aws

개발자 가이드

기한 클라우드

기한 클라우드: 개발자 가이드

Table of Contents

AWS Deadline Cloud란 무엇입니까?

AWS Deadline Cloud는 확장 가능한 처리 팜을 몇 분 안에 시작하고 실행할 수 있는 완전 관리형 AWS 서비스입니다. 작업을 예약하기 위한 사용자, 팜, 대기열 및 처리를 수행하는 작업자 플릿을 관리하기 위한 관리 콘솔을 제공합니다.

이 개발자 안내서는 다음을 포함한 다양한 사용 사례의 파이프라인, 도구 및 애플리케이션 개발자를 위한 것입니다.

- 파이프라인 개발자와 기술 책임자는 Deadline Cloud APIs 및 기능을 사용자 지정 프로덕션 파이프라인에 통합할 수 있습니다.
- 독립 소프트웨어 공급업체는 Deadline Cloud를 애플리케이션에 통합하여 디지털 콘텐츠 생성 아티스트와 사용자가 워크스테이션에서 Deadline Cloud 렌더링 작업을 원활하게 제출할 수 있습니다.
- 웹 및 클라우드 기반 서비스 개발자는 Deadline Cloud 렌더링을 플랫폼에 통합하여 고객이 자산을 제공하여 제품을 가상으로 볼 수 있습니다.

파이프라인의 모든 단계에서 직접 작업할 수 있는 도구를 제공합니다.

- 직접 또는 스크립트에서 사용할 수 있는 명령줄 인터페이스입니다.
- 널리 사용되는 11개 프로그래밍 언어용 AWS SDK입니다.
- 애플리케이션에서 호출할 수 있는 REST 기반 웹 인터페이스입니다.

사용자 지정 애플리케이션에서 다른 AWS 서비스 를 사용할 수도 있습니다. 예를 들어 다음을 사용할 수 있습니다.

- AWS CloudFormation - 팜, 대기열 및 플릿 생성 및 제거를 자동화합니다.
- Amazon CloudWatch는 작업에 대한 지표를 수집합니다.
- 디지털 자산 및 작업 출력을 저장하고 관리하기 위한 Amazon Simple Storage Service.
- AWS IAM Identity Center - 팜의 사용자 및 그룹을 관리합니다.

Open 작업 설명

Deadline Cloud는 Open Job Description(OpenJD) 사양을 사용하여 작업의 세부 정보를 지정합니다. OpenJD는 솔루션 간에 이동할 수 있는 작업을 정의하기 위해 개발되었습니다. 이를 사용하여 작업자 호스트에서 실행되는 명령 집합인 작업을 정의합니다.

Deadline Cloud에서 제공하는 제출자를 사용하여 OpenJD 작업 템플릿을 생성하거나 템플릿을 생성하려는 도구를 사용할 수 있습니다. 템플릿을 생성한 후 Deadline Cloud로 보냅니다. 제출자를 사용하는 경우 템플릿 전송을 담당합니다. 템플릿을 다른 방식으로 생성한 경우 Deadline Cloud 명령줄 작업을 호출하거나 AWS SDKs 중 하나를 사용하여 작업을 전송할 수 있습니다. 어느 쪽이든 Deadline Cloud는 지정된 대기열에 작업을 추가하고 작업을 예약합니다.

Deadline Cloud의 개념 및 용어

AWS Deadline Cloud를 시작하는 데 도움이 되도록이 주제에서는 몇 가지 주요 개념과 용어를 설명합니다.

팜 리소스

이 다이어그램은 Deadline Cloud farm 리소스가 함께 작동하는 방식을 보여줍니다.

팜

팜에는 작업 제출 및 실행과 관련된 다른 모든 리소스가 포함되어 있습니다. 팜은 서로 독립적이므로 프로덕션 환경을 분리하는 데 유용합니다.

대기열

대기열에는 연결된 플릿에 대한 예약 작업이 있습니다. 사용자는 대기열에 작업을 제출하고 대기열 내에서 우선 순위와 상태를 관리할 수 있습니다. 작업을 실행하려면 대기열을 대기열-플릿 연결이 있는 플릿과 연결해야 하며 대기열을 여러 플릿과 연결할 수 있습니다.

플릿

플릿에는 실행 중인 작업을 위한 컴퓨팅 용량이 포함되어 있습니다. 플릿은 서비스 관리형 또는 고객 관리형일 수 있습니다. 서비스 관리형 플릿은 Deadline Cloud에서 실행되며 Auto Scaling, 라이선스 및 소프트웨어 액세스와 같은 내장 기능을 포함합니다. 고객 관리형 플릿은 Amazon EC2 인스턴스 또는 온프레미스 서버와 같은 자체 컴퓨팅 리소스에서 실행됩니다.

Budget

예산은 작업 활동에 대한 지출 임계값을 설정하고 임계값에 도달하면 작업 일정 중지와 같은 작업을 수행할 수 있습니다.

대기열 환경

대기열 환경은 워크로드 환경을 설정하거나 해제하기 위해 각 작업자에서 실행되는 스크립트를 정의합니다. 환경 변수를 설정하고, 소프트웨어를 설치하고, 자산 스토리지를 구성하는 데 유용합니다.

스토리지 프로필

스토리지 프로파일은 호스트 및 워크스테이션 그룹에 대한 구성으로, 파일 시스템에서 데이터가 있는 위치를 알려줍니다. Deadline Cloud는에서 제출Windows되고에서 실행되는 작업과 같이 다르게 구성된 호스트에서 작업을 실행할 때 스토리지 프로파일을 사용하여 경로를 매핑합니다Linux.

Limit

제한을 사용하면 부동 라이선스와 같은 공유 리소스의 사용량을 추적하고 작업 간에 할당되는 방식을 제어할 수 있습니다. 제한은 대기열 제한 연결이 있는 대기열과 연결됩니다.

모니터링

모니터는 Deadline Cloud 모니터 웹 애플리케이션의 URL을 구성하여 최종 사용자가 작업을 모니터링하고 관리할 수 있도록 합니다. 브라우저 또는 Deadline Cloud Monitor 데스크톱 애플리케이션을 통해 액세스할 수 있습니다.

작업 실행 리소스

이 다이어그램은 Deadline Cloud 작업 리소스가 함께 작동하는 방식을 보여줍니다.

작업

작업은 사용자가 Deadline Cloud에 제출하여 사용 가능한 작업자에 대해 예약하고 실행하는 일련의 작업입니다. 작업은 3D 장면을 렌더링하거나 시뮬레이션을 실행할 수 있습니다. 작업은 런타임 환경 및 프로세스와 작업별 파라미터를 정의하는 재사용 가능한 작업 템플릿에서 생성됩니다. 작업에는 수행할 작업을 정의하는 단계와 작업이 포함되며 우선 순위, 최대 작업자 수 및 재시도 설정으로 구성할 수 있습니다.

작업 우선 순위

작업 우선 순위는 Deadline Cloud가 대기열에서 작업을 처리하는 대략적인 순서입니다. 작업 우선 순위를 1~100으로 설정할 수 있으며, 일반적으로 우선 순위가 높은 작업이 먼저 처리됩니다. 우선 순위가 동일한 작업은 수신된 순서대로 처리됩니다.

작업 속성

작업 속성은 렌더링 작업을 제출할 때 정의하는 설정입니다. 프레임 범위, 출력 경로, 작업 연결, 렌더링 가능한 카메라 등을 예로 들 수 있습니다. 속성은 렌더링이 제출되는 DCC에 따라 달라집니다.

단계

단계는 작업 파라미터 값을 제외하고 동일한 많은 작업을 실행하기 위한 템플릿을 제공하는 작업의 일부입니다. 단계는 다른 단계에 종속될 수 있으므로 순차적 또는 병렬 실행 경로로 복잡한 워크플로를 생성할 수 있습니다. 렌더링 작업에서 단계는 종종 프레임 렌더링 명령을 정의하고 프레임 번호를 작업 파라미터로 사용합니다.

Task

작업은 Deadline Cloud에서 가장 작은 작업 단위입니다. 작업은 단계의 일부이며 작업자가 실행하며 작업의 일부로 수행해야 하는 개별 작업을 나타냅니다. 작업은 특정 파라미터로 구성할 수 있으며 기능과 가용성에 따라 작업자에게 할당됩니다. 렌더링 작업에서 작업은 종종 단일 프레임을 렌더링합니다.

작업자

작업자는 플릿의 일부이며 작업에서 작업을 실행합니다. GPU 액셀러레이터, CPU 아키텍처, 운영 체제와 같은 특정 기능으로 작업자를 구성할 수 있습니다. 서비스 관리형 플릿에서는 플릿이 스케일 아웃 및 스케일 인될 때 작업자가 자동으로 생성됩니다.

Instance

플릿은 CPU 리소스에 인스턴스를 사용합니다. 인스턴스는 Amazon EC2 성능 인스턴스입니다. Deadline Cloud는 온디맨드 및 스팟 인스턴스를 사용합니다.

온디맨드 인스턴스

온디맨드 인스턴스는 초 단위로 요금이 책정되고, 장기 약정이 없으며, 중단되지 않습니다.

스팟 인스턴스

스팟 인스턴스는 예약되지 않은 용량으로, 할인된 가격으로 사용할 수 있지만 온디맨드 요청으로 인해 중단될 수 있습니다.

대기 및 저장

대기 및 저장 기능은 더 저렴한 비용으로 지연된 작업 예약을 제공하며 온디맨드 및 스팟 요청에 의해 중단될 수 있습니다. 대기 및 저장은 Deadline Cloud 서비스 관리형 플릿 내에서만 사용할 수 있습니다.

Wait and Save는 AWS Deadline Cloud에서 시각적 컴퓨팅 워크로드의 실행을 관리하기 위한 것입니다. 자세한 내용은 AWS 서비스 약관을 참조하세요.

세션

세션은 작업에 대한 작업자의 작업 시퀀스를 나타냅니다. 단일 세션 중에 작업자에게 여러 작업이 할당되어 차례로 실행될 수 있습니다. 세션에는 작업 작업을 실행하기 전에 환경을 구성하고 자산을 로드하는 설정 작업이 있는 경우가 많습니다.

세션 작업

세션 작업은 환경 설정, 작업 실행, 자산 동기화와 같이 세션 중에 수행되는 특정 작업을 나타냅니다.

기타 중요한 개념 및 용어

사용량 탐색기

Usage Explorer는 Deadline Cloud Monitor의 기능입니다. 대략적인 예상 비용 및 사용량을 제공합니다.

예산 관리자

예산 관리자는 Deadline Cloud 모니터의 일부입니다. 예산 관리자를 사용하여 예산을 생성하고 관리합니다. 또한 이를 사용하여 예산 내에서 유지되도록 활동을 제한할 수 있습니다.

Deadline Cloud 클라이언트 라이브러리

오픈 소스 클라이언트 라이브러리에는 Deadline Cloud를 관리하기 위한 명령줄 인터페이스와 라이브러리가 포함되어 있습니다. 이 기능에는 Open Job Description 사양을 기반으로 Deadline Cloud에 작업 번들 제출, 작업 연결 출력 다운로드, 명령줄 인터페이스(CLI)를 사용한 팜 모니터링이 포함됩니다.

디지털 콘텐츠 생성 애플리케이션(DCC)

디지털 콘텐츠 생성 애플리케이션(DCCs)은 디지털 콘텐츠를 생성하는 타사 제품입니다. Deadline Cloud는 Autodesk Maya, Blender, Maxon Cinema 4D와 같은 많은 DCCs와 기본적으로 통합되어 있어 DCC 내에서 작업을 제출하고 사전 구성된 소프트웨어 및 라이선스를 사용하여 서비스 관리형 플릿에서 렌더링할 수 있습니다.

작업 첨부 파일

작업 연결은 텍스처, 3D 모델, 조명 리그와 같은 작업의 일부로 자산을 업로드하고 다운로드하는 데드라인 클라우드 기능입니다. 작업 연결은 Amazon S3에 저장되므로 공유 네트워크 스토리지가 필요하지 않습니다.

작업 템플릿

작업 템플릿은 런타임 환경과 Deadline Cloud 작업의 일부로 실행되는 모든 프로세스를 정의합니다.

Deadline Cloud 제출자

Deadline Cloud 제출자는 사용자가 DCC 내에서 작업을 쉽게 제출할 수 있는 DCC용 플러그인입니다.

라이선스 엔드포인트

라이선스 엔드포인트를 사용하면 타사 제품에 대한 Deadline Cloud의 사용량 기반 라이선스를 VPC 내에서 사용할 수 있습니다. 이 모델은 사용한 만큼 지불되며 사용한 시간과 분에 대한 요금이 부과됩니다. 라이선스 엔드포인트는 팜에 연결되지 않으며 독립적으로 사용할 수 있습니다.

태그

태그는 AWS 리소스에 할당할 수 있는 레이블입니다. 각 태그는 사용자가 정의하는 키와 선택적 값으로 구성됩니다. 태그를 사용하면 용도, 소유자 또는 환경별로 리소스를 AWS 다양한 방식으로 분류할 수 있습니다.

사용량 기반 라이선스(UBL)

사용량 기반 라이선스(UBL)는 일부 타사 제품에 사용할 수 있는 온디맨드 라이선스 모델입니다. 이 모델은 사용한 만큼 지불되며 사용한 시간과 분에 대한 요금이 부과됩니다.

기한 클라우드 아키텍처 지침

이 주제에서는 Deadline Cloud를 사용하여 워크로드를 위한 안정적이고 안전하며 효율적이고 비용 효율적인 렌더 팜을 설계하고 구축하기 위한 지침과 모범 사례를 제공합니다. 이 지침을 사용하면 안정적이고 효율적인 워크로드를 구축하여 혁신에 집중하고 비용을 절감하며 고객 경험을 개선하는 데 도움이 될 수 있습니다.

이 콘텐츠는 CTO(최고 기술 책임자), 아키텍트, 개발자 및 운영 팀 팀원을 대상으로 합니다.

end-to-end 렌더링 워크플로에는 작업 생성, 자산 액세스, 작업 모니터링과 같은 프로세스의 여러 계층에 있는 솔루션이 필요합니다. Deadline Cloud는 렌더링 프로세스의 각 계층에 대해 여러 솔루션을 제공합니다. 각 계층의 Deadline Cloud 옵션 중에서 선택하여 사용 사례에 맞는 워크플로를 설계할 수 있습니다.

각 계층에 대해 사용 사례에 가장 적합한 접근 방식을 결정해야 합니다. 이러한 정의는 엄격한 시나리오 정의가 아니며 Deadline Cloud를 사용하는 유일한 방법은 아닙니다. 대신 Deadline Cloud가 비즈니스 또는 워크플로에 어떻게 적합할 수 있는지 이해하는 데 도움이 되는 상위 수준의 개념 세트입니다. Deadline Cloud 워크로드를 작업 소스, 작업 제출, 애플리케이션 관리, 애플리케이션 라이선싱, 자산 액세스, 출력 관리 및 작업자 인프라 관리 계층으로 분리할 수 있습니다.

일반적으로 아래 지정된 특정 조합을 제외하고 한 계층의 모든 시나리오를 다른 계층의 다른 시나리오와 mix-and-match시킬 수 있습니다.


Job Source
Interactive Workflow
Automated Workflow
Job Submission
Integrated Submitter
Custom Job Definition
Application Management
Conda Application Management
Custom Application Management
Application Licensing
Deadline Cloud UBL
Custom Licensing
Asset Access
Job Attachments
Custom Storage
Job Monitoring
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution
Worker Infrastructure

작업 소스

작업 소스는 새 작업이 Deadline Cloud에서 렌더링할 시스템에 들어가는 액세스 포인트입니다. 개괄적으로 보면 인간의 상호 작용과 자동화된 컴퓨터 시스템의 두 가지 주요 작업 소스가 있습니다.

대화형 워크플로

이 시나리오에서 아티스트 또는 기타 크리에이티브 역할은 Deadline Cloud 팜에서 처리할 기본 작업 생성기입니다. 일반적으로 이러한 작업의 출력은 대규모 프로젝트 또는 팀의 기본 아티팩트입니다. 업계 표준 디지털 콘텐츠 생성(DCC) 도구와 같은 소프트웨어를 사용하여 작업을 수행합니다. Deadline Cloud 팜에 수동으로 작업을 제출하고 있으며 나중에 검토를 위해 출력을 보고 있습니다. 워크스테이션 자체는에서 관리하지 않습니다 AWS.

대부분의 경우 이러한 아티스트는 워크로드 애플리케이션 및 모니터링 계층에서 Deadline Cloud 통합 제출자와 Deadline Cloud 모니터를 사용합니다.

자동화된 워크플로

이 시나리오에서 고객이 소유한 프로그래밍 시스템은 Deadline Cloud 팜의 기본 작업 생성기입니다. 이는 3D 모델 또는 스캔에서 생성된 턴테이블 비디오와 같은 소매 파이프라인의 자산 생성일 수 있습니다. 이는 스포츠용 브로드캐스트 그래픽 및 플레이어 카드의 자동 컴포지팅일 수 있습니다. 이 시나리오의 주제는 개인이 각 작업을 Deadline Cloud에 수동으로 제출하는 것이 아니라 더 큰 시스템의 일부로 작업이 생성되는 것입니다.

자동 작업에서는 Deadline Cloud 통합 제출자와 Deadline Cloud 모니터를 사용하는 것이 덜 일반적입니다. 종종 작업 정의는 사용자가 작성한 사용자 지정 애플리케이션 개발이며 작업 출력은 승인 및 배포를 위해 디지털 자산 관리(DAM) 시스템 또는 미디어 자산 관리(MAM) 시스템으로 자동 전달됩니다.

작업 제출

작업은 OpenJobDescription 템플릿을 사용하여 Deadline Cloud에 제출됩니다. OpenJobDescription은 다양한 예약 시스템 배포 간에 이동할 수 있는 배치 처리 작업을 정의하기 위한 유연한 개방형 사양입니다. 작업 정의 파일은 작업의 파라미터, 작업의 단계, 작업 입력을 기반으로 단계를 파라미터화하는 방법, 처리를 수행하기 위해 작업자에서 실행할 실제 스크립트를 설명합니다. 워크로드 제출에 대한 아이디어는 이러한 작업 정의를 생성하는 방법, 작업 정의를 생성하는 사람, 작업 정의를 제출하는 방법입니다.

DCC와 통합된 제출자

Deadline Cloud 통합 제출자는 Deadline Cloud를 업계 표준 DCC 또는 소프트웨어 패키지와 연결하는 소프트웨어입니다. 통합 제출자는 렌더링, 복합 또는 기타 워크로드의 데이터 및 구성을 Deadline Cloud에서 이해할 수 있는 작업 템플릿으로 변환하는 방법을 결정합니다. Deadline Cloud 팀 또는 소프트웨어 패키지 생성자가 많은 통합 제출자를 생성하고 유지 관리하지만 원하는 애플리케이션에 아직 없는 경우 자체 제출자를 생성하고 유지 관리할 수 있습니다. Deadline Cloud 팀에서 지원하는 DCCs 세트는 유한합니다.

대화형 워크플로에는 일반적으로 통합 제출자가 포함되지만 항상 포함되는 것은 아닙니다. 템플릿화된 자동 워크플로의 경우 일반적인 워크플로는 아티스트가 DCC에서 템플릿 작업을 설정하고 작업 번들의 일회성 내보내기를 수행하는 것입니다. 이 작업 번들은 파라미터화된 방식으로 Deadline Cloud에서 특정 종류의 작업을 실행하는 방법을 정의합니다. 이 작업 번들은 자동화를 위해 자동화 워크플로 시나리오에 통합할 수 있습니다.

사용자 지정 작업 정의

사용자 지정 애플리케이션 및 워크플로의 경우 이러한 작업 정의가 생성되어 Deadline Cloud에 제출되는 방식을 완전히 제어할 수 있습니다. 예를 들어 전자 상거래 사이트는 판매자에게 판매 중인 객체의 3D 모델을 업로드하도록 요청할 수 있습니다. 이 업로드 후 전자 상거래 플랫폼은 Deadline Cloud에 제출할 작업 정의를 동적으로 생성하여 사이트에서 사용할 수 있는 다른 3D 객체와 일치하도록 공통 조명을 사용하여 공통 백그라운드에서 턴테이블 애니메이션을 자동으로 생성할 수 있습니다. 전자 상거래 플랫폼을 개발하는 동안 소프트웨어 개발자는 작업 정의를 생성하고, 최종적으로 판매자가 제공하는 파라미터와 함께 전자 상거래 플랫폼에 임베드하고, 플랫폼 제품 업로드 워크플로 중에이 작업을 제출하도록 플랫폼을 코딩합니다.

Deadline Cloud는 github의 샘플 리포지토리에 여러 샘플 작업 정의를 제공합니다.

애플리케이션 관리

작업이 Deadline Cloud에 제출되고 작업자에게 할당되면 작업 정의의 스크립트가 작업자에 대해 실행됩니다. 대부분의 경우이 스크립트는 애플리케이션을 호출하여 렌더러, 복합, 인코딩, 필터링 또는 기타 여러 컴퓨팅 집약적 작업과 같은 실제 처리를 수행합니다. 애플리케이션 관리는 작업자가 필요한 버전의 필수 소프트웨어를 사용할 수 있도록 하는 개념입니다.

원하는 패키지 관리 시스템을 사용하여 애플리케이션을 관리할 수 있지만 Deadline Cloud는 conda 패키지를 쉽게 사용할 수 있는 다양한 도구를 제공합니다. Conda는 오픈 소스, 교차 플랫폼, 언어에 구애받지 않는 패키지 관리자 및 환경 관리 시스템입니다.

서비스 관리형 플릿(SMF)에 대한 Deadline Cloud 관리형 conda 채널

서비스 관리형 플릿을 사용하는 경우 Deadline Cloud 관리형 conda 채널은 작업에서 사용하도록 자동으로 설정 및 구성됩니다. Deadline Cloud 서비스는이 conda 채널에서 여러 파트너 DCC 애플리케이션과 렌더를 제공합니다. 자세한 내용은 Deadline Cloud 사용 설명서의 대기열 환경 생성을 참조하세요. 이러한 패키지는 Deadline Cloud 서비스에 의해 자동으로 최신 상태로 유지되며 유지 관리가 필요하지 않습니다. 이 conda 채널은 서비스 관리형 플릿을 사용할 때만 사용할 수 있으며 고객 관리형 플릿을 사용할 때는 사용할 수 없습니다.

자체 관리형 conda 채널

Deadline Cloud 관리형 conda 채널을 사용할 수 없는 경우 Deadline Cloud 플릿에 애플리케이션을 설치, 패치 및 관리하는 방법을 결정해야 합니다. 한 가지 옵션은 설정 및 유지 관리하는 conda 채널을 생성하는 것입니다. 이는 Deadline Cloud 관리형 conda 채널과 가장 밀접하게 상호 작용합니다. 예를 들어 Deadline Cloud 관리형 conda 채널의 DCC를 사용하지만 특정 DCC 플러그인이 포함된 자체 패키지를 가져올 수 있습니다. 이 프로세스에 대한 자세한 내용은 S3를 사용하여 conda 채널 생성을 참조하세요.

사용자 지정 애플리케이션 관리

애플리케이션 관리의 경우 Deadline Cloud의 요구 사항은 작업자가 작업 스크립트를 실행할 때 PATH에서 애플리케이션을 사용할 수 있다는 것입니다.

이미 Rez 패키지를 빌드하고 유지 관리하는 경우 대기열 환경을 사용하여 Rez 리포지토리에서 애플리케이션을 설치할 수 있습니다. 대기열 환경의 예는 AWS Deadline Cloud GitHub org에서 찾을 수 있습니다.

수명이 긴 작업자 또는 시스템 이미지가 있는 고객 관리형 플릿에서 애플리케이션을 이미 관리하는 경우 애플리케이션 관리에 대기열 환경이 필요하지 않습니다. 애플리케이션이 작업 사용자의 경로에 나타나는지 확인하고 작업을 제출합니다.

애플리케이션 라이선스

Deadline Cloud에서 일반적으로 실행되는 많은 워크로드에는 소프트웨어 공급업체의 소프트웨어 라이선스가 필요합니다. 이러한 애플리케이션은 종종 시트당, CPU당 또는 호스트당 라이선스가 부여됩니다. Deadline Cloud에서 타사 소프트웨어를 사용할 때 타사 라이선스 계약을 준수하는지 확인하는 것은 사용자의 책임입니다. 오픈 소스 소프트웨어, 사용자 지정 소프트웨어 또는 기타 라이선스 없는 소프트웨어를 사용하는 경우이 계층을 구성할 필요가 없습니다. Deadline Cloud는 렌더 라이선스만 지원하고 워크스테이션 라이선스는 지원하지 않습니다.

서비스 관리형 플릿 및 사용량 기반 라이선스

Deadline Cloud 서비스 관리형 플릿을 사용하는 경우 지원되는 소프트웨어에 대해 사용량 기반 라이선스(UBL)가 자동으로 구성됩니다. 서비스 관리형 플릿에서 실행되는 작업은 지원되는 애플리케이션에 대해 환경 변수를 자동으로 설정하여 Deadline Cloud 라이선스 서버를 사용하도록 지시합니다. Deadline Cloud UBL을 사용하는 경우 라이선스가 부여된 애플리케이션을 사용한 시간에 대해서만 요금이 부과됩니다.

고객 관리형 플릿 및 사용량 기반 라이선스

서비스 관리형 플릿을 사용하지 않는 경우 Deadline Cloud 사용량 기반 라이선스(UBL)도 사용할 수 있습니다. 이 시나리오에서는 Deadline Cloud 라이선스 서버에 대한 액세스를 제공하는 선택한 VPC 서브넷에 IP 주소를 제공하는 Deadline Cloud 라이선스 엔드포인트를 설정합니다. 작업자에 대해 적절한 소프트웨어별 환경 변수를 구성하고 작업자에서 해당 라이선스 엔드포인트 IP 주소로의 네트워크 연결을 구성한 후 작업자는 지원되는 소프트웨어에 대한 라이선스를 체크아웃 및 체크인할 수 있습니다. 서비스 관리형 플릿에서 UBL을 사용할 때와 동일한 라이선스에 대해 시간당 요금이 부과됩니다.

사용자 지정 라이선스

Deadline Cloud UBL에서 지원하지 않는 애플리케이션을 사용하거나 아직 유효한 기존 라이선스가 있을 수 있습니다. 이 시나리오에서는 작업자(고객 또는 서비스 관리형)에서 라이선스 서버로의 네트워크 경로를 구성할 책임이 있습니다. 사용자 지정 라이선스에 대한 자세한 내용은 섹션을 참조하세요서비스 관리형 플릿을 사용자 지정 라이선스 서버에 연결.

자산 액세스

작업이 작업자에게 제출되고 애플리케이션이 구성된 후 작업에 필요한 자산 데이터에 액세스하도록 작업자를 구성해야 합니다. 이는 3D 데이터, 텍스처 데이터, 애니메이션 데이터, 비디오 프레임 또는 작업에 사용되는 기타 유형의 데이터일 수 있습니다.

먼저 데이터가 현재 저장되어 있는 위치를 고려합니다. 워크스테이션 하드 드라이브, 사용자 공동 작업 도구, 소스 제어, 온프레미스 또는 클라우드의 공유 파일 시스템, Amazon S3 또는 기타 여러 위치에 있을 수 있습니다.

다음으로 작업자가이 데이터에 액세스하는 데 필요한 사항을 고려합니다. 이 데이터는 회사 네트워크에서만 사용할 수 있나요? 데이터에 액세스하려면 어떤 자격 증명이 필요합니까? 작업을 처리할 것으로 예상되는 작업자 수로 작업을 지원하도록 데이터 소스가 조정되나요?

작업 첨부 파일

자산 액세스 메커니즘으로 시작하는 가장 쉬운 방법은 Deadline Cloud 작업 연결입니다. 작업 첨부 파일을 사용하여 작업을 제출하면 작업에 필요한 데이터가 작업에 필요한 파일을 지정하는 매니페스트 파일과 함께 Amazon S3 버킷에 업로드됩니다. 작업 연결을 사용하면 복잡한 네트워킹 또는 공유 스토리지 설정이 필요하지 않습니다. 파일은 한 번만 업로드되므로 후속 업로드가 더 빨리 완료됩니다. 작업자가 작업 처리를 완료하면 아티스트 또는 다른 클라이언트가 다운로드할 수 있도록 출력 데이터가 Amazon S3에 업로드됩니다. 작업 연결은 모든 크기의 플릿에 맞게 확장되며 간단하고 빠르게 온보딩하고 사용할 수 있습니다.

작업 연결은 모든 상황에 가장 적합한 도구가 아닙니다. 데이터가 이미 켜져 AWS있는 경우 작업 첨부 파일은 관련 전송 시간 및 스토리지 비용을 포함하여 데이터의 사본을 추가합니다. 작업 첨부 파일을 사용하려면 작업이 제출 시 필요한 데이터를 완전히 지정하여 데이터를 업로드할 수 있어야 합니다.

작업 연결을 사용하려면 Deadline Cloud 대기열에 연결된 작업 연결 버킷이 있어야 하며 대기열 역할을 사용하여 해당 버킷에 대한 액세스를 제공해야 합니다. 기본적으로 Deadline Cloud 통합 제출자는 모두 작업 연결을 지원합니다. Deadline Cloud 통합 제출자를 사용하지 않는 경우 Deadline Cloud python 라이브러리를 통합하여 작업 첨부 파일을 사용자 지정 소프트웨어와 함께 사용할 수 있습니다.

사용자 지정 스토리지 액세스

작업 첨부 파일을 사용하지 않는 경우 작업자가 작업에 필요한 데이터에 액세스할 수 있도록 해야 합니다. Deadline Cloud는 이를 지원하고 작업을 이식 가능한 상태로 유지하기 위한 다양한 도구를 제공합니다. 아티스트와 작업자를 위한 공유 네트워크 스토리지가 이미 있거나 LucidLink와 같은 외부 서비스를 선호하거나 기타 이유가 있는 경우 사용자 지정 스토리지 솔루션을 사용할 수 있습니다.

스토리지 프로파일을 사용하여 워크스테이션 및 작업자 호스트의 파일 시스템을 모델링합니다. 각 스토리지 프로필은 시스템 구성 중 하나의 운영 체제 및 파일 시스템 레이아웃을 설명합니다. 스토리지 프로파일을 사용하면 Windows 워크스테이션을 사용하는 아티스트가 Linux 작업자가 처리하는 작업을 제출하면 Deadline Cloud는 경로 매핑이 발생하므로 작업자가 구성한 데이터 스토리지에 액세스할 수 있습니다.

Deadline Cloud 서비스 관리형 플릿을 사용하는 경우 호스트 구성 스크립트 및 VPC 리소스 엔드포인트를 사용하면 작업자가 공유 스토리지 또는 VPC에서 사용할 수 있는 기타 서비스를 직접 탑재하고 액세스할 수 있습니다.

작업 모니터링 및 출력 관리

Deadline Cloud에 제출된 작업이 성공적으로 완료되면 사람 또는 프로세스가 Deadline Cloud 외부의 비즈니스 워크플로에서 사용할 작업 출력을 다운로드합니다. 작업 실패 후 작업 로그 및 모니터링 정보는 문제를 진단하는 데 도움이 됩니다.

Deadline Cloud 모니터

Deadline Cloud Monitor 애플리케이션은 웹 및 데스크톱에서 사용할 수 있습니다. 이 솔루션은 스토리지용 작업 첨부 파일을 사용하여 다양한 DCCs에 대화형 워크플로를 사용하는 스튜디오에 가장 적합합니다. 모니터는 IAM Identity Center를 사용할 때만 지원합니다. IAM Identity Center는 소비자 자격 증명(B2C) 솔루션이 아닌 인력 자격 증명 제품이므로 많은 B2C 시나리오에 적합하지 않습니다.

사용자 지정 모니터 애플리케이션

사용자의 모니터링 환경을 사용자 지정하거나, B2C 제품을 구축하거나, Deadline Cloud를 사용하여 고도로 특수화된 시스템을 구축하려는 경우 사용자 지정 모니터링 애플리케이션을 생성하도록 선택합니다. AWS Deadline Cloud API를 사용하여 전체 워크플로의 컨텍스트와 Deadline Cloud 개념을 결합하여이 사용자 지정 애플리케이션을 생성할 수 있습니다. 예를 들어 B2C 제품에는 사용자가 설정하는 자체 프로젝트 개념이 있을 수 있으며 애플리케이션은 동일한 인터페이스에 Deadline Cloud 작업을 중첩할 수 있습니다.

자동 모니터링 솔루션

일부 시나리오에서는 Deadline Cloud에 전용 모니터링 애플리케이션이 필요하지 않습니다. 이 시나리오는 Deadline Cloud를 사용하여 스포츠 또는 뉴스용 브로드캐스트 그래픽과 같은 파이프라인의 자산을 자동으로 렌더링하는 자동화된 워크플로에서 일반적입니다. 이 시나리오에서 Deadline Cloud API 및 EventBridge 이벤트는 승인을 위해 외부 Media Asset Management 시스템과 통합하고 프로세스의 다음 단계로 데이터를 이동하는 데 사용됩니다.

작업자 인프라 관리

Deadline Cloud 플릿은 Deadline Cloud 대기열에 제출된 작업을 처리할 수 있는 서버(작업자) 그룹이며 모든 Deadline Cloud 팜의 핵심 인프라입니다.

서비스 관리형 플릿

서비스 관리형 플릿에서 Deadline Cloud는 작업자 호스트, 운영 체제, 네트워킹, 패치 적용, 오토 스케일링 및 렌더 팜 실행의 기타 요인에 대한 책임을 집니다. 애플리케이션에 필요한 시스템 사양과 함께

원하는 최소 및 최대 작업자 수를 지정하면 Deadline Cloud가 나머지를 수행합니다. 서비스 관리형 플릿은 Deadline Cloud 관리형 conda 채널을 사용하여 산업 DCC 애플리케이션을 쉽게 관리할 수 있는 유일한 플릿 옵션입니다. 또한 Deadline Cloud UBL은 서비스 관리형 플릿으로 자동으로 구성됩니다. 비용 절감을 위해 대기 및 저장 플릿, 지연 방지 워크로드는 서비스 관리형 플릿을 사용해야만 사용할 수 있습니다.

고객 관리형 플릿

작업자 호스트와 환경을 더 잘 제어해야 하는 경우 고객 관리형 플릿을 사용합니다. 고객 관리형 플릿은 Deadline Cloud 온프레미스를 사용할 때 가장 적합합니다. 자세한 내용은 Deadline Cloud 고객 관리형 플릿 생성 및 사용를 참조하세요.

아키텍처 예제

기존 프로덕션 스튜디오

기존 프로덕션 스튜디오에는 여러 물리적 위치에서 서비스 렌더링 워크로드에 이르기까지 상당한 컴퓨팅, 스토리지 및 네트워킹 인프라가 필요합니다. 각 개별 소프트웨어 패키지 및 공급업체에는 버전 관리, 호환성 및 리소스 충돌을 해결하는 동안 충족해야 하는 고유한 하드웨어, 소프트웨어, 네트워킹 및 라이선스 요구 사항이 있습니다.

아티스트 워크스테이션, 렌더 노드, 네트워크 스토리지, 라이선스 서버, 작업 대기열 시스템, 모니터링 도구 및 자산 관리에 대한 별도의 인프라 요구 사항이 있는 것이 일반적입니다. 스튜디오는 일반적으로 여러 버전의 DCC 도구, 렌더러, 플러그인 및 사용자 지정 도구를 유지 관리하는 동시에 렌더 팜 전체에서 복잡한 라이선스 계약을 관리해야 합니다. 개발, 품질 보증 및 프로덕션 환경을 고려할 때 스튜디오 인프라가 더 복잡해집니다.

서비스 관리형 옵션을 사용하는 일반적인 Deadline Cloud 배포는 다음을 통해 이러한 많은 문제를 해결하거나 줄입니다.

- 통합 DCC 제출자를 통한 대화형 워크플로 작업 제출
- Deadline Cloud 관리형 conda 채널을 통한 애플리케이션 관리
- 지원되는 소프트웨어에 대해 자동으로 구성된 사용 기반 라이선스
- 작업 연결을 통한 자산 관리
- Deadline Cloud Monitor 애플리케이션을 통한 모니터링
- 서비스 관리형 플릿을 통한 인프라 관리

이 접근 방식을 사용하면 아티스트는 복잡한 인프라를 관리하지 않고도 익숙한 DCC 도구에서 확장 가능한 클라우드 렌더 팜으로 직접 작업을 제출할 수 있습니다. 서비스는 소프트웨어 배포, 라이선스, 데이터 전송 및 인프라 규모 조정을 자동으로 처리합니다. 아티스트는 웹 인터페이스 또는 데스크톱 애플리케이션을 통해 작업을 모니터링할 수 있으며 출력은 Amazon S3에 자동으로 저장되어 쉽게 액세스할 수 있습니다.

이 구성을 통해 스튜디오는 몇 분 만에 개발 및 프로덕션 환경을 만들고, 사용하는 컴퓨팅 및 라이선스에 대해서만 비용을 지불하고, 인프라 관리가 아닌 창의적인 작업에 집중할 수 있습니다. 서비스 관리형 접근 방식은 아티스트에게 친숙한 워크플로를 유지하면서 클라우드 렌더링을 채택할 수 있는 가장 빠른 경로를 제공합니다.


Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

클라우드의 Studio

현대의 시각적 효과와 애니메이션 스튜디오는 아티스트 워크스테이션을 포함한 전체 파이프라인을 클라우드로 점점 더 많이 이동하고 있습니다. 이 접근 방식은 온프레미스 인프라의 필요성을 없애고, 글로벌 협업을 가능하게 하며, 대화형 작업과 렌더링 모두에 원활한 규모 조정을 제공합니다. 그러나 클라우드 리소스를 관리하고, 데이터에 대한 지연 시간이 짧은 액세스를 보장하고, 클라우드 기반 워크스테이션을 렌더 팜과 통합하는 데 새로운 문제가 발생합니다.

일반적인 클라우드 네이티브 스튜디오에는 이러한 모든 구성 요소에서 클라우드 워크스테이션, 공유 스토리지, 렌더링 인프라 및 소프트웨어 배포를 관리하는 통합 접근 방식이 필요합니다. 기존 접근 방식은 성능, 비용 및 유연성의 균형을 맞추는 데 어려움을 겪는 복잡한 수동 관리형 시스템을 초래하는 경우가 많습니다.

클라우드 네이티브 스튜디오의 Deadline Cloud 배포는 다음을 사용하여 구현할 수 있습니다.

- 클라우드 워크스테이션의 통합 DCC 제출자를 통한 대화형 워크플로 작업 제출
- Deadline Cloud 관리형 conda 채널을 통한 애플리케이션 관리 렌더 노드
- 지원되는 소프트웨어에 대해 자동으로 구성된 사용 기반 라이선스
- 공유 프로젝트 데이터에 FSx for Windows File Server를 사용한 사용자 지정 스토리지 액세스
- Deadline Cloud Monitor 애플리케이션을 통한 모니터링
- 서비스 관리형 플릿을 사용한 인프라 관리

이 접근 방식을 통해 아티스트는 고성능 공유 스토리지에 직접 액세스할 수 있는 클라우드 기반 워크스테이션에서 작업하고 Deadline Cloud 팜에 작업을 원활하게 제출할 수 있습니다. 스튜디오는 동일한 conda 채널을 사용하여 두 워크스테이션 모두에서 소프트웨어 배포를 관리하고 노드를 렌더링하여 일관성을 보장하고 유지 관리 오버헤드를 줄일 수 있습니다.

이 구성의 주요 이점은 다음과 같습니다.

- 어디서나 워크스테이션에 액세스할 수 있는 아티스트와의 글로벌 협업
- 워크스테이션 및 렌더 노드 전반의 일관된 소프트웨어 환경
- 워크스테이션과 렌더 노드 모두에서 액세스할 수 있는 고성능 공유 스토리지
- 대화형 및 배치 컴퓨팅 리소스의 유연한 조정
- 클라우드의 모든 스튜디오 인프라에 대한 중앙 집중식 관리

이 시나리오의 스토리지 구성에는 일반적으로 다음이 포함됩니다.

- 클라우드 워크스테이션과 Deadline Cloud 작업자 모두 액세스할 수 있는 프로젝트 데이터용 FSx for Windows File Server
- 워크스테이션과 렌더 노드 간의 경로 매핑을 관리하기 위한 Deadline Cloud의 스토리지 프로파일
- VPC 리소스 엔드포인트 및 호스트 구성 스크립트를 사용하여 Deadline Cloud 작업자에 FSx 공유 직접 탑재

이러한 클라우드 네이티브 접근 방식을 통해 스튜디오는 온프레미스 인프라를 제거할 수 있으므로 익숙한 아티스트 워크플로를 유지하면서 모든 규모의 프로젝트를 신속하게 확장할 수 있습니다. 서비스 관리형 리소스와 고객 관리형 리소스를 혼합하여 관리 용이성과 특정 성능 요구 사항에 맞게 최적화할 수 있는 유연성을 제공합니다.

스튜디오는 Deadline Cloud와 함께 클라우드 워크스테이션을 활용하여 소규모 팀에서 대규모 프로덕션으로 원활하게 확장되는 완전 통합되고 전 세계적으로 액세스 가능한 프로덕션 파이프라인을 달성할 수 있습니다.

ECommerce 자동화

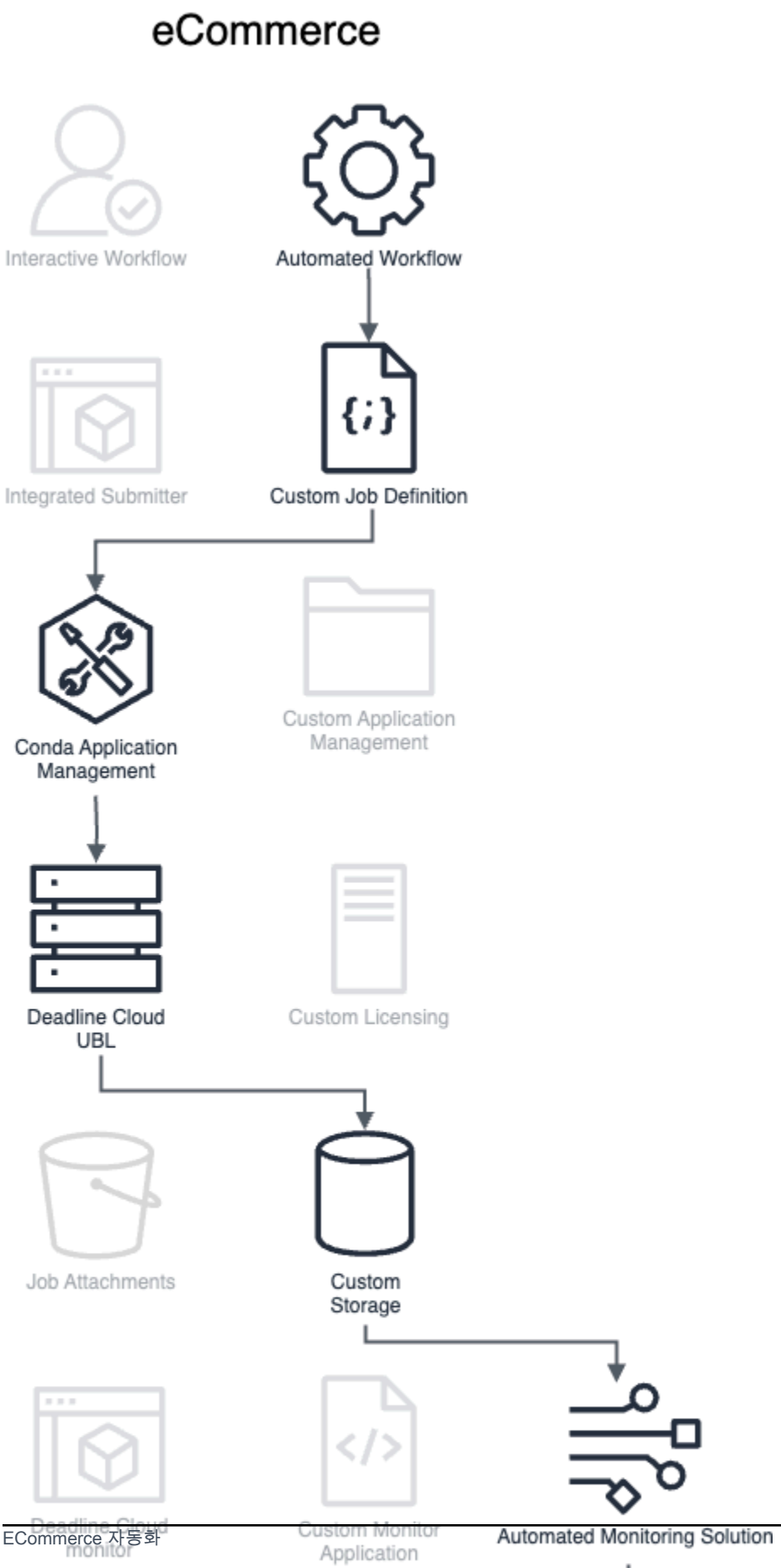
최신 전자 상거래 플랫폼에는 수백만 개의 항목에 걸쳐 풍부한 제품 시각화를 제공하기 위해 대규모로 자동화된 자산 생성이 필요합니다. 기존 접근 방식은 대량의 3D 모델을 표준화된 제품 미디어로 처리하는 데 상당한 인프라 투자가 필요하므로 처리 백로그를 생성하는 과소 프로비저닝된 시스템 또는 유휴 용량이 있는 과도 프로비저닝된 시스템이 발생하는 경우가 많습니다.

일반적인 자동 전자 상거래 워크플로는 제품 업로드 처리, 3D 모델 검증, 렌더 팜 관리, 출력 처리 및 제품 정보 시스템과의 통합을 처리해야 합니다. 이러한 워크플로를 관리하려면 일반적으로 일관된 품질을 보장하고 대규모로 비용 효율성을 유지하면서 여러 렌더링 애플리케이션, 컴퓨팅 리소스 및 데이터 처리 파이프라인을 조정해야 합니다.

전자 상거래 자동화를 위한 Deadline Cloud 배포는 다음을 사용하여 구현할 수 있습니다.

- 기존 전자 상거래 수집 애플리케이션의 사용자 지정 API 통합을 통한 자동화된 워크플로 작업 제출
- 표준화된 제품 시각화에 맞게 조정된 사용자 지정 작업 정의
- Deadline Cloud 관리형 conda 채널을 통한 애플리케이션 관리
- 지원되는 소프트웨어에 대해 자동으로 구성된 사용 기반 라이선스
- 자산 관리를 위한 직접 Amazon S3 통합
- 기존 제품 관리 시스템과 통합된 사용자 지정 모니터링 애플리케이션
- 탄력적 조정을 위한 서비스 관리형 플릿

이 접근 방식을 사용하면 하루에 수천 개의 제품을 처리할 수 있으므로 턴테이블 애니메이션과 같은 표준화된 제품 시각화가 자동으로 생성됩니다. 서비스 관리형 인프라는 작업자 재사용 및 최적화된 애플리케이션 배포를 통해 비용 효율성을 유지하면서 가변 수요에 맞게 자동으로 확장됩니다.


eCommerce
Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

Whitelabel/OEM/B2C 고객

기존 디지털 콘텐츠 생성(DCC) 소프트웨어에서는 일반적으로 사용자가 워크스테이션에서 로컬로 자체 렌더링 인프라 또는 프로세스 렌더링을 유지 관리해야 하므로 상당한 하드웨어 투자가 발생하거나 크리에이티브 워크플로를 방해하는 대기 시간이 길어집니다. 소프트웨어 공급업체의 경우 클라우드 렌더링 기능을 제공하려면 전통적으로 복잡한 인프라 및 결제 시스템을 구축하고 유지 관리해야 했습니다.

B2C 소프트웨어에 통합된 Deadline Cloud 배포를 사용하면 사용자의 친숙한 인터페이스 내에서 직접 원활한 클라우드 렌더링이 가능합니다. 이 통합은 다음을 결합합니다.

- DCC 애플리케이션에 내장된 대화형 워크플로 작업 제출
- 렌더 애플리케이션 배포를 위한 Deadline Cloud 관리형 conda 채널
- 자동으로 구성된 사용량 기반 라이선스
- 공급업체 관리형 스토리지를 사용한 작업 연결을 통한 자산 관리
- DCC 인터페이스에 직접 통합된 사용자 지정 모니터링
- 사용자 간에 공유되는 서비스 관리형 플릿

이 접근 방식을 사용하면 최종 사용자가 계정, 인프라 또는 복잡한 설정을 관리하지 않고도 소프트웨어 내에서 클릭 한 번으로 클라우드에 렌더링을 제출할 수 있습니다. 소프트웨어 공급업체는 다음과 같은 다중 테넌트 환경을 유지합니다.

- 사용자가 기존 소프트웨어 자격 증명을 통해 인증
- 작업은 사용자별 전용 대기열로 자동 라우팅됩니다.
- IAM 제어 스토리지 접두사를 사용하여 자산을 안전하게 격리합니다.
- 결제는 공급업체의 기존 시스템을 통해 처리됩니다.
- 작업 상태 및 출력이 사용자의 애플리케이션으로 직접 스트리밍됩니다.

공유 플릿 접근 방식은 웜 풀 작업자를 유지하고 시작 시간을 최소화하는 동시에 사용자 기반 전반의 리소스 사용률을 극대화하여 최적의 성능을 보장합니다. 이 구성을 통해 소프트웨어 공급업체는 추가 설정 또는 계정이 필요한 별도의 서비스가 아닌 원활한 제품 기능으로 클라우드 렌더링을 제공할 수 있습니다.

최종 사용자는 다음과 같은 이점을 누릴 수 있습니다.

- 익숙한 인터페이스에서 원클릭 제출

- 인프라 관리 없이 Pay-as-you-go 요금
- 공유 인프라를 통한 빠른 작업 시작 시간
- 완료된 렌더의 자동 다운로드 및 구성
- 모든 플랫폼에서 일관된 경험

이 통합 패턴을 통해 소프트웨어 공급업체는 전체 사용자 기반에 엔터프라이즈급 렌더링 기능을 제공하는 동시에 애플리케이션 고유의 느낌을 주는 간단하고 소비자 친화적인 환경을 유지할 수 있습니다.


Whitelabel B2C Customer
Interactive Workflow
Automated Workflow
Integrated Submitter
Custom Job Definition
Conda Application Management
Custom Application Management
Deadline Cloud UBL
Custom Licensing
Job Attachments
Custom Storage
Deadline Cloud monitor
Custom Monitor Application
Automated Monitoring Solution

Deadline Cloud 워크로드란 무엇입니까?

AWS Deadline Cloud를 사용하면 클라우드에서 애플리케이션을 실행하는 작업을 제출하고 비즈니스에 중요한 콘텐츠 또는 인사이트를 생성하기 위한 데이터를 처리할 수 있습니다. Deadline Cloud는 시각적 컴퓨팅 파이프라인의 요구 사항에 맞게 설계되었지만 다른 많은 사용 사례에 적용할 수 있는 사양인 Open Job Description(OpenJD)을 작업 템플릿의 구문으로 사용합니다. 일부 예제 워크로드에는 컴퓨터 그래픽 렌더링, 물리 시뮬레이션 및 사진 측정이 포함됩니다.

워크로드는 사용자가 CLI 또는 자동 생성된 GUI를 사용하여 대기열에 제출하는 간단한 작업 번들에서 애플리케이션 정의 워크로드에 대한 작업 번들을 동적으로 생성하는 통합 제출자 플러그인으로 확장됩니다.

프로덕션에서 워크로드가 발생하는 방식

프로덕션 컨텍스트의 워크로드와 Deadline Cloud를 통해 워크로드를 지원하는 방법을 이해하려면 워크로드가 어떻게 될지 고려하세요. 프로덕션에는 시각적 효과, 애니메이션, 게임, 제품 카탈로그 이미지, BIM(빌딩 정보 모델링)을 위한 3D 재구성 등을 생성하는 작업이 포함될 수 있습니다. 이 콘텐츠는 일반적으로 다양한 소프트웨어 애플리케이션과 사용자 지정 스크립팅을 실행하는 아티스트 또는 기술 전문가 팀이 생성합니다. 팀 구성원은 프로덕션 파이프라인을 사용하여 서로 간에 데이터를 전달합니다. 파이프라인에서 수행하는 많은 작업에는 사용자의 워크스테이션에서 실행되는 경우 며칠이 걸리는 집약적인 계산이 포함됩니다.

이러한 프로덕션 파이프라인의 몇 가지 작업 예는 다음과 같습니다.

- 사진 측정 애플리케이션을 사용하여 영화 세트의 사진을 처리하여 텍스처 디지털 메시를 재구성합니다.
- 3D 장면에서 파티클 시뮬레이션을 실행하여 TV 쇼의 폭발적인 시각적 효과에 디테일 계층을 추가합니다.
- 게임 레벨에 대한 데이터를 외부 릴리스에 필요한 형식으로 쿠킹하고 최적화 및 압축 설정을 적용합니다.
- 색상, 배경 및 조명의 변화를 포함하여 제품 카탈로그의 이미지 세트를 렌더링합니다.
- 3D 모델에서 사용자 지정 개발 스크립트를 실행하여 영화 감독이 사용자 지정으로 빌드하고 승인한 모습을 적용합니다.

이러한 작업에는 예술적 결과를 얻거나 출력 품질을 미세 조정하기 위해 조정할 수 있는 많은 파라미터가 포함됩니다. 애플리케이션 내에서 로컬로 프로세스를 실행하기 위한 버튼 또는 메뉴를 사용하여 이러한 파라미터 값을 선택하는 GUI가 있는 경우가 많습니다. 사용자가 프로세스를 실행하면 애플리케

이션과 호스트 컴퓨터 자체를 사용하여 다른 작업을 수행할 수 없습니다. 메모리에서 애플리케이션 상태를 사용하고 호스트 컴퓨터의 모든 CPU 및 메모리 리소스를 사용할 수 있기 때문입니다.

대부분의 경우 프로세스는 빠릅니다. 프로덕션 과정에서 품질 및 복잡성에 대한 요구 사항이 증가하면 프로세스 속도가 느려집니다. 개발 중에 30초가 걸린 캐릭터 테스트는 최종 프로덕션 캐릭터에 적용하면 3시간으로 쉽게 전환될 수 있습니다. 이러한 발전을 통해 GUI 내에서 수명 주기 시작한 워크로드는 너무 커져서 맞지 않을 수 있습니다. Deadline Cloud로 포팅하면 워크스테이션을 완전히 다시 제어하고 Deadline Cloud 모니터에서 더 많은 반복을 추적할 수 있으므로 이러한 프로세스를 실행하는 사용자의 생산성을 높일 수 있습니다.

Deadline Cloud에서 워크로드에 대한 지원을 개발할 때 목표로 해야 할 두 가지 수준의 지원이 있습니다.

- 병렬 처리나 속도 향상 없이 사용자 워크스테이션에서 Deadline Cloud 팜으로 워크로드 오프로드. 이렇게 하면 팜에서 사용 가능한 컴퓨팅 리소스를 충분히 활용하지 못할 수 있지만, 긴 작업을 배치 처리 시스템으로 전환하면 사용자가 자신의 워크스테이션에서 더 많은 작업을 수행할 수 있습니다.
- Deadline Cloud 팜의 수평적 규모를 활용하여 빠르게 완료할 수 있도록 워크로드의 병렬 처리를 최적화합니다.

워크로드를 병렬로 실행하는 방법이 명백한 경우가 있습니다. 예를 들어 컴퓨터 그래픽 렌더링의 각 프레임은 독립적으로 수행할 수 있습니다. 그러나이 병렬 처리를 멈춰서는 안 됩니다. 대신 장기 실행 워크로드를 Deadline Cloud로 오프로드하면 워크로드를 분할할 수 있는 명확한 방법이 없더라도 상당한 이점을 얻을 수 있습니다.

워크로드의 구성 요소

Deadline Cloud 워크로드를 지정하려면 Deadline Cloud CLI를 사용하여 사용자가 대기열에 제출하는 작업 번들을 구현합니다. 작업 번들을 생성하는 작업은 대부분 작업 템플릿을 작성하는 것이지만 워크로드에 필요한 애플리케이션을 제공하는 방법과 같은 요소가 더 많습니다. 다음은 Deadline Cloud에 대한 워크로드를 정의할 때 고려해야 할 필수 사항입니다.

- 실행할 애플리케이션입니다. 작업은 애플리케이션 프로세스를 시작할 수 있어야 하므로 사용 가능한 애플리케이션과 부동 라이선스 서버에 대한 액세스와 같이 애플리케이션이 사용하는 라이선스를 설치해야 합니다. 이는 일반적으로 팜 구성의 일부이며 작업 번들 자체에 포함되지 않습니다.
  - 대기열 환경을 사용하여 작업 구성
  - 고객 관리형 플릿을 라이선스 엔드포인트에 연결
- 작업 파라미터 정의. 작업을 제출하는 사용자 경험은 제공하는 파라미터의 영향을 크게 받습니다. 예제 파라미터에는 데이터 파일, 디렉터리 및 애플리케이션 구성이 포함됩니다.

  - 작업 번들의 파라미터 값 요소
- 파일 데이터 흐름. 작업이 실행되면 사용자가 제공한 파일에서 입력을 읽은 다음 출력을 새 파일로 씁니다. 작업 연결 및 경로 매핑 기능을 사용하려면 작업이 이러한 입력 및 출력에 대한 디렉터리 또는 특정 파일의 경로를 지정해야 합니다.
  - 작업에서 파일 사용
- 단계 스크립트입니다. 단계 스크립트는 올바른 명령줄 옵션으로 애플리케이션 바이너리를 실행하여 제공된 작업 파라미터를 적용합니다. 또한 워크로드 데이터 파일에 상대 경로 참조 대신 절대 경로가 포함된 경우 경로 매핑과 같은 세부 정보를 처리합니다.
  - 작업 번들에 대한 작업 템플릿 요소

워크로드 이식성

워크로드는 작업을 제출할 때마다 워크로드를 변경하지 않고 여러 시스템에서 실행할 수 있는 경우에 이식이 가능합니다. 예를 들어, 서로 다른 공유 파일 시스템이 탑재된 서로 다른 렌더 팜 또는 Linux 또는와 같은 서로 다른 운영 체제에서 실행될 수 있습니다Windows. 휴대용 작업 번들을 구현하면 사용자가 특정 팜에서 작업을 실행하거나 다른 사용 사례에 맞게 조정하는 것이 더 쉽습니다.

다음은 작업 번들을 이식할 수 있는 몇 가지 방법입니다.

- 작업 번들의 PATH 작업 파라미터 및 자산 참조를 사용하여 워크로드에 필요한 입력 데이터 파일을 완전히 지정합니다. 이 접근 방식을 사용하면 공유 파일 시스템을 기반으로 하는 팜과 Deadline Cloud 작업 연결 기능과 같이 입력 데이터를 복사하는 팜에 작업을 이식할 수 있습니다.
- 작업의 입력 파일에 대한 파일 경로 참조를 다른 운영 체제에서 재배치 및 사용할 수 있도록 설정합니다. 예를 들어 사용자가 Linux플릿에서 실행하기 위해 Windows 워크스테이션에서 작업을 제출하는 경우입니다.
  - 상대 파일 경로 참조를 사용하므로 해당 참조가 포함된 디렉터리가 다른 위치로 이동해도 참조는 여전히 확인됩니다. Blender와 같은 일부 애플리케이션은 상대 경로와 절대 경로 중에서 선택할 수 있습니다.
  - 상대 경로를 사용할 수 없는 경우는 OpenJD 경로 매핑 메타데이터를 지원하고 Deadline Cloud가 작업에 파일을 제공하는 방식에 따라 절대 경로를 변환합니다.
- 이동식 스크립트를 사용하여 작업에서 명령을 구현합니다. Python과 bash는 이러한 방식으로 사용할 수 있는 스크립팅 언어의 두 가지 예입니다. 플릿의 모든 작업자 호스트에 둘 다 제공하는 것을 고려해야 합니다.
  - python 또는와 같은 스크립트 인터프리터 바이너리를 스크립트 파일 이름과 bash함께 인수로 사용합니다. 이 접근 방식은에서 실행 비트가 설정된 스크립트 파일을 사용하는 것과 Windows비교하여를 포함한 모든 운영 체제에서 작동합니다Linux.
  - 다음 방법을 적용하여 휴대용 bash 스크립트를 작성합니다.

    - 템플릿 경로 파라미터를 작은따옴표로 확장하여 공백과 경로 구분자가 있는 Windows 경로를 처리합니다.
    - 에서 실행할 때 MinGW 자동 경로 변환과 관련된 문제가 있는지 Windows확인합니다. 예를 들어와 같은 AWS CLI 명령을와 유사한 명령`aws logs tail /aws/deadline/...`으로 변환`aws logs tail "C:/Program Files/Git/aws/deadline/..."`하고 로그를 올바르게 테일링하지 않습니다. 이 동작을 끄`MSYS_NO_PATHCONV=1`도록 변수를 설정합니다.
    - 대부분의 경우 모든 운영 체제에서 동일한 코드가 작동합니다. 코드가 서로 달라야 하는 경우 `if/else` 구문을 사용하여 사례를 처리합니다.

      ```
      if [[ "$(uname)" == MINGW* || "$(uname -s)" == MSYS_NT* ]]; then
          # Code for Windows
      elif [[ "$(uname)" == Darwin ]]; then
          # Code for MacOS
      else
          # Code for Linux and other operating systems
      fi
      ```

    - 를 사용하여 이동식 Python 스크립트`pathlib`를 작성하여 파일 시스템 경로 차이를 처리하고 운영별 기능을 피할 수 있습니다. Python 설명서에는 신호 라이브러리 설명서와 같이 이에 대한 주석이 포함되어 있습니다. Linux특정 기능 지원은 “가용성: Linux”로 표시됩니다.
- 작업 파라미터를 사용하여 애플리케이션 요구 사항을 지정합니다. 팜 관리자가 대기열 환경에 적용할 수 있는 일관된 규칙을 사용합니다.
    - 예를 들어 작업에 `CondaPackages` 및/또는 `RezPackages` 파라미터를 작업에 필요한 애플리케이션 패키지 이름과 버전을 나열하는 기본 파라미터 값과 함께 사용할 수 있습니다. 그런 다음 샘플 conda 또는 Rez 대기열 환경 중 하나를 사용하여 작업에 대한 가상 환경을 제공할 수 있습니다.

Deadline Cloud 리소스 시작하기

AWS Deadline Cloud에 대한 사용자 지정 솔루션 생성을 시작하려면 리소스를 설정해야 합니다. 여기에는 팜, 팜에 대해 하나 이상의 대기열, 대기열에 서비스를 제공할 작업자 플릿이 포함됩니다. Deadline Cloud 콘솔을 사용하여 리소스를 생성하거나를 사용할 수 있습니다 AWS Command Line Interface.

이 자습서에서는 AWS CloudShell 를 사용하여 간단한 개발자 팜을 생성하고 작업자 에이전트를 실행합니다. 그런 다음 파라미터 및 첨부 파일을 사용하여 간단한 작업을 제출 및 실행하고, 서비스 관리형 플릿을 추가하고, 완료되면 팜 리소스를 정리할 수 있습니다.

다음 섹션에서는 Deadline Cloud의 다양한 기능과 이러한 기능이 작동하고 함께 작동하는 방법을 소개합니다. 다음 단계는 새로운 워크로드 및 사용자 지정을 개발하고 테스트하는 데 유용합니다.

콘솔을 사용하여 팜을 설정하는 지침은 Deadline Cloud 사용 설명서의 시작하기를 참조하세요.

주제

- Deadline Cloud 팜 생성
- Deadline Cloud 작업자 에이전트 실행
- Deadline Cloud로 제출
- Deadline Cloud에서 작업 첨부 파일이 있는 작업 제출
- Deadline Cloud의 개발자 팜에 서비스 관리형 플릿 추가
- Deadline Cloud에서 팜 리소스 정리

Deadline Cloud 팜 생성

AWS Deadline Cloud에서 개발자 팜 및 대기열 리소스를 생성하려면 다음 절차와 같이 AWS Command Line Interface (AWS CLI)를 사용합니다. 또한 AWS Identity and Access Management (IAM) 역할과 고객 관리형 플릿(CMF)을 생성하고 플릿을 대기열에 연결합니다. 그런 다음를 구성 AWS CLI 하고 팜이 지정된 대로 설정되고 작동하는지 확인할 수 있습니다.

이 팜을 사용하여 Deadline Cloud의 기능을 탐색한 다음 새 워크로드, 사용자 지정 및 파이프라인 통합을 개발하고 테스트할 수 있습니다.

팜을 생성하려면

1. 세션을 엽니다 AWS CloudShell. CloudShell 창을 사용하여 AWS Command Line Interface (AWS CLI) 명령을 입력하여이 자습서의 예제를 실행합니다. 계속 진행하면서 CloudShell 창을 열어 둡니다.
2. 팜의 이름을 생성하고 해당 팜 이름을에 추가합니다`~/.bashrc`. 이렇게 하면 다른 터미널 세션에서 사용할 수 있습니다.

```
echo "DEV_FARM_NAME=DeveloperFarm" >> ~/.bashrc
source ~/.bashrc
```

3. 팜 리소스를 생성하고에 팜 ID를 추가합니다`~/.bashrc`.

```
aws deadline create-farm \
    --display-name "$DEV_FARM_NAME"

echo "DEV_FARM_ID=\$(aws deadline list-farms \
        --query \"farms[?displayName=='\$DEV_FARM_NAME'].farmId \
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

4. 대기열 리소스를 생성하고 대기열 ID를에 추가합니다. `~/.bashrc`.

```
aws deadline create-queue \
    --farm-id $DEV_FARM_ID \
    --display-name "$DEV_FARM_NAME Queue" \
    --job-run-as-user '{"posix": {"user": "job-user", "group": "job-group"},
 "runAs":"QUEUE_CONFIGURED_USER"}'

echo "DEV_QUEUE_ID=\$(aws deadline list-queues \
        --farm-id \$DEV_FARM_ID \
        --query \"queues[?displayName=='\$DEV_FARM_NAME Queue'].queueId \
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

5. 플릿에 대한 IAM 역할을 생성합니다. 이 역할은 대기열에서 작업을 실행하는 데 필요한 보안 자격 증명을 플릿의 작업자 호스트에 제공합니다.

```
aws iam create-role \
    --role-name "${DEV_FARM_NAME}FleetRole" \
    --assume-role-policy-document \
        '{
```

```
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "credentials.deadline.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }'
aws iam put-role-policy \
    --role-name "${DEV_FARM_NAME}FleetRole" \
    --policy-name WorkerPermissions \
    --policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "deadline:AssumeFleetRoleForWorker",
                        "deadline:UpdateWorker",
                        "deadline:DeleteWorker",
                        "deadline:UpdateWorkerSchedule",
                        "deadline:BatchGetJobEntity",
                        "deadline:AssumeQueueRoleForWorker"
                    ],
                    "Resource": "*",
                    "Condition": {
                        "StringEquals": {
                            "aws:PrincipalAccount": "${aws:ResourceAccount}"
                        }
                    }
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "logs:CreateLogStream"
                    ],
                    "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
                    "Condition": {
                        "StringEquals": {
                            "aws:PrincipalAccount": "${aws:ResourceAccount}"
```

```
                        }
                    }
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "logs:PutLogEvents",
                        "logs:GetLogEvents"
                    ],
                    "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
                    "Condition": {
                        "StringEquals": {
                            "aws:PrincipalAccount": "${aws:ResourceAccount}"
                        }
                    }
                }
            ]
        }'
```

6. 고객 관리형 플릿(CMF)을 생성하고 해당 플릿 ID를에 추가합니다~/.bashrc.

```
FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
        --query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"
aws deadline create-fleet \
    --farm-id $DEV_FARM_ID \
    --display-name "$DEV_FARM_NAME CMF" \
    --role-arn $FLEET_ROLE_ARN \
    --max-worker-count 5 \
    --configuration \
        '{
            "customerManaged": {
                "mode": "NO_SCALING",
                "workerCapabilities": {
                    "vCpuCount": {"min": 1},
                    "memoryMiB": {"min": 512},
                    "osFamily": "linux",
                    "cpuArchitectureType": "x86_64"
                }
            }
        }'

echo "DEV_CMF_ID=\$(aws deadline list-fleets \
        --farm-id \$DEV_FARM_ID \
        --query \"fleets[?displayName=='\$DEV_FARM_NAME CMF'].fleetId \
```

```
        | [0]\" --output text)" >> ~/.bashrc
source ~/.bashrc
```

7. CMF를 대기열에 연결합니다.

```
aws deadline create-queue-fleet-association \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --fleet-id $DEV_CMF_ID
```

8. Deadline Cloud 명령줄 인터페이스를 설치합니다.

```
pip install deadline
```

9. 기본 팜을 팜 ID로 설정하고 대기열을 이전에 생성한 대기열 ID로 설정하려면 다음 명령을 사용합니다.

```
deadline config set defaults.farm_id $DEV_FARM_ID
deadline config set defaults.queue_id $DEV_QUEUE_ID
```

10. (선택 사항) 팜이 사양에 따라 설정되었는지 확인하려면 다음 명령을 사용합니다.

    - 모든 팜 나열 - **`deadline farm list`**
    - 기본 팜의 모든 대기열 나열 - **`deadline queue list`**
    - 기본 팜의 모든 플릿 나열 - **`deadline fleet list`**
    - 기본 팜 가져오기 - **`deadline farm get`**
    - 기본 대기열 가져오기 - **`deadline queue get`**
    - 기본 대기열과 연결된 모든 플릿 가져오기 - **`deadline fleet get`**

다음 단계

팜을 생성한 후 플릿의 호스트에서 Deadline Cloud 작업자 에이전트를 실행하여 작업을 처리할 수 있습니다. Deadline Cloud 작업자 에이전트 실행을(를) 참조하세요.

Deadline Cloud 작업자 에이전트 실행

개발자 팜의 대기열에 제출하는 작업을 실행하려면 먼저 AWS 워커 호스트에서 Deadline Cloud 워커 에이전트를 개발자 모드로 실행해야 합니다.

이 자습서의 나머지 부분에서는 두 개의 AWS CloudShell 탭을 사용하여 개발자 팜에서 AWS CLI 작업을 수행합니다. 첫 번째 탭에서 작업을 제출할 수 있습니다. 두 번째 탭에서 작업자 에이전트를 실행할 수 있습니다.

> **Note**
>
> CloudShell 세션을 20분 이상 유휴 상태로 두면 시간이 초과되어 작업자 에이전트가 중지됩니다. 작업자 에이전트를 다시 시작하려면 다음 절차의 지침을 따릅니다.

작업자 에이전트를 시작하려면 먼저 Deadline Cloud 팜, 대기열 및 플릿을 설정해야 합니다. Deadline Cloud 팜 생성을(를) 참조하세요.

개발자 모드에서 작업자 에이전트를 실행하려면

1. 첫 번째 CloudShell 탭에서 팜을 열어 둔 상태에서 두 번째 CloudShell 탭을 연 다음 `demoenv-logs` 및 `demoenv-persist` 디렉터리를 생성합니다.

   ```
   mkdir ~/demoenv-logs
   mkdir ~/demoenv-persist
   ```

2. PyPI에서 Deadline Cloud 작업자 에이전트 패키지를 다운로드하여 설치합니다.

   > **Note**
   >
   > 에서는 에이전트 파일을 Python의 글로벌 사이트 패키지 디렉터리에 설치Windows해야 합니다. Python 가상 환경은 현재 지원되지 않습니다.

   ```
   python -m pip install deadline-cloud-worker-agent
   ```

3. 작업자 에이전트가 작업 실행을 위한 임시 디렉터리를 생성할 수 있도록 하려면 디렉터리를 생성합니다.

   ```
   sudo mkdir /sessions
   sudo chmod 750 /sessions
   sudo chown cloudshell-user /sessions
   ```

4. 에 추가`DEV_CMF_ID`한 변수 `DEV_FARM_ID` 및를 사용하여 개발자 모드에서 Deadline Cloud 작업자 에이전트를 실행합니다`~/.bashrc`.

```
deadline-worker-agent \
    --farm-id $DEV_FARM_ID \
    --fleet-id $DEV_CMF_ID \
    --run-jobs-as-agent-user \
    --logs-dir ~/demoenv-logs \
    --persistence-dir ~/demoenv-persist
```

작업자 에이전트가 UpdateWorkerSchedule API 작업을 초기화한 다음 폴링하면 다음 출력이 표시됩니다.

```
INFO    Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] # Worker Agent starting
[2024-03-27 15:51:01,292][INFO    ] AgentInfo
Python Interpreter: /usr/bin/python3
Python Version: 3.9.16 (main, Sep  8 2023, 00:00:00)  - [GCC 11.4.1 20230605 (Red
 Hat 11.4.1-2)]
Platform: linux
...
[2024-03-27 15:51:02,528][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params={'assignedSessions': {}, 'cancelSessionActions': {},
 'updateIntervalSeconds': 15} ...
[2024-03-27 15:51:17,635][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
[2024-03-27 15:51:32,756][INFO    ] # API.Resp # [deadline:UpdateWorkerSchedule]
(200) params=(Duplicate removed, see previous response) ...
...
```

5. 첫 번째 CloudShell 탭을 선택한 다음 플릿의 작업자를 나열합니다.

```
deadline worker list --fleet-id $DEV_CMF_ID
```

다음과 같은 출력이 표시됩니다.

```
Displaying 1 of 1 workers starting at 0

- workerId: worker-8c9af877c8734e89914047111f
  status: STARTED
  createdAt: 2023-12-13 20:43:06+00:00
```

프로덕션 구성에서 Deadline Cloud 작업자 에이전트는 호스트 시스템에서 여러 사용자 및 구성 디렉터리를 관리 사용자로 설정해야 합니다. 사용자만 액세스할 수 있는 자체 개발 팜에서 작업을 실행하므로 이러한 설정을 재정의할 수 있습니다.

다음 단계

이제 작업자 에이전트가 작업자 호스트에서 실행 중이므로 작업자에게 작업을 보낼 수 있습니다. 다음을 할 수 있습니다.

- Deadline Cloud로 제출 간단한 OpenJD 작업 번들을 사용합니다.
- Deadline Cloud에서 작업 첨부 파일이 있는 작업 제출는 서로 다른 운영 체제를 사용하여 워크스테이션 간에 파일을 공유합니다.

Deadline Cloud로 제출

작업자 호스트에서 Deadline Cloud 작업을 실행하려면 Open Job Description(OpenJD) 작업 번들을 생성하고 사용하여 작업을 구성합니다. 번들은 예를 들어 작업에 대한 입력 파일과 작업의 출력을 쓸 위치를 지정하여 작업을 구성합니다. 이 주제에는 작업 번들을 구성할 수 있는 방법의 예가 포함되어 있습니다.

이 섹션의 절차를 따르려면 먼저 다음을 완료해야 합니다.

- Deadline Cloud 팜 생성
- Deadline Cloud 작업자 에이전트 실행

AWS Deadline Cloud를 사용하여 작업을 실행하려면 다음 절차를 사용합니다. 첫 번째 AWS CloudShell 탭을 사용하여 개발자 팜에 작업을 제출합니다. 두 번째 CloudShell 탭을 사용하여 작업자 에이전트 출력을 봅니다.

주제

- simple_job 샘플 제출
- 파라미터와 simple_job 함께 제출
- 파일 I/O를 사용하여 simple_file_job 작업 번들 생성
- 다음 단계

simple_job 샘플 제출

팜을 생성하고 작업자 에이전트를 실행한 후 simple_job 샘플을 Deadline Cloud에 제출할 수 있습니다.

샘플을 Deadline Cloudsimple_job에 제출하려면

1. 첫 번째 CloudShell 탭을 선택합니다.
2. GitHub에서 샘플을 다운로드합니다.

```
cd ~
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
```

3. 작업 번들 샘플 디렉터리로 이동합니다.

```
cd ~/deadline-cloud-samples/job_bundles/
```

4. simple_job 샘플을 제출합니다.

```
deadline bundle submit simple_job
```

5. 두 번째 CloudShell 탭을 선택하여 호출BatchGetJobEntities, 세션 가져오기 및 세션 작업 실행에 대한 로깅 출력을 봅니다.

```
...
[2024-03-27 16:00:21,846][INFO    ] # Session.Starting
 # [session-053d77cef82648fe2] Starting new Session.
 [queue-3ba4ff683ff54db09b851a2ed8327d7b/job-d34cc98a6e234b6f82577940ab4f76c6]
[2024-03-27 16:00:21,853][INFO    ] # API.Req # [deadline:BatchGetJobEntity]
 resource={'farm-id': 'farm-3e24cfc9bbcd423e9c1b6754bc1',
 'fleet-id': 'fleet-246ee60f46d44559b6cce010d05', 'worker-id':
 'worker-75e0fce9c3c344a69bff57fcd83'} params={'identifiers': [{'jobDetails':
 {'jobId': 'job-d34cc98a6e234b6f82577940ab4'}}]} request_url=https://
scheduling.deadline.us-west-2.amazonaws.com/2023-10-12/farms/
farm-3e24cfc9bbcd423e /fleets/fleet-246ee60f46d44559b1 /workers/worker-
 75e0fce9c3c344a69b /batchGetJobEntity
[2024-03-27 16:00:22,013][INFO    ] # API.Resp # [deadline:BatchGetJobEntity](200)
 params={'entities': [{'jobDetails': {'jobId': 'job-d34cc98a6e234b6f82577940ab6',
 'jobRunAsUser': {'posix': {'user': 'job-user', 'group': 'job-group'},
 'runAs': 'QUEUE_CONFIGURED_USER'}, 'logGroupName': '/aws/deadline/
farm-3e24cfc9bbcd423e9c1b6754bc1/queue-3ba4ff683ff54db09b851a2ed83', 'parameters':
 '*REDACTED*', 'schemaVersion': 'jobtemplate-2023-09'}}], 'errors': []}
 request_id=a3f55914-6470-439e-89e5-313f0c6
```

```
[2024-03-27 16:00:22,013][INFO    ] # Session.Add #
 [session-053d77cef82648fea9c69827182] Appended new SessionActions.
 (ActionIds: ['sessionaction-053d77cef82648fea9c69827182-0'])
 [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,014][WARNING ] # Session.User #
 [session-053d77cef82648fea9c69827182] Running as the Worker Agent's
 user. (User: cloudshell-user) [queue-3ba4ff683ff54db09b851a2ed8b/job-
d34cc98a6e234b6f82577940ac6]
[2024-03-27 16:00:22,015][WARNING ] # Session.AWSCreds #
 [session-053d77cef82648fea9c69827182] AWS Credentials are not available: Queue has
 no IAM Role. [queue-3ba4ff683ff54db09b851a2ed8b/job-d34cc98a6e234b6f82577940ab6]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
 [session-053d77cef82648fea9c69827182] Logs streamed to: AWS CloudWatch
 Logs. (LogDestination: /aws/deadline/farm-3e24cfc9bbcd423e9c1b6754bc1/
queue-3ba4ff683ff54db09b851a2ed83/session-053d77cef82648fea9c69827181)
 [queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
[2024-03-27 16:00:22,026][INFO    ] # Session.Logs #
 [session-053d77cef82648fea9c69827182] Logs streamed to: local
 file. (LogDestination: /home/cloudshell-user/demoenv-logs/
queue-3ba4ff683ff54db09b851a2ed8b/session-053d77cef82648fea9c69827182.log)
 [queue-3ba4ff683ff54db09b851a2ed83/job-d34cc98a6e234b6f82577940ab4]
...
```

> **Note**
>
> 작업자 에이전트의 로깅 출력만 표시됩니다. 작업을 실행하는 세션에 대한 별도의 로그가 있습니다.

6. 첫 번째 탭을 선택한 다음 작업자 에이전트가 작성하는 로그 파일을 검사합니다.

   a. 작업자 에이전트 로그 디렉터리로 이동하여 내용을 확인합니다.

   ```
   cd ~/demoenv-logs
   ls
   ```

   b. 작업자 에이전트가 생성하는 첫 번째 로그 파일을 인쇄합니다.

