구현 안내서

AWS의 분산 로드 테스트

AWS의 분산 로드 테스트: 구현 안내서

Table of Contents

대규모 소프트웨어 애플리케이션 테스트 자동화

게시 날짜: 2019년 11월

AWS의 분산 로드 테스트를 사용하면 애플리케이션을 릴리스하기 전에 대규모 및 로드 시 소프트웨어 애플리케이션의 테스트를 자동화하여 병목 현상을 식별할 수 있습니다. 이 솔루션은 서버를 프로비저닝할 필요 없이 일정한 속도로 트랜잭션 레코드를 생성하는 수천 명의 연결된 사용자를 생성하고 시뮬레이션합니다.

이 솔루션은 AWS Fargate의 Amazon Elastic Container Service(Amazon ECS)를 활용하여 모든 시뮬레이션을 실행할 수 있는 컨테이너를 배포하고 다음 기능을 제공합니다.

- 독립적으로 실행할 수 있는 Amazon ECS on AWS Fargate 컨테이너를 배포하여 테스트 중인 소프트웨어의 로드 기능을 테스트합니다.
- 여러 AWS 리전에서 수만 명의 연결된 사용자를 시뮬레이션하여 트랜잭션 레코드를 지속적으로 생성합니다.
- 사용자 지정 JMeter 스크립트를 생성하여 애플리케이션 테스트를 사용자 지정합니다.
- 로드 테스트가 미래 날짜 또는 반복 날짜에 자동으로 시작되도록 예약합니다.
- 애플리케이션 로드 테스트를 동시에 실행하거나 여러 테스트를 동시에 실행합니다.

이 구현 가이드에서는 AWS의 분산 로드 테스트 솔루션, 참조 아키텍처 및 구성 요소, 배포 계획 고려사항, 솔루션을 Amazon Web Services(AWS) 클라우드에 배포하기 위한 구성 단계에 대한 개요를 제공합니다. 여기에는 보안 및 가용성에 대한 AWS 모범 사례를 사용하여이 솔루션을 배포하는 데 필요한 AWS 서비스를 시작하고 구성하는 AWS CloudFormation 템플릿에 대한 링크가 포함되어 있습니다.

환경에서이 솔루션의 기능을 사용하기 위한 대상에는 AWS 클라우드에서 설계한 실제 경험이 있는 IT 인프라 아키텍트, 관리자 및 DevOps 전문가가 포함됩니다.

이 탐색 테이블을 사용하여 다음 질문에 대한 답을 빠르게 찾을 수 있습니다.

다음을 수행하려는 경우 . . .	읽기 . . .
이 솔루션을 실행하는 데 드는 비용을 파악합니다. 미국 동부(버지니아 북부) 리전에서이 솔루션을 실행하는 데 드는 예상 비용은 AWS 리소스에 대해 매월 30.90 USD입니다.	비용

다음을 수행하려는 경우 . . .	읽기 . . .
이 솔루션의 보안 고려 사항을 이해합니다.	보안
이 솔루션의 할당량을 계획하는 방법을 파악합니다.	할당량
이 솔루션을 지원하는 AWS 리전을 파악합니다.	지원되는 AWS 리전
이 솔루션에 포함된 AWS CloudFormation 템플릿을 보거나 다운로드하여이 솔루션의 인프라 리소스("스택")를 자동으로 배포합니다.	AWS CloudFormation 템플릿
소스 코드에 액세스하고 선택적으로 AWS 클라우드 개발 키트(AWS CDK)를 사용하여 솔루션을 배포합니다.	GitHub 리포지토리

Features

솔루션은 다음 기능을 제공합니다.

Out-of-the-Box 구성 가능한 성능 테스트

즉시 사용할 수 있는 사전 구성된 성능 테스트를 포함합니다.

사용자 지정 가능한 애플리케이션 테스트

유연하고 정확한 테스트 사용자 지정을 통해 잠재적 문제를 식별할 수 있습니다. JMeter 스크립트를 사용하여 특정 요구 사항 및 시나리오에 맞게 테스트를 조정합니다.

높은 사용자 로드를 시뮬레이션합니다.

수만 명의 연결된 사용자를 시뮬레이션하여 애플리케이션을 스트레스 테스트할 수 있습니다.

연속 트랜잭션 생성

트랜잭션 레코드를 지속적으로 생성하여 일정한 로드에서 성능을 평가합니다.

실시간 모니터링

테스트 진행 상황 및 결과에 대한 실시간 모니터링을 제공합니다. 지정된 날짜 또는 반복 간격으로 자동으로 시작되도록 테스트를 예약합니다.

리전 요청 시뮬레이션

모든 리전의 사용자 요청을 시뮬레이션하여 글로벌 성능을 평가합니다.

엔드포인트 유연성

AWS 리전, 온프레미스 환경 또는 기타 클라우드 공급자에서 엔드포인트를 테스트합니다.

세부 테스트 결과

평균 응답 시간, 동시 사용자 수, 성공한 요청 및 실패한 요청을 포함한 포괄적인 테스트 결과를 봅니다.

직관적 웹 콘솔

테스트 관리 및 모니터링을 위한 easy-to-use 웹 콘솔을 제공합니다.

여러 프로토콜 지원

WebSocket, HTTP, HTTPS, JDBC, JMS, FTP 및 gRPC와 같은 다양한 프로토콜과 호환됩니다.

AWS Systems Manager의 기능인 AWS Service Catalog AppRegistry AWS Systems Manager와 통합

이 솔루션에는 솔루션의 CloudFormation 템플릿과 기본 리소스를 Service Catalog AppRegistry 및 Application Manager 모두에 애플리케이션으로 등록하는 Service Catalog AppRegistry 리소스가 포함되어 있습니다. 이 통합을 통해 솔루션의 리소스를 중앙에서 관리하고 애플리케이션 검색, 보고 및 관리 작업을 활성화합니다.

이점

솔루션은 다음과 같은 이점을 제공합니다.

포괄적인 성능 테스트 지원

철저한 애플리케이션 평가를 위한 로드, 스트레스 및 내구성 테스트를 촉진합니다.

성능 문제의 조기 감지

프로덕션 릴리스 전에 성능 문제와 병목 현상을 식별합니다.

실제 사용 시뮬레이션

실제 사용 패턴을 정확하게 미러링하여 병목 현상과 최적화 영역을 강조합니다.

세부 성능 개선 도우미

상당한 부하 시 소프트웨어 성능 및 복원력에 대한 인사이트를 제공합니다.

자동 성능 평가

수동 개입 없이 정기적인 성능 평가를 활성화합니다.

비용 효율적인 테스트

pay-as-you-go 모델을 제공하므로 전용 테스트 인프라 및 구독 요금이 필요하지 않습니다.

사용 사례

프로덕션 로드 시뮬레이션

새 버전을 시작하기 전에 프로덕션과 유사한 조건에서 웹 및 모바일 애플리케이션을 테스트합니다.

애플리케이션 성능 검증

애플리케이션이 성능 저하 없이 예상 사용자 트래픽을 처리할 수 있는지 확인합니다. 기본 리소스를 사용하여 애플리케이션 제한을 테스트하고 인프라 확장성을 평가합니다.

피크 로드 관리

인프라가 최대 부하 또는 예상치 못한 트래픽 급증을 관리할 수 있는지 확인하여 수요가 많을 때 안정성을 보장합니다.

성능 최적화

애플리케이션의 성능 프로파일을 이해하고 비효율적인 코드 실행, 데이터베이스 쿼리, 네트워크 지연 시간과 같은 병목 현상을 식별합니다.

빠른 테스트 시작

out-of-the-box 성능 테스트로 빠르게 테스트를 시작합니다.

사용자 지정 가능한 테스트

테스트를 특정 시나리오 및 요구 사항에 맞게 조정하여 시작된 동시 사용자 및 작업 수를 조정합니다.

예약된 테스트

회귀 테스트 및 지속적인 성능 모니터링을 위한 테스트를 예약하여 일관된 애플리케이션 성능을 보장합니다.

지리적 성능 평가

다양한 지리적 리전에서 애플리케이션 성능을 평가하여 글로벌 효율성을 보장합니다.

CI/CD 파이프라인 통합

성능 테스트를 CI/CD 파이프라인에 통합하여 개발 주기 동안 원활하고 자동화된 테스트를 수행합니다.

개념 및 정의

이 섹션에서는 이 솔루션과 관련된 핵심 개념 및 용어에 대해 설명합니다.

시나리오

테스트 이름, 설명, 작업 수, 동시성, AWS 리전, 램프 업, 대기, 테스트 유형, 일정 날짜 및 반복 구성을 포함한 테스트 정의입니다.

작업 수

테스트 시나리오를 실행하기 위해 Fargate 클러스터에서 시작할 컨테이너 수입니다. Fargate 리소스에 대한 계정 한도에 도달하면 추가 작업이 생성되지 않습니다. 하지만 이미 실행 중인 작업은 계속됩니다.

concurrency

작업당 생성된 동시 가상 사용자 수입니다. 기본 설정에 따른 권장 제한은 가상 사용자 200명입니다. 동시성은 CPU와 메모리에 의해 제한됩니다.

증가

대상 동시성에 도달하는 시간입니다.

에 대해 보류

대상 동시성을 유지하는 시간입니다.

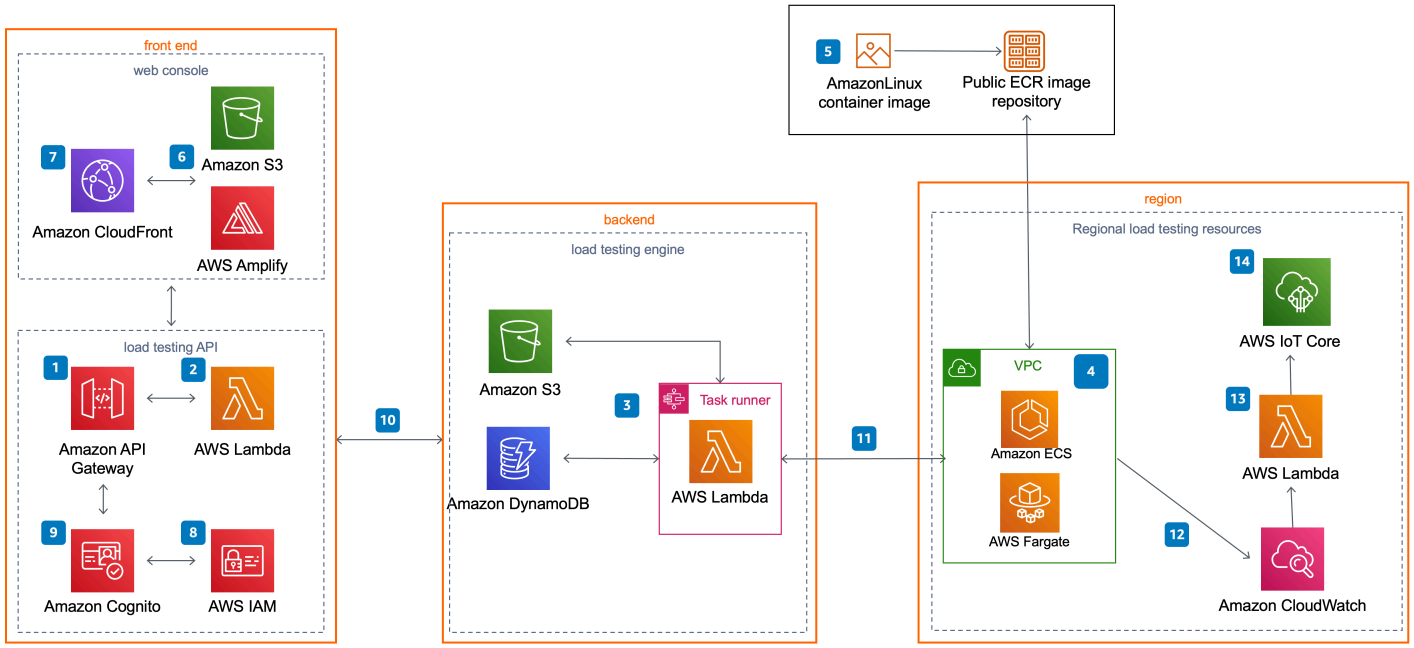
AWS 용어에 대한 일반적인 참조는 AWS 용어집을 참조하십시오.

아키텍처 개요

아키텍처 다이어그램

기본 파라미터로이 솔루션을 배포하면 AWS 계정에 다음 구성 요소가 배포됩니다.

AWS 아키텍처의 분산 로드 테스트



> **Note**
>
> AWS CloudFormation 리소스는 AWS Cloud Development Kit(AWS CDK) 구문에서 생성됩니다.

AWS CloudFormation 템플릿과 함께 배포된 솔루션 구성 요소의 상위 수준 프로세스 흐름은 다음과 같습니다.

1. 분산 로드 테스터 API는 Amazon API Gateway를 활용하여 솔루션의 마이크로서비스(AWS Lambda 함수)를 호출합니다.
2. 마이크로서비스는 테스트 데이터를 관리하고 테스트를 실행하는 비즈니스 로직을 제공합니다.

3. 이러한 마이크로서비스는 Amazon Simple Storage Service(Amazon S3), Amazon DynamoDB 및 AWS Step Functions와 상호 작용하여 테스트 시나리오 세부 정보 및 결과에 대한 스토리지를 제공하고 테스트 시나리오를 실행합니다.
4. Amazon Virtual Private Cloud(VPC) 네트워크 토폴로지는 AWS Fargate에서 실행되는 솔루션의 Amazon Elastic Container Service(Amazon ECS) 컨테이너를 포함하여 배포됩니다.
5. 컨테이너에는 AmazonLinux(블레이저 로드 테스트 프레임워크가 설치된 경우) OCI(Open Container Initiative) 호환 컨테이너 이미지가 포함되어 있습니다.이 이미지는 애플리케이션 성능을 테스트하기 위해 로드를 생성하는 데 사용됩니다. Taurus/Blazemeter는 오픈 소스 테스트 자동화 프레임워크입니다. 컨테이너 이미지는 AWS가 Amazon Elastic Container Registry(Amazon ECR) 퍼블릭 리포지토리에서 호스팅합니다. ECR 이미지 리포지토리에 대한 자세한 내용은 컨테이너 이미지 사용자 지정을 참조하세요.
6. AWS Amplify로 구동되는 웹 콘솔은 정적 웹 호스팅을 위해 구성된 Amazon S3 버킷에 배포됩니다.
7. Amazon CloudFront는 솔루션의 웹 사이트 버킷 콘텐츠에 대한 안전한 퍼블릭 액세스를 제공합니다.
8. 초기 구성 중에이 솔루션은 기본 솔루션 관리자 역할(IAM 역할)을 생성하고 고객이 지정한 사용자 이메일 주소로 액세스 초대를 보냅니다.
9. Amazon Cognito 사용자 풀은 콘솔 및 분산 로드 테스터 API에 대한 사용자 액세스를 관리합니다.
10. 이 솔루션을 배포한 후 웹 콘솔을 사용하여 일련의 작업을 정의하는 테스트 시나리오를 생성할 수 있습니다.
11. 마이크로서비스는이 테스트 시나리오를 사용하여 지정된 리전에서 AWS Fargate에서 Amazon ECS 작업을 실행합니다.
12. Amazon S3 및 DynamoDB에 결과를 저장하는 것 외에도 테스트가 완료되면 출력이 Amazon CloudWatch에 로깅됩니다.
13. 라이브 데이터 옵션을 선택하면 솔루션은 테스트 중에 테스트가 실행된 각 리전에 대해 AWS Fargate 작업에 대한 Amazon CloudWatch logs를 Lambda 함수로 전송합니다.
14. 그런 다음 Lambda 함수는 기본 스택이 배포된 리전의 AWS IoT Core에서 해당 주제에 데이터를 게시합니다. 웹 콘솔은 주제를 구독하며 테스트가 실행되는 동안 웹 콘솔에서 데이터를 볼 수 있습니다.

AWS Well-Architected 설계 고려 사항

이 솔루션은 고객이 클라우드에서 안정적이고 안전하며 효율적이고 비용 효율적인 워크로드를 설계하고 운영할 수 있도록 지원하는 AWS Well-Architected Framework의 모범 사례를 사용합니다.

이 섹션에서는 Well-Architected Framework의 설계 원칙과 모범 사례가 이 솔루션에 어떤 이점을 제공하는지 설명합니다.

