Amazon GameLift 101

AWS Black Belt Online Seminar

安藤 怜央

アマゾン ウェブ サービス ジャパン合同会社
ソリューションアーキテクト

2023/02

AWS Black Belt Online Seminarとは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWSの技術担当者が、AWSの各サービスやソリューションについてテーマごとに動画を公開します
- 動画を一時停止・スキップすることで、興味がある分野・項目だけの聴講も可能、スキマ時間の学習にもお役立ていただけます
- 以下のURLより、過去のセミナー含めた資料などをダウンロードすることができます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/
  - https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY

内容についての注意点

- 本資料では 2023 年 2 月時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(https://aws.amazon.com/)にてご確認ください
- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます
- 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます

自己紹介

名前：安藤 怜央（あんどう れおう）

所属：アマゾン ウェブ サービス ジャパン合同会社
技術統括本部 ゲームエンターテインメント
ソリューション部
ソリューション アーキテクト

好きな AWS サービス：

Amazon GameLift

AWS Control Tower

AWS Systems Manager

本セミナーの対象者

- 専用ゲームサーバーの開発や運用で課題を感じている方
- AWS を使ってマルチプレイヤーゲームの開発を検討されている方
- Amazon GameLift について基礎的な内容を抑えたい方

アジェンダ

1. マルチプレイヤーゲームにおける専用ゲームサーバーホスティングの課題
2. Amazon GameLift の概観
3. GameLift Managed Hosting の仕組み
4. GameLift Managed Hosting のリソース
5. GameLift ゲームサーバーのライフサイクル
6. 料金情報
7. まとめ

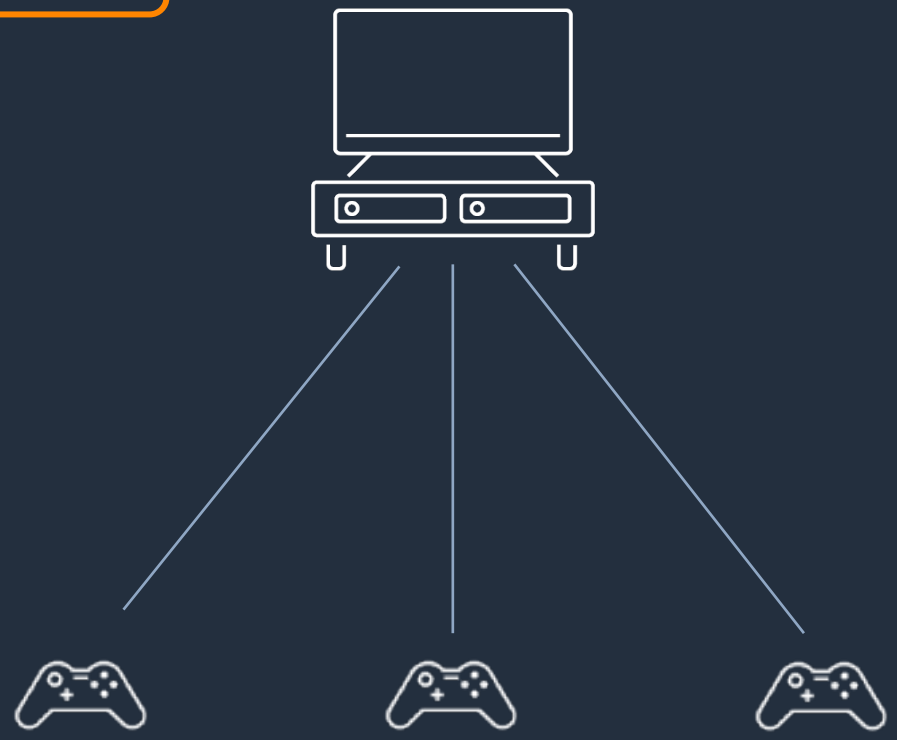
アジェンダ

1. マルチプレイヤーゲームにおける専用ゲームサーバーホスティングの課題
2. Amazon GameLift の概観
3. GameLift Managed Hosting の仕組み
4. GameLift Managed Hosting のリソース
5. GameLift ゲームサーバーのライフサイクル
6. 料金情報
7. まとめ

