십시오.

개인 식별 정보(PII)

직접 보거나 다른 관련 데이터와 함께 짝을 지을 때 개인의 신원을 합리적으로 추론하는 데 사용할 수 있는 정보입니다. PII의 예로는 이름, 주소, 연락처 정보 등이 있습니다.

PII

개인 식별 정보를 참조하세요.

플레이북

클라우드에서 핵심 운영 기능을 제공하는 등 마이그레이션과 관련된 작업을 캡처하는 일련의 사전 정의된 단계입니다. 플레이북은 스크립트, 자동화된 런북 또는 현대화된 환경을 운영하는 데 필요한 프로세스나 단계 요약의 형태를 취할 수 있습니다.

PLC

프로그래밍 가능한 로직 컨트롤러를 참조하세요.

PLM

제품 수명 주기 관리를 참조하세요.

정책

권한을 정의하거나(자격 증명 기반 정책 참조), 액세스 조건을 지정하거나(리소스 기반 정책 참조), 조직의 모든 계정에 대한 최대 권한을 정의할 수 있는 객체 AWS Organizations 입니다(서비스 제어 정책 참조).

다국어 지속성

데이터 액세스 패턴 및 기타 요구 사항을 기반으로 독립적으로 마이크로서비스의 데이터 스토리지 기술 선택. 마이크로서비스가 동일한 데이터 스토리지 기술을 사용하는 경우 구현 문제가 발생하거나 성능이 저하될 수 있습니다. 요구 사항에 가장 적합한 데이터 스토어를 사용하면 마이크로서비스를 더 쉽게 구현하고 성능과 확장성을 높일 수 있습니다. 자세한 내용은 마이크로서비스에서 데이터 지속성 활성화를 참조하십시오.

포트폴리오 평가

마이그레이션을 계획하기 위해 애플리케이션 포트폴리오를 검색 및 분석하고 우선순위를 정하는 프로세스입니다. 자세한 내용은 마이그레이션 준비 상태 평가를 참조하십시오.

조건자

`false`일반적으로 `WHERE` 절에 있는 `true` 또는를 반환하는 쿼리 조건입니다.

조건자 푸시다운

전송 전에 쿼리의 데이터를 필터링하는 데이터베이스 쿼리 최적화 기법입니다. 이렇게 하면 관계형 데이터베이스에서 검색하고 처리해야 하는 데이터의 양이 줄어들고 쿼리 성능이 향상됩니다.

예방적 제어

이벤트 발생을 방지하도록 설계된 보안 제어입니다. 이 제어는 네트워크에 대한 무단 액세스나 원치 않는 변경을 방지하는 데 도움이 되는 1차 방어선입니다. 자세한 내용은 Implementing security controls on AWS의 Preventative controls를 참조하십시오.

보안 주체

작업을 수행하고 리소스에 액세스할 수 AWS 있는의 엔터티입니다. 이 엔터티는 일반적으로 , AWS 계정 IAM 역할 또는 사용자의 루트 사용자입니다. 자세한 내용은 IAM 설명서의 역할 용어 및 개념의 보안 주체를 참조하십시오.

설계에 따른 개인 정보 보호

전체 개발 프로세스를 통해 개인 정보를 고려하는 시스템 엔지니어링 접근 방식입니다.

프라이빗 호스팅 영역

Amazon Route 53에서 하나 이상의 VPC 내 도메인과 하위 도메인에 대한 DNS 쿼리에 응답하는 방법에 대한 정보가 담긴 컨테이너입니다. 자세한 내용은 Route 53 설명서의 프라이빗 호스팅 영역 작업을 참조하십시오.

사전 예방적 제어

규정 미준수 리소스의 배포를 방지하도록 설계된 보안 제어입니다. 이러한 제어는 리소스가 프로비저닝되기 전에 리소스를 스캔합니다. 리소스가 컨트롤을 준수하지 않으면 프로비저닝되지 않습니다. 자세한 내용은 AWS Control Tower 설명서의 제어 참조 가이드를 참조하고 보안 제어 구현의 사전 예방적 제어를 참조하세요. AWS

제품 수명 주기 관리(PLM)

설계, 개발 및 출시부터 성장 및 성숙도, 거부 및 제거에 이르기까지 전체 수명 주기 동안 제품의 데이터 및 프로세스 관리.

프로덕션 환경

환경을 참조하세요.