운영 우수성

이 섹션에서는 운영 우수성 요소의 원칙과 모범 사례를 사용하여 이 솔루션을 설계한 방법을 설명합니다.

- CloudFormation을 사용하여 코드형 인프라로 정의된 리소스입니다.
- 솔루션은 다양한 단계에서 지표를 Amazon CloudWatch에 푸시하여 인프라, Lambda 함수, Amazon ECS 작업, Amazon S3 버킷 및 나머지 솔루션 구성 요소에 대한 관찰성을 제공합니다.

보안

이 섹션에서는 보안 요소의 원칙과 모범 사례를 사용하여 이 솔루션을 설계한 방법을 설명합니다.

- Amazon Cognito는 웹 UI 앱 사용자를 인증하고 승인합니다.
- 모든 서비스 간 통신은 해당 AWS Identity and Access Management(IAM) 역할을 사용합니다.
- 솔루션에서 사용하는 모든 역할은 최소 권한 액세스를 따릅니다. 여기에는 전송을 수행하는 데 필요한 최소 권한만 포함됩니다.
- S3 버킷을 포함한 모든 데이터 스토리지는 저장 데이터를 암호화합니다.
- Amazon Cognito 사용자 풀은 콘솔 및 분산 로드 테스터 API Gateway 엔드포인트에 대한 사용자 액세스를 관리합니다.
- 해당하는 경우 로깅, 추적 및 버전 관리가 활성화됩니다.
- 네트워크 액세스는 기본적으로 프라이빗이며, 사용 가능한 경우 Amazon Virtual Private Cloud(Amazon VPC) 엔드포인트가 켜져 있습니다.

신뢰성

이 섹션에서는 신뢰성 요소의 원칙과 모범 사례를 사용하여 이 솔루션을 설계한 방법을 설명합니다.

- 솔루션은 가능한 경우 AWS Serverless Services(예: Lambda, API Gateway, Amazon S3, AWS Step Functions, Amazon DynamoDB 및 AWS Fargate)를 사용하여 서비스 장애로부터 고가용성과 복구를 보장합니다.
- 모든 컴퓨팅 처리는 Lambda 함수 또는 AWS Fargate의 Amazon ECS를 사용합니다.
- 데이터는 DynamoDB 및 Amazon S3에 저장되므로 기본적으로 여러 가용 영역에 유지됩니다.

성능 효율성

이 섹션에서는 성능 효율성 요소의 원칙과 모범 사례를 사용하여 이 솔루션을 설계한 방법을 설명합니다.

- 이 솔루션은 필요에 따라 수평적으로 확장할 수 있는 서버리스 아키텍처를 사용합니다.
- 이 솔루션은 AWS Lambda, Amazon API Gateway, AWS S3, AWS Step Functions, Amazon DynamoDB DynamoDB, Amazon ECS, AWS Fargate, Amazon Cognito와 같이이 솔루션에서 AWS 서비스를 지원하는 모든 리전에서 시작할 수 있습니다.
- 이 솔루션은 전체적으로 관리형 서비스를 사용하여 리소스 프로비저닝 및 관리의 운영 부담을 줄입니다.
- AWS 서비스가 변경되면 일관성을 유지하기 위해 솔루션이 자동으로 테스트되고 매일 배포됩니다. 또한 솔루션 아키텍트와 주제 전문가가 실험하고 개선할 영역에 대해 검토합니다.

비용 최적화

이 섹션에서는 비용 최적화 요소의 원칙과 모범 사례를 사용하여 이 솔루션을 설계한 방법을 설명합니다.

- 솔루션은 서버리스 아키텍처를 사용하므로 고객은 사용한 만큼만 요금을 청구받습니다.
- Amazon DynamoDB는 온디맨드로 용량을 확장하므로 사용하는 용량에 대해서만 비용을 지불하면 됩니다.
- AWS Fargate의 AWS ECS를 사용하면 선결제 비용 없이 사용하는 컴퓨팅 리소스에 대해서만 비용을 지불할 수 있습니다.

지속 가능성

이 섹션에서는 지속 가능성 요소의 원칙과 모범 사례를 사용하여 이 솔루션을 설계한 방법을 설명합니다.

- 이 솔루션은 관리형 서버리스 서비스를 사용하여 온프레미스 서비스를 지속적으로 운영하는 것에 비해 백엔드 서비스의 환경 영향을 최소화합니다.
- 서버리스 서비스를 사용하면 필요에 따라 스케일 업 또는 스케일 다운할 수 있습니다.

아키텍처 세부 정보

이 섹션에서는 이 솔루션을 구성하는 구성 요소 및 AWS 서비스와 이러한 구성 요소가 함께 작동하는 방식에 대한 아키텍처 세부 정보를 설명합니다.

AWS의 분산 로드 테스트 솔루션은 프런트 엔드와 백엔드라는 두 가지 상위 수준 구성 요소로 구성됩니다.

프런트 엔드

프런트엔드는 로드 테스트 API와 솔루션의 백엔드와 상호 작용하는 데 사용하는 웹 콘솔로 구성됩니다.

로드 테스트 API

AWS의 분산 로드 테스트는 솔루션의 RESTful API를 호스팅하도록 Amazon API Gateway를 구성합니다. 사용자는 포함된 웹 콘솔 및 RESTful API를 통해 테스트 데이터와 안전하게 상호 작용할 수 있습니다. API는 Amazon DynamoDB에 저장된 테스트 데이터에 액세스하기 위한 "정문" 역할을 합니다. APIs를 사용하여 솔루션에 빌드하는 모든 확장 기능에 액세스할 수도 있습니다.

이 솔루션은 Amazon Cognito 사용자 풀의 사용자 인증 기능을 활용합니다. 사용자를 성공적으로 인증한 후 Amazon Cognito는 콘솔이 솔루션의 APIs(Amazon API Gateway 엔드포인트)에 요청을 제출하도록 허용하는 데 사용되는 JSON 웹 토큰을 발급합니다. HTTPS 요청은 토큰이 포함된 권한 부여 헤더와 함께 콘솔에서 APIs로 전송됩니다.

요청에 따라 API Gateway는 적절한 AWS Lambda 함수를 호출하여 DynamoDB 테이블에 저장된 데이터에 필요한 작업을 수행하고, 테스트 시나리오를 Amazon S3에 JSON 객체로 저장하고, Amazon CloudWatch 지표 이미지를 검색하고, 테스트 시나리오를 AWS Step Functions 상태 시스템에 제출합니다.

솔루션의 API에 대한 자세한 내용은이 가이드의 분산 로드 테스트 API 섹션을 참조하세요.

웹 콘솔

이 솔루션에는 테스트를 구성 및 실행하고, 실행 중인 테스트를 모니터링하고, 자세한 테스트 결과를 보는 데 사용할 수 있는 웹 콘솔이 포함되어 있습니다. 콘솔은 Amazon S3에서 호스팅되고 Amazon CloudFront를 통해 액세스하는 ReactJS 애플리케이션입니다. 애플리케이션은 AWS Amplify를 활용하여 Amazon Cognito와 통합하여 사용자를 인증합니다. 또한 웹 콘솔에는 AWS IoT Core에서 해당 주제를 구독하는 실행 중인 테스트의 라이브 데이터를 볼 수 있는 옵션이 포함되어 있습니다.

웹 콘솔은이 로드 테스트 솔루션과 상호 작용하는 방법을 보여주도록 설계되었습니다. 프로덕션 환경에서는 특정 요구 사항에 맞게 웹 콘솔을 사용자 지정하거나 자체 콘솔을 구축하는 것이 좋습니다.

웹 콘솔 URL은 CloudFormation 출력에서 콘솔로 찾을 수 있는 CloudFront 배포 도메인 이름입니다. CloudFormation CloudFormation 템플릿을 시작하면 웹 콘솔 URL과 로그인하기 위한 일회용 암호가 포함된 이메일도 받게 됩니다.

Backend

백엔드는 테스트에 대한 로드를 생성하는 데 사용하는 컨테이너 이미지 파이프라인과 로드 테스트 엔진으로 구성됩니다. 프런트엔드를 통해 백엔드와 상호 작용합니다. 또한 각 테스트에 대해 시작된 AWS Fargate의 Amazon ECS 태스크에는 고유한 테스트 식별자(ID)로 태그가 지정됩니다. 이러한 테스트 ID 태그를 사용하여이 솔루션의 비용을 모니터링할 수 있습니다. 자세한 내용은 AWS Billing and Cost Management 사용 설명서의 사용자 정의 비용 할당 태그를 참조하세요.

컨테이너 이미지 파이프라인

이 솔루션은 AmazonLinux로 빌드된 컨테이너 이미지를 blazemeter 로드 테스트 프레임워크가 설치된 기본 이미지로 활용합니다. 이 이미지는 Amazon Elastic Container Registry(Amazon ECR) 퍼블릭 리포지토리에서 호스팅됩니다. 이미지는 AWS Fargate 클러스터의 Amazon ECS에서 작업을 실행하는 데 사용됩니다.

자세한 내용은이 가이드의 컨테이너 이미지 사용자 지정 섹션을 참조하세요.

인프라 테스트

기본 템플릿 외에도 솔루션은 여러 리전에서 테스트를 실행하는 데 필요한 리소스를 시작하는 보조 템플릿을 생성합니다. 템플릿은 Amazon S3에 저장되며 템플릿에 대한 링크는 웹 콘솔에 제공됩니다. 보조 템플릿은 라이브 데이터를 처리하기 위한 VPC, AWS Fargate 클러스터 및 Lambda 함수를 생성합니다.

보조 리전을 시작하는 방법에 대한 자세한 내용은이 가이드의 다중 리전 배포 섹션을 참조하세요.

로드 테스트 엔진

Distributed Load Testing 솔루션은 Amazon Elastic Container Service(Amazon ECS) 및 AWS Fargate를 사용하여 여러 리전에서 수천 명의 연결된 사용자를 시뮬레이션하여 초당 선택 트랜잭션 수를 생성합니다.

포함된 웹 콘솔을 사용하여 테스트의 일부로 실행할 작업에 대한 파라미터를 정의합니다. 솔루션은 이러한 파라미터를 사용하여 JSON 테스트 시나리오를 생성하고 Amazon S3에 저장합니다.

AWS Step Functions 상태 시스템은 AWS Fargate 클러스터에서 Amazon ECS 작업을 실행하고 모니터링합니다. AWS Step Functions 상태 시스템에는 ecr-checker AWS Lambda 함수, task-status-checker AWS Lambda 함수, task-runner AWS Lambda 함수, task-canceler AWS Lambda 함수 및 결과 구문 분석기 AWS Lambda 함수가 포함되어 있습니다. 워크플로에 대한 자세한 내용은이 가이드의 워크플로 테스트 섹션을 참조하세요. 테스트 결과에 대한 자세한 내용은이 가이드의 테스트 결과 섹션을 참조하세요. 테스트 취소 워크플로에 대한 자세한 내용은이 가이드의 테스트 취소 워크플로 섹션을 참조하세요.

라이브 데이터를 선택하면 솔루션은 해당 리전의 Fargate 작업에 해당하는 CloudWatch 로그를 통해 각 리전에서 real-time-data-publisher Lambda 함수를 시작합니다. 그런 다음 솔루션은 기본 스택을 시작한 리전 내에서 AWS IoT Core의 주제에 데이터를 처리하고 게시합니다. 자세한 내용은이 가이드의 라이브 데이터 섹션을 참조하세요.

이 솔루션의 AWS 서비스

이 솔루션에는 다음 AWS 서비스가 포함되어 있습니다.

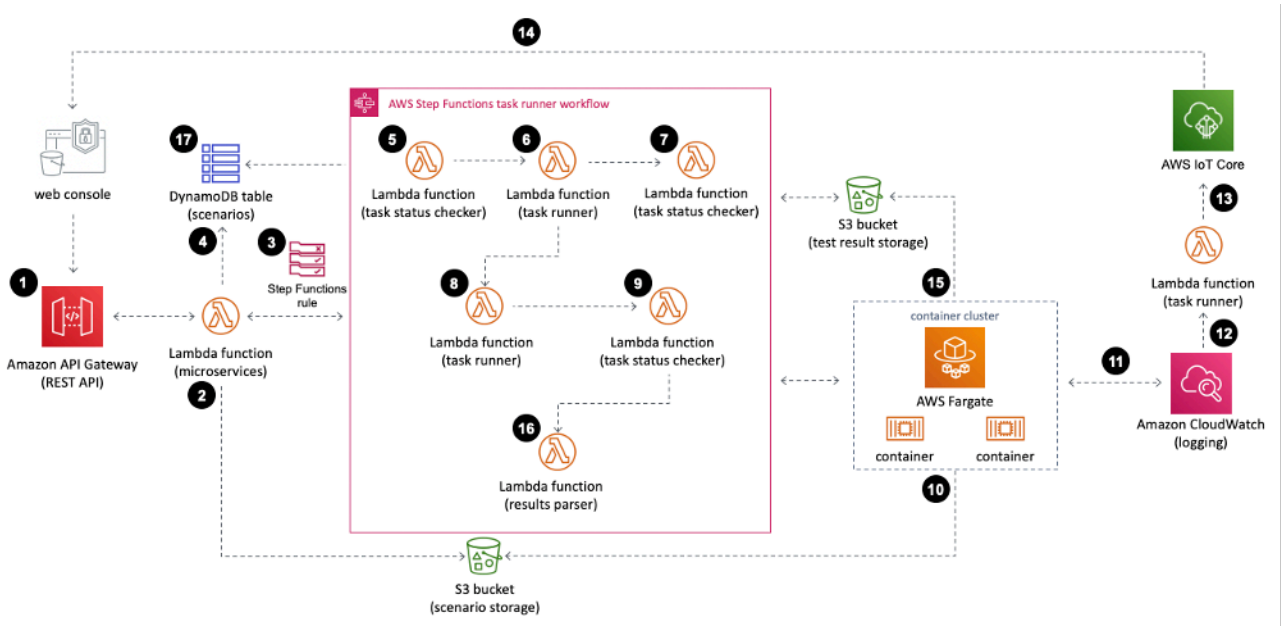
AWS 서비스	설명
Amazon API Gateway	Core. 솔루션에서 REST API 엔드포인트를 호스팅합니다.
CloudFormation	Core. 솔루션 인프라에 대한 배포를 관리합니다.
Amazon CloudFront	Core. Amazon S3에서 호스팅되는 웹 콘텐츠를 제공합니다.
Amazon CloudWatch	Core. 솔루션 로그 및 지표를 저장합니다.
Amazon Cognito	Core. API에 대한 사용자 관리 및 인증을 처리합니다.
Amazon DynamoDB	Core. 배포 정보를 저장하고 시나리오 세부 정보 및 결과를 테스트합니다.
Amazon Elastic Container Service	Core. AWS Fargate 컨테이너에서 독립적인 Amazon ECS 작업을 배포하고 관리합니다.
AWS Fargate	Core. 호스트 솔루션의 Amazon ECS 컨테이너

AWS 서비스	설명
AWS Identity and Access Management	Core. 사용자 역할 및 권한 관리를 처리합니다.
Lambda	Core. APIs 구현, 테스트 결과 구문 분석 및 작업자/리더 작업 시작을 위한 로직을 제공합니다.
AWS Step Functions	Core. 지정된 리전의 AWS Fargate 태스크에서 Amazon ECS 컨테이너 프로비저닝을 오케스트레이션합니다.
Amplify	지원. AWS Amplify로 구동되는 웹 콘솔을 제공합니다.
Amazon CloudWatch Events	지원. 지정된 날짜 또는 반복 날짜에 자동으로 시작되도록 테스트를 예약합니다.
Amazon Elastic 컨테이너 레지스트리	지원. 컨테이너 이미지를 퍼블릭 ECR 리포지토리에 호스팅합니다.
AWS IoT Core	지원. AWS IoT Core에서 해당 주제를 구독하여 실행 중인 테스트에 대한 라이브 데이터를 볼 수 있습니다.
AWS Systems Manager	지원. 리소스 운영 및 비용 데이터에 대한 애플리케이션 수준의 리소스 모니터링 및 시각화를 제공합니다.
Amazon S3	지원. 정적 웹 콘텐츠, 로그, 지표 및 테스트 데이터를 호스팅합니다.
Amazon Virtual Private Cloud	지원. AWS Fargate에서 실행되는 솔루션의 Amazon ECS 컨테이너를 포함합니다.

