game tech

安定したゲームリリースのための Well Architectedと負荷テスト

三上 卓也
アマゾン ウェブ サービス ジャパン株式会社
ソリューションアーキテクト

2019.11.20

自己紹介

三上 卓也(みかみ たくや)

ゲームエンターテイメントソリューション部
ソリューションアーキテクト

主にゲーム企業のお客様を担当し、お客様の
AWS活用を様々な形でご支援しています

好きなAWSのサービス：**Amazon QuickSight**

game tech

セッション構成

- ゲームリリースと性能について
- Well-Architectedの概要
- Well-Architected -パフォーマンス効率-
- システムテスト
- リリース直前の対応
- AWS 活用事例

ゲームリリースと性能について

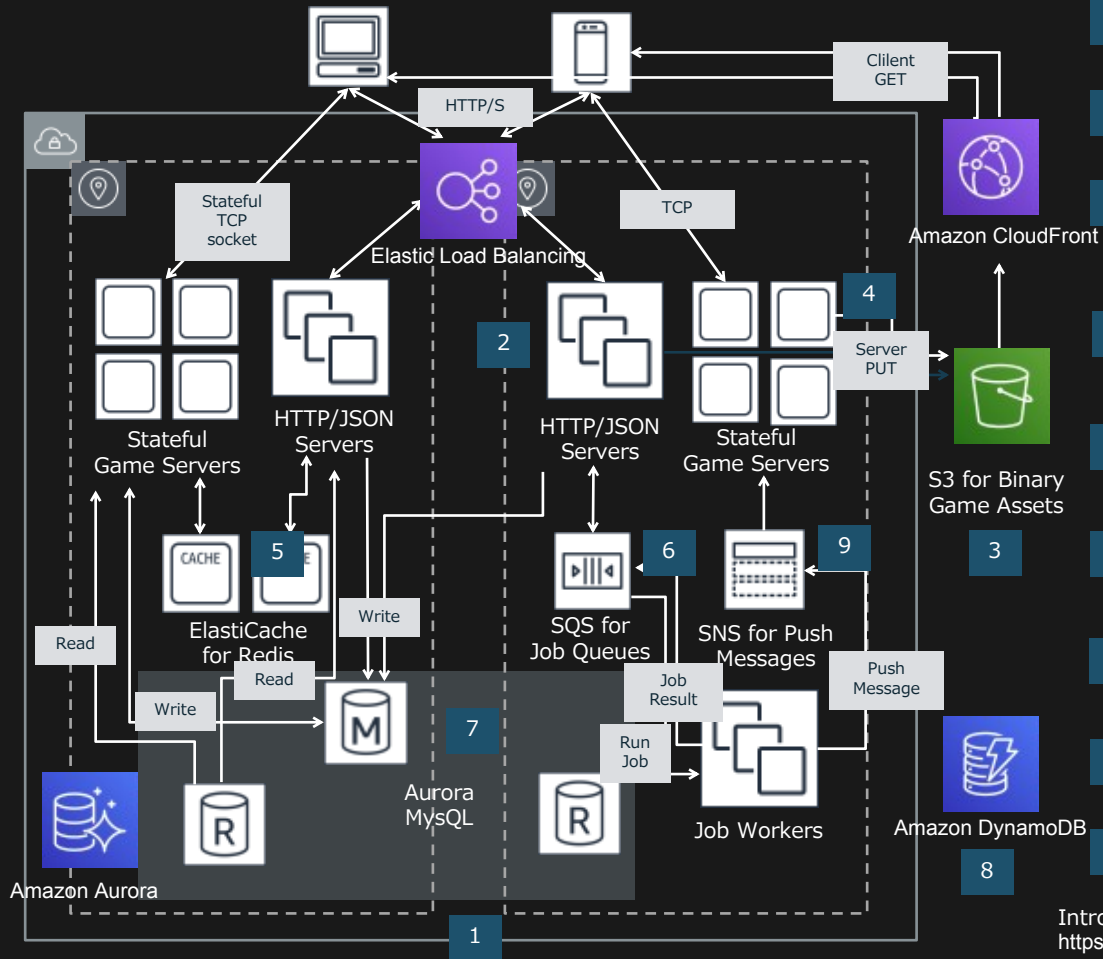
ゲームのワークロード

- リリース時に大量のアクセス
- 定期的なアップデート、イベント
- ユーザやメディアの評価
- SNSによる急なスパイク

スパイキーで高負荷なワークロードへ対応できる
アーキテクチャが求められる

game tech

ゲームのリファレンスアーキテクチャ



1 可用性向上のため2つのアベイラビリティゾーンにそれぞれ共通の機能を配置する。(図ではスペースの都合上一部省略)

2 HTTP/JSONサーバはELBのバックエンドで稼働し、Auto Scalingにより要求に応じて自動的にスケールする。

3 ゲーム上で利用するすべてのバイナリデータはS3上に配置。ダウンロードのパフォーマンス向上とコスト削減のためにCloudFront(CDN)を利用。

4 チャットやマルチプレイのゲームのためにステートフルなソケット通信を行う場合はHTTP/JSONサーバとは別にインスタンスを用意。

5 ゲームの規模が大きくなるにつれてDBの負荷も増加した場合、キャッシュ(ElastiCache)の導入も検討。よく利用する参照クエリのオフロードを実施。

6 非同期実行のJobはSQSを使ってコーディネート。２つ以上のコンポーネントとの疎結合アーキテクチャを構築。

7 DB負荷が増加した場合、参照クエリについてはReader(Read Replica)を増やすことで負荷分散が可能。また、Writer(Master)からReaderに参照クエリを移すことでWriterの負荷も削減。

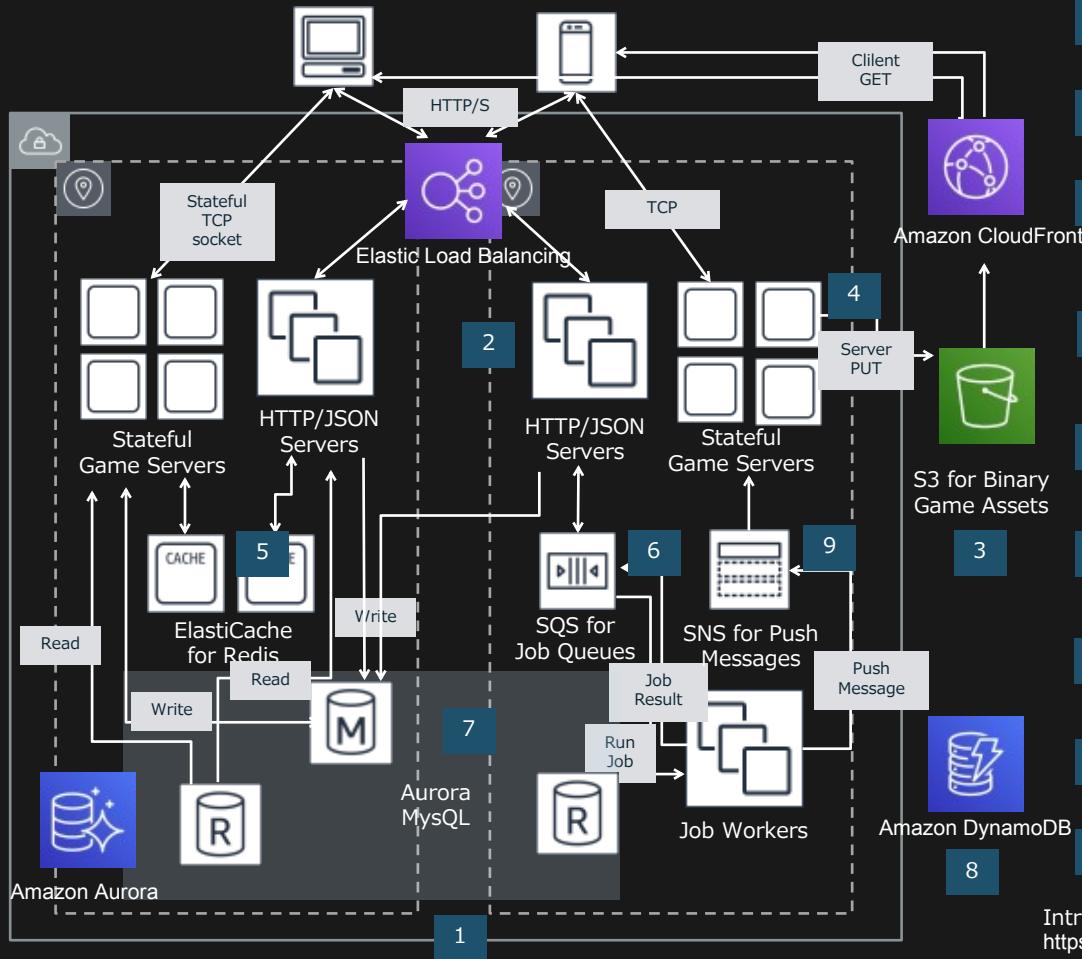
8 リーダーボードなどNoSQLを利用することでアドバンテージがあるものについてはElastiCacheやDynamoDBを利用。