프로그래밍 가능한 로직 컨트롤러(PLC)

제조 분야에서는 기계를 모니터링하고 제조 프로세스를 자동화하는 매우 안정적이고 적응력이 뛰어난 컴퓨터입니다.

프롬프트 체인

한 LLM 프롬프트의 출력을 다음 프롬프트의 입력으로 사용하여 더 나은 응답을 생성합니다. 이 기법은 복잡한 작업을 하위 작업으로 나누거나 예비 응답을 반복적으로 구체화하거나 확장하는 데 사용됩니다. 이를 통해 모델 응답의 정확성과 관련성을 개선하고 보다 세분화되고 개인화된 결과를 얻을 수 있습니다.

가명화

데이터세트의 개인 식별자를 자리 표시자 값으로 바꾸는 프로세스입니다. 가명화는 개인 정보를 보호하는 데 도움이 될 수 있습니다. 가명화된 데이터는 여전히 개인 데이터로 간주됩니다.

게시/구독(pub/sub)

마이크로서비스 간의 비동기 통신을 지원하여 확장성과 응답성을 개선하는 패턴입니다. 예를 들어 마이크로서비스 기반 MES에서 마이크로서비스는 다른 마이크로서비스가 구독할 수 있는 채널에 이벤트 메시지를 게시할 수 있습니다. 시스템은 게시 서비스를 변경하지 않고도 새 마이크로서비스를 추가할 수 있습니다.

Q

쿼리 계획

SQL 관계형 데이터베이스 시스템의 데이터에 액세스하는 데 사용되는 지침과 같은 일련의 단계입니다.

쿼리 계획 회귀

데이터베이스 서비스 최적화 프로그램이 데이터베이스 환경을 변경하기 전보다 덜 최적의 계획을 선택하는 경우입니다. 통계, 제한 사항, 환경 설정, 쿼리 파라미터 바인딩 및 데이터베이스 엔진 업데이트의 변경으로 인해 발생할 수 있습니다.

R

RACI 매트릭스

책임, 책임, 상담, 정보 제공(RACI)을 참조하세요.

RAG

Retrieval Augmented Generation을 참조하세요.

랜섬웨어

결제가 완료될 때까지 컴퓨터 시스템이나 데이터에 대한 액세스를 차단하도록 설계된 악성 소프트웨어입니다.

RASCI 매트릭스

책임, 책임, 상담, 정보 제공(RACI)을 참조하세요.

RCAC

행 및 열 액세스 제어를 참조하세요.

읽기 전용 복제본

읽기 전용 용도로 사용되는 데이터베이스의 사본입니다. 쿼리를 읽기 전용 복제본으로 라우팅하여 기본 데이터베이스의 로드를 줄일 수 있습니다.

재설계

7R을 참조하세요.

Recovery Point Objective(RPO)

마지막 데이터 복구 시점 이후 허용되는 최대 시간입니다. 이에 따라 마지막 복구 시점과 서비스 중단 사이에 허용되는 데이터 손실로 간주되는 범위가 결정됩니다.

Recovery Time Objective(RTO)

서비스 중단과 서비스 복원 사이의 허용 가능한 지연 시간입니다.

리팩터링

7R을 참조하세요.

리전

지리적 영역의 AWS 리소스 모음입니다. 각 AWS 리전 는 내결함성, 안정성 및 복원력을 제공하기 위해 서로 격리되고 독립적입니다. 자세한 내용은 계정에서 사용할 수 있는 지정을 참조 AWS 리전 하세요.

회귀

숫자 값을 예측하는 ML 기법입니다. 예를 들어, '이 집은 얼마에 팔릴까?'라는 문제를 풀기 위해 ML 모델은 선형 회귀 모델을 사용하여 주택에 대해 알려진 사실(예: 면적)을 기반으로 주택의 매매 가격을 예측할 수 있습니다.

리호스팅

7R을 참조하세요.

release

배포 프로세스에서 변경 사항을 프로덕션 환경으로 승격시키는 행위입니다.

재배치

7R을 참조하세요.

리플랫포밍

7R을 참조하세요.

재구매

7R을 참조하세요.

복원력

중단에 저항하거나 복구할 수 있는 애플리케이션의 기능입니다. 에서 복원력을 계획할 때 고가용성 및 재해 복구가 일반적인 고려 사항입니다 AWS 클라우드. 자세한 내용은 AWS 클라우드 복원력을 참조하세요.

리소스 기반 정책

Amazon S3 버킷, 엔드포인트, 암호화 키 등의 리소스에 연결된 정책입니다. 이 유형의 정책은 액세스가 허용된 보안 주체, 지원되는 작업 및 충족해야 하는 기타 조건을 지정합니다.