   ```
   cat worker-agent-bootstrap.log
   ```

   이 파일에는 Deadline Cloud API를 호출하여 플릿에 작업자 리소스를 생성한 다음 플릿 역할을 수임한 방법에 대한 작업자 에이전트 출력이 포함되어 있습니다.

c. 작업자 에이전트가 플릿에 조인할 때 로그 파일 출력을 인쇄합니다.

```
cat worker-agent.log
```

이 로그에는 작업자 에이전트가 수행하는 모든 작업에 대한 출력이 포함되지만 해당 리소스의 IDs를 제외하고 작업을 실행하는 대기열에 대한 출력은 포함되지 않습니다.

d. 대기열 리소스 ID와 동일한 디렉터리의 각 세션에 대한 로그 파일을 인쇄합니다.

```
cat $DEV_QUEUE_ID/session-*.log
```

작업이 성공하면 로그 파일 출력은 다음과 유사합니다.

```
cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)

2024-03-27 16:00:22,026 WARNING Session running with no AWS Credentials.
2024-03-27 16:00:22,404 INFO
2024-03-27 16:00:22,405 INFO ================================================
2024-03-27 16:00:22,405 INFO --------- Running Task
2024-03-27 16:00:22,405 INFO ================================================
2024-03-27 16:00:22,406 INFO ------------------------------------------------
2024-03-27 16:00:22,406 INFO Phase: Setup
2024-03-27 16:00:22,406 INFO ------------------------------------------------
2024-03-27 16:00:22,406 INFO Writing embedded files for Task to disk.
2024-03-27 16:00:22,406 INFO Mapping: Task.File.runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_fileswa_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,406 INFO Wrote: runScript -> /sessions/
session-053d77cef82648fea9c698271812a/embedded_fileswa_gj55_/tmp2u9yqtsz
2024-03-27 16:00:22,407 INFO ------------------------------------------------
2024-03-27 16:00:22,407 INFO Phase: Running action
2024-03-27 16:00:22,407 INFO ------------------------------------------------
2024-03-27 16:00:22,407 INFO Running command /sessions/
session-053d77cef82648fea9c698271812a/tmpzuzxpslm.sh
2024-03-27 16:00:22,414 INFO Command started as pid: 471
2024-03-27 16:00:22,415 INFO Output:
2024-03-27 16:00:22,420 INFO Welcome to AWS Deadline Cloud!
2024-03-27 16:00:22,571 INFO
2024-03-27 16:00:22,572 INFO ================================================
2024-03-27 16:00:22,572 INFO --------- Session Cleanup
2024-03-27 16:00:22,572 INFO ================================================
```

```
2024-03-27 16:00:22,572 INFO Deleting working directory: /sessions/
session-053d77cef82648fea9c698271812a
```

7. 작업에 대한 정보를 인쇄합니다.

```
deadline job get
```

작업을 제출하면 시스템이 작업을 기본값으로 저장하므로 작업 ID를 입력할 필요가 없습니다.

파라미터와 simple_job 함께 제출

파라미터를 사용하여 작업을 제출할 수 있습니다. 다음 절차에서는 사용자 지정 메시지를 포함하도록 simple_job 템플릿을 편집하고 simple_job를 제출한 다음 세션 로그 파일을 인쇄하여 메시지를 확인합니다.

파라미터를 사용하여 simple_job 샘플을 제출하려면

1. 첫 번째 CloudShell 탭을 선택한 다음 작업 번들 샘플 디렉터리로 이동합니다.

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. simple_job 템플릿의 내용을 인쇄합니다.

```
cat simple_job/template.yaml
```

`Message` 파라미터가 있는 `parameterDefinitions` 섹션은 다음과 같아야 합니다.

```
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
```

3. 파라미터 값과 함께 simple_job 샘플을 제출한 다음 작업 실행이 완료될 때까지 기다립니다.

```
deadline bundle submit simple_job \
    -p "Message=Greetings from the developer getting started guide."
```

4. 사용자 지정 메시지를 보려면 최신 세션 로그 파일을 확인합니다.

```
cd ~/demoenv-logs
```

```
cat $DEV_QUEUE_ID/$(ls -t $DEV_QUEUE_ID | head -1)
```

파일 I/O를 사용하여 simple_file_job 작업 번들 생성

렌더링 작업은 장면 정의를 읽고 이미지를 렌더링한 다음 해당 이미지를 출력 파일에 저장해야 합니다. 이미지를 렌더링하는 대신 작업을 계산하여 입력의 해시를 생성하여이 작업을 시뮬레이션할 수 있습니다.

파일 I/O를 사용하여 simple_file_job 작업 번들을 생성하려면

1. 첫 번째 CloudShell 탭을 선택한 다음 작업 번들 샘플 디렉터리로 이동합니다.

   ```
   cd ~/deadline-cloud-samples/job_bundles/
   ```

2. 새 이름을 `simple_job` 사용하여의 복사본을 만듭니다`simple_file_job`.

   ```
   cp -r simple_job simple_file_job
   ```

3. 다음과 같이 작업 템플릿을 편집합니다.

   > **Note**
   >
   > 이러한 단계에는 nano를 사용하는 것이 좋습니다. 를 사용하려면를 사용하여 붙여넣기 모드를 설정해야 Vim합니다`:set paste`.

   a. 텍스트 편집기에서 템플릿을 엽니다.

      ```
      nano simple_file_job/template.yaml
      ```

   b. 다음 `type`, `objectType`및를 추가합니다`dataFlow`parameterDefinitions.

      ```
      - name: InFile
        type: PATH
        objectType: FILE
        dataFlow: IN
      - name: OutFile
        type: PATH
        objectType: FILE
      ```

```
  dataFlow: OUT
```

c. 다음 bash 스크립트 명령을 입력 파일에서 읽고 출력 파일에 쓰는 파일의 끝에 추가합니다.

```
        # hash the input file, and write that to the output
        sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"
```

업데이트된는 다음과 정확히 일치해야 `template.yaml` 합니다.

```
specificationVersion: 'jobtemplate-2023-09'
name: Simple File Job Bundle Example
parameterDefinitions:
- name: Message
  type: STRING
  default: Welcome to AWS Deadline Cloud!
- name: InFile
  type: PATH
  objectType: FILE
  dataFlow: IN
- name: OutFile
  type: PATH
  objectType: FILE
  dataFlow: OUT
steps:
- name: WelcomeToDeadlineCloud
  script:
    actions:
      onRun:
        command: '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      type: TEXT
      runnable: true
      data: |
        #!/usr/bin/env bash
        echo "{{Param.Message}}"

        # hash the input file, and write that to the output
        sha256sum "{{Param.InFile}}" > "{{Param.OutFile}}"
```

> **Note**
>
> 에서 간격을 조정하려면 들여쓰기 대신 공백을 사용해야 `template.yaml`합니다.

d. 파일을 저장하고 텍스트 편집기를 종료합니다.

4. 입력 및 출력 파일의 파라미터 값을 제공하여 simple_file_job을 제출합니다.

```
deadline bundle submit simple_file_job \
    -p "InFile=simple_job/template.yaml" \
    -p "OutFile=hash.txt"
```

5. 작업에 대한 정보를 인쇄합니다.

```
deadline job get
```

- 다음과 같은 출력이 표시됩니다.

```
parameters:
  Message:
    string: Welcome to AWS Deadline Cloud!
  InFile:
    path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/
template.yaml
  OutFile:
    path: /local/home/cloudshell-user/BundleFiles/JobBundle-Examples/hash.txt
```

- 상대 경로만 제공했지만 파라미터에는 전체 경로가 설정되어 있습니다. 는 경로의 유형이 일 때 파라미터로 제공되는 모든 경로에 현재 작업 디렉터리를 AWS CLI 조인합니다PATH.
- 다른 터미널 창에서 실행되는 작업자 에이전트가 작업을 픽업하고 실행합니다. 이 작업은 다음 명령을 사용하여 볼 수 있는 `hash.txt` 파일을 생성합니다.

```
cat hash.txt
```

이 명령은 다음과 유사한 출력을 인쇄합니다.

```
eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3 /local/home/
cloudshell-user/BundleFiles/JobBundle-Examples/simple_job/template.yaml
```

다음 단계

Deadline Cloud CLI를 사용하여 간단한 작업을 제출하는 방법을 학습한 후 다음을 탐색할 수 있습니다.

- Deadline Cloud에서 작업 첨부 파일이 있는 작업 제출 다른 운영 체제를 실행하는 호스트에서 작업을 실행하는 방법을 알아봅니다.
- Deadline Cloud의 개발자 팜에 서비스 관리형 플릿 추가 Deadline Cloud에서 관리하는 호스트에서 작업을 실행합니다.
- Deadline Cloud에서 팜 리소스 정리 -이 자습서에서 사용한 리소스를 종료합니다.

Deadline Cloud에서 작업 첨부 파일이 있는 작업 제출

많은 팜은 공유 파일 시스템을 사용하여 작업을 제출하는 호스트와 작업을 실행하는 호스트 간에 파일을 공유합니다. 예를 들어 이전 `simple_file_job` 예제에서 로컬 파일 시스템은 작업을 제출하는 탭 1과 작업자 에이전트를 실행하는 탭 2에서 실행되는 AWS CloudShell 터미널 창 간에 공유됩니다.

공유 파일 시스템은 제출자 워크스테이션과 작업자 호스트가 동일한 로컬 영역 네트워크에 있을 때 유용합니다. 데이터를 액세스하는 워크스테이션 근처의 온프레미스에 저장하는 경우 클라우드 기반 팜을 사용하면 지연 시간이 긴 VPN을 통해 파일 시스템을 공유하거나 클라우드에서 파일 시스템을 동기화해야 합니다. 이러한 옵션 중 어느 것도 설정하거나 작동하기가 쉽지 않습니다.

AWS Deadline Cloud는 이메일 첨부 파일과 유사한 작업 첨부 파일이 있는 간단한 솔루션을 제공합니다. 작업 첨부 파일을 사용하여 작업에 데이터를 연결합니다. 그런 다음 Deadline Cloud는 Amazon Simple Storage Service(Amazon S3) 버킷에 작업 데이터를 전송하고 저장하는 세부 정보를 처리합니다.

콘텐츠 생성 워크플로는 종종 반복적입니다. 즉, 사용자가 수정된 파일의 작은 하위 집합으로 작업을 제출합니다. Amazon S3 버킷은 콘텐츠 주소 지정 스토리지에 작업 첨부 파일을 저장하므로 각 객체의 이름은 객체 데이터의 해시를 기반으로 하며 디렉터리 트리의 콘텐츠는 작업에 연결된 매니페스트 파일 형식으로 저장됩니다.

이 섹션의 절차를 따르려면 먼저 다음을 완료해야 합니다.

- Deadline Cloud 팜 생성
- Deadline Cloud 작업자 에이전트 실행

작업 연결로 작업을 실행하려면 다음 단계를 완료하세요.

주제

대기열에 작업 첨부 파일 구성 추가

대기열에서 작업 연결을 활성화하려면 계정의 대기열 리소스에 작업 연결 구성을 추가합니다.

대기열에 작업 연결 구성을 추가하려면

1. 첫 번째 CloudShell 탭을 선택한 다음 다음 명령 중 하나를 입력하여 Amazon S3 버킷을 작업 연결에 사용합니다.
   - 기존 프라이빗 Amazon S3 버킷이 없는 경우 새 S3 버킷을 생성하고 사용할 수 있습니다.

     ```
     DEV_FARM_BUCKET=$(echo $DEV_FARM_NAME \
         | tr '[:upper:]' '[:lower:]')-$(xxd -l 16 -p /dev/urandom)
     if [ "$AWS_REGION" == "us-east-1" ]; then LOCATION_CONSTRAINT=
     else LOCATION_CONSTRAINT="--create-bucket-configuration \
         LocationConstraint=${AWS_REGION}"
     fi
     aws s3api create-bucket \
         $LOCATION_CONSTRAINT \
         --acl private \
         --bucket ${DEV_FARM_BUCKET}
     ```

   - 프라이빗 Amazon S3 버킷이 이미 있는 경우 *MY_BUCKET_NAME*를 버킷 이름으로 바꿔 사용할 수 있습니다.

     ```
     DEV_FARM_BUCKET=MY_BUCKET_NAME
     ```

2. Amazon S3 버킷을 생성하거나 선택한 후 버킷 이름을 `~/.bashrc`에 추가하여 다른 터미널 세션에서 버킷을 사용할 수 있도록 합니다.

   ```
   echo "DEV_FARM_BUCKET=$DEV_FARM_BUCKET" >> ~/.bashrc
    source ~/.bashrc
   ```

3. 대기열에 대한 AWS Identity and Access Management (IAM) 역할을 생성합니다.

```
aws iam create-role --role-name "${DEV_FARM_NAME}QueueRole" \
    --assume-role-policy-document \
        '{
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "credentials.deadline.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }'
aws iam put-role-policy \
    --role-name "${DEV_FARM_NAME}QueueRole" \
    --policy-name S3BucketsAccess \
    --policy-document \
            '{
                "Version": "2012-10-17",
                "Statement": [
                {
                    "Action": [
                        "s3:GetObject*",
                        "s3:GetBucket*",
                        "s3:List*",
                        "s3:DeleteObject*",
                        "s3:PutObject",
                        "s3:PutObjectLegalHold",
                        "s3:PutObjectRetention",
                        "s3:PutObjectTagging",
                        "s3:PutObjectVersionTagging",
                        "s3:Abort*"
                    ],
                    "Resource": [
                        "arn:aws:s3:::'$DEV_FARM_BUCKET'",
                        "arn:aws:s3:::'$DEV_FARM_BUCKET'/*"
                    ],
                    "Effect": "Allow"
                }
            ]
            }'
```

4. 작업 연결 설정과 IAM 역할을 포함하도록 대기열을 업데이트합니다.

```
QUEUE_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
        --query "Account" --output text):role/${DEV_FARM_NAME}QueueRole"
aws deadline update-queue \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --role-arn $QUEUE_ROLE_ARN \
    --job-attachment-settings \
        '{
            "s3BucketName": "'$DEV_FARM_BUCKET'",
            "rootPrefix": "JobAttachments"
        }'
```

5. 대기열을 업데이트했는지 확인합니다.

```
deadline queue get
```

다음과 같은 출력이 표시됩니다.

```
...
jobAttachmentSettings:
  s3BucketName: DEV_FARM_BUCKET
  rootPrefix: JobAttachments
roleArn: arn:aws:iam::ACCOUNT_NUMBER:role/DeveloperFarmQueueRole
...
```

작업 첨부 파일simple_file_job과 함께 제출

작업 첨부 파일을 사용하는 경우 작업 번들은 Deadline Cloud에 PATH 파라미터 사용과 같은 작업의 데이터 흐름을 결정할 수 있는 충분한 정보를 제공해야 합니다. 의 경우 Deadline Cloud에 데이터 흐름이 입력 `template.yaml` 파일 및 출력 파일에 있음을 알리도록 파일을 simple_file_job편집했습니다.

대기열에 작업 첨부 파일 구성을 추가한 후 작업 첨부 파일과 함께 simple_file_job 샘플을 제출할 수 있습니다. 이렇게 하면 로깅 및 작업 출력을 보고 작업 첨부 파일이 simple_file_job 있는이 작동하는지 확인할 수 있습니다.

작업 첨부 파일이 있는 simple_file_job 작업 번들을 제출하려면

1. 첫 번째 CloudShell 탭을 선택한 다음 `JobBundle-Samples` 디렉터리를 엽니다.

2.
```
cd ~/deadline-cloud-samples/job_bundles/
```

3. simple_file_job을 대기열에 제출합니다. 업로드를 확인하라는 메시지가 표시되면를 입력합니다**y**.

```
deadline bundle submit simple_file_job \
    -p InFile=simple_job/template.yaml \
    -p OutFile=hash-jobattachments.txt
```

4. 작업 첨부 파일 데이터 전송 세션 로그 출력을 보려면 다음 명령을 실행합니다.

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --job-id $JOB_ID \
        --query "sessions[0].sessionId" \
        --output text)
cat ~/demoenv-logs/$DEV_QUEUE_ID/$SESSION_ID.log
```

5. 세션 내에서 실행된 세션 작업을 나열합니다.

```
aws deadline list-session-actions \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --job-id $JOB_ID \
    --session-id $SESSION_ID
```

다음과 같은 출력이 표시됩니다.

```
{
    "sessionactions": [
        {
            "sessionActionId": "sessionaction-123-0",
            "status": "SUCCEEDED",
            "startedAt": "<timestamp>",
            "endedAt": "<timestamp>",
            "progressPercent": 100.0,
            "definition": {
                "syncInputJobAttachments": {}
            }
        },
        {
```

```
            "sessionActionId": "sessionaction-123-1",
            "status": "SUCCEEDED",
            "startedAt": "<timestamp>",
            "endedAt": "<timestamp>",
            "progressPercent": 100.0,
            "definition": {
                "taskRun": {
                    "taskId": "task-abc-0",
                    "stepId": "step-def"
                }
            }
        }
    ]
}
```

첫 번째 세션 작업은 입력 작업 첨부 파일을 다운로드한 반면, 두 번째 작업은 이전 단계에서와 같이 작업을 실행한 다음 출력 작업 첨부 파일을 업로드했습니다.

6. 출력 디렉터리를 나열합니다.

```
ls *.txt
```

와 같은 출력hash.txt은 디렉터리에 존재하지만 작업의 출력 파일이 아직 다운로드되지 않았기 때문에 hash-jobattachments.txt 존재하지 않습니다.

7. 최신 작업에서 출력을 다운로드합니다.

```
deadline job download-output
```

8. 다운로드한 파일의 출력을 봅니다.

```
cat hash-jobattachments.txt
```

다음과 같은 출력이 표시됩니다.

```
eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3  /tmp/openjd/
session-123/assetroot-abc/simple_job/template.yaml
```

작업 첨부 파일이 Amazon S3에 저장되는 방법 이해

AWS Command Line Interface (AWS CLI)를 사용하여 Amazon S3 버킷에 저장된 작업 첨부 파일에 대한 데이터를 업로드하거나 다운로드할 수 있습니다. Deadline Cloud가 Amazon S3에 작업 첨부 파일을 저장하는 방법을 이해하면 워크로드 및 파이프라인 통합을 개발할 때 도움이 됩니다.

Deadline Cloud 작업 첨부 파일이 Amazon S3에 저장되는 방법을 검사하려면

1. 첫 번째 CloudShell 탭을 선택한 다음 작업 번들 샘플 디렉터리를 엽니다.

```
cd ~/deadline-cloud-samples/job_bundles/
```

2. 작업 속성을 검사합니다.

```
deadline job get
```

다음과 같은 출력이 표시됩니다.

```
parameters:
  Message:
    string: Welcome to AWS Deadline Cloud!
  InFile:
    path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/simple_job/
template.yaml
  OutFile:
    path: /home/cloudshell-user/deadline-cloud-samples/job_bundles/hash-
jobattachments.txt
attachments:
  manifests:
  - rootPath: /home/cloudshell-user/deadline-cloud-samples/job_bundles/
    rootPathFormat: posix
    outputRelativeDirectories:
    - .
    inputManifestPath: farm-3040c59a5b9943d58052c29d907a645d/queue-
cde9977c9f4d4018a1d85f3e6c1a4e6e/Inputs/
f46af01ca8904cd8b514586671c79303/0d69cd94523ba617c731f29c019d16e8_input.xxh128
    inputManifestHash: f95ef91b5dab1fc1341b75637fe987ee
  fileSystem: COPIED
```

첨부 파일 필드에는 작업이 실행될 때 사용하는 입력 및 출력 데이터 경로를 설명하는 매니페스트 구조 목록이 포함되어 있습니다. 에서 작업을 제출한 시스템의 로컬 디렉터리 경로를

rootPath 확인합니다. 매니페스트 파일이 포함된 Amazon S3 객체 접미사를 보려면를 검토합니다inputManifestFile. 매니페스트 파일에는 작업 입력 데이터의 디렉터리 트리 스냅샷에 대한 메타데이터가 포함되어 있습니다.

3. Amazon S3 매니페스트 객체를 Pretty-print하여 작업의 입력 디렉터리 구조를 확인합니다.

```
MANIFEST_SUFFIX=$(aws deadline get-job \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID \
     --job-id $JOB_ID \
     --query "attachments.manifests[0].inputManifestPath" \
     --output text)
 aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Manifests/$MANIFEST_SUFFIX - | jq .
```

다음과 같은 출력이 표시됩니다.

```
{
     "hashAlg": "xxh128",
     "manifestVersion": "2023-03-03",
     "paths": [
     {
         "hash": "2ec297b04c59c4741ed97ac8fb83080c",
         "mtime": 1698186190000000,
         "path": "simple_job/template.yaml",
         "size": 445
     }
     ],
     "totalSize": 445
 }
```

4. 출력 작업 첨부 파일에 대한 매니페스트를 포함하는 Amazon S3 접두사를 구성하고 그 아래에 객체를 나열합니다.

```
SESSION_ACTION=$(aws deadline list-session-actions \
    --farm-id $DEV_FARM_ID \
    --queue-id $DEV_QUEUE_ID \
    --job-id $JOB_ID \
    --session-id $SESSION_ID \
    --query "sessionActions[?definition.taskRun != null] | [0]")
STEP_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.stepId)
TASK_ID=$(echo $SESSION_ACTION | jq -r .definition.taskRun.taskId)
```

```
TASK_OUTPUT_PREFIX=JobAttachments/Manifests/$DEV_FARM_ID/$DEV_QUEUE_ID/$JOB_ID/
$STEP_ID/$TASK_ID/
aws s3api list-objects-v2 --bucket $DEV_FARM_BUCKET --prefix $TASK_OUTPUT_PREFIX
```

출력 작업 연결은 작업 리소스에서 직접 참조되지 않고 대신 팜 리소스 IDs를 기반으로 Amazon S3 버킷에 배치됩니다.

5. 특정 세션 작업 ID에 대한 최신 매니페스트 객체 키를 가져온 다음 매니페스트 객체를 예쁘게 인쇄합니다.

```
SESSION_ACTION_ID=$(echo $SESSION_ACTION | jq -r .sessionActionId)
 MANIFEST_KEY=$(aws s3api list-objects-v2 \
     --bucket $DEV_FARM_BUCKET \
     --prefix $TASK_OUTPUT_PREFIX \
     --query "Contents[*].Key" --output text \
     | grep $SESSION_ACTION_ID \
     | sort | tail -1)
 MANIFEST_OBJECT=$(aws s3 cp s3://$DEV_FARM_BUCKET/$MANIFEST_KEY -)
 echo $MANIFEST_OBJECT | jq .
```

출력hash-jobattachments.txt에 다음과 같은 파일 속성이 표시됩니다.

```
{
     "hashAlg": "xxh128",
     "manifestVersion": "2023-03-03",
     "paths": [
     {
         "hash": "f60b8e7d0fabf7214ba0b6822e82e08b",
         "mtime": 1698785252554950,
         "path": "hash-jobattachments.txt",
         "size": 182
     }
     ],
     "totalSize": 182
 }
```

작업에는 작업 실행당 하나의 매니페스트 객체만 있지만 일반적으로 작업 실행당 더 많은 객체를 가질 수 있습니다.

6. Data 접두사 아래에서 콘텐츠 주소 지정 Amazon S3 스토리지 출력을 봅니다.

```
FILE_HASH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].hash)
FILE_PATH=$(echo $MANIFEST_OBJECT | jq -r .paths[0].path)
aws s3 cp s3://$DEV_FARM_BUCKET/JobAttachments/Data/$FILE_HASH -
```

다음과 같은 출력이 표시됩니다.

```
eaa2df5d34b54be5ac34c56a24a8c237b8487231a607eaf530a04d76b89c9cd3  /tmp/openjd/
session-123/assetroot-abc/simple_job/template.yaml
```

다음 단계

Deadline Cloud CLI를 사용하여 첨부 파일이 있는 작업을 제출하는 방법을 학습한 후 다음을 탐색할 수 있습니다.

- Deadline Cloud로 제출 작업자 호스트에서 OpenJD 번들을 사용하여 작업을 실행하는 방법을 알아봅니다.
- Deadline Cloud의 개발자 팜에 서비스 관리형 플릿 추가 Deadline Cloud에서 관리하는 호스트에서 작업을 실행합니다.
- Deadline Cloud에서 팜 리소스 정리 -이 자습서에서 사용한 리소스를 종료합니다.

Deadline Cloud의 개발자 팜에 서비스 관리형 플릿 추가

AWS CloudShell 는 더 큰 워크로드를 테스트하기에 충분한 컴퓨팅 용량을 제공하지 않습니다. 또한 여러 작업자 호스트에 작업을 배포하는 작업을 수행하도록 구성되지 않았습니다.

CloudShell을 사용하는 대신 개발자 팜에 Auto Scaling 서비스 관리형 플릿(SMF)을 추가할 수 있습니다. SMF는 대규모 워크로드에 충분한 컴퓨팅 용량을 제공하며 여러 작업자 호스트에 작업을 분산해야 하는 작업을 처리할 수 있습니다.

SMF를 추가하기 전에 Deadline Cloud 팜, 대기열 및 플릿을 설정해야 합니다. Deadline Cloud 팜 생성을(를) 참조하세요.

개발자 팜에 서비스 관리형 플릿을 추가하려면

1. 첫 번째 AWS CloudShell 탭을 선택한 다음 서비스 관리형 플릿을 생성하고 해당 플릿 ID를에 추가합니다`.bashrc`. 이 작업을 통해 다른 터미널 세션에서 사용할 수 있습니다.

```
FLEET_ROLE_ARN="arn:aws:iam::$(aws sts get-caller-identity \
        --query "Account" --output text):role/${DEV_FARM_NAME}FleetRole"
 aws deadline create-fleet \
     --farm-id $DEV_FARM_ID \
     --display-name "$DEV_FARM_NAME SMF" \
     --role-arn $FLEET_ROLE_ARN \
     --max-worker-count 5 \
     --configuration \
         '{
             "serviceManagedEc2": {
                 "instanceCapabilities": {
                     "vCpuCount": {
                         "min": 2,
                         "max": 4
                     },
                     "memoryMiB": {
                         "min": 512
                     },
                     "osFamily": "linux",
                     "cpuArchitectureType": "x86_64"
                 },
                 "instanceMarketOptions": {
                     "type": "spot"
                 }
             }
         }'

 echo "DEV_SMF_ID=$(aws deadline list-fleets \
         --farm-id $DEV_FARM_ID \
         --query "fleets[?displayName=='$DEV_FARM_NAME SMF'].fleetId \
         | [0]" --output text)" >> ~/.bashrc
 source ~/.bashrc
```

2. SMF를 대기열에 연결합니다.

```
aws deadline create-queue-fleet-association \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID \
     --fleet-id $DEV_SMF_ID
```

3. 대기열simple_file_job에 제출합니다. 업로드를 확인하라는 메시지가 표시되면를 입력합니다**y**.

```
deadline bundle submit simple_file_job \
    -p InFile=simple_job/template.yaml \
    -p OutFile=hash-jobattachments.txt
```

4. SMF가 올바르게 작동하는지 확인합니다.

```
deadline fleet get
```

- 작업자를 시작하는 데 몇 분 정도 걸릴 수 있습니다. 플릿이 실행 중임을 확인할 때까지 `deadline fleet get` 명령을 반복합니다.
- 서비스 관리형 플릿queueFleetAssociationsStatus의는 입니다ACTIVE.
- SMF는에서 GROWING로 autoScalingStatus 변경됩니다STEADY.

상태는 다음과 비슷합니다.

```
fleetId: fleet-2cc78e0dd3f04d1db427e7dc1d51ea44
farmId: farm-63ee8d77cdab4a578b685be8c5561c4a
displayName: DeveloperFarm SMF
description: ''
status: ACTIVE
autoScalingStatus: STEADY
targetWorkerCount: 0
workerCount: 0
minWorkerCount: 0
maxWorkerCount: 5
```

5. 제출한 작업에 대한 로그를 봅니다. 이 로그는 CloudShell 파일 시스템이 아닌 Amazon CloudWatch Logs의 로그에 저장됩니다.

```
JOB_ID=$(deadline config get defaults.job_id)
SESSION_ID=$(aws deadline list-sessions \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --job-id $JOB_ID \
        --query "sessions[0].sessionId" \
        --output text)
aws logs tail /aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID \
    --log-stream-names $SESSION_ID
```

다음 단계

서비스 관리형 플릿을 생성하고 테스트한 후에는 불필요한 요금이 발생하지 않도록 생성한 리소스를 제거해야 합니다.

- Deadline Cloud에서 팜 리소스 정리 -이 자습서에서 사용한 리소스를 종료합니다.

Deadline Cloud에서 팜 리소스 정리

새 워크로드 및 파이프라인 통합을 개발하고 테스트하기 위해이 자습서에서 생성한 Deadline Cloud 개발자 팜을 계속 사용할 수 있습니다. 개발자 팜이 더 이상 필요하지 않은 경우 Amazon CloudWatch Logs에서 팜, 플릿, 대기열, AWS Identity and Access Management (IAM) 역할 및 로그를 포함한 리소스를 삭제할 수 있습니다. 이러한 리소스를 삭제한 후 리소스를 사용하려면 자습서를 다시 시작해야 합니다. 자세한 내용은 Deadline Cloud 리소스 시작하기 단원을 참조하십시오.

개발자 팜 리소스를 정리하려면

1. 첫 번째 CloudShell 탭을 선택한 다음 대기열에 대한 모든 대기열-플릿 연결을 중지합니다.

```
FLEETS=$(aws deadline list-queue-fleet-associations \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --query "queueFleetAssociations[].fleetId" \
        --output text)
for FLEET_ID in $FLEETS; do
    aws deadline update-queue-fleet-association \
        --farm-id $DEV_FARM_ID \
        --queue-id $DEV_QUEUE_ID \
        --fleet-id $FLEET_ID \
        --status STOP_SCHEDULING_AND_CANCEL_TASKS
done
```

2. 대기열 플릿 연결을 나열합니다.

```
aws deadline list-queue-fleet-associations \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID
```

출력이를 보고할 때까지 명령을 다시 실행해야 할 수 있습니다. `"status": "STOPPED"`그런 다음 다음 단계로 진행할 수 있습니다. 이 프로세스는 완료하는 데 몇 분 정도 걸립니다.

```
{
    "queueFleetAssociations": [
        {
            "queueId": "queue-abcdefgh01234567890123456789012id",
            "fleetId": "fleet-abcdefgh01234567890123456789012id",
            "status": "STOPPED",
            "createdAt": "2023-11-21T20:49:19+00:00",
            "createdBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName",
            "updatedAt": "2023-11-21T20:49:38+00:00",
            "updatedBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName"
        },
        {
            "queueId": "queue-abcdefgh01234567890123456789012id",
            "fleetId": "fleet-abcdefgh01234567890123456789012id",
            "status": "STOPPED",
            "createdAt": "2023-11-21T20:32:06+00:00",
            "createdBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName",
            "updatedAt": "2023-11-21T20:49:39+00:00",
            "updatedBy": "arn:aws:sts::123456789012:assumed-role/RoleToBeAssumed/
MySessionName"
        }
    ]
}
```

3. 대기열에 대한 모든 대기열-플릿 연결을 삭제합니다.

```
for FLEET_ID in $FLEETS; do
     aws deadline delete-queue-fleet-association \
         --farm-id $DEV_FARM_ID \
         --queue-id $DEV_QUEUE_ID \
         --fleet-id $FLEET_ID
 done
```

4. 대기열과 연결된 모든 플릿을 삭제합니다.

```
for FLEET_ID in $FLEETS; do
     aws deadline delete-fleet \
         --farm-id $DEV_FARM_ID \
         --fleet-id $FLEET_ID
```

```
done
```

5. 대기열을 삭제합니다.

```
aws deadline delete-queue \
     --farm-id $DEV_FARM_ID \
     --queue-id $DEV_QUEUE_ID
```

6. 팜을 삭제합니다.

```
aws deadline delete-farm \
     --farm-id $DEV_FARM_ID
```

7. 팜의 다른 AWS 리소스를 삭제합니다.

   a. 플릿 AWS Identity and Access Management (IAM) 역할을 삭제합니다.

```
aws iam delete-role-policy \
     --role-name "${DEV_FARM_NAME}FleetRole" \
     --policy-name WorkerPermissions
aws iam delete-role \
     --role-name "${DEV_FARM_NAME}FleetRole"
```

   b. 대기열 IAM 역할을 삭제합니다.

```
aws iam delete-role-policy \
     --role-name "${DEV_FARM_NAME}QueueRole" \
     --policy-name S3BucketsAccess
aws iam delete-role \
     --role-name "${DEV_FARM_NAME}QueueRole"
```

   c. Amazon CloudWatch Logs 로그 그룹을 삭제합니다. 각 대기열과 플릿에는 고유한 로그 그룹이 있습니다.

```
aws logs delete-log-group \
     --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_QUEUE_ID"
aws logs delete-log-group \
     --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_CMF_ID"
aws logs delete-log-group \
     --log-group-name "/aws/deadline/$DEV_FARM_ID/$DEV_SMF_ID"
```

Deadline Cloud에 제출할 작업 빌드

작업 번들을 사용하여 Deadline Cloud에 작업을 제출합니다. 작업 번들은 Open Job Description(OpenJD) 작업 템플릿과 작업을 렌더링하는 데 필요한 모든 자산 파일을 포함한 파일 모음입니다.

작업 템플릿은 작업자가 자산을 처리하고 액세스하는 방법을 설명하고 작업자가 실행하는 스크립트를 제공합니다. 작업 번들을 사용하면 아티스트, 기술 책임자 및 파이프라인 개발자가 로컬 워크스테이션 또는 온프레미스 렌더 팜에서 Deadline Cloud에 복잡한 작업을 쉽게 제출할 수 있습니다. 작업 번들은 확장 가능한 온디맨드 컴퓨팅 리소스가 필요한 대규모 시각 효과, 애니메이션 또는 기타 미디어 렌더링 프로젝트에서 작업하는 팀에 특히 유용합니다.

로컬 파일 시스템을 사용하여 작업 번들을 생성하여 파일을 저장하고 텍스트 편집기를 사용하여 작업 템플릿을 생성할 수 있습니다. 번들을 생성한 후 Deadline Cloud CLI 또는 Deadline Cloud 제출자와 같은 도구를 사용하여 Deadline Cloud에 작업을 제출합니다.

작업자 간에 공유된 파일 시스템에 자산을 저장하거나 Deadline Cloud 작업 연결을 사용하여 작업자가 액세스할 수 있는 S3 버킷으로 자산 이동을 자동화할 수 있습니다. 또한 작업 연결은 작업의 출력을 워크스테이션으로 다시 이동하는 데 도움이 됩니다.

다음 섹션에서는 Deadline Cloud에 작업 번들을 생성하고 제출하는 방법에 대한 자세한 지침을 제공합니다.

주제

Deadline Cloud에 대한 Open Job Description(OpenJD) 템플릿

작업 번들은 AWS Deadline Cloud에 대한 작업을 정의하는 데 사용하는 도구 중 하나입니다. 작업 첨부 파일에 사용하는 파일 및 디렉터리와 같은 추가 정보와 함께 Open Job Description(OpenJD) 템플릿을

그룹화합니다. Deadline Cloud 명령줄 인터페이스(CLI)를 사용하여 작업 번들을 사용하여 실행할 대기열에 대한 작업을 제출합니다.

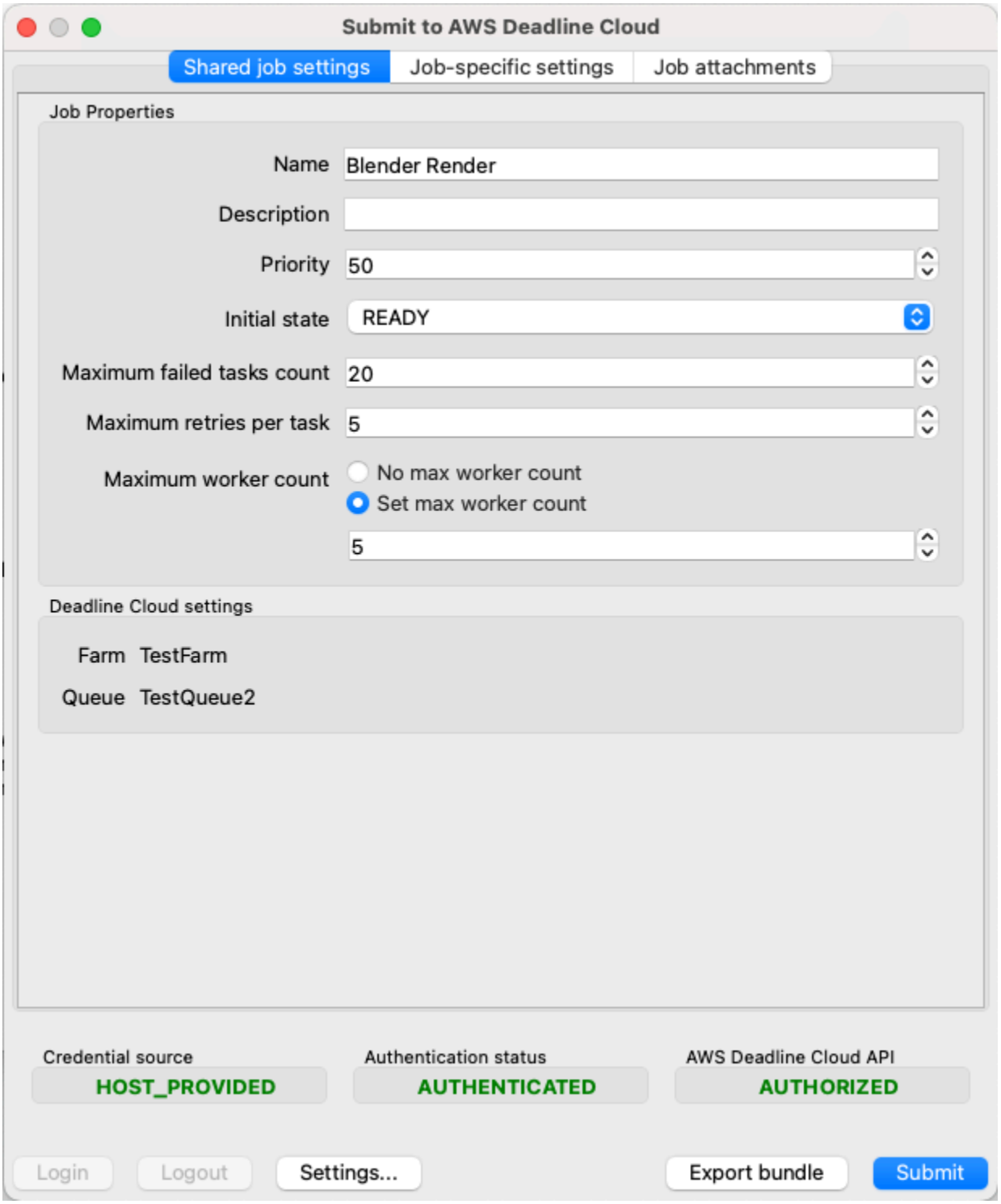
작업 번들은 OpenJD 작업 템플릿, 작업을 정의하는 기타 파일, 작업에 대한 입력으로 필요한 작업별 파일을 포함하는 디렉터리 구조입니다. 작업을 정의하는 파일을 YAML 또는 JSON 파일로 지정할 수 있습니다.

유일한 필수 파일은 `template.yaml` 또는 입니다`template.json`. 다음 파일을 포함할 수도 있습니다.

```
/template.yaml (or template.json)
/asset_references.yaml (or asset_references.json)
/parameter_values.yaml (or parameter_values.json)
/other job-specific files and directories
```

Deadline Cloud CLI 및 작업 연결을 사용하여 사용자 지정 작업 제출에 작업 번들을 사용하거나 그래픽 제출 인터페이스를 사용할 수 있습니다. 예를 들어 GitHub의 Blender 샘플은 다음과 같습니다. Blender 샘플 디렉터리에서 다음 명령을 사용하여 샘플을 실행하려면

```
deadline bundle gui-submit blender_render
```



작업별 설정 패널은 작업 템플릿에 정의된 작업 파라미터의 `userInterface` 속성에서 생성됩니다.

명령줄을 사용하여 작업을 제출하려면 다음과 유사한 명령을 사용할 수 있습니다.

```
deadline bundle submit \
```

```
    --yes \
    --name Demo \
     -p BlenderSceneFile=location of scene file \
     -p OutputDir=file pathe for job output \
      blender_render/
```

또는 deadline Python 패키지에서 deadline.client.api.create_job_from_job_bundle 함수를 사용할 수 있습니다.

Autodesk Maya 플러그인과 같이 Deadline Cloud와 함께 제공되는 모든 작업 제출자 플러그인은 제출을 위한 작업 번들을 생성한 다음 Deadline Cloud Python 패키지를 사용하여 Deadline Cloud에 작업을 제출합니다. 워크스테이션의 작업 기록 디렉터리에서 또는 제출자를 사용하여 제출된 작업 번들을 볼 수 있습니다. 다음 명령을 사용하여 작업 기록 디렉터리를 찾을 수 있습니다.

```
deadline config get settings.job_history_dir
```

작업이 Deadline Cloud 작업자에서 실행 중인 경우 작업에 대한 정보를 제공하는 환경 변수에 액세스할 수 있습니다. 환경 변수는 다음과 같습니다.

변수 이름	Available
DEADLINE_FARM_ID	모든 작업
DEADLINE_FLEET_ID	모든 작업
DEADLINE_WORKER_ID	모든 작업
DEADLINE_QUEUE_ID	모든 작업
DEADLINE_JOB_ID	모든 작업
DEADLINE_STEP_ID	작업 작업
DEADLINE_SESSION_ID	모든 작업
DEADLINE_TASK_ID	작업 작업
DEADLINE_SESSIONACTION_ID	모든 작업

주제

작업 번들에 대한 작업 템플릿 요소

작업 템플릿은 런타임 환경과 Deadline Cloud 작업의 일부로 실행되는 프로세스를 정의합니다. 템플릿에서 파라미터를 생성하여 프로그래밍 언어의 함수와 마찬가지로 입력 값만 다른 작업을 생성하는 데 사용할 수 있습니다.

Deadline Cloud에 작업을 제출하면 대기열에 적용된 모든 대기열 환경에서 실행됩니다. 대기열 환경은 Open Job Description(OpenJD) 외부 환경 사양을 사용하여 빌드됩니다. 자세한 내용은 OpenJD GitHub 리포지토리의 환경 템플릿을 참조하세요.

OpenJD 작업 템플릿을 사용하여 작업을 생성하는 방법에 대한 소개는 OpenJD GitHub 리포지토리에서 작업 생성 소개를 참조하세요. 추가 정보는 작업 실행 방법에서 확인할 수 있습니다. OpenJD GitHub 리포지토리의 `samples` 디렉터리에 있는에 작업 템플릿 샘플이 있습니다.

작업 템플릿을 YAML 형식(`template.yaml`) 또는 JSON 형식()으로 정의할 수 있습니다`template.json`. 이 섹션의 예제는 YAML 형식으로 표시됩니다.

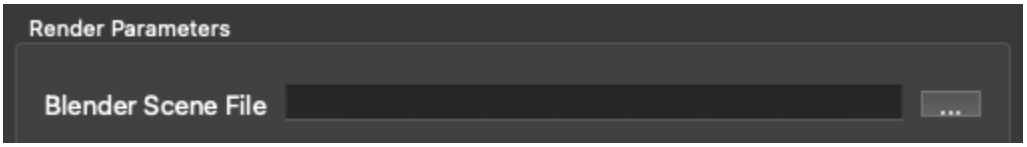
예를 들어 `blender_render` 샘플의 작업 템플릿은 입력 파라미터를 파일 경로`BlenderSceneFile`로 정의합니다.

```
- name: BlenderSceneFile
  type: PATH
  objectType: FILE
  dataFlow: IN
  userInterface:
    control: CHOOSE_INPUT_FILE
    label: Blender Scene File
    groupLabel: Render Parameters
    fileFilters:
    - label: Blender Scene Files
      patterns: ["*.blend"]
    - label: All Files
      patterns: ["*"]
  description: >
    Choose the Blender scene file to render. Use the 'Job Attachments' tab
```

```
    to add textures and other files that the job needs.
```

userInterface 속성은 명령을 사용하고 Autodesk Maya와 같은 애플리케이션의 작업 제출 플러그인 내에서 deadline bundle gui-submit 명령줄 모두에 대해 자동으로 생성된 사용자 인터페이스의 동작을 정의합니다.

이 예제에서 BlenderSceneFile 파라미터 값을 입력하기 위한 UI 위젯은 파일만 표시하는 .blend 파일 선택 대화 상자입니다.



userInteface 요소 사용에 대한 자세한 예는 GitHub의 deadline-cloud-samples 리포지토리에서 gui_control_showcase 샘플을 참조하세요. deadline-cloud-samples

objectType 및 dataFlow 속성은 작업 번들에서 작업을 제출할 때 작업 첨부 파일의 동작을 제어합니다. 이 경우 objectType: FILE 및 BlenderSceneFile는의 값이 작업 첨부 파일의 입력 파일임을 dataFlow:IN 의미합니다.

반대로 OutputDir 파라미터의 정의에는 objectType: DIRECTORY 및가 dataFlow: OUT있습니다.

```
- name: OutputDir
  type: PATH
  objectType: DIRECTORY
  dataFlow: OUT
  userInterface:
    control: CHOOSE_DIRECTORY
    label: Output Directory
    groupLabel: Render Parameters
  default: "./output"
  description: Choose the render output directory.
```

OutputDir 파라미터 값은 작업 첨부 파일에서 작업이 출력 파일을 쓰는 디렉터리로 사용됩니다.

objectType 및 dataFlow 속성에 대한 자세한 내용은 Open Job Description 사양의 JobPathParameterDefinition을 참조하세요.

나머지 blender_render 작업 템플릿 샘플은 작업의 워크플로를 애니메이션의 각 프레임이 별도의 작업으로 렌더링되는 단일 단계로 정의합니다.

```
steps:
- name: RenderBlender
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      type: INT
      range: "{{Param.Frames}}"
  script:
    actions:
      onRun:
        command: bash
        # Note: {{Task.File.Run}} is a variable that expands to the filename on the
 worker host's
        # disk where the contents of the 'Run' embedded file, below, is written.
        args: ['{{Task.File.Run}}']
    embeddedFiles:
      - name: Run
        type: TEXT
        data: |
          # Configure the task to fail if any individual command fails.
          set -xeuo pipefail

          mkdir -p '{{Param.OutputDir}}'

          blender --background '{{Param.BlenderSceneFile}}' \
                  --render-output '{{Param.OutputDir}}/{{Param.OutputPattern}}' \
                  --render-format {{Param.Format}} \
                  --use-extension 1 \
                  --render-frame {{Task.Param.Frame}}
```

예를 들어 Frames 파라미터 값이 인 경우 1-1010개의 작업을 정의합니다. 각 에는 Frame 파라미터 값이 다릅니다. 작업을 실행하려면:

1. 와 같이 임베디드 파일의 data 속성에 있는 모든 변수 참조가 확장됩니다--render-frame 1.
2. data 속성의 내용은 디스크의 세션 작업 디렉터리에 있는 파일에 기록됩니다.
3. 작업의 onRun 명령이 로 확인bash *location of embedded file*되고 실행됩니다.

임베디드 파일, 세션 및 경로 매핑 위치에 대한 자세한 내용은 Open Job Description 사양의 작업 실행 방법을 참조하세요.

deadline-cloud-samples/job_bundles 리포지토리에는 작업 템플릿의 더 많은 예와 Open Job Descriptions 사양과 함께 제공되는 템플릿 샘플이 있습니다.

작업 템플릿에 대한 작업 청킹

작업 청킹을 사용하면 여러 작업을 청크라는 단일 작업 단위로 그룹화할 수 있습니다. 예를 들어 렌더링 작업에서 Deadline Cloud는 명령 호출당 하나의 프레임 대신 여러 프레임을 함께 디스패치할 수 있습니다. 이렇게 하면 각 작업에 대한 애플리케이션 시작 오버헤드가 줄어들고 총 작업 런타임이 단축됩니다. 자세한 내용은 OpenJD Wiki에서 한 번에 여러 프레임 실행을 참조하세요.

OpenJD는 작업 템플릿에 선택적 기능을 추가하는 확장을 지원합니다. 작업 청킹은 `TASK_CHUNKING` 확장을 추가하여 활성화됩니다. 청킹을 사용하려면 작업 템플릿에 확장을 추가하고 `CHUNK[INT]` 작업 파라미터 유형을 사용합니다. 동일한 `deadline bundle submit` 명령을 사용하여 청크 작업을 제출합니다. 예를 들어 다음 작업 템플릿은 프레임을 10 청크로 렌더링합니다.

```
specificationVersion: 'jobtemplate-2023-09'
extensions:
  - TASK_CHUNKING
name: Blender Render with Contiguous Chunking
parameterDefinitions:
  - name: BlenderSceneFile
    type: PATH
    objectType: FILE
    dataFlow: IN
  - name: Frames
    type: STRING
    default: "1-100"
  - name: OutputDir
    type: PATH
    objectType: DIRECTORY
    dataFlow: OUT
    default: "./output"
steps:
  - name: RenderBlender
    parameterSpace:
      taskParameterDefinitions:
        - name: Frame
          type: CHUNK[INT]
          range: "{{Param.Frames}}"
          chunks:
            defaultTaskCount: 10
            rangeConstraint: CONTIGUOUS
```

```
    script:
      actions:
        onRun:
          command: bash
          args: ["{{Task.File.Run}}"]
      embeddedFiles:
        - name: Run
          type: TEXT
          data: |
            set -xeuo pipefail

            mkdir -p '{{Param.OutputDir}}'

            # Parse the chunk range (e.g., "1-10") into start and end frames
            START_FRAME="$(echo '{{Task.Param.Frame}}' | cut -d- -f1)"
            END_FRAME="$(echo '{{Task.Param.Frame}}' | cut -d- -f2)"

            blender --background '{{Param.BlenderSceneFile}}' \
                    --render-output '{{Param.OutputDir}}/output_####' \
                    --render-format PNG \
                    --use-extension 1 \
                    -s "$START_FRAME" \
                    -e "$END_FRAME" \
                    --render-anim
```

이 예제에서 Deadline Cloud는 100프레임을 `1-10`, `11-20`등과 같은 청크로 나눕니다. `{{Task.Param.Frame}}` 변수는와 같은 범위 표현식으로 확장됩니다`1-10`. `rangeConstraint`는로 설정되므로 `CONTIGUOUS`범위는 항상 `start-end` 형식입니다. 스크립트는이 범위를 구문 분석하고와 함께 `-s` 및 `-e` 옵션을 사용하여 시작 및 종료 프레임을 Blender에 전달합니다`--render-anim`.

`chunks` 속성은 다음 필드를 지원합니다.

- `defaultTaskCount` – (필수) 단일 청크로 결합할 작업 수입니다. 최대값은 150입니다.
- `rangeConstraint` – (필수) `CONTIGUOUS`인 경우 청크는 항상와 같은 연속 범위입니다`1-10`. `NONCONTIGUOUS`인 경우 청크는와 같은 임의의 집합일 수 있습니다`1,3,7-10`.
- `targetRuntimeSeconds` – (선택 사항) 각 청크에 대한 초 단위의 대상 런타임입니다. Deadline Cloud는 일부 청크가 완료되면이 대상에 접근하도록 청크 크기를 동적으로 조정할 수 있습니다.

연속 청크와 비연속 청크가 모두 있는 기본 및 Blender 예제를 포함한 추가 태스크 청킹 예제는 GitHub의 Deadline Cloud 샘플 리포지토리에서 태스크 청킹 샘플을 참조하세요.

> **고객 관리형 플릿 요구 사항**
>
> 작업 청킹에는 호환되는 작업자 에이전트 버전이 필요합니다. 고객 관리형 플릿을 사용하는 경우 청킹으로 작업을 제출하기 전에 작업자 에이전트가 업데이트되었는지 확인합니다. 서비스 관리형 플릿은 항상 호환되는 작업자 에이전트 버전을 사용합니다.

> **청크 작업에 대한 출력 다운로드**
>
> 청크 작업에서 단일 작업에 대한 출력을 다운로드하면 Deadline Cloud는 전체 청크에 대한 출력을 다운로드합니다. 예를 들어 프레임 1~10이 함께 처리된 경우 프레임 3에 대한 출력을 다운로드하면 모든 프레임 1~10이 포함됩니다. 이 기능을 사용하려면 `deadline-cloud` 버전 0.53.3 이상이 필요합니다.

작업 번들의 파라미터 값 요소

작업 제출 시 값을 설정할 필요가 없도록 파라미터 파일을 사용하여 작업 템플릿의 일부 작업 파라미터 또는 작업 번들의 CreateJob 작업 요청 인수의 값을 설정할 수 있습니다. 작업 제출을 위한 UI를 사용하면 이러한 값을 수정할 수 있습니다.

작업 템플릿을 YAML 형식(`parameter_values.yaml`) 또는 JSON 형식()으로 정의할 수 있습니다`parameter_values.json`. 이 섹션의 예제는 YAML 형식으로 표시됩니다.

YAML에서 파일의 형식은 다음과 같습니다.

```
parameterValues:
- name: <string>
  value: <integer>, <float>, or <string>
- name: <string>
  value: <integer>, <float>, or <string>ab
... repeating as necessary
```

`parameterValues` 목록의 각 요소는 다음 중 하나여야 합니다.

- 작업 템플릿에 정의된 작업 파라미터입니다.
- 작업을 제출하는 대기열의 대기열 환경에 정의된 작업 파라미터입니다.
- 작업을 생성할 때 `CreateJob` 작업에 전달되는 특수 파라미터입니다.

- `deadline:priority` - 값은 정수여야 합니다. 우선 순위 파라미터로 `CreateJob` 작업에 전달됩니다.
- `deadline:targetTaskRunStatus` - 값은 문자열이어야 합니다. 작업에 targetTaskRunStatus 파라미터`CreateJob`로 전달됩니다.
- `deadline:maxFailedTasksCount` - 값은 정수여야 합니다. maxFailedTasksCount 파라미터로 `CreateJob` 작업에 전달됩니다.
- `deadline:maxRetriesPerTask` - 값은 정수여야 합니다. maxRetriesPerTask 파라미터로 `CreateJob` 작업에 전달됩니다.
- `deadline:maxWorkercount` - 값은 정수여야 합니다. maxWorkerCount 파라미터로 `CreateJob` 작업에 전달됩니다.

작업 템플릿은 실행할 특정 작업이 아닌 항상 템플릿입니다. 파라미터 값 파일을 사용하면 일부 파라미터에이 파일에 정의된 값이 없는 경우 작업 번들이 템플릿 역할을 하거나 모든 파라미터에 값이 있는 경우 특정 작업 제출 역할을 할 수 있습니다.

예를 들어 blender_render 샘플에는 파라미터 파일이 없으며 해당 작업 템플릿은 기본값이 없는 파라미터를 정의합니다. 이 템플릿은 작업을 생성하기 위한 템플릿으로 사용해야 합니다. 이 작업 번들을 사용하여 작업을 생성한 후 Deadline Cloud는 작업 기록 디렉터리에 새 작업 번들을 작성합니다.

예를 들어 다음 명령을 사용하여 작업을 제출하는 경우:

```
deadline bundle gui-submit blender_render/
```

새 작업 번들에는 지정된 파라미터가 포함된 `parameter_values.yaml` 파일이 포함되어 있습니다.

```
% cat ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-JobBundle-Demo/
parameter_values.yaml
parameterValues:
- name: deadline:targetTaskRunStatus
  value: READY
- name: deadline:maxFailedTasksCount
  value: 10
- name: deadline:maxRetriesPerTask
  value: 5
- name: deadline:priority
  value: 75
- name: BlenderSceneFile
  value: /private/tmp/bundle_demo/bmw27_cpu.blend
```

```
- name: Frames
  value: 1-10
- name: OutputDir
  value: /private/tmp/bundle_demo/output
- name: OutputPattern
  value: output_####
- name: Format
  value: PNG
- name: CondaPackages
  value: blender
- name: RezPackages
  value: blender
```

다음 명령을 사용하여 동일한 작업을 생성할 수 있습니다.

```
deadline bundle submit ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-
JobBundle-Demo/
```

> **Note**
>
> 제출하는 작업 번들은 작업 기록 디렉터리에 저장됩니다. 다음 명령을 사용하여 해당 디렉터리의 위치를 찾을 수 있습니다.
>
> ```
> deadline config get settings.job_history_dir
> ```

작업 번들에 대한 자산 참조 요소

Deadline Cloud 작업 연결을 사용하여 워크스테이션과 Deadline Cloud 간에 파일을 주고받을 수 있습니다. 자산 참조 파일에는 입력 파일 및 디렉터리와 첨부 파일의 출력 디렉터리가 나열됩니다. 이 파일의 모든 파일과 디렉터리를 나열하지 않으면 `deadline bundle gui-submit` 명령을 사용하여 작업을 제출할 때 선택할 수 있습니다.

작업 첨부 파일을 사용하지 않는 경우이 파일은 영향을 주지 않습니다.

작업 템플릿을 YAML 형식(`asset_references.yaml`) 또는 JSON 형식()으로 정의할 수 있습니다`asset_references.json`. 이 섹션의 예제는 YAML 형식으로 표시됩니다.

YAML에서 파일의 형식은 다음과 같습니다.

```
assetReferences:
    inputs:
        # Filenames on the submitting workstation whose file contents are needed as
        # inputs to run the job.
        filenames:
        - list of file paths
        # Directories on the submitting workstation whose contents are needed as inputs
        # to run the job.
        directories:
        - list of directory paths

    outputs:
        # Directories on the submitting workstation where the job writes output files
        # if running locally.
        directories:
        - list of directory paths

    # Paths referenced by the job, but not necessarily input or output.
    # Use this if your job uses the name of a path in some way, but does not explicitly
 need
    # the contents of that path.
    referencedPaths:
    - list of directory paths
```

Amazon S3에 업로드할 입력 또는 출력 파일을 선택할 때 Deadline Cloud는 파일 경로를 스토리지 프로파일에 나열된 경로와 비교합니다. 스토리지 프로파일의 각 SHARED유형 파일 시스템 위치는 워크스테이션 및 작업자 호스트에 탑재된 네트워크 파일 공유를 추상화합니다. Deadline Cloud는 이러한 파일 공유 중 하나에 없는 파일만 업로드합니다.

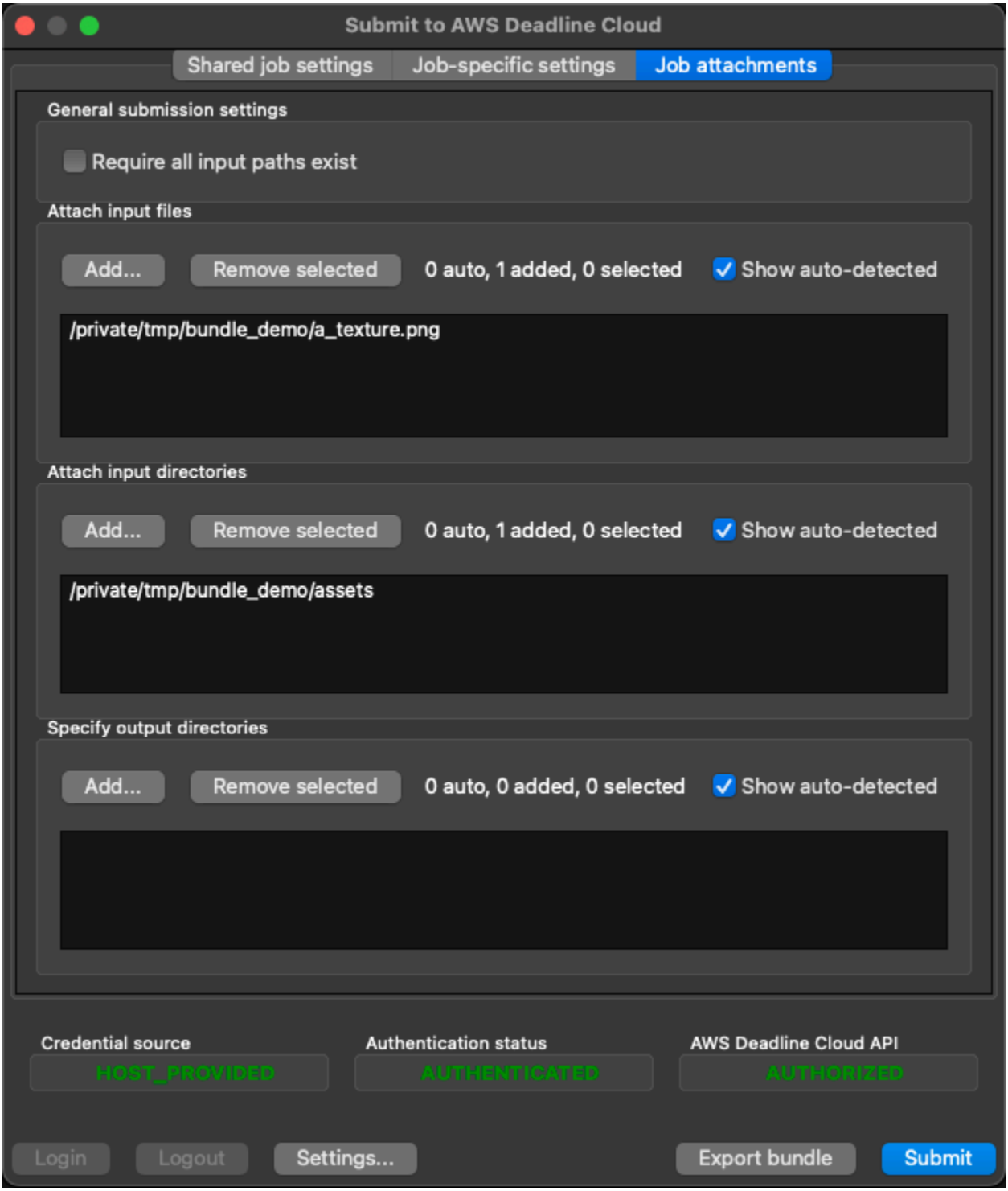
스토리지 프로파일 생성 및 사용에 대한 자세한 내용은 Deadline Cloud 사용 설명서의 Deadline Cloud의 공유 스토리지를 참조하세요. AWS

Example- Deadline Cloud GUI에서 생성한 자산 참조 파일

다음 명령을 사용하여 blender_render 샘플을 사용하여 작업을 제출합니다.

```
deadline bundle gui-submit blender_render/
```

작업 첨부 파일 탭에서 작업에 몇 가지 파일을 추가합니다.



작업을 제출한 후 작업 기록 디렉터리의 작업 번들에서 `asset_references.yaml` 파일을 보고 YAML 파일의 자산을 볼 수 있습니다.

```
% cat ~/.deadline/job_history/\(default\)/2024-06/2024-06-20-01-JobBundle-Demo/
asset_references.yaml
```

```
assetReferences:
  inputs:
    filenames:
    - /private/tmp/bundle_demo/a_texture.png
    directories:
    - /private/tmp/bundle_demo/assets
  outputs:
    directories: []
  referencedPaths: []
```

작업에서 파일 사용

AWS Deadline Cloud에 제출하는 많은 작업에는 입력 및 출력 파일이 있습니다. 입력 파일 및 출력 디렉터리는 공유 파일 시스템과 로컬 드라이브의 조합에 있을 수 있습니다. 작업은 해당 위치에서 콘텐츠를 찾아야 합니다. Deadline Cloud는 작업이 필요한 파일을 찾는 데 도움이 되도록 작업 연결과 스토리지 프로파일이라는 두 가지 기능을 제공합니다.

작업 첨부 파일은 몇 가지 이점을 제공합니다.

- Amazon S3를 사용하여 호스트 간에 파일 이동
- 워크스테이션에서 작업자 호스트로 또는 그 반대로 파일 전송
- 기능을 활성화하는 대기열의 작업에 사용 가능
- 주로 서비스 관리형 플릿과 함께 사용되지만 고객 관리형 플릿과도 호환됩니다.

스토리지 프로파일을 사용하여 워크스테이션 및 작업자 호스트에서 공유 파일 시스템 위치의 레이아웃을 매핑합니다. 이 매핑은 기반 워크스테이션 및 Windows기반 작업자 호스트를 사용한 교차 플랫폼 설정과 같이 워크스테이션과 Linux작업자 호스트의 위치가 다를 때 작업이 공유 파일 및 디렉터리를 찾는 데 도움이 됩니다. 파일 시스템 구성의 스토리지 프로파일 맵은 작업 첨부 파일에서도 Amazon S3를 통해 호스트 간에 통신하는 데 필요한 파일을 식별하는 데 사용됩니다.

작업 첨부 파일을 사용하지 않고 워크스테이션과 작업자 호스트 간에 파일 및 디렉터리 위치를 다시 매핑할 필요가 없는 경우 스토리지 프로파일로 파일 공유를 모델링할 필요가 없습니다.

주제

- 샘플 프로젝트 인프라
- 스토리지 프로파일 및 경로 매핑

샘플 프로젝트 인프라

작업 연결 및 스토리지 프로파일 사용을 시연하려면 두 개의 별도 프로젝트로 테스트 환경을 설정합니다. Deadline Cloud 콘솔을 사용하여 테스트 리소스를 생성할 수 있습니다.

1. 아직 생성하지 않았다면 테스트 팜을 생성합니다. 팜을 생성하려면 팜 생성의 절차를 따릅니다.
2. 두 프로젝트 각각에서 작업에 대해 두 개의 대기열을 생성합니다. 대기열을 생성하려면 대기열 생성의 절차를 따릅니다.

   a. 라는 첫 번째 대기열을 생성합니다**Q1**. 다음 구성을 사용하고 다른 모든 항목에 기본값을 사용합니다.

      - 작업 연결에서 새 Amazon S3 버킷 생성을 선택합니다.
      - 고객 관리형 플릿과의 연결 활성화를 선택합니다.
      - 사용자로 실행의 경우 POSIX 사용자와 그룹 모두에 **jobuser**를 입력합니다.
      - 대기열 서비스 역할의 경우 라는 새 역할을 생성합니다. **AssetDemoFarm-Q1-Role**
      - 기본 conda 대기열 환경 확인란의 선택을 취소합니다.

   b. 라는 두 번째 대기열을 생성합니다**Q2**. 다음 구성을 사용하고 다른 모든 항목에 기본값을 사용합니다.

      - 작업 연결에서 새 Amazon S3 버킷 생성을 선택합니다.
      - 고객 관리형 플릿과의 연결 활성화를 선택합니다.
      - 사용자로 실행의 경우 POSIX 사용자와 그룹 모두에 **jobuser**를 입력합니다.
      - 대기열 서비스 역할의 경우 라는 새 역할을 생성합니다. **AssetDemoFarm-Q2-Role**
      - 기본 conda 대기열 환경 확인란의 선택을 취소합니다.

3. 두 대기열에서 작업을 실행하는 단일 고객 관리형 플릿을 생성합니다. 플릿을 생성하려면 고객 관리형 플릿 생성의 절차를 따릅니다. 다음 구성을 사용합니다.

   - 이름에는를 사용합니다**DemoFleet**.
   - 플릿 유형에서 고객 관리형을 선택합니다.
   - 플릿 서비스 역할의 경우 AssetDemoFarm-Fleet-Role이라는 새 역할을 생성합니다.
   - 플릿을 대기열과 연결하지 마세요.

테스트 환경에서는 네트워크 파일 공유를 사용하여 호스트 간에 공유되는 파일 시스템이 3개 있다고 가정합니다. 이 예제에서 위치의 이름은 다음과 같습니다.

- FSCommon - 두 프로젝트 모두에 공통적인 입력 작업 자산을 포함합니다.
- FS1 - 프로젝트 1에 대한 입력 및 출력 작업 자산을 포함합니다.
- FS2 - 프로젝트 2에 대한 입력 및 출력 작업 자산을 포함합니다.

또한 테스트 환경은 다음과 같이 3개의 워크스테이션이 있다고 가정합니다.

- WSAll - 개발자가 모든 프로젝트에 사용하는 Linux기반 워크스테이션입니다. 공유 파일 시스템 위치는 다음과 같습니다.
  - FSCommon: /shared/common
  - FS1: /shared/projects/project1
  - FS2: /shared/projects/project2
- WS1 - 프로젝트 1에 사용되는 Windows기반 워크스테이션입니다. 공유 파일 시스템 위치는 다음과 같습니다.
  - FSCommon: S:\
  - FS1: Z:\
  - FS2: 사용할 수 없음
- WS1 - 프로젝트 2에 사용되는 macOS기반 워크스테이션입니다. 공유 파일 시스템 위치는 다음과 같습니다.
  - FSCommon: /Volumes/common
  - FS1: 사용할 수 없음
  - FS2: /Volumes/projects/project2

마지막으로 플릿의 작업자에 대한 공유 파일 시스템 위치를 정의합니다. 다음 예제에서는이 구성을 로 참조합니다WorkerConfig. 공유 위치는 다음과 같습니다.

- FSCommon: /mnt/common
- FS1: /mnt/projects/project1
- FS2: /mnt/projects/project2

이 구성과 일치하는 공유 파일 시스템, 워크스테이션 또는 작업자를 설정할 필요가 없습니다. 데모를 위해 공유 위치가 존재할 필요는 없습니다.

스토리지 프로파일 및 경로 매핑

스토리지 프로파일을 사용하여 워크스테이션 및 작업자 호스트의 파일 시스템을 모델링합니다. 각 스토리지 프로필은 시스템 구성 중 하나의 운영 체제 및 파일 시스템 레이아웃을 설명합니다. 이 주제에서는 Deadline Cloud가 작업에 대한 경로 매핑 규칙을 생성할 수 있도록 스토리지 프로파일을 사용하여 호스트의 파일 시스템 구성을 모델링하는 방법과 이러한 경로 매핑 규칙이 스토리지 프로파일에서 생성되는 방법을 설명합니다.

Deadline Cloud에 작업을 제출할 때 작업에 대한 선택적 스토리지 프로필 ID를 제공할 수 있습니다. 이 스토리지 프로필은 제출 워크스테이션의 파일 시스템을 설명합니다. 작업 템플릿의 파일 경로가 사용하는 원래 파일 시스템 구성을 설명합니다.

스토리지 프로파일을 플릿과 연결할 수도 있습니다. 스토리지 프로파일은 플릿에 있는 모든 작업자 호스트의 파일 시스템 구성을 설명합니다. 파일 시스템 구성이 다른 작업자가 있는 경우 해당 작업자는 팜의 다른 플릿에 할당되어야 합니다.

경로 매핑 규칙은 작업에서 경로를 지정하는 방법에서 작업자 호스트의 경로 실제 위치로 경로를 다시 매핑하는 방법을 설명합니다. Deadline Cloud는 작업의 스토리지 프로파일에 설명된 파일 시스템 구성을 작업을 실행 중인 플릿의 스토리지 프로파일과 비교하여 이러한 경로 매핑 규칙을 도출합니다.

주제

스토리지 프로파일을 사용하여 공유 파일 시스템 위치 모델링

스토리지 프로파일은 호스트 구성 중 하나의 파일 시스템 구성을 모델링합니다. 샘플 프로젝트 인프라에는 네 가지 호스트 구성이 있습니다. 이 예제에서는 각각에 대해 별도의 스토리지 프로파일을 생성합니다. 다음 중 하나를 사용하여 스토리지 프로파일을 생성할 수 있습니다.

- CreateStorageProfile API
- AWS::Deadline::StorageProfile CloudFormation 리소스
- AWS 콘솔

스토리지 프로파일은 각각 Deadline Cloud에 호스트에서 제출되거나 호스트에서 실행되는 작업과 관련된 파일 시스템 위치의 위치 및 유형을 알려주는 파일 시스템 위치 목록으로 구성됩니다. 스토리지 프로파일은 작업과 관련된 위치만 모델링해야 합니다. 예를 들어 공유 FSCommon 위치는 WS1의 워크스테이션에 S:\있으므로 해당 파일 시스템 위치는 다음과 같습니다.

```
{
    "name": "FSCommon",
    "path": "S:\\",
    "type": "SHARED"
}
```

다음 명령을 사용하여의를 WorkerConfig 사용하여 워크스테이션 구성 WS2, 및 WS1에 대한 스토리지 프로파일WS3과 작업자 구성을 생성합니다. AWS CLI AWS CloudShell

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WSAll \
  --os-family LINUX \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/shared/common"},
      {"name": "FS1", "type":"SHARED", "path":"/shared/projects/project1"},
      {"name": "FS2", "type":"SHARED", "path":"/shared/projects/project2"}
  ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WS1 \
  --os-family WINDOWS \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"S:\\"},
      {"name": "FS1", "type":"SHARED", "path":"Z:\\"}
   ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WS2 \
  --os-family MACOS \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/Volumes/common"},
```

```
      {"name": "FS2", "type":"SHARED", "path":"/Volumes/projects/project2"}
  ]'

aws deadline create-storage-profile --farm-id $FARM_ID \
  --display-name WorkerCfg \
  --os-family LINUX \
  --file-system-locations \
  '[
      {"name": "FSCommon", "type":"SHARED", "path":"/mnt/common"},
      {"name": "FS1", "type":"SHARED", "path":"/mnt/projects/project1"},
      {"name": "FS2", "type":"SHARED", "path":"/mnt/projects/project2"}
  ]'
```

> **Note**
>
> 팜의 모든 스토리지 프로파일에서 name 속성에 대해 동일한 값을 사용하여 스토리지 프로파일의 파일 시스템 위치를 참조해야 합니다. Deadline Cloud는 이름을 비교하여 경로 매핑 규칙을 생성할 때 서로 다른 스토리지 프로파일의 파일 시스템 위치가 동일한 위치를 참조하고 있는지 확인합니다.

플릿의 스토리지 프로파일 구성

플릿의 모든 작업자에 대해 파일 시스템 위치를 모델링하는 스토리지 프로파일을 포함하도록 플릿을 구성할 수 있습니다. 플릿에 있는 모든 작업자의 호스트 파일 시스템 구성은 플릿의 스토리지 프로필과 일치해야 합니다. 파일 시스템 구성이 다른 작업자는 별도의 플릿에 있어야 합니다.

`WorkerConfig` 스토리지 프로파일을 사용하도록 플릿의 구성을 설정하려면 AWS CLI의를 사용합니다AWS CloudShell.

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of FLEET_ID to your fleet's identifier
FLEET_ID=fleet-00112233445566778899aabbccddeeff
# Change the value of WORKER_CFG_ID to your storage profile named WorkerConfig
WORKER_CFG_ID=sp-00112233445566778899aabbccddeeff

FLEET_WORKER_MODE=$( \
  aws deadline get-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
   --query '.configuration.customerManaged.mode' \
)
```

```
FLEET_WORKER_CAPABILITIES=$( \
  aws deadline get-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
   --query '.configuration.customerManaged.workerCapabilities' \
)

aws deadline update-fleet --farm-id $FARM_ID --fleet-id $FLEET_ID \
  --configuration \
  "{
    \"customerManaged\": {
      \"storageProfileId\": \"$WORKER_CFG_ID\",
      \"mode\": $FLEET_WORKER_MODE,
      \"workerCapabilities\": $FLEET_WORKER_CAPABILITIES
    }
  }"
```

대기열의 스토리지 프로필 구성

대기열의 구성에는 대기열에 제출된 작업에 액세스해야 하는 공유 파일 시스템 위치의 대소문자를 구분하는 이름 목록이 포함됩니다. 예를 들어 대기열에 제출된 작업에는 파일 시스템 위치 FSCommon 및가 Q1 필요합니다FS1. 대기열에 제출된 작업에는 파일 시스템 위치와 FSCommon이 Q2 필요합니다FS2.

이러한 파일 시스템 위치가 필요하도록 대기열의 구성을 설정하려면 다음 스크립트를 사용합니다.

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of QUEUE2_ID to queue Q2's identifier
QUEUE2_ID=queue-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --required-file-system-location-names-to-add FSComm FS1

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --required-file-system-location-names-to-add FSComm FS2
```

대기열의 구성에는에 제출된 작업 및 해당 대기열과 연결된 플릿에 적용되는 허용된 스토리지 프로파일 목록도 포함됩니다. 대기열에 필요한 모든 파일 시스템 위치에 대해 파일 시스템 위치를 정의하는 스토리지 프로파일만 대기열의 허용된 스토리지 프로파일 목록에 허용됩니다.

대기열에 허용되는 스토리지 프로필 목록에 없는 스토리지 프로필과 함께 제출하면 작업이 실패합니다. 언제든지 스토리지 프로파일이 없는 작업을 대기열에 제출할 수 있습니다. WSAll 및 레이블이 지정된 워크스테이션 구성에는 대기열에 필요한 파일 시스템 위치(FSCommon 및 FS1)가 WS1 있습니다Q1. 대기열에 작업을 제출할 수 있어야 합니다. 마찬가지로 워크스테이션 구성 WSAll 및는 대기열에 대한 요구 사항을 WS2 충족합니다Q2. 해당 대기열에 작업을 제출할 수 있어야 합니다. 다음 스크립트를 사용하여 이러한 스토리지 프로파일로 작업을 제출할 수 있도록 두 대기열 구성을 모두 업데이트합니다.

```
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff
# Change the value of WS1 to the identifier of the WS1 storage profile
WS1_ID=sp-00112233445566778899aabbccddeeff
# Change the value of WS2 to the identifier of the WS2 storage profile
WS2_ID=sp-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WSALL_ID $WS1_ID

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --allowed-storage-profile-ids-to-add $WSALL_ID $WS2_ID
```

대기열에 허용되는 WS2 스토리지 프로파일 목록에 스토리지 프로파일을 추가하면 실패Q1합니다.

```
$ aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WS2_ID

An error occurred (ValidationException) when calling the UpdateQueue operation: Storage
 profile id: sp-00112233445566778899aabbccddeeff does not have required file system
 location: FS1
```

스토리지 WS2 프로파일에 해당 대기열에 Q1 필요한 라는 파일 시스템 위치에 대한 정의FS1가 포함되어 있지 않기 때문입니다.

대기열의 허용된 스토리지 프로필 목록에 없는 스토리지 프로필로 구성된 플릿 연결도 실패합니다. 예제:

```
$ aws deadline create-queue-fleet-association --farm-id $FARM_ID \
   --fleet-id $FLEET_ID \
   --queue-id $QUEUE1_ID
```

```
An error occurred (ValidationException) when calling the CreateQueueFleetAssociation
 operation: Mismatch between storage profile ids.
```

오류를 해결하려면 대기열Q1과 대기열 모두에 허용되는 스토리지 프로파일 WorkerConfig 목록에 라는 스토리지 프로파일을 추가합니다Q2. 그런 다음 플릿의 작업자가 두 대기열에서 작업을 실행할 수 있도록 플릿을 이러한 대기열과 연결합니다.

```
# Change the value of FLEET_ID to your fleet's identifier
FLEET_ID=fleet-00112233445566778899aabbccddeeff
# Change the value of WORKER_CFG_ID to your storage profile named WorkerCfg
WORKER_CFG_ID=sp-00112233445566778899aabbccddeeff

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --allowed-storage-profile-ids-to-add $WORKER_CFG_ID

aws deadline update-queue --farm-id $FARM_ID --queue-id $QUEUE2_ID \
  --allowed-storage-profile-ids-to-add $WORKER_CFG_ID

aws deadline create-queue-fleet-association --farm-id $FARM_ID \
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE1_ID

aws deadline create-queue-fleet-association --farm-id $FARM_ID \
  --fleet-id $FLEET_ID \
  --queue-id $QUEUE2_ID
```

스토리지 프로파일에서 경로 매핑 규칙 도출

경로 매핑 규칙은 작업에서 작업자 호스트의 경로 실제 위치로 경로를 다시 매핑하는 방법을 설명합니다. 작업이 작업자에서 실행 중인 경우 작업의 스토리지 프로파일을 작업자 플릿의 스토리지 프로파일과 비교하여 작업에 대한 경로 매핑 규칙을 도출합니다.

Deadline Cloud는 대기열 구성의 각 필수 파일 시스템 위치에 대한 매핑 규칙을 생성합니다. 예를 들어 대기열에 WSAll 스토리지 프로파일과 함께 제출된 작업에는 경로 매핑 규칙Q1이 있습니다.

- FSComm: /shared/common -> /mnt/common
- FS1: /shared/projects/project1 -> /mnt/projects/project1

Deadline Cloud는 FSComm 및 FS1 파일 시스템 위치에 대한 규칙을 생성하지만 WSAll 및 WorkerConfig 스토리지 프로파일이 모두를 정의하더라도 FS2 파일 시스템 위치에 대한 규칙은 생

성하지 않습니다FS2. 이는 대기열 Q1의 필수 파일 시스템 위치 목록이 이기 때문입니다["FSComm", "FS1"].

Open Job Description의 경로 매핑 규칙 파일을 출력하는 작업을 제출한 다음 작업이 완료된 후 세션 로그를 읽어 특정 스토리지 프로파일과 함께 제출된 작업에 사용할 수 있는 경로 매핑 규칙을 확인할 수 있습니다.

