사용자 가이드

AWS 메인프레임 현대화

AWS 메인프레임 현대화: 사용자 가이드

Table of Contents

메인프레임 AWS 현대화란?

AWS 메인프레임 현대화는 메인프레임 애플리케이션을 관리형 런타임 환경으로 현대화하는 데 도움이 됩니다. AWS 마이그레이션과 현대화를 계획하고 구현하는 데 도움이 되는 도구와 리소스를 제공합니다. 기존 Mainframe 애플리케이션을 분석하고, COBOL 또는 PL/I를 사용하여 개발 또는 업데이트하고, 애플리케이션의 지속적 통합 및 지속적 전달(CI/CD)을 위한 자동화된 파이프라인을 구현할 수 있습니다. 고객의 요구에 따라 자동 리팩토링과 리플랫포밍 패턴 중에서 선택할 수 있습니다. 고객이 메인프레임 워크로드를 마이그레이션하도록 지원하는 컨설턴트인 경우 초기 계획부터 마이그레이션 후 클라우드 운영에 이르기까지 마이그레이션 및 현대화 여정의 모든 단계에서 AWS 메인프레임 현대화 도구를 사용할 수 있습니다.

메인프레임 현대화를 사용하면 AWS 메인프레임 애플리케이션의 런타임 환경을 효율적으로 생성 및 관리하고 현대화된 애플리케이션을 관리 및 AWS 모니터링할 수 있습니다.

주제

> **Note**
>
> 메인프레임 현대화 프로젝트를 위해 AWS 메인프레임 마이그레이션 컴피턴시 파트너 또는 전문 서비스에 문의한 적이 있습니까? AWS 그렇지 않은 경우 프로젝트에 전문가를 참여시키는 것이 좋습니다.
>
> - AWS 메인프레임 현대화 컴피턴시 파트너
> - AWS Professional Services

AWS 메인프레임 현대화의 기능 및 사용 사례는 혁신적인 현대화 접근 방식을 지원합니다. 이 접근 방식은 민첩성을 개선하여 단기적으로 이익을 얻고 나중에 최적화하고 혁신할 수 있는 많은 기회를 제공합니다. 자세한 설명은 현대화 접근법 섹션을 참조하세요.

메인프레임 현대화의 특징 AWS

AWS 메인프레임 현대화 기능은 다음 사용 사례를 지원합니다.

- 평가: AWS 메인프레임 현대화의 평가 기능은 마이그레이션 및 현대화 프로젝트를 평가, 범위 지정 및 계획하는 데 도움이 될 수 있습니다.
- 리팩터링: AWS Blu Age를 기반으로 하는 리팩터링을 사용하여 기존 애플리케이션 프로그래밍 언어를 변환하고, 매크로 서비스 또는 마이크로서비스를 생성하고, 사용자 인터페이스 (UI) 및 애플리케이션 소프트웨어 스택을 현대화할 수 있습니다.

  AWS 이제 싱글 사인온을 통해 Blu Insights를 이용할 수 있습니다. AWS Management Console 더 이상 AWS Blu Insights 자격 증명을 별도로 관리할 필요가 없습니다. 에서 직접 AWS AWS Blu Age 코드베이스와 변환 센터 기능에 모두 액세스할 수 있습니다. AWS Management Console
- 리플랫포밍: Micro Focus Enterprise 솔루션을 통해 대부분의 애플리케이션 소스 코드가 변경 없이 다시 컴파일되는 애플리케이션을 이식할 수 있습니다.
- 개발자 IDE: AWS 메인프레임 현대화는 온디맨드 통합 개발 환경 (IDE) 을 제공하므로 개발자는 스마트 편집 및 디버깅, 즉각적인 코드 컴파일, 유닛 테스트를 통해 코드를 더 빠르게 작성할 수 있습니다.
- 관리형 런타임: AWS 메인프레임 현대화 관리형 실행 환경은 클러스터를 지속적으로 모니터링하여 자가 복구 컴퓨팅과 자동화된 크기 조정을 통해 엔터프라이즈 워크로드가 계속 실행되도록 합니다.
- 지속적 통합 및 전달 (CI/CD): AWS 메인프레임 현대화의 CI/CD 기능은 애플리케이션 개발팀이 코드 변경을 더 자주, 안정적으로 제공할 수 있도록 지원하여 마이그레이션 속도를 높이고 품질을 높이며 새로운 비즈니스 기능의 출시 시간을 줄이는 데 도움이 됩니다. time-to-market
- 다른 AWS 서비스와의 통합: AWS 메인프레임 현대화 지원, 반복 가능한 배포, 보안 및 규정 준수 강화 AWS CloudFormation AWS PrivateLink AWS Key Management Service
- 가용성 확대: 이제 미국 동부 (오하이오), 미국 서부 (캘리포니아 북부), 아시아 태평양 (뭄바이), 아시아 태평양 (서울), 아시아 태평양 (싱가포르), 아시아 태평양 (도쿄), 유럽 (런던), 유럽 (파리) 에서 AWS 메인프레임 현대화를 이용할 수 있습니다.

메인프레임 현대화 기능에 대한 자세한 내용은 을 참조하십시오. AWS https://aws.amazon.com/mainframe-modernization/features/

패턴

AWS Blu Age에서 제공하는 자동 리팩토링 패턴은 기능적 동등성을 유지하면서 전체 레거시 애플리케이션 스택과 해당 데이터 계층을 최신 Java 기반 애플리케이션으로 변환하여 현대화를 가속화하는 데 중점을 둡니다. 이 자동 변환 과정에서 Angular 기반 프런트 엔드, API 지원 Java 백엔드 및 최신 데이터 저장소에 액세스하는 데이터 계층을 포함하는 다중 계층 애플리케이션을 생성합니다. 리팩토링 프로세스는 레거시 스택과 동일한 기능을 제공하여 프로젝트 자동화를 향상시켜 속도, 품질 및 비용 절감을 통해 비즈니스 이점을 더 빨리 달성합니다. 자세한 내용은 AWS Mainframe Modernization 자동 리팩터링을 참조하세요.

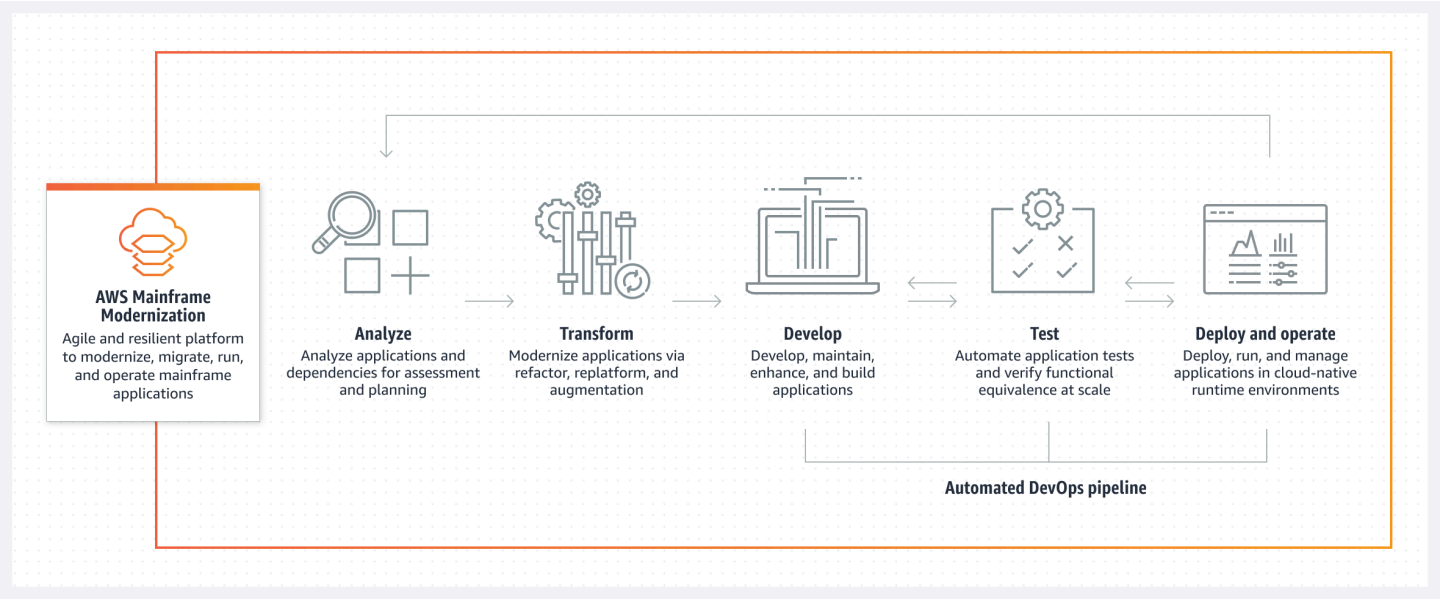
Micro Focus Enterprise Suite에서 제공하는 리플랫포밍 패턴은 애플리케이션 자산 및 팀에 미치는 영향을 최소화하기 위해 애플리케이션 언어, 코드 및 아티팩트를 보존하는 데 중점을 둡니다. 이를 통해 고객은 애플리케이션 지식과 기술을 유지할 수 있습니다. 애플리케이션 변경은 제한적이지만 이 패턴은 인프라 및 프로세스의 현대화도 촉진합니다. 인프라는 최신 클라우드 기반 관리 서비스로 변경되고 프로세스는 애플리케이션 개발 및 IT 운영의 모범 사례를 따르도록 변경됩니다. 자세한 내용은 AWS Mainframe Modernization 리플랫포밍을 참조하세요.

메인프레임 현대화를 시작하는 방법 AWS

사용해보기! 메인프레임 현대화가 제공하는 기능을 이해하는 데 도움이 되는 자습서와 샘플 애플리케이션을 제공합니다. AWS 전체 자습서를 보려면 둘 중 하나 자습서: AWS Blu Age를 위한 관리형 런타임 또는 자습서: Micro Focus의 관리형 런타임 를 선택하십시오. step-by-step

자동 리팩토링에 관심이 있다면 에서 AWS Blu Age 도구를 확인해 보세요. BluInsights AppStream 2.0을 설정하여 AWS 블루에이지 개발자 IDE 또는 마이크로 포커스 엔터프라이즈 분석기 및 마이크로 포커스 엔터프라이즈 개발자 도구에 액세스할 수도 있습니다.

튜토리얼과 샘플 애플리케이션을 통해서만 AWS 메인프레임 현대화가 제공하는 기능을 이해할 수 있습니다. 현대화 프로젝트를 시작할 준비가 되면 현대화 프로젝트의 단계 및 작업에 현대화 접근법 대한 세부 정보를 위해 를 참조하세요.



관련 서비스

자동 리팩토링을 위한 Blu Insights 외에도 메인프레임 현대화와 함께 다음 서비스를 사용할 수 있습니다. AWS AWS

- 마이그레이션된 데이터베이스를 호스팅하기 위한 Amazon RDS.
- 애플리케이션 바이너리와 정의 파일을 저장하기 위한 Amazon S3.
- 애플리케이션 데이터를 저장하기 위한 Amazon FSx 또는 Amazon EFS.
- Amazon에서 AppStream 마이크로 포커스 엔터프라이즈 애널라이저 및 마이크로 포커스 엔터프라이즈 개발자 도구를 이용할 수 있습니다.
- AWS CloudFormation 마이그레이션된 애플리케이션을 위한 CI/CD를 설정하는 데 사용할 수 있는 자동화된 DevOps 파이프라인용.
- AWS Migration Hub
- AWS DMS 데이터베이스 마이그레이션용.

AWS 메인프레임 현대화에 액세스

현재는 https://console.aws.amazon.com/m2/ 콘솔을 통해 AWS 메인프레임 현대화에 액세스할 수 있습니다. 메인프레임 현대화가 가능한 지역 목록은 의 AWS 메인프레임 현대화 엔드포인트 및 할당량을 참조하십시오AWS . Amazon Web Services 일반 참조

메인프레임 현대화를 처음 사용하십니까? AWS

AWS 메인프레임 현대화를 처음 사용하는 경우 먼저 다음 섹션을 읽는 것이 좋습니다.

- 시작하기
- 설정

요금

AWS 메인프레임 현대화는 관리형 런타임 환경을 지원하는 인스턴스 사용에 대한 요금을 부과합니다. 또한 AWS 메인프레임 현대화는 추가 비용 없이 일부 도구를 제공합니다. 메인프레임 현대화와 관련하여 사용하는 다른 AWS 서비스에 대해 발생하는 요금은 사용자가 부담해야 합니다. AWS AWS 메인프레임 현대화 사용에 대한 가격 변경 사항이 적용되기 30일 전에 통지합니다. AWS 자세한 내용은 메인프레임 현대화를 통한 메인프레임 현대화를 참조하십시오. AWS

AWS Blu Insights를 사용하면 변환 센터 사용에 대한 비용을 지불합니다. 자세한 내용은 AWS Mainframe Modernization 요금을 참조하세요.

AWS 메인프레임 현대화 설정

AWS 메인프레임 현대화를 사용하기 전에 사용자 또는 관리자가 몇 가지 단계를 완료해야 합니다.

주제

가입하여 AWS 계정

계정이 없는 경우 다음 단계를 완료하여 계정을 만드세요. AWS 계정

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

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

   에 AWS 계정가입하면 AWS 계정 루트 사용자a가 생성됩니다. 루트 사용자는 계정의 모든 AWS 서비스 및 리소스에 액세스하는 권한이 있습니다. 보안 모범 사례는 관리 사용자에게 관리자 액세스 권한을 할당하고, 루트 사용자만 루트 사용자 액세스 권한이 필요한 태스크를 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 사용자 생성

등록한 AWS 계정후에는 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 보호하고 AWS IAM Identity Center활성화하고 생성하십시오 AWS 계정 루트 사용자.

보안을 유지하세요 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 AWS Management Console소유자로 로그인합니다. 다음 페이지에서 암호를 입력합니다.

루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.

2. 루트 사용자에 대해 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자 (콘솔) 에 대한 가상 MFA 디바이스 활성화를 참조하십시오.

관리 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.

2. IAM Identity Center에서 관리 사용자에게 관리 액세스 권한을 부여합니다.

   를 ID 소스로 사용하는 방법에 대한 자습서는 사용 설명서의 기본값으로 IAM Identity Center 디렉터리사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리 AWS IAM Identity Center

관리 사용자로 로그인

- IAM 자격 증명 센터 사용자로 로그인하려면 IAM 자격 증명 센터 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털에 로그인을 참조하십시오.AWS 로그인

AWS 메인프레임 현대화 시작하기

AWS 메인프레임 현대화를 시작하려면 서비스와 각 런타임 엔진을 소개하는 자습서를 따르십시오.

주제

- 자습서: AWS Blu Age를 위한 관리형 런타임
- 자습서: Micro Focus의 관리형 런타임

계속 학습하려면 다음 자습서를 참조하십시오.

- 자습서: BankDemo 샘플 애플리케이션을 위한 Micro Focus 빌드 설정
- 튜토리얼: Micro Focus Enterprise Developer와 함께 사용할 CI/CD 파이프라인 설정

자습서: AWS Blu Age를 위한 관리형 런타임

이 자습서에서는 AWS Blu Age 현대화 애플리케이션을 AWS Mainframe Modernization 런타임 환경에 배포하는 방법을 보여줍니다.

주제

- 필수 조건
- 1단계: 데모 애플리케이션 업로드
- 2단계: 애플리케이션 정의 만들기
- 3단계: 런타임 환경 생성
- 4단계: 애플리케이션 생성
- 5단계: 애플리케이션 배포
- 6단계: 애플리케이션 시작
- 7단계: 애플리케이션 액세스
- 8단계: 애플리케이션 테스트
- 리소스 정리

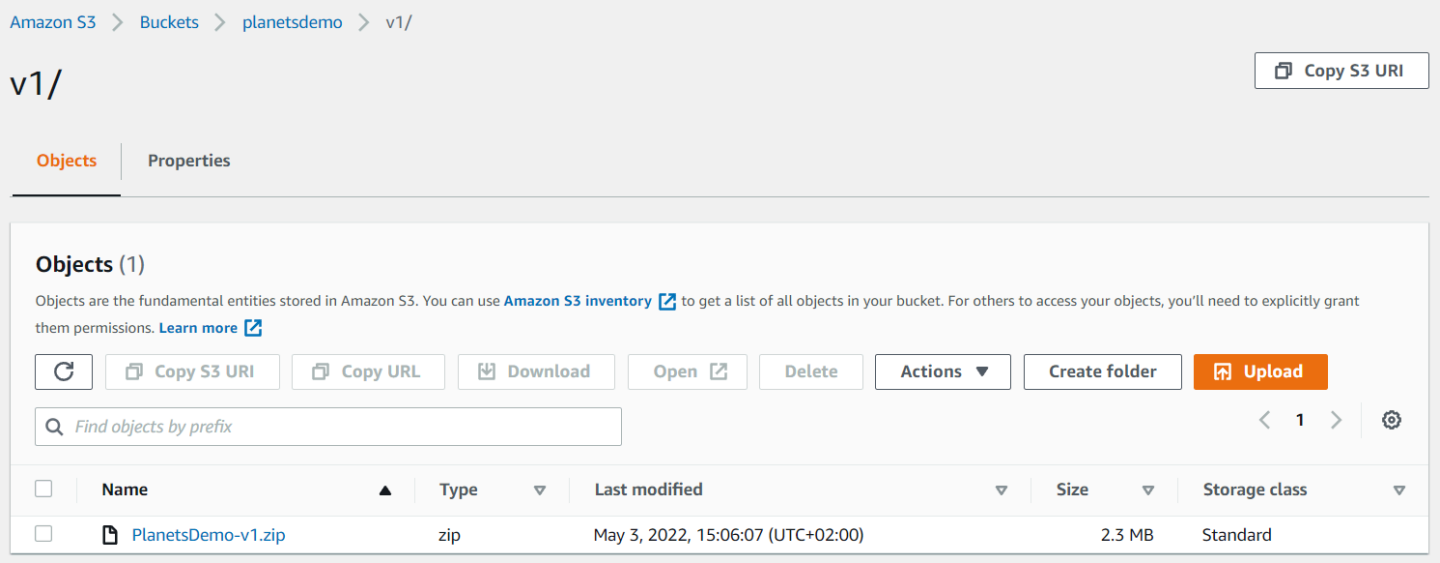
필수 조건

이 자습서를 완료하려면 데모 애플리케이션 아카이브 PlanetsDemo-v1.zip 를 다운로드하십시오.

실행 중인 데모 애플리케이션에 액세스하려면 최신 브라우저가 필요합니다. 이 브라우저를 데스크톱에서 실행할지, Amazon Elastic Compute Cloud 인스턴스(예: VPC 내)에서 실행하는지에 따라 보안 설정이 결정됩니다.

1단계: 데모 애플리케이션 업로드

Amazon S3 버킷에 애플리케이션 개정 버전을 업로드합니다. 애플리케이션을 배포할 AWS 리전 위치와 동일한 위치에 이 버킷이 있는지 확인하세요. 다음 예제는 planetsdemo라는 이름의 버킷을 보여 줍니다. 이 버킷에는 키 접두사 또는 폴더의 이름이 v1이고 이름이 `planetsdemo-v1.zip`인 아카이브가 있습니다.



> **Note**
>
> 버킷의 폴더는 필수입니다.

2단계: 애플리케이션 정의 만들기

애플리케이션을 관리형 런타임에 배포하려면 AWS Mainframe Modernization 애플리케이션 정의가 필요합니다. 이 정의는 애플리케이션 위치 및 설정을 설명하는 JSON 파일입니다. 다음 예제는 데모 애플리케이션에 대한 이러한 애플리케이션 정의입니다.

```
{
    "template-version": "2.0",
    "source-locations": [{
        "source-id": "s3-source",
```

```
        "source-type": "s3",
        "properties": {
            "s3-bucket": "planetsdemo",
            "s3-key-prefix": "v1"
        }
    }],
    "definition": {
        "listeners": [{
            "port": 8196,
            "type": "http"
        }],
        "ba-application": {
            "app-location": "${s3-source}/PlanetsDemo-v1.zip"
        }
    }
}
```

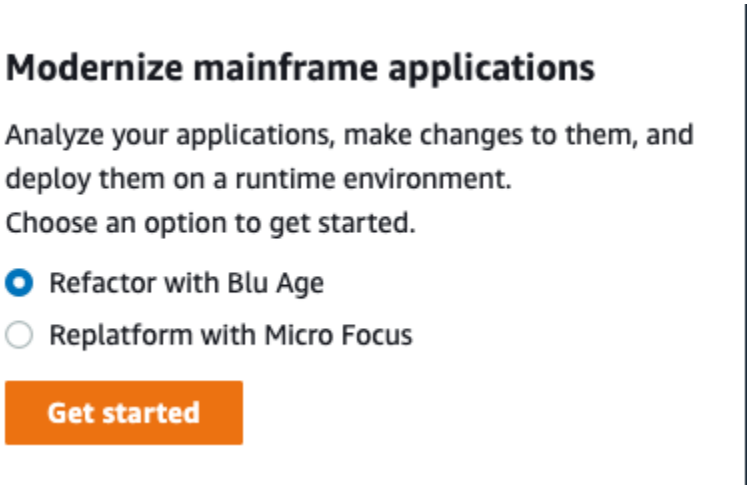
`s3-bucket` 항목을 샘플 애플리케이션 zip 파일을 저장한 버킷 이름으로 변경합니다.

애플리케이션 정의에 대한 자세한 내용은 AWS 블루 에이지 애플리케이션 정의 샘플를 참조하세요.

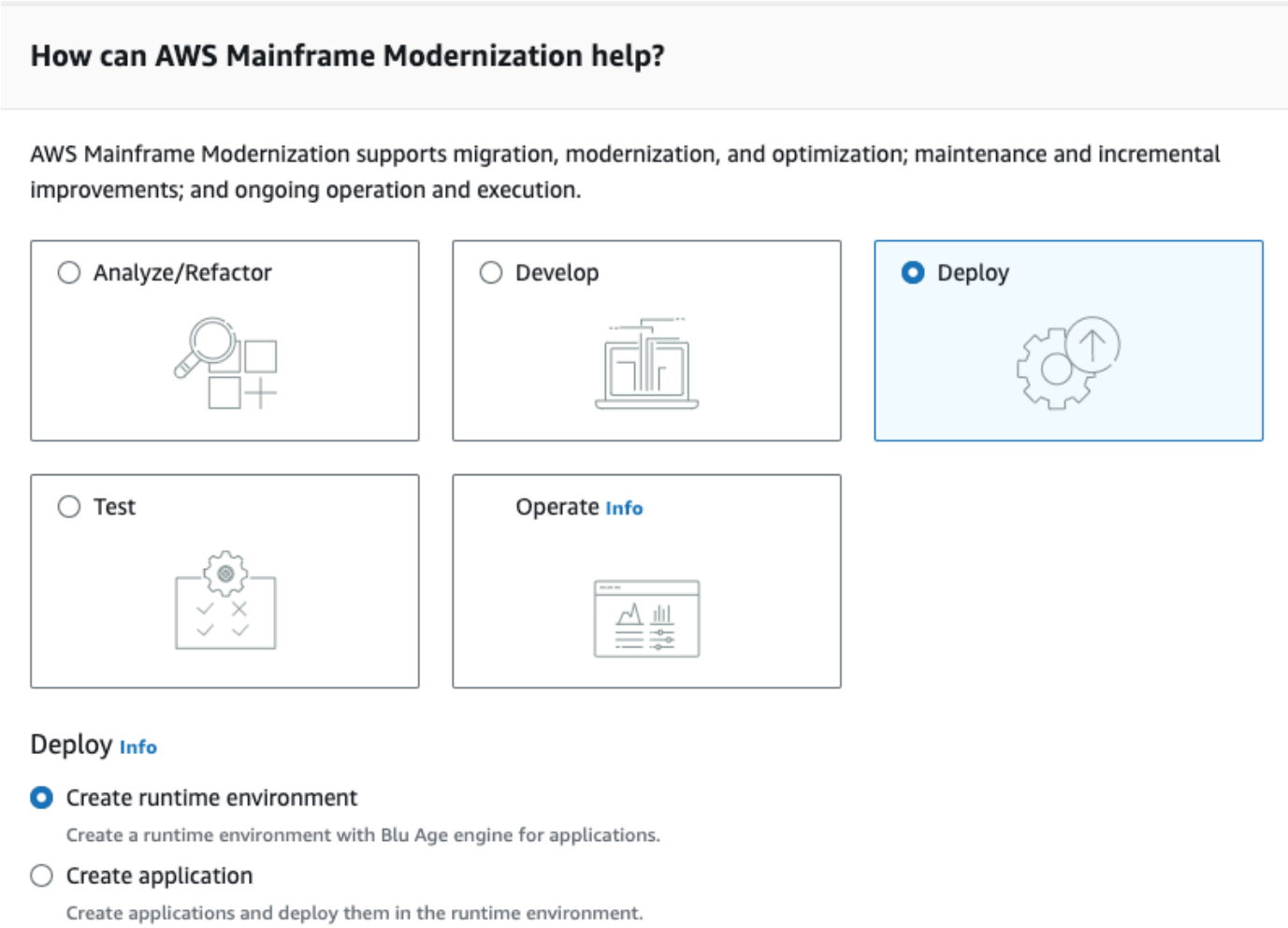
3단계: 런타임 환경 생성

AWS Mainframe Modernization 런타임 환경을 생성하려면 다음 단계를 수행합니다.

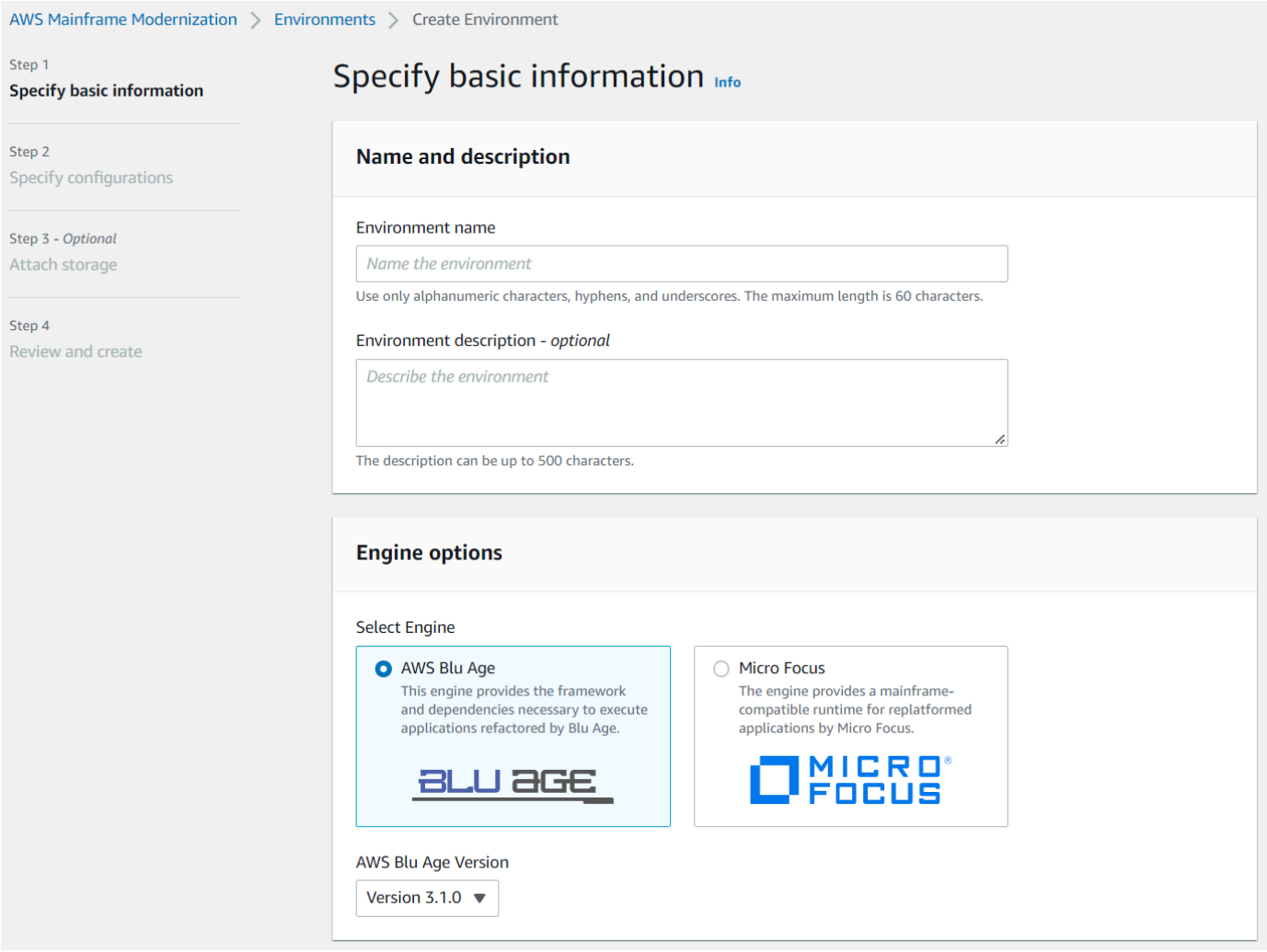
1. AWS Mainframe Modernization 콘솔을 엽니다.
2. AWS 리전 섹터에서 환경을 생성할 리전을 선택합니다. 1단계: 데모 애플리케이션 업로드에서 S3 버킷을 생성한 리전과 AWS 리전는 일치해야 합니다.
3. Mainframe 애플리케이션 현대화 애플리케이션에서 Blu-Age 기반 리팩터링을 선택한 다음 시작하기를 선택합니다.



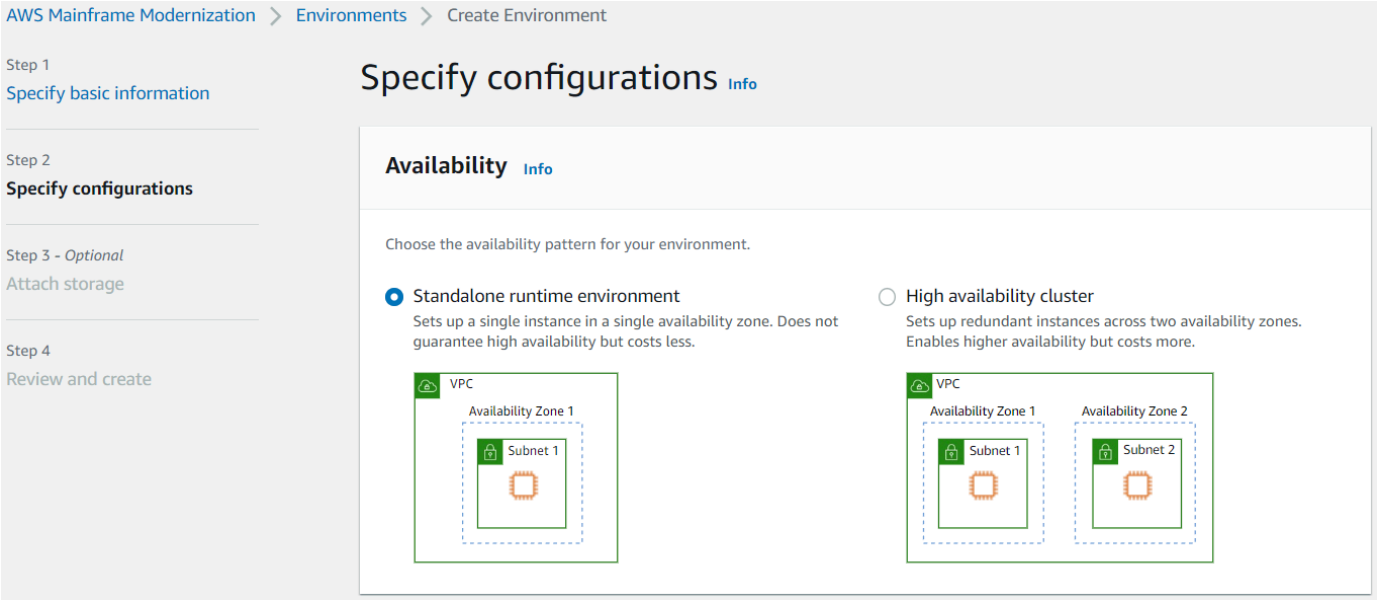
4. AWS Mainframe Modernization에 도움이 되는 방법에서 배포와 런타임 환경 생성을 선택합니다.



5. 왼쪽 탐색 창에서 컴퓨팅 환경을 선택한 다음 환경 생성을 선택합니다. 기본 정보 지정 페이지에서 환경의 이름과 설명을 입력한 다음 AWS Blu Age 엔진이 선택되었는지 확인합니다. 선택적으로 생성된 리소스에 태그를 추가할 수 있습니다. 다음을 선택합니다.

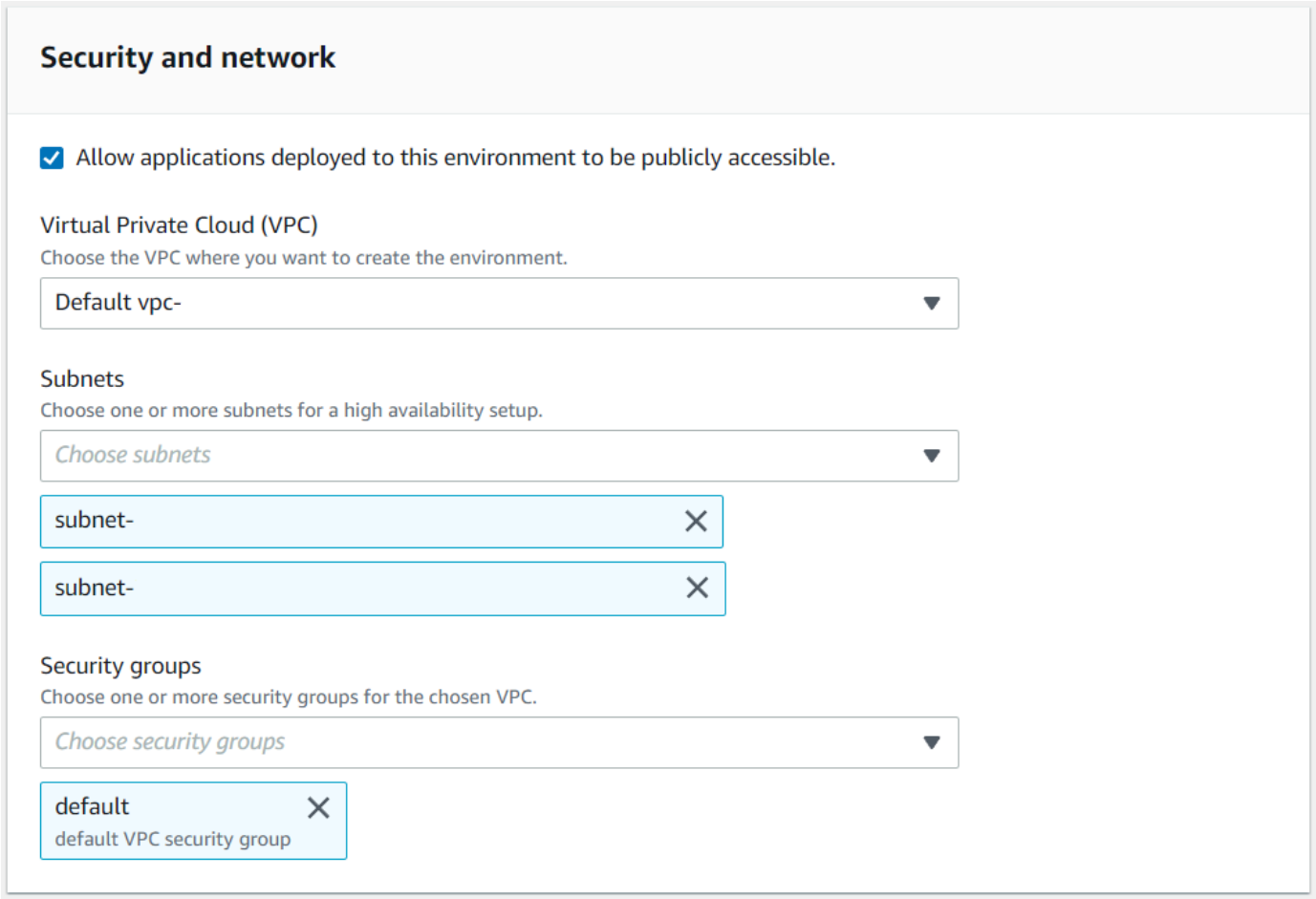


6. 구성 지정 페이지에서 독립형 런타임 환경을 선택합니다.



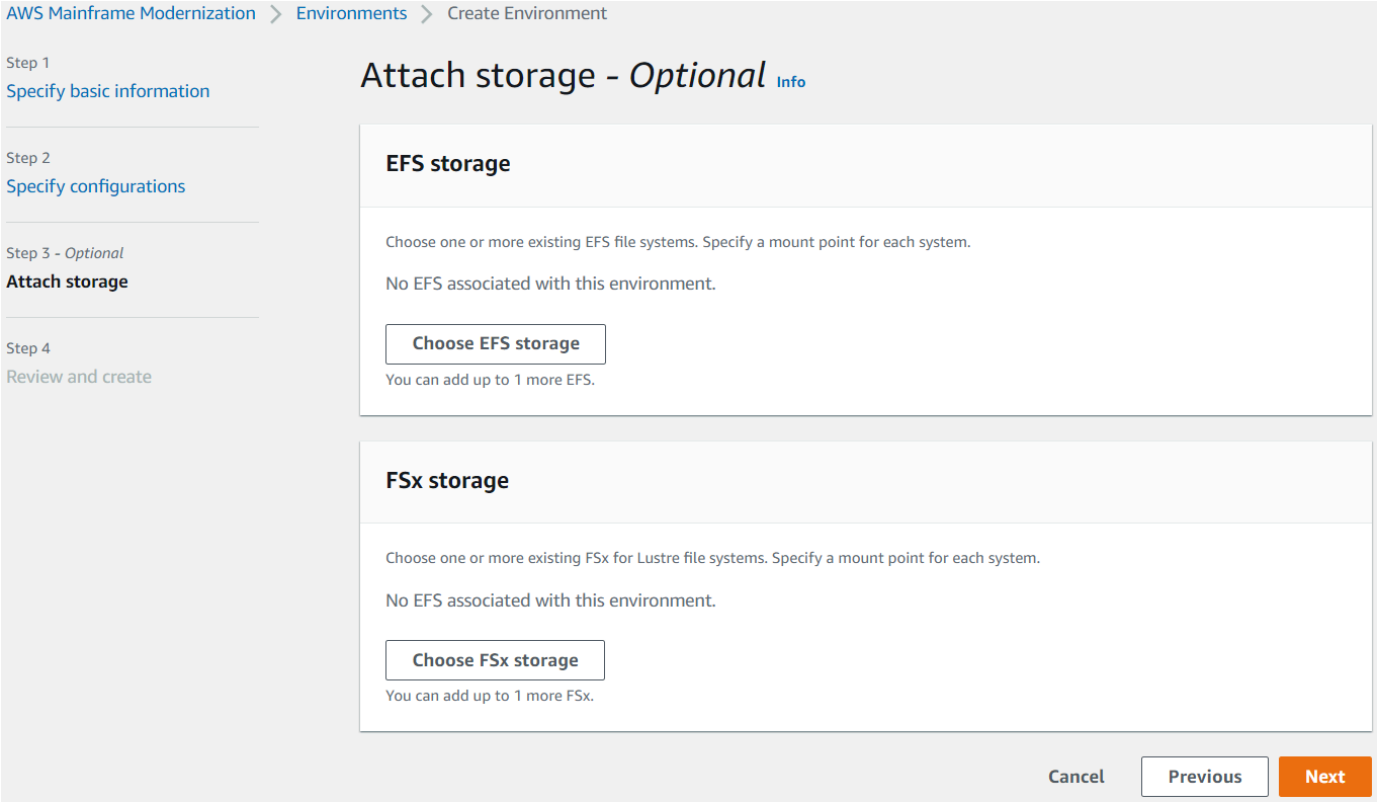
7. 보안 및 네트워크에서 다음과 같이 변경합니다.

   - 이 환경에 배포된 애플리케이션을 공개적으로 액세스할 수 있도록 허용을 선택합니다. 이 옵션은 데스크톱에서 액세스할 수 있도록 애플리케이션에 퍼블릭 IP 주소를 할당합니다.
   - VPC를 선택합니다. 기본값을 사용할 수 있습니다.
   - 서브넷 두 개를 선택합니다. 서브넷에서 퍼블릭 IP 주소 할당을 허용하는지 확인하세요.
   - 보안 그룹을 선택합니다. 기본값을 사용할 수 있습니다. 선택한 보안 그룹이 브라우저 IP 주소로부터 애플리케이션 정의의 `listener` 속성에 지정한 포트에 액세스할 수 있도록 허용하는지 확인하세요. 자세한 설명은 2단계: 애플리케이션 정의 만들기 섹션을 참조하세요.



   선택한 VPC 외부에서 애플리케이션에 액세스하려는 경우 해당 VPC의 인바운드 규칙이 제대로 구성되어 있는지 확인하세요. 자세한 설명은 애플리케이션 URL에 접근할 수 없습니다 섹션을 참조하세요.

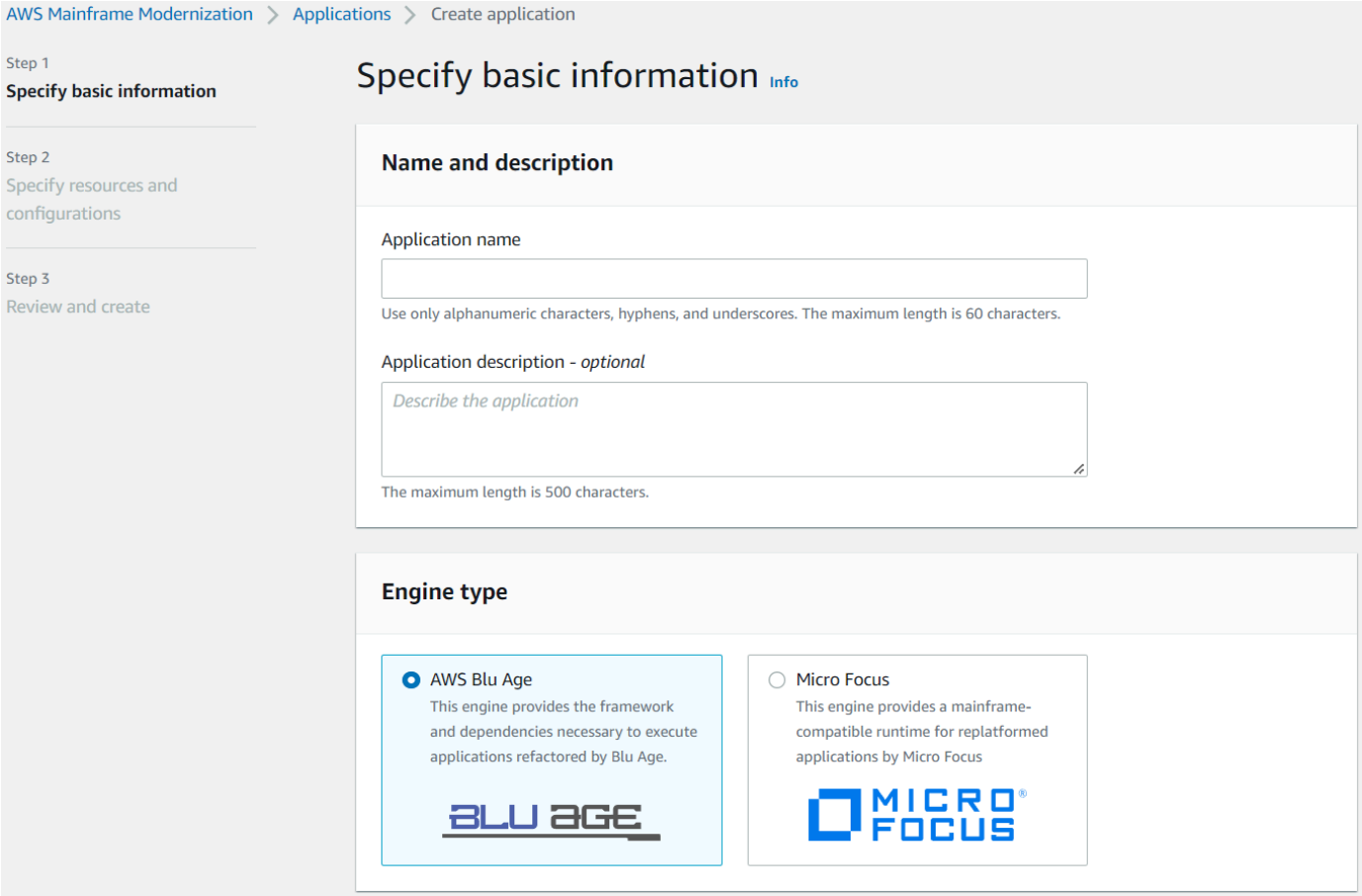
8. 다음을 선택합니다.
9. 스토리지 연결 - 선택 사항에서 기본 선택 사항을 그대로 두고 다음을 선택합니다.



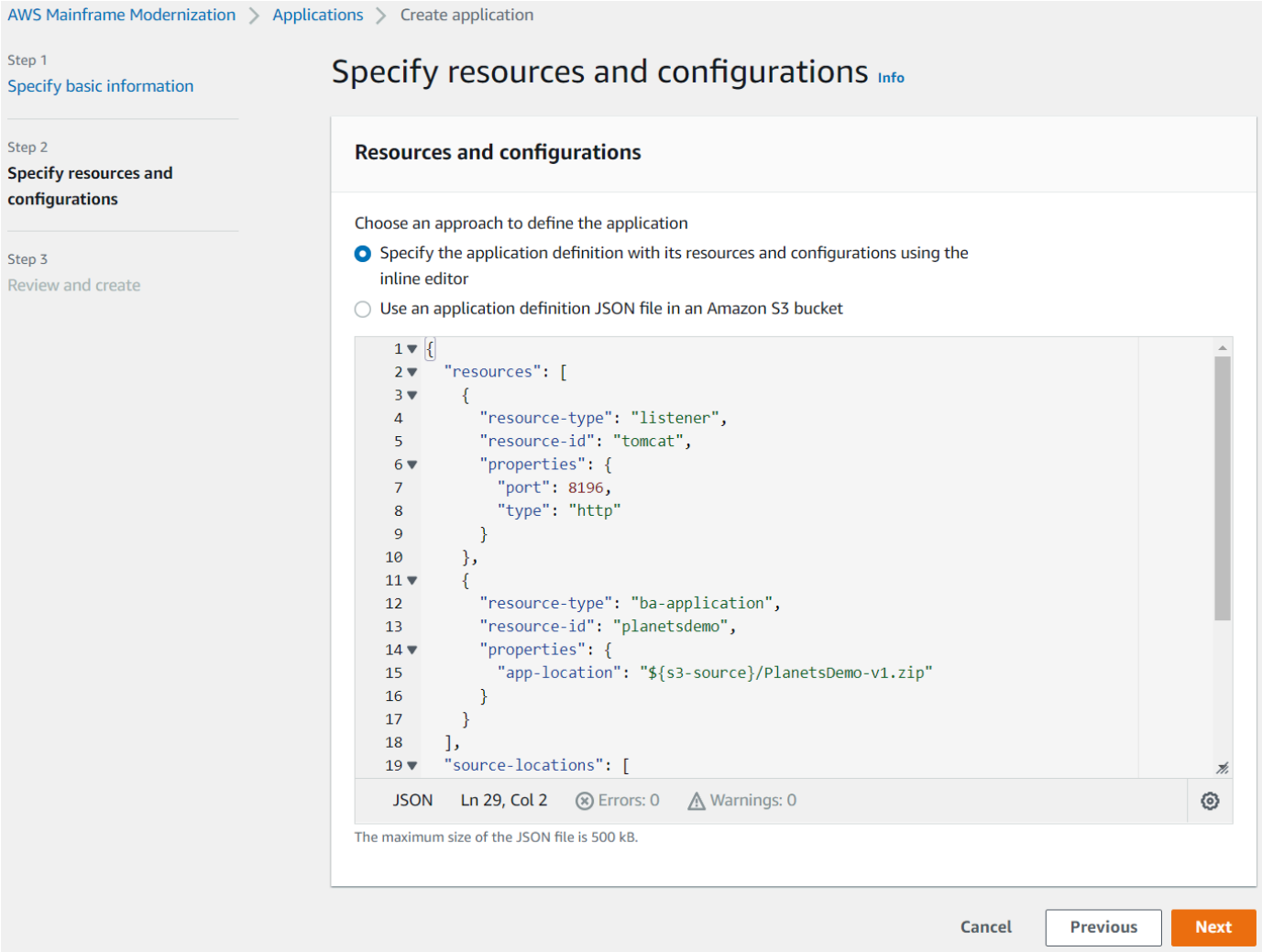
10. 일정 유지 관리에서 기본 설정 없음을 선택한 후 다음을 선택합니다.
11. 검토 및 만들기에서 정보를 검토한 다음 환경 만들기를 선택합니다.

4단계: 애플리케이션 생성

1. AWS Management Console에서 AWS Mainframe Modernization로 이동하세요.
2. 탐색 창에서 애플리케이션(Applications)을 선택한 다음 애플리케이션 생성(Create application)을 선택합니다. 기본 정보 지정 페이지에서 애플리케이션의 이름과 설명을 입력하고 AWS Blu Age 엔진이 선택되었는지 확인합니다. 다음을 선택합니다.



3. 리소스 및 구성 지정 페이지에서 the section called “2단계: 애플리케이션 정의 만들기”에서 생성한 업데이트된 애플리케이션 정의 JSON을 복사하여 붙여넣습니다.

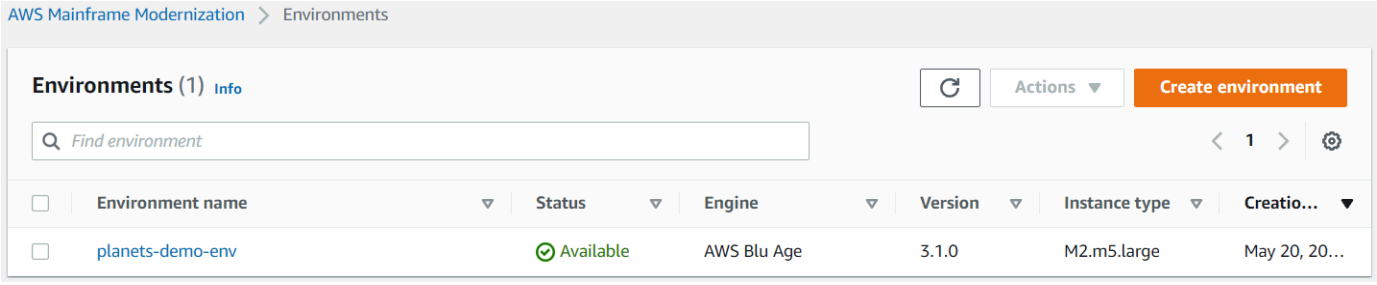


4. 검토 및 생성에서 선택 사항을 검토한 다음 애플리케이션 생성을 선택합니다.

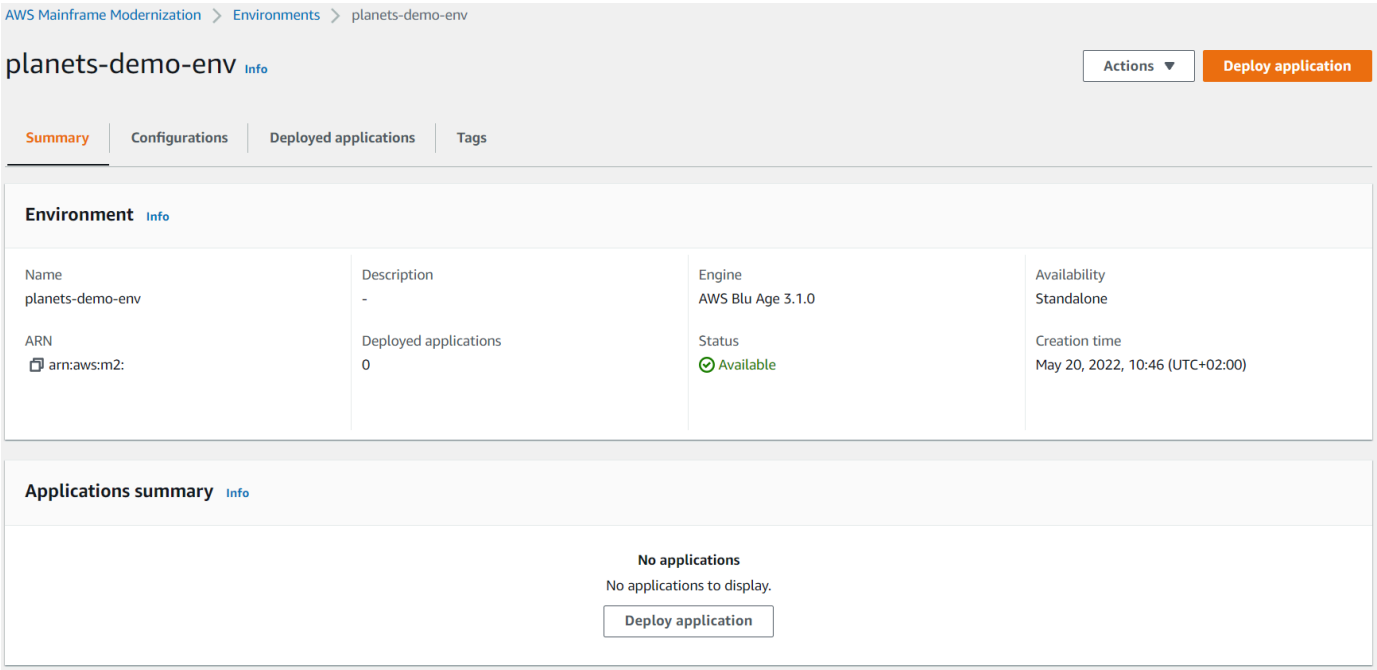
5단계: 애플리케이션 배포

AWS Mainframe Modernization 런타임 환경과 애플리케이션을 모두 성공적으로 생성하고 둘 다 사용 가능 상태가 되면 애플리케이션을 런타임 환경에 배포할 수 있습니다. 이렇게 하려면 다음 단계를 완료하세요.

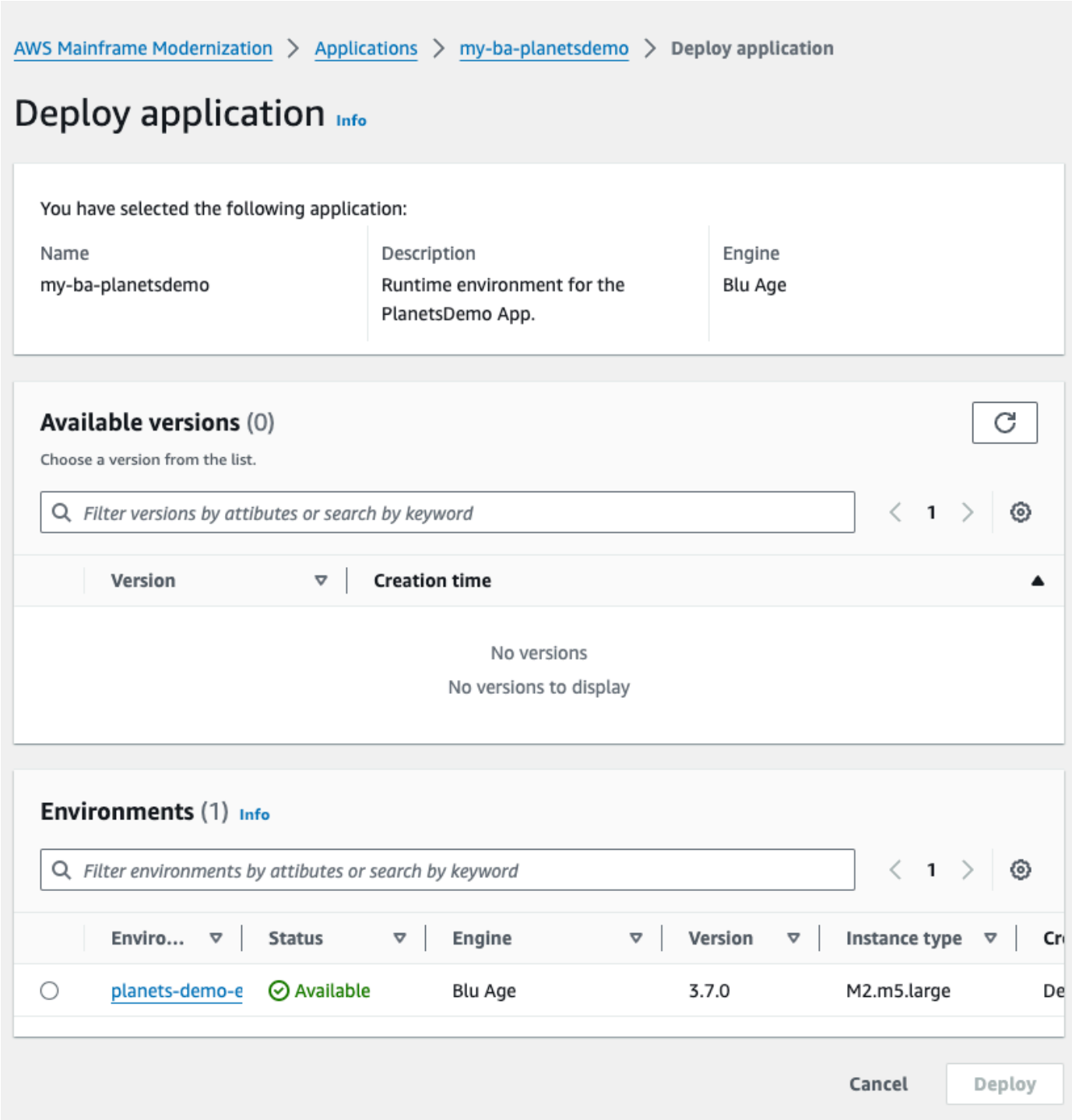
1. AWS 관리 콘솔에서 AWS Mainframe Modernization으로 이동하세요. 탐색 창에서 환경을 선택합니다. 환경 목록 페이지가 표시됩니다.



2. 이전에 만든 런타임 환경을 선택합니다. 환경 세부 정보 페이지가 표시됩니다.
3. 애플리케이션 배포를 선택합니다.



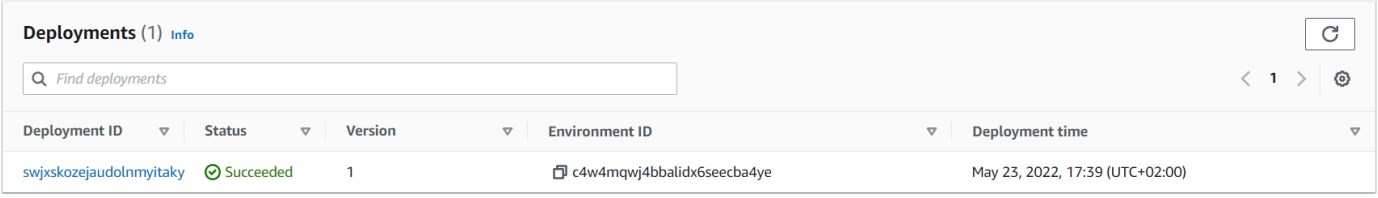
4. 이전에 만든 애플리케이션을 선택한 다음 애플리케이션을 배포할 버전을 선택합니다. 그런 다음 배포를 선택합니다.



5. 애플리케이션 배포가 완료될 때까지 기다리세요. 애플리케이션이 성공적으로 배포되었습니다라는 메시지가 포함된 배너가 표시됩니다.

6단계: 애플리케이션 시작

1. AWS Management Console에서 AWS Mainframe Modernization로 이동하여 애플리케이션을 선택합니다.
2. 애플리케이션 페이지로 이동하고 배포를 선택합니다. 애플리케이션 상태는 성공이어야 합니다.



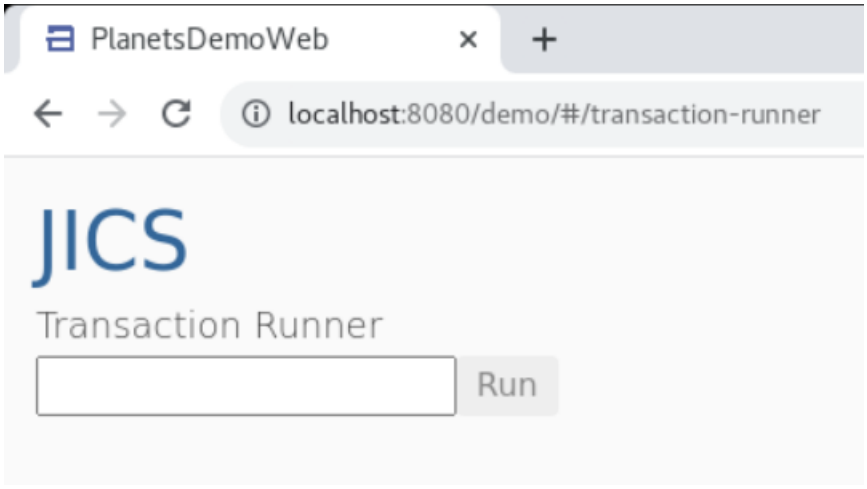
3. 작업을 선택한 후 애플리케이션 시작을 선택합니다.

7단계: 애플리케이션 액세스

1. 애플리케이션이 실행 상태가 될 때까지 기다리세요. 애플리케이션이 성공적으로 시작되었습니다라는 메시지가 포함된 배너가 표시됩니다.
2. 애플리케이션 DNS 호스트 이름을 복사합니다. 애플리케이션의 애플리케이션 정보 섹션에서 이 호스트 이름을 찾을 수 있습니다.
3. 브라우저에서 `http://{hostname}:{portname}/PlanetsDemo-web-1.0.0/`으로 이동하세요.

   - `hostname`은 이전에 DNS 호스트 이름이 복사되었습니다.
   - `portname`은 2단계: 애플리케이션 정의 만들기에서 생성한 애플리케이션 정의에 정의된 Tomcat 포트입니다.

   JICS 화면이 나타납니다.



애플리케이션에 액세스할 수 없는 경우 애플리케이션 URL에 접근할 수 없습니다를 참조하세요.

> **Note**
>
> 애플리케이션에 액세스할 수 없고 보안 그룹의 인바운드 규칙에 포트 8196에서 '내 IP'가 선택된 경우 포트 8196에서 LB i/p의 트래픽을 허용하는 규칙을 지정합니다.

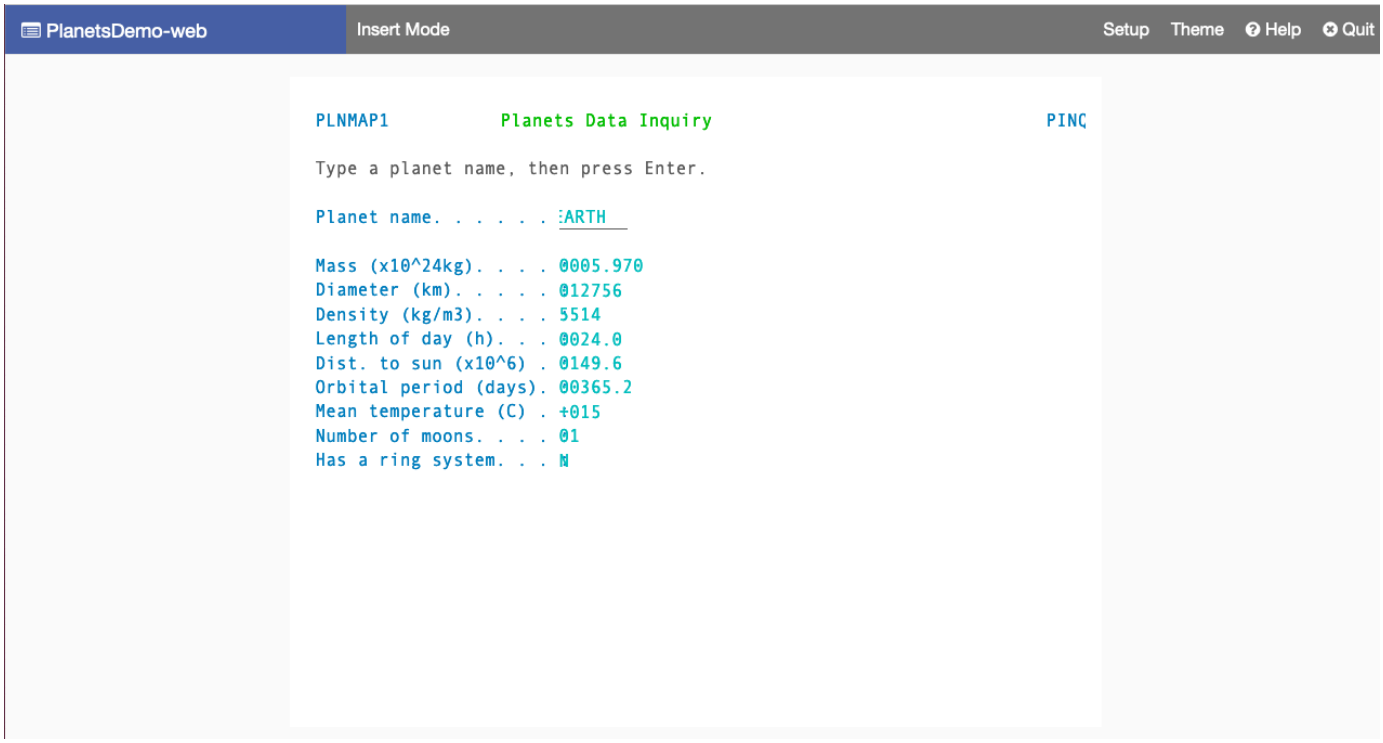
8단계: 애플리케이션 테스트

이 단계에서는 마이그레이션된 애플리케이션에서 트랜잭션을 실행합니다.

1. JICS 화면에서 입력 필드에 `PINQ`를 입력하고 실행을 선택하거나 Enter 키를 눌러 애플리케이션 트랜잭션을 시작합니다.

   데모 앱 화면이 나타나야 합니다.

2. 해당 필드에 행성 이름을 입력하고 Enter 키를 누릅니다.



행성에 대한 세부 정보를 볼 수 있을 것입니다.

리소스 정리

이 자습서 용도로 생성한 리소스가 더 이상 필요하지 않은 경우 삭제하여 추가 비용이 발생하지 않도록 하세요. 이렇게 하려면 다음 단계를 완료합니다.

- AWS Mainframe Modernization 애플리케이션이 아직 실행 중이면 중지하세요.
- 애플리케이션을 삭제합니다. 자세한 설명은 AWS 메인프레임 현대화 애플리케이션 삭제 섹션을 참조하세요.
- 런타임 환경을 삭제합니다. 자세한 내용은 AWS 메인프레임 현대화 런타임 환경 삭제을(를) 참조하세요.

자습서: Micro Focus의 관리형 런타임

이 자습서에서는 Micro Focus 런타임 엔진을 사용하여 AWS 메인프레임 현대화 관리형 런타임 환경에서 CardDemo 샘플 애플리케이션을 배포하고 실행하는 방법을 보여줍니다. CardDemo 샘플 애플리케이션은 메인프레임 현대화 사용 사례를 위한 기술을 테스트하고 AWS 소개하고 파트너십을 맺기 위해 개발된 간소화된 신용 카드 애플리케이션입니다.

이 자습서에서는 다른 AWS 서비스에서 리소스를 생성합니다. 여기에는 Amazon 심플 스토리지 서비스, Amazon 관계형 데이터베이스 서비스AWS Key Management Service, AWS Secrets Manager 등이 포함됩니다.

주제

- 필수 조건
- 1단계: Amazon S3 버킷 생성 및 로드
- 2단계: 데이터베이스 생성 및 구성
- 3단계: 생성 및 구성 AWS KMS key
- 4단계: AWS Secrets Manager 데이터베이스 시크릿 생성 및 구성
- 5단계: 런타임 환경 만들기
- 6단계: 애플리케이션 만들기
- 7단계: 애플리케이션 배포
- 8단계: 데이터 세트 가져오기
- 9단계: 애플리케이션 시작
- 10단계: CardDemo CICS 애플리케이션에 연결

- 리소스 정리
- 다음 단계

필수 조건

- CICS 연결을 사용하려면 3270 에뮬레이터에 액세스할 수 있어야 합니다. 타사 웹 사이트에서 무료 및 평가판 3270 에뮬레이터를 사용할 수 있습니다. 또는 AWS 메인프레임 현대화 AppStream 2.0 Micro Focus 인스턴스를 시작하고 Rumba 3270 에뮬레이터 (무료로 제공되지 않음) 를 사용할 수도 있습니다.

  2.0에 대한 자세한 내용은 을 참조하십시오. AppStream the section called “자습서: Enterprise Analyzer 및 Enterprise Developer용 AppStream 2.0 설정”

  > **Note**
  >
  > 스택을 생성할 때는 엔터프라이즈 분석기 (EA) 가 아닌 엔터프라이즈 개발자 (ED) 옵션을 선택하십시오.

- CardDemo 샘플 애플리케이션을 다운로드하고 다운로드한 파일을 로컬 디렉토리에 압축을 풉니다. 이 디렉토리에는 라는 제목의 하위 디렉토리가 포함됩니다. `CardDemo`
- 이 자습서에서 생성한 리소스를 정의할 수 있는 계정의 VPC를 식별하십시오. VPC는 최소 두 개의 가용 영역에 서브넷이 있어야 합니다. Amazon VPC에 대한 자세한 내용은 Amazon VPC의 작동 방식을 참조하십시오.

1단계: Amazon S3 버킷 생성 및 로드

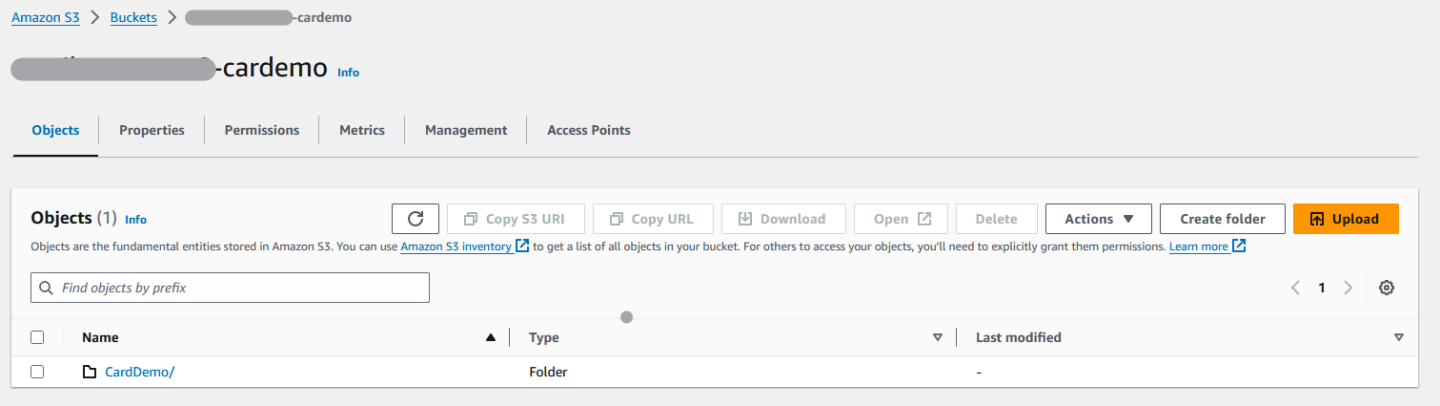
이 단계에서는 Amazon S3 버킷을 생성하고 이 버킷에 CardDemo 파일을 업로드합니다. 이 자습서의 뒷부분에서는 이러한 파일을 사용하여 AWS 메인프레임 현대화 Micro Focus 관리형 런타임 환경에서 CardDemo 샘플 애플리케이션을 배포하고 실행합니다.

> **Note**
>
> 새 S3 버킷을 만들 필요는 없지만 선택한 버킷은 이 자습서에서 사용하는 다른 리소스와 동일한 지역에 있어야 합니다.

Amazon S3 버킷을 생성하려면

1. Amazon S3 콘솔을 열고 [버킷 생성] 을 선택합니다.
2. 일반 구성에서 AWS 메인프레임 현대화 Micro Focus 관리형 런타임을 구축하려는 AWS 지역을 선택합니다.
3. 버킷 이름을 입력합니다 (예:). `yourname-aws-region-carddemo` 기본 설정을 유지하고 [Create bucket] 을 선택합니다. 또는 기존 Amazon S3 버킷에서 설정을 복사한 다음 버킷 생성을 선택할 수도 있습니다.
4. 방금 생성한 버킷을 선택한 다음 Upload를 선택합니다.
5. [업로드] 섹션에서 [Add Folder] 를 선택한 다음 로컬 컴퓨터에서 `CardDemo` 디렉터리를 탐색합니다.
6. 업로드를 선택하여 업로드 프로세스를 시작합니다. 업로드 시간은 연결 속도에 따라 달라집니다.
7. 업로드가 완료되면 모든 파일이 성공적으로 업로드되었는지 확인한 다음 [Close] 를 선택합니다.

이제 Amazon S3 버킷에 `CardDemo` 폴더가 포함됩니다.



S3 버킷에 대한 자세한 내용은 Amazon S3 버킷 생성, 구성 및 사용을 참조하십시오.

2단계: 데이터베이스 생성 및 구성

이 단계에서는 아마존 관계형 데이터베이스 서비스 (Amazon RDS) 에서 PostgreSQL 데이터베이스를 생성합니다. 자습서의 경우 이 데이터베이스에는 CardDemo 샘플 애플리케이션이 신용 카드 거래와 관련된 고객 작업에 사용하는 데이터 세트가 포함되어 있습니다.

Amazon RDS에 데이터베이스 생성

1. Amazon RDS 콘솔을 엽니다.

2. 데이터베이스 인스턴스를 생성하려는 AWS 리전을 선택합니다.
3. 탐색 창에서 Databases(데이터베이스)를 선택합니다.
4. [데이터베이스 생성] 을 선택한 다음 [표준 생성] 을 선택합니다.
5. 엔진 유형에서 PostgreSQL을 선택합니다.
6. 15 이상의 엔진 버전을 선택합니다.

> **Note**
>
> 이 자습서의 뒷부분에서 필요하므로 엔진 버전을 저장하십시오.

7. Templates(템플릿) 섹션에서 Free tier(프리 티어)를 선택합니다.
8. DB 인스턴스 식별자를 의미 있는 것으로 변경하십시오 (예:)`MicroFocus-Tutorial`.
9. 에서는 마스터 자격 증명을 관리하지 마십시오AWS Secrets Manager. 대신 마스터 비밀번호를 입력하고 확인하십시오.

> **Note**
>
> 데이터베이스에 사용하는 사용자 이름과 암호를 저장합니다. 이 자습서의 다음 단계에서는 데이터를 안전하게 저장합니다.

10. 연결에서 AWS 메인프레임 현대화 관리형 런타임 환경을 만들 VPC를 선택합니다.
11. 데이터베이스 생성를 선택합니다.

Amazon RDS에서 사용자 지정 파라미터 그룹을 만들려면

1. Amazon RDS 콘솔 탐색 창에서 파라미터 그룹을 선택한 다음 파라미터 그룹 생성을 선택합니다.
2. 파라미터 그룹 생성 창에서 파라미터 그룹 패밀리에 대해 데이터베이스 버전과 일치하는 Postgres 옵션을 선택합니다.

> **Note**
>
> 일부 Postgres 버전에는 유형이 필요합니다. 필요한 경우 DB 파라미터 그룹을 선택합니다. 파라미터 그룹의 그룹 이름과 설명을 입력합니다.

3. 생성을 선택하세요.

사용자 지정 파라미터 그룹을 구성하려면

1. 새로 만든 파라미터 그룹을 선택합니다.
2. [Actions]를 선택한 후 [Edit]를 선택합니다.
3. 필터를 `max_prepared_transactions` 켜고 파라미터 값을 100으로 변경합니다.
4. 변경 사항 저장(Save Changes)을 선택합니다.

사용자 정의 파라미터 그룹을 데이터베이스에 연결하려면

1. Amazon RDS 콘솔 탐색 창에서 [Databases] 를 선택한 다음 수정하려는 데이터베이스 인스턴스를 선택합니다.
2. Modify(수정)를 선택합니다. Modify DB instance(DB 인스턴스 수정) 페이지가 나타납니다.

> **Note**
>
> 데이터베이스 생성 및 백업을 완료할 때까지는 수정 옵션을 사용할 수 없으며, 이 작업에는 몇 분이 걸릴 수 있습니다.

3. DB 인스턴스 수정 페이지에서 추가 구성으로 이동하여 DB 파라미터 그룹을 해당 파라미터 그룹으로 변경합니다. 목록에 파라미터 그룹이 없는 경우 올바른 데이터베이스 버전으로 생성되었는지 확인하세요.
4. Continue를 선택하고 수정 요약을 확인합니다.
5. 변경 사항을 즉시 적용하려면 [즉시 적용] 을 선택합니다.
6. DB 인스턴스 수정을 선택하여 변경 사항을 저장합니다.

매개 변수 그룹에 대한 자세한 내용은 매개 변수 그룹 작업을 참조하세요.

> **Note**
>
> Amazon Aurora PostgreSQL 데이터베이스를 AWS 메인프레임 현대화와 함께 사용할 수도 있지만 프리 티어 옵션은 없습니다. 자세한 내용은 Amazon Aurora PostgreSQL 사용을 참조하십시오.

3단계: 생성 및 구성 AWS KMS key

Amazon RDS 인스턴스의 자격 증명을 안전하게 저장하려면 먼저 자격 증명을 생성하십시오. AWS KMS key

AWS KMS key 생성

1. 키 관리 서비스 콘솔을 엽니다.
2. 키 생성을 선택합니다.
3. 키 유형은 기본값인 Symmetric을 유지하고 키 사용에 대해서는 암호화 및 복호화를 그대로 사용합니다.
4. 다음을 선택합니다.
5. 키에 별칭 (예:) 과 선택적 설명을 입력합니다. `MicroFocus-Tutorial-RDS-Key`
6. 다음을 선택합니다.
7. 사용자 또는 역할 옆의 체크박스를 선택하여 키 관리자를 할당합니다.
8. [다음] 을 선택한 후 다시 [다음] 을 선택합니다.
9. 검토 화면에서 키 정책을 편집한 후 다음을 입력합니다.

```
{
     "Sid" : "Allow access for Mainframe Modernization Service",
     "Effect" : "Allow",
        "Principal" : {
           "Service" : "m2.amazonaws.com"
                 },
      "Action" : "kms:Decrypt",
      "Resource" : "*"
},
```

   이 정책은 이 특정 키 정책을 사용하여 AWS 메인프레임 현대화 암호 해독 권한을 부여합니다.
10. Finish를 선택하여 키를 생성합니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 키 만들기를 참조하십시오.

4단계: AWS Secrets Manager 데이터베이스 시크릿 생성 및 구성

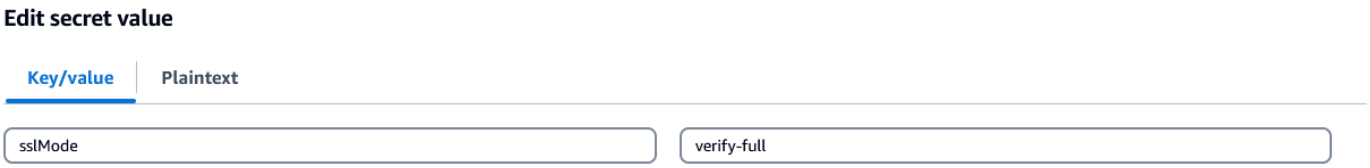
이제 AWS Secrets Manager 및 를 사용하여 데이터베이스 자격 증명을 안전하게 AWS KMS key 저장하십시오.

AWS Secrets Manager데이터베이스 시크릿 생성 및 구성하기

1. Secrets Manager 콘솔을 엽니다.
2. 탐색 창에서 암호를 선택합니다.
3. 시크릿에서 새 시크릿 저장을 선택합니다.
4. Amazon RDS 데이터베이스의 보안 유형을 자격 증명으로 설정합니다.
5. 데이터베이스를 생성할 때 지정한 자격 증명을 입력합니다.
6. 암호화 키에서 3단계에서 생성한 키를 선택합니다.
7. 데이터베이스 섹션에서 이 자습서를 위해 만든 데이터베이스를 선택하고 다음을 선택합니다.
8. 시크릿 이름에 이름 (예:) `MicroFocus-Tutorial-RDS-Secret` 과 선택적 설명을 입력합니다.
9. [리소스 권한] 섹션에서 [권한 편집] 을 선택하고 내용을 다음 정책으로 바꿉니다.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect" : "Allow",
            "Principal" : {
                "Service" : "m2.amazonaws.com"
            },
            "Action" : "secretsmanager:GetSecretValue",
            "Resource" : "*"
        }
    ]
}
```

10. 저장을 선택합니다.
11. 다음 화면에서 다음을 선택한 다음 Store를 선택합니다. 비밀 목록을 새로 고쳐 새 암호를 확인하십시오.
12. 새로 만든 암호를 선택하고 `Secret ARN` 자습서 뒷부분에서 필요하므로 기록해 둡니다.
13. 암호의 개요 탭에서 암호 값 검색을 선택합니다.
14. 편집을 선택한 다음 행 추가를 선택합니다.
15. 값이 다음과 같은 키 추가`verify-full`: `sslMode`

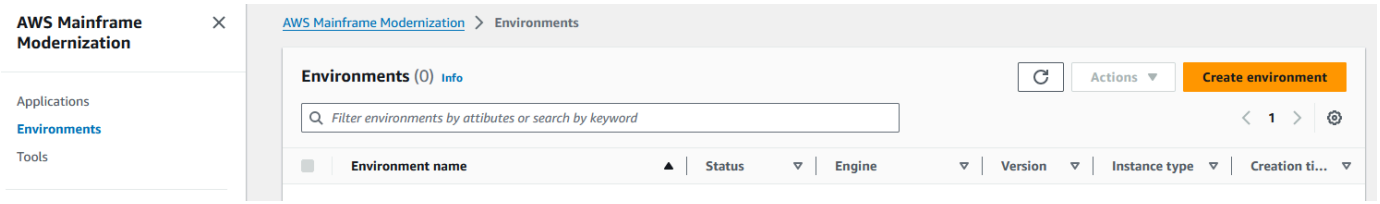


16. 저장을 선택합니다.

5단계: 런타임 환경 만들기

런타임 환경 만들기

1. AWS Mainframe Modernization 콘솔을 엽니다.
2. 탐색 창에서 환경을 선택합니다. 그런 다음 환경 만들기를 선택합니다.



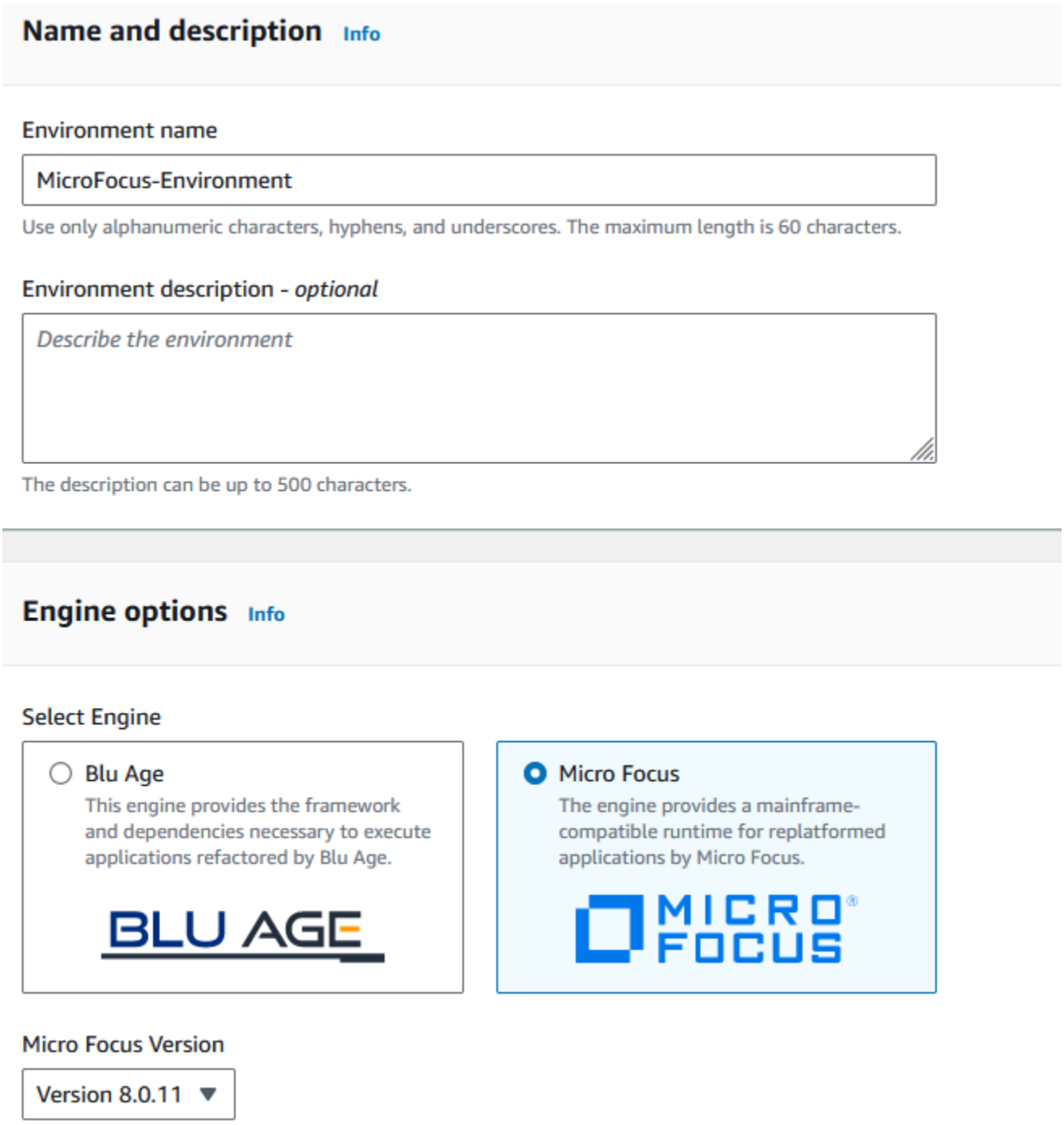
3. 기본 정보 지정에서

   a. 환경 이름으로 `MicroFocus-Environment` 입력합니다.

   b. 엔진 옵션에서 마이크로 포커스가 선택되어 있는지 확인합니다.

   c. 최신 마이크로 포커스 버전을 선택하세요.

   d. 다음을 선택합니다.



4. 환경을 구성합니다

   a. 가용성에서 고가용성 클러스터를 선택합니다.

   b. Resources (리소스) 에서 인스턴스 유형 중 하나 `M2.c5.large` 또는 `M2.m5.large` 원하는 인스턴스 수를 선택합니다. 최대 두 개의 인스턴스를 지정합니다.

c. 보안 및 네트워크에서 이 환경에 배포된 애플리케이션을 공개적으로 액세스할 수 있도록 허용을 선택하고 퍼블릭 서브넷을 두 개 이상 선택합니다.

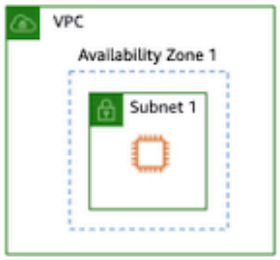
d. 다음을 선택합니다.

Specify configurations Info

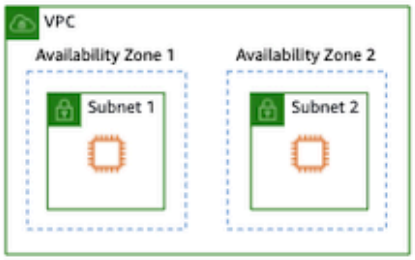
Availability Info

Choose the availability pattern for your environment.

- Standalone runtime environment
  Sets up a single instance in a single availability zone. Does not guarantee high availability but costs less.



- High availability cluster
  Sets up redundant instances across two availability zones. Enables higher availability but costs more.



Resources

Instance type
Choose the instance type for your high availability cluster.

M2.m5.large

Desired capacity
Specify the desired number of instances.

2

Security and network

☑ Allow applications deployed to this environment to be publicly accessible.

Virtual Private Cloud (VPC)
Choose the VPC where you want to create the environment.

Default vpc-15

Subnets
Choose one or more subnets for a high availability setup.

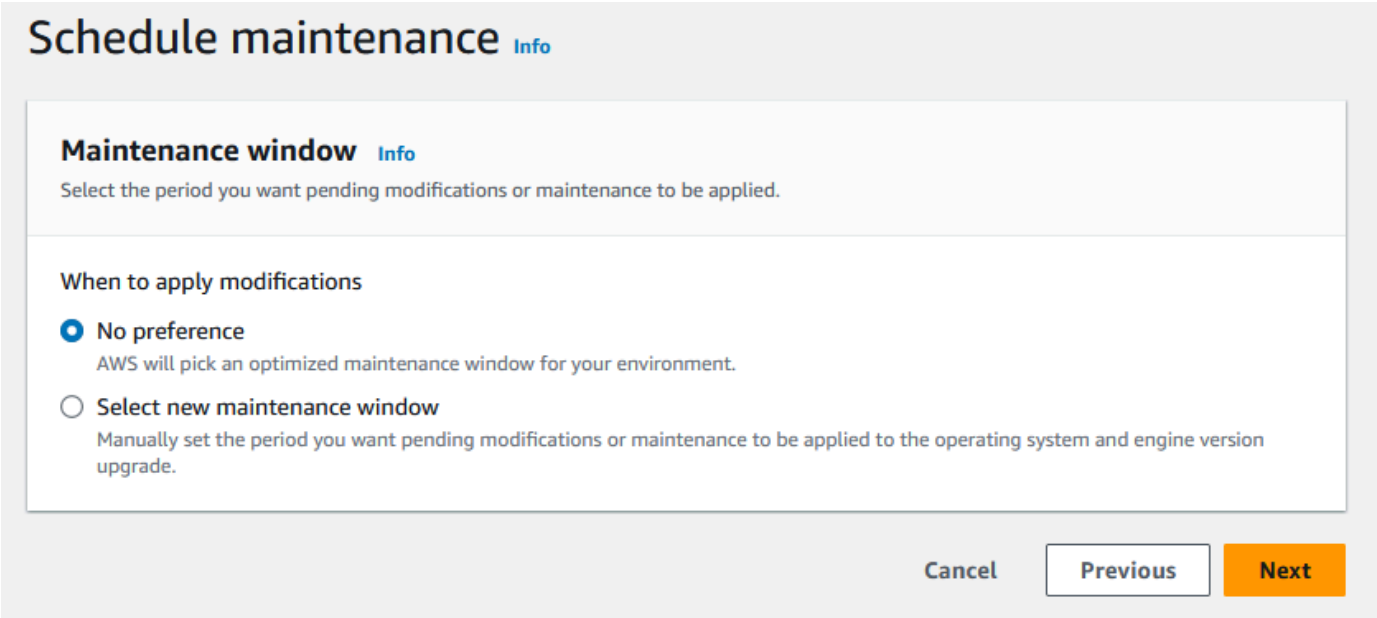
Choose subnets

subnet-56f1a | us-west-2a ×

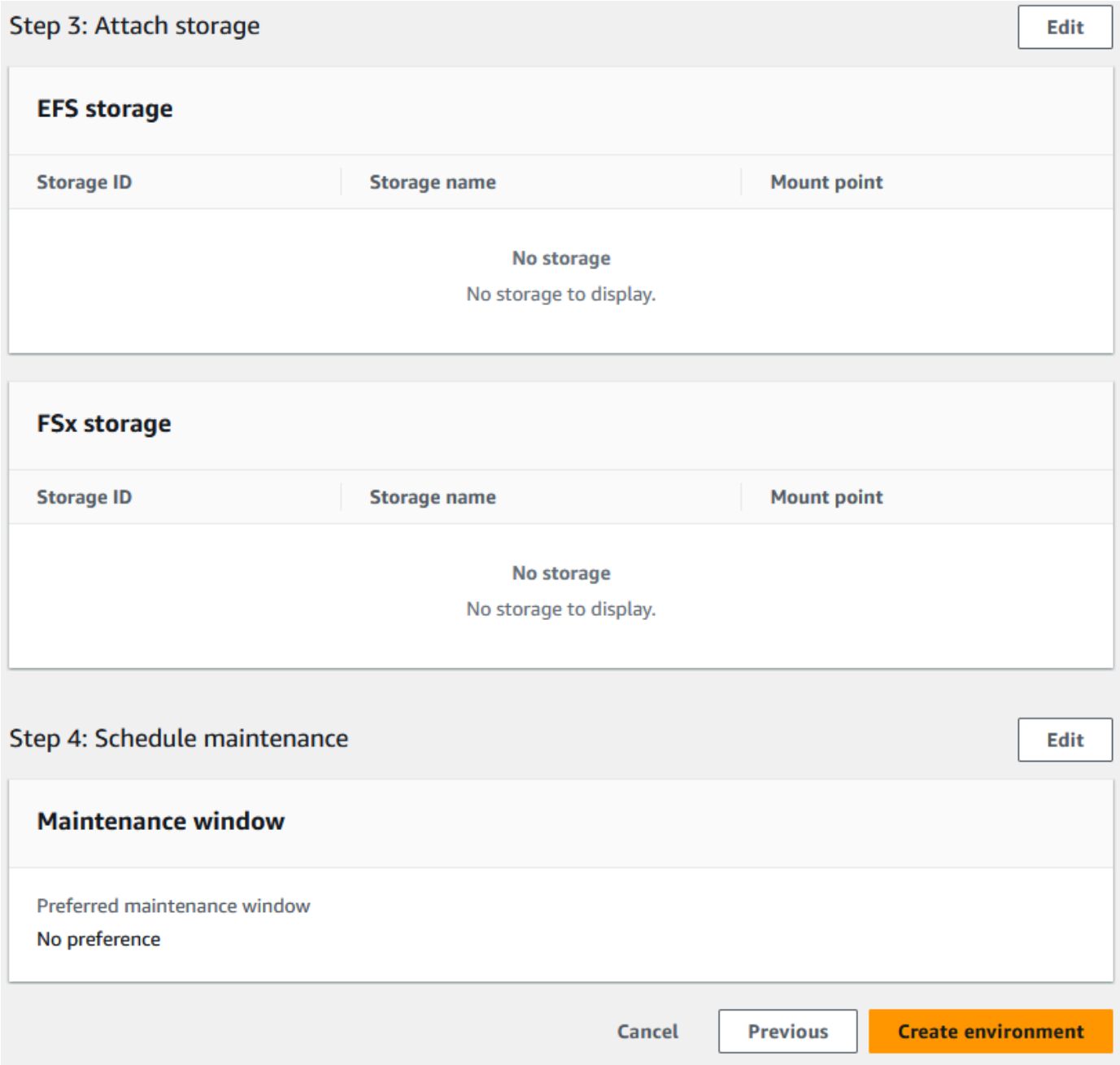
subnet-6685 | us-west-2b ×

Security groups
Choose one or more security groups for the chosen VPC.

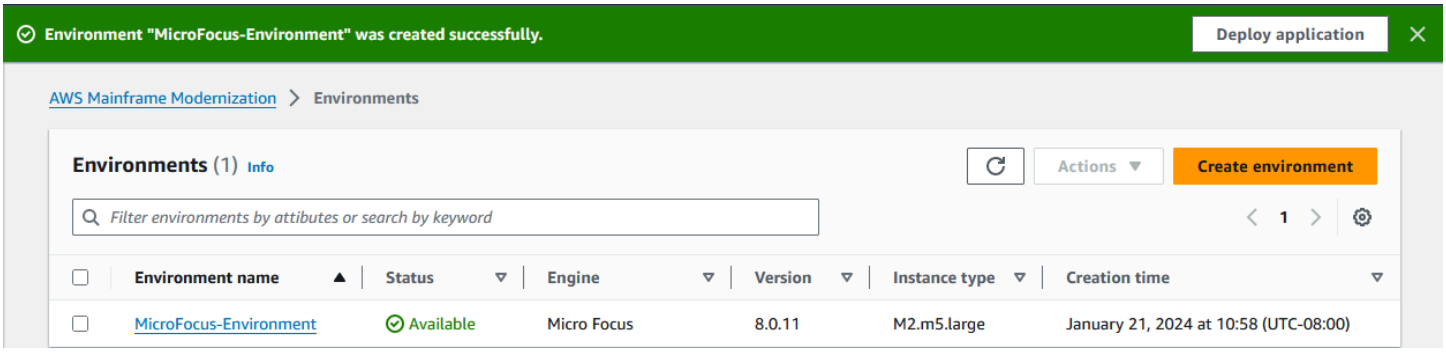
5. 스토리지 연결 페이지에서 다음을 선택합니다.
6. 일정 유지 관리 페이지에서 기본 설정 없음을 선택하고 다음을 선택합니다.



7. 검토 및 생성 페이지에서 런타임 환경에 제공한 모든 구성을 검토한 다음 환경 생성을 선택합니다.



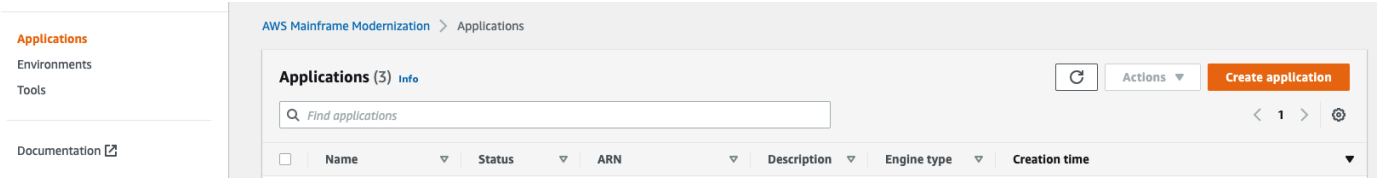
환경을 만들면 `Environment` *`name`* `was created successfully`가 적힌 배너가 나타나고 상태 필드가 사용 가능으로 바뀝니다. 환경 생성 프로세스는 몇 분 정도 걸리지만 실행 중에 다음 단계를 계속할 수 있습니다.



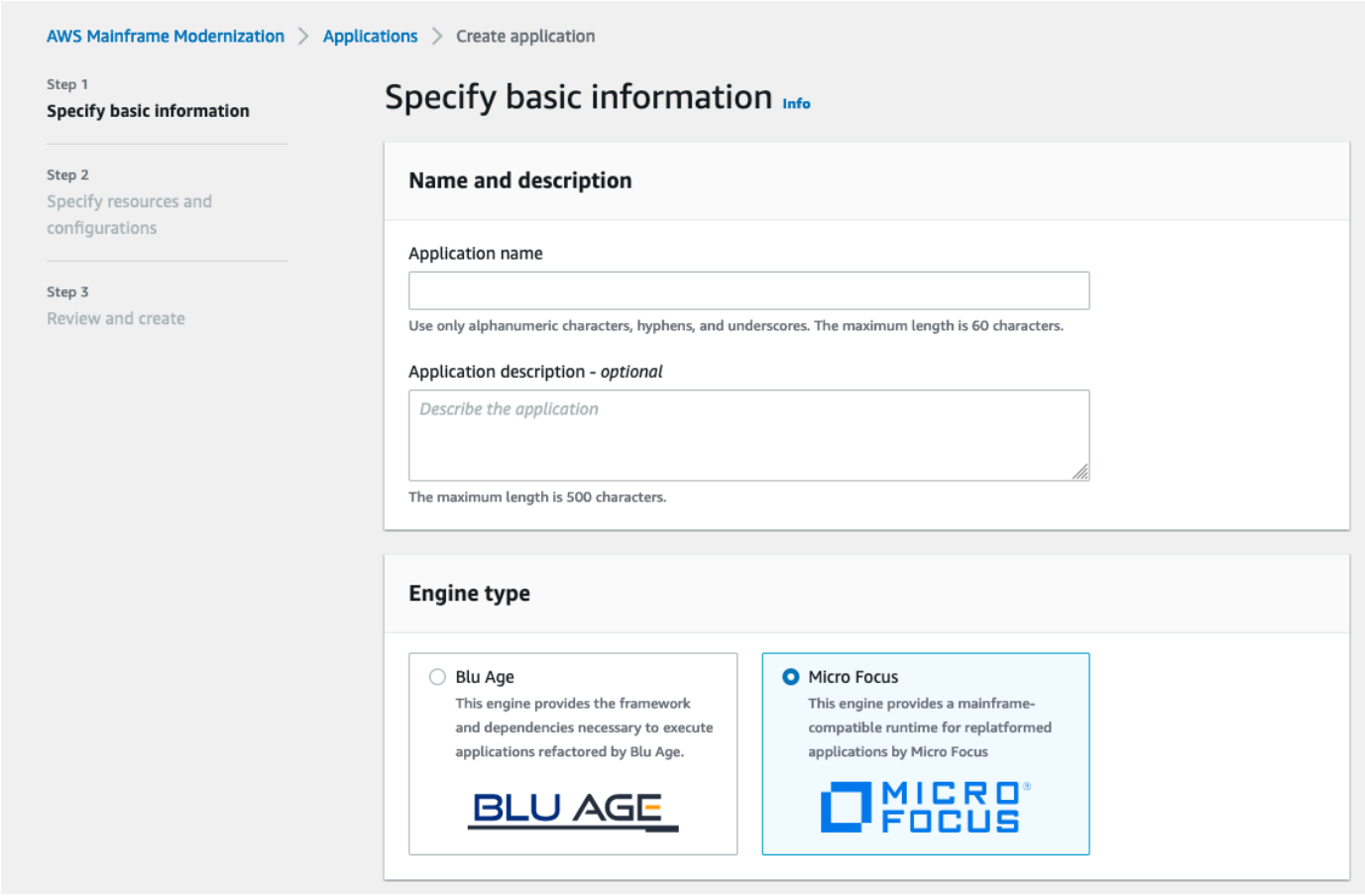
6단계: 애플리케이션 만들기

애플리케이션을 생성하려면

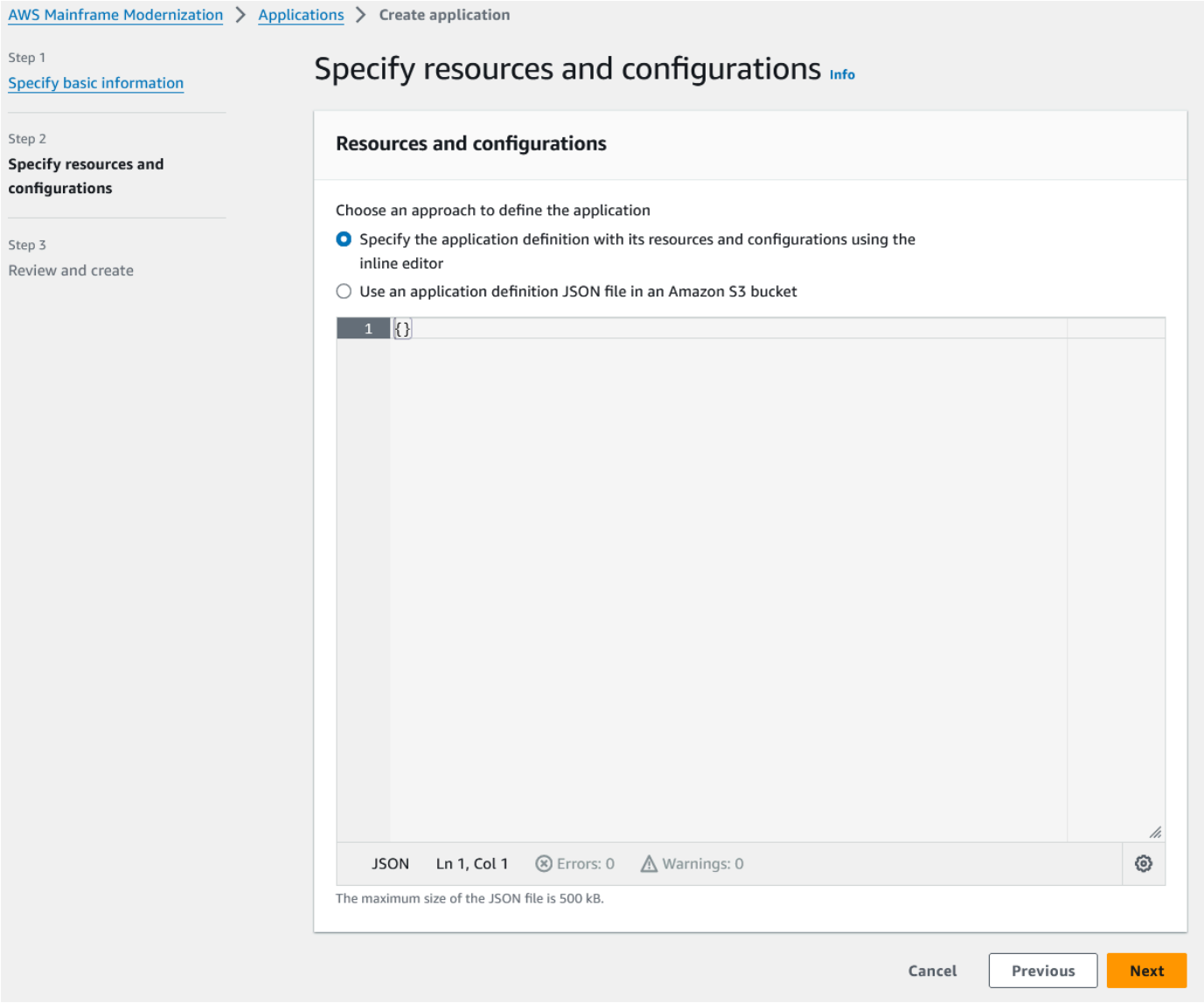
1. 탐색 창에서 애플리케이션을 선택합니다. 그 다음 애플리케이션 생성을 선택합니다.



2. 응용 프로그램 만들기 페이지의 기본 정보 지정에서 응용 프로그램 이름을 입력하고 `MicroFocus-CardDemo` 엔진 유형에서 Micro Focus가 선택되었는지 확인합니다. 다음을 선택합니다.



3. 리소스 및 구성 지정에서 인라인 편집기를 사용하여 리소스 및 구성과 함께 응용 프로그램 정의를 지정하는 옵션을 선택합니다.



편집기에 다음 애플리케이션 정의를 입력합니다.

```
{
  "template-version": "2.0",
  "source-locations": [
    {
      "source-id": "s3-source",
      "source-type": "s3",
      "properties": {
        "s3-bucket": "yourname-aws-region-carddemo",
        "s3-key-prefix": "CardDemo"
      }
    }
```

```
  ],
  "definition": {
    "listeners": [
      {
        "port": 6000,
        "type": "tn3270"
      }
    ],
    "dataset-location": {
      "db-locations": [
        {
          "name": "Database1",
          "secret-manager-arn":
 "arn:aws:secretsmanager:Region:123456789012:secret:MicroFocus-Tutorial-RDS-Secret-
xxxxxx"
        }
      ]
    },
    "batch-settings": {
      "initiators": [
        {
          "classes": [
               "A",
               "B"
            ],
          "description": "initiator_AB...."
        },
        {
          "classes": [
                "C",
                "D"
             ],
          "description": "initiator_CD...."
        }
      ],
      "jcl-file-location": "${s3-source}/catalog/jcl"
    },
    "cics-settings": {
      "binary-file-location": "${s3-source}/loadlib",
      "csd-file-location": "${s3-source}/rdef",
      "system-initialization-table": "CARDSIT"
    },
    "xa-resources": [
      {
```

```
        "name": "XASQL",
        "secret-manager-arn":
 "arn:aws:secretsmanager:Region:123456789012:secret:MicroFocus-Tutorial-RDS-Secret-
xxxxxx",
        "module": "${s3-source}/xa/ESPGSQLXA64.so"
      }
    ]
  }
}
```

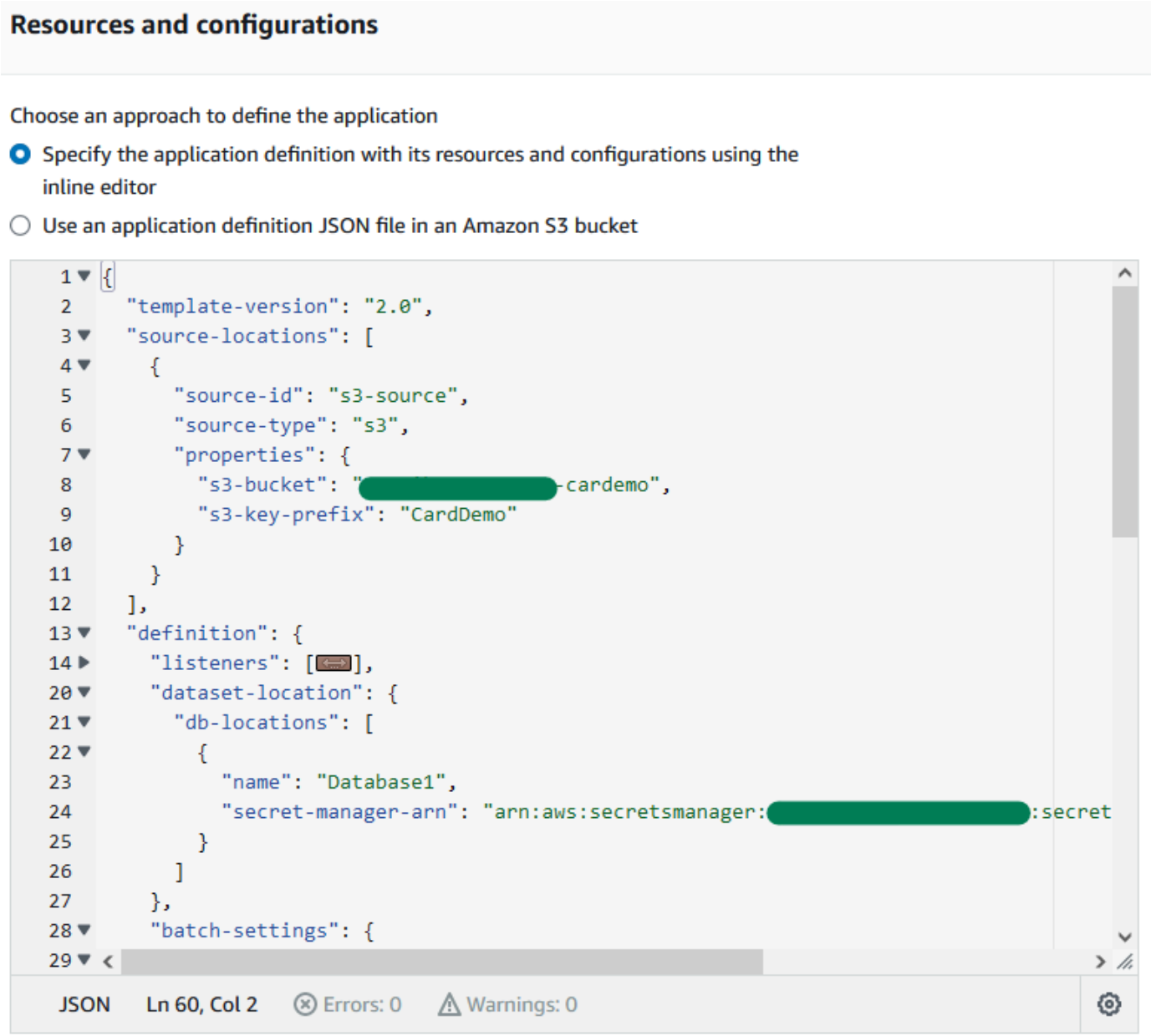
> **Note**
>
> 파일은 변경될 수 있습니다.

4. 다음과 같이 소스 위치의 속성 개체에서 애플리케이션 JSON을 편집합니다.

   a. 의 값을 `s3_bucket` 1단계에서 생성한 Amazon S3 버킷의 이름으로 대체합니다.

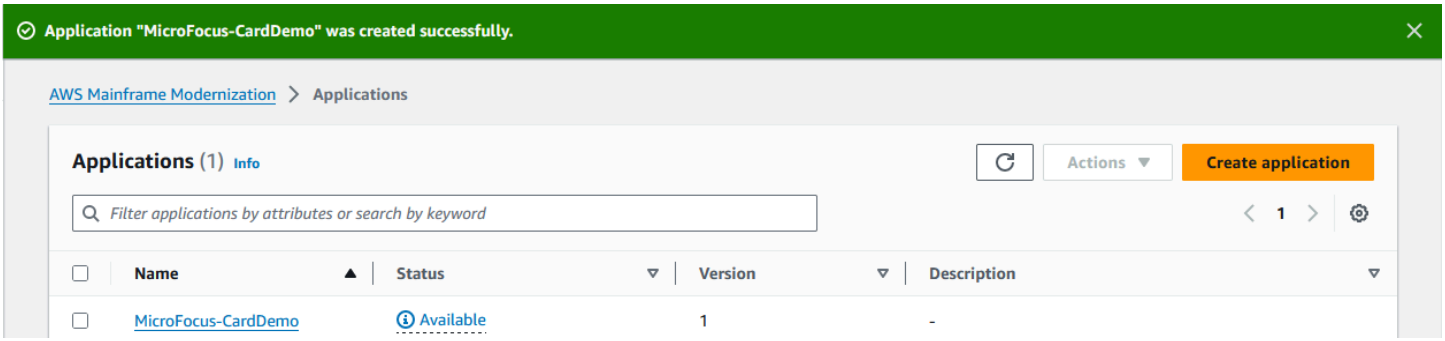
   b. 의 값을 CardDemo 샘플 파일을 업로드한 폴더 (key prefix) `s3-key-prefix` 로 바꿉니다. Amazon S3 버킷에 `CardDemo` 디렉터리를 직접 업로드한 경우 디렉터리를 변경할 `s3-key-prefix` 필요가 없습니다.

   c. 두 `secret-manager-arn` 값을 모두 4단계에서 만든 데이터베이스 암호의 ARN으로 바꿉니다.



애플리케이션 정의에 대한 자세한 내용은 Micro Focus 애플리케이션 정의를 참조하세요.

5. 다음을 선택하여 계속 진행합니다.
6. 검토 및 생성 페이지에서 제공한 정보를 검토한 다음 애플리케이션 생성을 선택합니다.

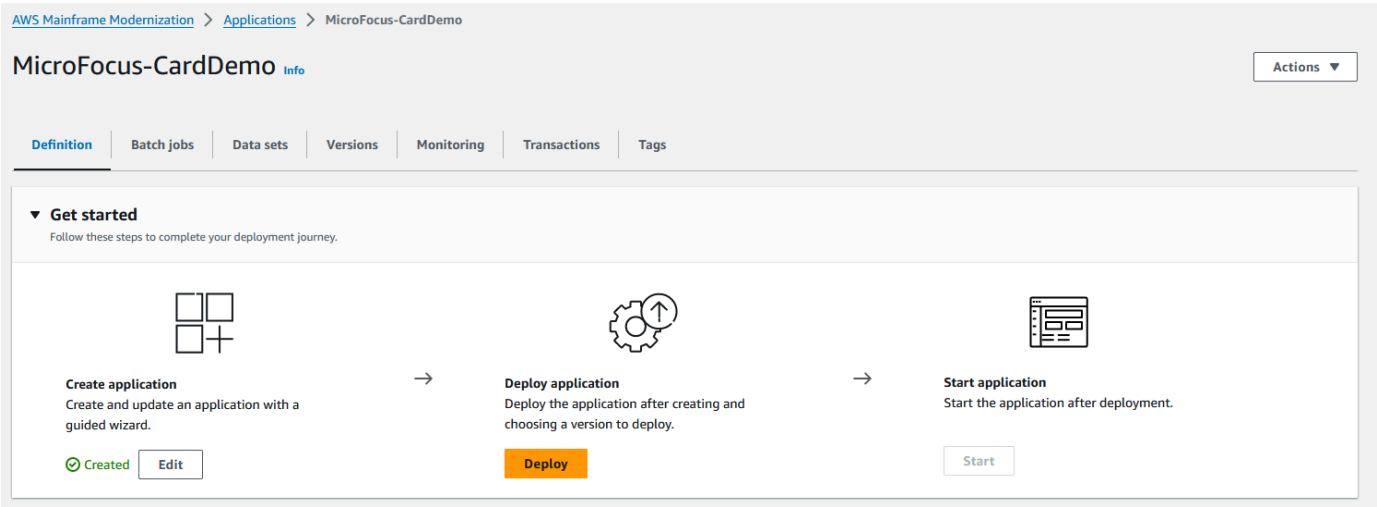


애플리케이션을 만들면 다음과 같은 배너가 나타납니다`Application` *`name`* `was created successfully`. 그러면 상태 필드가 사용 가능으로 바뀝니다.

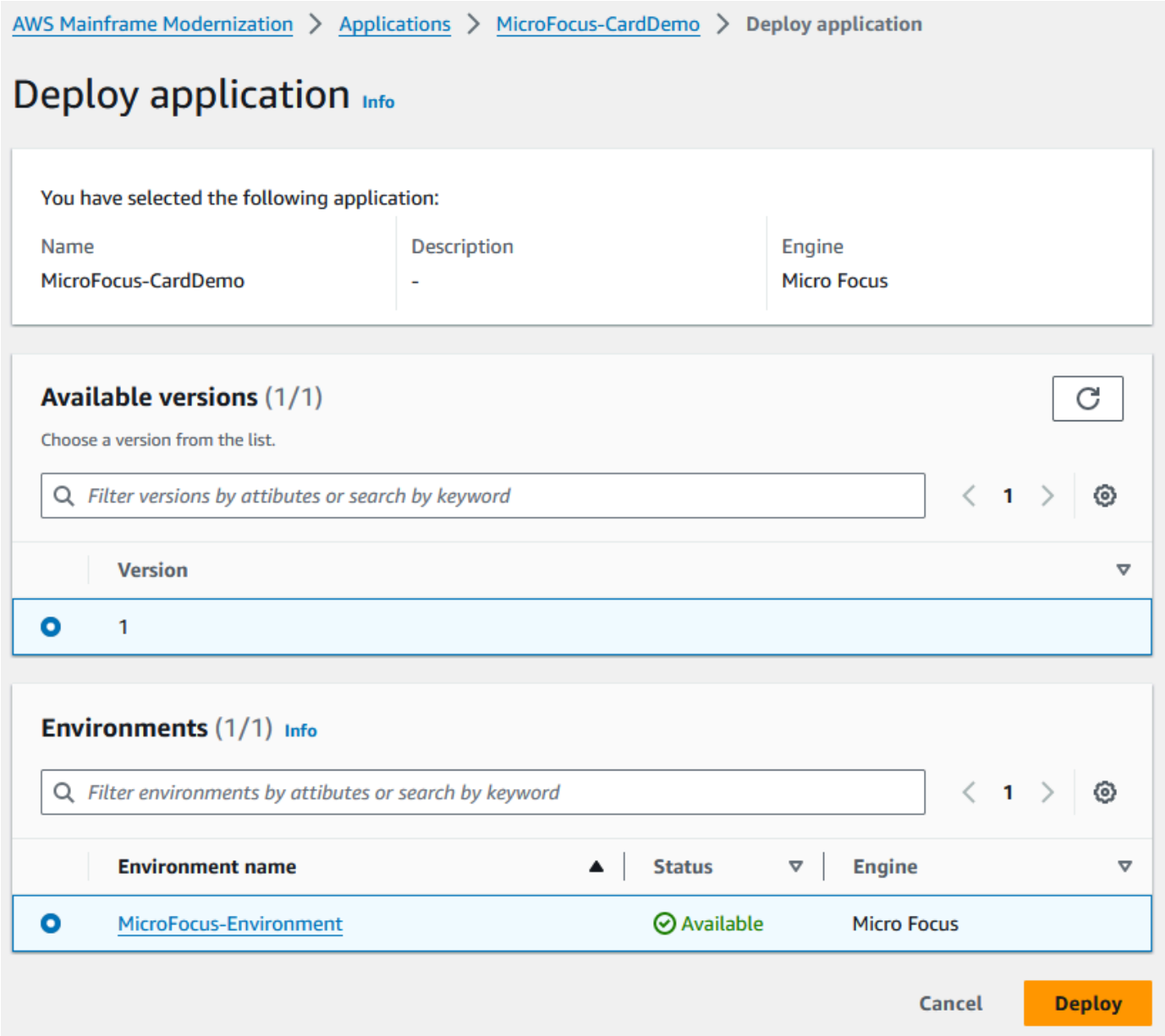
7단계: 애플리케이션 배포

애플리케이션 배포

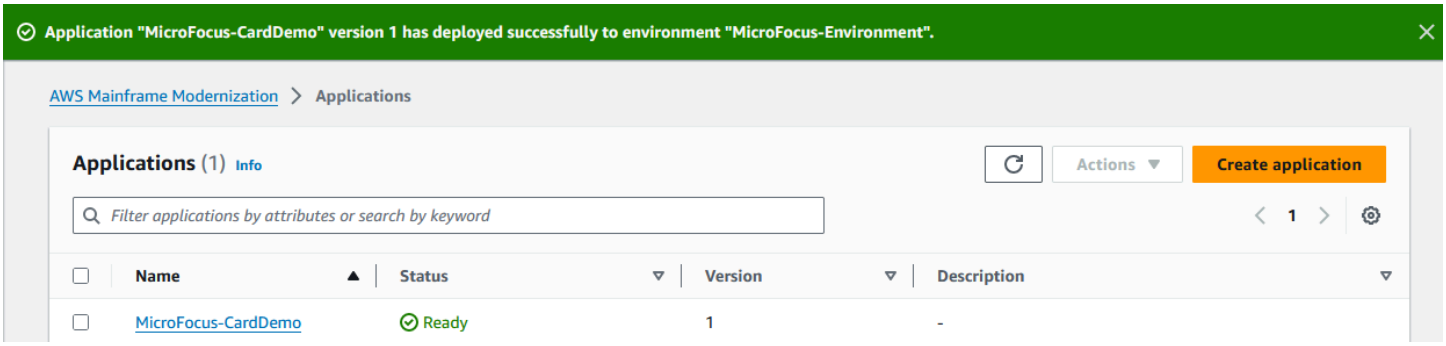
1. 탐색 창에서 애플리케이션을 선택한 다음 선택합니다`MicroFocus-CardDemo`.
2. 애플리케이션 배포에서 배포를 선택합니다.



3. 최신 버전의 애플리케이션과 이전에 만든 환경을 선택한 다음 [Deploy] 를 선택합니다.



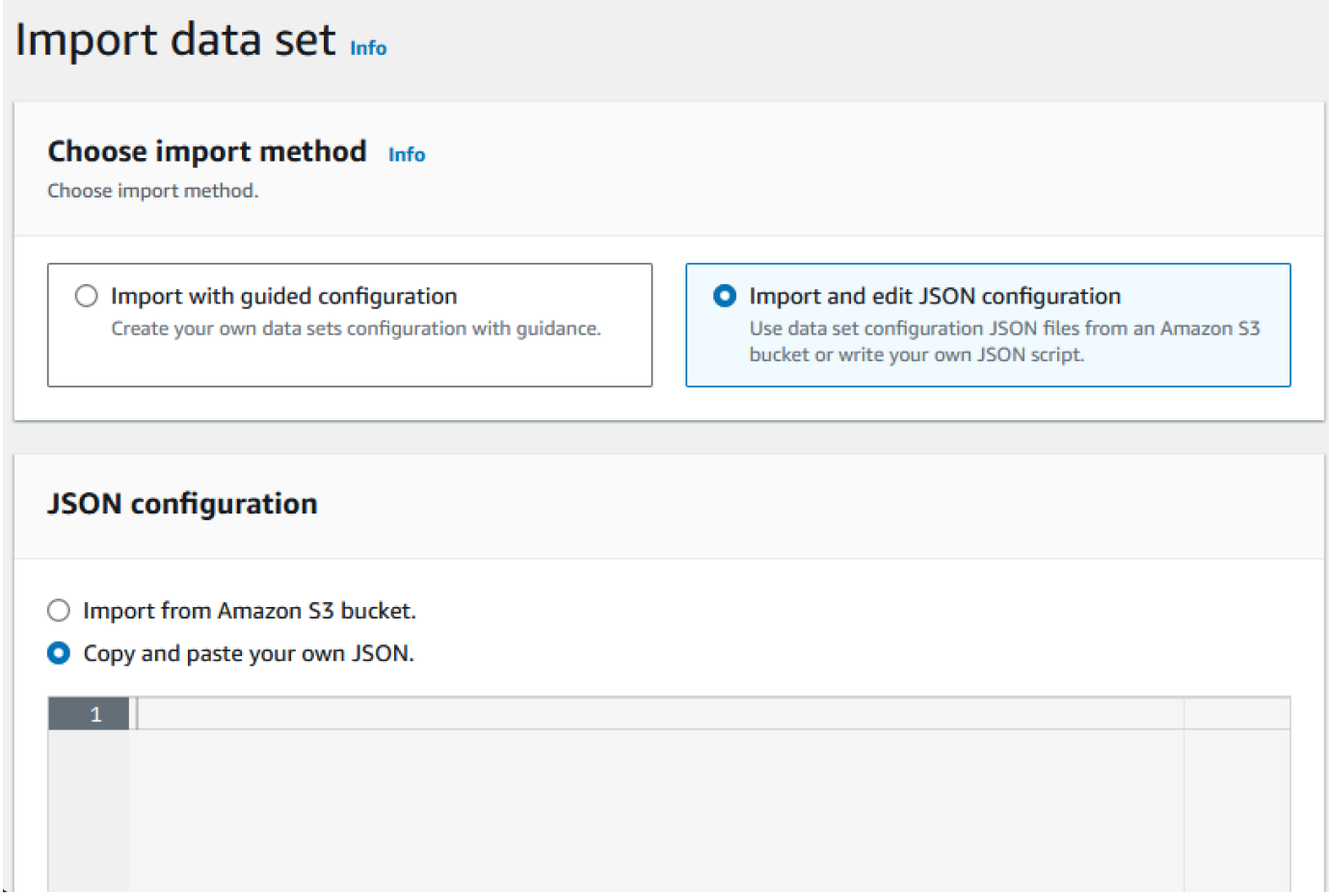
CardDemo 애플리케이션이 성공적으로 배포되면 상태가 Ready로 변경됩니다.



8단계: 데이터 세트 가져오기

데이터 세트를 가져오려면

1. 탐색 창에서 애플리케이션을 선택한 다음 애플리케이션을 선택합니다.
2. 데이터 세트 탭을 선택합니다. 그런 다음 가져오기를 선택합니다.
3. JSON 구성 가져오기 및 편집을 선택한 다음 자체 JSON 복사 및 붙여넣기 옵션을 선택합니다.



4. 다음 JSON을 복사하여 붙여넣되 아직 "제출"을 선택하지 마십시오. 이 JSON에는 데모 애플리케이션에 필요한 모든 데이터 세트가 포함되어 있지만 Amazon S3 버킷 세부 정보가 필요합니다.

```
{
    "dataSets": [
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
```

```
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 300,
                    "max": 300
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDDATA.VSAM.AIX.PATH",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 16
                        }
                    }
                },
                "recordLength": {
                    "min": 150,
                    "max": 150
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS.DAT"
            }
        },
```

```
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 150,
                    "max": 150
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 50,
                    "max": 50
                }
```

```
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CUSTDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 9,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 500,
                    "max": 500
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CUSTDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDXREF.VSAM.AIX.PATH",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 25
                        }
```