AWS의 분산 로드 테스트 작동 방식

다음 세부 분석에는 테스트 시나리오 실행과 관련된 단계가 나와 있습니다.

테스트 워크플로



1. 웹 콘솔을 사용하여 구성 세부 정보가 포함된 테스트 시나리오를 솔루션의 API에 제출합니다.
2. 테스트 시나리오 구성은 Amazon Simple Storage Service(Amazon S3)에 JSON 파일()로 업로드됩니다`s3://<bucket-name>/test-scenarios/<$TEST_ID>/<$TEST_ID>.json`.
3. AWS Step Functions 상태 시스템은 테스트 ID, 작업 수, 테스트 유형 및 파일 유형을 AWS Step Functions 상태 시스템 입력으로 사용하여 실행됩니다. 테스트가 예약된 경우 먼저 지정된 날짜에 AWS Step Functions를 트리거하는 CloudWatch Events 규칙을 생성합니다. 일정 워크플로에 대한 자세한 내용은이 가이드의 일정 워크플로 테스트 섹션을 참조하세요.
4. 구성 세부 정보는 Amazon DynamoDB 테이블 시나리오에 저장됩니다.
5. AWS Step Functions 작업 실행기 워크플로에서 task-status-checker AWS Lambda 함수는 Amazon Elastic Container Service(Amazon ECS) 작업이 동일한 테스트 ID에 대해 이미 실행 중인지 확인합니다. 동일한 테스트 ID를 가진 작업이 실행 중인 것으로 확인되면 오류가 발생합니다. AWS Fargate 클러스터에서 실행 중인 Amazon ECS 작업이 없는 경우 함수는 테스트 ID, 작업 수 및 테스트 유형을 반환합니다.
6. 작업 실행기 AWS Lambda 함수는 이전 단계에서 작업 세부 정보를 가져오고 AWS Fargate 클러스터에서 Amazon ECS 작업자 작업을 실행합니다. Amazon ECS API는 RunTask 작업을 사용하여 작업자 작업을 실행합니다. 이러한 작업자 작업이 시작된 다음 테스트를 시작하기 위해 리더 작업의 시작 메시지를 기다립니다. RunTask 작업은 정의당 10개의 작업으로 제한됩니다. 작업 수가 10개를 초과하는 경우 모든 작업자 작업이 시작될 때까지 작업 정의가 여러 번 실행됩니다. 또한 함수는 결과 구문 분석 AWS Lambda 함수에서 현재 테스트를 구분하는 접두사를 생성합니다.
7. task-status-checker AWS Lambda 함수는 모든 Amazon ECS 작업자 작업이 동일한 테스트 ID로 실행 중인지 확인합니다. 태스크가 여전히 프로비저닝 중인 경우 1분 동안 기다렸다가 다시 확인합니

다. 모든 Amazon ECS 작업이 실행되면 테스트 ID, 작업 수, 테스트 유형, 모든 작업 IDs 및 접두사를 반환하여 작업 실행기 함수에 전달합니다.

8. Task-runner AWS Lambda 함수가 다시 실행되며, 이번에는 리더 노드 역할을 하는 단일 Amazon ECS 작업을 시작합니다. 이 ECS 태스크는 테스트를 동시에 시작하기 위해 각 작업자 태스크에 테스트 시작 메시지를 보냅니다.
9. task-status-checker AWS Lambda 함수는 Amazon ECS 작업이 동일한 테스트 ID로 실행 중인지 다시 확인합니다. 태스크가 계속 실행 중인 경우 1분 동안 기다렸다가 다시 확인합니다. 실행 중인 Amazon ECS 작업이 없으면 테스트 ID, 작업 수, 테스트 유형 및 접두사를 반환합니다.
10. 작업 실행기 AWS Lambda 함수가 AWS Fargate 클러스터에서 Amazon ECS 작업을 실행하면 각 작업은 Amazon S3에서 테스트 구성을 다운로드하고 테스트를 시작합니다.
11. 테스트가 실행되면 평균 응답 시간, 동시 사용자 수, 성공한 요청 수, 각 작업에 대한 실패한 요청 수가 Amazon CloudWatch에 로깅되고 CloudWatch 대시보드에서 볼 수 있습니다.
12. 테스트에 라이브 데이터를 포함시킨 경우 솔루션은 구독 필터를 사용하여 CloudWatch에서 실시간 테스트 결과를 필터링합니다. 그런 다음 솔루션은 Lambda 함수에 데이터를 전달합니다.
13. 그런 다음 Lambda 함수는 수신된 데이터를 구조화하여 AWS IoT Core 주제에 게시합니다.
14. 웹 콘솔은 테스트를 위해 AWS IoT Core 주제를 구독하고 주제에 게시된 데이터를 수신하여 테스트가 실행되는 동안 실시간 데이터를 그래프로 표시합니다.
15. 테스트가 완료되면 컨테이너 이미지는 세부 보고서를 XML 파일로 Amazon S3로 내보냅니다. 각 파일에는 파일 이름에 대한 UUID가 부여됩니다. 예: s3://dlte-bucket/test-scenarios/<$TEST_ID>/results/<$UUID>.json.
16. XML 파일이 Amazon S3에 업로드되면 결과 구문 분석기 AWS Lambda 함수는 접두사로 시작하는 XML 파일의 결과를 읽고 모든 결과를 구문 분석하여 하나의 요약된 결과로 집계합니다.
17. 결과 구문 분석기 AWS Lambda 함수는 집계 결과를 Amazon DynamoDB 테이블에 기록합니다.

설계 고려 사항

지원되는 애플리케이션

이 솔루션은 AWS 계정에서 애플리케이션으로 네트워크 연결이 있는 한 클라우드 기반 애플리케이션과 온프레미스 애플리케이션을 지원합니다. 솔루션은 HTTP 또는 HTTPSAPIs를 지원합니다. 또한 HTTP 요청 헤더를 제어할 수 있으므로 권한 부여 또는 사용자 지정 헤더를 추가하여 토큰 또는 API 키를 전달할 수 있습니다.

JMeter 스크립트 지원

이 솔루션의 사용자 인터페이스(UI)를 사용하여 테스트 시나리오를 생성할 때 JMeter 테스트 스크립트를 사용할 수 있습니다. JMeter 스크립트 파일을 선택하면 <stack-name>-scenariosbucket Amazon Simple Storage Service(Amazon S3) 버킷에 업로드됩니다. Amazon Elastic Container Service(Amazon ECS) 태스크가 실행 중이면 JMeter 스크립트가 <stack-name>-scenariosbucket Amazon S3 버킷에서 다운로드되고 테스트가 실행됩니다.

JMeter 입력 파일이 있는 경우 JMeter 스크립트와 함께 입력 파일을 압축할 수 있습니다. 테스트 시나리오를 생성할 때 zip 파일을 선택할 수 있습니다.

플러그인을 포함하려는 경우 번들 zip 파일의/플러그 하위 디렉터리에 포함된 모든 .jar 파일이 JMeter 확장 디렉터리에 복사되고 로드 테스트에 사용할 수 있습니다.

> **Note**
>
> JMeter 스크립트 파일에 JMeter 입력 파일을 포함하는 경우 JMeter 스크립트 파일에 입력 파일의 상대 경로를 포함해야 합니다. 또한 입력 파일은 상대 경로에 있어야 합니다. 예를 들어 JMeter 입력 파일과 스크립트 파일이 /home/user 디렉터리에 있고 JMeter 스크립트 파일에서 입력 파일을 참조하는 경우 입력 파일의 경로는 이어야 합니다.INPUT_FILES. 대신 /home/user/INPUT_FILES를 사용하면 입력 파일을 찾을 수 없으므로 테스트가 실패합니다.

JMeter 플러그인을 포함하는 경우 zip 파일의 루트 내에서 .jar 파일을 /plugins라는 하위 디렉터리에 번들링해야 합니다. zip 파일의 루트를 기준으로 jar 파일의 경로는 ./plugins/BUNDLED_PLUGIN.jar여야 합니다.

JMeter 스크립트를 사용하는 방법에 대한 자세한 내용은 JMeter 사용 설명서를 참조하세요.

테스트 예약

나중에 실행할 테스트를 예약하거나 지금 실행 옵션을 사용할 수 있습니다. 테스트를 향후 일회성 실행으로 예약하거나 첫 번째 실행 날짜와 계획된 반복을 지정하는 반복 테스트를 설정할 수 있습니다. 반복 옵션에는 일별, 주별, 격주 및 월별이 포함됩니다. 예약의 작동 방식에 대한 자세한 내용은이 가이드의 예약 테스트 워크플로 섹션을 참조하세요.

버전 3.3.0부터 AWS의 분산 로드 테스트를 통해 사용자는 cron 표현식을 사용하여 로드 테스트를 예약할 수 있습니다. 일정 실행을 선택한 다음 CRON 탭을 선택하여 수동으로 cron 값을 입력하거나 드

롭다운 필드를 사용합니다. cronExpiryDate는 예약된 테스트 실행 날짜와 일치해야 합니다. 다음 실행 날짜(UTC)를 검토하여 일정을 확인합니다.

> **Note**
>
> - 테스트 기간: 예약 시 총 테스트 기간을 고려합니다. 예를 들어, 램프 업 시간이 10분이고 대기 시간이 40분인 테스트는 완료하는 데 약 80분이 걸립니다.
> - 최소 간격: 예약된 테스트 사이의 간격이 예상 테스트 기간보다 긴지 확인합니다. 예를 들어 테스트에 약 80분이 걸리는 경우 3시간마다 실행되도록 예약합니다.
> - 시간당 제한: 예상 테스트 기간이 1시간 미만인 경우에도 시스템에서는 1시간 차이로만 테스트를 예약할 수 없습니다.

동시 테스트

이 솔루션에는 각 테스트에 대한 Amazon CloudWatch 대시보드가 포함되어 있으며 Amazon ECS 클러스터에서 해당 테스트에 대해 실행 중인 모든 작업의 결합된 출력을 실시간으로 표시합니다. CloudWatch 대시보드에는 평균 응답 시간, 동시 사용자 수, 성공한 요청 수, 실패한 요청 수가 표시됩니다. 각 지표는 초 단위로 집계되며 대시보드는 1분마다 업데이트됩니다.

사용자 관리

초기 구성 중에 Amazon Cognito가 솔루션의 웹 콘솔에 대한 액세스 권한을 부여하는 데 사용하는 사용자 이름과 이메일 주소를 제공합니다. 콘솔은 사용자 관리를 제공하지 않습니다. 사용자를 추가하려면 Amazon Cognito 콘솔을 사용해야 합니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 사용자 풀에서 사용자 관리를 참조하세요.

리전 배포

이 솔루션은 특정 AWS 리전에서만 사용할 수 있는 Amazon Cognito를 사용합니다. 따라서 Amazon Cognito를 사용할 수 있는 리전에이 솔루션을 배포해야 합니다. 리전별 최신 서비스 가용성은 AWS 리전 서비스 목록을 참조하세요.

배포 계획

이 섹션에서는 솔루션을 배포하기 전에 발생하는 비용, 보안, 리전 및 기타 고려 사항에 대해 설명합니다.

비용

이 솔루션을 실행하는 동안 사용되는 AWS 서비스의 비용은 사용자의 책임입니다. 이 솔루션을 실행하는 데 드는 총 비용은 실행된 로드 테스트 수, 해당 로드 테스트 기간, 테스트의 일부로 사용되는 데이터의 양에 따라 달라집니다. 이 개정부터 미국 동부(버지니아 북부) 리전의 기본 설정으로이 솔루션을 실행하는 데 드는 비용은 매월 약 30.90 USD입니다.

다음 표는 1개월 동안 미국 동부(버지니아 북부) 리전의 기본 파라미터를 사용하여이 솔루션을 배포하기 위한 샘플 비용 분석을 제공합니다.

AWS 서비스	Dimensions	비용[USD]
AWS Fargate	30시간 동안 실행되는 온디맨드 작업 10개(vCPUs개와 4GB 메모리 사용)	29.62 USD
Amazon DynamoDB	온디맨드 쓰기 용량 단위 1,000개 온디맨드 읽기 용량 단위 1,000개	$0.0015
AWS Lambda	요청 1,000개 총 기간 10분	1.25 USD
Step Functions	1,000개의 상태 전환	0.025 USD
합계:		매월 30.90 USD

비용 관리에 도움이 되도록 AWS Cost Explorer를 통해 예산을 생성하는 것이 좋습니다. 요금은 변경될 수 있습니다. 자세한 내용은 이 솔루션에 사용되는 각 AWS 서비스의 요금 웹 페이지를 참조하세요.

> **Important**
>
> 버전 1.3.0부터 CPU는 vCPU 2개로 증가하고 메모리는 4GB로 증가합니다. 이러한 변경으로 인해이 솔루션의 이전 버전에 비해 예상 비용이 증가합니다. 로드 테스트에서 이러한 AWS 리소스 증가를 요구하지 않는 경우 리소스를 줄일 수 있습니다. 자세한 내용은이 가이드의 컨테이너 리소스 증가 섹션을 참조하세요.

> **Note**
>
> 이 솔루션은 테스트를 실행할 때 라이브 데이터를 포함하는 옵션을 제공합니다. 이 기능을 사용하려면 추가 비용이 발생하는 추가 AWS Lambda 함수와 AWS IoT Core 주제가 필요합니다.

요금은 변경될 수 있습니다. 자세한 내용은이 솔루션에서 사용할 각 AWS 서비스의 요금 웹 페이지를 참조하세요.

보안

AWS 인프라에 시스템을 구축하면 사용자와 AWS 간에 보안 책임이 공유됩니다. AWS는 호스트 운영 체제, 가상화 계층 및 서비스가 운영되는 시설의 물리적 보안을 포함한 구성 요소를 운영, 관리 및 제어하므로이 공동 책임 모델은 운영 부담을 줄입니다. AWS 보안에 대한 자세한 내용은 AWS Cloud Security를 참조하십시오.

IAM 역할

AWS Identity and Access Management(IAM) 역할을 통해 고객은 AWS 클라우드의 서비스 및 사용자에게 세분화된 액세스 정책 및 권한을 할당할 수 있습니다. 이 솔루션은 솔루션의 AWS Lambda 함수에 리전 리소스를 생성할 수 있는 액세스 권한을 부여하는 IAM 역할을 생성합니다.

Amazon CloudFront

이 솔루션은 Amazon Simple Storage Service(Amazon S3) 버킷에 호스팅된 웹 콘솔을 배포합니다. 지연 시간을 줄이고 보안을 개선하기 위해이 솔루션에는 솔루션의 웹 사이트 버킷 콘텐츠에 대한 퍼블릭 액세스를 제공하는 Amazon CloudFront CloudFront 배포가 포함됩니다. 자세한 내용을 알아보려면 Amazon CloudFront 개발자 안내서의 오리진 액세스 ID(OAI)를 사용하여 Amazon S3 콘텐츠에 대한 액세스 제한을 참조하세요.

AWS Fargate 보안 그룹

기본적으로이 솔루션은 AWS Fargate 보안 그룹의 아웃바운드 규칙을 퍼블릭에 엽니다. AWS Fargate가 모든 곳으로 트래픽을 전송하는 것을 차단하려면 아웃바운드 규칙을 특정 Classless Inter-Domain Routing(CIDR)으로 변경합니다.

또한이 보안 그룹에는 포트 50,000의 로컬 트래픽을 동일한 보안 그룹에 속하는 모든 소스로 허용하는 인바운드 규칙이 포함되어 있습니다. 이는 컨테이너가 서로 통신할 수 있도록 하는 데 사용됩니다.

네트워크 스트레스 테스트

네트워크 스트레스 테스트 정책에 따라이 솔루션을 사용할 책임은 사용자에게 있습니다. 이 정책은 Amazon EC2 인스턴스에서 다른 Amazon EC2 인스턴스Amazon EC2, AWS 속성/서비스 또는 외부 엔드포인트와 같은 다른 위치로 직접 대량 네트워크 테스트를 실행하려는 경우와 같은 상황을 다룹니다. 이러한 테스트를 스트레스 테스트, 로드 테스트 또는 게임데이 테스트라고도 합니다. 대부분의 고객 테스트는이 정책에 해당되지 않지만, 1분 이상, 1Gbps(초당 10억 비트) 또는 1Gpps(초당 10억 패킷)를 초과하는 트래픽을 총 1분 이상 지속할 것으로 생각되면이 정책을 참조하세요.

