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: novembro de 2019

O teste de carga distribuído na AWS ajuda você a automatizar o teste de seus aplicativos de software em grande escala e em carga para identificar gargalos antes de lançar seu aplicativo. Essa solução cria e simula milhares de usuários conectados, gerando registros transacionais em um ritmo constante, sem a necessidade de provisionar servidores.

Essa solução utiliza o Amazon Elastic Container Service (Amazon ECS) no AWS Fargate para implantar contêineres que podem executar todas as suas simulações e oferece os seguintes recursos:

- Implante o Amazon ECS em contêineres do AWS Fargate que podem ser executados de forma independente para testar os recursos de carga do software que está sendo testado.
- Simule dezenas de milhares de usuários conectados em várias regiões da AWS, gerando registros transacionais em um ritmo contínuo.
- Personalize seus testes de aplicativos criando JMeter scripts personalizados.
- Agende os testes de carga para que comecem automaticamente em uma data futura ou em datas recorrentes.
- Execute os testes de carga do aplicativo simultaneamente ou execute vários testes simultaneamente.

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 essas perguntas:

Se você deseja...	Leia…
Conheça 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
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
Acesse o código-fonte e, opcionalmente, use o AWS Cloud Development Kit (AWS CDK) para implantar a solução.	GitHub repositório

Atributos

A solução fornece os seguintes atributos:

Out-of-the-Box Testes de desempenho configuráveis

Inclui testes de desempenho pré-configurados disponíveis para uso imediato.

Testes de aplicação personalizáveis

Permite a personalização de testes flexível e precisa para identificar possíveis problemas e adaptar os testes a requisitos e cenários específicos usando JMeter scripts.

Simula a alta carga de usuários

Capaz de simular dezenas de milhares de usuários conectados para testar o estresse de seu aplicativo.

Geração contínua de transações

Gera registros transacionais continuamente para avaliar o desempenho sob carga constante.

monitoramento em tempo real

Fornece monitoramento em tempo real do progresso e dos resultados do teste e permite que você agende os testes para serem iniciados automaticamente em datas especificadas ou em intervalos recorrentes.

Simulação de solicitação regional

Simule solicitações de usuários de qualquer região para avaliar o desempenho global.

Flexibilidade do endpoint

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

Resultados detalhados do teste

Visualize resultados de testes abrangentes, incluindo tempo médio de resposta, número de usuários simultâneos, solicitações bem-sucedidas e solicitações malsucedidas.

Console Web intuitivo

Oferece um console easy-to-use web para gerenciar e monitorar testes.

Suporta vários protocolos

Compatível com vários protocolos WebSocket, como HTTP, HTTPS, JDBC, JMS, FTP e gRPC.

Benefícios

A solução oferece os seguintes benefícios:

Suporta testes de desempenho abrangentes

Facilita testes de carga, estresse e resistência para uma avaliação completa da aplicação.

Detecção precoce de problemas de desempenho

Identifica problemas de desempenho e gargalos antes do lançamento da produção.

Simulação de uso no mundo real

Espelha com precisão os padrões de uso do mundo real para destacar gargalos e áreas de otimização.

Insights detalhados de desempenho

Fornece informações sobre desempenho e resiliência do software sob carga significativa.

Avaliação automatizada de desempenho

Permite avaliações regulares de desempenho sem intervenção manual.

Testes econômicos

Oferece um pay-as-you-go modelo, eliminando a necessidade de uma infraestrutura de teste dedicada e taxas de assinatura.

Casos de uso

Simule a carga de produção

Teste aplicativos web e móveis em condições semelhantes às de produção antes de lançar uma nova versão.

Valide o desempenho do aplicativo

Garanta que seu aplicativo possa lidar com o tráfego esperado do usuário sem degradação. Teste os limites do aplicativo usando recursos padrão e avalie a escalabilidade da infraestrutura.

Gerencie cargas de pico

Verifique se sua infraestrutura pode gerenciar picos de carga ou picos de tráfego inesperados, garantindo estabilidade sob alta demanda.