マルチプレイヤーゲームとは

他のプレイヤーと一緒に同じゲーム環境で対戦・協力して遊ぶゲームモードのこと

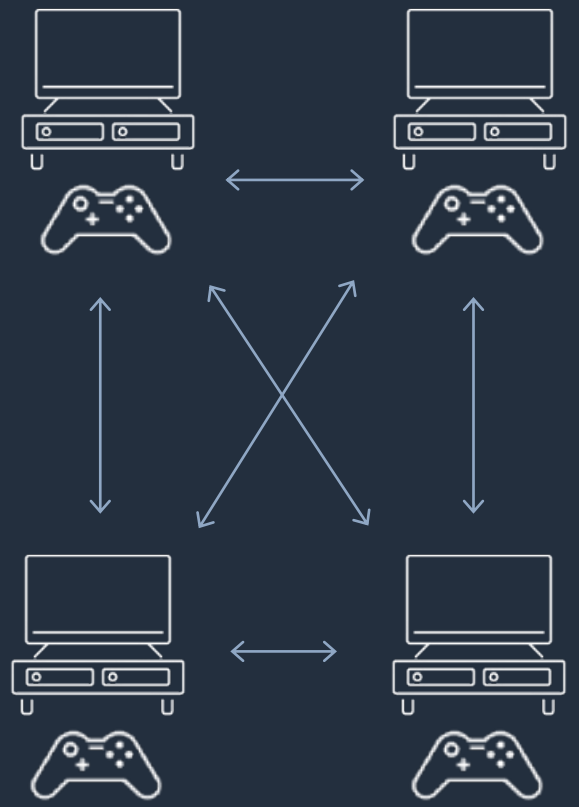
ローカル



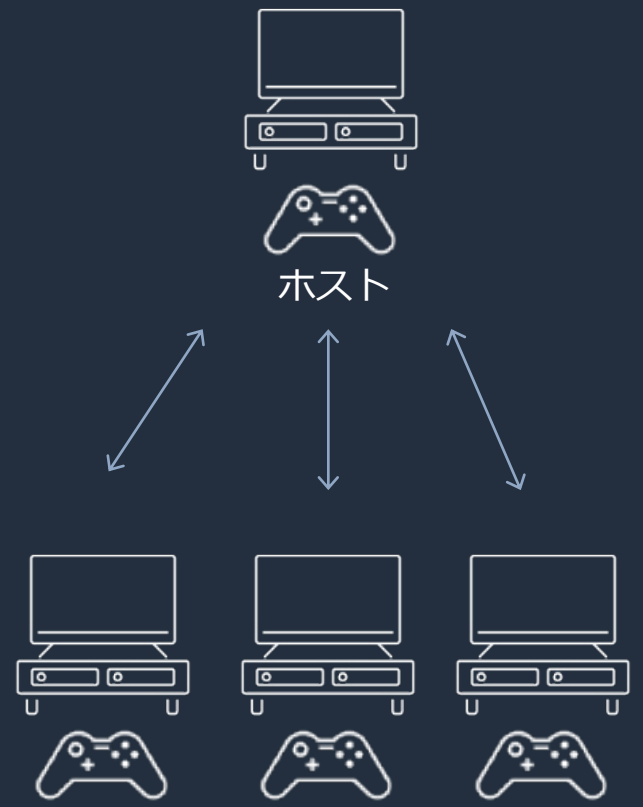
オンライン

オンラインマルチプレイヤーゲームのネットワーク構造

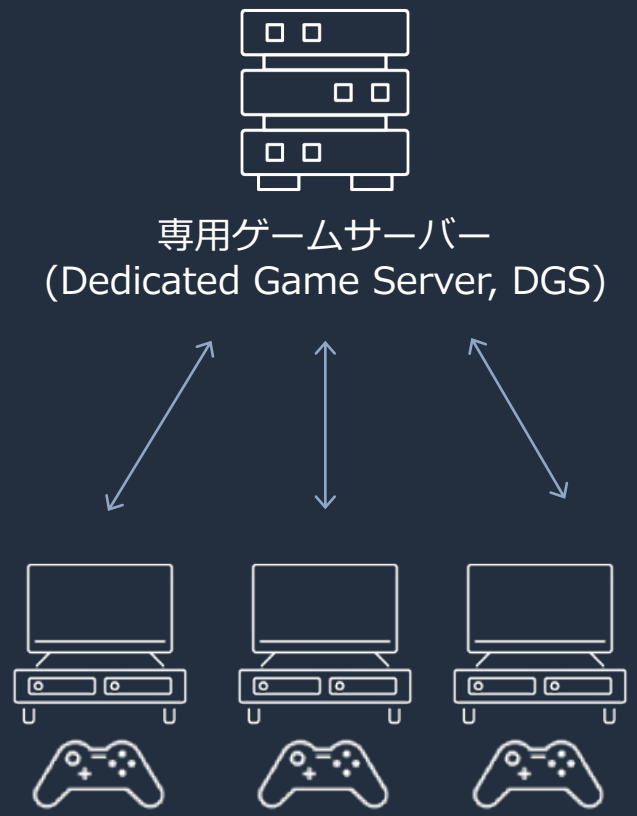
Peer to Peer (P2P)