RACI(Responsible, Accountable, Consulted, Informed) 매트릭스

마이그레이션 활동 및 클라우드 운영에 참여하는 모든 당사자의 역할과 책임을 정의하는 매트릭스입니다. 매트릭스 이름은 매트릭스에 정의된 책임 유형에서 파생됩니다. 실무 담당자 (R), 의사 결정권자 (A), 업무 수행 조언자 (C), 결과 통보 대상자 (I). 지원자는 (S) 선택사항입니다. 지원자를 포함하면 매트릭스를 RASCI 매트릭스라고 하고, 지원자를 제외하면 RACI 매트릭스라고 합니다.

대응 제어

보안 기준에서 벗어나거나 부정적인 이벤트를 해결하도록 설계된 보안 제어입니다. 자세한 내용은 Implementing security controls on AWS의 Responsive controls를 참조하십시오.

retain

7R을 참조하세요.

사용 중지

7R을 참조하세요.

검색 증강 세대(RAG)

응답을 생성하기 전에 LLM이 훈련 데이터 소스 외부에 있는 신뢰할 수 있는 데이터 소스를 참조하는 생성형 AI 기술입니다. 예를 들어 RAG 모델은 조직의 지식 기반 또는 사용자 지정 데이터에 대한 의미 검색을 수행할 수 있습니다. 자세한 내용은 RAG란 무엇입니까?를 참조하십시오.

교체

공격자가 보안 인증 정보에 액세스하는 것을 더 어렵게 만들기 위해 보안 암호를 주기적으로 업데이트하는 프로세스입니다.

행 및 열 액세스 제어(RCAC)

액세스 규칙이 정의된 기본적이고 유연한 SQL 표현식을 사용합니다. RCAC는 행 권한과 열 마스크로 구성됩니다.

RPO

복구 시점 목표를 참조하세요.

RTO

복구 시간 목표를 참조하세요.

런북

특정 작업을 수행하는 데 필요한 일련의 수동 또는 자동 절차입니다. 일반적으로 오류율이 높은 반복 작업이나 절차를 간소화하기 위해 런북을 만듭니다.

S

SAML 2.0

많은 ID 제공업체(idP)에서 사용하는 개방형 표준입니다. 이 기능을 사용하면 연동 SSO(Single Sign-On)를 AWS Management Console 사용할 수 있으므로 사용자는 조직의 모든 사용자에 대해 IAM에서 사용자를 생성하지 않고도에 로그인하거나 AWS API 작업을 호출할 수 있습니다. SAML 2.0 기반 페더레이션에 대한 자세한 내용은 IAM 설명서의 SAML 2.0 기반 페더레이션 정보를 참조하십시오.

SCADA

감독 제어 및 데이터 획득을 참조하세요.

SCP

서비스 제어 정책을 참조하세요.

secret

에는 암호 또는 사용자 자격 증명과 같이 암호화된 형식으로 저장하는 AWS Secrets Manager기밀 또는 제한된 정보가 있습니다. 보안 암호 값과 메타데이터로 구성됩니다. 보안 암호 값은 바이너리, 단일 문자열 또는 여러 문자열일 수 있습니다. 자세한 내용은 Secrets Manager 설명서의 Secrets Manager 보안 암호에 무엇이 있습니까?를 참조하세요.

설계별 보안

전체 개발 프로세스에서 보안을 고려하는 시스템 엔지니어링 접근 방식입니다.

보안 제어

위협 행위자가 보안 취약성을 악용하는 능력을 방지, 탐지 또는 감소시키는 기술적 또는 관리적 가드레일입니다. 보안 제어에는 예방, 탐지, 대응 및 사전 예방의 네 가지 주요 유형이 있습니다.

보안 강화

공격 표면을 줄여 공격에 대한 저항력을 높이는 프로세스입니다. 더 이상 필요하지 않은 리소스 제거, 최소 권한 부여의 보안 모범 사례 구현, 구성 파일의 불필요한 기능 비활성화 등의 작업이 여기에 포함될 수 있습니다.

보안 정보 및 이벤트 관리(SIEM) 시스템

보안 정보 관리(SIM)와 보안 이벤트 관리(SEM) 시스템을 결합하는 도구 및 서비스입니다. SIEM 시스템은 서버, 네트워크, 디바이스 및 기타 소스에서 데이터를 수집, 모니터링 및 분석하여 위협과 보안 침해를 탐지하고 알림을 생성합니다.