퍼블릭 사용자 인터페이스에 대한 액세스 제한

IAM 및 Amazon Cognito에서 제공하는 인증 및 권한 부여 메커니즘을 넘어 퍼블릭 사용자 인터페이스에 대한 액세스를 제한하려면 AWS WAF(웹 애플리케이션 방화벽) 보안 자동화 솔루션을 사용합니다.

이 솔루션은 일반적인 웹 기반 공격을 필터링하는 AWS WAF 규칙 세트를 자동으로 배포합니다. 사용자는 AWS WAF 웹 액세스 제어 목록(웹 ACL)에 포함된 규칙을 정의하는 사전 구성된 보호 기능 중에서 선택할 수 있습니다.

지원되는 AWS 리전

이 솔루션은 현재 일부 AWS 리전에서 사용할 수 없는 Amazon Cognito 서비스를 사용합니다. 리전별 AWS 서비스의 최신 가용성은 AWS 리전 서비스 목록을 참조하세요.

AWS의 분산 로드 테스트는 다음 AWS 리전에서 사용할 수 있습니다.

리전 이름	
미국 동부(오하이오)	아시아 태평양(도쿄)
미국 동부(버지니아 북부)	캐나다(중부)

리전 이름	
미국 서부(캘리포니아 북부)	유럽(프랑크푸르트)
미국 서부(오리건)	유럽(아일랜드)
아시아 태평양(뭄바이)	유럽(런던)
아시아 태평양(서울)	유럽(파리)
아시아 태평양(싱가포르)	유럽(스톡홀름)
아시아 태평양(시드니)	남아메리카(상파울루)

할당량

서비스 할당량(제한이라고도 함)은 AWS 계정의 최대 서비스 리소스 또는 작업 수입니다.

이 솔루션의 AWS 서비스에 대한 할당량

이 솔루션에 구현된 각 서비스의 할당량이 충분한지 확인하세요. 자세한 내용은 AWS 서비스 할당량을 참조하세요.

다음 링크를 사용하여 해당 서비스의 페이지로 이동합니다. 페이지를 전환하지 않고 설명서의 모든 AWS 서비스에 대한 서비스 할당량을 보려면 대신 PDF의 서비스 엔드포인트 및 할당량 페이지에서 정보를 확인합니다.

AWS CloudFormation 할당량

AWS 계정에는이 솔루션에서 스택을 시작할 때 알아야 할 AWS CloudFormation 할당량이 있습니다. 이러한 할당량을 이해하면 이 솔루션을 성공적으로 배포하지 못하는 제한 오류를 방지할 수 있습니다. 자세한 내용은 AWS CloudFormation 사용 설명서의에서 AWS CloudFormation 할당량을 참조하세요 AWS CloudFormation.

로드 테스트 할당량

AWS Fargate 시작 유형을 사용하여 Amazon ECS에서 실행할 수 있는 최대 태스크 수는 태스크의 vCPU 크기를 기반으로 합니다. AWS의 분산 로드 테스트의 기본 태스크 크기는 vCPU 2개입니다. 현재 기본 할당량을 보려면 Amazon ECS 서비스 할당량을 참조하세요. 현재 계정 할당량은 나열된 할당

량과 다를 수 있습니다. 계정별 할당량을 확인하려면 AWS Management Console에서 Fargate 온디맨드 vCPU 리소스 수에 대한 서비스 할당량을 확인합니다. 증가를 요청하는 방법에 대한 지침은 AWS 일반 참조 안내서의 AWS 서비스 할당량을 참조하세요.

AmazonLinux 이미지(Blazemeter가 설치된 경우) 컨테이너 이미지는 작업당 동시 연결을 제한하지 않지만 무제한 사용자 수를 지원할 수 있음을 의미하지는 않습니다. 컨테이너가 테스트를 위해 생성할 수 있는 동시 사용자 수를 확인하려면이 가이드의 사용자 수 결정 섹션을 참조하세요.

> **Note**
>
> 기본 설정을 기반으로 한 동시 사용자의 권장 제한은 200명입니다.

동시 테스트

이 솔루션에는 각 테스트에 대한 Amazon CloudWatch 대시보드가 포함되어 있으며 Amazon ECS 클러스터에서 해당 테스트에 대해 실행 중인 모든 작업의 결합된 출력을 실시간으로 표시합니다. CloudWatch 대시보드에는 평균 응답 시간, 동시 사용자 수, 성공한 요청 수, 실패한 요청 수가 표시됩니다. 각 지표는 초 단위로 집계되며 대시보드는 1분마다 업데이트됩니다.

Amazon EC2 테스트 정책

네트워크 트래픽이 1Gbps 미만으로 유지되는 한이 솔루션을 사용하여 로드 테스트를 실행하기 위해 AWS의 승인이 필요하지 않습니다. 테스트에서 1Gbps 이상이 생성되면 AWS에 문의하십시오. 자세한 내용은 Amazon EC2 테스트 정책을 참조하세요.

Amazon CloudFront 로드 테스트 정책

CloudFront 엔드포인트를 로드 테스트하려는 경우 Amazon CloudFront 개발자 안내서의 로드 테스트 지침을 참조하세요. 또한 트래픽을 여러 작업 및 리전에 분산하는 것이 좋습니다. 로드 테스트에 최소 30분의 램프 업 시간을 제공합니다. 초당 500,000개 이상의 요청을 보내거나 300Gbps 이상의 데이터를 요구하는 로드 테스트의 경우 먼저 트래픽 전송에 대한 사전 승인을 받는 것이 좋습니다. CloudFront는 CloudFront 서비스 가용성에 영향을 미치는 승인되지 않은 로드 테스트 트래픽을 제한할 수 있습니다.

솔루션 배포

이 솔루션은 AWS CloudFormation 템플릿 및 스택을 사용하여 배포를 자동화합니다. CloudFormation 템플릿은이 솔루션에 포함된 AWS 리소스와 해당 속성을 지정합니다. CloudFormation 스택은 템플릿에 설명된 리소스를 프로비저닝합니다.

배포 프로세스 개요

이 섹션의 step-by-step 지침에 따라 솔루션을 구성하고 계정에 배포합니다.

솔루션을 시작하기 전에이 가이드의 앞부분에서 설명한 비용, 아키텍처, 네트워크 보안 및 기타 고려 사항을 검토하세요.

배포에 소요되는 시간: 약 15분

AWS CloudFormation 템플릿

이 솔루션을 배포하기 전에이 솔루션의 CloudFormation 템플릿을 다운로드할 수 있습니다. 이 솔루션은 AWS CloudFormation을 사용하여 AWS에서 분산 로드 테스트의 배포를 자동화합니다. 여기에는 배포 전에 다운로드할 수 있는 다음 AWS CloudFormation 템플릿이 포함되어 있습니다.

View template

distributed-load-testing-on-aws.template -이 템플릿을 사용하여 솔루션 및 모든 관련 구성 요소를 시작합니다. 기본 구성은 이 솔루션 섹션의 AWS 서비스에 있는 코어 및 지원 서비스를 배포하지만 특정 요구 사항에 맞게 템플릿을 사용자 지정할 수 있습니다.

> **Note**
>
> AWS CloudFormation 리소스는 AWS Cloud Development Kit(AWS CDK) 구문에서 생성됩니다. 이전에이 솔루션을 배포한 경우 업데이트 지침은 솔루션 업데이트를 참조하세요.

스택 시작

> **Important**
>
> v3.2.6 이전 버전에서 최신 버전으로 스택을 업데이트하는 경우 스택을 업데이트하기 전에 이 섹션을 읽어보세요.

자동 배포를 시작하기 전에이 가이드에서 설명하는 아키텍처 및 기타 고려 사항을 검토하세요. 이 섹션의 step-by-step 지침에 따라 AWS에서 분산 로드 테스트를 구성하고 계정에 배포합니다.

배포에 소요되는 시간: 약 15분

> **Important**
>
> 이 솔루션에는 익명화된 운영 지표를 AWS로 전송하는 옵션이 포함되어 있습니다. 당사는 이 데이터를 사용하여 고객이 이 솔루션과 관련 서비스 및 제품을 어떻게 사용하는지 더 잘 이해합니다. AWS는이 설문 조사를 통해 수집된 데이터를 소유합니다. 데이터 수집에는 AWS 개인 정보 보호 고지가 적용됩니다.
>
> 이 기능을 옵트아웃하려면 템플릿을 다운로드하고 AWS CloudFormation 매핑 섹션을 수정한 다음 AWS CloudFormation 콘솔을 사용하여 업데이트된 템플릿을 업로드하고 솔루션을 배포합니다. 자세한 내용은 이 안내서의 익명화된 데이터 수집 섹션을 참조하세요.

이 자동화된 AWS CloudFormation 템플릿은 AWS에 분산 로드 테스트를 배포합니다.

> **Note**
>
> 이 솔루션을 실행하는 동안 사용되는 AWS 서비스의 비용은 사용자가 부담합니다. 자세한 내용은이 가이드의 비용 섹션을 방문하여이 솔루션에 사용되는 각 AWS 서비스의 요금 웹 페이지를 참조하세요.

1. AWS Management Console에 로그인하고 아래 버튼을 선택하여 distributed-load-testing-on-aws AWS CloudFormation 템플릿을 시작합니다.

또는 템플릿을 자체 구현의 시작점으로 다운로드할 수도 있습니다.

2. 이 템플릿은 기본적으로 미국 동부(버지니아 북부) 리전에서 실행됩니다. 다른 AWS 리전에서이 솔루션을 시작하려면 콘솔 탐색 모음에서 리전 선택기를 사용합니다.

> **Note**
>
> 이 솔루션은 현재 특정 AWS 리전에서만 사용할 수 있는 Amazon Cognito를 사용합니다. 따라서 Amazon Cognito를 사용할 수 있는 AWS 리전에서이 솔루션을 시작해야 합니다. 리전별 최신 서비스 가용성은 AWS 리전 서비스 목록을 참조하세요.

3. 스택 생성 페이지에서 Amazon S3 URL 텍스트 상자에 올바른 템플릿 URL이 표시되는지 확인하고 다음을 선택합니다.
4. 스택 세부 정보 지정 페이지에서 솔루션 스택 이름을 할당합니다.
5. 파라미터에서 템플릿의 파라미터를 검토하고 필요에 따라 수정합니다. 이 솔루션은 다음과 같은 기본값을 사용합니다.

파라미터	Default	설명
관리자 이름	<입력 필수>	초기 솔루션 관리자의 사용자 이름입니다.
관리자 이메일	<## ##>	관리자 사용자의 이메일 주소입니다. 시작 후 콘솔 로그인 지침이 포함된 이메일이이 주소로 전송됩니다.
기존 VPC ID	<선택 사항 입력>	사용하려는 VPC가 있고 이미 생성된 경우 스택이 배포된 리전과 동일한 리전에 있는 기존 VPC의 ID를 입력합니다. 예: vpc-1a2b3c4d5e6f.
첫 번째 기존 서브넷	<선택 사항 입력>	기존 VPC 내 첫 번째 서브넷의 ID입니다. 이 서브넷은 테스트를 실행하기 위해 컨테이너 이미지를 가져오려면 인터

파라미터	Default	설명
		넷으로의 경로가 필요합니다. 예: subnet-7h8i9j0k.
두 번째 기존 서브넷	<선택 사항 입력>	기존 VPC 내 두 번째 서브넷의 ID입니다. 이 서브넷은 테스트를 실행하기 위해 컨테이너 이미지를 가져오려면 인터넷으로의 경로가 필요합니다. 예를 들어 subnet-1x2y3z입니다.
AWS Fargate VPC CIDR 블록	192.168.0.0/16	기존 VPC에 값을 제공하지 않으면 솔루션 생성 Amazon VPC의 CIDR 블록에 AWS Fargate의 IP 주소가 포함됩니다.
AWS Fargate 서브넷 A CIDR 블록	192.168.0.0/20	기존 VPC에 값을 제공하지 않으면 CIDR 블록에 Amazon VPC 서브넷 A의 IP 주소가 포함됩니다.
AWS Fargate 서브넷 B CIDR 블록	192.168.16.0/20	기존 VPC에 값을 제공하지 않으면 CIDR 블록에 Amazon VPC 서브넷 B의 IP 주소가 포함됩니다.
AWS Fargate 보안 그룹 CIDR 블록	0.0.0.0/0	Amazon ECS 컨테이너 아웃바운드 액세스를 제한하는 CIDR 블록입니다.

6. 다음을 선택합니다.
7. Configure stack options(스택 옵션 구성) 페이지에서 Next(다음)를 선택합니다.
8. 검토 페이지에서 설정을 검토하고 확인합니다. 템플릿이 AWS Identity and Access Management(IAM) 리소스를 생성할 것임을 확인하는 확인란을 선택합니다.
9. [스택 생성(Create stack)]을 선택하여 스택을 배포합니다.

AWS CloudFormation 콘솔의 상태 열에서 스택의 상태를 볼 수 있습니다. 약 15분 후에 CREATE_COMPLETE 상태가 수신됩니다.

> **Note**
>
> 기본 AWS Lambda 함수 외에도이 솔루션에는 초기 구성 중에 또는 리소스가 업데이트되거나 삭제될 때만 실행되는 사용자 지정 리소스 Lambda 함수가 포함되어 있습니다.

이 솔루션을 실행하면 사용자 지정 리소스 Lambda 함수가 비활성화됩니다. 그러나 연결된 리소스를 관리하는 데 필요하므로이 함수를 삭제하지 마십시오.

다중 리전 배포

배포에 소요되는 시간: 약 5분

여러 리전에서 테스트를 실행할 수 있습니다. Distributed Load Testing 솔루션을 배포하면 Amazon S3 버킷 3개가 생성됩니다. 이 솔루션은 보조 리전 스택을 생성하여 Amazon S3 시나리오 버킷에 저장합니다.

> **Note**
>
> 버킷 이름 지정 규칙은 *<stack-name>* -`dlttestrunnerstoragedltscenariosbucket이며, 버킷 이름의 <_[0-9][0-9]..-<[0-9][0-9].._ 키워드 시나리오는 S3 콘솔로 이동한 다음 버킷으로 이동하여 찾을 수 있습니다.

다중 리전 배포를 실행하려면 테스트를 실행하려는 리전의 Amazon S3 시나리오 버킷에 저장된 리전 CloudFormation 템플릿을 배포해야 합니다. 다음을 수행하여 리전 템플릿을 설치할 수 있습니다.

1. 솔루션의 웹 콘솔에서 상단 메뉴의 리전 관리로 이동합니다.
2. 클립보드 아이콘을 사용하여 Amazon S3에서 CloudFormation 템플릿 링크를 복사합니다.
3. AWS CloudFormation 콘솔에 로그인하고 올바른 리전을 선택합니다.
4. 스택 생성 페이지에서 Amazon S3 URL 텍스트 상자에 올바른 템플릿 URL이 표시되는지 확인하고 다음을 선택합니다.
5. 스택 세부 정보 지정 페이지에서 솔루션 스택 이름을 할당합니다.
6. 파라미터에서 템플릿의 파라미터를 검토하고 필요에 따라 수정합니다. 이 솔루션은 다음과 같은 기본값을 사용합니다.

파라미터	Default	설명
기존 VPC ID	<선택 사항 입력>	사용하려는 VPC가 있고 이미 생성된 경우 스택이 배포된 리전과 동일한 리전에 있는 기존 VPC의 ID를 입력합니다. 예: vpc-1a2b3c4d5e6f.
첫 번째 기존 서브넷	<선택 사항 입력>	기존 VPC 내 첫 번째 서브넷의 ID입니다. 이 서브넷은 테스트를 실행하기 위해 컨테이너 이미지를 가져오려면 인터넷으로의 경로가 필요합니다. 예: subnet-7h8i9j0k.
두 번째 기존 서브넷	<선택 사항 입력>	기존 VPC 내 두 번째 서브넷의 ID입니다. 이 서브넷은 테스트를 실행하기 위해 컨테이너 이미지를 가져오려면 인터넷으로의 경로가 필요합니다. 예를 들어 subnet-1x2y3z입니다.
AWS Fargate VPC CIDR 블록	192.168.0.0/16	기존 VPC에 값을 제공하지 않으면 솔루션 생성 Amazon VPC의 CIDR 블록에 AWS Fargate의 IP 주소가 포함됩니다.
AWS Fargate 서브넷 A CIDR 블록	192.168.0.0/20	기존 VPC에 값을 제공하지 않으면 CIDR 블록에 Amazon VPC 서브넷 A의 IP 주소가 포함됩니다.