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSALL storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

aws deadline create-job --farm-id $FARM_ID --queue-id $QUEUE1_ID \
  --priority 50 \
  --storage-profile-id $WSALL_ID \
  --template-type JSON --template \
  '{
    "specificationVersion": "jobtemplate-2023-09",
    "name": "DemoPathMapping",
    "steps": [
      {
        "name": "ShowPathMappingRules",
        "script": {
          "actions": {
            "onRun": {
              "command": "/bin/cat",
              "args": [ "{{Session.PathMappingRulesFile}}" ]
            }
          }
        }
      }
    ]
  }'
```

Deadline Cloud CLI를 사용하여 작업을 제출하는 경우 구성 settings.storage_profile_id 설정은 CLI로 제출된 작업이 보유할 스토리지 프로파일을 설정합니다. WSAll 스토리지 프로파일로 작업을 제출하려면 다음을 설정합니다.

```
deadline config set settings.storage_profile_id $WSALL_ID
```

샘플 인프라에서 실행 중인 것처럼 고객 관리형 작업자를 실행하려면 Deadline Cloud 사용 설명서의 작업자 에이전트 실행의 절차에 따라 작업자를 실행합니다 AWS CloudShell. 이전에 이러한 지침을 따랐다면 먼저 `~/demoenv-logs` 및 `~/demoenv-persist` 디렉터리를 삭제합니다. 또한 방향이 참조하는 `DEV_FARM_ID` 및 `DEV_CMF_ID` 환경 변수의 값을 다음과 같이 설정한 후 참조합니다.

```
DEV_FARM_ID=$FARM_ID
DEV_CMF_ID=$FLEET_ID
```

작업이 실행된 후 작업의 로그 파일에서 경로 매핑 규칙을 볼 수 있습니다.

```
cat demoenv-logs/${QUEUE1_ID}/*.log
...
JJSON log results (see below)
...
```

로그에는 FS1 및 FSComm 파일 시스템에 대한 매핑이 포함되어 있습니다. 가독성을 위해 다시 포맷된 로그 항목은 다음과 같습니다.

```
{
    "version": "pathmapping-1.0",
    "path_mapping_rules": [
        {
            "source_path_format": "POSIX",
            "source_path": "/shared/projects/project1",
            "destination_path": "/mnt/projects/project1"
        },
        {
            "source_path_format": "POSIX",
            "source_path": "/shared/common",
            "destination_path": "/mnt/common"
        }
    ]
```

스토리지 프로파일이 다른 작업을 제출하여 경로 매핑 규칙이 어떻게 변경되는지 확인할 수 있습니다.

작업 첨부 파일을 사용하여 파일 공유

작업 첨부 파일을 사용하여 파일을 공유 디렉터리에서 작업에 사용할 수 없게 하고, 파일이 공유 디렉터리에 기록되지 않은 경우 출력 파일을 캡처합니다. 작업 연결은 Amazon S3를 사용하여 호스트 간에

파일을 복원합니다. 파일은 S3 버킷에 저장되며 콘텐츠가 변경되지 않은 경우 파일을 업로드할 필요가 없습니다.

호스트는 파일 시스템 위치를 공유하지 않으므로 서비스 관리형 플릿에서 작업을 실행할 때 작업 연결을 사용해야 합니다. 작업 연결은 작업 번들에 셸 또는 Python 스크립트가 포함된 경우와 같이 작업의 입력 또는 출력 파일이 공유 네트워크 파일 시스템에 저장되는 경우 고객 관리형 플릿에도 유용합니다.

Deadline Cloud CLI 또는 Deadline Cloud 제출자를 사용하여 작업 번들을 제출하면 작업 첨부 파일은 작업의 스토리지 프로파일과 대기열의 필수 파일 시스템 위치를 사용하여 작업자 호스트에 있지 않으며 작업 제출의 일부로 Amazon S3에 업로드해야 하는 입력 파일을 식별합니다. 또한 이러한 스토리지 프로파일은 Deadline Cloud가 워크스테이션에서 사용할 수 있도록 Amazon S3에 업로드해야 하는 작업자 호스트 위치의 출력 파일을 식별하는 데 도움이 됩니다.

작업 연결 예제에서는 샘플 프로젝트 인프라 및의 팜, 플릿, 대기열 및 스토리지 프로파일 구성을 사용합니다스토리지 프로파일 및 경로 매핑. 이 섹션보다 먼저 해당 섹션을 살펴봐야 합니다.

다음 예제에서는 샘플 작업 번들을 시작점으로 사용한 다음 수정하여 작업 첨부 파일의 기능을 살펴봅니다. 작업 번들은 작업에서 작업 첨부 파일을 사용하는 가장 좋은 방법입니다. 디렉터리의 Open Job Description 작업 템플릿을 작업 번들을 사용하는 작업에 필요한 파일 및 디렉터리를 나열하는 추가 파일과 결합합니다. 작업 번들에 대한 자세한 내용은 섹션을 참조하세요Deadline Cloud에 대한 Open Job Description(OpenJD) 템플릿.

작업으로 파일 제출

Deadline Cloud를 사용하면 작업 워크플로가 작업자 호스트의 공유 파일 시스템 위치에서 사용할 수 없는 입력 파일에 액세스할 수 있습니다. 작업 연결을 사용하면 렌더링 작업이 로컬 워크스테이션 드라이브 또는 서비스 관리형 플릿 환경에만 있는 파일에 액세스할 수 있습니다. 작업 번들을 제출할 때 작업에 필요한 입력 파일 및 디렉터리 목록을 포함할 수 있습니다. Deadline Cloud는 공유되지 않은 이러한 파일을 식별하고 로컬 시스템에서 Amazon S3로 업로드한 다음 작업자 호스트로 다운로드합니다. 입력 자산을 렌더 노드로 전송하는 프로세스를 간소화하여 분산 작업 실행에 필요한 모든 파일에 액세스할 수 있도록 합니다.

작업 번들에서 직접 작업에 대한 파일을 지정하고, 환경 변수 또는 스크립트를 사용하여 제공하는 작업 템플릿의 파라미터를 사용하고, 작업의 `assets_references` 파일을 사용할 수 있습니다. 이러한 방법 중 하나 또는 세 가지 방법의 조합을 사용할 수 있습니다. 로컬 워크스테이션에서 변경된 파일만 업로드하도록 작업에 대한 번들의 스토리지 프로파일을 지정할 수 있습니다.

이 섹션에서는 GitHub의 예제 작업 번들을 사용하여 Deadline Cloud가 업로드할 작업의 파일을 식별하는 방법, Amazon S3에서 해당 파일이 구성되는 방법, 작업을 처리하는 작업자 호스트가 사용할 수 있는 방법을 보여줍니다.

주제

- Deadline Cloud가 Amazon S3에 파일을 업로드하는 방법
- Deadline Cloud가 업로드할 파일을 선택하는 방법
- 작업에서 작업 연결 입력 파일을 찾는 방법

Deadline Cloud가 Amazon S3에 파일을 업로드하는 방법

이 예제는 Deadline Cloud가 워크스테이션 또는 작업자 호스트에서 Amazon S3로 파일을 업로드하여 공유할 수 있도록 하는 방법을 보여줍니다. GitHub 및 Deadline Cloud CLI의 샘플 작업 번들을 사용하여 작업을 제출합니다.

먼저 Deadline Cloud 샘플 GitHub 리포지토리를 AWS CloudShell 환경에 복제한 다음 `job_attachments_devguide` 작업 번들을 홈 디렉터리에 복사합니다.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide ~/
```

Deadline Cloud CLI를 설치하여 작업 번들을 제출합니다.

```
pip install deadline --upgrade
```

`job_attachments_devguide` 작업 번들에는 파일 시스템 위치가 작업 파라미터로 전달되는 bash 셸 스크립트를 실행하는 작업이 포함된 단일 단계가 있습니다. 작업 파라미터의 정의는 다음과 같습니다.

```
...
- name: ScriptFile
  type: PATH
  default: script.sh
  dataFlow: IN
  objectType: FILE
...
```

`dataFlow` 속성 값은 작업 연결에 `ScriptFile` 파라미터 값이 작업에 대한 입력임을 `IN` 알려줍니다. `default` 속성 값은 작업 번들 디렉터리의 상대 위치이지만 절대 경로일 수도 있습니다. 이 파라미터 정의는 작업 번들의 디렉터리에 있는 `script.sh` 파일을 작업을 실행하는 데 필요한 입력 파일로 선언합니다.

그런 다음 Deadline Cloud CLI에 스토리지 프로파일이 구성되어 있지 않은지 확인한 다음 작업을 대기열에 제출합니다. Q1

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id ''

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

이 명령이 실행된 후 Deadline Cloud CLI의 출력은 다음과 같습니다.

```
Submitting to Queue: Q1
...
Hashing Attachments  [####################################]  100%
Hashing Summary:
    Processed 1 file totaling 39.0 B.
    Skipped re-processing 0 files totaling 0.0 B.
    Total processing time of 0.0327 seconds at 1.19 KB/s.

Uploading Attachments  [####################################]  100%
Upload Summary:
    Processed 1 file totaling 39.0 B.
    Skipped re-processing 0 files totaling 0.0 B.
    Total processing time of 0.25639 seconds at 152.0 B/s.

Waiting for Job to be created...
Submitted job bundle:
   job_attachments_devguide/
Job creation completed successfully
job-74148c13342e4514b63c7a7518657005
```

작업을 제출하면 Deadline Cloud는 먼저 `script.sh` 파일을 해시한 다음 Amazon S3에 업로드합니다.

Deadline Cloud는 S3 버킷을 콘텐츠 주소 지정 스토리지로 취급합니다. 파일은 S3 객체에 업로드됩니다. 객체 이름은 파일 콘텐츠의 해시에서 파생됩니다. 두 파일의 콘텐츠가 동일한 경우 파일의 위치나 이름에 관계없이 해시 값이 동일합니다. 이 콘텐츠 주소 지정 스토리지를 사용하면 Deadline Cloud가 이미 사용 가능한 파일을 업로드하지 않아도 됩니다.

AWS CLI를 사용하여 Amazon S3에 업로드된 객체를 볼 수 있습니다.

```
# The name of queue `Q1`'s job attachments S3 bucket
Q1_S3_BUCKET=$(
  aws deadline get-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
    --query 'jobAttachmentSettings.s3BucketName' | tr -d '"'
)

aws s3 ls s3://$Q1_S3_BUCKET --recursive
```

두 개의 객체가 S3에 업로드되었습니다.

- `DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128` -의 내용입니다`script.sh`. 객체 키`87cb19095dd5d78fcaf56384ef0e6241`의 값은 파일 내용의 해시이며 확장자는 해시 값이 128비트 xxhash로 계산되었음을 `xxh128` 나타냅니다.
- `DeadlineCloud/Manifests/<farm-id>/<queue-id>/Inputs/<guid>/a1d221c7fd97b08175b3872a37428e8c_input` - 작업 제출을 위한 매니페스트 객체입니다. 값 `<farm-id>`, 및 `<queue-id><guid>`는 팜 식별자, 대기열 식별자 및 무작위 16진수 값입니다. 이 예제`a1d221c7fd97b08175b3872a37428e8c`의 값은가 위치한 디렉터리`/home/cloudshell-user/job_attachments_devguide`인 문자열에서 계산된 해시 값`script.sh`입니다.

매니페스트 객체에는 작업 제출의 일부로 S3에 업로드된 특정 루트 경로의 입력 파일에 대한 정보가 포함됩니다. 이 매니페스트 파일(`aws s3 cp s3://$Q1_S3_BUCKET/<objectname>`)을 다운로드합니다. 콘텐츠는 다음과 유사합니다.

```
{
    "hashAlg": "xxh128",
    "manifestVersion": "2023-03-03",
    "paths": [
        {
            "hash": "87cb19095dd5d78fcaf56384ef0e6241",
            "mtime": 1721147454416085,
            "path": "script.sh",
            "size": 39
        }
    ],
    "totalSize": 39
}
```

이는 파일이 업로드`script.sh`되었고 해당 파일 콘텐츠의 해시가 임을 나타냅니다`87cb19095dd5d78fcaf56384ef0e6241`. 이 해시 값은 객체 이름의 값과 일치합니다`DeadlineCloud/Data/87cb19095dd5d78fcaf56384ef0e6241.xxh128`. Deadline Cloud는 이 파일의 콘텐츠에 대해 다운로드할 객체를 파악하는 데 사용됩니다.

이 파일의 전체 스키마는 GitHub에서 사용할 수 있습니다.

CreateJob 작업을 사용할 때 매니페스트 객체의 위치를 설정할 수 있습니다. GetJob 작업을 사용하여 위치를 확인할 수 있습니다.

```
{
    "attachments": {
        "file system": "COPIED",
        "manifests": [
            {
                "inputManifestHash": "5b0db3d311805ea8de7787b64cbbe8b3",
                "inputManifestPath": "<farm-id>/<queue-id>/Inputs/<guid>/
a1d221c7fd97b08175b3872a37428e8c_input",
                "rootPath": "/home/cloudshell-user/job_attachments_devguide",
                "rootPathFormat": "posix"
            }
        ]
    },
    ...
}
```

Deadline Cloud가 업로드할 파일을 선택하는 방법

작업 첨부 파일이 Amazon S3에 업로드할 때 작업에 대한 입력으로 고려하는 파일 및 디렉터리는 다음과 같습니다.

- 또는 값을 사용하여 작업 번들의 작업 템플릿에 정의된 모든 `PATH`유형 작업 파라미터의 `dataFlow` 값`IN`입니다`INOUT`.
- 작업 번들의 자산 참조 파일에 입력으로 나열된 파일 및 디렉터리입니다.

스토리지 프로파일이 없는 작업을 제출하면 업로드 대상으로 간주되는 모든 파일이 업로드됩니다. 스토리지 프로파일이 있는 작업을 제출하면 파일이 대기열에 필요한 파일 시스템 위치이기도 한 스토리지 프로파일의 `SHARED`유형 파일 시스템 위치에 있는 경우 Amazon S3에 업로드되지 않습니다. 이러한 위치는 작업을 실행하는 작업자 호스트에서 사용할 수 있을 것으로 예상되므로 S3에 업로드할 필요가 없습니다.

이 예제에서는 AWS CloudShell 환경의 WSAll에서 SHARED 파일 시스템 위치를 생성한 다음 해당 파일 시스템 위치에 파일을 추가합니다. 다음 명령을 사용합니다.

```
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

sudo mkdir -p /shared/common /shared/projects/project1 /shared/projects/project2
sudo chown -R cloudshell-user:cloudshell-user /shared

for d in /shared/common /shared/projects/project1 /shared/projects/project2; do
  echo "File contents for $d" > ${d}/file.txt
done
```

그런 다음 작업에 대한 입력으로 생성한 모든 파일이 포함된 작업 번들에 자산 참조 파일을 추가합니다. 다음 명령을 사용합니다.

```
cat > ${HOME}/job_attachments_devguide/asset_references.yaml << EOF
assetReferences:
  inputs:
    filenames:
    - /shared/common/file.txt
    directories:
    - /shared/projects/project1
    - /shared/projects/project2
EOF
```

그런 다음 WSAll 스토리지 프로파일이 있는 작업을 제출하도록 Deadline Cloud CLI를 구성한 다음 작업 번들을 제출합니다.

```

# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

Deadline Cloud는 작업을 제출할 때 Amazon S3에 두 개의 파일을 업로드합니다. S3에서 작업에 대한 매니페스트 객체를 다운로드하여 업로드된 파일을 볼 수 있습니다.

```
for manifest in $( \
  aws deadline get-job --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID \
    --query 'attachments.manifests[].inputManifestPath' \
    | jq -r '.[]'
); do
  echo "Manifest object: $manifest"
  aws s3 cp --quiet s3://$Q1_S3_BUCKET/DeadlineCloud/Manifests/$manifest /dev/stdout |
 jq .
done
```

이 예제에는 다음 내용이 포함된 단일 매니페스트 파일이 있습니다.

```
{
    "hashAlg": "xxh128",
    "manifestVersion": "2023-03-03",
    "paths": [
        {
            "hash": "87cb19095dd5d78fcaf56384ef0e6241",
            "mtime": 1721147454416085,
            "path": "home/cloudshell-user/job_attachments_devguide/script.sh",
            "size": 39
        },
        {
            "hash": "af5a605a3a4e86ce7be7ac5237b51b79",
            "mtime": 1721163773582362,
            "path": "shared/projects/project2/file.txt",
            "size": 44
        }
    ],
    "totalSize": 83
}
```

매니페스트에 GetJob 작업을 사용하여가 "/"`rootPath`인지 확인합니다.

```
aws deadline get-job --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID --query
 'attachments.manifests[*]'
```

입력 파일 세트의 루트 경로는 항상 해당 파일의 가장 긴 공통 하위 경로입니다. 작업이 Windows 대신에서 제출되었고 다른 드라이브에 있었기 때문에 공통 하위 경로가 없는 입력 파일이 있는 경우 각 드

라이브에 별도의 루트 경로가 표시됩니다. 매니페스트의 경로는 항상 매니페스트의 루트 경로를 기준으로 하므로 업로드된 입력 파일은 다음과 같습니다.

- `/home/cloudshell-user/job_attachments_devguide/script.sh` - 작업 번들의 스크립트 파일입니다.
- `/shared/projects/project2/file.txt` - 대기열에 필요한 SHARED 파일 시스템 위치 목록에 없는 `WSAll` 스토리지 프로파일의 파일 시스템 위치에 있는 파일입니다Q1.

파일 시스템 위치FSCommon(`/shared/common/file.txt`) 및 FS1 (`/shared/projects/project1/file.txt`)의 파일은 목록에 없습니다. 이는 이러한 파일 시스템 위치가 `WSAll` 스토리지 프로파일SHARED에 있고 둘 다 대기열의 필수 파일 시스템 위치 목록에 있기 때문입니다Q1.

GetStorageProfileForQueue 작업을 사용하여 특정 스토리지 프로파일과 함께 제출된 작업에 SHARED 대해 고려되는 파일 시스템 위치를 볼 수 있습니다. `WSAll` 대기열의 스토리지 프로파일을 쿼리하려면 다음 명령을 Q1 사용합니다.

```
aws deadline get-storage-profile --farm-id $FARM_ID --storage-profile-id $WSALL_ID

aws deadline get-storage-profile-for-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID --
storage-profile-id $WSALL_ID
```

작업에서 작업 연결 입력 파일을 찾는 방법

작업이 Deadline Cloud가 작업 첨부 파일을 사용하여 Amazon S3에 업로드하는 파일을 사용하려면 작업자 호스트의 파일 시스템을 통해 해당 파일을 사용할 수 있어야 합니다. 작업 세션이 작업자 호스트에서 실행되면 Deadline Cloud는 작업에 대한 입력 파일을 작업자 호스트의 로컬 드라이브의 임시 디렉터리에 다운로드하고 작업의 각 루트 경로에 대한 경로 매핑 규칙을 로컬 드라이브의 파일 시스템 위치에 추가합니다.

이 예제에서는 AWS CloudShell 탭에서 Deadline Cloud 작업자 에이전트를 시작합니다. 이전에 제출한 작업의 실행을 완료한 다음 로그 디렉터리에서 작업 로그를 삭제합니다.

```
rm -rf ~/devdemo-logs/queue-*
```

다음 스크립트는 작업 번들을 수정하여 세션의 임시 작업 디렉터리에 있는 모든 파일과 경로 매핑 규칙 파일의 내용을 표시한 다음 수정된 번들로 작업을 제출합니다.

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
```

```
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

cat > ~/job_attachments_devguide/script.sh << EOF
#!/bin/bash

echo "Session working directory is: \$(pwd)"
echo
echo "Contents:"
find . -type f
echo
echo "Path mapping rules file: \$1"
jq . \$1
EOF

cat > ~/job_attachments_devguide/template.yaml << EOF
specificationVersion: jobtemplate-2023-09
name: "Job Attachments Explorer"
parameterDefinitions:
- name: ScriptFile
  type: PATH
  default: script.sh
  dataFlow: IN
  objectType: FILE
steps:
- name: Step
  script:
    actions:
      onRun:
        command: /bin/bash
        args:
        - "{{Param.ScriptFile}}"
        - "{{Session.PathMappingRulesFile}}"
EOF

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/
```

AWS CloudShell 환경에서 작업자가 작업을 실행한 후 작업 실행 로그를 볼 수 있습니다.

```
cat demoenv-logs/queue-*/session*.log
```

로그는 세션에서 첫 번째로 발생하는 작업이 작업자에게 다운로드되는 두 개의 입력 파일임을 보여줍니다.

```
2024-07-17 01:26:37,824 INFO ================================================
2024-07-17 01:26:37,825 INFO --------- Job Attachments Download for Job
2024-07-17 01:26:37,825 INFO ================================================
2024-07-17 01:26:37,825 INFO Syncing inputs using Job Attachments
2024-07-17 01:26:38,116 INFO Downloaded 142.0 B / 186.0 B of 2 files (Transfer rate:
 0.0 B/s)
2024-07-17 01:26:38,174 INFO Downloaded 186.0 B / 186.0 B of 2 files (Transfer rate:
 733.0 B/s)
2024-07-17 01:26:38,176 INFO Summary Statistics for file downloads:
Processed 2 files totaling 186.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.09752 seconds at 1.91 KB/s.
```

다음은 작업에서 `script.sh` 실행한 출력입니다.

- 작업이 제출될 때 업로드된 입력 파일은 세션의 임시 디렉터리에서 이름이 "assetroot"로 시작하는 디렉터리 아래에 있습니다.
- 입력 파일의 경로는 작업의 입력 매니페스트()에 대한 루트 경로 대신 "assetroot" 디렉터리를 기준으로 재배치되었습니다"/".
- 경로 매핑 규칙 파일에는 "assetroot" 디렉터리의 절대 경로"/"로 다시 매핑하는 추가 규칙이 포함되어 있습니다.

예제:

```
2024-07-17 01:26:38,264 INFO Output:
2024-07-17 01:26:38,267 INFO Session working directory is: /sessions/session-5b33f
2024-07-17 01:26:38,267 INFO
2024-07-17 01:26:38,267 INFO Contents:
2024-07-17 01:26:38,269 INFO ./tmp_xdhbsdo.sh
2024-07-17 01:26:38,269 INFO ./tmpdi00052b.json
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/shared/projects/project2/
file.txt
2024-07-17 01:26:38,269 INFO ./assetroot-assetroot-3751a/home/cloudshell-user/
job_attachments_devguide/script.sh
2024-07-17 01:26:38,269 INFO
```

```
2024-07-17 01:26:38,270 INFO Path mapping rules file: /sessions/session-5b33f/
tmpdi00052b.json
2024-07-17 01:26:38,282 INFO {
2024-07-17 01:26:38,282 INFO   "version": "pathmapping-1.0",
2024-07-17 01:26:38,282 INFO   "path_mapping_rules": [
2024-07-17 01:26:38,282 INFO     {
2024-07-17 01:26:38,282 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,282 INFO       "source_path": "/shared/projects/project1",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/projects/project1"
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/shared/common",
2024-07-17 01:26:38,283 INFO       "destination_path": "/mnt/common"
2024-07-17 01:26:38,283 INFO     },
2024-07-17 01:26:38,283 INFO     {
2024-07-17 01:26:38,283 INFO       "source_path_format": "POSIX",
2024-07-17 01:26:38,283 INFO       "source_path": "/",
2024-07-17 01:26:38,283 INFO       "destination_path": "/sessions/session-5b33f/
assetroot-assetroot-3751a"
2024-07-17 01:26:38,283 INFO     }
2024-07-17 01:26:38,283 INFO   ]
2024-07-17 01:26:38,283 INFO }
```

> **Note**
>
> 제출하는 작업에 서로 다른 루트 경로를 가진 여러 매니페스트가 있는 경우 각 루트 경로에 대해 서로 다른 "assetroot"-named 디렉터리가 있습니다.

입력 파일, 디렉터리 또는 파일 시스템 위치 중 하나의 재배치된 파일 시스템 위치를 참조해야 하는 경우 작업에서 경로 매핑 규칙 파일을 처리하고 재매핑을 직접 수행하거나 작업 번들의 작업 템플릿에 PATH 유형 작업 파라미터를 추가하고 해당 파라미터의 값으로 재매핑해야 하는 값을 전달할 수 있습니다. 예를 들어 다음 예제에서는 이러한 작업 파라미터 중 하나를 갖도록 작업 번들을 수정한 다음 파일 시스템 위치가 값으로 인 작업을 제출합니다/shared/projects/project2.

```
cat > ~/job_attachments_devguide/template.yaml << EOF
specificationVersion: jobtemplate-2023-09
name: "Job Attachments Explorer"
parameterDefinitions:
- name: LocationToRemap
  type: PATH
```

```
steps:
- name: Step
  script:
    actions:
      onRun:
        command: /bin/echo
        args:
        - "The location of {{RawParam.LocationToRemap}} in the session is
 {{Param.LocationToRemap}}"
EOF

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID
 job_attachments_devguide/ \
  -p LocationToRemap=/shared/projects/project2
```

이 작업의 실행에 대한 로그 파일에는 다음과 같은 출력이 포함됩니다.

```
2024-07-17 01:40:35,283 INFO Output:
2024-07-17 01:40:35,284 INFO The location of /shared/projects/project2 in the session
 is /sessions/session-5b33f/assetroot-assetroot-3751a
```

작업에서 출력 파일 가져오기

이 예제는 Deadline Cloud가 작업이 생성하는 출력 파일을 식별하는 방법, Amazon S3에 해당 파일을 업로드할지 여부, 워크스테이션에서 해당 출력 파일을 가져올 수 있는 방법을 보여줍니다.

이 예제에서는 `job_attachments_devguide_output` 작업 번들 대신 `job_attachments_devguide` 작업 번들을 사용합니다. 먼저 Deadline Cloud 샘플 GitHub 리포지토리의 복제본에서 AWS CloudShell 환경의 번들 사본을 만듭니다.

```
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide_output ~/
```

이 작업 번들과 `job_attachments_devguide` 작업 번들의 중요한 차이점은 작업 템플릿에 새 작업 파라미터를 추가하는 것입니다.

```
...
parameterDefinitions:
...
- name: OutputDir
  type: PATH
  objectType: DIRECTORY
```

```
  dataFlow: OUT
  default: ./output_dir
  description: This directory contains the output for all steps.
...
```

파라미터의 dataFlow 속성에는 값이 있습니다OUT. Deadline Cloud는 또는 값이 있는 dataFlow 작업 파라미터의 값을 작업의 출력OUTINOUT으로 사용합니다. 이러한 종류의 작업 파라미터에 값으로 전달된 파일 시스템 위치가 작업을 실행하는 작업자의 로컬 파일 시스템 위치로 다시 매핑되는 경우 Deadline Cloud는 해당 위치에서 새 파일을 찾아 작업 출력으로 Amazon S3에 업로드합니다.

작동 방식을 확인하려면 먼저 AWS CloudShell 탭에서 Deadline Cloud 작업자 에이전트를 시작합니다. 이전에 제출한 작업의 실행을 완료합니다. 그런 다음 로그 디렉터리에서 작업 로그를 삭제합니다.

```
rm -rf ~/devdemo-logs/queue-*
```

그런 다음이 작업 번들을 사용하여 작업을 제출합니다. CloudShell에서 실행 중인 작업자를 실행한 후 로그를 확인합니다.

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID ./
job_attachments_devguide_output
```

로그는 파일이 출력으로 감지되어 Amazon S3에 업로드되었음을 보여줍니다.

```
2024-07-17 02:13:10,873 INFO ------------------------------------------------
2024-07-17 02:13:10,873 INFO Uploading output files to Job Attachments
2024-07-17 02:13:10,873 INFO ------------------------------------------------
2024-07-17 02:13:10,873 INFO Started syncing outputs using Job Attachments
2024-07-17 02:13:10,955 INFO Found 1 file totaling 117.0 B in output directory: /
sessions/session-7efa/assetroot-assetroot-3751a/output_dir
2024-07-17 02:13:10,956 INFO Uploading output manifest to
 DeadlineCloud/Manifests/farm-0011/queue-2233/job-4455/step-6677/
task-6677-0/2024-07-17T02:13:10.835545Z_sessionaction-8899-1/
c6808439dfc59f86763aff5b07b9a76c_output
```

```
2024-07-17 02:13:10,988 INFO Uploading 1 output file to S3: s3BucketName/DeadlineCloud/
Data
2024-07-17 02:13:11,011 INFO Uploaded 117.0 B / 117.0 B of 1 file (Transfer rate: 0.0
 B/s)
2024-07-17 02:13:11,011 INFO Summary Statistics for file uploads:
Processed 1 file totaling 117.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.02281 seconds at 5.13 KB/s.
```

또한 로그는 Deadline Cloud가 대기열의 작업 첨부 파일에서 사용하도록 구성된 Amazon S3 버킷에 새 매니페스트 객체를 생성했음을 보여줍니다Q1. 매니페스트 객체의 이름은 출력을 생성한 작업의 팜, 대기열, 작업, 단계, 작업, 타임스탬프 및 sessionaction 식별자에서 파생됩니다. 이 매니페스트 파일을 다운로드하여 Deadline Cloud가이 작업에 대한 출력 파일을 배치한 위치를 확인합니다.

```
# The name of queue `Q1`'s job attachments S3 bucket
Q1_S3_BUCKET=$(
  aws deadline get-queue --farm-id $FARM_ID --queue-id $QUEUE1_ID \
    --query 'jobAttachmentSettings.s3BucketName' | tr -d '"'
)

# Fill this in with the object name from your log
OBJECT_KEY="DeadlineCloud/Manifests/..."

aws s3 cp --quiet s3://$Q1_S3_BUCKET/$OBJECT_KEY /dev/stdout | jq .
```

매니페스트는 다음과 같습니다.

```
{
  "hashAlg": "xxh128",
  "manifestVersion": "2023-03-03",
  "paths": [
    {
      "hash": "34178940e1ef9956db8ea7f7c97ed842",
      "mtime": 1721182390859777,
      "path": "output_dir/output.txt",
      "size": 117
    }
  ],
  "totalSize": 117
}
```

이는 작업 입력 파일이 저장되는 것과 동일한 방식으로 출력 파일의 콘텐츠가 Amazon S3에 저장됨을 보여줍니다. 입력 파일과 마찬가지로 출력 파일은 파일의 해시와 접두사가 포함된 객체 이름과 함께 S3에 저장됩니다`DeadlineCloud/Data`.

```
$ aws s3 ls --recursive s3://$Q1_S3_BUCKET | grep 34178940e1ef9956db8ea7f7c97ed842
2024-07-17 02:13:11        117 DeadlineCloud/
Data/34178940e1ef9956db8ea7f7c97ed842.xxh128
```

Deadline Cloud 모니터 또는 Deadline Cloud CLI를 사용하여 워크스테이션에 작업 출력을 다운로드할 수 있습니다.

```
deadline job download-output --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id $JOB_ID
```

제출된 `OutputDir` 작업의 작업 파라미터 값은 `./output_dir`이므로 출력은 작업 번들 디렉터리 `output_dir` 내에서 라는 디렉터리에 다운로드됩니다. 절대 경로 또는 다른 상대 위치를의 값으로 지정한 경우 `OutputDir`출력 파일이 대신 해당 위치에 다운로드됩니다.

```
$ deadline job download-output --farm-id $FARM_ID --queue-id $QUEUE1_ID --job-id
 $JOB_ID
Downloading output from Job 'Job Attachments Explorer: Output'

Summary of files to download:
    /home/cloudshell-user/job_attachments_devguide_output/output_dir/output.txt (1
 file)

You are about to download files which may come from multiple root directories. Here are
 a list of the current root directories:
[0] /home/cloudshell-user/job_attachments_devguide_output
> Please enter the index of root directory to edit, y to proceed without changes, or n
 to cancel the download (0, y, n) [y]:

Downloading Outputs  [####################################]  100%
Download Summary:
    Downloaded 1 files totaling 117.0 B.
    Total download time of 0.14189 seconds at 824.0 B/s.
    Download locations (total file counts):
        /home/cloudshell-user/job_attachments_devguide_output (1 file)
```

종속 단계의 단계에서 파일 사용

이 예제는 작업의 한 단계가 동일한 작업에서 의존하는 단계에서 출력에 액세스하는 방법을 보여줍니다.

한 단계의 출력을 다른 단계에서 사용할 수 있도록 Deadline Cloud는 세션에서 작업을 실행하기 전에 해당 출력을 다운로드하는 추가 작업을 세션에 추가합니다. 출력을 사용해야 하는 단계의 종속성으로 해당 단계를 선언하여에서 출력을 다운로드할 단계를 지정합니다.

이 예제에서는 `job_attachments_devguide_output` 작업 번들을 사용합니다. 먼저 Deadline Cloud 샘플 GitHub 리포지토리의 복제본에서 AWS CloudShell 환경을 복사합니다. 기존 단계 이후에만 실행되고 해당 단계의 출력을 사용하는 종속 단계를 추가하도록 수정합니다.

```
cp -r deadline-cloud-samples/job_bundles/job_attachments_devguide_output ~/

cat >> job_attachments_devguide_output/template.yaml << EOF
- name: DependentStep
  dependencies:
  - dependsOn: Step
  script:
    actions:
      onRun:
        command: /bin/cat
        args:
        - "{{Param.OutputDir}}/output.txt"
EOF
```

이 수정된 작업 번들로 생성된 작업은 두 개의 개별 세션으로 실행되며, 하나는 "Step" 단계의 작업에 대한 세션이고 다른 하나는 "DependentStep" 단계의 작업에 대한 세션입니다.

먼저 CloudShell 탭에서 Deadline Cloud 작업자 에이전트를 시작합니다. 이전에 제출한 작업의 실행을 완료한 다음 로그 디렉터리에서 작업 로그를 삭제합니다.

```
rm -rf ~/devdemo-logs/queue-*
```

다음으로 수정된 작업 번들을 사용하여 `job_attachments_devguide_output` 작업을 제출합니다. CloudShell 환경의 작업자에서 실행이 완료될 때까지 기다립니다. 두 세션의 로그를 살펴봅니다.

```
# Change the value of FARM_ID to your farm's identifier
FARM_ID=farm-00112233445566778899aabbccddeeff
```

```
# Change the value of QUEUE1_ID to queue Q1's identifier
QUEUE1_ID=queue-00112233445566778899aabbccddeeff
# Change the value of WSALL_ID to the identifier of the WSAll storage profile
WSALL_ID=sp-00112233445566778899aabbccddeeff

deadline config set settings.storage_profile_id $WSALL_ID

deadline bundle submit --farm-id $FARM_ID --queue-id $QUEUE1_ID ./
job_attachments_devguide_output

# Wait for the job to finish running, and then:

cat demoenv-logs/queue-*/session-*
```

라는 단계의 작업에 대한 세션 로그에는 두 DependentStep가지 별도의 다운로드 작업이 실행됩니다.

```
2024-07-17 02:52:05,666 INFO ================================================
2024-07-17 02:52:05,666 INFO --------- Job Attachments Download for Job
2024-07-17 02:52:05,667 INFO ================================================
2024-07-17 02:52:05,667 INFO Syncing inputs using Job Attachments
2024-07-17 02:52:05,928 INFO Downloaded 207.0 B / 207.0 B of 1 file (Transfer rate: 0.0
 B/s)
2024-07-17 02:52:05,929 INFO Summary Statistics for file downloads:
Processed 1 file totaling 207.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.03954 seconds at 5.23 KB/s.

2024-07-17 02:52:05,979 INFO
2024-07-17 02:52:05,979 INFO ================================================
2024-07-17 02:52:05,979 INFO --------- Job Attachments Download for Step
2024-07-17 02:52:05,979 INFO ================================================
2024-07-17 02:52:05,980 INFO Syncing inputs using Job Attachments
2024-07-17 02:52:06,133 INFO Downloaded 117.0 B / 117.0 B of 1 file (Transfer rate: 0.0
 B/s)
2024-07-17 02:52:06,134 INFO Summary Statistics for file downloads:
Processed 1 file totaling 117.0 B.
Skipped re-processing 0 files totaling 0.0 B.
Total processing time of 0.03227 seconds at 3.62 KB/s.
```

첫 번째 작업은 "Step"이라는 단계에서 사용하는 `script.sh` 파일을 다운로드합니다. 두 번째 작업은 해당 단계에서 출력을 다운로드합니다. Deadline Cloud는 해당 단계에서 생성된 출력 매니페스트를 입력 매니페스트로 사용하여 다운로드할 파일을 결정합니다.

동일한 로그의 후반부에서 "DependentStep"이라는 단계의 출력을 볼 수 있습니다.

```
2024-07-17 02:52:06,213 INFO Output:
2024-07-17 02:52:06,216 INFO Script location: /sessions/session-5b33f/
assetroot-assetroot-3751a/script.sh
```

작업에 대한 리소스 제한 생성

Deadline Cloud에 제출된 작업은 여러 작업 간에 공유되는 리소스에 따라 달라질 수 있습니다. 예를 들어 팜에는 특정 리소스에 대한 유동 라이선스보다 더 많은 작업자가 있을 수 있습니다. 또는 공유 파일 서버는 제한된 수의 작업자에게만 동시에 데이터를 제공할 수 있습니다. 경우에 따라 하나 이상의 작업이 이러한 리소스를 모두 클레임하여 새 작업자가 시작할 때 사용할 수 없는 리소스로 인해 오류가 발생할 수 있습니다.

이를 해결하기 위해 이러한 제한된 리소스에 대한 제한을 사용할 수 있습니다. Deadline Cloud는 제한된 리소스의 가용성을 고려하고 해당 정보를 사용하여 리소스를 사용할 수 없는 리소스로 인해 작업이 실패할 가능성이 낮도록 새 작업자가 시작할 때 리소스를 사용할 수 있도록 합니다.

전체 팜에 대한 제한이 생성됩니다. 대기열에 제출된 작업은 대기열과 연결된 제한만 획득할 수 있습니다. 대기열과 연결되지 않은 작업에 대한 제한을 지정하면 작업이 호환되지 않으며 실행되지 않습니다.

제한을 사용하려면

- 한도 생성
- 한도와 대기열 연결
- 제한이 필요한 작업 제출

> **Note**
>
> 제한과 연결되지 않은 대기열에 제한된 리소스가 있는 작업을 실행하는 경우 해당 작업은 모든 리소스를 사용할 수 있습니다. 제한된 리소스가 있는 경우 리소스를 사용하는 대기열에 있는 작업의 모든 단계가 제한과 연결되어 있는지 확인합니다.

팜에 정의되고 대기열과 연결되며 작업에 지정된 제한의 경우 다음 네 가지 중 하나가 발생할 수 있습니다.

- 제한을 생성하고 대기열에 연결한 다음 작업 템플릿에 제한을 지정하면 작업이 실행되고 제한에 정의된 리소스만 사용됩니다.
- 제한을 생성하여 작업 템플릿에 지정하지만 제한을 대기열과 연결하지 않으면 작업이 호환되지 않는 것으로 표시되고 실행되지 않습니다.
- 제한을 생성하고 대기열과 연결하지 않으며 작업 템플릿에 제한을 지정하지 않으면 작업이 실행되지만 제한을 사용하지 않습니다.
- 제한을 전혀 사용하지 않으면 작업이 실행됩니다.

제한을 여러 대기열에 연결하면 대기열은 제한에 의해 제한된 리소스를 공유합니다. 예를 들어 100개의 제한을 생성하고 대기열 하나가 60개의 리소스를 사용하는 경우 다른 대기열은 40개의 리소스만 사용할 수 있습니다. 리소스가 릴리스되면 모든 대기열의 작업에서 리소스를 가져올 수 있습니다.

Deadline Cloud는 한도에서 제공하는 리소스를 모니터링하는 데 도움이 되는 두 가지 AWS CloudFormation지표를 제공합니다. 현재 사용 중인 리소스 수와 한도에서 사용 가능한 최대 리소스 수를 모니터링할 수 있습니다. 자세한 내용은 Deadline Cloud 개발자 안내서의 리소스 제한 지표를 참조하세요.

작업 템플릿의 작업 단계에 제한을 적용합니다. `hostRequirements` 단계의 `amounts` 섹션에 한도의 금액 요구 사항 이름을 지정하고 동일한의 한도`amountRequirementName`가 작업 대기열과 연결된 경우이 단계에 예약된 작업은 리소스의 한도에 의해 제한됩니다.

단계에 도달하는 제한으로 제한되는 리소스가 필요한 경우 추가 작업자가 해당 단계의 작업을 선택하지 않습니다.

작업 단계에 제한을 두 개 이상 적용할 수 있습니다. 예를 들어 단계에서 서로 다른 두 소프트웨어 라이선스를 사용하는 경우 각 라이선스에 대해 별도의 제한을 적용할 수 있습니다. 단계에 두 가지 제한이 필요하고 리소스 중 하나에 대한 한도에 도달하면 리소스를 사용할 수 있을 때까지 추가 작업자가 해당 단계의 작업을 선택하지 않습니다.

제한 중지 및 삭제

대기열과 제한 간의 연결을 중지하거나 삭제하면 제한을 사용하는 작업은이 제한이 필요한 단계에서 작업 예약을 중지하고 단계에 대한 새 세션 생성을 차단합니다.

준비 상태인 태스크는 준비 상태로 유지되며 대기열과 한도 간의 연결과 함께 태스크가 자동으로 재개됩니다. 작업을 다시 대기열에 넣을 필요가 없습니다.

대기열과 한도 간의 연결을 중지하거나 삭제할 때 작업 실행을 중지하는 방법에 대한 두 가지 선택 사항이 있습니다.

- 작업 중지 및 취소 - 제한을 획득한 세션이 있는 작업자는 모든 작업을 취소합니다.
- 작업 중지 및 완료 - 한도를 획득한 세션이 있는 작업자는 작업을 완료합니다.

콘솔을 사용하여 제한을 삭제하면 작업자는 작업을 완료하면 즉시 또는 결국 실행을 중지합니다. 연결이 삭제되면 다음과 같은 상황이 발생합니다.

- 제한이 필요한 단계는 호환되지 않는 것으로 표시됩니다.
- 제한이 필요하지 않은 단계를 포함하여 이러한 단계가 포함된 전체 작업이 취소됩니다.
- 작업이 호환되지 않음으로 표시되어 있습니다.

한도와 연결된 대기열에 한도의 금액 요구 사항 이름과 일치하는 플릿 기능이 있는 연결된 플릿이 있는 경우 해당 플릿은 지정된 한도로 작업을 계속 처리합니다.

한도 생성

Deadline Cloud 콘솔 또는 Deadline Cloud API의 CreateLimit 작업을 사용하여 제한을 생성합니다. 한도는 팜에 대해 정의되지만 대기열과 연결됩니다. 제한을 생성한 후 하나 이상의 대기열에 연결할 수 있습니다.

제한을 생성하려면

1. Deadline Cloud 콘솔(Deadline Cloud 콘솔) 대시보드에서 대기열을 생성할 팜을 선택합니다.
2. 한도를 추가할 팜을 선택하고 한도 탭을 선택한 다음 한도 생성을 선택합니다.
3. 제한에 대한 세부 정보를 제공합니다. 금액 요구 사항 이름은 작업 템플릿에서 제한을 식별하는 데 사용되는 이름입니다. 접두사와 **`amount.`** 금액 이름으로 시작해야 합니다. 금액 요구 사항 이름은 한도와 연결된 대기열에서 고유해야 합니다.
4. 최대량 설정을 선택하면이 한도에서 허용하는 총 리소스 수입니다. 최대 금액 없음을 선택하면 리소스 사용량이 제한되지 않습니다. 리소스 사용량이 제한되지 않더라도 `CurrentCount` Amazon CloudWatch 지표가 생성되므로 사용량을 추적할 수 있습니다. 자세한 내용은 Deadline Cloud 개발자 안내서의 CloudWatch 지표를 참조하세요.
5. 제한을 사용해야 하는 대기열을 이미 알고 있는 경우 지금 선택할 수 있습니다. 제한을 생성하기 위해 대기열을 연결할 필요가 없습니다.
6. 한도 생성을 선택합니다.

한도와 대기열 연결

한도를 생성한 후 하나 이상의 대기열을 한도와 연결할 수 있습니다. 제한과 연결된 대기열만 제한에 지정된 값을 사용합니다.

Deadline Cloud 콘솔 또는 Deadline Cloud API의 CreateQueueLimitAssociation 작업을 사용하여 대기열과의 연결을 생성합니다.

대기열을 한도와 연결하려면

1. Deadline Cloud 콘솔(Deadline Cloud 콘솔) 대시보드에서 제한을 대기열과 연결할 팜을 선택합니다.
2. 한도 탭을 선택하고 대기열을 연결할 한도를 선택한 다음 한도 편집을 선택합니다.
3. 대기열 연결 섹션에서 한도와 연결할 대기열을 선택합니다.
4. 변경 사항 저장을 선택합니다.

제한이 필요한 작업 제출

제한을 작업 또는 작업 단계의 호스트 요구 사항으로 지정하여 제한을 적용합니다. 단계에서 제한을 지정하지 않고 해당 단계에서 연결된 리소스를 사용하는 경우 작업이 예약될 때 단계의 사용량은 제한에 포함되지 않습니다.

일부 Deadline Cloud 제출자를 사용하면 호스트 요구 사항을 설정할 수 있습니다. 제출자에서 한도의 금액 요구 사항 이름을 지정하여 한도를 적용할 수 있습니다.

제출자가 호스트 요구 사항 추가를 지원하지 않는 경우 작업에 대한 작업 템플릿을 편집하여 제한을 적용할 수도 있습니다.

작업 번들의 작업 단계에 제한을 적용하려면

1. 텍스트 편집기를 사용하여 작업에 대한 작업 템플릿을 엽니다. 작업 템플릿은 작업의 작업 번들 디렉터리에 있습니다. 자세한 내용은 Deadline Cloud 개발자 안내서의 작업 번들을 참조하세요.
2. 제한을 적용할 단계의 단계 정의를 찾습니다.
3. 단계 정의에 다음을 추가합니다. *amount.name* 한도의 금액 요구 사항 이름으로 바꿉니다. 일반적으로를 사용하려면 `min` 값을 1로 설정해야 합니다.

   YAML

   ```
   hostRequirements:
   ```

```
    amounts:
    - name: amount.name
      min: 1
```

JSON

```
"hostRequirements": {
    "amounts": [
        {
            "name": "amount.name",
            "min": "1"
        }
    }
}
```

다음과 같이 작업 단계에 여러 제한을 추가할 수 있습니다. *amount.name_1* 및 *amount.name_2*를 한도의 금액 요구 사항 이름으로 바꿉니다.

YAML

```
  hostRequirements:
    amounts:
    - name: amount.name_1
      min: 1
    - name: amount.name_2
      min: 1
```

JSON

```
"hostRequirements": {
    "amounts": [
        {
            "name": "amount.name_1",
            "min": "1"
        },
        {
            "name": "amount.name_2",
            "min": "1"
        }
    }
```

```
}
```

4. 작업 템플릿에 대한 변경 사항을 저장합니다.

Deadline Cloud에 작업을 제출하는 방법

AWS Deadline Cloud에 작업을 제출하는 방법에는 여러 가지가 있습니다. 이 섹션에서는 Deadline Cloud에서 제공하는 도구를 사용하거나 워크로드에 맞는 사용자 지정 도구를 생성하여 작업을 제출할 수 있는 몇 가지 방법을 설명합니다.

- 터미널에서 - 작업 번들을 처음 개발하는 경우 또는 작업을 제출하는 사용자가 명령줄을 사용하는 것이 편한 경우
- 스크립트에서 - 워크로드 사용자 지정 및 자동화
- 애플리케이션에서 - 사용자의 작업이 애플리케이션에 있는 경우 또는 애플리케이션의 컨텍스트가 중요한 경우.

다음 예제에서는 `deadline` Python 라이브러리와 `deadline` 명령줄 도구를 사용합니다. 둘 다 PyPi에서 사용할 수 있으며 GitHub에서 호스팅됩니다.

주제

- 터미널에서 Deadline Cloud에 작업 제출
- 스크립트를 사용하여 Deadline Cloud에 작업 제출
- 애플리케이션 내에서 작업 제출

터미널에서 Deadline Cloud에 작업 제출

작업 번들과 Deadline Cloud CLI만 사용하면 사용자 또는 더 많은 기술 사용자가 작업 번들 작성을 빠르게 반복하여 작업 제출을 테스트할 수 있습니다. 다음 명령을 사용하여 작업 번들을 제출합니다.

```
deadline bundle submit <path-to-job-bundle>
```

번들에 기본값이 없는 파라미터가 포함된 작업 번들을 제출하는 경우 `-p` / `--parameter` 옵션으로 지정할 수 있습니다.

```
deadline bundle submit <path-to-job-bundle> -p <parameter-name>=<parameter-value> -
p ...
```

사용 가능한 옵션의 전체 목록을 보려면 도움말 명령을 실행합니다.

```
deadline bundle submit --help
```

GUI를 사용하여 Deadline Cloud에 작업 제출

또한 Deadline Cloud CLI에는 사용자가 작업을 제출하기 전에 제공해야 하는 파라미터를 볼 수 있는 그래픽 사용자 인터페이스가 함께 제공됩니다. 사용자가 명령줄과 상호 작용하지 않으려는 경우 대화 상자를 여는 바탕 화면 바로 가기를 작성하여 특정 작업 번들을 제출할 수 있습니다.

```
deadline bundle gui-submit <path-to-job-bundle>
```

사용자가 작업 번들을 선택할 수 있도록 `--browse` 옵션을 사용합니다.

```
deadline bundle gui-submit --browse
```

사용 가능한 옵션의 전체 목록을 보려면 도움말 명령을 실행합니다.

```
deadline bundle gui-submit --help
```

스크립트를 사용하여 Deadline Cloud에 작업 제출

Deadline Cloud에 작업 제출을 자동화하려면 bash, Powershell 및 배치 파일과 같은 도구를 사용하여 작업을 스크립팅할 수 있습니다.

환경 변수 또는 기타 애플리케이션에서 작업 파라미터를 채우는 등의 기능을 추가할 수 있습니다. 여러 작업을 연속으로 제출하거나 제출할 작업 번들 생성을 스크립트로 작성할 수도 있습니다.

Python을 사용하여 작업 제출

또한 Deadline Cloud에는 서비스와 상호 작용할 수 있는 오픈 소스 Python 라이브러리가 있습니다. 소스 코드는 GitHub에서 사용할 수 있습니다.

이 라이브러리는 pip()를 통해 pypi에서 사용할 수 있습니다`pip install deadline`. Deadline Cloud CLI 도구에서 사용하는 것과 동일한 라이브러리입니다.

```
from deadline.client import api

job_bundle_path = "/path/to/job/bundle"
job_parameters = [
```

```
    {
        "name": "parameter_name",
        "value": "parameter_value"
    },
]

job_id = api.create_job_from_job_bundle(
    job_bundle_path,
    job_parameters
)
print(job_id)
```

deadline bundle gui-submit 명령과 같은 대화 상자를 생성하려면에서 show_job_bundle_submitter 함수를 사용할 수 있습니다deadline.client.ui.job_bundle_submitter.

다음 예시에서는 Qt 애플리케이션을 시작하고 작업 번들 제출자를 보여줍니다.

```
# The GUI components must be installed with pip install "deadline[gui]"
import sys
from qtpy.QtWidgets import QApplication
from deadline.client.ui.job_bundle_submitter import show_job_bundle_submitter

app = QApplication(sys.argv)
submitter = show_job_bundle_submitter(browse=True)
submitter.show()
app.exec()
print(submitter.create_job_response)
```

에서 SubmitJobToDeadlineDialog 클래스를 사용하여 대화 상자를 직접 만들 수 있습니다deadline.client.ui.dialogs.submit_job_to_deadline_dialog. 값을 전달하고, 고유한 작업별 탭을 포함하고, 작업 번들이 생성(또는 전달)되는 방식을 결정할 수 있습니다.

애플리케이션 내에서 작업 제출

사용자가 작업을 쉽게 제출할 수 있도록 애플리케이션에서 제공하는 스크립팅 런타임 또는 플러그인 시스템을 사용할 수 있습니다. 사용자는 익숙한 인터페이스를 가지고 있으며 워크로드를 제출할 때 사용자를 지원하는 강력한 도구를 만들 수 있습니다.

애플리케이션에 작업 번들 임베드

이 예제는 애플리케이션에서 사용할 수 있는 작업 번들을 제출하는 방법을 보여줍니다.

사용자에게 이러한 작업 번들에 대한 액세스 권한을 부여하려면 Deadline Cloud CLI를 시작하는 메뉴 항목에 포함된 스크립트를 생성합니다.

다음 스크립트를 사용하면 사용자가 작업 번들을 선택할 수 있습니다.

```
deadline bundle gui-submit --install-gui
```

대신 메뉴 항목에 특정 작업 번들을 사용하려면 다음을 사용합니다.

```
deadline bundle gui-submit </path/to/job/bundle> --install-gui
```

그러면 사용자가 작업 파라미터, 입력 및 출력을 수정한 다음 작업을 제출할 수 있는 대화 상자가 열립니다. 사용자가 애플리케이션에서 제출할 작업 번들마다 메뉴 항목이 다를 수 있습니다.

작업 번들로 제출하는 작업에 제출 간에 유사한 파라미터와 자산 참조가 포함된 경우 기본 작업 번들의 기본값을 입력할 수 있습니다.

애플리케이션에서 정보 가져오기

사용자가 제출에 수동으로 추가할 필요가 없도록 애플리케이션에서 정보를 가져오려면 Deadline Cloud를 애플리케이션과 통합하면 사용자가 애플리케이션을 종료하거나 명령줄 도구를 사용하지 않고도 익숙한 인터페이스를 사용하여 작업을 제출할 수 있습니다.

애플리케이션에 Python 및 pyside/pyqt를 지원하는 스크립팅 런타임이 있는 경우 Deadline Cloud 클라이언트 라이브러리의 GUI 구성 요소를 사용하여 UI를 생성할 수 있습니다. 예제는 GitHub의 Deadline Cloud for Maya integration을 참조하세요.

Deadline Cloud 클라이언트 라이브러리는 강력한 통합 사용자 경험을 제공하는 데 도움이 되는 다음과 같은 작업을 제공합니다.

- 애플리케이션 SDK를 호출하여 환경 변수 및에서 대기열 환경 파라미터, 작업 파라미터 및 자산 참조를 가져옵니다.
- 작업 번들에서 파라미터를 설정합니다. 원본 번들을 수정하지 않으려면 번들 사본을 만들고 사본을 제출해야 합니다.

`deadline bundle gui-submit` 명령을 사용하여 작업 번들을 제출하는 경우 애플리케이션에서 정보를 전달하려면 `parameter_values.yaml` 및 `asset_references.yaml` 파일을 프로그래밍 방식으로 전달해야 합니다. 이러한 파일에 대한 자세한 내용은 섹션을 참조하세요Deadline Cloud에 대한 Open Job Description(OpenJD) 템플릿.

OpenJD에서 제공하는 것보다 더 복잡한 제어가 필요하거나, 사용자로부터 작업을 추상화해야 하거나, 통합이 애플리케이션의 시각적 스타일과 일치하도록 하려면 Deadline Cloud 클라이언트 라이브러리를 호출하여 작업을 제출하는 자체 대화 상자를 작성할 수 있습니다.

Deadline Cloud에서 작업 예약

작업을 생성한 후 AWS Deadline Cloud는 대기열과 연결된 하나 이상의 플릿에서 처리되도록 예약합니다. 특정 작업을 처리하는 플릿은 일정 구성, 플릿에 대해 구성된 기능 및 특정 단계의 호스트 요구 사항에 따라 선택됩니다.

다음 섹션에서는 작업 예약 프로세스에 대한 세부 정보를 제공합니다.

구성 예약

대기열에서 예약 구성을 설정하여 Deadline Cloud가 대기열에서 작업을 예약하는 방법을 구성할 수 있습니다. 예약 구성은 작업자가 작업 간에 분산되는 방식을 제어합니다.

Deadline Cloud 콘솔을 사용하거나 CreateQueue 또는 UpdateQueue APIs.

다음과 같은 세 가지 예약 구성이 있습니다.

- 우선 순위, first-in-first-out(`priorityFifo`) - 가장 높은 우선 순위, 가장 빨리 제출된 작업을 먼저 예약합니다(기본값).
- 우선순위, 균형(`priorityBalanced`) - 가장 높은 우선순위로 작업 간에 작업자를 균등하게 분산합니다.
- 가중, 균형(`weightedBalanced`) - 가중 공식을 사용하여 작업자가 작업 간에 분산되는 방식을 결정합니다.

모든 예약 구성에서 진행 중인 작업은 새 예약 결정이 내려지기 전에 완료될 때까지 실행됩니다. 작업이 실행되는 동안 예약 구성을 변경하면 다음에 작업자가 할당될 때만 변경 사항이 적용됩니다. 실행 중인 작업은 중단되거나 재할당되지 않습니다.

우선 순위, first-in-first-out

우선 순위, first-in-first-out(`priorityFifo`)은 새 대기열의 기본 예약 구성입니다. Deadline Cloud는 우선 순위가 가장 높은 작업에 작업자를 먼저 할당합니다. 여러 작업이 동일한 우선 순위를 공유하는 경우 가장 오래된(가장 빨리 제출된) 작업은 사용 가능한 모든 작업자를 먼저 받습니다.

엄격한 작업 순서를 원하는 경우 우선 순위 FIFO를 사용합니다. 이 구성은 순차 파이프라인 단계 또는 다음 작업이 시작되기 전에 각 작업이 완료되어야 하는 배치 처리와 같이 작업이 제출된 순서대로 한 번에 하나씩 완료해야 하는 경우에 적합합니다.

이 구성에는 추가 파라미터가 없습니다.

우선순위, 균형

우선순위, 균형(`priorityBalanced`)은 가장 높은 우선순위 수준에서 모든 작업에 작업자를 균등하게 분산합니다. 우선순위가 가장 높은 작업이 하나만 있는 경우 Deadline Cloud는 해당 작업에 모든 작업자를 할당합니다. 여러 작업이 가장 높은 우선 순위를 공유하면 작업자는 서로 균등하게 분할됩니다. 작업자를 균등하게 나눌 수 없는 경우 추가 작업자는 우선 순위가 가장 높은 작업 간에 분산됩니다.

여러 아티스트 또는 사용자가 동일한 우선 순위로 작업을 제출하고 각 사용자에게 즉각적인 피드백이 필요한 경우 우선 순위 밸런스를 사용합니다. 이 구성을 사용하면 단일 작업이 사용 가능한 모든 작업자를 독점하지 않으므로 제출 직후 모든 사용자에게 작업자가 할당됩니다.

작업에 작업자 공유보다 남은 작업이 적은 경우 잉여 작업자는 동일한 우선 순위 수준에서 다른 작업에 재배포됩니다. 우선 순위가 가장 높은 모든 작업이 완전히 할당되면 잉여 작업자가 다음으로 높은 우선 순위 수준의 작업으로 캐스케이드됩니다.

이 구성에는 다음 파라미터가 있습니다.

`renderingTaskBuffer`

> 작업자 고정을 제어합니다. 작업자는 렌더링 작업의 차이가 `renderingTaskBuffer` 값을 초과하는 경우에만 현재 작업에서 동일한 우선 순위의 다른 작업으로 전환합니다. 값이 높을수록 작업자가 현재 작업을 더 오래 수행할 수 있으므로 컨텍스트 전환이 줄어듭니다. 기본값은 `1`입니다.

가중치, 균형

가중치, 균형(`weightedBalanced`)은 공식을 사용하여 각 작업의 가중치를 계산합니다. Deadline Cloud는 가장 가중치가 높은 작업에 작업자를 먼저 할당합니다. 여러 작업의 가중치가 동일한 경우 작업자는 그 사이에 분산됩니다.

우선순위, 오류율 및 제출 시간이 다양한 작업 간에 작업자가 분산되는 방식을 세밀하게 제어해야 하는 경우 가중치 기반 밸런스를 사용합니다. 이 구성은 작업 우선 순위, 작업 기간, 오류 처리 및 작업자 고정성 간의 균형을 조정하려는 복잡한 렌더 팜 환경에 적합합니다.

각 작업의 가중치는 다음과 같이 계산됩니다.

```
weight = (job.Priority * priorityWeight) +
         (job.Errors * errorWeight) +
         ((currentTimeInSeconds - job.SubmissionTime) * submissionTimeWeight) +
         ((job.RenderingTasks - renderingTaskBuffer) * renderingTaskWeight)
```

구성 renderingTaskBuffer 요소는 작업자가 현재 작업 중인 경우에만 적용됩니다. renderingTaskWeight는 일반적으로 음수 값으로 설정되어 할당된 작업자가 있는 작업이 더 낮은 가중치를 받아 다른 작업을 대기열의 맨 앞으로 가져옵니다. 또한 errorWeight는 일반적으로 음수이므로 오류가 있는 작업의 우선 순위가 해제됩니다. 최소 및 최대 우선 순위 작업에 일정 재정의를 사용할 수 있습니다.

이 구성에는 다음과 같은 파라미터가 있습니다.

priorityWeight

작업의 우선 순위에 적용되는 가중치입니다. 양수 값은 우선 순위가 높은 작업이 먼저 예약됨을 의미합니다. 기본값은 100.0입니다. 범위: 0 ~. 10000

errorWeight

작업의 오류 수에 적용되는 가중치입니다. 음수 값은 오류가 없는 작업이 먼저 예약됨을 의미합니다. 기본값은 -10.0입니다. 범위: -10000 ~. 10000

submissionTimeWeight

작업의 제출 시간(초)에 적용되는 가중치입니다. 양수 값은 이전에 제출된 작업이 먼저 예약됨을 의미합니다. 기본값은 3.0입니다. 범위: 0 ~. 10000

renderingTaskWeight

작업에 대해 현재 렌더링 중인 작업 수에 적용되는 가중치입니다. 음수 값은 작업자 수가 적은 작업이 다음에 예약됨을 의미합니다. 기본값은 -100.0입니다. 범위: -10000 ~. 10000

renderingTaskBuffer

렌더링 작업 가중치가 적용되기 전의 렌더링 작업 수입니다. 양수 값은 작업자의 현재 작업을 유지합니다. 기본값은 1입니다. 범위: 0 ~. 1000

maxPriorityOverride

선택 사항. 로 설정하면 가중 공식에 관계없이 최대 우선 순위(100)인 alwaysScheduleFirst작업이 항상 다른 작업보다 먼저 예약됩니다. 여러 작업에 최대 우선 순위가 있는 경우 표준 가중치 공

식을 사용하여 타이가 끊어집니다. 재정의가 없는 경우 최대 우선 순위 작업은 특별한 처리 없이 표준 가중 공식을 사용합니다.

`minPriorityOverride`

선택 사항. 로 설정하면 가중 공식에 관계없이 최소 우선 순위(0)인 `alwaysScheduleLast`작업은 항상 다른 작업 이후에 예약됩니다. 여러 작업에 최소 우선 순위가 있는 경우 표준 가중 공식을 사용하여 타이가 끊어집니다. 재정의가 없는 경우 최소 우선 순위 작업은 특별한 처리 없이 표준 가중치 공식을 사용합니다.

플릿 호환성 확인

작업을 생성한 후 Deadline Cloud는 작업이 제출된 대기열과 연결된 플릿의 기능과 비교하여 작업의 각 단계에 대한 호스트 요구 사항을 확인합니다. 플릿이 호스트 요구 사항을 충족하는 경우 작업이 `READY` 상태로 전환됩니다.

작업의 단계에 대기열과 연결된 플릿이 충족할 수 없는 요구 사항이 있는 경우 단계의 상태가 로 설정됩니다`NOT_COMPATIBLE`. 또한 작업의 나머지 단계는 취소됩니다.

플릿에 대한 기능은 플릿 수준에서 설정됩니다. 플릿의 작업자가 작업의 요구 사항을 충족하더라도 플릿이 작업의 요구 사항을 충족하지 않으면 작업에서 작업이 할당되지 않습니다.

다음 작업 템플릿에는 단계의 호스트 요구 사항을 지정하는 단계가 있습니다.

```
name: Sample Job With Host Requirements
specificationVersion: jobtemplate-2023-09
steps:
- name: Step 1
  script:
    actions:
      onRun:
        args:
        - '1'
        command: /usr/bin/sleep
  hostRequirements:
    amounts:
    # Capabilities starting with "amount." are amount capabilities. If they start with
 "amount.worker.",
    # they are defined by the OpenJD specification. Other names are free for custom
 usage.
    - name: amount.worker.vcpu
```

```
      min: 4
      max: 8
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

이 작업은 다음 기능을 갖춘 플릿에 예약할 수 있습니다.

```
{
    "vCpuCount": {"min": 4, "max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
```

다음 기능이 있는 플릿에는이 작업을 예약할 수 없습니다.

```
{
    "vCpuCount": {"min": 4},
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
    The vCpuCount has no maximum, so it exceeds the maximum vCPU host requirement.

{
    "vCpuCount": {"max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
}
    The vCpuCount has no minimum, so it doesn't satisfy the minimum vCPU host
 requirement.

{
    "vCpuCount": {"min": 4, "max": 8},
    "memoryMiB": {"min": 1024},
    "osFamily": "windows",
    "cpuArchitectureType": "x86_64"
}
    The osFamily doesn't match.
```

플릿 조정

호환되는 서비스 관리형 플릿에 작업이 할당되면 플릿이 자동으로 조정됩니다. 플릿의 작업자 수는 플릿이 실행할 수 있는 작업 수에 따라 변경됩니다.

작업이 고객 관리형 플릿에 할당되면 작업자가 이미 존재하거나 이벤트 기반 Auto Scaling을 사용하여 생성할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 EventBridge를 사용하여 Auto Scaling 이벤트 처리를 참조하세요. Amazon EC2 Auto Scaling

세션

작업의 작업은 하나 이상의 세션으로 나뉩니다. 작업자는 세션을 실행하여 환경을 설정하고 작업을 실행한 다음 환경을 해제합니다. 각 세션은 작업자가 수행해야 하는 하나 이상의 작업으로 구성됩니다.

작업자가 섹션 작업을 완료하면 추가 세션 작업을 작업자에게 보낼 수 있습니다. 작업자는 세션의 기존 환경과 작업 연결을 재사용하여 작업을 보다 효율적으로 완료합니다.

서비스 관리형 플릿 작업자의 경우 세션이 종료된 후 세션 디렉터리가 삭제되지만 다른 디렉터리는 세션 간에 유지됩니다. 이 동작을 사용하면 여러 세션에서 재사용할 수 있는 데이터에 대한 캐싱 전략을 구현할 수 있습니다. 세션 간에 데이터를 캐싱하려면 작업을 실행하는 사용자의 홈 디렉터리 아래에 저장합니다. 예를 들어 conda 패키지는 작업자 및 Windows 작업자의 `C:\Users\job-user\.conda-pkgs`에 있는 작업 사용자의 홈 디렉터리 `/home/job-user/.conda-pkgs` 아래에 캐시됩니다Linux. 이 데이터는 작업자가 종료될 때까지 계속 사용할 수 있습니다.

작업 첨부 파일은 Deadline Cloud CLI 작업 번들의 일부로 사용하는 제출자가 생성합니다. `create-job` AWS CLI 명령의 `--attachments` 옵션을 사용하여 작업 첨부 파일을 생성할 수도 있습니다. 환경은 두 곳, 즉 특정 대기열에 연결된 대기열 환경과 작업 템플릿에 정의된 작업 및 단계 환경으로 정의됩니다.

세션 작업 유형에는 네 가지가 있습니다.

- `syncInputJobAttachments` - 입력 작업 첨부 파일을 작업자에게 다운로드합니다.
- `envEnter` - 환경에 대한 `onEnter` 작업을 수행합니다.
- `taskRun` - 작업에 대한 `onRun` 작업을 수행합니다.
- `envExit` - 환경에 대한 `onExit` 작업을 수행합니다.

다음 작업 템플릿에는 단계 환경이 있습니다. 단계 환경을 설정하는 `onEnter` 정의, 실행할 작업을 정의하는 `onRun` 정의, 단계 환경을 제거하는 `onExit` 정의가 있습니다. 이 작업에 대해 생성된 세션에는 `envEnter` 작업, 하나 이상의 `taskRun` 작업, `envExit` 작업이 포함됩니다.

```
name: Sample Job with Maya Environment
specificationVersion: jobtemplate-2023-09
steps:
- name: Maya Step
  stepEnvironments:
  - name: Maya
    description: Runs Maya in the background.
    script:
      embeddedFiles:
      - name: initData
        filename: init-data.yaml
        type: TEXT
        data: |
          scene_file: MyAwesomeSceneFile
          renderer: arnold
          camera: persp
      actions:
        onEnter:
          command: MayaAdaptor
          args:
          - daemon
          - start
          - --init-data
          - file://{{Env.File.initData}}
        onExit:
          command: MayaAdaptor
          args:
          - daemon
          - stop
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      range: 1-5
      type: INT
  script:
    embeddedFiles:
    - name: runData
      filename: run-data.yaml
      type: TEXT
      data: |
        frame: {{Task.Param.Frame}}
    actions:
      onRun:
```

```
        command: MayaAdaptor
        args:
        - daemon
        - run
        - --run-data
        - file://{{ Task.File.runData }}
```

세션 작업 파이프라이닝

세션 작업 파이프라이닝을 사용하면 스케줄러가 작업자에게 여러 세션 작업을 사전 할당할 수 있습니다. 그런 다음 작업자는 이러한 작업을 순차적으로 실행하여 작업 간 유휴 시간을 줄이거나 제거할 수 있습니다.

초기 할당을 생성하기 위해 스케줄러는 하나의 작업으로 세션을 생성하고, 작업자는 작업을 완료한 다음, 스케줄러는 작업 기간을 분석하여 향후 할당을 결정합니다.

스케줄러가 유효하려면 작업 기간 규칙이 있습니다. 1분 미만의 작업의 경우 스케줄러는 Power-of-2 성장 패턴을 사용합니다. 예를 들어 1초 작업의 경우 스케줄러는 2개의 새 작업을 할당한 다음 4개, 8개를 할당합니다. 1분 이상 태스크의 경우 스케줄러는 새 태스크를 하나만 할당하고 파이프라이닝은 비활성화된 상태로 유지됩니다.

파이프라인 크기를 계산하기 위해 스케줄러는 다음을 수행합니다.

- 완료된 작업의 평균 작업 기간을 사용합니다.
- 작업자를 1분 동안 바쁘게 유지하는 것을 목표로 합니다.
- 동일한 세션 내의 작업만 고려합니다.
- 작업자 간에 기간 데이터를 공유하지 않음

세션 작업 피플라이닝을 사용하면 작업자가 즉시 새 작업을 시작하고 스케줄러 요청 사이에 대기 시간이 없습니다. 또한 장기 실행 프로세스를 위해 작업자 효율성과 작업 분산을 개선합니다.

또한 우선순위가 더 높은 새 작업이 있는 경우 작업자는 현재 세션이 종료되고 우선순위가 더 높은 작업의 새 세션이 할당되기 전에 이전에 할당된 모든 작업을 완료합니다.

단계 종속성

Deadline Cloud는 한 단계가 시작하기 전에 다른 단계가 완료될 때까지 대기하도록 단계 간 종속성 정의를 지원합니다. 한 단계에 대해 둘 이상의 종속성을 정의할 수 있습니다. 종속성이 있는 단계는 모든 종속성이 완료될 때까지 예약되지 않습니다.

작업 템플릿이 순환 종속성을 정의하면 작업이 거부되고 작업 상태가 로 설정됩니다CREATE_FAILED.

다음 작업 템플릿은 두 단계로 작업을 생성합니다.는에 StepB 따라 달라집니다StepA.는가 성공적으로 StepA 완료된 후에StepB만 실행됩니다.

작업이 생성된 후는 StepA READY 상태가 되고 StepB는 PENDING 상태가 됩니다. 가 StepA 완료되면 READY 상태로 StepB 이동합니다. 가 StepA 실패하거나 StepA가 취소되면 CANCELED 상태가 StepB 됩니다.

여러 단계에 종속성을 설정할 수 있습니다. 예를 들어 StepC가 StepA 및에 모두 의존하는 경우 StepBStepC는 다른 두 단계가 완료될 때까지 시작되지 않습니다.

단계 종속성에는 다음과 같은 제한이 있습니다.

- 단계당 종속성 - 단계는 최대 128개의 다른 단계에 따라 달라질 수 있습니다.
- 단계당 소비자 - 최대 32개의 다른 단계가 단일 단계에 따라 달라질 수 있습니다.

```
name: Step-Step Dependency Test
specificationVersion: 'jobtemplate-2023-09'
steps:
- name: A
  script:
    actions:
      onRun:
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
        data: |
          #!/bin/env bash

          set -euo pipefail

          sleep 1
          echo Task A Done!
- name: B
  dependencies:
  - dependsOn: A # This means Step B depends on Step A
  script:
    actions:
      onRun:
```

```
        command: bash
        args: ['{{ Task.File.run }}']
    embeddedFiles:
      - name: run
        type: TEXT
        data: |
          #!/bin/env bash

          set -euo pipefail

          sleep 1
          echo Task B Done!
```

Deadline Cloud에서 작업 수정

다음 AWS Command Line Interface (AWS CLI) update 명령을 사용하여 작업 구성을 수정하거나 작업, 단계 또는 작업의 대상 상태를 설정할 수 있습니다.

- aws deadline update-job
- aws deadline update-step
- aws deadline update-task

다음 update 명령 예제에서 각각을 사용자 고유의 정보로 바꿉*user input placeholder*니다.

Example- 작업 다시 대기열에 추가

단계 종속성이 없는 한 작업의 모든 작업은 READY 상태로 전환됩니다. 종속성이 있는 단계는 복원될 PENDING 때 READY 또는 중 하나로 전환됩니다.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status PENDING
```

Example- 작업 취소

상태가 SUCCEEDED 없거나 로 FAILED 표시된 작업의 모든 작업CANCELED.

```
aws deadline update-job \
```

```
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status CANCELED
```

Example- 작업 실패 표시

작업에서 상태가 인 모든 작업은 SUCCEEDED 변경되지 않습니다. 다른 모든 작업은 로 표시됩니다FAILED.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status FAILED
```

Example- 작업 성공 표시

작업의 모든 작업이 SUCCEEDED 상태로 이동합니다.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status SUCCEEDED
```

Example- 작업 일시 중지

SUCCEEDED, CANCELED또는 FAILED 상태의 작업 작업은 변경되지 않습니다. 다른 모든 작업은 로 표시됩니다SUSPENDED.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--target-task-run-status SUSPENDED
```

Example- 작업의 우선 순위 변경

대기열에 있는 작업의 우선 순위를 업데이트하여 예약된 순서를 변경합니다. 우선 순위가 높은 작업이 일반적으로 먼저 예약됩니다.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--priority 100
```

Example- 허용된 실패한 작업 수 변경

나머지 작업이 취소되기 전에 작업이 가질 수 있는 최대 실패 작업 수를 업데이트합니다.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-failed-tasks-count 200
```

Example- 허용되는 작업 재시도 횟수 변경

작업이 실패하기 전에 작업에 대한 최대 재시도 횟수를 업데이트합니다. 최대 재시도 횟수에 도달한 작업은이 값이 증가할 때까지 다시 대기열에 넣을 수 없습니다.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--max-retries-per-task 10
```

Example- 작업 아카이브

작업의 수명 주기 상태를 로 업데이트합니다ARCHIVED. 보관된 작업은 예약하거나 수정할 수 없습니다. FAILED, CANCELEDSUCCEEDED, 또는 SUSPENDED 상태의 작업만 아카이브할 수 있습니다.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--lifecycle-status ARCHIVED
```

Example- 작업 이름 변경

작업의 표시 이름을 업데이트합니다. 작업 이름은 최대 128자까지 가능합니다.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--name "New Job Name"
```

Example- 작업 설명 변경

작업에 대한 설명을 업데이트합니다. 설명은 최대 2,048자까지 가능합니다. 기존 설명을 제거하려면 빈 문자열을 전달합니다.

```
aws deadline update-job \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--description "New Job Description"
```

Example- 단계 다시 대기열에 추가

단계 종속성이 없는 한 단계의 모든 작업은 READY 상태로 전환됩니다. 종속성이 있는 단계의 작업은 READY 또는 중 하나로 전환PENDING되고 작업이 복원됩니다.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status PENDING
```

Example- 단계 취소

상태가 SUCCEEDED 없거나 로 표시된 단계의 모든 작업FAILED입니다CANCELED.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status CANCELED
```

Example- 실패한 단계 표시

상태가 인 단계의 모든 작업은 SUCCEEDED 변경되지 않습니다. 다른 모든 작업은 로 표시됩니다FAILED.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status FAILED
```

Example- 단계 성공 표시

단계의 모든 작업은 로 표시됩니다SUCCEEDED.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUCCEEDED
```

Example- 단계 일시 중지

SUCCEEDED, CANCELED또는 FAILED 상태의 단계에서 작업은 변경되지 않습니다. 다른 모든 작업은 로 표시됩니다SUSPENDED.

```
aws deadline update-step \
--farm-id farmID \
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--target-task-run-status SUSPENDED
```

Example- 작업 상태 변경

update-task Deadline Cloud CLI 명령을 사용하면 작업이 지정된 상태로 전환됩니다.

```
aws deadline update-task \
--farm-id farmID \
```

```
--queue-id queueID \
--job-id jobID \
--step-id stepID \
--task-id taskID \
--target-task-run-status SUCCEEDED | SUSPENDED | CANCELED | FAILED | PENDING
```

Deadline Cloud 고객 관리형 플릿 생성 및 사용

고객 관리형 플릿(CMF)을 생성하면 처리 파이프라인을 완전히 제어할 수 있습니다. 각 작업자의 네트워크 및 소프트웨어 환경을 정의합니다. Deadline Cloud는 작업의 리포지토리 및 스케줄러 역할을 합니다.

작업자는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스, 코로케이션 시설의 작업자 또는 온프레미스 작업자일 수 있습니다. 각 작업자는 Deadline Cloud 작업자 에이전트를 실행해야 합니다. 모든 작업자는 Deadline Cloud 서비스 엔드포인트에 액세스할 수 있어야 합니다.

다음 주제에서는 Amazon EC2 인스턴스를 사용하여 기본 CMF를 생성하는 방법을 보여줍니다.

주제

고객 관리형 플릿 생성

고객 관리형 플릿(CMF)을 생성하려면 다음 단계를 완료합니다.

Deadline Cloud console

Deadline Cloud 콘솔을 사용하여 고객 관리형 플릿을 생성하려면

1. Deadline Cloud 콘솔을 엽니다.
2. 팜을 선택합니다. 사용 가능한 팜 목록이 표시됩니다.
3. 작업할 팜의 이름을 선택합니다.

4. 플릿 탭을 선택한 다음 플릿 생성을 선택합니다.
5. 플릿의 이름을 입력합니다.
6. (선택 사항) 플릿에 대한 설명을 입력합니다.
7. 플릿 유형에 대해 고객 관리형을 선택합니다.
8. 플릿의 서비스 액세스를 선택합니다.

   a. 보다 세분화된 권한 제어를 위해 각 플릿에 새 서비스 역할 생성 및 사용 옵션을 사용하는 것이 좋습니다. 이 옵션은 기본적으로 설정되어 있습니다.

   b. 서비스 역할 선택을 선택하여 기존 서비스 역할을 사용할 수도 있습니다.
9. 선택 사항을 검토한 후 다음을 선택합니다.
10. 플릿의 운영 체제를 선택합니다. 플릿의 모든 작업자는 공통 운영 체제를 가지고 있어야 합니다.
11. 호스트 CPU 아키텍처를 선택합니다.
12. 플릿의 워크로드 요구 사항을 충족하기 위해 최소 및 최대 vCPU 및 메모리 하드웨어 기능을 선택합니다.
13. Auto Scaling 유형을 선택합니다. 자세한 내용은 EventBridge를 사용하여 Auto Scaling 이벤트 처리를 참조하세요.

    - 규모 조정 없음: 온프레미스 플릿을 생성하고 있으며 Deadline Cloud Auto Scaling을 옵트아웃하려고 합니다.
    - 조정 권장 사항: Amazon Elastic Compute Cloud(Amazon EC2) 플릿을 생성하고 있습니다.
14. (선택 사항) 화살표를 선택하여 기능 추가 섹션을 확장합니다.
15. (선택 사항) GPU 기능 추가 - 선택 사항 확인란을 선택한 다음 최소 및 최대 GPUs와 메모리를 입력합니다.
16. 선택 사항을 검토한 후 다음을 선택합니다.
17. (선택 사항) 사용자 지정 작업자 기능을 정의한 후 다음을 선택합니다.
18. 드롭다운을 사용하여 플릿과 연결할 대기열을 하나 이상 선택합니다.

    > **Note**
    >
    > 플릿은 모두 동일한 신뢰 경계에 있는 대기열에만 연결하는 것이 좋습니다. 이 권장 사항은 동일한 작업자에서 작업을 실행하는 간의 강력한 보안 경계를 보장합니다.
19. 대기열 연결을 검토한 후 다음을 선택합니다.

20. (선택 사항) 기본 Conda 대기열 환경의 경우 작업에서 요청한 conda 패키지를 설치하는 대기열 환경을 생성합니다.

> **Note**
>
> conda 대기열 환경은 작업에서 요청한 conda 패키지를 설치하는 데 사용됩니다. 일반적으로 CMFs에는 기본적으로 필요한 conda 명령이 설치되지 않으므로 CMFs와 연결된 대기열에서 conda 대기열 환경을 선택 취소해야 합니다.

21. (선택 사항) CMF에 태그를 추가합니다. 자세한 내용은 AWS 리소스 태그 지정을 참조하세요.
22. 플릿 구성을 검토하고 변경한 다음 플릿 생성을 선택합니다.
23. 플릿 탭을 선택한 다음 플릿 ID를 기록해 둡니다.

AWS CLI

AWS CLI 를 사용하여 고객 관리형 플릿을 생성하려면

1. 터미널을 엽니다.
2. 새 편집기`fleet-trust-policy.json`에서를 생성합니다.

   a. 다음 IAM 정책을 추가하여 *`ITALICIZED`* 텍스트를 계정 AWS ID 및 Deadline Cloud 팜 ID로 바꿉니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
```

```
                    "aws:SourceArn":
 "arn:aws:deadline:*:111122223333:farm/FARM_ID"
                }
            }
        }
    ]
}
```

b. 변경 내용을 저장합니다.