Otimize o desempenho

Entenda o perfil de desempenho do seu aplicativo e identifique gargalos, como execução ineficiente de código, consultas ao banco de dados e latência de rede.

Início rápido do teste

Comece a testar rapidamente com testes out-of-the-box de desempenho.

Testes personalizáveis

Adapte os testes a cenários e requisitos específicos, ajustando o número de usuários simultâneos e tarefas lançadas.

Teste programado

Agende testes para testes de regressão e monitoramento contínuo do desempenho, garantindo o desempenho consistente do aplicativo.

Avaliação de desempenho geográfico

Avalie o desempenho do aplicativo em diferentes regiões geográficas para garantir a eficiência global.

Integração de pipeline de CI/CD

Integre testes de desempenho em seu CI/CD pipeline para testes contínuos e automatizados durante os ciclos de desenvolvimento.

Conceitos e definições

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

cenário

Definição do teste, incluindo nome, descrição, contagem de tarefas, simultaneidade, região da AWS, ramp-up, espera, tipo de teste, data da programação e 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

O número de usuários virtuais simultâneos gerados por tarefa. O limite recomendado com base nas configurações padrão é de 200 usuários virtuais. A simultaneidade é limitada pela CPU e pela memória. Para testes baseados no Apache JMeter, quanto maior o número de usuários virtuais, maior 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 mais baixos de usuários virtuais para 1 tarefa e monitorar as CloudWatch métricas do ECS para o Task Cluster. Consulte as métricas de utilização de clusters do Amazon ECS.

ramp-up

A hora de atingir a simultaneidade desejada.

aguarde

É hora de manter a simultaneidade desejada.

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, que 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 fornecer armazenamento para os detalhes e resultados do cenário de teste e executar cenários de 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 incluem a AmazonLinuximagem de contêiner compatível com a Open Container Initiative (OCI) (com a estrutura de teste de carga do blazemeter instalada), usada para gerar carga para testar o desempenho do seu aplicativo. Taurus/Blazemeter é uma estrutura de automação de testes de código aberto. A imagem do contêiner é hospedada pela AWS em um repositório público do Amazon Elastic Container Registry (Amazon ECR). Para obter mais informações sobre o repositório de imagens ECR, consulte Personalização de imagens de contêiner.

6. Um console web desenvolvido pelo AWS Amplify é implantado em um bucket Amazon 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, essa solução também cria uma função padrão de administrador da solução (função 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 e à API do testador de carga distribuído.
10. Depois de implantar essa solução, você pode usar o console web para criar um cenário de teste que define uma série de tarefas.
11. Os microsserviços usam esse cenário de teste para executar o Amazon ECS em tarefas do AWS Fargate nas regiões especificadas.
12. Além de armazenar os resultados no Amazon S3 e no DynamoDB, quando o teste é concluído, a saída é registrada na Amazon. CloudWatch
13. Se você selecionar a opção de dados ativos, a solução enviará os CloudWatch registros da Amazon para as tarefas do AWS Fargate para uma função Lambda durante o teste, para cada região em que o teste foi executado.
14. Em seguida, 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 você pode ver os dados enquanto o teste é executado no console web.

Considerações sobre o 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.

- Recursos definidos como infraestrutura como uso de código CloudFormation.

- A solução envia métricas para a Amazon CloudWatch em vários estágios para fornecer observabilidade na infraestrutura: funções Lambda, tarefas do Amazon ECS, buckets do Amazon S3 e o restante dos componentes da solução.

Segurança

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

- O Amazon Cognito autentica e autoriza usuários de aplicativos de interface de usuário da web.
- Todas as comunicações entre serviços usam funções aplicáveis do AWS Identity and Access Management (IAM).
- Todas as funções usadas pela solução seguem o acesso com privilégios mínimos. Eles contêm apenas as permissões mínimas necessárias para realizar a tarefa.
- Todo o armazenamento de dados, incluindo os buckets do S3, criptografa os dados em repouso.
- Um grupo de usuários do Amazon Cognito gerencia o acesso dos usuários ao console e aos endpoints do API Gateway do testador de carga distribuído.
- O registro, o rastreamento e o controle de versão são ativados quando aplicável.
- O acesso à rede é privado por padrão, com os endpoints da Amazon Virtual Private Cloud (Amazon VPC) ativados quando disponíveis.

