AWS Distributed Load Testing solution

There are several compelling reasons why using AWS's distributed load testing solution can be advantageous:

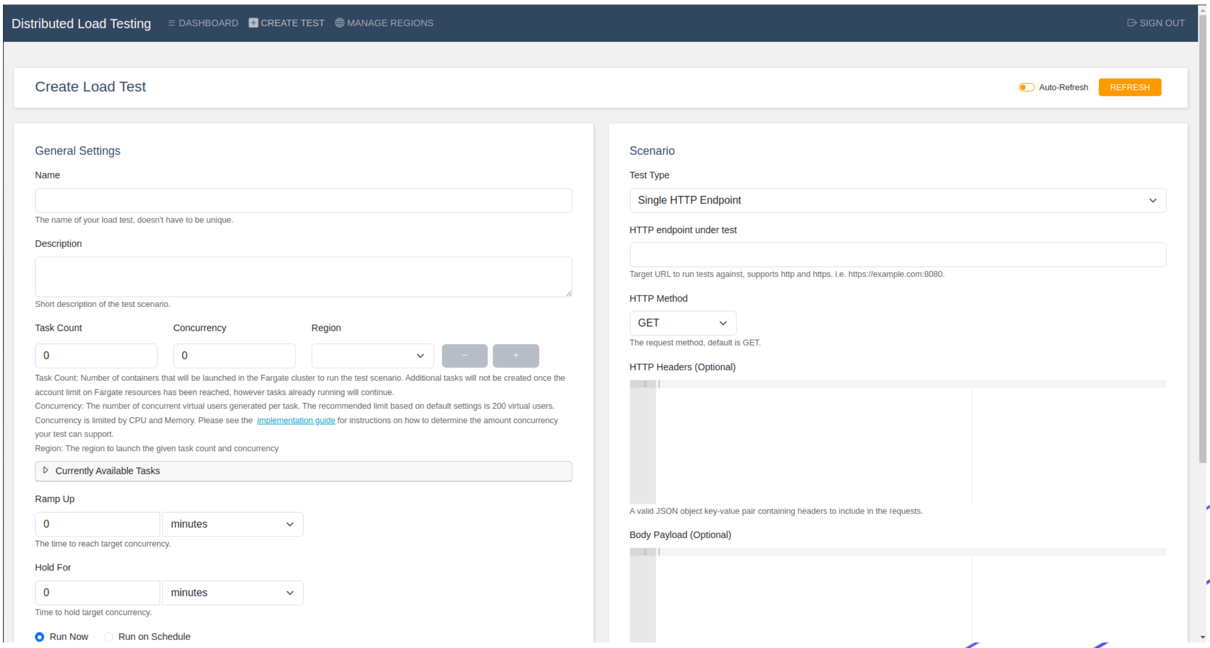
1. **Scalability:** AWS's distributed load testing solution allows you to simulate high loads and traffic on your applications or websites. It can automatically scale up or down based on demand, ensuring that you can test your system's performance under various scenarios effectively.
2. **Cost-effective:** With AWS's pay-as-you-go pricing model, you only pay for the resources you consume during load testing. This makes it a cost-effective option compared to maintaining and managing an on-premises load testing infrastructure.
3. **Easy setup and configuration:** AWS provides pre-configured templates and managed services that simplify the setup and configuration process for load testing. You can quickly launch the necessary resources and tools through AWS Management Console, AWS CLI, or SDKs.
4. **Global reach:** AWS has data centers located across the globe, allowing you to conduct load tests from various geographic locations. This helps you assess your application's performance and responsiveness for users in different regions.
5. **Integration with AWS services:** AWS's load testing solution seamlessly integrates with other AWS services like CloudWatch, which allows you to monitor the performance of your application during load tests, analyze metrics, and identify bottlenecks easily.
6. **Security and compliance:** AWS provides a highly secure and compliant infrastructure. Load testing can involve sensitive data, and AWS ensures that your tests adhere to industry-specific compliance standards, giving you peace of mind.
7. **Flexibility and customization:** AWS allows you to customize load testing scenarios according to your specific requirements. You can adjust the number of virtual users, test duration, and other parameters to suit your needs accurately.
8. **Real-time analytics and reporting:** AWS's distributed load testing solution provides real-time insights into your application's performance. This enables you to identify performance issues promptly and optimize your system for better user experience.

9. **Managed services:** AWS handles the underlying infrastructure and maintenance tasks, allowing you to focus solely on load testing and analyzing results. This reduces operational overhead and frees up resources for other critical tasks.
10. **Experimentation and testing before deployment:** Using AWS's distributed load testing solution, you can assess your application's performance before deploying it to a production environment. This helps you proactively address potential issues and ensures a smoother launch.

You can automate AWS Distributed Load Testing solution deployment by running ready-to-go CloudFormation template.

1. Download the file and change the file type to .yml.
2. Create new CloudFormation stack with given template.
3. Set the stack parameters – You can create resources in existing VPC and subnets or create a new one.
4. After the stack creation go to CloudFront distribution URL and Sign in. You should receive the password by email. If You can't get the password for Your user You can reset it with this AWS CLI command:

   aws cognito-idp admin-set-user-password --user-pool-id <your-user-pool-id> --username <username> --password <password> --permanent
5. In the Distributed Load Testing console You can create test scenarios accordingly to Your needs.



Create Load Test

General Settings

Name

The name of your load test, doesn't have to be unique.

Description

Short description of the test scenario.

Task Count: 0
Concurrency: 0
Region:

Task Count: Number of containers that will be launched in the Fargate cluster to run the test scenario. Additional tasks will not be created once the account limit on Fargate resources has been reached, however tasks already running will continue.
Concurrency: The number of concurrent virtual users generated per task. The recommended limit based on default settings is 200 virtual users. Concurrency is limited by CPU and Memory. Please see the implementation guide for instructions on how to determine the amount concurrency your test can support.
Region: The region to launch the given task count and concurrency

Ramp Up: 0 minutes

The time to reach target concurrency.

Hold For: 0 minutes

Time to hold target concurrency.

● Run Now ○ Run on Schedule

Scenario

Test Type: Single HTTP Endpoint

HTTP endpoint under test

Target URL to run tests against, supports http and https. i.e. https://example.com:8080.

HTTP Method: GET

The request method, default is GET.

HTTP Headers (Optional)

A valid JSON object key-value pair containing headers to include in the requests.

Body Payload (Optional)