```
                    }
                },
                "recordLength": {
                    "min": 50,
                    "max": 50
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 350,
                    "max": 350
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.USRSEC.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
```

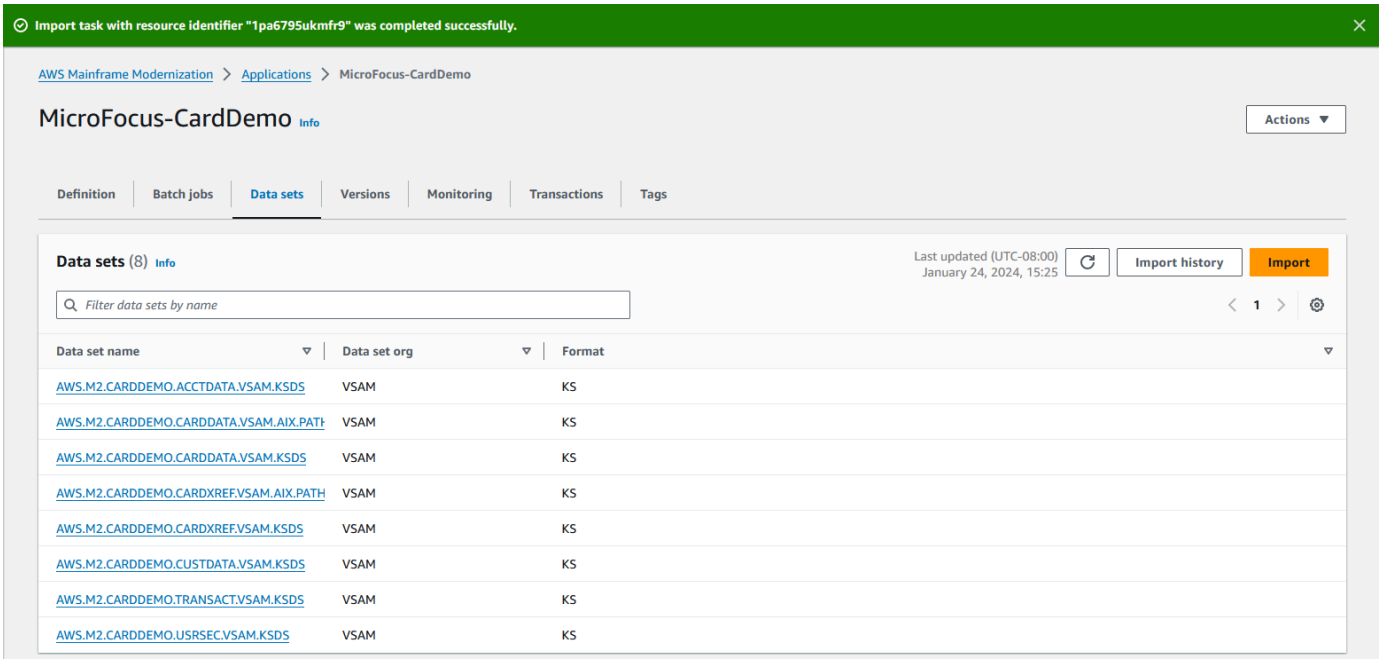
```
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 8,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 80,
                    "max": 80
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo/catalog/data/
AWS.M2.CARDDEMO.USRSEC.VSAM.KSDS.DAT"
            }
        }
    ]
}
```

5. 각 항목 <s3-bucket-name> (8개) 을 CardDemo 폴더가 포함된 Amazon S3 버킷의 이름으로 바꿉니다 (예:)your-name-aws-region-carddemo.

> **Note**
>
> Amazon S3에 있는 폴더의 Amazon S3 URI를 복사하려면 폴더를 선택한 다음 [Amazon S3 URI 복사] 를 선택합니다.

6. 제출을 선택합니다.

   가져오기가 완료되면 다음 `Import task with resource identifier` *name* `was completed successfully.` 메시지와 함께 배너가 나타납니다. 가져온 데이터 세트 목록이 표시됩니다.

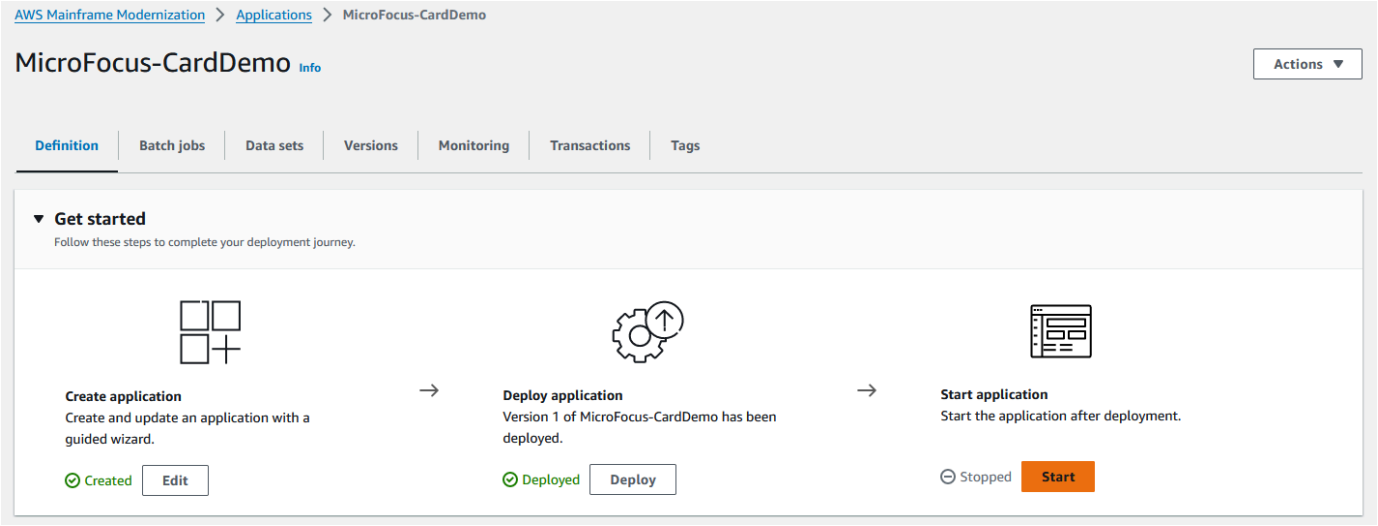


데이터 세트 탭에서 가져오기 기록을 선택하여 모든 데이터 세트 가져오기 상태를 볼 수도 있습니다.

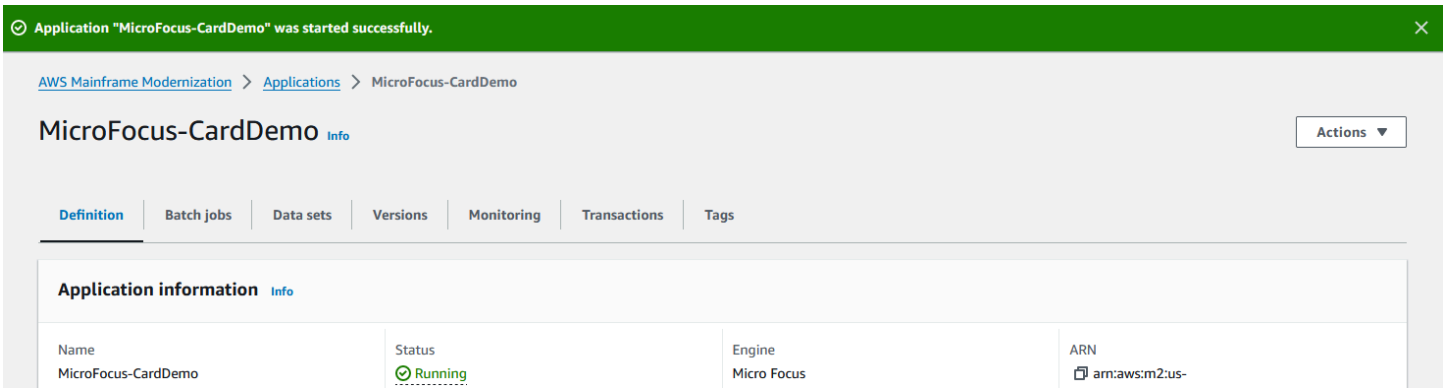
9단계: 애플리케이션 시작

애플리케이션을 시작하려면

1. 탐색 창에서 [Applications] 를 선택한 다음 해당 애플리케이션을 선택합니다.
2. [애플리케이션 시작] 을 선택합니다.



CardDemo 애플리케이션이 성공적으로 실행되기 시작하면 다음 메시지와 함께 배너가 나타납니다Application *name* was started successfully. 상태 필드가 실행 중으로 변경됩니다.



10단계: CardDemo CICS 애플리케이션에 연결

연결하기 전에 애플리케이션에 지정한 VPC 및 보안 그룹이 연결하려는 네트워크 인터페이스에 적용한 VPC 및 보안 그룹과 동일한지 확인하십시오.

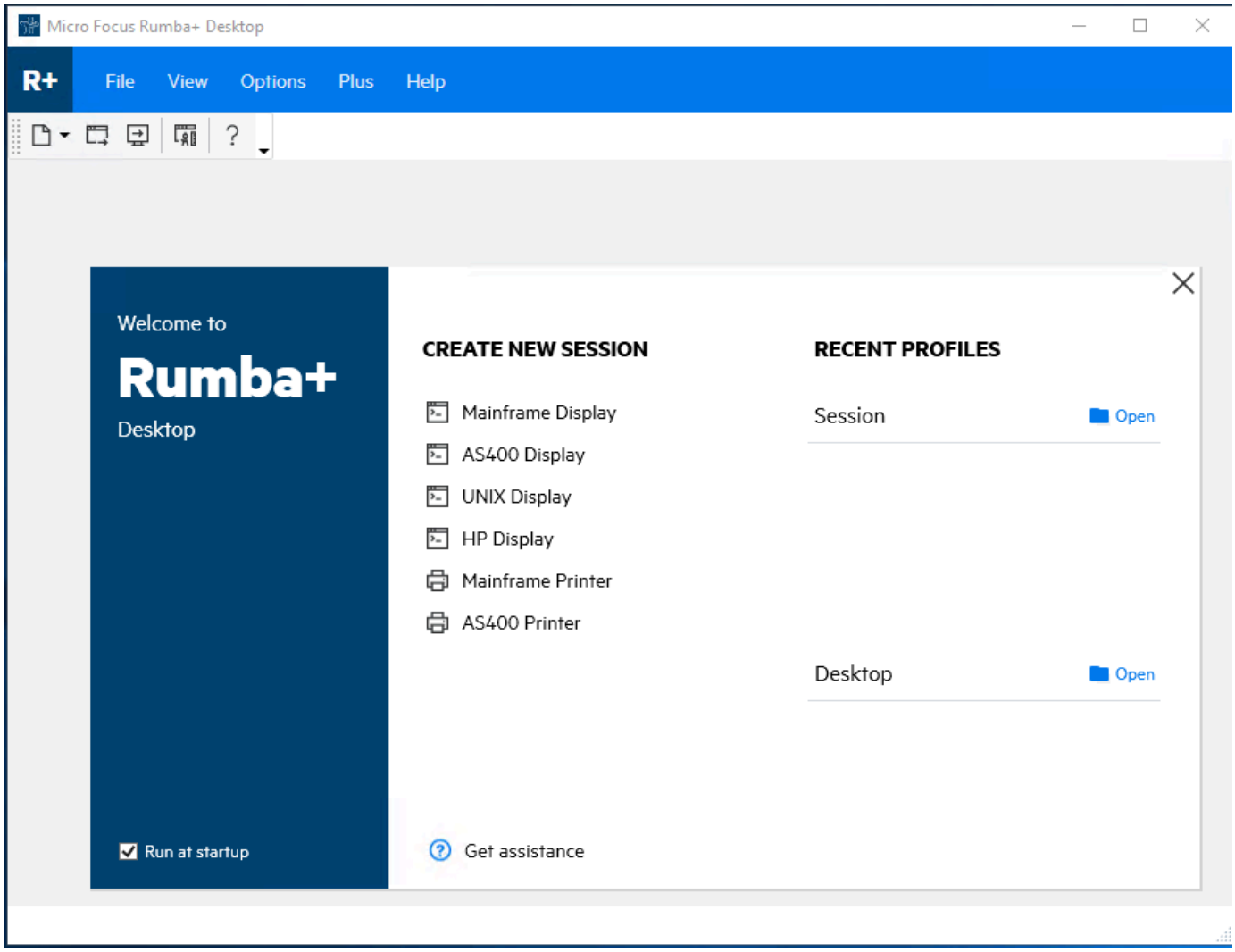
TN3270 연결을 구성하려면 애플리케이션의 DNS 호스트 이름과 포트도 필요합니다.

터미널 에뮬레이터를 사용하여 애플리케이션을 구성하고 메인프레임에 연결하려면

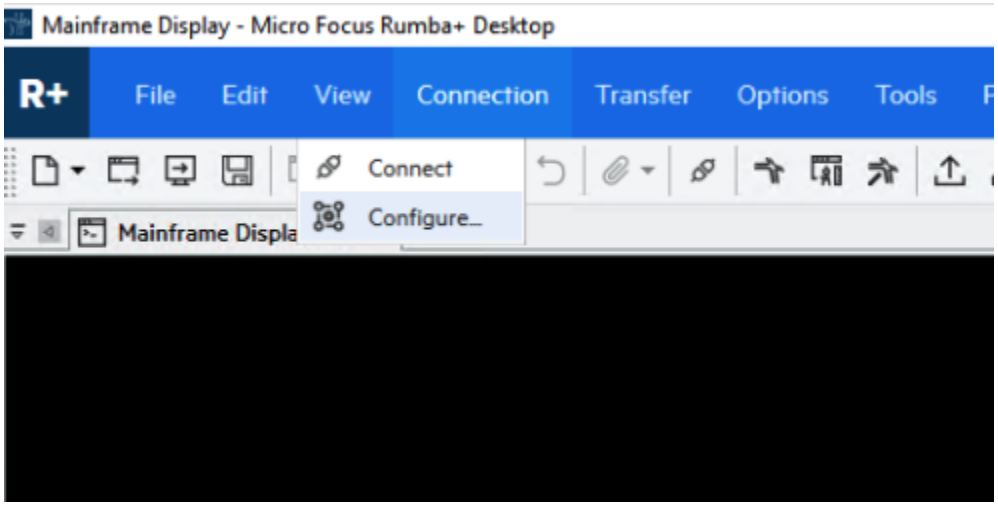
1. AWS메인프레임 현대화 콘솔을 열고 애플리케이션을 선택한 다음 선택합니다. MicroFocus-CardDemo
2. 복사 아이콘을 선택하여 DNS 호스트 이름을 복사합니다. 또한 포트 번호도 기록해 두십시오.
3. 터미널 에뮬레이터를 시작합니다. 이 자습서에서는 Micro Focus Rumba+를 사용합니다.

   > **Note**
   >
   > 구성 단계는 에뮬레이터마다 다릅니다.

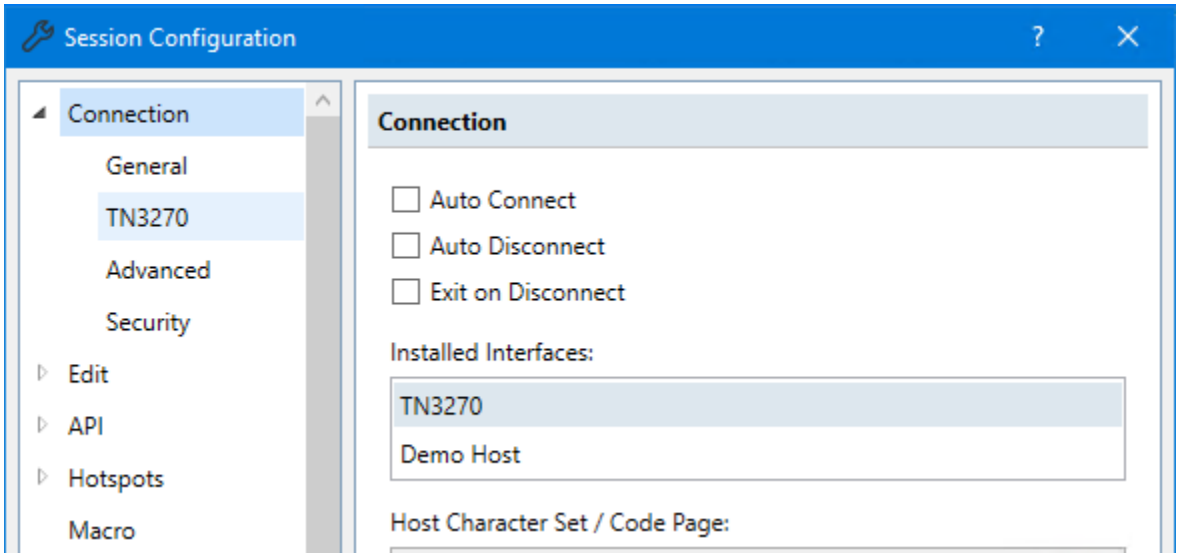
4. 메인프레임 디스플레이를 선택합니다.



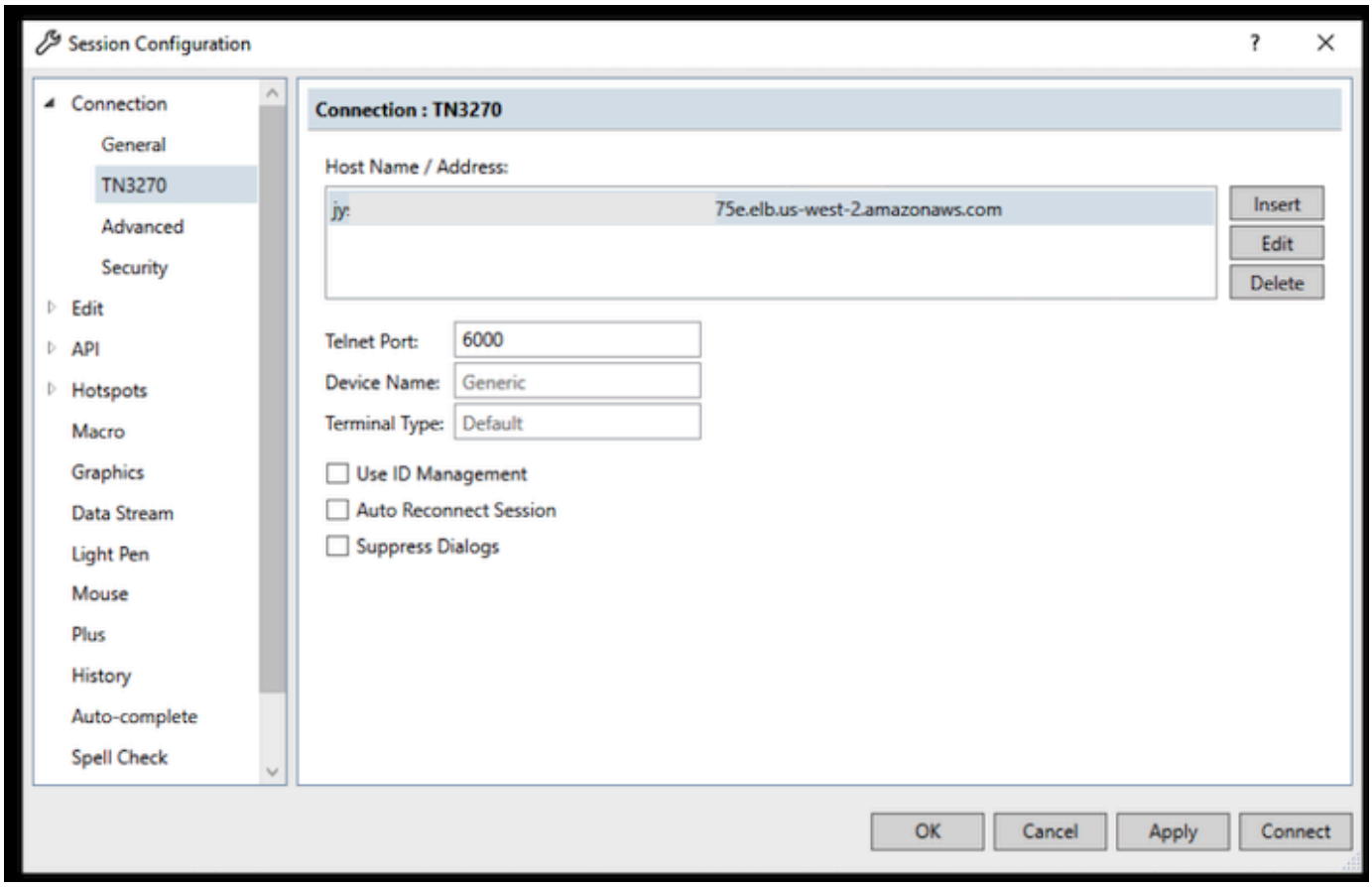
5. 연결을 선택한 다음 구성을 선택합니다.



6. 설치된 인터페이스에서 을 선택한 **TN3270** 다음 연결 메뉴에서 **TN3270** 다시 선택합니다.



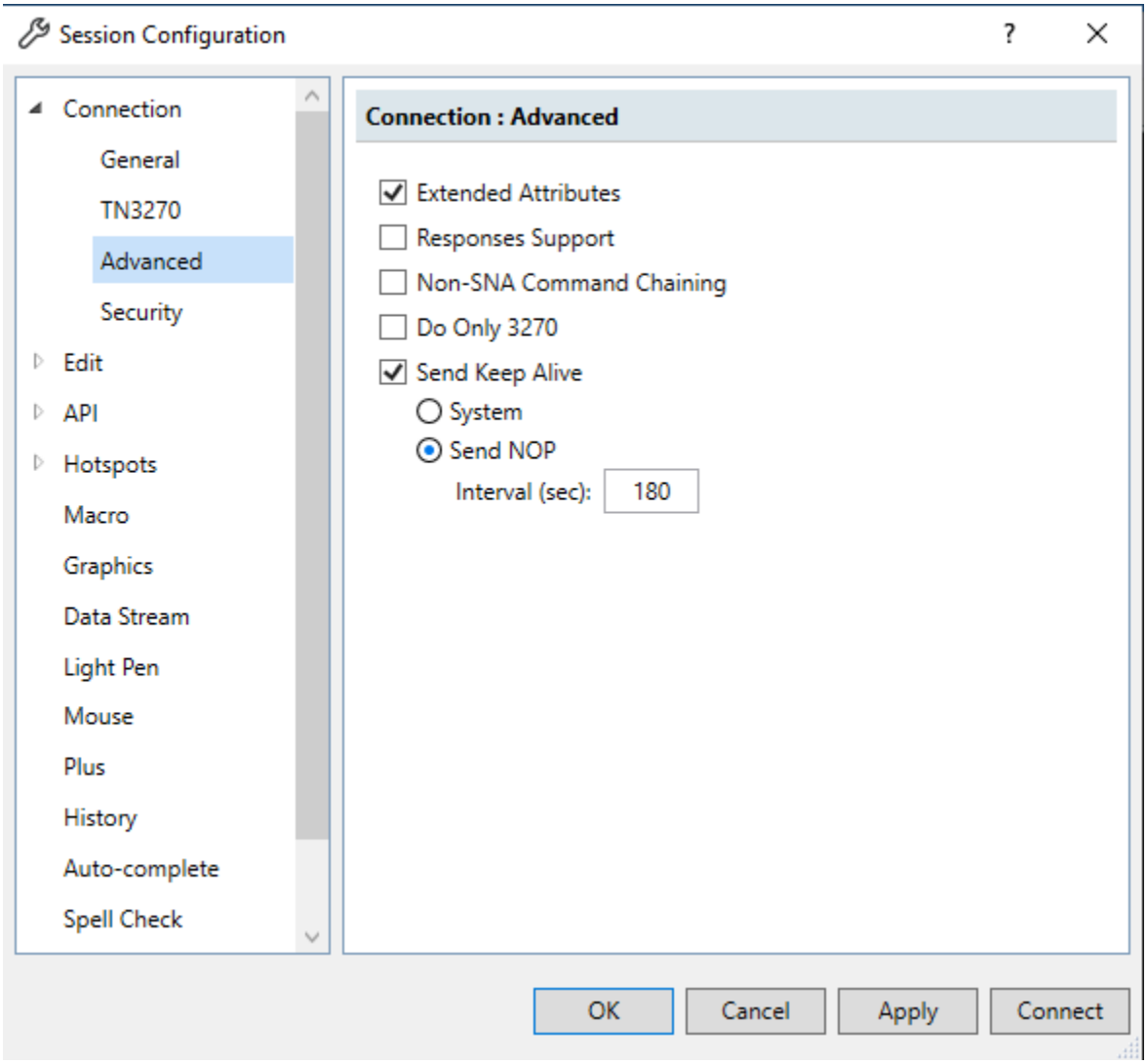
7. 삽입을 선택하고 응용 프로그램에 붙여넣습니다. DNS Hostname 텔넷 포트를 **6000** 지정합니다.



> **Note**
>
> 브라우저에서 AWS AppStream 2.0을 사용하고 있고 값을 붙여넣는 데 문제가 있는 경우 AppStream 2.0 사용자 문제 해결을 참조하십시오.

8. 연결에서 고급을 선택한 다음 [Send Keep Alive] 및 [NOP 보내기] 를 선택하고 간격에 180을 입력합니다.

> **Note**
>
> TN3270 터미널의 keep alive 설정을 180초 이상으로 구성하면 Network Load Balancer에서 연결이 끊기지 않도록 하는 데 도움이 됩니다.



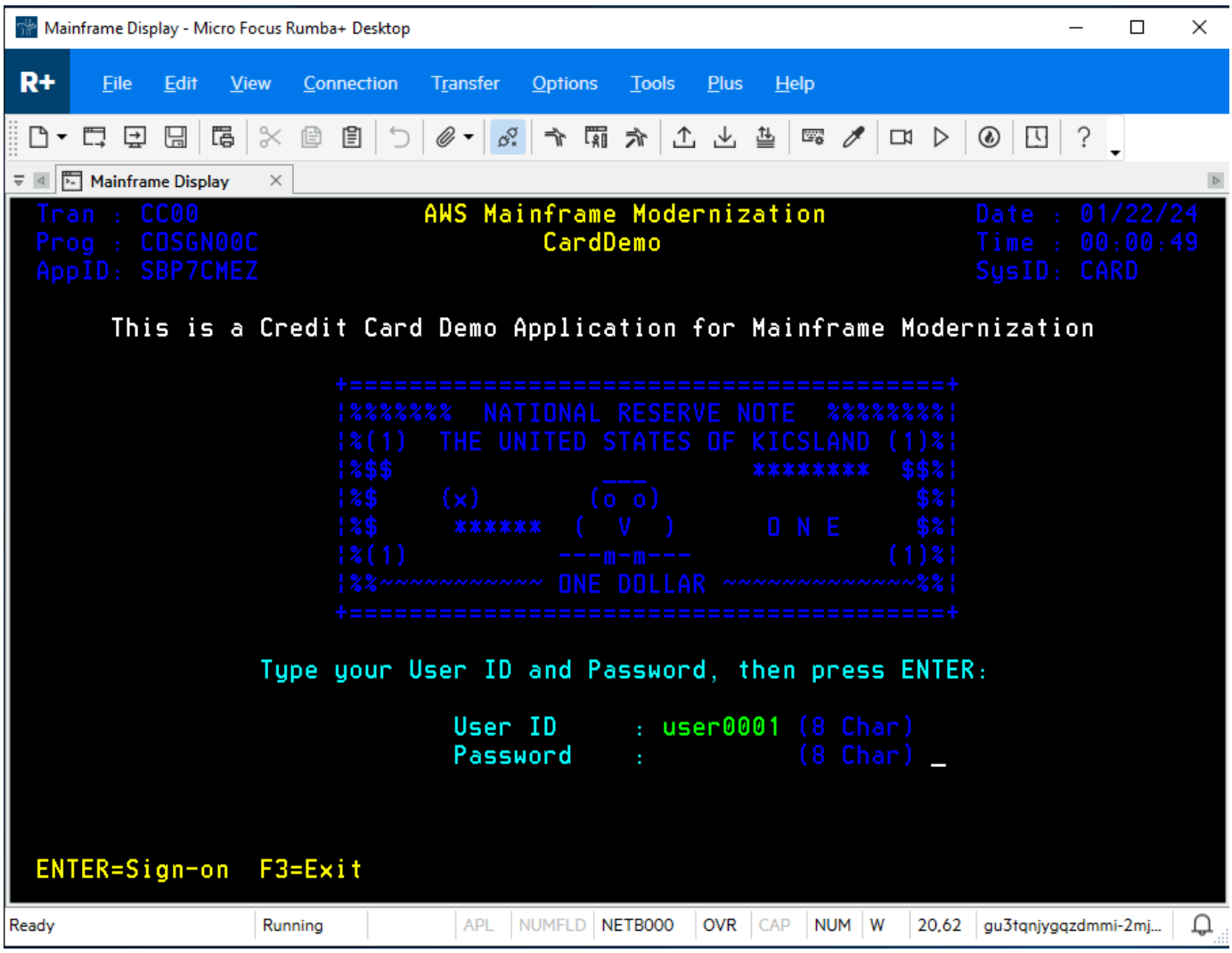
9. 연결을 선택합니다.

> **Note**
>
> 연결이 실패할 경우:
>
> - AppStream 2.0을 사용하는 경우 애플리케이션 환경에 지정된 VPC 및 보안 그룹이 2.0 플릿과 동일한지 확인하십시오. AppStream
> - VPC 도달성 분석기를 사용하여 연결을 분석합니다. 콘솔을 통해 Reachability 분석기에 액세스할 수 있습니다.
> - 진단 단계로, 어디서든 포트 6000에 대한 트래픽을 허용하도록 애플리케이션에 대한 보안 그룹 인바운드 규칙을 추가하거나 변경해 보십시오 (예: CIDR 블록 0.0.0.0/0). 연결

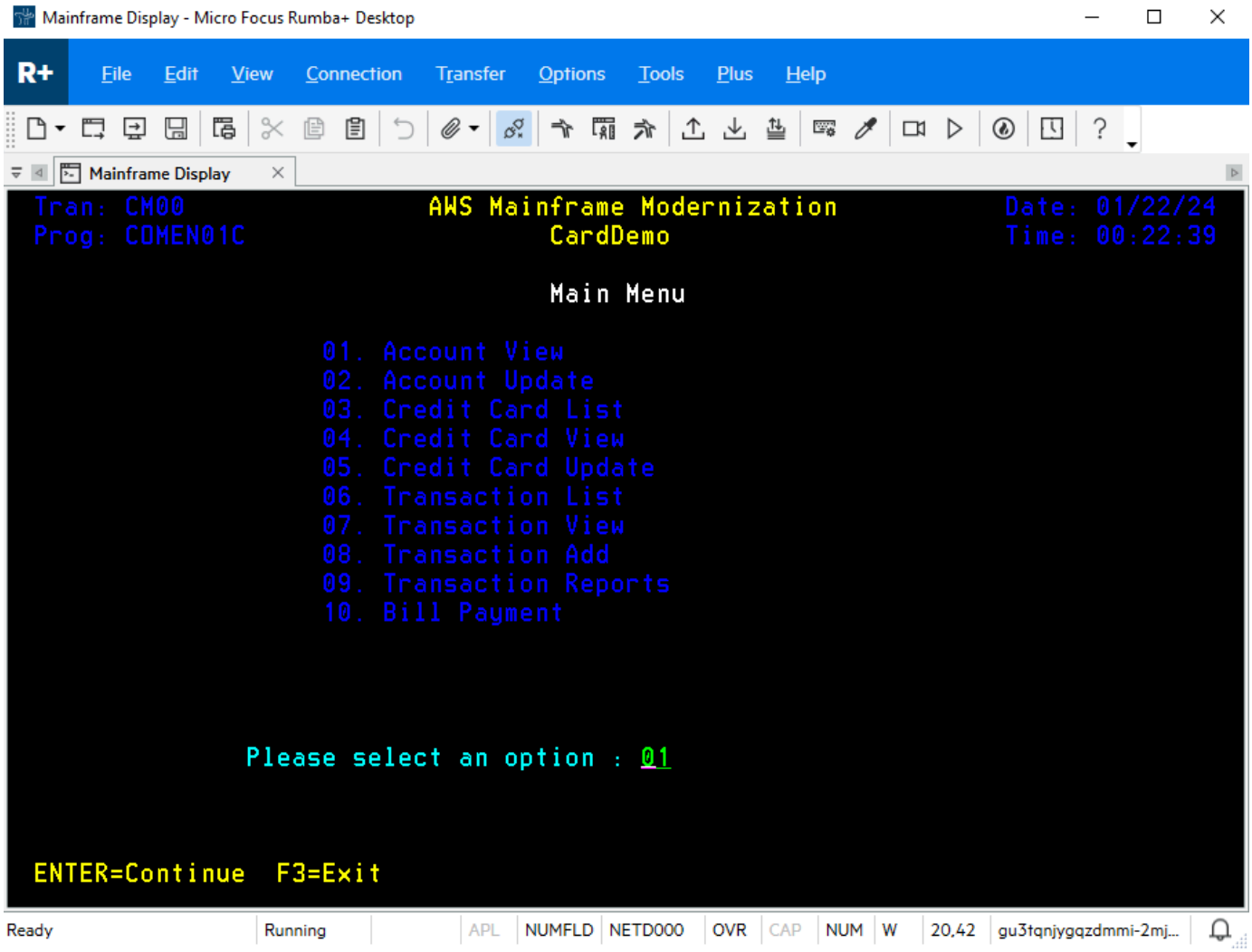
에 성공하면 보안 그룹이 트래픽을 차단하고 있다는 것을 알 수 있습니다. 보안 그룹 소스를 좀 더 구체적인 것으로 변경하십시오. 보안 그룹에 대한 자세한 내용은 보안 그룹 기본 사항을 참조하십시오.

10. 사용자 이름과 암호를 `USER0001` `password` 입력합니다.

> **Note**
>
> Rumba에서 지우기의 기본값은 이고 ctrl-shift-z 재설정의 기본값은 ctrl-r입니다.

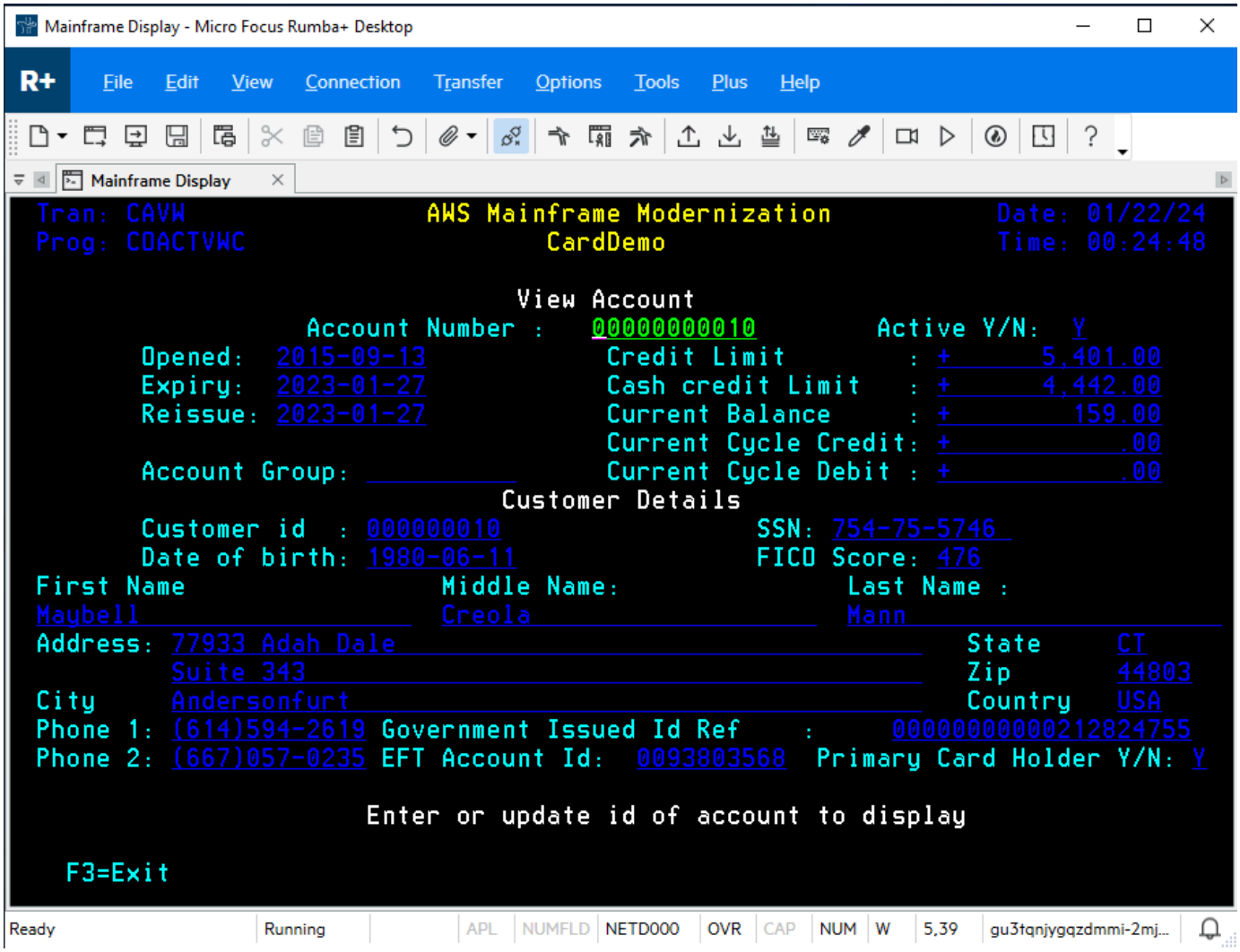


11. 로그인에 성공하면 CardDemo 애플리케이션을 탐색할 수 있습니다.
12. 계정 보기를 입력하려면 를 `01` 입력합니다.



13. 계좌 번호를 입력하고 `00000000010` 키보드에서 Enter 키를 누릅니다.

> **Note**
>
> 다른 유효한 계정은 `0000000011` 및 `00000000020` 입니다.



14. 누르면 F3 메뉴가 종료되고 트랜잭션을 F3 종료하려면 누릅니다.

리소스 정리

이 자습서 용도로 생성한 리소스가 더 이상 필요하지 않은 경우 삭제하여 추가 비용이 발생하지 않도록 하세요. 이렇게 하려면 다음 단계를 완료합니다.

- 필요한 경우 애플리케이션을 중지하십시오.
- 애플리케이션을 삭제합니다. 자세한 설명은 AWS 메인프레임 현대화 애플리케이션 삭제 섹션을 참조하세요.
- 런타임 환경을 삭제합니다. 자세한 설명은 AWS 메인프레임 현대화 런타임 환경 삭제 섹션을 참조하세요.

- 이 자습서를 위해 생성한 Amazon S3 버킷을 삭제합니다. 자세한 내용을 알아보려면 Amazon S3 사용 설명서의 버킷 삭제를 참조하세요.
- 이 자습서를 위해 생성한 AWS Secrets Manager 암호를 삭제하십시오. 자세한 내용은 비밀 삭제를 참조하십시오.
- 이 자습서를 위해 만든 KMS 키를 삭제하십시오. 자세한 내용은 AWS KMS 키 삭제를 참조하십시오.
- 이 자습서를 위해 생성한 Amazon RDS 데이터베이스를 삭제합니다. 자세한 내용은 Amazon RDS 사용 설명서의 EC2 인스턴스 및 DB 인스턴스 삭제를 참조하십시오.
- 포트 6000에 대한 보안 그룹 규칙을 추가한 경우 규칙을 삭제하십시오.

다음 단계

현대화된 애플리케이션을 위한 개발 환경을 설정하는 방법을 알아보려면 자습서: Micro Focus 엔터프라이즈 분석기 및 Micro Focus 엔터프라이즈 개발자와 함께 사용할 AppStream 2.0 설정을 참조하십시오.

현대화 접근법

마이그레이션은 복잡하며 변수가 많습니다. AWS 메인프레임 현대화는 민첩성을 개선하고 나중에 최적화하고 혁신할 수 있는 많은 기회를 제공함으로써 단기적으로 성과를 거둘 수 있는 혁신적인 접근 방식을 제공합니다. 또한 AWS 메인프레임 현대화는 과정을 단순화하면서도 고객의 회사 및 비즈니스의 세부 사항을 존중하는 데 도움이 됩니다. AWS 메인프레임 현대화가 지원하는 두 가지 주요 접근 방식은 자동화된 리팩토링 또는 리플랫포밍입니다. 어떤 것을 선택할지는 고객의 상황에 따라 달라집니다.

자동 리팩토링은 AWS Blu Age 도구를 사용하여 코드, 데이터 및 종속성을 최신 언어, 데이터 스토어 및 프레임워크로 자동 변환하는 동시에 동일한 비즈니스 기능과의 기능적 동등성을 보장합니다.

리플랫포밍은 Micro Focus 도구를 사용하여 AWS에서 메인프레임 워크로드를 애자일 서비스로 전환합니다.

현대화 여정을 단계별로 생각해 볼 수 있습니다. 첫 번째 단계에는 평가, 동원, 마이그레이션 및 현대화의 세 단계가 포함됩니다. 다음 단계에는 더 많은 혁신 기회를 식별할 수 있는 운영 및 최적화 단계가 포함됩니다.

주제

평가 단계

가장 높은 수준에서 평가 단계에서는 마이그레이션할 준비가 되었는지 여부를 살펴봅니다. 비즈니스 사례를 정의한 다음 AWS에서 제공하는 워크숍과 몰입 데이(데모 및 실습)를 통해 팀을 교육합니다. 워크숍과 몰입 데이는 다양한 주제를 다룹니다. 이러한 작업은 AWS 메인프레임 현대화 외부에서 수행됩니다.

동원 단계

동원 단계에서는 프로젝트를 시작하고 메인프레임 애플리케이션에서 데이터를 추출하여 마이그레이션 도구로 수집하는 검색 프로세스를 실행합니다. 마이그레이션할 애플리케이션을 식별하고 파일럿할 애플리케이션 몇 개를 선택합니다. 비즈니스 사례를 구체화하고, 마이그레이션 계획을 작성하고, 보안

및 규정 준수, 계정 거버넌스, 운영 모델을 처리하는 방법을 결정합니다. 팀에서 적합한 인재를 영입하여 최고의 클라우드 센터를 구축하세요. 파일럿을 실행하고 배운 내용을 문서화합니다. 마이그레이션 계획과 비즈니스 사례를 구체화합니다. 이러한 작업의 대부분은 AWS 메인프레임 현대화 외부에서 수행됩니다.

마이그레이션 및 현대화 단계

마이그레이션 및 현대화 단계는 각 애플리케이션에 적용되며 인력 배정, 심층 탐색, 적합한 애플리케이션 아키텍처 파악, 애플리케이션 런타임 환경 설정AWS, 코드 재구성 또는 리팩토링, 다른 시스템과의 통합, 테스트 등 여러 작업으로 구성됩니다. 단계가 끝나면 플랫폼이 변경되거나 리팩토링된 애플리케이션을 프로덕션 환경에 배포하고 AWS의 새 시스템으로 전환합니다. 이러한 작업의 대부분 또는 전부는 AWS 메인프레임 현대화, 다른 AWS 서비스 또는 AWS 메인프레임 현대화가 제공하는 도구에서 수행됩니다.

자동 리팩토링을 사용하려면 Blu Insights를 참조하세요. AWS 이제 싱글 사인온을 AWS Management Console 통해 Blu Insights를 사용할 수 있습니다. 더 이상 AWS Blu Insights 자격 증명을 별도로 관리할 필요가 없습니다. AWS Management Console에서 직접 AWS AWS Blu Age 코드베이스와 변환 센터 기능ㅇ에 모두 액세스할 수 있습니다.

메인프레임에서 AWS으로 데이터를 마이그레이션하려면 AWS SCT 및 AWS Database Migration Service를 사용하는 것이 좋습니다. 자세한 내용은 AWS 스키마 변환 툴 사용 가이드의 AWS 스키마 변환 툴은 무엇인가요? 및 AWS Database Migration Service 사용 가이드의 AWS 데이터베이스 마이그레이션 서비스는 무엇인가요?를 참조하세요.

운영 및 최적화 단계

운영 및 최적화 단계에서는 배포된 애플리케이션을 모니터링하고, 리소스를 관리하고, 보안 및 규정 준수를 최신 상태로 유지하는 데 중점을 둡니다. 또한 마이그레이션된 워크로드를 최적화할 기회를 평가합니다.

개념

AWS 메인프레임 현대화는 메인프레임 워크로드를 마이그레이션, 현대화하고 실행하는 데 도움이 되는 도구와 리소스를 제공합니다. AWS

주제

애플리케이션

메인프레임 현대화에서 실행 중인 메인프레임 워크로드. AWS 일련의 배치 작업, 대화형 트랜잭션(CICS 또는 IMS) 또는 기타 구성 요소가 애플리케이션을 구성합니다. 범위를 정의합니다. CICS 트랜잭션 또는 배치 작업과 같이 워크로드에 필요한 구성 요소나 리소스를 정의하고 지정해야 합니다.

애플리케이션 정의

메인프레임 현대화에서 실행되는 애플리케이션 (메인프레임 워크로드) 에 필요한 구성 요소 및 리소스의 정의 또는 사양 AWS 다양한 런타임 환경으로 표현되는 여러 단계(프리프로덕션, 프로덕션)에서 동일한 정의를 재사용할 수 있기 때문에 애플리케이션 자체에서 정의를 분리하는 것이 중요합니다.

배치 작업

배치 작업은 사용자 개입 없이 실행되도록 구성된 예약된 프로그램입니다. AWS 메인프레임 현대화에서는 배치 작업 JCL 파일과 배치 작업 바이너리가 모두 Amazon S3 버킷에서 준비되며 두 위치 모두 애플리케이션 정의 파일에 제공됩니다. 일괄 작업을 실행하면 AWS 메인프레임 현대화는 다음과 같은 상태 값을 보고합니다.

제출 중

배치 작업이 제출되는 중입니다.

보류

배치 작업이 보류 상태입니다.

발송 중

배치 작업이 발송되는 중입니다.

실행 중

작업이 현재 실행 중입니다.

취소 중

배치 작업이 취소되는 중입니다.

취소됨

배치 작업이 취소되었습니다.

성공

배치 작업 실행이 성공적으로 완료되었습니다.

실패

배치 작업이 실패했습니다.

경고와 함께 성공

사소한 오류가 보고되어 배치 작업 실행이 성공적으로 완료되었습니다. GetBatchJobExecution 응답의 일부로 반환된 작업 조건 코드는 오류의 원인을 나타냅니다.

구성

환경 또는 응용 프로그램의 특성. 환경 구성은 엔진 유형, 엔진 버전, 가용성 패턴, 선택적 파일 시스템 구성 등으로 구성됩니다.

애플리케이션 구성은 정적이거나 동적일 수 있습니다. 정적 구성은 새 버전을 배포하여 애플리케이션을 업데이트할 때만 변경됩니다. 일반적으로 추적 기능을 켜거나 끄는 것과 같은 운영 활동인 동적 구성은 업데이트하는 즉시 변경됩니다.

데이터 세트

애플리케이션에서 사용할 데이터가 들어 있는 파일입니다.

환경

하나 이상의 애플리케이션을 호스팅하기 위해 생성된 AWS 컴퓨팅 리소스, 런타임 엔진 및 구성 세부 정보의 명명된 조합입니다.

메인프레임 현대화

레거시 메인프레임 환경에서 기존 메인프레임 환경으로 애플리케이션을 마이그레이션하는 프로세스입니다. AWS

마이그레이션 여정

레거시 애플리케이션을 마이그레이션하고 현대화하는 end-to-end 프로세스로, 일반적으로 평가, 동원, 마이그레이션 및 현대화, 운영 및 최적화 단계로 구성됩니다.

탑재 지점

시스템 내에 저장된 파일에 대한 액세스를 제공하는 파일 시스템의 디렉터리입니다.

자동 리팩터링

최신 클라우드 환경에서 실행하기 위해 레거시 애플리케이션 아티팩트를 현대화하는 프로세스입니다. 여기에는 코드 및 데이터 변환이 포함될 수 있습니다. 자세한 내용은 AWS 메인프레임 현대화 자동 리팩터링을 참조하세요.

플랫폼 재구성

애플리케이션과 애플리케이션 아티팩트를 한 컴퓨팅 플랫폼에서 다른 컴퓨팅 플랫폼으로 이동하는 프로세스입니다. 자세한 내용은 AWS 메인프레임 현대화 리플랫포밍을 참조하세요

Resource

컴퓨터 시스템 내의 물리적 또는 가상 구성 요소.

런타임 엔진

애플리케이션 실행을 용이하게 하는 소프트웨어.

Blu Age를 통한 AWS 애플리케이션 자동 리팩토링

AWS Blu Age를 통한 자동 리팩토링은 메인프레임 애플리케이션을 마이그레이션하고 현대화하기 위한 솔루션을 제공합니다. end-to-end 프로세스의 두 단계는 다음과 같습니다.

- 인벤토리 분석
- 종속성 분석
- 코드 자동 변환
- 테스트 시나리오 캡처 및 관리

메인프레임 현대화 콘솔에서 싱글 사인온을 통해 사용할 수 있는 Blu Insights 도구에서 이전 단계를 완료할 수 있습니다. AWS Blu Insights에 대한 자세한 내용은 Blu Insights 설명서를 참조하세요.

변환된 소스 코드에 만족하셨다면 이제 다음 단계를 완료할 AWS차례입니다.

- 리팩터링된 애플리케이션을 빌드하고 배포하세요.
- AWS 메인프레임 현대화에서 애플리케이션을 배포하고 모니터링하십시오.

AWS Blu Age Runtime (비관리형) 은 Blu Age 관리형과 함께 제공되는 AWS 메인프레임 현대화 서비스 중 하나입니다. AWS AWS Blu Age Managed를 사용하면 현대화된 애플리케이션을 AWS관리형 환경에 배포하여 환경을 단순화할 수 있으므로 현대화된 애플리케이션을 실행하는 기본 인프라를 관리할 필요가 없습니다. 반면 AWS Blu Age Runtime (비관리형) 을 사용하면 현대화된 애플리케이션을 자체 AWS 계정에 배포하여 자체 인프라를 관리할 수 있습니다. AWS Blu Age Runtime (비관리형) 을 사용하면 현대화된 애플리케이션을 실행하는 데 필요한 모든 기술 구성 요소를 원하는 방식으로 유연하게 운영할 수 있습니다.

AWS 블루 에이지 런타임 (비관리형) 은 Amazon EC2와 Amazon ECS에서 배포할 수 있습니다. AWS Fargate

주제

- AWS 블루에이지 출시 노트
- AWS 블루 에이지 런타임 컨셉
- AWS 블루에이지 런타임 구성 및 구성 파일
- AWS 블루에이지 런타임 API
- AWS 블루 에이지 런타임 (비관리형) 설치

- Blu Age 개발자 IDE로 소스 코드 수정

AWS 블루에이지 출시 노트

이 섹션에는 버전 3.5.0 이후 버전의 AWS Blu Age 런타임 및 현대화 도구의 릴리스 노트가 최신 버전부터 버전 번호별로 정리되어 있습니다.

> **Note**
>
> 이 문서 이전의 릴리스 노트는 Blu Age 딜리버리 서비스에 문의하십시오. AWS 최신 Blu Insights 기능에 대한 자세한 내용은 Blu Insights 릴리스를 참조하세요.

주제

릴리스 노트 3.10.0

AWS Blu Age 런타임 및 현대화 도구의 이번 릴리스는 모든 변환 및 실행 단계에서 성능과 견고성을 높이기 위해 노력하는 제품 전반의 핵심 기본 업그레이드 및 개선에 중점을 두고 있습니다. 이번 릴리스의 일부 주요 기능 및 변경 사항은 다음과 같습니다.

- Java 8에서 Java 17로 버전을 업그레이드하면 보안 및 성능이 향상되고 고객이 최신 언어로 구현된 애플리케이션을 배포 및 실행하고 최신 타사 프레임워크 버전을 사용할 수 있습니다.
- 사용자 또는 작업 간의 대규모 공유 메모리 공간 관리, 애플리케이션 또는 인스턴스 재시작 후 재사용 가능한 데이터 저장에 대한 추가 지원.
- 레코드의 하위 집합을 점진적으로 검색할 수 있는 페이지 매김 메커니즘을 사용하여 Blusam의 대규모 데이터 세트에 더 빠르게 액세스할 수 있습니다.

이 릴리스에 포함된 변경 사항에 대한 자세한 내용은 다음 단원을 참조하세요.

런타임 릴리스 3.10.0

이 런타임은 자바17, 스프링2.7, 앵귤러16을 기반으로 합니다.

zOS

새로운 기능

- Blusam - 페이지를 사용하여 색인을 저장하고 로드하는 페이지 매김 메커니즘을 통해 대규모 데이터 세트에 대한 지원이 추가되었습니다.

개선 사항

- 문자열에서 숫자로의 낮은 우선 순위 변환을 처리하도록 DataUtils .compare를 개선했습니다.
- YML 속성 DataSimplifier를 통해 잘못된 값으로 ByteRange no가 생성되었는지 확인하는 지원이 추가되었습니다. byteRangeBounds확인
- 빈 문자가 있는 a의 초기화를 지원하도록 removeSosi () 를 GraphicAlphanumericType 개선했습니다.
- 작업 운영을 위한 견고성 추가 및 안전한 GDG 상태 읽기
- Blusam - .removeCache () 라는 새로운 메서드를 통해 블루섬 데이터 세트의 Ehcache를 지우는 지원이 추가되었습니다. CoreBluesamManager
- Blusam - 일반 Blusam 데이터 세트의 삭제/이름 변경 동작이 개선되었습니다.

- Redis - 데이터 세트 잠금 해제 및 레코드 잠금 지우기에 대한 지원 강화
- JICS - 실패한 요청에 대한 오류 메시지를 개선했습니다.
- JCL - 점 문자를 기반으로 하는 ControlM 변수 연결에 대한 지원이 추가되었습니다.
- JCL - GDG 파일에 대한 쓰기 고급 (ADV) 에 대한 지원이 추가되었습니다.
- JCL - 모든 GDG 파일을 삭제한 후 현재 세대 번호에 대한 지원 강화
- JCL - 데이터세트 생성 시 카탈로그에서 RDW/RecordSize 읽기에 대한 지원 강화
- JCL - 데이터 출력 레코드 크기로 파일을 열 때 리소스 개체 (from AbstractSequentialFile) 를 업데이트하는 지원이 추가되었습니다.
- JCL - IDCAMS 성능 향상
- JCL - “문자”의 별칭으로 “CHAR”를 추가하여 인쇄 명세서에 대한 지원을 강화했습니다.
- 정렬 - Blusam 고정 길이 데이터 세트에서 가변 길이의 데이터 세트로 복사 작업에 대한 지원이 향상되었습니다.
- SORT - 일부 특정 문장을 처리할 수 있도록 정렬 문법을 개선했습니다.

AS400

새로운 기능

- 사용자 공간 및 관련 API에 대한 지원이 추가되었습니다.
- SNDPGMMSG의 TOMSGQ 파라미터에 대한 지원이 추가되고 메시지 대기열이 구현되었습니다.
- CL - OVRPRTF 명령에 대한 파일 및 SPLFNAME 매개 변수에 대한 지원이 추가되었습니다.
- CL - CPYF 명령으로 해당 파티션 테이블의 라이브러리를 처리할 수 있는 지원이 추가되었습니다.
- CL - CHGCURLLIB 명령 처리 및 쿼리 작성 시 현재 라이브러리 고려에 대한 지원 추가
- CL - cl 명령을 호출 스택 추적의 일부로 처리하기 위한 지원이 추가되었습니다.

개선 사항

- 콜 스택 MessageHandlingBuilder 트레이스 엔트리를 더 잘 처리하도록 개선되었습니다.
- ContextPreConstruct 기능의 Parallel 실행이 개선되었습니다.
- SFLINZ에서 레코드를 생성할 때 표시되는 속성이 개선되었습니다.
- SAVOBJ가 여러 출력 파일을 처리할 수 있도록 개선되었습니다.
- Java 프로그램에서 호출될 programCallStack 때 이를 추가하여 그루비 프로그램 처리를 개선했습니다.

- 도움말 모달의 상단 위치 감지 기능이 개선되었습니다.
- SNDPGM/MSG에 TomSGQ 매개변수가 제공될 때 TopGMQ 기능이 개선되었습니다.
- 사전 정의된 메시지 가져오기 및 메시지 로더 기능이 개선되었습니다.
- 콘텐츠의 구분자에 대한 CPYTOIMPF 처리가 개선되었습니다.
- 읽기 레코드의 릴리스 잠금이 개선되었습니다.

횡단 기능

새로운 기능

- 프론트엔드의 시스템 메시지 번역 추가
- 프로그램 호출 스택을 ExecutionContext 반환하는 새 메서드를 추가했습니다.
- 실제 환경에 관계없이 줄 구분자 설정 (데이터 단순화용)
- SQL 모델 JSON 경로를 구성할 수 있는 기능 추가

개선 사항

- 비교 방법을 DataUtils 개선했습니다. compareAlphInt() 패딩이 포함된 경우
- 커서 쿼리의 예외에 대한 사용자 지정 동작을 허용하는 플래그 생성
- 그래픽 저값 db 변환 개선

써드파티

- CVE-2024-21634, CVE-2023-34055, CVE-2023-34462, IN1-JAVA-ORGSPRINGFRAMEWORKSECURITY-5905484, CVE-2023-46120, CVE-2023-6481, CVE-2023-6378, CVE-2023-5072 등의 문제를 완화하기 위한 업그레이드)

현대화 도구 릴리스 3.10.0

zOS

개선 사항

- 코볼 - ABS 기능에 대한 지원 추가
- JCL - 향상된 변수 범위: JOB 대신 STEP에 첨부

- 저/고 값에 대한 향상된 커서 파라미터 삽입
- 특히 원격 트랜잭션의 경우 CSD 파싱이 개선되었습니다.

AS400

개선 사항

- 제어 수준 표시기의 빈 체크 표시가 제거되었습니다.
- IMPORT/EXPORT 키워드의 외부 이름에 대한 지원이 추가되었습니다.
- 필드에 %LEN 지원 추가
- CL - CLLE 언어의 새 연산자에 대한 지원이 추가되었습니다.
- CL - 중첩된 IF에 대한 지원 추가
- COBOL - 여러 키와 함께 사용할 때 START 명령 처리 기능이 개선되었습니다.
- DSPF - 레코드 번호를 사용한 커서 위치 처리 개선
- DSPF - 부호있는 숫자, 숫자 전용 필드 및 대규모 필드의 서식을 개선했습니다.
- DSPF - 화면 일반 도움말의 제목 결정을 개선했습니다.
- DSPF - 입력/출력 사양 지원 개선
- DSPF - 숫자 필드 검증 시 그룹화 구분 기호 처리 개선
- 출력/DDS 레코드 매핑 개선
- 참조 필드를 확인할 수 있는 프린터 파일 REFFLT 키워드 기능이 개선되었습니다.
- RPG - “ALL FREE” 문장에 대한 지원 강화
- RPG - 조건 파싱 개선 및 결과 태그 없이 CABXX를 처리할 수 있는 지원 추가
- RPG - 숫자 필드의 입력 사양 처리 개선
- RPG - IF/ELSEIF/WHEN 조건 내에서 프로시저 호출 처리를 개선했습니다.
- RPG - dspf 파일에서 호출 시 READ 명령 처리 개선
- RPG - 존재하지 않는 DDS를 참조하는 파일에 대한 지원을 개선합니다.
- 물리적 레코드 형식 이름 전달 시 REFFLD 처리 개선
- 'return'을 db 열 이름으로 사용하는 지원이 추가되었습니다.

횡단 기능

새로운 기능

- Oracle - 내장 함수를 저장하기 위해 SYS보다 사용자를 정의할 수 있게 되었습니다.

개선 사항

- Java 버전을 v8에서 v17로 업그레이드했습니다.
- 클러스터 열 이름의 SQL 조건 개선
- 뷰에서 ORDER BY 절에 대한 지원이 추가되었습니다.

릴리스 노트 3.9.0

이번 AWS Blu Age Runtime and Modernization Tools 릴리스는 작업 실행을 한 단계 끌어올리는 새로운 기능과 함께 고가용성 아키텍처의 성능을 높이기 위해 노력하는 제품 전반의 여러 가지 개선 사항에 초점을 맞추고 있습니다. 이번 릴리스의 일부 주요 기능 및 변경 사항은 다음과 같습니다.

- Angular 13에서 Angular 16으로의 버전 업그레이드를 통해 보안이 강화되고 고객의 온라인 애플리케이션 성능을 개선하는 최신 기능이 지원됩니다.
- AS400에는 교차 작업 지원 기능이 추가되어 작업 간에 문의 메시지를 동시에 보낼 수 있어 현대화된 작업에서 디커플링이 가능해진다는 점이 주요 특징입니다.
- 연결 풀 최적화, 연결 보안 강화, 데이터 세트 잠금 메커니즘 업그레이드 등 Redis의 사용 성능이 개선되었습니다.

이 릴리스에 포함된 변경 사항에 대한 자세한 내용은 다음 단원을 참조하세요.

런타임 릴리스 3.9.0

zOS

새로운 기능

- 정렬 프로그램: VSAM 입력을 고정 길이로 업데이트
- JHDB DB: 구성 가능한 제한 시간 추가

개선 사항

- 파일 연결에 사용되는 경우 스트림에 줄 구분자 지원이 향상되었습니다.
- 연결된 순차 파일 열기 지원이 향상되었습니다. 파일을 연 DataSetIndex 후 초기화

- NumericEditedType a가 숫자 값에 영향을 받는 경우 가상 십진 구분 기호에 대한 지원이 향상되었습니다.
- 음수 값이 없는 경우에 대한 지원이 향상되었습니다. NumericEditedType
- IDCAMS: 이제 .yml에 정의된 “인코딩” 속성을 사용하여 SYSIN 카드를 읽을 수 있습니다. application-utility-pgm
- IDCAMS: DEFINE CLUSTER 문에서 FILE(..) 인수를 지원하도록 문법이 업데이트되었습니다.
- INFUTILB: DD SYSREC의 DCB 파라미터를 재정의하는 DFSIGDCB 인수에 대한 지원이 추가되었습니다.
- INFUTIL: 'DFSIGDCB YES' 파라미터에 대한 지원이 향상되었습니다.
- 대용량 입력 파일을 처리할 수 있도록 SPLICE가 개선되었습니다.
- DFSORT: 비고 필드 처리가 개선되었습니다.
- DFSORT: (서명된/서명되지 않은) 자유 양식 숫자 형식(SFF/UFF)에 대한 지원이 추가되었습니다.
- 정렬: OPTION PRINT 및 OPTION ROUTE 문에 대한 구문 분석 지원이 추가되었습니다.
- SORT/ICEMAN: 묶인 나누기 연산에 대한 지원(DIV 연산자가 있는 필드)이 추가되었습니다.
- 일반 키를 사용한 CICS READ 지원이 향상되었습니다.
- StringUtils.chargraphic 함수가 그래픽 유형에서 SOSI를 제거하도록 수정되었습니다.
- 에서 성능을 향상시키세요. DataUtils isDoubleByte인코딩
- JCL: 임시 데이터 세트의 KEEP 처리 모드에 대한 지원이 향상되었습니다. 시스템이 처리를 PASS로 변경합니다.
- JCL: DCB 파라미터를 동적으로 처리합니다.
- JCL: 잘못된 값에 대해 SUM FIELDS 출력이 향상되었습니다.
- JCL: CommonDDUtils::getContent는 이제 카탈로그에서 recordSize를 검색합니다.
- JCL: 데이터 세트 생성 시 카탈로그에서 rdw/recordSize 속성을 읽습니다.
- JCL: 동일한 작업 단계에서 DD의 DCB 파라미터를 다른 DD로 복사할 수 있는 DCB=.MYDD에 대한 지원이 추가되었습니다.
- JCL: 레코드 크기 상속 시스템이 개선되었습니다.
- JCL: (Redis) 전용 데이터세트 잠금 추가
- Redis: 독립형 모드에 대한 SSL 지원이 추가되었습니다.
- Redis: 동기화된 Redis 잠금 수(잠금 포함)가 추가되었습니다.
- Redis: Redis 잠금에 대해 Pool 파라미터가 지원됩니다.

- Redis: Redis를 사용하여 메타데이터 새로 고침이 최적화되었습니다.
- Redis: Redis 클러스터 지원이 개선되었습니다.
- IO 모드의 열린 잠금이 개선되었습니다.
- 데이터 세트 잠금 성능이 개선되고 미사용 잠금이 삭제되었습니다.
- 파일 등록 취소 시 데이터 세트 경로가 개선되었습니다.
- 미리 가져오기 기간 캐시 무효화가 개선되었습니다.
- 스레드 안전 유틸리티 데이터 소스 공급자 사용에 대한 지원이 추가되었습니다.
- datasetState 무효 검사가 향상되었습니다.
- 이미 연 데이터 세트를 다시 열지 않도록 지원이 향상되었습니다.
- 작업 최종 연산을 위한 견고성이 추가되었습니다.
- 중복이 허용되는 키의 인덱스 순서 지원이 향상되었습니다.
- 건너뛰기 목록 직렬화 순서에 대한 지원이 향상되었습니다.
- 인덱스 순서 문제를 진단하는 데 도움이 되는 디버그 덤프 기능에 대한 지원이 추가되었습니다.
- 메타데이터 새로 고침에 대한 지원이 향상되었습니다.
- Blusam 대량 읽기 지원 강화

AS400

새로운 기능

- 애플리케이션 컨텍스트 레지스트리 생성
- DSPF 키워드 CLRL(NO)과 레코드 잠금 모니터링 지원
- 키드 지원 DataQueue
- 배치 작업에 대한 INQUIRY 메시지 지원
- AS400 COBOL용 프로그램 설명 프린터 파일 지원 추가
- RMVJOBSCDE cl 명령 처리
- RUNSQL/DLYJOB 개선
- CHKOBJ: 파라미터 LIB에 대한 레거시 오류 코드 발생
- SNDPGMMSG: 문자열 파라미터 지원
- RTVDTAARA: LDA의 개선된 하위 문자열
- DSPFD: 특정 파일 이름에 지원되는 FILE 파라미터 추가

- RUNQRY: QRY PARAM의 sql 파일 지원
- CRTDUPOB: 데이터 영역 간 데이터 복사 지원
- SBMJOB: 명령어를 사용하도록 변환합니다. JobQueueManager
- OPNQRYF: Qtemp 라이브러리에 대한 지원 추가
- CRTDUPOBJ: 파티션 콘텐츠 복사 로직 개선
- CRTDUPOBJ: 뷰에 대한 Qtemp의 지원 추가
- RTVSYSVAL: CL 명령의 SYSVAL 값, QDATFMT 지원
- CHKOBJ: OUTQ에 대한 지원 추가
- RTVJOBA: SWS 파라미터 지원
- SNDPGMMSG 및 RCVMSG: MSGF, MSGFLIB, MSGDTA, MSGTYPE, KEYVAR, MSGKEY, MSGID를 지원하는 추가 파라미터

개선 사항

- WORKSTATION I/O 카드 지원이 개선되었습니다.
- 이전 메시지를 오버레이하는 메시지 설정 처리가 개선되었습니다.
- 배열 메시지 라인에 대한 추가 메시지 정보를 지원합니다.
- EVAL, SortA, 비유법 내부의 독립형 배열 래퍼 액세스가 개선되었습니다.
- 온라인 신청 종료 시 DAO 정리를 개선합니다.
- 추가 날짜 형식에 대한 지원이 추가되고 문자열 입력 처리가 개선되었습니다.
- CL 명령에서 시스템 값 도우미 클래스 디코딩 및 빌드 매개변수를 추가하여 SYSVAL의 CVTDAT 처리를 개선했습니다. SbmJob
- 구성 요소 스캔에서 gapwalk-cl-command com.netfective.bluage.gapwalk.rt.blu4iv 패키지를 제거했습니다.
- 메시지 대기열 API에 대한 사전 정의된 메시지 지원이 개선되었습니다.
- 다른 프로그램에서 작성한 for 레코드에 대한 지원이 개선되었습니다. retrieveSubfileRecord
- 메시지 대기열 API에 대한 즉각적인 메시지 지원이 개선되었습니다.
- 작업 제출 시 로컬 데이터 영역 처리가 개선되었습니다.
- 서버가 시작되면 JobQueues 자동으로 시작됩니다.
- applicationContext 구성을 사용하여 SBMJOB의 파라미터를 디코딩합니다.
- 시스템 제공 오류 메시지가 개선되었습니다.

- RTVMSG가 중첩된 하위 디렉터리에서 .properties 파일을 검색할 수 있도록 합니다.
- 잘못되거나 유효하지 않은 포인터에 바인딩된 엔터티의 재설정을 처리합니다.
- RCVMSG의 MSGID 및 MsgFile 이름을 문자열로 MessageHandlingBuilder 표시하도록 개선되었습니다.
- 메시지 큐 API의 withMsgFile 이름 메서드가 개선되었습니다.
- 데이터 영역 잠금 메커니즘이 개선되었습니다.
- RTVMBRD: 파라미터 FILE의 소문자 및 대문자를 지원합니다.
- CRTDUPOBJ: 뷰 처리가 개선되었습니다.
- CPYTOSTMF: 연결 처리가 개선되었습니다.
- CPYF: 플랫 파일에서 복사하는 동안 디렉터리 이름을 처리하는 기능이 개선되었습니다.
- RCVF: Groovy와 Java의 DEV/RCDFMT 파라미터와 RCDFMT 변환을 올바르게 처리합니다.
- RCVF: 후속 호출을 처리하고 커서 재설정을 방지합니다.
- CPYF: 플랫 파일에서 쓰기 지원을 추가했습니다.
- CRTDUPOBJ: Qtemp 라이브러리를 사용하여 새 obj 처리를 추가했습니다.
- CHGDTAARA: 데이터 영역 최대 길이를 256에서 2,000으로 늘렸습니다.
- SAVOBJ: 저장된 레코드가 삽입 순서대로 되어 있는지 확인합니다.
- RTVDTAARA: 값이 검색됩니다(잘려지지 않음).
- CHKOBJ: 멤버가 존재하지 않는 경우 올바른 모니터 메시지를 반환합니다.
- RTVDTAARA: LDA 하위 문자열에 대한 지원을 추가합니다.
- RTVDTAARA: RTNVAR 파라미터에 지정된 변수 길이까지의 공백을 반환합니다.
- RTVDTAARA: 시작 및 길이에 대한 정수 파라미터와 최신 변환 형식을 지원합니다.
- CHGDTAARA: 하한과 상한을 포함하는 파라미터에 대한 지원을 추가합니다.
- CHKOBJ: 파라미터 객체 유형의 VIEW 값을 처리합니다.
- CHKOBJ: 뷰가 있는 경우 멤버에 관계없이 결과가 true로 설정됩니다.

횡단 기능

새로운 기능

- .txt 파일에 대한 보고서 생성 처리
- Secrets Manager에 currentSchema XA 데이터 소스 속성 추가

- 이미 열려 있는 커서가 열려 있을 때 프레임워크가 SQLCODE 오류 502를 유발하도록 database.cursor.raise.already.opened.error yml 속성 추가

개선 사항

- Amazon EC2 패키징의 AWS 블루 에이지에 갭워크 폼을 추가했습니다.
- 기본적으로 새 신호 처리기 패러다임을 사용합니다.
- 처리가 MOD 또는 OLD인 경우 잠금 지원을 추가합니다.
- 데이터베이스 날짜 시간 패턴을 저장하기 위한 캐시가 추가되었습니다.
- 의 검사 기능이 개선되었습니다. PackedType
- 다음과 같은 DataUtils 레코드의 .setTo 기능을 개선하십시오. VariableSizeArray
- 실행 유닛과 관련하여 MQ SYNCPOINT 옵션을 처리합니다.
- 프레임워크가 롤백 트랜잭션에 SQLCODE를 설정할 수 있도록 활성화했습니다.
- 엔진 키 암호에 따른 자동 드라이버 클래스 이름을 추가했습니다.
- 프로그램/트랜잭션 제한 시간
- 커서 접근 시 롤백 후 커서 위치를 복원했습니다.

써드파티

- SnakeYAML, 레디슨 및 Amazon SDK 업그레이드, YamlBeans 제거 (CVE-2022-25857, CVE-2023-24621, CVE-2023-42809, CVE-2023-44487 완화)

현대화 도구 릴리스 3.9.0

zOS

개선 사항

- 문자열 유형 대상의 소스로서 XML-TEXT에 대한 지원이 향상되었습니다.
- X/(Y/Z) 분할 패턴을 지원하도록 STM에서 UML로의 워크플로가 개선되었습니다.
- JHDB DB: 데이터베이스 업데이트 전에 ROLLBACK 호출을 수락합니다.
- JHDB DB: 트랜잭션이 종료된 경우에도 ROLLBACK을 수락합니다(NOP).
- JCL: 단계 검증 기능이 개선되었습니다.

- SORT: 영역 소수점 음수 값으로 SUM 함수를 처리합니다.
- COBOL: 문자열 리터럴에서 작은따옴표/큰따옴표 이스케이프에 대한 지원을 추가합니다.

AS400

개선 사항

- 내장 함수 %editc의 앞에 0을 추가하여 편집 코드 X의 처리가 개선되었습니다.
- 입력 전용 필드 초기 값 처리가 개선되었습니다.
- 대화에 도움이 되는 작업 키가 추가되었습니다.
- 하단에 나타나는 동적 표의 바닥글을 기록합니다.
- 실제 RECORD-KEY를 지정하는 파일에 대해 KEY PHASE 없이 START 명령을 처리했습니다.
- 플로트 및: :pow 유형에 대한 기본값 추가 NumberUtils
- LIKE(IN)를 사용하여 변수를 정의하는 지원이 추가되었습니다.
- 선택적 요소 생략을 지원하도록 FOR 루프 처리가 업데이트되었습니다.
- 레코드를 CTDATA 배열 이름과 연결하도록 RPG 구문 분석이 업데이트되었습니다.
- CABxx 문의 지표 처리가 개선되었습니다.
- COMMIT 키워드의 선택적 파라미터를 지원합니다.
- LF의 FORMAT 키워드 지원이 개선되었습니다.
- 높음 및 같음(또는 낮음 및 같음) 표시가 있는 LOOKUP 작업 코드가 관리됩니다.
- 큰따옴표로 선언된 PF 키 이름을 처리했습니다.
- 선행 0이 표시되도록 EDTCDE X 처리가 개선되었습니다.
- 이름이 지정되지 않은 레이블이 생성되지 않는 프린터 파일의 MSGCON 지원이 개선되었습니다.
- CONTENT 필드가 여러 데이터 구조에서 공유됩니다.
- ERRSFL 파라미터를 SFLMSG/SFLMSGID와 조합하여 처리했습니다.
- 완전 무료 RPG의 proc 선언 전 기본 코드 범위가 개선되었습니다.
- 구문 분석 조건 제어 사양이 추가되었습니다.
- 데이터 홀더 매퍼의 setErrSfl () 메서드에 대한 지원이 개선되었습니다.
- 내부적으로 생성된 변수에 대한 유형 확인이 개선되었습니다.
- Z-ADD 옵코드에 대한 지원이 개선되었습니다.
- DFT 값이 있는 상수 필드 처리가 개선되었습니다.

- 프로그램 상태 ds 내의 정수 필드 지원이 개선되었습니다.
- ENTRY 파라미터의 표시기 할당을 처리했습니다.
- ref/reffield 키워드를 통해 전파되는 키워드의 필터가 개선되었습니다.
- 이름 없는 데이터 구조 지원 DataArea
- 포인터 데이터 유형 처리가 개선되었습니다.
- LIKE 키워드로 변수를 정의하는 데 사용되는 처리된 배열 요소가 출력 필드의 배열 액세스를 지원합니다.
- 숫자만 표시하는 부호 있는 숫자에 대한 지원이 개선되었습니다.
- O 카드의 논리적 관계가 지원됩니다.
- 영숫자로 %CHAR의 케이스를 테스트합니다.
- 제어 사양 키워드 main이 지원됩니다.
- 프린터 파일에 2개의 파라미터가 있는 EDTCDE
- FullFreeRPG 파싱 개선
- 바닥글이 올바르게 배치되도록 동적 테이블이 향상되었습니다.
- ALL 비유성 상수를 사용하여 숫자 유형을 초기화하는 지원이 추가되었습니다.
- 동일한 물리적 파일을 참조하는 여러 RPG 논리 파일의 처리가 향상되었습니다.
- 최신 화면에서 수정된 필드를 더 잘 감지할 수 있습니다.
- 동적 필드와의 모달 동기화가 지원됩니다.
- 부호 있는 출력 전용 숫자 필드의 처리가 개선되었습니다.
- WORKSTATION I/O 카드 지원을 개선했습니다.

횡단 기능

새로운 기능

- 데이터 마이그레이션 도구: 바이트를 읽을 때 ebcdicFilesWith VarcharIn VARCHAR 2바이트 길이를 고려할 수 있도록 VB 속성을 추가했습니다.
- 오류를 기록하는 공통 API를 구현했습니다.
- COBOL2 모델, RPG, 정의 2 모델 BluAgeErrorDictionaryUtils 및 에 오류 및/또는 정보를 기록하기 위한 공통 API의 구현 및 사용 CycleBuilder FieldsProcessor
- 다양한 격리 조항 정의를 지원하도록 SQL 문법이 개선되었습니다.

개선 사항

- Angular 버전을 v16으로 업그레이드했습니다.
- Angular: ajv 버전을 6에서 8.9로 업그레이드했습니다.

써드파티

- Groovy를 버전 2.4.15로 업그레이드했습니다.

릴리스 노트 3.8.0

AWS Blu Age Runtime and Modernization Tools의 이번 릴리스는 캐싱 성능 개선 및 단일 배포판 내 명령 지원 통합과 함께 제품 전반의 여러 횡단 기능 개선에 중점을 두고 있습니다. 이번 릴리스의 일부 주요 기능 및 변경 사항은 다음과 같습니다.

- Spring 2.5에서 Spring 2.7로 버전을 업그레이드하여 플랫폼의 유지 관리 지원, 성능 및 보안을 강화합니다.
- 이전에 CL 스크립팅을 사용하던 현대화된 애플리케이션의 사용 및 배포를 용이하게 하기 위해 over-the-counter 배포판의 일부로 82개 이상의 CL 명령을 통합할 수 있습니다.
- 관리 서비스로 통합 가져오기, 데이터 세트 메타데이터 정보 나열 기능 등 BluSAM 데이터 세트를 더 잘 운영하고 상호작용하는 데 사용할 수 있는 새로운 API가 있습니다.
- 클러스터 모드에서의 가용성, 고가용성 데이터 검색, 비밀 사용 표준화 등 성능 개선 및 Redis의 사용 확장.

이 릴리스에 포함된 변경 사항에 대한 자세한 내용은 다음 단원을 참조하세요.

런타임 릴리스 3.8.0

zOS

새로운 기능

- 키 정의를 문자열로 처리 DynamicFileBuilder
- DFSORT: OUTFIL TRAILER1 + DFSORT 문법 초기화에 다중 항목에 대한 지원이 추가되었습니다
- CommonDDUtils 툴: 인스트림 데이터의 레코드 크기 처리
- 인덱스 파일: GENKEY 옵션 처리

개선 사항

- 별도의 JAR에 내장된 외부화된 BluSam 로딩 서비스
- 임시 파일 저장을 위한 위치 설정 지원이 추가되었습니다.
- 다중 노드 케이스의 공유 캐시 메커니즘 개선
- 공유 캐시 사용: IDCAMS 검증 최적화
- 임베디드 셀렉트를 위한 ROWID 인젝션 개선
- JCL: 이제 각 인스트림 작업 절차가 별개의 Groovy 파일로 생성됩니다.
- IDCAMS JCL 카드의 커버리지가 card-demo-v 2개인지 확인하십시오.
- BlusAM: 여러 인스턴스를 사용할 때 중복된 워밍업을 피하세요
- 캐시 하이드레이션 시 메모리 사용량 감소
- Jedis 풀 구성 지원
- 파일 연결에 사용할 경우 스트림에 줄 구분자를 추가했습니다
- IDCAMS 유틸리티에서 EBCDIC 카드 지원 + 블록 코멘트(/* ... /)
- 데이터베이스 지원 쿼리: level49를 SQL로 변환할 때 2바이트 문자열 지원
- DFSORT 문법: 17개의 제어문 구현 + 그 중 2개를 통합(OMIT/INCLUDE)
- 그래픽 개선 INFUTILB를 가져올 수 있습니다
- 가변 크기 테이블로 파일 읽기 지원
- 마지막 바이트의 첫 번째 비트가 'E'인 경우 ZonedType 니블 부호가 있는 경우 지원
- DFSORT/ICETOOL은 레코드가 CHANGE 찾기 상수와 일치하지 않는 경우 NOMATCH=(..) 인수에 대한 지원을 추가합니다
- Redis 클러스터 호환성
- Groovy 종료 코드를 기반으로 한 작업 상태(실패) 처리
- CICS SYNCPOINT ROLLBACK 지원이 개선되었습니다.
- 미리 가져오기 기간이 Redis 캐시 사용을 최적화합니다.
- JCL/GROOVY: DISP=(,PASS)일 때 이전 단계의 데이터 세트에서 isRDW 속성을 상속
- 가변 크기 배열을 사용하여 데이터의 부분 사본 처리

AS400

새로운 기능

- 디스플레이 파일용 I/O 카드 지원
- DSPF 키워드 ERRMSGID 및 CHKMSGID에 대한 추가 메시지 정보 지원
- 프런트 엔드 화면의 여러 오류 메시지 지원
- 응용 프로그램 내 82개 CL 명령에 대한 지원이 추가 또는 개선되었습니다. gapwalk-cl-command

개선 사항

- 커밋 컨트롤 하에서 DELETE 및 READ에 대한 지원 개선
- ConvertDate 내장 %dec 내부
- 시행된 XSS 보안 헤더
- STM 생성의 견고성 및 일관성 향상(자유 형식 RPG의 연속 줄, 소수점 부분에 대한 쉼표, 정의/선언의 자유 형식 블록 처리 개선)
- DataHolderMapper 향상된 생성력
- 견고성 추가 및 범위 변경 DataAreaFactory
- 탭 키의 초점 이동 개선
- Jasper 보고서 생성 성능 개선
- 0초 패딩으로 10진수 표시 개선
- INFDS의 ROW/COL 필드에 대한 지원 개선
- 화면에서 수정된 필드에 대한 지원 개선
- 생성된 보고서 이름 및 경로에 대한 게터 추가
- 데이터 큐 길이 개선
- Spring Boot 2.7의 새로운 표준에 맞게 작업 대기열의 자동 구성 개선
- 여러 개의 동시 세션을 위한 워크스테이션 업데이트 개선

횡단 기능

새로운 기능

- Packed에 대한 잘못된 데이터 허용 오차 없음에 대한 지원
- 데이터 세트 엔드포인트를 나열하기 위한 페이지 매김/필터링 추가

개선 사항

- 빈 문자열과 열 비교의 향상된 ORACLE 쿼리 변환 전략
- DSNTEP 및 INFUTILB 유틸리티 프로그램을 사용하여 BLOB DB2를 처리합니다. BLOB DB2는 이제 BYTEA 유형 포스트그레스로 현대화되었습니다.
- 커서의 마지막 항목 삭제 개선
- RRDS 파일 삭제에 대한 향상된 지원
- 블루샘 시크릿 성능 개선 AWS
- SQL 프레임워크의 데이터베이스 연결 처리 개선
- 표준화된 AWS 다중 데이터 소스 비밀 관리자 키
- 성능 회귀 수정
- 에 대한 검사 기능 개선 PackedType
- 에 대한 저값 처리 개선 PackedType
- cognito 연결을 위한 업그레이드된 스프링 보안 패키징
- DB2 대상 데이터베이스에 코드시프트 포인트 인코딩 및 디코딩을 적용하지 않음

써드파티

- Spring Boot 업그레이드: 2.5에서 2.7로 업그레이드

현대화 도구 릴리스 3.8.0

zOS

새로운 기능

- JCL: 캐리지 리턴이 "\r"인 스트림 처리

개선 사항

- ON SIZE 오류 조항이 있는 DIVID를 현대화할 때 0으로 나누지 않도록 로깅 개선
- JCL: 프로시저의 프로시저 호출에 대한 지원 강화
- 모호한 필드가 있는 경우 FORMATTIME CICS 명령의 OF 키워드 지원
- JCL: 변수의 Â¥ 문자 지원
- JCL: 이전 단계를 기반으로 RC 계산

- PL1 SUBSTR을 사용하는 경우 문자열 대신 바이트 비교
- 단일 소스에서 다차원 배열을 초기화하는 기능 개선
- IF 블록에서 단일 SQL 쿼리를 포함할 때 COBOL의 구문 분석 개선

AS400

새로운 기능

- CL의 중첩된 IF 문 지원
- RPG 자유 형식의 ENDDO 문에 대한 지원 개선

개선 사항

- 컨트롤 레벨 컨디셔닝 지원 개선
- LIKE를 통한 프로토타입 반품 개선
- %months, %year, %days 함수 처리에 대한 지원 개선
- 전체 화면에 대한 도움말 기능 지원
- 파라미터로 전달되는 비유성 BLANKS 조정
- "" 연산자를 사용한 표현식 EVAL 개선
- KEY PHASE 없이 START 명령 처리
- 키워드 LIKEREC 처리가 개선되었습니다.
- 이름이 지정되지 않은 서브필드 개선
- 서명되지 않은 형식을 반환하는 프로시저 개선
- RESET 작업(무료 RPG), %CHAR 및 %DEC 내장 기능에 대한 지원이 개선되었습니다.
- 내장 함수 %LOOKUPXX가 향상되었습니다.
- 프로토타입이 없는 프로시저에서 LIKEDS 키워드에 대한 지원 개선
- Dim 키워드 배열 유형(VAR, AUTO) 처리
- XFOOT에 대한 지원 개선
- COBOL: RENAMES 필드에 대한 지원 개선
- CL: while(true) 조건 지원
- LIKE 키워드를 사용한 독립형 배열의 처리 개선
- 내장 함수 %INT 개선

- RPG 폴 프리 구문 분석 개선
- 링키지의 배열 지원 개선
- CL2GROOVY: 문 선택 지원
- DSPF 키워드 “ERRMSGID”의 개선
- 앞에 0이 있는 바이트 초기화 처리 개선
- 숫자 필드의 승인된 값 개선
- 자유 형식 EVAL 문을 위한 익스텐더 H 처리
- CL에서 Groovy로: LDA의 하위 문자열 지원
- 레코드의 RESET 지원이 개선
- 참조를 통한 EDTCDE 및 EDTWRD 처리 기능 개선
- DDS 필드를 사용한 입력 필드 매핑 개선
- MOVEA 문자를 IN 배열로 변환하기 위한 지원이 개선되었습니다.
- LIKEDS 키워드를 사용한 프로토타입 개선
- DSPF 키워드 DSPATR에 대한 지원이 개선되었습니다.
- +/-를 사용한 D 카드의 구문 분석 개선
- 프로그램 호출의 견고성 추가
- 현장 해결 프로세스에 견고성 추가

횡단 기능

개선 사항

- FrontEnd: IME 입력에 대한 붙여넣기 이벤트 시뮬레이션

써드파티

- Spring Boot 업그레이드: 2.5에서 2.7로 업그레이드

릴리스 노트 3.7.0

AWS Blu Age 런타임 및 현대화 도구의 이번 릴리스에는 주로 명령 및 유틸리티를 더 잘 지원하기 위한 개선 사항, AWS Secrets Manager와의 통합 기능 및 새로운 모니터링 기능이 포함되어 있습니다. 이 릴리스의 주요 변경 사항 중 일부는 다음과 같습니다.

- 이제 여러 런타임 구성 요소가 AWS Secrets Manager를 사용하여 주로 유틸리티 데이터 소스, TS 대기열용 Redis, 캐시 및 잠금과 관련된 현대화된 애플리케이션의 보안 설정을 강화할 수 있습니다. BluSam
- 상태, 기간, 볼륨 등과 같은 리소스 사용 최적화 및 운영 관리를 위한 트랜잭션, 배치 및 JVM 메트릭을 검색할 수 있는 모니터링 엔드포인트.
- RPG에서 IBM MQ 호출을 지원하는 새로운 기능과 향상된 JCL SORT 및 IDCAMS 변환 커버리지를 지원합니다.

이 릴리스에 포함된 변경 사항에 대한 자세한 내용은 다음 단원을 참조하세요.

런타임 릴리스 3.7.0

주제

- zOS
- AS400
- 횡단 기능

zOS

새로운 기능

- 문법과 같은 SQL을 사용하여 프로그램 유틸리티 애플리케이션과 관련된 구문 분석 쿼리를 개선합니다. (V7-9401)
- 오프셋 시 인덱싱된 가변 크기 배열 처리(V7-9904)
- 24:00:00 시간 형식으로 DB2에 SQL TIME 열 삽입 지원(V7-10023)
- FOR ROWS 및 ATOMIC 옵션이 있는 배열에서 SQL 쿼리 삽입 지원(V7-10105)
- JCL SORT - IFTHEN (V7-10124) 을 사용하여 TranscodeTool 아웃렉을 지원하도록 개선되었습니다.
- JCL 정렬 - OUTREC 명령에 DATE 키워드에 대한 지원 추가(V7-10125)
- JCL - 인스트림 프로시저 지원 추가(V7-10223)

개선 사항

- “PASS” 처리가 표시된 데이터 세트는 모든 작업 단계에서 사용할 수 있어야 함(V7-9504)

- JCL 속성 SCHENV(V7-9570) 지원
- CTLCHAR 옵션을 통한 SEND 지원(V7-9714)
- COBOL - ACCEPT 명령문(V7-9875)에서 다른 줄 구분자 문자 세트를 처리
- 다중 롤백 방지(V7-9958)
- MOD 처리를 사용하여 GDG 파일(V7-10031) 끝에 추가 가능
- 최적화: PuTall 리팩터링(V7-10063)
- PutAll 리팩토링: 페이지 매김 추가 (V7-10063)
- Jedis 클라이언트 읽기 타임아웃을 구성 가능하게 설정(V7-10063)
- UseSsl 독립형 모드 지원 (V7-10114)
- 파일을 성공적으로 연 후 EIBDS를 지원(V7-10147)
- 파일 제어 요청 후 EIBDS 지원(V7-10147)
- CICS SYNCPOINT 지원 개선(V7-10187)
- BluesamRedisSerializer: 메타데이터 지속성 문제 (V7-10202)
- TS 대기열에 대한 Redis AWS Secrets Manager 지원(V7-10204)
- DD 이름 크기 사용자 지정에 대한 JCLBCICS 지원(V7-10224)
- IDCAMS DELETE 명령문에 절대 경로에 대한 지원 추가(V7-10308)

AS400

새로운 기능

- AS400 스크린을 위한 도움말 기능 구현(V7-9673)

개선 사항

- INFDS에 있는 레코드 수(V7-9377)

횡단 기능

새로운 기능

- Amazon에 로그를 전송하기 위한 EC2 런타임 지원 CloudWatch (D87990246)
- 배치, 트랜잭션 및 JVM에 대한 지표를 검색할 수 있는 새 엔드포인트가 추가되었습니다 (D88393832)

개선 사항

- 유틸리티 페이지(V7-9570)용 데이터 소스 AWS Secrets Manager 지원
- DSNUTILB DISCARD(V7-9798)에 대한 Db2 지원 추가
- 기본 SYSPRINT 및 SYSPUNCH 파일(V7-10098)에서 기본 시스템 출력 스트림 대신 로거에 쓰기 지원
- AWS Secrets Manager (V7-10238) 에서 BluSam Redis 캐시 및 잠금 연결 속성 지원
- Db2 XA AWS 시크릿(V7-10258)에서의 SSL 연결 지원
- IDCAMS 재현 및 검증을 위한 메타데이터 업데이트(V7-10281)
- 향상된 IDCAMS 어벤드 리턴 코드 관리(V7-10307)

현대화 도구 릴리스 3.7.0

주제

- zOS
- AS400
- 횡단 기능

zOS

새로운 기능

- PLI - 배열 횡단면 및 2차원 배열에 대한 향상된 할당(V7-9830)

AS400

새로운 기능

- 컨트롤 레벨 인디케이터 처리(V7-9227)
- EXTNAME 파라미터 지원 *입력(V7-9897)
- 향상된 Goto 재작성: SELECT OTHER 명령문에 있는 태그에 대한 지원(V7-9973)
- Refshit DSPF 키워드 지원(V7-10049)

개선 사항

- 파일 설명 키워드 EXTIND(*Linux)(V7-7404) 처리 개선
- SQLDDS 파일 변환 개선(V7-7687)
- AS400 파일(V7-9062)에 대해 더 이상 파일 객체가 생성되지 않음
- 파일 설명 키워드 EXTDESC(V7-9268)의 처리가 개선
- 내장된 %CHAR(V7-9311)의 처리 개선
- SFLEND(V7-9322)를 사용하지 않은 마지막 레코드의 페이지 다운 지원 개선
- 접두어가 붙은 데이터 구조(V7-9436)에 대한 지원 개선
- %SIZE를 사용하여 정의된 치수에 대한 지원(V7-9472)
- 큰 따옴표로 선언된 PF 필드 이름 처리 지원(V7-9557)
- 파일 작업 개선 - 대소문자 구분 안 함(V7-9785)
- *USER로 초기화된 필드 지원(V7-9806)
- AS400(V7-9840)의 COMP 유형 지원
- (아니요) (V7-9922) 에서의 COBOL400 파싱이 개선되었습니다. InvalidKey
- SCAN 작업 처리 개선(V7-9971)
- GOTO 옵코드(V7-9973)에 대한 지원 개선
- 예외 작업 처리 개선(V7-9977)
- 접두사 지원 개선(V7-10000)
- RPG에서의 MQ 직접적 호출 지원(V7-10007)
- %LOOKUP 내장 기능(키 배열 데이터 구조) 개선(V7-10022)
- *모든 작업 닫기 지원(V7-10036)
- UPDATE AS ROW CHANGE SQLDDS 명령문 지원(V7-10051)
- 리터럴 값 유형 Long(V7-10073)을 처리하도록 개선
- RPG 문법 개선(서브루틴 이름으로 INZ 키워드 사용)(V7-10074)
- 분수 부분이 비어 있는 숫자 값을 지원하도록 RPG 문법 개선(V7-10077)
- CL과 외부 파일(V7-10081) 간에 공유되는 필드에 대한 지원 개선
- DDS 조건부 표시기(V7-10084)에 대한 지원 개선
- COBOL 프로그램을 통한 DDS 바이너리 형식 지원(V7-10100)
- 연결을 통한 이름 충돌 개선(V7-10109)
- 기본 프로시저와 익스포트 프로시저의 혼합 지원(V7-10112)

- 하위 DataStructure 프로시저에서의 지원 개선 (V7-10113)
- CLEAR(V7-10126)에 대한 지원 개선
- DO 루프(V7-10134)에 대한 지원 개선
- 풀 프리 RPG(V7-10151)에서 SQLTYPE 지원
- DDS 키워드에 대한 조건 구문 분석 개선(V7-10155)
- 향상된 DSL 생성(V7-10163)
- 조건이 이진 표현식인 경우의 프로세스 표시기가 개선되었습니다. (V7-10164)
- GoTOS 및 Else 조건 개선(V7-10168)
- DSPF의 유형 타임 및 타임스탬프 지원(V7-10173)
- DDS(V7-10183)의 연속 라인 구문 분석 개선
- DRENAMES FLD OF RECORD를 위한 COBOL 지원(V7-10195)
- DSPF 필드의 조건부 표시기 구문 분석 개선(V7-10221)
- DDS 키워드 NOALTSEQ(V7-10288)의 구문 분석 지원
- 지원 도움말 메뉴 및 숨겨진 필드(V7-10314)
- DSPF 도움말 키워드 온전성 검사 개선(V7-10328)
- 더 이상 참조 필드에 모든 키워드를 전파하지 않음(V7-10347)

횡단 기능

새로운 기능

- 데이터 마이그레이션 - CLOB 데이터 처리(V7-9665)

개선 사항

- (V7-10225) 를 통해 JCL 속성 SCHENV를 작업에서 PROC GROOVY 정의로 전파하기 JobContext
- FrontEnd - 테두리가 없는 경우 창 크기 조정 (V7-10358)

릴리스 노트 3.6.0

AWS Blu Age 런타임 및 현대화 도구의 이번 릴리스는 zOS 및 AS400 레거시 마이그레이션을 위한 새로운 기능을 제공합니다. 이 기능은 주로 CICS 지원 메커니즘 확장, JCL 기능 보완, 동시 및 대용량 기

능의 성능 최적화, 기능 추가에 중점을 두고 있습니다. multi-data-source 이 릴리스의 주요 변경 사항 중 일부는 다음과 같습니다.

- JCL 동적 파일 처리 개선, 현재 명령문 확장 및 연결된 데이터 세트 관리, 단일 블록에서 여러 명령문 실행, 배치에서 프로그램으로의 데이터 전송.
- 여러 CICS 리소스 유형에 대한 조회를 포함하여 여러 CICS 명령에 대한 지원이 향상되었습니다.
- Blu Age Runtime Utility를 사용할 때 서로 다른 데이터베이스를 사용할 수 있는 기능은 비즈니스 데이터가 여러 소스에 분산되는 시나리오에 가장 적합합니다.

이 릴리스에 포함된 변경 사항에 대한 자세한 내용은 다음 단원을 참조하세요.

런타임 릴리스 3.6.0

주제

- zOS
- AS400
- 횡단 기능

zOS

새로운 기능

- JCL - DynamicFileBuilder - 향상된 파일 핸들 관리 (V7-9408)
- INFUTILB UNLOAD 유틸리티(V7-9554)를 호출할 때 일부 내장 SQL DB2 함수의 형식 변환 기능 향상
- 향상된 PLI 다차원 배열 할당(V7-9592)
- 파일로의 sysout 리디렉션 처리(V7-9992)

개선 사항

- DB2 RDBMS(V7-9155)에 대한 저장 프로시저의 트리거링 추가
- 정렬은 PDF 형식(V7-9286)으로의 변환 처리
- JCL/GROOVY - DUMMY 데이터셋(V7-9424)을 지원하도록 REPRO 명령문 개선
- CICS UNLOCK 지원 개선(V7-9606)
- 유니온(V7-9648)의 기본값 크기 처리

- JCL/GROOVY는 연결된 데이터 세트(V7-9653)에서 다양한 종료/처리를 처리
- BluSAM 데이터 세트에 대해 PageSize 구성 지원(V7-9680)
- DSNUTIL - DB2LUW(V7-9697)에서 24:00 :00을 유효 시간으로 로드
- NumberUtils.ne () NumberUtils /.eq () (V7-9731) 에서 고가치 (0xff) 비교 지원
- JCL/GROOVY - DO ... 지원 IDCAMS IF-THEN-ELSE 절의 THEN 키워드를 사용하면 단일 블록에서 여러 명령문 실행 가능(V7-9750)
- 잘못된 JHDB를 JHDB 외부 프로그램으로 호출했습니다 (V7-9782). BatchRunner
- SORT OUTFIL 제어 카드의 공백 문자 지원(V7-9808)
- CICS READ PREV 지원 개선(V7-9845)
- 데이터 세트 인덱스(V7-9864)에 대한 동시 액세스 개선
- CICS REWRITE 지원 개선(V7-9873)
- COBOL - ACCEPT 명령문의 여러 줄 SYSIN을 지원하여 배치(JCL)에서 프로그램(COBOL)으로 데이터를 전달할 수 있음(V7-9875)
- 그루비 - 파일 생성 ConcatenatedFileConfiguration 단계에서 더 나은 처리 능력 (V7-9876)
- IDCAMS UTILITY - DEFINE PATH 명령문 처리(V7-9878)
- SORT BUILD - TRAN 옵션 조정 및 암시적 공백 처리(V7-9925)
- GENERIC 옵션 지원(V7-9939)을 통해 CICS DELETE 개선
- CICS STARTBR 및 ENDBR 지원 개선(V7-9952)
- 동시 액세스 시 클로즈 퍼포먼스 개선(V7-9953)
- 시작 시 파일 상태 처리 개선(V7-9991)
- 그루비 - (V7-10012) 에서 getDisposition ()/getNormalTermination()/getAbnormalTermination() 호출 허용 ConcatenatedFileConfiguration

AS400

새로운 기능

- COMMIT 키워드에 대한 외부 표시기 지원(V7-6035)
- SFLCTL 쓰기 후 ReadC 루프 재설정(V7-8061)
- CALL 중 LR 표시기 지원(V7-9250)
- 여러 줄의 입력 필드를 처리하기 위한 새로운 유형의 동적 필드(분할) 추가(V7-9370)

- 기본/보조 파일 지원(V7-9390)
- 이제 작업을 제출할 때 로컬 데이터 영역이 직접적으로 호출된 작업 전달(V7-9775)
- 데이터 영역에 대한 QTEMP 지원 및 데이터 영역 가치 생성 지원. (V7-9916)
- 약정 제어 - 약정 제어 활성화/비활성화 지원(V7-9956)
- COMMIT 키워드에 대한 외부 지표 지원

개선 사항

- 0 값 표시 및 EDTWRD(V7-8933) 개선
- DSPF 키워드 “CHKMSGID” 지원(V7-9125)
- 배치 종료 시 SQL 커밋 트랜잭션(V7-9232)
- 필드 및 데이터 구조(V7-9265)에 대한 키워드 EXPORT 및 IMPORT 지원 개선
- Support DateHelper (V7-9461) 에서 소문자 입력 지원
- *CYMD를 *ISO(숫자)로 변환하는 지원(V7-9488)
- 가변 필드(표현식의 왼쪽 및 오른쪽)에 대한 내장 %len 핸들 개선(V7-9733)
- 내장 함수 '%LOOKUPXX' XX ("LE","LT","GE","GT")에 대한 지원 개선(V7-10064)

횡단 기능

새로운 기능

- CICS - 옵션 상태에 대한 조회 트랜잭션 개선(V7-9712)
- JCL - 시스템 출력 파일(V7-9797)을 사용하여 sysprint의 부하 개선
- CICS - INQUIRE TSQUEUE 개선(V7-9823)
- CICS - 옵션 사용자 ID에 대한 조회 터미널 개선(V7-9906)

개선 사항

- 공백(V7-8047)과의 비교 처리 개선
- Jics 및 BluSam (V7-8847) 에 대한 로깅을 개선합니다.
- 동적 필드에 대한 BMS 확장 속성 SOSI 및 프로그래밍 기호 F8 지원(V7-8857)
- 프로그램 파라미터(V7-9138)의 버퍼 오버플로우 처리

- Blusam 잠금 레지스트리(V7-9505)의 스레드 쓰기 동시성 개선
- 유틸리티-PGM(V7-9570)에 대한 다중 데이터 소스 구성 지원
- Blusam 레코드 레벨 잠금 전용 모드(V7-9626)
- 서버 재시작 시에도 메타데이터 지속성이 유지되는지 확인(V7-9748)
- 예외 시 DAO 정리 개선(브라우저 닫기)(V7-9790)
- INFUTILB 시스펀치 (V7-9799) 지원 DummyFile
- (V7-9935) 의 음수 값에 대한 지원 강화 NumericEditedType

현대화 도구 릴리스 3.6.0

주제

- zOS
- AS400
- 횡단 기능

zOS

새로운 기능

- JCL - 프로시저 종료를 위한 로깅 향상(V7-8509)
- PL1 - 데이터 유형에 대한 백 생성 기능 향상 (V7-8917) PakedLong
- JCL - 파일에 “종료” 마커가 포함된 경우 프로시저 종료에 대한 로깅 개선 // (V7-9509)
- PL1 - 고정 소수점 및 SYSIN 스트림을 통한 GET EDIT에 대한 지원 강화(V7-9593)
- DB2 - VARGRAPHIC DB2 유형(V7-9809)에 대한 지원 강화
- CICS - 옵션 LOGMESSAGE(V7-9969)의 명령 쿼리 보안 개선
- PL1 - CHARG/차그래피 내장(V7-9989)을 위한 백 생성 기능 개선

개선 사항

- PL1- INCLUDEX 키워드(V7-9588)에 대한 지원 강화
- PL/I - CHARGRAPHIC 키워드를 모든 메서드 호출의 유효한 매개변수로 처리(V7-9589)
- 특수 문자 @ # $ §.로 이름을 지정할 때 PL1 호스트 변수 해상도 개선. (V7-9654)

- COBOL - C01... C12 및 S01... S05 키워드를 구문 분석 단계에서 고급 명령문 작성의 파라미터로 지원(V7-9669)

AS400

새로운 기능

- 분석기에서 SQL-DDS 변환 지원(V7-7687)
- SQL-DDS 파일 탐지 자동화(V7-7687)
- SQL-DDS 사전 처리 구현(V7-7687)
- ALIGN 키워드 지원(V7-9254)
- DSPF 및 멀티 디밍 어레이 지원 ExtName (V7-9663)
- 코볼 InvalidKey 라이트에 대한 지원 성명서 (V7-9793)

개선 사항

- TESB 옵코드 개선(V7-8865)
- DECFMT 온 포커스(V7-8933) 지원 개선
- MOVE에서 결과 인디케이터 처리(V7-9224)
- 필드 및 데이터 구조에 대한 키워드 TEMPLATE 지원 개선(V7-9278)
- LIKEDS 개선(LIKEDS를 사용하여 정의된 DS는 자동으로 검증됨)(V7-9302)
- COBOL - 지표 구조 생성 개선(V7-9423)
- 프로토타입의 Const 파라미터가 읽기 전용이 아님(V7-9437)
- 편집 코드 “Y"(V7-9443)를 사용하여 EDTCDE 키워드 개선
- PSD 및 INFDS에서 *ROUTINE 필드 생성 지원(V7-9487)
- 필드 XXX를 독립형으로 재작성(재작성 시 기본값 손실) 개선(재작성 시 기본값 손실)(V7-9522)
- DSPF 키워드에 대한 지원 개선(V7-9658)
- 바이너리에서 ZEROES 기본값 처리하기(V7-9666)
- 암시적 포인터 지원(V7-9719)
- 파라미터 하나로 내장된 직접적 호출 %size를 처리하는 기능 개선(V7-9730)
- 내장 호출(%ELEM)(V7-9736)의 데이터 구조 참조 처리 개선
- 정의 사양(V7-9738)에서 LIKE 참조를 사용하는 필드의 부호있는 길이 처리 개선

- REWRITE(V7-9791) 개선
- DDS 파일(V7-9803)에서 인덱스 생성 기능 개선
- 잘못된 숫자 값(V7-9813) 으로 매퍼의 안정성 개선
- SQLModel 및 모든 인덱스 파일 생성 개선(V7-9818)
- 검증된 DS 지원 개선(V7-9863)
- LOOKUP 지원 개선(파라미터에 DS와 같은 독립형 필드 사용)(V7-9961)
- 인디케이터 LIKE 개선(V7-9985)
- MVR(V7-9995)에서 결과 인디케이터 처리하기
- 물결표가 있는 문자 N 지원(V7-10021)
- SQLDDS 레거시 파일(V7-10067)에서 최신 DDL 파일 생성 개선

횡단 기능

새로운 기능

- yml 속성을 사용하여 리소스 위치를 사용자 지정합니다.(D88816105)
- COBOL - GO TO/PERFORM... 를 사용하지 않고 인라인 PERFORM에서 종료할 수 있는 EXIT PERFORM 명령문 지원 THROUGH(V7-9582)
- 글로벌 메타데이터로 고려할 기본 레거시 인코딩을 지정합니다. (V7-9883)

개선 사항

- 마스크 생성 기능 향상(V7-9602)
- 컨텍스트 워밍업 개선(V7-9621)
- 문자셋 CUSTOM930 스레드를 안전하게 만듭니다. (V7-9674)
- MOVEA 개선(V7-9773)

릴리스 노트 3.5.0

이번 AWS Blu Age Runtime and Modernization Tools 릴리스는 zOS 및 AS400 레거시 마이그레이션을 위한 새로운 기능을 제공하며, 이는 주로 데이터세트와 메시징 최적화에 중점을 두고 있으며 변환 프로세스의 결과로서 확장된 Java 기능을 제공합니다. 이 릴리스의 주요 변경 사항 중 일부는 다음과 같습니다.

- 기존의 Groovy 스크립트 기능 외에도 CL 프로그램을 Java로 마이그레이션하여 현대화된 다른 프로그램과의 통합을 용이하게 하고 최종 프로그래밍 언어를 통합하여 고객 학습 과정을 단순화할 수 있습니다.
- 새로운 데이터 벌크 기능을 사용하여 Redis에서 데이터 세트를 로드하는 시간을 줄이고 성능을 최적화합니다.
- 작업 단계 내에서 데이터 세트를 운영하고 전달하여 기존 데이터 세트 동작을 현대화할 수 있습니다.
- VB 입력 파일 및 Java 11의 간소화된 마이그레이션을 지원하도록 SQL 마이그레이션을 확장했습니다.
- 추가 헤더, 확장된 GET/PUT 지원, 대기열 메타데이터 자동 검색 등 IBM MQ와의 보다 빠른 통합을 위한 여러 가지 새로운 메커니즘.
- 데이터 세트 메타데이터를 위한 REST 엔드포인트 및 S3 버킷의 데이터 세트 가져오기.

이 릴리스에 포함된 변경 사항에 대한 자세한 내용은 다음 단원을 참조하세요.

런타임 릴리스 3.5.0

주제

- zOS
- AS400
- 횡단 기능

zOS

새로운 기능

- JCL SORT - 새 키워드 오버레이 처리(V7-9409)
- ZOS COBOL - 플로팅 차트(V7-9404)에 대한 지원 강화
- T RedisJics 대기열에서 & T로 연결되는 포트 (V7-9212) RedisTemplate ListOperations
- ZOS JCL - (V7-9012) 를 통해 정의된 경우 임시 디렉토리의 경로를 파일 디렉토리로 확장합니다. UserDefinedParameters
- FUNCTION ORD-MAX를 ALL(모든 배열 항목)로 처리하기(V7-9366)
- 이제 Redis(V7-9212)에 TS 대기열을 저장할 때 접두사가 붙은 키와 사람이 읽을 수 있는 키 사용
- blusam API용 데이터 세트 엔드포인트 가져오기 추가
- JCL - 이름에 #(V7-9136)과 같은 특수 문자가 포함된 배치 작업에 대한 지원 추가

- 이제 TSModel 페치가 온디맨드 방식으로 강력하게 수행(V7-9212)

개선 사항

- LNK 파일(V7-6022)에서 버전이 지정되지 않은 INCLUDE 지원
- MQ - 향상된 인코딩 지원(V7-9652)
- 다양한 문자 유형에 대한 2바이트 또는 혼합 문자셋 지원 개선(V7-9596)
- JCL - IDCAM의 filesDirectory 구성 지원: 비 VSAM 명령문 삭제(V7-9609)
- 파일에서 로드되는 ESDS 및 RRDS 데이터 세트를 위한 벌크 모드 지원(V7-8639)
- 입력 모드에서 빈 ESDS를 여는 작업을 처리합니다. (V7-9287)
- ORD/UNORD 약어 지원을 통한 클러스터 정의 명령문 개선(V7-9451)
- BluSam 레디 스 락 성능 개선 (V7-8639)
- DATA() 인수 범위(V7-9337)에 제공된 레코드 크기를 지원하도록 DEFINE CLUSTER 명령문 개선
- DEFINE CLUSTER 명령문(V7-9419)에 BUFFERSPACE/UNIQUE 속성 지원 추가
- 가변 길이 레코드 데이터세트의 BluSam 읽기 작업을 개선합니다. (V7-9391)
- CICS ADDRESS는 누락된 CWA를 null로 올바르게 표시(V7-9491)
- 엔드 록에서 불필요한 쓰기 제거(V7-8639)
- 캐시에 Redis 캐시 템플릿 삽입을 처리합니다(V7-9510).
- BPXWDYN 파라미터(V7-9417)를 올바르게 디코딩
- LISTCAT 내보내기 소비 개선(V7-9201)
- BluSam TS 대기열 이름 (V7-9212) 에서 인쇄할 수 없는 문자 지원
- 맵 세트가 null인 필드(V7-9486)에 대한 핸들 수신 맵 작성
- 동적 액세스 BluesamRelativeFile 모드의 삭제 및 재작성 작업을 개선합니다. (V7-8989)

AS400

새로운 기능

- 표준 DS/STM 피벗(V7-9427)을 통해 CL 파일을 자바 프로그램으로 생성하는 기능 추가
- ADD 모드가 있는 Support 입력 파일(V7-9378)
- cl 명령 OPNQRYF (쿼리 파일 열기) 를 지원하도록 정렬 순서 및 검색 관리를 개선하고 SHARE 매개변수에 대한 지원을 추가했습니다. OverrideItem (V7-9364)

개선 사항

- SFLNXTCHG 지원 (V7-8061) UpdateSubfile
- CL 명령 실행 시 CL 컨텍스트 범위 수정(V7-9624)
- 프로그램 BPXWDYN(V7-9417)에 대한 반환 코드 처리
- 로컬 모니터를 지웁니다. (V7-9624)
- DSPF 키워드 RTNCSRLOC(V7-9389) 지원
- setOnGreaterOrEqual() 1을 1로 설정하지 않음 (V7-9342)
- UpdateSubfileRecord (V7-9376) 에서 필드 캐시 업데이트
- SFLNXTCHG(V7-8061) 내보내기 지원

횡단 기능

새로운 기능

- 리터럴 그래픽 문자열의 G 접두사는 무시합니다. (V7-9420)
- ZOS COBOL - 일부 특수 구조에 대한 Fiedl.initialize() 지원 강화(V7-9485)
- 컨텍스트를 비동기적으로 초기화하여 프로그램 시작 성능 향상(V7-9446)
- SQL 릴리즈: 열린 준비 명령문 및. ResulSet (V7-9422)
- JMS MQ 향상 - MQ PUT용 MQRFH2 지원/ V7-7085 - 기본 큐 관리자 지원(V7-9400)
- SQL 관리 - SET 명령의 파라미터에 대한 Lambda 변환 활성화(V7-9492)
- ZOS MQ JMS - MQCOMIT 및 MQBACK에 대한 지원 추가(V7-9399)
- ZOS IBMQ - MQINQ(V7-9544)에 대한 지원 강화
- 더블바이트 인코딩을 사용할 때 문자열 대신 바이트를 사용하여 CONCAT 작업을 처리합니다. (V7-8932)
- ZOS IBMQ - SET_ALL_CONTEXT 옵션을 사용하여 PUT 명령 지원 강화(V7-9544)

개선 사항

- gdg 파일 이름을 $ 문자로 처리하기(V7-9066)
- SQL 진단은 이전 SQL 문이 성공하면 1을 NUMBER 절로 반환합니다. (V7-9410)
- 길이가 null이 아닌 필드에 대한 개요(V7-7536)

- 내장 PL1 GRAPHIC 기능 지원(V7-9245)
- MQ - MQGMO 필드 설정에 대한 버전 지원 추가(V7-9500)
- JMS MQ GET - 메시지 반환 데이터 길이 개선(V7-9502)
- ROSET 컨텍스트에서 가져온 항목 수를 사용하여 sqlerrd(3)를 설정합니다. (V7-9371)

현대화 도구 릴리스 3.5.0

주제

- zOS
- AS400
- 횡단 기능

zOS

새로운 기능

- zOS PLI - 이진 표현식을 사용한 대입 시 별표 색인 지원(V7-9178)
- JCL에서 BatchScript - A “//”로 설정하면 작업 실행이 종료됩니다 (V7-9304).
- ZOS PLI - 플로팅 문자 및 로그인 숫자 편집 유형 지원 강화(V7-8982)
- COBOL - 내장된 SUM 함수 지원(V7-9367)
- JCL- 선택적으로 null 명령문(//) 다음에 데드 코드를 주석 처리(V7-9202)
- JCL- 조건문에서 '|' 연산자 지원(V7-9499)
- PL/I - 구문 분석 예외 방지를 위한 사전 처리 단계의 사전 컴파일 지시문 설명(V7-9507)

개선 사항

- 구분자를 사용한 스트림 정의 처리(V7-9615)
- LISTCAT 익스포트 처리 개선. (V7-9201)
- PL/I - 암시적 'null' 인수를 지원하도록 개선 사항(V7-9204)

AS400

새로운 기능

- DDS 키워드 CONCAT(V7-9439) 지원
- DSPF 키워드에 대해 생성된 자바 코드를 리팩터링합니다. (V7-7700)
- 데이터 구조 정의 내 필드에 대한 다양한 키워드 지원(V7-9029)

개선 사항

- 논리적 관계 AND/OR(V7-9352)의 구문 분석 개선
- COBOL vo와 dEntity(V7-9449) 간의 매핑 개선
- 수치 입력에 초점을 맞춘 경우 빈 값 표시(V7-9374)
- SQL 선언 커서의 로컬 변수(V7-9456)
- 빈 DS로 인한 스코프 문제(V7-9466)
- col 80 이후에는 구문 분석하기 전에 줄 잘라내기(V7-9632)
- 정의 사양(V7-9358)에서 키워드(DIM, LIKE,...)의 필드 참조 및 내장 호출 처리 개선
- SQL 코멘트(--) 지원(V7-9632)
- FullFree 구문 분석, 날짜/시간/타임스탬프 (V7-9542) 입력
- 파싱에서 가져온 SQLCA 포함 (V7-9333) FullFree
- 제어 수준 지원 향상. (V7-9610)
- *BLANKS(V7-9668)를 사용한 핸들 DS 비교
- DDS(V7-9318)의 다중 인디케이터에 대한 지원 개선
- 다중 DSPF 프로그램에 대한 지원을 개선합니다(V7-9657).
- LIKE(배열에서 유사한 데이터 구조의 대소문자 및 유사한 데이터 구조의 대/소문자)를 사용하여 필드 처리 개선(V7-9213)
- 무료 RPG, 리터럴에 따른 핸들 연속(V7-9686)
- 프로그램 종료 기록 지원 개선(V7-9452)
- CALL 명령문의 LINKAGE 문구 지원. (V7-9685)
- CASXX 운영 코드(CASXX 그룹이 없는 CASBB)(V7-9357)
- FullFreeRPG 파싱 개선 (V7-9457)
- 내장 %LEN은 DS를 인수로 지원하지 않음(V7-9267)
- 요인 2가 *ALL'X... '일 때 MOVEA 개선(V7-9228)
- RENAME 필드를 통한 지정 지원(V7-9385)

횡단 기능

새로운 기능

- SQL 마이그레이션 도구 - ebcdic 로드 단계에서 가변 레코드 길이에 대한 OID 옵션을 추가합니다. (V7-9380)
- SQL 마이그레이션 도구 - OID 옵션의 Java 11 지원(V7-9599)

개선 사항

- 중첩 어레이(V7-9595)에 대한 지원 개선
- Â¬ 문자를 !로 바꾸세요. Â¬의 경우 원본 인코딩으로 지원됩니다. (V7-9465)
- JCL - 작업 단계 간에 데이터 세트를 공유하기 위한 PASS 정상 종료 지원(V7-9504)
- VARCHAR 및 null이 가능한 db 열 유형을 처리하는 경우 ORACLE의 열 정의에 ON NULL을 적용합니다. (V7-9681)
- 스프링 인젝션 컴플라이언스 개선(V7-9635)

AWS 블루 에이지 런타임 컨셉

AWS Blu Age Runtime의 기본 개념을 이해하면 자동화된 리팩토링으로 애플리케이션을 현대화하는 방법을 이해하는 데 도움이 될 수 있습니다.

주제

- AWS 블루에이지 런타임 하이레벨 아키텍처
- AWS 현대화된 애플리케이션의 블루 에이지 구조
- 데이터 단순화

AWS 블루에이지 런타임 하이레벨 아키텍처

레거시 프로그램을 Java로 현대화하기 위한 AWS Blu Age 솔루션의 일부인 AWS Blu Age Runtime은 현대화된 애플리케이션을 위한 통합된 REST 기반 진입점과 레거시 구조를 제공하는 라이브러리와 프로그램 코드 구성의 표준화를 통해 이러한 애플리케이션을 위한 실행 프레임워크를 제공합니다.

이러한 현대화된 애플리케이션은 메인프레임 및 미드레인지 프로그램 (다음 문서에서는 “레거시”라고 함) 을 웹 기반 아키텍처로 현대화하기 위한 AWS Blu Age Automated Refactor 프로세스의 결과입니다.

AWS Blu Age Runtime의 목표는 톰캣, 스프링, 게터/세터, 유창한 API와 같은 친숙한 환경과 관용구를 사용하지만 레거시 프로그램 동작 (등기능), 성능 (프로그램 실행 시간 및 리소스 소비 관련), Java 개발자가 현대화된 프로그램을 쉽게 유지 관리할 수 있도록 하는 것입니다.

주제

AWS 블루에이지 런타임 컴포넌트

AWS Blu Age 런타임 환경은 두 종류의 구성 요소로 구성되어 있습니다.

- “공유 폴더”라고도 하는 자바 라이브러리 세트(jar 파일)로, 레거시 구문과 명령문을 제공합니다.
- 현대화된 프로그램에 공통 프레임워크 및 서비스를 제공하는 Spring 기반 웹 애플리케이션을 포함하는 웹 애플리케이션 세트(war 파일)입니다.

다음 섹션에서는 이러한 두 구성 요소의 역할을 자세히 설명합니다.

AWS 블루에이지 라이브러리

AWS Blu Age 라이브러리는 모든 현대화된 자바 프로그램에서 사용할 수 있도록 표준 톰캣 클래스 경로에 추가된 `shared/` 하위 폴더에 저장된 jar 파일 세트입니다. 이 회사의 목표는 Java 프로그래밍 환경에서 기본적으로 사용하거나 쉽게 사용할 수 있는 것이 아니라 기존 개발 환경에서 흔히 볼 수 있는 기능을 제공하는 것입니다. 이러한 기능은 Java 개발자에게 최대한 친숙한 방식으로 제공됩니다 (getters/setters, class-based, fluent API). 중요한 예로 데이터 단순화 라이브러리가 있습니다. 이 라이브러리는 기존 메모리 레이아웃 및 조작 구조(COBOL, PL1 또는 RPG 언어에서 사용됨)를 Java 프로그램에 제공합니다. 이러한 jar는 레거시 프로그램에서 생성된 현대화된 Java 코드의 핵심 종속성입니다. 데이터 형식에 대한 자세한 내용은 데이터 단순화 섹션을 참조하세요.

웹 애플리케이션

웹 애플리케이션 아카이브(WAR)는 Tomcat 애플리케이션 서버에 코드와 애플리케이션을 배포하는 표준 방법입니다. AWS Blu Age 런타임의 일부로 제공되는 프로그램은 레거시 환경 및 트랜잭션 모니터 (JCL 배치, CICS, IMS 등) 를 재현하는 일련의 실행 프레임워크와 관련 필수 서비스를 제공하는 것을 목표로 합니다.

가장 중요한 것은 `gapwalk-application`(종종 “Gapwalk”로 줄임) 인데, 이는 트랜잭션, 프로그램 및 배치 실행을 트리거하고 제어할 수 있는 통합된 REST 기반 진입점 세트를 제공합니다. 자세한 설명은 AWS 블루에이지 런타임 API 섹션을 참조하세요.

이 웹 애플리케이션은 Java 실행 스레드와 리소스를 할당하여 현대화된 프로그램을 설계된 컨텍스트에서 실행하도록 합니다. 이러한 재현 환경의 예는 다음 섹션에서 자세히 설명되어 있습니다.

다른 웹 애플리케이션은 레거시 프로그램에서 사용할 수 있고 기존 프로그램에서 호출할 수 있는 프로그램을 에뮬레이션하는 프로그램을 실행 환경(보다 정확하게는 아래에 설명된 “프로그램 레지스트리”)에 추가합니다. 이러한 두 가지 중요한 범주는 다음과 같습니다.

- OS 제공 프로그램 에뮬레이션: JCL 기반 배치는 특히 표준 환경의 일부로 다양한 파일 및 데이터베이스 조작 프로그램을 직접적으로 호출할 수 있을 것으로 기대합니다. 예를 들면 `SORT/DFSORT` 또는 `IDCAMS`입니다. 이를 위해 이러한 동작을 재현하는 Java 프로그램이 제공되며 기존 프로그램과 동일한 규칙을 사용하여 호출할 수 있습니다.
- “드라이버”는 실행 프레임워크 또는 미들웨어에서 진입점으로 제공하는 특수 프로그램입니다. 예를 들어 IMS 환경에서 실행되는 COBOL 프로그램이 IMS 관련 서비스(IMS DB, MFS를 통한 사용자 대화 등)에 액세스할 때 사용하는`CBLTDLI`이 있습니다.

프로그램 레지스트리

이러한 구조, 프레임워크 및 서비스에 참여하고 이를 활용하기 위해 기존 Java 프로그램에서 현대화된 Java 프로그램은 AWS 현대화된 애플리케이션의 블루 에이지 구조에 설명된 특정 구조를 준수합니다. 시작 시 AWS Blu Age Runtime은 이러한 모든 프로그램을 공통 “프로그램 레지스트리”에 수집하여 나중에 호출 (및 상호 호출) 할 수 있도록 합니다. 프로그램 레지스트리는 서로 호출하는 프로그램을 동시에 현대화할 필요가 없기 때문에 느슨한 결합과 분해 가능성을 제공합니다.

실행 환경

자주 접하는 레거시 환경과 안무를 확인할 수 있습니다.

- 일단 Java 프로그램과 Groovy 스크립트로 현대화된 JCL 기반 배치는 동기(차단) 또는 비동기(분리) 방식으로 시작할 수 있습니다. 후자의 경우 REST 엔드포인트를 통해 실행을 모니터링할 수 있습니다.
- AWS 블루에이지 서브시스템은 다음을 통해 CICS와 유사한 실행 환경을 제공합니다.
  - CICS “실행 수준” 안무를 준수하면서 CICS 트랜잭션을 시작하고 관련 프로그램을 실행하는 데 사용되는 진입점입니다,
  - 리소스 정의를 위한 외부 스토리지,

  - EXEC CICS 명령문을 재생하는 동종의 Java 유창한 API 세트,
  - 임시 스토리지 큐, 임시 데이터 큐 또는 파일 액세스와 같은 CICS 서비스를 재생하는 플러그형 클래스 세트(Amazon Managed Service for Apache Flink, Amazon Simple Queue Service 또는 TD 큐용 RabbitMQ와 같이 일반적으로 여러 구현이 가능함),
  - 사용자 대면 애플리케이션의 경우 BMS 화면 설명 형식이 Angular 웹 애플리케이션으로 현대화되고 해당 “유사 대화형” 대화 상자가 지원됩니다.
- 마찬가지로 다른 하위 시스템은 IMS 메시지 기반 안무를 제공하고 MFS 형식의 UI 화면 현대화를 지원합니다.
- 또한 세 번째 하위 시스템을 사용하면 DSPF(디스플레이 파일) 지정 화면의 현대화를 포함하여 ISeries와 유사한 환경에서 프로그램을 실행할 수 있습니다.

이러한 모든 환경은 다음과 같은 공통 OS 수준 서비스를 기반으로 합니다.

- 레거시 메모리 할당 및 레이아웃 에뮬레이션(데이터 단순화),
- COBOL “런 유닛” 실행 및 파라미터 전달 메커니즘(CALL 명령문)의 Java 스레드 기반 복제,
- Bluesam 라이브러리 세트를 통해 연결된 플랫 VSAM 및 GDG 데이터 세트 조직의 에뮬레이션
- RDBMS(EXEC SQL 명령문)와 같은 데이터 저장소에 대한 액세스.

무상태 및 세션 처리

AWS Blu Age Runtime의 중요한 기능은 현대화된 프로그램을 실행할 때 고가용성 (HA) 및 수평적 확장성 시나리오를 가능하게 하는 것입니다.

이를 위한 초석은 무상태이며, HTTP 세션 처리를 예로 들 수 있습니다.

세션 처리

Tomcat은 웹 기반이므로 이를 위한 중요한 메커니즘은 HTTP 세션 처리(Tomcat 및 Spring에서 제공)와 무상태 디자인입니다. 따라서 무상태 설계는 다음을 기반으로 합니다.

- 사용자는 HTTPS를 통해 연결합니다.
- 애플리케이션 서버는 로드 밸런서 뒤에 배포됩니다.
- 사용자가 애플리케이션에 처음 연결하면 애플리케이션이 인증되고 애플리케이션 서버가 식별자(일반적으로 쿠키 내에)를 생성합니다
- 이 식별자는 외부 캐시(데이터 저장소)에 사용자 컨텍스트를 저장하고 외부 캐시(데이터 저장소)에서 사용자 컨텍스트를 검색하는 데 사용됩니다.

쿠키 관리는 AWS Blu Age 프레임워크와 기본 Tomcat 서버에 의해 자동으로 수행되며, 이는 사용자에게 투명합니다. 사용자 인터넷 브라우저가 이를 자동으로 관리합니다.

Gapwalk 웹 애플리케이션은 세션 상태(컨텍스트)를 다양한 데이터 저장소에 저장할 수 있습니다.

- 아마존 포 ElastiCache 레디스용
- Redis 클러스터
- 메모리 맵에 저장(개발 및 독립 실행형 환경에만 해당되며 HA에는 적합하지 않음).

고가용성 및 무상태

일반적으로 AWS Blu Age 프레임워크의 설계 원칙은 스테이트리스 (Stateless) 입니다. 레거시 프로그램 동작을 재현하는 데 필요한 대부분의 비일시적 상태는 애플리케이션 서버 내에 저장되지 않고 외부의 공통 “단일 정보 소스”를 통해 공유됩니다.

이러한 상태의 예로는 CICS의 임시 저장소 대기열 또는 리소스 정의가 있으며, 이러한 상태를 위한 일반적인 외부 저장소는 Redis와 호환되는 서버 또는 관계형 데이터베이스입니다.

이러한 설계가 부하 분산 및 공유 세션과 결합되어 대부분의 사용자 대면 대화 상자(OLTP, “온라인 트랜잭션 처리”)를 여러 “노드"(여기서는 톰캣 인스턴스) 간에 배포할 수 있게 됩니다.

실제로 사용자는 어느 서버에서나 트랜잭션을 실행할 수 있으며 다음 트랜잭션의 직접적 호출이 다른 서버에서 수행되더라도 신경 쓰지 않아도 됩니다. 그런 다음 Auto Scaling으로 인해 또는 비정상 서버를 교체하기 위해 새 서버가 생성되면 연결 가능하고 정상적인 모든 서버가 예상대로 트랜잭션을 실행하여 적절한 결과(예상 반환값, 데이터베이스의 예상 데이터 변경 등)를 제공할 수 있습니다.

AWS 현대화된 애플리케이션의 블루 에이지 구조

이 문서에서는 개발자가 다음과 같은 다양한 작업을 수행할 수 있도록 AWS 메인프레임 현대화 리팩토링 도구를 사용하여 현대화된 애플리케이션의 구조에 대해 자세히 설명합니다.

- 애플리케이션을 원활하게 탐색합니다.
- 현대화된 애플리케이션에서 직접적으로 호출할 수 있는 사용자 지정 프로그램 개발
- 현대화된 애플리케이션을 안전하게 리팩터링합니다.

다음 사항에 대한 기본 지식이 이미 있다고 가정합니다.

- 레코드, 데이터 세트 및 레코드에 대한 액세스 모드(인덱스, 순차), VSAM, 실행 유닛, jcl 스크립트, CICS 개념 등과 같은 기존의 일반적인 코딩 개념.

- 스프링 프레임워크를 사용한 java 코딩.
- 문서 전체에서 가독성을 높이기 `short class names`를 사용합니다. 자세한 내용은 AWS Blu Age 런타임 요소의 해당 정식 이름 검색 및 타사 요소의 해당 정규화된 이름 AWS Blu Age 정식 이름 매핑. 타사의 완전한 자격을 갖춘 이름 매핑 검색하기를 참조하십시오.
- 모든 아티팩트와 샘플은 샘플 COBOL/CICS 애플리케이션의 현대화 프로세스 출력에서 가져온 것입니다. CardDemo

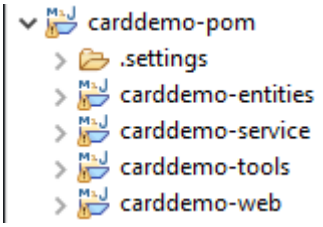
주제

- 아티팩트 조직
- 프로그램 실행 및 직접적으로 호출
- 직접 프로그램을 작성하세요
- 완전한 이름 매핑

아티팩트 조직

AWS Blu Age 현대화 애플리케이션은 JEE 서버에 배포할 수 있는 Java 웹 애플리케이션 (.war) 으로 패키징됩니다. 일반적으로 서버는 AWS Blu Age Velocity 런타임을 내장하는 Tomcat 인스턴스입니다. 이 런타임은 현재 Springboot와 Angular (UI 부분용) 프레임워크를 기반으로 구축되어 있습니다.

war는 여러 구성 요소 아티팩트(.jar)를 집계합니다. 각 jar는 현대화 프로세스의 결과물인 전용 자바 프로젝트를(maven 도구를 사용하여) 컴파일한 결과입니다.



기본 조직은 다음 구조를 취합니다.

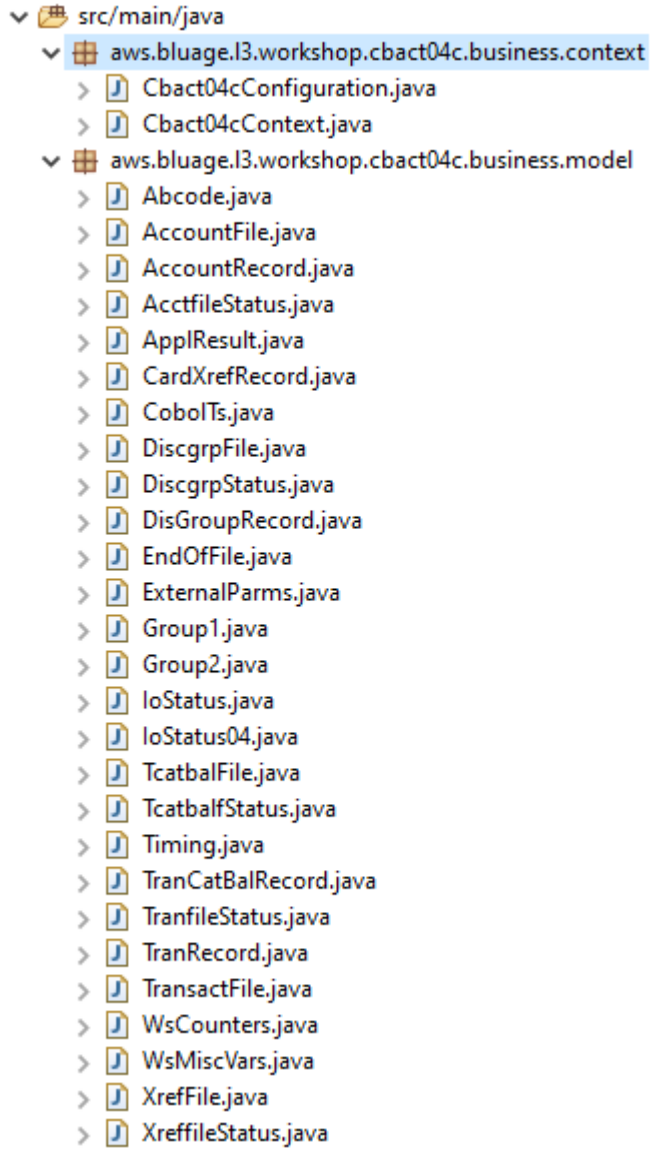
- 엔티티 프로젝트: 비즈니스 모델 및 컨텍스트 요소를 포함합니다. 프로젝트 이름은 일반적으로 “-entities”로 끝납니다. 일반적으로 특정 레거시 COBOL 프로그램의 경우 이는 I/O 섹션(데이터 세트) 및 데이터 분할의 현대화에 해당합니다. 2개 이상의 엔티티 프로젝트가 있을 수 있습니다.
- 서비스 프로젝트: 레거시 비즈니스 로직 현대화 요소가 포함되어 있습니다. 일반적으로 COBOL 프로그램의 프로시저 구분입니다. 프로젝트를 두 개 이상 선택할 수 있습니다.
- 유틸리티 프로젝트: 다른 프로젝트에서 사용하는 공유 공통 도구 및 유틸리티를 포함합니다.

- 웹 프로젝트: 해당하는 경우 UI 관련 요소의 현대화가 포함됩니다. 일괄 전용 현대화 프로젝트에는 사용되지 않습니다. 이러한 UI 요소는 CICS BMS 맵, IMS MFS 구성 요소 및 기타 메인프레임 UI 소스에서 가져올 수 있습니다. 프로젝트를 두 개 이상 선택할 수 있습니다.

엔티티 프로젝트 콘텐츠

> **Note**
>
> 다음 설명은 COBOL 및 PL/I 현대화 결과에만 적용됩니다. RPG 현대화 결과는 다른 레이아웃을 기반으로 합니다.

리팩터링이 발생하기 전에 엔티티 프로젝트의 패키지 조직을 현대화된 프로그램에 연결합니다. 다음 두 가지 방법으로 이 작업을 수행할 수 있습니다. 선호하는 방법은 코드 생성 메커니즘을 트리거하기 전에 작동하는 리팩터링 툴박스를 사용하는 것입니다. 이는 고급 작업이며 교육에 설명되어 있습니다. BluAge 자세한 내용은 리팩터링 워크샵을 참조하세요. 이 방법을 사용하면 Java 코드를 나중에 다시 생성할 수 있는 기능을 보존하여 향후 추가 개선 사항 등을 활용할 수 있습니다. 다른 방법은 적용할 수 있는 자바 리팩터링 접근 방식을 사용하여 생성된 소스 코드에서 직접 정기적인 자바 리팩터링을 수행하는 것입니다. 위험은 사용자가 감수해야 합니다.



프로그램 관련 클래스

현대화된 각 프로그램은 비즈니스.컨텍스트 패키지와 비즈니스.모델 패키지라는 두 개의 패키지와 관련되어 있습니다.

- *base package*.*program*.business.context

  business.context 하위 패키지에는 구성 클래스와 컨텍스트 클래스라는 두 개의 클래스가 포함되어 있습니다.

  - 프로그램의 구성 클래스 하나는 해당 프로그램에 대한 특정 구성 세부 정보(예: 문자 기반 데이터 요소를 나타내는 데 사용할 문자 집합, 데이터 구조 요소를 채우는 데 사용되는 기본 바이트 값 등)를 포함합니다. 클래스 이름은 “구성”으로 끝납니다.

`@org.springframework.context.annotation.Configuration` 주석으로 표시되며 올바른 설정 `Configuration` 객체를 반환해야 하는 단일 메서드가 포함되어 있습니다.

Cbact04cConfiguration.java

```java
package aws.bluage.l3.workshop.cbact04c.business.context;

import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;

/**
 * Creates Datasimplifier configuration for the Cbact04cContext context.
 */
@org.springframework.context.annotation.Configuration
@Lazy
public class Cbact04cConfiguration {

    @Bean(name = "Cbact04cContextConfiguration")
    public Configuration configuration() {
        return new ConfigurationBuilder()
                .encoding(Charset.forName("CP1047"))
                .humanReadableEncoding(Charset.forName("ISO-8859-15"))
                .initDefaultByte(0)
                .build();
    }

}

```

- 하나의 컨텍스트 클래스는 프로그램 서비스 클래스(아래 참조) 와 모델 하위 패키지(아래 참조)의 데이터 구조(`Record`) 및 데이터 세트(`File`)를 연결하는 다리 역할을 합니다. 클래스 이름은 “Context”로 끝나며 `RuntimeContext` 클래스의 하위 클래스입니다.

Cbact04cContext.java

```
@Component("aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cContext")
@Import({
    aws.bluage.l3.workshop.cbact04c.business.model.TcatbalFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.XrefFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.DiscgrpFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.AccountFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.TransactFile.class
})
@Lazy
@Scope("prototype")
public class Cbact04cContext extends JicsRuntimeContext {

    @Autowired
    private TcatbalFile tcatbalFile;

    @Autowired
    private XrefFile xrefFile;

    @Autowired
    private DiscgrpFile discgrpFile;

    @Autowired
    private AccountFile accountFile;

    @Autowired
    private TransactFile transactFile;

    private IndexedFile tcatbalFileFile;

    private IndexedFile xrefFileFile;

    private IndexedFile discgrpFileFile;

    private IndexedFile accountFileFile;

    private SequentialFile transactFileFile;

    private TranCatBalRecord tranCatBalRecord;
    private TcatbalfStatus tcatbalfStatus;
    private CardXrefRecord cardXrefRecord;
```

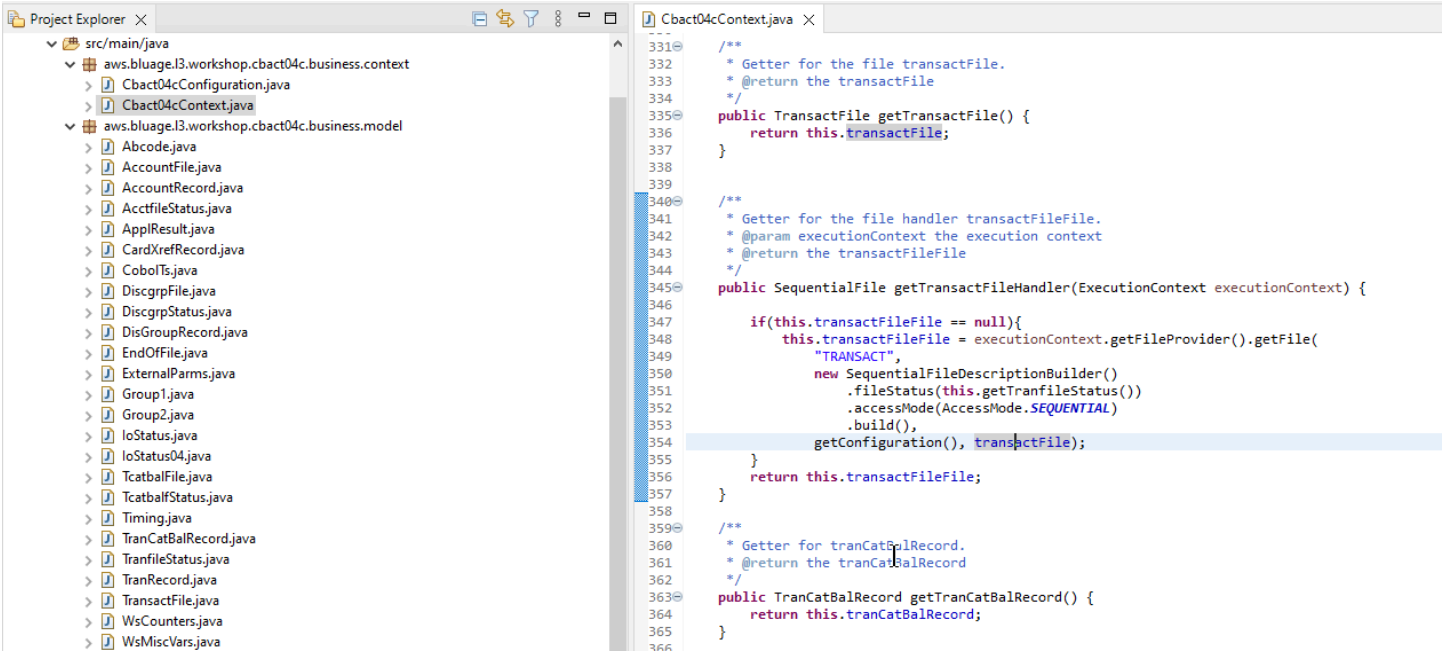
- *base package*.*program*.business.model

  모델 하위 패키지에는 해당 프로그램이 사용할 수 있는 모든 데이터 구조가 들어 있습니다. 예를 들어, 01 수준 COBOL 데이터 구조는 모델 하위 패키지의 클래스에 해당합니다(하위 수준 데이터 구조는 고유한 01 수준 구조의 속성임). 01 데이터 구조를 현대화하는 방법은 데이터 단순화 섹션을 참조하세요.

DiscgrpFile.java

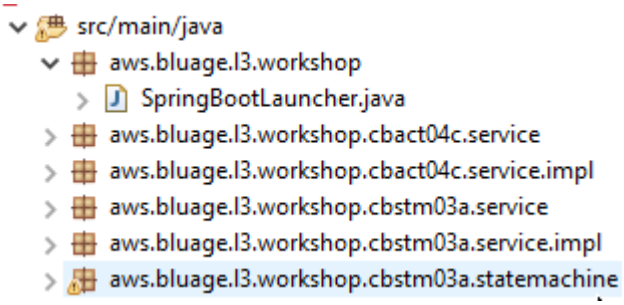
```java
package aws.bluage.l3.workshop.cbact04c.business.model;

import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
import com.netfective.bluage.gapwalk.datasimplifier.data.structure.Elementary;
import com.netfective.bluage.gapwalk.datasimplifier.data.structure.Group;
import com.netfective.bluage.gapwalk.datasimplifier.entity.ElementaryRangeReference;
import com.netfective.bluage.gapwalk.datasimplifier.entity.RangeReference;
import com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity;
import com.netfective.bluage.gapwalk.datasimplifier.metadata.type.AlphanumericType;
import com.netfective.bluage.gapwalk.datasimplifier.metadata.type.ZonedType;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

/**
 * Data simplifier file DiscgrpFile.
 *
 * <p>About 'fdDiscgrpRec' field, <br>uml entity: aws.bluage.l3.workshop.cbact04c.business.model.FdDiscgrpRec
 * <br></p>
 *
 */
@Component("aws.bluage.l3.workshop.cbact04c.business.model.DiscgrpFile")
@Lazy
@Scope("prototype")
public class DiscgrpFile extends RecordEntity {

    private final Group root = new Group(getData());
    private final Group fdDiscgrpRec = new Group(root);
    private final Group fdDiscgrpKey = new Group(fdDiscgrpRec);
    private final Elementary fdDisAcctGroupId = new Elementary(fdDiscgrpKey,new AlphanumericType(10));
    private final Elementary fdDisTranTypeCd = new Elementary(fdDiscgrpKey,new AlphanumericType(2));
    private final Elementary fdDisTranCatCd = new Elementary(fdDiscgrpKey,new ZonedType(4, 0, false));
    private final Elementary fdDiscgrpData = new Elementary(fdDiscgrpRec,new AlphanumericType(34));
```

모든 `RecordEntity` 클래스는 비즈니스 레코드 표현에 대한 액세스를 나타내는 클래스를 확장합니다. 일부 레코드는 `File`에 묶여 있기 때문에 특별한 용도를 가지고 있습니다. `Recorda`와 a 사이의 `File` 바인딩은 파일 객체를 만들 때 컨텍스트 클래스에 있는 해당* FileHandler 메서드에서 이루어집니다. 예를 들어, 다음 목록은 TransactfileFile `File` 가 `Record` (모델 하위 패키지에서) TransactFile에 바인딩되는 방법을 보여줍니다.



서비스 프로젝트 내용

모든 서비스 프로젝트에는 아키텍처의 백본으로 사용되는 전용 Springboot 애플리케이션이 함께 제공됩니다. 이는 서비스 자바 소스의 기본 패키지에 있는 `SpringBootLauncher` 이름이 지정된 클래스를 통해 구체화됩니다.



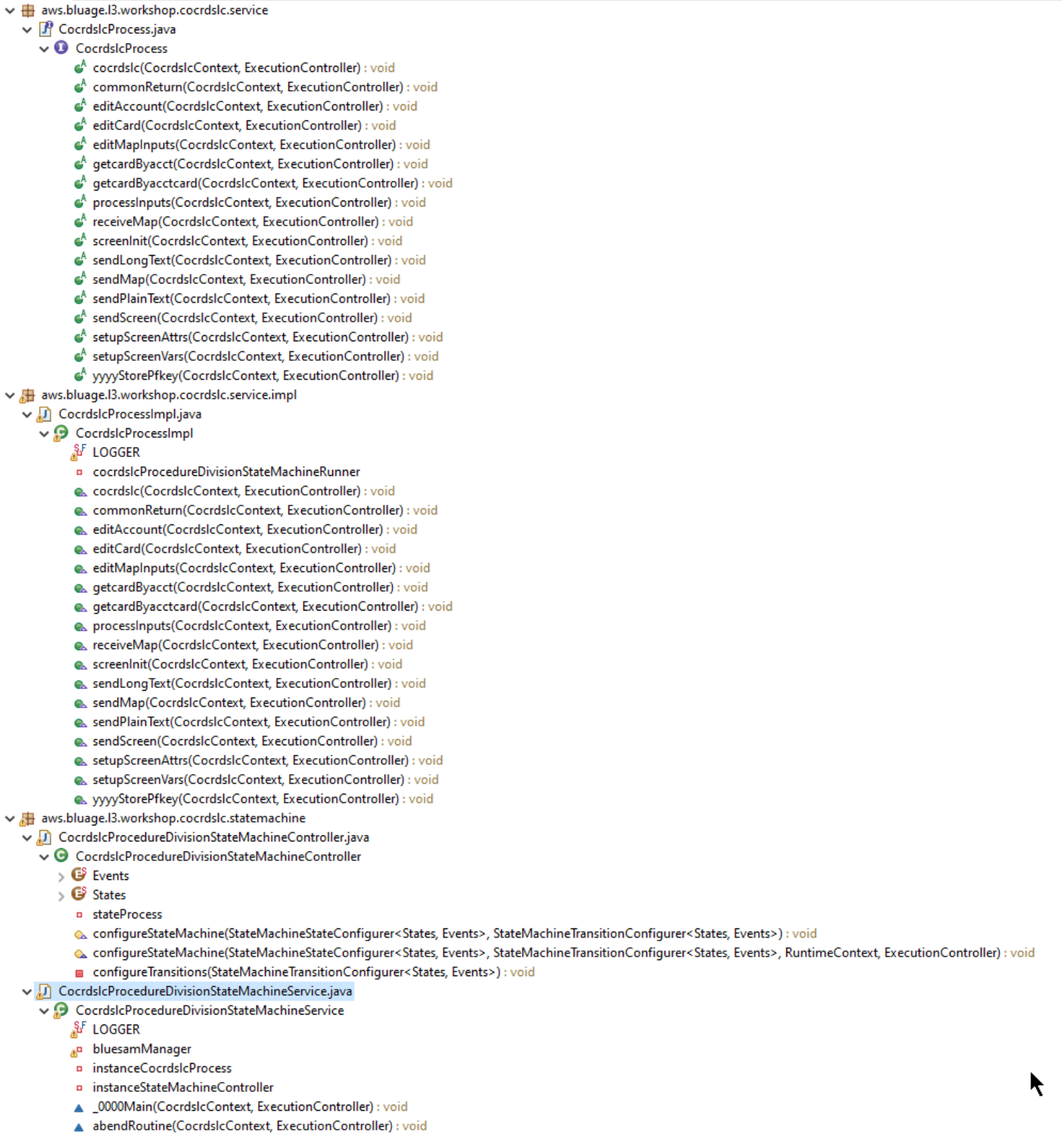
이 수업의 주요 역할은 다음과 같습니다.

- 프로그램 클래스와 관리되는 리소스(데이터 소스/트랜잭션 관리자/데이터 세트 매핑/등...) 를 연결합니다.
- `ConfigurableApplicationContext`를 프로그램에 제공합니다.
- 스프링 컴포넌트로 표시된 모든 클래스 검색(`@Component`).
- 프로그램이 `ProgramRegistry`에 제대로 등록되었는지 확인하세요. 이 등록을 담당하는 initialize 메서드를 참조하세요.

```
/**
 * Initialization method called when the spring application is ready.
 * Register all programs and services to the gapwalk shared context.
 * @param event the application ready event
 */
@EventListener
public void initialize(ApplicationReadyEvent event) {
    Map<String, ProgramContainer> programContainers = event.getApplicationContext().getBeansOfType(ProgramContainer.class);
    programContainers.values().forEach(ProgramRegistry::registerProgram);
    Map<String, ServiceContainer> serviceContainers = event.getApplicationContext().getBeansOfType(ServiceContainer.class);
    serviceContainers.values().forEach(ServiceRegistry::registerService);
}
```

프로그램 관련 아티팩트

사전 리팩터링 없이 비즈니스 로직 현대화 결과는 레거시 프로그램당 2~3개의 패키지로 구성됩니다.



가장 포괄적인 케이스는 세 가지 패키지로 구성됩니다.

- *base package*.*program*.service에는 Program Process라는 인터페이스가 포함되어 있습니다. 여기에는 기존 실행 제어 흐름을 유지하면서 비즈니스 로직을 처리하는 비즈니스 메서드가 포함되어 있습니다.
- *base package*.*program*.service.impl: 앞에서 설명한 Process 인터페이스의 ProcessImpl 구현인 Program이라는 클래스를 포함합니다. 여기에서 기존 명령문을 AWS 블루 에이지 프레임워크에 따라 자바 명령문으로 "번역"합니다.

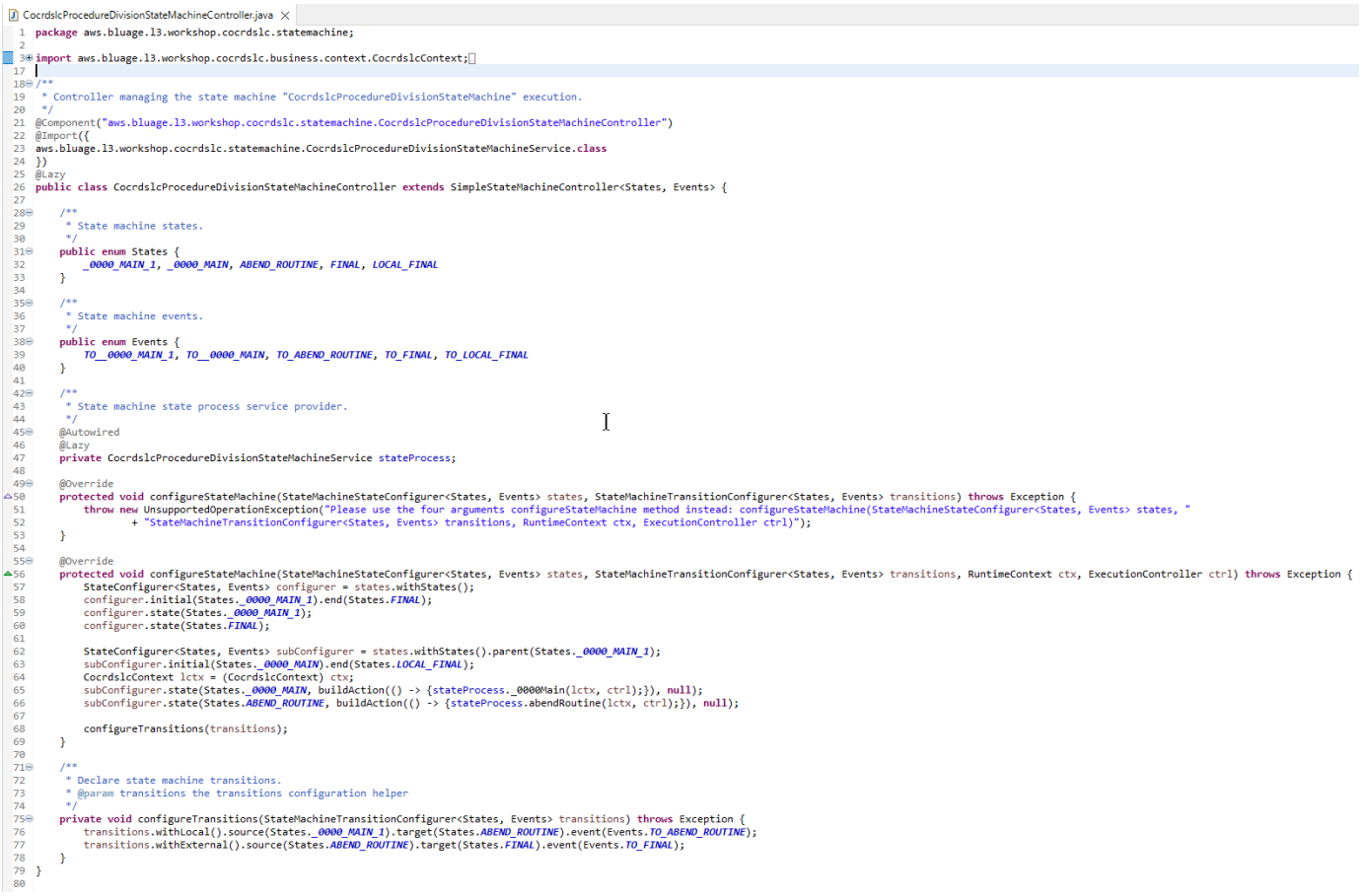
CocrdslcProcessImpl.java

```
210     /**
211      * Process operation sendScreen.
212      *
213      * @param ctx
214      * @param ctrl
215      */
216     @Override
217     public void sendScreen(final CocrdslcContext ctx, final ExecutionController ctrl) {
218         ctx.getCcWorkAreas().setCcardNextMapset(ctx.getWsLiterals().getLitThismapset());
219         ctx.getCcWorkAreas().setCcardNextMap(ctx.getWsLiterals().getLitThismap());
220         ctx.getCarddemoCommarea().setCdemoPgmReenter(true);
221         SendMapBuilder.newInstance(ctx.getDfheiblk(), ctx)
222         .withMap(ctx.getCcWorkAreas().getCcardNextMap())
223         .withMapset(ctx.getCcWorkAreas().getCcardNextMapset())
224         .withData(ctx.getGroup1().getCcrdslaoReference())
225         .withCursor()
226         .withErase()
227         .withFreeKB()
228         .execute();
229         ctx.getWsMiscStorage().setWsRespCd(ctx.getDfheiblk().getEibresp());
230     }
231
232     /**
233      * Process operation processInputs.
234      *
235      * @param ctx
236      * @param ctrl
237      */
238     @Override
239     public void processInputs(final CocrdslcContext ctx, final ExecutionController ctrl) {
240         receiveMap(ctx, ctrl);
241         editMapInputs(ctx, ctrl);
242         ctx.getCcWorkAreas().setCcardErrorMsg(ctx.getWsMiscStorage().getWsReturnMsg());
243         ctx.getCcWorkAreas().setCcardNextProg(ctx.getWsLiterals().getLitThispgm());
244         ctx.getCcWorkAreas().setCcardNextMapset(ctx.getWsLiterals().getLitThismapset());
245         ctx.getCcWorkAreas().setCcardNextMap(ctx.getWsLiterals().getLitThismap());
246     }
247
```

- *base package*.*program*.statemachine: 이 패키지가 항상 있는 것은 아닙니다. 레거시 제어 흐름을 현대화할 때 스테이트 머신 접근 방식 (즉, 스프링 StateMachine 프레임워크 사용) 을 사용하여 레거시 실행 흐름을 적절하게 처리해야 하는 경우에 필요합니다.

  이 경우 상태 시스템 하위 패키지에는 다음과 같은 두 개의 클래스가 포함됩니다.

- *Program*ProcedureDivisionStateMachineController: StateMachineController(스테이트 머신의 실행을 제어하는 데 필요한 작업 정의) 및 StateMachineRunner(스테이트 머신 실행에 필요한 작업 정의) 인터페이스를 구현하는 클래스를 확장하는 클래스로, 예를 들어 샘플 사례의 SimpleStateMachineController와 같이 Spring 스테이트 머신 메카닉을 구동하는 데 사용됩니다.



```
CocrdslcProcedureDivisionStateMachineController.java
package aws.bluage.l3.workshop.cocrdslc.statemachine;

import aws.bluage.l3.workshop.cocrdslc.business.context.CocrdslcContext;

/**
 * Controller managing the state machine "CocrdslcProcedureDivisionStateMachine" execution.
 */
@Component("aws.bluage.l3.workshop.cocrdslc.statemachine.CocrdslcProcedureDivisionStateMachineController")
@Import({
aws.bluage.l3.workshop.cocrdslc.statemachine.CocrdslcProcedureDivisionStateMachineService.class
})
@Lazy
public class CocrdslcProcedureDivisionStateMachineController extends SimpleStateMachineController<States, Events> {

    /**
     * State machine states.
     */
    public enum States {
        _0000_MAIN_1, _0000_MAIN, ABEND_ROUTINE, FINAL, LOCAL_FINAL
    }

    /**
     * State machine events.
     */
    public enum Events {
        TO__0000_MAIN_1, TO__0000_MAIN, TO_ABEND_ROUTINE, TO_FINAL, TO_LOCAL_FINAL
    }

    /**
     * State machine state process service provider.
     */
    @Autowired
    @Lazy
    private CocrdslcProcedureDivisionStateMachineService stateProcess;

    @Override
    protected void configureStateMachine(StateMachineStateConfigurer<States, Events> states, StateMachineTransitionConfigurer<States, Events> transitions) throws Exception {
        throw new UnsupportedOperationException("Please use the four arguments configureStateMachine method instead: configureStateMachine(StateMachineStateConfigurer<States, Events> states, "
                + "StateMachineTransitionConfigurer<States, Events> transitions, RuntimeContext ctx, ExecutionController ctrl)");
    }

    @Override
    protected void configureStateMachine(StateMachineStateConfigurer<States, Events> states, StateMachineTransitionConfigurer<States, Events> transitions, RuntimeContext ctx, ExecutionController ctrl) throws Exception {
        StateConfigurer<States, Events> configurer = states.withStates();
        configurer.initial(States._0000_MAIN_1).end(States.FINAL);
        configurer.state(States._0000_MAIN_1);
        configurer.state(States.FINAL);

        StateConfigurer<States, Events> subConfigurer = states.withStates().parent(States._0000_MAIN_1);
        subConfigurer.initial(States._0000_MAIN).end(States.LOCAL_FINAL);
        CocrdslcContext lctx = (CocrdslcContext) ctx;
        subConfigurer.state(States._0000_MAIN, buildAction(() -> {stateProcess._0000Main(lctx, ctrl);}), null);
        subConfigurer.state(States.ABEND_ROUTINE, buildAction(() -> {stateProcess.abendRoutine(lctx, ctrl);}), null);

        configureTransitions(transitions);
    }

    /**
     * Declare state machine transitions.
     * @param transitions the transitions configuration helper
     */
    private void configureTransitions(StateMachineTransitionConfigurer<States, Events> transitions) throws Exception {
        transitions.withLocal().source(States._0000_MAIN_1).target(States.ABEND_ROUTINE).event(Events.TO_ABEND_ROUTINE);
        transitions.withExternal().source(States.ABEND_ROUTINE).target(States.FINAL).event(Events.TO_FINAL);
    }
}
```

스테이트 머신 컨트롤러는 가능한 다양한 상태와 상태 간의 전환을 정의하여 주어진 프로그램에 대한 기존 실행 제어 흐름을 재현합니다.

상태 머신을 빌드할 때 컨트롤러는 상태 머신 패키지에 있는 관련 서비스 클래스에 정의되고 아래에 설명된 메서드를 참조합니다.

```
subConfigurer.state(States._0000_MAIN, buildAction(() ->
 {stateProcess._0000Main(lctx, ctrl);}), null);
subConfigurer.state(States.ABEND_ROUTINE, buildAction(() ->
 {stateProcess.abendRoutine(lctx, ctrl);}), null);
```

- *Program*ProcedureDivisionStateMachineService: 이 서비스 클래스는 앞에서 설명한 것처럼 상태 시스템 컨트롤러가 생성하는 상태 머신에 바인딩되는 데 필요한 몇 가지 비즈니스 로직을 나타냅니다.

  이 클래스의 메서드에 있는 코드는 스테이트 머신 컨트롤러에 정의된 이벤트를 사용합니다.

CocrdslcProcedureDivisionStateMachineService.java

```
59      /**
60       * State process operation _0000Main.
61       *
62       * @param ctx
63       * @param ctrl
64       */
65      void _0000Main(CocrdslcContext ctx, ExecutionController ctrl) {
66          ctx.getDfheiblk().bind(ArgUtils.get(ctx, 0));
67          ctx.getDfhcommarea().bind(ArgUtils.get(ctx, 1));
68
69          /*
70          ****************************************************************
71          Program:     COCRDSLC.CBL                                     *
72          Layer:       Business logic                                   *
73          Function:    Accept and process credit card detail request    *
74          ****************************************************************
75          Copyright Amazon.com, Inc. or its affiliates.
76          All Rights Reserved.
77          Licensed under the Apache License, Version 2.0 (the \"License\").
78          You may not use this file except in compliance with the License.
79          You may obtain a copy of the License at
80          http://www.apache.org/licenses/LICENSE-2.0
81          Unless required by applicable law or agreed to in writing,
82          software distributed under the License is distributed on an
83          \"AS IS\" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
84          either express or implied. See the License for the specific
85          language governing permissions and limitations under the License
86          ****************************************************************
87          Ver: CardDemo_v1.0-15-g27d6c6f-68 Date: 2022-07-19 23:16:00 CDT */
88          instanceStateMachineController.registerSignalHandler(Events.TO_ABEND_ROUTINE, "!ABEND");
89          HandleAbendBuilder.newInstance(ctx.getDfheiblk(), ctx).execute().handleException();
90          ctx.getCcWorkAreas().getCcWorkAreaReference().getField().initialize();
91          ctx.getWsMiscStorage().getField().initialize();
92          DataUtils.initialize(ctx.getWsCommarea().getWsCommareaReference());
93
```

CocrdslcProcedureDivisionStateMachineService.java

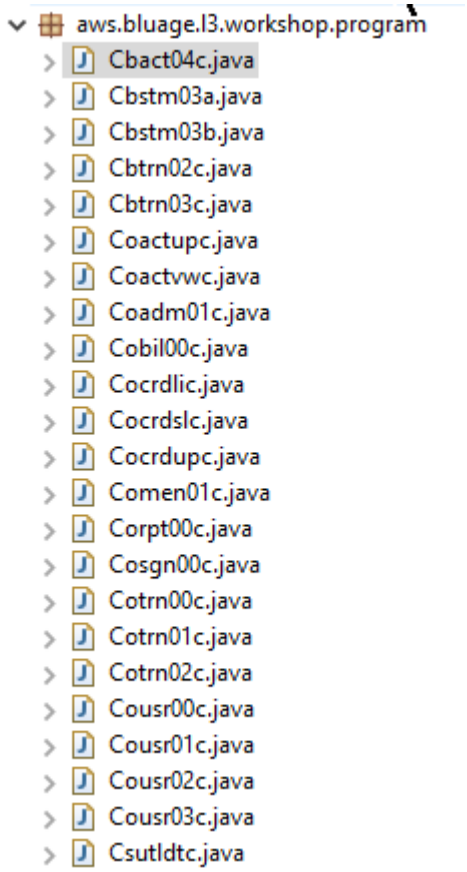
```
221      *
222      * @param ctx
223      * @param ctrl
224      */
225     void abendRoutine(CocrdslcContext ctx, ExecutionController ctrl) {
226         if (DataUtils.isLowValue(ctx.getAbendData().getAbendMsgReference())) {
227             ctx.getAbendData().setAbendMsg("UNEXPECTED ABEND OCCURRED.");
228         }
229         ctx.getAbendData().setAbendCulprit(ctx.getWsLiterals().getLitThispgm());
230         SendTextBuilder.newInstance(ctx.getDfheiblk(), ctx)
231         .withData(ctx.getAbendData())
232         .withLength(134)
233         .execute();
234         HandleAbendBuilder.newInstance(ctx.getDfheiblk(), ctx).cancel().execute().handleException();
235         AbendBuilder.newInstance(ctx.getDfheiblk(), ctx).withAbendCode("9999").execute().handleException();
236
237         /*
238         Ver: CardDemo_v1.0-15-g27d6c6f-68 Date: 2022-07-19 23:12:33 CDT */
239         instanceStateMachineController.sendEvent(Events.TO_FINAL);
240
241     }
242 }
243
```

또한 상태 시스템 서비스는 앞에서 설명한 프로세스 서비스 구현을 직접적으로 호출합니다.

CocrdslcProcedureDivisionStateMachineService.java

```
166
167             /*
168             ****************************************************************
169             COMING FROM CREDIT CARD LIST SCREEN
170             SELECTION CRITERIA ALREADY VALIDATED
171             ****************************************************************** */
172         } else if (ctx.getCarddemoCommarea().isCdemoPgmEnter() && DataUtils.compare(ctx.getCarddemoCommarea().getCdemoFromProgramReference(), ctx.getWsLiterals().getLitCclistpgmReference()) == 0) {
173             ctx.getWsMiscStorage().setInputOk(true);
174             ctx.getCcWorkAreas().setCcAcctIdN(ctx.getCarddemoCommarea().getCdemoAcctId());
175             ctx.getCcWorkAreas().setCcCardNumN(ctx.getCarddemoCommarea().getCdemoCardNum());
176             instanceCocrdslcProcess.getcardByacctcard(ctx, ctrl);
177             instanceCocrdslcProcess.sendMap(ctx, ctrl);
178             instanceCocrdslcProcess.commonReturn(ctx, ctrl);
179         } else if (ctx.getCarddemoCommarea().isCdemoPgmEnter()) {
180
181             /*
182             ****************************************************************
183             COMING FROM SOME OTHER CONTEXT
184             SELECTION CRITERIA TO BE GATHERED
185             ****************************************************************** */
186             instanceCocrdslcProcess.sendMap(ctx, ctrl);
187             instanceCocrdslcProcess.commonReturn(ctx, ctrl);
188         } else if (ctx.getCarddemoCommarea().isCdemoPgmReenter()) {
189             instanceCocrdslcProcess.processInputs(ctx, ctrl);
190             if (ctx.getWsMiscStorage().isInputError()) {
191                 instanceCocrdslcProcess.sendMap(ctx, ctrl);
192                 instanceCocrdslcProcess.commonReturn(ctx, ctrl);
193             } else {
194                 instanceCocrdslcProcess.getcardByacctcard(ctx, ctrl);
195                 instanceCocrdslcProcess.sendMap(ctx, ctrl);
196                 instanceCocrdslcProcess.commonReturn(ctx, ctrl);
197             }
198         } else {
199             ctx.getAbendData().setAbendCulprit(ctx.getWsLiterals().getLitThispgm());
200             ctx.getAbendData().setAbendCode("0001");
201             DataUtils.setToBlank(ctx.getAbendData().getAbendReasonReference());
202             ctx.getWsMiscStorage().setWsReturnMsg("UNEXPECTED DATA SCENARIO");
203             instanceCocrdslcProcess.sendPlainText(ctx, ctrl);
204         }
205
206         /*
207         If we had an error setup error message that slipped through
208         Display and return */
209         if (ctx.getWsMiscStorage().isInputError()) {
210             ctx.getCcWorkAreas().setCcardErrorMsg(ctx.getWsMiscStorage().getWsReturnMsg());
211             instanceCocrdslcProcess.sendMap(ctx, ctrl);
212             instanceCocrdslcProcess.commonReturn(ctx, ctrl);
213         }
214         instanceCocrdslcProcess.commonReturn(ctx, ctrl);
```

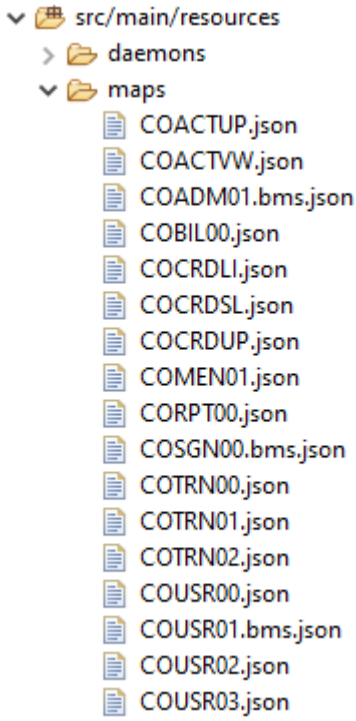
또한 *base package*.program 이름이 지정된 패키지는 프로그램당 하나의 클래스를 수집하여 프로그램 진입점 역할을 하므로 중요한 역할을 합니다(자세한 내용은 나중에 설명 참조). 각 클래스는 프로그램 진입점을 나타내는 표시인 Program 인터페이스를 구현합니다.



기타 아티팩트

- BMS 맵 컴패니언

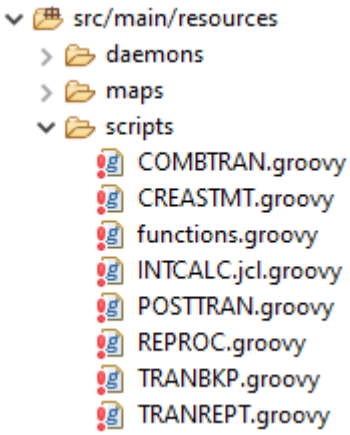
  프로그램 관련 아티팩트 외에도 서비스 프로젝트에는 다양한 용도의 다른 아티팩트가 포함될 수 있습니다. CICS 온라인 애플리케이션을 현대화하는 경우 현대화 프로세스는 json 파일을 생성하여 /src/main/resources 폴더의 맵 폴더에 저장합니다.



Blu Age 런타임은 이러한 json 파일을 사용하여 SEND MAP 문에서 사용하는 레코드를 화면 필드에 바인딩합니다.

- Groovy 스크립트

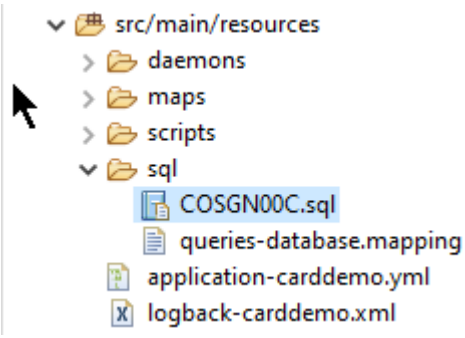
  레거시 응용 프로그램에 JCL 스크립트가 있었다면 이러한 스크립트는 Groovy 스크립트로 현대화되어 /src/main/resources/scripts 폴더에 저장되었습니다(자세한 내용은 나중에 설명).



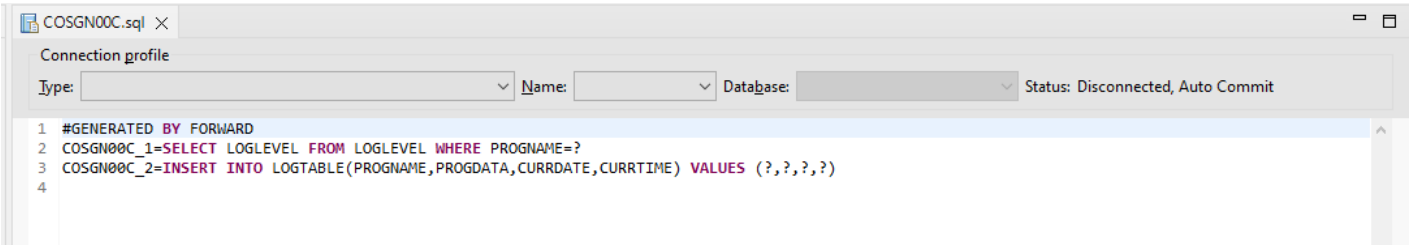
  이러한 스크립트는 일괄 작업(전용, 비대화형, CPU 집약적 데이터 처리 워크로드)을 시작하는 데 사용됩니다.

- SQL 파일

기존 응용 프로그램에서 SQL 쿼리를 사용하는 경우 해당 현대화된 SQL 쿼리가 이름 지정 패턴 프로그램인 .sql을 사용하여 전용 속성 파일에 수집되었습니다. 여기서 프로그램은 해당 쿼리를 사용하는 프로그램의 이름입니다.



이러한 sql 파일의 내용은 현대화된 프로그램이 지정된 쿼리를 실행하는 데 사용하는(키=쿼리) 항목의 모음으로, 각 쿼리는 고유한 키와 연결됩니다.

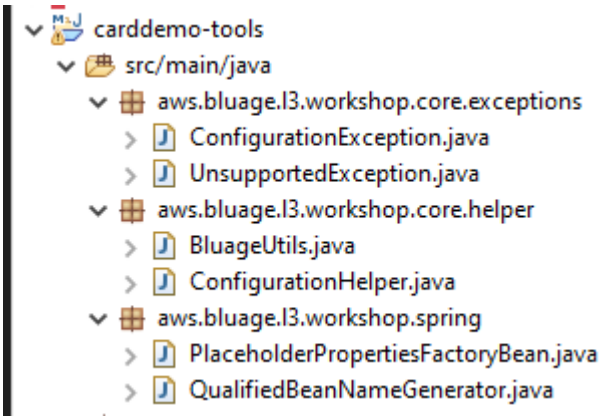


예를 들어, COSGN00C 프로그램은 “COSGN00C_1"(sql 파일의 첫 번째 항목) 키를 사용하여 쿼리를 실행하고 있습니다.

```
327     /**
328      * Process operation getProgramLogLevel.
329      *
330      * ****************************************************************
331      *                       GET PROGRAM LOG LEVEL
332      * ****************************************************************
333      *
334      * @param ctx
335      * @param ctrl
336      */
337     @Override
338     public void getProgramLogLevel(final Cosgn00cContext ctx, final ExecutionController ctrl) {
339         SQLExecutorBuilder.newInstance(ctrl, ctx, ctx.getSqlca())
340             .mapInParameter(SQLParameterBuilder.newInstance(ctx.getLogData().getLogProgramName()).type(JDBCType.CHAR))
341             .mapOutParameter(SQLParameterBuilder.newInstance(ctx.getLogData().getLogProgramLevelReference()))
342             .execute("COSGN00C_1");
343         if (ctx.getSqlca().getSqlcode() == 100) {
344             ctx.getLogData().setLogProgramLevel("N");
345         }
346     }
347
```

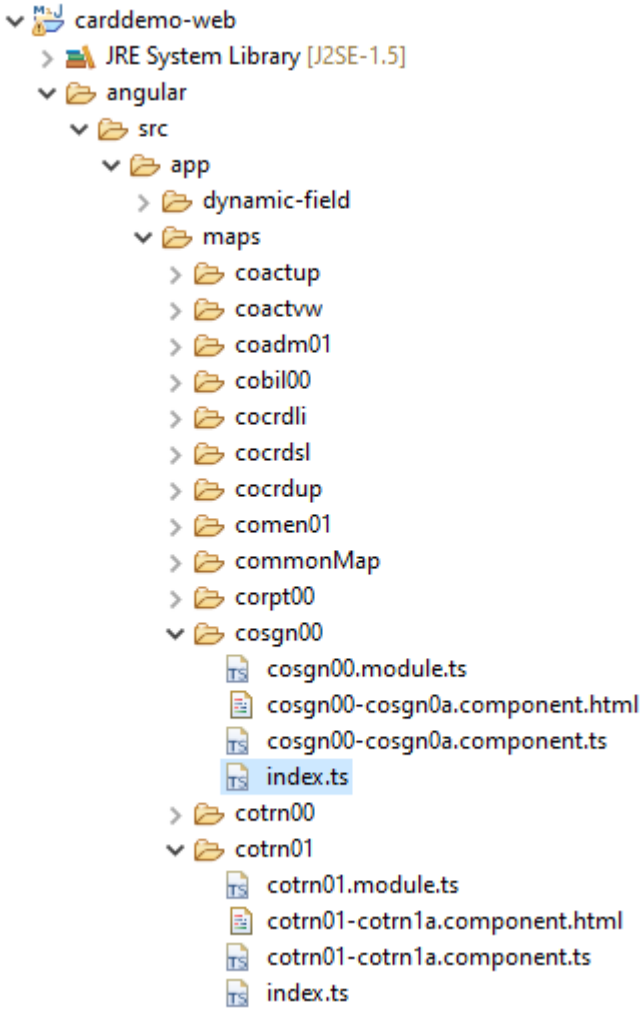
유틸리티 프로젝트 컨텐츠

이름이 “-tools”로 끝나는 유틸리티 프로젝트에는 다른 모든 프로젝트에서 사용할 수 있는 기술 유틸리티 세트가 포함되어 있습니다.



웹 프로젝트 콘텐츠

웹 프로젝트는 레거시 UI 요소를 현대화할 때만 존재합니다. 현대화된 애플리케이션 프론트엔드를 구축하는 데 사용되는 최신 UI 요소는 Angular를 기반으로 합니다. 현대화 아티팩트를 보여주는 데 사용되는 샘플 애플리케이션은 메인프레임에서 실행되는 COBOL/CICS 애플리케이션입니다. CICS 시스템은 맵을 사용하여 UI 화면을 나타냅니다. 해당하는 최신 요소는 모든 맵에 Typescript 파일과 함께 제공되는 html 파일입니다.



웹 프로젝트는 애플리케이션의 프런트엔드 측면만 관리합니다. 유틸리티 및 엔티티 프로젝트에 의존하는 서비스 프로젝트는 백엔드 서비스를 제공합니다. 프론트엔드와 백엔드 간의 연결은 표준 Blu Age 런타임 배포판의 일부인 Gapwalk-Application이라는 웹 애플리케이션을 통해 이루어집니다. AWS

프로그램 실행 및 직접적으로 호출

레거시 시스템에서 프로그램은 COBOL CALL 문과 같은 CALL 메커니즘을 통해 스스로를 직접적으로 호출하고 필요할 때 인수를 전달할 수 있는 독립형 실행 파일로 컴파일됩니다. 관련 아티팩트의 특성이 기존 애플리케이션과 다르기 때문에 현대화된 애플리케이션은 동일한 기능을 제공하지만 다른 접근 방식을 사용합니다.

현대화된 측면에서 보면 프로그램 진입점은 `Program` 인터페이스를 구현하는 특정 클래스이고 Spring 구성 요소(@Component)이며 서비스 프로젝트의 *`base package`*`.program` 이름이 지정된 패키지에 있습니다.

프로그램 등록

현대화된 애플리케이션을 호스팅하는 Tomcat 서버가 시작될 때마다 서비스 Springboot 애플리케이션도 시작되어 프로그램 등록이 트리거됩니다. `ProgramRegistry`라는 이름의 전용 레지스트리는 프로그램 항목으로 채워지며, 각 프로그램은 알려진 프로그램 식별자당 하나의 항목으로 식별자를 사용하여 등록됩니다. 즉, 프로그램이 여러 다른 식별자로 알려진 경우 레지스트리에는 식별자 수만큼의 항목이 포함됩니다.

특정 프로그램의 등록은 () 메서드가 반환한 식별자 모음을 기반으로 합니다. getProgramIdentifiers

Cbact04c.java

```java
package aws.bluage.l3.workshop.program;

import aws.bluage.l3.workshop.SpringBootLauncher;

/**
 * Reference the spring application of program CBACT04C.
 *  Provides an access to the contained program for the run unit.
 */
@Component
@Import({
    aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cConfiguration.class,
    aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cContext.class,
    aws.bluage.l3.workshop.cbact04c.service.impl.Cbact04cProcessImpl.class
})
public class Cbact04c implements Program {
    /**
     * Unique identifiers for the contained program.
     */
    private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("CBACT04C").collect(Collectors.toSet()));

    /**
     * Main program identifier for the contained program.
     */
    private static final String programIdentifier = "CBACT04C";
    @Autowired
    PlatformTransactionManager transactionManager;

    @Autowired
    Map<String, DataSource> datasources;
    @Autowired
    BeanFactory beanFactory;
    /**
     * {@inheritDoc}
     */
    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        return SpringBootLauncher.getCac();
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void updateExecutionContext(ExecutionContext executionContext) {
        executionContext.setDatasources(datasources);
        executionContext.setDatabaseSupport(ExecutionContext.DatabaseSupport.POSTGRE);
        executionContext.setSqlcaVersion(ExecutionContext.SqlcaVersion.getEnum("ansi-comp5"));
        executionContext.setTransactionManager(transactionManager);
        executionContext.setUseSQLDateNewParadigm(true);
        executionContext.setUseSQLTrimStringType(false);
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public Set<String> getProgramIdentifiers() {
        return programIdentifiers;
    }
```

이 예제에서는 프로그램이 'CBACT04C' 이름으로 한 번 등록됩니다(programIdentifiers 컬렉션의 내용 참조). Tomecat 로그에는 모든 프로그램 등록이 표시됩니다. 프로그램 등록은 선언된 프로그램 식별자에만 의존하고 프로그램 클래스 이름 자체에는 의존하지 않습니다. 하지만 일반적으로 프로그램 식별자와 프로그램 클래스 이름은 정렬됩니다.

Blu Age 런타임 배포판의 일부인 다양한 유틸리티 AWS Blu Age 웹 애플리케이션에서 가져온 유틸리티 프로그램에도 동일한 등록 메커니즘이 AWS 적용됩니다. 예를 들어, Gapwalk-Utility-Pgm 웹앱은 z/OS 시스템 유틸리티(IDCAMS, ICEGENER, SORT 등)와 동일한 기능을 제공하며 현대화된 프로그램이나 스크립트로 직접적으로 호출할 수 있습니다. Tomcat 시작 시 등록된 사용 가능한 모든 유틸리티 프로그램은 Tomcat 로그에 기록됩니다.

스크립트 및 데몬 등록

Tomcat 시작 시 /src/main/resources/scripts 폴더 계층 구조에 있는 멋진 스크립트에 대해서도 비슷한 등록 프로세스가 발생합니다. 스크립트 폴더 계층 구조를 탐색하고 검색된 모든 groovy 스크립트(특수 함수.groovy 예약 스크립트 제외)는 짧은 이름(스크립트 파일 이름 중 첫 번째 점 문자 앞에 있는 부분)을 검색 키로 사용하여 `ScriptRegistry`에 등록됩니다.

> **Note**
>
> - 여러 스크립트에 동일한 등록 키를 생성하는 파일 이름이 있는 경우 최신 스크립트만 등록되어 해당 키에 대해 이전에 발생한 등록을 덮어씁니다.
> - 위의 내용을 고려해 볼 때, 등록 메커니즘으로 인해 계층 구조가 단순해지고 예기치 않은 덮어쓰기가 발생할 수 있으므로 하위 폴더를 사용할 때는 주의해야 합니다. 등록 프로세스에는 계층 구조가 포함되지 않습니다. 일반적으로 /scripts/A/myscript.groovy와 /scripts/B/myscript.groovy는 /scripts/A/myscript.groovy를 덮어쓰게 되면 /scripts/B/myscript.groovy를 덮어씁니다.

/src/main/resources/daemons 폴더에 있는 멋진 스크립트는 약간 다르게 처리됩니다. 여전히 일반 스크립트로 등록되어 있지만, Tomcat 시작 시 바로 비동기적으로 한 번 실행됩니다.

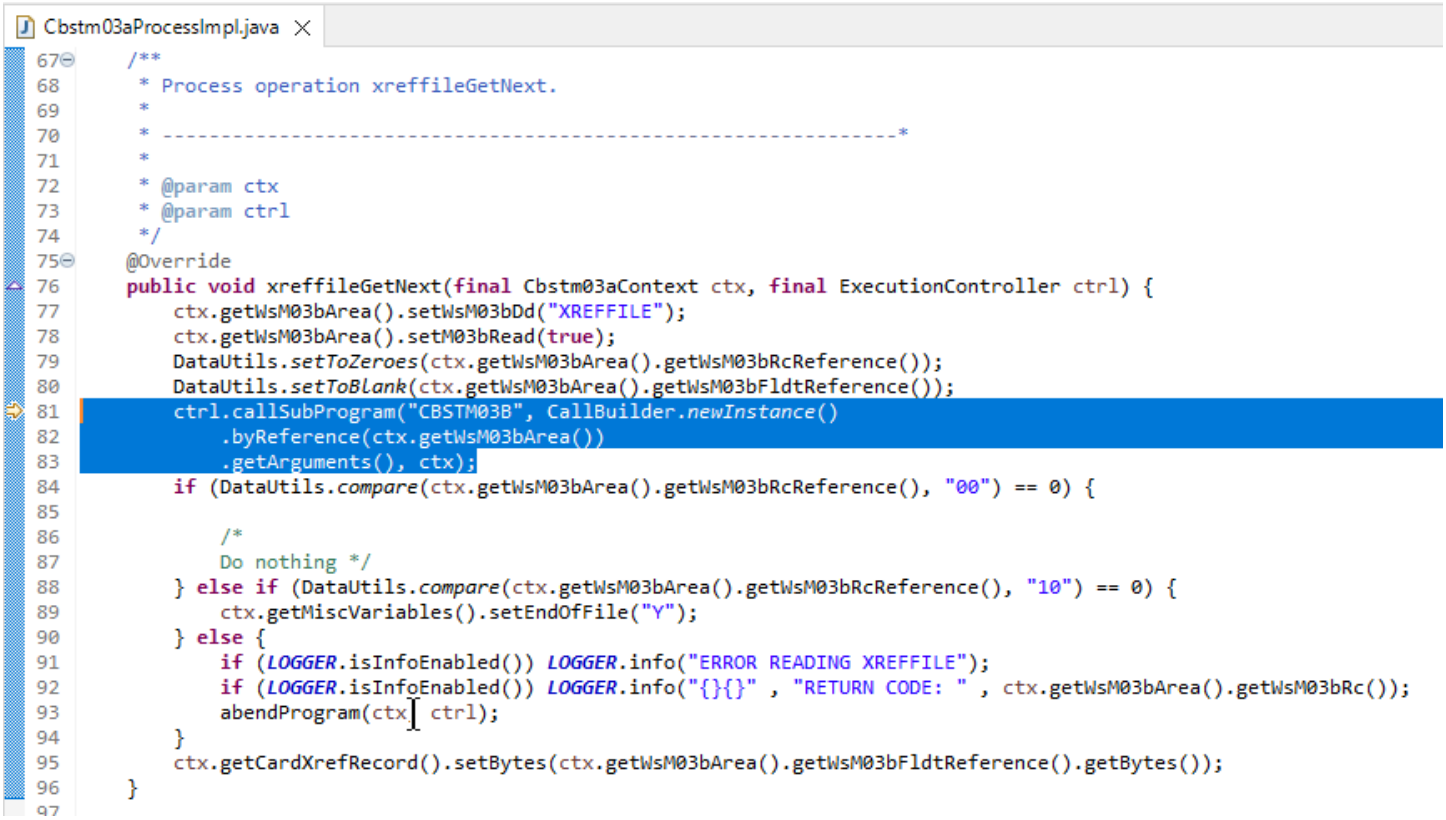
스크립트가 `ScriptRegistry`에 등록된 후에는 Gapwalk-Application에서 제공하는 전용 엔드포인트를 사용하여 REST 직접적으로 호출을 통해 스크립트를 시작할 수 있습니다. 자세한 내용을 알아보려면 해당 문서를 참조하세요.

프로그램 직접적 호출 프로그램

각 프로그램은 다른 프로그램을 하위 프로그램으로 직접 호출하여 파라미터를 전달할 수 있습니다. 프로그램은 이를 위해 `ExecutionController` 인터페이스 구현(대부분의 경우 `ExecutionControllerImpl` 인스턴스가 됨)을 사용하고 프로그램 직접적 호출 인수를 구축하기 위해 작성하기 위해 `CallBuilder`라는 유창한 API 메커니즘과 함께 인터페이스 구현을 사용합니다.

모든 프로그램 메서드는 `RuntimeContext` 및 `ExecutionController`를 메서드 인수로 모두 사용하므로 `ExecutionController`를 항상 사용하여 다른 프로그램을 직접적으로 호출할 수 있습니다.

예를 들어, CBST03A 프로그램이 CBST03B 프로그램을 하위 프로그램으로 직접적으로 호출하여 파라미터를 전달하는 방법을 보여주는 다음 다이어그램을 참조하세요.



- `ExecutionController.callSubProgram`의 첫 번째 인수는 직접적으로 호출할 프로그램의 식별자입니다. 즉, 프로그램 등록에 사용되는 식별자 중 하나입니다. 위 단락 참조).
- `CallBuilder`를 기반으로 빌드한 결과인 두 번째 인수는 직접적으로 호출자에서 수신자에게 전달된 데이터에 해당하는 `Record`의 배열입니다.
- 세 번째이자 마지막 인수는 직접적 호출자 `RuntimeContext` 인스턴스입니다.

세 인수는 모두 필수이며 null이 될 수 없지만 두 번째 인수는 빈 배열일 수 있습니다.

피호출자는 원래 그렇게 하도록 설계된 경우에만 전달된 파라미터를 처리할 수 있습니다. 레거시 COBOL 프로그램의 경우 LINKAGE 섹션과 LINKAGE 요소를 사용할 수 있도록 프로시저 분할을 위한 USING 조항이 있어야 합니다.

예를 들어, 해당하는 CBSTM03B .CBL 코볼 소스 파일을 참조하세요.

github.com/aws-samples/aws-mainframe-modernization-carddemo/blob/main/app/cbl/CBSTM03B.CBL

```
 98
 99          LINKAGE SECTION.
100          01  LK-M03B-AREA.
101              05  LK-M03B-DD          PIC X(08).
102              05  LK-M03B-OPER        PIC X(01).
103                88  M03B-OPEN       VALUE 'O'.
104                88  M03B-CLOSE      VALUE 'C'.
105                88  M03B-READ       VALUE 'R'.
106                88  M03B-READ-K     VALUE 'K'.
107                88  M03B-WRITE      VALUE 'W'.
108                88  M03B-REWRITE    VALUE 'Z'.
109              05  LK-M03B-RC          PIC X(02).
110              05  LK-M03B-KEY         PIC X(25).
111              05  LK-M03B-KEY-LN      PIC S9(4).
112              05  LK-M03B-FLDT        PIC X(1000).
113
114          PROCEDURE DIVISION USING LK-M03B-AREA.
115
```

따라서 CBSTM03B 프로그램은 단일 `Record` 값을 파라미터로 사용합니다(크기 1의 배열). 이것은 byReference() 및 getArguments() 메서드 체인을 사용하여 `CallBuilder`가 빌드하고 있는 것입니다.

`CallBuilder` Fluent API 클래스에는 피호출자에게 전달할 인수 배열을 채우는 데 사용할 수 있는 여러 메서드가 있습니다.

- asPointer (RecordAdaptable): 포인터 종류의 인수를 참조로 추가합니다. 포인터는 대상 데이터 구조의 주소를 나타냅니다.
- byReference (RecordAdaptable): 참조로 인수를 추가합니다. 호출자는 피호출자가 수행한 수정 내용을 볼 수 있습니다.
- byReference (RecordAdaptable...): 이전 메서드의 varargs 변형입니다.
- byValue(Object): 값을 기준으로 `Record`로 변환된 인수를 추가합니다. 호출자는 피호출자가 수행한 수정 내용을 볼 수 없습니다.

- byValue (RecordAdaptable): 이전 메서드와 동일하지만 인수는 a로 직접 사용할 수 있습니다. `RecordAdaptable`
- byValueWith경계 (Object, int, int): `Record` a로 변환된 인수를 추가하고 주어진 경계로 정의된 바이트 배열 부분을 값별로 추출합니다.

마지막으로 getArguments 메서드는 추가된 모든 인수를 수집하여 `Record`의 배열로 반환합니다.

> **Note**
>
> 인수 배열에 필요한 크기가 맞는지, 항목이 적절하게 정렬되고, 메모리 레이아웃이 연결 요소에 대한 예상 레이아웃과 호환되는지 확인하는 것은 직접적으로 호출자의 책임입니다.

프로그램을 직접적으로 호출하는 스크립트

groovy 스크립트에서 등록된 프로그램을 직접적으로 호출하려면 `MainProgramRunner` 인터페이스를 구현하는 클래스 인스턴스를 사용해야 합니다. 일반적으로 이러한 인스턴스는 Spring 사용을 통해 얻을 수 있습니다. ApplicationContext

REPROC.groovy

```groovy
// Import
import com.netfective.bluage.gapwalk.rt.provider.ScriptRegistry
import com.netfective.bluage.gapwalk.rt.call.MainProgramRunner
import com.netfective.bluage.gapwalk.io.support.FileConfigurationUtils
import com.netfective.bluage.gapwalk.rt.job.support.DefaultJobContext
import com.netfective.bluage.gapwalk.rt.utils.GroovyUtils
import com.netfective.bluage.gapwalk.rt.io.support.FileConfiguration
import com.netfective.bluage.gapwalk.rt.shared.AbendException
import com.netfective.bluage.gapwalk.rt.call.exception.GroovyExecutionException
// Variables
mpr = applicationContext.getBean("com.netfective.bluage.gapwalk.rt.call.ExecutionController", MainProgramRunner.class)
TreeMap mapTransfo = [:]
```

`MainProgramRunner` 인터페이스를 사용할 수 있게 되면 runProgram 메서드를 사용하여 프로그램을 직접적으로 호출하고 대상 프로그램의 식별자를 파라미터로 전달합니다.

REPROC.groovy

```
50 //*********************************************************************
51 //*                              STEPS                                *
52 //*********************************************************************
53 // STEP PRC001 - PGM - IDCAMS*****************************************************
54 def stepPRC001(Object shell, Map params, Map programResults){
55     shell.with {
56         if (checkValidProgramResults(programResults)) {
57             return execStep("PRC001", "IDCAMS", programResults, {
58                 mpr
59                     .withFileConfigurations(new FileConfigurationUtils()
60                         .systemOut("SYSPRINT")
61                         .output("*")
62                         .build()
63                         .bluesam("FILEIN")
64                         .dataset("NULLFILE")
65                         .disposition("SHR")
66                         .build()
67                         .bluesam("FILEOUT")
68                         .dataset("NULLFILE")
69                         .disposition("SHR")
70                         .build()
71                         .fileSystem("SYSIN")
72                         .path("&CNTLLIB(REPROCT)")
73                         .disposition("SHR")
74                         .build()
75                         .getFileConfigurations(fcmap))
76                     .withParameters(params)
77                     .runProgram("IDCAMS")
78                 })
79         }
80     }
81 }
```

이전 예제에서 작업 단계는 IDCAMS(파일 처리 유틸리티 프로그램)를 직접적으로 호출하여 실제 데이터 세트 정의와 해당 논리적 식별자 간의 매핑을 제공합니다.

데이터 세트를 처리할 때 기존 프로그램은 대부분 논리적 이름을 사용하여 데이터 세트를 식별합니다. 스크립트에서 프로그램을 직접적으로 호출할 때 스크립트는 논리적 이름을 실제 물리적 데이터 세트와 매핑해야 합니다. 이러한 데이터 세트는 파일 시스템, Blusam 저장소에 있을 수도 있고 인라인 스트림, 여러 데이터 세트의 연결 또는 GDG 생성으로 정의될 수도 있습니다.

withFileConfiguration 메서드를 사용하여 데이터 세트의 논리적-물리적 맵을 작성하고 호출된 프로그램에서 사용할 수 있도록 합니다.

직접 프로그램을 작성하세요

스크립트나 기타 현대화된 프로그램을 직접적으로 호출할 프로그램을 직접 작성하는 것은 일반적인 작업입니다. 일반적으로 현대화 프로젝트에서는 실행 가능한 레거시 프로그램이 현대화 프로세스에서

지원하지 않는 언어로 작성되었거나 소스가 손실된 경우(예, 그럴 수 있음) 또는 프로그램이 소스를 사용할 수 없는 유틸리티인 경우 프로그램을 직접 작성합니다.

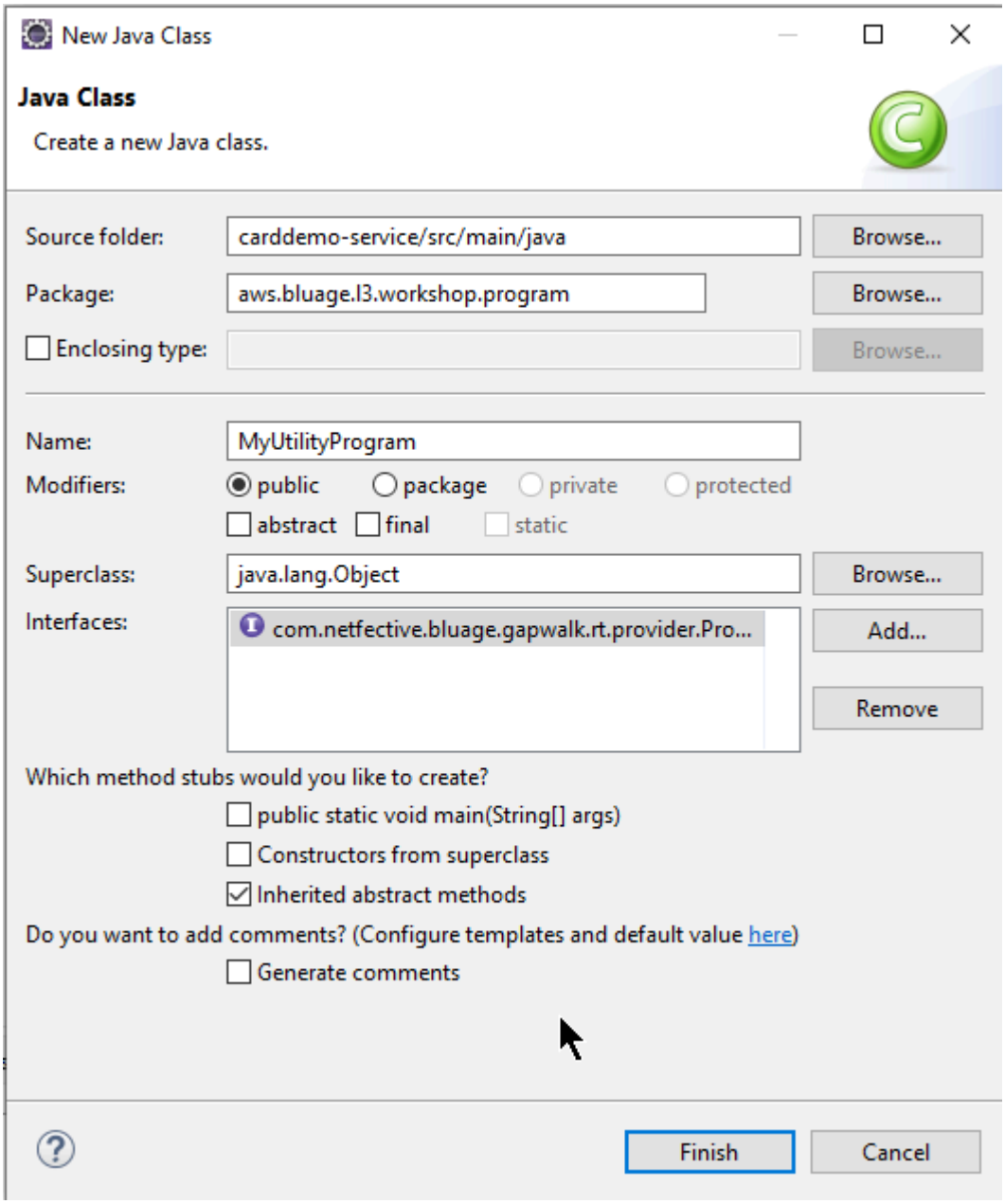
이 경우 누락된 프로그램을 Java로 직접 작성해야 할 수도 있습니다(프로그램 예상 동작이 무엇인지, 프로그램 인수의 메모리 레이아웃(있는 경우) 등에 대해 충분히 알고 있다고 가정하면 됩니다 Java 프로그램은 이 문서에 설명된 프로그램 메커니즘을 준수해야 다른 프로그램과 스크립트에서 실행할 수 있습니다.

프로그램을 사용할 수 있게 하려면 다음 두 가지 필수 단계를 완료해야 합니다.

- 등록 및 직접적 호출이 가능하도록 `Program` 인터페이스를 제대로 구현하는 클래스를 작성하세요.
- 다른 프로그램/스크립트에서도 볼 수 있도록 프로그램이 제대로 등록되었는지 확인하세요.

프로그램 구현 작성

IDE를 사용하여 `Program` 인터페이스를 구현하는 새 Java 클래스를 만드세요.



다음 이미지는 구현해야 할 모든 필수 메서드를 생성하는 Eclipse IDE를 보여줍니다.

MyUtilityProgram.java

```
1  package aws.bluage.l3.workshop.program;
2
3  import java.util.Set;
10
11 public class MyUtilityProgram implements Program {
12
13     @Override
14     public ConfigurableApplicationContext getSpringApplication() {
15         // TODO Auto-generated method stub
16         return null;
17     }
18
19     @Override
20     public Set<String> getProgramIdentifiers() {
21         // TODO Auto-generated method stub
22         return null;
23     }
24
25     @Override
26     public Context getContext() {
27         // TODO Auto-generated method stub
28         return null;
29     }
30
31     @Override
32     public void run(ExecutionController ctrl) {
33         // TODO Auto-generated method stub
34
35     }
36
37 }
38
```

Spring 통합

먼저 클래스를 Spring 컴포넌트로 선언해야 합니다. 클래스에 `@Component` 주석을 달아주세요.

```
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.stereotype.Component;

import com.netfective.bluage.gapwalk.rt.call.ExecutionController;
import com.netfective.bluage.gapwalk.rt.context.Context;
import com.netfective.bluage.gapwalk.rt.provider.Program;

import aws.bluage.l3.workshop.SpringBootLauncher;

@Component
public class MyUtilityProgram implements Program {
```

다음으로 필요한 메서드를 제대로 구현하세요. 이 샘플의 맥락에서 현대화된 모든 프로그램이 이미 들어 있는 패키지에 `MyUtilityProgram`를 추가했습니다. 이 배치를 통해 프로그램은 기존 Springboot

애플리케이션을 사용하여 getSpringApplication 메서드 `ConfigurableApplicationContext` 구현에 필요한 것을 제공할 수 있습니다.

```
public class MyUtilityProgram implements Program {

    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        return SpringBootLauncher.getCac();
    }
```

자체 프로그램을 위해 다른 위치를 선택할 수도 있습니다. 예를 들어, 특정 프로그램을 다른 전담 서비스 프로젝트에서 찾을 수 있습니다. 주어진 서비스 프로젝트에 자체 Springboot 애플리케이션이 있는지 확인하세요. 그러면 ApplicationContext (a여야 `ConfigurableApplicationContext` 함) 을 검색할 수 있습니다.

프로그램에 ID 부여

다른 프로그램 및 스크립트에서 직접적으로 호출할 수 있으려면 프로그램에 하나 이상의 식별자를 지정해야 합니다. 식별자는 시스템 내에 등록된 다른 기존 프로그램과 충돌하지 않아야 합니다. 기존 레거시 프로그램을 대체해야 하는 이유 때문에 식별자를 선택할 수 있습니다. 이 경우 기존 프로그램 전체에서 발견된 CALL 발생 횟수에 따라 예상 식별자를 사용해야 합니다. 레거시 시스템에서는 대부분의 프로그램 식별자 길이가 8자입니다.

프로그램에서 수정할 수 없는 식별자 세트를 만드는 것도 이 작업을 수행하는 한 가지 방법입니다. 다음 예에서는 “MYUTILPG”를 단일 식별자로 선택하는 방법을 보여 줍니다.