보안 응답 자동화

보안 이벤트에 자동으로 응답하거나 해결하도록 설계된 사전 정의되고 프로그래밍된 작업입니다. 이러한 자동화는 보안 모범 사례를 구현하는 데 도움이 되는 탐지 또는 대응 AWS 보안 제어 역할을 합니다. 자동 응답 작업의 예로는 VPC 보안 그룹 수정, Amazon EC2 인스턴스 패치 적용 또는 자격 증명 교체 등이 있습니다.

서버 측 암호화

데이터를 AWS 서비스 수신하는가 대상에서 데이터를 암호화합니다.

서비스 제어 정책(SCP)

AWS Organizations에 속한 조직의 모든 계정에 대한 권한을 중앙 집중식으로 제어하는 정책입니다. SCP는 관리자가 사용자 또는 역할에 위임할 수 있는 작업에 대해 제한을 설정하거나 가드레일을 정의합니다. SCP를 허용 목록 또는 거부 목록으로 사용하여 허용하거나 금지할 서비스 또는 작업을 지정할 수 있습니다. 자세한 내용은 AWS Organizations 설명서의 서비스 제어 정책을 참조하세요.

서비스 엔드포인트

에 대한 진입점의 URL입니다 AWS 서비스. 엔드포인트를 사용하여 대상 서비스에 프로그래밍 방식으로 연결할 수 있습니다. 자세한 내용은 AWS 일반 참조의 AWS 서비스 엔드포인트를 참조하십시오.

서비스 수준에 관한 계약(SLA)

IT 팀이 고객에게 제공하기로 약속한 내용(예: 서비스 가동 시간 및 성능)을 명시한 계약입니다.

서비스 수준 지표(SLI)

오류율, 가용성 또는 처리량과 같은 서비스의 성능 측면에 대한 측정입니다.

서비스 수준 목표(SLO)

서비스 수준 지표로 측정되는 서비스의 상태를 나타내는 대상 지표입니다.

공동 책임 모델

클라우드 보안 및 규정 준수를 AWS 위해와 공유하는 책임을 설명하는 모델입니다. AWS 는 클라우드의 보안을 담당하는 반면, 사용자는 클라우드의 보안을 담당합니다. 자세한 내용은 공동 책임 모델을 참조하십시오.

SIEM

보안 정보 및 이벤트 관리 시스템을 참조하세요.

단일 장애 지점(SPOF)

애플리케이션을 중단시킬 수 있는 애플리케이션의 중요한 단일 구성 요소에서 장애가 발생한 경우.

SLA

서비스 수준 계약을 참조하세요.

SLI

서비스 수준 표시기를 참조하세요.

SLO

서비스 수준 목표를 참조하세요.

분할 앤 시드 모델

현대화 프로젝트를 확장하고 가속화하기 위한 패턴입니다. 새로운 기능과 제품 릴리스가 정의되면 핵심 팀이 분할되어 새로운 제품 팀이 만들어집니다. 이를 통해 조직의 역량과 서비스 규모를 조정하고, 개발자 생산성을 개선하고, 신속한 혁신을 지원할 수 있습니다. 자세한 내용은 에서 애플리케이션 현대화에 대한 단계별 접근 방식을 참조하세요 AWS 클라우드.

SPOF

단일 장애 지점을 참조하세요.

스타 스키마

하나의 큰 팩트 테이블을 사용하여 트랜잭션 또는 측정된 데이터를 저장하고 하나 이상의 작은 차원 테이블을 사용하여 데이터 속성을 저장하는 데이터베이스 조직 구조입니다. 이 구조는 데이터 웨어하우스 또는 비즈니스 인텔리전스용으로 설계되었습니다.

Strangler Fig 패턴

레거시 시스템을 폐기할 수 있을 때까지 시스템 기능을 점진적으로 다시 작성하고 교체하여 모놀리식 시스템을 현대화하기 위한 접근 방식. 이 패턴은 무화과 덩굴이 나무로 자라 결국 숙주를 압도

하고 대체하는 것과 비슷합니다. Martin Fowler가 모놀리식 시스템을 다시 작성할 때 위험을 관리하는 방법으로 이 패턴을 도입했습니다. 이 패턴을 적용하는 방법의 예는 컨테이너 및 Amazon API Gateway를 사용하여 기존의 Microsoft ASP.NET(ASMX) 웹 서비스를 점진적으로 현대화하는 방법을 참조하십시오.