> **Note**
>
> O teste de carga distribuído na AWS cria vários grupos de CloudWatch registros com base no serviço que está sendo usado. O período de retenção dos registros varia de acordo com o armazenamento e o custo do volume de eventos de registro gerados. A solução cria registros de insights de contêineres para as tarefas do ECS; esses registros têm retenção de registros configurada para 1 dia. Os registros criados para os serviços AWS Step Functions, registros personalizados do ECS Load Testing e registros do API Gateway são configurados com um período de retenção de 1 ano. Os registros de tempo de execução do AWS Lambda têm retenção de registros configurada para 2 anos. Os registros de execução do API Gateway têm a retenção de registros configurada para nunca expirar. Você pode alterar os períodos de retenção de registros com base nos seus requisitos no CloudWatch console.

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 excelência operacional.

- 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 custo

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

- 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.

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 dois componentes de alto nível: um front-end e um back-end.

Front-end

O front-end consiste em uma API de teste de carga e um console web que você usa para interagir com o back-end da solução.

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 os dados de teste de forma segura por meio do console web e RESTful da API incluídos. 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 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 console web foi projetado para demonstrar como você pode interagir com essa solução de teste de carga. Em um ambiente de produção, recomendamos personalizar o console web para atender às suas necessidades específicas ou criar seu próprio console.

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.

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 utiliza uma imagem de contêiner criada AmazonLinuxcomo imagem base com a estrutura de teste de carga do blazômetro instalada. Essa imagem está hospedada em um repositório público do Amazon Elastic Container Registry (Amazon ECR). A imagem é usada 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 modelo principal, a solução cria um modelo secundário para lançar os recursos necessários para executar testes em várias regiões. O modelo é armazenado no Amazon S3 e um link para o modelo é fornecido no console web. Os modelos secundários criam uma VPC, um cluster AWS Fargate e uma função Lambda para processar dados ativos.

Para obter mais informações sobre como iniciar uma região secundária, 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 conectados, em várias regiões, gerando um número selecionado de transações por segundo.

Você define os parâmetros para as tarefas que serão executadas como parte do teste usando o console web incluído. A solução usa esses parâmetros para gerar um cenário de teste JSON e o armazena no Amazon S3.

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.

Serviços da AWS nesta solução

Os seguintes serviços da AWS estão incluídos nessa solução:

Serviço da AWS	Descrição
Amazon API Gateway	Principal. Hospeda endpoints da API REST na solução.
AWS CloudFormation	Principal. Gerencia implantações para a infraestrutura da solução.

Serviço da AWS	Descrição
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
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.

Serviço da AWS	Descrição
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.

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 do 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.

Considerações sobre design

Aplicações compatíveis

Essa solução oferece suporte a aplicativos baseados em nuvem e aplicativos locais, desde que você tenha uma conexão de rede da sua conta da AWS com seu aplicativo. A solução oferece suporte para APIs o uso de HTTP ou HTTPS. Você também tem controle sobre os cabeçalhos da solicitação HTTP, para poder adicionar cabeçalhos de autorização ou personalizados para transmitir tokens ou chaves de API.

JMeter suporte a scripts

Ao criar um cenário de teste usando a interface de usuário (UI) dessa solução, você pode usar um script JMeter de teste. Depois de selecionar o arquivo de JMeter script, ele é carregado no bucket <stack-name>-scenariosbucket do Amazon Simple Storage Service (Amazon S3). Quando as tarefas do Amazon Elastic Container Service (Amazon ECS) estão em execução, JMeter o script é baixado do bucket do <stack-name>Amazon S3 -scenariosbucket e o teste é executado.

