aws

Amazon EMR 서버리스 사용 설명서

아마존 EMR

아마존 EMR: Amazon EMR 서버리스 사용 설명서

Table of Contents

Amazon EMR 서버리스란 무엇입니까?

Amazon EMR 서버리스는 서버리스 런타임 환경을 EMR 제공하는 Amazon용 배포 옵션입니다. 이를 통해 Apache Spark 및 Apache Hive와 같은 최신 오픈 소스 프레임워크를 사용하는 분석 애플리케이션의 운영이 간소화됩니다. EMR서버리스를 사용하면 이러한 프레임워크로 애플리케이션을 실행하기 위해 클러스터를 구성, 최적화, 보호 또는 운영할 필요가 없습니다.

EMR서버리스를 사용하면 데이터 처리 작업에 리소스가 과도하게 또는 부족하게 프로비저닝되는 것을 방지할 수 있습니다. EMR서버리스는 애플리케이션에 필요한 리소스를 자동으로 결정하고, 작업을 처리하는 데 필요한 리소스를 확보하고, 작업이 완료되면 리소스를 릴리스합니다. 대화형 데이터 분석과 같이 몇 초 내에 애플리케이션에 응답해야 하는 사용 사례의 경우 애플리케이션을 생성할 때 애플리케이션에 필요한 리소스를 사전 초기화할 수 있습니다.

EMR서버리스를 사용하면 오픈 소스 호환성, 동시성EMR, 인기 프레임워크의 최적화된 런타임 성능 등 Amazon의 이점을 계속 누릴 수 있습니다.

EMR서버리스는 오픈 소스 프레임워크를 사용하여 애플리케이션을 쉽게 운영하고자 하는 고객에게 적합합니다. 빠른 작업 시작, 자동 용량 관리 및 간단한 비용 관리를 제공합니다.

개념

이 섹션에서는 서버리스 사용 설명서 전체에 나타나는 EMR 서버리스 용어 및 개념을 다룹니다. EMR

릴리스 버전

Amazon EMR 릴리스는 빅 데이터 생태계의 오픈 소스 애플리케이션 세트입니다. 각 릴리스에는 EMR 서버리스에서 애플리케이션을 실행할 수 있도록 배포 및 구성하도록 선택한 다양한 빅 데이터 애플리케이션, 구성 요소 및 기능이 포함되어 있습니다. 애플리케이션을 생성할 때 릴리스 버전을 지정해야 합니다. 애플리케이션에서 사용할 Amazon EMR 릴리스 버전과 오픈 소스 프레임워크 버전을 선택합니다. 시험판 버전에 대한 자세한 내용은 을 참조하십시오Amazon EMR 서버리스 릴리스 버전.

애플리케이션

EMR서버리스를 사용하면 오픈 소스 분석 프레임워크를 사용하는 EMR 서버리스 애플리케이션을 하나 이상 만들 수 있습니다. 애플리케이션을 만들려면 다음 속성을 지정해야 합니다.

- 사용하려는 오픈 소스 프레임워크 버전의 Amazon EMR 릴리스 버전입니다. 릴리스 버전을 확인하려면 을 참조하십시오Amazon EMR 서버리스 릴리스 버전.
- 애플리케이션에서 사용할 특정 런타임 (예: Apache Spark 또는 Apache Hive)

애플리케이션을 생성한 후 애플리케이션에 데이터 처리 작업 또는 대화형 요청을 제출할 수 있습니다.

각 EMR 서버리스 애플리케이션은 다른 애플리케이션과는 별도로 안전한 Amazon Virtual Private Cloud (VPC) 에서 실행됩니다. 또한 다음을 사용할 수 있습니다. AWS Identity and Access Management (IAM) 정책을 사용하여 애플리케이션에 액세스할 수 있는 사용자 및 역할을 정의합니다. 또한 한도를 지정하여 애플리케이션에서 발생하는 사용 비용을 제어하고 추적할 수 있습니다.

다음 작업을 수행해야 하는 경우 여러 애플리케이션을 만드는 것을 고려해 보십시오.

- 다양한 오픈소스 프레임워크 사용
- 사용 사례별로 다른 버전의 오픈소스 프레임워크를 사용하십시오.
- 한 버전에서 다른 버전으로 업그레이드할 때 A/B 테스트를 수행하십시오.
- 테스트 및 프로덕션 시나리오를 위한 별도의 논리적 환경 유지
- 독립적인 비용 관리 및 사용량 추적을 통해 여러 팀을 위한 별도의 논리적 환경을 제공합니다.
- 서로 다른 line-of-business 애플리케이션을 분리하십시오.

EMR서버리스는 지역 내 여러 가용 영역에서 워크로드가 실행되는 방식을 단순화하는 지역 서비스입니다. EMR서버리스에서 애플리케이션을 사용하는 방법에 대한 자세한 내용은 을 참조하십시오. 애플리케이션과의 상호 작용

작업 실행

작업 실행은 EMR 서버리스 애플리케이션에 제출된 요청을 말하며, 이 요청은 애플리케이션이 비동기적으로 실행되어 완료 시점을 추적합니다. 작업의 예로는 Apache Hive 응용 프로그램에 제출하는 HiveQL 쿼리나 PySpark Apache Spark 응용 프로그램에 제출하는 데이터 처리 스크립트가 있습니다. 작업을 제출할 때는 작업이 액세스하는 데 사용할 작성된 런타임 역할을 지정해야 합니다. IAM AWS 리소스 (예: Amazon S3 객체) 애플리케이션에 여러 작업 실행 요청을 제출할 수 있으며, 각 작업 실행은 다른 런타임 역할을 사용하여 액세스할 수 있습니다. AWS 있습니다. EMR서버리스 애플리케이션은 작업을 받는 즉시 실행을 시작하고 여러 작업 요청을 동시에 실행합니다. EMR서버리스에서 작업을 실행하는 방법에 대한 자세한 내용은 을 참조하십시오. 작업 실행

작업자

EMR서버리스 애플리케이션은 내부적으로 워커를 사용하여 워크로드를 실행합니다. 이러한 워커의 기본 크기는 애플리케이션 유형 및 Amazon EMR 릴리스 버전을 기반으로 합니다. 작업 실행을 예약할 때 이러한 크기를 재정의할 수 있습니다.

작업을 제출하면 EMR 서버리스는 애플리케이션이 작업에 필요한 리소스를 계산하고 작업자를 예약합니다. EMR서버리스는 워크로드를 작업으로 분류하고, 이미지를 다운로드하고, 작업자를 공급 및 설정하고, 작업이 완료되면 작업을 종료합니다. EMR서버리스는 작업의 모든 단계에서 필요한 워크로드 및 병렬성을 기반으로 작업자를 자동으로 늘리거나 줄입니다. 이 자동 크기 조정을 사용하면 애플리케이션이 워크로드를 실행하는 데 필요한 작업자 수를 추정할 필요가 없습니다.

사전 초기화된 용량

EMR서버리스는 작업자가 초기화되고 몇 초 만에 응답할 수 있도록 사전 초기화된 용량 기능을 제공합니다. 이 용량은 애플리케이션을 위한 작업자 풀을 효과적으로 생성합니다. 각 애플리케이션에 대해 이 기능을 구성하려면 애플리케이션의 `initial-capacity` 파라미터를 설정하십시오. 사전 초기화된 용량을 구성하면 작업을 즉시 시작할 수 있으므로 반복적인 애플리케이션과 시간에 민감한 작업을 구현할 수 있습니다. 사전 초기화된 작업자에 대한 자세한 내용은 을 참조하십시오. 애플리케이션 구성

EMR스튜디오

EMRStudio는 EMR 서버리스 애플리케이션을 관리하는 데 사용할 수 있는 사용자 콘솔입니다. 처음 EMR 서버리스 애플리케이션을 만들 때 계정에 EMR Studio가 없는 경우 Studio가 자동으로 생성됩니다. Amazon EMR 콘솔에서 EMR Studio에 액세스하거나 ID 공급자 (IdP) 에서 ID 센터를 IAM 통해 IAM 페더레이션 액세스를 활성화할 수 있습니다. 이렇게 하면 사용자가 Amazon EMR 콘솔에 직접 액세스하지 않고도 Studio에 액세스하고 EMR 서버리스 애플리케이션을 관리할 수 있습니다. EMR 서버리스 애플리케이션이 EMR Studio와 연동되는 방식에 대해 자세히 알아보려면 및 을 참조하십시오Studio 콘솔에서 애플리케이션과 상호 작용하기 EMR. EMR스튜디오 콘솔에서 작업 실행

서버리스를 시작하기 위한 사전 요구 사항 EMR

주제

가입해 보세요. AWS 계정

가지고 있지 않은 경우 AWS 계정다음 단계를 완료하여 새로 만드세요.

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/등록 열기.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   가입할 때 AWS 계정, 그리고 AWS 계정 루트 사용자생성됩니다. 루트 사용자는 모두에 액세스할 수 있습니다. AWS 서비스 및 계정 내 리소스 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료되면 확인 이메일을 보냅니다. 언제라도 https://aws.amazon.com/로 이동하여 내 계정을 선택하여 현재 계정 활동을 확인하고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

가입한 후 AWS 계정보안을 유지하세요 AWS 계정 루트 사용자, 활성화 AWS IAM Identity Center일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 생성하십시오.

보안을 유지하세요. AWS 계정 루트 사용자

1. 에 로그인하기AWS Management Console루트 사용자를 선택하고 다음을 입력하여 계정 소유자로 등록하십시오. AWS 계정 이메일 주소. 다음 페이지에서 비밀번호를 입력합니다.

루트 사용자로 로그인하는 데 도움이 필요하면 에서 루트 사용자로 로그인을 참조하십시오. AWS 로그인 사용 설명서.

2. 루트 사용자에 대한 다단계 인증 (MFA) 을 켜십시오.

   지침은 다음을 위한 가상 MFA 장치 활성화를 참조하십시오. AWS 계정 사용 설명서의 루트 IAM 사용자 (콘솔).

관리자 액세스 권한이 있는 사용자 생성

1. IAMID 센터를 활성화합니다.

   지침은 활성화를 참조하십시오. AWS IAM Identity Center의 AWS IAM Identity Center 사용 설명서.

2. IAMID 센터에서 사용자에게 관리 액세스 권한을 부여하십시오.

   사용에 대한 자습서는 IAM Identity Center 디렉터리 ID 소스로 사용하려면 기본값으로 사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리의 AWS IAM Identity Center 사용자 가이드.

관리 액세스 권한이 있는 사용자로 로그인

- IAMIdentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 URL 로그인을 사용하십시오.

  IAMIdentity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 로그인을 참조하십시오. AWS포털에 접속할 수 있습니다. AWS 로그인 사용자 가이드.

추가 사용자에게 액세스 권한 할당

1. IAMIdentity Center에서 최소 권한 권한 적용 모범 사례를 따르는 권한 집합을 생성하십시오.

   지침은 에서 권한 집합 만들기를 참조하십시오. AWS IAM Identity Center 사용 설명서.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   자세한 지침은 그룹 추가를 참조하십시오. AWS IAM Identity Center 사용 설명서.

권한 부여

프로덕션 환경에서는 더 세밀한 정책을 사용하는 것이 좋습니다. 이러한 정책의 예는 을 참조하십시오서버리스의 사용자 액세스 정책 예제 EMR. 액세스 관리에 대한 자세한 내용은 액세스 관리를 참조하십시오. AWS IAM사용 설명서의 리소스.

샌드박스 환경에서 EMR 서버리스를 시작해야 하는 사용자의 경우 다음과 유사한 정책을 사용하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRStudioCreate",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:CreateStudioPresignedUrl",
                "elasticmapreduce:DescribeStudio",
                "elasticmapreduce:CreateStudio",
                "elasticmapreduce:ListStudios"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EMRServerlessFullAccess",
            "Effect": "Allow",
            "Action": [
                "emr-serverless:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowEC2ENICreationWithEMRTags",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
```

```
                }
            }
        },
        {
            "Sid": "AllowEMRServerlessServiceLinkedRoleCreation",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/*"
        }
    ]
}
```

액세스 권한을 제공하려면 사용자, 그룹 또는 역할에 권한을 추가하세요:

- 다음 사용자 및 그룹 AWS IAM Identity Center:

  권한 세트를 생성합니다. 의 권한 집합 만들기의 지침을 따르세요. AWS IAM Identity Center 사용 설명서.

- ID 공급자를 IAM 통해 관리되는 사용자:

  ID 페더레이션을 위한 역할을 생성합니다. 사용 IAM설명서의 타사 ID 공급자 (페더레이션) 를 위한 역할 생성의 지침을 따르세요.

- IAM사용자:
  - 사용자가 맡을 수 있는 역할을 생성합니다. 사용 설명서의 IAM사용자 역할 생성에 나와 있는 지침을 따르십시오. IAM
  - (권장되지 않음)정책을 사용자에게 직접 연결하거나 사용자를 사용자 그룹에 추가합니다. 사용 설명서의 사용자 (콘솔) 에 권한 추가의 IAM 지침을 따르십시오.

프로그래밍 방식 액세스 권한 부여

사용자와 상호 작용하려면 프로그래밍 방식의 액세스가 필요합니다. AWS 외부 AWS Management Console. 프로그래밍 방식의 액세스 권한을 부여하는 방법은 액세스하는 사용자 유형에 따라 다릅니다. AWS.

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
작업 인력 ID (IAMID 센터에서 사용자 관리)	임시 자격 증명을 사용하여 프로그래밍 방식 요청에 서명할 수 있습니다. AWS CLI, AWS SDKs, 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 를 위해 AWS CLI구성을 참조하십시오. AWS CLI 사용하려면 AWS IAM Identity Center의 AWS Command Line Interface 사용자 가이드. • ... 용 AWS SDKs, 도구 및 AWS APIs, 의 IAMID 센터 인증을 참조하십시오. AWS SDKs및 도구 참조 가이드.
IAM	임시 자격 증명을 사용하여 프로그래밍 방식 요청에 서명할 수 있습니다. AWS CLI, AWS SDKs, 또는 AWS APIs.	임시 자격 증명 사용의 지침에 따라 AWSIAM사용 설명서의 리소스.
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 프로그래밍 방식 요청에 서명할 수 있습니다. AWS CLI, AWS SDKs, 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 를 위해 AWS CLIIAM사용자 자격 증명을 사용한 인증을 참조하십시오. AWS Command Line Interface 사용 설명서. • ... 용 AWS SDKs및 도구에 대한 자세한 내용은 장기 자격 증명을 사용한 인증을 참조하십시오. AWS SDKs및 도구 참조 가이드.

프로그래밍 방식 액세스가 필요한 사용자는 누구인가요?	To	액세스 권한을 부여하는 사용자
		• 에 대한 AWS APIs사용 설명서의 IAM사용자 액세스 키 관리를 참조하십시오. IAM

설치 및 구성 AWS CLI

EMRAPIs서버리스를 사용하려면 최신 버전의 서버를 설치해야 합니다. AWS Command Line Interface (AWS CLI). 필요 없어요 AWS CLI EMRStudio 콘솔에서 EMR 서버리스를 사용하려면 의 CLI 단계를 따르지 않고도 시작할 수 있습니다. 콘솔에서 EMR 서버리스 시작하기

설정하려면 AWS CLI

1. 최신 버전을 설치하려면 AWS CLI macOS, Linux 또는 Windows의 경우 최신 버전 설치 또는 업데이트를 참조하십시오. AWS CLI.
2. 구성하려면 AWS CLI 그리고 액세스의 보안 설정 AWS 서비스 EMR서버리스를 포함한 자세한 내용은 빠른 구성을 참조하십시오. `aws configure`
3. 설정을 확인하려면 명령 프롬프트에 다음 DataBrew 명령을 입력합니다.

   ```
   aws emr-serverless help
   ```

   AWS CLI 명령은 기본값을 사용합니다. AWS 리전 매개 변수나 프로필로 설정하지 않는 한 구성에서. 설정하려면 AWS 리전 파라미터를 사용하여 각 명령에 `--region` 파라미터를 추가할 수 있습니다.

   설정하려면 AWS 리전 프로필을 사용하여 먼저 `~/.aws/config` 파일 또는 파일에 이름이 지정된 프로필을 추가합니다 (Microsoft Windows의 `%UserProfile%/.aws/config` 경우). 명명된 프로필의 단계를 따르십시오. AWS CLI. 다음으로 설정하세요 AWS 리전 및 기타 설정은 다음 예제와 유사한 명령을 사용하여 설정합니다.

   ```
   [profile emr-serverless]
   aws_access_key_id = ACCESS-KEY-ID-OF-IAM-USER
   aws_secret_access_key = SECRET-ACCESS-KEY-ID-OF-IAM-USER
   region = us-east-1
   output = text
   ```

콘솔 열기

이 섹션의 콘솔 관련 주제 대부분은 Amazon 콘솔에서 시작됩니다. EMR 아직 로그인하지 않은 경우 AWS 계정로그인한 다음 Amazon EMR 콘솔을 열고 다음 섹션으로 계속 진행하여 Amazon을 계속 시작하십시오EMR.

Amazon EMR 서버리스 시작하기

이 자습서는 샘플 Spark 또는 Hive 워크로드를 배포할 때 EMR 서버리스를 시작하는 데 도움이 됩니다. 자체 애플리케이션을 만들고, 실행하고, 디버깅하게 됩니다. 이 자습서의 대부분의 부분에서 기본 옵션을 보여줍니다.

EMR서버리스 애플리케이션을 시작하기 전에 다음 작업을 완료하십시오.

주제

- 서버리스를 사용할 EMR 수 있는 권한을 부여하십시오.
- EMR서버리스용 스토리지 준비
- EMRStudio를 생성하여 대화형 워크로드를 실행합니다.
- 작업 런타임 역할 생성
- 콘솔에서 EMR 서버리스 시작하기
- 부터 시작하기 AWS CLI

서버리스를 사용할 EMR 수 있는 권한을 부여하십시오.

EMR서버리스를 사용하려면 서버리스에 대한 EMR 권한을 부여하는 정책이 연결된 사용자 또는 IAM 역할이 필요합니다. 사용자를 생성하고 해당 사용자에게 적절한 정책을 연결하려면 의 지침을 따르십시오. 권한 부여

EMR서버리스용 스토리지 준비

이 자습서에서는 S3 버킷을 사용하여 서버리스 애플리케이션을 사용하여 실행할 샘플 Spark 또는 Hive 워크로드의 출력 파일과 로그를 저장합니다. EMR 버킷을 생성하려면 Amazon 심플 스토리지 서비스 콘솔 사용 설명서의 버킷 생성의 지침을 따르십시오. 에 대한 추가 참조는 새로 생성한 버킷의 *DOC-EXAMPLE-BUCKET* 이름으로 바꾸십시오.

EMRStudio를 생성하여 대화형 워크로드를 실행합니다.

EMRStudio에서 호스팅되는 노트북을 통해 EMR 서버리스를 사용하여 대화형 쿼리를 실행하려면 S3 버킷과 EMR서버리스가 Workspace를 생성할 수 있는 최소 서비스 역할을 지정해야 합니다. 설정 단계는 Amazon EMR 관리 가이드의 EMR 스튜디오 설정을 참조하십시오. 대화형 워크로드에 대한 자세한 내용은 을 참조하십시오Studio를 통해 EMR 서버리스로 대화형 워크로드 실행 EMR.

작업 런타임 역할 생성

EMRServerless에서 작업을 실행하면 특정 사용자에게 세분화된 권한을 제공하는 런타임 역할을 사용합니다. AWS 서비스 및 런타임 시 리소스. 이 자습서에서는 퍼블릭 S3 버킷이 데이터와 스크립트를 호스팅합니다. 버킷은 출력을 *DOC-EXAMPLE-BUCKET* 저장합니다.

작업 런타임 역할을 설정하려면 먼저 EMR 서버리스가 새 역할을 사용할 수 있도록 신뢰 정책이 포함된 런타임 역할을 생성합니다. 그런 다음 필요한 S3 액세스 정책을 해당 역할에 연결합니다. 다음 단계는 프로세스를 안내합니다.

Console

1. IAM 콘솔(https://console.aws.amazon.com/iam/)로 이동합니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할 생성을 선택합니다.
4. 역할 유형에서 사용자 지정 신뢰 정책을 선택하고 다음 신뢰 정책을 붙여넣습니다. 이를 통해 Amazon EMR 서버리스 애플리케이션에 제출한 작업은 다른 애플리케이션에 액세스할 수 있습니다. AWS 서비스 귀하를 대신하여.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

5. 다음을 선택하여 권한 추가 페이지로 이동한 다음 정책 생성을 선택합니다.
6. 새 탭에 정책 생성 페이지가 열립니다. JSON아래에 정책을 붙여넣습니다.

> **Important**
>
> 아래 정책을 *DOC-EXAMPLE-BUCKET* 에서 생성한 실제 버킷 이름으로 EMR서버리스용 스토리지 준비 바꾸십시오. 이는 S3 액세스를 위한 기본 정책입니다. 작업 런타임

역할 예제에 대한 자세한 내용은 을 참조하십시오Amazon EMR 서버리스의 Job 런타임 역할.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadAccessForEMRSamples",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::*.elasticmapreduce",
                "arn:aws:s3:::*.elasticmapreduce/*"
            ]
        },
        {
            "Sid": "FullAccessToOutputBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        },
        {
            "Sid": "GlueCreateAndReadDataCatalog",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:CreateDatabase",
                "glue:GetDataBases",
                "glue:CreateTable",
                "glue:GetTable",
```

```
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:GetTables",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetUserDefinedFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

7. 정책 검토 페이지에서 정책 이름 (예:) 을 입력합니다EMRServerlessS3AndGlueAccessPolicy.
8. 연결 권한 정책 페이지를 새로 고치고 선택합니다EMRServerlessS3AndGlueAccessPolicy.
9. 이름, 검토 및 생성 페이지에서 역할 이름에 역할 이름 (예:) 을 입력합니다EMRServerlessS3RuntimeRole. 이 IAM 역할을 만들려면 역할 생성을 선택합니다.

CLI

1. IAM역할에 사용할 신뢰 정책이 emr-serverless-trust-policy.json 포함된 이름을 가진 파일을 만드십시오. 파일에는 다음 정책이 포함되어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "EMRServerlessTrustPolicy",
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
            "Service": "emr-serverless.amazonaws.com"
        }
    }]
}
```

2. 라는 IAM 역할을 생성합니다EMRServerlessS3RuntimeRole. 이전 단계에서 만든 신뢰 정책을 사용합니다.

```
aws iam create-role \
    --role-name EMRServerlessS3RuntimeRole \
    --assume-role-policy-document file://emr-serverless-trust-policy.json
```

출력의 ARN를 참고합니다. 작업을 제출하는 동안 새 ARN 역할의 를 사용합니다. 이를 다음으로는 이라고 *job-role-arn* 합니다.

3. 워크로드에 대한 IAM 정책을 `emr-sample-access-policy.json` 정의하는 이름을 가진 파일을 생성하십시오. 이렇게 하면 퍼블릭 S3 버킷에 저장된 스크립트 및 데이터에 대한 읽기 액세스와 읽기-쓰기 액세스가 제공됩니다. *DOC-EXAMPLE-BUCKET*

> **Important**
>
> 아래 정책을 *DOC-EXAMPLE-BUCKET* 에서 생성한 실제 버킷 이름으로 바꾸십시오. EMR서버리스용 스토리지 준비 이 정책은 다음과 같은 기본 정책입니다. AWS Glue 및 S3 액세스. 작업 런타임 역할 예제에 대한 자세한 내용은 을 참조하십시오Amazon EMR 서버리스의 Job 런타임 역할.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadAccessForEMRSamples",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::*.elasticmapreduce",
                "arn:aws:s3:::*.elasticmapreduce/*"
            ]
        },
        {
            "Sid": "FullAccessToOutputBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
```

```
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        },
        {
            "Sid": "GlueCreateAndReadDataCatalog",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:CreateDatabase",
                "glue:GetDataBases",
                "glue:CreateTable",
                "glue:GetTable",Understanding default application behavior,
 including auto-start and auto-stop, as well as maximum capacity and worker
 configurations for configuring an application with &EMRServerless;.
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:GetTables",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetUserDefinedFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

4. 3단계에서 만든 정책 파일을 EMRServerlessS3AndGlueAccessPolicy 사용하여 이름을 지정한 정책을 생성합니다. IAM 다음 단계에서 새 정책을 사용할 것이므로 출력에서 를 기록해 두십시오. ARN ARN

```
aws iam create-policy \
    --policy-name EMRServerlessS3AndGlueAccessPolicy \
    --policy-document file://emr-sample-access-policy.json
```

출력에 새 정책이 ARN 표시되는지 확인하십시오. 다음 단계에서 이를 *policy-arn* 대체해 보겠습니다.

5. IAM정책을 EMRServerlessS3AndGlueAccessPolicy 작업 런타임 역할에 연결합니다EMRServerlessS3RuntimeRole.

```
aws iam attach-role-policy \
    --role-name EMRServerlessS3RuntimeRole \
    --policy-arn policy-arn
```

콘솔에서 EMR 서버리스 시작하기

완료할 단계

- 1단계: EMR 서버리스 애플리케이션 생성
- 2단계: 작업 실행 또는 대화형 워크로드 제출
- 3단계: 애플리케이션 UI 및 로그 보기
- 4단계: 정리

1단계: EMR 서버리스 애플리케이션 생성

다음과 같이 EMR 서버리스로 새 애플리케이션을 생성합니다.

1. 에 로그인하십시오. AWS Management Console https://console.aws.amazon.com/emr에서 아마존 EMR 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 [EMRServerless] 를 선택하여 서버리스 랜딩 페이지로 이동합니다. EMR
3. EMR서버리스 애플리케이션을 만들거나 관리하려면 Studio UI가 필요합니다. EMR
   - EMR스튜디오가 이미 설치된 경우 AWS 리전 애플리케이션을 만들려는 위치에서 애플리케이션 관리를 선택하여 EMR 스튜디오로 이동하거나 사용할 스튜디오를 선택합니다.
   - EMR스튜디오가 없는 경우 AWS 리전 애플리케이션을 생성하려는 경우 시작하기를 선택한 다음 [Studio 생성 및 시작] 을 선택합니다. EMR서버리스는 사용자가 애플리케이션을 만들고 관리할 수 있도록 EMR Studio를 자동으로 생성합니다.
4. 새 탭에서 열리는 Create studio UI에서 애플리케이션의 이름, 유형 및 릴리스 버전을 입력합니다. 일괄 작업만 실행하려면 일괄 작업에만 기본 설정 사용을 선택합니다. 대화형 워크로드의 경우 대

화형 워크로드에 기본 설정 사용을 선택합니다. 이 옵션을 사용하여 대화형 지원 애플리케이션에서 일괄 작업을 실행할 수도 있습니다. 필요한 경우 나중에 이러한 설정을 변경할 수 있습니다.

자세한 내용은 스튜디오 만들기를 참조하십시오.

5. 애플리케이션 생성을 선택하여 첫 번째 애플리케이션을 생성합니다.

다음 섹션으로 계속 2단계: 작업 실행 또는 대화형 워크로드 제출 진행하여 작업 실행 또는 대화형 워크로드를 제출하세요.

2단계: 작업 실행 또는 대화형 워크로드 제출

Spark job run

이 자습서에서는 PySpark 스크립트를 사용하여 여러 텍스트 파일에서 나타나는 고유한 단어 수를 계산합니다. 공개 읽기 전용 S3 버킷은 스크립트와 데이터세트를 모두 저장합니다.

Spark 작업을 실행하려면

1. 다음 명령을 사용하여 샘플 스크립트를 `wordcount.py` 새 버킷에 업로드합니다.

```
aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
scripts/wordcount.py s3://DOC-EXAMPLE-BUCKET/scripts/
```

2. 1단계: EMR 서버리스 애플리케이션 생성완료하면 EMR Studio의 애플리케이션 세부 정보 페이지로 이동합니다. 거기에서 작업 제출 옵션을 선택합니다.
3. 작업 제출 페이지에서 다음을 완료하십시오.

   - 이름 필드에 작업 실행이라고 부르려는 이름을 입력합니다.
   - 런타임 역할 필드에 생성한 역할의 이름을 입력합니다작업 런타임 역할 생성.
   - 스크립트 위치 필드에 `s3://DOC-EXAMPLE-BUCKET/scripts/wordcount.py` S3로 입력합니다URI.
   - 스크립트 인수 필드에 를 입력합니다`["s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/output"]`.
   - Spark 속성 섹션에서 텍스트로 편집을 선택하고 다음 구성을 입력합니다.

```
--conf spark.executor.cores=1 --conf spark.executor.memory=4g --
conf spark.driver.cores=1 --conf spark.driver.memory=4g --conf
 spark.executor.instances=1
```

4. 작업 실행을 시작하려면 작업 제출을 선택합니다.
5. Job run (작업 실행) 탭에서 새 작업 실행이 [실행 중] 상태인 것을 확인할 수 있습니다.

Hive job run

자습서의 이 부분에서는 테이블을 만들고, 몇 개의 레코드를 삽입하고, 개수 집계 쿼리를 실행합니다. Hive 작업을 실행하려면 먼저 단일 작업의 일부로 실행할 모든 Hive 쿼리를 포함하는 파일을 만들고 S3에 파일을 업로드한 다음 Hive 작업을 시작할 때 이 S3 경로를 지정하십시오.

Hive 작업을 실행하려면

1. Hive 작업에서 실행하려는 모든 쿼리가 들어 `hive-query.ql` 있는 파일을 만드십시오.

```
create database if not exists emrserverless;
use emrserverless;
create table if not exists test_table(id int);
drop table if exists Values__Tmp__Table__1;
insert into test_table values (1),(2),(2),(3),(3),(3);
select id, count(id) from test_table group by id order by id desc;
```

2. 다음 명령을 `hive-query.ql` 사용하여 S3 버킷에 업로드합니다.

```
aws s3 cp hive-query.ql s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql
```

3. 1단계: EMR 서버리스 애플리케이션 생성완료하면 EMR Studio의 애플리케이션 세부 정보 페이지로 이동합니다. 거기에서 작업 제출 옵션을 선택합니다.
4. 작업 제출 페이지에서 다음을 완료하십시오.
   - 이름 필드에 작업 실행이라고 부르려는 이름을 입력합니다.
   - 런타임 역할 필드에 생성한 역할의 이름을 입력합니다작업 런타임 역할 생성.
   - 스크립트 위치 필드에 `s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql` S3로 입력합니다URI.
   - Hive 속성 섹션에서 텍스트로 편집을 선택하고 다음 구성을 입력합니다.

     ```
     --hiveconf hive.log.explain.output=false
     ```

   - [Job 구성] 섹션에서 [Edit asJSON] 를 선택하고 다음을 입력합니다JSON.

```
{
   "applicationConfiguration":
   [{
       "classification": "hive-site",
          "properties": {
              "hive.exec.scratchdir": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-
hive/hive/scratch",
              "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/warehouse",
              "hive.driver.cores": "2",
              "hive.driver.memory": "4g",
              "hive.tez.container.size": "4096",
              "hive.tez.cpu.vcores": "1"
           }
    }]
}
```

5. 작업 실행을 시작하려면 작업 제출을 선택합니다.
6. Job run (작업 실행) 탭에서 새 작업 실행이 [실행 중] 상태인 것을 확인할 수 있습니다.

Interactive workload

Amazon EMR 6.14.0 이상에서는 EMR Studio에서 호스팅되는 노트북을 사용하여 서버리스에서 Spark용 대화형 워크로드를 실행할 수 있습니다. EMR 권한 및 사전 요구 사항을 포함한 자세한 내용은 을 참조하십시오. Studio를 통해 EMR 서버리스로 대화형 워크로드 실행 EMR

애플리케이션을 만들고 필요한 권한을 설정한 후에는 다음 단계를 사용하여 Studio에서 대화형 전자 필기장을 실행하십시오. EMR

1. EMRStudio의 워크스페이스 탭으로 이동합니다. Amazon S3 스토리지 위치 및 EMR스튜디오 서비스 역할을 계속 구성해야 하는 경우 화면 상단의 배너에서 스튜디오 구성 버튼을 선택합니다.
2. 노트북에 액세스하려면 워크스페이스를 선택하거나 새 워크스페이스를 생성하십시오. 빠른 실행을 사용하여 새 탭에서 워크스페이스를 열 수 있습니다.
3. 새로 열린 탭으로 이동합니다. 왼쪽 탐색 메뉴에서 Compute 아이콘을 선택합니다. 컴퓨팅 유형으로 EMR 서버리스를 선택합니다.
4. 이전 섹션에서 만든 대화형 지원 애플리케이션을 선택합니다.

5. 런타임 역할 필드에 EMR 서버리스 애플리케이션이 작업 실행을 위해 맡을 수 있는 IAM 역할 이름을 입력합니다. 런타임 역할에 대해 자세히 알아보려면 Amazon EMR Serverless 사용 설명서의 Job 런타임 역할을 참조하십시오.
6. 연결을 선택합니다. 최대 1분이 소요될 수 있습니다. 첨부하면 페이지가 새로 고쳐집니다.
7. 커널을 선택하고 노트북을 시작합니다. EMR서버리스에서 예제 노트북을 찾아보고 작업 공간에 복사할 수도 있습니다. 예제 노트북에 액세스하려면 왼쪽 탐색의 **{...}**메뉴로 이동하여 노트북 파일 이름에 있는 `serverless` 노트북을 검색하십시오.
8. 노트북에서 드라이버 로그 링크와 Apache Spark UI에 대한 링크에 액세스할 수 있습니다. Apache Spark UI는 작업 모니터링을 위한 메트릭을 제공하는 실시간 인터페이스입니다. 자세한 내용은 Amazon EMR EMR Serverless 사용 설명서의 서버리스 애플리케이션 및 작업 모니터링을 참조하십시오.

애플리케이션을 Studio 작업 영역에 연결하면 아직 실행 중이 아닌 경우 애플리케이션 시작이 자동으로 트리거됩니다. 응용 프로그램을 미리 시작하고 작업 영역에 연결하기 전에 준비된 상태로 유지할 수도 있습니다.

3단계: 애플리케이션 UI 및 로그 보기

애플리케이션 UI를 보려면 먼저 작업 실행을 식별해야 합니다. Spark UI 또는 Hive Tez UI 옵션은 작업 유형에 따라 해당 작업 실행 옵션의 첫 번째 행에서 사용할 수 있습니다. 적절한 옵션을 선택합니다.

Spark UI를 선택한 경우 Executors 탭을 선택하여 드라이버 및 실행기 로그를 볼 수 있습니다. Hive Tez UI를 선택한 경우 모든 작업 탭을 선택하여 로그를 확인하십시오.

작업 실행 상태가 성공으로 표시되면 S3 버킷에서 작업의 출력을 볼 수 있습니다.

4단계: 정리

생성한 애플리케이션은 15분 동안 사용하지 않으면 자동으로 중지되지만, 다시 사용하지 않을 리소스는 릴리스하는 것이 좋습니다.

애플리케이션을 삭제하려면 애플리케이션 목록 페이지로 이동합니다. 생성한 애플리케이션을 선택하고 작업 → 중지를 선택하여 애플리케이션을 중지합니다. 응용 프로그램이 STOPPED 상태가 되면 동일한 응용 프로그램을 선택하고 작업 → 삭제를 선택합니다.

Spark 및 Hive 작업 실행에 대한 더 많은 예를 보려면 및 을 참조하십시오스파크 잡스. 하이브 잡스

부터 시작하기 AWS CLI

1단계: EMR 서버리스 애플리케이션 생성

emr-serverless create-application명령을 사용하여 첫 번째 EMR 서버리스 애플리케이션을 생성합니다. 사용하려는 애플리케이션 버전과 관련된 애플리케이션 유형 및 Amazon EMR 릴리스 라벨을 지정해야 합니다. 애플리케이션 이름은 선택 사항입니다.

Spark

Spark 애플리케이션을 생성하려면 다음 명령을 실행합니다.

```
aws emr-serverless create-application \
    --release-label emr-6.6.0 \
    --type "SPARK" \
    --name my-application
```

Hive

Hive 애플리케이션을 만들려면 다음 명령을 실행합니다.

```
aws emr-serverless create-application \
    --release-label emr-6.6.0 \
    --type "HIVE" \
    --name my-application
```

출력에 반환된 애플리케이션 ID를 기록해 둡니다. 이 ID를 사용하여 응용 프로그램을 시작하고 작업을 제출하는 동안 이 ID를 사용합니다. 이 ID를 사용하면 다음과 같습니다*application-id*.

다음으로 넘어가기 2단계: EMR 서버리스 애플리케이션에 작업 실행 제출 전에 신청서가 다음과 같은 CREATED 상태에 도달했는지 확인하십시오 get-applicationAPI.

```
aws emr-serverless get-application \
    --application-id application-id
```

EMR서버리스는 요청된 작업을 수용할 수 있는 작업자를 생성합니다. 기본적으로 요청 시 생성되지만 애플리케이션을 생성할 때 initialCapacity 매개변수를 설정하여 사전 초기화된 용량을 지정할 수도 있습니다. 또한 응용 프로그램이 매개 변수와 함께 사용할 수 있는 총 최대 용량을 제한할 수 있습니

다. `maximumCapacity` 이러한 옵션에 대해 자세히 알아보려면 애플리케이션 구성 단원을 참조하세요.

2단계: EMR 서버리스 애플리케이션에 작업 실행 제출

이제 EMR 서버리스 애플리케이션에서 작업을 실행할 준비가 되었습니다.

Spark

이 단계에서는 PySpark 스크립트를 사용하여 여러 텍스트 파일에서 나타나는 고유한 단어 수를 계산합니다. 공개 읽기 전용 S3 버킷은 스크립트와 데이터세트를 모두 저장합니다. 애플리케이션은 Spark 런타임의 출력 파일과 로그 데이터를 사용자가 생성한 S3 버킷의 `/output` 및 `/logs` 디렉터리로 전송합니다.

Spark 작업을 실행하려면

1. 다음 명령을 사용하여 실행할 샘플 스크립트를 새 버킷에 복사합니다.

```
aws s3 cp s3://us-east-1.elasticmapreduce/emr-containers/samples/wordcount/
scripts/wordcount.py s3://DOC-EXAMPLE-BUCKET/scripts/
```

2. 다음 명령에서 애플리케이션 *application-id* ID로 대체하십시오. 에서 ARN 생성한 런타임 *job-role-arn* 역할로 작업 런타임 역할 생성 대체하십시오. 대체 *job-run-name* 작업 실행을 호출하려는 이름을 사용하십시오. 모든 *DOC-EXAMPLE-BUCKET* 문자열을 생성한 Amazon S3 버킷으로 바꾸고 경로에 추가합니다`/output`. 그러면 EMR 서버리스가 애플리케이션의 출력 파일을 복사할 수 있는 새 폴더가 버킷에 생성됩니다.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --name job-run-name \
    --job-driver '{
        "sparkSubmit": {
          "entryPoint": "s3://DOC-EXAMPLE-BUCKET/scripts/wordcount.py",
          "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET/emr-serverless-spark/
output"],
          "sparkSubmitParameters": "--conf spark.executor.cores=1
 --conf spark.executor.memory=4g --conf spark.driver.cores=1 --conf
 spark.driver.memory=4g --conf spark.executor.instances=1"
        }
    }'
```

3. 출력에 반환된 작업 실행 ID를 기록해 둡니다. 다음 단계에서 이 *job-run-id* ID로 바꾸십시오.

Hive

이 자습서에서는 테이블을 만들고, 몇 개의 레코드를 삽입하고, 개수 집계 쿼리를 실행합니다. Hive 작업을 실행하려면 먼저 단일 작업의 일부로 실행할 Hive 쿼리를 모두 포함하는 파일을 만들고 S3에 파일을 업로드한 다음 Hive 작업을 시작할 때 이 S3 경로를 지정하십시오.

Hive 작업을 실행하려면

1. Hive 작업에서 실행하려는 모든 쿼리가 들어 `hive-query.ql` 있는 파일을 만드십시오.

```
create database if not exists emrserverless;
use emrserverless;
create table if not exists test_table(id int);
drop table if exists Values__Tmp__Table__1;
insert into test_table values (1),(2),(2),(3),(3),(3);
select id, count(id) from test_table group by id order by id desc;
```

2. 다음 명령을 `hive-query.ql` 사용하여 S3 버킷에 업로드합니다.

```
aws s3 cp hive-query.ql s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-
query.ql
```

3. 다음 명령에서 자체 애플리케이션 *application-id* ID로 대체하십시오. 에서 만든 런타임 *job-role-arn* ARN 역할로 작업 런타임 역할 생성 대체하십시오. 모든 *DOC-EXAMPLE-BUCKET* 문자열을 생성한 Amazon S3 버킷으로 바꾸고 경로에 `/output` 와 `/logs` 를 추가합니다. 그러면 버킷에 새 폴더가 생성되며, EMR 서버리스에서 애플리케이션의 출력 및 로그 파일을 복사할 수 있습니다.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
          "query": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-
query.ql",
          "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
```

```
    --configuration-overrides '{
      "applicationConfiguration": [{
        "classification": "hive-site",
          "properties": {
            "hive.exec.scratchdir": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-
hive/hive/scratch",
            "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/warehouse",
            "hive.driver.cores": "2",
            "hive.driver.memory": "4g",
            "hive.tez.container.size": "4096",
            "hive.tez.cpu.vcores": "1"
            }
        }],
        "monitoringConfiguration": {
          "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/logs"
           }
        }
    }'
```

4. 출력에 반환된 작업 실행 ID를 기록해 둡니다. 다음 단계에서 이 *job-run-id* ID로 바꾸십시오.

3단계: 작업 실행 결과 검토

작업 실행은 일반적으로 완료하는 데 3-5분 정도 걸립니다.

Spark

다음 명령을 사용하여 Spark 작업의 상태를 확인할 수 있습니다.

```
aws emr-serverless get-job-run \
    --application-id application-id \
    --job-run-id job-run-id
```

로그 대상을 로 설정하면 에서 s3://*DOC-EXAMPLE-BUCKET*/emr-serverless-spark/logs 이 특정 작업 실행에 대한 로그를 찾을 수 있습니다. s3://*DOC-EXAMPLE-BUCKET*/emr-serverless-spark/logs/applications/*application-id*/jobs/*job-run-id*

Spark 애플리케이션의 경우 EMR 서버리스는 30초마다 이벤트 로그를 S3 로그 대상의 sparklogs 폴더로 푸시합니다. 작업이 완료되면 드라이버 및 실행기의 Spark 런타임 로그가 작업

자 유형별로 적절한 이름을 가진 폴더 (예: 또는) 에 업로드됩니다. driver executor 작업 출력은 에 업로드됩니다. PySpark s3://*DOC-EXAMPLE-BUCKET*/output/

Hive

다음 명령을 사용하여 Hive 작업의 상태를 확인할 수 있습니다.

```
aws emr-serverless get-job-run \
    --application-id application-id \
    --job-run-id job-run-id
```

로그 대상을 로 설정하면 에서 s3://*DOC-EXAMPLE-BUCKET*/emr-serverless-hive/logs 이 특정 작업 실행에 대한 로그를 찾을 수 있습니다. s3://*DOC-EXAMPLE-BUCKET*/emr-serverless-hive/logs/applications/*application-id*/jobs/*job-run-id*

Hive 애플리케이션의 경우 EMR 서버리스는 Hive 드라이버를 S3 로그 대상의 HIVE_DRIVER 폴더에 지속적으로 업로드하고 Tez 작업 로그는 S3 로그 대상의 TEZ_TASK 폴더에 업로드합니다. 작업 실행이 SUCCEEDED 상태에 도달하면 monitoringConfiguration 필드에 지정한 Amazon S3 위치에서 Hive 쿼리의 출력을 사용할 수 있게 됩니다. configurationOverrides

4단계: 정리

이 자습서 작업을 마치면 생성한 리소스를 삭제해 보십시오. 다시 사용하지 않을 리소스는 릴리스하는 것이 좋습니다.

애플리케이션 삭제

애플리케이션을 삭제하려면 다음 명령을 사용합니다.

```
aws emr-serverless delete-application \
    --application-id application-id
```

S3 로그 버킷을 삭제합니다.

S3 로깅 및 출력 버킷을 삭제하려면 다음 명령을 사용합니다. 에서 생성된 S3 버킷의 실제 *DOC-EXAMPLE-BUCKET* 이름으로 바꾸십시오EMR서버리스용 스토리지 준비.

```
aws s3 rm s3://DOC-EXAMPLE-BUCKET --recursive
aws s3api delete-bucket --bucket DOC-EXAMPLE-BUCKET
```

작업 런타임 역할 삭제

런타임 역할을 삭제하려면 역할에서 정책을 분리하십시오. 그런 다음 역할과 정책을 모두 삭제할 수 있습니다.

```
aws iam detach-role-policy \
    --role-name EMRServerlessS3RuntimeRole \
    --policy-arn policy-arn
```

역할을 삭제하려면 다음 명령을 사용합니다.

```
aws iam delete-role \
    --role-name EMRServerlessS3RuntimeRole
```

역할에 연결된 정책을 삭제하려면 다음 명령을 사용합니다.

```
aws iam delete-policy \
    --policy-arn policy-arn
```

Spark 및 Hive 작업 실행에 대한 추가 예제는 및 을 참조하십시오스파크 잡스. 하이브 잡스

애플리케이션과의 상호 작용

이 섹션에서는 다음을 사용하여 Amazon EMR 서버리스 애플리케이션과 상호 작용하는 방법을 다룹니다. AWS CLI 그리고 Spark 및 Hive 엔진의 기본값도 포함됩니다.

주제

애플리케이션 상태

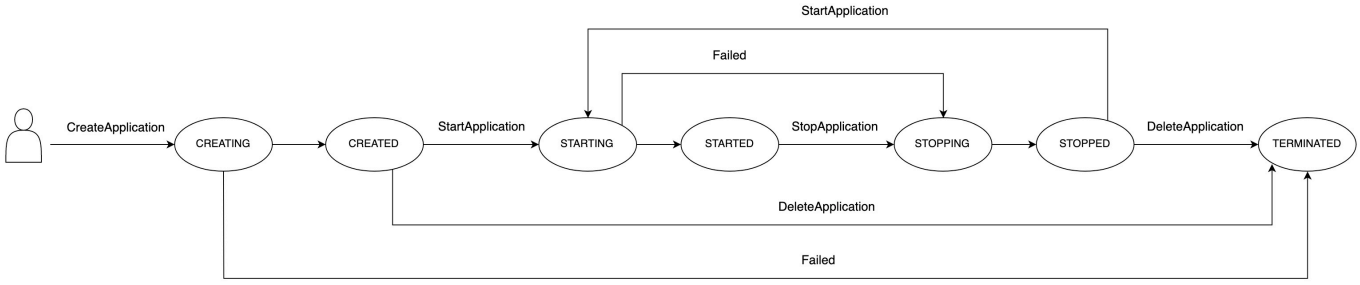
EMR서버리스로 애플리케이션을 생성하면 애플리케이션 실행이 해당 상태로 전환됩니다. `CREATING` 그런 다음 작업은 성공(`0` 코드로 종료)하거나 실패(0이 아닌 코드로 종료)할 때까지 다음 상태를 통과합니다.

애플리케이션은 다음과 같은 상태를 가질 수 있습니다.

State	설명
[생성 중]	응용 프로그램이 준비 중이며 아직 사용할 준비가 되지 않았습니다.
Created	애플리케이션이 생성되었지만 아직 용량이 프로비저닝되지 않았습니다. 애플리케이션을 수정하여 초기 용량 구성을 변경할 수 있습니다.
[시작됨]	애플리케이션이 시작되고 용량을 프로비저닝하고 있습니다.

State	설명
시작됨	신청서가 새 작업을 수락할 준비가 되었습니다. 응용 프로그램은 이 상태일 때만 작업을 수락합니다.
[중지 중]	모든 작업이 완료되었으며 애플리케이션에서 용량을 확보하고 있습니다.
중지됨	애플리케이션이 중지되고 애플리케이션에서 실행 중인 리소스가 없습니다. 애플리케이션을 수정하여 초기 용량 구성을 변경할 수 있습니다.
종료됨	신청이 종료되어 지원서 목록에 표시되지 않습니다.

다음 다이어그램은 서버리스 애플리케이션 상태의 EMR 궤적을 보여줍니다.



Studio 콘솔에서 애플리케이션과 상호 작용하기 EMR

EMRStudio 콘솔에서 EMR 서버리스 애플리케이션을 만들고, 보고, 관리할 수 있습니다. EMRStudio 콘솔로 이동하려면 콘솔에서 시작하기의 지침을 따르십시오.

애플리케이션 생성

애플리케이션 생성 페이지에서 다음 단계에 따라 EMR 서버리스 애플리케이션을 생성할 수 있습니다.

1. 이름 필드에 애플리케이션을 호출하려는 이름을 입력합니다.
2. 유형 필드에서 애플리케이션 유형으로 Spark 또는 Hive를 선택합니다.

3. 출시 버전 필드에서 릴리스 번호를 선택합니다. EMR
4. 아키텍처 옵션에서 사용할 명령어 세트 아키텍처를 선택합니다. 자세한 내용은 Amazon EMR 서버리스 아키텍처 옵션 단원을 참조하십시오.

   - arm64 — 64비트 ARM 아키텍처, 그래비톤 프로세서 사용
   - x86_64 — 64비트 x86 아키텍처, x86 기반 프로세서 사용
5. 나머지 필드에는 기본 설정과 사용자 지정 설정이라는 두 가지 응용 프로그램 설치 옵션이 있습니다. 이 필드는 선택사항입니다.

   기본 설정 - 기본 설정을 통해 사전 초기화된 용량으로 애플리케이션을 빠르게 생성할 수 있습니다. 여기에는 Spark용 드라이버 하나와 실행기 하나, Hive용 드라이버 하나와 Tez Task 한 개가 포함됩니다. 기본 설정으로는 네트워크에 연결할 수 없습니다. VPCs 15분 동안 유휴 상태인 경우 응용 프로그램이 중지되고 작업 제출 시 자동으로 시작되도록 구성되어 있습니다.

   사용자 지정 설정 - 사용자 지정 설정을 통해 다음 속성을 수정할 수 있습니다.

   - 사전 초기화된 용량 - 드라이버 및 실행자 또는 Hive Tez Task 작업자 수, 각 작업자의 크기
   - 애플리케이션 제한 — 애플리케이션의 최대 용량.
   - 애플리케이션 동작 - 애플리케이션의 자동 시작 및 자동 중지 동작.
   - 네트워크 연결 - 리소스에 대한 네트워크 연결. VPC
   - 태그 - 애플리케이션에 할당할 수 있는 사용자 지정 태그입니다.

   사전 초기화된 용량, 애플리케이션 제한 및 애플리케이션 동작에 대한 자세한 내용은 을 참조하십시오. 애플리케이션 구성 네트워크 연결에 대한 자세한 내용은 을 참조하십시오. 액세스 구성 VPC
6. 애플리케이션을 생성하려면 애플리케이션 생성을 선택합니다.

애플리케이션 나열

애플리케이션 목록 페이지에서 기존의 모든 EMR 서버리스 애플리케이션을 볼 수 있습니다. 애플리케이션 이름을 선택하여 해당 애플리케이션의 세부 정보 페이지로 이동할 수 있습니다.

애플리케이션 관리

애플리케이션 목록 페이지 또는 특정 애플리케이션의 세부 정보 페이지에서 애플리케이션에 대해 다음 작업을 수행할 수 있습니다.

애플리케이션 시작

애플리케이션을 수동으로 시작하려면 이 옵션을 선택합니다.

애플리케이션 중지

애플리케이션을 수동으로 중지하려면 이 옵션을 선택합니다. 응용 프로그램을 중지하려면 실행 중인 작업이 없어야 합니다. 애플리케이션 상태 전환에 대한 자세한 내용은 을 참조하십시오애플리케이션 상태.

애플리케이션 구성

애플리케이션 구성 페이지에서 애플리케이션의 선택적 설정을 편집합니다. 대부분의 애플리케이션 설정을 변경할 수 있습니다. 예를 들어 애플리케이션의 릴리스 라벨을 변경하여 다른 버전의 EMR Amazon으로 업그레이드하거나 아키텍처를 x86_64에서 arm64로 전환할 수 있습니다. 다른 선택적 설정은 애플리케이션 생성 페이지의 사용자 지정 설정 섹션에 있는 설정과 동일합니다. 애플리케이션 설정에 대한 자세한 내용은 을 참조하십시오애플리케이션 생성.

애플리케이션 삭제

애플리케이션을 수동으로 삭제하려면 이 옵션을 선택합니다. 삭제하려면 애플리케이션을 중지해야 합니다. 애플리케이션 상태 전환에 대한 자세한 내용은 을 참조하십시오애플리케이션 상태.

에서 애플리케이션과 상호 작용하기 AWS CLI

에서 AWS CLI개별 애플리케이션을 생성, 설명 및 삭제할 수 있습니다. 한 눈에 볼 수 있도록 모든 애플리케이션을 나열할 수도 있습니다. 이 섹션에서는 이러한 작업을 수행하는 방법을 설명합니다. 애플리케이션 시작, 중지 및 업데이트와 같은 애플리케이션 작업에 대한 자세한 내용은 EMR서버리스 API 레퍼런스를 참조하십시오. 다음을 사용하여 EMR API 서버리스를 사용하는 방법에 대한 예는 AWS SDK for Java GitHub 리포지토리의 Java 예제를 참조하십시오. 다음을 API 사용하여 EMR 서버리스를 사용하는 방법에 대한 예는 다음과 같습니다. AWS SDK for Python (Boto), GitHub 저장소의 Python 예제를 참조하십시오.

애플리케이션을 만들려면 를 사용하십시오`create-application`. `SPARK`또는 를 `HIVE` 애플리케이션으로 지정해야 `type` 합니다. 이 명령은 애플리케이션ARN, 이름 및 ID를 반환합니다.

```
aws emr-serverless create-application \
--name my-application-name \
--type 'application-type' \
--release-label release-version
```

애플리케이션을 설명하려면 해당 애플리케이션을 get-application 사용하고 제공하십시오application-id. 이 명령은 애플리케이션의 상태 및 용량 관련 구성을 반환합니다.

```
aws emr-serverless get-application \
--application-id application-id
```

애플리케이션을 모두 나열하려면 를 호출하십시오. list-applications 이 명령은 모든 응용 프로그램과 동일한 속성을 get-application 반환하지만 모든 응용 프로그램을 포함합니다.

```
aws emr-serverless list-applications
```

애플리케이션을 삭제하려면 를 delete-application 호출하여 입력하십시오application-id.

```
aws emr-serverless delete-application \
--application-id application-id
```

애플리케이션 구성

EMR서버리스를 사용하면 사용하는 애플리케이션을 구성할 수 있습니다. 예를 들어 애플리케이션이 확장할 수 있는 최대 용량을 설정하고, 운전자와 작업자가 응답할 수 있도록 사전 초기화된 용량을 구성하고, 애플리케이션 수준에서 공통 런타임 및 모니터링 구성 세트를 지정할 수 있습니다. 다음 페이지는 서버리스를 사용할 EMR 때 애플리케이션을 구성하는 방법을 설명합니다.

주제

- 애플리케이션 동작 이해
- 사전 초기화된 용량
- 서버리스의 기본 애플리케이션 구성 EMR

애플리케이션 동작 이해

기본 애플리케이션 동작

자동 시작 - 응용 프로그램은 기본적으로 작업 제출 시 자동 시작되도록 구성되어 있습니다. 이 기능을 끌 수 있습니다.

자동 중지 - 응용 프로그램은 기본적으로 15분 동안 유휴 상태일 때 자동 중지되도록 구성되어 있습니다. 애플리케이션이 STOPPED 상태로 변경되면 사전 초기화된 구성된 용량을 모두 해제합니다. 응용 프로그램이 자동 중지되기 전의 유휴 시간을 수정하거나 이 기능을 끌 수 있습니다.

최대 용량

애플리케이션이 확장할 수 있는 최대 용량을 구성할 수 있습니다. 메모리 (GB) 및 디스크 (GB) 측면에서 CPU 최대 용량을 지정할 수 있습니다.

> **Note**
>
> 작업자 수에 작업자 크기를 곱하여 지원되는 작업자 크기에 비례하도록 최대 용량을 구성하는 것이 좋습니다. 예를 들어 메모리 2인 16GBvCPUs, 디스크 20GB인 작업자 50명으로 애플리케이션을 제한하려면 최대 용량을 메모리는 100vCPUs, 800GB, 디스크는 1000GB로 설정합니다.

지원되는 작업자 구성

다음 표에는 EMR 서버리스에 지정할 수 있는 지원되는 작업자 구성 및 크기가 나와 있습니다. 워크로드의 필요에 따라 다양한 크기의 드라이버와 실행기를 구성할 수 있습니다.

CPU	메모리	기본 임시 스토리지
1 v CPU	1GB 단위로 최소 2GB, 최대 8GB	20기가바이트 - 200기가바이트
2 v CPU	최소 4GB, 최대 16GB, 1GB 단위로 추가	20기가바이트 - 200기가바이트
4 v CPU	최소 8GB, 최대 30GB (1GB 단위로)	20기가바이트 - 200기가바이트
8 v CPU	최소 16GB, 최대 60GB (4GB 단위로)	20기가바이트 - 200기가바이트
16 v CPU	최소 32GB, 최대 120GB (8GB 단위로)	20기가바이트 - 200기가바이트

CPU— 각 작업자는 1, 2, 4, 8 또는 16을 가질 수 vCPUs 있습니다.

메모리 — 각 작업자는 위 표에 나열된 제한 내에서 GB 단위로 지정된 메모리를 가집니다. 스파크 작업에는 메모리 오버헤드가 발생합니다. 즉, 사용하는 메모리가 지정된 컨테이너 크기

보다 큽니다. 이 오버헤드는 속성과 함께 지정됩니다. spark.driver.memoryOverhead spark.executor.memoryOverhead 오버헤드의 기본값은 컨테이너 메모리의 10% 이며, 최소 384MB입니다. 작업자 크기를 선택할 때는 이 오버헤드를 고려해야 합니다.

예를 들어 작업자 인스턴스로 4개를 vCPUs 선택하고 사전 초기화된 스토리지 용량이 30GB인 경우 Spark 작업의 실행기 메모리 값을 약 27GB로 설정해야 합니다. 이렇게 하면 사전 초기화된 용량을 최대한 활용할 수 있습니다. 사용 가능한 메모리는 27GB에 27GB (2.7GB) 의 10% 를 더하여 총 29.7GB 가 됩니다.