파라미터	Default	설명
AWS Fargate 서브넷 B CIDR 블록	192.168.16.0/20	기존 VPC에 값을 제공하지 않으면 CIDR 블록에 Amazon VPC 서브넷 B의 IP 주소가 포함됩니다.
AWS Fargate 보안 그룹 CIDR 블록	0.0.0.0/0	Amazon ECS 컨테이너 아웃바운드 액세스를 제한하는 CIDR 블록입니다.

7. 다음을 선택합니다.
8. Configure stack options(스택 옵션 구성) 페이지에서 Next(다음)를 선택합니다.
9. 검토 페이지에서 설정을 검토하고 확인합니다. 템플릿이 AWS Identity and Access Management(IAM) 리소스를 생성할 것임을 확인하는 확인란을 선택해야 합니다.
10. [스택 생성(Create stack)]을 선택하여 스택을 배포합니다.

    AWS CloudFormation 콘솔의 상태 열에서 스택의 상태를 볼 수 있습니다. 약 5분 후에 CREATE_COMPLETE 상태를 받게 됩니다.

리전이 성공적으로 배포되면 웹 콘솔에 표시됩니다. 테스트를 생성하면 새 리전이 리전 관리 모달에 나열됩니다. 테스트 생성 시 선택하여 테스트에서이 리전을 사용할 수 있습니다. 이 솔루션은 시나리오 테이블에서 시작된 각 리전에 대해 DynamoDB 항목을 생성합니다. 여기에는 해당 리전의 테스트 리소스와 관련된 필수 정보가 포함되어 있습니다. 웹 콘솔에서 리전별로 테스트 결과를 정렬할 수 있습니다. API 제약으로 인해 Amazon CloudWatch 지표에 그래프를 작성하여 다중 리전 테스트에서 모든 리전의 집계 결과만 볼 수 있습니다. 테스트가 완료되면 테스트 결과에서 그래프의 소스 코드를 찾을 수 있습니다.

> **Note**
>
> 웹 콘솔 없이 리전 스택을 시작할 수 있습니다. Amazon S3 시나리오 버킷의 리전 템플릿 링크를 가져와서 필요한 리전에서 리전 스택을 시작할 때 소스로 제공합니다. 또는 템플릿을 다운로드하여 원하는 리전의 소스로 업로드할 수 있습니다.

Service Catalog AppRegistry를 사용하여 솔루션 모니터링

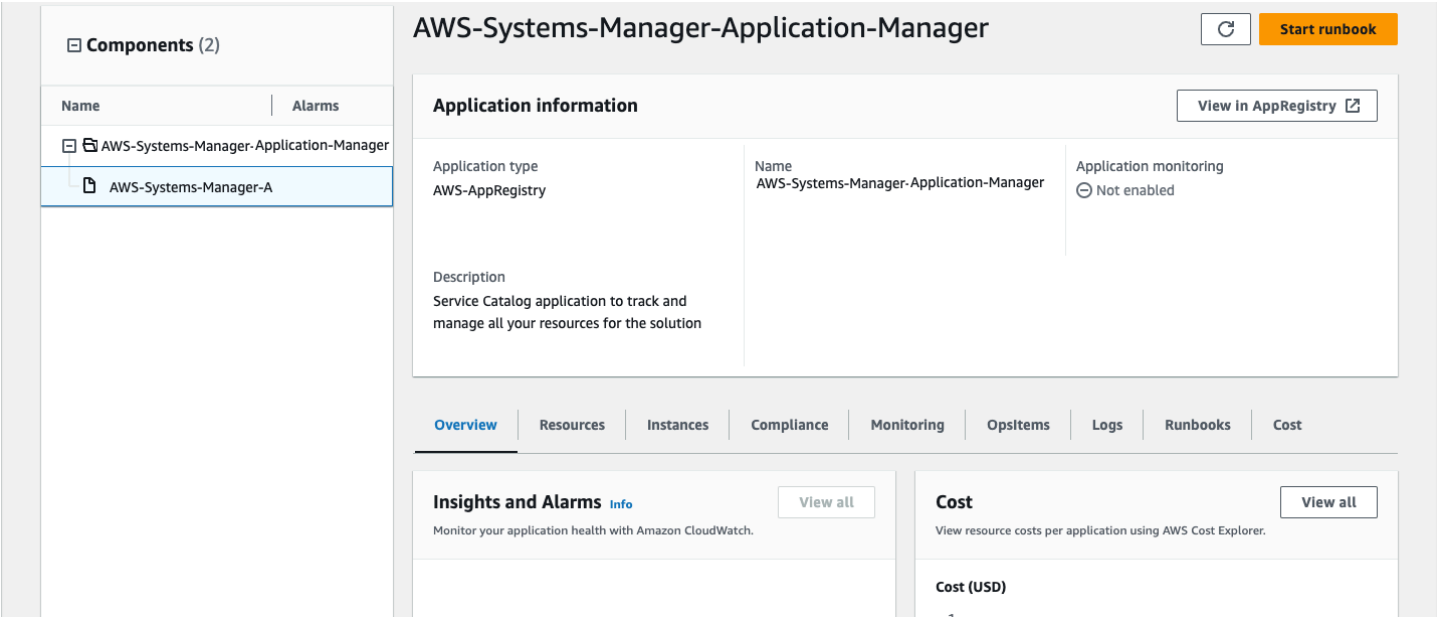
이 솔루션에는 CloudFormation 템플릿과 기본 리소스를 Service Catalog AppRegistry 및 AWS Systems Manager Application Manager의 애플리케이션으로 등록하는 Service Catalog AppRegistry 리소스가 포함되어 있습니다.

AWS Systems Manager Application Manager는이 솔루션과 해당 리소스에 대한 애플리케이션 수준 보기를 제공하므로 다음을 수행할 수 있습니다.

- 중앙 위치에서 리소스, 스택 및 AWS 계정에서 배포된 리소스 비용,이 솔루션과 관련된 로그를 모니터링합니다.
- 애플리케이션의 컨텍스트에서이 솔루션의 리소스(예: 배포 상태, CloudWatch 경보, 리소스 구성 및 운영 문제)에 대한 작업 데이터를 봅니다.

  다음 그림은 Application Manager의 솔루션 스택에 대한 애플리케이션 보기의 예를 보여줍니다.

Application Manager의 AWS 솔루션 스택을 보여줍니다.



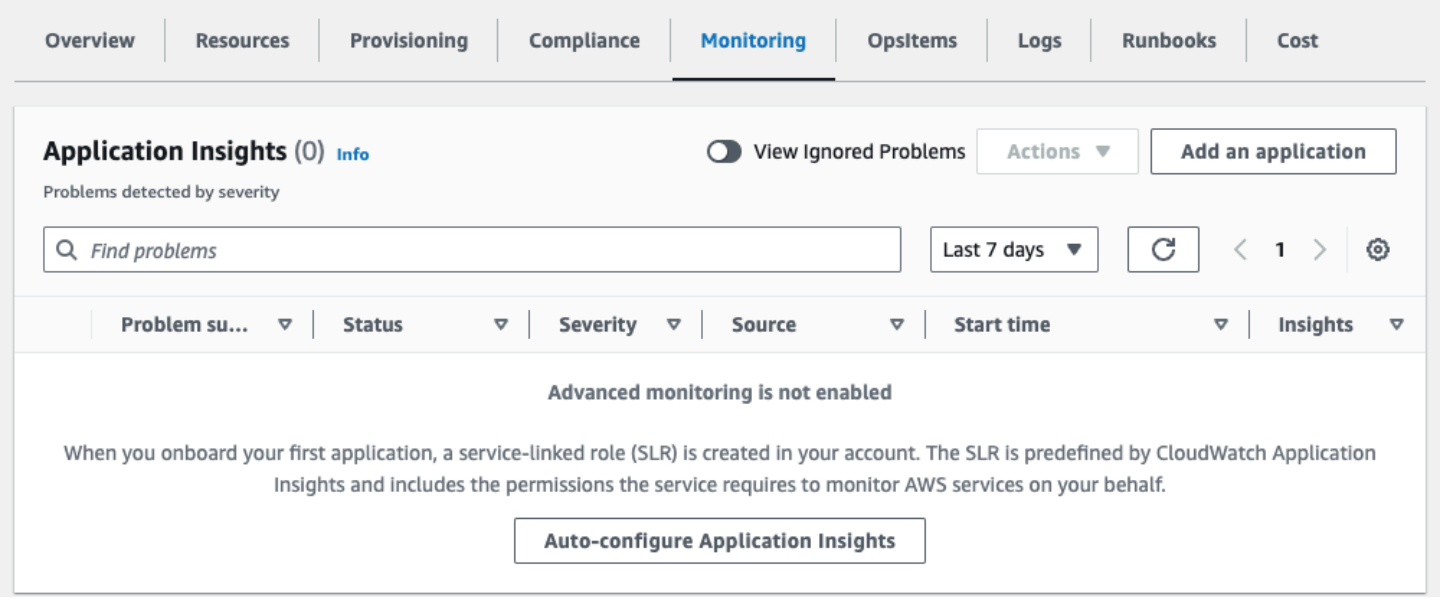
CloudWatch Application Insights 활성화

1. Systems Manager 콘솔에 로그인합니다.
2. 탐색 창에서 Application Manager를 선택합니다.
3. 애플리케이션에서이 솔루션의 애플리케이션 이름을 검색하고 선택합니다.

애플리케이션 이름은 애플리케이션 소스 열에 앱 레지스트리가 있으며 솔루션 이름, 리전, 계정 ID 또는 스택 이름의 조합이 있습니다.

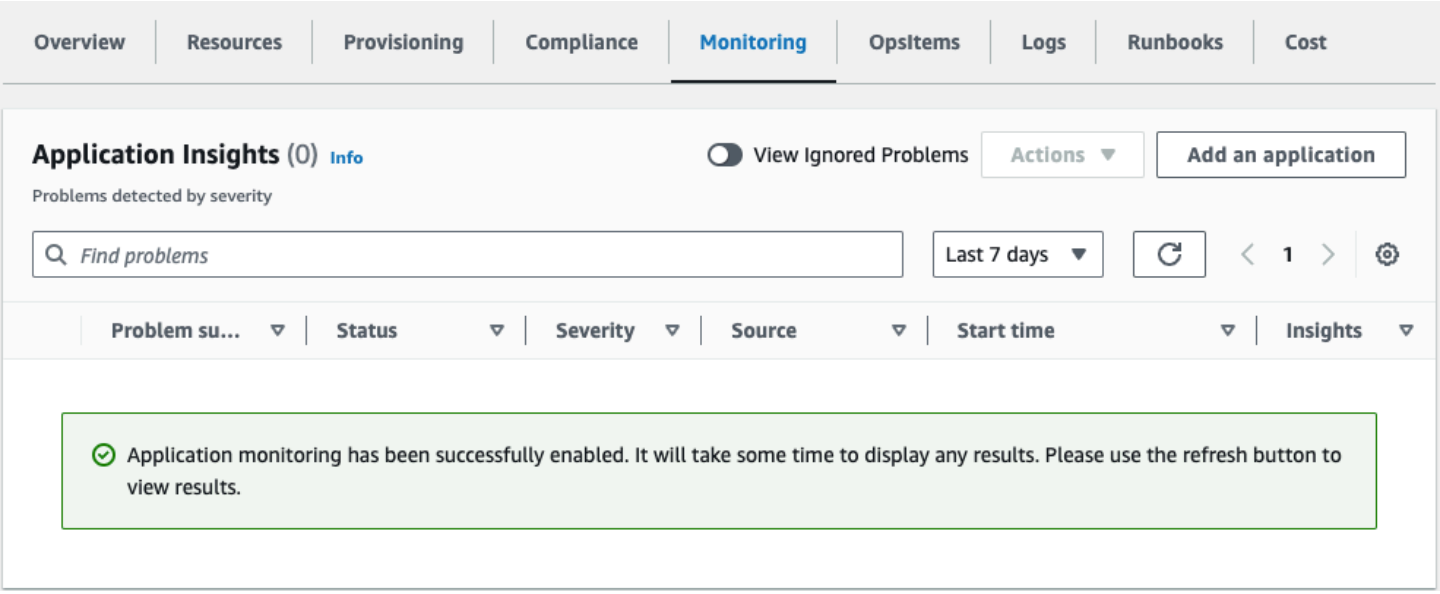
4. 구성 요소 트리에서 활성화하려는 애플리케이션 스택을 선택합니다.
5. 모니터링 탭의 Application Insights에서 Application Insights 자동 구성을 선택합니다.

감지된 문제가 없고 고급 모니터링이 활성화되지 않았음을 보여주는 Application Insights 대시보드



이제 애플리케이션 모니터링이 활성화되고 다음 상태 상자가 나타납니다.

성공적인 모니터링 활성화 메시지를 보여주는 Application Insights 대시보드

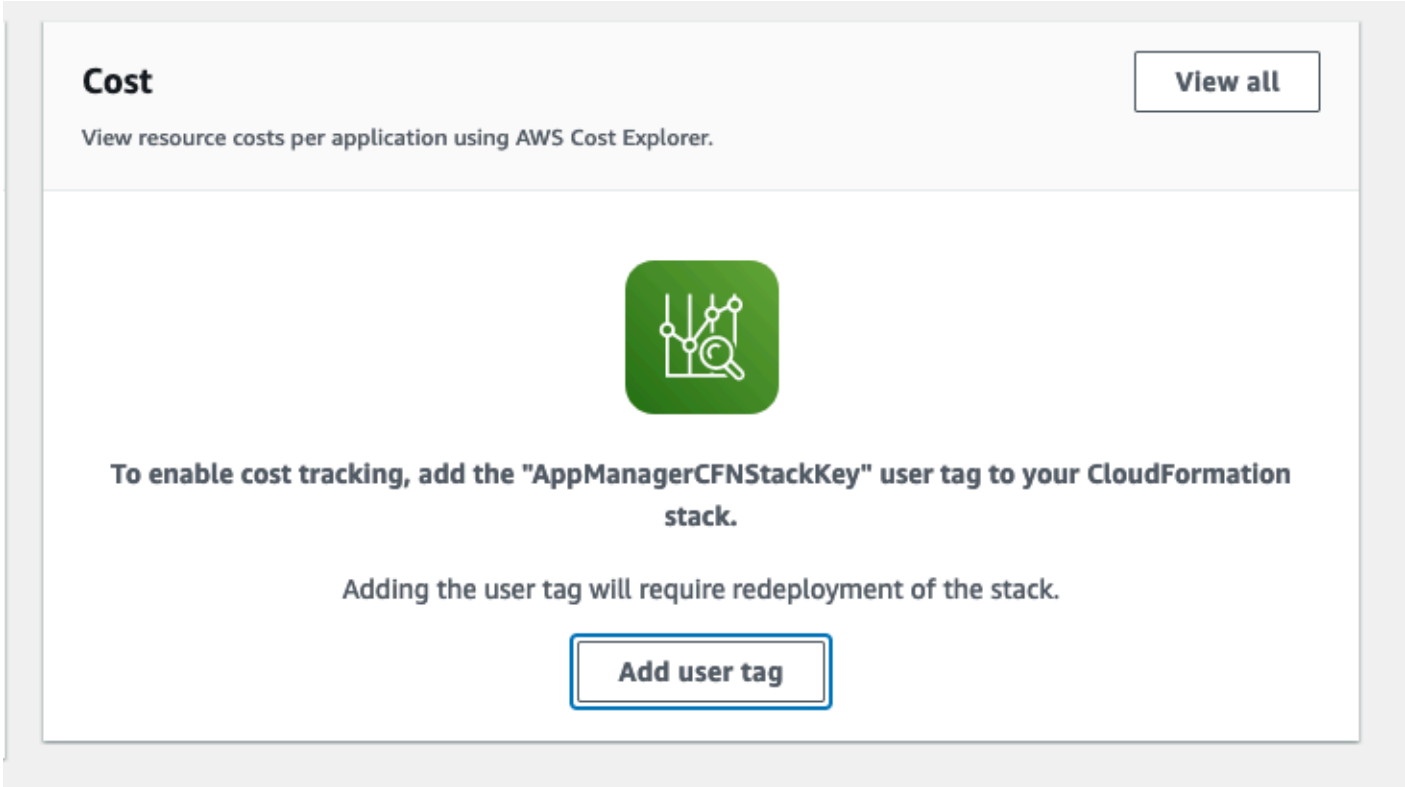


솔루션과 연결된 비용 태그 확인

솔루션과 관련된 비용 할당 태그를 활성화한 후 이 솔루션의 비용을 보려면 비용 할당 태그를 확인해야 합니다. 비용 할당 태그를 확인하려면 다음을 수행합니다.

