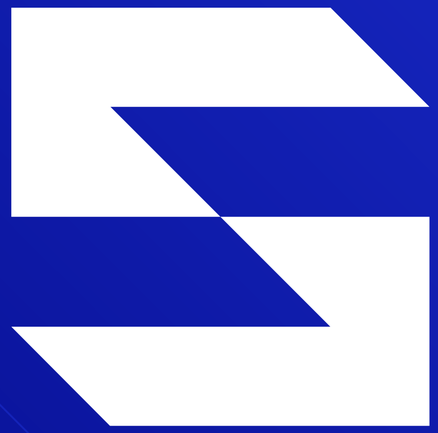


Serverless Operations

人気番組の新作配信を安定起動させた、サーバーレスな AWS 分散負荷試験ソリューション「Distributed Load Testing」を使った負荷試験の仕組み

Serverless Operations

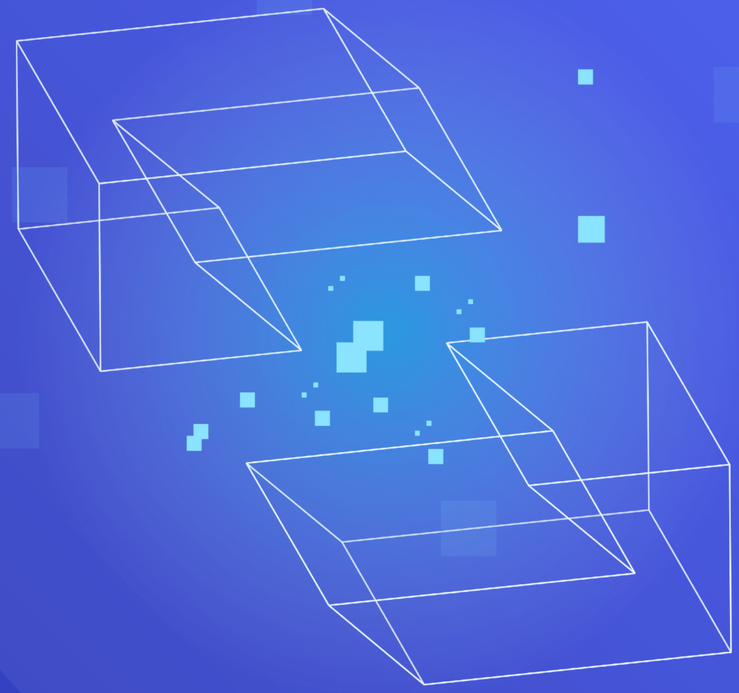
金　仙優（Sonu Kim）

会社紹介

サーバーレスで
クラウドの価値を最大限に

Serverless Operations はこれまでグローバルの第一線で培ってきたクラウド技術、アマゾンウェブサービス（AWS）の豊富な実績と知見を活かし、お客さまのサーバーレスに関するさまざまな課題を解決します。