3. `fleet-policy.json` 생성.

a. 다음 IAM 정책을 추가합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "deadline:AssumeFleetRoleForWorker",
                "deadline:UpdateWorker",
                "deadline:DeleteWorker",
                "deadline:UpdateWorkerSchedule",
                "deadline:BatchGetJobEntity",
                "deadline:AssumeQueueRoleForWorker"
            ],
            "Resource": "arn:aws:deadline:*:111122223333:*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream"
            ],
            "Resource": "arn:aws:logs:*:*:*://deadline/*",
            "Condition": {
```

```
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:GetLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:*:/aws/deadline/*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalAccount": "${aws:ResourceAccount}"
                }
            }
        }
    ]
}
```

b. 변경 내용을 저장합니다.

4. 플릿의 작업자가 사용할 IAM 역할을 추가합니다.

```
aws iam create-role --role-name FleetWorkerRoleName --assume-role-policy-
document file://fleet-trust-policy.json
aws iam put-role-policy --role-name FleetWorkerRoleName --policy-name
 FleetWorkerPolicy --policy-document file://fleet-policy.json
```

5. `create-fleet-request.json` 생성.

a. 다음 IAM 정책을 추가하여 ITALICIZED 텍스트를 CMF 값으로 바꿉니다.

> **Note**
>
> 에서 *ROLE_ARN*을 찾을 수 있습니다`create-cmf-fleet.json`.
> *OS_FAMILY*의 경우 `linux` `macos` 또는 중 하나를 선택해야 합니다`windows`.

```
{
    "farmId": "FARM_ID",
    "displayName": "FLEET_NAME",
```

```
    "description": "FLEET_DESCRIPTION",
    "roleArn": "ROLE_ARN",
    "minWorkerCount": 0,
    "maxWorkerCount": 10,
    "configuration": {
        "customerManaged": {
            "mode": "NO_SCALING",
            "workerCapabilities": {
                "vCpuCount": {
                    "min": 1,
                    "max": 4
                },
                "memoryMiB": {
                    "min": 1024,
                    "max": 4096
                },
                "osFamily": "OS_FAMILY",
                "cpuArchitectureType": "x86_64",
            },
        },
    }
}
```

b. 변경 내용을 저장합니다.

6. 플릿을 생성합니다.

```
aws deadline create-fleet --cli-input-json file://create-fleet-request.json
```

작업자 호스트 설정 및 구성

작업자 호스트는 Deadline Cloud 작업자를 실행하는 호스트 시스템을 말합니다. 이 섹션에서는 작업자 호스트를 설정하고 특정 요구 사항에 맞게 구성하는 방법을 설명합니다. 각 작업자 호스트는 작업자 에이전트라는 프로그램을 실행합니다. 작업자 에이전트는 다음을 담당합니다.

- 작업자 수명 주기 관리.
- 할당된 작업, 진행 상황 및 결과를 동기화합니다.
- 실행 중인 작업 모니터링.
- 구성된 대상으로 로그 전달.

제공된 Deadline Cloud 작업자 에이전트를 사용하는 것이 좋습니다. 작업자 에이전트는 오픈 소스이며 기능 요청을 권장하지만 필요에 맞게 개발하고 사용자 지정할 수도 있습니다.

다음 섹션의 작업을 완료하려면 다음이 필요합니다.

Linux

- Linux기반 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스입니다. Amazon Linux 2023을 사용하는 것이 좋습니다.
- sudo 권한
- Python 3.9 이상

Windows

- Windows기반 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스입니다. Windows Server 2022을 권장합니다.
- 작업자 호스트에 대한 관리자 액세스
- 모든 사용자에 대해 설치된 Python 3.9 이상

Python 가상 환경 생성 및 구성

Python 3.9 이상을 설치하고에 배치한 Linux 경우에서 Python 가상 환경을 생성할 수 있습니다PATH.

> **Note**
>
> 에서는 Windows에이전트 파일을 Python의 글로벌 사이트 패키지 디렉터리에 설치해야 합니다. Python 가상 환경은 현재 지원되지 않습니다.

Python 가상 환경을 생성하고 활성화하려면

1. 터미널을 root 사용자로 엽니다(또는 sudo / 사용su).
2. Python 가상 환경을 생성하고 활성화합니다.

```
python3 -m venv /opt/deadline/worker
```

```
source /opt/deadline/worker/bin/activate
pip install --upgrade pip
```

Deadline Cloud 작업자 에이전트 설치

Python을 설정하고에서 가상 환경을 생성한 후 Deadline Cloud 작업자 에이전트 Python 패키지를 Linux설치합니다.

작업자 에이전트 Python 패키지를 설치하려면

Linux

1. 터미널을 `root` 사용자로 엽니다(또는 `sudo` / 사용`su`).
2. PyPI에서 Deadline Cloud 작업자 에이전트 패키지를 다운로드하여 설치합니다.

```
/opt/deadline/worker/bin/python -m pip install deadline-cloud-worker-agent
```

Windows

1. 관리자 명령 프롬프트 또는 PowerShell 터미널을 엽니다.
2. PyPI에서 Deadline Cloud 작업자 에이전트 패키지를 다운로드하여 설치합니다.

```
python -m pip install deadline-cloud-worker-agent
```

Windows 작업자 호스트에 긴 경로 이름(250자 이상)이 필요한 경우 다음과 같이 긴 경로 이름을 활성화해야 합니다.

Windows 작업자 호스트의 긴 경로를 활성화하려면

1. 긴 경로 레지스트리 키가 활성화되어 있는지 확인합니다. 자세한 내용은 Microsoft 웹 사이트에서 로그 경로를 활성화하는 레지스트리 설정을 참조하세요.
2. 데스크톱용 Windows SDK C++ x86 앱을 설치합니다. 자세한 내용은 Windows 개발 센터의 Windows SDK를 참조하세요.
3. 작업자 에이전트가 설치된 환경에서 Python 설치 위치를 엽니다. 기본값은 `C:\Program Files\Python311`입니다. 라는 실행 파일이 있습니다`pythonservice.exe`.

4. 동일한 pythonservice.exe.manifest 위치에 라는 새 파일을 생성합니다. 다음을 추가합니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity type="win32" name="pythonservice" processorArchitecture="x86"
 version="1.0.0.0"/>
  <application xmlns="urn:schemas-microsoft-com:asm.v3">
    <windowsSettings>
      <longPathAware xmlns="http://schemas.microsoft.com/SMI/2016/
WindowsSettings">true</longPathAware>
    </windowsSettings>
  </application>
</assembly>
```

5. 명령 프롬프트를 열고 생성한 매니페스트 파일의 위치에서 다음 명령을 실행합니다.

```
"C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x86\mt.exe" -manifest
 pythonservice.exe.manifest -outputresource:pythonservice.exe;#1
```

다음과 유사한 출력 화면이 표시되어야 합니다.

```
Microsoft (R) Manifest Tool
Copyright (c) Microsoft Corporation.
All rights reserved.
```

이제 작업자는 긴 경로에 액세스할 수 있습니다. 정리하려면 pythonservice.exe.manifest 파일을 제거하고 SDK를 제거합니다.

Deadline Cloud 작업자 에이전트 구성

Deadline Cloud 작업자 에이전트 설정은 세 가지 방법으로 구성할 수 있습니다. install-deadline-worker 도구를 실행하여 운영 체제 설정을 사용하는 것이 좋습니다.

작업자 에이전트는 Windows에서 도메인 사용자로 실행을 지원하지 않습니다. 도메인 사용자로 작업을 실행하려면 작업 실행을 위해 대기열 사용자를 구성할 때 도메인 사용자 계정을 지정할 수 있습니다. 자세한 내용은 Deadline Cloud 사용 설명서의 Deadline Cloud 대기열의 7단계를 참조하세요. AWS

명령줄 인수 - 명령줄에서 Deadline Cloud 작업자 에이전트를 실행할 때 인수를 지정할 수 있습니다. 일부 구성 설정은 명령줄 인수를 통해 사용할 수 없습니다. 사용 가능한 모든 명령줄 인수를 보려면를 입력합니다`deadline-worker-agent --help`.

환경 변수 - 로 시작하는 환경 변수를 설정하여 Deadline Cloud 작업자 에이전트를 구성할 수 있습니다`DEADLINE_WORKER_`. 예를 들어 사용 가능한 모든 명령줄 인수를 보려면 작업자 에이전트의 출력을 상세하게 `export DEADLINE_WORKER_VERBOSE=true` 설정할 수 있습니다. 자세한 예제 및 정보는 Linux 또는 `/etc/amazon/deadline/worker.toml.example`의 섹션을 참조`C:\ProgramData\Amazon\Deadline\Config\worker.toml.example`하세요Windows.

구성 파일 - 작업자 에이전트를 설치하면 Linux 또는의 `C:\ProgramData\Amazon\Deadline\Config\worker.toml` `/etc/amazon/deadline/worker.toml`에 구성 파일이 생성됩니다 Windows. 작업자 에이전트는 시작 시이 구성 파일을 로드합니다. 예제 구성 파일(`/etc/amazon/deadline/worker.toml.example` Linux 또는 `C:\ProgramData\Amazon\Deadline\Config\worker.toml.example`의 Windows)을 사용하여 특정 요구 사항에 맞게 기본 작업자 에이전트 구성 파일을 조정할 수 있습니다.

마지막으로 소프트웨어를 배포하고 예상대로 작동한 후 작업자 에이전트에 대해 자동 종료를 활성화하는 것이 좋습니다. 이렇게 하면 필요할 때 작업자 플릿을 확장하고 작업이 완료되면 종료할 수 있습니다. Auto Scaling을 사용하면 필요한 리소스만 사용할 수 있습니다. Auto Scaling 그룹에서 시작한 인스턴스를 종료하려면 `worker.toml` 구성 파일에 `shutdown_on_stop=true`를 추가해야 합니다.

자동 종료를 활성화하려면

`root` 사용자로서:

- 파라미터를 사용하여 작업자 에이전트를 설치합니다**`--allow-shutdown`**.

  Linux

  입력:

  ```
  /opt/deadline/worker/bin/install-deadline-worker \
    --farm-id FARM_ID \
    --fleet-id FLEET_ID \
    --region REGION \
    --allow-shutdown
  ```

  Windows

  입력:

```
install-deadline-worker ^
  --farm-id FARM_ID ^
  --fleet-id FLEET_ID ^
  --region REGION ^
  --allow-shutdown
```

작업 사용자 및 그룹 생성

이 섹션에서는 에이전트 사용자와 대기열에 jobRunAsUser 정의된 간의 필수 사용자 및 그룹 관계를 설명합니다.

Deadline Cloud 작업자 에이전트는 호스트에서 전용 에이전트별 사용자로 실행되어야 합니다. 작업자가 대기열 작업을 특정 운영 체제 사용자 및 그룹으로 실행하도록 Deadline Cloud 대기열의 jobRunAsUser 속성을 구성해야 합니다. 이 구성은 작업에 있는 공유 파일 시스템 권한을 제어할 수 있음을 의미합니다. 또한 작업과 작업자 에이전트 사용자 간의 중요한 보안 경계 역할을 합니다.

Linux 작업 사용자 및 그룹

로컬 작업자 에이전트 사용자 및를 설정하려면 다음 요구 사항을 충족해야 jobRunAsUser합니다. Active Directory 또는 LDAP와 같은 Linux 플러그형 인증 모듈(PAM)을 사용하는 경우 프로시저가 다를 수 있습니다.

작업자 에이전트를 설치할 때 작업자 에이전트 사용자와 공유 jobRunAsUser 그룹이 설정됩니다. 기본값은 deadline-worker-agent 및 deadline-job-users이지만 작업자 에이전트를 설치할 때 변경할 수 있습니다.

```
install-deadline-worker \
    --user AGENT_USER_NAME \
    --group JOB_USERS_GROUP
```

명령은 루트 사용자로 실행해야 합니다.

- 각 에는 일치하는 기본 그룹이 있어야 jobRunAsUser 합니다. adduser 명령을 사용하여 사용자를 생성하면 일반적으로 일치하는 기본 그룹이 생성됩니다.

  ```
  adduser -r -m jobRunAsUser
  ```

- 의 기본 그룹은 작업자 에이전트 사용자의 보조 그룹jobRunAsUser입니다. 공유 그룹을 사용하면 작업자 에이전트가 실행 중인 작업에 파일을 사용할 수 있도록 할 수 있습니다.

```
usermod -a -G jobRunAsUser deadline-worker-agent
```

- 는 공유 작업 그룹의 구성원이어야 jobRunAsUser 합니다.

```
usermod -a -G deadline-job-users jobRunAsUser
```

- 는 작업자 에이전트 사용자의 기본 그룹에 속해서는 jobRunAsUser 안 됩니다. 작업자 에이전트가 작성한 민감한 파일은 에이전트의 기본 그룹이 소유합니다. 이 그룹의 일부jobRunAsUser인 경우 작업자에서 실행 중인 작업에서 작업자 에이전트 파일에 액세스할 수 있습니다.
- 기본값은 작업자가 속한 팜의 리전과 일치해야 AWS 리전 합니다. 이는 작업자의 모든 jobRunAsUser 계정에 적용해야 합니다.

```
sudo -u jobRunAsUser aws configure set default.region aws-region
```

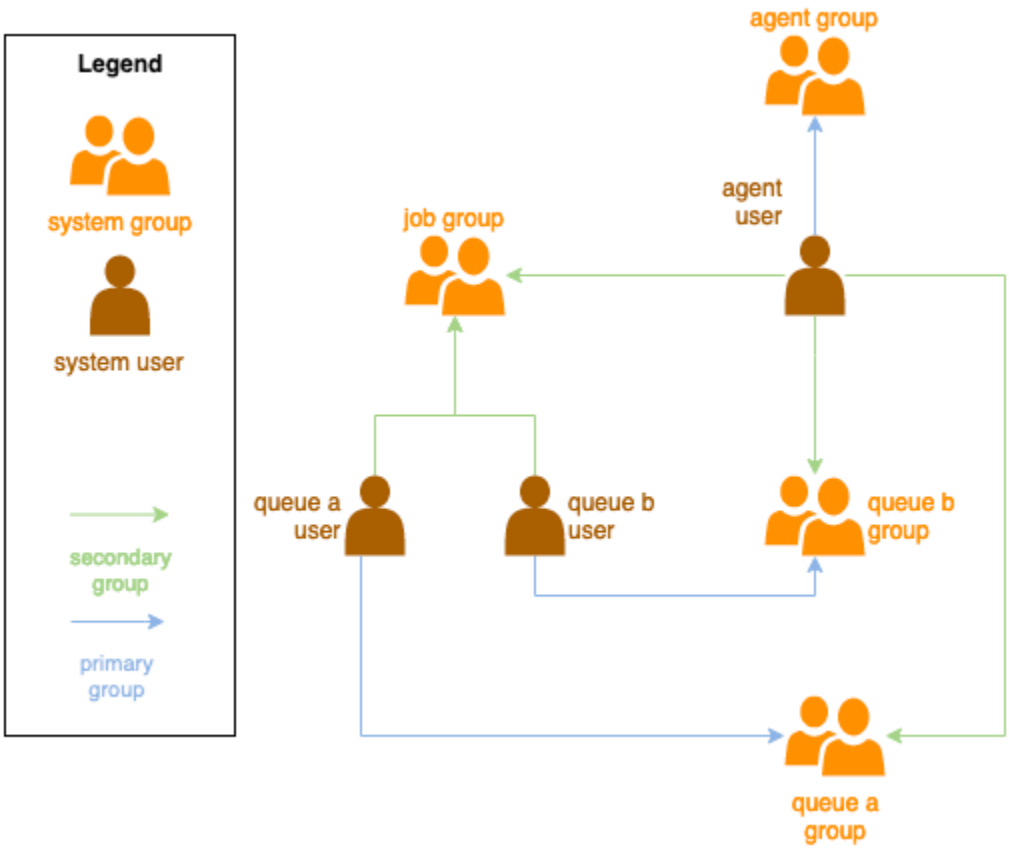
- 작업자 에이전트 사용자는 sudo 명령을 로 실행할 수 있어야 합니다jobRunAsUser. 다음 명령을 실행하여 편집기를 열어 새 sudoers 규칙을 생성합니다.

```
visudo -f /etc/sudoers.d/deadline-worker-job-user
```

파일에 다음을 추가합니다.

```
# Allows the Deadline Cloud worker agent OS user to run commands
 # as the queue OS user without requiring a password.
deadline-worker-agent ALL=(jobRunAsUser) NOPASSWD:ALL
```

다음 다이어그램은 에이전트 사용자와 플릿과 연결된 대기열의 jobRunAsUser 사용자 및 그룹 간의 관계를 보여줍니다.



Windows 사용자

Windows 사용자를 로 사용하려면 다음 요구 사항을 충족해야 `jobRunAsUser`합니다.

- 모든 대기열 `jobRunAsUser` 사용자가 있어야 합니다.
- 암호는 대기열의 `JobRunAsUser` 필드에 지정된 보안 암호 값과 일치해야 합니다. 지침은 Deadline Cloud 사용 설명서의 Deadline Cloud 대기열의 7단계를 참조하세요. AWS
- 에이전트-사용자는 해당 사용자로 로그온할 수 있어야 합니다.

작업자 호스트 보안

작업자 호스트를 설정할 때 보안 모범 사례를 따라 민감한 정보를 보호하고 적절한 액세스 제어를 유지하세요.

로그 폴더 권한 구성

작업자 에이전트는 호스트 구성 스크립트 및 작업 실행의 민감한 정보가 포함될 수 있는 로그 파일을 작성합니다. `install-deadline-worker` 명령은 보안 권한이 있는 로그 디렉터리를 생성합니다. 설

치 전에 디렉터리를 수동으로 생성해야 하는 경우 다음 절차를 사용하여 서비스 관리형 플릿에서 사용하는 권한과 일치시킵니다.

Linux

에서 로그 디렉터리 권한을 구성하려면 Linux

1. 로그 디렉터리를 생성합니다.

```
sudo mkdir -p /var/log/amazon/deadline
```

2. 소유자와 그룹을 작업자 에이전트 사용자로 설정합니다.

```
sudo chown -R deadline-worker:deadline-worker /var/log/amazon/deadline
```

3. 권한을 750으로 설정합니다.

```
sudo chmod -R 750 /var/log/amazon/deadline
```

이러한 권한은 작업자 에이전트 사용자 및 그룹만 로그 파일에 액세스할 수 있도록 하여 작업 사용자 및 기타 권한이 없는 사용자가 잠재적으로 민감한 정보를 읽지 못하도록 합니다.

Windows

에서 로그 디렉터리 권한을 구성하려면 Windows

1. 관리자 PowerShell 터미널을 엽니다.
2. 로그 디렉터리를 생성합니다.

```
New-Item -ItemType Directory -Force -Path "$env:PROGRAMDATA\Amazon\Deadline
\Logs"
```

3. 작업자 에이전트 사용자 및 관리자만 허용하도록 제한된 ACLs을 구성합니다.

```
$acl = Get-Acl "$env:PROGRAMDATA\Amazon\Deadline\Logs"
$acl.SetAccessRuleProtection($true, $false)
$acl.Access | ForEach-Object { $acl.RemoveAccessRule($_) }
$agentRule = New-Object
 System.Security.AccessControl.FileSystemAccessRule("deadline-worker",
 "FullControl", "ContainerInherit,ObjectInherit", "None", "Allow")
```

```
$adminRule = New-Object
 System.Security.AccessControl.FileSystemAccessRule("Administrators",
 "FullControl", "ContainerInherit,ObjectInherit", "None", "Allow")
$acl.AddAccessRule($agentRule)
$acl.AddAccessRule($adminRule)
Set-Acl "$env:PROGRAMDATA\Amazon\Deadline\Logs" $acl
```

이러한 명령은 로그 디렉터리에 대한 액세스를 작업자 에이전트 사용자 및 관리자 그룹으로만 제한하여 작업 사용자 및 기타 권한 없는 사용자가 잠재적으로 민감한 정보를 읽지 못하도록 합니다.

Windows 작업 사용자 보안 암호에 대한 액세스 관리

로 대기열을 구성할 때는 AWS Secrets Manager 보안 암호를 지정Windows`jobRunAsUser`해야 합니다. 이 보안 암호의 값은 형식의 JSON 인코딩 객체여야 합니다.

```
{
    "password": "JOB_USER_PASSWORD"
}
```

작업자가 대기열의 구성된 로 작업을 실행하려면 `jobRunAsUser`플릿의 IAM 역할에 보안 암호 값을 가져올 수 있는 권한이 있어야 합니다. 보안 암호가 고객 관리형 KMS 키를 사용하여 암호화된 경우 플릿의 IAM 역할에는 KMS 키를 사용하여 복호화할 수 있는 권한도 있어야 합니다.

이러한 보안 암호에 대한 최소 권한 원칙을 따르는 것이 좋습니다. 즉, 대기열 → `jobRunAsUser` `windows` →의 보안 암호 값을 가져올 수 있는 액세스 권한은 다음과 같아`passwordArn`야 합니다.

- 플릿과 대기열 간에 대기열-플릿 연결이 생성될 때 플릿 역할에 부여됨
- 플릿과 대기열 간에 대기열-플릿 연결이 삭제될 때 플릿 역할에서 취소됨

또한 `jobRunAsUser` 암호가 포함된 AWS Secrets Manager 보안 암호는 더 이상 사용되지 않을 때 삭제해야 합니다.

암호 보안 암호에 대한 액세스 권한 부여

Deadline Cloud 플릿은 대기열과 플릿이 연결될 때 대기열의 `jobRunAsUser` 암호 보안 암호에 저장된 암호에 액세스해야 합니다. Secrets Manager 리소스 정책을 사용하여 AWS 플릿 역할에 대한 액세

스 권한을 부여하는 것이 좋습니다. 이 지침을 엄격하게 준수하면 보안 암호에 액세스할 수 있는 플릿 역할을 더 쉽게 확인할 수 있습니다.

보안 암호에 대한 액세스 권한을 부여하려면

1. AWS 보안 암호에 대한 Secret Manager 콘솔을 엽니다.
2. 리소스 권한 섹션에서 양식의 정책 설명을 추가합니다.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:role/FleetRole"
      },
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource" : "arn:aws:secretsmanager:us-
east-1:111122223333:secret:YourSecretName-ABC123"
    }
  ]
}
```

암호 보안 암호에 대한 액세스 취소

플릿이 더 이상 대기열에 액세스할 필요가 없는 경우 대기열의 암호 보안 암호에 대한 액세스를 제거합니다`jobRunAsUser`. AWS Secrets Manager 리소스 정책을 사용하여 플릿 역할에 대한 액세스 권한을 부여하는 것이 좋습니다. 이 지침을 엄격하게 준수하면 보안 암호에 액세스할 수 있는 플릿 역할을 더 쉽게 확인할 수 있습니다.

보안 암호에 대한 액세스 권한을 취소하려면

1. AWS 보안 암호에 대한 Secret Manager 콘솔을 엽니다.
2. 리소스 권한 섹션에서 양식의 정책 설명을 제거합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:role/FleetRole"
      },
      "Action" : [
        "secretsmanager:GetSecretValue"
      ],
      "Resource" : "arn:aws:secretsmanager:us-
east-1:111122223333:secret:YourSecretName-ABC123"
    }
  ]
}
```

작업에 필요한 소프트웨어 설치 및 구성

Deadline Cloud 작업자 에이전트를 설정한 후 작업을 실행하는 데 필요한 소프트웨어로 작업자 호스트를 준비할 수 있습니다.

연결된가 있는 대기열에 작업을 제출하면 `jobRunAsUser`작업이 해당 사용자로 실행됩니다. 절대 경로가 아닌 명령을 사용하여 작업을 제출하는 경우 해당 사용자의에서 PATH 해당 명령을 찾아야 합니다.

Linux에서는 다음 중 하나에서 사용자에 PATH 대해를 지정할 수 있습니다.

- `~/.bashrc` 또는 `~/.bash_profile`
- `/etc/profile.d/*` 및와 같은 시스템 구성 파일 `/etc/profile`
- shell 시작 스크립트: `/etc/bashrc`.

Windows에서는 다음 중 하나에서 사용자에 PATH 대해를 지정할 수 있습니다.

- 사용자별 환경 변수
- 시스템 전체 환경 변수

디지털 콘텐츠 생성 도구 어댑터 설치

Deadline Cloud는 인기 있는 디지털 콘텐츠 생성(DCC) 애플리케이션을 사용하기 위한 OpenJobDescription 어댑터를 제공합니다. 고객 관리형 플릿에서 이러한 어댑터를 사용하려면 DCC 소프트웨어와 애플리케이션 어댑터를 설치해야 합니다. 그런 다음 시스템 검색 경로(예: PATH 환경 변수)에서 소프트웨어의 실행 프로그램을 사용할 수 있는지 확인합니다.

고객 관리형 플릿에 DCC 어댑터를 설치하려면

1. 터미널을 엽니다.
   a. Linux에서 `root` 사용자로 터미널을 엽니다(또는 `sudo` / 사용`su`).
   b. 에서 관리자 명령 프롬프트 또는 PowerShell 터미널을 Windows엽니다.
2. Deadline Cloud 어댑터 패키지를 설치합니다.

```
pip install deadline deadline-cloud-for-maya deadline-cloud-for-nuke deadline-
cloud-for-blender deadline-cloud-for-3ds-max
```

AWS 자격 증명 구성

작업자 수명 주기의 초기 단계는 부트스트래핑입니다. 이 단계에서 작업자 에이전트 소프트웨어는 플릿에 작업자를 생성하고 추가 작업을 위해 플릿의 역할에서 AWS 자격 증명을 가져옵니다.

AWS credentials for Amazon EC2

Deadline Cloud 작업자 호스트 권한을 사용하여 Amazon EC2에 대한 IAM 역할을 생성하려면

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
2. 탐색 창에서 역할을 선택한 다음 역할 생성을 선택합니다.
3. AWS 서비스를 선택합니다.
4. 서비스 또는 사용 사례로 EC2를 선택한 후 다음을 선택합니다.
5. 필요한 권한을 부여하려면 `AWSDeadlineCloud-WorkerHost` AWS 관리형 정책을 연결합니다.

On-premises AWS credentials

온프레미스 작업자는 자격 증명을 사용하여 Deadline Cloud에 액세스합니다. 가장 안전한 액세스를 위해 IAM Roles Anywhere를 사용하여 작업자를 인증하는 것이 좋습니다. 자세한 내용은 IAM Roles Anywhere를 참조하세요.

테스트를 위해 자격 AWS 증명에 IAM 사용자 액세스 키를 사용할 수 있습니다. 제한적인 인라인 정책을 포함하여 IAM 사용자의 만료를 설정하는 것이 좋습니다.

> **Important**
>
> 다음 경고에 주의하세요.
>
> - 계정의 루트 자격 증명을 사용하여 AWS 리소스에 액세스하지 마십시오. 이 보안 인증은 계정 액세스에 제한이 없고 취소하기 어렵습니다.
> - 금지 사항. 애플리케이션 파일에 리터럴 액세스 키나 보안 인증 정보를 넣으면 안 됩니다. 이를 어기는 경우, 예를 들어 프로젝트를 퍼블릭 리포지토리에 업로드하면 뜻하지 않게 보안 인증이 노출될 위험이 있습니다.
> - 금지 사항. 프로젝트 영역에 보안 인증이 포함된 파일을 포함하지 마세요.
> - 액세스 키를 보호합니다. 계정 식별자를 찾는 데 도움이 되더라도 액세스 키를 권한 없는 당사자에게 제공하지 마세요. 이렇게 하면 다른 사람에게 계정에 대한 영구 액세스 권한을 부여할 수 있습니다.
> - 공유 AWS 자격 증명 파일에 저장된 모든 자격 증명은 일반 텍스트로 저장됩니다.

자세한 내용은 AWS 일반 참조의 AWS 액세스 키 관리 모범 사례를 참조하세요.

IAM 사용자를 생성합니다.

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
2. 탐색 창에서 사용자를 선택하고 사용자 생성을 선택합니다.
3. 사용자 이름을 지정합니다. 에 대한 사용자 액세스 제공 AWS Management Console 확인란의 선택을 취소한 후 다음을 선택합니다.
4. 정책 직접 연결을 선택합니다.
5. 권한 정책 목록에서 AWSDeadlineCloud-WorkerHost 정책을 선택한 후 다음을 선택합니다.
6. 사용자 세부 정보를 검토한 다음 사용자 생성을 선택합니다.

제한된 기간으로 사용자 액세스 제한

생성하는 모든 IAM 사용자 액세스 키는 장기 자격 증명입니다. 이러한 자격 증명이 잘못 처리된 경우 만료되도록 하려면 키가 더 이상 유효하지 않은 날짜를 지정하는 인라인 정책을 생성하여 자격 증명에 시간 제한을 적용할 수 있습니다.

1. 방금 생성한 IAM 사용자를 엽니다. 권한 탭에서 권한 추가를 선택한 다음 인라인 정책 생성을 선택합니다.
2. JSON 편집기에서 다음 권한을 지정합니다. 이 정책을 사용하려면 예제 정책의 `aws:CurrentTime` 타임스탬프 값을 자신의 시간 및 날짜로 바꿉니다.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {
          "aws:CurrentTime": "2024-01-01T00:00:00Z"
        }
      }
    }
  ]
}
```

액세스 키 생성

1. 사용자 세부 정보 페이지에서 보안 자격 증명 탭을 선택합니다. 액세스 키 섹션에서 액세스 키 생성을 선택합니다.
2. 기타에 키를 사용할 것인지 표시한 다음 다음을 선택하고 액세스 키 생성을 선택합니다.
3. 액세스 키 검색 페이지에서 표시를 선택하여 사용자의 보안 액세스 키 값을 표시합니다. 자격 증명을 복사하거나 .csv 파일을 다운로드할 수 있습니다.

사용자 액세스 키 저장

- 작업자 호스트 시스템의 에이전트 사용자의 AWS 자격 증명 파일에 사용자 액세스 키를 저장합니다.
  - Linux에서 파일은에 있습니다. `~/.aws/credentials`
  - Windows에서 파일은에 있습니다. `%USERPROFILE\.aws\credentials`

  다음 키를 바꿉니다.

```
[default]
aws_access_key_id=ACCESS_KEY_ID
aws_secret_access_key=SECRET_ACCESS_KEY
```

> **Important**
>
> 이 IAM 사용자가 더 이상 필요하지 않은 경우 AWS 보안 모범 사례에 맞게 제거하는 것이 좋습니다. 에 액세스할 AWS IAM Identity Center 때를 통해 인간 사용자에게 임시 자격 증명을 사용하도록 요구하는 것이 좋습니다 AWS.

고객 관리형 플릿에 대한 작업자 호스트 데이터 흐름

이 주제에서는 연락한 엔드포인트, 사용된 프로토콜 및 전송된 데이터를 포함하여 AWS Deadline Cloud(Deadline Cloud) 작업자 호스트가 작업 중에 수행하는 네트워크 연결에 대해 설명합니다. 이 정보는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 및 온프레미스 작업자를 포함한 고객 관리형 플릿(CMF) 작업자에게 적용됩니다. 이 정보를 사용하여 방화벽 규칙을 구성하거나, VPC 엔드포인트를 생성하거나, 보안 감사를 수행하거나, 작업자 호스트에 대한 네트워크 정책을 계획할 수 있습니다. 서비스 관리형 플릿 네트워킹에 대한 자세한 내용은 섹션을 참조하세요인터네트워크 트래픽 개인 정보 보호.

모든 작업자 통신은 아웃바운드 전용입니다. 작업자 호스트는 모든 연결을 시작합니다. 인바운드 연결을 허용할 필요는 없습니다. 모든 연결은 포트 443을 통해 HTTPS(TLS 1.2 이상)를 사용합니다.

이 주제에는 다음 섹션이 포함되어 있습니다.

- the section called “엔드포인트 및 프로토콜”

- the section called “작업자가 사용하는 API 작업”
- the section called “전송된 기타 데이터”
- the section called “프라이빗 연결 옵션”

엔드포인트 및 프로토콜

다음 표에는 작업자 호스트가 작업 중에 연결하는 AWS 서비스 엔드포인트가 나열되어 있습니다. 각 서비스에 대한 리전 엔드포인트의 전체 목록은 AWS 일반 참조의 서비스 엔드포인트 및 할당량을 참조하세요.

작업자 호스트 엔드포인트 참조

AWS 서비스	엔드포인트	포트/프로토콜	용도	필수
Deadline Cloud(예약)	`scheduling.deadline. [Region].amazonaws.com`	443/HTTPS	작업자 등록, 작업 폴링, 상태 업데이트, 자격 증명 교환, 작업 엔터티 검색. the section called “작업자가 사용하는 API 작업”을(를) 참조하세요.	항상
Amazon CloudWatch Logs(CloudWatch Logs)	`logs.[Region].amazonaws.com`	443/HTTPS	작업자 에이전트 및 세션 로그 전송.	항상
Amazon Simple Storage Service(Amazon S3)	`s3.[Region].amazonaws.com`	443/HTTPS	작업 연결 업로드 및 다운로드.	작업 첨부 파일을 사용하는 경우

작업에서 다른 AWS 서비스를 사용하는 경우 해당 서비스 엔드포인트에 대한 아웃바운드 연결을 허용해야 할 수도 있습니다.

작업자가 사용하는 API 작업

다음 API 작업은 모두 `scheduling.deadline.[Region].amazonaws.com` 엔드포인트를 사용합니다. 각 작업의 전체 요청 및 응답 스키마는 Deadline Cloud API 참조를 참조하세요.

부트스트랩 단계

작업자 호스트가 시작되면 작업자 에이전트가 플릿에 등록합니다. 부트스트랩 자격 증명에는 `AWSDeadlineCloud-WorkerHost` AWS 관리형 정책의 권한 또는 이에 상응하는 사용자 지정 권한이 필요합니다. 부트스트랩 단계에서는 다음 API 작업을 사용합니다.

- `CreateWorker` - 작업자를 플릿에 등록합니다. 호스트 이름과 IP 주소를 전송합니다. 작업자 ID를 수신합니다.
- `AssumeFleetRoleForWorker` - 플릿 역할 자격 증명을 가져옵니다. 작업자 에이전트가 후속 작업에 사용하는 임시 AWS 자격 증명을 수신합니다.

운영 단계

부트스트랩 후 작업자 에이전트는 작업에 대해 폴링하고 세션을 처리합니다. 플릿 역할에는 `AWSDeadlineCloud-FleetWorker` AWS 관리형 정책의 권한 또는 이에 상응하는 사용자 지정 권한이 필요하며 다음 API 작업을 사용합니다.

- `UpdateWorker` - 예를 들어 종료 `STOPPED` 중에 작업자 상태를 로 업데이트합니다.
- `UpdateWorkerSchedule` - 작업 할당에 대한 폴링입니다. 완료 상태, 진행률, 진행 메시지 및 출력 매니페스트 해시를 포함한 세션 작업 상태 업데이트를 전송합니다. 할당된 세션(작업 ID, 대기열 ID, 세션 작업, 로그 구성), 취소 요청, 원하는 작업자 상태 및 업데이트 간격을 수신합니다.
- `BatchGetJobEntity` - 할당된 작업에 대한 작업 세부 정보를 가져옵니다. 작업 개체 식별자를 전송합니다. 작업 세부 정보, 환경 세부 정보 및 작업 연결 세부 정보를 수신합니다.
- `AssumeFleetRoleForWorker` - 플릿 역할 자격 증명을 주기적으로 새로 고칩니다.
- `AssumeQueueRoleForWorker` - 특정 대기열로 범위가 지정된 대기열 역할 자격 증명을 가져옵니다. 작업자는 이러한 자격 증명을 사용하여 Amazon S3의 작업 연결에 액세스합니다.

전송된 기타 데이터

Deadline Cloud Schedule API 작업 외에도 작업자 호스트는 다음 데이터를 다른 AWS 서비스로 전송합니다.

로그 데이터

작업자 에이전트는 `PutLogEvents` API 작업을 사용하여 작업자 에이전트 로그 및 세션 로그(작업 프로세스의 stdout 및 stderr)를 CloudWatch Logs로 보냅니다.

작업 첨부 파일

작업자는 `GetObject` 및 `PutObject` API 작업을 사용하여 Amazon S3를 통해 입력 및 출력 파일을 전송합니다. 작업자는이 액세스를 `AssumeQueueRoleForWorker` 위해를 통해 얻은 대기열 역할 자격 증명을 사용합니다.

원격 측정(선택 사항)

작업자 에이전트는 충돌 보고서와 같은 운영 지표를 전송합니다. 원격 측정 수집을 옵트아웃할 수 있습니다. 자세한 내용은 옵트아웃 단원을 참조하십시오.

프라이빗 연결 옵션

AWS PrivateLink 를 사용하여 퍼블릭 인터넷을 통과하지 않고도 VPC 내에서 CMF 작업자 호스트와 Deadline Cloud 간의 트래픽을 유지할 수 있습니다. 온프레미스 작업자의 경우 AWS Direct Connect (Direct Connect) 또는 VPN 연결과 결합 AWS PrivateLink 할 수 있습니다. 자세한 내용은 인터페이스 엔드포인트를 AWS Deadline Cloud 사용한 액세스(AWS PrivateLink) 단원을 참조하십시오.

작업자 호스트의 구성 테스트

작업자 에이전트를 설치하고, 작업을 처리하는 데 필요한 소프트웨어를 설치하고, 작업자 에이전트의 AWS 자격 증명을 구성한 후에는 AMI플릿에 대한를 생성하기 전에 설치가 작업을 처리할 수 있는지 테스트해야 합니다. 다음을 테스트해야 합니다.

- Deadline Cloud 작업자 에이전트가 시스템 서비스로 실행되도록 올바르게 구성되어 있습니다.
- 작업자가 작업을 위해 연결된 대기열을 폴링합니다.
- 작업자가 플릿과 연결된 대기열로 전송된 작업을 성공적으로 처리했는지 여부.

구성을 테스트하고 대표 작업을 성공적으로 처리할 수 있게 되면 구성된 작업자를 사용하여 Amazon EC2 작업자AMI용 또는 온프레미스 작업자용 모델을 생성할 수 있습니다.

> **Note**
>
> Auto Scaling 플릿의 작업자 호스트 구성을 테스트하는 경우 다음과 같은 상황에서 작업자를 테스트하는 데 어려움을 겪을 수 있습니다.

- 대기열에 작업이 없는 경우 Deadline Cloud는 작업자 시작 직후 작업자 에이전트를 중지합니다.
- 작업자 에이전트가 중지 시 호스트를 종료하도록 구성된 경우 대기열에 작업이 없으면 에이전트가 시스템을 종료합니다.

이러한 문제를 방지하려면 자동 확장되지 않는 스테이징 플릿을 사용하여 작업자를 구성하고 테스트합니다. 작업자 호스트를 테스트한 후를 베이킹하기 전에 올바른 플릿 ID를 설정해야 합니다AMI.

작업자 호스트 구성을 테스트하려면

1. 운영 체제 서비스를 시작하여 작업자 에이전트를 실행합니다.

   Linux

   루트 쉘에서 다음 명령을 실행합니다.

   ```
   systemctl start deadline-worker
   ```

   Windows

   관리자 명령 프롬프트 또는 PowerShell 터미널에서 다음 명령을 입력합니다.

   ```
   sc.exe start DeadlineWorker
   ```

2. 작업자를 모니터링하여 작업을 시작하고 폴링하는지 확인합니다.

   Linux

   루트 쉘에서 다음 명령을 실행합니다.

   ```
   systemctl status deadline-worker
   ```

   명령은 다음과 같은 응답을 반환해야 합니다.

   ```
   Active: active (running) since Wed 2023-06-14 14:44:27 UTC; 7min ago
   ```

응답이 이와 같지 않으면 다음 명령을 사용하여 로그 파일을 검사합니다.

```
tail -n 25 /var/log/amazon/deadline/worker-agent.log
```

Windows

관리자 명령 프롬프트 또는 PowerShell 터미널에서 다음 명령을 입력합니다.

```
sc.exe query DeadlineWorker
```

명령은 다음과 같은 응답을 반환해야 합니다.

```
STATE   : 4 RUNNING
```

응답에가 포함되어 있지 않은 경우 작업자 로그 파일을 RUNNING검사합니다. 및 관리자 PowerShell 프롬프트를 열고 다음 명령을 실행합니다.

```
Get-Content -Tail 25 -Path $env:PROGRAMDATA\Amazon\Deadline\Logs\worker-
agent.log
```

3. 플릿과 연결된 대기열에 작업을 제출합니다. 작업은 플릿이 처리하는 작업을 대표해야 합니다.
4. Deadline Cloud 모니터 또는 CLI를 사용하여 작업 진행 상황을 모니터링합니다. 작업이 실패하면 세션 및 작업자 로그를 확인합니다.
5. 작업이 성공적으로 완료될 때까지 필요에 따라 작업자 호스트의 구성을 업데이트합니다.
6. 테스트 작업이 성공하면 작업자를 중지할 수 있습니다.

Linux

루트 쉘에서 다음 명령을 실행합니다.

```
systemctl stop deadline-worker
```

Windows

관리자 명령 프롬프트 또는 PowerShell 터미널에서 다음 명령을 입력합니다.

```
sc.exe stop DeadlineWorker
```

Amazon Machine Image 생성

Amazon Elastic Compute Cloud(Amazon EC2AMI) 고객 관리형 플릿(CMF)에서 사용할 Amazon Machine Image ()을 생성하려면이 섹션의 작업을 완료합니다. 계속하기 전에 Amazon EC2 인스턴스를 생성해야 합니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스 시작을 참조하세요. Amazon EC2

> **Important**
>
> 를 생성하면 Amazon EC2 인스턴스에 연결된 볼륨의 스냅샷이 AMI 생성됩니다. 인스턴스에 설치된 모든 소프트웨어는 유지되므로에서 인스턴스를 시작할 때 재사용되는 인스턴스가 유지됩니다AMI. 플릿에 적용하기 전에 패치 전략을 채택하고 업데이트된 소프트웨어AMI로 새를 정기적으로 업데이트하는 것이 좋습니다.

Amazon EC2 인스턴스 준비

를 빌드AMI하기 전에 작업자 상태를 삭제해야 합니다. 작업자 상태는 작업자 에이전트가 시작될 때까지 유지됩니다. 이 상태가에 지속되면 해당 상태에서 시작된 AMI모든 인스턴스가 동일한 상태를 공유합니다.

기존 로그 파일도 삭제하는 것이 좋습니다. AMI를 준비할 때 로그 파일은 Amazon EC2 인스턴스에 남아 있을 수 있습니다. 이러한 파일을 삭제하면 AMI를 사용하는 작업자 플릿에서 발생할 수 있는 문제를 진단할 때 혼동이 최소화됩니다.

또한 Amazon EC2가 시작될 때 Deadline Cloud 작업자 에이전트가 시작되도록 작업자 에이전트 시스템 서비스를 활성화해야 합니다.

마지막으로 작업자 에이전트 자동 종료를 활성화하는 것이 좋습니다. 이를 통해 작업자 플릿은 필요할 때 스케일 업하고 렌더링 작업이 완료되면 종료할 수 있습니다. 이 Auto Scaling은 필요에 따라 리소스만 사용하도록 하는 데 도움이 됩니다.

Amazon EC2 인스턴스를 준비하려면

1. Amazon EC2 콘솔을 엽니다.
2. Amazon EC2 인스턴스 시작 자세한 내용은 인스턴스 시작을 참조하세요.
3. ID 제공업체(IdP)에 연결하도록 호스트를 설정한 다음 필요한 공유 파일 시스템을 탑재합니다.
4. 자습서를 따라 Deadline Cloud 작업자 에이전트 설치, 작업자 에이전트 구성,를 수행합니다작업 사용자 및 그룹 생성.

5. VFX 참조 플랫폼과 호환되는 소프트웨어를 실행하기 위해 Amazon Linux 2023 AMI 기반를 준비하는 경우 몇 가지 요구 사항을 업데이트해야 합니다. 자세한 내용은 AWS Deadline Cloud 사용 설명서의 VFX 참조 플랫폼 호환성을 참조하세요.
6. 터미널을 엽니다.
   a. Linux에서 `root` 사용자로 터미널을 엽니다(또는 `sudo` / 사용`su`).
   b. 에서 관리자 명령 프롬프트 또는 PowerShell 터미널을 Windows엽니다.
7. 작업자 서비스가 실행되고 있지 않고 부팅 시 시작되도록 구성되어 있는지 확인합니다.
   a. Linux에서를 실행합니다.

```
systemctl stop deadline-worker
systemctl enable deadline-worker
```

   b. 에서 Windows를 실행합니다.

```
sc.exe stop DeadlineWorker
sc.exe config DeadlineWorker start= auto
```

8. 작업자 상태를 삭제합니다.
   a. Linux에서를 실행합니다.

```
rm -rf /var/lib/deadline/*
```

   b. 에서 Windows를 실행합니다.

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Cache\*
```

9. 로그 파일을 삭제합니다.
   a. Linux에서를 실행합니다.

```
rm -rf /var/log/amazon/deadline/*
```

   b. 에서 Windows를 실행합니다.

```
del /Q /S %PROGRAMDATA%\Amazon\Deadline\Logs\*
```

10. 에서는 시작 메뉴에 있는 Amazon EC2Launch 설정 애플리케이션을 실행하여 인스턴스의 최종 호스트 준비 및 종료를 완료하는 Windows것이 좋습니다.

> **Note**
>
> 반드시 Sysprep 없이 종료를 선택하고 Sysprep으로 종료를 선택하면 안 됩니다. Sysprep으로 종료하면 모든 로컬 사용자를 사용할 수 없게 됩니다. 자세한 내용은 Windows 인스턴스용 사용 설명서의 사용자 지정 AMI 생성 주제의 시작하기 전 섹션을 참조하세요.

빌드 AMI

를 빌드하려면 AMI

1. Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 인스턴스를 선택한 다음 인스턴스를 선택합니다.
3. 인스턴스 상태를 선택한 다음 인스턴스 중지를 선택합니다.
4. 인스턴스가 중지된 후 작업을 선택합니다.
5. 이미지 및 템플릿을 선택한 다음 이미지 생성을 선택합니다.
6. 이미지 이름을 입력합니다.
7. (선택 사항) 이미지에 대한 설명을 입력합니다.
8. 이미지 생성을 선택합니다.

Amazon EC2 Auto Scaling 그룹을 사용하여 플릿 인프라 생성

이 섹션에서는 Amazon EC2 Auto Scaling 플릿을 생성하는 방법을 설명합니다.

아래 CloudFormation YAML 템플릿을 사용하여 Amazon EC2 Auto Scaling(Auto Scaling) 그룹, 두 개의 서브넷이 있는 Amazon Virtual Private Cloud(Amazon VPC), 인스턴스 프로파일 및 인스턴스 액세스 역할을 생성합니다. 이는 서브넷에서 Auto Scaling을 사용하여 인스턴스를 시작하는 데 필요합니다.

렌더링 요구 사항에 맞게 인스턴스 유형 목록을 검토하고 업데이트해야 합니다.

CloudFormation YAML 템플릿에 사용되는 리소스 및 파라미터에 대한 전체 설명은 AWS CloudFormation 사용 설명서의 Deadline Cloud 리소스 유형 참조를 참조하세요.

Amazon EC2 Auto Scaling 플릿을 생성하려면

1. 다음 예제를 사용하여 FarmID, FleetID및 AMIId 파라미터를 정의하는 CloudFormation 템플릿을 생성합니다. 템플릿을 로컬 컴퓨터의 .YAML 파일에 저장합니다.

```
AWSTemplateFormatVersion: 2010-09-09
Description: Amazon Deadline Cloud customer-managed fleet
Parameters:
  FarmId:
    Type: String
    Description: Farm ID
  FleetId:
    Type: String
    Description: Fleet ID
  AMIId:
    Type: String
    Description: AMI ID for launching workers
Resources:
  deadlineVPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: 100.100.0.0/16
  deadlineWorkerSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: !Join
        - ' '
        - - Security group created for Deadline Cloud workers in the fleet
          - !Ref FleetId
      GroupName: !Join
        - ''
        - - deadlineWorkerSecurityGroup-
          - !Ref FleetId
      SecurityGroupEgress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: '-1'
      SecurityGroupIngress: []
      VpcId: !Ref deadlineVPC
  deadlineIGW:
    Type: 'AWS::EC2::InternetGateway'
    Properties: {}
  deadlineVPCGatewayAttachment:
    Type: 'AWS::EC2::VPCGatewayAttachment'
    Properties:
      VpcId: !Ref deadlineVPC
```

```
      InternetGatewayId: !Ref deadlineIGW
  deadlinePublicRouteTable:
    Type: 'AWS::EC2::RouteTable'
    Properties:
      VpcId: !Ref deadlineVPC
  deadlinePublicRoute:
    Type: 'AWS::EC2::Route'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref deadlineIGW
    DependsOn:
      - deadlineIGW
      - deadlineVPCGatewayAttachment
  deadlinePublicSubnet0:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref deadlineVPC
      CidrBlock: 100.100.16.0/22
      AvailabilityZone: !Join
        - ''
        - - !Ref 'AWS::Region'
          - a
  deadlineSubnetRouteTableAssociation0:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      SubnetId: !Ref deadlinePublicSubnet0
  deadlinePublicSubnet1:
    Type: 'AWS::EC2::Subnet'
    Properties:
      VpcId: !Ref deadlineVPC
      CidrBlock: 100.100.20.0/22
      AvailabilityZone: !Join
        - ''
        - - !Ref 'AWS::Region'
          - c
  deadlineSubnetRouteTableAssociation1:
    Type: 'AWS::EC2::SubnetRouteTableAssociation'
    Properties:
      RouteTableId: !Ref deadlinePublicRouteTable
      SubnetId: !Ref deadlinePublicSubnet1
  deadlineInstanceAccessAccessRole:
    Type: 'AWS::IAM::Role'
```

```
    Properties:
      RoleName: !Join
        - '-'
        - - deadline
          - InstanceAccess
          - !Ref FleetId
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: ec2.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Path: /
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy'
        - 'arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore'
        - 'arn:aws:iam::aws:policy/AWSDeadlineCloud-WorkerHost'
  deadlineInstanceProfile:
    Type: 'AWS::IAM::InstanceProfile'
    Properties:
      Path: /
      Roles:
        - !Ref deadlineInstanceAccessAccessRole
  deadlineLaunchTemplate:
    Type: 'AWS::EC2::LaunchTemplate'
    Properties:
      LaunchTemplateName: !Join
        - ''
        - - deadline-LT-
          - !Ref FleetId
      LaunchTemplateData:
        NetworkInterfaces:
          - DeviceIndex: 0
            AssociatePublicIpAddress: true
            Groups:
              - !Ref deadlineWorkerSecurityGroup
            DeleteOnTermination: true
        ImageId: !Ref AMIId
        InstanceInitiatedShutdownBehavior: terminate
        IamInstanceProfile:
          Arn: !GetAtt
            - deadlineInstanceProfile
            - Arn
```

```
        MetadataOptions:
          HttpTokens: required
          HttpEndpoint: enabled

  deadlineAutoScalingGroup:
    Type: 'AWS::AutoScaling::AutoScalingGroup'
    Properties:
      AutoScalingGroupName: !Join
        - ''
        - - deadline-ASG-autoscalable-
          - !Ref FleetId
      MinSize: 0
      MaxSize: 10
      VPCZoneIdentifier:
        - !Ref deadlinePublicSubnet0
        - !Ref deadlinePublicSubnet1
      NewInstancesProtectedFromScaleIn: true
      MixedInstancesPolicy:
        InstancesDistribution:
          OnDemandBaseCapacity: 0
          OnDemandPercentageAboveBaseCapacity: 0
          SpotAllocationStrategy: capacity-optimized
          OnDemandAllocationStrategy: lowest-price
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref deadlineLaunchTemplate
            Version: !GetAtt
              - deadlineLaunchTemplate
              - LatestVersionNumber
          Overrides:
            - InstanceType: m5.large
            - InstanceType: m5d.large
            - InstanceType: m5a.large
            - InstanceType: m5ad.large
            - InstanceType: m5n.large
            - InstanceType: m5dn.large
            - InstanceType: m4.large
            - InstanceType: m3.large
            - InstanceType: r5.large
            - InstanceType: r5d.large
            - InstanceType: r5a.large
            - InstanceType: r5ad.large
            - InstanceType: r5n.large
            - InstanceType: r5dn.large
```

```
            - InstanceType: r4.large
      MetricsCollection:
        - Granularity: 1Minute
          Metrics:
            - GroupMinSize
            - GroupMaxSize
            - GroupDesiredCapacity
            - GroupInServiceInstances
            - GroupTotalInstances
            - GroupInServiceCapacity
            - GroupTotalCapacity
```

2. https://console.aws.amazon.com/cloudformation CloudFormation 콘솔을 엽니다.

   CloudFormation 콘솔을 사용하여 생성한 템플릿 파일 업로드 지침을 사용하여 스택을 생성합니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 CloudFormation 콘솔에서 스택 생성을 참조하세요.

> **Note**
>
> - 작업자의 Amazon EC2 인스턴스에 연결된 IAM 역할의 자격 증명은 작업을 포함하여 해당 작업자에서 실행되는 모든 프로세스에서 사용할 수 있습니다. 작업자는 작업할 수 있는 최소 권한이 있어야 합니다. `deadline:CreateWorker` `deadline:AssumeFleetRoleForWorker`.
> - 작업자 에이전트는 대기열 역할에 대한 자격 증명을 얻고 작업을 실행하여 사용하도록 구성합니다. Amazon EC2 인스턴스 프로파일 역할에는 작업에 필요한 권한이 포함되어서는 안 됩니다.

Deadline Cloud 규모 권장 기능을 사용하여 Amazon EC2 플릿 자동 규모 조정

Deadline Cloud는 Amazon EC2 Auto Scaling(Auto Scaling) 그룹을 활용하여 Amazon EC2 고객 관리형 플릿(CMF)을 자동으로 확장합니다. 플릿 모드를 구성하고 계정에 필요한 인프라를 배포하여 플릿을 자동으로 확장해야 합니다. 배포한 인프라는 모든 플릿에서 작동하므로 한 번만 설정하면 됩니다.

기본 워크플로는 플릿 모드를 자동으로 조정하도록 구성하면 Deadline Cloud는 권장 플릿 크기가 변경될 때마다 해당 플릿에 대한 EventBridge 이벤트를 전송합니다(이벤트 하나에 플릿 ID, 권장 플

릿 크기 및 기타 메타데이터 포함). 관련 이벤트를 필터링하고 Lambda가 이벤트를 소비하도록 하는 EventBridge 규칙이 있습니다. Lambda는 Amazon EC2 Auto Scaling과 통합되어 Amazon EC2 AutoScalingGroup 플릿을 자동으로 확장합니다.

플릿 모드를 로 설정 EVENT_BASED_AUTO_SCALING

플릿 모드를 로 구성합니다EVENT_BASED_AUTO_SCALING. 콘솔을 사용하여이 작업을 수행하거나를 사용하여 CreateFleet 또는 UpdateFleet API를 직접 호출 AWS CLI 할 수 있습니다. 모드가 구성되면 Deadline Cloud는 권장 플릿 크기가 변경될 때마다 EventBridge 이벤트 전송을 시작합니다.

- 예제 UpdateFleet 명령:

```
aws deadline update-fleet \
  --farm-id FARM_ID \
  --fleet-id FLEET_ID \
  --configuration file://configuration.json
```

- 예제 CreateFleet 명령:

```
aws deadline create-fleet \
  --farm-id FARM_ID \
  --display-name "Fleet name" \
  --max-worker-count 10 \
  --configuration file://configuration.json
```

다음은 위의 CLI 명령에 configuration.json 사용되는의 예입니다(--configuration file://configuration.json).

- 플릿에서 Auto Scaling을 활성화하려면 모드를 로 설정해야 합니다EVENT_BASED_AUTO_SCALING.
- workerCapabilities는 CMF를 생성할 때 CMF에 할당된 기본값입니다. CMF에서 사용 가능한 리소스를 늘려야 하는 경우 이러한 값을 변경할 수 있습니다.

플릿 모드를 구성한 후 Deadline Cloud는 해당 플릿에 대한 플릿 크기 권장 이벤트를 보내기 시작합니다.

```
{
    "customerManaged": {
        "mode": "EVENT_BASED_AUTO_SCALING",
        "workerCapabilities": {
```

```
            "vCpuCount": {
                "min": 1,
                "max": 4
            },
            "memoryMiB": {
                "min": 1024,
                "max": 4096
            },
            "osFamily": "linux",
            "cpuArchitectureType": "x86_64"
        }
    }
}
```

CloudFormation 템플릿을 사용하여 Auto Scaling 스택 배포

이벤트를 필터링하는 EventBridge 규칙, 이벤트를 소비하고 Auto Scaling을 제어하는 Lambda, 처리되지 않은 이벤트를 저장하는 SQS 대기열을 설정할 수 있습니다. 다음 CloudFormation 템플릿을 사용하여 스택의 모든 항목을 배포합니다. 리소스를 성공적으로 배포한 후에는 작업을 제출할 수 있으며 플릿이 자동으로 확장됩니다.

```
Resources:
  AutoScalingLambda:
    Type: 'AWS::Lambda::Function'
    Properties:
      Code:
        ZipFile: |-
          """
          This lambda is configured to handle "Fleet Size Recommendation Change"
          messages. It will handle all such events, and requires
          that the ASG is named based on the fleet id. It will scale up/down the fleet
          based on the recommended fleet size in the message.

          Example EventBridge message:
          {
              "version": "0",
              "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
              "detail-type": "Fleet Size Recommendation Change",
              "source": "aws.deadline",
              "account": "111122223333",
              "time": "2017-12-22T18:43:48Z",
              "region": "us-west-1",
              "resources": [],
```

```
            "detail": {
                "farmId": "farm-12345678900000000000000000000000",
                "fleetId": "fleet-12345678900000000000000000000000",
                "oldFleetSize": 1,
                "newFleetSize": 5,
            }
        }
        """

        import json
        import boto3
        import logging

        logger = logging.getLogger()
        logger.setLevel(logging.INFO)

        auto_scaling_client = boto3.client("autoscaling")

        def lambda_handler(event, context):
            logger.info(event)
            event_detail = event["detail"]
            fleet_id = event_detail["fleetId"]
            desired_capacity = event_detail["newFleetSize"]

            asg_name = f"deadline-ASG-autoscalable-{fleet_id}"
            auto_scaling_client.set_desired_capacity(
                AutoScalingGroupName=asg_name,
                DesiredCapacity=desired_capacity,
                HonorCooldown=False,
            )

            return {
                'statusCode': 200,
                'body': json.dumps(f'Successfully set desired_capacity for {asg_name}
to {desired_capacity}')
            }
     Handler: index.lambda_handler
     Role: !GetAtt
       - AutoScalingLambdaServiceRole
       - Arn
     Runtime: python3.11
   DependsOn:
     - AutoScalingLambdaServiceRoleDefaultPolicy
     - AutoScalingLambdaServiceRole
```

```
  AutoScalingEventRule:
    Type: 'AWS::Events::Rule'
    Properties:
      EventPattern:
        source:
          - aws.deadline
        detail-type:
          - Fleet Size Recommendation Change
      State: ENABLED
      Targets:
        - Arn: !GetAtt
            - AutoScalingLambda
            - Arn
          DeadLetterConfig:
            Arn: !GetAtt
              - UnprocessedAutoScalingEventQueue
              - Arn
          Id: Target0
          RetryPolicy:
            MaximumRetryAttempts: 15
  AutoScalingEventRuleTargetPermission:
    Type: 'AWS::Lambda::Permission'
    Properties:
      Action: 'lambda:InvokeFunction'
      FunctionName: !GetAtt
        - AutoScalingLambda
        - Arn
      Principal: events.amazonaws.com
      SourceArn: !GetAtt
        - AutoScalingEventRule
        - Arn
  AutoScalingLambdaServiceRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
        Version: 2012-10-17
      ManagedPolicyArns:
        - !Join
          - ''
```

```
          - - 'arn:'
            - !Ref 'AWS::Partition'
            - ':iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'
  AutoScalingLambdaServiceRoleDefaultPolicy:
    Type: 'AWS::IAM::Policy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'autoscaling:SetDesiredCapacity'
            Effect: Allow
            Resource: '*'
        Version: 2012-10-17
      PolicyName: AutoScalingLambdaServiceRoleDefaultPolicy
      Roles:
        - !Ref AutoScalingLambdaServiceRole
  UnprocessedAutoScalingEventQueue:
    Type: 'AWS::SQS::Queue'
    Properties:
      QueueName: deadline-unprocessed-autoscaling-events
    UpdateReplacePolicy: Delete
    DeletionPolicy: Delete
  UnprocessedAutoScalingEventQueuePolicy:
    Type: 'AWS::SQS::QueuePolicy'
    Properties:
      PolicyDocument:
        Statement:
          - Action: 'sqs:SendMessage'
            Condition:
              ArnEquals:
                'aws:SourceArn': !GetAtt
                  - AutoScalingEventRule
                  - Arn
            Effect: Allow
            Principal:
              Service: events.amazonaws.com
            Resource: !GetAtt
              - UnprocessedAutoScalingEventQueue
              - Arn
        Version: 2012-10-17
      Queues:
        - !Ref UnprocessedAutoScalingEventQueue
```

플릿 상태 확인 수행

플릿을 생성한 후에는 사용자 지정 상태 확인을 구축하여 플릿이 정상 상태를 유지하고 인스턴스가 중단되지 않도록 하여 불필요한 비용을 방지해야 합니다. GitHub의 Deadline Cloud 플릿 상태 확인 배포를 참조하세요. 상태 확인은 감지되지 않은 상태로 실행 중인 Amazon Machine Image, 시작 템플릿 또는 네트워크 구성의 우발적인 변경 위험을 낮출 수 있습니다.

Deadline Cloud 서비스 관리형 플릿 구성 및 사용

서비스 관리형 플릿(SMF)은 Deadline Cloud에서 관리하는 작업자 모음입니다. SMF를 사용하면 처리 수요에 대한 플릿 조정을 관리하거나 작업 완료 후 플릿 크기를 줄일 필요가 없습니다.

SMF가 기본 conda 대기열 환경을 사용하여 대기열과 연결되면 Deadline Cloud는 적절한 소프트웨어 패키지로 플릿의 작업자를 구성합니다. 지원되는 파트너 애플리케이션의 경우 AWS Deadline Cloud 사용 설명서의 기본 conda 대기열 환경을 참조하세요.

대부분의 경우 워크로드를 처리하기 위해 SMF를 변경할 필요가 없습니다. 그러나 일부 상황에서는 플릿을 변경해야 할 수 있습니다.

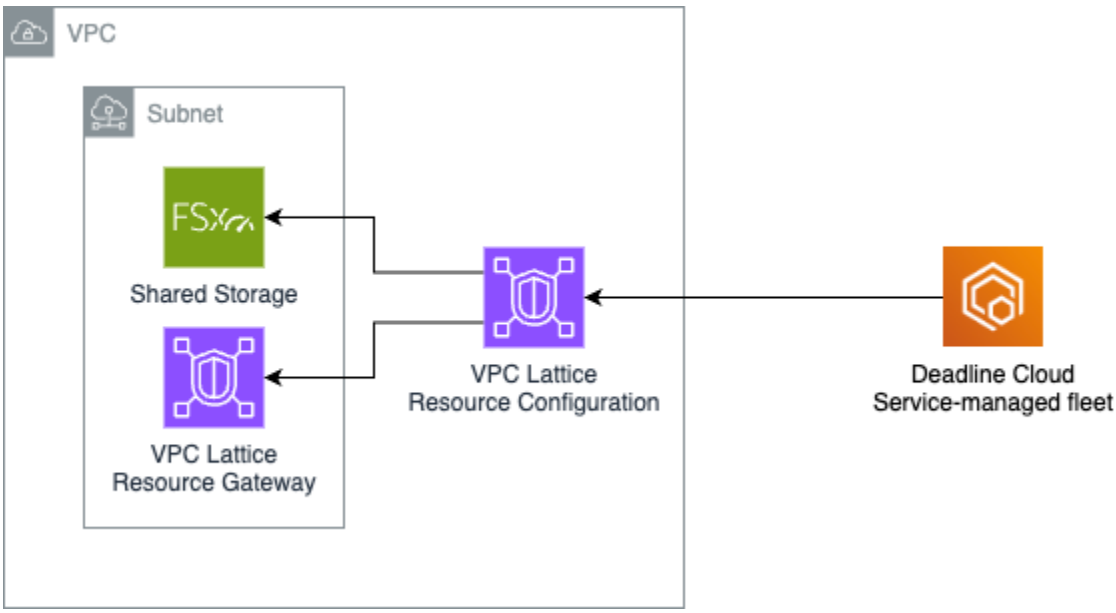
> **Note**
>
> 호스트 구성 스크립트를 사용하여 작업자에 사용자 지정 소프트웨어를 설치하려면 섹션을 참조하세요관리자 권한으로 호스트 구성 스크립트 실행.

주제

- VPC 리소스 엔드포인트를 사용하여 VPC 리소스를 SMF에 연결
- 서비스 관리형 플릿에서 작업 연결 사용

VPC 리소스 엔드포인트를 사용하여 VPC 리소스를 SMF에 연결

Deadline Cloud 서비스 관리형 플릿(SMF)용 Amazon VPC 리소스 엔드포인트를 사용하면 네트워크 파일 시스템(NFS), 라이선스 서버 및 데이터베이스와 같은 VPC 리소스를 Deadline Cloud 작업자와 연결할 수 있습니다. 이 기능을 사용하면 Deadline Cloud의 완전관리형 플랫폼을 활용하는 동시에 VPC 내의 기존 인프라와 통합할 수 있습니다.



> **Tip**
>
> Amazon FSx 클러스터를 설정하고 서비스 관리형 플릿에 연결하는 참조 CloudFormation 템플릿은 GitHub의 Deadline Cloud 샘플 리포지토리에서 smf_vpc_fsx를 참조하세요.

VPC 리소스 엔드포인트 작동 방식

VPC 리소스 엔드포인트는 VPC Lattice를 사용하여 Deadline Cloud SMF 작업자와 VPC의 리소스 간에 보안 연결을 생성합니다. 연결은 단방향입니다. 즉, 작업자가 VPC의 리소스에 대한 연결을 설정하고 데이터를 주고받을 수 있지만 VPC의 리소스는 작업자에 대한 연결을 설정할 수 없습니다.

VPC 리소스를 Deadline Cloud 서비스 관리형 플릿에 연결하면 작업자가 프라이빗 도메인 이름을 사용하여 VPC의 리소스에 액세스할 수 있습니다. 또한 VPC Lattice를 통해 작업자에서 VPC 리소스로 트래픽이 흐르고 VPC의 리소스는 VPC Lattice 리소스 게이트웨이에서 오는 트래픽을 확인합니다.

자세한 내용은 VPC Lattice 사용 설명서를 참조하세요.

사전 조건

Deadline Cloud 서비스 관리형 플릿에 VPC 리소스를 연결하기 전에 다음이 있는지 확인합니다.

- 연결하려는 리소스가 포함된 VPC가 있는 AWS 계정입니다.
- VPC Lattice 리소스를 생성하고 관리할 수 있는 IAM 권한.

- 하나 이상의 서비스 관리형 플릿이 있는 Deadline Cloud 팜입니다.
- 액세스하려는 VPC 리소스(FSx, NFS, 라이선스 서버 등).

VPC 리소스 엔드포인트 설정

VPC 리소스 엔드포인트를 설정하려면 VPC Lattice 및에서 리소스를 생성한 AWS RAM다음 해당 리소스를 Deadline Cloud의 플릿에 연결해야 합니다. SMF에 대한 VPC 리소스 엔드포인트를 설정하려면 다음 단계를 완료하세요.

1. VPC Lattice에서 리소스 게이트웨이를 생성하려면 VPC Lattice 사용 설명서의 리소스 게이트웨이 생성을 참조하세요.
2. VPC Lattice에서 리소스 구성을 생성하려면 VPC Lattice 사용 설명서의 리소스 구성 생성을 참조하세요.
3. Deadline Cloud 플릿과 리소스를 공유하려면 리소스 공유를 생성합니다 AWS RAM. 지침은 리소스 공유 생성을 참조하세요.

   리소스 공유를 생성하는 동안 보안 주체에 드롭다운에서 서비스 보안 주체를 선택한 다음를 입력합니다**`fleets.deadline.amazonaws.com`**.
4. 리소스 구성을 Deadline Cloud 플릿에 연결하려면 다음 단계를 완료하세요.

   a. 아직 없는 경우 Deadline Cloud 콘솔을 엽니다.

   b. 탐색 창에서 팜을 선택한 다음 팜을 선택합니다.

   c. 플릿 탭을 선택한 다음 플릿을 선택합니다.

   d. 구성(Configurations) 탭을 선택합니다.

   e. VPC 리소스 엔드포인트에서 편집을 선택합니다.

   f. 생성한 리소스 구성을 선택한 다음 변경 사항 저장을 선택합니다.

VPC 리소스에 액세스

VPC 리소스를 플릿에 연결한 후 작업자는 다음 형식의 프라이빗 도메인 이름을 사용하여 액세스할 수 있습니다. `<resource_config_id>.resource-endpoints.deadline.<region>.amazonaws.com`

이 도메인은 비공개이며 작업자만 액세스할 수 있습니다(인터넷 또는 워크스테이션에서 액세스할 수 없음).

작업자의 VPC 리소스에 대한 액세스를 탑재하거나 구성하려면 호스트 구성 스크립트를 사용합니다. 호스트 구성 스크립트는 작업자가 시작할 때 관리자 권한으로 실행되므로 파일 시스템을 탑재하거나, 네트워크 설정을 구성하거나, 기타 설정 작업을 수행할 수 있습니다.

인증 및 보안

인증이 필요한 리소스의 경우 AWS Secrets Manager에 보안 인증을 안전하게 저장하고, 호스트 구성 스크립트 또는 작업 스크립트에서 보안 암호에 액세스하고, 액세스를 제어하는 적절한 파일 시스템 권한을 구현합니다. 여러 플릿에서 리소스를 공유할 때 보안에 미치는 영향을 고려합니다. 예를 들어 두 플릿이 동일한 공유 스토리지에 연결된 경우 한 플릿에서 실행되는 작업은 다른 플릿에서 생성된 자산에 액세스할 수 있습니다.

기술적 고려 사항

VPC 리소스 엔드포인트를 사용할 때는 다음 사항을 고려하세요.

- 작업자에서 VPC 리소스로만 연결을 시작할 수 있으며, VPC 리소스에서 작업자로는 연결을 시작할 수 없습니다.
- 연결이 설정되면 리소스 구성이 연결 해제된 경우에도 연결이 재설정될 때까지 유지됩니다.
- VPC Lattice 연결은 추가 비용 없이 가용 영역 간의 연결을 자동으로 처리합니다. 리소스 게이트웨이는 VPC 리소스와 가용 영역을 공유해야 하므로 모든 가용 영역에 걸쳐 리소스 게이트웨이를 구성하는 것이 좋습니다.
- VPC 리소스 엔드포인트를 통과하는 트래픽은 모든 사용 사례와 호환되지 않는 NAT(Network Address Translation)를 사용합니다. 예를 들어 Microsoft Active Directory(AD)는 NAT를 통해 연결할 수 없습니다.

VPC Lattice 할당량에 대한 자세한 내용은 VPC Lattice 할당량을 참조하세요.

문제 해결

VPC 리소스 엔드포인트에 문제가 발생하면 다음을 확인하세요.

- "mount.nfs: 탑재 중 서버에서 액세스가 거부됨"과 같은 오류 메시지가 표시되면 NFS 볼륨의 클라이언트 구성을 업데이트해야 할 수 있습니다.
- Amazon EC2 인스턴스 또는 AWS CloudShell VPC에서 테스트하여 리소스 구성 설정을 확인합니다.
- 간단한 CLI 작업을 사용하여 Deadline Cloud 연결을 테스트합니다. 자세한 내용은 GitHub의 Deadline Cloud 샘플을 참조하세요.

- 연결 실패가 발생하는 경우 리소스 게이트웨이의 보안 그룹에 대한 설정을 확인합니다.
- VPC 액세스 로그를 활성화하여 연결을 모니터링합니다.

서비스 관리형 플릿에서 작업 연결 사용

작업 첨부 파일은 Amazon Simple Storage Service(Amazon S3)를 사용하여 워크스테이션과 Deadline Cloud 작업자 간에 파일을 전송합니다. 작업 첨부 파일을 단독으로 사용하거나 공유 스토리지와 함께 사용하여 로컬에 저장된 작업 스크립트, 구성 파일 또는 프로젝트 자산과 같이 다른 작업과 공유되지 않는 작업에 보조 데이터를 연결할 수 있습니다.

작업 첨부 파일의 작동 방식에 대한 자세한 내용은 Deadline Cloud 사용 설명서의 작업 첨부 파일을 참조하세요. 작업 번들에서 입력 및 출력 파일을 지정하는 방법에 대한 자세한 내용은 섹션을 참조하세요작업 첨부 파일을 사용하여 파일 공유.

파일 시스템 모드 선택

첨부 파일이 있는 작업을 제출할 때 `fileSystem` 속성을 설정하여 작업자가 Amazon S3에서 파일을 로드하는 방법을 선택할 수 있습니다.

- COPIED(기본값) - 작업이 시작되기 전에 모든 파일을 로컬 디스크에 다운로드합니다. 모든 작업에 대부분의 입력 파일이 필요한 경우에 가장 적합합니다.
- 가상 - 온디맨드 방식으로 파일을 다운로드하는 가상 파일 시스템을 탑재합니다. 작업에 입력 파일의 하위 집합만 필요한 경우에 가장 적합합니다. Linux SMF 작업자에서만 사용할 수 있습니다.

> **Important**
>
> 가상 모드 캐싱은 메모리 소비를 늘릴 수 있으며 모든 워크로드에 최적화되지는 않습니다. 프로덕션 작업을 실행하기 전에 워크로드를 테스트하는 것이 좋습니다.

파일 시스템 모드 구성에 대한 자세한 내용은 Deadline Cloud 사용 설명서의 가상 파일 시스템을 참조하세요.

전송 성능 최적화

Amazon S3에서 SMF 작업자로 파일을 동기화하는 속도는 플릿의 Amazon Elastic Block Store(Amazon EBS) 볼륨 구성에 따라 달라집니다. 기본적으로 SMF 작업자는 기준 성능 설정과 함께

gp3 Amazon EBS 볼륨을 사용합니다. 입력 파일이 크거나 작은 파일이 많은 워크로드의 경우 Amazon EBS 처리량 및 IOPS 설정을 늘려 전송 속도를 개선할 수 있습니다. AWS Command Line Interface () 를 사용하여 이러한 설정을 업데이트할 수 있습니다AWS CLI.

처리량(MiB/s)

볼륨에서 데이터를 읽거나 쓸 수 있는 속도입니다. gp3 볼륨의 기본값은 125MiB/s이고, 최대값은 1,000MiB/s입니다. 대용량 순차 파일 전송의 경우를 늘립니다.

IOPS

초당 입력/출력 작업 수입니다. gp3 볼륨의 기본값은 3,000 IOPS이고, 최대값은 16,000 IOPS입니다. 많은 작은 파일을 전송할 때를 늘립니다.

> **Note**
>
> Amazon EBS 처리량과 IOPS를 늘리면 플릿 비용이 증가합니다. 요금 정보는 Deadline Cloud 요금을 참조하세요.

를 사용하여 기존 플릿의 Amazon EBS 설정을 업데이트하려면 AWS CLI

- 다음 명령을 실행합니다.

```
aws deadline update-fleet \
  --farm-id farm-0123456789abcdef0 \
  --fleet-id fleet-0123456789abcdef0 \
  --configuration '{
    "serviceManagedEc2": {
      "instanceCapabilities": {
        "vCpuCount": {"min": 4},
        "memoryMiB": {"min": 8192},
        "osFamily": "linux",
        "cpuArchitectureType": "x86_64",
        "rootEbsVolume": {
          "sizeGiB": 250,
          "iops": 6000,
          "throughputMiB": 500
        }
      },
      "instanceMarketOptions": {"type": "spot"}
    }
```

```
  }'
```

작업 출력 다운로드

작업이 완료되면 Deadline Cloud CLI 또는 AWS Deadline Cloud Monitor(Deadline Cloud Monitor)를 사용하여 출력 파일을 다운로드합니다.

```
deadline job download-output --job-id job-0123456789abcdef0
```

작업이 완료되면 출력을 자동으로 다운로드하려면 Deadline Cloud 사용 설명서의 자동 다운로드를 참조하세요.

작업자에 사용자 지정 소프트웨어 배포 및 구성

AWS Deadline Cloud는 작업자에 사용자 지정 소프트웨어, 플러그인 및 도구를 배포하고 구성하는 여러 방법을 제공합니다. 선택하는 방법은 관리자 권한이 필요한지 여부, 소프트웨어 변경 빈도, 소프트웨어를 모든 작업에서 사용할 수 있는지 아니면 특정 작업에서만 사용할 수 있는지 여부 등 요구 사항에 따라 달라집니다.

배포 방법 선택

다음 표를 사용하여 사용 사례에 적합한 배포 방법을 선택합니다.

기준	대기열 환경	호스트 구성 스크립트	사용자 지정 conda 패키지
필요한 관리자 권한	아니요	예	아니요
실행 시	세션 시작	작업자 시작	세션 시작
Scope	대기열 또는 작업당	플릿의 모든 작업자	대기열 또는 작업당
작업 제출로 제어할 수 있음	예	아니요	예
설정 복잡성	낮음	중간	높음
최적의 용도	간단한 플러그인, 스크립트, 환경 변수	시스템 드라이버, Docker, 스토리지 마운트	종속성이 있는 복잡한 애플리케이션

빠른 결정 가이드:

- 관리자 또는 루트 권한이 필요합니까? 호스트 구성 스크립트를 사용합니다.
- 관리자 권한이 없는 간단한 플러그인 또는 스크립트 대기열 환경을 사용합니다.
- 버전 관리가 필요한 복잡한 애플리케이션 사용자 지정 conda 패키지를 생성합니다.

대기열 환경을 사용하여 작업 구성

AWS Deadline Cloud는 대기열 환경을 사용하여 작업자에 소프트웨어를 구성합니다. 환경을 사용하면 세션의 모든 작업에 대해 설정 및 제거와 같은 시간이 많이 걸리는 작업을 한 번 수행할 수 있습니다. 세션을 시작하거나 중지할 때 작업자에서 실행할 작업을 정의합니다. 대기열에 대한 환경, 대기열에서 실행되는 작업 및 작업에 대한 개별 단계를 구성할 수 있습니다.

환경을 대기열 환경 또는 작업 환경으로 정의합니다. Deadline Cloud 콘솔 또는 deadline:CreateQueueEnvironment 작업을 사용하여 대기열 환경을 생성하고 제출하는 작업의 작업 템플릿에서 작업 환경을 정의합니다. 환경에 대한 Open Job Description(OpenJD) 사양을 따릅니다. 자세한 내용은 GitHub의 OpenJD 사양에서 <Environment>를 참조하세요.

name 및 외에도 description각 환경에는 호스트에서 환경을 정의하는 두 개의 필드가 포함되어 있습니다. 스크립트는 다음과 같습니다.

- script -이 환경이 작업자에서 실행될 때 수행되는 작업입니다.
- variables - 환경에 들어갈 때 설정되는 환경 변수 이름/값 페어 세트입니다.

script 또는 중 하나 이상을 설정해야 합니다variables.