9 複数のモバイルプラットフォームにプッシュ通知を簡単に送信するためにSNSやPinPointを利用。

game tech

Introduction to Scalable Gaming Patterns on AWS
https://d0.awsstatic.com/whitepapers/aws-scalable-gaming-patterns.pdf

ゲームのリファレンスアーキテクチャ



1 可用性向上のため2つのアベイラビリティゾーンにそれぞれ共通の機能を配置する。(図ではスペースの都合上一部省略)

2 HTTP/JSONサーバはELBのバックエンドで稼働し、Auto Scalingにより要求に応じて自動的にスケールする。

3 ゲーム上で利用するすべてのバイナリデータはS3上に配置。ダウンロードのパフォーマンス向上とコスト削減のためにCloudFront(CDN)を利用。

4 チャットやマルチプレイのゲームのためにステートフルなソケット通信を行う場合はHTTP/JSONサーバとは別にインスタンスを用意。

5 ゲームの規模が大きくなるにつれてDBの負荷も増加した場合、キャッシュ(ElastiCache)の導入も検討。よく利用する参照クエリのオフロードを実施。

6 非同期実行のJobはSQSを使ってコーディネート。２つ以上のコンポーネントとの疎結合アーキテクチャを構築。

7 DB負荷が増加した場合、参照クエリについてはReader(Read Replica)を増やすことで負荷分散が可能。また、Writer(Master)からReaderに参照クエリを移すことでWriterの負荷も削減。

8 リーダーボードなどNoSQLを利用することでアドバンテージがあるものについてはElastiCacheやDynamoDBを利用。

9 複数のモバイルプラットフォームにプッシュ通知を簡単に送信するためにSNSやPinPointを利用。

game tech

Introduction to Scalable Gaming Patterns on AWS
https://d0.awsstatic.com/whitepapers/aws-scalable-gaming-patterns.pdf

リリースまでのステップ

企画　プリプロ　設計　実装　テスト　リリース直前　リリース・・・

AWS Well-Architected

Well-Architected
Framework

システムテスト

リリース前の
サポート活用

これらを活用し安定したゲームリリースを実現

game tech

リリースまでのステップ

Well-Architected
Framework

Well-Architected 概要

AWS Well-Architected Frameworkとは?

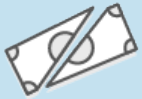
クラウドにおけるシステム設計・運用のベストプラクティス集

運用性

セキュリティ

信頼性

パフォーマンス効率



コストの最適化

AWSのソリューションアーキテクト(SA)が、
10年以上に渡り、**数多くのお客様**へ
技術支援をしてきた経験の集大成

効率的なTransformationを
Well-Architected FrameworkとAWSのSAが**お手伝い**

AWS Well-Architected

Framework

クラウド最適化の近道

game tech

AWS Well-Architected Frameworkの構成要素

① ベストプラクティスが記載された**Well-Architectedホワイトペーパー**

② 設計・構築・運用を支援する**AWSのソリューションアーキテクト(SA)**

③ [NEW]「ベストプラクティスとの**ギャップ**」 をセルフチェック出来る**AWS Well-Architected Tool**

①AWS Well-Architected Frameworkホワイトペーパー

運用の優秀性

セキュリティ

信頼性

パフォーマンス効率

コストの最適化

② AWSのSA

③ W-A Tool

AWS Well-Architected

game tech

AWS Well-Architected Frameworkの活用シーン

様々なフェーズでAWS W-Aを活用できる

要件検討	設計	構築	運用
システム要件の確認	**ベストプラクティスに則った設計**	**サービス開始前の確認**	**継続的なKAIZEN**
システム要件を検討する際の材料として、ホワイトペーパーを活用	ホワイトペーパーで、ベストプラクティスを理解し、設計を実施(リスクの把握)、W-Aレビューも実施	サービス開始前に、W-Aレビューでベストプラクティスに則っているかの確認(リスクの把握)	運用中のシステムに対して、W-Aレビューを実施して、リスクや改善出来る項目の洗い出しを実施

09/26 Well-Architected Toolが日本語対応

他言語に先駆けて日本語対応！東京リージョンもサポート！

- 構成検討段階で利用
- リリース直前の確認
- 健康診断のように定期的に確認
- 開発会社にToolによる確認を依頼し、出力レポートを共有してもらう
- 必要に応じてSAと一緒に確認

game tech

AWS Well-Architected Frameworkホワイトペーパー

設計原則と(質問と回答形式)のベストプラクティス集

- 運用の優秀性
- セキュリティ
- 信頼性
- パフォーマンス効率
- コストの最適化

AWS Well-Architected Frameworkホワイトペーパー

設計原則と(質問と回答形式)のベストプラクティス集

運用の優秀性	セキュリティ	信頼性	パフォーマンス効率	コストの最適化

- システム要件を満たすためのコンピューティングリソースの効率的な使用方法
- 需要の変化やテクノロジーの進化に応じてその効率を維持する方法
- パフォーマンス効率を高めるためのアーキテクチャに関するベストプラクティス

game tech

Well-Architected
パフォーマンス効率

パフォーマンス効率 -設計指針-

最新テクノロジーの標準化

Democratize advanced technologies

数分でグローバルに展開

Go global in minutes

サーバレスアーキテクチャを使用

Use serverless architectures

より頻繁に実験可能

Experiment more often

システムを深く理解

Mechanical sympathy

game tech

パフォーマンス効率 | 選択

選択

- ✓ システムのワークロードに合わせてアプローチを選択する
- ✓ 複数のアプローチを組み合わせて構成する

レビュー

- ✓ インフラストラクチャをコードで定義して簡単にパフォーマンステストのプロセスを構築可能にする
- ✓ AWSでは顧客のニーズに基づいたサービス、機能を継続的にリリースし、パフォーマンス効率を改善する可能性がある