디스크 - 최소 크기가 20GB에서 최대 200GB인 임시 스토리지 디스크로 각 작업자를 구성할 수 있습니다. 작업자당 구성한 20GB를 초과하는 추가 스토리지에 대한 비용만 지불하면 됩니다.

사전 초기화된 용량

EMR서버리스는 운전자와 작업자가 미리 초기화되어 몇 초 만에 응답할 수 있도록 하는 선택적 기능을 제공합니다. 이렇게 하면 애플리케이션을 위한 작업자 수가 많이 모이는 효과가 있습니다. 이 기능을 사전 초기화된 용량이라고 합니다. 이 기능을 구성하려면 응용 프로그램의 initialCapacity 매개변수를 사전 초기화하려는 작업자 수로 설정할 수 있습니다. 작업자 용량이 사전 초기화되면 작업이 즉시 시작됩니다. 이는 반복적인 애플리케이션과 시간에 민감한 작업을 구현하려는 경우에 적합합니다.

작업을 제출할 때 작업자를 사용할 수 initialCapacity 있는 경우 작업은 해당 리소스를 사용하여 실행을 시작합니다. 해당 작업자가 이미 다른 작업에서 사용 중이거나 작업에 필요한 리소스가 가용 양보다 많은 경우 응용 프로그램은 응용 프로그램에 설정된 최대 리소스 한도까지 추가 작업자를 요청하여 가져옵니다. initialCapacity 작업 실행이 끝나면 사용한 작업자가 해제되고 응용 프로그램에 사용할 수 있는 리소스 수가 반환됩니다. initialCapacity 응용 프로그램은 작업 실행이 initialCapacity 끝난 후에도 해당 리소스를 유지합니다. 애플리케이션은 작업을 실행하는 데 더 이상 필요하지 않은 시점 initialCapacity 이후에도 초과 리소스를 릴리스합니다.

사전 초기화된 용량을 사용할 수 있으며 애플리케이션이 시작되면 바로 사용할 수 있습니다. 애플리케이션이 중지되면 사전 초기화된 용량은 비활성화됩니다. 요청된 사전 초기화된 용량이 생성되어 사용할 준비가 된 경우에만 애플리케이션이 해당 STARTED 상태로 이동합니다. 애플리케이션이 STARTED 상태에 있는 동안 EMR 서버리스는 사전 초기화된 용량을 작업이나 대화형 워크로드에서 사용하거나 사용할 수 있도록 유지합니다. 이 기능은 출시되거나 장애가 발생한 컨테이너의 용량을 복원합니다. 이렇게 하면 InitialCapacity 매개변수에서 지정하는 작업자 수가 유지됩니다. 사전 초기화된 용량이 없는 애플리케이션의 상태는 에서 CREATED 로 즉시 변경될 수 있습니다. STARTED

일정 기간 (기본값 15분) 동안 사용되지 않을 경우 사전 초기화된 용량을 해제하도록 애플리케이션을 구성할 수 있습니다. 새 작업을 제출하면 중지된 응용 프로그램이 자동으로 시작됩니다. 이러한 자동

시작 및 중지 구성은 응용 프로그램을 만들 때 설정하거나 응용 프로그램이 CREATED 또는 STOPPED 상태일 때 변경할 수 있습니다.

InitialCapacity개수를 변경하고 각 워커의 컴퓨팅 구성 (예: 메모리CPU, 디스크) 을 지정할 수 있습니다. 부분적으로 수정할 수 없으므로 값을 변경할 때 모든 컴퓨팅 구성을 지정해야 합니다. 애플리케이션이 CREATED 또는 STOPPED 상태일 때만 구성을 변경할 수 있습니다.

> **Note**
>
> 애플리케이션의 리소스 사용을 최적화하려면 컨테이너 크기를 사전 초기화된 용량 작업자 크기에 맞추는 것이 좋습니다. 예를 들어 Spark 실행자 크기를 2로 CPUs 구성하고 메모리를 8GB로 구성했지만 사전 초기화된 용량 작업자 크기가 CPUs 4이고 메모리가 16GB인 경우 Spark 실행자는 이 작업에 배정될 때 작업자 리소스의 절반만 사용합니다.

Spark 및 Hive의 사전 초기화된 용량 사용자 지정

특정 빅데이터 프레임워크에서 실행되는 워크로드에 대해 사전 초기화된 용량을 추가로 사용자 지정할 수 있습니다. 예를 들어 Apache Spark에서 워크로드를 실행할 때 드라이버로 시작하는 작업자 수와 실행자로 시작하는 작업자 수를 지정할 수 있습니다. 마찬가지로 Apache Hive를 사용할 때도 Hive 드라이버로 시작하는 작업자 수와 Tez 작업을 실행할 작업자 수를 지정할 수 있습니다.

사전 초기화된 용량으로 Apache Hive를 실행하는 애플리케이션 구성

다음 API 요청은 Amazon EMR 릴리스 emr-6.6.0을 기반으로 Apache Hive를 실행하는 애플리케이션을 생성합니다. 애플리케이션은 각각 2v CPU 및 4GB의 메모리가 있는 사전 초기화된 Hive 드라이버 5개와 각각 4v 및 8GB의 메모리를 가진 사전 초기화된 Tez 태스크 워커 50명으로 시작합니다. CPU 이 애플리케이션에서 Hive 쿼리를 실행하면 먼저 사전 초기화된 작업자를 사용하고 즉시 실행을 시작합니다. 사전 초기화된 작업자가 모두 바쁘고 더 많은 Hive 작업이 제출된 경우 애플리케이션을 총 400v 및 1024GB까지 확장할 수 있습니다. CPU 선택적으로 작업자 또는 작업자의 용량을 생략할 수 있습니다. DRIVER TEZ_TASK

```
aws emr-serverless create-application \
  --type "HIVE" \
  --name my-application-name \
  --release-label emr-6.6.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
```

```
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB"
        }
    },
    "TEZ_TASK": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
  }'
```

사전 초기화된 용량으로 Apache Spark를 실행하는 애플리케이션 구성

다음 API 요청은 Amazon EMR 릴리스 6.6.0을 기반으로 Apache Spark 3.2.0을 실행하는 애플리케이션을 생성합니다. 애플리케이션은 각각 2v CPU 및 4GB의 메모리가 있는 사전 초기화된 Spark 드라이버 5개와 각각 4v 및 8GB 메모리가 있는 사전 초기화된 실행기 50개로 시작합니다. CPU 이 애플리케이션에서 Spark 작업을 실행하면 먼저 사전 초기화된 작업자를 사용하고 즉시 실행을 시작합니다. 사전 초기화된 작업자가 모두 바쁘고 더 많은 Spark 작업이 제출된 경우 애플리케이션을 총 400v CPU 및 1024GB까지 확장할 수 있습니다. 또는 의 용량을 생략할 수도 있습니다. `DRIVER EXECUTOR`

> **Note**
>
> Spark는 드라이버 및 실행기에 필요한 메모리에 10% 기본값으로 구성 가능한 메모리 오버헤드를 추가합니다. 작업에 사전 초기화된 작업자를 사용하려면 초기 용량 메모리 구성이 작업 및 오버헤드가 요청한 메모리보다 커야 합니다.

```
aws emr-serverless create-application \
  --type "SPARK" \
  --name my-application-name \
  --release-label emr-6.6.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
```

```
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB"
        }
    },
    "EXECUTOR": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB"
        }
    }
}' \
--maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
}'
```

서버리스의 기본 애플리케이션 구성 EMR

동일한 애플리케이션에서 제출하는 모든 작업에 대해 애플리케이션 수준에서 공통 런타임 및 모니터링 구성 세트를 지정할 수 있습니다. 이렇게 하면 각 작업에 대해 동일한 구성을 제출해야 하는 데 따르는 추가 오버헤드가 줄어듭니다.

다음과 같은 시점에 구성을 수정할 수 있습니다.

- 작업 제출 시 애플리케이션 수준 구성을 선언하십시오.
- 작업 실행 중에 기본 구성을 재정의합니다.

다음 섹션에서는 자세한 내용과 추가 컨텍스트를 위한 예를 제공합니다.

응용 프로그램 수준에서 구성 선언

애플리케이션에서 제출하는 작업에 대해 애플리케이션 수준 로깅 및 런타임 구성 속성을 지정할 수 있습니다.

`monitoringConfiguration`

응용 프로그램과 함께 제출하는 작업의 로그 구성을 지정하려면 필드를 사용하십시오. `monitoringConfiguration` EMR서버리스 로깅에 대한 자세한 내용은 을 참조하십시오로그 저장.

`runtimeConfiguration`

런타임 구성 속성 (예:) 을 지정하려면 `runtimeConfiguration` 필드에 구성 객체를 제공하십시오. `spark-defaults` 이는 애플리케이션과 함께 제출하는 모든 작업의 기본 구성에 영향을 줍니다. 자세한 내용은 Hive 구성 오버라이드 파라미터 및 Spark 구성 오버라이드 파라미터 단원을 참조하세요.

사용 가능한 구성 분류는 특정 EMR 서버리스 릴리스에 따라 다릅니다. 예를 들어 사용자 지정 Log4j에 대한 `spark-executor-log4j2` 분류는 릴리스 `spark-driver-log4j2` 6.8.0 이상에서만 사용할 수 있습니다. 애플리케이션별 속성 목록은 및 을 참조하십시오. 스파크 작업 속성 Hive 작업 속성

Apache Log4j2 속성을 구성할 수도 있습니다. AWS Secrets Manager 데이터 보호 및 애플리케이션 레벨에서의 Java 17 런타임을 위한 것입니다.

애플리케이션 수준에서 Secrets Manager 암호를 전달하려면 암호로 EMR 서버리스 애플리케이션을 만들거나 업데이트해야 하는 사용자 및 역할에 다음 정책을 연결하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerPolicy",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:secretsmanager:your-secret-arn"
        }
    ]
}
```

암호에 대한 사용자 지정 정책을 만드는 방법에 대한 자세한 내용은 다음과 같은 권한 정책 예를 참조하십시오. AWS Secrets Manager의 AWS Secrets Manager 사용 설명서.

> **Note**
>
> 응용 프로그램 수준에서 `runtimeConfiguration` 지정한 내용은 `applicationConfiguration` 에 `StartJobRun`API매핑됩니다.

예제 선언

다음 예제는 를 사용하여 `create-application` 기본 구성을 선언하는 방법을 보여줍니다.

```
aws emr-serverless create-application \
    --release-label release-version  \
    --type SPARK \
    --name my-application-name \
    --runtime-configuration '[
        {
            "classification": "spark-defaults",
            "properties": {
                "spark.driver.cores": "4",
                "spark.executor.cores": "2",
                "spark.driver.memory": "8G",
                "spark.executor.memory": "8G",
                "spark.executor.instances": "2",

 "spark.hadoop.javax.jdo.option.ConnectionDriverName":"org.mariadb.jdbc.Driver",
                "spark.hadoop.javax.jdo.option.ConnectionURL":"jdbc:mysql://db-host:db-
port/db-name",
                "spark.hadoop.javax.jdo.option.ConnectionUserName":"connection-user-
name",
                "spark.hadoop.javax.jdo.option.ConnectionPassword":
 "EMR.secret@SecretID"
            }
        },
        {
            "classification": "spark-driver-log4j2",
            "properties": {
                "rootLogger.level":"error",
                "logger.IdentifierForClass.name": "classpathForSettingLogger",
                "logger.IdentifierForClass.level": "info"
            }
        }
    ]' \
    --monitoring-configuration '{
```

```
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING/logs/app-level"
        },
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }'
```

작업 실행 중 구성 재정의

를 사용하여 애플리케이션 구성 및 모니터링 구성에 대한 구성 재정의를 지정할 수 있습니다. StartJobRunAPI EMR그런 다음 서버리스는 애플리케이션 수준과 작업 수준에서 지정하는 구성을 병합하여 작업 실행을 위한 구성을 결정합니다.

병합이 발생할 때의 세분성 수준은 다음과 같습니다.

- **ApplicationConfiguration**- 분류 유형, 예. `spark-defaults`
- **MonitoringConfiguration**- 구성 유형, 예`s3MonitoringConfiguration`.

> **Note**
>
> 에서 제공하는 구성의 우선 순위는 애플리케이션 수준에서 제공하는 구성을 StartJobRun대체합니다.

우선 순위 순위에 대한 자세한 내용은 및 을 참조하십시오Hive 구성 오버라이드 파라미터. Spark 구성 오버라이드 파라미터

작업을 시작할 때 특정 구성을 지정하지 않으면 해당 구성이 응용 프로그램에서 상속됩니다. 작업 수준에서 구성을 선언하면 다음 작업을 수행할 수 있습니다.

- 기존 구성 재정의 - `StartJobRun` 요청에 동일한 구성 매개변수를 재정의 값과 함께 제공하십시오.
- 추가 구성 추가 - 새 구성 매개변수를 지정하려는 값과 함께 `StartJobRun` 요청에 추가합니다.
- 기존 구성 제거 - 애플리케이션 런타임 구성을 제거하려면 제거하려는 구성의 키를 제공하고 구성에 `{}` 대한 빈 선언을 전달하십시오. 작업 실행에 필요한 매개 변수가 포함된 분류는 제거하지 않는 것이 좋습니다. 예를 들어 Hive 작업에 필요한 속성을 제거하려고 하면 작업이 실패합니다.

  응용 프로그램 모니터링 구성을 제거하려면 관련 구성 유형에 적합한 방법을 사용하십시오.

- **cloudWatchLoggingConfiguration**- 제거하려면 cloudWatchLogging enabled 플래그를 다음과 같이 false 전달하십시오.
- **managedPersistenceMonitoringConfiguration**- 관리형 지속성 설정을 제거하고 기본 활성화 상태로 돌아가려면 구성에 {} 대한 빈 선언을 전달하십시오.
- **s3MonitoringConfiguration**- 제거하려면 s3MonitoringConfiguration 구성에 {} 대한 빈 선언을 전달하십시오.

예제 오버라이드

다음 예는 에서 작업을 제출하는 동안 수행할 수 있는 다양한 작업을 보여줍니다. start-job-run

```
aws emr-serverless start-job-run \
    --application-id your-application-id \
    --execution-role-arn your-job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/
wordcount/scripts/wordcount.py",
            "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET-OUTPUT/wordcount_output"]
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [
            {
                // Override existing configuration for spark-defaults in the
 application
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.cores": "2",
                    "spark.executor.cores": "1",
                    "spark.driver.memory": "4G",
                    "spark.executor.memory": "4G"
                }
            },
            {
                // Add configuration for spark-executor-log4j2
                "classification": "spark-executor-log4j2",
                "properties": {
                    "rootLogger.level": "error",
                    "logger.IdentifierForClass.name": "classpathForSettingLogger",
                    "logger.IdentifierForClass.level": "info"
```

```
                }
            },
            {
                // Remove existing configuration for spark-driver-log4j2 from the
application
                "classification": "spark-driver-log4j2",
                "properties": {}
            }
        ],
        "monitoringConfiguration": {
            "managedPersistenceMonitoringConfiguration": {
                // Override existing configuration for managed persistence
                "enabled": true
            },
            "s3MonitoringConfiguration": {
                // Remove configuration of S3 monitoring
            },
            "cloudWatchLoggingConfiguration": {
                // Add configuration for CloudWatch logging
                "enabled": true
            }
        }
    }'
```

작업 실행 시 및 에 설명된 우선 순위 재정의 순위에 따라 다음 분류 및 구성이 적용됩니다. Hive 구성 오버라이드 파라미터 Spark 구성 오버라이드 파라미터

- 분류는 직무 수준에서 지정된 속성으로 `spark-defaults` 업데이트됩니다. 이 분류에는 에 `StartJobRun` 포함된 속성만 고려됩니다.
- 분류는 기존 분류 `spark-executor-log4j2` 목록에 추가됩니다.
- `spark-driver-log4j2`분류가 제거됩니다.
- 의 구성이 작업 수준의 구성으로 `managedPersistenceMonitoringConfiguration` 업데이트 됩니다.
- 의 구성이 `s3MonitoringConfiguration` 제거됩니다.
- 에 대한 구성이 기존 모니터링 구성에 `cloudWatchLoggingConfiguration` 추가됩니다.

EMR서버리스 이미지 사용자 지정

Amazon EMR 6.9.0부터 Amazon Serverless에서는 사용자 지정 이미지를 사용하여 애플리케이션 종속성과 런타임 환경을 단일 컨테이너로 패키징할 수 있습니다. EMR 이렇게 하면 워크로드 종속성을 관리하는 방법이 단순해지고 패키지의 이동성이 향상됩니다. EMR서버리스 이미지를 사용자 지정하면 다음과 같은 이점이 있습니다.

- 워크로드에 최적화된 패키지를 설치하고 구성합니다. Amazon EMR 런타임 환경의 공개 배포에서는 이러한 패키지가 널리 제공되지 않을 수 있습니다.
- EMR서버리스를 로컬 개발 및 테스트를 포함하여 조직 내에서 현재 확립된 빌드, 테스트 및 배포 프로세스와 통합합니다.
- 조직 내 규정 준수 및 거버넌스 요구 사항을 충족하는 확립된 보안 프로세스 (예: 이미지 스캔) 를 적용합니다.
- 자체 버전의 JDK 및 Python을 애플리케이션에 사용할 수 있습니다.

EMR서버리스는 이미지를 직접 만들 때 기본으로 사용할 수 있는 이미지를 제공합니다. 기본 이미지는 이미지가 EMR 서버리스와 상호 작용하는 데 필요한 필수 jar, 구성 및 라이브러리를 제공합니다. Amazon ECR 퍼블릭 갤러리에서 기본 이미지를 찾을 수 있습니다. 애플리케이션 유형 (Spark 또는 Hive) 및 릴리스 버전과 일치하는 이미지를 사용하십시오. 예를 들어 Amazon EMR 릴리스 6.9.0에서 애플리케이션을 생성하는 경우 다음 이미지를 사용하십시오.

유형	이미지
Spark	`public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest`
Hive	`public.ecr.aws/emr-serverless/hive/emr-6.9.0:latest`

사전 조건

EMR서버리스 사용자 지정 이미지를 생성하기 전에 다음 사전 요구 사항을 완료하십시오.

1. 동일한 위치에 Amazon ECR 리포지토리 생성 AWS 리전 EMR서버리스 애플리케이션을 시작하는 데 사용합니다. Amazon ECR 프라이빗 리포지토리를 생성하려면 프라이빗 리포지토리 생성을 참조하십시오.

2. 사용자에게 Amazon ECR 리포지토리에 대한 액세스 권한을 부여하려면 이 리포지토리의 이미지로 EMR 서버리스 애플리케이션을 만들거나 업데이트하는 사용자 및 역할에 다음 정책을 추가하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ECRRepositoryListGetPolicy",
            "Effect": "Allow",
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:DescribeImages"
            ],
            "Resource": "ecr-repository-arn"
        }
    ]
}
```

Amazon ECR 자격 증명 기반 정책의 예시를 더 보려면 Amazon Elastic 컨테이너 레지스트리 ID 기반 정책 예제를 참조하십시오.

1단계: 서버리스 기본 이미지에서 사용자 지정 이미지 생성 EMR

먼저 선호하는 기본 이미지를 사용하는 FROM 지침으로 시작하는 Dockerfile을 생성합니다. FROM지침 다음에 이미지에 적용하려는 모든 수정 사항을 포함할 수 있습니다. 기본 이미지는 자동으로 를 USER 로 설정합니다hadoop. 이 설정에 포함된 모든 수정 사항에 대한 권한이 없을 수도 있습니다. 해결 방법으로 USER 를 root 로 설정하고 이미지를 수정한 다음 USER 다시 로 hadoop:hadoop 설정하십시오. 일반적인 사용 사례의 샘플을 보려면 을 참조하십시오EMR서버리스에서 사용자 지정 이미지 사용.

```
# Dockerfile
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root
# MODIFICATIONS GO HERE

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

Dockerfile을 만든 후 다음 명령어를 사용하여 이미지를 빌드합니다.

```
# build the docker image
docker build . -t aws-account-id.dkr.ecr.region.amazonaws.com/my-
repository[:tag]or[@digest]
```

2단계: 로컬에서 이미지 유효성 검사

EMR서버리스는 사용자 지정 이미지를 정적으로 검사하여 기본 파일, 환경 변수 및 올바른 이미지 구성을 검증할 수 있는 오프라인 도구를 제공합니다. 도구 설치 및 실행 방법에 대한 자세한 내용은 Amazon EMR 서버리스 이미지를 CLI GitHub 참조하십시오.

도구를 설치한 후 다음 명령을 실행하여 이미지를 검증합니다.

```
amazon-emr-serverless-image \
validate-image -r emr-6.9.0 -t spark \
-i aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/@digest
```

다음과 비슷한 출력이 표시될 것입니다.

```
Amazon EMR Serverless - Image CLI
Version: 0.0.1
... Checking if docker cli is installed
... Checking Image Manifest
[INFO] Image ID: 9e2f4359cf5beb466a8a2ed047ab61c9d37786c555655fc122272758f761b41a
[INFO] Created On: 2022-12-02T07:46:42.586249984Z
[INFO] Default User Set to hadoop:hadoop : PASS
[INFO] Working Directory Set to  : PASS
[INFO] Entrypoint Set to /usr/bin/entrypoint.sh : PASS
[INFO] HADOOP_HOME is set with value: /usr/lib/hadoop : PASS
[INFO] HADOOP_LIBEXEC_DIR is set with value: /usr/lib/hadoop/libexec : PASS
[INFO] HADOOP_USER_HOME is set with value: /home/hadoop : PASS
[INFO] HADOOP_YARN_HOME is set with value: /usr/lib/hadoop-yarn : PASS
[INFO] HIVE_HOME is set with value: /usr/lib/hive : PASS
[INFO] JAVA_HOME is set with value: /etc/alternatives/jre : PASS
[INFO] TEZ_HOME is set with value: /usr/lib/tez : PASS
[INFO] YARN_HOME is set with value: /usr/lib/hadoop-yarn : PASS
[INFO] File Structure Test for hadoop-files in /usr/lib/hadoop: PASS
[INFO] File Structure Test for hadoop-jars in /usr/lib/hadoop/lib: PASS
[INFO] File Structure Test for hadoop-yarn-jars in /usr/lib/hadoop-yarn: PASS
[INFO] File Structure Test for hive-bin-files in /usr/bin: PASS
```

```
[INFO] File Structure Test for hive-jars in /usr/lib/hive/lib: PASS
[INFO] File Structure Test for java-bin in /etc/alternatives/jre/bin: PASS
[INFO] File Structure Test for tez-jars in /usr/lib/tez: PASS
---------------------------------------------------------------
Overall Custom Image Validation Succeeded.
---------------------------------------------------------------
```

3단계: Amazon ECR 리포지토리에 이미지 업로드

다음 명령을 사용하여 Amazon ECR 이미지를 Amazon ECR 리포지토리로 푸시합니다. 이미지를 리포지토리로 푸시할 수 있는 올바른 IAM 권한이 있는지 확인하십시오. 자세한 내용은 Amazon ECR 사용 설명서의 이미지 푸시를 참조하십시오.

```
# login to ECR repo
aws ecr get-login-password --region region | docker login --username AWS --password-
stdin aws-account-id.dkr.ecr.region.amazonaws.com

# push the docker image
docker push aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/@digest
```

4단계: 사용자 지정 이미지로 애플리케이션 생성 또는 업데이트

선택: AWS Management Console 탭 또는 AWS CLI 원하는 애플리케이션 실행 방법에 따라 탭을 누른 후 다음 단계를 완료하십시오.

Console

1. https://console.aws.amazon.com/emr에서 EMR 스튜디오 콘솔에 로그인합니다. 애플리케이션으로 이동하거나 애플리케이션 만들기의 지침에 따라 새 애플리케이션을 생성하십시오.
2. EMR서버리스 애플리케이션을 만들거나 업데이트할 때 사용자 지정 이미지를 지정하려면 애플리케이션 설치 옵션에서 사용자 지정 설정을 선택합니다.
3. 사용자 지정 이미지 설정 섹션에서 이 응용 프로그램에 사용자 지정 이미지 사용 확인란을 선택합니다.
4. Amazon ECR 이미지를 이미지 URI 필드에 URI 붙여넣습니다. EMR서버리스는 애플리케이션의 모든 작업자 유형에 이 이미지를 사용합니다. 다른 사용자 지정 이미지를 선택하고 각 작업자 유형에 URIs 대해 서로 다른 Amazon ECR 이미지를 붙여넣을 수도 있습니다.

CLI

- `image-configuration`파라미터를 사용하여 애플리케이션을 생성합니다. EMR서버리스는 이 설정을 모든 작업자 유형에 적용합니다.

```
aws emr-serverless create-application \
--release-label emr-6.9.0 \
--type SPARK \
--image-configuration '{
    "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/
@digest"
}'
```

각 작업자 유형별로 다른 이미지 설정을 사용하여 응용 프로그램을 만들려면 `worker-type-specifications` 매개 변수를 사용하십시오.

```
aws emr-serverless create-application \
--release-label emr-6.9.0 \
--type SPARK \
--worker-type-specifications '{
    "Driver": {
        "imageConfiguration": {
            "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-
repository:tag/@digest"
        }
    },
    "Executor" : {
        "imageConfiguration": {
            "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-
repository:tag/@digest"
        }
    }
}'
```

응용 프로그램을 업데이트하려면 `image-configuration` 매개 변수를 사용합니다. EMR서버리스는 이 설정을 모든 작업자 유형에 적용합니다.

```
aws emr-serverless update-application \
--application-id application-id \
--image-configuration '{
```

```
    "imageUri": "aws-account-id.dkr.ecr.region.amazonaws.com/my-repository:tag/
@digest"
}'
```

5단계: EMR 서버리스가 사용자 지정 이미지 리포지토리에 액세스하도록 허용

Amazon ECR 리포지토리에 다음 리소스 정책을 추가하여 EMR 서버리스 서비스 보안 주체가 이 리포지토리의get,describe, download 요청을 사용할 수 있도록 하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Emr Serverless Custom Image Support",
      "Effect": "Allow",
      "Principal": {
        "Service": "emr-serverless.amazonaws.com"
      },
      "Action": [
        "ecr:BatchGetImage",
        "ecr:DescribeImages",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Condition":{
        "StringEquals":{
          "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
        }
      }
    }
  ]
}
```

보안 모범 사례로서 aws:SourceArn 조건 키를 리포지토리 정책에 추가하십시오. IAM글로벌 조건 키는 aws:SourceArn EMR 서버리스가 ARN 애플리케이션에만 리포지토리를 사용하도록 합니다. Amazon ECR 리포지토리 정책에 대한 자세한 내용은 프라이빗 리포지토리 생성을 참조하십시오.

고려 사항 및 제한

사용자 지정 이미지로 작업할 때는 다음 사항을 고려하십시오.

- 애플리케이션의 유형 (Spark 또는 Hive) 및 릴리스 라벨 (예:) 과 일치하는 올바른 기본 이미지를 사용하십시오. `emr-6.9.0`
- EMR서버리스는 Docker 파일의 `[CMD]` `[ENTRYPOINT]` 명령이나 명령을 무시합니다. Docker 파일에 있는 일반적인 지침 (예`[COPY]`:,,) 을 사용하십시오. `[RUN]` `[WORKDIR]`
- 사용자 지정 이미지를 만들 `TEZ_HOME` 때 환경 변수 `JAVA_HOMESPARK_HOME,HIVE_HOME`, 를 수정해서는 안 됩니다.
- 사용자 지정 이미지의 크기는 5GB를 초과할 수 없습니다.
- Amazon EMR 기본 이미지에서 바이너리나 jar를 수정하면 애플리케이션 또는 작업 시작이 실패할 수 있습니다.
- Amazon ECR 리포지토리는 동일한 위치에 있어야 합니다. AWS 리전 EMR서버리스 애플리케이션을 시작하는 데 사용하는 것입니다.

액세스 구성 VPC

Amazon Redshift 클러스터VPC, Amazon 데이터베이스 또는 엔드포인트가 있는 Amazon S3 VPC 버킷과 같은 사용자 내의 데이터 스토어에 연결하도록 EMR 서버리스 애플리케이션을 구성할 수 있습니다. RDS EMR서버리스 애플리케이션은 내부 데이터 스토어에 대한 아웃바운드 연결을 제공합니다. VPC 기본적으로 EMR 서버리스는 애플리케이션에 대한 인바운드 액세스를 차단하여 보안을 강화합니다.

> **Note**
>
> 애플리케이션에 외부 Hive 메타스토어 데이터베이스를 사용하려면 VPC 액세스를 구성해야 합니다. 외부 Hive 메타스토어를 구성하는 방법에 대한 자세한 내용은 메타스토어 구성을 참조하십시오.

애플리케이션 생성

애플리케이션 생성 페이지에서 사용자 지정 설정을 선택하고 서버리스 애플리케이션이 사용할 수 있는 서브넷VPC, 보안 그룹을 지정할 수 있습니다. EMR

VPCs

데이터 저장소가 포함된 가상 사설 클라우드 (VPC) 의 이름을 선택합니다. 애플리케이션 생성 페이지에는 선택한 모든 항목이 VPCs 나열됩니다. AWS 리전.

서브넷

데이터 저장소가 VPC 포함된 서브넷을 선택합니다. 애플리케이션 생성 페이지에는 내 데이터 저장소의 모든 서브넷이 나열됩니다. VPC

선택한 서브넷은 프라이빗 서브넷이어야 합니다. 즉, 서브넷의 관련 라우팅 테이블에는 인터넷 게이트웨이가 없어야 합니다.

인터넷에 대한 아웃바운드 연결을 위해서는 서브넷에 게이트웨이를 사용하는 아웃바운드 경로가 있어야 합니다. NAT 게이트웨이를 구성하려면 NAT 게이트웨이 사용을 참조하십시오. NAT

Amazon S3 연결의 경우 서브넷에 NAT 게이트웨이 또는 VPC 엔드포인트가 구성되어 있어야 합니다. S3 VPC 엔드포인트를 구성하려면 게이트웨이 엔드포인트 생성을 참조하십시오.

다른 기기와의 연결용 AWS 서비스 Amazon DynamoDB와 같은 외부 환경에서는 엔드포인트 VPC 또는 게이트웨이를 구성해야 합니다. VPC NAT 엔드포인트를 구성하려면 VPC AWS 서비스VPC엔드포인트 사용을 참조하십시오.

작업자는 아웃바운드 트래픽을 VPC 통해 내 데이터 저장소에 연결할 수 있습니다. 기본적으로 EMR 서버리스는 작업자에 대한 인바운드 액세스를 차단하여 보안을 개선합니다.

를 사용하는 경우 AWS Config, EMR Serverless는 모든 작업자에 대한 Elastic Network 인터페이스 항목 레코드를 생성합니다. 이 리소스와 관련된 비용을 피하려면 사용을 `AWS::EC2::NetworkInterface` 중지하는 것이 좋습니다. AWS Config.

> **Note**
>
> 여러 가용 영역에서 여러 서브넷을 선택하는 것이 좋습니다. 선택한 서브넷에 따라 EMR 서버리스 애플리케이션을 시작하는 데 사용할 수 있는 가용 영역이 결정되기 때문입니다. 각 작업자는 시작된 서브넷의 IP 주소를 사용합니다. 지정된 서브넷에 시작하려는 작업자 수에 해당하는 충분한 IP 주소가 있는지 확인하십시오. 서브넷 계획에 대한 자세한 내용은 을 참조하십시오. the section called “서브넷 계획 모범 사례”

보안 그룹

데이터 스토어와 통신할 수 있는 보안 그룹을 하나 이상 선택하십시오. 애플리케이션 생성 페이지에는 내 모든 보안 그룹이 나열됩니다VPC. EMR서버리스는 이러한 보안 그룹을 서브넷에 연결된 엘라스틱 네트워크 인터페이스와 연결합니다. VPC

> **Note**
>
> 서버리스 애플리케이션을 위한 EMR 별도의 보안 그룹을 생성하는 것이 좋습니다. 이렇게 하면 네트워크 규칙을 더 효율적으로 분리하고 관리할 수 있습니다. 예를 들어 Amazon Redshift 클러스터와 통신하려면 아래 예와 같이 Redshift와 EMR 서버리스 보안 그룹 간의 트래픽 규칙을 정의할 수 있습니다.

Example 예 — Amazon Redshift 클러스터와의 통신

1. 서버리스 보안 그룹 중 하나에서 Amazon Redshift 보안 그룹에 인바운드 EMR 트래픽에 대한 규칙을 추가합니다.

유형	프로토콜	포트 범위	소스
모두 TCP	TCP	5439	`emr-serverless-security-group`

2. EMR서버리스 보안 그룹 중 하나의 아웃바운드 트래픽에 대한 규칙을 추가합니다. 이 작업은 두 가지 방법으로 수행할 수 있습니다. 먼저 모든 포트에 대한 아웃바운드 트래픽을 개방할 수 있습니다.

유형	프로토콜	포트 범위	대상
모든 트래픽	TCP	ALL	0.0.0.0/0

또는 아웃바운드 트래픽을 Amazon Redshift 클러스터로 제한할 수 있습니다. 이는 애플리케이션이 Amazon Redshift 클러스터와 통신해야 하고 다른 어떤 것과도 통신하지 않는 경우에만 유용합니다.

유형	프로토콜	포트 범위	소스
모두 TCP	TCP	5439	`redshift-security-group`

애플리케이션 구성

애플리케이션 구성 페이지에서 기존 EMR 서버리스 애플리케이션의 네트워크 구성을 변경할 수 있습니다.

작업 실행 세부 정보 보기

Job run 세부 정보 페이지에서 특정 실행에 대한 작업에서 사용하는 서브넷을 볼 수 있습니다. 단, 작업은 지정된 서브넷에서 선택한 하나의 서브넷에서만 실행됩니다.

서브넷 계획 모범 사례

AWS 리소스는 Amazon에서 사용 가능한 IP 주소의 하위 집합인 서브넷에서 생성됩니다. VPC 예를 들어 /16 넷마스크가 VPC 있는 경우 사용 가능한 IP 주소는 최대 65,536개이며, 서브넷 마스크를 사용하여 여러 개의 소규모 네트워크로 분할할 수 있습니다. 예를 들어, 이 범위를 각각 /17 마스크와 32,768개의 사용 가능한 IP 주소를 사용하는 두 개의 서브넷으로 분할할 수 있습니다. 서브넷은 가용 영역 내에 있으며 여러 영역에 걸쳐 있을 수 없습니다.

서브넷은 EMR 서버리스 애플리케이션 확장 제한을 염두에 두고 설계해야 합니다. 예를 들어 4명의 vCpu 작업자를 요청하는 애플리케이션이 있고 최대 vCpu 4,000개까지 확장할 수 있는 경우 애플리케이션에는 총 1,000개의 네트워크 인터페이스에 대해 최대 1,000명의 작업자가 필요합니다. 여러 가용 영역에 걸쳐 서브넷을 생성하는 것이 좋습니다. 이를 통해 EMR 서버리스가 작업을 재시도하거나 가용 영역에 장애가 발생하는 예기치 않은 상황이 발생하더라도 다른 가용 영역에서 사전 초기화된 용량을 프로비저닝할 수 있습니다. 따라서 2개 이상의 가용 영역에 있는 각 서브넷에는 1,000개 이상의 사용 가능한 IP 주소가 있어야 합니다.

1,000개의 네트워크 인터페이스를 프로비저닝하려면 마스크 크기가 22보다 작거나 같은 서브넷이 필요합니다. 22개 이상의 마스크는 요구 사항을 충족하지 못합니다. 예를 들어 서브넷 마스크 /23은 512개의 IP 주소를 제공하는 반면, /22의 마스크는 1024개를 제공하고, /21의 마스크는 2048개의 IP 주소를 제공합니다. 다음은 다양한 가용 영역에 할당할 수 있는 /16 넷마스크의 /22 마스크가 있는 4개의

VPC 서브넷의 예입니다. 각 서브넷의 처음 네 개의 IP 주소와 마지막 IP 주소는 다음에 예약되어 있기 때문에 사용 가능한 IP 주소와 사용 가능한 IP 주소 간에는 다섯 개의 차이가 있습니다. AWS.

서브넷 ID	서브넷 주소	서브넷 마스크	IP 주소 범위	사용 가능한 IP 주소	사용 가능한 IP 주소
1	10.0.0.0	255.255.252.0/22	10.0.0.0 - 10.0.3.255	1,024	1,019
2	10.0.4.0	255.255.252.0/22	10.0.4.0 - 10.0.7.255	1,024	1,019
3	10.0.8.0	255.255.252.0/22	10.0.4.0 - 10.0.7.255	1,024	1,019
4	10.0.12.0	255.255.252.0/22	10.0.12.0 - 10.0.15.255	1,024	1,019

작업량이 대규모 작업자에게 가장 적합한지 평가해야 합니다. 작업자 크기가 더 크면 필요한 네트워크 인터페이스가 더 적습니다. 예를 들어 애플리케이션 확장 제한이 4,000개인 vCpu 작업자 16명을 사용하는 경우 네트워크 인터페이스를 프로비저닝하려면 최대 250명의 작업자, 총 250개의 사용 가능한 IP 주소가 vCpu 필요합니다. 250개의 네트워크 인터페이스를 프로비저닝하려면 마스크 크기가 24보다 작거나 같은 여러 가용 영역에 서브넷이 있어야 합니다. 24보다 큰 마스크 크기는 250개 미만의 IP 주소를 제공합니다.

여러 애플리케이션에서 서브넷을 공유하는 경우 모든 애플리케이션의 집합적 확장 제한을 염두에 두고 각 서브넷을 설계해야 합니다. 예를 들어 vCpu 작업자 4명을 요청하는 애플리케이션이 3개이고 각 애플리케이션이 vCpu 계정 수준 서비스 기반 할당량 12,000개로 최대 4000개까지 확장할 수 vCpu 있는 경우 각 서브넷에는 3000개의 사용 가능한 IP 주소가 필요합니다. 사용하려는 IP 주소 수가 충분하지 않은 경우 사용 가능한 IP 주소 수를 늘려 보십시오. VPC 추가 클래스 없는 도메인 간 라우팅 (CIDR) 블록을 사용자 블록에 연결하여 이 작업을 수행할 수 있습니다. VPC 자세한 내용은 Amazon VPC 사용 설명서의 추가 IPv4 CIDR 블록을 사용자 블록과 연결을 참조하십시오. VPC

온라인에서 제공되는 여러 도구 중 하나를 사용하여 서브넷 정의를 빠르게 생성하고 사용 가능한 IP 주소 범위를 검토할 수 있습니다.

Amazon EMR 서버리스 아키텍처 옵션

Amazon EMR Serverless 애플리케이션의 명령 세트 아키텍처는 애플리케이션이 작업을 실행하는 데 사용하는 프로세서 유형을 결정합니다. EMR아마존은 애플리케이션에 x86_64와 arm64라는 두 가지 아키텍처 옵션을 제공합니다. EMR서버리스는 최신 세대의 인스턴스가 제공되는 대로 자동으로 업데이트하므로 애플리케이션에서 추가 작업 없이 새 인스턴스를 사용할 수 있습니다.

주제

x86_64 아키텍처 사용

x86_64 아키텍처는 x86 64비트 또는 x64라고도 합니다. x86_64는 서버리스 애플리케이션의 기본 옵션입니다. EMR 이 아키텍처는 x86 기반 프로세서를 사용하며 대부분의 타사 도구 및 라이브러리와 호환됩니다.

대부분의 애플리케이션은 x86 하드웨어 플랫폼과 호환되며 기본 x86_64 아키텍처에서 성공적으로 실행될 수 있습니다. 하지만 애플리케이션이 64비트와 호환되는 경우 arm64로 전환하여 Graviton 프로세서를 사용하여 성능ARM, 컴퓨팅 파워 및 메모리를 개선할 수 있습니다. arm64 아키텍처에서 인스턴스를 실행하는 것이 x86 아키텍처에서 동일한 크기의 인스턴스를 실행할 때보다 비용이 적게 듭니다.

arm64 아키텍처 사용 (그라비톤)

AWS Graviton 프로세서는 다음에 의해 맞춤 설계되었습니다. AWS 64비트 ARM 네버스 코어를 사용하며 Arm64 아키텍처 (Arch64 또는 64비트라고도 함) 를 활용합니다. ARM The AWS EMR서버리스에서 사용할 수 있는 Graviton 프로세서 제품군에는 Graviton3 및 Graviton2 프로세서가 포함됩니다. 이러한 프로세서는 x86_64 아키텍처에서 실행되는 동급 워크로드에 비해 Spark 및 Hive 워크로드에 대해 뛰어난 가격 대비 성능을 제공합니다. EMR서버리스는 최신 프로세서로 업그레이드하기 위한 별도의 노력 없이 가능한 경우 최신 프로세서를 자동으로 사용합니다.

Graviton을 지원하는 새 애플리케이션 출시

다음 방법 중 하나를 사용하여 arm64 아키텍처를 사용하는 애플리케이션을 시작하십시오.

AWS CLI

Graviton 프로세서를 사용하여 응용 프로그램을 시작하려면 AWS CLI에서 `architecture` 매개 ARM64 변수로 지정합니다. `create-application` API 다른 매개변수에서 애플리케이션에 적합한 값을 입력합니다.

```
aws emr-serverless create-application \
 --name my-graviton-app \
 --release-label emr-6.8.0 \
 --type "SPARK" \
 --architecture "ARM64" \
 --region us-west-2
```

EMR Studio

EMRStudio의 Graviton 프로세서를 사용하여 응용 프로그램을 시작하려면 응용 프로그램을 만들거나 업데이트할 때 아키텍처 옵션으로 arm64를 선택합니다.

Graviton을 사용하도록 기존 애플리케이션을 구성합니다.

다음과 함께 Graviton (arm64) 아키텍처를 사용하도록 기존 Amazon EMR 서버리스 애플리케이션을 구성할 수 있습니다. SDK AWS CLI또는 스튜디오. EMR

기존 애플리케이션을 x86에서 arm64로 변환하려면

1. 최신 메이저 버전을 사용하고 있는지 확인하십시오. AWS CLI SDK/는 `architecture` 파라미터를 지원합니다.
2. 실행 중인 작업이 없는지 확인한 다음 애플리케이션을 중지하십시오.

   ```
   aws emr-serverless stop-application \
    --application-id application-id \
    --region us-west-2
   ```

3. Graviton을 사용하도록 응용 프로그램을 업데이트하려면 에서 `architecture` 매개 변수를 지정하십시오ARM64. `update-application` API

   ```
   aws emr-serverless update-application \
    --application-id application-id \
    --architecture 'ARM64' \
    --region us-west-2
   ```

4. 응용 프로그램 CPU 아키텍처가 현재 상태인지 확인하려면 ARM64 를 사용하십시오. `get-application` API

```
aws emr-serverless get-application \
 --application-id application-id \
 --region us-west-2
```

5. 준비가 되면 애플리케이션을 다시 시작합니다.

```
aws emr-serverless start-application \
 --application-id application-id \
 --region us-west-2
```

Graviton 사용 시 고려 사항

Graviton 지원을 위해 arm64를 사용하여 EMR 서버리스 애플리케이션을 시작하기 전에 다음 사항을 확인하십시오.

라이브러리 호환성

아키텍처 옵션으로 Graviton (arm64) 을 선택할 때는 타사 패키지 및 라이브러리가 64비트 아키텍처와 호환되는지 확인하십시오. ARM 선택한 아키텍처와 호환되는 Python 가상 환경으로 Python 라이브러리를 패키징하는 방법에 대한 자세한 내용은 을 참조하십시오EMR서버리스에서 Python 라이브러리 사용하기.

ARM64비트를 사용하도록 Spark 또는 Hive 워크로드를 구성하는 방법에 대한 자세한 내용은 다음을 참조하십시오. AWS Graviton 시작하기 리포지토리가 켜져 있습니다. GitHub 이 리포지토리에는 Graviton ARM 기반 사용을 시작하는 데 도움이 되는 필수 리소스가 포함되어 있습니다.

EMR서버리스를 사용하여 S3 익스프레스 원 존으로 데이터 가져오기

Amazon EMR 릴리스 7.2.0 이상에서는 Amazon S3 Express One Zone 스토리지 클래스와 함께 EMR 서버리스를 사용하여 작업 및 워크로드를 실행할 때 성능을 개선할 수 있습니다. S3 Express One Zone은 고성능 단일 영역 Amazon S3 스토리지 클래스로, 지연 시간에 민감한 대부분의 애플리케이션에 일관되게 10밀리초 미만의 데이터 액세스를 제공합니다. 릴리스 시점에 S3 Express One Zone은 Amazon S3에서 지연 시간이 가장 낮고 성능은 가장 뛰어난 클라우드 객체 스토리지를 제공합니다.

사전 조건

- S3 Express One Zone 권한 — S3 Express One Zone이 처음에 S3 객체에서 또는 같은 `GET` 작업을 수행하는 `LIST` 경우 `PUT` 스토리지 클래스가 사용자를 대신하여 호출합니다. `CreateSession` IAM 정책에서 `s3express:CreateSession` 권한을 허용해야 합니다. 이렇게 하려면 S3A 커넥터는 를 호출할 수 있습니다. `CreateSession` API 이 권한이 있는 정책 예시는 S3 Express One Zone 시작하기 섹션을 참조하세요.
- S3A 커넥터 - S3 Express One Zone 스토리지 클래스를 사용하는 Amazon S3 버킷의 데이터에 액세스하도록 Spark를 구성하려면 Apache 하둡 커넥터를 사용해야 합니다.S3A. 커넥터를 사용하려면 모든 S3가 이 `s3a` 구성표를 URIs 사용해야 합니다. 그렇지 않은 경우 `s3` 및 `s3n` 체계에 대해 사용하는 파일 시스템 구현을 변경할 수 있습니다.

`s3` 체계를 변경하려면 다음 클러스터 구성을 지정하세요.

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

`s3n` 체계를 변경하려면 다음 클러스터 구성을 지정하세요.

```
[
```

```
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3n.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3n.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

S3 Express One Zone 시작하기

다음 단계에 따라 S3 익스프레스 원 존을 시작하십시오.

1. VPC엔드포인트를 생성합니다. 엔드포인트를 `com.amazonaws.us-west-2.s3express` VPC 엔드포인트에 추가합니다.
2. Amazon EMR 서버리스 시작하기 단계를 따라 Amazon EMR 릴리스 레이블이 7.2.0 이상인 애플리케이션을 생성하십시오.
3. 새로 생성된 VPC 엔드포인트, 프라이빗 서브넷 그룹 및 보안 그룹을 사용하도록 애플리케이션을 구성하십시오.
4. 작업 실행 역할에 `CreateSession` 권한을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": [
                "s3express:CreateSession"
            ]
        }
    ]
}
```

5. 작업을 실행합니다. 참고로 S3 익스프레스 원 존 버킷에 액세스하려면 `S3A` 스킴을 사용해야 합니다.

```
aws emr-serverless start-job-run \
--application-id <application-id> \
--execution-role-arn <job-role-arn> \
```

```
--name <job-run-name> \
--job-driver '{
 "sparkSubmit": {


 "entryPoint": "s3a://<DOC-EXAMPLE-BUCKET>/scripts/wordcount.py",
 "entryPointArguments":["s3a://<DOC-EXAMPLE-BUCKET>/emr-serverless-spark/output"],
 "sparkSubmitParameters": "--conf spark.executor.cores=4
 --conf spark.executor.memory=8g --conf spark.driver.cores=4
 --conf spark.driver.memory=8g --conf spark.executor.instances=2
 --conf spark.hadoop.fs.s3a.change.detection.mode=none
 --conf spark.hadoop.fs.s3a.endpoint.region={<AWS_REGION>}
 --conf spark.hadoop.fs.s3a.select.enabled=false
 --conf spark.sql.sources.fastS3PartitionDiscovery.enabled=false
 }'
```

작업 실행

애플리케이션을 프로비저닝한 후 애플리케이션에 작업을 제출할 수 있습니다. 이 섹션에서는 사용 방법을 다룹니다. AWS CLI 이러한 작업을 실행하려면 또한 이 섹션에서는 EMR 서버리스에서 사용할 수 있는 각 애플리케이션 유형의 기본값을 식별합니다.

주제

작업 실행 상태

Amazon EMR Serverless 작업 대기열에 작업 실행을 제출하면 작업 실행이 `SUBMITTED` 상태가 됩니다. 작업 상태는 ~에서 `FAILED`SUCCESS, 또는 `CANCELLING` 에 도달할 `RUNNING` 때까지 `SUBMITTED` 전달됩니다.

다음은 가능한 작업 실행 상태입니다.

State	설명
제출됨	작업 실행을 EMR 서버리스에 제출할 때의 초기 작업 상태입니다. 응용 프로그램에 대한 작업 일정이 잡힐 때까지 대기합니다. EMR서버리스가 우선 순위를 지정하고 작업 실행 일정을 잡기 시작합니다.

State	설명
보류중	스케줄러는 작업 실행을 평가하여 응용 프로그램 실행의 우선 순위를 지정하고 실행 일정을 잡습니다.
스케줄링됨	EMR서버리스는 애플리케이션의 작업 실행을 예약하고 작업 실행을 위한 리소스를 할당하고 있습니다.
[실행 중]	EMR서버리스는 작업에 처음에 필요한 리소스를 할당했으며 작업은 애플리케이션에서 실행되고 있습니다. Spark 애플리케이션에서 이는 Spark 드라이버 프로세스가 `running` 상태임을 의미합니다.
실패	EMR서버리스가 작업 실행을 애플리케이션에 제출하지 못했거나 성공적으로 완료되지 못했습니다. 이 작업 `StateDetails` 실패에 대한 추가 정보는 을 참조하십시오.
Success	작업 실행이 성공적으로 완료되었습니다.
취소 중	에서 `CancelJobRun` API 작업 실행 취소를 요청했거나 작업 실행 시간이 초과되었습니다. EMR서버리스가 애플리케이션에서 작업을 취소하고 리소스를 릴리스하려고 합니다.
취소됨	작업 실행이 성공적으로 취소되었으며 사용된 리소스가 릴리스되었습니다.

EMR스튜디오 콘솔에서 작업 실행

EMR서버리스 애플리케이션에 작업 실행을 제출하고 EMR Studio 콘솔에서 작업을 볼 수 있습니다. EMRStudio 콘솔에서 EMR 서버리스 애플리케이션을 만들거나 탐색하려면 콘솔에서 시작하기의 지침을 따르십시오.

작업 제출

작업 제출 페이지에서 다음과 같이 EMR 서버리스 애플리케이션에 작업을 제출할 수 있습니다.

Spark

1. 이름 필드에 작업 실행의 이름을 입력합니다.
2. 런타임 역할 필드에 EMR 서버리스 애플리케이션이 작업 실행을 위해 맡을 수 있는 IAM 역할 이름을 입력합니다. 런타임 역할에 대한 자세한 내용은 을 참조하십시오Amazon EMR 서버리스의 Job 런타임 역할.
3. 스크립트 위치 필드에 스크립트의 Amazon S3 위치 또는 JAR 실행하려는 위치를 입력합니다. Spark 작업의 경우 스크립트는 Python (`.py`) 파일 또는 () 파일일 수 있습니다. JAR `.jar`
4. 스크립트 위치가 JAR 파일인 경우 작업의 진입점인 클래스 이름을 Main class 필드에 입력합니다.
5. (선택 사항) 나머지 필드의 값을 입력합니다.

   - 스크립트 인수 — 기본 JAR 또는 Python 스크립트에 전달하려는 인수를 입력합니다. 코드는 이러한 매개 변수를 읽습니다. 배열의 각 인수를 쉼표로 구분합니다.
   - 스파크 속성 - Spark 속성 섹션을 펼치고 이 필드에 Spark 구성 매개변수를 입력합니다.

     > **Note**
     >
     > Spark 드라이버 및 실행기 크기를 지정하는 경우 메모리 오버헤드를 고려해야 합니다. 속성 및 에서 메모리 오버헤드 값을 지정합니다. `spark.driver.memoryOverhead` `spark.executor.memoryOverhead` 메모리 오버헤드의 기본값은 컨테이너 메모리의 10% 이며, 최소 384MB입니다. 실행기 메모리와 메모리 오버헤드를 모두 합하면 작업자 메모리를 초과할 수 없습니다. 예를 들어 30GB 작업자의 최대 `spark.executor.memory` 용량은 27GB여야 합니다.

   - 작업 구성 - 이 필드에 작업 구성을 지정합니다. 이러한 작업 구성을 사용하여 응용 프로그램의 기본 구성을 재정의할 수 있습니다.
   - 추가 설정 — 활성화 또는 비활성화 AWS Data Catalog를 메타스토어로 통합하고 애플리케이션 로그 설정을 수정합니다. 메타스토어 구성에 대한 자세한 내용은 을 참조하십시오. 메타스토어 구성 애플리케이션 로깅 옵션에 대한 자세한 내용은 을 참조하십시오. 로그 저장
   - 태그 - 애플리케이션에 사용자 지정 태그를 할당합니다.

6. 작업 제출을 선택합니다.

Hive

1. 이름 필드에 작업 실행 이름을 입력합니다.
2. 런타임 역할 필드에 EMR 서버리스 애플리케이션이 작업 실행을 위해 맡을 수 있는 IAM 역할 이름을 입력합니다.
3. 스크립트 위치 필드에 스크립트의 Amazon S3 위치 또는 JAR 실행하려는 위치를 입력합니다. Hive 작업의 경우 스크립트는 Hive (`.sql`) 파일이어야 합니다.
4. (선택 사항) 나머지 필드의 값을 입력합니다.

   - 초기화 스크립트 위치 - Hive 스크립트가 실행되기 전에 테이블을 초기화하는 스크립트의 위치를 입력합니다.
   - Hive 속성 - Hive 속성 섹션을 확장하고 이 필드에 Hive 구성 매개 변수를 입력합니다.
   - 작업 구성 - 모든 작업 구성을 지정합니다. 이러한 작업 구성을 사용하여 응용 프로그램의 기본 구성을 재정의할 수 있습니다. Hive 작업의 경우 `hive.exec.scratchdir` 및 `hive.metastore.warehouse.dir` 는 구성의 `hive-site` 필수 속성입니다.

     ```
     {
         "applicationConfiguration": [
             {
                 "classification": "hive-site",
                 "configurations": [],
                 "properties": {
                     "hive.exec.scratchdir": "s3://DOC-EXAMPLE_BUCKET/hive/
     scratch",
                     "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE_BUCKET/hive/
     warehouse"
                 }
             }
         ],
         "monitoringConfiguration": {}
     }
     ```

   - 추가 설정 — 활성화 또는 비활성화 AWS Data Catalog를 메타스토어로 통합하고 애플리케이션 로그 설정을 수정합니다. 메타스토어 구성에 대한 자세한 내용은 을 참조하십시오. 메타스토어 구성 애플리케이션 로깅 옵션에 대한 자세한 내용은 을 참조하십시오. 로그 저장
   - 태그 - 애플리케이션에 모든 사용자 정의 태그를 할당합니다.

5. 작업 제출을 선택합니다.

작업 실행 보기

응용 프로그램 세부 정보 페이지의 Job run 탭에서 작업 실행을 보고 작업 실행에 대해 다음과 같은 작업을 수행할 수 있습니다.

작업 취소 - RUNNING 상태에 있는 작업 실행을 취소하려면 이 옵션을 선택합니다. 작업 실행 전환에 대한 자세한 내용은 을 참조하십시오작업 실행 상태.

작업 복제 - 이전 작업 실행을 복제하고 다시 제출하려면 이 옵션을 선택합니다.

에서 작업 실행 AWS CLI

에서 개별 작업을 생성, 설명 및 삭제할 수 있습니다. AWS CLI. 모든 작업을 나열하여 한 눈에 볼 수도 있습니다.

새 작업을 제출하려면 를 사용하십시오`start-job-run`. 실행하려는 응용 프로그램의 ID를 작업별 속성과 함께 제공하십시오. Spark 예제는 을 참조하십시오. 스파크 잡스 Hive 예제는 을 참조하십시오. 하이브 잡스 이 명령은`application-id`,ARN, new를 반환합니다. `job-id`

각 작업 실행에는 설정된 제한 시간이 있습니다. 작업 실행 시간이 이 기간을 초과하면 EMR Serverless에서 자동으로 작업을 취소합니다. 기본 제한 시간은 12시간입니다. 작업 실행을 시작할 때 이 시간 제한 설정을 작업 요구 사항에 맞는 값으로 구성할 수 있습니다. `executionTimeoutMinutes`속성을 사용하여 값을 구성합니다.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --execution-timeout-minutes 15 \
  --job-driver '{
    "hive": {
        "query": "s3://DOC-EXAMPLE-BUCKET/scripts/create_table.sql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/hive/
scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/hive/warehouse"
    }
   }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
```

```
        "properties": {
            "hive.client.cores": "2",
            "hive.client.memory": "4GIB"
        }
    }]
}'
```

작업을 설명하려면 를 사용하십시오`get-job-run`. 이 명령은 작업별 구성과 새 작업에 설정된 용량을 반환합니다.

```
aws emr-serverless get-job-run \
--job-run-id job-id \
--application-id application-id
```

작업을 나열하려면 를 사용하십시오. `list-job-runs` 이 명령은 작업 유형, 상태 및 기타 상위 수준 속성을 포함하는 간략한 속성 집합을 반환합니다. 모든 작업을 보지 않으려는 경우 표시하려는 최대 작업 수 (최대 50개) 를 지정할 수 있습니다. 다음 예제에서는 마지막 두 개의 작업 실행을 보도록 지정합니다.

```
aws emr-serverless list-job-runs \
--max-results 2 \
--application-id application-id
```

작업을 취소하려면 를 사용하십시오`cancel-job-run`. 취소하려는 `job-id` 작업의 `application-id` 및 이름을 입력합니다.