작업 템플릿에서 둘 이상의 환경을 정의할 수 있습니다. 각 환경은 템플릿에 나열된 순서대로 적용됩니다. 이를 사용하여 환경의 복잡성을 관리할 수 있습니다.

Deadline Cloud의 기본 대기열 환경은 conda 패키지 관리자를 사용하여 환경에 소프트웨어를 로드하지만 다른 패키지 관리자를 사용할 수 있습니다. 기본 환경은 로드해야 하는 소프트웨어를 지정하기 위해 두 개의 파라미터를 정의합니다. 이러한 변수는 Deadline Cloud에서 제공하는 제출자가 설정하지만 기본 환경을 사용하는 자체 스크립트 및 애플리케이션에서 설정할 수 있습니다. 스크립트는 다음과 같습니다.

- CondaPackages - 작업에 설치할 conda 패키지 일치 사양의 공백으로 구분된 목록입니다. 예를 들어 Blender 제출자는를 Blender 3.6의 렌더링 프레임blender=3.6에 추가합니다.
- CondaChannels - 패키지를 설치할 conda 채널의 공백으로 구분된 목록입니다. 서비스 관리형 플릿의 경우 deadline-cloud 채널에서 패키지가 설치됩니다. 다른 채널을 추가할 수 있습니다.

OpenJD 대기열 환경을 사용하여 작업 환경 제어

대기열 환경을 사용하여 렌더링 작업에 대한 사용자 지정 환경을 정의할 수 있습니다. 대기열 환경은 특정 대기열에서 실행되는 작업에 대한 환경 변수, 파일 매핑 및 기타 설정을 제어하는 템플릿입니

다. 이를 통해 대기열에 제출된 작업의 실행 환경을 워크로드의 요구 사항에 맞게 조정할 수 있습니다. AWS Deadline Cloud는 Open Job Description(OpenJD) 환경을 적용할 수 있는 세 가지 중첩 레벨인 대기열, 작업 및 단계를 제공합니다. 대기열 환경을 정의하면 다양한 유형의 작업에 대해 일관되고 최적화된 성능을 보장하고, 리소스 할당을 간소화하고, 대기열 관리를 간소화할 수 있습니다.

대기열 환경은 AWS 관리 콘솔에서 또는를 사용하여 AWS 계정의 대기열에 연결하는 템플릿입니다 AWS CLI. 대기열에 대해 하나의 환경을 생성하거나 실행 환경을 생성하기 위해를 적용한 여러 대기열 환경을 생성할 수 있습니다. 이 접근 방식을 사용하면 단계별로 환경을 생성하고 테스트하여 작업에 올바르게 작동하는지 확인할 수 있습니다.

작업 및 단계 환경은 대기열에서 작업을 생성하는 데 사용하는 작업 템플릿에 정의됩니다. OpenJD 구문은 이러한 다양한 형식의 환경에서 동일합니다. 이 섹션에서는 작업 템플릿 내에 이를 보여줍니다.

주제

- 대기열 환경에서 환경 변수 설정
- 대기열 환경에서 경로 설정
- 대기열 환경에서 백그라운드 데몬 프로세스 실행

대기열 환경에서 환경 변수 설정

많은 애플리케이션 및 프레임워크는 환경 변수를 사용하여 기능 설정, 로깅 수준 및 표시 구성을 제어합니다. Open Job Description(OpenJD) 환경을 사용하여 범위 내의 모든 태스크 명령이 상속하는 환경 변수를 설정할 수 있습니다.

환경 변수 범위

AWS Deadline Cloud는 대기열에 연결하는 대기열 환경의 환경 변수를 적용합니다. 작업 템플릿 내에서 OpenJD 환경을 사용하여 작업 및 단계 수준에서 환경 변수를 정의할 수도 있습니다. 더 좁은 범위에서 정의된 변수는 더 넓은 범위에서 동일한 이름의 변수를 재정의합니다.

- 대기열 환경 - Deadline Cloud의 대기열에 연결하는 템플릿입니다. 변수는 대기열에 제출된 모든 작업에 적용됩니다. 고정 값에 대한 `variables` 맵으로 변수를 설정하거나 동적 값에 스크립트를 사용할 수 있습니다.
- 작업 환경 - 작업 템플릿의 `jobEnvironments`에 정의되어 있습니다. 변수는 작업의 모든 단계와 작업에 적용됩니다. 작업 수준 변수는 동일한 이름의 대기열 수준 변수를 재정의합니다.
- 단계 환경 - 작업 템플릿의 `stepEnvironments`에 정의되어 있습니다. 변수는 해당 단계의 태스크에만 적용됩니다. 단계 수준 변수는 동일한 이름의 작업 수준 또는 대기열 수준 변수를 재정의합니다.

대기열 환경에서 변수 설정

고정 값에 대한 variables 맵을 사용하거나 동적 값에 대한 onEnter 작업과 script 함께를 사용하여 대기열 환경에서 환경 변수를 설정할 수 있습니다.

다음 대기열 환경 템플릿은 variables 맵을 사용하여 QT_QPA_PLATFORM 변수를 로 설정offscreen하므로 Qt 프레임워크를 사용하는 애플리케이션이 대화형 디스플레이 없이 작업자 호스트에서 실행될 수 있습니다.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  variables:
    QT_QPA_PLATFORM: offscreen
```

가상 환경 수정 PATH 또는 활성화와 같은 동적 값의 경우 형식으로 줄을 인쇄하여 stdout하는 스크립트openjd_env: *VAR*=*value*를 사용합니다. openjd_env: 접두사는 필수입니다. 접두사 없이 export, 또는 기타 echo쉘 메커니즘을 사용하면 변수가 작업 및 작업에 전파되지 않습니다.

다음 대기열 환경 템플릿은 스크립트를 사용하여 QT_QPA_PLATFORM 변수를 설정합니다.

```
specificationVersion: 'environment-2023-09'
environment:
  name: QtOffscreen
  script:
    actions:
      onEnter:
        command: bash
        args:
        - "{{Env.File.Enter}}"
    embeddedFiles:
    - name: Enter
      type: TEXT
      data: |
        #!/bin/env bash
        set -euo pipefail
        echo "openjd_env: QT_QPA_PLATFORM=offscreen"
```

대기열 환경을 대기열에 연결하려면 Deadline Cloud 콘솔 또는를 사용합니다 AWS CLI. 자세한 내용은 AWS Deadline Cloud 사용 설명서의 대기열 환경 생성을 참조하세요. 다음 AWS CLI 명령은 템플릿 파일에서 대기열 환경을 생성합니다.

```
aws deadline create-queue-environment \
    --farm-id FARM_ID \
    --queue-id QUEUE_ID \
    --priority 1 \
    --template-type YAML \
    --template file://my-queue-env.yaml
```

conda 가상 환경 생성 및 활성화와 같은 보다 복잡한 예제는 GitHub의 Deadline Cloud 대기열 환경 샘플을 참조하세요.

작업 템플릿에서 변수 설정

작업 템플릿에서 jobEnvironments 또는 stepEnvironments 항목에 variables 맵을 추가합니다. 각 항목은 키-값 페어로, 여기서 키는 변수 이름이고 값은 변수 값입니다.

다음 작업 템플릿은 QT_QPA_PLATFORM 환경 변수를 로 설정offscreen하여 Qt Framework를 사용하는 애플리케이션이 대화형 디스플레이 없이 작업자 호스트에서 실행되도록 합니다.

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    QT_QPA_PLATFORM: offscreen
```

단일 환경 정의에서 여러 변수를 설정할 수 있습니다.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_PROJECT_ID: my-project-id
    JOB_ENDPOINT_URL: https://my-host-name/my/path
    QT_QPA_PLATFORM: offscreen
```

{{Param.*ParameterName*}} 구문을 사용하여 변수 값의 작업 파라미터를 참조할 수 있습니다.

```
jobEnvironments:
- name: JobEnv
  variables:
```

```
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
```

특정 단계의 작업 수준 변수를 재정의하려면 변수 이름이 동일한 `stepEnvironments` 항목을 정의합니다. 다음 예제에서는 `JOB_PROJECT_ID` 작업 수준에서 값을 로 정의`project-12`한 다음 단계 수준에서 값을 로 재정의합니다`step-project-12`. 단계의 작업은 단계 수준 값을 사용합니다.

```
specificationVersion: 'jobtemplate-2023-09'
name: MyJob
jobEnvironments:
- name: JobEnv
  variables:
    JOB_PROJECT_ID: project-12
steps:
- name: MyStep
  stepEnvironments:
  - name: StepEnv
    variables:
      JOB_PROJECT_ID: step-project-12
```

시도: 환경 변수 샘플 실행

Deadline Cloud 샘플 리포지토리에는 환경 변수 설정 및 보기를 보여주는 작업 번들이 포함되어 있습니다. 샘플 작업 템플릿은 작업 및 단계 수준에서 변수를 정의한 다음 병합된 결과를 인쇄하는 작업을 실행합니다. 다음 절차에 따라 샘플을 실행하고 결과를 검사합니다.

사전 조건

1. 대기열 및 연결된 Linux 플릿이 있는 Deadline Cloud 팜이 없는 경우 Deadline Cloud 콘솔의 안내 온보딩 환경에 따라 기본 설정으로 Deadline Cloud 팜을 생성합니다.
2. 워크스테이션에 Deadline Cloud CLI 및 AWS Deadline Cloud 모니터가 없는 경우 Deadline Cloud 제출자 설정의 단계를 따릅니다.
3. `git`를 사용하여 Deadline Cloud 샘플 GitHub 리포지토리를 복제합니다.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
cd deadline-cloud-samples/job_bundles
```

샘플 실행

1. Deadline Cloud CLI를 사용하여 `job_env_vars` 샘플을 제출합니다.

```
deadline bundle submit job_env_vars
```

2. Deadline Cloud 모니터에서 진행 상황을 모니터링할 새 작업을 선택합니다. 대기열과 연결된 Linux 플릿에 작업자를 사용할 수 있게 되면 몇 초 내에 작업이 완료됩니다. 작업을 선택한 다음 작업 패널의 오른쪽 상단 메뉴에서 로그 보기를 선택합니다.

세션 작업과 정의 비교

로그 보기에는 세 가지 세션 작업이 표시됩니다. 텍스트 편집기에서 job_env_vars/template.yaml 파일을 열어 각 작업을 작업 템플릿의 정의와 비교합니다.

1. JobEnv 세션 시작 작업을 선택합니다. 로그 출력에는 설정 중인 작업 수준 환경 변수가 표시됩니다.

```
Setting: JOB_VERBOSITY=MEDIUM
Setting: JOB_EXAMPLE_PARAM=An example parameter value
Setting: JOB_PROJECT_ID=project-12
Setting: JOB_ENDPOINT_URL=https://internal-host-name/some/path
Setting: QT_QPA_PLATFORM=offscreen
```

   작업 템플릿의 다음 줄은이 환경을 정의합니다.

```
jobEnvironments:
- name: JobEnv
  variables:
    JOB_VERBOSITY: MEDIUM
    JOB_EXAMPLE_PARAM: "{{Param.ExampleParam}}"
    JOB_PROJECT_ID: project-12
    JOB_ENDPOINT_URL: https://internal-host-name/some/path
    QT_QPA_PLATFORM: offscreen
```

2. StepEnv 세션 시작 작업을 선택합니다. 로그 출력에는 재정의된를 포함한 단계 수준 변수가 표시됩니다JOB_PROJECT_ID.

```
Setting: STEP_VERBOSITY=HIGH
Setting: JOB_PROJECT_ID=step-project-12
```

   작업 템플릿의 다음 줄은이 환경을 정의합니다.

```
stepEnvironments:
```

```
- name: StepEnv
  variables:
    STEP_VERBOSITY: HIGH
    JOB_PROJECT_ID: step-project-12
```

3. 작업 실행 세션 작업을 선택합니다. 로그 출력에는 작업에 사용할 수 있는 병합된 환경 변수가 표시됩니다. 는 단계 수준 값를 JOB_PROJECT_ID 사용합니다step-project-12.

```
Environment variables starting with JOB_*:
JOB_ENDPOINT_URL=https://internal-host-name/some/path
JOB_EXAMPLE_PARAM='An example parameter value'
JOB_PROJECT_ID=step-project-12
JOB_VERBOSITY=MEDIUM

Environment variables starting with STEP_*:
STEP_VERBOSITY=HIGH
```

대기열 환경에서 경로 설정

OpenJD 환경을 사용하여 환경에서 새 명령을 제공합니다. 먼저 스크립트 파일이 포함된 디렉터리를 생성한 다음 해당 디렉터리를 PATH 환경 변수에 추가하여 스크립트의 실행 파일이 매번 디렉터리 경로를 지정하지 않고도 실행할 수 있도록 합니다. 환경 정의의 변수 목록은 변수를 수정하는 방법을 제공하지 않으므로 대신 스크립트를 실행하여 수정합니다. 스크립트는 사물을 설정하고를 수정PATH한 후 명령을 사용하여 변수를 OpenJD 런타임으로 내보냅니다echo "openjd_env: PATH=$PATH".

사전 조건

다음 단계를 수행하여 Deadline Cloud 샘플 github 리포지토리의 환경 변수를 사용하여 샘플 작업 번들을 실행합니다.

1. 대기열 및 연결된 Linux 플릿이 있는 Deadline Cloud 팜이 없는 경우 Deadline Cloud 콘솔의 안내 온보딩 환경에 따라 기본 설정으로 Deadline Cloud 팜을 생성합니다.
2. 워크스테이션에 Deadline Cloud CLI 및 Deadline Cloud 모니터가 없는 경우 사용 설명서의 Deadline Cloud 제출자 설정의 단계를 따르세요.
3. git를 사용하여 Deadline Cloud 샘플 GitHub 리포지토리를 복제합니다.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

경로 샘플 실행

1. Deadline Cloud CLI를 사용하여 `job_env_with_new_command` 샘플을 제출합니다.

```
$ deadline bundle submit job_env_with_new_command
Submitting to Queue: MySampleQueue
...
```

2. Deadline Cloud 모니터에 새 작업이 표시되고 진행 상황을 모니터링할 수 있습니다. 대기열과 연결된 Linux플릿에 작업의 작업을 실행할 수 있는 작업자가 있으면 작업이 몇 초 내에 완료됩니다. 작업을 선택한 다음 작업 패널의 오른쪽 상단 메뉴에서 로그 보기 옵션을 선택합니다.

   오른쪽에는 RandomSleepCommand 시작과 작업 실행이라는 두 가지 세션 작업이 있습니다. 창 중앙의 로그 뷰어는 오른쪽에서 선택한 세션 작업에 해당합니다.

세션 작업과 정의 비교

이 섹션에서는 Deadline Cloud 모니터를 사용하여 세션 작업을 작업 템플릿에 정의된 위치와 비교합니다. 이전 섹션에서 계속됩니다.

텍스트 편집기에서 job_env_with_new_command/template.yaml 파일을 엽니다. 세션 작업을 작업 템플릿에 정의된 위치와 비교합니다.

1. Deadline Cloud 모니터에서 RandomSleepCommand 세션 시작 작업을 선택합니다. 다음과 같이 로그 출력이 표시됩니다.

```
 2024/07/16 17:25:32-07:00
 2024/07/16 17:25:32-07:00 ==============================================
 2024/07/16 17:25:32-07:00 --------- Entering Environment: RandomSleepCommand
 2024/07/16 17:25:32-07:00 ==============================================
 2024/07/16 17:25:32-07:00 ----------------------------------------------
 2024/07/16 17:25:32-07:00 Phase: Setup
 2024/07/16 17:25:32-07:00 ----------------------------------------------
 2024/07/16 17:25:32-07:00 Writing embedded files for Environment to disk.
 2024/07/16 17:25:32-07:00 Mapping: Env.File.Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
 2024/07/16 17:25:32-07:00 Mapping: Env.File.SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
 2024/07/16 17:25:32-07:00 Wrote: Enter -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpbt8j_c3f
 2024/07/16 17:25:32-07:00 Wrote: SleepScript -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmperastlp4
```

```
 2024/07/16 17:25:32-07:00 ------------------------------------------
 2024/07/16 17:25:32-07:00 Phase: Running action
 2024/07/16 17:25:32-07:00 ------------------------------------------
 2024/07/16 17:25:32-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpbwrquq5u.sh
 2024/07/16 17:25:32-07:00 Command started as pid: 2205
 2024/07/16 17:25:32-07:00 Output:
 2024/07/16 17:25:33-07:00 openjd_env: PATH=/sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/bin:/opt/conda/condabin:/home/job-
user/.local/bin:/home/job-user/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/var/lib/snapd/snap/bin
 No newer logs at this moment.
```

작업 템플릿의 다음 줄이이 작업을 지정했습니다.

```
jobEnvironments:
- name: RandomSleepCommand
  description: Adds a command 'random-sleep' to the environment.
  script:
    actions:
      onEnter:
        command: bash
        args:
        - "{{Env.File.Enter}}"
    embeddedFiles:
    - name: Enter
      type: TEXT
      data: |
        #!/bin/env bash
        set -euo pipefail

        # Make a bin directory inside the session's working directory for providing
 new commands
        mkdir -p '{{Session.WorkingDirectory}}/bin'

        # If this bin directory is not already in the PATH, then add it
        if ! [[ ":$PATH:" == *':{{Session.WorkingDirectory}}/bin:'* ]]; then
          export "PATH={{Session.WorkingDirectory}}/bin:$PATH"

          # This message to Open Job Description exports the new PATH value to the
 environment
          echo "openjd_env: PATH=$PATH"
        fi
```

```
        # Copy the SleepScript embedded file into the bin directory
        cp '{{Env.File.SleepScript}}' '{{Session.WorkingDirectory}}/bin/random-
sleep'
        chmod u+x '{{Session.WorkingDirectory}}/bin/random-sleep'
    - name: SleepScript
      type: TEXT
      runnable: true
      data: |
        ...
```

2. Deadline Cloud 모니터에서 StepEnv 세션 시작 작업을 선택합니다. 다음과 같이 로그 출력이 표시됩니다.

```
2024/07/16 17:25:33-07:00
2024/07/16 17:25:33-07:00 ================================================
2024/07/16 17:25:33-07:00 --------- Running Task
2024/07/16 17:25:33-07:00 ================================================
2024/07/16 17:25:33-07:00 ------------------------------------------------
2024/07/16 17:25:33-07:00 Phase: Setup
2024/07/16 17:25:33-07:00 ------------------------------------------------
2024/07/16 17:25:33-07:00 Writing embedded files for Task to disk.
2024/07/16 17:25:33-07:00 Mapping: Task.File.Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
2024/07/16 17:25:33-07:00 Wrote: Run -> /sessions/session-
ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/embedded_filesf3tq_1os/tmpdrwuehjf
2024/07/16 17:25:33-07:00 ------------------------------------------------
2024/07/16 17:25:33-07:00 Phase: Running action
2024/07/16 17:25:33-07:00 ------------------------------------------------
2024/07/16 17:25:33-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-ab132a51b9b54d5da22cbe839dd946baaw1c8hk5/tmpz81iaqfw.sh
2024/07/16 17:25:33-07:00 Command started as pid: 2256
2024/07/16 17:25:33-07:00 Output:
2024/07/16 17:25:34-07:00 + random-sleep 12.5 27.5
2024/07/16 17:26:00-07:00 Sleeping for duration 26.90
2024/07/16 17:26:00-07:00 ------------------------------------------------
2024/07/16 17:26:00-07:00 Uploading output files to Job Attachments
2024/07/16 17:26:00-07:00 ------------------------------------------------
```

3. 작업 템플릿의 다음 줄이이 작업을 지정했습니다.

```
steps:
- name: EnvWithCommand
```

```
  script:
    actions:
      onRun:
        command: bash
        args:
        - '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      type: TEXT
      data: |
        set -xeuo pipefail

        # Run the script installed into PATH by the job environment
        random-sleep 12.5 27.5
  hostRequirements:
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

대기열 환경에서 백그라운드 데몬 프로세스 실행

많은 렌더링 사용 사례에서 애플리케이션 및 장면 데이터를 로드하는 데 상당한 시간이 걸릴 수 있습니다. 작업이 모든 프레임에 대해 다시 로드하는 경우 대부분의 시간을 오버헤드에 소비합니다. 애플리케이션을 백그라운드 데몬 프로세스로 한 번 로드하고 장면 데이터를 로드한 다음 프로세스 간 통신(IPC)을 통해 명령을 전송하여 렌더링을 수행할 수 있는 경우가 많습니다.

많은 오픈 소스 Deadline Cloud 통합이이 패턴을 사용합니다. Open Job Description 프로젝트는 지원되는 모든 운영 체제에서 강력한 IPC 패턴을 갖춘 어댑터 런타임 라이브러리를 제공합니다.

이 패턴을 보여주기 위해 Python 및 bash 코드를 사용하여 백그라운드 데몬을 구현하고 작업과 통신할 IPC를 구현하는 독립형 샘플 작업 번들이 있습니다. 데몬은 Python에서 구현되며 작업을 처리할 시기에 대한 POSIX SIGUSR1 신호를 수신합니다. 작업 세부 정보는 특정 JSON 파일의 데몬으로 전달되고 작업 실행 결과는 다른 JSON 파일로 반환됩니다.

사전 조건

Deadline Cloud samples github 리포지토리에서 데몬 프로세스를 사용하여 샘플 작업 번들을 실행하려면 다음 단계를 수행합니다.

1. 대기열 및 연결된 Linux 플릿이 있는 Deadline Cloud 팜이 없는 경우 Deadline Cloud 콘솔의 안내 온보딩 환경에 따라 기본 설정으로 Deadline Cloud 팜을 생성합니다.
2. 워크스테이션에 Deadline Cloud CLI 및 Deadline Cloud 모니터가 없는 경우 사용 설명서의 Deadline Cloud 제출자 설정의 단계를 따르세요.
3. `git`를 사용하여 Deadline Cloud 샘플 GitHub 리포지토리를 복제합니다.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
 Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

데몬 샘플 실행

1. Deadline Cloud CLI를 사용하여 `job_env_daemon_process` 샘플을 제출합니다.

```
 git clone https://github.com/aws-deadline/deadline-cloud-samples.git
Cloning into 'deadline-cloud-samples'...
 ...
cd deadline-cloud-samples/job_bundles
```

2. Deadline Cloud 모니터 애플리케이션에서는 새 작업이 표시되고 진행 상황을 모니터링할 수 있습니다. 대기열과 연결된 Linux플릿에 작업의 작업을 실행할 수 있는 작업자가 있으면 약 1분 후에 완료됩니다. 작업 중 하나를 선택한 상태에서 작업 패널의 오른쪽 상단 메뉴에서 로그 보기 옵션을 선택합니다.

   오른쪽에는 DaemonProcess 시작과 Task 실행이라는 두 가지 세션 작업이 있습니다. 창 중앙의 로그 뷰어는 오른쪽에서 선택한 세션 작업에 해당합니다.

   모든 작업에 대한 로그 보기 옵션을 선택합니다. 타임라인에는 세션의 일부로 실행된 나머지 작업과 환경을 종료한 `Shut down DaemonProcess` 작업이 표시됩니다.

데몬 로그 보기

1. 이 섹션에서는 Deadline Cloud 모니터를 사용하여 세션 작업을 작업 템플릿에 정의된 위치와 비교합니다. 이전 섹션에서 계속됩니다.

   텍스트 편집기에서 job_env_daemon_process/template.yaml 파일을 엽니다. 세션 작업을 작업 템플릿에 정의된 위치와 비교합니다.

2. Deadline Cloud Monitor에서 Launch DaemonProcess 세션 작업을 선택합니다. 다음과 같이 로그 출력이 표시됩니다.

```
 2024/07/17 16:27:20-07:00
 2024/07/17 16:27:20-07:00 ==============================================
 2024/07/17 16:27:20-07:00 --------- Entering Environment: DaemonProcess
 2024/07/17 16:27:20-07:00 ==============================================
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Phase: Setup
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Writing embedded files for Environment to disk.
 2024/07/17 16:27:20-07:00 Mapping: Env.File.Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Mapping: Env.File.Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
 2024/07/17 16:27:20-07:00 Mapping: Env.File.DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
 2024/07/17 16:27:20-07:00 Wrote: Enter -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/enter-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Wrote: Exit -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/exit-daemon-
process-env.sh
 2024/07/17 16:27:20-07:00 Wrote: DaemonScript -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
script.py
 2024/07/17 16:27:20-07:00 Wrote: DaemonHelperFunctions -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Phase: Running action
 2024/07/17 16:27:20-07:00 ----------------------------------------------
 2024/07/17 16:27:20-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmp_u8slys3.sh
 2024/07/17 16:27:20-07:00 Command started as pid: 2187
 2024/07/17 16:27:20-07:00 Output:
```

```
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_LOG=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_PID=2223
 2024/07/17 16:27:21-07:00 openjd_env: DAEMON_BASH_HELPER_SCRIPT=/sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/daemon-
helper-functions.sh
```

작업 템플릿의 다음 줄이이 작업을 지정했습니다.

```
  stepEnvironments:
  - name: DaemonProcess
    description: Runs a daemon process for the step's tasks to share.
    script:
      actions:
        onEnter:
          command: bash
          args:
          - "{{Env.File.Enter}}"
        onExit:
          command: bash
          args:
          - "{{Env.File.Exit}}"
      embeddedFiles:
      - name: Enter
        filename: enter-daemon-process-env.sh
        type: TEXT
        data: |
          #!/bin/env bash
          set -euo pipefail

          DAEMON_LOG='{{Session.WorkingDirectory}}/daemon.log'
          echo "openjd_env: DAEMON_LOG=$DAEMON_LOG"
          nohup python {{Env.File.DaemonScript}} > $DAEMON_LOG 2>&1 &
          echo "openjd_env: DAEMON_PID=$!"
          echo "openjd_env:
 DAEMON_BASH_HELPER_SCRIPT={{Env.File.DaemonHelperFunctions}}"

          echo 0 > 'daemon_log_cursor.txt'
    ...
```

3. Deadline Cloud Monitor에서 작업 실행: N 세션 작업 중 하나를 선택합니다. 다음과 같이 로그 출력이 표시됩니다.

```
2024/07/17 16:27:22-07:00
 2024/07/17 16:27:22-07:00 ================================================
 2024/07/17 16:27:22-07:00 --------- Running Task
 2024/07/17 16:27:22-07:00 ================================================
 2024/07/17 16:27:22-07:00 Parameter values:
 2024/07/17 16:27:22-07:00 Frame(INT) = 2
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Phase: Setup
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Writing embedded files for Task to disk.
 2024/07/17 16:27:22-07:00 Mapping: Task.File.Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
 2024/07/17 16:27:22-07:00 Wrote: Run -> /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/embedded_fileswy00x5ra/run-task.sh
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Phase: Running action
 2024/07/17 16:27:22-07:00 ------------------------------------------------
 2024/07/17 16:27:22-07:00 Running command sudo -u job-user -i setsid -w /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/tmpv4obfkhn.sh
 2024/07/17 16:27:22-07:00 Command started as pid: 2301
 2024/07/17 16:27:22-07:00 Output:
 2024/07/17 16:27:23-07:00 Daemon PID is 2223
 2024/07/17 16:27:23-07:00 Daemon log file is /sessions/
session-972e21d98dde45e59c7153bd9258a64dohwg4yg1/daemon.log
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 === Previous output from daemon
 2024/07/17 16:27:23-07:00 ===
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 Sending command to daemon
 2024/07/17 16:27:23-07:00 Received task result:
 2024/07/17 16:27:23-07:00 {
 2024/07/17 16:27:23-07:00   "result": "SUCCESS",
 2024/07/17 16:27:23-07:00   "processedTaskCount": 1,
 2024/07/17 16:27:23-07:00   "randomValue": 0.2578537967668988,
 2024/07/17 16:27:23-07:00   "failureRate": 0.1
 2024/07/17 16:27:23-07:00 }
 2024/07/17 16:27:23-07:00
 2024/07/17 16:27:23-07:00 === Daemon log from running the task
 2024/07/17 16:27:23-07:00 Loading the task details file
 2024/07/17 16:27:23-07:00 Received task details:
 2024/07/17 16:27:23-07:00 {
 2024/07/17 16:27:23-07:00  "pid": 2329,
 2024/07/17 16:27:23-07:00  "frame": 2
```

```
2024/07/17 16:27:23-07:00 }
2024/07/17 16:27:23-07:00 Processing frame number 2
2024/07/17 16:27:23-07:00 Writing result
2024/07/17 16:27:23-07:00 Waiting until a USR1 signal is sent...
2024/07/17 16:27:23-07:00 ===
2024/07/17 16:27:23-07:00
2024/07/17 16:27:23-07:00 ------------------------------------------------
2024/07/17 16:27:23-07:00 Uploading output files to Job Attachments
2024/07/17 16:27:23-07:00 ------------------------------------------------
```

작업 템플릿의 다음 줄은이 작업을 지정한 것입니다. ``` 단계:

```
steps:
- name: EnvWithDaemonProcess
  parameterSpace:
    taskParameterDefinitions:
    - name: Frame
      type: INT
      range: "{{Param.Frames}}"

  stepEnvironments:
    ...

  script:
    actions:
      onRun:
        timeout: 60
        command: bash
        args:
        - '{{Task.File.Run}}'
    embeddedFiles:
    - name: Run
      filename: run-task.sh
      type: TEXT
      data: |
        # This bash script sends a task to the background daemon process,
        # then waits for it to respond with the output result.

        set -euo pipefail

        source "$DAEMON_BASH_HELPER_SCRIPT"

        echo "Daemon PID is $DAEMON_PID"
```

```
        echo "Daemon log file is $DAEMON_LOG"

        print_daemon_log "Previous output from daemon"

        send_task_to_daemon "{\"pid\": $$, \"frame\": {{Task.Param.Frame}} }"
        wait_for_daemon_task_result

        echo Received task result:
        echo "$TASK_RESULT" | jq .

        print_daemon_log "Daemon log from running the task"

  hostRequirements:
    attributes:
    - name: attr.worker.os.family
      anyOf:
      - linux
```

작업에 대한 애플리케이션 제공

대기열 환경을 사용하여 애플리케이션을 로드하여 작업을 처리할 수 있습니다. Deadline Cloud 콘솔을 사용하여 서비스 관리형 플릿을 생성할 때 conda 패키지 관리자를 사용하여 애플리케이션을 로드하는 대기열 환경을 생성할 수 있습니다.

다른 패키지 관리자를 사용하려면 해당 관리자에 대한 대기열 환경을 생성할 수 있습니다. Rez를 사용하는 예제는 섹션을 참조하세요다른 패키지 관리자 사용.

Deadline Cloud는 다양한 렌더링 애플리케이션을 환경에 로드할 수 있는 conda 채널을 제공합니다. Deadline Cloud가 디지털 콘텐츠 생성 애플리케이션에 제공하는 제출자를 지원합니다.

conda-forge용 소프트웨어를 로드하여 작업에 사용할 수도 있습니다. 다음 예제에서는 Deadline Cloud에서 제공하는 대기열 환경을 사용하여 작업을 실행하기 전에 애플리케이션을 로드하는 작업 템플릿을 보여줍니다.

주제

- conda 채널에서 애플리케이션 가져오기
- 다른 패키지 관리자 사용

conda 채널에서 애플리케이션 가져오기

원하는 소프트웨어를 설치하는 Deadline Cloud 작업자를 위한 사용자 지정 대기열 환경을 생성할 수 있습니다. 이 예제 대기열 환경은 콘솔에서 서비스 관리형 플릿에 사용하는 환경과 동일한 동작을 합니다. conda를 직접 실행하여 환경을 생성합니다.

환경은 작업자에서 실행되는 모든 Deadline Cloud 세션에 대해 새 conda 가상 환경을 생성한 다음 완료되면 환경을 삭제합니다.

Conda는 다운로드한 패키지를 캐시하므로 다시 다운로드할 필요가 없지만 각 세션은 모든 패키지를 환경에 연결해야 합니다.

환경은 Deadline Cloud가 작업자에 대한 세션을 시작할 때 실행되는 세 가지 스크립트를 정의합니다. 첫 번째 스크립트는 `onEnter` 작업이 호출될 때 실행됩니다. 나머지 두 개를 호출하여 환경 변수를 설정합니다. 스크립트 실행이 완료되면 지정된 모든 환경 변수가 설정된 상태에서 conda 환경을 사용할 수 있습니다.

예제의 최신 버전은 GitHub의 deadline-cloud-samples 리포지토리에서 conda_queue_env_console_equivalent.yaml을 참조하세요. deadline-cloud-samples

conda 채널에서 사용할 수 없는 애플리케이션을 사용하려는 경우 Amazon S3에서 conda 채널을 생성한 다음 해당 애플리케이션에 대한 자체 패키지를 빌드할 수 있습니다. 자세한 내용은 S3를 사용하여 conda 채널 생성 섹션을 참조하세요.

conda-forge에서 오픈 소스 라이브러리 가져오기

이 섹션에서는 `conda-forge` 채널에서 오픈 소스 라이브러리를 사용하는 방법을 설명합니다. 다음 예제는 `polars` Python 패키지를 사용하는 작업 템플릿입니다.

작업은 Deadline Cloud에 패키지를 가져올 위치를 알려주는 대기열 환경에 정의된 `CondaPackages` 및 `CondaChannels` 파라미터를 설정합니다.

파라미터를 설정하는 작업 템플릿의 섹션은 다음과 같습니다.

```
- name: CondaPackages
  description: A list of conda packages to install. The job expects a Queue Environment
 to handle this.
  type: STRING
  default: polars
- name: CondaChannels
  description: A list of conda channels to get packages from. The job expects a Queue
 Environment to handle this.
```

```
  type: STRING
  default: conda-forge
```

전체 예제 작업 템플릿의 최신 버전은 stage_1_self_contained_template/template.yaml을 참조하세요. conda 패키지를 로드하는 대기열 환경의 최신 버전은 GitHub의 deadline-cloud-samples 리포지토리에서 conda_queue_env_console_equivalent.yaml을 참조하세요. deadline-cloud-samples

기한 클라우드 채널Blender에서 가져오기

다음 예제는 `deadline-cloud` conda 채널Blender에서 가져오는 작업 템플릿을 보여줍니다. 이 채널은 Deadline Cloud가 디지털 콘텐츠 생성 소프트웨어에 제공하는 제출자를 지원하지만, 동일한 채널을 사용하여 자체 용도로 소프트웨어를 로드할 수 있습니다.

`deadline-cloud` 채널에서 제공하는 소프트웨어 목록은 AWS Deadline Cloud 사용 설명서의 기본 대기열 환경을 참조하세요.

이 작업은 대기열 환경에 정의된 `CondaPackages` 파라미터를 설정하여 Deadline Cloud에 환경 Blender으로 로드하도록 지시합니다.

파라미터를 설정하는 작업 템플릿의 섹션은 다음과 같습니다.

```
- name: CondaPackages
  type: STRING
  userInterface:
    control: LINE_EDIT
    label: Conda Packages
    groupLabel: Software Environment
  default: blender
  description: >
    Tells the queue environment to install Blender from the deadline-cloud conda
 channel.
```

전체 예제 작업 템플릿의 최신 버전은 blender_render/template.yaml을 참조하세요. conda 패키지를 로드하는 대기열 환경의 최신 버전은의 deadline-cloud-samples 리포지토리에서 conda_queue_env_console_equivalent.yaml을 참조하세요GitHub.

다른 패키지 관리자 사용

Deadline Cloud의 기본 패키지 관리자는 conda입니다. 와 같은 다른 패키지 관리자를 사용해야 하는 경우 Rez대신 패키지 관리자를 사용하는 스크립트가 포함된 사용자 지정 대기열 환경을 생성할 수 있습니다.

이 예제 대기열 환경은 콘솔에서 서비스 관리형 플릿에 사용하는 환경과 동일한 동작을 제공합니다. conda 패키지 관리자를 로 대체합니다Rez.

환경은 Deadline Cloud가 작업자에 대한 세션을 시작할 때 실행되는 세 가지 스크립트를 정의합니다. 첫 번째 스크립트는 `onEnter` 작업이 호출될 때 실행됩니다. 나머지 두 개를 호출하여 환경 변수를 설정합니다. 스크립트 실행이 완료되면 지정된 모든 Rez 환경 변수가 설정된 상태에서 환경을 사용할 수 있습니다.

이 예제에서는 Rez 패키지에 공유 파일 시스템을 사용하는 고객 관리형 플릿이 있다고 가정합니다.

예제의 최신 버전은의 deadline-cloud-samples 리포지토리에서 rez_queue_env.yaml을 참조하세요 GitHub.

S3를 사용하여 conda 채널 생성

작업이 `deadline-cloud` 또는 `conda-forge` 채널에서 사용할 수 없는 애플리케이션을 실행해야 하는 경우 사용자 지정 conda 채널을 호스팅하여 자체 패키지를 제공할 수 있습니다. AWS Deadline Cloud(Deadline Cloud) 콘솔에서 대기열을 생성하면 콘솔은 기본적으로 conda 대기열 환경을 추가합니다. 작업에 패키지를 사용할 수 있도록 하려면 대기열 환경에 사용자 지정 채널을 추가합니다.

conda 채널은 파일 시스템 또는 Amazon Simple Storage Service(Amazon S3) 버킷을 포함하여 다양한 방식으로 호스팅할 수 있는 정적 호스팅 콘텐츠입니다. Deadline Cloud 팜이 자산에 공유 파일 시스템을 사용하는 경우 해당 팜의 모든 경로를 채널 이름으로 사용할 수 있습니다. AWS Identity and Access Management (IAM) 권한을 사용하여 더 광범위한 액세스를 위해 Amazon S3 버킷에서 채널을 호스팅할 수 있습니다.

로컬에서 패키지를 빌드하고 테스트한 다음 채널에 게시할 수 있습니다. 로컬에서 패키지를 빌드하면 인프라 설정 없이 패키지 빌드 레시피에서 반복을 쉽게 시작할 수 있습니다. 또한 Deadline Cloud 패키지 빌드 대기열을 사용하여 패키지를 빌드하고 채널에 게시할 수 있습니다. 패키지 빌드 대기열은 여러 운영 체제 및 액셀러레이터 구성에 대한 패키지 유지 관리를 간소화합니다. 어디에서나 버전을 업데이트하고 패키지 빌드의 전체 세트를 제출할 수 있습니다.

스튜디오 및 Deadline Cloud 팜에 대한 채널을 여러 가지 방법으로 구성할 수 있습니다. 하나의 Amazon S3 채널을 사용하고 이를 사용하도록 모든 워크스테이션과 팜 호스트를 구성할 수 있습니다. 채널이 두 개 이상 있고 AWS DataSync (DataSync)를 사용하여 미러링을 설정할 수도 있습니다. 예를 들어 Deadline Cloud 패키지 빌드 대기열은 워크스테이션 및 온프레미스 팜 호스트를 위해 온프레미스에서 미러링되는 Amazon S3 채널에 게시할 수 있습니다.

주제

로컬에서 패키지 빌드 및 테스트

Amazon S3에 패키지를 게시하거나 Deadline Cloud 팜에서 CI/CD 자동화를 설정하기 전에 로컬 파일 시스템 채널을 사용하여 워크스테이션에서 conda 패키지를 빌드하고 테스트할 수 있습니다. 이 접근 방식을 사용하면 레시피를 로컬에서 빠르게 반복하고 패키지를 확인할 수 있습니다.

`rattler-build publish` 명령은 레시피를 빌드하고, 결과 패키지를 채널에 복사하고, 한 번에 채널을 인덱싱합니다. 로컬 파일 시스템 디렉터리를 대상으로 지정하면 디렉터리가 없는 경우가 자동으로 채널을 `rattler-build` 생성하고 초기화합니다.

다음 지침은 Blender의 Deadline Cloud 샘플 리포지토리의 4.5 샘플 레시피를 사용합니다GitHub. 샘플 리포지토리에서 다른 레시피를 대체하거나 자체 레시피를 사용할 수 있습니다.

사전 조건

시작하기 전에 워크스테이션에 다음 도구를 설치합니다.

- pixi - 패키지를 설치하고 테스트`rattler-build`하는 데 사용하는 패키지 관리자입니다. pixi.sh pixi를 설치합니다.
- 래틀러 빌드 - Deadline Cloud conda 레시피에서 사용하는 패키지 빌드 도구입니다. pixi를 설치한 후 다음 명령을 실행하여를 설치합니다`rattler-build`.

```
pixi global install rattler-build
```

- git - 샘플 리포지토리를 복제하는 데 필요합니다. 에서 용 Windowsgit는 일부 Windows 샘플 레시피에 필요한 `bash`쉘도 제공합니다. Windows

로컬 채널에 패키지 빌드 및 게시

이 절차에서는 Deadline Cloud 샘플 리포지토리를 복제하고 `rattler-build publish`를 사용하여 패키지를 빌드하고 로컬 파일 시스템 채널에 게시합니다.

> **Note**
>
> 대규모 애플리케이션은 소스 아카이브, 추출된 파일 및 빌드 출력을 위해 수십 GB의 여유 디스크 공간이 필요할 수 있습니다. 패키지 빌드 출력을 위해 사용 가능한 공간이 충분한 디스크를 사용해야 합니다.

로컬 채널에 패키지를 빌드하고 게시하려면

1. Deadline Cloud 샘플 리포지토리를 복제합니다.

   ```
   git clone https://github.com/aws-deadline/deadline-cloud-samples.git
   ```

2. 디렉터리를 `conda_recipes`로 변경합니다.

   ```
   cd deadline-cloud-samples/conda_recipes
   ```

3. 다음 명령을 실행하여 Blender 4.5 레시피를 빌드하고 패키지를 로컬 채널 디렉터리에 게시합니다.

   Linux 및에서 다음 명령을 macOS실행합니다.

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml \
       --to file://$HOME/my-conda-channel \
       --build-number=+1
   ```

   Windows (cmd)에서 다음 명령을 실행합니다.

   ```
   rattler-build publish blender-4.5/recipe/recipe.yaml ^
       --to file://%USERPROFILE%/my-conda-channel ^
       --build-number=+1
   ```

   `rattler-build publish` 명령은 다음 작업을 수행합니다.

   - 레시피에서 패키지를 빌드합니다.

- 디렉터리가 없는 경우 채널 디렉터리를 생성합니다.
- 패키지 파일을 채널에 복사합니다.
- 패키지 관리자가 패키지를 찾을 수 있도록 채널을 인덱싱합니다.

패키지 레시피가 conda-forge와 같은 특정 채널의 패키지에 의존하는 경우 명령에 `-c conda-forge`를 추가합니다.

> **빌드 번호 정보**
>
> `--build-number=+1` 옵션은 대상 채널에 이미 있는 빌드 번호를 기반으로 다음 빌드 번호를 자동으로 선택합니다. 채널에서 패키지를 덮어쓰지 않는 것이 가장 좋습니다. 패키지의 파일 이름이 동일한 경우 항상 새 빌드 번호로 빌드합니다. 를 사용하면 프로덕션 채널 또는 프로덕션을 미러링하는 스테이징 채널에 빌드할 때 이를 `--build-number=+1` 달성할 수 있습니다. 빌드 번호를 직접 제어하려면와 같은 특정 값으로 설정할 수 있습니다`--build-number=7`. 옵션을 생략하면는 `recipe.yaml` 파일에 정의된 빌드 번호를 `rattler-build` 사용합니다.

에 대한 자세한 내용은 래틀러 빌드 게시 설명서를 `rattler-build publish`참조하세요.

빌드 디버깅

빌드에 실패하면는 사용자가 조사할 수 있도록 빌드 디렉터리를 `rattler-build` 보존합니다. 다음 명령을 실행하여 모든 환경 변수가 빌드 중에 설정된 상태로 빌드 환경에서 대화형 셸을 엽니다.

```
rattler-build debug shell
```

디버그 쉘에서 파일을 수정하고, 개별 빌드 명령을 실행하고, 종속성을 추가하여 문제를 격리할 수 있습니다. 자세한 내용은 래틀러 빌드 설명서의 빌드 디버깅을 참조하세요.

패키지 테스트

패키지를 빌드하고 게시한 후 임시 pixi 프로젝트를 생성합니다. 프로젝트를 사용하여 로컬 채널에서 패키지를 설치하고 올바르게 작동하는지 확인합니다.

패키지를 테스트하려면

1. 임시 테스트 디렉터리를 생성하고 로컬 채널로 pixi 프로젝트를 초기화합니다.

Linux 및에서 다음 명령을 macOS실행합니다.

```
mkdir package-test-env
cd package-test-env
pixi init --channel file://$HOME/my-conda-channel
```

Windows (cmd)에서 다음 명령을 실행합니다.

```
mkdir package-test-env
cd package-test-env
pixi init --channel file://%USERPROFILE%/my-conda-channel
```

2. 프로젝트에 패키지를 추가합니다.

```
pixi add blender=4.5
```

3. 패키지가 올바르게 작동하는지 확인합니다.

```
pixi run blender --version
```

`pixi run` 명령은 프로젝트 디렉터리에 대한 conda 환경을 활성화하고 그 안에서 지정된 명령을 실행합니다. 환경은 프로젝트 디렉터리에 유지되므로 다른 터미널에서 동일한 `pixi run` 명령을 사용할 수 있습니다.

패키지에 만족하면 Amazon S3 conda 채널에 패키지를 게시하여 Deadline Cloud 작업자가 패키지를 설치할 수 있도록 할 수 있습니다. S3 conda 채널에 패키지 게시를 참조하세요.

채널에서 패키지 제거

lockfile은 해시로 특정 패키지를 참조하므로 프로덕션에 사용하는 채널에서 패키지를 제거하지 마세요. 패키지를 제거하면 해당 잠금 파일에서 환경을 다시 생성할 수 없습니다. 개발 및 테스트 채널의 경우 채널 디렉터리에서 `.conda` 파일을 삭제한 다음 채널을 다시 인덱싱하여 특정 패키지를 제거할 수 있습니다. 먼저를 설치합니다`rattler-index`.

```
pixi global install rattler-index
```

그런 다음 패키지 파일을 삭제하고 채널을 다시 인덱싱합니다.

Linux 및에서 다음 명령을 macOS실행합니다.

```
rm $HOME/my-conda-channel/linux-64/blender-4.5.0-hb0f4dca_1.conda
rattler-index fs $HOME/my-conda-channel
```

Windows (cmd)에서 다음 명령을 실행합니다.

```
del %USERPROFILE%\my-conda-channel\win-64\blender-4.5.0-hb0f4dca_1.conda
rattler-index fs %USERPROFILE%\my-conda-channel
```

패키지 파일은 `linux-64`, `win-64`또는와 같은 플랫폼별 하위 디렉터리에 저장됩니다`osx-arm64`. 이러한 하위 디렉터리의 내용을 나열하여 제거하려는 패키지의 정확한 파일 이름을 찾습니다.

정리

테스트 후 테스트 프로젝트와 로컬 채널을 제거할 수 있습니다.

테스트 리소스를 정리하려면

1. 테스트 프로젝트 디렉터리를 제거합니다.

   Linux 및에서 다음 명령을 macOS실행합니다.

   ```
   rm -rf package-test-env
   ```

   Windows (cmd)에서 다음 명령을 실행합니다.

   ```
   rmdir /s /q package-test-env
   ```

2. 로컬 conda 채널 디렉터리를 제거합니다.

   Linux 및에서 다음 명령을 macOS실행합니다.

   ```
   rm -rf $HOME/my-conda-channel
   ```

   Windows (cmd)에서 다음 명령을 실행합니다.

   ```
   rmdir /s /q %USERPROFILE%\my-conda-channel
   ```

3. (선택 사항) 빌드된 패키지 파일이 포함된 `rattler-build` 출력 디렉터리를 제거합니다.

   Linux 및에서 다음 명령을 macOS실행합니다.

```
rm -rf deadline-cloud-samples/conda_recipes/output
```

Windows (cmd)에서 다음 명령을 실행합니다.

```
rmdir /s /q deadline-cloud-samples\conda_recipes\output
```

Amazon S3 conda 채널에 패키지 게시

Amazon Simple Storage Service(Amazon S3) 버킷에 conda 패키지를 게시하여 AWS Deadline Cloud(Deadline Cloud) 작업자가 실행 작업을 위해 설치할 수 있습니다. `rattler-build publish` 명령은 로컬 파일 시스템 채널과 동일한 방식으로 Amazon S3에서 작동합니다. 명령은 레시피를 빌드하고 결과를 게시하거나 이미 빌드한 패키지 파일을 게시할 수 있습니다. 두 경우 모두 명령은 패키지를 버킷에 업로드하고 한 단계로 채널을 인덱싱합니다.

`rattler-build publish` 명령은 표준 자격 증명 체인을 AWS 사용하여를 인증하므로 모든 AWS 도구처럼 AWS 구성을 사용합니다. 자격 증명 구성에 대한 자세한 내용은 AWS Command Line Interface (AWS CLI) 사용 설명서의 구성 및 자격 증명 파일 설정을 참조하세요.

사전 조건

Amazon S3에 패키지를 게시하기 전에 다음 사전 조건을 완료합니다.

- pixi 및rattler-build - pixi.sh pixi를 설치한 다음를 설치합니다`rattler-build`.

  ```
  pixi global install rattler-build
  ```

- git - 샘플 리포지토리를 복제하는 데 필요합니다. 에서 용 Windowsgit는 일부 Windows 샘플 레시피에 필요한 `bash`쉘도 제공합니다. Windows
- Amazon S3 버킷 - conda 채널로 사용할 Amazon S3 버킷입니다. Deadline Cloud 팜의 작업 연결 버킷을 사용하거나 별도의 버킷을 생성할 수 있습니다.
- AWS 자격 증명 - `aws configure` 명령 또는 `aws login` 명령을 사용하여 워크스테이션에서 자격 증명을 구성합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 AWS CLI설정을 참조하세요.
- IAM 권한 - (선택 사항) 자격 증명의 권한 범위를 줄이기 위해 Amazon S3 버킷 및 사용하는 채널 접두사(예:)에 대해 다음 권한만 부여하는 (IAM) 정책을 사용할 AWS Identity and Access Management 수 있습니다. `/Conda/*`

- `s3:GetObject`
- `s3:PutObject`
- `s3:DeleteObject`
- `s3:ListBucket`
- `s3:GetBucketLocation`

Amazon S3 채널에 패키지 게시

`s3://` 대상과 `rattler-build publish` 함께를 사용하여 Amazon S3 conda 채널에 패키지를 게시합니다. 채널이 버킷에 없는 경우는 채널을 자동으로 `rattler-build` 초기화합니다. 시작하기 전에 사전 조건을 완료했는지 확인합니다.

다음 예제에서는 Blender의 Deadline Cloud 샘플 리포지토리에서 4.5 샘플 레시피를 게시합니다 GitHub. 샘플 리포지토리에서 다른 레시피를 대체하거나 자체 레시피를 사용할 수 있습니다.

> **Note**
>
> 대규모 애플리케이션은 소스 아카이브, 추출된 파일 및 빌드 출력을 위해 수십 GB의 여유 디스크 공간이 필요할 수 있습니다. 패키지 빌드 출력을 위해 사용 가능한 공간이 충분한 디스크를 사용해야 합니다.

Amazon S3 채널에 패키지를 게시하려면

1. Deadline Cloud 샘플 리포지토리를 복제합니다.

```
git clone https://github.com/aws-deadline/deadline-cloud-samples.git
```

2. 디렉터리를 `conda_recipes`로 변경합니다.

```
cd deadline-cloud-samples/conda_recipes
```

3. 다음 명령을 실행합니다. *`amzn-s3-demo-bucket`*을 버킷 이름으로 바꿉니다.

```
rattler-build publish blender-4.5/recipe/recipe.yaml --to s3://amzn-s3-demo-bucket/
Conda/Default --build-number=+1
```

/Conda/Default 접두사는 버킷 내에서 채널을 구성합니다. 다른 접두사를 사용할 수 있지만 접두사는 채널을 참조하는 모든 명령 및 대기열 구성에서 일관되어야 합니다.

> **빌드 번호 정보**
>
> --build-number=+1 옵션은 대상 채널에 이미 있는 빌드 번호를 기반으로 다음 빌드 번호를 자동으로 선택합니다. 채널에서 패키지를 덮어쓰지 않는 것이 가장 좋습니다. 패키지의 파일 이름이 동일한 경우 항상 새 빌드 번호로 빌드합니다. 를 사용하면 프로덕션 채널 또는 프로덕션을 미러링하는 스테이징 채널에 빌드할 때 이를 --build-number=+1 달성할 수 있습니다. 빌드 번호를 직접 제어하려면와 같은 특정 값으로 설정할 수 있습니다--build-number=7. 옵션을 생략하면는 recipe.yaml 파일에 정의된 빌드 번호를 rattler-build 사용합니다.

패키지 레시피가 conda-forge와 같은 특정 채널의 패키지에 의존하는 경우 명령에 -c conda-forge를 추가합니다.

로컬 빌드의 파일과 같이 이미 빌드한 패키지 .conda 파일을 게시할 수도 있습니다. *amzn-s3-demo-bucket*을 버킷 이름으로 바꿉니다.

```
rattler-build publish output/linux-64/blender-4.5.0-hb0f4dca_0.conda \
    --to s3://amzn-s3-demo-bucket/Conda/Default
```

채널 초기화 또는 재인덱싱

rattler-build publish를 사용하여 패키지를 게시하면 채널이 아직 없는 경우 명령이 채널을 자동으로 초기화합니다. 대부분의 경우 채널을 수동으로 초기화하거나 다시 인덱싱할 필요가 없습니다.

다음과 같은 상황에서 채널을 수동으로 초기화하거나 다시 인덱싱해야 할 수 있습니다.

- 예를 들어 Deadline Cloud 대기열 환경이 채널에 연결할 수 있는지 확인하기 위해 패키지를 게시하기 전에 빈 채널을 생성하려고 합니다.
- 를 사용하는 대신 Amazon S3 도구를 사용하여 .conda 파일을 직접 업로드하거나 삭제rattler-build publish했으며 채널 인덱스가 최신 상태가 아닙니다.

빈 채널 초기화

빈 채널을 초기화하려면 repodata.json 파일을 생성하여 채널 접두사의 noarch 하위 디렉터리에 업로드합니다. *amzn-s3-demo-bucket*을 버킷 이름으로 바꿉니다.

```
echo '{"info":{"subdir":"noarch"},"packages":{},"packages.conda":{},"removed":
[],"repodata_version":1}' > empty_channel_repodata.json
aws s3api put-object --body empty_channel_repodata.json --key Conda/Default/noarch/
repodata.json --bucket amzn-s3-demo-bucket
```

/Conda/Default 접두사는 대기열 환경에서 사용하는 채널 접두사와 일치해야 합니다. 채널을 초기화한 후를 사용하여 채널에 패키지를 게시할 수 있습니다rattler-build publish.

채널 재인덱싱

채널 인덱스가 오래된 경우 rattler-index를 사용하여 채널의 패키지 파일에서 인덱스를 다시 빌드합니다. 먼저를 설치합니다rattler-index.

```
pixi global install rattler-index
```

그런 다음 채널을 다시 인덱싱합니다. *amzn-s3-demo-bucket*을 버킷 이름으로 바꿉니다.

```
rattler-index s3 s3://amzn-s3-demo-bucket/Conda/Default
```

패키지 테스트

패키지를 게시한 후 임시 pixi 프로젝트를 생성하여 패키지가 올바르게 작동하는지 확인합니다. 프로젝트는 Amazon S3 채널에서 패키지를 설치합니다.

패키지를 테스트하려면

1. 임시 테스트 디렉터리를 생성하고 Amazon S3 채널을 사용하여 pixi 프로젝트를 초기화합니다. *amzn-s3-demo-bucket*을 버킷 이름으로 바꿉니다.

   ```
   mkdir package-test-env
   cd package-test-env
   pixi init --channel s3://amzn-s3-demo-bucket/Conda/Default
   ```

2. 프로젝트에 패키지를 추가합니다.

   ```
   pixi add blender=4.5
   ```

3. 패키지가 올바르게 작동하는지 확인합니다.

```
pixi run blender --version
```

`pixi run` 명령은 프로젝트 디렉터리에 대한 conda 환경을 활성화하고 그 안에서 지정된 명령을 실행합니다. 환경은 프로젝트 디렉터리에 유지되므로 다른 터미널에서 동일한 `pixi run` 명령을 사용할 수 있습니다.

채널에서 패키지 제거

Lockfile은 해시로 특정 패키지를 참조하므로 프로덕션에 사용하는 채널에서 패키지를 제거하지 마세요. 패키지를 제거하면 해당 잠금 파일에서 환경을 다시 생성할 수 없습니다. 개발 및 테스트 채널의 경우 버킷에서 `.conda` 파일을 삭제한 다음 채널을 다시 인덱싱하여 특정 패키지를 제거할 수 있습니다.

패키지 파일을 삭제한 다음 채널을 다시 인덱싱합니다. *amzn-s3-demo-bucket*을 버킷 이름으로 바꿉니다.

```
aws s3 rm s3://amzn-s3-demo-bucket/Conda/Default/linux-64/blender-4.5.0-
hb0f4dca_1.conda
```

파일을 삭제한 후 채널을 다시 인덱싱하여 채널 메타데이터를 업데이트합니다. 지침은 채널 재인덱싱을 참조하세요.

패키지 파일은 `linux-64`, `win-64`또는와 같은 플랫폼별 하위 디렉터리에 저장됩니다`osx-arm64`. 하위 디렉터리의 패키지를 나열하려면 다음 명령을 실행합니다.

```
aws s3 ls s3://amzn-s3-demo-bucket/Conda/Default/linux-64/
```

정리

테스트 후 테스트 프로젝트 디렉터리를 제거합니다.

테스트 리소스를 정리하려면

- 테스트 프로젝트 디렉터리를 제거합니다.

  Linux 및에서 다음 명령을 macOS실행합니다.

```
rm -rf package-test-env
```

Windows (cmd)에서 다음 명령을 실행합니다.

```
rmdir /s /q package-test-env
```

빌드 디버깅

빌드에 실패하면는 사용자가 조사할 수 있도록 빌드 디렉터리를 `rattler-build` 보존합니다. 다음 명령을 실행하여 모든 환경 변수가 빌드 중에 설정된 상태로 빌드 환경에서 대화형 셸을 엽니다.

```
rattler-build debug shell
```

디버그 쉘에서 파일을 수정하고, 개별 빌드 명령을 실행하고, 종속성을 추가하여 문제를 격리할 수 있습니다. 자세한 내용은 래틀러 빌드 설명서의 빌드 디버깅을 참조하세요.

다른 플랫폼을 위한 패키지 빌드

명령은 `rattler-build publish` 명령이 실행되는 워크스테이션의 운영 체제에 대한 패키지를 빌드합니다. Deadline Cloud 플릿이 워크스테이션과 다른 운영 체제를 사용하거나 패키지에 다른 호스트 요구 사항이 있는 경우 다음 옵션을 사용할 수 있습니다.

- 대상 운영 체제와 일치하는 호스트`rattler-build publish`에서를 실행합니다. 예를 들어 실행 중인 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스Linux를 사용하여 Linux플릿에 대한 패키지를 빌드합니다.
- Deadline Cloud 패키지 빌드 대기열을 사용하여 대상 플랫폼에서 빌드를 자동화합니다. 패키지 빌드 대기열 생성을 참조하세요.
- (고급) 교차 컴파일을 사용하여 워크스테이션과 다른 플랫폼용 패키지를 빌드합니다. 자세한 내용은 래틀러 빌드 설명서의 교차 컴파일을 참조하세요.

다음 단계

Amazon S3 conda 채널에 패키지를 게시한 후 채널을 사용하도록 Deadline Cloud 대기열을 구성합니다.

- 사용자 지정 conda 패키지에 대한 프로덕션 대기열 권한 구성 - 프로덕션 대기열에 Amazon S3 conda 채널에 대한 읽기 전용 액세스 권한을 부여합니다.
- 대기열 환경에 conda 채널 추가 - Amazon S3 conda 채널에서 패키지를 설치하도록 대기열 환경을 구성합니다.

사용자 지정 conda 패키지에 대한 프로덕션 대기열 권한 구성

프로덕션 대기열에는 대기열의 S3 버킷에 있는 /Conda 접두사에 대한 읽기 전용 권한이 필요합니다. 프로덕션 대기열과 연결된 역할의 AWS Identity and Access Management (IAM) 페이지를 열고 다음을 사용하여 정책을 수정합니다.

1. Deadline Cloud 콘솔을 열고 패키지 빌드 대기열의 대기열 세부 정보 페이지로 이동합니다.
2. 대기열 서비스 역할을 선택한 다음 대기열 편집을 선택합니다.
3. 대기열 서비스 역할 섹션으로 스크롤한 다음 IAM 콘솔에서이 역할 보기를 선택합니다.
4. 권한 정책 목록에서 대기열의 AmazonDeadlineCloudQueuePolicy를 선택합니다.
5. 권한 탭에서 편집을 선택합니다.
6. 다음과 같이 대기열 서비스 역할에 새 섹션을 추가합니다. *amzn-s3-demo-bucket* 및 *111122223333*을 자체 버킷 및 계정으로 바꿉니다.

```
{
   "Effect": "Allow",
   "Sid": "CustomCondaChannelReadOnly",
   "Action": [
    "s3:GetObject",
    "s3:ListBucket"
   ],
   "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "111122223333"
    }
   }
  },
```

대기열 환경에 conda 채널 추가

S3 conda 채널을 사용하려면 Deadline Cloud에 제출하는 작업의 CondaChannels 파라미터에 s3://amzn-s3-demo-bucket/Conda/Default 채널 위치를 추가해야 합니다. Deadline Cloud와 함께 제공된 제출자는 사용자 지정 conda 채널 및 패키지를 지정하는 필드를 제공합니다.

프로덕션 대기열의 conda 대기열 환경을 편집하여 모든 작업을 수정하지 않아도 됩니다. 다음 절차를 수행하세요.

1. Deadline Cloud 콘솔을 열고 프로덕션 대기열의 대기열 세부 정보 페이지로 이동합니다.
2. 환경 탭을 선택합니다.
3. Conda 대기열 환경을 선택한 다음 편집을 선택합니다.
4. JSON 편집기를 선택한 다음 스크립트에서에 대한 파라미터 정의를 찾습니다`CondaChannels`.
5. 새로 생성된 S3 conda 채널로 시작`default: "deadline-cloud"`하도록 선을 편집합니다.

```
default: "s3://amzn-s3-demo-bucket/Conda/Default deadline-cloud"
```

서비스 관리형 플릿은 기본적으로 conda에 대한 유연한 채널 우선 순위를 활성화합니다. Blender 4.5가 새 채널과 `deadline-cloud` 채널 모두에 `blender=4.5` 있는지 요청하는 작업의 경우 채널 목록의 첫 번째 채널에서 패키지를 가져옵니다. 지정된 패키지 버전을 첫 번째 채널에서 찾을 수 없는 경우 패키지 버전에 대해 후속 채널이 확인됩니다.

고객 관리형 플릿의 경우 Deadline Cloud 샘플 리포지토리의 conda 대기열 환경 샘플 중 하나를 사용하여 conda 패키지 사용을 활성화할 수 있습니다. GitHub

애플리케이션 또는 플러그인에 대한 conda 패키지 생성

conda 패키지는 모든 언어로 작성된 소프트웨어의 압축된 아카이브입니다. Conda는 다양한 운영 체제 및 아키텍처 조합을 지원하므로 Python 및 기타 언어용 라이브러리와 Nuke 함께 BlenderMaya, 및와 같은 전체 애플리케이션을 패키징할 수 있습니다. conda 패키지에 대한 자세한 내용은 conda 설명서의 패키지를 참조하세요.

conda 패키지를 사용하려면 가상 환경에 설치합니다. conda 가상 환경에는 패키지가 설치된 접두사 디렉터리가 있습니다. 패키지를 설치하면 지원되는 경우 파일의 하드링크 또는 리링크가 사용되므로 동일한 패키지로 여러 환경을 생성해도 상당한 추가 디스크 공간이 사용되지 않습니다. 가상 환경을 사용하려면 가상 환경을 활성화하여 환경 변수를 설정합니다. 활성화는 패키지가 제공하는 스크립트를 실행하여 각 패키지에 PATH 또는 기타 환경 변수를 수정할 수 있는 기회를 제공합니다. Conda 패키지에는 일반적으로 애플리케이션 또는 라이브러리가 포함되어 있지만 유연한 활성화를 통해 공유 파일 시스템에 설치된 애플리케이션을 가리킬 수도 있습니다.

사용자 지정 패키지를 만들려면 세 단계가 필요합니다. 레시피에는 빌드 지침이 포함되어 있고, 패키지는 빌드된 아티팩트(`.conda` 또는 `.tar.bz2` 파일)이며, 채널은 설치를 위해 패키지를 호스팅합니

다. `rattler-build publish` 명령은 세 단계를 모두 처리합니다. 즉, 레시피를 패키지에 빌드하여 채널에 게시하거나 패키지 아티팩트를 직접 가져와서 게시할 수 있습니다.

conda-forge 커뮤니티는 광범위한 오픈 소스 소프트웨어용 패키지 레시피를 유지 관리하고 `conda-forge` 채널에서 패키지 아티팩트를 호스팅합니다. 를 패키지 소스`conda-forge`로 포함하도록 대기열을 구성한 다음 실행할 conda-forge 패키지에 의존하는 사용자 지정 패키지를 빌드할 수 있습니다. Linux의 경우 conda-forge는 일관된 컴파일 및 연결 옵션이 선택된 상태로 CUDA 지원을 포함한 전체 컴파일러 도구 체인을 호스팅합니다. conda-forge 패키지를 자체 레시피의 종속성으로 사용하거나 동일한 환경에서 사용자 지정 패키지와 함께 설치할 수 있습니다.

종속성을 포함한 전체 애플리케이션을 conda 패키지로 결합할 수 있습니다. Deadline Cloud가 서비스 관리형 플릿의 기한 클라우드 채널에 제공하는 패키지는이 바이너리 재패키징 접근 방식을 사용합니다. 이렇게 하면 conda 가상 환경에 맞게 설치와 동일한 파일이 구성됩니다.

> **Note**
>
> 대규모 애플리케이션은 소스 아카이브, 추출된 파일 및 빌드 출력을 위해 수십 GB의 여유 디스크 공간이 필요할 수 있습니다. 패키지 빌드 출력을 위해 사용 가능한 공간이 충분한 디스크를 사용해야 합니다.

애플리케이션 패키징

conda용 애플리케이션을 다시 패키징할 때는 다음 두 가지 목표가 있습니다.

- 애플리케이션의 대부분의 파일은 기본 conda 가상 환경 구조와 별개여야 합니다. 그런 다음 환경은 애플리케이션을 conda-forge와 같은 다른 소스의 패키지와 혼합할 수 있습니다.
- conda 가상 환경이 활성화되면 PATH 환경 변수에서 애플리케이션을 사용할 수 있어야 합니다.

conda용 애플리케이션을 다시 패키징하려면

1. 애플리케이션을와 같은 하위 디렉터리에 설치하는 conda 빌드 레시피를 작성합니다`$CONDA_PREFIX/opt/<application-name>`. 이렇게 하면 `bin` 및와 같은 표준 접두사 디렉터리와 구분됩니다`lib`.
2. symlink 또는 시작 스크립트를 `$CONDA_PREFIX/bin`에 추가하여 애플리케이션 바이너리를 실행합니다.

또는 `conda activate` 명령이 실행할 activate.d 스크립트를 생성하여 PATH에 애플리케이션 바이너리 디렉터리를 추가합니다. 에서 symlink가 환경이 생성될 수 있는 모든 곳에서 지원되지 않는 Windows경우 애플리케이션 시작 또는 activate.d 스크립트를 대신 사용합니다.

3. 일부 애플리케이션은 Deadline Cloud 서비스 관리형 플릿에 기본적으로 설치되지 않은 라이브러리에 의존합니다. 예를 들어 X11 창 시스템은 일반적으로 비대화형 작업에 필요하지 않지만 일부 애플리케이션에서는 여전히 그래픽 인터페이스 없이 실행해야 합니다. 생성한 패키지 내에서 이러한 종속성을 제공해야 합니다.
4. 애플리케이션이 플러그인을 지원하는 경우 플러그인 패키지가 가상 환경의 애플리케이션과 통합되기 위해 따라야 하는 명확한 규칙을 제공합니다. 예를 들어 Maya 2026년 샘플 레시피는 Maya 플러그인에 대한이 규칙을 문서화합니다.
5. 패키징하는 애플리케이션의 저작권 및 라이선스 계약을 준수해야 합니다. Conda 채널에 프라이빗 Amazon S3 버킷을 사용하여 배포를 제어하고 팜에 대한 패키지 액세스를 제한하는 것이 좋습니다.

`deadline-cloud` 채널의 패키지에 대한 샘플 레시피는의 Deadline Cloud 샘플 리포지토리에서 사용할 수 있습니다GitHub.

플러그인 패키징

애플리케이션 플러그인은 자체 conda 패키지로 패키징할 수 있습니다. 플러그인 패키지를 생성할 때 다음 지침을 따릅니다.

- 빌드 레시피에 호스트 애플리케이션 패키지를 빌드 및 실행 종속성으로 포함합니다`recipe.yaml`. 빌드 레시피가 호환되는 패키지에만 설치되도록 버전 제약 조건을 사용합니다.
- 플러그인 등록에 대한 호스트 애플리케이션 패키지 규칙을 따릅니다.

어댑터 패키지

일부 Deadline Cloud 애플리케이션 통합은 애플리케이션 인터페이스를 확장하여 작업 템플릿 작성을 간소화하는 어댑터를 사용합니다. 어댑터는 백그라운드 데몬 실행, 상태 보고, 경로 매핑 적용을 지원하는 명령줄 인터페이스입니다. 자세한 내용은의 Open Job Description Adaptor 런타임을 참조하세요GitHub. 예를 들어의 deadline-cloud-for-mayaGitHub에는 통합 작업 제출 GUI와 서비스 관리형 플릿의 `maya-openjd` 패키지로 사용할 수 있는 Maya 어댑터가 포함되어 있습니다.

Deadline Cloud 제출자 GUIs의 작업 제출에는 작업을 실행하기 위한 가상 환경에 포함할 conda 패키지를 지정하는 `CondaPackages` 파라미터 값이 포함됩니다. 의 `CondaPackages` 파라미터 값은

Maya 일반적으로과 같maya=2026.* maya-openjd=0.15.* maya-mtoa으며 플러그인 패키지에 대한 대체 항목을 포함할 수 있습니다. 대기열 환경에서 작업 실행을 위한 conda 가상 환경을 설정하면 이러한 패키지 이름과 버전 제약 조건이 호환되도록 확인되고 실행해야 하는 모든 종속성 패키지가 추가됩니다. 각 어댑터 및 플러그인 패키지는의 버전, Python의 Maya버전 및 기타 종속성을 포함하여 호환되는 항목을 지정합니다.

의 maya-openjd 레시피와 같은 샘플을 사용하여 자체 어댑터 패키지를 빌드하려면 conda-forge에서 제공하는 Python 및 기타 종속성을 위한 패키지를 기반으로 빌드GitHub할 수 있습니다. 종속성을 충족하려면 먼저 기한과 openjd-adaptor-runtime 레시피를 빌드해야 할 수 있습니다.

에 대한 conda 빌드 레시피 생성 Blender

Blender는 무료로 사용할 수 있고 conda로 패키징할 수 있으므로 AWS Deadline Cloud(Deadline Cloud)에 대한 conda 패키지를 생성하는 방법을 배우는 데 좋은 출발점입니다. Blender Foundation은 여러 운영 체제에 대한 애플리케이션 아카이브를 제공합니다. 의 Deadline Cloud Blender 샘플 리포지토리의 4.5 샘플 레시피는 이러한 아카이브를 conda 패키지로 GitHub 패키징합니다.

레시피 이해

recipe.yaml 파일은 패키지 메타데이터, 소스 URLs 및 빌드 옵션을 래틀러 빌드 템플릿 구문으로 정의합니다. 레시피는 버전 번호를 한 번 지정하고 운영 체제에 따라 다른 소스 URLs을 제공합니다.

의 build 섹션에서는 이진 재배치 및 동적 공유 객체(DSO) 연결 검사를 recipe.yaml 비활성화합니다. 이러한 옵션은 임의의 디렉터리 접두사에서 conda 가상 환경에 설치할 때 패키지가 작동하는 방식을 제어합니다. build 섹션에 사용되는 기본값은 각 종속성 라이브러리를 별도로 패키징하도록 설계되었지만 애플리케이션을 바이너리로 다시 패키징할 때는 변경해야 합니다. Blender는 애플리케이션 아카이브가 재배치 가능성을 염두에 두고 빌드되므로 RPATH 조정이 필요하지 않습니다. 재배치 가능성 추가 예제는 Maya용 conda 레시피 생성을 참조하세요.

패키지 빌드 중에 build.sh 또는 build_win.sh 스크립트가 실행되어 환경에 파일을 설치합니다. 이러한 스크립트는 설치 파일을에 복사하고, $PREFIX/bin (에서Linux)에서 symlink를 생성하고$PREFIX/opt/blender,와 같은 환경 변수를 구성하는 활성화 스크립트를 설정합니다BLENDER_LOCATION. 에서 활성화 스크립트Windows는 symlink를 생성하는 대신 Blender 디렉터리를 PATH에 추가합니다.

Windows 빌드 스크립트는 플랫폼 간 일관성을 위해 cmd.exe .bat 파일 bash 대신를 사용합니다. 용 git를 Windows 설치하여 패키지 빌드bash를 제공할 수 있습니다.

레시피에는 자동 패키지 빌드 작업을 Deadline Cloud에 제출하기 위한 conda 플랫폼 및 메타데이터를 지정하는 deadline-cloud.yaml 파일도 포함되어 있습니다. 자세한 내용은 패키지 빌드 작업 제출을 참조하세요.

Blender 패키지 빌드

`rattler-build publish`를 사용하여 Blender 4.5 레시피를 빌드하고 패키지를 채널에 게시합니다. 테스트를 위해 로컬 파일 시스템 채널에 게시하거나 프로덕션 사용을 위해 Amazon S3 채널에 직접 게시할 수 있습니다. 로컬에서 패키지 빌드 및 테스트에서 설정을 완료한 경우 `conda_recipes` 디렉터리에서 다음 명령을 실행합니다.

```
rattler-build publish blender-4.5/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

다른 게시 옵션의 경우:

- Amazon S3 채널에 게시하려면 S3 conda 채널에 패키지 게시를 참조하세요.
- Deadline Cloud 패키지 빌드 대기열을 사용하여 빌드를 자동화하려면 Deadline Cloud를 사용하여 패키지 빌드 자동화를 참조하세요.

Blender 렌더 작업으로 패키지 테스트

Blender 4.5 패키지를 빌드한 후 렌더 작업으로 테스트할 수 있습니다. Blender 장면이 없는 경우 Blender 데모 파일 페이지에서 Blender 3.5 - 코지 키친 장면을 다운로드합니다. Deadline Cloud 샘플 리포지토리에는 로컬 및 클라우드 테스트 모두에 사용할 수 있는 `blender_render` 작업 번들과 conda 대기열 환경이 포함되어 있습니다.

로컬 테스트

작업 설명 열기 CLI를 사용하여 워크스테이션에서 작업 템플릿을 실행할 수 있습니다. 를 사용하여 CLI를 설치합니다`pip`.

```
pip install openjd-cli
```

샘플 리포지토리의 `job_bundles` 디렉터리에서 다음 명령을 실행합니다. */path/to/scene.blend*를 Blender 장면 파일의 경로로 바꿉니다.

```
openjd run blender_render/template.yaml \
    --environment ../queue_environments/conda_queue_env_pyrattler.yaml \
    -p CondaPackages=blender=4.5 \
    -p CondaChannels=file://$HOME/my-conda-channel \
    -p BlenderSceneFile=/path/to/scene.blend \
```

```
    -p Frames=1
```

`--environment` 옵션은에 지정된 패키지를 사용하여 conda 가상 환경을 생성하는 conda 대기열 환경을 적용합니다`CondaPackages`. `CondaChannels` 파라미터는 대기열 환경에 패키지를 찾을 위치를 알려줍니다. 로컬 채널 대신 Amazon S3 채널에 게시한 경우 `file://` 경로를 `s3://` 채널 URL로 바꿉니다.

Deadline Cloud에서 테스트

Amazon S3 conda 채널을 사용하도록 프로덕션 대기열을 구성한 후 Deadline Cloud에 렌더링 작업을 제출할 수 있습니다. 샘플 리포지토리의 `job_bundles` 디렉터리에서 다음 명령을 실행합니다.

```
deadline bundle submit blender_render \
    -p CondaPackages=blender=4.5 \
    -p BlenderSceneFile=/path/to/scene.blend \
    -p Frames=1
```

Deadline Cloud 모니터를 사용하여 작업 진행 상황을 추적합니다. 모니터에서 작업의 작업을 선택하고 로그 보기를 선택합니다. Conda 세션 시작 작업을 선택하여 Amazon S3 채널에서 패키지를 찾았는지 확인합니다.

에 대한 conda 빌드 레시피 생성 Autodesk Maya

와 같은 상용 애플리케이션은와 같은 오픈 소스 애플리케이션과 비교하여 추가 패키징 요구 사항을 Autodesk Maya 도입합니다Blender. Blender 레시피는 오픈 소스 라이선스에 따라 간단한 재배치 가능 아카이브를 패키징합니다. 상용 애플리케이션은 설치 관리자를 통해 배포되는 경우가 많으며 라이선스 관리 구성이 필요합니다.

상용 애플리케이션에 대한 고려 사항

상용 애플리케이션을 패키징할 때는 다음 고려 사항이 적용됩니다. 세부 정보는 각가에 적용되는 방법을 보여줍니다Maya.

- 라이선스 - 애플리케이션의 라이선스 권한 및 제한을 이해합니다. 라이선스 관리 시스템을 구성해야 할 수 있습니다. Autodesk 클라우드 권한에 대한 구독 혜택 FAQ를 읽고의 클라우드 권한을 이해합니다Maya. Autodesk 제품은 일반적으로 구성에 관리자 액세스가 필요한 `ProductInformation.pit` 파일을 사용합니다. 씬 클라이언트를 위한 제품 기능은 재배치 가능한 대안을 제공합니다. 자세한 내용은 Maya 및 MotionBuilder용 씬 클라이언트 라이선싱을 참조하세요.
- 시스템 라이브러리 종속성 - 일부 애플리케이션은 서비스 관리형 플릿 워커 호스트에 설치되지 않은 라이브러리에 의존합니다. Maya는 프리타입 및 fontconfig를 포함한 라이브러리에 의존합니다.

AL2023dnf과 같은 시스템 패키지 관리자에서 이러한 라이브러리를 사용할 수 있는 경우 패키지 관리자를 소스로 사용할 수 있습니다. RPM 패키지는 재배치가 가능하도록 빌드되지 않았으므로와 같은 도구를 사용하여 Maya 설치 접두사 내의 종속성을 patchelf 해결해야 합니다.

- 설치를 위한 관리자 액세스 - 일부 설치 관리자는 관리자 액세스가 필요합니다. 서비스 관리형 플릿은 관리자 액세스를 제공하지 않으므로 별도의 시스템에 애플리케이션을 설치하고 패키지 빌드에 대한 파일 아카이브를 생성해야 합니다. 용 Windows 설치 관리자에는이 접근 방식이 Maya 필요합니다. 레시피의 README.md 새로 시작된 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 사용하여 반복 가능한 절차를 문서화합니다.
- 플러그인 통합 - 샘플 Maya 패키지는 애플리케이션을 사용자 수준 구성에서 격리MAYA_NO_HOME=1하도록 정의하고 플러그인 패키지가 가상 환경 내에 .mod 파일을 배치할 수 MAYA_MODULE_PATH 있도록에 모듈 검색 경로를 추가합니다. 전체 플러그인 통합 규칙은 Maya 2026년 샘플 레시피를 참조하세요.

레시피 이해

recipe.yaml 파일은 패키지 메타데이터를 래틀러 빌드 템플릿 구문으로 정의합니다. 파일의 다음 섹션을 검토합니다.

- source - sha256 해시를 포함하여 설치 관리자 아카이브를 참조합니다. 에서 Linux소스는 Autodesk 설치 관리자 아카이브입니다. 에서 소스Windows에는 설치 관리자 아카이브와 Maya 클라우드 배포를 준비Autodesk하는 cleanMayaForCloud.py의 스크립트가 모두 포함됩니다. 예를 들어 새 버전을 패키징할 때 소스 파일을 변경할 때 해시를 업데이트합니다.
- build -가 Maya 사용하는 라이브러리 및 바이너리 디렉터리에서 자동 메커니즘이 제대로 작동하지 않기 때문에 기본 바이너리 재배치 및 DSO 연결 검사를 끕니다. 에서 레시피Linux는 상대 RPATHs를 수동으로 설정하기 위한 빌드 종속성patchelf으로를 포함합니다.
- 정보 - conda 채널의 콘텐츠를 검색하거나 처리하기 위한 애플리케이션에 대한 메타데이터입니다.

빌드 스크립트(의 경우 build.shLinux,의 경우 build_win.shWindows)에는 각 단계를 설명하는 설명이 포함되어 있습니다. 스크립트는 다음과 같은 주요 작업을 수행합니다.

- 설치 프로그램 추출 - Maya 설치 파일을 conda 접두사로 추출합니다. Linux 및 Windows 스크립트는 설치 프로그램 형식으로 인해 이를 다르게 처리합니다. 자세한 내용은 빌드 스크립트를 참조하세요.
- 시스템 라이브러리 종속성 설치 -에서 스크립트는가 Maya 필요하지만 서비스 관리형 Linux플릿 호스트에 없는 시스템 라이브러리를 다운로드하고 추출합니다. 스크립트는 이러한 라이브러리를 Maya lib 디렉터리에 복사하여 conda 환경에서 사용할 수 있도록 합니다.

- patchelf를 사용하여 상대 RPATHs 설정 $ORIGIN-에서 스크립트는 Linux를 사용하여 공유 라이브러리patchelf --add-rpath에 상대 경로를 추가합니다. 이 접근 방식은 Conda 환경에서 를 사용하지 않도록 LD_LIBRARY_PATH Conda 권장 사항을 따릅니다. 스크립트는 라이브러리를 여러 디렉터리 수준(lib, lib/python*/site-packages, lib/python*/lib-dynload)으로 패치하므로 각 라이브러리는 자체 위치에 상대적인 종속성을 찾을 수 있습니다. 레시피는 DT_RUNPATH 대신를 설정하는 모범 사례를 따르DT_RPATH므로 디버깅에 필요할 때가 검색 경로를 재정의LD_LIBRARY_PATH할 수 있습니다.
- 씬 클라이언트 라이선스 구성 - 스크립트는 시스템 수준 관리자 액세스가 필요하지 않고 ProductInformation.pit 파일을 conda 환경 내에 배치할 수 있도록에 설명된 대로 씬 클라이언트 라이선스를 Autodesk 설정합니다.
- 활성화 스크립트 설정 - 스크립트는 MAYA_LOCATION, MAYA_VERSION, MAYA_NO_HOME, 등의 환경 변수를 설정하는 활성화 및 비활성화 스크립트를 생성합니다MAYA_MODULE_PATH. 에서는 Deadline Cloud 샘플 대기열 환경이 Windows에서 환경을 .bat 활성화bash하는 데를 사용하기 때문에 스크립트가 .sh 및 활성화 파일을 모두 생성합니다Windows.

Maya 패키지 빌드

Maya 패키지를 빌드하기 전에 Autodesk 계정에서 Maya 설치 관리자를 다운로드합니다. 의 경우 아카이브를 conda_recipes/archive_files 디렉터리에 직접 Linux배치합니다. 의 경우 README.md 절차에 Windows따라 아카이브를 생성합니다.

rattler-build publish를 사용하여 패키지를 빌드하고 게시합니다. Maya 레시피에는 conda-forge에서 사용할 수 Linux있는에 대한 빌드 종속성patchelf으로가 필요합니다. -c conda-forge를 추가하여 빌드 중에 종속성을 사용할 수 있도록 합니다. conda_recipes 디렉터리에서 다음 명령을 실행합니다.