モニタリング

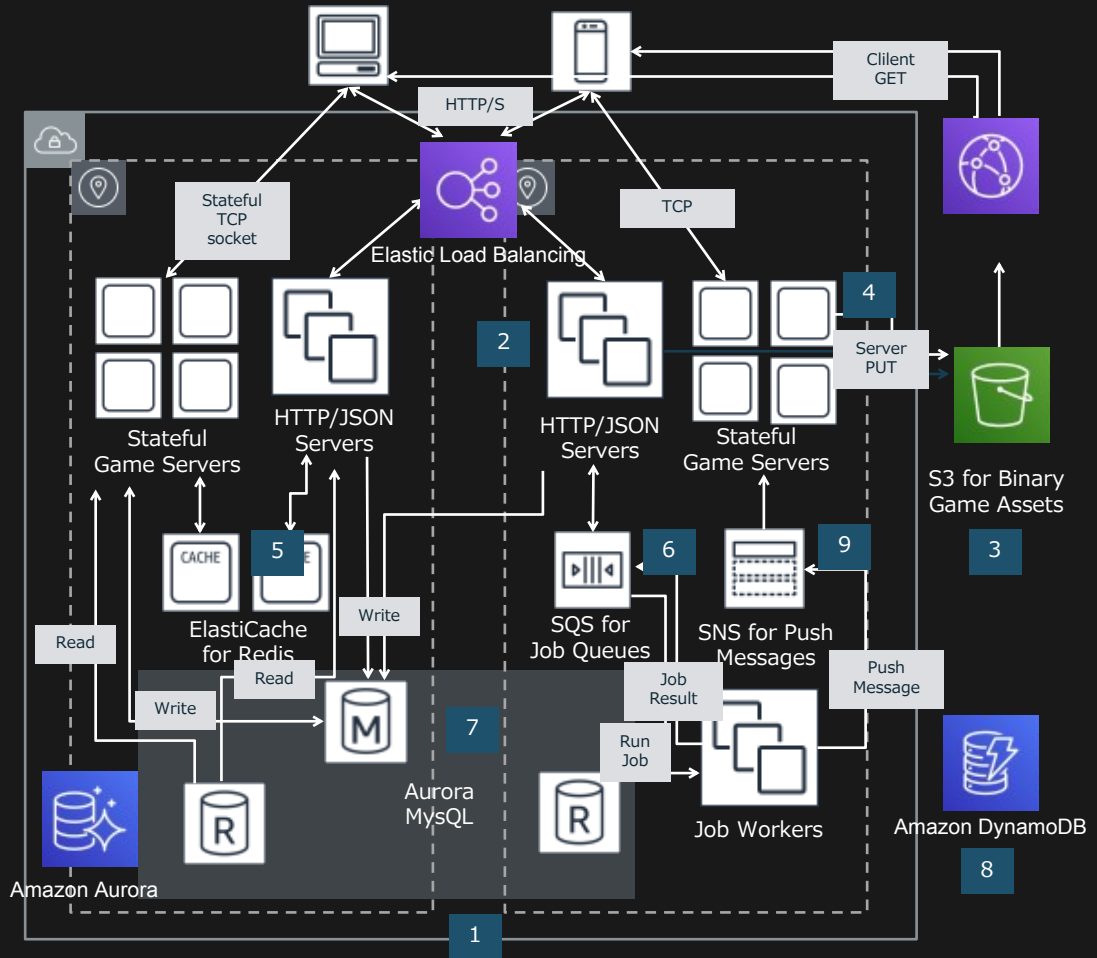
- ✓ 問題発生時にすぐ修正できるように、システムのパフォーマンスを監視する
- ✓ アラームを設定し自動化されたアクションをトリガーする

トレードオフ

- ✓ システムを設計するときに最適なアプローチを確保できるようにトレードオフを検討する

game tech

ゲーム機能に求められるワークロードを考慮



ゲームでは様々な機能を提供

- アセット配信
- マッチメイキング
- マルチプレイ
- リーダーボード
- etc.

機能に求められるワークロードに応じてサービスを選択

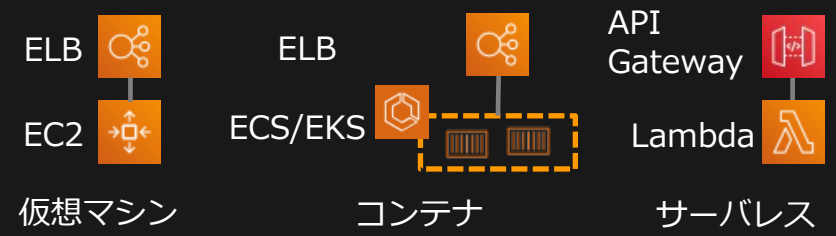
- コンピュート
- ストレージ
- データベース
- ネットワーク

game tech

各技術で適切なサービスを選択

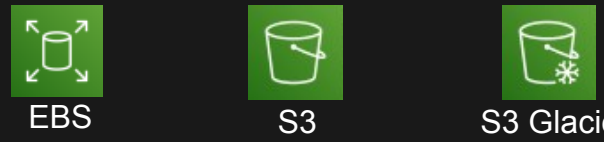
コンピュート

- 仮想マシン/コンテナ/サーバレスのソリューション
- ワークロードに合わせてインスタンスを選択可能



ストレージ

- EBSではワークロードでタイプを選択
  →SSD(io1,gp2) 、HDD(st1,sc1)
- サイズの大きいアセットファイルはS3に格納し、CloudFrontで配信
- ログのアーカイブにはS3 Glacierを活用

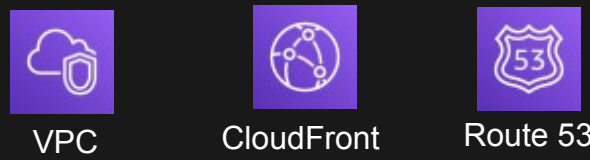


データベース

- 4つの要素、アクセスパターン/データベース特性/設定オプション/運用負荷を考慮して選択
- 詳細は後述

ネットワーク

- レイテンシー、スループットに合わせて選択
- CloudFrontによりアセットデータ配信の高速化
- マルチプレイサーバをユーザに近いリージョンに配置



game tech

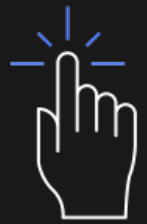
ゲームにおけるデータベースとユースケース

データベースタイプ	ユースケース	AWSのサービス
リレーショナル	インベントリ、ショップ、取引アイテム、クエストの状態、リーダーボード	Amazon Aurora Amazon RDS Amazon Redshift
キーバリュー	プレイヤー/アイテムの統計/プロファイル、特にアップグレードとエンチャント、ゲームの状態、クエストデータ	Amazon DynamoDB
インメモリ	リーダーボード、セッション管理、プレイヤーの統計/プロファイル、メトリック、チャット また本項目の多くのデータをすばやく保存し、取得するために使用されますが、一時的であるため、最終的にはディスクベースの DBに書き込む必要あり	Amazon ElastiCache ElastiCache for Redis ElastiCache for Memcached
ドキュメント	クエスト/プレイヤー/アイテムの説明	Amazon DocumentDB (with MongoDB compatibility)
グラフ	チート検出、フレンド、アイテム/クエストの推奨、ダイナミック NPC 関係	Amazon Neptune
時系列	プレイヤーパターン、エコノミー	Amazon Timestream
台帳	エコノミー、アイテムの取引/オークション、ゲームからゲームに移行できるアイテム/キャラクター	Amazon Quantum Ledger Database

Making a game? There's a database for that. https://aws.amazon.com/gametech/database/?nc1=h_ls

game tech

パフォーマンス効率 | レビュー



選択

- ✓ システムのワークロードに合わせてアプローチを選択する
- ✓ 複数のアプローチを組み合わせて構成する

レビュー

- ✓ インフラストラクチャをコードで定義して簡単にパフォーマンステストのプロセスを構築可能にする
- ✓ AWSでは顧客のニーズに基づいたサービス、機能を継続的にリリースし、パフォーマンス効率を改善する可能性がある

モニタリング

- ✓ 問題発生時にすぐ修正できるように、システムのパフォーマンスを監視する
- ✓ アラームを設定し自動化されたアクションをトリガーする

トレードオフ

- ✓ システムを設計するときに最適なアプローチを確保できるようにトレードオフを検討する

環境構築のコード化

CloudFormationを活用することで試験環境を繰り返し構築が可能となり、
様々な条件や新しいAWSサービスのメトリクス取得してレビューを実施
グローバル展開時に別リージョンへの環境構築が容易に

AWS Trusted Advisor

AWS Config

AWS CloudFormation

AWS Systems Manager

AWS OpsWorks

Amazon CloudWatch

game tech

コスト効率の良い負荷テスト

コスト効率良く、パフォーマンス効率及び新サービスの検証を実施

game tech

AWSサービスの活用

クラウドサービスを活用することで既存のゲーム機能を効率良く実装

Lambda@Edge を使ったゲームプレイヤーへのカスタムコンテンツの配信

by AWS Japan Staff | on 10 SEP 2019 | in Amazon CloudFront, AWS Lambda, Game Development, Lambda@Edge | Permalink | Share