```
aws emr-serverless cancel-job-run \
--job-run-id job-id \
--application-id application-id
```

에서 작업을 실행하는 방법에 대한 자세한 내용은 AWS CLIEMR서버리스 API 참조를 참조하십시오.

셔플 최적화 디스크 사용

Amazon EMR 릴리스 7.1.0 이상에서는 Apache Spark 또는 Hive 작업을 실행할 때 셔플에 최적화된 디스크를 사용하여 I/O 집약적인 워크로드의 성능을 개선할 수 있습니다. 셔플 최적화 디스크는 표준 디스크와 비교하여 더 높은 IOPS (초당 I/O 작업 수) 을 제공하므로 데이터 이동 속도가 빨라지고 셔플 작업 중 지연 시간이 줄어듭니다. 셔플에 최적화된 디스크를 사용하면 작업자당 최대 2TB의 디스크 크기를 연결할 수 있으므로 워크로드 요구 사항에 맞게 적절한 용량을 구성할 수 있습니다.

주요 이점

셔플 최적화 디스크는 다음과 같은 이점을 제공합니다.

- IOPS고성능 — 셔플 최적화 디스크는 표준 IOPS 디스크보다 더 높은 성능을 제공하므로 Spark 및 Hive 작업 및 기타 셔플 집약적 워크로드 중에 더 효율적이고 빠른 데이터 셔플링이 가능합니다.
- 더 큰 디스크 크기 — 셔플에 최적화된 디스크는 작업자당 20GB에서 2TB까지의 디스크 크기를 지원하므로 워크로드에 따라 적절한 용량을 선택할 수 있습니다.

시작하기

워크플로에서 셔플 최적화 디스크를 사용하려면 다음 단계를 참조하십시오.

Spark

1. 다음 명령으로 EMR 서버리스 릴리스 7.1.0 애플리케이션을 생성합니다.

```
aws emr-serverless create-application \
  --type "SPARK" \
  --name my-application-name \
  --release-label emr-7.1.0 \
  --region <AWS_REGION>
```

2. 매개 변수를 `spark.emr-serverless.driver.disk.type` 포함하거나 셔플에 최적화된 디스크로 `spark.emr-serverless.executor.disk.type` 실행되도록 Spark 작업을 구성하십시오. 사용 사례에 따라 매개 변수 중 하나 또는 둘 다를 사용할 수 있습니다.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
            "entryPointArguments": ["1"],
            "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi
            --conf spark.executor.cores=4
            --conf spark.executor.memory=20g
            --conf spark.driver.cores=4
            --conf spark.driver.memory=8g
            --conf spark.executor.instances=1
            --conf spark.emr-serverless.executor.disk.type=shuffle_optimized"
```

```
        }
    }'
```

자세한 내용은 Spark 작업 속성을 참조하십시오.

Hive

1. 다음 명령을 사용하여 EMR 서버리스 릴리스 7.1.0 애플리케이션을 생성합니다.

```
aws emr-serverless create-application \
  --type "HIVE" \
  --name my-application-name \
  --release-label emr-7.1.0 \
  --region <AWS_REGION>
```

2. 매개 변수를 hive.driver.disk.type 포함하거나 셔플에 최적화된 디스크로 hive.tez.disk.type 실행되도록 Hive 작업을 구성하십시오. 사용 사례에 따라 매개 변수 중 하나 또는 둘 다를 사용할 수 있습니다.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "s3://<DOC-EXAMPLE-BUCKET>/emr-serverless-hive/query/hive-
query.ql",
            "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "hive.exec.scratchdir": "s3://<DOC-EXAMPLE-BUCKET>/emr-
serverless-hive/hive/scratch",
                "hive.metastore.warehouse.dir": "s3://<DOC-EXAMPLE-BUCKET>/emr-
serverless-hive/hive/warehouse",
                "hive.driver.cores": "2",
                "hive.driver.memory": "4g",
                "hive.tez.container.size": "4096",
                "hive.tez.cpu.vcores": "1",
                "hive.driver.disk.type": "shuffle_optimized",
```

```
                "hive.tez.disk.type": "shuffle_optimized"
            }
        }]
    }'
```

자세한 내용은 Hive 작업 속성을 참조하십시오.

사전 초기화된 용량으로 애플리케이션 구성

Amazon EMR 릴리스 7.1.0을 기반으로 애플리케이션을 생성하려면 다음 예를 참조하십시오. 이러한 애플리케이션에는 다음과 같은 속성이 있습니다.

- 사전 초기화된 Spark 드라이버 5개 (각 드라이버에는 2vCPU, 4GB 메모리, 셔플 최적화 디스크 50GB) 가 있습니다.
- 각각 4vCPU, 8GB 메모리, 셔플 최적화 디스크 500GB를 갖춘 사전 초기화된 실행기 50개

이 응용 프로그램이 Spark 작업을 실행할 때는 먼저 사전 초기화된 작업자를 사용한 다음 온디맨드 작업자를 최대 400v 용량인 1024GB까지 확장합니다. CPU 또는 중 하나의 용량을 생략할 수도 있습니다. DRIVER EXECUTOR

Spark

```
aws emr-serverless create-application \
  --type "SPARK" \
  --name <my-application-name> \
  --release-label emr-7.1.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB",
            "disk": "50GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    },
    "EXECUTOR": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB",
```

```
            "disk": "500GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
  }'
```

Hive

```
aws emr-serverless create-application \
  --type "HIVE" \
  --name <my-application-name> \
  --release-label emr-7.1.0 \
  --initial-capacity '{
    "DRIVER": {
        "workerCount": 5,
        "workerConfiguration": {
            "cpu": "2vCPU",
            "memory": "4GB",
            "disk": "50GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    },
    "EXECUTOR": {
        "workerCount": 50,
        "workerConfiguration": {
            "cpu": "4vCPU",
            "memory": "8GB",
            "disk": "500GB",
            "diskType": "SHUFFLE_OPTIMIZED"
        }
    }
  }' \
  --maximum-capacity '{
    "cpu": "400vCPU",
    "memory": "1024GB"
  }'
```

스트리밍 작업

EMR서버리스의 스트리밍 작업은 거의 실시간으로 스트리밍 데이터를 분석하고 처리할 수 있는 작업 모드입니다. 이러한 장기 실행 작업은 스트리밍 데이터를 폴링하고 데이터가 도착하는 대로 결과를 지속적으로 처리합니다. 스트리밍 작업은 실시간에 가까운 분석, 사기 탐지 및 추천 엔진과 같이 실시간 데이터 처리가 필요한 작업에 가장 적합합니다. EMR서버리스 스트리밍 작업은 내장된 작업 복원력, 실시간 모니터링, 향상된 로그 관리, 스트리밍 커넥터와의 통합과 같은 최적화를 제공합니다.

스트리밍 작업의 몇 가지 사용 사례는 다음과 같습니다.

- 실시간에 가까운 분석 — Amazon EMR Serverless의 스트리밍 작업을 사용하면 스트리밍 데이터를 거의 실시간으로 처리할 수 있으므로 로그 데이터, 센서 데이터 또는 클릭스트림 데이터와 같은 연속 데이터 스트림에 대한 실시간 분석을 수행하여 통찰력을 얻고 최신 정보를 기반으로 시기적절한 결정을 내릴 수 있습니다.
- 사기 탐지 — 데이터 스트림을 분석하고 의심스러운 패턴이나 이상 징후가 발생할 때 이를 식별할 때 스트리밍 작업을 사용하여 금융 거래, 신용 카드 작업 또는 온라인 활동에서 거의 실시간으로 사기 탐지를 실행할 수 있습니다.
- 추천 엔진 - 스트리밍 작업은 사용자 활동 데이터를 처리하고 추천 모델을 업데이트할 수 있습니다. 이렇게 하면 행동과 선호도에 따라 개인화된 실시간 추천이 가능해집니다.
- 소셜 미디어 분석 — 스트리밍 작업은 트윗, 댓글, 게시물과 같은 소셜 미디어 데이터를 처리할 수 있으므로 조직은 트렌드, 감정 분석을 모니터링하고 브랜드 평판을 거의 실시간으로 관리할 수 있습니다.
- 사물 인터넷 (IoT) 분석 — 스트리밍 작업은 IoT 장치, 센서 및 연결된 기계에서 나오는 고속 데이터 스트림을 처리하고 분석할 수 있으므로 이상 탐지, 예측 유지 관리 및 기타 IoT 분석 사용 사례를 실행할 수 있습니다.
- 클릭스트림 분석 — 스트리밍 작업은 웹 사이트 또는 모바일 애플리케이션의 클릭스트림 데이터를 처리하고 분석할 수 있습니다. 이러한 데이터를 사용하는 기업은 분석을 실행하여 사용자 행동에 대해 자세히 알아보고, 사용자 경험을 개인화하고, 마케팅 캠페인을 최적화할 수 있습니다.
- 로그 모니터링 및 분석 — 스트리밍 작업은 서버, 애플리케이션 및 네트워크 장치의 로그 데이터도 처리할 수 있습니다. 이를 통해 이상 탐지, 문제 해결, 시스템 상태 및 성능을 제공합니다.

주요 이점

EMR서버리스의 스트리밍 작업은 다음 요소의 조합인 작업 복원력을 자동으로 제공합니다.

- 자동 재시도 - EMR 서버리스는 사용자의 수동 입력 없이 실패한 모든 작업을 자동으로 재시도합니다.

- 가용 영역 (AZ) 복원력 — 원래 AZ에 문제가 발생하는 경우 EMR 서버리스는 스트리밍 작업을 정상 AZ로 자동 전환합니다.
- 로그 관리:
  - 로그 순환 — 보다 효율적인 디스크 스토리지 관리를 위해 EMR 서버리스는 긴 스트리밍 작업에 대해 로그를 주기적으로 교체합니다. 이렇게 하면 디스크 공간을 모두 차지할 수 있는 로그 누적을 방지할 수 있습니다.
  - 로그 압축 - 관리형 지속성 상태에서 로그 파일을 효율적으로 관리하고 최적화하는 데 도움이 됩니다. 또한 압축은 관리형 스파크 기록 서버를 사용할 때 디버그 경험을 개선합니다.

지원되는 데이터 소스 및 데이터 싱크

EMR서버리스는 다양한 입력 데이터 소스 및 출력 데이터 싱크와 함께 작동합니다.

- 지원되는 입력 데이터 소스 — Amazon Kinesis Data Streams, Apache Kafka용 아마존 매니지드 스트리밍, 자체 관리형 아파치 Kafka 클러스터. 기본적으로 Amazon EMR 릴리스 7.1.0 이상에는 Amazon Kinesis Data Streams 커넥터가 포함되어 있으므로 추가 패키지를 빌드하거나 다운로드할 필요가 없습니다.
- 지원되는 출력 데이터 싱크 — AWS Glue Data Catalog 테이블, Amazon S3, Amazon Redshift, MySQL, Postgre SQL 오라클, 오라클, 마이크로소프트, 아파치 아이스버그SQL, 델타 레이크, 아파치 후디

고려 사항 및 제한

스트리밍 작업을 사용할 때는 다음 고려 사항과 제한 사항을 염두에 두십시오.

- 스트리밍 작업은 Amazon EMR 릴리스 7.1.0 이상에서 지원됩니다.
- EMR서버리스는 스트리밍 작업이 오래 실행될 것으로 예상하므로 실행 제한 시간을 설정하여 작업 런타임을 제한할 수 없습니다.
- 스트리밍 작업은 구조화된 스트리밍 프레임워크 위에 구축된 Spark 엔진과만 호환됩니다.
- EMR서버리스는 스트리밍 작업을 무기한 재시도하며 최대 시도 횟수를 사용자 지정할 수 없습니다. 실패한 시도 횟수가 시간당 설정된 임계값을 초과한 경우 작업 재시도를 중지하도록 스래시 방지 기능이 자동으로 포함됩니다. 기본 임계값은 1시간 동안 5번의 시도 실패입니다. 이 임계값은 시도 횟수를 1~10회 사이로 구성할 수 있습니다. 자세한 내용은 Job Resiliance를 참조하십시오.

- 스트리밍 작업에는 런타임 상태와 진행 상황을 저장하는 체크포인트가 있으므로 EMR Serverless는 최신 체크포인트에서 스트리밍 작업을 재개할 수 있습니다. 자세한 내용은 Apache Spark 설명서의 체크포인트를 사용한 오류 복구를 참조하십시오.

스트리밍 작업 시작하기

스트리밍 작업을 시작하는 방법을 알아보려면 다음 지침을 참조하십시오.

1. Amazon EMR 서버리스 시작하기 단계를 따라 애플리케이션을 생성하십시오. 단, 애플리케이션은 Amazon EMR 릴리스 7.1.0 이상을 실행해야 합니다.
2. 애플리케이션이 준비되면 다음과 같이 스트리밍 작업을 `STREAMING` 제출하도록 `mode` 파라미터를 설정합니다. AWS CLI 예시.

```
aws emr-serverless start-job-run \
--application-id <APPPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<streaming script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
            --conf spark.executor.memory=16g
            --conf spark.driver.cores=4
            --conf spark.driver.memory=16g
            --conf spark.executor.instances=3"
    }
}'
```

지원되는 스트리밍 커넥터

스트리밍 커넥터를 사용하면 스트리밍 소스에서 데이터를 쉽게 읽을 수 있으며 스트리밍 싱크에 데이터를 쓸 수도 있습니다.

지원되는 스트리밍 커넥터는 다음과 같습니다.

아마존 Kinesis Data Streams 커넥터

Apache Spark용 Amazon Kinesis Data Streams 커넥터를 사용하면 Amazon Kinesis Data Streams의 데이터를 소비하고 Amazon Kinesis Data Streams에 데이터를 쓰는 스트리밍 애플리케이션 및 파이프

라인을 구축할 수 있습니다. 커넥터는 샤드당 최대 2MB/초의 전용 읽기 처리량으로 향상된 팬아웃 사용을 지원합니다. 기본적으로 Amazon EMR Serverless 7.1.0 이상에는 커넥터가 포함되어 있으므로 추가 패키지를 빌드하거나 다운로드할 필요가 없습니다. 커넥터에 대한 자세한 내용은 의 spark-sql-kinesis-connector 페이지를 참조하십시오. GitHub

다음은 Kinesis Data Streams 커넥터 종속성을 사용하여 작업 실행을 시작하는 방법의 예입니다.

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<Kinesis-streaming-script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
                --conf spark.executor.memory=16g
                --conf spark.driver.cores=4
                --conf spark.driver.memory=16g
                --conf spark.executor.instances=3
                --jars /usr/share/aws/kinesis/spark-sql-kinesis/lib/spark-streaming-
sql-kinesis-connector.jar"
    }
}'
```

Kinesis Data Streams에 연결하려면 액세스 권한이 VPC 있는 서버리스 애플리케이션을 구성하고 EMR 엔드포인트를 사용하여 VPC 프라이빗 액세스를 허용하거나 게이트웨이를 사용하여 NAT 퍼블릭 액세스를 허용해야 합니다. 자세한 내용은 액세스 구성을 참조하십시오. VPC 또한 작업 런타임 역할에 필요한 데이터 스트림에 액세스하는 데 필요한 읽기 및 쓰기 권한이 있는지 확인해야 합니다. 작업 런타임 역할을 구성하는 방법에 대한 자세한 내용은 Amazon EMR Serverless의 작업 런타임 역할을 참조하십시오. 필요한 모든 권한의 전체 목록은 의 spark-sql-kinesis-connector 페이지를 참조하십시오. GitHub

아파치 카프카 커넥터

Spark 구조화된 스트리밍용 Apache Kafka 커넥터는 Spark 커뮤니티의 오픈 소스 커넥터이며 Maven 리포지토리에서 사용할 수 있습니다. 이 커넥터를 사용하면 Spark 구조적 스트리밍 애플리케이션이 자체 관리형 Apache Kafka 및 Amazon Managed Streaming for Apache Kafka에서 데이터를 읽고 쓸 수 있습니다. 커넥터에 대한 자세한 내용은 Apache Spark 설명서의 구조적 스트리밍+Kafka 통합 가이드를 참조하십시오.

다음 예제는 작업 실행 요청에 Kafka 커넥터를 포함하는 방법을 보여줍니다.

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<Kafka-streaming-script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
                --conf spark.executor.memory=16g
                --conf spark.driver.cores=4
                --conf spark.driver.memory=16g
                --conf spark.executor.instances=3
                --packages org.apache.spark:spark-sql-
kafka-0-10_2.12:<KAFKA_CONNECTOR_VERSION>"
    }
}'
```

Apache Kafka 커넥터 버전은 EMR 서버리스 릴리스 버전과 해당 Spark 버전에 따라 다릅니다. 올바른 Kafka 버전을 찾으려면 구조적 스트리밍 + Kafka 통합 가이드를 참조하십시오.

인증과 함께 Apache Kafka용 Amazon 관리형 스트리밍을 사용하려면 IAM Kafka 커넥터를 Amazon에 연결할 수 있도록 다른 종속성을 포함해야 합니다. MSK IAM 자세한 내용은 의 리포지토리를 참조하십시오. aws-msk-iam-auth GitHub 또한 작업 런타임 역할에 필요한 IAM 권한이 있는지 확인해야 합니다. 다음 예제는 커넥터를 IAM 인증과 함께 사용하는 방법을 보여줍니다.

```
aws emr-serverless start-job-run \
--application-id <APPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--mode 'STREAMING' \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "s3://<Kafka-streaming-script>",
        "entryPointArguments": ["s3://<DOC-EXAMPLE-BUCKET-OUTPUT>/output"],
        "sparkSubmitParameters": "--conf spark.executor.cores=4
                --conf spark.executor.memory=16g
                --conf spark.driver.cores=4
                --conf spark.driver.memory=16g
                --conf spark.executor.instances=3
```

```
                --packages org.apache.spark:spark-sql-
kafka-0-10_2.12:<KAFKA_CONNECTOR_VERSION>,software.amazon.msk:aws-msk-iam-
auth:<MSK_IAM_LIB_VERSION>"
    }
}'
```

Kafka 커넥터와 Amazon의 IAM 인증 라이브러리를 사용하려면 액세스 권한이 MSK 있는 EMR 서버리스 애플리케이션을 구성해야 합니다. VPC Maven 종속성에 액세스하려면 서브넷이 인터넷에 액세스할 수 있어야 하고 NAT 게이트웨이를 사용해야 합니다. 자세한 내용은 액세스 구성을 참조하십시오. VPC Kafka 클러스터에 액세스하려면 서브넷이 네트워크에 연결되어 있어야 합니다. 이는 Kafka 클러스터가 자체 관리형인지 또는 Apache Kafka용 Amazon Managed Streaming을 사용하든 관계없이 적용됩니다.

스트리밍 작업 로그 관리

로그 로테이션

스트리밍 작업은 Spark 애플리케이션 로그 및 이벤트 로그의 로그 로테이션을 지원합니다. 로그 순환은 긴 스트리밍 작업에서 사용 가능한 디스크 공간을 모두 차지할 수 있는 대용량 로그 파일을 생성하는 것을 방지합니다. 로그 회전은 디스크 저장소를 절약하고 디스크 공간 부족으로 인한 작업 실패를 방지하는 데 도움이 됩니다. 자세한 내용은 로그 순환을 참조하십시오.

로그 압축

스트리밍 작업은 관리형 로깅을 사용할 수 있을 때마다 Spark 이벤트 로그의 로그 압축도 지원합니다. 관리형 로깅에 대한 자세한 내용은 관리형 스토리지를 사용한 로깅을 참조하십시오. 스트리밍 작업은 오랫동안 실행될 수 있으며 시간이 지남에 따라 이벤트 데이터의 양이 누적되어 로그 파일 크기가 크게 증가할 수 있습니다. Spark 히스토리 서버는 이러한 이벤트를 읽고 Spark 애플리케이션 UI용 메모리에 로드합니다. 이 프로세스는 특히 Amazon S3에 저장된 이벤트 로그가 매우 큰 경우 높은 지연 시간과 비용을 초래할 수 있습니다.

로그를 압축하면 이벤트 로그 크기가 줄어들기 때문에 Spark History Server는 한 번에 1GB 이상의 이벤트 로그를 로드하지 않아도 됩니다. 자세한 내용은 Apache Spark 설명서의 모니터링 및 계측을 참조하십시오.

스파크 잡스

`type`파라미터가 로 설정된 애플리케이션에서 Spark 작업을 실행할 수 있습니다. SPARK 잡스는 Amazon EMR 릴리스 버전과 호환되는 Spark 버전과 호환되어야 합니다. 예를 들어 Amazon EMR 릴

리스 6.6.0에서 작업을 실행하는 경우 작업은 Apache Spark 3.2.0과 호환되어야 합니다. 각 릴리스의 애플리케이션 버전에 대한 자세한 내용은 을 참조하십시오. Amazon EMR 서버리스 릴리스 버전

Spark 작업 매개변수

를 StartJobRunAPI사용하여 Spark 작업을 실행할 때 다음 매개변수를 지정할 수 있습니다.

필수 파라미터

- 스파크 작업 런타임 역할
- Spark 작업 드라이버 파라미터
- Spark 구성 오버라이드 파라미터
- Spark 동적 리소스 할당 최적화

스파크 작업 런타임 역할

애플리케이션이 ARN Spark 작업을 실행하는 **executionRoleArn**데 사용하는 IAM 역할을 지정하는 데 사용합니다. 이 역할에는 다음 권한이 포함되어야 합니다.

- 데이터가 있는 S3 버킷 또는 기타 데이터 소스에서 읽기
- 스크립트나 파일이 PySpark 있는 S3 버킷 또는 접두사에서 읽습니다. JAR
- 최종 출력을 작성하려는 S3 버킷에 기록합니다.
- 다음을 지정하는 S3 버킷 또는 접두사에 로그를 기록합니다. S3MonitoringConfiguration
- KMSKMS키를 사용하여 S3 버킷의 데이터를 암호화하는 경우 키에 대한 액세스
- 에 대한 액세스 AWS Spark를 사용하는 경우 Glue 데이터 카탈로그 SQL

Spark 작업에서 다른 데이터 소스에서 데이터를 읽거나 다른 데이터 소스에서 데이터를 쓰는 경우 이 IAM 역할에 적절한 권한을 지정하십시오. IAM역할에 이러한 권한을 제공하지 않으면 작업이 실패할 수 있습니다. 자세한 내용은 Amazon EMR 서버리스의 Job 런타임 역할 및 로그 저장 단원을 참조하세요.

Spark 작업 드라이버 파라미터

작업에 입력을 제공하는 **jobDriver**데 사용합니다. 작업 드라이버 매개 변수는 실행하려는 작업 유형에 대해 하나의 값만 허용합니다. Spark 작업의 경우 매개 변수 값은 입니다sparkSubmit. 이 작업 유형을 사용하여 Spark 제출을 통해 Scala, Java PySpark, SparkR 및 기타 지원되는 작업을 실행할 수 있습니다. Spark 작업에는 다음과 같은 매개 변수가 있습니다.

- **sparkSubmitParameters**— 작업에 전송하려는 추가 Spark 매개 변수입니다. 이 매개 변수를 사용하면 드라이버 메모리 또는 실행기 수와 같은 기본 Spark 속성 (예: or 인수에 정의된 속성) 을 재정의할 수 있습니다. `--conf` `--class`
- **entryPointArguments**— 기본 JAR 파일이나 Python 파일에 전달하려는 인수의 배열입니다. entrypoint 코드를 사용하여 이러한 파라미터를 읽는 작업을 처리해야 합니다. 배열의 각 인수를 쉼표로 구분합니다.
- **entryPoint**— 실행하려는 기본 JAR 또는 Python 파일에 대한 Amazon S3의 참조입니다. Scala 또는 JAR Java를 실행 중인 경우 `--class` 인수 `SparkSubmitParameters` 사용에 기본 항목 클래스를 지정하십시오.

자세한 내용은 Launching Applications with spark-submit을 참조하세요.

Spark 구성 오버라이드 파라미터

모니터링 수준 **configurationOverrides**및 애플리케이션 수준 구성 속성을 재정의하는 데 사용합니다. 이 매개 변수는 다음 두 필드가 있는 JSON 개체를 받아들입니다.

- **monitoringConfiguration**- 이 필드를 사용하여 EMR 서버리스 작업에서 Spark 작업의 로그를 저장할 Amazon S3 URL (`s3MonitoringConfiguration`) 를 지정합니다. 동일한 버킷을 사용하여 이 버킷을 생성했는지 확인하십시오. AWS 계정 애플리케이션을 호스팅하는 곳이고 같은 곳에 AWS 리전 작업이 실행되고 있는 위치.
- **applicationConfiguration**— 애플리케이션의 기본 구성을 재정의하려면 이 필드에 구성 객체를 제공할 수 있습니다. 간단한 구문을 사용하여 구성을 제공하거나 파일의 구성 객체를 참조할 수 있습니다. JSON 구성 객체는 분류, 속성 및 선택적 중첩 구성으로 이루어져 있습니다. 속성은 해당 파일에서 재정의하려는 설정으로 구성됩니다. 단일 개체에 여러 응용 프로그램에 대해 여러 분류를 지정할 수 있습니다. JSON

> **Note**
>
> 사용 가능한 구성 분류는 특정 EMR 서버리스 릴리스에 따라 다릅니다. 예를 들어 사용자 지정 Log4j에 대한 `spark-executor-log4j2` 분류는 릴리스 `spark-driver-log4j2` 6.8.0 이상에서만 사용할 수 있습니다.

애플리케이션 오버라이드와 Spark 제출 매개 변수에서 동일한 구성을 사용하는 경우 Spark 제출 매개 변수가 우선합니다. 구성의 우선 순위는 가장 높은 것부터 가장 낮은 것까지 다음과 같습니다.

- EMR서버리스가 생성 SparkSession 시 제공하는 구성.
- --conf인수와 sparkSubmitParameters 함께 제공하는 구성.
- 응용 프로그램의 일부로 제공하는 구성은 작업을 시작할 때 재정의됩니다.
- 응용 프로그램을 만들 runtimeConfiguration 때 구성의 일부로 제공하는 구성.
- Amazon이 릴리스에 EMR 사용하는 구성을 최적화했습니다.
- 애플리케이션의 기본 오픈 소스 구성.

애플리케이션 수준에서 구성을 선언하고 작업 실행 중 구성을 재정의하는 방법에 대한 자세한 내용은 을 참조하십시오. 서버리스의 기본 애플리케이션 구성 EMR

Spark 동적 리소스 할당 최적화

EMR서버리스에서 리소스 사용을 최적화하는 dynamicAllocationOptimization 데 사용합니다. Spark 구성 true 분류에서 이 속성을 로 설정하면 Spark가 실행자를 요청하고 취소하는 속도를 EMR 서버리스가 작업자를 생성하고 해제하는 속도에 더 잘 맞도록 실행기 리소스 할당을 최적화하라는 의미입니다. EMR 이렇게 함으로써 EMR 서버리스는 여러 단계에서 작업자를 더 최적으로 재사용하므로 동일한 성능을 유지하면서 여러 단계로 구성된 작업을 실행할 때 비용을 절감할 수 있습니다.

이 속성은 모든 Amazon EMR 릴리스 버전에서 사용할 수 있습니다.

다음은 를 사용한 샘플 구성 dynamicAllocationOptimization 분류입니다.

```
[
  {
    "Classification": "spark",
    "Properties": {
      "dynamicAllocationOptimization": "true"
    }
  }
]
```

동적 할당 최적화를 사용하는 경우 다음 사항을 고려하세요.

- 이 최적화는 동적 리소스 할당을 활성화한 Spark 작업에 사용할 수 있습니다.
- 비용 효율성을 극대화하려면 작업량에 따라 작업 수준 설정 spark.dynamicAllocation.maxExecutors 또는 애플리케이션 수준의 최대 용량 설정을 사용하여 작업자에 대한 상한선을 구성하는 것이 좋습니다.

- 단순한 작업에서는 비용 개선을 보지 못할 수도 있습니다. 예를 들어 작업이 작은 데이터세트에서 실행되거나 한 단계에서 실행이 완료되는 경우 Spark에는 더 많은 수의 실행자나 여러 스케일링 이벤트가 필요하지 않을 수 있습니다.
- 큰 단계, 작은 단계, 다시 큰 단계로 이어지는 시퀀스의 작업은 작업 런타임이 저하될 수 있습니다. EMR서버리스는 리소스를 더 효율적으로 사용하므로 대규모 단계에서 사용할 수 있는 작업자 수가 줄어들어 런타임이 길어질 수 있습니다.

스파크 작업 속성

다음 표에는 Spark 작업을 제출할 때 재정의할 수 있는 선택적 Spark 속성과 해당 기본값이 나와 있습니다.

키	설명	기본값
`spark.archives`	Spark가 각 실행자의 작업 디렉터리에 추출하는 쉼표로 구분된 아카이브 목록입니다. 지원되는 파일 유형에는,, 등이 있습니다. `.jar` `.tar.gz` `.tgz` `.zip` 추출할 디렉터리 이름을 지정하려면 추출하려는 파일 이름 # 뒤에 추가합니다. 예: `file.zip#directory` .	NULL
`spark.authenticate`	Spark의 내부 연결 인증을 켜는 옵션입니다.	TRUE
`spark.driver.cores`	드라이버가 사용하는 코어 수	4
`spark.driver.extraJavaOptions`	Spark 드라이버용 추가 Java 옵션.	NULL
`spark.driver.memory`	드라이버가 사용하는 메모리 양	14G
`spark.dynamicAllocation.enabled`	동적 리소스 할당을 활성화하는 옵션. 이 옵션은 워크로드에 따라 애플리케이션에 등록된	TRUE

키	설명	기본값
	실행자 수를 늘리거나 줄입니다.	
spark.dynamicAllocation.executorIdleTimeout	Spark에서 실행기를 제거하기 전까지 실행기가 유휴 상태를 유지할 수 있는 시간입니다. 이는 동적 할당을 활성화한 경우에만 적용됩니다.	60초
spark.dynamicAllocation.initialExecutors	동적 할당을 활성화한 경우 실행할 초기 실행자 수입니다.	3
spark.dynamicAllocation.maxExecutors	동적 할당을 활성화한 경우 실행자 수의 상한값입니다.	6.10.0 이상의 경우 infinity 6.9.0 이하의 경우 100
spark.dynamicAllocation.minExecutors	동적 할당을 활성화한 경우의 실행자 수 하한입니다.	0
spark.emr-serverless.allocation.batch.size	실행자 할당의 각 주기에서 요청할 컨테이너의 수. 각 할당 주기 사이에는 1초의 간격이 있습니다.	20
spark.emr-serverless.driver.disk	Spark 드라이버 디스크.	20G
spark.emr-serverless.driverEnv. *[KEY]*	Spark 드라이버에 환경 변수를 추가하는 옵션입니다.	NULL
spark.emr-serverless.executor.disk	Spark 실행기 디스크.	20G
spark.emr-serverless.memoryOverheadFactor	드라이버 및 실행기 컨테이너 메모리에 추가할 메모리 오버헤드를 설정합니다.	0.1

키	설명	기본값
spark.emr-serverless.driver.disk.type	Spark 드라이버에 연결된 디스크 유형입니다.	표준
spark.emr-serverless.executor.disk.type	Spark 실행기에 연결된 디스크 유형.	표준
spark.executor.cores	각 실행기가 사용하는 코어 수	4
spark.executor.extraJavaOptions	Spark 실행기를 위한 추가 Java 옵션.	NULL
spark.executor.instances	할당할 Spark 실행기 컨테이너의 수입니다.	3
spark.executor.memory	각 실행기가 사용하는 메모리 양	14G
spark.executorEnv.*[KEY]*	Spark 실행기에 환경 변수를 추가하는 옵션입니다.	NULL
spark.files	각 실행자의 작업 디렉토리에 저장할 쉼표로 구분된 파일 목록입니다. 실행기에서 이러한 파일의 파일 경로에 액세스할 수 있습니다. SparkFiles.get(*fileName*)	NULL
spark.hadoop.hive.metastore.client.factory.class	Hive 메타스토어 구현 클래스.	NULL
spark.jars	드라이버 및 실행기의 런타임 클래스 경로에 추가할 추가 jar	NULL

키	설명	기본값
spark.network.crypto.enabled	기반 암호화를 AES 켜는 옵션입니다. RPC 여기에는 Spark 2.2.0에 추가된 인증 프로토콜이 포함됩니다.	FALSE
spark.sql.warehouse.dir	관리형 데이터베이스 및 테이블의 기본 위치입니다.	의 값 $PWD/spark-warehouse
spark.submit.pyFiles	PYTHONPATH Python용 앱에 배치할 .zip.egg, 또는 .py 파일을 쉼표로 구분한 목록입니다.	NULL

다음 표에는 기본 Spark 제출 매개변수가 나열되어 있습니다.

키	설명	기본값
archives	Spark가 각 실행자의 작업 디렉터리에 추출하는 쉼표로 구분된 아카이브 목록입니다.	NULL
class	애플리케이션의 기본 클래스 (Java 및 Scala 앱용).	NULL
conf	임의의 Spark 구성 속성.	NULL
driver-cores	드라이버가 사용하는 코어 수	4
driver-memory	드라이버가 사용하는 메모리 양.	14G
executor-cores	각 실행자가 사용하는 코어 수	4
executor-memory	실행자가 사용하는 메모리 양.	14G
files	각 실행자의 작업 디렉토리에 넣을 쉼표로 구분된 파일	NULL

키	설명	기본값
	목록. 실행기에서 이러한 파일의 파일 경로에 액세스할 수 있습니다. SparkFiles.get(*fileName*)	
jars	드라이버 및 실행기 클래스 경로에 포함할 쉼표로 구분된 jar 목록입니다.	NULL
num-executors	실행할 실행자 수.	3
py-files	PYTHONPATH Python 앱용 앱에 배치할 .zip.egg, 또는 .py 파일을 쉼표로 구분한 목록입니다.	NULL
verbose	추가 디버그 출력을 켜는 옵션입니다.	NULL

스파크 예제

다음 예제는 를 사용하여 Python 스크립트를 실행하는 StartJobRun API 방법을 보여줍니다. 이 예제를 사용하는 end-to-end 자습서는 을 참조하십시오Amazon EMR 서버리스 시작하기. EMR서버리스 샘플 GitHub 리포지토리에서 PySpark 작업을 실행하고 Python 종속성을 추가하는 방법에 대한 추가 예제를 찾을 수 있습니다.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/
wordcount/scripts/wordcount.py",
            "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET-OUTPUT/wordcount_output"],
            "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
 spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g --
conf spark.executor.instances=1"
```

```
        }
    }'
```

다음 예제는 를 사용하여 Spark를 실행하는 StartJobRun API 방법을 보여줍니다. JAR

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
            "entryPointArguments": ["1"],
            "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.executor.cores=4 --conf spark.executor.memory=20g --conf spark.driver.cores=4 --
conf spark.driver.memory=8g --conf spark.executor.instances=1"
        }
    }'
```

하이브 잡스

type매개 변수가 로 설정된 애플리케이션에서 Hive 작업을 실행할 수 있습니다. HIVE 작업은 Amazon EMR 릴리스 버전과 호환되는 Hive 버전과 호환되어야 합니다. 예를 들어 Amazon EMR 릴리스 6.6.0이 설치된 애플리케이션에서 작업을 실행하는 경우 작업은 Apache Hive 3.1.2와 호환되어야 합니다. 각 릴리스의 애플리케이션 버전에 대한 자세한 내용은 을 참조하십시오. Amazon EMR 서버리스 릴리스 버전

Hive 작업 매개 변수

를 사용하여 Hive 작업을 실행하는 경우 다음 매개 변수를 지정해야 합니다. StartJobRunAPI

필수 파라미터

- Hive 작업 런타임 역할
- Hive 작업 드라이버 파라미터
- Hive 구성 오버라이드 파라미터

Hive 작업 런타임 역할

애플리케이션이 Hive 작업을 실행하는 **executionRoleArn**데 사용하는 IAM 역할을 지정하는 데 사용합니다. ARN 이 역할에는 다음 권한이 포함되어야 합니다.

- 데이터가 있는 S3 버킷 또는 기타 데이터 소스에서 읽기
- Hive 쿼리 파일 및 init 쿼리 파일이 있는 S3 버킷 또는 접두사에서 읽습니다.
- Hive Scratch 디렉터리와 Hive 메타스토어 웨어하우스 디렉터리가 있는 S3 버킷을 읽고 쓸 수 있습니다.
- 최종 출력을 기록하려는 S3 버킷에 기록하십시오.
- 다음을 지정하는 S3 버킷 또는 접두사에 로그를 기록합니다. `S3MonitoringConfiguration`
- KMSKMS키를 사용하여 S3 버킷의 데이터를 암호화하는 경우 키에 대한 액세스
- 에 대한 액세스 AWS Glue Data 카탈로그

Hive 작업에서 다른 데이터 소스에서 데이터를 읽거나 다른 데이터 소스에서 데이터를 쓰는 경우 이 IAM 역할에 적절한 권한을 지정하십시오. IAM역할에 이러한 권한을 제공하지 않으면 작업이 실패할 수 있습니다. 자세한 내용은 Amazon EMR 서버리스의 Job 런타임 역할 단원을 참조하십시오.

Hive 작업 드라이버 파라미터

작업에 입력을 제공하는 **`jobDriver`**데 사용합니다. 작업 드라이버 매개 변수는 실행하려는 작업 유형에 대해 하나의 값만 허용합니다. 작업 `hive` 유형으로 지정하는 경우 EMR 서버리스는 Hive 쿼리를 `jobDriver` 매개 변수에 전달합니다. Hive 작업에는 다음과 같은 매개 변수가 있습니다.

- **`query`**— 실행하려는 Hive 쿼리 파일에 대한 Amazon S3의 참조입니다.
- **`parameters`**— 재정의하려는 추가 Hive 구성 속성은 다음과 같습니다. 속성을 재정의하려면 속성을 이 매개 변수에 as로 전달하십시오. `--hiveconf` *`property=value`* 변수를 재정의하려면 변수를 이 매개 변수에 로 전달하십시오. `--hivevar` *`key=value`*
- **`initQueryFile`**— init Hive 쿼리 파일입니다. Hive는 쿼리 전에 이 파일을 실행하며 이 파일을 사용하여 테이블을 초기화할 수 있습니다.

Hive 구성 오버라이드 파라미터

모니터링 수준 **`configurationOverrides`**및 애플리케이션 수준 구성 속성을 재정의하는 데 사용합니다. 이 매개 변수는 다음 두 필드가 있는 JSON 개체를 받아들입니다.

- **`monitoringConfiguration`**— 이 필드를 사용하여 EMR 서버리스 작업에서 Hive 작업의 로그를 저장할 Amazon S3 URL (`s3MonitoringConfiguration`) 를 지정합니다. 동일한 버킷을 사용하여 이 버킷을 생성했는지 확인하십시오. AWS 계정 애플리케이션을 호스팅하는 곳이고 같은 곳에 AWS 리전 작업이 실행되고 있는 위치

- **`applicationConfiguration`**— 이 필드에 구성 객체를 제공하여 애플리케이션의 기본 구성을 재정의할 수 있습니다. 간단한 구문을 사용하여 구성을 제공하거나 파일의 구성 객체를 참조할 수 있습니다. JSON 구성 객체는 분류, 속성 및 선택적 중첩 구성으로 이루어져 있습니다. 속성은 해당 파일에서 재정의하려는 설정으로 구성됩니다. 단일 개체에 여러 응용 프로그램에 대해 여러 분류를 지정할 수 있습니다. JSON

> **Note**
>
> 사용 가능한 구성 분류는 특정 EMR 서버리스 릴리스에 따라 다릅니다. 예를 들어 사용자 지정 Log4j에 대한 `spark-executor-log4j2` 분류는 릴리스 `spark-driver-log4j2` 6.8.0 이상에서만 사용할 수 있습니다.

애플리케이션 오버라이드와 Hive 매개 변수에서 동일한 구성을 전달하면 Hive 매개 변수가 우선 적용됩니다. 다음 목록은 구성 순위를 가장 높은 우선 순위에서 가장 낮은 우선 순위로 나열합니다.

- Hive 파라미터의 일부로 제공하는 구성. `--hiveconf` *`property=value`*
- 애플리케이션의 일부로 제공하는 구성은 작업을 시작할 때 재정의됩니다.
- 응용 프로그램을 만들 `runtimeConfiguration` 때 구성의 일부로 제공하는 구성.
- Amazon이 릴리스에 EMR 할당하는 최적화된 구성.
- 애플리케이션의 기본 오픈 소스 구성.

애플리케이션 수준에서 구성을 선언하고 작업 실행 중 구성을 재정의하는 방법에 대한 자세한 내용은 을 참조하십시오. 서버리스의 기본 애플리케이션 구성 EMR

Hive 작업 속성

다음 표에는 Hive 작업을 제출할 때 구성해야 하는 필수 속성이 나와 있습니다.

설정	설명
`hive.exec.scratchdir`	EMR서버리스가 Hive 작업 실행 중에 임시 파일을 생성하는 Amazon S3 위치입니다.
`hive.metastore.warehouse.dir`	Hive의 관리형 테이블에 대한 데이터베이스의 Amazon S3 위치입니다.

다음 표에는 Hive 작업을 제출할 때 재정의할 수 있는 선택적 Hive 속성과 해당 기본값이 나와 있습니다.

설정	설명	기본값
`fs.s3.customAWSCredentialsProvider`	The AWS 사용하려는 자격 증명 제공자.	com.Amazonaws.Auth.d efaultAWSCredentials ProviderChain
`fs.s3a.aws.credentials.provider`	더 AWS S3A 파일 시스템에서 사용하려는 자격 증명 공급자.	com.amazonaws.Auth.d efaultAWSCredentials ProviderChain
`hive.auto.convert.join`	입력 파일 크기에 따라 공통 조인을 맵조인으로 자동 변환하는 옵션입니다.	TRUE
`hive.auto.convert.join.noconditionaltask`	Hive가 입력 파일 크기에 따라 공통 조인을 맵조인으로 변환할 때 최적화를 켜는 옵션입니다.	TRUE
`hive.auto.convert.join.noconditionaltask.size`	조인은 이 크기 미만의 맵조인으로 직접 변환됩니다.	최적값은 Tez 작업 메모리를 기반으로 계산됩니다.
`hive.cbo.enable`	Calcite 프레임워크를 사용하여 비용 기반 최적화를 활성화하는 옵션입니다.	TRUE
`hive.cli.tez.session.async`	Hive 쿼리가 컴파일되는 동안 백그라운드 Tez 세션을 시작하는 옵션입니다. 로 설정하면 Hive `false` 쿼리가 컴파일된 후 Tez AM이 실행됩니다.	TRUE
`hive.compute.query.using.stats`	메타스토어에 저장된 통계로 특정 쿼리에 응답하도록 Hive를 활성화하는 옵션입니다. 기	TRUE

설정	설명	기본값
	본 통계의 경우 로 설정합니다. hive.stats.autogather TRUE 고급 쿼리 컬렉션을 보려면 를 analyze table queries 실행하세요.	
hive.default.fileformat	CREATE TABLE명령문의 기본 파일 형식입니다. STORED AS [FORMAT]명령에서 지정하는 경우 이를 명시적으로 재정의할 수 있습니다. CREATE TABLE	TEXTFILE
hive.driver.cores	Hive 드라이버 프로세스에 사용할 코어 수.	2
hive.driver.disk	Hive 드라이버의 디스크 크기.	20G
hive.driver.disk.type	Hive 드라이버의 디스크 유형.	표준
hive.tez.disk.type	tez 작업자의 디스크 크기.	표준
hive.driver.memory	Hive 드라이버 프로세스당 사용할 메모리 양 Hive CLI 및 Tez 애플리케이션 마스터는 이 메모리를 20% 의 헤드룸과 동일하게 공유합니다.	6G
hive.emr-serverless.launch.env.[*KEY*]	Hive 드라이버, Tez AM 및 Tez 작업과 같은 모든 Hive 관련 프로세스에서 *KEY* 환경 변수를 설정하는 옵션입니다.	
hive.exec.dynamic.partition	/에서 동적 파티션을 켜는 옵션. DML DDL	TRUE

설정	설명	기본값
hive.exec.dynamic.partition.mode	엄격 모드를 사용할지 아니면 비엄격 모드를 사용할지를 지정하는 옵션입니다. 엄격 모드에서는 실수로 모든 파티션을 덮어쓸 경우를 대비하여 하나 이상의 정적 파티션을 지정해야 합니다. 비엄격 모드에서는 모든 파티션을 동적으로 사용할 수 있습니다.	strict
hive.exec.max.dynamic.partitions	Hive가 생성하는 총 동적 파티션의 최대 수입니다.	1000
hive.exec.max.dynamic.partitions.pernode	Hive가 각 매퍼 및 리듀서 노드에 생성하는 동적 파티션의 최대 수입니다.	100
hive.exec.orc.split.strategy	BI, ETL 또는 값 중 하나가 필요합니다. HYBRID 이는 사용자 수준 구성이 아닙니다. BI쿼리 실행 대신 분할 생성에 소요되는 시간을 줄이도록 지정합니다. ETL분할 생성에 더 많은 시간을 할애하도록 지정합니다. HYBRID휴리스틱을 기반으로 위 전략 중 하나를 선택하도록 지정합니다.	HYBRID
hive.exec.reducers.bytes.per.reducer	감속기당 크기. 기본값은 256MB입니다. 입력 크기가 1G인 경우 작업에는 4개의 리듀서가 사용됩니다.	256000000
hive.exec.reducers.max	리듀서의 최대 개수.	256

설정	설명	기본값
hive.exec.stagingdir	Hive가 테이블 위치 내부 및 속성에 지정된 스크래치 디렉터리 위치에 생성하는 임시 파일을 저장하는 디렉터리의 이름입니다. hive.exec.scratchdir	.hive-staging
hive.fetch.task.conversion	NONEMINIMAL, 또는 MORE 값 중 하나가 필요합니다. Hive는 일부 쿼리를 단일 FETCH 작업으로 변환할 수 있습니다. 이렇게 하면 지연 시간이 최소화됩니다.	MORE
hive.groupby.position.alias	Hive가 명령문에서 열 위치 별칭을 사용하도록 하는 옵션입니다. GROUP BY	FALSE
hive.input.format	기본 입력 형식입니다. 에서 문제가 발생할 HiveInputFormat 경우 로 설정하십시오CombineHiveInputFormat .	org.apache.hadoop.hive.ql.io.CombineHiveInputFormat
hive.log.explain.output	Hive 로그에 있는 모든 쿼리의 확장 출력 설명을 켜는 옵션입니다.	FALSE
hive.log.level	하이브 로깅 수준.	INFO
hive.mapred.reduce.tasks.speculative.execution	리듀서의 투기 출시를 활성화하는 옵션입니다. Amazon EMR 6.10.x 이하에서만 지원됩니다.	TRUE

설정	설명	기본값
hive.max-task-containers	최대 동시 컨테이너 수. 구성된 매퍼 메모리에 이 값을 곱하여 계산과 작업 선점 사용을 분할하는 사용 가능한 메모리를 결정합니다.	1000
hive.merge.mapfiles	맵 전용 작업 종료 시 작은 파일을 병합하도록 하는 옵션입니다.	TRUE
hive.merge.size.per.task	작업 종료 시 병합된 파일의 크기.	256000000
hive.merge.tezfiles	테즈 종료 시 작은 파일을 병합할 수 있도록 설정하는 옵션입니다. DAG	FALSE
hive.metastore.client.factory.class	IMetaStoreClient 인터페이스를 구현하는 객체를 생성하는 팩토리 클래스의 이름.	com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory
hive.metastore.glue.catalogid	만약 AWS Glue Data Catalog는 메타스토어로 작동하지만 다른 곳에서 실행됩니다. AWS 계정 작업 이외의 ID는 AWS 계정 작업이 실행되고 있는 위치.	NULL
hive.metastore.uris	메타스토어 클라이언트가 URI 원격 메타스토어에 연결하는 데 사용하는 쓰리프트입니다.	NULL
hive.optimize.ppd	조건부 푸시다운을 켜는 옵션입니다.	TRUE

설정	설명	기본값
hive.optimize.ppd.storage	스토리지 핸들러에 대한 조건부 푸시다운을 활성화하는 옵션입니다.	TRUE
hive.orderby.position.alias	Hive가 명령문에서 열 위치 별칭을 사용하도록 하는 옵션입니다. ORDER BY	TRUE
hive.prewarm.enabled	Tez의 컨테이너 프리워밍을 켜는 옵션입니다.	FALSE
hive.prewarm.numcontainers	Tez를 위해 프리워밍할 컨테이너의 개수	10
hive.stats.autogather	명령 실행 중에 Hive가 기본 통계를 자동으로 수집하도록 하는 옵션입니다. INSERT OVERWRITE	TRUE
hive.stats.fetch.column.stats	메타스토어에서 열 통계 가져오기를 끄는 옵션입니다. 열 수가 많으면 열 통계를 가져오는 데 비용이 많이 들 수 있습니다.	FALSE
hive.stats.gather.num.threads	partialscan 및 noscan 분석 명령이 파티션을 나눈 테이블에 사용하는 스레드 수입니다. 이는 구현하는 파일 형식 StatsProvidingRecordReader (예:ORC) 에만 적용됩니다.	10
hive.strict.checks.cartesian.product	엄격한 데카르트 조인 검사를 활성화하는 옵션. 이러한 검사는 데카르트 곱 (크로스 조인)을 허용하지 않습니다.	FALSE

설정	설명	기본값
hive.strict.checks.type.safety	엄격한 유형 안전 검사를 켜고 및 둘 bigint 다와의 비교를 해제하는 옵션입니다. string double	TRUE
hive.support.quoted.identifiers	NONE또는 값을 COLUMN 예상합니다. NONE식별자에는 영숫자와 밑줄 문자만 사용할 수 있음을 의미합니다. COLUMN열 이름에는 모든 문자가 포함될 수 있음을 의미합니다.	COLUMN
hive.tez.auto.reducer.parallelism	Tez 자동 감속기 병렬 기능을 켜는 옵션입니다. Hive는 여전히 데이터 크기를 추정하고 병렬 처리 추정치를 설정합니다. Tez는 소스 꼭짓점의 출력 크기를 샘플링하고 필요에 따라 런타임에 추정치를 조정합니다.	TRUE
hive.tez.container.size	Tez 작업 프로세스당 사용할 메모리 양.	6144
hive.tez.cpu.vcores	각 Tez 작업에 사용할 코어 수	2
hive.tez.disk.size	각 태스크 컨테이너의 디스크 크기.	20G
hive.tez.input.format	Tez AM에서 스플릿 생성을 위한 입력 형식입니다.	org.apache.hadoop.hive.ql.io.HiveInputFormat
hive.tez.min.partition.factor	자동 감속기 병렬 처리를 켤 때 Tez가 지정하는 리듀서의 하한입니다.	0.25

설정	설명	기본값
hive.vectorized.execution.enabled	벡터화된 쿼리 실행 모드를 켜는 옵션입니다.	TRUE
hive.vectorized.execution.reduce.enabled	쿼리 실행 축소 측의 벡터화 모드를 켜는 옵션입니다.	TRUE
javax.jdo.option.ConnectionDriverName	메타스토어의 드라이버 클래스 이름. JDBC	org.apache.derby.jdbc.EmbeddedDriver
javax.jdo.option.ConnectionPassword	메타스토어 데이터베이스와 관련된 비밀번호입니다.	NULL
javax.jdo.option.ConnectionURL	메타스토어의 JDBC 연결 문자열입니다. JDBC	jdbc:derby:;databaseName=metastore_db;create=true
javax.jdo.option.ConnectionUserName	메타스토어 데이터베이스와 관련된 사용자 이름.	NULL
mapreduce.input.fileinputformat.split.maxsize	입력 형식이 다음과 같을 때 분할 계산 시 분할할 수 있는 최대 크기입니다. org.apache.hadoop.hive.ql.io.CombineHiveInputFormat 값이 0이면 제한이 없음을 나타냅니다.	0
tez.am.dag.cleanup.on.completion	완료 시 셔플 데이터 정리를 켜는 옵션입니다. DAG	TRUE

설정	설명	기본값
tez.am.emr-serverless.launch.env.[*KEY*]	Tez AM 프로세스에서 *KEY* 환경 변수를 설정하는 옵션입니다. Tez AM의 경우 이 값이 값보다 우선합니다. hive.emr-serverless.launch.env.[*KEY*]	
tez.am.log.level	EMR서버리스가 Tez 앱 마스터에 전달하는 루트 로깅 레벨입니다.	INFO
tez.am.sleep.time.before.exit.millis	EMR서버리스는 AM 종료 ATS 요청 후 이 기간이 지나면 이벤트를 푸시해야 합니다.	0
tez.am.speculation.enabled	속도가 느린 작업의 예상 실행을 유발하는 옵션입니다. 이렇게 하면 컴퓨터가 불량하거나 느려서 일부 작업이 느리게 실행되는 경우 작업 대기 시간을 줄이는 데 도움이 될 수 있습니다. Amazon EMR 6.10.x 이하에서만 지원됩니다.	FALSE
tez.am.task.max.failed.attempts	작업이 실패하기 전에 특정 작업에 대해 실패할 수 있는 최대 시도 횟수입니다. 수동으로 종료한 시도는 이 수치에는 포함되지 않습니다.	3
tez.am.vertex.cleanup.height	종속 꼭짓점이 모두 완성되면 Tez AM이 버텍스 셔플 데이터를 삭제하는 거리입니다. 값이 0이면 이 기능이 꺼집니다. Amazon EMR 버전 6.8.0 이상에서는 이 기능을 지원합니다.	0

설정	설명	기본값
tez.client.asynchronous-stop	EMR서버리스가 Hive 드라이버를 종료하기 전에 ATS 이벤트를 푸시하도록 하는 옵션입니다.	FALSE
tez.grouping.max-size	그룹화된 분할의 크기 상한 (바이트). 이 제한은 지나치게 큰 분할을 방지합니다.	1073741824
tez.grouping.min-size	그룹화된 분할의 크기 하한 (바이트). 이 한도는 너무 많은 작은 분할을 방지합니다.	16777216
tez.runtime.io.sort.mb	Tez가 출력을 정렬할 때의 소프트 버퍼 크기가 정렬됩니다.	최적값은 Tez 태스크 메모리를 기반으로 계산됩니다.
tez.runtime.unordered.output.buffer.size-mb	Tez가 디스크에 직접 기록하지 않는 경우 사용할 버퍼 크기입니다.	최적값은 Tez 태스크 메모리를 기반으로 계산됩니다.
tez.shuffle-vertex-manager.max-src-fraction	EMR서버리스가 현재 버텍스에 대한 모든 작업을 스케줄링하기 전에 완료해야 하는 소스 작업의 비율 (연결의 경우). ScatterGather 현재 버텍스에서 스케줄링할 준비가 된 작업 수는 ~ 사이에서 선형적으로 증가합니다. min-fraction max-fraction 기본값은 기본값이거나 tez.shuffle-vertex-manager.min-src-fraction 둘 중 큰 값입니다.	0.75

설정	설명	기본값
`tez.shuffle-vertex-manager.min-src-fraction`	EMR서버리스가 현재 버텍스에 대한 작업을 스케줄링하기 전에 완료해야 하는 소스 작업의 비율 (연결의 경우). `ScatterGather`	0.25
`tez.task.emr-serverless.launch.env.[ KEY]`	Tez 작업 프로세스에서 *KEY* 환경 변수를 설정하는 옵션입니다. Tez 태스크의 경우 이 값이 값보다 우선합니다. `hive.emr-serverless.launch.env.[ KEY]`	
`tez.task.log.level`	EMR서버리스가 Tez 태스크에 전달하는 루트 로깅 레벨입니다.	`INFO`
`tez.yarn.ats.event.flush.timeout.millis`	AM이 종료되기 전에 이벤트가 플러시될 때까지 대기해야 하는 최대 시간입니다.	300000

하이브 작업 예제

다음 코드 예제는 를 사용하여 Hive 쿼리를 실행하는 방법을 보여줍니다. `StartJobRun` API

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql",
            "parameters": "--hiveconf hive.log.explain.output=false"
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
```

```
            "properties": {
                "hive.exec.scratchdir": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/
hive/scratch",
                "hive.metastore.warehouse.dir": "s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/warehouse",
                "hive.driver.cores": "2",
                "hive.driver.memory": "4g",
                "hive.tez.container.size": "4096",
                "hive.tez.cpu.vcores": "1"
            }
        }]
    }'
```

EMR서버리스 샘플 리포지토리에서 Hive 작업을 실행하는 방법에 대한 추가 예를 찾을 수 있습니다. GitHub

EMR서버리스 Job 레질리언스

EMR서버리스 릴리스 7.1.0 이상에는 작업 복구 지원이 포함되어 있으므로 사용자가 직접 입력하지 않아도 실패한 작업을 자동으로 재시도합니다. 작업 복원력의 또 다른 이점은 AZ에 문제가 발생할 경우 EMR 서버리스가 작업 실행을 다른 가용 영역 (AZ) 으로 이동한다는 것입니다.

작업에 대한 작업 복원력을 활성화하려면 작업에 대한 재시도 정책을 설정하십시오. 재시도 정책은 어느 시점에서든 작업이 실패할 경우 EMR 서버리스가 작업을 자동으로 다시 시작하도록 합니다. 재시도 정책은 일괄 작업과 스트리밍 작업 모두에 지원되므로 사용 사례에 따라 작업 복원력을 사용자 지정할 수 있습니다. 다음 표에서는 일괄 처리 및 스트리밍 작업 전반의 작업 복원력 동작과 차이를 비교합니다.

	배치 작업	스트리밍 작업
기본 동작	작업을 다시 실행하지 않습니다.	작업을 실행하는 동안 애플리케이션이 체크포인트를 생성하므로 항상 작업 실행을 재시도합니다.
재시도 지점	Batch 작업에는 체크포인트가 없으므로 EMR Serverless는 항상 작업을 처음부터 다시 실행합니다.	스트리밍 작업은 체크포인트를 지원하므로 런타임 상태와 진행 상황을 Amazon S3의 체크포인트 위치에 저장하도록 스트리밍 쿼리를 구성할 수 있습

	배치 작업	스트리밍 작업
		니다. EMR서버리스는 체크포인트에서 작업 실행을 재개합니다. 자세한 내용은 Apache Spark 설명서의 체크포인트를 사용한 오류 복구를 참조하십시오.
최대 재시도 횟수	최대 10회의 재시도를 허용합니다.	스트리밍 작업에는 스래시 방지 제어 기능이 내장되어 있어 1시간 후에도 작업이 계속 실패하면 애플리케이션에서 작업 재시도를 중지합니다. 1시간 내 기본 재시도 횟수는 5회입니다. 이 재시도 횟수를 1~10회 사이로 구성할 수 있습니다. 최대 시도 횟수는 사용자 지정할 수 없습니다. 값이 1이면 재시도가 없음을 나타냅니다.

EMR서버리스는 작업 재실행을 시도할 때 시도 횟수와 함께 작업을 인덱싱하므로 시도 전반에 걸쳐 작업의 수명 주기를 추적할 수 있습니다.

EMR서버리스 API 작업이나 다음을 사용할 수 있습니다. AWS CLI 작업 복원력을 변경하거나 작업 복원력과 관련된 정보를 볼 수 있습니다. 자세한 내용은 EMR서버리스 API 가이드를 참조하십시오.