Se você tiver arquivos JMeter de entrada, poderá compactar os arquivos de entrada 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 seus arquivos JMeter de entrada e 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.

Suporte ao script K6

A solução oferece suporte a testes baseados na estrutura K6. O K6 é lançado com a licença AGPL-3.0. A solução exibe uma mensagem de confirmação de licença ao criar um novo teste para o K6. O arquivo de teste K6 junto com todos os arquivos de entrada necessários podem ser incluídos em um arquivo e enviados para o cenário de teste usando a opção de upload.

Suporte ao script Locust

A solução oferece suporte a testes baseados na estrutura Locust. O arquivo de teste do Locust junto com todos os arquivos de entrada necessários podem ser incluídos em um arquivo e enviados para o cenário de teste usando a opção de upload.

Agendamento de testes

Você pode agendar testes para serem executados em uma data futura ou usar a opção Executar agora. Você pode agendar um teste como uma execução única no futuro ou configurar um teste recorrente no qual você especifica a data da primeira execução e a recorrência planejada. As opções de recorrência incluem: diária, semanal, quinzenal e mensal. Para obter mais informações sobre como o agendamento funciona, consulte a seção Fluxo de trabalho de agendamento de testes deste guia.

A partir da versão 3.3.0, o Distributed Load Testing na AWS permite que os usuários programem testes de carga usando expressões cron. Selecione Executar de acordo com o cronograma e, em seguida, a guia CRON para inserir manualmente um valor cron ou usar os campos suspensos. cronExpiryDate Deve corresponder à data de execução do teste programada. Revise as datas da próxima execução (UTC) para confirmar sua programação.

> **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 inclui um CloudWatch painel da Amazon para cada teste e exibe a saída combinada de todas as tarefas executadas para esse teste no cluster do Amazon ECS em tempo real. O CloudWatch painel exibe o tempo médio de resposta, o número de usuários simultâneos, o número de solicitações bem-sucedidas e o número de solicitações malsucedidas. Cada métrica é agregada a cada segundo e o painel é atualizado 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 o custo, a segurança, as regiões e outras considerações antes da implantação da solução.

Custo

Você é responsável pelo custo dos serviços da AWS usados ao executar essa solução. O custo total da execução dessa solução depende do número de testes de carga executados, da duração desses testes de carga e da quantidade de dados usados como parte dos testes. A partir dessa revisão, o custo da execução dessa 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 value=SO0062. 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.

> **Important**
>
> A partir da versão 1.3.0, a CPU é aumentada para 2 vCPU e a memória é aumentada para 4 GB. Essas alterações aumentam o custo estimado em comparação com as versões anteriores dessa solução. Se seus testes de carga não exigirem esses aumentos em seus recursos da AWS, você poderá reduzi-los. Para obter informações adicionais, 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.

Os preços estão sujeitos a alterações. Para obter detalhes completos, consulte a página de preços de cada serviço da AWS que você usará nesta solução.

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 sua carga operacional porque a AWS opera, gerencia e controla os componentes, incluindo o sistema operacional do host, a camada de virtualização e a segurança física das instalações nas quais os serviços operam. Para obter mais informações sobre a segurança da AWS, acesse AWS Cloud Security.

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.

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.

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 da Amazon para outros locais, como outras EC2 instâncias da 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.

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)
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)

Nome da região	
Ásia-Pacífico (Sydney)	América do Sul (São Paulo)

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 ver as cotas de serviço de todos os serviços da AWS na documentação sem trocar de página, veja as informações na página de endpoints e cotas do serviço 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 AmazonLinux imagem (com o Blazemeter instalado) do contêiner 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 inclui um CloudWatch painel da Amazon para cada teste e exibe a saída combinada de todas as tarefas executadas para esse teste no cluster do Amazon ECS em tempo real. O CloudWatch painel exibe o tempo médio de resposta, o número de usuários simultâneos, o número de solicitações bem-sucedidas e o número de solicitações malsucedidas. Cada métrica é agregada a cada segundo e o painel é atualizado a cada minuto.