モバイルゲームのクライアントアプリを起動した時に、いつも最新のコンテンツを入手できるようにしたいですよね？そうすれば、ゲームへ変更を加えて、新しい機能を追加した際、変更内容をすぐにプレイヤーに提供できるようになります。

しかし、すべてのプレイヤーがゲームを最新バージョンに更新したことを常に保証することはできません。では、古いバージョンのゲームクライアントが以前のゲーム形式 でデータを取得できるようにするためにはどうすればいいでしょうか？またプレイヤーがゲームを実行するたびに、データがチェックおよびダウンロードされますが、どうやって高速な配信を保証すれば良いでしょうか？

何をすべきか

この問題には様々な解決方法があります。ほとんどの解決方法はプレイヤーのリクエストをみて、どのファイルをレスポンスで返すかを決めるコードを実行する web 機能を実装します。

https://aws.amazon.com/jp/blogs/news/how-to-deliver-custom-game-content-to-players-using-lambdaedge/

概要

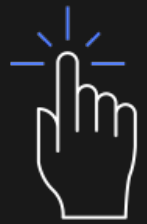
- ゲームのバージョンごとに別のバックエンドを利用するケース
- ユーザがインストールしているバージョンごとにエンドポイントを変える
- クライアントのリクエストにインストールされているバージョンを含め、Lambda@Edgeでそのバージョンに応じてエンドポイントを返す

メリット

- オリジンの負荷低減
- ユーザエクスペリエンスの向上

game tech

パフォーマンス効率| モニタリング



選択

- ✓ システムのワークロードに合わせてアプローチを選択する
- ✓ 複数のアプローチを組み合わせて構成する

レビュー

- ✓ インフラストラクチャをコードで定義して簡単にパフォーマンステストのプロセスを構築可能にする
- ✓ AWSでは顧客のニーズに基づいたサービス、機能を継続的にリリースし、パフォーマンス効率を改善する可能性がある