```
rattler-build publish maya-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

다른 게시 옵션의 경우:

- Amazon S3 채널에 게시하려면 S3 conda 채널에 패키지 게시를 참조하세요.
- Deadline Cloud 패키지 빌드 대기열을 사용하여 빌드를 자동화하려면 Deadline Cloud를 사용하여 패키지 빌드 자동화를 참조하세요. Linux 및 Windows 패키지를 모두 빌드하려면 submit-package-job 스크립트와 함께 --all-platforms 옵션을 사용합니다.

Maya 및 로 턴테이블 샘플을 렌더링하려면 MtoA 플러그인 및 Maya 어댑터 패키지를 모두 Arnold빌드합니다. 세 패키지를 모두 게시한 후 Deadline Cloud 샘플 리포지토리의 Maya/ 작업 번들이 포함된 턴테Arnold이블을 사용하여 테스트 렌더링 작업을 제출할 수 있습니다. Maya 렌더링 작업을 사용하여 패키지 테스트를 참조하세요.

Maya 어댑터용 conda 빌드 레시피 생성

`maya-openjd` 패키지는 AWS Deadline Cloud(Deadline Cloud) 작업 제출Maya과 통합되는 어댑터를 제공합니다. Deadline Cloud 제출자 GUI를 사용하여 Maya 렌더링 작업을 제출하면 `CondaPackages` 파라미터에 `maya` 패키지`maya-openjd`가 함께 포함됩니다. 어댑터는 작업 세션 중에 시작Maya, 렌더링 파라미터 전달 및 애플리케이션 수명 주기 관리를 처리합니다. 어댑터에 대한 자세한 내용은 어댑터 패키지를 참조하세요.

레시피 이해

maya-openjd 샘플 레시피는 PyPI에 게시된 deadline-cloud-for-maya 소스 패키지에서 어댑터를 빌드합니다. recipe.yaml은를 사용하여 `pip` conda 환경에 패키지를 설치합니다.

레시피는 먼저 빌드해야 하는 Deadline Cloud 샘플 리포지토리의 Python 및 다른 두 패키지에 따라 달라집니다.

- deadline - Deadline Cloud 클라이언트 라이브러리입니다.
- openjd-adaptor-runtime – Open Job Description 어댑터 런타임입니다.

Python 및 기타 종속성은 conda-forge에서 사용할 수 있으므로 어댑터 패키지를 빌드할 때 `rattler-build publish` 명령`-c conda-forge`에를 추가합니다.

어댑터 패키지 빌드

`maya-openjd` 패키지는 Deadline Cloud 샘플 리포지토리의 다른 두 패키지에 따라 달라집니다. `conda_recipes` 디렉터리에서 순서대로 세 패키지를 모두 빌드합니다. 각 명령의 `-c conda-forge` 옵션은 Python 및 Python 라이브러리의 레시피 종속성을 충족하는 것입니다.

`deadline` 패키지를 빌드합니다.

```
rattler-build publish deadline/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
```

```
    -c conda-forge
```

`openjd-adaptor-runtime` 패키지를 빌드합니다.

```
rattler-build publish openjd-adaptor-runtime/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

`maya-openjd` 패키지를 빌드합니다.

```
rattler-build publish maya-openjd/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1 \
    -c conda-forge
```

다른 게시 옵션의 경우:

- Amazon S3 채널에 게시하려면 S3 conda 채널에 패키지 게시를 참조하세요.
- Deadline Cloud 패키지 빌드 대기열을 사용하여 빌드를 자동화하려면 Deadline Cloud를 사용하여 패키지 빌드 자동화를 참조하세요.

Autodesk Maya to Arnold (MtoA) 플러그인용 conda 빌드 레시피 생성

Maya to Arnold (MtoA) 플러그인은 Arnold렌더러를 내에서 옵션으로 추가합니다Maya. MtoA 샘플 레시피는 플러그인을 호스트 애플리케이션 패키지와 통합되는 별도의 conda 패키지로 패키징하는 방법을 보여줍니다.

레시피 이해

recipe.yaml은 빌드 및 실행 요구 사항 모두에 대해 `maya` 패키지에 대한 종속성을 지정합니다. 이 종속성은 버전 제약 조건을 사용하므로 플러그인은 호환되는 Maya 버전으로만 설치됩니다.

레시피는 Maya 레시피와 동일한 소스 아카이브를 사용합니다. 빌드 스크립트는 Maya 패키지가에서 구성하는 `$PREFIX/usr/autodesk/maya$MAYA_VERSION/modules` 디렉터리에 `mtoa.mod` 파일을 설치하고 MtoA 생성합니다`MAYA_MODULE_PATH`. Arnold 및는 동일한 라이선스 기술을 Maya 사용하므로 Maya 패키지에 이미 Arnold 필요한 라이선스 정보가 포함되어 있습니다.

MtoA 패키지 빌드

는 빌드 Maya 시에 MtoA 종속되므로 패키지를 빌드하기 Maya 전에 MtoA 패키지를 빌드합니다. `rattler-build publish`를 사용하여 패키지를 빌드하고 게시합니다. `conda_recipes` 디렉터리에서 다음 명령을 실행합니다.

```
rattler-build publish maya-mtoa-2026/recipe/recipe.yaml \
    --to file://$HOME/my-conda-channel \
    --build-number=+1
```

`rattler-build publish` 명령은 종속성을 해결할 때 대상 채널을 가장 높은 우선 순위의 채널로 사용하므로 이전에 게시한 `maya` 패키지를 자동으로 사용할 수 있습니다.

다른 게시 옵션의 경우:

- Amazon S3 채널에 게시하려면 S3 conda 채널에 패키지 게시를 참조하세요.
- Deadline Cloud 패키지 빌드 대기열을 사용하여 빌드를 자동화하려면 Deadline Cloud를 사용하여 패키지 빌드 자동화를 참조하세요.

Maya 렌더 작업을 사용하여 패키지 테스트

Maya, MtoA및 `maya-openjd` 패키지를 빌드한 후 렌더링 작업으로 테스트할 수 있습니다. Deadline Cloud 샘플 리포지토리에는 Maya 및를 사용하여 애니메이션을 렌더링하는 Maya/ 작업 번들이 포함된 턴테Arnold이블이 포함되어 있습니다Arnold. 또한 작업 번들은 FFmpeg를 사용하여 `conda-forge` 채널에서 사용할 수 있는 비디오를 인코딩합니다.

로컬 테스트

작업 설명 열기 CLI를 사용하여 워크스테이션에서 작업 템플릿을 실행할 수 있습니다. 를 사용하여 CLI를 설치합니다`pip`.

```
pip install openjd-cli
```

샘플 리포지토리의 `job_bundles` 디렉터리에서 다음 명령을 실행합니다. 이 `ErrorOnArnoldLicenseFail=false` 파라미터는 Arnold에 라이선스를 사용할 수 없을 때 실패하는 대신 워터마크를 사용하여 렌더링하도록 지시합니다.

```
openjd run turntable_with_maya_arnold/template.yaml \
```

```
    --environment ../queue_environments/conda_queue_env_pyrattler.yaml \
    -p CondaPackages="maya maya-mtoa maya-openjd ffmpeg" \
    -p CondaChannels="file://$HOME/my-conda-channel conda-forge" \
    -p ErrorOnArnoldLicenseFail=false \
    -p FrameRange=1-5
```

`--environment` 옵션은에 지정된 패키지를 사용하여 conda 가상 환경을 생성하는 conda 대기열 환경을 적용합니다`CondaPackages`. `CondaChannels` 파라미터에는 사용자 지정 패키지의 로컬 채널과의 로컬 채널`conda-forge`이 모두 포함됩니다`ffmpeg`. 로컬 채널 대신 Amazon S3 채널에 게시한 경우 `file://` 경로를 `s3://` 채널 URL로 바꿉니다.

작업이 완료되면 렌더링된 출력이 `turntable_with_maya_arnold/output/` 디렉터리에 있습니다.

Deadline Cloud에서 테스트

Amazon S3 conda 채널을 사용하도록 프로덕션 대기열을 구성한 후 Deadline Cloud에 렌더링 작업을 제출합니다. Conda 대기열 환경의 `CondaChannels` 파라미터에 `conda-forge` 채널을 추가하여 어댑터에 필요한 `ffmpeg` 및 Python 종속성의 소스를 제공합니다. 샘플 리포지토리의 `job_bundles` 디렉터리에서 다음 명령을 실행합니다.

```
deadline bundle submit turntable_with_maya_arnold
```

Deadline Cloud 모니터를 사용하여 작업 진행 상황을 추적합니다. 모니터에서 작업의 작업을 선택하고 로그 보기를 선택합니다. Conda 세션 시작 작업을 선택하여 `maya`, `maya-mtoa`및 `maya-openjd` 패키지가 Amazon S3 채널에서 발견되었는지 확인합니다.

Deadline Cloud를 사용하여 패키지 빌드 자동화

CI/CD 워크플로의 경우 또는 여러 운영 체제용 패키지를 빌드해야 하는 경우 Deadline Cloud 패키지 빌드 대기열을 생성할 수 있습니다. 대기열은 패키지를 빌드하고 Amazon Simple Storage Service(Amazon S3) conda 채널에 게시하는 플릿의 빌드 작업을 예약합니다. 이렇게 하면 필요한 모든 구성에서 소프트웨어 릴리스에 대한 지속적인 패키지 빌드를 유지 관리할 수 있습니다.

AWS CloudFormation (CloudFormation) 템플릿을 사용하거나 Deadline Cloud 콘솔에서 수동으로 패키지 빌드 대기열을 생성할 수 있습니다. CloudFormation 템플릿은 프로덕션 대기열과 패키지 빌드 대기열이 이미 구성된 전체 팜을 배포합니다. 콘솔에서 대기열을 생성하면 개별 설정을 더 잘 제어할 수 있습니다.

를 사용하여 패키지 빌드 대기열 생성 CloudFormation

CloudFormation 템플릿을 사용하여 패키지 빌드 대기열이 포함된 Deadline Cloud 팜을 생성할 수 있습니다. 템플릿은 프라이빗 Amazon S3 conda 채널을 사용하여 프로덕션 대기열과 패키지 빌드 대기열을 구성합니다.

템플릿을 배포하기 전에 작업 첨부 파일과 conda 채널을 보관할 Amazon S3 버킷을 생성합니다. Amazon S3 콘솔에서 버킷을 생성할 수 있습니다. 템플릿을 배포할 때 버킷 이름이 필요합니다.

CloudFormation 템플릿을 배포하려면

1. 의 Deadline Cloud 샘플 리포지토리에서 deadline-cloud-starter-farm-template.yaml 템플릿을 다운로드합니다GitHub.
2. CloudFormation 콘솔에서 스택 생성을 선택한 다음 새 리소스 사용(표준)을 선택합니다.
3. 옵션을 선택하여 템플릿 파일을 업로드한 다음 `deadline-cloud-starter-farm-template.yaml` 파일을 업로드합니다.
4. **`StarterFarm`**와 같은 스택의 이름을 입력하고 작업 연결을 위한 Amazon S3 버킷의 이름과 conda 채널을 제공합니다.
5. CloudFormation 콘솔 단계에 따라 스택 생성을 완료합니다.

템플릿 파라미터 및 사용자 지정 옵션에 대한 자세한 내용은의 Deadline Cloud 샘플 리포지토리에서 스타터 팜 README를 참조하세요GitHub.

콘솔에서 패키지 빌드 대기열 생성

Deadline Cloud 사용 설명서의 대기열 생성 지침을 따릅니다. 다음과 같이 변경합니다.

- 5단계에서 기존 Amazon S3 버킷을 선택합니다. 빌드 아티팩트가 일반적인 Deadline Cloud 연결과 분리**`DeadlineCloudPackageBuild`**되도록와 같은 루트 폴더 이름을 지정합니다.
- 6단계에서는 패키지 빌드 대기열을 기존 플릿과 연결하거나 현재 플릿이 적합하지 않은 경우 완전히 새로운 플릿을 생성할 수 있습니다.
- 9단계에서 패키지 빌드 대기열에 대한 새 서비스 역할을 생성합니다. 권한을 수정하여 대기열에 패키지를 업로드하고 conda 채널을 다시 인덱싱하는 데 필요한 권한을 부여합니다.

패키지 빌드 대기열 권한 구성

패키지 빌드 대기열이 대기열의 Amazon S3 버킷에 있는 /Conda 접두사에 액세스하도록 허용하려면 대기열의 역할을 수정하여 읽기/쓰기 액세스 권한을 부여해야 합니다. 이 역할에는 패키지 빌드 작업이 새 패키지를 업로드하고 채널을 다시 인덱싱할 수 있도록 다음 권한이 필요합니다.

- `s3:GetObject`
- `s3:PutObject`
- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:DeleteObject`

1. Deadline Cloud 콘솔을 열고 패키지 빌드 대기열의 대기열 세부 정보 페이지로 이동합니다.
2. 대기열 서비스 역할을 선택한 다음 대기열 편집을 선택합니다.
3. 대기열 서비스 역할 섹션으로 스크롤한 다음 IAM 콘솔에서이 역할 보기를 선택합니다.
4. 권한 정책 목록에서 대기열의 AmazonDeadlineCloudQueuePolicy를 선택합니다.
5. 권한 탭에서 편집을 선택합니다.
6. 다음과 같이 대기열 서비스 역할에 새 섹션을 추가합니다. *amzn-s3-demo-bucket* 및 *111122223333*을 자체 버킷 및 계정으로 바꿉니다.

```
{
   "Effect": "Allow",
   "Sid": "CustomCondaChannelReadWrite",
   "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:DeleteObject",
    "s3:ListBucket",
    "s3:GetBucketLocation"
   ],
   "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/Conda/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceAccount": "111122223333"
    }
```

```
  }
 },
```

패키지 빌드 작업 제출

패키지 빌드 대기열을 생성하고 대기열 권한을 구성한 후 작업을 제출하여 conda 패키지를 빌드할 수 있습니다. 의 Deadline Cloud 샘플 `submit-package-job` 리포지토리의 스크립트는 conda 레시피에 대한 빌드 작업을 GitHub 제출합니다.

다음 항목이 필요합니다.

- 워크스테이션에 설치된 Deadline Cloud CLI입니다.
- 활성 AWS Deadline Cloud Monitor(Deadline Cloud Monitor) 로그인 세션입니다.
- Deadline Cloud 샘플 리포지토리의 복제본입니다.

패키지 빌드 작업을 제출하려면

1. Deadline Cloud 구성 GUI를 열고 패키지 빌드 대기열에 기본 팜과 대기열을 설정합니다.

   ```
   deadline config gui
   ```

2. 샘플 리포지토리의 `conda_recipes` 디렉터리로 변경합니다.

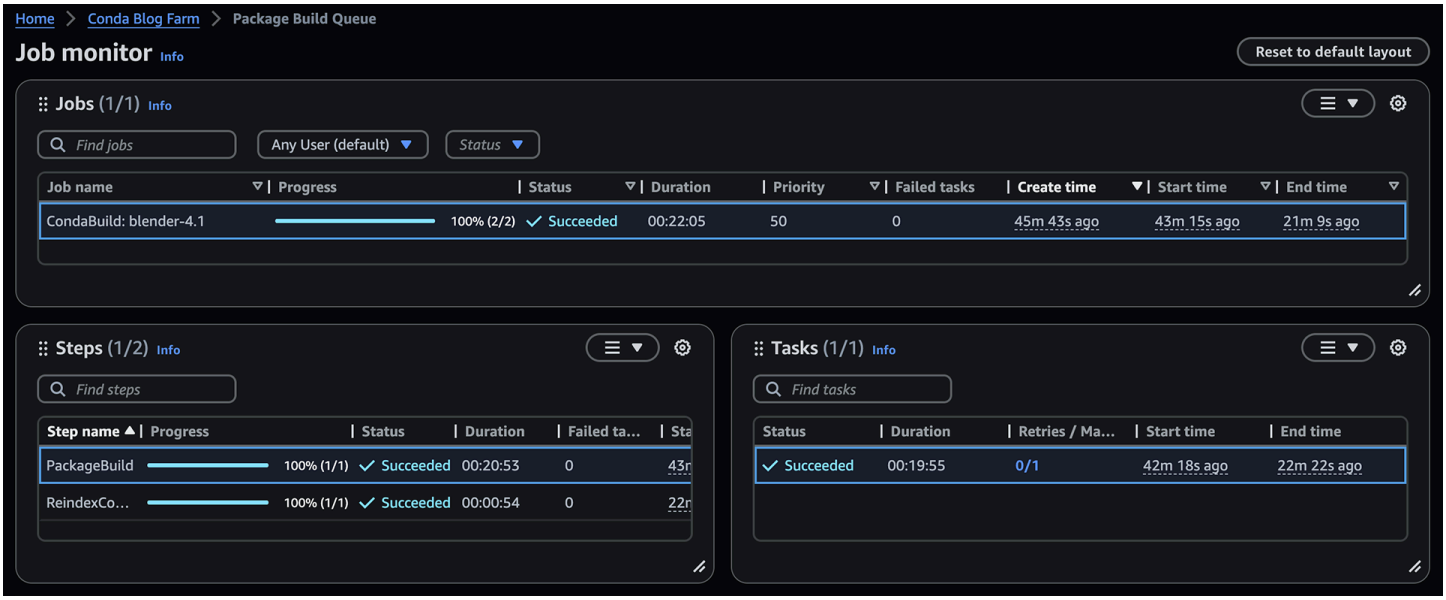
   ```
   cd deadline-cloud-samples/conda_recipes
   ```

3. 레시피 디렉터리를 사용하여 `submit-package-job` 스크립트를 실행합니다. 다음 예제에서는 Blender 4.5 레시피를 빌드합니다.

   ```
   ./submit-package-job blender-4.5/
   ```

   레시피에 아직 다운로드하지 않은 소스 아카이브가 필요한 경우 스크립트는 다운로드 지침을 제공합니다. 아카이브를 다운로드하고 스크립트를 다시 실행합니다.

작업을 제출한 후 Deadline Cloud 모니터를 사용하여 작업의 진행 상황과 상태를 확인합니다.



모니터는 작업의 두 단계, 즉 패키지를 빌드한 다음 conda 채널을 다시 인덱싱하는 단계를 보여줍니다. 패키지 빌드 단계의 작업을 마우스 오른쪽 버튼으로 클릭하고 로그 보기를 선택하면 모니터에 세션 작업이 표시됩니다.

- 첨부 파일 동기화 - 입력 작업 첨부 파일을 복사하거나 가상 파일 시스템을 탑재합니다.
- Conda 시작 - 대기열 환경 작업입니다. 빌드 작업은 conda 패키지를 지정하지 않으므로이 작업은 빠르게 완료됩니다.
- CondaBuild Env 시작 - conda 패키지를 빌드하고 채널을 다시 인덱싱하는 데 필요한 소프트웨어를 사용하여 conda 가상 환경을 생성합니다.
- 작업 실행 - 패키지를 빌드하고 결과를 Amazon S3에 업로드합니다.

작업이 실행되면 Amazon CloudWatch(CloudWatch)로 로그를 전송합니다. 작업이 완료되면 모든 작업에 대한 로그 보기를 선택하여 환경 설정 및 해체에 대한 추가 로그를 확인합니다.

관리자 권한으로 호스트 구성 스크립트 실행

호스트 구성 스크립트를 사용하면 서비스 관리형 플릿 작업자에 대해 소프트웨어 설치와 같은 관리 작업을 수행할 수 있습니다. 이러한 스크립트는 승격된 권한(sudo의 Linux,의 관리자Windows)으로 실행되므로 시스템에 맞게 작업자를 유연하게 구성할 수 있습니다.

Deadline Cloud는 작업자가 `STARTING` 상태가 된 후 작업을 실행하기 전에 스크립트를 실행합니다.

> **Important**
>
> 스크립트는 승격된 권한으로 실행됩니다. 스크립트에 보안 문제가 발생하지 않도록 하는 것은 사용자의 책임입니다.
> 호스트 구성 스크립트를 사용하는 경우 플릿의 상태를 모니터링할 책임이 있습니다.

호스트 구성 스크립트의 일반적인 용도는 다음과 같습니다.

- 관리자 액세스가 필요한 소프트웨어 설치
- Docker 컨테이너 설치
- 와 같은 타사 클라우드 스토리지 솔루션 설치LucidLink. 연습은 M&E 블로그 AWS 용의 Deadline Cloud에 대한 서비스 관리형 플릿 스크립트로 LucidLink 설정을 참조하세요.

콘솔 또는를 사용하여 호스트 구성 스크립트를 생성하고 업데이트할 수 있습니다 AWS CLI.

Console

1. 플릿 세부 정보 페이지에서 구성 탭을 선택합니다.
2. 스크립트 필드에 승격된 권한으로 실행할 스크립트를 입력합니다. 가져오기를 선택하여 워크스테이션에서 스크립트를 로드할 수 있습니다.
3. 스크립트를 실행하기 위한 제한 시간을 초 단위로 설정합니다. 기본값은 300초(5분)입니다.
4. 변경 사항 저장을 선택하여 스크립트를 저장합니다.

Create with CLI

다음 AWS CLI 명령을 사용하여 호스트 구성 스크립트가 있는 플릿을 생성합니다. *## ###* 텍스트를 정보로 바꿉니다.

```
aws deadline create-fleet \
--farm-id farm-12345 \
--display-name "fleet-name" \
--max-worker-count 1 \
--configuration '{
"serviceManagedEc2": {
  "instanceCapabilities": {
    "vCpuCount": {"min": 2},
    "memoryMiB": {"min": 4096},
```

```
    "osFamily": "linux",
    "cpuArchitectureType": "x86_64"
  },
  "instanceMarketOptions": {"type":"spot"}
}
}' \
--role-arn arn:aws:iam::111122223333:role/role-name \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

Update with CLI

다음 AWS CLI 명령을 사용하여 플릿의 호스트 구성 스크립트를 업데이트합니다. `## ###` 텍스트를 정보로 바꿉니다.

```
aws deadline update-fleet \
--farm-id farm-12345 \
--fleet-id fleet-455678 \
--host-configuration '{ "scriptBody": "script body", "scriptTimeoutSeconds": timeout
 value}'
```

다음 스크립트는 다음을 보여줍니다.

- 스크립트에 사용할 수 있는 환경 변수
- 해당 AWS 자격 증명은 쉘에서 작동합니다.
- 스크립트가 승격된 셸에서 실행되고 있는지 여부

Linux

다음 스크립트를 사용하여 스크립트가 `root` 권한으로 실행되고 있음을 표시합니다.

```
# Print environment variables
set
# Check AWS Credentials
aws sts get-caller-identity
```

Windows

다음 PowerShell 스크립트를 사용하여 스크립트가 관리자 권한으로 실행 중임을 표시합니다.

```
Get-ChildItem env: | ForEach-Object { "$($_.Name)=$($_.Value)" }
aws sts get-caller-identity
function Test-AdminPrivileges {
  $currentUser = New-Object
 Security.Principal.WindowsPrincipal([Security.Principal.WindowsIdentity]::GetCurrent())
  $isAdmin =
 $currentUser.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)

  return $isAdmin
}

if (Test-AdminPrivileges) {
  Write-Host "The current PowerShell session is elevated (running as
 Administrator)."
} else {
  Write-Host "The current PowerShell session is not elevated (not running as
 Administrator)."
}
exit 0
```

호스트 구성 스크립트 문제 해결

호스트 구성 스크립트를 실행하는 경우:

- 성공 시: 작업자가 작업을 실행합니다.
- 실패 시(0이 아닌 종료 코드 또는 충돌):
  - 작업자 종료

  플릿은 최신 호스트 구성 스크립트를 사용하여 새 작업자를 자동으로 시작합니다.

스크립트를 모니터링하려면:

1. Deadline Cloud 콘솔에서 플릿 페이지를 엽니다.
2. 작업자 보기를 선택하여 Deadline Cloud 모니터를 엽니다.
3. 모니터 페이지에서 작업자 상태를 봅니다.

> **Tip**
>
> 호스트 구성 스크립트를 테스트할 때 스크립트에서 반복하는 동안 여러 작업자가 시작되지 않도록 플릿의 최대 작업자 수를 1로 설정합니다.

중요 정보:

- 오류로 인해 종료된 작업자는 모니터의 작업자 목록에서 사용할 수 없습니다. CloudWatch Logs를 사용하여 다음 로그 그룹의 작업자 로그를 봅니다.

  ```
  /aws/deadline/farm-XXXXX/fleet-YYYYY
  ```

  해당 로그 그룹 내에서 라는 스트림을 찾습니다`worker-ZZZZZ`.
- CloudWatch Logs는 구성된 보존 기간에 따라 작업자 로그를 보존합니다.

호스트 구성 스크립트 실행 모니터링

호스트 구성 스크립트를 사용하면 Deadline Cloud 작업자를 완벽하게 제어할 수 있습니다. 소프트웨어 패키지를 설치하거나, 운영 체제 파라미터를 재구성하거나, 공유 파일 시스템을 탑재할 수 있습니다. 이 고급 기능과 수천 명의 작업자로 확장할 수 있는 Deadline Cloud의 기능을 사용하면 구성 스크립트가 성공적으로 실행되거나 실패한 시기를 모니터링할 수 있습니다.

호스트 구성 스크립트 실행을 모니터링하려면 다음 솔루션을 사용하는 것이 좋습니다.

CloudWatch Logs 모니터링

모든 플릿 호스트 구성 로그는 플릿의 CloudWatch 로그 그룹, 특히 작업자의 CloudWatch 로그 스트림으로 스트리밍됩니다. 예를 들어 `/aws/deadline/farm-123456789012/fleet-777788889999`는 팜 `123456789012`, 플릿에 대한 로그 그룹입니다`777788889999`.

각 작업자는와 같은 전용 로그 스트림을 프로비저닝합니다`worker-123456789012`. 호스트 구성 로그에는 호스트 구성 스크립트 실행 및 호스트 구성 스크립트 실행 완료, 종료 코드: 0과 같은 로그 배너가 포함됩니다. 스크립트의 종료 코드는 완료된 배너에 포함되며 CloudWatch 도구를 사용하여 쿼리할 수 있습니다.

CloudWatch Logs Insights

CloudWatch Logs Insights는 로그 정보를 분석하는 고급 기능을 제공합니다. 예를 들어 다음 Log Insights 쿼리는 호스트 구성 종료 코드에 대한 구문 분석을 시간별로 정렬합니다.

```
fields @timestamp, @message, @logStream, @log
| filter @message like /Finished running Host Configuration Script/
| parse @message /exit code: (?<exit_code>\d+)/
| display @timestamp, exit_code
| sort @timestamp desc
```

CloudWatch Logs Insights에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서에서 CloudWatch Logs Insights를 사용하여 로그 데이터 분석하기를 참조하세요.

작업자 에이전트 구조화 로깅

Deadline Cloud의 작업자 에이전트는 구조화된 JSON 로그를 CloudWatch에 게시합니다. 작업자 에이전트는 작업자 상태를 분석하기 위한 다양한 구조화된 로그를 제공합니다. 자세한 내용은 GitHub의 Deadline Cloud 작업자 에이전트 로깅을 참조하세요.

구조화된 로그의 속성은 Log Insights의 필드에 압축이 풀립니다. 이 CloudWatch 기능을 사용하여 호스트 구성 시작 실패를 계산하고 분석할 수 있습니다. 예를 들어 개수 및 빈 쿼리를 사용하여 실패 발생 빈도를 확인할 수 있습니다.

```
fields @timestamp, @message, @logStream, @log
| sort @timestamp desc
| filter message like /Worker Agent host configuration failed with exit code/
| stats count(*) by exit_code, bin(1h)
```

지표 및 경보에 대한 CloudWatch 지표 필터

CloudWatch 지표 필터를 설정하여 로그에서 CloudWatch 지표를 생성할 수 있습니다. 지표 필터를 사용하면 호스트 구성 스크립트 실행을 모니터링하기 위한 경보 및 대시보드를 생성할 수 있습니다.

지표 필터를 생성하려면

1. CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택한 다음 로그 그룹을 선택합니다.
3. 플릿의 로그 그룹을 선택합니다.
4. 지표 필터 생성을 선택합니다.

5. 다음 중 하나를 사용하여 필터 패턴을 정의합니다.

   - 성공 지표의 경우:

   ```
   {$.message = "*Worker Agent host configuration succeeded.*"}
   ```

   - 실패 지표의 경우:

   ```
   {$.exit_code != 0 && $.message = "*Worker Agent host configuration failed with
    exit code*"}
   ```

6. 다음을 선택하여 다음 값으로 지표를 생성합니다.

   - 지표 네임스페이스: 지표 네임스페이스(예: **MyDeadlineFarm**)
   - 지표 이름: 요청한 지표 이름(예: **host_config_failure**)
   - 지표 값: **1** (각 인스턴스는 1개)
   - 기본값: 비워 둡니다.
   - 단위: **Count**

지표 필터를 생성한 후 승격된 호스트 구성 실패율에 대한 조치를 취하도록 표준 CloudWatch 경보를 구성하거나 day-to-day 운영 및 모니터링을 위해 CloudWatch 대시보드에 지표를 추가할 수 있습니다.

자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 필터 및 패턴 구문을 참조하세요.

Deadline Cloud에서 소프트웨어 라이선스 사용

Deadline Cloud는 작업에 대한 소프트웨어 라이선스를 제공하는 두 가지 방법을 제공합니다.

- 사용량 기반 라이선스(UBL) -는 플릿이 작업 처리를 사용하는 시간을 기준으로 추적하고 청구합니다. 설정된 라이선스 수가 없으므로 필요에 따라 플릿을 확장할 수 있습니다. UBL은 서비스 관리형 플릿의 표준입니다. 고객 관리형 플릿의 경우 UBL용 Deadline Cloud 라이선스 엔드포인트를 연결할 수 있습니다. UBL은 Deadline Cloud 작업자가 렌더링할 수 있는 라이선스를 제공하며 DCC 애플리케이션에 대한 라이선스를 제공하지 않습니다.
- 기존 라이선스 사용(BYOL) - 서비스 또는 고객 관리형 플릿에서 기존 소프트웨어 라이선스를 사용할 수 있습니다. BYOL을 사용하여 Deadline Cloud 사용량 기반 라이선스에서 지원하지 않는 소프트웨어의 라이선스 서버에 연결할 수 있습니다. 사용자 지정 라이선스 서버에 연결하여 서비스 관리형 플릿에 BYOL을 사용할 수 있습니다.

주제

BYOL과 UBL 결합

작업자가 기존 라이선스를 먼저 사용하고 BYOL 라이선스가 소진되면 자동으로 Deadline Cloud 사용 기반 라이선스로 돌아가도록 BYOL과 UBL을 결합할 수 있습니다. 이 접근 방식은 기존 라이선스 수가 제한적이지만 피크 워크로드 중에 해당 용량을 초과하여 확장해야 하는 경우에 유용합니다.

결합된 라이선스의 작동 방식

결합된 라이선스를 구성할 때 대기열 환경은 BYOL 라이선스 서버가 UBL 라이선스 엔드포인트 앞에 나열되도록 라이선스 환경 변수를 설정합니다. 대부분의 타사 애플리케이션은 환경 변수에 나타나는 순서대로 라이선스 서버를 확인합니다. 작업자가 라이선스를 요청하면 애플리케이션은 먼저 BYOL 라이선스 서버에 문의합니다. BYOL 라이선스를 사용할 수 없는 경우 애플리케이션은 UBL 라이선스 엔드포인트로 돌아갑니다.

에 제공된 BYOL 대기열 환경 템플릿은이 폴백 동작을 자동으로 서비스 관리형 플릿을 사용자 지정 라이선스 서버에 연결 구성합니다. 대기열 환경의 Python 스크립트는 BYOL 라이선스 서버 주소를 기존

UBL 라이선스 환경 변수에 우선합니다. 결합된 라이선스를 사용하려면 대체 동작을 원하는 제품에 대해 대기열 환경 스크립트의 UBL 섹션을 유지합니다.

특정 제품에 대해 UBL 대체 없이 BYOL만 사용하려면 스크립트에서 해당 제품의 UBL 섹션을 제거하고 라이선스 환경 변수를 대기열 환경의 `variables` 섹션에 직접 추가합니다. 예를 들어, Cinema 4D에는 BYOL만 사용하려면 스크립트에서 Cinema 4D 섹션을 제거하고 `variables` 섹션에 `g_licenseServerRLM: 127.0.0.1:7057`를 추가합니다.

예: UBL 폴백과 함께 BYOL Cinema 4D 라이선스 사용

온프레미스 네트워크의 라이선스 서버에 기존 Cinema 4D 라이선스가 있는 스튜디오를 생각해 보세요. 스튜디오는 Deadline Cloud 렌더링에 이러한 라이선스를 사용하려고 하지만 모든 BYOL 라이선스가 사용 중일 때 UBL로 돌아와 라이선스 수를 초과하여 확장하려고 합니다.

이 설정을 구성하려면의 단계에 따라 대기열 환경 템플릿을 다음과 서비스 관리형 플릿을 사용자 지정 라이선스 서버에 연결 같이 변경합니다.

UBL 폴백을 사용하여 BYOL Cinema 4D 라이선스를 구성하려면

1. `LicenseInstanceId` 파라미터를 Cinema 4D 라이선스 서버에 액세스할 수 있는 라이선스 서버 또는 프록시의 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 ID로 설정합니다.
2. 포트를 포함하도록 `LicensePorts` 파라미터를 설정합니다`7057`(Cinema 4D RLM 라이선스 포트).
3. Python 스크립트에서 BYOL 서버 앞에 있는 Cinema 4D 섹션을 UBL 구성으로 유지합니다.

   ```
   # Cinema4D
   os.environ["g_licenseServerRLM"] = f"127.0.0.1:7057;
   {os.environ.get('g_licenseServerRLM', '')}"
   print(f"openjd_env: g_licenseServerRLM={os.environ['g_licenseServerRLM']}")
   ```

   이 구성은를 `g_licenseServerRLM`로 설정합니다`127.0.0.1:7057;`*`UBL_endpoint`*`:7057`. Cinema 4D는 `127.0.0.1:7057` 처음에 BYOL 서버를 확인합니다. 라이선스를 사용할 수 없는 경우 Cinema 4D는 UBL 엔드포인트로 돌아갑니다.
4. 구성을 정리하기 위해 사용하지 않는 제품(예: Arnold, Nuke 또는 SideFX)의 섹션을 제거합니다.

UBL 폴백 없이 BYOL만 사용하는 다른 제품도 있는 경우 해당 라이선스 환경 변수를 대기열 환경의 `variables` 섹션에 직접 추가하고 Python 스크립트에서 해당 섹션을 제거합니다.

결합된 라이선스에 대한 고려 사항

결합된 라이선스를 사용할 때는 다음 사항을 고려해야 합니다.

- 일부 애플리케이션은 단일 환경 변수에서 여러 라이선스 서버를 지원하지 않습니다. 예를 들어 V-Ray는 대신 XML 구성 파일을 사용합니다. 대기열 환경 템플릿은 V-Ray 구성을 별도로 처리합니다. 자세한 내용은의 대기열 환경 템플릿에서 V-Ray 섹션을 참조하세요서비스 관리형 플릿을 사용자 지정 라이선스 서버에 연결.
- 환경 변수의 라이선스 서버 순서에 따라 우선 순위가 결정됩니다. 기존 라이선스가 UBL 라이선스보다 먼저 사용되도록 BYOL 서버를 먼저 나열합니다.
- Windows 작업자의 경우 환경 변수의 라이선스 서버 항목을 콜론(`;`) 대신 세미콜론()으로 구분합니다`:`. 라이선스 환경 변수 구성에 대한 자세한 내용은 섹션을 참조하세요고객 관리형 플릿을 라이선스 엔드포인트에 연결.
- 고객 관리형 플릿에서 UBL 폴백을 사용하려면 라이선스 엔드포인트를 설정합니다. 자세한 내용은 고객 관리형 플릿을 라이선스 엔드포인트에 연결 단원을 참조하십시오.

서비스 관리형 플릿을 사용자 지정 라이선스 서버에 연결

Deadline Cloud 서비스 관리형 플릿과 함께 사용할 자체 라이선스 서버를 가져올 수 있습니다. 자체 라이선스를 가져오려면 팜의 대기열 환경을 사용하여 라이선스 서버를 구성할 수 있습니다. 라이선스 서버를 구성하려면 이미 팜과 대기열이 설정되어 있어야 합니다.

소프트웨어 라이선스 서버에 연결하는 방법은 플릿의 구성과 소프트웨어 공급업체의 요구 사항에 따라 달라집니다. 일반적으로 다음 두 가지 방법 중 하나로 서버에 액세스합니다.

- 라이선스 서버로 직접 이동합니다. 작업자는 인터넷을 사용하여 소프트웨어 공급업체의 라이선스 서버에서 라이선스를 받습니다. 모든 작업자가 서버에 연결할 수 있어야 합니다.
- 라이선스 프록시를 통해. 작업자는 로컬 네트워크의 프록시 서버에 연결합니다. 프록시 서버만 인터넷을 통해 공급업체의 라이선스 서버에 연결할 수 있습니다.

아래 지침에 따라 Amazon EC2 Systems Manager(SSM)를 사용하여 작업자 인스턴스에서 라이선스 서버 또는 프록시 인스턴스로 포트를 전달합니다. 아래 예제에서는 라이선스 서버가 라이선스를 제공할 수 없는 경우 Deadline Cloud의 사용량 기반 라이선스가 사용됩니다. 라이선스를 소진한 후 사용량 기반 라이선스를 사용하지 않으려는 파이프라인 또는 제품에 적용되지 않는 섹션을 제거합니다.

주제

1단계: 대기열 환경 구성

라이선스 서버에 액세스하도록 대기열의 대기열 환경을 구성할 수 있습니다. 먼저 다음 방법 중 하나를 사용하여 라이선스 서버 액세스로 구성된 AWS 인스턴스가 있는지 확인합니다.

- 라이선스 서버 - 인스턴스는 라이선스 서버를 직접 호스팅합니다.
- 라이선스 프록시 - 인스턴스는 라이선스 서버에 대한 네트워크 액세스 권한을 가지며 라이선스 서버 포트를 라이선스 서버로 전달합니다. 라이선스 프록시 인스턴스를 구성하는 방법에 대한 자세한 내용은 섹션을 참조하세요2단계: (선택 사항) 라이선스 프록시 인스턴스 설정.

라이선스 환경 변수 구성에 대한 자세한 내용은 섹션을 참조하세요3단계: 렌더링 애플리케이션을 엔드포인트에 연결. 사용자 지정 라이선스 서버 설정의 경우 라이선스 서버 주소는 Amazon VPC 엔드포인트 대신 localhost로 유지됩니다.

대기열 역할에 필요한 권한을 추가하려면

1. Deadline Cloud 콘솔에서 대시보드로 이동을 선택합니다.
2. 대시보드에서 팜을 선택한 다음 구성할 대기열을 선택합니다.
3. 대기열 세부 정보 > 서비스 역할에서 역할을 선택합니다.
4. 권한 추가를 선택한 다음 인라인 정책 생성을 선택합니다.
5. JSON 정책 편집기를 선택한 다음 다음 텍스트를 복사하여 편집기에 붙여 넣습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
```

```
            ],
            "Resource": [
                "arn:aws:ssm:us-east-1::document/AWS-
StartPortForwardingSession",
                "arn:aws:ec2:us-east-1:111122223333:instance/instance_id"
            ]
        }
    ]
}
```

6. 새 정책을 저장하기 전에 정책 텍스트에서 다음 값을 바꿉니다.
   - 을 팜이 위치한 AWS 리전region으로 바꿉니다.
   - 를 사용 중인 라이선스 서버 또는 프록시 인스턴스의 인스턴스 IDinstance_id로 바꿉니다.
   - account_id을 팜이 포함된 AWS 계정 번호로 바꿉니다.
7. 다음을 선택합니다.
8. 정책 이름에를 입력합니다**LicenseForwarding**.
9. 정책 생성을 선택하여 변경 사항을 저장하고 필요한 권한이 있는 정책을 생성합니다.

대기열에 새 대기열 환경을 추가하려면

1. 아직 대시보드로 이동하지 않았다면 기한 클라우드 콘솔에서 대시보드로 이동을 선택합니다.
2. 대시보드에서 팜을 선택한 다음 구성할 대기열을 선택합니다.
3. 대기열 환경 > 작업 > YAML을 사용하여 새로 생성을 선택합니다.
4. 다음 텍스트를 복사하여 YAML 스크립트 편집기에 붙여 넣습니다.

   Windows

```
specificationVersion: "environment-2023-09"
parameterDefinitions:
 - name: LicenseInstanceId
   type: STRING
   description: >
    The Instance ID of the license server/proxy instance
   default: ""
 - name: LicenseInstanceRegion
   type: STRING
   description: >
```

```
    The region containing this farm
   default: ""
 - name: LicensePorts
   type: STRING
   description: >
    Comma-separated list of ports to be forwarded to the license server/proxy
    instance. Example: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
   default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
environment:
 name: BYOL License Forwarding
 variables:
  example_LICENSE: 2701@localhost
 script:
  actions:
   onEnter:
    command: bash
    args: [ "{{Env.File.Enter}}" ]
   onExit:
    command: bash
    args: [ "{{Env.File.Exit}}" ]
  embeddedFiles:
   - name: Enter
     type: TEXT
     runnable: True
     data: |
      curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/
windows/SessionManagerPlugin.zip" -o "{{Session.WorkingDirectory}}/ssm-
plugin.zip"
      powershell -Command "Expand-Archive -Path '{{Session.WorkingDirectory}}/
ssm-plugin.zip' -DestinationPath '{{Session.WorkingDirectory}}/ssm-plugin'
 -Force; Expand-Archive -Path '{{Session.WorkingDirectory}}/ssm-plugin/
package.zip' -DestinationPath '{{Session.WorkingDirectory}}/ssm-plugin/package'
 -Force"
      conda activate
      python "{{Env.File.StartSession}}" "{{Session.WorkingDirectory}}/ssm-
plugin/package/bin/session-manager-plugin.exe"
   - name: Exit
     type: TEXT
     runnable: True
     data: |
      echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
      for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
   - name: StartSession
     type: TEXT
```

```
    data: |
     import boto3
     import json
     import subprocess
     import sys
     import os
     import tempfile

     instance_id = "{{Param.LicenseInstanceId}}"
     region = "{{Param.LicenseInstanceRegion}}"
     license_ports_list = "{{Param.LicensePorts}}".split(",")

     ssm_client = boto3.client("ssm", region_name=region)
     pids = []

     for port in license_ports_list:
       session_response = ssm_client.start_session(
         Target=instance_id,
         DocumentName="AWS-StartPortForwardingSession",
         Parameters={"portNumber": [port], "localPortNumber": [port]}
       )

       cmd = [
         sys.argv[1],
         json.dumps(session_response),
         region,
         "StartSession",
         "",
         json.dumps({"Target": instance_id}),
         f"https://ssm.{region}.amazonaws.com"
       ]

       process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
       pids.append(process.pid)
       print(f"SSM Port Forwarding Session started for port {port}")

     print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in pids)}")

     # Enabling UBL after the BYOL has run out requires prepending the BYOL
configuration to the existing license setup
     # Remove the sections that do not apply to your pipeline, or you do not
want to use UBL after exhausting the BYOL licenses.
     # The port numbers used may not match what your license server is serving.
```

```
      # Arnold
      os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost;
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
      print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

      # Cinema4D
      os.environ["g_licenseServerRLM"] = f"localhost:7057;
{os.environ.get('g_licenseServerRLM', '')}"
      print(f"openjd_env:
 g_licenseServerRLM={os.environ['g_licenseServerRLM']}")

      # Nuke
      os.environ["foundry_LICENSE"] = f"6101@localhost;
{os.environ.get('foundry_LICENSE', '')}"
      print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

      # SideFX
      os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
      print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

      # Redshift and Red Giant
      os.environ["redshift_LICENSE"] = f"7054@localhost;7055@localhost;
{os.environ.get('redshift_LICENSE', '')}"
      print(f"openjd_env: redshift_LICENSE={os.environ['redshift_LICENSE']}")

      # V-Ray doesn't support multiple license servers in a single environment
 variable
      # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
      vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
      xml_content = """<VRLClient>
        <LicServer>
          <Host>localhost</Host>
          <Port>30304</Port>"""

      if vray_license and vray_license.startswith('licset://'):
          server_parts = vray_license.removeprefix('licset://').split(':')
          if len(server_parts) >= 2:
              xml_content += f"""
          <Host1>{server_parts[0]}</Host1>
          <Port1>{server_parts[1]}</Port1>"""
```

```
    xml_content += """
        <User></User>
        <Pass></Pass>
      </LicServer>
    </VRLClient>"""

    temp_dir = tempfile.gettempdir()
    xml_path = os.path.join(temp_dir, 'vrlclient.xml')

    with open(xml_path, 'w') as f:
        f.write(xml_content)

    os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
    print(f"openjd_env:
VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

    # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the vrlclient.xml
file is used.
    os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
    print(f"openjd_env:
VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

    # Print out the created xml file's contents
    print(f"V-Ray configuration file: {xml_path}")
    with open(xml_path, 'r') as f:
        print(f"{f.read()}")
```

Linux

```
specificationVersion: "environment-2023-09"
parameterDefinitions:
 - name: LicenseInstanceId
   type: STRING
   description: >
    The Instance ID of the license server/proxy instance
   default: ""
 - name: LicenseInstanceRegion
   type: STRING
   description: >
    The region containing this farm
```

```
    default: ""
  - name: LicensePorts
    type: STRING
    description: >
     Comma-separated list of ports to be forwarded to the license server/proxy
     instance. Example: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
    default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
environment:
 name: BYOL License Forwarding
 variables:
  example_LICENSE: 2701@localhost
 script:
  actions:
   onEnter:
    command: bash
    args: [ "{{Env.File.Enter}}" ]
   onExit:
    command: bash
    args: [ "{{Env.File.Exit}}" ]
  embeddedFiles:
   - name: Enter
     type: TEXT
     runnable: True
     data: |
      curl https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio -iv
 --to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
 {{Session.WorkingDirectory}}/session-manager-plugin
      chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
      conda activate
      python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/session-
manager-plugin
   - name: Exit
     type: TEXT
     runnable: True
     data: |
      echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
      for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
   - name: StartSession
     type: TEXT
     data: |
      import boto3
      import json
      import subprocess
```

```
      import sys
      import os
      import tempfile

      instance_id = "{{Param.LicenseInstanceId}}"
      region = "{{Param.LicenseInstanceRegion}}"
      license_ports_list = "{{Param.LicensePorts}}".split(",")

      ssm_client = boto3.client("ssm", region_name=region)
      pids = []

      for port in license_ports_list:
        session_response = ssm_client.start_session(
          Target=instance_id,
          DocumentName="AWS-StartPortForwardingSession",
          Parameters={"portNumber": [port], "localPortNumber": [port]}
        )

        cmd = [
          sys.argv[1],
          json.dumps(session_response),
          region,
          "StartSession",
          "",
          json.dumps({"Target": instance_id}),
          f"https://ssm.{region}.amazonaws.com"
        ]

        process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
 stderr=subprocess.DEVNULL)
        pids.append(process.pid)
        print(f"SSM Port Forwarding Session started for port {port}")

      print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in pids)}")

      # Enabling UBL after the BYOL has run out requires prepending the BYOL
 configuration to the existing license setup
      # Remove the sections that do not apply to your pipeline, or you do not
 want to use UBL after exhausting the BYOL licenses.
      # The port numbers used may not match what your license server is serving.

      # Arnold
      os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost:
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
```

```
      print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

      # Nuke
      os.environ["foundry_LICENSE"] = f"6101@localhost:
{os.environ.get('foundry_LICENSE', '')}"
      print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

      # SideFX
      os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
      print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

      # Redshift and Red Giant
      os.environ["redshift_LICENSE"] = f"7054@localhost:7055@localhost:
{os.environ.get('redshift_LICENSE', '')}"
      print(f"openjd_env: redshift_LICENSE={os.environ['redshift_LICENSE']}")

      # V-Ray doesn't support multiple license servers in a single environment
 variable
      # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
      vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
      xml_content = """<VRLClient>
        <LicServer>
          <Host>localhost</Host>
          <Port>30304</Port>"""

      if vray_license and vray_license.startswith('licset://'):
          server_parts = vray_license.removeprefix('licset://').split(':')
          if len(server_parts) >= 2:
              xml_content += f"""
          <Host1>{server_parts[0]}</Host1>
          <Port1>{server_parts[1]}</Port1>"""

      xml_content += """
          <User></User>
          <Pass></Pass>
        </LicServer>
      </VRLClient>"""

      temp_dir = tempfile.gettempdir()
      xml_path = os.path.join(temp_dir, 'vrlclient.xml')
```

```
    with open(xml_path, 'w') as f:
        f.write(xml_content)

    os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
    print(f"openjd_env:
VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

    # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the vrlclient.xml
file is used.
    os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
    print(f"openjd_env:
VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

    # Print out the created xml file's contents
    print(f"V-Ray configuration file: {xml_path}")
    with open(xml_path, 'r') as f:
        print(f"{f.read()}")
```

5. 대기열 환경을 저장하기 전에 필요에 따라 환경 텍스트를 다음과 같이 변경합니다.

   - 환경을 반영하도록 다음 파라미터의 기본값을 업데이트합니다.
     - LicenseInstanceID - 라이선스 서버 또는 프록시 인스턴스의 Amazon EC2 인스턴스 ID입니다.
     - LicenseInstanceRegion - 팜이 포함된 AWS 리전
     - LicensePorts - 라이선스 서버 또는 프록시 인스턴스로 전달할 쉼표로 구분된 포트 목록(예: 2700,2701)
   - 기존 보유 라이선스 사용(BYOL)이 소진된 후 사용량 기반 라이선스(UBL)를 사용하려면 해당 포트가 라이선스 서버에 맞는지 확인하세요. BYOL이 소진된 후 UBL을 사용하지 않으려면 필요한 라이선스 환경 변수를 변수 섹션에 추가합니다.

     이러한 변수는 DCCs 라이선스 서버 포트의 localhost로 전달해야 합니다. 예를 들어 Foundry 라이선스 서버가 포트 6101에서 수신 대기 중인 경우 변수를 로 추가합니다**foundry_LICENSE: 6101@localhost**.
6. (선택 사항) Priority를 0으로 설정하거나 여러 대기열 환경 간에 우선 순위를 다르게 정렬하도록 변경할 수 있습니다.
7. 대기열 환경 생성을 선택하여 새 환경을 저장합니다.

   대기열 환경이 설정되면이 대기열에 제출된 작업은 구성된 라이선스 서버에서 라이선스를 검색합니다.

2단계: (선택 사항) 라이선스 프록시 인스턴스 설정

라이선스 서버를 사용하는 대신 라이선스 프록시를 사용할 수 있습니다. 라이선스 프록시를 생성하려면 라이선스 서버에 대한 네트워크 액세스 권한이 있는 새 Amazon Linux 2023 인스턴스를 생성합니다. 필요한 경우 VPN 연결을 사용하여이 액세스를 구성할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPN 연결을 참조하세요.

Deadline Cloud에 대한 라이선스 프록시 인스턴스를 설정하려면이 절차의 단계를 따릅니다. 이 새 인스턴스에서 다음 구성 단계를 수행하여 라이선스 서버로 라이선스 트래픽을 전달할 수 있도록 합니다.

1. HAProxy 패키지를 설치하려면를 입력합니다.

```
sudo yum install haproxy
```

2. /etc/haproxy/haproxy.cfg 구성 파일의 listen license-server 섹션을 다음과 같이 업데이트합니다.

   a. LicensePort1 및 LicensePort2를 라이선스 서버로 전달할 포트 번호로 바꿉니다. 필요한 수의 포트를 수용할 수 있도록 쉼표로 구분된 값을 추가하거나 제거합니다.

   b. LicenseServerHost를 라이선스 서버의 호스트 이름 또는 IP 주소로 바꿉니다.

```
lobal
    log         127.0.0.1 local2
    chroot      /var/lib/haproxy
    user        haproxy
    group       haproxy
    daemon

defaults
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s

listen license-server
     bind *:LicensePort1,*:LicensePort2
     server license-server LicenseServerHost
```

3. HAProxy 서비스를 활성화하고 시작하려면 다음 명령을 실행합니다.

```
sudo systemctl enable haproxy
sudo service haproxy start
```

단계를 완료한 후 전달 대기열 환경에서 localhost로 전송된 라이선스 요청은 지정된 라이선스 서버로 전달되어야 합니다.

3단계: CloudFormation 템플릿 설정

CloudFormation 템플릿을 사용하여 자체 라이선스를 사용하도록 전체 팜을 구성할 수 있습니다.

1. 다음 단계에서 제공된 템플릿을 수정하여 BYOLQueueEnvironment의 변수 섹션에 필요한 라이선스 환경 변수를 추가합니다.
2. 다음 CloudFormation 템플릿을 사용합니다.

```
AWSTemplateFormatVersion: 2010-09-09
Description: "Create &ADC; resources for BYOL"

Parameters:
  LicenseInstanceId:
    Type: AWS::EC2::Instance::Id
    Description: Instance ID for the license server/proxy instance
  LicensePorts:
    Type: String
    Description: Comma-separated list of ports to forward to the license instance

Resources:
  JobAttachmentBucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Sub byol-example-ja-bucket-${AWS::AccountId}-${AWS::Region}
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256

  Farm:
    Type: AWS::Deadline::Farm
    Properties:
      DisplayName: BYOLFarm
```

```
  QueuePolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLQueuePolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - s3:GetObject
              - s3:PutObject
              - s3:ListBucket
              - s3:GetBucketLocation
            Resource:
              - !Sub ${JobAttachmentBucket.Arn}
              - !Sub ${JobAttachmentBucket.Arn}/job-attachments/*
            Condition:
              StringEquals:
                aws:ResourceAccount: !Sub ${AWS::AccountId}
          - Effect: Allow
            Action: logs:GetLogEvents
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
          - Effect: Allow
            Action:
              - s3:ListBucket
              - s3:GetObject
            Resource:
              - "*"
            Condition:
              ArnLike:
                s3:DataAccessPointArn:
                  - arn:aws:s3:*:*:accesspoint/deadline-software-*
              StringEquals:
                s3:AccessPointNetworkOrigin: VPC

  BYOLSSMPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLSSMPolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
```

```
          - Effect: Allow
            Action:
              - ssm:StartSession
            Resource:
              - !Sub arn:aws:ssm:${AWS::Region}::document/AWS-
StartPortForwardingSession
              - !Sub arn:aws:ec2:${AWS::Region}:${AWS::AccountId}:instance/
${LicenseInstanceId}


  WorkerPolicy:
    Type: AWS::IAM::ManagedPolicy
    Properties:
      ManagedPolicyName: BYOLWorkerPolicy
      PolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - logs:CreateLogStream
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*
            Condition:
              ForAnyValue:StringEquals:
                aws:CalledVia:
                  - deadline.amazonaws.com
          - Effect: Allow
            Action:
              - logs:PutLogEvents
              - logs:GetLogEvents
            Resource: !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-
group:/aws/deadline/${Farm.FarmId}/*


  QueueRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: BYOLQueueRole
      ManagedPolicyArns:
        - !Ref QueuePolicy
        - !Ref BYOLSSMPolicy
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
```

```
          - Effect: Allow
            Action:
              - sts:AssumeRole
            Principal:
              Service:
                - credentials.deadline.amazonaws.com
                - deadline.amazonaws.com
            Condition:
              StringEquals:
                aws:SourceAccount: !Sub ${AWS::AccountId}
              ArnEquals:
                aws:SourceArn: !Ref Farm

  WorkerRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: BYOLWorkerRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AWSDeadlineCloud-FleetWorker
        - !Ref WorkerPolicy
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Action:
              - sts:AssumeRole
            Principal:
              Service: credentials.deadline.amazonaws.com


  Queue:
    Type: AWS::Deadline::Queue
    Properties:
      DisplayName: BYOLQueue
      FarmId: !GetAtt Farm.FarmId
      RoleArn: !GetAtt QueueRole.Arn
      JobRunAsUser:
        Posix:
          Group: ""
          User: ""
        RunAs: WORKER_AGENT_USER
      JobAttachmentSettings:
        RootPrefix: job-attachments
        S3BucketName: !Ref JobAttachmentBucket
```

```
Fleet:
  Type: AWS::Deadline::Fleet
  Properties:
    DisplayName: BYOLFleet
    FarmId: !GetAtt Farm.FarmId
    MinWorkerCount: 1
    MaxWorkerCount: 2
    Configuration:
      ServiceManagedEc2:
        InstanceCapabilities:
          VCpuCount:
            Min: 4
            Max: 16
          MemoryMiB:
            Min: 4096
            Max: 16384
          OsFamily: LINUX
          CpuArchitectureType: x86_64
        InstanceMarketOptions:
          Type: on-demand
    RoleArn: !GetAtt WorkerRole.Arn

QFA:
  Type: AWS::Deadline::QueueFleetAssociation
  Properties:
    FarmId: !GetAtt Farm.FarmId
    FleetId: !GetAtt Fleet.FleetId
    QueueId: !GetAtt Queue.QueueId

CondaQueueEnvironment:
  Type: AWS::Deadline::QueueEnvironment
  Properties:
    FarmId: !GetAtt Farm.FarmId
    Priority: 5
    QueueId: !GetAtt Queue.QueueId
    TemplateType: YAML
    Template: |
      specificationVersion: 'environment-2023-09'
      parameterDefinitions:
      - name: CondaPackages
        type: STRING
        description: >
```

```
            This is a space-separated list of conda package match specifications to
 install for the job.
            E.g. "blender=3.6" for a job that renders frames in Blender 3.6.

            See https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/
pkg-specs.html#package-match-specifications
          default: ""
          userInterface:
            control: LINE_EDIT
            label: Conda Packages
        - name: CondaChannels
          type: STRING
          description: >
            This is a space-separated list of conda channels from which to install
 packages. &ADC; SMF packages are
            installed from the "deadline-cloud" channel that is configured by
 &ADC;.

            Add "conda-forge" to get packages from the https://conda-forge.org/
 community, and "defaults" to get packages
            from Anaconda Inc (make sure your usage complies with https://
www.anaconda.com/terms-of-use).
          default: "deadline-cloud"
          userInterface:
            control: LINE_EDIT
            label: Conda Channels
        environment:
          name: Conda
          script:
            actions:
              onEnter:
                command: "conda-queue-env-enter"
                args: ["{{Session.WorkingDirectory}}/.env", "--packages",
 "{{Param.CondaPackages}}", "--channels", "{{Param.CondaChannels}}"]
              onExit:
                command: "conda-queue-env-exit"

  BYOLQueueEnvironment:
    Type: AWS::Deadline::QueueEnvironment
    Properties:
      FarmId: !GetAtt Farm.FarmId
      Priority: 10
      QueueId: !GetAtt Queue.QueueId
      TemplateType: YAML
```

```
      Template: !Sub |
        specificationVersion: "environment-2023-09"
        parameterDefinitions:
        - name: LicenseInstanceId
          type: STRING
          description: >
            The Instance ID of the license server/proxy instance
          default: ""
        - name: LicenseInstanceRegion
          type: STRING
          description: >
            The region containing this farm
          default: ""
        - name: LicensePorts
          type: STRING
          description: >
            Comma-separated list of ports to be forwarded to the license server/
proxy
            instance. Example:
 "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
          default: "2701,2702,7075,2703,6101,1715,1716,1717,7054,7055,30304"
        environment:
        name: BYOL License Forwarding
        variables:
          example_LICENSE: 2701@localhost
        script:
          actions:
          onEnter:
            command: bash
            args: [ "{{Env.File.Enter}}" ]
          onExit:
            command: bash
            args: [ "{{Env.File.Exit}}" ]
          embeddedFiles:
          - name: Enter
            type: TEXT
            runnable: True
            data: |
              curl https://s3.amazonaws.com/session-manager-downloads/plugin/
latest/linux_64bit/session-manager-plugin.rpm -Ls | rpm2cpio - | cpio -iv
 --to-stdout ./usr/local/sessionmanagerplugin/bin/session-manager-plugin >
 {{Session.WorkingDirectory}}/session-manager-plugin
              chmod +x {{Session.WorkingDirectory}}/session-manager-plugin
              conda activate
```

```
              python {{Env.File.StartSession}} {{Session.WorkingDirectory}}/
session-manager-plugin
          - name: Exit
            type: TEXT
            runnable: True
            data: |
              echo Killing SSM Manager Plugin PIDs: $BYOL_SSM_PIDS
              for pid in ${BYOL_SSM_PIDS//,/ }; do kill $pid; done
          - name: StartSession
            type: TEXT
            data: |
              import boto3
              import json
              import subprocess
              import sys
              import os
              import tempfile

              instance_id = "{{Param.LicenseInstanceId}}"
              region = "{{Param.LicenseInstanceRegion}}"
              license_ports_list = "{{Param.LicensePorts}}".split(",")

              ssm_client = boto3.client("ssm", region_name=region)
              pids = []

              for port in license_ports_list:
                session_response = ssm_client.start_session(
                  Target=instance_id,
                  DocumentName="AWS-StartPortForwardingSession",
                  Parameters={"portNumber": [port], "localPortNumber": [port]}
                )

                cmd = [
                  sys.argv[1],
                  json.dumps(session_response),
                  region,
                  "StartSession",
                  "",
                  json.dumps({"Target": instance_id}),
                  f"https://ssm.{region}.amazonaws.com"
                ]

                process = subprocess.Popen(cmd, stdout=subprocess.DEVNULL,
 stderr=subprocess.DEVNULL)
```

```
                pids.append(process.pid)
                print(f"SSM Port Forwarding Session started for port {port}")

            print(f"openjd_env: BYOL_SSM_PIDS={','.join(str(pid) for pid in
 pids)}")

            # Enabling UBL after the "bring your own license" (BYOL) has run out
 requires prepending the BYOL configuration to the existing license setup
            # Remove the sections that do not apply to your pipeline, or you do
 not want to use UBL after exhausting the BYOL licenses.
            # The port numbers used may not match what your license server is
 serving.

            # Arnold
            os.environ["ADSKFLEX_LICENSE_FILE"] = f"2701@localhost:
{os.environ.get('ADSKFLEX_LICENSE_FILE', '')}"
            print(f"openjd_env:
 ADSKFLEX_LICENSE_FILE={os.environ['ADSKFLEX_LICENSE_FILE']}")

            # Nuke
            os.environ["foundry_LICENSE"] = f"6101@localhost:
{os.environ.get('foundry_LICENSE', '')}"
            print(f"openjd_env: foundry_LICENSE={os.environ['foundry_LICENSE']}")

            # SideFX
            os.environ["SESI_LMHOST"] = f"localhost:1715;
{os.environ.get('SESI_LMHOST', '')}"
            print(f"openjd_env: SESI_LMHOST={os.environ['SESI_LMHOST']}")

            # Redshift and Red Giant
            os.environ["redshift_LICENSE"] = f"7054@localhost:7055@localhost:
{os.environ.get('redshift_LICENSE', '')}"
            print(f"openjd_env:
 redshift_LICENSE={os.environ['redshift_LICENSE']}")

            # V-Ray doesn't support multiple license servers in a single
 environment variable
            # See https://documentation.chaos.com/space/LIC5/125050770/Sharing+a
+License+Configuration+in+a+Network
            vray_license = os.environ.get('VRAY_AUTH_CLIENT_SETTINGS', '')
            xml_content = """<VRLClient>
              <LicServer>
                <Host>localhost</Host>
                <Port>30304</Port>"""
```

```
            if vray_license and vray_license.startswith('licset://'):
                server_parts = vray_license.removeprefix('licset://').split(':')
                if len(server_parts) >= 2:
                    xml_content += f"""
                <Host1>{server_parts[0]}</Host1>
                <Port1>{server_parts[1]}</Port1>"""

            xml_content += """
                <User></User>
                <Pass></Pass>
              </LicServer>
            </VRLClient>"""

            temp_dir = tempfile.gettempdir()
            xml_path = os.path.join(temp_dir, 'vrlclient.xml')

            with open(xml_path, 'w') as f:
                f.write(xml_content)

            os.environ["VRAY_AUTH_CLIENT_FILE_PATH"] = temp_dir
            print(f"openjd_env:
VRAY_AUTH_CLIENT_FILE_PATH={os.environ['VRAY_AUTH_CLIENT_FILE_PATH']}")

            # Clear the existing VRAY_AUTH_CLIENT_SETTINGS so only the
vrlclient.xml file is used.
            os.environ["VRAY_AUTH_CLIENT_SETTINGS"] = ''
            print(f"openjd_env:
VRAY_AUTH_CLIENT_SETTINGS={os.environ['VRAY_AUTH_CLIENT_SETTINGS']}")

            # Print out the created xml file's contents
            print(f"V-Ray configuration file: {xml_path}")
            with open(xml_path, 'r') as f:
                print(f"{f.read()}")
```

3. CloudFormation 템플릿을 배포할 때 다음 파라미터를 제공합니다.

   - LicenseInstanceID로 LicenseInstanceID 업데이트 Amazon EC2
   - 라이선스 서버 또는 프록시 인스턴스로 전달할 쉼표로 구분된 포트 목록으로 LicensePorts를 업데이트합니다(예: 2700,2701).
   - 템플릿**example_LICENSE: 2700@localhost**에서를 교체하여 라이선스 환경 변수 추가

4. 템플릿을 배포하여 자체 라이선스 가져오기 기능을 사용하여 팜을 설정합니다.

고객 관리형 플릿을 라이선스 엔드포인트에 연결

AWS Deadline Cloud 사용량 기반 라이선스 서버는 일부 타사 제품에 대한 온디맨드 라이선스를 제공합니다. 사용량 기반 라이선스를 사용하면 원하는 대로 비용을 지불할 수 있습니다. 사용한 시간에 대해서만 요금이 부과됩니다. 사용량 기반 라이선스는 Deadline Cloud 작업자가 렌더링할 수 있는 라이선스를 제공하며 DCC 애플리케이션에 대한 라이선스는 제공하지 않습니다.

Deadline Cloud 작업자가 라이선스 서버와 통신할 수 있는 한 Deadline Cloud 사용량 기반 라이선스 서버를 모든 플릿 유형과 함께 사용할 수 있습니다. 라이선스 서버는 서비스 관리형 플릿에 자동으로 설정됩니다. 다음 설정은 고객 관리형 플릿에만 필요합니다.

라이선스 서버를 생성하려면 타사 라이선스에 대한 트래픽을 허용하는 팜의 VPC에 대한 보안 그룹이 필요합니다.

주제

1단계: 보안 그룹 생성

Amazon VPC 콘솔을 사용하여 팜의 VPC에 대한 보안 그룹을 생성합니다. 다음 인바운드 규칙을 허용하도록 보안 그룹을 구성합니다.

- Autodesk Maya 및 Arnold – 2701 - 2702, TCP, IPv4, IPv6
- 시네마 4D – 7057, TCP, IPv4, IPv6
- Foundry Nuke – 6101, TCP, IPv4, IPv6
- Red Giant – 7055, TCP, IPV4
- Redshift – 7054, TCP, IPv4, IPv6
- SideFX Houdini, Mantra 및 Karma – 1715 - 1717, TCP, IPv4, IPv6
- VRay – 30304, TCP, IPV4

각 인바운드 규칙의 소스는 플릿의 작업자 보안 그룹입니다.

보안 그룹 생성에 대한 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 보안 그룹 생성을 참조하세요.

2단계: 라이선스 엔드포인트 설정

라이선스 엔드포인트는 타사 제품의 라이선스 서버에 대한 액세스를 제공합니다. 라이선스 요청은 라이선스 엔드포인트로 전송됩니다. 엔드포인트는 이를 적절한 라이선스 서버로 라우팅합니다. 라이선스 서버는 사용 제한 및 권한을 추적합니다. Deadline Cloud에서 라이선스 엔드포인트를 생성하면 VPC에 AWS PrivateLink 인터페이스 엔드포인트가 프로비저닝됩니다. 이러한 엔드포인트는 표준 AWS PrivateLink 요금에 따라 요금이 청구됩니다. 자세한 내용은 AWS PrivateLink 요금을 참조하십시오.

적절한 권한을 사용하여 라이선스 엔드포인트를 생성할 수 있습니다. 라이선스 엔드포인트를 생성하는 데 필요한 정책은 라이선스 엔드포인트 생성을 허용하는 정책을 참조하세요.

Deadline Cloud 콘솔의 대시보드에서 라이선스 엔드포인트를 생성할 수 있습니다.

1. 왼쪽 탐색 창에서 라이선스 엔드포인트를 선택한 다음 라이선스 엔드포인트 생성을 선택합니다.
2. 라이선스 엔드포인트 생성 페이지에서 다음을 완료합니다.
   - VPC를 선택합니다.
   - Deadline Cloud 작업자를 포함하는 서브넷을 선택합니다. 최대 10개의 서브넷을 선택할 수 있습니다.
   - 1단계에서 생성한 보안 그룹을 선택합니다. 더 복잡한 시나리오를 위해 최대 10개의 보안 그룹을 선택할 수 있습니다.
   - (선택 사항) 새 태그 추가를 선택하고 하나 이상의 태그를 추가합니다. 최대 50개의 태그를 추가할 수 있습니다.
3. 라이선스 엔드포인트 생성을 선택합니다. 라이선스 엔드포인트가 생성되면 라이선스 엔드포인트 페이지에 표시됩니다.
4. 측정된 제품 섹션에서 제품 추가를 선택한 다음 라이선스 엔드포인트에 추가할 제품을 선택합니다. 추가를 선택합니다.

라이선스 엔드포인트에서 제품을 제거하려면 측정된 제품 섹션에서 제품을 선택한 다음 제거를 선택합니다. 확인에서 제거를 다시 선택합니다.

3단계: 렌더링 애플리케이션을 엔드포인트에 연결

라이선스 엔드포인트가 설정된 후 애플리케이션은 타사 라이선스 서버를 사용하는 것과 동일하게 이를 사용합니다. 일반적으로 Microsoft Windows 레지스트리 키와 같은 환경 변수 또는 기타 시스템 설정을 라이선스 서버 포트 및 주소로 설정하여 애플리케이션에 대한 라이선스 서버를 구성합니다.

라이선스 엔드포인트 DNS 이름을 가져오려면 콘솔에서 라이선스 엔드포인트를 선택한 다음 DNS 이름 섹션에서 복사 아이콘을 선택합니다.

구성 예제

Example– Autodesk Maya 및 Arnold

> **Note**
>
> Autodesk Maya와 Arnold를 함께 사용하거나 별도로 사용할 수 있습니다. Autodesk Maya용 포트 2702와 Arnold용 포트 2701을 사용합니다.

Autodesk Maya의 경우 환경 변수를 다음과 `ADSKFLEX_LICENSE_FILE` 같이 설정합니다.

```
2702@VPC_Endpoint_DNS_Name
```

Arnold의 경우 환경 변수를 다음과 `ADSKFLEX_LICENSE_FILE` 같이 설정합니다.

```
2701@VPC_Endpoint_DNS_Name
```

Autodesk Maya 및 Arnold의 경우 환경 변수를 다음과 `ADSKFLEX_LICENSE_FILE` 같이 설정합니다.

```
2702@VPC_Endpoint_DNS_Name:2701@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Windows 작업자의 경우 엔드포인트를 분리하려면 콜론(:) 대신 세미콜론(;)을 사용합니다.

Example- 시네마 4D

환경 변수를 다음과 `g_licenseServerRLM` 같이 설정합니다.

```
VPC_Endpoint_DNS_Name:7057
```

환경 변수를 생성한 후에는 다음과 유사한 명령줄을 사용하여 이미지를 렌더링할 수 있습니다.

```
"C:\Program Files\Maxon Cinema 4D 2025\Commandline.exe" -render ^
    "C:\Users\User\MyC4DFileWithRedshift.c4d" -frame 0 ^
    -oimage "C:\Users\Administrator\User\MyOutputImage.png
```

Example- Foundry Nuke

환경 변수를 다음과 `foundry_LICENSE` 같이 설정합니다.

```
6101@VPC_Endpoint_DNS_Name
```

라이선스가 제대로 작동하는지 테스트하기 위해 터미널에서 Nuke를 실행할 수 있습니다.

```
~/nuke/Nuke14.0v5/Nuke14.0 -x
```

Example- Red Giant

환경 변수를 다음과 `redshift_LICENSE` 같이 설정합니다.

```
7055@VPC_Endpoint_DNS_Name
```

Red Giant와 Redshift의 `redshift_LICENSE` 환경 변수는 동일합니다. 두 애플리케이션을 모두 사용하려면 환경 변수를 다음과 같이 설정할 수 있습니다.

```
7054@VPC_Endpoint_DNS_Name:7055@VPC_Endpoint_DNS_Name
```

> **Note**
>
> Windows 작업자의 경우 엔드포인트를 분리하려면 콜론(:) 대신 세미콜론(;)을 사용합니다.

라이선스가 제대로 작동하는지 테스트하려면 After Effects 및 Red Giant가 설치되어 있어야 합니다. 그런 다음 다음과 유사한 명령을 사용하여 프로젝트를 렌더링할 수 있습니다.

```
C:\Program Files\Adobe\Adobe After Effects 2025\Support Files\aerender.exe -comp "Comp
 1" -project
```

```
          C:\Users\MyUser\myAfterEffectsProjectUsingRedGiant.aep -output
            C:\Users\MyUser\myMovieWithRedGiant.mp4
```

Example- Redshift

환경 변수를 다음과 `redshift_LICENSE` 같이 설정합니다.

```
7054@VPC_Endpoint_DNS_Name
```

환경 변수를 생성한 후에는 다음과 유사한 명령줄을 사용하여 이미지를 렌더링할 수 있습니다.

```
C:\ProgramData\redshift\bin\redshiftCmdLine.exe ^
    C:\demo\proxy\RS_Proxy_Demo.rs ^
    -oip C:\demo\proxy\images
```

Example- SideFX Houdini, Mantra 및 Karma

다음 명령을 실행합니다.

```
/opt/hfs19.5.640/bin/hserver -S
 "http://VPC_Endpoint_DNS_Name:1715;http://VPC_Endpoint_DNS_Name:1716;http://
VPC_Endpoint_DNS_Name:1717;"
```

라이선스가 제대로 작동하는지 테스트하려면 다음 명령을 통해 Houdini 장면을 렌더링할 수 있습니다.

```
/opt/hfs19.5.640/bin/hython ~/forpentest.hip -c "hou.node('/out/mantra1').render()"
```

Example- VRay

환경 변수를 다음과 `VRAY_AUTH_CLIENT_SETTINGS` 같이 설정합니다.

```
licset://VPC_Endpoint_DNS_Name:30304
```

환경 변수를 다음과 `VRAY_AUTH_CLIENT_FILE_PATH` 같이 설정합니다.

```
/null
```

라이선스가 제대로 작동하는지 테스트하려면 다음과 유사한 명령을 사용하여 VRay에서 이미지를 렌더링할 수 있습니다.

```
/usr/Chaos/V-Ray/bin/vray -sceneFile=/root/my_scene.vrscene -display=0
```

4단계: 라이선스 엔드포인트 삭제

고객 관리형 플릿을 삭제할 때는 라이선스 엔드포인트를 삭제해야 합니다. 라이선스 엔드포인트를 삭제하지 않으면 고정 비용이 계속 청구 AWS PrivateLink 됩니다.

Deadline Cloud 콘솔의 대시보드에서 라이선스 엔드포인트를 삭제할 수 있습니다.

1. 왼쪽 탐색 창에서 라이선스 엔드포인트를 선택합니다.
2. 삭제할 엔드포인트를 선택하고 삭제를 선택한 다음 삭제를 다시 선택하여 확인합니다.

Deadline Cloud에서 AI 에이전트 사용

AI 에이전트를 사용하여 Deadline Cloud에서 작업 번들을 작성하고, conda 패키지를 개발하고, 작업 문제를 해결합니다. 이 주제에서는 AI 에이전트가 무엇인지, 에이전트와 효과적으로 협력하기 위한 핵심 사항, 에이전트가 Deadline Cloud를 이해하는 데 도움이 되는 리소스에 대해 설명합니다.

AI 에이전트는 대규모 언어 모델(LLM)을 사용하여 작업을 자율적으로 수행하는 소프트웨어 도구입니다. AI 에이전트는 피드백을 기반으로 파일을 읽고 쓰고, 명령을 실행하고, 솔루션을 반복할 수 있습니다. 예를 들어 Kiro 및 IDE 통합 어시스턴트와 같은 명령줄 도구가 있습니다.

AI 에이전트 작업에 대한 요점

다음 핵심 사항은 Deadline Cloud에서 AI 에이전트를 사용할 때 더 나은 결과를 얻는 데 도움이 됩니다.