1. Systems Manager 콘솔에 로그인합니다.
2. 탐색 창에서 Application Manager를 선택합니다.
3. 애플리케이션에서 이 솔루션의 애플리케이션 이름을 선택합니다.
4. 개요 탭의 비용에서 사용자 태그 추가를 선택합니다.

Application Cost 사용자 태그 추가 화면을 보여주는 스크린샷



5. 사용자 태그 추가 페이지에서 `confirm`를 입력한 다음 사용자 태그 추가를 선택합니다.

활성화 프로세스가 완료되고 태그 데이터가 표시되는 데 최대 24시간 정도 걸릴 수 있습니다.

솔루션과 관련된 비용 할당 태그 활성화

이 솔루션과 연결된 비용 태그를 확인한 후 비용 할당 태그를 활성화하여이 솔루션의 비용을 확인해야 합니다. 비용 할당 태그는 조직의 관리 계정에서만 활성화할 수 있습니다.

비용 할당 태그를 활성화하려면 다음을 수행합니다.

1. AWS Billing and Cost Management and Cost Management 콘솔에 로그인합니다.
2. 탐색 창에서 비용 할당 태그를 선택합니다.
3. 비용 할당 태그 페이지에서 `AppManagerCFNStackKey` 태그를 필터링한 다음 표시된 결과에서 태그를 선택합니다.
4. 활성화를 선택합니다.

AWS Cost Explorer

AWS Cost Explorer와의 통합을 통해 Application Manager 콘솔 내에서 애플리케이션 및 애플리케이션 구성 요소와 관련된 비용의 개요를 볼 수 있습니다. Cost Explorer는 시간 경과에 따른 AWS 리소스 비용 및 사용량을 확인하여 비용을 관리하는 데 도움이 됩니다.

1. AWS Cost Management 콘솔에 로그인합니다.
2. 탐색 메뉴에서 Cost Explorer를 선택하여 시간 경과에 따른 솔루션의 비용 및 사용량을 확인합니다.

솔루션 업데이트

이전에 솔루션을 배포한 경우 다음 절차에 따라 솔루션의 CloudFormation 스택을 업데이트하여 솔루션 프레임워크의 최신 버전을 가져옵니다.

1. CloudFormation 콘솔에 로그인하고 기존 CloudFormation 스택을 선택한 다음 업데이트를 선택합니다.
2. 현재 템플릿 교체를 선택합니다.
3. 템플릿 지정에서 다음을 수행합니다.
   a. Amazon S3 URL을 선택합니다.
   b. 최신 템플릿의 링크를 복사합니다.
   c. Amazon S3 URL 상자에 링크를 붙여넣습니다.
   d. Amazon S3 URL텍스트 상자에 올바른 템플릿 URL이 표시되는지 확인합니다.
   e. Next(다음)를 선택합니다.
   f. 다음을 다시 선택합니다.
4. 파라미터에서 템플릿의 파라미터를 검토하고 필요에 따라 수정합니다. 파라미터에 대한 자세한 내용은 스택 시작을 참조하세요.
5. Next(다음)를 선택합니다.
6. Configure stack options(스택 옵션 구성) 페이지에서 Next(다음)를 선택합니다.
7. 검토 페이지에서 설정을 검토하고 확인합니다.
8. 템플릿이 IAM 리소스를 생성할 수 있음을 확인하는 상자를 선택합니다.
9. 변경 세트 보기를 선택하고 변경 사항을 확인합니다.
10스택 생성을 선택하여 스택을 배포합니다.

AWS CloudFormation 콘솔의 상태 열에서 스택의 상태를 볼 수 있습니다. 약 15분 후에 `UPDATE_COMPLETE` 상태를 받게 됩니다.

v3.2.6 이전 DLT 버전에서 최신 버전으로 업데이트하면 스택 업데이트가 실패합니다.

1. distributed-load-testing-on-aws.template를 다운로드합니다.
2. 템플릿을 열고 조건:으로 이동하여 DLTCommonResourcesAppRegistryCondition을 찾습니다.

3. 다음과 비슷한 내용이 표시되어야 합니다.

```
Conditions:
DLTCommonResourcesAppRegistryConditionCCEF54F8:
Fn::Equals:
- "true"
- "true"
```

4. 두 번째 true 값을 false로 변경합니다.

```
Conditions:
DLTCommonResourcesAppRegistryConditionCCEF54F8:
Fn::Equals:
- "true"
- "false"
```

5. 사용자 지정 템플릿을 사용하여 스택을 업데이트합니다.
6. 이 스택은 스택에서 앱 레지스트리 관련 리소스를 제거합니다. 따라서 업데이트를 완료해야 합니다.
7. 최신 템플릿 URL을 사용하여 다른 스택 업데이트를 수행하여 앱 레지스트리 애플리케이션 리소스를 스택에 다시 추가합니다.

> **Note**
>
> AWS Systems Manager Application Manager는이 솔루션과 해당 리소스에 대한 애플리케이션 수준 보기를 제공하므로 다음을 수행할 수 있습니다.
>
> 1. 중앙 위치에서 리소스, 스택 및 AWS 계정에서 배포된 리소스 비용,이 솔루션과 관련된 로그를 모니터링합니다.
> 2. 배포 상태, CloudWatch 경보, 리소스 구성 및 운영 문제와 같은 애플리케이션의 컨텍스트에서이 솔루션의 리소스에 대한 작업 데이터를 봅니다.

문제 해결

알려진 문제 해결은 알려진 오류를 완화하기 위한 지침을 제공합니다. 이러한 지침으로 문제가 해결되지 않는 경우 AWS Support에 문의하세요.이 솔루션에 대한 AWS 지원 사례를 개설하기 위한 지침을 제공합니다.

알려진 문제 해결

문제: 기존 VPC를 사용 중이며 테스트가 실패하고 상태가 실패로 설정되면 다음 오류 메시지가 표시됩니다.

```
Test might have failed to run.
```

- 해상도:*

서브넷이 지정된 VPC에 존재하고 인터넷 게이트웨이 또는 NAT 게이트웨이를 통해 인터넷으로 연결되는 경로가 있는지 확인합니다. AWS Fargate는 테스트를 성공적으로 실행하기 위해 퍼블릭 리포지토리에서 컨테이너 이미지를 가져올 수 있는 액세스 권한이 필요합니다.

문제: 테스트 실행에 너무 오래 걸리거나 무기한으로 멈춤

- 해상도:*

테스트를 취소하고 AWS Fargate를 확인하여 모든 작업이 중지되었는지 확인합니다. 중지하지 않은 경우 모든 Fargate 작업을 수동으로 중지합니다. 계정의 온디맨드 Fargate 작업 제한을 확인하여 원하는 작업 수를 시작할 수 있는지 확인합니다. 또한 Lambda 작업 실행기 함수의 CloudWatch 로그를 확인하여 Fargate 작업을 시작할 때 실패에 대한 더 많은 인사이트를 얻을 수 있습니다. CloudWatch ECS 로그에서 실행 중인 Fargate 컨테이너에서 발생하는 작업에 대한 세부 정보를 확인합니다.

AWS Support에 문의

AWS 개발자 지원, AWS 비즈니스 지원 또는 AWS 엔터프라이즈 지원이 있는 경우 지원 센터를 사용하여이 솔루션에 대한 전문가 지원을 받을 수 있습니다. 이후 단원에서는 그 방법에 대해서 설명합니다.

사례 생성

1. 지원 센터에 로그인합니다.

2. 사례 생성을 선택합니다.

어떻게 도와드릴까요?

1. 기술 선택
2. 서비스에서 솔루션을 선택합니다.
3. 범주에서 AWS의 분산 로드 테스트를 선택합니다.
4. 심각도에서 사용 사례에 가장 적합한 옵션을 선택합니다.
5. 서비스, 범주 및 심각도를 입력하면 인터페이스가 일반적인 문제 해결 질문에 대한 링크를 채웁니다. 이러한 링크로 질문을 해결할 수 없는 경우 다음 단계: 추가 정보를 선택합니다.

추가 정보

1. 제목에 질문 또는 문제를 요약하는 텍스트를 입력합니다.
2. 설명에서 문제를 자세히 설명합니다.
3. 파일 연결을 선택합니다.
4. AWS Support에서 요청을 처리하는 데 필요한 정보를 첨부합니다.

사례를 더 빠르게 해결할 수 있도록 지원

1. 요청된 정보를 입력합니다.
2. 다음 단계: 지금 해결하거나 문의하기를 선택합니다.

지금 해결하거나 문의하기

1. 지금 해결 솔루션을 검토합니다.
2. 이러한 솔루션의 문제를 해결할 수 없는 경우 문의하기를 선택하고 요청된 정보를 입력한 다음 제출을 선택합니다.

솔루션 제거

AWS 관리 콘솔에서 또는 AWS 명령줄 인터페이스를 사용하여 AWS에서 분산 로드 테스트를 제거할 수 있습니다. 이 솔루션에서 생성한 콘솔, 시나리오 및 Amazon Simple Storage Service(Amazon S3) 버킷 로깅을 수동으로 삭제해야 합니다. AWS 솔루션 구현은 보존할 데이터를 저장한 경우 자동으로 삭제되지 않습니다.

> **Note**
>
> 리전 스택을 배포한 경우 기본 스택을 삭제하기 전에 해당 리전에서 스택을 삭제해야 합니다.

AWS 관리 콘솔 사용

1. AWS CloudFormation 콘솔에 로그인합니다.
2. 스택 페이지에서이 솔루션의 설치 스택을 선택합니다.
3. Delete(삭제)를 선택합니다.

AWS 명령줄 인터페이스 사용

환경에서 AWS 명령줄 인터페이스(AWS CLI)를 사용할 수 있는지 확인합니다. 설치 지침은 AWS CLI 사용 설명서의 AWS 명령줄 인터페이스란 무엇입니까?를 참조하세요. AWS CLI를 사용할 수 있는지 확인한 후 다음 명령을 실행합니다.

```
$ aws cloudformation delete-stack --stack-name <installation-stack-name>
```

Amazon S3 버킷 삭제

이 솔루션은 실수로 데이터가 손실되지 않도록 AWS CloudFormation 스택을 삭제하기로 결정한 경우 솔루션에서 생성한 Amazon S3 버킷(옵트인 리전에 배포용)을 유지하도록 구성됩니다. 솔루션을 제거한 후 데이터를 보존할 필요가 없는 경우이 S3 버킷을 수동으로 삭제할 수 있습니다. 다음 단계에 따라 Amazon S3 버킷을 삭제합니다.

1. Amazon S3 콘솔에 로그인합니다.
2. 왼쪽 탐색 창에서 버킷을 선택합니다.

3. 이름으로 버킷 찾기 필드에이 솔루션 스택의 이름을 입력합니다.
4. 솔루션의 S3 버킷 중 하나를 선택하고 비우기를 선택합니다.
5. 확인 필드에 영구적으로 삭제를 입력하고 비우기를 선택합니다.
6. 방금 비운 S3 버킷 이름을 선택하고 삭제를 선택합니다.
7. 확인 필드에 S3 버킷 이름을 입력하고 버킷 삭제를 선택합니다.

모든 S3 버킷을 삭제할 때까지 3~7단계를 반복합니다.

AWS CLI를 사용하여 S3 버킷을 삭제하려면 다음 명령을 실행합니다.

```
$ aws s3 rb s3://<bucket-name> --force
```

솔루션 사용

이 섹션에는 테스트 결과, 테스트 예약 워크플로 및 라이브 데이터를 포함하여 AWS에서 분산 로드 테스트를 사용하는 방법에 대한 정보가 포함되어 있습니다.

테스트 결과

AWS의 분산 로드 테스트는 로드 테스트 프레임워크를 활용하여 대규모로 애플리케이션 테스트를 실행합니다. 테스트가 완료되면 다음 결과가 포함된 세부 보고서가 생성됩니다.

- 평균 응답 시간 - 테스트에서 생성된 모든 요청에 대한 평균 응답 시간입니다.
- 평균 지연 시간 - 테스트에서 생성된 모든 요청에 대한 초 단위의 평균 지연 시간입니다.
- 평균 연결 시간 - 테스트에서 생성된 모든 요청에 대해 호스트에 연결하는 데 걸리는 초 단위의 평균 시간입니다.
- 평균 대역폭 - 테스트에서 생성된 모든 요청에 대한 평균 대역폭입니다.
- 총 수 - 총 요청 수입니다.
- 성공 수 - 성공한 총 요청 수입니다.
- 오류 수 - 총 오류 수입니다.
- 초당 요청 수 - 테스트에서 생성된 모든 요청에 대한 초당 평균 요청 수입니다.
- 백분위수 - 테스트에 대한 응답 시간의 백분위수입니다. 최대 응답 시간은 100%이고, 최소 응답 시간은 0%입니다.

> **Note**
>
> 테스트 결과는 콘솔에 표시됩니다. `Scenarios` Amazon S3 버킷의 `Results` 폴더에서 원시 테스트 결과에 대한 XML 파일을 볼 수 있습니다.

Taurus 테스트 결과에 대한 자세한 내용은 Taurus 사용 설명서의 테스트 보고서 생성을 참조하세요.

테스트 예약 워크플로

웹 콘솔을 사용하여 로드 테스트를 예약합니다. 테스트를 예약할 때 다음 워크플로가 실행됩니다.

- 예약 옵션으로 로드 테스트가 생성되면 일정 파라미터가 Amazon API Gateway를 통해 솔루션의 API로 전송됩니다.
- 그런 다음 API는 파라미터를 CloudWatch Events 규칙을 생성하는 Lambda 함수에 전달합니다.이 규칙은 지정된 날짜에 실행되도록 예약됩니다.
- 테스트가 일회성 테스트인 경우 CloudWatch Events 규칙은 지정된 날짜에 실행됩니다. `api-services` Lambda 함수는 테스트 워크플로를 통해 새 테스트를 실행합니다.
- 테스트가 반복 테스트인 경우 CloudWatch Events 규칙이 지정된 날짜에 활성화됩니다. `api-services` Lambda 함수는 현재 CloudWatch Events 규칙을 삭제하고 생성 시 즉시 실행되는 다른 규칙을 생성한 다음 지정된 반복 빈도에 따라 반복적으로 실행합니다.

사용자 수 결정

컨테이너가 테스트에 지원할 수 있는 사용자 수는 사용자 수를 점진적으로 늘리고 Amazon CloudWatch에서 성능을 모니터링하여 확인할 수 있습니다. CPU 및 메모리 성능이 제한에 도달하는 것을 관찰하면 컨테이너가 기본 구성(vCPU 2개 및 메모리 4GB)에서 해당 테스트에 지원할 수 있는 최대 사용자 수에 도달한 것입니다. 다음 예제를 사용하여 테스트의 동시 사용자 제한을 확인할 수 있습니다.

1. 사용자가 200명 이하인 테스트를 생성합니다.
2. 테스트가 실행되는 동안 CloudWatch 콘솔을 사용하여 CPU와 메모리를 모니터링합니다.
   a. 왼쪽 탐색 창의 Container Insights에서 성능 모니터링을 선택합니다.
   b. 성능 모니터링 페이지의 왼쪽 드롭다운 메뉴에서 ECS 클러스터를 선택합니다.
   c. 오른쪽 드롭다운 메뉴에서 Amazon Elastic Container Service(Amazon ECS) 클러스터를 선택합니다.
3. 모니터링하는 동안 CPU와 메모리를 관찰합니다. CPU가 75%를 초과하지 않거나 메모리가 85%를 초과하지 않는 경우(일회성 피크 무시) 사용자 수가 많은 다른 테스트를 실행할 수 있습니다.

테스트가 리소스 제한을 초과하지 않은 경우 1~3단계를 반복합니다. 선택적으로 컨테이너 리소스를 늘려 동시 사용자 수를 늘릴 수 있습니다. 그러나 이로 인해 비용이 더 많이 듭니다. 자세한 내용은이 가이드의 컨테이너 리소스 증가 섹션을 참조하세요.