serverless.co.jp



Maximize the cloud value with serverless

アジェンダ

- 負荷試験とは何か
- サーバーレスなワークロードに、なぜ負荷試験を行うのか
- 負荷試験ツールの紹介と活用
- 注意事項
- 事例紹介

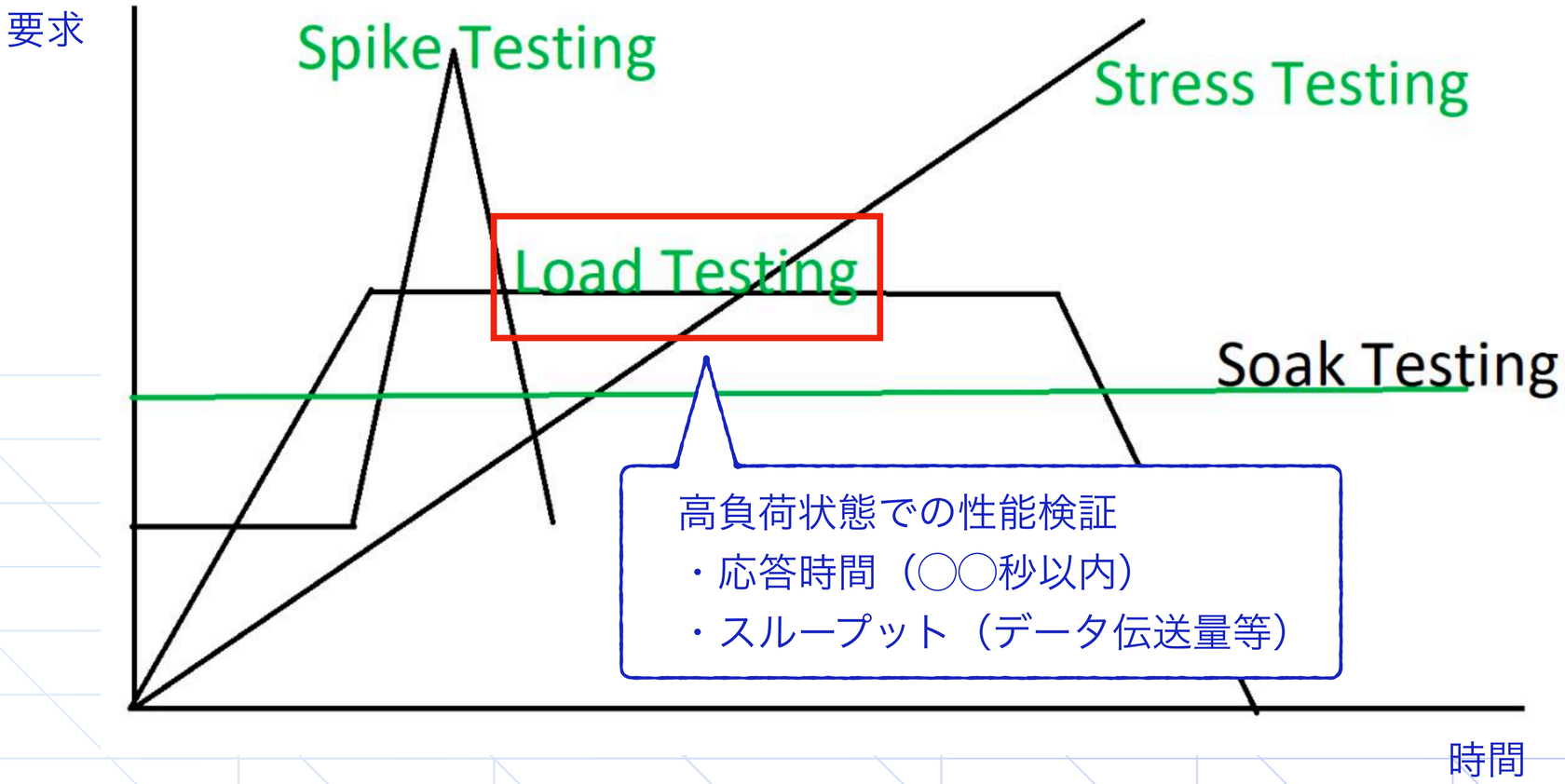
負荷試験とは？

- 特定の条件化でシステムやワークロードに擬似的な負荷を与えて処理能力を測定し、
  実運用に耐えられるかどうかをリリース前に検証するテスト

- 広範囲で負荷試験と言っているが、目的と内容に応じて複数のアプローチがある

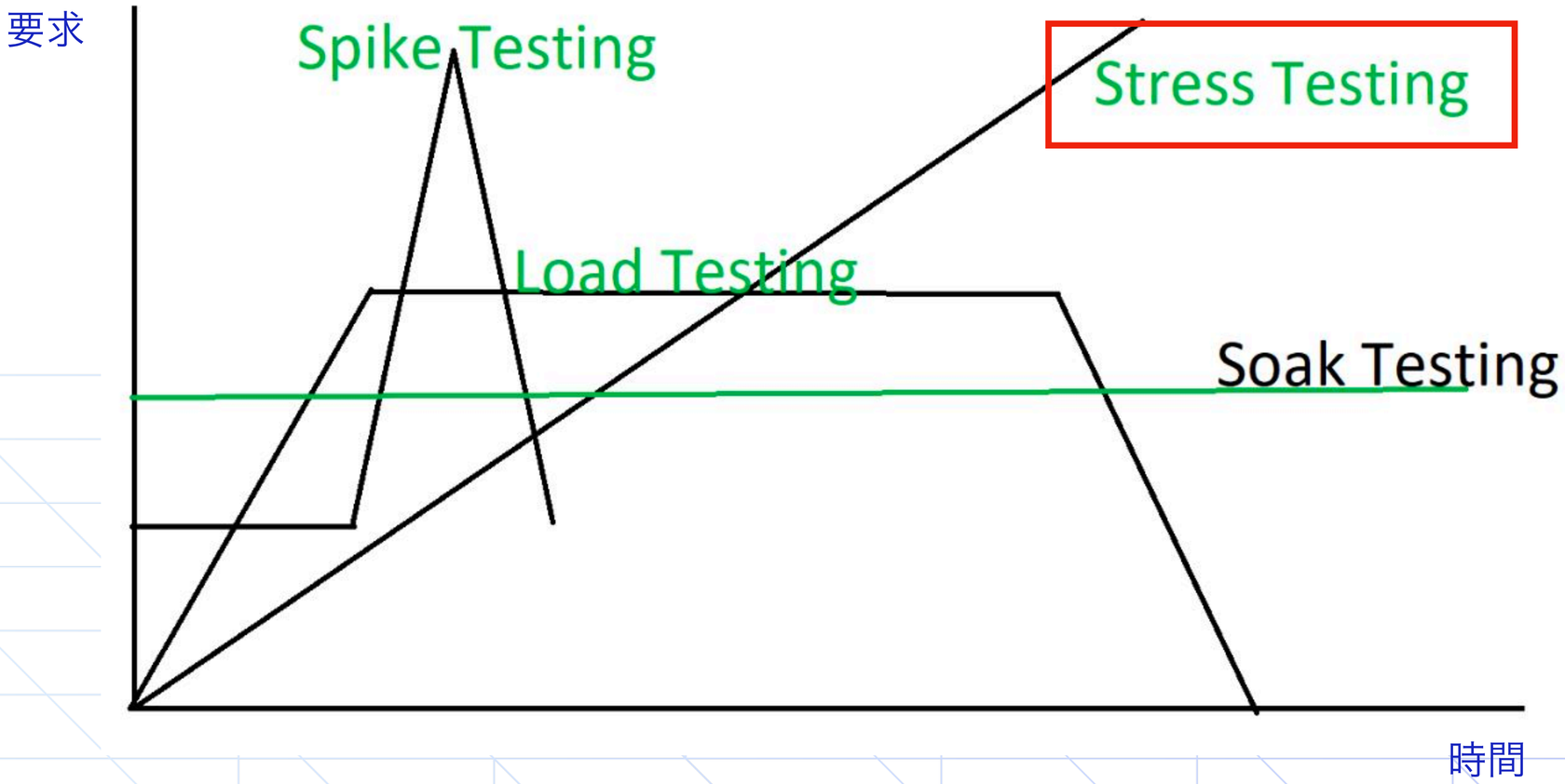
  - 負荷（Load）テスト
  - ストレス（Stress）テスト、スパイク（Spike）テスト
  - 耐久（Endurance/Soak）テスト

負荷試験とは？


要求
Spike Testing
Stress Testing
Load Testing
Soak Testing
高負荷状態での性能検証
・応答時間（〇〇秒以内）
・スループット（データ伝送量等）
時間

