Guia de implementação

Teste de carga distribuído na AWS

Teste de carga distribuído na AWS: Guia de implementação

Table of Contents

Automatize o teste de seus aplicativos de software em grande escala

Data de publicação: dezembro de 2025

O teste de carga distribuído na AWS ajuda você a automatizar os testes de desempenho de seus aplicativos de software em grande escala para identificar gargalos antes de lançar seu aplicativo. Essa solução simula milhares de usuários conectados gerando solicitações HTTP a uma taxa sustentada sem a necessidade de provisionar servidores.

Essa solução utiliza o Amazon Elastic Container Service (Amazon ECS) no AWS Fargate para implantar contêineres que executam suas simulações de teste de carga e oferece os seguintes recursos:

- Implante o Amazon ECS em contêineres do AWS Fargate que são executados de forma independente para testar a capacidade de carga do seu aplicativo.
- Simule dezenas de milhares de usuários simultâneos em várias regiões da AWS, gerando solicitações em um ritmo contínuo.
- Personalize seus testes de aplicativos usando JMeterK6, scripts de teste Locust ou uma configuração simples de endpoint HTTP.
- Agende testes de carga para serem executados imediatamente, em uma data e hora futuras ou em uma programação recorrente.
- Execute vários testes de carga simultaneamente em diferentes cenários e regiões.

Este guia de implementação fornece uma visão geral da solução Distributed Load Testing on AWS, sua arquitetura e componentes de referência, considerações para planejar a implantação e etapas de configuração para implantar a solução na nuvem da Amazon Web Services (AWS). Ele inclui links para um CloudFormation modelo da AWS que lança e configura os serviços da AWS necessários para implantar essa solução usando as melhores práticas da AWS para segurança e disponibilidade.

O público-alvo para usar os recursos e capacidades dessa solução em seu ambiente inclui arquitetos de infraestrutura de TI, administradores e DevOps profissionais com experiência prática em arquitetura na nuvem da AWS.

Use esta tabela de navegação para encontrar rapidamente respostas para estas perguntas:

Se você deseja...	Leia…
Conhecer o custo da execução dessa solução. O custo estimado para executar essa solução na região Leste dos EUA (Norte da Virgínia) é de USD $30,90 por mês para recursos da AWS.	Custos
Entenda as considerações de segurança dessa solução.	Segurança
Saiba como planejar cotas para essa solução.	Cotas
Saiba quais regiões da AWS oferecem suporte a essa solução.	Regiões da AWS com suporte
Saiba mais sobre o MCP Server opcional para análise de testes de carga assistidos por IA.	Integração do servidor MCP
Visualize ou baixe o CloudFormation modelo da AWS incluído nesta solução para implantar automaticamente os recursos de infraestrutura (a “pilha”) dessa solução.	CloudFormation Modelo da AWS
Acessar o código-fonte e, opcionalmente, usar o AWS Cloud Development Kit (AWS CDK) para implantar a solução.	GitHub repositório

Recursos

A solução fornece os seguintes atributos:

Multiple Test Framework Support

Suporta JMeter scripts de teste K6 e Locust, bem como testes simples de endpoint HTTP sem a necessidade de scripts personalizados. Para obter mais informações, consulte Tipos de teste na seção Detalhes da arquitetura.

Simulação de alta carga de usuário

Simula dezenas de milhares de usuários virtuais simultâneos para testar seu aplicativo sob condições de carga realistas.

Distribuição de carga em várias regiões

Distribui testes de carga em várias regiões da AWS para simular o tráfego de usuários distribuído geograficamente e avaliar o desempenho global.

Programação flexível de testes

Agenda testes para serem executados imediatamente, em uma data e hora futuras específicas ou em uma programação recorrente usando expressões cron para testes de regressão automatizados.

monitoramento em tempo real

Fornece streaming de dados ao vivo opcional para monitorar o progresso do teste com métricas em tempo real, incluindo tempos de resposta, contagens de usuários virtuais e taxas de sucesso de solicitações.

Resultados de testes abrangentes

Exibe resultados de testes detalhados com métricas de desempenho, percentis (p50, p90, p95, p99), análise de erros e artefatos que podem ser baixados para análise off-line.

Comparação de linha de base

Designa execuções de teste de linha de base para comparação de desempenho para rastrear melhorias ou regressões ao longo do tempo.

Flexibilidade do endpoint

Testa qualquer endpoint HTTP ou HTTPS em todas as regiões da AWS, ambientes locais ou outros provedores de nuvem.

Console Web intuitivo

Fornece um console baseado na web para criar, gerenciar e monitorar testes sem a necessidade de interação com a linha de comando.

Análise assistida por IA (opcional)

Integra-se às ferramentas de desenvolvimento de IA por meio do servidor Model Context Protocol (MCP) para análise inteligente dos dados de teste de carga.

Support a vários protocolos

Suporta vários protocolos, incluindo HTTP, HTTPS, JDBC WebSocket, JMS, FTP e gRPC por meio de scripts de teste personalizados.

Benefícios

A solução oferece os seguintes benefícios:

Teste de desempenho abrangente

Suporta testes de carga, testes de estresse e testes de resistência para avaliar minuciosamente o desempenho do aplicativo sob várias condições.

Detecção precoce de problemas

Identifica gargalos de desempenho, vazamentos de memória e problemas de escalabilidade antes da implantação na produção, reduzindo o risco de interrupções.

Simulação de uso no mundo real

Simula com precisão o comportamento do usuário e os padrões de tráfego do mundo real para validar o desempenho do aplicativo em condições realistas.

Insights de desempenho acionáveis

Fornece métricas detalhadas, percentis e análise de erros para entender o comportamento do aplicativo e orientar os esforços de otimização.

Fluxos de trabalho de teste automatizados

Permite testes programados e recorrentes para monitoramento contínuo do desempenho e testes de regressão sem intervenção manual.

Infraestrutura econômica

Usa contêineres AWS Fargate sem servidor com pay-per-use preços, eliminando a necessidade de infraestrutura de testes dedicada e taxas de assinatura contínuas.

Implantação rápida de testes

Implanta e escala a infraestrutura de teste em minutos sem provisionar ou gerenciar servidores.

Fácil interrogação dos resultados do teste

Integra-se às ferramentas de desenvolvimento de IA por meio de um servidor opcional do Model Context Protocol (MCP), permitindo consultas em linguagem natural e análise inteligente de dados de teste de carga para insights e solução de problemas mais rápidos.

Casos de uso

Validação de pré-produção

Teste aplicativos web e móveis em condições de carga semelhantes às de produção antes de lançar uma nova versão para validar o desempenho e identificar problemas.

Planejamento de capacidade

Determine o número máximo de usuários simultâneos que seu aplicativo pode suportar com a infraestrutura atual e identifique quando é necessário escalar.

Verificação de carga de pico

Verifique se sua infraestrutura pode lidar com picos de carga, picos de tráfego sazonais ou picos inesperados na demanda sem degradação do desempenho.

Otimização de desempenho

Identifique gargalos de desempenho, como consultas lentas ao banco de dados, execução ineficiente de código, latência de rede ou restrições de recursos.

Teste de regressão

Agende testes de carga recorrentes para detectar regressões de desempenho introduzidas por novas implantações de código ou mudanças na infraestrutura.

Avaliação de desempenho global

Avalie o desempenho dos aplicativos de várias regiões geográficas para garantir uma experiência de usuário consistente para um público global.

Teste de carga da API

Teste os endpoints REST APIs, GraphQL ou microsserviços para validar os tempos de resposta, a taxa de transferência e as taxas de erro sob carga.

Integração de pipeline de CI/CD

Integre testes de desempenho automatizados em pipelines contínuos de integração e implantação para detectar problemas de desempenho no início do ciclo de desenvolvimento.

Teste de serviços de terceiros

Teste o desempenho e a confiabilidade de terceiros APIs ou serviços dos quais seu aplicativo depende sob várias condições de carga.

Conceitos e definições

Esta seção descreve os conceitos básicos e define a terminologia específica desta solução:

cenário

Definição do teste, incluindo o nome, a descrição, a contagem de tarefas, a simultaneidade, a região da AWS, o ramp-up, a espera, o tipo de teste, a data do agendamento e as configurações de recorrência.

contagem de tarefas

Número de contêineres que serão lançados no cluster Fargate para executar o cenário de teste. Tarefas adicionais não serão criadas quando o limite da conta nos recursos do Fargate for atingido. No entanto, as tarefas já em execução continuarão.

concurrency

A simultaneidade (número de usuários virtuais simultâneos por tarefa). A simultaneidade recomendada com base nas configurações padrão é 200. A simultaneidade é limitada pela CPU e pela memória. Para testes baseados no Apache JMeter, uma maior simultaneidade aumenta a memória utilizada pela JVM na tarefa do ECS. A definição de tarefas padrão do ECS cria tarefas com 4 GB de memória. É recomendável começar com valores de simultaneidade mais baixos para 1 tarefa e monitorar as CloudWatch métricas do ECS para o Grupo de Tarefas. Consulte as métricas de utilização de clusters do Amazon ECS.

ramp-up

O período de tempo para aumentar gradualmente de zero até o nível de simultaneidade desejado.

aguarde

O período de tempo para manter o nível de simultaneidade desejado após a conclusão da aceleração.

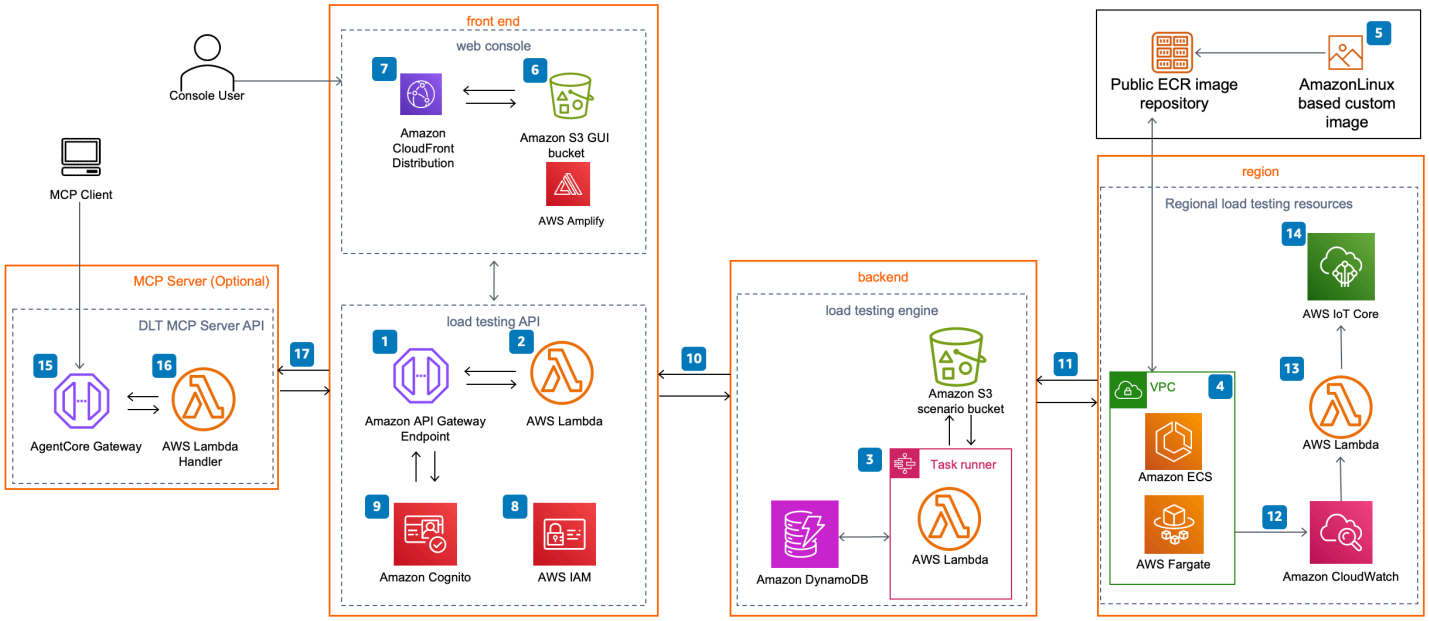
Para obter uma referência geral dos termos da AWS, consulte o Glossário da AWS.

Visão geral da arquitetura

Diagrama de arquitetura

A implantação dessa solução com os parâmetros padrão implanta os seguintes componentes em sua conta da AWS.

Teste de carga distribuído na arquitetura da AWS na AWS



> **Note**
>
> Os CloudFormation recursos da AWS são criados a partir de construções do AWS Cloud Development Kit (AWS CDK).

O fluxo de processo de alto nível para os componentes da solução implantados com o CloudFormation modelo da AWS é o seguinte:

1. Uma API de testador de carga distribuído utiliza o Amazon API Gateway para invocar os microsserviços da solução (funções do AWS Lambda).
2. Os microsserviços fornecem a lógica de negócios para gerenciar dados de teste e executar os testes.

3. Esses microsserviços interagem com o Amazon Simple Storage Service (Amazon S3), o Amazon DynamoDB e o AWS Step Functions para armazenar detalhes e resultados do cenário de teste e orquestrar a execução do teste.
4. Uma topologia de rede da Amazon Virtual Private Cloud (Amazon VPC) é implantada contendo os contêineres Amazon Elastic Container Service (Amazon ECS) da solução executados no AWS Fargate.
5. Os contêineres usam uma imagem base do Amazon Linux 2023 com a estrutura de teste de carga Taurus instalada. O Taurus é uma estrutura de automação de testes de código aberto que suporta K6 JMeter, Locust e outras ferramentas de teste. A imagem do contêiner é compatível com a Open Container Initiative (OCI) e é hospedada pela AWS em um repositório público do Amazon Elastic Container Registry (Amazon ECR). Para obter mais informações, consulte Personalização da imagem do contêiner.
6. Um console web desenvolvido pelo AWS Amplify é implantado em um bucket S3 configurado para hospedagem estática na web.
7. CloudFrontA Amazon fornece acesso público e seguro ao conteúdo do bucket do site da solução.
8. Durante a configuração inicial, a solução cria uma função de administrador padrão (função do IAM) e envia um convite de acesso para um endereço de e-mail de usuário especificado pelo cliente.
9. Um grupo de usuários do Amazon Cognito gerencia o acesso do usuário ao console, à API do testador de carga distribuído e ao servidor MCP.
10. Depois de implantar essa solução, você pode usar o console web ou APIs criar e executar cenários de teste que definam uma série de tarefas.
11. Os microsserviços usam esse cenário de teste para executar tarefas do ECS no Fargate nas regiões especificadas.
12. Quando o teste é concluído, a solução armazena os resultados no S3 e no DynamoDB e registra a saída na Amazon. CloudWatch
13. Se você habilitar a opção de dados ao vivo, a solução enviará CloudWatch registros das tarefas do Fargate para uma função Lambda durante o teste para cada região em que o teste é executado.
14. A função Lambda publica os dados no tópico correspondente no AWS IoT Core na região em que a pilha principal foi implantada. O console web se inscreve no tópico e exibe dados em tempo real durante a execução do teste.

> **Note**
>
> As etapas a seguir descrevem a integração opcional do MCP Server para análise de testes de carga assistidos por IA. Esse componente só será implantado se você selecionar a opção MCP Server durante a implantação da solução.

15. Um cliente MCP (ferramenta de desenvolvimento de IA) se conecta ao endpoint do AWS AgentCore Gateway para acessar os dados da solução Distributed Load Testing por meio do Model Context Protocol. AgentCore O Gateway valida o token de autenticação Cognito do usuário para garantir o acesso autorizado ao servidor MCP.
16. Após a autenticação bem-sucedida, o AgentCore Gateway encaminha a solicitação da ferramenta MCP para a função Lambda do DLT MCP Server. A função Lambda retorna os dados estruturados ao AgentCore Gateway, que os envia de volta ao cliente MCP para análises e insights assistidos por IA.
17. A função Lambda processa a solicitação e consulta os recursos apropriados da AWS (tabelas do DynamoDB, buckets do S3 ou CloudWatch logs) para recuperar os dados de teste de carga solicitados.

Considerações de design do AWS Well-Architected

Essa solução usa as melhores práticas do AWS Well-Architected Framework, que ajuda os clientes a projetar e operar cargas de trabalho confiáveis, seguras, eficientes e econômicas na nuvem.

Esta seção descreve como os princípios de design e as melhores práticas do Well-Architected Framework beneficiam essa solução.

Excelência operacional

Esta seção descreve como arquitetamos essa solução usando os princípios e as melhores práticas do pilar de excelência operacional.

- Todos os recursos são definidos como infraestrutura como código usando CloudFormation modelos da AWS gerados a partir de construções do AWS CDK.
- A solução envia métricas CloudWatch em vários estágios para fornecer observabilidade em funções Lambda, tarefas do ECS, buckets S3 e outros componentes da solução.

Segurança

Esta seção descreve como arquitetamos essa solução usando os princípios e as melhores práticas do pilar de segurança.

- Cognito autentica e autoriza usuários do console web e solicitações de API.
- Todas as comunicações entre serviços usam funções do AWS Identity and Access Management (IAM) com menos privilégios de acesso, contendo somente as permissões mínimas necessárias.
- Todo o armazenamento de dados, incluindo buckets S3 e tabelas do DynamoDB, criptografa dados em repouso usando chaves gerenciadas pela AWS.
- O registro, o rastreamento e o controle de versão são habilitados quando aplicável para fins de auditoria e conformidade.
- O acesso à rede é privado por padrão, com endpoints VPC habilitados quando disponíveis para manter o tráfego na rede da AWS.

Note

A solução cria vários grupos de CloudWatch registros com períodos de retenção variáveis com base no volume de registros e nas considerações de custo:

Tipo de log	Período de retenção
Informações sobre contêineres do ECS	1 dia
Step Functions, registros personalizados do ECS, registros de acesso ao API Gateway	1 ano
Registros de tempo de execução do Lambda	2 anos
Registros de execução do API Gateway	Nunca expire

Você pode modificar esses períodos de retenção no CloudWatch console com base em seus requisitos.

Confiabilidade

Esta seção descreve como arquitetamos essa solução usando os princípios e as melhores práticas do pilar de confiabilidade.

- A solução usa serviços sem servidor da AWS sempre que possível (exemplos: Lambda, API Gateway, Amazon S3, AWS Step Functions, Amazon DynamoDB e AWS Fargate) para garantir alta disponibilidade e recuperação de falhas no serviço.
- Todo o processamento computacional usa funções Lambda ou Amazon ECS no AWS Fargate.
- Os dados são armazenados no DynamoDB e no Amazon S3, portanto, persistem em várias zonas de disponibilidade por padrão.

Eficiência de desempenho

Esta seção descreve como arquitetamos essa solução usando os princípios e as melhores práticas do pilar de eficiência de desempenho.

- A solução usa uma arquitetura sem servidor com a capacidade de escalar horizontalmente conforme necessário.
- A solução pode ser lançada em qualquer região que ofereça suporte aos serviços da AWS nessa solução, como: AWS Lambda, Amazon API Gateway, Amazon S3, AWS Step Functions, Amazon DynamoDB, Amazon ECS, AWS Fargate e Amazon Cognito.
- A solução usa serviços gerenciados por toda parte para reduzir a carga operacional do provisionamento e gerenciamento de recursos.
- A solução é automaticamente testada e implantada diariamente para obter consistência à medida que os serviços da AWS mudam, bem como revisada por arquitetos de soluções e especialistas no assunto em relação às áreas a serem experimentadas e aprimoradas.

Otimização de custos

Esta seção descreve como arquitetamos essa solução usando os princípios e as melhores práticas do pilar de otimização de custos.

- A solução usa arquitetura sem servidor; portanto, os clientes só são cobrados pelo que usam.
- O Amazon DynamoDB escala a capacidade sob demanda, então você paga somente pela capacidade que você usa.

- O AWS ECS no AWS Fargate permite que você pague somente pelos recursos computacionais que você usa, sem despesas iniciais.
- O AWS AgentCore Gateway serve como um proxy econômico baseado em Lambda para a API de teste de carga distribuída, eliminando a necessidade de infraestrutura dedicada e reduzindo custos por meio de preços sem servidor. pay-per-request

Sustentabilidade

Esta seção descreve como arquitetamos essa solução usando os princípios e as melhores práticas do pilar de sustentabilidade.

- A solução usa serviços gerenciados sem servidor para minimizar o impacto ambiental dos serviços de back-end em comparação com os serviços locais em operação contínua.
- Os serviços de tecnologia sem servidor permitem aumentar a escala da solução verticalmente conforme necessário.

Detalhes de arquitetura

Esta seção descreve os componentes e os serviços da AWS que compõem essa solução e os detalhes da arquitetura sobre como esses componentes funcionam juntos.

A solução Distributed Load Testing on AWS consiste em três componentes de alto nível: um front-end, um back-end e um servidor MCP opcional.

Front-end

O front-end fornece as interfaces para interagir com a solução e inclui:

- Uma API de teste de carga para acesso programático
- Um console web para criar, programar e executar testes de desempenho
- Um servidor MCP opcional para análise assistida por IA de resultados e erros de testes

API de teste de carga

O teste de carga distribuído na AWS configura o Amazon API Gateway para hospedar a RESTful API da solução. Os usuários podem interagir com o sistema de teste de carga de forma segura por meio do console web incluído, da RESTful API e do servidor MCP opcional. A API atua como uma “porta de entrada” para acesso aos dados de teste armazenados no Amazon DynamoDB. Você também pode usar o APIs para acessar qualquer funcionalidade estendida incorporada à solução.

Essa solução aproveita os recursos de autenticação de usuários dos grupos de usuários do Amazon Cognito. Depois de autenticar um usuário com sucesso, o Amazon Cognito emite um token web JSON que é usado para permitir que o console envie solicitações para a solução (endpoints APIs do Amazon API Gateway). As solicitações HTTPS são enviadas pelo console para o APIs com o cabeçalho de autorização que inclui o token.

Com base na solicitação, o API Gateway invoca a função apropriada do AWS Lambda para realizar as tarefas necessárias nos dados armazenados nas tabelas do DynamoDB, armazenar cenários de teste como objetos JSON no Amazon S3, recuperar imagens de métricas da CloudWatch Amazon e enviar cenários de teste para a máquina de estado do AWS Step Functions.

Para obter mais informações sobre a API da solução, consulte a seção API de teste de carga distribuída deste guia.

Console web

Essa solução inclui um console web que você pode usar para configurar e executar testes, monitorar testes em execução e visualizar resultados detalhados dos testes. O console é um aplicativo ReactJS criado com o Cloudscape, um sistema de design de código aberto para criar aplicativos web intuitivos. O console está hospedado no Amazon S3 e acessado pela Amazon. CloudFront O aplicativo utiliza o AWS Amplify para se integrar ao Amazon Cognito para autenticar usuários. O console web também contém uma opção para visualizar dados ao vivo para um teste em execução, no qual ele se inscreve no tópico correspondente no AWS IoT Core.

O URL do console web é o nome do domínio de CloudFront distribuição que pode ser encontrado nas CloudFormation saídas como Console. Depois de iniciar o CloudFormation modelo, você também receberá um e-mail contendo o URL do console web e a senha de uso único para fazer login nele.

Servidor MCP (opcional)

O servidor opcional Model Context Protocol (MCP) fornece uma interface adicional para ferramentas de desenvolvimento de IA acessarem e analisarem dados de teste de carga por meio de interações de linguagem natural. Esse componente só será implantado se você selecionar a opção MCP Server durante a implantação da solução.

O MCP Server permite que agentes de IA consultem resultados de testes, analisem métricas de desempenho e obtenham insights sobre seus dados de teste de carga usando ferramentas como Amazon Q, Claude e outros assistentes de IA compatíveis com MCP. Para obter informações detalhadas sobre a arquitetura e a configuração do MCP Server, consulte MCP Server nesta seção.

Back-end

O back-end consiste em um pipeline de imagem de contêiner e um mecanismo de teste de carga que você usa para gerar carga para os testes. Você interage com o back-end por meio do front-end. Além disso, as tarefas do Amazon ECS no AWS Fargate lançadas para cada teste são marcadas com um identificador de teste (ID) exclusivo. Essas etiquetas de identificação de teste podem ser usadas para ajudá-lo a monitorar os custos dessa solução. Para obter informações adicionais, consulte as tags de alocação de custos definidas pelo usuário no Guia do usuário do AWS Billing and Cost Management.

Pipeline de imagens de containers

Essa solução usa uma imagem de contêiner criada com o Amazon Linux 2023 como imagem base com a estrutura de teste de carga Taurus instalada. O Taurus é uma estrutura de automação de testes de código aberto que suporta K6 JMeter, Locust e outras ferramentas de teste. A AWS hospeda essa imagem em um repositório público do Amazon Elastic Container Registry (Amazon ECR). A solução usa essa imagem para executar tarefas no Amazon ECS no cluster AWS Fargate.

Para obter mais informações, consulte a seção Personalização da imagem do contêiner deste guia.

Testando a infraestrutura

Além do CloudFormation modelo principal, a solução fornece um modelo regional para lançar os recursos necessários para a execução de testes em várias regiões. A solução armazena esse modelo no Amazon S3 e fornece um link para ele no console web. Cada pilha regional inclui uma VPC, um cluster AWS Fargate e uma função Lambda para processar dados ativos.

Para obter mais informações sobre como implantar a infraestrutura de teste em regiões adicionais, consulte a seção Implantação multirregional deste guia.

Motor de teste de carga

A solução Distributed Load Testing usa o Amazon Elastic Container Service (Amazon ECS) e o AWS Fargate para simular milhares de usuários simultâneos em várias regiões, gerando solicitações HTTP a uma taxa sustentada.

Você define os parâmetros de teste usando o console web incluído. A solução usa esses parâmetros para gerar um cenário de teste JSON e armazená-lo no Amazon S3. Para obter mais informações sobre scripts de teste e parâmetros de teste, consulte Tipos de teste nesta seção.

Uma máquina de estado do AWS Step Functions executa e monitora tarefas do Amazon ECS em um cluster do AWS Fargate. A máquina de estado do AWS Step Functions inclui uma função AWS Lambda, uma função AWS Lambda, uma função AWS Lambda, uma função task-status-checker AWS Lambda executora de tarefas, uma função AWS Lambda canceladora de tarefas e uma função AWS Lambda de análise de resultados. Para obter mais informações sobre o fluxo de trabalho, consulte a seção Testar fluxo de trabalho deste guia. Para obter mais informações sobre os resultados do teste, consulte a seção Resultados do teste deste guia. Para obter mais informações sobre o fluxo de trabalho de cancelamento de teste, consulte a seção Fluxo de trabalho de cancelamento de teste deste guia.

Se você selecionar dados ativos, a solução iniciará uma função real-time-data-publisher Lambda em cada região pelos CloudWatch registros que correspondem às tarefas do Fargate nessa região. Em seguida, a solução processa e publica os dados em um tópico no AWS IoT Core na região em que você lançou a pilha principal. Para obter mais informações, consulte a seção Dados ativos deste guia.

Servidor MCP

A integração opcional do servidor Model Context Protocol (MCP) permite que os agentes de IA acessem e analisem programaticamente seus dados de teste de carga por meio de interações de linguagem natural. Esse componente só será implantado se você selecionar a opção MCP Server durante a implantação da solução.