서브넷

VPC의 IP 주소 범위입니다. 서브넷은 단일 가용 영역에 상주해야 합니다.

감독 제어 및 데이터 획득(SCADA)

제조에서 하드웨어와 소프트웨어를 사용하여 물리적 자산과 프로덕션 작업을 모니터링하는 시스템입니다.

대칭 암호화

동일한 키를 사용하여 데이터를 암호화하고 복호화하는 암호화 알고리즘입니다.

합성 테스트

사용자 상호 작용을 시뮬레이션하여 잠재적 문제를 감지하거나 성능을 모니터링하는 방식으로 시스템을 테스트합니다. Amazon CloudWatch Synthetics를 사용하여 이러한 테스트를 생성할 수 있습니다.

시스템 프롬프트

LLM에 컨텍스트, 지침 또는 지침을 제공하여 동작을 지시하는 기법입니다. 시스템 프롬프트는 컨텍스트를 설정하고 사용자와의 상호 작용을 위한 규칙을 설정하는 데 도움이 됩니다.

T

tags

AWS 리소스를 구성하기 위한 메타데이터 역할을 하는 키-값 페어입니다. 태그를 사용하면 리소스를 손쉽게 관리, 식별, 정리, 검색 및 필터링할 수 있습니다. 자세한 내용은 AWS 리소스에 태그 지정을 참조하십시오.

대상 변수

지도 ML에서 예측하려는 값으로, 결과 변수라고도 합니다. 예를 들어, 제조 설정에서 대상 변수는 제품 결함일 수 있습니다.

작업 목록

런북을 통해 진행 상황을 추적하는 데 사용되는 도구입니다. 작업 목록에는 런북의 개요와 완료해야 할 일반 작업 목록이 포함되어 있습니다. 각 일반 작업에 대한 예상 소요 시간, 소유자 및 진행 상황이 작업 목록에 포함됩니다.

테스트 환경

환경을 참조하세요.

훈련

ML 모델이 학습할 수 있는 데이터를 제공하는 것입니다. 훈련 데이터에는 정답이 포함되어야 합니다. 학습 알고리즘은 훈련 데이터에서 대상(예측하려는 답)에 입력 데이터 속성을 매핑하는 패턴을 찾고, 이러한 패턴을 캡처하는 ML 모델을 출력합니다. 그런 다음 ML 모델을 사용하여 대상을 모르는 새 데이터에 대한 예측을 할 수 있습니다.

전송 게이트웨이

VPC와 온프레미스 네트워크를 상호 연결하는 데 사용할 수 있는 네트워크 전송 허브입니다. 자세한 내용은 AWS Transit Gateway 설명서의 전송 게이트웨이란 무엇입니까?를 참조하세요.

트렁크 기반 워크플로

개발자가 기능 브랜치에서 로컬로 기능을 구축하고 테스트한 다음 해당 변경 사항을 기본 브랜치에 병합하는 접근 방식입니다. 이후 기본 브랜치는 개발, 프로덕션 이전 및 프로덕션 환경에 순차적으로 구축됩니다.

신뢰할 수 있는 액세스

사용자를 대신하여 AWS Organizations 및 해당 계정에서 조직에서 작업을 수행하도록 지정하는 서비스에 권한 부여. 신뢰할 수 있는 서비스는 필요할 때 각 계정에 서비스 연결 역할을 생성하여 관리 작업을 수행합니다. 자세한 내용은 설명서의 다른 AWS 서비스와 AWS Organizations 함께 사용을 참조하세요 AWS Organizations .

튜닝

ML 모델의 정확도를 높이기 위해 훈련 프로세스의 측면을 여러 변경하는 것입니다. 예를 들어, 레이블링 세트를 생성하고 레이블을 추가한 다음 다양한 설정에서 이러한 단계를 여러 번 반복하여 모델을 최적화하는 방식으로 ML 모델을 훈련할 수 있습니다.

피자 두 판 팀

피자 두 판이면 충분한 소규모 DevOps 팀. 피자 두 판 팀 규모는 소프트웨어 개발에 있어 가능한 최상의 공동 작업 기회를 보장합니다.

U

불확실성

예측 ML 모델의 신뢰성을 저해할 수 있는 부정확하거나 불완전하거나 알려지지 않은 정보를 나타내는 개념입니다. 불확실성에는 두 가지 유형이 있습니다. 인식론적 불확실성은 제한적이고 불완전한 데이터에 의해 발생하는 반면, 우연한 불확실성은 데이터에 내재된 노이즈와 무작위성에 의해 발생합니다. 자세한 내용은 Quantifying uncertainty in deep learning systems 가이드를 참조하십시오.