기본적으로 EMR 서버리스는 일괄 작업을 다시 실행하지 않습니다. 일괄 작업에 대한 재시도를 활성화하려면 일괄 작업 실행을 시작할 때 `maxAttempts` 매개변수를 구성하십시오. 이 `maxAttempts` 매개변수는 배치 작업에만 적용됩니다. 기본값은 1이며, 이는 작업을 다시 실행하지 않음을 의미합니다. 허용되는 값은 1~10 (포함) 입니다.

다음 예제는 작업 실행 시작 시 최대 시도 횟수를 10회까지 지정하는 방법을 보여줍니다.

```
aws emr-serverless start-job-run
 --application-id <APPLICATION_ID> \
 --execution-role-arn <JOB_EXECUTION_ROLE> \
 --mode 'BATCH' \
 --retry-policy '{
```

```
    "maxAttempts": 10
 }' \
 --job-driver '{
    "sparkSubmit": {
         "entryPoint": "/usr/lib/spark/examples/jars/spark-examples-does-not-
exist.jar",
         "entryPointArguments": ["1"],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
     }
}'
```

EMR서버리스는 실패할 경우 스트리밍 작업을 무기한 재시도합니다. 반복되는 복구할 수 없는 실패로 인한 스래싱을 방지하려면 를 사용하여 스트리밍 작업 재시도에 maxFailedAttemptsPerHour 대한 스래시 방지 제어를 구성하십시오. 이 매개 변수를 사용하면 서버리스가 재시도를 중지하기 1시간 전까지 허용되는 최대 실패 시도 횟수를 지정할 수 있습니다. EMR 기본값은 5입니다. 허용되는 값은 1~10 (포함) 입니다.

```
aws emr-serverless start-job-run
 --application-id <APPPLICATION_ID> \
 --execution-role-arn <JOB_EXECUTION_ROLE> \
 --mode 'STREAMING' \
 --retry-policy '{
    "maxFailedAttemptsPerHour": 7
 }' \
 --job-driver '{
    "sparkSubmit": {
         "entryPoint": "/usr/lib/spark/examples/jars/spark-examples-does-not-
exist.jar",
         "entryPointArguments": ["1"],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
     }
}'
```

다른 작업 실행 API 작업을 사용하여 작업에 대한 정보를 얻을 수도 있습니다. 예를 들어 attempt 매개 변수를 작업과 함께 사용하여 특정 GetJobRun 작업 시도에 대한 세부 정보를 얻을 수 있습니다. attempt매개변수를 포함하지 않으면 작업은 최근 시도에 대한 정보를 반환합니다.

```
aws emr-serverless get-job-run \
    --job-run-id job-run-id \
    --application-id application-id \
    --attempt 1
```

ListJobRunAttempts작업은 작업 실행과 관련된 모든 시도에 대한 정보를 반환합니다.

```
aws emr-serverless list-job-run-attempts \
  --application-id application-id \
  --job-run-id job-run-id
```

GetDashboardForJobRun작업을 수행하면 작업 실행을 UIs 위해 응용 프로그램에 액세스하는 데 사용할 수 있는 데이터가 생성되고 반환됩니다. URL attempt매개 변수를 사용하면 특정 시도에 URL 대한 정보를 얻을 수 있습니다. attempt매개변수를 포함하지 않으면 작업은 최근 시도에 대한 정보를 반환합니다.

```
aws emr-serverless get-dashboard-for-job-run \
    --application-id application-id \
    --job-run-id job-run-id \
    --attempt 1
```

재시도 정책으로 작업 모니터링

또한 Job Resiliency 지원에는 EMR서버리스 작업 실행 재시도라는 새 이벤트가 추가되었습니다. EMR 서버리스는 작업을 재시도할 때마다 이 이벤트를 게시합니다. 이 알림을 사용하여 작업 재시도를 추적할 수 있습니다. 이벤트에 대한 자세한 내용은 Amazon EventBridge 이벤트를 참조하십시오.

재시도 정책을 사용한 로깅

EMR서버리스가 작업을 재시도할 때마다 해당 시도는 자체 로그 세트를 생성합니다. EMR서버리스가 이러한 로그를 덮어쓰지 CloudWatch 않고 Amazon S3와 Amazon에 성공적으로 전송할 수 있도록 하기 위해 EMR 서버리스는 S3 로그 경로 및 CloudWatch 로그 스트림 이름 형식에 접두사를 추가하여 작업 시도 횟수를 포함합니다.

다음은 형식이 어떻게 보이는지에 대한 예시입니다.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'.
```

이 형식을 사용하면 EMR 서버리스가 각 작업 시도에 대한 모든 로그를 Amazon S3 및 및 내의 지정된 위치에 게시할 수 있습니다. CloudWatch 자세한 내용은 로그 저장을 참조하십시오.

> **Note**
>
> EMR서버리스는 모든 스트리밍 작업 및 재시도가 활성화된 모든 일괄 작업에 이 접두사 형식만 사용합니다.

메타스토어 구성

Hive 메타스토어는 스키마, 파티션 이름, 데이터 유형 등 테이블에 대한 구조적 정보를 저장하는 중앙 위치입니다. EMR서버리스를 사용하면 작업에 액세스할 수 있는 메타스토어에 이 테이블 메타데이터를 유지할 수 있습니다.

Hive 메타스토어에는 두 가지 옵션이 있습니다.

- The AWS Glue Data 카탈로그
- 외부 아파치 하이브 메타스토어

사용 AWS 메타스토어로서의 Glue 데이터 카탈로그

다음을 사용하도록 Spark 및 Hive 작업을 구성할 수 있습니다. AWS Glue 데이터 카탈로그를 메타스토어로 사용합니다. 영구 메타스토어가 필요하거나 다른 애플리케이션, 서비스 또는 다른 애플리케이션에서 공유하는 메타스토어가 필요한 경우 이 구성을 사용하는 것이 좋습니다. AWS 계정. 데이터 카탈로그에 대한 자세한 내용은 데이터 카탈로그 채우기를 참조하십시오. AWS Glue 데이터 카탈로그. 자세한 내용은 AWS Glue 가격 책정, 참조 AWS Glue 가격 책정.

다음을 사용하도록 EMR 서버리스 작업을 구성할 수 있습니다. AWS Glue 데이터 카탈로그를 둘 중 하나에 AWS 계정 사용 중인 애플리케이션으로 또는 다른 애플리케이션으로 AWS 계정.

다음을 구성하십시오. AWS Glue Data 카탈로그

데이터 카탈로그를 구성하려면 사용할 EMR 서버리스 애플리케이션 유형을 선택합니다.

Spark

EMRStudio를 사용하여 EMR 서버리스 Spark 애플리케이션으로 작업을 실행하는 경우 AWS Glue 데이터 카탈로그는 기본 메타스토어입니다.

또는 를 사용하는 경우 SDKs AWS CLI작업 실행 `sparkSubmit` 매개 변수에서 `spark.hadoop.hive.metastore.client.factory.class` 구성을 로

com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory 설정할 수 있습니다. 다음 예는 를 사용하여 데이터 카탈로그를 구성하는 방법을 보여줍니다. AWS CLI.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/code/pyspark/extreme_weather.py",
            "sparkSubmitParameters": "--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGl
 --conf spark.driver.cores=1 --conf spark.driver.memory=3g --conf
 spark.executor.cores=4 --conf spark.executor.memory=3g"
        }
    }'
```

또는 Spark SparkSession 코드에서 새 항목을 만들 때 이 구성을 설정할 수 있습니다.

```
from pyspark.sql import SparkSession

spark = (
    SparkSession.builder.appName("SparkSQL")
    .config(
        "spark.hadoop.hive.metastore.client.factory.class",
        "com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
    )
    .enableHiveSupport()
    .getOrCreate()
)

# we can query tables with SparkSQL
spark.sql("SHOW TABLES").show()

# we can also them with native Spark
print(spark.catalog.listTables())
```

Hive

EMR서버리스 Hive 애플리케이션의 경우 데이터 카탈로그가 기본 메타스토어입니다. 즉, EMR 서버리스 Hive 애플리케이션에서 작업을 실행하면 Hive는 메타스토어 정보를 동일한 데이터 카탈로

그의 데이터 카탈로그에 기록합니다. AWS 계정 애플리케이션과 동일합니다. 데이터 카탈로그를 메타스토어로 사용하기 위한 가상 사설 클라우드 (VPC) 는 필요하지 않습니다.

Hive 메타스토어 테이블에 액세스하려면 필요한 항목을 추가하세요. AWS IAM권한 설정에 설명된 Glue 정책 AWS Glue.

EMR서버리스에 대한 교차 계정 액세스를 구성하고 AWS Glue Data 카탈로그

EMR서버리스에 대한 교차 계정 액세스를 설정하려면 먼저 다음 계정에 로그인해야 합니다. AWS 계정:

- `AccountA`— An AWS 계정 EMR서버리스 애플리케이션을 만든 곳.
- `AccountB`— An AWS 계정 여기에는 a가 들어 있습니다. AWS EMR서버리스 작업 실행에서 액세스하려는 Glue 데이터 카탈로그입니다.

1. 의 관리자 또는 기타 인증된 ID가 의 데이터 카탈로그에 리소스 정책을 `AccountB` 연결하는지 확인하십시오. `AccountB` 이 정책은 카탈로그의 리소스에 대한 작업을 수행할 수 있는 `AccountA` 특정 교차 계정 권한을 부여합니다`AccountB`.

```
{
  "Version" : "2012-10-17",
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::accountA:role/job-runtime-role-A"
        ]},
    "Action" : [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
```

```
        "glue:GetUserDefinedFunctions"
    ],
    "Resource": ["arn:aws:glue:region:AccountB:catalog"]
  } ]
}
```

2. 역할이 데이터 카탈로그 리소스에 액세스할 수 AccountA 있도록 EMR 서버리스 작업 런타임 역할에 IAM 정책을 추가합니다. AccountB

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase",
        "glue:GetDataBases",
        "glue:CreateTable",
        "glue:GetTable",
        "glue:UpdateTable",
        "glue:DeleteTable",
        "glue:GetTables",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:CreatePartition",
        "glue:BatchCreatePartition",
        "glue:GetUserDefinedFunctions"
      ],
      "Resource": ["arn:aws:glue:region:AccountB:catalog"]
    }
  ]
}
```

3. 작업 실행을 시작합니다. 이 단계는 AccountA 의 EMR 서버리스 애플리케이션 유형에 따라 약간 다릅니다.

   Spark

   다음 예와 같이 hive-site 분류에 spark.hadoop.hive.metastore.glue.catalogid 속성을 설정합니다. Replace *##-####-ID* 데이터 카탈로그의 ID 입력 AccountB

```
aws emr-serverless start-job-run \
```

```
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
    "sparkSubmit": {
        "query": "s3://DOC-EXAMPLE-BUCKET/hive/scripts/create_table.sql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/
hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/
hive/warehouse"
    }
  }' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "spark.hadoop.hive.metastore.glue.catalogid": "AccountB-catalog-id"
        }
    }]
}'
```

Hive

다음 예와 같이 `hive-site` 분류에서 `hive.metastore.glue.catalogid` 속성을 설정합니다. Replace *##-####-ID* 데이터 카탈로그의 ID 입력 AccountB

```
aws emr-serverless start-job-run \
--application-id "application-id" \
--execution-role-arn "job-role-arn" \
--job-driver '{
    "hive": {
    "query": "s3://DOC-EXAMPLE-BUCKET/hive/scripts/create_table.sql",
    "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/hive/
scratch --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/hive/
warehouse"
    }
}' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.metastore.glue.catalogid": "AccountB-catalog-id"
        }
    }]
```

```
}'
```

사용 시 고려 사항 AWS Glue Data 카탈로그

Hive 스크립트에 보조 JARs ADD JAR 변수를 추가할 수 있습니다. 추가 고려 사항은 사용 시 고려 사항을 참조하십시오. AWS Glue 데이터 카탈로그.

외부 Hive 메타스토어 사용

Amazon Aurora 또는 Amazon for My와 같은 외부 하이브 메타스토어에 연결하도록 EMR 서버리스 스파크 및 하이브 작업을 구성할 수 있습니다. RDS SQL 이 섹션에서는 Amazon RDS Hive 메타스토어를 설정하고, 외부 메타스토어를 사용하도록 EMR 서버리스 작업을 구성하고VPC, 구성하는 방법을 설명합니다.

외부 Hive 메타스토어 생성

1. 만들기의 지침에 따라 프라이빗 서브넷이 있는 Amazon Virtual Private Cloud (AmazonVPC) 를 생성하십시오. VPC
2. 새 Amazon VPC 및 프라이빗 서브넷으로 EMR 서버리스 애플리케이션을 생성하십시오. 로 EMR 서버리스 애플리케이션을 구성하면 먼저 지정한 각 서브넷에 대해 Elastic Network 인터페이스가 프로비저닝됩니다. VPC 그런 다음 지정된 보안 그룹을 해당 네트워크 인터페이스에 연결합니다. 이렇게 하면 애플리케이션에 액세스 제어가 제공됩니다. 설정 방법에 대한 자세한 내용은 VPC 을 참조하십시오액세스 구성 VPC.
3. Amazon의 프라이빗 서브넷에 My SQL 또는 Aurora Postgre SQL 데이터베이스를 생성하십시오. VPC Amazon RDS 데이터베이스를 만드는 방법에 대한 자세한 내용은 Amazon RDS DB 인스턴스 생성을 참조하십시오.
4. Amazon RDS DB 인스턴스 수정에 나와 있는 단계에 따라 EMR 서버리스 보안 그룹으로부터의 JDBC 연결을 허용하도록 My SQL 또는 Aurora 데이터베이스의 보안 그룹을 수정하십시오. 서버리스 보안 그룹 중 하나에서 RDS 보안 그룹에 인바운드 트래픽에 대한 규칙을 추가합니다. EMR

유형	프로토콜	포트 범위	소스
모두 TCP	TCP	3306	`emr-serverless-security-group`

스파크 옵션 구성

사용 JDBC

Amazon for My RDS 또는 Amazon SQL Aurora My 인스턴스 기반의 Hive 메타스토어에 연결하도록 EMR 서버리스 Spark 애플리케이션을 구성하려면 연결을 사용하십시오. SQL JDBC w를 작업 `--jars` 실행의 `mariadb-connector-java.jar` 파라미터로 전달하십시오. `spark-submit`

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-BUCKET/mariadb-connector-
java.jar
            --conf
 spark.hadoop.javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
            --conf spark.hadoop.javax.jdo.option.ConnectionUserName=<connection-user-
name>
            --conf spark.hadoop.javax.jdo.option.ConnectionPassword=<connection-
password>
            --conf spark.hadoop.javax.jdo.option.ConnectionURL=<JDBC-Connection-
string>
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
    }
}'
```

다음 코드 예제는 Amazon의 Hive 메타스토어와 상호 작용하는 Spark 진입점 스크립트입니다. RDS

```
from os.path import expanduser, join, abspath
from pyspark.sql import SparkSession
```

```
from pyspark.sql import Row
# warehouse_location points to the default location for managed databases and tables
warehouse_location = abspath('spark-warehouse')
spark = SparkSession \
    .builder \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .enableHiveSupport() \
    .getOrCreate()
spark.sql("SHOW DATABASES").show()
spark.sql("CREATE EXTERNAL TABLE `sampledb`.`sparknyctaxi`(`dispatching_base_num`
 string, `pickup_datetime` string, `dropoff_datetime` string, `pulocationid` bigint,
 `dolocationid` bigint, `sr_flag` bigint) STORED AS PARQUET LOCATION 's3://<s3 prefix>/
nyctaxi_parquet/'")
spark.sql("SELECT count(*) FROM sampledb.sparknyctaxi").show()
spark.stop()
```

중고품 서비스 사용

Amazon for My RDS 또는 Amazon SQL Aurora My 인스턴스를 기반으로 하는 하이브 메타스토어에 연결하도록 EMR 서버리스 하이브 애플리케이션을 구성할 수 있습니다. SQL 이렇게 하려면 기존 Amazon EMR 클러스터의 마스터 노드에서 스리프트 서버를 실행하십시오. 이 옵션은 EMR 서버리스 작업 구성을 단순화하는 데 사용하려는 중고품 서버가 있는 Amazon EMR 클러스터가 이미 있는 경우에 적합합니다.

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/thriftscript.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-BUCKET/mariadb-connector-
java.jar
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
```

```
    }
}'
```

다음 코드 예제는 Thrift 프로토콜을 사용하여 Hive 메타스토어에 연결하는 진입점 스크립트 (`thriftscript.py`) 입니다. 단, 외부 Hive 메타스토어에서 읽을 수 있도록 `hive.metastore.uris` 속성을 설정해야 합니다.

```
from os.path import expanduser, join, abspath
from pyspark.sql import SparkSession
from pyspark.sql import Row
# warehouse_location points to the default location for managed databases and tables
warehouse_location = abspath('spark-warehouse')
spark = SparkSession \
    .builder \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .config("hive.metastore.uris","thrift://thrift-server-host:thift-server-port") \
    .enableHiveSupport() \
    .getOrCreate()
spark.sql("SHOW DATABASES").show()
spark.sql("CREATE EXTERNAL TABLE sampledb.`sparknyctaxi`( `dispatching_base_num`
 string, `pickup_datetime` string, `dropoff_datetime` string, `pulocationid` bigint,
 `dolocationid` bigint, `sr_flag` bigint) STORED AS PARQUET LOCATION 's3://<s3 prefix>/
nyctaxi_parquet/'")
spark.sql("SELECT * FROM sampledb.sparknyctaxi").show()
spark.stop()
```

Hive 옵션을 구성하십시오.

사용 JDBC

Amazon RDS My SQL 또는 Amazon Aurora 인스턴스에서 외부 Hive 데이터베이스 위치를 지정하려는 경우 기본 메타스토어 구성을 재정의할 수 있습니다.

> **Note**
>
> Hive에서는 메타스토어 테이블에 여러 개의 쓰기를 동시에 수행할 수 있습니다. 두 작업 간에 메타스토어 정보를 공유하는 경우 동일한 메타스토어 테이블의 다른 파티션에 쓰는 경우를 제외하고는 동일한 메타스토어 테이블에 동시에 쓰지 않도록 하세요.

`hive-site`분류에서 다음 구성을 설정하여 외부 Hive 메타스토어를 활성화하십시오.

```
{
    "classification": "hive-site",
    "properties": {
        "hive.metastore.client.factory.class":
 "org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClientFactory",
        "javax.jdo.option.ConnectionDriverName": "org.mariadb.jdbc.Driver",
        "javax.jdo.option.ConnectionURL": "jdbc:mysql://db-host:db-port/db-name",
        "javax.jdo.option.ConnectionUserName": "username",
        "javax.jdo.option.ConnectionPassword": "password"
    }
}
```

중고품 서버 사용

Amazon for My RDS 또는 Amazon SQL Aurora M을 기반으로 하는 하이브 메타스토어에 연결하도록 EMR 서버리스 하이브 애플리케이션을 구성할 수 있습니다. ySQLinstance 이렇게 하려면 기존 Amazon EMR 클러스터의 기본 노드에서 중고품 서버를 실행하십시오. 이 옵션은 이미 쓰리프트 서버를 실행하는 Amazon EMR 클러스터가 있고 EMR 서버리스 작업 구성을 사용하려는 경우에 적합합니다.

EMR서버리스가 원격 스리프트 메타스토어에 액세스할 수 있도록 `hive-site` 분류에 다음 구성을 설정하십시오. 단, 외부 Hive 메타스토어에서 읽을 수 있도록 `hive.metastore.uris` 속성을 설정해야 합니다.

```
{
    "classification": "hive-site",
    "properties": {
        "hive.metastore.client.factory.class":
 "org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClientFactory",
        "hive.metastore.uris": "thrift://thrift-server-host:thirft-server-port"
    }
}
```

외부 메타스토어 사용 시 고려 사항

- JDBCMariaDB와 호환되는 데이터베이스를 메타스토어로 구성할 수 있습니다. 이러한 데이터베이스의 예로는 MariaDB, SQL My 및 Amazon RDS Aurora용 데이터베이스가 있습니다.
- 메타스토어는 자동 초기화되지 않습니다. 메타스토어가 Hive 버전의 스키마로 초기화되지 않은 경우 Hive 스키마 도구를 사용하세요.

- EMR서버리스는 Kerberos 인증을 지원하지 않습니다. Kerberos 인증이 포함된 중고품 메타스토어 서버는 서버리스 스파크 또는 Hive 작업과 함께 사용할 수 없습니다. EMR

다른 곳의 S3 데이터에 액세스 AWS EMR서버리스 계정

Amazon EMR 서버리스 작업은 하나에서 실행할 수 있습니다. AWS 계정을 지정하고 다른 버킷에 속하는 Amazon S3 버킷의 데이터에 액세스하도록 구성합니다. AWS 계정. 이 페이지에서는 EMR 서버리스에서 S3에 대한 계정 간 액세스를 구성하는 방법을 설명합니다.

EMR서버리스에서 실행되는 작업은 S3 버킷 정책 또는 위임된 역할을 사용하여 다른 곳에서 Amazon S3의 데이터에 액세스할 수 있습니다. AWS 계정.

사전 조건

Amazon EMR Serverless에 대한 교차 계정 액세스를 설정하려면 두 서버에 로그인한 상태에서 작업을 완료해야 합니다. AWS 계정:

- **`AccountA`**— 이것은 AWS Amazon EMR 서버리스 애플리케이션을 생성한 계정 교차 계정 액세스를 설정하기 전에 이 계정에 다음을 준비해야 합니다.
  - 작업을 실행하려는 Amazon EMR 서버리스 애플리케이션.
  - 애플리케이션에서 작업을 실행하는 데 필요한 권한이 있는 작업 실행 역할. 자세한 내용은 Amazon EMR 서버리스의 Job 런타임 역할 단원을 참조하십시오.
- **`AccountB`**— 이것은 AWS Amazon EMR 서버리스 작업에서 액세스하려는 S3 버킷이 들어 있는 계정입니다.

S3 버킷 정책을 사용하여 계정 간 S3 데이터에 액세스하십시오.

에서 S3 버킷에 액세스하려면 account B from account A다음 정책을 의 S3 버킷에 연결합니다.account B.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions 1",
         "Effect": "Allow",
         "Principal": {
```

```
            "AWS": "arn:aws:iam::AccountA:root"
         },
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::bucket_name_in_AccountB"
         ]
      },
      {
         "Sid": "Example permissions 2",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::AccountA:root"
         },
         "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:DeleteObject"
         ],
         "Resource": [
            "arn:aws:s3:::bucket_name_in_AccountB/*"
         ]
      }
   ]
}
```

S3 버킷 정책을 사용한 S3 교차 계정 액세스에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 예제 2: 버킷 소유자에게 계정 간 버킷 권한 부여를 참조하십시오.

위임된 역할을 사용하여 계정 간 S3 데이터에 액세스하십시오.

Amazon EMR Serverless에 대한 교차 계정 액세스를 설정하는 또 다른 방법은 다음 `AssumeRole` 작업을 수행하는 것입니다. AWS Security Token Service (AWS STS). AWS STS 사용자를 위해 권한이 제한된 임시 자격 증명을 요청할 수 있는 글로벌 웹 서비스입니다. 생성한 임시 보안 자격 증명을 사용하여 EMR 서버리스 및 Amazon S3를 API `AssumeRole` 호출할 수 있습니다.

다음 단계는 위임된 역할을 사용하여 서버리스에서 계정 간 S3 데이터에 액세스하는 방법을 보여줍니다. EMR

1. Amazon S3 버킷을 생성합니다.*cross-account-bucket*, 에서AccountB. 자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 버킷 생성을 참조하세요. DynamoDB에 대한 크

로스 계정 액세스를 원하는 경우 AccountB에서 DynamoDB 테이블을 생성할 수도 있습니다. 자세한 내용은 Amazon DynamoDB 개발자 안내서의 DynamoDB 테이블 생성을 참조하십시오.

2. 액세스할 수 있는 Cross-Account-Role-B IAM AccountB 역할을 생성하십시오.*cross-account-bucket*.

   a. 에 로그인하십시오. AWS Management Console 에서 IAM 콘솔을 엽니다 https://console.aws.amazon.com/iam/.

   b. 역할을 선택하고 새 역할(Cross-Account-Role-B)을 생성합니다. IAM역할을 생성하는 방법에 대한 자세한 내용은 IAM 사용 설명서에서 IAM역할 생성을 참조하십시오.

   c. 액세스 Cross-Account-Role-B 권한을 지정하는 IAM 정책을 생성하십시오.*cross-account-bucket* 다음 정책 설명에서 볼 수 있듯이 S3 버킷. 그런 다음 IAM 정책을 에 Cross-Account-Role-B 연결합니다. 자세한 내용은 IAM사용 설명서의 IAM정책 생성을 참조하십시오.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::cross-account-bucket",
                "arn:aws:s3:::cross-account-bucket/*"
            ]
        }
    ]
}
```

DynamoDB 액세스가 필요한 경우 교차 계정 DynamoDB 테이블에 액세스할 권한을 지정하는 정책을 IAM 생성하십시오. 그런 다음 정책을 에 연결합니다. IAM Cross-Account-Role-B 자세한 내용은 사용 설명서의 Amazon DynamoDB: 특정 테이블에 대한 액세스 허용을 참조하십시오. IAM

다음은 CrossAccountTable DynamoDB 테이블에 대한 액세스를 허용하는 정책입니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:*",
```

```
            "Resource": "arn:aws:dynamodb:MyRegion:AccountB:table/CrossAccountTable"
        }
    ]
}
```

3. Cross-Account-Role-B 역할에 대한 신뢰 관계를 편집합니다.

   a. 역할에 대한 신뢰 관계를 구성하려면 2단계에서 Cross-Account-Role-B 생성한 역할에 대해 IAM 콘솔에서 신뢰 관계 탭을 선택합니다.

   b. 신뢰 관계 편집을 선택합니다.

   c. 다음 정책 문서를 추가합니다. 이렇게 하면 Job-Execution-Role-A Cross-Account-Role-B 역할을 AccountA 맡을 수 있습니다.

```
{
"Version": "2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::AccountA:role/Job-Execution-Role-A"
        },
        "Action": "sts:AssumeRole"
    }
  ]
}
```

4. Job-Execution-Role-AAccountA그랜트 AWS STS AssumeRole맡을 수 있는 권한Cross-Account-Role-B.

   a. IAM콘솔에서 AWS 계정AccountA, 선택Job-Execution-Role-A.

   b. 다음 정책 명령을 Job-Execution-Role-A에 추가하여 Cross-Account-Role-B 역할에서 AssumeRole 작업을 허용합니다.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
        }
    ]
```

```
}
```

가정된 역할 예제

하나의 위임된 역할을 사용하여 계정의 모든 S3 리소스에 액세스할 수 있으며, Amazon EMR 6.11 이상에서는 서로 다른 계정 간 S3 버킷에 액세스할 때 여러 IAM 역할을 맡도록 구성할 수 있습니다.

주제

- 하나의 위임된 역할로 S3 리소스에 액세스할 수 있습니다.
- 여러 위임된 역할을 통해 S3 리소스에 액세스할 수 있습니다.

하나의 위임된 역할로 S3 리소스에 액세스할 수 있습니다.

> **Note**
>
> 단일 수임된 역할을 사용하도록 작업을 구성하면 스크립트를 포함하여 작업 전체의 모든 S3 리소스가 해당 역할을 사용합니다. `entryPoint`

단일 위임된 역할을 사용하여 계정 B의 모든 S3 리소스에 액세스하려면 다음 구성을 지정하십시오.

1. EMRFS구성을 `fs.s3.customAWSCredentialsProvider` 로 `spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAW` 지정하십시오.
2. Spark의 경우 `spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` 및 `spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` 를 사용하여 드라이버 및 실행기의 환경 변수를 지정합니다.
3. Hive의 경우 `hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARNtez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN`, 및 `tez.task.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN` 를 사용하여 Hive 드라이버, Tez 애플리케이션 마스터 및 Tez 작업 컨테이너에서 환경 변수를 지정합니다.

다음 예제는 위임된 역할을 사용하여 계정 간 액세스로 EMR 서버리스 작업 실행을 시작하는 방법을 보여줍니다.

Spark

다음 예제는 위임된 역할을 사용하여 S3에 대한 교차 계정 액세스가 가능한 EMR 서버리스 Spark 작업 실행을 시작하는 방법을 보여줍니다.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "entrypoint_location",
            "entryPointArguments": [":argument_1:", ":argument_2:"],
            "sparkSubmitParameters": "--conf spark.executor.cores=4 --conf
 spark.executor.memory=20g --conf spark.driver.cores=4 --conf spark.driver.memory=8g
 --conf spark.executor.instances=1"
        }
    }' \
     --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "spark-defaults",
            "properties": {
                "spark.hadoop.fs.s3.customAWSCredentialsProvider":
 "spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentials
                "spark.emr-serverless.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
            }
        }]
    }'
```

Hive

다음 예제는 위임된 역할을 사용하여 S3에 대한 교차 계정 액세스가 가능한 EMR 서버리스 Hive 작업 실행을 시작하는 방법을 보여줍니다.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "query_location",
            "parameters": "hive_parameters"
```

```
        }
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.AssumeRoleAWSCredentialsProvider",
                "hive.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "tez.am.emr-serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B",
                "tez.task.emr-
serverless.launch.env.ASSUME_ROLE_CREDENTIALS_ROLE_ARN":
 "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
            }
        }]
    }'
```

여러 위임된 역할을 통해 S3 리소스에 액세스할 수 있습니다.

EMR서버리스 릴리스 6.11.0 이상에서는 여러 계정 간 버킷에 액세스할 때 여러 IAM 역할을 맡도록 구성할 수 있습니다. 계정 B에서 위임된 역할이 서로 다른 S3 리소스에 액세스하려면 작업 실행 시 다음 구성을 사용하십시오.

1. EMRFS구성을 `fs.s3.customAWSCredentialsProvider` 로 `com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCredenti` 지정하십시오.
2. S3 버킷 이름에서 수임할 계정 B의 IAM 역할로의 매핑을 정의하는 EMRFS 구성을 `fs.s3.bucketLevelAssumeRoleMapping` 지정합니다. 값은 의 형식이어야 `bucket1->role1;bucket2->role2` 합니다.

예를 들어 버킷에 액세스하는 `arn:aws:iam::AccountB:role/Cross-Account-Role-B-1` 데를 사용하고 `bucket1` `arn:aws:iam::AccountB:role/Cross-Account-Role-B-2` 버킷에 액세스하는 데 사용할 수 `bucket2` 있습니다. 다음 예는 여러 수임된 역할을 통해 계정 간 액세스로 EMR 서버리스 작업 실행을 시작하는 방법을 보여줍니다.

Spark

다음 예제는 여러 수임된 역할을 사용하여 EMR 서버리스 Spark 작업 실행을 만드는 방법을 보여줍니다.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "entrypoint_location",
            "entryPointArguments": [":argument_1:", ":argument_2:"],
            "sparkSubmitParameters": "--conf spark.executor.cores=4 --conf
 spark.executor.memory=20g --conf spark.driver.cores=4 --conf spark.driver.memory=8g
 --conf spark.executor.instances=1"
        }
    }' \
     --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "spark-defaults",
            "properties": {
                "spark.hadoop.fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.BucketLevelAssumeRoleCredentialsProvider"
                "spark.hadoop.fs.s3.bucketLevelAssumeRoleMapping":
 "bucket1->arn:aws:iam::AccountB:role/Cross-Account-Role-B-1;bucket2-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-2"
            }
        }]
    }'
```

Hive

다음 예는 여러 수임된 역할을 사용하여 EMR 서버리스 Hive 작업 실행을 생성하는 방법을 보여줍니다.

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "hive": {
            "query": "query_location",
            "parameters": "hive_parameters"
        }
```

```
    }' \
    --configuration-overrides '{
        "applicationConfiguration": [{
            "classification": "hive-site",
            "properties": {
                "fs.s3.customAWSCredentialsProvider":
 "com.amazonaws.emr.serverless.credentialsprovider.AssumeRoleAWSCredentialsProvider",
                "fs.s3.bucketLevelAssumeRoleMapping": "bucket1-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-1;bucket2-
>arn:aws:iam::AccountB:role/Cross-Account-Role-B-2"
            }
        }]
    }'
```

EMR서버리스 오류 문제 해결

다음 정보를 사용하면 Amazon EMR Serverless를 사용할 때 발생할 수 있는 일반적인 문제를 진단하고 해결하는 데 도움이 됩니다.

주제

- 오류: 최대 허용 용량 한도를 초과했습니다.
- 오류: 구성된 최대 용량을 초과했습니다. 나중에 다시 시도하세요.
- 오류: S3 액세스가 거부되었습니다. 필요한 S3 리소스에서 작업 런타임 역할의 S3 액세스 권한을 확인하십시오.
- 오류 ModuleNotFoundError: 이름이 지정된 모듈이 없습니다<module>. EMR서버리스에서 Python 라이브러리를 사용하는 방법에 대한 사용 설명서를 참조하십시오.
- 오류: 실행 역할이 <role name>없거나 필요한 신뢰 관계로 설정되지 않았으므로 실행 역할을 맡을 수 없습니다.

오류: 최대 허용 용량 한도를 초과했습니다.

이 오류는 애플리케이션이 구성된 최대 용량 제한을 초과했기 때문에 EMR 서버리스에서 작업을 제출할 수 없었음을 나타냅니다. 애플리케이션의 최대 용량 제한을 늘리십시오.

오류: 구성된 최대 용량을 초과했습니다. 나중에 다시 시도하세요.

이 오류는 애플리케이션이 구성된 최대 용량 제한을 초과했기 때문에 EMR 서버리스에서 새 작업을 시작할 수 없었음을 나타냅니다. 애플리케이션의 최대 용량 제한을 늘리십시오.

오류: S3 액세스가 거부되었습니다. 필요한 S3 리소스에서 작업 런타임 역할의 S3 액세스 권한을 확인하십시오.

이 오류는 작업에 S3 리소스에 대한 액세스 권한이 없음을 나타냅니다. 작업 런타임 역할에 작업에 필요한 S3 리소스에 액세스할 수 있는 권한이 있는지 확인하십시오. 런타임 역할에 대한 자세한 내용은 을 참조하십시오Amazon EMR 서버리스의 Job 런타임 역할.

오류 ModuleNotFoundError: 이름이 지정된 모듈이 없습니다<module>. EMR서버리스에서 Python 라이브러리를 사용하는 방법에 대한 사용 설명서를 참조하십시오.

이 오류는 Python 모듈을 Spark 작업에 사용할 수 없음을 나타냅니다. 종속 Python 라이브러리를 작업에 사용할 수 있는지 확인합니다. Python 라이브러리를 패키징하는 방법에 대한 자세한 내용은 을 참조하십시오EMR서버리스에서 Python 라이브러리 사용하기.

오류: 실행 역할이 <role name>없거나 필요한 신뢰 관계로 설정되지 않았으므로 실행 역할을 맡을 수 없습니다.

이 오류는 작업에 지정한 작업 런타임 역할이 존재하지 않거나 역할에 EMR 서버리스 권한에 대한 신뢰 관계가 없음을 나타냅니다. IAM역할이 존재하는지 확인하고 역할의 신뢰 정책을 제대로 설정했는지 확인하려면 의 Amazon EMR 서버리스의 Job 런타임 역할 지침을 참조하십시오.

Studio를 통해 EMR 서버리스로 대화형 워크로드 실행 EMR

개요

대화형 애플리케이션은 대화형 기능이 EMR 활성화된 서버리스 애플리케이션입니다. Amazon EMR 서버리스 대화형 애플리케이션을 사용하면 Amazon Studio에서 관리되는 Jupyter 노트북으로 대화형 워크로드를 실행할 수 있습니다. EMR 이를 통해 데이터 엔지니어, 데이터 과학자, 데이터 분석가는 EMR Studio를 사용하여 Amazon S3 및 Amazon DynamoDB와 같은 데이터 스토어의 데이터 세트를 사용하여 대화형 분석을 실행할 수 있습니다.

EMR서버리스의 대화형 애플리케이션 사용 사례는 다음과 같습니다.

- 데이터 엔지니어는 EMR Studio의 IDE 경험을 사용하여 ETL 스크립트를 만듭니다. 스크립트는 온프레미스에서 데이터를 수집하고, 분석을 위해 데이터를 변환하고, Amazon S3에 데이터를 저장합니다.
- 데이터 사이언티스트는 노트북을 사용하여 데이터 세트를 탐색하고 기계 학습 (ML) 모델을 학습시켜 데이터 세트의 이상을 감지합니다.
- 데이터 분석가는 데이터세트를 탐색하고 비즈니스 대시보드와 같은 애플리케이션을 업데이트하기 위한 일일 보고서를 생성하는 스크립트를 만듭니다.

사전 조건

EMR서버리스에서 대화형 워크로드를 사용하려면 다음 요구 사항을 충족해야 합니다.

- EMRAmazon EMR 6.14.0 이상에서는 서버리스 대화형 애플리케이션이 지원됩니다.
- 대화형 애플리케이션에 액세스하고, 제출한 워크로드를 실행하고, EMR Studio에서 대화형 노트북을 실행하려면 특정 권한과 역할이 필요합니다. 자세한 내용은 대화형 워크로드에 필요한 권한 단원을 참조하십시오.

대화형 워크로드에 필요한 권한

EMR서버리스에 액세스하는 데 필요한 기본 권한 외에도 IAM ID 또는 역할에 대한 추가 권한을 구성해야 합니다.

대화형 애플리케이션에 액세스하려면

EMRStudio의 사용자 및 작업 영역 권한을 설정합니다. 자세한 내용은 Amazon EMR 관리 안내서의 EMR Studio 사용자 권한 구성을 참조하십시오.

서버리스로 EMR 제출한 워크로드를 실행하려면

작업 런타임 역할을 설정합니다. 자세한 내용은 작업 런타임 역할 생성 단원을 참조하십시오.

Studio에서 EMR 대화형 노트북을 실행하려면

Studio 사용자에 대한 IAM 정책에 다음과 같은 추가 권한을 추가하십시오.

- **`emr-serverless:AccessInteractiveEndpoints`**- 지정한 대화형 응용 프로그램에 액세스하고 연결할 수 있는 권한을 `Resource` 부여합니다. 이 권한은 EMR Studio Workspace에서 EMR 서버리스 애플리케이션에 연결하는 데 필요합니다.
- **`iam:PassRole`**- 애플리케이션에 연결할 때 사용하려는 IAM 실행 역할에 액세스할 수 있는 권한을 부여합니다. EMRStudio Workspace에서 EMR 서버리스 애플리케이션에 연결하려면 적절한 `PassRole` 권한이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessInteractiveAccess",
            "Effect": "Allow",
            "Action": "emr-serverless:AccessInteractiveEndpoints",
            "Resource": "arn:aws:emr-serverless:Region:account:/applications/*"
        },
        {
            "Sid": "EMRServerlessRuntimeRoleAccess",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "interactive-execution-role-ARN",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        }
    ]
}
```

대화형 애플리케이션 구성

다음 상위 단계를 사용하여 Amazon EMR Studio의 대화형 기능을 갖춘 EMR 서버리스 애플리케이션을 생성하십시오. AWS Management Console.

1. 의 Amazon EMR 서버리스 시작하기 단계에 따라 애플리케이션을 생성하십시오.
2. 그런 다음 EMR Studio에서 작업 영역을 시작하고 EMR 서버리스 애플리케이션에 컴퓨팅 옵션으로 연결합니다. 자세한 내용은 EMR서버리스 시작하기 설명서의 2단계에서 대화형 워크로드 탭을 참조하십시오.

애플리케이션을 Studio Workspace에 연결하면 애플리케이션이 아직 실행 중이 아닌 경우 애플리케이션 시작이 자동으로 트리거됩니다. 또한 응용 프로그램을 미리 시작하고 Workspace에 연결하기 전에 준비된 상태로 유지할 수도 있습니다.

대화형 응용 프로그램 관련 고려 사항

- EMRAmazon EMR 6.14.0 이상에서는 서버리스 대화형 애플리케이션이 지원됩니다.
- EMRStudio는 EMR 서버리스 대화형 애플리케이션과 통합된 유일한 클라이언트입니다. EMR서버리스 대화형 애플리케이션에서는 작업 공간 협업, SQL Explorer, 노트북의 프로그래밍 방식 실행과 같은 EMR Studio 기능이 지원되지 않습니다.
- 대화형 애플리케이션은 Spark 엔진에서만 지원됩니다.
- 대화형 애플리케이션은 Python 3 PySpark 및 Spark Scala 커널을 지원합니다.
- 단일 대화형 애플리케이션에서 최대 25개의 동시 노트북을 실행할 수 있습니다.
- 대화형 응용 프로그램이 있는 자체 호스팅 Jupyter 노트북을 지원하는 엔드포인트 또는 API 인터페이스는 없습니다.
- 최적화된 시작 환경을 위해 드라이버와 실행기에 대해 사전 초기화된 용량을 구성하고 애플리케이션을 미리 시작하는 것이 좋습니다. 애플리케이션을 미리 시작하면 Workspace에 연결할 준비가 되었는지 확인해야 합니다.

```
aws emr-serverless start-application \
--application-id your-application-id
```

- 기본적으로 `autoStopConfig` 애플리케이션에는 활성화되어 있습니다. 이렇게 하면 30분의 유휴 시간이 지나면 애플리케이션이 종료됩니다. OR 요청의 일부로 이 구성을 변경할 `create-application` 수 `update-application` 있습니다.

- 대화형 응용 프로그램을 사용하는 경우 노트북을 실행하기 위해 미리 초기화된 커널, 드라이버 및 실행기 용량을 구성하는 것이 좋습니다. 각 Spark 대화형 세션에는 커널 하나와 드라이버 하나가 필요하므로 EMR 서버리스는 사전 초기화된 모든 드라이버에 대해 사전 초기화된 커널 워커를 유지 관리합니다. 기본적으로 EMR 서버리스는 드라이버에 대해 사전 초기화된 용량을 지정하지 않더라도 전체 애플리케이션에서 커널 작업자 1명의 사전 초기화된 용량을 유지합니다. 각 커널 워커는 4v CPU 및 16GB의 메모리를 사용합니다. 현재 요금 정보는 Amazon EMR 요금 페이지를 참조하십시오.
- v CPU 서비스 할당량이 충분해야 합니다. AWS 계정 대화형 워크로드를 실행하기 위함입니다. Lake Formation 지원 워크로드를 실행하지 않는 경우 최소 24v를 사용하는 것이 좋습니다. CPU 그럴 경우 최소 28v를 사용하는 것이 좋습니다. CPU
- EMR서버리스는 노트북이 60분 이상 유휴 상태인 경우 노트북에서 커널을 자동으로 종료합니다. EMR서버리스는 노트북 세션 중에 완료된 마지막 활동을 기준으로 커널 유휴 시간을 계산합니다. 현재 커널 유휴 제한 시간 설정을 수정할 수 없습니다.
- 대화형 워크로드로 Lake Formation을 `spark.emr-serverless.lakeformation.enabled` 활성화하려면 EMR서버리스 애플리케이션을 생성할 때 구성을 `runtime-configuration` 개체의 `spark-defaults` 분류 `true` 아래로 설정하십시오. EMR서버리스에서 Lake Formation을 활성화하는 방법에 대해 자세히 알아보려면 EMRAmazon에서 Lake Formation 활성화를 참조하십시오.

Apache Livy 엔드포인트를 통해 EMR 서버리스로 대화형 워크로드를 실행하십시오.

Amazon EMR 릴리스 6.14.0 이상에서는 EMR 서버리스 애플리케이션을 생성하는 동안 Apache Livy 엔드포인트를 생성 및 활성화하고 자체 호스팅 노트북이나 사용자 지정 클라이언트를 통해 대화형 워크로드를 실행할 수 있습니다. Apache Livy 엔드포인트는 다음과 같은 이점을 제공합니다.

- Jupyter 노트북을 통해 Apache Livy 엔드포인트에 안전하게 연결하고 Apache Livy의 인터페이스를 사용하여 Apache Spark 워크로드를 관리할 수 있습니다. REST
- Apache Spark 워크로드의 데이터를 사용하는 대화형 웹 애플리케이션에 Apache Livy REST API 작업을 사용하십시오.

사전 조건

EMR서버리스에서 Apache Livy 엔드포인트를 사용하려면 다음 요구 사항을 충족해야 합니다.

- Amazon EMR 서버리스 시작하기의 단계를 완료하십시오.

- Apache Livy 엔드포인트를 통해 대화형 워크로드를 실행하려면 특정 권한과 역할이 필요합니다. 자세한 내용은 대화형 워크로드에 필요한 권한을 참조하십시오.

필수 권한

EMR서버리스에 액세스하는 데 필요한 권한 외에도 Apache Livy 엔드포인트에 액세스하고 애플리케이션을 실행하려면 IAM 역할에 다음 권한을 추가해야 합니다.

- `emr-serverless:AccessLivyEndpoints`— 지정한 Livy 지원 애플리케이션에 액세스하고 연결할 수 있는 권한을 부여합니다. `Resource` Apache Livy 엔드포인트에서 사용 가능한 REST API 작업을 실행하려면 이 권한이 필요합니다.
- `iam:PassRole`— Apache Livy 세션을 생성하는 동안 IAM 실행 역할에 액세스할 수 있는 권한을 부여합니다. EMR서버리스는 이 역할을 사용하여 워크로드를 실행합니다.
- `emr-serverless:GetDashboardForJobRun`— Spark Live UI 및 드라이버 로그 링크를 생성할 권한을 부여하고 Apache Livy 세션 결과의 일부로 로그에 대한 액세스 권한을 제공합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "EMRServerlessInteractiveAccess",
            "Effect": "Allow",
            "Action": "emr-serverless:AccessLivyEndpoints",
            "Resource": "arn:aws:emr-serverless:<AWS_REGION>:account:/applications/*"
        },
        {
            "Sid": "EMRServerlessRuntimeRoleAccess",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "execution-role-ARN",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
        },
        {
            "Sid": "EMRServerlessDashboardAccess",
            "Effect": "Allow",
            "Action": "emr-serverless:GetDashboardForJobRun",
```

```
            "Resource": "arn:aws:emr-serverless:<AWS_REGION>:account:/applications/*"
        }
    ]
}
```

시작하기

1. Apache Livy를 지원하는 애플리케이션을 만들려면 다음 명령을 실행합니다.

```
 aws emr-serverless create-application \
--name my-application-name \
--type 'application-type' \
 --release-label <Amazon EMR-release-version>
--interactive-configuration '{"livyEndpointEnabled": true}'
```

2. EMR서버리스에서 애플리케이션을 생성한 후 애플리케이션을 시작하여 Apache Livy 엔드포인트를 사용할 수 있도록 합니다.

```
 aws emr-serverless start-application \
 --application-id application-id
```

   다음 명령을 사용하여 애플리케이션 상태를 확인합니다. 상태가 `STARTED` 되면 Apache Livy 엔드포인트에 액세스할 수 있습니다.

```
aws emr-serverless get-application \
--region <AWS_REGION> --application-id >application_id>
```

3. 다음을 사용하여 URL 엔드포인트에 액세스하십시오.

```
https://_<application-id>_.livy.emr-serverless-
services._<AWS_REGION>_.amazonaws.com
```

엔드포인트가 준비되면 사용 사례에 따라 워크로드를 제출할 수 있습니다. SIGv4프로토콜을 사용하여 엔드포인트에 대한 모든 요청에 서명하고 인증 헤더를 전달해야 합니다. 다음 방법을 사용하여 워크로드를 실행할 수 있습니다.

- HTTP클라이언트 — 사용자 지정 클라이언트와 함께 Apache Livy 엔드포인트 API 작업을 제출해야 합니다. HTTP

- Sparkmagic 커널 — sparkmagic 커널을 로컬에서 실행하고 Jupyter 노트북을 사용하여 대화형 쿼리를 제출해야 합니다.

HTTP클라이언트

Apache Livy 세션을 생성하려면 요청 본문의 `conf` 파라미터를 `emr-serverless.session.executionRoleArn` 제출해야 합니다. 다음 예는 샘플 `POST /sessions` 요청입니다.

```
{
    "kind": "pyspark",
    "heartbeatTimeoutInSecond": 60,
    "conf": {
        "emr-serverless.session.executionRoleArn": "<executionRoleArn>"
    }
}
```

다음 표에서는 사용 가능한 모든 Apache Livy API 작업에 대해 설명합니다.

API오퍼레이션	설명
GET/세션	모든 활성 대화형 세션의 목록을 반환합니다.
POST/session	스파크 또는 파이스파크를 통해 새로운 대화형 세션을 만듭니다.
GET/세션/ <*sessionId* >	세션 정보를 반환합니다.
GET/세션/ <*sessionId* /세션//상태	세션 상태를 반환합니다.
DELETE/세션/ <*sessionId* >	세션을 중지하고 삭제합니다.
GET/세션/ <*sessionId* /세션//성명서	세션의 모든 명령문을 반환합니다.
POST/세션/ <*sessionId* /세션//성명서	세션에서 명령문을 실행합니다.
GET/세션/ <*sessionId* /세션//성명서/ <*statementId* >	세션에서 지정된 명령문의 세부 정보를 반환합니다.

API오퍼레이션	설명
POST/세션/ <*sessionId* /세션//성명서/ <*statementId* >/성명서//취소	이 세션에서 지정된 명령문을 취소합니다.

Apache Livy 엔드포인트로 요청 전송

클라이언트에서 Apache Livy 엔드포인트로 직접 요청을 보낼 수도 있습니다. HTTP 이렇게 하면 노트북 외부에서 사용 사례에 맞는 코드를 원격으로 실행할 수 있습니다.

엔드포인트로 요청을 보내기 전에 먼저 다음 라이브러리를 설치해야 합니다.

```
pip3 install botocore awscrt requests
```

다음은 엔드포인트에 직접 HTTP 요청을 보내는 샘플 Python 스크립트입니다.

```
from botocore import crt
import requests
from botocore.awsrequest import AWSRequest
from botocore.credentials import Credentials
import botocore.session
import json, pprint, textwrap

endpoint = 'https://<application_id>.livy.emr-serverless-
services-<AWS_REGION>.amazonaws.com'
headers = {'Content-Type': 'application/json'}

session = botocore.session.Session()
signer = crt.auth.CrtS3SigV4Auth(session.get_credentials(), 'emr-serverless',
 '<AWS_REGION>')


### Create session request

data = {'kind': 'pyspark', 'heartbeatTimeoutInSecond': 60, 'conf': { 'emr-
serverless.session.executionRoleArn': 'arn:aws:iam::123456789012:role/role1'}}

request = AWSRequest(method='POST', url=endpoint + "/sessions", data=json.dumps(data),
 headers=headers)

request.context["payload_signing_enabled"] = False
```

```
signer.add_auth(request)

prepped = request.prepare()

r = requests.post(prepped.url, headers=prepped.headers, data=json.dumps(data))

pprint.pprint(r.json())


### List Sessions Request

request = AWSRequest(method='GET', url=endpoint + "/sessions", headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r2 = requests.get(prepped.url, headers=prepped.headers)
pprint.pprint(r2.json())


### Get session state

session_url = endpoint + r.headers['location']

request = AWSRequest(method='GET', url=session_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r3 = requests.get(prepped.url, headers=prepped.headers)

pprint.pprint(r3.json())


### Submit Statement

data = {
```

```
      'code': "1 + 1"
}

statements_url = endpoint + r.headers['location'] + "/statements"

request = AWSRequest(method='POST', url=statements_url, data=json.dumps(data),
 headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r4 = requests.post(prepped.url, headers=prepped.headers, data=json.dumps(data))

pprint.pprint(r4.json())

### Check statements results

specific_statement_url = endpoint + r4.headers['location']

request = AWSRequest(method='GET', url=specific_statement_url, headers=headers)

request.context["payload_signing_enabled"] = False

signer.add_auth(request)

prepped = request.prepare()

r5 = requests.get(prepped.url, headers=prepped.headers)

pprint.pprint(r5.json())


### Delete session


session_url = endpoint + r.headers['location']

request = AWSRequest(method='DELETE', url=session_url, headers=headers)

request.context["payload_signing_enabled"] = False
```

```
signer.add_auth(request)

prepped = request.prepare()

r6 = requests.delete(prepped.url, headers=prepped.headers)

pprint.pprint(r6.json())
```

스파크매직 커널

sparkmagic을 설치하기 전에 구성을 완료했는지 확인하세요. AWS sparkmagic을 설치하려는 인스턴스의 자격 증명

1. 설치 단계에 따라 sparkmagic을 설치하세요. 처음 네 단계만 수행하면 된다는 점에 유의하세요.
2. sparkmagic 커널은 사용자 지정 인증자를 지원하므로 인증자를 sparkmagic 커널과 통합하여 모든 요청이 서명되도록 할 수 있습니다. SIGv4
3. EMR서버리스 사용자 지정 인증자를 설치하세요.

   ```
   pip install emr-serverless-customauth
   ```
4. 이제 sparkmagic 구성 json 파일에 사용자 지정 인증자 및 Apache Livy 엔드포인트에 URL 대한 경로를 제공하십시오. 다음 명령을 사용하여 구성 파일을 엽니다.

   ```
   vim ~/.sparkmagic/config.json
   ```

   다음은 샘플 `config.json` 파일입니다.

   ```
   {
   "kernel_python_credentials" : {
       "username": "",
       "password": "",
       "url": "https://<application-id>.livy.emr-serverless-
   services.<AWS_REGION>.amazonaws.com",
       "auth": "Custom_Auth"
     },

     "kernel_scala_credentials" : {
       "username": "",
       "password": "",
       "url": "https://<application-id>.livy.emr-serverless-
   services.<AWS_REGION>.amazonaws.com",
   ```

```
    "auth": "Custom_Auth"
  },
  "authenticators": {
    "None": "sparkmagic.auth.customauth.Authenticator",
    "Basic_Access": "sparkmagic.auth.basic.Basic",
    "Custom_Auth":
 "emr_serverless_customauth.customauthenticator.EMRServerlessCustomSigV4Signer"
  },
  "livy_session_startup_timeout_seconds": 600,
  "ignore_ssl_errors": false
}
```

5. Jupyter 랩을 시작합니다. 마지막 단계에서 설정한 사용자 지정 인증을 사용해야 합니다.
6. 그런 다음 다음 노트북 명령어와 코드를 실행하여 시작할 수 있습니다.

```
%%info //Returns the information about the current sessions.
```

```
%%configure -f //Configure information specific to a session. We supply
 executionRoleArn in this example. Change it for your use case.
{
    "driverMemory": "4g",
    "conf": {
          "emr-serverless.session.executionRoleArn":
 "arn:aws:iam::123456789012:role/JobExecutionRole"
    }
}
```

```
<your code>//Run your code to start the session
```

내부적으로 각 명령은 구성된 Apache Livy 엔드포인트를 통해 각 Apache Livy API 작업을 호출합니다. URL 그런 다음 사용 사례에 따라 지침을 작성할 수 있습니다.

고려 사항

Apache Livy 엔드포인트를 통해 대화형 워크로드를 실행할 때는 다음 고려 사항을 고려하십시오.

- EMR서버리스는 호출자 보안 주체를 사용하여 세션 수준 격리를 유지합니다. 세션을 생성한 발신자 주체만 해당 세션에 액세스할 수 있습니다. 보다 세분화된 격리를 위해 자격 증명을 사용할 때 소스 ID를 구성할 수 있습니다. 이 경우 EMR 서버리스는 발신자 주체와 소스 ID 모두를 기반으로 세션 수

준 격리를 적용합니다. 소스 ID에 대한 자세한 내용은 수임된 역할로 수행된 작업 모니터링 및 제어를 참조하십시오.

- Apache Livy 엔드포인트는 EMR 서버리스 릴리스 6.14.0 이상에서 지원됩니다.
- 아파치 Livy 엔드포인트는 아파치 스파크 엔진에서만 지원됩니다.
- 아파치 Livy 엔드포인트는 스칼라 스파크 및 을 지원합니다. PySpark
- 애플리케이션에서는 기본적으로 `autoStopConfig` 활성화되어 있습니다. 즉, 15분 동안 유휴 상태가 되면 애플리케이션이 종료됩니다. OR `update-application` 요청의 일부로 이 구성을 변경할 `create-application` 수 있습니다.
- 단일 Apache Livy 엔드포인트 지원 애플리케이션에서 최대 25개의 동시 세션을 실행할 수 있습니다.
- 최상의 시작 환경을 위해 드라이버 및 실행자에 대해 사전 초기화된 용량을 구성하는 것이 좋습니다.
- Apache Livy 엔드포인트에 연결하기 전에 애플리케이션을 수동으로 시작해야 합니다.
- v CPU 서비스 할당량이 충분해야 합니다. AWS 계정 Apache Livy 엔드포인트로 대화형 워크로드를 실행하기 위함입니다. 최소 24v를 사용하는 것이 좋습니다. CPU
- 기본 Apache Livy 세션 제한 시간은 1시간입니다. 명령문을 한 시간 동안 실행하지 않으면 Apache Livy는 세션을 삭제하고 드라이버와 실행기를 해제합니다. 이 구성은 변경할 수 없습니다.
- 활성 세션만 Apache Livy 엔드포인트와 상호 작용할 수 있습니다. 세션이 완료, 취소 또는 종료되면 Apache Livy 엔드포인트를 통해 세션에 액세스할 수 없습니다.

로깅 및 모니터링

모니터링은 EMR 서버리스 애플리케이션 및 작업의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 다중 지점 오류가 발생할 경우 이를 보다 쉽게 디버깅할 수 있도록 EMR 서버리스 솔루션의 모든 부분에서 모니터링 데이터를 수집해야 합니다.

주제

로그 저장

서버리스에서 작업 진행 상황을 모니터링하고 작업 실패 문제를 해결하려면 EMR 서버리스에서 애플리케이션 로그를 저장하고 제공하는 방법을 EMR 선택할 수 있습니다. 작업 실행을 제출할 때 관리형 스토리지, Amazon S3, Amazon을 로깅 옵션으로 지정할 수 있습니다. CloudWatch

를 사용하여 사용하려는 로그 유형 및 로그 위치를 지정하거나 기본 유형 및 위치를 그대로 사용할 수 있습니다. CloudWatch CloudWatch 로그에 대한 자세한 내용은 을 참조하십시오the section called “아마존 CloudWatch”. 관리형 스토리지와 S3 로깅의 경우, 다음 표는 관리형 스토리지, Amazon S3 버킷 또는 둘 다를 선택할 경우 기대할 수 있는 로그 위치 및 UI 가용성을 보여줍니다.

옵션	이벤트 로그	컨테이너 로그	애플리케이션 UI
관리형 스토리지	관리형 스토리지에 저장	관리형 스토리지에 저장	지원
관리형 스토리지와 S3 버킷 모두	두 곳에 모두 저장	S3 버킷에 저장	지원
Amazon S3 버킷	S3 버킷에 저장	S3 버킷에 저장	지원되지 않음 [1]

[1] 관리형 스토리지 옵션을 선택한 상태로 유지하는 것이 좋습니다. 그렇지 않으면 기본 제공 애플리케이션을 사용할 수 없습니다UIs.

관리형 스토리지를 사용한 EMR 서버리스 로깅

기본적으로 EMR 서버리스는 최대 30일 동안 Amazon EMR 관리 스토리지에 애플리케이션 로그를 안전하게 저장합니다.

> **Note**
>
> 기본 옵션을 끄면 Amazon에서 사용자를 대신하여 작업 문제를 해결할 EMR 수 없습니다.

EMRStudio에서 이 옵션을 끄려면 허용을 선택 취소하십시오. AWS 작업 제출 페이지의 추가 설정 섹션에서 30일 동안 로그를 보존하려면 확인란을 선택합니다.

이 옵션을 끄려면 다음을 사용하십시오. AWS CLI작업 실행을 제출할 때 `managedPersistenceMonitoringConfiguration` 구성을 사용하십시오.