- 근거 제공 - AI 에이전트는 관련 설명서, 사양 및 예제에 액세스할 수 있을 때 최상의 성능을 발휘합니다. 에이전트가 특정 설명서 페이지를 가리키도록 하고, 기존 예제 코드를 참조로 공유하고, 관련 오픈 소스 리포지토리를 로컬 워크스페이스에 복제하고, 타사 애플리케이션에 대한 설명서를 제공하여 근거를 제공할 수 있습니다.
- 성공 기준 지정 - 에이전트에 대한 예상 결과 및 기술 요구 사항을 정의합니다. 예를 들어 에이전트에게 작업 번들을 개발하도록 요청할 때 작업 입력, 파라미터 및 예상 출력을 지정합니다. 사양에 대해 잘 모르는 경우 에이전트에게 먼저 옵션을 제안하도록 요청한 다음 요구 사항을 함께 구체화합니다.
- 피드백 루프 활성화 - AI 에이전트는 솔루션을 테스트하고 피드백을 받을 수 있을 때 더 효과적으로 반복합니다. 첫 번째 시도에서 작동 솔루션을 기대하는 대신 에이전트에게 솔루션을 실행하고 결과를 검토할 수 있는 권한을 부여합니다. 이 접근 방식은 에이전트가 상태 업데이트, 로그 및 검증 오류에 액세스할 수 있을 때 효과적입니다. 예를 들어 작업 번들을 개발할 때 에이전트가 작업을 제출하고 로그를 검토하도록 허용합니다.
- 반복 예상 - 컨텍스트가 양호하더라도 에이전트는 트랙에서 벗어나거나 환경과 일치하지 않는 가정을 할 수 있습니다. 에이전트가 작업에 접근하는 방식을 관찰하고 그 과정에서 지침을 제공합니다. 에이전트가 어려움을 겪는 경우 누락된 컨텍스트를 추가하고, 특정 로그 파일을 가리켜 오류를 찾고, 검색 시 요구 사항을 구체화하고, 에이전트가 피해야 할 사항을 명시적으로 설명하는 부정적인 요구 사항을 추가합니다.

에이전트 컨텍스트에 대한 리소스

다음 리소스는 AI 에이전트가 Deadline Cloud 개념을 이해하고 정확한 출력을 생성하는 데 도움이 됩니다.

- Deadline Cloud Model Context Protocol(MCP) 서버 - Model Context Protocol을 지원하는 에이전트의 경우, deadline-cloud 리포지토리에는 작업과 상호 작용하기 위한 MCP 서버가 포함된 Deadline Cloud 클라이언트가 포함되어 있습니다.
- AWS설명서 MCP 서버 - MCP를 지원하는 에이전트의 경우 에이전트에게 Deadline Cloud 사용 설명서 및 개발자 안내서를 포함한 AWS설명서에 대한 직접 액세스 권한을 부여하도록 AWS 설명서 MCP 서버를 구성합니다.
- Open Job Description 사양 - GitHub의 Open Job Description 사양은 작업 템플릿의 스키마를 정의합니다. 에이전트가 작업 템플릿의 구조와 구문을 이해해야 하는 경우이 리포지토리를 참조하세요.
- deadline-cloud-samples - deadline-cloud-samples 리포지토리에는 일반적인 애플리케이션 및 사용 사례를 위한 샘플 작업 번들, conda 레시피 및 CloudFormation템플릿이 포함되어 있습니다.
- aws-deadline GitHub 조직 - aws-deadline GitHub 조직에는 다른 통합의 예로 사용할 수 있는 많은 타사 애플리케이션에 대한 참조 플러그인이 포함되어 있습니다.

프롬프트 예제: 작업 번들 작성

다음 예제 프롬프트는 AI 에이전트를 사용하여 AI 이미지를 생성하기 위해 LoRA(낮은 순위 적응) 어댑터를 훈련하는 작업 번들을 생성하는 방법을 보여줍니다. 프롬프트는 앞에서 설명한 주요 요점을 보여줍니다. 즉, 관련 리포지토리를 가리켜 근거를 제공하고, 작업 번들 출력의 성공 기준을 정의하고, 반복 개발을 위한 피드백 루프를 간략하게 설명합니다.

```
Write a pair of job bundles for Deadline Cloud that use the diffusers Python library to
 train a LoRA adapter on a set of images and then generate images from it.

Requirements:
- The training job takes a set of JPEG images as input, uses an image description,
 LoRA rank, learning rate, batch size, and number of training steps as parameters, and
 outputs a `.safetensors` file.
- The generation job takes the `.safetensors` file as input and the number of images
 to generate, then outputs JPEG images. The jobs use Stable Diffusion 1.5 as the base
 model.
- The jobs run `diffusers` as a Python script. Install the necessary packages using
 conda by setting the job parameters:
    - `CondaChannels`: `conda-forge`
    - `CondaPackages`: list of conda packages to install

For context, clone the following repositories to your workspace and review their
 documentation and code:
- OpenJobDescription specification: https://github.com/OpenJobDescription/openjd-
specifications/blob/mainline/wiki/2023-09-Template-Schemas.md
```

```
- Deadline Cloud sample job bundles: https://github.com/aws-deadline/deadline-cloud-
samples/tree/mainline/job_bundles
- diffusers library: https://github.com/huggingface/diffusers

Read through the provided context before you start. To develop a job bundle, iterate
 with the following steps until the submitted job succeeds. If a step fails, update the
 job bundle and restart the loop:
1. Create a job bundle.
2. Validate the job template syntax: `openjd check`
3. Submit the job to Deadline Cloud: `deadline bundle submit`
4. Wait for the job to complete: `deadline job wait`
5. View the job status and logs: `deadline job logs`
6. Download the job output: `deadline job download-output`

To verify the training and generation jobs work together, iterate with the following
 steps until the generation job produces images that resemble the dog in the training
 data:
1. Develop and submit a training job using the training images in `./exdog`
2. Wait for the job to succeed then download its output.
3. Develop and submit a generation job using the LoRA adapter from the training job.
4. Wait for the job to succeed then download its output.
5. Inspect the generated images. If they resemble the dog in the training data, you're
 done. Otherwise, review the job template, job parameters, and job logs to identify and
 fix the issue.
```

AWSDeadline Cloud 모니터링

모니터링은 AWSDeadline Cloud(Deadline Cloud) 및 AWS솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. AWS솔루션의 모든 부분에서 모니터링 데이터를 수집하여 다중 지점 장애가 발생할 경우 보다 쉽게 디버깅할 수 있습니다. Deadline Cloud 모니터링을 시작하기 전에 다음 질문에 대한 답변을 포함하는 모니터링 계획을 생성해야 합니다.

- 모니터링의 목표
- 모니터링할 리소스
- 이러한 리소스를 모니터링하는 빈도
- 사용할 모니터링 도구
- 모니터링 작업을 수행할 사람
- 문제 발생 시 알려야 할 대상

AWS및 Deadline Cloud는 리소스를 모니터링하고 잠재적 인시던트에 대응하는 데 사용할 수 있는 도구를 제공합니다. 이러한 도구 중 일부는 모니터링을 자동으로 수행하며, 일부 도구는 수동 개입이 필요합니다. 모니터링 작업을 최대한 자동화해야 합니다.

- Amazon CloudWatch는 AWS리소스와 AWS에서 실행하는 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임곗값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch에서 Amazon EC2 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용자 안내서를 참조하세요.

  Deadline Cloud에는 세 가지 CloudWatch 지표가 있습니다.
- Amazon CloudWatch Logs로 Amazon EC2 인스턴스, CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임계값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구성이 뛰어난 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용자 안내서를 참조하세요.
- Amazon EventBridge를 사용하여 AWS서비스를 자동화하고 애플리케이션 가용성 문제 또는 리소스 변경과 같은 시스템 이벤트에 자동으로 대응할 수 있습니다. AWS서비스의 이벤트는 거의 실시간으로 EventBridge로 전달됩니다. 원하는 이벤트만 표시하도록 간단한 규칙을 작성한 후 규칙과 일치하는 이벤트 발생 시 실행할 자동화 작업을 지정할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용 설명서를 참조하세요.

- AWS CloudTrail는 AWS계정에 의해 또는 계정을 대신하여 수행된 API 호출 및 관련 이벤트를 캡처하고 사용자가 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 호출한 사용자 및 계정 AWS, 호출이 수행된 소스 IP 주소, 호출이 발생한 시기를 식별할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

주제

를 사용하여 Deadline Cloud API 호출 로깅 AWS CloudTrail

Deadline Cloud 는 사용자AWS CloudTrail, 역할 또는가 수행한 작업에 대한 레코드를 제공하는 서비스인와 통합됩니다 AWS 서비스. CloudTrail은에 대한 모든 API 호출을 이벤트 Deadline Cloud 로 캡처합니다. 캡처되는 호출에는 Deadline Cloud 콘솔의 호출과 Deadline Cloud API 작업에 대한 코드 호출이 포함됩니다. CloudTrail에서 수집한 정보를 사용하여 수행된 요청, 요청이 수행된 Deadline Cloud IP 주소, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. 자격 증명을 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트 사용자로 했는지 사용자 보안 인증으로 했는지 여부.
- IAM Identity Center 사용자를 대신하여 요청이 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자의 임시 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화되며 CloudTrail 이벤트 기록에 자동으로 액세스할 수 있습니다. CloudTrail 이벤트 기록은 지난 90일 간 AWS 리전의 관리 이벤트에 대해 보기, 검색 및 다운로드가 가능하고, 수정이 불가능한 레코드를 제공합니다. 자세한 설명은 AWS CloudTrail 사용 설명서의 CloudTrail 이벤트 기록 작업을 참조하세요. 이벤트 기록 보기는 CloudTrail 요금이 부과되지 않습니다.

AWS 계정 지난 90일 동안의 이벤트를 지속적으로 기록하려면 추적 또는 CloudTrail Lake 이벤트 데이터 스토어를 생성합니다.

CloudTrail 추적

CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 를 사용하여 생성된 모든 추적 AWS Management Console 은 다중 리전입니다. AWS CLI를 사용하여 단일 리전 또는 다중 리전 추적을 생성할 수 있습니다. 계정 AWS 리전 의 모든에서 활동을 캡처하므로 다중 리전 추적을 생성하는 것이 좋습니다. 단일 리전 추적을 생성하는 경우 추적의 AWS 리전에 로깅된 이벤트만 볼 수 있습니다. 추적에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS 계정에 대한 추적 생성 및 조직에 대한 추적 생성을 참조하세요.

CloudTrail에서 추적을 생성하여 진행 중인 관리 이벤트의 사본 하나를 Amazon S3 버킷으로 무료로 전송할 수는 있지만, Amazon S3 스토리지 요금이 부과됩니다. CloudTrail 요금에 관한 자세한 내용은 AWS CloudTrail 요금을 참조하세요. Amazon S3 요금에 관한 자세한 내용은 Amazon S3 요금을 참조하세요.

CloudTrail Lake 이벤트 데이터 스토어

CloudTrail Lake를 사용하면 이벤트에 대해 SQL 기반 쿼리를 실행할 수 있습니다. CloudTrail Lake는 행 기반 JSON 형식의 기존 이벤트를 Apache ORC 형식으로 변환합니다. ORC는 빠른 데이터 검색에 최적화된 열 기반 스토리지 형식입니다. 이벤트는 이벤트 데이터 스토어로 집계되며, 이벤트 데이터 스토어는 고급 이벤트 선택기를 적용하여 선택한 기준을 기반으로 하는 변경 불가능한 이벤트 컬렉션입니다. 이벤트 데이터 스토어에 적용하는 선택기는 어떤 이벤트가 지속되고 쿼리에 사용 가능한지를 제어합니다. CloudTrail Lake에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CloudTrail Lake 작업을 참조하세요.

CloudTrail Lake 이벤트 데이터 스토어 및 쿼리에는 비용이 발생합니다. 이벤트 데이터 스토어를 생성할 때 이벤트 데이터 스토어에 사용할 요금 옵션을 선택합니다. 요금 옵션에 따라 이벤트 모으기 및 저장 비용과 이벤트 데이터 스토어의 기본 및 최대 보존 기간이 결정됩니다. CloudTrail 요금에 관한 자세한 내용은 AWS CloudTrail 요금을 참조하세요.

Deadline Cloud CloudTrail의 데이터 이벤트

데이터 이벤트는 리소스 기반 또는 리소스에서 수행된 리소스 작업에 대한 정보를 제공합니다(예: Amazon S3 객체 읽기 또는 쓰기). 이를 데이터 플레인 작업이라고도 합니다. 데이터 이벤트는 흔히 대량 활동입니다. 기본적으로 CloudTrail은 데이터 이벤트를 로깅하지 않습니다. CloudTrail 이벤트 기록은 데이터 이벤트를 기록하지 않습니다.

데이터 이벤트에는 추가 요금이 적용됩니다. CloudTrail 요금에 관한 자세한 내용은 AWS CloudTrail 요금을 참조하세요.

CloudTrail 콘솔 AWS CLI또는 CloudTrail API 작업을 사용하여 Deadline Cloud 리소스 유형에 대한 데이터 이벤트를 로깅할 수 있습니다. 데이터 이벤트를 로깅하는 방법에 관한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS Management Console을 사용한 데이터 이벤트 로깅 및 AWS Command Line Interface를 사용한 이벤트 데이터 로깅을 참조하세요.

다음 표에는 데이터 이벤트를 로깅할 수 있는 Deadline Cloud 리소스 유형이 나열되어 있습니다. 데이터 이벤트 유형(콘솔) 열에는 CloudTrail 콘솔의 데이터 이벤트 유형 목록에서 선택할 값이 표시됩니다. resources.type 값 열에는 AWS CLI 또는 CloudTrail APIs를 사용하여 고급 이벤트 선택기를 구성할 때 지정하는 `resources.type` 값이 표시됩니다. CloudTrail에 로깅되는 데이터 API 열에는 리소스 유형에 대해 CloudTrail에 로깅된 API 직접 호출이 표시됩니다.

데이터 이벤트 유형(콘솔)	resources.type 값	CloudTrail에 로깅되는 데이터 API
기한 플릿	`AWS::Deadline::Fleet`	• SearchWorkers
기한 대기열	`AWS::Deadline::Fleet`	• SearchJobs
기한 작업자	`AWS::Deadline::Worker`	• GetWorker • ListSessionsForWorker • UpdateWorkerSchedule • BatchGetJobEntity • ListWorkers
기한 작업	`AWS::Deadline::Job`	• ListStepConsumers • UpdateTask • ListJobs • GetStep • ListSteps • GetJob • GetTask • GetSession • ListSessions • CreateJob • ListSessionActions

데이터 이벤트 유형(콘솔)	resources.type 값	CloudTrail에 로깅되는 데이터 API
		• ListTasks • CopyJobTemplate • UpdateSession • UpdateStep • UpdateJob • ListJobParameterDefinitions • GetSessionAction • ListStepDependencies • SearchTasks • SearchSteps

`eventName`, `readOnly` 및 `resources.ARN` 필드를 필터링하여 중요한 이벤트만 로깅하도록 고급 이벤트 선택기를 구성할 수 있습니다. 이러한 필드에 관한 자세한 내용은 AWS CloudTrail API 참조의 AdvancedFieldSelector 섹션을 참조하세요.

Deadline Cloud CloudTrail의 관리 이벤트

관리 이벤트는의 리소스에서 수행되는 관리 작업에 대한 정보를 제공합니다 AWS 계정. 이를 컨트롤 플레인 작업이라고도 합니다. 기본적으로 CloudTrail은 관리 이벤트를 로깅합니다.

AWS Deadline Cloud 는 다음 Deadline Cloud 컨트롤 플레인 작업을 관리 이벤트로 CloudTrail에 기록합니다.

- associate-member-to-farm
- associate-member-to-fleet
- associate-member-to-job
- associate-member-to-queue
- assume-fleet-role-for-read
- assume-fleet-role-for-worker
- assume-queue-role-for-read
- assume-queue-role-for-user

- assume-queue-role-for-worker
- 생성-예산
- create-farm
- create-fleet
- create-license-endpoint
- 생성-제한
- create-monitor
- create-queue
- create-queue-environment
- create-queue-fleet-association
- create-queue-limit-association
- create-storage-profile
- 작업자 생성
- 삭제-예산
- delete-farm
- delete-fleet
- delete-license-endpoint
- 삭제 제한
- delete-metered-product
- delete-monitor
- delete-queue
- delete-queue-environment
- delete-queue-fleet-association
- delete-queue-limit-association
- delete-storage-profile
- 작업자 삭제
- disassociate-member-from-farm
- disassociate-member-from-fleet
- disassociate-member-from-job
- disassociate-member-from-queue

- get-application-version
- get-budget
- get-farm
- get-feature-map
- get-fleet
- get-license-endpoint
- get-limit
- get-monitor
- get-queue
- get-queue-environment
- get-queue-fleet-association
- get-queue-limit-association
- get-sessions-statistics-aggregation
- get-storage-profile
- get-storage-profile-for-queue
- list-available-metered-products
- list-budgets
- list-farm-members
- list-farms
- list-fleet-members
- list-fleets
- list-job-members
- list-license-endpoints
- list-limit
- list-metered-products
- list-monitor
- list-queue-environments
- list-queue-fleet-associations
- list-queue-limit-associations
- list-queue-members

- list-queues
- list-storage-profiles
- list-storage-profiles-for-queue
- list-tags-for-resource
- put-metered-product
- start-sessions-statistics-aggregation
- tag-resource
- untag-resource
- update-budget
- update-farm
- update-플릿
- 업데이트 제한
- update-monitor
- update-queue
- update-queue-environment
- update-queue-fleet-association
- update-queue-limit-association
- update-storage-profile
- 작업자 업데이트

Deadline Cloud 이벤트 예제

이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청된 API 작업, 작업 날짜와 시간, 요청 파라미터 등에 관한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 추적이 아니므로 이벤트가 특정 순서로 표시되지 않습니다.

다음 예제는 `CreateFarm` 작업을 시연하는 CloudTrail 이벤트를 보여줍니다.

```
{
    "eventVersion": "0",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLE-PrincipalID:EXAMPLE-Session",
```

```
        "arn": "arn:aws:sts::111122223333:assumed-role/EXAMPLE-UserName/EXAMPLE-
Session",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE-accessKeyId",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLE-PrincipalID",
                "arn": "arn:aws:iam::111122223333:role/EXAMPLE-UserName",
                "accountId": "111122223333",
                "userName": "EXAMPLE-UserName"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-03-08T23:25:49Z"
            }
        }
    },
    "eventTime": "2021-03-08T23:25:49Z",
    "eventSource": "deadline.amazonaws.com",
    "eventName": "CreateFarm",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "EXAMPLE-userAgent",
    "requestParameters": {
        "displayName": "example-farm",
        "kmsKeyArn": "arn:aws:kms:us-west-2:111122223333:key/111122223333",
        "X-Amz-Client-Token": "12abc12a-1234-1abc-123a-1a11bc1111a",
        "description": "example-description",
        "tags": {
            "purpose_1": "e2e"
            "purpose_2": "tag_test"
        }
    },
    "responseElements": {
        "farmId": "EXAMPLE-farmID"
    },
    "requestID": "EXAMPLE-requestID",
    "eventID": "EXAMPLE-eventID",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333"
```

```
    "eventCategory": "Management",
}
```

JSON 예제는 이벤트를 식별하는 데 도움이 될 수 있는 AWS 리전 IP 주소 및 `requestParameters`"" 및 "`displayName`"와 같은 기타 "`kmsKeyArn`"를 보여줍니다.

CloudTrail 레코드 콘텐츠에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail record contents를 참조하세요.

CloudWatch를 사용하여 모니터링

Amazon CloudWatch(CloudWatch)는 원시 데이터를 수집하여 읽기 가능하며 실시간에 가까운 지표로 처리합니다. https://console.aws.amazon.com/cloudwatch/ CloudWatch 콘솔을 열어 Deadline Cloud 지표를 보고 필터링할 수 있습니다.

이러한 통계는 15개월 동안 보관되므로 기록 정보에 액세스하여 웹 애플리케이션 또는 서비스의 성능을 더 잘 파악할 수 있습니다. 특정 임계값을 주시하다가 해당 임계값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

Deadline Cloud에는 태스크 로그와 작업자 로그라는 두 가지 종류의 로그가 있습니다. 작업 로그는 실행 로그를 스크립트 또는 DCC 실행으로 실행하는 경우입니다. 작업 로그에는 자산 로드, 타일 렌더링 또는 찾을 수 없는 텍스처와 같은 이벤트가 표시될 수 있습니다.

작업자 로그에는 작업자 에이전트 프로세스가 표시됩니다. 여기에는 작업자 에이전트가 시작되거나, 자신을 등록하거나, 진행 상황을 보고하거나, 구성을 로드하거나, 작업을 완료하는 경우와 같은 상황이 포함될 수 있습니다.

이러한 로그의 네임스페이스는 입니다`/aws/deadline/*`.

Deadline Cloud의 경우 작업자는 이러한 로그를 CloudWatch Logs에 업로드합니다. 기본적으로 로그는 만료되지 않습니다. 작업이 대량의 데이터를 출력하는 경우 추가 비용이 발생할 수 있습니다. 자세한 내용은 Amazon CloudWatch 요금을 참조하세요.

각 로그 그룹에 대해 보존 정책을 조정할 수 있습니다. 보존 기간이 짧을수록 이전 로그가 제거되고 스토리지 비용을 절감하는 데 도움이 될 수 있습니다. 로그를 보관하려면 로그를 제거하기 전에 Amazon Simple Storage Service에 보관할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 콘솔을 사용하여 Amazon S3로 로그 데이터 내보내기를 참조하세요. Amazon CloudWatch

> **Note**
>
> CloudWatch 로그 읽기는에 의해 제한됩니다AWS. 많은 아티스트를 온보딩하려는 경우 AWS 고객 지원에 문의하여 CloudWatch의 할당`GetLogEvents`량 증가를 요청하는 것이 좋습니다. 또한 디버깅하지 않을 때는 로그 테일링 포털을 닫는 것이 좋습니다.

자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch Logs 할당량을 참조하세요. Amazon CloudWatch

CloudWatch 지표

Deadline Cloud는 Amazon CloudWatch로 지표를 전송합니다. AWS Management ConsoleAWS CLI, 또는 API를 사용하여 Deadline Cloud가 CloudWatch로 보내는 지표를 나열할 수 있습니다. 기본적으로 각 데이터 포인트는 활동 시작 시간 이후 1분을 포함합니다. AWS Management Console또는를 사용하여 사용 가능한 지표를 보는 방법에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 사용 가능한 지표 보기를 AWS CLI참조하세요.

고객 관리형 플릿 지표

`AWS/DeadlineCloud` 네임스페이스에는 고객 관리형 플릿에 대한 다음 지표가 포함되어 있습니다.

지표	설명	단위
`RecommendedFleetSize`	Deadline Cloud에서 작업을 처리하는 데 사용할 것을 권장하는 작업자 수입니다. 이 지표를 사용하여 플릿의 작업자 수를 확장하거나 축소할 수 있습니다.	개수
`UnhealthyWorkerCount`	정상이 아닌 작업을 처리하도록 할당된 작업자 수입니다.	개수

다음 차원을 사용하여 고객 관리형 플릿 지표를 구체화할 수 있습니다.

차원	설명
FarmId	이 차원은 지정된 팜에 요청하는 데이터를 필터링합니다.
FleetId	이 차원은 지정된 작업자 플릿에 요청하는 데이터를 필터링합니다.

라이선싱 지표

AWS/DeadlineCloud 네임스페이스에는 라이선스에 대한 다음 지표가 포함되어 있습니다.

지표	설명	단위
LicensesInUse	사용 중인 라이선스 세션 수입니다.	개수

다음 차원을 사용하여 라이선스 지표를 세분화할 수 있습니다.

차원	설명
FleetId	이 차원을 사용하여 데이터를 지정된 서비스 관리형 플릿으로 필터링합니다. 고객 관리형 플릿의 경우 LicenseEndpointId 차원을 대신 사용합니다.
LicenseEndpointId	이 차원을 사용하여 데이터를 지정된 라이선스 엔드포인트로 필터링합니다.
제품	이 차원을 사용하여 데이터를 지정된 측정 제품으로 필터링합니다.

리소스 제한 지표

AWS/DeadlineCloud 네임스페이스에는 리소스 제한에 대한 다음 지표가 포함되어 있습니다.

지표	설명	단위
CurrentCount	사용 중인이 제한을 기준으로 모델링된 리소스 수입니다.	개수
MaxCount	이 제한을 기준으로 모델링된 최대 리소스 수입니다. API를 사용하여 maxCount 값을 -1로 설정하면 Deadline Cloud는 MaxCount 지표를 내보내지 않습니다.	개수

다음 차원을 사용하여 동시 제한 지표를 구체화할 수 있습니다.

차원	설명
FarmId	이 차원은 지정된 팜에 요청하는 데이터를 필터링합니다.
LimitId	이 차원은 사용자가 요청한 데이터를 지정된 한도로 필터링합니다.

권장되는 경보

CloudWatch를 사용하면 지표를 감시하고 알림을 보내거나 임계값 위반 시 다른 작업을 수행하는 경보를 생성할 수 있습니다. CloudWatch 경보 구성에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

다음 Deadline Cloud 지표에 대한 경보를 설정하는 것이 좋습니다.

LicensesInUse

차원: FleetId, LicenseEndpointId

경보 설명: 이 경보는 서비스 관리형 플릿 또는 라이선스 엔드포인트의 활성 라이선스 세션이 계정 할당량에 근접하는 시점을 감지합니다. 이 경우 라이선스 세션에 대한 계정 할당량을 늘릴 수 있

습니다. Service Quotas를 사용하여 현재 할당량을 확인하고 증가를 요청합니다. 자세한 내용은 Service Quotas 사용 설명서를 참조하세요.

의도: 할당량 한도에 도달하기 전에 사용량을 모니터링하여 라이선스 체크아웃 실패를 방지합니다.

통계: Maximum

권장 임계값: 라이선스 세션 할당량의 90%

임곗값 정당화: 한도에 도달하기 전에 조치를 취할 수 있도록 임곗값을 할당량의 백분율로 설정합니다.

기간: 1분

경보를 보낼 데이터 포인트: 1

평가 기간: 1

비교 연산자: `GREATER_THAN_THRESHOLD`

추가 리소스

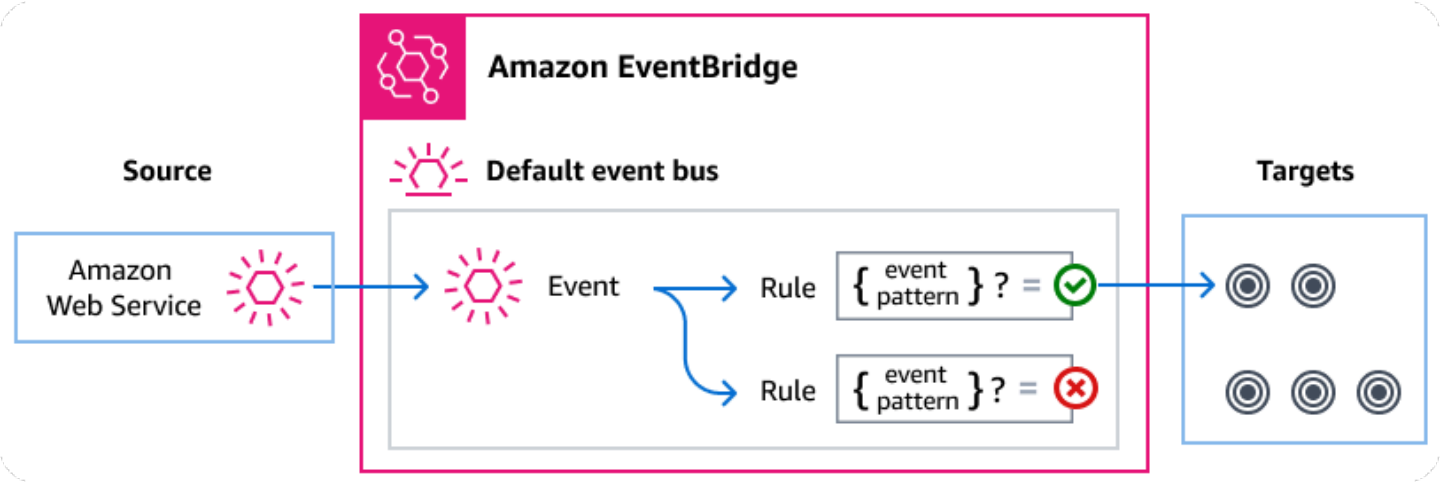
- Amazon CloudWatch 사용 설명서
- Service Quotas 사용 설명서

를 사용하여 Deadline Cloud 이벤트 관리 Amazon EventBridge

Amazon EventBridge 는 이벤트를 사용하여 애플리케이션 구성 요소를 함께 연결하는 서버리스 서비스이므로 확장 가능한 이벤트 기반 애플리케이션을 더 쉽게 구축할 수 있습니다. 이벤트 기반 아키텍처는 이벤트를 내보내고 이에 응답하여 함께 작동하는 느슨하게 결합된 소프트웨어 시스템을 구축하는 스타일입니다. 이벤트는 리소스나 환경의 변화를 나타냅니다.

작동 방식은 다음과 같습니다.

많은 AWS 서비스와 마찬가지로 Deadline Cloud는 이벤트를 생성하고 EventBridge 기본 이벤트 버스로 전송합니다. (기본 이벤트 버스는 모든 AWS 계정에 자동으로 프로비저닝됩니다.) 이벤트 버스는 이벤트를 수신하여 0개 이상의 목적지 또는 대상에 전달하는 라우터입니다. 이벤트 버스에 대해 지정한 규칙은 이벤트가 도착할 때 이벤트를 평가합니다. 각 규칙은 이벤트가 규칙의 이벤트 패턴과 일치하는지 여부를 확인합니다. 이벤트가 일치하면 이벤트 버스는 이벤트를 지정된 대상에게 전송합니다.



주제

- Deadline Cloud 이벤트
- EventBridge 규칙을 사용하여 Deadline Cloud 이벤트 전달
- Deadline Cloud 이벤트 세부 정보 참조

Deadline Cloud 이벤트

Deadline Cloud는 다음 이벤트를 기본 EventBridge 이벤트 버스로 자동으로 전송합니다. 규칙의 이벤트 패턴과 일치하는 이벤트는 최대한 지정된 대상으로 전달됩니다. 이벤트는 비순차적으로 전달될 수 있습니다.

자세한 내용을 알아보려면 Amazon EventBridge 사용 설명서의 EventBridge 이벤트를 참조하세요.

이벤트 세부 정보 유형	설명
예산 임계값 도달	대기열이 할당된 예산의 백분율에 도달하면 전송됩니다.
작업 수명 주기 상태 변경	작업의 수명 주기 상태가 변경될 때 전송됩니다.
작업 실행 상태 변경	작업 내 작업의 전체 상태가 변경될 때 전송됩니다.
단계 수명 주기 상태 변경	작업에서 단계의 수명 주기 상태가 변경될 때 전송됩니다.
단계 실행 상태 변경	단계에서 작업의 전체 상태가 변경될 때 전송됩니다.
작업 실행 상태 변경	작업 상태가 변경될 때 전송됩니다.

EventBridge 규칙을 사용하여 Deadline Cloud 이벤트 전달

EventBridge 기본 이벤트 버스가 Deadline Cloud 이벤트를 대상으로 보내도록 하려면 규칙을 생성해야 합니다. 각 규칙에는 이벤트 버스에서 수신된 각 이벤트와 EventBridge 일치하는 이벤트 패턴이 포함되어 있습니다. 이벤트 데이터가 지정된 이벤트 패턴과 일치하는 경우는 해당 이벤트를 규칙의 대상(들)에 EventBridge 전달합니다.

이벤트 버스 규칙 생성에 대해 자세히 알아보려면 EventBridge 사용 설명서의 이벤트에 대응하는 규칙 생성을 참조하세요.

Deadline Cloud 이벤트와 일치하는 이벤트 패턴 생성

각 이벤트 패턴은 다음을 포함하는 JSON 객체입니다.

- 이벤트를 전송하는 서비스를 식별하는 `source` 속성입니다. Deadline Cloud 이벤트의 경우 소스는 입니다`aws.deadline`.
- (선택 사항): 일치시킬 이벤트 유형의 배열을 포함하는 `detail-type` 속성입니다.
- (선택 사항): 일치시킬 다른 이벤트 데이터를 포함하는 `detail` 속성입니다.

예를 들어 다음 이벤트 패턴은 Deadline Cloud`farmId`에 지정된에 대한 모든 플릿 크기 권장 변경 이벤트와 일치합니다.

```
{
  "source": ["aws.deadline"],
  "detail-type": ["Fleet Size Recommendation Change"],
  "detail": {
     "farmId": "farm-12345678900000000000000000000000"
  }
}
```

자세한 내용은 EventBridge 사용 설명서의 이벤트 패턴을 참조하세요.

Deadline Cloud 이벤트 세부 정보 참조

AWS 서비스의 모든 이벤트에는 이벤트의 소스인 AWS 서비스, 이벤트가 생성된 시간, 이벤트가 발생한 계정 및 리전 등 이벤트에 대한 메타데이터가 포함된 공통 필드 세트가 있습니다. 이러한 일반 필드에 대한 정의는 Amazon EventBridge 사용 설명서의 이벤트 구조 참조를 참조하세요.

또한 각 이벤트에는 해당 특정 이벤트와 관련된 데이터를 포함하는 `detail` 필드가 있습니다. 아래 참조는 다양한 Deadline Cloud 이벤트에 대한 세부 정보 필드를 정의합니다.

EventBridge 를 사용하여 Deadline Cloud 이벤트를 선택하고 관리할 때는 다음 사항에 유의하는 것이 좋습니다.

- Deadline Cloud의 모든 이벤트에 대한 `source` 필드는 로 설정됩니다`aws.deadline`.
- `detail-type` 필드는 이벤트 유형을 지정합니다.

  예를 들어 `Fleet Size Recommendation Change`입니다.
- `detail` 필드는 해당 특정 이벤트와 관련된 데이터를 포함합니다.

Deadline Cloud 이벤트와 일치하도록 규칙을 활성화하는 이벤트 패턴을 구성하는 방법에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 이벤트 패턴을 참조하세요.

이벤트 및가 이벤트를 EventBridge 처리하는 방법에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon EventBridge 이벤트를 참조하세요.

주제

- 예산 임계값 도달 이벤트
- 플릿 크기 권장 사항 변경 이벤트
- 작업 수명 주기 상태 변경 이벤트
- 작업 실행 상태 변경 이벤트
- 단계 수명 주기 상태 변경 이벤트
- 단계 실행 상태 변경 이벤트
- 작업 실행 상태 변경 이벤트

예산 임계값 도달 이벤트

Budget Threshold Reached 이벤트를 사용하여 사용된 예산의 비율을 모니터링할 수 있습니다. Deadline Cloud는 사용된 백분율이 다음 임계값을 초과할 때 이벤트를 전송합니다.

- 10, 20, 30, 40, 50, 60, 70, 75, 80, 85, 90, 95, 96, 97, 98, 99, 100

Deadline Cloud가 예산 임계값 도달 이벤트를 전송하는 빈도는 예산이 한도에 가까워지면 증가합니다. 이 빈도를 사용하면 예산이 한도에 가까워지면 예산을 면밀히 모니터링하고 초과 지출을 방지하기 위한 조치를 취할 수 있습니다. 자체 예산 임계값을 설정할 수도 있습니다. Deadline Cloud는 사용량이 사용자 지정 임계값을 통과하면 이벤트를 전송합니다.

예산 금액을 변경하면 Deadline Cloud가 다음에 Budget Threshold Reached 이벤트를 보낼 때 사용된 예산의 현재 비율을 기반으로 합니다. 예를 들어 한도에 도달한 100 USD 예산에 50 USD를 추가하는 경우 다음 Budget Threshold Reached 이벤트는 예산이 75%임을 나타냅니다.

다음은 `Budget Threshold Reached` 이벤트의 세부 정보 필드입니다.

`source` 및 `detail-type` 필드는 Deadline Cloud 이벤트에 대한 특정 값을 포함하므로 아래에 포함되어 있습니다. 모든 이벤트에 포함된 다른 메타데이터 필드의 정의는 Amazon EventBridge 사용 설명서의 이벤트 구조 참조를 참조하세요.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Budget Threshold Reached",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "budgetId": "budget-12345678900000000000000000000000",
        "thresholdInPercent": 0
    }
}
```

`detail-type`

이벤트의 유형을 식별합니다.

이 이벤트의 경우 이 값은 `Budget Threshold Reached`입니다.

`source`

이벤트를 생성한 서비스를 식별합니다. Deadline Cloud 이벤트의 경우이 값은 입니다`aws.deadline`.

`detail`

이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이벤트를 생성하는 서비스에 따라 이 필드의 내용이 결정됩니다.

이 이벤트의 경우 이 데이터에는 다음이 포함됩니다.

`farmId`

작업이 포함된 팜의 식별자입니다.

`budgetId`

임계값에 도달한 예산의 식별자입니다.

`thresholdInPercent`

사용된 예산의 백분율입니다.

플릿 크기 권장 사항 변경 이벤트

이벤트 기반 Auto Scaling을 사용하도록 플릿을 구성하면 Deadline Cloud는 플릿을 관리하는 데 사용할 수 있는 이벤트를 전송합니다. 이러한 각 이벤트에는 플릿의 현재 크기와 요청된 크기에 대한 정보가 포함되어 있습니다. EventBridge 이벤트와 예제 Lambda 함수를 사용하여 이벤트를 처리하는 예제는 Deadline Cloud 규모 조정 권장 기능을 사용하여 Amazon EC2 플릿 자동 규모 조정을 참조하세요.

다음과 같은 경우 플릿 크기 권장 사항 변경 이벤트가 전송됩니다.

- 권장 플릿 크기가 변경되고 oldFleetSize가 newFleetSize와 다른 경우.
- 서비스가 실제 플릿 크기가 권장 플릿 크기와 일치하지 않음을 감지하는 경우. GetFleet 작업 응답에서 workerCount에서 실제 플릿 크기를 가져올 수 있습니다. 이는 활성 Amazon EC2 인스턴스가 Deadline Cloud 워커로 등록되지 않을 때 발생할 수 있습니다.

다음은 `Fleet Size Recommendation Change` 이벤트의 세부 정보 필드입니다.

`source` 및 `detail-type` 필드는 Deadline Cloud 이벤트에 대한 특정 값을 포함하므로 아래에 포함되어 있습니다. 모든 이벤트에 포함된 다른 메타데이터 필드의 정의는 Amazon EventBridge 사용 설명서의 이벤트 구조 참조를 참조하세요.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Fleet Size Recommendation Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
```

```
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "fleetId": "fleet-12345678900000000000000000000000",
        "oldFleetSize": 1,
        "newFleetSize": 5,
    }
}
```

`detail-type`

이벤트의 유형을 식별합니다.

이 이벤트의 경우 이 값은 `Fleet Size Recommendation Change`입니다.

`source`

이벤트를 생성한 서비스를 식별합니다. Deadline Cloud 이벤트의 경우이 값은 입니다`aws.deadline`.

`detail`

이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이벤트를 생성하는 서비스에 따라 이 필드의 내용이 결정됩니다.

이 이벤트의 경우 이 데이터에는 다음이 포함됩니다.

`farmId`

작업이 포함된 팜의 식별자입니다.

`fleetId`

크기 변경이 필요한 플릿의 식별자입니다.

`oldFleetSize`

플릿의 현재 크기입니다.

`newFleetSize`

플릿에 권장되는 새 크기입니다.

작업 수명 주기 상태 변경 이벤트

작업을 생성하거나 업데이트할 때 Deadline Cloud는 수명 주기 상태를 설정하여 가장 최근 사용자가 시작한 작업의 상태를 표시합니다.

작업이 생성되는 시점을 포함하여 수명 주기 상태 변경에 대해 작업 수명 주기 상태 변경 이벤트가 전송됩니다.

다음은 `Job Lifecycle Status Change` 이벤트의 세부 정보 필드입니다.

`source` 및 `detail-type` 필드는 Deadline Cloud 이벤트에 대한 특정 값을 포함하므로 아래에 포함되어 있습니다. 모든 이벤트에 포함된 다른 메타데이터 필드의 정의는 Amazon EventBridge 사용 설명서의 이벤트 구조 참조를 참조하세요.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Job Lifecycle Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
        "lifecycleStatus": "UPDATE_SUCCEEDED"
    }
}
```

`detail-type`

이벤트의 유형을 식별합니다.

이 이벤트의 경우 이 값은 `Job Lifecycle Status Change`입니다.

`source`

이벤트를 생성한 서비스를 식별합니다. Deadline Cloud 이벤트의 경우이 값은 입니다`aws.deadline`.

`detail`

이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이벤트를 생성하는 서비스에 따라 이 필드의 내용이 결정됩니다.

이 이벤트의 경우 이 데이터에는 다음이 포함됩니다.

`farmId`

작업이 포함된 팜의 식별자입니다.

`queueId`

작업이 포함된 대기열의 식별자입니다.

`jobId`

작업의 식별자입니다.

`previousLifecycleStatus`

작업이 떠나는 수명 주기 상태입니다. 이 필드는 작업을 처음 제출할 때 포함되지 않습니다.

`lifecycleStatus`

작업이 입력하는 수명 주기 상태입니다.

작업 실행 상태 변경 이벤트

작업은 여러 작업으로 구성됩니다. 각 작업에는 상태가 있습니다. 모든 작업의 상태가 결합되어 작업의 전체 상태를 제공합니다. 자세한 내용은 Deadline Cloud 사용 설명서의 Deadline Cloud의 작업 상태를 참조하세요. AWS

작업 실행 상태 변경 이벤트는 다음과 같은 경우에 전송됩니다.

- 결합된 `taskRunStatus` 필드가 변경됩니다.
- 작업이 준비 상태에 있지 않으면 작업이 다시 대기열에 추가됩니다.

다음과 같은 경우 작업 실행 상태 변경 이벤트가 전송되지 않습니다.

- 작업이 먼저 생성됩니다. 작업 생성을 모니터링하려면 작업 수명 주기 상태 변경 이벤트에서 변경 사항을 모니터링합니다.
- 작업의 `taskRunStatusCounts` 필드는 변경되지만 결합된 작업 실행 상태는 변경되지 않습니다.

다음은 `Job Run Status Change` 이벤트의 세부 정보 필드입니다.

`source` 및 `detail-type` 필드는 Deadline Cloud 이벤트에 대한 특정 값을 포함하므로 아래에 포함되어 있습니다. 모든 이벤트에 포함된 다른 메타데이터 필드의 정의는 Amazon EventBridge 사용 설명서의 이벤트 구조 참조를 참조하세요.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Job Run Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "previousTaskRunStatus": "RUNNING",
        "taskRunStatus": "SUCCEEDED",
        "taskRunStatusCounts": {
            "PENDING": 0,
            "READY": 0,
            "RUNNING": 0,
            "ASSIGNED": 0,
            "STARTING": 0,
            "SCHEDULED": 0,
            "INTERRUPTING": 0,
            "SUSPENDED": 0,
            "CANCELED": 0,
            "FAILED": 0,
            "SUCCEEDED": 20,
            "NOT_COMPATIBLE": 0
       }
    }
}
```

`detail-type`

이벤트의 유형을 식별합니다.

이 이벤트의 경우 이 값은 `Job Run Status Change`입니다.

`source`

이벤트를 생성한 서비스를 식별합니다. Deadline Cloud 이벤트의 경우이 값은 입니다`aws.deadline`.

`detail`

이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이벤트를 생성하는 서비스에 따라 이 필드의 내용이 결정됩니다.

이 이벤트의 경우 이 데이터에는 다음이 포함됩니다.

`farmId`

작업이 포함된 팜의 식별자입니다.

`queueId`

작업이 포함된 대기열의 식별자입니다.

`jobId`

작업의 식별자입니다.

`previousTaskRunStatus`

작업이 나가고 있는 작업 실행 상태입니다.

`taskRunStatus`

작업이 입력하는 작업 실행 상태입니다.

`taskRunStatusCounts`

각 상태의 작업 수입니다.

단계 수명 주기 상태 변경 이벤트

이벤트를 생성하거나 업데이트할 때 Deadline Cloud는 작업의 수명 주기 상태를 설정하여 가장 최근 사용자가 시작한 작업의 상태를 설명합니다.

다음과 같은 경우 단계 수명 주기 상태 변경 이벤트가 전송됩니다.

- 단계 업데이트가 시작됩니다(UPDATE_IN_PROGRESS).
- 단계 업데이트가 성공적으로 완료되었습니다(UPDATE_SUCCEEDED).
- 단계 업데이트 실패(UPDATE_FAILED).

단계가 처음 생성되면 이벤트가 전송되지 않습니다. 단계 생성을 모니터링하려면 작업 수명 주기 상태 변경 이벤트에서 변경 사항을 모니터링합니다.

다음은 Step Lifecycle Status Change 이벤트의 세부 정보 필드입니다.

source 및 detail-type 필드는 Deadline Cloud 이벤트에 대한 특정 값을 포함하므로 아래에 포함되어 있습니다. 모든 이벤트에 포함된 다른 메타데이터 필드의 정의는 Amazon EventBridge 사용 설명서의 이벤트 구조 참조를 참조하세요.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Step Lifecycle Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "previousLifecycleStatus": "UPDATE_IN_PROGRESS",
        "lifecycleStatus": "UPDATE_SUCCEEDED"
    }
}
```

detail-type

이벤트의 유형을 식별합니다.

이 이벤트의 경우 이 값은 Step Lifecycle Status Change입니다.

source

이벤트를 생성한 서비스를 식별합니다. Deadline Cloud 이벤트의 경우이 값은 입니다aws.deadline.

detail

이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이벤트를 생성하는 서비스에 따라 이 필드의 내용이 결정됩니다.

이 이벤트의 경우 이 데이터에는 다음이 포함됩니다.

`farmId`

작업이 포함된 팜의 식별자입니다.

`queueId`

작업이 포함된 대기열의 식별자입니다.

`jobId`

작업의 식별자입니다.

`stepId`

현재 작업 단계의 식별자입니다.

`previousLifecycleStatus`

단계가 나가고 있는 수명 주기 상태입니다.

`lifecycleStatus`

단계가 입력하는 수명 주기 상태입니다.

단계 실행 상태 변경 이벤트

작업의 각 단계는 여러 작업으로 구성됩니다. 각 작업에는 상태가 있습니다. 작업 상태가 결합되어 단계 및 작업의 전체 상태를 제공합니다.

단계 실행 상태 변경 이벤트는 다음과 같은 경우에 전송됩니다.

- 결합된가 `taskRunStatus` 변경됩니다.
- 단계가 준비 상태에 있지 않으면 단계가 다시 대기열에 추가됩니다.

다음과 같은 경우에는 이벤트가 전송되지 않습니다.

- 단계가 먼저 생성됩니다. 단계 생성을 모니터링하려면 작업 수명 주기 상태 변경 이벤트에서 변경 사항을 모니터링합니다.
- 단계의는 `taskRunStatusCounts` 변경되지만 결합된 단계 작업 실행 상태는 변경되지 않습니다.

다음은 `Step Run Status Change` 이벤트의 세부 정보 필드입니다.

source 및 detail-type 필드는 Deadline Cloud 이벤트에 대한 특정 값을 포함하므로 아래에 포함되어 있습니다. 모든 이벤트에 포함된 다른 메타데이터 필드의 정의는 Amazon EventBridge 사용 설명서의 이벤트 구조 참조를 참조하세요.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Step Run Status Change",
    "source": "aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "previousTaskRunStatus": "RUNNING",
        "taskRunStatus": "SUCCEEDED",
        "taskRunStatusCounts": {
            "PENDING": 0,
            "READY": 0,
            "RUNNING": 0,
            "ASSIGNED": 0,
            "STARTING": 0,
            "SCHEDULED": 0,
            "INTERRUPTING": 0,
            "SUSPENDED": 0,
            "CANCELED": 0,
            "FAILED": 0,
            "SUCCEEDED": 20,
            "NOT_COMPATIBLE": 0
       }
    }
}
```

`detail-type`

이벤트의 유형을 식별합니다.

이 이벤트의 경우 이 값은 `Step Run Status Change`입니다.

`source`

이벤트를 생성한 서비스를 식별합니다. Deadline Cloud 이벤트의 경우이 값은 입니다`aws.deadline`.

`detail`

이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이벤트를 생성하는 서비스에 따라 이 필드의 내용이 결정됩니다.

이 이벤트의 경우 이 데이터에는 다음이 포함됩니다.

`farmId`

작업이 포함된 팜의 식별자입니다.

`queueId`

작업이 포함된 대기열의 식별자입니다.

`jobId`

작업의 식별자입니다.

`stepId`

현재 작업 단계의 식별자입니다.

`previousTaskRunStatus`

단계가 나가고 있는 실행 상태입니다.

`taskRunStatus`

단계가 입력하는 실행 상태입니다.

`taskRunStatusCounts`

각 상태의 단계 작업 수입니다.

작업 실행 상태 변경 이벤트

`runStatus` 필드는 작업이 실행될 때 업데이트됩니다. 이벤트는 다음과 같은 경우에 전송됩니다.

- 작업의 실행 상태가 변경됩니다.
- 작업이 준비 상태에 있지 않으면 작업이 다시 대기열에 추가됩니다.

다음과 같은 경우에는 이벤트가 전송되지 않습니다.

- 작업이 먼저 생성됩니다. 작업 생성을 모니터링하려면 작업 수명 주기 상태 변경 이벤트에서 변경 사항을 모니터링합니다.

다음은 `Task Run Status Change` 이벤트의 세부 정보 필드입니다.

`source` 및 `detail-type` 필드는 Deadline Cloud 이벤트에 대한 특정 값을 포함하므로 아래에 포함되어 있습니다. 모든 이벤트에 포함된 다른 메타데이터 필드의 정의는 Amazon EventBridge 사용 설명서의 이벤트 구조 참조를 참조하세요.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "Task Run Status Change",
    "source": "aws.aws.deadline",
    "account": "111122223333",
    "time": "2017-12-22T18:43:48Z",
    "region": "aa-example-1",
    "resources": [],
    "detail": {
        "farmId": "farm-12345678900000000000000000000000",
        "queueId": "queue-12345678900000000000000000000000",
        "jobId": "job-12345678900000000000000000000000",
        "stepId": "step-12345678900000000000000000000000",
        "taskId": "task-12345678900000000000000000000000-0",
        "previousRunStatus": "RUNNING",
        "runStatus": "SUCCEEDED"
    }
}
```

`detail-type`

이벤트의 유형을 식별합니다.

이 이벤트의 경우 이 값은 `Fleet Size Recommendation Change`입니다.

`source`

이벤트를 생성한 서비스를 식별합니다. Deadline Cloud 이벤트의 경우이 값은 입니다`aws.deadline`.

`detail`

이벤트에 대한 정보를 포함하는 JSON 객체입니다. 이벤트를 생성하는 서비스에 따라 이 필드의 내용이 결정됩니다.

이 이벤트의 경우 이 데이터에는 다음이 포함됩니다.

`farmId`

작업이 포함된 팜의 식별자입니다.

`queueId`

작업이 포함된 대기열의 식별자입니다.

`jobId`

작업의 식별자입니다.

stepId

현재 작업 단계의 식별자입니다.

`taskId`

실행 중인 작업의 식별자입니다.

`previousRunStatus`

작업이 나가고 있는 실행 상태입니다.

`runStatus`

작업이 입력하는 실행 상태입니다.

를 사용하여 세션 통계 집계 데이터 쿼리 AWS CLI

비용을 추적하거나, 리소스 사용량을 분석하거나, 리소스를 가장 많이 사용하는 사용자를 식별하려면 AWS Command Line Interface (AWS CLI)를 사용하여 AWS Deadline Cloud(Deadline Cloud) 팜에 대한 집계된 세션 통계를 쿼리할 수 있습니다. 세션 통계 API는 대기열, 플릿, 인스턴스 유형 또는 사용자와 같은 다양한 차원별로 그룹화할 수 있는 비용, 런타임 및 사용량에 대한 데이터를 제공합니다.

세션 통계 쿼리는 비동기 프로세스입니다. 먼저 집계 요청을 시작한 다음 집계 ID를 사용하여 결과를 검색합니다.

집계 요청 시작

집계 요청을 시작하려면 `start-sessions-statistics-aggregation` 명령을 실행합니다. 다음 예시에서는 특정 대기열의 사용자 ID별로 통계를 그룹화합니다. *## ###* 텍스트를 정보로 바꿉니다.

```
aws deadline start-sessions-statistics-aggregation \
    --farm-id farm-id \
    --resource-ids '{"queueIds":["queue-id"]}' \
    --start-time 2025-11-24T10:00:00Z \
    --end-time 2025-11-25T18:00:00Z \
    --group-by '["USER_ID"]' \
    --period HOURLY \
    --statistics '["SUM"]' \
    --timezone UTC-08:00 \
    --region region-name
```

`QUEUE_ID`, , `FLEET_ID`, `JOB_ID` `INSTANCE_TYPE`또는와 같은 다른 차원별로 통계를 그룹화할 수 있습니다`LICENSE_PRODUCT`. 사용 가능한 모든 파라미터에 대한 자세한 내용은 AWS CLI 명령 참조의 start-sessions-statistics-aggregation을 참조하세요.

응답에는 집계 ID가 포함됩니다.

```
{
    "aggregationId": "92b35143f2d04641979bc9b777232f38"
}
```

결과 검색

집계 ID로 `get-sessions-statistics-aggregation` 명령을 실행하여 결과를 검색합니다. `## # ##` 텍스트를 정보로 바꿉니다.

```
aws deadline get-sessions-statistics-aggregation \
    --farm-id farm-id \
    --aggregation-id aggregation-id \
    --region region-name
```

다음 예제에서는 사용자 ID별로 통계를 그룹화할 때의 응답을 보여줍니다. `userId` 필드에는 사용자를 식별하기 위해 사용자 이름에 매핑해야 하는 UUID가 포함되어 있습니다.

```
{
    "statistics": [
        {
            "userId": "f9c1f3f0-1031-70dc-4d25-30d7225b04a0",
            "count": 1,
            "costInUsd": {
                "sum": 0.0
            },
            "runtimeInSeconds": {
                "sum": 53.773
            },
            "aggregationStartTime": "2025-11-24T22:00:00Z",
            "aggregationEndTime": "2025-11-24T23:00:00Z"
        }
    ],
    "status": "COMPLETED"
}
```

와 연결된 사용자 이름을 찾으려면 섹션을 `userId`참조하세요the section called “userID를 사용하여 사용자 메타데이터 검색”.

API에 대한 자세한 내용은 Deadline Cloud API 참조를 참조하세요.

주제

- the section called “userID를 사용하여 사용자 메타데이터 검색”

ID 스토어에서 userID를 사용하여 사용자 메타데이터 및 속성 검색

> **Note**
>
> 이 절차는 동일한 사용자 ID 형식을 사용하는 SearchJobs API에서 반환되는 `createdBy` 필드에도 적용됩니다.

세션 통계의 `userId` 필드에는 다음 값 중 하나가 포함됩니다.

- AWS Identity and Access Management (IAM) 역할 ARN, 예: `arn:aws:sts::123456789012:assumed-role/Admin/user-Isengard`.
- IAM Identity Center 사용자 ID(UUID), 예: `f9c1f3f0-1031-70dc-4d25-30d7225b04a0`.

IAM 역할 ARNs의 경우 사용자 이름은 ARN 자체에 표시됩니다. IAM Identity Center 사용자 IDs의 경우 IAM Identity Center Identity Store API를 사용하여 사용자 이름을 조회할 수 있습니다.

IAM Identity Center 사용자 ID와 연결된 사용자 이름을 식별하려면 다음 절차를 사용합니다. 시작하기 전에 IAM Identity Center 설정에서 Identity Store ID를 가져옵니다. 자세한 내용은 the section called "ID 스토어 ID 찾기" 단원을 참조하십시오.

사용자 ID를 매핑하려면

1. 다음 명령을 실행하여 *IdentityStoreId*를 Identity Store ID로 바꾸고 *userUUID*를 세션 통계 응답`userId`의 로 바꿉니다.

```
aws identitystore describe-user \
    --identity-store-id IdentityStoreId \
    --user-id userUUID
```

2. 사용자 이름이 포함된 응답을 검토합니다.

```
{
    "UserName": "jdoe",
    "UserId": "f9c1f3f0-1031-70dc-4d25-30d7225b04a0",
    "Name": {
        "FamilyName": "Doe",
        "GivenName": "Jane"
    },
```

```
    "DisplayName": "Jane Doe",
    "Emails": [{
        "Value": "jdoe@example.com",
        "Type": "work",
        "Primary": true
    }],
    "IdentityStoreId": "d-xxxxxxxxxx"
}
```

ID 스토어 ID 찾기

사용자 IDs 사용자 이름에 매핑하려면 Identity Store ID가 필요합니다. IAM Identity Center 콘솔 또는를 사용하여 Identity Store ID를 찾을 수 있습니다 AWS CLI.

콘솔

콘솔을 사용하여 Identity Store ID를 찾으려면 다음 절차를 사용합니다.

1. AWS Management Console에 로그인하고 IAM Identity Center 콘솔을 엽니다.
2. 탐색 창에서 설정을 선택합니다.
3. IAM Identity Center Identity Store ID 값을 복사합니다. 형식은 d-xxxxxxxxxx입니다.

AWS CLI

다음 명령을 실행하여 *region-name*을 IAM Identity Center 인스턴스가 구성된 리전으로 바꿉니다.

```
aws sso-admin list-instances --region region-name
```

응답에는 IdentityStoreId가 포함됩니다.

```
{
    "Instances": [
        {
            "CreatedDate": "2025-11-19T15:45:55.160000-08:00",
            "IdentityStoreId": "d-xxxxxxxxxx",
            "InstanceArn": "arn:aws:sso:::instance/ssoins-xxxxxxxxxxxxxxxx",
            "OwnerAccountId": "123456789012",
            "Status": "ACTIVE"
        }
```

```
    ]
}
```

사용자 매핑 확인

사용자 ID를 사용자 이름에 매핑한 후 IAM Identity Center 콘솔에서 사용자 ID가 예상 사용자와 일치하는지 확인할 수 있습니다. 사용자 매핑을 확인하려면 다음 절차를 사용합니다.

1. AWS Management Console에 로그인하고 IAM Identity Center 콘솔을 엽니다.
2. 탐색 창에서 사용자를 선택합니다.
3. AWS CLI 응답에서 사용자 이름을 선택합니다.
4. 일반 정보 섹션에서 사용자 ID가 세션 통계의와 일치하는`userId`지 확인합니다.

추가 리소스

- IAM Identity Center 사용 설명서
- IAM Identity Center Identity Store API 참조

의 보안 Deadline Cloud

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 서비스 에서 실행되는 인프라를 보호할 책임이 있습니다 AWS 클라우드. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 AWS 서비스 프로그램 제공 범위 내 규정 준수 AWS Deadline Cloud참조하세요.
- 클라우드의 보안 - 사용자의 책임은 AWS 서비스 사용하는에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다 Deadline Cloud. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 Deadline Cloud 를 구성하는 방법을 보여줍니다. 또한 Deadline Cloud 리소스를 모니터링하고 보호하는 데 도움이 AWS 서비스 되는 다른를 사용하는 방법을 알아봅니다.

주제

의 데이터 보호 Deadline Cloud

AWS 공동 책임 모델 은 AWS Deadline Cloud의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 일반 데이터 보호 규정 (GDPR) 센터를 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 Deadline Cloud 또는 기타 AWS 서비스 에서 콘솔, API AWS CLI또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

Deadline Cloud 작업 템플릿의 이름 필드에 입력된 데이터는 결제 또는 진단 로그에도 포함될 수 있으며 기밀 또는 민감한 정보를 포함해서는 안 됩니다.

주제

- 저장 시 암호화

저장 시 암호화

AWS Deadline Cloud 는 AWS Key Management Service (AWS KMS)에 저장된 암호화 키를 사용하여 저장 데이터를 암호화하여 민감한 데이터를 보호합니다. 유휴 시 암호화는를 사용할 수 AWS 리전 있는 모든에서 사용할 수 Deadline Cloud 있습니다.

데이터를 암호화하면 유효한 키가 없는 사용자 또는 애플리케이션에서 디스크에 저장된 민감한 데이터를 읽을 수 없습니다. 유효한 관리형 키가 있는 당사자만 데이터를 해독할 수 있습니다.

Deadline Cloud 는 서비스 관리형 플릿 워커 인스턴스가 종료될 때 Amazon Elastic Block Store 볼륨을 삭제합니다.

가 저장 데이터 암호화를 AWS KMS 위해를 Deadline Cloud 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요키 관리.

전송 중 암호화

전송 중인 데이터의 경우 TLS(전송 계층 보안) 1.2 또는 1.3을 AWS Deadline Cloud 사용하여 서비스와 작업자 간에 전송된 데이터를 암호화합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다. 또한 Virtual Private Cloud(VPC)를 사용하는 경우 AWS PrivateLink 를 사용하여 VPC와 간에 프라이빗 연결을 설정할 수 있습니다 Deadline Cloud.

키 관리

새 팜을 생성할 때 다음 키 중 하나를 선택하여 팜 데이터를 암호화할 수 있습니다.

- AWS 소유 KMS 키 - 팜을 생성할 때 키를 지정하지 않으면 기본 암호화 유형입니다. KMS 키는에서 소유합니다 AWS Deadline Cloud. AWS 소유 키를 보거나 관리하거나 사용할 수 없습니다. 그러나 데이터를 암호화하는 키를 보호하기 위해 조치를 취할 필요는 없습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS 소유 키를 참조하세요.
- 고객 관리형 KMS 키 - 팜을 생성할 때 고객 관리형 키를 지정합니다. 팜 내의 모든 콘텐츠는 KMS 키로 암호화됩니다. 키는 계정에 저장되며 사용자가 생성, 소유 및 관리하며 AWS KMS 요금이 적용됩니다. KMS 키를 완전히 제어할 수 있습니다. 다음과 같은 작업을 수행할 수 있습니다.

- 키 정책 수립 및 유지 관리
- IAM 정책 및 권한 부여 수립 및 유지
- 키 정책 활성화 및 비활성화
- 태그 추가
- 키 별칭 만들기

Deadline Cloud 팜에 사용되는 고객 소유 키는 수동으로 교체할 수 없습니다. 키의 자동 교체가 지원됩니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 소유 키를 참조하세요.

고객 관리형 키를 생성하려면 AWS Key Management Service 개발자 안내서의 대칭 고객 관리형 키 생성 단계를 따르세요.

에서 AWS KMS 권한 부여를 Deadline Cloud 사용하는 방법

Deadline Cloud 고객 관리형 키를 사용하려면에 권한 부여가 필요합니다. 고객 관리형 키로 암호화된 팜을 생성하면는 지정한 KMS 키에 대한 액세스 권한을 `CreateGrant` AWS KMS 에 요청하여 사용자를 대신하여 권한 부여를 Deadline Cloud 생성합니다.

Deadline Cloud 는 여러 권한 부여를 사용합니다. 각 권한 부여는 데이터를 암호화하거나 해독해야 Deadline Cloud 하는의 여러 부분에서 사용됩니다. Deadline Cloud 또한 권한 부여를 사용하여 Amazon Simple Storage Service, Amazon Elastic Block Store 또는 OpenSearch와 같이 사용자를 대신하여 데이터를 저장하는 데 사용되는 다른 AWS 서비스에 대한 액세스를 허용합니다.

가 서비스 관리형 플릿에서 시스템을 관리할 Deadline Cloud 수 있는 권한 부여에는 서비스 보안 주체 `GranteePrincipal` 대신의 계정 번호와 역할이 포함됩니다 Deadline Cloud . 일반적으로는 아니지만 팜에 지정된 고객 관리형 KMS 키를 사용하여 서비스 관리형 플릿의 작업자에 대한 Amazon EBS 볼륨을 암호화하는 데 필요합니다.

고객 관리형 키 정책

키 정책에서는 고객 관리형 키에 대한 액세스를 제어합니다. 각 키에는 키를 사용할 수 있는 사용자와 키 사용 방법을 결정하는 문이 포함된 정확히 하나의 키 정책이 있어야 합니다. 고객 관리형 키를 생성할 때 키 정책을 지정할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 관리형 키에 대한 액세스 관리를 참조하세요.

CreateFarm에 대한 최소 IAM 정책

고객 관리형 키를 사용하여 콘솔 또는 `CreateFarm` API 작업을 사용하여 팜을 생성하려면 다음 AWS KMS API 작업을 허용해야 합니다.

- `kms:CreateGrant` - 고객 관리형 키에 권한 부여를 추가합니다. 콘솔에 지정된 AWS KMS 키에 대한 액세스 권한을 부여합니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 권한 부여 사용을 참조하세요.
- `kms:Decrypt` -가 팜의 데이터를 복호화 Deadline Cloud 할 수 있습니다.
- `kms:DescribeKey` -가 키를 Deadline Cloud 검증할 수 있도록 고객 관리형 키 세부 정보를 제공합니다.
- `kms:GenerateDataKey` -가 고유한 데이터 키를 사용하여 데이터를 암호화 Deadline Cloud 하도록 허용합니다.

다음 정책 설명은 `CreateFarm` 작업에 필요한 권한을 부여합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineCreateGrants",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:CreateGrant",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234567890abcdef0",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
```

```
}
```

읽기 전용 작업에 대한 최소 IAM 정책

팜, 대기열 및 플릿에 대한 정보 가져오기와 같은 읽기 전용 Deadline Cloud 작업에 고객 관리형 키를 사용합니다. 다음 AWS KMS API 작업을 허용해야 합니다.

- kms:Decrypt -가 팜의 데이터를 복호화 Deadline Cloud 할 수 있습니다.
- kms:DescribeKey -가 키를 Deadline Cloud 검증할 수 있도록 고객 관리형 키 세부 정보를 제공합니다.

다음 정책 설명은 읽기 전용 작업에 필요한 권한을 부여합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadOnly",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

읽기-쓰기 작업에 대한 최소 IAM 정책

팜, 대기열 및 플릿 생성 및 업데이트와 같은 읽기-쓰기 Deadline Cloud 작업에 고객 관리형 키를 사용합니다. 다음 AWS KMS API 작업을 허용해야 합니다.

- kms:Decrypt -가 팜의 데이터를 복호화 Deadline Cloud 할 수 있습니다.
- kms:DescribeKey -가 키를 Deadline Cloud 검증할 수 있도록 고객 관리형 키 세부 정보를 제공합니다.
- kms:GenerateDataKey -가 고유한 데이터 키를 사용하여 데이터를 암호화 Deadline Cloud 하도록 허용합니다.

다음 정책 설명은 CreateFarm 작업에 필요한 권한을 부여합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DeadlineReadWrite",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "deadline.us-west-2.amazonaws.com"
                }
            }
        }
    ]
}
```

암호화 키 모니터링

Deadline Cloud 팜에 AWS KMS 고객 관리형 키를 사용하는 경우 AWS CloudTrail 또는 Amazon CloudWatch Logs를 사용하여가 로 Deadline Cloud 보내는 요청을 추적할 수 있습니다 AWS KMS.

권한 부여에 대한 CloudTrail 이벤트

다음 예제 CloudTrail 이벤트는 권한 부여가 생성될 때, 일반적으로 `CreateFarm`, `CreateMonitor`또는 `CreateFleet` 작업을 호출할 때 발생합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/Admin/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T02:05:26Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T02:05:35Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "operations": [
            "CreateGrant",
```

```
            "Decrypt",
            "DescribeKey",
            "Encrypt",
            "GenerateDataKey"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:deadline:farmId": "farm-abcdef12345678900987654321fedcba",
                "aws:deadline:accountId": "111122223333"
            }
        },
        "granteePrincipal": "deadline.amazonaws.com",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
        "retiringPrincipal": "deadline.amazonaws.com"
    },
    "responseElements": {
        "grantId": "6bbe819394822a400fe5e3a75d0e9ef16c1733143fff0c1fc00dc7ac282a18a0",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "readOnly": false,
    "resources": [
        {
            "accountId": "AWS Internal",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE44444"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

복호화를 위한 CloudTrail 이벤트

다음 예제 CloudTrail 이벤트는 고객 관리형 KMS 키를 사용하여 값을 해독할 때 발생합니다.

```
{
    "eventVersion": "1.08",
```

```
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/SampleRole",
                "accountId": "111122223333",
                "userName": "SampleRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T18:46:51Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:51:44Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:deadline:farmId": "farm-abcdef1234567890987654321fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
    },
    "responseElements": null,
    "requestID": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeffffff",
    "eventID": "ffffffff-eeee-dddd-cccc-bbbbbbaaaaaa",
    "readOnly": true,
    "resources": [
```

```
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

암호화를 위한 CloudTrail 이벤트

다음 예제 CloudTrail 이벤트는 고객 관리형 KMS 키를 사용하여 값을 암호화할 때 발생합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:SampleUser01",
        "arn": "arn:aws::sts::111122223333:assumed-role/SampleRole/SampleUser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE",
                "arn": "arn:aws::iam::111122223333:role/SampleRole",
                "accountId": "111122223333",
                "userName": "SampleRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-04-23T18:46:51Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "deadline.amazonaws.com"
    },
    "eventTime": "2024-04-23T18:52:40Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
```

```
    "awsRegion": "us-west-2",
    "sourceIPAddress": "deadline.amazonaws.com",
    "userAgent": "deadline.amazonaws.com",
    "requestParameters": {
        "numberOfBytes": 32,
        "encryptionContext": {
            "aws:deadline:farmId": "farm-abcdef1234567890098765432 1fedcba",
            "aws:deadline:accountId": "111122223333",
            "aws-crypto-public-key": "AotL+SAMPLEVALUEiOMEXAMPLEaaqNOTREALaGTESTONLY
+p/5H+EuKd4Q=="
        },
        "keyId": "arn:aws::kms:us-
west-2:111122223333:key/abcdef12-3456-7890-0987-654321fedcba"
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws::kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLE33333"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


고객 관리형 KMS 키 삭제

AWS Key Management Service (AWS KMS)에서 고객 관리형 KMS 키를 삭제하면 파괴적이고 위험할 수 있습니다. 이렇게 하면 키와 연결된 키 구성 요소와 모든 메타데이터가 되돌릴 수 없는 방식으로 삭제됩니다. 고객 관리형 KMS 키가 삭제된 후에는 해당 키로 암호화된 데이터를 더 이상 복호화할 수 없습니다. 키를 삭제하면 데이터를 복구할 수 없게 됩니다.

따라서는 KMS 키를 삭제하기 전에 고객에게 최대 30일의 대기 기간을 AWS KMS 제공합니다. 기본 대기 기간은 30일입니다.

대기 기간에 대해

고객 관리형 KMS 키를 삭제하는 것은 파괴적이고 잠재적으로 위험하기 때문에 대기 기간을 7~30일로 설정해야 합니다. 기본 대기 기간은 30일입니다.

그러나 실제 대기 기간은 예약한 기간보다 최대 24시간 더 길 수 있습니다. 키가 삭제될 실제 날짜와 시간을 가져오려면 DescribeKey 작업을 사용합니다. AWS KMS 콘솔의 키 세부 정보 페이지에 있는 일반 구성 섹션에서 예약된 삭제 날짜를 확인할 수도 있습니다. 시간대를 확인하세요.

대기 기간 동안 고객 관리형 키의 상태 및 키 상태는 삭제 대기 중입니다.

- 삭제 대기 중인 고객 관리형 KMS 키는 어떠한 암호화 작업에도 사용할 수 없습니다.
- AWS KMS 는 삭제 보류 중인 고객 관리형 KMS 키의 백업 키를 교체하지 않습니다.

고객 관리형 KMS 키 삭제에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 마스터 키 삭제를 참조하세요.

인터네트워크 트래픽 개인 정보 보호

AWS Deadline Cloud 는 연결을 보호하기 위해 Amazon Virtual Private Cloud(VPC)를 지원합니다. Amazon VPC는 Virtual Private Cloud(VPC)의 보안을 강화하고 모니터링하기 위해 사용할 수 있는 여러 가지 기능을 제공합니다.

VPC 내에서 실행되는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 사용하여 고객 관리형 플릿(CMF)을 설정할 수 있습니다. 사용할 Amazon VPC 엔드포인트를 배포하면 CMF의 작업자와 Deadline Cloud 엔드포인트 간의 AWS PrivateLink트래픽이 VPC 내에 유지됩니다. 또한 인스턴스에 대한 인터넷 액세스를 제한하도록 VPC를 구성할 수 있습니다.

서비스 관리형 플릿에서는 작업자가 인터넷에서 연결할 수 없지만 인터넷에 액세스할 수 있고 인터넷을 통해 Deadline Cloud 서비스에 연결됩니다. 각 서비스 관리형 플릿은 자체 격리된 네트워크에서 실행되며 작업자 인스턴스는 개별 고객 전용으로 유지됩니다.

옵트아웃

AWS Deadline Cloud 는 개발 및 개선에 도움이 되는 특정 운영 정보를 수집합니다 Deadline Cloud. 수집된 데이터에는 AWS 계정 ID 및 사용자 ID와 같은 항목이 포함되어 있으므로에 문제가 있는 경우 사용자를 올바르게 식별할 수 있습니다 Deadline Cloud. 또한 리소스 IDs(해당하는 경우 FarmID 또는 QueueID), 제품 이름(예: JobAttachments, WorkerAgent 등), 제품 버전과 같은 Deadline Cloud 특정 정보를 수집합니다.

애플리케이션 구성을 사용하여이 데이터 수집을 옵트아웃하도록 선택할 수 있습니다. 클라이언트 워크스테이션과 플릿 작업자 Deadline Cloud모두와 상호 작용하는 각 컴퓨터는 별도로 옵트아웃해야 합니다.

Deadline Cloud 모니터 - 데스크톱

Deadline Cloud 모니터 - 데스크톱은 충돌 발생 시점 및 애플리케이션 열기 시점과 같은 운영 정보를 수집하여 애플리케이션에 문제가 있는 시점을 파악하는 데 도움이 됩니다. 이 운영 정보 수집을 옵트아웃하려면 설정 페이지로 이동하여 데이터 수집 켜기를 선택 취소하여 Deadline Cloud Monitor의 성능을 측정합니다.

옵트아웃한 후에는 데스크톱 모니터가 더 이상 운영 데이터를 전송하지 않습니다. 이전에 수집된 모든 데이터는 유지되며 서비스를 개선하는 데 계속 사용될 수 있습니다. 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요.

AWS Deadline Cloud CLI 및 도구

AWS Deadline Cloud CLI, 제출자 및 작업자 에이전트는 모두 충돌이 발생하는 시기 및 작업이 제출되는 시기와 같은 운영 정보를 수집하여 이러한 애플리케이션에 문제가 있는 시기를 파악하는 데 도움이 됩니다. 이 운영 정보 수집을 옵트아웃하려면 다음 방법 중 하나를 사용합니다.

- 터미널에를 입력합니다**`deadline config set telemetry.opt_out true`**.

  이렇게 하면 현재 사용자로 실행될 때 CLI, 제출자 및 작업자 에이전트가 옵트아웃됩니다.
- Deadline Cloud 작업자 에이전트를 설치할 때 **`--telemetry-opt-out`** 명령줄 인수를 추가합니다. 예를 들어 **`./install.sh --farm-id $FARM_ID --fleet-id $FLEET_ID --telemetry-opt-out`**입니다.
- 작업자 에이전트, CLI 또는 제출자를 실행하기 전에 환경 변수를 설정합니다. **`DEADLINE_CLOUD_TELEMETRY_OPT_OUT=true`**

옵트아웃한 후에는 Deadline Cloud 도구가 더 이상 운영 데이터를 전송하지 않습니다. 이전에 수집된 모든 데이터는 유지되며 서비스를 개선하는 데 계속 사용될 수 있습니다. 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요.

Deadline Cloud의 Identity and Access Management

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 누가 Deadline Cloud 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청(참조AWS Deadline Cloud 자격 증명 및 액세스 문제 해결)
- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출(Deadline Cloud와 IAM의 작동 방식 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성(Deadline Cloud에 대한 자격 증명 기반 정책 예제 참조)

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

기타 정책 유형

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Deadline Cloud와 IAM의 작동 방식

IAM을 사용하여 Deadline Cloud에 대한 액세스를 관리하기 전에 Deadline Cloud에서 사용할 수 있는 IAM 기능을 알아봅니다.

AWS Deadline Cloud와 함께 사용할 수 있는 IAM 기능

IAM 특성	Deadline Cloud 지원
자격 증명 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACL	아니요
ABAC(정책의 태그)	예

IAM 특성	Deadline Cloud 지원
임시 보안 인증	예
전달 액세스 세션(FAS)	예
서비스 역할	예
서비스 연결 역할	아니요

Deadline Cloud 및 기타에서 대부분의 IAM 기능을 AWS 서비스 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

Deadline Cloud에 대한 자격 증명 기반 정책

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 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Deadline Cloud에 대한 자격 증명 기반 정책 예제

Deadline Cloud 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Deadline Cloud에 대한 자격 증명 기반 정책 예제.

Deadline Cloud 내의 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 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 정의합

니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM에서 교차 계정 리소스 액세스를 참조하세요.

Deadline Cloud에 대한 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

Deadline Cloud 작업 목록을 보려면 서비스 승인 참조의 AWS Deadline Cloud에서 정의한 작업을 참조하세요.

Deadline Cloud의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
deadline
```

단일 문에서 여러 작업을 지정하려면 쉼표로 구분합니다.