モニタリング

- ✓ 問題発生時にすぐ修正できるように、システムのパフォーマンスを監視する
- ✓ アラームを設定し自動化されたアクションをトリガーする

トレードオフ

- ✓ システムを設計するときに最適なアプローチを確保できるようにトレードオフを検討する

パフォーマンス効率 -モニタリング-

アプリケーションの
正常性をチェックする

アプリケーションを
元の状態に回復する

トラブルの原因を
調査する

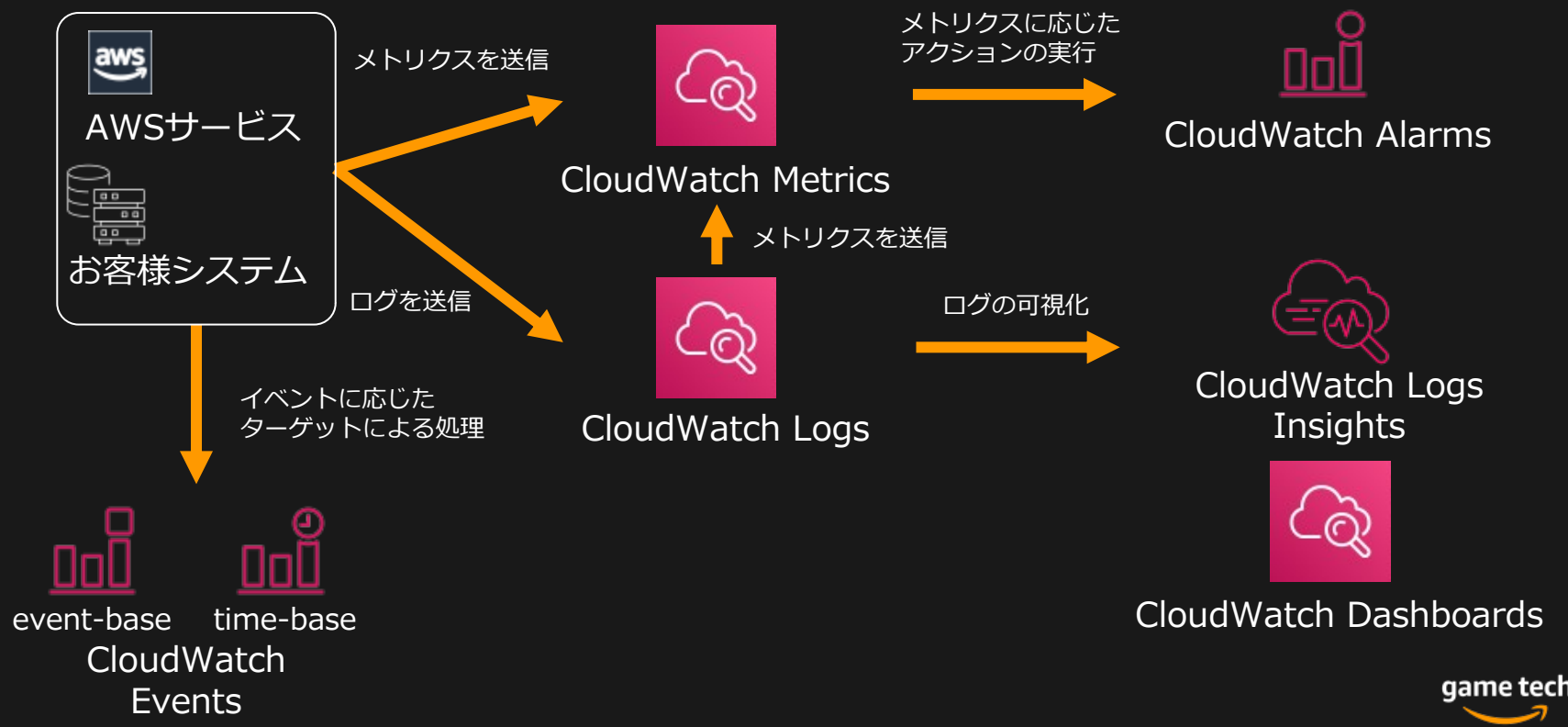
ユーザー行動を
分析する

キャパシティを
分析する

game tech

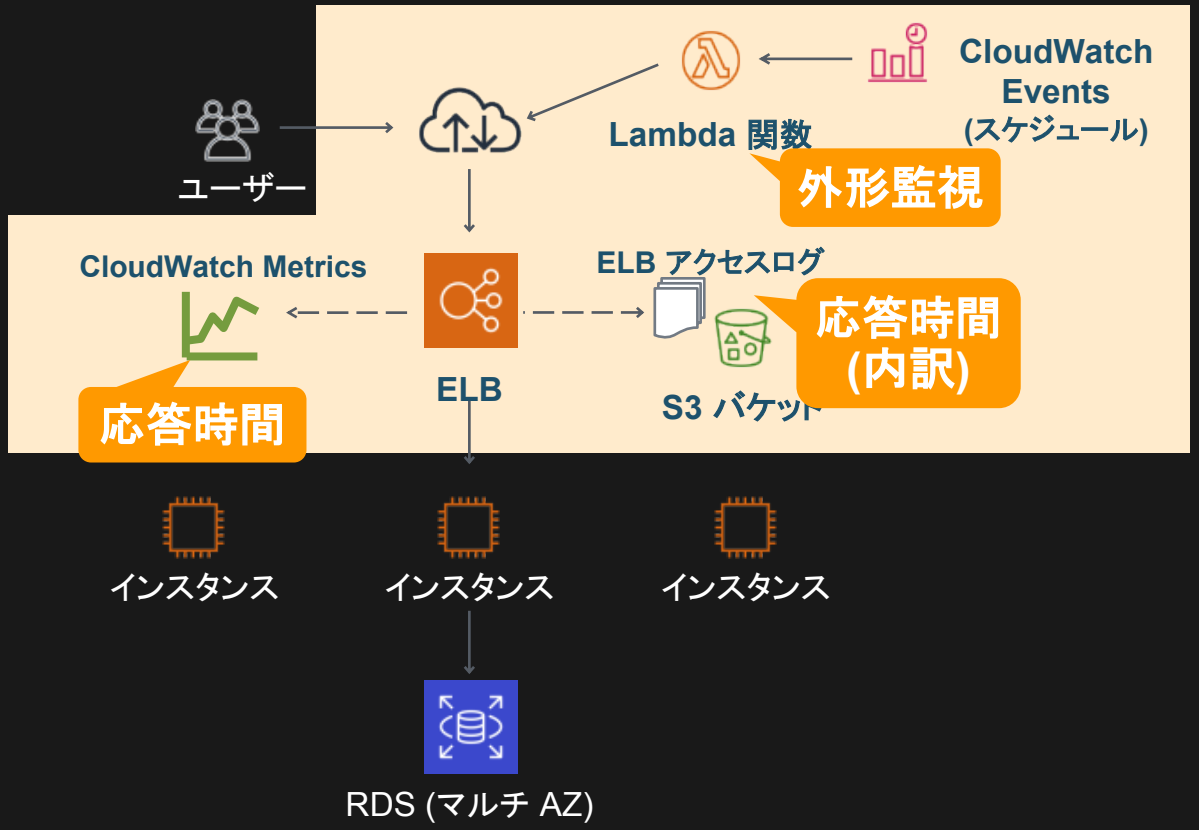
Amazon CloudWatch

CloudWatch はモニタリングに関する様々な機能を提供

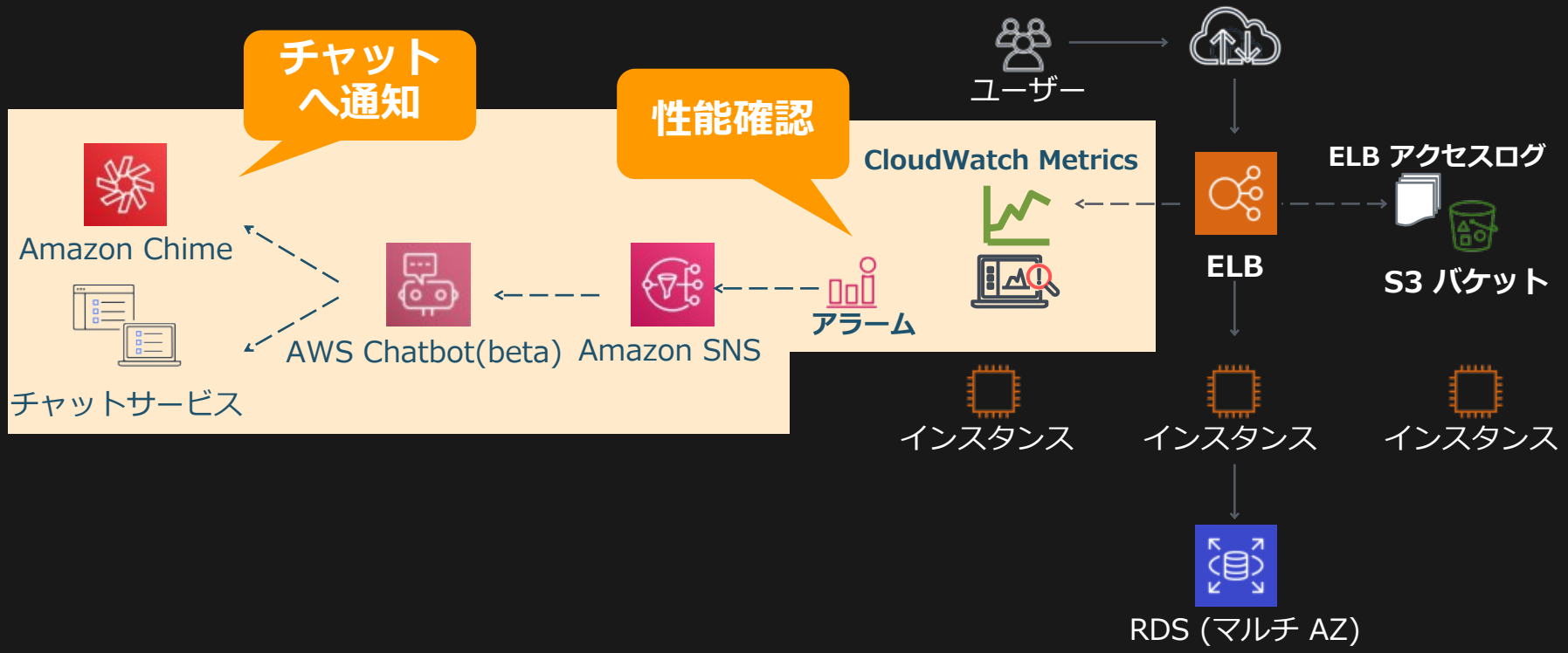


game tech

外形監視によりサービス状態を把握

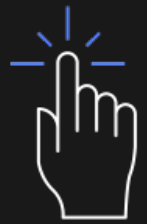


収集したメトリクスからボトルネックを確認



game tech

パフォーマンス効率 | トレードオフ



選択

- ✓ システムのワークロードに合わせてアプローチを選択する
- ✓ 複数のアプローチを組み合わせて構成する

レビュー

- ✓ インフラストラクチャをコードで定義して簡単にパフォーマンステストのプロセスを構築可能にする
- ✓ AWSでは顧客のニーズに基づいたサービス、機能を継続的にリリースし、パフォーマンス効率を改善する可能性がある

モニタリング

- ✓ 問題発生時にすぐ修正できるように、システムのパフォーマンスを監視する
- ✓ アラームを設定し自動化されたアクションをトリガーする

トレードオフ

- ✓ システムを設計するときに最適なアプローチを確保できるようにトレードオフを検討する

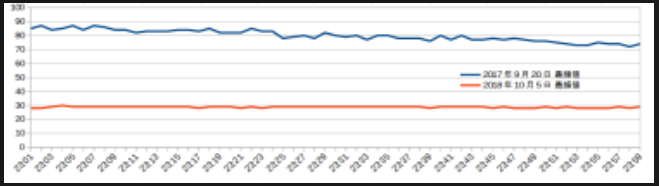
トレードオフを考慮したアーキテクチャ

技術により消費するものと得られるメリットのトレードオフを考慮する

技術	ワークロード	消費	メリット	AWSサービス
キャッシュ	Read-heavy	Space(Memory)	Time	• ElastiCacheによるクエリ結果のキャッシュ • CloudFrontによるコンテンツキャッシュ
パーティショニング シャーディング	Write-heavy	Size & Complexity	Time	• RDSで水平/垂直分割時の処理を実装 • パーティション管理をDynamoDBに任せる
圧縮	Large data	Time	Space	• CloudFrontでgzip圧縮による配信 • Redshiftで圧縮してデータを格納
バッファリング	Many requests	Space & Time	Efficiency	• SQSで疎結合なアーキテクチャ