```
{
    "monitoringConfiguration": {
        "managedPersistenceMonitoringConfiguration": {
            "enabled": false
        }
    }
}
```

Amazon S3 버킷을 사용한 EMR 서버리스 로깅

작업에서 Amazon S3로 로그 데이터를 전송하려면 먼저 작업 런타임 역할에 대한 권한 정책에 다음 권한을 포함해야 합니다. 로깅 버킷 *DOC-EXAMPLE-BUCKET-LOGGING* 이름으로 바꾸십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET-LOGGING/*"
            ]
        }
    ]
}
```

Amazon S3 버킷에서 로그를 저장하도록 설정하려면 AWS CLI작업 실행을 시작할 때 `s3MonitoringConfiguration` 구성을 사용하십시오. 이렇게 하려면 구성에 다음을 `--configuration-overrides` 제공하십시오.

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING/logs/"
        }
    }
}
```

재시도가 활성화되지 않은 일괄 작업의 경우 EMR 서버리스는 로그를 다음 경로로 보냅니다.

```
'/applications/<applicationId>/jobs/<jobId>'
```

EMR서버리스 릴리스 7.1.0 이상에서는 스트리밍 작업 및 일괄 작업에 대한 재시도를 지원합니다. 재시도가 활성화된 상태에서 작업을 실행하는 경우 EMR 서버리스는 로그 경로 접두사에 시도 횟수를 자동으로 추가하므로 로그를 더 잘 구분하고 추적할 수 있습니다.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'
```

Amazon을 통한 EMR 서버리스 로깅 CloudWatch

EMR서버리스 애플리케이션에 작업을 제출할 때 애플리케이션 로그를 저장하는 CloudWatch 옵션으로 Amazon을 선택할 수 있습니다. 이를 통해 CloudWatch 로그 인사이트 및 라이브 테일과 같은 CloudWatch 로그 분석 기능을 사용할 수 있습니다. 추가 분석 등을 CloudWatch 위해 로그를 다른 시스템으로 스트리밍할 수도 있습니다. OpenSearch

EMR서버리스는 드라이버 로그에 대한 실시간 로깅을 제공합니다. CloudWatch 라이브 테일 기능을 사용하거나 CloudWatch CLI tail 명령을 통해 실시간으로 로그를 볼 수 있습니다.

EMR서버리스의 경우 기본적으로 CloudWatch 로깅이 비활성화됩니다. 활성화하려면 의 AWS CLI 구성을 참조하십시오.

> **Note**
>
> CloudWatch Amazon은 로그를 실시간으로 게시하므로 작업자에게 더 많은 리소스가 필요합니다. 낮은 작업자 수용 인원을 선택하면 작업 실행 시간에 미치는 영향이 커질 수 있습니다. CloudWatch 로깅을 활성화하는 경우 더 큰 작업자 용량을 선택하는 것이 좋습니다. 초당 트랜잭션 (TPS) 속도가 너무 낮으면 로그 게시가 제한될 수도 있습니다. `PutLogEvents` CloudWatch 제한 구성은 서버리스를 포함한 모든 서비스에 적용됩니다. EMR 자세한 내용은 로그의 스로틀링을 어떻게 결정하나요? 를 참조하십시오. CloudWatch 켜짐 AWS re:포스트.

로그인에 필요한 권한 CloudWatch

작업에서 CloudWatch Amazon으로 로그 데이터를 보내려면 먼저 작업 런타임 역할에 대한 권한 정책에 다음 권한을 포함해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:AWS ##:111122223333:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
               "logs:PutLogEvents",
               "logs:CreateLogGroup",
               "logs:CreateLogStream",
               "logs:DescribeLogStreams"
            ],
            "Resource": [
               "arn:aws:logs:AWS ##:111122223333:log-group:my-log-group-name:*"
            ]
        }
    ]
}
```

AWS CLI

Amazon에서 EMR 서버리스 로그를 CloudWatch 저장하도록 설정하려면 AWS CLI작업 실행을 시작할 때 `cloudWatchLoggingConfiguration` 구성을 사용하십시오. 이렇게 하려면 다음과 같은 구성 재정의를 제공하십시오. 선택적으로 로그 그룹 이름, 로그 스트림 접두사 이름, 로그 유형 및 암호화 키를 제공할 수도 있습니다. ARN

선택적 값을 지정하지 않는 경우 는 로그를 기본 로그 스트림과 함께 기본 로그 그룹에 CloudWatch `/aws/emr-serverless` 게시합니다. `/applications/`*`applicationId`*`/jobs/`*`jobId`*`/`*`worker-type`*

EMR서버리스 릴리스 7.1.0 이상에서는 스트리밍 작업 및 일괄 작업에 대한 재시도를 지원합니다. 작업 재시도를 활성화한 경우 EMR 서버리스는 로그 경로 접두사에 시도 횟수를 자동으로 추가하므로 로그를 더 잘 구분하고 추적할 수 있습니다.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/worker-type'
```

다음은 EMR 서버리스의 기본 설정으로 Amazon CloudWatch 로깅을 활성화하는 데 필요한 최소 구성을 보여줍니다.

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true
         }
     }
}
```

다음 예는 EMR 서버리스용 Amazon CloudWatch 로깅을 활성화할 때 지정할 수 있는 모든 필수 및 선택적 구성을 보여줍니다. 지원되는 `logTypes` 값도 이 예제 아래에 나열되어 있습니다.

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true, // Required
            "logGroupName": "Example_logGroup", // Optional
            "logStreamNamePrefix": "Example_logStream", // Optional
            "encryptionKeyArn": "key-arn", // Optional
            "logTypes": {
                "SPARK_DRIVER": ["stdout", "stderr"] //List of values
```

```
            }
        }
    }
}
```

기본적으로 EMR 서버리스는 드라이버 stdout 및 stderr 로그만 게시합니다. CloudWatch 다른 로그를 원하는 경우 필드를 사용하여 컨테이너 역할과 해당 로그 유형을 지정할 수 있습니다. `logTypes`

다음 목록은 `logTypes` 구성에 지정할 수 있는 지원되는 작업자 유형을 보여줍니다.

Spark

- `SPARK_DRIVER : ["STDERR", "STDOUT"]`
- `SPARK_EXECUTOR : ["STDERR", "STDOUT"]`

Hive

- `HIVE_DRIVER : ["STDERR", "STDOUT", "HIVE_LOG", "TEZ_AM"]`
- `TEZ_TASK : ["STDERR", "STDOUT", "SYSTEM_LOGS"]`

순환 로그

Amazon EMR 서버리스는 Spark 애플리케이션 로그와 이벤트 로그를 교체할 수 있습니다. 로그 로테이션은 디스크 공간을 모두 차지할 수 있는 대용량 로그 파일을 생성하는 장기 실행 작업의 문제를 해결하는 데 도움이 됩니다. 로그를 순환하면 디스크 저장소를 절약하고 디스크에 더 이상 남은 공간이 없으므로 작업 실패 횟수를 줄일 수 있습니다.

로그 회전은 기본적으로 활성화되며 Spark 작업에만 사용할 수 있습니다.

Spark 이벤트 로그

> **Note**
>
> Spark 이벤트 로그 로테이션은 모든 Amazon EMR 릴리스 라벨에서 사용할 수 있습니다.

EMRServerless는 단일 이벤트 로그 파일을 생성하는 대신 일정 간격으로 이벤트 로그를 순환시키고 이전 이벤트 로그 파일을 제거합니다. 로그를 순환해도 S3 버킷에 업로드된 로그에는 영향을 주지 않습니다.

Spark 애플리케이션 로그

> **Note**
>
> Spark 애플리케이션 로그 로테이션은 모든 Amazon EMR 릴리스 라벨에서 사용할 수 있습니다.

EMR또한 서버리스는 드라이버와 실행자 (예: 및 파일) 의 스파크 애플리케이션 로그를 순환시킵니다. `stdout` `stderr` Spark History Server 및 Live UI 링크를 사용하여 Studio의 로그 링크를 선택하여 최신 로그 파일에 액세스할 수 있습니다. 로그 파일은 최신 로그의 잘린 버전입니다. 이전의 순환된 로그를 보려면 로그를 저장할 때 Amazon S3 위치를 지정해야 합니다. 자세한 내용은 Amazon S3 버킷을 사용한 EMR 서버리스 로깅을 참조하십시오.

다음 위치에서 최신 로그 파일을 찾을 수 있습니다. EMR서버리스는 15초마다 파일을 새로 고칩니다. 이러한 파일의 범위는 0MB에서 128MB 사이입니다.

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/stderr.gz
```

다음 위치에는 이전에 회전된 파일이 들어 있습니다. 각 파일은 128MB입니다.

```
<example-S3-logUri>/applications/<application-id>/jobs/<job-id>/SPARK_DRIVER/archived/
stderr_<index>.gz
```

Spark 실행기에도 동일한 동작이 적용됩니다. 이 변경은 S3 로깅에만 적용됩니다. 로그 로테이션으로 인해 Amazon에 업로드된 로그 스트림은 변경되지 않습니다 CloudWatch.

EMR서버리스 릴리스 7.1.0 이상에서는 스트리밍 및 일괄 작업에 대한 재시도를 지원합니다. 작업에 대해 재시도를 활성화한 경우 EMR 서버리스는 해당 작업의 로그 경로에 접두사를 추가하여 로그를 더 잘 추적하고 서로 구분할 수 있도록 합니다. 이 경로에는 순환된 모든 로그가 포함됩니다.

```
'/applications/<applicationId>/jobs/<jobId>/attempts/<attemptNumber>/'.
```

로그 암호화

관리형 스토리지를 통한 EMR 서버리스 로그 암호화

자체 KMS 키로 관리형 스토리지의 로그를 암호화하려면 작업 실행을 제출할 때 `managedPersistenceMonitoringConfiguration` 구성을 사용하십시오.

```
{
```

```
    "monitoringConfiguration": {
        "managedPersistenceMonitoringConfiguration" : {
            "encryptionKeyArn": "key-arn"
        }
    }
}
```

Amazon S3 버킷을 사용한 EMR 서버리스 로그 암호화

Amazon S3 버킷의 로그를 자체 KMS 키로 암호화하려면 작업 실행을 제출할 때 `s3MonitoringConfiguration` 구성을 사용하십시오.

```
{
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET-LOGGING/logs/",
            "encryptionKeyArn": "key-arn"
        }
    }
}
```

Amazon을 사용한 EMR 서버리스 로그 암호화 CloudWatch

CloudWatch Amazon에서 자체 KMS 키로 로그를 암호화하려면 작업 실행을 제출할 때 `cloudWatchLoggingConfiguration` 구성을 사용하십시오.

```
{
    "monitoringConfiguration": {
        "cloudWatchLoggingConfiguration": {
            "enabled": true,
            "encryptionKeyArn": "key-arn"
         }
     }
}
```

로그 암호화에 필요한 권한

이 섹션의 내용

- 필수 사용자 권한
- Amazon S3 및 관리형 스토리지에 대한 암호화 키 권한

- Amazon의 암호화 키 권한 CloudWatch

필수 사용자 권한

작업을 제출하거나 로그 또는 애플리케이션을 보는 사용자는 키를 사용할 권한이 UIs 있어야 합니다. KMS키 정책이나 사용자, 그룹 또는 역할에 대한 IAM 정책에서 권한을 지정할 수 있습니다. 작업을 제출하는 사용자에게 KMS 키 권한이 없는 경우 EMR Serverless는 작업 실행 제출을 거부합니다.

키 정책 예시

다음 키 정책은 `kms:GenerateDataKey` 및 에 대한 권한을 제공합니다`kms:Decrypt`.

```
{
    "Effect": "Allow",
    "Principal":{
       "AWS": "arn:aws:iam::111122223333:user/user-name"
     },
     "Action": [
       "kms:GenerateDataKey",
       "kms:Decrypt"
      ],
     "Resource": "*"
 }
```

예제 IAM 정책

다음 IAM 정책은 `kms:GenerateDataKey` 및 에 대한 권한을 제공합니다`kms:Decrypt`.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt"
      ],
     "Resource": "key-arn"
   }
}
```

Spark 또는 Tez UI를 시작하려면 사용자, 그룹 또는 역할에 다음과 `emr-serverless:GetDashboardForJobRun` API 같이 액세스할 수 있는 권한을 부여해야 합니다.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "emr-serverless:GetDashboardForJobRun"
      ]
   }
}
```

Amazon S3 및 관리형 스토리지에 대한 암호화 키 권한

관리형 스토리지 또는 S3 버킷에서 자체 암호화 키로 로그를 암호화하는 경우 다음과 같이 KMS 키 권한을 구성해야 합니다.

`emr-serverless.amazonaws.com`보안 주체는 해당 키에 대한 정책에서 다음과 같은 권한을 가지고 있어야 합니다KMS.

```
{
    "Effect": "Allow",
    "Principal":{
       "Service": "emr-serverless.amazonaws.com"
     },
     "Action": [
       "kms:Decrypt",
       "kms:GenerateDataKey"
      ],
     "Resource": "*"
     "Condition": {
       "StringLike": {
         "aws:SourceArn": "arn:aws:emr-serverless:region:aws-account-id:/
applications/application-id"
       }
     }
 }
```

보안 모범 사례로서 키 정책에 `aws:SourceArn` 조건 키를 추가하는 것이 좋습니다. KMS IAM글로벌 조건 키는 EMR 서버리스가 응용 ARN 프로그램에만 KMS 키를 사용하도록 하는 `aws:SourceArn` 데 도움이 됩니다.

작업 런타임 역할의 IAM 정책에는 다음과 같은 권한이 있어야 합니다.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt"
      ],
     "Resource": "key-arn"
   }
}
```

Amazon의 암호화 키 권한 CloudWatch

KMS키를 로그 그룹에 ARN 연결하려면 작업 런타임 역할에 대해 다음 IAM 정책을 사용하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "logs:AssociateKmsKey"
        ],
        "Resource": [
            "arn:aws:logs:AWS ##:111122223333:log-group:my-log-group-name:*"
        ]
    }
}
```

Amazon에 KMS 권한을 부여하도록 KMS 키 정책을 구성합니다 CloudWatch.

```
{
    "Version": "2012-10-17",
    "Id": "key-default-1",
    "Statement":
     {
        "Effect": "Allow",
        "Principal": {
            "Service": "logs.AWS ##.amazonaws.com"
        },
        "Action": [
            "kms:Decrypt",
```

```
            "kms:GenerateDataKey",
        ],
        "Resource": "*",
        "Condition": {
            "ArnLike": {
                "kms:EncryptionContext:aws:logs:arn": "arn:aws:logs:AWS #
#:111122223333:*"
            }
        }
    }
}
```

아마존 서버리스의 아파치 Log4j2 속성 구성 EMR

이 페이지에서는 에서 서버리스 작업에 대한 사용자 지정 Apache Log4j 2.x 속성을 구성하는 방법을 설명합니다. EMR `StartJobRun` 응용 프로그램 수준에서 Log4j 분류를 구성하려면 을 참조하십시오. 서버리스의 기본 애플리케이션 구성 EMR

아마존 서버리스의 스파크 로그4j2 속성 구성 EMR

Amazon EMR 릴리스 6.8.0 이상에서는 Apache Log4j 2.x 속성을 사용자 지정하여 세분화된 로그 구성을 지정할 수 있습니다. 이렇게 하면 서버리스에서 Spark 작업의 문제를 간단하게 해결할 수 있습니다. EMR 이러한 속성을 구성하려면 및 분류를 사용하십시오. `spark-driver-log4j2` `spark-executor-log4j2`

주제

- 스파크에 대한 Log4j2 분류
- 스파크의 Log4j2 구성 예제
- 샘플 스파크 작업의 Log4j2
- 스파크에 대한 Log4j2 고려 사항

스파크에 대한 Log4j2 분류

Spark 로그 구성을 사용자 지정하려면 에서 다음 분류를 사용하십시오. `applicationConfiguration` Log4j 2.x 속성을 구성하려면 다음을 사용하십시오. `properties`

`spark-driver-log4j2`

이 분류는 드라이버 `log4j2.properties` 파일의 값을 설정합니다.

spark-executor-log4j2

이 분류는 실행자의 log4j2.properties 파일에 있는 값을 설정합니다.

스파크의 Log4j2 구성 예제

다음 예제는 Spark 드라이버 및 실행기에 대한 Log4j2 구성을 사용자 applicationConfiguration 지정하기 위해 Spark 작업을 제출하는 방법을 보여줍니다.

작업을 제출할 때가 아니라 애플리케이션 수준에서 Log4j 분류를 구성하려면 을 참조하십시오. 서버리스의 기본 애플리케이션 구성 EMR

```
aws emr-serverless start-job-run \
    --application-id application-id \
    --execution-role-arn job-role-arn \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "/usr/lib/spark/examples/jars/spark-examples.jar",
            "entryPointArguments": ["1"],
            "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.executor.cores=4 --conf spark.executor.memory=20g --conf spark.driver.cores=4 --
conf spark.driver.memory=8g --conf spark.executor.instances=1"
        }
    }'
    --configuration-overrides '{
        "applicationConfiguration": [
             {
                "classification": "spark-driver-log4j2",
                "properties": {
                    "rootLogger.level":"error", // will only display Spark error logs
                    "logger.IdentifierForClass.name": "classpath for setting logger",
                    "logger.IdentifierForClass.level": "info"

                }
            },
            {
                "classification": "spark-executor-log4j2",
                "properties": {
                    "rootLogger.level":"error", // will only display Spark error logs
                    "logger.IdentifierForClass.name": "classpath for setting logger",
                    "logger.IdentifierForClass.level": "info"
                }
            }
```

```
    ]
  }'
```

샘플 스파크 작업의 Log4j2

다음 코드 샘플은 애플리케이션에 대한 사용자 지정 Log4j2 구성을 초기화하는 동안 Spark 애플리케이션을 생성하는 방법을 보여줍니다.

Python

Example - Python으로 스파크 작업에 Log4j2 사용하기

```
import os
import sys

from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

app_name = "PySparkApp"
if __name__ == "__main__":
    spark = SparkSession\
        .builder\
        .appName(app_name)\
        .getOrCreate()

    spark.sparkContext._conf.getAll()
    sc = spark.sparkContext
    log4jLogger = sc._jvm.org.apache.log4j
    LOGGER = log4jLogger.LogManager.getLogger(app_name)

    LOGGER.info("pyspark script logger info")
    LOGGER.warn("pyspark script logger warn")
    LOGGER.error("pyspark script logger error")

    // your code here

    spark.stop()
```

Spark 작업을 실행할 때 드라이버에 맞게 Log4j2를 사용자 정의하려면 다음 구성을 사용할 수 있습니다.

```
{
   "classification": "spark-driver-log4j2",
```

```
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
          "logger.PySparkApp.level": "info",
          "logger.PySparkApp.name": "PySparkApp"
      }
}
```

Scala

Example - Scala를 사용한 Spark 작업에 Log4j2 사용

```
import org.apache.log4j.Logger
import org.apache.spark.sql.SparkSession

object ExampleClass {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession
    .builder
    .appName(this.getClass.getName)
    .getOrCreate()

    val logger = Logger.getLogger(this.getClass);
    logger.info("script logging info logs")
    logger.warn("script logging warn logs")
    logger.error("script logging error logs")

// your code here
    spark.stop()
  }
}
```

Spark 작업을 실행할 때 드라이버에 맞게 Log4j2를 사용자 정의하려면 다음 구성을 사용할 수 있습니다.

```
{
   "classification": "spark-driver-log4j2",
      "properties": {
          "rootLogger.level":"error", // only display Spark error logs
          "logger.ExampleClass.level": "info",
          "logger.ExampleClass.name": "ExampleClass"
      }
}
```

스파크에 대한 Log4j2 고려 사항

다음 Log4j2.x 속성은 Spark 프로세스에 대해 구성할 수 없습니다.

- `rootLogger.appenderRef.stdout.ref`
- `appender.console.type`
- `appender.console.name`
- `appender.console.target`
- `appender.console.layout.type`
- `appender.console.layout.pattern`

구성할 수 있는 Log4j2.x 속성에 대한 자세한 내용은 의 파일을 참조하십시오. `log4j2.properties.template` GitHub

서버리스 모니터링 EMR

이 섹션에서는 Amazon EMR Serverless 애플리케이션 및 작업을 모니터링할 수 있는 방법을 다룹니다.

주제

- EMR서버리스 애플리케이션 및 작업 모니터링
- Prometheus용 Amazon 매니지드 서비스를 사용하여 Spark 지표를 모니터링하세요
- EMR서버리스 사용 지표

EMR서버리스 애플리케이션 및 작업 모니터링

EMR서버리스용 Amazon CloudWatch 지표를 사용하면 1분 CloudWatch 지표를 수신하고 CloudWatch 대시보드에 액세스하여 EMR 서버리스 애플리케이션의 near-real-time 운영 및 성능을 볼 수 있습니다.

EMR서버리스는 매분마다 지표를 전송합니다. CloudWatch EMR서버리스는 애플리케이션 수준뿐만 아니라 작업, 작업자 유형 및 수준에서도 이러한 지표를 내보냅니다. capacity-allocation-type

시작하려면 서버리스 리포지토리에 제공된 EMR 서버리스 CloudWatch 대시보드 템플릿을 사용하여 배포하십시오EMR. GitHub

> **Note**
>
> EMR서버리스 대화형 워크로드에는 애플리케이션 수준 모니터링만 활성화되며 작업자 유형 차원이 새로 생겼습니다. `Spark_Kernel` 대화형 워크로드를 모니터링하고 디버깅하려면 Studio Workspace에서 로그와 Apache Spark UI를 볼 수 있습니다. EMR

아래 표에는 네임스페이스 내에서 EMR 사용할 수 있는 서버리스 크기가 설명되어 있습니다. `AWS/EMRServerless`

서버리스 지표의 측정기준 EMR

측정기준	설명
`ApplicationId`	EMR서버리스 애플리케이션의 모든 지표에 대한 필터.
`JobId`	EMR서버리스 작업 실행의 모든 메트릭을 필터링합니다.
`WorkerType`	지정된 작업자 유형의 모든 지표를 필터링합니다. 예를 들어, Spark 작업을 기준으로 필터링하거나 Spark 작업에 `SPARK_EXECUTORS` 대해 필터링할 수 있습니다. `SPARK_DRIVER`
`CapacityAllocationType`	지정된 용량 할당 유형의 모든 지표를 필터링합니다. 예를 들어 사전 초기화된 용량과 `OnDemandCapacity` 그 밖의 모든 용량에 `PreInitCapacity` 대해 필터링할 수 있습니다.

애플리케이션 수준 모니터링

Amazon CloudWatch 지표를 사용하여 EMR 서버리스 애플리케이션 수준에서 용량 사용량을 모니터링할 수 있습니다. CloudWatch 대시보드에서 애플리케이션 용량 사용량을 모니터링하도록 단일 보기를 설정할 수도 있습니다.

EMR서버리스 애플리케이션 메트릭

지표	설명	기본 측정기준	보조 측정기준
CPUAllocated	vCPUs 할당된 총 개수.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
IdleWorkerCount	총 유휴 근로자 수.	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
MaxCPUAllowed	신청서에 CPU 허용되는 최대 한도입니다.	ApplicationId	N/A
MaxMemoryAllowed	애플리케이션에 허용되는 최대 메모리 (GB).	ApplicationId	N/A
MaxStorageAllowed	애플리케이션에 허용되는 최대 스토리지 용량 (GB).	ApplicationId	N/A
MemoryAllocated	할당된 총 메모리 (GB).	ApplicationId	ApplicationId , WorkerType , CapacityAllocationType
PendingCreationWorkerCount	생성을 보류 중인 총 작업자 수.	ApplicationId	ApplicationId , WorkerTyp

지표	설명	기본 측정기준	보조 측정기준
			e , CapacityA llocationType
RunningWo rkerCount	애플리케이션에서 사용 중인 총 작업자 수.	ApplicationId	Applicati onId , WorkerTyp e , CapacityA llocationType
StorageAl located	할당된 총 디스크 스토리지 (GB)	ApplicationId	Applicati onId , WorkerTyp e , CapacityA llocationType
TotalWork erCount	사용 가능한 총 작업자 수	ApplicationId	Applicati onId , WorkerTyp e , CapacityA llocationType

직무 수준 모니터링

Amazon EMR 서버리스는 다음과 같은 작업 수준 지표를 다음으로 전송합니다. Amazon CloudWatch 1분마다. 작업 실행 상태별로 집계 작업 실행에 대한 지표 값을 볼 수 있습니다. 각 지표의 단위는 개수입니다.

EMR서버리스 작업 수준 메트릭

지표	설명	기본 측정기준
SubmittedJobs	제출됨 상태의 작업 수.	ApplicationId
PendingJobs	보류 상태에 있는 작업 수.	ApplicationId
ScheduledJobs	예약됨 상태의 작업 수.	ApplicationId
RunningJobs	실행 중 상태인 작업 수.	ApplicationId
SuccessJobs	성공 상태에 있는 작업 수.	ApplicationId

지표	설명	기본 측정기준
FailedJobs	실패 상태인 작업 수.	ApplicationId
CancellingJobs	취소 상태에 있는 작업 수.	ApplicationId
CancelledJobs	취소됨 상태에 있는 작업 수.	ApplicationId

엔진별 애플리케이션을 사용하여 실행 중인 서버리스 작업과 완료된 EMR 서버리스 작업 모두에 대한 엔진별 지표를 모니터링할 수 있습니다. UIs 실행 중인 작업의 UI를 보면 실시간 업데이트가 포함된 라이브 애플리케이션 UI가 표시됩니다. 완료된 작업의 UI를 보면 영구 앱 UI가 표시됩니다.

작업 실행

실행 중인 EMR 서버리스 작업의 경우 엔진별 지표를 제공하는 실시간 인터페이스를 볼 수 있습니다. Apache Spark UI 또는 Hive Tez UI를 사용하여 작업을 모니터링하고 디버깅할 수 있습니다. 이러한 UIs 기능에 액세스하려면 Studio 콘솔을 사용하거나 EMR Studio 콘솔을 사용하여 보안 엔드포인트를 요청하십시오. URL AWS Command Line Interface.

완료된 작업

완료된 EMR 서버리스 작업의 경우 Spark 기록 서버 또는 Persistent Hive Tez UI를 사용하여 Spark 또는 Hive 작업 실행에 대한 작업 세부 정보, 단계, 작업 및 지표를 볼 수 있습니다. 이러한 UIs 기능에 액세스하려면 Studio 콘솔을 사용하거나 EMR Studio 콘솔을 사용하여 보안 엔드포인트를 요청하세요. URL AWS Command Line Interface.

Job 작업자 수준 모니터링

Amazon EMR Serverless는 AWS/EMRServerless 네임스페이스 및 Job Worker Metrics 지표 그룹에서 사용할 수 있는 다음과 같은 작업자 수준 지표를 Amazon에 보냅니다. CloudWatch EMR서버리스는 작업 수준, 작업자 유형 및 수준에서 작업을 실행하는 동안 개별 작업자로부터 데이터 포인트를 수집합니다. capacity-allocation-type ApplicationId차원으로 사용하여 동일한 애플리케이션에 속하는 여러 작업을 모니터링할 수 있습니다.

EMR서버리스 작업 작업자 수준 지표

지표	설명	단위	기본 측정기준	보조 측정기준
WorkerCpu Allocated	작업 실행 시 작업자에게 할당된	None	JobId	Applicati onId ,

지표	설명	단위	기본 측정기준	보조 측정기준
	총 v CPU 코어 수.			WorkerType 및 CapacityAllocationType
WorkerCpuUsed	한 작업 실행에서 작업자가 사용한 CPU v코어의 총 개수	None	JobId	ApplicationId , WorkerType 및 CapacityAllocationType
WorkerMemoryAllocated	작업 실행 시 작업자에게 할당된 총 메모리 (GB)	기가바이트 (GB)	JobId	ApplicationId , WorkerType 및 CapacityAllocationType
WorkerMemoryUsed	작업 실행 시 작업자가 사용한 총 메모리 (GB)	기가바이트 (GB)	JobId	ApplicationId , WorkerType 및 CapacityAllocationType
WorkerEphemeralStorageAllocated	작업 실행 중인 작업자에게 할당된 임시 스토리지의 바이트 수입니다.	기가바이트 (GB)	JobId	ApplicationId , WorkerType 및 CapacityAllocationType

지표	설명	단위	기본 측정기준	보조 측정기준
WorkerEphemeralStorageUsed	작업 실행 시 작업자가 사용하는 임시 스토리지의 바이트 수입니다.	기가바이트 (GB)	JobId	ApplicationId , WorkerType 및 CapacityAllocationType
WorkerStorageReadBytes	작업 실행 시 작업자가 스토리지에서 읽은 바이트 수입니다.	바이트	JobId	ApplicationId , WorkerType 및 CapacityAllocationType
WorkerStorageWriteBytes	작업 실행 시 작업자가 스토리지에 쓴 바이트 수입니다.	바이트	JobId	ApplicationId , WorkerType 및 CapacityAllocationType

아래 단계에서는 다양한 유형의 지표를 보는 방법을 설명합니다.

Console

콘솔을 사용하여 애플리케이션 UI에 액세스하려면

1. 콘솔에서 시작하기의 지침에 따라 EMR Studio의 EMR 서버리스 애플리케이션으로 이동합니다.
2. 실행 중인 작업에 대한 엔진별 애플리케이션 UIs 및 로그를 보려면:
   a. 상태가 있는 작업을 선택합니다. RUNNING
   b. 응용 프로그램 세부 정보 페이지에서 작업을 선택하거나 작업에 대한 작업 세부 정보 페이지로 이동합니다.

c. 디스플레이 UI 드롭다운 메뉴에서 Spark UI 또는 Hive Tez UI를 선택하여 작업 유형에 맞는 애플리케이션 UI로 이동합니다.

d. Spark 엔진 로그를 보려면 Spark UI의 실행자 탭으로 이동한 다음 드라이버의 로그 링크를 선택합니다. Hive 엔진 로그를 보려면 Hive Tez DAG UI에서 해당하는 로그 링크를 선택하십시오.

3. 엔진별 UIs 애플리케이션과 완료된 작업에 대한 로그를 보려면:

a. 상태가 있는 작업을 선택합니다. SUCCESS

b. 애플리케이션의 애플리케이션 세부 정보 페이지에서 작업을 선택하거나 해당 작업의 Job details 페이지로 이동합니다.

c. 디스플레이 UI 드롭다운 메뉴에서 Spark History Server 또는 Persistent Hive Tez UI를 선택하여 작업 유형에 맞는 애플리케이션 UI로 이동합니다.

d. Spark 엔진 로그를 보려면 Spark UI의 실행자 탭으로 이동한 다음 드라이버의 로그 링크를 선택합니다. Hive 엔진 로그를 보려면 Hive Tez DAG UI에서 해당하는 로그 링크를 선택하십시오.

AWS CLI

를 사용하여 애플리케이션 UI에 액세스하려면 AWS CLI

- 실행 중인 작업과 완료된 작업 모두에 대해 애플리케이션 UI에 액세스하는 데 사용할 수 URL 있는 파일을 생성하려면 를 호출하십시오 `GetDashboardForJobRunAPI`.

```
aws emr-serverless get-dashboard-for-job-run /
--application-id <application-id> /
--job-run-id <job-id>
```

URL생성한 내용은 1시간 동안 유효합니다.

Prometheus용 Amazon 매니지드 서비스를 사용하여 Spark 지표를 모니터링하세요

Amazon EMR 릴리스 7.1.0 이상에서는 서버리스를 Prometheus용 Amazon Managed Service와 통합하여 EMR 서버리스 작업 및 애플리케이션에 대한 Apache Spark 지표를 수집할 수 있습니다. EMR 이 통합은 작업을 제출하거나 다음 중 하나를 사용하여 애플리케이션을 생성할 때 사용할 수 있습니다. AWS 콘솔, EMR 서버리스 API 또는 AWS CLI.

사전 조건

Prometheus용 Amazon 관리형 서비스에 Spark 지표를 전달하려면 먼저 다음 사전 요구 사항을 완료해야 합니다.

- Prometheus용 아마존 매니지드 서비스 워크스페이스를 생성하십시오. 이 Workspace는 수집 엔드포인트 역할을 합니다. 엔드포인트 - 원격 쓰기에 URL 표시된 내용을 기록해 두십시오. URL EMR서버리스 애플리케이션을 생성할 URL 때 를 지정해야 합니다.
- 모니터링 목적으로 Amazon Managed Service for Prometheus에 작업 액세스 권한을 부여하려면 작업 실행 역할에 다음 정책을 추가하십시오.

```
{
    "Sid": "AccessToPrometheus",
    "Effect": "Allow",
    "Action": ["aps:RemoteWrite"],
    "Resource": "arn:aws:aps:<AWS_REGION>:<AWS_ACCOUNT_ID>:workspace/<WORKSPACE_ID>"
}
```

설정

사용하려면 AWS Prometheus용 Amazon 매니지드 서비스와 통합되는 애플리케이션을 생성하기 위한 콘솔

1. 애플리케이션을 생성하려면 Amazon EMR Serverless 시작하기를 참조하십시오.
2. 애플리케이션을 생성하는 동안 사용자 지정 설정 사용을 선택한 다음 구성하려는 필드에 정보를 지정하여 애플리케이션을 구성하십시오.
3. 애플리케이션 로그 및 지표에서 Prometheus용 Amazon Managed Service에 엔진 지표 전달을 선택한 다음 원격 쓰기를 지정합니다. URL
4. 원하는 기타 구성 설정을 지정한 다음 애플리케이션 생성 및 시작을 선택합니다.

를 사용하십시오. AWS CLI 또는 EMR 서버리스 API

다음을 사용할 수도 있습니다. AWS CLI 또는 EMR 서버리스를 API 사용하여 또는 명령을 실행할 때 EMR 서버리스 애플리케이션을 Prometheus용 Amazon Managed Service와 통합할 수 있습니다. `create-application` `start-job-run`

create-application

```
aws emr-serverless create-application \
--release-label emr-7.1.0 \
--type "SPARK" \
--monitoring-configuration '{
    "prometheusMonitoringConfiguration": {
        "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
    }
}'
```

start-job-run

```
aws emr-serverless start-job-run \
--application-id <APPPLICATION_ID> \
--execution-role-arn <JOB_EXECUTION_ROLE> \
--job-driver '{
    "sparkSubmit": {
        "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
        "entryPointArguments": ["10000"],
        "sparkSubmitParameters": "--conf spark.dynamicAllocation.maxExecutors=10"
    }
}' \
--configuration-overrides '{
     "monitoringConfiguration": {
        "prometheusMonitoringConfiguration": {
            "remoteWriteUrl": "https://aps-workspaces.<AWS_REGION>.amazonaws.com/
workspaces/<WORKSPACE_ID>/api/v1/remote_write"
        }
    }
}'
```

명령에 `prometheusMonitoringConfiguration` 포함하면 EMR 서버리스는 Spark 지표를 수집하여 `remoteWriteUrl` Prometheus용 Amazon Managed Service의 엔드포인트에 쓰는 에이전트와 함께 Spark 작업을 실행해야 합니다. 그런 다음 Prometheus용 Amazon Managed Service에서 Spark 지표를 사용하여 시각화, 알림 및 분석을 수행할 수 있습니다.

고급 구성 속성

EMR서버리스는 Spark 내의 구성 요소를 사용하여 Spark 지표를 수집하고 성능 데이터를 Prometheus용 Amazon Managed Service와 호환되는 데이터로 변환합니다. `PrometheusServlet` 기본적으로 EMR 서버리스는 Spark에서 기본값을 설정하고 를 사용하여 작업을 제출할 때 드라이버 및 실행자 지표를 구문 분석합니다. `PrometheusMonitoringConfiguration`

다음 표에는 Prometheus용 Amazon Managed Service로 지표를 보내는 Spark 작업을 제출할 때 구성할 수 있는 모든 속성이 설명되어 있습니다.

스파크 속성	기본값	설명
`spark.metrics.conf.*.sink.prometheusServlet.class`	org.apache.spark.metrics.sink.PrometheusServlet	Spark가 Prometheus용 아마존 매니지드 서비스에 지표를 전송하는 데 사용하는 클래스입니다. 기본 동작을 재정의하려면 사용자 지정 클래스를 지정하십시오.
`spark.metrics.conf.*.source.jvm.class`	org.apache.spark.metrics.source. JvmSource	Spark는 기본 Java 가상 머신에서 중요한 메트릭을 수집하고 전송하는 데 사용하는 클래스입니다. JVM메트릭 수집을 중지하려면 이 속성을 빈 문자열 (예:) 로 설정하여 이 속성을 비활성화합니다. "" 기본 동작을 재정의하려면 사용자 지정 클래스를 지정하십시오.
`spark.metrics.conf.driver.sink.prometheusServlet.path`	/metrics/프로메테우스	Prometheus용 Amazon 관리 서비스가 드라이버로부터 메트릭을 수집하는 데 사용하는 URL 구분입니다. 기본 동작을 재정의하려면 경로를 직접 지정하십시오. 드라이버 메트릭 수집을 중지하려면 이 속성을 빈 문자열 (예:) 로 설정하여 이 속성을 비활성화하십시오. ""

스파크 속성	기본값	설명
`spark.metrics.conf.executor.sink.prometheusServlet.path`	/metrics/Executor/프로메테우스	Prometheus용 Amazon 관리 서비스가 실행자로부터 메트릭을 수집하는 데 사용하는 URL 구분입니다. 기본 동작을 재정의하려면 경로를 직접 지정하십시오. 실행자 지표 수집을 중지하려면 이 속성을 빈 문자열(예:) 로 설정하여 이 속성을 비활성화하십시오. ""

Spark 메트릭에 대한 자세한 내용은 Apache Spark 메트릭을 참조하십시오.

고려 사항 및 제한

Prometheus용 Amazon Managed Service를 사용하여 EMR 서버리스에서 지표를 수집할 때는 다음 고려 사항 및 제한 사항을 고려하십시오.

- 서버리스와 EMR 함께 Prometheus용 Amazon 관리 서비스를 사용하기 위한 지원은 다음 지역에서만 제공됩니다. AWS 리전 여기서 Prometheus용 Amazon 매니지드 서비스를 일반적으로 이용할 수 있습니다.
- Prometheus용 Amazon Managed Service에서 Spark 지표를 수집하도록 에이전트를 실행하려면 작업자로부터 더 많은 리소스가 필요합니다. v 워커 1명과 같이 더 작은 워커 크기를 선택하면 작업 실행 CPU 시간이 늘어날 수 있습니다.
- 서버리스와 EMR 함께 Prometheus용 Amazon Managed Service를 사용하기 위한 EMR 지원은 Amazon 릴리스 7.1.0 이상에서만 사용할 수 있습니다.

EMR서버리스 사용 지표

Amazon CloudWatch 사용 지표를 사용하여 계정에서 사용하는 리소스를 파악할 수 있습니다. 이러한 지표를 사용하여 CloudWatch 그래프와 대시보드에서 서비스 사용량을 시각화할 수 있습니다.

EMR서버리스 사용량 지표는 Service Quotas에 해당합니다. 사용량이 서비스 할당량에 가까워지면 경고하는 경보를 구성할 수 있습니다. 자세한 내용은 Service Quotas 사용 설명서의 서비스 할당량 및 CloudWatch Amazon 경보를 참조하십시오.

EMR서버리스 서비스 할당량에 대한 자세한 내용은 을 참조하십시오. 엔드포인트 및 할당량 EMR Serverless

서버리스의 서비스 할당량 사용량 지표 EMR

EMR서버리스는 네임스페이스에 다음과 같은 서비스 할당량 사용 지표를 게시합니다. AWS/Usage

지표	설명
ResourceCount	계정에서 실행 중인 지정된 리소스의 총 수입니다. 리소스는 지표와 관련된 차원으로 정의됩니다.

EMR서버리스 서비스 할당량 사용량 지표의 크기

다음 측정기준을 사용하여 EMR 서버리스가 게시하는 사용량 지표를 구체화할 수 있습니다.

측정기준	값	설명
Service	EMR서버리스	의 이름 AWS 서비스 여기에는 리소스가 들어 있습니다.
Type	Resource	EMR서버리스가 보고하는 엔티티의 유형.
Resource	v. CPU	EMR서버리스가 추적하는 리소스 유형.
Class	None	EMR서버리스가 추적하는 리소스 클래스입니다.

다음을 통한 서버리스 자동화 EMR Amazon EventBridge

다음을 사용할 수 있음: Amazon EventBridge 자동화하기 위해 AWS 서비스 애플리케이션 가용성 문제 또는 리소스 변경과 같은 시스템 이벤트에 자동으로 대응합니다. EventBridge 변경 사항을 설명하는 시스템 이벤트의 스트림을 거의 실시간으로 제공합니다. AWS 있습니다. 원하는 이벤트만 표시하도

록 간단한 규칙을 작성한 후 규칙과 일치하는 이벤트 발생 시 실행할 자동화 작업을 지정할 수 있습니다. 를 사용하면 EventBridge 다음을 자동으로 수행할 수 있습니다.

- 를 호출하십시오. AWS Lambda 함수
- Amazon Kinesis Data Streams에 이벤트 중계
- 활성화 AWS Step Functions 상태 시스템
- 아마존 SNS 주제 또는 아마존 SQS 대기열에 알림

예를 들어, EventBridge EMR 서버리스와 함께 사용하는 경우 다음을 활성화할 수 있습니다. AWS Lambda ETL작업이 성공하면 작동하고 ETL 작업이 실패하면 Amazon SNS 주제에 알립니다.

EMR서버리스는 세 가지 종류의 이벤트를 내보냅니다.

- 애플리케이션 상태 변경 이벤트 — 애플리케이션의 모든 상태 변경을 발생시키는 이벤트입니다. 애플리케이션 상태에 대한 자세한 내용은 을 참조하십시오애플리케이션 상태.
- 작업 실행 상태 변경 이벤트 - 작업 실행의 모든 상태 변경을 발생시키는 이벤트입니다. 자세한 정보는 작업 실행 상태 섹션을 참조하세요.
- 작업 실행 재시도 이벤트 — Amazon EMR Serverless 릴리스 7.1.0 이상에서 작업 실행을 재시도할 때마다 발생하는 이벤트입니다.

샘플 서버리스 이벤트 EMR EventBridge

다음 예와 같이 EMR 서버리스에서 보고한 이벤트의 값은 `source` to로 `aws.emr-serverless` 지정됩니다.

애플리케이션 상태 변경 이벤트

다음 예제 이벤트는 `CREATING` 상태에 있는 애플리케이션을 보여줍니다.

```
{
    "version": "0",
    "id": "9fd3cf79-1ff1-b633-4dd9-34508dc1e660",
    "detail-type": "EMR Serverless Application State Change",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:16:31Z",
    "region": "us-east-1",
    "resources": [],
```

```
    "detail": {
        "applicationId": "00f1cbsc6anuij25",
        "applicationName": "3965ad00-8fba-4932-a6c8-ded32786fd42",
        "arn": "arn:aws:emr-serverless:us-east-1:111122223333:/
applications/00f1cbsc6anuij25",
        "releaseLabel": "emr-6.6.0",
        "state": "CREATING",
        "type": "HIVE",
        "createdAt": "2022-05-31T21:16:31.547953Z",
        "updatedAt": "2022-05-31T21:16:31.547970Z",
        "autoStopConfig": {
            "enabled": true,
            "idleTimeout": 15
        },
        "autoStartConfig": {
            "enabled": true
        }
    }
}
```

Job 실행 상태 변경 이벤트

다음 예제 이벤트는 SCHEDULED 상태에서 상태로 이동하는 작업 실행을 보여줍니다. RUNNING

```
{
    "version": "0",
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Run State Change",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "jobRunId": "00f1cbn5g4bb0c01",
        "applicationId": "00f1982r1uukb925",
        "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01",
        "releaseLabel": "emr-6.6.0",
        "state": "RUNNING",
        "previousState": "SCHEDULED",
        "createdBy": "arn:aws:sts::123456789012:assumed-role/
TestRole-402dcef3ad14993c15d28263f64381e4cda34775/6622b6233b6d42f59c25dd2637346242",
        "updatedAt": "2022-05-31T21:07:42.299487Z",
```

```
        "createdAt": "2022-05-31T21:07:25.325900Z"
    }
}
```

Job run 재시도 이벤트

다음은 작업 실행 재시도 이벤트의 예입니다.

```
{
    "version": "0",
    "id": "00df3ec6-5da1-36e6-ab71-20f0de68f8a0",
    "detail-type": "EMR Serverless Job Run Retry",
    "source": "aws.emr-serverless",
    "account": "123456789012",
    "time": "2022-05-31T21:07:42Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "jobRunId": "00f1cbn5g4bb0c01",
        "applicationId": "00f1982r1uukb925",
        "arn": "arn:aws:emr-serverless:us-east-1:123456789012:/
applications/00f1982r1uukb925/jobruns/00f1cbn5g4bb0c01",
        "releaseLabel": "emr-6.6.0",
        "createdBy": "arn:aws:sts::123456789012:assumed-role/
TestRole-402dcef3ad14993c15d28263f64381e4cda34775/6622b6233b6d42f59c25dd2637346242",
        "updatedAt": "2022-05-31T21:07:42.299487Z",
        "createdAt": "2022-05-31T21:07:25.325900Z",
        //Attempt Details
        "previousAttempt": 1,
        "previousAttemptState": "FAILED",
        "previousAttemptCreatedAt": "2022-05-31T21:07:25.325900Z",
        "previousAttemptEndedAt": "2022-05-31T21:07:30.325900Z",
        "newAttempt": 2,
        "newAttemptCreatedAt": "2022-05-31T21:07:30.325900Z"
    }
}
```

리소스에 태그 지정

태그를 사용하여 각 리소스에 자체 메타데이터를 할당하면 EMR 서버리스 리소스를 관리하는 데 도움이 됩니다. 이 섹션에서는 태그 함수의 개요를 제공하고 태그를 생성하는 방법을 보여줍니다.

주제

태그란 무엇입니까?

태그는 사용자가 할당하는 레이블입니다. AWS 리소스. 각 태그는 사용자가 정의하는 키와 값으로 구성됩니다. 태그를 사용하면 다음을 분류할 수 있습니다. AWS 리소스 (예: 목적, 소유자, 환경) 별로 동일한 유형의 리소스가 많은 경우 할당한 태그에 따라 특정 리소스를 빠르게 식별할 수 있습니다. 예를 들어 Amazon EMR Serverless 애플리케이션에 대한 태그 세트를 정의하여 각 애플리케이션의 소유자 및 스택 수준을 추적할 수 있습니다. 각 리소스 유형에 대해 일관된 태그 키 집합을 고안하는 것이 좋습니다.

태그가 리소스에 자동으로 할당되는 것은 아닙니다. 리소스에 태그를 추가한 후 언제든지 태그의 값을 수정하거나 리소스에서 태그를 제거할 수 있습니다. 태그는 Amazon EMR Serverless에서 의미론적 의미가 없으며 엄격하게 문자열로 해석됩니다. 해당 리소스에 대해 키가 기존 태그와 동일한 태그를 추가하는 경우 새 값이 이전 값을 덮어씁니다.

를 사용하면 소속된 IAM 사용자를 제어할 수 있습니다. AWS 계정에는 태그를 관리할 권한이 있습니다. 태그 기반 액세스 제어 정책 예제는 태그 기반 액세스 제어를 위한 정책 섹션을 참조하세요.

리소스에 태그 지정

신규 또는 기존 애플리케이션 및 작업 실행에 태그를 지정할 수 있습니다. Amazon EMR API 서버리스를 사용하는 경우 AWS CLI또는 AWS SDK관련 API 작업의 `tags` 파라미터를 사용하여 새 리소스에 태그를 적용할 수 있습니다. `TagResource`API작업을 사용하여 기존 리소스에 태그를 적용할 수 있습니다.

일부 리소스 생성 작업을 사용해서 리소스 생성 시 리소스의 태그를 지정할 수 있습니다. 이 경우 리소스를 생성하는 동안 태그를 적용할 수 없는 경우 리소스를 생성하지 못합니다. 이 매커니즘에서는 생성

중에 태그를 지정하려는 리소스가 지정된 태그와 함께 생성되거나 전혀 생성되지 않습니다. 생성 시 리소스에 태그를 지정하면 리소스를 만든 후 사용자 지정 태깅 스크립트를 실행할 필요가 없습니다.

다음 표에는 태그를 지정할 수 있는 Amazon EMR 서버리스 리소스가 설명되어 있습니다.

Resource	태그 지원	태그 전달 지원	생성 시 태깅 지원 (Amazon EMR 서버리스 API, AWS CLI, 및 AWS SDK)	API생성용 (생성 중에 태그 추가 가능)
애플리케이션	예	아니요. 애플리케이션과 관련된 태그는 해당 애플리케이션에 제출된 작업 실행에 전파되지 않습니다.	예	`CreateApplication`
작업 실행	예	아니요	예	`StartJobRun`

태깅 제한

태그에는 다음과 같은 기본 제한이 적용됩니다.

- 각 리소스는 최대 50개의 사용자 생성 태그를 가질 수 있습니다.
- 각 리소스에 대해 각 태그 키는 고유하며 하나의 값만 가질 수 있습니다.
- 최대 키 길이는 유니코드 문자 128자 (-8) 입니다. UTF
- 최대 값 길이는 -8의 유니코드 문자 256자입니다. UTF
- 허용되는 문자는 UTF -8로 표현할 수 있는 문자, 숫자, 공백 및 _.:/= + - @ 문자입니다.
- 태그 키는 빈 문자열일 수 없습니다. 태그 값에 빈 문자열은 지정할 수 있지만, null은 지정할 수 없습니다.
- 태그 키와 값은 대소문자를 구분합니다.
- `AWS:`키나 값에 접두사와 같은 대문자 또는 소문자 조합을 사용하지 마십시오. 다음 용도로만 사용할 수 있습니다. AWS 사용.

를 사용하여 태그 작업하기 AWS CLI 그리고 아마존 EMR 서버리스 API

다음을 사용하세요. AWS CLI 리소스의 태그를 추가, 업데이트, 나열 및 삭제하기 위한 명령 또는 Amazon EMR Serverless API 작업

Resource	태그 지원	태그 전달 지원
하나 이상의 태그를 추가하거나 덮어씁니다.	`tag-resource`	`TagResource`
리소스에 대한 태그를 나열합니다.	`list-tags-for-resource`	`ListTagsForResource`
하나 이상의 태그를 삭제합니다.	`untag-resource`	`UntagResource`

다음 예는 를 사용하여 리소스에 태그를 지정하거나 태그 해제하는 방법을 보여줍니다. AWS CLI.

기존 애플리케이션에 태그를 지정합니다.

다음 명령은 기존 애플리케이션에 태그를 지정합니다.

```
aws emr-serverless tag-resource --resource-arn resource_ARN --tags team=devs
```

기존 애플리케이션의 태그를 해제합니다.

다음 명령은 기존 애플리케이션에서 태그를 삭제합니다.

```
aws emr-serverless untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

리소스의 태그 목록

다음 명령은 기존 리소스와 연결된 태그를 나열합니다.

```
aws emr-serverless list-tags-for-resource --resource-arn resource_ARN
```

서버리스 튜토리얼 EMR

이 섹션에서는 EMR 서버리스 애플리케이션을 사용할 때의 일반적인 사용 사례를 설명합니다.

주제

아마존 EMR 서버리스에서 자바 17 사용하기

Amazon EMR 릴리스 6.11.0 이상에서는 Java 가상 머신 () 용 Java 17 런타임을 사용하도록 EMR 서버리스 스파크 작업을 구성할 수 있습니다. JVM 다음 방법 중 하나를 사용하여 Spark를 Java 17로 구성하십시오.

`JAVA_HOME`

EMR서버리스 6.11.0 이상의 JVM 설정을 재정의하려면 해당 설정 및 환경 분류에 `JAVA_HOME` 설정을 제공할 수 있습니다. `spark.emr-serverless.driverEnv` `spark.executorEnv`

x86_64

필수 속성을 설정하여 Java 17을 Spark 드라이버 및 실행기의 `JAVA_HOME` 구성으로 지정하십시오.

```
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-
corretto.x86_64/
```

```
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.x86_64/
```

arm_64

필수 속성을 설정하여 Java 17을 Spark 드라이버 및 JAVA_HOME 실행기의 구성으로 지정하십시오.

```
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-
corretto.aarch64/
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-17-amazon-corretto.aarch64/
```

spark-defaults

또는 spark-defaults 분류에 Java 17을 지정하여 EMR 서버리스 6.11.0 이상에 대한 JVM 설정을 재정의할 수 있습니다.

x86_64

분류에 Java 17을 지정하십시오. spark-defaults

```
{
"applicationConfiguration": [
         {
            "classification": "spark-defaults",
            "properties": {
               "spark.emr-serverless.driverEnv.JAVA_HOME" : "/usr/lib/jvm/java-17-
amazon-corretto.x86_64/",
               "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-17-amazon-
corretto.x86_64/"
            }
         }
      ]
}
```

arm_64

spark-defaults분류에 Java 17을 지정하십시오.

```
{
"applicationConfiguration": [
         {
```

```
            "classification": "spark-defaults",
            "properties": {
               "spark.emr-serverless.driverEnv.JAVA_HOME" : "/usr/lib/jvm/java-17-
amazon-corretto.aarch64/",
               "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-17-amazon-
corretto.aarch64/"
            }
         }
      ]
}
```

서버리스와 함께 아파치 후디 사용하기 EMR

서버리스 애플리케이션과 함께 Apache Hudi를 사용하려면 EMR

1. 해당 Spark 작업 실행에서 필요한 Spark 속성을 설정합니다.

   ```
   spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar
   spark.serializer=org.apache.spark.serializer.KryoSerializer
   ```

2. Hudi 테이블을 구성된 카탈로그에 동기화하려면 다음 중 하나를 지정하십시오. AWS Glue Data Catalog를 메타스토어로 사용하거나 외부 메타스토어를 구성하십시오. EMR서버리스는 Hudi hms 워크로드용 Hive 테이블의 동기화 모드를 지원합니다. EMR서버리스는 이 속성을 기본적으로 활성화합니다. 메타스토어 설정 방법에 대한 자세한 내용은 을 참조하십시오. 메타스토어 구성

   > **Important**
   >
   > EMR서버리스는 Hive 테이블의 Hudi 워크로드를 처리하기 위한 동기화 모드 JDBC 옵션으로 HIVEQL 또는 기능을 지원하지 않습니다. 자세히 알아보려면 동기화 모드를 참조하십시오.

   를 사용하는 경우 AWS Glue Data Catalog를 메타스토어로 사용하여 Hudi 작업에 대해 다음과 같은 구성 속성을 지정할 수 있습니다.

   ```
   --conf spark.jars=/usr/lib/hudi/hudi-spark-bundle.jar,
   --conf spark.serializer=org.apache.spark.serializer.KryoSerializer,
   --conf
    spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
   ```

EMRAmazon의 Apache Hudi 릴리스에 대한 자세한 내용은 Hudi 릴리스 기록을 참조하십시오.

서버리스와 함께 아파치 아이스버그 사용하기 EMR

서버리스 애플리케이션과 함께 Apache Iceberg를 사용하려면 EMR

1. 해당 Spark 작업 실행에서 필요한 Spark 속성을 설정합니다.

```
spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar
```

2. 다음 중 하나를 지정하십시오. AWS Data Catalog를 메타스토어로 붙이거나 외부 메타스토어를 구성합니다. 메타스토어 설정에 대한 자세한 내용은 을 참조하십시오. 메타스토어 구성

   Iceberg에 사용할 메타스토어 속성을 구성하십시오. 예를 들어, 다음을 사용하려는 경우 AWS Glue Data Catalog를 사용하여 애플리케이션 구성에서 다음 속성을 설정합니다.

```
spark.sql.catalog.dev.warehouse=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog
spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog
spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGl
```

   를 사용하는 경우 AWS Glue Data Catalog를 메타스토어로 사용하여 Iceberg 작업에 대해 다음과 같은 구성 속성을 지정할 수 있습니다.

```
--conf spark.jars=/usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar,
--conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions,
--conf spark.sql.catalog.dev=org.apache.iceberg.spark.SparkCatalog,
--conf spark.sql.catalog.dev.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog,
--conf spark.sql.catalog.dev.warehouse=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
--conf
 spark.hadoop.hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSG
```

EMRAmazon의 Apache Iceberg 릴리스에 대한 자세한 내용은 Iceberg 릴리스 기록을 참조하십시오.

EMR서버리스에서 Python 라이브러리 사용하기

Amazon EMR Serverless 애플리케이션에서 PySpark 작업을 실행할 때 다양한 Python 라이브러리를 종속 항목으로 패키징할 수 있습니다. 이를 위해 기본 Python 기능을 사용하거나, 가상 환경을 구축하거나, Python 라이브러리를 사용하도록 PySpark 작업을 직접 구성할 수 있습니다. 이 페이지에서는 각 접근 방식을 다룹니다.

네이티브 Python 기능 사용

다음 구성을 설정하면 Python 파일 (`.py`), 압축된 Python 패키지 () 및 Egg 파일 (`.zip`) 을 Spark 실행기에 업로드하는 데 사용할 PySpark 수 있습니다. `.egg`

```
--conf spark.submit.pyFiles=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/<.py|.egg|.zip file>
```

Python 가상 환경을 PySpark 작업에 사용하는 방법에 대한 자세한 내용은 PySpark 네이티브 기능 사용을 참조하십시오.