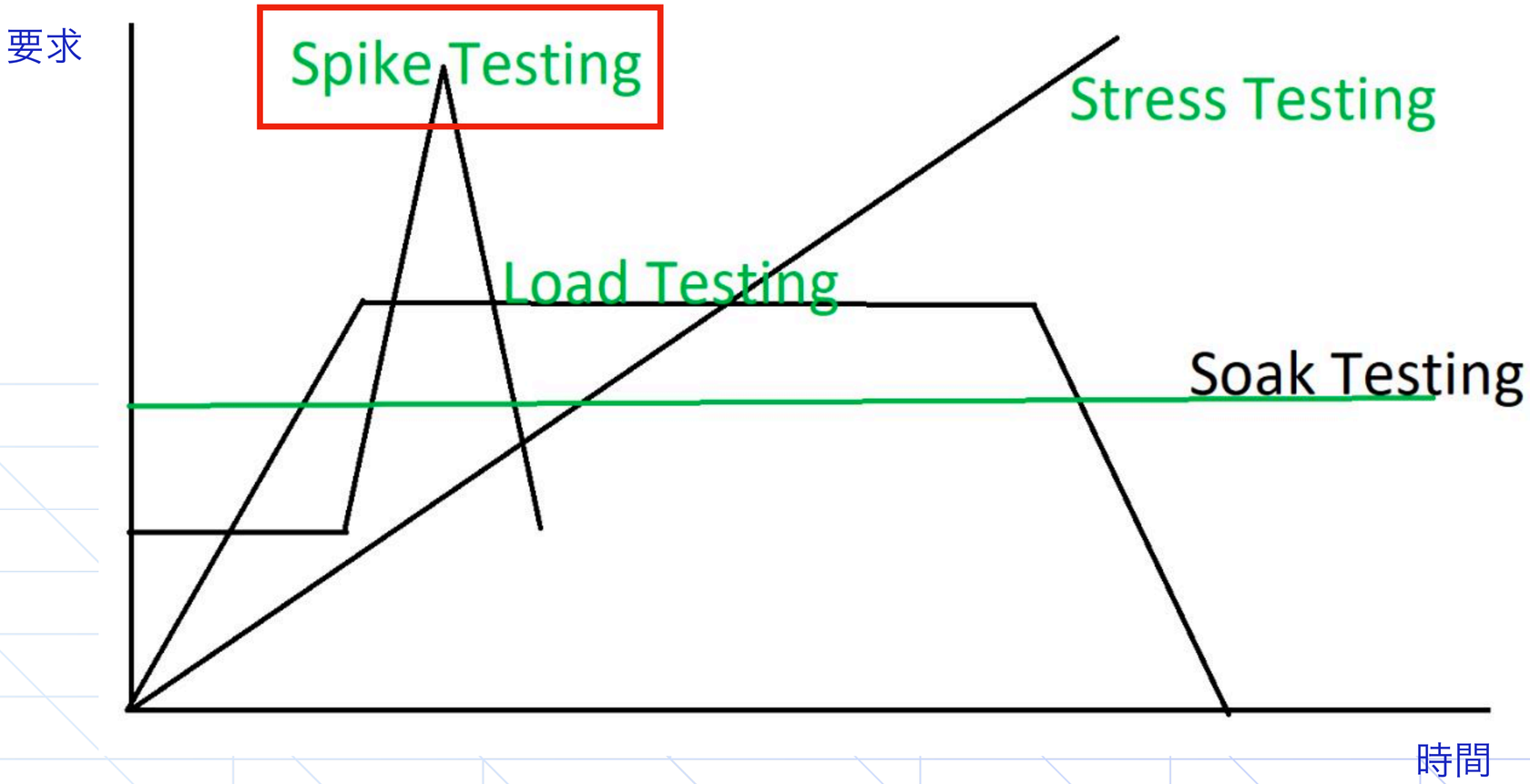
負荷試験とは？

システムが処理できる「限界」、
想定を超えた負荷が来た場合の検証

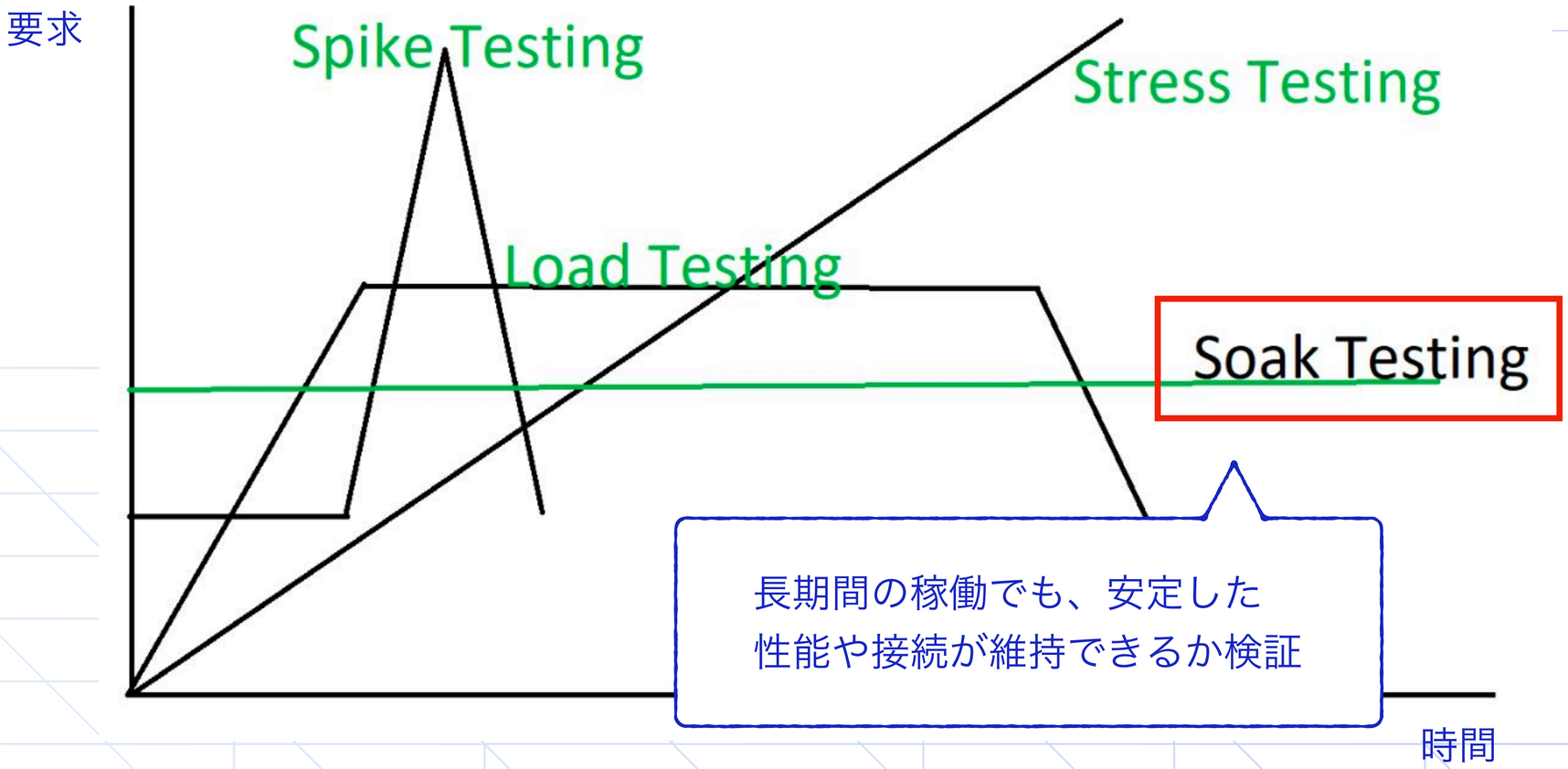


負荷試験とは？

要求が「短時間」で、
「急激に」増加する場合の検証



負荷試験とは？


要求
Spike Testing
Stress Testing
Load Testing
Soak Testing
長期間の稼働でも、安定した
性能や接続が維持できるか検証
時間

なぜ負荷試験を行うのか

- 一般的に負荷試験を行う理由
  - システム・アーキテクチャの限界
  - 本番を迎えるための予行練習
  - トラフィックが大きなワークロードの検証
- サーバーレスで負荷試験を行う必要があるだろうか？

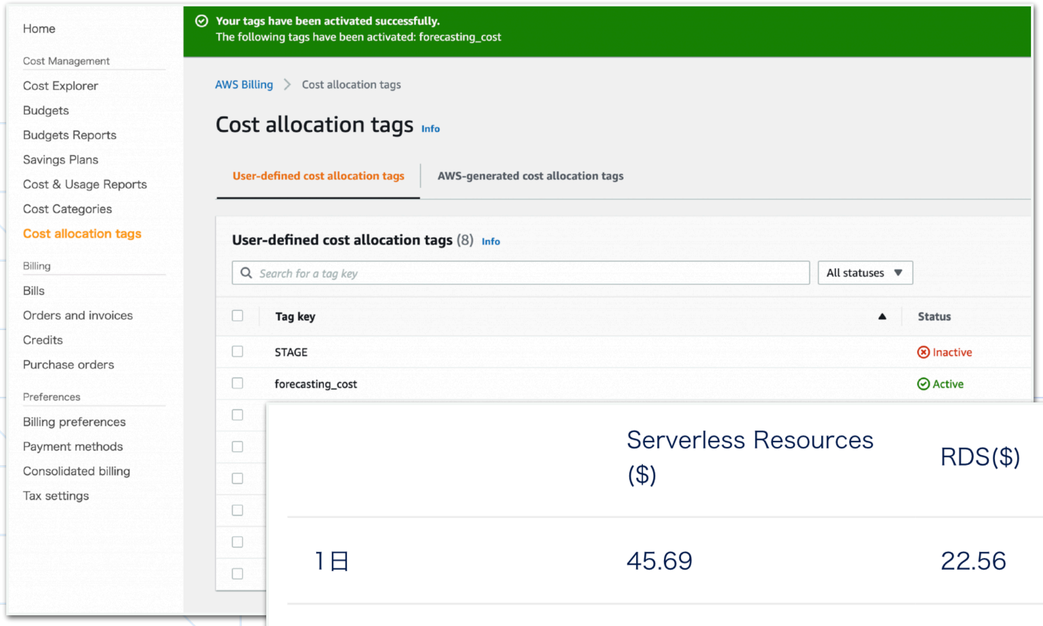
サーバーレスで負荷試験を行う理由１：コスト

- サーバーレスアーキテクチャの料金構成は実行単位での従量課金制
- 計算が複雑で、コストの予測や見積が難しい

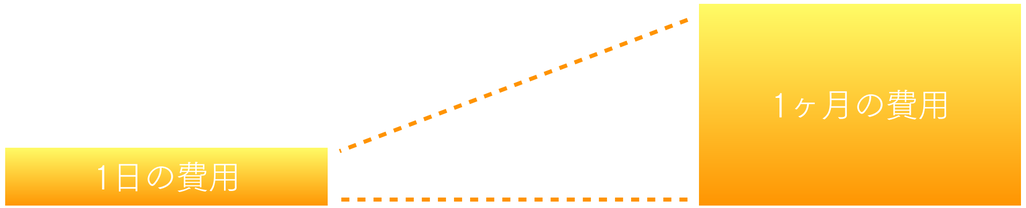
AWS IoT Core	No group applied	Asia Pacific (Tokyo)	0.00 USD	10.81 USD
Status: - **Description:** IoT core backend **Config summary**: Number of devices (MQTT) (10000), Average size of each message (5 KB), Average size of each record (1 KB), Average size of each record (1 KB), Average number of actions executed per rule (1), Average size of each message (5 KB), Number of messages for a device (9), Number of Shadow operations for a device (10), Number of rules triggered (10)				
AWS AppSync	No group applied	Asia Pacific (Tokyo)	0.00 USD	5.95 USD
Status: - **Description:** Frontend **Config summary**: Number of API Requests (1 million per month) Number Of Connected Clients (100), Average Active Duration per Connected Client (480 per day), Number Of Messages Published (100 per day), Number Of Messages Delivered (100 per day)				
Amazon Simple Storage Service (S3)	No group applied	Asia Pacific (Tokyo)	0.00 USD	0.73 USD
Status: - **Description**: **Config summary**: S3 Standard storage (1 GB per month), PUT, COPY, POST, LIST requests to S3 Standard (100000), GET, SELECT, and all other requests from S3 Standard (640000), Data returned by S3 Select (2 GB per month) DT Inbound: Not selected (0 TB per month), DT Outbound: Not selected (0 TB per month)				

コスト配分タグを設定して負荷試験を実施、コストを発生させる

- 1日程度経過すれば、AWS Cost Explorer からコスト配分タグでフィルタリングできる
- かけた負荷と同じ量のトラフィックが常時発生している場合のコストが正確にわかる



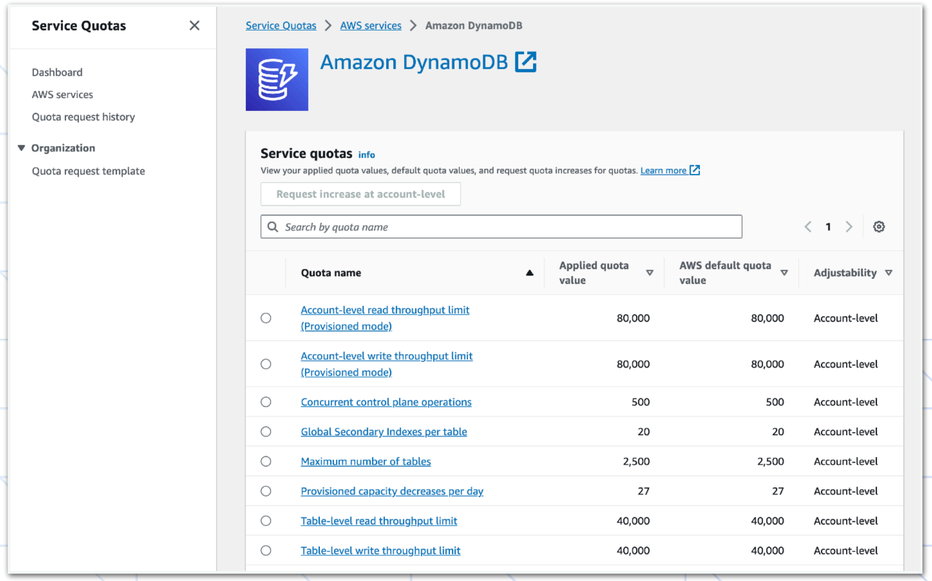
	Serverless Resources ($)	RDS($)	Summary($)
1日	45.69	22.56	68.25
1ヶ月	1,370	676.8	2,046.8