O MCP Server atua como uma ponte entre as ferramentas de desenvolvimento de IA e sua implantação de DLT, fornecendo uma interface padronizada para análise inteligente dos resultados dos testes de desempenho. A arquitetura integra vários serviços da AWS para criar uma interface segura e escalável para interações com agentes de IA:

AWS AgentCore Gateway

O AWS AgentCore Gateway é um serviço totalmente gerenciado que fornece hospedagem padronizada e gerenciamento de protocolos para servidores MCP. Nessa solução, o AgentCore Gateway serve como o endpoint público ao qual os agentes de IA se conectam ao solicitar acesso aos seus dados de teste de carga.

O serviço lida com toda a comunicação do protocolo MCP, incluindo descoberta de ferramentas, validação do token de autenticação e roteamento de solicitações. AgentCore O Gateway opera como um serviço multilocatário com proteções de segurança integradas contra ameaças comuns a endpoints públicos, ao mesmo tempo em que valida assinaturas e declarações de tokens do Cognito para cada solicitação.

Servidor DLT MCP Lambda

A função Lambda do DLT MCP Server é um componente personalizado sem servidor que processa solicitações MCP de agentes de IA e as traduz em consultas em seus recursos de DLT.

Essa função do Lambda atua como a camada de inteligência da integração MCP, recuperando resultados de testes de tabelas do DynamoDB, acessando artefatos de desempenho armazenados em buckets do S3 e consultando registros para obter informações detalhadas de execução.

CloudWatch A função Lambda implementa padrões de acesso somente para leitura e transforma dados brutos de DLT em formatos estruturados e amigáveis à IA que os agentes podem interpretar e analisar com facilidade.

Integração de autenticação

O sistema de autenticação aproveita sua infraestrutura existente de pool de usuários do Cognito para manter controles de acesso consistentes no console web e nas interfaces do MCP Server.

Essa integração usa autenticação baseada em token OAuth 2.0. Os usuários se autenticam uma vez por meio do processo de login do Cognito e recebem tokens que funcionam tanto para interações de interface de usuário quanto para acesso ao MCP Server. O sistema mantém os mesmos limites de permissão e controles de acesso da interface da web, garantindo que os usuários só possam acessar, por meio de agentes de IA, os mesmos dados de teste de carga que podem acessar pelo console.

Serviços da AWS nesta solução

Os seguintes serviços da AWS estão incluídos nessa solução:

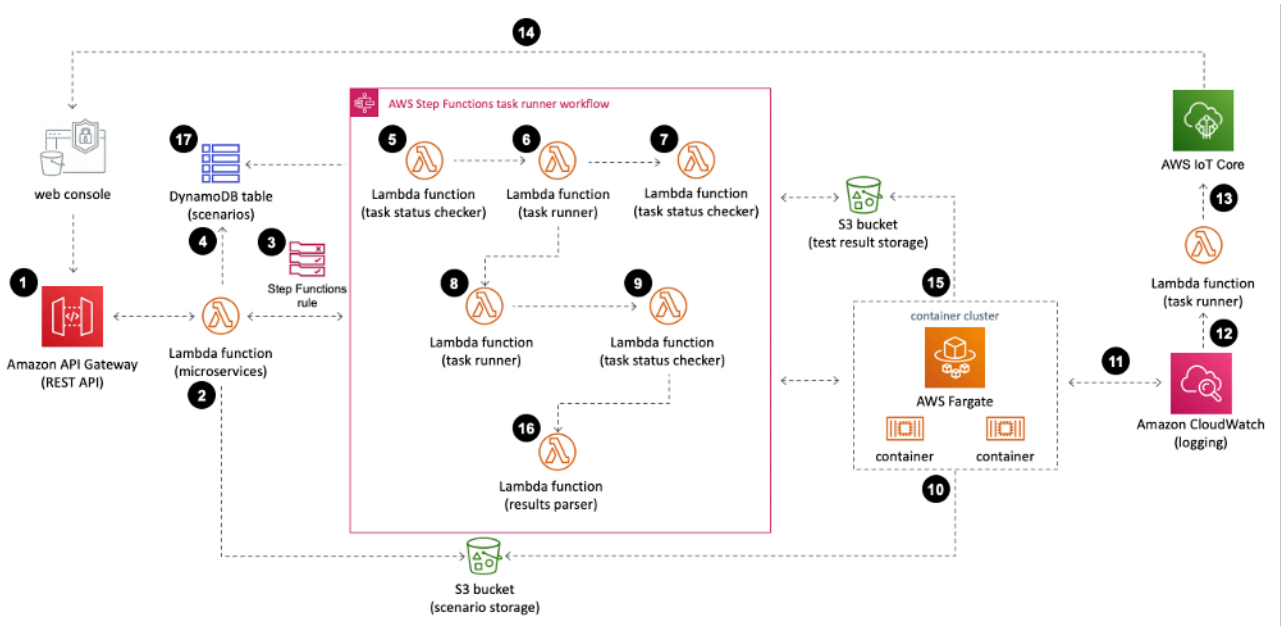
Serviço da AWS	Description
Amazon API Gateway	Principal. Hospeda endpoints da API REST na solução.
AWS CloudFormation	Principal. Gerencia implantações para a infraestrutura da solução.
Amazon CloudFront	Principal. Oferece o conteúdo da web hospedado no Amazon S3.
Amazon CloudWatch	Principal. Armazena os registros e as métricas da solução.
Amazon Cognito	Principal. Lida com o gerenciamento e a autenticação de usuários para a API.
Amazon DynamoDB	Principal. Armazena informações de implantação e detalhes e resultados do cenário de testes.
Amazon Elastic Container Service	Principal. Implanta e gerencia tarefas independentes do Amazon ECS em contêineres do AWS Fargate.
AWS Fargate	Principal. Hospeda os contêineres Amazon ECS da solução

Serviço da AWS	Description
AWS Identity and Access Management	Principal. Lida com o gerenciamento de funções e permissões do usuário.
AWS Lambda	Principal. Fornece lógica para APIs implementação, análise de resultados de testes e lançamento de workers/leader tarefas.
AWS Step Functions	Principal. Orquestra o provisionamento de contêineres do Amazon ECS em tarefas do AWS Fargate nas regiões especificadas
AWS Amplify	Suporte. Fornece um console web desenvolvido pelo AWS Amplify.
CloudWatch Eventos da Amazon	Suporte. Agenda os testes para que comecem automaticamente em uma data especificada ou em datas recorrentes.
Amazon Elastic Container Registry	Suporte. Hospeda a imagem do contêiner em um repositório ECR público.
AWS IoT Core	Suporte. Permite a visualização de dados ao vivo para um teste em execução ao se inscrever no tópico correspondente no AWS IoT Core.
AWS Systems Manager	Suporte. Fornece monitoramento de recursos em nível de aplicativo e visualização de operações de recursos e dados de custos.
Amazon S3	Suporte. Hospeda o conteúdo estático da web, registros, métricas e dados de testes.
Amazon Virtual Private Cloud	Suporte. Contém os contêineres Amazon ECS da solução em execução no AWS Fargate.
Amazon Bedrock AgentCore	Suporte, opcional. Hospeda o servidor opcional Remote Model Context Protocol (MCP) da solução para integração do agente de IA com a API.

Como funciona o teste de carga distribuída na AWS

A análise detalhada a seguir mostra as etapas envolvidas na execução de um cenário de teste.

Testar fluxos de trabalho



1. Você usa o console web para enviar um cenário de teste que inclui os detalhes da configuração para a API da solução.
2. A configuração do cenário de teste é carregada no Amazon Simple Storage Service (Amazon S3) como um arquivo JSON (). `s3://<bucket-name>/test-scenarios/<$TEST_ID>/<$TEST_ID>.json`
3. Uma máquina de estado do AWS Step Functions é executada usando o ID de teste, a contagem de tarefas, o tipo de teste e o tipo de arquivo como entrada da máquina de estado do AWS Step Functions. Se o teste for agendado, ele primeiro criará uma regra de CloudWatch eventos, que acionará o AWS Step Functions na data especificada. Para obter mais detalhes sobre o fluxo de trabalho de agendamento, consulte a seção Fluxo de trabalho de agendamento de testes deste guia.
4. Os detalhes da configuração são armazenados na tabela de cenários do Amazon DynamoDB.
5. No fluxo de trabalho do executor de tarefas do AWS Step Functions, a função task-status-checker AWS Lambda verifica se as tarefas do Amazon Elastic Container Service (Amazon ECS) já estão em execução com o mesmo ID de teste. Se tarefas com o mesmo ID de teste forem encontradas em execução, isso causará um erro. Se não houver tarefas do Amazon ECS em execução no cluster do AWS Fargate, a função retornará o ID do teste, a contagem de tarefas e o tipo de teste.
6. A função executora de tarefas do AWS Lambda obtém os detalhes da tarefa da etapa anterior e executa as tarefas de trabalho do Amazon ECS no cluster AWS Fargate. A API do Amazon ECS usa a RunTask ação para executar as tarefas do trabalhador. Essas tarefas de trabalho são iniciadas e, em seguida, aguardam uma mensagem inicial da tarefa líder para iniciar o teste. A

RunTask ação é limitada a 10 tarefas por definição. Se a contagem de tarefas for maior que 10, a definição da tarefa será executada várias vezes até que todas as tarefas do trabalhador tenham sido iniciadas. A função também gera um prefixo para distinguir o teste atual na função AWS Lambda de análise de resultados.

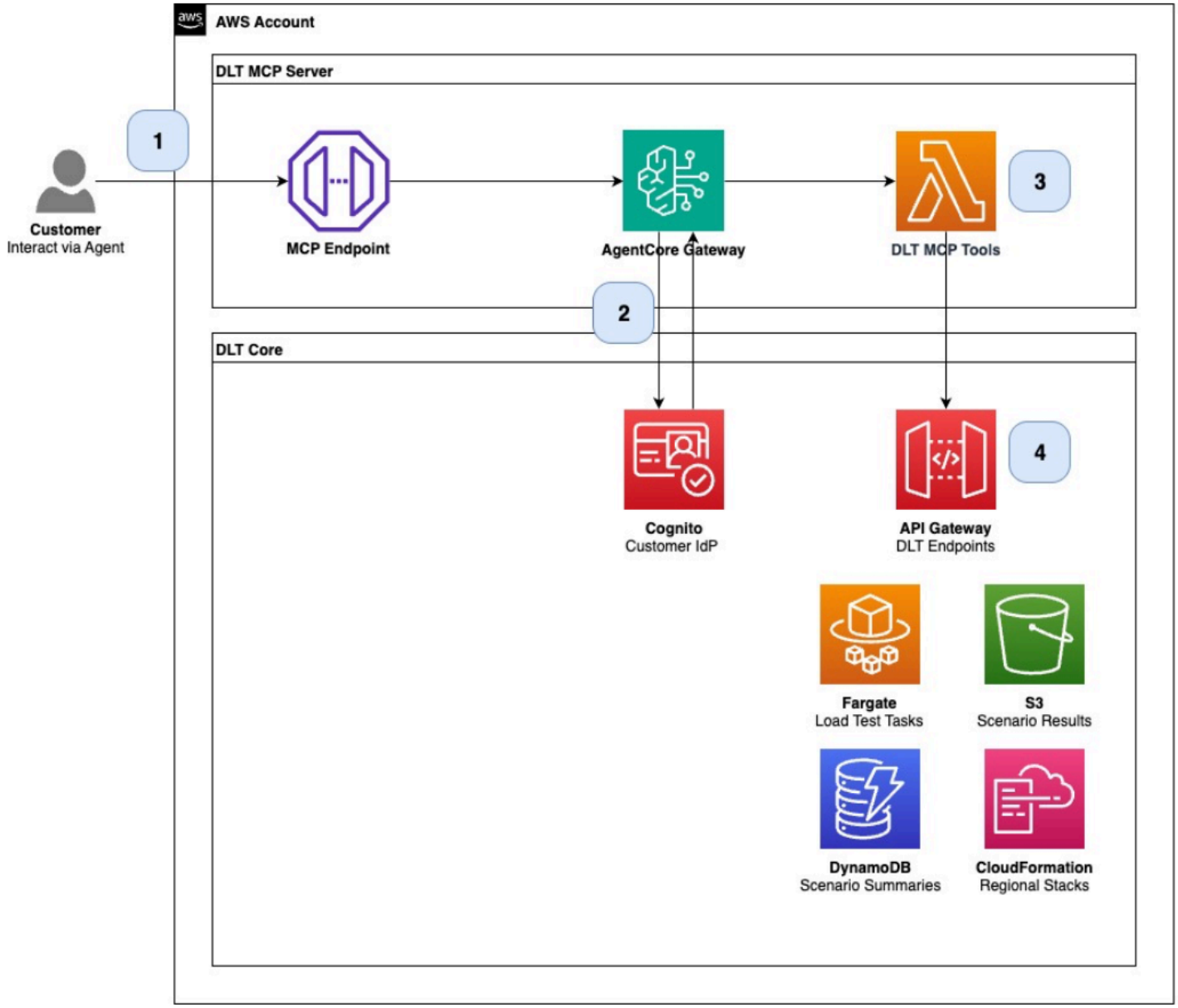
7. A função task-status-checker AWS Lambda verifica se todas as tarefas de trabalho do Amazon ECS estão sendo executadas com o mesmo ID de teste. Se as tarefas ainda estiverem sendo provisionadas, ele aguardará um minuto e verificará novamente. Depois que todas as tarefas do Amazon ECS estão em execução, ele retorna o ID do teste, a contagem de tarefas, o tipo de teste, todas as tarefas IDs e prefixos e os passa para a função executora de tarefas.
8. A função executora de tarefas do AWS Lambda é executada novamente, desta vez lançando uma única tarefa do Amazon ECS para atuar como o nó líder. Essa tarefa do ECS envia uma mensagem de início de teste para cada uma das tarefas do trabalhador para iniciar os testes simultaneamente.
9. A função task-status-checker AWS Lambda novamente verifica se as tarefas do Amazon ECS estão sendo executadas com o mesmo ID de teste. Se as tarefas ainda estiverem em execução, ele espera por um minuto e verifica novamente. Quando não há tarefas em execução do Amazon ECS, ele retorna o ID do teste, a contagem de tarefas, o tipo de teste e o prefixo.
10. Quando a função AWS Lambda executora de tarefas executa as tarefas do Amazon ECS no cluster AWS Fargate, cada tarefa baixa a configuração do teste do Amazon S3 e inicia o teste.
11. Depois que os testes são executados, o tempo médio de resposta, o número de usuários simultâneos, o número de solicitações bem-sucedidas e o número de solicitações malsucedidas para cada tarefa são registrados na Amazon CloudWatch e podem ser visualizados em um CloudWatch painel.
12. Se você incluiu dados ativos no teste, a solução filtra os resultados do teste em tempo real CloudWatch usando um filtro de assinatura. Em seguida, a solução passa os dados para uma função Lambda.
13. A função Lambda então estrutura os dados recebidos e os publica em um tópico do AWS IoT Core.
14. O console web se inscreve no tópico do AWS IoT Core para o teste e recebe os dados publicados no tópico para representar graficamente os dados em tempo real enquanto o teste está sendo executado.
15. Quando o teste é concluído, as imagens do contêiner exportam um relatório detalhado como um arquivo XML para o Amazon S3. Cada arquivo recebe um UUID para o nome do arquivo. Por exemplo, s3://dlte-bucket/test-scenarios/ <$TEST_ID> /results/ <$UUID> .json.

16. Quando os arquivos XML são carregados para o Amazon S3, a função AWS Lambda do analisador de resultados lê os resultados nos arquivos XML começando com o prefixo e analisa e agrega todos os resultados em um resultado resumido.
17. A função AWS Lambda do analisador de resultados grava o resultado agregado em uma tabela do Amazon DynamoDB.

Fluxo de trabalho do servidor MCP (opcional)

Se você implantar a integração opcional do MCP Server, os agentes de IA poderão acessar e analisar seus dados de teste de carga por meio do seguinte fluxo de trabalho:

Arquitetura do servidor MCP



1. Interação com o cliente — O cliente interage com o MCP da DLT por meio do MCP Endpoint hospedado pelo AWS Gateway. AgentCore Os agentes de IA se conectam a esse endpoint para solicitar acesso aos dados de teste de carga.
2. Autorização - O AgentCore gateway processa a autorização em relação ao cliente do aplicativo de pool de usuários do Solution Cognito. O gateway valida o token Cognito do usuário para garantir que ele tenha permissão para acessar o servidor DLT MCP. Os usuários autorizados recebem acesso com o acesso à ferramenta do agente limitado a operações somente de leitura.

3. Especificação da ferramenta - O AgentCore gateway se conecta à função Lambda do DLT MCP Server. Uma especificação de ferramenta define as ferramentas disponíveis que os agentes de IA podem usar para interagir com seus dados de teste de carga.
4. Acesso à API somente para leitura - A função Lambda tem como escopo o acesso somente para leitura à API por meio dos endpoints existentes do DLT API Gateway. A função fornece quatro operações principais:
   - Listar cenários - Recupere uma lista de cenários de teste da tabela de cenários do DynamoDB
   - Obtenha resultados de testes de cenários — Acesse resultados de testes detalhados para cenários específicos do DynamoDB e do S3
   - Obtenha executores de teste de carga do Fargate - consulte informações sobre a execução de tarefas do Fargate no cluster do ECS
   - Obtenha pilhas regionais disponíveis - Recupere informações sobre a infraestrutura regional implantada em CloudFormation

A integração do MCP Server aproveita a infraestrutura DLT existente (API Gateway, Cognito, DynamoDB, S3) para fornecer acesso seguro e somente leitura aos dados de teste para análises e insights baseados em IA.

Considerações sobre design

Esta seção descreve decisões de projeto e opções de configuração importantes para a solução Distributed Load Testing on AWS, incluindo aplicativos compatíveis, tipos de teste, opções de agendamento e considerações de implantação.

Aplicações compatíveis

Essa solução oferece suporte ao teste de aplicativos baseados em nuvem e aplicativos locais, desde que você tenha conectividade de rede da sua conta da AWS com seu aplicativo. A solução APIs suporta o uso de protocolos HTTP ou HTTPS.

Tipos de teste

O teste de carga distribuído na AWS oferece suporte a vários tipos de teste: testes simples de endpoint HTTP JMeter, K6 e Locust.

Testes simples de endpoint HTTP

O console web fornece uma interface de configuração de endpoint HTTP que permite testar qualquer endpoint HTTP ou HTTPS sem escrever scripts personalizados. Você define o URL do endpoint, seleciona o método HTTP (GET, POST, PUT, DELETE etc.) em um menu suspenso e, opcionalmente, adiciona cabeçalhos e cargas de corpo de solicitação personalizados. Essa configuração permite que você teste APIs com tokens de autorização personalizados, tipos de conteúdo ou quaisquer outros cabeçalhos HTTP e corpos de solicitação exigidos pelo seu aplicativo.

JMeter testes

Ao criar um cenário de teste usando o console web, você pode carregar um script JMeter de teste. A solução carrega o script no bucket S3 dos cenários. Quando as tarefas do Amazon ECS são executadas, elas baixam o JMeter script do S3 e executam o teste.

> **Important**
>
> Embora seu JMeter script possa definir simultaneidade (usuários virtuais), taxas de transação (TPS), tempos de aceleração e outros parâmetros de carregamento, a solução substituirá essas configurações pelos valores especificados na tela Traffic Shape durante a criação do teste. A configuração do Traffic Shape controla a contagem de tarefas, a simultaneidade (usuários virtuais por tarefa), a duração do aumento e a duração da espera para a execução do teste.

Se você tiver arquivos JMeter de entrada, poderá compactá-los junto com o JMeter script. Você pode escolher o arquivo zip ao criar um cenário de teste.

Se você quiser incluir plug-ins, todos os arquivos.jar incluídos em um subdiretório /plugins no arquivo zip incluído serão copiados para o diretório de JMeter extensões e estarão disponíveis para teste de carga.

> **Note**
>
> Se você incluir arquivos JMeter de entrada com seu arquivo de JMeter script, deverá incluir o caminho relativo dos arquivos de entrada em seu arquivo de JMeter script. Além disso, os arquivos de entrada devem estar no caminho relativo. Por exemplo, quando os arquivos JMeter de entrada e o arquivo de script estiverem em/home/user directory and you refer to the input files in the JMeter script file, the path of input files must be ./INPUT_FILES. If you

use /home/user/INPUT_FILES, o teste falhará porque não conseguirá encontrar os arquivos de entrada.

Se você incluir JMeter plug-ins, os arquivos.jar deverão ser agrupados em um subdiretório chamado /plugins na raiz do arquivo zip. Em relação à raiz do arquivo zip, o caminho para os arquivos jar deve ser. /Plugins/bundled_plugin.jar.

Para obter mais informações sobre como usar JMeter scripts, consulte o Manual do JMeter usuário.

Testes K6

A solução oferece suporte a testes baseados na estrutura K6. O K6 é lançado sob a licença AGPL-3.0. A solução exibe uma mensagem de confirmação de licença ao criar um novo teste K6. Você pode carregar o arquivo de teste K6 junto com todos os arquivos de entrada necessários em um arquivo de arquivamento.

Important

Embora seu script K6 possa definir simultaneidade (usuários virtuais), estágios, limites e outros parâmetros de carga, a solução substituirá essas configurações pelos valores especificados na tela Traffic Shape durante a criação do teste. A configuração do Traffic Shape controla a contagem de tarefas, a simultaneidade (usuários virtuais por tarefa), a duração do aumento e a duração da espera para a execução do teste.

testes de gafanhotos

A solução oferece suporte a testes baseados na estrutura Locust. Você pode carregar o arquivo de teste do Locust junto com todos os arquivos de entrada necessários em um arquivo compactado.

Important

Embora seu script Locust possa definir simultaneidade (contagem de usuários), taxa de geração e outros parâmetros de carga, a solução substituirá essas configurações pelos valores que você especificar na tela do Traffic Shape durante a criação do teste. A configuração do Traffic Shape controla a contagem de tarefas, a simultaneidade (usuários virtuais por tarefa), a duração do aumento e a duração da espera para a execução do teste.

Agendamento de testes

A solução fornece três opções de tempo de execução para executar testes de carga:

- Executar agora - Execute o teste de carga imediatamente após a criação
- Executar uma vez - Execute o teste em uma data e hora específicas no futuro
- Executar em um cronograma - Crie testes recorrentes usando expressões cron para definir o cronograma

Ao selecionar Executar uma vez, você especifica o tempo de execução no formato de 24 horas e a data de execução em que o teste de carga deve começar a ser executado.

Ao selecionar Executar em uma programação, você pode inserir manualmente uma expressão cron ou selecionar padrões cron comuns (como a cada hora, diariamente em um horário específico, dias da semana ou mensalmente). A expressão cron usa um formato de agendamento refinado com campos para minutos, horas, dia do mês, mês, dia da semana e ano. Você também deve especificar uma data de expiração, que define quando o teste agendado deve parar de ser executado. Para obter mais informações sobre como o agendamento funciona, consulte a seção Fluxo de trabalho de agendamento de testes deste guia.

> **Note**
>
> - Duração do teste: considere a duração total dos testes ao agendar. Por exemplo, um teste com tempo de aceleração de 10 minutos e tempo de espera de 40 minutos levará aproximadamente 80 minutos para ser concluído.
> - Intervalo mínimo: garanta que o intervalo entre os testes programados seja maior do que a duração estimada do teste. Por exemplo, se o teste levar cerca de 80 minutos, programe-o para ser executado no máximo a cada 3 horas.
> - Limitação horária: o sistema não permite que os testes sejam agendados com apenas uma hora de diferença, mesmo que a duração estimada do teste seja inferior a uma hora.

Testes simultâneos

Essa solução cria um CloudWatch painel da Amazon para cada teste que exibe a saída combinada de todas as tarefas executadas no cluster do Amazon ECS em tempo real. O CloudWatch painel

mostra o tempo médio de resposta, o número de usuários simultâneos, o número de solicitações bem-sucedidas e o número de solicitações malsucedidas. A solução agrega cada métrica por segundo e atualiza o painel a cada minuto.

Gerenciamento de usuários

Durante a configuração inicial, você fornece um nome de usuário e endereço de e-mail que o Amazon Cognito usa para conceder acesso ao console web da solução. O console não fornece administração de usuários. Para adicionar mais usuários, você deve usar o console do Amazon Cognito. Para obter mais informações, consulte Gerenciamento de usuários em grupos de usuários no Guia do desenvolvedor do Amazon Cognito.

Para migrar usuários existentes para grupos de usuários do Amazon Cognito, consulte o blog da AWS Abordagens para migrar usuários para grupos de usuários do Amazon Cognito.

Implantação regional

Essa solução usa o Amazon Cognito, que está disponível somente em regiões específicas da AWS. Portanto, você deve implantar essa solução em uma região onde o Amazon Cognito esteja disponível. Para obter a disponibilidade de serviços mais atual por região, consulte a Lista de serviços regionais da AWS.

Planeje a implantação

Esta seção descreve custo, segurança, regiões suportadas, cotas e outras considerações que você deve analisar antes de implantar a solução.

Custo

Você é responsável pelo custo dos serviços da AWS usados durante a execução dessa solução. O custo total depende do número de testes de carga executados, da duração desses testes e da quantidade de dados gerados. A partir dessa revisão, o custo estimado para executar essa solução com configurações padrão na região Leste dos EUA (Norte da Virgínia) é de aproximadamente 30,90 USD por mês.

A tabela a seguir fornece um exemplo de detalhamento de custos para implantar essa solução com os parâmetros padrão na região Leste dos EUA (Norte da Virgínia) por um mês.

Serviço da AWS	Dimensões	Custo [USD]
AWS Fargate	10 tarefas sob demanda (usando 2 V CPUs e 4 GB de memória) em execução por 30 horas	$29,62
Amazon DynamoDB	1.000 unidades de capacidade de gravação sob demanda 1.000 unidades de capacidade de leitura sob demanda	$0,0015
AWS Lambda	1.000 solicitações 10 minutos de duração total	$1,25
AWS Step Functions	1.000 transições de estado	0,025 USD
Total:		$30,90 por mês

Os recursos da solução são marcados com Key= SolutionId e SO0062 Value=. Você pode ativar a chave da tag SolutionId seguindo a documentação activating-tags. Depois que a tag for ativada, você poderá criar uma regra de categoria de custo seguindo a documentação para criar categorias de custo. Você pode visualizar o custo incorrido com a solução monitorando o console de categorias de custo e selecionando o nome da categoria de custo.

Recomendamos criar um orçamento por meio do AWS Cost Explorer para ajudar a gerenciar custos. Os preços estão sujeitos a alterações. Para obter detalhes completos, consulte a página de preços de cada serviço da AWS usado nesta solução.

> **Note**
>
> A configuração padrão da tarefa usa 2 v CPUs e 4 GB de memória por tarefa. Se seus testes de carga não exigirem esses recursos, você poderá reduzi-los para reduzir os custos. Por outro lado, você pode aumentar os recursos para oferecer suporte a uma maior simultaneidade por tarefa. Para obter mais informações, consulte a seção Aumentar os recursos do contêiner neste guia.