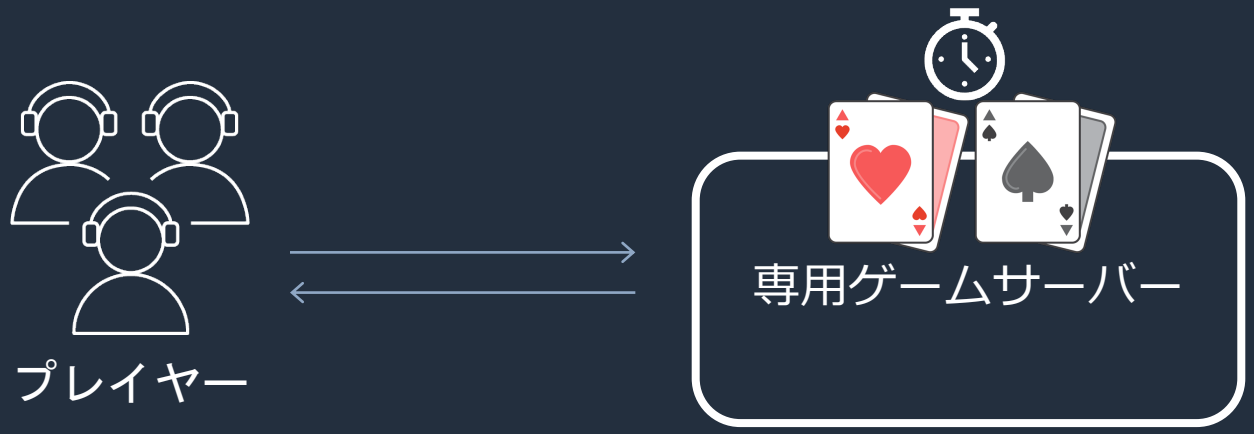
Listen Server



Dedicated Server

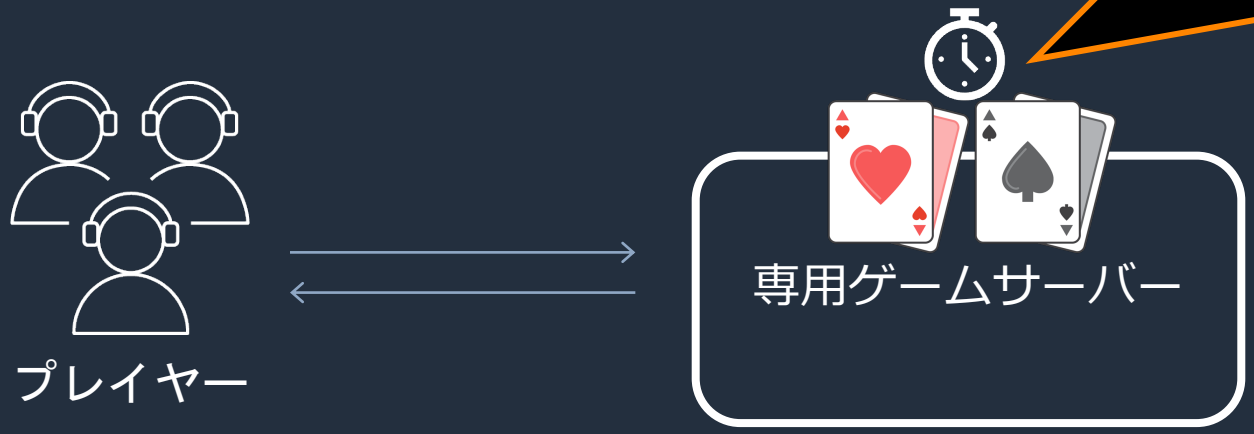


シンプルな例で考察してみる



シンプルな例で考察してみる

- サーバーでゲームの状況を保持（ステートフル）
  - ゲーム終了まで同じゲームサーバーに常時接続
  - プレイヤーの分散方法を考慮
- 応答速度が早いことが望ましい
  - 特にリアルタイムゲームではプレイヤー体験に直結