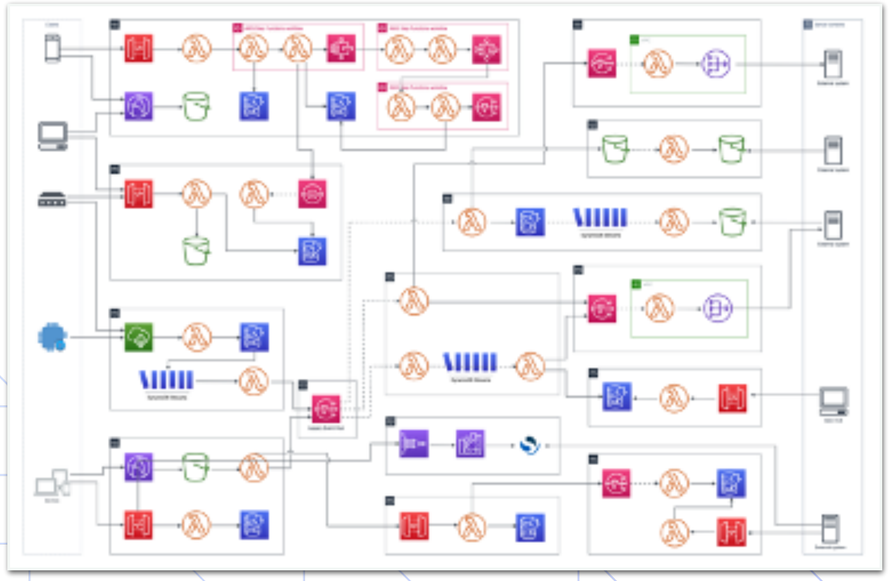
「pay-as-you-go」
使用量に応じて変動する特性を
利用して予測する

サーバーレスで負荷試験を行う理由２：上限緩和申請を行うときの検証

- AWSを含め、クラウドのサービスには利用量の上限や制限が設けられている
- 上限緩和申請を行って増やすことができるとしたら、何のサービスの、どの制限を、どこまで伸ばせば良いだろうか？



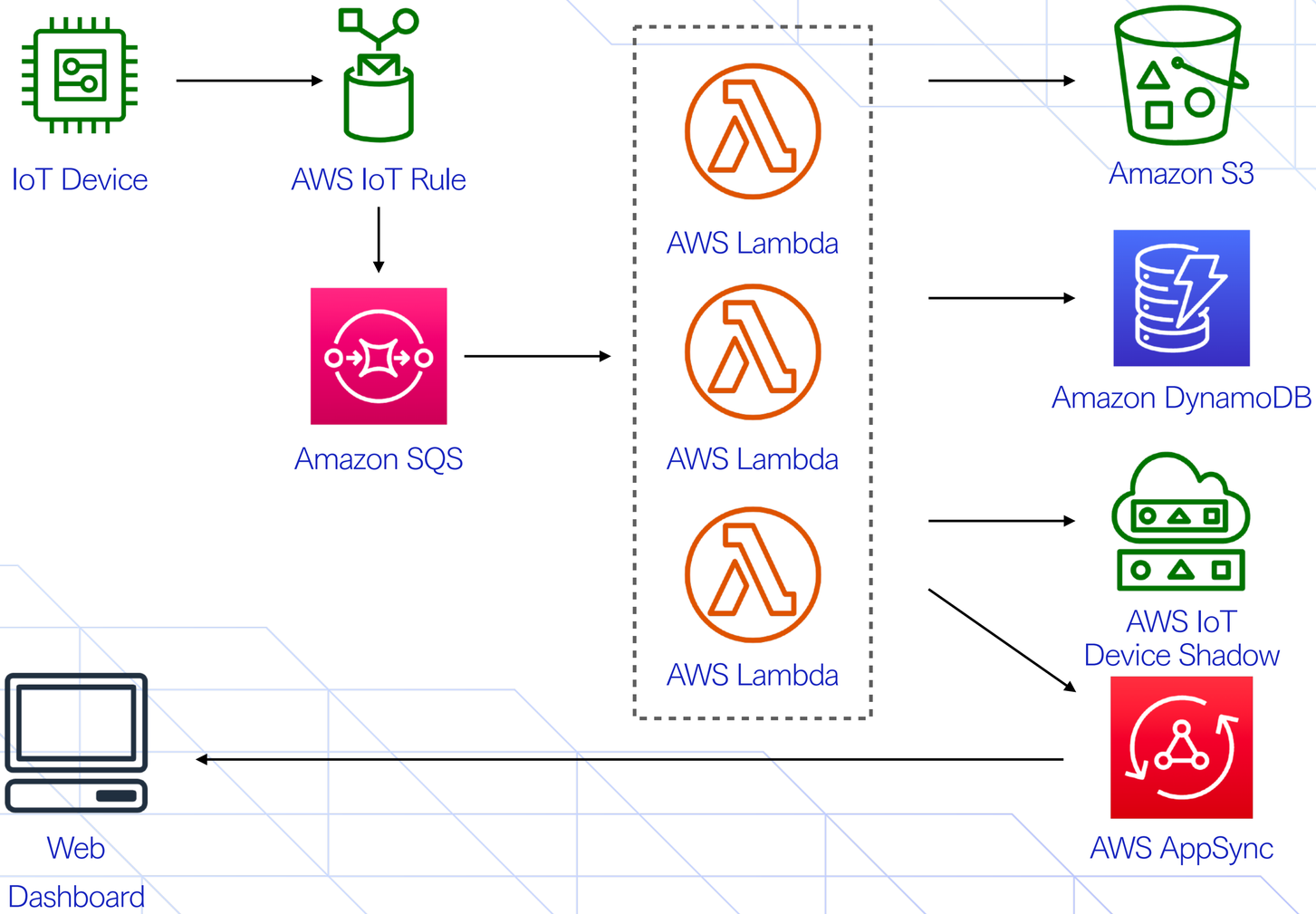
Amazon DynamoDBのサービスクオータの例



複雑化した構成の例

ボトルネックや気づきにくい問題点を早期発見

- 構成要素が増えて複雑化するほど、弱点やボトルネックに気づきにくい
- 個別サービスの制限事項をすべて事前に把握して対応することは、現実的に難しい

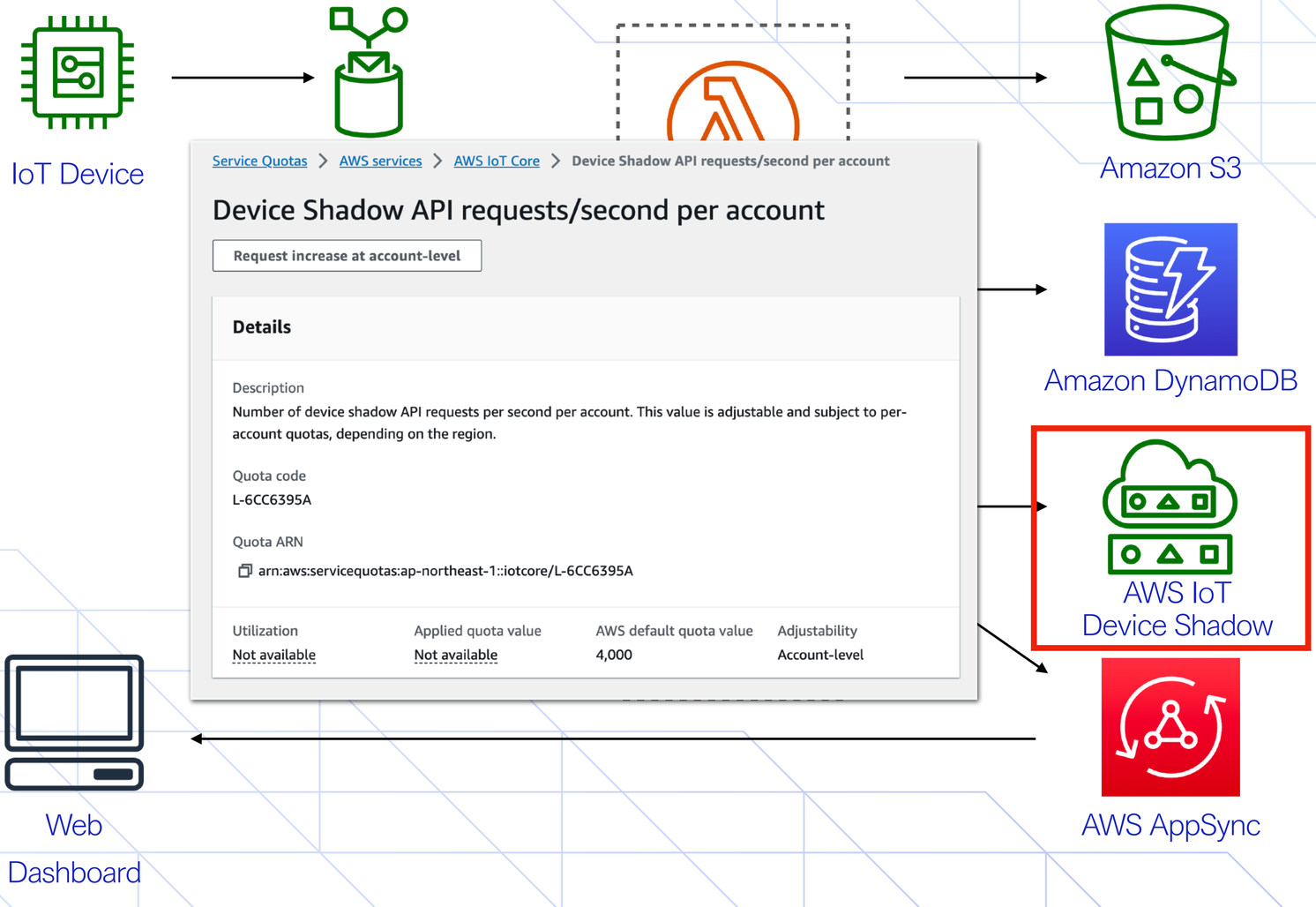


ボトルネックや気づきにくい問題点を早期発見

- 負荷をかけてみて、制限に引っかかる部分をリリース前に気づいて対応できる
- サービス上限緩和申請を行うときの説明と、上限緩和を説明する理由を具体的なデータに基づいて説明することができる