Python 가상 환경 구축하기

PySpark 작업에 사용할 여러 Python 라이브러리를 패키징하려면 격리된 Python 가상 환경을 만들 수 있습니다.

1. Python 가상 환경을 구축하려면 다음 명령을 사용합니다. 표시된 예제는 패키지를 `scipy` 가상 환경 패키지에 설치하고 아카이브를 Amazon S3 위치에 복사합니다. `matplotlib`

   > **Important**
   >
   > EMR서버리스에서 사용하는 것과 동일한 버전의 Python을 사용하는 유사한 Amazon Linux 2 환경, 즉 Amazon 릴리스 6.6.0용 Python 3.7.10에서 다음 명령을 실행해야 합니다. EMR 서버리스 샘플 리포지토리에서 예제 Dockerfile을 찾을 수 있습니다. EMR GitHub

```
# initialize a python virtual environment
python3 -m venv pyspark_venvsource
source pyspark_venvsource/bin/activate

# optionally, ensure pip is up-to-date
```

```
pip3 install --upgrade pip

# install the python packages
pip3 install scipy
pip3 install matplotlib

# package the virtual environment into an archive
pip3 install venv-pack
venv-pack -f -o pyspark_venv.tar.gz

# copy the archive to an S3 location
aws s3 cp pyspark_venv.tar.gz s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/

# optionally, remove the virtual environment directory
rm -fr pyspark_venvsource
```

2. Python 가상 환경을 사용하려면 속성을 설정한 상태로 Spark 작업을 제출하세요.

```
--conf spark.archives=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
pyspark_venv.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

원본 Python 바이너리를 재정의하지 않으면 이전 설정 순서의 두 번째 구성이 다음과 같다는 점에 유의하십시오. `--conf spark.executorEnv.PYSPARK_PYTHON=python`

Python 가상 환경을 PySpark 작업에 사용하는 방법에 대한 자세한 내용은 Virtualenv 사용을 참조하십시오. Spark 작업을 제출하는 방법에 대한 추가 예제는 을 참조하십시오. 스파크 잡스

Python 라이브러리를 사용하도록 PySpark 작업 구성

Amazon EMR 릴리스 6.12.0 이상에서는 추가 설정 없이 Pandas와 같은 인기 있는 데이터 과학 Python 라이브러리를 사용하도록 EMR 서버리스 PySpark 작업을 직접 구성할 수 있습니다. NumPyPyArrow

다음 예제는 PySpark 작업을 위해 각 Python 라이브러리를 패키징하는 방법을 보여줍니다.

NumPy

NumPy 수학, 정렬, 랜덤 시뮬레이션 및 기본 통계를 위한 다차원 배열 및 연산을 제공하는 과학 컴퓨팅을 위한 Python 라이브러리입니다. 사용하려면 NumPy 다음 명령을 실행하세요.

```
import numpy
```

pandas

pandas는 위에 구축된 Python 라이브러리입니다. NumPy 판다스 라이브러리는 데이터 과학자에게 데이터 구조 및 DataFrame데이터 분석 도구를 제공합니다. 판다를 사용하려면 다음 명령어를 실행하세요.

```
import pandas
```

PyArrow

PyArrow 작업 성능 향상을 위해 메모리 내 열 데이터를 관리하는 Python 라이브러리입니다. PyArrow 데이터를 열 형식으로 표현하고 교환하는 표준 방법인 Apache Arrow 언어 간 개발 사양을 기반으로 합니다. 사용하려면 PyArrow 다음 명령을 실행합니다.

```
import pyarrow
```

EMR서버리스에서 다양한 Python 버전 사용하기

의 사용 사례 외에도 EMR서버리스에서 Python 라이브러리 사용하기 Python 가상 환경을 사용하여 Amazon EMR 서버리스 애플리케이션용 Amazon EMR 릴리스에 패키징된 버전과 다른 Python 버전을 사용할 수 있습니다. 이를 위해서는 사용하려는 Python 버전으로 Python 가상 환경을 구축해야 합니다.

Python 가상 환경에서 작업을 제출하려면

1. 다음 예제의 명령을 사용하여 가상 환경을 구축하십시오. 이 예제는 Python 3.9.9를 가상 환경 패키지에 설치하고 아카이브를 Amazon S3 위치에 복사합니다.

   > **Important**
   >
   > Amazon EMR 릴리스 7.0.0 이상을 사용하는 경우 EMR 서버리스 애플리케이션에 사용하는 것과 유사한 Amazon Linux 2023 환경에서 명령을 실행해야 합니다.

릴리스 6.15.0 이하를 사용하는 경우 유사한 Amazon Linux 2 환경에서 다음 명령을 실행해야 합니다.

```
# install Python 3.9.9 and activate the venv
yum install -y gcc openssl-devel bzip2-devel libffi-devel tar gzip wget make
wget https://www.python.org/ftp/python/3.9.9/Python-3.9.9.tgz && \
tar xzf Python-3.9.9.tgz && cd Python-3.9.9 && \
./configure --enable-optimizations && \
make altinstall

# create python venv with Python 3.9.9
python3.9 -m venv pyspark_venv_python_3.9.9 --copies
source pyspark_venv_python_3.9.9/bin/activate

# copy system python3 libraries to venv
cp -r /usr/local/lib/python3.9/* ./pyspark_venv_python_3.9.9/lib/python3.9/

# package venv to archive.
# **Note** that you have to supply --python-prefix option
# to make sure python starts with the path where your
# copied libraries are present.
# Copying the python binary to the "environment" directory.
pip3 install venv-pack
venv-pack -f -o pyspark_venv_python_3.9.9.tar.gz --python-prefix /home/hadoop/
environment

# stage the archive in S3
aws s3 cp pyspark_venv_python_3.9.9.tar.gz s3://<path>

# optionally, remove the virtual environment directory
rm -fr pyspark_venv_python_3.9.9
```

2. Python 가상 환경을 사용하도록 속성을 설정하고 Spark 작업을 제출합니다.

```
# note that the archive suffix "environment" is the same as the directory where you
 copied the Python binary.
--conf spark.archives=s3://DOC-EXAMPLE-BUCKET/EXAMPLE-PREFIX/
pyspark_venv_python_3.9.9.tar.gz#environment
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/
python
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
```

```
--conf spark.executorEnv.PYSPARK_PYTHON=./environment/bin/python
```

Python 가상 환경을 PySpark 작업에 사용하는 방법에 대한 자세한 내용은 Virtualenv 사용을 참조하십시오. Spark 작업을 제출하는 방법에 대한 추가 예제는 을 참조하십시오. 스파크 잡스

델타 OSS 레이크와 EMR 서버리스 사용

아마존 EMR 버전 6.9.0 이상

> **Note**
>
> Amazon EMR 7.0.0 이상에서는 파일 이름을 로 바꾸는 델타 레이크 3.0.0을 사용합니다. `delta-core.jar` `delta-spark.jar` Amazon EMR 7.0.0 이상을 사용하는 경우 `delta-spark.jar` 구성에서 지정해야 합니다.

Amazon EMR 6.9.0 이상에는 Delta Lake가 포함되므로 더 이상 Delta Lake를 직접 패키징하거나 EMR 서버리스 작업에 `--packages` 플래그를 제공할 필요가 없습니다.

1. EMR서버리스 작업을 제출할 때는 다음 구성 속성이 있는지 확인하고 필드에 다음 매개 변수를 포함하십시오. `sparkSubmitParameters`

```
--conf spark.jars=/usr/share/aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/
delta-storage.jar
    --conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension
    --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog
```

2. 로컬을 `delta_sample.py` 생성하여 델타 테이블 생성 및 읽기를 테스트하십시오.

```
# delta_sample.py
    from pyspark.sql import SparkSession

    import uuid

    url = "s3://DOC-EXAMPLE-BUCKET/delta-lake/output/%s/" % str(uuid.uuid4())
    spark = SparkSession.builder.appName("DeltaSample").getOrCreate()

    ## creates a Delta table and outputs to target S3 bucket
```

```
    spark.range(5).write.format("delta").save(url)

    ## reads a Delta table and outputs to target S3 bucket
    spark.read.format("delta").load(url).show
```

3. 사용 AWS CLI, Amazon S3 버킷에 `delta_sample.py` 파일을 업로드합니다. 그런 다음 `start-job-run` 명령을 사용하여 기존 EMR 서버리스 애플리케이션에 작업을 제출합니다.

```
aws s3 cp delta_sample.py s3://DOC-EXAMPLE-BUCKET/code/

    aws emr-serverless start-job-run \
        --application-id application-id \
        --execution-role-arn job-role-arn \
        --name emr-delta \
        --job-driver '{
            "sparkSubmit": {
                "entryPoint": "s3://DOC-EXAMPLE-BUCKET/code/delta_sample.py",
                "sparkSubmitParameters": "--conf spark.jars=/usr/share/
aws/delta/lib/delta-core.jar,/usr/share/aws/delta/lib/delta-storage.jar --
conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog"
            }
        }'
```

Delta Lake에서 Python 라이브러리를 사용하려면 `delta-core` 라이브러리를 종속 항목으로 패키징하거나 사용자 지정 이미지로 사용하여 라이브러리를 추가할 수 있습니다.

또는 를 사용하여 `delta-core` JAR 파일에서 Python 라이브러리를 `SparkContext.addPyFile` 추가할 수 있습니다.

```
import glob
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
spark.sparkContext.addPyFile(glob.glob("/usr/share/aws/delta/lib/delta-core_*.jar")[0])
```

아마존 EMR 버전 6.8.0 이하

Amazon EMR 6.8.0 이하를 사용하는 경우 다음 단계에 따라 EMR 서버리스 OSS 애플리케이션과 함께 Delta Lake를 사용하십시오.

1. Amazon EMR Serverless 애플리케이션의 Spark 버전과 호환되는 Delta Lake의 오픈 소스 버전을 구축하려면 델타로 GitHub 이동하여 지침을 따르십시오.
2. 델타 레이크 라이브러리를 내 Amazon S3 버킷에 업로드합니다. AWS 계정.
3. 애플리케이션 구성에서 EMR 서버리스 작업을 제출할 때는 현재 버킷에 있는 Delta Lake JAR 파일을 포함하십시오.

```
--conf spark.jars=s3://DOC-EXAMPLE-BUCKET/jars/delta-core_2.12-1.1.0.jar
```

4. 델타 테이블에서 읽고 쓸 수 있는지 확인하려면 샘플 PySpark 테스트를 실행하세요.

```
from pyspark import SparkConf, SparkContext
    from pyspark.sql import HiveContext, SparkSession

    import uuid

    conf = SparkConf()
    sc = SparkContext(conf=conf)
    sqlContext = HiveContext(sc)

    url = "s3://DOC-EXAMPLE-BUCKET/delta-lake/output/1.0.1/%s/" % str(uuid.uuid4())

    ## creates a Delta table and outputs to target S3 bucket
    session.range(5).write.format("delta").save(url)

    ## reads a Delta table and outputs to target S3 bucket
    session.read.format("delta").load(url).show
```

에어플로우에서 EMR 서버리스 작업 제출

아파치 에어플로우의 Amazon 공급자는 EMR 서버리스 운영자를 제공합니다. 오퍼레이터에 대한 자세한 내용은 Apache Airflow 설명서에서 Amazon EMR 서버리스 오퍼레이터를 참조하십시오.

를 `EmrServerlessCreateApplicationOperator` 사용하여 Spark 또는 Hive 애플리케이션을 생성할 수 있습니다. 를 사용하여 새 `EmrServerlessStartJobOperator` 애플리케이션으로 하나 이상의 작업을 시작할 수도 있습니다.

Airflow 2.2.2가 설치된 Apache Airflow용 Amazon Managed Workflow (MWAA) 에서 연산자를 사용하려면 `requirements.txt` 파일에 다음 줄을 추가하고 새 파일을 사용하도록 MWAA 환경을 업데이트하십시오.

```
apache-airflow-providers-amazon==6.0.0
boto3>=1.23.9
```

Amazon 공급자의 릴리스 5.0.0에는 EMR 서버리스 지원이 추가되었으니 참고하십시오. 릴리스 6.0.0은 에어플로우 2.2.2와 호환되는 최신 버전입니다. 에어플로우 2.4.3이 켜진 상태에서 이후 버전을 사용할 수 있습니다. MWAA

다음 간략한 예제는 응용 프로그램을 만들고 여러 Spark 작업을 실행한 다음 응용 프로그램을 중지하는 방법을 보여줍니다. 전체 예제는 서버리스 샘플 리포지토리에서 확인할 수 있습니다. EMR GitHub `sparkSubmit`구성에 대한 추가 세부 정보는 을 참조하십시오스파크 잡스.

```
from datetime import datetime

from airflow import DAG
from airflow.providers.amazon.aws.operators.emr import (
    EmrServerlessCreateApplicationOperator,
    EmrServerlessStartJobOperator,
    EmrServerlessDeleteApplicationOperator,
)

# Replace these with your correct values
JOB_ROLE_ARN = "arn:aws:iam::account-id:role/emr_serverless_default_role"
S3_LOGS_BUCKET = "DOC-EXAMPLE-BUCKET"

DEFAULT_MONITORING_CONFIG = {
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {"logUri": f"s3://DOC-EXAMPLE-BUCKET/logs/"}
    },
}

with DAG(
    dag_id="example_endtoend_emr_serverless_job",
    schedule_interval=None,
    start_date=datetime(2021, 1, 1),
    tags=["example"],
    catchup=False,
) as dag:
    create_app = EmrServerlessCreateApplicationOperator(
        task_id="create_spark_app",
        job_type="SPARK",
        release_label="emr-6.7.0",
        config={"name": "airflow-test"},
```

```
    )

    application_id = create_app.output

    job1 = EmrServerlessStartJobOperator(
        task_id="start_job_1",
        application_id=application_id,
        execution_role_arn=JOB_ROLE_ARN,
        job_driver={
            "sparkSubmit": {
                "entryPoint": "local:///usr/lib/spark/examples/src/main/python/
pi_fail.py",
            }
        },
        configuration_overrides=DEFAULT_MONITORING_CONFIG,
    )

    job2 = EmrServerlessStartJobOperator(
        task_id="start_job_2",
        application_id=application_id,
        execution_role_arn=JOB_ROLE_ARN,
        job_driver={
            "sparkSubmit": {
                "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
                "entryPointArguments": ["1000"]
            }
        },
        configuration_overrides=DEFAULT_MONITORING_CONFIG,
    )

    delete_app = EmrServerlessDeleteApplicationOperator(
        task_id="delete_app",
        application_id=application_id,
        trigger_rule="all_done",
    )

    (create_app >> [job1, job2] >> delete_app)
```

서버리스에서 Hive 사용자 정의 함수 사용 EMR

Hive 사용자 정의 함수 (UDFs) 를 사용하면 레코드 또는 레코드 그룹을 처리하는 사용자 지정 함수를 만들 수 있습니다. 이 자습서에서는 기존 Amazon EMR Serverless UDF 애플리케이션과 함께 샘플을

사용하여 쿼리 결과를 출력하는 작업을 실행합니다. 애플리케이션 설정 방법을 알아보려면 을 참조하십시오. Amazon EMR 서버리스 시작하기

EMR서버리스에서 UDF a를 사용하려면

1. 샘플을 GitHubUDF보려면 로 이동하십시오. 리포지토리를 복제하고 사용하려는 git 브랜치로 전환합니다. `maven-compiler-plugin`리포지토리의 `pom.xml` 파일에서 소스를 포함하도록 업데이트하세요. 또한 대상 Java 버전 구성을 로 `1.8` 업데이트하십시오. `mvn package -DskipTests`를 실행하여 샘플이 포함된 JAR 파일을 생성합니다UDFs.
2. JAR파일을 생성한 후 다음 명령을 사용하여 S3 버킷에 업로드합니다.

```
aws s3 cp brickhouse-0.8.2-JS.jar s3://DOC-EXAMPLE-BUCKET/jars/
```

3. 샘플 UDF 함수 중 하나를 사용할 예제 파일을 생성합니다. 이 쿼리를 로 `udf_example.q` 저장하고 S3 버킷에 업로드합니다.

```
add jar s3://DOC-EXAMPLE-BUCKET/jars/brickhouse-0.8.2-JS.jar;
CREATE TEMPORARY FUNCTION from_json AS 'brickhouse.udf.json.FromJsonUDF';
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
 array(cast(0 as int))));
select from_json('{"key1":[0,1,2], "key2":[3,4,5,6], "key3":[7,8,9]}', map("",
 array(cast(0 as int))))["key1"][2];
```

4. 다음 Hive 작업을 제출하십시오.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
    "hive": {
        "query": "s3://DOC-EXAMPLE-BUCKET/queries/udf_example.q",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/scratch --hiveconf hive.metastore.warehouse.dir=s3://'$BUCKET'/emr-
serverless-hive/warehouse"
    }
}' --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "hive-site",
        "properties": {
            "hive.driver.cores": "2",
            "hive.driver.memory": "6G"
        }
```

```
    }],
    "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/logs/"
        }
    }
}'
```

5. get-job-run명령을 사용하여 작업 상태를 확인합니다. 상태가 로 변경될 때까지 기다리십시오SUCCESS.

```
aws emr-serverless get-job-run --application-id application-id --job-run-id job-id
```

6. 다음 명령을 사용하여 출력 파일을 다운로드합니다.

```
aws s3 cp --recursive s3://DOC-EXAMPLE-BUCKET/logs/applications/application-id/
jobs/job-id/HIVE_DRIVER/ .
```

stdout.gz파일은 다음과 비슷합니다.

```
{"key1":[0,1,2],"key2":[3,4,5,6],"key3":[7,8,9]}
2
```

EMR서버리스에서 사용자 지정 이미지 사용

주제

- 사용자 지정 Python 버전 사용
- 사용자 지정 Java 버전을 사용하세요.
- 데이터 사이언스 이미지를 빌드하세요.
- Apache Sedona를 사용한 지리공간 데이터 처리

사용자 지정 Python 버전 사용

다른 버전의 Python을 사용하도록 사용자 지정 이미지를 만들 수 있습니다. 예를 들어 Spark 작업에 Python 버전 3.10을 사용하려면 다음 명령을 실행합니다.

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest
```

```
USER root

# install python 3
RUN yum install -y gcc openssl-devel bzip2-devel libffi-devel tar gzip wget make
RUN wget https://www.python.org/ftp/python/3.10.0/Python-3.10.0.tgz && \
tar xzf Python-3.10.0.tgz && cd Python-3.10.0 && \
./configure --enable-optimizations && \
make altinstall

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

Spark 작업을 제출하기 전에 다음과 같이 Python 가상 환경을 사용하도록 속성을 설정합니다.

```
--conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=/usr/local/bin/python3.10
--conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=/usr/local/bin/python3.10
--conf spark.executorEnv.PYSPARK_PYTHON=/usr/local/bin/python3.10
```

사용자 지정 Java 버전을 사용하세요.

다음 예제는 Spark 작업에 Java 11을 사용하도록 사용자 지정 이미지를 빌드하는 방법을 보여줍니다.

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

# install JDK 11
RUN sudo amazon-linux-extras install java-openjdk11

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

Spark 작업을 제출하기 전에 다음과 같이 Spark 속성을 Java 11을 사용하도록 설정하십시오.

```
--conf spark.executorEnv.JAVA_HOME=/usr/lib/jvm/java-11-
openjdk-11.0.16.0.8-1.amzn2.0.1.x86_64
--conf spark.emr-serverless.driverEnv.JAVA_HOME=/usr/lib/jvm/java-11-
openjdk-11.0.16.0.8-
```

데이터 사이언스 이미지를 빌드하세요.

다음 예제는 Pandas 및 와 같은 일반적인 데이터 과학 Python 패키지를 포함하는 방법을 보여줍니다. NumPy

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

# python packages
RUN pip3 install boto3 pandas numpy
RUN pip3 install -U scikit-learn==0.23.2 scipy
RUN pip3 install sk-dist
RUN pip3 install xgboost

# EMR Serverless will run the image as hadoop
USER hadoop:hadoop
```

Apache Sedona를 사용한 지리공간 데이터 처리

다음 예제는 지리공간 처리를 위해 Apache Sedona를 포함하도록 이미지를 빌드하는 방법을 보여줍니다.

```
FROM public.ecr.aws/emr-serverless/spark/emr-6.9.0:latest

USER root

RUN yum install -y wget
RUN wget https://repo1.maven.org/maven2/org/apache/sedona/sedona-core-3.0_2.12/1.3.0-
incubating/sedona-core-3.0_2.12-1.3.0-incubating.jar -P /usr/lib/spark/jars/
RUN pip3 install apache-sedona

# EMRS will run the image as hadoop
USER hadoop:hadoop
```

아마존 서버리스에서 아파치 스파크에 Amazon Redshift 통합 사용 EMR

Amazon EMR 릴리스 6.9.0 이상에서는 모든 릴리스 이미지에 Apache Spark와 Amazon Redshift 간의 커넥터가 포함됩니다. 이 커넥터를 사용하면 Amazon EMR 서버리스에서 Spark를 사용하여 Amazon

Redshift에 저장된 데이터를 처리할 수 있습니다. 통합은 spark-redshift 오픈 소스 커넥터를 기반으로 합니다. 아마존 EMR 서버리스의 경우, 아파치 스파크에 대한 Amazon Redshift 통합이 기본 통합으로 포함됩니다.

주제

- 아파치 스파크용 Amazon Redshift 통합을 사용하여 Spark 애플리케이션 시작
- Apache Spark용 Amazon Redshift 통합으로 인증
- Amazon Redshift에서 읽고 쓰기
- Spark 커넥터 사용 시 고려 사항 및 제한 사항

아파치 스파크용 Amazon Redshift 통합을 사용하여 Spark 애플리케이션 시작

EMR서버리스 6.9.0과의 통합을 사용하려면 필수 Spark-Redshift 종속성을 Spark 작업에 전달해야 합니다. Redshift 커넥터 관련 라이브러리를 포함하는 `--jars` 데 사용합니다. `--jars` 옵션에서 지원하는 다른 파일 위치를 보려면 Apache Spark 설명서에서 Advanced Dependency Management 섹션을 참조하세요.

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

Amazon EMR 릴리스 6.10.0 이상에서는 `minimal-json.jar` 종속성이 필요하지 않으며 기본적으로 다른 종속성을 각 클러스터에 자동으로 설치합니다. 다음 예제에서는 Apache Spark용 Amazon Redshift 통합을 사용하여 Spark 애플리케이션을 시작하는 방법을 보여줍니다.

Amazon EMR 6.10.0 +

EMR서버리스 릴리스 6.10.0 이상의 Apache Spark에 대한 Amazon Redshift 통합을 사용하여 Amazon EMR 서버리스에서 Spark 작업을 시작하십시오.

```
spark-submit my_script.py
```

Amazon EMR 6.9.0

EMR서버리스 릴리스 6.9.0의 Apache Spark에 대한 Amazon Redshift 통합을 사용하여 Amazon EMR 서버리스에서 Spark 작업을 시작하려면 다음 예제와 같이 옵션을 사용하십시오`--jars`. `--jars`옵션과 함께 나열된 경로가 파일의 기본 경로라는 점에 유의하십시오. JAR

```
--jars
    /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,
    /usr/share/aws/redshift/spark-redshift/lib/spark-redshift.jar,
    /usr/share/aws/redshift/spark-redshift/lib/spark-avro.jar,
    /usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar
```

```
spark-submit \
  --jars /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,/usr/share/aws/redshift/
spark-redshift/lib/spark-redshift.jar,/usr/share/aws/redshift/spark-redshift/lib/
spark-avro.jar,/usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar \
  my_script.py
```

Apache Spark용 Amazon Redshift 통합으로 인증

사용 AWS Secrets Manager 자격 증명을 검색하고 Amazon Redshift에 연결하려면

Secrets Manager에 자격 증명을 저장하여 Amazon Redshift에 안전하게 인증하고 Spark 작업에서 이를 `GetSecretValue` API 호출하여 가져오도록 할 수 있습니다.

```
from pyspark.sql import SQLContextimport boto3

sc = # existing SparkContext
sql_context = SQLContext(sc)

secretsmanager_client = boto3.client('secretsmanager',
 region_name=os.getenv('AWS_REGION'))
secret_manager_response = secretsmanager_client.get_secret_value(
    SecretId='string',
    VersionId='string',
    VersionStage='string'
)
username = # get username from secret_manager_response
password = # get password from secret_manager_response
```

```
url = "jdbc:redshift://redshifthost:5439/database?user=" + username + "&password="
 + password

# Access to Redshift cluster using Spark
```

드라이버를 사용하여 Amazon Redshift에 인증하기 JDBC

내부에 사용자 이름과 비밀번호를 설정합니다. JDBC URL

에서 Amazon Redshift 데이터베이스 이름과 암호를 지정하여 Amazon Redshift 클러스터에 대한 Spark 작업을 인증할 수 있습니다. JDBC URL

> **Note**
>
> 에 데이터베이스 자격 증명을 전달하면 액세스 권한이 있는 모든 사용자도 자격 증명에 액세스할 URL 수 있습니다. URL 이 방법은 안전한 옵션이 아니므로 일반적으로 권장되지 않습니다.

애플리케이션의 보안이 문제가 아닌 경우 다음 형식을 사용하여 사용자 이름과 비밀번호를 설정할 수 있습니다 JDBCURL.

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

Amazon EMR 서버리스 작업 실행 역할을 통한 IAM 기반 인증 사용

Amazon EMR 서버리스 릴리스 6.9.0부터 Amazon Redshift JDBC 드라이버 2.1 이상이 환경에 패키징됩니다. JDBC드라이버 2.1 이상에서는 원시 사용자 이름과 암호를 지정하거나 포함하지 않을 수 있습니다. JDBC URL

대신 `jdbc:redshift:iam://` 스키마를 지정할 수 있습니다. 이렇게 하면 JDBC 드라이버가 EMR 서버리스 작업 실행 역할을 사용하여 자격 증명을 자동으로 가져오도록 명령합니다. 자세한 내용은 Amazon Redshift 관리 안내서의 IAM 자격 증명을 사용하도록 JDBC 또는 ODBC 연결 구성을 참조하십시오. 이에 대한 예는 다음과 URL 같습니다.

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

제공된 조건이 충족될 경우 작업 실행 역할에 필요한 권한은 다음과 같습니다.

권한	작업 실행 역할에 필요한 경우 조건
`redshift:GetClusterCredentials`	JDBC드라이버가 Amazon Redshift에서 자격 증명을 가져오는 데 필요합니다.
`redshift:DescribeCluster`	Amazon Redshift 클러스터를 지정하는 경우 필수이며 AWS 리전 JDBCURL엔드포인트 대신
`redshift-serverless:GetCredentials`	JDBC드라이버가 Amazon Redshift 서버리스에서 자격 증명을 가져오는 데 필요합니다.
`redshift-serverless:GetWorkgroup`	Amazon Redshift 서버리스를 사용하고 있고 작업 그룹 이름 및 지역을 지정하는 경우 필수입니다. URL

다른 곳에서 Amazon Redshift에 연결하기 VPC

VPCa 아래에 프로비저닝된 Amazon Redshift 클러스터 또는 Amazon Redshift 서버리스 워크그룹을 설정하는 경우 Amazon 서버리스 애플리케이션이 리소스에 액세스할 수 있도록 VPC 연결을 구성해야 합니다. EMR 서버리스 애플리케이션에서 VPC 연결을 구성하는 방법에 대한 자세한 내용은 을 참조하십시오. EMR 액세스 구성 VPC

- 프로비저닝된 Amazon Redshift 클러스터 또는 Amazon Redshift 서버리스 워크그룹이 공개적으로 액세스할 수 있는 경우, 서버리스 애플리케이션을 생성할 때 게이트웨이가 연결된 하나 이상의 프라이빗 서브넷을 지정할 수 있습니다. NAT EMR
- 프로비저닝된 Amazon Redshift 클러스터 또는 Amazon Redshift 서버리스 워크그룹에 공개적으로 액세스할 수 없는 경우 에 설명된 대로 Amazon Redshift VPC 클러스터용 Amazon Redshift 관리형 엔드포인트를 생성해야 합니다. 액세스 구성 VPC 또는 Amazon Redshift 관리 안내서의 Amazon Redshift 서버리스에 연결에 설명된 대로 Amazon Redshift 서버리스 워크그룹을 생성할 수 있습니다. 서버리스 애플리케이션을 생성할 때 지정한 프라이빗 서브넷에 클러스터 또는 하위 그룹을 연결해야 합니다. EMR

> **Note**
>
> IAM기반 인증을 사용하고 EMR 서버리스 애플리케이션용 프라이빗 서브넷에 NAT 게이트웨이가 연결되어 있지 않은 경우 Amazon Redshift 또는 Amazon Redshift Serverless의 해당 서

브넷에도 VPC 엔드포인트를 생성해야 합니다. 이렇게 하면 드라이버가 자격 증명을 가져올 수 JDBC 있습니다.

Amazon Redshift에서 읽고 쓰기

다음 코드 예제는 데이터 API 소스와 Spark를 PySpark 사용하여 Amazon Redshift 데이터베이스에서 샘플 데이터를 읽고 쓰는 데 사용합니다. SQL

Data source API

데이터 PySpark 소스를 사용하여 Amazon Redshift 데이터베이스에서 샘플 데이터를 읽고 쓰는 데 사용합니다. API

```
import boto3
from pyspark.sql import SQLContext

sc = # existing SparkContext
sql_context = SQLContext(sc)

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::account-id:role/role-name"

df = sql_context.read \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "table-name") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws-iam-role-arn") \
    .load()

df.write \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "table-name-copy") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws-iam-role-arn") \
    .mode("error") \
    .save()
```

SparkSQL

PySpark Spark를 사용하여 Amazon Redshift 데이터베이스에서 샘플 데이터를 읽고 쓰는 데 사용합니다. SQL

```
import boto3
import json
import sys
import os
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .enableHiveSupport() \
    .getOrCreate()

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::account-id:role/role-name"

bucket = "s3://path/for/temp/data"
tableName = "table-name" # Redshift table name

s = f"""CREATE TABLE IF NOT EXISTS {table-name} (country string, data string)
    USING io.github.spark_redshift_community.spark.redshift
    OPTIONS (dbtable '{table-name}', tempdir '{bucket}', url '{url}', aws_iam_role
 '{aws-iam-role-arn}' ); """

spark.sql(s)

columns = ["country" ,"data"]
data = [("test-country","test-data")]
df = spark.sparkContext.parallelize(data).toDF(columns)

# Insert data into table
df.write.insertInto(table-name, overwrite=False)
df = spark.sql(f"SELECT * FROM {table-name}")
df.show()
```

Spark 커넥터 사용 시 고려 사항 및 제한 사항

- Amazon의 Spark에서 Amazon JDBC Redshift로 EMR 연결하려면 SSL 켜는 것이 좋습니다.

- Amazon Redshift 클러스터의 자격 증명은 다음에서 관리하는 것이 좋습니다. AWS Secrets Manager 모범 사례입니다. 사용을 참조하십시오. AWS Secrets Manager 예를 들어 Amazon Redshift에 연결하기 위한 자격 증명을 검색합니다.
- Amazon Redshift 인증 파라미터에 `aws_iam_role` 대한 파라미터와 함께 IAM 역할을 전달하는 것이 좋습니다.
- 현재 `tempformat` 파라미터는 Parquet 형식을 지원하지 않습니다.
- Amazon S3 위치를 `tempdir` URI 가리킵니다. 이 임시 디렉터리는 자동으로 정리되지 않으므로, 추가 비용이 발생할 수 있습니다.
- Amazon Redshift에 대한 다음 권장 사항을 고려합니다.
  - Amazon Redshift 클러스터에 대한 퍼블릭 액세스를 차단하는 것이 좋습니다.
  - Amazon Redshift 감사 로깅을 켜는 것이 좋습니다.
  - Amazon Redshift 저장 데이터 암호화를 켜는 것이 좋습니다.
- Amazon S3에 대한 다음 권장 사항을 고려합니다.
  - Amazon S3 버킷에 대한 퍼블릭 액세스를 차단하는 것이 좋습니다.
  - Amazon S3 서버 측 암호화를 사용하여 사용된 Amazon S3 버킷을 암호화하는 것이 좋습니다.
  - Amazon S3 수명 주기 정책을 사용하여 Amazon S3 버킷에 대한 보존 규칙을 정의하는 것이 좋습니다.
  - Amazon은 EMR 항상 오픈 소스에서 이미지로 가져온 코드를 확인합니다. 보안을 위해 Spark에서 Amazon S3로의 다음 인증 방법은 지원되지 않습니다.
    - 설정 AWS 구성 분류의 `hadoop-env` 액세스 키
    - 인코딩 AWS 에 있는 액세스 키 `tempdir` URI

커넥터 사용 및 지원되는 파라미터에 대한 자세한 내용은 다음 리소스를 참조하세요.

- Amazon Redshift 관리 안내서의 Apache Spark용 Amazon Redshift 통합
- Github의 `spark-redshift` community repository

Amazon 서버리스를 사용하여 DynamoDB에 연결 EMR

이 자습서에서는 미국 지명 위원회의 일부 데이터를 Amazon S3 버킷에 업로드한 다음 Amazon 서버리스의 Hive 또는 Spark를 사용하여 쿼리할 수 있는 EMR Amazon DynamoDB 테이블로 데이터를 복사합니다.

1단계: Amazon S3 버킷에 데이터 업로드

Amazon S3 버킷을 생성하려면 Amazon 심플 스토리지 서비스 콘솔 사용 설명서의 버킷 생성에 나와 있는 지침을 따르십시오. 에 *DOC-EXAMPLE-BUCKET* 대한 참조를 새로 생성한 버킷의 이름으로 바꾸십시오. 이제 EMR 서버리스 애플리케이션에서 작업을 실행할 준비가 되었습니다.

1. 다음 명령을 사용하여 샘플 데이터 `features.zip` 아카이브를 다운로드합니다.

   ```
   wget https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
   features.zip
   ```

2. 아카이브에서 `features.txt` 파일을 추출하고 파일의 처음 몇 줄을 확인합니다.

   ```
   unzip features.zip
   head features.txt
   ```

   결과는 다음과 비슷해야 합니다.

   ```
   1535908|Big Run|Stream|WV|38.6370428|-80.8595469|794
   875609|Constable Hook|Cape|NJ|40.657881|-74.0990309|7
   1217998|Gooseberry Island|Island|RI|41.4534361|-71.3253284|10
   26603|Boone Moore Spring|Spring|AZ|34.0895692|-111.410065|3681
   1506738|Missouri Flat|Flat|WA|46.7634987|-117.0346113|2605
   1181348|Minnow Run|Stream|PA|40.0820178|-79.3800349|1558
   1288759|Hunting Creek|Stream|TN|36.343969|-83.8029682|1024
   533060|Big Charles Bayou|Bay|LA|29.6046517|-91.9828654|0
   829689|Greenwood Creek|Stream|NE|41.596086|-103.0499296|3671
   541692|Button Willow Island|Island|LA|31.9579389|-93.0648847|98
   ```

   여기서 각 줄의 필드는 고유한 식별자, 이름, 자연 지형 유형, 주, 위도 (도), 경도 (도), 높이 (피트)를 나타냅니다.

3. Amazon S3에 데이터 업로드

   ```
   aws s3 cp features.txt s3://DOC-EXAMPLE-BUCKET/features/
   ```

2단계: 하이브 테이블 생성

Apache Spark 또는 Hive를 사용하여 Amazon S3에 업로드된 데이터를 포함하는 새 하이브 테이블을 생성합니다.

Spark

Spark를 사용하여 Hive 테이블을 생성하려면 다음 명령을 실행합니다.

```
import org.apache.spark.sql.SparkSession

val sparkSession = SparkSession.builder().enableHiveSupport().getOrCreate()

sparkSession.sql("CREATE TABLE hive_features \
    (feature_id BIGINT, \
    feature_name STRING, \
    feature_class STRING, \
    state_alpha STRING, \
    prim_lat_dec DOUBLE, \
    prim_long_dec DOUBLE, \
    elev_in_ft BIGINT) \
    ROW FORMAT DELIMITED \
    FIELDS TERMINATED BY '|' \
    LINES TERMINATED BY '\n' \
    LOCATION 's3://DOC-EXAMPLE_BUCKET/features';")
```

이제 파일의 데이터로 Hive 테이블이 채워졌습니다. `features.txt` 데이터가 테이블에 있는지 확인하려면 다음 예와 같이 Spark SQL 쿼리를 실행하십시오.

```
sparkSession.sql(
    "SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;")
```

Hive

Hive로 Hive 테이블을 만들려면 다음 명령을 실행합니다.

```
CREATE TABLE hive_features
    (feature_id             BIGINT,
    feature_name            STRING ,
    feature_class           STRING ,
    state_alpha             STRING,
    prim_lat_dec            DOUBLE ,
    prim_long_dec           DOUBLE ,
    elev_in_ft              BIGINT)
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY '|'
    LINES TERMINATED BY '\n'
```

```
    LOCATION 's3://DOC-EXAMPLE-BUCKET/features';
```

이제 파일의 데이터를 포함하는 Hive 테이블이 생겼습니다. `features.txt` 데이터가 테이블에 있는지 확인하려면 다음 예와 같이 HiveQL 쿼리를 실행합니다.

```
SELECT state_alpha, COUNT(*) FROM hive_features GROUP BY state_alpha;
```

3단계: DynamoDB에 데이터 복사

Spark 또는 Hive를 사용하여 데이터를 새 DynamoDB 테이블로 복사합니다.

Spark

이전 단계에서 만든 Hive 테이블의 데이터를 DynamoDB로 복사하려면 DynamoDB로 데이터 복사의 1-3단계를 따르십시오. 그러면 라는 새 DynamoDB 테이블이 생성됩니다. `Features` 그런 다음 다음 예제와 같이 텍스트 파일에서 직접 데이터를 읽고 DynamoDB 테이블로 복사할 수 있습니다.

```
import com.amazonaws.services.dynamodbv2.model.AttributeValue
import org.apache.hadoop.dynamodb.DynamoDBItemWritable
import org.apache.hadoop.dynamodb.read.DynamoDBInputFormat
import org.apache.hadoop.io.Text
import org.apache.hadoop.mapred.JobConf
import org.apache.spark.SparkContext

import scala.collection.JavaConverters._

object EmrServerlessDynamoDbTest {

    def main(args: Array[String]): Unit = {

        jobConf.set("dynamodb.input.tableName", "Features")
        jobConf.set("dynamodb.output.tableName", "Features")
        jobConf.set("dynamodb.region", "region")

        jobConf.set("mapred.output.format.class",
 "org.apache.hadoop.dynamodb.write.DynamoDBOutputFormat")
        jobConf.set("mapred.input.format.class",
 "org.apache.hadoop.dynamodb.read.DynamoDBInputFormat")

        val rdd = sc.textFile("s3://DOC-EXAMPLE-BUCKET/ddb-connector/")
            .map(row => {
```

```
                val line = row.split("\\|")
                val item = new DynamoDBItemWritable()

                val elevInFt = if (line.length > 6) {
                    new AttributeValue().withN(line(6))
                } else {
                    new AttributeValue().withNULL(true)
                }

                item.setItem(Map(
                    "feature_id" -> new AttributeValue().withN(line(0)),
                    "feature_name" -> new AttributeValue(line(1)),
                    "feature_class" -> new AttributeValue(line(2)),
                    "state_alpha" -> new AttributeValue(line(3)),
                    "prim_lat_dec" -> new AttributeValue().withN(line(4)),
                    "prim_long_dec" -> new AttributeValue().withN(line(5)),
                    "elev_in_ft" -> elevInFt)
                    .asJava)
                (new Text(""), item)
        })
        rdd.saveAsHadoopDataset(jobConf)
    }
}
```

Hive

이전 단계에서 만든 Hive 테이블의 데이터를 DynamoDB로 복사하려면 DynamoDB로 데이터 복사의 지침을 따르십시오.

4단계: DynamoDB에서 데이터 쿼리

Spark 또는 Hive를 사용하여 DynamoDB 테이블을 쿼리할 수 있습니다.

Spark

이전 단계에서 생성한 DynamoDB 테이블에서 데이터를 쿼리하려면 SQL Spark 또는 Spark를 사용할 수 있습니다. MapReduce API

Example — Spark를 사용하여 DynamoDB 테이블을 쿼리합니다. SQL

다음 Spark SQL 쿼리는 모든 기능 유형의 목록을 알파벳 순으로 반환합니다.

```
val dataFrame = sparkSession.sql("SELECT DISTINCT feature_class \
```

```
    FROM ddb_features \
    ORDER BY feature_class;")
```

다음 Spark SQL 쿼리는 문자 M으로 시작하는 모든 호수의 목록을 반환합니다.

```
val dataFrame = sparkSession.sql("SELECT feature_name, state_alpha \
    FROM ddb_features \
    WHERE feature_class = 'Lake' \
    AND feature_name LIKE 'M%' \
    ORDER BY feature_name;")
```

다음 Spark SQL 쿼리는 1마일보다 큰 특징을 3개 이상 포함하는 모든 상태의 목록을 반환합니다.

```
val dataFrame = sparkSession.dql("SELECT state_alpha, feature_class, COUNT(*) \
    FROM ddb_features \
    WHERE elev_in_ft > 5280 \
    GROUP by state_alpha, feature_class \
    HAVING COUNT(*) >= 3 \
    ORDER BY state_alpha, feature_class;")
```

Example — 스파크를 사용하여 DynamoDB 테이블을 쿼리합니다. MapReduce API

다음 MapReduce 쿼리는 모든 기능 유형의 목록을 알파벳순으로 반환합니다.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => (pair._1, pair._2.getItem))
    .map(pair => pair._2.get("feature_class").getS)
    .distinct()
    .sortBy(value => value)
    .toDF("feature_class")
```

다음 MapReduce 쿼리는 문자 M으로 시작하는 모든 호수의 목록을 반환합니다.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => (pair._1, pair._2.getItem))
    .filter(pair => "Lake".equals(pair._2.get("feature_class").getS))
    .filter(pair => pair._2.get("feature_name").getS.startsWith("M"))
    .map(pair => (pair._2.get("feature_name").getS,
 pair._2.get("state_alpha").getS))
```

```
    .sortBy(_._1)
    .toDF("feature_name", "state_alpha")
```

다음 MapReduce 쿼리는 1마일 이상의 특징을 3개 이상 포함하는 모든 주의 목록을 반환합니다.

```
val df = sc.hadoopRDD(jobConf, classOf[DynamoDBInputFormat], classOf[Text],
 classOf[DynamoDBItemWritable])
    .map(pair => pair._2.getItem)
    .filter(pair => pair.get("elev_in_ft").getN != null)
    .filter(pair => Integer.parseInt(pair.get("elev_in_ft").getN) > 5280)
    .groupBy(pair => (pair.get("state_alpha").getS, pair.get("feature_class").getS))
    .filter(pair => pair._2.size >= 3)
    .map(pair => (pair._1._1, pair._1._2, pair._2.size))
    .sortBy(pair => (pair._1, pair._2))
    .toDF("state_alpha", "feature_class", "count")
```

Hive

이전 단계에서 만든 DynamoDB 테이블의 데이터를 쿼리하려면 DynamoDB 테이블의 데이터 쿼리의 지침을 따르십시오.

교차 계정 액세스 설정

EMR서버리스에 대한 교차 계정 액세스를 설정하려면 다음 단계를 완료하십시오. 이 예제에서 `AccountA` 는 Amazon EMR 서버리스 애플리케이션을 생성한 계정이고 는 Amazon DynamoDB가 위치한 `AccountB` 계정입니다.

1. 에서 DynamoDB 테이블을 생성합니다. `AccountB` 자세한 내용은 1단계: 테이블 생성을 참조하십시오.
2. DynamoDB 테이블에 액세스할 수 `AccountB` 있는 `Cross-Account-Role-B` IAM 역할을 생성합니다.
   a. 에 로그인하십시오. AWS Management Console 에서 IAM 콘솔을 엽니다 https://console.aws.amazon.com/iam/.
   b. 역할을 선택하고 라는 새 역할을 생성합니다`Cross-Account-Role-B`. IAM역할을 만드는 방법에 대한 자세한 내용은 사용 설명서에서 IAM역할 만들기를 참조하십시오.
   c. 교차 계정 DynamoDB 테이블에 액세스할 수 있는 권한을 부여하는 IAM 정책을 생성합니다. 그런 다음 정책을 에 연결합니다. IAM `Cross-Account-Role-B`

다음은 CrossAccountTable DynamoDB 테이블에 대한 액세스 권한을 부여하는 정책입니다.

```
{"Version": "2012-10-17",
    "Statement": [
        {"Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:region:AccountB:table/
CrossAccountTable"
        }
    ]
}
```

d. Cross-Account-Role-B 역할에 대한 신뢰 관계를 편집합니다.

역할에 대한 신뢰 관계를 구성하려면 IAM 콘솔에서 2단계: Cross-Account Role-B에서 생성한 역할에 대한 신뢰 관계 탭을 선택합니다.

신뢰 관계 편집을 선택하고 다음 정책 문서를 추가합니다. 이 Job-Execution-Role-A 문서에서는 이 Cross-Account-Role-B 역할을 AccountA 맡을 수 있습니다.

```
{"Version": "2012-10-17",
  "Statement": [
    {"Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::AccountA:role/Job-Execution-Role-A"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

e. 수임할 AccountA 수 Job-Execution-Role-A 있는 - STS Assume role 권한을 부여하세요Cross-Account-Role-B.

IAM콘솔에서 AWS 계정 AccountA, 선택합니다Job-Execution-Role-A. 다음 정책 명령을 Job-Execution-Role-A에 추가하여 Cross-Account-Role-B 역할에서 AssumeRole 작업을 허용합니다.

```
{"Version": "2012-10-17",
    "Statement": [
        {"Effect": "Allow",
```

```
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::AccountB:role/Cross-Account-Role-B"
        }
    ]
}
```

f. 코어 사이트 `com.amazonaws.emr.AssumeRoleAWSCredentialsProvider` 분류에서와 같이 값을 사용하여 `dynamodb.customAWSCredentialsProvider` 속성을 설정합니다. ARN값이 인 환경 변수를 `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` 설정합니다. `Cross-Account-Role-B`

3. 를 사용하여 Spark 또는 Hive 작업을 실행합니다. `Job-Execution-Role-A`

고려 사항

DynamoDB 커넥터를 아파치 스파크와 함께 사용할 때의 고려 사항

- SQLSpark는 스토리지 핸들러 옵션을 사용한 Hive 테이블 생성을 지원하지 않습니다. 자세한 내용은 Apache Spark 설명서의 Hive 테이블의 스토리지 형식 지정을 참조하십시오.
- SQLSpark는 스토리지 핸들러를 사용한 `STORED BY` 작업을 지원하지 않습니다. 외부 Hive 테이블을 통해 DynamoDB 테이블과 상호 작용하려면 먼저 Hive를 사용하여 테이블을 생성하십시오.
- 쿼리를 DynamoDB 쿼리로 변환하기 위해 DynamoDB 커넥터는 조건자 푸시다운을 사용합니다. 술어 푸시다운은 DynamoDB 테이블의 파티션 키에 매핑된 열을 기준으로 데이터를 필터링합니다. 조건부 푸시다운은 Spark와 함께 커넥터를 사용할 때만 작동하며 Spark에서는 작동하지 않습니다. SQL MapReduce API

DynamoDB 커넥터를 아파치 하이브와 함께 사용할 때의 고려 사항

최대 매퍼 수 조정

- `SELECT`쿼리를 사용하여 DynamoDB에 매핑되는 외부 Hive 테이블에서 데이터를 읽는 경우 서버리스의 맵 작업 수는 DynamoDB EMR 테이블에 구성된 총 읽기 처리량을 맵 작업당 처리량으로 나눈 값으로 계산됩니다. 맵 작업당 기본 처리량은 100입니다.
- Hive 작업은 DynamoDB에 구성된 읽기 처리량에 따라 EMR 서버리스 애플리케이션당 구성된 최대 컨테이너 수를 초과하는 맵 작업 수를 사용할 수 있습니다. 또한 장기 실행 Hive 쿼리는 DynamoDB 테이블의 프로비저닝된 읽기 용량을 모두 소비할 수 있습니다. 이는 다른 사용자에게 부정적인 영향을 미칩니다.

- dynamodb.max.map.tasks속성을 사용하여 맵 작업의 상한선을 설정할 수 있습니다. 또한 이 속성을 사용하여 작업 컨테이너 크기에 따라 각 맵 작업에서 읽는 데이터의 양을 조정할 수 있습니다.
- Hive 쿼리 수준에서 또는 start-job-run 명령 hive-site 분류에서 dynamodb.max.map.tasks 속성을 설정할 수 있습니다. 이 값은 1보다 크거나 같아야 합니다. Hive가 쿼리를 처리할 때 결과 Hive 작업은 DynamoDB 테이블에서 읽을 dynamodb.max.map.tasks 때의 값만 사용합니다.

작업별 쓰기 처리량 조정

- EMR서버리스의 작업당 쓰기 처리량은 DynamoDB 테이블에 구성된 총 쓰기 처리량을 속성 값으로 나누어 계산합니다. mapreduce.job.maps Hive의 경우 이 속성의 기본값은 2입니다. 따라서 Hive 작업의 마지막 단계에 있는 처음 두 작업은 쓰기 처리량을 모두 소비할 수 있습니다. 이로 인해 같은 작업이나 다른 작업에 있는 다른 작업의 쓰기 제한이 발생합니다.
- 쓰기 제한을 방지하기 위해 최종 단계의 작업 수 또는 작업당 할당하려는 쓰기 처리량을 기반으로 mapreduce.job.maps 속성 값을 설정할 수 있습니다. 서버리스의 start-job-run 명령 mapred-site 분류에서 이 속성을 설정합니다. EMR

보안

클라우드 보안: AWS 최우선 과제입니다. 로서 AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 두 회사 간의 공동 책임입니다. AWS 그리고 당신. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드 보안 — AWS 실행 중인 인프라를 보호할 책임이 있습니다. AWS 의 서비스 AWS 클라우드. AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 제3자 감사자는 보안 조치의 일환으로 당사 보안의 효과를 정기적으로 테스트하고 확인합니다. AWS 규정 준수 프로그램. Amazon EMR Serverless에 적용되는 규정 준수 프로그램에 대해 알아보려면 다음을 참조하십시오. AWS 규정 준수 프로그램별 범위 내 서비스.
- 클라우드에서의 보안 — 귀하의 책임은 다음에 따라 결정됩니다. AWS 사용하는 서비스. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amazon EMR Serverless를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 이 장의 주제에서는 Amazon EMR Serverless를 구성하고 기타를 사용하는 방법을 보여줍니다. AWS 보안 및 규정 준수 목표를 충족하기 위한 서비스.

주제

Amazon EMR 서버리스의 보안 모범 사례

Amazon EMR Serverless는 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용해 주십시오.

최소 권한의 원칙 적용

EMR서버리스는 실행 IAM 역할과 같은 역할을 사용하는 애플리케이션에 대한 세분화된 액세스 정책을 제공합니다. 애플리케이션 조치와 로그 대상에 대한 액세스 등 작업에 필요한 최소한의 권한만 실행 역할에 부여하는 것이 좋습니다. 또한 애플리케이션 코드가 변경될 때마다 정기적으로 권한을 확인하기 위해 작업을 감사하는 것이 좋습니다.

신뢰할 수 없는 애플리케이션 코드 격리

EMR서버리스는 서로 다른 EMR 서버리스 애플리케이션에 속하는 작업 간에 완전한 네트워크 격리를 생성합니다. 작업 수준 격리가 필요한 경우 작업을 여러 서버리스 애플리케이션으로 격리하는 것을 고려해 보십시오. EMR

역할 기반 액세스 제어 () 권한 RBAC

관리자는 서버리스 애플리케이션에 대한 역할 기반 액세스 제어 (RBAC) 권한을 엄격하게 제어해야 합니다. EMR

데이터 보호

The AWS 공동 책임 모델은 Amazon EMR 서버리스의 데이터 보호에 적용됩니다. 이 모델에 설명된 바와 같이 AWS 모든 시스템을 운영하는 글로벌 인프라를 보호하는 책임이 있습니다. AWS 클라우드. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 이 콘텐츠에는 에 대한 보안 구성 및 관리 작업이 포함됩니다. AWS 사용하는 서비스. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시를 참조하십시오FAQ. 유럽의 데이터 보호에 대한 자세한 내용은 다음을 참조하십시오. AWS 공동 책임 모델 및 관련 GDPR 블로그 게시물 AWS 보안 블로그.

데이터 보호를 위해 다음을 보호하는 것이 좋습니다. AWS 계정 자격 증명 및 개별 계정 설정 AWS ID 및 액세스 관리 (IAM). 이러한 방식에서는 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정마다 다단계 인증 (MFA) 을 사용하십시오.
- SSL/를 사용하여 다음과 TLS 통신할 수 있습니다. AWS 있습니다. TLS1.2 이상을 권장합니다.

- 다음을 사용하여 사용자 활동 로깅을 API 설정하고 AWS CloudTrail.
- 사용 AWS 암호화 솔루션, 모든 기본 보안 제어 기능 포함 AWS 서비스.
- Amazon S3에 저장된 개인 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- Amazon EMR 서버리스 암호화 옵션을 사용하여 저장된 데이터와 전송 중인 데이터를 암호화합니다.
- 액세스 시 FIPS 140-2개의 검증된 암호화 모듈이 필요한 경우 AWS 명령줄 인터페이스 또는 API an을 통해 엔드포인트를 사용하십시오. FIPS 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준 (FIPS) 140-2를 참조하십시오.

이름 필드와 같은 자유 형식 필드에 고객 계정 번호와 같은 중요 식별 정보를 절대 입력하지 마십시오. 여기에는 Amazon EMR 서버리스 또는 다른 서버를 사용하는 경우가 포함됩니다. AWS 콘솔을 사용하는 서비스, API AWS CLI, 또는 AWS SDKs. Amazon EMR Serverless 또는 기타 서비스에 입력하는 모든 데이터는 진단 로그에 포함되도록 선택될 수 있습니다. 외부 서버에 데이터를 제공할 때는 해당 서버에 대한 요청을 URL 검증하기 위한 자격 증명 정보를 포함시키지 마십시오. URL

저장 중 암호화

데이터 암호화는 권한 없는 사용자가 클러스터 및 관련 데이터 스토리지 시스템에서 데이터를 읽는 것을 방지하는 데 도움이 됩니다. 여기에는 영구 미디어에 저장된 데이터인 유휴 데이터와 네트워크를 이동하는 동안 가로챌 수 있는 데이터인 전송 중 데이터가 포함됩니다.

데이터 암호화에는 키와 인증서가 필요합니다. 에서 관리하는 키를 비롯한 여러 옵션 중에서 선택할 수 있습니다. AWS Key Management Service, Amazon S3에서 관리하는 키, 그리고 사용자가 제공하는 사용자 지정 공급자의 키와 인증서. 를 사용하는 경우 AWS KMS 키 제공자의 경우 암호화 키의 저장 및 사용에 대한 요금이 부과됩니다. 자세한 내용은 단원을 참조하세요.AWS KMS 가격.

암호화 옵션을 지정하기 전에 사용할 키 및 인증서 관리 시스템을 결정합니다. 그런 다음 암호화 설정의 일부로 지정하는 사용자 지정 제공업체에 대한 키와 인증서를 생성합니다.

Amazon S3에 저장된 EMRFS 데이터에 대한 저장 중 암호화

각 EMR 서버리스 애플리케이션은 EMRFS (EMR파일 시스템) 이 포함된 특정 릴리스 버전을 사용합니다. Amazon S3 암호화는 Amazon S3에서 읽고 쓰는 EMR 파일 시스템 (EMRFS) 객체와 함께 작동합니다. 저장 중 암호화를 활성화할 때 Amazon S3 서버 측 암호화 (SSE) 또는 클라이언트 측 암호화 (CSE) 를 기본 암호화 모드로 지정할 수 있습니다. 선택적으로 버킷별 암호화 재정의를 사용하여 개별 버킷에 서로 다른 암호화 방법을 지정할 수 있습니다. Amazon S3 암호화의 활성화 여부에 관계없이 전송 계층 보안 (TLS) 은 EMR 클러스터 노드와 Amazon S3 간에 전송되는 EMRFS 객체를 암호화합니

다. 고객 관리 키와 CSE 함께 Amazon S3를 사용하는 경우, EMR 서버리스 애플리케이션에서 작업을 실행하는 데 사용되는 실행 역할에 해당 키에 대한 액세스 권한이 있어야 합니다. Amazon S3 암호화에 대한 자세한 내용은 Amazon Simple Storage 서비스 개발자 안내서의 암호화를 사용한 데이터 보호를 참조하십시오.

> **Note**
>
> 사용하는 경우 AWS KMS암호화 키의 저장 및 사용에는 요금이 부과됩니다. 자세한 내용은 단원을 참조하세요.AWS KMS 가격.

Amazon S3 서버 측 암호화

Amazon S3 서버 측 암호화를 설정하면 Amazon S3에서 데이터를 디스크에 쓸 때 객체 수준에서 데이터를 암호화하고 데이터에 액세스할 때 데이터의 암호를 해독합니다. 자세한 내용은 Amazon Simple Storage Service 개발자 안내서의 서버 측 암호화를 사용한 데이터 보호를 참조하십시오. SSE

Amazon SSE EMR Serverless에서 지정할 때 두 가지 다른 키 관리 시스템 중에서 선택할 수 있습니다.

- SSE-S3 - Amazon S3가 사용자를 대신하여 키를 관리합니다. EMR서버리스에서는 추가 설정이 필요하지 않습니다.
- SSE- KMS - 다음을 사용합니다. AWS KMS key EMR서버리스에 적합한 정책을 설정하기 위함입니다. EMR서버리스에서는 추가 설정이 필요하지 않습니다.

사용하려면 AWS KMS Amazon S3에 쓰는 데이터를 암호화하는 경우 를 사용할 때 두 가지 옵션이 `StartJobRun` API 있습니다. Amazon S3에 쓰는 모든 항목에 대해 암호화를 활성화하거나 특정 버킷에 쓰는 데이터에 대해 암호화를 활성화할 수 있습니다. 에 `StartJobRun` API 대한 자세한 내용은 EMR서버리스 API 참조를 참조하십시오.