차별화되지 않은 작업

애플리케이션을 만들고 운영하는 데 필요하지만 최종 사용자에게 직접적인 가치를 제공하거나 경쟁 우위를 제공하지 못하는 작업을 헤비 리프팅이라고도 합니다. 차별화되지 않은 작업의 예로는 조달, 유지보수, 용량 계획 등이 있습니다.

상위 환경

환경을 참조하세요.

V

정리

스토리지를 회수하고 성능을 향상시키기 위해 증분 업데이트 후 정리 작업을 수반하는 데이터베이스 유지 관리 작업입니다.

버전 제어

리포지토리의 소스 코드 변경과 같은 변경 사항을 추적하는 프로세스 및 도구입니다.

VPC 피어링

프라이빗 IP 주소를 사용하여 트래픽을 라우팅할 수 있게 하는 두 VPC 간의 연결입니다. 자세한 내용은 Amazon VPC 설명서의 VPC 피어링이란?을 참조하십시오.

취약성

시스템 보안을 손상시키는 소프트웨어 또는 하드웨어 결함입니다.

W

웜 캐시

자주 액세스하는 최신 관련 데이터를 포함하는 버퍼 캐시입니다. 버퍼 캐시에서 데이터베이스 인스턴스를 읽을 수 있기 때문에 주 메모리나 디스크에서 읽는 것보다 빠릅니다.

웜 데이터

자주 액세스하지 않는 데이터입니다. 이런 종류의 데이터를 쿼리할 때는 일반적으로 적절히 느린 쿼리가 허용됩니다.

창 함수

현재 레코드와 어떤 식으로든 관련된 행 그룹에 대해 계산을 수행하는 SQL 함수입니다. 창 함수는 이동 평균을 계산하거나 현재 행의 상대 위치를 기반으로 행 값에 액세스하는 등의 작업을 처리하는 데 유용합니다.

워크로드

고객 대면 애플리케이션이나 백엔드 프로세스 같이 비즈니스 가치를 창출하는 리소스 및 코드 모음입니다.

워크스트림

마이그레이션 프로젝트에서 특정 작업 세트를 담당하는 직무 그룹입니다. 각 워크스트림은 독립적이지만 프로젝트의 다른 워크스트림을 지원합니다. 예를 들어, 포트폴리오 워크스트림은 애플리케이션 우선순위 지정, 웨이브 계획, 마이그레이션 메타데이터 수집을 담당합니다. 포트폴리오 워크스트림은 이러한 자산을 마이그레이션 워크스트림에 전달하고, 마이그레이션 워크스트림은 서버와 애플리케이션을 마이그레이션합니다.

WORM

쓰기를 한 번, 많이 읽기를 참조하세요.

WQF

AWS 워크로드 검증 프레임워크를 참조하세요.

한 번 쓰기, 많이 읽기(WORM)

데이터를 한 번 쓰고 데이터가 삭제되거나 수정되지 않도록 하는 스토리지 모델입니다. 권한 있는 사용자는 필요한 만큼 데이터를 읽을 수 있지만 변경할 수는 없습니다. 이 데이터 스토리지 인프라는 변경할 수 없는 것으로 간주됩니다.

Z

제로데이 익스플로잇

제로데이 취약성을 활용하는 공격, 일반적으로 맬웨어입니다.

제로데이 취약성

프로덕션 시스템의 명백한 결함 또는 취약성입니다. 위협 행위자는 이러한 유형의 취약성을 사용하여 시스템을 공격할 수 있습니다. 개발자는 공격의 결과로 취약성을 인지하는 경우가 많습니다.

제로샷 프롬프트

LLM에 작업 수행에 대한 지침을 제공하지만 작업에 도움이 될 수 있는 예제(샷)는 제공하지 않습니다. LLM은 사전 훈련된 지식을 사용하여 작업을 처리해야 합니다. 제로샷 프롬프트의 효과는 작업의 복잡성과 프롬프트의 품질에 따라 달라집니다. 스크린샷이 거의 없는 프롬프트도 참조하세요.

좀비 애플리케이션

평균 CPU 및 메모리 사용량이 5% 미만인 애플리케이션입니다. 마이그레이션 프로젝트에서는 이러한 애플리케이션을 사용 중지하는 것이 일반적입니다.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.