```
"Action": [
      "deadline:action1",
      "deadline:action2"
         ]
```

Deadline Cloud 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Deadline Cloud에 대한 자격 증명 기반 정책 예제.

Deadline Cloud에 대한 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Deadline Cloud 리소스 유형 및 해당 ARNs 목록을 보려면 서비스 승인 참조의 AWS Deadline Cloud에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 AWS Deadline Cloud에서 정의한 작업을 참조하세요.

Deadline Cloud 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Deadline Cloud에 대한 자격 증명 기반 정책 예제.

Deadline Cloud에 사용되는 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같음(equals) 또는 미만(less than)과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Deadline Cloud 조건 키 목록을 보려면 서비스 승인 참조의 AWS Deadline Cloud에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 AWS Deadline Cloud에서 정의한 작업을 참조하세요.

Deadline Cloud 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요Deadline Cloud에 대한 자격 증명 기반 정책 예제.

Deadline CloudACLs

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

ABAC와 Deadline Cloud

ABAC 지원(정책의 태그): 예

속성 기반 액세스 제어(ABAC)는 태그라고 불리는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. IAM 엔터티 및 AWS 리소스에 태그를 연결한 다음 보안 주체의 태그가 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계할 수 있습니다.

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.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

Deadline Cloud에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션 또는 전환 역할을 사용할 때 자동으로 생성됩니다. 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 것이 AWS 좋습니다. 자세한 내용은 IAM 사용 설명서의 IAM의 임시 보안 자격 증명 및 IAM으로 작업하는 AWS 서비스 섹션을 참조하세요.

Deadline Cloud에 대한 전달 액세스 세션

전달 액세스 세션(FAS) 지원: 예

전달 액세스 세션(FAS)은를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

Deadline Cloud의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스 AWS에 권한을 위임할 역할 생성을 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Deadline Cloud 기능이 중단될 수 있습니다. Deadline Cloud가 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

Deadline Cloud의 서비스 연결 역할

서비스 연결 역할 지원: 아니요

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 태스크를 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 나타나 AWS 계정 며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하세요. 서비스 연결 역할 열에서 Yes가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

Deadline Cloud에 대한 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할에는 Deadline Cloud 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARNs 형식을 포함하여 Deadline Cloud에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 AWS Deadline Cloud에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- Deadline Cloud 콘솔 사용
- 콘솔에 액세스하기 위한 정책
- 대기열에 작업을 제출하는 정책
- 라이선스 엔드포인트 생성을 허용하는 정책
- 특정 팜 대기열 모니터링을 허용하는 정책

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 Deadline Cloud 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

Deadline Cloud 콘솔 사용

AWS Deadline Cloud 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은에서 Deadline Cloud 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요는 없습니다. 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 Deadline Cloud 콘솔을 계속 사용할 수 있도록 하려면 Deadline Cloud *ConsoleAccess* 또는 *ReadOnly* AWS 관리형 정책도 엔터티에 연결합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

콘솔에 액세스하기 위한 정책

Deadline Cloud 콘솔의 모든 기능에 대한 액세스 권한을 부여하려면 전체 액세스 권한을 부여하려는 사용자 또는 역할에이 자격 증명 정책을 연결합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "EC2InstanceTypeSelection",
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeInstanceTypeOfferings",
            "ec2:DescribeInstanceTypes",
            "ec2:GetInstanceTypesFromInstanceRequirements",
            "pricing:GetProducts"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "VPCResourceSelection",
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ViewVpcLatticeResources",
        "Effect": "Allow",
        "Action": [
            "vpc-lattice:ListResourceConfigurations",
```

```
            "vpc-lattice:GetResourceConfiguration",
            "vpc-lattice:GetResourceGateway"
        ],
        "Resource": ["*"]
    },
    {
        "Sid": "ManageVpcEndpointsViaDeadline",
        "Effect": "Allow",
        "Action": [
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints",
        "ec2:CreateTags"
        ],
        "Resource": ["*"],
        "Condition": {
        "StringEquals": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ChooseJobAttachmentsBucket",
        "Effect": "Allow",
        "Action": ["s3:GetBucketLocation", "s3:ListAllMyBuckets"],
        "Resource": "*"
    },
    {
        "Sid": "CreateDeadlineCloudLogGroups",
        "Effect": "Allow",
        "Action": ["logs:CreateLogGroup"],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/deadline/*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ValidateDependencies",
        "Effect": "Allow",
        "Action": ["s3:ListBucket"],
        "Resource": "*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
```

```
        "Sid": "RoleSelection",
        "Effect": "Allow",
        "Action": ["iam:GetRole", "iam:ListRoles",
 "iam:ListAttachedRolePolicies"],
        "Resource": "*"
    },
    {
        "Sid": "PassRoleToDeadlineCloud",
        "Effect": "Allow",
        "Action": ["iam:PassRole"],
        "Condition": {
        "StringLike": { "iam:PassedToService": "deadline.amazonaws.com" }
    },
        "Resource": "*"
    },
    {
        "Sid": "KMSKeySelection",
        "Effect": "Allow",
        "Action": ["kms:ListKeys", "kms:ListAliases"],
        "Resource": "*"
    },
    {
        "Sid": "IdentityStoreReadOnly",
        "Effect": "Allow",
        "Action": [
            "identitystore:DescribeUser",
            "identitystore:DescribeGroup",
            "identitystore:ListGroups",
            "identitystore:ListUsers",
            "identitystore:IsMemberInGroups",
            "identitystore:ListGroupMemberships",
            "identitystore:ListGroupMembershipsForMember",
            "identitystore:GetGroupMembershipId"
    ],
        "Resource": "*"
    },
    {
        "Sid": "OrganizationAndIdentityCenterIdentification",
        "Effect": "Allow",
        "Action": [
            "sso:ListDirectoryAssociations",
            "organizations:DescribeAccount",
            "organizations:DescribeOrganization",
            "sso:DescribeRegisteredRegions",
```

```
            "sso:GetManagedApplicationInstance",
            "sso:GetSharedSsoConfiguration",
            "sso:ListInstances",
            "sso:GetApplicationAssignmentConfiguration",
            "sso:GetSSOStatus",
            "sso:ListRegions",
            "sso:DescribeRegion"
    ],
        "Resource": "*"
    },
    {
        "Sid": "ManagedDeadlineCloudIDCApplication",
        "Effect": "Allow",
        "Action": [
            "sso:CreateApplication",
            "sso:PutApplicationAssignmentConfiguration",
            "sso:PutApplicationAuthenticationMethod",
            "sso:PutApplicationGrant",
            "sso:DeleteApplication",
            "sso:UpdateApplication"
    ],
        "Resource": "*",
        "Condition": {
        "StringLike": { "aws:CalledViaFirst": "deadline.amazonaws.com" }
        }
    },
    {
        "Sid": "ChooseSecret",
        "Effect": "Allow",
        "Action": ["secretsmanager:ListSecrets"],
        "Resource": "*"
    },
    {
        "Sid": "DeadlineMembershipActions",
        "Effect": "Allow",
        "Action": [
            "deadline:AssociateMemberToFarm",
            "deadline:AssociateMemberToFleet",
            "deadline:AssociateMemberToQueue",
            "deadline:AssociateMemberToJob",
            "deadline:DisassociateMemberFromFarm",
            "deadline:DisassociateMemberFromFleet",
            "deadline:DisassociateMemberFromQueue",
            "deadline:DisassociateMemberFromJob",
```

```
            "deadline:ListFarmMembers",
            "deadline:ListFleetMembers",
            "deadline:ListQueueMembers",
            "deadline:ListJobMembers"
    ],
        "Resource": ["*"]
    },
    {
        "Sid": "DeadlineControlPlaneActions",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateMonitor",
            "deadline:GetMonitor",
            "deadline:UpdateMonitor",
            "deadline:DeleteMonitor",
            "deadline:ListMonitors",
            "deadline:CreateFarm",
            "deadline:GetFarm",
            "deadline:UpdateFarm",
            "deadline:DeleteFarm",
            "deadline:ListFarms",
            "deadline:CreateQueue",
            "deadline:GetQueue",
            "deadline:UpdateQueue",
            "deadline:DeleteQueue",
            "deadline:ListQueues",
            "deadline:CreateFleet",
            "deadline:GetFleet",
            "deadline:UpdateFleet",
            "deadline:DeleteFleet",
            "deadline:ListFleets",
            "deadline:ListWorkers",
            "deadline:CreateQueueFleetAssociation",
            "deadline:GetQueueFleetAssociation",
            "deadline:UpdateQueueFleetAssociation",
            "deadline:DeleteQueueFleetAssociation",
            "deadline:ListQueueFleetAssociations",
            "deadline:CreateQueueEnvironment",
            "deadline:GetQueueEnvironment",
            "deadline:UpdateQueueEnvironment",
            "deadline:DeleteQueueEnvironment",
            "deadline:ListQueueEnvironments",
            "deadline:CreateLimit",
            "deadline:GetLimit",
```

```
            "deadline:UpdateLimit",
            "deadline:DeleteLimit",
            "deadline:ListLimits",
            "deadline:CreateQueueLimitAssociation",
            "deadline:GetQueueLimitAssociation",
            "deadline:DeleteQueueLimitAssociation",
            "deadline:UpdateQueueLimitAssociation",
            "deadline:ListQueueLimitAssociations",
            "deadline:CreateStorageProfile",
            "deadline:GetStorageProfile",
            "deadline:UpdateStorageProfile",
            "deadline:DeleteStorageProfile",
            "deadline:ListStorageProfiles",
            "deadline:ListStorageProfilesForQueue",
            "deadline:ListBudgets",
            "deadline:TagResource",
            "deadline:UntagResource",
            "deadline:ListTagsForResource",
            "deadline:CreateLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:ListAvailableMeteredProducts",
            "deadline:ListMeteredProducts",
            "deadline:PutMeteredProduct",
            "deadline:DeleteMeteredProduct",
            "deadline:GetMonitorSettings",
            "deadline:UpdateMonitorSettings"
        ],
        "Resource": ["*"]
      }]
}
```

대기열에 작업을 제출하는 정책

이 예제에서는 특정 팜의 특정 대기열에 작업을 제출할 수 있는 권한을 부여하는 범위 축소 정책을 생성합니다.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SubmitJobsFarmAndQueue",
            "Effect": "Allow",
            "Action": "deadline:CreateJob",
            "Resource": "arn:aws:deadline:us-east-1:111122223333:farm/FARM_A/
queue/QUEUE_B/job/*"
        }
    ]
}
```

라이선스 엔드포인트 생성을 허용하는 정책

이 예제에서는 라이선스 엔드포인트를 생성하고 관리하는 데 필요한 권한을 부여하는 범위 축소 정책을 생성합니다. 이 정책을 사용하여 팜과 연결된 VPC에 대한 라이선스 엔드포인트를 생성합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "CreateLicenseEndpoint",
        "Effect": "Allow",
        "Action": [
            "deadline:CreateLicenseEndpoint",
            "deadline:DeleteLicenseEndpoint",
            "deadline:GetLicenseEndpoint",
            "deadline:ListLicenseEndpoints",
            "deadline:PutMeteredProduct",
            "deadline:DeleteMeteredProduct",
            "deadline:ListMeteredProducts",
            "deadline:ListAvailableMeteredProducts",
            "ec2:CreateVpcEndpoint",
            "ec2:DescribeVpcEndpoints",
            "ec2:DeleteVpcEndpoints"
        ],
        "Resource": [
            "arn:aws:deadline:*:111122223333:*",
            "arn:aws:ec2:*:111122223333:vpc-endpoint/*"
        ]
```

```
    }]
}
```

특정 팜 대기열 모니터링을 허용하는 정책

이 예제에서는 특정 팜의 특정 대기열에서 작업을 모니터링할 수 있는 권한을 부여하는 범위 축소 정책을 생성합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "MonitorJobsFarmAndQueue",
        "Effect": "Allow",
        "Action": [
            "deadline:SearchJobs",
            "deadline:ListJobs",
            "deadline:GetJob",
            "deadline:SearchSteps",
            "deadline:ListSteps",
            "deadline:ListStepConsumers",
            "deadline:ListStepDependencies",
            "deadline:GetStep",
            "deadline:SearchTasks",
            "deadline:ListTasks",
            "deadline:GetTask",
            "deadline:ListSessions",
            "deadline:GetSession",
            "deadline:ListSessionActions",
            "deadline:GetSessionAction"
        ],
        "Resource": [
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B",
            "arn:aws:deadline:us-east-1:123456789012:farm/FARM_A/queue/QUEUE_B/*"
        ]
    }]
}
```

AWS Deadline Cloud에 대한 관리형 정책

AWS 관리형 정책은에서 생성하고 관리하는 독립 실행형 정책입니다 AWS. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 관리형 정책에 정의된 권한을 AWS 업데이트하는 AWS 경우 업데이트는 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향을 미칩니다. AWS AWS 서비스 는 새가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 될 때 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용자 가이드의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: AWSDeadlineCloud-FleetWorker

`AWSDeadlineCloud-FleetWorker` 정책을 (IAM) 자격 증명에 연결할 수 있습니다 AWS Identity and Access Management .

이 정책은이 플릿의 작업자에게 서비스에 연결하고 서비스에서 작업을 수신하는 데 필요한 권한을 부여합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `deadline` - 보안 주체가 플릿에서 작업자를 관리할 수 있도록 허용합니다.

정책 세부 정보의 JSON 목록은 AWS 관리형 정책 참조 가이드의 AWSDeadlineCloud-FleetWorker를 참조하세요.

AWS 관리형 정책: AWSDeadlineCloud-WorkerHost

`AWSDeadlineCloud-WorkerHost` 정책을 IAM ID에 연결할 수 있습니다.

이 정책은 서비스에 처음 연결하는 데 필요한 권한을 부여합니다. Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 프로파일로 사용할 수 있습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `deadline` - 사용자가 작업자를 생성하고, 작업자의 플릿 역할을 수임하고, 작업자에 태그를 적용할 수 있도록 허용합니다.

정책 세부 정보의 JSON 목록은 AWS 관리형 정책 참조 가이드의 AWSDeadlineCloud-WorkerHost를 참조하세요.

AWS 관리형 정책: AWSDeadlineCloud-UserAccessFarms

`AWSDeadlineCloud-UserAccessFarms` 정책을 IAM ID에 연결할 수 있습니다.

이 정책을 통해 사용자는 자신이 속한 팜과 멤버십 수준에 따라 팜 데이터에 액세스할 수 있습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `deadline` - 사용자가 팜 데이터에 액세스할 수 있도록 허용합니다.
- `ec2` - 사용자가 Amazon EC2 인스턴스 유형에 대한 세부 정보를 볼 수 있도록 허용합니다.
- `identitystore` - 사용자가 사용자 및 그룹 이름을 볼 수 있도록 허용합니다.
- `kms` - 사용자가 AWS Key Management Service (IAM Identity Center AWS KMS) 인스턴스에 대해 AWS IAM Identity Center () 고객 관리형 키를 구성할 수 있습니다.

정책 세부 정보의 JSON 목록은 AWS 관리형 정책 참조 가이드의 AWSDeadlineCloud-UserAccessFarms를 참조하세요.

AWS 관리형 정책: AWSDeadlineCloud-UserAccessFleets

`AWSDeadlineCloud-UserAccessFleets` 정책을 IAM ID에 연결할 수 있습니다.

이 정책을 통해 사용자는 자신이 속한 팜과 멤버십 수준에 따라 플릿 데이터에 액세스할 수 있습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `deadline` - 사용자가 팜 데이터에 액세스할 수 있도록 허용합니다.
- `ec2` - 사용자가 Amazon EC2 인스턴스 유형에 대한 세부 정보를 볼 수 있도록 허용합니다.
- `identitystore` - 사용자가 사용자 및 그룹 이름을 볼 수 있도록 허용합니다.

정책 세부 정보의 JSON 목록은 AWS 관리형 정책 참조 가이드의 AWSDeadlineCloud-UserAccessFleets를 참조하세요.

AWS 관리형 정책: AWSDeadlineCloud-UserAccessJobs

`AWSDeadlineCloud-UserAccessJobs` 정책을 IAM ID에 연결할 수 있습니다.

이 정책을 통해 사용자는 자신이 속한 팜과 멤버십 수준에 따라 작업 데이터에 액세스할 수 있습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `deadline` - 사용자가 팜 데이터에 액세스할 수 있도록 허용합니다.
- `ec2` - 사용자가 Amazon EC2 인스턴스 유형에 대한 세부 정보를 볼 수 있도록 허용합니다.
- `identitystore` - 사용자가 사용자 및 그룹 이름을 볼 수 있도록 허용합니다.

정책 세부 정보의 JSON 목록은 AWS 관리형 정책 참조 가이드의 AWSDeadlineCloud-UserAccessJobs를 참조하세요.

AWS 관리형 정책: AWSDeadlineCloud-UserAccessQueues

`AWSDeadlineCloud-UserAccessQueues` 정책을 IAM ID에 연결할 수 있습니다.

이 정책을 통해 사용자는 자신이 속한 팜과 멤버십 수준에 따라 대기열 데이터에 액세스할 수 있습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `deadline` - 사용자가 팜 데이터에 액세스할 수 있도록 허용합니다.
- `ec2` - 사용자가 Amazon EC2 인스턴스 유형에 대한 세부 정보를 볼 수 있도록 허용합니다.
- `identitystore` - 사용자가 사용자 및 그룹 이름을 볼 수 있도록 허용합니다.

정책 세부 정보의 JSON 목록은 AWS 관리형 정책 참조 가이드의 AWSDeadlineCloud-UserAccessQueues를 참조하세요.

AWS 관리형 정책에 대한 기한 클라우드 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후 Deadline Cloud의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 Deadline Cloud Document History 페이지에서 RSS 피드를 구독하세요.

변경	설명	Date
AWSDeadlineCloud-UserAccessFarms – 변경	Deadline Cloud는 IAM Identity Center 인스턴스에서 AWS KMS 고객 관리형 키를 사용할 수 `kms:Decrypt` 있도록 새 작업을 추가했습니다.	2025년 12월 22일
AWSDeadlineCloud-WorkerHost – 변경	Deadline Cloud는 플릿의 작업자`deadline:TagResource` 와 연결된 태그를 추가하고 볼 수 `deadline:ListTagsForResource` 있도록 새로운 작업 및를 추가했습니다.	2025년 5월 30일
AWSDeadlineCloud-UserAccessFarms – 변경 AWSDeadlineCloud-UserAccessJobs – 변경	Deadline Cloud는 작업을 다시 제출할 수 `deadline:ListJobParameterDefinitions` 있도록 새 작업 `deadline:GetJobTemplate` 및를 추가했습니다.	2024년 10월 7일

변경	설명	Date
AWSDeadlineCloud-UserAccessQueues – 변경		
Deadline Cloud에서 변경 사항 추적 시작	Deadline Cloud가 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2024년 4월 2일

서비스 역할

Deadline Cloud가 IAM 서비스 역할을 사용하는 방법

Deadline Cloud는 IAM 역할을 자동으로 수임하고 작업자, 작업 및 Deadline Cloud 모니터에 임시 자격 증명을 제공합니다. 이 접근 방식은 역할 기반 액세스 제어를 통해 보안을 유지하면서 수동 자격 증명 관리를 제거합니다.

모니터, 플릿 및 대기열을 생성할 때 Deadline Cloud가 사용자를 대신하여 수임하는 IAM 역할을 지정합니다. 그런 다음 작업자와 Deadline Cloud 모니터는 이러한 역할로부터 액세스하기 위한 임시 자격 증명을 받습니다 AWS 서비스.

플릿 역할

Deadline Cloud 작업자에게 작업을 수신하고 해당 작업에 대한 진행 상황을 보고하는 데 필요한 권한을 부여하도록 플릿 역할을 구성합니다.

일반적으로이 역할을 직접 구성할 필요가 없습니다. 이 역할은 Deadline Cloud 콘솔에서 생성하여 필요한 권한을 포함할 수 있습니다. 다음 가이드를 사용하여 문제 해결을 위한이 역할의 세부 사항을 이해합니다.

프로그래밍 방식으로 플릿을 생성하거나 업데이트할 때 `CreateFleet` 또는 `UpdateFleet` API 작업을 사용하여 플릿 역할 ARN을 지정합니다.

플릿 역할이 수행하는 작업

플릿 역할은 작업자에게 다음과 같은 권한을 제공합니다.

- 새 작업을 수신하고 진행 중인 작업에 대한 진행 상황을 Deadline Cloud 서비스에 보고합니다.
- 작업자 수명 주기 및 상태 관리
- 작업자 로그에 대해 Amazon CloudWatch Logs에 로그 이벤트 기록

플릿 역할 신뢰 정책 설정

플릿 역할은 Deadline Cloud 서비스를 신뢰하고 특정 팜으로 범위를 지정해야 합니다.

가장 좋은 방법은 신뢰 정책에 혼동된 대리자 보호를 위한 보안 조건이 포함되어야 한다는 것입니다. 혼동된 대리자 보호에 대한 자세한 내용은 Deadline Cloud 사용 설명서의 혼동된 대리자를 참조하세요.

- `aws:SourceAccount`는 동일한의 리소스만이 역할을 수임 AWS 계정 할 수 있도록 합니다.
- `aws:SourceArn`는 역할 가정을 특정 Deadline Cloud 팜으로 제한합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDeadlineCredentialsService",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:REGION:YOUR_ACCOUNT_ID:farm/YOUR_FARM_ID"
        }
      }
    }
  ]
}
```

플릿 역할 권한 연결

플릿 역할에 다음 AWS 관리형 정책을 연결합니다.

AWSDeadlineCloud-FleetWorker

이 관리형 정책은 다음에 대한 권한을 제공합니다.

- `deadline:AssumeFleetRoleForWorker` - 작업자가 자격 증명을 새로 고칠 수 있습니다.

- `deadline:UpdateWorker` - 작업자가 상태를 업데이트할 수 있습니다(예: 종료 시 중지됨).
- `deadline:UpdateWorkerSchedule` - 작업 가져오기 및 진행 상황 보고.
- `deadline:BatchGetJobEntity` - 작업 정보를 가져오는 데 사용됩니다.
- `deadline:AssumeQueueRoleForWorker` - 작업 실행 중에 대기열 역할 자격 증명에 액세스하는 데 사용됩니다.

암호화된 팜에 대한 KMS 권한 추가

KMS 키를 사용하여 팜을 생성한 경우 플릿 역할에 이러한 권한을 추가하여 작업자가 팜의 암호화된 데이터에 액세스할 수 있도록 합니다.

KMS 권한은 팜에 연결된 KMS 키가 있는 경우에만 필요합니다. `kms:ViaService` 조건은 형식을 사용해야 합니다`deadline.{region}.amazonaws.com`.

플릿을 생성할 때 해당 플릿에 대한 CloudWatch Logs 로그 그룹이 생성됩니다. 작업자의 권한은 Deadline Cloud 서비스에서 특정 작업자에 대한 로그 스트림을 생성하는 데 사용됩니다. 작업자를 설정하고 실행한 후 작업자는 이러한 권한을 사용하여 로그 이벤트를 CloudWatch Logs로 직접 전송합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateLogStream",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "deadline.REGION.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "ManageLogEvents",
```

```
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
    },
    {
      "Sid": "ManageKmsKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey"
      ],
      "Resource": "YOUR_FARM_KMS_KEY_ARN",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "deadline.REGION.amazonaws.com"
        }
      }
    }
  ]
}
```

플릿 역할 수정

플릿 역할에 대한 권한은 사용자 지정할 수 없습니다. 설명된 권한은 항상 필요하며 추가 권한을 추가하는 것은 영향을 주지 않습니다.

고객 관리형 플릿 호스트 역할

Amazon EC2 인스턴스 또는 온프레미스 호스트에서 고객 관리형 플릿을 사용하는 경우 WorkerHost 역할을 설정합니다.

WorkerHost 역할이 수행하는 작업

WorkerHost 역할은 고객 관리형 플릿 호스트에서 작업자를 부트스트랩합니다. 호스트가 다음을 수행하는 데 필요한 최소한의 권한을 제공합니다.

- Deadline Cloud에서 작업자 생성
- 플릿 역할을 수임하여 운영 자격 증명을 가져옵니다.

- 플릿 태그를 사용하여 작업자에 태그 지정(태그 전파가 활성화된 경우)

WorkerHost 역할 권한 설정

다음 AWS 관리형 정책을 WorkerHost 역할에 연결합니다.

AWSDeadlineCloud-WorkerHost

이 관리형 정책은 다음에 대한 권한을 제공합니다.

- `deadline:CreateWorker` - 호스트가 새 작업자를 등록할 수 있도록 허용합니다.
- `deadline:AssumeFleetRoleForWorker` - 호스트가 플릿 역할을 수임하도록 허용합니다.
- `deadline:TagResource` - 생성 중에 작업자에 태그를 지정할 수 있습니다(활성화된 경우).
- `deadline:ListTagsForResource` - 전파를 위해 플릿 태그를 읽을 수 있습니다.

부트스트랩 프로세스 이해

WorkerHost 역할은 초기 작업자 시작 중에만 사용됩니다.

1. 작업자 에이전트는 WorkerHost 자격 증명을 사용하여 호스트에서 시작합니다.
2. Deadline Cloud에 등록`deadline:CreateWorker`하기 위해를 호출합니다.
3. 그런 다음를 호출`deadline:AssumeFleetRoleForWorker`하여 플릿 역할 자격 증명을 가져옵니다.
4. 이 시점부터 작업자는 모든 작업에 플릿 역할 자격 증명만 사용합니다.

작업자 실행을 시작한 후에는 WorkerHost 역할이 사용되지 않습니다. 서비스 관리형 플릿에는이 정책이 필요하지 않습니다. 서비스 관리형 플릿에서는 부트스트래핑이 자동으로 수행됩니다.

대기열 역할

대기열 역할은 작업을 처리할 때 작업자가 수임합니다. 이 역할은 작업을 완료하는 데 필요한 권한을 제공합니다.

프로그래밍 방식으로 대기열을 생성하거나 업데이트할 때 `CreateQueue` 또는 `UpdateQueue` API 작업을 사용하여 대기열 역할 ARN을 지정합니다.

대기열 역할 신뢰 정책 설정

대기열 역할은 Deadline Cloud 서비스를 신뢰해야 합니다.

가장 좋은 방법은 신뢰 정책에 혼동된 대리자 보호를 위한 보안 조건이 포함되어야 한다는 것입니다. 혼동된 대리자 보호에 대한 자세한 내용은 Deadline Cloud 사용 설명서의 혼동된 대리자를 참조하세요.

- `aws:SourceAccount`는 동일한의 리소스만이 역할을 수임할 AWS 계정 수 있도록 합니다.
- `aws:SourceArn`는 역할 가정을 특정 Deadline Cloud 팜으로 제한합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "credentials.deadline.amazonaws.com",
          "deadline.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:deadline:us-west-2:123456789012:farm/{farm-id}"
        }
      }
    }
  ]
}
```

대기열 역할 권한 이해

대기열 역할은 단일 관리형 정책을 사용하지 않습니다. 대신 콘솔에서 대기열을 구성하면 Deadline Cloud는 구성에 따라 대기열에 대한 사용자 지정 정책을 생성합니다.

이 자동으로 생성된 정책은 다음에 대한 액세스를 제공합니다.

작업 첨부 파일

작업 입력 및 출력 파일에 대해 지정된 Amazon S3 버킷에 대한 읽기 및 쓰기 액세스 권한:

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:ListBucket",
    "s3:GetBucketLocation"
  ],
  "Resource": [
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET",
    "arn:aws:s3:::YOUR_JOB_ATTACHMENTS_BUCKET/YOUR_PREFIX/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceAccount": "YOUR_ACCOUNT_ID"
    }
  }
}
```

작업 로그

이 대기열의 작업에 대한 CloudWatch Logs 읽기 액세스. 각 대기열에는 자체 로그 그룹이 있고 각 세션에는 자체 로그 스트림이 있습니다.

```
{
  "Effect": "Allow",
  "Action": [
    "logs:GetLogEvents"
  ],
  "Resource": "arn:aws:logs:REGION:YOUR_ACCOUNT_ID:log-group:/aws/
deadline/YOUR_FARM_ID/*"
}
```

타사 소프트웨어

Deadline Cloud에서 지원하는 타사 소프트웨어(예: Maya, Blender 등)를 다운로드할 수 있는 액세스 권한:

```
{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
```

```
    "s3:GetObject"
  ],
  "Resource": "*",
  "Condition": {
    "ArnLike": {
      "s3:DataAccessPointArn": "arn:aws:s3:*:*:accesspoint/deadline-software-*"
    },
    "StringEquals": {
      "s3:AccessPointNetworkOrigin": "VPC"
    }
  }
}
```

작업에 대한 권한 추가

작업에 액세스해야 AWS 서비스 하는에 대한 권한을 대기열 역할에 추가합니다. OpenJobDescription 단계 스크립트를 작성할 때 AWS CLI 및 SDK는 대기열 역할의 자격 증명을 자동으로 사용합니다. 이를 사용하여 작업을 완료하는 데 필요한 추가 서비스에 액세스합니다.

사용 사례 예시:

- 사용자 지정 데이터 가져오기
- 사용자 지정 라이선스 서버로 터널링할 수 있는 SSM 권한
- 사용자 지정 지표를 내보내기 위한 CloudWatch
- 동적 워크플로를 위한 새 작업을 생성할 수 있는 Deadline Cloud 권한

대기열 역할 자격 증명 사용 방법

Deadline Cloud는 대기열 역할 자격 증명을 제공하여 다음을 수행합니다.

- 작업 실행 중 작업자
- Deadline Cloud CLI를 통한 사용자 및 작업 연결 및 로그와 상호 작용할 때 모니터링

Deadline Cloud는 각 대기열에 대해 별도의 CloudWatch Logs 로그 그룹을 생성합니다. 작업은 대기열 역할 자격 증명을 사용하여 대기열의 로그 그룹에 로그를 씁니다. Deadline Cloud CLI 및 모니터는 대기열 역할(를 통해`deadline:AssumeQueueRoleForRead`)을 사용하여 대기열의 로그 그룹에서 작업 로그를 읽습니다. Deadline Cloud CLI 및 모니터는 대기열 역할(를 통해`deadline:AssumeQueueRoleForUser`)을 사용하여 작업 첨부 파일 데이터를 업로드하거나 다운로드합니다.

역할 모니터링

Deadline Cloud 모니터 웹 및 데스크톱 애플리케이션에 Deadline Cloud 리소스에 대한 액세스 권한을 부여하도록 모니터 역할을 구성합니다.

프로그래밍 방식으로 모니터를 생성하거나 업데이트할 때 `CreateMonitor` 또는 `UpdateMonitor` API 작업을 사용하여 모니터 역할 ARN을 지정합니다.

모니터 역할이 수행하는 작업

모니터 역할을 통해 Deadline Cloud Monitor는 최종 사용자에게 다음에 대한 액세스 권한을 제공할 수 있습니다.

- Deadline Cloud Integrated Submitters, CLI 및 모니터에 필요한 기본 기능
- 최종 사용자를 위한 사용자 지정 기능

모니터 역할 신뢰 정책 설정

모니터 역할은 Deadline Cloud 서비스를 신뢰해야 합니다.

가장 좋은 방법은 신뢰 정책에 혼동된 대리자 보호를 위한 보안 조건이 포함되어야 한다는 것입니다. 혼동된 대리자 보호에 대한 자세한 내용은 Deadline Cloud 사용 설명서의 혼동된 대리자를 참조하세요.

`aws:SourceAccount`는 동일한의 리소스만이 역할을 수임할 AWS 계정 수 있도록 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.deadline.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "YOUR_ACCOUNT_ID"
        }
      }
    }
  ]
```

```
}
```

모니터 역할 권한 연결

기본 작업을 위해 모니터 역할에 다음 AWS 관리형 정책을 모두 연결합니다.

- AWSDeadlineCloud-UserAccessFarms
- AWSDeadlineCloud-UserAccessFleets
- AWSDeadlineCloud-UserAccessJobs
- AWSDeadlineCloud-UserAccessQueues

모니터 역할의 작동 방식

Deadline Cloud 모니터를 사용하면 서비스 사용자가 AWS IAM Identity Center (IAM Identity Center)를 사용하여 로그인하고 모니터 역할이 수임됩니다. 위임된 역할 자격 증명은 모니터 애플리케이션에서 팜, 플릿, 대기열 및 기타 정보 목록을 포함하여 모니터 UI를 표시하는 데 사용됩니다.

Deadline Cloud Monitor 데스크톱 애플리케이션을 사용하는 경우 최종 사용자가 제공한 프로필 이름에 해당하는 명명된 자격 증명 프로필을 사용하여 워크스테이션에서 이러한 AWS 자격 증명을 추가로 사용할 수 있습니다. AWS SDK 및 도구 참조 가이드에서 명명된 프로파일에 대해 자세히 알아봅니다.

이 명명된 프로파일은 Deadline CLI 및 제출자가 Deadline Cloud 리소스에 액세스하는 방법입니다.

고급 사용 사례에 맞게 모니터 역할 사용자 지정

모니터 역할을 사용자 지정하여 사용자가 각 액세스 수준(뷰어, 기고자, 관리자, 소유자)에서 수행할 수 있는 작업을 수정하거나 고급 워크플로에 대한 권한을 추가할 수 있습니다.

액세스 수준 권한 사용자 지정

모니터 역할에 연결된 4개의 AWS 관리형 정책은 각 액세스 수준이 수행할 수 있는 작업을 제어합니다. 모니터 역할에 사용자 지정 정책을 추가하여 `deadline:MembershipLevel` 조건 키를 사용하여 특정 액세스 수준에 대한 권한을 부여하거나 제한할 수 있습니다.

예를 들어 기여자가 작업(일반적으로 관리자 및 소유자로 제한됨)을 업데이트하고 취소하도록 허용하려면 다음과 같은 정책을 추가합니다.

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

```
      "Effect": "Allow",
      "Action": "deadline:UpdateJob",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "deadline:MembershipLevel": "CONTRIBUTOR"
        }
      }
    }
  ]
}
```

이 정책을 사용하면 기여자는 작업을 제출하는 것 외에도 작업을 업데이트하고 취소할 수 있습니다.

고급 워크플로에 대한 권한 추가

모니터 역할에 사용자 지정 IAM 정책을 추가하여 모든 모니터 사용자에게 추가 권한을 부여할 수 있습니다. 이는 사용자가 표준 Deadline Cloud 기능 AWS 서비스 이상으로에 액세스해야 하는 고급 스크립팅 워크플로에 유용합니다.

모니터 역할을 수정할 때 다음 지침을 따르십시오.

- 관리형 정책을 제거하지 마세요. 이러한 정책을 제거하면 모니터 기능이 중단됩니다.

Deadline Cloud Monitor가 모니터 역할 자격 증명을 사용하는 방법

Deadline Cloud Monitor는 인증 시 모니터 역할 자격 증명을 자동으로 가져옵니다. 이 기능을 사용하면 데스크톱 애플리케이션이 표준 웹 브라우저에서 사용할 수 있는 것 이상의 향상된 모니터링 기능을 제공할 수 있습니다.

Deadline Cloud 모니터로 로그인하면 AWS CLI 또는 다른 AWS 도구와 함께 사용할 수 있는 프로필이 자동으로 생성됩니다. 이 프로필은 모니터 역할 자격 증명을 사용하여 모니터 역할의 권한에 AWS 서비스 따라에 프로그래밍 방식으로 액세스할 수 있습니다.

Deadline Cloud 제출자는 동일한 방식으로 작동합니다. Deadline Cloud 모니터에서 생성한 프로파일을 사용하여 적절한 역할 권한 AWS 서비스 으로에 액세스합니다.

Deadline Cloud 역할의 고급 사용자 지정

추가 권한으로 Deadline Cloud 역할을 확장하여 기본 렌더링 워크플로를 넘어 고급 사용 사례를 활성화할 수 있습니다. 이 접근 방식은 Deadline Cloud의 액세스 관리 시스템을 활용하여 대기열 멤버십에 AWS 서비스 따라 추가에 대한 액세스를 제어합니다.

와의 팀 협업 AWS CodeCommit

대기열 역할에 AWS CodeCommit 권한을 추가하여 프로젝트 리포지토리에서 팀 협업을 활성화합니다. 이 접근 방식은 추가 사용 사례에 Deadline Cloud의 액세스 관리 시스템을 사용합니다. 특정 대기열에 액세스할 수 있는 사용자만 이러한 AWS CodeCommit 권한을 받게 되므로 Deadline Cloud 대기열 멤버십을 통해 프로젝트별 리포지토리 액세스를 관리할 수 있습니다.

이는 아티스트가 렌더링 워크플로의 일부로 AWS CodeCommit 리포지토리에 저장된 프로젝트별 자산, 스크립트 또는 구성 파일에 액세스해야 하는 시나리오에 유용합니다.

대기열 역할에 AWS CodeCommit 권한 추가

대기열 역할에 다음 권한을 추가하여 AWS CodeCommit 액세스를 활성화합니다.

```
{
  "Effect": "Allow",
  "Action": [
    "codecommit:GitPull",
    "codecommit:GitPush",
    "codecommit:GetRepository",
    "codecommit:ListRepositories"
  ],
  "Resource": "arn:aws:codecommit:REGION:YOUR_ACCOUNT_ID:PROJECT_REPOSITORY"
}
```

아티스트 워크스테이션에서 자격 증명 공급자 설정

AWS CodeCommit 액세스에 Deadline Cloud 대기열 자격 증명을 사용하도록 각 아티스트 워크스테이션을 구성합니다. 이 설정은 워크스테이션당 한 번 수행됩니다.

자격 증명 공급자를 구성하려면

1. 자격 증명 공급자 프로파일을 AWS 구성 파일(~/.aws/config)에 추가합니다.

   ```
   [profile queue-codecommit]
   credential_process = deadline queue export-credentials --farm-id farm-
   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX --queue-id queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   ```

2. AWS CodeCommit 리포지토리에이 프로파일을 사용하도록 Git을 구성합니다.

   ```
   git config --global credential.https://git-codecommit.REGION.amazonaws.com.helper
    '!aws codecommit credential-helper --profile queue-codecommit $@'
   ```

```
git config --global credential.https://git-
codecommit.REGION.amazonaws.com.UseHttpPath true
```

farm-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 및 *queue-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX#* 실제 팜 및 대기열 IDs. *REGION*을 AWS 리전(예: us-west-2)으로 바꿉니다.

대기열 자격 증명 AWS CodeCommit 과 함께 사용

구성되면 Git 작업은 AWS CodeCommit 리포지토리에 액세스할 때 대기열 역할 자격 증명을 자동으로 사용합니다. deadline queue export-credentials 명령은 다음과 같은 임시 자격 증명을 반환합니다.

```
{
  "Version": 1,
  "AccessKeyId": "ASIA...",
  "SecretAccessKey": "...",
  "SessionToken": "...",
  "Expiration": "2025-11-10T23:02:23+00:00"
}
```

이러한 자격 증명은 필요에 따라 자동으로 새로 고쳐지며 Git 작업은 원활하게 작동합니다.

```
git clone https://git-codecommit.REGION.amazonaws.com/v1/repos/PROJECT_REPOSITORY
git pull
git push
```

이제 아티스트는 별도의 AWS CodeCommit 자격 증명 없이 대기열 권한을 사용하여 프로젝트 리포지토리에 액세스할 수 있습니다. 특정 대기열에 액세스할 수 있는 사용자만 연결된 리포지토리에 액세스할 수 있으므로 Deadline Cloud의 대기열 멤버십 시스템을 통해 세분화된 액세스 제어를 사용할 수 있습니다.

AWS Deadline Cloud 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 Deadline Cloud 및 IAM 작업 시 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- Deadline Cloud에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음

- 내 외부의 사람이 내 Deadline Cloud 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

Deadline Cloud에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 표시되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 deadline:*GetWidget* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 deadline:GetWidget on resource: my-example-widget
```

이 경우, deadline:*GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 권한이 없다는 오류가 수신되면 Deadline Cloud에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예제 오류는 라는 IAM 사용자가 콘솔을 사용하여 Deadline Cloud에서 작업을 수행하려고 marymajor 할 때 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 iam:PassRole작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 Deadline Cloud 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

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 제

어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- Deadline Cloud가 이러한 기능을 지원하는지 여부를 알아보려면 섹션을 참조하세요Deadline Cloud와 IAM의 작동 방식.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

에 대한 규정 준수 검증 Deadline Cloud

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 범위 내를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

의 복원력 Deadline Cloud

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.는 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제공하며,이 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워킹과 연결됩니다. 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.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS Deadline Cloud 는 작업 연결 S3 버킷에 저장된 데이터를 백업하지 않습니다. S3 버전 관리 또는 와 같은 표준 Amazon S3 백업 메커니즘을 사용하여 작업 연결 데이터의 백업을 활성화할 수 있습니다AWS Backup. S3

Deadline Cloud의 인프라 보안

관리형 서비스인 AWS Deadline Cloud는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요 AWS .

AWS 게시된 API 호출을 사용하여 네트워크를 통해 Deadline Cloud에 액세스합니다. 클라이언트는 다음을 지원해야 합니다.

- Transport Layer Security(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

Deadline Cloud는 AWS PrivateLink Virtual Private Cloud(VPC) 엔드포인트 정책 사용을 지원하지 않습니다. 엔드포인트에 대한 전체 액세스 권한을 부여하는 AWS PrivateLink 기본 정책을 사용합니다. 자세한 내용은 AWS PrivateLink 사용 설명서의 기본 엔드포인트 정책을 참조하세요.

Deadline Cloud의 구성 및 취약성 분석

AWS 는 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.

- 공동 책임 모델
- Amazon Web Services: 보안 프로세스의 개요(백서)

AWS Deadline Cloud는 서비스 관리형 또는 고객 관리형 플릿에서 작업을 관리합니다.

- 서비스 관리형 플릿의 경우 Deadline Cloud는 게스트 운영 체제를 관리합니다.
- 고객 관리형 플릿의 경우 운영 체제를 관리할 책임은 사용자에게 있습니다.

AWS Deadline Cloud의 구성 및 취약성 분석에 대한 자세한 내용은 섹션을 참조하세요.

- Deadline Cloud의 보안 모범 사례

교차 서비스 혼동된 대리인 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 가장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(직접 호출하는 서비스)가 다른 서비스(직접 호출되는 서비스)를 직접 호출할 때 발생할 수 있습니다. 직접 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 위탁자를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

리소스 정책에서 aws:SourceArn 및 aws:SourceAccount 전역 조건 컨텍스트 키를 사용하여 리소스에 다른 서비스를 AWS Deadline Cloud 제공하는 권한을 제한하는 것이 좋습니다. 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 `aws:SourceArn`을 사용하세요. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 `aws:SourceAccount`를 사용하세요.

혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 Amazon 리소스 이름(ARN)이 포함된 `aws:SourceArn` 전역 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모르거나 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드 문자(*)를 포함한 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용합니다. 예를 들어 `arn:aws:deadline:*:`*`123456789012`*`:*`입니다.

만약 `aws:SourceArn` 값에 Amazon S3 버킷 ARN과 같은 계정 ID가 포함되어 있지 않은 경우, 권한을 제한하려면 두 글로벌 조건 컨텍스트 키를 모두 사용해야 합니다.

다음 예제에서는의 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 Deadline Cloud 방지하는 방법을 보여줍니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
```

```
      "Service": "deadline.amazonaws.com"
    },
    "Action": "deadline:CreateFarm",
    "Resource": [
      "*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:deadline:*:111122223333:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "111122223333"
      }
    }
  }
}
```

인터페이스 엔드포인트를 AWS Deadline Cloud 사용한 액세스 (AWS PrivateLink)

AWS PrivateLink 를 사용하여 VPC와 간에 프라이빗 연결을 생성할 수 있습니다 AWS Deadline Cloud. 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 Direct Connect 연결을 사용하지 않고 VPC에 있는 Deadline Cloud 것처럼에 액세스할 수 있습니다. VPC의 인스턴스에서 Deadline Cloud API에 액세스하는 데는 퍼블릭 IP 주소가 필요하지 않습니다.

AWS PrivateLink에서 제공되는 인터페이스 엔드포인트를 생성하여 이 프라이빗 연결을 설정합니다. 인터페이스 엔드포인트에 대해 사용 설정하는 각 서브넷에서 엔드포인트 네트워크 인터페이스를 생성합니다. 이는 Deadline Cloud로 향하는 트래픽의 진입점 역할을 하는 요청자 관리형 네트워크 인터페이스입니다.

Deadline Cloud 에는 듀얼 스택 엔드포인트도 있습니다. 듀얼 스택 엔드포인트는 IPv6 및 IPv4를 통한 요청을 지원합니다.

자세한 내용은 AWS PrivateLink 안내서의 AWS PrivateLink를 통해 AWS 서비스 에 액세스를 참조하세요.

에 대한 고려 사항 Deadline Cloud

에 대한 인터페이스 엔드포인트를 설정하기 전에 AWS PrivateLink 가이드의 인터페이스 VPC 엔드포인트를 사용하여 AWS 서비스 액세스를 Deadline Cloud참조하세요.

Deadline Cloud 는 인터페이스 엔드포인트를 통해 모든 API 작업을 호출할 수 있도록 지원합니다.

기본적으로 인터페이스 엔드포인트를 통해에 대한 전체 액세스 Deadline Cloud 가 허용됩니다. 또는 보안 그룹을 엔드포인트 네트워크 인터페이스와 연결하여 인터페이스 엔드포인트를 Deadline Cloud 통해에 대한 트래픽을 제어할 수 있습니다.

Deadline Cloud 는 VPC 엔드포인트 정책도 지원합니다. 자세한 정보는 AWS PrivateLink 가이드의 엔드포인트 정책을 사용하여 VPC 엔드포인트에 대한 액세스 제어를 참조하세요.

Deadline Cloud 엔드포인트

Deadline Cloud 는를 사용하여 서비스에 액세스하는 데 4개의 엔드포인트를 사용합니다. AWS PrivateLink 하나는 IPv4용이고 다른 하나는 IPv6용입니다.

작업자는 `scheduling.deadline.region.amazonaws.com` 엔드포인트를 사용하여 대기열에서 작업을 가져오고, 진행 상황을 보고하고 Deadline Cloud, 작업 출력을 다시 보냅니다. 고객 관리형 플릿을 사용하는 경우 관리 작업을 사용하지 않는 한 생성해야 하는 유일한 엔드포인트는 예약 엔드포인트입니다. 예를 들어 작업이 더 많은 작업을 생성하는 경우 관리 엔드포인트가 `CreateJob` 작업을 호출하도록 활성화해야 합니다.

Deadline Cloud 모니터는 `management.deadline.region.amazonaws.com`를 사용하여 대기열 및 플릿을 생성 및 수정하거나 작업, 단계 및 작업 목록을 가져오는 등 팜의 리소스를 관리합니다.

AWS SDKs 및 CLI는 엔드포인트에 `management` 및 `scheduling` 접두사를 자동으로 추가합니다. 이 동작을 비활성화하려면 SDK 및 도구 참조 안내서의 호스트 접두사 삽입 섹션을 참조하세요. AWS SDKs

Deadline Cloud 에는 다음 AWS 서비스 엔드포인트에 대한 엔드포인트도 필요합니다.

- 인터넷 연결이 없는 서브넷에서 고객 관리형 플릿을 설정하는 경우 작업자가 로그를 작성할 수 있도록 Amazon CloudWatch Logs에 대한 VPC 엔드포인트를 생성해야 합니다. 자세한 내용은 CloudWatch를 사용한 모니터링을 참조하세요.
- 작업 첨부 파일을 사용하는 경우 작업자가 첨부 파일에 액세스할 수 있도록 Amazon Simple Storage Service(Amazon S3)용 VPC 엔드포인트를 생성해야 합니다. 자세한 내용은 의 작업 첨부 Deadline Cloud 파일을 참조하세요.

에 대한 엔드포인트 생성 Deadline Cloud

Amazon VPC 콘솔 또는 AWS Command Line Interface ()를 Deadline Cloud 사용하여 용 인터페이스 엔드포인트를 생성할 수 있습니다AWS CLI. 자세한 내용은 AWS PrivateLink 안내서의 인터페이스 엔드포인트 생성을 참조하세요.

다음 서비스 이름을 Deadline Cloud 사용하여에 대한 관리 및 예약 엔드포인트를 생성합니다. *region*을 배포한 AWS 리전 로 바꿉니다 Deadline Cloud.

```
com.amazonaws.region.deadline.management
```

```
com.amazonaws.region.deadline.scheduling
```

Deadline Cloud 는 듀얼 스택 엔드포인트를 지원합니다.

인터페이스 엔드포인트에 대해 프라이빗 DNS를 활성화하면 기본 리전 DNS 이름을 Deadline Cloud 사용하여에 API 요청을 할 수 있습니다. 예를 들어 작업자 작업의 `scheduling.deadline.us-east-1.amazonaws.com` 경우 , 다른 모든 작업`management.deadline.us-east-1.amazonaws.com`의 경우 입니다.

고객 관리형 플릿이 인터넷 연결 없이 서브넷에 있는 경우 다음 서비스 이름을 사용하여 CloudWatch Logs 엔드포인트를 생성해야 합니다.

```
com.amazonaws.region.logs
```

작업 첨부 파일을 사용하여 파일을 전송하는 경우 다음 서비스 이름을 사용하여 Amazon S3 엔드포인트를 생성해야 합니다.

```
com.amazonaws.region.s3
```

제한된 네트워크 환경

Deadline Cloud는 아티스트 또는 로컬 워크스테이션의 다른 사용자가 사용하는 도구를 제공합니다. 이러한 도구를 사용하려면 API AWS 및 웹 엔드포인트에 액세스해야 기능을 수행할 수 있습니다. 차세대 방화벽(NGFW) 또는 보안 웹 게이트웨이(SWG)와 같은 웹 콘텐츠 필터링 솔루션을 사용하여 특정 AWS 도메인 또는 URL 엔드포인트에 대한 액세스를 필터링하는 경우 웹 콘텐츠 필터링 솔루션 허용 목록에 다음 도메인 또는 URL 엔드포인트를 추가해야 합니다.

AWS 허용 목록에 대한 API 엔드포인트

, 모니터 AWS Management Console, CLI 및 통합 제출자와 같은 Deadline Cloud 클라이언트 도구에는 Deadline Cloud 외에도 AWS APIs에 대한 액세스 권한이 필요합니다. 이러한 엔드포인트는 IPv4만 지원합니다.

- `scheduling.deadline.[Region].amazonaws.com`
- `management.deadline.[Region].amazonaws.com`
- `logs.[Region].amazonaws.com`
- `ec2.[Region].amazonaws.com`
- `s3.[Region].amazonaws.com`
- `sts.[Region].amazonaws.com`
- `identitystore.[Region].amazonaws.com`

허용 목록에 추가할 웹 도메인

Deadline Cloud 모니터를 작동하려면 다음 도메인에 액세스해야 합니다.

AWS 로그인을 위한 도메인 허용 목록에 대한 자세한 내용은 AWS 로그인 사용 설명서의 허용 목록에 추가할 도메인을 참조하세요.

- `downloads.deadlinecloud.amazonaws.com`
- `d2ev1rdnjzhmnr.cloudfront.net`
- `prod.log.shortbread.aws.dev`
- `prod.tools.shortbread.aws.dev`
- `prod.log.shortbread.analytics.console.aws.a2z.com`
- `prod.tools.shortbread.analytics.console.aws.a2z.com`
- `global.help-panel.docs.aws.a2z.com`
- `[Region].signin.aws`
- `[Region].signin.aws.amazon.com`
- `sso.[Region].amazonaws.com`
- `portal.sso.[Region].amazonaws.com`

- `oidc.[Region].amazonaws.com`
- `assets.sso-portal.[Region].amazonaws.com`

허용 목록에 대한 환경별 엔드포인트

이러한 도메인은 Deadline Cloud의 특정 구성에 따라 달라집니다. 추가 Deadline Cloud 모니터 또는 대기열이 생성된 경우 추가 도메인을 허용 목록에 추가해야 합니다.

- *[Directory ID or alias]*`.awsapps.com`

  이 도메인은 IAM Identity Center 설정에 연결되어 있으며 동일한 IAM Identity Center 인스턴스를 사용하는이의 모든 설정에 대해 동일해야 합니다. 엔터프라이즈 관리자는 IAM Identity Center 콘솔의 설정 → AWS 액세스 포털 URL에서 정확한 값을 찾을 수 있습니다.

- *[Monitor alias].[Region]*`.deadlinecloud.amazonaws.com`

  이 도메인은 Deadline Cloud의 Monitor 설정을 위한 것입니다. 아티스트는 브라우저 또는 Deadline Cloud Monitor 애플리케이션에이 링크를 입력합니다. 향후 추가 계정 또는 리전에 Deadline Cloud가 설정된 경우이 도메인이 변경됩니다. 이 값은 대시보드 → 모니터 개요 → 모니터 세부 정보 → URL의 Deadline Cloud 콘솔에서 찾을 수 있습니다.

- *[Bucket name].[Region]*`.s3.amazonaws.com`

  이 도메인은 Deadline Cloud 대기열에서 사용하는 작업 연결 버킷의 도메인입니다. 각 대기열에는 고유한 작업 연결 버킷이 구성될 수 있습니다. 정확한 버킷 이름은 Deadline Cloud 콘솔의 대기열 → 대기열 세부 정보 → 작업 첨부 파일에서 찾을 수 있습니다. 작업 연결에 대한 자세한 내용은 대기열 설명서를 참조하세요.

Deadline Cloud의 보안 모범 사례

AWS Deadline Cloud(Deadline Cloud)는 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 사용자의 환경에 적절하지 않거나 충분하지 않을 수 있으므로 규정이 아닌 참고용으로만 사용하세요.

> **Note**
>
> 많은 보안 주제의 중요성에 대한 자세한 내용은 공동 책임 모델을 참조하세요.

데이터 보호

데이터 보호를 위해 자격 AWS 계정 증명을 보호하고 AWS Identity and Access Management (IAM)을 사용하여 개별 계정을 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail.
- 내의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon Simple Storage Service(S3)에 저장된 개인 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령행 인터페이스 또는 API를 통해 AWS 에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우, FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2 섹션을 참조하세요.

명칭 필드와 같은 자유 형식 필드에 고객 계정 번호와 같은 중요 식별 정보를 절대 입력하지 마세요. 이 권장 사항에는 AWS Deadline Cloud 또는 기타에서 콘솔 AWS CLI, API 또는 AWS SDKs를 AWS 서비스 사용하여 작업하는 경우가 포함됩니다. Deadline Cloud 또는 기타 서비스에 입력하는 모든 데이터가 진단 로그에 포함되도록 선택될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함하지 마십시오.

AWS Identity and Access Management 권한

사용자, AWS Identity and Access Management (IAM) 역할을 사용하고 사용자에게 최소 권한을 부여하여 AWS 리소스에 대한 액세스를 관리합니다. AWS 액세스 자격 증명을 생성, 배포, 교체 및 취소하기 위한 자격 증명 관리 정책 및 절차를 수립합니다. 자세한 설명은 IAM 사용자 가이드의 IAM 모범 사례 섹션을 참조하세요.

사용자 및 그룹으로 작업 실행

Deadline Cloud에서 대기열 기능을 사용하는 경우 OS 사용자에게 대기열 작업에 대한 최소 권한 권한이 있도록 운영 체제(OS) 사용자와 기본 그룹을 지정하는 것이 가장 좋습니다.

“사용자로 실행”(및 그룹)을 지정하면 대기열에 제출된 작업의 모든 프로세스가 해당 OS 사용자를 사용하여 실행되고 해당 사용자의 연결된 OS 권한을 상속합니다.

플릿 및 대기열 구성이 결합되어 보안 태세를 설정합니다. 대기열 측에서 “사용자로 작업 실행” 및 IAM 역할을 지정하여 대기열 작업에 OS 및 AWS 권한을 사용할 수 있습니다. 플릿은 특정 대기열에 연결된 경우 대기열 내에서 작업을 실행하는 인프라(작업자 호스트, 네트워크, 탑재된 공유 스토리지)를 정의합니다. 작업자 호스트에서 사용할 수 있는 데이터는 하나 이상의 연결된 대기열의 작업에서 액세스해야 합니다. 사용자 또는 그룹을 지정하면 다른 대기열, 설치된 다른 소프트웨어 또는 작업자 호스트에 액세스할 수 있는 다른 사용자로부터 작업의 데이터를 보호하는 데 도움이 됩니다. 대기열에 사용자가 없는 경우 모든 대기열 사용자를 가장(sudo)할 수 있는 에이전트 사용자로 실행됩니다. 이러한 방식으로 사용자가 없는 대기열은 권한을 다른 대기열로 에스컬레이션할 수 있습니다.

네트워킹

트래픽이 가로채거나 리디렉션되지 않도록 하려면 네트워크 트래픽이 라우팅되는 방식과 위치를 보호하는 것이 중요합니다.

다음과 같은 방법으로 네트워킹 환경을 보호하는 것이 좋습니다.

- Amazon Virtual Private Cloud(VPC) 서브넷 라우팅 테이블을 보호하여 IP 계층 트래픽이 라우팅되는 방식을 제어합니다.
- Amazon Route 53(Route 53)을 팜 또는 워크스테이션 설정에서 DNS 공급자로 사용하는 경우 Route 53 API에 대한 액세스를 보호하세요.
- 온프레미스 워크스테이션 또는 기타 데이터 센터를 사용하는 AWS 등 외부에서 Deadline Cloud에 연결하는 경우 온프레미스 네트워킹 인프라를 보호합니다. 여기에는 라우터, 스위치 및 기타 네트워킹 디바이스의 DNS 서버 및 라우팅 테이블이 포함됩니다.

작업 및 작업 데이터

Deadline Cloud 작업은 작업자 호스트의 세션 내에서 실행됩니다. 각 세션은 작업자 호스트에서 하나 이상의 프로세스를 실행하므로 일반적으로 출력을 생성하려면 데이터를 입력해야 합니다.

이 데이터를 보호하기 위해 대기열을 사용하여 운영 체제 사용자를 구성할 수 있습니다. 작업자 에이전트는 대기열 OS 사용자를 사용하여 세션 하위 프로세스를 실행합니다. 이러한 하위 프로세스는 대기열 OS 사용자의 권한을 상속합니다.

이러한 하위 프로세스가 액세스를 처리하는 데이터에 대한 액세스를 보호하려면 모범 사례를 따르는 것이 좋습니다. 자세한 내용은 공동 책임 모델을 참조하십시오.

팜 구조

다양한 방법으로 Deadline Cloud 플릿 및 대기열을 정렬할 수 있습니다. 그러나 특정 배열에는 보안에 영향을 미칩니다.

팜은 Deadline Cloud 리소스를 플릿, 대기열 및 스토리지 프로파일을 포함한 다른 팜과 공유할 수 없기 때문에 가장 안전한 경계 중 하나가 있습니다. 그러나 팜 내에서 외부 AWS 리소스를 공유할 수 있으므로 보안 경계가 손상됩니다.

적절한 구성을 사용하여 동일한 팜 내의 대기열 간에 보안 경계를 설정할 수도 있습니다.

다음 모범 사례에 따라 동일한 팜에서 보안 대기열을 생성합니다.

- 동일한 보안 경계 내의 대기열에만 플릿을 연결합니다. 다음 사항에 유의하세요.
  - 작업자 호스트에서 작업이 실행된 후 임시 디렉터리 또는 대기열 사용자의 홈 디렉터리와 같은 데이터가 뒤쳐질 수 있습니다.
  - 작업을 제출하는 대기열에 관계없이 동일한 OS 사용자가 서비스 소유 플릿 작업자 호스트에서 모든 작업을 실행합니다.
  - 작업은 작업자 호스트에서 실행 중인 프로세스를 그대로 둘 수 있으므로 다른 대기열의 작업이 실행 중인 다른 프로세스를 관찰할 수 있습니다.
- 동일한 보안 경계 내의 대기열만 작업 연결을 위해 Amazon S3 버킷을 공유하는지 확인합니다.
- 동일한 보안 경계 내의 대기열만 OS 사용자를 공유하는지 확인합니다.
- 팜에 통합된 다른 모든 AWS 리소스를 경계에 보호합니다.

작업 연결 대기열

작업 연결은 Amazon S3 버킷을 사용하는 대기열과 연결됩니다.

- 작업 연결은 Amazon S3 버킷의 루트 접두사에 쓰고 읽습니다. `CreateQueue` API 호출에서이 루트 접두사를 지정합니다.
- 버킷에는 대기열 사용자에게 버킷 및 루트 접두사에 대한 액세스 권한을 부여하는 역할을 `Queue Role`지정하는 해당이 있습니다. 대기열을 생성할 때 작업 연결 버킷 및 루트 접두사와 함께 `Queue Role` Amazon 리소스 이름(ARN)을 지정합니다.
- `AssumeQueueRoleForRead`, `AssumeQueueRoleForUser`및 `AssumeQueueRoleForWorker` API 작업에 대한 승인된 호출은에 대한 임시 보안 자격 증명 세트를 반환합니다`Queue Role`.

대기열을 생성하고 Amazon S3 버킷과 루트 접두사를 재사용하면 권한이 없는 당사자에게 정보가 공개될 위험이 있습니다. 예를 들어 QueueA와 QueueB는 동일한 버킷과 루트 접두사를 공유합니다. 보안 워크플로에서 ArtistA는 QueueA에 액세스할 수 있지만 QueueB에는 액세스할 수 없습니다. 그러나 여러 대기열이 버킷을 공유하는 경우 ArtistA는 QueueA와 동일한 버킷 및 루트 접두사를 사용하기 때문에 QueueB 데이터의 데이터에 액세스할 수 있습니다. QueueA

콘솔은 기본적으로 안전한 대기열을 설정합니다. 공통 보안 경계에 속하지 않는 한 대기열에 Amazon S3 버킷과 루트 접두사의 고유한 조합이 있는지 확인합니다.

대기열을 격리하려면 버킷 및 루트 접두사에 대한 대기열 액세스만 허용`Queue Role`하도록를 구성해야 합니다. 다음 예제에서는 각 *## ####* 리소스별 정보로 바꿉니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME",

 "arn:aws:s3:::JOB_ATTACHMENTS_BUCKET_NAME/JOB_ATTACHMENTS_ROOT_PREFIX/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "111122223333"
                }
            }
        },
        {
            "Action": [
                "logs:GetLogEvents"
            ],
            "Effect": "Allow",
```

```
            "Resource": "arn:aws:logs:us-east-1:111122223333:log-group:/aws/
deadline/FARM_ID/*"
        }
    ]
}
```

또한 역할에 대한 신뢰 정책을 설정해야 합니다. 다음 예제에서는 *## ###* 텍스트를 리소스별 정보로 바꿉니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "deadline.amazonaws.com"
            },
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        },
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": "credentials.deadline.amazonaws.com"
            },
            "Condition": {
```

```
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:deadline:us-
east-1:111122223333:farm/FARM_ID"
                }
            }
        }
    ]
}
```

사용자 지정 소프트웨어 Amazon S3 버킷

에 다음 문Queue Role을 추가하여 Amazon S3 버킷의 사용자 지정 소프트웨어에 액세스할 수 있습니다. 다음 예에서 *SOFTWARE_BUCKET_NAME*을 S3 버킷 이름으로 바꾸고 *BUCKET_ACCOUNT_OWNER*를 버킷을 소유한 AWS 계정 ID로 바꿉니다.

```
"Statement": [
    {
        "Action": [
            "s3:GetObject",
            "s3:ListBucket"
        ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME",
            "arn:aws:s3:::SOFTWARE_BUCKET_NAME/*"
        ],
        "Condition": {
         "StringEquals": {
            "aws:ResourceAccount": "BUCKET_ACCOUNT_OWNER"
         }
      }
    }
]
```

Amazon S3 보안 모범 사례에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 Amazon Amazon S3 보안 모범 사례를 참조하세요.

작업자 호스트

각 사용자가 할당된 역할에 대해서만 작업을 수행할 수 있도록 작업자 호스트를 보호합니다.

작업자 호스트를 보호하려면 다음 모범 사례를 따르는 것이 좋습니다.

- 호스트 구성 스크립트를 사용하면 작업자의 보안 및 작업이 변경될 수 있습니다. 구성이 잘못되면 작업자가 불안정해지거나 작동이 중지될 수 있습니다. 이러한 실패를 디버깅하는 것은 사용자의 책임입니다.
- 대기열에 제출된 작업이 동일한 보안 경계 내에 있지 않는 한 여러 대기열에 동일한 `jobRunAsUser` 값을 사용하지 마세요.
- 작업자 에이전트가 실행되는 OS 사용자의 `jobRunAsUser` 이름으로 대기열을 설정하지 마십시오.
- 대기열 사용자에게 의도한 대기열 워크로드에 필요한 최소 권한 OS 권한을 부여합니다. 에이전트 프로그램 파일 또는 기타 공유 소프트웨어를 작업할 수 있는 파일 시스템 쓰기 권한이 없는지 확인합니다.
- 의 루트 사용자Linux와의 `Administrator` 소유 계정만 Windows 소유하며 작업자 에이전트 프로그램 파일을 수정할 수 있는지 확인합니다.
- Linux 작업자 호스트에서는 작업자 에이전트 사용자가 대기열 사용자로 프로세스를 시작할 수 `/etc/sudoers` 있도록에서 `umask` 재정의를 구성하는 것이 좋습니다. 이 구성은 다른 사용자가 대기열에 기록된 파일에 액세스할 수 없도록 하는 데 도움이 됩니다.
- 신뢰할 수 있는 개인에게 작업자 호스트에 대한 최소 권한 액세스 권한을 부여합니다.
- 로컬 DNS 재정의 구성 파일(`/etc/hosts` Linux 및 Windows) 및 워크스테이션 및 작업자 호스트 운영 체제에서 테이블을 라우팅할 수 `C:\Windows\system32\etc\hosts` 있는 권한을 제한합니다.
- 워크스테이션 및 작업자 호스트 운영 체제의 DNS 구성에 대한 권한을 제한합니다.
- 운영 체제와 설치된 모든 소프트웨어를 정기적으로 패치합니다. 이 접근 방식에는 제출자, 어댑터, 작업자 에이전트, OpenJD 패키지 등과 같이 Deadline Cloud와 함께 특별히 사용되는 소프트웨어가 포함됩니다.
- Windows 대기열에 강력한 암호를 사용합니다`jobRunAsUser`.
- 대기열의 암호를 정기적으로 교체합니다`jobRunAsUser`.
- Windows 암호 보안 암호에 대한 최소 권한 액세스를 보장하고 사용하지 않는 보안 암호를 삭제합니다.
- 대기열에 향후 실행할 `jobRunAsUser` 일정 명령을 부여하지 마십시오.
  - 에서 `cron` 및에 대한 이러한 계정 액세스를 Linux거부합니다`at`.
  - 에서 Windows 작업 스케줄러에 대한 이러한 계정 액세스를 Windows거부합니다.

> **Note**
>
> 운영 체제 및 설치된 소프트웨어를 정기적으로 패치하는 것의 중요성에 대한 자세한 내용은 공동 책임 모델을 참조하세요.

호스트 구성 스크립트

- 호스트 구성 스크립트를 사용하면 작업자의 보안 및 작업이 변경될 수 있습니다. 구성이 잘못되면 작업자가 불안정해지거나 작동이 중지될 수 있습니다. 이러한 실패를 디버깅하는 것은 사용자의 책임입니다.

워크스테이션

Deadline Cloud에 액세스할 수 있는 워크스테이션을 보호하는 것이 중요합니다. 이 접근 방식은 Deadline Cloud에 제출하는 모든 작업이에 청구되는 임의 워크로드를 실행할 수 없도록 하는 데 도움이 됩니다 AWS 계정.

아티스트 워크스테이션을 보호하려면 다음 모범 사례를 따르는 것이 좋습니다. 자세한 내용은 공동 책임 모델을 참조하세요.

- Deadline Cloud를 AWS포함하여 액세스를 제공하는 지속적인 자격 증명을 보호합니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.
- 신뢰할 수 있는 보안 소프트웨어만 설치합니다.
- 사용자가 자격 증명 공급자와 연동하여 임시 자격 증명 AWS 으로에 액세스하도록 요구합니다.
- Deadline Cloud 제출자 프로그램 파일에 대한 보안 권한을 사용하여 변조를 방지합니다.
- 신뢰할 수 있는 개인에게 아티스트 워크스테이션에 대한 최소 권한을 부여합니다.
- Deadline Cloud Monitor를 통해 얻은 제출자 및 어댑터만 사용합니다.
- 로컬 DNS 재정의 구성 파일(`/etc/hosts` Linux 및 macOS, Windows) 및 워크스테이션 및 작업자 호스트 운영 체제에서 테이블을 라우팅할 수 `C:\Windows\system32\etc\hosts` 있는 권한을 제한합니다.
- 워크스테이션 및 작업자 호스트 운영 체제`/etc/resolve.conf`에 대한 권한을 로 제한합니다.
- 운영 체제와 설치된 모든 소프트웨어를 정기적으로 패치합니다. 이 접근 방식에는 제출자, 어댑터, 작업자 에이전트, OpenJD 패키지 등과 같이 Deadline Cloud와 함께 특별히 사용되는 소프트웨어가 포함됩니다.

다운로드한 소프트웨어의 신뢰성 확인

파일 변조를 방지하기 위해 설치 관리자를 다운로드한 후 소프트웨어의 신뢰성을 확인합니다. 이 절차는 Windows 및 Linux 시스템 모두에서 작동합니다.

Windows

다운로드한 파일의 신뢰성을 확인하려면 다음 단계를 완료하세요.

1. 다음 명령에서를 확인하려는 파일로 *file* 바꿉니다. 예를 들어 ***C:\PATH\TO\MY\DeadlineCloudSubmitter-windows-x64-installer.exe*** 입니다. 또한 *signtool-sdk-version*를 설치된 SignTool SDK 버전으로 바꿉니다. 예를 들어 **10.0.22000.0**입니다.

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin\signtool-sdk-version\x86\signtool.exe" verify /vfile
   ```

2. 예를 들어 다음 명령을 실행하여 Deadline Cloud 제출자 설치 관리자 파일을 확인할 수 있습니다.

   ```
   "C:\Program Files (x86)\Windows Kits\10\bin
   \10.0.22000.0\x86\signtool.exe" verify /v DeadlineCloudSubmitter-
   windows-x64-installer.exe
   ```

Linux

다운로드한 파일의 신뢰성을 확인하려면 gpg 명령줄 도구를 사용합니다.

1. 다음 명령을 실행하여 OpenPGP 키를 가져옵니다.

   ```
    gpg --import --armor <<EOF
   -----BEGIN PGP PUBLIC KEY BLOCK-----

   mQINBGlANDUBEACg6zffjN43gqe5ryPhk+wQM10rEdvmItw4WPWaVsN+/at/OIJw
   MGCagSYXcgR+jKbsHQOQoEQdo5SrxxHjpKTEs3KQhGvf+ehrU1Ac7koXKIBWtes+
   BI9F0slRECz0nXTOy/cd/90RXjpF07mreTLIKNIbybULfad82nYykpITjFr5XRGj
   /shYkucxRQZdwkgkIYyV25pPICPd2RsX+Zua85jV8mCqVffDfRXvgcPe3+ofClj/
   2CE8UfUIqO8Csua4YEkSqr3aeoTOEFT4kuQR5nFXVzorOEkQtO3gB35KNWKMlIOU
   2vA+wyoL7nWSii4yfYtW3EZ+3gq6HxvnT9Zs8MC53uTOiOdamASXecYREwGmY/io
   6n5XTEA/35LNbl4A756vSTZ7h4VFJAN5BpuqxstI1D7ou94skoSmcPoC/iniTvY9
   kZylU5OCH/nifMAHM2a5jrQel80cW4oko9eyc8ENQpSy15JElFOKFf7D/4tcZJLF
   F0VBTXbhfvq3dPfoq94IWt7p54Ovwj0S//CEu3jZYbNl2QC/3YiHE2H2XyGCQbq6
   ```

```
2MjcuxLnEapoRIqfbi8GPtCWVPzm28WGyKIDofWICczzeJFFJnvzrY3wRG64ibKJ
bR/uedwua1UuiC482V1FD5ffmzSSs8ktTp9hgj7RGDXlc9NTcF1jHxG9hwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBJmXd7So2csyehiIYsg71N18bhtjBQJpQDQ1AhsvBQkDwmcABQsJ
CAcCAiICBhUKCQgLAgQWAgMBAh4HAheAAAoJEMg71N18bhtjk2UP/3h4KlEzZO/7
BxRmkbixuo1QuqOGvA6tXbSWaM8QH5jglcvL12PZLALklLT4v82uCsLR1lF8/Tch
cCl0SZEOFIS+XxAaw1Xfai6jlyLhabOwKF2ylq5eJlLcw1lh2nAArDRb4fLD0m1g
Dfqetq/XEpyXpOSkWxGRV4RlUdjQfytxrmcUnsT5/fk5f9VDdblu6K/lEmwfyYjB
lXv0uUCkqPot0SmbvOh3PY3Hi3n54ncy8NfTeV+TUvSe3C1s1zNl8aqHoTxJB/eU
kp+LFZ9m+igpSYnKeglKnytylH3KGCjTHglT/QXnI1wNTqmj1kFBVwtt/y1mtnA+
CPIUHP1CtbKsHaLtpp4llBm5TVtPN/Wqqicn5QLl4khg7R4K+V2aaA4ubY6p1tG9
0fFhN5tTnHDSKWMfmb83wfh5Zkcg85c3egjoit+wgGQRAQVqbznx7NqAHs9VoDIu
SPcAr+C329AOBzod4gyNGH7Ah5DkMITo4O4+axnAU9yhFOHcMJmTIask/fNg1Aum
OqYPMUwcgv1GZjLaTJyfGGC1xALsYR0KHnwIehD06MHR/Z98bGkcV8+Y0q8UPsd1
VN1fc1rjCJh/AT3w6owvG4DaEwspseSjzHv16mW4e2N6Uu23SPzgQsJ5qYN2g8D+
P7N9LGDfP8DaYc5JM9mlyFmYI2Q94ufL
=rY5l
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

2. OpenPGP 키를 신뢰할지 여부를 결정합니다. 위 키를 신뢰할지 여부를 결정할 때 고려해야 할 몇 가지 요소는 다음과 같습니다.

   - 이 웹 사이트에서 GPG 키를 가져오는 데 사용한 인터넷 연결은 안전합니다.
   - 이 웹 사이트에 액세스하는 디바이스는 안전합니다.
   - AWS 는이 웹 사이트에서 OpenPGP 퍼블릭 키의 호스팅을 보호하기 위한 조치를 취했습니다.

3. OpenPGP 키를 신뢰하기로 결정한 경우 다음 예와 gpg 마찬가지로에서 신뢰하도록 키를 편집합니다.

```
$ gpg --edit-key 0xB840C08C29A90796A071FAA5F6CD3CE6B76F3CEF

    gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.
    This is free software: you are free to change and redistribute it.
    There is NO WARRANTY, to the extent permitted by law.


    pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                         trust: unknown       validity: unknown
    [ unknown] (1). AWS Deadline Cloud example@example.com

    gpg> trust
```

```
   pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                        trust: unknown       validity: unknown
   [ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com

   Please decide how far you trust this user to correctly verify other users'
keys
   (by looking at passports, checking fingerprints from different sources,
etc.)

     1 = I don't know or won't say
     2 = I do NOT trust
     3 = I trust marginally
     4 = I trust fully
     5 = I trust ultimately
     m = back to the main menu

   Your decision? 5
   Do you really want to set this key to ultimate trust? (y/N) y

   pub  4096R/4BF0B8D2  created: 2023-06-23  expires: 2025-06-22  usage: SCEA
                        trust: ultimate      validity: unknown
   [ unknown] (1). AWS Deadline Cloud aws-deadline@amazon.com
   Please note that the shown key validity is not necessarily correct
   unless you restart the program.

   gpg> quit
```

4. Deadline Cloud 제출자 설치 프로그램 확인

   Deadline Cloud 제출자 설치 프로그램을 확인하려면 다음 단계를 완료하세요.

   a. Deadline Cloud 제출자 설치 프로그램의 서명 파일을 다운로드합니다.

      서명 파일 다운로드(.sig)

   b. 다음을 실행하여 Deadline Cloud 제출자 설치 프로그램의 서명을 확인합니다.

      ```
      gpg --verify ./DeadlineCloudSubmitter-linux-x64-installer.run.sig ./
      DeadlineCloudSubmitter-linux-x64-installer.run
      ```

5. Deadline Cloud 모니터 확인

> **Note**
>
> 서명 파일 또는 플랫폼별 방법을 사용하여 Deadline Cloud Monitor 다운로드를 확인할 수 있습니다. 플랫폼별 방법은 다운로드한 파일 유형에 따라 Linux (Debian) 탭, Linux (RPM) 탭 또는 Linux (AppImage) 탭을 참조하세요.

서명 파일을 사용하여 Deadline Cloud Monitor 데스크톱 애플리케이션을 확인하려면 다음 단계를 완료하세요.

a. Deadline Cloud Monitor 설치 관리자에 해당하는 서명 파일을 다운로드합니다.

- .deb 서명 파일 다운로드
- .rpm 서명 파일 다운로드
- .AppImage 서명 파일 다운로드

b. 서명을 확인합니다.

.deb의 경우:

```
gpg --verify ./deadline-cloud-monitor_amd64.deb.sig ./deadline-cloud-
monitor_amd64.deb
```

.rpm의 경우:

```
gpg --verify ./deadline-cloud-monitor.x86_64.rpm.sig ./deadline-cloud-
monitor.x86_64.rpm
```

.AppImage의 경우:

```
gpg --verify ./deadline-cloud-monitor_amd64.AppImage.sig ./deadline-cloud-
monitor_amd64.AppImage
```

c. 출력이 다음과 비슷한지 확인합니다.

```
gpg: Signature made Mon Apr 1 21:10:14 2024 UTC

gpg: using RSA key B840C08C29A90796A071FAA5F6CD3CF6B7
```

출력에 문구가 포함된 경우 서명이 성공적으로 확인되었으며 Deadline Cloud Monitor 설치 스크립트를 실행할 수 있음을 `Good signature from "AWS Deadline Cloud"`의 미합니다.

기록 키

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGX6GQsBEADduUtJgqSXI+q76O6fsFwEYKmbnlyL0xKvlq32EZuyv0otZo5L
le4m5Gg52AzrvPvDiUTLooAlvYeozaYyirIGsK08Ydz0Ftdjroiuh/mw9JSJDJRI
rnRn5yKet1JFezkjopA3pjsTBP6lW/mb1bDBDEwwwtH0x9lV7A03FJ9T7Uzu/qSh
qO/UYdkafro3cPASvkqgDt2tCvURfBcUCAjZVFcLZcVD5iwXacxvKsxxS/e7kuVV
I1+VGT8Hj8XzWYhjCZxOLZk/fvpYPMyEEujN0fYUp6RtMIXve0C9awwMCy5nBG2J
eE2Ol5DsCpTaBd4Fdr3LWcSs8JFA/YfP9auL3NczOozPoVJt+fw8CBlVIXO0J7l5
hvHDjcC+5v0wxqAlMG6+f/SX7CT8FXK+L3iOJ5gBYUNXqHSxUdv8kt76/KVmQa1B
Akl+MPKpMq+lhw++S3G/lXqwWaDNQbRRw7dSZHymQVXvPp1nsqc3hV7KlOM+6s6g
1g4mvFY4lf6DhptwZLWyQXU8rBQpojvQfiSmDFrFPWFi5BexesuVnkGIolQoklKx
AVUSdJPVEJCteyy7td4FPhBaSqT5vW3+ANbr9b/uoRYWJvn17dN0cc9HuRh/Ai+I
nkfECo2WUDLZ0fEKGjGyFX+todWvJXjvc5kmE9Ty5vJp+M9Vvb8jd6t+mwARAQAB
tCxBV1MgRGVhZGxpbmUgQ2xvdWQgPGF3cy1kZWFkbGluZUBhbWF6b24uY29tPokC
VwQTAQgAQRYhBLhAwIwpqQeWoHH6pfbNPOa3bzzvBQJl+hkLAxsvBAUJA8JnAAUL
CQgHAgIiAgYVCgkICwIDFgIBAh4HAheAAAoJEPbNPOa3bzzvKswQAJXzKSAY8sY8
F6Eas2oYwIDDdDurs8FiEnFghjUEO6MTt9AykF/jw+CQg2UzFtEyObHBymhgmhXE
3buVeom96tgM3ZDfZu+sxi5pGX6oAQnZ6riztN+VpkpQmLgwtMGpSML13KLwnv2k
WK8mrR/fPMkfdaewB7A6RIUYiW33GAL4KfMIs8/vIwIJw99NxHpZQVoU6dFpuDtE
1OuxGcCqGJ7mAmo6H/YawSNp2Ns80gyqIKYo7o3LJ+WRroIRlQyctq8gnR9JvYXX
42ASqLq5+OXKo4qh81blXKYqtc176BbbSNFjWnzIQgKDgNiHFZCdcOVgqDhwO15r
NICbqqwwNLj/Fr2kecYx180Ktpl0jOOw5IOyh3bf3MVGWnYRdjvA1v+/CO+55N4g
z0kf50Lcdu5RtqV10XBCifn28pecqPaSdYcssYSRl5DLiFktGbNzTGcZZwITTKQc
af8PPdTGtnnb6P+cdbW3bt9MVtN5/dgSHLThnS8MPEuNCtkTnpXshuVuBGgwBMdb
qUC+HjqvhZzbwns8dr5WI+6HWNBFgGANn6ageYl58vVp0UkuNP8wcWjRARciHXZx
ku6W2jPTHDWGNrBQO2Fx7fd2QYJheIPPAShHcfJO+xgWCof45D0vAxAJ8gGg9Eq+
gFWhsx4NSHn2gh1gDZ41Ou/4exJ1lwPM
=uVaX
-----END PGP PUBLIC KEY BLOCK-----
EOF
```

Linux (AppImage)

Linux .AppImage 바이너리를 사용하는 패키지를 확인하려면 먼저 Linux 탭에서 1~3단계를 완료한 다음 다음 단계를 완료합니다.

1. GitHub의 AppImageUpdate 페이지에서 validate-x86_64.AppImage 파일을 다운로드합니다.
2. 파일을 다운로드한 후 실행 권한을 추가하려면 다음 명령을 실행합니다.

```
chmod a+x ./validate-x86_64.AppImage
```

3. 실행 권한을 추가하려면 다음 명령을 실행합니다.

```
chmod a+x ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
```

4. Deadline Cloud Monitor 서명을 확인하려면 다음 명령을 실행합니다.

```
./validate-x86_64.AppImage ./deadline-cloud-monitor_<APP_VERSION>_amd64.AppImage
```

출력에 문구가 포함된 경우 서명이 성공적으로 확인되었으며 Deadline Cloud Monitor 설치 스크립트를 안전하게 실행할 수 있음을 `Validation successful`의미합니다.

Linux (Debian)

Linux .deb 바이너리를 사용하는 패키지를 확인하려면 먼저 Linux 탭에서 1~3단계를 완료합니다.

dpkg은 대부분의 debian 기반 Linux 배포에서 핵심 패키지 관리 도구입니다. 도구를 사용하여 .deb 파일을 확인할 수 있습니다.

1. Deadline Cloud Monitor .deb 파일을 다운로드합니다.

   Deadline Cloud Monitor 다운로드(.deb)

2. .deb 파일을 확인합니다.

```
dpkg-sig --verify deadline-cloud-monitor_amd64.deb
```

3. 출력은 다음과 유사합니다.

```
Processing deadline-cloud-monitor_amd64.deb...
GOODSIG _gpgbuilder B840C08C29A90796A071FAA5F6CD3C 171200
```

4. .deb 파일을 확인하려면 GOODSIG가 출력에 있는지 확인합니다.

Linux (RPM)

Linux .rpm 바이너리를 사용하는 패키지를 확인하려면 먼저 Linux 탭에서 1~3단계를 완료합니다.

1. Deadline Cloud Monitor .rpm 파일을 다운로드합니다.

   Deadline Cloud Monitor 다운로드(.rpm)

2. .rpm 파일을 확인합니다.

```
gpg --export --armor "Deadline Cloud" > key.pub
sudo rpm --import key.pub
rpm -K deadline-cloud-monitor.x86_64.rpm
```

3. 출력은 다음과 유사합니다.

```
deadline-cloud-monitor.x86_64.rpm: digests signatures OK
```

4. .rpm 파일을 확인하려면 digests signatures OK가 출력에 있는지 확인합니다.

문서 이력

AWSDeadline Cloud 업데이트에 대한 자세한 내용은 Deadline Cloud 릴리스 정보를 참조하세요.

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