```
@Component
public class MyUtilityProgram implements Program {

    /**
     * Unique identifiers for the contained program.
     */
    private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("MYUTILPG").collect(Collectors.toSet()));


    public ConfigurableApplicationContext getSpringApplication() {

    @Override
    public Set<String> getProgramIdentifiers() {
        return programIdentifiers;
    }
```

프로그램을 컨텍스트에 연결

프로그램에는 컴패니언 `RuntimeContext` 인스턴스가 필요합니다. 현대화된 프로그램의 경우 AWS Blu Age는 레거시 프로그램의 일부인 데이터 구조를 사용하여 컴패니언 컨텍스트를 자동으로 생성합니다.

프로그램을 직접 작성하는 경우 컴패니언 컨텍스트도 작성해야 합니다.

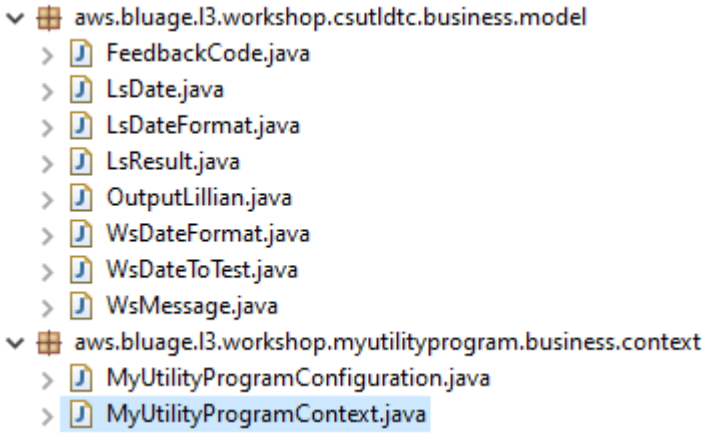
프로그램 관련 클래스를 참조하면 프로그램에는 최소한 두 개의 컴패니언 클래스가 필요하다는 것을 알 수 있습니다.

- 새 구성 클래스.
- 구성을 사용하는 컨텍스트 클래스.

유틸리티 프로그램에서 추가 데이터 구조를 사용하는 경우 해당 데이터 구조도 작성하여 컨텍스트에서 사용해야 합니다.

이러한 클래스는 Spring 프레임워크에서 컨텍스트 구성 요소 및 구성을 처리하도록 하려면 응용 프로그램 시작 시 스캔되는 패키지 계층 구조의 일부인 패키지에 있어야 합니다.

엔티티 프로젝트에서 새로 만든 *base package*`.myutilityprogram.business.context` 패키지에 최소한의 구성과 컨텍스트를 작성해 보겠습니다.



다음은 구성 내용입니다. 인근의 현대화된 다른 프로그램과 유사한 구성 빌드를 사용하고 있습니다. 특정 요구 사항에 맞게 사용자 지정해야 할 수도 있습니다.

MyUtilityProgramConfiguration.java

```java
package aws.bluage.l3.workshop.myutilityprogram.business.context;

import java.nio.charset.Charset;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Lazy;

import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
import com.netfective.bluage.gapwalk.datasimplifier.configuration.ConfigurationBuilder;

/**
 * Creates Datasimplifier configuration for the MyUtilityProgram context.
 */
@org.springframework.context.annotation.Configuration
@Lazy
public class MyUtilityProgramConfiguration {

    @Bean(name = "MyUtilityProgramContextConfiguration")
    public Configuration configuration() {
        return new ConfigurationBuilder()
                .encoding(Charset.forName("CP1047"))
                .humanReadableEncoding(Charset.forName("ISO-8859-15"))
                .initDefaultByte(0)
                .build();
    }
}
```

참고:

- 일반적인 명명 규칙은 구성입니다. ProgramName
- @org.springframework.context.annotation.Configuration 및 @Lazy 주석을 사용해야 합니다.
- 콩 이름은 일반적으로 ProgramNameContextConfiguration 규칙을 따르지만 필수 사항은 아닙니다. 프로젝트 전체에서 빈 이름이 충돌하지 않도록 하세요.
- 구현할 단일 메서드는 `Configuration` 객체를 반환해야 합니다. `ConfigurationBuilder` 유창한 API를 사용하면 쉽게 만들 수 있습니다.

그리고 관련 컨텍스트:

MyUtilityProgramContext.java

```java
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
import com.netfective.bluage.gapwalk.rt.context.RuntimeContext;

@Component("aws.bluage.l3.workshop.mytuilityprogram.business.context.MyUtilityProgramContext")
@Lazy
@Scope("prototype")
public class MyUtilityProgramContext extends RuntimeContext{

    protected MyUtilityProgramContext(@Qualifier("MyUtilityProgramContextConfiguration") Configuration configuration) {
        super(configuration);
    }

    @Override
    public void cleanUp() {
        // TODO implement clean-up of associated data structures if any
    }

    @Override
    protected void doReset() {
        // TODO implement reset of associated data structures if any
    }

}
```

참고

- 컨텍스트 클래스는 기존 `Context` 인터페이스 구현(`RuntimeContext` 또는 `JicsRuntimeContext`, JICS 전용 `RuntimeContext` 항목으로 개선됨)을 확장해야 합니다.
- 일반 명명 규칙은 ProgramName컨텍스트입니다.
- 프로토타입 구성 요소로 선언하고 @Lazy 주석을 사용해야 합니다.
- 생성자는 @Qualifier 주석을 사용하여 적절한 구성 클래스를 대상으로 하는 관련 구성을 참조합니다.
- 유틸리티 프로그램에서 몇 가지 추가 데이터 구조를 사용하는 경우 다음과 같아야 합니다.
  - 작성되어 *`base package`*`.business.model` 패키지에 추가되었습니다.
  - 컨텍스트에서 참조되었습니다. 기존의 다른 컨텍스트 클래스를 살펴보고 필요에 따라 데이터 구조 클래스를 참조하고 컨텍스트 메서드(생성자/정리/재설정) 를 조정하는 방법을 살펴보세요.

이제 전용 컨텍스트를 사용할 수 있으므로 새 프로그램에서 해당 컨텍스트를 사용하도록 해 보겠습니다.

MyUtilityProgram.java

```
import aws.bluage.l3.workshop.SpringBootLauncher;

@Component
@Import({
    aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramConfiguration.class,
    aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramContext.class
})
public class MyUtilityProgram implements Program {

    @Autowired
    BeanFactory beanFactory;

    /**
     * Unique identifiers for the contained program.
     */
    private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("MYUTILPG").collect(Collectors.toSet()));

    private static final String programIdentifier = "MYUTILPG";

    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        return SpringBootLauncher.getCac();
    }

    @Override
    public Set<String> getProgramIdentifiers() {
        return programIdentifiers;
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public String getProgramMainIdentifier() {
        return programIdentifier;
    }


    @Override
    public Context getContext() {
        return ProgramContextStore.getOrCreate(
        getProgramMainIdentifier(),
        aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramContext.class,
        beanFactory);
    }

```

참고:

- GetContext 메서드는 `ProgramContextStore` 클래스의 getOrCreate 메서드와 자동으로 연결된 Spring을 사용하여 표시된 대로 엄격하게 구현되어야 합니다. `BeanFactory` `ProgramContextStore`에 프로그램 컨텍스트를 저장하는 데 단일 프로그램 식별자가 사용됩니다. 이 식별자를 '프로그램 기본 식별자'라고 합니다.
- 동반 구성 및 컨텍스트 클래스는 `@Import` 스프링 주석을 사용하여 참조해야 합니다.

비즈니스 로직 구현

프로그램 스켈레톤이 완료되면 새 유틸리티 프로그램에 대한 비즈니스 로직을 구현하세요.

프로그램의 `run` 메서드로 이 작업을 수행하세요. 이 메서드는 다른 프로그램이나 스크립트에서 프로그램을 직접적으로 호출할 때마다 실행됩니다.

즐거운 코딩 되세요!

프로그램 등록 처리

마지막으로 새 프로그램이 `ProgramRegistry`에 제대로 등록되었는지 확인하세요. 이미 다른 프로그램이 들어 있는 패키지에 새 프로그램을 추가한 경우에는 더 이상 할 일이 없습니다. 응용 프로그램 시작 시 새 프로그램이 선택되어 모든 인접 프로그램에 등록됩니다.

다른 프로그램 위치를 선택한 경우 Tomcat 시작 시 프로그램이 제대로 등록되었는지 확인해야 합니다. 이를 구현하는 방법에 대한 영감을 얻으려면 서비스 프로젝트에서 생성된 SpringbootLauncher 클래스의 initialize 메서드를 살펴보세요 (참조). 서비스 프로젝트 내용

Tomecat 시작 로그를 확인하세요. 모든 프로그램 등록이 기록됩니다. 프로그램이 성공적으로 등록되면 일치하는 로그 항목을 찾을 수 있습니다.

프로그램이 제대로 등록되었는지 확인되면 비즈니스 로직 코딩에 대한 반복 작업을 시작할 수 있습니다.

완전한 이름 매핑

이 섹션에는 AWS Blu Age 목록과 현대화된 애플리케이션에서 사용할 수 있는 타사의 정식 이름 매핑 목록이 수록되어 있습니다.

AWS Blu Age 정식 이름 매핑.

짧은 이름	정규화된 인덱스 이름
`CallBuilder`	`com.netfective.bluage.gapwalk.runtime.statements.CallBuilder`
`Configuration`	`com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration`
`ConfigurationBuilder`	`com.netfective.bluage.gapwalk.datasimplifier.configuration.ConfigurationBuilder`
`ExecutionController`	`com.netfective.bluage.gapwalk.rt.call.ExecutionController`

짧은 이름	정규화된 인덱스 이름
ExecutionControllerImpl	com.netfective.bluage.gapwalk.rt.call.internal.ExecutionControllerImpl
File	com.netfective.bluage.gapwalk.rt.io.File
MainProgramRunner	com.netfective.bluage.gapwalk.rt.call.MainProgramRunner
Program	com.netfective.bluage.gapwalk.rt.provider.Program
ProgramContextStore	com.netfective.bluage.gapwalk.rt.context.ProgramContextStore
ProgramRegistry	com.netfective.bluage.gapwalk.rt.provider.ProgramRegistry
Record	com.netfective.bluage.gapwalk.datasimplifier.data.Record
RecordEntity	com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity
RuntimeContext	com.netfective.bluage.gapwalk.rt.context.RuntimeContext
SimpleStateMachineController	com.netfective.bluage.gapwalk.rt.statemachine.SimpleStateMachineController
StateMachineController	com.netfective.bluage.gapwalk.rt.statemachine.StateMachineController

짧은 이름	정규화된 인덱스 이름
StateMachineRunner	com.netfective.bluage.gapwalk.rt.statemachine.StateMachineRunner

타사의 완전한 자격을 갖춘 이름 매핑

짧은 이름	정규화된 인덱스 이름
@Autowired	org.springframework.beans.factory.annotation.Autowired
@Bean	org.springframework.context.annotation.Bean
BeanFactory	org.springframework.beans.factory.BeanFactory
@Component	org.springframework.stereotype.Component
ConfigurableApplicationContext	org.springframework.context.ConfigurableApplicationContext
@Import	org.springframework.context.annotation.Import
@Lazy	org.springframework.context.annotation.Lazy

데이터 단순화

메인프레임 및 미드레인지 시스템(다음 항목에서는 “레거시” 시스템이라고 함)에서 자주 사용되는 COBOL, PL/I 또는 RPG와 같은 프로그래밍 언어는 메모리에 대한 저수준 액세스를 제공합니다. 이 액세스는 그룹 또는 배열을 통해 집계될 수도 있는 영역 지정, 압축 또는 영숫자와 같은 기본 유형을 통해 액세스하는 메모리 레이아웃에 중점을 둡니다.

지정된 프로그램에서는 입력된 필드를 통해 그리고 바이트에 대한 직접 액세스(원시 메모리)를 통해 주어진 메모리에 액세스하는 방식이 공존합니다. 예를 들어 COBOL 프로그램은 호출자에게 인수를 연속 바이트 세트 (LINKAGE) 로 전달하거나 동일한 방식 (레코드) 으로 파일에서 데이터를 읽고 쓰는 한편, 카피북으로 구성된 유형이 지정된 필드로 이러한 메모리 범위를 해석합니다.

이러한 원시 및 구조적 메모리 액세스의 조합, 정확한 바이트 수준의 메모리 레이아웃에 대한 의존성, 영역 지정 또는 패킹과 같은 레거시 유형은 Java 프로그래밍 환경에서 기본적으로 사용할 수도 없고 쉽게 사용할 수도 없는 기능입니다.

기존 프로그램을 Java로 현대화하기 위한 AWS Blu Age 솔루션의 일부인 Data Simplifier 라이브러리는 이러한 구조를 현대화된 Java 프로그램에 제공하고 Java 개발자에게 최대한 친숙한 방식 (게터/세터, 바이트 배열, 클래스 기반) 으로 이러한 구조를 제공합니다. 이는 이러한 프로그램에서 생성된 현대화된 Java 코드의 핵심 종속성입니다.

간단히 설명하자면, 다음 설명의 대부분은 COBOL 구조를 기반으로 하지만 대부분의 개념이 비슷하므로 PL1과 RPG 데이터 레이아웃 현대화에 동일한 API를 사용할 수 있습니다.

주제

- 기본 클래스
- 데이터 바인딩 및 액세스
- 논의된 Java 유형의 FQN

기본 클래스

읽기 쉽도록 이 문서에서는 Blu Age API 인터페이스 및 클래스의 Java 약칭을 사용합니다. AWS 자세한 설명은 논의된 Java 유형의 FQN 섹션을 참조하세요.

저수준 메모리 표현

가장 낮은 레벨에서는 메모리(빠르고 무작위 방식으로 액세스할 수 있는 연속적인 바이트 범위)가 `Record` 인터페이스로 표시됩니다. 이 인터페이스는 기본적으로 고정된 크기의 바이트 배열을 추상화한 것입니다. 따라서 기본 바이트에 액세스하거나 수정할 수 있는 setter와 getter를 제공합니다.

구조화된 데이터 표현

COBOL DATA DIVISION에 있는 “01 데이터 항목” 또는 “01 카피북”과 같은 정형 데이터를 표현하기 위해 `RecordEntity` 클래스의 서브클래스가 사용됩니다. 이런 것들은 보통 손으로 직접 작성하지 않고, AWS 블루 에이지 현대화 툴을 통해 해당하는 레거시 구조에서 생성됩니다. 기본 구조와 API에 대해

알아두면 여전히 유용하므로 현대화된 프로그램의 코드가 이를 어떻게 사용하는지 이해할 수 있습니다. COBOL의 경우 해당 코드는 프로시저 부서에서 생성된 Java입니다.

생성된 코드는 RecordEntity 서브클래스와 함께 각 "01 데이터 항목"을 나타내며, 이를 구성하는 각 기본 필드 또는 집계는 트리로 구성된 전용 Java 필드로 표시됩니다(루트 항목을 제외한 각 항목에는 상위 항목이 있음).

설명을 위해 다음은 COBOL 데이터 항목의 예시와 이를 현대화하는 해당 AWS Blu Age 생성 코드입니다.

```
01 TST2.
 02 FILLER PIC X(4).
 02 F1     PIC 9(2) VALUE 42.
 02 FILLER PIC X.
 02        PIC 9(3) VALUE 123.
 02 F2     PIC X VALUE 'A'.
```

```
public class Tst2 extends RecordEntity {

    private final Group root = new Group(getData()).named("TST2");
    private final Filler filler = new Filler(root,new AlphanumericType(4));
    private final Elementary f1 = new Elementary(root,new ZonedType(2, 0, false),new
 BigDecimal("42")).named("F1");
    private final Filler filler1 = new Filler(root,new AlphanumericType(1));
    private final Filler filler2 = new Filler(root,new ZonedType(3, 0, false),new
 BigDecimal("123"));
    private final Elementary f2 = new Elementary(root,new
 AlphanumericType(1),"A").named("F2");


    /**
     * Instantiate a new Tst2 with a default record.
     * @param configuration the configuration
     */
    public Tst2(Configuration configuration) {
        super(configuration);
        setupRoot(root);
    }
    /**
     * Instantiate a new Tst2 bound to the provided record.
     * @param configuration the configuration
     * @param record the existing record to bind
```

```
     */
    public Tst2(Configuration configuration, RecordAdaptable record) {
        super(configuration);
        setupRoot(root, record);
    }

    /**
     * Gets the reference for attribute f1.
     * @return the f1 attribute reference
     */
    public ElementaryRangeReference getF1Reference() {
        return f1.getReference();
    }

    /* *
     * Getter for f1 attribute.
     * @return f1 attribute
     */
    public int getF1() {
        return f1.getValue();
    }

    /**
     * Setter for f1 attribute.
     * @param f1 the new value of f1
     */
    public void setF1(int f1) {
        this.f1.setValue(f1);
    }
    /**
     * Gets the reference for attribute f2.
     * @return the f2 attribute reference
     */
    public ElementaryRangeReference getF2Reference() {
        return f2.getReference();
    }

    /**
     * Getter for f2 attribute.
     * @return f2 attribute
     */
    public String getF2() {
        return f2.getValue();
    }
```

```
    /**
     * Setter for f2 attribute.
     * @param f2 the new value of f2
     */
    public void setF2(String f2) {
        this.f2.setValue(f2);
    }
}
```

기본 필드

Elementary 클래스의 필드(또는 이름이 지정되지 않은 경우 Filler)는 레거시 데이터 구조의 "리프"를 나타냅니다. 이 필드는 연속된 기본 바이트 범위("범위")와 연결되며, 일반적으로 해당 바이트를 해석하고 수정하는 방법을 나타내는 형식(파라미터화될 수도 있음)을 사용합니다(값을 각각 바이트 배열에서 디코딩하거나 바이트 배열로 값을 "디코딩"하고 "인코딩"함).

모든 기본 유형은 RangeType의 서브클래스입니다. 일반적인 유형은 다음과 같습니다.

COBOL 타입	Data Simplifier 유형
PIC X(n)	AlphanumericType
PIC 9(n)	ZonedType
PIC 9(n) COMP-3	PackedType
PIC 9(n) COMP-5	BinaryType

집계 필드

집계 필드는 콘텐츠(기타 집계 또는 기본 필드)의 메모리 레이아웃을 구성합니다. 필드 자체에는 기본 유형이 없습니다.

Group 필드는 메모리의 연속 필드를 나타냅니다. 포함된 각 필드는 메모리에 동일한 순서로 배치됩니다. 첫 번째 필드는 메모리의 그룹 필드 위치를 기준으로 0 오프셋되고 두 번째 필드는 0 + (size in bytes of first field) 오프셋으로 배치됩니다. 동일한 포함 필드 아래에 있는 COBOL 필드의 시퀀스를 나타내는 데 사용됩니다.

Union 필드는 동일한 메모리에 액세스하는 여러 필드를 나타냅니다. 포함된 각 필드는 메모리의 통합 필드 위치를 기준으로 오프셋 0에 배치됩니다. 예를 들어 COBOL “REDEFINES” 구문을 나타내는 데 사용됩니다(첫 번째 Union 하위 항목은 재정의된 데이터 항목, 두 번째 하위 항목은 첫 번째 재정의 등).

배열 필드(Repetition의 서브클래스) 는 하위 필드(집계 자체 또는 기본 항목)의 메모리 내 반복을 나타냅니다. 이러한 하위 레이아웃을 지정된 개수만큼 메모리에 배치하고 각 레이아웃은 오프셋 `index * (size in bytes of child)`으로 배치합니다. 이들은 COBOL “OCCERS” 구문을 나타내는 데 사용됩니다.

프리미티브

일부 현대화 사례에서는 “프리미티브”를 사용하여 독립적인 “루트” 데이터 항목을 표시할 수도 있습니다. 이러한 것들은 용도가 매우 RecordEntity 비슷하지만 그것에서 나온 것도 아니며 생성된 코드를 기반으로 하지도 않습니다. 대신 AWS Blu Age 런타임에서 인터페이스의 서브클래스로 직접 제공됩니다. Primitive 제공되는 클래스의 예로는 Alphanumeric 또는 ZonedDecimal 등이 있습니다.

데이터 바인딩 및 액세스

구조화된 데이터와 기본 데이터를 연결하는 방법은 여러 가지가 있습니다.

이 목적을 위한 중요한 인터페이스는 RecordAdaptable 기본 데이터에 대한 '쓰기 가능한 뷰'를 제공하는 Record를 가져오는 데 사용하는 RecordAdaptable입니다. 아래에서 볼 수 있듯이 여러 클래스가 RecordAdaptable를 구현합니다. 반대로 AWS Blu Age API와 저수준 메모리를 조작하는 코드(예: 프로그램 인수, 파일 I/O 레코드, CICS 쉼표, 할당된 메모리 등) 는 a를 해당 메모리의 핸들러로 사용하는 경우가 많습니다. RecordAdaptable

COBOL 현대화의 경우 대부분의 데이터 항목은 해당 프로그램 실행의 수명 기간 동안 수정되는 메모리와 연관됩니다. 이를 위해 RecordEntity 서브클래스는 생성된 상위 객체(프로그램 컨텍스트)에서 한 번 인스턴스화되고 RecordEntity 바이트 크기에 따라 Record 기본 객체를 인스턴스화합니다.

LINKAGE 요소를 프로그램 인수와 연결하거나 SET ADDRESS OF 구문을 현대화하는 것과 같은 다른 COBOL의 경우에는 RecordEntity 인스턴스를 제공된 RecordAdaptable와 연결해야 합니다. 이를 위해 다음과 같은 두 가지 메커니즘이 있습니다.

- RecordEntity 인스턴스가 이미 존재하는 경우 RecordEntity.bind(RecordAdaptable) 메서드(에서 상속됨Bindable)를 사용하여 이 인스턴스가 이 RecordAdaptable를 “가리키도록” 만들 수 있습니다. 그러면 RecordEntity에서 호출되는 모든 getter 또는 setter는 기본 RecordAdaptable 바이트에 의해 백업(읽기 또는 쓰기 바이트) 됩니다.
- RecordEntity를 인스턴스화하려는 경우 RecordAdaptable를 받아들이는 생성된 생성자를 사용할 수 있습니다.

반대로, 현재 구조화된 데이터에 바인딩된 Record도 액세스할 수 있습니다. 이를 위해 RecordEntity는 RecordAdaptable를 구현하므로 해당 인스턴스에서 getRecord()를 직접적으로 호출할 수 있습니다.

마지막으로, 많은 COBOL 또는 CICS 동사는 읽기 또는 쓰기 목적으로 단일 필드에 액세스할 수 있어야 합니다. RangeReference 클래스는 이러한 액세스를 나타내는 데 사용됩니다. RecordEntity 생성된 getXXXReference() 메서드(XXX액세스 필드)에서 해당 인스턴스를 가져와 런타임 메서드에 전달할 수 있습니다. RangeReference는 일반적으로 전체 RecordEntity 또는 Group에 액세스하는 데 사용되며, 서브클래스 ElementaryRangeReference는 Elementary 필드에 대한 액세스를나타냅니다.

참고로, 위의 관찰 내용은 서브클래스에 적용되는데, Primitive 서브클래스는 생성된 코드 대신 Blu Age 런타임에서 제공하는 것과 유사한 동작을 구현하기 위해 노력하기 때문입니다. RecordEntity AWS 이를 위해 Primitive의 모든 서브클래스는 RecordAdaptable, ElementaryRangeReference 및 Bindable 인터페이스를 구현하여 RecordEntity 서브 클래스 및 기본 필드 기본 필드 대신 사용할 수 있도록 만듭니다.

논의된 Java 유형의 FQN

다음 표에는 이 섹션에서 설명하는 Java 유형의 정규화된 이름이 나와 있습니다.

짧은 이름	정규화된 인덱스 이름
Alphanumeric	com.netfective.bluage.gapwalk.datasimplifier.elementary.Alphanumeric
AlphanumericType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.AlphanumericType
BinaryType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.BinaryType
Bindable	com.netfective.bluage.gapwalk.datasimplifier.data.Bindable

짧은 이름	정규화된 인덱스 이름
Elementary	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Elementary
ElementaryRangeReference	com.netfective.bluage.gapwalk.datasimplifier.entity.ElementaryRangeReference
Filler	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Filler
Group	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Group
PackedType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.PackedType
Primitive	com.netfective.bluage.gapwalk.datasimplifier.elementary.Primitive
RangeReference	com.netfective.bluage.gapwalk.datasimplifier.entity.RangeReference
RangeType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.RangeType
Record	com.netfective.bluage.gapwalk.datasimplifier.data.Record

짧은 이름	정규화된 인덱스 이름
RecordAdaptable	com.netfective.bluage.gapwalk.datasimplifier.data.RecordAdaptable
RecordEntity	com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity
Repetition	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Repetition
Union	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Union
ZonedDecimal	com.netfective.bluage.gapwalk.datasimplifier.elementary.ZonedDecimal
ZonedType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.ZonedType

AWS 블루에이지 런타임 구성 및 구성 파일

Velocity 프레임워크와 클라이언트 코드는 Spring Boot 프레임워크를 사용하는 웹 애플리케이션입니다. Spring 기능을 활용하여 가능한 여러 위치 및 우선 순위 규칙이 포함된 구성을 제공합니다. groovy 스크립트, sql 등과 같은 다른 많은 파일을 제공하는 데에도 유사한 우선 순위 규칙이 있습니다.

Velocity 프레임워크에는 필요한 경우 옵트인할 수 있는 추가 선택적 웹 애플리케이션도 포함되어 있습니다.

주제

- 애플리케이션 구성 기본

- 애플리케이션 우선 순위
- 데이터베이스용 JNDI
- 시크릿 사용 AWS
- 기타 파일(groovy, SQL 등)
- 추가 웹 애플리케이션
- 속성 활성화
- Gapwalk 애플리케이션에 대한 인증 구성

애플리케이션 구성 기본

애플리케이션 구성을 처리하는 기본 방법은 애플리케이션 서버 폴더에 제공되는 전용 YAML 파일을 사용하는 것입니다. `config` 다음과 같은 두 가지 기본 YAML 구성 파일이 있습니다.

- `application-main.yaml`
- `application-`*`profile`*`.yaml`(애플리케이션 생성 중에 *`profile`* 값이 설정되는 경우).

첫 번째 파일은 프레임워크 즉 `Gapwalk-application.war`를 구성하는 것이고 두 번째 파일은 클라이언트 애플리케이션을 위한 추가 옵션을 위한 것입니다. 이는 스프링 프로파일을 사용할 때 작동합니다. Gapwalk 애플리케이션은 `main` 프로파일을 사용하고 클라이언트 애플리케이션은 *`profile`* 프로파일을 사용합니다.

다음은 대표적인 주요 YAML 파일을 보여줍니다.

```
##########################################
##### JICS datasource configuration #####
##########################################
datasource:
  jicsDs:
    driver-class-name : org.postgresql.Driver
    url: jdbc:postgresql://localhost/jics
    username: jics
    password: jics
    type : org.postgresql.ds.PGSimpleDataSource

#####################################################
##### Embedded Bluesam datasource configuration #####
#####################################################
  bluesamDs :
    driver-class-name : org.postgresql.Driver
    url : jdbc:postgresql://localhost/bluesam
    username : bluesam
    password : bluesam
    type : org.postgresql.ds.PGSimpleDataSource

##########################################
##### Embedded Bluesam configuration #####
##########################################
bluesam :
  remote : false
  cache : ehcache
  persistence : pgsql #pgsql, mssql, xodus...
  ehcache:
    resource-pool:
      size: 4GB
  write-behind:
```

다음 예제에서는 일반적인 클라이언트 YAML 파일을 보여줍니다.

```
# Logback context logger integration.
logging.config : classpath:logback-[redacted].xml
# Limits Spring logger output.
logging.level.org.springframework.beans.factory.support.DefaultListableBeanFactory : WARN
logging.level.org.springframework.statemachine : WARN
# If the datasource support mode is not static-xa, spring JTA transactions autoconfiguration must me disabled
spring.jta.enabled : false

spring:
  aws:
    client:
      datasources:
        names: primary
        primary:
          secret: arn:aws:secretsmanager:[redacted]

spring.jta.atomikos.datasource.primary.unique-resource-name: primary
spring.jta.atomikos.datasource.primary.xa-data-source-class-name: org.postgresql.xa.PGXADataSource
spring.jta.atomikos.datasource.primary.maxPoolSize: 20
spring.jta.atomikos.datasource.primary.autoCommit: false
```

YAML 파일의 내용에 대해 자세히 알아보려면 속성 활성화 섹션을 참조하세요.

애플리케이션 우선 순위

이러한 구성 파일에는 Spring 우선 순위 규칙이 적용됩니다. 특히:

- `application-main`YAML 파일은 Gapwalk 메인 워 파일에 기본값으로 나타나며 `config` 폴더에 있는 파일이 이를 대체합니다.
- 클라이언트 애플리케이션 구성에서도 동일한 작업을 수행해야 합니다
- 서버 시작 시 명령줄에서 추가 파라미터를 전달할 수 있습니다. YAML 파일을 재정의합니다.

자세한 내용은 공식 Spring Boot 설명서를 참조하세요.

데이터베이스용 JNDI

데이터베이스 구성은 Tomcat의 context.xml 파일에 있는 JNDI와 함께 제공될 수 있습니다. 이러한 구성은 YAML 구성을 재정의합니다. 하지만 이 방법을 사용하면 보안 인증을 암호 매니저에 래핑할 수 없다는 점에 유의하세요(아래 참조).

다음 예제는 JICS 및 데이터베이스의 샘플 구성을 보여줍니다. BluSam

```
<Resource auth="Container" driverClassName="org.postgresql.Driver" initialSize="0"
 maxIdle="5"
   maxOpenPreparedStatements="-1" maxTotal="10" maxWaitMillis="-1" name="jdbc/jics"
   poolPreparedStatements="true" testOnBorrow="false" type="javax.sql.DataSource"
   url="jdbc:postgresql://XXXX.rds.amazonaws.com:5432/XXXX" username="XXXX"
 password="XXXX" />
```

jdbc/jics

JICS `jdbc/jics` 데이터베이스용이고 `jdbc/bluesam` ('e'에 주의) blusam 데이터베이스용입니다.

url="jdbc:postgresql://XXXX.rds.amazonaws.com:5432/XXXX" username="XXXX" password="XXXX"

데이터베이스 URL, 사용자 이름 및 암호.

시크릿 사용 AWS

자격 증명이 포함된 일부 리소스 구성은 AWS 암호를 사용하여 추가로 보호할 수 있습니다. 중요한 데이터를 시크릿에 저장하고 YAML 구성에서 AWS 시크릿에 대한 참조를 포함시켜 톰캣 시작 시 시크릿 콘텐츠를 즉석에서 선택하도록 하는 것이 핵심이다.

Aurora에 대한 암호

Aurora 데이터베이스 구성(jics, blusam, customer db 등의 경우)은 내장된 데이터베이스 암호를 사용하여 해당 데이터베이스에서 모든 관련 필드를 자동으로 채웁니다.

> **Note**
>
> dbname 키는 선택 사항이며, 데이터베이스 구성에 따라 비밀에 포함되거나 포함되지 않을 수 있습니다. 여기에 수동으로 추가하거나 YAML 파일에 이름을 제공하여 추가할 수 있습니다.

기타 암호

다른 비밀은 단일 비밀번호를 가진 리소스(특히 password-protected redis 캐시)에 대한 것입니다. 이 경우 단일 `password` 키로 다른 유형의 암호를 사용해야 합니다.

비밀에 대한 YAML 참조

는 다양한 리소스의 비밀 ARN을 `application-main.yaml` 참조할 수 있습니다. 가장 중요한 것은 다음과 같습니다.

- JICS 데이터베이스 자격 증명은 다음과 같습니다. `spring.aws.jics.db.secret`
- JICS TS는 Redis 자격 증명을 다음과 같이 대기열에 넣습니다. `spring.aws.client.jics.queues.ts.redis.secret`
- `spring.aws.client.bluesam.db.secret`가 있는 Blusam 데이터베이스 보안 인증
- `spring.aws.client.bluesam.redis.secret`가 있는 Blusam 캐시 비밀번호
- `spring.aws.client.bluesam.locks.redis.secret`가 있는 Blusam 잠금 캐시 비밀번호

다음 예제에서는 YAML 파일에서 이러한 암호를 선언하는 방법을 보여줍니다.

```
spring:
  aws:
    client:
      bluesam:
        locks:
          redis:
            secret: arn:aws:secretsmanager:XXXX
```

```
        db:
          dbname: bluesam
          secret: arn:aws:secretsmanager:XXXX
        redis:
          secret: arn:aws:secretsmanager:XXXX
      jics:
         queues:
          ts:
           redis:
            secret: arn:aws:secretsmanager:XXXX
    jics:
      db:
        secret: arn:aws:secretsmanager:XXXX
```

dbname: bluesam

이 예시에서는 데이터베이스 이름이 암호에 없고 대신 여기에 입력됩니다.

클라이언트 `application-profile.yaml`는 클라이언트 데이터베이스의 비밀 ARN을 참조할 수 있습니다. 이를 위해서는 아래 예와 같이 데이터 소스를 나열하기 위한 추가 속성이 필요합니다.

```
spring:
  aws:
    client:
      datasources:
        names: primary,host
        primary:
          secret: arn:aws:secretsmanager:XXXX
        host:

          secret: arn:aws:secretsmanager:XXXX
```

이름: primary,host

primary 및 host라는 두 개의 클라이언트 데이터 원본이 있고 각각 데이터베이스와 보안 인증이 있는 예제입니다.

dbname: mydb

이 예제에서 “host” 데이터베이스의 이름은 암호에 없고 대신 여기에 입력되지만, “primary” 데이터베이스의 경우 암호에 있습니다.

XA가 지원하는 비밀 키는 없습니다.

- 엔진 (포스트그레스/오라클/db2/mssql)
- 포트
- dbname
- 현재 스키마
- 사용자 이름
- 암호
- url

`spring.aws.rds.ssl.cert-pathyml` 속성 외에 `sslMode` 암호 키 값 (`disable/allow/prefer/require/verify-ca/verify-full`) `postgres` 만 있으면 SSL로 연결할 수 있습니다.

XA가 지원하는 비밀 키

클라이언트 데이터베이스가 XA를 사용하는 경우 비밀 값을 통해 하위 xa-속성이 지원됩니다.

- host
- 포트
- dbname
- 현재 스키마
- 사용자 이름
- 암호
- url
- SSL 연결 (참/거짓)

그러나 다른 xa 속성 (예: `maxPoolSize` 또는`driverType`) 의 경우 여전히 일반 YAML 키를 제공해야 합니다. `spring.jta.atomikos.datasource.XXXX.unique-resource-name`

비밀 값은 YAML 속성을 재정의합니다.

기타 파일(groovy, SQL 등)

고객 프로젝트에서 사용하는 다른 파일은 스프링 구성용 파일과 유사한 우선 순위 규칙을 사용합니다. 예시:

- Groovy 스크립트는 `scripts` 폴더 또는 하위 폴더에 있는 `.groovy` 파일입니다.
- SQL 스크립트는 `sql` 폴더 또는 하위 폴더에 있는 `.sql` 파일입니다.
- 대몬(daemon) 스크립트는 폴더 또는 `daemons` 하위 폴더에 있는 `.groovy` 파일입니다.
- 쿼리 데이터베이스 매핑 파일은 `sql` 폴더 하위 폴더에 있는 `queries-database.mapping` 파일 이름을 가진 파일입니다.
- Jasper 템플릿은 `templates` 폴더 또는 하위 폴더에 있는 `.jrxml` 파일입니다.
- 데이터 세트 카탈로그는 `catalog` 폴더에 있는 `.json` 파일입니다.
- 링크 파일은 `lnk` 폴더에 있는 `.json` 파일입니다.

이러한 모든 위치는 시스템 속성 또는 클라이언트 yml 속성을 통해 재정의할 수 있습니다.

- Groovy 스크립트의 경우: `configuration.scripts`
- SQL 스크립트의 경우: `configuration.sql`
- 대몬(daemon) 스크립트의 경우: `configuration.daemons`
- 쿼리 데이터베이스 매핑 파일의 경우: `configuration.databaseMapping`
- Jasper 템플릿의 경우: `configuration.templates`
- 데이터 세트 카탈로그의 경우: `configuration.catalog`
- Lnk 파일의 경우: `configuration.lnk`

속성을 찾을 수 없는 경우 파일은 위에서 언급한 기본 위치에서 가져옵니다. 조회는 먼저 tomcat 작업 디렉토리를 루트로 사용하고 마지막으로 응용 프로그램 war 파일에서 수행됩니다.

추가 웹 애플리케이션

Velocity 프레임워크의 `webapps-extra` 폴더에 추가 웹 애플리케이션이 포함되어 있습니다. Tomcat 서버는 기본적으로 이러한 애플리케이션을 제공하지 않습니다.

이러한 웹 애플리케이션에 대한 옵트인은 현대화 프로젝트에 따라 다르며 원하는 war 파일을 `webapps-extra` 폴더에서 `webapps` 폴더로 이동하면 됩니다. 그 이후에는 다음 시작 시 tomcat 서버가 전쟁을 처리하게 됩니다.

파일 및 위에서 설명한 것처럼 일부 프로젝트별 추가 구성을 각 추가 war의 YAML 구성 파일에 추가할 수도 있습니다. `application-main.yml` 추가 war 파일은 다음과 같습니다.

- `gapwalk-utility-pgm.war`: ZOS 유틸리티 프로그램에 대한 지원을 포함하며 해당 구성으로 `application-utility-pgm.yaml`를 사용합니다.
- `gapwalk-cl-command.war`: AS/400 유틸리티 프로그램에 대한 지원을 포함하며 해당 구성으로 `application-cl-command.yaml`를 사용합니다.
- `gapwalk-hierarchical-support.war`: IMS/MFS 트랜잭션 지원을 포함하고 해당 구성으로 `application-jhdb.yaml`를 사용합니다

속성 활성화

Spring Boot에서 `application-main.yml` 애플리케이션은 수신 포트, 데이터베이스 연결 등과 같은 다양한 종류의 속성을 정의하는 구성 파일입니다.

주제

YML 표기법

다음 설명서에서는 와 같은 속성을 YAML 형식으로 다음과 같이 작성합니다.
`parent.child1.child2=true`

```
parent:
 child1:
   child2: true
```

퀵 스타트 / 사용 사례

다음 사용 사례는 해당 키 및 값의 예를 보여줍니다.

- 기본 애플리케이션-main.yml

```
----
#### DEFAULT APPLICATION-MAIN.YML FILE      #####
#### SHOWING USEFUL CONFIGURATION ELEMENTS #####
#### SHOULD BE OVERRIDDEN AND EXTERNALIZED   #####
```

```
#################################
##### Logging configuration #####
#################################
logging:
  config: classpath:logback-main.xml
  level.org.springframework.beans.factory.support.DefaultListableBeanFactory : WARN

#################################
##### Spring configuration #####
#################################
spring:
  quartz:
    auto-startup: false
    scheduler-name: Default
    properties:
      org.quartz.threadPool.threadCount: 1
  jta:
    enabled: false
    atomikos.properties.maxTimeout : 600000
    atomikos.properties.default-jta-timeout : 100000
  jpa:
# DISABLE OpenEntityManagerInViewInterceptor
    open-in-view: false
    # Fix Postgres JPA Error:
    # Method org.postgresql.jdbc.PgConnection.createClob() is not yet implemented.
    properties.hibernate.temp.use_jdbc_metadata_defaults : false
#####################################
##### Jics tables configuration #####
#####################################

    # The dialect should match the jics datasource choice
    database-platform : org.hibernate.dialect.PostgreSQLDialect #
 org.hibernate.dialect.PostgreSQLDialect, org.hibernate.dialect.SQLServerDialect

    # those properties can be used to create and initialize jics tables
 automatically.
#    properties:
#      hibernate:
#      globally_quoted_identifiers: true
#        hbm2ddl:
#           import_files_sql_extractor :
 org.hibernate.tool.hbm2ddl.MultipleLinesSqlCommandExtractor
#           import_files : file:./setup/initJics.sql
```

```
#           auto : create

#########################
###### Level 2 cache #####
#########################
#        cache:
#          use_second_level_cache: true
#          use_query_cache: true
#          region:
#            factory_class: org.hibernate.cache.ehcache.EhCacheRegionFactory
#      javax:
#        persistence:
#          sharedCache:
#            mode: ENABLE_SELECTIVE
#########################
###### Redis settings #####
#########################
  session:
    store-type: none #redis

########################################
##### JICS datasource configuration #####
########################################
datasource:
  jicsDs:
    driver-class-name : org.postgresql.Driver # org.postgresql.Driver,
 com.microsoft.sqlserver.jdbc.SQLServerDriver
    url: jdbc:postgresql://localhost/jics # jdbc:postgresql://localhost:5433/jics,
 jdbc:sqlserver://localhost\SQLEXPRESS:1434;databasename=jics;
    username: jics
    password: jics
    type : org.postgresql.ds.PGSimpleDataSource #
 org.postgresql.ds.PGSimpleDataSource,
 com.microsoft.sqlserver.jdbc.SQLServerDataSource

#####################################################
##### Embedded Bluesam datasource configuration #####
#####################################################
  bluesamDs :
    driver-class-name : org.postgresql.Driver # org.postgresql.Driver,
 com.microsoft.sqlserver.jdbc.SQLServerDriver
    url : jdbc:postgresql://localhost/bluesam # jdbc:postgresql://localhost:5433/
jics, jdbc:sqlserver://localhost\SQLEXPRESS:1434;databasename=jics;
    username : bluesam
```

```
    password : bluesam
    type : org.postgresql.ds.PGSimpleDataSource #
 org.postgresql.ds.PGSimpleDataSource,
 com.microsoft.sqlserver.jdbc.SQLServerDataSource

########################################
##### Embedded Bluesam configuration #####
########################################
bluesam :
  remote : false
  cache : ehcache
  persistence : pgsql #pgsql, mssql, xodus...
  ehcache:
    resource-pool:
      size: 4GB
  write-behind:
    enabled: true
  pgsql :
    dataSource : bluesamDs

#########################
##### Jics settings #####
#########################
rabbitmq.host: localhost
jics:
  cache: false #redis
  resource-definitions.store-type: jpa # default value: jpa, other possible value:
 redis
  redis.hostname: 127.0.0.1 # Redis server host.
  redis.password: redis # Login password of the redis server.
  redis.port: 6379 # Redis server port.
  redis.username: # Redis username
  redis.mode: standalone # Redis mode. Possible values: standalone, cluster
jics.disableSyncpoint : false
#jics.initList:
#jics.parameters.datform: DDMMYY
#jics.parameters.applid: VELOCITY
#jics.parameters.sysid: CICS
#jics.parameters.eibtrmid: TERM
#jics.parameters.userid: MYUSERID
#jics.parameters.username: MYUSERNAME
#jics.parameters.opid: XXX
#jics.parameters.cwa.length: 0
#jics.parameters.netname: MYNETNAME
```

```
#jics.parameters.jobname: MJOBNAME
#jics.parameters.sysname: SYSNAME

###############################################
##### Jics RunUnitLauncher pool settings #####
###############################################
#jics.runUnitLauncherPool.enable: false
#jics.runUnitLauncherPool.size: 20
#jics.runUnitLauncherPool.validationInterval: 1000

#########################
##### Jhdb settings #####
#########################
#jhdb.lterm: LTERMVAL
#jhdb.identificationCardData: SomeIDData

###################################
##### DateHelper configuration ####
###################################
#forcedDate: "2013-08-26T12:59:58+01:57"

#############################
##### Sort configuration ####
#############################
#externalSort.threshold: 256MB

###################################
##### Server timeout (10 min)  ####
###################################
spring.mvc.async.request-timeout: 600000

#############################
##### DATABASE STATISTICS  ####
#############################
databaseStatistics : false

#####################
##### CALLS GRAPH ####
#####################
callGraph : false

#############################
##### SQL SHIFT CODE POINT ####
#############################
```

```
# Code point 384 match unicode character \u0180
sqlCodePointShift : 384

################################
##### LOCK TIMEOUT RECORD  ####
################################
# Blu4IV record lock timeout
lockTimeout : 100

################################
##### REPORTS OUTPUT PATH ####
################################
reportOutputPath: reports

################################
##### TASK EXECUTOR       ####
################################
taskExecutor:
  corePoolSize: 5
  maxPoolSize: 10
  queueCapacity: 50
  allowCoreThreadTimeOut: false

################################
##### PROGRAM NOT FOUND   ####
################################
stopExecutionWhenProgNotFound: false

#########################################################
#####  DISP DEFAULT VALUE (to be removed one day) ####
#########################################################
defaultKeepExistingFiles: true

#####################################
#####  JOBQUEUE CONFIGURATION    ####
#####################################
jobqueue:
  api.enabled: false
  impl: none # possible values: quartz, none
  schedulers:  # list of schedulers
    -
      name: queue1
      threadCount: 5
    -
```

```
      name: queue2
      threadCount: 5

################################################################################
##### QUERY BUILDING                                                        ##
# useConcatCondition : false by default
# if true, in the query, the where condition is build with key concatenation  ##
################################################################################
# query.useConcatCondition: true
----
```

- LISTCAT 명령과 함께 가변 길이 파일 사용

```
[**/*.*]
encoding=IBM930
reencoding=false

[global]
listcat.variablelengthpreprocessor.enabled=true
listcat.variablelengthpreprocessor.type=rdw
# use "rdw" if your .listcat file contains a set of records (RDW)
# use "bdw" if your .listcat file contains a set of blocks (bdw)
```

- LOAD/UNLOAD 유틸리티에 널 바이트 표시기 값 제공

```
# Unload properties
# For date/time: if use database configuration is enabled, formats are ignored
# For nbi; use hexadecimal syntax to specify the byte value
# - When the value is null in database : the value dumped to the file is filled by
 low value characters and the NBI is
# equal to the byte 6F (the ? character)
# - When the value is not null in database and the column is nullable: the NBI is
 equal to the byte 00 (low value) and NOT
# equal to the byte 40 (space)
unload:
  sqlCodePointShift: 0
  nbi:
    whenNull: "6F"
    whenNotNull: "00"
```

```
  useDatabaseConfiguration: false
  format:
    date: MM/dd/yyyy
    time: HH.mm.ss
    timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
```

기본 애플리케이션에 사용할 수 있는 속성

이 표는 키/값 파라미터를 전체적으로 보여줍니다.

키	유형	기본값	설명
`logging.config`	경로	classpath:logback-main.xml	로그백 구성 파일에 대한 참조를 위한 표준 키입니다. 다른 표준 로깅 키도 사용할 수 있습니다.
`spring.jta.enabled`	부울	false	표준 키. 데이터 소스 지원 모드가 static-xa가 아닌 경우 스프링 JTA 트랜잭션 자동 구성을 비활성화해야 합니다.
`datasource.jicsDs + -driver-class-name + -url + -username + -password + -type`	하위 키가 있는 표준 스프링 데이터소스		Jics 데이터베이스에 대한 연결 정보를 포함합니다. 또는 xref:../configuration/configuration.adoc[Configuration]에 설명된 대로 AWS 암호를 사용하는 것이 좋습니다.
`datasource.bluesamDs + -driver-class-name + -url`	하위 키가 있는 표준 스프링 데이터소스		Blusam 데이터베이스에 대한 연결 정보를 포함합니다. 또는 xref:../configurat

키	유형	기본값	설명
+ -username + -password + -type			ion/configuration.adoc[Configuration]에 설명된 대로 AWS 암호를 사용하는 것이 좋습니다.
bluesam.disabled	부울	false	blusam을 완전히 비활성화할지 여부.
bluesam.cache	문자열		설정하지 않으면 blusam 캐시가 사용되지 않습니다. 가능한 값(캐시 구현) 은 eccache 및 redis입니다.
forcedDate	문자열		제공된 날짜가 있는 경우 날짜를 지정된 날짜로 강제 적용합니다.
frozenDate	부울	true	날짜를 고정할지 여부를 지정합니다. forcedDate 도 설정된 경우에만 적용됩니다.
externalSort.threshold	데이터 크기(예: 12MB)		정렬 임계값: 외부(병합) 정렬로 전환해야 하는 시점.
jics.parameters.datform	문자열	MMDDYY	날짜 양식.

키	유형	기본값	설명
`jics.initList``	문자열		초기화 jics 목록(쉼표로 구분됨). 존재하는 경우 CICS 목록 중에서 tomcat 시작 시 활성화할 목록의 이름을 쉼표로 구분하여 정의합니다. 예제 값: `$UUU,DFH$IVPL,PEZ1` . 그러면 해당 목록에 포함된 그룹과 해당 기본 리소스 정의에 계단식으로 전달되어 런타임에서 이를 볼 수 있습니다. 기본적으로 비어 있습니다.
`jics.parameters.applid`	문자열	VELOCITY	JICS에서 애플리케이션을 식별하기 위해 적용하는 애플리케이션입니다(최소 4자, 최대 길이 없음).
`jics.parameters.sysid`	문자열	CICS	시스템 ID(SYSID).
`jics.parameters.eibtrmid`	문자열	TERM	터미널 식별자(최대 4자, 최소 1자).

키	유형	기본값	설명
jics.parameters.userid	문자열		사용자 ID(최대 8자, 최소 문자 없음). 값이 제공되지 않은 경우(기본적으로 비어 있음) HTTP 세션 ID가 사용자 ID로 사용됩니다.
jics.parameters.username	문자열	MYUSERNAME	사용자 이름(최대 10자, 최소 1자).
jics.parameters.netname	문자열	MYNETNAME	네트워크 이름(최대 8자, 최소 1자).
jics.parameters.opid	문자열	XXX	3자리 운영자 식별.
jics.parameters.jobname`	문자열	MJOBNAME	직무 이름.
jics.parameters.sysname	문자열	SYSNAME	AS400 시스템 이름(시스템 이름).
jics.parameters.cwa.length	number	0	공용 작업 영역(cwa) 길이.
jics.parameters.charset	문자열	CP037	JICS는 전 세계적으로 사용되는 문자 집합입니다.

키	유형	기본값	설명
jics.parameters.tsqimpl	문자열	bluesam	JICS 임시 스토리지 큐(TSQ) 구현(허용되는 값은 bluesam/memory/redis)
jics.queues.ts.redis.hostname	문자열	127.0.0.1	jics cache redis 서버의 호스트 이름입니다.
jics.queues.ts.redis.port	number	6379	jics cache redis 서버의 포트입니다.
jics.queues.ts.redis.password	문자열	redis	jics cache redis 서버의 암호입니다.
jics.queues.ts.redis.username	문자열		jics cache redis 서버의 사용자 이름입니다. 기본값은 공백입니다(사용자 이름 없음).
jics.queues.ts.redis.mode	문자열	독립형	jics cache 모드. 가능한 값은 standalone 또는 cluster입니다. 기본값은 standalone 입니다.
lockTimeout	number	500	잠금 제한 시간(밀리초).

키	유형	기본값	설명
sqlCodePointShift	number		선택 사항입니다. SQL 코드 포인트 시프트. 레거시 rdbms 데이터를 최신 rdbms로 마이그레이션할 때 발생할 수 있는 제어 문자의 코드포인트를 이동합니다. 예를 들어 \u0180 유니코드 문자와 일치하도록 384를 지정할 수 있습니다.
sqlIntegerOverflowAllowed	부울	false	SQL 정수 오버플로를 허용할지 여부, 즉 호스트 변수에 더 큰 값을 배치할 수 있는지 여부를 지정합니다.
database.cursor.overflow.allowed	부울	true	커서 오버플로를 허용할지 여부를 지정합니다. 커서의 위치에 상관없이 커서에서 다음 직접적 호출을 수행하도록 true로 설정합니다. 커서에서 다음 직접적 호출을 수행하기 전에 커서가 마지막 위치에 있는지 확인하려면 false로 설정합니다. 커서를 스크롤할 수 있는 경우(민감 또는 무감각)에만 사용할 수 있습니다.

키	유형	기본값	설명
reportOutputPath	문자열	/reports	보고서 출력 경로.
spring.session.store-type	문자열	없음	고가용성 환경을 위한 세션 캐시. 가능한 값은 none 또는 redis입니다. 기본값은 none입니다.
stopExecutionWhenProgNotFound	부울	true	프로그램을 찾을 수 없는 경우 실행을 중지할지 여부를 지정합니다. true로 설정하면 프로그램을 찾을 수 없는 경우 실행을 중단합니다.
forceHR	부울	false	콘솔 또는 파일 출력에서 사람이 읽을 수 있는 SYSPRINT를 사용할지 여부를 지정합니다.
rollbackOnRTE	부울	false	런타임 예외 시 암시적 실행 단위 트랜잭션을 롤백할지 여부를 지정합니다.
sctThreadLimit	long	5	스크립트 트리거에 대한 스레드 한도입니다.

키	유형	기본값	설명
dataSimplifier.onInvalidNumericData	문자열	reject	유효하지 않은 숫자 데이터를 디코딩할 때 대응하는 방법. 허용되는 값은 reject /tolerates paces /tolerates paceslowvalues /toleratemost 입니다. 기본값은 reject입니다.
filesDirectory	문자열		일괄 입력/출력 파일을 위한 디렉토리입니다.
ims.messages.extendedSize	부울	false	ims 메시지에 extendedSize를 설정할지 여부를 지정합니다.
defaultKeepExistingFiles	부울	false	데이터 세트 기본 이전 값을 설정할지 여부를 지정합니다.
jics.db.ddlScriptLocation	문자열		Jics dll 스크립트 위치. .sql 스크립트를 사용하여 jics 데이터베이스 스키마를 시작할 수 있습니다. 기본적으로 비어 있습니다. 예: ./jics/sql/jics.sql .

키	유형	기본값	설명
jics.db.schemaTestQueryLocation	문자열		jics 스키마의 개체 수(있는 경우)를 반환하는 고유한 쿼리를 포함해야 하는 sql 파일의 위치입니다.
jics.db.dataScriptLocation	문자열		메인프레임에서 CSD 가져오기를 구문 분석하여 Analyzer에서 준비한 initJics.sql 스크립트의 위치.
jics.db.dataTestQueryLocation	문자열		개체 수를 반환할 것으로 예상되는 단일 SQL 쿼리를 포함하는 SQL 스크립트의 위치(예: jics 프로그램 테이블의 레코드 수 계산). 개수가 0이면 jics.db.dataScriptLocation 스크립트를 사용하여 데이터베이스를 로드하고 그렇지 않으면 데이터베이스 로드를 건너뛰게 됩니다.
jics.data.dataJsonInitLocation	문자열		
jics.xa.agent.timeout	number		

키	유형	기본값	설명
query.use ConcatCon dition	부울	false	키 조건을 키 연결로 빌드할지 여부를 지정합니다.
system.qdecfmt	문자열		
dispositi on.checke xistence	부울	false	DISP SHR 또는 OLD를 사용하는 데이터세트의 파일 존재 여부에 대한 검사를 해제할지 여부를 지정합니다.
useContro lMVariable	부울	false	변수 대체에 control-M 사양을 사용할지 여부를 지정합니다.
card.encoding	문자열	CP1145	카드 인코딩: useContro lMVariable`. 와 함께 사용
mapTransf o.prefixes	문자열	&,@,%%	controlM 변수를 변환할 때 사용할 접두사 목록. 각각 쉼표로 구분됩니다.
checkinpu tfilesize	부울	false	파일 크기가 레코드 크기의 배수인 경우 검사를 해제할지 여부를 지정합니다.
stepFailW henAbend	부울	true	단계가 실패하거나 실행을 완료한 경우 제한을 발생시킬지 여부를 지정합니다.

키	유형	기본값	설명
bluesam.fileLoading.commitInterval	number	100000	bluesam 커밋 간격.
uppercaseUserInput	부울	true	사용자 입력이 대문자여야 하는지 여부를 지정합니다.
jhdb.lterm	문자열		IMS 에뮬레이션의 경우 공통 논리적 터미널 ID를 강제로 적용할 수 있습니다. 설정하지 않으면 sessionId가 사용됩니다.
jhdb.identificationCardData	문자열		일부 “운영자 ID 카드 데이터”를 CARD 파라미터로 지정된 MID 필드에 하드코딩하는 데 사용됩니다. 기본적으로 비어 있으며 입력 제한이 없습니다.
encoding	문자열	ASCII	프로젝트에서 사용되는 인코딩(groovy 파일에서는 사용되지 않음). 올바른 CP1047,IBM930,ASCII,UTF-8 인코딩이 필요합니다.

키	유형	기본값	설명
cl.configuration.context.encoding	문자열	CP297	CL 파일의 인코딩. 올바른 CP1047,IBM930,ASCII,UTF-8 인코딩이 필요합니다. 기본 값은 CP297입니다
cl.zonedMode	문자열	EBCDIC_STRICT	제어 언어(CL) 명령을 인코딩 또는 디코딩하기 위한 모드입니다. 허용되는 값은 EBCDIC_STRICT /EBCDIC_MODIFIED /AS400입니다.
ims.programs	문자열		사용할 IMS 프로그램 목록. 각 파라미터는 세미콜론(;)으로 구분하고 각 트랜잭션은 쉼표(,)로 구분합니다. 예: PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;
jhdb.configuration.context.encoding	문자열	CP297	JHDB(자바 계층적 데이터베이스) 인코딩. 유효한 인코딩 문자열 CP1047,IBM930,ASCII,UTF-8이 필요합니다.

키	유형	기본값	설명
jhdb.metadata.extrapath	문자열	파일:. /setup/	psbs 및 dbds 폴더의 런타임별 추가 루트 폴더를 지정하는 구성 파라미터입니다.
jhdb.checkpointPersistence	문자열	없음	체크포인트 지속성 모드. 허용되는 값은 none /add /end입니다. 새 체크포인트를 생성하여 레지스트리에 추가할 때 체크포인트를 유지하는 데 add를 사용합니다. 서버 종료 시 체크포인트를 유지하는 데 end를 사용합니다. 다른 값을 입력하면 지속성이 비활성화됩니다. 레지스트리에 새 체크포인트가 추가될 때마다 기존의 모든 체크포인트가 직렬화되고 파일이 지워진다는 점에 유의하세요. 파일의 기존 데이터에 추가되지 않습니다. 따라서 체크포인트 수에 따라 성능에 어느 정도 영향을 미칠 수 있습니다.

키	유형	기본값	설명
jhdb.checkpointPath	문자열	파일:. /setup/	jhdb.checkpointPersistence 이 none가 아닌 경우 이 파라미터를 사용하여 체크포인트 지속성 경로(checkpoint.dat 파일 저장 위치)를 설정할 수 있습니다. 레지스트리에 포함된 모든 체크포인트 데이터는 직렬화되고 제공된 폴더에 있는 파일(checkpoint.dat)에 백업됩니다. 이 백업에는 체크포인트 데이터(scriptId, stepId, 데이터베이스 위치 및 체크포인트 영역) 만 관련된다는 점에 유의하세요.
jhdb.navigation.cachenexts	number	5000	RDBMS의 계층 탐색에 사용되는 캐시 기간(밀리초)입니다.
jhdb.use-db-prefix	부울	true	RDBMS의 계층 탐색에서 데이터베이스 접두사를 사용할지 여부를 지정합니다.
jhdb.query.limitJoinUsage	부울	true	RDBMS 그래프에서 제한 조인 사용량 파라미터를 사용할지 여부를 지정합니다.

키	유형	기본값	설명
`taskExecutor.corePoolSize`	number	5	groovy 스크립트를 통해 터미널에서 트랜잭션을 시작하면 새 스레드가 생성됩니다. 이 파라미터를 사용하여 코어 풀 크기를 설정합니다.
`taskExecutor.maxPoolSize`	number	10	groovy 스크립트를 통해 터미널에서 트랜잭션을 시작하면 새 스레드가 생성됩니다. 이 파라미터를 사용하여 최대 풀 크기(최대 병렬 스레드 수)를 설정합니다.
`taskExecutor.queueCapacity`	number	50	groovy 스크립트를 통해 터미널에서 트랜잭션을 시작하면 새 스레드가 생성됩니다. 이 파라미터를 사용하여 큐 크기를 설정합니다.(= `taskExecutor.maxPoolSize` 도달 시 보류 중인 트랜잭션의 최대 수)

키	유형	기본값	설명
taskExecutor.allowCoreThreadTimeOut	부울	false	JCIS에서 코어 스레드의 타임아웃을 허용할지 여부를 지정합니다. 이렇게 하면 0이 아닌 대기열과 함께 사용해도 동적으로 확장 및 축소할 수 있습니다(대기열이 가득 차면 최대 풀 크기가 커지기 때문입니다).
jics.runUnitLauncherPool.enable	부울	false	JICS에서 실행 유닛 런처 풀을 활성화할지 여부를 지정합니다.
jics.runUnitLauncherPool.size	number	20	실행 유닛 런처 풀 크기(JICS).
jics.runUnitLauncherPool.validationInterval	number	1000	JICS 내 실행 유닛 런처 풀의 유효성 검사 간격(밀리초 단위).
spring.aws.application.credentials	문자열	null	JICS의 자격 증명 프로필 파일에서 AWS 자격 증명을 로드합니다.
jics.queues.sqs.region	문자열	eu-west-1	JICS에서 사용되는 AWS 심플 큐 서비스용 AWS 리전입니다.

키	유형	기본값	설명
mq.queues .sqs.region	문자열	eu-west-3	AWS SQS MQ 서비스를 위한 AWS 지역입니다.
quartz.scheduler.stand-by-if-error	부울	false	작업 스케줄러가 대기 모드인 경우 작업 실행을 트리거할지 여부를 지정합니다. true인 경우 활성화된 경우 작업 실행이 트리거되지 않습니다.
databaseStatistics	부울	false	SQL 빌더가 통계 정보를 수집하고 표시할 수 있도록 허용할지 여부를 지정합니다.
dbDateFormat	문자열	yyyy-MM-dd	db 대상 날짜 형식.
dbTimeFormat	문자열	HH:mm:ss	db 목표 시간 형식.
dbTimestampFormat	문자열	yyyy-MM-dd HH:mm:ss.SSSSSS	db 대상 타임스탬프 형식.
dateTimeFormat	문자열	ISO	에서는 데이터베이스 날짜 타임스탬프 유형을 데이터 단순화 엔티티에 넘기는 방법을 dateTimeFormat 설명합니다. 허용되는 값은 ISO /EUR /EUR /USA /LOCAL입니다
localDateFormat	문자열		현지 날짜 형식 목록. 각 형식을 \로 구분하세요.

키	유형	기본값	설명
localTimeFormat	문자열		현지 시간 형식 목록. 각 형식을 \과 같이 구분합니다
localTimestampFormat	문자열		로컬 타임스탬프 형식 목록. 각 형식을 로 \구분합니다.
pgmDateFormat	문자열	yyyy-MM-dd	날짜 시간 형식.
pgmTimeFormat	문자열	HH.MM.ss	pgm(프로그램) 실행에 사용되는 시간 형식입니다.
pgmTimestampFormat	문자열	YYYY-MM-DD-HH.mm.ss.ssssss	타임스탬프 형식.
cacheMetadata	부울	true	데이터베이스 메타데이터를 캐시할지 여부를 지정합니다.
forceDisableSQLTrimStringType	부울	false	모든 SQL 문자열 파라미터의 트림을 비활성화할지 여부를 지정합니다.
fetchSize	number		커서의 fetchSize 값입니다. 로드/언로드 유틸리티로 청크를 사용하여 데이터를 가져올 때 사용합니다.
check-groovy-file	부울	true	등록하기 전에 groovy 파일 내용을 확인할지 여부를 지정합니다.

키	유형	기본값	설명
`qtemp.uuid.length`	number	9	QTEMP 고유 번호는 길이입니다.
`qtemp.dblog`	부울	false	QTEMP 데이터베이스 로깅 활성화 여부.
`qtemp.cleanup.threshold.hours`	number	0	`qtemp.dblog` 활성화 시기를 지정합니다. db 파티션 수명(시간).
`sort.function`	문자열		blu4iv 데이터베이스의 정렬 함수 이름.

선택적 웹 애플리케이션에 사용할 수 있는 속성

현대화된 애플리케이션에 따라 z/OS, AS/400 또는 IMS/MFS와 같은 종속성에 대한 지원을 나타내는 선택적 웹 애플리케이션을 하나 이상 구성해야 할 수 있습니다. 다음 표에는 각 선택적 웹 애플리케이션을 구성하는 데 사용할 수 있는 키/값 파라미터 목록이 나와 있습니다.

gapwalk-utility-pgm.war

이 선택적 웹 애플리케이션에는 Z/OS 유틸리티 프로그램에 대한 지원이 포함되어 있습니다.

이 표는 이 애플리케이션의 키/값 파라미터를 전체적으로 보여줍니다.

키	유형	기본값	설명
`logging.config`	경로	클래스 경로: logback-utility.xml	로그백 구성 파일에 대한 참조를 위한 표준 키입니다. 다른 표준 로깅 키도 사용할 수 있습니다.
`spring.jta.enabled`	부울	false	표준 키. 데이터 소스 지원 모드가 static-xa가 아닌 경우 스프링 JTA 트랜잭션 자동 구

키	유형	기본값	설명
			성을 비활성화해야 합니다.
`spring.datasource.primary.jndi-name`	문자열	jdbc/primary	JNDI를 사용하는 경우 기본 데이터 소스의 jndi 이름(Java 이름 지정 및 디렉터리 인터페이스).
`primary.datasource + -driver-class-name + -url + -username + -password`	하위 키가 있는 표준 스프링 데이터소스		JNDI를 사용하지 않는 경우 애플리케이션 데이터베이스의 연결 정보를 포함합니다. 현대화된 애플리케이션 yml 파일의 구성과 동일해야 합니다. 또는 xref:.. 에 설명된 대로 AWS 암호를 사용하는 것이 좋습니다. /구성/구성.adoc [구성].
`encoding`	문자열	ASCII	유틸리티 프로그램에 사용되는 인코딩. 올바른 `CP1047,IBM930,ASCII,UTF-8` 인코딩이 필요합니다.
`sysPunchEncoding`	문자열	ASCII	syspunch 인코딩 문자 세트. 올바른 인코딩 `CP1047,IBM930,ASCII,UTF-8` 이 필요합니다.

키	유형	기본값	설명
zonedMode	문자열	EBCDIC_STRICT	구역화된 데이터 유형을 인코딩 또는 디코딩하기 위한 모드입니다. 허용되는 값은 EBCDIC_STRICT/EBCDIC_MODIFIED/AS400입니다.
unload.chunkSize	number	0	언로드 유틸리티에 사용되는 청크 크기입니다.
unload.sqlCodePointShift	number	0	언로드 유틸리티의 SQL 코드 포인트 시프트. 문자 이동 프로세스를 실행합니다. DB2의 대상 데이터베이스가 Postgresql인 경우 필요합니다.
unload.columnFiller	문자열	space	언로드 유틸리티 열 필러.
unload.varCharIsNull	부울	false	INFTILB 프로그램에서 이 파라미터를 사용합니다. 이 파라미터를 설정하면 빈(공백) 값을 가진 null을 허용하지 않는 모든 필드는 빈 문자열을 반환합니다. true

키	유형	기본값	설명
unload.useDatabaseConfiguration	부울	false	언로드 유틸리티에서 application-main.yml의 날짜 또는 시간 구성을 사용할지 여부를 지정합니다.
unload.format.date	문자열	MM/dd/yyyy	unload.useDatabaseConfiguration 이 활성화된 경우 언로드 유틸리티에서 사용할 날짜 형식입니다.
unload.format.time	문자열	HH.MM.ss	활성화된 경우, 언로드 유틸리티에서 사용할 시간 unload.useDatabaseConfiguration 형식입니다.
unload.format.timestamp	문자열	YYYY-MM-DD-HH.mm.ss.ssssss	활성화된 경우 언로드 유틸리티에서 사용할 타임스탬프 형식입니다. unload.useDatabaseConfiguration
unload.nbi.whenNull	16진수	6F	데이터베이스의 값이 null일 때 추가할 널 바이트 표시기(nbi) 값입니다.

키	유형	기본값	설명
unload.nbi.whenNotNull	16진수	00	데이터베이스의 값이 null이 아닐 때 추가할 널 바이트 표시기(nbi) 값입니다.
unload.nbi.writeNullIndicator	부울	false	언로드 출력 파일에 널 인디케이터를 내보낼지 여부를 지정합니다.
unload.fetchSize	number	0	언로드 유틸리티에서 커서를 처리할 때 페치 크기를 조정할 수 있습니다.
treatLargeNumberAsInteger	부울	false	큰 숫자를 Integer과 같이 처리할지 여부를 지정합니다. 기본적으로 BigDecimal 와 같이 취급됩니다.
load.batchSize	number	0	로드 유틸리티 배치 크기.
load.format.localDate	문자열	dd.MM.yyyy\dd/MM/yyyy\yyyy-MM-dd	사용할 로드 유틸리티 로컬 날짜 형식입니다.
load.format.localTime	문자열	HH:mm:ss\HH.mm.ss	사용할 로드 유틸리티 현지 시간 형식입니다.
load.format.dbDate	문자열	yyyy-MM-dd	사용할 로드 유틸리티 데이터베이스 형식입니다.
load.format.dbTime	문자열	HH:mm:ss	로드 유틸리티 데이터베이스를 사용할 시간입니다.

키	유형	기본값	설명
load.sqlCodePointShift	number	0초	로드 유틸리티의 SQL 코드 포인트 시프트. 문자 이동 프로세스를 실행합니다. DB2의 대상 데이터베이스가 Postgresql인 경우 필요합니다.
forcedDate	문자열		제공된 날짜가 있는 경우 날짜를 지정된 날짜로 강제 적용합니다.
frozenDate	부울	true	날짜를 고정할지 여부를 지정합니다. forcedDate 도 설정된 경우에만 적용됩니다.
jcl.type	문자열	mvs	.jcl 파일 형식입니다. 허용되는 값은 jcl/vse입니다. IDCAMS 유틸리티 PRINT/REPRO 명령은 vse jcl이 아닌 경우 파일이 비어 있는 경우 4를 반환합니다.
hasGraphic	부울	false	INFUTILB 유틸리티가 GRAPHIC DB2 열을 처리해야 하는지 여부.

gapwalk-cl-command.war

이 선택적 웹 애플리케이션에는 AS/400 유틸리티 프로그램에 대한 지원이 포함되어 있습니다.

이 표는 이 애플리케이션의 키/값 파라미터를 전체적으로 보여줍니다.

키	유형	기본값	설명
`logging.config`	경로	클래스 경로: logback-utility.xml	로그백 구성 파일에 대한 참조를 위한 표준 키입니다. 다른 표준 로깅 키도 사용할 수 있습니다.
`spring.jta.enabled`	부울	false	표준 키. 데이터 소스 지원 모드가 static-xa가 아닌 경우 스프링 JTA 트랜잭션 자동 구성을 비활성화해야 합니다.
`spring.datasource.primary.jndi-name`	문자열	jdbc/primary	JNDI를 사용하는 경우 기본 데이터 소스의 jndi 이름(Java 이름 지정 및 디렉터리 인터페이스).
`primary.datasource + -driver-class-name + -url + -username + -password`	하위 키가 있는 표준 스프링 데이터소스		JNDI를 사용하지 않는 경우 애플리케이션 데이터베이스의 연결 정보를 포함합니다. 현대화된 애플리케이션 yml 파일의 구성과 동일해야 합니다. 또는 xref:.. 에 설명된 대로 AWS 암호를 사용하는 것이 좋습니다. /구성/구성.adoc [구성].
`encoding`	문자열	ASCII	유틸리티 프로그램에 사용되는 인코딩. 올바른

키	유형	기본값	설명
			CP1047,IBM930,ASCII,UTF-8 인코딩이 필요합니다.
zonedMode	문자열	EBCDIC_STRICT	구역화된 데이터 유형을 인코딩 또는 디코딩하기 위한 모드입니다. 허용되는 값은 EBCDIC_STRICT /EBCDIC_MODIFIED /AS400입니다.
commands-off	문자열		해제할 명령 목록(쉼표로 구분됨). 허용되는 값은 PGM_BASIC ,RCVMSG,SNDRCVF,CHGVAR,Q 니다. 기존 프로그램을 사용하지 않도록 설정하거나 덮어쓰려는 경우에 유용합니다. PGM_BASIC 는 디버그 목적으로 설계된 특정 속도 프로그램입니다.

gapwalk-hierarchical-support.war

이 선택적 웹 애플리케이션에는 IMS/MFS 트랜잭션 지원이 포함되어 있습니다.

이 표는 이 애플리케이션의 키/값 파라미터를 전체적으로 보여줍니다.

키	유형	기본값	설명
logging.config	경로	클래스 경로: logback-utility.xml	로그백 구성 파일에 대한 참조를 위한 표준 키입니다. 다른 표준

키	유형	기본값	설명
			로깅 키도 사용할 수 있습니다.
spring.jta.enabled	부울	false	표준 키. 데이터 소스 지원 모드가 static-xa가 아닌 경우 스프링 JTA 트랜잭션 자동 구성을 비활성화해야 합니다.
jhdb.configuration.context.encoding	문자열		JHDB(Java 계층적 데이터베이스) 인코딩. 유효한 인코딩 문자열 CP1047,IBM930,ASCII,UTF-8 이 필요합니다.

키	유형	기본값	설명
jhdb.checkpointPersistence	문자열	없음	체크포인트 지속성 모드. 허용되는 값은 none /add /end입니다. 새 체크포인트를 생성하여 레지스트리에 추가할 때 체크포인트를 유지하는 데 add를 사용합니다. 서버 종료 시 체크포인트를 유지하는 데 end를 사용합니다. 다른 값을 입력하면 지속성이 비활성화됩니다. 레지스트리에 새 체크포인트가 추가될 때마다 기존의 모든 체크포인트가 직렬화되고 파일이 지워진다는 점에 유의하세요. 파일의 기존 데이터에 추가되지 않습니다. 따라서 체크포인트 수에 따라 성능에 어느 정도 영향을 미칠 수 있습니다.

Gapwalk 애플리케이션에 대한 인증 구성

이 섹션에서는 Cognito, Azure AD, Keycloak 등과 같은 ID 공급자 (IdP) 를 사용하여 Gapwalk 애플리케이션에 대한 OAuth2 인증을 구성하는 방법에 대해 설명합니다.

주제

- 전제 조건
- Amazon Cognito 설정
- Gapwalk 애플리케이션에 Amazon Cognito 통합

전제 조건

이 자습서에서는 Amazon Cognito를 IdP로 사용하고 planetDemo를 현대화된 프로젝트로 사용합니다.

다른 모든 외부 ID 공급자를 사용할 수 있습니다. 이 ClientRegistration 정보는 IDP에서 가져와야 하며 갭워크 인증에 필요합니다. 자세한 내용은 IdP 문서를 참조하세요.

정보 ClientRegistration :

client-id

의 ID는 PlanetDemo입니다. 이 ClientRegistration 예에서는 PlanetDemo입니다.

client-secret

클라이언트 암호.

인증 엔드포인트

인증 서버의 인증 엔드포인트 URI입니다.

토큰 엔드포인트

권한 부여 서버의 토큰 엔드포인트 URI.

jwks 엔드포인트

권한 부여 서버에서 발급한 JSON 웹 서명의 유효성을 검사하기 위한 키가 포함된 JSON 웹 키(JWK)를 가져오는 데 사용되는 URI입니다.

리디렉션 URI

액세스 권한이 부여된 경우 권한 부여 서버가 최종 사용자를 리디렉션하는 URI입니다.

Amazon Cognito 설정

먼저 배포된 gapwalk 애플리케이션과 함께 테스트 목적으로 사용할 Amazon Cognito 사용자 풀과 사용자를 생성하고 구성합니다.

> **Note**
>
> 다른 IdP를 사용하는 경우 이 단계를 건너뛰어도 됩니다.

사용자 풀 생성

1. 에서 Amazon Cognito로 이동하여 자격 증명을 사용하여 인증하십시오. AWS Management Console AWS
2. 사용자 풀(User Pools)을 선택합니다.
3. 사용자 풀 생성을 선택합니다.
4. 로그인 환경 구성에서 Cognito 사용자 풀 기본 공급자 유형을 유지합니다. Cognito 사용자 풀 로그인 옵션을 하나 또는 여러 개 선택할 수 있습니다. 지금은 사용자 이름을 선택한 후 다음을 선택합니다.
5. 보안 요구 사항 구성에서 기본값을 유지하고 No MFA (MFA) 를 선택하고 다음을 선택하여 다단계 인증을 비활성화합니다.
6. 보안 조치로 셀프 등록 사용을 사용 중단하고 다음을 선택합니다.
7. Cognito로 이메일 보내기를 선택합니다. 다음을 선택합니다.
8. 앱 통합에서 사용자 풀의 이름을 선택합니다. 호스팅된 인증 페이지에서 Cognito 호스팅 UI 사용을 선택합니다.
9. 단순화를 위해 도메인에서 Cognito 도메인 사용을 선택하고 도메인 접두사를 입력합니다(예:). `https://planetsdemo` 데모 앱을 클라이언트로 추가해야 합니다.

    1. 초기 앱 클라이언트에서 기밀 클라이언트를 선택합니다. 앱 클라이언트 이름(예`planetsdemo`:)을 입력하고 클라이언트 암호 생성을 선택합니다.
    2. 허용된 콜백 URL에 인증 후 사용자를 리디렉션할 URL을 입력합니다. 임시 http://localhost:8080/planetsdemo URL을 사용한 다음 나중에 편집할 수도 있습니다.
    3. 고급 앱 클라이언트 설정 및 속성 읽기 및 쓰기 권한 섹션의 기본값을 유지하세요.
    4. 다음을 선택합니다.
10. 검토 및 생성에서 선택 사항을 확인한 다음 사용자 풀 생성을 선택합니다.

자세한 내용은 사용자 풀 생성을 참조하세요.

사용자 생성

자체 등록이 비활성화되었으므로 Amazon Cognito 사용자를 생성하세요. AWS Management Console 에서 Amazon Cognito 콘솔로 이동합니다. 생성한 사용자 풀을 선택한 다음 사용자에서 사용자 생성을 선택합니다.

사용자 정보에서 이메일 초대 보내기를 선택하고 사용자 이름과 이메일 주소를 입력한 다음 비밀번호 생성을 선택합니다. 사용자 생성을 선택합니다.

Gapwalk 애플리케이션에 Amazon Cognito 통합

이제 Amazon Cognito 사용자 풀과 사용자가 준비되었으므로 현대화된 애플리케이션의 `main-application.yml` 파일로 이동하여 다음 코드를 추가합니다.

```
spring:
  security:
    oauth2:
      client:
        registration:
          cognito:
            client-id: client-id
            client-name: client-name
            client-secret: client-secret
            provider: cognito
            authorization-grant-type: authorization_code
            scope: openid
            redirect-uri: "{baseUrl}/login/oauth2/code/{registrationId}"
        provider:
          cognito:
            issuerUri: ${gapwalk-application.security.issuerUri}
            authorization-uri: ${gapwalk-application.security.domainName}/oauth2/
authorize
            jwk-set-uri: ${gapwalk-application.security.issuerUri}/.well-known/
jwks.json
            token-uri: ${gapwalk-application.security.domainName}/oauth2/token
            user-name-attribute: cognito:username
      resourceserver:
        jwt:
          jwk-set-uri: ${gapwalk-application.security.issuerUri}/.well-known/jwks.json

gapwalk-application.security: enabled
gapwalk-application.security.identity: oauth

gapwalk-application.security.issuerUri: https://cognito-idp.region.amazonaws.com/pool-
id
gapwalk-application.security.domainName: your-cognito-domain
```

다음과 같이 자리 표시자를 바꿉니다.

1. 에서 Amazon Cognito로 이동하여 자격 증명을 사용하여 인증하십시오. AWS Management Console AWS

2. 사용자 풀을 선택하고 생성한 사용자 풀을 선택합니다. 사용자 풀 ID에서 *pool-id*를 찾을 수 있습니다.
3. 원하는 앱을 ***your-cognito-domain***찾을 수 있는 앱 통합을 선택하고 앱 클라이언트 및 분석으로 이동하여 앱을 선택하십시오.
4. 앱 클라이언트: YouApp에서 *client-name*, *client-id* 및 *client-secret*(클라이언트 암호 표시)를 찾을 수 있습니다.
5. *region-id*는 Amazon Cognito 사용자 및 `eu-west-3` 사용자 풀을 생성한 지역 ID에 해당합니다 (예:
6. redirect-uri의 경우 인증 후 사용자를 리디렉션할 URI를 입력합니다.

지금 Gapwalk 애플리케이션을 배포하고 이전에 생성한 사용자를 사용하여 앱에 로그인할 수 있습니다.

> **Note**
>
> 로그인할 때 “속성에 'cognito:username' 속성이 없습니다”라는 예외 오류가 발생하면 다음 줄을 삭제하세요.: cognito:username user-name-attribute

AWS 블루에이지 런타임 API

AWS Blu Age Runtime은 여러 웹 애플리케이션을 사용하여 REST 엔드포인트를 노출하여 REST 클라이언트를 사용하여 현대화된 애플리케이션과 상호 작용할 수 있는 방법을 제공합니다 (예: 스케줄러를 사용하여 작업 호출).

이 문서의 목적은 사용 가능한 REST 엔드포인트를 나열하여 다음에 대한 세부 정보를 제공하는 것입니다.

- 이들의 역할
- 제대로 사용하는 방법

엔드포인트 목록은 제공된 서비스의 특성과 엔드포인트를 노출하는 웹 애플리케이션에 따라 카테고리별로 구성되어 있습니다.

POSTMAN, Thunder Client, CURL, 웹 브라우저 등과 같은 전용 도구를 사용하여 REST 엔드포인트를 사용하는 데 대한 기본 지식을 이미 알고 있다고 가정합니다.) 직접 코드를 작성하여 API 직접 호출을 수행할 수도 있습니다.

주제

URL 빌드

아래의 각 웹 애플리케이션은 모든 엔드포인트에서 공유하는 루트 경로를 정의합니다. 그러면 각 엔드포인트가 자체 전용 경로를 추가합니다. 사용할 결과 URL은 경로를 연결한 결과입니다. 예를 들어 Gapwalk-Application의 첫 번째 엔드포인트를 고려하면 다음과 같은 결과를 얻을 수 있습니다.

- 루트 web-application 경로용 `/gapwalk-application`.
- 전용 엔드포인트 경로용 `/scripts`.

사용할 결과 URL은 `http://`*`server`*`:`*`port`*`/gapwalk-application/scripts`입니다

서버

서버 이름(해당 웹 애플리케이션을 호스팅하는 이름)을 가리킵니다.

포트

서버에 의해 노출된 포트.

Gapwalk 애플리케이션

Gapwalk 웹 애플리케이션의 엔드포인트는 루트 경로 `/gapwalk-application`를 사용합니다.

주제

일괄 Job(현대화된 JCL 등) 관련 엔드포인트

일괄 작업은 동기적으로 또는 비동기적으로 실행할 수 있습니다(아래 세부 정보 참조). 일괄 작업은 기존 스크립트(JCL)의 현대화에 따른 멋진 스크립트를 사용하여 실행되고 있습니다.

주제

- 배포된 스크립트 목록
- 스크립트를 동기적으로 실행
- 스크립트를 비동기적으로 실행
- 트리거된 스크립트 목록
- 작업 실행 세부 정보 검색
- 비동기적으로 실행되어 종료될 수 있는 스크립트를 나열합니다
- 동기적으로 실행된 스크립트 중 종료될 수 있는 스크립트를 나열합니다
- 특정 작업 실행 중단
- 재시작이 가능하도록 기존 체크포인트를 나열합니다
- 작업 재시작(동기식)
- 작업 재시작(비동기식)
- 비동기 작업 실행에 대한 스레드 제한 설정

배포된 스크립트 목록

- 지원되는 메서드: GET
- 경로: `/scripts`
- 인수: 없음
- 이 엔드포인트는 서버에 배포된 groovy 스크립트 목록을 문자열로 반환합니다. 결과 문자열은 활성 링크(실행 가능한 스크립트별 링크 -- 아래 샘플 참조)가 있는 HTML 페이지이므로 이 엔드포인트는 주로 웹 브라우저에서 사용하기 위한 것입니다.

샘플 응답:

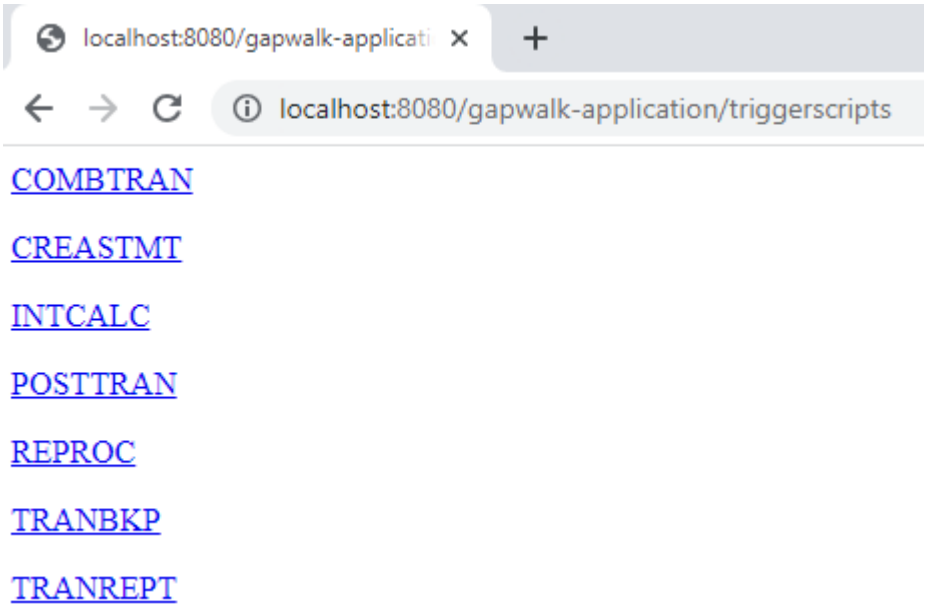
```
<p><a href=./script/COMBTRAN>COMBTRAN</a></p><p><a href=./script/CREASTMT>CREASTMT</
a></p><p><a href=./script/INTCALC>INTCALC</a></p><p><a href=./script/
POSTTRAN>POSTTRAN</a></p><p><a href=./script/REPROC>REPROC</a></p><p><a href=./script/
```

```
TRANBKP>TRANBKP</a></p><p><a href=./script/TRANREPT>TRANREPT</a></p><p><a href=./
script/functions>functions</a></p>
```

> **Note**
>
> 링크는 나열된 각 스크립트를 동기적으로 시작하는 데 사용할 URL을 나타냅니다.

- 지원되는 메서드: GET
- 경로: `/triggerscripts`
- 인수: 없음
- 이 엔드포인트는 서버에 배포된 groovy 스크립트 목록을 문자열로 반환합니다. 결과 문자열은 활성 링크(실행 가능한 스크립트별 링크 -- 아래 샘플 참조) 가 있는 HTML 페이지이므로 이 엔드포인트는 주로 웹 브라우저에서 사용하기 위한 것입니다.

  이전 엔드포인트 응답과 달리 링크는 나열된 각 스크립트를 비동기적으로 시작하는 데 사용할 URL을 나타냅니다.



스크립트를 동기적으로 실행

이 엔드포인트에는 GET 및 POST 사용 전용 경로가 있는 두 가지 변형이 있습니다(아래 참조).

- 지원되는 메서드: GET

- 경로: `/script/{scriptId:.+}`
- 지원되는 메서드: POST
- 경로: `/post/script/{scriptId:.+}`
- 인수:
  - 실행할 스크립트의 식별자
  - 선택 사항: 요청 파라미터(`Map<String,String>`로 표시)를 사용하여 스크립트에 전달할 파라미터. 지정된 파라미터는 호출된 groovy 스크립트의 바인딩에 자동으로 추가됩니다.
- 직접적 호출은 추가 파라미터(제공된 경우)를 사용하여 지정된 식별자를 사용하여 스크립트를 시작하고 스크립트 실행이 완료될 때까지 기다린 후 다음 중 하나와 같은 메시지(`String`)를 반환합니다.
  - “완료.” (작업 실행이 원활하게 실행된 경우).
  - 작업 실행 중 발생한 문제에 대한 세부 정보가 포함된 JSON 오류 메시지입니다. 서버 로그에서 추가 세부 정보를 검색하여 작업 실행에 어떤 문제가 발생했는지 파악할 수 있습니다.

```
{
    "exitCode": -1,
    "stepName": "STEP15",
    "program": "CBACT04C",
    "status": "Error"
}
```

서버 로그를 살펴보면 이것이 배포 문제라는 것을 알 수 있습니다(예상한 프로그램이 제대로 배포되지 않아 찾을 수 없어 작업 실행이 실패합니다).

```
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.BatchWebController          - --> executing script INTCALC
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.BatchWebController          - Bound jobContext 419695287 - GDGEventsQueueHandler :907380469
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.ScriptControlTower          - Added jobExecutor [a65c2791-864f-43c9-972a-b5f2353389e6] to Sync Script Control Tower.
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.j.s.JobExecutor               - a65c2791-864f-43c9-972a-b5f2353389e6 - worker :Thread-26 [1547512424]
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.j.s.JobExecutor               - Triggered script: INTCALC - [a65c2791-864f-43c9-972a-b5f2353389e6] - jobContext [419695287]
2023-06-09_10-27-29-613 | [JOB] INTCALC - Started
2023-06-09_10-27-29-651 | [STEP] STEP15 - Started
2023-06-09 10:27:29 default        ERROR - c.n.b.g.r.c.i.ExecutionControllerImpl    - Could not find program "CBACT04C" in the program registry.
2023-06-09 10:27:29 default        ERROR - c.n.b.g.r.c.i.ExecutionControllerImpl    - Could not find program "CBACT04C" in the program registry.
2023-06-09_10-27-29-760 | Program not found => not executed !
2023-06-09_10-27-29-761 | [STEP] STEP15 - Ended
2023-06-09_10-27-29-772 | [JOB] INTCALC - Ended
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.j.s.DefaultJobContext          - Job [419695287] - starting final operation
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.j.s.DefaultJobContext          - End of job [419695287]
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.s.ScriptControlTower          - Removed jobExecutor [a65c2791-864f-43c9-972a-b5f2353389e6] from Script Control Tower.
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.s.ScriptControlTower          - Remaining jobExecutors:0
```

> **Note**
>
> 동기 호출은 단시간 실행 작업에만 사용해야 합니다. 장시간 실행되는 작업은 비동기적으로 시작하는 것이 좋습니다(아래 전용 엔드포인트 참조).

스크립트를 비동기적으로 실행

- 지원되는 메서드: GET/POST
- 경로: `/triggerscript/{scriptId:.+}`
- 인수:
  - 실행할 스크립트의 식별자
  - 선택 사항: 요청 파라미터(`Map<String,String>`로 표시)를 사용하여 스크립트에 전달할 파라미터. 지정된 파라미터는 호출된 groovy 스크립트의 https://docs.groovy-lang.org/latest/html/api/groovy/lang/Binding.html[바인딩]에 자동으로 추가됩니다.
- 위의 동기 모드와 달리 엔드포인트는 응답을 보내기 위해 작업 실행이 완료될 때까지 기다리지 않습니다. 사용 가능한 스레드를 찾을 수 있는 경우 작업 실행이 한 번에 시작되고 작업 실행을 나타내는 고유 식별자인 작업 실행 ID와 함께 호출자에게 즉시 응답이 전송됩니다. 이 응답은 작업 실행 상태를 쿼리하거나 오작동으로 간주되는 작업 실행을 강제 종료하는 데 사용할 수 있습니다. 리소스 레이블의 형식은 다음과 같습니다.

  ```
  Triggered script <script identifier> [unique job execution id] @ <date and time>
  ```

- 작업 비동기 실행은 고정된 제한된 수의 스레드를 사용하므로 사용 가능한 스레드를 찾을 수 없는 경우 작업 실행이 시작되지 않을 수 있습니다. 이 경우 반환되는 메시지는 다음과 같습니다.

  ```
  Script [<script identifier>] NOT triggered - Thread limit reached (<actual thread
   limit>) - Please retry later or increase thread limit.
  ```

  스레드 제한을 늘리는 방법을 알아보려면 아래 `settriggerthreadlimit` 엔드포인트를 참조하세요.

샘플 응답:

```
Triggered script INTCALC [d43cbf46-4255-4ce2-aac2-79137573a8b4] @ 06-12-2023 16:26:15
```

고유한 작업 실행 식별자를 사용하면 필요한 경우 서버 로그에서 관련 로그 항목을 빠르게 검색할 수 있습니다. 또한 아래에 설명된 다른 여러 엔드포인트에서도 사용됩니다.

트리거된 스크립트 목록

- 지원되는 메서드: GET

- 경로: `/triggeredscripts/{status:.+}`, `/triggeredscripts/{status:.+}/{namefilter}`
- 인수:
  - 상태(필수): 검색할 트리거된 스크립트의 상태입니다. 가능한 값은 다음과 같습니다.
    - `all`: 작업이 아직 실행 중인지 여부에 관계없이 모든 작업 실행 세부 정보를 표시합니다.
    - `running`: 현재 실행 중인 작업에 대한 작업 세부 정보만 표시합니다.
    - `done`: 실행이 끝난 작업에 대한 작업 세부 정보만 표시합니다.
    - `killed`: 전용 엔드포인트를 사용하여 실행이 강제로 중단된 작업에 대한 작업 세부 정보만 표시합니다(아래 참조).
    - `triggered`: 트리거되었지만 아직 시작되지 않은 작업에 대한 작업 세부 정보만 표시합니다.
    - `failed`: 실행이 실패로 표시된 작업에 대한 작업 세부 정보만 표시합니다.
    - _namefilter(선택 사항)_: 지정된 스크립트 식별자에 대한 실행만 검색합니다.
- 작업 실행 세부 정보 모음을 JSON으로 반환합니다. 자세한 설명은 Job Execution 세부 정보 메시지 구조 섹션을 참조하세요.

샘플 응답:

```
[
    {
      "scriptId": "INTCALC",
      "caller": "127.0.0.1",
      "identifier": "d43cbf46-4255-4ce2-aac2-79137573a8b4",
      "startTime": "06-12-2023 16:26:15",
      "endTime": "06-12-2023 16:26:15",
      "status": "DONE",
      "executionResult": "{ \"exitCode\": -1, \"stepName\": \"STEP15\", \"program\":
 \"CBACT04C\", \"status\": \"Error\" }",
      "executionMode": "ASYNCHRONOUS"
    }
 ]
```

작업 실행 세부 정보 검색

- 지원되는 메서드: GET
- 경로: `/getjobexecutioninfo/{jobexecutionid:.+}`
- 인수:

  - jobexecutionid(필수): 해당 작업 실행 세부 정보를 검색하기 위한 고유한 작업 실행 식별자입니다.
- 반환값: 단일 작업 실행 세부 정보(Job Execution 세부 정보 메시지 구조 참조)를 나타내는 JSON 문자열, 지정된 식별자에 대한 작업 실행 세부 정보를 찾을 수 없는 경우 빈 응답.

비동기적으로 실행되어 종료될 수 있는 스크립트를 나열합니다

- 지원되는 메서드: GET
- 경로: `/killablescripts`
- 비동기적으로 시작되었지만 현재 실행 중이고 강제로 종료될 수 있는 작업의 작업 실행 식별자 컬렉션을 반환합니다(아래 `/kill` 엔드포인트 참조).

동기적으로 실행된 스크립트 중 종료될 수 있는 스크립트를 나열합니다

- 지원되는 메서드: GET
- 경로: `/killablesyncscripts`
- 동기적으로 시작되어 현재 실행 중이고 강제로 종료될 수 있는 작업의 작업 실행 식별자 컬렉션을 반환합니다(아래 `/kill` 엔드포인트 참조).

특정 작업 실행 중단

- 지원되는 메서드: GET
- 경로: `/kill/{identifier:.+}`
- 인수: 작업 실행 식별자(필수): 강제로 종료될 작업 실행을 가리키는 고유한 작업 실행 식별자입니다.
- 반환값: 작업 실행 중단 시도 결과를 자세히 설명하는 텍스트 메시지입니다. 메시지에는 스크립트 식별자, 작업 실행 고유 식별자, 실행 중단이 발생한 날짜 및 시간이 포함됩니다. 지정된 식별자에 대해 실행 중인 작업 실행을 찾을 수 없는 경우 대신 오류 메시지가 반환됩니다.

> ⚠ Warning
>
> - 런타임은 대상 작업 실행을 제대로 종료하기 위해 최선을 다합니다. 따라서 AWS Blu Age 런타임은 작업 중단이 비즈니스에 미치는 영향을 최소화하려고 하기 때문에 /kill 엔드포인트의 응답이 호출자에게 도달하는 데 시간이 약간 걸릴 수 있습니다.

- 작업 실행을 강제 종료하는 것은 데이터 손실이나 손상을 비롯한 직접적인 비즈니스 결과를 초래할 수 있으므로 가볍게 해서는 안 됩니다. 특정 작업 실행이 중단되고 데이터 수정 수단이 명확하게 식별된 경우에만 사용해야 합니다.
- 작업을 중단하면 추가 조사(사후 분석) 를 통해 무엇이 잘못되었는지 파악하고 적절한 수정 조치를 취해야 합니다.
- 어떤 경우든 실행 중인 작업을 중단하려는 시도는 경고 수준 메시지와 함께 서버 로그에 기록됩니다.

재시작이 가능하도록 기존 체크포인트를 나열합니다

작업 재시작 가능성은 스크립트가 체크포인트를 CheckpointRegistry에 등록하여 작업 실행 진행 상황을 추적할 수 있는지에 달려 있습니다. 작업 실행이 제대로 종료되지 않고 재시작 체크포인트가 등록된 경우 체크포인트 위의 단계를 실행할 필요 없이 마지막으로 등록된 체크포인트에서 간단히 작업 실행을 다시 시작할 수 있습니다.

- 지원되는 메서드: GET
- 경로: `/restarts`
- 실행이 제대로 이루어지지 않았다가 종료되지 않은 작업을 다시 시작하는 데 사용할 수 있는 기존 재시작 지점 목록을 html 페이지로 반환합니다. 스크립트에서 체크포인트를 등록하지 않은 경우 페이지 내용은 “등록된 체크포인트 없음”으로 표시됩니다.

작업 재시작(동기식)

- 지원되는 메서드: GET
- 경로: `/restart/{hashcode}`
- 인수: hashcode(정수 - 필수): 제공된 해시코드를 체크포인트 값으로 사용하여 이전에 중단된 작업 실행을 다시 시작합니다(유효한 체크포인트 값을 검색하는 방법을 알아보려면 위의 `/restarts` 엔드포인트를 참조하세요).
- 반품: 위의 `script` 반환 설명을 참조하세요.

작업 재시작(비동기식)

- 지원되는 메서드: GET
- 경로: `/triggerrestart/{hashcode}`

- 인수: hashcode(정수 - 필수): 제공된 해시코드를 체크포인트 값으로 사용하여 이전에 중단된 작업 실행을 다시 시작합니다(유효한 체크포인트 값을 검색하는 방법을 알아보려면 위의 `/restarts` 엔드포인트를 참조하세요).
- 반품: 위의 `triggerscript` 반환 설명을 참조하세요.

비동기 작업 실행에 대한 스레드 제한 설정

작업 비동기 실행은 JVM의 전용 스레드 풀을 사용합니다. 이 풀에는 사용 가능한 스레드 수에 대한 고정된 제한이 있습니다. 사용자는 호스트 성능(CPU 수, 사용 가능한 메모리 등) 에 따라 제한을 조정할 수 있습니다. 기본적으로 스레드 제한은 5개 스레드로 설정됩니다.

- 지원되는 메서드: GET
- 경로: `/settriggerthreadlimit/{threadlimit:.+}`
- 인수(정수): 적용할 새 스레드 제한. 반드시 양의 정수여야 합니다.
- 새 스레드 제한과 이전 스레드 제한을 제공하는 메시지(`String`)를 반환하거나, 제공된 스레드 제한 값이 유효하지 않은 경우(양수 정수가 아님) 오류 메시지를 반환합니다.

샘플 응답:

```
Set thread limit for Script Tower Control to 10 (previous value was 5)
```

현재 실행 중인 트리거된 작업 실행 횟수 계산

- 지원되는 메서드: GET
- 경로: `/countrunningtriggeredscripts`
- 비동기적으로 시작된 실행 중인 작업 수와 스레드 제한(동시에 실행할 수 있는 트리거된 작업의 최대 수) 을 나타내는 메시지를 반환합니다.

샘플 응답:

```
0 triggered script(s) running (limit =10)
```

> **Note**
>
> 작업을 시작하기 전에 스레드 제한에 도달하지 않았는지(이로 인해 작업이 시작되지 않음) 되었는지 확인하는 데 사용할 수 있습니다.

작업 실행 정보 제거

작업 실행 정보는 서버가 가동되는 동안 서버 메모리에 남아 있습니다. 가장 오래된 정보는 더 이상 관련이 없으므로 메모리에서 지우는 것이 편리할 수 있습니다. 이것이 이 엔드포인트의 목적입니다.

- 지원되는 메서드: GET
- 경로: `/purgejobinformation/{age:.+}`
- 인수: 제거할 정보의 보존 기간(시간) 을 나타내는 양수 정수 값입니다.
- 다음 정보가 포함된 메시지를 반환합니다.
  - 제거된 작업 실행 정보가 보관 목적으로 저장되는 제거 파일의 이름입니다.
  - 제거된 작업 실행 정보 수.
  - 메모에 남아 있는 작업 실행 정보 수

지표 엔드포인트

JVM

이 엔드포인트는 JVM과 관련된 사용 가능한 메트릭을 반환합니다.

- 지원되는 메서드: GET
- 경로: `/metrics/jvm`
- 인수: 없음
- 다음 정보가 포함된 메시지를 반환합니다.
  - threadActiveCount: 활성 스레드 수.
  - jvmMemoryUsed: Java 가상 머신에서 활발히 사용하는 메모리.
  - jvmMemoryMax: Java 가상 머신에 허용되는 최대 메모리입니다.
  - jvmMemoryFree: 사용 가능한 메모리는 현재 Java 가상 머신에서 사용하고 있지 않습니다.

세션

이 엔드포인트는 현재 열려 있는 HTTP 세션과 관련된 메트릭을 반환합니다.

- 지원되는 메서드: GET
- 경로: `/metrics/session`
- 인수: 없음
- 다음 정보가 포함된 메시지를 반환합니다.
  - 세션 수: 서버에서 현재 유지 관리하는 활성 사용자 세션 수입니다.

일괄

- 지원되는 메서드: GET
- 경로: `/metrics/batch`
- 인수:
  - startTimestamp(선택 사항, 숫자): 데이터 필터링을 위한 시작 타임스탬프입니다.
  - endTimestamp(선택 사항, 숫자): 데이터 필터링의 종료 타임스탬프입니다.
  - page(선택 사항, 번호): 페이지 매김을 위한 페이지 번호입니다.
  - pageSize(선택 사항, 숫자): 페이지 매김의 페이지당 항목 수입니다.
- 다음 정보가 포함된 메시지를 반환합니다.
  - content: 일괄 실행 지표 목록.
  - pageNumber: 페이지 매김의 현재 페이지 번호.
  - pageSize: 페이지당 표시되는 항목 수입니다.
  - totalPages: 사용 가능한 총 페이지 수입니다.
  - numberOfElements: 현재 페이지의 항목 수.
  - last: 마지막 페이지의 부울 플래그.
  - first: 첫 페이지의 부울 플래그.

트랜잭션

- 지원되는 메서드: GET
- 경로: `/metrics/transaction`
- 인수:

  - startTimestamp(선택 사항, 숫자): 데이터 필터링을 위한 시작 타임스탬프입니다.
  - endTimestamp(선택 사항, 숫자): 데이터 필터링의 종료 타임스탬프입니다.
  - page(선택 사항, 번호): 페이지 매김을 위한 페이지 번호입니다.
  - pageSize(선택 사항, 숫자): 페이지 매김의 페이지당 항목 수입니다.
- 다음 정보가 포함된 메시지를 반환합니다.
  - content: 트랜잭션 실행 지표 목록.
  - pageNumber: 페이지 매김의 현재 페이지 번호.
  - pageSize: 페이지당 표시되는 항목 수입니다.
  - totalPages: 사용 가능한 총 페이지 수입니다.
  - numberOfElements: 현재 페이지의 항목 수.
  - last: 마지막 페이지의 부울 플래그.
  - first: 첫 페이지의 부울 플래그.

기타 엔드포인트

이러한 엔드포인트를 사용하여 등록된 프로그램 또는 서비스를 나열하고, 상태를 확인하고, JICS 트랜잭션을 관리할 수 있습니다.

주제

등록된 프로그램 목록

- 지원되는 메서드: GET
- 경로: `/programs`
- 등록된 프로그램 목록을 html 페이지로 반환합니다. 각 프로그램은 기본 프로그램 식별자로 지정됩니다. 현대화된 레거시 프로그램과 유틸리티 프로그램(IDCAMS, IEBGENER 등)이 모두 목록에 반환됩니다. 사용 가능한 유틸리티 프로그램은 tomcat 서버에 배포된 유틸리티 웹 애플리케이션에 따

라 달라진다는 점에 유의하세요. 예를 들어 z/OS 유틸리티 지원 프로그램은 현대화된 iSeries 자산과 관련이 없으므로 사용하지 못할 수 있습니다.

등록된 서비스 목록

- 지원되는 메서드: GET
- 경로: `/services`
- 등록된 런타임 서비스 목록을 html 페이지로 반환합니다. 주어진 서비스는 AWS Blu Age 런타임에서 유틸리티로 제공되며, 예를 들어 그루비 스크립트에서 사용할 수 있습니다. Blusam 로드 서비스(레거시 데이터세트에서 Blusam 데이터세트를 생성하는 서비스) 가 해당 범주에 속합니다.

샘플 응답:

```
<p>BluesamESDSFileLoader</p><p>BluesamKSDSFileLoader</p><p>BluesamRRDSFileLoader</p>
```

상태 확인

- 지원되는 메서드: GET
- 경로: `/`
- gapwalk-애플리케이션이 실행 중임을 나타내는 간단한 메시지를 반환합니다(`Jics application is running.`)

사용 가능한 JICS 트랜잭션 나열

- 지원되는 메서드: GET
- 경로: `/transactions`
- 사용 가능한 모든 JICS 트랜잭션을 나열하는 html 페이지를 반환합니다. 이는 JICS 요소(기존 CICS 요소의 현대화)가 있는 환경에서만 의미가 있습니다.

샘플 응답:

```
<p>INQ1</p><p>MENU</p><p>MNT2</p><p>ORD1</p><p>PRNT</p>
```

JICS 트랜잭션 시작

- 지원되는 메서드: GET, POST

- 경로: `/jicstransrunner/{jtrans:.+}`
- 인수:
  - JICS 트랜잭션 식별자(문자열, 필수): 시작할 JICS 트랜잭션의 식별자(최대 8자)
  - 필수: 맵<String, Object>으로 트랜잭션에 전달할 추가 입력 데이터. 이 맵의 내용은 JICS 트랜잭션에서 사용될 COMMAREA를 공급하는 데 사용됩니다. 트랜잭션을 실행하는 데 데이터가 필요하지 않은 경우 맵이 비어 있을 수 있습니다.
  - 선택 사항: 주어진 트랜잭션의 실행 환경을 사용자 지정하기 위한 HTTP 헤더 항목. 다음과 같은 헤더 키가 지원됩니다.
    - `jics-channel`: 이번 거래 개시에 따라 실행될 프로그램에서 사용할 JICS CHANNEL의 이름입니다.
    - `jics-container`: 이번 JICS 트랜잭션 실행에 사용될 JICS CONTAINER의 이름.
    - `jics-startcode`: JICS 트랜잭션 시작 시 사용할 STARTCODE(문자열, 최대 2자). 가능한 값은 STARTCODE를 참조하세요(페이지 아래로 이동).
    - `jicxa-xid`: 호출자가 시작한 “글로벌 트랜잭션”(XA)의 XID(X/오픈 트랜잭션 식별자 XID 구조)로, 현재 JICS 트랜잭션 실행에 참여합니다.
- 반환: `com.netfective.bluage.gapwalk.rt.shared.web.TransactionResultBean` JICS 트랜잭션 시작 결과를 나타내는 JSON 직렬화.

구조에 대한 자세한 내용은 트랜잭션 시작 결과 구조 섹션을 참조하세요.

JICS 트랜잭션 시작(대안)

- 지원되는 메서드: GET, POST
- path: `/jicstransaction/{jtrans:.+}`
- 인수:

  JICS 트랜잭션 식별자(문자열, 필수)

  시작할 JICS 트랜잭션의 식별자(최대 8자)

  필수: 트랜잭션에 Map<String, Object>으로 전달할 추가 입력 데이터

  이 맵의 내용은 JICS 트랜잭션에서 사용될 COMMAREA를 공급하는 데 사용됩니다. 트랜잭션을 실행하는 데 데이터가 필요하지 않은 경우 맵이 비어 있을 수 있습니다.

  선택 사항: 주어진 트랜잭션의 실행 환경을 사용자 지정하기 위한 HTTP 헤더 항목.

  다음과 같은 헤더 키가 지원됩니다.

  - `jics-channel`: 이번 거래 개시에 따라 실행될 프로그램에서 사용할 JICS CHANNEL의 이름입니다.
  - `jics-container`: 이번 JICS 트랜잭션 실행에 사용될 JICS CONTAINER의 이름.
  - `jics-startcode`: JICS 트랜잭션 시작 시 사용할 STARTCODE(문자열, 최대 2자). 가능한 값은 STARTCODE(페이지 아래로 이동)를 참조하세요.
  - `jicxa-xid`: 호출자가 시작한 "글로벌 트랜잭션"(XA)의 XID(X/오픈 트랜잭션 식별자 XID 구조)로, 현재 JICS 트랜잭션 실행에 참여합니다.
- 반환: `com.netfective.bluage.gapwalk.rt.shared.web.RecordHolderBean` JICS 트랜잭션 시작 결과를 나타내는 JSON 직렬화. 구조에 대한 자세한 내용은 거래 시작 기록 결과 구조에서 확인할 수 있습니다.

작업 대기열 관련 엔드포인트

Job Queue는 AS400 작업 제출 메커니즘에 대한 AWS 블루 에이지 지원입니다. AS400 작업 대기열은 특정 스레드 풀에서 작업을 실행하는 데 사용됩니다. 작업 대기열은 해당 대기열에서 동시에 실행할 수 있는 최대 프로그램 수에 해당하는 이름 및 최대 스레드 수로 정의됩니다. 대기열에 제출된 작업이 최대 스레드 수보다 많은 경우 작업은 스레드를 사용할 수 있을 때까지 대기합니다.

대기열에 있는 작업의 전체 상태 목록은 큐에 있는 작업의 가능한 상태을 참조하세요.

작업 대기열에 대한 작업은 다음 전용 엔드포인트를 통해 처리됩니다. 다음 루트 URL을 사용하여 Gapwalk 애플리케이션 URL에서 이러한 작업을 호출할 수 있습니다. `http://server:port/gapwalk-application/jobqueue`

주제

- 사용 가능한 대기열 나열
- 작업 대기열 시작 또는 재시작
- 시작을 위해 작업을 제출하세요
- 예약된 모든 작업을 나열합니다
- "보류 중" 상태인 모든 작업을 나열하세요
- 모든 활성 작업 목록
- 시작 대기 중인 모든 작업을 나열하세요
- "보류 중" 상태인 모든 작업을 해제하세요
- 해당 작업 이름에 대해 "보류 중" 상태인 모든 작업을 릴리스하세요
- 주어진 작업을 릴리즈하여 작업 번호를 확인하세요

사용 가능한 대기열 나열

- 지원되는 메서드: GET
- 경로: `list-queues`
- 사용 가능한 대기열 목록을 상태와 함께 JSON 키-값 목록으로 반환합니다.

샘플 응답:

```
{"Default":"STAND_BY","queue1":"STARTED","queue2":"STARTED"}
```

가능한 작업 대기열 상태는 다음과 같습니다.

STAND_BY

작업 대기열이 시작되기를 기다리고 있습니다.

STARTED

작업 대기열이 가동되어 실행 중입니다.

UNKNOWN

작업 대기열 상태를 확인할 수 없습니다.

작업 대기열 시작 또는 재시작

- 지원되는 메서드: POST
- 경로: `/restart/{name}`
- 인수: 시작/재시작할 큐의 이름(문자열) - 필수.
- 엔드포인트는 아무 것도 반환하지 않고 http 상태를 기반으로 시작/재시작 작업의 결과를 표시합니다.

  HTTP 200

  시작/재시작 작업이 순조롭게 진행되었습니다. 지정된 작업 대기열이 이제 시작되었습니다.

  HTTP 404

  작업 대기열이 존재하지 않습니다.

HTTP 503

시작/재시작 시도 중에 예외가 발생했습니다(서버 로그를 검사하여 무엇이 잘못되었는지 확인해야 함).

시작을 위해 작업을 제출하세요

- 지원되는 메서드: POST
- 인수: 요청 본문으로 필수이며, `com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage` 객체의 JSON 직렬화입니다. 자세한 설명은 작업 입력 제출 섹션을 참조하세요.
- 반환: 원본 `SubmitJobMessage`과 작업이 제출되었는지 여부를 나타내는 로그가 포함된 JSON.

예약된 모든 작업을 나열합니다

- 지원되는 메서드: GET
- 경로: `list-jobs`
- 반환값: 예약된 모든 작업의 목록(JSON 문자열). 샘플 응답은 예정된 작업 응답 목록을 참조하세요.

“보류 중” 상태인 모든 작업을 나열하세요

- 지원되는 메서드: GET
- 경로: `list-jobs-hold`
- 반환값: 예약된 모든 작업의 목록(JSON 문자열). 샘플 응답은 '보류 중' 작업 응답 목록을 참조하세요.

모든 활성 작업 목록

- 지원되는 메서드: GET
- 경로: `list-jobs-active`
- 반환: 상태가 활성인 모든 작업의 목록(JSON 문자열). 응답은 예정된 작업 응답 목록의 응답과 구조가 비슷합니다.

시작 대기 중인 모든 작업을 나열하세요

- 지원되는 메서드: GET
- 경로: `list-jobs-waiting`
- 반환값: EXECUTION_WAIT 상태의 모든 작업(스레드를 시작할 수 있을 때까지 기다리는 작업)의 목록을 JSON 문자열로 표시합니다. 응답은 the section called “예정된 작업 응답 목록”의 응답과 구조가 비슷합니다.

“보류 중” 상태인 모든 작업을 해제하세요

- 지원되는 메서드: POST
- 경로: `release-all`
- 반환: 릴리스 시도 작업의 결과를 나타내는 메시지입니다. 다음과 같은 두 가지 경우가 발생할 수 있습니다.
  - HTTP 200 및 “모든 작업이 성공적으로 릴리스되었습니다!”라는 메시지 모든 작업이 성공적으로 릴리스된 경우.
  - HTTP 503 및 메시지: “작업이 공개되지 않았습니다. 알 수 없는 오류가 발생했습니다. 출시 시도에 문제가 발생한 경우 자세한 내용은 로그를 참조하세요.

해당 작업 이름에 대해 “보류 중” 상태인 모든 작업을 릴리스하세요

주어진 작업 이름에 대해 작업 번호가 서로 다른 여러 개의 작업을 제출할 수 있습니다(작업 실행의 통일성은 두 개<job name, job number>로 보장됨). 엔드포인트는 지정된 작업 이름을 가진 “보류 중” 상태인 모든 작업 제출을 릴리스하려고 시도합니다.

- 지원되는 메서드: POST
- 경로: `/release/{name}`
- 인수: 검색할 작업 이름(문자열). 필수.
- 반환: 릴리스 시도 작업의 결과를 나타내는 메시지입니다. 다음과 같은 두 가지 경우가 발생할 수 있습니다.
  - HTTP 200 및 “<name>(<number of released jobs>) 그룹 내 작업이 성공적으로 릴리스되었습니다!”라는 메시지 작업이 성공적으로 공개되었습니다.
  - HTTP 503 및 릴리스 시도에 문제가 발생한 경우 “<name> 그룹 내 작업이 공개되지 않았습니다. 알 수 없는 오류가 발생했습니다. 자세한 정보는 로그를 참조하세요”라는 메시지.

주어진 작업을 릴리즈하여 작업 번호를 확인하세요

엔드포인트는 해당 쌍<job name, job number>에 대해 “보류 중”인 고유한 작업 제출을 릴리스하려고 시도합니다.

- 지원되는 메서드: POST
- 경로: `/release/{name}/{number}`
- 인수:

  이름

  검색할 작업 이름(문자열). 필수.

  number

  찾으려는 작업 번호(정수). 필수.

  반환

  릴리즈 시도 작업의 결과를 나타내는 메시지입니다. 다음과 같은 두 가지 경우가 발생할 수 있습니다.

  - HTTP 200 및 메시지 “" Job이 <name/number> 성공적으로 릴리스되었습니다!” 작업이 성공적으로 릴리스된 경우.
  - HTTP 503 및 릴리스 시도에 문제가 발생한 경우 “<name/number> 작업이 릴리스되지 않았습니다. 알 수 없는 오류가 발생했습니다. 자세한 정보는 로그를 참조하세요”라는 메시지.

Blusam 애플리케이션 콘솔 REST 엔드포인트

Blusam 애플리케이션 콘솔은 현대화된 VSAM 데이터 세트의 관리를 단순화하도록 설계된 API입니다. Blusam 웹 애플리케이션의 엔드포인트는 루트 경로 `/bac`를 사용합니다.

주제

데이터 세트 관련 엔드포인트

다음 엔드포인트를 사용하여 특정 데이터 세트를 만들거나 관리합니다.

주제

데이터 세트 생성

데이터 세트 엔드포인트를 생성하면 데이터 세트 정의를 생성할 수 있으며 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/bluesamservice/createDataSet`
- 인수:

  이름

  (필수, 문자열): 데이터 세트의 이름입니다.

  유형

  (필수, 문자열): 데이터 세트 유형. 가능한 값은 ESDS, KSDS, RRDS입니다.

  recordSize

  (선택 사항, 문자열): 데이터 세트에 있는 각 레코드의 최대 크기입니다.

  fixedLength

  (선택 사항, 부울): 레코드 길이가 고정되었는지 여부를 나타냅니다.

  압축

  (선택 사항, 부울): 데이터 세트가 압축되었는지 여부를 나타냅니다.

cacheEnable

(선택 사항, 부울): 데이터 세트에 캐싱이 활성화되었는지 여부를 나타냅니다.

alternativeKeys

(선택 사항, 키 목록):

- 오프셋(필수, 숫자)
- 길이(필수, 숫자)
- 이름(필수, 번호)

- 새로 만든 데이터 세트를 나타내는 json 파일을 반환합니다.

샘플 요청:

```
POST /api/services/rest/bluesamservice/createDataSet
{
  "name": "DATASET",
  "checked": false,
  "records": [],
  "primaryKey": {
    "name": "PK"
  },
  "alternativeKeys": [
    {
      "offset": 10,
      "length": 10,
      "name": "ALTK_0"
    }
  ],
  "type": "ESDS",
  "recordSize": 10,
  "compression": true,
  "cacheEnable": true
}
```

샘플 응답:

```
{
    "dataSet": {
      "name": "DATASET",
      "checked": false,
```

```
      "nbRecords": 0,
      "keyLength": -1,
      "recordSize": 10,
      "compression": false,
      "fixLength": true,
      "type": "ESDS",
      "cacheEnable": false,
      "cacheWarmup": false,
      "cacheEviction": "100ms",
      "creationDate": 1686744961234,
      "modificationDate": 1686744961234,
      "records": [],
      "primaryKey": {
        "name": "PK",
        "offset": null,
        "length": null,
        "columns": null,
        "unique": true
      },
      "alternativeKeys": [
        {
          "offset": 10,
          "length": 10,
          "name": "ALTK_0"
        }
      ],
      "readLimit": 0,
      "readEncoding": null,
      "initCharacter": null,
      "defaultCharacter": null,
      "blankCharacter": null,
      "strictZoned": null,
      "decimalSeparator": null,
      "currencySign": null,
      "pictureCurrencySign": null
    },
    "message": null,
    "result": true
  }
```

파일 업로드

이 엔드포인트는 서버에 파일을 업로드할 수 있습니다. 파일은 각 특정 사용자에 해당하는 임시 폴더에 저장됩니다. 이 엔드포인트는 파일을 업로드해야 할 때마다 사용해야 합니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/bluesamservice/upload`
- 인수:

  파일

  (필수, 멀티파트/양식 데이터): 업로드할 파일입니다.
- 업로드의 상태를 반영하는 부울을 반환합니다

데이터 세트 로드

앞서 설명한 데이터 세트 생성 엔드포인트를 사용하여 데이터 세트 정의를 생성한 후에는 업로드된 파일에 연결된 레코드를 특정 데이터 세트에 로드할 수 있습니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/bluesamservice/loadDataSet`
- 인수:

  이름

  (필수, 문자열): 데이터 세트의 이름입니다.
- 요청 상태와 로드된 데이터 세트를 반환합니다.

Amazon S3 버킷에서 데이터 세트를 로드할 수 있습니다

Amazon S3 버킷에서 listcat 파일을 사용하여 데이터 세트를 로드합니다.

- 지원되는 메서드: GET
- 경로: `` `/api/services/rest/bluesamservice/loadDataSetFromS3` ``
- 인수:

  listcatFileS3Location

  (필수, 문자열): listcat 파일의 Amazon S3 위치입니다.

datasetFileS3Location

(필수, 문자열): 데이터 세트 파일의 Amazon S3 위치입니다.

region

(필수, 문자열): 파일이 AWS 리전 저장되는 Amazon S3입니다.

- 새로 생성된 데이터 세트를 반환합니다

샘플 요청:

```
/BAC/api/services/rest/bluesamservice/loadDataSetFromS3?region=us-
east-1&listcatFileS3Location=s3://bucket-name/listcat.json&datasetFileS3Location=s3://
bucket-name/dataset.DAT
```

Amazon S3 버킷으로 데이터 세트를 내보냅니다

지정된 Amazon S3 버킷으로 데이터 세트를 내보냅니다.

- 지원되는 메서드: GET
- 경로: `/api/services/rest/bluesamservice/exportDataSetToS3`
- 인수:

s3Location

(필수, 문자열): 데이터 세트를 내보낼 Amazon S3 위치입니다.

datasetName

(필수, 문자열): 내보낼 데이터 세트의 이름.

region

(필수, 문자열): Amazon S3 AWS 리전 버킷의.

- 내보낸 데이터 세트를 반환합니다

샘플 요청:

```
/BAC/api/services/rest/bluesamservice/exportDataSetToS3?region=eu-
west-1&s3Location=s3://bucket-name/dump&datasetName=dataset
```

데이터 세트 생성

데이터 세트에서 모든 레코드를 지웁니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/bluesamservice/clearDataSet`
- 인수:

  이름

  (필수, 문자열): 지울 데이터 세트의 이름.
- 반환: 요청 상태를 반환합니다.

데이터 스토어 삭제

데이터 세트 정의 및 레코드를 삭제합니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/bluesamservice/deleteDataSet`
- 인수:

  이름

  (필수, 문자열): 삭제할 데이터 세트의 이름.
- 요청 상태와 삭제된 데이터 세트를 반환합니다.

데이터 세트 레코드 개수

이 엔드포인트는 데이터 세트와 관련된 레코드 수를 반환합니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/bluesamservice/countRecords`
- 인수:

  이름

  (필수, 문자열): 데이터 세트의 이름입니다.
- 반환값: 레코드 수

대량 데이터 세트 관련 엔드포인트

다음 엔드포인트를 사용하여 여러 데이터 세트를 한 번에 만들거나 관리할 수 있습니다.

주제

데이터 세트 내보내기

- 지원되는 메서드: GET
- 경로: `/api/services/rest/bluesamservice/exportDataSet`
- 인수:

  이름

  (필수, 문자열): 삭제할 데이터 세트의 이름.

  datasetOutputFile

  (선택 사항, 문자열): 내보낸 데이터 세트를 서버에서 저장하는 경로

  rdw

  (선택 사항, 부울): RDW 필드를 익스포트해야 합니다.
- 요청 상태와 내보낸 데이터 세트를 포함하는 파일을 반환합니다.

여러 데이터 세트 생성

- 지원되는 방법: POST
- 경로: `/api/services/rest/bluesamservice/createAllDataSets`
- 인수:

- 데이터 세트의 목록

  이름

  (필수, 문자열): 데이터 세트의 이름입니다.

  유형

  (필수, 문자열): 데이터 세트 유형. 가능한 값은 ESDS, KSDS, RRDS입니다.

  recordSize

  (선택 사항, 문자열): 데이터 세트에 있는 각 레코드의 최대 크기입니다.

  fixedLength

  (선택 사항, 부울): 레코드 길이가 고정되었는지 여부를 나타냅니다.

  압축

  (선택 사항, 부울): 데이터 세트가 압축되었는지 여부를 나타냅니다.

  cacheEnable

  (선택 사항, 부울): 데이터 세트에 캐싱이 활성화되었는지 여부를 나타냅니다.
- 반환값: 요청 상태 및 새로 생성된 데이터 세트.

모든 데이터 세트를 나열합니다

- 지원되는 메서드: GET
- 경로: `/api/services/rest/bluesamservice/listDataSet`
- 인수: 없음
- 반환값: 요청 상태 및 데이터 세트 목록.

다이렉트: 모든 데이터 세트를 나열합니다

- 지원되는 메서드: GET
- 경로: `/api/services/rest/bluesamservice/directListDataSet`
- 인수: 없음
- 반환값: 요청 상태 및 데이터 세트 목록.

모든 데이터 세트를 삭제합니다

- 지원되는 방법: POST
- 경로: `/api/services/rest/bluesamservice/removeAll`
- 인수: 없음
- 반환값: 요청 상태를 나타내는 부울.

listcat 파일에서 데이터 세트 정의 가져오기

- 지원되는 방법: POST
- 경로: `/api/services/rest/bluesamservice/getDataSetsDefinitionFromListcat`
- 인수:

  paramFilePath

  (필수, 문자열): listcat 파일의 경로입니다.
- 반환값: 데이터 세트 목록

업로드된 listcat 파일에서 데이터 세트 정의 가져오기

- 지원되는 방법: POST
- 경로: `` `/api/services/rest/bluesamservice/getDataSetsDefinitionFromUploadedListcat` ``
- 인수: 없음
- 반환값: 데이터 세트 목록

json 파일에서 listcat 불러오기

- 지원되는 메서드: GET
- 경로: `/api/services/rest/bluesamservice/loadListcatFromJsonFile`
- 인수:

  filePath

  (필수, 문자열): listcat 파일의 경로입니다.
- 반환값: 데이터 세트 목록

레코드

다음 엔드포인트를 사용하여 데이터 세트 내에서 레코드를 만들거나 관리합니다.

주제

레코드 생성

이 엔드포인트는 새 레코드를 생성할 수 있도록 허용합니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/crud/createRecord`
- 인수:

  데이터 세트

  (필수, DataSet): 데이터 세트 객체

  마스크

  (필수, 마스크): 마스크 객체.
- 요청 상태와 생성된 레코드를 반환합니다.

데이터 세트 읽기

이 엔드포인트는 데이터 세트를 읽을 수 있도록 합니다.

- 지원되는 메서드: GET
- 경로: `/api/services/rest/crud/readDataSet`
- 인수:

  데이터 세트

  (필수, DataSet): 데이터 세트 개체.

- 요청 상태 및 레코드와 함께 데이터 세트를 반환합니다.

레코드 삭제

이 엔드포인트는 데이터 세트의 레코드를 삭제할 수 있습니다. 인증이 필요합니다.

- 지원되는 방법: DELETE
- 경로: `/api/services/rest/crud/deleteRecord`
- 인수:

  데이터 세트

  (필수, DataSet): 데이터 세트 개체

  레코드

  (필수, 레코드): 삭제할 레코드
- 삭제 상태를 반환합니다.

레코드 업데이트

이 엔드포인트는 데이터 세트와 관련된 레코드를 업데이트할 수 있습니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/crud/updateRecord`
- 인수:

  데이터 세트

  (필수, DataSet): 데이터 세트 개체

  레코드

  (필수, 레코드): 업데이트할 레코드
- 요청 상태 및 레코드와 함께 데이터 세트를 반환합니다.

레코드 저장

이 엔드포인트는 마스크를 사용하여 데이터 세트에 레코드를 저장할 수 있도록 합니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/crud/saveRecord`
- 인수:

  데이터 세트

  (필수, DataSet): 데이터 세트 개체

  레코드

  (필수, 레코드): 저장할 레코드
- 요청 상태 및 레코드와 함께 데이터 세트를 반환합니다.

마스크

다음 엔드포인트를 사용하여 데이터 세트에 마스크를 로드하거나 적용합니다.

주제

- 마스크 로드
- 마스킹 적용
- 마스킹 필터 적용

마스크 로드

이 엔드포인트를 사용하면 특정 데이터 세트와 관련된 모든 마스크를 검색할 수 있습니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/crud/loadMasks`
- 인수:

  데이터 세트

  (필수, DataSet): 데이터 세트 개체
- 요청 상태와 마스킹 목록을 반환합니다.

마스킹 적용

이 엔드포인트를 사용하면 특정 데이터 세트에 마스크를 적용할 수 있습니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/crud/applyMask`
- 인수:

  데이터 세트

  (필수, DataSet): 데이터 세트 개체

  마스크

  (필수, Mask): 데이터 세트 객체
- 요청 상태와 마스크가 적용된 데이터 세트를 반환합니다.

마스킹 필터 적용

이 엔드포인트를 사용하면 마스크와 필터를 특정 데이터 세트에 적용할 수 있습니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/crud/applyMaskFilter`
- 인수:

  데이터 세트

  (필수, DataSet): 데이터 세트 개체

  마스크

  (필수, Mask): 데이터 세트 객체
- 요청 상태와 마스크 및 필터가 적용된 데이터 세트를 반환합니다.

기타

다음 엔드포인트를 사용하여 데이터 세트의 캐시를 관리하거나 데이터 세트 특성을 확인할 수 있습니다

주제

- 워밍업 캐시 확인
- 체크 캐시가 활성화됨
- 캐시 활성화

- 할당된 Ram 캐시 확인
- 지속성 확인
- 지원되는 데이터 세트 유형 확인
- 상태 확인 서버

워밍업 캐시 확인

특정 데이터 세트에 대해 웜업 캐시가 활성화되어 있는지 확인합니다. 인증이 필요합니다.

- 지원되는 방법: POST
- 경로: `/api/services/rest/bluesamservice/warmupCache`
- 인수:

  이름

  (필수, 문자열): 데이터 세트의 이름입니다.
- 반환값: 워밍업 캐시가 활성화되어 있으면 true이고 그렇지 않으면 false입니다.

체크 캐시가 활성화됨

특정 데이터 세트에 대해 캐시가 활성화되었는지 확인합니다. 인증이 필요합니다.

- 지원되는 메서드: GET
- 경로: `/api/services/rest/bluesamservice/isEnableCache`
- 인수: 없음
- 캐싱이 활성화된 경우 true를 반환합니다.

캐시 활성화

- 지원되는 메서드: GET
- 경로: `/api/services/rest/bluesamservice/enableDisableCache/{enable}`
- 인수:

  enable

  (필수, 부울): true로 설정하면 캐싱이 활성화됩니다.

- 반품 없음

할당된 Ram 캐시 확인

이 엔드포인트를 통해 할당된 RAM 캐시 메모리를 검색할 수 있습니다. 인증이 필요합니다.

- 지원되는 메서드: GET
- 경로: `/api/services/rest/bluesamservice/allocatedRamCache`
- 인수: 없음
- 반환값: 문자열로 나타낸 메모리의 크기

지속성 확인

- 지원되는 메서드: GET
- 경로: `/api/services/rest/bluesamservice/persistence`
- 인수: 없음
- 반환값: 문자열로 사용된 지속성

지원되는 데이터 세트 유형 확인

- 경로: `/api/services/rest/bluesamservice/getDataSetTypes`
- 인수: 없음
- 반환값: 지원되는 데이터 세트 유형 목록을 문자열 목록으로 표시합니다.

상태 확인 서버

- 지원되는 메서드: GET
- 경로: `/api/services/rest/bluesamservice/serverIsUp`
- 인수: 없음
- 반품: 없음

사용자

다음 엔드포인트를 사용하여 사용자 상호 작용을 관리하세요.

주제

로그인

- 지원되는 방법: POST
- 경로: `/api/services/security/servicelogin/login`
- 인수:

  사용자 이름

  (필수 문자열)

  비밀번호

  (필수 문자열)
- 로그인한 사용자의 사용자 이름과 역할을 반환합니다

샘플 응답

```
{"login":"some-user","roles":[{"id":0,"roleName":"ROLE_ADMIN"}]}
```

사용자 계정 확인

- 지원되는 방법: POST
- 경로: `/api/services/security/servicelogin/hasAccount`
- 인수: 없음
- 반환값: 사용자가 이미 로그인한 경우 true

로그인

- 지원되는 방법: POST

- 경로: `/api/services/security/servicelogin/recorduser`
- 인수: 없음
- 반환값: 사용자가 이미 로그인한 경우 true

모든 사용자 나열

- 지원되는 메서드: GET
- 경로: `/api/services/security/servicelogin/listusers`
- 인수: 없음
- 반환값: 모든 사용자 목록

로그아웃

- 지원되는 방법: POST
- 경로: `/api/services/security/servicelogout/logout`
- 인수: 없음
- 반환값: 사용자가 성공적으로 로그아웃한 경우 true입니다.

JICS 애플리케이션 콘솔

JICS 구성 요소는 기존 CICS 리소스의 현대화를 위한 AWS Blu Age 지원입니다. JICS 애플리케이션 콘솔 웹 애플리케이션은 JICS 리소스 관리 전용으로 사용됩니다. 다음 엔드포인트를 사용하면 JAC 사용자 인터페이스와 상호 작용하지 않고도 관리 작업을 수행할 수 있습니다. 엔드포인트에 인증이 필요할 때마다 요청에는 인증 세부 정보(일반적으로 기본 인증에 필요한 사용자 이름/암호)가 포함되어야 합니다. JICS 애플리케이션 콘솔 웹 애플리케이션의 엔드포인트는 루트 경로를 사용합니다. `/jac/`

주제

- JICS 리소스 관리
- JAC 사용자 관리 엔드포인트

JICS 리소스 관리

다음 엔드포인트는 모두 JICS 리소스 관리와 관련이 있으므로 JICS 관리자는 매일 리소스를 처리할 수 있습니다.

주제

- TSMODEL 업데이트

JICS LISTS 및 GROUPS 나열

LIST 및 GROUPS는 JICS 구성 요소 내의 주요 소유 컨테이너 리소스입니다. 모든 JICS 리소스는 GROUP에 속해야 합니다. 그룹은 LISTS에 속할 수 있지만 필수 사항은 아닙니다. 특정 JICS 환경에서는 LISTS가 존재하지 않을 수도 있지만, 대부분의 경우 LISTS는 리소스를 위한 추가 구성 계층을 제공하기 위해 존재합니다. CICS 리소스 조직에 대한 자세한 내용은 CICS 리소스를 참조하세요.

- 지원되는 메서드: GET
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/listJicsListsAndGroups`
- 반환값: 직렬화된 JicsContainer 객체 목록 (LIST 및 GROUPS 모두) 을 JSON으로 표시합니다.

샘플 응답:


```
[
    {
      "name": "Resources",
      "children": [
        {
          "jacType": "JACList",
          "name": "MURACHS",
          "isActive": true,
          "children": [
            {
              "jacType": "JACGroup",
              "name": "MURACHS",
              "isActive": true,
              "children": []
            }
          ]
        },
        {
          "jacType": "JACGroup",
          "name": "TEST",
          "isActive": true,
          "children": []
        }
      ],
```

```
      "isExpanded": true
    }
  ]
```

JIC GROUPS 나열

- 지원되는 메서드: GET
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/listJicsGroups`
- 반환값: 직렬화된 JicsContainer 객체 (그룹) 를 JSON으로 나열한 목록입니다. GROUPS가 소유한 LIST 정보 없이 반환됩니다.

샘플 응답:

```
[
    {
      "jacType": "JACGroup",
      "name": "MURACHS",
      "isActive": true,
      "children": []
    },
    {
      "jacType": "JACGroup",
      "name": "TEST",
      "isActive": true,
      "children": []
    }
  ]
```

지정된 LIST에 대한 JICS GROUPS를 나열하세요

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/listGroupsForList`
- 인수: 우리가 찾고 있는 GROUPS가 속한 JICS LIST를 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACList` 객체의 JSON 직렬화입니다.

  샘플 요청:

```
{
    "jacType":"JACList",
    "name":"MURACHS",
    "isActive":true
  }
```

- 반환값: 지정된 LIST에 첨부된 직렬화된 JicsContainer 객체 (그룹) 를 JSON으로 나열한 목록입니다. GROUPS가 소유한 LIST 정보 없이 반환됩니다.

  샘플 응답:

```
[
    {
      "jacType": "JACGroup",
      "name": "MURACHS",
      "isActive": true,
      "children": []
    }
  ]
```

해당 GROUP의 JIC 리소스를 나열하세요

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/listResourcesForGroup`
- 인수: 우리가 찾고 있는 JICS GROUP가 속한 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACGroup` 객체의 JSON 직렬화입니다. GROUP에 대해 모든 필드를 지정할 필요가 없지만 이름은 필수입니다.

  샘플 요청:

```
{
    "jacType":"JACGroup",
    "name":"MURACHS",
    "isActive":true
  }
```

- 반환값: 지정된 GROUP이 소유한 직렬화된 JicsResource 객체 목록. 객체는 특별한 순서 없이 반환되며 유형(PROGRAM, TRANSACTION, FILE 등)이 다릅니다.

지정된 GROUP의 JIC 리소스 나열(이름을 사용하는 대체 방법)

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/listResourcesForGroupName`
- 인수: 찾고 있는 리소스를 소유한 GROUP의 이름.
- 반환값: 지정된 GROUP이 소유한 직렬화된 JicsResource 객체 목록. 객체는 특별한 순서 없이 반환되며 유형(PROGRAM, TRANSACTION, FILE 등)이 다릅니다

여러 LISTS의 소유 GROUPS 편집

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/editGroupsList`
- 인수: 하위 GROUPS가 포함된 LISTS 모음을 JSON으로 표현한 것.

  샘플 요청:

```
[
  {
        "jacType": "JACList",
        "name": "MURACHS",
        "isActive": true,
        "children": [
          {
            "jacType": "JACGroup",
            "name": "MURACHS",
            "isActive": true,
            "children": []
          },
          {
            "jacType": "JACGroup",
            "name": "TEST",
            "isActive": true,
            "children": []
          }
        ]
  }
]
```

이 편집 이전에는 “MURACHS”라는 이름의 그룹만 “MURACHS”라는 이름의 LIST에 속했습니다. 이번 편집을 통해 “TEST”라는 이름의 그룹을 “MURACHS”라는 이름의 LIST에 “추가”합니다.

- 부울 값을 반환합니다.을 반환합니다. 값이 'true'인 경우 LISTS 수정 사항이 기본 JICS 스토리지에 제대로 지속된 것입니다.

링크 삭제

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/deleteList`
- 인수: 삭제할 JIC 목록을 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACList` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 LIST 삭제가 제대로 작동한 것입니다.

GROUP 삭제

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/deleteGroup`
- 인수: 삭제할 JIC GROUP을 나타내는 JSON 페이로드. 이것은 `com.netfective.bluage.jac.entities.JACGroup` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 GROUP 삭제가 제대로 작동한 것입니다.

트랜잭션 삭제

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/deleteTransaction`
- 인수: 삭제할 JICS 트랜잭션을 나타내는 JSON 페이로드. 이것은 `com.netfective.bluage.jac.entities.JACTransaction` 객체의 JSON 직렬화입니다.

- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 TRANSACTION 삭제가 제대로 작동한 것입니다.

PROGRAM 삭제

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/deleteProgram`
- 인수: 삭제할 JICS 프로그램을 나타내는 JSON 페이로드. 이것은 `com.netfective.bluage.jac.entities.JACProgram` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 PROGRAM 삭제가 제대로 작동한 것입니다.

FILE 삭제

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/deleteFile`
- 인수: 삭제할 JIS 파일을 나타내는 JSON 페이로드. 이것은 `com.netfective.bluage.jac.entities.JACFile` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 FILE 삭제가 제대로 작동한 것입니다.

TDQUE 삭제

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/deleteTDQueue`
- 인수: 삭제할 JIC TDQUEUE를 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACTDQueue` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 TDQUEUE 삭제가 제대로 작동한 것입니다.

모델 삭제

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/deleteTSModel`
- 인수: 삭제할 JIC TSMODEL을 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACTSModel` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JIS 스토리지에서 TSMODEL 삭제가 제대로 작동한 것입니다.

LIST 만들기

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/createList`
- 인수: 생성할 JIC LIST을 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACList` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'이면 기본 JICS 저장소에 목록이 제대로 생성된 것입니다.

> **Note**
>
> LIST는 항상 빈 상태로 생성됩니다. LIST에 GROUPS을 첨부하려면 다른 작업이 필요합니다.

GROUP 만들기

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/createGroup`
- 인수: 생성할 JIC GROUP을 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACGroup` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에 GROUP이 제대로 생성된 것입니다.

> **Note**
>
> GROUP은 항상 빈 상태로 생성됩니다. RESOURCES를 GROUP에 연결하려면 추가 작업이 필요합니다(리소스를 생성하면 해당 GROUP에 리소스가 자동으로 연결됩니다).

일반 RESOURCES 생성 고려 사항

다음 엔드포인트는 모두 JICS RESOURCES 생성과 관련이 있으며 몇 가지 공통 제약 조건을 공유합니다. 엔드포인트로 전송할 요청 페이로드에서는 `groupName` 필드의 값을 지정해야 합니다.

GROUP 소유권 제약:

기존 그룹에 연결하지 않으면 리소스를 생성할 수 없으며, 엔드포인트는 groupName을 사용하여 이 리소스가 연결될 그룹을 검색합니다. `groupName`는 기존 GROUP의 이름을 가리켜야 합니다. `groupName`가 JICS 기본 스토리지의 기존 그룹을 가리키지 않는 경우 HTTP STATUS 400과 함께 오류 메시지가 전송됩니다.

GROUP 내 유니시티 제약 조건:

지정된 이름을 가진 지정된 리소스는 지정된 그룹 내에서 고유해야 합니다. 유니시티 검사는 각 리소스 생성 엔드포인트에서 수행됩니다. 지정된 페이로드가 유니시티 제약 조건을 준수하지 않는 경우 엔드포인트는 HTTP STATUS 400(BAD REQUEST)이 포함된 응답을 보냅니다. 아래 샘플 응답을 참조하세요.

샘플 페이로드: 'TEST' 그룹에 트랜잭션 'ARIT' 생성을 시도했지만 해당 이름을 가진 트랜잭션이 이 GROUP에 이미 있습니다.

```
{
    "jacType":"JACTransaction",
    "name":"ARIT",
    "groupName":"TEST",
    "isActive":true
  }
```

다음과 같은 오류 응답을 받았습니다.

```
{
    "timestamp": 1686759054510,
    "status": 400,
```

```
    "error": "Bad Request",
    "path": "/jac/api/services/rest/jicsservice/createTransaction"
  }
```

서버 로그를 검사하면 문제의 원인을 확인할 수 있습니다.

```
2023-06-14 18:10:54 default         TRACE - o.s.w.m.HandlerMethod
        - Arguments: [java.lang.IllegalArgumentException: Transaction already
 present in the group, org.springframework.security.web.header.HeaderWriterFilter
$HeaderWriterResponse@e34f6b8]
2023-06-14 18:10:54 default         ERROR - c.n.b.j.a.WebConfig                    -
 400
java.lang.IllegalArgumentException: Transaction already present in the group
 at
 com.netfective.bluage.jac.server.services.rest.impl.JicsServiceImpl.createElement(JicsServiceI
```

TRANSACTION 생성

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/createTransaction`
- 인수: 생성할 JICS TRANSACTION을 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACTransaction` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 트랜잭션이 제대로 생성된 것입니다.

PROGRAM 생성

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/createProgram`
- 인수: 생성할 JICS PROGRAM을 나타내는 JSON 페이로드. 이것은 `com.netfective.bluage.jac.entities.JACProgram` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에 PROGRAM이 제대로 생성된 것입니다.

FILE 생성

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/createFile`
- 인수: 생성할 JIS FILE을 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACFile` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 FILE이 기본 JICS 저장소에 제대로 생성된 것입니다.

TDQUE 생성

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/createTDQueue`
- 인수: 생성할 JIS TDQUEUE를 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACTDQueue` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에 TDQUEUE가 제대로 생성된 것입니다.

TSMODEL 생성

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/createTSModel`
- 인수: 생성할 JIC TSMODEL을 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACTSModel` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JIS 스토리지에 TSMODEL이 제대로 생성된 것입니다.

LIST 업데이트

- 지원되는 메서드: POST
- 인증이 필요합니다

- 경로: `/api/services/rest/jicsservice/updateList`
- 인수: 업데이트할 JIC LIST를 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACList` 객체의 JSON 직렬화입니다. LIST의 하위 요소를 제공할 필요가 없습니다. LIST 업데이트 메커니즘은 이를 고려하지 않습니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 LIST가 제대로 업데이트된 것입니다.

LIST 'isActive' 플래그를 업데이트하면 LIST의 모든 소유 요소, 즉 LIST가 소유한 모든 GROUPS와 해당 GROUPS가 소유한 모든 RESOURCES에 전파됩니다. 이는 여러 GROUPS에서 한 번의 작업으로 많은 리소스를 비활성화할 수 있는 편리한 방법입니다.

GROUP 업데이트

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/updateGroup`
- 인수: 업데이트할 JICS GROUP을 나타내는 JSON 페이로드. 이것은 `com.netfective.bluage.jac.entities.JACGroup` 객체의 JSON 직렬화입니다. GROUP의 하위 요소를 제공할 필요가 없습니다. GROUP 업데이트 메커니즘은 이를 고려하지 않습니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 GROUP은 기본 JICS 스토리지에서 제대로 업데이트된 것입니다.

> **Note**
>
> GROUP 'IsActive' 플래그를 업데이트하면 GROUP이 소유한 모든 요소, 즉 GROUP이 소유한 모든 RESOURCES에 전파됩니다. 이는 주어진 GROUP 내에서 한 번의 작업으로 많은 리소스를 비활성화할 수 있는 편리한 방법입니다.

일반 RESOURCES 업데이트 고려 사항

다음 엔드포인트는 모두 JICS RESOURCES 업데이트에 관한 것입니다. `groupName` 필드 값이 기본 JICS RESOURCE의 기존 GROUP을 가리키는 경우 필드를 사용하여 모든 JIS 리소스의 소유 GROUP을 변경할 수 있습니다. 그렇지 않으면 엔드포인트에서 잘못된 요청 응답(HTTP STATUS 400)을 받게 됩니다.

TRANSACTION 업데이트

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/updateTransaction`
- 인수: 업데이트할 JICS TRANSACTION을 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACTransaction` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 TRANSACTION이 제대로 업데이트된 것입니다.

PROGRAM 업데이트

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/updateProgram`
- 인수: 업데이트할 JICS PROGRAM을 나타내는 JSON 페이로드. 이것은 `com.netfective.bluage.jac.entities.JACProgram` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 PROGRAM이 제대로 업데이트된 것입니다.

FILE 업데이트

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/updateFile`
- 인수: 업데이트할 JIS FILE을 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACFile` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 파일이 제대로 업데이트된 것입니다.

TDQUEUE 업데이트

- 지원되는 메서드: POST
- 인증이 필요합니다

- 경로: `/api/services/rest/jicsservice/updateTDQueue`
- 인수: 업데이트할 JIC TDQUEUE를 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACTDQueue` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 TDQueue가 제대로 업데이트된 것입니다.

TSMODEL 업데이트

- 지원되는 메서드: POST
- 인증이 필요합니다
- 경로: `/api/services/rest/jicsservice/updateTSModel`
- 인수: 업데이트할 JIC TSMODEL을 나타내는 JSON 페이로드입니다. 이것은 `com.netfective.bluage.jac.entities.JACTSModel` 객체의 JSON 직렬화입니다.
- 부울 값을 반환합니다. 값이 'true'인 경우 기본 JICS 스토리지에서 TSMODEL이 제대로 업데이트된 것입니다.

JAC 사용자 관리 엔드포인트

주제

- 사용자 삭제
- 사용자로 로그인
- 시스템에 사용자가 한 명 이상 존재하는지 테스트
- 새 사용자 기록
- 사용자 표시
- 사용자 표시
- 현재 사용자 로그아웃
- Jics 서버 상태

사용자 삭제

- 지원되는 메서드: POST
- 인증 및 ADMIN 권한이 필요합니다

- 경로: `/api/services/security/servicelogin/deleteuser`
- 인수: 스토리지에서 제거될 사용자를 나타내는 `com.netfective.bluage.jac.entities.SignOn` 객체의 JSON 직렬화.
- 부울 값을 반환합니다. 값이 'true'인 경우 사용자가 JICS 시스템에서 제대로 제거된 것입니다.

> **Note**
>
> 이 작업은 취소할 수 없으며 삭제된 사용자는 JAC 애플리케이션에 다시 연결할 수 없습니다. 주의해서 사용하세요.

사용자로 로그인

- 지원되는 메서드: POST
- 인증 및 ADMIN 권한이 필요합니다
- 경로: `/api/services/security/servicelogin/login`
- 현재 요청에 자격 증명이 제공된 사용자를 나타내는 `com.netfective.bluage.jac.entities.SignOn` 객체의 JSON 직렬화를 반환합니다. 비밀번호는 반환된 객체의 뷰에서 숨겨집니다. 사용자에게 부여된 역할이 나열됩니다.

샘플 응답:

```
{
    "login": "sadmin",
    "password": null,
    "roles": [
      {
        "id": 0,
        "roleName": "ROLE_SUPER_ADMIN"
      }
    ]
  }
```

시스템에 사용자가 한 명 이상 존재하는지 테스트

- 지원되는 메서드: GET
- 경로: `/api/services/security/servicelogin/hasAccount`

- 부울 값을 반환합니다. 이 값은 기본 슈퍼 관리자 사용자가 아닌 한 명 이상의 사용자가 JICS 시스템에서 생성된 경우 true입니다.

새 사용자 기록

- 지원되는 메서드: POST
- 인증 및 ADMIN 권한이 필요합니다
- 경로: `/api/services/security/servicelogin/recorduser`
- 인수: 스토리지에 추가할 사용자를 나타내는 `com.netfective.bluage.jac.entities.SignOn` 객체의 JSON 직렬화. 사용자의 역할을 정의해야 합니다. 그렇지 않으면 사용자가 JAC 기능 및 엔드포인트를 사용하지 못할 수 있습니다.

샘플 요청:

```
{
    "login": "simpleuser",
    "password": "simpleuser",
    "roles": [
      {
        "id": 2,
        "roleName": "ROLE_USER"
      }
    ]
  }
```

새 사용자를 기록할 때는 다음 역할만 사용할 수 있습니다.

- ROLE_ADMIN: JICS 리소스 및 사용자를 관리할 수 있습니다.
- ROLE_USER: JICS 리소스는 관리할 수 있지만 사용자는 관리할 수 없습니다.

사용자 표시

- 지원되는 메서드: GET
- 인증 및 ADMIN 권한이 필요합니다
- 경로: `/api/services/security/servicelogin/listusers`
- JSON으로 직렬화된 목록을 `com.netfective.bluage.jac.entities.SignOn` 반환합니다.

사용자 표시

- 지원되는 메서드: GET
- 인증 및 ADMIN 권한이 필요합니다
- 경로: `/api/services/security/servicelogin/listusers`
- JSON으로 직렬화된 목록을 `com.netfective.bluage.jac.entities.SignOn` 반환합니다.

현재 사용자 로그아웃

- 지원되는 메서드: GET
- 경로: `/api/services/security/servicelogout/logout`
- 현재 사용자의 로그아웃에 성공하면 "{"success":true}"라는 JSON 메시지를 반환합니다(관련 HTTP 세션은 무효화됨).

Jics 서버 상태

- 지원되는 메서드: GET
- 경로: `/api/services/rest/jicsserver/serverIsUp`
- HTTP STATUS 200이라는 응답을 받으면 JICS 서버가 작동 중임을 나타냅니다.

데이터 구조

이 섹션에서는 다양한 데이터 구조에 대해 자세히 설명합니다.

주제

- Job Execution 세부 정보 메시지 구조
- 트랜잭션 시작 결과 구조
- 거래 시작 기록 결과 구조
- 큐에 있는 작업의 가능한 상태
- 작업 입력 제출
- 예정된 작업 응답 목록
- '보류 중' 작업 응답 목록

Job Execution 세부 정보 메시지 구조

각 작업 실행 세부 정보에는 다음 필드가 있습니다.

scriptId

호출된 스크립트의 식별자.

caller

호출자의 IP 주소.

식별자

고유한 작업 실행 식별자.

startTime

작업 실행이 시작된 날짜와 시간입니다.

endTime

작업 실행이 종료된 날짜와 시간입니다.

status

작업 실행에 대한 상태입니다. 다음 중 한 가지 가능한 값은 다음과 같습니다.

- `DONE`: 작업 실행이 정상적으로 종료되었습니다.
- `TRIGGERED`: 작업 실행이 트리거되었지만 아직 시작되지 않았습니다.
- `RUNNING`: 작업 실행이 실행 중입니다.
- `KILLED`: 작업 실행이 중단되었습니다.
- `FAILED`: 작업 실행에 실패했습니다.

executionResult

작업 실행에 대한 결과를 요약하는 메시지입니다. 이 메시지는 작업 실행이 아직 완료되지 않은 경우의 단순 메시지이거나 다음 필드가 포함된 JSON 구조일 수 있습니다.

- exitCode: 숫자로 구성된 종료 코드입니다. 음수 값은 실패 상황을 나타냅니다.
- 프로그램: 작업에서 실행한 최신 프로그램입니다.
- 상태: 다음 중 한 가지 가능한 값:
  - `Error`: exitCode = -1인 경우, 이는 작업 실행 중에 발생하는 (기술적) 오류에 해당합니다.
  - `Failed`: exitCode = -2 인 경우, 이는 서비스 프로그램 실행 중에 발생한 장애(예: ABEND 상황)에 해당합니다.

- Succeeded: exitCode >= 0인 경우
- stepName: 작업에서 가장 최근에 실행된 단계의 이름입니다.

executionMode

작업이 시작된 방식에 따라 동기식 또는 비동기식입니다.

샘플 출력:

```
{
    "scriptId": "INTCALC",
    "caller": "127.0.0.1",
    "identifier": "97d410be-efa7-4bd3-b7b9-d080e5769771",
    "startTime": "06-09-2023 11:42:41",
    "endTime": "06-09-2023 11:42:42",
    "status": "DONE",
    "executionResult": "{ \"exitCode\": -1, \"stepName\": \"STEP15\", \"program\":
 \"CBACT04C\", \"status\": \"Error\" }",
    "executionMode": "ASYNCHRONOUS"
  }
```

트랜잭션 시작 결과 구조

이 구조에는 다음과 같은 필드가 포함될 수 있습니다.

outCome

트랜잭션 실행 결과를 나타내는 문자열. 가능한 값은 다음과 같습니다.

- Success: 트랜잭션 실행이 제대로 끝났습니다.
- Failure: 트랜잭션 실행이 제대로 종료되지 않았습니다. 몇 가지 문제가 발생했습니다.

commarea

COMMAREA 최종 값을 바이트64로 인코딩된 바이트 배열로 나타내는 문자열. 빈 문자열일 수도 있습니다.

containerRecord

(선택 사항) 컨테이너의 레코드 내용을 바이트64로 인코딩된 바이트 배열로 나타내는 문자열.

serverDescription

요청을 처리한 서버에 대한 정보를 포함할 수 있습니다(디버깅 목적). 빈 문자열일 수도 있습니다.

abendCode

(선택 사항) 실행된 거래에서 참조한 프로그램이 승인된 경우, 승인 코드 값이 이 필드에 문자열로 반환됩니다.

샘플 응답:

Success

```
{
    "outCome": "Success",
    "commarea": "",
    "serverDescription": ""
  }
```

실패

```
{
    "outCome": "Failure",
    "commarea": "",
    "serverDescription": "",
    "abendCode": "AEIA"
  }
```

거래 시작 기록 결과 구조

이 구조에는 다음과 같은 필드가 포함될 수 있습니다.

recordContent

COMMAREA의 레코드 내용을 바이트64로 인코딩된 바이트 배열로 나타내는 문자열.

containerRecord

CONTAINER의 레코드 내용을 바이트64로 인코딩된 바이트 배열로 나타내는 문자열.

serverDescription

요청을 처리한 서버에 대한 정보를 포함할 수 있습니다(디버깅 목적). 빈 문자열일 수도 있습니다.

샘플 응답:

Success

```
{
    "recordContent": "",
    "serverDescription": ""
}
```

큐에 있는 작업의 가능한 상태

대기열에서 작업의 상태는 다음과 같습니다.

ACTIVE

작업이 현재 큐에서 실행되고 있습니다.

EXECUTION_WAIT

작업이 스레드를 사용할 수 있을 때까지 기다리고 있습니다.

SCHEDULED

작업은 특정 날짜 및 시간에 실행되도록 예약됩니다.

HOLD

작업이 실행되기 전에 릴리즈되기를 기다리고 있습니다.

완료됨

작업이 성공적으로 실행되었습니다.

FAILED

작업 실행에 실패했습니다.

UNKNOWN

상태를 알 수 없습니다.

작업 입력 제출

제출 작업 입력은 `com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage` 객체의 JSON 직렬화입니다. 아래 샘플 입력에는 해당 Bean의 모든 필드가 나와 있습니다.

샘플 입력:

```
{
```

```
    "messageQueueName":null,
    "scheduleDate":null,
    "scheduleTime":null,
    "programName":"PTA0044",
    "programParams":
     {"wmind":"B"},
    "localDataAreaValue":"",
    "userName":"USER1",
    "jobName":"PTA0044",
    "jobNumber":9,
    "jobPriority":5,
    "executionDate":"20181231",
    "jobQueue":"queue1",
    "jobOnHold":false
}
```

jobNumber

jobNumber가 0인 경우 작업 번호 시퀀스의 다음 번호를 사용하여 작업 번호가 자동으로 생성됩니다. 이 값은 0으로 설정해야 합니다(테스트용 제외).

jobPriority

AS400 기본 작업 우선 순위는 5입니다. 유효 범위는 0-9이며, 0이 가장 높은 우선 순위입니다.

jobOnHold

작업이 보류 상태로 제출되면 즉시 실행되지 않고 누군가 작업을 “해제”할 때만 실행됩니다. REST API(/release 또는 /release-all) 를 사용하여 작업을 릴리스할 수 있습니다.

scheduleDate 및 scheduleTime

이 값들이 null이 아닌 경우 작업은 지정된 날짜 및 시간에 실행됩니다.

날짜

MMddyy 또는 ddMMyyyy 형식으로 제공할 수 있습니다(입력 크기에 따라 사용되는 형식이 결정됨).

Time

HHmm 또는 HHmmss 형식으로 제공 가능(입력 크기에 따라 사용되는 형식이 결정됨)

programParams

이것은 프로그램에 맵으로 전달됩니다.

예정된 작업 응답 목록

목록 작업 큐 엔드포인트의 구조입니다. 해당 작업을 제출하는 데 사용된 작업 제출 메시지는 응답의 일부라는 점에 유의하세요. 추적 또는 테스트/재제출 목적으로 사용할 수 있습니다. 작업이 완료되면 시작 날짜와 종료 날짜도 입력됩니다.


```
[
    {
      "qrtzJobGroup": "PTA0044-PTA0044",
      "qrtzJobName": "PTA0044-168156109957800",
      "status": "HOLD",
      "qrtzDelay": 0,
      "startDate": null,
      "endDate": null,
      "jobName": "PTA0044",
      "userName": "USER1",
      "jobNumber": 9,
      "jobPriority": 5,
      "jobQueue": "queue1",
      "message": {
        "messageQueueName": null,
        "scheduleDate": null,
        "scheduleTime": null,
        "programName": "PTA0044",
        "programParams": {
          "wmind": "B"
        },
        "localDataAreaValue": "",
        "userName": "USER1",
        "jobName": "PTA0044",
        "jobNumber": 9,
        "jobPriority": 5,
        "executionDate": "20181231",
        "jobQueue": "queue1",
        "jobOnHold": true
      }
    },
    {
      "qrtzJobGroup": "PTA0044-PTA0044",
      "qrtzJobName": "PTA0044-166196954877600",
      "status": "COMPLETED",
      "qrtzDelay": 0,
      "startDate": "2022-10-13T22:48:34.025+00:00",
```

```
      "endDate": "2022-10-13T22:52:54.475+00:00",
      "jobName": "PTA0044",
      "userName": "USER1",
      "jobNumber": 9,
      "jobPriority": 5,
      "jobQueue": "queue1",
      "message": {
        "messageQueueName": null,
        "scheduleDate": null,
        "scheduleTime": null,
        "programName": "PTA0044",
        "programParams": {
          "wmind": "B"
        },
        "localDataAreaValue": "",
        "userName": "USER1",
        "jobName": "PTA0044",
        "jobNumber": 9,
        "jobPriority": 5,
        "executionDate": "20181231",
        "jobQueue": "queue1",
        "jobOnHold": true
      }
    }
  ]
```

'보류 중' 작업 응답 목록

반환된 모든 작업이 “보류 중”이라는 점을 제외하면 구조는 이전 작업과 비슷합니다.

```
[
    {
      "qrtzJobGroup": "PTA0044-PTA0044",
      "qrtzJobName": "PTA0044-168156109957800",
      "status": "HOLD",
      "qrtzDelay": 0,
      "startDate": null,
      "endDate": null,
      "jobName": "PTA0044",
      "userName": "USER1",
      "jobNumber": 9,
      "jobPriority": 5,
```

```
      "jobQueue": "queue1",
      "message": {
        "messageQueueName": null,
        "scheduleDate": null,
        "scheduleTime": null,
        "programName": "PTA0044",
        "programParams": {
          "wmind": "B"
        },
        "localDataAreaValue": "",
        "userName": "USER1",
        "jobName": "PTA0044",
        "jobNumber": 9,
        "jobPriority": 5,
        "executionDate": "20181231",
        "jobQueue": "queue1",
        "jobOnHold": true
      }
    }
  ]
```

AWS 블루 에이지 런타임 (비관리형) 설치

이 섹션에서는 인프라에서 AWS Blu Age 런타임 (비관리형) 을 설정하는 단계를 설명합니다. AWS

주제

- AWS 블루 에이지 런타임 사전 요구 사항
- AWS 블루에이지 런타임 온보딩
- AWS Blu Age 런타임을 위한 인프라 설정 요구 사항 (비관리형)
- AWS Amazon ECS에서의 블루에이지 런타임 배포는 에서 관리합니다. AWS Fargate
- AWS Amazon EC2에서의 블루에이지 런타임 배포

AWS 블루 에이지 런타임 사전 요구 사항

AWS 블루에이지 런타임 (비관리형) 은 여러 릴리스 버전에서 사용할 수 있습니다. 현대화 프로젝트가 진행 중인 경우 구현 및 테스트 목적으로 런타임의 증분 버전이 필요할 수 있습니다. 요구 사항을 정의하려면 AWS Blu Age 제공 관리자에게 문의하십시오.

AWS Blu Age 런타임 (비관리형) 온보딩 프로세스를 시작하기 전에 다음을 수행하십시오.

- 계정이 있는지 확인하세요. AWS
- Blu Age로 리팩토링된 현대화된 애플리케이션이 있는지 확인하세요. AWS
- AWS 지역과 컴퓨팅 (Amazon EC2 또는 Amazon ECS 사용) 을 선택합니다. AWS Fargate
- 사용하려는 AWS 블루에이지 런타임 버전을 선택합니다.
- AWS Blu Age 런타임 (비관리형) 을 실행하는 데 필요한 추가 구성 요소를 검토하고 the section called “인프라 설정 요구 사항” 검증하십시오.

> **Note**
>
> AWS 블루 에이지 런타임 (비관리형) 의 기능을 테스트하려면 -v1.zip 에서 다운로드할 수 있는 데모 애플리케이션을 `Planets Demo` 사용할 수 있습니다. PlanetsDemo

AWS 블루에이지 런타임 온보딩

시작하려면 AWS Support 케이스를 만들어 AWS 블루에이지 런타임에 액세스하기 위한 온보딩을 요청하세요. 요청서에 AWS 계정 ID, 사용하려는 AWS 지역, 컴퓨팅 선택 및 런타임 버전을 포함하십시오. 어떤 버전이 필요한지 확실하지 않은 경우 AWS Blu Age 배송 관리자에게 문의하십시오.

AWS Amazon EC2의 블루 에이지 런타임 (비관리형)

AWS Blu Age Runtime (비관리형) 아티팩트를 지역 및 컴퓨팅 선택에 따라 다양한 Amazon S3 버킷에 저장합니다. Amazon EC2의 AWS Blu Age 런타임 (비관리형) AWS 리전 용 버킷에 액세스하려면 다음 표에 나열된 이름을 사용하십시오.

AWS 리전	버킷 릴리스	버킷 빌드
미국 동부(오하이오)	aws-bluage-runtime-artifacts-055777665268-us-east-2	aws-bluage-runtime-artifacts-dev-055777665268-us-east-2
미국 동부(버지니아 북부)	aws-bluage-runtime-artifacts-139023371234-us-east-1	aws-bluage-runtime-artifacts-dev-139023371234-us-east-1
미국 서부(캘리포니아 북부)	aws-bluage-runtime-artifacts-788454048782-us-west-1	aws-bluage-runtime-artifacts-dev-788454048782-us-west-1

AWS 리전	버킷 릴리스	버킷 빌드
미국 서부(오리건)	aws-bluage-runtime-artifacts-836771190483-us-west-2	aws-bluage-runtime-artifacts-dev-836771190483-us-west-2
유럽(아일랜드)	aws-bluage-runtime-artifacts-925278190477-eu-west-1	aws-bluage-runtime-artifacts-dev-925278190477-eu-west-1
유럽(파리)	aws-bluage-runtime-artifacts-673009995881-eu-west-3	aws-bluage-runtime-artifacts-dev-673009995881-eu-west-3
유럽(프랑크푸르트)	aws-bluage-runtime-artifacts-485196800481-eu-central-1	aws-bluage-runtime-artifacts-dev-485196800481-eu-central-1
남아메리카(상파울루)	aws-bluage-runtime-artifacts-737536804457-sa-east-1	aws-bluage-runtime-artifacts-dev-737536804457-sa-east-1
아시아 태평양(도쿄)	aws-bluage-runtime-artifacts-445578176276-ap-노스이스트-1	aws-bluage-runtime-artifacts-dev-445578176276-ap-노스이스트-1
아시아 태평양(시드니)	aws-bluage-runtime-artifacts-726160321909-ap-사우스이스트-2	aws-bluage-runtime-artifacts-dev-726160321909-ap-사우스이스트-2

AWS Fargate에서 관리하는 아마존 ECS의 블루 에이지 런타임 (비관리형)

AWS Blu Age Runtime (비관리형) 아티팩트를 지역 및 컴퓨팅 선택에 따라 다양한 Amazon S3 버킷에 저장합니다. Fargate에서 관리하는 Amazon ECS의 AWS 블루 에이지 런타임 (비관리형) AWS 리전 용 버킷에 액세스하려면 다음 표에 나열된 이름을 사용하십시오.

AWS 리전	버킷 릴리스	버킷 빌드
미국 동부(오하이오)	aws-bluage-runtime-fargate-rel-483416914331-us-east-2	aws-bluage-runtime-fargate-dev-483416914331-us-east-2

AWS 리전	버킷 릴리스	버킷 빌드
미국 동부(버지니아 북부)	aws-bluage-runtime-fargate-rel-308472162679-us-east-1	aws-bluage-runtime-fargate-dev-308472162679-us-east-1
미국 서부(캘리포니아 북부)	aws-bluage-runtime-fargate-rel-343763094578-us-west-1	aws-bluage-runtime-fargate-dev-343763094578-us-west-1
미국 서부(오리건)	aws-bluage-runtime-fargate-rel-688933007849-us-west-2	aws-bluage-runtime-fargate-dev-688933007849-us-west-2
유럽(아일랜드)	aws-bluage-runtime-fargate-rel-140138033705-eu-west-1	aws-bluage-runtime-fargate-dev-140138033705-eu-west-1
유럽(파리)	aws-bluage-runtime-fargate-rel-339712948211-eu-west-3	aws-bluage-runtime-fargate-dev-339712948211-eu-west-3
유럽(프랑크푸르트)	aws-bluage-runtime-fargate-rel-339712918892-eu-central-1	aws-bluage-runtime-fargate-dev-339712918892-eu-central-1
남아메리카(상파울루)	aws-bluage-runtime-fargate-rel-767397998881-sa-east-1	aws-bluage-runtime-fargate-dev-767397998881-sa-east-1
아시아 태평양(도쿄)	aws-bluage-runtime-fargate-rel-891377400849-ap-노스이스트-1	aws-bluage-runtime-fargate-dev-891377400849-ap-노스이스트-1
아시아 태평양(시드니)	aws-bluage-runtime-fargate-rel-533267435478-ap-사우스이스트-2	aws-bluage-runtime-fargate-dev-533267435478-ap-사우스이스트-2

명령줄을 사용하여 버킷의 콘텐츠를 나열합니다

온보딩된 후 터미널에서 다음 명령을 실행하여 나열할 수 있습니다.

```
aws s3 ls bucket-name
```

이전 표에 나와 있는 버킷 이름으로 바꾸십시오. `bucket-name` AWS 리전

이 명령은 다음과 같은 다양한 버전의 AWS Blu Age 런타임 (비관리형) 런타임에 해당하는 폴더 목록을 반환합니다.

```
PRE 3.3.0.1/
PRE 3.3.0.2/
```

지원되는 최신 버전을 사용하는 것이 좋습니다. 그렇게 할 수 없다면 애플리케이션 리팩토링 단계에서 검증된 런타임 버전을 사용하십시오. 특정 버전에 사용 가능한 프레임워크를 나열하려면 다음 명령을 실행합니다.

`aws s3 ls s3://bucket-name/version/Framework/`

사용 중인 버킷 이름과 원하는 버전으로 bucket-name 바꾸십시오 AWS 리전 . version 다음은 예입니다.

`aws s3 ls s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/Framework/`

이 명령은 다음과 같은 프레임워크 목록을 반환합니다.

```
2023-06-05 10:26:52   97960225 aws-bluage-runtime-3.4.0.tar.gz
2023-06-05 10:27:12         45 aws-bluage-runtime-3.4.0.tar.gz.checksumSHA256
2023-06-05 10:27:14  138497123 aws-bluage-webapps-3.4.0.tar.gz
2023-06-05 10:27:44         45 aws-bluage-webapps-3.4.0.tar.gz.checksumSHA256
```

프레임워크 다운로드

예를 들어 프레임워크를 다운로드하여 기존 Amazon EC2 인스턴스의 속도 런타임 버전을 업그레이드할 수 있습니다.

`aws s3 cp s3://bucket-name/version/Framework/ folder-of-your-choice --recursive`

위치:

folder-of-your-choice

프레임워크를 다운로드하려는 폴더 경로.

예: `aws s3 cp s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/Framework/ . --recursive`

이 명령으로 다음 출력이 생성됩니다.

```
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/
Framework/aws-bluage-runtime-3.4.0.tar.gz.checksumSHA256 to ./aws-bluage-
runtime-3.4.0.tar.gz.checksumSHA256
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/
Framework/aws-bluage-webapps-3.4.0.tar.gz.checksumSHA256 to ./aws-bluage-
webapps-3.4.0.tar.gz.checksumSHA256
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/Framework/aws-
bluage-webapps-3.4.0.tar.gz to ./aws-bluage-webapps-3.4.0.tar.gz
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/3.4.0/Framework/aws-
bluage-runtime-3.4.0.tar.gz to ./aws-bluage-runtime-3.4.0.tar.gz
```

다음과 같이 프레임워크 파일을 나열할 수 있습니다.

```
ls -l
```

이 명령으로 다음 출력이 생성됩니다.