켜려면 AWS KMS Amazon S3에 쓰는 모든 데이터를 암호화하려면 를 호출할 때 다음 명령을 사용하십시오 `StartJobRun`API.

```
--conf spark.hadoop.fs.s3.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.serverSideEncryption.kms.keyId=<kms_id>
```

활성화하려면 AWS KMS 특정 버킷에 쓰는 데이터를 암호화하려면 를 호출할 때 다음 명령을 사용하십시오 `StartJobRun`API.

```
--conf spark.hadoop.fs.s3.bucket.<DOC-EXAMPLE-BUCKET>.enableServerSideEncryption=true
--conf spark.hadoop.fs.s3.bucket.<DOC-EXAMPLE-
BUCKET>.serverSideEncryption.kms.keyId=<kms-id>
```

SSE고객 제공 키 (SSE-C) 는 서버리스에서 사용할 수 없습니다. EMR

Amazon S3 클라이언트 측 암호화

Amazon S3 클라이언트 측 암호화를 사용하면 Amazon S3 암호화 및 복호화가 모든 Amazon 릴리스에서 제공되는 EMRFS 클라이언트에서 수행됩니다. EMR 객체는 Amazon S3에 업로드되기 전에 암호화되고 다운로드된 후 암호 해독됩니다. 지정하는 공급자는 클라이언트가 사용하는 암호화 키를 제공합니다. 클라이언트는 에서 제공한 키를 사용할 수 있습니다. AWS KMS (CSE-KMS) 또는 클라이언트 측 루트 키 (CSE-C) 를 제공하는 사용자 지정 Java 클래스 암호화 세부 사항은 지정된 공급자와 KMS 복호화되거나 암호화되는 객체의 메타데이터에 따라 CSE -와 -C 간에 CSE 약간 다릅니다. 고객 관리 키와 CSE 함께 Amazon S3를 사용하는 경우, EMR 서버리스 애플리케이션에서 작업을 실행하는 데 사용되는 실행 역할에 해당 키에 대한 액세스 권한이 있어야 합니다. 추가 KMS 요금이 부과될 수 있습니다. 이러한 차이점에 대한 자세한 내용은 Amazon Simple Storage Service 개발자 안내서의 클라이언트 측 암호화를 사용한 데이터 보호를 참조하십시오.

로컬 디스크 암호화

임시 스토리지에 저장된 데이터는 업계 표준 -256 암호화 알고리즘을 사용하는 서비스 소유 키로 암호화됩니다. AES

키 관리

키를 자동으로 교체하도록 구성할 KMS 수 있습니다. KMS 이렇게 하면 1년에 한 번 키가 로테이션되고 이전 키는 무기한 저장되므로 데이터를 계속 해독할 수 있습니다. 자세한 내용은 고객 마스터 키 교체를 참조하십시오.

전송 중 암호화

Amazon EMR Serverless에서는 다음과 같은 애플리케이션별 암호화 기능을 사용할 수 있습니다.

- Spark
  - 기본적으로 Spark 드라이버와 실행자 간의 통신은 인증되고 내부적으로 이루어집니다. RPC드라이버와 실행자 간의 통신은 암호화됩니다.
- Hive
  - 사이의 통신 AWS Glue 메타스토어 및 EMR 서버리스 애플리케이션은 를 통해 실행됩니다. TLS

Amazon S3 버킷 IAM 정책의 aws: SecureTransport 조건을 사용하여 HTTPS (TLS) 를 통한 암호화된 연결만 허용해야 합니다.

Amazon EMR 서버리스의 ID 및 Access Management (IAM)

AWS Identity and Access Management (IAM) 는 AWS 서비스 이를 통해 관리자는 다음 항목에 대한 액세스를 안전하게 제어할 수 있습니다. AWS 있습니다. IAM관리자는 Amazon EMR Serverless 리소스를 사용할 수 있는 인증 (로그인) 및 권한 부여 (권한 보유) 를 받을 수 있는 사용자를 제어합니다. IAM는 AWS 서비스 추가 비용 없이 사용할 수 있습니다.

주제

고객

사용 방법 AWS Identity and Access Management (IAM) 는 Amazon EMR 서버리스에서 수행하는 작업에 따라 다릅니다.

서비스 사용자 — Amazon EMR Serverless 서비스를 사용하여 작업을 수행하는 경우 관리자가 필요한 자격 증명과 권한을 제공합니다. 더 많은 Amazon EMR Serverless 기능을 사용하여 작업을 수행함에 따라 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. Amazon EMR 서버리스의 기능에 액세스할 수 없는 경우 을 참조하십시오Amazon EMR 서버리스 ID 및 액세스 문제 해결.

서비스 관리자 — 회사에서 Amazon EMR 서버리스 리소스를 담당하고 있다면 Amazon 서버리스에 대한 전체 액세스 권한을 가지고 있을 것입니다. EMR 서비스 사용자가 액세스해야 하는 Amazon EMR

Serverless 기능 및 리소스를 결정하는 것은 여러분의 몫입니다. 그런 다음 IAM 관리자에게 서비스 사용자의 권한을 변경해 달라는 요청을 제출해야 합니다. 이 페이지의 정보를 검토하여 의 기본 개념을 IAM 이해하십시오. 회사에서 Amazon EMR Serverless를 사용하는 방법에 대한 자세한 내용은 IAM 을 참조하십시오Amazon EMR 서버리스의 ID 및 Access Management (IAM).

IAM관리자 — IAM 관리자라면 Amazon EMR Serverless에 대한 액세스를 관리하기 위한 정책을 작성하는 방법에 대해 자세히 알아보는 것이 좋습니다. 에서 IAM 사용할 수 있는 Amazon EMR Serverless ID 기반 정책의 예를 보려면 을 참조하십시오. 서버리스를 위한 샘플 ID 기반 정책 EMR

ID를 통한 인증

인증은 로그인하는 방법입니다. AWS ID 자격 증명 사용. 인증 (로그인) 을 받아야 합니다. AWS다음과 같이) AWS 계정 루트 사용자 IAM사용자로서, 또는 IAM 역할을 맡아서.

에 로그인할 수 있습니다. AWS ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 사용할 수 있습니다. AWS IAM Identity Center 페더레이션 ID의 예로는 (IAMID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 있습니다. 페더레이션 ID로 로그인하는 경우 관리자는 이전에 역할을 사용하여 ID 페더레이션을 설정했습니다. IAM 액세스하는 경우 AWS 페더레이션을 사용하면 간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 로그인할 수 있습니다. AWS Management Console 또는 AWS 액세스 포털. 로그인에 대한 자세한 내용은 AWS로그인하는 방법을 참조하십시오. AWS 계정의 AWS 로그인 사용자 가이드.

액세스하는 경우 AWS 프로그래밍 방식으로 AWS 자격 증명을 사용하여 요청에 암호로 서명할 수 있는 소프트웨어 개발 키트 (SDKCLI) 와 명령줄 인터페이스 () 를 제공합니다. 사용하지 않는 경우 AWS 도구를 사용하려면 직접 요청에 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 서명을 참조하십시오. AWS APIIAM사용 설명서의 요청.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예: AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA) 을 사용할 것을 권장합니다. 자세한 내용은 의 다단계 인증을 참조하십시오. AWS IAM Identity Center 사용 설명서 및 다단계 인증 사용 () MFA AWS(출처: IAM 사용 설명서).

AWS 계정 루트 사용자

생성할 때 AWS 계정모든 계정에 완전히 액세스할 수 있는 하나의 로그인 ID로 시작합니다. AWS 서비스 및 계정 내 리소스. 이 ID를 다음과 같이 부릅니다. AWS 계정 루트 사용자는 계정을 만들 때 사용한 이메일 주소와 암호로 로그인하여 액세스할 수 있습니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작

업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 작업의 전체 목록은 사용 설명서의 루트 사용자 자격 증명이 필요한 작업을 참조하십시오. IAM

페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 비롯한 수동 사용자가 ID 공급자와의 페더레이션을 사용하여 액세스하도록 하는 것입니다. AWS 서비스 임시 자격 증명을 사용하여

페더레이션 ID는 기업 사용자 디렉토리의 사용자, 웹 ID 제공업체, AWS Directory Service, ID 센터 디렉터리 또는 액세스하는 모든 사용자 AWS 서비스 ID 소스를 통해 제공된 자격 증명을 사용합니다. 페더레이션된 ID가 액세스하는 경우 AWS 계정역할을 맡고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해서는 다음을 사용하는 것이 좋습니다. AWS IAM Identity Center. IAMIdentity Center에서 사용자 및 그룹을 만들거나 자체 ID 소스의 사용자 및 그룹 집합에 연결하고 동기화하여 모든 위치에서 사용할 수 있습니다. AWS 계정 및 애플리케이션. ID 센터에 대한 자세한 내용은 IAM ID 센터란 IAM 무엇입니까? 를 참조하십시오. ... 에서 AWS IAM Identity Center 사용자 가이드.

IAM 사용자 및 그룹

IAM사용자는 내 정체성에 속해 있습니다. AWS 계정 이는 한 사람이나 애플리케이션에 대한 특정 권한을 가지고 있습니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명을 가진 IAM 사용자를 만드는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 특정 사용 사례에서 IAM 사용자의 장기 자격 증명이 필요한 경우에는 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 사용 설명서의 장기 자격 증명이 필요한 사용 사례에 대한 정기적인 액세스 키 IAM 교체를 참조하십시오.

IAM그룹은 IAM 사용자 컬렉션을 지정하는 ID입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어 이름을 지정한 IAMAdmins그룹을 만들고 해당 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증 정보를 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세히 알아보려면 사용 설명서의 역할 대신 IAM 사용자를 만드는 시기를 참조하십시오. IAM

IAM역할

IAM역할은 내 안의 정체성입니다. AWS 계정 여기에는 특정 권한이 있습니다. 사용자와 비슷하지만 특정 IAM 사용자와는 관련이 없습니다. 에서 일시적으로 IAM 역할을 맡을 수 있습니다. AWS Management Console 역할을 바꿔서 말이죠. 를 호출하여 역할을 맡을 수 있습니다. AWS CLI 또는

AWS API오퍼레이션을 사용하거나 사용자 지정을 사용합니다URL. 역할 사용 방법에 대한 자세한 내용은 사용 IAM설명서의 IAM역할 사용을 참조하십시오.

IAM임시 자격 증명이 있는 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션을 위한 역할에 대한 자세한 내용은 IAM사용 설명서의 타사 ID 제공자를 위한 역할 생성을 참조하십시오. IAMIdentity Center를 사용하는 경우 권한 집합을 구성합니다. ID가 인증된 후 액세스할 수 있는 대상을 제어하기 위해 IAM Identity Center는 권한 집합을 역할의 상관 관계와 연결합니다. IAM 권한 집합에 대한 자세한 내용은 권한 집합의 사용 권한 집합을 참조하십시오. AWS IAM Identity Center 사용 설명서.
- 임시 IAM 사용자 권한 — IAM 사용자 또는 역할은 역할을 맡아 특정 작업에 대해 일시적으로 다른 권한을 부여받을 수 있습니다. IAM
- 계정 간 액세스 - IAM 역할을 사용하여 다른 계정의 사용자 (신뢰할 수 있는 사용자) 가 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 하지만 일부 경우에는 AWS 서비스역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 계정 간 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 사용 설명서의 IAM 계정 간 리소스 액세스를 참조하십시오. IAM
- 서비스 간 액세스 — 일부 AWS 서비스 다른 기능 사용 AWS 서비스. 예를 들어, 서비스를 호출하면 해당 서비스가 Amazon에서 애플리케이션을 EC2 실행하거나 Amazon S3에 객체를 저장하는 것이 일반적입니다. 서비스는직접적으로 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 경우 AWS, 귀하는 주도자로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS전화를 건 주체의 권한을 사용합니다. AWS 서비스, 요청과 결합 AWS 서비스 다운스트림 서비스에 요청하기. FAS요청은 서비스가 다른 서비스와의 상호 작용이 필요한 요청을 수신할 때만 이루어집니다. AWS 서비스 또는 완료해야 할 리소스. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS요청 시 적용되는 정책 세부 정보는 전달 액세스 세션을 참조하십시오.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 간주하는 IAM 역할입니다. IAM관리자는 내부에서 IAM 서비스 역할을 만들고, 수정하고, 삭제할 수 있습니다. 자세한 내용은 권한을 위임하기 위한 역할 만들기를 참조하십시오. AWS 서비스(출처: IAM 사용 설명서).
  - 서비스 연결 역할 - 서비스 연결 역할은 다음과 연결된 서비스 역할 유형입니다. AWS 서비스. 서비스가 사용자를 대신하여 작업을 수행하는 역할을 맡을 수 있습니다. 서비스 연결 역할은 다음과

같습니다. AWS 계정 서비스가 소유합니다. IAM관리자는 서비스 연결 역할에 대한 권한을 볼 수 있지만 편집할 수는 없습니다.

- Amazon에서 실행되는 애플리케이션 EC2 — IAM 역할을 사용하여 EC2 인스턴스에서 실행 중이고 다음을 생성하는 애플리케이션에 대한 임시 자격 증명을 관리할 수 있습니다. AWS CLI 또는 AWS API요청. EC2인스턴스 내에 액세스 키를 저장하는 것보다 이 방법이 더 좋습니다. 할당하려면 AWS EC2인스턴스에 역할을 부여하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며, 이를 통해 EC2 인스턴스에서 실행 중인 프로그램이 임시 자격 증명을 얻을 수 있습니다. 자세한 내용은 사용 설명서의 IAM역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 IAM 참조하십시오.

IAM역할을 사용할지 IAM 사용자를 사용할지 알아보려면 사용 설명서의 IAM 역할 생성 시기 (사용자 대신) 를 IAM참조하십시오.

정책을 사용한 액세스 관리

에서 액세스를 제어할 수 있습니다. AWS 정책을 생성하여 정책에 연결함으로써 AWS ID 또는 리소스. 정책은 다음의 객체입니다. AWS 이는 ID 또는 리소스와 연결될 경우 해당 권한을 정의합니다. AWS 보안 주체 (사용자, 루트 사용자 또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 다음 위치에 저장됩니다. AWS JSON문서로. JSON정책 문서의 구조 및 내용에 대한 자세한 내용은 IAM사용 설명서의 JSON 정책 개요를 참조하십시오.

관리자는 다음을 사용할 수 있습니다. AWS JSON정책을 통해 누가 무엇에 액세스할 수 있는지 지정합니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. IAM관리자는 IAM 정책을 생성하여 필요한 리소스에서 작업을 수행할 수 있는 권한을 사용자에게 부여할 수 있습니다. 그러면 관리자가 역할에 IAM 정책을 추가할 수 있으며, 사용자는 역할을 수임할 수 있습니다.

IAM정책은 작업을 수행하는 데 사용하는 방법에 관계없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 에서 역할 정보를 가져올 수 있습니다. AWS Management Console, AWS CLI, 또는 AWS API.

보안 인증 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를

제어합니다. ID 기반 정책을 만드는 방법을 알아보려면 사용 설명서의 IAM정책 생성을 참조하십시오. IAM

보안 인증 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 조직의 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립형 정책입니다. AWS 계정. 관리형 정책에는 다음이 포함됩니다. AWS 관리형 정책 및 고객 관리형 정책. 관리형 정책과 인라인 정책 중에서 선택하는 방법을 알아보려면 IAM사용 설명서의 관리형 정책과 인라인 정책 중 선택을 참조하십시오.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예로는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 또는 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 사용할 수 없습니다. AWS 리소스 기반 정책의 관리형 정책. IAM

액세스 제어 목록 () ACLs

액세스 제어 목록 (ACLs) 은 리소스에 액세스할 수 있는 권한을 가진 주체 (계정 구성원, 사용자 또는 역할) 를 제어합니다. ACLs정책 문서 형식을 사용하지는 않지만 리소스 기반 정책과 JSON 비슷합니다.

아마존 S3, AWS WAF, VPC Amazon은 지원하는 서비스의 예입니다ACLs. 자세한 내용은 Amazon 심플 스토리지 서비스 개발자 안내서의 액세스 제어 목록 (ACL) 개요를 참조하십시오. ACLs

기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 - 권한 경계는 ID 기반 정책이 IAM 엔티티 (IAM사용자 또는 역할) 에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나

에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 내용은 사용 IAM설명서의 IAM 엔티티의 권한 경계를 참조하십시오.

- 서비스 제어 정책 (SCPs) — SCPs 조직 또는 OU (조직 구성 단위) 에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations. AWS Organizations 여러 개를 그룹화하고 중앙에서 관리하는 서비스입니다. AWS 계정 귀사가 소유한 것입니다. 조직의 모든 기능을 사용하도록 설정하면 일부 또는 모든 계정에 서비스 제어 정책 (SCPs) 을 적용할 수 있습니다. 각 항목을 포함하여 구성원 계정의 엔티티에 대한 권한을 SCP 제한합니다. AWS 계정 루트 사용자. Organizations 및 SCPs 에 대한 자세한 내용은 의 서비스 제어 정책을 참조하십시오. AWS Organizations 사용 설명서.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 보안 인증 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 내용은 IAM사용 설명서의 세션 정책을 참조하십시오.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 방법을 알아보려면 AWS 여러 정책 유형이 관련된 경우 요청을 허용할지 여부를 결정하려면 IAM사용 설명서의 정책 평가 로직을 참조하십시오.

EMR서버리스의 작동 방식 IAM

Amazon EMR Serverless에 대한 액세스를 관리하는 IAM 데 사용하기 전에 Amazon 서버리스에서 사용할 수 있는 IAM 기능에 대해 알아보십시오. EMR

IAM서버리스에서 사용할 수 있는 기능 EMR

IAM기능	Amazon EMR 서버리스 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	아니요

IAM기능	Amazon EMR 서버리스 지원
ACLs	아니요
ABAC(정책의 태그)	예
임시 보안 인증	예
보안 주체 권한	예
서비스 역할	아니요
서비스 링크 역할	예

EMR서버리스 및 기타 방식을 개괄적으로 파악하려면 AWS 서비스가 대부분의 IAM 기능과 함께 작동합니다. AWSIAM사용 IAM 설명서에서 함께 사용할 수 있는 서비스

서버리스를 위한 ID 기반 정책 EMR

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. ID 기반 정책을 만드는 방법을 알아보려면 사용 설명서의 IAM정책 생성을 참조하십시오. IAM

IAMID 기반 정책을 사용하면 허용 또는 거부된 작업 및 리소스는 물론 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 보안 인증 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 사용 IAM설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

서버리스를 위한 샘플 ID 기반 정책 EMR

Amazon EMR 서버리스 ID 기반 정책의 예를 보려면 을 참조하십시오. 서버리스의 ID 기반 정책 예제 EMR

서버리스 내의 리소스 기반 정책 EMR

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예로는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 또는 AWS 서비스.

계정 간 액세스를 활성화하려면 다른 계정의 전체 계정 또는 IAM 엔티티를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하십시오. 보안 주체와 리소스가 다른 경우 AWS 계정신뢰할 수 있는 계정의 IAM 관리자는 주체 개체 (사용자 또는 역할) 에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM사용 설명서의 계정 간 리소스 액세스를 참조하십시오. IAM

EMR서버리스를 위한 정책 조치

정책 작업 지원: 예

관리자는 다음을 사용할 수 있습니다. AWS JSON정책을 통해 누가 무엇에 액세스할 수 있는지 지정합니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

정책 `Action` 요소는 JSON 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 일반적으로 정책 조치의 이름은 관련 조치와 동일합니다. AWS API오퍼레이션. 일치하는 작업이 없는 권한 전용 작업과 같은 몇 가지 예외가 있습니다. API 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

EMR서버리스 작업 목록을 보려면 서비스 권한 부여 참조의 Amazon EMR Serverless용 작업, 리소스 및 조건 키를 참조하십시오.

EMR서버리스의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
emr-serverless
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "emr-serverless:action1",
      "emr-serverless:action2"
         ]
```

Amazon EMR 서버리스 ID 기반 정책의 예를 보려면 을 참조하십시오. 서버리스의 ID 기반 정책 예제 EMR

서버리스를 위한 정책 리소스 EMR

정책 리소스 지원: 예

관리자는 다음을 사용할 수 있습니다. AWS JSON정책을 통해 누가 무엇에 액세스할 수 있는지 지정합니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

ResourceJSON정책 요소는 작업이 적용되는 하나 또는 여러 개의 객체를 지정합니다. 문장에는 Resource또는 NotResource요소가 반드시 추가되어야 합니다. Amazon 리소스 이름 (ARN) 을 사용하여 리소스를 지정하는 것이 가장 좋습니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 태스크를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Amazon EMR 서버리스 리소스 유형 및 해당 ARNs 유형의 목록을 보려면 서비스 인증 참조의 Amazon EMR Serverless에서 정의한 리소스를 참조하십시오. 각 리소스에 지정할 수 있는 작업을 알아보려면 Amazon ARN EMR Serverless의 작업, 리소스 및 조건 키를 참조하십시오.

Amazon EMR 서버리스 ID 기반 정책의 예를 보려면 을 참조하십시오. 서버리스의 ID 기반 정책 예제 EMR

서버리스의 정책 조건 키 EMR

서비스별 정책 조건 키 지원	아니요

관리자는 다음을 사용할 수 있습니다. AWS JSON정책을 통해 누가 무엇에 액세스할 수 있는지 지정합니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

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.

명령문에 여러 Condition 요소를 지정하거나 단일 Condition 요소에 여러 키를 지정하는 경우 AWS 논리 AND 연산을 사용하여 요소를 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 AWS 논리 OR 연산을 사용하여 조건을 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어 리소스에 IAM 사용자 이름이 태그가 지정된 경우에만 리소스에 대한 액세스 권한을 IAM 사용자에게 부여할 수 있습니다. 자세한 내용은 IAM사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하십시오.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모두 보려면 AWS 글로벌 조건 키는 다음을 참조하십시오. AWSIAM사용 설명서의 글로벌 조건 컨텍스트 키

Amazon EMR Serverless 조건 키 목록을 확인하고 조건 키를 사용할 수 있는 작업 및 리소스를 알아보려면 서비스 권한 부여 참조의 Amazon EMR Serverless용 작업, 리소스 및 조건 키를 참조하십시오.

모든 Amazon EC2 작업은 aws:RequestedRegion 및 ec2:Region 조건 키를 지원합니다. 자세한 내용은 예: 특정 지역에 대한 액세스 제한을 참조하십시오.

서버리스의 EMR 액세스 제어 목록 (ACLs)

지원ACLs: 아니요

액세스 제어 목록 (ACLs) 은 리소스에 액세스할 수 있는 권한을 가진 주체 (계정 구성원, 사용자 또는 역할) 를 제어합니다. ACLs정책 문서 형식을 사용하지는 않지만 리소스 기반 정책과 JSON 비슷합니다.

서버리스를 사용한 속성 기반 액세스 제어 () ABAC EMR

지원 ABAC (정책의 태그)	예

속성 기반 액세스 제어 (ABAC) 는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. In AWS, 이러한 속성을 태그라고 합니다. IAM엔티티 (사용자 또는 역할) 및 여러 엔티티에 태그를 첨부할 수 있

습니다. AWS 있습니다. 의 ABAC 첫 번째 단계는 엔티티와 리소스에 태그를 지정하는 것입니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC빠르게 성장하는 환경에서 유용하며 정책 관리가 복잡해지는 상황에도 도움이 됩니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

에 대한 자세한 내용은 What is ABAC? 를 참조하십시오. ABAC IAM사용 설명서에서 설정 ABAC 단계가 포함된 자습서를 보려면 사용 IAM설명서의 속성 기반 액세스 제어 사용 (ABAC) 을 참조하십시오.

서버리스에서 임시 자격 증명 사용 EMR

임시 자격 증명 지원: 예

약간 AWS 서비스 임시 자격 증명을 사용하여 로그인할 때는 작동하지 않습니다. 다음을 포함한 추가 정보는 다음과 같습니다. AWS 서비스 임시 자격 증명으로 작업하려면 다음을 참조하십시오. AWS 서비스IAM사용 IAM 설명서에서 함께 사용할 수 있습니다.

에 로그인하면 임시 자격 증명을 사용하는 것입니다. AWS Management Console 사용자 이름과 암호를 제외한 모든 방법을 사용합니다. 예를 들어, 액세스할 때 AWS 회사의 Single Sign-On (SSO) 링크를 사용하면 이 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM사용 설명서의 역할 전환 (콘솔) 을 참조하십시오.

를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다. AWS CLI 또는 AWS API. 그러면 해당 임시 자격 증명을 사용하여 액세스할 수 있습니다. AWS. AWS 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성할 것을 권장합니다. 자세한 내용은 의 임시 보안 자격 증명을 참조하십시오. IAM

서버리스의 서비스 간 보안 주체 권한 EMR

순방향 액세스 세션 지원 (FAS): 예

IAM사용자 또는 역할을 사용하여 작업을 수행하는 경우 AWS, 귀하는 주도자로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS전화를 건 주체의 권한을 사용합니다. AWS 서비스, 요청과 결합 AWS 서비스 다운스트림 서비스에 요청하기. FAS요청은 서비스가 다른 서비스와의 상호 작용이 필요한 요청을 수신할 때만 이루어집니다. AWS 서

비스 또는 완료해야 할 리소스. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS요청 시 적용되는 정책 세부 정보는 전달 액세스 세션을 참조하십시오.

EMR서버리스의 서비스 역할

서비스 역할 지원	아니요

서버리스의 서비스 연결 역할 EMR

서비스 링크 역할 지원	예

서비스 연결 역할을 만들거나 관리하는 방법에 대한 자세한 내용은 을 참조하십시오. AWS 함께 작동하는 서비스. IAM 서비스 연결 역할 열에서 Yes(이)가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

서버리스용 서비스 연결 역할 사용 EMR

Amazon EMR 서버리스 사용 AWS Identity and Access Management (IAM) 서비스 연결 역할. 서비스 연결 역할은 서버리스에 직접 연결되는 고유한 IAM 역할 유형입니다. EMR 서비스 연결 역할은 EMR 서버리스에서 미리 정의하며 서비스가 다른 역할을 호출하는 데 필요한 모든 권한을 포함합니다. AWS 사용자를 대신하여 서비스를 제공합니다.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 EMR 서버리스를 더 쉽게 설정할 수 있습니다. EMR서버리스는 서비스 연결 역할의 권한을 정의하며, 달리 정의되지 않는 한 서버리스만 EMR 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 엔티티에 연결할 수 없습니다. IAM

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스 액세스 권한을 실수로 제거할 수 없으므로 EMR 서버리스 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 다른 서비스에 대한 자세한 내용은 을 참조하십시오. AWS 함께 작동하는 서비스에서 서비스 IAM 연결 역할 열에서 '예'라고 표시된 서비스를 찾아보세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

서버리스의 서비스 연결 역할 권한 EMR

EMR서버리스는 이름이 지정된 서비스 연결 역할을 사용하여 호출할 수 있도록 합니다. AWSServiceRoleForAmazonEMRServerless AWS APIs여러분을 대신해서 말이죠.

AWSServiceRoleForAmazonEMRServerless 서비스 연결 역할은 다음 서비스가 역할을 맡을 것으로 신뢰합니다.

- `ops.emr-serverless.amazonaws.com`

이름이 지정된 역할 권한 정책을 `AmazonEMRServerlessServiceRolePolicy` 통해 EMR 서버리스는 지정된 리소스에서 다음 작업을 완료할 수 있습니다.

> **Note**
>
> 관리형 정책 내용이 변경되므로 여기에 표시된 정책이 최신 상태가 아닐 수 있습니다. 에서 가장 많은 up-to-date 정책 A를 mazonEMRServerless ServiceRolePolicy 확인하세요. AWS Management Console.

- 작업: `ec2:CreateNetworkInterface`
- 작업: `ec2:DeleteNetworkInterface`
- 작업: `ec2:DescribeNetworkInterfaces`
- 작업: `ec2:DescribeSecurityGroups`
- 작업: `ec2:DescribeSubnets`
- 작업: `ec2:DescribeVpcs`
- 작업: `ec2:DescribeDhcpOptions`
- 작업: `ec2:DescribeRouteTables`
- 작업: `cloudwatch:PutMetricData`

다음은 전체 `AmazonEMRServerlessServiceRolePolicy` 정책입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EC2PolicyStatement",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
```

```
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeRouteTables"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchPolicyStatement",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": [
                        "AWS/EMRServerless",
                        "AWS/Usage"
                    ]
                }
            }
        }
    ]
}
```

EMR서버리스 보안 주체가 이 역할을 맡을 수 있도록 하기 위해 다음과 같은 신뢰 정책이 이 역할에 첨부되었습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ops.emr-serverless.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
```

```
        }
    ]
}
```

IAM엔티티 (예: 사용자, 그룹 또는 역할) 가 서비스 연결 역할을 생성, 편집 또는 삭제할 수 있도록 권한을 구성해야 합니다. 자세한 내용은 사용 설명서의 서비스 연결 역할 권한을 참조하십시오. IAM

서버리스용 서비스 연결 역할 생성 EMR

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. 에서 새 EMR 서버리스 애플리케이션을 만드는 경우 AWS Management Console (EMRStudio 사용), AWS CLI, 또는 AWS API, EMR 서버리스는 사용자를 대신하여 서비스 연결 역할을 생성합니다. IAM엔티티 (예: 사용자, 그룹 또는 역할) 가 서비스 연결 역할을 생성, 편집 또는 삭제할 수 있도록 권한을 구성해야 합니다.

를 사용하여 AWSServiceRoleForAmazonEMRServerless 서비스 연결 역할을 만들려면 IAM

서비스 연결 역할을 만들어야 하는 IAM 엔티티에 대한 권한 정책에 다음 설명을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 새 EMR 서버리스 애플리케이션을 만들면 서버리스가 서비스 EMR 연결 역할을 다시 생성합니다.

IAM콘솔을 사용하여 서버리스 사용 사례와 함께 서비스 연결 역할을 만들 수도 있습니다. EMR 에서 AWS CLI 또는 AWS API, 서비스 이름을 사용하여 서비스 연결 역할을 생성합니다. `ops.emr-serverless.amazonaws.com` 자세한 내용은 사용 설명서의 서비스 연결 역할 만들기를 참조하십시오. IAM 이 서비스 연결 역할을 삭제하면 동일한 프로세스를 사용하여 역할을 다시 생성할 수 있습니다.

서버리스의 서비스 연결 역할 편집 EMR

EMR다양한 엔티티가 역할을 참조할 수 있으므로 서버리스에서는 AWSServiceRoleForAmazonEMRServerless 서비스 연결 역할을 편집할 수 없습니다. 는 편집할 수 없습니다. AWS EMR서버리스 서비스 연결 역할이 사용하는 -소유 IAM 정책입니다. 이 정책에는 서버리스에 필요한 모든 권한이 포함되어 있기 때문입니다. EMR 하지만 를 사용하여 역할에 대한 설명을 편집할 수 있습니다. IAM

를 사용하여 AWSServiceRoleForAmazonEMRServerless 서비스 연결 역할의 설명을 편집하려면 IAM

서비스 연결 역할의 설명을 편집해야 하는 IAM 엔티티에 대한 권한 정책에 다음 설명을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "iam: UpdateRoleDescription"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

자세한 내용은 사용 설명서의 서비스 연결 역할 편집을 참조하십시오. IAM

서버리스의 서비스 연결 역할 삭제 EMR

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 이렇게 하면 적극적으로 모니터링되거나 유지 관리되지 않는 미사용 엔티티가 발생하지 않습니다. 하지만 서비스 연결 역할을 삭제하려면 먼저 모든 지역의 모든 EMR 서버리스 애플리케이션을 삭제해야 합니다.

> **Note**
>
> 역할과 관련된 리소스를 삭제하려고 할 때 EMR 서버리스 서비스가 역할을 사용하는 경우 삭제가 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

를 사용하여 AWSServiceRoleForAmazonEMRServerless 서비스 연결 역할을 삭제하려면 IAM

서비스 연결 역할을 삭제해야 하는 IAM 엔티티에 대한 권한 정책에 다음 설명을 추가합니다.

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/ops.emr-serverless.amazonaws.com/
AWSServiceRoleForAmazonEMRServerless*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "ops.emr-
serverless.amazonaws.com"}}
}
```

를 사용하여 서비스 연결 역할을 수동으로 삭제하려면 IAM

IAM콘솔을 사용하여 AWS CLI, 또는 AWS API AWSServiceRoleForAmazonEMRServerless 서비스 연결 역할을 삭제하려면 자세한 내용은 사용 설명서의 서비스 연결 역할 삭제를 참조하십시오. IAM

EMR서버리스 서비스 연결 역할이 지원되는 지역

EMR서버리스는 서비스를 사용할 수 있는 모든 지역에서 서비스 연결 역할을 사용할 수 있도록 지원합니다. 자세한 내용은 단원을 참조하세요.AWS 리전 및 엔드포인트.

Amazon EMR 서버리스의 Job 런타임 역할

EMR서버리스 작업 실행이 사용자 대신 다른 서비스를 호출할 때 맡을 수 있는 IAM 역할 권한을 지정할 수 있습니다. 여기에는 모든 데이터 소스, 대상 및 기타 데이터에 대한 Amazon S3에 대한 액세스가 포함됩니다. AWS Amazon Redshift 클러스터 및 DynamoDB 테이블과 같은 리소스. 역할을 생성하는 방법에 대한 자세한 내용은 을 참조하십시오. 작업 런타임 역할 생성

샘플 런타임 정책

다음과 같은 런타임 정책을 작업 런타임 역할에 연결할 수 있습니다. 다음 작업 런타임 정책은 다음을 허용합니다.

- EMR샘플이 포함된 Amazon S3 버킷에 대한 읽기 액세스.
- S3 버킷에 대한 전체 액세스.
- 액세스 생성 및 읽기 권한 AWS Glue 데이터 카탈로그.

다른 사용자에게 액세스 권한을 추가하려면 AWS DynamoDB와 같은 리소스의 경우 런타임 역할을 생성할 때 정책에 해당 리소스에 대한 권한을 포함해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadAccessForEMRSamples",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::*.elasticmapreduce",
                "arn:aws:s3:::*.elasticmapreduce/*"
            ]
        },
        {
            "Sid": "FullAccessToS3Bucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        },
        {
            "Sid": "GlueCreateAndReadDataCatalog",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:CreateDatabase",
                "glue:GetDataBases",
                "glue:CreateTable",
                "glue:GetTable",
                "glue:UpdateTable",
                "glue:DeleteTable",
```

```
                "glue:GetTables",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetUserDefinedFunctions"
            ],
            "Resource": ["*"]
        }
    ]
}
```

역할 권한 전달

IAM권한 정책을 사용자 역할에 연결하여 사용자가 승인된 역할만 전달하도록 허용할 수 있습니다. 이를 통해 관리자는 특정 작업 런타임 역할을 EMR 서버리스 작업에 전달할 수 있는 사용자를 제어할 수 있습니다. 권한 설정에 대한 자세한 내용은 사용자에게 역할을 전달할 수 있는 권한 부여를 참조하십시오. AWS 서비스.

다음은 EMR 서버리스 서비스 보안 주체에 작업 런타임 역할을 전달할 수 있는 예제 정책입니다.

```
{
     "Effect": "Allow",
     "Action": "iam:PassRole",
     "Resource": "arn:aws:iam::1234567890:role/JobRuntimeRoleForEMRServerless",
        "Condition": {
                "StringLike": {
                    "iam:PassedToService": "emr-serverless.amazonaws.com"
                }
            }
}
```

서버리스의 사용자 액세스 정책 예제 EMR

서버리스 애플리케이션과 상호 작용할 때 각 사용자가 수행하기를 원하는 작업에 따라 사용자에 대한 세분화된 정책을 설정할 수 있습니다. EMR 다음 정책은 사용자에게 적합한 권한을 설정하는 데 도움이 될 수 있는 예시입니다. 이 섹션에서는 EMR 서버리스 정책에만 초점을 맞춥니다. EMRStudio 사용자 정책 샘플은 EMRStudio 사용자 권한 구성을 참조하십시오. 정책을 IAM 사용자 (주체) 에게 연결하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM정책 관리를 참조하십시오.

파워 유저 정책

EMR서버리스에 필요한 모든 작업을 허용하려면 AmazonEMRServerlessFullAccess 정책을 생성하여 필요한 IAM 사용자, 역할 또는 그룹에 연결하십시오.

다음은 고급 사용자가 EMR 서버리스 애플리케이션을 생성 및 수정하고 작업 제출 및 디버깅과 같은 기타 작업을 수행할 수 있도록 허용하는 샘플 정책입니다. 여기에는 EMR 서버리스가 다른 서비스에 필요한 모든 작업이 표시됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessActions",
            "Effect": "Allow",
            "Action": [
                "emr-serverless:CreateApplication",
                "emr-serverless:UpdateApplication",
                "emr-serverless:DeleteApplication",
                "emr-serverless:ListApplications",
                "emr-serverless:GetApplication",
                "emr-serverless:StartApplication",
                "emr-serverless:StopApplication",
                "emr-serverless:StartJobRun",
                "emr-serverless:CancelJobRun",
                "emr-serverless:ListJobRuns",
                "emr-serverless:GetJobRun"
            ],
            "Resource": "*"
        }
    ]
}
```

네트워크 연결을 활성화하면 EMR 서버리스 애플리케이션이 Amazon EC2 엘라스틱 네트워크 인터페이스 (ENIs) 를 생성하여 VPC 리소스와 통신합니다. VPC 다음 정책은 모든 새 EC2 ENIs 애플리케이션이 EMR 서버리스 애플리케이션 컨텍스트에서만 생성되도록 합니다.

> **Note**
>
> EMR서버리스 애플리케이션을 시작하는 EC2 ENIs 경우를 제외하고는 사용자가 생성할 수 없도록 이 정책을 설정하는 것이 좋습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEC2ENICreationWithEMRTags",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "ops.emr-serverless.amazonaws.com"
                }
            }
        }
}
```

EMR서버리스 액세스를 특정 서브넷으로 제한하려면 각 서브넷에 태그 조건을 지정하면 됩니다. 이 IAM 정책을 통해 EMR 서버리스 애플리케이션은 허용된 서브넷 내에서만 생성할 EC2 ENIs 수 있습니다.

```
{
    "Sid": "AllowEC2ENICreationInSubnetAndSecurityGroupWithEMRTags",
    "Effect": "Allow",
    "Action": [
        "ec2:CreateNetworkInterface"
    ],
    "Resource": [
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/KEY": "VALUE"
        }
    }
}
```

> **Important**
>
> 첫 번째 응용 프로그램을 만드는 관리자 또는 고급 사용자인 경우 EMR 서버리스 서비스 연결 역할을 만들 수 있도록 권한 정책을 구성해야 합니다. 자세한 내용은 서버리스용 서비스 연결 역할 사용 EMR을 참조하십시오.

다음 IAM 정책에 따라 계정에 EMR 서버리스 서비스 연결 역할을 만들 수 있습니다.

```
{
   "Sid":"AllowEMRServerlessServiceLinkedRoleCreation",
   "Effect":"Allow",
   "Action":"iam:CreateServiceLinkedRole",
   "Resource":"arn:aws:iam::account-id:role/aws-service-role/ops.emr-
serverless.amazonaws.com/AWSServiceRoleForAmazonEMRServerless"
}
```

데이터 엔지니어 정책

다음은 사용자에게 EMR 서버리스 애플리케이션에 대한 읽기 전용 권한과 작업 제출 및 디버그 기능을 허용하는 샘플 정책입니다. 이 정책은 작업을 명시적으로 거부하지 않기 때문에 다른 정책 설명을 사용하더라도 지정된 작업에 대한 액세스를 허용할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessActions",
            "Effect": "Allow",
            "Action": [
                "emr-serverless:ListApplications",
                "emr-serverless:GetApplication",
                "emr-serverless:StartApplication",
                "emr-serverless:StartJobRun",
                "emr-serverless:CancelJobRun",
                "emr-serverless:ListJobRuns",
                "emr-serverless:GetJobRun"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

액세스 제어에 태그 사용

세분화된 액세스 제어를 위해 태그 조건을 사용할 수 있습니다. 예를 들어 팀 이름이 태그가 지정된 EMR 서버리스 애플리케이션에만 작업을 제출할 수 있도록 한 팀의 사용자를 제한할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EMRServerlessActions",
            "Effect": "Allow",
            "Action": [
                "emr-serverless:ListApplications",
                "emr-serverless:GetApplication",
                "emr-serverless:StartApplication",
                "emr-serverless:StartJobRun",
                "emr-serverless:CancelJobRun",
                "emr-serverless:ListJobRuns",
                "emr-serverless:GetJobRun"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Team": "team-name"
                }
            }
        }
    ]
}
```

태그 기반 액세스 제어를 위한 정책

ID 기반 정책의 조건을 사용하여 태그를 기반으로 애플리케이션 및 작업 실행에 대한 액세스를 제어할 수 있습니다.

다음 예는 EMR 서버리스 조건 키와 함께 조건 연산자를 사용하는 다양한 시나리오와 방법을 보여줍니다. 이러한 IAM 정책 설명은 데모용이며 프로덕션 환경에서는 사용해서는 안 됩니다. 다양한 방법으로 정책 명령문을 결합하여 요구 사항에 따라 권한을 부여하거나 거부할 수 있습니다. 계획 및 테스트 IAM 정책에 대한 자세한 내용은 IAM사용 설명서를 참조하십시오.

> **Important**
>
> 태깅 작업에 대한 권한을 명시적으로 거부하는 것은 중요한 고려 사항입니다. 이렇게 하면 사용자가 리소스에 태그를 지정하지 못하므로, 부여할 의도가 없는 권한이 부여되지 않도록 방지할 수 있습니다. 리소스에 대한 태그 지정 작업이 거부되지 않는 경우 사용자는 태그를 수정하여 태그 기반 정책의 의도를 우회할 수 있습니다. 태그 지정 작업을 거부하는 정책의 예제는 태그를 추가 또는 제거하는 액세스 거부 섹션을 참조하세요.

아래 예는 EMR 서버리스 애플리케이션에서 허용되는 작업을 제어하는 데 사용되는 ID 기반 권한 정책을 보여줍니다.

특정 태그 값이 있는 리소스에서만 작업 허용

다음 정책 예제에서는 `StringEquals` 조건 연산자가 태그 부서의 값과 `dev` 일치시키려고 합니다. 태그 부서가 애플리케이션에 추가되지 않았거나 값을 `dev` 포함하지 않는 경우 정책이 적용되지 않으며 이 정책에서는 해당 작업을 허용하지 않습니다. 작업을 허용하는 다른 정책 설명이 없는 경우 사용자는 이 값을 가진 이 태그가 있는 응용 프로그램만 사용할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:GetApplication"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

또한 조건 연산자를 사용하여 여러 태그 값을 지정할 수 있습니다. 예를 들어 `department` 태그에 값이 `dev` 포함된 애플리케이션에서 작업을 허용하려면 이전 예제의 조건 블록을 다음과 같이 바꿀 수 있습니다. `test`

```
"Condition": {
        "StringEquals": {
          "emr-serverless:ResourceTag/department": ["dev", "test"]
        }
      }
```

리소스 생성 시 태그 지정이 필요함

아래 예제에서는 응용 프로그램을 만들 때 태그를 적용해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "emr-serverless:RequestTag/department": "dev"
        }
      }
    }
  ]
}
```

다음 정책 설명에서는 애플리케이션에 어떤 값이든 포함할 수 있는 department 태그가 있는 경우에만 사용자가 애플리케이션을 생성할 수 있도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-serverless:CreateApplication"
      ],
      "Resource": "*",
      "Condition": {
        "Null": {
```

```
          "emr-serverless:RequestTag/department": "false"
        }
      }
    }
  ]
}
```

태그를 추가 또는 제거하는 액세스 거부

이 정책은 값이 아닌 dev 태그가 있는 EMR 서버리스 애플리케이션에서 department 태그를 추가하거나 제거하는 것을 사용자가 금지합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-serverless:TagResource",
        "emr-serverless:UntagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "emr-serverless:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```

서버리스의 ID 기반 정책 예제 EMR

기본적으로 사용자와 역할에는 Amazon EMR Serverless 리소스를 만들거나 수정할 권한이 없습니다. 또한 다음을 사용하여 작업을 수행할 수도 없습니다. AWS Management Console, AWS Command Line Interface (AWS CLI), 또는 AWS API. 사용자에게 필요한 리소스에서 작업을 수행할 수 있는 권한을 부여하기 위해 IAM 관리자는 IAM 정책을 생성할 수 있습니다. 그러면 관리자가 역할에 IAM 정책을 추가할 수 있으며, 사용자는 역할을 수임할 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 만드는 방법을 알아보려면 사용 IAM설명서에서 IAM 정책 생성을 참조하십시오.

각 리소스 유형의 형식을 비롯하여 Amazon EMR Serverless에서 정의한 작업 및 리소스 유형에 ARNs 대한 자세한 내용은 서비스 인증 참조의 Amazon EMR Serverless용 작업, 리소스 및 조건 키를 참조하십시오.

주제

- 정책 모범 사례
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

> **Note**
>
> EMR서버리스는 관리형 정책을 지원하지 않으므로 아래 나열된 첫 번째 관행은 적용되지 않습니다.

ID 기반 정책은 누군가가 사용자 계정에서 Amazon EMR Serverless 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부를 결정합니다. 이러한 조치로 인해 비용이 발생할 수 있습니다. AWS 계정. ID 기반 정책을 만들거나 편집할 때는 다음 지침 및 권장 사항을 따르십시오.

- 시작해 보세요. AWS 관리형 정책 및 최소 권한 권한으로의 이동 — 사용자와 워크로드에 권한 부여를 시작하려면 다음을 사용하십시오. AWS 여러 일반 사용 사례에 대한 권한을 부여하는 관리형 정책. 다음 사이트에서 사용할 수 있습니다. AWS 계정. 를 정의하여 권한을 더 줄이는 것이 좋습니다. AWS 사용 사례에 맞는 고객 관리형 정책. 자세한 내용은 단원을 참조하세요.AWS 관리형 정책 또는 AWSIAM사용자 가이드의 직무 관리 정책
- 최소 권한 적용 — IAM 정책으로 권한을 설정하는 경우 작업 수행에 필요한 권한만 부여하십시오. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. 를 사용하여 권한을 IAM 적용하는 방법에 대한 자세한 내용은 사용 설명서의 정책 및 권한을 참조하십시오. IAM IAM
- IAM정책의 조건을 사용하여 액세스를 추가로 제한 - 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, 를 사용하여 모든 요청을 전송하도록 지정하는 정책 조건을 작성할 수 SSL 있습니다. 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. AWS 서비스예: AWS CloudFormation. 자세한 내용은 IAM사용 설명서의 IAMJSON정책 요소: 조건을 참조하십시오.
- IAMAccess Analyzer를 사용하여 IAM 정책을 검증하여 안전하고 기능적인 권한을 보장합니다. IAM Access Analyzer는 새 정책과 기존 정책을 검증하여 정책이 IAM 정책 언어 (JSON) 및 IAM 모범 사

례를 준수하는지 확인합니다. IAMAccess Analyzer는 안전하고 기능적인 정책을 작성하는 데 도움이 되는 100개 이상의 정책 검사와 실행 가능한 권장 사항을 제공합니다. 자세한 내용은 사용 설명서의 IAMAccess Analyzer 정책 검증을 참조하십시오. IAM

- 다단계 인증 필요 (MFA) - 사용자 또는 루트 IAM 사용자가 필요한 시나리오가 있는 경우 AWS 계정 보안을 강화하려면 MFA 켜십시오. API작업 호출 MFA 시기를 요구하려면 정책에 MFA 조건을 추가하세요. 자세한 내용은 IAM사용 설명서의 MFA -보호된 API 액세스 구성을 참조하십시오.

의 모범 사례에 IAM 대한 자세한 내용은 IAM사용 설명서의 보안 모범 사례를 참조하십시오. IAM

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 하는 정책을 만드는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 다음을 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다. AWS CLI 또는 AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
```

```
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon EMR 서버리스 업데이트 AWS 관리형 정책

업데이트에 대한 세부 정보 보기 AWS 이 서비스가 이러한 변경 사항을 추적하기 시작한 이후 Amazon EMR Serverless의 정책을 관리했습니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 Amazon EMR Serverless Document 기록 페이지에서 RSS 피드를 구독하십시오.

변경 사항	설명	날짜
A mazonEMRServerless ServiceRolePolicy — 기존 정책 업데이트	Amazon EMR 서버리스는 A mazonEMRServerless ServiceRolePolicy 정책에 새 Sid CloudWatchPolicySt atement 및 EC2Policy Statement 를 추가했습니다.	2024년 1월 25일
A mazonEMRServerless ServiceRolePolicy — 기존 정책 업데이트	Amazon EMR Serverless는 Amazon EMR Serverless가 네임스페이스의 v CPU 사용량에 대한 집계된 계정 지표를 게시할 수 있는 새로운 권한을 추가했습니다. "AWS/Usage"	2023년 4월 20일
Amazon EMR 서버리스가 변경 사항 추적을 시작했습니다.	Amazon EMR 서버리스는 자사의 변경 사항을 추적하기 시작했습니다. AWS 관리형 정책.	2023년 4월 20일

Amazon EMR 서버리스 ID 및 액세스 문제 해결

다음 정보를 사용하면 Amazon EMR Serverless 및 에서 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 IAM 해결하는 데 도움이 됩니다.

주제

Amazon EMR 서버리스에서 작업을 수행할 권한이 없습니다.

만약 AWS Management Console 작업을 수행할 권한이 없다는 메시지가 표시되면 관리자에게 도움을 요청해야 합니다. 관리자는 사용자 이름과 비밀번호를 제공한 사람입니다.

다음 예제 오류는 `mateojackson` 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `emr-serverless:`*`GetWidget`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
serverless:GetWidget on resource: my-example-widget
```

이 경우 Mateo는 *`my-example-widget`* 작업을 사용하여 `emr-serverless:`*`GetWidget`* 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

저는 IAM을 수행할 권한이 없습니다. PassRole

작업을 수행할 권한이 없다는 오류가 발생하는 경우 Amazon EMR Serverless에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다. `iam:PassRole`

약간 AWS 서비스 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 라는 IAM 사용자가 Amazon EMR Serverless에서 콘솔을 사용하여 작업을 `marymajor` 수행하려고 할 때 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole` 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 다음 연락처로 문의하십시오. AWS 관리자에게. 관리자는 로그인 자격 증명을 제공한 사람입니다.

제 외부 사람들을 허용하고 싶어요. AWS 내 Amazon EMR 서버리스 리소스에 액세스하기 위한 계정

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록 (ACLs) 을 지원하는 서비스의 경우 해당 정책을 사용하여 사용자에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- Amazon EMR Serverless에서 이러한 기능을 지원하는지 알아보려면 을 참조하십시오Amazon EMR 서버리스의 ID 및 Access Management (IAM).
- 여러 지역의 리소스에 대한 액세스를 제공하는 방법을 알아보려면 AWS 계정 소유한 사용자는 다른 IAM 사용자에게 액세스 권한 제공을 참조하십시오. AWS 계정IAM사용 설명서에 있는 소유권
- 리소스에 대한 액세스 권한을 제3자에게 제공하는 방법을 알아보려면 AWS 계정액세스 제공을 참조하십시오. AWS 계정IAM사용 설명서의 제3자가 소유합니다.
- ID 페더레이션을 통해 액세스를 제공하는 방법을 알아보려면 사용 설명서의 외부 인증된 사용자에게 액세스 제공 (ID 페더레이션) 을 IAM 참조하십시오.
- 계정 간 액세스에 대한 역할 사용과 리소스 기반 정책의 차이점을 알아보려면 사용 설명서의 계정 간 리소스 액세스를 참조하십시오. IAM IAM

다음과 함께 EMR 서버리스 사용 AWS Lake Formation 세밀한 액세스 제어용

개요

Amazon EMR 릴리스 7.2.0 이상에서는 다음을 활용할 수 있습니다. AWS Lake Formation S3가 지원하는 데이터 카탈로그 테이블에 세밀한 액세스 제어를 적용하기 위함입니다. 이 기능을 사용하면 테이블, 행, 열 및 셀 수준의 액세스 제어를 구성할 수 있습니다.read Amazon EMR 서버리스 Spark 작업 내

의 쿼리. Apache Spark 배치 작업 및 대화형 세션에 대한 세분화된 액세스 제어를 구성하려면 Studio를 사용하십시오. EMR Lake Formation 및 이를 EMR 서버리스와 함께 사용하는 방법에 대한 자세한 내용은 다음 섹션을 참조하십시오.

Amazon EMR 서버리스를 다음과 함께 사용하기 AWS Lake Formation 추가 요금이 발생합니다. 자세한 내용은 Amazon EMR 요금을 참조하십시오.

EMR서버리스의 작동 방식 AWS Lake Formation

Lake Formation과 함께 EMR 서버리스를 사용하면 각 Spark 작업에 권한 계층을 적용하여 EMR 서버리스가 작업을 실행할 때 Lake Formation 권한 제어를 적용할 수 있습니다. EMR서버리스는 Spark 리소스 프로필을 사용하여 두 개의 프로필을 생성하여 작업을 효과적으로 실행합니다. 사용자 프로필은 사용자 제공 코드를 실행하는 반면, 시스템 프로필은 Lake Formation 정책을 적용합니다. 자세한 내용은 무엇입니까를 참조하십시오. AWS Lake Formation및 고려 사항 및 제한 사항.

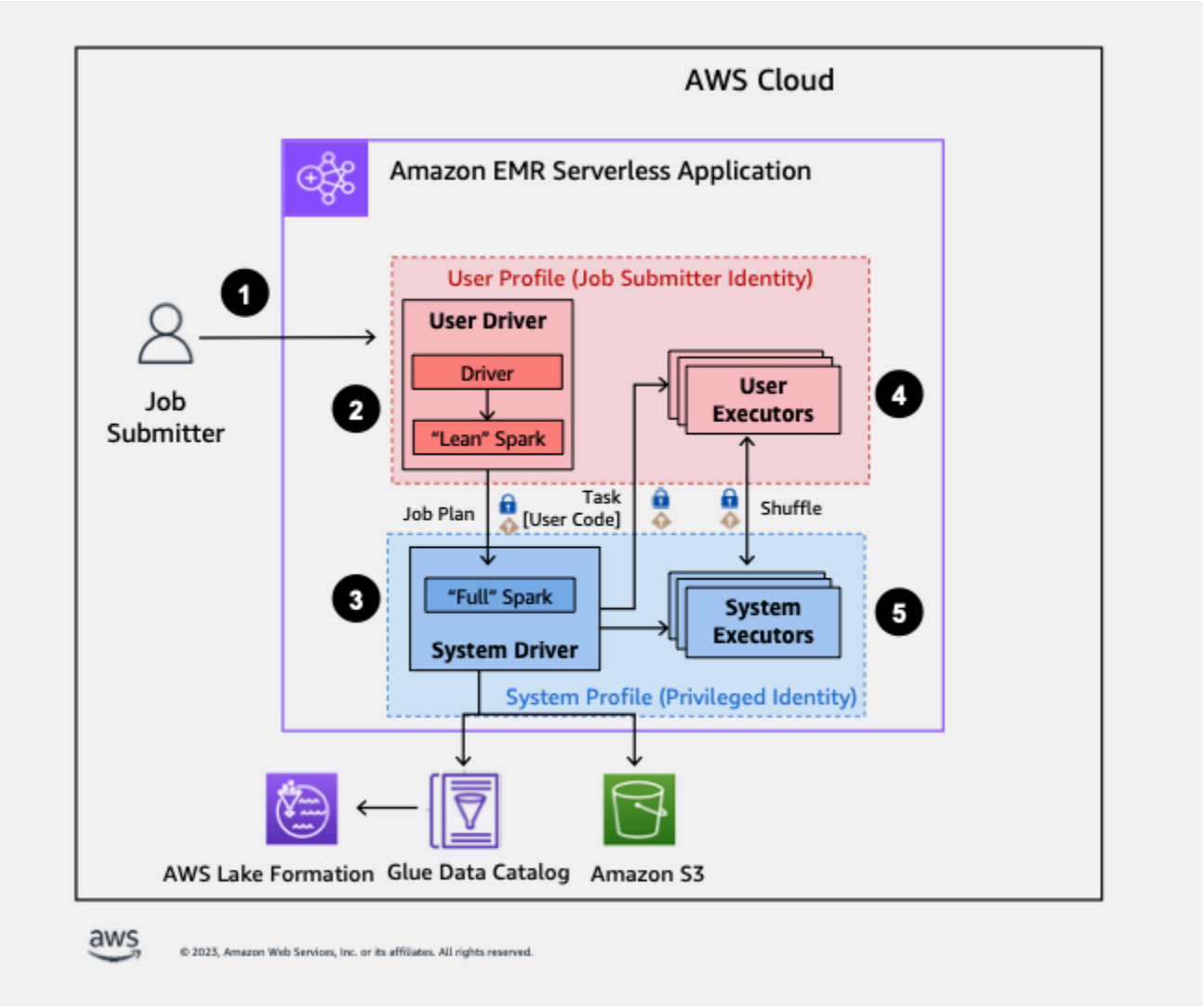
Lake Formation에서 사전 초기화된 용량을 사용하는 경우 최소 두 개의 Spark 드라이버를 사용하는 것이 좋습니다. Lake Formation이 활성화된 각 작업에는 두 개의 Spark 드라이버가 사용됩니다. 하나는 사용자 프로필용이고 다른 하나는 시스템 프로필용입니다. 최상의 성능을 얻으려면 Lake Formation을 사용하는 작업에는 Lake Formation을 사용하지 않는 경우와 비교하여 두 배의 드라이버를 사용해야 합니다.

EMR서버리스에서 Spark 작업을 실행할 때는 동적 할당이 리소스 관리 및 클러스터 성능에 미치는 영향도 고려해야 합니다. 리소스 프로필당 최대 실행자 수 구성은 `spark.dynamicAllocation.maxExecutors` 사용자 및 시스템 실행자 모두에 적용됩니다. 이 수를 허용되는 최대 실행자 수와 같도록 구성하면 사용 가능한 리소스를 모두 사용하는 한 유형의 실행기로 인해 작업 실행이 중단되어 작업 실행 시 다른 실행자가 작업을 실행할 수 없게 될 수 있습니다.

리소스가 부족하지 않도록 EMR 서버리스는 리소스 프로필당 기본 최대 실행자 수를 값의 90% 로 설정합니다. `spark.dynamicAllocation.maxExecutors` 0에서 1 사이의 `spark.dynamicAllocation.maxExecutorsRatio` 값으로 지정할 경우 이 구성을 재정의할 수 있습니다. 또한 다음 속성을 구성하여 리소스 할당 및 전체 성능을 최적화할 수도 있습니다.

- `spark.dynamicAllocation.cachedExecutorIdleTimeout`
- `spark.dynamicAllocation.shuffleTracking.timeout`
- `spark.cleaner.periodicGC.interval`

다음은 EMR 서버리스가 Lake Formation 보안 정책으로 보호되는 데이터에 액세스하는 방법에 대한 개괄적인 개요입니다.



1. 사용자가 Spark 작업을 다음 주소로 제출합니다. AWS Lake Formation-지원 서버리스 애플리케이션EMR.
2. EMR서버리스는 사용자 드라이버에 작업을 보내고 사용자 프로필에서 작업을 실행합니다. 사용자 드라이버는 작업을 시작하거나, 실행기를 요청하거나, S3 또는 Glue Catalog에 액세스할 수 없는 린 버전의 Spark를 실행합니다. 작업 계획을 세웁니다.
3. EMR서버리스는 시스템 드라이버라는 두 번째 드라이버를 설정하고 시스템 프로필에서 권한이 있는 ID를 사용하여 이를 실행합니다. EMR서버리스는 통신을 위해 두 드라이버 사이에 암호화된 TLS 채널을 설정합니다. 사용자 드라이버는 채널을 사용하여 작업 계획을 시스템 드라이버에 보냅니다. 시스템 드라이버는 사용자가 제출한 코드를 실행하지 않습니다. Spark를 완전히 실행하고 S3 및 데이터 카탈로그와 통신하여 데이터에 액세스합니다. 실행자를 요청하고 Job Plan을 일련의 실행 단계로 컴파일합니다.

4. EMR그런 다음 서버리스는 사용자 드라이버 또는 시스템 드라이버를 사용하여 실행기에서 스테이지를 실행합니다. 모든 단계의 사용자 코드는 사용자 프로필 실행자에서만 실행됩니다.
5. 로 보호되는 데이터 카탈로그 테이블에서 데이터를 읽는 스테이지 AWS Lake Formation 또는 보안 필터를 적용하는 시스템은 시스템 실행자에게 위임됩니다.

아마존에서 Lake Formation 활성화 EMR

Lake Formation을 활성화하려면 EMR서버리스 애플리케이션을 생성할 때 런타임 구성 매개변수에 대해 `true` `spark-defaults` 언더분류를 `spark.emr-serverless.lakeformation.enabled` 설정해야 합니다.