Política de EC2 testes da Amazon

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 EC2 testes da Amazon.

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

Ativando ou configurando CloudWatch alarmes

É recomendável monitorar continuamente os recursos da solução após a implantação. Você pode considerar a configuração de CloudWatch métricas para os recursos criados pela solução.

Métricas CloudFront de distribuição da Amazon

Analise as métricas de CloudFront distribuição e configure alarmes para monitorar e receber notificações.

Métricas para Amazon API Gateway

Analise as dimensões e métricas do Amazon API Gateway e configure alarmes para monitorar e receber notificações.

Implante a solução

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.

Visão geral do processo de implantação

Siga as step-by-step instruções nesta seção para configurar e implantar a solução em sua conta.

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

Tempo para implantação: aproximadamente 15 minutos

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

> **Important**
>
> Se você estiver atualizando a pilha de uma versão anterior à v3.2.6 para a versão mais recente, leia esta seção antes de atualizar a pilha.

Antes de iniciar a implantação automatizada, revise a arquitetura e outras considerações discutidas neste guia. Siga as step-by-step instruções nesta seção para configurar e implantar testes de carga distribuídos na AWS em sua conta.

Tempo para implantação: aproximadamente 15 minutos

> **Important**
>
> Essa solução inclui uma opção para enviar métricas operacionais anônimas 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.
> Para optar por não usar esse recurso, baixe o modelo, modifique a seção de CloudFormation mapeamento da AWS e use o CloudFormation console da AWS para carregar seu modelo atualizado e implantar a solução. Para mais informações, consulte a seção Coleta de dados anonimizados deste guia.

Esse CloudFormation modelo automatizado da AWS implanta testes de carga distribuídos na 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.

1. Faça login no AWS Management Console e selecione o botão abaixo para iniciar o CloudFormation modelo distributed-load-testing-on -aws AWS.

Como alternativa, você também 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.

Parameter	Padrão	Descrição
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

Parameter	Padrão	Descrição
		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 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 uma 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

Parameter	Padrão	Descrição
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 aos contêineres do Amazon ECS.
Atualização automática da imagem do contêiner	`Yes`	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.

6. Escolha Avançar.
7. Na página Configurar opções de pilha, selecione Avançar.
8. Na página Revisar, verifique e confirme as configurações. Marque a caixa 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 para implantação: aproximadamente 5 minutos

Você pode executar testes em várias regiões. Quando você implanta a solução Distributed Load Testing, ela cria três buckets Amazon S3. A solução cria uma pilha regional secundária e a armazena no bucket de cenários do Amazon S3.

> **Note**
>
> A convenção de nomenclatura do bucket é ***<stack-name>*** -` dlttestrunnerstoragedltscenariosbucket ***<_[0-9][0-9]..-<[0-9][0-9].._*** com a palavra-chave scenarios no nome do bucket, que você pode localizar navegando até o console do S3 e depois Buckets.

Para executar uma implantação multirregional, você deve implantar o CloudFormation modelo regional, que é armazenado no bucket de cenários do Amazon S3, nas regiões em que você deseja executar o teste. Você pode instalar o modelo regional fazendo o seguinte:

1. No console web da solução, navegue até Gerenciar regiões no menu superior.
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.

Parameter	Padrão	Descrição
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

Parameter	Padrão	Descrição
		sub-rede precisa de uma rota para a Internet para extrair a imagem do contêiner para 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 uma 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 aos contêineres 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 forem implantadas com sucesso, elas aparecerão no console da web. Quando você cria um teste, a nova região será listada no modal Gerenciar regiões. Você pode usar essa

região em um teste selecionando-a na criação do teste. A solução cria um item do DynamoDB para cada região lançada 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. Devido às restrições da API, você pode visualizar os resultados agregados de todas as regiões em um teste multirregional somente ao representá-los graficamente nas métricas da Amazon. CloudWatch Você pode encontrar o código-fonte do gráfico nos resultados do teste quando o teste for concluído.

> **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

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

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 Amazon S3 URL.
   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. Selecione a caixa reconhecendo que o modelo pode 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.