```
total 230928
-rw-rw-r-- 1 cloudshell-user cloudshell-user  97960225 Jun  5 10:26 aws-bluage-
runtime-3.4.0.tar.gz
-rw-rw-r-- 1 cloudshell-user cloudshell-user        45 Jun  5 10:27 aws-bluage-
runtime-3.4.0.tar.gz.checksumSHA256
-rw-rw-r-- 1 cloudshell-user cloudshell-user 138497123 Jun  5 10:27 aws-bluage-
webapps-3.4.0.tar.gz
-rw-rw-r-- 1 cloudshell-user cloudshell-user        45 Jun  5 10:27 aws-bluage-
webapps-3.4.0.tar.gz.checksumSHA256
```

AWS Blu Age 런타임을 위한 인프라 설정 요구 사항 (비관리형)

이 항목에서는 AWS Blu Age 런타임 (비관리형) 을 실행하는 데 필요한 최소 인프라 구성에 대해 설명합니다. 다음 절차는 선택한 컴퓨터에 AWS Blu Age Runtime (비관리형) 을 설정하여 Blu Age Runtime에 현대화된 애플리케이션을 배포하는 방법을 설명합니다. AWS 생성하는 리소스는 애플리케이션 도메인 전용 서브넷이 있는 Amazon VPC에 있어야 합니다.

보안 그룹 생성

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 여세요.
2. 왼쪽 탐색 창의 보안에서 보안 그룹을 선택합니다.
3. 가운데 창에서 보안 그룹 생성을 선택합니다.

4. 보안 그룹 이름 필드에 **M2BluagePrivateLink-SG**를 입력합니다.
5. 인바운드 규칙 섹션에서 규칙 추가를 선택합니다.
6. 유형에서 HTTPS를 선택합니다.
7. 소스에 VPC CIDR을 입력합니다.
8. 아웃바운드 규칙 섹션에서 규칙 추가를 선택합니다.
9. 유형에서 HTTPS를 선택합니다.
10. 대상 주소(Destination)에 **0.0.0.0/0**을 입력합니다.
11. 보안 그룹 생성을 선택합니다.

Amazon VPC 엔드포인트 생성

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 여세요.
2. 왼쪽 탐색 창의 Virtual Private Cloud에서 엔드포인트를 선택합니다.
3. 가운데 창에서 엔드포인트 생성을 선택합니다.
4. 서비스 섹션에서 검색 필드에 을 입력한 **SQS** 다음 해당 지역에 해당하는 Amazon SQS 서비스를 선택합니다.
5. VPC 섹션에서 이전 단계에서 만든 Amazon VPC를 선택합니다.
6. 서브넷 섹션에서 애플리케이션 도메인용으로 생성한 서브넷을 선택합니다.
7. 보안 그룹 섹션에서 M2 BluagePrivateLink -SG를 선택합니다.
8. Create endpoint(엔드포인트 생성)을 선택합니다.

IAM 정책 생성

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창의 액세스 관리에서 정책을 선택합니다.
3. 가운데 창에서 정책 생성을 선택합니다.
4. 정책 편집기 섹션에서 JSON을 선택합니다.
5. 편집기에 표시되는 모든 JSON을 다음 JSON으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
```

```
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 정책을 사용자 지정하는 데 필요한 추가 세부 정보가 필요한 경우 AWS Blu Age 배송 관리자 또는 계정 관리자에게 문의하십시오.

6. 다음을 선택합니다.
7. 정책 이름을 입력한 후 정책 생성을 선택합니다.

IAM 역할 생성

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창의 액세스 관리에서 역할을 선택합니다.
3. 가운데 창에서 역할 생성을 선택합니다.
4. 사용 사례 섹션에서 컴퓨팅 선택에 따라 EC2 또는 Elastic Container Service (그리고 Elastic Container Service Task) 를 선택합니다.
5. 다음을 선택합니다.
6. 검색 상자에 이전에 생성한 정책의 이름을 입력합니다.
7. 정책 왼쪽에 있는 확인란을 선택합니다.
8. 다음을 선택합니다.
9. 역할 이름을 입력한 다음 Create role(역할 생성)을 선택합니다.

Amazon EC2에서 AWS 블루 에이지 런타임 실행

Amazon EC2 인스턴스 생성

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

2. 인스턴스 시작을 선택합니다.
3. 인스턴스 유형으로는 에 나열된 유형 중 하나를 선택합니다. the section called “AWS 블루 에이지 런타임용 Amazon EC2 인스턴스 유형 (Amazon EC2)”
4. 키 페어 섹션에서 기존 키 페어를 선택하거나 새로 하나를 생성합니다.
5. 네트워크 설정 섹션에서 기존 보안 그룹 선택을 선택합니다.
6. 일반 보안 그룹의 경우 M2 BluagePrivateLink -SG를 선택합니다.
7. 고급 세부 정보 섹션을 확장합니다.
8. IAM 인스턴스 프로파일에서 앞서 생성한 IAM 역할을 선택합니다.
9. 인스턴스 시작을 선택합니다.

Amazon EC2 인스턴스의 상태가 실행 중으로 변경되면 인스턴스에 연결하고 다음 소프트웨어 구성 요소를 설치합니다.

- Java Runtime Environment(JRE) 11.
- Apache Tomcat 9.
- AWS 블루 에이지 런타임 (Amazon EC2). Apache Tomcat 설치 폴더의 루트에 AWS Blu Age 런타임을 설치합니다 (일부 파일은 추가되지만 다른 파일은 덮어쓰여집니다).

추가 웹앱은 별도의 폴더에 설치하세요. 이 경우 Apache Tomcat을 설치한 다음, 그 위에 아카이브를 설치하는 프로세스를 반복합니다.

Amazon ECS에서 AWS 블루 에이지 런타임을 실행하는 사람이 관리합니다. AWS Fargate

Amazon ECS 클러스터와 작업 역할을 생성하여 나중에 AWS Fargate 작업을 시작할 때 사용합니다. 작업 역할에는 이전에 생성한 정책을 포함하십시오. 시나리오에 따라 작업 역할에 추가 IAM 정책을 추가해야 할 수도 있습니다.

AWS 블루 에이지 런타임용 Amazon EC2 인스턴스 유형 (Amazon EC2)

다음은 AWS 블루 에이지 런타임 (Amazon EC2) 에 사용할 수 있는 Amazon EC2 인스턴스 유형 목록입니다.

```
t3.xlarge
t3.small
t3.large
```

```
t2.small
t2.large
r6i.xlarge
r6i.large
r6i.4xlarge
r6i.2xlarge
r5b.xlarge
r5b.large
r5b.2xlarge
r3.xlarge
m6i.xlarge
m6i.large
m6i.8xlarge
m6i.4xlarge
m6i.2xlarge
m6i.16xlarge
m5zn.xlarge
m5zn.large
m5zn.3xlarge
m5zn.2xlarge
m5.xlarge
m5.large
m5.8xlarge
m5.4xlarge
m5.2xlarge
m5.16xlarge
m5.12xlarge
c6i.xlarge
c6i.large
c6i.8xlarge
c6i.4xlarge
c6i.2xlarge
c6i.16xlarge
c5.xlarge
c5.large
c5.9xlarge
c5.4xlarge
c5.2xlarge
c5.18xlarge
c5.12xlarge
```

AWS Amazon ECS에서의 블루에이지 런타임 배포는 에서 관리합니다. AWS Fargate

이 섹션의 항목에서는 Managed Amazon ECS에서 AWS Blu Age Runtime을 설정하는 방법 AWS Fargate, 런타임 버전을 업데이트하는 방법, Amazon CloudWatch 경보를 사용하여 배포를 모니터링하는 방법, 라이선스가 부여된 종속성을 추가하는 방법을 설명합니다.

주제

- Amazon ECS에서 AWS 블루 에이지 런타임 설정 관리 AWS Fargate
- 에서 관리하는 Amazon ECS의 AWS 블루 에이지 런타임 업그레이드 AWS Fargate
- Amazon ECS의 AWS 블루 에이지 런타임을 위한 Amazon CloudWatch 알람은 에서 관리합니다. AWS Fargate
- Amazon ECS의 AWS Blu Age 런타임에서 라이선스가 부여된 종속성 설정은 에서 관리합니다. AWS Fargate

Amazon ECS에서 AWS 블루 에이지 런타임 설정 관리 AWS Fargate

이 주제에서는 에서 관리하는 Amazon ECS의 AWS Blu Age Runtime을 사용하여 PlanetsDemo 샘플 애플리케이션을 설정하고 배포하는 방법을 설명합니다. AWS Fargate

AWS 에서 관리하는 Amazon ECS의 블루에이지 런타임을 AWS Fargate Linux/X86에서 사용할 수 있습니다.

주제

- 필수 조건
- 설정
- PlanetsDemo 애플리케이션을 테스트합니다.

필수 조건

시작하기 전에 다음 사전 조건을 충족하는지 확인합니다.

- AWS CLI 구성의 단계에 AWS CLI 따라 구성합니다.
- the section called “AWS 블루 에이지 런타임 사전 요구 사항” 및 the section called “AWS 블루에이지 런타임 온보딩”를 완료합니다.

- 바이너리로 AWS Fargate 관리되는 Amazon ECS에서 AWS 블루에이지 런타임을 다운로드하십시오. 지침은 the section called “AWS 블루에이지 런타임 온보딩” 섹션을 참조하십시오.
- 아파치 톰캣 9 바이너리를 다운로드하십시오.
- 애플리케이션 아카이브를 다운로드하십시오. PlanetsDemo
- JICS용 Amazon Aurora PostgreSQL 데이터베이스를 생성하고 해당 데이터베이스에서 쿼리를 실행합니다. `PlanetsDemo-v1/jics/sql/initJics.sql` Amazon Aurora PostgreSQL 데이터베이스를 생성하는 방법에 대한 자세한 내용은 Aurora PostgreSQL DB 클러스터 생성 및 연결을 참조하십시오.

설정

샘플 애플리케이션을 설정하려면 다음 단계를 완료하십시오. PlanetsDemo

1. Apache Tomcat 바이너리를 다운로드한 후 내용을 추출하고 폴더로 이동합니다. `conf` 편집할 `catalina.properties` 파일을 열고 다음 `common.loader` 줄로 시작하는 줄을 바꿉니다.

```
common.loader="${catalina.base}/lib","${catalina.base}/lib/
*.jar","${catalina.home}/lib","${catalina.home}/lib/*.jar","${catalina.home}/
shared","${catalina.home}/shared/*.jar","${catalina.home}/extra","${catalina.home}/
extra/*.jar"
```

2. tar 명령을 사용하여 아파치 톰캣 폴더를 압축하여 `tar.gz` 아카이브를 빌드합니다.
3. Dockerfile을 준비하여 제공된 런타임 바이너리와 Apache Tomcat 서버 바이너리를 기반으로 사용자 지정 이미지를 빌드하세요. 다음 예제 Dockerfile을 참조하십시오. 목표는 Apache Tomcat 9를 설치한 다음 Apache Tomcat 9 설치 디렉토리의 루트에서 추출한 AWS Blu Age Runtime (Amazon ECS용 AWS Fargate) 을 설치한 다음, 라는 이름의 현대화된 샘플 애플리케이션을 설치하는 것입니다. PlanetsDemo

> **Note**
>
> 이 예제 Dockerfile에서 사용되는 install-gapwalk.sh 및 install-app.sh 스크립트의 내용은 도커파일 뒤에 나열되어 있습니다.

```
FROM --platform=linux/x86_64 amazonlinux:2

RUN mkdir -p /workdir/apps
WORKDIR /workdir
```

```
COPY install-gapwalk.sh .
COPY install-app.sh .
RUN chmod +x install-gapwalk.sh
RUN chmod +x install-app.sh

# Install Java and AWS CLI v2-y
RUN yum install sudo java-17-amazon-corretto unzip tar -y
RUN sudo yum remove awscli -y
RUN curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
 "awscliv2.zip"
RUN sudo unzip awscliv2.zip
RUN sudo ./aws/install

#·Installation dir
RUN mkdir -p /usr/local/velocity/installation/gapwalk
# Copy PlanetsDemo archive to a dedicated apps dir
COPY PlanetsDemo-v1.zip /workdir/apps/

# Copy resources (tomcat, blu age runtime) to installation dir
COPY tomcat.tar.gz /usr/local/velocity/installation/tomcat.tar.gz
COPY aws-bluage-on-fargate-runtime-3.10.0.15.tar.gz /usr/local/velocity/
installation/gapwalk/gapwalk-bluage-on-fargate.tar.gz

# run relevant installation scripts
RUN ./install-gapwalk.sh
RUN ./install-app.sh

EXPOSE 8080
EXPOSE 8081
# ...

# Run Command to start Tomcat server
CMD ["sh", "-c", "sudo /bluage-on-fargate/tomcat.gapwalk/velocity/startup.sh
 $ECS_CONTAINER_METADATA_URI_V4 $AWS_CONTAINER_CREDENTIALS_RELATIVE_URI"]
```

install-gapwalk.sh 내용은 다음과 같습니다.

```
#!/bin/sh

# Vars
TEMP_DIR=/bluage-on-fargate/tomcat.gapwalk/temp

# Install
```

```
echo "Installing Gapwalk and Tomcat"
sudo rm -rf /bluage-on-fargate
mkdir -p ${TEMP_DIR}
# Copy Blu Age runtime and tomcat archives to temporary extraction dir
sudo cp /usr/local/velocity/installation/gapwalk/gapwalk-bluage-on-fargate.tar.gz
 ${TEMP_DIR}
sudo cp /usr/local/velocity/installation/tomcat.tar.gz ${TEMP_DIR}
#·Create velocity dir
mkdir -p /bluage-on-fargate/tomcat.gapwalk/velocity
#·Extract tomcat files
tar -xvf ${TEMP_DIR}/tomcat.tar.gz -C ${TEMP_DIR}
# Copy all tomcat files to velocity dir
cp -fr ${TEMP_DIR}/apache-tomcat-9.x.x/* /bluage-on-fargate/tomcat.gapwalk/velocity
# Remove default webapps of Tomcat
rm -f /bluage-on-fargate/tomcat.gapwalk/velocity/webapps/*
# Extract Blu Age runtime at velocity dir
tar -xvf ${TEMP_DIR}/gapwalk-bluage-on-fargate.tar.gz -C /bluage-on-fargate/
tomcat.gapwalk
# Remove temporary extraction dir
sudo rm -rf ${TEMP_DIR}
```

install-app.sh 내용은 다음과 같습니다.

```
#!/bin/sh

APP_DIR=/workdir/apps
TOMCAT_GAPWALK_DIR=/bluage-on-fargate/tomcat.gapwalk

unzip ${APP_DIR}/PlanetsDemo-v1.zip -d ${APP_DIR}
cp -r ${APP_DIR}/webapps/* ${TOMCAT_GAPWALK_DIR}/velocity/webapps/
cp -r ${APP_DIR}/config/* ${TOMCAT_GAPWALK_DIR}/velocity/config/
```

4. 폴더에 있는 `application-main.yml` 파일의 다음 스니펫에 사전 요구 사항의 일부로 만든 데이터베이스의 연결 정보를 제공하십시오. `{TOMCAT_GAPWALK_DIR}/config` 자세한 내용은 Aurora PostgreSQL DB 클러스터를 참조하세요.

```
datasource:
   jicsDs:
     driver-class-name :
     url:
     username:
     password:
```

```
      type :
```

5. 이미지를 빌드하여 Amazon ECR 리포지토리로 푸시합니다. 지침은 Amazon Elastic 컨테이너 레지스트리 사용 설명서의 Docker 이미지 푸시를 참조하십시오.
6. https://console.aws.amazon.com/ecs/v2에서 콘솔을 엽니다.
7. 탐색 창에서 태스크 정의를 선택합니다.
8. 시작 유형에서 선택하십시오. AWS Fargate
9. 일부로 생성한 작업 역할을 선택합니다the section called “인프라 설정 요구 사항”.
10. 이미지를 컨테이너에 첨부합니다.
11. 양식 작성을 마친 다음 [Create] 를 선택합니다.
12. 왼쪽 탐색 창에서 클러스터를 선택한 다음 목록에서 클러스터를 선택합니다.
13. 클러스터의 세부 정보 페이지에 있는 [서비스] 탭에서 [Create] 를 선택합니다.
14. 작업 정의를 선택합니다.
15. 네트워킹 섹션을 확장하고 구성원으로 만든 VPC, 서브넷, 보안 그룹을 구성합니다. the section called “인프라 설정 요구 사항”
16. Amazon ECS 서비스를 배포하십시오.

배포가 실패할 경우 로그를 확인하십시오. 이를 찾으려면 Amazon ECS Managed by AWS Fargate: 의 작업 페이지로 이동한 다음 로그 탭을 선택합니다. C로 시작하고 숫자가 오는 오류 코드 (예: CXXXX) 를 발견하면 오류 메시지를 기록해 두십시오. 예를 들어, 오류 코드 C5102 오류는 잘못된 인프라 구성을 나타내는 일반적인 오류입니다. 또한 실행 중인 작업 내부를 탐색하고 AWS Blu Age Runtime (Amazon EC2의 경우) 과 유사한 몇 가지 명령을 실행할 수 있습니다. 자세한 내용은 Amazon Elastic 컨테이너 서비스 개발자 안내서의 Amazon ECS Exec 디버깅을 위한 사용을 참조하십시오.

대화형 셸을 열려면 로컬 시스템에서 다음 명령을 실행하십시오.

```
aws ecs execute-command --cluster your_cluster_name --container your_container_name --
task task_id --interactive --command /bin/sh
```

PlanetsDemo 애플리케이션을 테스트합니다.

배포된 PlanetsDemo 애플리케이션의 상태를 확인하려면 `load-balancer-DNS-namelistener-port`, 교체한 후 설정에 맞는 올바른 값을 `web-binary-name` 사용하여 다음 명령을 실행합니다.

```
curl http://load-balancer-DNS-name:listener-port/gapwalk-application/
```

애플리케이션이 실행 중인 경우 다음 출력 메시지가 표시됩니다`Jics application is running`.

그런 다음 다음 명령을 실행합니다.

```
curl http://load-balancer-DNS-name:listener-port/jac/api/services/rest/jicsservice/
```

애플리케이션이 실행 중인 경우 다음과 같은 출력 메시지가 표시됩니다`Jics application is running`.

```
Jics application is running
```

Blusam을 구성한 경우 다음 명령을 실행하면 빈 응답이 표시될 수 있습니다.

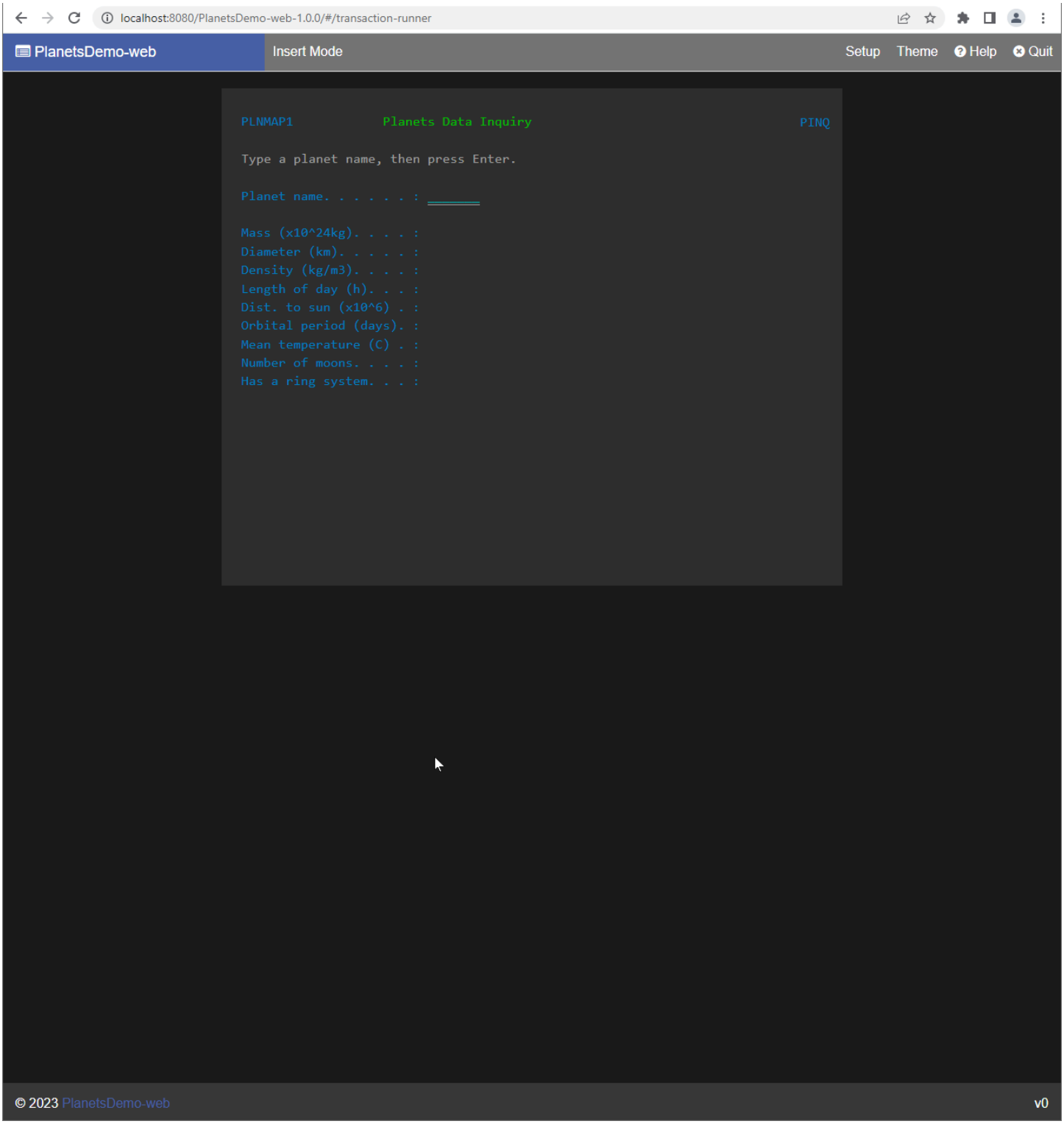
```
curl http://load-balancer-DNS-name:listener-port/bac/api/services/rest/bluesamserver/
serverIsUp
```

웹 바이너리의 이름을 기록해 둡니다 (변경되지 않은 경우 PlanetsDemo -web-1.0.0). PlanetsDemo애플리케이션에 액세스하려면 다음 형식의 URL을 사용합니다.

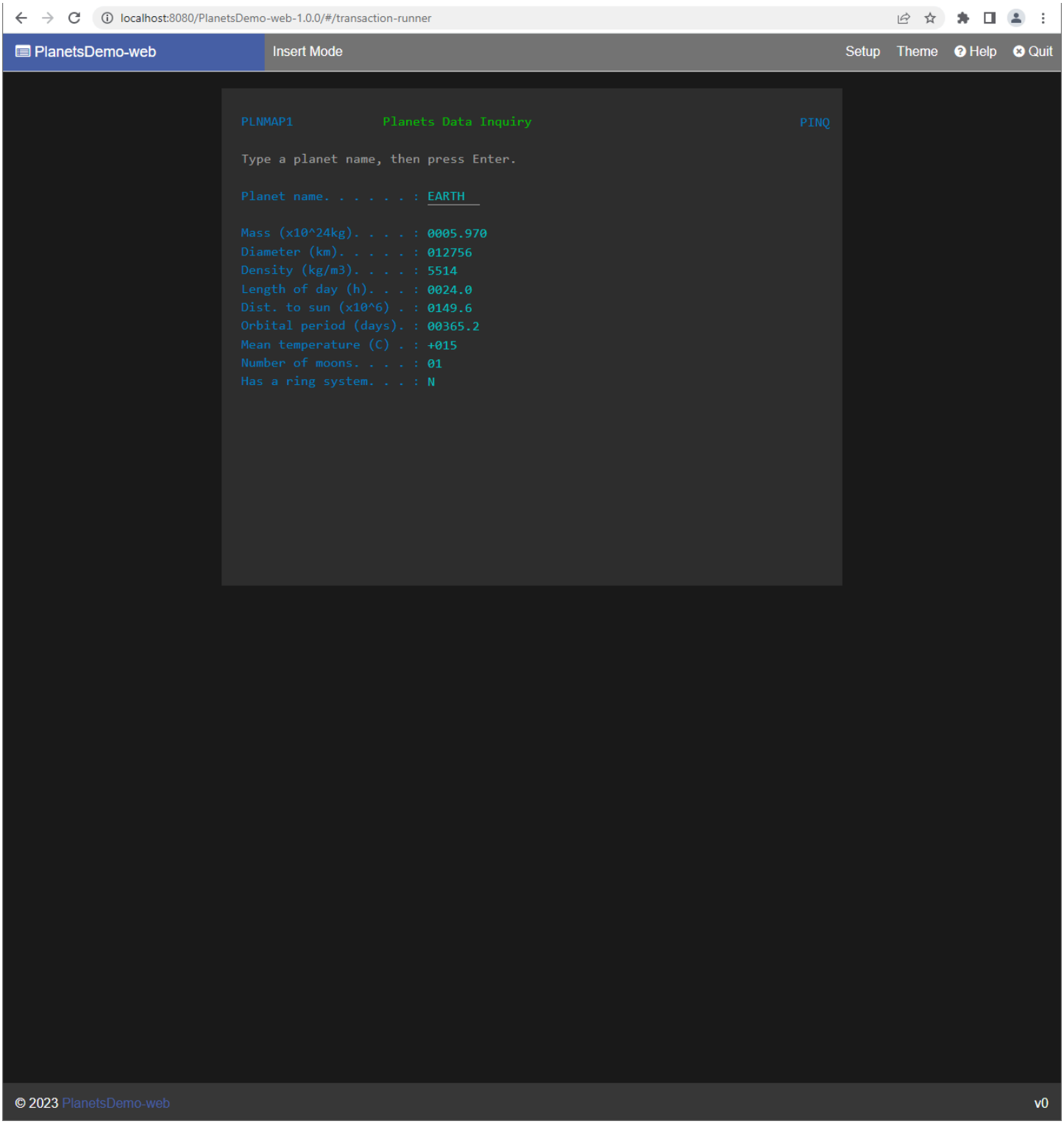
```
https://load-balancer-DNS-name:listener-port/web-binary-name
```

PlanetsDemo 애플리케이션이 시작되면 홈 페이지가 표시됩니다.

필터 텍스트 상자에 을 입력한 후 Enter 키를 누릅니다. 데이터 조회 페이지가 표시됩니다.



예를 들어, PlanetsDemo 이름 필드에 EARTH를 입력한 다음 Enter 키를 누릅니다. 입력한 행성의 페이지가 표시됩니다.



에서 관리하는 Amazon ECS의 AWS 블루 에이지 런타임 업그레이드 AWS Fargate

이 가이드에서는 에서 관리하는 Amazon ECS의 AWS Blu Age 런타임을 업그레이드하는 방법을 설명합니다. AWS Fargate

주제

필수 조건

시작하기 전에 다음 사전 조건을 충족하는지 확인합니다.

- the section called “AWS 블루 에이지 런타임 사전 요구 사항” 및 the section called “AWS 블루에이지 런타임 온보딩”를 완료합니다.
- 업그레이드하려는 AWS 블루 에이지 런타임 버전을 다운로드하십시오. 자세한 설명은 the section called “AWS 블루에이지 런타임 온보딩” 섹션을 참조하세요. 프레임워크는 두 개의 바이너리 파일 (`aws-bluage-runtime-x.x.x.x.tar.gz`및`aws-bluage-webapps-x.x.x.x.tar.gz`) 로 구성되어 있습니다.

벨로시티 런타임 업그레이드

벨로시티 런타임을 업그레이드하려면 다음 단계를 완료합니다.

1. 원하는 AWS 블루에이지 런타임 버전으로 Docker 이미지를 다시 빌드하세요. 지침은 the section called “Amazon ECS에서 AWS 블루 에이지 런타임 설정 관리 AWS Fargate” 섹션을 참조하십시오.
2. 도커 이미지를 Amazon ECR 리포지토리로 푸시합니다.
3. Amazon ECS 서비스를 중지하고 다시 시작합니다.
4. 로그를 확인합니다.

AWS 블루 에이지 런타임이 성공적으로 업그레이드되었습니다.

Amazon ECS의 AWS 블루 에이지 런타임을 위한 Amazon CloudWatch 알람은 에서 관리합니다. AWS Fargate

배포된 애플리케이션에서 예외가 발생할 때마다 알림을 더 잘 CloudWatch 보이도록 하려면 애플리케이션 로그를 수신하도록 설정하고 발생 가능한 오류를 경고하는 경보를 추가하십시오.

알람 설정

CloudWatch 로그를 사용하면 애플리케이션과 필요에 따라 원하는 수의 지표와 경보를 구성할 수 있습니다.

특히, Amazon ECS 클러스터를 생성하는 동안 직접 사용 알림에 대한 사전 경보를 설정하여 오류 발생 시 알림을 받을 수 있습니다. AWS Blu Age 제어 시스템과의 연결 오류를 강조 표시하려면 로그에 문자열 “Error C”와 관련된 메트릭을 추가하십시오. 그런 다음 이 지표에 반응하는 경보를 정의할 수 있습니다.

Amazon ECS의 AWS Blu Age 런타임에서 라이선스가 부여된 종속성 설정은 에서 관리합니다. AWS Fargate

이 주제에서는 에서 관리하는 Amazon ECS의 AWS Blu Age Runtime과 함께 사용할 수 있는 추가 라이선스 종속성을 설정하는 방법을 설명합니다. AWS Fargate

주제

- 필수 조건
- 개요

필수 조건

시작하기 전에 다음 사전 조건을 충족하는지 확인합니다.

- the section called “AWS 블루 에이지 런타임 사전 요구 사항” 및 the section called “AWS 블루에이지 런타임 온보딩”를 완료합니다.
- 소스에서 다음 종속성을 가져오세요.

Oracle 데이터베이스

Oracle 데이터베이스 드라이버를 제공하세요. 예를 들어 ojdbc8-19.8.0.jar를 예로 들 수 있습니다.

IBM MQ 연결

IBM MQ 클라이언트를 공급하세요. 예를 들어 com.ibm.mq.allclient-9.3.0.15.jar를 예로 들 수 있습니다.

이 종속성 버전에서는 다음과 같은 전이적 종속성도 제공합니다.

- javax.jms-api-2.0.1.jar

- json-20080701.jar

DDS 프린터 파일

Jasper 보고서 라이브러리를 제공하세요. 예를 들어, jasperreports-6.16.0.jar이지만 최신 버전도 호환될 수 있습니다.

이 종속성 버전에서는 다음과 같은 전이적 종속성도 제공합니다.

- castor-core-1.4.1.jar
- castor-xml-1.4.1.jar
- commons-digester-2.1.jar
- ecj-3.21.0.jar
- itext-2.1.7.js8.jar
- javax.inject-1.jar
- jcommon-1.0.23.jar
- jfreechart-1.0.19.jar

개요

존속성 설치를 위해 다음 단계를 완료합니다.

1. 필요에 따라 위의 종속성을 Docker 이미지 빌드 폴더에 복사합니다.
2. JICS 또는 Blusam 데이터베이스가 Oracle에서 호스팅되는 경우 Oracle 데이터베이스 드라이버를 제공하십시오. *your-tomcat-path*/extra
3. Dockerfile에서 이러한 종속성을 에 복사하십시오. *your-tomcat-path*/extra
4. 도커 이미지를 빌드한 다음 Amazon ECR로 푸시합니다.
5. Amazon ECS 서비스를 중지하고 다시 시작합니다.
6. 로그를 확인합니다.

AWS Amazon EC2에서의 블루에이지 런타임 배포

이 섹션의 항목에서는 Amazon EC2에서 AWS Blu Age Runtime (비관리형) 을 설정하는 방법, 런타임 버전을 업데이트하는 방법, Amazon 경보를 사용하여 배포를 모니터링하는 방법, 라이선스가 부여된 종속성을 추가하는 방법을 설명합니다. CloudWatch

주제

Amazon EC2에서 AWS 블루 에이지 런타임 (비관리형) 설정

이 주제에서는 Amazon EC2에서 AWS Blu Age Runtime (비관리형) 을 사용하여 PlanetsDemo 샘플 애플리케이션을 설정하고 배포하는 방법을 설명합니다.

주제

필수 조건

시작하기 전에 다음 사전 조건을 충족하는지 확인합니다.

- AWS CLI 구성의 단계에 AWS CLI 따라 구성합니다.
- the section called “AWS 블루 에이지 런타임 사전 요구 사항” 및 the section called “AWS 블루에이지 런타임 온보딩”를 완료합니다.
- Amazon EC2에서 최신 AWS 블루 에이지 런타임을 포함하는 Amazon EC2 인스턴스를 생성합니다. 자세한 내용은 Amazon EC2 Linux 인스턴스 시작하기를 참조하세요.
- 예를 들어 SSM을 사용하여 Amazon EC2 인스턴스에 성공적으로 연결할 수 있는지 확인하세요.
- 에서 AWS 블루 에이지 런타임 (Amazon EC2) 을 다운로드하고 압축을 풉니다. *your-tomcat-path*/* 지침은 the section called “AWS 블루에이지 런타임 온보딩” 섹션을 참조하십시오.
- PlanetsDemo애플리케이션 아카이브를 다운로드하십시오.
- 아카이브의 압축을 풀고 선택한 Amazon S3 버킷에 애플리케이션을 업로드합니다.
- JICS용 Amazon Aurora PostgreSQL 데이터베이스를 생성하고 해당 데이터베이스에서 쿼리를 실행합니다. `PlanetsDemo-v1/jics/sql/initJics.sql` Amazon Aurora PostgreSQL 데이터베이스를 생성하는 방법에 대한 자세한 내용은 Aurora PostgreSQL DB 클러스터 생성 및 연결을 참조하십시오.

설정

샘플 애플리케이션을 설정하려면 다음 단계를 완료하십시오. PlanetsDemo

1. Amazon EC2 인스턴스에 연결하고 아파치 톰캣 9 설치 `conf` 폴더 아래의 폴더로 이동합니다. 편집할 `catalina.properties` 파일을 열고 다음 `common.loader` 줄로 시작하는 줄을 바꿉니다.

```
common.loader="${catalina.base}/lib","${catalina.base}/lib/
*.jar","${catalina.home}/lib","${catalina.home}/lib/*.jar","${catalina.home}/
shared","${catalina.home}/shared/*.jar","${catalina.home}/extra","${catalina.home}/
extra/*.jar"
```

2. *<your-tomcat-path>*/webapps 폴더로 이동합니다.
3. 다음 명령을 사용하여 Amazon S3 버킷에서 PlanetsDemo -v1/webapps/ 폴더에서 사용 가능한 PlanetsDemo 바이너리를 복사합니다.

```
aws s3 cp s3://path-to-demo-app-webapps/ . --recursive
```

> **Note**
>
> 이전에 아카이브를 압축 해제한 버킷의 올바른 Amazon S3 `path-to-demo-app-webapps` URI로 교체하십시오 PlanetsDemo .

4. `PlanetsDemo-v1/config/` 폴더의 콘텐츠를 *<your-tomcat-path>*/config/에 복사합니다.
5. 사전 요구 사항의 일부로 생성한 데이터베이스의 연결 정보를 파일의 다음 스니펫에 제공하십시오. `application-main.yml` 자세한 내용은 Aurora PostgreSQL DB 클러스터를 참조하세요.

```
datasource:
   jicsDs:
     driver-class-name :
     url:
     username:
     password:
     type :
```

6. Apache Tomcat 서버를 시작하고 로그를 확인합니다.

```
your-tomcat-path/startup.sh
```

```
tail -f your-tomcat-path/logs/catalina.log
```

C로 시작하고 숫자가 오는 오류 코드 (예: CXXXXX) 를 발견하면 오류 메시지를 기록해 두십시오. 예를 들어, 오류 코드 C5102 오류는 잘못된 인프라 구성을 나타내는 일반적인 오류입니다.

PlanetsDemo 애플리케이션을 테스트하십시오.

배포된 PlanetsDemo 애플리케이션의 상태를 확인하려면 `load-balancer-DNS-name`listener-port, 교체한 후 설정에 맞는 올바른 값을 `web-binary-name` 사용하여 다음 명령을 실행합니다.

```
curl http://load-balancer-DNS-name:listener-port/gapwalk-application/
```

애플리케이션이 실행 중인 경우 다음 출력 메시지가 표시됩니다`Jics application is running`.

그런 다음 다음 명령을 실행합니다.

```
curl http://load-balancer-DNS-name:listener-port/jac/api/services/rest/jicsservice/
```

애플리케이션이 실행 중인 경우 다음과 같은 출력 메시지가 표시됩니다`Jics application is running`.

```
Jics application is running
```

Blusam을 구성한 경우 다음 명령을 실행하면 빈 응답이 표시될 수 있습니다.

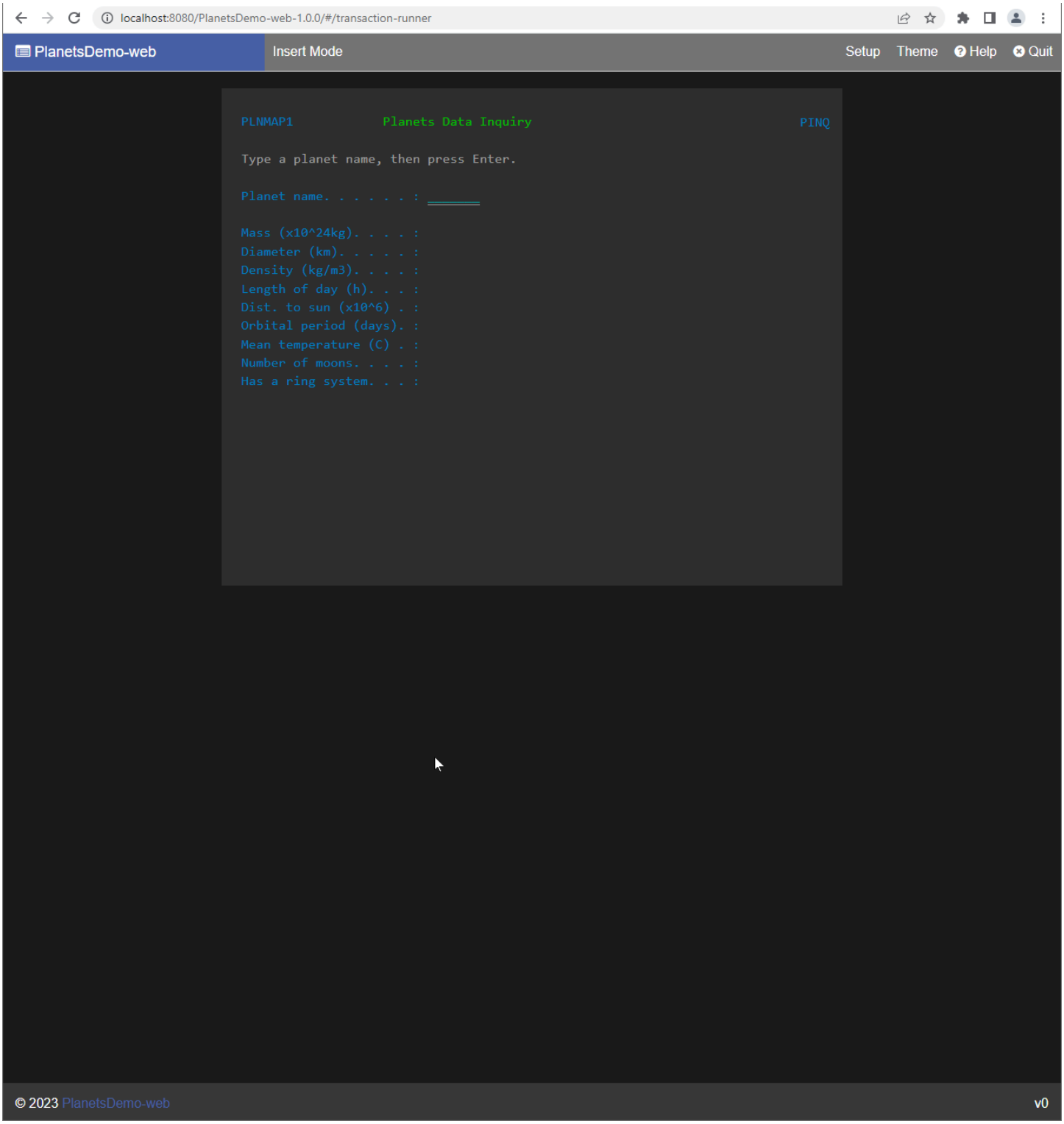
```
curl http://load-balancer-DNS-name:listener-port/bac/api/services/rest/bluesamserver/
serverIsUp
```

웹 바이너리의 이름을 기록해 둡니다 (변경되지 않은 경우 PlanetsDemo -web-1.0.0). PlanetsDemo 애플리케이션에 액세스하려면 다음 형식의 URL을 사용합니다.

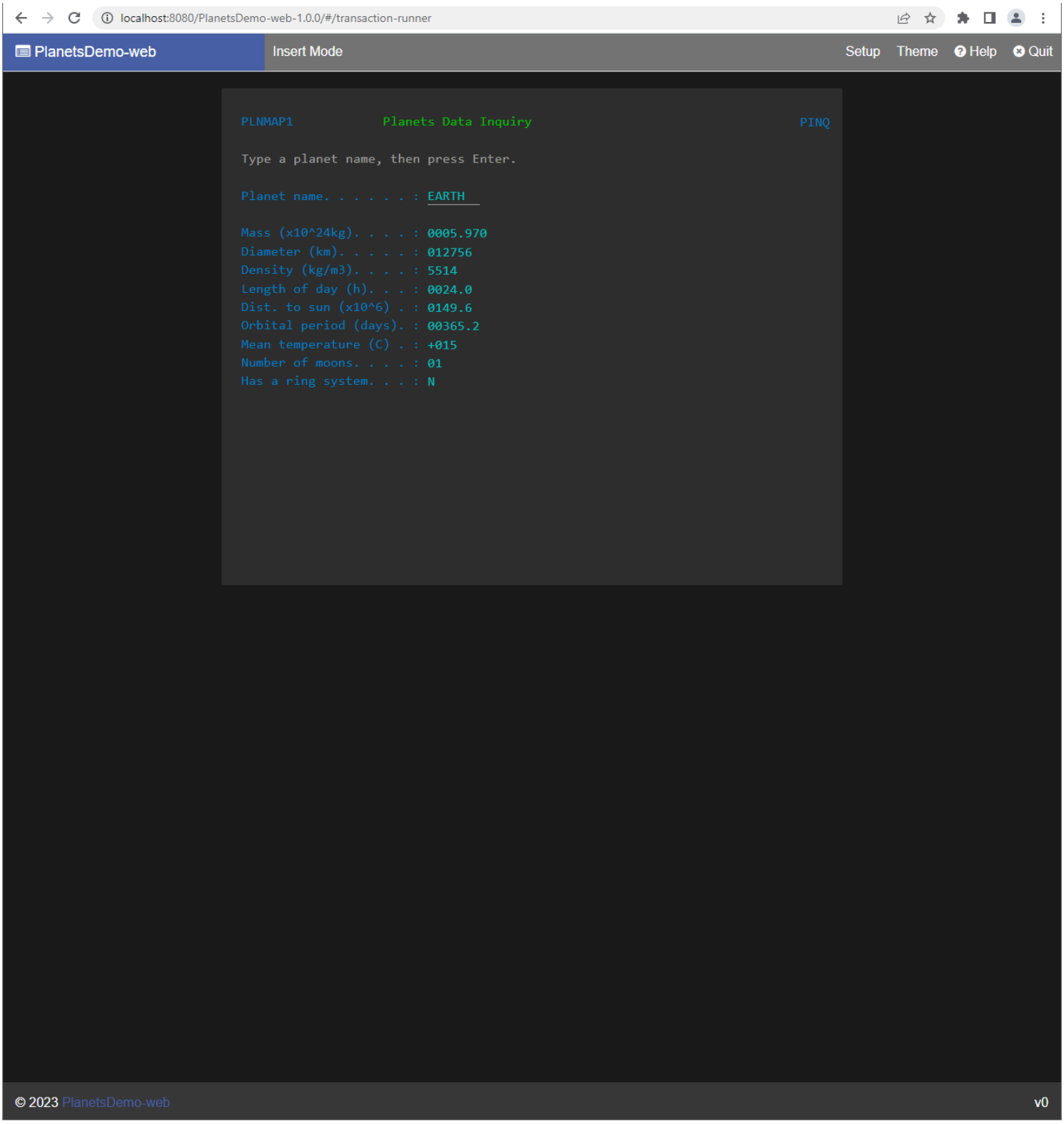
```
https://load-balancer-DNS-name:listener-port/web-binary-name
```

PlanetsDemo 애플리케이션이 시작되면 홈 페이지가 표시됩니다.

필터 텍스트 상자에 을 입력한 후 Enter 키를 누릅니다. 데이터 조회 페이지가 표시됩니다.



예를 들어, PlanetsDemo 이름 필드에 EARTH를 입력한 다음 Enter 키를 누릅니다. 입력한 행성의 페이지가 표시됩니다.



Amazon AWS EC2의 블루 에이지 런타임 업그레이드

이 안내서는 Amazon EC2에서 AWS 블루 에이지 런타임을 업그레이드하는 방법을 설명합니다.

주제

필수 조건

시작하기 전에 다음 사전 조건을 충족하는지 확인합니다.

- the section called “AWS 블루 에이지 런타임 사전 요구 사항” 및 the section called “AWS 블루에이지 런타임 온보딩”를 완료합니다.
- 최신 AWS 블루 에이지 런타임을 포함하는 Amazon EC2 인스턴스가 있는지 확인하십시오. 자세한 내용은 Amazon EC2 Linux 인스턴스 시작하기를 참조하세요.
- 예를 들어 SSM을 사용하여 Amazon EC2 인스턴스에 성공적으로 연결할 수 있는지 확인하십시오.
- 업그레이드하려는 AWS 블루 에이지 런타임 버전 (Amazon EC2) 을 다운로드하십시오. 자세한 내용은 the section called “AWS 블루 에이지 런타임 (비관리형) 설치” 프레임워크는 두 개의 바이너리 파일인 `aws-bluage-runtime-x.x.x.x.tar.gz` 및 `aws-bluage-webapps-x.x.x.x.tar.gz`로 구성되어 있습니다.

개요

벨로시티 런타임을 업그레이드하려면 다음 단계를 완료합니다.

1. Amazon EC2 인스턴스에 연결하고 다음 명령을 실행하여 사용자를 su로 변경합니다.

   ```
   sudo su
   ```

   이 자습서의 명령을 실행하려면 수퍼유저 권한이 필요합니다.
2. 각 바이너리 파일당 하나씩 두 개의 폴더를 생성합니다.
3. 각 폴더의 이름은 바이너리 파일과 같은 이름으로 지정합니다.
4. 각 바이너리 파일을 해당 폴더에 복사합니다.

> **Warning**
>
> 각 바이너리를 추출하면 이름이 같은 폴더가 생성됩니다. 따라서 같은 위치에 있는 두 바이너리 파일을 차례로 추출하면 내용을 덮어쓰게 됩니다.

5. 바이너리를 추출하려면 다음 명령을 사용합니다. 각 폴더에서 명령을 실행합니다.

```
tar xvf aws-bluage-runtime-x.x.x.x.tar.gz
tar xvf aws-bluage-webapps-x.x.x.x.tar.gz
```

6. 다음 명령을 사용하여 Apache Tomcat 서비스를 중지합니다.

```
systemctl stop tomcat.service
systemctl stop tomcat-webapps.service
```

7. <your-tomcat-path>/shared/의 내용을 aws-bluage-runtime-x.x.x.x/velocity/shared/의 내용으로 바꾸세요.
8. <your-tomcat-path>/webapps/gapwalk-application.war를 aws-bluage-runtime-x.x.x.x/velocity/webapps/gapwalk-application.war로 바꿉니다.
9. <your-tomcat-path>/webapps/의 war 파일, 즉 bac.war 및 jac.war를 aws-bluage-webapps-x.x.x.x/velocity/webapps/의 같은 파일로 바꾸세요.
10. 다음 명령을 실행하여 Apache Tomcat 서비스를 시작합니다.

```
systemctl start tomcat.service
systemctl start tomcat-webapps.service
```

11. 로그를 확인합니다.

다음 명령을 실행하여 배포된 애플리케이션의 상태를 확인합니다.

```
curl http://localhost:8080/gapwalk-application/
```

다음 메시지가 나타납니다.

```
Jics application is running
```

```
curl http://localhost:8181/jac/api/services/rest/jicsservice/
```

다음 메시지가 나타납니다.

```
Jics application is running
```

```
curl http://localhost:8181/bac/api/services/rest/bluesamserver/serverIsUp
```

응답은 비워 두어야 합니다.

AWS Blu Age 런타임이 성공적으로 업그레이드되었습니다.

AWS 블루 에이지 런타임 (Amazon EC2) 아마존 알람 CloudWatch

배포된 애플리케이션에서 예외가 발생하여 애플리케이션이 유예 기간에 들어갈 때 알림을 더 잘 보이게 하려면 애플리케이션 로그를 CloudWatch 수신하도록 설정하고 발생 가능한 오류를 경고하는 경보를 추가할 수 있습니다.

CloudWatch 로깅 배포

기본적으로 `application-main.yml` 파일에는 이라는 다른 로깅 구성 `logback-cloudwatch.yml` 파일에 대한 참조가 포함됩니다.

```
logging:
  config: classpath:logback-cloudwatch.xml
```

두 파일 모두 config 폴더에 있으며, 다음 섹션에 설명된 대로 이렇게 CloudWatch 로깅이 구성됩니다.

로깅 구성 CloudWatch

`logback-cloudwatch.xml` 파일의 콘텐츠는 다음과 같습니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration>
<configuration>

    <appender name="console" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%date{yyyy-MM-dd HH:mm:ss.SSS,UTC}  %level --- [%thread{15}]
 %logger{40} : %msg%n%xThrowable</pattern>
        </encoder>
    </appender>

    <appender name="cloudwatch"
 class="com.netfective.bluage.runtime.cloudwatchlogger.CloudWatchAppender">
        <logGroup>BluAgeRuntimeOnEC2-Logs</logGroup>
        <logStream>%date{yyyy-MM-dd,UTC}.%instanceId.%uuid</logStream>
```

```
        <layout>
            <pattern>%date{yyyy-MM-dd HH:mm:ss.SSS,UTC}  %level --- [%thread{15}]
 %logger{40} : %msg%n%xThrowable</pattern>
        </layout>
        <appender-ref ref="console" />
    </appender>

    <root level="INFO">
        <appender-ref ref="cloudwatch" />
    </root>
</configuration>
```

<appender name="cloudwatch"/> 요소 외부의 모든 항목은 표준 로그백 구성입니다. 이 파일에는 두 개의 어펜더가 있습니다. 하나는 콘솔에 로그를 보내는 콘솔 어펜더이고 다른 하나는 로그를 보내는 CloudWatch 어펜더입니다. CloudWatch

`root` 요소의 `level` 속성은 전체 애플리케이션의 로깅 수준을 지정합니다.

태그 내 필수 값은 다음과 같습니다. <appender name="cloudwatch"/>

- <logGroup/>:로그 그룹 이름을 설정합니다. CloudWatch 값이 지정하지 않은 경우 기본값은 `BluAgeRuntimeOnEC2-Logs`입니다. 로그 그룹이 존재하지 않으면 자동 생성됩니다. 이 동작은 아래에서 설명하는 구성을 통해 변경할 수 있습니다.
- <logStream/>: LogStream (로그 그룹 내부) 의 이름을 설정합니다. CloudWatch

선택적 값:

- <region/>: 로그 스트림이 기록될 지역을 재정의합니다. 기본적으로 로그는 EC2 인스턴스와 동일한 리전에 이동합니다.
- <layout/>: 로그 메시지가 사용할 패턴.
- <maxbatchsize/>: CloudWatch 작업당 전송할 최대 로그 메시지 수입니다.
- <maxbatchtimemillis/>: CloudWatch 로그를 기록하는 데 걸리는 시간 (밀리초).
- <maxqueuewaittimemillis/>: 내부 로그 대기열에 요청을 삽입하려고 시도하는 데 걸리는 시간(밀리초).
- <internalqueuesize/>: 내부 대기열의 최대 크기입니다.
- <createlogdests/>: 로그 그룹과 로그 스트림이 없는 경우 이를 생성합니다.
- <initialwaittimemillis/>: 시작 시 스레드를 휴면 상태로 만들려는 시간입니다. 이 초기 대기 시간 동안 로그가 처음 누적될 수 있습니다.

- <maxeventmessagesize/>: 로그 이벤트의 최대 크기. 이 크기를 초과하는 로그는 전송되지 않습니다.
- <truncateeventmessages/>: 너무 긴 메시지는 잘라냅니다.
- <printrejectedevents/>: 비상 첨부 파일을 사용합니다.

CloudWatch 설정

위 구성에서 로그를 올바르게 푸시하려면 Amazon EC2 IAM 인스턴스 프로필 역할을 업데이트하여 'BluAgeRuntimeOnEC2-logs' 로그 그룹 및 해당 로그 스트림에 대한 추가 권한을 부여하십시오. CloudWatch

- `logs:CreateLogStream`
- `logs:DescribeLogStreams`
- `logs:CreateLogGroup`
- `logs:PutLogEvents`
- `logs:DescribeLogGroups`

알람 설정

CloudWatch 로그 덕분에 애플리케이션과 필요에 따라 다양한 지표와 경보를 구성할 수 있습니다. 특히, 사용 알림에 대한 사전 경보를 설정하여 애플리케이션이 유예 기간에 빠질 수 있는 오류가 발생할 경우 경고를 보내고 결국에는 아예 작동하지 않도록 할 수 있습니다. 이를 위해 로그에 “Error C5001" 문자열과 관련된 메트릭을 추가할 수 있습니다. 이 메트릭은 AWS Blu Age 제어 시스템과의 연결 오류를 강조 표시합니다. 그런 다음 이 지표에 반응하는 경보를 정의할 수 있습니다.

Amazon EC2의 AWS 블루 에이지 런타임에서 라이선스가 부여된 종속성 설정

이 가이드에서는 Amazon EC2의 AWS Blu Age Runtime과 함께 사용할 수 있는 추가 라이선스 종속성을 설정하는 방법을 설명합니다.

주제

- 필수 조건
- 개요
- JAC 및 BAC 웹앱의 종속성을 설정합니다

필수 조건

시작하기 전에 다음 사전 조건을 충족하는지 확인합니다.

- the section called “AWS 블루 에이지 런타임 사전 요구 사항” 및 the section called “AWS 블루에이지 런타임 온보딩”를 완료합니다.
- Amazon EC2에서 최신 AWS 블루 에이지 런타임을 포함하는 Amazon EC2 인스턴스가 있는지 확인하십시오. 자세한 내용은 Amazon EC2 Linux 인스턴스 시작하기를 참조하세요.
- 예를 들어 SSM을 사용하여 Amazon EC2 인스턴스에 성공적으로 연결할 수 있는지 확인하십시오.
- 해당 소스에서 다음 종속성을 가져오십시오.

Oracle 데이터베이스

Oracle 데이터베이스 드라이버를 제공하세요. ojdbc8-19.8.0.jar 버전을 사용하여 AWS 블루 에이지 런타임 (Amazon EC2 기반) 기능을 테스트했지만 최신 버전과 호환될 수 있습니다.

IBM MQ 연결

IBM MQ 클라이언트를 공급하세요. com.ibm.mq.allclient-9.3.0.15.jar 버전을 사용하여 AWS 블루 에이지 런타임 (Amazon EC2 기반) 기능을 테스트했지만 최신 버전과 호환될 수 있습니다.

이 종속성 버전에서는 다음과 같은 전이적 종속성도 제공합니다.

- javax.jms-api-2.0.1.jar
- json-20080701.jar

DDS 프린터 파일

Jasper 보고서 라이브러리를 제공하세요. jasperreports-6.16.0.jar를 사용하여 AWS 블루 에이지 런타임 (Amazon EC2 기반) 기능을 테스트했지만 최신 버전도 호환될 수 있습니다.

이 종속성 버전에서는 다음과 같은 전이적 종속성도 제공합니다.

- castor-core-1.4.1.jar
- castor-xml-1.4.1.jar
- commons-digester-2.1.jar
- ecj-3.21.0.jar

- itext-2.1.7.js8.jar
- javax.inject-1.jar
- jcommon-1.0.23.jar
- jfreechart-1.0.19.jar

개요

존속성 설치를 위해 다음 단계를 완료합니다.

1. Amazon EC2 인스턴스에 연결하고 다음 명령을 실행하여 사용자를 su로 변경합니다.

   ```
   sudo su
   ```

   이 자습서의 명령을 실행하려면 수퍼유저 권한이 필요합니다.
2. <your-tomcat-path>/extra/ 폴더로 이동합니다.

   ```
   cd <your-tomcat-path>/extra/
   ```
3. 필요에 따라 위의 종속성을 이 폴더에 복사하세요.
4. 다음 명령을 실행하여 tomcat.서비스를 중지하고 시작합니다.

   ```
   systemctl stop tomcat.service
   ```

   ```
   systemctl start tomcat.service
   ```
5. 서비스 상태를 확인하여 실행 중인지 확인합니다.

   ```
   systemctl status tomcat.service
   ```
6. 로그를 확인합니다.

JAC 및 BAC 웹앱의 종속성을 설정합니다

1. JICS 또는 Blusam 데이터베이스를 Oracle에서 호스팅하는 경우 Oracle 데이터베이스 드라이버를 <your-tomcat-path>/extra에서 제공해야 합니다
2. 폴더가 아직 없으면 새로 만드세요.
3. 아파치 톰캣 서버를 중지했다가 다시 시작하십시오.

4. 로그를 확인합니다.

Blu Age 개발자 IDE로 소스 코드 수정

AWS관리형 블루에이지 런타임 엔진을 사용하는 경우 AWS 블루에이지 개발자를 사용하여 생성된 소스 코드를 수정할 수 있습니다. 어떤 이유로 현대화된 코드를 업데이트해야 하거나 레거시 소스 코드의 일부를 현대화할 수 없는 경우 이 방법을 사용하는 것이 좋습니다. Amazon AppStream 2.0을 통해 블루 에이지 디벨로퍼에 액세스할 수 있습니다. 이 섹션에서는 2.0에서 블루에이지 개발자를 설정하는 방법을 설명합니다. AppStream 또한 샘플 애플리케이션을 사용하여 Blu Age Developer를 사용하여 소스 코드를 업데이트하는 방법도 설명합니다. PlanetsDemo

주제

- 튜토리얼: AWS 블루 에이지 개발자 IDE용 AppStream 2.0 설정
- 튜토리얼: 2.0에서 AWS AppStream 블루 에이지 개발자 사용

튜토리얼: AWS 블루 에이지 개발자 IDE용 AppStream 2.0 설정

AWS 메인프레임 현대화는 Amazon 2.0을 통해 여러 도구를 제공합니다. AppStream AppStream 2.0은 애플리케이션을 다시 작성하지 않고도 데스크톱 애플리케이션을 사용자에게 스트리밍할 수 있는 안전한 완전 관리형 애플리케이션 스트리밍 서비스입니다. AppStream 2.0은 사용자가 원하는 디바이스에서 반응성이 뛰어나고 유동적인 사용자 경험을 제공하면서 필요한 애플리케이션에 즉시 액세스할 수 있도록 합니다. AppStream 2.0을 사용하여 런타임 엔진별 도구를 호스팅하면 고객 애플리케이션 팀이 웹 브라우저에서 직접 도구를 사용하고 Amazon S3 버킷 또는 리포지토리에 저장된 애플리케이션 파일과 상호 작용할 수 있습니다. CodeCommit

2.0의 브라우저 지원에 대한 자세한 내용은 Amazon AppStream AppStream 2.0 관리 안내서의 시스템 요구 사항 및 기능 지원 (웹 브라우저) 을 참조하십시오. AppStream 2.0을 사용할 때 문제가 발생하는 경우 Amazon AppStream 2.0 관리 안내서의 AppStream 2.0 사용자 문제 해결을 참조하십시오.

이 문서에서는 AppStream 2.0 플릿에서 AWS Blu Age Developer IDE를 설정하는 방법을 설명합니다.

주제

- 전제 조건
- 1단계: Amazon S3 버킷 생성
- 2단계: S3 버킷에 정책 연결
- 3단계: Amazon S3 버킷에 파일 업로드

- 4단계: AWS CloudFormation 템플릿 다운로드
- 5단계: 다음을 사용하여 플릿 생성 AWS CloudFormation
- 6단계: 인스턴스 액세스
- 리소스 정리

전제 조건

2.0 미만의 AWS 블루에이지 개발자 IDE를 설정하는 데 필요한 아티팩트가 들어 있는 아카이브 파일을 다운로드하십시오. AppStream

> **Note**
>
> 대용량 파일입니다. 작업 시간 초과에 문제가 있는 경우 Amazon EC2 인스턴스를 사용하여 업로드 및 다운로드 성능을 개선하는 것이 좋습니다.

1단계: Amazon S3 버킷 생성

생성하려는 AppStream 2.0 AWS 리전 플릿과 동일한 위치에 Amazon S3 버킷을 생성합니다. 이 버킷에는 이 자습서를 완료하는 데 필요한 아티팩트가 포함됩니다.

2단계: S3 버킷에 정책 연결

이 자습서를 위해 생성한 버킷에 다음 정책을 연결합니다. 생성한 버킷의 실제 이름으로 MYBUCKET을 바꿔야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowAppStream2.0ToRetrieveObjects",
        "Effect": "Allow",
        "Principal": {
            "Service": "appstream.amazonaws.com"
        },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::MYBUCKET/*"
    }]
}
```

3단계: Amazon S3 버킷에 파일 업로드

사전 요구 사항에서 다운로드한 파일의 압축을 풀고 `appstream` 폴더를 버킷에 업로드합니다. 이 폴더를 업로드하면 버킷에 올바른 구조가 만들어집니다. 자세한 내용을 알아보려면 Amazon S3 사용 안내서의 객체 업로드를 참조하세요.

4단계: AWS CloudFormation 템플릿 다운로드

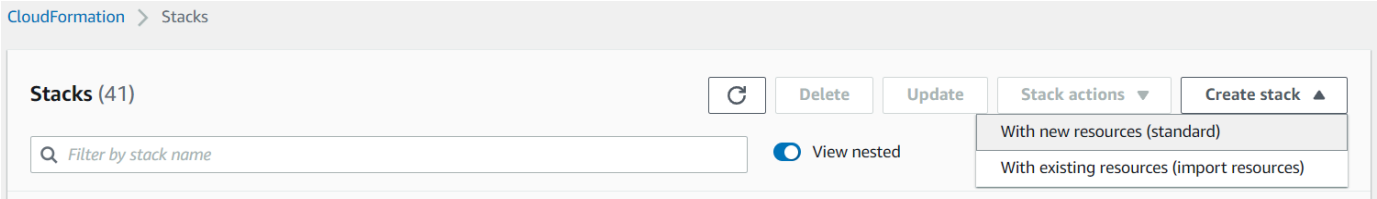
다음 AWS CloudFormation 템플릿을 다운로드하세요. AppStream 2.0 플릿을 생성하고 채우려면 이러한 템플릿이 필요합니다.

- cfn-m2- .yaml appstream-elastic-fleet-linux
- cfn-m2- appstream-bluage-dev-tools -linux.yaml
- cfn-m2 appstream-bluage-shared-linux - .yaml
- cfn-m2- appstream-chrome-linux .yaml
- cfn-m2- appstream-eclipse-jee-linux .yaml
- cfn-m2- appstream-pgadmin-linux .yaml

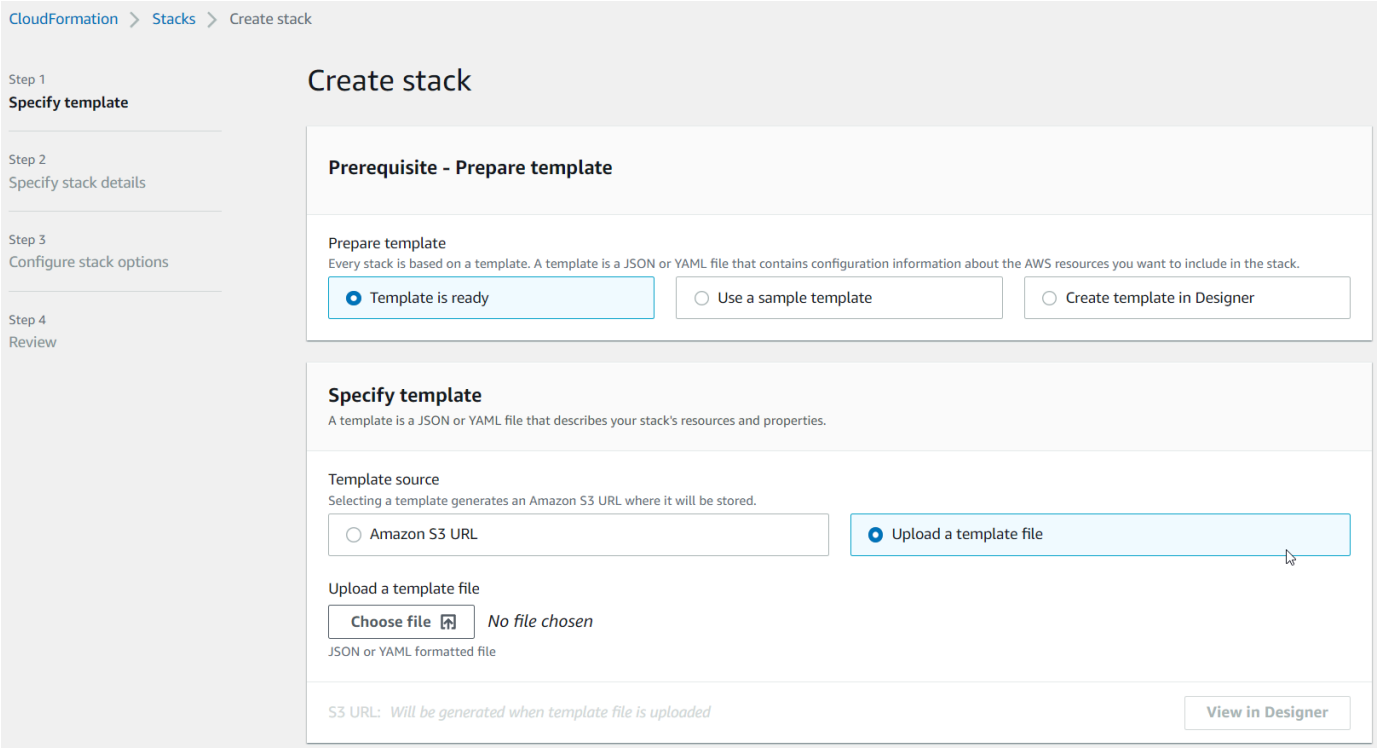
5단계: 다음을 사용하여 플릿 생성 AWS CloudFormation

이 단계에서는 `cfn-m2-appstream-elastic-fleet-linux.yaml` AWS CloudFormation 템플릿을 사용하여 AWS Blu Age Developer IDE를 호스팅할 AppStream 2.0 플릿 및 스택을 생성합니다. 플릿과 스택을 만든 후에는 이전 단계에서 다운로드한 다른 AWS CloudFormation 템플릿을 실행하여 Developer IDE 및 기타 필수 도구를 설치합니다.

1. AWS 관리 콘솔로 AWS CloudFormation 이동하여 [Stacks] 를 선택합니다.
2. 스택에서 스택 생성 및 새 리소스 사용(표준)을 선택합니다.



3. 스택 생성에서 템플릿 준비 완료를 선택하고 템플릿 파일을 업로드합니다.



4. 파일 선택을 선택하고 `cfn-m2-appstream-elastic-fleet-linux.yaml` 파일로 이동합니다. 다음을 선택합니다.
5. 스택 세부 정보 지정(Specify Stack details) 페이지에 다음 정보를 입력하세요.
   - 스택의 이름을 입력합니다.
   - 기본 보안 그룹과 해당 보안 그룹의 서브넷 2개

   > **Note**
   >
   > 보안 그룹의 두 서브넷은 서로 다른 가용 영역에 있어야 합니다.

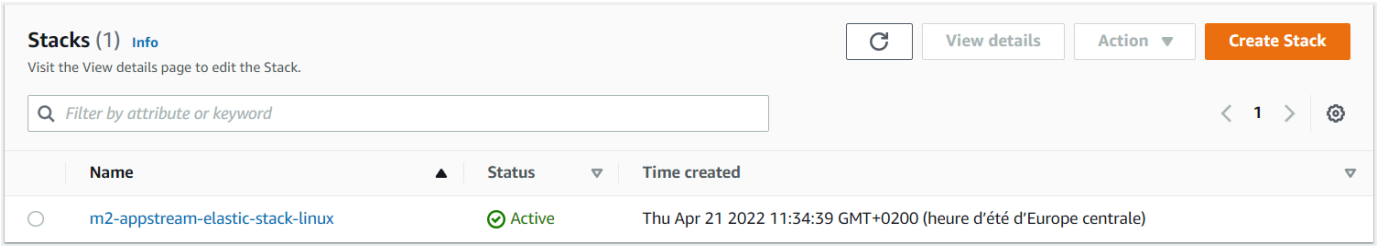
6. 다음을 선택한 다음 다음을 다시 선택합니다.
7. 사용자 지정 이름으로 IAM 리소스를 생성할 AWS CloudFormation 수 있음을 인정합니다를 선택합니다. 그런 다음 제출을 선택합니다.
8. 플릿을 생성한 후 다운로드한 다른 템플릿으로 CloudFormation 스택을 생성하여 애플리케이션 설정을 완료하십시오. BucketName매번 올바른 S3 버킷을 가리키도록 업데이트해야 합니다. BucketName CloudFormation콘솔에서 편집할 수 있습니다. 템플릿 파일을 직접 편집하고 `S3Bucket` 속성을 업데이트하는 방법도 있습니다.

> **Note**
>
> 다운로드한 템플릿은 `appstream/bluage/developer-ide/`라는 폴더 구조를 가진 S3 버킷에서 자산을 찾을 것으로 예상합니다. 버킷은 생성한 AWS 리전 플릿과 동일해야 합니다.

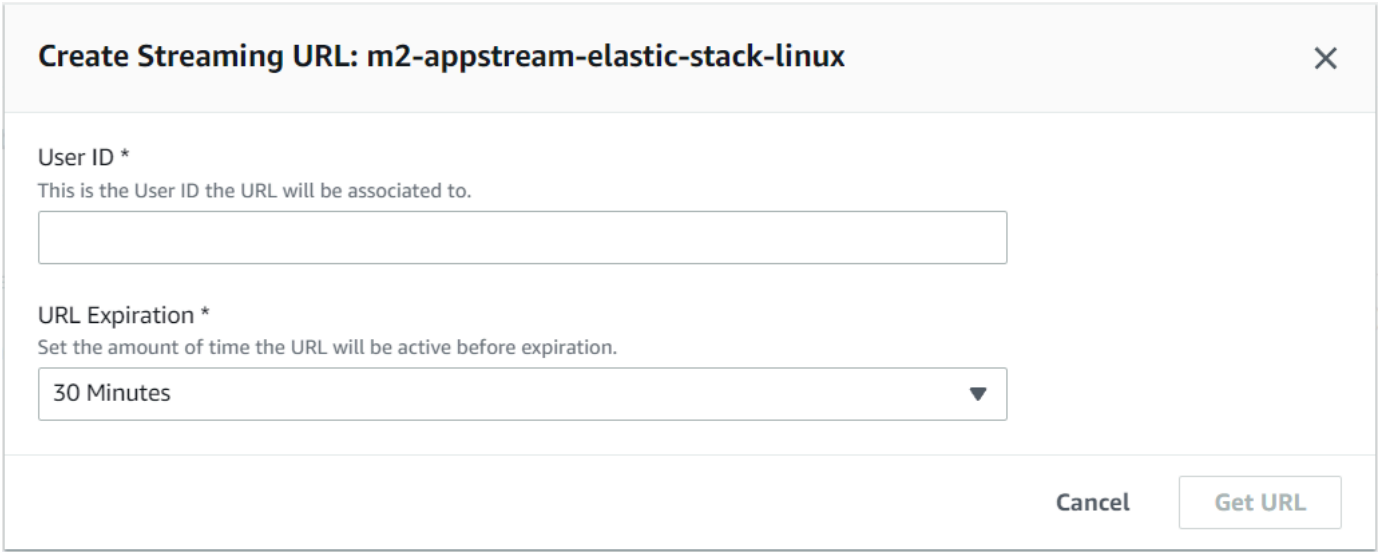
6단계: 인스턴스 액세스

플릿을 생성하고 시작한 후에는 네이티브 클라이언트를 통해 플릿에 액세스할 수 있는 임시 링크를 생성할 수 있습니다.

1. 에서 AppStream 2.0으로 이동하여 이전에 생성한 스택을 선택합니다. AWS Management Console



2. 스택 세부 정보 페이지에서 작업을 선택한 다음 스트리밍 URL 생성을 선택합니다.



3. 스트리밍 URL 생성에서 임의의 사용자 ID와 URL 만료 시간을 입력한 다음 URL 가져오기를 선택합니다. 브라우저나 네이티브 클라이언트로 스트리밍하는 데 사용할 수 있는 URL을 얻을 수 있습니다. 네이티브 클라이언트로 스트리밍하는 것이 좋습니다.

리소스 정리

생성된 스택과 플릿을 정리하는 절차는 AppStream 2.0 플릿 및 스택 생성을 참조하십시오.

AppStream 2.0 객체를 삭제한 경우 사용자 또는 계정 관리자가 애플리케이션 설정 및 홈 폴더의 S3 버킷을 정리할 수도 있습니다.

> **Note**
>
> 특정 사용자의 홈 폴더는 모든 플릿에서 고유하므로 동일한 계정에서 다른 AppStream 2.0 스택이 활성 상태인 경우 해당 폴더를 유지해야 할 수 있습니다.

AppStream 2.0 콘솔에서는 사용자를 삭제할 수 없습니다. 대신 AWS CLI와 함께 서비스 API를 사용해야 합니다. 자세한 내용은 Amazon AppStream 2.0 관리 안내서의 사용자 풀 관리를 참조하십시오.

튜토리얼: 2.0에서 AWS AppStream 블루 에이지 개발자 사용

이 튜토리얼에서는 AppStream 2.0 버전의 AWS Blu Age Developer에 액세스하고 샘플 애플리케이션과 함께 사용하여 기능을 시험해 보는 방법을 보여줍니다. 이 자습서를 마치면 자체 애플리케이션에 동일한 단계를 사용할 수 있습니다.

주제

1단계: 데이터베이스 생성

이 단계에서는 Amazon RDS를 사용하여 관리형 PostgreSQL 데이터베이스를 생성합니다. 이 데이터베이스는 데모 애플리케이션이 구성 정보를 저장하는 데 사용합니다.

1. Amazon RDS 콘솔을 엽니다.
2. 데이터베이스 > 데이터베이스 생성을 선택합니다.
3. 표준 생성 > PostgreSQL을 선택하고 기본 버전을 그대로 둔 다음 프리 티어를 선택합니다.
4. DB 인스턴스 식별자를 선택합니다.
5. 보안 인증 설정에서 AWS Secrets Manager에서 마스터 보안 인증 관리를 선택합니다. 자세한 내용은 Amazon RDS 사용 설명서의 Amazon RDS 및 AWS Secrets Manager을 사용한 암호 관리를 참조하세요.
6. VPC가 AppStream 2.0 인스턴스에 사용하는 것과 동일한지 확인하십시오. 관리자에게 이 값을 요청할 수 있습니다.
7. VPC 보안 그룹의 경우 신규 생성을 선택합니다.
8. 퍼블릭 액세스를 예로 설정합니다.
9. 다른 기본값을 그대로 둡니다. 이 값을 검토하세요.
10. 데이터베이스 생성를 선택합니다.

인스턴스에서 데이터베이스 서버에 액세스할 수 있게 하려면 Amazon RDS에서 데이터베이스 서버를 선택합니다. 연결 및 보안에서 데이터베이스 서버의 VPC 보안 그룹을 선택합니다. 이 보안 그룹은 이전에 사용자를 위해 생성되었으므로 RDS 관리 콘솔로 생성의 설명과 비슷한 설명이 있어야 합니다. 작업 > 인바운드 규칙 편집을 선택하고 규칙 추가를 선택한 다음 PostgreSQL 유형의 규칙을 생성합니다. 규칙 소스의 경우 보안 그룹 기본값을 사용합니다. 소스 필드에 소스 이름을 입력한 다음 제안된 ID를 수락할 수 있습니다. 마지막으로 규칙 저장을 선택합니다.

2단계: 환경에 액세스

이 단계에서는 2.0의 AWS 블루 에이지 개발 환경에 액세스합니다. AppStream

1. AppStream 2.0 인스턴스에 액세스하는 적절한 방법은 관리자에게 문의하십시오. 가능한 클라이언트 및 구성에 대한 일반 정보는 Amazon AppStream AppStream 2.0 관리 안내서의 2.0 액세스 방법 및 클라이언트를 참조하십시오. 최상의 경험을 위해 네이티브 클라이언트를 사용하는 것을 고려해 보세요.
2. AppStream 2.0에서는 데스크톱을 선택합니다.

3단계: 런타임 설정

이 단계에서는 AWS Blu Age 런타임을 설정합니다. 런타임을 처음 실행할 때 설정해야 하며 런타임 업그레이드 알림을 받으면 다시 설정해야 합니다. 이 단계를 수행하면 `.m2` 폴더가 채워집니다.

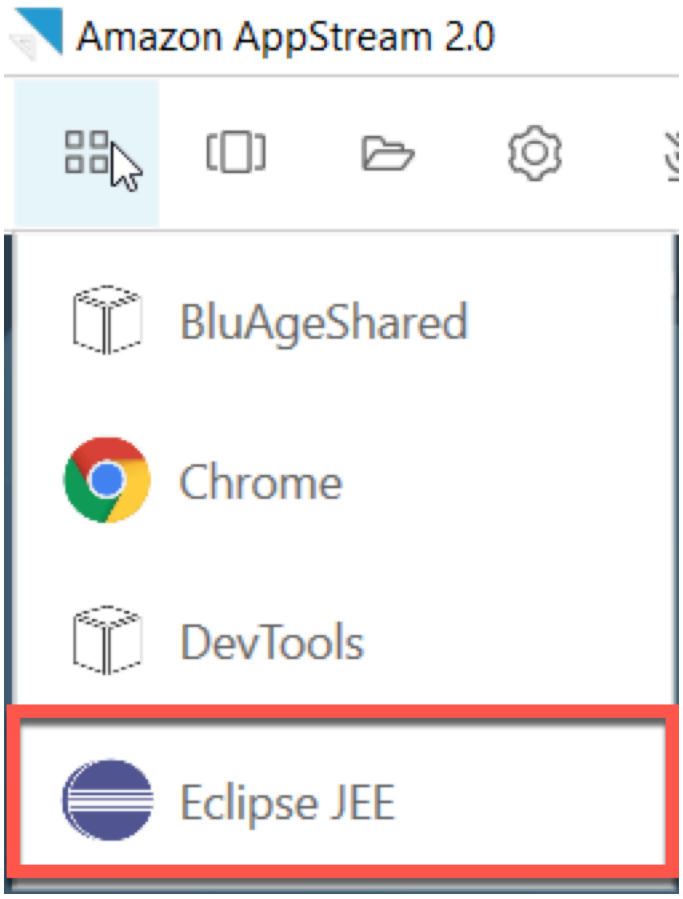
1. 메뉴 막대에서 애플리케이션을 선택한 다음 터미널을 선택합니다.
2. 다음 명령을 입력합니다.

```
~/_install-velocity-runtime.sh
```

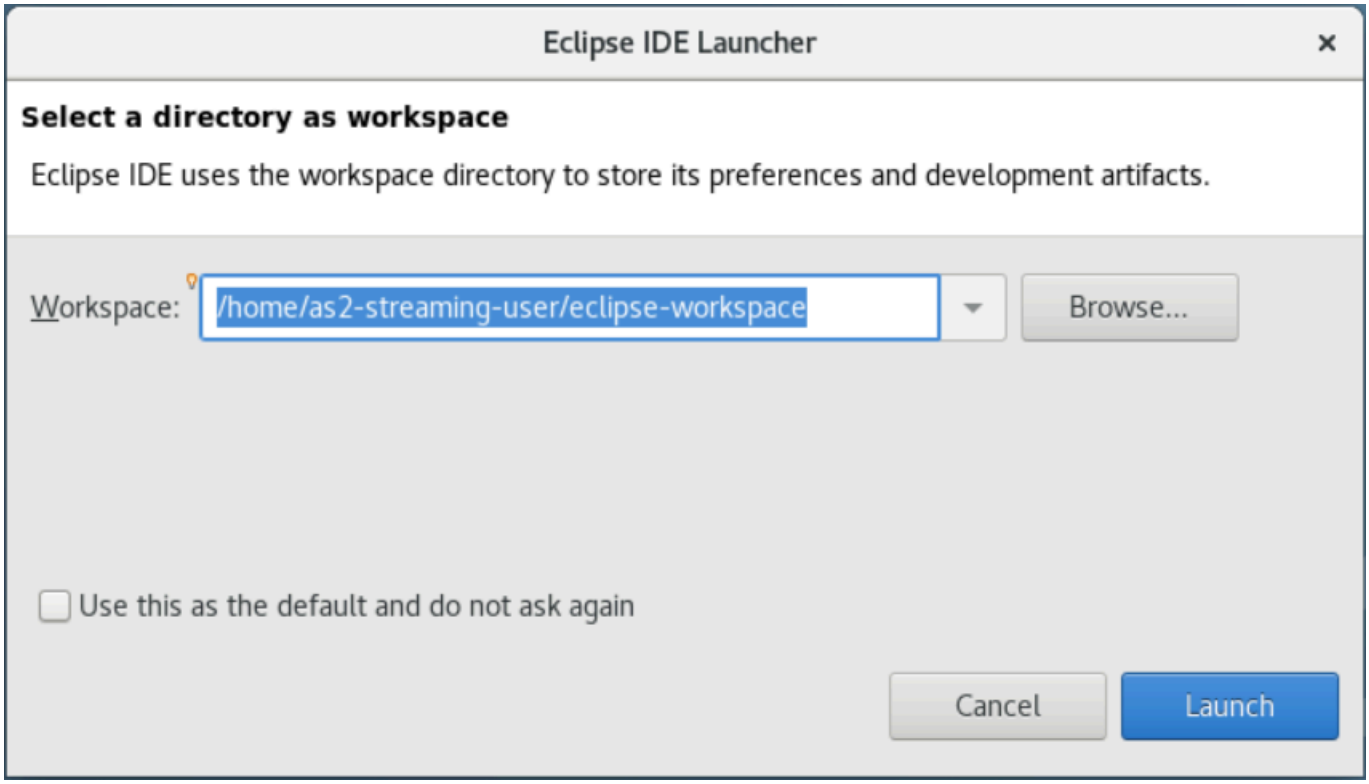
4단계: Eclipse IDE 시작

이 단계에서는 Eclipse IDE를 시작하고 작업 영역을 만들려는 위치를 선택합니다.

1. AppStream 2.0에서는 도구 모음에서 애플리케이션 실행 아이콘을 선택한 다음 Eclipse JEE를 선택합니다.



2. 시작 프로그램이 열리면 작업 영역을 만들려는 위치를 입력하고 시작을 선택합니다.



선택적으로 다음과 같이 명령줄에서 Eclipse를 시작할 수 있습니다.

```
~/eclipse &
```

5단계: Maven 프로젝트 설정

이 단계에서는 Planets 데모 애플리케이션에 사용할 용도로만 Maven 프로젝트를 가져옵니다.

1. PlanetsDemo-pom.zip 파일을 홈 폴더에 업로드하세요. 네이티브 클라이언트 "내 파일" 기능을 사용하여 이 작업을 수행할 수 있습니다.
2. `unzip` 명령줄 도구를 사용하여 파일을 추출합니다.
3. 압축이 풀린 폴더 내부를 탐색하고 텍스트 편집기에서 `pom.xml` 프로젝트의 루트를 엽니다.
4. 설치된 AWS Blu Age 런타임과 일치하도록 `gapwalk.version` 속성을 편집하세요.

   설치된 버전이 확실하지 않은 경우 터미널에서 다음 명령을 실행합니다.

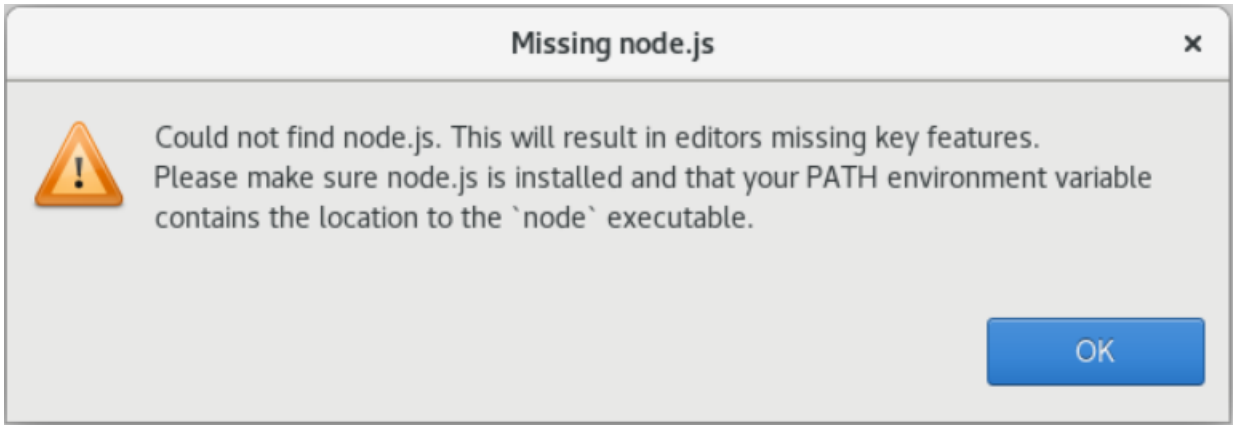
   ```
   cat ~/runtime-version.txt
   ```

   이 명령은 현재 사용 가능한 런타임 버전(예:`3.1.0-b3257-dev`)을 인쇄합니다.

> **Note**
>
> `gapwalk.version`에 `-dev` 접미사를 포함시키지 마세요. 예를 들어 유효한 값은 `<gapwalk.version>3.1.0-b3257</gapwalk.version>`입니다.

5. Eclipse에서 파일을 선택한 다음 가져오기를 선택합니다. 가져오기 대화 상자 창에서 Maven을 확장하고 기존 Maven 프로젝트를 선택합니다. 다음을 선택합니다.
6. Maven 프로젝트 가져오기에서 추출한 파일의 위치를 입력하고 완료를 선택합니다.

   다음 팝업은 무시해도 됩니다. Maven은 프로젝트의 Angular(*-web) 부분을 빌드하기 위해 `node.js`의 로컬 사본을 다운로드합니다.

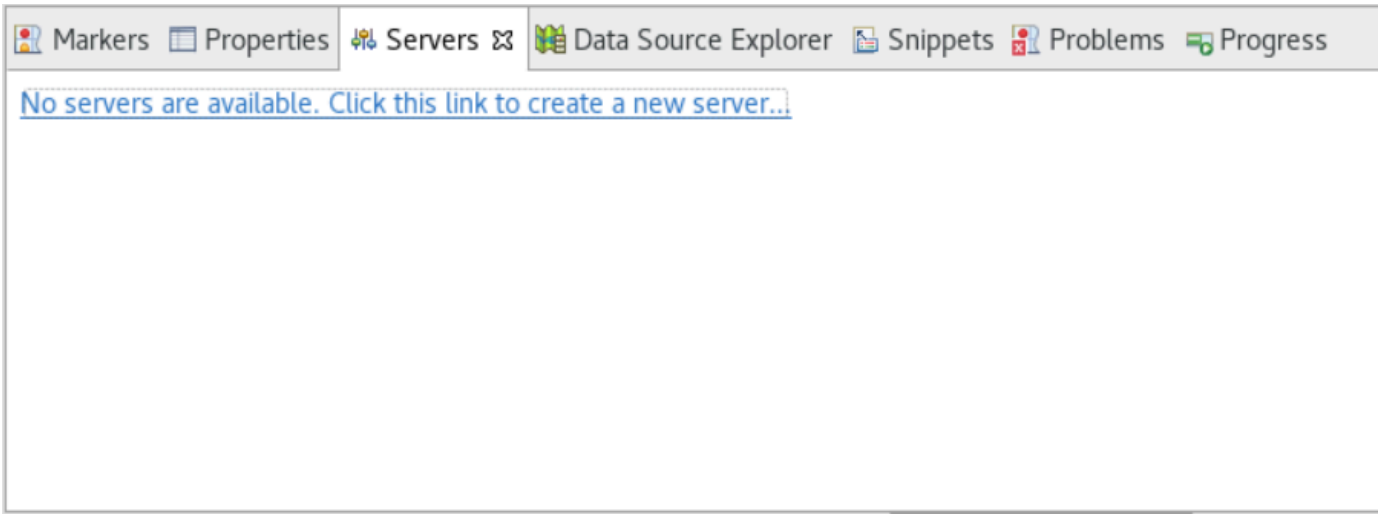


   빌드가 끝날 때까지 기다리세요. 진행률 보기에서 빌드를 팔로우할 수 있습니다.
7. Eclipse에서 프로젝트를 선택하고 다음으로 실행을 선택합니다. 그 다음 Maven 설치를 선택합니다. Maven 설치가 성공하면 `PlanetsDemoPom/PlanetsDemo-web/target/PlanetsDemo-web-1.0.0.war` 아래에 `war` 파일이 생성됩니다.

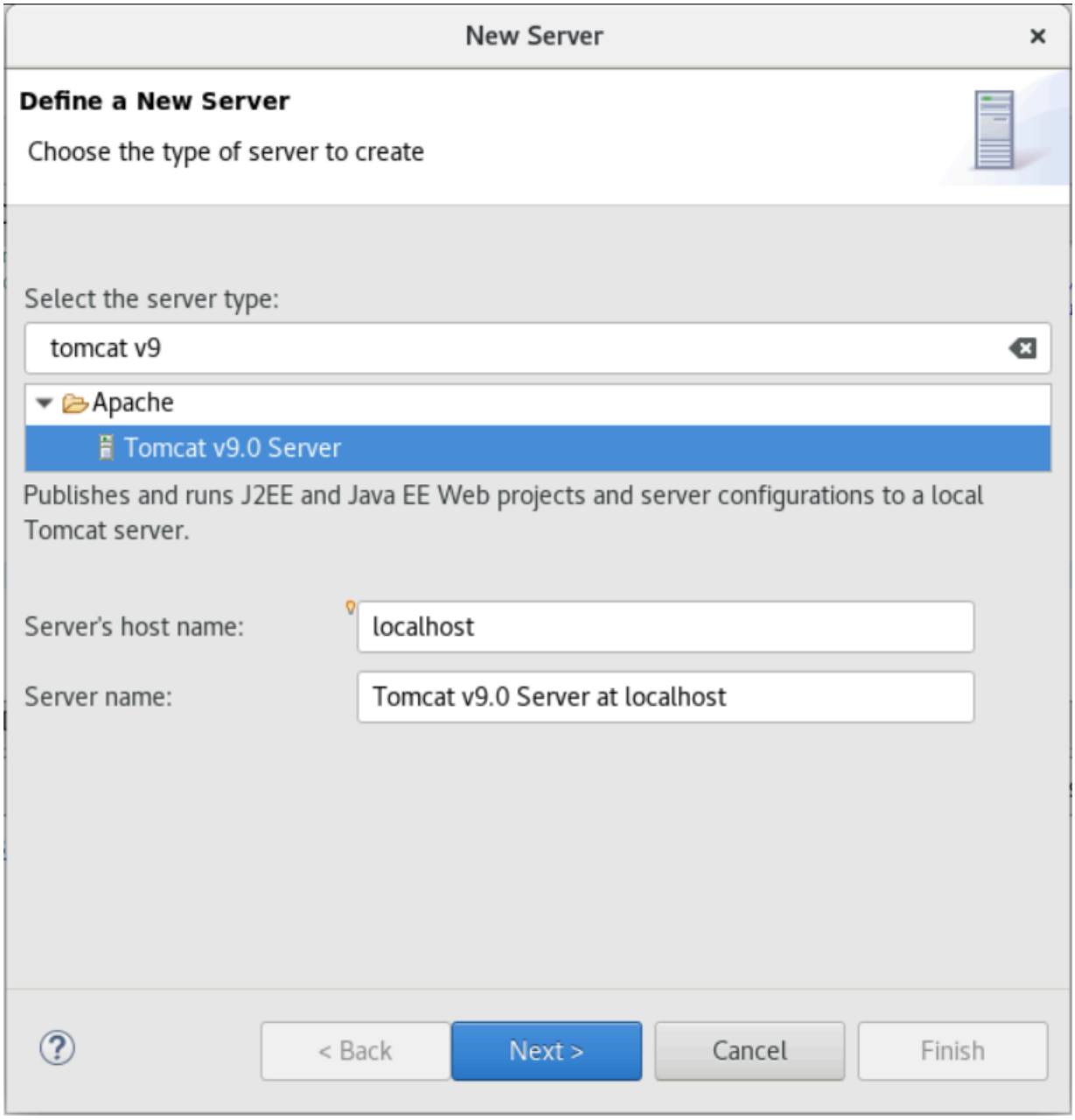
6단계: Tomcat 서버 구성

이 단계에서는 컴파일된 애플리케이션을 배포하고 시작하는 Tomcat 서버를 구성합니다.

1. Eclipse에서 창 > 보기 표시 > 서버를 선택하여 서버 보기를 표시합니다.

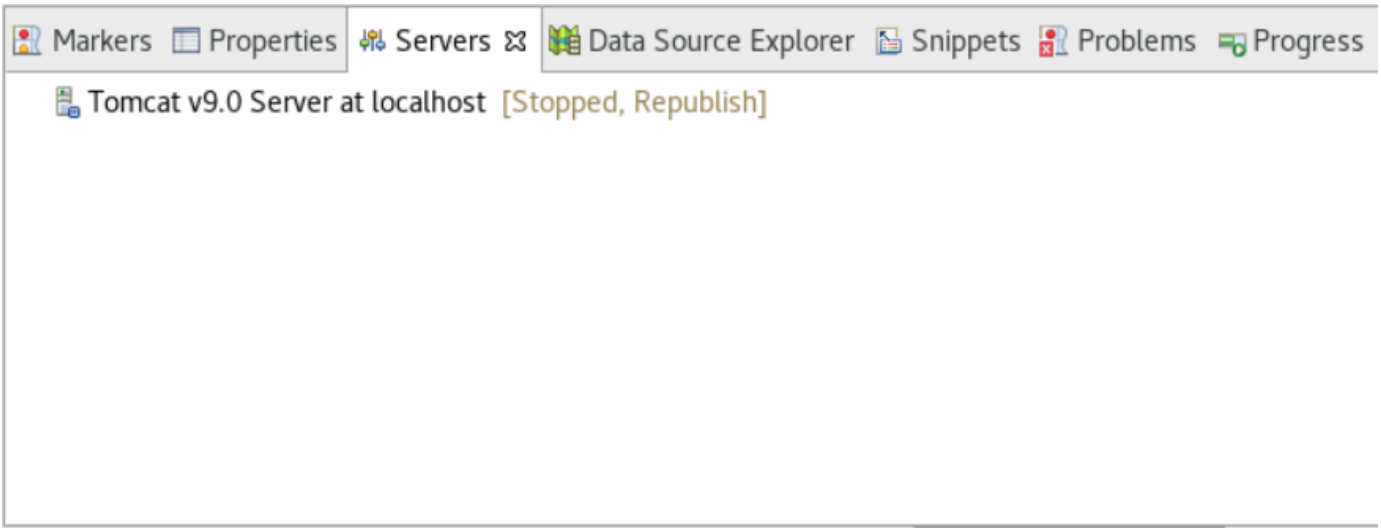


2. 사용할 수 있는 서버 없음을 선택합니다. 새 서버를 만들려면 이 링크를 클릭하세요.... 새 서버 마법사가 나타납니다. 마법사의 서버 유형 선택 필드에 tomcat v9를 입력하고 Tomcat v9.0 서버를 선택합니다. 다음을 선택합니다.



3. 찾아보기를 선택하고 홈 폴더의 루트에 있는 tomcat 폴더를 선택합니다. JRE를 기본값으로 두고 완료를 선택합니다.

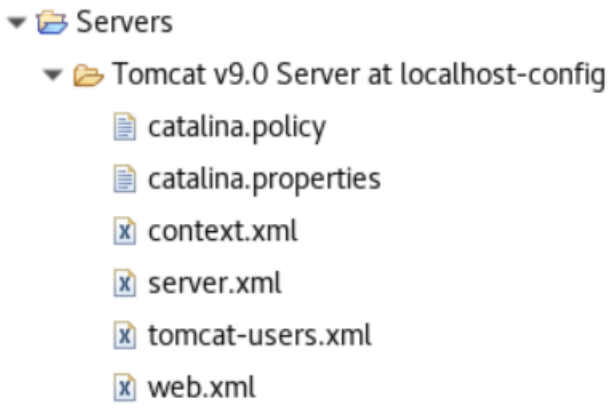
   작업 영역에 서버 프로젝트가 생성되고 이제 서버 보기에서 Tomcat v9.0 서버를 사용할 수 있습니다. 컴파일된 애플리케이션이 배포되고 시작되는 곳은 다음과 같습니다.



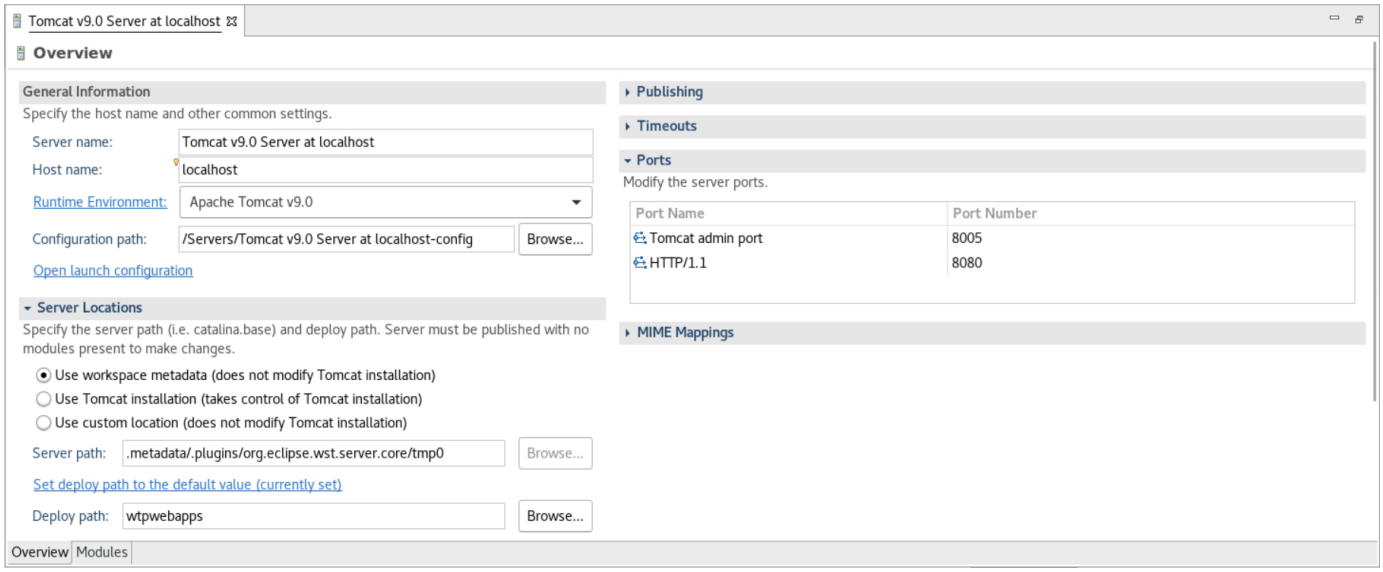
7단계: Tomcat에 배포

이 단계에서는 Planets 데모 애플리케이션을 Tomcat 서버에 배포하여 애플리케이션을 실행할 수 있습니다.

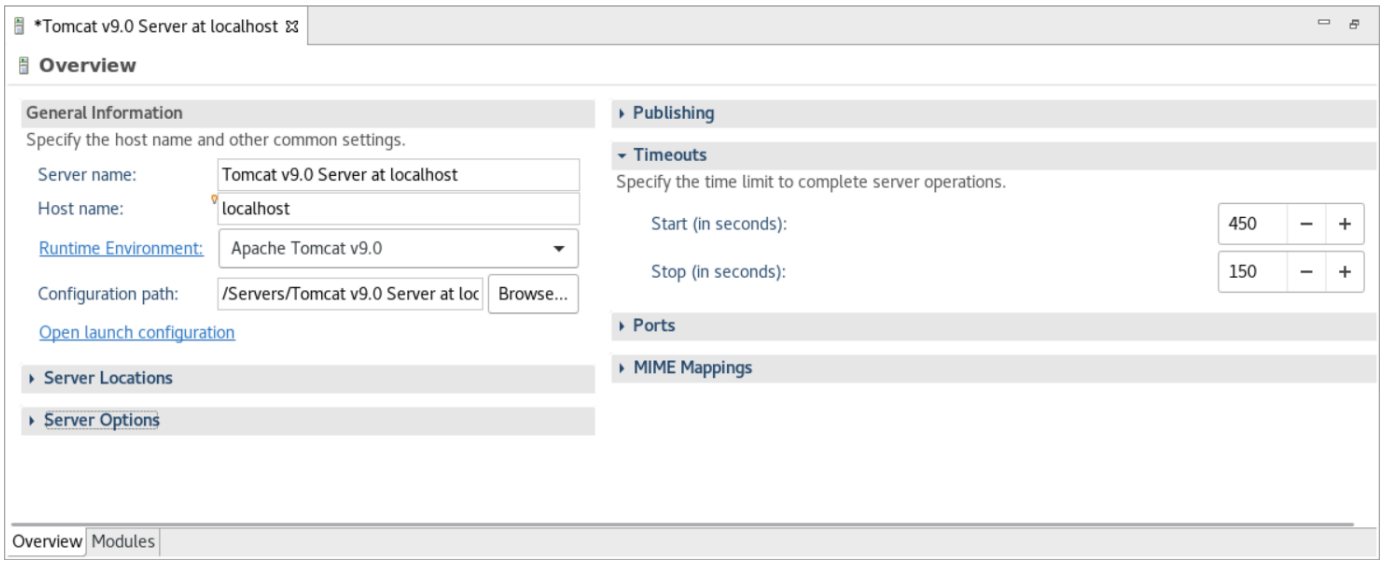
1. `PlanetsDemo-web` 파일을 선택하고 다음으로 실행 > Maven 설치를 선택합니다. npm으로 컴파일된 프론트엔드가.war로 제대로 컴파일되고 Eclipse에서 인식되도록 하려면 `PlanetsDemo-web`을 다시 선택하고 새로 고침을 선택합니다.
2. PlanetsDemo-runtime.zip 파일을 인스턴스에 업로드하고 액세스 가능한 위치에서 파일의 압축을 풉니다. 이렇게 하면 데모 애플리케이션이 필요한 구성 폴더와 파일에 액세스할 수 있습니다.
3. `PlanetsDemo-runtime/tomcat-config`의 내용을 Tomcat 서버용으로 만든 `Servers/Tomcat v9.0...` 하위 폴더에 복사합니다.



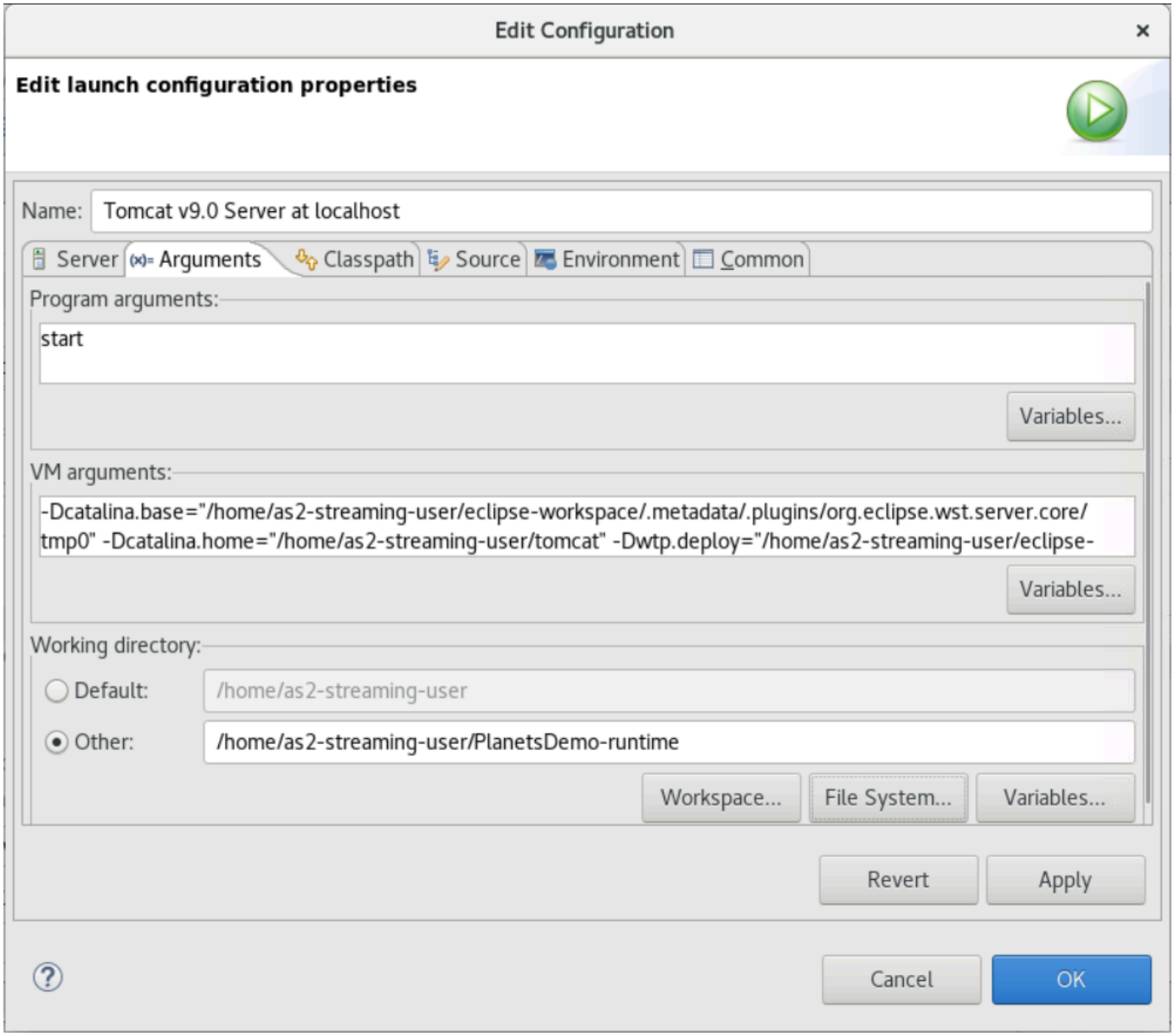
4. `tomcat v9.0` 서버 보기에서 서버 항목을 엽니다. 서버 속성 편집기가 나타납니다.



5. 다음과 같이 개요 탭에서 시간 제한 값을 시작의 경우 450초, 중지의 경우 150초로 늘립니다.



6. 시작 구성 열기를 선택합니다. 그러면 마법사가 표시됩니다. 마법사에서 인수 폴더로 이동한 다음 작업 디렉터리에 대해 기타를 선택합니다. 파일 시스템을 선택하고 이전에 압축을 푼 `PlanetsDemo-runtime` 폴더로 이동합니다. 이 폴더에는 config라는 직접 하위 폴더가 있어야 합니다.

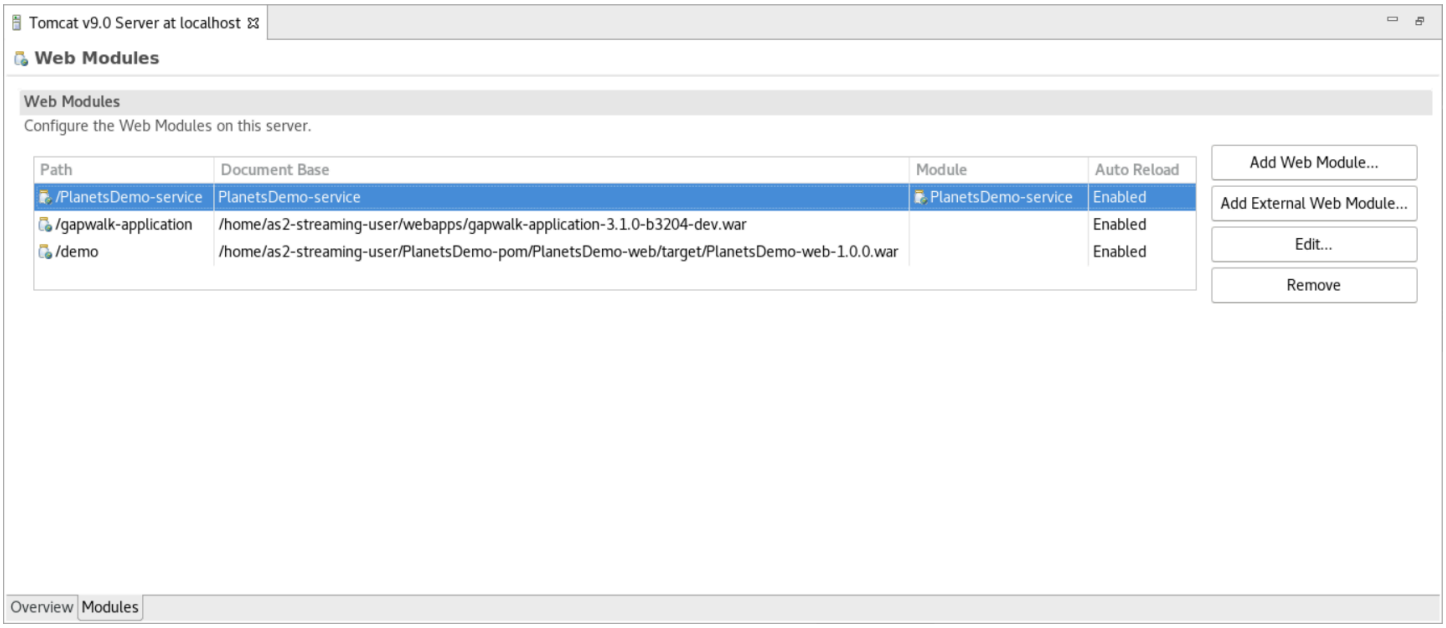


7. 서버 속성 편집기의 모듈 탭을 선택하고 다음과 같이 변경합니다.

   - 웹 모듈 추가를 선택하고 `PlanetsDemo-service`를 추가합니다.
   - 외부 웹 모듈 추가를 선택합니다. 웹 모듈 추가 대화 창이 나타납니다. 다음과 같이 변경합니다.
     - 문서 베이스에서 찾아보기를 선택하고 `~/webapps/gapwalk-application...war`로 이동합니다
     - 경로에 `/gapwalk-application`를 입력합니다.
   - 확인을 선택합니다.
   - 외부 웹 모듈 추가를 다시 선택하고 다음과 같이 변경합니다.

  - 문서 베이스의 경우 프런트엔드 .war(`PlanetsDemo-web/target`에 있음)로 경로를 입력합니다
  - 경로에 `/demo`를 입력합니다
- 확인을 선택합니다.
- 편집기 수정 내용을 저장합니다(Ctrl + S).

편집기 내용은 다음과 비슷해야 합니다.



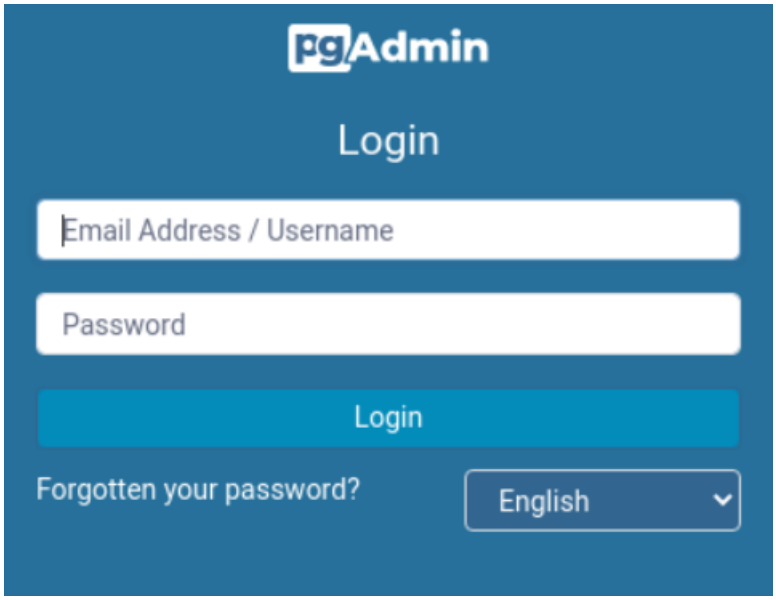
8단계: JICS 데이터베이스 생성

이 단계에서는 1단계: 데이터베이스 생성에서 생성한 데이터베이스에 연결합니다.

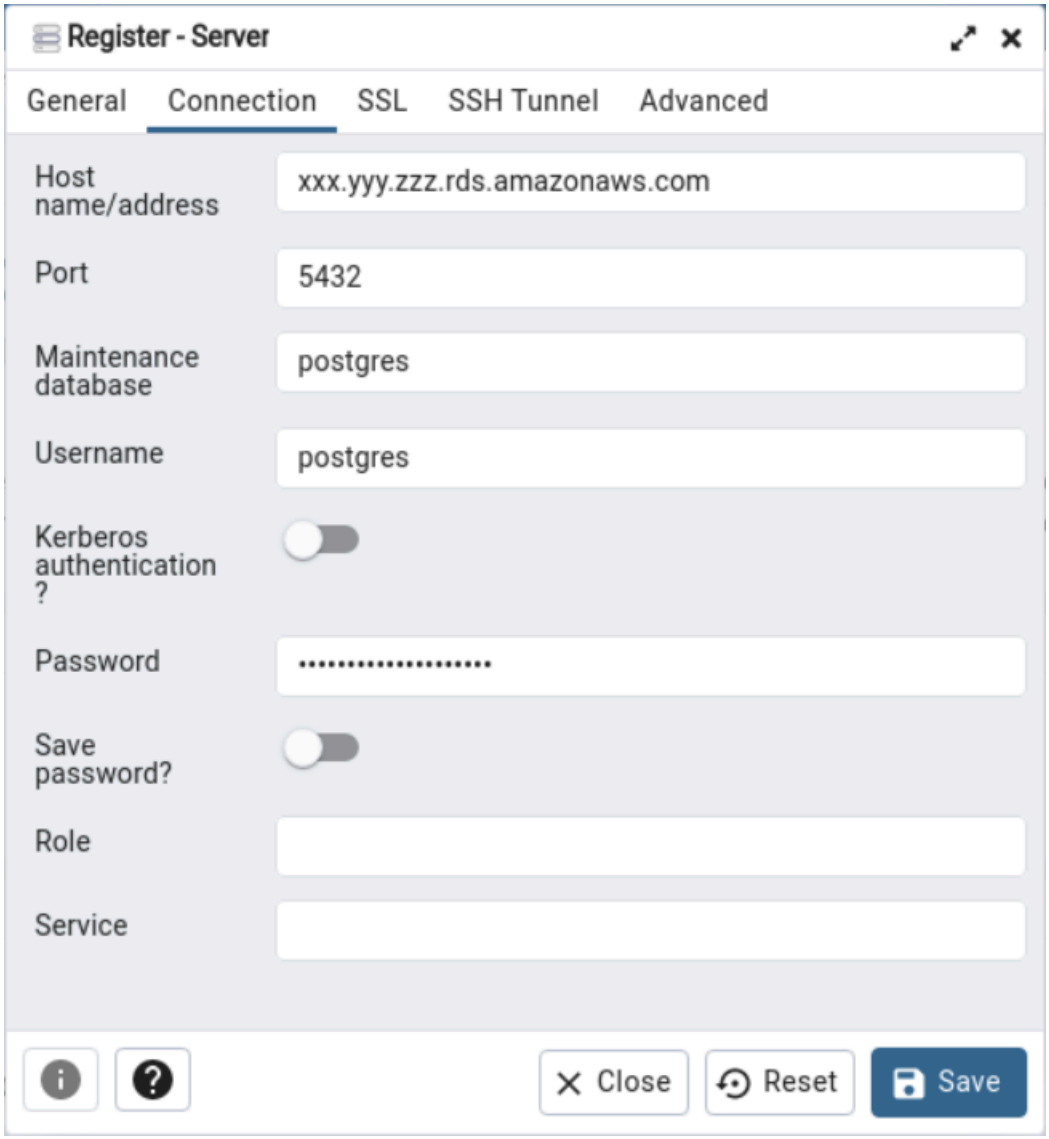
1. AppStream 2.0 인스턴스에서 터미널에서 다음 명령을 실행하여 실행합니다`pgAdmin`.

   ```
   ./pgadmin-start.sh
   ```

2. 이메일 주소와 암호를 로그인 식별자로 선택하세요. 제공된 URL(일반적으로 http://127.0.0.1:5050)을 기록해 둡니다. 인스턴스에서 Google Chrome을 실행하고 URL을 복사하여 브라우저에 붙여넣은 다음 식별자를 사용하여 로그인합니다.



3. 로그인한 후 신규 서버 추가를 선택하고 다음과 같이 이전에 만든 데이터베이스에 대한 연결 정보를 입력합니다.

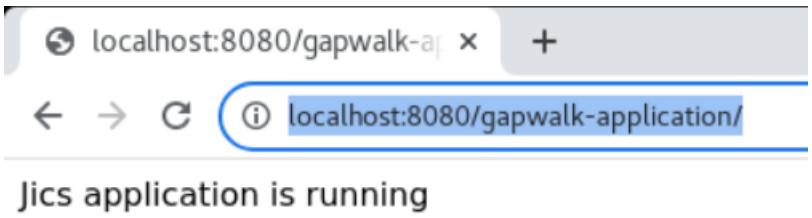


4. 데이터베이스 서버에 연결할 때 개체 > 만들기 > 데이터베이스를 사용하여 jics라는 새 데이터베이스를 생성합니다.
5. 데모 앱에서 사용한 데이터베이스 연결 정보를 편집합니다. 이 정보는 `PlanetsDemo-runtime/config/application-main.yml`에 정의되어 있습니다. `jicsDs` 항목을 검색합니다. Amazon RDS 콘솔에서 `username` 및 `password`에 대한 값을 검색하고 해당 데이터베이스로 이동하려면 해당 데이터베이스로 이동하세요. 구성 탭의 Master Credentials ARN에서 Secrets Manager에서 관리를 선택합니다. 그런 다음 Secrets Manager 콘솔의 암호에서 암호 값 검색을 선택합니다.

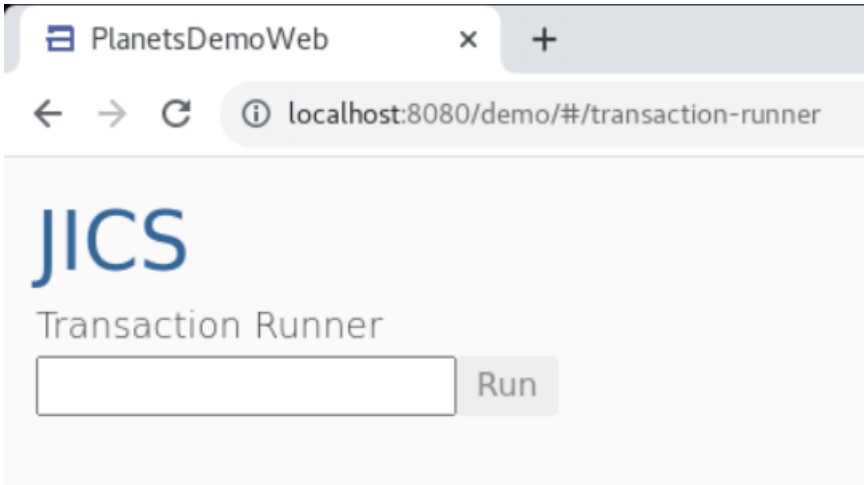
8단계: 애플리케이션 시작 및 테스트

이 단계에서는 테스트할 수 있도록 Tomcat 서버와 데모 애플리케이션을 시작합니다.

1. Tomcat 서버와 이전에 배포한 애플리케이션을 시작하려면 서버 보기에서 서버 항목을 선택하고 시작을 선택합니다. 시작 로그를 표시하는 콘솔이 나타납니다.
2. 서버 보기에서 서버 상태를 확인하거나 콘솔에서 [xxx] 밀리초 내에 서버 시작 메시지가 표시될 때까지 기다리세요. 서버가 시작된 후 gapwalk-applicatio이 제대로 배포되었는지 확인합니다. 이렇게 하려면 Google Chrome 브라우저에서 http://localhost:8080/gapwalk-application URL에 액세스하세요. 다음과 같은 모양이어야 합니다.



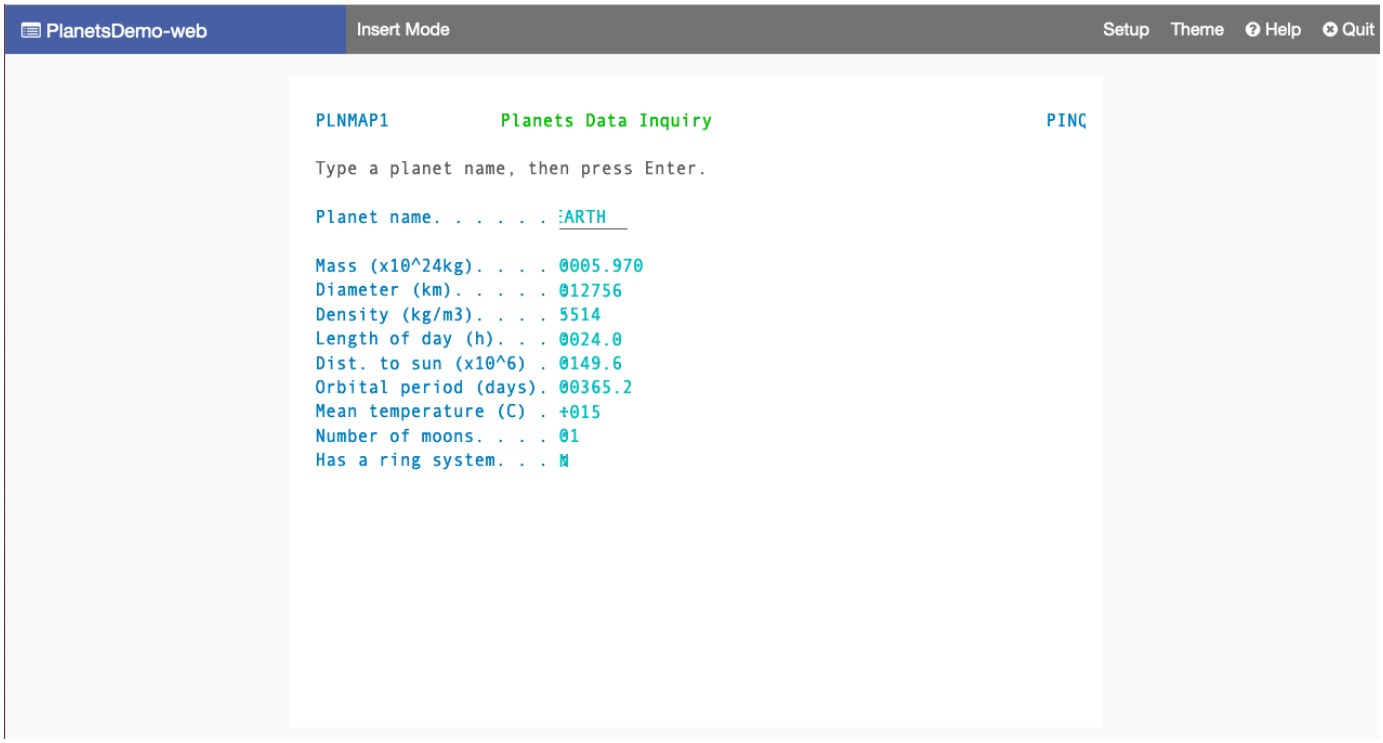
3. Google Chrome의 http://localhost:8080/demo에서 배포된 애플리케이션 프론트엔드에 액세스할 수 있습니다. 다음 트랜잭션 시작 페이지가 표시되어야 합니다.



4. 애플리케이션 트랜잭션을 시작하려면 입력 필드에 `PINQ`를 입력하고 실행을 선택하거나 Enter 키를 누릅니다.

   데모 앱 화면이 나타나야 합니다.

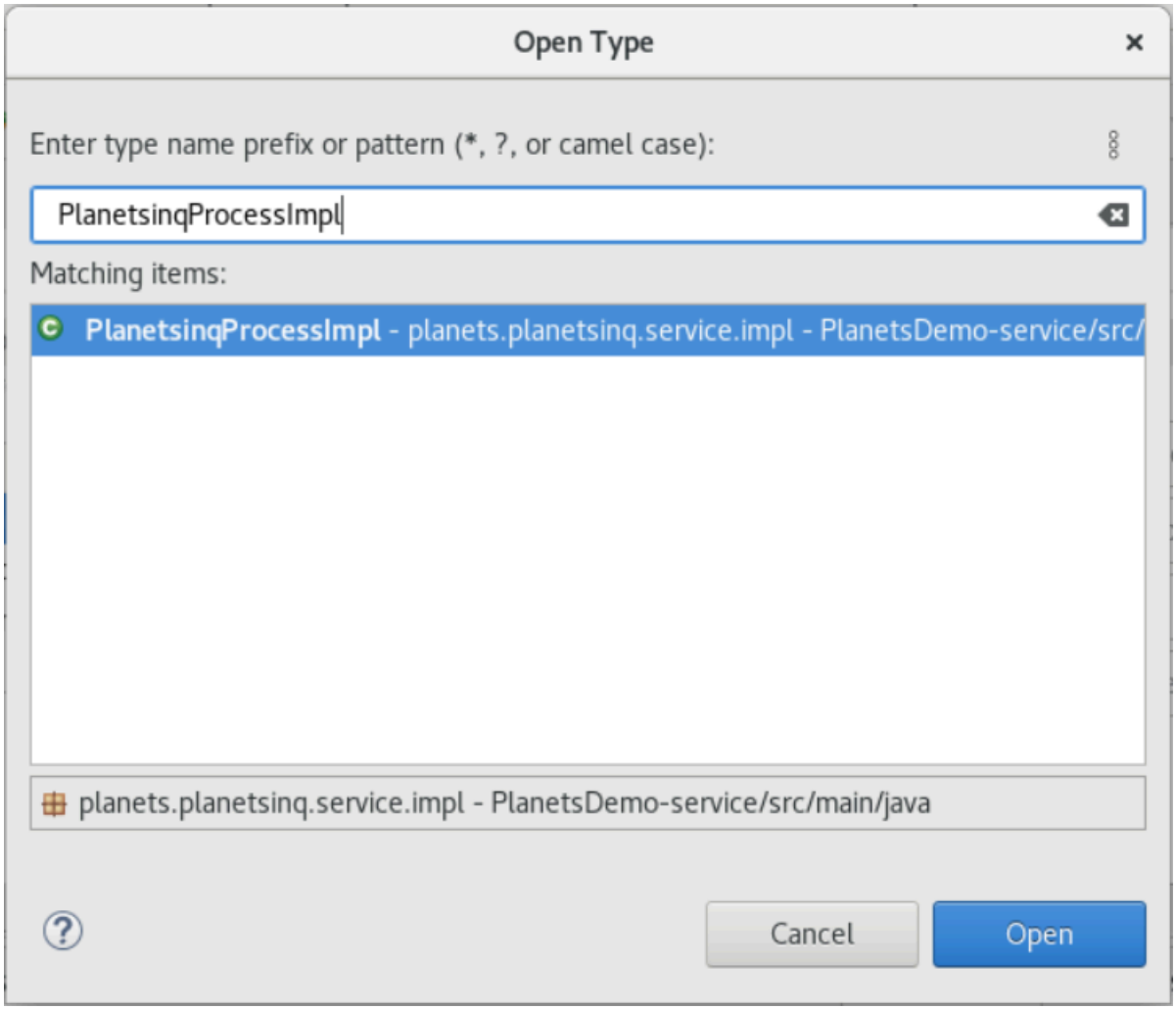
5. 해당 필드에 행성 이름을 입력하고 Enter 키를 누릅니다.



10단계: 애플리케이션 디버그

이 단계에서는 표준 Eclipse 디버깅 기능을 사용하여 테스트합니다. 이러한 기능은 현대화된 애플리케이션에서 작업할 때 사용할 수 있습니다.

1. 기본 서비스 클래스를 열려면 Ctrl + Shift + T를 누른 다음 `PlanetsinqProcessImpl`를 입력합니다.



2. `searchPlanet` 메서드로 이동한 다음 중단점을 입력하세요.
3. 서버 이름을 선택하고 디버그에서 재시작을 선택합니다.
4. 그런 다음 이전 단계를 반복합니다. 즉, 애플리케이션에 접속하여 행성 이름을 입력하고 Enter 키를 누릅니다.

   Eclipse는 `searchPlanet` 메서드에서 애플리케이션을 중지합니다. 이제 검사할 수 있습니다.

리소스 정리

이 자습서에 사용할 때 생성한 리소스가 더 이상 필요하지 않은 경우 삭제하여 추가 요금이 발생하지 않도록 하세요. 다음 단계를 완료합니다.

- Planets 애플리케이션이 아직 실행 중이면 중지하세요.
- 1단계: 데이터베이스 생성에서 생성한 데이터베이스를 삭제합니다. 자세한 내용은 DB 인스턴스 삭제를 참조하세요.

Micro Focus를 통한 애플리케이션 리플랫포밍

이 섹션에서는 리플랫포밍 프로세스의 각 단계를 설명합니다. 여기에는 모든 작업에 대해 설명하고 Amazon EC2의 AWS 메인프레임 현대화 런타임을 구성하고 운영하는 데 대한 정보가 포함되어 있습니다.

주제

Micro Focus Runtime(Amazon EC2) 설정

AWS 메인프레임 현대화는 Micro Focus 라이선스 제품이 포함된 여러 Amazon 머신 이미지 (AMI) 를 제공합니다. 이러한 AMI를 사용하면 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 신속하게 프로비저닝하여 제어 및 관리하는 Micro Focus 환경을 지원할 수 있습니다. 이 주제에서는 이러한 AMI에 액세스하고 시작하는 데 필요한 단계를 제공합니다. 이러한 AMI를 사용하는 것은 전적으로 선택 사항이며 이 사용 설명서의 자습서를 완료하는 데 반드시 필요한 것은 아닙니다.

주제

- License Manager에 필수 권한을 부여하세요
- Amazon Machine Images를 구독하세요
- AWS 메인프레임 현대화 마이크로 포커스 인스턴스 시작
- 인터넷에 액세스할 수 없는 서브넷 또는 VPC
- 라이선스 문제 해결

필수 조건

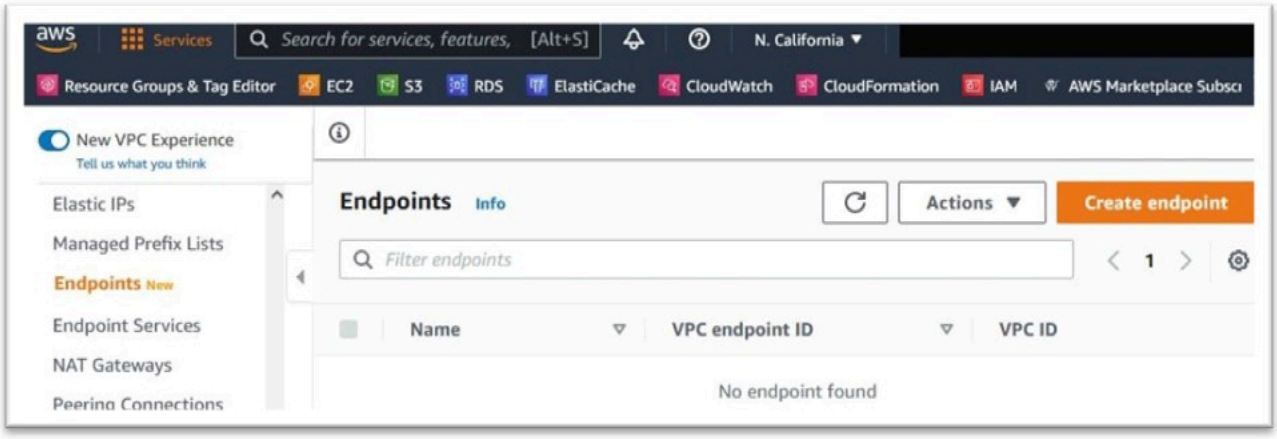
다음 사전 조건을 충족하는지 확인합니다.

- Amazon EC2 인스턴스가 생성될 계정에 대한 관리자 액세스가 생성됩니다.
- Amazon EC2 인스턴스가 생성될 AWS 리전 위치를 식별하고 AWS 메인프레임 현대화 서비스를 사용할 수 있는지 확인합니다. AWS 리전별 서비스를 참조하세요. 서비스가 제공되는 리전을 선택했는지 확인합니다.
- Amazon EC2 인스턴스가 생성될 Amazon Virtual Private Cloud(Amazon VPC)를 식별하세요.

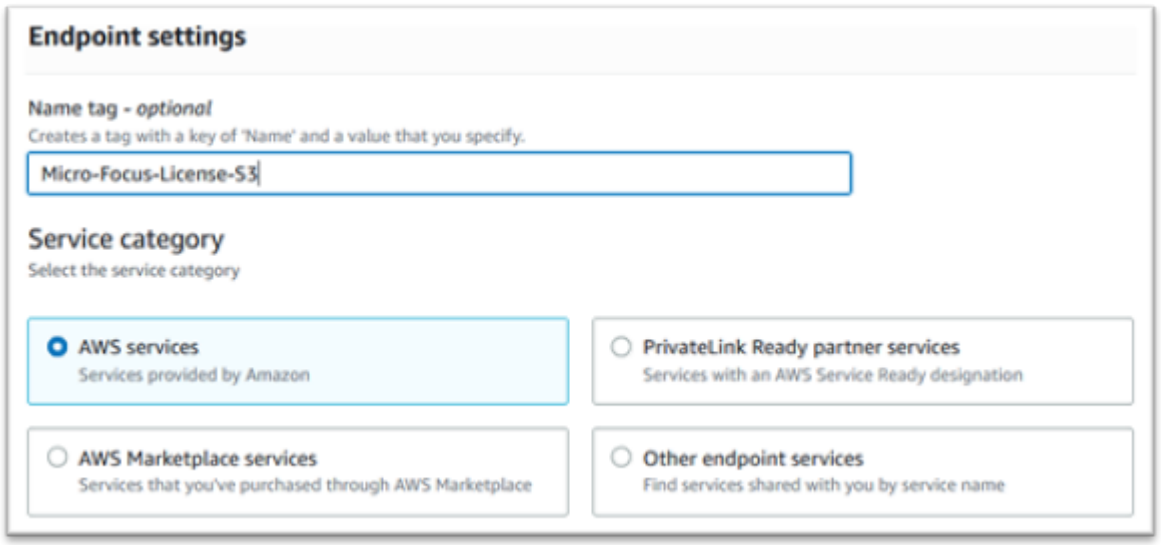
Amazon S3용 Amazon VPC 엔드포인트

이 섹션에서는 Amazon S3용 Amazon VPC 엔드포인트를 생성합니다.

1. AWS Management Console에서 Amazon VPC로 이동합니다.
2. 탐색 창에서 엔드포인트를 선택합니다.
3. Create endpoint(엔드포인트 생성)을 선택합니다.



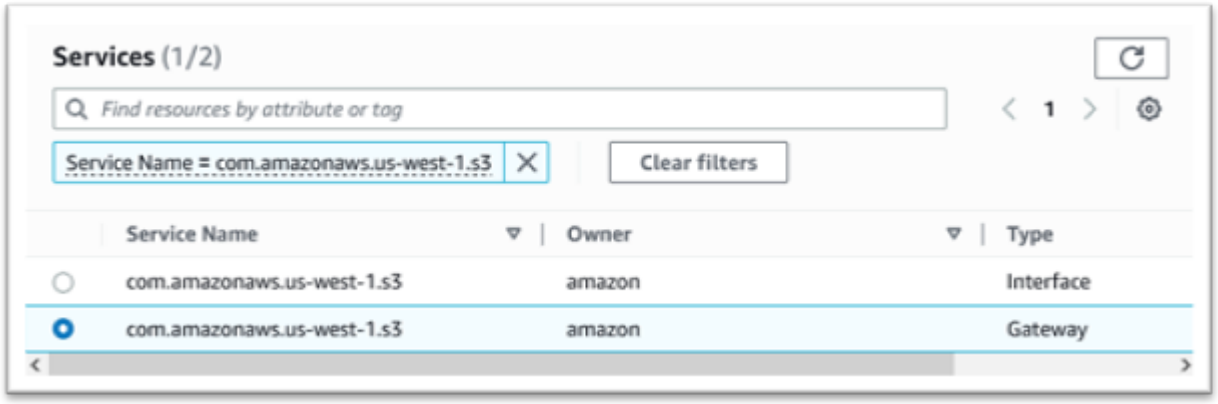
4. 의미 있는 이름 태그를 입력합니다(예: “Micro-Focus-License-S3”).
5. 서비스 카테고리에서 AWS 서비스를 선택합니다.



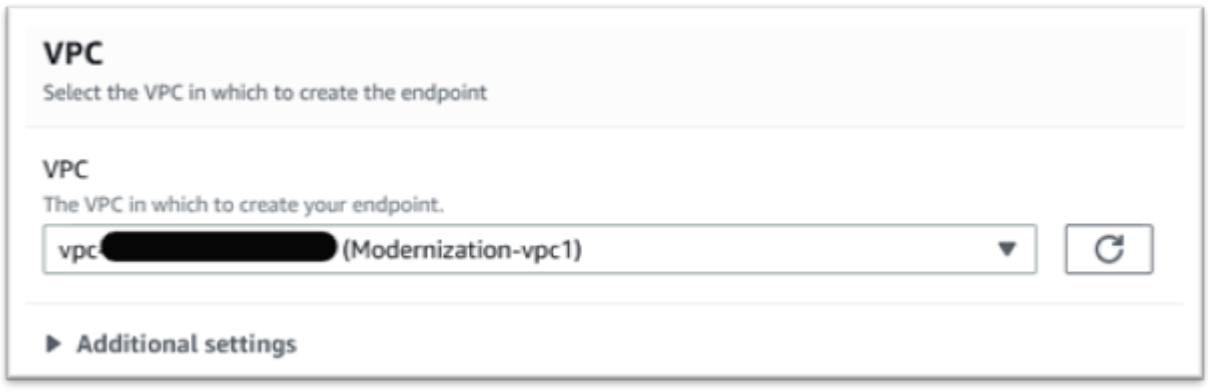
6. 서비스에서 Amazon S3 Gateway인 com.amazonaws.[region].s3를 찾습니다.

   us-west-1에 대해 com.amazonaws.us-west-1.s3과 같을 것입니다.

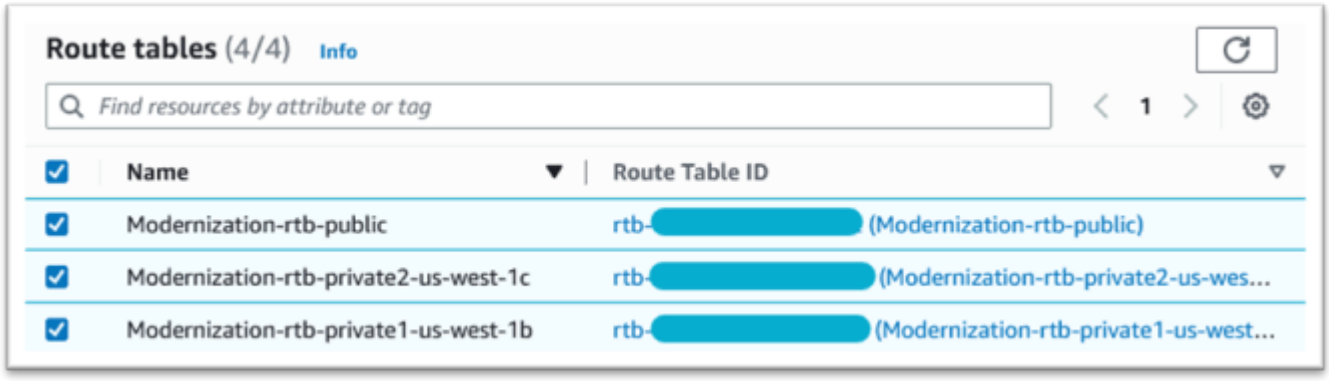
7. 게이트웨이 서비스를 선택합니다.



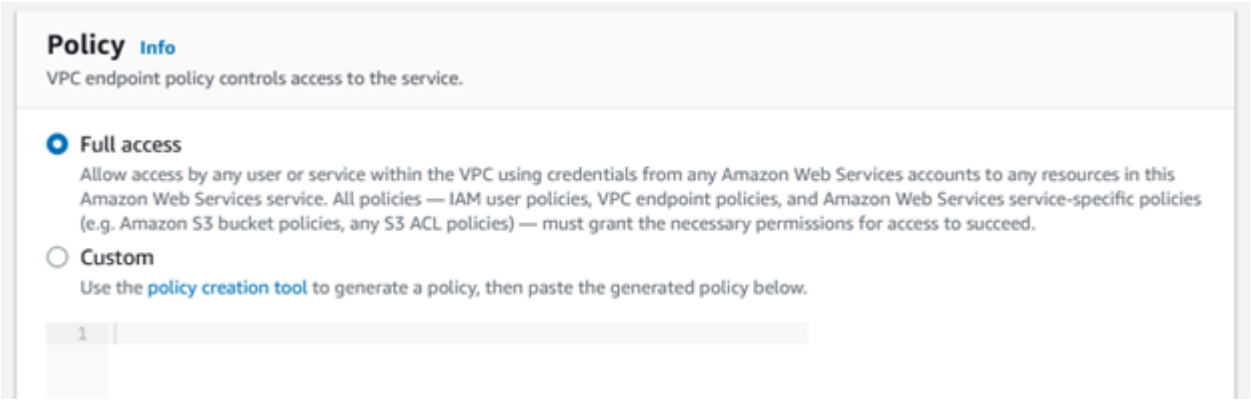
8. VPC의 경우 사용할 VPC를 선택합니다.



9. VPC의 모든 라우팅 테이블을 선택합니다.



10. 정책에서 전체 액세스를 선택합니다.



11. 엔드포인트 생성을 선택합니다.

계정에 대한 허용 목록 업데이트 요청

AWS 담당자와 상의하여 계정을 메인프레임 현대화 AMI의 허용 목록에 올리십시오. AWS 다음 정보를 제공합니다.

- ID. AWS 계정
- Amazon VPC 엔드포인트가 생성된 AWS 리전 위치입니다.
- Amazon S3용 Amazon VPC 엔드포인트에서 생성된 Amazon VPC Amazon S3 엔드포인트 ID. 이는 com.amazonaws.[region].s3 Gateway 엔드포인트트의 vpce-xxxxxxxxxxxxxxxxx ID입니다.
- 모든 Micro Focus Enterprise Suite AMI Amazon EC2 인스턴스에 필요한 라이선스 수입니다.

  CPU 코어당 하나의 라이선스가 필요합니다(대부분의 Amazon EC2 인스턴스의 경우 vCPU 2개당).

  자세한 정보는 CPU 옵션 최적화를 참조하세요.

  요청된 번호는 나중에 다음을 통해 조정할 수 AWS있습니다.

> **Note**
>
> AWS 담당자는 Allowlist 요청에 대한 지원 티켓을 열어야 합니다. 직접 요청할 수 없으며 요청을 완료하는 데 며칠이 걸릴 수 있습니다.

역할 생성 AWS Identity and Access Management

AWS 메인프레임 현대화 Amazon EC2 인스턴스에서 사용할 AWS Identity and Access Management 정책과 역할을 생성합니다. IAM 콘솔을 통해 역할을 생성하면 동일한 이름의 관련 인스턴스 프로파일이 생성됩니다. 이 인스턴스 프로파일을 Amazon EC2 인스턴스에 할당하면 Micro Focus 라이선스를 할당할 수 있습니다. 자세한 내용은 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

IAM 정책 만들기

IAM 정책이 먼저 생성되며 그 다음 역할에 연결됩니다.

1. 으로 이동하십시오. AWS Identity and Access Management AWS Management Console
2. 정책을 선택한 다음 정책 만들기를 선택합니다.

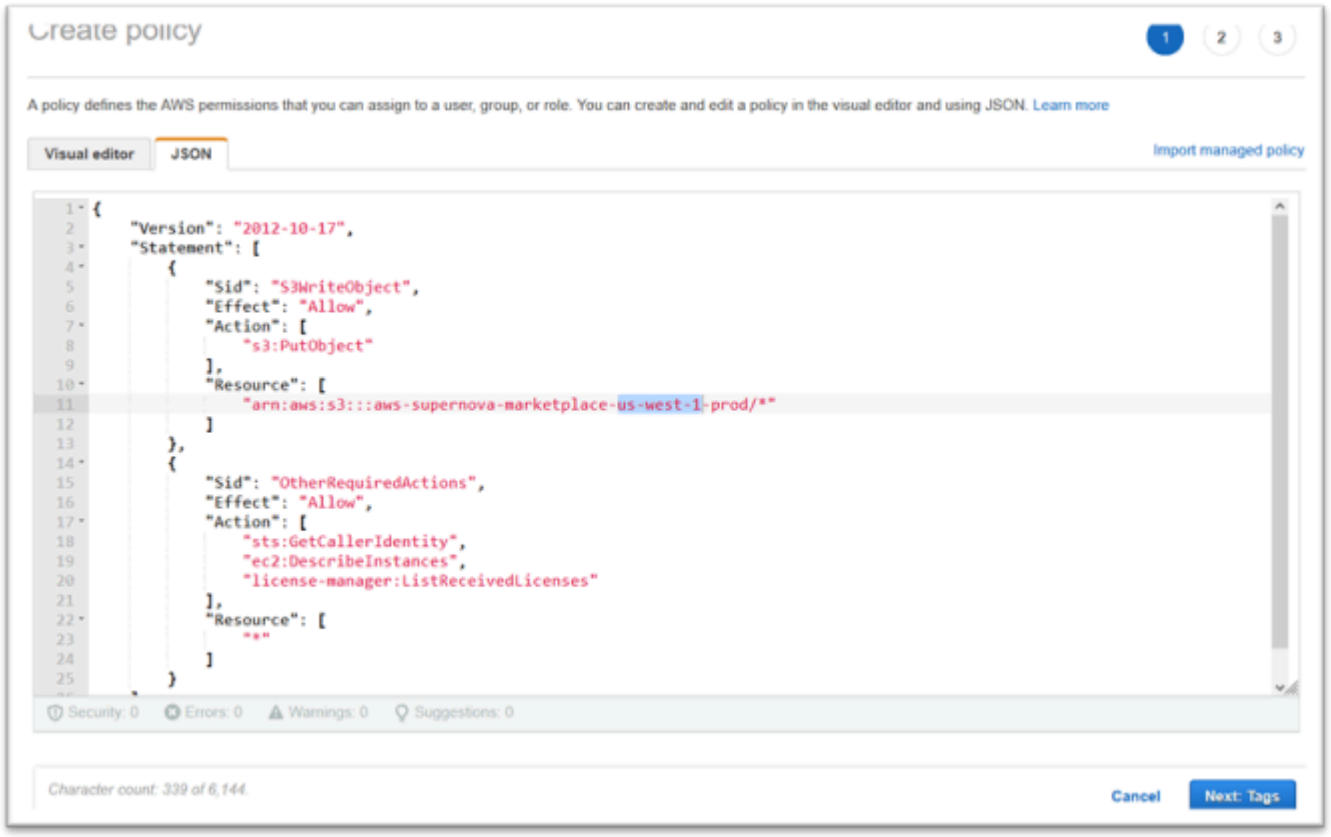
3. JSON 탭을 선택합니다.

4. 다음 JSON을 Amazon S3 엔드포인트가 정의된 AWS 리전 위치로 바꾸고 `us-west-1` JSON을 복사하여 정책 편집기에 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3WriteObject",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-supernova-marketplace-us-west-1-prod/*"
            ]
        },
        {
            "Sid": "OtherRequiredActions",
            "Effect": "Allow",
            "Action": [
                "sts:GetCallerIdentity",
                "ec2:DescribeInstances",
                "license-manager:ListReceivedLicenses"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
```

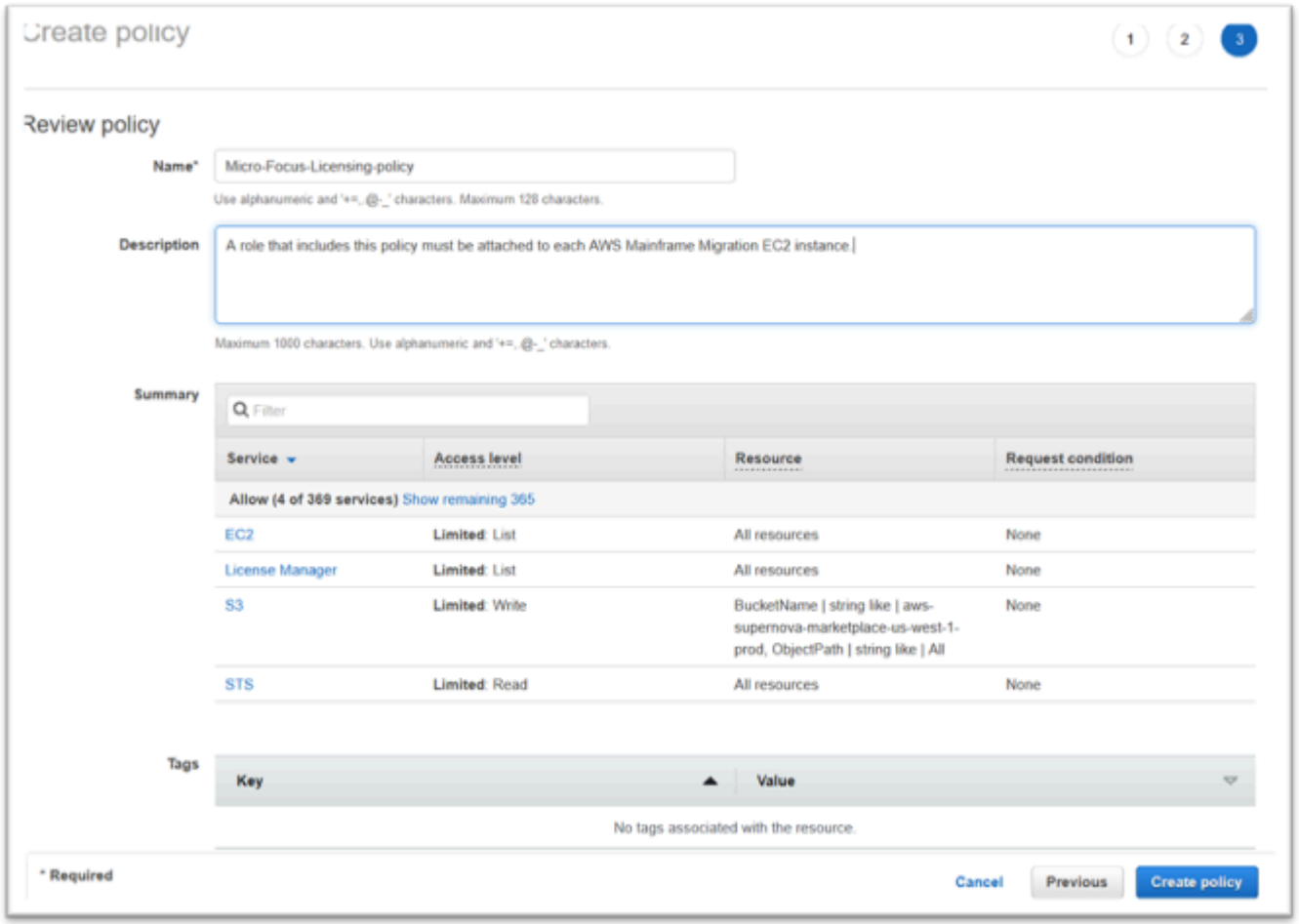
```
}
```

> **Note**
>
> Sid `OtherRequiredActions` 아래의 작업은 리소스 수준 권한을 지원하지 않기 때문에 리소스 요소에 `*`를 지정해야 합니다.



5. 다음: 태그를 선택합니다.

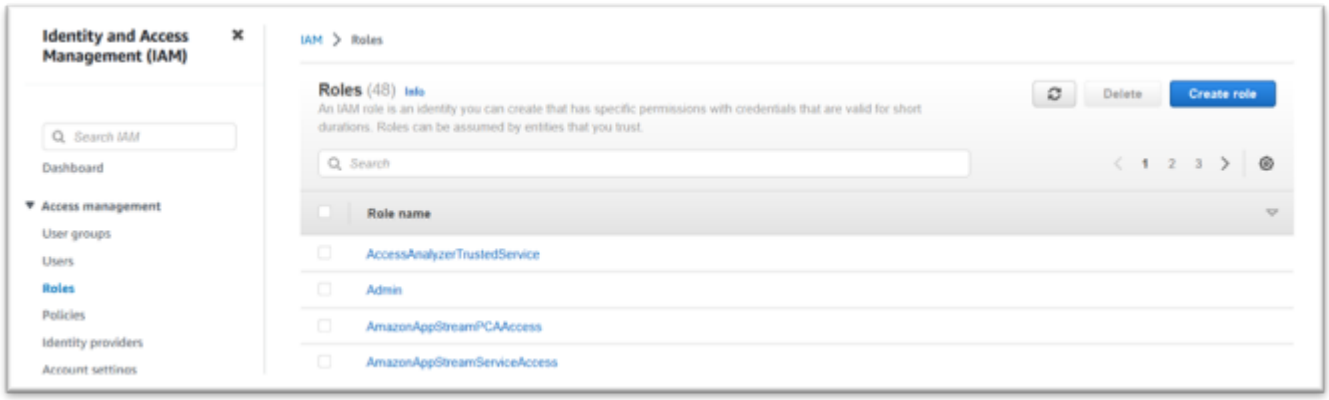
6. 원하는 경우 태그를 입력하고 다음: 검토를 선택합니다.
7. 정책 이름을 입력합니다(예: “Micro Focus 라이선싱 정책”). 선택적으로 설명을 입력합니다 (예: “이 정책이 포함된 역할을 각 AWS 메인프레임 현대화 Amazon EC2 인스턴스에 연결해야 합니다.”).



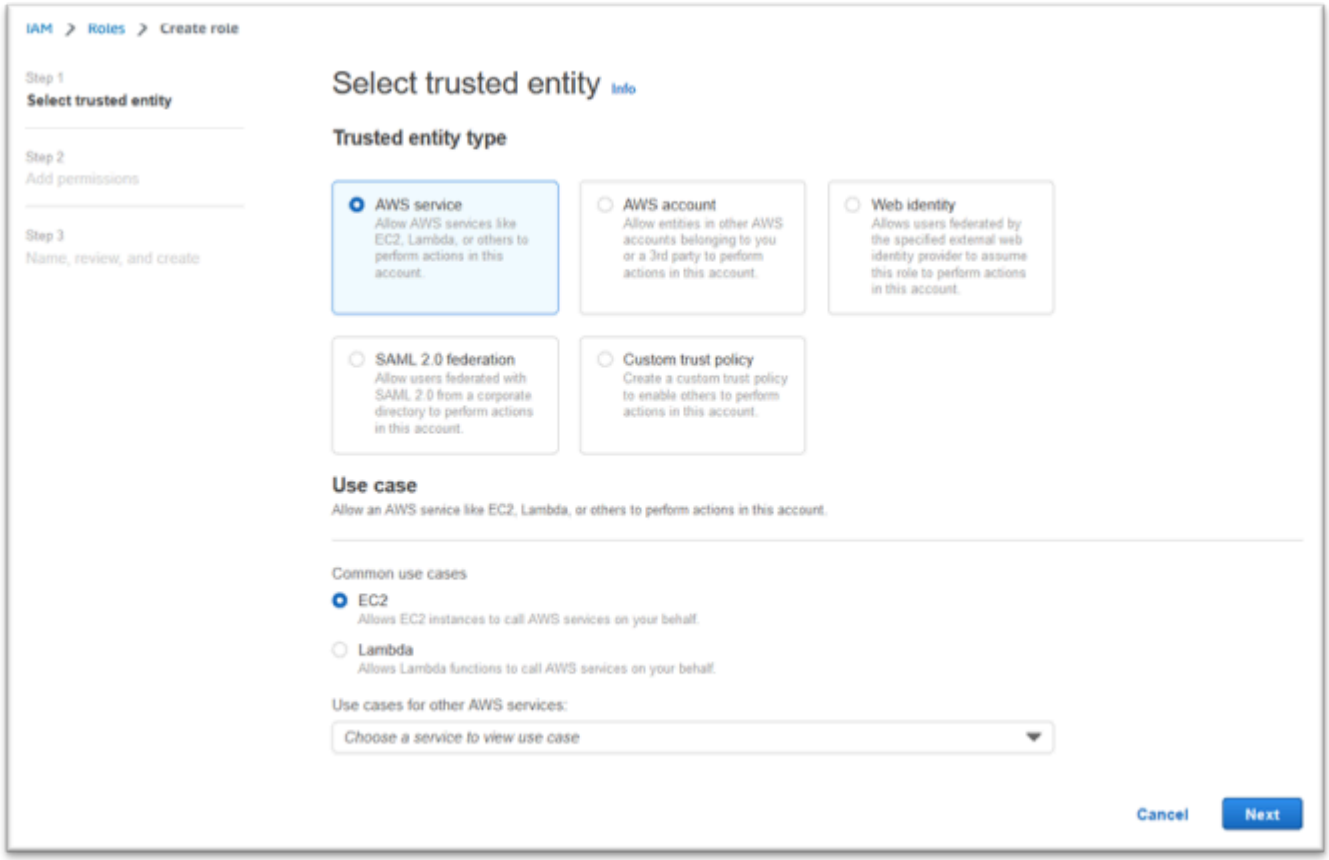
8. 정책 생성을 선택합니다.

IAM 역할 생성

1. AWS Management Console에서 IAM으로 이동합니다.
2. 역할을 선택한 다음 역할 생성을 선택합니다.

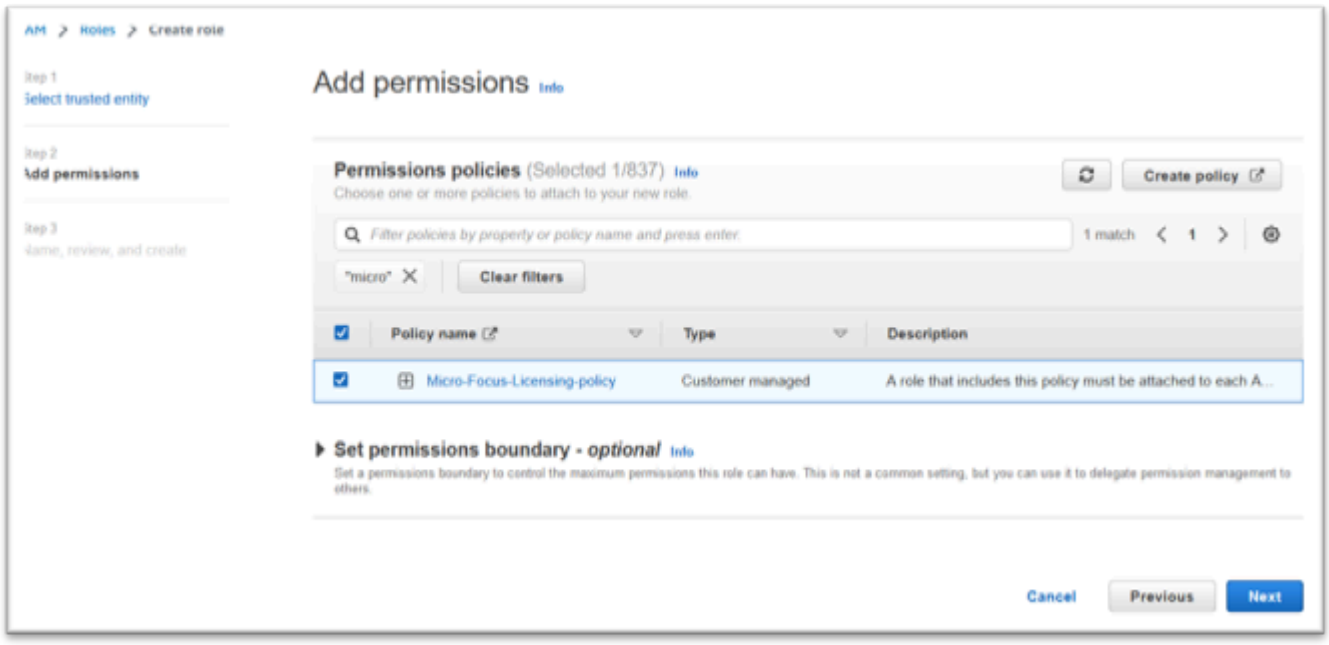


3. 신뢰할 수 있는 엔티티 유형은 AWS 서비스로 두고 EC2 일반 사용 사례를 선택하세요.

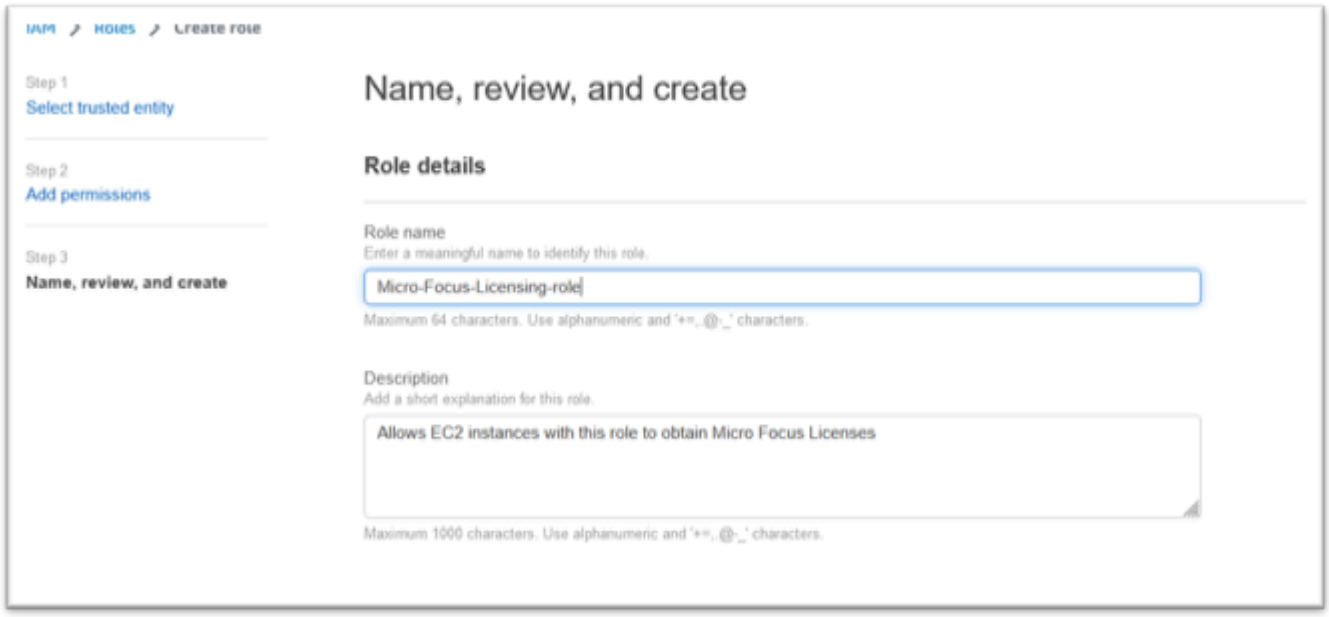


4. 다음을 선택합니다.
5. 필터에 “Micro”를 입력하고 Enter 키를 눌러 필터를 적용합니다.
6. 방금 생성한 정책을 선택합니다(예: “Micro Focus 라이선스 정책”).

7. 다음을 선택합니다.



8. 역할 이름을 입력합니다(예: “Micro Focus 라이선싱 역할”).
9. 설명을 자신의 설명으로 바꾸세요(예: “이 역할을 가진 Amazon EC2 인스턴스에서 Micro Focus 라이선스를 취득할 수 있도록 허용”).



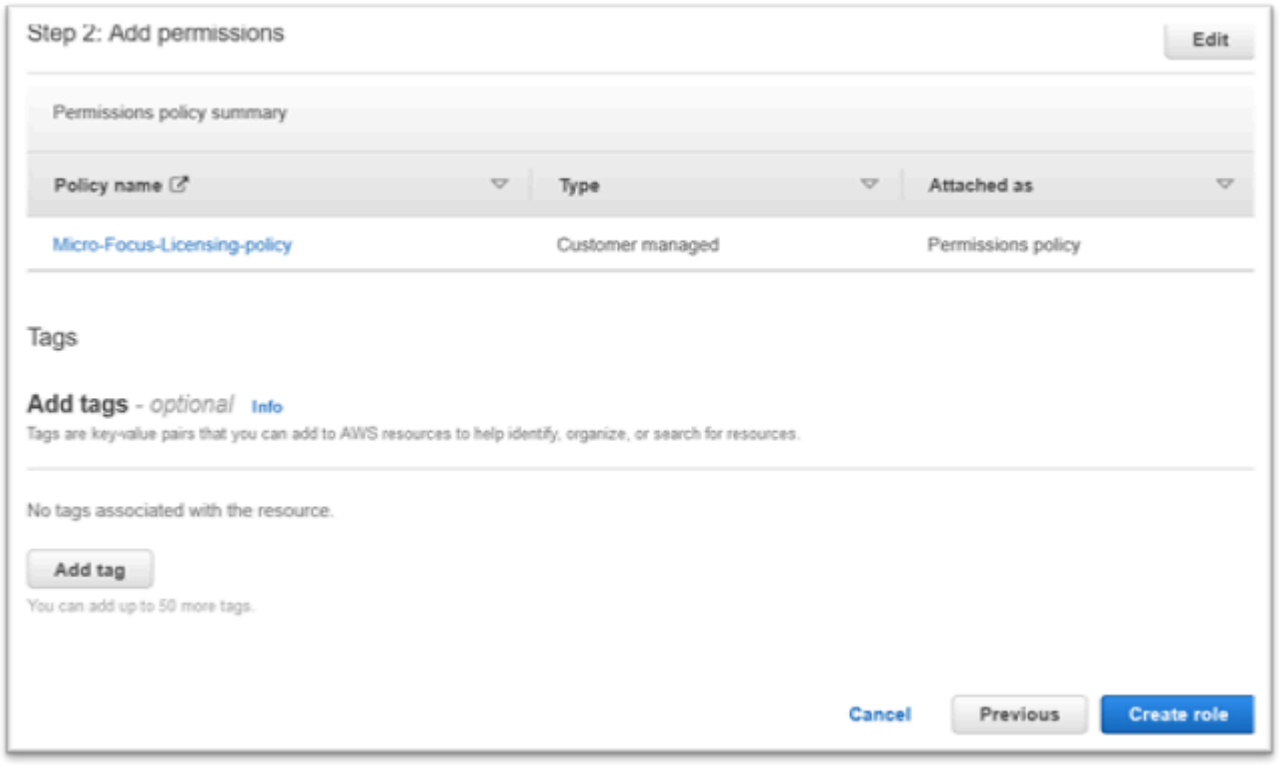
10. 1단계: 신뢰할 수 있는 엔티티 선택에서 JSON을 검토하고 JSON에 다음 값이 있는지 확인합니다.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

> **Note**
>
> 효과, 조치, 주체의 순서는 중요하지 않습니다.

11. 2단계: 권한 추가에 라이선스 정책이 표시되는지 확인하세요.