> **Note**
>
> Essa solução oferece a opção de incluir dados ativos ao executar um teste. Esse recurso requer uma função adicional do AWS Lambda e um tópico do AWS IoT Core que incorrem em custos extras.

Custos adicionais do MCP Server (opcional)

A tabela a seguir fornece um detalhamento dos custos da integração do MCP Server com os preços na região Leste dos EUA (Norte da Virgínia) por um mês.

Componente do serviço	Dimensões	Custo [USD]
AgentCore Gateway - Indexação de ferramentas	10 ferramentas × 0,02 USD por 100 ferramentas	$0,002
AgentCore Gateway - API de pesquisa	10.000 interações × 0,025 USD por 1.000	$0,25

Componente do serviço	Dimensões	Custo [USD]
AgentCore Gateway - Invocações de API	50.000 invocações × 0,005 USD por 1.000	$0,25
AWS Lambda Função	Variável com base no uso (cargas de trabalho típicas)	$5,00 - $20,00
Custo adicional total estimado:		$5,50 - $20,50 por mês

Os preços estão sujeitos a alterações. Para obter detalhes completos sobre os preços do AgentCore Gateway, consulte os preços do Amazon Bedrock (seção AgentCore Gateway). Para obter os preços do Lambda, consulte os preços do AWS Lambda.

Segurança

Quando você cria sistemas na infraestrutura da AWS, as responsabilidades de segurança são compartilhadas entre você e a AWS. Esse modelo de responsabilidade compartilhada reduz seus encargos operacionais, pois a AWS opera, gerencia e controla os componentes, incluindo o sistema operacional do host, a camada de virtualização e a segurança física das instalações onde os serviços operam. Para obter mais informações sobre segurança da AWS, visite Segurança da Nuvem AWS.

Perfis do IAM

As funções do AWS Identity and Access Management (IAM) permitem que os clientes atribuam políticas e permissões de acesso granulares a serviços e usuários na nuvem da AWS. Essa solução cria funções do IAM que concedem às funções do AWS Lambda da solução acesso para criar recursos regionais.

Amazon CloudFront

Essa solução implanta uma interface de usuário da web hospedada em um bucket do Amazon S3, que é distribuído pela Amazon. CloudFront Para ajudar a reduzir a latência e melhorar a segurança, essa solução inclui uma CloudFront distribuição com uma identidade de acesso de origem, que é um CloudFront usuário que fornece acesso público ao conteúdo do bucket do site da solução. Por padrão, a CloudFront distribuição usa o TLS 1.2 para impor o nível mais alto de protocolo de

segurança. Para obter mais informações, consulte Restringir o acesso a uma origem do Amazon S3 no CloudFront Amazon Developer Guide.

CloudFront ativa mitigações de segurança adicionais para acrescentar cabeçalhos de segurança HTTP à resposta de cada visualizador. Para obter mais informações, consulte Adicionar ou remover cabeçalhos HTTP nas CloudFront respostas.

Essa solução usa o CloudFront certificado padrão, que tem um protocolo de segurança mínimo suportado de TLS v1.0. Para impor o uso do TLS v1.2 ou do TLS v1.3, você deve usar um certificado SSL personalizado em vez do certificado padrão. CloudFront Para obter mais informações, consulte Como configuro minha CloudFront distribuição para usar um SSL/TLS certificado.

Amazon API Gateway

Essa solução implanta endpoints do Amazon API Gateway otimizados para borda RESTful APIs para fornecer a funcionalidade de teste de carga usando o endpoint padrão do API Gateway em vez de um domínio personalizado. Para otimizar a borda APIs usando o endpoint padrão, o API Gateway usa a política de segurança TLS-1-0. Para obter mais informações, consulte Como trabalhar com REST APIs no Guia do desenvolvedor do Amazon API Gateway.

Essa solução usa o certificado padrão do API Gateway, que tem um protocolo de segurança mínimo suportado de TLS v1.0. Para impor o uso do TLS v1.2 ou do TLS v1.3, você deve usar um domínio personalizado com um certificado SSL personalizado em vez do certificado padrão do API Gateway. Para obter mais informações, consulte Configuração de nomes de domínio personalizados para REST APIs.

Grupo de segurança AWS Fargate

Por padrão, essa solução abre a regra de saída do grupo de segurança do AWS Fargate para o público. Se você quiser impedir que o AWS Fargate envie tráfego para qualquer lugar, altere a regra de saída para um roteamento entre domínios sem classe (CIDR) específico.

Esse grupo de segurança também inclui uma regra de entrada que permite tráfego local na porta 50.000 para qualquer fonte que pertença ao mesmo grupo de segurança. Isso é usado para permitir que os contêineres se comuniquem entre si.

Amazon VPC

VPC: uma nuvem privada virtual (VPC) baseada no serviço Amazon VPC oferece uma rede privada e logicamente isolada na nuvem da AWS.

Você pode especificar sua própria VPC nos CloudFormation parâmetros da AWS durante a implantação. A VPC é usada exclusivamente pelas tarefas do ECS que geram carga; o console web e a API não são implantados nessa VPC. Se você não especificar uma VPC existente, a solução criará uma nova VPC com a configuração de rede necessária. Se você optar por usar uma VPC existente, ela deverá atender aos seguintes requisitos para executar as tarefas de teste de carga com êxito.

Requisitos da VPC

Os requisitos mínimos para que uma VPC seja usada com testes de carga distribuída na AWS estão listados abaixo.

- O VPC deve conter pelo menos dois AZs
- A VPC deve conter pelo menos duas sub-redes, cada uma em uma AZ separada
- As sub-redes VPC podem ser públicas ou privadas, mas devem usar a mesma configuração (pública OU privada)
- A VPC deve fornecer acesso aos endpoints para ECR, CloudWatch Logs, S3 e IoT Core.
- A VPC deve fornecer acesso aos serviços visados pelos testes de carga.

> **Note**
>
> Se você não tiver uma VPC que atenda a esses critérios, poderá criar uma VPC com o assistente de VPC rapidamente. Para obter mais informações, consulte Criar uma VPC.

As sub-redes públicas podem atender a esses requisitos incluindo o seguinte:

- Um gateway de internet conectado à VPC
- Uma rota para o gateway da Internet (0.0.0.0/0)

As sub-redes privadas podem atender a esses requisitos por meio do uso de gateways NAT ou endpoints VPC, conforme descrito abaixo.

Opção 1: NAT Gateway

- Implemente um gateway NAT em cada AZ com sub-redes privadas

- Configurar tabelas de rotas para rotear o tráfego vinculado à Internet (0.0.0.0/0) por meio do NAT Gateway

Opção 2: VPC Endpoints

Crie os seguintes endpoints de VPC em sua VPC:

- Endpoint da API Amazon ECR: `com.amazonaws.<region>.ecr.api`
- Endpoint Amazon ECR DKR: `com.amazonaws.<region>.ecr.dkr`
- Endpoint CloudWatch do Amazon Logs: `com.amazonaws.<region>.logs`
- Endpoint do Amazon S3 Gateway: `com.amazonaws.<region>.s3`
- Endpoint do AWS IoT Core (necessário se estiver usando os gráficos de dados ao vivo) `com.amazonaws.<region>.iot.data`

Outras configurações de VPC também podem funcionar.

> **Important**
>
> O grupo de segurança conectado a cada interface do VPC endpoint deve permitir o tráfego TCP de entrada na porta 443 do grupo de segurança de tarefas do ECS.

Configuração do grupo de segurança

Durante a implantação, a solução criará um grupo de segurança em sua VPC para permitir o seguinte tráfego com tarefas no cluster do ECS:

- Todo o tráfego de saída
- Tráfego de entrada na porta 50000 de outras tarefas no mesmo grupo de segurança, para facilitar a coordenação entre as tarefas do trabalhador e do líder.

Teste de estresse de rede

Você é responsável por usar essa solução de acordo com a política de teste de estresse de rede. Essa política abrange situações como quando você planeja executar testes de rede de alto volume diretamente de suas instâncias do Amazon EC2 para outros locais, como outras instâncias do

Amazon EC2, propriedades/serviços da AWS ou endpoints externos. Esses testes às vezes são chamados de testes de estresse, testes de carga ou testes de dia de jogo. A maioria dos testes com clientes não se enquadra nessa política; no entanto, consulte essa política se você acredita que gerará tráfego que se sustenta, em conjunto, por mais de 1 minuto, mais de 1 Gbps (1 bilhão de bits por segundo) ou mais de 1 Gpps (1 bilhão de pacotes por segundo).

Restringindo o acesso à interface pública do usuário

Para restringir o acesso à interface de usuário pública além dos mecanismos de autenticação e autorização fornecidos pelo IAM e pelo Amazon Cognito, use a solução de automação de segurança AWS WAF (web application firewall).

Essa solução implanta automaticamente um conjunto de regras do AWS WAF que filtram ataques comuns baseados na web. Os usuários podem selecionar recursos de proteção pré-configurados que definem as regras incluídas em uma lista de controle de acesso à web (web ACL) do AWS WAF.

Segurança do servidor MCP (opcional)

Se você implantar a integração opcional do MCP Server, a solução usará o AWS AgentCore Gateway para fornecer acesso seguro aos dados de teste de carga para agentes de IA. AgentCore O Gateway valida os tokens de autenticação do Amazon Cognito para cada solicitação, garantindo que somente usuários autorizados possam acessar o MCP Server. A função Lambda do MCP Server implementa padrões de acesso somente para leitura, impedindo que agentes de IA modifiquem as configurações ou os resultados dos testes. Todas as interações do MCP Server usam os mesmos limites de permissão e controles de acesso do console web.

Regiões da AWS compatíveis

Essa solução usa o serviço Amazon Cognito, que atualmente não está disponível em todas as regiões da AWS. Para obter a disponibilidade mais atual dos serviços da AWS por região, consulte a Lista de serviços regionais da AWS.

O teste de carga distribuído na AWS está disponível nas seguintes regiões da AWS:

Nome da região	
Leste dos EUA (Ohio)	Ásia-Pacífico (Tóquio)

Nome da região	
Leste dos EUA (Norte da Virgínia)	Canadá (Central)
Oeste dos EUA (Norte da Califórnia)	Europa (Frankfurt)
Oeste dos EUA (Oregon)	Europa (Irlanda)
Ásia-Pacífico (Mumbai)	Europa (Londres)
Ásia-Pacífico (Seul)	Europa (Paris)
Ásia-Pacífico (Singapura)	Europa (Estocolmo)
Ásia-Pacífico (Sydney)	América do Sul (São Paulo)

Regiões da AWS compatíveis com o servidor MCP (opcional)

Se você planeja implantar a integração opcional do MCP Server, você deve implantar a solução em uma região da AWS onde o AWS AgentCore Gateway esteja disponível. O recurso MCP Server está disponível somente nas seguintes regiões da AWS:

Nome da região	Código da região
Leste dos EUA (Norte da Virgínia)	us-east-1
Oeste dos EUA (Oregon)	us-west-2
Ásia-Pacífico (Singapura)	ap-southeast-1
Ásia-Pacífico (Sydney)	ap-southeast-2
Ásia-Pacífico (Tóquio)	ap-northeast-1
Europa (Frankfurt)	eu-central-1
Europa (Irlanda)	eu-west-1
Europa (Londres)	eu-west-2

Nome da região	Código da região
Europa (Paris)	eu-west-3

Para obter a disponibilidade mais atual do AWS AgentCore Gateway por região, consulte os endpoints e cotas do AWS AgentCore Gateway no Guia do desenvolvedor do AWS AgentCore Gateway.

Cotas

Service quotas, ou limites, representam o máximo de recursos ou operações de serviço permitidos em uma conta AWS.

Cotas para serviços da AWS nesta solução

Verifique se você tem cota suficiente para cada um dos serviços implementados nessa solução. Para obter mais informações, consulte Cotas dos serviços da AWS.

Use os links a seguir para acessar a página desse serviço. Para visualizar as cotas de serviço para todos os serviços da AWS na documentação sem alternar páginas, consulte as informações na página Endpoints e cotas de serviços no PDF.

CloudFormation Cotas da AWS

Sua conta da AWS tem CloudFormation cotas da AWS que você deve conhecer ao lançar a pilha nesta solução. Ao compreender essas cotas, você pode evitar erros de limitação que o impediriam de implantar essa solução com êxito. Para obter mais informações, consulte as CloudFormation cotas da AWS no Guia do CloudFormation usuário da AWS.

Cotas de teste de carga

O número máximo de tarefas que podem ser executadas no Amazon ECS usando o tipo de execução do AWS Fargate é baseado no tamanho da vCPU das tarefas. O tamanho padrão da tarefa no teste de carga distribuída na AWS é de 2 vCPU. Para ver as cotas padrão atuais, consulte as cotas de serviço do Amazon ECS. As cotas da conta corrente podem ser diferentes das cotas listadas. Para verificar as cotas específicas de uma conta, verifique a cota de serviço para a contagem de recursos de vCPU sob demanda do Fargate no AWS Management Console. Para

obter instruções sobre como solicitar um aumento, consulte as cotas de serviços da AWS no Guia de referência geral da AWS.

A imagem do contêiner Amazon Linux 2023 (com o Taurus instalado) não limita as conexões simultâneas por tarefa, mas isso não significa que ela possa suportar um número ilimitado de usuários. Para determinar o número de usuários simultâneos que os contêineres podem gerar para um teste, consulte a seção Determinar o número de usuários deste guia.

> **Note**
>
> O limite recomendado para usuários simultâneos com base nas configurações padrão é de 200 usuários.

Testes simultâneos

Essa solução cria um CloudWatch painel da Amazon para cada teste que exibe a saída combinada de todas as tarefas executadas no cluster do Amazon ECS em tempo real. O CloudWatch painel mostra o tempo médio de resposta, o número de usuários simultâneos, o número de solicitações bem-sucedidas e o número de solicitações malsucedidas. A solução agrega cada métrica por segundo e atualiza o painel a cada minuto.

Política de testes do Amazon EC2

Você não precisa da aprovação da AWS para executar testes de carga usando essa solução, desde que seu tráfego de rede permaneça abaixo de 1 Gbps. Se seu teste gerar mais de 1 Gbps, entre em contato com a AWS. Para obter mais informações, consulte a Política de testes do Amazon EC2.

Política de teste CloudFront de carga da Amazon

Se você planeja testar a carga de um CloudFront endpoint, consulte as diretrizes de teste de carga no Amazon CloudFront Developer Guide. Também recomendamos distribuir o tráfego em várias tarefas e regiões. Forneça pelo menos 30 minutos de tempo de aceleração para o teste de carga. Para testes de carga que enviam mais de 500.000 solicitações por segundo ou exigem dados de mais de 300 Gbps, recomendamos primeiro obter uma pré-aprovação para enviar o tráfego. CloudFront pode limitar o tráfego de teste de carga não aprovado que afeta CloudFront a disponibilidade do serviço.

Monitorando a solução após a implantação

Depois de implantar a solução, recomendamos monitorar continuamente os recursos da solução usando CloudWatch alarmes e métricas da Amazon.

Configurando CloudWatch alarmes

Você pode configurar CloudWatch alarmes para monitorar as principais métricas e receber notificações quando os limites forem excedidos. Considere configurar alarmes para os seguintes recursos:

Métricas CloudFront de distribuição da Amazon

Monitore o desempenho e os erros da CloudFront distribuição. Para obter mais informações, consulte as métricas CloudFront de distribuição no Amazon CloudFront Developer Guide.

Métricas do Amazon API Gateway

Monitore taxas de solicitação de API, latência e erros. Para obter mais informações, consulte as dimensões e métricas do Amazon API Gateway no Guia do desenvolvedor do Amazon API Gateway.

Métricas da função AWS Lambda

Monitore as invocações, a duração, os erros e as limitações da função Lambda para os microsserviços da solução.

Métricas do Amazon ECS e do AWS Fargate

Monitore a utilização da CPU e da memória da tarefa durante os testes de carga para garantir recursos adequados.

Métricas do Amazon DynamoDB

Monitore o consumo de capacidade de leitura e gravação, as solicitações limitadas e a latência.

Contrate um especialista

Contratos de curto prazo do AWS Countdown Premium para testes de carga distribuída na AWS

Nossos engenheiros da AWS fornecem orientação especializada sobre fundamentos de testes de desempenho, desenvolvimento de scripts e análise de resultados. Inscreva-se agora.

Visão geral

Os contratos de curto prazo do AWS Countdown Premium (CDP) fornecem orientação especializada para organizações que realizam testes de desempenho em grande escala. Por meio de um modelo “colaborativodo-it-yourself”, os engenheiros da AWS oferecem supervisão estratégica e conhecimento técnico, enquanto sua equipe mantém a responsabilidade pela execução. Engenheiros especialistas da AWS estão disponíveis dentro de uma semana após a inscrição, sem a necessidade de contratos de longo prazo.

Modelo de serviço

Os engenheiros do CDP trabalham junto com sua equipe para fornecer orientação e supervisão em toda a implementação do teste de desempenho. Essa abordagem direta garante que você receba orientação especializada enquanto desenvolve capacidades internas. O serviço é ideal para organizações com recursos de teste existentes que precisam de experiência especializada em AWS para implementar testes de carga distribuída na AWS de forma eficaz.

O que os engenheiros do CDP oferecem

Os engenheiros do CDP orientam você nos fundamentos dos testes de desempenho e nos testes de carga distribuída na arquitetura da AWS. Eles fornecem orientação sobre JMeter a estrutura de scripts K6 e Locust e o desenvolvimento de scripts de teste, auxiliam na implantação de CloudFormation modelos e avaliam os resultados dos testes com recomendações de otimização de desempenho. O suporte inclui análise de utilização de recursos, alinhamento de melhores práticas e end-to-end orientação desde a configuração inicial até a análise de resultados, permitindo a transferência de conhecimento para sua equipe.

Responsabilidades do cliente

Sua equipe lida com configurações em nível de aplicativo, desenvolvimento de scripts de teste e verificação de cenários de teste. Você mantém a responsabilidade pela execução e pelas operações reais do teste, incluindo todas as atividades de teste antes, durante e depois dos eventos de teste de desempenho.

Benefícios principais

Os contratos de curto prazo do CDP oferecem riscos reduzidos por meio de supervisão especializada, orientação contextual específica para sua carga de trabalho, recomendações de otimização de desempenho, resolução mais rápida de problemas, alinhamento de melhores práticas e suporte abrangente, mantendo a propriedade e o desenvolvimento de capacidades de sua equipe.

Arquiteturas suportadas

O teste de carga distribuído na AWS oferece suporte a testes de aplicativos web APIs, microsserviços e arquiteturas sem servidor em grande escala, aproveitando a solução Distributed Load Testing on AWS. Os recursos de teste vão muito além desses casos de uso comuns, incluindo bancos de dados, TCP/UDP protocolos, diretórios LDAP, servidores de e-mail SMTP e muitos outros sistemas e protocolos que exigem validação de desempenho sob carga.

Conceitos Básicos

Organizações interessadas em contratos de curto prazo da CDP para testes de carga distribuída na AWS podem se inscrever diretamente no site da AWS aqui e selecionar “Implementação de caso de uso” para sua área de foco.

Fora do escopo

O CDP não fornece desenvolvimento de scripts de teste personalizados (somente orientação), gerencia operações de execução de testes nem cria laboratórios ou workshops práticos personalizados. O suporte no local também está fora do escopo.

Implante a solução

O AWS Launch Wizard é o método de implantação recomendado para essa solução. Ele fornece:

- Uma experiência de configuração guiada com painéis de ajuda detalhados em cada etapa
- Uma página centralizada para monitorar a integridade de todas as suas implantações
- Indicação de quando há uma versão mais recente da solução disponível para implantação ou atualização

Como alternativa, você pode implantar a solução diretamente usando um CloudFormation modelo da AWS.

Visão geral do processo de implantação

Antes de implantar a solução, analise o custo, a arquitetura, a segurança e outras considerações discutidas anteriormente neste guia.

Tempo de implantação: aproximadamente 15 minutos para a pilha principal, mais 5 minutos para cada região adicional

> **Note**
>
> Essa solução inclui métricas de coleta de dados para a AWS. Usamos esses dados para entender melhor como os clientes usam essa solução e os serviços e produtos relacionados. A AWS é proprietária dos dados coletados por meio dessa pesquisa. A coleta de dados está sujeita ao Aviso de Privacidade da AWS.

> **Note**
>
> Você é responsável pelo custo dos serviços da AWS usados ao executar essa solução. Para obter mais detalhes, visite a seção Custo neste guia e consulte a página de preços de cada serviço da AWS usado nesta solução.

Implante usando o AWS Launch Wizard

Essa solução apresenta um processo de implantação guiada usando o AWS Launch Wizard. Siga estas etapas para implantar testes de carga distribuídos na AWS em sua conta.

1. Faça login no AWS Management Console e selecione o botão abaixo para iniciar o processo de implantação.

2. Se houver mais de um padrão de implantação disponível para a solução, selecione aquele que é mais aplicável ao seu caso de uso.
3. Selecione uma versão para implantar. A versão mais recente é recomendada.
4. Clique no botão Iniciar assistente de implantação.

Em seguida, você seguirá uma série de etapas para coletar as informações necessárias para implantar a solução. Serão necessários aproximadamente 15 minutos para provisionar os recursos necessários.

Selecione sua implantação na lista de Implantação para ver seu status.

Implemente usando a AWS CloudFormation

Essa solução usa CloudFormation modelos e pilhas da AWS para automatizar sua implantação. Os CloudFormation modelos especificam os recursos da AWS incluídos nessa solução e suas propriedades. A CloudFormation pilha provisiona os recursos descritos nos modelos.

CloudFormation Modelo da AWS

Você pode baixar o CloudFormation modelo dessa solução antes de implantá-la. Essa solução usa CloudFormation a AWS para automatizar a implantação de testes de carga distribuídos na AWS. Ele inclui o seguinte CloudFormation modelo da AWS, que você pode baixar antes da implantação:

distribu

load-testing-on-aws.template - Use esse modelo para iniciar a solução e todos os componentes

associados. A configuração padrão implanta os serviços principais e de suporte encontrados nos serviços da AWS nesta seção de solução, mas você pode personalizar o modelo para atender às suas necessidades específicas.

> **Note**
>
> Os CloudFormation recursos da AWS são criados a partir de construções do AWS Cloud Development Kit (AWS CDK). Se você já implantou essa solução, consulte Atualizar a solução para obter instruções de atualização.

Iniciar a pilha

Siga estas etapas para implantar a solução Distributed Load Testing on AWS em sua conta. Esse CloudFormation modelo automatizado da AWS implanta testes de carga distribuídos na AWS.

1. Faça login no AWS Management Console e selecione o botão para iniciar o CloudFormation modelo.

   Como alternativa, você pode baixar o modelo como ponto de partida para sua própria implementação.
2. Por padrão, o modelo é iniciado na região Leste dos EUA (Norte da Virgínia). Para iniciar essa solução em uma região diferente da AWS, use o seletor de região na barra de navegação do console.

   > **Note**
   >
   > Essa solução usa o Amazon Cognito, que atualmente está disponível somente em regiões específicas da AWS. Portanto, você deve iniciar essa solução em uma região da AWS em que o Amazon Cognito esteja disponível. Para obter a disponibilidade de serviços mais atual por região, consulte a Lista de serviços regionais da AWS.
3. Na página Criar pilha, verifique se o URL de modelo completo é apresentado na caixa de texto Amazon S3 URL e escolha Avançar.
4. Na página Especificar detalhes da pilha, insira um nome para a pilha.

5. Em Parâmetros, revise os parâmetros do modelo e modifique-os conforme necessário. Esta solução usa os valores padrão apresentados a seguir.

Parâmetro	Padrão	Description
Nome do administrador	<Requer entrada>	Nome de usuário do administrador da solução inicial.
E-mail do administrador	*<Requires input>*	Endereço de e-mail do usuário administrador. Após o lançamento, um e-mail será enviado para esse endereço com as instruções de login do console.
ID de VPC existente	<Optional input>	Se você tem uma VPC que deseja usar e já foi criada, insira o ID de uma VPC existente na mesma região em que a pilha foi implantad a. Por exemplo, vpc-1a2b3 c4d5e6f.
Primeira sub-rede existente	<Optional input>	O ID da primeira sub-rede em sua VPC existente. Essa sub-rede precisa de uma rota para a Internet para extrair a imagem do contêiner para executar testes. Por exemplo, subnet-7h8i9j0k.

Parâmetro	Padrão	Description
Segunda sub-rede existente	<Optional input>	O ID da segunda sub-rede dentro da VPC existente. Essa sub-rede precisa de uma rota para a Internet para extrair a imagem do contêiner para executar testes. Por exemplo, subnet-1x2y3z.
Forneça um bloco CIDR válido para a solução criar VPC	192.168.0.0/16	Você pode deixar esse parâmetro em branco se estiver usando uma VPC existente.
Forneça um bloco CIDR válido para a sub-rede A para que a solução crie VPC	192.168.0.0/20	Bloco CIDR para a sub-rede A da VPC do AWS Fargate
Forneça um bloco CIDR válido para a sub-rede B para que a solução crie uma VPC	192.168.16.0/20	Bloco CIDR para a sub-rede B da VPC do AWS Fargate
Forneça um bloco CIDR para permitir o tráfego de saída das tarefas do Fargate	0.0.0.0/0	Bloco CIDR que restringe o acesso de saída ao contêiner do Amazon ECS.
Atualização automática da imagem do contêiner	No	Use automaticamente a imagem mais atualizada e segura até a próxima versão secundária. A seleção No exibirá a imagem conforme lançada originalmente, sem nenhuma atualização de segurança.

Parâmetro	Padrão	Description
Implantar servidor MCP opcional	No	Implante o servidor MCP remoto opcional, usando o AgentCore Gateway para conectar aplicativos de IA ao teste de carga distribuído na AWS.

6. Escolha Avançar.
7. Na página Configurar opções de pilha, selecione Avançar.
8. Na página Revisar, verifique e confirme as configurações. Marque a caixa de seleção confirmando que o modelo criará recursos do AWS Identity and Access Management (IAM).
9. Selecione Create stack (Criar pilha) para implantar a pilha.

Você pode ver o status da pilha no CloudFormation console da AWS na coluna Status. Você deve receber o status CREATE_COMPLETE em aproximadamente 15 minutos.

> **Note**
>
> Além da função primária do AWS Lambda, essa solução inclui a função Lambda de recurso personalizado, que é executada somente durante a configuração inicial ou quando os recursos são atualizados ou excluídos.

Ao executar essa solução, a função Lambda de recurso personalizado fica inativa. No entanto, não exclua essa função, pois ela é necessária para gerenciar os recursos associados.

Implantação em várias regiões

Tempo de implantação: aproximadamente 5 minutos por região

Você pode executar testes em várias regiões.

Quando você implanta a solução Distributed Load Testing, ela cria um CloudFormation modelo regional no bucket S3 dos cenários. O URL desse modelo está listado nas CloudFormation saídas da sua pilha principal sob a chave “RegionalCFTemplate”.