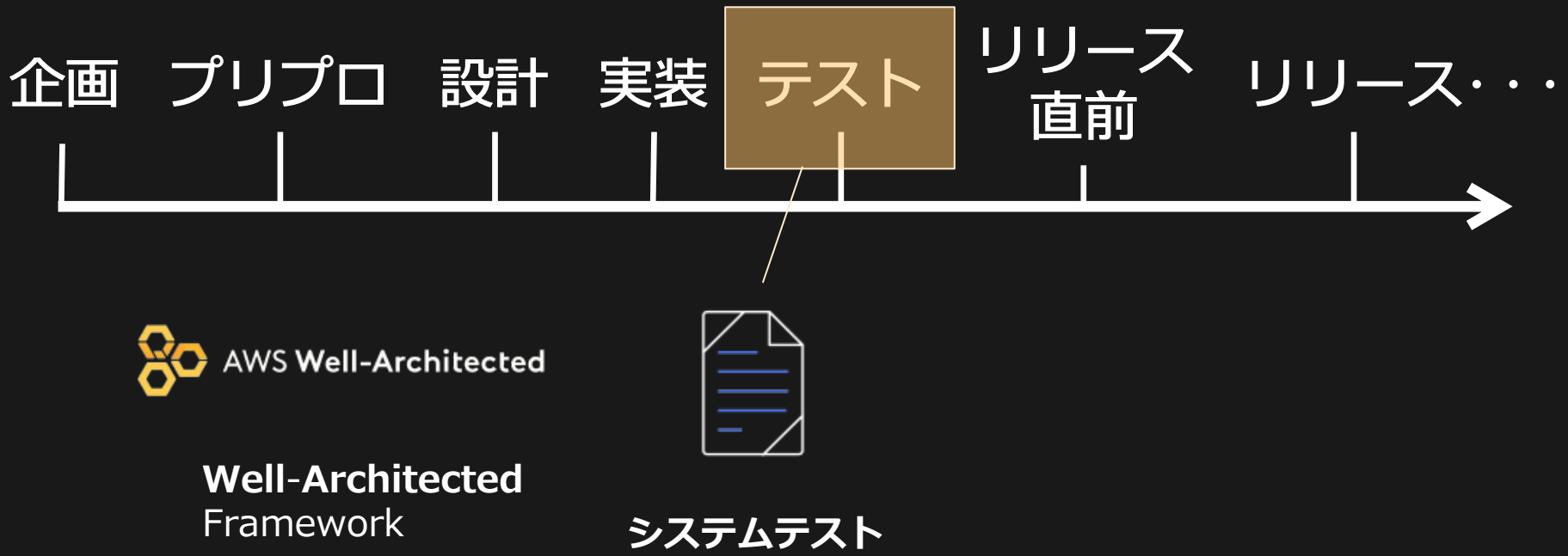
Fate/Grand Orderにおける
大規模なデータベース移行と負荷試験

- **大規模なデータベース移行を実施する上でのデータベース設計の見直し**
  - データ量増加と集中アクセスにより、APIレイテンシー増大
  - DB垂直分割によるスケール問題と運用問題が発生し、水平分割を実施
- **移行後に想定トラフィックを捌けることを検証するための負荷試験手法**
  - 長期間実行、ユーザに近い挙動を再現などを要素を考慮して負荷試験を行う
  - ピーク時の２倍のアクセスで目標達成
- 結果
  - FGO3周年キャンペーン7倍のアクセスがlatencyは微増

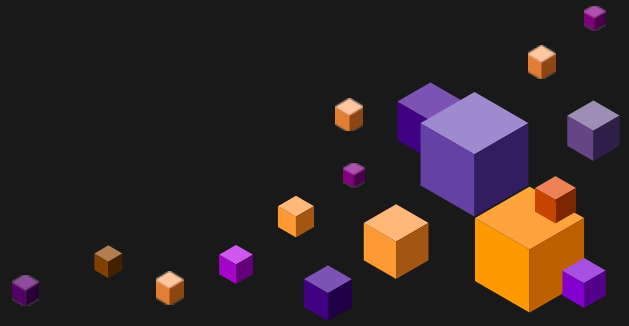


game tech

リリースまでのステップ



システムテスト



オンラインゲームで実施するテスト

機能テスト

- デバッガによるゲームテスト

非機能テスト

負荷テスト

- 性能要件を満たしているかの確認
- ボトルネック箇所の特定
- スケール特性の把握
- AWS側のスロットリングや追加の上限緩和の要否を判断するため
- 長時間テスト

障害テスト

- 各コンポーネントの障害におけるサービスへの影響を把握

ペネトレーションテスト

- セキリュティ脆弱性を把握

game tech

負荷テストにまつわる問題

- スケジュールに追われて負荷テスト省略
- クライアントから十分な負荷をかけられていない
- ゴールが設定されておらず、どこまで負荷をかける？
- ボトルネックが特定できないまま終了
- RDSのスケールアップだけのチューニング
- AWSへの事前申請がされておらず、負荷がかからない

リリース時のトラブルは負荷テストによって事前に防ぐことができるケースが多いため、確実に実施する必要がある

game tech

負荷テストで確認したいポイント

- 実際のワークロードでテストを行っているか？(単純ベンチマークではない)
- 目的のスループット/レイテンシが処理できるか？
- システム上、どこにボトルネック箇所が存在するか？(RDB? 外部API?)
- 各コンポーネントがスケールすると、どのようなパフォーマンス影響があるか？
- 各コンポーネントの障害時、どのようなサービス影響があるか？

システム特性を把握し、改善点を明らかにする

負荷テストのゴール設定例

何を達成すれば良いのか、ゴール設定を正しくすることが重要

ビジネス側の指標
DAU(Daily Active User)

システム側の指標
スループット(rps)

最大rps = DAU x ユーザあたりのアクセス数 x ピーク時倍率 / 86,400(秒)

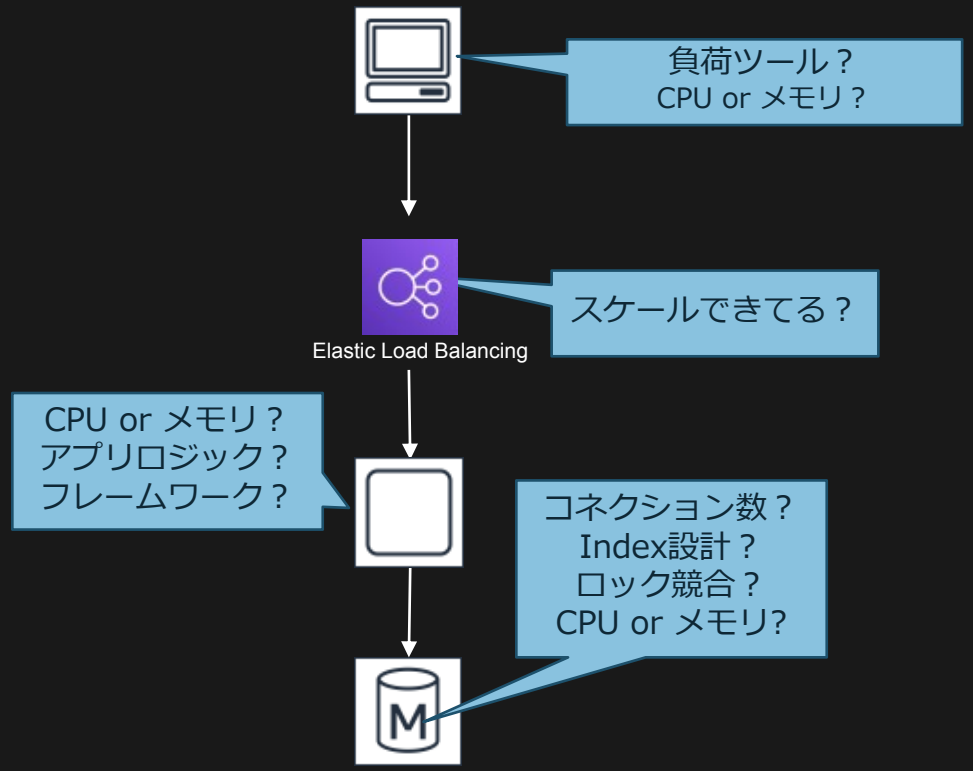
ex.