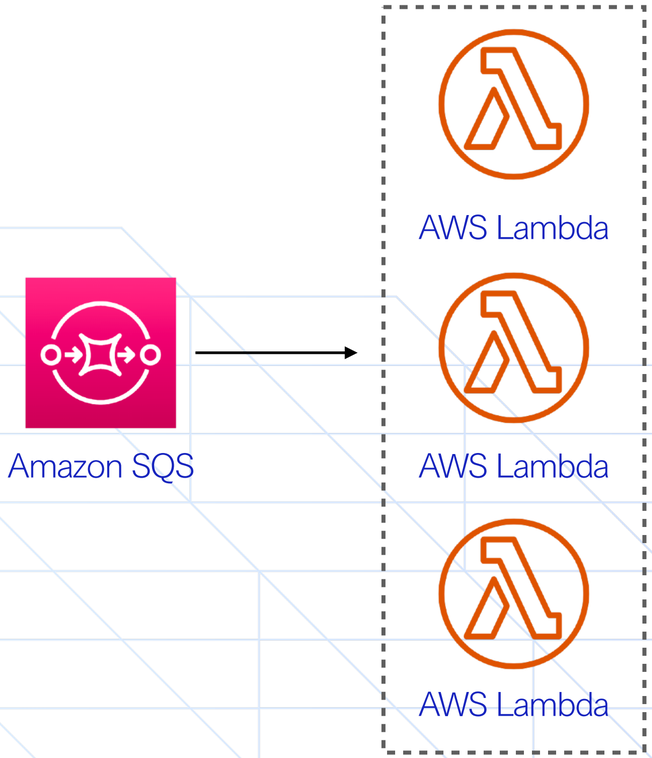
ボトルネックや気づきにくい問題点を早期発見

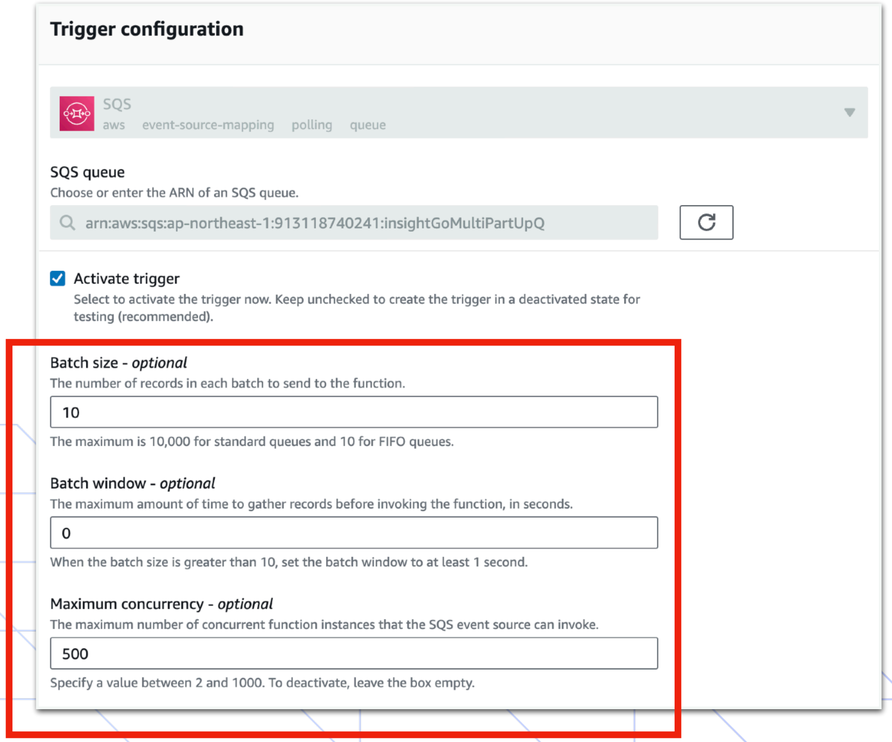
- 負荷をかけてみて、制限に引っかかる部分をリリース前に気づいて対応できる
- サービス上限緩和申請を行うときの説明と、上限緩和を説明する理由を具体的なデータに基づいて説明することができる



サーバーレスで負荷試験を行う理由３：最適化とチューニング

- 同じ構成でも、たとえば AWS Lambda の起動に関しても細かく設定が行える
- 最適なパフォーマンスを実現するために、負荷試験結果を見てチューニングを行うことがおすすめ





負荷試験の結果から最適化を行うための手段を考える

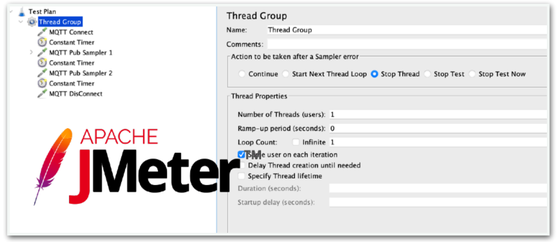
- コールドスタートの時間が関係することもあるので、複数回実施して確認する
- リリース直後にスパイク時のパフォーマンスを安定させたい場合
  Provisioned Concurrency の設定も検討する

実施結果

- [A]: 最初のファイルチャンク～ダウンロードURL配信までの経過時間
- [B]: 最初のファイルチャンク～ダウンロードURL配信までの経過時間
- [C]: 最初のファイルチャンク～ダウンロードURL配信までの経過時間

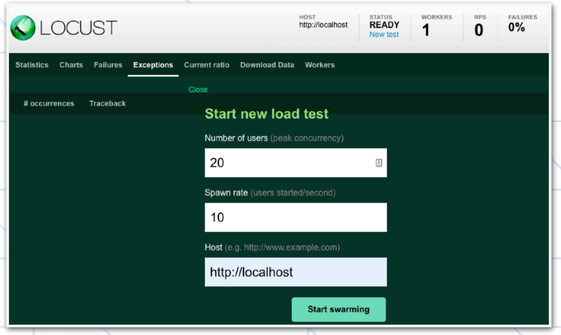
実行区分	**[A] 平均**	[A] 最短	[A] 最長	[B] 平均	[B] 最短	[B] 最長	**[C] 平均**	[C] 最短	[C] 最長
1回目 (Cold Start)	**6847 ms**	873 ms	9195 ms	7624 ms	1587 ms	9404 ms	**7690 ms**	1643 ms	9469 ms
2回目 (Warm Start)	**2955 ms**	580 ms	4706 ms	3390 ms	1056 ms	4718 ms	**3453 ms**	1116 ms	4778 ms