Para executar um teste multirregional, você deve implantar o CloudFormation modelo regional em cada região em que deseja executar o teste.

> **Note**
>
> Cada conta da AWS pode usar somente uma pilha regional por região. Além disso, a pilha regional não pode ser usada na mesma região da pilha principal.

Você pode instalar o modelo regional da seguinte forma:

1. No console web da solução, navegue até Painel no menu à esquerda.
2. Use o ícone da área de transferência para copiar o link do CloudFormation modelo no Amazon S3.
3. Faça login no CloudFormation console da AWS e selecione a região correta.
4. Na página Criar pilha, verifique se o URL de modelo completo é apresentado na caixa de texto Amazon S3 URL e escolha Avançar.
5. Na página Especificar detalhes da pilha, insira um nome para a pilha.
6. Em Parâmetros, revise os parâmetros do modelo e modifique-os conforme necessário. Esta solução usa os valores padrão apresentados a seguir.

Parâmetro	Padrão	Description
ID de VPC existente	<Optional input>	Se você tem uma VPC que deseja usar e já foi criada, insira o ID de uma VPC existente na mesma região em que a pilha foi implantada. Por exemplo, vpc-1a2b3c4d5e6f.
Primeira sub-rede existente	<Optional input>	O ID da primeira sub-rede em sua VPC existente. Essa sub-rede precisa de uma rota para a Internet para extrair a imagem do contêiner para

Parâmetro	Padrão	Description
		executar testes. Por exemplo, subnet-7h8i9j0k.
Segunda sub-rede existente	<Optional input>	O ID da segunda sub-rede dentro da VPC existente. Essa sub-rede precisa de uma rota para a Internet para extrair a imagem do contêiner para executar testes. Por exemplo, subnet-1x2y3z.
Forneça um bloco CIDR válido para a solução criar VPC	192.168.0.0/16	Se você não fornecer valores para uma VPC existente, o bloco CIDR da Amazon VPC criada pela solução conterá o endereço IP do AWS Fargate.
Forneça um bloco CIDR para permitir o tráfego de saída das tarefas do Fargate	0.0.0.0/0	Bloco CIDR que restringe o acesso de saída ao contêiner do Amazon ECS.

7. Escolha Avançar.
8. Na página Configurar opções de pilha, selecione Avançar.
9. Na página Revisar, verifique e confirme as configurações. Certifique-se de marcar a caixa confirmando que o modelo criará recursos do AWS Identity and Access Management (IAM).
10. Selecione Create stack (Criar pilha) para implantar a pilha.

    Você pode ver o status da pilha no CloudFormation console da AWS na coluna Status. Você deve receber o status CREATE_COMPLETE em aproximadamente cinco minutos.

Quando as regiões são implantadas com sucesso, elas aparecem no console da web. Quando você cria um teste, todas as regiões disponíveis são listadas no Painel e na Criação do Cenário. Você pode adicionar uma região a um teste na etapa Traffic Shape da criação do cenário.

A solução cria um item do DynamoDB para cada região implantada na tabela de cenários, que contém as informações necessárias sobre os recursos de teste nessa região. Você pode classificar os resultados do teste no console web por região. Para ver os resultados agregados em todas as regiões em um teste multirregional, use as métricas da Amazon CloudWatch . Você pode encontrar o código-fonte do gráfico nos resultados do teste após a conclusão do teste.

> **Note**
>
> Você pode iniciar a pilha regional sem o console web. Obtenha um link para o modelo regional no bucket de cenários do Amazon S3 e forneça-o como fonte ao iniciar a pilha regional na região necessária. Como alternativa, você pode baixar o modelo e carregá-lo como fonte para a região desejada.

Atualizar a solução

A atualização da solução aplica os recursos, patches de segurança e correções de erros mais recentes à sua implantação. Para atualizar para a versão mais recente, consulte a seção apropriada com base no seu método de implantação original: AWS Launch Wizard ou AWS CloudFormation.

> **Important**
>
> Antes de atualizar, certifique-se de que nenhum teste de carga esteja em execução no momento. O processo de atualização pode interromper temporariamente a disponibilidade da solução.

Atualize usando o AWS Launch Wizard

O console exibe automaticamente a versão mais recente disponível da solução no menu suspenso Versão de implantação. Se você já implantou a solução, siga este procedimento para atualizar sua implantação para a versão mais recente.

1. Acesse Launch Wizard Deployments.
2. Selecione a implantação que você deseja atualizar.
3. Escolha Ações e, em seguida, Atualizar versão de implantação.
4. Selecione a versão mais recente entre as versões de implantação disponíveis.
5. Revise a configuração.
6. Faça as alterações necessárias em cada etapa.
7. Confirme a atualização.

Atualize usando a AWS CloudFormation

Se você já implantou a solução, siga este procedimento para atualizar a CloudFormation pilha para a versão mais recente.

1. Entre no CloudFormation console, selecione sua CloudFormation pilha existente e selecione Atualizar pilha.
2. Selecione Fazer uma atualização direta.

3. Selecione Substituir modelo existente.
4. Em Especificar modelo:
   a. Selecione URL do Amazon S3.
   b. Copie o link do modelo mais recente.
   c. Cole o link na caixa de URL do Amazon S3.
   d. Verifique se o URL do modelo correto aparece na caixa de texto URL do Amazon S3.
   e. Escolha Próximo.
   f. Escolha Avançar novamente.
5. Em Parâmetros, revise os parâmetros do modelo e modifique-os conforme necessário. Consulte Iniciar a pilha para obter detalhes sobre os parâmetros.
6. Escolha Avançar.
7. Na página Configurar opções de pilha, selecione Avançar.
8. Na página Revisar, verifique e confirme as configurações.
9. Marque a caixa de seleção confirmando que o modelo poderá criar recursos do IAM.
10. Escolha Exibir conjunto de alterações e verifique as alterações.
11. Selecione Criar pilha para implantar a pilha.

Você pode ver o status da pilha no CloudFormation console da AWS na coluna Status. Você deve receber um `UPDATE_COMPLETE` status em aproximadamente 15 minutos.

> **Note**
>
> Se você tiver problemas de autenticação do Amazon Cognito ao fazer login usando seu navegador após o upgrade da pilha, atualize seu navegador (Ctrl+Shift+R no Windows/Linux ou Cmd+Shift+R no Mac) para limpar os dados em cache e tente novamente.

Solução de problemas de atualizações de versões anteriores à v3.3.0

> **Note**
>
> Esta seção se aplica somente às atualizações de versões anteriores à v3.3.0. Se você estiver atualizando a partir da versão 3.3.0 ou posterior, siga o procedimento de atualização padrão por meio do AWS Launch Wizard ou da AWS. CloudFormation

1. Baixe o distributed-load-testing-on-aws.template.
2. Abra o modelo, navegue até `Conditions:` e procure`DLTCommonResourcesAppRegistryCondition`.
3. Você deverá ver algo semelhante a:

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

4. Altere o segundo `true` valor para`false`:

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

5. Use o modelo personalizado para atualizar sua pilha seguindo as etapas em Atualizar usando a AWS CloudFormation.
6. Essa atualização remove os recursos relacionados ao registro do aplicativo da pilha, permitindo que a atualização seja concluída com êxito.
7. Execute outra atualização da pilha usando o URL do modelo mais recente.

Atualizando pilhas regionais

Se você implantou a solução em várias regiões, deverá atualizar cada pilha regional separadamente. Siga o procedimento de atualização padrão para cada CloudFormation pilha regional nas regiões em que você implantou a infraestrutura de teste.

Gerenciador de aplicativos do AWS Systems Manager

Depois de atualizar a solução, o AWS Systems Manager Application Manager fornece uma visão em nível de aplicativo da solução e de seus recursos. Você pode usar o Application Manager para:

- Monitore recursos, custos de recursos implantados em pilhas, contas e registros da AWS a partir de um local central.
- Visualize os dados operacionais dos recursos da solução no contexto de um aplicativo, como status de implantação, CloudWatch alarmes, configurações de recursos e problemas operacionais.

Solução de problemas

A resolução de problemas conhecidos fornece instruções para mitigar erros conhecidos. Se essas instruções não resolverem seu problema, o Contact AWS Support fornece instruções para abrir um caso do AWS Support para essa solução.

Resolução de problemas conhecidos

Problema: você está usando uma VPC existente e seus testes falham com o status Falha, resultando na seguinte mensagem de erro:

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

- Resolução:

Certifique-se de que as sub-redes existam na VPC especificada e que tenham uma rota para a Internet com um gateway da Internet ou um gateway NAT. O AWS Fargate precisa de acesso para extrair a imagem do contêiner do repositório público para executar testes com sucesso.

Problema: os testes estão demorando muito para serem executados ou estão paralisados indefinidamente

- Resolução:

Cancele o teste e verifique o AWS Fargate para garantir que todas as tarefas tenham sido interrompidas. Se elas não tiverem parado, interrompa manualmente todas as tarefas do Fargate. Verifique os limites de tarefas sob demanda do Fargate em sua conta para garantir que você possa iniciar o número de tarefas desejadas. Você também pode verificar os CloudWatch registros da função de execução de tarefas do Lambda para obter mais informações sobre falhas ao iniciar tarefas do Fargate. Verifique os registros do CloudWatch ECS para obter detalhes sobre o que está acontecendo nos contêineres Fargate em execução.

Problema: os testes estão começando, mas não estão sendo concluídos ou o estado das tarefas do ECS é desconhecido

- Resolução:

Se você selecionou a opção de fornecer uma VPC existente na conta em que a solução foi implantada, certifique-se de que a VPC usada pelas tarefas do ECS tenha endereços IP livres suficientes para iniciar o número de tarefas fornecidas na entrada de teste. A definição de tarefa do ECS usa a imagem do ECR que precisa de um gateway da Internet ou de uma rota para a Internet para que o serviço do ECS possa provisionar as tarefas baixando a imagem ECR da solução em aws-solutions/ -. distributed-load-testing-on aws-load-tester Se você não puder fornecer uma rota para a Internet, pois todas as sub-redes na VPC são privadas, você pode hospedar a imagem ECR em sua conta usando o cache pull through do ECR. Atualize a definição da tarefa com o novo URI da imagem ECR e crie uma nova revisão. Depois que a definição da tarefa é atualizada, a configuração da solução na tabela do DynamoDB precisa ser atualizada para usar a nova revisão. O nome da tabela do DynamoDB pode ser encontrado na guia de saídas CloudFormation da pilha abaixo da chave. ScenariosTable Atualize o atributo TaskDefinition para o item com a chave testId e o valor region- [SOLUTION-DEPLOYED-REGION].

Problema: os testes precisam usar um endpoint que seja privado ou não esteja disponível por meio do gateway da Internet

- Resolução:

Ao testar endpoints de API privados que não são acessíveis por meio do gateway da Internet, considere as seguintes abordagens:

1. Configuração de rede: garanta que as tabelas de rotas de sub-rede usadas pelas tarefas do ECS sejam atualizadas com uma rota para o intervalo de endereços IP do endpoint privado que está sendo testado. Isso permite que o tráfego de teste alcance o endpoint privado em sua VPC.
2. Resolução de DNS: para domínios personalizados, defina as configurações de DNS em sua VPC para resolver o nome de domínio do endpoint privado. Consulte a documentação do VPC DNS para obter instruções detalhadas.
3. VPC Endpoints: se estiver testando serviços da AWS, considere usar VPC endpoints (PrivateLinkAWS) para estabelecer conectividade privada. Por exemplo, para testar um API Gateway privado, você pode criar um VPC endpoint para o API Gateway. Consulte a documentação do Private API Gateway.
4. Emparelhamento de VPC: se o endpoint privado estiver em uma VPC diferente, estabeleça o emparelhamento de VPC entre a VPC em que a solução está implantada e a VPC que contém o endpoint privado. Configure as tabelas de rotas apropriadas em ambas VPCs. Consulte a documentação do VPC Peering.

5. Transit Gateway: Para cenários de rede mais complexos envolvendo vários VPCs, considere usar o AWS Transit Gateway para rotear o tráfego entre a VPC da solução e a VPC que contém o endpoint privado. Consulte a documentação do Transit Gateway.
6. Grupos de segurança: garanta que os grupos de segurança associados às suas tarefas do ECS permitam tráfego de saída para o endpoint privado, e que os grupos de segurança do endpoint privado permitam o tráfego de entrada das tarefas do ECS.

Para testar balanceadores de carga de aplicativos internos ou instâncias do EC2, certifique-se de que os intervalos de CIDR da VPC não se sobreponham e que as rotas necessárias estejam configuradas nas tabelas de rotas.

Problema: os testes estão sendo concluídos, mas os resultados não estão disponíveis na interface

- Resolução:

Se o teste tiver sido concluído, mas os resultados não estiverem disponíveis na interface do usuário, os arquivos de resultados ainda deverão estar disponíveis no bucket do S3 a partir das tarefas do ECS que executaram os testes. Essa é uma limitação conhecida na solução. Na arquitetura atual, a solução usa uma função Lambda de análise de resultados para resumir os resultados de várias tarefas do ECS, que são então armazenadas como um item na tabela do DynamoDB. A tabela do DynamoDB tem um limite de tamanho máximo de item de 400 KB. Essa limitação é atingida dependendo da complexidade do script de teste, da simultaneidade e do número de tarefas que estão sendo usadas. O erro não significa que o teste está falhando; indica que o processo para resumir os resultados e armazená-los na tabela do DynamoDB para operações CRUD falhou. Os resultados ainda estão disponíveis no bucket do S3 para o cenário de teste.

Entrar em contato com o AWS Support

Se você tem o AWS Business Support+, o AWS Enterprise Support ou o Unified Operations, você pode usar o AWS Support Center para obter assistência especializada com essa solução. As seções a seguir dão instruções.

Criar caso

1. Faça login na Central de suporte.
2. Escolha Criar caso.

Como podemos ajudar?

1. Escolha técnico
2. Em Serviço, selecione Soluções.
3. Em Categoria, selecione Teste de carga distribuído na AWS.
4. Em Severidade, selecione a opção que melhor corresponda ao seu caso de uso.
5. Quando você insere o Serviço, a Categoria e a Gravidade, a interface preenche links para perguntas comuns de solução de problemas. Se você não conseguir resolver suas dúvidas com esses links, escolha Próxima etapa: Informações adicionais.

Mais informações

1. Em Assunto, insira um texto resumindo sua pergunta ou problema.
2. Para Descrição, descreva o problema em detalhes, incluindo o nome desse produto e a versão que você está usando, como este exemplo: Teste de carga distribuído no AWS vX.Y.Z.
3. Selecione Anexar arquivos.
4. Anexe as informações que o AWS Support precisa para processar a solicitação.

Ajude-nos a resolver seu caso com mais rapidez

1. Insira as informações solicitadas.
2. Escolha Próxima etapa: solucione ou entre em contato conosco.

Solucione ou entre em contato conosco

1. Analise as soluções Solucionar agora.
2. Se você não conseguir resolver seu problema com essas soluções, escolha Fale conosco, insira as informações solicitadas e escolha Enviar.

Desinstalar a solução

Você pode desinstalar a solução Distributed Load Testing on AWS do AWS Management Console ou usando a AWS Command Line Interface. Você deve excluir manualmente o console, o cenário e os buckets de registro do Amazon Simple Storage Service (Amazon S3) criados por essa solução. As implementações de soluções da AWS não as excluem automaticamente caso você tenha dados para reter.

> **Note**
>
> Se você implantou pilhas regionais, você deve excluir as pilhas nessas regiões antes de excluir a pilha principal.

Como usar o AWS Management Console

AWS CloudFormation

1. Faça login no CloudFormation console da AWS.
2. Na página Pilhas, selecione a pilha de instalação dessa solução.
3. Escolha Excluir.

AWS Launch Wizard

1. Faça login no console do AWS Launch Wizard.
2. Na página Launch Wizard Deployments, selecione a implantação dessa solução.
3. Escolha Ações e Excluir.
4. Confirme a exclusão.

Usar a AWS Command Line Interface

Determine se a AWS Command Line Interface (AWS CLI) está disponível em seu ambiente. Para obter instruções de instalação, consulte O que é a AWS Command Line Interface no Guia do Usuário da AWS CLI. Depois de confirmar que a AWS CLI está disponível, execute o seguinte comando:

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

Excluindo os buckets do Amazon S3

Essa solução está configurada para reter os buckets Amazon S3 criados pela solução (para implantação em uma região opcional) se você decidir excluir a pilha da AWS para evitar perda acidental de dados. CloudFormation Depois de desinstalar a solução, você pode excluir manualmente esse bucket do S3 se não precisar reter os dados. Siga estas etapas para excluir o bucket do Amazon S3.

1. Faça login no console do Amazon S3.
2. No painel de navegação à esquerda, escolha Buckets.
3. No campo Localizar compartimentos por nome, insira o nome da pilha dessa solução.
4. Selecione um dos buckets S3 da solução e escolha Vazio.
5. Digite excluir permanentemente no campo de verificação e escolha Esvaziar.
6. Selecione o bucket do S3 que você acabou de esvaziar e escolha Excluir.
7. Insira o nome do bucket do S3 no campo de verificação e escolha Excluir bucket.

Repita as etapas de 4 a 7 até excluir todos os buckets do S3.

Para excluir o bucket do S3 usando o AWS CLI, execute o seguinte comando:

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

Uso da solução

Esta seção fornece um guia abrangente para usar a solução Distributed Load Testing on AWS, desde a criação do primeiro cenário de teste até a análise de resultados detalhados. O fluxo de trabalho inclui criar um cenário de teste, executar um teste e explorar os resultados do teste.

Crie um cenário de teste

A criação de um cenário de teste envolve quatro etapas principais: definir as configurações gerais, definir o cenário, moldar os padrões de tráfego e revisar sua configuração.

Etapa 1: configurações gerais

Configure os parâmetros básicos para seu teste de carga, incluindo nome do teste, descrição e opções gerais de configuração.

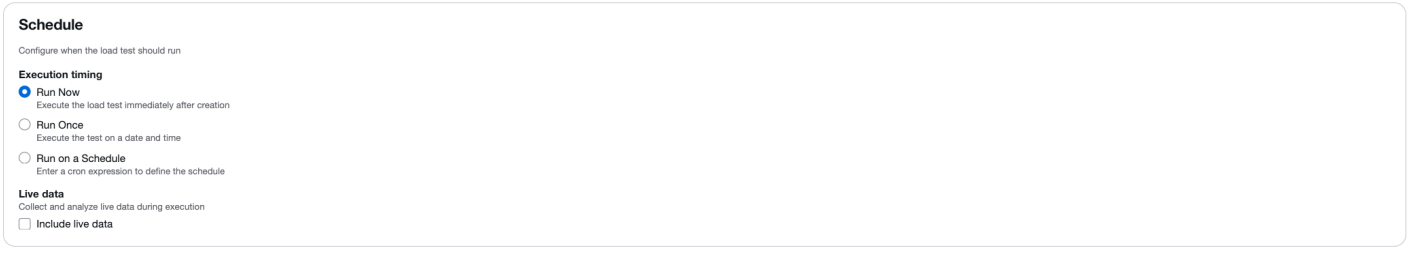
Identificação do teste

- Nome do teste (obrigatório) - Um nome descritivo para seu cenário de teste
- Descrição do teste (Obrigatório) - Detalhes adicionais sobre a finalidade e a configuração do teste
- Tags (opcional) - adicione até 5 tags para categorizar e organizar seus cenários de teste

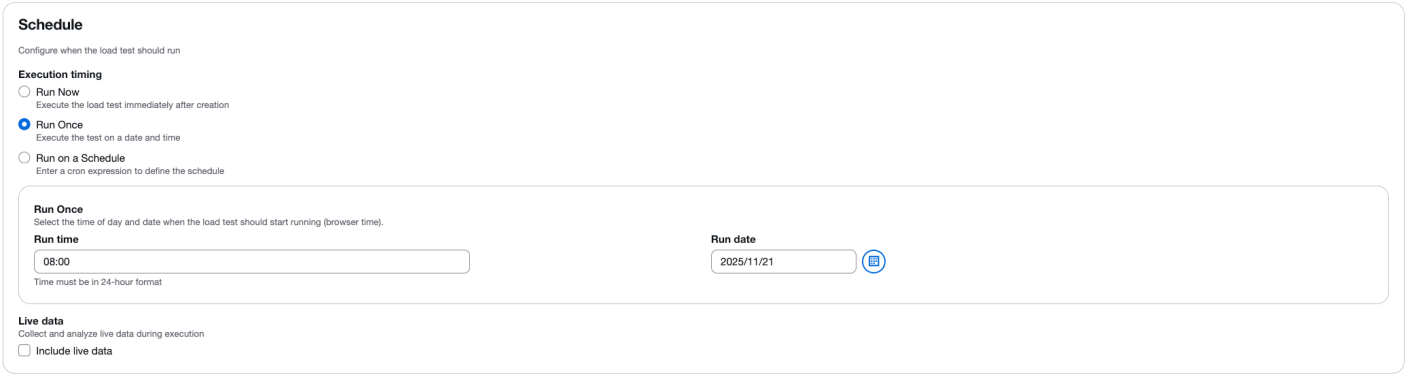
Opções de agendamento

Configure quando o teste deve ser executado:

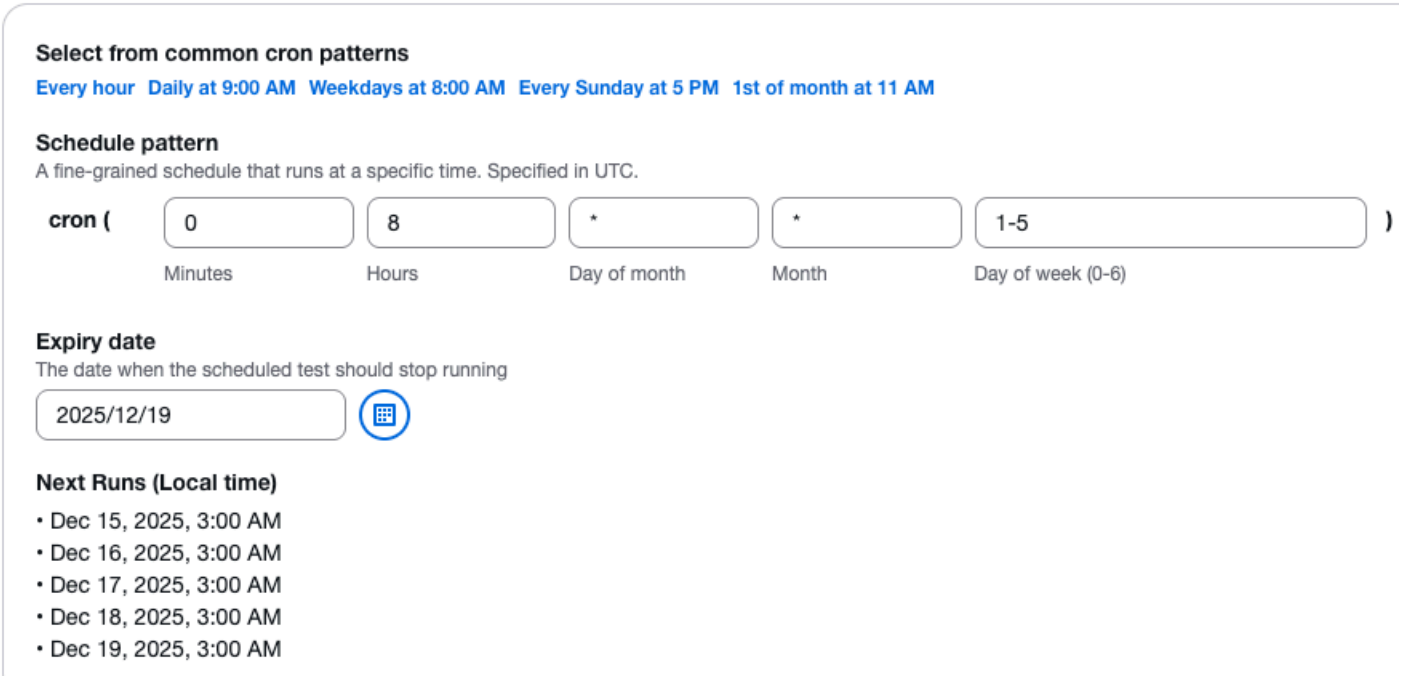
- Executar agora - Execute o teste imediatamente após a criação.



- Executar uma vez - Agende o teste para ser executado em uma data e hora específicas.



- Executar em um cronograma - Use o agendamento baseado em cron para executar testes automaticamente em intervalos regulares. Você pode selecionar padrões comuns (a cada hora, diariamente, semanalmente) ou definir uma expressão cron personalizada.



Fluxo de trabalho de agendamento

Quando você agenda um teste, ocorre o seguinte fluxo de trabalho:

- Os parâmetros do cronograma são enviados para a API da solução por meio do Amazon API Gateway.
- A API passa os parâmetros para uma função Lambda que cria uma regra de CloudWatch eventos programada para ser executada na data especificada.

- Para testes únicos (Run Once), a regra CloudWatch Events é executada na data especificada e a função `api-services` Lambda executa o teste.
- Para testes recorrentes (Executar em um cronograma), a regra de CloudWatch eventos é ativada na data especificada, e a função `api-services` Lambda cria uma nova regra que é executada imediatamente e de forma recorrente com base na frequência especificada.

Dados dinâmicos

Marque a caixa de seleção Incluir dados ativos para visualizar métricas em tempo real enquanto o teste está sendo executado. Quando ativado, você pode monitorar:

- Tempo médio de resposta.
- Contagens de usuários virtuais.
- Solicitações bem-sucedidas contam.
- Contagens de solicitações falhadas.