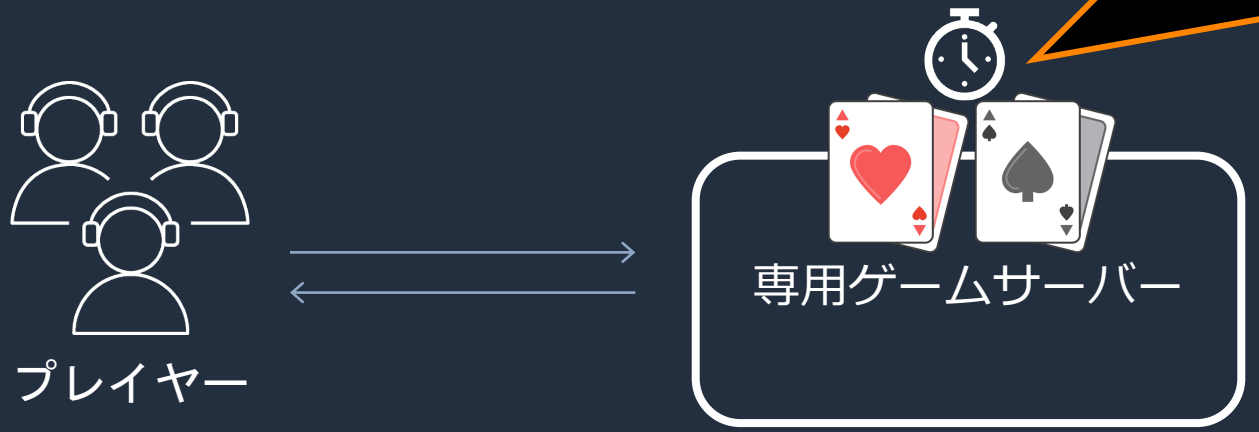


シンプルな例で考察してみる

- サーバーでゲームの状況を保持（ステートフル）
  - ゲーム終了まで同じゲームサーバーに常時接続
  - プレイヤーの分散方法を考慮
- 応答速度が早いことが望ましい
  - 特にリアルタイムゲームではプレイヤー体験に直結

シンプルな例で考察してみる

- サーバーでゲームの状況を保持（ステートフル）
  - ゲーム終了まで同じゲームサーバーに常時接続
  - プレイヤーの分散方法を考慮
- 応答速度が早いことが望ましい
  - 特にリアルタイムゲームではプレイヤー体験に直結