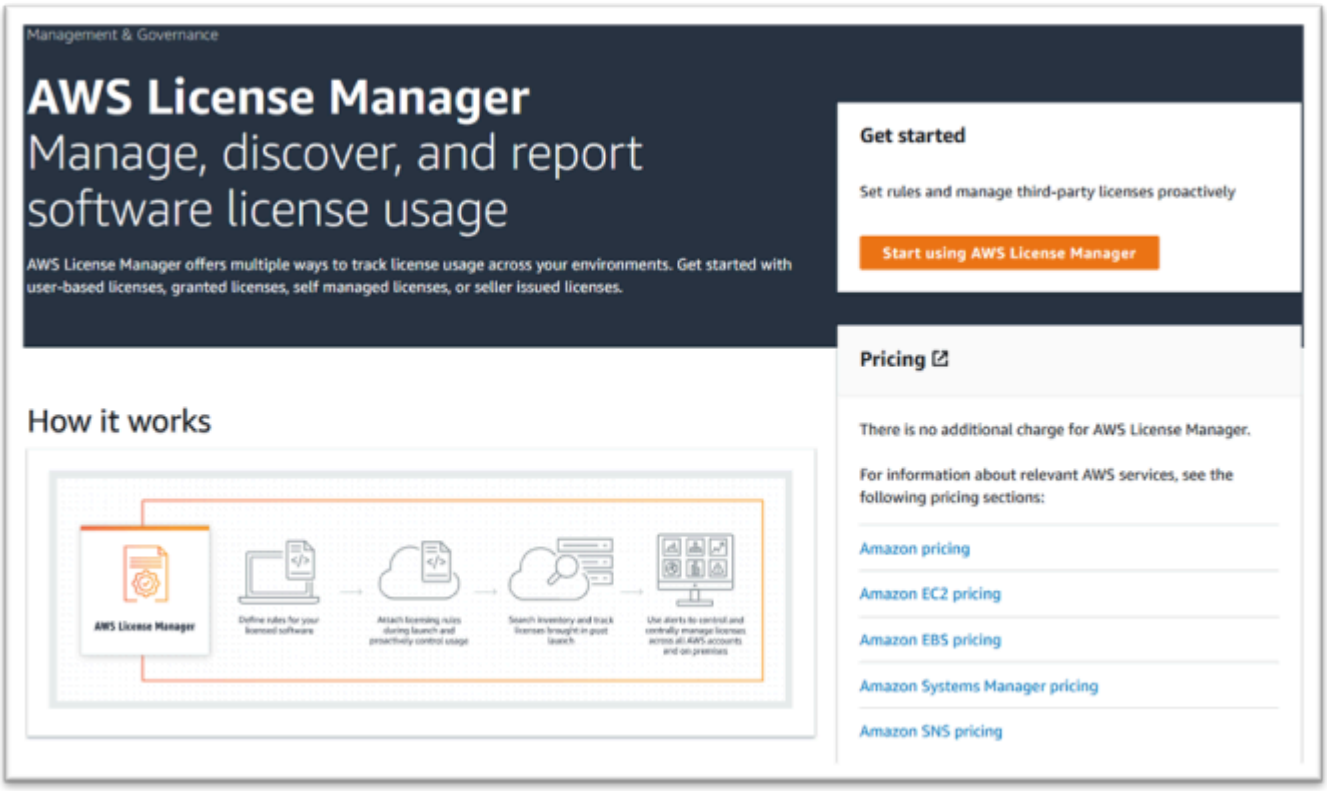


12. 역할 생성을 선택합니다.

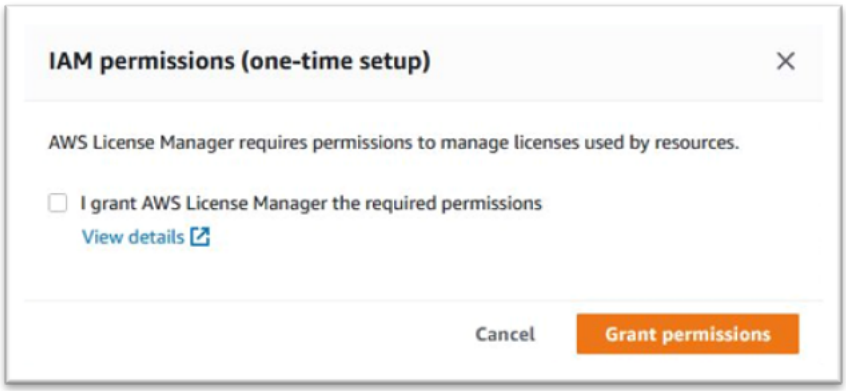
허용 목록 요청이 완료되면 다음 단계로 계속합니다.

License Manager에 필수 권한을 부여하세요

1. AWS License Manager 으로 이동합니다. AWS Management Console



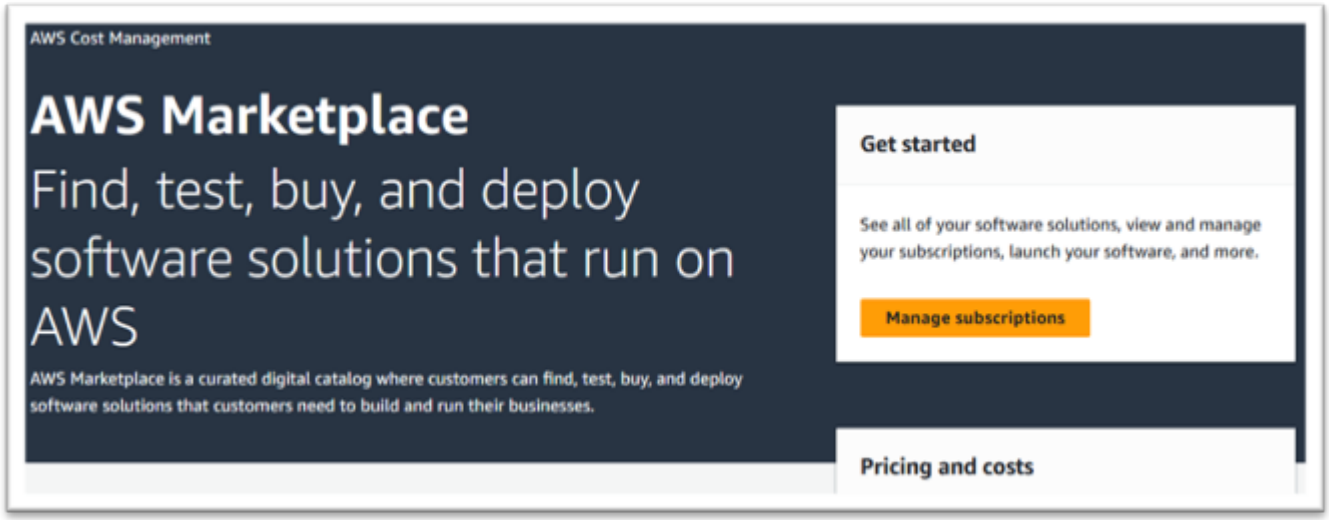
2. AWS License Manager 사용 시작을 선택합니다.
3. 다음 팝업이 표시되면 세부 정보를 확인한 다음 확인란을 선택하고 권한 부여를 누르세요.



Amazon Machine Images를 구독하세요

제품에 가입한 후 AWS Marketplace 제품의 AMI에서 인스턴스를 시작할 수 있습니다.

1. 에서 AWS Marketplace 구독으로 이동합니다. AWS Management Console
2. 구독 관리를 선택합니다.



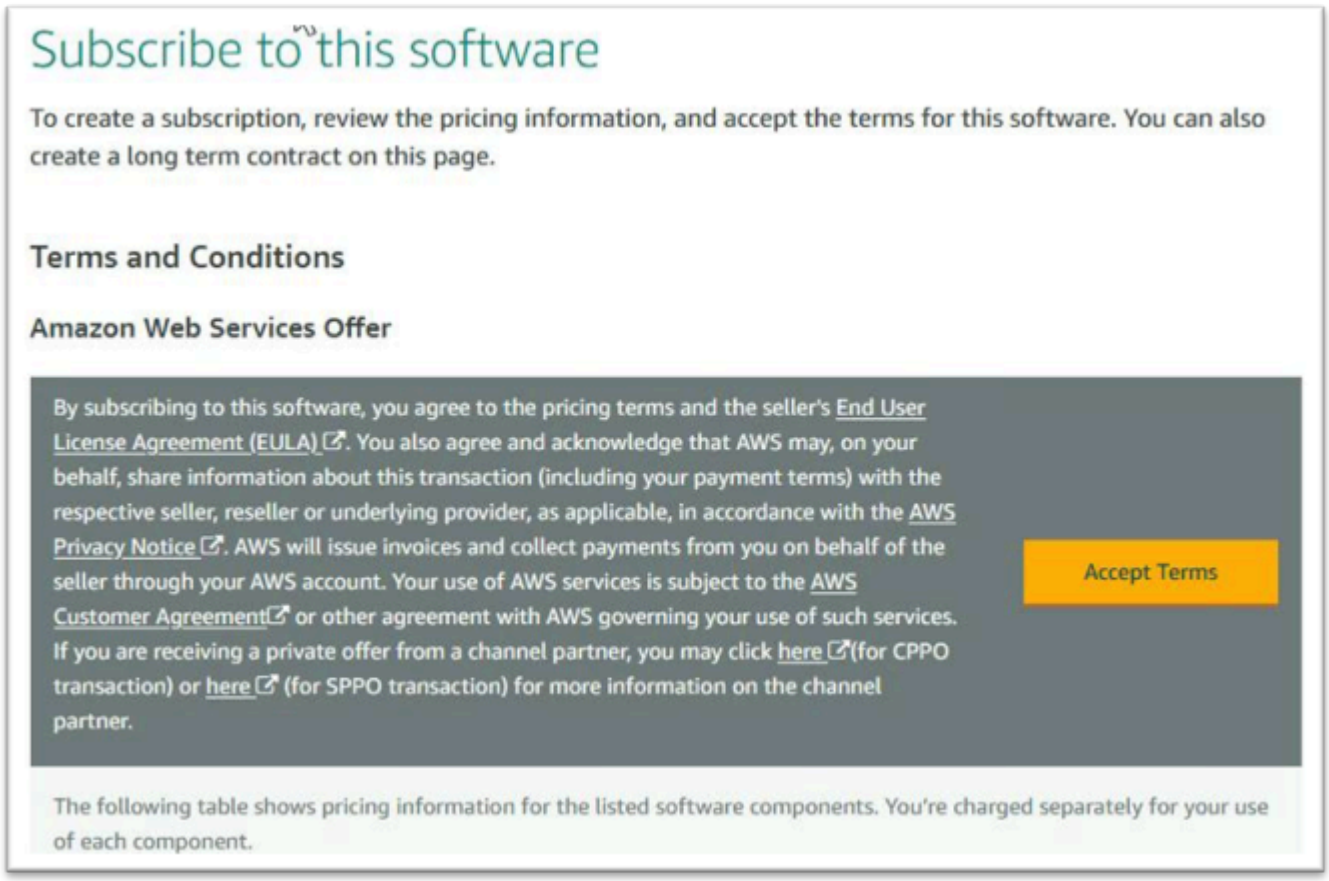
3. 다음 링크 중 하나를 복사하여 브라우저 주소 표시줄에 붙여넣습니다.

> **Note**
>
> 사용 권한이 부여된 제품 중 하나에 대한 링크만 선택하세요.

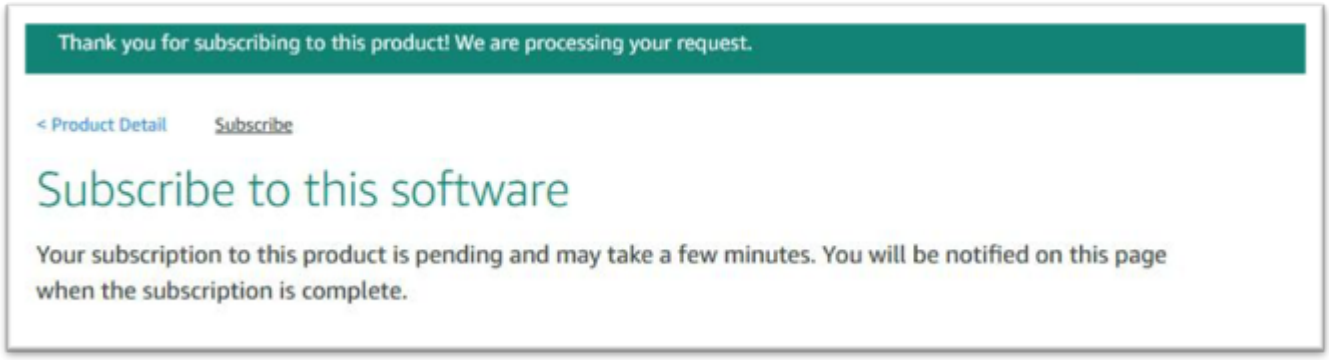
- Enterprise 서버: https://aws.amazon.com/marketplace/pp/prodview-g5emev63l7blc
- Windows용 Enterprise 서버: https://aws.amazon.com/marketplace/pp/prodview-lwybsiyikbhc2
- Enterprise 개발자: https://aws.amazon.com/marketplace/pp/prodview-77qmpr42yzxwk
- Visual Studio 2022를 사용하는 Enterprise 개발자: https://aws.amazon.com/marketplace/pp/prodview-m4l3lqiszo6cm
- Enterprise 분석기: https://aws.amazon.com/marketplace/pp/prodview-tttheylcmcihm
- Windows용 Enterprise 빌드 툴: https://aws.amazon.com/marketplace/pp/prodview-2rw35bbt6uozi
- Enterprise 저장 프로시저: https://aws.amazon.com/marketplace/pp/prodview-zoeyqnsdsj6ha
- SQL Server 2019를 사용한 Enterprise 저장 프로시저: https://aws.amazon.com/marketplace/pp/prodview-ynfklquwubnz4

4. 구독 계속을 선택합니다.

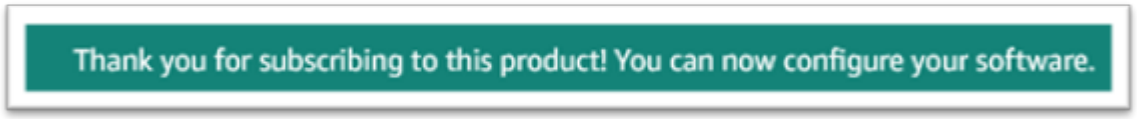
5. 이용 약관에 동의하는 경우 약관 동의를 선택합니다.



6. 이 프로세스에는 몇 분이 걸릴 수 있습니다.

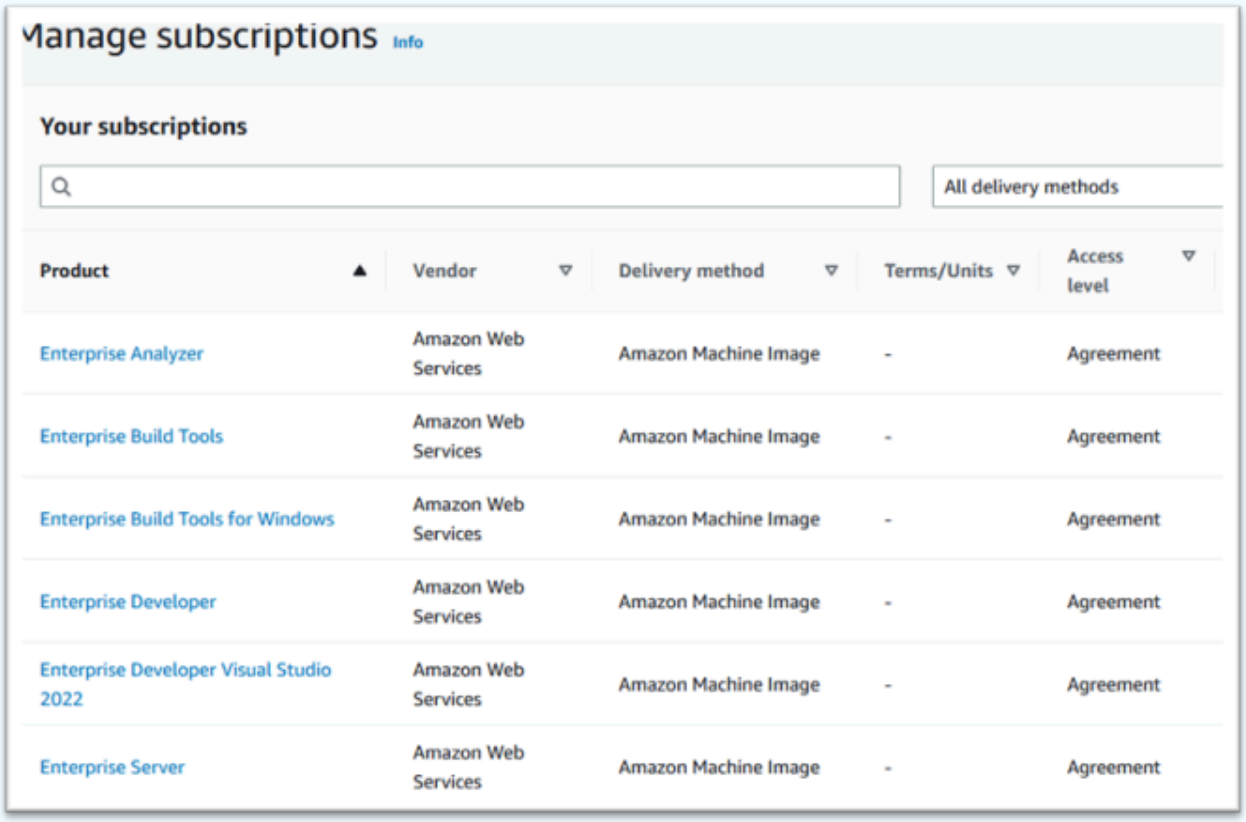


7. 감사 메시지가 표시된 후 3단계의 다음 링크를 복사하여 붙여넣으면 구독을 계속 추가할 수 있습니다.



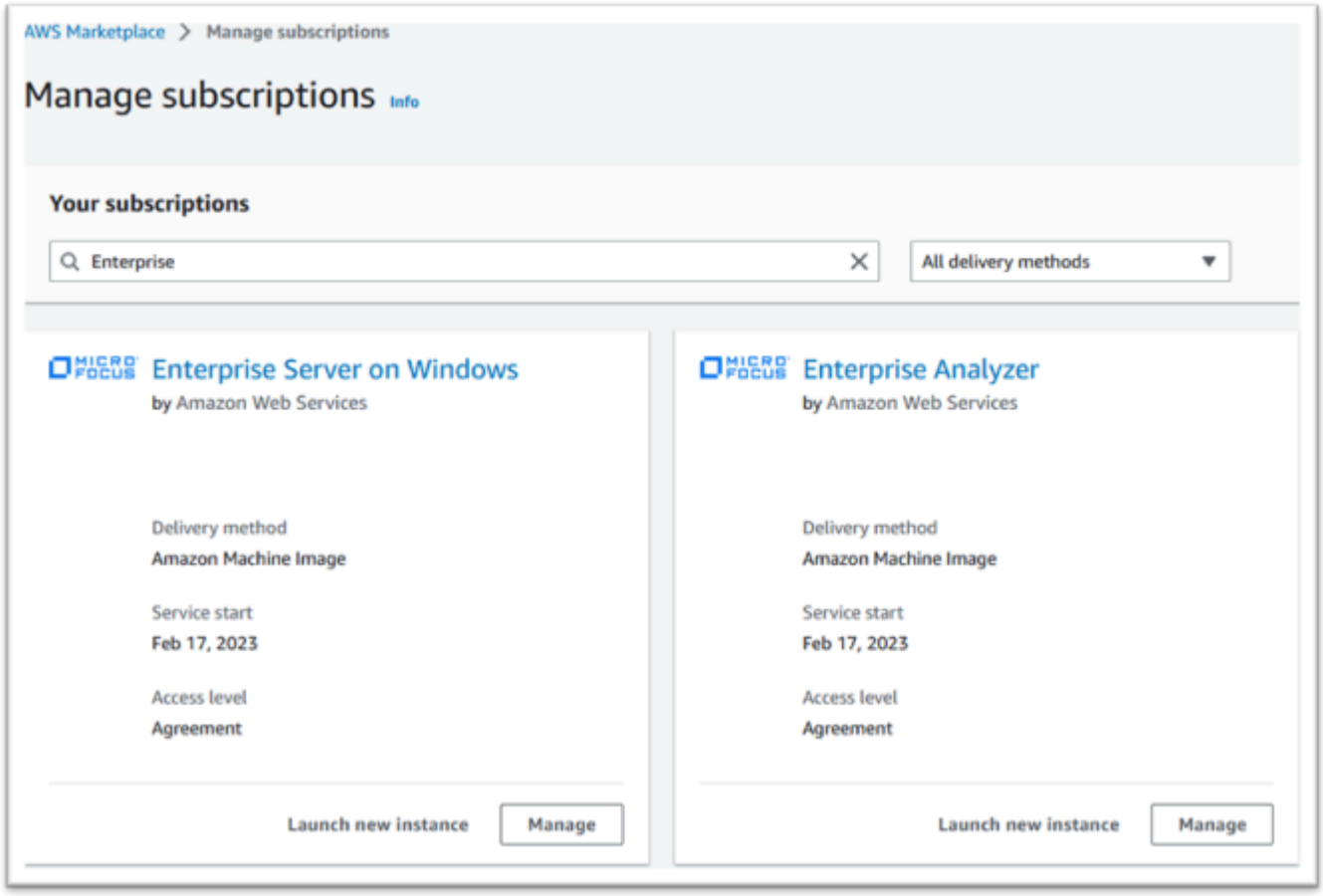
8. 구독 관리에 구독한 AMI가 모두 표시되면 중지하세요.

> **Note**
>
> 패널 환경설정(톱니바퀴 아이콘)은 보기를 표로 표시하도록 설정되어 있습니다.



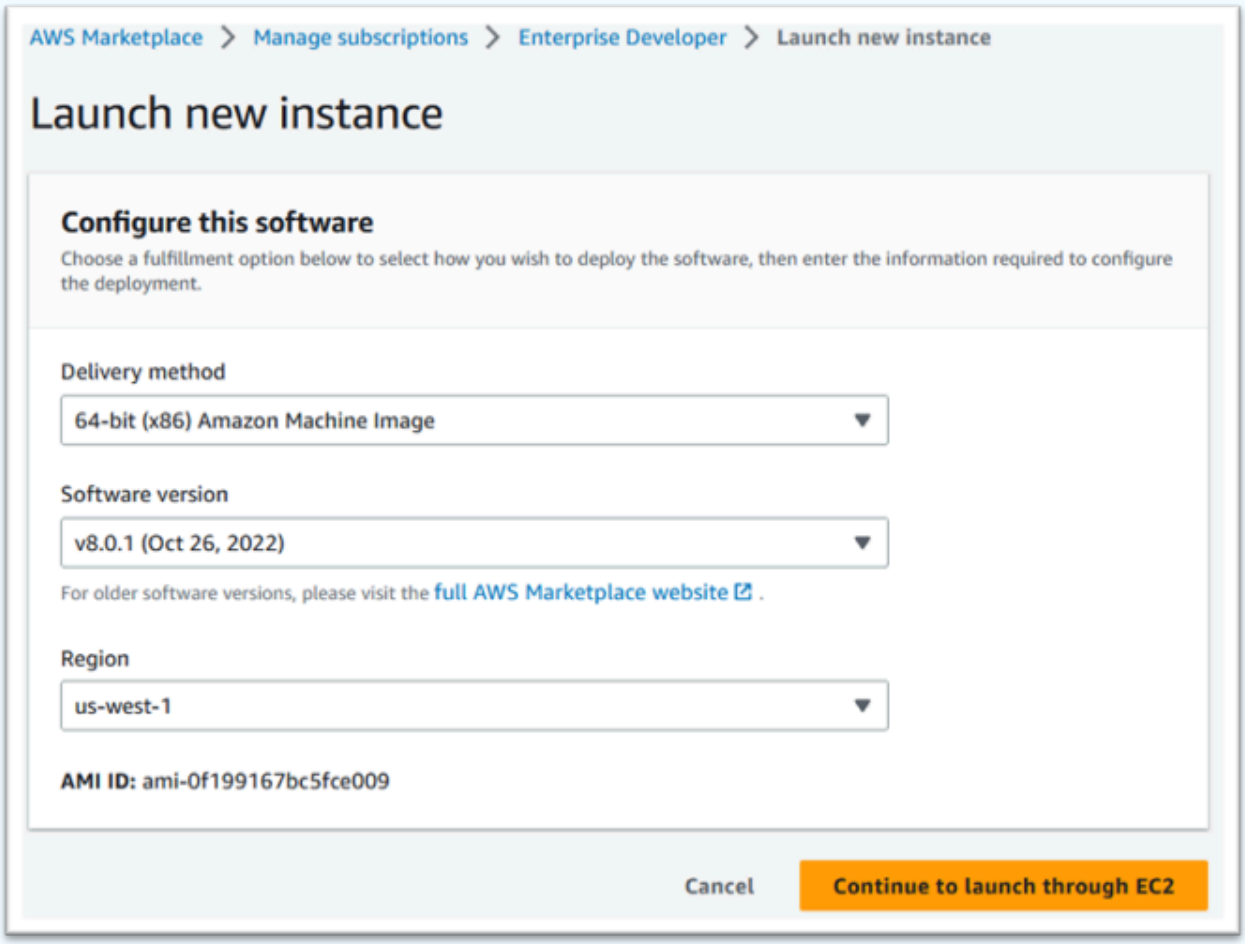
AWS 메인프레임 현대화 마이크로 포커스 인스턴스 시작

1. 에서 AWS Marketplace 구독으로 이동하십시오. AWS Management Console
2. 시작할 AMI를 찾고 새 인스턴스 시작을 선택합니다.



3. 새 인스턴스 시작 대화 상자에서 허용 목록에 있는 리전이 선택되어 있는지 확인합니다.
4. 계속을 눌러 EC2를 통해 시작하세요.

> **Note**
>
> 다음 예는 엔터프라이즈 개발자 AMI의 출시를 보여 주지만 프로세스는 모든 AWS 메인프레임 현대화 AMI에서 동일합니다.

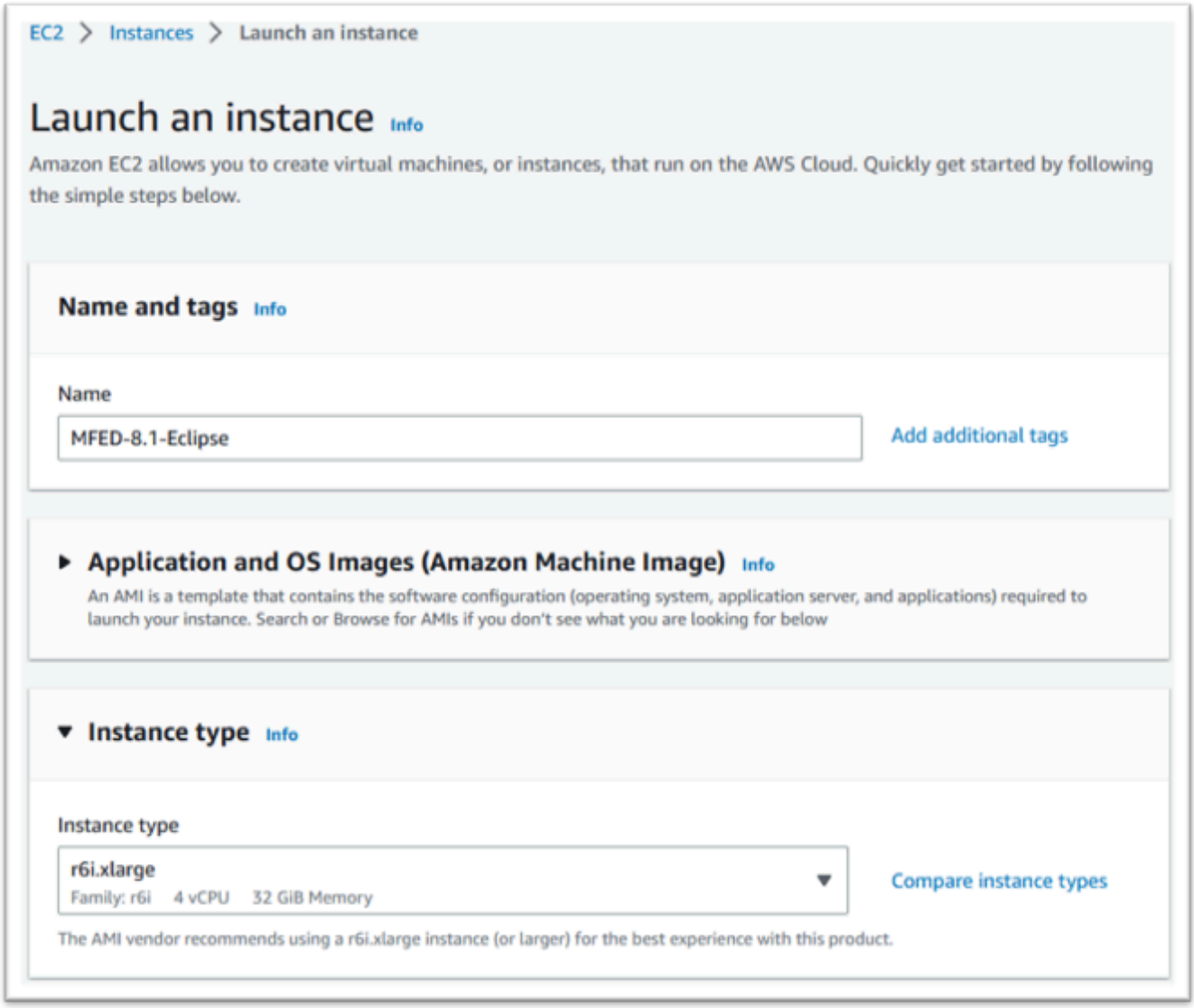


5. 서버 이름에 DNS 이름을 입력합니다.
6. 인스턴스 유형을 선택합니다.

   선택한 인스턴스 유형은 프로젝트 성과 및 비용 요구 사항에 따라 결정되어야 합니다. 권장되는 시작 지점은 다음과 같습니다.

   - Enterprise 분석기의 경우 r6i.xlarge
   - Enterprise 개발자용, r6i.large
   - Enterprise 서버의 독립 실행형 인스턴스의 경우 r6i.xlarge
   - 스케일 아웃이 가능한 Micro Focus Performance Availability Cluster(PAC)의 경우 r6i.large

   > **Note**
   >
   > 스크린샷의 애플리케이션 및 OS 이미지 섹션은 축소되었습니다.



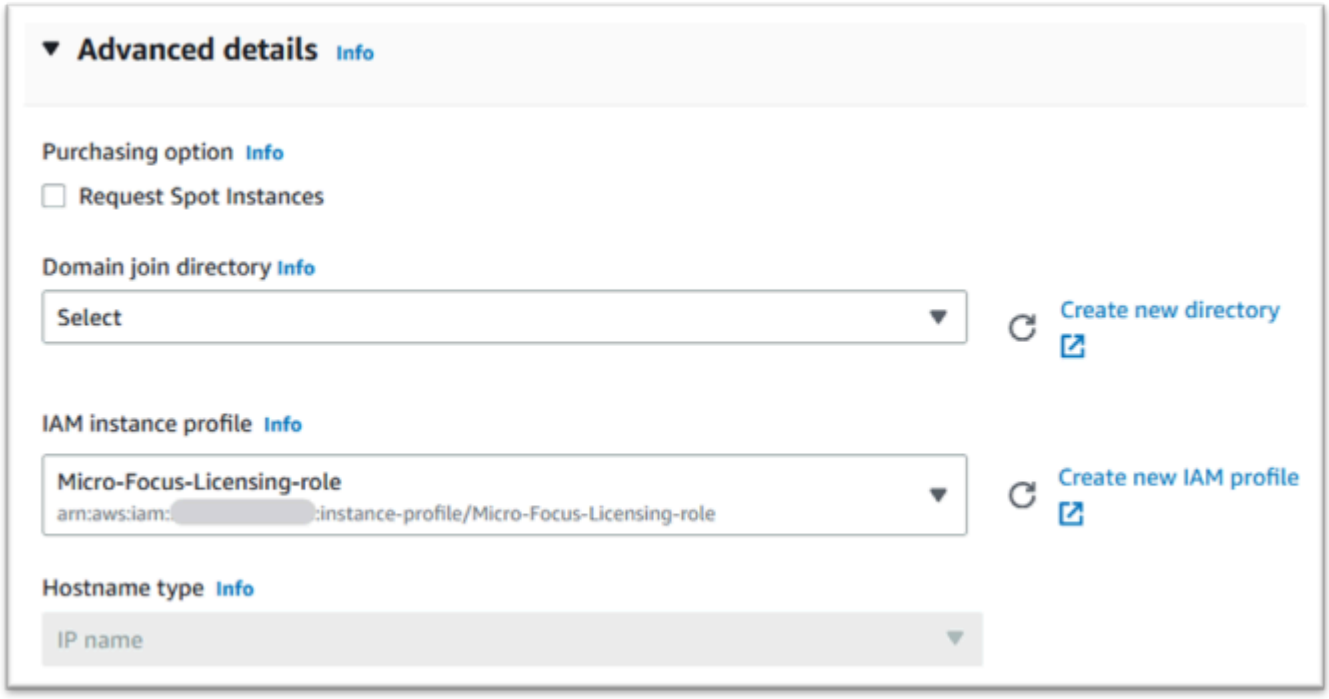
7. 키 쌍(표시되지 않음)을 선택하거나 생성(및 저장)합니다.

   Linux 인스턴스에 대한 키 쌍의 쌍세한 내용은 Amazon EC2 키 쌍 및 Linux 인스턴스를 참조하세요.

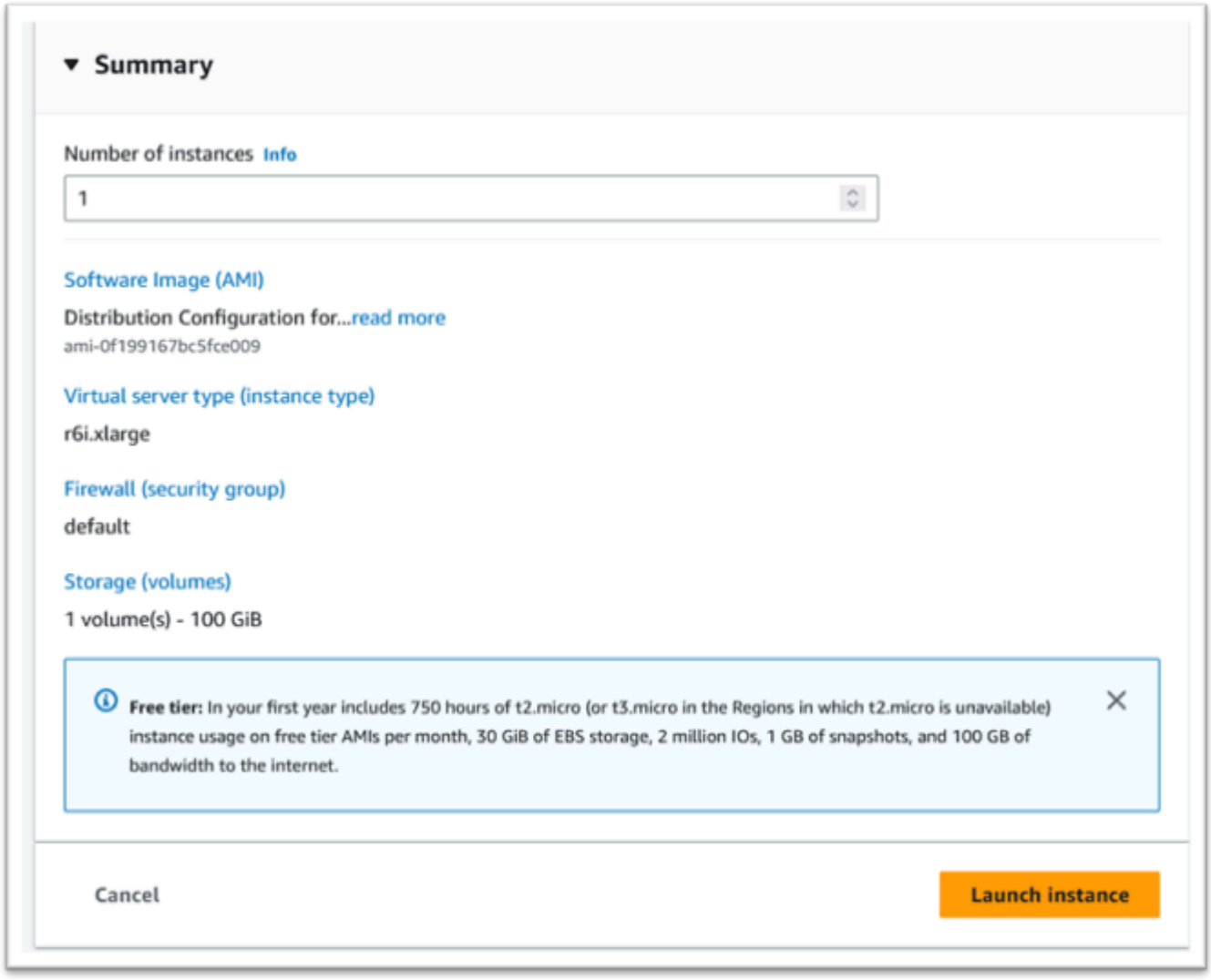
   키 쌍 및 Windows 인스턴스에 대한 자세한 내용은 Amazon EC2 키 쌍 및 Windows 인스턴스를 참조하세요.
8. 네트워크 설정을 편집하고 허용 목록에 있는 VPC와 적절한 서브넷을 선택합니다.
9. 보안 그룹을 선택하거나 생성합니다. Enterprise 서버 EC2 인스턴스인 경우 일반적으로 포트 86과 10086으로의 TCP 트래픽을 허용하여 Micro Focus 구성을 관리합니다.
10. Amazon EC2 인스턴스용 스토리지를 구성할 수도 있습니다.

11. 중요 - 고급 세부 정보를 확장하고 IAM 인스턴스 프로필에서 이전에 생성한 라이선스 역할(예: “Micro-focus-Licensing-role”)을 선택합니다.

> **Note**
>
> 이 단계를 놓친 경우, 인스턴스를 생성한 후 EC2 인스턴스에 대한 작업 메뉴의 보안 옵션에서 IAM 역할을 수정할 수 있습니다.

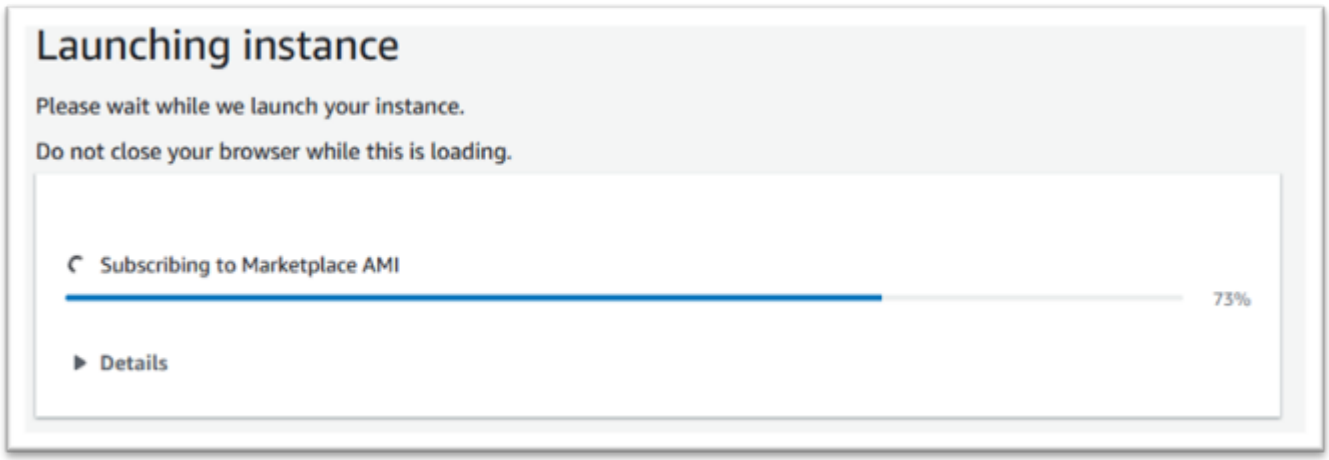


12. 요약을 검토하고 시작 인스턴스를 푸시하세요.

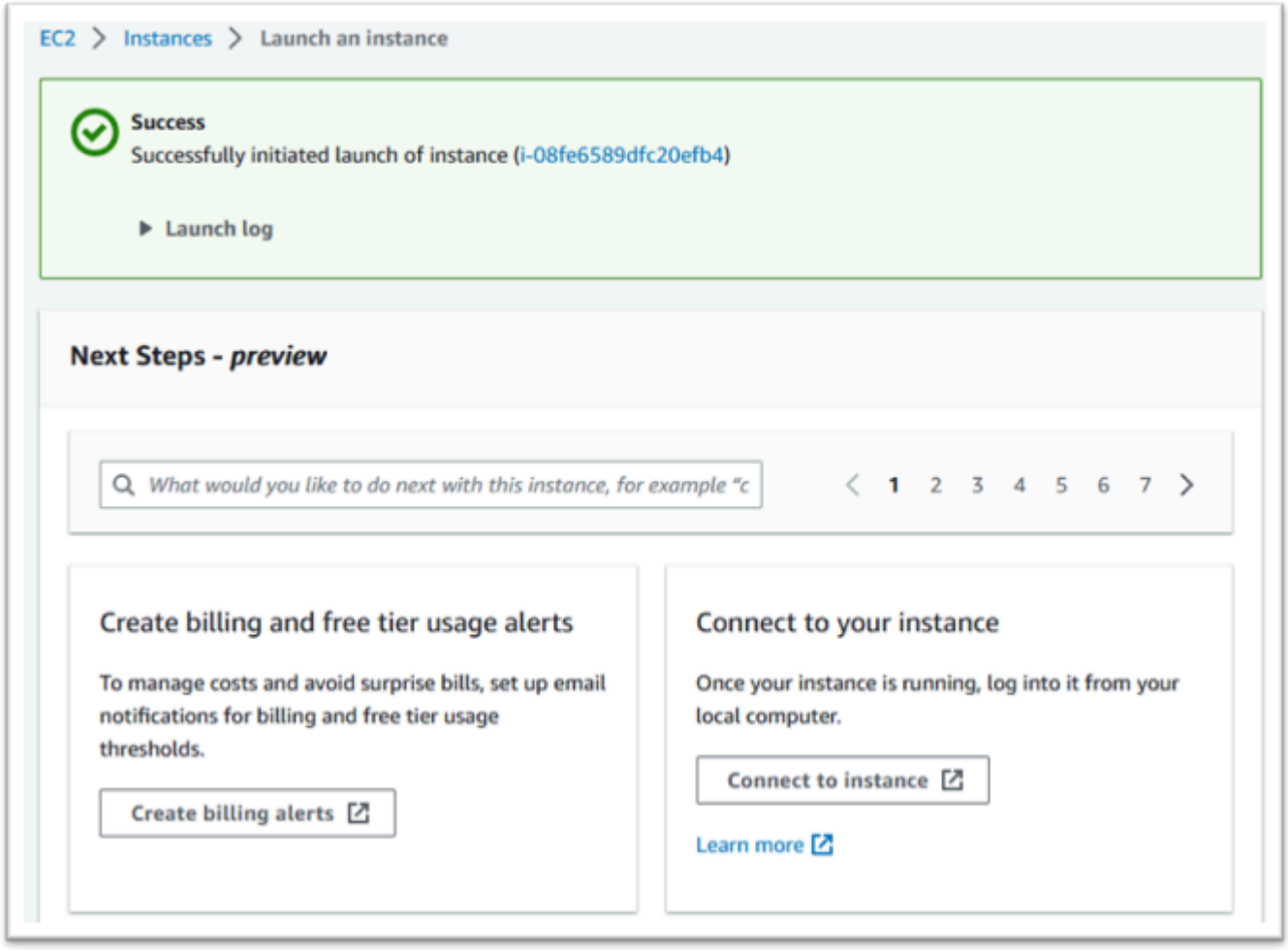


13. 잘못된 가상 서버 유형을 선택하면 인스턴스 시작이 실패합니다.

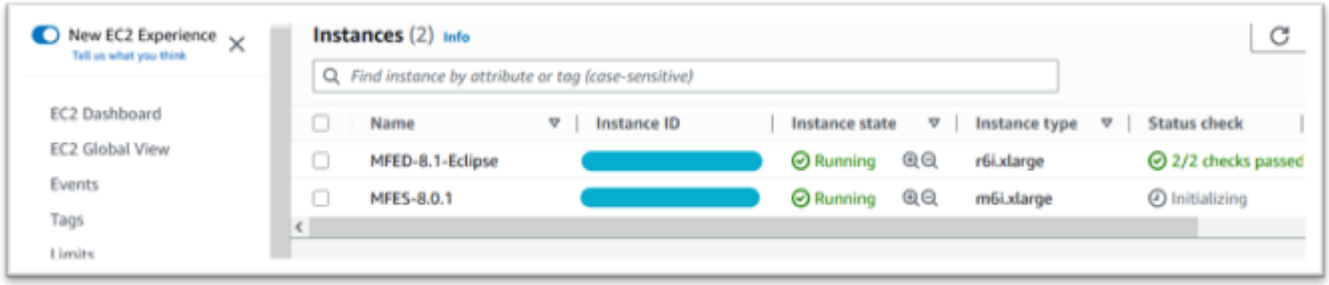
    이 경우 인스턴스 구성 편집을 선택하고 인스턴스 유형을 변경하세요.



14. “성공” 메시지가 표시되면 인스턴스에 연결을 선택하여 연결 세부 정보를 가져옵니다.



15. 또는 AWS Management Console에서 EC2로 이동할 수도 있습니다.
16. 새 인스턴스의 상태를 보려면 인스턴스를 선택합니다.



인터넷에 액세스할 수 없는 서브넷 또는 VPC

서브넷 또는 VPC에 아웃바운드 인터넷 액세스가 없는 경우 이러한 추가 변경을 수행하세요.

라이선스 관리자는 다음 AWS 서비스에 액세스할 수 있어야 합니다.

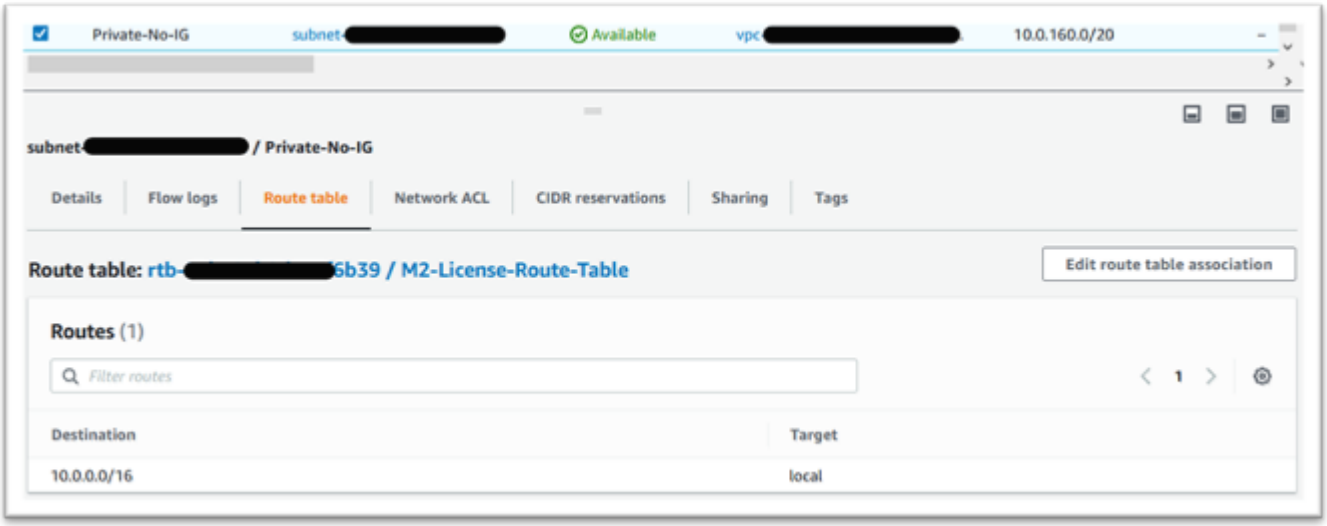
- com.amazonaws.*region*.s3
- com.amazonaws.*region*.ec2
- com.amazonaws.*region*.license-manager
- com.amazonaws.*region*.sts

이전 단계에서는 게이트웨이 엔드포인트로 com.amazonaws.*region*.s3를 정의했습니다. 이 엔드포인트에는 인터넷 액세스가 불가능한 모든 서브넷에 대한 라우팅 테이블 항목이 필요합니다.

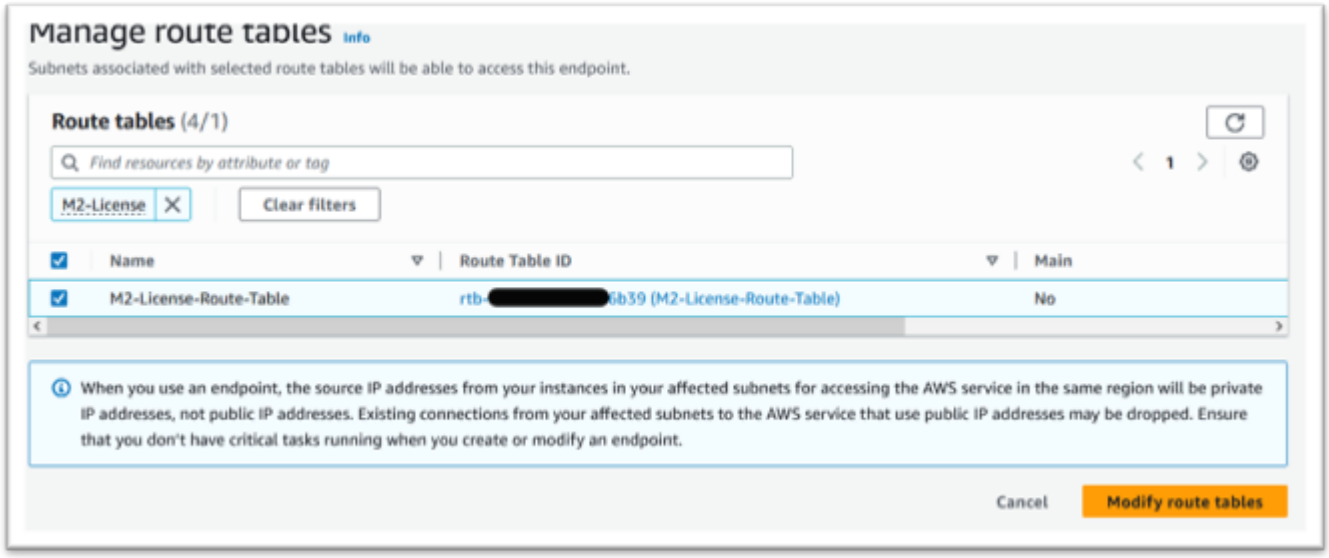
추가 서비스 3개는 인터페이스 엔드포인트로 정의됩니다.

Amazon S3 엔드포인트에 대한 라우팅 테이블 항목 추가

1. 에서 VPC로 이동하여 [서브넷] 을 AWS Management Console 선택합니다.
2. Amazon EC2 인스턴스가 생성될 서브넷을 선택하고 라우팅 테이블 탭을 선택합니다.
3. 라우팅 테이블 ID의 후행 몇 자리 숫자를 기록해 두세요. 아래 이미지의 6b39를 예로 들 수 있습니다.



4. 탐색 창에서 엔드포인트를 선택합니다.
5. 이전에 생성한 엔드포인트를 선택한 다음, 엔드포인트의 라우팅 테이블 탭 또는 작업 드롭다운에서 라우팅 테이블 관리를 선택합니다.
6. 앞서 식별한 숫자를 사용하여 라우팅 테이블을 선택하고 라우팅 테이블 수정을 누릅니다.

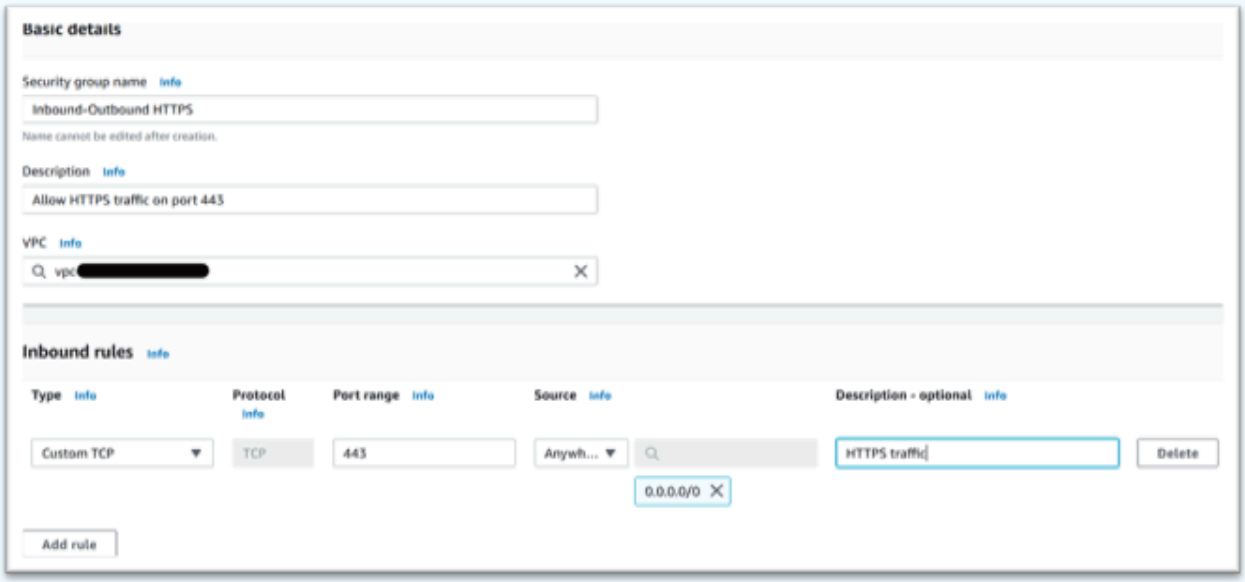


필수 보안 그룹 정의

Amazon EC2 및 License Manager 서비스는 포트 443을 통해 HTTPS를 통해 통신합니다. AWS STS 이 통신은 양방향으로 이루어지므로 인스턴스가 서비스와 통신할 수 있도록 인바운드 및 아웃바운드 규칙이 필요합니다.

1. AWS Management Console에서 Amazon VPC로 이동합니다.
2. 탐색 창에서 보안 그룹과 보안 그룹 생성을 차례로 선택합니다.
3. 보안 그룹 이름과 설명을 입력합니다(예: “인바운드-아웃바운드 HTTPS”).
4. VPC 선택 영역에서 X를 눌러 기본 VPC를 제거하고 S3 엔드포인트가 포함된 VPC를 선택합니다.
5. 어디서든 포트 443의 TCP 트래픽을 허용하는 인바운드 규칙을 추가합니다.

> **Note**
>
> 소스를 제한하여 인바운드(및 아웃바운드 규칙)를 추가로 제한할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 보안 그룹을 사용한 AWS 리소스 트래픽 제어를 참조하십시오.

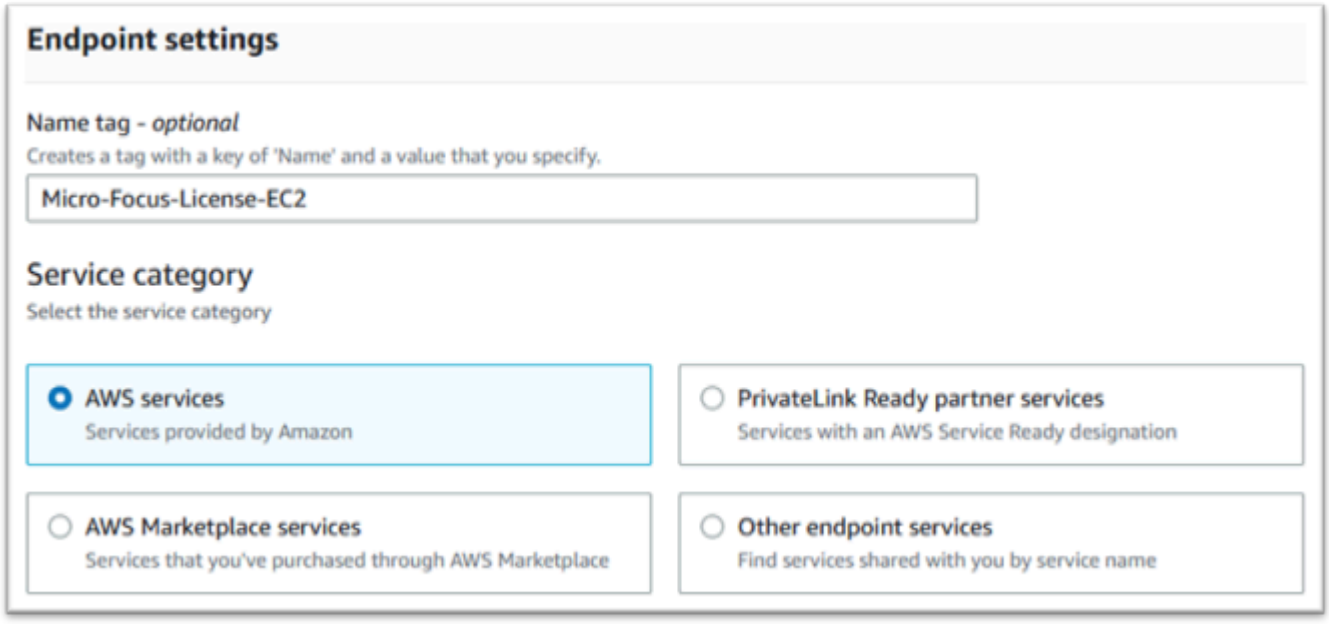


6. 보안 그룹을 생성합니다.

서비스 엔드포인트 생성

이 프로세스를 각 서비스에 대해 한 번씩 세 번 반복합니다.

1. 에서 Amazon VPC로 이동하여 엔드포인트를 AWS Management Console 선택합니다.
2. 엔드포인트 생성을 누릅니다.
3. 이름을 입력합니다(예: “Micro-Focus-License-EC2”, “Micro-Focus-License-STS”, or “Micro-Focus-License-Manager”).
4. AWS 서비스 서비스 카테고리를 선택합니다.



5. 서비스에서 다음 중 하나에 해당하는 인터페이스 서비스를 검색합니다.

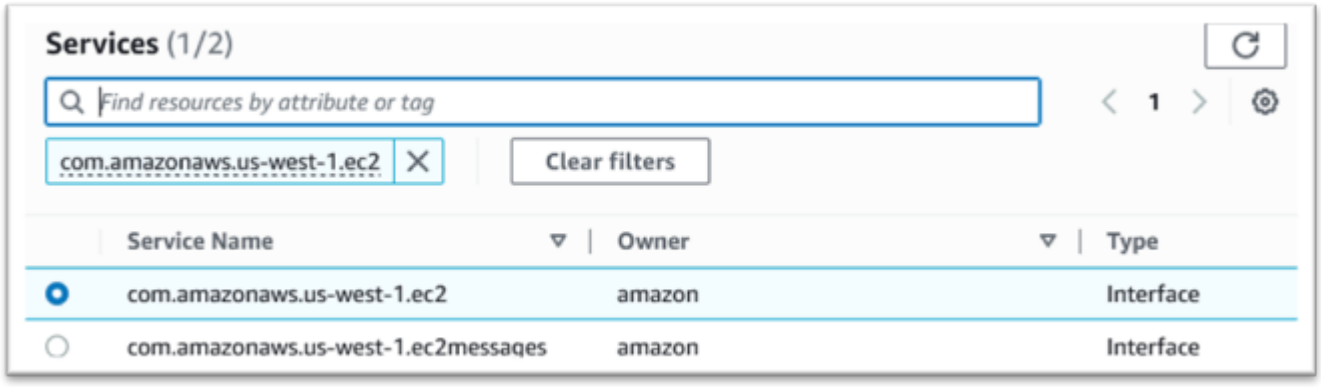
   - “com.amazonaws.*region*.ec2”
   - “com.amazonaws.*region*.sts”
   - “com.amazonaws.*region*.license-manager”

   예:

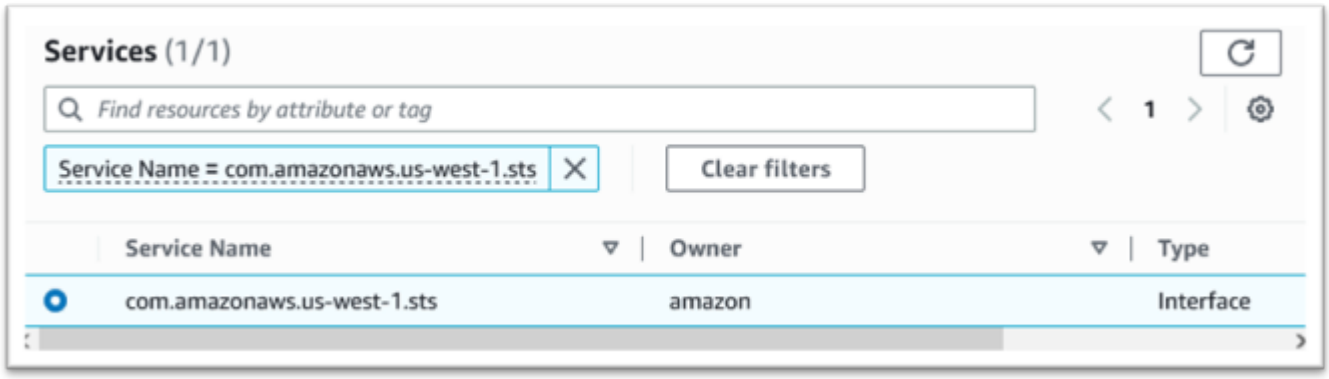
   - “com.amazonaws.us-west-1.ec2”
   - “com.amazonaws.us-west-1.sts”
   - “com.amazonaws.us-west-1.license-manager”

6. 일치하는 인터페이스 서비스를 선택합니다.

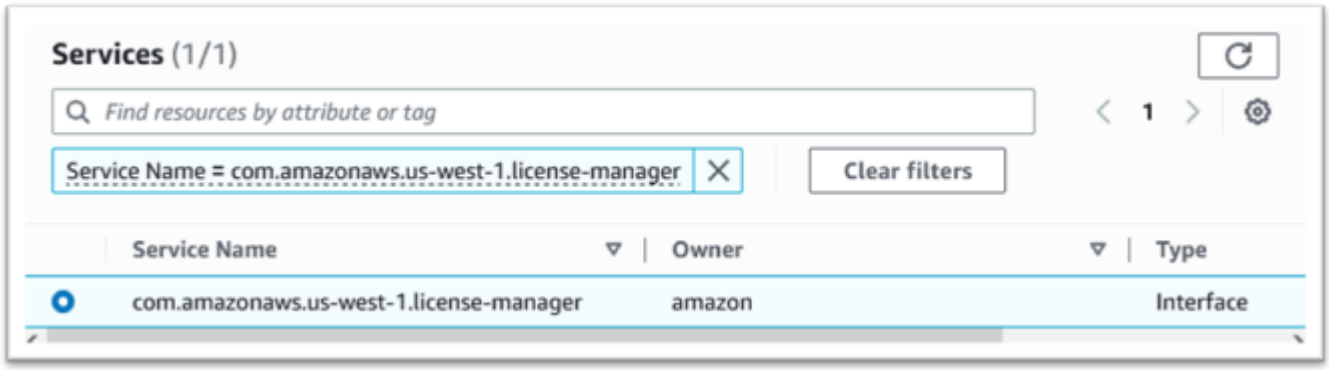
   com.amazonaws.***region***.ec2:



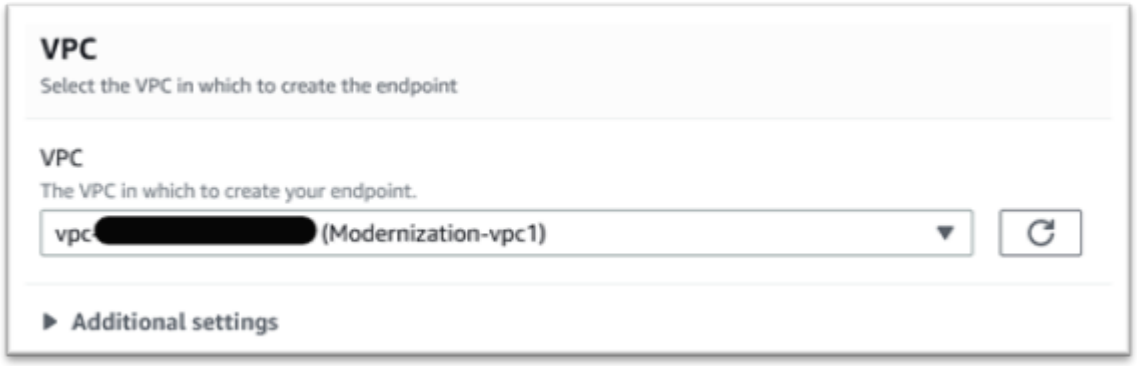
com.amazonaws.***region***.sts:



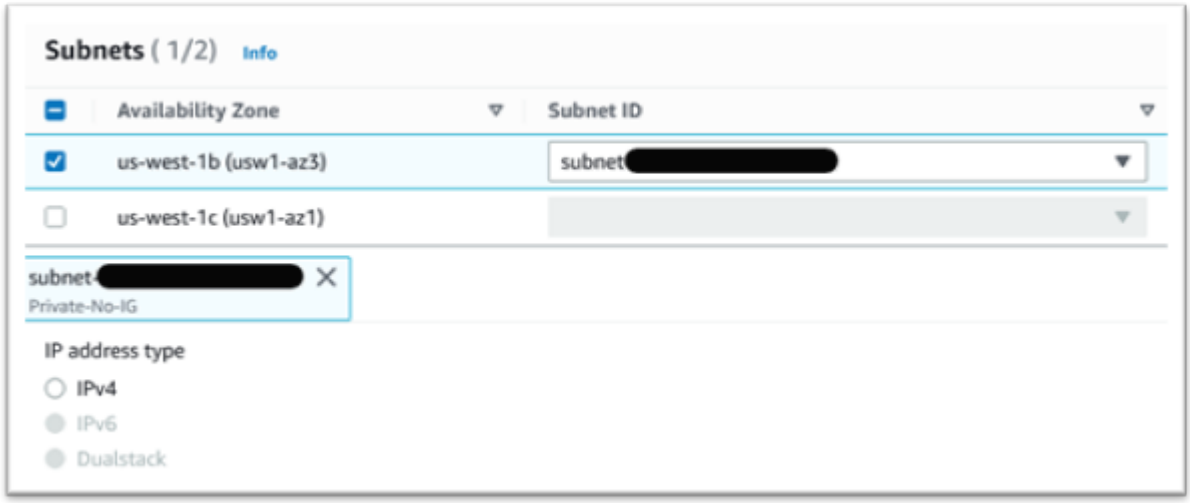
com.amazonaws.***region***.license-manager:



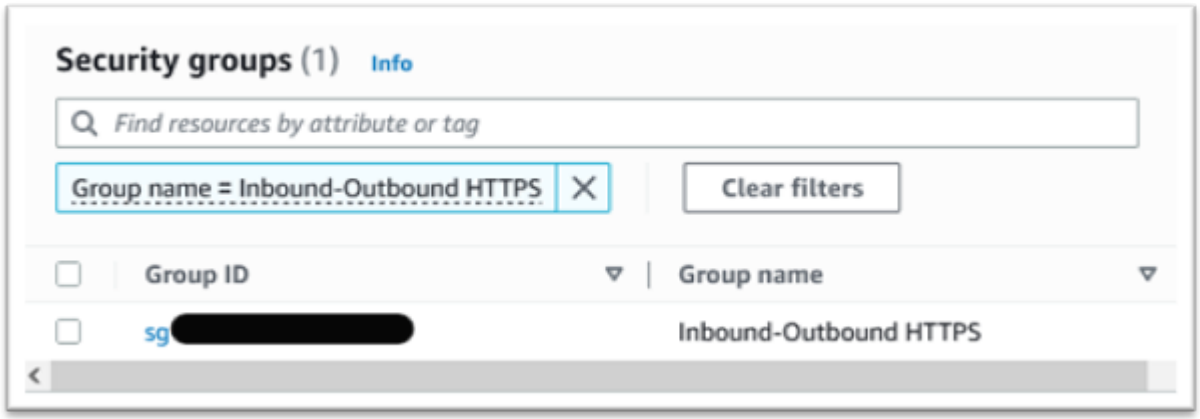
7. VPC에서 인스턴스의 VPC를 선택합니다.



8. VPC의 가용 영역과 서브넷을 선택합니다.



9. 보안 그룹을 생성하려면 생성을 선택합니다.



10. 정책에서 전체 액세스를 선택합니다.

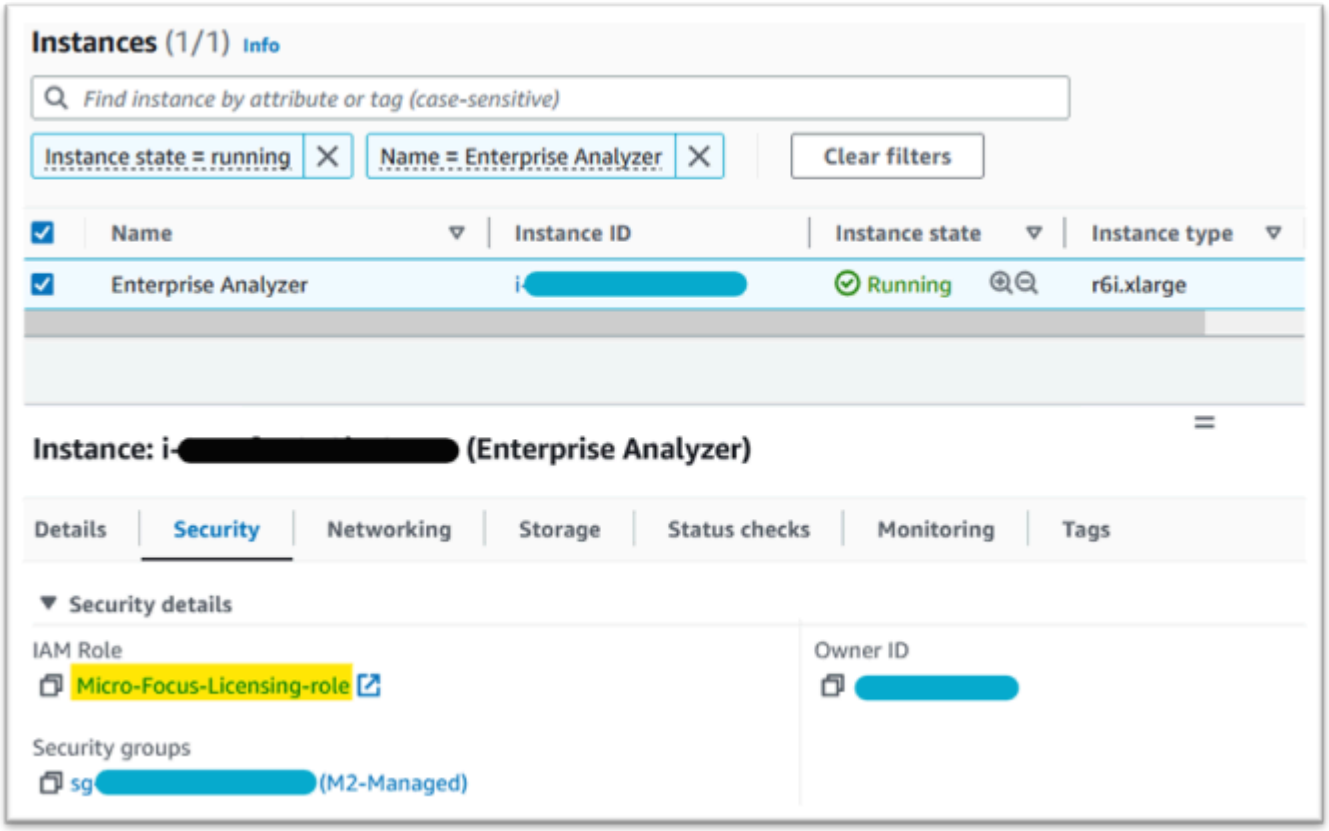
11. 엔드포인트 생성을 선택합니다.
12. 나머지 인터페이스에 대해 이 절차를 반복합니다.

라이선스 문제 해결

AMI에 액세스하거나 사용하는 데 문제가 있는 경우 다음 정보가 도움이 될 수 있습니다.

Amazon EC2 인스턴스에 IAM 라이선스 역할이 있는지 확인

Amazon EC2 인스턴스 세부 정보의 보안 탭에서 확인할 수 있습니다. 이는 작업 드롭다운 메뉴의 보안 옵션을 사용하여 변경할 수 있습니다.



Reachability Analyzer 사용

콘솔 페이지에서 Reachability 분석기를 찾을 수 있습니다. AWS Network Manager

AMI에서 생성된 Amazon EC2 인스턴스와 Amazon S3 VPC 엔드포인트 사이의 경로를 생성하고 분석합니다.

Amazon EC2 인스턴스에서 인터넷에 액세스할 수 없는 경우 4개 엔드포인트 모두에 대한 경로 분석을 반복합니다.

Reachability Analyzer에 대한 자세한 내용은 도달성 분석기 가이드의 Reachability Analyzer로 시작하기를 참조하세요.

라이선스 데몬 실행

Windows Enterprise 개발자의 경우 명령 프롬프트에서 다음 명령을 사용합니다.

```
“C:\Program Files (x86)\Micro Focus\Enterprise Developer\AdoptOpenJDK\bin\java” -jar
 "C:\Program Files (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

그리고 출력을 검사합니다. SLF4J 메시지를 무시하고 첫 번째 예외를 찾아보세요.

Enterprise Analyzer의 경우 명령 프롬프트에서 다음 명령을 사용하세요.

```
"C:\Program Files (x86)\Micro Focus\AdoptOpenJDK\bin\java" -jar "C:\Program Files
 (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

그리고 출력을 검사합니다. SLF4J 메시지를 무시하고 첫 번째 예외를 찾아보세요.

Linux에서 다음을 실행합니다.

```
java -jar /var/microfocuslicensing/bin/aws-license-daemon.jar
```

SLF4J 메시지는 무시하고 첫 번째 예외를 찾아보세요.

예를 들어, Amazon S3 리소스를 사용할 수 없는 경우 예외는 다음과 같습니다.

```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

Exception in thread "main" software.amazon.awssdk.services.s3.model.S3Exception: Access
 Denied (Service: S3, Status Code: 403, Request ID: P6
```

예외 메시지는 사용할 수 없는 리소스를 나타냅니다. 구성 값을 이 항목에 표시된 값과 비교하세요.

튜토리얼: Micro Focus Enterprise Analyzer 및 Micro Focus Enterprise Developer를 통해 AppStream 2.0 설정하기

AWS Mainframe Modernization은 Amazon AppStream 2.0을 통해 여러 도구를 제공합니다. AppStream 2.0은 애플리케이션을 다시 작성하지 않고도 데스크톱 애플리케이션을 사용자에게 스트리밍할 수 있는 안전한 완전 관리형 애플리케이션 스트리밍 서비스입니다. AppStream 2.0을 통해 사용자는 필요한 애플리케이션에 즉시 액세스할 수 있으며 선택한 디바이스에서 빠르게 응답하는 원활한 사용자 환경을 이용할 수 있습니다. AppStream 2.0을 사용하여 런타임 엔진 전용 도구를 호스팅하면 고객 애플리케이션 팀이 웹 브라우저에서 직접 도구를 사용하고 Amazon S3 버킷 또는 CodeCommit 리포지토리에 저장된 애플리케이션 파일과 상호 작용할 수 있습니다.

AppStream 2.0의 브라우저 지원에 대한 자세한 내용은 Amazon AppStream 2.0 관리 안내서의 시스템 요구 사항 및 기능 지원(웹 브라우저)을 참조하세요. AppStream 2.0을 사용할 때 문제가 발생하는 경우 Amazon AppStream 2.0 관리 안내서의 AppStream 2.0 사용자 문제 해결을 참조하세요.

이 문서는 고객 운영 팀 구성원을 대상으로 합니다. AWS Mainframe Modernization과 함께 사용되는 Micro Focus Enterprise Analyzer 및 Micro Focus Enterprise Developer 도구를 호스팅하도록 Amazon AppStream 2.0 플릿 및 스택을 설정하는 방법을 설명합니다. 일반적으로 AWS Mainframe Modernization 접근 방식의 마이그레이션 및 현대화 단계에서는 Micro Focus Enterprise Analyzer를 사용하고 Micro Focus Enterprise Developer는 일반적으로 평가 단계에서 사용됩니다. Enterprise Analyzer와 Enterprise Developer를 모두 사용하려는 경우 각 도구에 대해 별도의 플릿과 스택을 만들어야 합니다. 라이선스 조건이 다르기 때문에 각 도구마다 자체 플릿과 스택이 필요합니다.

> **Important**
>
> 이 자습서의 단계는 다운로드 가능한 AWS CloudFormation 템플릿 cfn-m2-appstream-fleet-ea-ed.yml을 기반으로 합니다.

주제

필수 조건

- cfn-m2-appstream-fleet-ea-ed.yml 템플릿을 다운로드하세요.
- 기본 VPC 및 보안 그룹의 ID를 가져옵니다. 기본 VPC에 대한 자세한 내용은 Amazon VPC 사용 설명서의 기본 VPC를 참조하세요. 기본 보안 그룹에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 기본 및 사용자 지정 보안 그룹을 참조하세요.
- 다음 권한이 활성화되어 있는지 확인하세요.
  - AppStream 2.0에서 스택, 플릿 및 사용자를 생성할 수 있는 권한.
  - 템플릿을 사용하여 AWS CloudFormation에서 스택을 생성할 수 있는 권한.
  - Amazon S3에 버킷을 생성하고 버킷에 파일을 업로드하는 권한.

- IAM에서 보안 인증(`access_key_id` 및 `secret_access_key`)을 다운로드할 수 있는 권한.

1단계: AppStream 2.0 이미지 가져오기

이 단계에서는 Enterprise Analyzer 및 Enterprise Developer용 AppStream 2.0 이미지를 AWS 계정과 공유합니다.

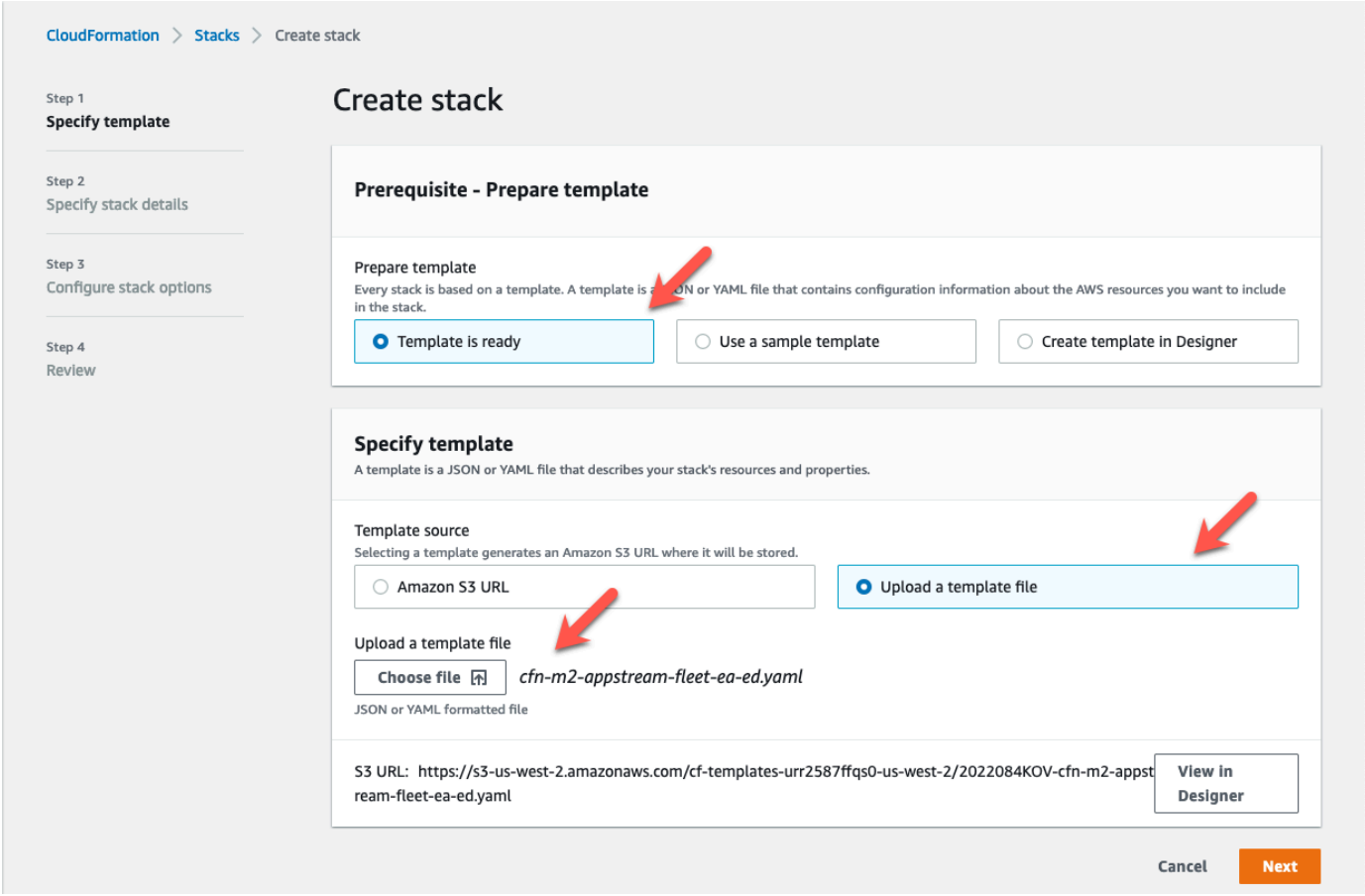
1. https://console.aws.amazon.com/m2/에서 AWS Mainframe Modernization 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 도구를 선택합니다.
3. 자산 분석, 개발 및 구축에서 내 AWS 계정과 자산 공유를 선택합니다.

2단계: AWS CloudFormation 템플릿을 사용하여 스택 생성

이 단계에서는 다운로드한 AWS CloudFormation 템플릿을 사용하여 Micro Focus Enterprise Analyzer를 실행하기 위한 AppStream 2.0 스택과 플릿을 생성합니다. 각 도구에는 AppStream 2.0의 자체 플릿과 스택이 필요하기 때문에 나중에 이 단계를 반복하여 Micro Focus Enterprise Developer를 실행하기 위한 또 다른 AppStream 2.0 스택과 플릿을 만들 수 있습니다. AWS CloudFormation 스택에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 템플릿 사용을 참조하세요.

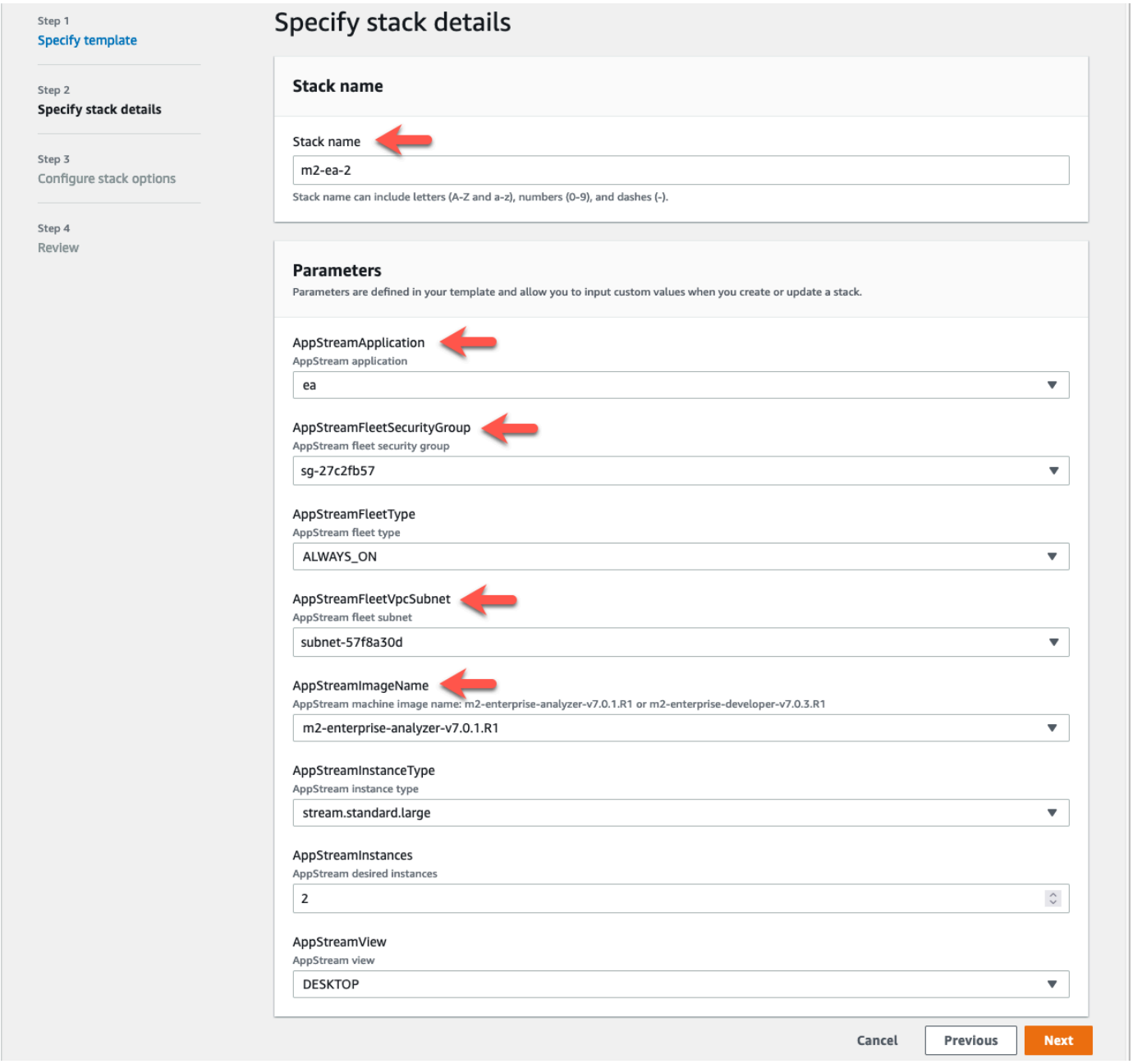
> **Note**
>
> AWS Mainframe Modernization은 Enterprise Analyzer 및 Enterprise Developer 사용에 대한 표준 AppStream 2.0 가격 책정에 추가 요금을 추가합니다. 자세한 내용은 AWS Mainframe Modernization 요금을 참조하세요.

1. 필요한 경우 cfn-m2-appstream-fleet-ea-ed.yml 템플릿을 다운로드하세요.
2. AWS CloudFormation 콘솔을 스택 생성 및 새 리소스 포함(표준)을 선택합니다.
3. 사전 조건 - 템플릿 준비에서 템플릿 준비 완료를 선택합니다.
4. 템플릿 지정에서 템플릿 파일 업로드를 선택합니다.
5. 템플릿 파일 업로드에서 파일 선택을 선택하고 cfn-m2-appstream-fleet-ea-ed.yml 템플릿을 업로드합니다.
6. 다음(Next)을 선택합니다.



7. 스택 세부 정보 지정에서 다음 정보를 입력합니다.

   - 스택 이름에 원하는 이름을 입력합니다. 예: **`m2-ea`**.
   - AppStreamApplication에서 ea를 선택합니다.
   - AppStreamFleetSecurityGroup에서 기본 VPC의 기본 보안 그룹을 선택합니다.
   - AppStreamFleetVpcSubnet에서 기본 VPC 내의 서브넷을 선택합니다.
   - AppStreamImageName에서 `m2-enterprise-analyzer`로 시작하는 이미지를 선택합니다. 이 이미지에는 현재 지원되는 버전의 Micro Focus Enterprise Analyzer 도구가 포함되어 있습니다.
   - 다른 필드의 기본값을 그대로 사용하고 다음을 선택합니다.



8. 기본값을 수락하고 다시 다음을 선택합니다.
9. 검토 시 모든 매개 변수가 의도한 대로인지 확인하세요.
10. 아래로 스크롤하여 AWS CloudFormation에서 사용자 지정 이름으로 IAM 리소스를 생성할 수 있음을 승인합니다를 선택하고 스택 생성을 선택합니다.

스택과 플릿을 생성하는 데 20~30분 정도 걸립니다. 새로 고침을 선택하여 발생하는 AWS CloudFormation 이벤트를 확인할 수 있습니다.

3단계: AppStream 2.0에서 사용자 생성

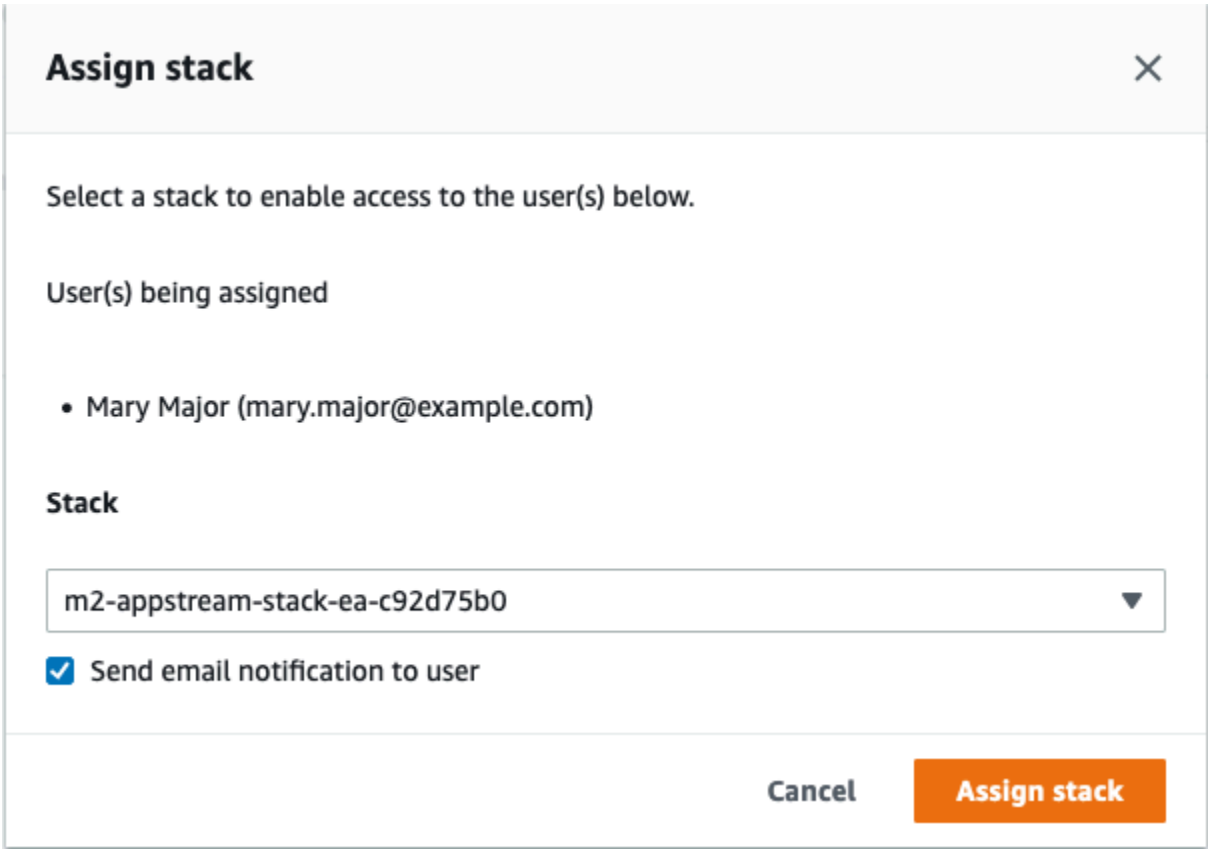
AWS CloudFormation이 스택 생성을 완료하기를 기다리는 동안 AppStream 2.0에서 한 명 이상의 사용자를 생성할 수 있습니다. AppStream 2.0에서 Enterprise Analyzer를 사용할 수 있는 사용자입니다. 각 사용자의 이메일 주소를 지정해야 하며, 각 사용자에게 Amazon S3에서 버킷을 생성하고, 버킷에 파일을 업로드하고, 콘텐츠를 매핑할 버킷에 연결할 수 있는 충분한 권한이 있는지 확인해야 합니다.

1. AppStream 2.0 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 사용자 풀을 선택합니다.
3. 사용자 생성을 선택합니다.
4. 사용자가 AppStream 2.0 사용 초대를 받을 수 있는 이메일 주소, 이름, 성을 입력하고 사용자 생성을 선택합니다.
5. 필요한 경우 이 단계를 반복하여 사용자를 더 생성하세요. 각 사용자의 이메일 주소는 고유해야 합니다.

AppStream 2.0 사용자 생성에 대한 자세한 내용은 Amazon AppStream 2.0 관리 안내서의 AppStream 2.0 사용자 풀을 참조하세요.

AWS CloudFormation가 스택 생성을 완료하면 다음과 같이 생성한 사용자를 스택에 배정할 수 있습니다.

1. AppStream 2.0 콘솔을 엽니다.
2. 사용자 이름을 선택합니다.
3. 작업을 선택한 다음 스택 할당을 선택합니다.
4. 스택 할당에서 `m2-appstream-stack-ea`로 시작하는 스택을 선택합니다.
5. 스택 할당을 선택합니다.



사용자를 스택에 할당하면 AppStream 2.0에서 제공된 주소로 사용자에게 이메일을 보냅니다. 이 이메일에는 AppStream 2.0 로그인 페이지로 연결되는 링크가 포함되어 있습니다.

4단계: AppStream 2.0에 로그인

이 단계에서는 3단계: AppStream 2.0에서 사용자 생성에서 생성한 사용자에게 AppStream 2.0이 보낸 이메일의 링크를 사용하여 AppStream 2.0에 로그인합니다.

1. AppStream 2.0에서 보낸 이메일에 제공된 링크를 사용하여 AppStream 2.0에 로그인합니다.
2. 암호를 변경하라는 메시지가 표시되면 암호를 변경하세요. 표시되는 AppStream 2.0 화면은 다음과 비슷합니다.

3. 데스크톱을 선택합니다.
4. 작업 표시줄에서 검색을 선택하고 **`D:`**를 입력하여 홈 폴더로 이동합니다.

> **Note**
>
> 이 단계를 건너뛰면 홈 폴더에 액세스하려고 할 때 `Device not ready` 오류가 발생할 수 있습니다.

AppStream 2.0에 로그인하는 데 문제가 있는 경우 언제든지 다음 단계를 사용하여 AppStream 2.0 플릿을 다시 시작하고 로그인을 다시 시도할 수 있습니다.

1. AppStream 2.0 콘솔을 엽니다.
2. 왼쪽 탐색에서 플릿을 선택합니다.
3. 사용하려는 플릿을 선택합니다.
4. 작업을 선택한 다음 중지를 선택합니다.
5. 플릿이 멈출 때까지 기다리세요.
6. 작업을 선택한 다음 시작을 선택합니다.

이 프로세스는 약 10분 정도 걸릴 수 있습니다.

5단계: Amazon S3의 버킷 확인(선택 사항)

스택을 생성하는 데 사용한 AWS CloudFormation 템플릿으로 완료한 작업 중 하나는 Amazon S3에 두 개의 버킷을 생성하는 것이었습니다. 이 버킷은 작업 세션 전반에 걸쳐 사용자 데이터와 애플리케이션 설정을 저장하고 복원하는 데 필요합니다. 이러한 버킷은 다음과 같습니다.

- 이름은 `appstream2-`로 시작합니다. 이 버킷은 AppStream 2.0(`D:\PhotonUser\My Files\Home Folder`)의 홈 폴더에 데이터를 매핑합니다.

  > **Note**
  >
  > 홈 폴더는 특정 이메일 주소별로 고유하며 해당 AWS 계정의 모든 플릿과 스택에서 공유됩니다. 홈 폴더의 이름은 사용자 이메일 주소의 SHA256 해시이며, 해당 해시를 기반으로 한 경로에 저장됩니다.

- 이름은 `appstream-app-settings-`로 시작합니다. 이 버킷에는 AppStream 2.0의 사용자 세션 정보가 들어 있으며 브라우저 즐겨찾기, IDE 및 애플리케이션 연결 프로필, UI 사용자 지정과 같은 설정이 포함됩니다. 자세한 내용은 Amazon AppStream 2.0 관리 안내서의 홈 애플리케이션 설정 지속성 작업을 참조하세요.

버킷이 생성되었는지 확인하려면 다음 단계를 따르세요.

1. Amazon S3 콘솔을 엽니다.
2. 왼쪽 탐색에서 버킷을 선택합니다.
3. 이름으로 버킷 찾기에서 **`appstream`**을 입력하여 목록을 필터링합니다.

버킷이 표시되면 추가 조치가 필요하지 않습니다. 단, 버킷이 존재한다는 사실만 알아두세요. 버킷이 보이지 않으면 AWS CloudFormation 템플릿 실행이 완료되지 않았거나 오류가 발생한 것입니다. AWS CloudFormation 콘솔로 이동하여 스택 생성 메시지를 검토하세요.

다음 단계

이제 AppStream 2.0 인프라가 설정되었으므로 Enterprise Analyzer를 설정하고 사용을 시작할 수 있습니다. 자세한 내용은 자습서: AppStream 2.0에서 Enterprise Analyzer 설정 섹션을 참조하세요. Enterprise Developer를 설정할 수도 있습니다. 자세한 내용은 자습서: AppStream 2.0에서 Micro Focus Enterprise Developer 설정 섹션을 참조하세요.

리소스 정리

생성된 스택과 플릿을 정리하는 절차는 AppStream 2.0 플릿 및 스택 생성에 설명되어 있습니다.

AppStream 2.0 객체가 삭제되면 계정 관리자는 필요에 따라 애플리케이션 설정 및 홈 폴더에 대한 Amazon S3 버킷을 정리할 수도 있습니다.

> **Note**
>
> 특정 사용자의 홈 폴더는 모든 플릿에서 고유하므로 동일한 계정에서 다른 AppStream 2.0 스택이 활성 상태인 경우 홈 폴더를 유지해야 할 수 있습니다.

마지막으로, AppStream 2.0에서는 현재 콘솔을 사용하여 사용자를 삭제할 수 없습니다. 대신 CLI와 함께 서비스 API를 사용해야 합니다. 자세한 내용은 Amazon AppStream 2.0 관리 안내서의 사용자 풀 관리를 참조하세요.

자습서: AppStream 2.0에서 Enterprise Analyzer 설정

이 자습서에서는 Micro Focus Enterprise Analyzer를 설정하여 하나 이상의 Mainframe 애플리케이션을 분석하는 방법을 설명합니다. Enterprise Analyzer 도구는 애플리케이션 소스 코드 및 시스템 정의에 대한 분석을 기반으로 여러 보고서를 제공합니다.

이 설정은 팀 협업을 촉진하기 위해 설계되었습니다. 설치 시 Amazon S3 버킷을 사용하여 소스 코드를 가상 디스크와 공유합니다. 이렇게 하면 Windows 시스템에서 Rclone 을 사용할 수 있습니다. PostgreSQL을 실행하는 일반 Amazon RDS 인스턴스를 사용하면 팀원 누구나 요청된 모든 보고서에 액세스할 수 있습니다.

또한 팀 구성원은 가상 Amazon S3 백업 디스크를 개인 컴퓨터에 마운트하고 워크스테이션에서 원본 버킷을 업데이트할 수 있습니다. 다른 온프레미스 내부 시스템에 연결되어 있는 경우 해당 컴퓨터에서 스크립트나 다른 형태의 자동화를 사용할 수 있습니다.

설정은 AWS Mainframe Modernization이 고객과 공유하는 AppStream 2.0 Windows 이미지를 기반으로 합니다. 또한 설치는 튜토리얼: Micro Focus Enterprise Analyzer 및 Micro Focus Enterprise Developer를 통해 AppStream 2.0 설정하기에 설명된 대로 AppStream 2.0 플릿 및 스택의 생성을 기반으로 합니다.

> **Important**
>
> 이 자습서의 단계에서는 다운로드 가능한 AWS CloudFormation 템플릿 cfn-m2-appstream-fleet-ea-ed.yml을 사용하여 AppStream 2.0을 설정한다고 가정합니다. 자세한 내용은 튜토리얼: Micro Focus Enterprise Analyzer 및 Micro Focus Enterprise Developer를 통해 AppStream 2.0 설정하기 섹션을 참조하세요.
> 이 자습서의 단계를 수행하려면 Enterprise Analyzer 플릿 및 스택을 설정하고 실행 중이어야 합니다.

Enterprise Analyzer 기능 및 결과물에 대한 전체 설명은 Micro Focus 웹 사이트의 Enterprise Analyzer 설명서를 참조하세요.

이미지 콘텐츠

Enterprise Analyzer 애플리케이션 자체 외에도 이미지에는 다음과 같은 도구 및 라이브러리가 포함되어 있습니다.

타사 도구

- Python
- Rclone
- pgAdmin
- git-scm
- PostgreSQL ODBC 드라이버

`C:\Users\Public`에 있는 라이브러리

- Enterprise Developer를 위한 BankDemo 소스 코드 및 프로젝트 정의: `m2-bankdemo-template.zip`.
- Mainframe용 MFA 설치 패키지: `mfa.zip`. 자세한 내용은 Micro Focus Enterprise Developer 설명서의 Mainframe 액세스 개요를 참조하세요.
- Rclone용 명령 및 구성 파일(자습서에서의 사용 지침): `m2-rclone.cmd` 및 `m2-rclone.conf`.

주제

- 사전 조건

사전 조건

- 분석하려는 고객 애플리케이션의 소스 코드와 시스템 정의를 S3 버킷에 업로드합니다. 시스템 정의에는 CICS CSD, DB2 객체 정의 등이 포함됩니다. 애플리케이션 아티팩트를 구성하려는 방식에 적합한 폴더 구조를 버킷 내에 만들 수 있습니다. 예를 들어 BankDemo 샘플의 압축을 풀면 다음과 같은 구조가 됩니다.

```
demo
     |--> jcl
     |--> RDEF
     |--> transaction
     |--> xa
```

- PostgreSQL을 실행하는 Amazon RDS 인스턴스를 생성 및 시작합니다. 이 인스턴스는 Enterprise Analyzer에서 생성된 데이터와 결과를 저장합니다. 이 인스턴스를 애플리케이션 팀의 모든 구성원과 공유할 수 있습니다. 또한 데이터베이스에 `m2_ea`라는(또는 다른 적절한 이름) 빈 스키마를 만드세요. 인증된 사용자가 이 스키마에서 항목을 생성, 삽입, 업데이트 및 삭제할 수 있도록 하는 자격 증명을 정의하세요. Amazon RDS 콘솔 또는 계정 관리자로부터 데이터베이스 이름, 서버 엔드포인트 URL 및 TCP 포트를 얻을 수 있습니다.
- AWS 계정에 프로그래밍 방식으로 액세스할 수 있도록 설정했는지 확인하세요. 자세한 내용은 Amazon Web Services 일반 참조에서 프라그매틱 액세스를 참조하세요.

1단계: 설정

1. AppStream 2.0에서 보낸 환영 이메일 메시지에서 받은 URL을 사용하여 AppStream 2.0을 사용하여 세션을 시작하세요.
2. 이메일을 사용자 ID로 사용하고 영구 암호를 정의하세요.
3. Enterprise Analyzer 스택을 선택합니다.

4. AppStream 2.0 메뉴 페이지에서 데스크톱을 선택하여 플릿이 스트리밍하는 Windows 데스크톱에 도달합니다.

2단계: Amazon S3 기반 가상 폴더를 Windows에서 생성

> **Note**
>
> AWS Mainframe Modernization 미리 보기에서 이미 Rclone을 사용한 경우 `C:\Users\Public`에 있는 최신 버전으로 `m2-rclone.cmd`를 업데이트해야 합니다.

1. 파일 탐색기를 사용하여 제공된 `m2-rclone.conf` 및 `m2-rclone.cmd` 파일을 홈 폴더 `C:\Users\PhotonUser\My Files\Home Folder`에 복사합니다.
2. AWS 리전와 AWS 액세스 키 및 해당 비밀번호로 `m2-rclone.conf` 구성 매개 변수를 업데이트하세요.

```
[m2-s3]
type = s3
provider = AWS
access_key_id = YOUR-ACCESS-KEY
secret_access_key = YOUR-SECRET-KEY
region = YOUR-REGION
acl = private
server_side_encryption = AES256
```

3. `m2-rclone.cmd`에서 다음과 같이 변경합니다.

   - `your-s3-bucket`를 Amazon S3 버킷의 이름으로 변경합니다. 예: `m2-s3-mybucket`.
   - Amazon S3 버킷 키로 `your-s3-folder-key`를 변경합니다. 예: `myProject`.
   - 애플리케이션 파일이 포함된 Amazon S3 버킷에서 애플리케이션 파일을 동기화하려는 디렉토리의 경로로 `your-local-folder-path`를 변경합니다. 예: `D:\PhotonUser\My Files\Home Folder\m2-new`. AppStream 2.0에서 세션 시작 및 종료 시 제대로 백업 및 복원하려면 이 동기화된 디렉터리가 홈 폴더의 하위 디렉터리여야 합니다.

```
:loop
timeout /T 10
```

```
"C:\Program Files\rclone\rclone.exe" sync m2-s3:your-s3-bucket/your-s3-folder-
key "D:\PhotonUser\My Files\Home Folder\your-local-folder-path" --config "D:
\PhotonUser\My Files\Home Folder\m2-rclone.conf"
goto :loop
```

4. cd to `C:\Users\PhotonUser\My Files\Home Folder` Windows 명령 프롬프트를 열고 필요한 경우 `m2-rclone.cmd`를 실행합니다. 이 명령 스크립트는 연속 루프를 실행하여 Amazon S3 버킷과 키를 10초마다 로컬 폴더에 동기화합니다. 필요에 따라 타임아웃을 조정할 수 있습니다. Windows 파일 탐색기의 Amazon S3 버킷에 있는 애플리케이션의 소스 코드를 확인해야 합니다.

작업 중인 세트에 새 파일을 추가하거나 기존 파일을 업데이트하려면 Amazon S3 버킷에 파일을 업로드하세요. 그러면 파일이`m2-rclone.cmd` 에서 정의한 다음 반복에서 디렉터리에 동기화됩니다. 마찬가지로 일부 파일을 삭제하려면 Amazon S3 버킷에서 삭제합니다. 다음 동기화 작업에서는 로컬 디렉터리에서 파일이 삭제됩니다.

3단계: Amazon RDS 인스턴스용 ODBC 소스 생성

1. EA_Admin 도구를 시작하려면 브라우저 창 왼쪽 상단의 애플리케이션 선택기 메뉴로 이동하여 MF EA_Admin을 선택합니다.
2. 관리 메뉴에서 ODBC 데이터 소스를 선택하고 사용자 DSN 탭에서 추가를 선택합니다.
3. Create New Data Source(새 데이터 소스 생성) 대화 상자에서 PostgreSQL Unicode 드라이버를 선택한 다음 Finish(마침)를 선택합니다.
4. PostgreSQL 유니코드 ODBC 드라이버(psqlODBC) 설치 대화 상자에서 원하는 데이터 소스 이름을 정의하고 기록해 둡니다. 이전에 만든 RDS 인스턴스의 값을 사용하여 다음 파라미터를 완성하세요.

   Description

   이 데이터베이스 연결을 빠르게 식별하는 데 도움이 되는 선택적 설명입니다.

   데이터베이스

   앞에서 만든 Amazon RDS 데이터베이스.

   Server

   Amazon RDS 엔드포인트.

   Port

   Amazon RDS 포트.

사용자 이름

Amazon RDS 인스턴스에 정의되어 있습니다.

암호

Amazon RDS 인스턴스에 정의되어 있습니다.

5. 테스트를 선택하여 Amazon RDS에 성공적으로 연결되었는지 확인한 다음 저장을 선택하여 새 사용자 DSN을 저장합니다.
6. 적절한 작업 영역 생성을 확인하는 메시지가 표시될 때까지 기다린 다음 확인을 선택하여 ODBC 데이터 소스 사용을 끝내고 EA_Admin 도구를 닫습니다.
7. 다시 응용 프로그램 선택기 메뉴로 이동한 다음 Enterprise Analyzer를 선택하여 도구를 시작합니다. 새로 생성을 선택합니다.
8. 작업 영역 구성 창에서 작업 영역 이름을 입력하고 위치를 정의합니다. 이 구성에서 작업하는 경우 작업 공간은 Amazon S3 기반 디스크가 될 수 있고, 원하는 경우 홈 폴더가 될 수 있습니다.
9. 기타 데이터베이스 선택을 선택하여 Amazon RDS 인스턴스에 연결합니다.
10. 옵션에서 Postgre 아이콘을 선택한 다음 확인을 선택합니다.
11. Windows 설정의 경우 옵션 - 연결 매개 변수 정의에서 생성한 데이터 소스의 이름을 입력합니다. 데이터베이스 이름, 스키마 이름, 사용자 이름 및 암호도 입력합니다. 확인을 선택합니다.
12. Enterprise Analyzer가 결과를 저장하는 데 필요한 모든 테이블, 인덱스 등을 생성할 때까지 기다리세요. 이 단계는 몇 분이 걸릴 수 있습니다. Enterprise Analyzer는 데이터베이스와 작업 영역을 사용할 준비가 되면 이를 확인합니다.
13. 애플리케이션 선택기 메뉴로 다시 이동한 다음 Enterprise Analyzer를 선택하여 도구를 시작합니다.
14. 새로 선택한 작업 영역 위치에 Enterprise Analyzer 시작 창이 나타납니다. 확인을 선택합니다.
15. 왼쪽 창에서 리포지토리로 이동하여 리포지토리 이름을 선택한 다음 작업 영역에 파일/폴더 추가를 선택합니다. 애플리케이션 코드가 저장된 폴더를 선택하여 작업 영역에 추가합니다. 원하는 경우 이전 BankDemo 예제 코드를 사용할 수 있습니다. Enterprise Analyzer에서 해당 파일을 확인하라는 메시지가 표시되면 확인을 선택하여 초기 Enterprise Analyzer 확인 보고서를 시작합니다. 신청 규모에 따라 완료하는 데 몇 분이 소요될 수 있습니다.
16. 작업 영역을 확장하여 작업 영역에 추가한 파일 및 폴더를 확인하세요. 차트 뷰어 창의 상단 사분면에도 객체 유형과 순환 복잡성 보고서가 표시됩니다.

이제 Enterprise Analyzer를 사용하여 필요한 모든 작업을 수행할 수 있습니다.

후속 세션

1. AppStream 2.0에서 보낸 환영 이메일 메시지에서 받은 URL을 사용하여 AppStream 2.0을 사용하여 세션을 시작하세요.
2. 이메일과 영구 암호를 사용하여 로그인합니다.
3. Enterprise Analyzer 스택을 선택합니다.
4. 이 옵션을 사용하여 작업 공간 파일을 공유하는 경우 Amazon S3 백업 디스크에 연결하기 위해 `Rclone`를 시작하세요.
5. Enterprise Analyzer를 실행하여 작업을 수행하세요.

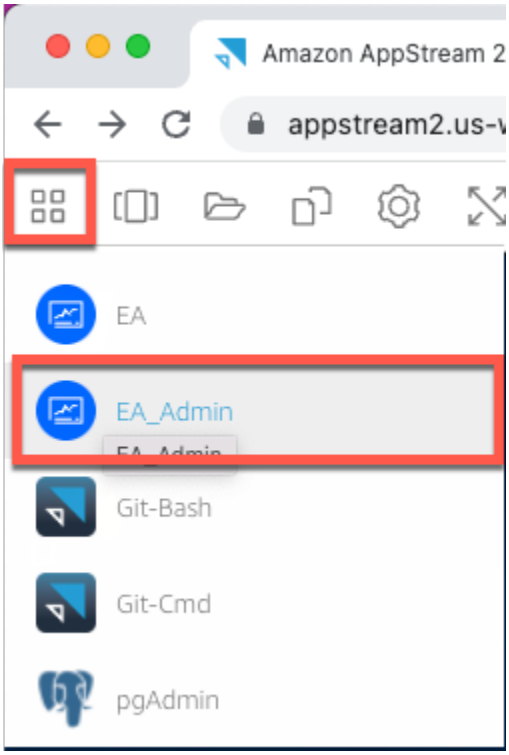
작업 영역 연결 문제 해결

Enterprise Analyzer 작업 영역에 다시 연결하려고 하면 다음과 같은 오류가 표시될 수 있습니다.

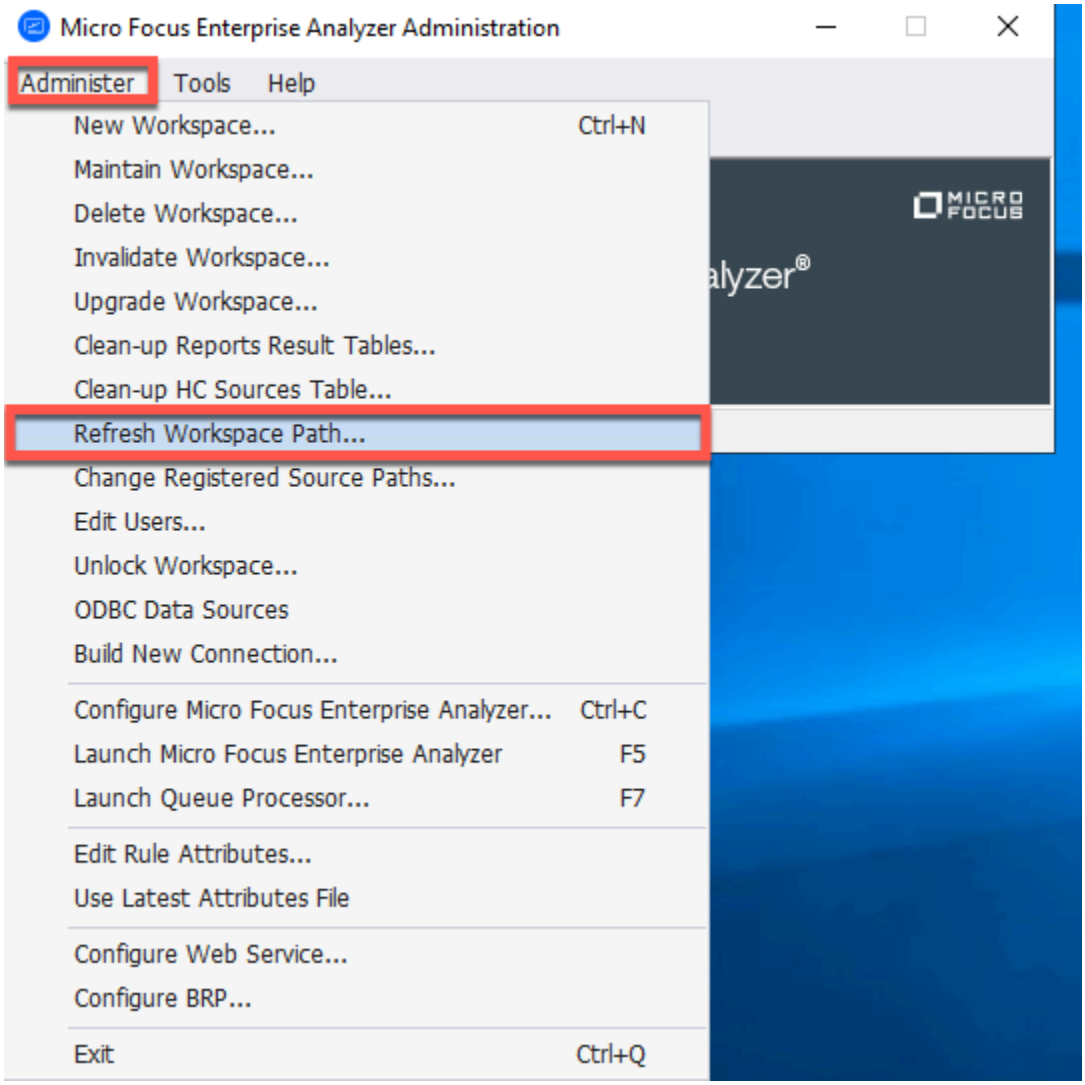
```
Cannot access the workspace directory D:\PhotonUser\My Files\Home Folder\EA_BankDemo.
 The workspace has been created on a non-shared disk of the EC2AMAZ-E6LC33H computer.
 Would you like to correct the workspace directory location?
```

이 문제를 해결하려면 확인을 선택하여 메시지를 지우고 다음 단계를 완료하세요.

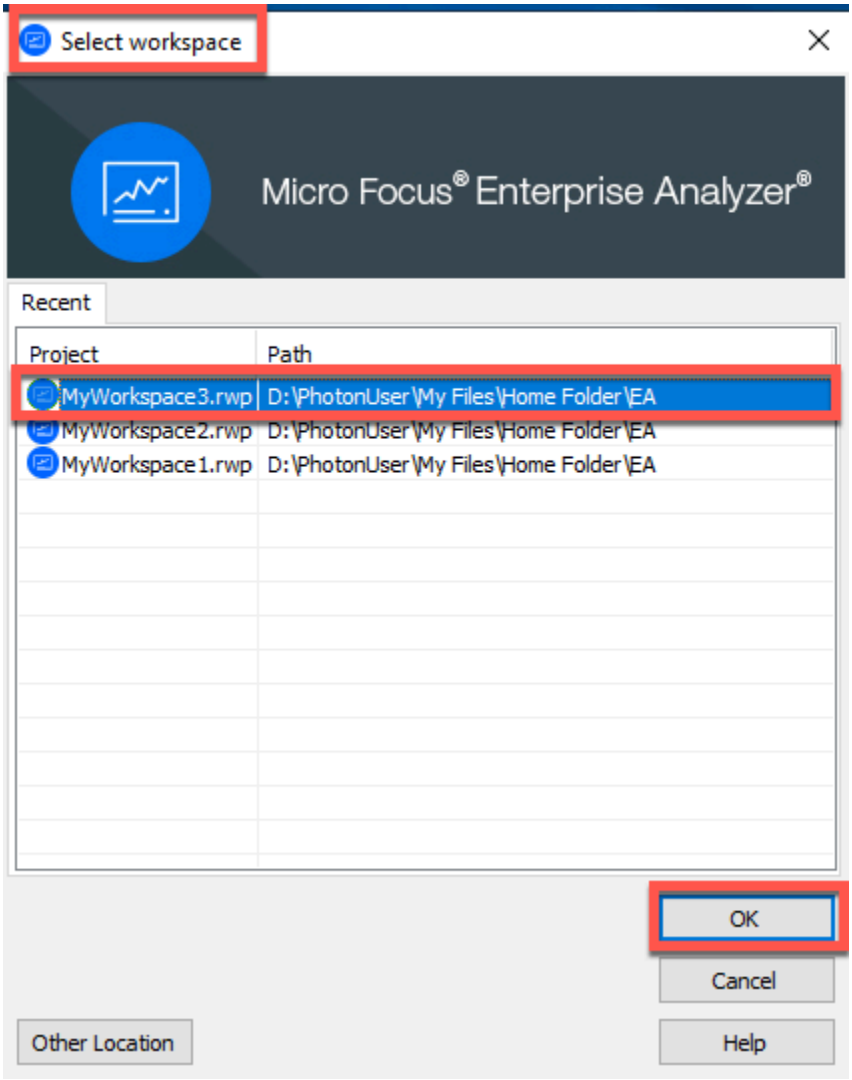
1. AppStream 2.0의 경우 도구 모음에서 애플리케이션 실행 아이콘을 선택한 다음 EA_Admin을 선택하여 Micro Focus Enterprise Analyzer 관리 도구를 시작합니다.



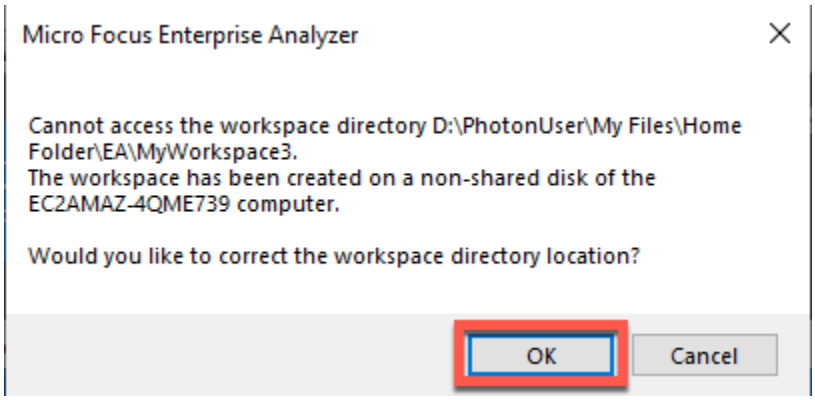
2. 관리 메뉴에서 작업 영역 경로 새로 고침...을 선택합니다.



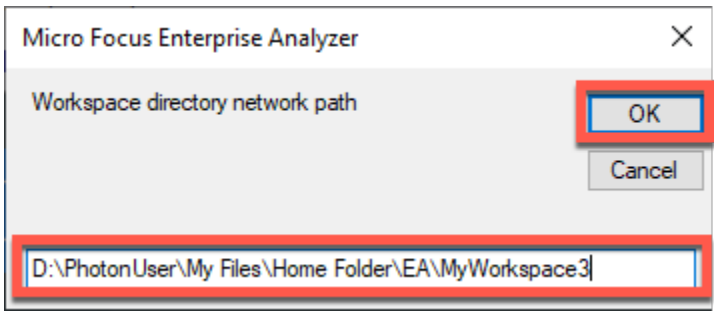
3. 작업 영역 선택에서 원하는 작업 영역을 선택한 다음 확인을 선택합니다.



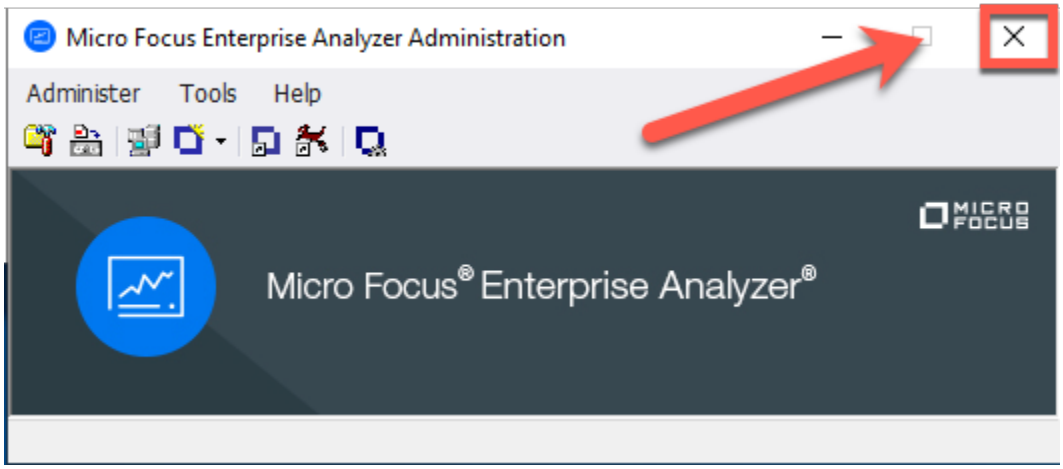
4. 확인을 선택하여 오류 메시지를 확인합니다.



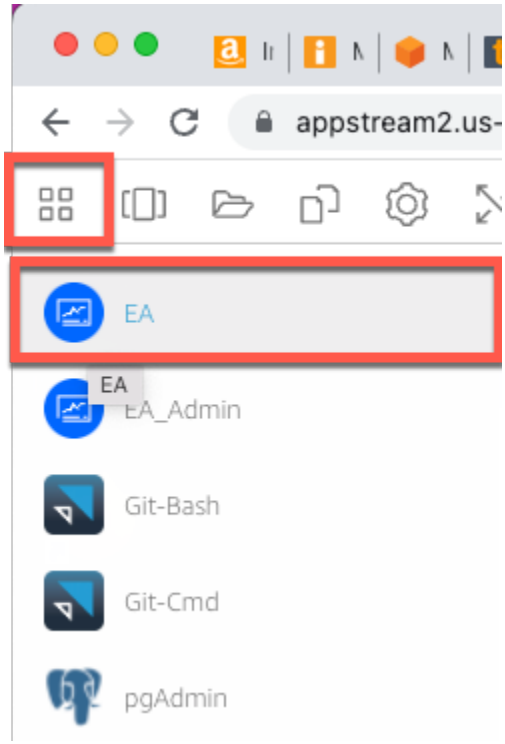
5. 작업 영역 디렉토리 네트워크 경로에서 작업 영역에 대한 올바른 경로를 입력합니다(예: `D:\PhotonUser\My Files\Home Folder\EA\MyWorkspace3`).



6. Micro Focus Enterprise Analyzer 관리 도구를 닫습니다.



7. AppStream 2.0의 경우 도구 모음에서 애플리케이션 실행 아이콘을 선택한 다음 EA를 선택하여 Micro Focus Enterprise Analyzer를 시작합니다.



8. 3~5단계를 반복합니다.

이제 Micro Focus Enterprise Analyzer가 기존 작업 영역이 함께 열릴 것입니다.

리소스 정리

이 자습서에 사용할 때 생성한 리소스가 더 이상 필요하지 않은 경우 삭제하여 추가 요금이 발생하지 않도록 하세요. 다음 단계를 완료합니다.

- EA_Admin 도구를 사용하여 작업 영역을 삭제하세요.
- 이 자습서를 위해 생성한 S3 버킷을 삭제합니다. 자세한 내용을 알아보려면 Amazon S3 사용 설명서의 버킷 삭제를 참조하세요.
- 이 자습서용으로 생성한 데이터베이스를 선택합니다. 자세한 내용은 DB 인스턴스 삭제를 참조하세요.

자습서: AppStream 2.0에서 Micro Focus Enterprise Developer 설정

이 자습서에서는 Enterprise Developer 기능을 사용하여 하나 이상의 메인프레임 애플리케이션을 유지 관리, 컴파일 및 테스트하기 위해 Micro Focus Enterprise Developer를 설정하는 방법을 설명합니다. 설정은 AWS Mainframe Modernization이 고객과 공유하고 AppStream 2.0 플릿 및 스택의 생성을 기반으로 합니다.

> **Important**
>
> 이 자습서의 단계에서는 다운로드 가능한 AWS CloudFormation 템플릿 cfn-m2-appstream-fleet-ea-ed.yaml을 사용하여 AppStream 2.0을 설정한다고 가정합니다. 자세한 내용은 튜토리얼: Micro Focus Enterprise Analyzer 및 Micro Focus Enterprise Developer를 통해 AppStream 2.0 설정하기 섹션을 참조하세요.
> Enterprise Developer 플릿 및 스택이 가동되고 실행될 때 이 설정의 단계를 수행해야 합니다.

Enterprise Developer v7 기능 및 결과물에 대한 전체 설명을 보려면 Micro Focus 사이트에서 최신 온라인 설명서(v7.0)를 확인하세요.

이미지 컨텐츠

이미지에는 Enterprise Developer 자체 외에도 Rumba(TN3270 에뮬레이터)가 포함되어 있습니다. 다음과 같은 도구 및 라이브러리도 포함되어 있습니다.

타사 도구

- Python
- Rclone
- pgAdmin
- git-scm
- PostgreSQL ODBC 드라이버

`C:\Users\Public`에 있는 라이브러리

- Enterprise Developer를 위한 BankDemo 소스 코드 및 프로젝트 정의: `m2-bankdemo-template.zip`.
- Mainframe용 MFA 설치 패키지: `mfa.zip`. 자세한 내용은 Micro Focus Enterprise Developer 설명서의 Mainframe 액세스 개요를 참조하세요.
- Rclone용 명령 및 구성 파일(자습서에서의 사용 지침): `m2-rclone.cmd` 및 `m2-rclone.conf`.

CodeCommit 리포지토리에 아직 로드되지 않았지만 Amazon S3 버킷에서 사용할 수 있는 소스 코드에 액세스해야 하는 경우(예: 소스 코드를 git에 처음 로드하기 위해) 자습서: AppStream 2.0에서 Enterprise Analyzer 설정에 설명된 절차에 따라 가상 Windows 디스크를 생성합니다.

주제

- 사전 조건
- 1단계: 개별 Enterprise Developer 사용자에 의한 구성
- 2단계: Windows에서 Amazon S3 기반 가상 폴더 생성(선택 사항)
- 3단계: 리포지토리 복제
- 후속 세션
- 리소스 정리

사전 조건

- 유지 관리할 애플리케이션의 소스 코드와 함께 로드된 하나 이상의 CodeCommit 리포지토리. 리포지토리 설정은 위의 CI/CD 파이프라인 요구 사항과 일치해야 두 도구를 조합하여 시너지 효과를 낼 수 있습니다.

- 각 사용자는 AWS CodeCommit의 인증 및 액세스 제어에 있는 정보에 따라 계정 관리자가 정의한 CodeCommit 리포지토리 또는 리포지토리에 대한 보안 인증을 가지고 있어야 합니다. 이러한 보안 인증의 구조는 AWS CodeCommit의 인증 및 액세스 제어에서 검토되며 CodeCommit의 IAM 인증에 대한 전체 참조는 CodeCommit 권한 참조에 있습니다. 관리자는 각 리포지토리의 역할별 보안 인증을 사용하고 해당 저장소에서 수행해야 하는 특정 작업 집합으로 사용자의 권한 부여를 제한하여 고유한 역할에 대해 고유한 IAM 정책을 정의할 수 있습니다. 따라서 CodeCommit 리포지토리의 각 유지 관리자에 대해 계정 관리자는 기본 사용자를 생성하고 CodeCommit 액세스에 적합한 IAM 정책 또는 정책을 선택하여 이 사용자에게 필요한 리포지토리에 액세스할 수 있는 권한을 부여합니다.

1단계: 개별 Enterprise Developer 사용자에 의한 구성

1. IAM 보안 인증 구성:
    1. https://console.aws.amazon.com/iam/에서 AWS 콘솔로 이동합니다.
    2. AWS CodeCommit 사용 설명서의 Git 보안 인증을 사용하는 HTTPS 사용자용 설정의 3단계에 설명된 절차를 따르세요.
    3. IAM이 생성한 CodeCommit 관련 로그인 보안 인증을 복사합니다. 이 정보를 표시하고 복사한 다음 로컬 컴퓨터의 보안 파일에 붙여넣거나 보안 인증 다운로드를 선택하여 이 정보를.CSV 파일로 다운로드합니다. CodeCommit에 접속하려면 이 정보가 필요합니다.
2. 환영 이메일에 수신된 URL을 기반으로 AppStream 2.0으로 세션을 시작하세요. 이메일을 사용자 이름으로 사용하고 비밀번호를 생성하세요.
3. Enterprise Developer 스택을 선택합니다.
4. 메뉴 페이지에서 데스크톱을 선택하면 플릿이 스트리밍하는 Windows 데스크톱에 도달합니다.

2단계: Windows에서 Amazon S3 기반 가상 폴더 생성(선택 사항)

Rclone이 필요한 경우(위 참조) Windows에서 Amazon S3 기반 가상 폴더를 생성하세요(모든 애플리케이션 아티팩트가 CodeCommit 액세스에서만 제공되는 경우 선택 사항).

> **Note**
>
> AWS Mainframe Modernization 프리뷰에서 이미 Rclone을 사용한 경우 `C:\Users\Public`에 있는 최신 버전으로 `m2-rclone.cmd`를 업데이트해야 합니다.

1. 파일 탐색기를 사용하여 제공된 m2-rclone.conf 및 m2-rclone.cmd 파일을 홈 폴더 C:\Users\PhotonUser\My Files\Home Folder에 복사합니다.
2. AWS 리전와 AWS 액세스 키 및 해당 비밀번호로 m2-rclone.conf 구성 매개 변수를 업데이트하세요.

```
[m2-s3]
type = s3
provider = AWS
access_key_id = YOUR-ACCESS-KEY
secret_access_key = YOUR-SECRET-KEY
region = YOUR-REGION
acl = private
server_side_encryption = AES256
```

3. m2-rclone.cmd에서 다음과 같이 변경합니다.

   - your-s3-bucket를 Amazon S3 버킷의 이름으로 변경합니다. 예: m2-s3-mybucket.
   - Amazon S3 버킷 키로 your-s3-folder-key를 변경합니다. 예: myProject.
   - 애플리케이션 파일이 포함된 Amazon S3 버킷에서 애플리케이션 파일을 동기화하려는 디렉토리의 경로로 your-local-folder-path를 변경합니다. 예: D:\PhotonUser\My Files\Home Folder\m2-new. AppStream 2.0에서 세션 시작 및 종료 시 제대로 백업 및 복원하려면 이 동기화된 디렉터리가 홈 폴더의 하위 디렉터리여야 합니다.

```
:loop
timeout /T 10
"C:\Program Files\rclone\rclone.exe" sync m2-s3:your-s3-bucket/your-s3-folder-
key "D:\PhotonUser\My Files\Home Folder\your-local-folder-path" --config "D:
\PhotonUser\My Files\Home Folder\m2-rclone.conf"
goto :loop
```

4. cd to C:\Users\PhotonUser\My Files\Home Folder Windows 명령 프롬프트를 열고 필요한 경우 m2-rclone.cmd를 실행합니다. 이 명령 스크립트는 연속 루프를 실행하여 Amazon S3 버킷과 키를 10초마다 로컬 폴더에 동기화합니다. 필요에 따라 타임아웃을 조정할 수 있습니다. Windows 파일 탐색기의 Amazon S3 버킷에 있는 애플리케이션의 소스 코드를 확인해야 합니다.

작업 중인 세트에 새 파일을 추가하거나 기존 파일을 업데이트하려면 Amazon S3 버킷에 파일을 업로드하세요. 그러면 파일이m2-rclone.cmd 에서 정의한 다음 반복에서 디렉터리에 동기화됩니다. 마찬

가지로 일부 파일을 삭제하려면 Amazon S3 버킷에서 삭제합니다. 다음 동기화 작업에서는 로컬 디렉터리에서 파일이 삭제됩니다.

3단계: 리포지토리 복제

1. 브라우저 창의 왼쪽 상단에 있는 애플리케이션 선택기 메뉴로 이동하여 Enterprise Developer를 선택합니다.
2. 작업 영역 위치로 `C:\Users\PhotonUser\My Files\Home Folder`(일명`D: \PhotonUser \My Files\Home Folder`)를 선택하여 홈 폴더에서 Enterprise Developer에 필요한 작업 영역 생성을 완료하세요.
3. Enterprise Developer에서는 프로젝트 탐색기로 이동하여 CodeCommit 리포지토리를 복제하고 가져오기, ... 가져오기, Git, Git 복제 URI에서프로젝트를 선택합니다. 그런 다음 CodeCommit별 로그인 보안 인증을 입력하고 Eclipse 대화 상자를 완료하여 코드를 가져옵니다.

CodeCommit git 리포지토리가 이제 로컬 작업 영역에 복제되었습니다.

이제 Enterprise Developer 작업 영역이 애플리케이션에 대한 유지 관리 작업을 시작할 준비가 되었습니다. 특히 Enterprise Developer와 통합된 Microfocus Enterprise Server(ES)의 로컬 인스턴스를 사용하여 애플리케이션을 대화식으로 디버깅하고 실행하여 변경 사항을 로컬에서 검증할 수 있습니다.

> **Note**
>
> 로컬 Enterprise Server 인스턴스를 포함한 로컬 Enterprise Developer 환경은 Windows에서 실행되는 반면 AWS Mainframe Modernization은 Linux에서 실행됩니다. 새 애플리케이션을 CodeCommit에 커밋하고 이 타겟에 맞게 다시 빌드한 후 새 애플리케이션을 프로덕션에 롤아웃하기 전에 AWS Mainframe Modernization에서 제공하는 Linux 환경에서 보완 테스트를 실행하는 것이 좋습니다.

후속 세션

CodeCommit 리포지토리를 복제하기 위한 홈 폴더와 같이 AppStream 2.0에서 관리하는 폴더를 선택하면 세션 간에 투명하게 저장 및 복원됩니다. 다음에 애플리케이션으로 작업해야 할 때 다음 단계를 완료합니다.

1. 환영 이메일에 수신된 URL을 기반으로 AppStream 2.0으로 세션을 시작하세요.
2. 이메일과 영구 암호로 로그인합니다.

3. Enterprise Developer 스택을 선택합니다.
4. 이 옵션을 사용하여 작업 영역 파일을 공유하는 경우 `Rclone`를 시작하여 Amazon S3 지원 디스크에 연결하세요(위 참조).
5. Enterprise Developer를 실행하여 작업을 수행하세요.

리소스 정리

이 튜토리얼에서 만든 리소스가 더 이상 필요하지 않은 경우 비용이 청구되지 않도록 하려면 리소스에 대해 비용이 청구되지 않도록 하세요. 다음 단계를 완료합니다.

- 이 자습서에서 만든 CodeCommit 리포지토리를 삭제합니다. 자세한 내용은 사용 AWS CodeCommit 사용 안내서의 CodeCommit 리포지토리 삭제를 참조하세요.
- 이 자습서용으로 생성한 데이터베이스를 삭제합니다. 자세한 내용은 DB 인스턴스 삭제를 참조하세요.

Micro Focus Enterprise Analyzer 및 Micro Focus Enterprise Developer 스트리밍 세션에 대한 자동화 설정

세션 시작 및 종료 시 스크립트를 자동으로 실행하여 고객 상황에 맞는 자동화를 허용할 수 있습니다. 이 AppStream 2.0 기능에 대한 자세한 내용은 Amazon AppStream 2.0 관리 가이드의 세션 스크립트를 사용하여 AppStream 2.0 사용자의 스트리밍 환경 관리를 참조하세요.

이 기능을 사용하려면 최소한 다음 버전의 Enterprise Analyzer 및 Enterprise Developer 이미지가 있어야 합니다.

- `m2-enterprise-analyzer-v8.0.4.R1`
- `m2-enterprise-developer-v8.0.4.R1`

주제

- 세션 시작 시 자동화 설정
- 세션 종료 시 자동화 설정

세션 시작 시 자동화 설정

사용자가 AppStream 2.0에 연결할 때 자동화 스크립트를 실행하려면 스크립트를 만들고 이름을 `m2-user-setup.cmd`로 지정하세요. AppStream 2.0 홈 폴더에 스크립트를 사용자의 AppStream 2.0 홈 폴더에 저장합니다. AWS Mainframe Modernization이 제공하는 AppStream 2.0 이미지는 해당 위치에서 해당 이름의 스크립트를 찾아 해당 이름이 있는 경우 실행합니다.

> **Note**
>
> 스크립트 지속 시간은 AppStream 2.0에서 설정한 제한(현재 60초)을 초과할 수 없습니다. 자세한 내용은 Amazon AppStream 2.0 관리 안내서의 스트리밍 세션 시작 전 스크립트 실행을 참조하세요.

세션 종료 시 자동화 설정

사용자가 AppStream 2.0과의 연결을 끊을 때 자동화 스크립트를 실행하려면 스크립트를 만들고 이름을 `m2-user-teardown.cmd`로 지정하세요. AppStream 2.0 홈 폴더에 스크립트를 사용자의 AppStream 2.0 홈 폴더에 저장합니다. AWS Mainframe Modernization이 제공하는 AppStream 2.0 이미지는 해당 위치에서 해당 이름의 스크립트를 찾아 해당 이름이 있는 경우 실행합니다.

> **Note**
>
> 스크립트 지속 시간은 AppStream 2.0에서 설정한 제한(현재 60초)을 초과할 수 없습니다. 자세한 내용은 Amazon AppStream 2.0 관리 안내서의 스트리밍 세션 종료 후 스크립트 실행을 참조하세요.

Enterprise Developer에서 데이터 세트를 테이블 및 열로 보기

Micro Focus 런타임을 사용하여 Mainframe Modernization에 배포된 AWS Mainframe 데이터 세트에 액세스할 수 있습니다. Micro Focus Enterprise Developer 인스턴스에서 마이그레이션된 데이터 세트를 테이블 및 열로 볼 수 있습니다. 데이터 세트를 이 방법으로 보면 다음을 수행할 수 있습니다.

- 마이그레이션된 데이터 파일에 대한 `SQL SELECT` 작업을 수행합니다.
- 애플리케이션을 변경하지 않고 마이그레이션된 Mainframe 애플리케이션 외부에 데이터를 노출할 수 있습니다.
- 데이터를 쉽게 필터링하고 CSV 또는 기타 파일 형식으로 저장할 수 있습니다.

> **Note**
>
> 1단계와 2단계는 일회성 활동입니다. 각 데이터 세트에 대해 3단계와 4단계를 반복하여 데이터베이스 뷰를 생성합니다.

주제

사전 조건

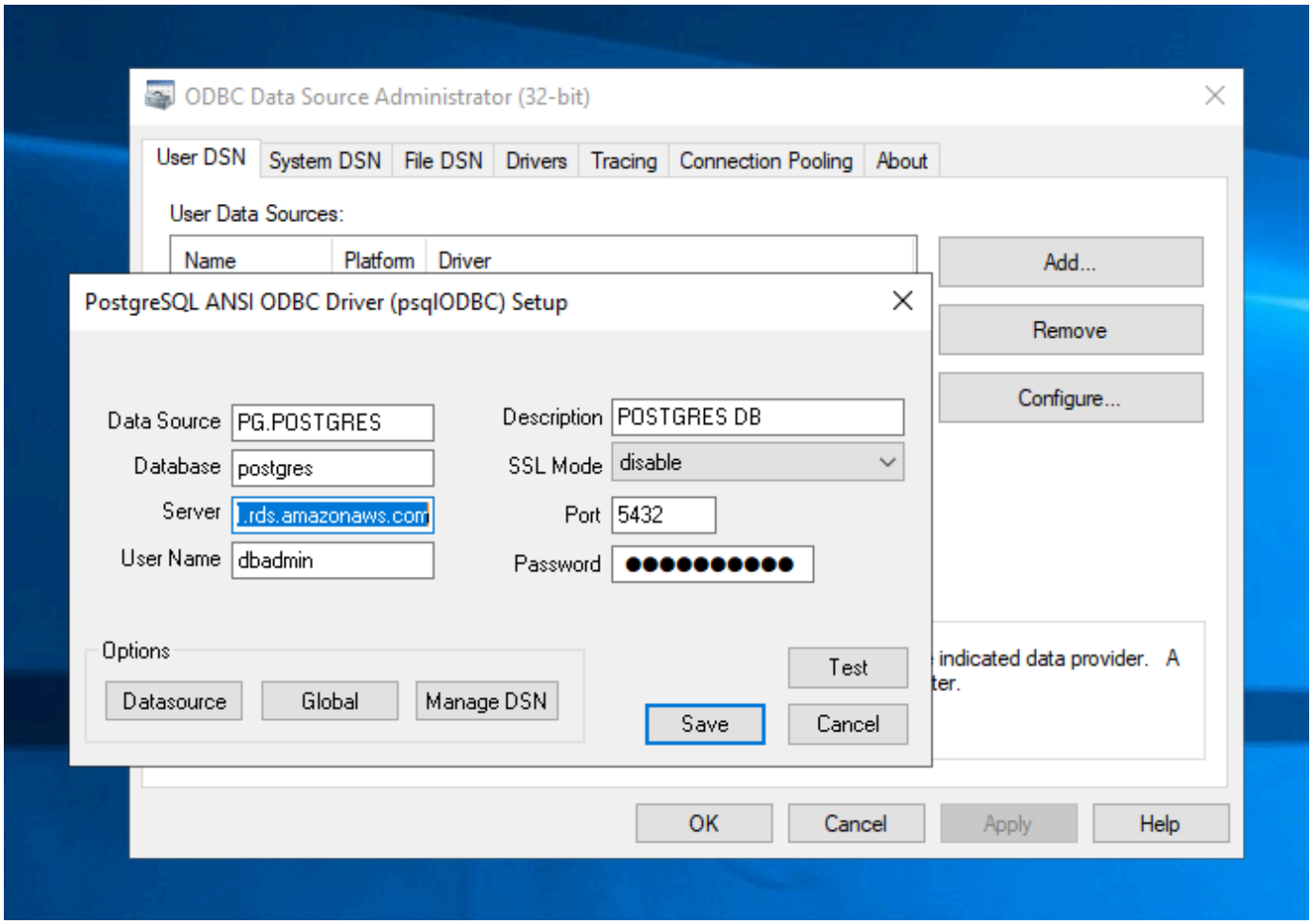
- AppStream 2.0을 통해 Micro Focus Enterprise Developer 데스크톱에 액세스할 수 있어야 합니다.
- Micro Focus 런타임 엔진을 사용하여 AWS Mainframe Modernization에 따라 애플리케이션을 배포하고 실행해야 합니다.
- Aurora PostgreSQL 호환 버전에서 애플리케이션 데이터를 저장하고 있습니다.

1단계: Micro Focus 데이터 스토어(Amazon RDS 데이터베이스)에 대한 ODBC 연결 설정

이 단계에서는 테이블과 열로 보려는 데이터가 들어 있는 데이터베이스에 대한 ODBC 연결을 설정합니다. 이 단계는 일회성 단계입니다.

1. AppStream 2.0 스트리밍 URL을 사용하여 Micro Focus Enterprise Developer 데스크톱에 로그인합니다.
2. ODBC 데이터 원본 관리자를 열고 사용자 DSN을 선택한 다음 추가를 선택합니다.
3. 새 데이터 소스 생성에서 PostgreSQL ANSI를 선택한 다음 마침을 선택합니다.
4. 다음과 같이 필요한 데이터베이스 정보를 제공하여 `PG.POSTGRES`에 대한 데이터 소스를 생성합니다.

```
Data Source : PG.POSTGRES
Database    : postgres
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
Password    : user_password
```



5. 테스트를 선택하여 연결이 제대로 작동하는지 확인합니다. 테스트가 성공하면 `Connection successful` 메시지가 표시됩니다.

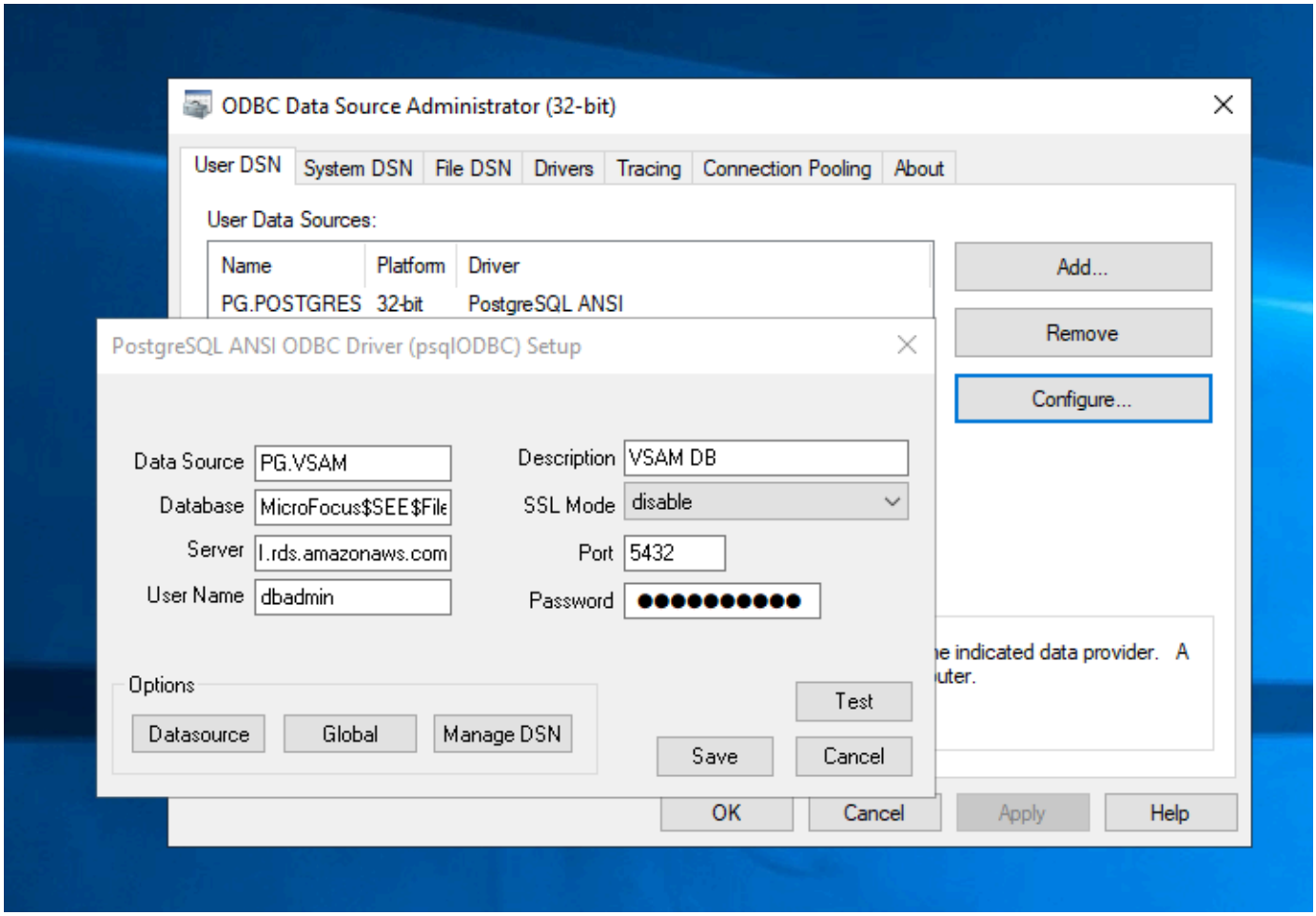
   테스트에 실패하면 다음 정보를 검토하세요.

   - Amazon RDS 문제 해결
   - Amazon RDS DB 인스턴스에 연결할 때 발생하는 문제를 해결하려면 어떻게 해야 합니까?

6. 데이터 소스를 저장합니다.

7. PG.VSAM에 대한 데이터 원본을 만들고 연결을 테스트하고, 데이터 원본을 저장합니다. 데이터베이스에 대한 다음 정보를 제공합니다.

```
Data Source : PG.VSAM
Database    : MicroFocus$SEE$Files$VSAM
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
Password    : user_password
```



2단계: MFDBFH.cfg 파일 생성

이 단계에서는 Micro Focus 데이터 스토어를 설명하는 구성 파일을 생성합니다. 구성 단계는 일회성 단계입니다.

1. 홈 폴더(예: D:\PhotonUser\My Files\Home Folder\MFED\cfg\MFDBFH.cfg)에서 다음 내용이 포함된 MFDBFH.cfg 파일을 생성합니다.

```
<datastores>
       <server name="ESPACDatabase" type="postgresql" access="odbc">
        <dsn name="PG.POSTGRES" type="database" dbname="postgres"/>
        <dsn name="PG.VSAM" type="datastore" dsname="VSAM"/>
       </server>
      </datastores>
```

2. 다음 명령을 실행하여 Micro Focus 데이터 스토어를 쿼리하여 MFDBFH 구성을 확인합니다.

```
*##*
*## Test the connection by running the following commands*
*##*

set MFDBFH_CONFIG="D:\PhotonUser\My Files\Home Folder\MFED\cfg\MFDBFH.cfg"

dbfhdeploy list sql://ESPACDatabase/VSAM?folder=/DATA
```

3단계: 카피북 레이아웃을 위한 구조(STR) 파일 생성

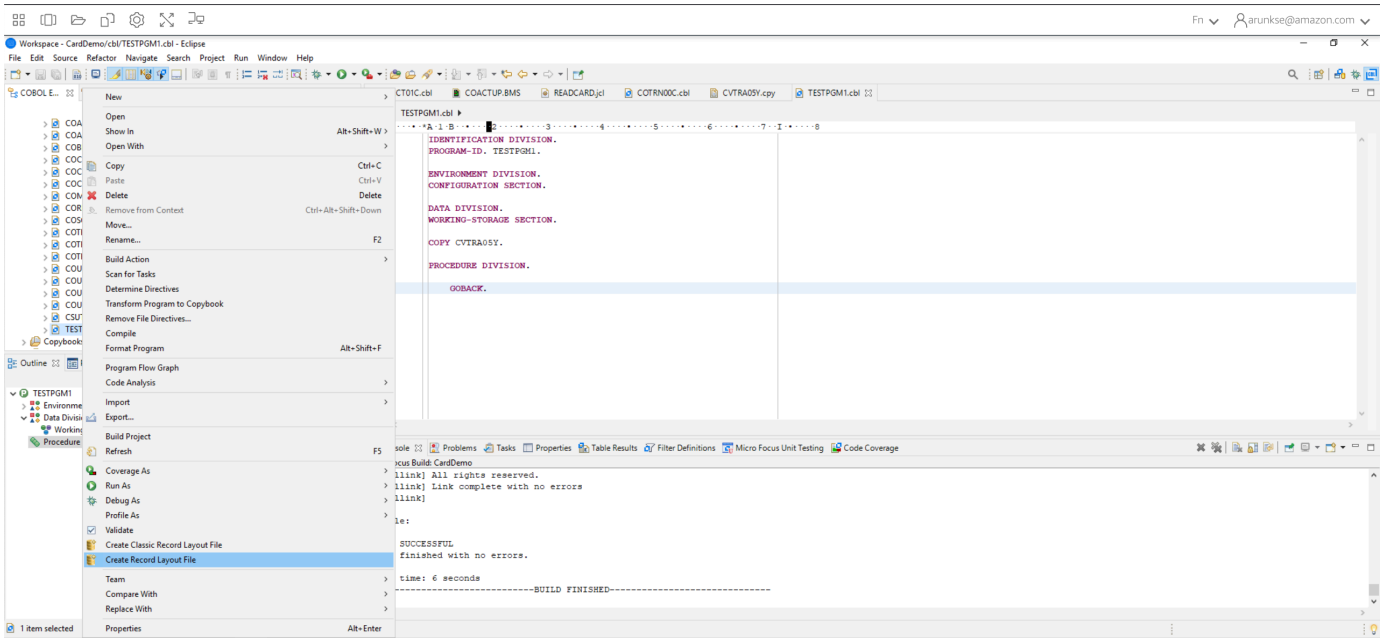
이 단계에서는 나중에 이 파일을 사용하여 데이터 세트에서 데이터베이스 보기를 만들 수 있도록 카피북 레이아웃용 구조 파일을 만듭니다.

1. 카피북과 관련된 프로그램을 컴파일하세요. 카피북을 사용하는 프로그램이 없는 경우 카피북의 COPY 문을 사용하여 다음과 같은 간단한 프로그램을 만들고 컴파일하세요.

```
IDENTIFICATION DIVISION.
      PROGRAM-ID. TESTPGM1.

      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.

      DATA DIVISION.
      WORKING-STORAGE SECTION.

      COPY CVTRA05Y.

      PROCEDURE DIVISION.
```

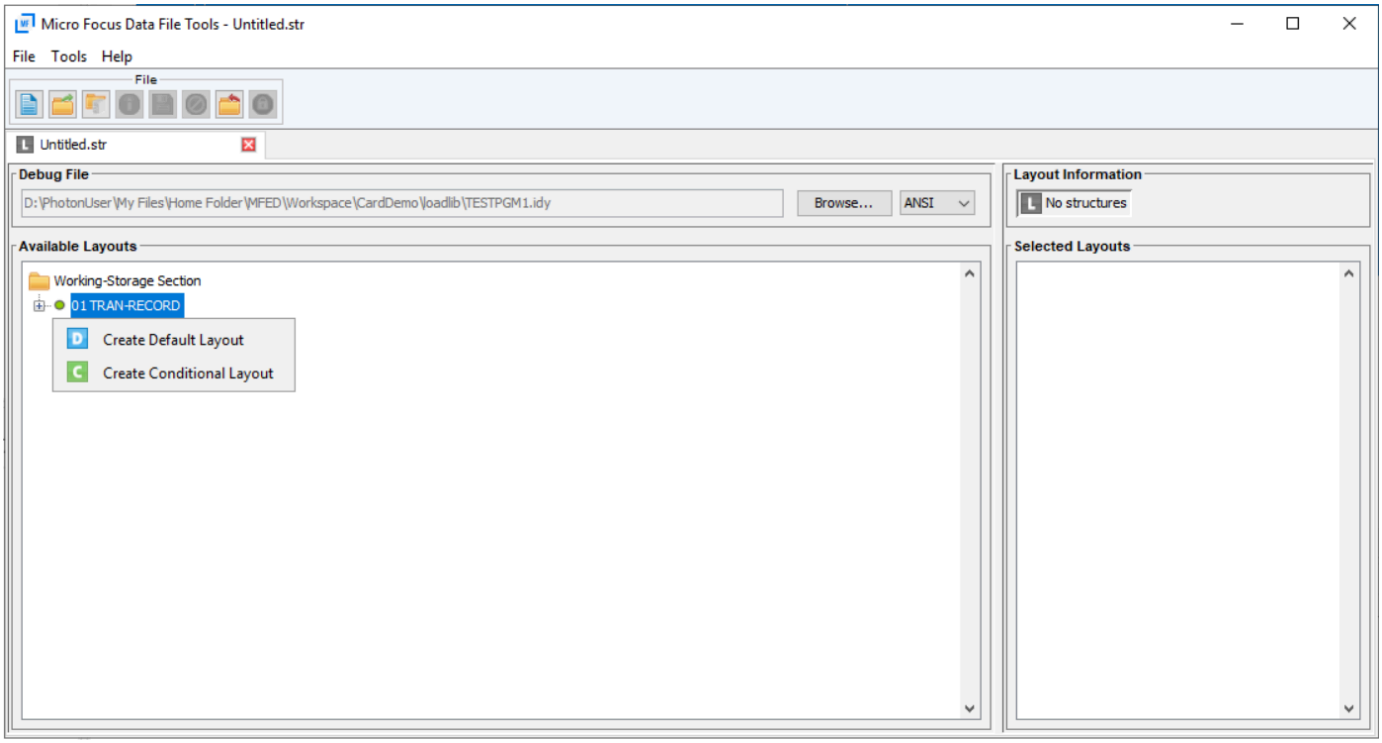
```
       GOBACK.
```

2. 컴파일에 성공하면 프로그램을 마우스 오른쪽 버튼으로 클릭하고 레코드 레이아웃 파일 생성을 선택합니다. 그러면 컴파일 중에 생성된.idy 파일을 사용하여 Micro Focus 데이터 파일 도구가 열립니다.

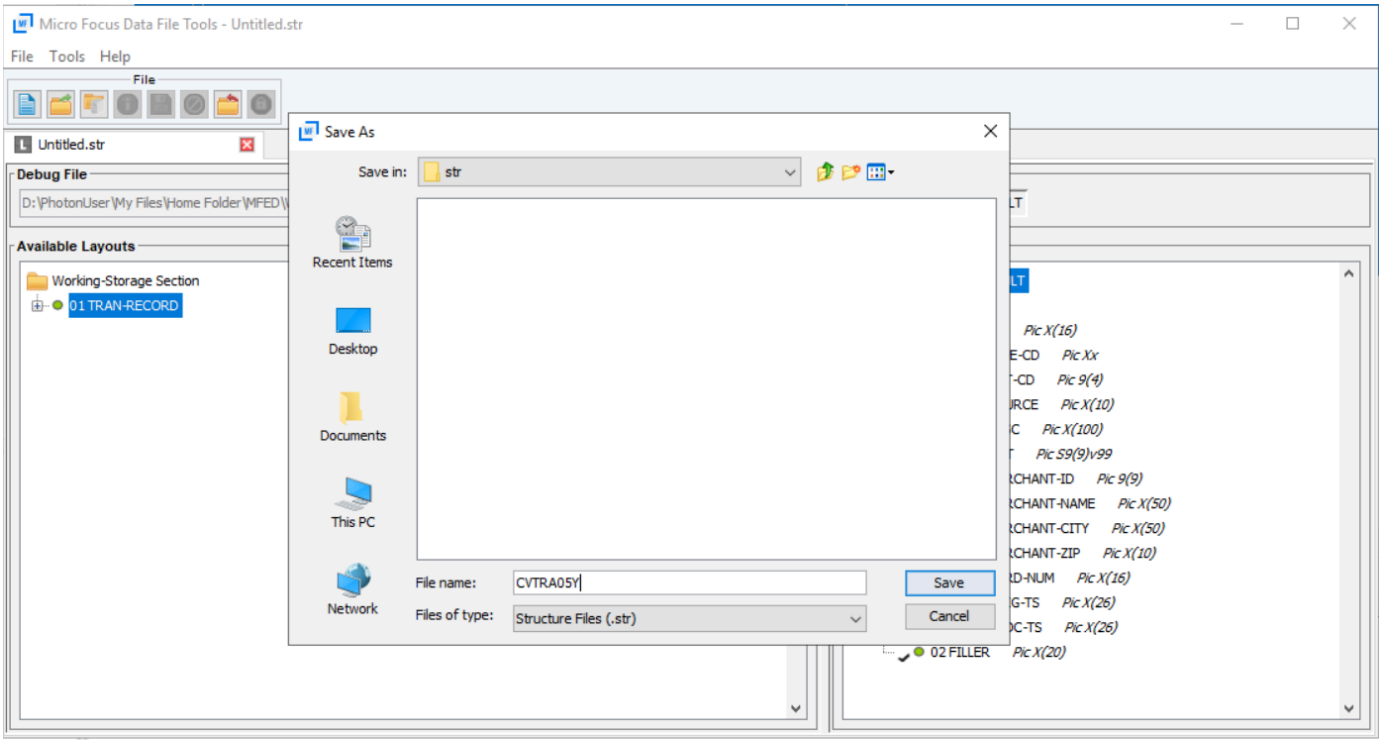


3. 레코드 구조를 마우스 오른쪽 버튼으로 클릭하고 레이아웃에 따라 기본 레이아웃 생성(단일 구조) 또는 조건부 레이아웃 생성(다중 구조)을 선택합니다.

   자세한 내용은 Micro Focus 설명서의 구조 파일 및 레이아웃 생성을 참조하세요.



4. 레이아웃을 생성한 후 메뉴에서 파일을 선택한 다음 다른 이름으로 저장을 선택합니다. 홈 폴더에서 파일을 찾아 카피북과 동일한 파일 이름으로 저장합니다. `str`라는 폴더를 만들고 여기에 모든 구조 파일을 저장할 수 있습니다.



4단계: 구조(STR) 파일을 사용하여 데이터베이스 뷰 생성

이 단계에서는 이전에 만든 구조 파일을 사용하여 데이터 세트에 대한 데이터베이스 보기를 만듭니다.

- dbfhview 명령을 사용하여 다음 예와 같이 Micro Focus 데이터 스토어에 이미 있는 데이터 세트에 대한 데이터베이스 보기를 생성합니다.

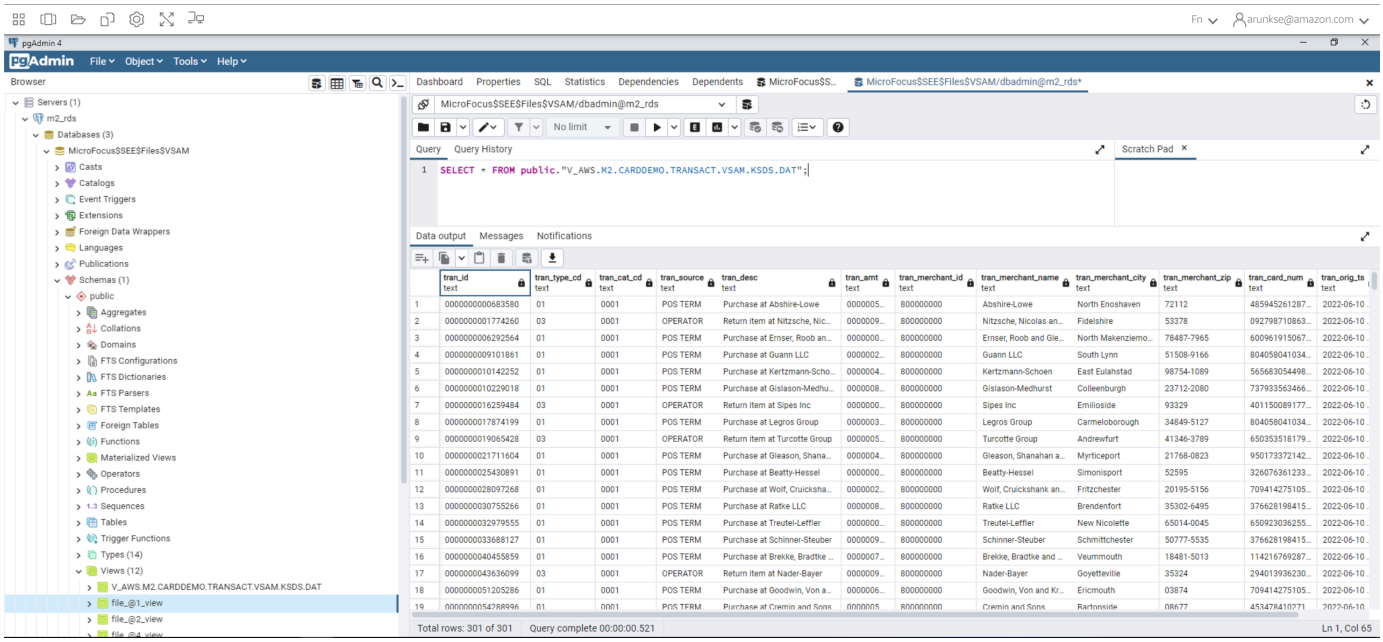
```
##
      ## The below command creates database view for VSAM file
 AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS
      ## using the STR file CVTRA05Y.str
      ##

      dbfhview -create -struct:"D:\PhotonUser\My Files\Home Folder\MFED\str
\CVTRA05Y.str" -name:V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT -file:sql://
ESPACDatabase/VSAM/AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT?folder=/DATA

      ##
      ## Output:
      ##

      Micro Focus Database File Handler - View Generation Tool Version 8.0.00
      Copyright (C) 1984-2022 Micro Focus. All rights reserved.

      VGN0017I Using structure definition 'TRAN-RECORD-DEFAULT'
      VGN0022I View 'V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT' installed in
 datastore 'sql://espacdatabase/VSAM'
      VGN0002I The operation completed successfully
```

5단계: Micro Focus 데이터 세트를 테이블 및 열로 보기

이 단계에서는 쿼리를 실행하여 테이블 및 열과 같은 데이터 세트를 볼 수 있도록 pgAdmin를 사용하여 데이터베이스에 연결합니다.

- pgAdmin을 사용하여 데이터베이스 MicroFocusSEEFiles$VSAM에 연결하고 4단계에서 만든 데이터베이스 보기를 쿼리합니다.

```
SELECT * FROM public."V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT";
```



튜토리얼: Micro Focus Enterprise Developer에서의 템플릿 사용

이 자습서에서는 Micro Focus Enterprise Developer를 사용하여 템플릿과 사전 정의된 프로젝트를 사용하는 방법을 설명합니다. 세 가지 사용 사례를 다룹니다. 모든 사용 사례는 BankDemo 샘플에 제공된 샘플 코드를 사용합니다. 샘플을 다운로드하려면 `bankdemo.zip`를 선택합니다.

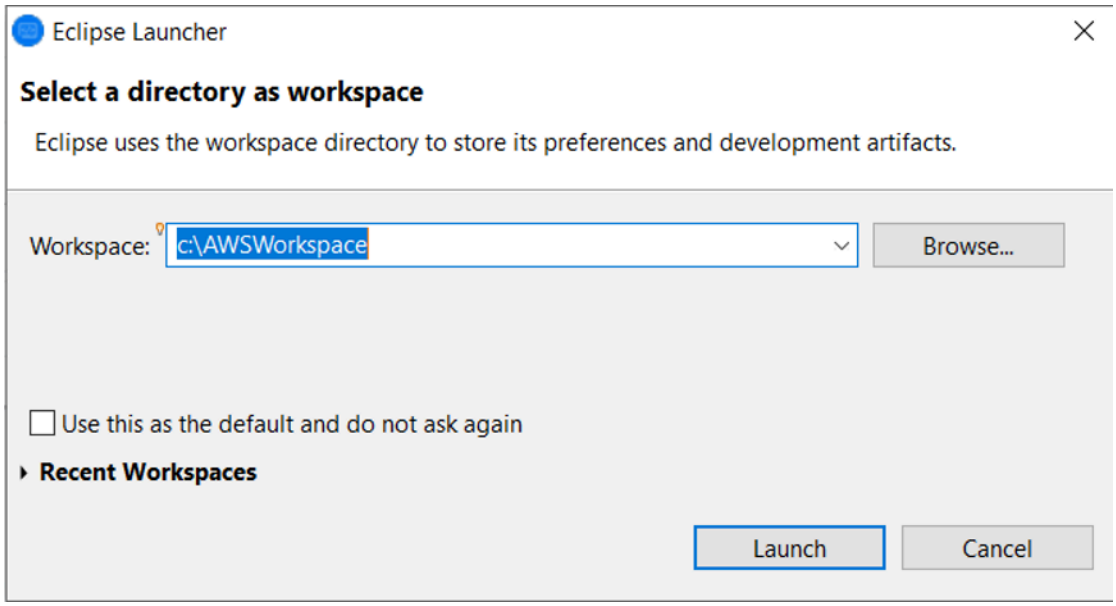
> **Important**
>
> Windows용 Enterprise Developer 버전을 사용하는 경우 컴파일러에서 생성한 바이너리는 Enterprise Developer와 함께 제공된 엔터프라이즈 서버에서만 실행할 수 있습니다. Linux 기반인 AWS Mainframe Modernization 런타임에서는 실행할 수 없습니다.