負荷試験ツールの選定



Apache JMeter

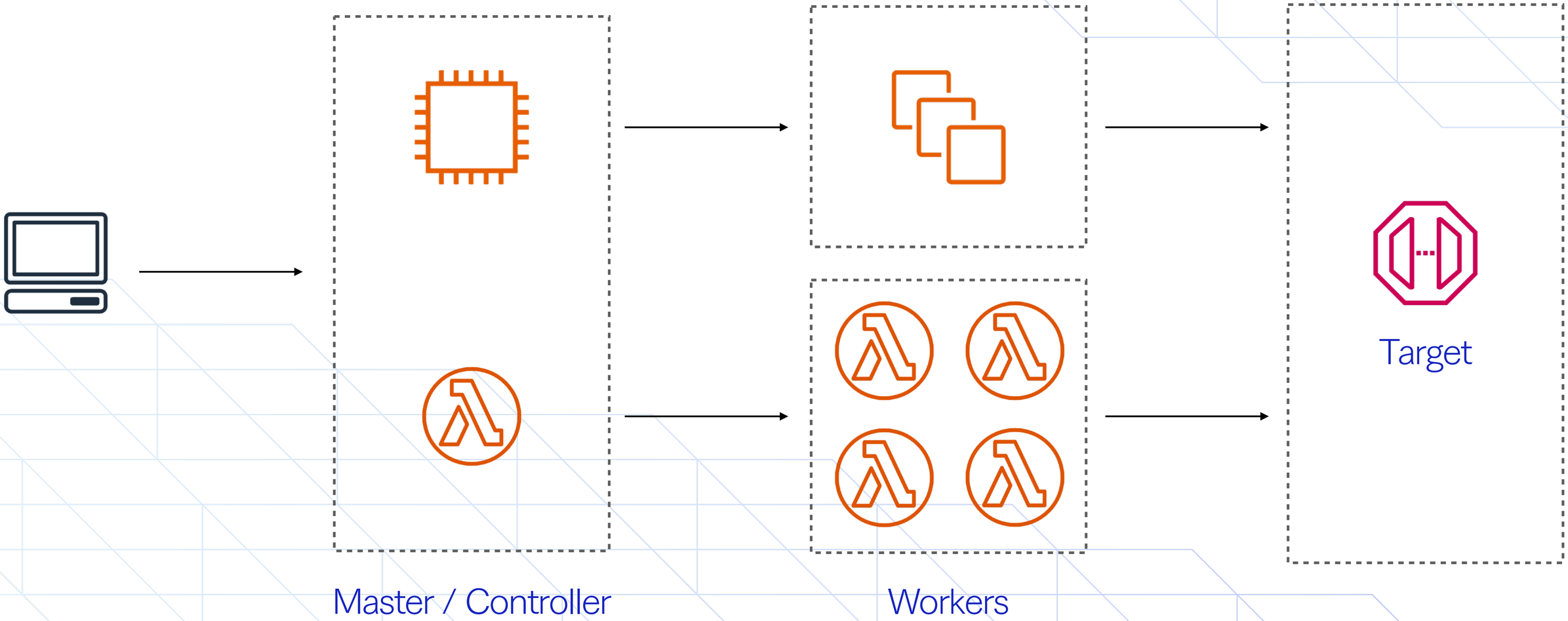
Serverless-artillery



LOCUST

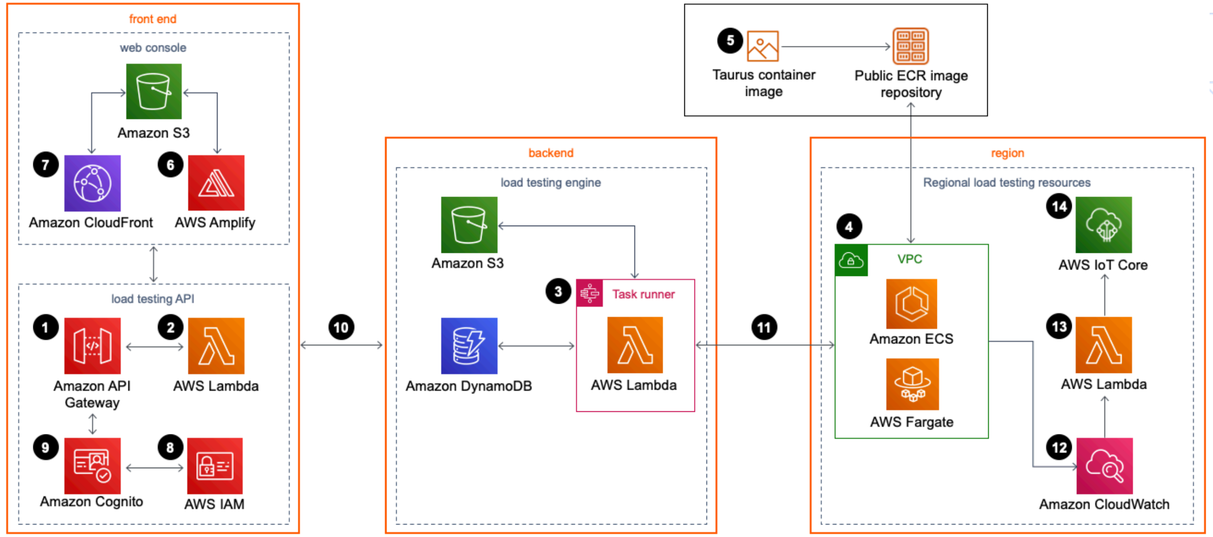
Distributed Load Testing on AWS

負荷試験ツールの基本的な構成

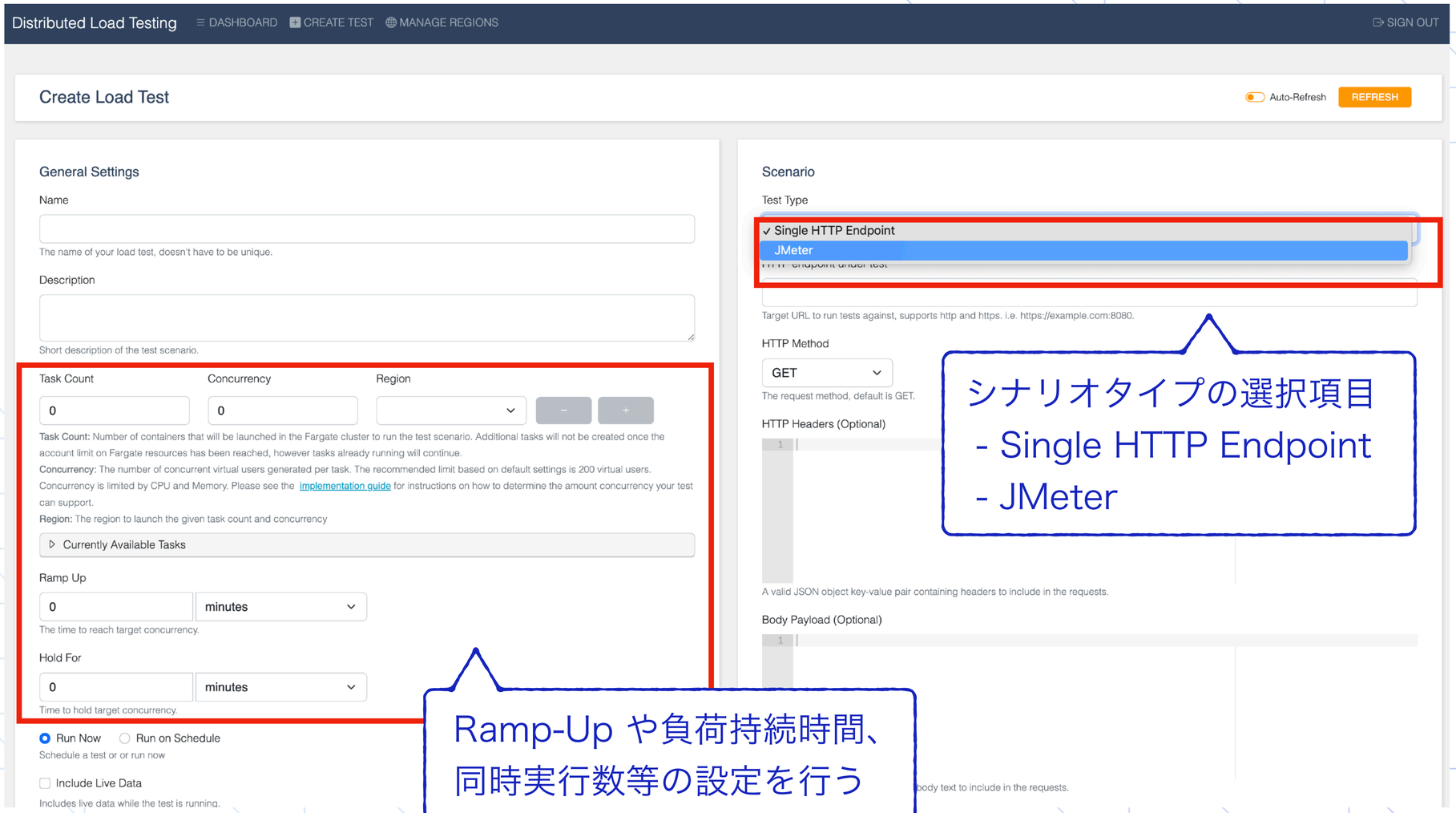


Distributed Load Testing on AWS

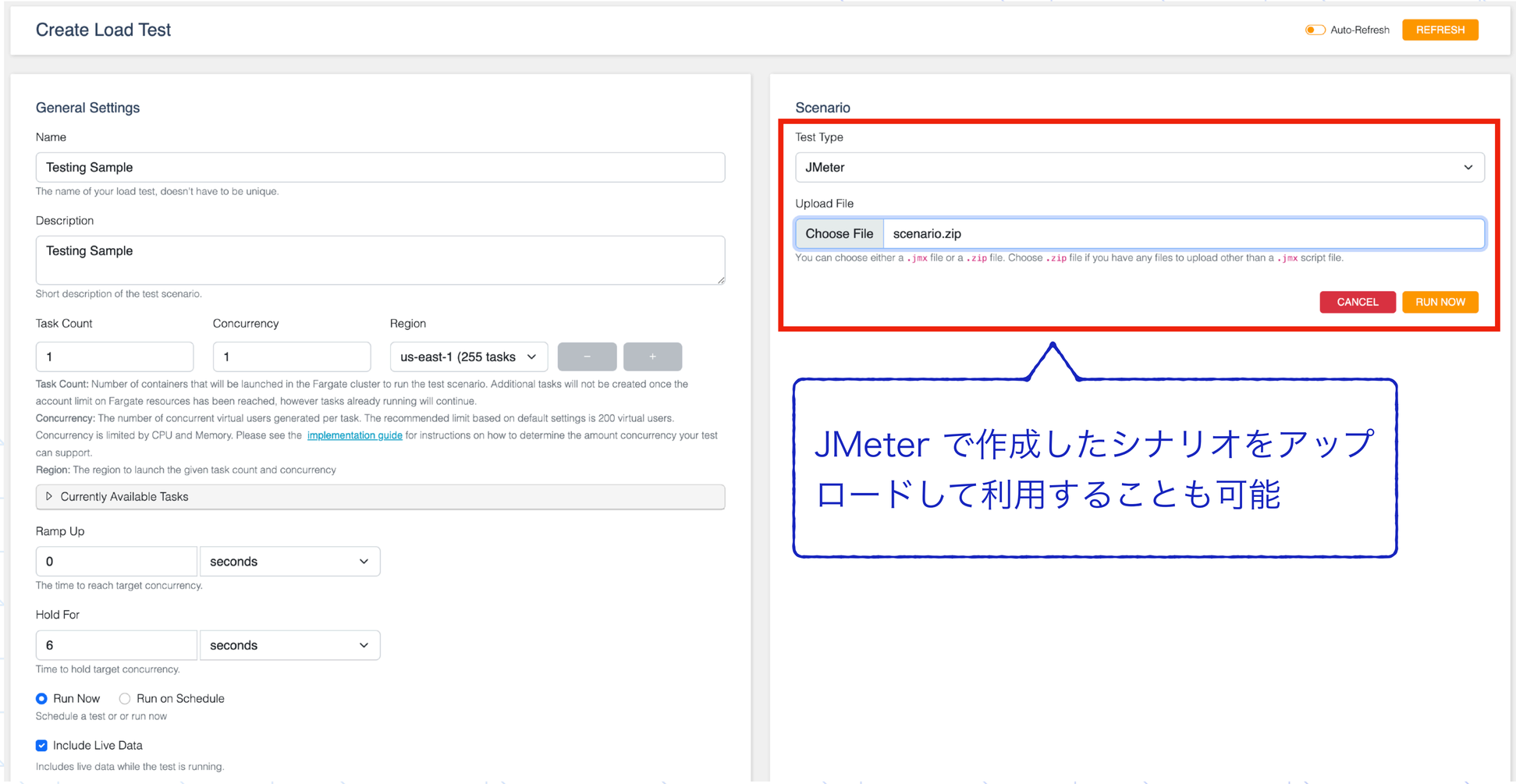
- サーバーレス構成で、AWS公式サイトからAWS　CloudFormationテンプレートを展開してすぐ構築できる
- 操作のための Web UI とAWS Fargateのテストワーカーで構成されている
- 公式の説明が手厚く、JMeterで作成したシナリオを流用できるので利用しやすい



Distributed Load Testing on AWS



Distributed Load Testing on AWS



JMeter で作成したシナリオをアップロードして利用することも可能

Distributed Load Testing on AWS

テスト結果のサマリーとRPSなどが
Web UIからすぐ確認できる

Load Test Details

START EDIT DELETE

ID	Yrxxb6Dvuo
NAME	Testing Sample
DESCRIPTION	Testing Sample
FULL TEST RESULTS	Link to results files in S3
	You must be logged into your AWS account to access the results in S3
ZIP	Download

STATUS	COMPLETE
STARTED AT	2023-04-18 08:04:43
ENDED AT	2023-04-18 08:06:55
TASK COUNT	us-east-1 : 1 (1 completed)
CONCURRENCY	us-east-1 : 1
RAMP UP	0s
HOLD FOR	6s

Test Results Info

us-east-1

SUMMARY
MQTT CONNECT
MQTT PUB SAMPLER 1
MQTT PUB SAMPLER 2
MQTT DISCONNECT

Avg Response Time	Avg Latency	Avg Connection Time	Avg Bandw
0.20675s	0.00275s	0.00000s	1.55 Bp
Total Count: 4	Success Count: 4	Error Count: 0	Requests P

Percentile Response Time

100%: 0.785s
99.9%: 0.785s

CloudWatchMetrics-us-east-1

Seconds
1.00

Test Results Info

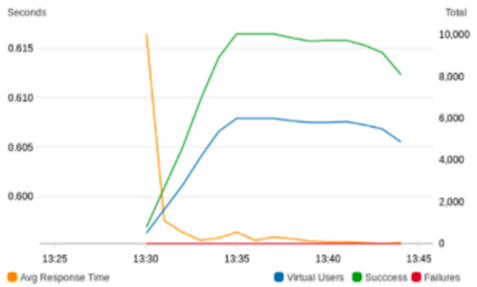
SUMMARY

Avg Response Time	Avg Latency	Avg Connection Time	Avg Bandwidth
0.59579s	0.59578s	0.44645s	842.58 Bps
Total Count: 125854	Success Count: 125854	Error Count: 0	Requests Per Second: 142.21