Ao atualizar de versões DLT anteriores à v3.2.6 para a v3.3.x e da v3.3.x para a mais recente, a atualização da pilha falha

1. Faça o download do distributed-load-testing-on-aws.template.

2. Abra o modelo e navegue até Condições: e procure DLTCommon ResourcesAppRegistryCondition
3. Você deverá ver algo semelhante a:

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

4. Altere o segundo valor verdadeiro para falso:

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

5. Use o modelo personalizado para atualizar sua pilha.
6. Essa pilha remove os recursos relacionados ao registro de aplicativos da pilha. Portanto, a atualização deve ser concluída.
7. Execute outra atualização da pilha usando o URL do modelo mais recente para adicionar novamente os recursos do aplicativo de registro de aplicativos à sua pilha.

> **Note**
>
> O AWS Systems Manager Application Manager oferece uma visão em nível de aplicativo dessa solução e de seus recursos para que você possa:
>
> 1. Monitore seus recursos, custos dos recursos implantados em pilhas, contas da AWS e registros associados a essa solução a partir de um local central.
> 2. Visualize os dados operacionais dos recursos dessa 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 ou EC2 instâncias internas, 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 Developer Support, o AWS Business Support ou o AWS Enterprise Support, você pode usar o Support Center para obter assistência especializada com essa solução. As seções a seguir dão instruções.

Criar caso

1. Faça login no Support Center.
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. Em Descrição, descreva o problema em detalhes.
3. Escolha 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.

Resolva agora ou entre em contato conosco

1. Analise as soluções Solve now.
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

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

Usando a interface de linha de comando da AWS

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

```
$ 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
```

Use a solução

Esta seção inclui informações sobre como usar a solução Distributed Load Testing on AWS, incluindo resultados de testes, fluxo de trabalho de agendamento de testes e dados ao vivo.

Resultados do teste

O teste de carga distribuído na AWS utiliza a estrutura de testes de carga para executar testes de aplicativos em grande escala. Quando um teste é concluído, um relatório detalhado é gerado contendo os seguintes resultados.

- 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 geradas pelo teste.
- 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.
- Percentil - O percentil do tempo de resposta para o teste. O tempo máximo de resposta é 100%; o tempo mínimo de resposta é 0%.

> **Note**
>
> Os resultados do teste são exibidos no console. Você pode visualizar os arquivos XML dos resultados brutos do teste na `Results` pasta do bucket do `Scenarios` Amazon S3.

Para obter mais informações sobre os resultados do teste Taurus, consulte Geração de relatórios de teste no Manual do usuário do Taurus.

Fluxo de trabalho de agendamento de testes

Use o console web para agendar um teste de carga. Ao agendar um teste, o seguinte fluxo de trabalho é executado:

- Quando um teste de carga é criado com a opção de agendamento, os parâmetros do cronograma são enviados para a API da solução por meio do Amazon API Gateway.
- Em seguida, a API passa os parâmetros para uma função Lambda que cria uma regra de CloudWatch eventos, que será programada para ser executada na data especificada.
- Se o teste for um teste único, a regra de CloudWatch Eventos será executada na data especificada. A função `api-services` Lambda executa um novo teste por meio do fluxo de trabalho de teste.
- Se o teste for recorrente, a regra de CloudWatch Eventos será ativada na data especificada. A função `api-services` Lambda é executada, o que exclui a regra de CloudWatch eventos atual e cria outra regra que é executada imediatamente quando criada e de forma recorrente a partir de então, com base na frequência de recorrência especificada.

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). 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, os recursos do contêiner podem ser aumentados para permitir um número maior de usuários simultâneos. No entanto, isso resulta em um custo maior. Para obter detalhes, consulte a seção Aumentar os recursos do contêiner deste guia.