> **Note**
>
> 정확한 결과를 얻으려면 동시 사용자 제한을 결정할 때 한 번에 하나의 테스트만 실행합니다. 모든 테스트는 동일한 클러스터를 사용하며 CloudWatch 컨테이너 인사이트는 클러스터를 기

반으로 성능 데이터를 집계합니다. 이로 인해 두 테스트가 CloudWatch 컨테이너 인사이트에 동시에 보고되어 단일 테스트에 대한 리소스 사용률 지표가 부정확해집니다.

엔진당 사용자 보정에 대한 자세한 내용은 BlazeMeter 설명서의 타우루스 테스트 보정을 참조하세요.

라이브 데이터

테스트를 실행할 때 라이브 데이터를 선택적으로 포함하여 발생하는 상황에 대한 실시간 인사이트를 얻을 수 있습니다. Fargate 태스크의 CloudWatch 로그 그룹에는 라이브 데이터 옵션이 포함된 테스트 결과에 대한 구독 필터가 포함되어 있습니다. 솔루션이 패턴을 찾으면 데이터를 구조화하고 AWS IoT Core 주제에 게시하는 Lambda 함수를 시작합니다. 웹 콘솔은 주제를 구독하고 수신 데이터를 수신하며 1초 간격으로 집계된 데이터를 그래프로 표시합니다. 웹 콘솔에는 평균 응답 시간, 가상 사용자, 성공 및 실패의 4가지 그래프가 포함되어 있습니다.

> **Note**
>
> 데이터는 임시 데이터이며 테스트가 실행되는 동안 발생하는 상황을 확인하는 데만 사용됩니다. 테스트가 완료되면 솔루션은 결과 데이터를 DynamoDB 및 Amazon S3에 저장합니다. 웹 콘솔은 최대 5,000개의 데이터 포인트를 유지하며, 그 이후에는 가장 오래된 데이터가 최신 데이터로 대체됩니다. 페이지가 새로 고쳐지면 그래프가 비어 있고 사용 가능한 다음 데이터 포인트에서 시작됩니다.

테스트 취소 워크플로

웹 콘솔에서 로드 테스트를 취소하면 솔루션이 다음 테스트 취소 워크플로를 실행합니다.

1. 취소 요청이 `microservices` API로 전송됩니다.
2. `microservices` API는 현재 시작된 모든 작업이 중지될 때까지 작업을 취소하는 `task-canceler` Lambda 함수를 호출합니다.
3. `task-runner` Lambda 함수를 처음 호출한 후에도 `task-canceler` Lambda 함수가 계속 실행되면 작업이 계속 시작됩니다. `task-runner` Lambda 함수가 완료되면 AWS Step Functions는 `Cancel Test` 단계로 계속 진행하여 `task-canceler` Lambda 함수를 다시 실행하여 나머지 작업을 중지합니다.

개발자 안내서

이 섹션에서는 솔루션의 소스 코드와 추가 사용자 지정을 제공합니다.

소스 코드

GitHub 리포지토리를 방문하여이 솔루션의 템플릿 및 스크립트를 다운로드하고 사용자 지정을 다른 사용자와 공유합니다.

컨테이너 이미지 사용자 지정

이 솔루션은 AWS에서 관리하는 퍼블릭 Amazon Elastic Container Registry(Amazon ECR) 이미지 리포지토리를 사용하여 구성된 테스트를 실행하는 데 사용되는 이미지를 저장합니다. 컨테이너 이미지를 사용자 지정하려면 이미지를 다시 빌드하고 자체 AWS 계정의 ECR 이미지 리포지토리로 푸시할 수 있습니다.

이 솔루션을 사용자 지정하려면 기본 컨테이너 이미지를 사용하거나 필요에 맞게이 컨테이너를 편집할 수 있습니다. 솔루션을 사용자 지정하는 경우 사용자 지정 솔루션을 빌드하기 전에 다음 코드 샘플을 사용하여 환경 변수를 선언합니다.

```
#!/bin/bash
export REGION=aws-region-code # the AWS region to launch the solution (e.g. us-east-1)
export BUCKET_PREFIX=my-bucket-name # prefix of the bucket name without the region code
export BUCKET_NAME=$BUCKET_PREFIX-$REGION # full bucket name where the code will reside
export SOLUTION_NAME=my-solution-name
export VERSION=my-version # version number for the customized code
export PUBLIC_ECR_REGISTRY=public.ecr.aws/awssolutions/distributed-load-testing-on-
aws-load-tester # replace with the container registry and image if you want to use a
 different container image export PUBLIC_ECR_TAG=v3.1.0 # replace with the container
 image tag if you want to use a different container image
```

컨테이너 이미지를 사용자 지정하도록 선택한 경우 프라이빗 이미지 리포지토리 또는 AWS 계정의 퍼블릭 이미지 리포지토리에서 호스팅할 수 있습니다. 이미지 리소스는 코드 베이스에 있는 `deployment/ecr/distributed-load-testing-on-aws-load-tester` 디렉터리에 있습니다.

이미지를 빌드하여 호스트 대상으로 푸시할 수 있습니다.

- 프라이빗 Amazon ECR 리포지토리 및 이미지는 Amazon ECR 사용 설명서의 Amazon ECR 프라이빗 리포지토리 및 프라이빗 이미지를 참조하세요.

- 퍼블릭 Amazon ECR 리포지토리 및 이미지는 Amazon ECR 퍼블릭 사용 설명서의 Amazon ECR 퍼블릭 리포지토리 및 퍼블릭 이미지를 참조하세요.

자체 이미지를 생성한 후에는 사용자 지정 솔루션을 빌드하기 전에 다음 환경 변수를 선언할 수 있습니다.

```
#!/bin/bash
export PUBLIC_ECR_REGISTRY=YOUR_ECR_REGISTRY_URI # e.g. YOUR_ACCOUNT_ID.dkr.ecr.us-
east-1.amazonaws.com/YOUR_IMAGE_NAME
export PUBLIC_ECR_TAG=YOUR_ECR_TAG # e.g. latest, v2.0.0
```

다음 예제에서는 컨테이너 파일을 보여줍니다.

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2023-minimal

RUN dnf update -y && \
    dnf install -y python3.11 python3.11-pip java-21-amazon-corretto bc procps jq
 findutils unzip && \
    dnf clean all

ENV PIP_INSTALL="pip3.11 install --no-cache-dir"


# install bzt
RUN $PIP_INSTALL --upgrade bzt awscli setuptools==70.0.0

# install bzt tools
RUN bzt -install-tools -o modules.install-
checker.exclude=selenium,gatling,tsung,siege,ab,k6,external-results-
loader,locust,junit,testng,rspec,mocha,nunit,xunit,wdio
RUN rm -rf /root/.bzt/selenium-taurus
RUN mkdir /bzt-configs /tmp/artifacts
ADD ./load-test.sh /bzt-configs/
ADD ./*.jar /bzt-configs/
ADD ./*.py /bzt-configs/

RUN chmod 755 /bzt-configs/load-test.sh
RUN chmod 755 /bzt-configs/ecslistener.py
RUN chmod 755 /bzt-configs/ecscontroller.py
RUN chmod 755 /bzt-configs/jar_updater.py
RUN python3.11 /bzt-configs/jar_updater.py
```

```
# Remove jar files from /tmp
RUN rm -rf /tmp/jmeter-plugins-manager-1.7*

# Add settings file to capture the output logs from bzt cli
RUN mkdir -p /etc/bzt.d && echo '{"settings": {"artifacts-dir": "/tmp/artifacts"}}' > /
etc/bzt.d/90-artifacts-dir.json

WORKDIR /bzt-configs
ENTRYPOINT ["./load-test.sh"]
```

이 디렉터리에는 컨테이너 파일 외에도 Taurus/Blazemeter 프로그램을 실행하기 전에 Amazon S3에서 테스트 구성을 다운로드하는 다음 bash 스크립트가 포함되어 있습니다.

```
#!/bin/bash

# set a uuid for the results xml file name in S3
UUID=$(cat /proc/sys/kernel/random/uuid)
pypid=0
echo "S3_BUCKET:: ${S3_BUCKET}"
echo "TEST_ID:: ${TEST_ID}"
echo "TEST_TYPE:: ${TEST_TYPE}"
echo "FILE_TYPE:: ${FILE_TYPE}"
echo "PREFIX:: ${PREFIX}"
echo "UUID:: ${UUID}"
echo "LIVE_DATA_ENABLED:: ${LIVE_DATA_ENABLED}"
echo "MAIN_STACK_REGION:: ${MAIN_STACK_REGION}"

cat /proc/self/cgroup
TASK_ID=$(cat /proc/self/cgroup | grep -oE '[a-f0-9]{32}' | head -n 1)
echo $TASK_ID

sigterm_handler() {
  if [ $pypid -ne 0 ]; then
    echo "container received SIGTERM."
    kill -15 $pypid
    wait $pypid
    exit 143 #128 + 15
  fi
}
trap 'sigterm_handler' SIGTERM

echo "Download test scenario"
```

```
aws s3 cp s3://$S3_BUCKET/test-scenarios/$TEST_ID-$AWS_REGION.json test.json --region
 $MAIN_STACK_REGION

# Set the default log file values to jmeter
LOG_FILE="jmeter.log"
OUT_FILE="jmeter.out"
ERR_FILE="jmeter.err"
KPI_EXT="jtl"

# download JMeter jmx file
if [ "$TEST_TYPE" != "simple" ]; then
  # setting the log file values to the test type
  LOG_FILE="${TEST_TYPE}.log"
  OUT_FILE="${TEST_TYPE}.out"
  ERR_FILE="${TEST_TYPE}.err"

  # set variables based on TEST_TYPE
  if [ "$TEST_TYPE" == "jmeter" ]; then
    EXT="jmx"
    TYPE_NAME="JMeter"
    # Copy *.jar to JMeter library path. See the Taurus JMeter path: https://
gettaurus.org/docs/JMeter/
    JMETER_LIB_PATH=`find ~/.bzt/jmeter-taurus -type d -name "lib"`
    echo "cp $PWD/*.jar $JMETER_LIB_PATH"
    cp $PWD/*.jar $JMETER_LIB_PATH

  fi

  if [ "$FILE_TYPE" != "zip" ]; then
    aws s3 cp s3://$S3_BUCKET/public/test-scenarios/$TEST_TYPE/$TEST_ID.$EXT ./ --
region $MAIN_STACK_REGION
  else
    aws s3 cp s3://$S3_BUCKET/public/test-scenarios/$TEST_TYPE/$TEST_ID.zip ./ --region
 $MAIN_STACK_REGION
    unzip $TEST_ID.zip
    echo "UNZIPPED"
    ls -l
    # only looks for the first test script file.
    TEST_SCRIPT=`find . -name "*.${EXT}" | head -n 1`
    echo $TEST_SCRIPT
    if [ -z "$TEST_SCRIPT" ]; then
      echo "There is no test script (.${EXT}) in the zip file."
      exit 1
    fi
```

```
    sed -i -e "s|$TEST_ID.$EXT|$TEST_SCRIPT|g" test.json

    # copy bundled plugin jars to jmeter extension folder to make them available to
 jmeter
    BUNDLED_PLUGIN_DIR=`find $PWD -type d -name "plugins" | head -n 1`
    # attempt to copy only if a /plugins folder is present in upload
    if [ -z "$BUNDLED_PLUGIN_DIR" ]; then
      echo "skipping plugin installation (no /plugins folder in upload)"
    else
      # ensure the jmeter extensions folder exists
      JMETER_EXT_PATH=`find ~/.bzt/jmeter-taurus -type d -name "ext"`
      if [ -z "$JMETER_EXT_PATH" ]; then
        # fail fast - if plugins bundled they will be needed for the tests
        echo "jmeter extension path (~/.bzt/jmeter-taurus/**/ext) not found - cannot
 install bundled plugins"
        exit 1
      fi
      cp -v $BUNDLED_PLUGIN_DIR/*.jar $JMETER_EXT_PATH
    fi
  fi
fi

#Download python script
if [ -z "$IPNETWORK" ]; then
    python3.11 -u $SCRIPT  $TIMEOUT &
    pypid=$!
    wait $pypid
    pypid=0
else
    aws s3 cp s3://$S3_BUCKET/Container_IPs/${TEST_ID}_IPHOSTS_${AWS_REGION}.txt ./ --
region $MAIN_STACK_REGION
    export IPHOSTS=$(cat ${TEST_ID}_IPHOSTS_${AWS_REGION}.txt)
    python3.11 -u $SCRIPT $IPNETWORK $IPHOSTS
fi

echo "Running test"

stdbuf -i0 -o0 -e0 bzt test.json -o modules.console.disable=true | stdbuf -i0 -o0 -e0
 tee -a result.tmp | sed -u -e "s|^|$TEST_ID $LIVE_DATA_ENABLED |"
CALCULATED_DURATION=`cat result.tmp | grep -m1 "Test duration" | awk -F ' ' '{ print
 $5 }' | awk -F ':' '{ print ($1 * 3600) + ($2 * 60) + $3 }'`

# upload custom results to S3 if any
```

```
# every file goes under $TEST_ID/$PREFIX/$UUID to distinguish the result correctly
if [ "$TEST_TYPE" != "simple" ]; then
  if [ "$FILE_TYPE" != "zip" ]; then
    cat $TEST_ID.$EXT | grep filename > results.txt
  else
    cat $TEST_SCRIPT | grep filename > results.txt
  fi

  if [ -f results.txt ]; then
    sed -i -e 's/<stringProp name="filename">//g' results.txt
    sed -i -e 's/<\/stringProp>//g' results.txt
    sed -i -e 's/ //g' results.txt

    echo "Files to upload as results"
    cat results.txt

    files=(`cat results.txt`)
    extensions=()
    for f in "${files[@]}"; do
      ext="${f##*.}"
      if [[ ! " ${extensions[@]} " =~ " ${ext} " ]]; then
        extensions+=("$ext")
      fi
    done

    # Find all files in the current folder with the same extensions
    all_files=()
    for ext in "${extensions[@]}"; do
      for f in *."$ext"; do
        all_files+=("$f")
      done
    done

    for f in "${all_files[@]}"; do
      p="s3://$S3_BUCKET/results/$TEST_ID/${TYPE_NAME}_Result/$PREFIX/$UUID/$f"
      if [[ $f = /* ]]; then
        p="s3://$S3_BUCKET/results/$TEST_ID/${TYPE_NAME}_Result/$PREFIX/$UUID$f"
      fi

        echo "Uploading $p"
        aws s3 cp $f $p --region $MAIN_STACK_REGION
    done
    fi
fi
```

```
if [ -f /tmp/artifacts/results.xml ]; then

  # Insert the Task ID at the same level as <FinalStatus>
  curl -s $ECS_CONTAINER_METADATA_URI_V4/task
  Task_CPU=$(curl -s $ECS_CONTAINER_METADATA_URI_V4/task | jq '.Limits.CPU')
  Task_Memory=$(curl -s $ECS_CONTAINER_METADATA_URI_V4/task | jq '.Limits.Memory')
  START_TIME=$(curl -s "$ECS_CONTAINER_METADATA_URI_V4/task" | jq -r
 '.Containers[0].StartedAt')
  # Convert start time to seconds since epoch
  START_TIME_EPOCH=$(date -d "$START_TIME" +%s)
  # Calculate elapsed time in seconds
  CURRENT_TIME_EPOCH=$(date +%s)
  ECS_DURATION=$((CURRENT_TIME_EPOCH - START_TIME_EPOCH))


  sed -i.bak 's/<\/FinalStatus>/<TaskId>'"$TASK_ID"'<\/TaskId><\/FinalStatus>/' /tmp/
artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<TaskCPU>'"$Task_CPU"'<\/TaskCPU><\/FinalStatus>/' /tmp/
artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<TaskMemory>'"$Task_Memory"'<\/TaskMemory><\/
FinalStatus>/' /tmp/artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<ECSDuration>'"$ECS_DURATION"'<\/ECSDuration><\/
FinalStatus>/' /tmp/artifacts/results.xml

  echo "Validating Test Duration"
  TEST_DURATION=$(grep -E '<TestDuration>[0-9]+.[0-9]+</TestDuration>' /tmp/artifacts/
results.xml | sed -e 's/<TestDuration>//' | sed -e 's/<\/TestDuration>//')

  if (( $(echo "$TEST_DURATION > $CALCULATED_DURATION" | bc -l) )); then
    echo "Updating test duration: $CALCULATED_DURATION s"
    sed -i.bak.td 's/<TestDuration>[0-9]*\.[0-9]*<\/TestDuration>/
<TestDuration>'"$CALCULATED_DURATION"'<\/TestDuration>/' /tmp/artifacts/results.xml
  fi

  if [ "$TEST_TYPE" == "simple" ]; then
    TEST_TYPE="jmeter"
  fi

  echo "Uploading results, bzt log, and JMeter log, out, and err files"
  aws s3 cp /tmp/artifacts/results.xml s3://$S3_BUCKET/results/${TEST_ID}/${PREFIX}-
${UUID}-${AWS_REGION}.xml --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/bzt.log s3://$S3_BUCKET/results/${TEST_ID}/bzt-${PREFIX}-
${UUID}-${AWS_REGION}.log --region $MAIN_STACK_REGION
```