Percentile Response Time

100%: 1.175s
99.9%: 0.650s
99%: 0.611s
95%: 0.603s
90%: 0.601s
50%: 0.595s
0%: 0.586s

CloudWatch Metrics



Results History

RunTime	AvgRt	AvgLt	AvgCt	100%	99.9%	99.0%	95%	90%	50%	0%
2021-07-12 13:45:45	0.59579	0.59578	0.44645	1.175	0.650	0.611	0.603	0.601	0.595	0.586

注意事項

- 一般的な検証では、いきなり大量の負荷をかけない
  （100, 1000, 10000 というように少しずつ増やしながら検証）

- ログやトレーシング等は、負荷試験により料金が増える可能性がある点
  特にログに関しては、'ERROR' レベルにするなどの工夫を行う

- 連携システムに影響を与えるような場合は、事前にモック化するなどの対応が必要

注意事項

- AWSの例ではネットワーク観点などで事前申請が必要なケースがあるため、
  事前にドキュメントを読んで内容を確認しておく

製品 / コンピューティング / Amazon EC2

Amazon EC2 Testing Policy

ページの内容

ネットワーク負荷テスト

ネットワーク負荷テスト

このポリシーは、Amazon EC2 インスタンスから、別の Amazon EC2 インスタンス、AWS のプロパティ/サービス、外部エンドポイントなどの他のロケーションに向けて、ボリュームの大きなネットワークテストの実行を計画中のお客様に関係します。これらのテストは、ストレステスト、ロードテスト、ゲームデイテストと呼ばれる場合もあります。このポリシー では、あるテストにおいて、正規のトラフィックまたはテストトラフィックが、意図した特定のターゲットアプリケーションに大量に送信される場合に、これを "ネットワーク負荷テスト" と見なします。エンドポイントやインフラストラクチャには、そのようなトラフィックを処理できることが期待されています。このポリシーが通常の本番トラフィックに適用されることはありません。ネットワーク負荷テストは、通常の本番使用とは異なり、多くの場合特定のエンドポイントをターゲットとし、ソースやターゲットの集中したものを含めさまざまなトラフィックパターンを持ち、通常よりもボリュームの大きなトラフィックが連続し、予想される制限を偶発的に超えてしまう場合もあります。このような相違点があるため、ネットワーク負荷テストの間には、外部エンドポイント、他のお客様、AWS のサービスに対して意図しない潜在的リスクが発生する可能性があります。

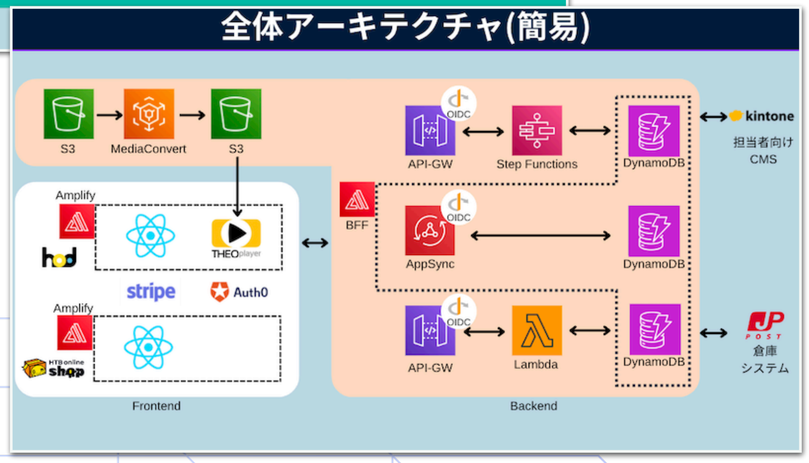
https://aws.amazon.com/jp/ec2/testing/?nc1=h_ls

Amazon EC2 testing policy

You do not need approval from AWS to run load tests using this solution as long as your network traffic stays below 1 Gbps. If your test will generate more than 1 Gbps, contact AWS. For more information, refer to the Amazon EC2 Testing Policy.