DAU=10万
ユーザあたりのアクセス数 = 20回
ピーク時倍率 = 5.0

10万 x 20回 x 5.0 / 86,400 = 115.7rps

game tech

ボトルネックの特定

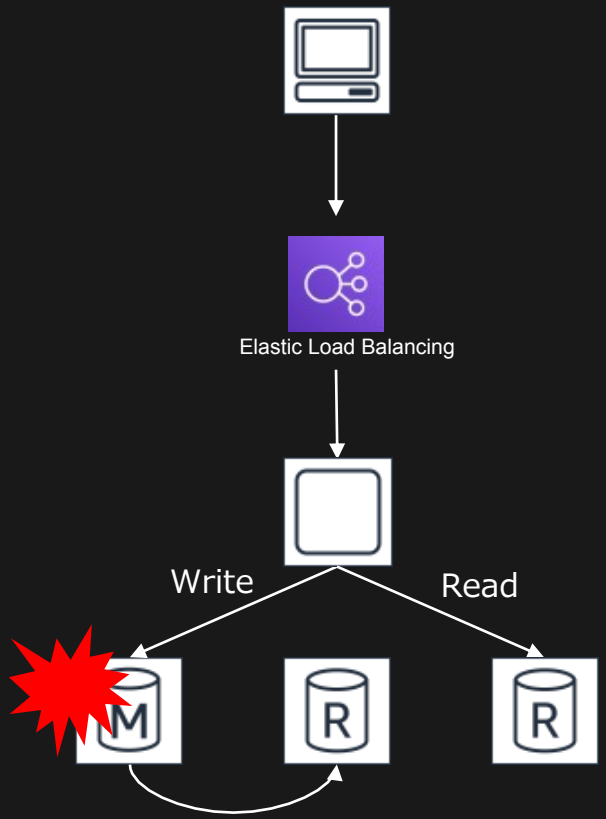


- 取得可能なメトリクスを確認し、必要があれば有効化を行う (ex. RDSのEnhanced Monitoring, Performance Insight, Performance Schema)
- アプリケーションのプロファイラも有効
- 全体の負荷テストでボトルネックが特定できない場合は、個々のコンポーネントを対象とする
- ドキュメントを確認してAWSサービスのメトリクス内容を把握する

Auroraを含めた負荷テストで気をつけるべき点

- 本番のワークロードで負荷テストを実施
  - キャッシュ周りの動作を正確につかむために1時間以上のテストを推奨
- パラメータグループはまずデフォルト設定のまま
  - innodb_buffer_poolは搭載メモリの70%までを推奨
- max_connectionsは最大16,000
  - インスタンス数 x コネクション数を把握
  - アプリ側のスケールアウトによるコネクション溢れに注意
- ロックによるパフォーマンス影響もチェック

障害テスト



- 実際のワークロードによる負荷テスト中に障害テストを実施することで以下を確認
  - アプリケーション側のエラーハンドリング
  - レスポンス低下時のサービス影響
  - データ不整合の有無
- Auroraでは障害挿入クエリを実行して障害をシミュレーション可能
  ex.) ALTER SYSTEM CRASH
- ChaosMonkeyによりAWS上の本番システムでインスタンスをランダムに落としまくる(Chaos Engineering)

game tech

セキュリティテスト

AWS上のセキュリティテストの規定

- 2019年3月より以下の8 つのサービスへ事前承認無くテストが許可
  - EC2,NATゲートウェイ,ELB,RDS,Aurora,CloudFront,APIゲートウェイ,Lambda,Lightsail,Beanstalk
- 以下のアクティビティは禁止
  - Route 53 ホストゾーン経由の DNS ゾーンウォーキング, DoS, DDoS, ポートフラッディング,プロトコルフラッディング,リクエストフラッディング

Amazon Inspectorのエージェントレス型ネットワークアセスメント

- エージェントレスでインスタンスの空きポートを確認可能
- エージェントインストールでさらに内部の脆弱性なども診断可能

Finding	On instance i-01de622f7c540bcc5, TCP port 23 which is associated with 'Telnet' is reachable from the internet
Severity	High
Description	On this instance, TCP port 23, which is associated with Telnet, is reachable from the internet. You can install the Inspector agent on this instance and re-run the assessment to check for any process listening on this port. The instance i-01de622f7c540bcc5 is located in VPC vpc-e6a88f9d and has an attached ENI eni-ea992e72 which uses network ACL acl-56d65d2c. The port is reachable from the internet through Security Group sg-5e945516 and IGW igw-f93d5680
Recommendation	Edit the Security Group sg-5e945516 to remove access from the internet on port 23

https://aws.amazon.com/jp/security/penetration-testing/

game tech

テストの際に行うAWSへの申請(1)

ELB pre-warming(暖気) 申請

- 急激なリクエスト増加に対応するための事前申請
- CLB/ALBが対象(NLBは必要なし)
- Business/Enterpriseサポートが対象
- サポートケースにより申請

上限緩和申請

- AWSの各サービスにはハードリミット/ソフトリミットがある
  - ハードリミット＝増やすことができない上限
  - ソフトリミット=申請により増やすことができる上限
  - サポートケースにより申請

AWSサービス制限: https://docs.aws.amazon.com/ja_jp/general/latest/gr/aws_service_limits.html

実施の1~2週間前の申請をお願いいたします

game tech

テストの際に行うAWSへの申請(2)

ネットワーク負荷テスト申請

- 以下の1つ以上の条件に当てはまるトラフィックを生成する場合に申請が必要
  - 持続的に1分以上の間 1 Gbps（秒間 1億ビット）あるいは 1Gpps（秒間 1億パケット）を超えるトラフィック
  - 悪用目的や悪意があるように見えるトラフィック
  - テスト対象以外（ルーティングや共有サービスのインフラストラクチャなど）へ潜在的に影響を及ぼしうるトラフィック
- aws-security-simulated-event@amazon.com にメールで申請

Amazon EC2 Test Policy: https://aws.amazon.com/jp/ec2/testing/

ペネトレーションテスト申請

- 適性利用規約で禁止されているセキュリティ違反やネットワーク不正使用とテストによるシミュレーションを区別するために申請が必要
- すべての侵入テストに許可が必要
- リクエストフォームから申請
- 容量測定のためのネットワークベースの DDoS シミュレーションは明示的に禁止されている

侵入テスト: https://aws.amazon.com/jp/security/penetration-testing/

実施の1~2週間前の申請をお願いいたします

game tech

AWSサポートの活用(おすすめ！)

- 開発/試験時のトラブル解決を効率化
- 開発の段階での加入をお勧め
- 全AWSサービスが対象
- お問い合わせ回数は無制限
- **日本語サポート**

	ベーシック	開発者	ビジネス	エンタープライズ
				最優先対応
				テクニカル アカウント マネージャー
			Full Trusted Advisor	Full Trusted Advisor
技術Q&A		✓ Web(メール) ✓ 平日 9-18 時 ✓ 最短 12 時間	✓ Web(メール) ✓ **電話・チャット** ✓ **24/365 対応** ✓ **最短 60 分**	✓ Web(メール) ✓ 電話・チャット ✓ 24/365 対応 ✓ 最短 15 分
料金Q&A	料金Q&A	料金Q&A	料金Q&A	料金Q&A (コンシェルジュ)
フォーラム	フォーラム	フォーラム	フォーラム	フォーラム

開発環境

開発 + プロダクション環境

リリースまでのステップ

企画　プリプロ　設計　実装　テスト　リリース直前　リリース・・・

AWS Well-Architected

Well-Architected
Framework

システムテスト

リリース前の
サポート活用

game tech

リリース直前の対応

AWSへの申請(再び)

- ELB(CLB/ALB)のpre-warming(暖気運転)申請
- 上限緩和申請
  - CloudFront(データ転送レート:40Gbps/リクエスト:100,000/s)
  - 負荷テストで必要だと判断されたAWSサービスの上限緩和
  - 可能な限り早めに申請(サービスや量によってはリードタイム必要)

リソースの確保

EC2やRDSではリクエストに対応するために必要なオンデマンドキャパシティが一時的に不足することがある (InsufficientInstanceCapacity)