> **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

Dados dinâmicos

Opcionalmente, você pode incluir dados ativos ao executar um teste para obter informações em tempo real sobre o que está ocorrendo. O grupo de CloudWatch registros das tarefas do Fargate contém um filtro de assinatura para resultados de testes que incluem a opção de dados ativos. Se a solução encontrar o padrão, ela inicia uma função Lambda que estrutura os dados e os publica em um tópico do AWS IoT Core. O console web se inscreve no tópico, recebe os dados recebidos e representa graficamente os dados agregados em intervalos de um segundo. O console web contém quatro gráficos: tempo médio de resposta, usuários virtuais, sucessos e fracassos.

> **Note**
>
> Os dados são efêmeros e servem apenas para ver o que está acontecendo durante a execução do teste. Depois que o teste é concluído, a solução armazena os dados dos resultados no DynamoDB e no Amazon S3. 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.

Fluxo de trabalho de cancelamento de

Quando você cancela um teste de carga do console web, a solução executa o seguinte fluxo de trabalho de cancelamento de teste.

1. A solicitação de cancelamento é enviada para a `microservices` API.
2. A `microservices` API chama a função `task-canceler` Lambda, que cancela tarefas até que todas as tarefas lançadas atualmente sejam interrompidas.
3. Se a função `task-runner` Lambda continuar sendo executada após a chamada inicial para a função `task-canceler` Lambda, as tarefas continuarão sendo iniciadas. Quando a função `task-runner` Lambda é concluída, o AWS Step Functions continua na `Cancel Test` etapa, que executa a função `task-canceler` Lambda novamente para interromper qualquer tarefa restante.

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 se as atualizações automáticas não forem selecionadas. A equipe de soluções da AWS usa o ECR Enhanced Scanning para detectar vulnerabilidades e exposições comuns (CVEs) na imagem base e nos pacotes instalados. Quando possível, a equipe publicará 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, se estiverem na mesma versão secundária, a tag stable será atualizada automaticamente e uma tag de imagem adicional será criada no formato`<solution-version>_<date-of-fix>`. Se uma versão principal ou secundária for lançada, será necessária uma atualização completa para obter a versão mais recente da imagem, pois a tag estável será incrementada para que sua versão corresponda à versão da solução. Se optar por atualizações automáticas, as alterações na imagem, incluindo as correções de erros menores CVEs e menores, serão aplicadas automaticamente à imagem até a última versão secundária correspondente.

Versões

Os clientes da versão mais recente da solução receberão automaticamente patches de segurança e correções de erros menores e ininterruptas se optarem por atualizações automáticas de imagens. A imagem exibirá automaticamente a imagem mais recente até a versão secundária correspondente mais recente. Para bloquear o contêiner em uma versão específica, a definição da tarefa pode ser editada para especificar que o contêiner use uma versão de imagem específica usando a versão marcada da imagem. As atualizações automáticas também podem ser desativadas selecionando Não às atualizações automáticas CloudFormation ao iniciar a pilha. Isso iniciará a versão da imagem correspondente à versão da solução.

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.

Cenários

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

Tarefas

- GET /tasks
- OPÇÕES/tarefas

Regiões

- GET /regiões
- OPÇÕES/REGIÕES

GET /scenarios

Descrição

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

Resposta

Nome	Descrição
`data`	Uma lista de cenários, incluindo ID, nome, descrição, status e tempo de execução de cada teste

POST /cenários

Descrição

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

Corpo da solicitação

Nome	Descrição
`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`)

Nome	Descrição
`fileType`	O tipo de arquivo de upload (por exemplo,`none`,`script`,`zip`)
`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`)
`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

Nome	Descrição
`testId`	O ID exclusivo do teste
`testName`	O nome do teste
`status`	O status do teste

OPÇÕES/CENÁRIOS

Descrição

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

Resposta

Nome	Descrição
`testId`	O ID exclusivo do teste
`testName`	O nome do teste
`status`	O status do teste

GET /scenarios/ {testId}

Descrição

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

Parâmetro de solicitação

`testId`

- O ID exclusivo do teste

  Tipo: string

  Obrigatório: Sim

Resposta

Nome	Descrição
`testId`	O ID exclusivo do teste

Nome	Descrição
`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`)
`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
`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}