주제

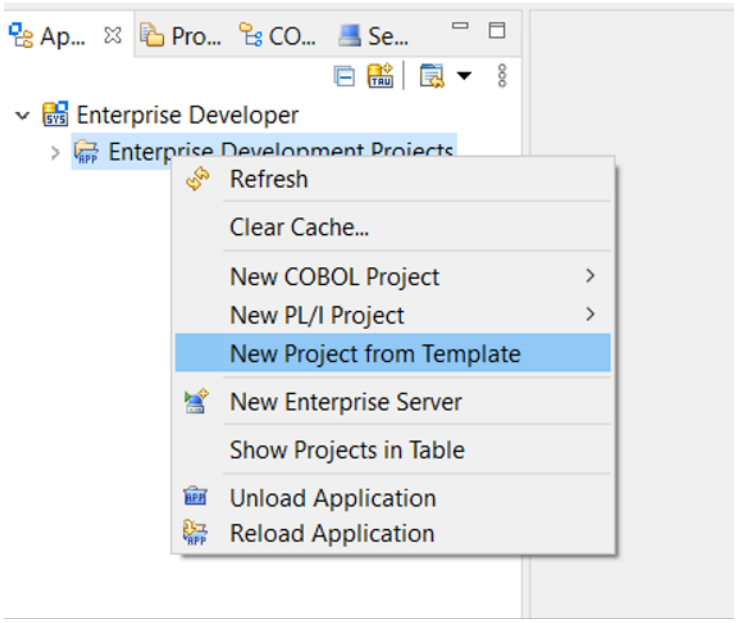
사용 사례 1 - 소스 구성 요소가 포함된 COBOL 프로젝트 템플릿 사용

이 사용 사례에서는 데모 사전 설정 단계의 일부로 소스 구성 요소를 템플릿 디렉토리 구조에 복사해야 합니다. bankdemo.zip에서 소스 사본이 두 개 생기지 않도록 원래 AWSTemplates.zip 전달과 다르게 변경되었습니다.

1. Enterprise Developer를 시작하고 선택한 작업 공간을 지정합니다.

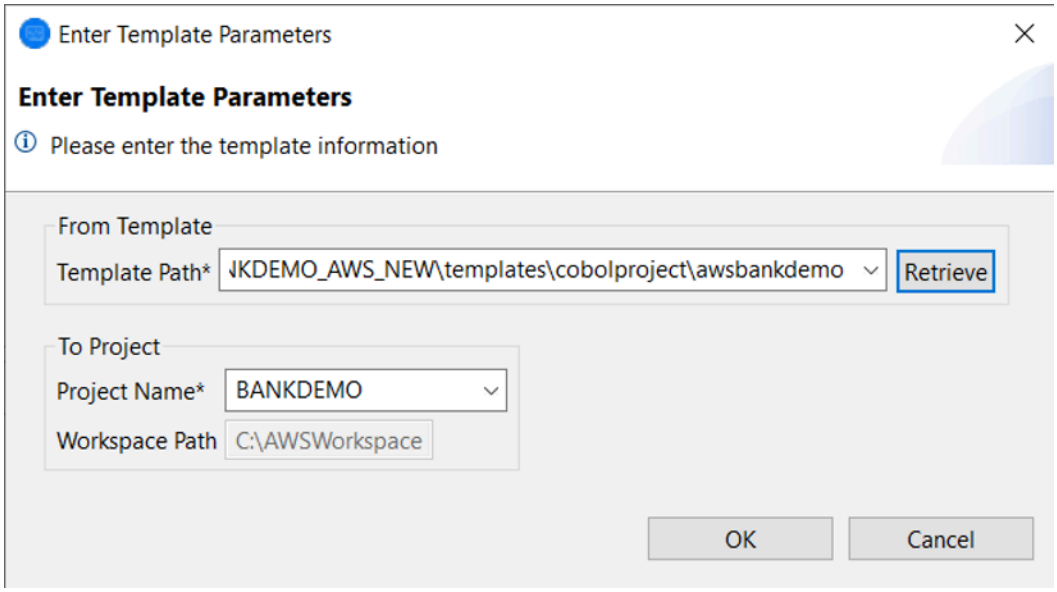


2. 애플리케이션 탐색기 보기의 Enterprise Development 프로젝트 트리 보기 항목에 있는 컨텍스트 메뉴의 템플릿에서 새 프로젝트를 선택합니다.

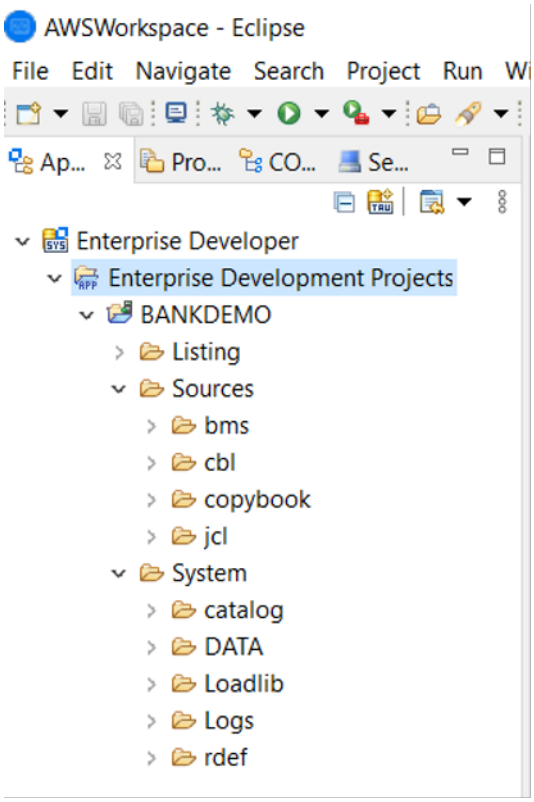


3. 표시된 대로 템플릿 매개 변수를 입력합니다.

> **Note**
>
> 템플릿 경로는 ZIP이 추출된 위치를 참조합니다.



4. 확인을 선택하면 제공된 템플릿을 기반으로 완전한 소스 및 실행 환경 구조를 갖춘 로컬 개발 Eclipse 프로젝트가 생성됩니다.



이 System 구조에는 BANKDEMO에 필요한 항목이 포함된 전체 리소스 정의 파일, 항목이 추가된 필수 카탈로그 및 해당 ASCII 데이터 파일이 포함되어 있습니다.

소스 템플릿 구조에는 모든 소스 항목이 포함되므로 이러한 파일은 로컬 프로젝트에 복사되므로 Enterprise Developer에서 자동으로 빌드됩니다.

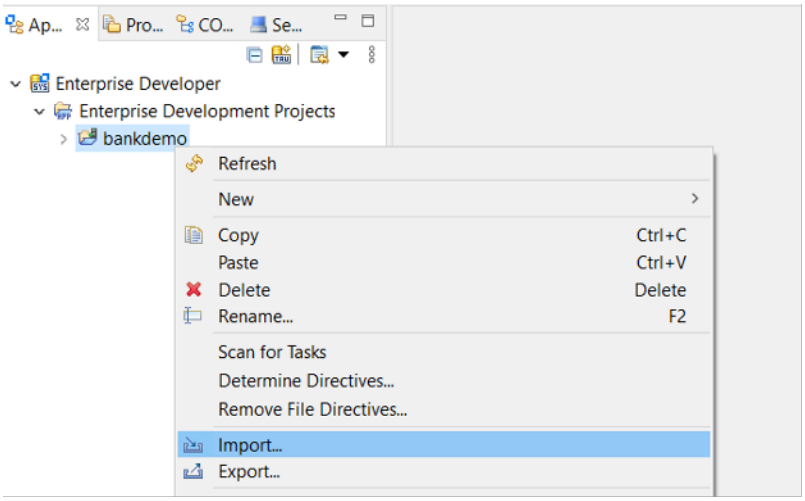
사용 사례 2 - 소스 구성 요소 없이 COBOL 프로젝트 템플릿 사용

1~3단계는 사용 사례 1 - 소스 구성 요소가 포함된 COBOL 프로젝트 템플릿 사용과 동일합니다.

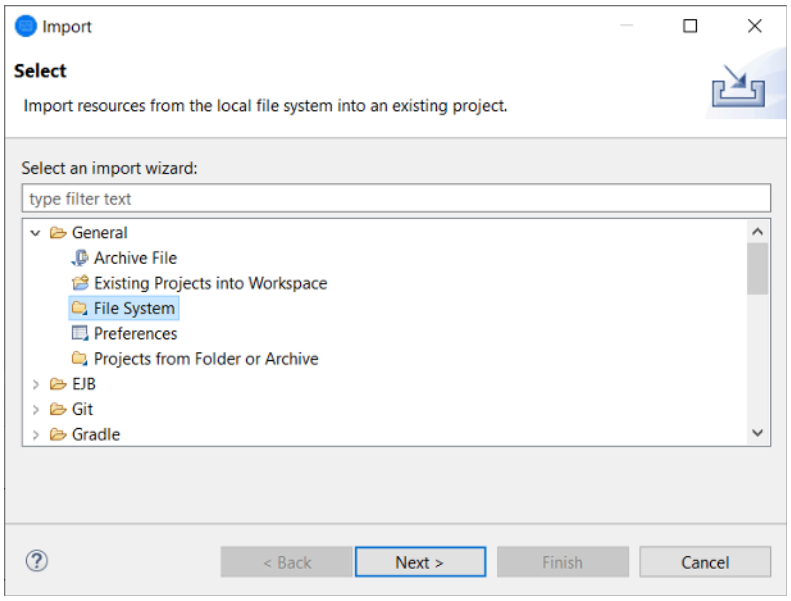
이 사용 사례의 System 구조에는 BankDemo의 필수 항목이 포함된 전체 리소스 정의 파일, 항목이 추가된 필수 카탈로그 및 해당 ASCII 데이터 파일도 포함됩니다.

하지만 템플릿 소스 구조에는 구성 요소가 포함되어 있지 않습니다. 사용 중인 소스 리포지토리에서 프로젝트로 이러한 항목을 가져와야 합니다.

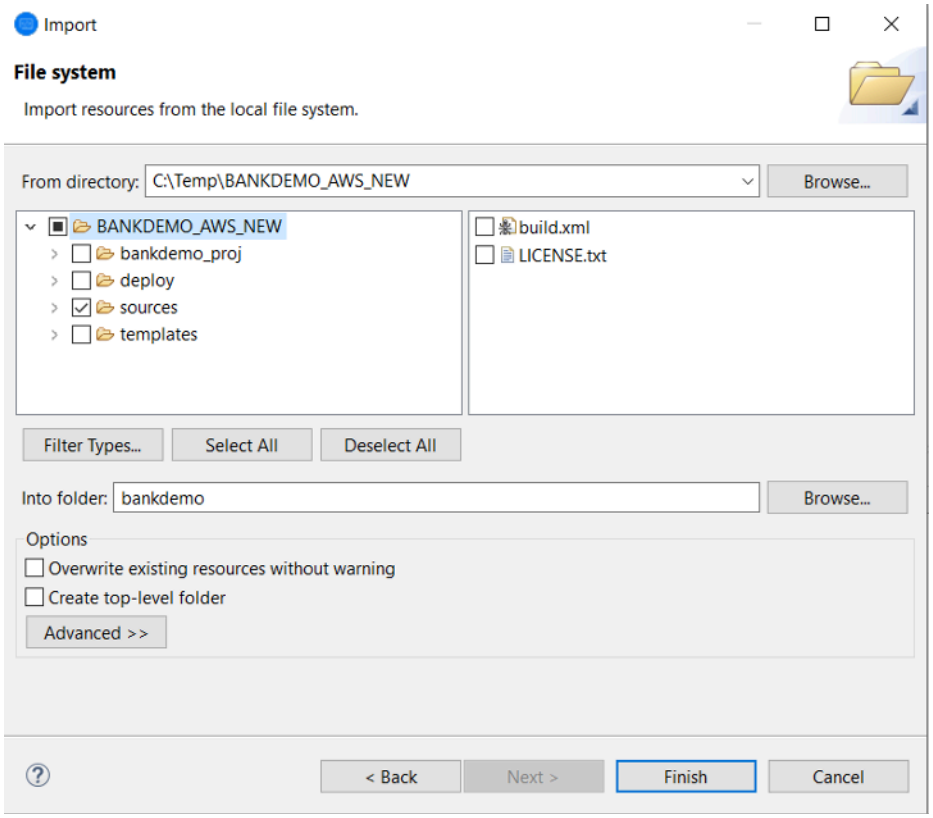
1. 프로젝트 이름을 선택합니다. 관련 컨텍스트 메뉴에서 가져오기를 선택합니다.



2. 표시되는 대화 상자의 일반 섹션에서 파일 시스템을 선택한 후 다음을 선택합니다.



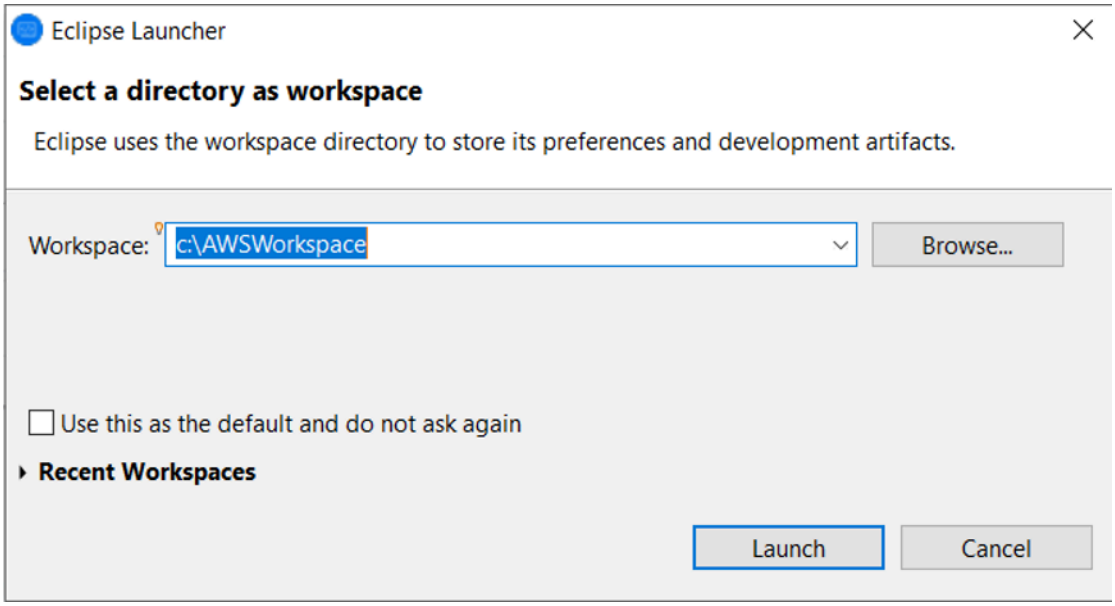
3. 저장소 폴더를 가리키도록 파일 시스템을 검색하여 디렉터리에서 필드를 채웁니다. 가져오려는 폴더(예: `sources`)를 모두 선택합니다. `Into folder` 필드가 미리 채워져 있습니다. 마침을 클릭합니다.



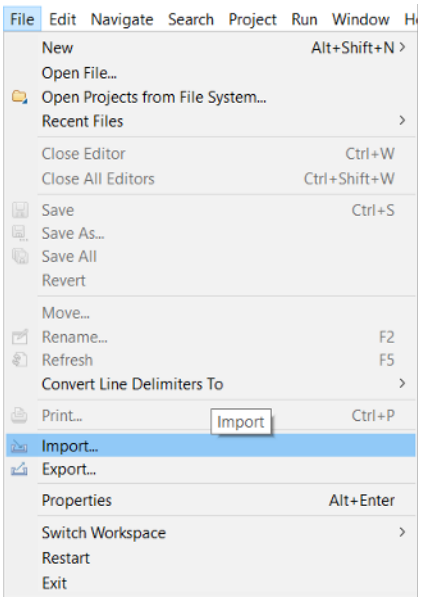
소스 템플릿 구조에 모든 소스 항목이 포함되면 Enterprise Developer에서 해당 항목이 자동으로 빌드됩니다.

사용 사례 3 - 사전 정의된 COBOL 프로젝트를 사용하여 소스 폴더에 연결

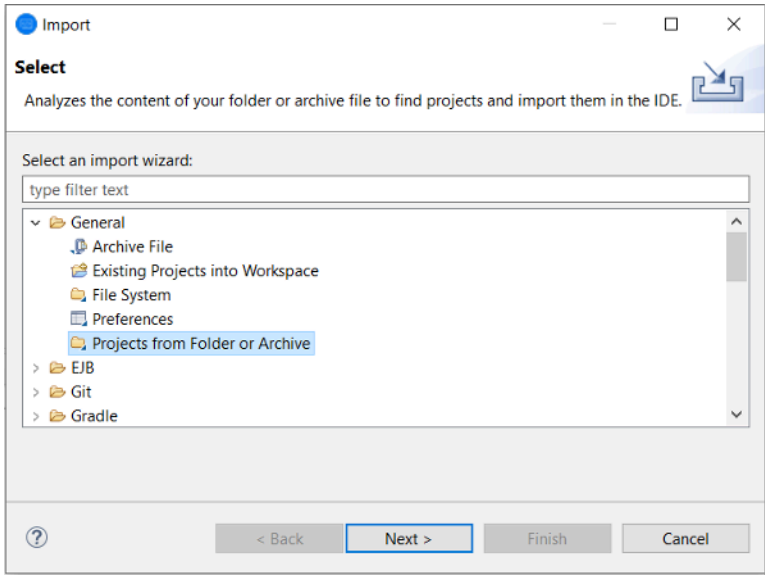
1. Enterprise Developer를 시작하고 선택한 작업 공간을 지정합니다.



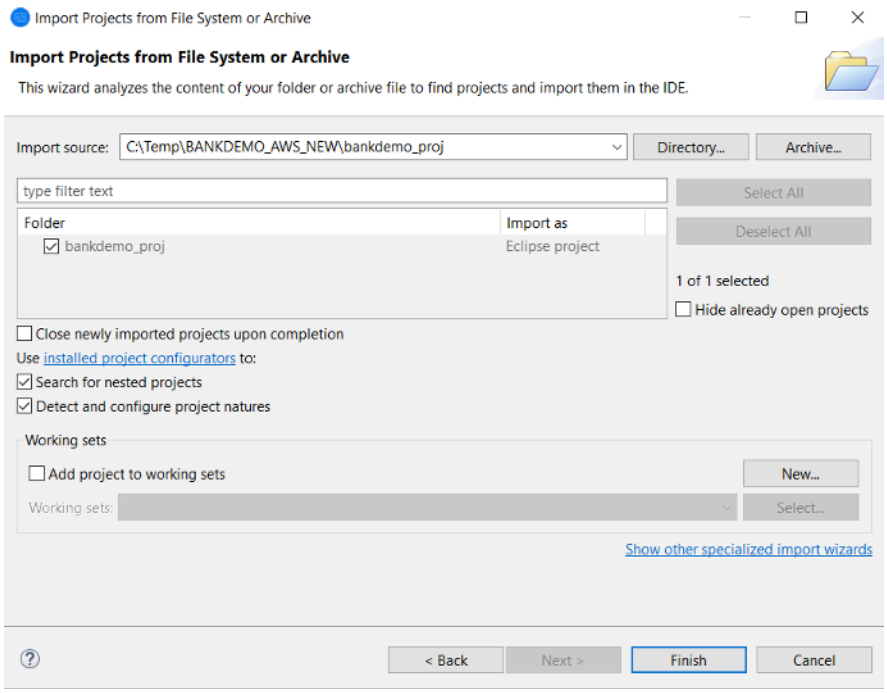
2. 파일 메뉴에서 가져오기를 선택합니다.



3. 표시되는 대화 상자의 일반에서 폴더 또는 아카이브의 프로젝트를 선택하고 다음을 선택합니다.



4. 가져오기 소스를 채우고 디렉터리를 선택한 다음 파일 시스템을 탐색하여 사전 정의된 프로젝트 폴더를 선택합니다. 안에 포함된 프로젝트에는 동일한 저장소의 소스 폴더로 연결되는 링크가 있습니다.

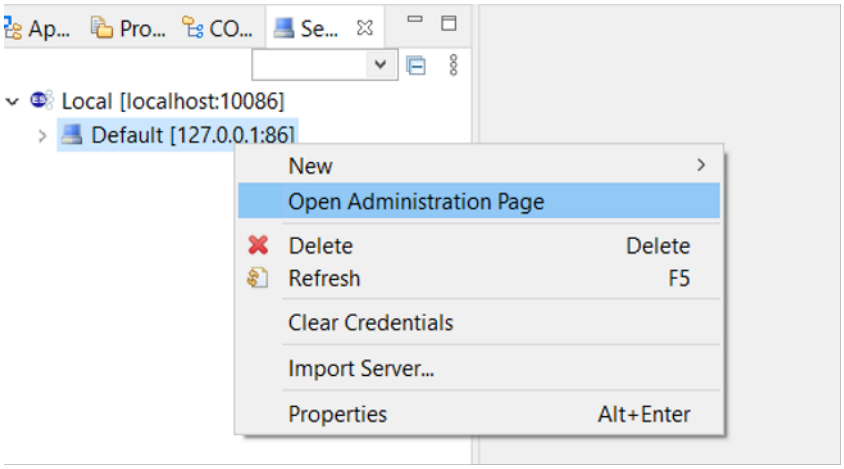


마침을 클릭합니다.

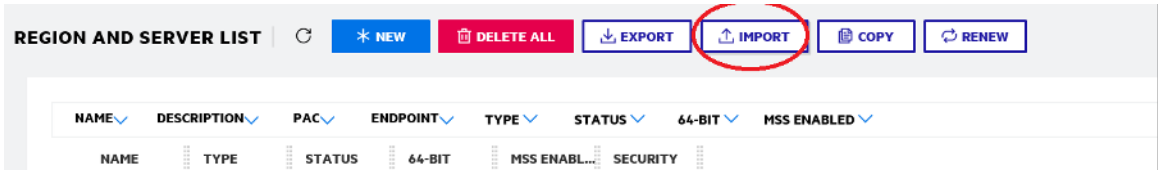
프로젝트는 소스 폴더 링크로 채워지므로 코드가 자동으로 빌드됩니다.

리전 정의 JSON 템플릿 사용

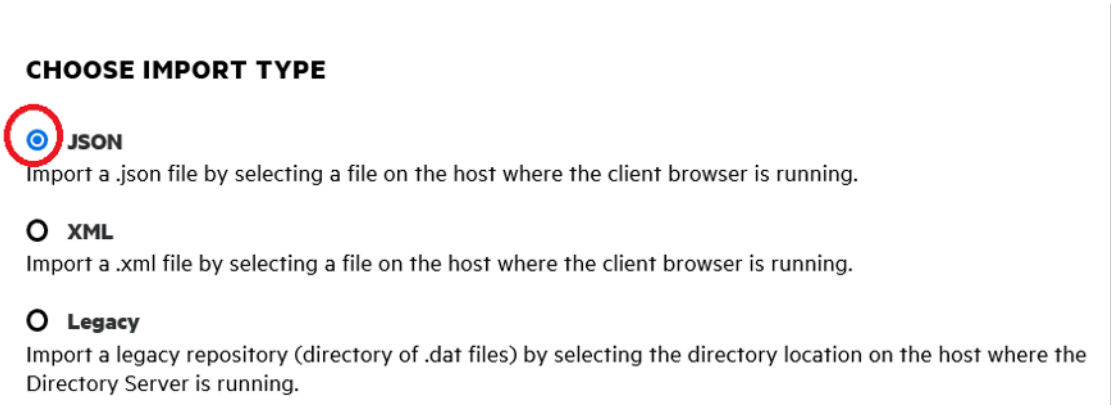
1. 서버 탐색기 보기로 전환합니다. 관련 컨텍스트 메뉴에서 관리 페이지 열기를 선택합니다. 그러면 기본 브라우저가 시작됩니다.



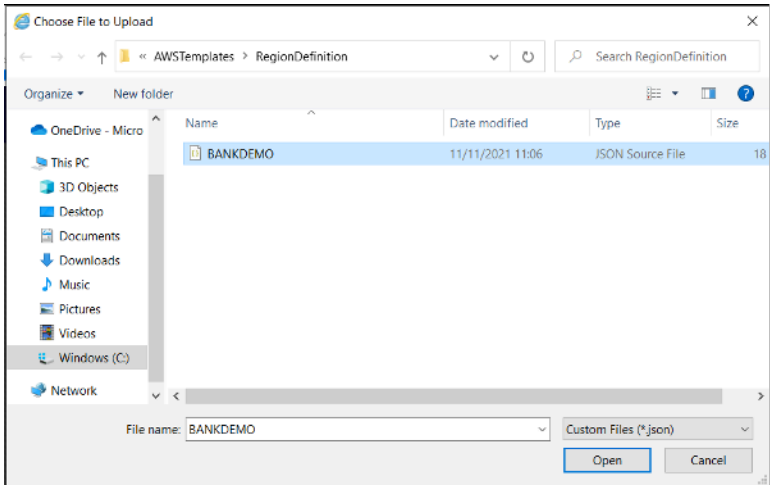
2. 표시되는 엔터프라이즈 서버 공용 웹 관리(ESCWA) 화면에서 가져오기를 선택합니다.



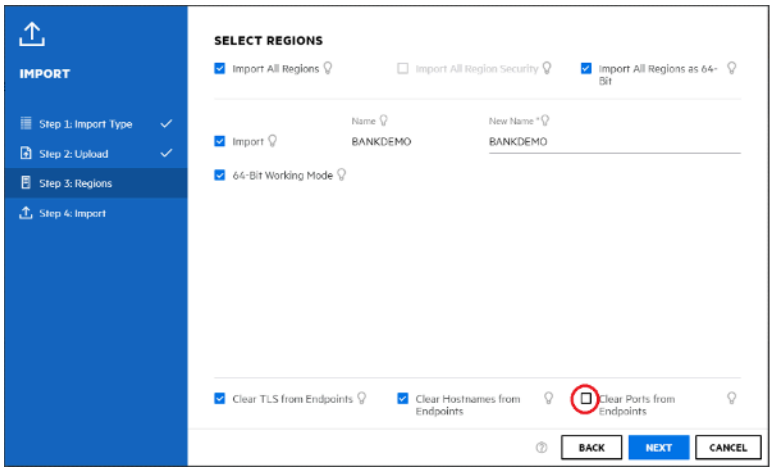
3. JSON 가져오기 유형을 선택하고 다음을 선택합니다.



4. 제공된 `BANKDEMO.JSON` 파일을 업로드합니다.

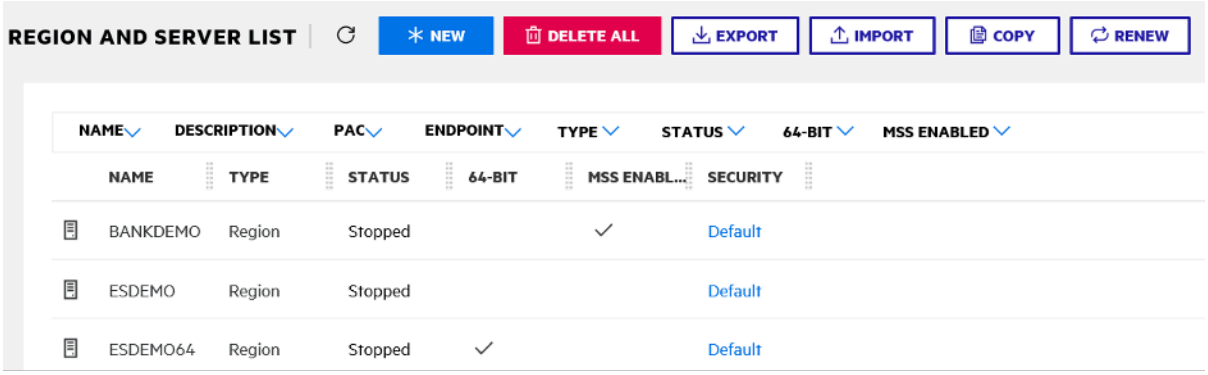


선택한 후 다음을 선택합니다.

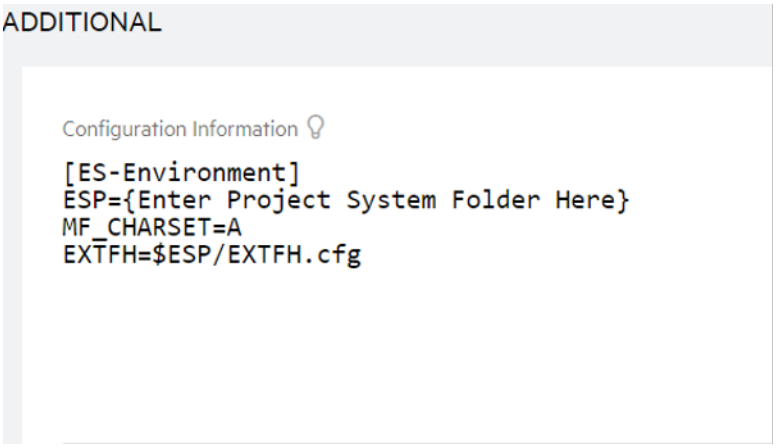


리전 선택 패널에서 엔드포인트에서 포트 지우기 옵션이 선택되지 않았는지 확인한 다음 가져오기 수행 패널이 표시될 때까지 패널을 통해 다음을 계속 선택합니다. 그런 다음 가져오기를 선택합니다.

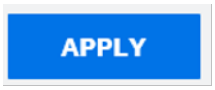
마지막으로 마침을 클릭합니다. 그러면 BANKDEMO 리전이 서버 목록에 추가됩니다.



5. BANKDEMO 리전의 일반 속성으로 이동하세요.
6. 구성 섹션으로 스크롤합니다.
7. ESP 환경 변수는 이전 단계에서 만든 Eclipse 프로젝트와 관련된 `System` 폴더로 설정해야 합니다. 이것은 `workspacefolder/projectname/System`이어야 합니다.

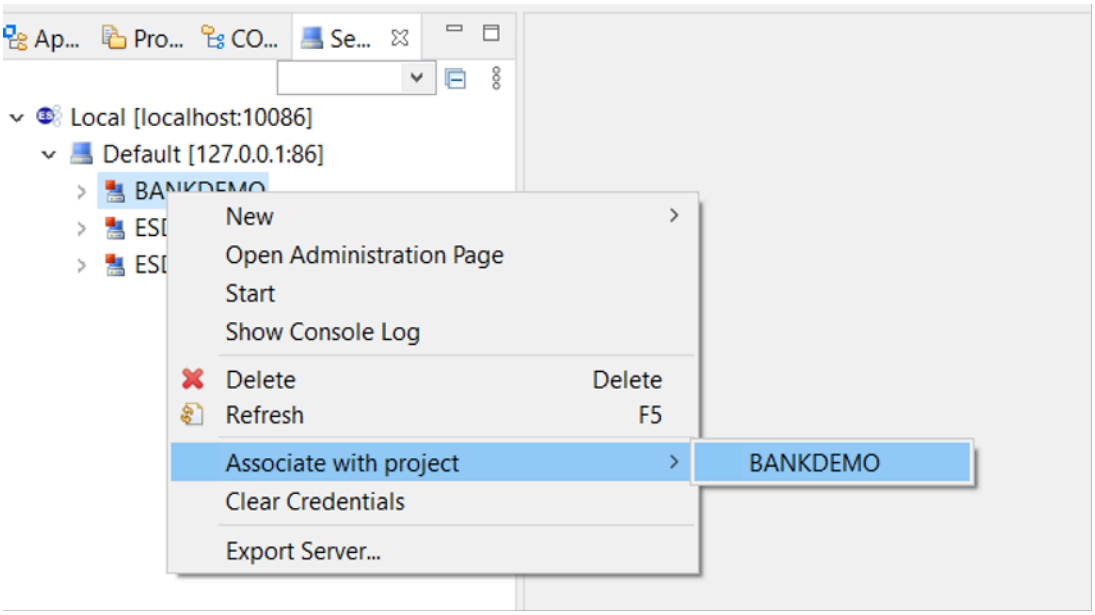


8. 적용를 클릭합니다.



이제 이 리전은 Eclipse COBOL 프로젝트와 함께 실행되도록 완전히 구성되었습니다.

9. 마지막으로 Enterprise Developer로 돌아가서 가져온 리전을 프로젝트에 연결합니다.



이제 완전히 작동하는 BankDemo 버전과 함께 Enterprise Developer 환경을 사용할 준비가 되었습니다. 해당 리전을 대상으로 코드를 편집, 컴파일 및 디버그할 수 있습니다.

> **Important**
>
> Windows용 Enterprise Developer 버전을 사용하는 경우 컴파일러에서 생성된 바이너리는 Enterprise Developer와 함께 제공된 엔터프라이즈 서버에서만 실행할 수 있습니다. Linux 기반인 AWS Mainframe Modernization 런타임에서는 실행할 수 없습니다.

자습서: BankDemo 샘플 애플리케이션을 위한 Micro Focus 빌드 설정

AWS Mainframe Modernization은 마이그레이션된 애플리케이션을 위한 빌드와 지속적 통합/지속적 전달(CI/CD) 파이프라인을 설정할 수 있는 기능을 제공합니다. 이러한 빌드와 파이프라인은 AWS CodeBuild, AWS CodeCommit 및 AWS CodePipeline를 사용하여 이러한 기능을 제공합니다. CodeBuild는 소스 코드를 컴파일하고 단위 테스트를 실행하며 배포할 준비가 완료된 아티팩트를 생성하는 완전 관리형 빌드 서비스입니다. CodeCommit은 AWS 클라우드에 Git 리포지토리를 비공개로 저장하고 관리할 수 있는 버전 제어 서비스입니다. CodePipeline은 소프트웨어 릴리스에 필요한 단계를 모델링, 시각화 및 자동화하는 데 사용할 수 있는 지속적 전달 서비스입니다.

이 자습서에서는 AWS CodeBuild을 사용하여 Amazon S3에서 BankDemo 샘플 애플리케이션 소스 코드를 컴파일한 다음 컴파일된 코드를 Amazon S3로 다시 내보내는 방법을 보여줍니다.

AWS CodeBuild은 소스 코드를 컴파일하고 테스트를 실행하며 배포 준비가 완료된 소프트웨어 패키지를 생성하는 종합 관리형 지속 통합 서비스입니다. CodeBuild를 사용하면 사전 패키징된 빌드 환경을 사용하거나 혹은 자체 빌드 도구를 사용하는 사용자 지정 빌드 환경을 만들 수 있습니다. 이 데모 시나리오에서는 두 번째 옵션을 사용합니다. 사전 패키징된 Docker 이미지를 사용하는 CodeBuild 빌드 환경으로 구성되어 있습니다.

> **Important**
>
> Mainframe Modernization 프로젝트를 시작하기 전에 Mainframe용 AWS Migration Acceleration Program(MAP)에 대해 알아보거나 AWS Mainframe 전문가에게 문의하여 Mainframe 애플리케이션을 현대화하는 데 필요한 단계에 대해 알아보는 것이 좋습니다.

주제

사전 조건

이 자습서를 시작하기 전에 다음 사전 조건을 완료합니다.

- BankDemo 샘플 애플리케이션을 다운로드하고 폴더에 압축을 풉니다. 소스 폴더에는 COBOL 프로그램과 카피북, CICS BMS 정의가 들어 있습니다. JCL을 빌드할 필요는 없지만 참조용 JCL 폴더도 포함되어 있습니다. 이 폴더에는 빌드에 필요한 메타 파일도 들어 있습니다.
- AWS Mainframe Modernization 콘솔에서 도구를 선택합니다. 자산 분석, 개발 및 구축에서 내 AWS 계정과 자산 공유를 선택합니다.

1단계: Amazon S3 버킷 생성

이 단계에서는 두 개의 Amazon S3 버킷을 생성합니다. 첫 번째는 소스 코드를 보관하는 입력 버킷이고 다른 하나는 빌드 출력을 보관하는 출력 버킷입니다. 자세한 내용은 Amazon S3 사용 설명서의 Amazon S3 버킷 생성, 구성 및 사용을 참조하세요.

1. 입력 버킷을 만들려면 Amazon S3 콘솔에 로그인하고 버킷 생성을 선택합니다.
2. 일반 구성에서 버킷의 이름을 제공하고 버킷을 만들려는 위치인 AWS 리전을 지정합니다. 예제 이름은 `regionId`가 버킷의 AWS 리전이고 `accountId`가 AWS 계정 ID인 `codebuild-regionId-accountId-input-bucket`입니다.

   > **Note**
   >
   > 미국 동부(버지니아 북부)와 다른 AWS 리전에서 버킷을 생성하는 경우 `LocationConstraint` 매개 변수를 지정하세요. 자세한 내용은 Amazon Simple Storage Service API 참조의 버킷 생성을 참조하세요.

3. 다른 모든 설정을 유지하고 버킷 생성을 선택합니다.
4. 1-3단계를 반복하여 출력 버킷을 생성합니다. 예제 이름은 `regionId`가 버킷의 AWS 리전이고 `accountId`가 AWS 계정 ID인 `codebuild-regionId-accountId-output-bucket`입니다.

   이러한 버킷에 대해 어떤 이름을 선택하더라도 선택한 경우 이 자습서 전체에서 해당 이름을 사용해야 합니다.

2단계: 빌드 사양 생성

이 단계에서는 빌드 사양 파일을 생성합니다. 이 파일은 CodeBuild에서 빌드를 실행하기 위한 빌드 명령과 관련 설정을 YAML 형식으로 제공합니다. 자세한 내용은 AWS CodeBuild 사용 설명서의 CodeBuild용 빌드 사양 참조 필드를 참조하세요.

1. 사전 요구 사항으로 압축을 푼 `buildspec.yml` 디렉터리에 이름이 지정된 파일을 생성합니다.
2. 다음 콘텐츠를 파일에 추가하고 저장합니다. 이 파일은 변경할 필요가 없습니다.

```
version: 0.2
env:
  exported-variables:
    - CODEBUILD_BUILD_ID
    - CODEBUILD_BUILD_ARN
```

```
phases:
  install:
    runtime-versions:
      python: 3.7
  pre_build:
    commands:
      - echo Installing source dependencies...
      - ls -lR $CODEBUILD_SRC_DIR/source
  build:
    commands:
      - echo Build started on `date`
      - /start-build.sh -Dbasedir=$CODEBUILD_SRC_DIR/source -Dloaddir=
$CODEBUILD_SRC_DIR/target
  post_build:
    commands:
      - ls -lR $CODEBUILD_SRC_DIR/target
      - echo Build completed on `date`
artifacts:
  files:
    - $CODEBUILD_SRC_DIR/target/**
```

여기서 CODEBUILD_BUILD_ID, CODEBUILD_BUILD_ARN, $CODEBUILD_SRC_DIR/source 및 $CODEBUILD_SRC_DIR/target은 CodeBuild 내에서 사용할 수 있는 환경 변수입니다. 자세한 내용은 빌드 환경의 환경 변수 참조하세요.

이때 다음과 같이 디렉터리가 나타나야 합니다.

```
(root directory name)
    |-- build.xml
    |-- buildspec.yml
    |-- LICENSE.txt
    |-- source
         |... etc.
```

3. 폴더의 내용을 BankDemo.zip이라는 이름의 파일로 압축합니다. 이 자습서에서는 폴더를 압축할 수 없습니다. 대신 폴더의 내용 압축을 통해 BankDemo.zip 파일에 압축합니다.

3단계: 소스 파일 업로드

이 단계에서는 BankDemo 샘플 애플리케이션의 소스 코드를 Amazon S3 입력 버킷에 업로드합니다.

1. Amazon SNS 콘솔에 로그인하고 왼쪽 탐색 창에서 버킷을 선택합니다. 그런 다음 이전에 생성한 입력 버킷을 선택합니다.
2. 객체에서 업로드를 선택합니다.
3. 파일 및 폴더 섹션에서 파일 추가를 선택합니다.
4. `BankDemo.zip` 파일로 이동하여 선택합니다.
5. 업로드를 선택합니다.

4단계: IAM 정책 생성

이 단계에서는 두 개의 IAM 정책을 생성합니다. 하나의 정책은 AWS Mainframe Modernization에 Micro Focus 빌드 도구가 포함된 도커 이미지에 액세스하고 사용할 수 있는 권한을 부여합니다. 이 정책은 고객을 위해 맞춤화되지 않았습니다. 다른 정책에서는 AWS Mainframe Modernization이 입력 및 출력 버킷 및 CodeBuild에서 생성하는 Amazon CloudWatch Logs와 상호 작용할 수 있는 권한을 부여합니다.

IAM 정책 생성에 대해 자세히 알아보려면 IAM 사용자 설명서의 IAM 정책 편집을 참조하세요.

Docker 이미지에 액세스하기 위한 정책을 만들려면

1. IAM 콘솔에서 다음 정책 문서를 복사하여 정책 편집기에 붙여 넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:*:673918848628:repository/m2-enterprise-build-
tools"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::aws-m2-repo-*/*"
        }
    ]
}
```

2. 정책의 이름을 제공합니다(예: `m2CodeBuildPolicy`).

AWS Mainframe Modernization가 버킷 및 로그와 상호 작용할 수 있도록 허용하는 정책을 만들려면

1. IAM 콘솔에서 다음 정책 문서를 복사하여 정책 편집기에 붙여 넣습니다. `regionId`를 AWS 리전으로, `accountId`를 자신의 AWS 계정로 업데이트해야 합니다.

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
              "Action": [
                  "logs:CreateLogGroup",
                  "logs:CreateLogStream",
                  "logs:PutLogEvents"
              ],
              "Resource": [
                  "arn:aws:logs:regionId:accountId:log-group:/aws/codebuild/
codebuild-bankdemo-project",
                  "arn:aws:logs:regionId:accountId:log-group:/aws/codebuild/
codebuild-bankdemo-project:*"
              ],
              "Effect": "Allow"
          },
          {
              "Action": [
                  "s3:PutObject",
                  "s3:GetObject",
                  "s3:GetObjectVersion",
                  "s3:GetBucketAcl",
                  "s3:GetBucketLocation",
                  "s3:List*"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::codebuild-regionId-accountId-input-bucket",
                "arn:aws:s3:::codebuild-regionId-accountId-input-bucket/*",
                "arn:aws:s3:::codebuild-regionId-accountId-output-bucket",
                "arn:aws:s3:::codebuild-regionId-accountId-output-bucket/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

2. 정책의 이름을 제공합니다(예: `BankdemoCodeBuildRolePolicy`).

5단계 - IAM 역할 생성

이 단계에서는 이전에 만든 IAM 정책을 이 새 IAM 역할에 연결한 후 CodeBuild가 AWS 리소스와 상호 작용할 수 있도록 하는 새 IAM 역할을 생성합니다.

서비스 역할 생성에 대한 자세한 내용을 알아보려면 IAM 사용자 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

1. IAM 콘솔에 로그인하고 왼쪽 탐색 창에서 역할을 선택합니다.
2. 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔터티 유형에서 AWS 서비스를 선택합니다.
4. 다른 AWS 서비스의 사용 사례에서 CodeBuild를 선택한 다음 CodeBuild를 다시 선택합니다.
5. 다음을 선택합니다.
6. 권한 추가 페이지에서 다음을 선택합니다. 나중에 역할에 정책을 할당합니다.
7. 역할 세부 정보에서 역할의 이름을 제공합니다(예: `BankdemoCodeBuildServiceRole`).
8. 신뢰할 수 있는 엔티티 선택에서 정책 문서가 다음과 같은지 확인합니다.

```
{
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "codebuild.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

9. 역할 생성을 선택합니다.

6단계: IAM 정책을 IAM 역할에 연결합니다

이 단계에서 IAM 정책을 이전 단계에서 생성한 BankdemoCodeBuildServiceRole IAM 역할에 연결합니다.

1. IAM 콘솔에 로그인하고 왼쪽 탐색 창에서 역할을 선택합니다.
2. 역할에서 이전에 생성한 역할을 선택합니다(예: BankdemoCodeBuildServiceRole).
3. 권한 정책에서 권한 추가와 정책 연결을 차례로 선택합니다.
4. 기타 권한 정책에서 이전에 만든 정책(예: m2CodeBuildPolicy 및 BankdemoCodeBuildRolePolicy)을 선택합니다.
5. 정책 연결을 선택합니다.

7단계: CodeBuild 프로젝트 생성

이 단계에서는 CodeBuild에서 프로젝트를 생성합니다.

1. CodeBuild 콘솔에 로그인하고 빌드 프로젝트 생성을 선택합니다.
2. 프로젝트 구성 섹션에서 프로젝트 이름을 입력합니다(예: codebuild-bankdemo-project).
3. 소스 섹션의 소스 공급자에서 Amazon S3를 선택한 다음 이전에 생성한 입력 버킷을 선택합니다(예: codebuild-regionId-accountId-input-bucket).
4. S3 객체 키 또는 S3 폴더 필드에 S3 버킷에 업로드한 zip 파일의 이름을 입력합니다. 이 파일의 파일 이름은 bankdemo.zip입니다.
5. 환경 섹션에서 사용자 지정 이미지를 선택합니다.
6. 환경 유형 필드에서 Linux를 선택합니다.
7. 이미지 레지스트리에서 기타 레지스트리를 선택합니다.
8. 외부 레지스트리 URL 필드에 673918848628.dkr.ecr.us-west-2.amazonaws.com/m2-enterprise-build-tools:latest를 입력합니다

9. 서비스 역할에서 기존 서비스 역할을 선택하고, 역할 ARN 필드에서 이전에 생성한 서비스 역할(예: `BankdemoCodeBuildServiceRole`)을 선택합니다.
10. 빌드 사양에서 빌드 사양 파일 사용을 선택합니다.
11. 아티팩트 섹션의 유형에서 Amazon S3를 선택한 다음 출력 버킷을 선택합니다(예: `codebuild-regionId-accountId-output-bucket`).
12. 이름 필드에 빌드 출력 아티팩트를 포함하려는 버킷의 폴더 이름을 입력합니다(예: `bankdemo-output.zip`).
13. 아티팩트 패키징에서 압축을 선택합니다.
14. 빌드 프로젝트 생성을 선택합니다.

8단계: 빌드 시작

이 단계에서는 빌드를 시작합니다.

1. CodeBuild 콘솔에 로그인합니다.
2. 왼쪽 탐색 창에서 빌드 프로젝트를 선택합니다.
3. 이전에 만든 빌드 프로젝트(예: `codebuild-bankdemo-project`)를 선택합니다.
4. 빌드 시작를 선택합니다.

이 명령은 빌드를 시작합니다. 빌드는 비동기적으로 실행됩니다. 명령의 출력은 속성 ID가 포함된 JSON입니다. 이 속성 ID는 방금 시작한 빌드의 CodeBuild 빌드 ID를 참조합니다. CodeBuild 콘솔에서 빌드의 상태를 볼 수 있습니다. 또한 콘솔에서 빌드 실행에 대한 자세한 로그를 볼 수 있습니다. 자세한 내용은 AWS CodeBuild 사용 설명서의 세부 빌드 정보 보기를 참조하세요.

현재 단계가 완료되면 빌드가 성공적으로 완료되고 컴파일된 아티팩트가 Amazon S3에서 준비되었음을 의미합니다.

9단계: 출력 아티팩트 다운로드

이 단계에서는 Amazon S3에서 출력 아티팩트를 다운로드합니다. Micro Focus 빌드 도구는 여러 가지 실행 파일 유형을 생성할 수 있습니다. 이 자습서에서는 공유 객체를 생성합니다.

1. Amazon S3 콘솔에 로그인합니다.
2. 버킷 섹션에서 출력 버킷의 이름을 선택합니다(예: `codebuild-regionId-accountId-output-bucket`).

3. 다운로드를 선택합니다.
4. 다운로드한 파일의 압축을 풉니다. 대상 폴더로 이동하여 빌드 아티팩트를 확인하세요. 여기에는 `.so` Linux 공유 객체가 포함됩니다.

리소스 정리

이 자습서 용도로 생성한 리소스가 더 이상 필요하지 않은 경우 삭제하여 추가 비용이 발생하지 않도록 하세요. 이렇게 하려면 다음 단계를 완료합니다.

- 이 자습서를 위해 생성한 S3 버킷을 삭제하세요. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 삭제를 참조하세요.
- 이 자습서에서 생성한 정책을 선택합니다. 자세한 내용은 IAM 사용자 설명서에서 IAM 정책 생성을 참조하세요.
- 이 자습서에서 생성한 IAM 역할을 선택합니다. 자세한 내용은 IAM 사용 설명서에서 역할 또는 인스턴스 프로필 삭제를 참조하세요.
- 이 자습서를 위해 만든 CodeBuild 프로젝트를 삭제합니다. 자세한 내용은 AWS CodeBuild 사용 설명서에서 CodeBuild에서 빌드 프로젝트 삭제를 참조하세요.

튜토리얼: Micro Focus Enterprise Developer와 함께 사용할 CI/CD 파이프라인 설정

이 자습서에서는 Micro Focus Enterprise Developer에서 BankDemo 샘플 애플리케이션을 가져오고, 편집하고, 컴파일하고, 실행한 다음 변경 사항을 적용하여 CI/CD 파이프라인을 트리거하는 방법을 보여줍니다.

목차

사전 조건

다음 파일을 다운로드하세요.

- `basic-infra.yaml`
  - 유럽(프랑크푸르트) 리전에서 다운로드.
  - 미국 동부(버지니아 북부) 리전에서 다운로드.
- `pipeline.yaml`
  - 유럽(프랑크푸르트) 리전에서 다운로드.
  - 미국 동부(버지니아 북부) 리전에서 다운로드.
- `m2-code-sync-function.zip`
  - 유럽(프랑크푸르트) 리전에서 다운로드.
  - 미국 동부(버지니아 북부) 리전에서 다운로드.
- `config_git.yml`

  - 유럽(프랑크푸르트) 리전에서 다운로드.
  - 미국 동부(버지니아 북부) 리전에서 다운로드.
- BANKDEMO-source.zip
  - 유럽(프랑크푸르트) 리전에서 다운로드.
  - 미국 동부(버지니아 북부) 리전에서 다운로드.
- BANKDEMO-exercise.zip
  - 유럽(프랑크푸르트) 리전에서 다운로드.
  - 미국 동부(버지니아 북부) 리전에서 다운로드.

각 파일의 용도는 다음과 같습니다.

`basic-infra.yaml`

이 AWS CloudFormation 템플릿은 CI/CD 파이프라인에 필요한 기본 인프라(VPC, Amazon S3 버킷 등) 를 생성합니다.

`pipeline.yaml`

이 AWS CloudFormation 템플릿은 Lambda 함수에서 파이프라인 스택을 시작하는 데 사용됩니다. 이 템플릿이 공개적으로 액세스할 수 있는 Amazon S3 버킷에 있어야 합니다. 이 버킷에 대한 링크를 `basic-infra.yaml` 템플릿의 `PipelineTemplateURL` 파라미터 기본값으로 추가합니다.

`m2-code-sync-function.zip`

이 Lambda 함수는 `config_git.yaml`를 기반으로 하는 디렉터리 구조인 CodeCommit 리포지토리를 생성하고 `pipeline.yaml`를 사용하여 파이프라인 스택을 시작합니다. AWS 메인프레임 현대화가 지원되는 모든 AWS 리전에서 공개적으로 액세스할 수 있는 Amazon S3 버킷에서 이 zip 파일을 사용할 수 있는지 확인하세요. 파일을 하나의 AWS 리전 버킷에 저장하고 전체 AWS 리전의 버킷에 복제하는 것이 좋습니다. 특정 항목을 식별하는 접미사가 포함된 버킷의 이름 지정 규칙 AWS 리전(예:`m2-cicd-deployment-source-eu-west-1`)을 사용하고, `m2-cicd-deployment-source` 접두사를 `DeploymentSourceBucket` 매개 변수의 기본값으로 추가하고, `SourceSyncLambdaFunction` 리소스용 `basic-infra.yaml` 템플릿에서 해당 버킷을 참조하면서 AWS CloudFormation 대체 함수 `!Sub {DeploymentSourceBucket}-${AWS::Region}`를 사용하여 전체 버킷을 구성하세요.

`config_git.yml`

CodeCommit 디렉터리 구조 정의. 자세한 내용은 샘플 YAML 트리거 파일 config_git.yml 섹션을 참조하세요.

`BANKDEMO-source.zip`.

CodeCommit 리포지토리에서 생성된 BankDemo 소스 코드 및 구성 파일입니다.

`BANKDEMO-exercise.zip`.

CodeCommit 리포지토리에서 만든 튜토리얼 연습용 BankDemo 소스입니다.

CI/CD 파이프라인 기본 인프라 생성

AWS CloudFormation 템플릿 `basic-infra.yaml`을 사용하여 AWS CloudFormation 콘솔을 통해 CI/CD 파이프라인 기본 인프라 스택을 생성합니다. 이 스택은 애플리케이션 코드와 데이터를 업로드하는 Amazon S3 버킷과 AWS CodeCommit 리포지토리 및 AWS CodePipeline 파이프라인과 같은 기타 필수 리소스를 생성하는 지원 AWS Lambda 함수를 생성합니다.

> **Note**
>
> 이 스택을 시작하려면 IAM, Amazon S3, Lambd 및 AWS CloudFormation를 관리할 수 있는 권한과 AWS KMS를 사용할 권한이 필요합니다.

1. AWS Management Console에 로그인하여 https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
2. 다음 옵션 중 하나를 사용하여 새 스택을 만듭니다.
   - 스택 생성을 선택합니다. 현재 실행 중인 스택이 있는 경우에는 이 옵션이 유일한 옵션입니다.
   - Stacks(스택) 페이지에서 Create Stack(스택 생성)을 선택합니다. 현재 실행 중인 스택이 없는 경우에만 이 옵션이 표시됩니다.
3. 템플릿 지정 페이지에서:
   - Prepare template(템플릿 준비)에서 Template is ready(템플릿 준비가 완료되었습니다)를 선택합니다.
   - 템플릿 지정에서 Amazon S3 URL을 템플릿 소스로 선택하고 AWS 리전에 따라 다음 URL 중 하나를 입력합니다.
     - `https://m2-us-east-1.s3.us-east-1.amazonaws.com/cicd/mf/basic-infra.yaml`
     - `https://m2-eu-central-1.s3.eu-central-1.amazonaws.com/cicd/mf/basic-infra.yaml`

- 설정을 수락하려면 Next(다음)를 선택합니다.

스택 생성 페이지가 열립니다.

Specify stack details

Stack name

Stack name

m2-cicd-pipeline-foundation

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Networking Configuration

Do you want to use an existing VPC in your account?

If you select 'Yes', then you must provide the VPC ID and the Subnet IDs.

No (Create one)

Which VPC ID should be used?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Which private subnet ID should be used?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Which private subnet ID in a different AZ should be used for HA?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Enter the CIDR block that should be used for the new VPC

If you selected 'No (Create one)' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

11.11.0.0/16

CIDR bits for creating subnets. Choose 5 for /27, 6 for /26, 7 for /25, 8 for /24 range

If you selected 'No (Create one)' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

5

Deployment Configuration

Name of the S3 bucket which contains the source files for this stack deployment

Don't change unless you know what you are doing.

m2

Name of the source package file for the infrastructure Lambda function

Don't change unless you know what you are doing.

cicd/mf/m2-code-sync-function.zip

Full URL of the pipeline CloudFormation template file

Don't change unless you know what you are doing.

https://m2-us-east-1.s3.us-east-1.amazonaws.com/cicd/mf/pipeline.yaml

What name prefix to use for the new S3 buckets?

A name prefix for the S3 buckets that will be created by this stack.

mf

Cancel Previous Next

다음과 같이 변경합니다.

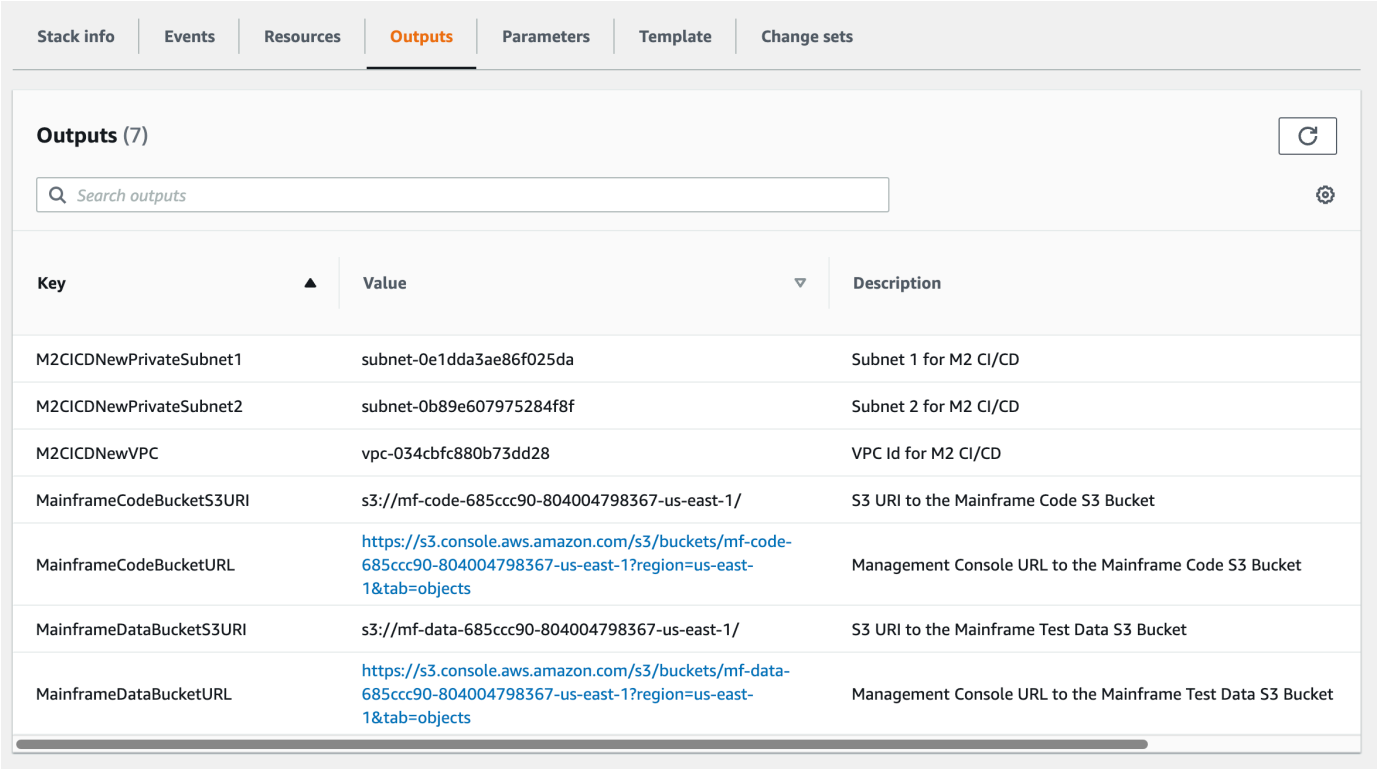
- 네트워킹 구성의 스택 이름 및 매개 변수에 적절한 값을 제공하세요.
- 배포 구성의 대부분의 파라미터는 적절하게 미리 채워지므로 수정할 필요가 없습니다. 에 따라 파이프라인 AWS CloudFormation 템플릿을 다음 Amazon S3 URL 중 하나로 변경합니다. AWS 리전
  - `https://m2-us-east-1.s3.amazonaws.com/cicd/mf/pipeline.yaml`
  - `https://m2-eu-central-1.s3.eu-central-1.amazonaws.com/cicd/mf/pipeline.yaml`
- 다음을 선택합니다.

> **Note**
>
> AWS CloudFormation 템플릿을 직접 수정하지 않은 이상 기본 파라미터 값을 변경하지 마세요.

4. 스택 옵션 구성에서 다음을 선택합니다.
5. Capacapity에서 내 대신에 IAM 역할을 생성할 수 있는 AWS CloudFormation에 대한 권한을 부여하기 위해 AWS CloudFormation가 IAM 리소스를 생성할 수 있음을 인정합니다를 선택합니다. 스택 생성을 선택합니다.

> **Note**
>
> 이 스택이 프로비저닝되는 데 3~5분이 걸릴 수 있습니다.

6. 스택이 성공적으로 생성되면 새로 프로비저닝된 스택의 출력 섹션으로 이동합니다. 여기에서 메인프레임 코드와 종속 파일을 업로드해야 하는 Amazon S3 버킷을 찾을 수 있습니다.



AWS CodeCommit 리포지토리 및 CI/CD 파이프라인을 생성합니다.

이 단계에서는 CodeCommit 리포지토리를 생성하고 파이프라인 스택 생성을 생성하기 위해 AWS CloudFormation를 직접적으로 호출하는 Lambda 함수를 호출하여 CI/CD 파이프라인 스택을 프로비저닝합니다.

1. BankDemo 샘플 애플리케이션을 로컬 시스템에 다운로드합니다.
2. 로컬 시스템에서 CI/CD 파이프라인 기본 인프라 생성에서 생성한 Amazon S3 버킷으로 `bankdemo.zip`를업로드합니다.
3. `config_git.yml`를 다운로드합니다.
4. 필요한 경우 다음과 같이 `config_git.yml`를 수정하세요.

   - 고유한 대상 리포지토리 이름, 대상 브랜치 및 커밋 메시지를 추가합니다.

   ```
   repository-config:
     target-repository: bankdemo-repo
     target-branch: main
     commit-message: Initial commit for bankdemo-repo main branch
   ```

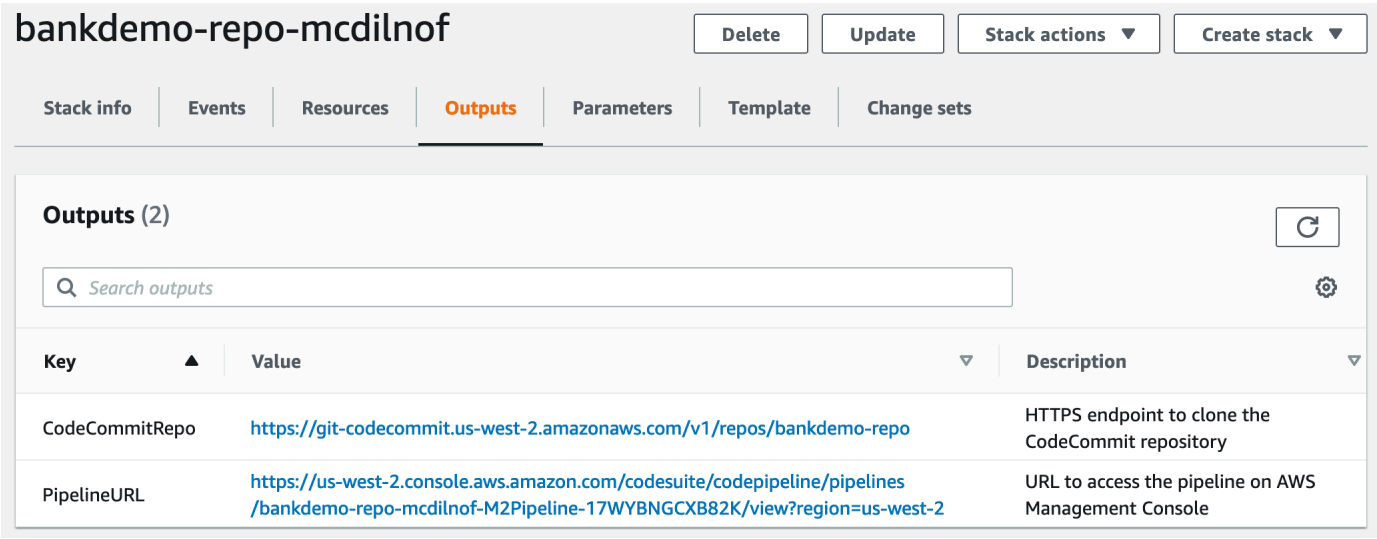
   - 알림을받을 이메일 주소를 입력합니다.

```
pipeline-config:
  # Send pipeline failure notifications to these email addresses
  alert-notifications:
    - myname@mycompany.com
  # Send notifications for manual approval before production deployment to these
 email addresses
  approval-notifications:
    - myname@mycompany.com
```

5. CodeCommit 리포지토리 폴더 구조의 정의가 포함된 `config_git.yml` 파일을 CI/CD 파이프라인 기본 인프라 생성에서 생성한 Amazon S3 버킷에 업로드합니다. 그러면 리포지토리와 파이프라인을 자동으로 프로비저닝하는 Lambda 함수가 직접적으로 호출됩니다.

   그러면 `config_git.yml` 파일에 정의된 `target-repository`에 제공된 이름(예: `bankdemo-repo`)으로 CodeCommit 리포지토리가 만들어집니다.

   또한 Lambda 함수는 AWS CloudFormation를 통해 CI/CD 파이프라인 스택을 생성합니다. AWS CloudFormation 스택에는 제공된 `target-repository` 이름과 동일한 접두사가 있고 그 뒤에 임의의 문자열이 옵니다(예: `bankdemo-repo-`*`01234567`*. CodeCommit 리포지토리 URL과 생성된 파이프라인에 액세스하는 데 필요한 URL을 AWS 관리 콘솔에서 찾을 수 있습니다.



6. CodeCommit 리포지토리 생성이 완료되면 CI/CD 파이프라인이 즉시 트리거되어 전체 CI/CD를 수행합니다.
7. 파일이 푸시되면 파이프라인이 자동으로 트리거되어 빌드하여 스테이징 단계에 배포하고 일부 테스트를 실행한 후 수동 승인을 기다린 후 프로덕션 환경에 배포합니다.

샘플 YAML 트리거 파일 config_git.yml

```
repository-config:
  target-repository: bankdemo-repo
  target-branch: main
  commit-message: Initial commit for bankdemo-repo main branch
  directory-structure:
    - '/':
        files:
        - build.xml
        - '*.yaml'
        - '*.yml'
        - '*.xml'
        - 'LICENSE.txt'
        readme: |
          # Root Folder
          - 'build.xml' : Build configuration for the application
    - tests:
        files:
        - '*.py'
        readme: |
          # Test Folder
          - '*.py' : Test scripts
    - config:
        files:
        - 'BANKDEMO.csd'
        - 'BANKDEMO.json'
        - 'BANKDEMO_ED.json'
        - 'dfhdrdat'
        - 'ESPGSQLXA.dll'
        - 'ESPGSQLXA64.so'
        - 'ESPGSQLXA64_S.so'
        - 'EXTFH.cfg'
        - 'm2-2021-04-28.normal.json'
        - 'MFDBFH.cfg'
        - 'application-definition-template-config.json'
        readme: |
          # Config Folder
          This folder contains the application configuration files.
          - 'BANKDEMO.csd'     : CICS Resource definitions export file
          - 'BANKDEMO.json'    : Enterprise Server configuration
          - 'BANKDEMO_ED.json' : Enterprise Server configuration for ED
          - 'dfhdrdat'         : CICS resource definition file
```

```
          - 'ESPGSQLXA.dll'    : XA switch module Windows
          - 'ESPGSQLXA64.so'   : XA switch module Linux
          - 'ESPGSQLXA64_S.so' : XA switch module Linux
          - 'EXTFH.cfg'        : Micro Focus File Handler configuration
          - 'm2-2021-04-28.normal.json' : M2 request document
          - 'MFDBFH.cfg'       : Micro Focus Database File Handler
          - 'application-definition-template-config.json' : Application definition for
 M2
    - source:
        subdirs:
        - .settings:
            files:
              - '.bms.mfdirset'
              - '.cbl.mfdirset'
        - copybook:
            files:
              - '*.cpy'
              - '*.inc'
            readme: |
              # Copy folder
              This folder contains the source for COBOL copy books, PLI includes, ...
              - .cpy COBOL copybooks
              - .inc PLI includes
#        - ctlcards:
#            files:
#              - '*.ctl'
#              - 'KBNKSRT1.txt'
#            readme: |
#              # Control Card folder
#              This folder contains the source for Batch Control Cards
#              - .ctl Control Cards
        - ims:
            files:
              - '*.dbd'
              - '*.psb'
            readme: |
              # ims folder
              This folder contains the IMS DB source files with the extensions
              - .dbd for IMS DBD source
              - .psb for IMS PSB source
        - jcl:
            files:
              - '*.jcl'
              - '*.ctl'
```

```
              - 'KBNKSRT1.txt'
              - '*.prc'
            readme: |
              # jcl folder
              This folder contains the JCL source files with the extensions
              - .jcl
#        - proclib:
#            files:
#              - '*.prc'
#            readme: |
#              # proclib folder
#              This folder contains the JCL procedures referenced via PROCLIB
 statements in the JCL with extensions
#              - .prc
        - rdbms:
            files:
              - '*.sql'
            readme: |
              # rdbms folder
              This folder contains any DB2 related source files with extensions
              - .sql for any kind of SQL source
        - screens:
            files:
            - '*.bms'
            - '*.mfs'
            readme: |
              # screens folder
              This folder contains the screens source files with the extensions
              - .bms for CICS BMS screens
              - .mfs for IMS MFS screens
            subdirs:
            - .settings:
                files:
                    - '*.bms.mfdirset'
        - cobol:
            files:
              - '*.cbl'
              - '*.pli'
            readme: |
              # source folder
              This folder contains the program source files with the extensions
              - .cbl for COBOL source
              - .pli for PLI source
            subdirs:
```

```
            - .settings:
                files:
                   - '*.cbl.mfdirset'
    - tests:
        files:
        - 'test_script.py'
        readme: |
          # tests Folder
          This folder contains the application test scripts
pipeline-config:
  alert-notifications:
    - myname@mycompany.com
  approval-notifications:
    - myname@mycompany.com
```

Enterprise Developer AppStream 2.0 생성

AppStream 2.0에서 Micro Focus Enterprise Developer를 설정하려면 자습서: AppStream 2.0에서 Micro Focus Enterprise Developer 설정을 참조하세요.

CodeCommit 리포지토리를 Enterprise Developer에 연결하려면 샘플 YAML 트리거 파일 config_git.yml의 `target-repository`에서 지정된 이름을 사용합니다.

Enterprise Developer 설치 및 테스트

주제

- Enterprise Developer의 BankDemo CodeCommit 리포지토리 복제
- BankDemo 메인프레임 COBOL 프로젝트를 생성하고 애플리케이션을 빌드합니다
- 테스트를 위한 로컬 BankDemo CICS 및 배치 환경을 생성합니다
- Enterprise Developer에서 BANKDEMO 서버를 시작합니다
- Rumba 3270 터미널을 시작합니다
- BankDemo 거래를 실행하세요
- Enterprise Developer의 BANKDEMO 서버를 중지하세요

Enterprise Developer AppStream 2.0 생성에서 만든 Enterprise Developer AppStream 2.0 인스턴스에 연결합니다.

1. Windows 시작에서 Enterprise Developer를 시작합니다. Micro Focus Enterprise Developer를 선택한 다음 Eclipse용 Enterprise Developer를 선택합니다. 처음 시작하는 경우 시간이 좀 걸릴 수 있습니다.
2. Eclipse 런처의 작업 영역에서: `C:\Users\<username>\workspace`를 입력한 다음 실행을 선택합니다.

   > **Note**
   >
   > AppStream 2.0 인스턴스에 다시 연결한 후 동일한 위치를 선택해야 합니다. 작업 영역 선택은 영구적이지 않습니다.

3. 시작 에서 COBOL 퍼스펙티브 열기를 선택합니다. 이는 새 작업 공간의 경우 처음 표시될 때만 표시됩니다.

Enterprise Developer의 BankDemo CodeCommit 리포지토리 복제

1. Window / Perspective / Open Perspective / Other ... Git을 선택합니다.
2. Git 리포지토리 복제를 선택합니다.
3. Git 리포지토리 복제에 다음 정보를 입력합니다.

   - 위치 URI에 CodeCommit 리포지토리의 HTTPS URL을 입력합니다.

     > **Note**
     >
     > 관리 콘솔에서 CodeCommit 리포지토리의 클론 URL AWS HTTPS를 복사하여 여기에 붙여넣습니다. URI는 호스트 경로와 리포지토리 경로로 분할됩니다.

   - 인증 사용자 및 암호에서 사용자 CodeCommit 리포지토리 보안 인증을 선택하고 보안 저장에서 저장을 선택합니다.
4. 브랜치 선택에서 메인 브랜치를 선택한 후 다음을 선택합니다.
5. 로컬 대상의 디렉터리에서 `C:\Users\<username>\workspace`를 입력하고 마침을 선택합니다.

   `BANKDEMO [main]`가 Git 리포지토리 보기에 표시되면 복제 프로세스가 완료됩니다.

BankDemo 메인프레임 COBOL 프로젝트를 생성하고 애플리케이션을 빌드합니다

1. COBOL 퍼스펙티브로 변경하세요.
2. 프로젝트에서 자동 빌드를 사용 중단합니다.
3. 파일에서 새로 만들기를 선택한 다음 메인프레임 COBOL 프로젝트를 선택합니다.
4. 새 메인프레임 COBOL 프로젝트에서 다음 정보를 입력합니다.
   - 프로젝트 이름에 `BankDemo`을 입력합니다.
   - Micro Focus 템플릿 [64비트]를 선택합니다.
   - 마침을 클릭합니다.
5. COBOL 익스플로러에서 새 BankDemo 프로젝트를 확장하세요.

   > **Note**
   >
   > 대괄호 안의 `[BANKDEMO main]`은 프로젝트가 로컬 BankDemo CodeCommit 리포지토리와 연결되어 있음을 나타냅니다.

6. 트리 뷰에 COBOL 프로그램, 카피북, BMS 소스 및 JCL 파일에 대한 항목이 표시되지 않는 경우 BankDemo 프로젝트 컨텍스트 메뉴에서 새로 고침을 선택하세요.
7. BankDemo 컨텍스트 메뉴에서 속성 / Micro Focus / 프로젝트 설정 / COBOL 을 선택합니다.
   - 문자 세트 - ASCII를 선택합니다.
   - Apply(적용)를 선택한 다음 OK(확인)를 선택합니다.
8. BMS 및 COBOL 소스 빌드가 즉시 시작되지 않는 경우 프로젝트 메뉴에서 자동 빌드 옵션이 사용 설정되어 있는지 확인하세요.

   빌드 출력은 콘솔 뷰에 표시되며 몇 분 후 `BUILD SUCCESSFUL` 및 `Build finished with no errors` 메시지와 함께 완료될 것입니다.

   이제 BankDemo 애플리케이션을 컴파일하고 로컬에서 실행할 준비가 되었습니다.

테스트를 위한 로컬 BankDemo CICS 및 배치 환경을 생성합니다

1. COBOL 탐색기에서 `BANKDEMO / config`를 펼치세요.
2. 에디터에서 `BANKDEMO_ED.json`를 엽니다.

3. ED_Home= 문자열을 찾아 D:\\<username>\\workspace\\BANKDEMO와 같이 Enterprise Developer 프로젝트를 가리키도록 경로를 변경합니다. 경로 정의에 이중 슬래시(\\)가 사용된다는 점에 유의하세요.
4. 파일을 저장하고 닫습니다.
5. 서버 탐색기를 선택합니다.
6. 기본 컨텍스트 메뉴에서 관리 페이지 열기를 선택합니다. Micro Focus 엔터프라이즈 서버 관리 페이지는 기본 브라우저에서 열립니다.
7. AppStream 2.0 세션의 경우에만 로컬 테스트를 위해 로컬 엔터프라이즈 서버 리전을 보존할 수 있도록 다음과 같이 변경하세요.

   - 디렉토리 서버/기본값에서 속성/구성을 선택합니다.
   - 리포지토리 위치를 D:\<username>\My Files\Home Folder\MFDS로 바꿉니다.

   > **Note**
   >
   > AppStream 2.0 인스턴스에 새로 연결할 때마다 5~8단계를 완료해야 합니다.

8. 디렉토리 서버/기본값에서 가져오기를 선택하고 다음 단계를 완료하세요.

   - 1단계: 가져오기 유형에서 JSON을 선택하고 다음을 선택합니다.
   - 2단계: 업로드에서 파란색 사각형으로 파일을 클릭하여 업로드합니다.
   - 업로드할 파일 선택에서 다음을 입력합니다.
     - 파일 이름: D:\<username>\workspace\BANKDEMO\config\BANKDEMO_ED.json.
     - 열기(Open)를 선택합니다.
   - 다음을 선택합니다.
   - 3단계: 리전에서 엔드포인트에서 포트 지우기를 수행합니다.
   - 다음을 선택합니다.
   - 4단계: 가져오기에서 가져오기를 선택합니다.
   - 마침을 클릭합니다.

   이제 목록에 새 서버 이름 BANKDEMO이 표시됩니다.

Enterprise Developer에서 BANKDEMO 서버를 시작합니다

1. Enterprise Developer를 선택합니다.
2. 서버 탐색기에서 기본값을 선택한 다음 컨텍스트 메뉴에서 새로 고침을 선택합니다.

   이제 서버 목록에 BANKDEMO도 표시되어야 합니다.
3. BANKDEMO를 선택하세요.
4. 컨텍스트 메뉴에서 프로젝트와 연결을 선택한 다음 BANKDEMO를 선택합니다.
5. 컨텍스트 메뉴에서 시작을 선택합니다.

   콘솔 보기에는 서버 시작 로그가 표시되어야 합니다.

   `BANKDEMO CASSI5030I PLTPI Phase 2 List(PI) Processing Completed` 메시지가 표시되면 서버는 CICS BANKDEMO 애플리케이션을 테스트할 준비가 된 것입니다.

Rumba 3270 터미널을 시작합니다

1. Windows 시작에서 Micro Focus Rumba+ 데스크탑/Rumba+ 데스크탑을 실행합니다.
2. 시작에서 새 세션 만들기/메인프레임 디스플레이 만들기를 선택합니다.
3. 메인프레임 디스플레이에서 연결/ 구성을 선택합니다.
4. 세션 구성에서 연결/TN3270를 선택합니다.
5. 호스트 이름/주소에서 삽입을 선택하고 IP 주소 `127.0.0.1`를 입력합니다.
6. 텔넷 포트에 `6000` 포트를 입력합니다.
7. Apply(적용)를 선택합니다.
8. 연결을 선택합니다.

   CICS 시작 화면에는 1행 메시지 `This is the Micro Focus MFE CICS region BANKDEMO`가 있는 화면이 표시됩니다.
9. Ctrl+Shift+Z를 누르면 화면이 지워집니다.

BankDemo 거래를 실행하세요

1. 빈 화면에서 `BANK`를 입력합니다.
2. BANK10 화면의 사용자 ID.....: 입력 필드에 `guest`를 입력하고 Enter를 누릅니다.
3. BANK20 화면의 대출 비용 계산 전 입력 필드에 /(슬래시 전달)을 입력하고 Enter 키를 누릅니다.

4. BANK70 화면에서:

   - 빌리고 싶은 금액...에 `10000`를 입력하세요.
   - 이자율.....:에 `5.0`를 입력하세요.
   - 대출 기간.....에 `10`를 입력하세요.
   - Enter를 누릅니다.

   다음 결과가 표시되어야 합니다.

   ```
   Resulting monthly payment.............:   $1023.06
   ```

   이것으로 Enterprise Developer의 BANKDEMO 애플리케이션 설정이 완료됩니다.

Enterprise Developer의 BANKDEMO 서버를 중지하세요

1. 서버 탐색기에서 기본값을 선택한 다음 컨텍스트 메뉴에서 새로 고침을 선택합니다.
2. BANKDEMO를 선택합니다.
3. 컨텍스트 메뉴에서 중지를 선택합니다.

   콘솔 보기에는 서버 중지 로그가 표시되어야 합니다.

   `Server: BANKDEMO stopped successfully` 메시지가 표시되면 서버가 성공적으로 종료된 것입니다.

연습 1: BANKDEMO 애플리케이션의 대출 계산 향상

주제

- Enterprise Developer 코드 분석에 대출 분석 규칙 추가
- 1단계: 대출 계산을 위한 코드 분석 수행
- 2단계: CICS BMS 맵 및 COBOL 프로그램 수정 및 테스트
- 3단계: COBOL 프로그램에 총액 계산 추가
- 4단계: 변경 사항 커밋 및 CI/CD 파이프라인 실행

이 시나리오에서는 코드에 샘플 변경을 적용하고, 코드를 배포하고, 테스트하는 프로세스를 단계별로 살펴봅니다.

대출 부서에서는 대출 계산 화면(BANK70) 에 총 대출 금액을 표시하는 새 필드를 만들려고 합니다. 이를 위해서는 BMS 화면 MBANK70 .CBL을 변경하여 새 필드를 추가하고 관련 카피북이 있는 해당 화면 처리 프로그램인 SBANK70P.CBL을 추가해야 합니다. 또한 추가 공식을 사용하여 BBANK70P .CBL의 대출 계산 루틴을 확장해야 합니다.

이 연습을 완료하려면 다음 사전 조건을 충족합니다.

- BANKDEMO-exercise.zip를 `D:\PhotonUser\My Files\Home Folder`에 다운로드하세요.
- `D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise`에 zip 파일의 압축을 풉니다.
- `D:\PhotonUser\My Files\Home Folder\AnalysisRules` 폴더를 만듭니다.
- `BANKDEMO-exercise` 폴더의 규칙 파일을 `D:\PhotonUser\My Files\Home Folder \AnalysisRules`로 복사합니다.

> **Note**
>
> *.CBL 및 *.CPY의 코드 변경 내용은 이 연습의 1~6열에 EXER01 표시로 표시되어 있습니다.

Enterprise Developer 코드 분석에 대출 분석 규칙 추가

Micro Focus Enterprise Analyzer에 정의된 분석 규칙을 엔터프라이즈 분석기에서 내보낸 후 Enterprise Developer로 가져와서 Enterprise Developer 프로젝트의 소스에서 동일한 분석 규칙을 실행할 수 있습니다.

1. `Window/Preferences/Micro Focus/COBOL/Code Analysis/Rules`를 엽니다.
2. 편집...을 선택하고 `Loan+Calculation+Update.General-1.xml` 규칙 파일이 들어 있는 `D:\PhotonUser\My Files\Home Folder\AnalysisRules` 폴더 이름을 입력합니다.
3. 마침을 클릭합니다.
4. 적용을 선택한 다음 닫기를 선택합니다.
5. BANKDEMO 프로젝트 컨텍스트 메뉴에서 코드 분석을 선택합니다.

   대출 계산 업데이트 항목이 표시되어야 합니다.

1단계: 대출 계산을 위한 코드 분석 수행

새 분석 규칙을 사용하여 식, 명령문 및 변수의 *PAYMENT*, *LOAN* 및 *RATE* 검색 패턴과 일치하는 COBOL 프로그램 및 코드 라인을 식별하려고 합니다. 이렇게 하면 코드를 탐색하고 필요한 코드 변경을 식별하는 데 도움이 됩니다.

1. BANKDEMO 프로젝트 컨텍스트 메뉴에서 코드 분석/대출 계산 업데이트를 선택합니다.

   그러면 검색 규칙이 실행되고 코드 분석이라는 새 탭에 결과가 나열됩니다. 오른쪽 하단의 녹색 진행률 표시줄이 사라지면 분석 실행이 완료됩니다.

   코드 분석 탭에는 검색 패턴과 일치하는 명령문, 식 및 변수를 나열하는 BBANK20P.CBL, BBANK70P.CBL 및 SBANK70P.CBL의 확장된 목록이 표시되어야 합니다.

   BBANK20P.CBL의 결과를 보면 검색 패턴과 일치하는 이동된 리터럴만 있습니다. 따라서 이 프로그램은 무시할 수 있습니다.
2. 탭 메뉴 막대에서 - 아이콘을 선택하면 모두 축소됩니다.
3. 두 번 클릭하여 원하는 SBANK70P.CBL를 펼치고 순서대로 원하는 줄을 선택하면 소스가 어떻게 열리고 소스 코드에서 선택한 줄이 강조 표시되는지 확인할 수 있습니다. 식별된 소스 라인이 모두 표시되어 있는 것도 확인할 수 있습니다.

2단계: CICS BMS 맵 및 COBOL 프로그램 수정 및 테스트

먼저 BMS MBANK70.BMS 맵과 화면 처리 프로그램 SBANK70P.CBL 및 CBANKDAT.CPY 카피북을 변경하여 새 필드를 표시합니다. 이 연습에서는 불필요한 코딩을 방지하기 위해 D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise\Exercise01 폴더에서 수정된 소스 모듈을 사용할 수 있습니다. 일반적으로 개발자는 코드 분석 결과를 사용하여 소스를 탐색하고 수정합니다. 시간이 있고 수동으로 변경하려면 *MBANK70.BMS 및 SBANK70P.CBL의 수동 변경(선택 사항) *에 제공된 정보를 사용하여 수동으로 변경하세요.

빠르게 변경하려면 다음 파일을 복사하세요.

1. ..\BANKDEMO-exercise\Exercis01\screens\MBANK70.BMS ~ D:\PhotonUser \workspace\bankdemo\source\screens.
2. .\BANKDEMO-exercise\Exercis01\cobol\SBANK70P.CBL ~ D:\PhotonUser \workspace\bankdemo\source\cobol.
3. ..\BANKDEMO-exercise\Exercis01\copybook\CBANKDAT.CPY ~ D:\PhotonUser \workspace\bankdemo\source\copybook.

4. 변경 사항의 영향을 받는 모든 프로그램이 컴파일되도록 하려면 프로젝트/...지우기/모든 프로젝트 지우기를 선택하세요.

MBANK70.BMS 및 SBANK70P.CBL를 수동으로 변경하려면 다음 단계를 완료하세요.

- BMS MBANK70.BMS 소스를 수동으로 변경하려면 PAYMENT 필드 뒤에 추가하세요.
  - TXT08 및 INITIAL 값 “총 대출액”과 동일한 속성의 TXT09
  - 결제와 동일한 속성을 가진 총액

테스트 변경

변경 사항을 테스트하려면 다음 단계를 반복합니다.

1. Enterprise Developer에서 BANKDEMO 서버를 시작합니다
2. Rumba 3270 터미널을 시작합니다
3. BankDemo 거래를 실행하세요

   또한 이제 Total Loan Amount.....: 텍스트도 볼 수 있을 것입니다.
4. Enterprise Developer의 BANKDEMO 서버를 중지하세요

3단계: COBOL 프로그램에 총액 계산 추가

두 번째 단계에서는 BBANK70P.CBL를 변경하고 총 대출 금액에 대한 계산을 추가합니다. 필요한 변경 사항이 포함된 준비된 출처는 D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise\Exercise01 폴더에서 확인할 수 있습니다. 시간이 있고 수동으로 변경하려면 *BBANK70P.CBL의 수동 변경(선택 사항)*에 제공된 정보를 사용하여 수동으로 변경하세요.

빠르게 변경하려면 다음 파일을 복사하세요.

- ..\BANKDEMO-exercise\Exercis01\source\cobol\BBANK70P.CBL ~ D:\PhotonUser\workspace\bankdemo\source\cobol.

BBANK70P.CBL을 수동으로 변경하려면 다음 단계를 완료합니다.

- 코드 분석 결과를 사용하여 필요한 변경을 식별하세요.

테스트 변경

변경 사항을 테스트하려면 다음 단계를 반복합니다.

1. Enterprise Developer에서 BANKDEMO 서버를 시작합니다
2. Rumba 3270 터미널을 시작합니다
3. BankDemo 거래를 실행하세요

   또한 이제 `Total Loan Amount.....................: $10230.60` 텍스트도 볼 수 있을 것입니다.
4. Enterprise Developer의 BANKDEMO 서버를 중지하세요

4단계: 변경 사항 커밋 및 CI/CD 파이프라인 실행

중앙 CodeCommit 리포지토리에 변경 사항을 적용하고 CI/CD 파이프라인을 트리거하여 변경 사항을 빌드, 테스트 및 배포합니다.

1. BANKDEMO 프로젝트의 컨텍스트 메뉴에서 팀/커밋을 선택합니다.
2. Git 스테이징 탭에서 `Added Total Amount Calculation` 커밋 메시지를 입력합니다.
3. 커밋 및 푸시...를 선택합니다.
4. CodePipeline 콘솔을 열고 파이프라인 실행 상태를 확인합니다.

> **Note**
>
> Enterprise Developer 또는 팀 함수 커밋 또는 푸시에 문제가 있는 경우 Git Bash 명령줄 인터페이스를 사용하세요.

연습 2: BankDemo 애플리케이션에서 대출 계산 추출

주제

- 1단계: 대출 계산 루틴을 COBOL 섹션으로 리팩터링
- 2단계: 대출 계산 루틴을 독립형 COBOL 프로그램으로 추출
- 3단계: 변경 사항을 적용하고 CI/CD 파이프라인을 실행합니다

다음 연습에서는 또 다른 샘플 변경 요청을 처리합니다. 이 시나리오에서 대출 부서는 대출 계산 루틴을 독립형 웹 서비스로 재사용하려고 합니다. 루틴은 COBOL에 유지되어야 하며 기존 CICS COBOL 프로그램 BBANK70P.CBL에서 여전히 호출할 수 있어야 합니다.

1단계: 대출 계산 루틴을 COBOL 섹션으로 리팩터링

첫 번째 단계에서는 대출 계산 루틴을 COBOL 섹션으로 추출합니다. 다음 단계에서 코드를 독립형 COBOL 프로그램으로 추출하려면 이 단계가 필요합니다.

1. COBOL 편집기에서 BBANK70P.CBL를 엽니다.
2. 편집기의 콘텍스트 메뉴 코드 분석/대출 계산 업데이트 중에서 선택합니다. 이렇게 하면 현재 소스에서 분석 규칙에 정의된 패턴만 스캔합니다.
3. 코드 분석 탭의 결과에서 첫 번째 산술 명령문 DIVIDE WS-LOAN-INTEREST BY 12을 찾으세요.
4. 명령문을 두 번 클릭하여 편집기의 소스 라인으로 이동합니다. 대출 계산 루틴의 첫 번째 명세서입니다.
5. 섹션으로 추출할 대출 계산 루틴에 대해 다음 코드 블록을 표시하세요.

```
DIVIDE WS-LOAN-INTEREST BY 12
             GIVING WS-LOAN-INTEREST ROUNDED.
           COMPUTE WS-LOAN-MONTHLY-PAYMENT ROUNDED =
             ((WS-LOAN-INTEREST * ((1 + WS-LOAN-INTEREST)
                 ** WS-LOAN-TERM)) /
             (((1 + WS-LOAN-INTEREST) * WS-LOAN-TERM) - 1 ))
                 * WS-LOAN-PRINCIPAL.
EXER01     COMPUTE WS-LOAN-TOTAL-PAYMENT =
EXER01               (WS-LOAN-MONTHLY-PAYMENT * WS-LOAN-TERM).
```

6. 편집기의 컨텍스트 메뉴에서 섹션으로 리팩터링/추출...을 선택합니다.
7. 새 섹션 이름: LOAN-CALCULATION을 입력합니다.
8. 확인을 선택합니다.

   표시된 코드 블록은 이제 새 LOAN-CALCULATION 섹션으로 추출되었으며 코드 블록은 PERFROM LOAN-CALCULATION 명령문으로 대체되었습니다.

테스트 변경

변경 사항을 테스트하려면 다음 단원에서 설명하는 단계를 반복합니다.

1. Enterprise Developer에서 BANKDEMO 서버를 시작합니다
2. Rumba 3270 터미널을 시작합니다
3. BankDemo 거래를 실행하세요

   또한 이제 `Total Loan Amount.....: $10230.60` 텍스트도 볼 수 있을 것입니다.
4. Enterprise Developer의 BANKDEMO 서버를 중지하세요

> **Note**
>
> 위의 단계를 수행하여 코드 블록을 섹션으로 추출하지 않으려면 1단계의 수정된 소스를 `..\BANKDEMO-exercise\Exercis02\Step1\cobol\BBANK70P.CBL`에서 `D:\PhotonUser\workspace\bankdemo\source\cobol`로 복사할 수 있습니다.

2단계: 대출 계산 루틴을 독립형 COBOL 프로그램으로 추출

2단계에서는 `LOAN-CALCULATION` 섹션의 코드 블록을 독립형 프로그램으로 추출하고 원본 코드는 새 하위 프로그램을 직접적으로 호출하는 코드로 대체됩니다.

1. 편집기에서 `BBANK70P.CBL`를 열고 1단계에서 만든 새 `PERFORM LOAN-CALCULATION` 명령문을 찾으세요.
2. 섹션 이름 안에 커서를 놓습니다. 회색으로 표시됩니다.
3. 컨텍스트 메뉴에서 리팩터링->프로그램으로 섹션/단락 추출...을 선택합니다.
4. 프로그램으로 섹션/단락 추출에서 새 파일 이름인 LOANCALC.CBL을 입력합니다.
5. 확인을 선택합니다.

   새 `LOANCALC.CBL` 프로그램이 편집기에서 열립니다.
6. 아래로 스크롤하여 직접적 호출 인터페이스용으로 추출 및 생성되는 코드를 검토하세요.
7. `BBANK70P.CBL`에서 편집기를 선택하고 `LOAN-CALCULATION SECTION`으로 이동하세요. 생성되는 코드를 검토하여 `LOANCALC.CBL` 새 하위 프로그램을 직접적으로 호출하세요.

> **Note**
>
> `CALL` 명령문은 CICS 제어 블록을 사용하여 `LOANCALC`를 직접적으로 호출하기 위해 `DFHEIBLK` 및 `DFHCOMMAREA`를 사용합니다. 새 `LOANCALC.CBL` 하위 프로그램을

CICS가 아닌 프로그램으로 호출하려면 코멘트 아웃 또는 삭제를 통해 직접적 호출에서 DFHEIBLK 및 DFHCOMMAREA를 제거해야 합니다.

테스트 변경

변경 사항을 테스트하려면 다음 단원에서 설명하는 단계를 반복합니다.

1. Enterprise Developer에서 BANKDEMO 서버를 시작합니다
2. Rumba 3270 터미널을 시작합니다
3. BankDemo 거래를 실행하세요

   또한 이제 `Total Loan Amount.....................: $10230.60` 텍스트도 볼 수 있을 것입니다.
4. Enterprise Developer의 BANKDEMO 서버를 중지하세요

> **Note**
>
> 위의 단계를 수행하여 코드 블록을 섹션으로 추출하지 않으려면 1단계의 수정된 소스를 `..\BANKDEMO-exercise\Exercis02\Step2\cobol\BBANK70P.CBL` 및 `LOANCALC.CBL` 에서 `D:\PhotonUser\workspace\bankdemo\source\cobol`으로 복사할 수 있습니다.

3단계: 변경 사항을 적용하고 CI/CD 파이프라인을 실행합니다

중앙 CodeCommit 리포지토리에 변경 사항을 커밋하고 CI/CD 파이프라인을 트리거하여 변경 사항을 빌드, 테스트 및 배포합니다.

1. BANKDEMO 프로젝트의 컨텍스트 메뉴에서 팀/커밋을 선택합니다.
2. Git 스테이징 탭에서
   - 언스테이징 스테이지 LOANCALC.CBL 및 LOANCALC.CBL.mfdirset에 추가합니다.
   - `Added Total Amount Calculation` 커밋 메시지를 입력합니다.
3. 커밋 및 푸시...를 선택합니다.
4. CodePipeline 콘솔을 열고 파이프라인 실행 상태를 확인합니다.

> **Note**
>
> Enterprise Developer 또는 Teams 함수 커밋 또는 푸시에 문제가 있는 경우 Git Bash 명령 줄 인터페이스를 사용하세요.

리소스 정리

이 튜토리얼에서 만든 리소스가 더 이상 필요하지 않은 경우 비용이 청구되지 않도록 하려면 리소스에 대해 비용이 청구되지 않도록 하세요. 다음 단계를 완료합니다.

- 코드파이프라인 파이프라인을 삭제합니다. 자세한 내용을 알아보려면 AWS CodePipeline 사용 설명서의 코드파이프라인에서 파이프라인 만들기를 참조하세요.
- CodeCommit 리포지토리를 삭제합니다. 자세한 내용은 사용 AWS CodeCommit 사용 설명서의 CodeCommit 리포지토리 삭제를 참조하세요.
- S3 버킷을 삭제합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 삭제를 참조하세요.
- AWS CloudFormation 스택을 삭제합니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation 콘솔에서 스택 삭제를 참조하세요.

AWS 메인프레임 현대화의 Batch 유틸리티

메인프레임 애플리케이션은 주로 배치 유틸리티 프로그램을 사용하여 데이터 정렬, FTP를 사용한 파일 전송, DB2와 같은 데이터베이스로 데이터 로드, 데이터베이스에서 데이터 언로드 등과 같은 특정 기능을 수행합니다.

애플리케이션을 메인프레임 현대화로 마이그레이션할 때는 AWS 메인프레임에서 사용한 것과 동일한 작업을 수행할 수 있는 기능적으로 동등한 대체 유틸리티가 필요합니다. 이러한 유틸리티 중 일부는 이미 AWS 메인프레임 현대화 런타임 엔진의 일부로 제공될 수 있지만, 당사는 다음과 같은 대체 유틸리티를 제공하고 있습니다.

- M2SFTP - SFTP 프로토콜을 사용하여 파일을 안전하게 전송할 수 있습니다.
- M2WAIT - 배치 작업의 다음 단계를 계속하기 전에 지정된 시간 동안 기다립니다.
- TXT2PDF - 텍스트 파일을 PDF 형식으로 변환합니다.
- M2DFUTIL - 메인프레임 ADRDSSU 유틸리티에서 제공하는 지원과 유사한 데이터 세트에 대한 백업, 복원, 삭제 및 복사 기능을 제공합니다.

- M2RUNCMD - Micro Focus 명령, 스크립트 및 시스템 호출을 JCL에서 직접 실행할 수 있습니다.

당사는 고객 피드백을 기반으로 이러한 배치 유틸리티를 개발하고 메인프레임 유틸리티와 동일한 기능을 제공하도록 설계했습니다. 목표는 메인프레임에서 메인프레임 현대화로 최대한 원활하게 전환하는 것입니다 AWS .

주제

이진 위치

이러한 유틸리티는 Micro Focus Enterprise Developer(ED) 및 Micro Focus Enterprise Server(ES) 제품에 사전 설치되어 있습니다. ED 및 ES의 모든 변형에 대해서는 다음 위치에서 찾을 수 있습니다.

- Linux: `/opt/aws/m2/microfocus/utilities/64bit`
- Windows(32비트): `C:\AWS\M2\MicroFocus\Utilities\32bit`
- Windows(64비트): `C:\AWS\M2\MicroFocus\Utilities\64bit`

M2SFTP 배치 유틸리티

M2SFTP는 보안 File Transfer 프로토콜(SFTP)을 사용하여 시스템 간에 안전한 파일 전송을 수행하도록 설계된 JCL 유틸리티 프로그램입니다. 프로그램은 Putty SFTP 클라이언트를 사용하여 실제 `psftp` 파일 전송을 수행합니다. 이 프로그램은 메인프레임 FTP 유틸리티 프로그램과 유사하게 작동하며 사용자 및 암호 인증을 사용합니다.

> **Note**
>
> 퍼블릭 키 인증은 지원되지 않습니다.

메인프레임 FTP JCL을 SFTP를 사용하도록 변환하려면 PGM=FTP를 PGM=M2SFTP로 변경하세요.

주제

지원하는 플랫폼

다음 플랫폼 중 하나에서 M2SFTP를 사용할 수 있습니다.

- AWS 메인프레임 현대화 마이크로 포커스 매니지드
- Micro Focus 런타임(Amazon EC2)
- 모든 종류의 Micro Focus 엔터프라이즈 디벨로퍼(ED) 및 Micro Focus 엔터프라이즈 서버(ES) 제품.

종속성 설치

윈도우에 퍼티 SFTP 클라이언트를 설치하려면

- PuTTY SFTP 클라이언트를 다운로드하고 설치합니다.

Linux에 퍼티 SFTP 클라이언트 설치하기:

- 다음 명령을 실행하여 Putty SFTP 클라이언트를 설치합니다.

```
sudo yum -y install putty
```

메인프레임 현대화 관리를 위한 AWS M2SFTP 구성

마이그레이션된 애플리케이션이 AWS 메인프레임 현대화 관리형에서 실행되는 경우 다음과 같이 M2SFTP 구성을 해야 합니다.

- MFFTP에 적합한 Micro Focus 엔터프라이즈 서버 환경 변수를 설정합니다. 다음은 몇 가지 예입니다.

  - `MFFTP_TEMP_DIR`
  - `MFFTP_SENDEOL`
  - `MFFTP_TIME`
  - `MFFTP_ABEND`

  이러한 변수를 원하는 만큼 적게 또는 많이 설정할 수 있습니다. `ENVAR DD` 명령문을 사용하여 JCL에서 설정할 수 있습니다. 이러한 변수에 대한 자세한 내용은 Micro Focus 설명서의 MFFTP 제어 변수를 참조하세요.

구성을 테스트하려면 샘플 JCL를 참조하세요.

Amazon EC2에서 AWS 메인프레임 현대화 런타임을 위해 M2SFTP 구성 (2.0 포함) AppStream

마이그레이션된 애플리케이션이 Amazon EC2의 AWS 메인프레임 현대화 런타임에서 실행되는 경우 다음과 같이 M2SFTP 구성을 하십시오.

1. 배치 유틸리티의 바이너리 위치를 포함하도록 Micro Focus JES 프로그램 경로를 변경하세요. 여러 경로를 지정해야 하는 경우 Linux에서는 콜론(`:`)을 사용하여 경로를 구분하고 Windows에서는 세미콜론(`;`)을 사용하여 경로를 구분하세요.

   - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
   - Windows(32비트): `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows(64비트): `C:\AWS\M2\MicroFocus\Utilities\64bit`
2. MFFTP에 적합한 Micro Focus 엔터프라이즈 서버 환경 변수를 설정합니다. 다음은 몇 가지 예입니다.

   - `MFFTP_TEMP_DIR`
   - `MFFTP_SENDEOL`

- MFFTP_TIME
- MFFTP_ABEND

이러한 변수를 원하는 만큼 적게 또는 많이 설정할 수 있습니다. ENVAR DD 명령문을 사용하여 JCL에서 설정할 수 있습니다. 이러한 변수에 대한 자세한 내용은 Micro Focus 설명서의 MFFTP 제어 변수를 참조하세요.

구성을 테스트하려면 샘플 JCL를 참조하세요.

샘플 JCL

다음 샘플 JCL 파일 중 하나를 사용할 수 있습니다.

M2SFTP1 .jcl

이 JCL은 M2SFTP를 직접적으로 호출하여 원격 SFTP 서버로 파일을 보내는 방법을 보여줍니다. ENVVAR DD 명령문에 설정된 환경 변수를 확인하세요.

```
//M2SFTP1 JOB 'M2SFTP1',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*-------------------------------------------------------------------**
//* Sample SFTP JCL step to send a file to SFTP server*
//*-------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2SFTP,
//            PARM='127.0.0.1 (EXIT=99 TIMEOUT 300'
//*
//SYSFTPD  DD  *
RECFM FB
LRECL 80
SBSENDEOL CRLF
MBSENDEOL CRLF
TRAILINGBLANKS FALSE
/*
//NETRC    DD  *
machine 127.0.0.1 login sftpuser password sftppass
/*
//SYSPRINT DD  SYSOUT=*
```

```
//OUTPUT   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//INPUT    DD  *
type a
locsite notrailingblanks
cd files
put 'AWS.M2.TXT2PDF1.PDF' AWS.M2.TXT2PDF1.pdf
put 'AWS.M2.CARDDEMO.CARDDATA.PS' AWS.M2.CARDDEMO.CARDDATA.PS1.txt
quit
/*
//ENVVAR   DD *
MFFTP_VERBOSE_OUTPUT=ON
MFFTP_KEEP=N
/*
//*
//
```

M2SFTP2 .jcl

이 JCL은 M2SFTP를 직접적으로 호출하여 원격 SFTP 서버로부터 파일을 수신하는 방법을 보여줍니다. ENVVAR DD 명령문에 설정된 환경 변수를 확인하세요.

```
//M2SFTP2 JOB 'M2SFTP2',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*-------------------------------------------------------------------**
//* Sample SFTP JCL step to receive a file from SFTP server*
//*-------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2SFTP
//*
//SYSPRINT DD  SYSOUT=*
//OUTPUT   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//INPUT    DD  *
open 127.0.0.1
sftpuser
sftppass
cd files
locsite recfm=fb lrecl=150
get AWS.M2.CARDDEMO.CARDDATA.PS.txt +
```

```
'AWS.M2.CARDDEMO.CARDDATA.PS2' (replace
quit
/*
//ENVVAR   DD *
MFFTP_VERBOSE_OUTPUT=ON
MFFTP_KEEP=N
/*
//*
//
```

> **Note**
>
> FTP 자격 증명을 NETRC 파일에 저장하고 권한이 있는 사용자만 액세스할 수 있도록 제한하는 것이 좋습니다.

Putty SFTP(PSFTP) 클라이언트 명령 참조

PSFTP 클라이언트가 모든 FTP 명령을 지원하는 것은 아닙니다. 다음 목록은 PSFTP가 지원하는 모든 명령을 보여줍니다.

Command	설명
!	로컬 명령 실행
bye	SFTP 세션을 끝내세요
cd	원격 작업 디렉터리 변경
chmod	파일 권한 및 모드 변경
close	SFTP 세션을 종료하되 PSFTP를 종료하지 마세요
del	원격 서버의 파일 삭제
dir	원격 파일 나열
exit	SFTP 세션을 마칩니다
get	서버에서 로컬 시스템으로 파일 다운로드

Command	설명
도움	도움 제공
lcd	로컬 작업 디렉터리 변경
lpwd	로컬 작업 디렉터리 인쇄
ls	원격 파일 나열
mget	여러 파일을 한 번에 다운로드
mkdir	원격 서버에 디렉터리 생성
mput	한 번에 여러 파일을 업로드할 수 있습니다
mv	원격 서버의 파일 이동 또는 이름 변경
open	호스트에 연결
put	로컬 시스템에서 서버로 파일 업로드
PWD	원격 작업 디렉터리 인쇄
종료	SFTP 세션을 마칩니다
reget	파일 계속 다운로드
ren	원격 서버의 파일 이동 또는 이름 변경
reput	파일 계속 업로드
rm	원격 서버의 파일 삭제
rmdir	원격 서버의 디렉터리 제거

다음 단계

SFTP를 사용하여 Amazon 심플 스토리지 서비스에 파일을 업로드하고 다운로드하려면 다음 블로그 게시물에 설명된 대로 M2SFTP 과 함께 사용할 수 있습니다. AWS Transfer Family

- AWS SFTP 논리 디렉터리를 사용하여 간단한 데이터 배포 서비스를 구축합니다.
- 사용을 위한 암호 인증을 활성화합니다. AWS Transfer for SFTPAWS Secrets Manager

M2WAIT 배치 유틸리티

M2WAIT는 시간을 초, 분 또는 시간 단위로 지정하여 JCL 스크립트에 대기 기간을 도입할 수 있는 메인프레임 유틸리티 프로그램입니다. 대기하려는 시간을 입력 파라미터로 전달하여 JCL에서 직접 M2WAIT를 직접적으로 호출할 수 있습니다. 내부적으로 M2WAIT 프로그램은 Micro Focus 제공 모듈 C$SLEEP을 호출하여 지정된 시간 동안 대기합니다.

> **Note**
>
> Micro Focus 별칭을 사용하여 JCL 스크립트에 있는 내용을 대체할 수 있습니다. 자세한 내용은 Microsoft 설명서의 JES 별칭을 참조하세요.

주제

- 지원하는 플랫폼
- 메인프레임 현대화를 위한 M2WAIT 구성 (관리형) AWS
- Amazon EC2에서 AWS 메인프레임 현대화 런타임을 위해 M2WAIT 구성 (2.0 포함) AppStream
- 샘플 ACL

지원하는 플랫폼

다음 플랫폼 중 하나에서 M2WAIT를 사용할 수 있습니다.

- AWS 메인프레임 현대화 마이크로 포커스 매니지드
- Micro Focus 런타임(Amazon EC2)
- 모든 종류의 Micro Focus Enterprise Developer(ED) 및 Micro Focus Enterprise Server(ES) 제품.

메인프레임 현대화를 위한 M2WAIT 구성 (관리형) AWS

마이그레이션된 애플리케이션이 AWS 메인프레임 현대화 관리형에서 실행되는 경우 다음과 같이 M2WAIT 구성을 해야 합니다.

- 샘플 ACL와 같이 입력 파라미터를 전달하여 JCL에서 M2WAIT 프로그램을 사용하세요.

Amazon EC2에서 AWS 메인프레임 현대화 런타임을 위해 M2WAIT 구성 (2.0 포함) AppStream

마이그레이션된 애플리케이션이 Amazon EC2의 AWS 메인프레임 현대화 런타임에서 실행되는 경우 다음과 같이 M2WAIT 구성을 하십시오.

1. 배치 유틸리티의 바이너리 위치를 포함하도록 Micro Focus JES 프로그램 경로를 변경하세요. 여러 경로를 지정해야 하는 경우 Linux에서는 콜론(:)을 사용하여 경로를 구분하고 Windows에서는 세미콜론(;)을 사용하여 경로를 구분하세요.
   - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
   - Windows(32비트): `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows(64비트): `C:\AWS\M2\MicroFocus\Utilities\64bit`
2. 샘플 ACL과 같이 입력 파라미터를 전달하여 JCL에서 M2WAIT 프로그램을 사용하세요.

샘플 ACL

설치를 테스트하려면 `M2WAIT1.jcl` 프로그램을 사용할 수 있습니다.

이 샘플 JCL은 M2WAIT를 직접적으로 호출하고 여러 지속 시간을 전달하는 방법을 보여줍니다.

```
//M2WAIT1 JOB 'M2WAIT',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*-------------------------------------------------------------------**
//* Wait for 12 Seconds*
//*-------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2WAIT,PARM='S012'
//SYSOUT DD SYSOUT=*
//*
//*-------------------------------------------------------------------**
//* Wait for 0 Seconds (defaulted to 10 Seconds)*
//*-------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=M2WAIT,PARM='S000'
```

```
//SYSOUT DD SYSOUT=*
//*
//*--------------------------------------------------------------------**
//* Wait for 1 Minute*
//*--------------------------------------------------------------------**
//*
//STEP03 EXEC PGM=M2WAIT,PARM='M001'
//SYSOUT DD SYSOUT=*
//*
//
```

TXT2PDF 배치 유틸리티

TXT2PDF 는 텍스트 파일을 PDF 파일로 변환하는 데 일반적으로 사용되는 메인프레임 유틸리티 프로그램입니다. 이 유틸리티는 TXT2PDF(z/OS 프리웨어)에 동일한 소스 코드를 사용합니다. AWS 메인프레임 현대화 Micro Focus 런타임 환경에서 실행되도록 수정했습니다.

주제

지원하는 플랫폼

다음 플랫폼 중 하나에서 TXT2PDF를 사용할 수 있습니다.

- AWS 메인프레임 현대화 마이크로 포커스 매니지드
- Micro Focus 런타임(Amazon EC2)
- 모든 종류의 Micro Focus Enterprise Developer(ED) 및 Micro Focus Enterprise Server(ES) 제품.

메인프레임 현대화를 위한 TXT2PDF 구성 (관리형) AWS

마이그레이션된 애플리케이션이 AWS 메인프레임 현대화 관리형에서 실행되는 경우 다음과 같이 TXT2PDF 구성을 하십시오.

- `AWS.M2.REXX.EXEC`라는 REXX EXEC 라이브러리를 생성합니다. 이 REXX 모듈을 다운로드하고 라이브러리에 복사하세요.
  - `TXT2PDF.rex` - TXT2PDF z/OS freeware(수정됨)
  - `TXT2PDFD.rex` - TXT2PDF z/OS freeware(수정되지 않음)
  - `TXT2PDFX.rex` - TXT2PDF z/OS freeware(수정됨)
  - `M2GETOS.rex` - To check the OS type(Windows 또는 Linux)

구성을 테스트하려면 샘플 ACL를 참조하세요.

Amazon EC2에서 AWS 메인프레임 현대화 런타임을 위해 TXT2PDF 구성 (2.0 포함) AppStream

마이그레이션된 애플리케이션이 Amazon EC2의 AWS 메인프레임 현대화 런타임에서 실행되는 경우 다음과 같이 TXT2PDF 구성을 하십시오.

1. Micro Focus 환경 변수 `MFREXX_CHARSET`를 적절한 값(예: ASCII 데이터의 경우 "`A`")으로 설정합니다.

   > **Important**
   >
   > 잘못된 값을 입력하면 데이터 변환 문제(EBCDIC에서 ASCII로)가 발생하여 결과 PDF를 읽을 수 없거나 작동하지 않을 수 있습니다. `MFREXX_CHARSET`를 `MF_CHARSET`와 일치하도록 설정하는 것이 좋습니다.

2. 배치 유틸리티의 바이너리 위치를 포함하도록 Micro Focus JES 프로그램 경로를 변경하세요. 여러 경로를 지정해야 하는 경우 Linux에서는 콜론(`:`)을 사용하여 경로를 구분하고 Windows에서는 세미콜론(`;`)을 사용하여 경로를 구분하세요.
   - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
   - Windows(32비트): `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows(64비트): `C:\AWS\M2\MicroFocus\Utilities\64bit`
3. `AWS.M2.REXX.EXEC`라는 REXX EXEC 라이브러리를 생성합니다. 이 REXX 모듈을 다운로드하고 라이브러리에 복사하세요.
   - `TXT2PDF.rex` - TXT2PDF z/OS freeware(수정됨)
   - `TXT2PDFD.rex` - TXT2PDF z/OS freeware(수정되지 않음)

- `TXT2PDFX.rex` - TXT2PDF z/OS freeware(수정됨)
- `M2GETOS.rex` - To check the OS type(Windows 또는 Linux)

구성을 테스트하려면 샘플 ACL를 참조하세요.

샘플 ACL

다음 샘플 JCL 파일 중 하나를 사용할 수 있습니다.

TXT2PDF1.jcl

이 샘플 JCL 파일은 TXT2PDF 변환에 DD 이름을 사용합니다.

```
//TXT2PDF1 JOB 'TXT2PDF1',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*--------------------------------------------------------------------**
//* PRE DELETE*
//*--------------------------------------------------------------------**
//*
//PREDEL  EXEC PGM=IEFBR14
//*
//DD01     DD DSN=AWS.M2.TXT2PDF1.PDF.VB,
//            DISP=(MOD,DELETE,DELETE)
//*
//DD02     DD DSN=AWS.M2.TXT2PDF1.PDF,
//            DISP=(MOD,DELETE,DELETE)
//*
//*--------------------------------------------------------------------**
//* CALL TXT2PDF TO CONVERT FROM TEXT TO PDF (VB)*
//*--------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=IKJEFT1B
//*
//SYSEXEC  DD DISP=SHR,DSN=AWS.M2.REXX.EXEC
//*
//INDD     DD *
1THIS IS THE FIRST LINE ON THE PAGE 1
0THIS IS THE THIRD LINE ON THE PAGE 1
-THIS IS THE   6TH LINE ON THE PAGE 1
THIS IS THE   7TH LINE ON THE PAGE 1
```

```
+_____________________________________ - OVERSTRIKE 7TH LINE
1THIS IS THE FIRST LINE ON THE PAGE 2
0THIS IS THE THIRD LINE ON THE PAGE 2
-THIS IS THE   6TH LINE ON THE PAGE 2
THIS IS THE   7TH LINE ON THE PAGE 2
+_____________________________________ - OVERSTRIKE 7TH LINE
/*
//*
//OUTDD    DD DSN=AWS.M2.TXT2PDF1.PDF.VB,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=256,DSORG=PS,RECFM=VB,BLKSIZE=0)
//*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DDNAME=SYSIN
//*
//SYSIN    DD *
%TXT2PDF BROWSE Y IN DD:INDD +
OUT DD:OUTDD +
CC YES
/*
//*
//*-------------------------------------------------------------------**
//* CONVERT PDF (VB) TO PDF (LSEQ - BYTE STREAM)*
//*-------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=VB2LSEQ
//*
//INFILE   DD DSN=AWS.M2.TXT2PDF1.PDF.VB,DISP=SHR
//*
//OUTFILE  DD DSN=AWS.M2.TXT2PDF1.PDF,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=256,DSORG=PS,RECFM=LSEQ,BLKSIZE=0)
//*
//SYSOUT   DD SYSOUT=*
//*
//
```

TXT2PDF2.jcl

이 샘플 JCL은 TXT2PDF 변환에 DSN 이름을 사용합니다.

```
//TXT2PDF2 JOB 'TXT2PDF2',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
```

```
//* All Rights Reserved.*
//*
//*--------------------------------------------------------------------**
//* PRE DELETE*
//*--------------------------------------------------------------------**
//*
//PREDEL  EXEC PGM=IEFBR14
//*
//DD01     DD DSN=AWS.M2.TXT2PDF2.PDF.VB,
//            DISP=(MOD,DELETE,DELETE)
//*
//DD02     DD DSN=AWS.M2.TXT2PDF2.PDF,
//            DISP=(MOD,DELETE,DELETE)
//*
//*--------------------------------------------------------------------**
//* CALL TXT2PDF TO CONVERT FROM TEXT TO PDF (VB)*
//*--------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=IKJEFT1B
//*
//SYSEXEC  DD DISP=SHR,DSN=AWS.M2.REXX.EXEC
//*
//INDD     DD *
1THIS IS THE FIRST LINE ON THE PAGE 1
0THIS IS THE THIRD LINE ON THE PAGE 1
-THIS IS THE   6TH LINE ON THE PAGE 1
THIS IS THE   7TH LINE ON THE PAGE 1
+_____________________________________ - OVERSTRIKE 7TH LINE
1THIS IS THE FIRST LINE ON THE PAGE 2
0THIS IS THE THIRD LINE ON THE PAGE 2
-THIS IS THE   6TH LINE ON THE PAGE 2
THIS IS THE   7TH LINE ON THE PAGE 2
+_____________________________________ - OVERSTRIKE 7TH LINE
/*
//*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DDNAME=SYSIN
//*
//SYSIN    DD *
%TXT2PDF BROWSE Y IN DD:INDD +
OUT 'AWS.M2.TXT2PDF2.PDF.VB' +
CC YES
/*
//*
```

```
//*--------------------------------------------------------------------**
//* CONVERT PDF (VB) TO PDF (LSEQ - BYTE STREAM)*
//*--------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=VB2LSEQ
//*
//INFILE   DD DSN=AWS.M2.TXT2PDF2.PDF.VB,DISP=SHR
//*
//OUTFILE  DD DSN=AWS.M2.TXT2PDF2.PDF,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=256,DSORG=PS,RECFM=LSEQ,BLKSIZE=0)
//*
//SYSOUT   DD SYSOUT=*
//*
//
```

수정

AWS 메인프레임 현대화 Micro Focus 런타임 환경에서 TXT2PDF 프로그램을 실행하기 위해 다음과 같이 변경했습니다.

- Micro Focus REXX 런타임과의 호환성을 보장하기 위한 소스 코드 변경
- 프로그램이 Windows 및 Linux 운영 체제에서 모두 실행될 수 있도록 변경되었습니다.
- EBCDIC 및 ASCII 런타임을 모두 지원하도록 수정

참조

TXT2PDF 참조 및 소스 코드:

- 텍스트를 PDF로 변환
- z/OS Freeware TCP/IP 및 메일 도구
- TXT2PDF 사용자 참조 가이드

M2DFUTIL 배치 유틸리티

M2DFUTIL은 메인프레임 ADRDSSU 유틸리티에서 제공하는 지원과 마찬가지로 데이터 세트에 백업, 복원, 삭제 및 복사 기능을 제공하는 JCL 유틸리티 프로그램입니다. 이 프로그램은 ADRDSSU의 많은 SYSIN 파라미터를 유지하므로, 최신 유틸리티로 마이그레이션하는 프로세스가 간소화됩니다.

주제

지원하는 플랫폼

다음 플랫폼 중 하나에서 M2DFUTIL을 사용할 수 있습니다.

- Windows의 Micro Focus ES(64비트 및 32비트)
- Linux의 Micro Focus ES(64비트)

플랫폼 요구 사항

M2DFUTIL은 정규 표현식 테스트를 수행하기 위한 스크립트 호출에 따라 달라집니다. Windows에서는 Windows Services for Linux(WSL)를 설치해야 이 스크립트를 실행할 수 있습니다.

향후 지원 예정

현재는 메인프레임 ADRDSSU 유틸리티에서 사용할 수 없지만, 추후 다음과 같은 기능이 제공됩니다.

- M2 관리형
- VSAM
- 파일 이름 변경에 대한 COPY 지원
- RESTORE에 대한 RENAME 지원
- 다중 INCLUDE 및 EXCLUDE
- DSORG, CREDT, EXPDT별 하위 선택을 위한 BY 절
- 대기열에 넣기 실패를 재시도하기 위한 MWAIT 절
- DUMP/RESTORE를 위한 S3 스토리지 지원

자산 위치

이 유틸리티의 로드 모듈 이름은 Linux에서는 `M2DFUTIL.so`, Windows에서는 `M2DFUTIL.dll`입니다. 이 로드 모듈은 다음 위치에서 찾을 수 있습니다.

- Linux: `/opt/aws/m2/microfocus/utilities/64bit`
- Windows(32비트): `C:\AWS\M2\MicroFocus\Utilities\32bit`
- Windows(64비트): `C:\AWS\M2\MicroFocus\Utilities\64bit`

정규 표현식 테스트에 사용되는 스크립트의 이름은 `compare.sh`입니다. 다음 위치에서 이 스크립트를 찾을 수 있습니다.

- Linux: `/opt/aws/m2/microfocus/utilities/scripts`
- Windows(32비트): `C:\AWS\M2\MicroFocus\Utilities\scripts`

Amazon EC2에서 M2DFUTIL 또는 AWS 메인프레임 현대화 런타임 구성 (2.0 포함) AppStream

다음을 사용하여 엔터프라이즈 서버 리전을 구성합니다.

- [ES-환경]에 다음 변수를 추가합니다.
  - `M2DFUTILS_BASE_LOC` - DUMP 출력의 기본 위치
  - `M2DFUTILS_SCRIPTPATH` - 자산 위치에 문서화된 `compare.sh` 스크립트의 위치
  - `M2DFUTILS_VERBOSE` - [VERBOSE 또는 NORMAL]. 이는 `SYSPRINT` 출력의 세부 수준을 제어합니다.
- 로드 모듈 경로가 `JES\Configuration\JES Program Path` 설정에 추가되었는지 확인합니다.
- 유틸리티 디렉터리의 스크립트에 실행 권한이 있는지 확인합니다. Linux 환경에서 `chmod + x <script name>` 명령을 사용하여 실행 권한을 추가할 수 있습니다.

일반 구문

DUMP

현재 카탈로그 위치에서 백업 위치로 파일을 복사하는 기능을 제공합니다. 이 위치는 현재 파일 시스템이어야 합니다.

프로세스

DUMP는 다음을 수행합니다.

1. 대상 위치 디렉터리를 생성합니다.
2. 대상 위치 디렉터리를 PDS 멤버로 카탈로그화합니다.
3. INCLUDE 파라미터를 처리하여 포함할 파일을 결정합니다.
4. EXCLUDE 파라미터를 처리하여 포함된 파일을 선택 취소합니다.
5. 덤프되는 파일을 삭제할지 여부를 결정합니다.
6. 처리해야 할 파일을 대기열에 넣습니다.
7. 파일을 복사합니다.
8. 복사된 파일 카탈로그 DCB 정보를 대상 위치의 사이드 파일로 내보내 향후 RESTORE 작업을 지원합니다.

명령문

```
DUMP
TARGET ( TARGET LOCATION  )    -
INCLUDE ( DSN. )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

필수 파라미터

DUMP의 필수 파라미터는 다음과 같습니다.

- SYSPRINT DD NAME - 추가 로깅 정보를 포함하기 위해 사용합니다.
- TARGET - 대상 위치입니다. 다음 중 하나일 수 있습니다.
  - 덤프 위치의 전체 경로
  - M2DFUTILS_BASE_LOC 변수에 정의된 위치에 생성된 하위 디렉터리 이름
- INCLUDE - 이름이 지정된 단일 DSNAME 또는 유효한 메인프레임 DSN 검색 문자열
- EXCLUDE - 이름이 지정된 단일 DSNAME 또는 유효한 메인프레임 DSN 검색 문자열

선택적 파라미터

- CANCEL - 오류가 발생하면 취소합니다. 처리된 파일은 유지됩니다.
- (기본값) IGNORE - 모든 오류를 무시하고 종료될 때까지 처리합니다.
- DELETE - ENQ 오류가 발생하지 않으면 파일이 삭제되고 카탈로그화되지 않습니다.

DELETE

파일을 대량 삭제하고 카탈로그화를 해제할 수 있는 기능을 제공합니다. 파일은 백업되지 않습니다.

프로세스

DELETE는 다음을 수행합니다.

1. INCLUDE 파라미터를 처리하여 포함할 파일을 결정합니다.
2. EXCLUDE 파라미터를 처리하여 포함된 파일을 선택 취소합니다.
3. 처리해야 할 파일을 대기열에 넣습니다. 처리를 OLD, DELETE, KEEP으로 설정합니다.

명령문

```
DELETE
INCLUDE ( DSN )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

필수 파라미터

DELETE의 필수 파라미터는 다음과 같습니다.

- SYSPRINT DD NAME - 추가 로깅 정보를 포함하기 위해 사용합니다.
- INCLUDE - 이름이 지정된 단일 DSNAME 또는 유효한 메인프레임 DSN 검색 문자열
- EXCLUDE - 이름이 지정된 단일 DSNAME 또는 유효한 메인프레임 DSN 검색 문자열

선택적 파라미터

- CANCEL - 오류가 발생하면 취소합니다. 처리된 파일은 유지됩니다.
- (기본값) IGNORE - 모든 오류를 무시하고 종료될 때까지 처리합니다.

RESTORE

DUMP를 사용하여 이전에 백업한 파일을 복원하는 기능을 제공합니다. RENAME을 사용하여 복원된 DSNAME을 변경하지 않는 한, 파일은 카탈로그화된 원래 위치에 복원됩니다.

프로세스

RESTORE는 다음을 수행합니다.

1. 소스 위치 디렉터리의 유효성을 검사합니다.
2. 카탈로그 내보내기 파일을 처리하여 포함할 파일을 결정합니다.
3. EXCLUDE 파라미터를 처리하여 포함된 파일을 선택 취소합니다.
4. 처리해야 할 파일을 대기열에 넣습니다.
5. 내보내기 정보를 기반으로 카탈로그화되지 않은 카탈로그 파일.
6. 파일이 이미 카탈로그화되어 있고 내보내기 카탈로그 정보가 동일한 경우 REPLACE 옵션이 설정되었다면 RESTORE는 카탈로그화된 데이터 세트를 대체합니다.

명령문

```
RESTORE
SOURCE ( TARGET LOCATION )
INCLUDE ( DSN )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ REPLACE]
```

필수 파라미터

RESTORE의 필수 파라미터는 다음과 같습니다.

- SYSPRINT DD NAME - 추가 로깅 정보를 포함하기 위해 사용합니다.
- SOURCE - 소스 위치입니다. 다음 중 하나일 수 있습니다.
  - 덤프 위치의 전체 경로
  - M2DFUTILS_BASE_LOC 변수에 정의된 위치에 생성된 하위 디렉터리 이름
- INCLUDE - 이름이 지정된 단일 DSNAME 또는 유효한 메인프레임 DSN 검색 문자열
- EXCLUDE - 이름이 지정된 단일 DSNAME 또는 유효한 메인프레임 DSN 검색 문자열

선택적 파라미터

- CANCEL - 오류가 있으면 취소합니다. 처리된 파일은 유지됩니다.
- (기본값) IGNORE - 모든 오류를 무시하고 종료될 때까지 처리합니다.
- REPLACE - 복원 중인 파일이 이미 카탈로그화되어 있고 카탈로그 레코드가 동일한 경우 카탈로그화된 파일을 교체합니다.

샘플 JCL

DUMP 작업

이 작업을 수행하면 TESTDUMP라는 하위 디렉터리가 생성됩니다. 이는 M2DFUTILS_BASE_LOC 변수로 지정된 기본 백업 위치입니다. M2DFUTILS.TESTDUMP라는 이름의 백업용 PDS 라이브러리를 생성합니다. 내보낸 카탈로그 데이터는 CATDUMP.DAT라는 백업 디렉터리의 행 순차 파일에 저장됩니다. 선택한 모든 파일이 이 백업 디렉터리에 복사됩니다.

```
//M2DFDMP JOB 'M2DFDMP',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
//SYSPRINT DD DSN=TESTDUMP.SYSPRINT,
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(RECFM=LSEQ,LRECL=256)
//SYSIN    DD *
DUMP TARGET(TESTDUMP)               -
     INCLUDE(TEST.FB.FILE*.ABC)     -
 CANCEL
/*
//
```

DELETE 작업

이 작업은 카탈로그에서 INCLUDE 파라미터와 일치하는 모든 파일을 삭제합니다.

```
/M2DFDEL JOB 'M2DFDEL',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
//SYSPRINT DD DSN=TESTDEL.SYSPRINT,
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(RECFM=LSEQ,LRECL=256)
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
  DELETE                            -
```

```
     INCLUDE(TEST.FB.FILE*.ABC)        -
 CANCEL
/*
 //
```

RESTORE 작업

이 작업은 INCLUDE 파라미터와 일치하는 파일을 TESTDUMP 백업 위치에서 복원합니다. 카탈로그화된 파일이 CATDUMP 내보내기에 있는 파일과 동일하고 REPLACE 옵션이 지정된 경우 카탈로그화된 파일이 교체됩니다.

```
//M2DFREST JOB 'M2DFREST',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
////SYSPRINT DD DSN=TESTREST.SYSPRINT,
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(RECFM=LSEQ,LRECL=256)
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
RESTORE SOURCE(TESTDUMP)               -
     INCLUDE(TEST.FB.FILE*.ABC)        -
 IGNORE
 REPLACE
/*
//
```

M2RUNCMD 배치 유틸리티

배치 유틸리티 프로그램인 M2RUNCMD를 사용하면 Micro Focus 명령, 스크립트 및 시스템 호출을 터미널이나 명령 프롬프트에서 실행하는 대신 JCL에서 직접 실행할 수 있습니다. 명령의 출력은 배치 작업의 스풀 로그에 기록됩니다.

주제

- 지원하는 플랫폼
- Amazon EC2에서 AWS 메인프레임 현대화 런타임을 위해 M2RUNCMD 구성 (2.0 포함) AppStream
- 샘플 JCL

지원하는 플랫폼

다음 플랫폼에서 M2RUNCMD를 사용할 수 있습니다.

- Micro Focus 런타임(Amazon EC2)
- 모든 종류의 Micro Focus Enterprise Developer(ED) 및 Micro Focus Enterprise Server(ES) 제품.

Amazon EC2에서 AWS 메인프레임 현대화 런타임을 위해 M2RUNCMD 구성 (2.0 포함) AppStream

마이그레이션된 애플리케이션이 Amazon EC2의 AWS 메인프레임 현대화 런타임에서 실행되는 경우 다음과 같이 M2RUNCMD 구성을 하십시오.

- 배치 유틸리티의 바이너리 위치를 포함하도록 Micro Focus JES 프로그램 경로를 변경하세요. 여러 경로를 지정해야 하는 경우 Linux에서는 콜론(:)을 사용하여 경로를 구분하고 Windows에서는 세미콜론(;)을 사용하여 경로를 구분합니다.
  - Linux: `/opt/aws/m2/microfocus/utilities/64bit`
  - Windows(32비트): `C:\AWS\M2\MicroFocus\Utilities\32bit`
  - Windows(64비트): `C:\AWS\M2\MicroFocus\Utilities\64bit`

샘플 JCL

다음 샘플 JCL 중 하나를 사용하여 설치를 테스트할 수 있습니다.

RUNSCRL1.jcl

이 샘플 JCL은 스크립트를 생성하여 실행합니다. 첫 번째 단계는 `/tmp/TEST_SCRIPT.sh`라는 스크립트를 생성하고 SYSUT1 인스트림 데이터의 내용을 포함합니다. 두 번째 단계에서는 실행 권한을 설정하고 첫 번째 단계에서 만든 스크립트를 실행합니다. 두 번째 단계만 수행하도록 선택하여 기존 Micro Focus 및 시스템 명령을 실행할 수도 있습니다.

```
//RUNSCRL1 JOB 'RUN SCRIPT',CLASS=A,MSGCLASS=X,TIME=1440
//*
//*
//*--------------------------------------------------------------------*
//*  CREATE SCRIPT (LINUX)
//*--------------------------------------------------------------------*
//*
//STEP0010 EXEC PGM=IEBGENER
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

```
//*
//SYSUT1   DD *
#!/bin/bash

set -x

## ECHO PATH ENVIRONMNET VARIABLE
echo $PATH

## CLOSE/DISABLE VSAM FILE
casfile -r$ES_SERVER -oc  -ed -dACCTFIL

## OPEN/ENABLE VSAM FILE
casfile -r$ES_SERVER -ooi -ee -dACCTFIL

exit $?
/*
//SYSUT2   DD DSN=&&TEMP,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(RECFM=LSEQ,LRECL=300,DSORG=PS,BLKSIZE=0)
//*MFE: %PCDSN='/tmp/TEST_SCRIPT.sh'
//*
//*--------------------------------------------------------------------*
//*   RUN SCRIPT (LINUX)                                               *
//*--------------------------------------------------------------------*
//*
//STEP0020 EXEC PGM=RUNCMD
//*
//SYSOUT  DD  SYSOUT=*
//*
//SYSIN   DD *
*RUN SCRIPT
 sh /tmp/TEST_SCRIPT.sh
/*
//
```

SYSOUT

실행된 명령 또는 스크립트의 출력이 SYSOUT 로그에 기록됩니다. 수행된 각 명령에 대해 명령, 출력 및 반환 코드가 표시됩니다.

```
************ CMD Start ************
```

```
CMD_STR: sh /tmp/TEST_SCRIPT.sh

CMD_OUT:


+ echo /opt/microfocus/EnterpriseServer/bin:/sbin:/bin:/usr/sbin:/usr/bin
/opt/microfocus/EnterpriseServer/bin:/sbin:/bin:/usr/sbin:/usr/bin
+ casfile -rMYDEV -oc -ed -dACCTFIL

-Return Code:   0

Highest return code:    0

+ casfile -rMYDEV -ooi -ee -dACCTFIL

-Return Code:   8

Highest return code:    8

+ exit 8


CMD_RC=8


************  CMD End  ************
```

RUNCMDL1.jcl

이 샘플 JCL은 RUNCMD를 사용하여 여러 명령을 실행합니다.

```
//RUNCMDL1 JOB 'RUN CMD',CLASS=A,MSGCLASS=X,TIME=1440
//*
//*
//*--------------------------------------------------------------------*
//*   RUN SYSTEM COMMANDS                                              *
//*--------------------------------------------------------------------*
//*
//STEP0001 EXEC PGM=RUNCMD
//*
//SYSOUT  DD  SYSOUT=*
//*
//SYSIN   DD *
*LIST DIRECTORY
```