本番直前に大量に起動する場合に要注意

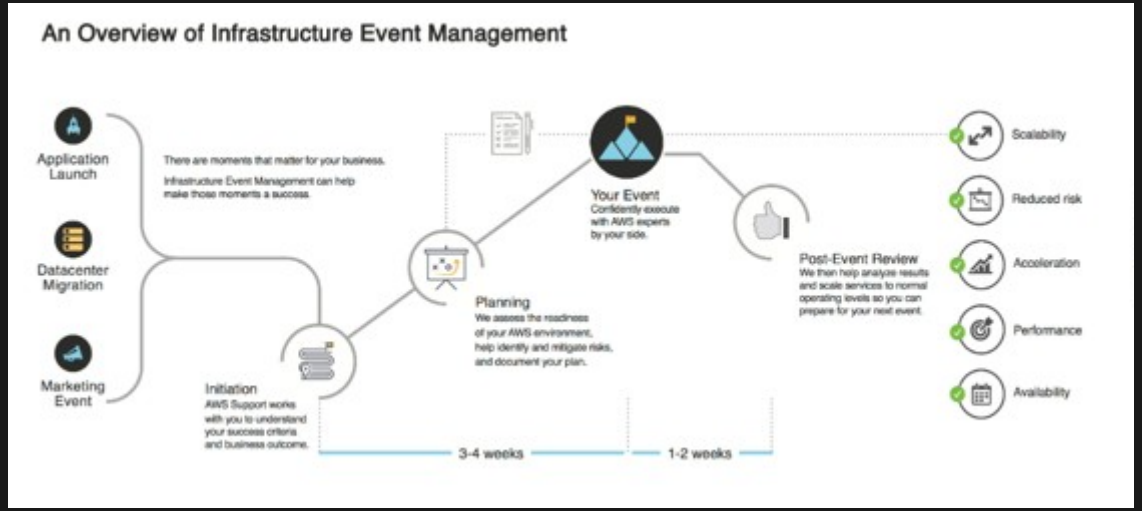
<u>ソリューション</u>

- リリースに間に合うように必要なインスタンスをあらかじめ起動しておく
- **オンデマンドキャパシティ予約**を利用する
- (リザーブドインスタンスを購入しておく)

https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/troubleshooting-launch.html

Infrastructure Event Managementの準備

- 本番リリース時に備えたガイダンス、本番時のリアルタイムのサポートを提供
- AWSサポートのエンタープライズサポートプランに含まれる
  (ビジネスサポートプランでも追加料金で利用可能)
- 大規模で重要度の高いゲームタイトルでの実施をおすすめ
- リリースの1~2ヶ月前にお申し込みをお願いします



https://aws.amazon.com/jp/premiumsupport/iem/

game tech

Go Live

企画　プリプロ　設計　実装　テスト　リリース直前　リリース・・・

AWS Well-Architected

Well-Architected
Framework

システムテスト

リリース前の
サポート活用

game tech

事例

ネクソンの多彩なゲームコンテンツポートフォリオを支える基盤と運用の最適化について

- クラウドオリエンテッドに設計したモバイルゲームタイトルの構築フェーズからリリース／運用で押さえるべきポイント！-

1. クラウドオリエンテッド

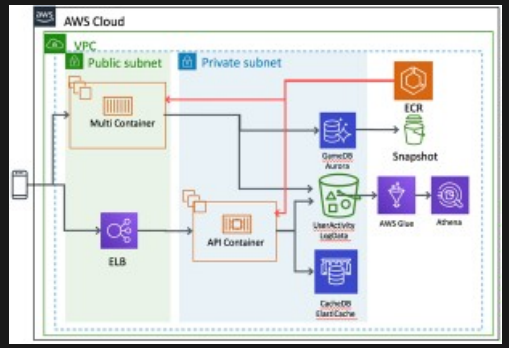
- 費用/運用適正化のためにマネージドサービスを積極的に採用して設計(Amazon Aurora/ Amazon ECS / Amazon Athena etc.)

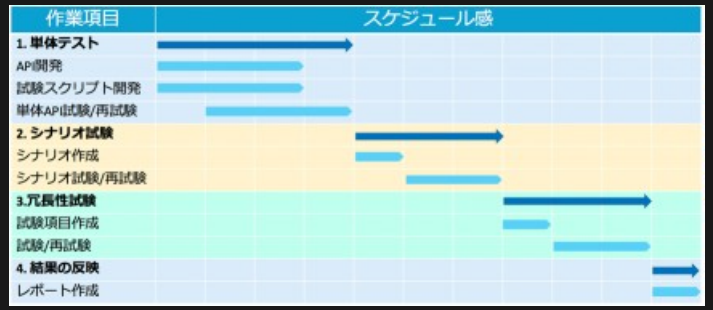
2. 設計/構築の工夫

- 入念な試験の実施により新規導入のサービスについて、高負荷時にでも十分な性能がでるか、フェイルオーバーなどの運用時に発生する動作に問題がないか確認

3. エンタープライズサポートの活用

- Trusted Advisorの確認を運用ルール化し、月に一度改善内容を計画して定期的な構成改善を実施
- TAM＆SAと構築段階から連携し問題を未然に防ぐ





game tech

ロマサガRSの大規模負荷を処理するAmazon ECS & Docker運用知見

概要

- 2018/12に(株)スクウェア・エニックスから23年ぶりに「ロマンシング サガ」完全新作 『ロマンシング サガ リ・ユニバース』がモバイル版としてリリース。リリース後1ヶ月弱で1,000万ダウンロードを達成した高トラフィックを処理するゲームシステムの裏側をアカツキがいかに構築したかを解説する。

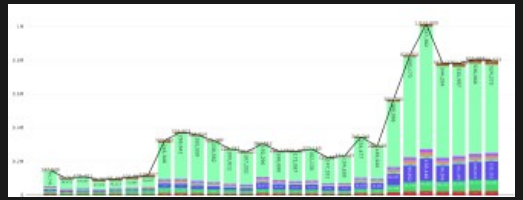
アーキテクチャについて

- サーバーサイドの開発言語はElixir
  - 並列性が高く十分に高速
- 100%Docker運用
  - オーケストレーションはECSを採用
  - 構成管理するプロセスは全てDocker化している
  - 開発はデプロイしているものと同じImageをローカルで起動
- 99%のAWSリソースをCloudFormationで管理
  - kumogata/kumogata2でコード管理

game tech

ロマサガRSの大規模負荷を処理するAmazon ECS & Docker運用知見

- **ロギング**
  - CloudWatch Logs：エラーのみフィルタしたアプリケーション / ミドルウェアログ
  - Datadog Logs Management：エラーのみフィルタしたミドルウェアログ
  - S3+Athena：アクセスログ等はS3に保存してAthenaで分析
- **監視**
  - CloudWatch ：エラー監視、生死監視、リソース監視、ログ監視
  - Datadog Infrastructure：エラー監視、生死監視、リソース監視、プロセス監視
  - Datadog APM：性能監視
  - Datadog Logs Management：ログ監視
  - Sentry：エラー監視
- **分析**
  - 行動ログをRedashで可視化



game tech

ロマサガRSの大規模負荷を処理するAmazon ECS & Docker運用知見

負荷対策について

- 大前提として負荷テスト & 改善フェーズは必ず確保する
- 負荷テストツールとしてLocustを採用
- 目的ごとの負荷テストを実施
  - 単性能テスト、過負荷テスト、高負荷テスト、障害テスト、長時間テストという目的別のテストを実施
- ECSのスケーリングを工夫して高速化
  - ECSのサービスオートスケーリングは使用せず、EC2のスケールアウトによるEC2起動完了イベントをCloudWatchEventsによりHookしてLambda実行しECSタスクを追加する

運用の工夫について

- CodeBuildを利用して複数Imageを並列性高くビルド実行
- Datadog InfrastructureのAlert検知によりLambdaを起動してECSタスクの自動復旧パイプラインを構築している

game tech

まとめ

まとめ

- Well-Architected の活用
  - ワークロードに合わせてサービスを選定して設計
  - 繰り返してテストできる環境を用意し新サービスも利用を検討
- テストの実施
  - 負荷テストを通して目標性能を満たしているか確認
- リリース前のサポート活用
  - リリース前には必要に応じてAWSと連携する

Thank you