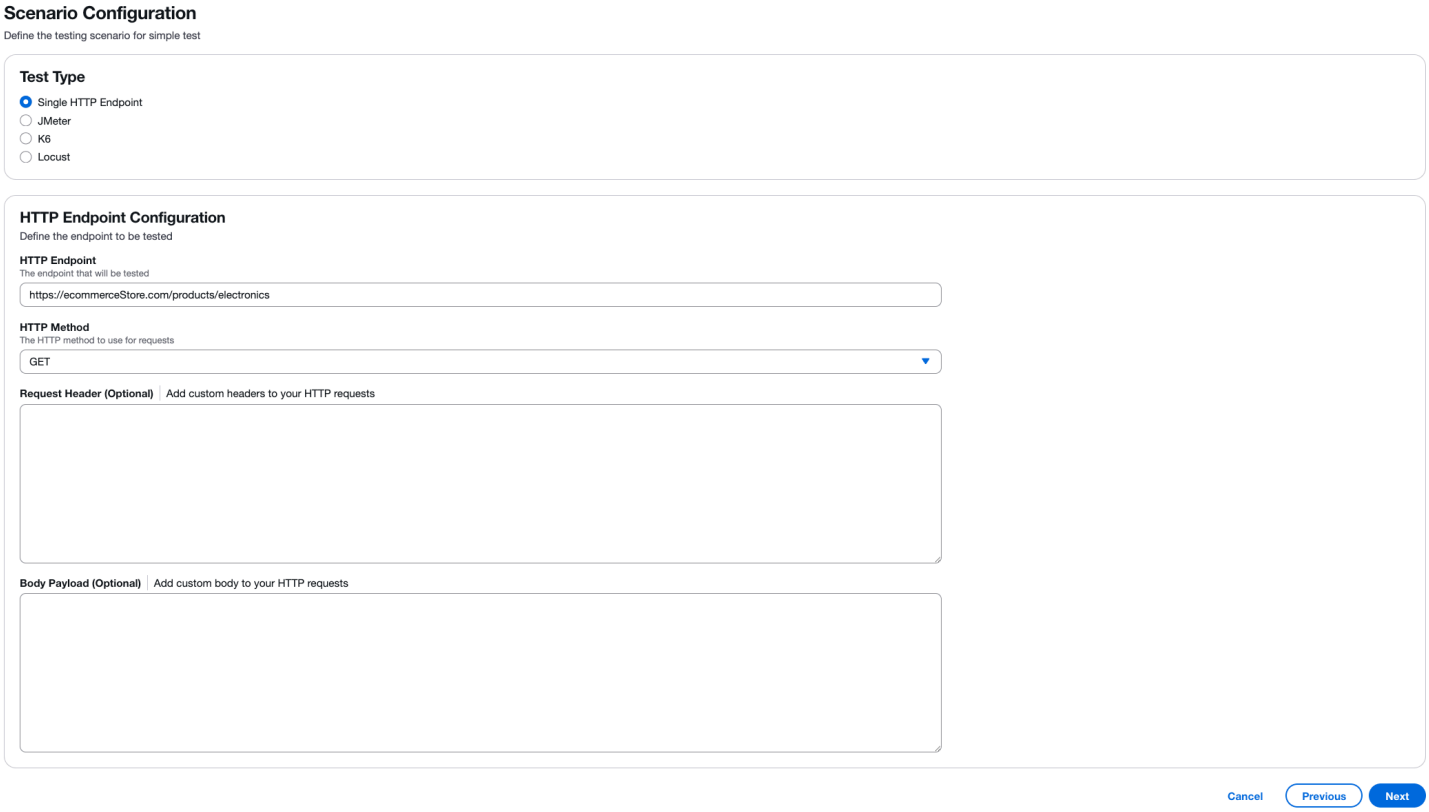
O recurso de dados ao vivo fornece gráficos em tempo real com dados agregados em intervalos de um segundo. Para obter mais informações, consulte Monitoramento com dados ativos.

Etapa 2: configuração do cenário

Defina o cenário de teste específico e selecione sua estrutura de teste preferida.

Seleção do tipo de teste

Escolha o tipo de teste de carga que você deseja realizar:



- Endpoint HTTP único - Teste um único endpoint de API ou página da web com configuração simples.
- JMeter- Faça upload JMeter de scripts de teste (arquivos.jmx ou arquivos.zip).
- K6 - Faça upload de scripts de teste K6 (arquivos.js ou arquivos.zip).
- Locust - Faça upload dos scripts de teste do Locust (arquivos.py ou arquivos.zip).

Imagem de configuração do endpoint HTTP: :images/test-types.png [Selecione o tipo de teste a ser executado] Quando “Endpoint HTTP único” for selecionado, defina estas configurações:

Endpoint HTTP (obrigatório)

Insira o URL completo do endpoint que você deseja testar. Por exemplo, .`https://api.example.com/users` Garanta que o endpoint seja acessível a partir da infraestrutura da AWS.

Método HTTP (obrigatório)

Selecione o método HTTP para suas solicitações. O padrão é `GET`. Outras opções incluem `POSTPUT`,`DELETE`, `PATCHHEAD`,, `OPTIONS` e.

Cabeçalho da solicitação (opcional)

Adicione cabeçalhos HTTP personalizados às suas solicitações. Os exemplos comuns incluem:

- `Content-Type: application/json`
- `Authorization: Bearer <token>`
- `User-Agent: LoadTest/1.0`

Escolha Adicionar cabeçalho para incluir vários cabeçalhos.

Carga útil do corpo (opcional)

Adicione o conteúdo do corpo da solicitação para solicitações POST ou PUT. Suporta formatos JSON, XML ou texto sem formatação. Por exemplo: `{"userId": 123, "action": "test"}`.

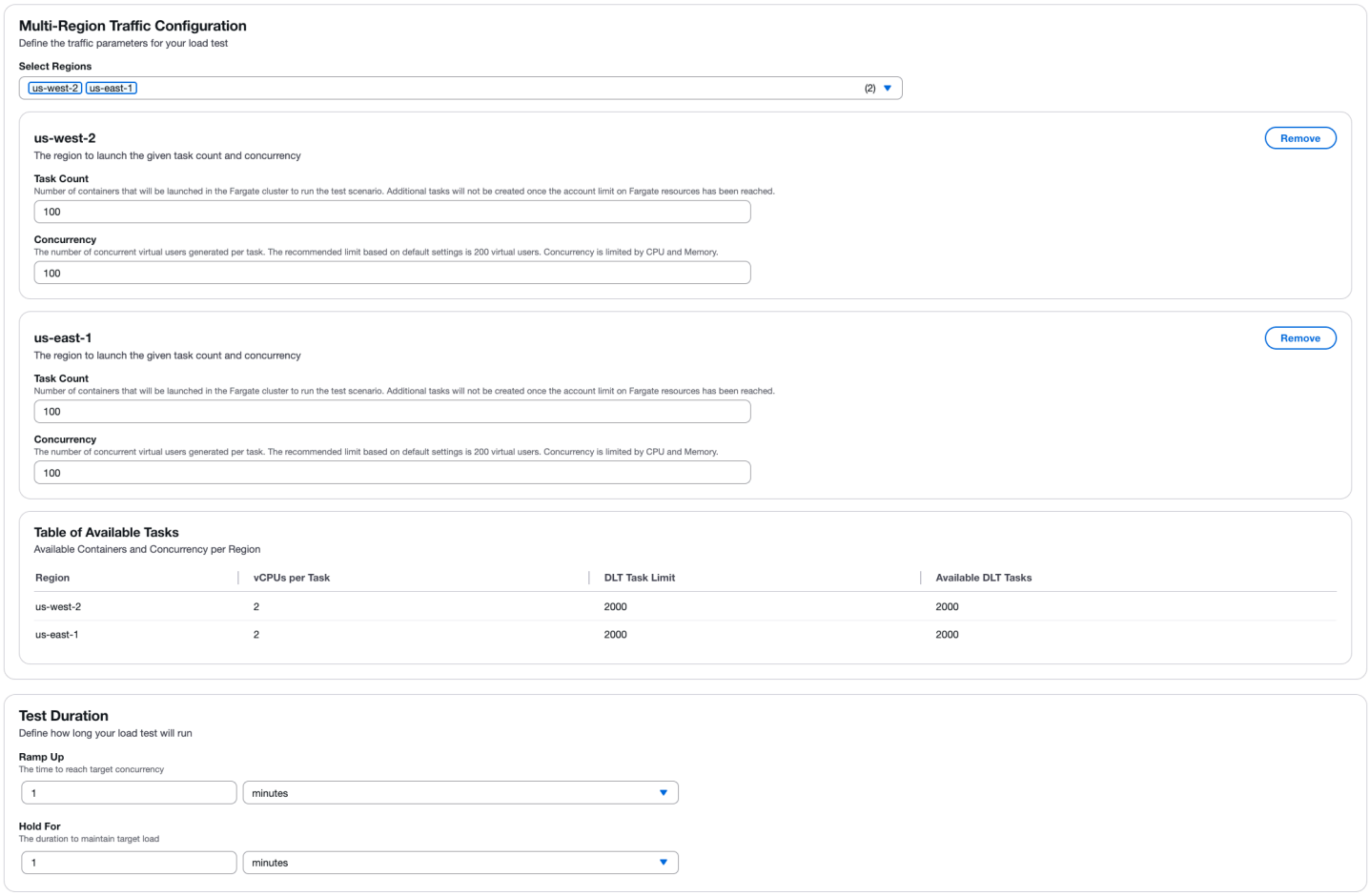
Scripts de estrutura de teste

Ao usar JMeter K6 ou Locust, carregue seu arquivo de script de teste ou um arquivo.zip contendo seu script de teste e arquivos de suporte. Pois JMeter, você pode incluir plug-ins personalizados em uma `/plugins` pasta dentro do seu arquivo.zip.

> **Important**
>
> Embora seu script de teste (JMeter, K6 ou Locust) possa definir simultaneidade (usuários virtuais), taxas de transação (TPS), tempos de aceleração e outros parâmetros de carregamento, a solução substituirá essas configurações pelos valores especificados na tela do Traffic Shape durante a criação do teste. A configuração do Traffic Shape controla a contagem de tarefas, a simultaneidade (usuários virtuais por tarefa), a duração do aumento e a duração da espera para a execução do teste.

Etapa 3: forma do tráfego

Configure como o tráfego será distribuído durante o teste, incluindo suporte multirregional.



Configuração de tráfego multirregional

Selecione uma ou mais regiões da AWS para distribuir seu teste de carga geograficamente. Para cada região selecionada, configure:

Contagem de tarefas

O número de contêineres (tarefas) que serão lançados no cluster Fargate para o cenário de teste. Tarefas adicionais não serão criadas quando a conta atingir o limite “O recurso Fargate foi atingido”.

Simultaneidade

O número de usuários virtuais simultâneos gerados por tarefa. O limite recomendado é baseado nas configurações padrão de 2 v CPUs por tarefa. A simultaneidade é limitada pelos recursos de CPU e memória.

Determine o número de usuários

O número de usuários que um contêiner pode suportar para um teste pode ser determinado aumentando gradualmente o número de usuários e monitorando o desempenho na Amazon CloudWatch. Depois de observar que o desempenho da CPU e da memória está se aproximando do limite, você atingiu o número máximo de usuários que um contêiner pode suportar para esse teste em sua configuração padrão (2 vCPU e 4 GB de memória).

Processo de calibração

Você pode começar a determinar os limites de usuários simultâneos para seu teste usando o exemplo a seguir:

1. Crie um teste com no máximo 200 usuários.
2. Enquanto o teste é executado, monitore a CPU e a memória usando o CloudWatch console:
   a. No painel de navegação esquerdo, em Container Insights, selecione Monitoramento de desempenho.
   b. Na página Monitoramento de desempenho, no menu suspenso esquerdo, selecione ECS Clusters.
   c. No menu suspenso à direita, selecione seu cluster do Amazon Elastic Container Service (Amazon ECS).
3. Durante o monitoramento, observe a CPU e a memória. Se a CPU não ultrapassar 75% ou a memória não ultrapassar 85% (ignore picos únicos), você poderá executar outro teste com um número maior de usuários.

Repita as etapas de 1 a 3 se o teste não exceder os limites de recursos. Opcionalmente, você pode aumentar os recursos do contêiner para permitir um número maior de usuários simultâneos. No entanto, isso resulta em um custo maior. Para obter detalhes, consulte o Guia do desenvolvedor.

> **Note**
>
> Para obter resultados precisos, execute somente um teste por vez ao determinar os limites de usuários simultâneos. Todos os testes usam o mesmo cluster, e o CloudWatch container insights agrega os dados de desempenho com base no cluster. Isso faz com que os dois testes sejam reportados ao CloudWatch Container Insights simultaneamente, o que resulta em métricas imprecisas de utilização de recursos para um único teste.

Para obter mais informações sobre como calibrar usuários por motor, consulte Calibrando um teste Taurus na documentação. BlazeMeter

> **Note**
>
> A solução exibe as informações de capacidade disponíveis para cada região, ajudando você a planejar sua configuração de teste dentro dos limites disponíveis.

Tabela de tarefas disponíveis

A Tabela de Tarefas Disponíveis exibe a disponibilidade de recursos para cada região selecionada:

- Região - O nome da região da AWS.
- v CPUs por tarefa - O número de virtuais CPUs alocados para cada tarefa (padrão: 2).
- Limite de tarefas DLT - O número máximo de tarefas que podem ser criadas com base nos limites do Fargate da sua conta (padrão: 2000).
- Tarefas DLT disponíveis - O número atual de tarefas disponíveis para uso na região (padrão: 2000).

Table of Available Tasks

Available Containers and Concurrency per Region

Region	vCPUs per Task	DLT Task Limit	Available DLT Tasks
us-west-2	2	2000	2000
us-east-1	2	2000	2000

Para aumentar o número de tarefas disponíveis ou v CPUs por tarefa, consulte o Guia do desenvolvedor.

Duração do teste

Defina por quanto tempo seu teste de carga será executado:

Ramp Up

A hora de atingir a simultaneidade desejada. A carga aumenta gradualmente de 0 para o nível de simultaneidade configurado durante esse período.

Espere por

A duração para manter a carga alvo. O teste continua em total concorrência durante esse período.

Etapa 4: revisar e criar

Revise todas as suas configurações antes de criar o cenário de teste. Verificar:

- Configurações gerais (nome, descrição, cronograma).
- Configuração do cenário (tipo de teste, endpoint ou script).
- Forma do tráfego (tarefas, usuários, duração, regiões).

Depois de revisar, escolha Criar para salvar seu cenário de teste.

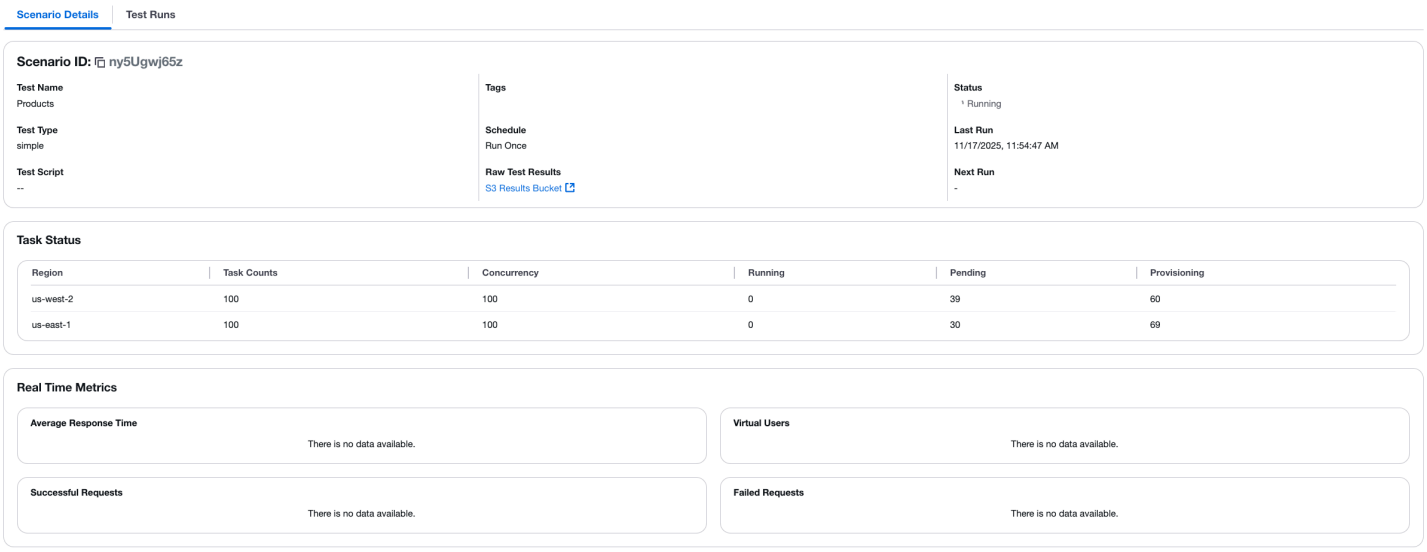
Gerenciando cenários de teste

Depois de criar um cenário de teste, você pode:

- Editar - Modifique a configuração do teste. Entre os casos de uso comuns estão:
  - Refinando a forma do tráfego para atingir a taxa de transação desejada.
- Copiar - Duplique um cenário de teste existente para criar variações. Entre os casos de uso comuns estão:
  - Atualizando endpoints ou adicionando headers/body parâmetros.
  - Adicionar ou modificar scripts de teste.
- Excluir - Remova cenários de teste que você não precisa mais.

Execute um cenário de teste

Depois de criar um cenário de teste, você pode executá-lo imediatamente ou programá-lo para ser executado em um horário específico no futuro. Quando você navega até um teste em execução, o console exibe a guia Detalhes do cenário com status e métricas da tarefa em tempo real.



Visualização de detalhes do cenário

A guia Detalhes do cenário exibe as principais informações sobre seu teste. O status da tarefa mostra informações em tempo real para cada região.

Tabela de status da tarefa

A tabela de status da tarefa mostra informações em tempo real para cada região:

- Região — A região da AWS em que as tarefas estão sendo executadas
- Contagem de tarefas - O número total de tarefas configuradas para a região
- Concorrência - O número de usuários virtuais por tarefa
- Em execução - Número de tarefas que estão executando o teste no momento
- Pendente - Número de tarefas esperando para serem iniciadas
- Provisionamento - Número de tarefas que estão sendo provisionadas

Fluxo de trabalho de execução de

Quando um teste é iniciado, ocorre o seguinte fluxo de trabalho:

1. Provisionamento de tarefas — A solução provisiona contêineres (tarefas) nas regiões especificadas da AWS. As tarefas aparecem na coluna “Provisionamento”.

2. Inicialização da tarefa - A solução continua a provisionar tarefas até que a contagem de tarefas alvo seja atingida em cada região. As tarefas passam de “Provisionamento” para “Pendente” e “Em execução”.
3. Geração de tráfego - Depois que a solução provisiona todas as tarefas em uma região, elas começam a enviar tráfego para o endpoint de destino.
4. Execução do teste - O teste é executado pela duração configurada (ramp-up + tempo de espera).
5. Análise de resultados - Quando o teste termina, uma tarefa de análise em segundo plano agrega e processa os resultados de todas as regiões.

Status de execução do teste

As execuções de teste podem ter os seguintes status:

- Agendado - O teste está programado para ser executado no futuro.
- Em execução - O teste está em andamento no momento.
- Cancelado - Um usuário cancelou uma execução de teste em andamento.
- Erro - A execução do teste encontrou um erro.
- Concluído - A execução do teste foi concluída com êxito e os resultados estão prontos.

Monitoramento com dados ao vivo

Se você ativou os dados ao vivo ao criar o cenário de teste, poderá visualizar métricas em tempo real enquanto o teste está sendo executado. A seção Métricas em tempo real exibe quatro gráficos que são atualizados continuamente à medida que o teste avança, com dados agregados em intervalos de um segundo.

Descrições gráficas

Tempo médio de resposta

Exibe o tempo médio de resposta em segundos para solicitações processadas por cada região. O eixo Y mostra o tempo de resposta em segundos e o eixo X mostra a hora do dia. Cada região é representada por uma cor diferente na legenda.

Usuários virtuais

Mostra o número de usuários virtuais simultâneos que geram carga ativamente em cada região. O gráfico mostra como os usuários virtuais crescem durante o teste e mantém o nível de simultaneidade desejado.

Solicitações bem-sucedidas

Exibe a contagem cumulativa de solicitações bem-sucedidas ao longo do tempo para cada região. O gráfico mostra a taxa na qual as solicitações bem-sucedidas estão sendo processadas.

Solicitações falhadas

Mostra a contagem cumulativa de solicitações malsucedidas ao longo do tempo para cada região. Uma contagem baixa ou zero indica uma execução saudável do teste.

Visualização multirregional

Ao executar testes em várias regiões, cada gráfico exibe dados de todas as regiões simultaneamente. A legenda na parte inferior de cada gráfico identifica qual cor representa cada região (por exemplo, us-west-2 e us-east-1).

Implementação técnica

O grupo de CloudWatch registros das tarefas do Fargate contém um filtro de assinatura que captura os resultados do teste. Quando o padrão é detectado, uma função Lambda estrutura os dados e os publica em um tópico do AWS IoT Core. O console web se inscreve nesse tópico e exibe as métricas em tempo real.

> **Note**
>
> Os dados ativos são efêmeros e só estão disponíveis durante a execução do teste. O console web persiste em no máximo 5.000 pontos de dados, após os quais os dados mais antigos são substituídos pelos mais novos. Se a página for atualizada, os gráficos ficarão em branco e começarão a partir do próximo ponto de dados disponível. Depois que o teste é concluído, a solução armazena os dados dos resultados no DynamoDB e no Amazon S3. Se ainda não houver dados disponíveis, os gráficos exibirão “Não há dados disponíveis”.

Cancelamento de um teste

Você pode cancelar um teste em execução no console web. Quando você cancela um teste, ocorre o seguinte fluxo de trabalho:

1. A solicitação de cancelamento é enviada para a `microservices` API
2. A `microservices` API chama a função `task-canceler` Lambda, que interrompe todas as tarefas lançadas atualmente.
3. Se a função `task-runner` Lambda continuar em execução após a chamada de cancelamento inicial, as tarefas poderão continuar sendo iniciadas brevemente
4. Quando a função `task-runner` Lambda é concluída, o AWS Step Functions prossegue para a `Cancel Test` etapa, que executa a função `task-canceler` Lambda novamente para interromper qualquer tarefa restante.

> **Note**
>
> Os testes cancelados demoram para concluir o processo de desligamento, pois a solução encerra todos os contêineres. O status do teste mudará para “Cancelado” quando todos os recursos forem limpos.

Explore os resultados do teste

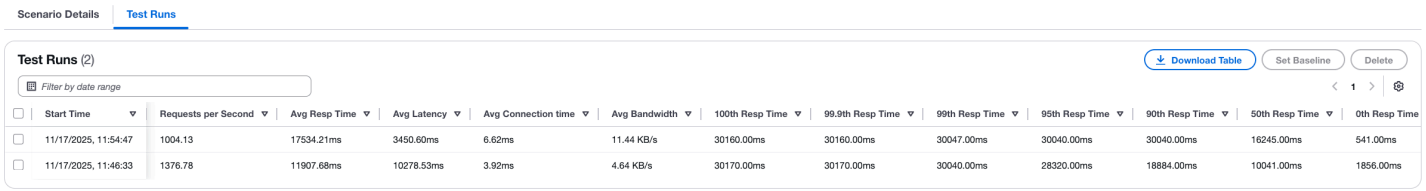
Depois que o trabalho de análise for concluído, os resultados do teste estarão disponíveis para análise. A solução fornece métricas e ferramentas abrangentes para ajudar você a entender o desempenho do seu aplicativo sob carga.

Métricas resumidas de execução de testes

Quando um teste é concluído, a solução gera um resumo que inclui as seguintes métricas:

- Tempo médio de resposta - O tempo médio de resposta, em segundos, para todas as solicitações geradas pelo teste.
- Latência média - A latência média, em segundos, de todas as solicitações geradas pelo teste.
- Tempo médio de conexão - O tempo médio, em segundos, necessário para se conectar ao host para todas as solicitações.
- Largura de banda média - A largura de banda média para todas as solicitações geradas pelo teste.
- Contagem total - O número total de solicitações.
- Contagem de sucesso - O número total de solicitações bem-sucedidas.
- Contagem de erros - O número total de erros.
- Solicitações por segundo - A média de solicitações por segundo para todas as solicitações geradas pelo teste.
- Percentis - Percentis de tempo de resposta, incluindo p50 (mediana), p90, p95 e p99, mostrando a distribuição dos tempos de resposta em todas as solicitações.

Tabela de execuções de teste



A tabela de execuções de teste exibe todas as execuções de teste históricas de um cenário. Você pode:

- Veja as métricas resumidas de cada teste executado.
- Defina um teste de linha de base para comparação de desempenho.
- Faça o download da tabela como um arquivo CSV.
- Alterne as colunas para personalizar sua visualização.
- Selecione uma execução de teste para ver os resultados detalhados.

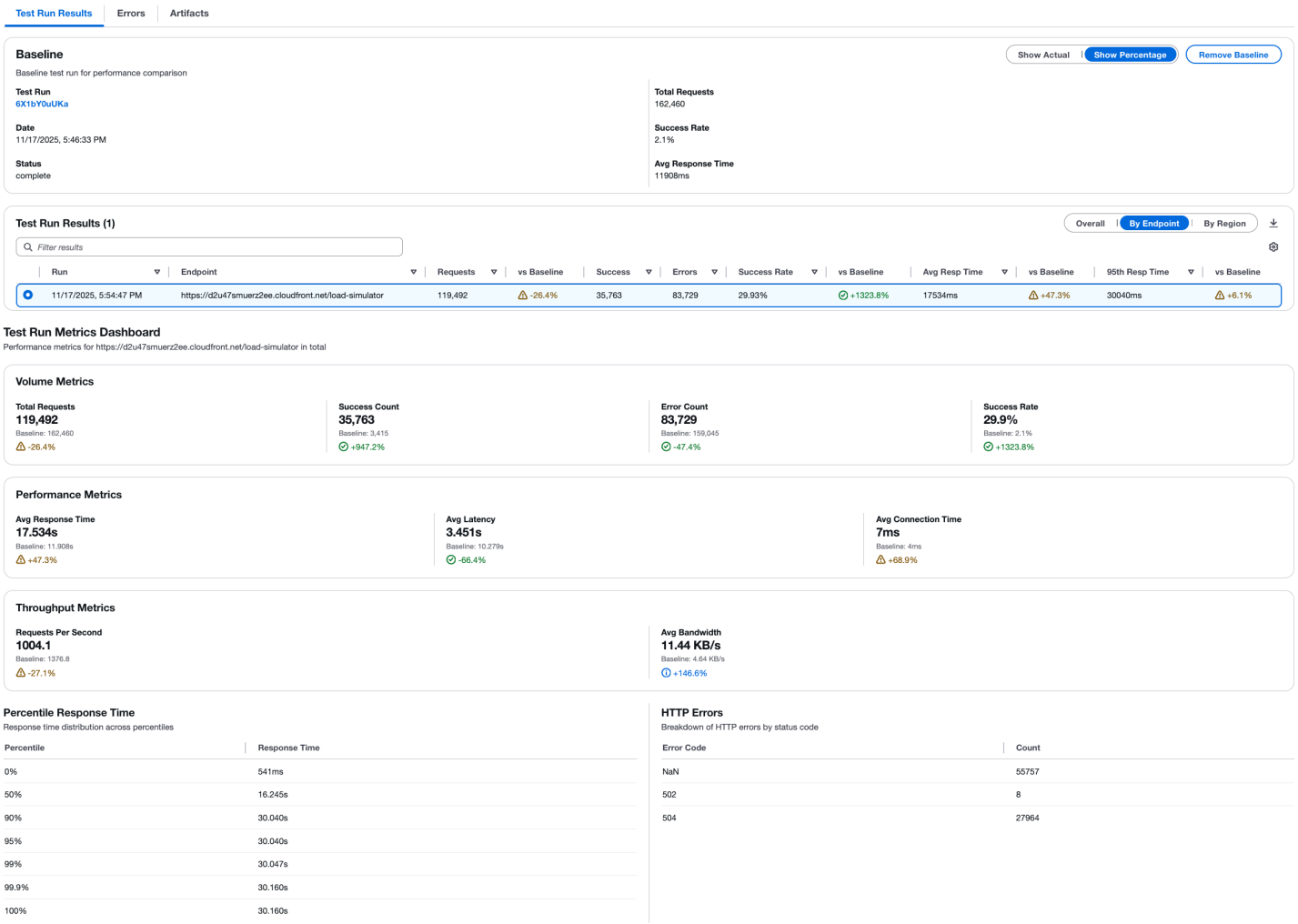
Comparação de linha de base

Você pode designar uma execução de teste como linha de base para comparar futuras execuções de teste com ela. Quando uma linha de base é definida:

- A tabela de execuções de teste mostra diferenças percentuais (+/-%) em comparação com a linha de base de cada métrica.
- O indicador de linha de base ajuda você a identificar rapidamente melhorias ou regressões de desempenho.
- Você pode alterar ou limpar a linha de base a qualquer momento.