```
 ls
*ECHO PATH ENVIRONMNET VARIABLE
 echo $PATH
/*
//
```

AWS Precise를 사용한 메인프레임 현대화 데이터 복제

AWS 메인프레임 현대화는 다양한 Amazon 머신 이미지 (AMI) 를 제공합니다. 이러한 AMI를 사용하면 Amazon EC2 인스턴스를 신속하게 프로비저닝할 수 있어 메인프레임 시스템에서 Precise를 사용할 때까지 데이터를 복제할 수 있는 맞춤형 환경을 구축할 수 있습니다. AWS 이 안내서는 이러한 AMI에 액세스하고 사용하는 데 필요한 단계를 제공합니다.

필수 조건

- Amazon EC2 인스턴스를 생성할 수 있는 AWS 계정에 대한 관리자 액세스 권한이 있는지 확인하십시오.
- Amazon EC2 인스턴스를 생성하려는 지역에서 AWS 메인프레임 현대화 서비스를 사용할 수 있는지 확인하십시오. 리전별 사용 가능한 AWS 서비스 목록을 참조하세요.
- Amazon EC2 인스턴스가 생성될 Amazon Virtual Private Cloud(Amazon VPC)를 식별하세요.
- Amazon VPC에서 Amazon EC2 인스턴스를 생성할 때는 연결된 라우팅 테이블에 인터넷 게이트웨이 또는 NAT 게이트웨이가 있는지 확인합니다.

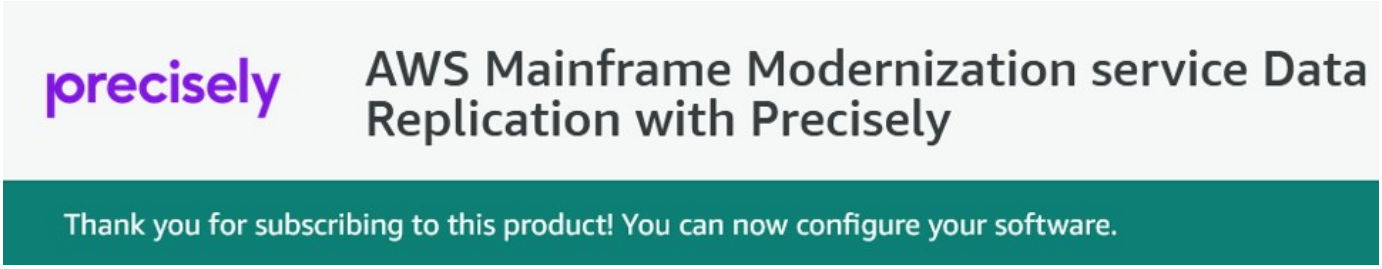
> **Note**
>
> 데이터를 성공적으로 복제하려면 AWS EC2 인스턴스가 AWS Marketplace에 대한 통신 액세스 권한을 가지고 있어야 합니다. AWS Marketplace와의 연결 문제가 있는 경우 복제 프로세스가 실패합니다.

Amazon Machine Image 구독

AWS Marketplace 제품을 구독하면 제품의 AMI에서 인스턴스를 시작할 수 있습니다.

1. AWS Management Console 로그인하고 https://console.aws.amazon.com/marketplace 에서 콘솔을 엽니다. AWS Marketplace
2. 구독 관리를 선택합니다.
3. https://aws.amazon.com/marketplace/pp/prodview-en3xrbgzbs3dk 링크를 복사하여 브라우저 주소 표시줄에 붙여 넣습니다.
4. 구독 계속을 선택합니다.

5. 이용 약관에 동의하는 경우 약관 동의를 선택합니다. 이 프로세스에는 몇 분이 걸릴 수 있습니다.
6. 아래와 같이 감사 메시지가 나타날 때까지 기다립니다. 이 메시지는 제품을 성공적으로 구독했음을 확인하는 메시지입니다.



7. 왼쪽 탐색 창에서 구독 관리를 선택합니다. 이 보기에는 구독한 모든 구독이 표시됩니다.

Precise를 사용하여 AWS 메인프레임 현대화 데이터 복제를 시작하십시오.

1. https://console.aws.amazon.com/marketplace 에서 AWS Marketplace 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 구독 관리를 선택합니다.
3. 시작하려는 AMI를 찾은 다음 새 인스턴스 실행을 선택합니다.
4. 리전에서 허용 목록에 있는 리전을 선택합니다.
5. 계속해서 EC2를 통해 시작을 선택하세요. 이 작업을 수행하면 Amazon EC2 콘솔로 이동합니다.
6. 서버 이름에 DNS 이름을 입력합니다.
7. 프로젝트 성능 및 비용 요구 사항에 맞는 인스턴스 유형을 선택합니다. 인스턴스 크기의 권장 시작점은 `c5.2xLarge`입니다.
8. 기존 키 페어를 선택하거나 새 페어를 생성 후 저장합니다. 키 페어에 대한 자세한 내용은 Amazon EC2 Linux 인스턴스용 사용 설명서에서 Amazon EC2 키 페어 및 Linux 인스턴스를 참조하세요.
9. 네트워크 설정을 편집하고 허용 목록에 있는 VPC와 적절한 서브넷을 선택합니다.
10. 기존 보안 그룹을 선택하거나 새 보안 그룹을 생성합니다. Precisely 서버 EC2 인스턴스를 사용한 데이터 복제의 경우 SSH 액세스 허용(기본적으로 포트 22에서) 외에도 TCP 트래픽을 기본 포트 2626으로 허용하는 것이 일반적입니다.
11. Amazon EC2 인스턴스의 스토리지를 구성합니다.
12. 요약을 검토하고 인스턴스 실행을 선택합니다. 성공적으로 실행되려면 인스턴스 유형이 유효해야 합니다. 시작에 실패할 경우 인스턴스 구성 편집을 선택하고 다른 인스턴스 유형을 선택합니다.
13. 성공 메시지가 표시되면 인스턴스에 연결을 선택합니다.

14. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
15. 왼쪽 탐색 창의 인스턴스 메뉴에서 인스턴스를 선택합니다.
16. 기본 창에서 인스턴스의 상태를 확인합니다.

IAM 정책 생성

AWS Marketplace 목록을 통해 배포한 AWS 메인프레임 현대화 EC2 인스턴스를 성공적으로 운영하려면 IAM 역할 및 정책을 구성해야 합니다. 특별히 맞춤화된 이 IAM 설정은 선택 사항이 아니라 Amazon EC2 인스턴스가 서비스와 상호 작용할 수 있도록 승인합니다. AWS Marketplace AWS 메인프레임 현대화는 IAM 역할 및 정책을 통해 정확한 요금 청구를 위해 필수적인 사용 데이터를 정확하게 기록할 수 있습니다. 이 구성을 구현하지 않으면 데이터 복제 시도가 실패하고 운영이 중단될 수 있습니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 정책을 선택합니다.
3. 정책을 처음으로 선택하는 경우 관리형 정책 소개 페이지가 나타납니다. 시작하기를 선택합니다.
4. 페이지 상단에서 정책 생성을 선택합니다.
5. 정책 편집기 섹션에서 JSON 옵션을 선택합니다.
6. 다음 JSON 정책을 입력합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": ["aws-marketplace:MeterUsage"],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

IAM 역할 생성

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할(Roles)을 선택한 후 역할 생성(Create role)을 선택합니다.
3. 신뢰할 수 있는 엔터티 유형(Trusted entity type) 섹션에서 AWS 서비스를 선택합니다.

4. 사용 사례 섹션의 서비스 또는 사용 사례에서 Amazon EC2를 선택합니다.
5. 다음을 선택합니다.
6. 정책 목록의 유형별 필터 드롭다운에서 고객 관리형을 선택하고 생성한 정책의 이름을 입력합니다. 정책 이름 옆에 있는 확인란을 선택합니다.
7. 다음을 선택합니다.
8. 역할의 이름과 설명(선택 사항)을 입력합니다.
9. 신뢰 정책 및 권한을 검토한 다음 역할 생성을 선택합니다.

Amazon EC2 인스턴스에 IAM 역할 연결

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 인스턴스를 선택합니다.
3. Amazon EC2 인스턴스를 선택합니다.
4. 작업 메뉴에서 보안을 선택한 다음 IAM 역할 수정을 선택합니다.
5. 인스턴스에 연결할 역할을 선택한 다음 IAM 역할 업데이트를 선택합니다.

Charon 통합

Charon-SSP 소개

1987년 Sun Microsystems는 32비트 RISC 프로세서인 SPARC V7 프로세서를 출시했습니다. 이후 오리지널 SPARC V7의 개정판인 SPARC V8이 1990년에 출시되었으며, 가장 눈에 띄는 변화로 하드웨어 나누기 및 곱하기 명령이 포함되었습니다. SPARC V8 프로세서는 SPARCstation 5, 10, 20과 같은 여러 서버와 워크스테이션의 기반을 형성했습니다. 1993년에는 SPARC V8에 이어 64비트 SPARC V9 프로세서가 출시되었습니다. 이 역시 Enterprise 250 및 450과 같은 여러 서버와 워크스테이션의 기반이 되었습니다.

하드웨어 노후화와 예비 부품 또는 리퍼비시 부품 부족으로 인해, 구형 SPARC 기반 워크스테이션과 서버용으로 개발된 소프트웨어 및 시스템을 유지 관리하기가 더 어려워졌습니다. 특정 end-of-life SPARC 기반 시스템에 대한 지속적인 수요를 충족하기 위해 Stromasys S.A. 는 Charon-SSP SPARC 에뮬레이터 제품 라인을 개발했습니다. 다음 제품은 지정된 기본 하드웨어 SPARC 시스템을 대체하는 소프트웨어 기반 가상 머신 대체품입니다. 에뮬레이션된 하드웨어 제품군에 대한 일반적인 개요는 다음과 같습니다.

Charon-SSP/4M은 다음과 같은 SPARC 하드웨어를 에뮬레이션합니다.

- Sun-4m 제품군(Sun SPARCstation 20으로 대표됨): 원래는 멀티프로세서 Sun-4 변형이었으며, SPARCServer 600MP 시리즈에 도입된 MBus 프로세서 모듈 버스를 기반으로 했습니다. Sun-4m 아키텍처는 나중에 SPARC V8 아키텍처 프로세서를 활용하는 SPAR Cstation 5와 같은 비MBus 단일 프로세서 시스템도 포함했습니다. SunOS 4.1.2부터 시작해서 Solaris 2.1에서 Solaris 9까지 지원됩니다. SPARCServer 600MP 지원은 Solaris 2.5.1 이후에 중단되었습니다.

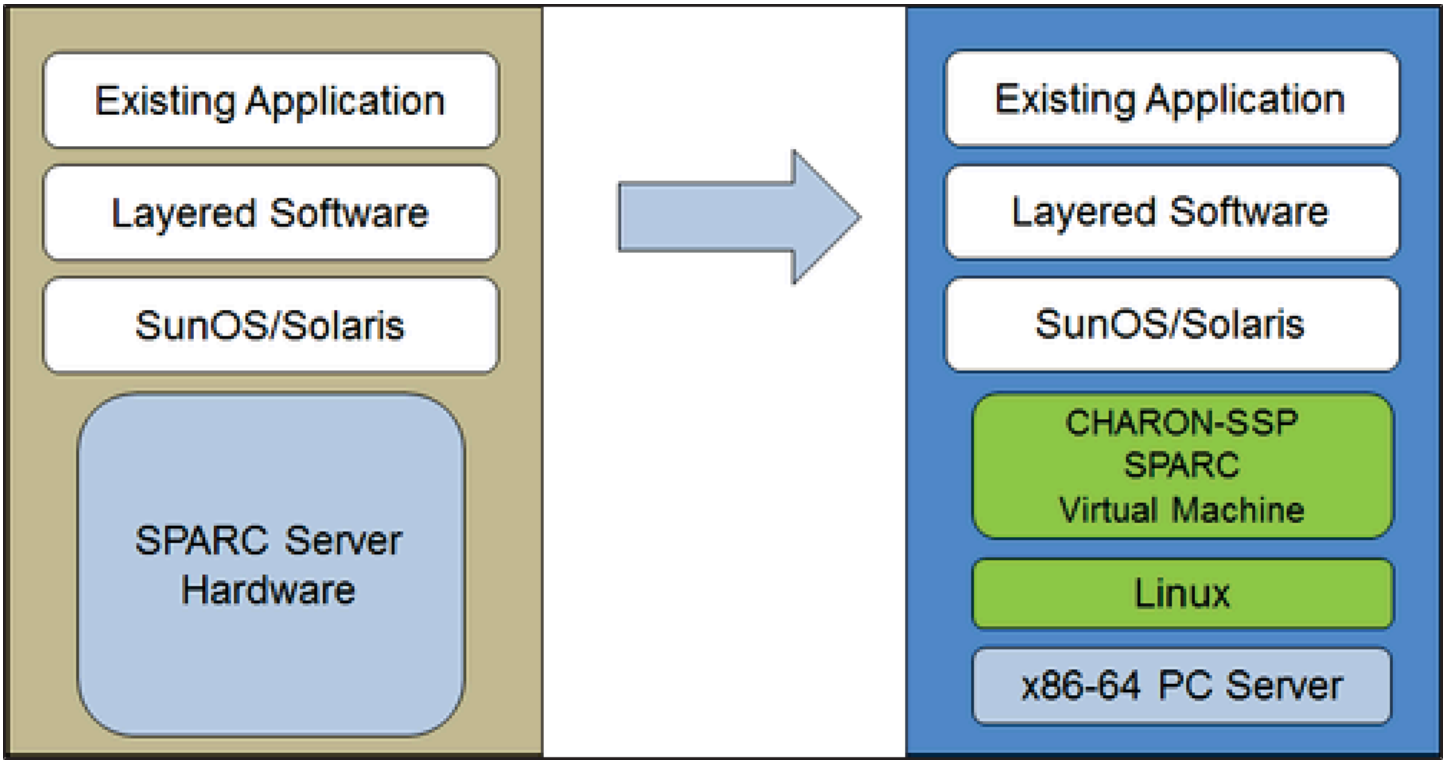
Charon-SSP/4U(+)는 다음과 같은 SPARC 하드웨어를 에뮬레이션합니다.

- Sun-4u 제품군(Sun Enterprise 450으로 대표됨): (UltraSPARC의 경우 U) - 이 변형에는 Sun Ultra 시리즈에서 처음 사용된 64비트 SPARC V9 프로세서 아키텍처와 UPA 프로세서 상호 연결이 도입되었습니다. 버전 2.5.1부터 시작하는 32비트 버전의 Solaris에서 지원됩니다. Sun-4u용 최초의 64비트 Solaris 릴리스는 Solaris 7이었습니다. UltraSPARC I 지원은 Solaris 9 이후 중단되었습니다. Solaris 10은 UltraSP ARC II부터 UltraSPARC IV에 이르는 Sun-4u 구현을 지원합니다.

Charon-SSP/4V(+)는 다음과 같은 SPARC 하드웨어를 에뮬레이션합니다.

- Sun-4v 제품군(SPARC T2 및 T4로 대표됨): 이 변형은 Ultra SPARC T1 멀티코어 프로세서에 도입된 Sun-4u에 하이퍼바이저 프로세서 가상화를 추가했습니다. 일부 하드웨어는 릴리스 3/05 HW2부터 Solaris 버전 10에서 지원되었습니다. Charon-SSP로 에뮬레이션된 하드웨어를 포함한 대부분의 모델에는 최신 버전의 Solaris 10이 필요합니다. 여러 Solaris 11 버전도 지원됩니다.

다음 이미지는 물리적 하드웨어를 에뮬레이터로 마이그레이션하는 기본 개념을 보여줍니다.



Charon-SSP 가상 시스템을 사용하면 Sun 및 Oracle SPARC 기반 컴퓨터 사용자는 원래 시스템 구성을 거의 또는 전혀 변경하지 않고도 기본 하드웨어를 교체할 수 있습니다. 즉, 다른 플랫폼으로 전환하거나 포트할 필요 없이 애플리케이션과 데이터를 계속 실행할 수 있습니다. Charon-SSP 소프트웨어는 상용 Intel 64비트 시스템에서 실행되므로, 투자한 대상을 계속해서 활용할 수 있습니다.

Charon-SSP/4U+는 Charon-SSP/4U와 동일한 가상 SPARC 플랫폼을 지원하며 Charon-SSP/4V+는 Charon-SSP/4V와 동일한 가상 SPARC 플랫폼을 지원합니다. 그러나 4U+ 및 4V+ 버전은 최신 CPU에서 Intel의 VTx/EPT 및 AMD의 AMD-v/NPT 하드웨어 지원 가상화 기술을 활용하여 더 나은 가상 CPU 성능을 제공합니다. Charon-SSP/4U+ 및 Charon-SSP/4V+에는 VT-x/EPT 또는 AMD-v/NPT를 지원하는 CPU가 필요하며 전용 호스트 시스템에 설치해야 합니다. VM(예: VMware)에서 이러한 제품 변형을 실행하는 것은 지원되지 않습니다.

> **Note**
>
> 클라우드 환경에서 Charon-SSP/4U+ 또는 4V+를 실행할 계획이라면 Stromasys 또는 Stromasys VAR에 문의하여 요구 사항에 대해 논의하세요.

지원되는 게스트 운영 체제

Charon-SSP/4M 가상 머신은 다음과 같은 게스트 운영 체제 릴리스를 지원합니다.

- SunOS 4.1.3~4.1.4
- Solaris 2.3~Solaris 9

Charon-SSP/4U(+) 가상 머신은 다음과 같은 게스트 운영 체제 릴리스를 지원합니다.

- Solaris 2.5.1~Solaris 10

Charon-SSP/4V(+) 가상 머신은 다음과 같은 게스트 운영 체제 릴리스를 지원합니다.

- Solaris 10(08/07 업데이트 4부터 시작), Solaris 11.1~Solaris 11.4

Charon-SSP/4V(+)의 경우 다음 사항을 참고하세요.

- 에뮬레이션된 SPARC T4의 경우 지원되는 Solaris 10 버전은 Oracle Solaris 10 1/13, Oracle Solaris 10 8/11 및 Solaris 10 9/10 또는 Oracle Solaris 10 8/11 패치 세트를 사용하는 Solaris 10 10/09입니다.
- 에뮬레이션된 SPARC T4 모델은 에뮬레이터에서 Solaris 11.4를 실행하기 위한 사전 조건입니다.
- Solaris 커널 영역은 지원되지 않습니다.

Charon-SSP 클라우드 인스턴스 사전 조건

인스턴스 유형이나 형태를 선택하여 클라우드의 Charon-SSP 호스트 인스턴스에 사용할 가상 하드웨어를 선택합니다. 인스턴스 유형이나 모양의 선택에 따라 Charon-SSP 가상 호스트 하드웨어의 하드웨어 특성(예: 가상 Charon 호스트 시스템의 CPU 코어 수 및 메모리 양)이 결정됩니다.

> **Note**
>
> Charon-SSP 마켓플레이스 이미지를 사용하여 인스턴스를 시작하면 모든 Linux 호스트 운영 체제 요구 사항이 충족됩니다.

최소 하드웨어 요구 사항은 다음과 같습니다.

크기 조정 가이드라인과 관련된 중요 사항:

- 아래의 크기 조정 지침(특히 호스트 CPU 코어 수 및 호스트 메모리와 관련된 지침)에는 최소 요구 사항이 나와 있습니다. 모든 배포 상황을 검토하고 필요에 따라 실제 호스트 크기를 조정해야 합니다. 예를 들어, 게스트 애플리케이션에서 높은 I/O 부하가 발생하는 경우 I/O에 사용할 수 있는 CPU 코어 수를 늘려야 합니다. 그리고 에뮬레이션된 CPU가 많은 시스템은 일반적으로 I/O 로드를 높일 수 있으니, I/O에 사용 가능한 CPU 코어 수를 늘려야 할 수 있습니다. 하이퍼스레딩 환경에서 최상의 성능을 발휘하려면 CPU 코어 수(실제/물리적 CPU)가 활성 에뮬레이터의 CPU 요구 사항을 충족하기에 충분해야 합니다. 그래야 워크로드가 많은 스레드가 하나의 물리적 CPU 코어를 공유하는 것을 피할 수 있습니다.
- 에뮬레이션된 CPU의 CPU 코어 할당과 I/O 처리를 위한 CPU 코어는 구성에 따라 결정됩니다. 이에 대한 자세한 내용 및 I/O 처리를 위한 CPU 코어의 기본 할당에 대한 자세한 내용은 일반 Charon-SSP 사용 설명서의 CPU 구성을 참조하세요.

> **⚠ 중요한 일반 정보**
>
> - 한 클라우드 인스턴스에서 다른 클라우드 인스턴스로 에뮬레이터 데이터를 빠르게 전송하려면, 모든 관련 에뮬레이터 데이터를 기존 인스턴스에서 쉽게 분리하여 새 인스턴스에 연결할 수 있는 별도의 디스크 볼륨에 저장하는 것이 좋습니다.
> - 인스턴스의 크기를 처음부터 올바르게 지정해야 합니다(아래 최소 요구 사항 확인). Charon-SSP AL의 Charon-SSP 라이선스는 인스턴스가 처음 시작될 때 생성됩니다. 나중에 다른 인스턴스 크기/유형으로 변경하여 CPU 코어 수가 달라지면 라이선스가 무효화되어 Charon 인스턴스가 시작되지 않습니다(새 인스턴스 필요). AutoVE 모드에서 Charon-SSP AL 인스턴스를 사용할 계획이라면 처음 시작하기 전에 AutoVE 서버 정보를 포함해야 합니다. 그렇지 않으면 퍼블릭 라이선스 서버가 사용됩니다. Charon-SSP VE의 라이선스는 라이선스 서버에서 캡처한 지문을 기반으로 생성됩니다. 라이선스 서버를 에뮬레이터 호스트에서 직접 실행하고 나중에 에뮬레이터 호스트에서 CPU 코어 수 변경 등을 요구하는 경우 라이선스는 무효화됩니다(새 라이선스 및 새 인스턴스 필요).

인스턴스 사전 조건

일반 CPU 요구 사항: Charon-SSP는 최신 x86-64 아키텍처 프로세서 기반 Amazon EC2 인스턴스를 지원합니다.

Charon-SSP의 최소 요구 사항:

- 호스트 시스템 CPU 코어의 최소 개수:
  - 호스트 운영 체제용 CPU 코어 하나 이상
  - 에뮬레이션된 각 SPARC 시스템의 경우:
    - 인스턴스의 에뮬레이션된 CPU 하나당 CPU 코어 1개
    - I/O 처리를 위한 추가 CPU 코어 1개 이상(서버 JIT 최적화를 사용하는 경우 최소 2개). 구성 옵션은 위에서 언급한 CPU 구성 섹션을 참조하세요. 기본적으로 Charon은 Charon 호스트에 표시되는 CPU 수의 1/3(최소 1, 반올림)을 I/O 처리에 할당합니다.
- 최소 메모리 요구 사항:
  - Linux 호스트 운영 체제용 RAM이 4GB 이상이어야 합니다. 실제 요구 사항은 더 높을 수 있으며, Linux 호스트에서 실행되는 비에뮬레이터 서비스의 요구 사항에 따라 달라집니다. Linux 호스트에 2GB 이상의 RAM을 권장하는 이전 권장 사항은 많은 시스템에서 여전히 유효하지만, Linux 운영 체제 및 애플리케이션의 요구 사항이 증가하면서 새로 설치해야 하는 업데이트된 권장 사항이 생겼습니다. 추가적으로 다음이 적용됩니다.
  - 에뮬레이션된 각 SPARC 시스템의 경우:
    - 에뮬레이션된 인스턴스의 구성 메모리
    - DIT 최적화, 에뮬레이터 요구 사항, 런타임 버퍼, SMP 및 그래픽 에뮬레이션을 지원하는 2GB의 RAM(서버 JIT를 사용하는 경우 6GB의 RAM)
- 최신 x86-64 CPU에서 하이퍼스레딩을 사용할 경우 물리적 CPU 코어 하나에서 스레드 2개를 실행하여 호스트 운영 체제에 2개의 논리적 CPU를 제공할 수 있습니다. 가능하면 Charon-SSP 호스트에서 하이퍼스레딩을 비활성화하세요. 하지만 이는 VMware 및 클라우드 환경에서 불가능한 경우가 많거나 하이퍼스레딩의 사용 여부가 불분명합니다. Charon-SSP 하이퍼스레딩 옵션을 사용하면 Charon-SSP가 이러한 환경에 적응할 수 있습니다. 자세한 구성 정보는 위에서 언급한 일반 Charon-SSP 사용 설명서의 CPU 구성 섹션을 참조하세요. 참고: 최상의 성능을 보장하려면 Charon-SSP 스레드가 물리적 CPU 코어를 공유하지 않아야 합니다. 구성된 에뮬레이터의 요구 사항을 충족하기에 충분한 물리적 코어를 호스트 시스템에서 사용할 수 있어야 합니다.
- 고객 요구 사항에 따라 하나 이상의 네트워크 인터페이스가 필요합니다.
- Charon-SSP/4U+ 및 Charon-SSP/4V+는 Intel VT-x/EPT 또는 AMD-v/NPT(베어메탈 인스턴스)를 지원하는 물리적 하드웨어에서 실행되어야 하므로, 모든 클라우드 환경에서 실행할 수는 없습니다. 해

당 하드웨어의 가용성에 대해서는 클라우드 제공업체의 설명서를 참조하세요. 또한, 다음 사항에 유의하세요.

- Charon-SSP/4U+ 및 Charon-SSP/4V+는 Stromasys에서 지원하는 Linux 커널을 사용할 때만 지원됩니다.
- 이러한 유형의 에뮬레이션된 SPARC 하드웨어가 필요한 경우 Stromasys 또는 Stromasys VAR에 문의하여 요구 사항을 자세히 논의하세요.

Charon용 클라우드 인스턴스 생성 및 구성 (새 GUI) AWS

이 섹션에는 2022년 AWS Management Console 봄의 상황이 반영되어 있습니다. 여전히 이전 콘솔을 사용하는 경우 AWS Charon-SSP 시작 안내서의 부록을 참조하십시오.

일반적인 사전 요구 사항

이 설명은 AWS에서의 Linux 인스턴스 기본 설정을 보여줍니다. 여기에는 특정 사전 조건이 나열되어 있지 않습니다. 하지만 사용 사례에 따라 다음 사전 조건을 고려하세요.

- 아마존 계정 및 AWS Marketplace 구독
  - 에서 AWS Linux 인스턴스를 설정하려면 관리자 액세스 권한이 있는 AWS 계정이 필요합니다.
  - 인스턴스를 시작하려는 AWS 지역을 식별하십시오. 사용하려는 AWS 서비스를 해당 리전에서 사용할 수 있는지 확인하세요. AWS 리전별 서비스를 참조하세요.
  - 인스턴스를 시작할 VPC와 서브넷을 확인합니다.
  - 인스턴스에 인터넷 액세스가 필요한 경우 VPC와 연결된 라우팅 테이블에 인터넷 게이트웨이가 있는지 확인하세요. 인스턴스에 온프레미스 네트워크에 대한 VPN 액세스가 필요한 경우 VPN 게이트웨이를 사용할 수 있는지 확인합니다. VPC와 서브넷의 정확한 구성은 네트워크 설계 및 애플리케이션 요구 사항에 따라 달라집니다.
  - 특정 AWS Marketplace 서비스를 구독하려면 에서 AWS Marketplace 구독을 선택한 다음 구독 관리를 선택합니다. AWS Management Console
  - 사용하려는 서비스를 검색하고 구독합니다. 구독을 완료하면 구독 관리 섹션에 구독이 나타납니다. 여기에서 새 인스턴스를 직접 시작할 수 있습니다.
- 인스턴스 하드웨어 및 소프트웨어 사전 조건은 인스턴스 사용 계획에 따라 달라집니다.
  - 옵션 1: 인스턴스를 Charon 에뮬레이터 호스트 시스템으로 사용합니다.
    - Charon 제품의 사용 설명서 및/또는 시작 안내서의 하드웨어 및 소프트웨어 사전 조건 섹션을 참조하여 Linux 인스턴스에서 충족해야 하는 정확한 하드웨어 및 소프트웨어 사전 조건을 파악

합니다. 인스턴스를 시작하는 데 사용하는 이미지와 선택한 인스턴스 유형에 따라 클라우드 인스턴스의 소프트웨어 및 하드웨어가 결정됩니다.

- 에뮬레이션된 레거시 시스템을 실행하려면 Charon 제품 라이선스가 필요합니다. 자세한 내용은 Charon 제품 설명서의 라이선스 정보를 참조하거나 Stromasys 담당자 또는 Stromasys VAR에 문의하세요.

- 옵션 2: 인스턴스를 전용 VE 라이선스 서버로 사용합니다.
  - 자세한 사전 조건은 VE 라이선스 서버 안내서를 참조하세요.

- Charon 에뮬레이터 제품에서 제공하는 에뮬레이션된 시스템에서 실행할 수 있는 특정 레거시 운영 체제에는 운영 체제의 원래 공급업체의 라이선스가 필요합니다. 사용자는 기존 운영 체제와 관련된 모든 라이선스 의무에 대한 책임이 있으며, 적절한 라이선스를 제공해야 합니다.

를 AWS Management Console 사용하여 새 인스턴스를 시작합니다.

새 인스턴스를 생성하려면

1. AWS Management Console 로그인하고 https://console.aws.amazon.com/ec2/ 에서 Amazon EC2 콘솔을 엽니다.
2. 인스턴스 시작을 선택합니다.
3. 인스턴스의 이름을 입력합니다.
4. AMI를 선택합니다. AMI는 클라우드 인스턴스를 시작하는 데 사용되는 사전 패키징된 이미지입니다. 여기에는 운영 체제와 해당 애플리케이션 소프트웨어가 포함됩니다. AMI 선택은 인스턴스를 어떻게 사용할 계획인지에 따라 달라집니다.

   - 인스턴스를 Charon 에뮬레이터 호스트 시스템으로 사용하려는 경우 다음과 같은 여러 AMI를 선택할 수 있습니다.
     - 사전 패키징된 Charon 마켓플레이스 이미지에서 Charon 호스트 시스템 설치: 이미지에는 기본 운영 체제와 사전 설치된 Charon 소프트웨어가 포함되어 있습니다.
       - 현재 클라우드 제공업체 마켓플레이스에서 어떤 옵션을 사용할 수 있는지 Stromasys 담당자에게 문의하세요.
       - 클라우드 제공업체 및 Stromasys 제품 릴리스 계획에 따라 2가지 변형이 있을 수 있습니다.
         - Stromasys가 운영하는 퍼블릭 라이선스 서버 또는 고객이 운영하는 프라이빗 AutoVE 라이선스 서버에서 사용하기 위한 자동 라이선스(AL)
         - 고객이 운영하는 프라이빗 VE 라이선스 서버와 함께 사용하기 위한 가상 환경(VE)

- Linux용 Charon 에뮬레이터 설치 RPM 패키지와 함께 일반적인 Charon 에뮬레이터 설치를 사용하여 Charon 호스트 시스템 설치:
  - 선택한 Charon 제품 및 버전에서 지원하는 배포판의 Linux AMI를 선택합니다. Stromasys 설명서 사이트에서 해당 제품의 사용 설명서를 참조하세요.
- 인스턴스를 전용 VE 라이선스 서버로 사용하려는 경우 라이선스 설명서의 VE 라이선스 서버 안내서에서 Linux 인스턴스의 요구 사항을 참조하세요.

필요한 AMI를 결정한 후 일치하는 Linux 또는 Charon 제품 AMI를 선택합니다. 필요한 AMI가 표시되지 않으면 더 많은 AMI 찾아보기를 선택합니다. 인스턴스 사용 계획과 일치하는 Linux AMI를 선택합니다. 다음 중 하나가 될 수 있습니다.

- 사전 패키징된 Charon VE 마켓플레이스 이미지로, AMI 이름에 문자열 've' 포함.
- 자동 라이선싱 또는 AutoVE용으로 사전 패키징된 Charon AL 마켓플레이스 이미지.
- RPM 제품 설치를 지원하는 Linux 버전.
- VE 라이선스 서버에 지원되는 Linux 버전.

5. 인스턴스 유형을 선택합니다. Amazon EC2는 CPU, 메모리, 스토리지, 네트워킹 용량을 다양하게 조합한 인스턴스 유형을 제공합니다. 사용하려는 Charon 제품의 요구 사항과 일치하는 인스턴스 유형을 선택합니다. 일부 마켓플레이스 이미지에는 인스턴스 유형 선택이 제한되어 있습니다.
6. 기존 키 페어를 선택하거나 새 페어를 생성 후 저장합니다. 기존 키 페어를 선택하는 경우 일치하는 프라이빗 키가 있는지 확인합니다. 없으면 인스턴스에 연결할 수 없습니다.

> **Note**
>
> 관리 시스템에서 지원한다면 RHEL 9.x, Rocky Linux 9.x, Oracle Linux 9.x의 경우 SSH 키 유형 ECDSA 또는 ED25519를 사용하세요. 이러한 유형을 사용하면 Charon 호스트의 기본 암호화 정책 설정을 보안 수준이 떨어지는 설정으로 변경하지 않아도 SSH 터널을 통해 해당 Charon 호스트 Linux 시스템에 연결할 수 있습니다. 예를 들어, 이는 Charon-SSP 관리자에게 중요합니다. Red Hat 설명서에서 시스템 전반의 암호화 정책 사용을 참조하십시오.

7. 네트워크 설정 섹션에서 편집을 선택합니다. 환경에 맞는 설정을 선택합니다.
   - VPC를 지정합니다.
   - 기존 서브넷을 지정하거나 새 서브넷을 생성합니다.

- 기본 인터페이스에 퍼블릭 IP 주소의 자동 할당을 활성화하거나 비활성화합니다. 자동 할당은 인스턴스에 단일 네트워크 인터페이스가 있는 경우에만 가능합니다.
- 기존 또는 새 사용자 지정 보안 그룹을 할당합니다. 보안 그룹은 인스턴스에 대한 액세스를 최소한 SSH 이상으로 허용해야 합니다. 인스턴스에서 실행하려는 애플리케이션에 필요한 모든 포트도 허용되어야 합니다. 인스턴스를 생성하고 나서 언제든지 보안 그룹을 수정할 수 있습니다.

8. 스토리지 섹션의 루트 볼륨(시스템 디스크)에서 환경에 적합한 크기를 선택합니다. Linux 시스템에 권장되는 최소 시스템 디스크 크기는 30GiB입니다. 가상 디스크 컨테이너 및 기타 스토리지 요구 사항을 위한 공간을 제공하려면 지금 또는 인스턴스를 시작한 후에 스토리지를 더 추가할 수 있습니다. 하지만 시스템 디스크 크기는 설치하려는 모든 애플리케이션과 유틸리티를 포함하여 Linux 시스템 요구 사항을 충족해야 합니다.

> **Note**
>
> Charon 애플리케이션 데이터(예: 디스크 이미지)를 위한 별도의 스토리지 볼륨을 생성하는 것이 좋습니다. 필요한 경우 나중에 해당 볼륨을 다른 인스턴스로 마이그레이션할 수 있습니다.

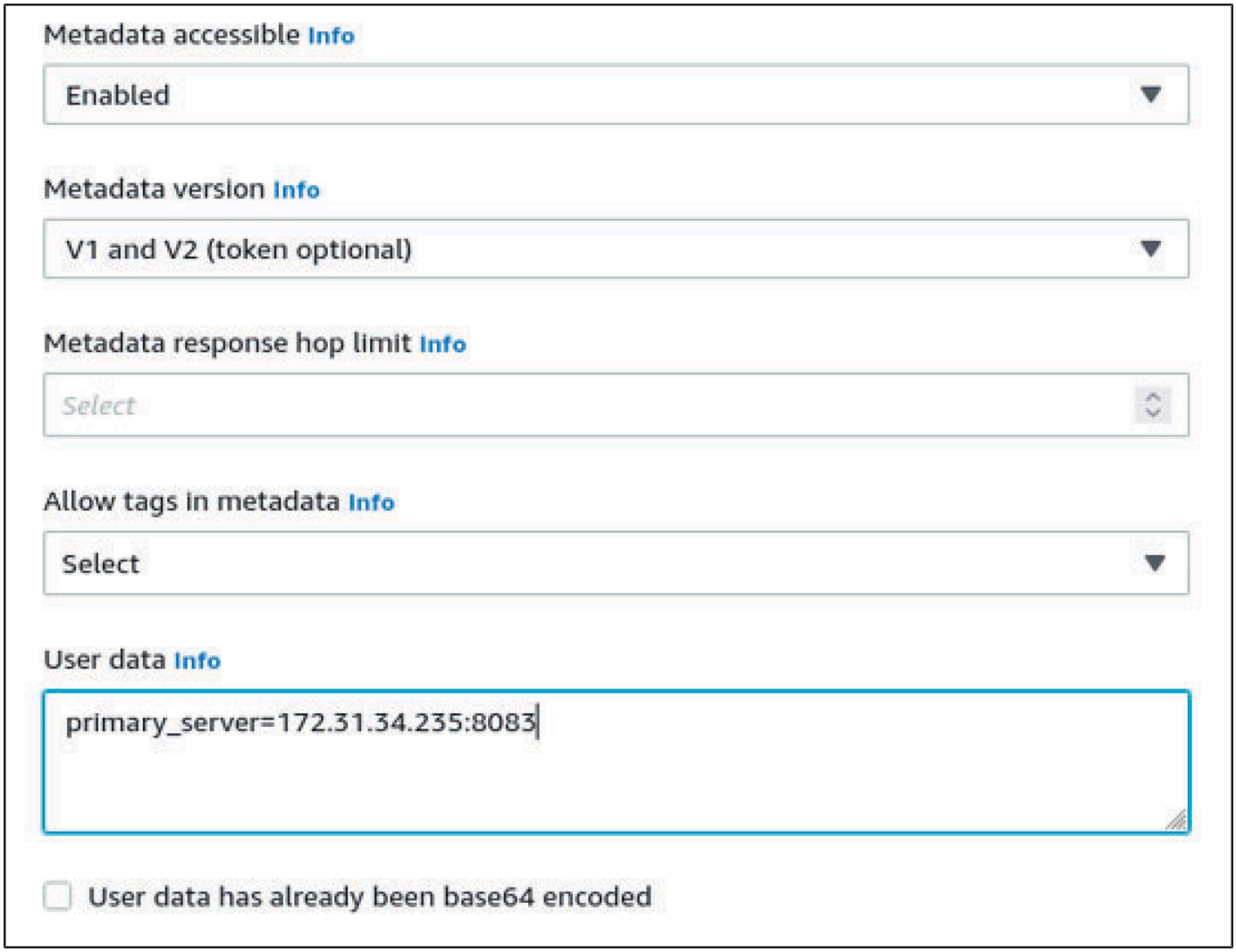
9. 고급 세부 정보 섹션을 확장하고 스크롤을 내려 CPU 옵션 지정을 선택합니다. 다음 이미지에는 Charon 에뮬레이터 환경에 유용할 가능성이 높은 3가지가 예시로 나와 있습니다.

10. 1.1.23 이전 버전의 VE 라이선스 서버 시스템의 경우 인스턴스에 필요한 IAM 역할을 할당해야 합니다. `ListUsers` 작업을 허용하는 역할이어야 합니다. 역할을 할당하려면 확장된 고급 세부 정

보 섹션의 IAM 인스턴스 프로파일에서 역할을 선택하거나 새 IAM 프로파일 생성을 선택합니다. 자세한 내용은 Amazon EC2의 IAM 역할을 참조하세요.

11. 인스턴스가 Charon AL AWS Marketplace 이미지를 기반으로 하고 Stromasys에서 운영하는 퍼블릭 라이선스 서버를 사용하려는 경우, 인스턴스를 시작하기 전에 인스턴스 구성에 해당 정보를 추가해야 합니다.

    다음 이미지와 같이 AutoVE 라이선스 서버의 정보를 입력합니다.



    유효한 사용자 데이터 구성 옵션은 다음과 같습니다.

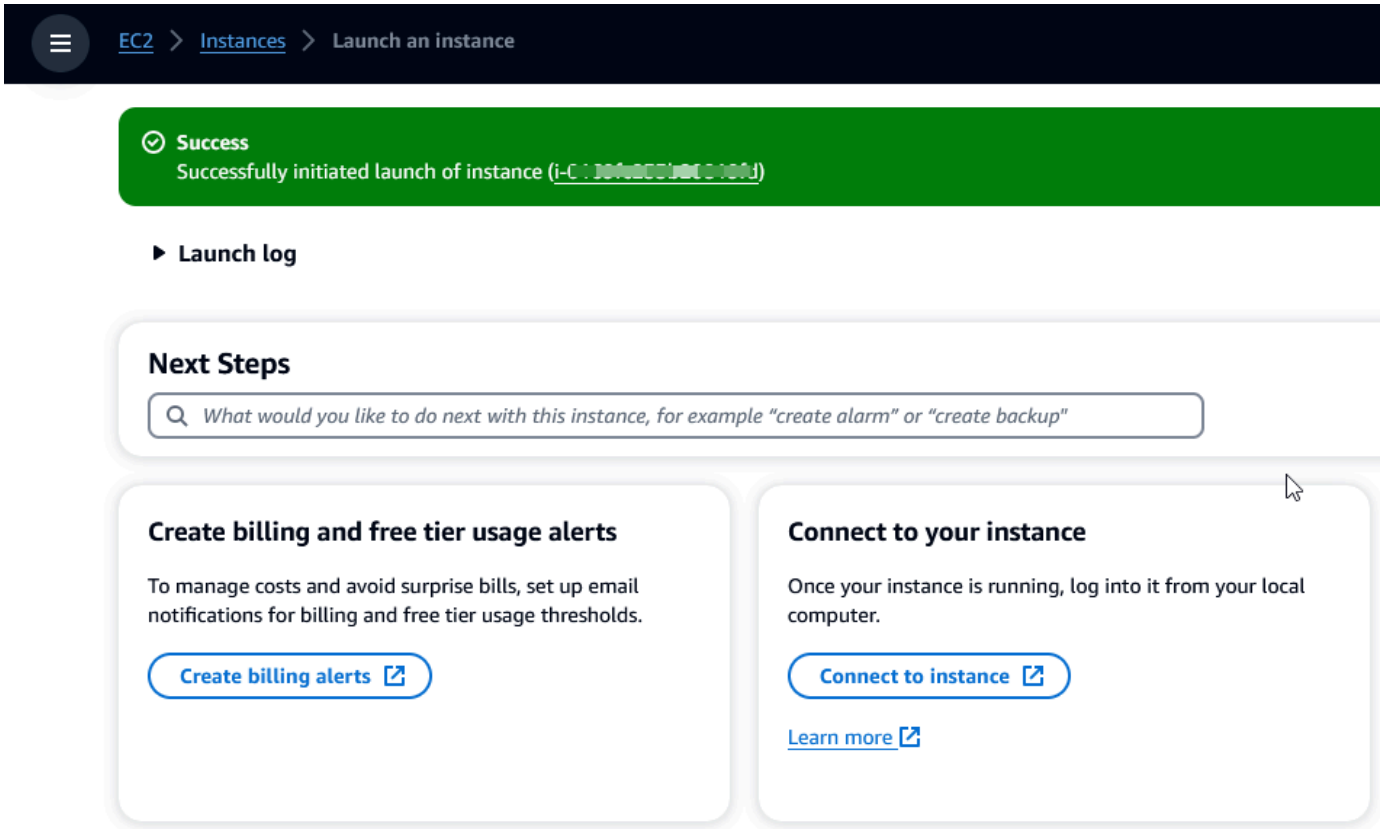
    - **primary_server=**<*ip-address*>[**:**<*port*>]
    - **backup_server=**<*ip-address*>[**:**<*port*>]

    위치

- <ip-address>는 해당되는 기본 및 백업 서버의 IP 주소를 나타냅니다.
- <port>는 라이선스 서버와 통신하는 데 사용되는 기본이 아닌 TCP 포트(기본값: TCP/8083)를 나타냅니다.

> **Note**
>
> AutoVE 모드를 활성화하려면 최초 실행 시 하나 이상의 라이선스 서버를 구성해야 합니다. 그렇지 않으면 인스턴스가 Stromasys에서 운영하는 퍼블릭 라이선스 서버 중 하나에 바인딩됩니다.

12. 요약 섹션에서 인스턴스 실행을 선택합니다. 잠시 후 다음과 같은 성공 메시지가 나타납니다.



13. 화면의 오른쪽 하단 모서리에서 모든 인스턴스 보기를 선택합니다.
14. 인스턴스의 세부 정보를 보려면 인스턴스 표의 인스턴스를 나타내는 행 왼쪽에 있는 확인란을 선택합니다. 인스턴스 세부 정보가 화면 하단에 나타납니다. 인스턴스에 연결하는 방법에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 연결을 참조하세요.

AWS NTT DATA를 통한 메인프레임 현대화 플랫폼 재구성

AWS 메인프레임 현대화는 다양한 Amazon 머신 이미지 (AMI) 를 제공합니다. 이러한 AMI를 사용하면 Amazon EC2 인스턴스를 신속하게 프로비저닝할 수 있어 NTT 데이터를 사용하여 메인프레임 애플리케이션을 재호스팅하고 재배치할 수 있는 맞춤형 환경을 구축할 수 있습니다. AWS 이 안내서는 이러한 AMI에 액세스하고 사용하는 데 필요한 단계를 제공합니다.

필수 조건

- Amazon EC2 인스턴스를 생성할 수 있는 AWS 계정에 대한 관리자 액세스 권한이 있는지 확인하십시오.
- Amazon EC2 인스턴스를 생성하려는 지역에서 AWS 메인프레임 현대화 서비스를 사용할 수 있는지 확인하십시오. 리전별 사용 가능한 AWS 서비스 목록을 참조하세요.
- Amazon EC2 인스턴스를 생성하려는 Amazon VPC를 식별합니다.

Amazon Machine Image 구독

AWS Marketplace 제품을 구독하면 제품의 AMI에서 인스턴스를 시작할 수 있습니다.

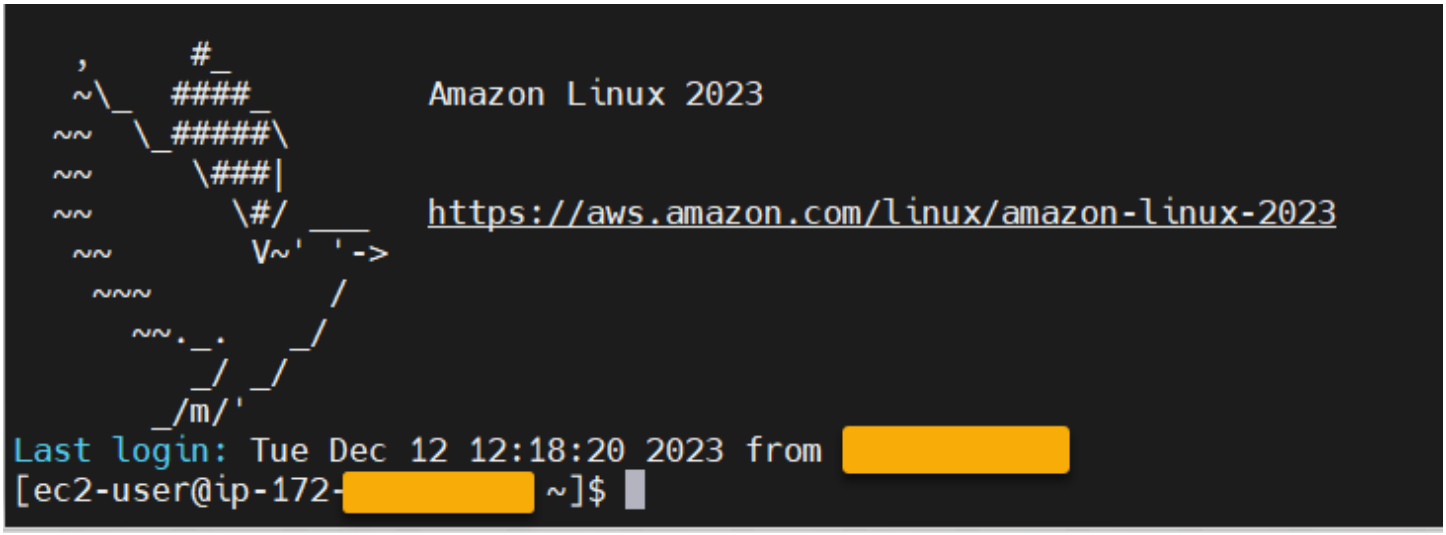
1. AWS Management Console 로그인하고 https://console.aws.amazon.com/marketplace 에서 콘솔을 엽니다. AWS Marketplace
2. 구독 관리를 선택합니다.
3. https://aws.amazon.com/marketplace/pp/prodview-eg227ymldsnx2 링크를 복사하여 브라우저 주소 표시줄에 붙여 넣습니다.
4. 구독 계속을 선택합니다.
5. 이용 약관에 동의하는 경우 약관 동의를 선택합니다. 이 프로세스에는 몇 분이 걸릴 수 있습니다.
6. 감사 메시지가 나타날 때까지 기다립니다. 이 메시지는 제품을 성공적으로 구독했음을 확인하는 메시지입니다.
7. 왼쪽 탐색 창에서 구독 관리를 선택합니다. 이 화면에는 모든 구독이 표시됩니다.

NTT AWS DATA 인스턴스로 메인프레임 현대화 리플랫폼 개편 시작

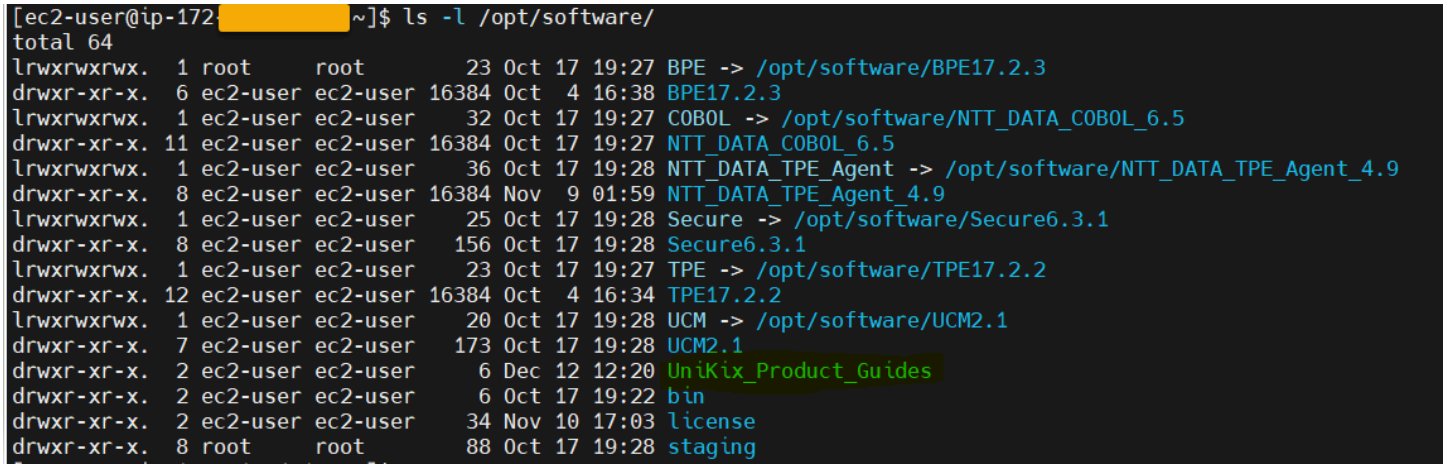
1. https://console.aws.amazon.com/marketplace 에서 콘솔을 엽니다. AWS Marketplace
2. 왼쪽 탐색 창에서 구독 관리를 선택합니다.
3. 시작하려는 AMI를 찾은 다음 새 인스턴스 실행을 선택합니다.
4. 리전에서 허용 목록에 있는 리전을 선택합니다.
5. 계속해서 EC2를 통해 시작을 선택하세요. 이 작업을 수행하면 Amazon EC2 콘솔로 이동합니다.
6. 서버 이름에 DNS 이름을 입력합니다.
7. 프로젝트 성능 및 비용 요구 사항에 맞는 인스턴스 유형을 선택합니다. 인스턴스 크기의 권장 시작점은 `c5.2xLarge`입니다.
8. 기존 키 페어를 선택하거나 새 페어를 생성 후 저장합니다. 키 페어에 대한 자세한 내용은 Amazon EC2 Linux 인스턴스용 사용 설명서에서 Amazon EC2 키 페어 및 Linux 인스턴스를 참조하세요.
9. 네트워크 설정을 편집하고 허용 목록에 있는 VPC와 적절한 서브넷을 선택합니다.
10. 기존 보안 그룹을 선택하거나 새 보안 그룹을 생성합니다. Enterprise Server Amazon EC2 인스턴스인 경우 일반적으로 포트 86과 10086으로의 TCP 트래픽을 허용하여 Micro Focus 구성을 관리합니다.
11. Amazon EC2 인스턴스의 스토리지를 구성합니다.
12. 요약을 검토하고 인스턴스 실행을 선택합니다. 성공적으로 실행되려면 인스턴스 유형이 유효해야 합니다. 시작에 실패할 경우 인스턴스 구성 편집을 선택하고 다른 인스턴스 유형을 선택합니다.
13. 성공 메시지가 표시되면 인스턴스에 연결을 선택합니다.
14. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
15. 왼쪽 탐색 창의 인스턴스 메뉴에서 인스턴스를 선택합니다.
16. 기본 창에서 인스턴스의 상태를 확인합니다.

NTT Data 사용 시작하기

Amazon EC2 인스턴스를 프로비저닝한 후 사용자 이름 `ec2-user`를 사용하여 SSH로 인스턴스에 연결합니다. 화면은 다음과 같은 이미지로 보입니다.



/opt/software/ 폴더 아래에는 다음 이미지와 같이 UniKix_Product_Guides 이름이 지정된 폴더가 있습니다.



UniKix_Product_Guides 폴더에는 이 Amazon EC2 인스턴스에 설치된 다음 구성 요소에 대한 설명서가 포함되어 있습니다.

- NTT DATA TPE
- NTT DATA BPE
- NTT DATA Enterprise COBOL
- NTT 데이터 UniKix 시큐어
- NTT 데이터 센트럴 매니저 UniKix

이전 이미지에 표시된 software 폴더에는 위에 나열된 구성 요소의 바이너리가 있습니다.

Amazon EC2 인스턴스를 성공적으로 검증한 후에는 NTT Data 설명서에 따라 NTT DATA를 통한 AWS 메인프레임 현대화 플랫폼 변경을 시작하십시오.

AWS 메인프레임 현대화 애플리케이션

AWS 메인프레임 현대화를 처음 사용하는 경우 시작하려면 다음 주제를 참조하세요.

- 메인프레임 AWS 현대화란?
- AWS 메인프레임 현대화 설정
- 자습서: AWS Blu Age를 위한 관리형 런타임
- 자습서: Micro Focus의 관리형 런타임

메인프레임 현대화의 애플리케이션에는 AWS 마이그레이션된 메인프레임 워크로드가 포함되어 있습니다. 애플리케이션은 메인프레임의 워크로드와 유사하며 런타임 환경과 연결되어 있습니다. 애플리케이션에 일괄 파일 및 데이터 세트를 추가하고 애플리케이션 실행 시 애플리케이션을 모니터링할 수 있습니다. 마이그레이션하는 각 워크로드에 대해 AWS 메인프레임 현대화 애플리케이션을 생성합니다. AWS 메인프레임 현대화 애플리케이션을 생성할 때는 애플리케이션을 생성할 때 애플리케이션이 실행되는 엔진을 지정합니다. 자동 리팩토링 패턴을 사용하는 경우 AWS Blu Age를 선택하고 플랫폼 변경 패턴을 사용하는 경우 Micro Focus를 선택합니다.

주제

- AWS 메인프레임 현대화 애플리케이션 만들기
- AWS 메인프레임 현대화 애플리케이션 배포
- AWS 메인프레임 현대화 애플리케이션 업데이트
- AWS 메인프레임 현대화 애플리케이션을 환경에서 삭제
- AWS 메인프레임 현대화 애플리케이션 삭제
- AWS 메인프레임 현대화 애플리케이션을 위한 일괄 작업 제출
- AWS 메인프레임 현대화 애플리케이션을 위한 데이터 세트 가져오기
- AWS 메인프레임 현대화 애플리케이션의 트랜잭션을 관리합니다
- 마이그레이션된 애플리케이션을 위한 AWS 리소스 생성
- 관리형 애플리케이션 구성
- AWS 메인프레임 현대화 애플리케이션 정의 레퍼런스
- AWS 메인프레임 현대화 데이터 세트 정의 참조

AWS 메인프레임 현대화 애플리케이션 만들기

AWS 메인프레임 현대화 콘솔을 사용하여 AWS 메인프레임 현대화 애플리케이션을 만들 수 있습니다.

이러한 지침에서는 AWS 메인프레임 현대화 설정의 단계를 완료한 것으로 가정합니다.

애플리케이션 생성

애플리케이션을 생성하려면

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 애플리케이션을 생성할 리전을 선택합니다.
3. 애플리케이션 페이지에서 애플리케이션 생성을 선택합니다.
4. 기본 정보 지정 페이지의 이름 및 설명 섹션에 애플리케이션 이름을 입력합니다.
5. (선택 사항) 애플리케이션 설명에 애플리케이션에 대한 설명을 입력합니다. 이 설명은 사용자들이 애플리케이션의 용도를 식별하는 데 도움이 됩니다.
6. 엔진 유형 섹션에서 자동 리팩토링을 사용하려면 Blu Age를 선택하고 플랫폼 재구성을 위해서는 Micro Focus를 선택합니다.
7. 고객 관리 AWS KMS 키를 사용하려면 KMS 키 섹션에서 암호화 설정 사용자 지정을 선택합니다. 자세한 설명은 메인프레임 현대화 서비스를 위한 유휴 데이터 암호화 AWS 섹션을 참조하세요.

   > **Note**
   >
   > 기본적으로 AWS 메인프레임 현대화는 AWS 메인프레임 현대화가 소유하고 관리하는 AWS KMS 키로 데이터를 암호화합니다. 하지만 고객 관리형 AWS KMS 키를 사용하도록 선택할 수 있습니다.

8. (선택 사항) 이름 또는 Amazon 리소스 이름(ARN) 별로 AWS KMS 키를 선택하거나 AWS KMS 키 생성을 선택하여 AWS KMS 콘솔로 이동하고 새 AWS KMS 키를 생성합니다.
9. (선택 사항) 태그 섹션에서 새 태그 추가를 선택하여 애플리케이션에 하나 이상의 애플리케이션 태그를 추가합니다. 애플리케이션 태그는 사용자 또는 AWS 리소스를 구성하고 관리하는 데 도움이 되는 사용자 지정 속성 레이블입니다.
10. 다음을 선택합니다.
11. 리소스 및 구성 섹션에서 인라인 편집기를 사용하여 애플리케이션 정의를 입력합니다. 또는 Amazon S3 버킷의 애플리케이션 정의 JSON 파일 사용을 선택하고 사용하려는 애플리케이션

정의의 위치를 제공하세요. 자세한 설명은 AWS 블루 에이지 애플리케이션 정의 샘플 또는 Micro Focus 애플리케이션 정의 섹션을 참조하세요.

12. 다음을 선택합니다.
13. 검토 및 생성 페이지에서 제공한 정보를 검토한 다음 데이터 소스 생성을 선택합니다.

AWS 메인프레임 현대화 애플리케이션 배포

AWS 메인프레임 현대화 콘솔을 사용하여 AWS 메인프레임 현대화 애플리케이션을 배포하세요.

이러한 지침에서는 AWS 메인프레임 현대화 설정의 단계를 완료한 것으로 가정합니다.

애플리케이션 배포

AWS 메인프레임 현대화 애플리케이션을 실행하려면 먼저 런타임 환경에 배포해야 합니다. 애플리케이션에는 버전이 둘 이상 있을 수 있습니다. 애플리케이션의 각 버전에는 고유한 애플리케이션 정의가 있습니다. 애플리케이션을 배포하려면 배포하려는 버전을 지정해야 합니다.

한 번에 한 버전의 애플리케이션만 배포할 수 있습니다. 애플리케이션 버전을 배포한 다음 다른 버전을 배포하기로 결정하는 경우, 실행 중인 애플리케이션을 먼저 중지해야 합니다.

애플리케이션 배포

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 추적을 생성할 리전을 선택합니다.
3. 애플리케이션 페이지에서 배포하고자 하는 애플리케이션을 선택합니다.
4. 애플리케이션 배포를 선택합니다.
5. 사용 가능한 버전 섹션에서 배포하려는 버전을 선택합니다.
6. 환경 섹션에서 애플리케이션을 실행할 런타임 환경을 선택합니다.
7. 배포를 선택합니다.

배포된 애플리케이션의 다른 버전을 배포하려면

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 추적을 생성할 리전을 선택합니다.
3. 애플리케이션 페이지에서 삭제하고자 하는 애플리케이션을 선택합니다.
4. 작업 메뉴에서 애플리케이션 중지를 선택합니다.

5. 애플리케이션이 중지된 후 애플리케이션 배포를 선택합니다.
6. 사용 가능한 버전 섹션에서 배포하려는 버전을 선택합니다. 환경 섹션에는 애플리케이션이 이미 배포되어 있는 환경이 미리 선택됩니다.
7. 배포를 선택합니다.

AWS 메인프레임 현대화 애플리케이션 업데이트

AWS 메인프레임 현대화 콘솔을 사용하여 AWS 메인프레임 현대화 애플리케이션을 업데이트하세요.

이러한 지침에서는 AWS 메인프레임 현대화 설정의 단계를 완료한 것으로 가정합니다.

애플리케이션 업데이트

AWS 메인프레임 현대화 애플리케이션에는 각각 고유한 애플리케이션 정의가 있는 여러 버전이 있을 수 있습니다. 애플리케이션을 업데이트하려면 새 애플리케이션 정의를 제공하세요. 이는 새 버전의 애플리케이션을 만듭니다.

애플리케이션을 업데이트하려면

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 업데이트하려는 애플리케이션이 생성된 리전을 선택합니다.
3. 애플리케이션 페이지에서 업데이트하려는 애플리케이션을 선택합니다.
4. 애플리케이션 세부 정보 페이지의 현재 정의 섹션에서 편집을 선택하여 현재 애플리케이션 정의를 업데이트합니다.
5. 애플리케이션 업데이트 페이지에서 인라인 편집기를 사용하여 현재 애플리케이션 정의를 업데이트합니다.

   또는 Amazon S3 버킷의 애플리케이션 정의 JSON 파일 사용을 선택하고 사용하려는 애플리케이션 정의의 위치를 제공하세요. 자세한 설명은 AWS 블루 에이지 애플리케이션 정의 샘플 또는 Micro Focus 애플리케이션 정의 섹션을 참조하세요.
6. 애플리케이션 정의 업데이트를 완료하면 업데이트를 선택합니다.

> **Note**
>
> 애플리케이션을 업데이트한 후에는 애플리케이션을 다시 배포해야 합니다. 자세한 설명은 AWS 메인프레임 현대화 애플리케이션 배포 섹션을 참조하세요.

AWS 메인프레임 현대화 애플리케이션을 환경에서 삭제

AWS 메인프레임 현대화 콘솔을 사용하여 환경에서 AWS 메인프레임 현대화 애플리케이션을 삭제할 수 있습니다.

이러한 지침에서는 AWS 메인프레임 현대화 설정의 단계를 완료한 것으로 가정합니다.

애플리케이션을 환경에서 삭제합니다

실행 중인 AWS 메인프레임 현대화 애플리케이션을 삭제해야 하는 경우 먼저 중지해야 합니다. 애플리케이션 페이지에서 애플리케이션 상태를 확인할 수 있습니다.

애플리케이션을 환경에서 삭제하려면

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 환경에서 삭제하려는 애플리케이션이 생성된 리전을 선택합니다.
3. 애플리케이션 페이지에서 환경에서 삭제하려는 애플리케이션을 선택한 다음 작업을 선택합니다.
4. (선택 사항) 애플리케이션 상태가 `Running`이면 애플리케이션 중지를 선택합니다.
5. 환경에서 삭제를 선택합니다.

   삭제 프로세스가 즉시 시작됩니다.

AWS 메인프레임 현대화 애플리케이션 삭제

AWS 메인프레임 현대화 콘솔을 사용하여 AWS 메인프레임 현대화 애플리케이션을 삭제합니다.

이러한 지침에서는 AWS 메인프레임 현대화 설정의 단계를 완료한 것으로 가정합니다.

애플리케이션을 삭제합니다

실행 중인 AWS 메인프레임 현대화 애플리케이션을 삭제해야 하는 경우 먼저 중지해야 합니다. 애플리케이션 페이지에서 애플리케이션 상태를 확인할 수 있습니다.

애플리케이션 삭제

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 삭제하려는 애플리케이션이 생성된 리전을 선택합니다.

3. 애플리케이션 페이지에서 삭제하려는 애플리케이션을 선택한 다음 작업을 선택합니다.
4. (선택 사항) 애플리케이션 상태가 Running이면 애플리케이션 중지를 선택합니다.
5. 애플리케이션 삭제를 선택합니다.
6. 애플리케이션 삭제 창에서 **delete**을 입력하여 삭제할 것인지 확인한 후 삭제를 선택합니다.

AWS 메인프레임 현대화 애플리케이션을 위한 일괄 작업 제출

AWS 메인프레임 현대화에서는 애플리케이션에 대한 일괄 작업을 제출할 수 있습니다. 일괄 작업을 제출 또는 취소하고 일괄 작업 실행에 대한 세부 정보를 검토할 수 있습니다. 일괄 작업을 제출할 때마다 AWS 메인프레임 현대화에서는 별도의 일괄 작업 실행을 생성합니다. 이 작업을 모니터링할 수 있습니다. 이름으로 일괄 작업을 검색하고 일괄 작업에 JCL 또는 스크립트 파일을 제공할 수 있습니다.

> **Important**
>
> 일괄 작업을 취소해도 작업은 삭제되지 않습니다. 일괄 작업의 특정 실행이 취소됩니다. 일괄 작업 기록은 일괄 작업 실행에 대한 세부 정보에서 계속 볼 수 있습니다.

일괄 작업에 하나 이상의 데이터 세트에 대한 액세스가 필요한 경우 AWS 메인프레임 현대화 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용하여 데이터 세트를 가져오세요. 자세한 설명은 AWS 메인프레임 현대화 애플리케이션을 위한 데이터 세트 가져오기 섹션을 참조하세요.

이러한 지침에서는 의 단계를 완료한 것으로 가정합니다.

일괄 작업 제출

일괄 작업을 제출하려면

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 일괄 작업을 제출하려는 애플리케이션이 생성된 리전을 선택합니다.
3. 애플리케이션 페이지에서 일괄 작업을 제출하려는 애플리케이션을 선택합니다.

> **Note**
>
> 일괄 작업을 애플리케이션에 제출하려면 먼저 애플리케이션을 성공적으로 배포해야 합니다.

4. 애플리케이션 세부 정보 페이지에서 일괄 작업을 선택합니다.
5. 작업 제출을 선택합니다.
6. 스크립트 섹션에서 선택에서 스크립트를 선택합니다. 원하는 스크립트를 이름으로 검색할 수 있습니다.
7. 작업 제출을 선택합니다.

AWS 메인프레임 현대화 애플리케이션을 위한 데이터 세트 가져오기

AWS 메인프레임 현대화를 사용하면 애플리케이션에 사용할 데이터 세트를 가져올 수 있습니다. Amazon S3 버킷에 저장된 JSON 파일에 데이터 세트를 지정하거나 데이터 세트 구성 값을 별도로 지정할 수 있습니다. 데이터 세트를 가져온 후 가져오기 작업의 세부 정보를 검토하여 원하는 데이터 세트를 가져왔는지 확인할 수 있습니다. 애플리케이션에 대해 카탈로그로 작성된 모든 데이터 세트가 콘솔에 함께 나열됩니다.

AWS 메인프레임 현대화 콘솔을 사용하여 AWS 메인프레임 현대화 애플리케이션을 위한 데이터 세트를 가져올 수 있습니다.

이러한 지침에서는 AWS 메인프레임 현대화 설정 및 AWS 메인프레임 현대화 애플리케이션 만들기의 단계를 완료한 것으로 가정합니다.

데이터 세트 가져오기

데이터 가져오기를 시작하려면

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 데이터 세트를 가져오려는 애플리케이션이 생성된 리전을 선택합니다.
3. 애플리케이션 페이지에서 데이터 세트를 가져오려는 애플리케이션을 선택합니다.
4. 애플리케이션 세부 정보 페이지에서 데이터 세트를 선택합니다.
5. 가져오기를 선택합니다.
6. 다음 중 하나를 수행합니다.
   - Amazon S3 버킷의 데이터 세트 구성 JSON 파일 사용을 선택하고 데이터 세트 구성 위치를 제공합니다.
   - 안내된 구성에 따라 데이터 세트 구성 값을 별도로 지정을 선택합니다. 구체적인 정의 세부 정보는 the section called “데이터 세트 정의 참조”를 참조하세요.

이름, 데이터 세트 구성(VSAM, GDG, PO, PS), 위치, 외부 Amazon S3 위치, 각 데이터 세트 구성 값의 파라미터 설정을 입력합니다. 구성 안내에서는 JSON 생성을 선택하여 입력한 내용을 바탕으로 JSON 구성을 검토할 수도 있습니다.

7. 제출을 선택합니다.

AWS 메인프레임 현대화 애플리케이션의 트랜잭션을 관리합니다

AWS 메인프레임 현대화를 사용하면 동일한 파일 및 프로그램을 사용하여 동일한 애플리케이션을 실행하도록 요청을 제출하는 다른 많은 사용자와 마찬가지로 요청에 따라 애플리케이션을 실행할 수 있습니다. 단일 트랜잭션은 필요한 처리를 수행하는 하나 이상의 애플리케이션 프로그램으로 구성됩니다.

이러한 지침에서는 AWS 메인프레임 현대화 설정 및 AWS 메인프레임 현대화 애플리케이션 만들기의 단계를 완료한 것으로 가정합니다.

애플리케이션 트랜잭션 관리

애플리케이션의 트랜잭션을 표시하고 편집할 수 있습니다.

애플리케이션의 트랜잭션을 관리하려면

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 실행하려는 애플리케이션을 만든 AWS 리전를 선택합니다.
3. 애플리케이션 페이지에서 트랜잭션을 관리하려는 애플리케이션을 선택합니다.
4. 트랜잭션 탭의 트랜잭션 리소스 아래 드롭다운 목록에서 원하는 리소스 표시 방법을 선택합니다. 거래 리소스, 그룹, 목록 또는 SIT에 따라 리소스을 표시할 수 있습니다.

    - 트랜잭션 리소스를 사용하면 파일 정의, 트랜잭션 정의, 프로그램 정의 또는 임시 데이터 대기열 정의에 따라 응용 프로그램 유형을 선택할 수 있습니다.

    > **Note**
    >
    > AWS 메인프레임 현대화 관리형 트랜잭션은 이보다 더 많은 리소스 유형을 지원하지만 여기서 직접 편집할 수는 없습니다. 다른 리소스는 외부에서 편집해야 하며 업데이트된 리소스 항목으로 애플리케이션을 다시 만들어야 합니다.

- 그룹은 트랜잭션 리소스의 모음입니다. 트랜잭션 리소스와 연결할 그룹을 선택할 수도 있습니다.
- 목록은 정렬된 그룹 모음입니다. 목록 보기에서 모든 거래 리소스와 그룹을 볼 수 있습니다. 스타트업 목록은 서버 초기화 시 로드되는 리소스를 결정합니다.
  - Blu Age 리팩터링 엔진을 사용하면 시작 시 포함할 목록을 지정하며 목록 수에는 제한이 없습니다.
  - Micro Focus 리플랫폼 엔진을 사용하면 하나의 SIT에서 최대 4개의 목록을 지정할 수 있습니다.
- SIT(시스템 초기화 테이블)에는 사용 가능한 모든 트랜잭션 구성이 표시됩니다. 속성(이름, 설명, 시작 목록)에 따라 SIT를 찾을 수 있습니다. 선택한 SIT와 연결할 목록을 선택할 수도 있습니다.

> **Note**
>
> SIT는 Micro Focus 리플랫폼 엔진에만 적용됩니다.

5. 모든 리소스 정보를 표시할 트랜잭션 리소스를 선택합니다. 또한 트랜잭션 리소스과 관련된 모든 속성을 보고 추가 속성을 보거나 편집할 수 있습니다.
6. 현재 트랜잭션 리소스와 관련된 정보를 편집하려면 편집을 선택합니다.

   a. 편집 페이지에서 리소스 설명을 추가하거나 변경할 수 있습니다.

      1. 목록에 표시된 트랜잭션 리소스의 경우 필요에 따라 목록을 추가, 제거 또는 재정렬할 수도 있습니다.
      2. 특정 리소스 유형(파일 정의, 트랜잭션 정의, 프로그램 정의, 임시 데이터 대기열 정의) 의 경우 개별 리소스 속성을 편집할 수 있습니다. 이러한 속성은 엔진 및 리소스 유형에 따라 달라집니다.

   b. 원하는 대로 변경한 후 변경 사항 저장을 선택합니다.

      리소스가 성공적으로 업데이트되면 메시지가 표시됩니다.

마이그레이션된 애플리케이션을 위한 AWS 리소스 생성

AWS에서 마이그레이션된 애플리케이션을 실행하려면 다른 AWS 서비스 리소스와 함께 일부 AWS 리소스를 생성해야 합니다. 생성해야 하는 리소스에는 다음이 포함됩니다.

- 애플리케이션 코드, 구성, 데이터 파일 및 기타 필수 아티팩트를 보관하는 S3 버킷.
- 애플리케이션에 필요한 데이터를 보관하기 위한 Amazon RDS 또는 Amazon Aurora 데이터베이스.
- 암호 생성 및 저장에는 AWS Secrets Manager에 필요한 AWS KMS key가 필요합니다.
- 데이터베이스 자격 증명을 보관하기 위한 Secrets Manager 암호입니다.

> **Note**
>
> 마이그레이션된 각 애플리케이션에는 고유한 리소스 세트가 필요합니다. 이는 최소 설정입니다. 애플리케이션에 Amazon Cognito 보안 암호 또는 MQ 대기열과 같은 추가 리소스가 필요할 수도 있습니다.

필요한 권한

다음 권한이 활성화되어 있는지 확인하세요.

- `s3:CreateBucket`, `s3:PutObject`
- `rds:CreateDBInstance`
- `kms:CreateKey`
- `secretsmanager:CreateSecret`

Amazon S3 버킷

리팩토링된 애플리케이션과 리플랫폼된 애플리케이션 모두 다음과 같이 구성한 S3 버킷이 필요합니다.

```
bucket-name/root-folder-name/application-name
```

bucket-name

Amazon S3 이름 지정 제약 조건 내에 있는 모든 이름. 이름을 버킷 AWS 리전 이름의 일부로 포함하는 것이 좋습니다. 마이그레이션된 애플리케이션을 배포할 계획인 동일 리전에 버킷을 생성해야 합니다.

root-folder-name

AWS 메인프레임 현대화 애플리케이션의 일부로 생성하는 애플리케이션 정의의 제약 조건을 충족하는 데 필요한 이름입니다. `root-folder-name`를 사용하여 여러 버전의 애플리케이션(예: V1 및 V2)을 구분할 수 있습니다.

application-name

마이그레이션된 애플리케이션의 이름 (예: PlanetsDemo 또는 BankDemo)

데이터베이스

리팩토링된 애플리케이션과 리플랫포밍된 애플리케이션 모두 데이터베이스가 필요할 수 있습니다. 각 런타임 엔진의 특정 요구 사항에 따라 데이터베이스를 생성, 구성 및 관리해야 합니다. AWS 메인프레임 현대화는 이 데이터베이스의 전송 중 암호화를 지원합니다. 데이터베이스에서 SSL을 활성화하는 경우 데이터베이스의 연결 세부 정보와 함께 데이터베이스 암호에서 `sslMode`를 지정해야 합니다. 자세한 설명은 AWS Secrets Manager 보안 암호 섹션을 참조하세요.

AWSBlu Age 리팩토링 패턴을 사용하고 BluSam 데이터베이스가 필요한 경우 AWS Blu Age 런타임 엔진에서는 Amazon Aurora PostgreSQL 데이터베이스를 기대하며, 이 데이터베이스를 생성, 구성 및 관리해야 합니다. 데이터베이스는 선택 사항입니다. BluSam 애플리케이션에 필요한 경우에만 이 데이터베이스를 생성하세요. 데이터베이스를 생성하려면 Amazon Aurora 사용 설명서의 Amazon Aurora DB 클러스터 생성에 나와 있는 단계를 따르세요.

Micro Focus 리플랫포밍 패턴을 사용하는 경우 Amazon RDS 또는 Amazon Aurora PostgreSQL 데이터베이스를 생성할 수 있습니다. 데이터베이스를 생성하려면 Amazon RDS 사용 설명서의 Amazon RDS DB 인스턴스 생성 또는 Amazon Aurora 사용 설명서의 Amazon Aurora DB 클러스터 생성의 단계를 따르세요.

두 런타임 엔진 모두 암호화하기 위해 AWS KMS key를 사용하여 AWS Secrets Manager에서 데이터베이스 자격 증명을 저장해야 합니다.

AWS Key Management Service 키

애플리케이션 데이터베이스의 자격 증명을 AWS Secrets Manager에 안전하게 저장해야 합니다. Secrets Manager에서 보안 암호를 생성하려면 AWS KMS key를 생성해야 합니다. KMS 키를 생성하려면 AWS Key Management Service 개발자 안내서의 키 생성의 단계를 따르세요.

키를 생성한 후에는 키 정책을 업데이트하여 AWS 메인프레임 현대화 암호 해독 권한을 부여해야 합니다. 다음 정책 설명을 추가합니다.

```
{
   "Effect" : "Allow",
   "Principal" : {
   "Service" : "m2.amazonaws.com"
   },
   "Action" : "kms:Decrypt",
   "Resource" : "*"
   }
```

AWS Secrets Manager 보안 암호

애플리케이션 데이터베이스의 자격 증명을 AWS Secrets Manager에 안전하게 저장해야 합니다. AWS Secrets Manager 사용 설명서의 암호 생성 단계를 따릅니다.

AWS 메인프레임 현대화는 이 데이터베이스의 전송 중 암호화를 지원합니다. 데이터베이스에서 SSL을 활성화하는 경우 데이터베이스의 연결 세부 정보와 함께 데이터베이스 암호에서 `sslMode`를 지정해야 합니다. `sslMode`, `verify-full`, `verify-ca` 또는 `disable`에 대한 다음 값을 지정할 수 있습니다.

키 생성 프로세스 중에 리소스 권한 - 선택 사항을 선택한 다음 권한 편집을 선택합니다. 정책 편집기에서 다음과 같은 리소스 기반 정책을 추가하여 암호화된 필드의 내용을 검색합니다.

```
{
   "Effect" : "Allow",
   "Principal" : {
   "Service" : "m2.amazonaws.com"
   },
   "Action" : "secretsmanager:GetSecretValue",
   "Resource" : "*"
   }
```

관리형 애플리케이션 구성

레거시 유틸리티에 대한 액세스를 포함하도록 애플리케이션을 구성할 수 있습니다. 추가 속성도 사용자 지정할 수 있습니다. 구성할 수 있는 항목과 위치를 이해하려면 AWS Blu Age 현대화 애플리케이션의 전체 구조를 이해하는 것이 도움이 됩니다.

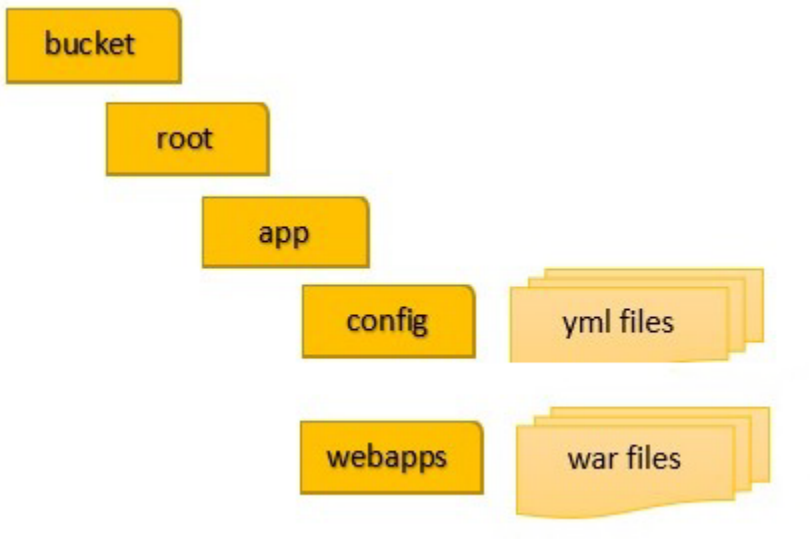
주제

- AWS 블루에이지 관리 애플리케이션의 구조

- 관리 대상 응용 프로그램의 유틸리티에 대한 액세스 구성
- AWS Blu Age 엔진의 구성 속성 추가

AWS 블루에이지 관리 애플리케이션의 구조

AWS Blu Age 리팩토링 패턴을 사용하는 경우 AWS Blu Age 런타임 엔진은 S3 버킷의 폴더 내에 다음과 같은 구조를 예상합니다`application-name`.



config

프로젝트의 YAML 파일이 들어 있습니다. 이 파일은 애플리케이션 전용 YAML 파일로, AWS 메인프레임 현대화가 `application-planetsdemo.yaml` 제공하고 자동으로 설정하는 `application-main.yaml` 파일이 아니라 일반적으로 비슷한 이름을 가집니다.

웹앱

애플리케이션용 `war` 파일이 들어 있습니다. 이러한 파일은 현대화 프로세스의 결과물입니다.

또한 애플리케이션에는 다음과 같은 선택적 폴더가 있을 수 있습니다.

jics/sql

애플리케이션의 JICS 데이터베이스를 초기화하는 `initJics.sql` 스크립트가 들어 있습니다.

스크립트

`war` 파일 내에서 직접 제공할 수도 있는 애플리케이션 스크립트가 들어 있습니다.

sql

`war` 파일 내에 직접 제공할 수도 있는 애플리케이션 SQL 파일이 들어 있습니다.

lnk

`war` 파일 내에 직접 제공할 수도 있는 애플리케이션 LNK 파일이 들어 있습니다.

애플리케이션의 Java 메모리 사용량 관리

애플리케이션의 Java 메모리 사용을 관리하려면 `application-name` 폴더에 이름이 `tomcat.properties`인 속성 파일을 추가합니다. 이 파일은 최소 Java 메모리 사용량을 지정하는 `xms` 속성과 최대 Java 메모리 사용량을 지정하는 `xmx` 속성 등 두 가지 속성을 가질 수 있습니다. 다음은 유효한 `tomcat.properties` 파일의 내용을 보여주는 예제입니다.

```
xms=512M
xmx=1G
```

이 두 속성에 대해 지정하는 값은 다음 단위 중 하나일 수 있습니다.

- 바이트: 단위를 지정하지 마세요.
- 킬로바이트: 값에 K를 추가합니다.
- 메가바이트: 값에 M을 추가합니다.
- 기가바이트: 값에 G를 추가합니다.

관리 대상 응용 프로그램의 유틸리티에 대한 액세스 구성

AWS Blu Age로 메인프레임 애플리케이션을 리팩토링할 때 애플리케이션이 해당 프로그램에 의존하는 경우 IDCAMS, INFUTILB, SORT 등과 같은 다양한 레거시 플랫폼 유틸리티 프로그램에 대한 지원을 제공해야 할 수 있습니다. AWS Blu Age 리팩토링은 현대화된 애플리케이션과 함께 배포되는 전용 웹 애플리케이션을 통해 이러한 액세스를 제공합니다. 이 웹 애플리케이션에는 사용자가 제공해야 하는 `application-utility-pgm.yml` 구성 파일이 필요합니다. 이 구성 파일을 제공하지 않으면 웹 응용 프로그램을 응용 프로그램과 함께 배포할 수 없으며 사용할 수 없게 됩니다.

주제

- 구성 속성

이 항목에서는 application-utility-pgm.yml 구성 파일에서 지정할 수 있는 모든 가능한 속성과 해당 기본값을 설명합니다. 이 항목에서는 필수 속성과 선택적 속성을 모두 설명합니다. 다음은 구성 파일의 예입니다. 여기에는 권장 순서대로 속성이 나열되어 있습니다. 이 예제는 사용자 본인의 구성 파일의 시작점으로 사용할 수 있습니다.

```
# If the datasource support mode is not static-xa, spring JTA transactions
 autoconfiguration must be disabled
 spring.jta.enabled: false
 logging.config: 'classpath:logback-utility.xml'

 # Encoding
 encoding: cp1047

 # Encoding to be used by INFUTILB and DSNUTILB to generate and read SYSPUNCH files
 sysPunchEncoding: cp1047

 # Utility database access
 spring.aws.client.datasources.primary.secret: `arn:aws:secretsmanager:us-
west-2:111122223333:secret:business-FfmXLG`

 treatLargeNumberAsInteger: false

 # Zoned mode : valid values = EBCDIC_STRICT, EBCDIC_MODIFIED, AS400
 zonedMode: EBCDIC_STRICT

 jcl.type: mvs

 # Unload properties
 # For date/time: if use database configuration is enabled, formats are ignored
 # For nbi; use hexadecimal syntaxe to specify the byte value
 unload:
  sqlCodePointShift: 384
  nbi:
    whenNull: "6F"
    whenNotNull: "00"
  useDatabaseConfiguration: false
  format:
    date: MM/dd/yyyy
    time: HH.mm.ss
    timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
  chunkSize:500
  fetchSize: 500
  varCharIsNull: false
```

```
 columnFiller: space

# Load properties
# Batch size for DSNUTILB Load Task
load:
 sqlCodePointShift: 384
 batchSize: 500
 format:
   localDate: dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd
   dbDate: yyyy-MM-dd
   localTime: 'HH:mm:ss|HH.mm.ss'
   dbTime: 'HH:mm:ss'

table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
```

구성 속성

구성 파일에서 다음과 같은 속성을 지정할 수 있습니다.

spring.jta.enabled

선택 사항입니다. JTA 지원이 활성화되는지 여부를 제어합니다. 유틸리티의 경우 이 값을 `false`로 설정하는 것이 좋습니다.

```
spring.jta.enabled : false
```

logging.config

필수 사항입니다. 전용 로거 구성 파일의 경로를 지정합니다. `logback-utility.xml` 이름을 사용하고 이 파일은 현대화된 애플리케이션의 일부로 사용하는 것이 좋습니다. 이러한 파일을 구성하는 일반적인 방법은 모든 로거 구성 파일을 같은 위치, 일반적으로 `/config`는 yaml 구성 파일이 들어 있는 `/config/logback` 하위 폴더에 배치하는 것입니다. 자세한 내용은 로그백 설명서의 챕터 3: 로그백 구성을 참조하세요.

```
logging.config : classpath:logback-utility.xml
```

인코딩

필수 사항입니다. 유틸리티 프로그램에서 사용하는 문자 세트를 지정합니다. 대부분의 경우 z/OS 플랫폼에서 마이그레이션할 때 이 문자 집합은 EBCDIC 변형이며 현대화된 응용 프로그램에 맞게 구성된 문자 집합과 일치해야 합니다. 설정하지 않은 경우 기본값은 ASCII입니다.

```
encoding : cp1047
```

sysPunchEncoding

선택 사항입니다. INFUTILB 및 DSNUTILB가 SYSPUNCH 파일을 생성하고 읽을 때 사용하는 문자 세트를 지정합니다. 레거시 플랫폼의 SYSPUNCH 파일을 그대로 사용하는 경우 이 값은 EBCDIC 변형이어야 합니다. 설정하지 않은 경우 기본값은 ASCII입니다.

```
sysPunchEncoding : cp1047
```

주요 데이터 소스 구성

LOAD 및 UNLOAD와 같은 일부 데이터베이스 관련 유틸리티의 경우 데이터 원본을 통해 대상 데이터베이스에 액세스해야 합니다. AWS 메인프레임 현대화 내의 다른 데이터 소스 정의와 마찬가지로 이 액세스도 반드시 사용해야 합니다. AWS Secrets Manager Secrets Manager에서 적절한 암호를 가리키는 속성은 다음과 같습니다.

spring.aws.client.datasources.primary.secret

선택 사항입니다. Secrets Manager에서 데이터 소스 속성을 포함하는 암호를 지정합니다.

```
spring.aws.client.datasources.primary.secret: datasource-secret-ARN
```

spring.aws.client.datasources.primary.dbname

선택 사항입니다. 데이터베이스 이름이 dbname 속성과 함께 데이터베이스 암호에 직접 제공되지 않는 경우 대상 데이터베이스 이름을 지정합니다.

```
spring.aws.client.datasources.primary.dbname: target-database-name
```

treatLargeNumberAsInteger

선택 사항입니다. 오라클 데이터베이스 엔진 세부 사항 및 DSNTEP2/DSNTEP4 유틸리티 사용과 관련이 있습니다. 이 플래그를 true로 설정하면 오라클 데이터베이스에서 오는 큰 수(NUMBER(38,0))는 정수로 처리됩니다. 기본값: `false`

```
treatLargeNumberAsInteger : false
```

zonedMode

선택 사항입니다. 영역 데이터 유형을 인코딩 또는 디코딩하도록 영역 모드를 설정합니다. 이 설정은 부호 숫자가 표시되는 방식에 영향을 줍니다. 유효한 값은 다음과 같습니다.

- EBCDIC_STRICT: 기본값. 표지판 처리에는 엄격한 정의를 사용하세요. 문자 집합이 EBCDIC인지 ASCII인지에 따라 부호 숫자 표현에는 다음 문자가 사용됩니다.
  - 바이트(Cn+Dn) 에 해당하는 EBCDIC 문자는 양수 및 음수 범위(+0 ~ +9, -0 ~ -9)를 나타냅니다. 문자는 {,A ~ I, }, J ~ R로 표시됩니다
  - 양수 및 음수 범위(+0 ~ +9, -0 ~ -9)를 나타내는 바이트(3n+7n)에 해당하는 ASCII 문자입니다. 문자는 0 ~ 9, p ~ y로 표시됩니다
- EBCDIC_MODIFIED: 표지판 처리에 수정된 정의를 사용합니다. EBDIC와 ASCII 모두에서 동일한 문자 목록이 부호 숫자를 나타냅니다. 즉, +0 ~ +9은{ + A ~ I에 매핑되고 -0 ~ -9는 } + J ~ R에 매핑됩니다. \
- AS400: iSeries(AS400) 플랫폼에서 제공되는 현대화된 레거시 자산에 사용합니다.

```
zonedMode:EBCDIC_STRICT
```

jcl.type

선택 사항입니다. 현대화된 JCL 스크립트의 레거시 유형을 나타냅니다. IDCAMS 유틸리티는 호출 JCL의 유형 vse이 다음과 같은 경우 이 설정을 사용하여 반환 코드를 조정합니다. 유효한 값은 다음과 같습니다.

- mvs(기본값)
- vse

```
jcl.type : mvs
```

데이터베이스: 언로드 유틸리티 관련 속성

이러한 속성을 사용하여 데이터베이스 테이블을 데이터 세트로 언로드하는 유틸리티를 구성할 수 있습니다. 다음과 같은 속성은 모두 선택 사항입니다.

이 예제에서는 가능한 모든 언로드 속성을 보여줍니다.

```
# Unload properties
 # For date/time: if use database configuration is enabled, formats are ignored
 # For nbi; use hexadecimal syntaxe to specify the byte value
 unload:
 sqlCodePointShift: 0
 nbi:
 whenNull: "6F"
 whenNotNull: "00"
 useDatabaseConfiguration: false
 format:
 date: MM/dd/yyyy
 time: HH.mm.ss
 timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
 chunkSize: 0
 fetchSize: 0
 varCharIsNull: false
 columnFiller: space
```

sqlCodePointShift

선택 사항입니다. 데이터에 사용되는 SQL 코드 포인트 시프트를 나타내는 정수 값을 지정합니다. 기본값은 0입니다. 즉, 코드 포인트 이동이 이루어지지 않습니다. 이 설정을 현대화된 애플리케이션에 사용되는 SQL 코드 포인트 이동 매개변수에 맞게 조정하세요. 코드 포인트 이동을 사용하는 경우 이 매개변수의 가장 일반적인 값은 384입니다.

```
unload.sqlCodePointShift: 0
```

nbi

선택 사항입니다. null 인디케이터 바이트를 지정합니다. 이 값은 데이터 값 오른쪽에 추가된 16진수 값(문자열)입니다. 가능한 값은 다음과 같습니다.

- whenNull: 데이터 값이 null인 경우 16진수 값을 추가합니다. 기본값은 6`입니다. 높은 값 FF이 대신 사용되는 경우도 있습니다.

```
unload.nbi.whenNull: "6F"
```

- whenNotNull: 데이터 값이 null은 아니지만 열에 null이 허용되는 경우 16진수 값을 추가합니다. 기본값은 `00`(낮은 값)입니다.

```
unload.nbi.whenNotNull: "00"
```

useDatabaseConfiguration

선택 사항입니다. 날짜 및 시간 형식 지정 속성을 지정합니다. 이는 UNLOAD 쿼리의 날짜/시간 객체를 처리하는 데 사용됩니다. 기본값은 `false`입니다.

- `true`로 설정하면 기본 구성 파일(`application-main.yml`)의 `pgmDateFormat`, `pgmTimeFormat`, 및 `pgmTimestampFormat` 속성을 사용합니다.
- `false`로 설정하면 다음과 같은 날짜 및 시간 형식 지정 속성을 사용합니다.
  - `unload.format.date`: 날짜 서식 패턴을 지정합니다. 기본값은 `MM/dd/yyyy`입니다.
  - `unload.format.time`: 시간 서식 패턴을 지정합니다. 기본값은 `HH.mm.ss`입니다.
  - `unload.format.timestamp`: 타임스탬프 서식 패턴을 지정합니다. 기본값은 `yyyy-MM-dd-HH.mm.ss.SSSSSS`입니다.

chunkSize

선택 사항입니다. SYSREC 데이터 세트를 생성하는 데 사용되는 데이터 청크의 크기를 지정합니다. 이러한 데이터 세트는 병렬 연산을 포함한 데이터 세트 언로드 작업의 대상입니다. 기본값은 `0`(청크 없음)입니다.

```
unload.chunkSize:0
```

fetchSize

선택 사항입니다. 데이터 페치 크기를 지정합니다. 값은 데이터 청크 전략을 사용할 때 한 번에 가져올 레코드 수입니다. 기본값: `0`.

```
unload.fetchSize:0
```

varCharIsNull

선택 사항입니다. 내용이 비어 있는 null로 지정할 수 없는 varchar 열을 처리하는 방법을 지정합니다. 기본값은 `false`입니다.

이 값을 true으로 설정하면 언로드 시 열 내용이 단일 공백 문자열 대신 빈 문자열로 처리됩니다. Oracle 데이터베이스 엔진 케이스에만 이 플래그를 true로 설정하세요.

```
unload.varCharIsNull: false
```

columnFiller

선택 사항입니다. varchar 열에 언로드된 열을 채우는 데 사용할 값을 지정합니다. 가능한 값은 공백 또는 하한 값입니다. 기본값은 공백입니다.

```
unload.columnFiller: space
```

데이터베이스 로드 관련 속성

이러한 속성을 사용하여 데이터 세트 레코드를 대상 데이터베이스에 로드하는 유틸리티(예: DSNUTILB) 를 구성할 수 있습니다. 다음과 같은 속성은 모두 선택 사항입니다.

이 예제에서는 가능한 모든 부하 속성을 보여줍니다.

```
# Load properties
 # Batch size for DSNUTILB Load Task
 load:
 sqlCodePointShift: 384
 batchSize: 500
 format:
 localDate: dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd
 dbDate: yyyy-MM-dd
 localTime: HH:mm:ss|HH.mm.ss
 dbTime: HH:mm:ss

 table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
```

sqlCodePoint시프트

선택 사항입니다. 데이터에 사용되는 SQL 코드 포인트 시프트를 나타내는 정수 값을 지정합니다. 기본값은 0이며, 이는 응용 프로그램에서 코드 포인트 이동을 수행하지 않음을 의미합니다. 이 설정을 현대화된 응용 프로그램에 사용되는 SQL 코드 포인트 이동 파라미터에 맞게 조정하세요. 코드 포인트 이동을 사용하는 경우 이 파라미터의 가장 일반적인 값은 384입니다.

```
load.sqlCodePointShift : 384
```

batchSize

선택 사항입니다. 실제 배치 명령문을 데이터베이스로 보내기 전에 처리할 레코드 수를 나타내는 정수 값을 지정합니다. 기본값은 0입니다.

```
load.batchSize: 500
```

형식

선택 사항입니다. 데이터베이스 로드 작업 중에 날짜/시간 변환에 사용할 날짜 및 시간 형식 지정 패턴을 지정합니다.

- `load.format.localDate`: 현지 날짜 형식 지정 패턴. 기본값은 `dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd`입니다.
- `load.format.dbDate`: 데이터베이스 날짜 형식 지정 패턴. 기본값은 `yyyy-MM-dd`입니다.
- `load.format.localTime`: 현지 시간 형식 지정 패턴. 기본값은 `HH:mm:ss|HH.mm.ss`입니다.
- `load.format.dbTime`: 데이터베이스 시간 형식 지정 패턴. 기본값은 `HH:mm:ss`입니다.

table-mappings

선택 사항입니다. 기존 테이블 이름과 최신 테이블 이름 간의 고객 제공 매핑 컬렉션을 지정합니다. DSNUTILB 유틸리티 프로그램은 이러한 매핑을 사용합니다.

값을 MODERN_TABLE_NAME : LEGACY_TABLE_NAME 형식으로 지정하세요

예:

```
table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
 ...
 TABLE_*N*_NAME : LEGACY_TABLE_*N*_NAME
```

> **Note**
>
> 유틸리티 응용 프로그램이 시작되면 제공된 모든 매핑을 명시적으로 기록합니다.

AWS Blu Age 엔진의 구성 속성 추가

리팩토링된 애플리케이션의 config 폴더에 파일을 추가하여 Blu Age 런타임 엔진의 새로운 기능에 액세스할 수 있습니다. AWS 이 파일 user-properties.yml의 이름을 지정해야 합니다. 이 파일은 응용 프로그램 정의를 대체하지는 않지만 확장합니다. 이 항목에서는 user-properties.yml 파일에 포함할 수 있는 속성에 대해 설명합니다.

> **Note**
>
> 일부 매개변수는 AWS 메인프레임 현대화 또는 애플리케이션 정의에 의해 제어되므로 변경할 수 없습니다. 애플리케이션의 애플리케이션 정의에 정의된 모든 파라미터는 user-properties.yml에서 지정한 파라미터보다 우선합니다.

리팩터링된 애플리케이션의 구조에 대한 자세한 내용은 AWS 블루에이지 관리 애플리케이션의 구조 단원을 참조하세요.

다음 다이어그램은 AWS Blu Age 샘플 애플리케이션의 구조 내에서 user-properties.yml 파일을 찾을 수 있는 위치를 보여줍니다. PlanetsDemo

```
PlanetsDemo-v1/
   ## config/
   #  ## application-PlanetsDemo.yml
   #  ## user-properties.yml
   ## jics/
   ## webapps/
```

구성 속성 참조

사용 가능한 속성 목록입니다. 모든 파라미터는 선택 사항입니다.

주제

- Gapwalk 애플리케이션 속성
- Gapwalk 배치 스크립트 속성
- Gapwalk Blugen 속성
- Gapwalk CL 명령 속성
- 갭워크 CL 러너 속성

- 갭워크 JHDB 속성
- 갭워크 JIC 속성
- 갭워크 런타임 속성
- 갭워크 유틸리티 프로그램 속성
- 기타 속성

Gapwalk 애플리케이션 속성

bluesam.fileLoading.commitInterval

선택 사항입니다. bluesam 커밋 간격.

유형: 숫자

기본값: 10240

카드 인코딩

선택 사항입니다. 카드 인코딩: `useControlMVariable`와 함께 사용할 수 있습니다.

타입: 문자열

기본값: CP1145

checkinputfilesize

선택 사항입니다. 파일 크기가 레코드 크기의 배수인 경우 검사를 해제할지 여부를 지정합니다.

유형: boolean

기본값: false

database.cursor.overflow.allowed

선택 사항입니다. 커서 오버플로를 허용할지 여부를 지정합니다. 커서의 위치에 상관없이 커서에서 다음 직접적 호출을 수행하도록 `true`로 설정합니다. 커서에서 다음 직접적 호출을 수행하기 전에 커서가 마지막 위치에 있는지 확인하려면 `false`로 설정합니다. 커서를 스크롤할 수 있는 경우에만 활성화(민감 또는 비민감)

유형: boolean

기본값: true

데이터 단순화. onInvalidNumeric데이터

선택 사항입니다. 유효하지 않은 숫자형 데이터를 디코딩할 때 대응하는 방법. 허용되는 값은 `reject`, `toleratespaces`, `toleratespaceslowvalues`, `toleratemost`입니다.

타입: 문자열

기본값: 거부

defaultKeepExisting파일

선택 사항입니다. 데이터세트 기본 이전 값을 설정할지 여부를 지정합니다.

유형: boolean

기본값: false

disposition.checkexistence

선택 사항입니다. DISP SHR 또는 OLD를 사용하는 데이터세트의 파일 존재 여부에 대한 검사를 해제할지 여부를 지정합니다.

유형: boolean

기본값: false

externalSort.threshold

선택 사항입니다. 정렬 임계값: 외부(병합) 정렬로 전환하는 시점.

타입: 문자열

기본값: null

```
externalSort.threshold: 12MB
```

forceHR

선택 사항입니다. 콘솔 또는 파일 출력에서 사람이 읽을 수 있는 SYSPRINT를 사용할지 여부를 지정합니다.

유형: boolean

기본값: false

forcedDate

선택 사항입니다. 데이터베이스에 특정 날짜 및 시간을 적용합니다. 개발 및 테스트 중에만 사용하세요.

기본값: null

```
forcedDate: 2022-08-26T12:59:58.123456+01:57
```

frozenDate

선택 사항입니다. 데이터베이스에 있는 레코드의 날짜 및 시간입니다. 개발 및 테스트 중에만 사용하세요.

기본값: false

```
frozenDate: false
```

ims.messages.extendedSize

선택 사항입니다. ims 메시지에 확장 크기를 설정할지 여부를 지정합니다.

유형: boolean

기본값: false

lockTimeout

선택 사항입니다. 지정된 기간 내에 잠금을 획득하지 못한 경우의 트랜잭션 제한 시간(밀리초)입니다.

유형: 숫자

기본값: 500

mapTransfo.prefixes

선택 사항입니다. controlM 변수를 변환할 때 사용할 접두사 목록. 각각 쉼표로 구분됩니다.

타입: 문자열

기본값: &,@,%%

쿼리. useConcatCondition

선택 사항입니다. 키 조건을 키 연결로 빌드할지 여부를 지정합니다.

유형: boolean

기본값: false

rollbackOnRTE

선택 사항입니다. 런타임 예외 시 암시적 실행 단위 트랜잭션을 롤백할지 여부를 지정합니다.

유형: boolean

기본값: false

sctThreadLimit

선택 사항입니다. 스크립트 트리거에 대한 스레드 한도입니다.

유형: 숫자

기본값: 5

sqlCodePoint쉬프트

선택 사항입니다. SQL 코드 포인트 시프트. 레거시 rdbms 데이터를 최신 rdbms로 마이그레이션할 때 발생할 수 있는 제어 문자의 코드포인트를 이동합니다. 예를 들어 유니코드 문자 \u0180와 일치하도록 384를 지정할 수 있습니다.

유형: 숫자

기본값: 0

sqlIntegerOverflow허용됨

선택 사항입니다. SQL 정수 오버플로를 허용할지 여부, 즉 호스트 변수에 더 큰 값을 배치할 수 있는지 여부를 지정합니다.

유형: boolean

기본값: false

stepFailWhen어벤드

선택 사항입니다. 단계가 실패하거나 실행을 완료한 경우 제한을 발생시킬지 여부를 지정합니다.

유형: boolean

기본값: true

stopExecutionWhenProgNotFound

선택 사항입니다. 프로그램을 찾을 수 없는 경우 실행을 중지할지 여부를 지정합니다. `true`로 설정하면 프로그램을 찾을 수 없는 경우 실행을 중단합니다.

유형: boolean

기본값: true

uppercaseUserInput

선택 사항입니다. 사용자 입력이 대문자여야 하는지 여부를 지정합니다.

유형: boolean

기본값: true

useControlMVariable

선택 사항입니다. 변수 대체에 control-M 사양을 사용할지 여부를 지정합니다.

유형: boolean

기본값: false

Gapwalk 배치 스크립트 속성

인코딩

선택 사항입니다. 배치 스크립트 프로젝트에서 사용되는 인코딩(groovy 제외). 유효한 인코딩 `CP1047`, `IBM930`, `ASCII`, `UTF-8`...이 필요합니다.

타입: 문자열

기본값: ASCII

Gapwalk Blugen 속성

managers.trancode

선택 사항입니다. 다이얼로그 매니저는 매핑을 트랜스코딩합니다. JICS 트랜잭션 코드를 다이얼로그 매니저에 매핑할 수 있습니다. 예상 형식은 `trancode1:dialogManager1;trancode2:dialogManager2;`입니다.

타입: 문자열

기본값: null

```
managers.trancode: OR12:MYDIALOG1
```

Gapwalk CL 명령 속성

commands-off

선택 사항입니다. 해제할 명령 목록(쉼표로 구분됨). 허용되는 값은 PGM_BASIC, RCVMSG, SNDRCVF, CHGVAR, QCLRDTAQ, RTVJOBA, ADDLFM, ADDPFM, RCVF, OVRDBF, DLTOVR, CPYF, SNDDTAQ입니다. 기존 프로그램을 사용하지 않도록 설정하거나 덮어쓰려는 경우에 유용합니다. PGM_BASIC는 디버그 목적으로 설계된 특정 속도 프로그램입니다.

타입: 문자열

기본값: null

spring.datasource.primary.jndi-name

선택 사항입니다. 기본 자바 네이밍 및 디렉터리 인터페이스(jndi) 데이터 소스.

타입: 문자열

기본값: jdbc/primary

zonedMode

선택 사항입니다. 구역화된 데이터 유형을 인코딩 또는 디코딩하기 위한 모드입니다. 허용되는 값은 EBCDIC_STRICT / EBCDIC_MODIFIED / AS400입니다.

타입: 문자열

기본값: EBCDIC_STRICT

갭워크 CL 러너 속성

cl.confiation.context.encoding

선택 사항입니다. CL 파일의 인코딩. 유효한 인코딩 CP1047, IBM930, ASCII, UTF-8...이 필요합니다.

타입: 문자열

기본값: CP297

cl.zonedMode

선택 사항입니다. 제어 언어(CL) 명령을 인코딩 또는 디코딩하기 위한 모드입니다. 허용되는 값은 `EBCDIC_STRICT` / `EBCDIC_MODIFIED` / `AS400`입니다.

타입: 문자열

기본값: EBCDIC_STRICT

갭워크 JHDB 속성

ims.programs

선택 사항입니다. 사용할 IMS 프로그램 목록. 각 파라미터는 세미콜론(`;`)으로 구분하고 각 트랜잭션은 쉼표(`,`)로 구분합니다. 예: `ims.programs: PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;`

타입: 문자열

기본값: null

jhdb.checkpointPath

선택 사항입니다. `jhdb.checkpointPersistence`이 `none`가 아닌 경우 이 파라미터를 사용하여 체크포인트 지속성 경로(checkpoint.dat 파일 저장 위치)를 설정할 수 있습니다. 레지스트리에 포함된 모든 체크포인트 데이터는 직렬화되고 제공된 폴더에 있는 파일(checkpoint.dat)에 백업됩니다. 이 백업에는 체크포인트 데이터(scriptId, stepId, 데이터베이스 위치 및 체크포인트 영역) 만 관련된다는 점에 유의하세요.

타입: 문자열

기본값: file:./setup/

jhdb.checkpointPersistence

선택 사항입니다. 체크포인트 지속성 모드. 허용되는 값은 `none` / `add` / `end`입니다. 새 체크포인트를 생성하여 레지스트리에 추가할 때 체크포인트를 유지하는 데 `add`를 사용합니다. 서버 종료 시 체크포인트를 유지하는 데 `end`를 사용합니다. 다른 값을 입력하면 지속성이 비활성화됩니다. 레지스트리에 새 체크포인트가 추가될 때마다 기존의 모든 체크포인트가 직렬화되고 파일이 지워진다

는 점에 유의하세요. 파일의 기존 데이터에 추가되지 않습니다. 따라서 체크포인트 수에 따라 성능에 어느 정도 영향을 미칠 수 있습니다.

타입: 문자열

기본값: 없음

jhdb.configuration.context.encoding

선택 사항입니다. JHDB(자바 계층적 데이터베이스) 인코딩. 유효한 인코딩 문자열 `CP1047`, `IBM930`, `ASCII`, `UTF-8`... 필요합니다.

타입: 문자열

기본값: CP297

jhdb. identificationCardData

선택 사항입니다. 일부 “운영자 ID 카드 데이터”를 CARD 파라미터로 지정된 MID 필드에 하드코딩하는 데 사용됩니다.

타입: 문자열

기본값: ""

jhdb.lterm

선택 사항입니다. IMS 에뮬레이션의 경우 공통 논리적 터미널 ID를 강제 적용할 수 있습니다. 설정하지 않으면 sessionId가 사용됩니다.

타입: 문자열

기본값: null

jhdb.metadata.extrapath

psbs 및 dbds 폴더의 런타임별 추가 루트 폴더를 지정하는 구성 파라미터입니다.

타입: 문자열

기본값: 파일:. /setup/

> **Note**
>
> 현재 배포 제약 조건의 경우 애플리케이션의 구성 디렉터리 또는 구성 디렉터리의 하위 디렉터리(예: config/setup)에 dbds 및 psbs 디렉터리를 복사해야 합니다.

```
config
|- setup
   |- dbds
   |- psbs
```

그리고 application-jhdb.yml에서 설정합니다.
`jhdb.metadata.extrapath: file: ./config/setup/`

jhdb.navigation.cacheNexts

선택 사항입니다. RDBMS의 계층적 탐색에 사용되는 캐시 기간(밀리초).

유형: 숫자

기본값: 5000

jhdb.query. limitJoinUsage

선택 사항입니다. RDBMS 그래프에서 제한 조인 사용량 파라미터를 사용할지 여부를 지정합니다.

유형: boolean

기본값: true

jhdb. use-db-prefix

선택 사항입니다. RDBMS의 계층 탐색에서 데이터베이스 접두사를 사용할지 여부를 지정합니다.

유형: boolean

기본값: true

갭워크 JIC 속성

jics.data. dataJsonInit위치

선택 사항입니다. 분석기가 CSD를 파싱하여 준비하고 jics 데이터베이스를 초기화하는 데 사용되는 json 파일의 위치,

타입: 문자열

기본값: ""

jics.db. dataScriptLocation

선택 사항입니다. Analyzer가 메인프레임에서 CSD 내보내기를 구문 분석하여 작성한 initJics.sql 스크립트의 위치.

타입: 문자열

기본값: ""

jics.db. dataTestQuery위치

선택 사항입니다. 객체 수를 반환할 것으로 예상되는 단일 SQL 쿼리를 포함하는 SQL 스크립트의 위치(예: jics 프로그램 테이블의 레코드 수 계산). 개수가 0이면 `jics.db.dataScriptLocation` 스크립트를 사용하여 데이터베이스를 로드하고 그렇지 않으면 데이터베이스 로드를 건너뛰게 됩니다.

타입: 문자열

기본값: ""

jics.db. ddlScriptLocation

선택 사항입니다. Jics dll 스크립트 위치. .sql 스크립트를 사용하여 jics 데이터베이스 스키마를 시작할 수 있습니다.

타입: 문자열

기본값: ""

```
jics.db.ddlScriptLocation: ./jics/sql/jics.sql
```

jics.db. schemaTestQuery위치

선택 사항입니다. jics 스키마의 개체 수(있는 경우)를 반환하는 고유한 쿼리를 포함해야 하는 sql 파일의 위치입니다.

타입: 문자열

기본값: ""

믹스. runUnitLauncher풀. 활성화

선택 사항입니다. JICS에서 실행 유닛 런처 풀을 활성화할지 여부를 지정합니다.

유형: boolean

기본값: false

JIC. runUnitLauncher풀 사이즈

선택 사항입니다. 실행 유닛 런처 풀 크기(JICS).

유형: 숫자

기본값: 20

지크. runUnitLauncher풀. 검증 간격

선택 사항: JICS에서 실행 장치 런처 풀의 유효성 검사 간격(밀리초 단위로 표시).

유형: 숫자

기본값: 1000

jics.queues.sqs.region

선택 사항입니다. AWS 리전 JICS에서 사용되는 Amazon SQS용입니다. 성능을 위해 배포된 애플리케이션과 동일한 지역을 설정하는 것이 좋지만 필수 사항은 아닙니다.

타입: 문자열

기본값: eu-west-1

jics.xa.agent.timeout

선택 사항입니다. 분산 트랜잭션 관리를 담당하는 xa 에이전트가 작업을 완료할 수 있는 최대 기간을 정의합니다.

유형: 숫자

기본값: null

mq.queues.sqs.region

선택 사항입니다. Amazon SQS MQ AWS 리전 서비스용입니다.

타입: 문자열

기본값: eu-west-3

태스크 실행자. allowCoreThreadTimeOut

선택 사항입니다. JCIS에서 코어 스레드의 타임아웃을 허용할지 여부를 지정합니다. 이렇게 하면 0이 아닌 대기열과 함께 사용해도 동적으로 확장 및 축소할 수 있습니다(대기열이 가득 차면 최대 풀 크기가 커지기 때문입니다).

유형: boolean

기본값: false

태스크 실행자. corePoolSize

선택 사항입니다. groovy 스크립트를 통해 터미널에서 트랜잭션을 시작하면 새 스레드가 생성됩니다. 이 파라미터를 사용하여 코어 풀 크기를 설정합니다.

유형: 숫자

기본값: 5

태스크 실행자. maxPoolSize

선택 사항입니다. 멋진 스크립트를 통해 터미널에서 트랜잭션을 시작하면 새 헤드가 생성됩니다. 이 파라미터를 사용하여 최대 풀 크기(최대 병렬 스레드 수) 를 설정합니다.

유형: 숫자

기본값: 10

taskExecutor.queueCapacity

선택 사항입니다. 멋진 스크립트를 통해 터미널에서 트랜잭션을 시작하면 새 헤드가 생성됩니다. 이 파라미터를 사용하여 큐 크기를 설정합니다.(= `taskExecutor.maxPoolSize` 도달 시 보류 중인 트랜잭션의 최대 수)

유형: 숫자

기본값: 50

갭워크 런타임 속성

cacheMetadata

선택 사항입니다. 데이터베이스 메타데이터를 캐시할지 여부를 지정합니다.

유형: boolean

기본값: true

check-groovy-file

선택 사항입니다. 등록하기 전에 groovy 파일 내용을 확인할지 여부를 지정합니다.

유형: boolean

기본값: true

databaseStatistics

선택 사항입니다. SQL 빌더가 통계 정보를 수집하고 표시할 수 있도록 허용할지 여부를 지정합니다.

유형: boolean

기본값: false

dateTimeFormat

선택 사항입니다. 에서는 데이터베이스 날짜 타임스탬프 유형을 데이터 단순화 엔티티로 넘기는 방법을 dateTimeFormat 설명합니다. 허용되는 값은 `ISO` / `EUR` / `USA` / `LOCAL`입니다

타입: 문자열

기본값: ISO

dbDateFormat

선택 사항입니다. 데이터베이스 대상 날짜 형식.

타입: 문자열

기본값: yyyy-MM-dd

dbTimeFormat

선택 사항입니다. 데이터베이스 대상 시간 형식.

타입: 문자열

기본값: HH:mm:ss

dbTimestampFormat

선택 사항입니다. 데이터베이스 대상 타임스탬프 형식.

타입: 문자열

기본값: yyyy-MM-dd HH:mm:ss.SSSSSS

fetchSize

선택 사항입니다. 커서의 fetchSize 값입니다. 로드/언로드 유틸리티로 청크를 사용하여 데이터를 가져올 때 사용합니다.

유형: 숫자

기본값: 10

SQL 강제 사용 안 함 TrimStringType

선택 사항입니다. 모든 SQL 문자열 파라미터의 트림을 비활성화할지 여부를 지정합니다.

유형: boolean

기본값: false

localDateFormat

선택 사항입니다. 현지 날짜 형식 목록. 각 형식을 |로 구분합니다.

타입: 문자열

localTimeFormat

선택 사항입니다. 현지 시간 형식 목록. 각 형식을 |로 구분합니다.

타입: 문자열

localTimestampFormat

선택 사항입니다. 로컬 타임스탬프 형식 목록. 각 형식을 |로 구분합니다.

타입: 문자열

기본값:

pgmDateFormat

선택 사항입니다. 프로그램에서 사용되는 날짜/시간 형식입니다.

타입: 문자열

기본값: yyyy-MM-dd

pgmTimeFormat

선택 사항입니다. pgm(프로그램) 실행에 사용되는 시간 형식입니다.

타입: 문자열

기본값: HH.mm.ss

pgmTimestampFormat

선택 사항입니다. 타임스탬프 형식.

타입: 문자열

기본값: yyyy-MM-dd-HH.mm.ss.SSSSSS

갭워크 유틸리티 프로그램 속성

jcl.type

선택사항. `.jcl`파일 유형. 허용되는 값은 `jcl` / `vse`입니다. IDCAMS 유틸리티 PRINT/REPRO 명령은 vse jcl이 아닌 경우 파일이 비어 있는 경우 4를 반환합니다.

타입: 문자열

기본값: mvs

listcat.variablelengthpreprocessor.enabled

선택 사항입니다. LISTCAT 명령에 대한 가변 길이 프리프로세서를 활성화할지 여부를 지정합니다.

유형: boolean

기본값: false

listcat.variablelengthpreprocessor.type

선택 사항입니다. listcat 파일에 포함된 객체 유형 (`listcat.variablelengthpreprocessor.enabled`를 활성화한 경우). 허용되는 값은 `rdw` / `bdw`입니다.

타입: 문자열

기본값: rdw

로드. 배치 크기

선택 사항입니다. 로드 유틸리티 배치 크기.

유형: 숫자

기본값: 0

load.format.dbDate

선택 사항입니다. 사용할 로드 유틸리티 데이터베이스 형식입니다.

타입: 문자열

기본값: yyyy-MM-dd

load.format.dbTime

선택 사항입니다. 로드 유틸리티 데이터베이스를 사용할 시간입니다.

타입: 문자열

기본값: HH:mm:ss

load.format.localDate

선택 사항입니다. 사용할 로드 유틸리티 로컬 날짜 형식입니다.

타입: 문자열

기본값: dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd

load.format.localTime

선택 사항입니다. 사용할 로드 유틸리티 현지 시간 형식입니다.

타입: 문자열

기본값: HH:mm:ss|HH.mm.ss

로드. sqlCodePoint시프트

선택 사항입니다. 로드 유틸리티의 SQL 코드 포인트 이동. 문자 이동 프로세스를 실행합니다. DB2의 대상 데이터베이스가 PostgreSQL인 경우 필요합니다.

유형: 숫자

기본값: 0

sysPunchEncoding

선택 사항입니다. syspunch 인코딩 문자 세트. 지원되는 값은 `Cp1047` / `ASCII`과 같습니다.

타입: 문자열

기본값: ASCII

treatLargeNumberAsInteger

선택 사항입니다. 큰 숫자를 `Integer`로 처리할지 여부를 지정합니다. 기본적으로 `BigDecimal`와 같이 취급됩니다.

유형: boolean

기본값: false

unload.chunkSize

선택 사항입니다. 언로드 유틸리티에 사용되는 청크 크기입니다.

유형: 숫자

기본값: 0

unload.columnFiller

선택 사항입니다. 언로드 유틸리티 열 필러.

타입: 문자열

기본값: 공백을

unload.fetchSize

선택 사항입니다. 언로드 유틸리티에서 커서를 처리할 때 페치 크기를 조정할 수 있습니다.

유형: 숫자

기본값: 0

unload.format.date

선택 사항입니다. `unload.useDatabaseConfiguration`가 활성화된 경우, 언로드 유틸리티에서 사용할 날짜 형식입니다. 자세한 내용은 unload.format.date를 참조하세요.

타입: 문자열

기본값: MM/dd/yyyy

unload.format.time

선택 사항입니다. `unload.useDatabaseConfiguration`가 활성화된 경우, 언로드 유틸리티에서 사용할 시간 형식입니다.

타입: 문자열

기본값: HH.mm.ss

unload.format.timestamp

선택 사항입니다. `unload.useDatabaseConfiguration`가 활성화된 경우, 언로드 유틸리티에서 사용할 타임스탬프 형식입니다.

타입: 문자열

기본값: yyyy-MM-dd-HH.mm.ss.SSSSSS

.nbi. 언로드하세요. whenNotNull

선택 사항입니다. 데이터베이스의 값이 null이 아닐 때 추가할 Null 바이트 표시기(nbi) 값입니다.

유형: 16진수

기본값: 00

unload.nbi.whenNull

선택 사항입니다. 데이터베이스의 값이 null일 때 추가할 Null 바이트 표시기(nbi) 값입니다.

유형: 16진수

기본값: 6F

.nbi를 언로드하십시오. writeNullIndicator

선택 사항입니다. 언로드 출력 파일에 널 인디케이터를 내보낼지 여부를 지정합니다.

유형: boolean

기본값: false

내리다. sqlCodePoint시프트

선택 사항입니다. 언로드 유틸리티의 SQL 코드 포인트 시프트. 문자 이동 프로세스를 실행합니다. DB2의 대상 데이터베이스가 PostgreSQL인 경우 필요합니다.

유형: 숫자

기본값: 0

언로드. useDatabaseConfiguration

선택 사항입니다. 언로드 유틸리티에서 application-main.yml의 날짜 또는 시간 구성을 사용할지 여부를 지정합니다.

유형: boolean

기본값: false

내리다. varCharIs널

선택 사항입니다. INFTILB 프로그램에서 이 파라미터를 사용합니다. 이 파라미터를 설정하면 빈(공백) 값이 있는 `true` null을 허용하지 않는 모든 필드는 빈 문자열을 반환합니다.

유형: boolean

기본값: false

기타 속성

qtemp.cleanup.threshold.hours

선택 사항입니다. `qtemp.dblog` 활성화 시기를 지정합니다. db 파티션 수명(시간).

유형: 숫자

기본값: 0

qtemp.dblog

선택 사항입니다. QTEMP 데이터베이스 로깅 활성화 여부.

유형: boolean

기본값: false

qtemp.uuid.length

선택 사항입니다. QTEMP 고유 번호는 길이입니다.

유형: 숫자

기본값: 9

퀴츠 스케줄러. stand-by-if-error

선택 사항입니다. 작업 스케줄러가 대기 모드인 경우 작업 실행을 트리거할지 여부를 지정합니다. true인 경우 활성화된 경우 작업 실행이 트리거되지 않습니다.

유형: boolean

기본값: false

warmUpCache

선택 사항입니다. 서버 시작 시 모든 데이터 통신 테이블 데이터를 워밍업 캐시에 로드할지 여부를 지정합니다.

유형: boolean

기본값: false

AWS 메인프레임 현대화 애플리케이션 정의 레퍼런스

AWS 메인프레임 현대화에서는 선택한 런타임 엔진에 맞는 애플리케이션 정의 JSON 파일에서 마이그레이션된 메인프레임 애플리케이션을 구성합니다. 애플리케이션 정의에는 일반 정보와 엔진별 정보가 모두 포함됩니다. 이 항목에서는 AWS Blu Age 및 Micro Focus 애플리케이션 정의를 모두 설명하고 모든 필수 및 선택적 요소를 식별합니다.

주제

- 정의 섹션 개요
- AWS 블루 에이지 애플리케이션 정의 샘플
- AWS 블루 에이지 정의 세부 정보
- Micro Focus 애플리케이션 정의
- Micro Focus 정의 세부 정보

일반 헤더 섹션

각 애플리케이션 정의는 템플릿 버전 및 소스 위치에 대한 일반 정보로 시작됩니다. 애플리케이션 정의의 현재 버전은 2.0입니다. 버전 1은 여전히 작동하지만 지원 중단될 예정입니다. 애플리케이션을 생성하거나 업데이트할 때는 버전 2를 사용하는 것이 좋습니다.

다음 구조를 사용하여 템플릿 버전과 소스 위치를 지정합니다.

```
"template-version": "2.0",
    "source-locations": [
        {
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket-aaa",
                "s3-key-prefix": "v1"
            }
        }
    ]
```

Note

S3 ARN을 s3-버킷으로 입력하려는 경우 다음 구문을 사용할 수 있습니다.

```
"template-version": "2.0",
  "source-locations": [
      {
          "source-id": "s3-source",
          "source-type": "s3",
          "properties": {
              "s3-bucket": "arn:aws:s3:::mainframe-deployment-bucket-aaa",
              "s3-key-prefix": "v1"
          }
```

```
        }
    ]
```

template-version

필수 사항입니다. 애플리케이션 정의 파일의 버전을 지정합니다. 이 값은 변경하지 마세요. 현재, 유일하게 허용되는 값은 2.0입니다. 문자열을 사용하여 `template-version`를 지정합니다.

source-locations

런타임 중에 응용 프로그램에 필요한 파일 및 기타 리소스의 위치를 지정합니다.

속성

소스 위치의 세부 정보를 제공합니다. 각 속성은 문자열로 지정됩니다.

- `s3-bucket` - 필수입니다. 원본 파일이 저장된 Amazon S3 버킷 이름입니다.
- `s3-key-prefix` - 필수입니다. 파일이 저장되는 Amazon S3 버킷의 폴더 이름을 지정합니다.

> **Note**
>
> 버킷 ARN이 아닌 Amazon S3 버킷 이름을 지정해야 합니다. 버킷 내 리소스의 절대 경로를 지정하지 마세요.

정의 섹션 개요

애플리케이션을 실행하는 데 필요한 서비스, 설정, 데이터 및 기타 일반 리소스의 리소스 정의를 지정합니다. 애플리케이션 정의를 업데이트하면 AWS 메인프레임 현대화는 애플리케이션 정의 JSON 파일의 이전 버전과 현재 버전 모두에 있는 `source-locations` 및 `definition` 목록을 비교하여 변경 사항을 감지합니다.

정의 섹션은 엔진별로 다르며 변경될 수도 있습니다. 다음 섹션에서는 두 엔진에 대한 샘플 엔진별 애플리케이션 정의를 보여줍니다.

AWS 블루 에이지 애플리케이션 정의 샘플

```
{
 "template-version": "2.0",
 "source-locations": [
        {
```

```
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket-aaa",
                "s3-key-prefix": "v1"
            }
        }
    ],
    "definition" : {
        "listeners": [{
            "port": 8194,
            "type": "http"
        }],
        "ba-application": {
            "app-location": "${s3-source}/murachs-v6/"
        },
        "blusam": {
            "db": {
                "nb-threads": 8,
                "batch-size": 10000,
                "name": "blusam",
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:blusam-FfmXLG"
            },
            "redis": {
                "hostname": "blusam.c3geul.ng.0001.usw2.cache.amazonaws.com",
                "port": 6379,
                "useSsl": true,
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:bluesamredis-nioefm"
            }
        }
    }
}
```

AWS 블루 에이지 정의 세부 정보

리스너 - 필수

AWS 메인프레임 현대화가 만든 Elastic Load Balancing을 통해 애플리케이션에 액세스하는 데 사용할 포트를 지정합니다. 다음 구조를 사용합니다.

```
"listeners": [{
```

```
            "port": 8194,
            "type": "http"
        }],
```

포트

필수 사항입니다. 잘 알려진 포트(0~1023)를 제외하고 사용 가능한 모든 포트를 사용할 수 있습니다. 8192 ~ 8199 사이의 값을 사용하는 것이 좋습니다. 이 포트에서 작동하는 다른 리스너나 애플리케이션이 없는지 확인하세요.

유형

필수 사항입니다. 현재 `http`만 지원됩니다.

AWS 블루 에이지 애플리케이션 - 필수

다음 구조를 사용하여 엔진이 애플리케이션 이미지 파일을 픽업하는 위치를 지정합니다.

```
"ba-application": {
            "app-location": "${s3-source}/murachs-v6/",
            "files-directory": "/m2/mount/myfolder",
            "enable-jics": <true|false>,
            "shared-app-location": "${s3-source}/shared/"
        },
```

app-location

애플리케이션 이미지 파일이 저장되는 Amazon S3의 특정 위치입니다.

app-location

선택 사항. 배치의 입력/출력 파일 위치. 환경 수준에서 Amazon EFS 또는 Amazon FSx 마운트 포인트 설정의 하위 폴더여야 합니다.

enable-jics

선택 사항입니다. JICS의 활성화 여부를 지정합니다. 기본값은 true입니다. 이 값을 false로 설정하면 JICS 데이터베이스가 생성되지 않습니다.

shared-app-location

선택 사항입니다. 공유 애플리케이션 요소가 저장되는 Amazon S3의 추가 위치입니다. 앱 위치와 같은 종류의 애플리케이션 구조를 포함할 수 있습니다.

BluSam - 선택 사항

다음 구조를 사용하여 BluSAM 데이터베이스와 Redis 캐시를 지정하세요.

```
"blusam": {
            "db": {
                "nb-threads": 8,
                "batch-size": 10000,
                "name": "blusam",
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:blusam-FfmXLG"
            },
            "redis": {
                "hostname": "blusam.c3geul.ng.0001.usw2.cache.amazonaws.com",
                "port": 6379,
                "useSsl": true,
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:bluesamredis-nioefm"
            }
        }
```

db

애플리케이션과 함께 사용되는 데이터베이스의 속성을 지정합니다. 데이터베이스는 Aurora PostgreSQL 데이터베이스여야 합니다. 다음과 같은 옵션을 지정할 수 있습니다.

- `nb-threads` – 선택 사항. blusam 엔진이 사용하는 쓰기 숨김 메커니즘에 사용되는 전용 스레드 수를 지정합니다. 기본값은 8입니다.
- `batch-size` – 선택 사항. 쓰기 숨김 메커니즘이 일괄 저장 작업을 시작하는 데 사용하는 임계값을 지정합니다. 임계값은 수정된 레코드가 지속되도록 일괄 저장 작업을 시작하는 수정된 레코드의 수를 나타냅니다. 트리거 자체는 배치 크기와 1초의 경과 시간(둘 중 먼저 도달하는 시점)의 조합을 기반으로 합니다. 기본값은 10000입니다.
- `name` – 선택 사항. 데이터베이스의 이름을 지정합니다.
- `secret-manager-arn` - 데이터베이스 자격 증명이 포함된 암호의 Amazon 리소스 이름(ARN)을 지정합니다. 자세한 설명은 4단계: AWS Secrets Manager 데이터베이스 시크릿 생성 및 구성 섹션을 참조하세요.

redis

애플리케이션이 성능 개선을 위해 필요한 임시 데이터를 중앙 위치에 저장하는 데 사용하는 Redis 캐시의 속성을 지정합니다. Redis 캐시를 암호화하고 암호로 보호하는 것이 좋습니다.

- hostname - Redis 캐시의 위치를 지정합니다.
- port - Redis 캐시가 통신을 보내고 받는 포트(일반적으로 6379)를 지정합니다.
- useSsl - Redis 캐시의 암호화 여부를 지정합니다. 캐시가 암호화되지 않은 경우 useSsl false로 설정합니다.
- secret-manager-arn - Redis 캐시 암호가 포함된 보안 암호의 Amazon 리소스 이름(ARN)을 지정합니다. Redis 캐시가 암호로 보호되지 않는 경우 secret-manager-arn를 지정하지 마세요. 자세한 설명은 4단계: AWS Secrets Manager 데이터베이스 시크릿 생성 및 구성 섹션을 참조하세요.

AWS 블루에이지 메시지 대기열 - 선택 사항

블루 에이지 애플리케이션에 대한 AWS JMS-MQ 연결 세부 정보를 지정합니다.

```
"message-queues": [
  {
      "product-type": "JMS-MQ",
      "queue-manager": "QMgr1",
      "channel": "mqChannel1",
      "hostname": "mqserver-host1",
      "port": 1414,
      "user-id": "app-user1",
      "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:sample/mq/test-279PTa"
  },
  {
      "product-type": "JMS-MQ",
      "queue-manager": "QMgr2",
      "channel": "mqChannel2",
      "hostname": "mqserver-host2",
      "port": 1412,
      "user-id": "app-user2",
      "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:sample/mq/test-279PTa"
  }
]
```

product-type

필수 사항입니다. 제품 유형을 지정합니다. 현재는 Blu Age 애플리케이션의 경우 “JMS-MQ”만 사용할 AWS 수 있습니다.

queue-manager

필수 사항입니다. 큐 매니저의 이름을 지정합니다.

채널

필수 사항입니다. 서버 연결 채널의 이름을 지정합니다.

hostname

필수 사항입니다. 메시지 큐 서버의 호스트 이름을 지정합니다.

포트

필수 사항입니다. 서버가 수신 중인 리스너 포트 번호를 지정합니다.

user-id

선택 사항입니다. 지정된 채널에서 메시지 큐 작업을 수행할 수 있는 사용자 계정 ID를 지정합니다.

secret-manager-arn

선택 사항입니다. 지정된 사용자의 비밀번호를 제공하는 Secrets Manager의 Amazon 리소스 이름(ARN)을 지정합니다.

Micro Focus 애플리케이션 정의

다음 샘플 정의 섹션은 Micro Focus 런타임 엔진용이며 필수 요소와 옵션 요소를 모두 포함합니다.

```
{
 "template-version": "2.0",
 "source-locations": [
        {
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket-aaa",
                "s3-key-prefix": "v1"
            }
        }
    ],
    "definition" : {
        "listeners": [{
            "port": 5101,
            "type": "tn3270"
```

```
        }],
        "dataset-location": {
            "db-locations": [{
                "name": "Database1",
                "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456"
            }]
        },
        "cognito-auth-handler": {
            "user-pool-id": "cognito-idp.us-west-2.amazonaws.com/us-west-2_rvYFnQIxL",
            "client-id": "58k05jb8grukjjsudm5hhn1v87",
            "identity-pool-id": "us-west-2:64464b12-0bfb-4dea-ab35-5c22c6c245f6"
        },
        "ldap-ad-auth-handler": {
            "ldap-ad-connection-secrets": [LIST OF AD-SECRETS]
        },
        "batch-settings": {
            "initiators": [{
                "classes": ["A", "B"],
                "description": "initiator...."
            }],
            "jcl-file-location": "${s3-source}/batch/jcl"
        },
        "cics-settings": {
            "binary-file-location": "${s3-source}/cics/binaries",
            "csd-file-location": "${s3-source}/cics/def",
            "system-initialization-table": "BNKCICV"
        },
        "xa-resources" : [{
            "name": "XASQL",
            "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456",
            "module": "${s3-source}/xa/ESPGSQLXA64.so"
        }]
    }
}
```

Micro Focus 정의 세부 정보

Micro Focus 애플리케이션 정의 파일의 정의 섹션 내용은 마이그레이션된 메인프레임 애플리케이션에 런타임에 필요한 리소스에 따라 다릅니다.

리스너 - 필수

다음 구조를 사용하여 리스너를 지정합니다.

```
"listeners": [{
    "port": 5101,
    "type": "tn3270"
}],
```

포트

tn3270의 경우 기본값은 5101입니다. 다른 유형의 서비스 리스너의 경우 포트가 다릅니다. 잘 알려진 포트(0~1023)를 제외하고 사용 가능한 모든 포트를 사용할 수 있습니다. 각 리스너에는 고유한 포트가 있어야 합니다. 리스너는 포트를 공유해서는 안 됩니다. 자세한 내용은 Micro Focus 엔터프라이즈 서버 설명서의 리스너 제어를 참조하세요.

유형

서비스 리스너 유형을 지정합니다. 자세한 내용은 Microsoft 엔터프라이즈 서버 설명서에서 리스너를 참조하세요.

데이터 세트 위치 - 필수

다음 구조를 사용하여 데이터 세트 위치를 지정합니다.

```
"dataset-location": {
            "db-locations": [{
                "name": "Database1",
                "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456"
            }],
 }
```

db-locations

마이그레이션된 애플리케이션이 생성하는 데이터 세트의 위치를 지정합니다. 현재 AWS 메인프레임 현대화는 단일 VSAM 데이터베이스의 데이터 세트만 지원합니다.

- `name` - 마이그레이션된 애플리케이션에서 생성하는 데이터 세트를 포함하는 데이터베이스 인스턴스의 이름을 지정합니다.
- `secret-manager-arn` - 데이터베이스 자격 증명이 포함된 암호의 Amazon 리소스 이름(ARN)을 지정합니다.

Amazon Cognito 인증 및 권한 부여 핸들러 - 선택 사항

AWS 메인프레임 현대화는 마이그레이션된 애플리케이션의 인증 및 권한 부여에 Amazon Cognito를 사용합니다. 다음 구조를 사용하여 Amazon Cognito 인증 핸들러를 지정합니다.

```
"cognito-auth-handler": {
            "user-pool-id": "cognito-idp.Region.amazonaws.com/Region_rvYFnQIxL",
            "client-id": "58k05jb8grukjjsudm5hhn1v87",
            "identity-pool-id": "Region:64464b12-0bfb-4dea-ab35-5c22c6c245f6"
}
```

user-pool-id

AWS 메인프레임 현대화가 마이그레이션된 애플리케이션의 사용자를 인증하는 데 사용하는 Amazon Cognito 사용자 풀을 지정합니다. 사용자 풀의 AWS 리전 사용자 풀은 메인프레임 현대화 애플리케이션의 사용자 풀과 일치해야 합니다. AWS 리전 AWS

client-id

인증된 사용자가 액세스할 수 있는 마이그레이션된 애플리케이션을 지정합니다.

identity-pool-id

인증된 사용자가 메인프레임 현대화에 액세스할 AWS 수 있는 자격 증명으로 사용자 풀 토큰을 교환하는 Amazon Cognito 자격 증명 풀을 지정합니다. 자격 증명 풀의 AWS 리전 용도는 메인프레임 현대화 애플리케이션의 자격 증명 풀과 일치해야 합니다 AWS 리전 . AWS

LDAP 및 Active Directory 핸들러 - 선택 사항

애플리케이션을 Active Directory(AD) 또는 모든 유형의 LDAP 서버와 통합하여 애플리케이션 사용자가 권한 부여 및 인증에 LDAP/AD 보안 인증 정보를 사용하도록 할 수 있습니다.

애플리케이션을 AD와 통합하려면

1. Micro Focus Enterprise Server 설명서의 Enterprise Server 보안을 위한 Active Directory 구성에 설명된 단계를 따릅니다.
2. 애플리케이션에 사용할 각 AD/LDAP 서버의 AD/LDAP 세부 정보가 포함된 AWS Secrets Manager 시크릿을 생성하십시오. 암호를 생성하는 방법에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 AWS Secrets Manager 암호 생성을 참조하십시오. 암호 유형에서 기타 유형의 암호를 선택하고 다음 키-값 페어를 포함합니다.

```
{
    "connectionPath"       : "<HOST-ADDRESS>:<PORT>",
    "authorizedId"         : "<USER-FULL-DN>",
    "password"             : "<PASSWORD>",
    "baseDn"               : "<BASE-FULL-DN>",
    "userClassDn"          : "<USER-TYPE>",
    "userContainerDn"      : "<USER-CONTAINER-DN>",
    "groupContainerDn"     : "<GROUP-CONTAINER-DN>",
    "resourceContainerDn"  : "<RESOURCE-CONTAINER-DN>"
}
```

⚠ 보안 권장 사항

- 의 경우connectionPath, AWS 메인프레임 현대화는 LDAP 및 LDAP를 통한 LDAP (LDAPS) 프로토콜을 지원합니다. LDAPS는 더 안전하며 네트워크 전송 시 보안 인증 정보가 드러나지 않도록 하므로, LDAPS를 사용하는 것이 좋습니다.
- authorizedId 및 password의 경우 애플리케이션 실행에 필요한 가장 제한적인 읽기 전용 및 확인 권한보다 많은 권한이 없는 사용자의 보안 인증 정보를 지정하는 것이 좋습니다.
- AD/LDAP 보안 인증 정보를 정기적으로 교체하는 것이 좋습니다.
- 사용자 이름 awsuser 또는 mfuser로 AD 사용자를 생성하지 마세요. 이 두 사용자 이름은 AWS 사용에 예약되어 있습니다.

다음은 예입니다.

```
{
    "connectionPath" : "ldaps://msad4.m2.example.people.aws.dev:636",
    "authorizedId" :
 "CN=LDAPUser,OU=Users,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "password" : "ADPassword",
    "userContainerDn" : "CN=Enterprise Server Users,CN=Micro Focus,CN=Program
 Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "groupContainerDn" : "CN=Enterprise Server Groups,CN=Micro Focus,CN=Program
 Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "resourceContainerDn" : "CN=Enterprise Server Resources,CN=Micro
 Focus,CN=Program Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev"
```

```
}
```

고객 관리형 KMS 키로 암호를 생성합니다. AWS 메인프레임 현대화에 암호에 대한 `GetSecretValue` 및 `DescribeSecret` 권한, KMS 키에 대한 권한을 부여해야 합니다. `Decrypt` `DescribeKey` 자세한 내용은 사용 설명서의 KMS 키 권한을 참조하십시오. AWS Secrets Manager

3. 애플리케이션 정의에 다음을 추가합니다.

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": [LIST OF AD/LDAP SECRETS]
}
```

다음은 예입니다.

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": ["arn:aws:secrets:1234:us-east-1:secret:123456"]
}
```

LDAP/AD 인증 핸들러는 Micro Focus 8.0.11 이상 버전에서 사용할 수 있습니다.

Batch 설정 - 필수

다음 구조를 사용하여 응용 프로그램의 일부로 실행되는 배치 작업에 필요한 세부 정보를 지정합니다.

```
"batch-settings": {
            "initiators": [{
                "classes": ["A","B"],
                "description": "initiator...."
            }],
            "jcl-file-location": "${s3-source}/batch/jcls"
 }
```

시작한 사용자

마이그레이션된 응용 프로그램이 성공적으로 시작될 때 시작되고 응용 프로그램이 중지될 때까지 계속 실행되는 배치 이니시에이터를 지정합니다. 시작한 사용자당 하나 이상의 클래스를 정의할 수 있습니다. 여러 시작한 사용자를 정의할 수도 있습니다. 예:

```
"batch-settings": {
```

```
            "initiators": [
                {
                  "classes": ["A", "B"],
                  "description": "initiator...."
                },
                {
                  "classes": ["C", "D"],
                  "description": "initiator...."
                }
            ],
            "jcl-file-location": "${s3-source}/batch/jcls"
}
```

자세한 내용은 Micro Focus 엔터프라이즈 서버 설명서의 배치 시작한 사용자 또는 프린터 SEP 정의하기를 참조하세요.

- `classes` - 시작한 사용자가 실행할 수 있는 작업 클래스를 지정합니다. 최대 36자의 유니코드 문자를 사용할 수 있습니다. A-Z 또는 0-9와 같은 문자를 사용할 수 있습니다.
- `description`- 시작한 사용자의 용도를 설명합니다.
- `jcl-file-location`- 마이그레이션된 응용 프로그램이 실행하는 배치 작업에 필요한 JCL 파일의 위치를 지정합니다.

CICS 설정 - 필수

다음 구조를 사용하여 응용 프로그램의 일부로 실행되는 CICS 트랜잭션에 필요한 세부 정보를 지정합니다.

```
"cics-settings": {
            "binary-file-location": "${s3-source}/cics/binaries",
            "csd-file-location": "${s3-source}/cics/def",
            "system-initialization-table": "BNKCICV"
        }
```

binary-file-location

CICS 트랜잭션 프로그램 파일의 위치를 지정합니다.

csd-file-location

이 응용 프로그램의 CICS 리소스 정의(CSD) 파일 위치를 지정합니다. 자세한 내용은 Micro Focus 엔터프라이즈 서버 설명서의 CICS 리소스 정의를 참조하세요.

system-initialization-table

마이그레이션된 애플리케이션이 사용하는 시스템 초기화 테이블(SIT)을 지정합니다. SIT 테이블 이름은 최대 8자까지 가능합니다. A~Z, 0~9, $, @ 및 #을 사용할 수 있습니다. 자세한 내용은 Micro Focus 엔터프라이즈 서버 설명서의 CICS 리소스 정의를 참조하세요.

XA 리소스 - 필수

다음 구조를 사용하여 애플리케이션에 필요한 XA 리소스에 필요한 세부 정보를 지정하세요.

```
"xa-resources" : [{
            "name": "XASQL",
            "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456",
            "module": "${s3-source}/xa/ESPGSQLXA64.so"
        }]
```

name

필수 사항입니다. XA 리소스 공유의 이름을 지정합니다.

secret-manager-arn

데이터베이스에 연결하기 위한 자격 증명이 포함된 보안 암호의 Amazon 리소스 이름(ARN)을 지정합니다.

모듈

RM 스위치 모듈 실행 파일의 위치를 지정합니다. 자세한 내용은 Micro Focus 엔터프라이즈 서버 설명서의 XAR 계획 및 설계를 참조하세요.

AWS 메인프레임 현대화 데이터 세트 정의 참조

애플리케이션에 처리에 필요한 데이터 세트가 몇 개 이상인 경우 AWS 메인프레임 현대화 콘솔에 데이터 세트를 하나씩 입력하는 것은 비효율적입니다. 대신 JSON 파일을 생성하여 각 데이터 세트를 지정하는 것이 좋습니다. 많은 파라미터가 공통적으로 사용되기는 하지만 JSON에서는 데이터 세트 유형이 다르게 지정됩니다. 이 문서에서는 다양한 유형의 데이터 세트를 가져오는 데 필요한 JSON의 세부 정보를 설명합니다.

> **Note**
>
> 데이터 세트를 가져오려면 먼저 메인프레임에서 AWS으로 데이터 세트를 전송해야 합니다. 그런 다음 데이터 세트를 메인프레임 형식에서 AWS가 사용할 수 있는 형식으로 변환했는지 확인해야 합니다. 필요한 경우 필요에 따라 데이터를 변환하고 변환된 데이터 세트를 Amazon S3에 저장합니다. 데이터 세트 정의 JSON 파일에 버킷 및 폴더 이름을 지정합니다.
> Micro Focus 런타임 엔진을 사용하는 경우 DFCONV 유틸리티를 사용하여 데이터 세트를 변환할 수 있습니다. Micro Focus 엔터프라이즈 개발자 및 엔터프라이즈 서버 이미지에 이 유틸리티가 포함되어 있습니다. 자세한 내용은 Micro Focus 엔터프라이즈 개발자 설명서의 DFCONV 배치 파일 변환을 참조하세요.

주제

공통 특성

몇 가지 파라미터는 모든 데이터 세트에 공통됩니다. 이러한 매개변수는 다음 영역을 다룹니다.

- 데이터 세트에 대한 정보(`datasetName`, `datasetOrg`, `recordLength`, `encoding`)
- 가져오려는 위치, 즉 데이터 세트의 소스 위치에 대한 정보. 이 위치는 메인프레임 상의 위치가 아닙니다. 이 경로에서 데이터 세트를 업로드한 Amazon S3 위치(`externalLocation`)입니다.
- 가져오려는 위치, 즉 데이터 세트의 대상 위치에 대한 정보. 이 위치는 런타임 엔진에 따라 데이터베이스 또는 파일 시스템입니다. (`storageType` 및 `relativePath`).
- 데이터 세트 유형에 대한 정보(특정 데이터 세트 유형, 형식, 인코딩 등).

각 데이터 세트 정의에는 동일한 JSON 구조가 있습니다. 다음 예제 JSON은 이러한 공통 파라미터를 모두 보여줍니다.

```
{
    "dataSet": {
```

```
        "storageType": "Database",
        "datasetName": "MFI01V.MFIDEMO.BNKACC",
        "relativePath": "DATA",
        "datasetOrg": {
            "type": {
                type-specific properties
                    ...
            },
        },
    },
}
```

다음은 모든 작업에 공통적인 속성입니다.

storageType

필수 사항입니다. 대상 위치에 적용됩니다. 데이터 세트를 데이터베이스에 저장할지 파일 시스템에 저장할지 지정합니다. 가능한 값은 `Database` 또는 `FileSystem`입니다.

- AWS Blu Age 런타임 엔진: 파일 시스템은 지원되지 않습니다. 데이터베이스를 사용해야 합니다.
- Micro Focus 런타임 엔진: 데이터베이스와 파일 시스템이 모두 지원됩니다. 데이터베이스에는 Amazon 관계형 데이터베이스 서비스 또는 Amazon Aurora를, 파일 시스템에는 Amazon Elastic File System 또는 Amazon FSx for Lustre를 사용할 수 있습니다.

datasetName

필수 사항입니다. 메인프레임에 표시되는 데이터 세트의 정규화된 이름을 지정합니다.

relativePath

필수 사항입니다. 대상 위치에 적용됩니다. 데이터베이스 또는 파일 시스템에서 데이터 세트의 상대적 위치를 지정합니다.

datasetOrg

필수 사항입니다. 데이터 세트 유형을 지정합니다. 가능한 값은 `vsam`, `gdg`, `ps`, `po` 또는 `unknown`입니다.

- AWS Blu Age 런타임 엔진: VSAM 유형 데이터 세트만 지원됩니다.
- Micro Focus 런타임 엔진: VSAM, GDG, PS, PO 또는 알 수 없는 유형의 데이터 세트가 지원됩니다.

> **Note**
>
> 애플리케이션에 COBOL 데이터 파일은 아니지만 PDF 또는 기타 바이너리 파일인 파일이 필요한 경우 다음과 같이 지정할 수 있습니다.
>
> ```
> "datasetOrg": {
>             "type": PS {
>                 "format": U
>             },
> ```

VSAM용 샘플 데이터 세트 요청 형식

- AWS Blu Age 런타임 엔진: 지원됩니다.
- Micro Focus 런타임 엔진: 지원됩니다.

VSAM 데이터 세트를 가져오는 경우 `datasetOrg`로 `vsam`를 지정하세요. JSON은 다음과 같을 것입니다.

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.VSAM.KSDS",
    "relativePath": "DATA",
    "datasetOrg": {
        "vsam": {

            "encoding": "A",
            "format": "KS",
            "primaryKey": {
                "length": 11,
                "offset": 0
            }
        }
    },
    "recordLength": {
        "min": 300,
        "max": 300
    }
```

```
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/catalog/data/AWS.M2.VSAM.KSDS.DAT"
}
```

VSAM 데이터 세트에는 다음과 같은 속성이 지원됩니다.

인코딩

필수 사항입니다. 데이터 세트의 문자 세트 인코딩을 지정합니다. 가능한 값은 ASCII(A), EBCDIC(E) 및 알 수 없음(?)입니다.

형식

필수 사항입니다. VSAM 데이터 세트 유형과 레코드 형식을 지정합니다.

- AWS Blu Age 런타임 엔진: 가능한 값은 ESDS(ES), KSDS(KS) 및 RRDS(RR)입니다. 레코드 형식은 고정되거나 가변적일 수 있습니다.
- Micro Focus 런타임 엔진: 가능한 값은 ESDS(ES), KSDS(KS) 및 RRDS(RR)입니다. VSAM 정의에는 레코드 형식을 포함하므로 별도로 지정할 필요가 없습니다.

primaryKeys

VSAM KSDS 데이터 세트에만 적용됩니다. 프라이머리 키를 지정합니다. 프라이머리 키 이름, 키 오프셋 및 키 길이로 구성됩니다. `name`는 선택 사항이며 `offset` 및 `length`는 필수입니다.

recordLength

필수 사항입니다. 레코드 길이를 지정합니다. 고정 길이 레코드 형식의 경우 이들 값이 일치해야 합니다.

- AWS Blu Age 런타임 엔진: VSAM ESDS, KSDS 및 RRDS의 경우 `min`는 선택 사항이며 `max`는 필수입니다.
- Micro Focus 런타임 엔진: `min` 및 `max`이 필수입니다.

externalLocation

필수 사항입니다. 원본 위치, 즉 데이터 세트를 업로드한 Amazon S3 버킷을 지정합니다.

Blu Age 엔진 전용 속성

AWS Blu Age 런타임 엔진은 VSAM 데이터 세트 압축을 지원합니다. 다음 예제는 JSON에서 이 속성을 지정하는 방법을 보여줍니다.

```
{
    common properties
        ...
        "datasetOrg": {
            "vsam": {
                common properties
                ...
                "compressed": boolean,
                common properties
                ...
            }
        }
}
```

압축 속성을 다음과 같이 지정합니다.

압축

선택 사항입니다. 이 데이터 세트의 인덱스를 압축된 값으로 저장할지 여부를 지정합니다. 데이터 세트가 큰 경우(일반적으로 100Mb 초과)이 있는 경우 이 플래그를 `true`로 설정하는 것이 좋습니다.

GDG Base의 샘플 데이터 세트 요청 형식

- AWS Blu Age 런타임 엔진: 지원되지 않습니다.
- Micro Focus 런타임 엔진: 지원됩니다.

GDG 기본 데이터 세트를 가져오는 경우 `datasetOrg`로 `gdg`를 지정하세요. 현대의 JSON 출력은 다음과 같을 것입니다.

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.GDG",
    "relativePath": "DATA",
    "datasetOrg": {
        "gdg": {
            "limit": "3",
            "rollDisposition": "Scratch and No Empty"
        }
    }
```

```
}
```

GDG 기본 데이터 세트에는 다음 속성이 지원됩니다.

제한

필수 사항입니다. 활성 세대 또는 편향 수를 지정합니다. GDG 기본 클러스터의 경우 최대값은 255입니다.

rollDisposition

선택 사항입니다. 최대값에 도달하거나 초과했을 때 생성 데이터 세트를 처리하는 방법을 지정합니다. 가능한 값은 `No Scratch and No Empty`, `Scratch and No Empty`, `Scratch and Empty`, 또는 `No Scratch and Empty`입니다. 기본값은 `Scratch and No Empty`입니다.

PS 또는 GDG 세대를 위한 샘플 데이터 세트 요청 형식

- AWS Blu Age 런타임 엔진: 지원되지 않습니다.
- Micro Focus 런타임 엔진: 지원됩니다.

PS 또는 GDG 세대 데이터 세트를 가져오는 경우 다음과 `datasetOrg`로 `ps`을 같이 지정하세요. JSON은 다음 예시와 같을 것입니다.

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.PS.FB",
    "relativePath": "DATA",
    "datasetOrg": {
        "ps": {
            "format": "FB",
            "encoding": "A"
        }
    },
    "recordLength": {
        "min": 300,
        "max": 300
    }
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/catalog/data/AWS.M2.PS.LSEQ"
}
```

```
}
```

PS 또는 GDG 세대 데이터 세트에 지원되는 속성은 다음과 같습니다.

형식

필수 사항입니다. 데이터 세트 레코드의 형식을 지정합니다. 가능한 값은 F, FA, FB, FBA, FBM, FBS, FM, FS, LSEQ, U, V, VA, VB, VBA, VBM, VBS, VM 및 VS입니다.

인코딩

필수 사항입니다. 데이터 세트의 문자 세트 인코딩을 지정합니다. 가능한 값은 ASCII(A), EBCDIC(E) 및 알 수 없음(?) 입니다

recordLength

필수 사항입니다. 레코드 길이를 지정합니다. 레코드의 최소(min) 길이와 최대(max) 길이를 모두 지정해야 합니다. 고정 길이 레코드 형식의 경우 이들 값이 일치해야 합니다.

externalLocation

필수 사항입니다. 원본 위치, 즉 데이터 세트를 업로드한 Amazon S3 버킷을 지정합니다.

PO용 샘플 데이터 세트 요청 형식

PO 데이터 세트를 가져오는 경우 datasetOrg로 po를 지정하세요. JSON은 다음 예시와 같을 것입니다.

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.PO.PROC",
    "relativePath": "DATA",
    "datasetOrg": {
        "po": {
            "format": "LSEQ",
            "encoding": "A",
            "memberFileExtensions": ["PRC"]
        }
    },
    "recordLength": {
        "min": 80,
        "max": 80
    }
```

```
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/source/proc/"
}
}
```

PO 데이터 세트에는 다음과 같은 속성이 지원됩니다.

형식

필수 사항입니다. 데이터 세트 레코드의 형식을 지정합니다. 가능한 값은 F, FA, FB, FBA, FBM, FBS, FM, FS, LSEQ, U, V, VA, VB, VBA, VBM, VBS, VM 및 VS입니다.

인코딩

필수 사항입니다. 데이터 세트의 문자 세트 인코딩을 지정합니다. 가능한 값은 ASCII(A), EBCDIC(E) 및 알 수 없음(?)입니다.

memberFileExtensions

필수 사항입니다. 하나 이상의 파일 이름 확장자가 포함된 배열을 지정하여 PDS 멤버로 포함할 파일을 지정할 수 있습니다.

recordLength

선택 사항입니다. 레코드 길이를 지정합니다. 레코드의 최소(min) 길이와 최대(max) 길이는 모두 선택 사항입니다. 고정 길이 레코드 형식의 경우 이들 값이 일치해야 합니다.

externalLocation

필수 사항입니다. 원본 위치, 즉 데이터 세트를 업로드한 Amazon S3 버킷을 지정합니다.

> **Note**
>
> Micro Focus 런타임 엔진의 현재 구현에서는 PDS 항목을 동적 데이터 세트로 추가합니다.

AWS 메인프레임 현대화의 관리형 런타임 환경

AWS를 처음 사용하는 경우 시작하려면 다음 단원을 참조하세요.

- 메인프레임 AWS 현대화란?
- AWS 메인프레임 현대화 설정
- AWS 메인프레임 현대화 시작하기
- 자습서: AWS Blu Age를 위한 관리형 런타임
- 자습서: Micro Focus의 관리형 런타임

AWS 메인프레임 현대화의 런타임 환경은 AWS 컴퓨팅 리소스, 런타임 엔진 및 사용자가 지정하는 구성 세부 사항의 명명된 조합입니다. 런타임 환경은 하나 이상의 애플리케이션을 호스팅합니다. AWS 메인프레임 현대화의 애플리케이션에는 마이그레이션된 메인프레임 워크로드가 포함됩니다. 생성하는 환경에 맞는 런타임 엔진을 선택할 수 있습니다. 자동 리팩토링 패턴을 사용하는 경우 AWS Blu Age를 선택하고 리플랫포밍 패턴을 사용하는 경우 Micro Focus를 선택합니다. 또한 애플리케이션에 적합한 컴퓨팅 리소스의 양을 선택하고 선택적으로 스토리지를 런타임 환경에 연결할 수 있습니다. AWS 메인프레임 현대화를 통해 Amazon CloudWatch 지표 및 로깅을 통해 런타임 환경을 모니터링할 수 있습니다.

주제

- AWS 메인프레임 현대화 런타임 환경을 만드세요
- AWS 메인프레임 현대화 런타임 환경 업데이트
- AWS 메인프레임 현대화 런타임 환경을 중단하세요
- AWS 메인프레임 현대화 런타임 환경을 재시작하세요
- AWS 메인프레임 현대화 런타임 환경 삭제

AWS 메인프레임 현대화 런타임 환경을 만드세요

AWS 메인프레임 현대화 콘솔을 사용하여 AWS 메인프레임 현대화 환경을 만들 수 있습니다.

이러한 지침에서는 AWS 메인프레임 현대화 설정의 단계를 완료한 것으로 가정합니다.

런타임 환경 만들기

런타임 환경 만들기

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 환경을 생성할 리전 리전을 선택합니다.
3. 환경 페이지에서 환경 생성을 선택합니다.
4. 추론 사양 페이지에서 다음 정보를 제공합니다.

    a. 이름 및 설명 섹션에 환경의 이름을 입력합니다.

    b. (선택 사항) 환경 설명 필드에 환경에 대한 설명을 입력합니다. 이 설명은 사용자 및 다른 사용자가 런타임 환경의 용도를 식별하는 데 도움이 될 수 있습니다.

    c. 엔진 옵션 섹션에서 자동 리팩토링을 위한 Blu Age를 선택하고 플랫폼 재구성을 위한 Micro Focus를 선택합니다.

    d. 선택한 엔진에 맞는 버전을 선택하세요.

    e. (선택 사항) 태그 섹션에서 새 태그 추가를 선택하여 애플리케이션에 하나 이상의 애플리케이션 태그를 추가합니다. 환경 태그는 사용자 또는 AWS 리소스를 구성하고 관리하는 데 도움이 되는 사용자 지정 속성 레이블입니다.

    f. 다음을 선택합니다.

5. 구성 지정 페이지에서 다음 정보를 제공합니다.

    a. 가용성 섹션에서 독립형 런타임 환경 또는 고가용성 클러스터를 선택합니다.

    가용성 패턴은 응용 프로그램 실행 시 사용할 수 있는 정도를 결정합니다. 독립 실행형은 개발 목적으로는 괜찮습니다. 고가용성은 항상 사용 가능해야 하는 애플리케이션을 위한 것입니다.

    b. 리소스에서 인스턴스 유형과 원하는 용량을 선택합니다.

    이러한 리소스는 런타임 환경을 호스팅하는 AWS 메인프레임 현대화 관리형 Amazon EC2 인스턴스입니다. 독립형 런타임 환경은 인스턴스 유형에 대해 두 가지 옵션을 제공하며 인스턴스 하나만 허용합니다. 고가용성 런타임 환경은 인스턴스 유형에 대해 두 가지 선택 사항을 제공하며 최대 두 개의 인스턴스를 허용합니다.

    자세한 내용은 Amazon EC2 인스턴스 유형을 참조하고 AWS 메인프레임 전문가에게 문의하여 지침을 받으세요.

6. 보안 및 네트워크 섹션에서 다음을 수행합니다.

a. 애플리케이션을 공개적으로 액세스할 수 있게 하려면 이 환경에 배포된 애플리케이션을 공개적으로 액세스할 수 있도록 허용을 선택합니다.

b. Virtual Private Cloud(VPC).

c. 고가용성 패턴을 사용하는 경우 두 개 이상의 서브넷을 선택하세요. AWS Blu Age 엔진과 함께 독립형 패턴을 사용하는 경우 두 개 이상의 서브넷을 선택하세요. Micro Focus 엔진에서 독립형 패턴을 사용하는 경우 서브넷 하나를 지정할 수 있습니다.

d. 선택한 VPC의 보안 그룹을 선택합니다.

> **Note**
>
> AWS 메인프레임 현대화는 런타임 환경에 대한 연결을 분산하기 위한 Network Load Balancer를 생성합니다. 보안 그룹 인바운드 규칙이 IP 주소에서 애플리케이션 정의의 `listener` 속성에 지정한 포트로의 액세스를 허용하는지 확인하세요. 자세한 내용은 Network Load Balancer 사용 설명서의 대상 등록을 참조하세요.

e. 고객 관리형 AWS KMS key를 사용하려면 KMS 키 필드에서 암호화 설정 사용자 지정을 선택합니다. 자세한 설명은 메인프레임 현대화 서비스를 위한 유휴 데이터 암호화 AWS 섹션을 참조하세요.

> **Note**
>
> 기본적으로 메인프레임 현대화는 AWS 메인프레임 현대화가 소유하고 관리하는 AWS KMS key로 데이터를 암호화합니다. 그러나 고객 관리 AWS KMS key를 사용할 수 있습니다.

f. (선택 사항) 이름 기준 AWS KMS key 또는 Amazon 리소스 이름(ARN)을 선택합니다. 또는 AWS KMS key 만들기를 선택하여 AWS KMS 콘솔로 이동하여 새 AWS KMS key을 생성할 수도 있습니다.

g. 다음을 선택합니다.

7. (선택 사항) 스토리지 연결 페이지에서 Amazon EFS 또는 Amazon FSx 파일 시스템을 하나 이상 선택한 후 다음을 선택합니다.

8. 유지 관리 기간 섹션에서 보류 중인 변경 사항을 환경에 적용할 시기를 선택합니다.

- 기본 설정 없음을 선택하면 AWS 메인프레임 현대화가 최적화된 유지 관리 기간을 선택합니다.
- 특정 유지 관리 기간을 지정하려면 새 유지 관리 기간 선택을 선택합니다. 유지 관리 기간의 시작 날짜, 시작 시간 및 기간을 선택할 수 있습니다.

유지 관리 기간에 대한 자세한 내용은 AWS 메인프레임 현대화 유지 관리 기간 섹션을 참조하세요.

다음을 선택합니다.

9. 검토 및 생성 페이지에서 제공한 정보를 검토한 다음 환경 만들기를 선택합니다.

AWS 메인프레임 현대화 런타임 환경 업데이트

AWS 메인프레임 현대화 콘솔을 사용하여 AWS 메인프레임 현대화 런타임 환경을 업데이트하세요. 런타임 엔진의 마이너 버전 또는 런타임 환경을 호스팅하는 인스턴스 유형을 업데이트할 수 있습니다. 업데이트를 즉시 적용할지 아니면 기본 유지 관리 기간 중에 적용할지 선택할 수 있습니다.

이러한 지침에서는 AWS 메인프레임 현대화 설정의 단계를 완료한 것으로 가정합니다.

런타임 환경 업데이트

런타임 환경 업데이트하기

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 업데이트하려는 환경이 생성된 리전을 선택합니다.
3. 환경 페이지에서 업데이트할 환경을 선택합니다.
4. 환경 개요 페이지에서 작업을 선택한 후 환경 URL 전환을 선택합니다.
5. 다음을 원하는 대로 변경합니다.
   - 엔진 옵션 섹션에서 원하는 엔진 버전을 선택합니다.
   - 리소스 섹션에서 원하는 인스턴스 유형을 선택합니다.
   - 유지 관리 기간 섹션에서 원하는 요일, 시간 및 기간을 선택합니다.

   > **Note**
   >
   > 유지 관리 기간 중에 적용할 수 있는 유일한 변경 사항은 엔진 버전에 대한 변경입니다. 다른 모든 변경 사항은 즉시 적용해야 합니다.

6. 다음을 선택합니다.

7. 이러한 변경 사항 적용 시기에서 즉시 또는 다음 유지 관리 기간 중을 선택합니다. 그 다음 환경 업데이트를 선택합니다.

즉시를 선택하면 환경 업데이트가 완료되었을 때 메시지가 표시됩니다.

AWS 메인프레임 현대화 유지 관리 기간

모든 런타임 환경에는 매주 1시간의 유지 관리 기간이 있습니다. 이 기간 동안 모든 시스템 변경 사항이 적용됩니다. 유지 관리 기간은 수정 사항 및 소프트웨어 패치 적용 시점을 조절할 수 있는 기회로 생각하면 됩니다. 유지 관리 이벤트가 지정된 주에 예약된 경우 할당된 한 시간 유지 관리 기간 중에 시작됩니다. 또한 대부분의 유지 관리 이벤트가 한 시간의 유지 관리 기간 중에 완료됩니다. 단, 대규모 유지 관리 이벤트는 완료하는 데 한 시간이 넘게 걸릴 수 있습니다.

리전별로 8시간 블록 시간 중에서 한 시간 유지 관리 기간이 임의로 선택됩니다. 런타임 환경을 만들 때 유지 관리 기간을 지정하지 않으면 AWS 메인프레임 현대화에서 임의로 선택한 요일에 1시간 유지 관리 기간을 배정합니다.

AWS 메인프레임 현대화는 유지 관리가 적용되는 동안 환경 인스턴스의 리소스 중 일부를 사용합니다. 유지 관리 중에 성능에 미미한 영향이 있거나 응용 프로그램이 중단되는 것을 확인할 수 있습니다.

다음 표는 기본 유지 관리 기간을 할당하는 각 리전의 시간 블록 목록입니다.

리전 이름	리전	시간 블록
미국 동부(버지니아 북부)	us-east-1	03:00~11:00 UTC
미국 서부(오레곤)	us-west-2	06:00~14:00 UTC
Asia Pacific (Mumbai)	ap-south-1	06:00~14:00 UTC
아시아 태평양(싱가포르)	ap-southeast-1	14:00~22:00 UTC
아시아 태평양(시드니)	ap-southeast-2	12:00~20:00 UTC
아시아 태평양(도쿄)	ap-northeast-1	13:00~21:00 UTC
Canada (Central)	ca-central-1	03:00~11:00 UTC

리전 이름	리전	시간 블록
유럽(프랑크푸르트)	eu-central-1	21:00–05:00 UTC
유럽(아일랜드)	eu-west-1	22:00~06:00 UTC
Europe (London)	eu-west-2	22:00~06:00 UTC
Europe (Paris)	eu-west-3	23:59~07:29 UTC
남아메리카(상파울루)	sa-east-1	00:00~08:00 UTC

AWS 메인프레임 현대화 런타임 환경을 중단하세요

AWS 메인프레임 현대화 콘솔을 사용하여 메인프레임 현대화 런타임 환경을 AWS 중지하세요. 환경을 중지하면 현재 애플리케이션 배포가 유지되며 환경을 다시 시작할 때까지 해당 환경에 대한 비용이 청구되지 않습니다.

이러한 지침에서는 AWS 메인프레임 현대화 설정의 단계를 완료한 것으로 가정합니다.

런타임 환경 중지

AWS 메인프레임 현대화 런타임 환경을 중지해야 하는 경우 업데이트 환경 섹션과 유사한 단계를 따르세요.

AWS 메인프레임 현대화 콘솔을 사용하여 AWS 메인프레임 현대화 런타임 환경을 중지할 수 있습니다. 환경을 중지하면 현재 애플리케이션 배포가 유지되며 환경을 다시 시작할 때까지 해당 환경에 대한 비용이 청구되지 않습니다.

런타임 환경 중지

AWS 메인프레임 현대화 런타임 환경을 중지하려면 업데이트 환경 섹션과 유사한 단계를 따르세요.

> **Note**
>
> 환경을 중지하기 전에 모든 애플리케이션을 중지해야 합니다.

런타임 환경을 중지하려면

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 중지하려는 환경이 생성된 리전을 선택합니다.
3. 환경 페이지에서 중지할 환경을 선택합니다.
4. 환경 개요 페이지에서 작업을 선택한 후 환경 편집을 선택합니다.
5. 환경 편집 페이지에서 리소스 섹션을 찾아 원하는 용량을 0으로 업데이트합니다.

> **Note**
>
> 환경을 중지하려면 즉시 중지하도록 선택할 수만 있습니다.

6. 다음을 선택합니다.
7. 이러한 변경 사항을 적용할 시기에서 즉시를 선택합니다. 환경 업데이트를 선택합니다.

환경 용량이 업데이트되면 메시지가 표시됩니다.

AWS 메인프레임 현대화 런타임 환경을 재시작하세요

AWS 메인프레임 현대화 콘솔을 사용하여 메인프레임 현대화 런타임 환경을 AWS 재시작하세요. 런타임 환경을 다시 시작하면 해당 환경에 대한 청구가 재개됩니다.

런타임 환경 재시작

AWS 메인프레임 현대화 런타임 환경을 재시작하려면 중지 환경 섹션과 유사한 단계를 따르세요.

런타임 환경을 다시 시작하려면

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 다시 시작하려는 환경이 생성된 리전을 선택합니다.
3. 환경 페이지에서 다시 시작할 환경을 선택합니다.
4. 환경 개요 페이지에서 작업을 선택한 후 환경 편집을 선택합니다.

> **Note**
>
> 독립형 환경에 필요한 용량은 1로만 업데이트할 수 있습니다. 런타임 환경을 다시 시작하려면 즉시 다시 시작하도록 선택할 수만 있습니다.

5. 환경 편집 페이지에서 리소스 섹션을 찾아 원하는 용량을 0에서 필요한 용량으로 업데이트합니다.
6. 다음을 선택합니다.
7. 이러한 변경 사항을 적용할 시기에서 즉시를 선택합니다. 그 다음 환경 업데이트를 선택합니다.

환경 용량이 업데이트되고 환경이 다시 시작되면 메시지가 표시됩니다.

AWS 메인프레임 현대화 런타임 환경 삭제

AWS 메인프레임 현대화 콘솔을 사용하여 메인프레임 현대화 런타임 환경을 AWS 삭제합니다.

이러한 지침에서는 AWS 메인프레임 현대화 설정의 단계를 완료한 것으로 가정합니다.

런타임 환경 삭제

AWS 메인프레임 현대화 런타임 환경을 삭제해야 하는 경우 먼저 해당 환경에서 배포된 애플리케이션을 모두 삭제해야 합니다. 애플리케이션이 배포된 런타임 환경은 삭제할 수 없습니다.

환경을 삭제하려면

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 중지하려는 환경이 생성된 리전을 선택합니다.
3. 환경 페이지에서 삭제하려는 환경을 선택한 다음 작업 및 환경 삭제를 선택합니다.
4. 환경 삭제 창에서 `delete`를 입력하여 런타임 환경을 삭제할 것인지 확인한 다음 삭제를 선택합니다.

AWS 메인프레임 현대화에서의 애플리케이션 테스트

AWS 애플리케이션 테스트는 AWS 메인프레임 현대화를 위한 프리뷰 릴리스이며 변경될 수 있습니다. 이 기능은 테스트 데이터 및 애플리케이션에만 사용하고, 프로덕션 환경에서는 사용하지 않는 것이 좋습니다.

AWS 메인프레임 현대화 애플리케이션 테스트는 마이그레이션 프로젝트에 대한 자동화된 기능적 동등성 테스트를 제공합니다.

주제

- AWS Mainframe Modernization Application Testing이란?
- AWS 메인프레임 현대화 애플리케이션 테스트 개념
- 자습서: CardDemo 샘플 애플리케이션 설치
- 자습서: AWS Amazon EC2에 배포된 AWS Blu CardDemo Age용 메인프레임 현대화 애플리케이션 테스트, 재생 및 비교
- AWS 메인프레임 현대화 애플리케이션 테스트 지원 데이터 세트 코드 페이지

AWS Mainframe Modernization Application Testing이란?

AWS Application Testing은 AWS Mainframe Modernization용 평가판 릴리스 단계에 있으므로, 변경될 수 있습니다. 이 기능은 테스트 데이터 및 애플리케이션에만 사용하고, 프로덕션 환경에서는 사용하지 않는 것이 좋습니다.

테스트는 마이그레이션 프로젝트에 상당한 영향을 미칩니다. 마이그레이션, 현대화 또는 논증 프로젝트에 들이는 시간과 노력의 최대 70%를 투자해야 할 수 있습니다. AWS Application Testing은 AWS Mainframe Modernization의 기능으로, 마이그레이션된 애플리케이션에 대한 자동화된 기능적 동등성 테스트를 제공합니다. 기능적 동등성 테스트를 통해 AWS 클라우드의 애플리케이션이 메인프레임의 애플리케이션과 동일한지 확인할 수 있습니다. AWS Application Testing은 메인프레임과 AWS 간 데이터 세트, 데이터베이스 레코드 및 온라인 3270 화면의 변경 사항을 자동으로 비교합니다. 나아가 Application Testing을 통해 테스트를 반복 실행할 수 있어 대상 아키텍처를 업데이트하고, 문제를 해결하고, 완전히 마이그레이션된 애플리케이션으로 진행하면서 테스트 시나리오를 여러 번 실행할 수 있습니다. 마이그레이션 후에도 계속해서 Application Testing을 회귀 테스트에 사용하여 런타임 엔진이

나 기타 구성 요소에 대한 업데이트로 인해 회귀가 발생하지 않는지 확인할 수 있습니다. Application Testing은 비용 효율적입니다. 대상 테스트 환경은 코드형 인프라(IaC) 개념을 활용하여 사용자가 제공하는 CloudFormation 템플릿을 통해 생성됩니다. Application Testing은 클라우드의 탄력성을 사용하여 마이그레이션 프로젝트를 가속화합니다. 필요한 만큼 많은 병렬 환경에서 독립적인 테스트 시나리오를 실행하여 테스트 일정을 줄일 수 있습니다.

주제

Application Testing을 처음 사용하는 경우

Application Testing을 처음 사용하는 경우 먼저 다음 섹션을 살펴보는 것이 좋습니다.

- Application Testing 개념
- 자습서: CardDemo 설치

Application Testing의 이점

Application Testing은 마이그레이션 프로세스에 도움이 되는 여러 가지 이점을 제공합니다.

- 테스트 가속화, 민첩성 및 유연성
- '메인프레임에서 한 번 기록하고 AWS에서 여러 번 재생'하는 테스트 개념
- 사용자 제공 CloudFormation 템플릿을 통한 대상 환경 IaC 생성
- 높은 수준의 테스트 반복성
- 확장성과 탄력성을 염두에 두고 클라우드용으로 구축
- 고도의 자동화를 통한 대규모 테스트
- 비용 효율성

AWS CloudFormation와의 통합

Application Testing은 AWS CloudFormation과 함께 코드형 인프라를 사용합니다. 이 설계 선택은 테스트 경험을 단순화하고 개선합니다. AWS CloudFormation에서는 요구 사항에 맞춰 더 나은 인프라를 정의할 수 있는 자율성과 독립성을 제공합니다. 여러 파라미터(인스턴스 크기, RDS 인스턴스, 최적 보안 그룹)를 독립적으로 선택하거나 정의할 수 있습니다. 애플리케이션이 테스트 조건에서 제대로 작동하는 데 필요한 Amazon SQS 대기열과 같은 리소스를 추가할 수 있습니다.

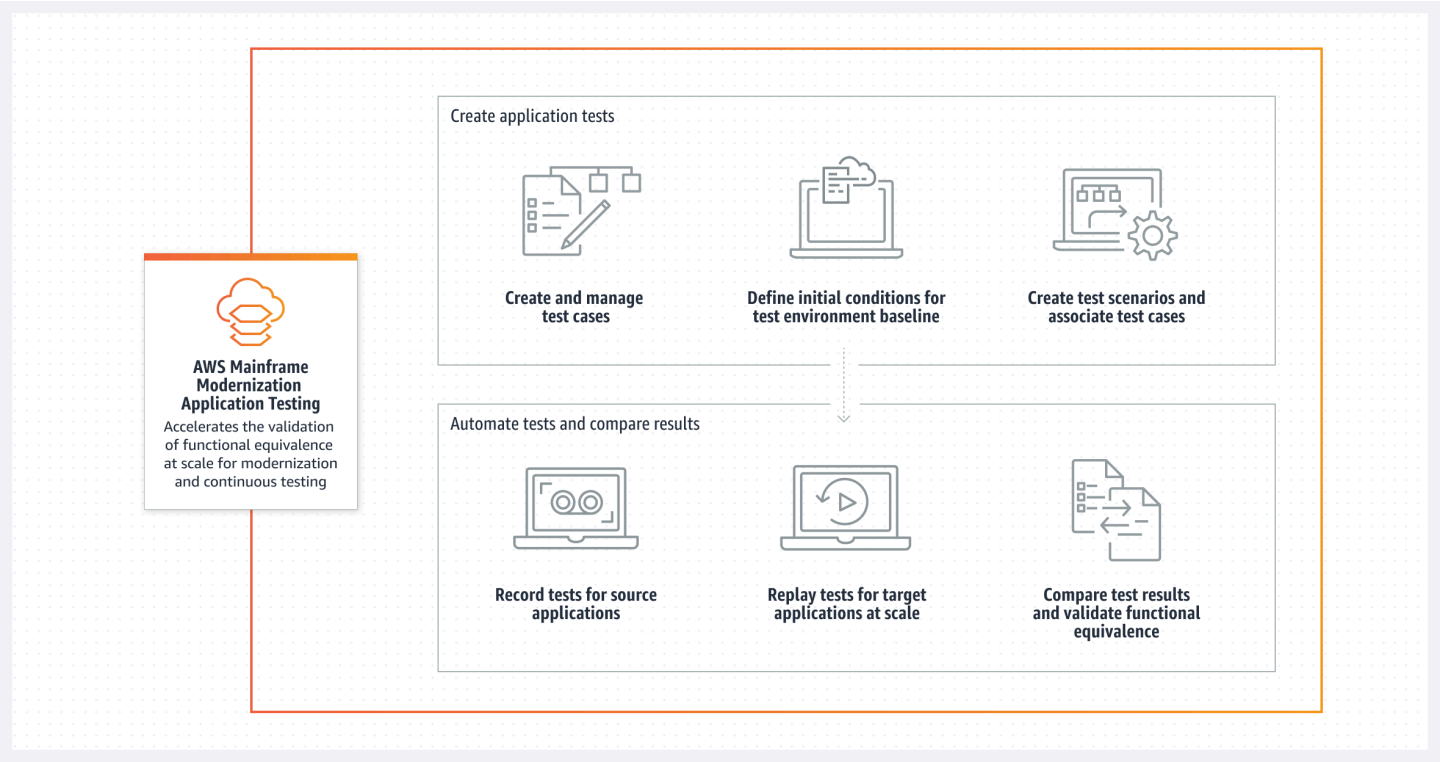
다운로드용으로 제공된 AWS CloudFormation 템플릿에서 몇 가지 일반적인 기능을 확인할 수 있습니다.

- Application Testing은 자체 네트워크 및 보안 정의를 통해 AWS Mainframe Modernization 런타임 환경 및 애플리케이션을 포함하여 완전히 격리된 스택을 생성합니다. 이 격리된 스택은 복원력을 제공하는데, 동일한 AWS 계정에 있는 다른 행위자가 테스트 활동을 방해할 수 없기 때문입니다. 또한, 시스템 운영자가 기본 VPC 또는 보안 그룹을 수정하여 테스트 활동 실패를 초래할 수 있는 상황도 방지할 수 있습니다.
- 보안 그룹을 사용하면 테스트에 사용되는 리소스에 대한 외부 액세스를 제어할 수도 있습니다. 예를 들어, 데이터베이스에는 기밀 데이터가 포함될 수 있습니다.
- 완전 격리는 VPC를 공유하는 다른 행위자가 트래픽을 스누핑하는 것을 방지합니다.
- 성능을 개선합니다. 예를 들어, 템플릿이 생성하는 AWS Mainframe Modernization 애플리케이션과 Amazon RDS 데이터베이스 간의 통신은 별도의 네트워크(프라이빗 VPC)에서 이루어지기에 다른 행위자가 트래픽 속도를 늦추지 못하도록 할 수 있습니다.

생성한 AWS CloudFormation 템플릿에도 이러한 기능을 구현하는 것이 좋습니다.

Application Testing 작동 방식

다음 그림은 Application Testing의 기능적 동등성 테스트 작동 방식을 간략하게 설명합니다.



- - AWS Mainframe Modernization File Transfer 또는 선호하는 다른 메인프레임 데이터 전송용 도구를 사용하여 소스에서 AWS로 입력 데이터를 전송할 수 있습니다.
- 소스와 대상 모두에서 동일한 비즈니스 로직을 실행합니다.
- Application Testing은 소스와 대상의 출력 데이터(데이터 세트, 관계형 데이터베이스 변경 사항, 3270 화면, 사용자 상호 작용)를 자동으로 비교합니다. 메인프레임에서 테스트 시나리오를 실행한 후 출력 데이터를 캡처하여 AWS에 전송한 후 대상에서 테스트 시나리오를 재생합니다. Application Testing은 AWS에서 실행된 테스트의 출력 데이터와 소스의 출력 데이터를 자동으로 비교합니다. 어떤 레코드가 동일한지, 동등한지, 다른지 또는 누락되었는지 한눈에 확인할 수 있습니다. 나아가 동등성 규칙을 정의하여 동일하지는 않아도 비즈니스 의미는 같은 레코드를 동일한 것으로 이해할 수 있습니다.

Application Testing에서 따르는 워크플로는 다음 단계로 구성됩니다.

1. 테스트 사례를 생성합니다. 테스트 사례는 테스트 작업의 최소 단위입니다. 테스트 사례를 만들 때는 소스와 대상 간의 기능적 동등성을 가장 잘 나타내는 비교 대상 데이터 유형도 식별해야 합니다.
2. 테스트 시나리오를 생성합니다. 테스트 시나리오는 관련 테스트 사례를 실행을 위한 특정 시퀀스로 그룹화합니다.
3. 초기 조건을 생성합니다. 초기 조건에서는 메인프레임에서 테스트를 기록하고 AWS CloudFormation을 사용하여 AWS에서 동일한 상태를 자동으로 생성하는 방법을 설명합니다.

4. 소스에서 기록하고 대상에서 재생합니다. 메인프레임에서 입력 및 출력 데이터 세트를 캡처하여 AWS에 업로드합니다. 그런 다음 AWS에서 테스트 시나리오를 다시 재생합니다.
5. 소스 및 대상 데이터 세트를 비교합니다. Application Testing은 소스와 대상의 출력 데이터 세트를 자동으로 비교하므로, 정확한 요소와 정확하지 않은 요소를 한눈에 확인할 수 있습니다.

테스트 시나리오의 최종 작업이자 전체 프로세스의 목표는 소스 테스트 실행과 대상 테스트 실행 간의 불일치를 식별하는 것입니다. Application Testing은 테스트 실행 중에 모든 상호 작용 채널에서 캡처된 데이터의 소스 버전과 대상 버전을 비교합니다. 또한, 테스트 사례에 정의된 관련 데이터의 최종 상태도 비교합니다.

관련 서비스

Application Testing은 AWS Mainframe Modernization의 기능입니다. 또한, AWS CloudFormation과 함께 코드형 인프라를 사용하여 테스트 반복성, 자동화 및 비용 효율성을 보장합니다. 자세한 내용은 다음을 참조하십시오.

- AWS 메인프레임 현대화
- AWS CloudFormation

Application Testing 액세스

AWS Mainframe Modernization 콘솔에서 왼쪽 탐색 메뉴의 Application Testing을 선택하여 Application Testing에 액세스할 수 있습니다.

Application Testing 요금

Application Testing 요금은 AWS Mainframe Modernization 요금에서 확인할 수 있습니다.

AWS 메인프레임 현대화 애플리케이션 테스트 개념

AWS 애플리케이션 테스트는 AWS 메인프레임 현대화를 위한 프리뷰 릴리스이며 변경될 수 있습니다. 이 기능은 테스트 데이터 및 애플리케이션에만 사용하고, 프로덕션 환경에서는 사용하지 않는 것이 좋습니다.

AWS 애플리케이션 테스트에서는 다른 테스트 서비스 또는 소프트웨어 패키지가 약간 다른 의미로 사용할 수 있는 용어를 사용합니다. 다음 섹션에서는 AWS 메인프레임 현대화 애플리케이션 테스트에서 이 용어를 사용하는 방법을 설명합니다.

주제

- 테스트 사례
- 테스트 시나리오
- 테스트 프로젝트
- 초기 조건
- 기록(캡처)
- 다시 재생
- Compare
- 데이터베이스 비교
- 데이터 세트 비교
- 비교 상태
- 동등성 규칙
- 최종-상태 데이터 세트 비교
- 상태-진행 데이터베이스 비교
- 기능적 동등성(FE)
- 온라인 3270 화면 비교
- Recording
- 재생 데이터
- 참조 데이터
- 기록, 재생, 비교
- 차이
- 동등성
- 소스 애플리케이션
- 대상 애플리케이션

테스트 사례

테스트 사례는 테스트 워크플로에서 가장 기본이 되는 개별 작업 단위입니다. 일반적으로 테스트 사례는 데이터를 수정하는 독립적인 비즈니스 로직 단위를 나타내는 데 사용됩니다. 각 테스트 사례에 대해 비교가 수행됩니다. 테스트 사례가 테스트 시나리오에 추가됩니다. 테스트 사례에는 테스트 사례에서 수정하는 데이터 아티팩트(데이터 세트, 데이터베이스)에 대한 메타데이터와 테스트 사례 실행 중에 트리거되는 비즈니스 함수(배치 작업, 3270 대화형 대화 상자 등)에 대한 메타데이터가 포함됩니다. 데이터 세트의 이름 및 코드 페이지를 예로 들 수 있습니다.

입력 데이터 → 테스트 사례 → 출력 데이터

테스트 사례는 온라인 또는 배치 유형일 수 있습니다.

- 온라인 테스트 사례는 사용자가 대화형 화면 대화 상자(3270)를 실행하여 새 비즈니스 데이터(데이터베이스 및/또는 데이터 세트 레코드)를 읽고 수정하거나 생성하는 테스트 사례입니다.
- 배치 테스트 사례는 새로운 비즈니스 데이터(데이터 세트 및/또는 데이터베이스 레코드)를 읽거나 처리하거나 수정하거나 생성하기 위해 배치를 제출해야 하는 테스트 사례입니다.

테스트 시나리오

테스트 시나리오는 순차적으로 하나씩 실행되는 일련의 테스트 사례입니다. 재생은 테스트 시나리오 수준에서 수행됩니다. 테스트 시나리오가 재생될 때 테스트 시나리오의 모든 테스트 사례는 대상 테스트 환경에서 실행됩니다. 참조 및 재생 테스트 아티팩트를 비교한 후 차이가 있는 경우 테스트 사례 수준에서 차이가 표시됩니다.

예: 테스트 시나리오 A:

테스트 사례 1, 테스트 사례 2, 테스트 사례 3 등

테스트 프로젝트

테스트 프로젝트는 원하는 테스트 기준에 도달하기 위한 테스트 시나리오 모음을 말합니다. 예를 들어, 특정 애플리케이션의 마이그레이션을 단일 테스트 프로젝트로 간주할 수 있습니다. 테스트 시나리오를 테스트 프로젝트로 그룹화하면 테스트 관리자가 통과/실패 테스트를 포함한 테스트 프로젝트 상태를 추적할 수 있습니다.

초기 조건

초기 조건에는 생성해야 하는 리소스 세트(컴퓨팅, 데이터 스토어 등)와 테스트 시나리오를 실행하기 전에 생성된 리소스에서 복원해야 하는 애플리케이션 데이터가 포함됩니다. 이로써 대상 테스트 환경

기준이 만들어지고 템플릿을 제공할 수 있습니다. AWS CloudFormation 템플릿을 사용하여 대상 테스트 환경을 만들고, 필요에 따라 테스트 시나리오에서 데이터베이스 레코드를 수정하는 경우 소스 데이터베이스에서 DDL을 추출할 수도 있습니다. 모든 테스트 시나리오는 초기 조건과 연결됩니다. 초기 조건을 여러 테스트 시나리오에 연결할 수 있습니다. 반복성과 결과 일관성을 유지하고 이미 변경된 데이터로 인한 거짓 긍정을 방지하려면 각 테스트 시나리오를 실행하기 전에 초기 조건을 복원해야 합니다.

데이터베이스 레코드를 수정하는 테스트 사례가 포함된 테스트 시나리오의 경우 초기 조건은 소스 데이터베이스 스키마 및 표의 DDL 내보내기도 참조합니다.

기록(캡처)

기록은 테스트 시나리오 수준에서 수행됩니다. 기록하는 동안 비교할 소스 메인프레임의 아티팩트, 데이터 세트 및 관계형 데이터베이스 CDC 저널이 포함된 Amazon S3 위치를 제공해야 합니다. 이는 소스 메인프레임의 참조 데이터로 간주됩니다. 재생 중에는 생성된 재생 데이터를 기록된 참조 데이터와 비교하여 애플리케이션이 동등한지 확인합니다.

다시 재생

재생은 테스트 시나리오 수준에서 수행됩니다. 리플레이 중에 AWS 메인프레임 현대화 애플리케이션 테스트는 관련 초기 조건에서 참조된 CloudFormation 스크립트를 사용하여 대상 테스트 환경을 만들고 애플리케이션을 실행합니다. 재생 중에 수정된 데이터 세트 및 데이터베이스 레코드는 캡처되어 메인프레임의 참조 데이터와 비교됩니다. 일반적으로 메인프레임에 한 번 기록한 후 기능이 동등해질 때까지 여러 번 재생합니다.

Compare

재생이 성공적으로 끝나면 자동으로 비교가 이루어집니다. 비교하는 동안 기록 단계에서 업로드하고 캡처한 참조 데이터를 재생 단계에서 생성된 재생 데이터와 비교합니다. 비교는 데이터 세트, 데이터베이스 레코드, 온라인 화면에 대해 개별 테스트 사례 수준에서 각각 수행됩니다.

데이터베이스 비교

Application Testing은 소스 애플리케이션과 대상 애플리케이션 간의 데이터베이스 레코드 변경 사항을 비교할 때 상태-진행 일치 기능을 사용합니다. 상태-진행 일치는 프로세스 종료 시 표의 행을 비교하는 것과 달리 INSERT, UPDATE, DELETE 문의 개별 실행 차이를 비교합니다. 상태-진행 일치는 변경된 데이터만 비교하고 트랜잭션 흐름에서 자체 수정 오류를 탐지하여 더 빠르고 정확한 비교를 제공하는 등 다른 방법보다 효율적입니다. Application Testing은 Changed Data Capture(CDC) 기술을 사용하여 개별 관계형 데이터베이스 변경 사항을 탐지하고 소스와 대상 간의 변경 사항을 비교할 수 있습니다.

관계형 데이터베이스 변경 사항은 SQL INSERT, UPDATE, DELETE 등의 데이터 수정 언어(DML) 문을 사용하여 테스트된 애플리케이션 코드로 인해 소스 및 대상에서 생성됩니다. 또한, 애플리케이션이 저장 프로시저를 사용하거나, 일부 표에 데이터베이스 트리거가 설정되었거나, 참조 무결성을 보장하기 위해 CASCADE DELETE를 사용하여 자동으로 추가 삭제를 트리거하는 경우에도 간접적으로 생성됩니다.

데이터 세트 비교

Application Testing은 소스(기록) 시스템과 대상(재생) 시스템에서 생성된 참조 및 재생 데이터 세트를 자동으로 비교합니다.

데이터 세트를 비교하려면

1. 소스와 대상 모두에서 동일한 입력 데이터(데이터 세트, 데이터베이스)로 시작합니다.
2. 소스 시스템(메인프레임)에서 테스트 사례를 실행합니다.
3. 생성된 데이터 세트를 캡처하여 Amazon S3 버킷에 업로드합니다. 원본의 입력 데이터세트를 CDC 저널, 화면 및 데이터세트를 AWS 사용하여 전송할 수 있습니다.
4. 테스트 사례를 기록할 때 메인프레임 데이터 세트가 업로드된 Amazon S3 버킷의 위치를 지정합니다.

재생이 완료되면 Application Testing은 출력 참조와 대상 데이터 세트를 자동으로 비교하여 레코드가 같은지, 동등한지, 다른지 또는 누락되었는지 보여줍니다. 예를 들어, 워크로드 실행 시점을 기준으로 하는 날짜 필드(일 +1일, 이번 달 말 등)는 자동으로 동등한 것으로 간주됩니다. 또한, 필요에 따라 동등성 규칙을 정의하여 동일하지 않아도 같은 비즈니스 의미를 지닌 레코드를 동등한 것으로 플래그 지정되도록 할 수 있습니다.

비교 상태

Application Testing에서는 IDENTICAL, EQUIVALENT, DIFFERENT의 비교 상태를 사용합니다.

IDENTICAL

소스 데이터와 대상 데이터가 완전히 동일합니다.

EQUIVALENT

소스 및 대상 데이터에는 동등한 것으로 간주되는 잘못된 차이가 포함되어 있습니다. 예를 들어, 날짜 또는 타임스탬프는 워크로드 실행 시점을 기준으로 할 때 기능적 동등성에 영향을 주지 않습니다. 동등성 규칙을 정의하여 이러한 차이점을 식별할 수 있습니다. 재생된 모든 테스트 시나리오를

참조 테스트 시나리오와 비교한 결과 IDENTICAL 또는 EQUIVALENT 상태가 표시되면 테스트 시나리오에서 기능적 동등성이 입증된 것입니다.

DIFFERENT

소스 및 대상 데이터에는 데이터 세트의 레코드 수가 상이하거나 같은 레코드에서 값이 차이가 나는 등 다른 점이 있습니다.

동등성 규칙

동등한 결과로 간주될 수 있는 잘못된 차이를 식별하기 위한 일련의 규칙입니다. 오프라인 기능 동등성 테스트(OFET)를 사용하면 소스 시스템과 대상 시스템 간에 일부 결과에 필연적으로 차이가 발생합니다. 예를 들어, 업데이트 타임스탬프는 설계에 따라 다릅니다. 동등성 규칙은 이러한 차이를 조정하고 비교 시 거짓 긍정을 방지하는 방법을 설명합니다. 예를 들어, 날짜가 특정 데이터 열의 런타임+2일인 경우 동등성 규칙은 이를 설명하고 참조 레코드의 동일한 열과 완전히 같은 값이 아닌 대상 시스템에서 대상+2일이 런타임인 시간을 허용합니다.

최종-상태 데이터 세트 비교

생성되거나 수정된 데이터 세트의 최종 상태(초기 상태부터 데이터 세트에 대한 모든 변경 또는 업데이트 포함)입니다. 데이터 세트의 경우 Application Testing은 테스트 사례 실행 종료 시 해당 데이터 세트의 레코드를 살펴보고 결과를 비교합니다.

상태-진행 데이터베이스 비교

데이터베이스 레코드에 수행된 변경 사항을 개별 삭제, 업데이트, 삽입(DML) 문의 시퀀스로 비교합니다. Application Testing은 소스 데이터베이스의 개별 변경 사항(테이블 행 삽입, 업데이트 또는 삭제)을 대상 데이터베이스와 비교하고 각 개별 변경 사항의 차이점을 식별합니다. 예를 들어, 개별 INSERT 문을 사용하여 대상 데이터베이스와 소스 데이터베이스의 값이 다른 행을 표에 삽입할 수 있습니다.

기능적 동등성(FE)

입력 데이터가 같을 때 관찰 가능한 모든 연산에서 동일한 결과를 산출하는 두 시스템은 기능적으로 동등한 것으로 간주됩니다. 예를 들어, 화면이나 데이터 세트 변경, 데이터베이스 변경을 통해 동일한 입력 데이터가 동일한 출력 데이터를 생성하는 경우 두 애플리케이션은 기능적으로 동등한 것으로 여깁니다.

온라인 3270 화면 비교

대상 시스템이 의 Blu Age 런타임으로 실행되고 있을 때 메인프레임 3270 화면의 출력을 현대화된 애플리케이션 웹 화면의 출력과 비교합니다. AWS AWS 클라우드그리고 대상 시스템이 AWS 클라우드의 Micro Focus 런타임에서 실행 중일 때 메인프레임 3270 화면의 출력을 리호스팅된 애플리케이션의 3270 화면과 비교합니다.

Recording

잘 알려진 데이터 상태를 복원한 후 참조 테스트 시나리오의 참조 데이터(하나 또는 여러 테스트 사례에 대해 순차적으로)를 소스 시스템에 캡처하거나 기록하는 작업입니다.

재생 데이터

재생 데이터는 대상 테스트 환경에서 테스트 시나리오를 재생하여 생성된 데이터를 설명하는 데 사용됩니다. 예를 들어, 메인프레임 현대화 서비스 애플리케이션에서 테스트 시나리오를 실행할 때 리플레이 데이터가 생성됩니다. AWS 그런 다음 재생 데이터를 소스에서 캡처한 참조 데이터와 비교합니다. 대상 환경에서 워크로드를 재생할 때마다 새로운 세대의 재생 데이터가 생성됩니다.

참조 데이터

참조 데이터는 소스 메인프레임에서 캡처된 데이터를 설명하는 데 사용됩니다. 재생(대상)에서 생성된 데이터를 비교할 참조입니다. 일반적으로 참조 데이터를 생성하는 메인프레임의 모든 레코드에 대해 많은 재생이 발생합니다. 이는 사용자가 일반적으로 메인프레임에서 애플리케이션의 올바른 상태를 캡처하고 현대화된 대상 애플리케이션에서 테스트 사례를 재생하여 동등성을 검증하기 때문입니다. 버그가 발견되면 수정되고 테스트 사례가 다시 재생됩니다. 재생하고, 버그를 수정하고, 다시 재생하여 발생 여부를 확인하는 과정이 여러 번 반복 실행되는 경우가 많습니다. 한 번 캡처해서 여러 번 재생하는 것을 테스트의 패러다임이라고 합니다.

기록, 재생, 비교

Application Testing은 3단계로 진행됩니다.

- 기록: 테스트 시나리오의 각 테스트 사례에 대해 메인프레임에서 생성된 참조 데이터를 캡처합니다. 여기에는 3270 온라인 화면, 데이터 세트, 데이터베이스 레코드가 포함될 수 있습니다.
  - 온라인 3270 화면의 경우 Blu Insights 터미널 에뮬레이터를 사용하여 소스 워크로드를 캡처해야 합니다. 자세한 내용은 Blu Insights 설명서를 참조하세요.
  - 데이터 세트의 경우 FTP 또는 메인프레임 현대화의 데이터 세트 전송 서비스 부분과 같은 공통 도구를 사용하여 메인프레임의 각 테스트 사례에서 생성된 데이터 세트를 캡처해야 합니다. AWS

  - 데이터베이스 변경의 경우 Precisely를 사용한 AWS Mainframe Modernization 데이터 복제 설명서를 참조하여 변경 내용이 포함된 CDC 저널을 캡처하고 생성합니다.
- 재생: 테스트 시나리오는 대상 환경에서 재생됩니다. 테스트 시나리오에 지정된 모든 테스트 사례가 실행됩니다. 데이터 세트, 관계형 데이터베이스 변경 사항 또는 3270 화면과 같이 개별 테스트 사례에서 생성된 특정 데이터 유형은 자동화를 통해 캡처됩니다. 이러한 데이터를 재생 데이터라고 하며, 기록 단계에서 캡처된 참조 데이터와 비교됩니다.

  > **Note**
  >
  > 관계형 데이터베이스를 변경하려면 초기 조건 템플릿에 DMS별 구성 옵션이 필요합니다. CloudFormation

- 비교: 소스 테스트 참조 데이터와 대상 재생 데이터를 비교하면 결과가 동일하거나, 다르거나, 동등하거나, 누락된 데이터로 표시됩니다.

차이

데이터 비교를 통해 참조 데이터 세트와 재생 데이터 세트 간에 차이가 감지되었음을 나타냅니다. 예를 들어, 온라인 3270 화면에서 소스 메인프레임과 대상 현대화 애플리케이션 간에 비즈니스 로직 관점에서 서로 다른 값을 표시하는 필드는 다른 것으로 간주됩니다. 또 다른 예로 데이터 세트의 레코드가 소스 애플리케이션과 대상 애플리케이션 간에 동일하지 않은 경우를 들 수 있습니다.

동등성

동등한 레코드는 참조 데이터 세트와 재생 데이터 세트가 달라도 비즈니스 로직 관점에서는 다르게 취급되어서는 안 되는 레코드입니다. 데이터 세트가 생성된 시점의 타임스탬프(워크로드 실행 시간)가 포함된 레코드를 예로 들 수 있습니다. 사용자 지정 가능한 등가 규칙을 사용하면 참조 데이터와 재생 데이터 간에 다른 값이 표시되더라도, 이러한 거짓 긍정 차이를 동등한 것으로 처리하도록 Application Testing에 지시할 수 있습니다.

소스 애플리케이션

비교할 소스 메인프레임 애플리케이션입니다.

대상 애플리케이션

테스트를 수행하고, 소스 애플리케이션과 비교하여 결함을 감지하고 소스 및 대상 애플리케이션 간 기능적 동등성을 확보하는 신규 또는 수정된 애플리케이션입니다. 대상 애플리케이션은 일반적으로 클라우드에서 실행됩니다. AWS

자습서: CardDemo 샘플 애플리케이션 설치

> AWS Application Testing은 AWS Mainframe Modernization용 평가판 릴리스 단계에 있으므로, 변경될 수 있습니다. 이 기능은 테스트 데이터 및 애플리케이션에만 사용하고, 프로덕션 환경에서는 사용하지 않는 것이 좋습니다.

이 자습서에서는 AWS Mainframe Modernization 관리형 서비스에서 Micro Focus를 사용하여 리플랫포밍을 위해 CardDemo 샘플 애플리케이션을 설정하고 AWS Mainframe Modernization Application Testing을 포함한 기능을 설정하는 데 도움이 되는 AWS CloudFormation 스택을 생성하게 됩니다. 이 자습서에서는 스택을 생성하는 데 사용할 샘플 AWS CloudFormation 템플릿을 설명합니다. 또한, 필요한 애플리케이션 아티팩트의 압축 파일도 제공합니다. 예제 템플릿은 데이터베이스, 런타임 환경, 애플리케이션 및 완전히 격리된 네트워크 환경을 제공합니다.

이 템플릿은 여러 AWS 리소스를 생성합니다. 이 템플릿에서 스택을 생성하면 비용이 청구됩니다.

사전 조건

- `IC3-card-demo-zip` 및 `datasets_Mainframe_ebcdic.zip`을 다운로드하고 압축을 해제합니다. 이 파일에는 AWS Application Testing에 사용할 수 있는 CardDemo 샘플 및 샘플 데이터 세트가 들어 있습니다.
- CardDemo 파일 및 기타 아티팩트를 보관할 Amazon S3 버킷을 만듭니다. 예: `my-carddemo-bucket`.

1단계: CardDemo 설치 준비

CardDemo 샘플 파일을 업로드하고 CardDemo 애플리케이션을 생성할 AWS CloudFormation 템플릿을 편집합니다.

1. 이전에 압축을 푼 `datasets_Mainframe_ebcdic` 및 `IC3-card-demo` 폴더를 버킷에 업로드합니다.

2. 버킷에서 `aws-m2-math-mf-carddemo.yaml` AWS CloudFormation 템플릿을 다운로드합니다. `IC3-card-demo` 폴더에 있습니다.
3. 다음과 같이 `aws-m2-math-mf-carddemo.yaml` AWS CloudFormation 템플릿을 편집합니다.

   - `BucketName` 파라미터를 이전에 정의한 버킷 이름(예: `my-carddemo-bucket`)으로 변경합니다.
   - `ImportJsonPath`를 `mf-carddemo-datasets-import.json` 파일의 버킷 내 위치로 변경합니다. 예를 들어, `s3://my-carddemo-bucket/IC3-card-demo/mf-carddemo-datasets-import.json`입니다. 이 값을 업데이트하면 출력 `M2ImportJson`이 정확한 값인지 확인할 수 있습니다.
   - (선택 사항) `EngineVersion` 및 `InstanceType` 파라미터를 표준에 맞게 조정합니다.

> **Note**
>
> `M2EnvironmentId` 및 `M2ApplicationId` 출력을 수정하지 마세요. Application Testing은 해당 값을 사용하여 상호 작용할 리소스를 찾습니다.

2단계: 필요한 모든 리소스 생성

이 자습서를 성공적으로 완료하는 데 필요한 모든 리소스를 생성하려면 사용자 지정 AWS CloudFormation 템플릿을 실행합니다. 이 템플릿은 테스트에 사용할 수 있도록 CardDemo 애플리케이션을 설정합니다.

1. AWS CloudFormation 콘솔에 로그인한 후 스택 생성을 선택한 다음 새 리소스 사용(표준)을 선택합니다.
2. 사전 조건 - 템플릿 준비에서 템플릿 준비 완료를 선택합니다.
3. 템플릿 지정에서 템플릿 파일 업로드를 선택한 다음 파일 선택을 선택합니다.
4. `aws-m2-math-mf-carddemo.yaml`을 다운로드한 위치로 이동하여 해당 파일을 선택한 후 다음을 선택합니다.
5. 스택 세부 정보 지정에서 쉽게 목록에서 찾을 수 있도록 스택 이름을 입력하고 다음을 선택합니다.
6. 스택 구성 옵션의 기본 값을 그대로 둔 후 다음을 선택합니다.
7. 검토에서 AWS CloudFormation이 생성하는 요소를 확인한 후 제출을 선택합니다.

AWS CloudFormation에서 스택을 생성하는 데 약 10~15분이 걸립니다.

> **Note**
>
> 템플릿은 생성되는 리소스 이름에 고유한 접미사를 추가하도록 설정되어 있습니다. 즉, 스택 템플릿의 여러 인스턴스를 병렬로 만들 수 있는데, 이는 여러 테스트 시나리오를 동시에 실행할 수 있는 Application Testing의 주요 기능입니다.

3단계: 애플리케이션 배포 및 시작

AWS CloudFormation에서 생성한 CardDemo 애플리케이션을 배포하고 실행 중인지 확인합니다.

1. AWS Mainframe Modernization 콘솔을 열고 왼쪽 탐색 메뉴에서 애플리케이션을 선택합니다.
2. `aws-m2-math-mf-carddemo-abc1d2e3`와 비슷한 이름의 CardDemo 애플리케이션을 선택합니다.
3. 작업을 선택한 후 애플리케이션 배포를 선택합니다.
4. 환경에서 애플리케이션에 해당하는 런타임 환경을 선택합니다. 이름 끝에 동일한 고유 식별자가 추가됩니다. 예: `aws-m2-math-mf-carddemo-abc1d2e3`.
5. [배포]를 선택합니다. 애플리케이션이 성공적으로 배포되고 준비 상태가 될 때까지 기다립니다.
6. 애플리케이션을 선택하고 작업과 애플리케이션 시작을 차례로 선택합니다. 애플리케이션이 실행 상태가 될 때까지 기다리세요.
7. 애플리케이션 세부 정보 페이지에서 실행 중인 애플리케이션에 연결하는 데 필요한 포트 및 DNS 호스트 이름을 복사합니다.

4단계: 초기 데이터 가져오기

CardDemo 샘플 애플리케이션을 사용하려면 초기 데이터 세트를 가져와야 합니다. 다음 단계를 완료합니다.

1. `mf-carddemo-datasets-import.json` 파일을 다운로드합니다.
2. 선호하는 텍스트 편집기에서 파일을 엽니다.
3. `s3Location` 파라미터를 찾고 생성한 Amazon S3 버킷을 가리키도록 값을 업데이트합니다.
4. `s3Location`이 발생하는 모든 항목에 대해 동일하게 변경한 다음 파일을 저장합니다.
5. Amazon S3 콘솔에 로그인하고 앞서 생성한 버킷으로 이동합니다.
6. 사용자 지정 `mf-carddemo-datasets-import.json` 파일을 업로드합니다.

7. AWS Mainframe Modernization 콘솔을 열고 왼쪽 탐색 메뉴에서 애플리케이션을 선택합니다.
8. CardDemo 애플리케이션을 선택합니다.
9. 데이터 세트를 선택하고 가져오기를 선택합니다.
10. Amazon S3에서 사용자 지정 JSON 파일을 업로드한 위치로 이동한 다음 제출을 선택합니다.

이 작업은 23개의 데이터 세트를 가져옵니다. 가져오기 작업의 결과를 모니터링하려면 콘솔을 확인하세요. 모든 데이터 세트를 성공적으로 가져오면 애플리케이션에 연결합니다.

> **Note**
>
> Application Testing에서 이 템플릿을 사용하면 출력 `M2ImportJson`에서 가져오기 프로세스를 자동으로 처리합니다.

5단계: CardDemo 애플리케이션에 연결

원하는 3270 에뮬레이터를 사용하여 CardDemo 샘플 애플리케이션에 연결합니다.

- 애플리케이션이 실행되면 3270 에뮬레이터를 사용하여 애플리케이션에 연결하고 필요한 경우 DNS 호스트 이름과 포트 이름을 지정합니다.

  예를 들어, 오픈 소스 c3270 에뮬레이터를 사용하는 경우 명령은 다음과 같습니다.