```
aws emr-serverless create-application \
    --release-label emr-7.2.0 \
    --runtime-configuration '{
     "classification": "spark-defaults",
     "properties": {
      "spark.emr-serverless.lakeformation.enabled": "true"
      }
    }' \
    --type "SPARK"
```

EMRStudio에서 새 애플리케이션을 만들 때 Lake Formation을 활성화할 수도 있습니다. 추가 구성에서 사용할 수 있는 세분화된 액세스 제어를 위해 Use Lake Formation을 선택하십시오.

Lake Formation을 EMR Serverless와 함께 사용하면 작업자 간 암호화가 기본적으로 활성화되므로 작업자 간 암호화를 명시적으로 다시 활성화하지 않아도 됩니다.

Spark 작업을 위한 Lake Formation 지원

개별 Spark 작업에 대해 Lake Formation을 `spark.emr-serverless.lakeformation.enabled` 활성화하려면 사용할 `spark-submit` 때 true로 설정하십시오.

```
--conf spark.emr-serverless.lakeformation.enabled=true
```

Job 런타임 역할 IAM 권한

Lake Formation 권한은 다음에 대한 액세스를 제어합니다. AWS Glue 데이터 카탈로그 리소스, Amazon S3 위치 및 해당 위치의 기본 데이터. IAM권한은 Lake Formation에 대한 접근을 통제하고

AWS Glue APIs 및 리소스. 데이터 카탈로그 (SELECT) 의 테이블에 액세스할 수 있는 Lake Formation 권한이 있더라도 작업에 대한 IAM 권한이 없으면 `glue:Get*` API 작업이 실패합니다.

다음은 S3에서 스크립트에 액세스할 수 있는 IAM 권한을 제공하고 S3에 로그를 업로드하는 방법에 대한 예제 정책입니다. AWS Glue API 권한 및 Lake Formation 액세스 권한.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ScriptAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
            "arn:aws:s3:::*.DOC-EXAMPLE-BUCKET/scripts",
            "arn:aws:s3:::*.DOC-EXAMPLE-BUCKET/*" ]
        },
        {
         "Sid": "LoggingAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
               "arn:aws:s3:::DOC-EXAMPLE-BUCKET/logs/*"
            ]
        },
        {
            "Sid": "GlueCatalogAccess",
            "Effect": "Allow",
            "Action": [
                 "glue:Get*",
                 "glue:Create*",
                 "glue:Update*"
            ],
            "Resource": ["*"]
        },
        {
            "Sid": "LakeFormationAccess",
            "Effect": "Allow",
```

```
            "Action": [
             "lakeformation:GetDataAccess"
             ],
            "Resource": ["*"]
        }
    ]
}
```

작업 런타임 역할에 대한 Lake Formation 권한 설정

먼저 Hive 테이블의 위치를 Lake Formation에 등록하십시오. 그런 다음 원하는 테이블에 작업 런타임 역할에 대한 권한을 생성합니다. Lake Formation에 대한 자세한 내용은 다음을 참조하십시오. AWS Lake Formation? 에서 AWS Lake Formation 개발자 가이드.

Lake Formation 권한을 설정한 후 Amazon EMR 서버리스에서 Spark 작업을 제출할 수 있습니다. Spark 작업에 대한 자세한 내용은 Spark 예제를 참조하십시오.

작업 실행 제출

Lake Formation 보조금 설정을 완료한 후에는 EMR서버리스에서 Spark 작업을 제출할 수 있습니다. Iceberg 작업을 실행하려면 다음 속성을 제공해야 합니다. `spark-submit`

```
--conf spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkSessionCatalog
--conf spark.sql.catalog.spark_catalog.warehouse=<S3_DATA_LOCATION>
--conf spark.sql.catalog.spark_catalog.glue.account-id=<ACCOUNT_ID>
--conf spark.sql.catalog.spark_catalog.client.region=<REGION>
--conf spark.sql.catalog.spark_catalog.glue.endpoint=https://
glue.<REGION>.amazonaws.com
```

오픈 테이블 형식 지원

Amazon EMR 릴리스 7.2.0에는 Lake Formation을 기반으로 하는 세분화된 액세스 제어에 대한 지원이 포함되어 있습니다. EMR서버리스는 Hive 및 Iceberg 테이블 유형을 지원합니다. 다음 표에는 지원되는 모든 작업이 설명되어 있습니다.

운영	Hive	Iceberg
DDL명령	IAM역할 권한만 있는 경우	IAM역할 권한만 있는 경우
증분 쿼리	해당 사항 없음	완전 지원

운영	Hive	Iceberg
시간 이동 쿼리	이 테이블 형식에는 해당되지 않습니다.	완전 지원
메타데이터 테이블	이 테이블 형식에는 해당되지 않습니다.	지원되지만 일부 테이블은 숨겨집니다. 자세한 내용은 고려 사항 및 제한 사항을 참조하십시오.
DML INSERT	IAM권한이 있는 경우에만	IAM권한이 있는 경우에만
DML UPDATE	이 테이블 형식에는 해당되지 않습니다.	IAM권한만 있는 경우
DML DELETE	이 테이블 형식에는 해당되지 않습니다.	IAM권한만 있는 경우
읽기 작업	완전 지원	완전 지원
저장 프로시저	해당 사항 없음	`register_table` 및 를 제외하고 지원됩니다`migrate`. 자세한 내용은 고려 사항 및 제한 사항을 참조하십시오.

고려 사항 및 제한

EMR서버리스와 함께 Lake Formation을 사용할 때는 다음 고려 사항 및 제한 사항을 고려하십시오.

> **Note**
>
> EMR서버리스에서 Spark 작업에 대해 Lake Formation을 활성화하면 작업에서 시스템 드라이버와 사용자 드라이버가 시작됩니다. 시작 시 사전 초기화된 용량을 지정한 경우 드라이버는 사전 초기화된 용량에서 프로비전되며 시스템 드라이버 수는 지정한 사용자 드라이버 수와 같습니다. 온디맨드 용량을 선택하면 EMR 서버리스는 사용자 드라이버 외에 시스템 드라이버도 시작합니다. Lake Formation을 사용한 EMR 서버리스 작업과 관련된 비용을 추정하려면 다음을 사용하십시오. AWS Pricing Calculator.

Lake EMR Formation 기반 Amazon 서버리스는 다음을 제외하고 지원되는 모든 EMR서버리스 지역에서 사용할 수 있습니다. AWS GovCloud (미국 동부) 및 AWS GovCloud (미국 서부).

- Amazon EMR Serverless는 아파치 하이브 및 아파치 아이스버그 테이블에 대해서만 Lake Formation을 통한 세밀한 액세스 제어를 지원합니다. 아파치 하이브 형식에는 파켓, xSv가 포함됩니다. ORC
- Lake Formation 지원 애플리케이션은 사용자 지정된 EMR 서버리스 이미지 사용을 지원하지 않습니다.
- Lake Formation 작업을 `DynamicResourceAllocation` 중단할 수는 없습니다.
- Lake Formation은 스파크 작업에서만 사용할 수 있습니다.
- EMRLake Formation을 사용하는 서버리스는 작업 전반에 걸쳐 단일 Spark 세션만 지원합니다.
- EMRLake Formation을 사용하는 서버리스는 리소스 링크를 통해 공유되는 계정 간 테이블 쿼리만 지원합니다.
- 다음은 지원되지 않습니다.
  - 복원력이 뛰어난 분산 데이터세트 () RDD
  - 스파크 스트리밍
  - Lake Formation에서 권한을 부여받은 상태에서 글을 쓰세요.
  - 중첩된 열에 대한 액세스 제어
- EMR서버리스는 다음을 포함하여 시스템 드라이버의 완전한 격리를 저해할 수 있는 기능을 차단합니다.
  - UDTsiveUDFs, H 및 사용자 정의 클래스를 포함하는 모든 사용자 정의 함수
  - 사용자 지정 데이터 소스
  - Spark 익스텐션, 커넥터 또는 메타스토어를 위한 추가 jar 제공
  - `ANALYZE TABLE` 명령
- 액세스 제어 `EXPLAIN PLAN` 및 제한된 정보 노출 금지 등의 DDL 작업을 시행하기 `DESCRIBE TABLE` 위함입니다.
- EMR서버리스는 Lake Formation 지원 애플리케이션의 시스템 드라이버 Spark 로그에 대한 액세스를 제한합니다. 시스템 드라이버는 더 많은 액세스 권한을 가지고 실행되므로 시스템 드라이버가 생성하는 이벤트 및 로그에는 민감한 정보가 포함될 수 있습니다. 권한이 없는 사용자나 코드가 이 민감한 데이터에 액세스하는 것을 방지하기 위해 EMR 서버리스는 시스템 드라이버 로그에 대한 액세스를 비활성화했습니다. 문제 해결은 다음 연락처로 문의하십시오. AWS 지원.
- Lake Formation에 테이블 위치를 등록한 경우 데이터 액세스 경로는 EMR 서버리스 작업 런타임 역할에 대한 IAM 권한에 관계없이 Lake Formation에 저장된 자격 증명을 통과합니다. 테이블 위치에

등록된 역할을 잘못 구성한 경우 S3 IAM 권한이 있는 역할을 사용하여 테이블 위치에 제출된 작업은 실패합니다.

- Lake Formation 테이블에 글을 쓸 때는 Lake Formation이 부여한 IAM 권한이 아닌 권한을 사용합니다. 작업 런타임 역할에 필요한 S3 권한이 있는 경우 이 역할을 사용하여 쓰기 작업을 실행할 수 있습니다.

Apache Iceberg를 사용할 때 고려할 사항 및 제한 사항은 다음과 같습니다.

- Apache Iceberg는 세션 카탈로그와 함께 사용할 수 있으며 임의로 명명된 카탈로그는 사용할 수 없습니다.
- Lake Formation에 등록된 Iceberg 테이블은 메타데이터 테이블`history`,, `metadata_log_entries` `snapshots` `filesmanifests`, 및 `refs` 만 지원합니다. Amazon은, `partitionspath`, 등 민감한 데이터가 있을 수 있는 열을 EMR 숨깁니다. `summaries` 이 제한은 Lake Formation에 등록되지 않은 아이스버그 테이블에는 적용되지 않습니다.
- Lake Formation에 등록하지 않은 테이블은 모든 Iceberg 저장 프로시저를 지원합니다. `register_table`및 `migrate` 프로시저는 어떤 테이블에서도 지원되지 않습니다.
- V1 대신 Iceberg DataFrameWriter V2를 사용하는 것이 좋습니다.

문제 해결

문제 해결 방법은 다음 섹션을 참조하십시오.

로깅

EMR서버리스는 Spark 리소스 프로필을 사용하여 작업 실행을 분할합니다. EMR서버리스는 사용자 프로필을 사용하여 제공된 코드를 실행하는 반면, 시스템 프로필은 Lake Formation 정책을 적용합니다. 사용자 프로필로 실행한 작업의 로그에 액세스할 수 있습니다.

라이브 UI 및 Spark 히스토리 서버

Live UI와 Spark 히스토리 서버에는 사용자 프로필에서 생성된 모든 Spark 이벤트와 시스템 드라이버에서 생성된 수정된 이벤트가 있습니다.

Executors 탭에서 사용자 및 시스템 드라이버의 모든 작업을 볼 수 있습니다. 하지만 로그 링크는 사용자 프로필에만 사용할 수 있습니다. 또한 출력 레코드 수와 같은 일부 정보가 Live UI에서 삭제됩니다.

Lake Formation 권한이 충분하지 않아 작업이 실패했습니다.

작업 런타임 역할에 액세스 DESCRIBE 중인 테이블에서 실행할 SELECT 수 있는 권한이 있는지 확인하십시오.

Job (RDD실행 실패)

EMR서버리스는 현재 Lake Formation을 지원하는 작업에서 복원력이 뛰어난 분산 데이터 세트 (RDD) 작업을 지원하지 않습니다.

Amazon S3의 데이터 파일에 액세스할 수 없습니다.

Lake Formation에 데이터 레이크의 위치를 등록했는지 확인하십시오.

보안 검증 예외

EMR서버리스에서 보안 검증 오류를 감지했습니다. 연락처 AWS 지원 지원.

공유 중 AWS 계정 전반의 Glue 데이터 카탈로그 및 테이블

여러 계정에서 데이터베이스와 테이블을 공유하면서도 Lake Formation을 계속 사용할 수 있습니다. 자세한 내용은 Lake Formation의 계정 간 데이터 공유 및 공유 방법을 참조하십시오. AWS Glue 데이터 카탈로그 및 테이블 교차 계정 사용 AWS Lake Formation?.

작업자 간 암호화

Amazon EMR 버전 6.15.0 이상에서는 Spark 작업 실행 중인 작업자 간에 상호 TLS 암호화된 통신을 활성화할 수 있습니다. 활성화되면 EMR Serverless는 작업 실행 시 프로비저닝된 각 작업자에 대해 고유한 인증서를 자동으로 생성하여 배포합니다. 이러한 작업자가 통신하여 제어 메시지를 교환하거나 셔플 데이터를 전송할 때는 상호 TLS 연결을 설정하고 구성된 인증서를 사용하여 서로의 ID를 확인합니다. 작업자가 다른 인증서를 확인할 수 없는 경우 TLS 핸드셰이크가 실패하고 EMR 서버리스는 이들 간의 연결을 중단합니다.

Lake Formation을 EMR 서버리스와 함께 사용하는 경우 상호 TLS 암호화가 기본적으로 활성화됩니다.

서버리스에서 EMR 상호 TLS 암호화 활성화

Spark 애플리케이션에서 상호 TLS 암호화를 활성화하려면 EMR서버리스 애플리케이션을 만들 때 `spark.ssl.internode.enabled` true로 설정하십시오. 를 사용하는 경우 AWS 콘솔을 사용하여

EMR 서버리스 애플리케이션을 만들려면 사용자 지정 설정 사용을 선택한 다음 애플리케이션 구성을 확장하고 다음을 입력합니다`runtimeConfiguration`.

```
aws emr-serverless create-application \
--release-label emr-6.15.0 \
--runtime-configuration '{
  "classification": "spark-defaults",
  "properties": {"spark.ssl.internode.enabled": "true"}
}' \
--type "SPARK"
```

개별 스파크 작업 실행에 대해 상호 TLS 암호화를 활성화하려면 를 사용할 때 `spark.ssl.internode.enabled` true로 설정하십시오. `spark-submit`

```
--conf spark.ssl.internode.enabled=true
```

EMR서버리스를 통한 데이터 보호를 위한 Secrets Manager

AWS Secrets Manager 데이터베이스 자격 증명, API 키 및 기타 비밀 정보를 보호하는 데 사용할 수 있는 비밀 저장소 서비스입니다. 그런 다음 코드에서 하드코딩된 자격 증명을 Secrets Manager에 대한 API 호출로 바꿀 수 있습니다. 이렇게 하면 암호가 없기 때문에 코드를 검사하는 누군가가 암호를 침해하는 것을 방지할 수 있습니다. 개요는 다음을 참조하십시오. AWS Secrets Manager 사용자 가이드.

Secrets Manager는 다음을 사용하여 비밀을 암호화합니다. AWS Key Management Service 키. 자세한 내용은 의 비밀 암호화 및 암호 해독을 참조하십시오. AWS Secrets Manager 사용 설명서.

사용자가 지정한 일정에 따라 Secrets Manager가 자동으로 보안 암호를 교체하도록 구성할 수 있습니다. 따라서 단기 보안 암호로 장기 보안 암호를 교체할 수 있어 손상 위험이 크게 줄어듭니다. 자세한 내용은 회전을 참조하십시오. AWS Secrets Manager 비밀 속의 비밀 AWS Secrets Manager 사용자 가이드.

Amazon EMR 서버리스는 다음과 통합됩니다. AWS Secrets Manager 그러면 Secrets Manager에 데이터를 저장하고 구성에서 비밀 ID를 사용할 수 있습니다.

EMR서버리스가 비밀을 사용하는 방법

Secrets Manager에 데이터를 저장하고 EMR 서버리스 구성에서 비밀 ID를 사용하면 민감한 구성 데이터를 일반 텍스트로 EMR 서버리스에 전달하고 외부에 노출하지 않습니다. APIs 키-값 쌍에 Secrets

Manager에 저장한 암호의 비밀 ID가 포함되어 있다고 지정하면 EMR Serverless는 작업 실행을 위해 작업자에게 구성 데이터를 보낼 때 암호를 검색합니다.

구성의 키-값 쌍에 Secrets Manager에 저장된 암호에 대한 참조가 포함되어 있음을 나타내려면 구성 값에 `EMR.secret@` 주석을 추가하십시오. 보안 ID 주석이 있는 구성 속성의 경우 EMR 서버리스는 Secrets Manager를 호출하여 작업 실행 시 암호를 확인합니다.

시크릿 생성 방법

암호를 생성하려면 생성의 단계를 따르십시오. AWS Secrets Manager시크릿은 AWS Secrets Manager 사용자 가이드. 3단계에서 일반 텍스트 필드를 선택하여 민감한 값을 입력합니다.

구성 분류에 암호를 입력하십시오.

다음 예제는 에서 구성 분류에 암호를 제공하는 방법을 보여줍니다. `StartJobRun` 응용 프로그램 수준에서 Secrets Manager에 대한 분류를 구성하려면 을 참조하십시오서버리스의 기본 애플리케이션 구성 EMR.

예시에서는 검색할 비밀의 *`SecretName`* 이름으로 바꾸십시오. 하이픈과 Secrets Manager가 비밀 끝에 추가하는 여섯 개의 문자를 차례로 포함합니다. ARN 자세한 내용은 시크릿 생성 방법 단원을 참조하십시오.

이 섹션의 내용

- 비밀 참조 지정 - Spark
- 시크릿 레퍼런스 지정 - Hive

비밀 참조 지정 - Spark

Example — Spark의 외부 Hive 메타스토어 컨피그레이션에 시크릿 레퍼런스를 지정합니다.

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://DOC-EXAMPLE-BUCKET/scripts/spark-jdbc.py",
            "sparkSubmitParameters": "--jars s3://DOC-EXAMPLE-BUCKET/mariadb-connector-
java.jar
            --conf
 spark.hadoop.javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
```

```
            --conf spark.hadoop.javax.jdo.option.ConnectionUserName=connection-user-
name
            --conf
 spark.hadoop.javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
            --conf spark.hadoop.javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name
            --conf spark.driver.cores=2
            --conf spark.executor.memory=10G
            --conf spark.driver.memory=6G
            --conf spark.executor.cores=4"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://DOC-EXAMPLE-BUCKET/spark/logs/"
        }
    }
}'
```

Example — 분류에 외부 Hive 메타스토어 구성에 대한 비밀 참조를 지정합니다. **spark-defaults**

```
{
        "classification": "spark-defaults",
        "properties": {

 "spark.hadoop.javax.jdo.option.ConnectionDriverName":"org.mariadb.jdbc.Driver"
            "spark.hadoop.javax.jdo.option.ConnectionURL":"jdbc:mysql://db-host:db-
port/db-name"
            "spark.hadoop.javax.jdo.option.ConnectionUserName":"connection-user-name"
            "spark.hadoop.javax.jdo.option.ConnectionPassword":
 "EMR.secret@SecretName",
        }
    }
```

시크릿 레퍼런스 지정 - Hive

Example — Hive의 외부 Hive 메타스토어 구성에서 비밀 참조를 지정합니다.

```
aws emr-serverless start-job-run \
  --application-id "application-id" \
  --execution-role-arn "job-role-arn" \
    --job-driver '{
```

```
        "hive": {
        "query": "s3://DOC-EXAMPLE-BUCKET/emr-serverless-hive/query/hive-query.ql",
        "parameters": "--hiveconf hive.exec.scratchdir=s3://DOC-EXAMPLE-BUCKET/emr-
serverless-hive/hive/scratch
                    --hiveconf hive.metastore.warehouse.dir=s3://DOC-EXAMPLE-BUCKET/
emr-serverless-hive/hive/warehouse
                    --hiveconf javax.jdo.option.ConnectionUserName=username
                    --hiveconf
 javax.jdo.option.ConnectionPassword=EMR.secret@SecretName
                    --hiveconf
 hive.metastore.client.factory.class=org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreCli
                    --hiveconf
 javax.jdo.option.ConnectionDriverName=org.mariadb.jdbc.Driver
                    --hiveconf javax.jdo.option.ConnectionURL=jdbc:mysql://db-host:db-
port/db-name"
        }
    }' \
    --configuration-overrides '{
        "monitoringConfiguration": {
        "s3MonitoringConfiguration": {
            "logUri": "s3://EXAMPLE-LOG-BUCKET"
        }
    }
}'
```

Example — 분류에 외부 Hive 메타스토어 구성에 대한 비밀 참조를 지정합니다. **hive-site**

```
{
    "classification": "hive-site",
    "properties": {
        "hive.metastore.client.factory.class":
 "org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClientFactory",
        "javax.jdo.option.ConnectionDriverName": "org.mariadb.jdbc.Driver",
        "javax.jdo.option.ConnectionURL": "jdbc:mysql://db-host:db-port/db-name",
        "javax.jdo.option.ConnectionUserName": "username",
        "javax.jdo.option.ConnectionPassword": "EMR.secret@SecretName"
    }
}
```

EMR서버리스가 암호를 검색할 수 있도록 액세스 권한을 부여합니다.

EMR서버리스가 Secrets Manager에서 보안 값을 검색할 수 있도록 하려면 암호를 만들 때 다음 정책 설명을 암호에 추가하십시오. EMR서버리스가 비밀 값을 읽을 수 있도록 하려면 고객 관리 KMS 키로

암호를 생성해야 합니다. 자세한 내용은 키에 KMS 대한 권한을 참조하십시오. AWS Secrets Manager 사용 설명서.

다음 정책에서는 애플리케이션의 *applicationId* ID로 대체하십시오.

시크릿에 대한 리소스 정책

암호에 대한 리소스 정책에 다음 권한을 포함해야 합니다. AWS Secrets Manager EMR서버리스가 비밀 값을 검색할 수 있도록 하기 위해서입니다. 특정 애플리케이션만 이 비밀번호를 검색할 수 있도록 하기 위해 선택적으로 정책에서 EMR 서버리스 애플리케이션 ID를 조건으로 지정할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Principal": {
        "Service": [
          "emr-serverless.amazonaws.com"
        ]
      },
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceArn": "arn:aws:emr-serverless:AWS ##:aws_account_id:/
applications/applicationId"
        }
      }
    }
  ]
}
```

고객 관리형 비밀번호에 대해 다음 정책을 적용하여 비밀번호를 생성하십시오. AWS Key Management Service (AWS KMS) 키:

고객 관리 정책 AWS KMS 키

```
{
    "Sid": "Allow EMR Serverless to use the key for decrypting secrets",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "emr-serverless.amazonaws.com"
        ]
    },
    "Action": [
        "kms:Decrypt",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": "secretsmanager.AWS ##.amazonaws.com"
        }
    }
}
```

비밀의 전개

순환은 암호를 정기적으로 업데이트하는 것입니다. 구성할 수 있습니다. AWS Secrets Manager 지정한 일정에 따라 암호가 자동으로 교체되도록 할 수 있습니다. 이렇게 하면 장기 비밀을 단기 비밀로 대체할 수 있습니다. 이렇게 하면 보안 침해 위험을 줄이는 데 도움이 됩니다. EMR서버리스는 작업이 실행 상태로 전환될 때 주석이 달린 구성에서 비밀 값을 검색합니다. 사용자 또는 프로세스가 Secrets Manager에서 비밀 값을 업데이트하는 경우 작업에서 업데이트된 값을 가져올 수 있도록 새 작업을 제출해야 합니다.

> **Note**
>
> 이미 실행 상태에 있는 작업은 업데이트된 암호 값을 가져올 수 없습니다. 이로 인해 작업이 실패할 수 있습니다.

EMR서버리스에서 Amazon S3 액세스 권한 부여 사용

EMR서버리스에 대한 S3 액세스 권한 개요

Amazon EMR 릴리스 6.15.0 이상에서는 Amazon S3 Access Grants가 서버리스에서 Amazon S3 데이터에 대한 액세스를 강화하는 데 사용할 수 있는 확장 가능한 액세스 제어 솔루션을 제공합니다. EMR S3 데이터에 대한 권한 구성이 복잡하거나 대규모인 경우 Access Grants를 사용하여 사용자, 역할 및 애플리케이션을 위한 S3 데이터 권한을 확장할 수 있습니다.

S3 Access Grants를 사용하면 런타임 역할에서 부여한 권한 또는 EMR 서버리스 애플리케이션에 대한 액세스 권한이 있는 ID에 연결된 IAM 역할 이상으로 Amazon S3 데이터에 대한 액세스를 확대할 수 있습니다.

자세한 내용은 Amazon 관리 안내서의 EMR Amazon용 S3 액세스 권한을 통한 액세스 EMR 관리 및 Amazon 단순 스토리지 서비스 사용 설명서의 S3 액세스 권한을 통한 액세스 관리를 참조하십시오.

이 섹션에서는 S3 액세스 허가를 사용하여 Amazon S3의 데이터에 대한 액세스를 제공하는 EMR 서버리스 애플리케이션을 시작하는 방법을 설명합니다. S3 Access Grants를 다른 Amazon EMR 배포와 함께 사용하는 단계는 다음 설명서를 참조하십시오.

- Amazon에서 S3 액세스 권한 부여 사용 EMR
- Amazon이 EMR 설치된 상태에서 S3 액세스 권한 부여 사용 EKS

데이터 관리를 위해 S3 액세스 그랜트로 EMR 서버리스 애플리케이션을 시작하십시오.

EMR서버리스에서 S3 Access Grants를 활성화하고 Spark 애플리케이션을 시작할 수 있습니다. 애플리케이션에서 S3 데이터에 대한 요청이 발생할 경우 Amazon S3에서는 특정 버킷, 접두사 또는 객체로 범위가 지정된 임시 보안 인증을 제공합니다.

1. EMR서버리스 애플리케이션을 위한 작업 실행 역할을 설정합니다. Spark 작업을 실행하고 S3 Access Grants를 사용하는 데 필요한 필수 IAM 권한을 포함하고 다음을 포함하십시오. `s3:GetDataAccess` `s3:GetAccessGrantsInstanceForPrefix`

```
{
    "Effect": "Allow",
    "Action": [
    "s3:GetDataAccess",
```

```
    "s3:GetAccessGrantsInstanceForPrefix"
    ],
    "Resource": [     //LIST ALL INSTANCE ARNS THAT THE ROLE IS ALLOWED TO QUERY
         "arn:aws_partition:s3:Region:account-id1:access-grants/default",
         "arn:aws_partition:s3:Region:account-id2:access-grants/default"
    ]
}
```

> **Note**
>
> S3에 직접 액세스할 수 있는 추가 권한이 있는 작업 실행 IAM 역할을 지정하면 사용자는 S3 Access Grants의 권한이 없더라도 해당 역할이 허용하는 데이터에 액세스할 수 있습니다.

2. 다음 예와 같이 Amazon EMR 릴리스 레이블이 6.15.0 이상이고 `spark-defaults` 분류를 사용하여 EMR 서버리스 애플리케이션을 시작합니다. *red text*의 값을 사용 시나리오에 적합한 값으로 바꾸세요.

```
aws emr-serverless start-job-run \
  --application-id application-id \
  --execution-role-arn job-role-arn \
  --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://us-east-1.elasticmapreduce/emr-containers/samples/
wordcount/scripts/wordcount.py",
            "entryPointArguments": ["s3://DOC-EXAMPLE-BUCKET-OUTPUT/
wordcount_output"],
            "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
 spark.executor.memory=4g --conf spark.driver.cores=1 --conf spark.driver.memory=4g
 --conf spark.executor.instances=1"
        }
    }' \
  --configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.hadoop.fs.s3.s3AccessGrants.enabled": "true",
          "spark.hadoop.fs.s3.s3AccessGrants.fallbackToIAM": "false"
         }
      }]
}'
```

서버리스의 경우 S3 액세스 권한 부여 고려 사항 EMR

EMR서버리스와 함께 Amazon S3 액세스 권한을 사용할 때의 중요한 지원, 호환성 및 동작 정보는 Amazon EMR관리 가이드의 Amazon EMR 관련 S3 액세스 권한 부여 고려 사항을 참조하십시오.

를 EMR 사용하여 Amazon 서버리스 API 호출 로깅 AWS CloudTrail

Amazon EMR 서버리스는 다음과 통합되어 있습니다. AWS CloudTrail, 사용자, 역할 또는 담당자가 수행한 작업에 대한 기록을 제공하는 서비스입니다. AWS EMR서버리스 서비스. CloudTrail EMR서버리스에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처된 호출에는 서버리스 콘솔에서의 호출 및 EMR 서버리스 작업에 대한 코드 호출이 EMR 포함됩니다. API 트레일을 생성하면 EMR 서버리스용 CloudTrail 이벤트를 포함하여 Amazon S3 버킷에 이벤트를 지속적으로 전송할 수 있습니다. 트레일을 구성하지 않아도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 계속 볼 수 있습니다. 에서 수집한 CloudTrail 정보를 사용하여 EMR 서버리스에 이루어진 요청, 요청이 이루어진 IP 주소, 요청한 사람, 요청 시기 및 추가 세부 정보를 확인할 수 있습니다.

자세한 내용은 CloudTrail 다음을 참조하십시오. AWS CloudTrail 사용자 가이드.

EMR서버리스 정보: CloudTrail

CloudTrail 다음에서 활성화되어 있습니다. AWS 계정 계정을 만들 때 EMR서버리스에서 활동이 발생하면 해당 활동이 다른 CloudTrail 이벤트와 함께 이벤트에 기록됩니다. AWS 이벤트 기록의 서비스 이벤트. 내 사이트에서 최근 이벤트를 보고, 검색하고, 다운로드할 수 있습니다. AWS 계정. 자세한 내용은 이벤트 기록으로 CloudTrail 이벤트 보기를 참조하십시오.

내 이벤트의 진행 중인 기록을 보려면 AWS 계정 EMR서버리스 이벤트를 포함하여 트레일을 생성하십시오. 트레일을 사용하면 CloudTrail Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 기본적으로 콘솔에서 트레일을 생성하면 트레일이 모든 사용자에게 적용됩니다. AWS 리전. 트레일은 해당 지역의 모든 지역에서 발생한 이벤트를 기록합니다. AWS 지정한 Amazon S3 버킷으로 로그 파일을 분할하고 전송합니다. 또한 다른 항목을 구성할 수 있습니다. AWS CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 이에 따라 조치를 취하기 위한 서비스입니다. 자세한 내용은 다음 자료를 참조하십시오.

- 추적 생성 개요
- CloudTrail 지원되는 서비스 및 통합
- 다음에 대한 Amazon SNS 알림 구성 CloudTrail
- 여러 지역에서 CloudTrail 로그 파일 수신 및 여러 계정으로부터 CloudTrail 로그 파일 수신

모든 EMR 서버리스 작업은 EMRAPI서버리스 참조에 의해 CloudTrail 기록되고 문서화됩니다. 예를 들어,,`CreateApplication`, `StartJobRun` 를 호출하면 `CancelJobRun` 로그 파일에 항목이 생성됩니다. CloudTrail

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 보안 인증 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트를 사용하여 이루어졌는지 아니면 AWS Identity and Access Management (IAM) 사용자 자격 증명.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 사람이 요청했는지 여부 AWS 서비스.

자세한 내용은 CloudTrail userIdentity 요소를 참조하십시오.

EMR서버리스 로그 파일 항목 이해

트레일은 지정한 Amazon 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.

다음 예제는 `CreateApplication` 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
        "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
        "accountId": "012345678910",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::012345678910:role/Admin",
                "accountId": "012345678910",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
```

```
                "creationDate": "2022-06-01T23:46:52Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-06-01T23:49:28Z",
    "eventSource": "emr-serverless.amazonaws.com",
    "eventName": "CreateApplication",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "PostmanRuntime/7.26.10",
    "requestParameters": {
        "name": "my-serverless-application",
        "releaseLabel": "emr-6.6",
        "type": "SPARK",
        "clientToken": "0a1b234c-de56-7890-1234-567890123456"
    },
    "responseElements": {
        "name": "my-serverless-application",
        "applicationId": "1234567890abcdef0",
        "arn": "arn:aws:emr-serverless:us-west-2:555555555555:/
applications/1234567890abcdef0"
    },
    "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
    "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "012345678910",
    "eventCategory": "Management"
}
```


Amazon EMR 서버리스에 대한 규정 준수 검증

EMR서버리스의 보안 및 규정 준수는 여러 평가의 일환으로 타사 감사자가 평가합니다. AWS 다음을 포함한 규정 준수 프로그램:

- 시스템 및 조직 제어 (SOC)
- 결제 카드 업계 데이터 보안 표준 (PCIDSS)
- 연방 위험 및 권한 관리 프로그램 (FedRAMP) 보통
- 건강 보험 양도 및 책임에 관한 법률 () HIPAA

AWS 자주 업데이트되는 다음 목록을 제공합니다. AWS 특정 규정 준수 프로그램 범위의 서비스: AWS 규정 준수 프로그램별 범위 내 서비스.

타사 감사 보고서는 다음을 사용하여 다운로드할 수 있습니다. AWS Artifact. 자세한 내용은 보고서 다운로드를 참조하십시오. AWS Artifact.

에 대한 자세한 내용은 AWS 규정 준수 프로그램은 다음을 참조하십시오. AWS 규정 준수 프로그램.

EMR서버리스를 사용할 때의 규정 준수 책임은 데이터의 민감도, 조직의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. EMR서버리스를 사용할 때 HIPAAPCI, 또는 Fed RAMP Moderate와 같은 표준을 준수해야 하는 경우 AWS 도움이 되는 리소스를 제공합니다.

- 보안 및 규정 준수에 중점을 둔 기본 환경을 구축하기 위한 아키텍처 고려 사항과 단계를 설명하는 보안 및 규정 준수 퀵 스타트 가이드 AWS.
- AWS 고객 규정 준수 가이드는 규정 준수의 관점에서 공동 책임 모델을 이해하는 데 도움이 될 수 있습니다. 이 가이드에는 보안 모범 사례가 요약되어 있습니다. AWS 서비스 또한 이 지침을 여러 프레임워크 (국립 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (), 국제 표준화 기구 (PCI) 포함) 의 보안 제어에 매핑하십시오. ISO
- AWS Config를 사용하여 리소스 구성이 내부 관행, 업계 지침 및 규정을 준수하는 정도를 평가할 수 있습니다.
- AWS 규정 준수 리소스는 해당 산업 및 지역에 적용할 수 있는 통합 문서 및 가이드 모음입니다.
- AWS Security Hub는 보안 상태를 포괄적으로 보여줍니다. AWS 보안 업계 표준 및 모범 사례를 준수하는지 확인할 수 있도록 도와줍니다.
- AWS Audit Manager— 이것은 AWS 서비스 지속적으로 감사할 수 있도록 도와줍니다. AWS 사용을 통해 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

Amazon EMR 서버리스의 레질리언스

The AWS 글로벌 인프라는 다음을 중심으로 구축됩니다. AWS 지역 및 가용 영역. AWS 리전은 물리적으로 분리되고 격리된 다수의 가용 리전을 제공하며 이러한 가용 리전은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

에 대한 자세한 내용은 AWS 지역 및 가용 영역을 참조하십시오. AWS 글로벌 인프라.

뿐만 아니라 AWS 글로벌 인프라인 Amazon EMR Serverless는 Amazon S3와의 통합을 통해 데이터 복원력과 백업 요구 사항을 지원하는 EMRFS 데 도움이 됩니다.

Amazon EMR 서버리스의 인프라 보안

EMRAmazon은 관리형 서비스로서 다음과 같은 보호를 받습니다. AWS 글로벌 네트워크 보안. 자세한 내용은 AWS 보안 서비스 및 방법 AWS 인프라 보호, 참조 AWS 클라우드 보안. 다음을 설계하려면 AWS 인프라 보안 모범 사례를 사용하는 환경을 보려면 보안 기둥의 인프라 보호를 참조하십시오. AWS 잘 설계된 프레임워크.

다음과 같이 사용합니다. AWS 네트워크를 EMR 통해 Amazon에 액세스하기 위한 API 호출을 게시했습니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안 (TLS). TLS1.2가 필요하고 TLS 1.3을 권장합니다.
- (임시 디피-헬만) 또는 (타원 곡선 임시 디피-헬만PFS) 와 같이 완벽한 순방향 기밀성 DHE () 을 갖춘 암호 제품군. ECDHE Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 액세스 키 ID와 보안 주체와 연결된 비밀 액세스 키를 사용하여 요청에 서명해야 합니다. IAM 또는 다음을 사용할 수 있습니다. AWS Security Token Service (AWS STS) 를 사용하여 요청에 서명하기 위한 임시 보안 자격 증명을 생성할 수 있습니다.

Amazon EMR 서버리스의 구성 및 취약성 분석

AWS 게스트 운영 체제 (OS) 및 데이터베이스 패치, 방화벽 구성, 재해 복구와 같은 기본 보안 작업을 처리합니다. 적합한 제3자가 이 절차를 검토하고 인증하였습니다. 자세한 내용은 다음 리소스를 참조하세요.

- Amazon EMR 서버리스에 대한 규정 준수 검증
- 공동 책임 모델
- Amazon Web Services: 보안 프로세스의 개요(백서)

엔드포인트 및 할당량 EMR Serverless

서비스 엔드포인트

프로그래밍 방식으로 연결하려면 AWS 서비스엔드포인트를 사용합니다. 엔드포인트는 URL 엔트리 포인트의 역할을 합니다. AWS 웹 서비스. 표준 외에도 AWS 엔드포인트, 일부 AWS 서비스 일부 지역에서 FIPS 엔드포인트를 제공합니다. 다음 표에는 서버리스의 서비스 엔드포인트가 나열되어 있습니다. EMR 자세한 내용은 단원을 참조하세요.AWS 서비스 엔드포인트.

지역명	지역	엔드포인트	프로토콜
미국 동부(오하이오)	us-east-2 (다음 가용 영역으로 제한됨: use2-az1use2-az2, 및use2-az3)	emr-serverless.us-east-2.amazonaws.com	HTTPS
미국 동부(버지니아 북부)	us-east-1 (다음 가용 영역으로 제한:use1-az1, use1-az2use1-az4,use1-az5, 및use1-az6)	emr-serverless.us-east-1.amazonaws.com emr-serverless-fips.us-east-1.amazonaws.com	HTTPS
미국 서부(캘리포니아 북부)	us-west-1	emr-serverless.us-west-1.amazonaws.com	HTTPS
미국 서부(오레곤)	us-west-2	emr-serverless.us-west-2.amazonaws.com	HTTPS

지역명	지역	엔드포인트	프로토콜
		emr-serverless-fips.us-west-2.amazonaws.com	
아프리카(케이프타운)	af-south-1	emr-serverless.af-south-1.amazonaws.com	HTTPS
아시아 태평양(홍콩)	ap-east-1	emr-serverless.ap-east-1.amazonaws.com	HTTPS
아시아 태평양(자카르타)	ap-southeast-3	emr-serverless.ap-southeast-3.amazonaws.com	HTTPS
아시아 태평양(뭄바이)	ap-south-1	emr-serverless.ap-south-1.amazonaws.com	HTTPS
아시아 태평양(오사카)	ap-northeast-3	emr-serverless.ap-northeast-3.amazonaws.com	HTTPS

지역명	지역	엔드포인트	프로토콜
아시아 태평양(서울)	ap-northeast-2	emr-serverless.ap-northeast-2.amazonaws.com	HTTPS
아시아 태평양(싱가포르)	ap-southeast-1	emr-serverless.ap-southeast-1.amazonaws.com	HTTPS
아시아 태평양(시드니)	ap-southeast-2	emr-serverless.ap-southeast-2.amazonaws.com	HTTPS
아시아 태평양(도쿄)	ap-northeast-1	emr-serverless.ap-northeast-1.amazonaws.com	HTTPS
캐나다(중부)	ca-central-1 (다음 가용 영역으로 제한: cac1-az1 및cac1-az2)	emr-serverless.ca-central-1.amazonaws.com	HTTPS
유럽(프랑크푸르트)	eu-central-1	emr-serverless.eu-central-1.amazonaws.com	HTTPS

지역명	지역	엔드포인트	프로토콜
유럽(아일랜드)	eu-west-1	emr-serverless.eu-west-1.amazonaws.com	HTTPS
유럽(런던)	eu-west-2	emr-serverless.eu-west-2.amazonaws.com	HTTPS
유럽(밀라노)	eu-south-1	emr-serverless.eu-south-1.amazonaws.com	HTTPS
유럽(파리)	eu-west-3	emr-serverless.eu-west-3.amazonaws.com	HTTPS
유럽(스페인)	eu-south-2	emr-serverless.eu-south-2.amazonaws.com	HTTPS
유럽(스톡홀름)	eu-north-1	emr-serverless.eu-north-1.amazonaws.com	HTTPS
중동(바레인)	me-south-1	emr-serverless.me-south-1.amazonaws.com	HTTPS

지역명	지역	엔드포인트	프로토콜
중동 (UAE)	me-central-1	emr-serverless.me-central-1.amazonaws.com	HTTPS
남아메리카(상파울루)	sa-east-1	emr-serverless.sa-east-1.amazonaws.com	HTTPS
AWS GovCloud (미국 동부)	us-gov-east-1	emr-serverless.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (미국 서부)	us-gov-west-1	emr-serverless.us-gov-west-1.amazonaws.com	HTTPS

Service quotas

서비스 할당량 (한도라고도 함) 은 서비스 리소스 또는 운영의 최대 수입니다. AWS 계정 사용할 수 있습니다. EMR서버리스는 1분마다 서비스 할당량 사용량 지표를 수집하여 네임스페이스에 게시합니다. `AWS/Usage`

> **Note**
>
> New AWS 계정의 초기 할당량은 낮지만 시간이 지남에 따라 증가할 수 있습니다. Amazon EMR 서버리스는 각 계정 내의 계정 사용을 모니터링합니다. AWS 리전그런 다음 사용량에 따라 할당량을 자동으로 늘립니다.

다음 표에는 서버리스의 서비스 할당량이 나와 있습니다. EMR 자세한 내용은 단원을 참조하세요.AWS 서비스 할당량.

명칭	기본 한도	조정 가능?	설명
계정당 최대 동시 접속 vCPUs	16	예	현재 계정에서 동시에 실행할 수 vCPUs 있는 최대 개수 AWS 리전.

API한도

다음은 귀하의 지역별 API 한도에 대한 설명입니다. AWS 계정.

Resource	기본 할당량
ListApplications	초당 10건의 거래. 초당 50건의 트랜잭션 버스트
CreateApplication	초당 1건의 트랜잭션 초당 25건의 트랜잭션 버스트
DeleteApplication	초당 1건의 트랜잭션 초당 25건의 트랜잭션 버스트
GetApplication	초당 10건의 트랜잭션 초당 50건의 트랜잭션 버스트
UpdateApplication	초당 1건의 트랜잭션 초당 25건의 트랜잭션 버스트
ListJobRuns	초당 1건의 트랜잭션 초당 25건의 트랜잭션 버스트
StartJobRun	초당 1건의 트랜잭션 초당 25건의 트랜잭션 버스트
GetDashboardForJobRun	초당 1건의 트랜잭션 초당 2건의 트랜잭션 버스트
CancelJobRun	초당 1건의 트랜잭션. 초당 25건의 트랜잭션 버스트

Resource	기본 할당량
GetJobRun	초당 10건의 트랜잭션 초당 50건의 트랜잭션 버스트
StartApplication	초당 1건의 트랜잭션 초당 25건의 트랜잭션 버스트
StopApplication	초당 1건의 트랜잭션 초당 25건의 트랜잭션 버스트

기타 고려 사항

다음 목록에는 서버리스에 대한 기타 고려 사항이 포함되어 있습니다EMR.

- EMR서버리스는 다음에서 사용할 수 있습니다. AWS 리전:
  - 미국 동부(오하이오)
  - 미국 동부(버지니아 북부)
  - 미국 서부(캘리포니아 북부)
  - 미국 서부(오레곤)
  - 아프리카(케이프타운)
  - 아시아 태평양(홍콩)
  - 아시아 태평양(자카르타)
  - 아시아 태평양(뭄바이)
  - 아시아 태평양(오사카)
  - 아시아 태평양(서울)
  - 아시아 태평양(싱가포르)
  - 아시아 태평양(시드니)
  - 아시아 태평양(도쿄)
  - 캐나다(중부)
  - 유럽(프랑크푸르트)
  - 유럽(아일랜드)
  - 유럽(런던)
  - 유럽(밀라노)
  - 유럽(파리)
  - 유럽(스페인)
  - 유럽(스톡홀름)
  - 중동(바레인)
  - 중동 () UAE
  - 남아메리카(상파울루)
  - AWS GovCloud (미국 동부)
  - AWS GovCloud (미국 서부)

이러한 지역과 관련된 엔드포인트 목록은 을 참조하십시오. 서비스 엔드포인트

- 작업 실행의 기본 제한 시간은 12시간입니다. `startJobRun`API또는 의 `executionTimeoutMinutes` 속성을 사용하여 이 설정을 변경할 수 있습니다. AWS SDK. 작업 실행 시간이 초과되지 않도록 `executionTimeoutMinutes` 하려면 0으로 설정할 수 있습니다. 예를 들어 스트리밍 응용 프로그램이 있는 경우 스트리밍 작업이 계속 `executionTimeoutMinutes` 실행되도록 0으로 설정할 수 있습니다.
- 의 `billedResourceUtilization` 속성은 다음과 같은 애그리게이트 vCPU, 메모리 및 스토리지를 `getJobRun` API 보여줍니다. AWS 작업 실행에 대해 요금이 청구되었습니다. 청구되는 리소스에는 작업자의 최소 1분 사용량과 작업자당 20GB가 넘는 추가 스토리지가 포함됩니다. 사전 초기화된 유휴 작업자의 사용량은 이러한 리소스에 포함되지 않습니다.
- VPC연결이 없으면 작업에서 일부에 액세스할 수 있습니다. AWS 서비스 동일한 엔드포인트 AWS 리전. 이러한 서비스에는 Amazon S3가 포함됩니다. AWS Glue, 아마존 CloudWatch 로그, AWS KMS, AWS Security Token Service, Amazon DynamoDB, 및 AWS Secrets Manager. VPC연결을 활성화하여 다른 사용자에게 액세스할 수 있습니다. AWS 서비스 를 통해 AWS PrivateLink하지만 꼭 그럴 필요는 없습니다. 외부 서비스에 액세스하려면 를 사용하여 애플리케이션을 만들 수 있습니다VPC.
- EMR서버리스는 지원하지 HDFS 않습니다. 작업자의 로컬 디스크는 EMR 서버리스가 작업 실행 중에 데이터를 셔플하고 처리하는 데 사용하는 임시 저장소입니다.
- EMR서버리스는 기존 서버를 지원하지 않습니다. emr-dynamodb-connector

Amazon EMR 서버리스 릴리스 버전

Amazon EMR 릴리스는 빅 데이터 생태계의 오픈 소스 애플리케이션 세트입니다. 각 릴리스에는 작업 실행 시 Amazon EMR Serverless에서 배포하고 구성하도록 선택한 빅 데이터 애플리케이션, 구성 요소 및 기능이 포함됩니다.

Amazon EMR 6.6.0 이상에서는 서버리스를 EMR 배포할 수 있습니다. 이전 Amazon EMR 릴리스 버전에서는 이 배포 옵션을 사용할 수 없습니다. 작업을 제출할 때 지원되는 다음 릴리스 중 하나를 지정해야 합니다.

주제

EMR Serverless 7.2.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 7.2.0.

애플리케이션	버전
Apache Spark	3.5.1

애플리케이션	버전
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR서버리스 7.2.0 릴리스 노트

- EMR서버리스를 통한 Lake Formation — 이제 사용할 수 있습니다 AWS Lake Formation S3가 지원하는 데이터 카탈로그 테이블에 세밀한 액세스 제어를 적용할 수 있습니다. 이 기능을 사용하면 EMR Serverless Spark 작업 내에서 읽기 쿼리에 대한 테이블, 행, 열 및 셀 수준의 액세스 제어를 구성할 수 있습니다. 자세한 내용은 the section called “Lake Formation for FGAC” 및 the section called “고려 사항” 단원을 참조하세요.

EMR Serverless 7.1.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 7.1.0.

애플리케이션	버전
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 7.0.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 7.0.0.

애플리케이션	버전
Apache Spark	3.5.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.15.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 6.15.0.

애플리케이션	버전
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR서버리스 6.15.0 릴리스 노트

- TLS지원 — Amazon EMR Serverless 릴리스 6.15.0 이상을 사용하면 Spark 작업 실행 중인 작업자 간에 상호 TLS 암호화된 통신을 활성화할 수 있습니다. 활성화되면 EMR 서버리스는 각 작업자에 대해 고유한 인증서를 자동으로 생성하며, 이 인증서는 작업자가 TLS 핸드셰이크 중에 이를 활용하여 서로를 인증하고 데이터를 안전하게 처리하기 위한 암호화된 채널을 구축합니다. 상호 암호화에 대한 자세한 내용은 작업자 간 TLS 암호화를 참조하십시오.

EMR Serverless 6.14.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 6.14.0.

애플리케이션	버전
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.13.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 6.13.0.

애플리케이션	버전
Apache Spark	3.4.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.12.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 6.12.0.

애플리케이션	버전
Apache Spark	3.4.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR Serverless 6.11.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 6.11.0.

애플리케이션	버전
Apache Spark	3.3.2
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR서버리스 6.11.0 릴리스 노트

- 다른 계정의 S3 리소스에 액세스 - 릴리스 6.11.0 이상에서는 서로 다른 Amazon S3 버킷에 액세스할 때 담당하도록 여러 IAM 역할을 구성할 수 있습니다. AWS 서버리스 계정. EMR

EMR Serverless 6.10.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 6.10.0.

애플리케이션	버전
Apache Spark	3.3.1
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR서버리스 6.10.0 릴리스 노트

- 릴리스 6.10.0 이상이 설치된 EMR 서버리스 애플리케이션의 경우 속성의 기본값은 입니다. `spark.dynamicAllocation.maxExecutors` `infinity` 이전 릴리스의 기본값은 입니다. `100` 자세한 내용은 스파크 작업 속성 단원을 참조하십시오.

EMR Serverless 6.9.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 6.9.0.

애플리케이션	버전
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.10.2

EMR서버리스 6.9.0 릴리스 노트

- 아파치 스파크를 위한 Amazon Redshift 통합은 아마존 EMR 릴리스 6.9.0 이상에 포함되어 있습니다. 이전의 오픈 소스 도구였던, 이 기본 통합은 Spark 커넥터로, Amazon Redshift와 Amazon Redshift Serverless에서 데이터를 읽고 쓰는 Apache Spark 애플리케이션을 빌드할 수 있습니다. 자세한 내용은 아마존 서버리스에서 아파치 스파크에 Amazon Redshift 통합 사용 EMR 단원을 참조하십시오.

- EMR서버리스 릴리스 6.9.0에는 다음과 같은 지원이 추가되었습니다. AWS 그라비톤2 (arm64) 아키텍처. `create-application`및 `architecture` 매개 변수를 사용하여 arm64 아키텍처를 선택할 수 `update-application` APIs 있습니다. 자세한 내용은 Amazon EMR 서버리스 아키텍처 옵션 단원을 참조하십시오.
- 이제 서버리스 Spark 및 Hive 애플리케이션에서 직접 Amazon DynamoDB 테이블을 내보내고, 가져오고, 쿼리하고, EMR 조인할 수 있습니다. 자세한 내용은 Amazon 서버리스를 사용하여 DynamoDB에 연결 EMR 단원을 참조하십시오.

알려진 문제

- Apache Spark용 Amazon Redshift 통합을 사용하고 Parquet 형식의 time, timetz, timestamp 또는 timestamptz(마이크로초 정밀도)를 사용하는 경우 커넥터는 시간 값을 가장 가까운 밀리초 값으로 반올림합니다. 해결 방법으로 텍스트 언로드 형식 `unload_s3_format` 파라미터를 사용합니다.

EMR Serverless 6.8.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 6.8.0.

애플리케이션	버전
Apache Spark	3.3.0
Apache Hive	3.1.3
Apache Tez	0.9.2

EMR Serverless 6.7.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 6.7.0.

애플리케이션	버전
Apache Spark	3.2.1
Apache Hive	3.1.3

애플리케이션	버전
Apache Tez	0.9.2

엔진별 변경, 개선 사항, 해결된 문제

다음 표에는 새로운 엔진별 기능이 나열되어 있습니다.

변경 사항	설명
기능	Tez 스케줄러는 이제 컨테이너 선점 대신 Tez 태스크 선점을 지원합니다.

EMR Serverless 6.6.0

다음 표에는 에서 사용할 수 있는 애플리케이션 버전이 나와 있습니다.EMR Serverless 6.6.0.

애플리케이션	버전
Apache Spark	3.2.0
Apache Hive	3.1.2
Apache Tez	0.9.2

EMR서버리스 초기 릴리스 노트

- EMR서버리스는 Spark 구성 분류를 지원합니다. `spark-defaults` 이 분류는 Spark 파일의 값을 변경합니다. `spark-defaults.conf` XML 구성 분류를 사용하면 애플리케이션을 사용자 지정할 수 있습니다. 자세한 내용은 애플리케이션 구성을 참조하세요.
- EMR서버리스는 Hive 구성 분류`hive-site`,, `tez-site` 및 를 지원합니다. `emrfs-site` `core-site` 이 분류는 Hive `hive-site.xml` 파일, Tez 파일, Amazon EMR EMRFS 설정 또는 하둡 `tez-site.xml` 파일의 값을 각각 변경할 수 있습니다. `core-site.xml` 구성 분류를 사용하면 애플리케이션을 사용자 지정할 수 있습니다. 자세한 내용은 애플리케이션 구성을 참조하세요.

엔진별 변경, 개선 사항 및 해결된 문제

- 다음 표에는 Hive 및 Tez 백포트가 나와 있습니다.

  Hive 및 Tez 변경 사항

변경 사항	설명
백포트	TEZ-4430: 속성 관련 문제 수정 `tez.task.launch.cmd-opts`
백포트	HIVE-25971: 캐시된 스레드 풀 열기로 인한 Tez 태스크 종료 지연 현상 수정

문서 이력

다음 표는 EMR 서버리스의 마지막 릴리스 이후 설명서에서 변경된 주요 내용을 설명합니다. RSS피드를 구독하면 이 설명서의 업데이트에 대한 자세한 내용을 확인할 수 있습니다.

변경 사항	설명	날짜
새로운 릴리스	EMR Serverless 7.2.0	2024년 7월 25일
새로운 릴리스	EMR Serverless 7.1.0	2024년 4월 17일
기존 정책 업데이트.	A mazonEMRServerless ServiceRolePolicy 정책에 새 Sid CloudWatchPolicySt atement EC2Policy Statement AND를 추가했습니다.	2024년 1월 25일
새로운 릴리스	EMR Serverless 7.0.0	2023년 12월 29일
새로운 릴리스	EMR Serverless 6.15.0	2023년 11월 17일
새 기능	EMR서버리스 (6.11 이상) 와 다른 계정의 Amazon S3 버킷에 액세스할 때 담당하도록 여러 IAM 역할을 구성하십시오.	2023년 10월 18일
새로운 릴리스	EMR Serverless 6.14.0	2023년 10월 17일
새 기능	서버리스의 기본 애플리케이션 구성 EMR	2023년 9월 25일
기본 Hive 속성으로 업데이트	hive.driver.disk , hive.tez.disk.size hive.tez.auto.redu cer.parallelism , 및 tez.grouping.min-s	2023년 9월 12일

	ize Hive 작업 속성의 기본값을 업데이트했습니다.	
새로운 릴리스	EMR Serverless 6.13.0	2023년 9월 11일
새로운 릴리스	EMR Serverless 6.12.0	2023년 7월 21일
새로운 릴리스	EMR Serverless 6.11.0	2023년 6월 8일
서비스 연결 역할 정책 업데이트	네임스페이스에 계정 수준 사용을 게시하도록 AmazonEMRServerlessServiceRolePolicy SLR역할을 업데이트했습니다. "AWS/Usage"	2023년 4월 20일
EMR Serverless 일반 가용성(GA)	EMR서버리스의 첫 번째 공개 릴리스입니다.	2022년 6월 1일

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