Resultados detalhados do teste

Selecionar uma execução de teste abre a visualização detalhada dos resultados com três guias: Resultados da execução do teste, erros e artefatos.



Informações básicas

Se uma execução de teste de linha de base for definida, ela será exibida na parte superior da página. Você pode escolher Mostrar real, Mostrar porcentagem ou Remover linha de base para controlar como as comparações da linha de base são exibidas.

Tabela de resultados da execução do teste

A tabela de resultados fornece métricas detalhadas com os seguintes recursos:

Visualizações de dimensões

Alterne entre três visualizações usando os botões de dimensão:

- Geral - Resultados agregados em todos os endpoints e regiões
- Por endpoint - Resultados divididos por endpoints individuais

- Por região — Resultados detalhados por região da AWS

Botões de ação

- Mostrar real - Exibir valores métricos reais
- Mostrar porcentagem - Exibir diferenças percentuais em relação à linha de base
- Remover linha de base - Limpe a comparação da linha de base

Exportação e personalização de dados

- Baixe a tabela de resultados como um arquivo CSV
- Alterne as colunas para personalizar sua visualização
- Filtre e classifique os dados para se concentrar em métricas específicas
- Filtre e classifique os dados para se concentrar em métricas específicas.

Aba Erros

A guia de erros fornece uma análise detalhada dos erros:

- Visualize as contagens de erros por tipo.
- Veja os erros agregados por teste geral ou por endpoint.
- Identifique padrões em solicitações com falha.
- Solucione problemas com endpoints ou regiões específicas.

Aba Artefatos

A guia de artefatos permite que você acesse todos os arquivos gerados durante a execução do teste:

- Visualize artefatos individuais (registros, arquivos de resultados).
- Baixe artefatos específicos para análise offline.
- Baixe todos os artefatos de execução de teste como um único arquivo.

Estrutura de resultados do S3

Na versão 4.0, a estrutura de resultados do S3 foi alterada para melhorar a organização:

- Nova estrutura -`scenario-id/test-run-id/results-files`.
- Estrutura legada - Os testes executados antes da versão 4.0 mostram todos os arquivos de resultados no nível da ID do cenário.

> **Note**
>
> Os resultados do teste são exibidos no console. Você também pode acessar os resultados brutos do teste diretamente no bucket do Amazon S3, abaixo da `Results` pasta. Para obter mais informações sobre os resultados do teste Taurus, consulte Gerando relatórios de teste no Manual do usuário do Taurus.

Integração de servidor MCP

Se você implantou o componente opcional do servidor MCP durante a implantação da solução, poderá integrar a solução de teste de carga distribuída às ferramentas de desenvolvimento de IA que oferecem suporte ao Model Context Protocol. O servidor MCP fornece acesso programático para recuperar, gerenciar e analisar testes de carga por meio de assistentes de IA.

Os clientes podem se conectar ao servidor DLT MCP usando o cliente de sua escolha (Amazon Q, Claude etc.), cada um com instruções de configuração ligeiramente diferentes. Esta seção fornece instruções de configuração para MCP Inspector, Amazon Q CLI, Cline e Amazon Q Suite.

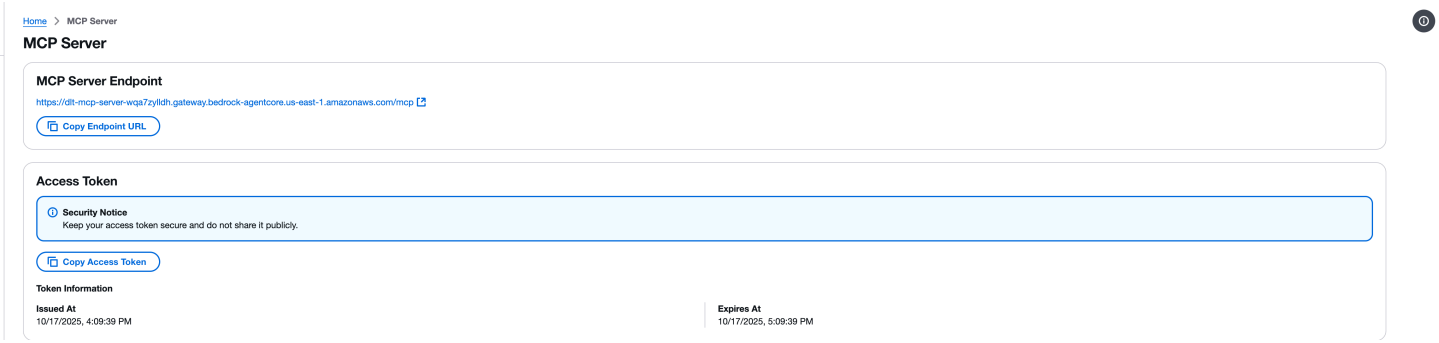
Etapa 1: Obtenha o endpoint MCP e o token de acesso

Antes de configurar qualquer cliente MCP, você precisa recuperar o endpoint do servidor MCP e o token de acesso do console web DLT.

1. Navegue até a página MCP Server no console web do Distributed Load Testing.
2. Localize a seção MCP Server Endpoint.
3. Copie o URL do endpoint usando o botão Copiar URL do endpoint. O URL do endpoint segue o formato: `https://{gateway-id}.gateway.bedrock-agentcore.{region}.amazonaws.com/mcp`
4. Localize a seção Token de acesso.
5. Copie o token de acesso usando o botão Copiar token de acesso.

Important

Mantenha seu token de acesso seguro e não o compartilhe publicamente. O token fornece acesso somente de leitura à sua solução de teste de carga distribuída por meio da interface MCP.



Passo 2: Teste com o MCP Inspector

O Model Context Protocol oferece o MCP Inspector, uma ferramenta para se conectar diretamente aos servidores MCP e invocar ferramentas. Isso fornece uma interface de usuário conveniente e exemplos de solicitações de rede para testar sua conexão com o servidor MCP antes de configurar clientes de IA.

Note

O MCP Inspector requer a versão 0.17 ou posterior. Todas as solicitações também podem ser feitas diretamente com o JSON RPC, mas o MCP Inspector fornece uma interface mais amigável.

Instale e inicie o MCP Inspector

1. Instale o npm, se necessário.
2. Execute o seguinte comando para iniciar o MCP Inspector:

```
npx @modelcontextprotocol/inspector
```

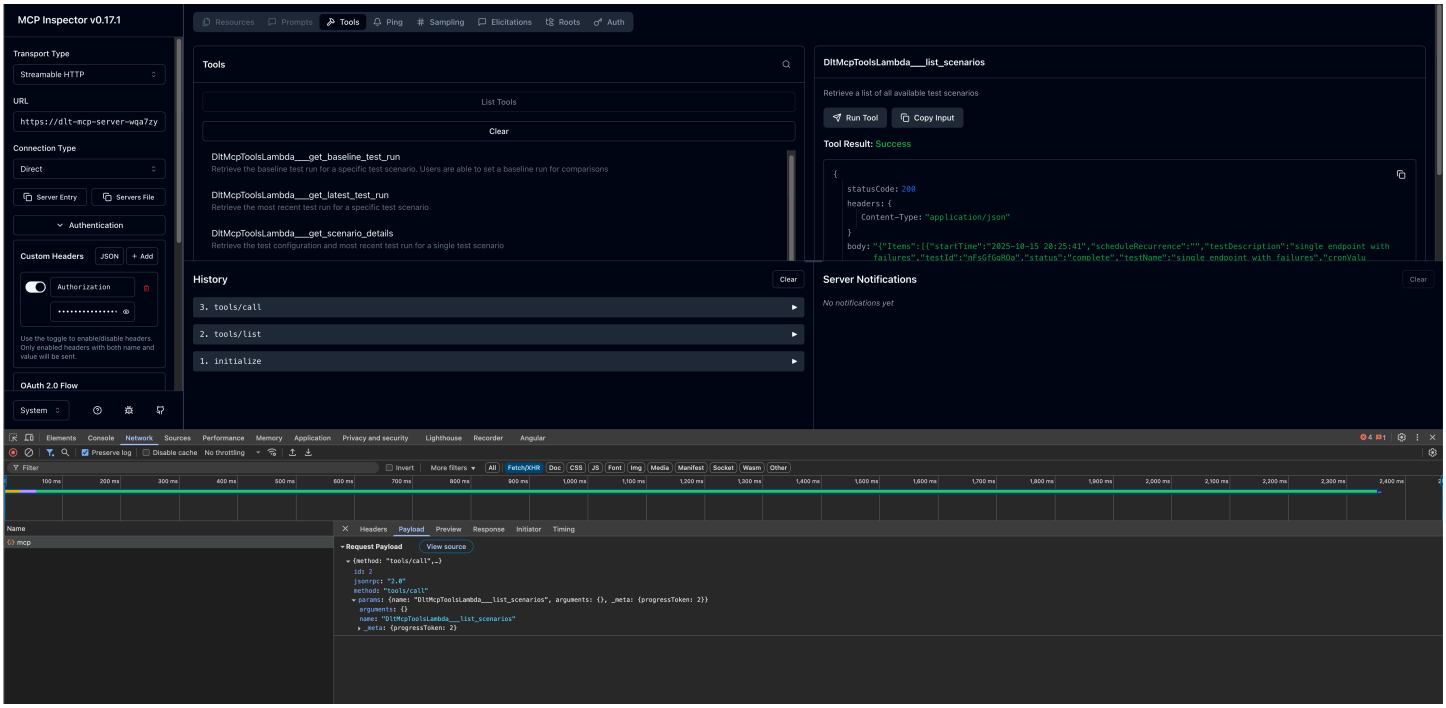
Configurar a conexão

1. Na interface do MCP Inspector, insira seu URL do MCP Server Endpoint.
2. Adicione um cabeçalho de autorização com seu token de acesso.
3. Clique em Connect para estabelecer a conexão.

Invocar ferramentas

Depois de conectado, você pode testar as ferramentas MCP disponíveis:

1. Navegue pela lista de ferramentas disponíveis no painel esquerdo.
2. Selecione uma ferramenta (por exemplo,`list_scenarios`).
3. Forneça todos os parâmetros necessários.
4. Clique em Invocar para executar a ferramenta e visualizar a resposta.



Etapa 3: configurar clientes de desenvolvimento de IA

Depois de verificar a conexão do servidor MCP com o MCP Inspector, você pode configurar seu cliente de desenvolvimento de IA preferido.

CLI do Amazon Q

O Amazon Q CLI fornece acesso por linha de comando ao desenvolvimento assistido por IA com a integração do servidor MCP.

Etapas de configuração

1. Edite o arquivo de configuração do `mcp.json`. Para obter mais informações sobre a localização do arquivo de configuração, consulte Configuração de servidores MCP remotos no Amazon Q Developer User Guide.
2. Adicione sua configuração de servidor DLT MCP:

```
{
  "mcpServers": {
    "dlt-mcp": {
      "type": "http",
      "url": "https://[api-id].execute-api.[region].amazonaws.com/[stage]/gateway/
backend-agent/sse/mcp",
```

```
      "headers": {
        "Authorization": "your_access_token_here"
      }
    }
  }
}
```

Verifique a configuração

1. Em um terminal, digite q para iniciar o Amazon Q CLI.
2. Digite `/mcp` para ver todos os servidores MCP disponíveis.
3. Digite `/tools` para ver as ferramentas disponíveis fornecidas por `dlt-mcp` e outros servidores MCP configurados.
4. Verifique se `dlt-mcp` foi inicializado com sucesso.

Cline

O Cline é um assistente de codificação de IA que oferece suporte à integração do servidor MCP.

Etapas de configuração

1. Em Cline, navegue até Gerenciar servidores MCP > Configurar > Configurar servidores MCP.
2. Atualize o arquivo `cline_mcp_settings.json`:

```
{
  "mcpServers": {
    "dlt-mcp": {
      "type": "streamableHttp",
      "url": "https://[api-id].execute-api.[region].amazonaws.com/[stage]/gateway/
backend-agent/sse/mcp",
      "headers": {
        "Authorization": "your_access_token_here"
      }
    }
  }
}
```

3. Salve o arquivo de configuração.
4. Reinicie o Cline para aplicar as alterações.

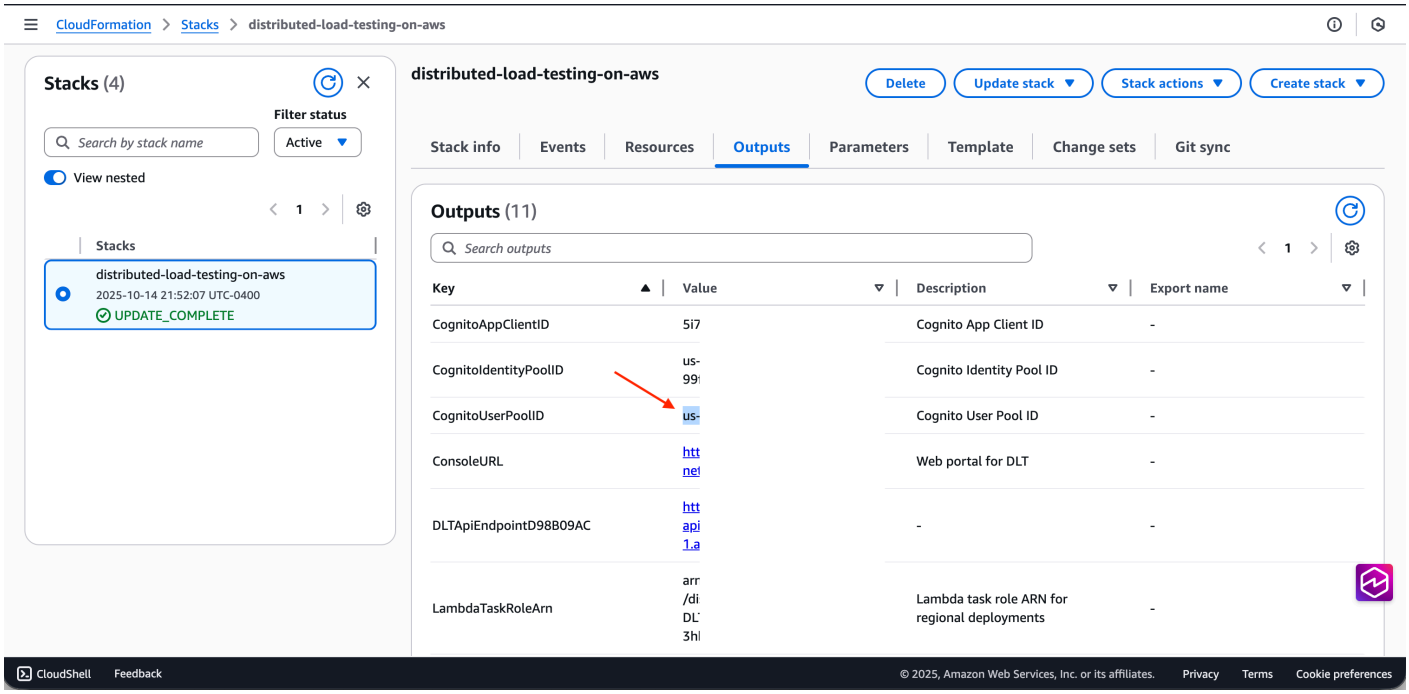
Suíte Amazon Q

O Amazon Q Suite fornece uma plataforma abrangente de assistente de IA com suporte para ações do servidor MCP.

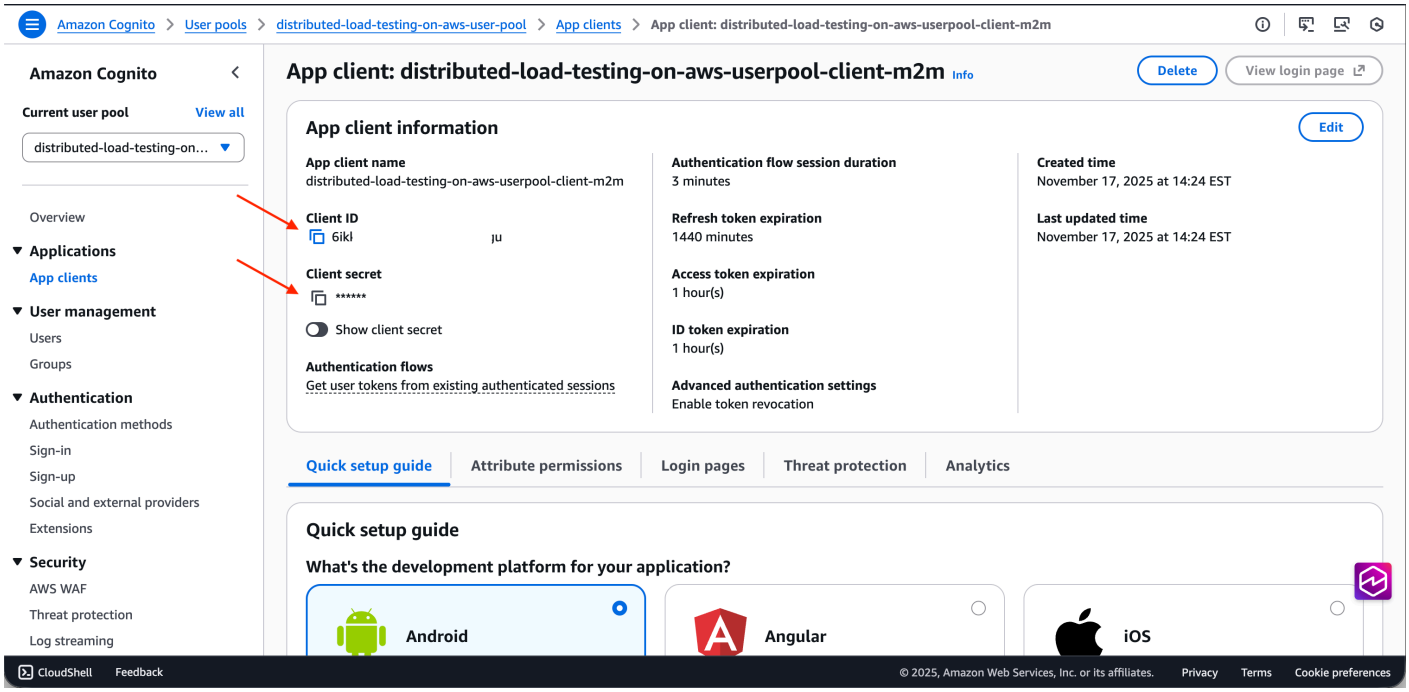
Pré-requisitos

Antes de configurar o servidor MCP no Amazon Q Suite, você precisa recuperar as OAuth credenciais do grupo de usuários do Cognito da sua implantação de DLT:

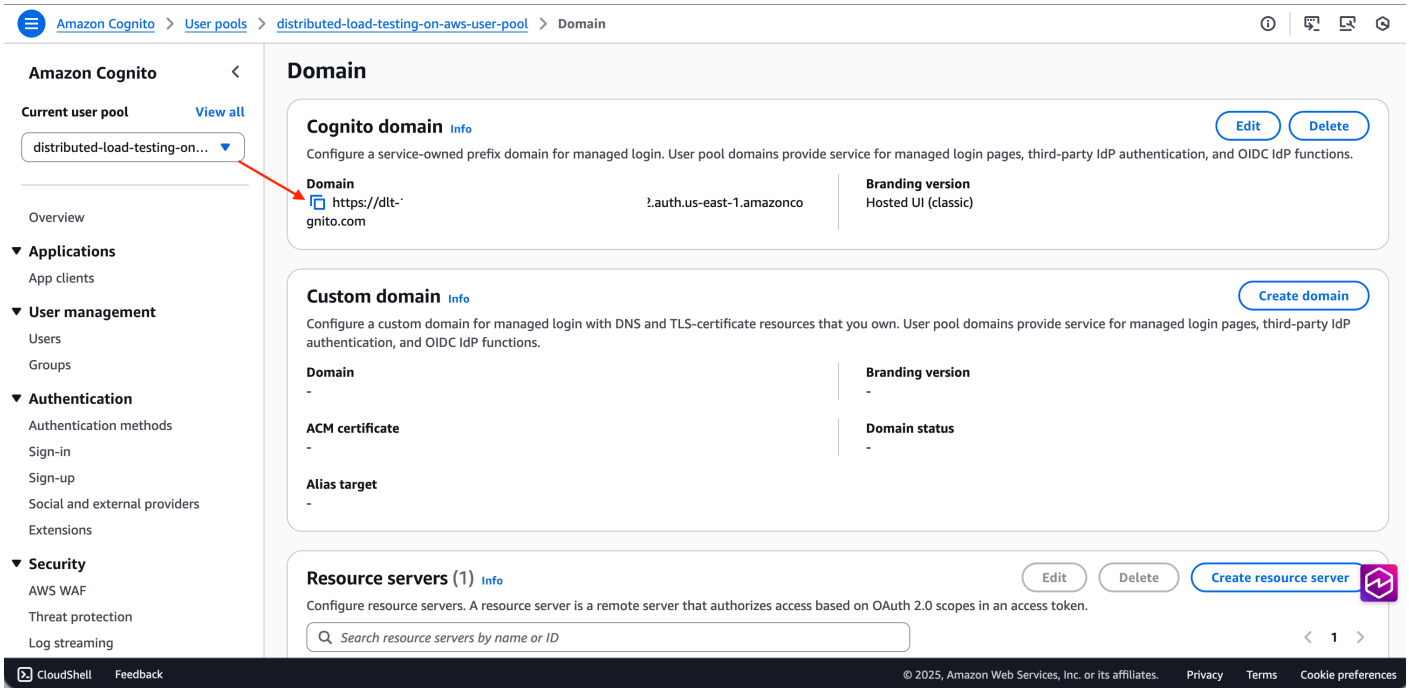
1. Navegue até o CloudFormation console da AWS.
2. Selecione a pilha de testes de carga distribuída.
3. Na guia Saídas, localize e copie o ID do Grupo de Usuários do Cognito associado à sua implantação de DLT.



4. Acesse o console do Amazon Cognito.
5. Selecione o grupo de usuários usando o ID do grupo de usuários nas CloudFormation saídas.
6. No painel de navegação à esquerda, selecione Integração de aplicativos > Clientes de aplicativos.



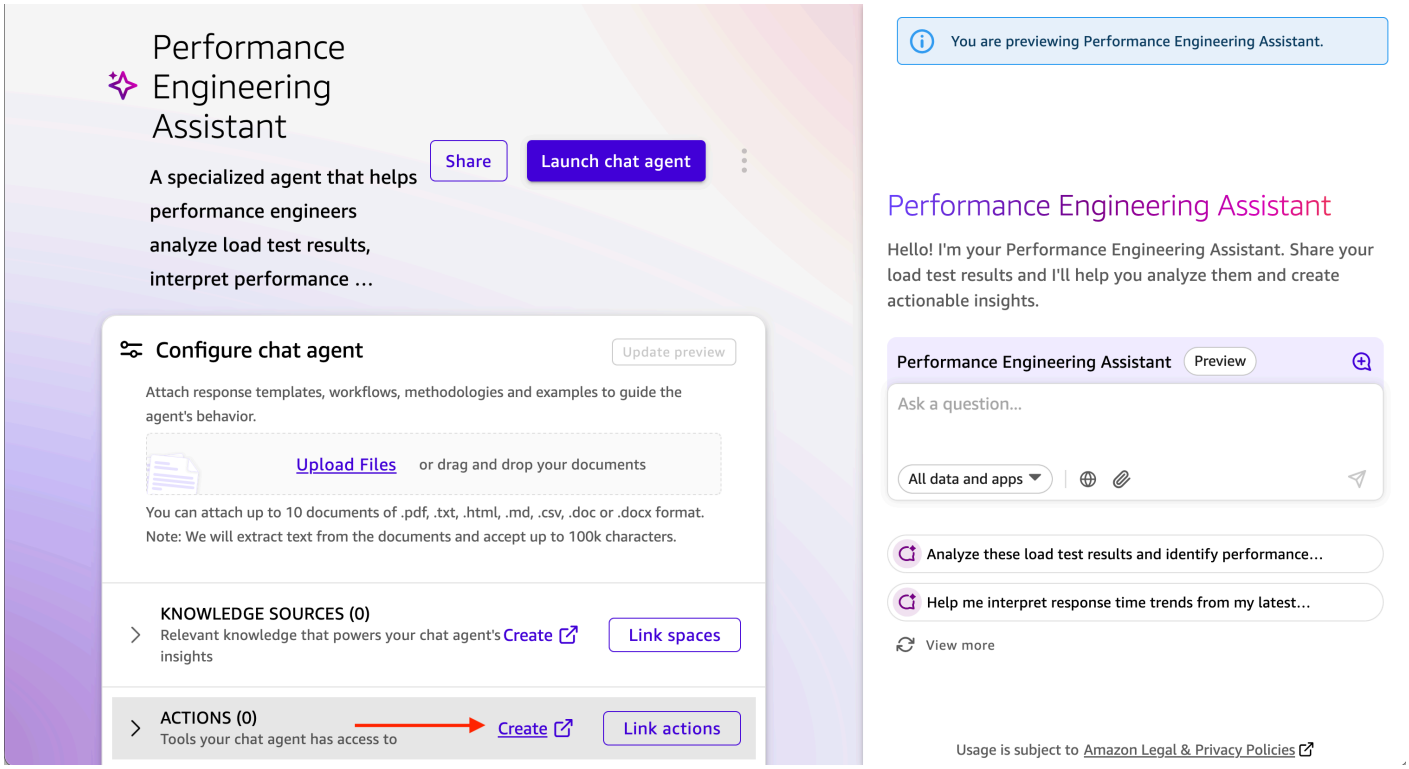
7. Localize o cliente do aplicativo com o nome terminado em `m2m` (machine-to-machine).
8. Copie o ID do cliente e o segredo do cliente.
9. Obtenha o domínio do grupo de usuários na guia Domínio.