  ```
  c3270 -port port-number DNS-hostname
  ```

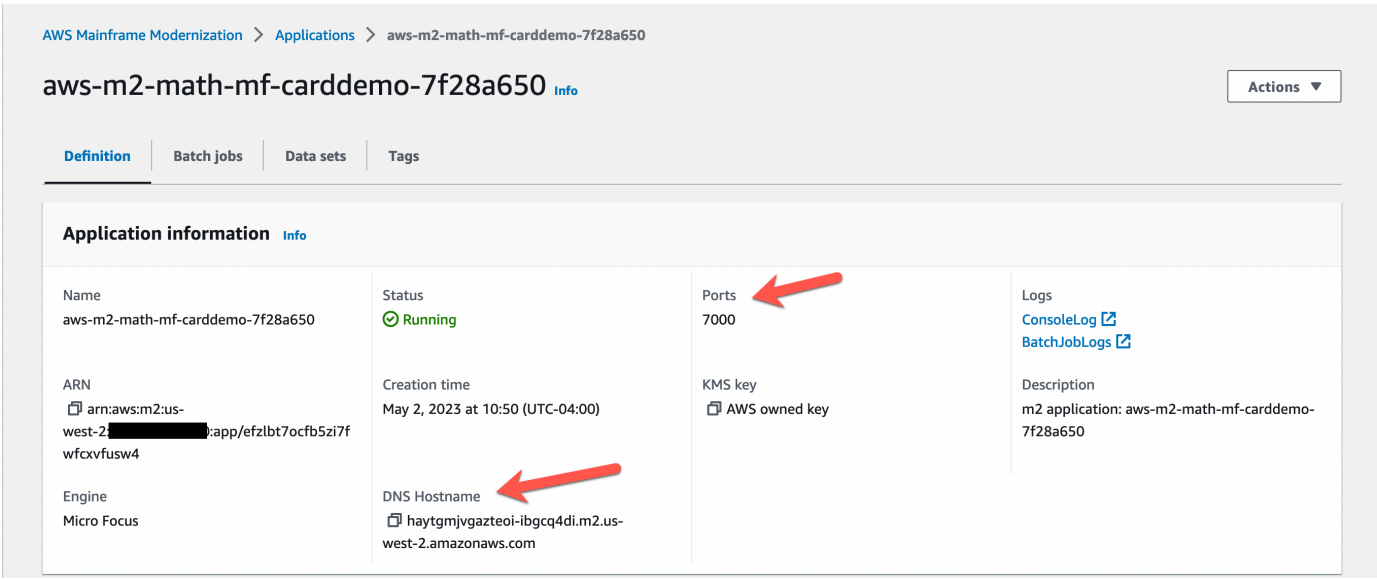
  포트

  애플리케이션 세부 정보 페이지에 지정된 포트입니다. 예를 들어, 6000입니다.

  Hostname

  애플리케이션 세부 정보 페이지에 지정된 DNS 호스트 이름입니다.

  다음 그림은 포트 및 DSN 호스트 이름을 찾을 수 있는 위치를 보여줍니다.



자습서: AWS Amazon EC2에 배포된 AWS Blu CardDemo Age용 메인프레임 현대화 애플리케이션 테스트, 재생 및 비교

AWS Application Testing은 AWS Mainframe Modernization용 평가판 릴리스 단계에 있으므로, 변경될 수 있습니다. 이 기능은 테스트 데이터 및 애플리케이션에만 사용하고, 프로덕션 환경에서는 사용하지 않는 것이 좋습니다.

이 자습서에서는 Amazon EC2에 배포된 AWS Blu Age에서 실행되는 CardDemo 애플리케이션과 테스트 워크로드를 재생하고 비교하는 데 필요한 단계를 완료합니다.

1단계: AWS 블루에이지 아마존 EC2 아마존 머신 이미지 (AMI) 확보

Amazon EC2 AMI에서 AWSAWS블루 에이지에 액세스하는 AWS 데 필요한 온보딩 단계는 블루 에이지 런타임 (Amazon EC2) 설정 자습서의 지침을 따르십시오.

2단계: AWS 블루에이지 AMI를 사용하여 Amazon EC2 인스턴스 시작

1. AWS 보안 인증 설정
2. Amazon S3 버킷에서 3.5.0 아마존 EC2 AMI 바이너리 파일 (CLI AWS 전용/블루 에이지 버전) 의 위치를 식별하십시오.

```
aws s3 ls s3://aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1/
```

```
aws s3 ls s3://aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1/3.5.0/AMI/
```

> **Note**
>
> Application Testing 기능은 prod의 4개 리전(us-east-1, sa-east-1, eu-central-1, ap-southeast-2)에서만 사용할 수 있습니다.

3. 다음 명령을 사용하여 계정의 AMI를 복원합니다.

```
aws ec2 create-restore-image-task --object-key 3.5.0/AMI/ami-0182ffe3b9d63925b.bin
 --bucket aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1 --region eu-west-1 --name
 "AWS BLUAGE RUNTIME AMI"
```

> **Note**
>
> AMI bin 파일 이름과 AMI를 생성하려는 리전을 바꿉니다.

4. Amazon EC2 인스턴스를 생성한 후 Amazon EC2 이미지 카탈로그에서 Amazon S3 버킷의 AMI를 복원한 올바른 AMI ID를 찾을 수 있습니다.

> **Note**
>
> 이 자습서에서 AMI ID는 ami-0d0fafcc636fd1e6d이며, 다른 구성 파일에서 이 ID를 제공된 ID로 변경해야 합니다.

1. aws ec2에 create-restore-image-task 오류가 발생하면 다음 명령을 사용하여 Python 및 CLI의 버전을 확인하십시오.

```
aws --version
```

> **Note**
>
> Python 버전은 >= 3이어야 하고 CLI 버전은 >= 2여야 합니다.

2. 이러한 버전이 더 이상 사용되지 않는 경우 CLI를 업데이트해야 합니다. CLI를 업데이트하려면 다음 단계를 따릅니다.

a. AWS CLI의 최신 버전 설치 또는 업데이트의 지침을 따릅니다.

b. 다음 명령을 사용하여 CLI v1을 제거합니다.

```
sudo yum remove awscli
```

c. 그리고 다음 명령을 사용하여 CLI v2를 설치합니다.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
 "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

d. 그리고 다음 명령을 사용하여 Python과 CLI의 버전을 확인합니다.

```
aws --version
```

3. 그런 다음 aws ec2를 다시 실행할 수 있습니다. create-restore-image-task

3단계: CardDemo 종속 파일을 S3에 업로드

databases, file-system, userdata 폴더의 내용을 복사합니다. CardDemo 애플리케이션을 다운로드하고 압축을 풉니다. 3개의 폴더를 이 설명서에 your-s3-bucket으로 나와 있는 버킷 중 하나에 복사해야 합니다.

4단계: 데이터베이스 로드 및 애플리케이션 초기화 CardDemo

애플리케이션에 필요한 데이터베이스 스냅샷을 생성하는 데 필요한 컴퓨팅 리소스로 사용할 임시 Amazon EC2 인스턴스를 생성합니다. CardDemo 이 EC2 인스턴스는 CardDemo 애플리케이션 자체를 실행하지 않고 나중에 사용할 데이터베이스 스냅샷을 생성합니다.

먼저 제공된 CloudFormation 템플릿 이름이 load-and-create-ba '-snapshots.yml'인 템플릿을 편집하십시오. 데이터베이스 스냅샷을 생성하는 데 사용되는 Amazon EC2 인스턴스를 생성하는 데 사용되는 CloudFormation 템플릿입니다.

1. EC2 인스턴스에 사용할 EC2 키 페어를 생성하고 제공합니다. 자세한 내용은 키 페어 생성을 참조하세요.

   예제

   ```
   Ec2KeyPair:
   ```

```
    Description: 'ec2 key pair'
    Default: 'm2-tests-us-west-2'
    Type: String
```

2. 이전 단계에서 database 폴더를 배치한 폴더의 Amazon S3 경로를 지정합니다.

```
S3DBScriptsPath:
    Description: 'S3 DB scripts folder path'
    Type: String
    Default: 's3://your-s3-bucket/databases'
```

3. 이전 단계에서 file-system 폴더를 배치한 폴더의 Amazon S3 경로를 지정합니다.

```
S3ApplicationFilesPath:
    Description: 'S3 application files folder path'
    Type: String
    Default: 's3://your-s3-bucket/file-system'
```

4. 이전 단계에서 userdata 폴더를 배치한 폴더의 Amazon S3 경로를 지정합니다.

```
S3UserDataPath:
    Description: 'S3 userdata folder path'
    Type: String
    Default: 's3://your-s3-bucket/userdata'
```

5. 다음 단계에서 사용할 결과 파일을 저장할 Amazon S3 경로도 지정합니다.

```
S3SaveProducedFilesPath:
    Description: 'S3 path folder to save produced files'
    Type: String
    Default: 's3://your-s3-bucket/post-produced-files'
```

6. 다음 템플릿을 사용하여 AMI ID를 이 자습서의 앞부분에서 가져온 올바른 ID로 변경합니다.

```

  BaaAmiId:
    Description: 'ami id (AL2) for ba anywhere'
    Default: 'ami-0bd41245734fd20d9'
    Type: String
```

- 로드 데이터베이스 실행으로 생성되는 세 개의 스냅샷의 이름을 선택적으로 변경할 수 있습니다. withCloudFormation 이러한 정보는 스택이 생성될 때 해당 CloudFormation 스택에서

볼 수 있으며 이 자습서의 뒷부분에서 사용할 예정입니다. 데이터베이스 스냅샷에 사용된 이름을 기록해 두세요.

```
SnapshotPrimary:
    Description: 'Snapshot Name DB BA Primary'
    Type: String
    Default: 'snapshot-primary'

  SnapshotBluesam:
    Description: 'Snapshot Name DB BA Bluesam'
    Type: String
    Default: 'snapshot-bluesam'

  SnapshotJics:
    Description: 'Snapshot Name DB BA Jics'
    Type: String
    Default: 'snapshot-jics'
```

> **Note**
>
> 이 문서에서는 스냅샷의 이름이 일관성을 유지한다고 가정합니다.

7. 스택 생성 버튼 및 마법사를 사용하여 CloudFormation CLI로 또는 AWS 콘솔을 실행합니다. 프로세스가 끝나면 RDS 콘솔에 선택한 이름 뒤에 고유 ID가 붙은 3개의 스냅샷이 표시됩니다. 다음 단계에서 해당 이름이 필요합니다.

> **Note**
>
> RDS는 AWS CloudFormation 템플릿에 정의된 스냅샷 이름에 접미사를 추가합니다. 다음 단계로 진행하기 전에 RDS에서 전체 스냅샷 이름을 얻어야 합니다.

CLI 명령 예-

```
aws cloudformation create-stack --stack-name load-and-create-ba-snapshots --
template-url https://your-apptest-bucket.s3.us-west-2.amazonaws.com/load-and-
create-ba-snapshots.yml --capabilities CAPABILITY_NAMED_IAM
```

또한 S3에 제공한 Amazon S3 경로에서 데이터 세트가 올바르게 생성되었는지 확인할 수 있습니다. SaveProducedFilesPath

5단계: AWS 블루에이지 런타임 시작 CloudFormation

CardDemo AWSBlu Age 애플리케이션과 함께 Amazon EC2 인스턴스를 실행하는 CloudFormation 데 사용합니다. YAML 파일을 편집하거나 CFN을 시작하는 동안 콘솔에서 값을 `m2-with-ba-using-snapshots-https-authentication.yml` 수정하여 CloudFormation 명명된 변수의 일부 변수를 바꿔야 합니다.

1. 다음 명령을 사용하여 AWS Blu Age 런타임에 `AllowedVpcEndpointPrincipals` 액세스하기 위해 VPC 엔드포인트에 도달할 계정을 지정하도록 수정하십시오.

```
AllowedVpcEndpointPrincipals:
    Description: 'comma-separated list of IAM users, IAM roles, or AWS accounts'
    Default: 'apptest.amazonaws.com'
    Type: String
```

2. 변수 값 및 SnapshotPrimaryDb SnapshotBlusamDb SnapshotJicsDb 스냅샷의 이름을 변경하십시오. 또한, 이전 단계에서 생성된 스냅샷 이름을 RDS에서 가져옵니다.

```
SnapshotPrimary:
    Description: 'Snapshot DB cluster for DB Primary'
    Type: String
    Default: 'snapshot-primary87d067b0'

  SnapshotBluesam:
    Description: 'Snapshot DB cluster for DB Bluesam'
    Type: String
    Default: 'snapshot-bluesam87d067b0'

  SnapshotJics:
    Description: 'Snapshot DB cluster for DB Jics'
    Type: String
    Default: 'snapshot-jics87d067b0'
```

> **Note**
>
> RDS는 스냅샷 이름에 고유한 접미사를 추가합니다.

3. 다음 명령을 사용하여 EC2 인스턴스용 Amazon EC2 키 페어를 제공합니다.

```
Ec2KeyPair:
    Description: 'ec2 key pair'
    Default: 'm2-tests-us-west-2'
    Type: String
```

4. 변수에 BaaAmiId대한 AMI 등록 프로세스 중에 획득한 AMI ID를 다음을 사용하여 입력합니다.

```
BaaAmiId:
    Description: 'ami id (AL2) for ba anywhere'
    Default: 'ami-0d0fafcc636fd1e6d'
    Type: String
```

5. 다음 명령을 사용하여 이전 단계에서 생성된 파일을 저장하는 데 사용한 Amazon S3 폴더 경로를 제공합니다.

```
S3ApplicationFilesPath:
    Description: 'bucket name'
    Type: String
    Default: 's3://your-s3-bucket/post-produced-files'
```

6. 마지막으로, s3- userdata-folder-path 의 폴더 경로를 제공하십시오.

```
S3UserDataPath:
    Description: 'S3 userdata folder path'
    Type: String
    Default: 's3://your-s3-bucket/userdata'
```

- (선택 사항) tomcat에 대해 HTTPS 모드와 기본 HTTP 인증을 활성화할 수 있습니다. 기본 설정도 사용할 수 있습니다.

> **Note**
>
> 기본적으로 HTTPS 모드는 비활성화되며 파라미터에서 HTTP 모드로 설정됩니다. BacHttpsMode

예:

```
BacHttpsMode:
    Description: 'http or https for Blue Age Runtime connection mode '
    Default: 'http'
    Type: String
    AllowedValues: [http, https]
```

- (선택 사항) HTTPS 모드를 활성화하려면 값을 HTTPS로 변경하고 변수 ACM의 값을 변경하여 ACM 인증서 ARN을 제공해야 합니다. CertArn

```
ACMCertArn:
    Type: String
    Description: 'ACM certificate ARN'
    Default: 'your arn certificate'
```

- (선택 사항) 파라미터가 false로 설정된 경우 기본 인증은 기본적으로 비활성화됩니다. WithBacBasicAuthentication 값을 true로 설정하여 활성화할 수 있습니다.

```
WithBacBasicAuthentication:
    Description: 'false or true for Blue Age Runtime Basic Authentication '
    Default: false
    Type: String
    AllowedValues: [true, false]
```

7. 구성을 완료하면 편집한 CloudFormation 템플릿을 사용하여 스택을 생성할 수 있습니다.

6단계: AWS 블루에이지 Amazon EC2 인스턴스 테스트

CloudFormation 템플릿을 수동으로 실행하여 애플리케이션용 AWS CardDemo Blu Age Amazon EC2 인스턴스를 생성하여 오류 없이 시작되도록 합니다. 이는 템플릿을 애플리케이션 테스트 기능과 함께 사용하기 전에 CloudFormation 템플릿과 모든 사전 요구 사항이 유효한지 확인하기 위한 것입니다.

CloudFormation 그런 다음 애플리케이션 테스트를 사용하여 재생 중에 대상 AWS Blu Age Amazon EC2 인스턴스를 자동으로 생성하고 초기 조건을 비교할 수 있습니다.

1. 스택 CloudFormation 생성 명령을 실행하여 AWS 블루에이지 Amazon EC2 인스턴스를 생성하고 이전 단계에서 편집한 m2 with-ba-using-snapshots - CloudFormation -https-authentication.yml 템플릿을 제공합니다.

```
aws cloudformation create-stack --stack-name load-and-create-ba-snapshots —-
template-url https://apptest-ba-demo.s3.us-west-2.amazonaws.com/m2-with-ba-using-
snapshots-https-authentication.yml --capabilities CAPABILITY_NAMED_IAM --region us-
west-2
```

> **Note**
>
> AWS블루 에이지 AMI가 복원된 정확한 지역을 지정해야 합니다.

2. 콘솔에서 실행 중인 Amazon EC2 인스턴스를 찾아 모든 것이 제대로 작동하는지 확인합니다. Session Manager를 사용하여 연결합니다.
3. Amazon EC2 인스턴스에 연결된 후 다음 명령을 사용합니다.

```
sudo su
cd /m2-anywhere/tomcat.gapwalk/velocity/logs
cat catalina.log
```

4. 로그에 예외나 오류가 없는지 확인합니다.
5. 그런 다음 아래 명령을 사용하여 애플리케이션이 응답하는지 확인합니다.

```
curl http://localhost:8080/gapwalk-application/
```

“Jics 애플리케이션이 실행 중입니다.” 메시지가 표시됩니다.

7단계: 이전 단계가 올바르게 완료되었는지 확인

다음 몇 단계에서는 AWS 메인프레임 현대화 애플리케이션 테스트를 사용하여 애플리케이션에서 생성된 데이터 세트를 재생하고 비교할 것입니다. CardDemo 이 단계를 제대로 수행하려면 자습서의 이전 단계를 모두 성공적으로 완료해야 합니다. 계속하기 전에 다음 사항을 확인하세요.

1. 템플릿을 통해 AWS CloudFormation Amazon EC2 인스턴스의 AWS 블루에이지를 성공적으로 생성했습니다.
2. Amazon EC2의 AWS 블루 에이지에 있는 Tomcat 서비스는 예외 없이 가동되고 있습니다.

애플리케이션과 함께 EC2 인스턴스를 실행하면 CardDemo 애플리케이션 테스트 콘솔에서 다음 단계를 완료하여 재생을 수행하고 배치 데이터 세트를 비교하십시오.

8단계. 초기 조건 생성

이 단계에서는 Amazon EC2에 AWS Blu Age CardDemo 애플리케이션을 배포하는 데 사용한 CloudFormation 템플릿을 제공하여 초기 조건을 생성합니다.

1. https://console.aws.amazon.com/m2/에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Application Testing을 선택합니다.
3. Application Testing에서 초기 조건 생성을 선택합니다.
4. 수정된 Amazon S3 경로와 리소스를 가리키는 스냅샷 ID가 있는 로컬 사본을 사용합니다.

9단계: 테스트 사례 생성

이 단계에서는 CardDemo 애플리케이션에서 만든 데이터 세트를 비교하는 데 사용할 테스트 사례를 만듭니다.

1. 새 테스트 사례를 생성합니다. 이름 및 설명을 입력합니다.
2. `CREASTMT.JCL`을 JCL 이름으로 지정합니다.
3. 테스트 사례 정의에 다음 데이터 세트를 추가합니다.

명칭	CCSID	RecordFormat	RecordLength
AWS.M2.CARDDEMO.STATEMNT.PS	"037"	FB	80
AWS.M2.CARDDEMO.STATEMNT.HTML	"037"	FB	100

> **Note**
>
> JCL 이름과 데이터 세트 세부 정보가 일치해야 합니다.

10단계: 테스트 시나리오 생성

1. 새 테스트 시나리오를 만들고 해당 시나리오의 이름과 설명을 입력합니다.
2. 이전 단계에서 생성한 테스트 사례를 테스트 시나리오에 추가합니다.
3. 테스트 시나리오가 생성되면 테스트 시나리오 개요 페이지에서 1단계에서 만든 초기 조건을 선택합니다.

11단계: 테스트 시나리오 기록

이 단계에서는 소스에서 테스트 사례를 실행합니다. 이를 위해서는 다음 단계를 따릅니다.

1. 애플리케이션의 메인프레임 실행에서 생성된 데이터 세트를 다운로드하고 실행합니다. CardDemo
2. Amazon S3 버킷에 압축을 푼 폴더를 업로드합니다. Amazon S3 버킷은 다른 Application Testing 리소스와 동일한 리전에 있어야 합니다.

> **Note**
>
> 이전 테스트 사례에서 전달한 데이터 세트 이름과 이름이 일치하는 파일이 2개 있어야 합니다.

3. 테스트 시나리오 개요 페이지에서 기록 버튼을 선택합니다.
4. 테스트 시나리오 레코드 페이지에서 소스 메인프레임에서 가져온 데이터 세트를 업로드한 Amazon S3 위치를 지정합니다.
5. 제출을 클릭하여 기록 프로세스를 시작합니다.

> **Note**
>
> 기록이 완료될 때까지 기다렸다가 재생하고 비교해 봅니다.

12단계: 재생 및 비교

Amazon EC2의 대상 AWS AWS Blu Age 환경에서 테스트 시나리오와 테스트 사례를 실행합니다. Application Testing은 재생 시 생성된 데이터 세트를 캡처하여 메인프레임에 기록된 참조 데이터 세트와 비교합니다.

1. 재생 및 비교를 선택합니다. 스택을 생성하고, 비교를 수행하고, CloudFormation 스택을 삭제하는 데 약 3분이 소요됩니다.

모든 작업이 완료되면 이 데모를 위해 의도적으로 생성된 몇 가지 차이점이 포함된 비교 결과를 얻을 수 있습니다.

AWS 메인프레임 현대화 애플리케이션 테스트 지원 데이터 세트 코드 페이지

AWS 애플리케이션 테스트는 AWS 메인프레임 현대화를 위한 프리뷰 릴리스이며 변경될 수 있습니다. 이 기능은 테스트 데이터 및 애플리케이션에만 사용하고, 프로덕션 환경에서는 사용하지 않는 것이 좋습니다.

다음 표를 사용하여 데이터의 코드화된 문자 집합 식별자 (CCSID) 가 애플리케이션 테스트에서 지원되는지 여부를 확인하십시오. AWS 데이터에 지원되지 않는 CCSID가 사용되는 경우 지원되는 CCSID로 변환하거나 당사에 문의하여 도움을 받는 것이 좋습니다.

CCSID	문자 집합	설명
37	IBM037, IBM-037, Cp037	호스트: 미국, 캐나다(ESA), 네덜란드, 포르투갈, 브라질, 호주, 뉴질랜드
273	IBM273, IBM-273, Cp273	호스트: 오스트리아, 독일
277	IBM277, IBM-277, Cp277	호스트: 덴마크, 노르웨이
278	IBM278, IBM-278, Cp278	호스트: 핀란드, 스웨덴
280	IBM280, IBM-280, Cp280	호스트: 이탈리아

CCSID	문자 집합	설명
284	IBM284, IBM-284, Cp284	호스트: 스페인, 라틴 아메리카 (스페인어)
285	IBM285, IBM-285, Cp285	호스트: 영국
297	IBM297, IBM-297, Cp297	호스트: 프랑스
300	IBM-300	JAPAN DB EBCDIC
301	IBM-301	PC 데이터: 일본 DB
437	IBM437, IBM-437, US-ASCII, ASCII, Cp437, US-ASCII	PC 데이터: PC 베이스 미국, 기타 여러 국가
500	IBM500, IBM-500, Cp500	호스트: 벨기에, 캐나다(AS/400), 스위스, 국제 라틴어-1
720	IBM-720	MSDOS ARABIC
737	IBM-737, x-IBM737	MSDOS GREEK
775	IBM775, IBM-775	MSDOS BALTIC
808	IBM-808	PC 데이터: 키릴 문자, 러시아, 유로 표시
813	ISO-8859-7, ISO8859_7	ISO 8859-7: 그리스
819	ISO-8859-1, ISO8859_1	ISO 8859-1: 라틴어-1 국가
833	IBM-808	KOREAN EBCDIC
834	IBM-834, x-IBM834	KOREAN DB EBCDIC
835	IBM-835	T-CHINESE DB EBCD
836	IBM-836	S-CHINESE EBCDIC
837	IBM-837	S-CHINESE EBCDIC

CCSID	문자 집합	설명
850	IBM850, IBM-850, Cp850	PC 데이터: 라틴어-1 국가
855	IBM855, IBM-855, Cp855	PC 데이터: 키릴 문자
856	IBM-856, x-IBM856, Cp856	PC 데이터: 히브리어
858	IBM00858, IBM-858, Cp858	PC 데이터: 라틴어-1 국가, 유로 표시
859	IBM-859	PC 데이터: LATIN-9
860	IBM860, IBM-860	PC 데이터: 포르투갈어
861	IBM861, IBM-861	PC 데이터: 아이슬란드
862	IBM862, IBM-862, Cp862	PC 데이터: 히브리어(마이그레이션)
863	IBM863, IBM-863	PC 데이터: 캐나다
865	IBM865, IBM-865, Cp865	PC 데이터: 덴마크/노르웨이
866	IBM866, IBM-866, Cp866	PC 데이터: 키릴 문자, 러시아
867	IBM-867	PC 데이터: 히브리어, 유로 표시
870	IBM870, IBM-870, Cp870	호스트: 라틴어-2 다국어
871	IBM871, IBM-871, Cp871	호스트: 아이슬란드
874	X-IBM874	PC 데이터: 태국어
875	IBM-856, x-IBM856, Cp856	호스트: 그리스
897	IBM-867	PC 데이터: 일본 SB
912	ISO-8859-1, ISO8859_1	ISO 8859-2: 라틴어-2 다국어
915	ISO-8859-1, ISO8859_1	ISO 8859-5: 키릴 문자

CCSID	문자 집합	설명
916	ISO-8859-1, ISO8859_1	ISO 8859-8: 히브리어
918	IBM918, IBM-918, Cp918	호스트: 우르두어
920	ISO-8859-1, ISO8859_1	ISO 8859-9: 라틴어-5(ECM A-128, 터키 TS-5881)
921	IBM-856, x-IBM856, Cp856	PC 데이터: 라트비아, 리투아니아
922	IBM-856, x-IBM856, Cp856	PC 데이터: 에스토니아
923	ISO-8859-15, Cp923, ISO8859_15_FDIS	ISO 8859-15: 라틴어-9
924	IBM-867	ISO 8859-15: 라틴어-9
927	IBM-867	PC 데이터: T-중국어
930	IBM-930, x-IBM930, Cp930	가타카나 호스트: 확장된 SBCS. 한자 호스트: 4,370개의 사용자 정의 문자를 포함하는 DBCS
932	IBM-932	PC 데이터: 일본 혼합
933	IBM-930, x-IBM930, Cp930	호스트: 확장된 SBCS. 호스트: 1,880개의 사용자 정의 문자와 11,172개의 전체 한글 문자를 포함하는 DBCS
935	IBM-930, x-IBM930, Cp930	호스트: 확장된 SBCS. 호스트: 1,880개의 사용자 정의 문자를 포함하는 DBCS.
937	IBM-930, x-IBM930, Cp930	호스트: 확장된 SBCS. 호스트: 6,204개의 사용자 정의 문자를 포함하는 DBCS

CCSID	문자 집합	설명
939	IBM-930, x-IBM930, Cp930	라틴 호스트: 확장된 SBCS. 한자 호스트: 4,370개의 사용자 정의 문자를 포함하는 DBCS.
942	IBM-942, IBM-942C, X-IBM942, X-IBM942c, Cp942, Cp942C	PC 데이터: 확장된 SBCS. PC 데이터: 1,880개의 사용자 정의 문자를 포함하는 DBCS
943	IBM-943, IBM-943C, Shift_JIS, windows-31j, windows-932, x-IBM943, x-IBM943C, Cp943, Cp943C, MS932	PC 데이터: SBCS. PC 데이터: 1,880개의 IBM 사용자 정의 문자를 포함하는 개방형 환경용 DBCS
947	IBM-947	T-CHINESE BIG-5
948	IBM-948, x-IBM948, Cp948	PC 데이터: 확장된 SBCS. PC 데이터: 6,204개의 사용자 정의 문자를 포함하는 DBCS
949	IBM-949, IBM-949C, x-IBM949, x-IBM949C, Cp949, Cp949C	IBM KS 코드 - PC 데이터: SBCS. IBM KS 코드 - PC 데이터: 1,880개의 사용자 정의 문자를 포함하는 DBCS
950	Big5, IBM-950, x-IBM950, Cp950	PC 데이터: SBCS(IBM BIG5). PC 데이터: 13,493개의 CNS, 566개의 IBM 선택, 6,204개의 사용자 정의 문자를 포함하는 DBCS
951	IBM-951	PC 데이터: IBM KS
954	EUC-JP, IBM-954, IBM-954C	G0: JIS X201 로만. G1: JIS X208-1990. G1: JIS X201 가타카나. G1: JIS X212

CCSID	문자 집합	설명
964	EUC-TW, IBM-964, x-IBM964, Cp964	G0: ASCII. G1: CNS 11643 플레인 1. G1: CNS 11643 플레인 2.
970	EUC-KR, x-IBM970, Cp970	G0: ASCII. G1: 1,880개의 사용자 정의 문자를 포함하는 KSC X5601-1989
971	IBM-971	KOREAN EUC
1006	IBM-1006, x-IBM1006, Cp1006	ISO-8: 우르두어
1025	IBM-1025, x-IBM1025, Cp1025	호스트: 다국어 키릴 문자
1026	IBM1026, IBM-1026, Cp1026	호스트: 라틴어-5(터키)
1027	IBM-1027	JAPAN LATIN EBCD
1041	IBM-1041	PC 데이터: 일본
1043	IBM-1043	PC 데이터: T-중국어
1046	IBM-1046, IBM-1046S, x-IBM1046	ARABIC - PC
1047	IBM1047, IBM-1047	호스트: 라틴어-1
1051	hp-roman8	HP EMULATION
1088	IBM-1088	PC 데이터: 한국 KS
1089	ISO-8859-6, ISO8859_6	ISO 8859-6: 아랍어
1097	IBM-1097, x-IBM1097, Cp1097	호스트: 페르시아어

CCSID	문자 집합	설명
1098	IBM-1098, x-IBM1098, Cp1098	PC 데이터: 페르시아어
1112	IBM-1112, x-IBM1112, Cp1112	호스트: 라트비아, 리투아니아
1114	IBM-1114	PC 데이터: T-CH SB
1115	IBM-1115	PC 데이터: S-CH GB
1122	IBM-1122, x-IBM1122, Cp1122	호스트: 에스토니아
1123	IBM-1123, x-IBM1123, Cp1123	호스트: 우크라이나 키릴 문자
1124	IBM-1124, x-IBM1124, Cp1124	8비트: 키릴 문자, 벨라루스
1140	IBM01140, IBM-1140, Cp1140	호스트: 미국, 캐나다(ESA), 네덜란드, 포르투갈, 브라질, 호주, 뉴질랜드, 유로 표시
1141	IBM01141, IBM-1141, Cp1141	호스트: 오스트리아, 독일, 유로 표시
1142	IBM01142, IBM-1142, Cp1142	호스트: 덴마크, 노르웨이, 유로 표시
1143	IBM01143, IBM-1143, Cp1143	호스트: 핀란드, 스웨덴, 유로 표시
1144	IBM01144, IBM-1144, Cp1144	호스트: 이탈리아, 유로 표시
1145	IBM01145, IBM-1145, Cp1145	호스트: 스페인, 라틴 아메리카(스페인어), 유로 표시
1146	IBM01146, IBM-1146, Cp1146	호스트: 영국, 유로 표시

CCSID	문자 집합	설명
1147	IBM01147, IBM-1147, Cp1147	호스트: 프랑스, 유로 표시
1148	IBM01148, IBM-1148, Cp1148	호스트: 벨기에, 캐나다(AS/400), 스위스, 국제 라틴어-1, 유로 표시
1149	IBM01149, IBM-1149, Cp1149	호스트: 아이슬란드, 유로 표시
1200	UTF-16BE	문자 집합이 65535인 유니코드. 바이트 순서 표시(BOM)가 없는 경우 UTF-16 BE(big-endian)로 가정.
1202	UTF-16LE	UTF-16 LE(IBM PUA 포함)
1204	UTF-16	UTF-16(IBM PUA 포함)
1208	UTF-8, UTF-8J, UTF8	문자 집합이 65535인 유니코드. UTF-8.
1232	UTF-32BE	UTF-32 BE(IBM PUA 포함)
1234	UTF-32LE	UTF-32LE(IBM PUA 포함)
1236	UTF-32	UTF-32(IBM PUA 포함)
1351	IBM-1351	JAPAN OPEN
1362	IBM-1362	KOREAN MS-WIN
1363	IBM-1363, IBM-1363C, windows-949, MS949	PC 데이터: MS Windows 한국어 SBCS. PC 데이터: MS Windows 코란 DBCS(11,172개의 전체 한글 포함)

CCSID	문자 집합	설명
1364	IBM-1364	호스트: 확장된 SBCS. 호스트: 1,880개의 사용자 정의 문자와 11,172개의 전체 한글 문자를 포함하는 DBCS
1370	IBM-1370	PC 데이터: 확장된 SBCS, 유로 표시. PC 데이터: 6,204개의 사용자 정의 문자를 포함하는 DBCS, 유로 표시
1371	IBM-1371	호스트: 확장된 SBCS, 유로 표시. 호스트: 6,204개의 사용자 정의 문자를 포함하는 DBCS, 유로 표시
1375	Big5-HKSCS	HKSCS용 혼합 Big-5 Ext
1380	IBM-1380	PC 데이터: S-CH GB
1381	IBM-1381, x-IBM1381, Cp1381	PC 데이터: 확장된 SBCS(IBM GB). PC 데이터: 31개의 IBM 선택, 1,880개의 사용자 정의 문자를 포함하는 DBCS(IBM GB)
1382	IBM-1382	S-CHINESE EUC
1383	EUC-CN, GB2312, IBM-1383, x-IBM1383, Cp1383	G0: ASCII. G1: GB 2312-80 세트
1385	IBM-1385	PC 데이터: S-CH GBK
1386	GBK, IBM-1386, windows-936, MS936	PC 데이터: S-중국어 GBK 및 T-중국어 IBM BIG-5. PC 데이터: S-중국어 GBK

CCSID	문자 집합	설명
1388	IBM-1388	호스트: 확장된 SBCS. 호스트: 1,880개의 사용자 정의 문자를 포함하는 DBCS
1390	IBM-1390	가타카나 호스트: 확장된 SBCS, 유로 표시. 한자 호스트: 6,205개의 사용자 정의 문자를 포함하는 DBCS
1399	IBM-1399	라틴 호스트: 확장된 SBCS, 유로 표시. 한자 호스트: 4,370개의 사용자 정의 문자를 포함하는 DBCS, 유로 표시
5050	JIS0201, JIS0208, JIS0212, JIS0201, JIS0208, JIS0212	G0: JIS X201 로만. G1: JIS X208-1990. G1: JIS X201 가타카나. G1: JIS X212
5054	ISO-2022-JP	JAPANESE TCP
5346	windows-1250, Cp1250	MS Windows: 라틴어-2, 유로 표시 버전 2
5347	windows-1251, Cp1251	MS Windows: 키릴 문자, 유로 표시 버전 2
5348	windows-1252, Cp1252	MS Windows: 라틴어-1 국가, 유로 표시 버전 2
5349	windows-1252, Cp1252	MS Windows: 그리스, 유로 표시 버전 2
5350	windows-1252, Cp1252	MS Windows: 튀르키예, 유로 표시 버전 2
5351	windows-1255, Cp1255	MS Windows: 히브리어, 유로 표시 버전 2

CCSID	문자 집합	설명
5352	windows-1256, windows-1256S, Cp1256	MS Windows: 아랍어, 유로 표시 버전 2
5353	windows-1257, Cp1257	MS Windows: 발트해 연안국가, 유로 표시 버전 2
5354	windows-1258, Cp1258	MS Windows: 베트남어, 유로 표시 버전 2
5488	GB18030	GB18030, 1바이트 데이터 GB18030, 2바이트 데이터 GB18030, 4바이트 데이터
9030	IBM-838, Cp838	호스트: 태국어 확장 SBCS
9066	IBM-874, Cp874	PC 데이터: 태국어 확장 SBCS
9400	CESU-8	CESU-8(IBM PUA 포함)
25546	ISO-2022-KR	KOREAN TCP
33722	IBM-33722, IBM-33722C	IBMeucJP

AWS 메인프레임 현대화에서의 파일 전송

AWS 메인프레임 현대화 File Transfer를 사용하면 메인프레임 데이터 세트를 Amazon S3로 전송 및 변환하여 메인프레임 현대화, 마이그레이션 및 증강 사용 사례를 활용할 수 있습니다.

주제

AWS Mainframe Modernization File Transfer란?

AWS 메인프레임 현대화 파일 전송을 사용하면 완전관리형 서비스로 데이터 세트와 파일을 전송 및 변환하여 AWS 메인프레임 현대화 서비스 및 Amazon S3로의 현대화, 마이그레이션 및 증강 사용 사례를 가속화하고 간소화할 수 있습니다.

주제

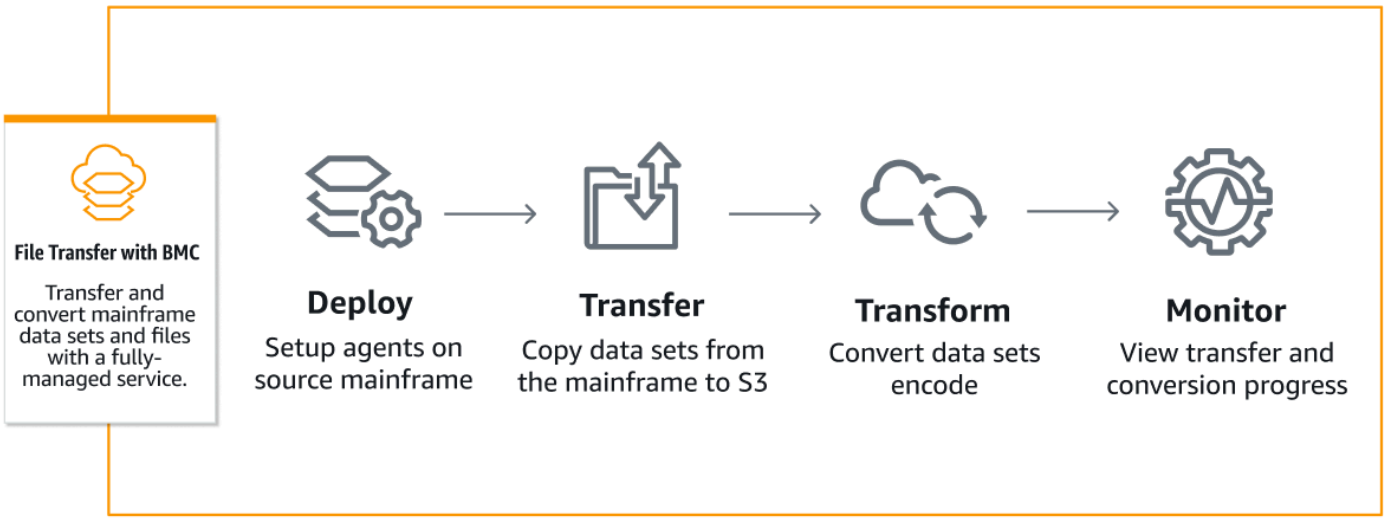
AWS Mainframe Modernization File Transfer의 이점

AWS Mainframe Modernization File Transfer를 사용하면 메인프레임에서 Amazon S3로 데이터 세트를 전송할 수 있습니다. 다음은 몇 가지 예입니다.

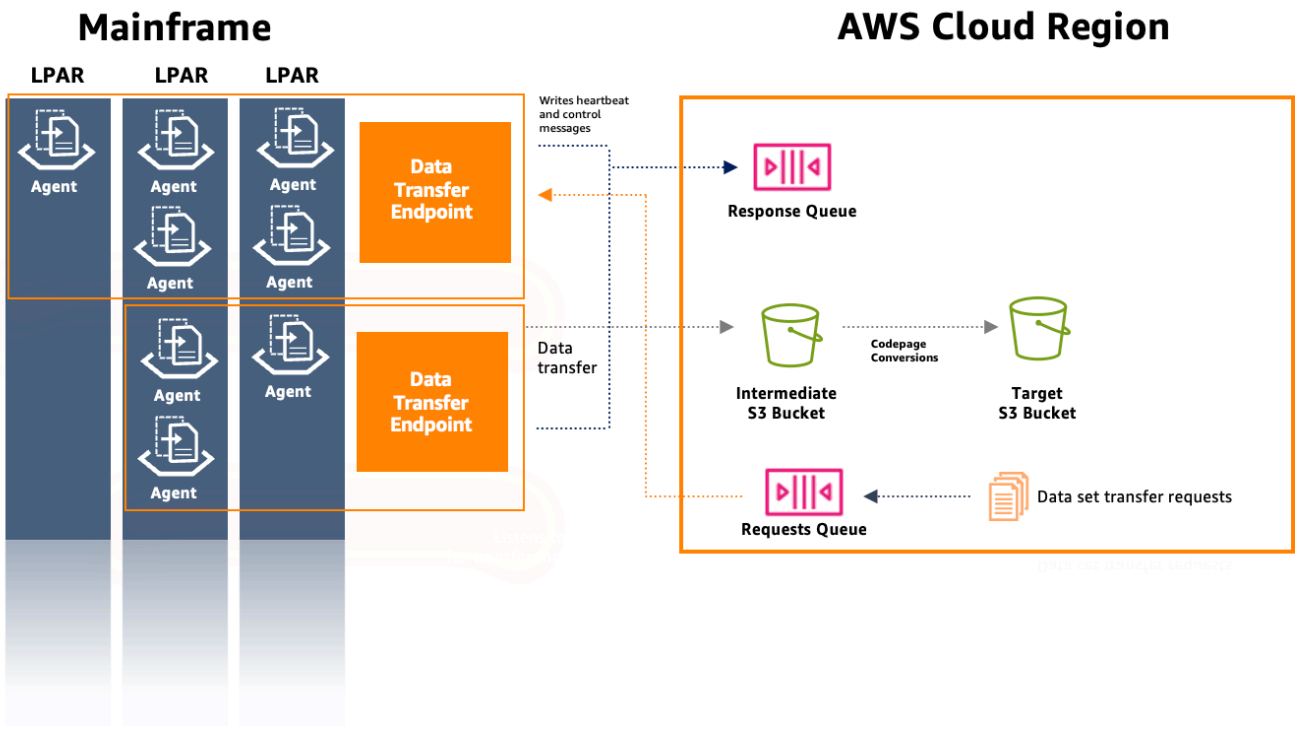
- 소스 메인프레임 데이터 세트 및 아티팩트 검색
- 자동 전송 및 데이터 세트 변환
- AWS로 데이터 세트를 더 빠르게 전송할 수 있는 확장성, 효율성 및 속도

AWS Mainframe Modernization File Transfer 작동 방식

다음 그림은 AWS Mainframe Modernization File Transfer가 작동하는 방식을 개념적 수준에서 개괄적으로 보여줍니다.



다음 그림은 AWS Mainframe Modernization File Transfer 기능의 아키텍처 개요입니다.



File Transfer 에이전트 설치

이 가이드에 step-by-step 따라 소스 메인프레임에 에이전트를 설치하고 에이전트를 구성하기 위한 사전 요구 사항을 완료하십시오.

주제

1단계: ISPF에 로그인

Interactive System Productivity Facility(ISPF) 세션에 로그인하세요. 이는 일반적으로 3270 터미널 에뮬레이터를 통해 이루어집니다.

2단계: z/FS에 데이터 세트 할당

ISPF의 데이터 세트 유틸리티를 사용하여 z/FS에 새 데이터 세트를 할당합니다. 일반적으로 이 작업은 1단계의 데이터 세트 목록 유틸리티에서 수행됩니다.

1. 옵션 3.4(유틸리티 --> 데이터 세트)로 이동합니다.
2. 'C' 키를 눌러 새로운 데이터 세트를 만듭니다.
3. 데이터 세트 이름을 입력합니다(예: 'yourhlq.M2AGENT.ZFS').
4. 데이터 세트 유형을 기본 크기가 1,000개 실린더이고 보조 크기가 200개인 '대형 포맷'으로 지정합니다.
5. 데이터 세트 구성(DSORG)을 PS로 설정하고 레코드 형식(RECFM)을 'U'(미정)로 설정합니다.
6. 생성 프로세스를 완료합니다.

3단계: 데이터 세트를 z/FS로 포맷

데이터 세트를 만든 후 z/FS 파일 시스템으로 포맷합니다.

이를 위한 한 방법은 다음과 같은 작업 제어 언어(JCL)를 사용하는 것입니다.

```
//FORMAT  EXEC PGM=IOEAGFMT,PARM='AGGRNAME(yourhlq.M2AGENT.ZFS),FORMAT,AGGRSIZE(1200)'
//SYSPRINT DD  SYSOUT=A
```

이 작업을 제출하고 성공적으로 완료되었는지 확인하세요.

4단계: 파일 시스템을 z/OS로 정의

`DEF FILESYSTEM` 명령을 사용하거나 배치 작업을 통해 z/FS를 z/OS로 정의합니다. 다음 명령을 사용합니다.

```
DEF FILESYSTEM('yourhlq.M2AGENT.ZFS') TYPE(ZFS) MODE(R/W) MOUNTPOINT('/usr/lpp/aws/m2-agent')
```

> **Note**
>
> 이 단계에는 시스템 수준의 권한이 필요합니다.

5단계: 파일 시스템 탑재

파일 시스템을 탑재하려면 `MOUNT` 명령을 사용합니다. ISPF의 명령줄에서 파일 시스템을 탑재하거나 일괄적으로 탑재할 수 있습니다.

예:

```
MOUNT FILESYSTEM('yourhlq.M2AGENT.ZFS') TYPE(ZFS) MODE(RDWR) MOUNTPOINT('/usr/lpp/aws/
m2-agent')
```

6단계: 탑재 확인

D OMVS,A 명령을 사용하거나 Unix 시스템 서비스(USS) 내에서 확인하여 파일 시스템이 올바르게 탑재되었는지 확인합니다.

7단계: OMV 입력

다음 명령을 사용하여 OMV를 입력합니다.

```
TSO OMVS
```

8단계: 에이전트 설치 디렉터리 환경 변수 설정

다음 명령을 사용하여 에이전트 설치 디렉터리 환경을 설정합니다.

```
export AGENT_DIR=/usr/lpp/aws/m2-agent
```

9단계: 작업 디렉터리 환경 변수 설정

다음 명령을 사용하여 작업 디렉터리 환경 변수를 설정합니다.

```
export WORK_DIR=$AGENT_DIR/tmp
```

10단계: 작업 디렉터리 생성

다음 명령을 사용하여 작업 디렉터리 환경을 설정합니다.

```
mkdir -p $WORK_DIR
```

11단계: AWS 메인프레임 현대화 tar 패키지를 z/OS의 작업 디렉터리에 복사

FTP나 다른 전송 방법을 사용할 때는 tar 파일이 바이너리 모드로 전송되는지 확인합니다.

12단계: 루트 사용자 수임

다음 명령을 사용하여 루트 사용자를 수임합니다.

```
su
```

다음 단계에 따라 에이전트 설치를 완료합니다.

> **Note**
>
> 이 단계를 계속하려면 먼저 루트 사용자를 수임해야 합니다.

1. 다음 명령을 사용하여 m2-agent 버전 환경 변수를 현재 설치 중인 버전으로 설정합니다.

   ```
   export M2_AGENT_VERSION=1.0.0
   ```

2. 다음 명령을 사용하여 에이전트 tar 패키지를 추출합니다.

   ```
   tar -xpf m2-agent-package-$M2_AGENT_VERSION.tar -C $AGENT_DIR
   ```

3. 다음 명령을 사용하여 현재 에이전트 설치 디렉터리에 대한 `current-version` 심볼릭 링크를 생성합니다.

   ```
   ln -s $AGENT_DIR/m2-agent-v$M2_AGENT_VERSION $AGENT_DIR/current-version
   ```

4. CPY#PDS를 업데이트하고 제출하여 File Transfer 에이전트 데이터 세트를 생성합니다.

   > **Note**
   >
   > JCL은 SYS2.AWS.M2 HLQ를 사용합니다.

   File Transfer 에이전트를 생성하려면 파라미터 행 000006-000012를 설정합니다. 또한 3개의 심볼릭 변수 HLQ, VOLSER, AGNTPATH를 업데이트하여 나중에 JCL에서 사용할 수 있도록 합니다.

   ```
   oedit $AGENT_DIR/current-version/installation/CPY#PDS
   submit $AGENT_DIR/current-version/installation/CPY#PDS
   ```

> **Note**
>
> 이 JCL은 메인프레임에 에이전트 설치의 특정 측면을 설정하는 데 맞게 조정되었습니다. 필요한 데이터 세트를 할당한 다음, Unix 파일 시스템의 특정 파일을 해당 데이터 세트로 복사합니다.

권한 및 STC 구성

1. 지침에 따라 `SYS2.AWS.M2.SAMPLIB(SEC#RACF)`(RACF 권한 설정용) 또는 `SYS2.AWS.M2.SAMPLIB(SEC#TSS)`(TSS 권한 설정용) 중 하나를 업데이트하고 제출합니다. 이러한 멤버는 이전 `CPY#PDS` 단계에서 생성되었습니다.
2. 기본 File Transfer 에이전트 디렉터리 경로(`/usr/lpp/aws/m2-agent`)가 변경된 경우 `SYS2.AWS.M2.SAMPLIB(M2AGENT)` STC JCL에서 PWD 내보내기를 업데이트합니다.
3. `SYS2.AWS.M2.SAMPLIB(M2AGENT)` JCL을 업데이트하고 `SYS1.PROCLIB`로 복사합니다.
4. 다음 명령을 사용하여 APF 목록에 `SYS2.AWS.M2.LOADLIB`를 추가합니다.

```
SETPROG APF ADD DSNAME(SYS2.AWS.M2.LOADLIB) SMS
```

5. 에이전트의 `logs` 및 `diag` 폴더의 그룹과 소유자를 에이전트 사용자/그룹(M2USER/M2GROUP)으로 설정합니다. 다음 명령을 사용합니다.

```
chown -R M2USER:M2GROUP $AGENT_DIR/current-version/logs
chown -R M2USER:M2GROUP $AGENT_DIR/current-version/diag
```

장기 액세스 보안 인증 정보를 사용하여 IAM 사용자 생성

IAM 사용자가 응답 및 요청 대기열을 사용하고 Amazon S3 버킷에 데이터 세트를 저장하는 데 필요한 메인프레임 에이전트를 생성해야 합니다.

이 사용자를 생성할 때는 다음 단계를 따릅니다.

1. 권한 옵션에서 바로 정책 연결을 선택합니다.
2. 사용자가 생성되면 보안 자격 증명 탭을 열고 액세스 키를 생성합니다. IAM 액세스 키를 생성하는 방법에 대한 자세한 내용은 IAM 사용자의 액세스 키 관리를 참조하세요.
3. 액세스 키 섹션에서 사용 사례를 묻는 메시지가 표시되면 기타를 선택합니다.

> **Note**
>
> 완료를 선택하기 전에 액세스 키 생성 마법사의 마지막 페이지에 표시된 액세스 키와 비밀 액세스 키를 저장합니다. 이러한 키는 메인프레임 에이전트를 구성하는 데 사용됩니다.

> **Note**
>
> IAM 역할과 신뢰 관계를 설정하는 데 사용되는 IAM 사용자 ARN을 저장합니다.

에이전트가 수임할 IAM 역할 생성

신뢰할 수 있는 엔터티 유형에 대한 사용자 지정 신뢰 정책으로 새 IAM 역할을 생성합니다. 정책은 다음 템플릿을 사용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DataTransferEndpointAgentSqsReceive",
            "Effect": "Allow",
            "Action": [
                "sqs:DeleteMessage",
                "sqs:ReceiveMessage"
            ],
            "Resource": "<data-transfer-endpoint-request-queue-arn>"
        },
        {
            "Sid": "DataTransferEndpointS3",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "<data-transfer-endpoint-intermediate-bucket-arn>/*"
        },
        {
            "Sid": "DataTransferEndpointAgentSqsSend",
            "Effect": "Allow",
            "Action": "sqs:SendMessage",
            "Resource": "<data-transfer-endpoint-response-queue-arn>"
        },
        {
```

```
            "Sid": "DataTransferEndpointAgentKmsDecrypt",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "<kms-key-id>"
        }
    ]
}
```

위치:

- `request-queue-arn` 및 `response-queue-arn`은 데이터 전송 엔드포인트 초기화 중에 생성된 요청 Amazon SQS 대기열의 ARN입니다.
- `transfer-bucket-arn`은 이전에 만든 전송 버킷의 ARN입니다.

> **Note**
>
> AWS 콘솔을 사용하여 이러한 모든 값을 조회할 수 있습니다.

> **Note**
>
> 나중에 메인프레임 에이전트를 구성할 때 사용할 역할 이름을 저장합니다.

에이전트 구성

File Transfer 에이전트를 구성하려면:

1. `$AGENT_DIR/current-version/config`로 이동합니다.
2. 다음 명령을 통해 에이전트의 구성 파일 `appication.properties`를 편집하여 환경 구성을 추가합니다.

   ```
   oedit $AGENT_DIR/current-version/config/application.properties
   ```

   예:

   ```
   agent.environments[0].account-id=<AWS_ACCOUNT_ID>
   agent.environments[0].agent-role-name=<AWS_IAM_ROLE_NAME>
   ```

```
agent.environments[0].access-key-id=<AWS_IAM_ROLE_ACCESS_KEY>
agent.environments[0].secret-access-id=<AWS_IAM_ROLE_SECRET_KEY>
agent.environments[0].bucket-name=<AWS_S3_BUCKET_NAME>
agent.environments[0].environment-name=<AWS_REGION>
agent.environments[0].region=<AWS_REGION>
```

위치:

- AWS_ACCOUNT_ID는 고객 계정의 ID입니다.
- AWS_IAM_ROLE_NAME은 the section called “에이전트가 수임할 IAM 역할 생성”에서 생성한 IAM 역할의 이름입니다.
- AWS_IAM_ROLE_ACCESS_KEY는 the section called “장기 액세스 보안 인증 정보를 사용하여 IAM 사용자 생성”에서 생성한 IAM 사용자의 액세스 키입니다.
- AWS_IAM_ROLE_SECRET_KEY는 the section called “장기 액세스 보안 인증 정보를 사용하여 IAM 사용자 생성”에서 생성한 IAM 사용자의 액세스 비밀 키입니다.
- AWS_S3_BUCKET_NAME은 데이터 전송 엔드포인트로 생성된 전송 버킷의 이름입니다.
- AWS_REGION은 File Transfer 에이전트를 구성하는 리전입니다.

> **Note**
>
> 괄호 안의 인덱스([0])가 각각에 대해 증가하는 한, 이러한 섹션이 여러 개 있을 수 있습니다.

변경 사항을 적용하려면 에이전트를 다시 시작해야 합니다.

요구 사항

1. 파라미터를 추가하거나 제거한 경우 에이전트를 중지하고 시작해야 합니다. CLI에서 다음 명령을 사용하여 File Transfer 에이전트를 시작합니다.

```
/S M2AGENT
```

M2 에이전트를 중지하려면 CLI에서 다음 명령을 사용합니다.

```
/P M2AGENT
```

2. 여러 환경을 정의하여 File Transfer 에이전트가 여러 지역 및 계정으로 전송하도록 할 수 있습니다 AWS .

```
#Region 1
agent.environments[0].account-id=AWS_ACCOUNT_ID
agent.environments[0].agent-role-name=AWS_IAM_ROLE_NAME
agent.environments[0].access-key-id=AWS_IAM_ROLE_ACCESS_KEY
agent.environments[0].secret-access-id=AWS_IAM_ROLE_SECRET_KEY
agent.environments[0].bucket-name=AWS_S3_BUCKET_NAME
agent.environments[0].environment-name=AWS_REGION
agent.environments[0].region=AWS_REGION

#Region 2
agent.environments[1].account-id=AWS_ACCOUNT_ID
agent.environments[1].agent-role-name=AWS_IAM_ROLE_NAME
agent.environments[1].access-key-id=AWS_IAM_ROLE_ACCESS_KEY
agent.environments[1].secret-access-id=AWS_IAM_ROLE_SECRET_KEY
agent.environments[1].bucket-name=AWS_S3_BUCKET_NAME
agent.environments[1].environment-name=AWS_REGION
agent.environments[1].region=AWS_REGION
```

데이터 전송 엔드포인트

데이터 전송 엔드포인트는 소스 메인프레임에서 에이전트의 고가용성, 확장성 및 간소화된 관리를 가능하게 합니다. 개별 에이전트는 메인프레임 LPAR에 설치되며, 데이터 전송 엔드포인트로 그룹화할 수 있습니다. 데이터 세트 전송 요청이 들어오면 데이터 전송 엔드포인트의 에이전트 하나가 전송을 처리합니다. 데이터 전송을 시작하려면 데이터 전송 엔드포인트에 있는 에이전트가 하나 이상 온라인 상태여야 합니다.

이 절차에서는 AWS 메인프레임 현대화 설정의 단계와 소스 메인프레임에서 File Transfer 에이전트 구성을 완료했다고 가정합니다.

데이터 전송 엔드포인트 생성

파일 전송을 위한 데이터 전송 엔드포인트를 생성하려면 메인프레임 현대화 콘솔에서 다음 단계를 따라야 합니다. AWS

데이터 전송 엔드포인트를 생성하려면

1. https://console.aws.amazon.com/m2/ 에서 AWS 메인프레임 현대화 콘솔을 엽니다.

2. AWS 리전 선택기에서 메인프레임에서 Amazon S3 버킷으로 파일을 전송할 지역을 선택합니다.
3. 데이터 전송 엔드포인트 페이지의 File Transfer에서 데이터 전송 엔드포인트 생성을 선택합니다.
4. 데이터 전송 엔드포인트 사전 조건 페이지에서 모든 지침을 읽고 해당 단계를 완료했는지 확인합니다. 확인이 완료되면 다음을 선택합니다.
5. 데이터 전송 엔드포인트 구성 페이지에서 데이터 전송 엔드포인트에 대한 기본 정보를 추가합니다.

    1. 기본 정보 섹션에서 데이터 전송 엔드포인트 이름, 설명 및 KMS 키를 입력합니다. KMS 키에 대한 자세한 내용은 키 생성을 참조하세요.

    > **Note**
    >
    > 데이터 전송 엔드포인트 이름은 소스 메인프레임에서 File Transfer 에이전트를 구성할 때 정의된 이름과 일치해야 합니다.

    > **Note**
    >
    > AWS 메인프레임 현대화 서비스가 암호화/복호화에 이러한 키를 읽고 사용할 수 있도록 하려면 KMS에 대해 다음과 같은 리소스 기반 정책을 추가해야 합니다.

    ```
    {
      "Sid" : "Enable AWS M2 Permissions",
      "Effect" : "Allow",
      "Principal" : {
          "Service" : "m2.amazonaws.com"
                },
       "Action" : [
           "kms:Encrypt",
           "kms:Decrypt"
                  ],
       "Resource" : "*"
    }
    ```

    2. 중간 데이터용 S3 위치, 즉 메인프레임에서 전송된 데이터 세트가 저장되는 중간 S3 위치를 지정합니다.

> **Note**
>
> 전송 작업을 위한 새 Amazon S3 버킷을 생성하는 것이 좋습니다. 자세한 내용은 버킷 생성을 참조하세요. S3 찾아보기 옵션을 선택하여 기존 Amazon S3 버킷을 찾아볼 수도 있습니다.

3. 필수 필드를 입력한 후 다음을 선택합니다.

6. 데이터 전송 엔드포인트 검토 및 생성 페이지에서 사전 조건을 완료했는지 확인하고 기본 정보를 검토합니다. 확인했으면 생성 및 연결을 선택합니다.
7. 연결 확인 페이지에서 에이전트 연결이 설정되면 모든 에이전트가 ID 및 하트비트와 함께 표시됩니다. 연결이 설정되면 “데이터 전송 엔드포인트가 성공적으로 생성되었습니다.”라는 메시지가 표시됩니다.

> **Note**
>
> “에이전트 연결을 설정하지 못했습니다.”라는 메시지가 표시되면 4단계를 다시 확인하여 필요한 사전 조건을 모두 완료했는지 살펴봅니다.

8. 마침을 클릭합니다.

모든 데이터 전송 엔드포인트 목록을 볼 수 있는 데이터 전송 엔드포인트 개요 페이지로 리디렉션됩니다. 사용 가능하거나 실패한 데이터 전송 엔드포인트도 확인할 수 있습니다.

또한 이름별로 데이터 전송 엔드포인트를 검색하고 사용 가능한 각 에이전트에 대한 추가 정보에 액세스할 수 있습니다.

전송 작업

전송 작업은 데이터 세트의 소스 인코딩 및 대상 인코딩을 정의하는 데 사용됩니다. 소스 인코딩은 원본 데이터 세트의 형식이고, 대상 인코딩은 이러한 데이터 세트가 대상 Amazon S3 버킷에 저장되는 형식입니다. 이러한 대상 버킷은 전송 작업에 의해 정의됩니다.

이 절차에서는 AWS 메인프레임 현대화 설정의 단계를 완료했다고 가정하며 the section called “데이터 전송 엔드포인트”를 설정합니다.

주제

- 전송 작업 생성

- 전송 작업 보기

전송 작업 생성

파일 전송을 위한 전송 작업을 생성하려면 AWS 메인프레임 현대화 콘솔에서 다음 단계를 따라야 합니다.

전송 작업을 생성하려면

> **Note**
> 새 전송 작업을 생성하려면 하나 이상의 데이터 전송 엔드포인트가 있어야 합니다.

1. https://console.aws.amazon.com/m2/ 에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 메인프레임에서 Amazon S3 버킷으로 파일을 전송할 지역을 선택합니다.
3. 전송 작업 페이지의 파일 전송에서 데이터 전송 엔드포인트를 선택하여 전송 작업을 생성합니다.
4. 데이터 전송 엔드포인트에 전송 작업이 없는 경우 전송 작업 생성을 선택하여 새 작업을 생성할 수 있습니다.
5. 전송 작업 생성 페이지에서 전송 작업의 속성을 설정합니다.
   - 이 페이지에서 전송 작업 이름, 설명, 비밀 키, 데이터 세트 검색 기준 등 전송 작업의 기본 정보를 입력합니다.

> **Note**
> - 데이터 전송 엔드포인트에 정의된 KMS 키를 사용하여 암호를 암호화합니다. 또한 암호에는 `userId` 및 `password` 키를 사용하여 메인프레임의 데이터 세트에 액세스하는 데 필요한 메인프레임 보안 인증 정보가 포함되어야 합니다. 자세한 내용은 AWS Secrets Manager 시크릿을 참조하십시오.
> - AWS 메인프레임 현대화 서비스가 보안 키에 액세스하여 데이터 전송 작업을 수행할 수 있도록 하려면 다음 리소스 기반 정책으로 보안 키를 구성해야 합니다.

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

```
    "Principal" : {
        "Service" : "m2.amazonaws.com"
    },
    "Action" : [ "secretsmanager:GetSecretValue",
                 "secretsmanager:DescribeSecret" ],
    "Resource" : "*"
  } ]
}
```

> **Note**
>
> 현재 지원되는 최대 데이터 세트 크기는 90GiB입니다.

- 파일의 대상 Amazon S3 버킷 위치를 입력하거나 찾아봅니다.
- 기본 데이터 전송 엔드포인트가 선택됩니다. 사용 가능한 엔드포인트에서 엔드포인트를 변경하도록 선택할 수도 있습니다.

6. 다음을 선택합니다.
7. 데이터 세트 선택 페이지에서 선택한 각 데이터 세트의 소스 인코딩 및 대상 인코딩을 수동으로 업데이트해야 합니다. 소스 인코딩은 원본 데이터세트 형식이고 대상 인코딩은 대상 데이터세트 형식이며 데이터 세트를 변환하는 데 사용됩니다.
8. 소스 및 대상 인코딩을 확인한 후 다음을 선택합니다.
9. 검토 및 생성 페이지에서 전송 작업에 대한 정보를 검토하거나 편집할 수 있습니다.
10. 전송 작업 생성을 선택합니다.

“전송 작업이 성공적으로 생성되었습니다.”라는 메시지가 표시됩니다.

전송 작업 보기

파일 전송을 위한 전송 작업을 보려면 AWS 메인프레임 현대화 콘솔에서 다음 단계를 따라야 합니다.

전송 작업을 보려면

1. https://console.aws.amazon.com/m2/ 에서 AWS 메인프레임 현대화 콘솔을 엽니다.
2. AWS 리전 선택기에서 메인프레임에서 Amazon S3 버킷으로 파일을 전송할 지역을 선택합니다.
3. 전송 작업 페이지의 파일 전송에서 전송 작업을 확인할 데이터 전송 엔드포인트를 선택합니다.

4. 기존 전송 작업이 있는 엔드포인트의 경우 전송 작업 섹션 아래에 해당 작업이 채워집니다. 이 목록에서 모든 전송 작업의 세부 정보를 보도록 선택할 수 있습니다.

자습서: AWS Mainframe Modernization File Transfer 시작하기

AWS 메인프레임 현대화 파일 전송을 사용하면 메인프레임 현대화, 마이그레이션 및 증강 사용 사례에 맞게 메인프레임 데이터 세트를 전송 및 변환할 수 있습니다.

이 자습서의 단계를 따라 하며 AWS Mainframe Modernization File Transfer의 작동 방식을 파악해 보세요.

개요

File Transfer는 다음과 같이 구성됩니다.

1. 소스 메인프레임에 설치할 에이전트.
2. 메인프레임 현대화 관리 서비스 콘솔에서 직접 데이터 세트 검색, 전송 및 변환 기능에 액세스할 수 있습니다. AWS

사용자는 메인프레임에서 Amazon S3 버킷으로 데이터 세트를 전송할 수 있습니다.

주제

- 1단계: 에이전트 바이너리 tar 패키지를 메인프레임 논리적 파티션으로 전송 AWS
- 2단계: 소스 메인프레임에 File Transfer 에이전트 구성
- 3단계: 데이터 전송 엔드포인트 생성
- 4단계: 전송 작업 생성
- 5단계: 전송 작업 진행 상황 보기

1단계: 에이전트 바이너리 tar 패키지를 메인프레임 논리적 파티션으로 전송 AWS

M2-agent tar 링크에서 tar 파일을 다운로드합니다.

2단계: 소스 메인프레임에 File Transfer 에이전트 구성

이 단계에서는 소스 메인프레임에서 AWS Mainframe Modernization File Transfer 에이전트를 구성하고 시작합니다. 에이전트는 File Transfer 서비스 기능과 소스 메인프레임 간의 통신을 용이하게 할 수 있어야 합니다. 메인프레임당 하나 이상의 에이전트가 필요합니다. 고가용성과 향상된 확장성을 위해 둘 이상의 에이전트를 시작할 수 있습니다.

the section called “File Transfer 에이전트 설치” 안내서의 지침에 따라 메인프레임에 File Transfer 에이전트 설치를 완료합니다.

3단계: 데이터 전송 엔드포인트 생성

the section called “데이터 전송 엔드포인트” 페이지의 단계에 따라 새 데이터 전송 엔드포인트를 생성합니다.

4단계: 전송 작업 생성

the section called “전송 작업” 페이지의 단계에 따라 전송 작업을 생성하고 관리합니다.

5단계: 전송 작업 진행 상황 보기

AWS 메인프레임 현대화 콘솔에서 전송 작업의 진행 상황을 볼 수 있습니다. 자세한 내용은 the section called “전송 작업 보기” 섹션을 참조하세요.

AWS 메인프레임 현대화의 보안

AWS에서 클라우드 보안을 가장 중요하게 생각합니다. AWS 고객은 보안에 매우 민감한 조직의 요구 사항에 부합하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS와 귀하의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드의 보안 - AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호합니다. AWS는 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램의 일환으로 보안 효과를 정기적으로 테스트하고 검증합니다. AWS 메인프레임 현대화에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하세요.
- 클라우드 내 보안 - 사용자의 책임은 사용하는 AWS 서비스에 의해 결정됩니다. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 AWS 메인프레임 현대화 사용 시 책임 분담 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 보안 및 규정 준수 목표에 맞게 AWS 메인프레임 현대화를 구성하는 방법을 보여줍니다. 또한 AWS 메인프레임 현대화 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

AWS 메인프레임 현대화는 IAM 정책에 따라 모든 작업을 허용해야 하는 AWS 메인프레임 현대화 관리 리소스인 자체 IAM 보호 리소스(애플리케이션, 환경, 배포 등)를 제공합니다.

플랫폼 재구성을 위한 AWS 메인프레임 현대화도 IAM에 의해 보호됩니다. IAM은 또한 표준 IAM 정책을 통해 원래 메인프레임 환경에서 파생된 정의된 리소스에 대한 특정 작업에 대해 보안 주체에게 권한을 부여하거나 거부합니다. AWS 메인프레임 현대화 플랫폼 변경 런타임은 애플리케이션이 보호된 리소스에서 이러한 작업을 시도할 때 IAM 권한 부여 서비스를 호출합니다. IAM은 표준 IAM 정책 평가 메커니즘에 따라 허용 또는 거부를 반환합니다.

목차

- 인터페이스 엔드포인트(AWS PrivateLink)를 사용하여 AWS Mainframe Modernization에 액세스

AWS 메인프레임 현대화의 데이터 보호

AWS 공동 책임 모델 공동 책임 모델 데이터 보호에 적용됩니다. AWS 이 모델에 설명된 대로 AWS 는 모든 것을 실행하는 글로벌 인프라를 보호하는 역할을 합니다. AWS 클라우드이 인프라에서 호스팅되는 콘텐츠에 대한 제어를 유지하는 것은 사용자의 책임입니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center OR AWS Identity and Access Management (IAM) 을 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이 방식을 사용하면 각 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 리소스와 통신하세요. AWS TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
- Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하여 Amazon S3에 저장된 민감한 데이터를 검색하고 보호합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-2로 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용하십시오. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하십시오.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔, API 또는 SDK를 AWS 서비스 사용하여 AWS 메인프레임 현대화 또는 기타 작업을 수행하는 경우가 포함됩니다. AWS CLI AWS 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 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

메인프레임 현대화가 수집하는 데이터 AWS

AWS 메인프레임 현대화는 사용자로부터 여러 유형의 데이터를 수집합니다.

- `Application configuration`: 애플리케이션을 구성하기 위해 생성하는 JSON 파일입니다. 여기에는 AWS 메인프레임 현대화가 제공하는 다양한 옵션에 대한 선택 사항이 포함되어 있습니다. 이 파일에는 애플리케이션 아티팩트가 저장되는 Amazon Simple Storage Service 경로나 데이터베이스 자격 증명이 저장되는 Amazon 리소스 이름 (ARN) 과 같은 종속 AWS 리소스에 AWS Secrets Manager 대한 정보도 들어 있습니다.
- `Application executable (binary)`: 이 바이너리는 사용자가 컴파일하여 메인프레임 현대화에 AWS 배포하려는 바이너리입니다.
- `Application JCL or scripts`: 이 소스 코드는 애플리케이션을 대신하여 배치 작업 또는 기타 처리를 관리합니다.
- `User application data`: 데이터 세트를 가져오면 AWS 메인프레임 현대화는 애플리케이션에서 액세스할 수 있도록 관계형 데이터베이스에 데이터 세트를 저장합니다.
- `Application source code`: Amazon AppStream 2.0을 통해 AWS 메인프레임 현대화는 코드를 작성하고 컴파일할 수 있는 개발 환경을 제공합니다.

AWS 메인프레임 현대화는 이 데이터를 기본적으로 에 저장합니다. AWS귀하로부터 수집한 데이터는 AWS 메인프레임 현대화가 관리하는 Amazon S3 버킷에 저장됩니다. 애플리케이션을 배포하면 AWS 메인프레임 현대화가 Amazon Elastic Block Store가 지원하는 Amazon Elastic Compute Cloud 인스턴스로 데이터를 다운로드합니다. 정리가 트리거되면 Amazon EBS 볼륨과 Amazon S3에서 데이터가 제거됩니다. Amazon EBS 볼륨은 싱글 테넌트이므로 고객 한 명당 하나의 인스턴스가 사용됩니다. 인스턴스는 절대 공유되지 않습니다. 런타임 환경을 삭제하면 Amazon EBS 볼륨도 삭제됩니다. 애플리케이션을 삭제하면 Amazon S3에서 아티팩트와 구성이 삭제됩니다.

애플리케이션 로그는 Amazon에 저장됩니다 CloudWatch. 고객 애플리케이션 로그 메시지도 CloudWatch 내보냅니다. CloudWatch 로그에는 고객의 민감한 데이터 (예: 디버그 메시지의 비즈니스 데이터 또는 보안 정보) 가 포함될 수 있습니다. 자세한 설명은 Amazon을 통한 AWS 메인프레임 현대화 모니터링 CloudWatch 섹션을 참조하세요.

또한 하나 이상의 Amazon Elastic File System 또는 Amazon FSx 파일 시스템을 런타임 환경에 연결하기로 선택하면 해당 시스템 내의 데이터가 AWS에 저장됩니다. 파일 시스템 사용을 중단하기로 결정한 경우 해당 데이터를 정리해야 합니다.

AWS 메인프레임 현대화가 애플리케이션 배포 및 데이터 세트 가져오기에 사용하는 Amazon S3 버킷에 데이터를 배치하면 사용 가능한 모든 Amazon S3 암호화 옵션을 사용하여 데이터를 보호할 수 있습니다. 또한 런타임 환경에 이러한 파일 시스템을 하나 이상 연결하는 경우 Amazon EFS 및 Amazon FSx 암호화 옵션을 사용할 수 있습니다.

메인프레임 현대화 서비스를 위한 유휴 데이터 암호화 AWS

AWS 메인프레임 현대화는 Amazon Simple Storage Service, Amazon DynamoDB, Amazon Elastic Block Store와 AWS Key Management Service 통합되어 데이터를 영구적으로 저장하는 모든 종속 리소스에 투명한 서버 측 암호화 (SSE) 를 제공합니다. AWS 메인프레임 현대화는 사용자를 위해 대칭 암호화 키를 생성하고 관리합니다. AWS KMS AWS KMS

기본적으로 저장된 데이터를 암호화하면 민감한 데이터를 보호하는 데 수반되는 운영 오버헤드와 복잡성을 줄이는 데 도움이 됩니다. 동시에 엄격한 암호화 규정 준수 및 규제 요구 사항이 필요한 애플리케이션을 마이그레이션할 수 있습니다.

런타임 환경 및 애플리케이션을 생성할 때는 이 암호화 계층을 비활성화하거나 다른 암호화 유형을 선택할 수 없습니다.

AWS 메인프레임 현대화 애플리케이션 및 런타임 환경을 위한 자체 고객 관리 키를 사용하여 Amazon S3 및 Amazon EBS 리소스를 암호화할 수 있습니다.

AWS 메인프레임 현대화 애플리케이션의 경우 이 키를 사용하여 애플리케이션 정의는 물론 서비스 계정에서 생성된 Amazon S3 버킷에 저장되는 JCL 파일과 같은 다른 애플리케이션 리소스를 암호화할 수 있습니다. 자세한 설명은 애플리케이션 생성 섹션을 참조하세요.

AWS 메인프레임 현대화 런타임 환경의 경우, AWS 메인프레임 현대화는 고객 관리 키를 사용하여 생성한 Amazon EBS 볼륨을 암호화하여 메인프레임 AWS 현대화 Amazon EC2 인스턴스 (서비스 계정에도 있음) 에 연결합니다. 자세한 설명은 런타임 환경 만들기 섹션을 참조하세요.

> **Note**
>
> DynamoDB 리소스는 메인프레임 현대화 서비스 내 AWS 관리형 키 AWS 계정을 사용하여 항상 암호화됩니다. 고객 관리형 키를 사용하여 DynamoDB 리소스를 암호화할 수 없습니다.

AWS 메인프레임 현대화는 다음 작업에 고객 관리 키를 사용합니다.

- 애플리케이션 재배치.
- AWS 메인프레임 현대화 Amazon EC2 인스턴스 교체.

AWS 메인프레임 현대화에서는 메인프레임 현대화 애플리케이션을 AWS 지원하기 위해 생성된 Amazon Relational Database Service 또는 Amazon Aurora 데이터베이스, Amazon Simple Queue

Service 대기열 및 ElastiCache Amazon 캐시를 암호화하는 데 고객 관리 키를 사용하지 않습니다. 이러한 대기열에는 고객 데이터가 포함되어 있지 않기 때문입니다.

자세한 내용은AWS Key Management Service 개발자 안내서의 고객 관리형 키 섹션을 참조하세요.

다음 표에는 메인프레임 현대화가 민감한 데이터를 암호화하는 방법이 요약되어 있습니다. AWS

데이터 유형	AWS 관리형 키 암호화	고객 관리형 키 암호화
`Definition` 특정 애플리케이션에 대한 애플리케이션 정의입니다.	활성화됨	활성화됨
`EnvironmentSummary` 런타임 환경에 대한 정보가 들어 있습니다.	활성화됨	활성화됨
`ApplicationSummary` AWS 메인프레임 현대화 애플리케이션에 대한 정보가 들어 있습니다.	활성화됨	활성화됨
`DeploymentSummary` AWS 메인프레임 현대화 응용 프로그램 배포에 대한 정보가 들어 있습니다.	활성화됨	활성화됨

> **Note**
>
> AWS 메인프레임 현대화를 통해 저장된 암호화를 통해 민감한 데이터를 보호하는 AWS 관리형 키 데 사용할 수 있는 추가 비용이 부과되지 않습니다. 하지만 고객 관리 키 사용에는 AWS KMS 요금이 부과됩니다. 요금에 대한 자세한 내용은 AWS Key Management Service 요금 부분을 참조하세요.

에 대한 자세한 내용은 AWS KMS을 참조하십시오 AWS Key Management Service.

AWS 메인프레임 현대화에서 지원금을 사용하는 방법 AWS KMS

AWS 메인프레임 현대화를 위해서는 고객 관리 키를 사용할 수 있는 보조금이 필요합니다.

애플리케이션 또는 런타임 환경을 생성하거나 고객 관리 키로 암호화된 AWS 메인프레임 현대화에 애플리케이션을 배포하는 경우, AWS 메인프레임 현대화는 사용자를 대신하여 요청을 전송하여 허가를 생성합니다. CreateGrant AWS KMS AWS KMS 부여는 AWS 메인프레임 현대화 팀이 고객 계정의 KMS 키에 액세스할 수 있도록 하는 데 사용됩니다.

AWS 메인프레임 현대화를 위해서는 고객 관리형 키를 다음과 같은 내부 운영에 사용할 수 있는 권한이 필요합니다.

- 애플리케이션, 런타임 환경 또는 애플리케이션 배포를 생성할 때 입력한 대칭 고객 관리 키 ID가 유효한지 AWS KMS 확인하도록 DescribeKey요청을 보내십시오.
- 메인프레임 현대화 런타임 환경을 AWS KMS AWS 호스팅하는 Amazon EC2 인스턴스에 연결된 Amazon EBS 볼륨을 암호화하라는 GenerateDataKey요청을 보냅니다.
- Amazon EBS의 암호화된 콘텐츠를 복호화하기 AWS KMS 위해 복호화 요청을 로 전송하십시오.

AWS 메인프레임 현대화는 런타임 환경을 만들고, 애플리케이션을 생성 또는 재배포하고, 배포를 생성할 때 Secrets Manager에 저장된 암호를 해독하기 위해 AWS KMS 권한 부여를 사용합니다. AWS 메인프레임 현대화가 제공하는 보조금은 다음과 같은 작업을 지원합니다.

- 런타임 환경 권한 부여 생성 또는 업데이트:
  - Decrypt
  - 암호화
  - ReEncryptFrom
  - ReEncryptTo
  - GenerateDataKey
  - DescribeKey
  - CreateGrant
- 애플리케이션 권한 부여 생성 또는 재배포:
  - GenerateDataKey
- 배포 그룹 만들기:

- Decrypt

언제든지 권한 부여에 대한 액세스 권한을 취소하거나 고객 관리형 키에 대한 서비스 액세스를 제거할 수 있습니다. 이렇게 하면 AWS 메인프레임 현대화는 고객 관리 키로 암호화된 데이터에 액세스할 수 없게 되며, 이는 데이터에 의존하는 운영에 영향을 미칩니다. 예를 들어 AWS 메인프레임 현대화에서 고객 관리 키로 암호화된 애플리케이션 정의에 해당 키를 부여하지 않고 액세스하려고 하면 애플리케이션 생성 작업이 실패합니다.

AWS 메인프레임 현대화는 사용자 애플리케이션 구성 (JSON 파일) 과 아티팩트 (바이너리 및 실행 파일) 를 수집합니다. 또한 AWS 메인프레임 현대화 운영에 사용되는 다양한 엔티티를 추적하는 메타데이터를 생성하고 로그와 지표를 생성합니다. 고객이 볼 수 있는 로그 및 지표는 다음과 같습니다.

- CloudWatch 애플리케이션 및 런타임 엔진 (Blu Age 또는 AWS Micro Focus) 을 반영하는 로그.
- CloudWatch 운영 대시보드의 지표.

또한 AWS 메인프레임 현대화는 서비스에 대한 측정, 활동 보고 등을 위한 사용 데이터와 지표를 수집합니다. 이 데이터는 고객이 볼 수 없습니다.

AWS 메인프레임 현대화는 데이터 유형에 따라 이 데이터를 다른 위치에 저장합니다. 업로드하는 고객 데이터는 Amazon S3 버킷에 저장됩니다. 서비스 데이터는 Amazon S3와 DynamoDB 모두에 저장됩니다. 애플리케이션을 배포하면 데이터와 서비스 데이터가 모두 Amazon EBS 볼륨에 다운로드됩니다. Amazon EFS 또는 Amazon FSx 스토리지를 런타임 환경에 연결하도록 선택하면 해당 파일 시스템에 저장된 데이터도 Amazon EBS 볼륨으로 다운로드됩니다.

저장 시 암호화는 기본적으로 구성됩니다. 비활성화하거나 변경할 수 없습니다. 현재는 해당 구성도 변경할 수 없습니다.

고객 관리형 키 생성

또는 API를 사용하여 대칭적인 고객 관리 키를 생성할 수 있습니다. AWS Management Console AWS KMS

대칭형 고객 관리형 키를 생성하려면

AWS Key Management Service 개발자 안내서의 대칭형 고객 관리형 키 생성 단계를 따르세요.

키 정책

키 정책은 고객 관리형 키에 대한 액세스를 제어합니다. 모든 고객 관리형 키에는 키를 사용할 수 있는 사람과 키를 사용하는 방법을 결정하는 문장이 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.

AWS 메인프레임 현대화 리소스와 함께 고객 관리 키를 사용하려면 키 정책에서 다음 API 작업을 허용해야 합니다.

- `kms:CreateGrant` – 고객 관리형 키에 권한 부여를 추가합니다. 지정된 KMS 키에 대한 제어 액세스 권한을 부여하여 메인프레임 현대화에 필요한 운영 AWS 권한을 부여할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 권한 부여 사용을 참조하세요.

  이를 통해 AWS 메인프레임 현대화는 다음을 수행할 수 있습니다.

  - 데이터 키가 암호화에 즉시 사용되지 않으므로 암호화된 데이터 키를 생성하여 저장하려면 `GenerateDataKey`를 직접적으로 호출합니다.
  - 저장된 암호화된 데이터 키를 사용하여 암호화된 데이터에 `Decrypt` 액세스하려면 직접적으로 호출하세요.
  - 서비스가 `RetireGrant`를 사용할 수 있도록 은퇴하는 보안 주체를 설정하세요.
- `kms:DescribeKey`— AWS 메인프레임 현대화가 키를 검증할 수 있도록 고객 관리 주요 세부 정보를 제공합니다.

AWS 메인프레임 현대화에는 고객의 주요 정책에 `kms:DescribeKey` 대한 권한 `kms:CreateGrant` 및 권한이 필요합니다. AWS 메인프레임 현대화는 이 정책을 사용하여 자체적으로 지원금을 마련합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "Enable IAM User Permissions",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::AccountId:role/ExampleRole"
        },
        "Action": [
            "kms:CreateGrant",
            "kms:DescribeKey"
        ],
        "Resource": "*"
    }]
}
```

> **Note**
>
> 위 예제에 표시된 역할은 및 `Principal` 와 같은 AWS 메인프레임 현대화 작업에 사용하는 역할입니다. `CreateApplication` `CreateEnvironment`

정책에서 사용 권한을 지정하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 참조하세요.

키 액세스 문제 해결에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 참조하세요.

AWS 메인프레임 현대화를 위한 고객 관리 키 지정

다음 리소스에 대해 고객 관리 키를 지정할 수 있습니다.

- 애플리케이션
- 환경

리소스를 생성할 때 KMS ID를 입력하여 키를 지정할 수 있습니다. KMS ID는 AWS 메인프레임 현대화에서 리소스에 저장된 민감한 데이터를 암호화하는 데 사용합니다.

- KMS ID— 고객 관리형 키의 키 식별자입니다. 키 ID, 키 ARN, 별칭 이름 또는 별칭 ARN을 입력합니다.

또는 를 사용하여 고객 관리 키를 지정할 수 있습니다. AWS Management Console AWS CLI

에서 런타임 환경을 만들 때 고객 관리 키를 AWS Management Console지정하려면 을 참조하십시오AWS 메인프레임 현대화 런타임 환경을 만드세요. 에서 애플리케이션을 생성할 때 고객 관리 키를 AWS Management Console지정하려면 을 참조하십시오AWS 메인프레임 현대화 애플리케이션 만들기.

를 사용하여 런타임 환경을 만들 때 고객 관리 키를 추가하려면 다음과 같이 `kms-key-id` 매개변수를 지정하십시오. AWS CLI

```
aws m2 create-environment –engine-type microfocus –instance-type M2.m5.large
--publicly-accessible –engine-version 7.0.3 –name test
--high-availability-config desiredCapacity=2
--kms-key-id myEnvironmentKey
```

를 사용하여 애플리케이션을 생성할 때 고객 관리 키를 추가하려면 다음과 같이 kms-key-id 매개변수를 지정하십시오. AWS CLI

```
aws m2 create-application –name test-application –description my description
--engine-type microfocus
--definition content="$(jq -c . raw-template.json | jq -R)"
--kms-key-id myApplicationKey
```

AWS 메인프레임 현대화 암호화 컨텍스트

암호화 컨텍스트는 데이터에 대한 추가 컨텍스트 정보를 포함하는 선택적 키-값 쌍 세트입니다.

AWS KMS 암호화 컨텍스트를 인증된 추가 데이터로 사용하여 인증된 암호화를 지원합니다. 데이터 암호화 요청에 암호화 컨텍스트를 포함하면 암호화 컨텍스트를 암호화된 데이터에 AWS KMS 바인딩합니다. 요청에 동일한 암호화 컨텍스트를 포함해야 이 데이터를 해독할 수 있습니다.

AWS 메인프레임 현대화 암호화 컨텍스트

AWS 메인프레임 현대화는 애플리케이션과 관련된 모든 AWS KMS 암호화 작업 (애플리케이션 생성 및 배포 생성) 에서 동일한 암호화 컨텍스트를 사용합니다. 여기서 키는 애플리케이션이고 aws:m2:app 값은 애플리케이션의 고유 식별자입니다.

Example

```
"encryptionContextSubset": {
        "aws:m2:app": "a1bc2defabc3defabc4defabcd"
}
```

모니터링을 위한 암호화 컨텍스트 사용

대칭적인 고객 관리 키를 사용하여 애플리케이션 또는 런타임 환경을 암호화하는 경우 감사 레코드 및 로그의 암호화 컨텍스트를 사용하여 고객 관리 키가 사용되는 방식을 식별할 수도 있습니다.

암호화 컨텍스트를 사용하여 고객 관리형 키에 대한 액세스 제어

그러나 암호화 컨텍스트를 사용하여 키 정책 및 IAM 정책에서 대칭 conditions에 대한 액세스를 제어할 수도 있습니다. 또한 권한 부여에서 암호화 컨텍스트 제약 조건을 사용할 수 있습니다.

AWS 메인프레임 현대화는 권한 부여의 암호화 컨텍스트 제약을 사용하여 계정 또는 지역의 고객 관리 키에 대한 액세스를 제어합니다. 권한 부여 제약 조건에 따라 권한 부여가 허용하는 작업은 지정된 암호화 컨텍스트를 사용해야 합니다. 다음은 AWS 메인프레임 현대화가 애플리케이션을 생성할 때 애플리케이션 아티팩트를 암호화하는 데 활용하는 권한 부여입니다.

```
//This grant is retired immediately after create application finish
{
   "grantee-principal": m2.us-west-2.amazonaws.com,
   "retiring-principal": m2.us-west-2.amazonaws.com,
   "operations": [
       "GenerateDataKey"
   ]
   "condition": {
        "encryptionContextSubset": {
            “aws:m2:app”: “a1bc2defabc3defabc4defabcd”
   }
}
```

AWS 메인프레임 현대화를 위한 암호화 키 모니터링

메인프레임 현대화 리소스와 함께 AWS KMS 고객 관리 키를 사용하는 경우 AWS CloudTrailAmazon CloudWatch Logs를 사용하여 AWS 메인프레임 현대화가 보내는 요청을 추적할 수 있습니다. AWS AWS KMS

런타임 환경의 예

다음은 AWS 메인프레임 현대화에서 고객 관리 키로 암호화된 `GenerateDataKey` 데이터에 `Decrypt` 액세스하기 위해 호출하는 `DescribeKeyCreateGrant`,, KMS 운영 모니터링 AWS CloudTrail 이벤트의 예입니다.

DescribeKey

AWS 메인프레임 현대화는 `DescribeKey` 작업을 통해 런타임 환경과 관련된 AWS KMS 고객 관리 키가 계정 및 지역에 존재하는지 확인합니다.

다음 예제 이벤트는 `DescribeKey` 작업을 기록합니다.

```
{
    "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:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T19:40:26Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-12-06T20:23:43Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.182",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "keyId": "00dd0db0-0000-0000-ac00-b0c000SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
```

```
}
```

CreateGrant

AWS KMS 고객 관리 키를 사용하여 런타임 환경을 암호화하면 AWS 메인프레임 현대화는 필요한 KMS 작업을 수행하도록 사용자를 대신하여 몇 가지 `CreateGrant` 요청을 보냅니다. AWS 메인프레임 현대화로 창출되는 일부 지원금은 사용 후 즉시 사용 중지됩니다. 런타임 환경을 삭제하면 사용 중지되는 것도 있습니다.

다음 예제 이벤트는 환경 생성 워크플로와 관련된 Lambda 실행 역할에 대한 `CreateGrant` 작업을 기록합니다.

```
{
    "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:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:11:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:09Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
```

```
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "operations": [
            "Encrypt",
            "Decrypt",
            "ReEncryptFrom",
            "ReEncryptTo",
            "GenerateDataKey",
            "GenerateDataKey",
            "DescribeKey",
            "CreateGrant"
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


다음 예제 이벤트는 Auto Scaling 그룹 서비스 연결 역할의 `CreateGrant` 작업을 기록합니다. 환경 생성 워크플로와 관련된 Lambda 실행 역할이 이 `CreateGrant` 작업을 직접적으로 호출합니다. 실행 역할에 Auto Scaling 그룹의 서비스 연결 역할에 대해 서브그랜트를 생성할 수 있는 권한을 부여합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65MZFUPM4JO:EnvironmentWorkflow-alpha-
CreateEnvironmentLambda7-HfxDj5zz86tr",
        "arn": "arn:aws:sts::111122223333:assumed-role/EnvironmentWorkflow-
alpha-CreateEnvironmentLambdaS-1AU4A8VNQEEKN/EnvironmentWorkflow-alpha-
CreateEnvironmentLambda7-HfxDj5zz86tr",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/EnvironmentWorkflow-alpha-
CreateEnvironmentLambdaS-1AU4A8VNQEEKN",
                "accountId": "111122223333",
                "userName": "EnvironmentWorkflow-alpha-
CreateEnvironmentLambdaS-1AU4A8VNQEEKN"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:22:28Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-12-06T20:23:09Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "54.148.236.160",
    "userAgent": "aws-sdk-java/2.18.21 Linux/4.14.255-276-224.499.amzn2.x86_64
 OpenJDK_64-Bit_Server_VM/11.0.14.1+10-LTS Java/11.0.14.1 vendor/Amazon.com_Inc. md/
internal exec-env/AWS_Lambda_java11 io/sync http/Apache cfg/retry-mode/legacy",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "operations": [
            "Encrypt",
            "Decrypt",
            "ReEncryptFrom",
```

```
            "ReEncryptTo",
            "GenerateDataKey",
            "GenerateDataKey",
            "DescribeKey",
            "CreateGrant"
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
}
```

GenerateDataKey

런타임 환경 리소스에 대한 AWS KMS 고객 관리 키를 활성화하면 Auto Scaling은 런타임 환경과 연결된 Amazon EBS 볼륨을 암호화하기 위한 고유한 키를 생성합니다. 리소스의 AWS KMS 고객 관리 키를 AWS KMS 지정하는 `GenerateDataKey` 요청을 보냅니다.

다음 예제 이벤트는 GenerateDataKey 작업을 기록합니다.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65EEXVIEH7D:AutoScaling",
        "arn": "arn:aws:sts::111122223333:assumed-role/AWSServiceRoleForAutoScaling/
AutoScaling",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling",
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAutoScaling"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:23:16Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "autoscaling.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:18Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "autoscaling.amazonaws.com",
    "userAgent": "autoscaling.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:ebs:id": "vol-080f7a32d290807f3"
        },
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "numberOfBytes": 64
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
```

```
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Decrypt

암호화된 런타임 환경에 액세스하는 경우 Amazon EBS는 저장된 암호화된 데이터 키를 사용하여 암호화된 데이터에 액세스하는 `Decrypt` 작업을 직접적으로 호출합니다.

다음 예제 이벤트는 `Decrypt` 작업을 기록합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ebs.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:22Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ebs.amazonaws.com",
    "userAgent": "ebs.amazonaws.com",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "encryptionContext": {
            "aws:ebs:id": "vol-080f7a32d290807f3"
        }
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
```

```
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "sharedEventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventCategory": "Management"
}
```

애플리케이션 예제

다음은 AWS 메인프레임 현대화에서 고객 관리 키로 암호화된 데이터에 GenerateDataKey 액세스하기 위해 호출한 KMS 운영 CreateGrant 및 모니터링 AWS CloudTrail 이벤트의 예입니다.

CreateGrant

AWS KMS 고객 관리형 키를 사용하여 애플리케이션 리소스를 암호화하면 Lambda 실행 역할이 사용자 대신 계정의 KMS 키에 액세스하라는 요청을 보냅니다CreateGrant. AWS 권한을 부여하면 Lambda 실행 역할이 고객 관리 키를 사용하여 고객 애플리케이션 리소스를 Amazon S3에 업로드할 수 있습니다. 이 권한 부여는 애플리케이션이 생성된 후 즉시 사용 중지됩니다.

다음 예제 이벤트는 CreateGrant 작업을 기록합니다.

```
{
    "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:Sampleuser01",
```

```
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T21:51:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T22:47:04Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "constraints": {
            "encryptionContextSubset": {
                "aws:m2:app": "a1bc2defabc3defabc4defabcd"
            }
        },
        "retiringPrincipal": "m2.us-west-2.amazonaws.com",
        "operations": [
            "GenerateDataKey"
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
```

```
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

GenerateDataKey

애플리케이션 리소스에 대한 AWS KMS 고객 관리 키를 활성화하면 Lambda 실행 역할이 고객 데이터를 암호화하고 Amazon Simple Storage Service에 업로드하는 데 사용하는 키를 생성합니다. Lambda 실행 역할은 AWS KMS 리소스의 고객 관리 키를 AWS KMS 지정하는 요청을 보냅니다`GenerateDataKey`.

다음 예제 이벤트는 `GenerateDataKey` 작업을 기록합니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65CLCEKKC7Z:ApplicationWorkflow-alpha-
CreateApplicationVersion-CstWZUn5R4u6",
        "arn": "arn:aws:sts::111122223333:assumed-role/ApplicationWorkflow-
alpha-CreateApplicationVersion-1IZRBZYDG20B/ApplicationWorkflow-alpha-
CreateApplicationVersion-CstWZUn5R4u6",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/ApplicationWorkflow-alpha-
CreateApplicationVersion-1IZRBZYDG20B",
                "accountId": "111122223333",
                "userName": "ApplicationWorkflow-alpha-
CreateApplicationVersion-1IZRBZYDG20B"
            },
            "webIdFederationData": {},
            "attributes": {
```

```
                "creationDate": "2022-12-06T23:28:32Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:29:08Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:m2:app": "a1bc2defabc3defabc4defabcd",
            "aws:s3:arn": "arn:aws:s3:::supernova-processedtemplate-111122223333-us-
west-2/111122223333/a1bc2defabc3defabc4defabcd/1/cics-transaction/ZBNKE35.so"
        },
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

배포 예시

다음은 AWS 메인프레임 현대화에서 고객 관리 키로 암호화된 데이터에 Decrypt 액세스하기 위해 KMS 작업을 위한 AWS CloudTrail 이벤트 CreateGrant 및 모니터링을 위한 이벤트입니다.

CreateGrant

AWS KMS 고객 관리 키를 사용하여 배포 리소스를 암호화하면 AWS 메인프레임 현대화가 사용자를 대신하여 두 개의 요청을 보냅니다. CreateGrant 첫 번째 권한 부여는 ListBatchJobScriptFiles 호출할 현재 Lambda 실행 역할에 대한 것으로, 배포가 완료되면 즉시 사용 중지됩니다. 두 번째 허가는 Amazon EC2가 Amazon S3에서 고객 애플리케이션 리소스를 다운로드할 수 있도록 범위를 좁힌 Amazon EC2 인스턴스 역할에 대한 것입니다. 이 부여는 애플리케이션이 런타임 환경에서 삭제되면 사용 중지됩니다.

다음 예제 이벤트는 CreateGrant 작업을 기록합니다.


```
{
    "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:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T21:51:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:40:07Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
```

```
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:m2:app": "a1bc2defabc3defabc4defabcd"
            }
        },
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


Decrypt

배포에 액세스하면 Amazon EC2가 `Decrypt` 작업을 직접적으로 호출하여 저장된 암호화된 데이터 키를 사용하여 Amazon S3에서 암호화된 고객 데이터를 복호화하고 다운로드합니다.

다음 예제 이벤트는 Decrypt 작업을 기록합니다.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65BSPZ37E6G:m2-hm-bqe367dxtfcpdbzmnhfzranisu",
        "arn": "arn:aws:sts::111122223333:assumed-role/
SupernovaEnvironmentInstanceScopeDownRole/m2-hm-bqe367dxtfcpdbzmnhfzranisu",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/
SupernovaEnvironmentInstanceScopeDownRole",
                "accountId": "111122223333",
                "userName": "SupernovaEnvironmentInstanceScopeDownRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T23:19:29Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:40:15Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:m2:app": "a1bc2defabc3defabc4defabcdm",
            "aws:s3:arn": "arn:aws:s3:::supernova-processedtemplate-111122223333-us-
west-2/111122223333/a1bc2defabc3defabc4defabcdm/1/cics-transaction/BBANK40P.so"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
```

```
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

자세히 알아보기

다음 리소스에서 키에 대한 추가 정보를 확인할 수 있습니다.

- AWS Key Management Service 기본 개념에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 참조하세요.
- AWS Key Management Service의 보안 모범 사례 에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 참조하세요.

전송 중 암호화

트랜잭션 워크로드의 일부인 대화형 애플리케이션의 경우 TN3270 프로토콜용 터미널 에뮬레이터와 AWS 메인프레임 현대화 서비스 엔드포인트 간의 데이터 교환은 전송 중에 암호화되지 않습니다. 애플리케이션에 전송 중 암호화가 필요한 경우 몇 가지 추가 터널링 메커니즘을 구현하는 것이 좋습니다.

AWS 메인프레임 현대화는 HTTPS를 사용하여 서비스 API를 암호화합니다. AWS 메인프레임 현대화 내의 다른 모든 통신은 HTTPS뿐만 아니라 서비스 VPC 또는 보안 그룹에 의해 보호됩니다. AWS 메인프레임 현대화는 애플리케이션 아티팩트, 구성 및 애플리케이션 데이터를 전송합니다. 애플리케이션 객체는 애플리케이션 데이터와 마찬가지로 사용자가 소유한 Amazon S3 버킷에서 복사됩니다. Amazon S3에 대한 링크를 사용하거나 파일을 로컬로 업로드하여 애플리케이션 구성을 제공할 수 있습니다.

전송 중 기본 암호화는 기본적으로 구성되지만 TN3270 프로토콜에는 적용되지 않습니다. AWS 메인프레임 현대화는 API 엔드포인트에 HTTPS를 사용하며, 이 역시 기본적으로 구성됩니다.

AWS 메인프레임 현대화를 위한 Identity 및 Access Management

AWS Identity and Access Management (IAM) 은 관리자가 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와줍니다. AWS IAM 관리자는 AWS 메인프레임 현대화 리소스를 사용할 수 있는 인증 (로그인) 및 권한 부여 (권한 보유) 를 받을 수 있는 사용자를 제어합니다. IAM은 추가 비용 없이 사용할 AWS 서비스 수 있습니다.

주제

고객

메인프레임 현대화에서 AWS 수행하는 작업에 따라 AWS Identity and Access Management (IAM) 사용 방식이 다릅니다.

서비스 사용자 — AWS 메인프레임 현대화 서비스를 사용하여 업무를 수행하는 경우 관리자가 필요한 자격 증명과 권한을 제공합니다. 더 많은 AWS 메인프레임 현대화 기능을 사용하여 작업을 수행함에 따라 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. AWS 메인프레임 현대화의 기능에 액세스할 수 없는 경우 AWS 메인프레임 현대화 ID 및 액세스 문제 해결 단원을 참조하세요.

서비스 관리자 — 회사에서 메인프레임 현대화 리소스를 담당하는 경우 AWS 메인프레임 현대화에 대한 전체 액세스 권한을 가지고 있을 것입니다. AWS 서비스 사용자가 액세스해야 하는 AWS 메인프레임 현대화 기능과 리소스를 결정하는 것은 여러분의 몫입니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사에서 IAM을 AWS 메인프레임 현대화와 함께 사용하는 방법에 대한 자세한 내용은 을 참조하십시오. AWS 메인프레임 현대화가 IAM과 함께 작동하는 방식

IAM 관리자 — IAM 관리자라면 메인프레임 현대화에 대한 액세스를 관리하기 위한 정책을 작성하는 방법에 대해 자세히 알고 싶을 것입니다. AWS IAM에서 사용할 수 있는 AWS 메인프레임 현대화 ID 기반 정책의 예를 보려면 을 참조하십시오. 메인프레임 현대화를 위한 ID 기반 정책 예제 AWS

자격 증명을 통한 인증

인증은 ID 자격 증명을 사용하여 로그인하는 방법입니다. AWS IAM 사용자로 인증 (로그인 AWS) 하거나 IAM 역할을 맡아 인증 (로그인) 해야 합니다. AWS 계정 루트 사용자

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 로그인할 수 있습니다. AWS AWS IAM Identity Center (IAM ID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예입니다. 연동 자격 증명으로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 액세스하는 경우 AWS 간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 내 로그인 방법을 참조하십시오. AWS AWS 계정

AWS 프로그래밍 방식으로 액세스하는 경우 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트 (SDK) 와 명령줄 인터페이스 (CLI) 를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청 서명을 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA) 을 사용할 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 IAM 사용자 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하세요.

AWS 계정 루트 사용자

계정을 AWS 계정만들 때는 먼저 계정의 모든 AWS 서비스 리소스에 대한 완전한 액세스 권한을 가진 하나의 로그인 ID로 시작합니다. 이 ID를 AWS 계정 루트 사용자라고 하며, 계정을 만들 때 사용한 이메일 주소와 비밀번호로 로그인하여 액세스할 수 있습니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 태스크의 전체 목록은 IAM 사용자 설명서의 루트 사용자 보안 인증이 필요한 태스크를 참조하세요.

연동 보안 인증

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 비롯한 수동 AWS 서비스 사용자가 ID 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 액세스하도록 하는 것입니다.

페더레이션 ID는 기업 사용자 디렉토리, 웹 ID 공급자, Identity Center 디렉터리의 사용자 또는 ID 소스를 통해 제공된 자격 증명을 사용하여 액세스하는 AWS 서비스 모든 사용자를 말합니다. AWS Directory Service페더레이션 ID에 AWS 계정액세스하면 이들이 역할을 맡고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 자체 ID 소스의 사용자 및 그룹 집합에 연결하고 동기화하여 모든 사용자 및 애플리케이션에서 사용할 수 있습니다. AWS 계정 IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇입니까?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한을 AWS 계정 가진 사용자 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용자 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 보안 인증입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증 정보를 가지고 있지만, 역할은 임시 보안 인증 정보만 제공합니다. 자세한 정보는 IAM 사용자 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가진 사용자 AWS 계정 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 AWS Management Console 전환하여 에서 일시적으로 IAM 역할을 맡을 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용자 설명서의 IAM 역할 사용을 참조하세요.

임시 보안 인증 정보가 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 연동 자격 증명에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 연동 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용자 설명서의 Creating a role for a third-party Identity Provider(서드 파티 자격 증명 공급자의 역할 만들기) 부분을 참조하세요. IAM 자격 증명 센터를 사용하는 경우 권한 집합을 구성합니다. 인증 후 아이덴티티가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관 짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트 섹션을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스경우에는 역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 교차 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용자 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.
- 서비스 간 액세스 — 일부는 다른 AWS 서비스서비스의 기능을 AWS 서비스 사용합니다. 예를 들어 서비스에서 직접 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 태스크를 수행할 수 있습니다.
  - 순방향 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 경우 보안 AWS 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 AWS 서비스서비스에 AWS 서비스 요청하기 위한 요청과 결합하여 사용합니다. FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용자 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.
  - 서비스 연결 역할 — 서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

- Amazon EC2에서 실행되는 애플리케이션 — IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 API 요청을 AWS CLI 하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. AWS 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증 정보를 얻을 수 있습니다. 자세한 정보는 IAM 사용자 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용자 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하세요.

정책을 사용한 액세스 관리

정책을 생성하고 이를 AWS ID 또는 리소스에 AWS 연결하여 액세스를 제어할 수 있습니다. 정책은 ID 또는 리소스와 연결될 때 AWS 해당 권한을 정의하는 객체입니다. AWS 주도자 (사용자, 루트 사용자 또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책의 권한이 요청 허용 또는 거부 여부를 결정합니다. 대부분의 정책은 JSON 문서로 AWS 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용자 설명서의 JSON 정책 개요를 참조하세요.

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.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole` 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI, 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

보안 인증 기반 정책

보안 인증 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 보안 인증에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용자 설명서의 IAM 정책 생성을 참조하세요.

보안 인증 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 내 여러 사용자, 그룹 및 역할에 연결할

수 있는 독립형 정책입니다. AWS 계정관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용자 설명서의 관리형 정책과 인라인 정책의 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 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 태스크를 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. IAM의 AWS 관리형 정책은 리소스 기반 정책에 사용할 수 없습니다.

액세스 제어 목록(ACL)

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 설명서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

ACL을 지원하는 서비스의 예로는 아마존 S3와 아마존 VPC가 있습니다. AWS WAF ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 ACL(액세스 제어 목록) 개요 섹션을 참조하세요.

기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 타입에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 자격 증명 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 특성입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 개체의 보안 인증 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용자 설명서의 IAM 엔터티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책 (SCP) - SCP는 조직 또는 조직 단위 (OU) 에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations AWS Organizations 사업체가 소유한 여러 AWS 계정 개를 그룹화하고 중앙에서 관리하는 서비스입니다. 조직에서 모든 특성을 활성화할 경우 서비스 제어 정책

(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 구성원 계정의 엔티티 (각 엔티티 포함) 에 대한 권한을 제한합니다. AWS 계정 루트 사용자조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.

- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 자격 증명 기반 정책의 교집합과 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용자 설명서의 세션 정책을 참조하세요.

여러 정책 타입

여러 정책 타입이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련되어 있을 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

AWS 메인프레임 현대화가 IAM과 함께 작동하는 방식

IAM을 사용하여 메인프레임 현대화에 대한 액세스를 관리하기 전에 AWS 메인프레임 현대화에 사용할 수 있는 IAM 기능에 대해 알아보십시오. AWS

메인프레임 현대화와 함께 사용할 수 있는 IAM 기능 AWS

IAM 특성	AWS 메인프레임 현대화 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	예
ACL	아니요
ABAC(정책의 태그)	예
임시 보안 인증	예

IAM 특성	AWS 메인프레임 현대화 지원
전달 액세스 세션(FAS)	예
Service roles(서비스 역할)	예
서비스 링크 역할	예

AWS 메인프레임 현대화 및 기타 AWS 서비스가 대부분의 IAM 기능과 어떻게 작동하는지 자세히 알아보려면 IAM 사용 설명서의 IAM과 연동되는AWS 서비스를 참조하십시오.

메인프레임 현대화를 위한 ID 기반 정책 AWS

ID 기반 정책 지원	예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용자 설명서의 IAM 정책 생성(Creating IAM policies)을 참조합니다.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 태스크와 리소스 뿐만 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 자격 증명 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용자 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

메인프레임 현대화를 위한 ID 기반 정책 사례 AWS

AWS 메인프레임 현대화 ID 기반 정책의 예를 보려면 을 참조하십시오. 메인프레임 현대화를 위한 ID 기반 정책 예제 AWS

메인프레임 현대화 내의 리소스 기반 정책 AWS

리소스 기반 정책 지원	아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이

러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 등이 포함될 수 있습니다. AWS 서비스

크로스 계정 액세스를 활성화하려는 경우 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념합니다. 보안 주체와 리소스가 다른 AWS 계정경우 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 개체 (사용자 또는 역할) 에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 개체에 자격 증명 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 정보는 IAM 사용자 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

AWS 메인프레임 현대화를 위한 정책 조치

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. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

AWS 메인프레임 현대화 조치 목록을 보려면 서비스 권한 부여 참조의 AWS 메인프레임 현대화에 의해 정의된 작업을 참조하십시오.

AWS 메인프레임 현대화의 정책 조치는 조치 앞에 다음 접두사를 사용합니다.

```
m2
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "m2:StartApplication",
```

```
        "m2:StopApplication"
           ]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, List(이)라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "m2:List*"
```

AWS 메인프레임 현대화 ID 기반 정책의 예를 보려면 을 참조하십시오. 메인프레임 현대화를 위한 ID 기반 정책 예제 AWS

메인프레임 현대화를 위한 정책 리소스 AWS

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. 보고서에는 Resource 또는 NotResource 요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

ARN을 사용하여 IAM 정책이 적용되는 리소스를 식별하여 특정 AWS 메인프레임 현대화 리소스에 대한 액세스를 제한할 수 있습니다. ARN 형식에 대한 자세한 내용은 AWS 일반 참조의 Amazon 리소스 이름(ARN)을 참조하세요.

예를 들어, AWS 메인프레임 현대화 환경에는 다음과 같은 ARN이 있습니다.

```
"Resource": "arn:aws:m2:regionId:accountId:env/service-generated-unique-identifier"
```

AWS 메인프레임 현대화 애플리케이션에는 다음과 같은 ARN이 있습니다.

```
"Resource": "arn:aws:m2:regionId:accountId:app/service-generated-unique-identifier"
```

모든 AWS 메인프레임 현대화 작업이 리소스 수준 권한을 지원하는 것은 아닙니다. 리소스 수준 권한을 지원하지 않는 작업의 경우 와일드카드(*)를 사용해야 합니다.

다음 AWS 메인프레임 현대화 작업은 리소스 수준 권한을 지원하지 않습니다.

```
ListApplications
            ListApplicationVersions
            ListBatchJobDefinitions
            ListBatchJobExecutions
            ListDataSetImportHistory
            ListDataSets
            ListDeployments
            ListEngineVersions
            ListEnvironments
            ListTagsForResource
```

AWS 메인프레임 현대화 리소스 유형 및 해당 ARN 목록을 보려면 서비스 권한 부여 참조의 메인프레임 현대화에 의해 정의된 리소스를 참조하십시오. AWS 각 리소스의 ARN을 지정할 수 있는 작업에 대해 알아보려면 AWS 메인프레임 현대화로 정의된 작업을 참조하십시오.

AWS 메인프레임 현대화 ID 기반 정책의 예를 보려면 을 참조하십시오. 메인프레임 현대화를 위한 ID 기반 정책 예제 AWS

AWS 메인프레임 현대화 API 권한: 조치, 리소스, 조건 참조

아래에 있는 표를 참조하여 IAM 자격 증명에 연결할 수 있는 권한 정책(자격 증명 기반 정책)을 작성할 수 있습니다. 표에는 다음이 포함됩니다.

- 각 AWS 메인프레임 현대화 API 작업
- 각 작업을 수행할 수 있는 권한을 부여할 수 있는 작업
- 권한을 부여할 수 있는 AWS 리소스

정책의 `Action` 필드에서 작업을 지정하고 정책의 `Resource` 필드에서 리소스 값을 지정합니다.

AWS 메인프레임 현대화 정책의 AWS 글로벌 조건 키를 사용하여 조건을 표현할 수 있습니다. 전체 AWS 키 목록은 IAM 사용 설명서의 사용 가능한 글로벌 조건 키를 참조하십시오.

> **Note**
>
> 작업을 지정하려면 `m2:` 접두사 다음에 API 작업 명칭을 사용합니다(예: `m2:CreateApplication`).

AWS 메인프레임 현대화 API 및 작업에 필요한 권한

AWS 메인프레임 현대화 API 운영	필수 권한 (API 작업)	리소스
CancelBatchJobExecution		애플리케이션
CreateApplication	`s3:GetObject` `s3:ListBucket`	애플리케이션
CreateDataSetImportTask	`m2:CreateDataSetImportTask` `s3:GetObject`	애플리케이션

CreateDeployment	`elasticloadbalancing:AddTags`
	`elasticloadbalancing:CreateListener`
	`elasticloadbalancing:CreateTargetGroup`
	`elasticloadbalancing:RegisterTargets`
		애플리케이션

CreateEnvironment	`ec2:CreateNetworkInterface`
	`ec2:CreateNetworkInterfacePermission`
	`ec2:DescribeNetworkInterfaces`
		환경

AWS 메인프레임 현대화 API 운영	필수 권한 (API 작업)	리소스
	ec2:DescribeSecurityGroups ec2:DescribeSubnets ec2:DescribeVpcAttribute ec2:DescribeVpcs ec2:ModifyNetworkInterfaceAttribute elasticfilesystem:DescribeMountTargets elasticloadbalancing:AddTags elasticloadbalancing:CreateLoadBalancer fsx:DescribeFileSystems iam:CreateServiceLinkedRole	
DeleteApplication	elasticloadbalancing:DeleteListener elasticloadbalancing:DeleteTargetGroup logs:DeleteLogDelivery	애플리케이션
DeleteApplicationFromEnvironment	elasticloadbalancing:DeleteListener elasticloadbalancing:DeleteTargetGroup	애플리케이션 환경

AWS 메인프레임 현대화 API 운영	필수 권한 (API 작업)	리소스
DeleteEnvironment	elasticloadbalancing:DeleteLoadBalancer	환경
GetApplication		애플리케이션
GetApplicationVersion		애플리케이션
GetBatchJobExecution		애플리케이션
GetDataSetDetails		애플리케이션
GetDataSetImportTask		애플리케이션
GetDeployment		애플리케이션
GetEnvironment		환경
ListApplications		*
ListApplicationVersions		*
ListBatchJobDefinitions		*
ListBatchJobExecutions		*
ListDataSetImportHistory		*
ListDataSets		*
ListDeployments		*
ListEngineVersions		*

AWS 메인프레임 현대화 API 운영	필수 권한 (API 작업)	리소스
ListEnvironments		*
ListTagsForResource		*
StartApplication		애플리케이션
StartBatchJob		애플리케이션
StopApplication		애플리케이션
TagResource		*
UntagResource		*
UpdateApplication	s3:GetObject s3:ListBucket	애플리케이션
UpdateEnvironment		환경

메인프레임 현대화를 위한 정책 조건 키 AWS

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` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 적음 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우 AWS (은)는 논리적 `AND` 작업을 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 는 논리적 `OR` 연산을 사용하여 조건을 AWS 평가합니다. 명령문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 정보는 IAM 사용자 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM 사용 AWS 설명서의 글로벌 조건 컨텍스트 키를 참조하십시오.

다음 조건 키는 AWS 메인프레임 현대화에만 해당됩니다.

```
m2:EngineType
            m2:InstanceType
```

AWS 메인프레임 현대화 조건 키 목록을 보려면 서비스 권한 부여 참조의 메인프레임 현대화를 위한 AWS 조건 키를 참조하십시오. 조건 키를 사용할 수 있는 작업 및 리소스에 대해 알아보려면 메인프레임 현대화로 정의된 작업을 참조하십시오. AWS

AWS 메인프레임 현대화 ID 기반 정책의 예를 보려면 을 참조하십시오. 메인프레임 현대화를 위한 ID 기반 정책 예제 AWS

AWS 메인프레임 현대화의 액세스 제어 목록(ACL)

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 설명서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

메인프레임 현대화를 통한 속성 기반 액세스 제어 (ABAC) AWS

ABAC 지원(정책의 태그)	예

ABAC(속성 기반 액세스 제어)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. 에서는 이러한 속성을 AWS태그라고 합니다. IAM 엔티티 (사용자 또는 역할) 및 여러 AWS 리소스에 태그를 첨부할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그를 기반으로 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 예(Yes)입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우 값은 부분적(Partial)입니다.

ABAC에 대한 자세한 정보는 IAM 사용자 설명서의 ABAC란 무엇인가요?를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용자 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

AWS 메인프레임 현대화와 함께 임시 자격 증명 사용

임시 보안 인증 정보 지원	예

임시 자격 증명을 사용하여 로그인하면 작동하지 AWS 서비스 않는 것도 있습니다. 임시 자격 증명을 사용하는 방법을 AWS 서비스 비롯한 추가 정보는 IAM 사용 설명서의 IAM과AWS 서비스 연동되는 내용을 참조하십시오.

사용자 이름과 암호를 제외한 다른 방법을 AWS Management Console 사용하여 로그인하면 임시 자격 증명을 사용하는 것입니다. 예를 들어 회사의 SSO (Single Sign-On) 링크를 AWS 사용하여 액세스하는 경우 이 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증 정보를 자동으로 생성합니다. 역할 전환에 대한 자세한 정보는 IAM 사용자 설명서의 역할로 전환(콘솔)을 참조하세요.

또는 API를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다 AWS CLI . AWS 그런 다음 해당 임시 자격 증명을 사용하여 액세스할 수 AWS있습니다. AWS 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성할 것을 권장합니다. 자세한 정보는 IAM의 임시 보안 인증 정보 섹션을 참조하세요.

AWS 메인프레임 현대화를 위한 포워드 액세스 세션

전달 액세스 세션(FAS) 지원	예

IAM 사용자 또는 역할을 사용하여 작업을 수행하는 AWS경우 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 서비스에 AWS 서비스 요청하기 위한 요청과 함께 사용합니다. AWS 서비

스 FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

> **Important**
>
> 이러한 토큰을 통해 AWS 메인프레임 현대화는 명시적인 동의 없이 고객 데이터에 액세스할 수 있습니다. 예를 들어, AWS 메인프레임 현대화는 고객의 명시적인 허가 없이 Amazon S3 버킷의 관련 비즈니스 데이터와 함께 애플리케이션 아티팩트를 배포합니다. 이에 따라 규정 준수 문서를 업데이트해야 할 수 있습니다.

AWS 메인프레임 현대화를 위한 서비스 역할

서비스 역할 지원	예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM role(IAM 역할)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용자 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

AWS 메인프레임 현대화는 활동 후크 (트랜잭션/작업 결석 또는 완료 등) 에 대한 서비스 역할을 지원합니다.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 메인프레임 현대화 기능이 AWS 손상될 수 있습니다. AWS 메인프레임 현대화가 이에 대한 지침을 제공하는 경우에만 서비스 역할을 편집하십시오.

메인프레임 현대화에서 IAM 역할 선택 AWS

Amazon EC2에서 실행되는 애플리케이션이 위임할 수 있는 IAM 역할을 이전에 생성한 경우, 시작 템플릿 또는 시작 구성을 생성할 때 이 역할을 선택할 수 있습니다. AWS 메인프레임 현대화에서는 선택할 수 있는 역할 목록을 제공합니다. 이러한 역할을 생성할 때 애플리케이션에 필요한 특정 API 호출에 대한 액세스를 제한하는 최소 권한 IAM 정책을 연결하는 것이 중요합니다. 자세한 내용을 알아보려면 Amazon EC2 Auto Scaling 사용 설명서의 Amazon EC2 인스턴스에서 실행되는 애플리케이션의 IAM 역할을 참조하세요.

메인프레임 현대화를 위한 서비스 연계 역할 AWS

서비스 링크 역할 지원	예

서비스 연결 역할은 다음과 연결된 서비스 역할 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

AWS 메인프레임 현대화 서비스 연결 역할을 생성하거나 관리하는 방법에 대한 자세한 내용은 을 참조하십시오. Mainframe Modernization을 위한 서비스 연결 역할 사용

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스 섹션을 참조하세요. Service-linked role(서비스 연결 역할) 열에서 Yes이(가) 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

메인프레임 현대화를 위한 ID 기반 정책 예제 AWS

기본적으로 사용자 및 역할에는 메인프레임 현대화 리소스를 만들거나 수정할 권한이 없습니다. AWS 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 API를 사용하여 작업을 수행할 수 없습니다. AWS 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용자 설명서의 IAM 정책 생성을 참조하세요.

각 리소스 유형의 ARN 형식을 포함하여 AWS 메인프레임 현대화로 정의된 작업 및 리소스 유형에 대한 자세한 내용은 서비스 권한 부여 참조의 AWS 메인프레임 현대화를 위한 작업, 리소스 및 조건 키를 참조하십시오.

주제

- 정책 모범 사례
- 메인프레임 현대화 콘솔 사용 AWS
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

ID 기반 정책은 누군가가 계정에서 메인프레임 현대화 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부를 결정합니다. AWS 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책으로 시작하여 최소 권한 권한으로 이동 — 사용자와 워크로드에 권한을 부여하려면 여러 일반적인 사용 사례에 권한을 부여하는 AWS 관리형 정책을 사용하십시오. 해당 내용은 에서 사용할 수 있습니다. AWS 계정사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 더 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용자 설명서의 AWS managed policies(관리형 정책) 또는 AWS managed policies for job functions(직무에 대한 관리형 정책)를 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 least-privilege permissions(최소 권한)으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용자 설명서에 있는 Policies and permissions in IAM(IAM의 정책 및 권한)을 참조하세요.
- Use conditions in IAM policies to further restrict access(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 생성할 수 있습니다. 예를 AWS 서비스들어 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 AWS CloudFormation있습니다. 자세한 정보는 IAM 사용자 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 권장 사항을 제공하여 안전하고 기능적인 정책을 생성하도록 돕습니다. 자세한 정보는 IAM 사용자 설명서의 IAM Access Analyzer policy validation(IAM Access Analyzer 정책 검증)을 참조하세요.
- 멀티 팩터 인증 (MFA) 필요 - IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 AWS 계정 MFA를 활성화하십시오. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용자 설명서의 Configuring MFA-protected API access(MFA 보호 API 액세스 구성)를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용자 설명서의 IAM의 보안 모범 사례를 참조하세요.

메인프레임 현대화 콘솔 사용 AWS

AWS 메인프레임 현대화 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한을 통해 귀사의 AWS 메인프레임 현대화 리소스를 나열하고 해당 리소스에 대한 세부 정보를 볼 수 있어야 합니다. AWS 계정최소 필수 권한보다 더 제한적인 ID 기반 정책을 만들면 콘솔이 해당 정책에 연결된 개체(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요는 없습니다. AWS 그 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 계속해서 메인프레임 현대화 콘솔을 사용할 수 있도록 하려면 AWS 메인프레임 현대화 `ConsoleAccess` 또는 `ReadOnly` AWS 관리형 정책도 AWS 엔티티에 연결하십시오. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다. AWS CLI AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
```

```
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 메인프레임 현대화 ID 및 액세스 문제 해결

다음 정보를 사용하면 AWS 메인프레임 현대화 및 IAM을 사용할 때 발생할 수 있는 일반적인 문제를 진단하고 해결하는 데 도움이 됩니다.

주제

- 저는 IAM을 수행할 권한이 없습니다. PassRole
- 제 외부 사람들도 제 AWS 메인프레임 현대화 AWS 계정 리소스에 액세스할 수 있도록 하고 싶습니다.

저는 IAM을 수행할 권한이 없습니다. PassRole

작업을 수행할 권한이 없다는 오류가 발생하는 경우 AWS 메인프레임 현대화로 역할을 넘길 수 있도록 정책을 업데이트해야 합니다. `iam:PassRole`

일부 AWS 서비스 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 AWS 메인프레임 현대화에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우 Mary가 `iam:PassRole` 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요하면 관리자에게 문의하세요. AWS 관리자는 로그인 자격 증명을 제공한 사람입니다.

제 외부 사람들도 제 AWS 메인프레임 현대화 AWS 계정 리소스에 액세스할 수 있도록 하고 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 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- AWS 메인프레임 현대화가 이러한 기능을 지원하는지 여부를 알아보려면 을 참조하십시오. AWS 메인프레임 현대화가 IAM과 함께 작동하는 방식
- 소유하고 있는 모든 AWS 계정 리소스에 대한 액세스를 제공하는 방법을 알아보려면 IAM 사용 설명서의 다른 AWS 계정 IAM 사용자에게 액세스 권한 제공을 참조하십시오.
- 제3자에게 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정AWS 계정 소유에 대한 액세스 제공을 참조하십시오.
- 자격 증명 연동을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용자 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(자격 증명 연동)을 참조하세요.
- 크로스 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용자 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조합니다.

Mainframe Modernization을 위한 서비스 연결 역할 사용

AWS Mainframe Modernization은 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Mainframe Modernization에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Mainframe Modernization에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 직접적으로 호출하기 위해 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 Mainframe Modernization을 더 쉽게 설정할 수 있습니다. Mainframe Modernization에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, Mainframe Modernization만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 Mainframe Modernization 리소스에 대한 액세스 권한을 부주의로 삭제할 수 없기 때문에 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용을 알아보려면 AWS IAM으로 작업하는 서비스를 참조하고 서비스 연결 역할 열에 예가 표시된 서비스를 찾으세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Mainframe Modernization을 위한 서비스 연결 역할 권한

Mainframe Modernization에서는 AWSServiceRoleForAWSM2라는 서비스 연결 역할을 사용합니다. 즉, VPC에 연결하고 파일 시스템과 같은 리소스에 액세스하도록 네트워크를 구성합니다.

AWSServiceRoleForAWSM2 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `m2.amazonaws.com`

이름이 AWSM2ServicePolicy인 연결 권한 정책은 Mainframe Modernization가 Site-to-Site VPN이 지정된 리소스에 대해 다음 작업을 수행하도록 허용합니다.

- Mainframe Modernization 환경을 위한 Amazon EC2 네트워크 인터페이스에 대한 권한을 생성, 삭제, 설명하고 첨부하여 고객 VPC에 대한 연결을 설정합니다.
- 고객이 Mainframe Modernization 환경에 연결하는 방법인 Elastic Load Balancing에서 항목을 등록 또는 등록 취소합니다.
- Amazon EFS 또는 Amazon FSx 파일 시스템 (사용 중인 경우)에 대해 설명하세요.
- 런타임 환경에서 고객의 CloudWatch로 지표를 내보냅니다.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeSubnets",
    "ec2:CreateNetworkInterface",
    "ec2:DeleteNetworkInterface",
    "ec2:DescribeNetworkInterfaces",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:ModifyNetworkInterfaceAttribute"
   ],
   "Resource": "*"
  },
  {
```

```
    "Effect": "Allow",
    "Action": [
     "elasticfilesystem:DescribeMountTargets"
    ],
    "Resource": "*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "elasticloadbalancing:RegisterTargets",
     "elasticloadbalancing:DeregisterTargets"
    ],
    "Resource": "*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "fsx:DescribeFileSystems"
    ],
    "Resource": "*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "cloudwatch:PutMetricData"
    ],
    "Resource": "*",
    "Condition": {
     "StringEquals": {
      "cloudwatch:namespace": [
       "AWS/M2"
      ]
     }
    }
   }
  ]
}
```

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 연결 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

Mainframe Modernization을 위한 서비스 연결 역할 생성

서비스 연결 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console, AWS CLI 또는 AWS API에서 런타임 환경을 생성할 때 Mainframe Modernization이 서비스 연결 역할을 자동으로 생성합니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 런타임 환경을 생성할 때 Mainframe Modernization은 서비스 연결 역할을 다시 자동으로 생성합니다.

Mainframe Modernization을 위한 서비스 연결 역할 편집

에서는 AWSServiceRoleForAWSM2 서비스 연결 역할을 편집할 수 없습니다. 서비스 연결 역할을 생성한 후에는 다양한 객체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

Mainframe Modernization을 위한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 따라서 적극적으로 모니터링하거나 유지하지 않는 미사용 엔터티가 없도록 합니다. 단, 서비스 연결 역할에 대한 리소스를 먼저 정리해야 수동으로 삭제할 수 있습니다.

> **Note**
>
> 리소스를 삭제하려 할 때 Mainframe Modernization 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하십시오.

AWSServiceRoleForAWSM2에서 사용하는 Mainframe Modernization 리소스를 삭제하려면

- Mainframe Modernization에서 런타임 환경을 삭제합니다. 환경 자체를 삭제하기 전에 먼저 환경에서 애플리케이션을 삭제해야 합니다.

IAM을 사용하여 수동으로 서비스 연결 역할을 삭제하려면

IAMIAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AWSServiceRoleForAWSM2 서비스 연결 역할을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하세요.

Mainframe Modernization 서비스 연결 역할을 지원하는 리전

Mainframe Modernization에서는 서비스를 사용할 수 있는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용을 알아보려면 AWS 서비스 엔드포인트를 참조하세요.

AWS 메인프레임 현대화를 위한 규정 준수 확인 확인

타사 감사자는 여러 AWS 규정 준수 프로그램의 일환으로 AWS 메인프레임 현대화의 보안 및 규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

특정 규정 준수 프로그램의 범위 내에 있는 AWS 서비스 목록은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하세요. 일반적인 정보는 AWS 규정 준수 프로그램을 참조하세요.

AWS Artifact를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하세요.

AWS 메인프레임 현대화 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 – 이 배포 안내서에서는 아키텍처 고려 사항에 관해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- HIPAA 보안 및 규정 준수 기술 백서 아키텍팅 - 이 백서는 기업에서 AWS를 사용하여 HIPAA를 준수하는 애플리케이션을 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 - 고객 조직이 속한 산업 및 위치에 적용될 수 있는 워크북 및 가이드 콜렉션입니다.
- AWS Config 개발자 가이드의 규칙을 사용하여 리소스 평가 – AWS Config를 사용하여 리소스 구성이 내부 사례, 업계 지침, 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub - 이 AWS 서비스는 보안 산업 표준 및 모범 사례 규정 준수 여부를 확인하는 데 도움이 되도록 AWS 내 보안 상태를 종합적으로 보여줍니다.

AWS Mainframe Modernization의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크를 통해 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 정보는 AWS 글로벌 인프라를 참조하세요.

AWS Mainframe Modernization의 인프라 보안

관리형 서비스인 AWS Mainframe Modernization은 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스와 AWS의 인프라 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안에 대한 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 Mainframe Modernization에 액세스합니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 보안 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)을 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

인터페이스 엔드포인트(AWS PrivateLink)를 사용하여 AWS Mainframe Modernization에 액세스

AWS PrivateLink를 사용하여 VPC와 AWS Mainframe Modernization 사이에 프라이빗 연결을 생성할 수 있습니다. 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결을 사용하지 않고 VPC에 있는 것처럼 메인프레임 현대화에 액세스할 수 있습니다. VPC의 인스턴스에서 메인프레임 현대화에 액세스하는 데 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 메인프레임 현대화로 향하는 트래픽의 진입점 역할을 하는 요청자 관리형 네트워크 인터페이스입니다.

자세한 내용은 AWS PrivateLink 가이드의 AWS PrivateLink를 통해 AWS 서비스에 액세스를 참조하세요.

메인프레임 고려 사항

메인프레임 현대화에 대한 인터페이스 엔드포인트를 설정하려면 먼저 AWS PrivateLink 안내서의 고려 사항을 검토합니다.

메인프레임 현대화는 인터페이스 엔드포인트 수행을 지원합니다.

메인프레임 현대화를 위한 인터페이스 엔드포인트 생성

Amazon VPC 콘솔 또는 AWS Command Line Interface(AWS CLI)을 사용하여 메인프레임 현대화의 인터페이스 엔드포인트를 생성할 수 있습니다. 자세한 내용은 AWS PrivateLink 가이드의 인터페이스 엔드포인트 생성을 참조하세요.

다음과 같은 서비스 이름을 사용하여 메인프레임 현대화의 인터페이스 엔드포인트를 생성합니다.

```
com.amazonaws.region.m2
```

인터페이스 엔드포인트에 프라이빗 DNS를 사용하도록 설정하는 경우, 리전에 대한 기본 DNS 이름을 사용하여 메인프레임 현대화에 API 요청을 할 수 있습니다. 예: `m2.us-east-1.amazonaws.com`.

인터페이스 엔드포인트에 엔드포인트 정책 생성

엔드포인트 정책은 인터페이스 인터페이스 엔드포인트에 연결할 수 있는 IAM 리소스입니다. 기본 엔드포인트 정책을 사용하면 인터페이스 엔드포인트를 통해 메인프레임 현대화에 대한 전체 액세스를 허용합니다. VPC에서 메인프레임 현대화에 허용되는 액세스를 제어하려면 사용자 지정 엔드포인트 정책을 인터페이스 엔드포인트에 연결합니다.

엔드포인트 정책은 다음 정보를 지정합니다.

- 작업(AWS 계정 사용자 및 IAM 역할)을 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

자세한 정보는 AWS PrivateLink 안내서의 엔드포인트 정책을 사용하여 서비스에 대한 액세스 제어를 참조하세요.

예제: 메인프레임 현대화 작업에 대한 VPC 엔드포인트 정책

다음은 사용자 지정 엔드포인트 정책의 예입니다. 이 정책은 엔드포인트에 연결될 때 모든 리소스의 모든 보안 주체에 대한 액세스 권한을 나열된 메인프레임 현대화 작업에 부여합니다.

```
//Example of an endpoint policy where access is granted to the
//listed AWS Mainframe Modernization actions for all principals on all resources
{"Statement": [
      {"Principal": "*",
         "Effect": "Allow",
         "Action": [
            "m2:ListApplications",
            "m2:ListEnvironments",
            "m2:ListDeployments"
         ],
         "Resource":"*"
      }
   ]
}

//Example of an endpoint policy where access is denied to all the
//AWS Mainframe Modernization CREATE actions for all principals on all resources
{"Statement": [
      {"Principal": "*",
         "Effect": "Deny",
         "Action": [
            "m2:Create*"
         ],
         "Resource":"*"
      }
   ]
}
```

AWS 메인프레임 현대화 모니터링

모니터링은 AWS 메인프레임 현대화 및 기타 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 있어 중요한 부분입니다. AWS는 AWS 메인프레임 현대화를 관찰하고, 문제 발생 시 보고하고, 적절한 경우 자동 조치를 취할 수 있도록 다음과 같은 모니터링 도구를 제공합니다.

- Amazon은 실행 중인 AWS 리소스와 애플리케이션을 AWS 실시간으로 CloudWatch 모니터링합니다. 지표를 수집 및 추적하고, 맞춤 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임계값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 Amazon EC2 인스턴스의 CPU 사용량 또는 기타 지표를 CloudWatch 추적하고 필요할 때 새 인스턴스를 자동으로 시작할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오.
- Amazon CloudWatch Logs를 사용하면 Amazon EC2 인스턴스 및 기타 소스에서 로그 파일을 모니터링 CloudTrail, 저장 및 액세스할 수 있습니다. CloudWatch 로그는 로그 파일의 정보를 모니터링하고 특정 임계값이 충족되면 알려줄 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하십시오.
- AWS CloudTrail계정에서 또는 AWS 계정을 대신하여 이루어진 API 호출 및 관련 이벤트를 캡처하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 어떤 사용자와 계정이 전화를 걸었는지 AWS, 어떤 소스 IP 주소에서 호출이 이루어졌는지, 언제 호출이 발생했는지 식별할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

Amazon을 통한 AWS 메인프레임 현대화 모니터링 CloudWatch

원시 데이터를 수집하여 읽을 수 있는 거의 실시간 지표로 처리하는 를 사용하여 AWS CloudWatch 메인프레임 현대화를 모니터링할 수 있습니다. 이러한 통계는 15개월간 보관되므로 기록 정보에 액세스하고 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 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 참조하십시오.

다음 표에는 AWS 메인프레임 현대화에 대한 지표와 차원이 나와 있습니다. 지표의 네임스페이스는 `AWS/M2`입니다.

런타임 환경 지표

지표	설명
CPUUtilization	환경 내 인스턴스의 CPU 사용률.

지표	설명
	차원: environmentId 단위: 백분율 유효 통계: 평균, 최소, 최대
InboundNetworkThroughput	환경 내 인스턴스의 인바운드 네트워크 처리량. 차원: environmentId 단위: 바이트/초 유효 통계: 평균, 최소, 최대
MemoryUtilization	환경 내 인스턴스의 메모리 사용률. 차원: environmentId 단위: 백분율 유효 통계: 평균, 최소, 최대
OutboundNetworkThroughput	환경 내 인스턴스의 아웃바운드 네트워크 처리량. 차원: environmentId 단위: 바이트/초 유효 통계: 평균, 최소, 최대

애플리케이션 지표

지표	설명
BatchJobCompletedCount	시간 간격 동안 완료된 작업 수입니다. 이 지표는 마이크로 포커스 및 AWS 블루 에이지 3.7.0 이상 릴리스에서 사용할 수 있습니다.

지표	설명
	차원: applicationId 단위: 개 유효 통계: Sum
BatchJobFailedCount	해당 시간 간격 동안 실패한 작업 수입니다. 이 지표는 마이크로 포커스와 AWS 블루 에이지 3.7.0 이상 릴리스에서 사용할 수 있습니다. 차원: applicationId 단위: 개 유효 통계: Sum
JvmMemoryFree	Java 가상 머신에서 현재 사용하고 있지 않은 사용 가능한 메모리의 양. 이 지표는 AWS Blu Age 런타임 엔진에서만 사용할 수 있습니다. AWS Blu Age 3.7.0 이상 버전에서 사용할 수 있습니다. 차원: applicationId 단위: 바이트 유효 통계: 평균, 최소, 최대

지표	설명
JvmMemoryMax	Java 가상 머신에 허용되는 최대 메모리 양입니다. 이 지표는 AWS Blu Age 런타임 엔진에서만 사용할 수 있습니다. AWS Blu Age 3.7.0 이상 버전에서 사용할 수 있습니다. 차원: applicationId 단위: 바이트 유효 통계: 평균, 최소, 최대
JvmMemoryUsed	Java 가상 머신이 적극적으로 사용하는 메모리 양. 이 지표는 AWS Blu Age 런타임 엔진에서만 사용할 수 있습니다. AWS Blu Age 3.7.0 이상 버전에서 사용할 수 있습니다. 차원: applicationId 단위: 바이트 유효 통계: 평균, 최소, 최대
ProcessesActiveCount	요청을 처리 중인 동시 서비스 실행 프로세스의 활성 수입니다. 이 지표는 Micro Focus 런타임 엔진에만 사용할 수 없습니다. 차원: applicationId 단위: 개 유효 통계: Sum

지표	설명
SessionCount	애플리케이션의 HTTP 세션 수. 이 지표는 AWS Blu Age 런타임 엔진에서만 사용할 수 있습니다. AWS Blu Age 3.7.0 이상 버전에서 사용할 수 있습니다. 차원: applicationId 단위: 개 유효 통계: 평균, 최소, 최대
SharedMemoryFree	엔터프라이즈 서버가 트랜잭션과 작업을 실행하는 데 필요한 모든 정보를 저장하는 데 사용할 수 있는 메모리입니다. 이 지표는 Micro Focus 런타임 엔진에만 사용할 수 없습니다. 차원: applicationId 단위: 개 유효 통계: 평균, 최소, 최대
ThreadActiveCount	요청을 처리 중인 엔진 스레드 수. 이 지표는 AWS Blu Age 런타임 엔진에서만 사용할 수 있습니다. AWS Blu Age 3.7.0 이상 버전에서 사용할 수 있습니다. 차원: applicationId 단위: 개 유효 통계: 평균, 최소, 최대

지표	설명
TransactionCompletedCount	시간 간격 동안 커밋된 트랜잭션 수입니다. 이 지표는 마이크로 포커스와 AWS 블루 에이지 3.7.0 이상 릴리스에서 사용할 수 있습니다. 차원: applicationId 단위: 개 유효 통계: Sum
TransactionFailedCount	해당 시간 간격 동안 실패한 트랜잭션 수입니다. 이 지표는 마이크로 포커스와 AWS 블루 에이지 3.7.0 이상 릴리스에서 사용할 수 있습니다. 차원: applicationId 단위: 개 유효 통계: Sum
TransactionResponseTime	사용자가 요청을 보낸 순간부터 애플리케이션이 요청이 완료되었다고 표시하는 시점까지의 시간입니다. 이 지표는 마이크로 포커스와 AWS 블루 에이지 3.7.0 이상 릴리스에서 사용할 수 있습니다. 차원: applicationId 단위: 밀리초 유효 통계: 평균, 최소, 최대

차원

차원	설명
applicationId	이 차원은 지표를 ID별로 식별된 애플리케이션으로 필터링합니다.
environmentId	이 차원은 ID를 기준으로 지표를 식별된 환경으로 필터링합니다.

AWS CloudTrail를 사용하여 AWS 메인프레임 현대화 API 호출 로깅

AWS 메인프레임 현대화는 AWS 메인프레임 현대화에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail와 통합됩니다. CloudTrail은 AWS 메인프레임 현대화에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 AWS 메인프레임 현대화 콘솔로부터의 호출과 AWS 메인프레임 현대화 API 작업에 대한 호출이 포함됩니다. 추적을 생성하면 AWS 메인프레임 현대화를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 AWS 메인프레임 현대화에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서 섹션을 참조하세요.

CloudTrail의 AWS 메인프레임 현대화 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. AWS 메인프레임 현대화에서 활동이 수행되면 해당 활동은 Event history(이벤트 기록)에서 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 정보는 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

AWS 메인프레임 현대화에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려는 경우 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 지역의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 추가적으로, CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하십시오.

- 추적 생성 개요

- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 수신
- 여러 계정에서 CloudTrail 로그 파일 수신

모든 AWS 메인프레임 현대화 작업은 CloudTrail에서 로깅되며 AWS 메인프레임 현대화 API 참조에 설명되어 있습니다. 예를 들어, `CreateApplication`, `CreateEnvironment`, `CreateDeployment` 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 보안 인증 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트 사용자로 했는지 사용자 보안 인증으로 했는지 여부.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증 정보를 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지.

자세한 내용은 CloudTrail userIdentity 요소를 참조하십시오.

AWS 메인프레임 현대화 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 전송할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 정렬된 스택 기록이 아니므로 특정 순서로 표시되지 않습니다.

다음은 `CreateApplication` 작업을 보여주는 CloudTrail 로그 항목이 나타낸 예시입니다.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAII6WZTHGYAEXAMPLE",
        "arn": "arn:aws:sts::444455556666:assumed-role/Admin/Mary_Major",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAII6WZTHGYAEXAMPLE",
```

```
                "arn": "arn:aws:iam::444455556666:role/Admin",
                "accountId": "444455556666",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-06-01T20:38:22Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-06-01T20:40:39Z",
    "eventSource": "m2.amazonaws.com",
    "eventName": "CreateApplication",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.196.65",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:91.0) Gecko/20100101
 Firefox/91.0",
    "requestParameters": {
        "clientToken": "1abc23de-f45g-6789-h01i-jkl2m3456789",
        "name": "MyApp",
        "description": "",
        "engineType": "microfocus",
        "definition": {
            "content": "{}"
        },
        "tags": {}
    },
    "responseElements": {
        "applicationVersion": 1,
        "Access-Control-Expose-Headers": "x-amzn-RequestId,x-amzn-ErrorType,x-amzn-
ErrorMessage,Date",
        "applicationArn": "arn:aws:m2:us-east-1:444455556666:app/
lsfhmwhw7fffrosff2lncwqcua",
        "applicationId": "lsfhmwhw7fffrosff2lncwqcua"
    },
    "requestID": "36982d38-fcde-4bfe-a89a-7bd78d43c926",
    "eventID": "d7f0fc36-46ae-4157-9a79-c79f385fda98",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "444455556666",
    "eventCategory": "Management"
```

```
}
```

문제 해결

이 섹션의 정보를 사용하면 AWS Blu Age 엔진과 Micro Focus 엔진을 모두 사용하는 AWS Mainframe Modernization 애플리케이션 및 런타임 환경에서 흔히 발생하는 오류를 해결하는 데 도움이 됩니다.

주제

오류: 데이터 세트 이름의 잠금이 해제될 때까지 기다리는 동안 시간이 초과되었습니다

- 엔진: AWS Blue Age
- 구성 요소: Blusam

AWSBlu Age 엔진을 사용하고 고가용성 패턴의 환경에서 실행되는 AWS 메인프레임 현대화 애플리케이션에 대한 Amazon CloudWatch 로그에 이 오류가 표시되면 다른 애플리케이션이 공유 데이터 세트를 잠근 상태임을 나타냅니다. 일반적으로 이 상황은 다른 애플리케이션이 충돌하거나 장애가 발생하여 잠금이 해제되지 않는 경우에 발생합니다.

이 오류가 발생하는 방법

애플리케이션 `example-app-1`이 쓰기 작업을 위해 `example-record-1` 레코드를 잠그려고 합니다. 이 작업을 수행하면 `example-record-1`을 소유하는 데이터 세트에 대한 잠금과 `example-record-1` 자체 잠금이 모두 생성됩니다. 이제 다른 애플리케이션`example-app-2`가 동일한 `example-record-1` 레코드를 잠그려고 합니다. 데이터 세트와 레코드가 이미 잠겼으므로 `example-app-2`는 잠금이 해제될 때까지 기다립니다. `example-app-1`에 충돌이 발생하는 경우 데이터 세트에 대한 잠금이 `example-dataset-1` 여전히 존재하므로 쓰기 시도가 취소되고 제한 시간 예외가 발생합니다. 이러한 교착 상태로 인해 모든 애플리케이션이 `example-dataset-1`에 접속할 수 없습니다.

이런 상황인지 어떻게 알 수 있나요?

실패한 애플리케이션을 찾아 오류 메시지에 언급된 것과 동일한 데이터 세트를 사용하는지 확인하세요. 애플리케이션이 고가용성 패턴의 런타임 환경에서 실행되고 있는지 확인하세요. 제한 시간 예외가 발생한 애플리케이션은 계속할 수 없으며 `Failed` 상태가 표시됩니다.

무엇을 할 수 있나요?

상황을 즉시 해결하려면 잠금을 강제로 해제할 수 있습니다. 향후 유사한 상황이 발생하지 않도록 Blusam 자동 복구 메커니즘을 제어하는 두 개의 매개 변수를 구성 할 수 있습니다.

잠금을 강제로 해제합니다

Blusam 잠금 관리자는 Amazon ElastiCache for Redis를 사용하여 애플리케이션 간에 공유 잠금을 제공합니다. 잠금을 해제하려면 Redis CLI 유틸리티를 사용합니다. ElastiCache 개별 레코드 잠금은 삭제할 수 없습니다. 소유한 데이터 세트에서 모든 잠금을 제거해야 합니다. 다음 단계를 완료합니다.

1. 다음 명령을 ElastiCache 사용하여 컴퓨터에 연결합니다.

   ```
   redis-cli -h hostname -p port
   ```

   ElastiCache 콘솔의 https://console.aws.amazon.com/elasticache/ ElastiCache 에서 세부 정보를 찾을 수 있습니다.
2. 암호를 입력합니다.
3. 실행하려는 명령을 다음과 같이 입력합니다.

Command	용도
KEYS *	기존 키를 모두 가져옵니다.
KEYS *YOUR_DATASET_NAME*	데이터 세트 잠금 키를 받으세요.
DEL *THE_RETURNED_KEY*	데이터 세트 잠금을 삭제합니다.
FLUSHDB	전체 레디스를 청소하세요.

Command	용도
	Warning Redis 캐시의 모든 데이터가 손실됩니다. Redis를 http 세션 처리와 같은 다른 용도로 사용하는 경우에는 Blusam 자동 복구 메커니즘 FLUSHDB 구성 사용하지 않는 것이 좋습니다.

Blusam 자동 복구 메커니즘 구성

Blusam 잠금 관리자에는 데이터 세트 또는 레코드의 교착 상태를 방지하는 자동 복구 메커니즘이 포함되어 있습니다. 애플리케이션 정의(`application-main.yml`)에서 다음 매개 변수를 조정하여 자동 복구 메커니즘을 구성할 수 있습니다.

- `locksDeadTime`: 애플리케이션이 잠금을 유지할 수 있는 최대 시간을 나타냅니다. 이 시간이 지나면 잠금이 만료된 것으로 선언되고 즉시 해제됩니다. `locksDeadTime` 값은 밀리초 단위이고 기본값은 1000입니다.
- `locksCheck`: 잠금 검사를 위한 Blusam 잠금 관리자 전략을 정의합니다. 모든 Blusam ElastiCache 락인에는 타임스탬프가 찍혀 있으며 만료 시간이 있습니다. `locksCheck` 매개 변수 값에 따라 만료된 잠금의 제거 여부가 결정됩니다.
  - `off`: 어떤 경우에도 확인이 실행되지 않습니다. 데드락 상태가 발생할 수 있습니다. (권장되지 않음)
  - `reboot`: AWS Mainframe Modernization 런타임 환경에서 실행되는 AWS Mainframe Modernization 애플리케이션 인스턴스가 시작되거나 재부팅될 때 검사가 실행됩니다. 만료된 모든 잠금은 즉시 해제됩니다. (기본값)
  - `timeout`: AWS Mainframe Modernization 런타임 환경에서 실행되는 AWS Mainframe Modernization 애플리케이션 인스턴스가 시작 또는 재부팅될 때 또는 데이터 세트를 잠그려고 시도하는 동안 제한 시간이 만료되면 검사가 실행됩니다. 만료된 잠금은 즉시 해제됩니다.

AWS Blu Age 애플리케이션의 애플리케이션 정의에 대한 자세한 내용은 AWS 블루 에이지 애플리케이션 정의 샘플를 참조하세요.

Blusam 잠금 관리자

고가용성 패턴을 사용하는 AWS Mainframe Modernization 런타임 환경의 맥락에서 AWS Blu Age 애플리케이션은 여러 번 배포될 수 있습니다. Blusam 데이터 세트를 처리하는 애플리케이션의 경우 동시 액세스 문제가 발생할 수 있습니다. Blusam 잠금 관리자는 사용하는 애플리케이션 간에 공유 잠금을 제공하여 데이터 무결성을 보장하고 레코드 및 데이터 세트에 대한 읽기 및 쓰기 액세스를 관리합니다. ElastiCache 이 메커니즘을 사용하면 둘 이상의 애플리케이션이 동시에 레코드를 읽을 수 있으며 한 번에 하나의 애플리케이션만 레코드를 쓸 수 있습니다.

쓰기 잠금

특정 레코드를 업데이트하거나 삭제하려면 애플리케이션이 먼저 레코드를 소유한 데이터 세트를 잠근 다음 레코드 자체를 잠가야 합니다. 레코드가 잠기면 데이터 세트 잠금이 해제되고 동일한 데이터셋의 다른 레코드를 사용할 수 있게 됩니다. 업데이트 또는 삭제 작업이 완료되면 보류된 레코드 잠금이 해제됩니다. 레코드를 업데이트할 수 있는 애플리케이션은 한 번에 하나뿐입니다. 이렇게 하면 정의된 애플리케이션 정책에서 해제 대기를 허용하는 경우 잠금이 해제될 때까지 다른 애플리케이션이 읽거나 쓸 수 없습니다.

읽기 잠금

레코드 또는 데이터 세트에 쓰기 잠금이 유지되지 않는 한 여러 애플리케이션에서 동시에 동일한 레코드를 읽을 수 있습니다. 쓰기 작업을 위해 레코드를 잠그려면 모든 읽기 잠금을 해제해야 합니다.

> **Note**
>
> Blusam 잠금 관리자는 동일한 잠금 메커니즘을 사용하여 지정된 애플리케이션의 여러 스레드에서 액세스를 처리합니다.

애플리케이션 URL에 접근할 수 없습니다

- 엔진: AWS Blu Age 및 Micro Focus
- 구성 요소: 애플리케이션

AWS Mainframe Modernization 런타임 환경에 생성하여 배포한 실행 중인 AWS Mainframe Modernization 애플리케이션의 URL에 액세스할 수 없는 경우 런타임 환경과 연결한 보안 그룹에 인바운드 규칙을 구성해야 할 수 있습니다.

이 오류가 발생하는 방법

런타임 환경을 만들 때 기본 보안 그룹을 포함하여 제공하는 보안 그룹에 VPC 외부에서 배포된 애플리케이션으로의 트래픽을 허용하도록 구성된 인바운드 규칙이 있어야 합니다(이러한 유형의 액세스를 허용하려면).

이런 상황인지 어떻게 알 수 있나요?

애플리케이션이 성공적으로 시작되어 정상적으로 실행되고 있지만 해당 URL을 사용하여 연결할 수 없습니다.

무엇을 할 수 있나요?

런타임 환경과 연결된 Amazon VPC 보안 그룹이 적절한 애플리케이션 포트를 통해 환경으로의 트래픽을 허용하는지 확인합니다. 보안 그룹 규칙을 확인하려면 다음 절차를 수행합니다.

1. https://console.aws.amazon.com/m2/에서 AWS Mainframe Modernization 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 환경을 선택합니다.
3. 연결하려는 애플리케이션을 호스팅하는 런타임 환경을 선택합니다.
4. 구성을 선택합니다.
5. 보안 및 네트워크에서 보안 그룹을 선택합니다. 링크를 클릭하면 Amazon VPC 콘솔에서 보안 그룹의 세부 정보가 열립니다.
6. 필요한 경우 인바운드 규칙 편집 을 선택하고 다음 규칙이 아직 없는 경우 추가합니다.

   유형

   사용자 지정 TCP

   Port

   8196 또는 애플리케이션 정의에 지정된 리스너 속성과 일치하는 포트. 자세한 설명은 2단계: 애플리케이션 정의 만들기 섹션을 참조하세요.

   소스(Source)

   애플리케이션을 호출하는 데 사용한 IP 주소. 드롭다운에서 myIP를 선택할 수 있습니다. 여전히 시간 제한 문제가 있다면 Anywhere IPV4 또는 Anywhere IPV6 선택해 보세요. 보안 그룹에 인바운드 규칙을 추가한 후 애플리케이션을 중지하고 다시 시작해야 합니다.

   자세한 내용은 Amazon VPC 사용 설명서의 보안 그룹 규칙 작업을 참조하세요.

AWS Blu Insights는 콘솔에서 열리지 않습니다

- 엔진: AWS Blu Age
- 구성 요소: Blu Insights

AWS Mainframe Modernization 콘솔에서 Blu Insights에 액세스하려고 하면 콘솔이 열리지 않고 새 탭이 즉시 닫힙니다.

이 오류가 발생하는 방법

Blu Insights에 액세스하는 데 사용하는 역할에 충분한 권한이 없습니다.

무엇을 할 수 있나요?

역할에 IAM 정책을 연결하여 Blu Insights에 액세스할 수 있도록 허용하세요. 정책에 최소한 다음 권한이 포함되어 있는지 확인하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "m2:GetSignedBluinsightsUrl"
            ],
            "Resource": "arn:aws:m2:region:account:*"
            }
    ]
}
```

`region` 및 `account`를 올바른 AWS 리전 및 AWS 계정로 교체합니다.

AWS Mainframe Modernization 사용 설명서의 문서

다음 표에서는 AWS Mainframe Modernization에 대한 문서 릴리스를 소개합니다.

변경 사항	설명	날짜
Micro Focus 튜토리얼의 관리형 런타임 업데이트	이 자습서에서는 Micro Focus 런타임 엔진을 사용하여 AWS 메인프레임 현대화 관리형 런타임 환경에서 CardDemo 샘플 애플리케이션을 배포하고 실행하는 방법을 보여줍니다.	2024년 2월 5일
AWS블루에이지 런타임 및 현대화 도구 버전 3.9.0의 릴리스 노트.	이번 AWS Blu Age 런타임 및 현대화 도구 릴리스는 작업 실행을 한 단계 끌어올리는 새로운 기능과 함께 고가용성 아키텍처의 성능을 높이기 위해 노력하는 제품 전반의 여러 가지 개선 사항에 초점을 맞추고 있습니다.	2023년 12월 18일
메인프레임과 AWS 간에 파일 전송	소스 메인프레임에서 AWS로 파일을 전송하는 새로운 기능이 출시되었습니다.	2023년 11월 27일
애플리케이션 트랜잭션 관리	AWS Mainframe Modernization을 위해 애플리케이션의 트랜잭션을 표시하고 편집하는 새로운 기능이 출시되었습니다.	2023년 10월 16일
AWS Blu Age 런타임 및 현대화 도구 버전 3.6.0의 릴리스 노트.	AWSBlu Age 런타임 및 현대화 도구의 이번 릴리스는 zOS 및 AS400 레거시 마이그레이션 모두에 대한 새로운 기능을 제공합니다. 이 기능은 주로	2023년 8월 4일

	CICS 지원 메커니즘 확장, JCL 기능 보완, 동시 및 대용량 기능의 성능 최적화, multi-data-source 기능 추가에 중점을 두고 있습니다.	
이제 애플리케이션이 중지되었을 때 새 버전의 애플리케이션을 배포할 수 있습니다.	이전에는 새 버전의 애플리케이션을 배포하려면 배포된 버전을 삭제해야 했습니다. 이제 배포된 버전을 중지하고 새 버전을 배포하기만 하면 됩니다.	2023년 7월 26일
간편한 Amazon EC2 배포를 위해 패키징된 AWS Blu Age 런타임	AWS이제 AWS Blu Age 런타임을 통한 Mainframe Modernization가 가능해짐에 따라 사용자 AWS 계정의 Amazon EC2 인스턴스에 전체 스택을 구성하고 배포할 수 있는 유연성이 향상되었습니다.	2023년 7월 6일
AWSAWS Blu Age Blu Insights에 싱글 사인온으로 접속하세요.	싱글 사인온을 통해 AWS Management Console에서 AWS AWS Blu Age Blu Insights를 이용할 수 있습니다.	2023년 3월 31일
GA 릴리스	AWS Mainframe Modernization 사용자 가이드의 GA 릴리스.	2022년 6월 8일
최초 릴리스	AWS Mainframe Modernization 사용 설명서의 초기 릴리스(공개 미리 보기).	2021년 11월 30일

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