事例：動画配信サービスでの人気番組の新作配信

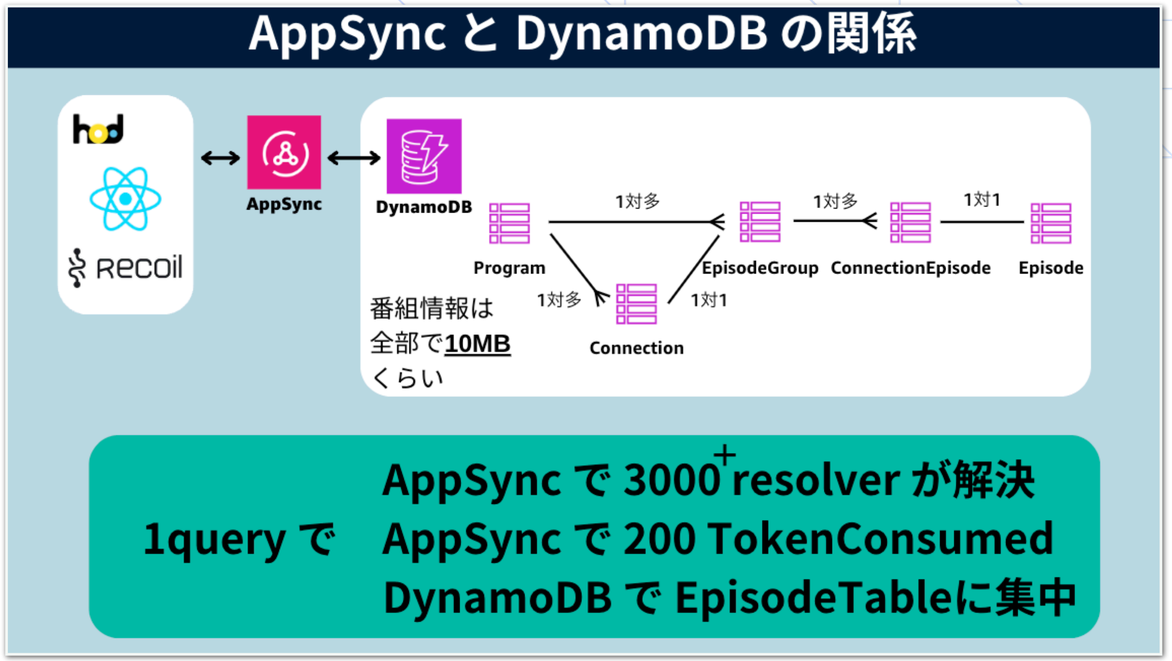
- 外注なく内製開発でフルサーバーレス構成のサービスを展開、運用中
- 人気番組の新作配信が開始する時刻が予定され、告知されている
- 公開時間に合わせて 5,000 TPS までの負荷に耐えることが要件



直面していた技術課題

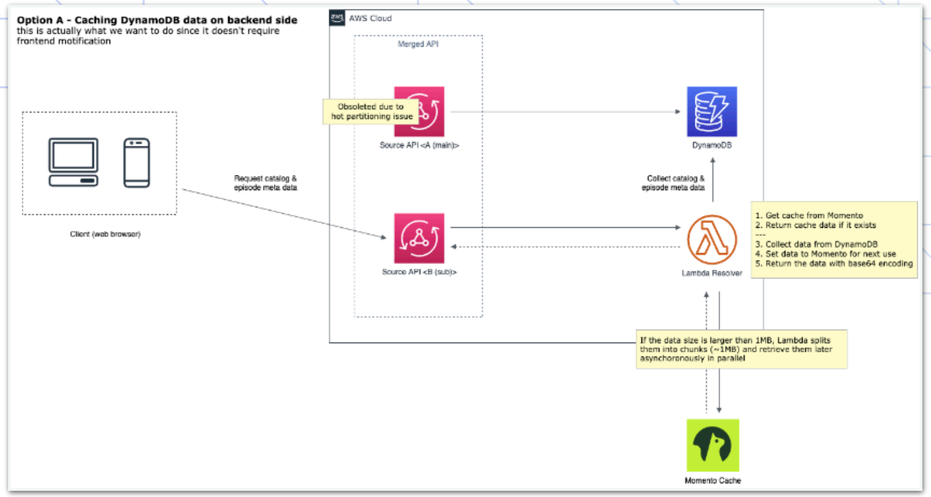
- 複雑かつサイズの大きい番組情報データを読み込むのに、AWS AppSyncとAmazon DynamoDBに想定以上の負担がかかっていた状態

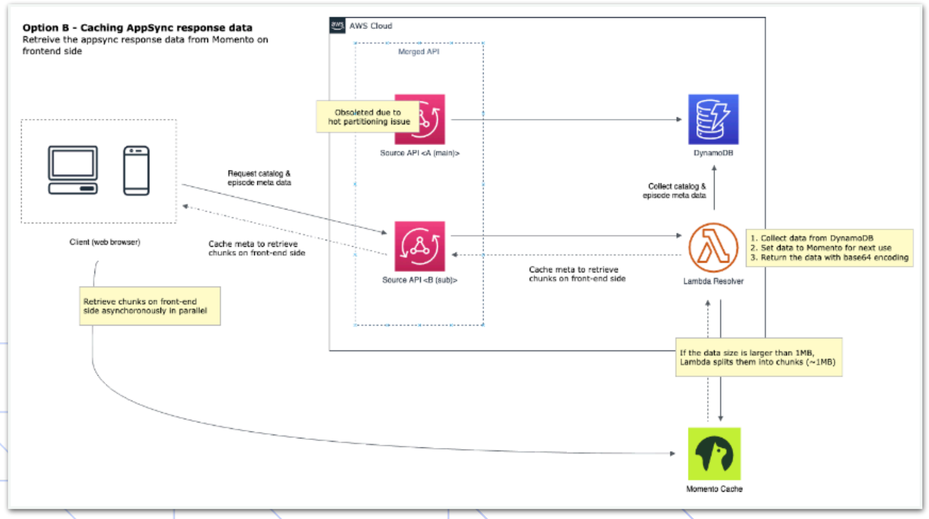
- 負荷試験を実施して、AWS AppSyncリゾルバーのTokenConsumedとAmazon DynamoDBのホットパーティショニング問題があることを確認



問題解決のための構成検討

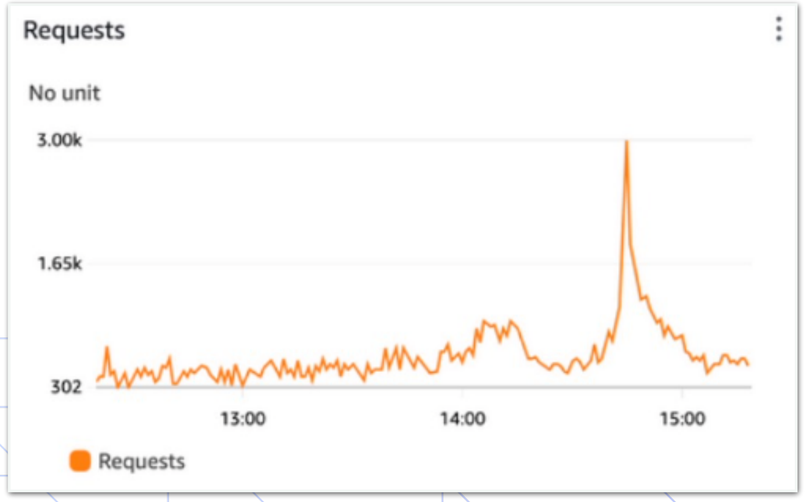
- キャッシュを導入してAWS内の負荷を軽減させ、AWS AppSyncのMerged APIを使って既存構成の変更を最小限に抑える
- さらにデータを圧縮・分割する仕組みを導入し、AWS AppSyncとAmazon DynamoDBへの負担が画期的に減らせたことを確認
- 別の機能でAWS AppSyncのTokenConsumedの上限緩和は一応必要だったため、必要な量のみ申請して承認

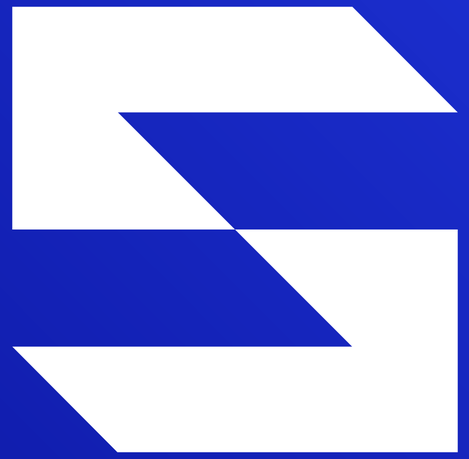




配信スタート及び結果

- テスト環境をセッティングして、AWS AppSyncのサービス上限緩和が完了した状態で負荷試験を実施
- 要件となっていた5,000TPSのスパイク発生時でもレイテンシーを維持しつつ耐えることを確認
- 新作配信開始後、想定した負荷に関する問題など特に発生せず、スムーズかつ通常どおり動画の視聴が可能





serverless.co.jp