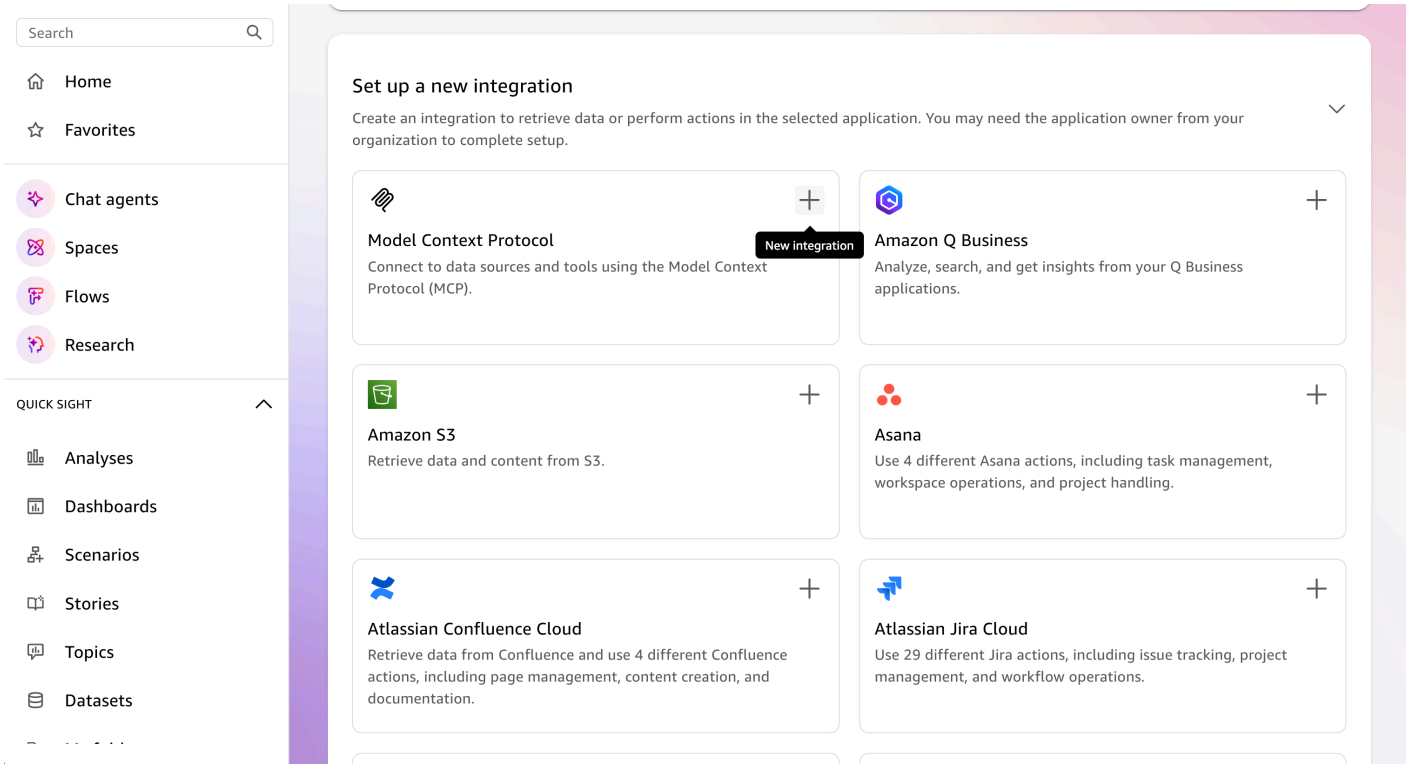


10. Crie o URL do endpoint do token anexando-o `/oauth2/token` ao final do domínio.

Etapas de configuração

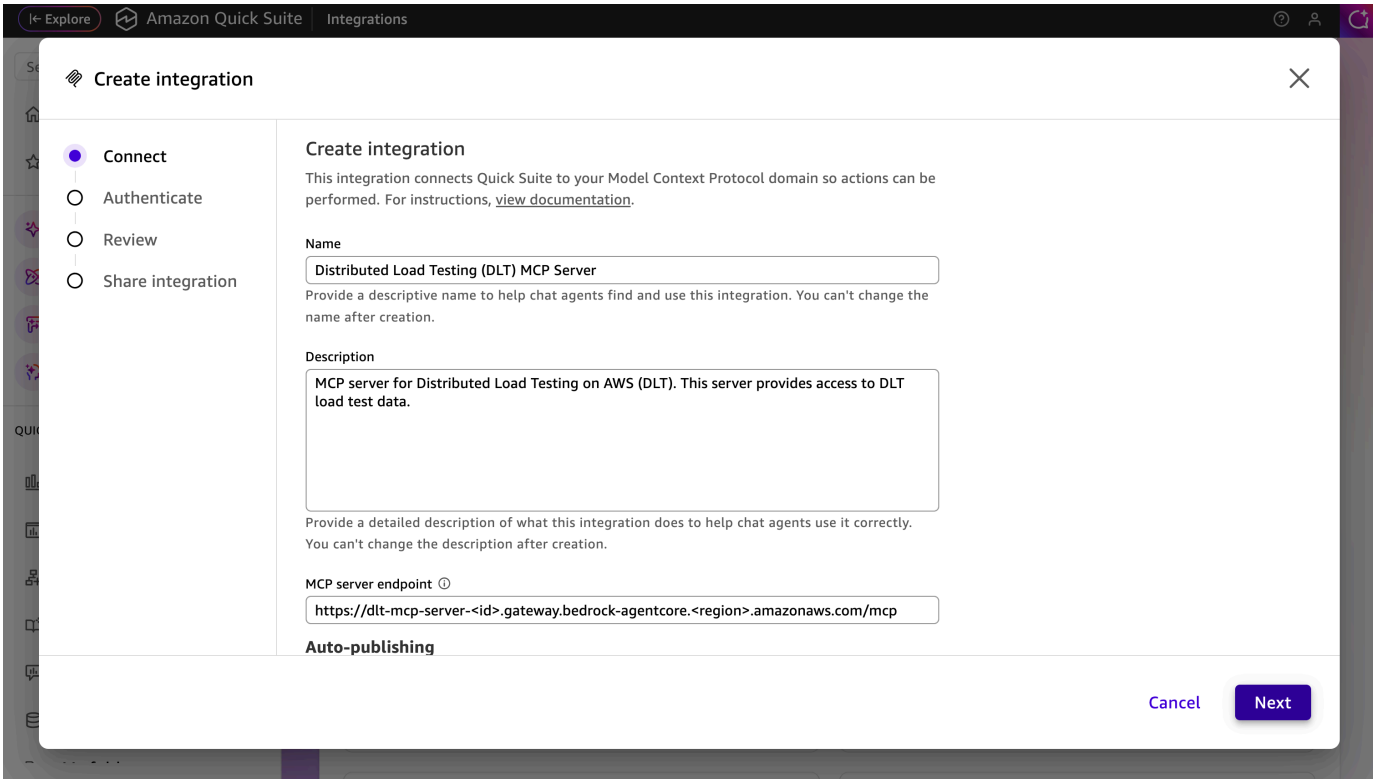
1. No Amazon Q Suite, crie um novo agente ou selecione um agente existente.
2. Adicione um prompt de agente que descreva como interagir com o servidor DLT MCP.
3. Adicione uma nova ação e selecione a ação do servidor MCP.



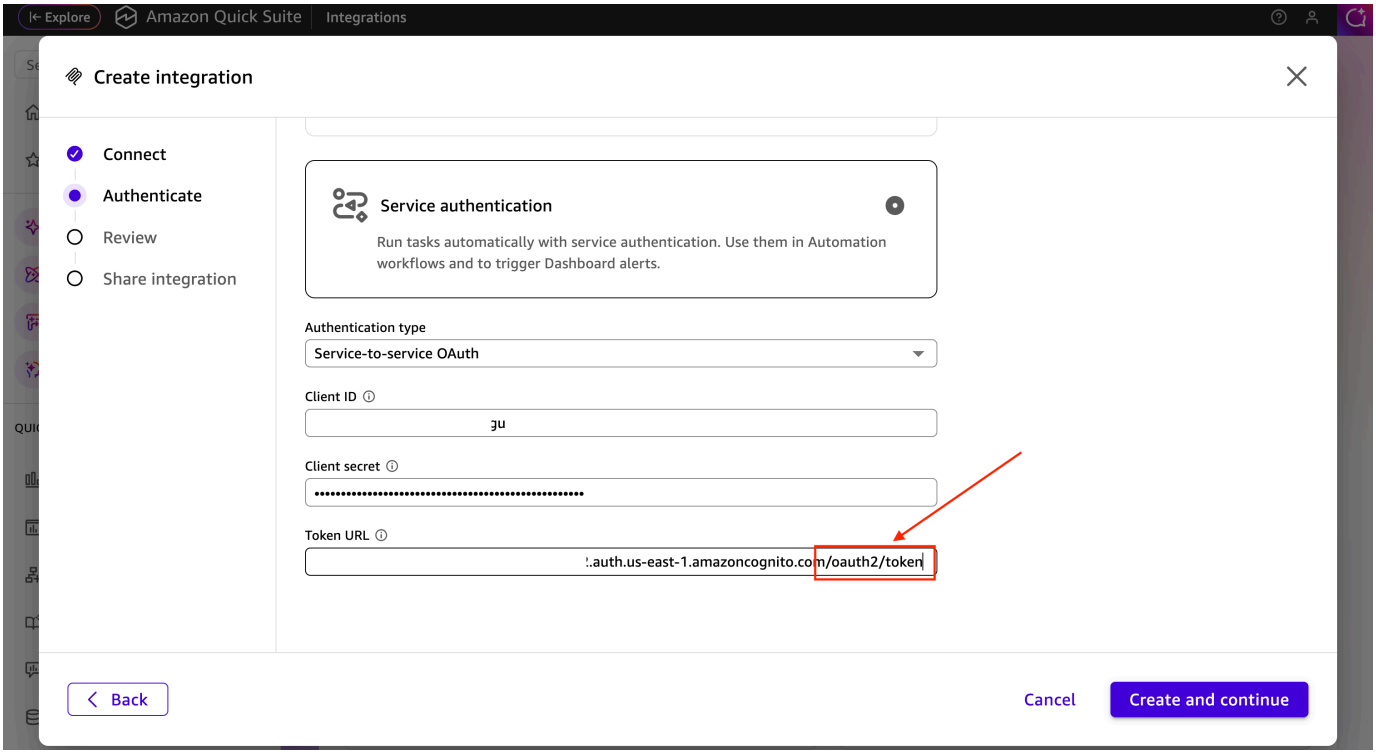


4. Configure os detalhes do servidor MCP:

- URL do servidor MCP: Seu endpoint DLT MCP



- Tipo de autenticação: Autenticação baseada em serviços
- Token Endpoint: Seu URL do endpoint do token Cognito
- ID do cliente: O ID do cliente do aplicativo m2m
- Segredo do cliente: O segredo do cliente do aplicativo m2m



5. Salve a configuração de ação do servidor MCP.
6. Adicione a nova ação do servidor MCP ao seu agente.

Inicie e teste o agente

1. Inicie o agente no Amazon Q Suite.
2. Inicie uma conversa com o agente usando instruções em linguagem natural.
3. O agente usará as ferramentas MCP para recuperar e analisar seus dados de teste de carga.

Exemplos de prompt

Os exemplos a seguir demonstram como interagir com seu assistente de IA para analisar dados de teste de carga por meio da interface MCP. Personalize o teste IDs, os intervalos de datas e os critérios para atender às suas necessidades específicas de teste.

Para obter informações detalhadas sobre as ferramentas MCP disponíveis e seus parâmetros, consulte a especificação das ferramentas MCP no Guia do desenvolvedor.

Consulta simples de resultados de testes

A interação da linguagem natural com o servidor MCP pode ser tão simples quanto `Show me the load tests that have completed in the last 24 hours with their associated completion status` ou pode ser mais descritiva, como

```
Use list_scenarios to find my load tests. Then use get_latest_test_run to show me the
 basic execution data and performance metrics for the most recent test. If the results
 look concerning, also get the detailed performance metrics using get_test_run.
```

Análise de desempenho interativa com divulgação progressiva

```
I need to analyze my load test performance, but I'm not sure which specific tests to
 focus on. Please help me by:

1. First, use list_scenarios to show me available test scenarios
2. Ask me which tests I want to analyze based on the list you show me
3. For my selected tests, use list_test_runs to get the test run history
4. Then use get_test_run with the test_run_id to get detailed response times,
 throughput, and error rates
5. If I want to compare tests, use get_baseline_test_run to compare against the
 baseline
6. If there are any issues, use get_test_run_artifacts to help me understand what went
 wrong

Please guide me through this step by step, asking for clarification whenever you need
 more specific information.
```

Validação da prontidão de produção

```
Help me validate if my API is ready for production deployment:

1. Use list_scenarios to find recent test scenarios
2. For the most recent test scenario, use get_latest_test_run to get basic execution
 data
3. Use get_test_run with that test_run_id to get detailed response times, error rates,
 and throughput
4. Use get_scenario_details with the test_id to show me what load patterns and
 endpoints were tested
5. If I have a baseline, use get_baseline_test_run to compare current results with the
 baseline
```

```
6. Provide a clear go/no-go recommendation based on the performance data
7. If there are any concerns, use get_test_run_artifacts to help identify potential
 issues

My SLA requirements are: response time under [X]ms, error rate under [Y]%.
```

Análise de tendências de desempenho

```
Analyze the performance trend for my load tests over the past [TIME_PERIOD]:

1. Use list_scenarios to get all test scenarios
2. For each scenario, use list_test_runs with start_date and end_date to get tests from
 that period
3. Use get_test_run for the key test runs to get detailed metrics
4. Use get_baseline_test_run to compare against the baseline
5. Identify any significant changes in response times, error rates, or throughput
6. If you detect performance degradation, use get_test_run_artifacts on the problematic
 tests to help identify causes
7. Present the trend analysis in a clear format showing whether performance is
 improving, stable, or degrading

Focus on completed tests and limit results to [N] tests if there are too many.
```

Solução de problemas em testes com falha

```
Help me troubleshoot my failed load tests:

1. Use list_scenarios to find test scenarios
2. For each scenario, use list_test_runs to find recent test runs
3. Use get_test_run with the test_run_id to get the basic execution data and failure
 information
4. Use get_test_run_artifacts to get detailed error messages and logs
5. Use get_scenario_details to understand what was being tested when it failed
6. If I have a similar test that passed, use get_baseline_test_run to identify
 differences
7. Summarize the causes of failure and suggest next steps for resolution

Show me the most recent [N] failed tests from the past [TIME_PERIOD].
```

Guia do desenvolvedor

Esta seção fornece o código-fonte da solução e personalizações adicionais.

Código-fonte

Visite nosso GitHub repositório para baixar os modelos e scripts dessa solução e compartilhar suas personalizações com outras pessoas.

Manutenção

Essa solução usa imagens do Docker com versões fixas que correspondem a cada versão da solução. Por padrão, as atualizações automáticas estão desativadas, dando a você controle total sobre quando e quais atualizações de versão são aplicadas à sua implantação. A equipe de soluções da AWS usa o Amazon ECR Enhanced Scanning para detectar vulnerabilidades e exposições comuns (CVEs) na imagem base e nos pacotes instalados. Quando possível, a equipe publica imagens corrigidas com a mesma tag de versão para resolver CVEs sem quebrar a compatibilidade com a versão da solução lançada.

Quando as imagens são corrigidas na mesma versão secundária, a tag stable é atualizada automaticamente e uma tag de imagem adicional é criada no formato`<solution-version>_<date-of-fix>`. Se uma versão principal ou secundária for lançada, você deverá realizar uma atualização completa para obter a versão mais recente da imagem, pois a tag estável é incrementada para corresponder à versão da solução.

Se você optar por atualizações automáticas durante a implantação, as alterações na imagem, incluindo patches de CVE e pequenas correções de erros, serão aplicadas automaticamente até a última versão secundária correspondente.

Versões

Por padrão, essa solução é implantada com as atualizações automáticas desativadas. Isso significa que a versão da imagem do contêiner está bloqueada para a versão específica correspondente à versão da solução implantada, fornecendo controle total sobre as atualizações da versão.

Se você optar por ativar as atualizações automáticas selecionando Sim durante a CloudFormation implantação, a solução receberá automaticamente patches de segurança e correções de erros

menores e ininterruptas até a versão secundária correspondente mais recente. Por exemplo, se você implantar a versão 4.0.0 com as atualizações automáticas ativadas, receberá atualizações até 4.0.x, mas não 4.1.0 ou superior.

Para controlar manualmente a versão da imagem do contêiner, você pode editar a definição da tarefa para especificar uma versão específica da imagem usando o formato da versão marcada. Isso permite fixar em uma versão específica da imagem, independentemente da configuração de atualizações automáticas.

Personalização da imagem do contêiner

Essa solução usa um repositório público de imagens do Amazon Elastic Container Registry (Amazon ECR) gerenciado pela AWS para armazenar a imagem usada para executar os testes configurados. Se quiser personalizar a imagem do contêiner, você pode reconstruir e enviar a imagem para um repositório de imagens ECR em sua própria conta da AWS.

Se quiser personalizar essa solução, você pode usar a imagem padrão do contêiner ou editar esse contêiner de acordo com suas necessidades. Se você personalizar a solução, use o exemplo de código a seguir para declarar as variáveis de ambiente antes de criar sua solução personalizada.

```
#!/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
```

Se você optar por personalizar a imagem do contêiner, poderá hospedá-la em um repositório de imagens privado ou em um repositório público de imagens em sua conta da AWS. Os recursos de imagem estão no `deployment/ecr/distributed-load-testing-on-aws-load-tester` diretório, localizado na base de código.

Você pode criar e enviar a imagem para o destino do host.

- Para repositórios e imagens privadas do Amazon ECR, consulte os repositórios privados e imagens privadas do Amazon ECR no Guia do usuário do Amazon ECR.

- Para repositórios e imagens públicas do Amazon ECR, consulte os repositórios públicos e imagens públicas do Amazon ECR no Guia do usuário público do Amazon ECR.

Depois de criar sua própria imagem, você pode declarar as seguintes variáveis de ambiente antes de criar sua solução personalizada.

```
#!/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, v3.4.0
```

O exemplo a seguir mostra o arquivo de contêiner.

```
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==78.1.1 h11 urllib3==2.2.2 && \
    $PIP_INSTALL --upgrade bzt
COPY ./.bzt-rc /root/.bzt-rc
RUN chmod 755 /root/.bzt-rc

# 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,robot,newman
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* && \
    rm -rf /usr/local/lib/python3.11/site-packages/setuptools-65.5.0.dist-info && \
    rm -rf /usr/local/lib/python3.11/site-packages/urllib3-1.26.17.dist-info

# 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"]
```

Além de um arquivo de contêiner, o diretório contém o seguinte script bash que baixa a configuração de teste do Amazon S3 antes de executar Taurus/Blazemeter o programa.

```
#!/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=$(grep -oE '[a-f0-9]{32}' /proc/self/cgroup | 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 $JMETER_LIB_PATH
  elif [ "$TEST_TYPE" == "k6" ]; then
    curl --output /tmp/artifacts/k6.rpm https://dl.k6.io/rpm/x86_64/k6-v0.58.0-
amd64.rpm
    rpm -ivh /tmp/artifacts/k6.rpm
    dnf install -y k6
    rm -rf /tmp/artifacts/k6.rpm
    EXT="js"
    KPI_EXT="csv"
    TYPE_NAME="K6"
  elif [ "$TEST_TYPE" == "locust" ]; then
    EXT="py"
    TYPE_NAME="Locust"

  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

    # If zip and locust, make sure to pick locustfile
    if [ "$TEST_TYPE" != "locust" ]; then
      TEST_SCRIPT=$(find . -name "*.${EXT}" | head -n 1)
    else
      TEST_SCRIPT=$(find . -name "locustfile.py" | head -n 1)
    fi
    # 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
```

Além do Dockerfile e do script bash, dois scripts Python também estão incluídos no diretório. Cada tarefa executa um script Python de dentro do script bash. As tarefas de trabalho executam o `ecslistener.py` script, enquanto a tarefa principal executa o `ecscontroller.py` script. O `ecslistener.py` script cria um soquete na porta 50000 e espera por uma mensagem. O

`ecscontroller.py` script se conecta ao soquete e envia a mensagem de início do teste para as tarefas do trabalhador, o que permite que elas iniciem simultaneamente.

API de teste de carga distribuída

Essa solução de teste de carga ajuda você a expor os dados dos resultados do teste de maneira segura. A API atua como uma "porta de entrada" para acesso aos dados de teste armazenados no Amazon DynamoDB. Você também pode usar o APIs para acessar qualquer funcionalidade estendida incorporada à solução.

Essa solução usa um grupo de usuários do Amazon Cognito integrado ao Amazon API Gateway para identificação e autorização. Quando um grupo de usuários é usado com a API, os clientes só podem chamar métodos ativados do grupo de usuários depois de fornecerem um token de identidade válido.

Para obter mais informações sobre a execução de testes diretamente por meio da API, consulte Solicitações de assinatura na documentação de referência da API REST do Amazon API Gateway.

As operações a seguir estão disponíveis na API da solução.

> **Note**
>
> Para obter mais informações `testScenario` e outros parâmetros, consulte cenários e exemplos de carga útil no GitHub repositório.

Informações da pilha

- OBTENHA /stack-info

Cenários

- GET /cenários
- POST /cenários
- OPÇÕES/CENÁRIOS
- GET /scenarios/ {testId}
- POST /scenarios/ {testId}
- EXCLUIR /scenarios/ {testId}

- OPÇÕES /cenários/ {testId}

Execuções de teste

- GET /scenarios/ {testId} /testruns
- GET /scenarios/ {testId} /testruns/ {} testRunId
- EXCLUIR /scenarios/ {testId} /testruns/ {} testRunId

Linha de base

- GET /scenarios/ {testId} /linha de base
- PUT /scenarios/ {testId} /linha de base
- EXCLUIR /scenarios/ {testId} /baseline

Tarefas

- GET /tasks
- OPÇÕES/tarefas

Regiões

- GET /regiões
- OPÇÕES/REGIÕES

OBTENHA /stack-info

Description

A `GET /stack-info` operação recupera informações sobre a pilha implantada, incluindo horário de criação, região e versão. Esse endpoint é usado pelo front-end.

Resposta

200 - Sucesso

Name (Nome)	Description
`created_time`	Carimbo de data/hora ISO 8601 quando a pilha foi criada (por exemplo,) `2025-09-09T19:40:22Z`
`region`	Região da AWS onde a pilha é implantada (por exemplo,) `us-east-1`
`version`	Versão da solução implantada (por exemplo,`v4.0.0`)

Respostas de erro

- `403`- Proibido: permissões insuficientes para acessar as informações da pilha
- `404`- Não encontrado: informações da pilha não disponíveis
- `500`- Erro interno do servidor

GET /cenários

Description

A `GET /scenarios` operação permite que você recupere uma lista de cenários de teste.

Resposta

Name (Nome)	Description
`data`	Uma lista de cenários, incluindo ID, nome, descrição, status, tempo de execução, tags, total de execuções e última execução de cada teste

POST /cenários

Description

A `POST /scenarios` operação permite criar ou agendar um cenário de teste.

Corpo da solicitação

Name (Nome)	Description
`testName`	O nome do teste
`testDescription`	A descrição do teste
`testTaskConfigs`	Um objeto que especifica `concurrency` (o número de execuções paralelas), `taskCount` (o número de tarefas necessárias para executar um teste) e `region` para o cenário
`testScenario`	A definição do teste, incluindo simultaneidade, horário do teste, host e método para o teste
`testType`	O tipo de teste (por exemplo,`simple`,`jmeter`)
`fileType`	O tipo de arquivo de upload (por exemplo,`none`,`script`,`zip`)
`tags`	Uma matriz de sequências de caracteres para categorizar os testes. Campo opcional com um comprimento máximo de 5 (por exemplo,`["blue", "3.0", "critical"]`)
`scheduleDate`	A data para realizar um teste. Fornecido somente ao agendar um teste (por exemplo,`2021-02-28`)
`scheduleTime`	A hora de fazer um teste. Fornecido somente ao agendar um teste (por exemplo,`21:07`)

Name (Nome)	Description
`scheduleStep`	A etapa do processo de agendamento. Fornecido somente ao agendar um teste recorrente. (As etapas disponíveis incluem `create` `estart`)
`cronvalue`	O valor cron para personalizar o agendamen to recorrente. Se usado, omita ScheduleDate e ScheduleTime.
`cronExpiryDate`	Data obrigatória para que o cron expire e não seja executado indefinidamente.
`recurrence`	A recorrência de um teste agendado. Fornecido somente ao agendar um teste recorrente (por exemplo,,`daily`, `weeklybiweekly`, ou) `monthly`

Resposta

Name (Nome)	Description
`testId`	O ID exclusivo do teste
`testName`	O nome do teste
`status`	O status do teste

OPÇÕES/CENÁRIOS

Description

A `OPTIONS /scenarios` operação fornece uma resposta para a solicitação com os cabeçalhos de resposta CORS corretos.

Resposta

Name (Nome)	Description
`testId`	O ID exclusivo do teste
`testName`	O nome do teste
`status`	O status do teste

GET /scenarios/ {testId}

Description

A `GET /scenarios/{testId}` operação permite que você recupere os detalhes de um cenário de teste específico.

Parâmetros de solicitação

`testId`

- O ID exclusivo do teste

  Tipo: String

  Obrigatório: Sim

`latest`

- Parâmetro de consulta para retornar somente a última execução do teste. O padrão é `true`

  Tipo: booliano

  Obrigatório: não

`history`

- Parâmetro de consulta para incluir o histórico de execução do teste na resposta. O padrão é `true`. Defina como `false` para excluir o histórico

  Tipo: booliano

  Obrigatório: não

Resposta

Name (Nome)	Description
`testId`	O ID exclusivo do teste
`testName`	O nome do teste
`testDescription`	A descrição do teste
`testType`	O tipo de teste executado (por exemplo,`simple`,`jmeter`)
`fileType`	O tipo de arquivo que é carregado (por exemplo,`none`,`script`,`zip`)
`tags`	Uma matriz de sequências de caracteres para categorizar testes
`status`	O status do teste
`startTime`	A hora e a data em que o último teste começou
`endTime`	A hora e a data em que o último teste terminou
`testScenario`	A definição do teste, incluindo simultaneidade, horário do teste, host e método para o teste
`taskCount`	O número de tarefas necessárias para executar o teste
`taskIds`	Uma lista de tarefas IDs para executar testes
`results`	Os resultados finais do teste
`history`	Uma lista dos resultados finais de testes anteriores (excluídos quando`history=false`)

Name (Nome)	Description
`totalRuns`	O número total de execuções de teste para esse cenário
`lastRun`	A data e hora da última execução do teste
`errorReason`	Uma mensagem de erro gerada quando ocorre um erro
`nextRun`	A próxima execução programada (por exemplo,`2017-04-22 17:18:00`)
`scheduleRecurrence`	A recorrência do teste (por exemplo,,`daily`, `weeklybiweekly`,`monthly`)

POST /scenarios/ {testId}

Description

A `POST /scenarios/{testId}` operação permite que você cancele um cenário de teste específico.

Parâmetro de solicitação

`testId`

- O ID exclusivo do teste

  Tipo: String

  Obrigatório: Sim

Resposta

Name (Nome)	Description
`status`	O status do teste

EXCLUIR /scenarios/ {testId}

Description

A `DELETE /scenarios/{testId}` operação permite que você exclua todos os dados relacionados a um cenário de teste específico.