```
  aws s3 cp /tmp/artifacts/$LOG_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.log --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/$OUT_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.out --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/$ERR_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.err --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/kpi.${KPI_EXT} s3://$S3_BUCKET/results/${TEST_ID}/kpi-
${PREFIX}-${UUID}-${AWS_REGION}.${KPI_EXT} --region $MAIN_STACK_REGION

else
  echo "An error occurred while the test was running."
fi
```

Dockerfile 및 bash 스크립트 외에도 디렉터리에 두 개의 Python 스크립트가 포함됩니다. 각 작업은 bash 스크립트 내에서 Python 스크립트를 실행합니다. 작업자 작업은 `ecslistener.py` 스크립트를 실행하는 반면 리더 작업은 `ecscontroller.py` 스크립트를 실행합니다. `ecslistener.py` 스크립트는 포트 50000에 소켓을 생성하고 메시지를 기다립니다. `ecscontroller.py` 스크립트는 소켓에 연결하여 테스트 시작 메시지를 작업자 작업에 전송하므로 동시에 시작할 수 있습니다.

분산 로드 테스트 API

이 로드 테스트 솔루션은 테스트 결과 데이터를 안전한 방식으로 노출하는 데 도움이 됩니다. API는 Amazon DynamoDB에 저장된 테스트 데이터에 액세스하기 위한 "정문" 역할을 합니다. APIs를 사용하여 솔루션에 빌드하는 모든 확장 기능에 액세스할 수도 있습니다.

이 솔루션은 식별 및 권한 부여를 위해 Amazon API Gateway와 통합된 Amazon Cognito 사용자 풀을 사용합니다. Amazon API Gateway 사용자 풀을 API와 함께 사용하는 경우 클라이언트는 유효한 자격 증명 토큰을 제공한 후에만 사용자 풀 활성화 메서드를 호출할 수 있습니다.

API를 통해 직접 테스트를 실행하는 방법에 대한 자세한 내용은 Amazon API Gateway REST API 참조 설명서의 서명 요청을 참조하세요.

솔루션의 API에서 다음 작업을 사용할 수 있습니다.

> **Note**
>
> `testScenario` 및 기타 파라미터에 대한 자세한 내용은 GitHub 리포지토리의 시나리오 및 페이로드 예제를 참조하세요.

시나리오

업무

리전

GET /scenarios

설명

GET /scenarios 작업을 통해 테스트 시나리오 목록을 검색할 수 있습니다.

응답

명칭	설명
data	각 테스트의 ID, 이름, 설명, 상태 및 실행 시간을 포함한 시나리오 목록

POST/시나리오

설명

POST /scenarios 작업을 통해 테스트 시나리오를 생성하거나 예약할 수 있습니다.

요청 본문

명칭	설명
testName	테스트의 이름입니다.
testDescription	테스트에 대한 설명
testTaskConfigs	시나리오에 region 대해 concurrency (병렬 실행 수), taskCount (테스트를 실행하는 데 필요한 작업 수) 및를 지정하는 객체입니다.
testScenario	동시성, 테스트 시간, 호스트 및 테스트 방법을 포함한 테스트 정의
testType	테스트 유형(예: , simplejmeter)
fileType	업로드 파일 유형(예: , nonescript, zip)
scheduleDate	테스트를 실행할 날짜입니다. 테스트를 예약하는 경우에만 제공됩니다(예: 2021-02-28).
scheduleTime	테스트를 실행하는 시간입니다. 테스트를 예약하는 경우에만 제공됩니다(예: 21:07).
scheduleStep	일정 프로세스의 단계입니다. 반복 테스트를 예약하는 경우에만 제공됩니다. (사용 가능한 단계에는 create 및가 포함됩니다start.)
cronvalue	반복 예약 사용자 지정을 위한 cron 값입니다. 사용되는 경우 scheduleDate 및 scheduleTime을 생략합니다.

명칭	설명
cronExpiryDate	cron이 만료되고 무기한 실행되지 않는 데 필요한 날짜입니다.
recurrence	예약된 테스트의 반복입니다. 반복 테스트를 예약하는 경우에만 제공됩니다(예: , dailyweeklybiweekly, 또는 monthly).

응답

명칭	설명
testId	테스트의 고유 ID
testName	테스트의 이름입니다.
status	테스트 상태

옵션/시나리오

설명

OPTIONS /scenarios 작업은 요청에 대한 응답을 올바른 CORS 응답 헤더와 함께 제공합니다.

응답

명칭	설명
testId	테스트의 고유 ID
testName	테스트의 이름입니다.
status	테스트 상태

GET /시나리오/{testId}

설명

`GET /scenarios/{testId}` 작업을 통해 특정 테스트 시나리오의 세부 정보를 검색할 수 있습니다.

요청 파라미터

`testId`

- 테스트의 고유 ID

  유형: 문자열

  필수 항목 여부: 예

응답

명칭	설명
`testId`	테스트의 고유 ID
`testName`	테스트의 이름입니다.
`testDescription`	테스트에 대한 설명
`testType`	실행 중인 테스트 유형(예: , `simplejmeter`)
`fileType`	업로드되는 파일 유형(예: , `nonescript`, `zip`)
`status`	테스트 상태
`startTime`	마지막 테스트가 시작된 시간 및 날짜
`endTime`	마지막 테스트가 종료된 시간 및 날짜
`testScenario`	동시성, 테스트 시간, 호스트 및 테스트 방법을 포함한 테스트 정의
`taskCount`	테스트를 실행하는 데 필요한 작업 수

명칭	설명
taskIds	테스트 실행을 위한 작업 IDs 목록
results	테스트의 최종 결과
history	과거 테스트의 최종 결과 목록
errorReason	오류가 발생할 때 생성되는 오류 메시지
nextRun	예약된 다음 실행(예: 2017-04-22 17:18:00)
scheduleRecurrence	테스트의 반복(예: , dailyweekly, biweekly, monthly)

POST/시나리오/{testId}

설명

POST /scenarios/{testId} 작업을 통해 특정 테스트 시나리오를 취소할 수 있습니다.

요청 파라미터

testId

- 테스트의 고유 ID

  유형: 문자열

  필수 항목 여부: 예

응답

명칭	설명
status	테스트 상태

DELETE/시나리오/{testId}

설명

DELETE /scenarios/{testId} 작업을 통해 특정 테스트 시나리오와 관련된 모든 데이터를 삭제할 수 있습니다.

요청 파라미터

testId

- 테스트의 고유 ID

  유형: 문자열

  필수 항목 여부: 예

응답

명칭	설명
status	테스트 상태

옵션/시나리오/{testId}

설명

OPTIONS /scenarios/{testId} 작업은 요청에 대한 응답을 올바른 CORS 응답 헤더와 함께 제공합니다.

응답

명칭	설명
testId	테스트의 고유 ID
testName	테스트의 이름입니다.

명칭	설명
testDescription	테스트에 대한 설명
testType	실행 중인 테스트 유형(예: , simplejmeter)
fileType	업로드되는 파일 유형(예: , nonescript, zip)
status	테스트 상태
startTime	마지막 테스트가 시작된 시간 및 날짜
endTime	마지막 테스트가 종료된 시간 및 날짜
testScenario	동시성, 테스트 시간, 호스트 및 테스트 방법을 포함한 테스트 정의
taskCount	테스트를 실행하는 데 필요한 작업 수
taskIds	테스트 실행을 위한 작업 IDs 목록
results	테스트의 최종 결과
history	과거 테스트의 최종 결과 목록
errorReason	오류가 발생할 때 생성되는 오류 메시지

GET/작업

설명

GET /tasks 작업을 통해 실행 중인 Amazon Elastic Container Service(Amazon ECS) 작업 목록을 검색할 수 있습니다.

응답

명칭	설명
tasks	테스트 실행을 위한 작업 IDs 목록

옵션/작업

설명

OPTIONS /tasks 작업 작업은 요청에 대한 응답을 올바른 CORS 응답 헤더와 함께 제공합니다.

응답

명칭	설명
taskIds	테스트 실행을 위한 작업 IDs 목록

GET/리전

설명

GET /regions 작업을 통해 해당 리전에서 테스트를 실행하는 데 필요한 리전 리소스 정보를 검색할 수 있습니다.

응답

명칭	설명
testId	리전 ID
ecsCloudWatchLogGroup	리전의 Amazon Fargate 작업에 대한 Amazon CloudWatch 로그 그룹의 이름입니다.
region	테이블의 리소스가 있는 리전
subnetA	리전에 있는 서브넷 중 하나의 ID입니다.
subnetB	리전에 있는 서브넷 중 하나의 ID입니다.
taskCluster	리전에 있는 AWS Fargate 클러스터의 이름
taskDefinition	리전에 있는 작업 정의의 ARN
taskImage	리전에 있는 작업 이미지의 이름입니다.

명칭	설명
taskSecurityGroup	리전에 있는 보안 그룹의 ID입니다.

옵션/리전

설명

OPTIONS /regions 작업은 요청에 대한 응답을 올바른 CORS 응답 헤더와 함께 제공합니다.

응답

명칭	설명
testId	리전 ID
ecsCloudWatchLogGroup	리전의 Amazon Fargate 작업에 대한 Amazon CloudWatch 로그 그룹의 이름입니다.
region	테이블의 리소스가 있는 리전
subnetA	리전에 있는 서브넷 중 하나의 ID입니다.
subnetB	리전에 있는 서브넷 중 하나의 ID입니다.
taskCluster	리전에 있는 AWS Fargate 클러스터의 이름
taskDefinition	리전에 있는 작업 정의의 ARN
taskImage	리전에 있는 작업 이미지의 이름입니다.
taskSecurityGroup	리전에 있는 보안 그룹의 ID입니다.

컨테이너 리소스 증가

현재 지원되는 사용자 수를 늘리려면 컨테이너 리소스를 늘리세요. 이를 통해 CPUs와 메모리를 늘려 동시 사용자의 증가를 처리할 수 있습니다.

새 작업 정의 개정 생성

1. Amazon Elastic Container Service 콘솔에 로그인합니다.
2. 왼쪽 탐색 메뉴에서 작업 정의를 선택합니다.
3. 이 솔루션에 해당하는 작업 정의 옆의 확인란을 선택합니다. 예: [replaceable]<stackName>`-EcsTaskDefinition-*<system-generated-random-Hash>*.
4. Create new revision(새 수정 생성)을 선택합니다.
5. 새 개정 생성 페이지에서 다음 작업을 수행합니다.
   a. 작업 크기에서 작업 메모리와 작업 CPU를 수정합니다.
   b. 컨테이너 정의에서 하드/소프트 메모리 제한을 검토합니다. 이 제한이 원하는 메모리보다 낮으면 컨테이너를 선택합니다.
   c. 컨테이너 편집 대화 상자에서 메모리 제한으로 이동하여 하드 제한을 원하는 메모리로 업데이트합니다.
   d. 업데이트를 선택합니다.
6. 새 개정 생성 페이지에서 생성을 선택합니다.
7. 작업 정의가 성공적으로 생성되면 새 작업 정의의 이름을 기록합니다. 이 이름에는 [replaceable]<stackName>`-EcsTaskDefinition-*<system-generated-random-Hash>*: [replaceable]와 같은 버전 번호가 포함됩니다<system-generated-versionNumber>.

DynamoDB 테이블 업데이트

1. DynamoDB 콘솔로 이동합니다.
2. 왼쪽 탐색 창에서 테이블 아래의 항목 탐색을 선택합니다.
3. 이 솔루션과 연결된 scenarios-table DynamoDB 테이블을 선택합니다. 예: [replaceable]<stackName>`-DLTTestRunnerStorageDLTScenariosTable-*<system-generated-random-Hash>*.
4. 작업 정의를 수정한 리전에 해당하는 항목을 선택합니다. 예를 들어 region-*<region-name>*입니다.
5. taskDefinition 속성을 새 작업 정의로 업데이트합니다.

레퍼런스

이 섹션에는이 솔루션의 고유한 지표를 수집하기 위한 선택적 기능, 관련 리소스에 대한 포인터,이 솔루션에 기여한 빌더 목록에 대한 정보가 포함되어 있습니다.

익명화된 데이터 수집

이 솔루션에는 익명화된 운영 지표를 AWS로 전송하는 옵션이 포함되어 있습니다. 당사는 이 데이터를 사용하여 고객이 이 솔루션과 관련 서비스 및 제품을 어떻게 사용하는지 더 잘 이해합니다. 호출되면 다음 정보가 수집되어 AWS로 전송됩니다.

- 솔루션 ID - AWS 솔루션 식별자
- 고유 ID(UUID) - 각 솔루션 배포에 대해 무작위로 생성된 고유 식별자
- 타임스탬프 - 데이터 수집 타임스탬프
- 테스트 유형 - 실행 중인 테스트 유형
- 파일 유형 - 업로드되는 파일 유형
- 작업 수 - 솔루션의 API를 통해 제출된 각 테스트의 작업 수
- 작업 기간 - 테스트를 실행하는 데 필요한 모든 작업의 총 실행 시간
- 테스트 결과 - 실행된 테스트의 결과

AWS는이 설문 조사를 통해 수집된 데이터를 소유합니다. 데이터 수집에는 AWS 개인정보 취급방침이 적용됩니다. 이 기능을 옵트아웃하려면 AWS CloudFormation 템플릿을 시작하기 전에 다음 단계를 완료하세요.

1. AWS CloudFormation 템플릿을 로컬 하드 드라이브에 다운로드합니다.
2. 텍스트 편집기를 사용하여 AWS CloudFormation 템플릿을 엽니다.
3. AWS CloudFormation 템플릿 매핑 섹션을 다음에서 수정합니다.

```
Solution:
  Config:
    SendAnonymousData: "Yes"
```

변경 후:

```
Solution:
```

```
  Config:
    SendAnonymousData: "No"
```

4. AWS CloudFormation 콘솔에 로그인합니다.
5. 스택 생성을 선택합니다.
6. 스택 생성 페이지, 템플릿 지정 섹션에서 템플릿 파일 업로드를 선택합니다.
7. 템플릿 파일 업로드에서 파일 선택을 선택하고 로컬 드라이브에서 편집한 템플릿을 선택합니다.
8. 다음을 선택하고이 가이드의 솔루션 배포 섹션에 있는 스택 시작의 단계를 따릅니다.

기여자

- Tom Nightingale
- 페르난도 딩러
- 이범석
- George Lenz
- 에린 McGill
- Dimitri Lopez
- 카미아 자바리
- Bassem Wanis
- 가빗 싱
- Nikhil Reddy

개정

버전별 개선 사항 및 수정 사항을 추적하려면 GitHub 리포지토리의 CHANGELOG.md를 방문하세요.

고지 사항

고객은 본 문서의 정보를 독립적으로 평가할 책임이 있습니다. 이 문서는 (a) 정보 제공 목적으로만 사용되며, (b) 예고 없이 변경될 수 있는 AWS의 현재 제품 제공 및 관행을 나타내며, (c) AWS 및 그 계열사, 공급업체 또는 라이선스 제공자로부터 어떠한 약속이나 보장도 생성하지 않습니다. AWS 제품 또는 서비스는 명시적이든 묵시적이든 어떠한 종류의 보증, 표현 또는 조건 없이 "있는 그대로" 제공됩니다. 고객에 대한 AWS의 책임과 책임은 AWS 계약의 적용을 받으며,이 문서는 AWS와 고객 간의 계약의 일부이거나 수정하지 않습니다.

AWS의 분산 로드 테스트는 Apache Software Foundation에서 제공되는 Apache 라이선스 버전 2.0의 약관에 따라 라이선스가 부여됩니다.

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