Descrição

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

Parâmetro de solicitação

`testId`

- O ID exclusivo do teste

  Tipo: string

  Obrigatório: Sim

Resposta

Nome	Descrição
`status`	O status do teste

EXCLUIR /scenarios/ {testId}

Descrição

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

Nome	Descrição
`status`	O status do teste

OPÇÕES /cenários/ {testId}

Descrição

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

Resposta

Nome	Descrição
`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`)
`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

Nome	Descrição
`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 /tasks

Descrição

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

Resposta

Nome	Descrição
`tasks`	Uma lista de tarefas IDs para executar testes

OPÇÕES/tarefas

Descrição

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

Resposta

Nome	Descrição
`taskIds`	Uma lista de tarefas IDs para executar testes

GET /regiões

Descrição

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

Resposta

Nome	Descrição
`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

Descrição

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

Resposta

Nome	Descrição
`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 atualmente suportados, aumente os recursos do contêiner. Isso permite que você aumente a memória CPUs e para lidar com o aumento de usuários simultâneos.

Criar uma nova revisão de definição de tarefa

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.
   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 a memória desejada.
   d. Selecione Atualizar.
6. Na página Criar nova revisão, escolha Criar.
7. Depois que a definição da tarefa for criada com êxito, registre o nome da nova definição da tarefa. Esse nome inclui o número da versão, por exemplo: `[replaceable]` <stackName>`- EcsTaskDefinition -*<system-generated-random-Hash>*: [substituível]`<system-generated-versionNumber>`.

Atualizar a tabela do DynamoDB

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 modificou a definição da tarefa. Por exemplo, região-*<region-name>*.
5. Atualize o atributo taskDefinition com a nova definição de tarefa.

Referência

Esta seção inclui informações sobre um recurso opcional para coletar métricas exclusivas para essa solução, indicadores para recursos relacionados e uma lista dos criadores que contribuíram para essa solução.

Coleta de dados anônima

Essa solução inclui uma opção para enviar métricas operacionais anônimas para a AWS. Usamos esses dados para entender melhor como os clientes usam essa solução e os serviços e produtos relacionados. Quando invocadas, as seguintes informações são coletadas e enviadas para a AWS:

- ID da solução — O identificador da solução da AWS
- ID exclusivo (UUID) - identificador exclusivo gerado aleatoriamente para cada implantação de solução
- Timestamp - Timestamp da coleta de dados
- Tipo de teste - O tipo de teste executado
- Tipo de arquivo - O tipo de arquivo que é carregado
- Contagem de tarefas - A contagem de tarefas para cada teste enviado por meio da API da solução
- Duração da tarefa - O tempo total de execução de todas as tarefas necessárias para executar um teste
- Resultado do teste - O resultado do teste que foi executado

A AWS é proprietária dos dados coletados por meio dessa pesquisa. A coleta de dados está sujeita à Política de Privacidade da AWS. Para optar por não usar esse recurso, conclua as etapas a seguir antes de lançar o CloudFormation modelo da AWS.

1. Faça o download do CloudFormation modelo da AWS em seu disco rígido local.
2. Abra o CloudFormation modelo da AWS com um editor de texto.
3. Modifique a seção CloudFormation de mapeamento de modelos da AWS a partir de:

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

para:

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

4. Faça login no CloudFormation console da AWS.
5. Selecione Criar pilha.
6. Na página Criar pilha, seção Especificar modelo, selecione Carregar um arquivo de modelo.
7. Em Carregar um arquivo de modelo, escolha Escolher arquivo e selecione o modelo editado em sua unidade local.
8. Escolha Avançar e siga as etapas em Iniciar a pilha na seção Implantar a solução deste guia.

Colaboradores

- Tom Nightingale
- Fernando Dingler
- Beomseok Lee
- George Lenz
- Erin McGill
- Dimitri López
- Kamyar Ziabari
- Bassem Wanis
- Garvit Singh
- Nikhil Reddy
- Simon Kroll

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.