Parâmetro de solicitação

`testId`

- O ID exclusivo do teste

  Tipo: String

  Obrigatório: Sim

Resposta

Name (Nome)	Description
`status`	O status do teste

OPÇÕES /cenários/ {testId}

Description

A `OPTIONS /scenarios/{testId}` operação fornece uma resposta para a solicitação com os cabeçalhos de resposta CORS corretos.

Resposta

Name (Nome)	Description
`testId`	O ID exclusivo do teste
`testName`	O nome do teste

Name (Nome)	Description
`testDescription`	A descrição do teste
`testType`	O tipo de teste executado (por exemplo,`simple`,`jmeter`)
`fileType`	O tipo de arquivo que é carregado (por exemplo,`none`,`script`,`zip`)
`status`	O status do teste
`startTime`	A hora e a data em que o último teste começou
`endTime`	A hora e a data em que o último teste terminou
`testScenario`	A definição do teste, incluindo simultaneidade, horário do teste, host e método para o teste
`taskCount`	O número de tarefas necessárias para executar o teste
`taskIds`	Uma lista de tarefas IDs para executar testes
`results`	Os resultados finais do teste
`history`	Uma lista dos resultados finais dos testes anteriores
`errorReason`	Uma mensagem de erro gerada quando ocorre um erro

GET /scenarios/ {testId} /testruns

Description

A `GET /scenarios/{testId}/testruns` operação recupera a execução do teste IDs para um cenário de teste específico, opcionalmente filtrada por intervalo de tempo. Quando`latest=true`, retorna somente a única execução de teste mais recente.

Parâmetros de solicitação

`testId`

- O ID do cenário de teste

  Tipo: String

  Obrigatório: Sim

`latest`

- Retorna somente a ID de execução de teste mais recente

  Tipo: booliano

  Padrão: `false`

  Exigido: Não

`start_timestamp`

- Registro de data e hora ISO 8601 para filtrar as execuções de teste (inclusive). Por exemplo, `2024-01-01T00:00:00Z`.

  Tipo: Cadeia de caracteres (formato de data e hora)

  Obrigatório: não

`end_timestamp`

- Registro de data e hora ISO 8601 para filtrar os testes até (inclusive). Por exemplo, `2024-12-31T23:59:59Z`.

  Tipo: Cadeia de caracteres (formato de data e hora)

  Obrigatório: não

`limit`

- Número máximo de execuções de teste a serem retornadas (ignorado quando`latest=true`)

  Tipo: Inteiro (mínimo: 1, máximo: 100)

  Padrão: `20`

  Exigido: Não

`next_token`

- Token de paginação da resposta anterior para obter a próxima página

  Tipo: string

  Obrigatório: não

Resposta

200 - Sucesso

Name (Nome)	Description
`testRuns`	Matriz de objetos de execução de teste, cada um contendo `testRunId` (string) e `startTime` (data e hora ISO 8601)
`pagination`	Objeto contendo `limit` (inteiro) e `next_token` (string ou nulo). O token é nulo se não houver mais resultados

Respostas de erro

- `400`- Formato ou parâmetros de timestamp inválidos
- `404`- Cenário de teste não encontrado
- `500`- Erro interno do servidor

Exemplo de uso

- Somente a última execução do teste: `GET /scenarios/test123/testruns?latest=true`
- Último dentro do intervalo de tempo: `GET /scenarios/test123/testruns?latest=true&start_timestamp=2024-01-01T00:00:00Z`
- Solicitação da próxima página: `GET /scenarios/test123/testruns?limit=20&next_token=eyJ0ZXN0SWQiOiJzZVFVeTEyTEtMIiwic3RhcnRUaW1lIjoiMjAyNC0wMS`

GET /scenarios/ {testId} /testruns/ {} testRunId

Description

A `GET /scenarios/{testId}/testruns/{testRunId}` operação recupera resultados e métricas completos para uma execução de teste específica. Opcionalmente, omita os resultados do histórico `history=false` para uma resposta mais rápida.

Parâmetros de solicitação

`testId`

- O ID do cenário de teste

  Tipo: String

  Obrigatório: Sim

`testRunId`

- O ID específico da execução do teste

  Tipo: String

  Obrigatório: Sim

`history`

- Inclua uma matriz de histórico na resposta. Defina como `false` para omitir o histórico para uma resposta mais rápida

  Tipo: booliano

  Padrão: `true`

  Exigido: Não

Resposta

200 - Sucesso

Name (Nome)	Description
`testId`	O ID exclusivo do teste (por exemplo,`seQUy12LKL`)
`testRunId`	O ID específico da execução do teste (por exemplo,`2DEwHItEne`)
`testDescription`	Descrição do teste de carga
`testType`	O tipo de teste (por exemplo,`simple`,`jmeter`)
`status`	O status da execução do teste:`complete`,`running`,`failed`, ou `cancelled`
`startTime`	A hora e a data em que o teste começou (por exemplo,`2025-09-09 21:01:00`)
`endTime`	A hora e a data em que o teste terminou (por exemplo,`2025-09-09 21:18:29`)
`succPercent`	Porcentagem de sucesso (por exemplo,`100.00`)
`testTaskConfigs`	Matriz de objetos de configuração de tarefas contendo `regiontaskCount` , e `concurrency`
`completeTasks`	Regiões de mapeamento de objetos para contagens de tarefas concluídas
`results`	Objeto contendo métricas detalhadas, incluindo `avg_lt` (latência média), percentis (`p0_0`,,`p50_0`,`p90_0`,`p95_0`,`p100_0`) `p99_0p99_9`, `avg_rt` (tempo médio de resposta), (tempo médio de conexão), `avg_ct` `stdev_rt` (tempo de resposta com desvio padrão),,`concurrency` , (contagem de

Name (Nome)	Description
	sucesso)`throughput` , `succ` (contagem de falhas),`fail`,, `bytes` `testDuration` `metricS3Location` , `rc` (matriz de códigos de resposta) e matriz `labels`
`testScenario`	Objeto contendo configuração de teste com`execution` ,`reporting` , e `scenarios` propriedades
`history`	Conjunto de resultados históricos de testes (excluídos quando`history=false`)

Respostas de erro

- `400`- TestId inválido ou testRunId
- `404`- Execução de teste não encontrada
- `500`- Erro interno do servidor

EXCLUIR /scenarios/ {testId} /testruns/ {} testRunId

Description

A `DELETE /scenarios/{testId}/testruns/{testRunId}` operação exclui todos os dados e artefatos relacionados a uma execução de teste específica. Os dados da execução do teste são removidos do DynamoDB, enquanto os dados reais do teste no S3 permanecem inalterados.

Parâmetros de solicitação

`testId`

- O ID do cenário de teste

  Tipo: String

  Obrigatório: Sim

`testRunId`

- O ID de execução de teste específico a ser excluído

  Tipo: String

  Obrigatório: Sim

Resposta

204 - Sucesso

A execução do teste foi excluída com sucesso (nenhum conteúdo retornado)

Respostas de erro

- `400`- TestId inválido ou testRunId
- `403`- Proibido: permissões insuficientes para excluir a execução do teste
- `404`- Execução de teste não encontrada
- `409`- Conflito: o teste está sendo executado no momento e não pode ser excluído
- `500`- Erro interno do servidor

GET /scenarios/ {testId} /linha de base

Description

A `GET /scenarios/{testId}/baseline` operação recupera o resultado do teste de linha de base designado para um cenário. Retorna o ID da execução do teste de linha de base ou os resultados completos da linha de base, dependendo do `data` parâmetro.

Parâmetros de solicitação

`testId`

- O ID do cenário de teste

  Tipo: String

  Obrigatório: Sim

`data`

- Retorne os dados completos da execução do teste de linha de base se`true`, caso contrário, apenas o testRunId

  Tipo: booliano

  Padrão: `false`

  Exigido: Não

Resposta

200 - Sucesso

Quando `data=false` (padrão):

Name (Nome)	Description
`testId`	O ID do cenário de teste (por exemplo,`seQUy12LKL`)
`baselineTestRunId`	O ID de execução do teste de linha de base (por exemplo,`2DEwHItEne`)

Quando`data=true`:

Name (Nome)	Description
`testId`	O ID do cenário de teste (por exemplo,`seQUy12LKL`)
`baselineTestRunId`	O ID de execução do teste de linha de base (por exemplo,`2DEwHItEne`)
`baselineData`	Objeto completo dos resultados da execução do teste (mesma estrutura que`GET /scenarios/{testId}/testruns/{testRunId}`)

Respostas de erro

- `400`- Parâmetro testId inválido
- `404`- Cenário de teste não encontrado ou nenhuma linha de base definida
- `500`- Erro interno do servidor

PUT /scenarios/ {testId} /linha de base

Description

A `PUT /scenarios/{testId}/baseline` operação designa uma execução de teste específica como linha de base para comparação de desempenho. Somente uma linha de base pode ser definida por cenário.

Parâmetros de solicitação

`testId`

- O ID do cenário de teste

  Tipo: String

  Obrigatório: Sim

Corpo da solicitação

Name (Nome)	Description
`testRunId`	O ID da execução do teste a ser definido como linha de base (por exemplo,`2DEwHItEne`)

Resposta

200 - Sucesso

Name (Nome)	Description
`message`	Mensagem de confirmação (por exemplo,`Baseline set successfully`)
`testId`	O ID do cenário de teste (por exemplo,`seQUy12LKL`)
`baselineTestRunId`	O ID de execução do teste de linha de base que foi definido (por exemplo,`2DEwHItEne`)

Respostas de erro

- `400`- TestId inválido ou testRunId
- `404`- Cenário de teste ou execução de teste não encontrado
- `409`- Conflito: a execução do teste não pode ser definida como linha de base (por exemplo, teste com falha)
- `500`- Erro interno do servidor

EXCLUIR /scenarios/ {testId} /baseline

Description

A `DELETE /scenarios/{testId}/baseline` operação limpa o valor da linha de base de um cenário definindo-o como uma string vazia.

Parâmetros de solicitação

`testId`

- O ID do cenário de teste

  Tipo: String

  Obrigatório: Sim

Resposta

204 - Sucesso

Linha de base eliminada com sucesso (nenhum conteúdo retornado)

Respostas de erro

- `400`- TestId inválido
- `500`- Erro interno do servidor

GET /tasks

Description

A `GET /tasks` operação permite que você recupere uma lista de tarefas em execução do Amazon Elastic Container Service (Amazon ECS).

Resposta

Name (Nome)	Description
`tasks`	Uma lista de tarefas IDs para executar testes

OPÇÕES/tarefas

Description

A operação de `OPTIONS /tasks` tarefas fornece uma resposta para a solicitação com os cabeçalhos de resposta CORS corretos.

Resposta

Name (Nome)	Description
`taskIds`	Uma lista de tarefas IDs para executar testes

GET /regiões

Description

A `GET /regions` operação permite que você recupere as informações de recursos regionais necessárias para executar um teste nessa região.

Resposta

Name (Nome)	Description
`testId`	O ID da região
`ecsCloudWatchLogGroup`	O nome do grupo de CloudWatch registros da Amazon para as tarefas do Amazon Fargate na região
`region`	A região na qual existem os recursos na tabela
`subnetA`	O ID de uma das sub-redes na região
`subnetB`	O ID de uma das sub-redes na região
`taskCluster`	O nome do cluster AWS Fargate na região
`taskDefinition`	O ARN da definição da tarefa na região
`taskImage`	O nome da imagem da tarefa na Região
`taskSecurityGroup`	O ID do grupo de segurança na região

OPÇÕES/REGIÕES

Description

A `OPTIONS /regions` operação fornece uma resposta para a solicitação com os cabeçalhos de resposta CORS corretos.

Resposta

Name (Nome)	Description
`testId`	O ID da região
`ecsCloudWatchLogGroup`	O nome do grupo de CloudWatch registros da Amazon para as tarefas do Amazon Fargate na região
`region`	A região na qual existem os recursos na tabela
`subnetA`	O ID de uma das sub-redes na região
`subnetB`	O ID de uma das sub-redes na região
`taskCluster`	O nome do cluster AWS Fargate na região
`taskDefinition`	O ARN da definição da tarefa na região
`taskImage`	O nome da imagem da tarefa na Região
`taskSecurityGroup`	O ID do grupo de segurança na região

Aumente os recursos do contêiner

Para aumentar o número de usuários virtuais simultâneos (simultaneidade) que seus testes de carga podem simular, você precisa aumentar os recursos de CPU e memória alocados para cada tarefa do Amazon ECS. Isso envolve a criação de uma nova revisão da definição de tarefa com limites de recursos mais altos e, em seguida, a atualização da configuração do DynamoDB da solução para usar a nova definição de tarefa em futuros testes.

Criar uma nova revisão de definição de tarefa

Siga estas etapas para criar uma nova definição de tarefa com maiores recursos de CPU e memória:

1. Faça login no console do Amazon Elastic Container Service.
2. No menu de navegação à esquerda, selecione Definições de tarefas.

3. Marque a caixa de seleção ao lado da definição da tarefa que corresponde a essa solução. Por exemplo, `[replaceable]` <stackName>`- EcsTaskDefinition -*<system-generated-random-Hash>*.
4. Escolha Create new revisional (Criar nova revisão).
5. Na página Criar nova revisão, execute as seguintes ações:
   a. Em Tamanho da tarefa, modifique a memória da tarefa e a CPU da tarefa para os valores desejados. Valores mais altos permitem mais usuários virtuais simultâneos por tarefa.
   b. Em Definições de contêiner, revise os limites de memória rígida/flexível. Se esse limite for menor que a memória desejada, escolha o contêiner.
   c. Na caixa de diálogo Editar contêiner, acesse Limites de memória e atualize o Limite rígido para que corresponda ou seja menor que a alocação de memória da tarefa.
   d. Selecione Atualizar.
6. Na página Criar nova revisão, escolha Criar.
7. Depois que a definição da tarefa for criada com sucesso, registre o ARN completo da definição da tarefa, incluindo o número da versão. Por exemplo: `[replaceable]` <stackName>`- EcsTaskDefinition -*<system-generated-random-Hash>*: [substituível]`<system-generated-versionNumber>`.

Atualizar a tabela do DynamoDB

Depois de criar a nova revisão da definição de tarefa, você deve atualizar a tabela do DynamoDB da solução para que futuros testes usem a nova definição de tarefa. Repita essas etapas para cada região da AWS em que você deseja usar a definição de tarefa atualizada:

1. Navegue até o console do DynamoDB.
2. No painel de navegação esquerdo, selecione Explorar itens em Tabelas.
3. Selecione a tabela do `scenarios-table` DynamoDB associada a essa solução. Por exemplo, `[replaceable]` <stackName>`- DLTTest RunnerStorage DLTScenarios Tabela-*<system-generated-random-Hash>*.
4. Selecione o item que corresponde à Região na qual você criou a nova revisão da definição de tarefa. Por exemplo, `region-[replaceable]` <region-name>`.
5. No editor do item, localize o atributo TaskDefinition e atualize seu valor com o ARN completo da definição da tarefa que você registrou na seção anterior (incluindo o número da versão).
6. Escolha Salvar alterações.

> **Note**
>
> A definição de tarefa atualizada só será usada para novas execuções de teste. Todos os testes atualmente em execução ou agendados continuarão usando a definição de tarefa anterior.

Especificação de ferramentas MCP

A solução Distributed Load Testing expõe um conjunto de ferramentas MCP que permitem que agentes de IA interajam com cenários e resultados de testes. Essas ferramentas fornecem recursos abstratos de alto nível que se alinham à forma como os agentes de IA processam as informações, permitindo que eles se concentrem em análises e insights em vez de contratos detalhados de API.

> **Note**
>
> Todas as ferramentas MCP fornecem acesso somente de leitura aos dados da solução. Nenhuma modificação nos cenários ou configurações de teste é suportada pela interface MCP.

cenários_de_lista

Description

A `list_scenarios` ferramenta recupera uma lista de todos os cenários de teste disponíveis com metadados básicos.

Endpoint

`GET /scenarios`

Parâmetros

Nenhum

Resposta

Name (Nome)	Description
`testId`	Identificador exclusivo para o cenário de teste
`testName`	Nome do cenário de teste
`status`	Status atual do cenário de teste
`startTime`	Quando o teste foi criado ou executado pela última vez
`testDescription`	Descrição do cenário de teste

get_scenario_details

Description

A `get_scenario_details` ferramenta recupera a configuração do teste e a execução mais recente do teste para um único cenário de teste.

Endpoint

```
GET /scenarios/<test_id>?history=false&results=false
```

Parâmetro de solicitação

`test_id`

- O identificador exclusivo para o cenário de teste

  Tipo: String

  Obrigatório: Sim

Resposta

Name (Nome)	Description
`testTaskConfigs`	Configuração de tarefas para cada região
`testScenario`	Definição e parâmetros do teste
`status`	Status atual do teste
`startTime`	Carimbo de data e hora de início do teste
`endTime`	Carimbo de data e hora de término do teste (se concluído)

execuções de teste de lista

Description

A `list_test_runs` ferramenta recupera uma lista de execuções de teste para um cenário de teste específico, classificadas da mais recente para a mais antiga. Retorna no máximo 30 resultados.

Endpoint

```
GET /scenarios/<testid>/testruns/?limit=<limit>
```

or

```
GET /scenarios/<testid>/testruns/?
limit=30&start_date=<start_date>&end_date=<end_date>
```

Parâmetros de solicitação

`test_id`

- O identificador exclusivo para o cenário de teste

  Tipo: String

  Obrigatório: Sim

`limit`

- Número máximo de execuções de teste a serem retornadas

  Tipo: inteiro

  Padrão: 20

  Máximo: 30

  Obrigatório: não

`start_date`

- Registro de data e hora ISO 8601 para filtrar execuções a partir de uma data específica

  Tipo: Cadeia de caracteres (formato de data e hora)

  Obrigatório: não

`end_date`

- Carimbo de data/hora ISO 8601 para filtrar execuções até uma data específica

  Tipo: Cadeia de caracteres (formato de data e hora)

  Obrigatório: não

Resposta

Name (Nome)	Description
`testRuns`	Conjunto de resumos de testes com métricas de desempenho e percentis para cada execução

get_test_run

Description

A `get_test_run` ferramenta recupera resultados detalhados de um único teste executado com detalhamentos regionais e de endpoints.

Endpoint

`GET /scenarios/<testid>/testruns/<testrunid>`

Parâmetros de solicitação

`test_id`

- O identificador exclusivo para o cenário de teste

  Tipo: String

  Obrigatório: Sim

`test_run_id`

- O identificador exclusivo para a execução de teste específica

  Tipo: String

  Obrigatório: Sim

Resposta

Name (Nome)	Description
`results`	Dados completos da execução do teste, incluindo análise dos resultados regionais, métricas específicas do endpoint, percentis de desempenho (p50, p90, p95, p99), contagens de sucesso e falha, tempos de resposta e latência e configuração de teste usada para a execução

get_latest_test_run

Description

A `get_latest_test_run` ferramenta recupera o teste mais recente para um cenário de teste específico.

Endpoint

```
GET /scenarios/<testid>/testruns/?limit=1
```

> **Note**
>
> Os resultados são classificados por tempo usando um Índice Secundário Global (GSI), garantindo que o teste mais recente seja retornado.

Parâmetro de solicitação

`test_id`

- O identificador exclusivo para o cenário de teste

  Tipo: String

  Obrigatório: Sim

Resposta

Name (Nome)	Description
`results`	Dados de execução de teste mais recentes com o mesmo formato de `get_test_run`

get_baseline_test_run

Description

A `get_baseline_test_run` ferramenta recupera a execução do teste de linha de base para um cenário de teste específico. A linha de base é usada para fins de comparação de desempenho.

Endpoint

```
GET /scenarios/<test_id>/baseline
```

Parâmetro de solicitação

`test_id`

- O identificador exclusivo para o cenário de teste

  Tipo: String

  Obrigatório: Sim

Resposta

Name (Nome)	Description
`baselineData`	Dados de execução de teste de linha de base para fins de comparação, incluindo todas as métricas e configurações da execução de linha de base designada

get_test_run_artefacts

Description

A `get_test_run_artifacts` ferramenta recupera informações do bucket do Amazon S3 para acessar artefatos de teste, incluindo registros, arquivos de erro e resultados.

Endpoint

`GET /scenarios/<testid>/testruns/<testrunid>`

Parâmetros de solicitação

`test_id`

- O identificador exclusivo para o cenário de teste

  Tipo: String

  Obrigatório: Sim

`test_run_id`

- O identificador exclusivo para a execução de teste específica

  Tipo: String

  Obrigatório: Sim

Resposta

Name (Nome)	Description
`bucketName`	Nome do bucket do S3 em que os artefatos são armazenados
`testRunPath`	Prefixo de caminho para o armazenamento atual de artefatos (versão 4.0+)
`testScenarioPath`	Prefixo de caminho para armazenamento de artefatos legados (pré-versão 4.0)

> **Note**
>
> Todas as ferramentas MCP aproveitam os endpoints de API existentes. Nenhuma modificação no subjacente APIs é necessária para oferecer suporte à funcionalidade MCP.

Referência

Esta seção inclui informações sobre coleta de dados, indicadores para recursos relacionados e uma lista dos criadores que contribuíram para essa solução.

Coleta de dados

Essa solução envia métricas operacionais para a AWS (os “Dados”) sobre o uso dessa solução. Usamos esses dados para entender melhor como os clientes usam essa solução e os serviços e produtos relacionados. A coleta desses dados pela AWS está sujeita ao Aviso de Privacidade da AWS.

Colaboradores

- Tom Nightingale
- Fernando Dingler
- Beomseok Lee
- George Lenz
- Erin McGill
- Dimitri López
- Kamyar Ziabari
- Bassem Wanis
- Garvit Singh
- Nikhil Reddy
- Simon Kroll
- Ahern Knox
- Ian Downard
- Owen Brady
- Jim Thario
- Thyag Ramachandran
- Yang Qin
- James Wang

Glossário

Este glossário define acrônimos e abreviações usados em todo o Guia de Implementação do Distributed Load Testing on AWS.

Protocolos e formatos técnicos

AGPL

Licença Pública Geral Affero. Uma licença de software de código aberto usada pela K6.

solicitações de

Interface de programação de aplicativos. Um conjunto de protocolos e ferramentas para criar aplicativos de software e permitir a comunicação entre diferentes sistemas.

CLI

Interface de linha de comando. Uma interface baseada em texto para interagir com software e sistemas operacionais.

NÚCLEOS

Compartilhamento de recursos entre origens. Um recurso de segurança que permite ou restringe aplicativos web executados em uma origem para acessar recursos de uma origem diferente.

CSV

Valores separados por vírgula. Um formato de arquivo usado para armazenar dados tabulares em texto simples, normalmente usado para exportação de dados.

gRPC

Chamada de procedimento remoto gRPC. Uma estrutura de código aberto de alto desempenho para chamadas de procedimentos remotos.

HTTP

Protocolo de transferência de hipertexto. O protocolo básico usado para transmitir dados na World Wide Web.

HTTPS

HTTP seguro. Uma extensão do HTTP que usa criptografia para comunicação segura em uma rede.

JSON

JavaScript Notação de objeto. Um formato leve de intercâmbio de dados que é fácil para humanos lerem e escreverem e fácil para máquinas analisarem e gerarem.

JWT

Token da Web JSON. Um meio compacto e seguro de representar reivindicações a serem transferidas entre duas partes para autenticação e autorização.

OAuth

Autorização aberta. Um padrão aberto para delegação de acesso comumente usado para autenticação e autorização baseadas em tokens.

REST

Transferência estadual representacional. Um estilo arquitetônico para projetar aplicativos em rede usando comunicação sem estado e métodos HTTP padrão.

SSE

Eventos enviados pelo servidor. Uma tecnologia push de servidor que permite que um cliente receba atualizações automáticas de um servidor por meio de uma conexão HTTP.

Interface do usuário

Interface de usuário. Os elementos visuais e controles por meio dos quais os usuários interagem com aplicativos de software.

URL

Localizador uniforme de recursos. O endereço usado para acessar recursos na Internet.

XML

Linguagem de marcação extensível. Uma linguagem de marcação que define regras para codificar documentos em um formato que seja legível por humanos e por máquina.

Termos de teste e banco de dados

FTP

Protocolo de transferência de arquivos. Um protocolo de rede padrão usado para transferir arquivos entre um cliente e um servidor.

GSI

Índice secundário global. Um recurso do DynamoDB que permite consultar dados usando uma chave alternativa.

JDBC

Conectividade do banco de dados Java. Uma API Java para conectar e executar consultas com bancos de dados.

JMS

Serviço de mensagens Java. Uma API Java para enviar mensagens entre dois ou mais clientes.

TPS

Transações por segundo. Uma medida do número de transações que um sistema pode processar em um segundo.

AWS e os termos do sistema

ARN

Nome de recurso da Amazon. Um identificador exclusivo para recursos da AWS usado para especificar recursos em todos os serviços da AWS.

ISO

Organização Internacional de Padronização. Uma organização independente e não governamental que desenvolve padrões internacionais. Referenciado neste guia para o formato de carimbo de data/hora ISO 8601.

SLA

Contrato de nível de serviço. Um compromisso entre um provedor de serviços e um cliente que define o nível de serviço esperado.

UUID

Identificador universalmente exclusivo. Um número de 128 bits usado para identificar informações de forma exclusiva em sistemas de computador.

vCPU

Unidade de processamento central virtual. Um processador virtual atribuído a uma máquina virtual ou contêiner, representando uma parte da capacidade de processamento da CPU física.

Termos de teste de carga

concurrency

O número de usuários virtuais simultâneos por tarefa. Esse parâmetro controla quantos usuários simulados cada tarefa do Fargate gera durante um teste de carga.

pilha regional

Uma CloudFormation pilha implantada em uma região da AWS para fornecer infraestrutura de teste para testes de carga em várias regiões.

contagem de tarefas

O número de contêineres Fargate (tarefas) lançados para executar um cenário de teste. A carga total gerada é igual à contagem de tarefas multiplicada pela simultaneidade.

cenário de teste

Um teste de carga configurado, incluindo tipo de teste, endpoints de destino, contagem de tarefas, simultaneidade, duração e outros parâmetros.

Revisões

Visite o CHANGELOG.md em nosso GitHub repositório para acompanhar melhorias e correções específicas da versão.

Avisos

Os clientes são responsáveis por fazer uma avaliação independente das informações contidas neste documento. Este documento: (a) serve apenas para fins informativos, (b) representa as ofertas e práticas atuais de produtos da AWS, que estão sujeitas a alterações sem aviso prévio, e (c) não cria nenhum compromisso ou garantia da AWS e de suas afiliadas, fornecedores ou licenciadores. Os produtos ou serviços da AWS são fornecidos “no estado em que se encontram”, sem garantias, representações ou condições de qualquer tipo, expressas ou implícitas. As responsabilidades e obrigações da AWS para com seus clientes são controladas pelos contratos da AWS, e este documento não faz parte nem modifica nenhum acordo entre a AWS e seus clientes.

O teste de carga distribuído na AWS é licenciado de acordo com os termos da Licença Apache Versão 2.0, disponível na The Apache Software Foundation.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.