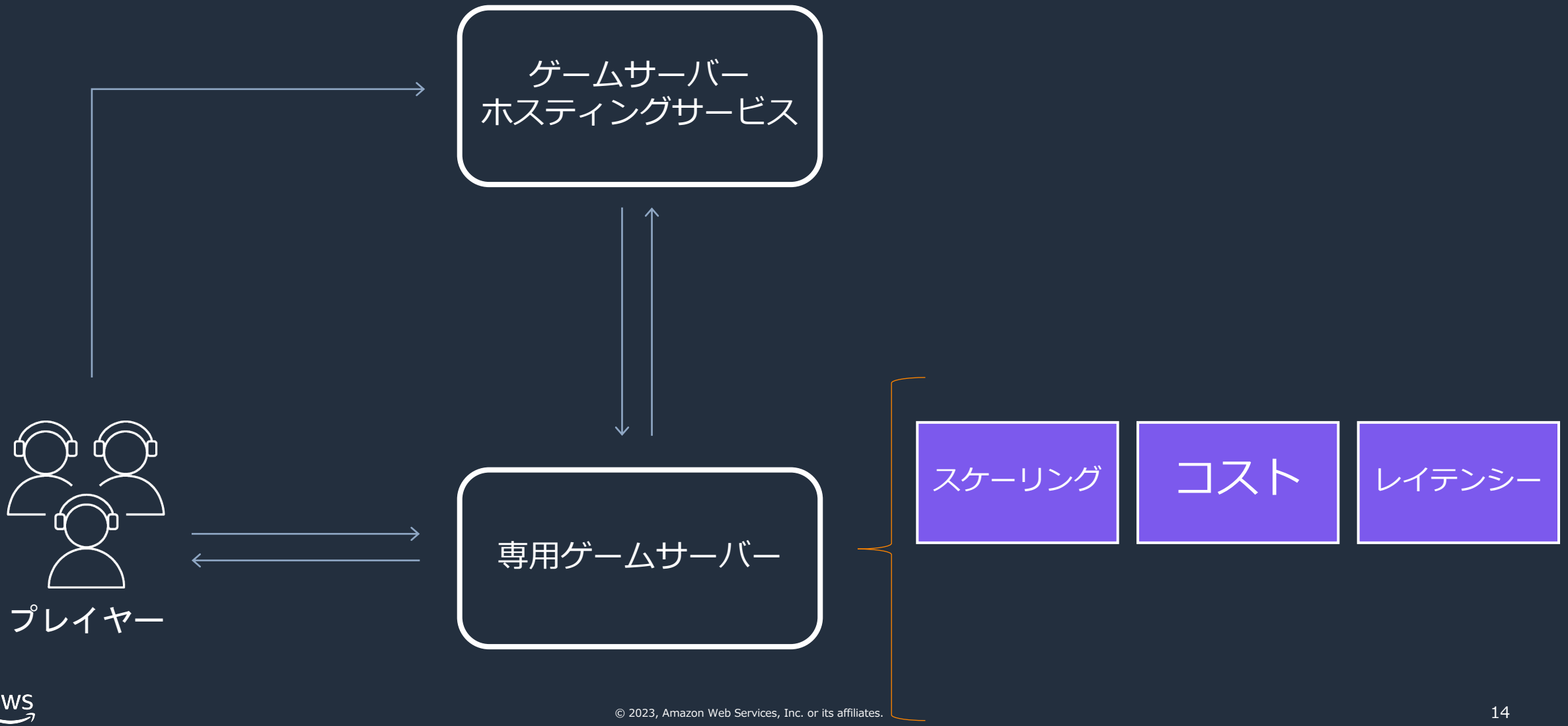


スケーリング

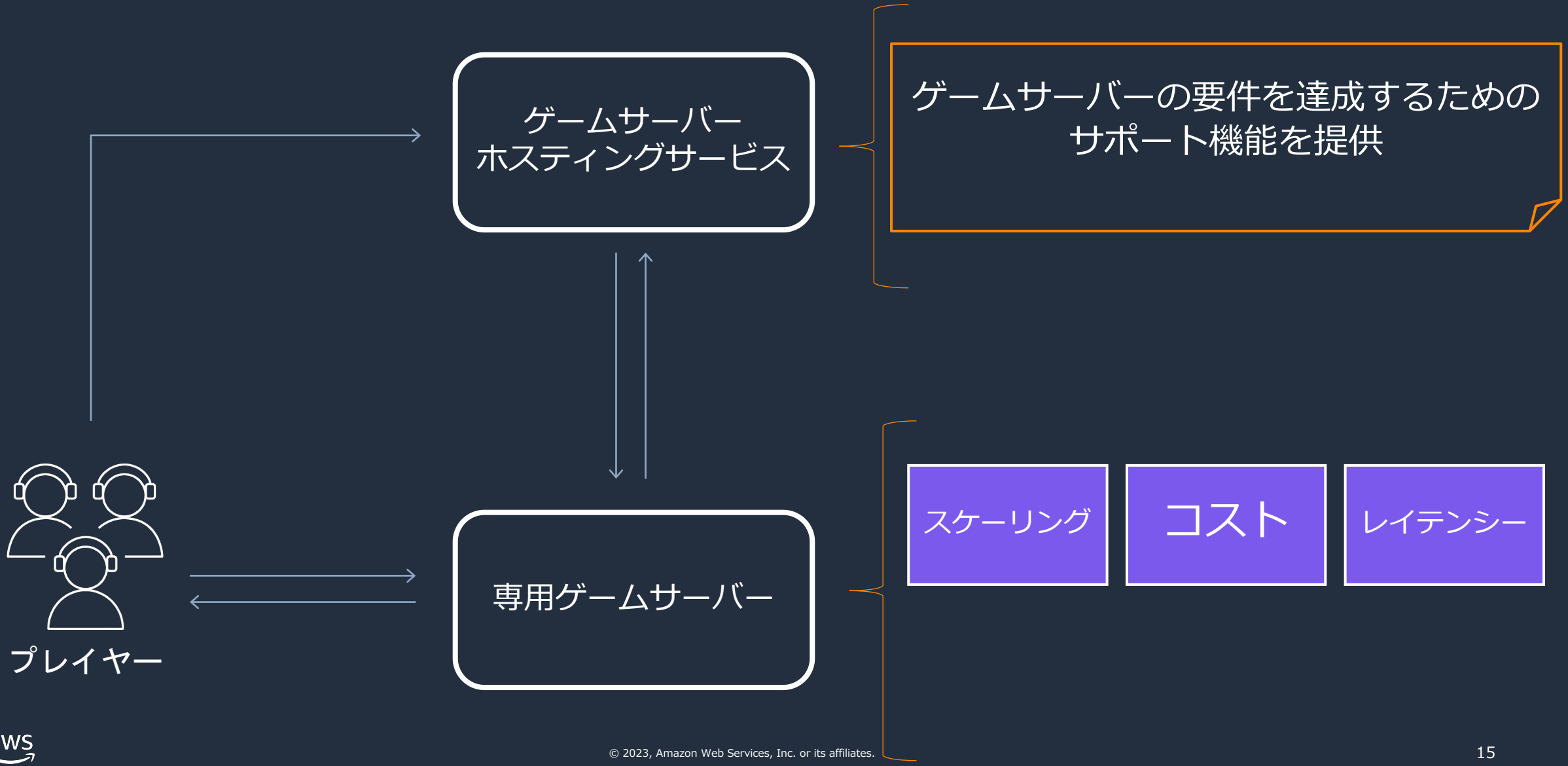
コスト

レイテンシー

ゲームサーバーをホストするための主な機能



ゲームサーバーをホストするための主な機能



ゲームサーバーをホストするための主な機能

インフラストラクチャ管理

コンピューティングリソースを調達し、ゲームサーバーのロードと実行を管理する

ゲームセッション配置

利用可能なゲームサーバーを探し、プレイヤーをゲームセッションに追加する

セッション管理

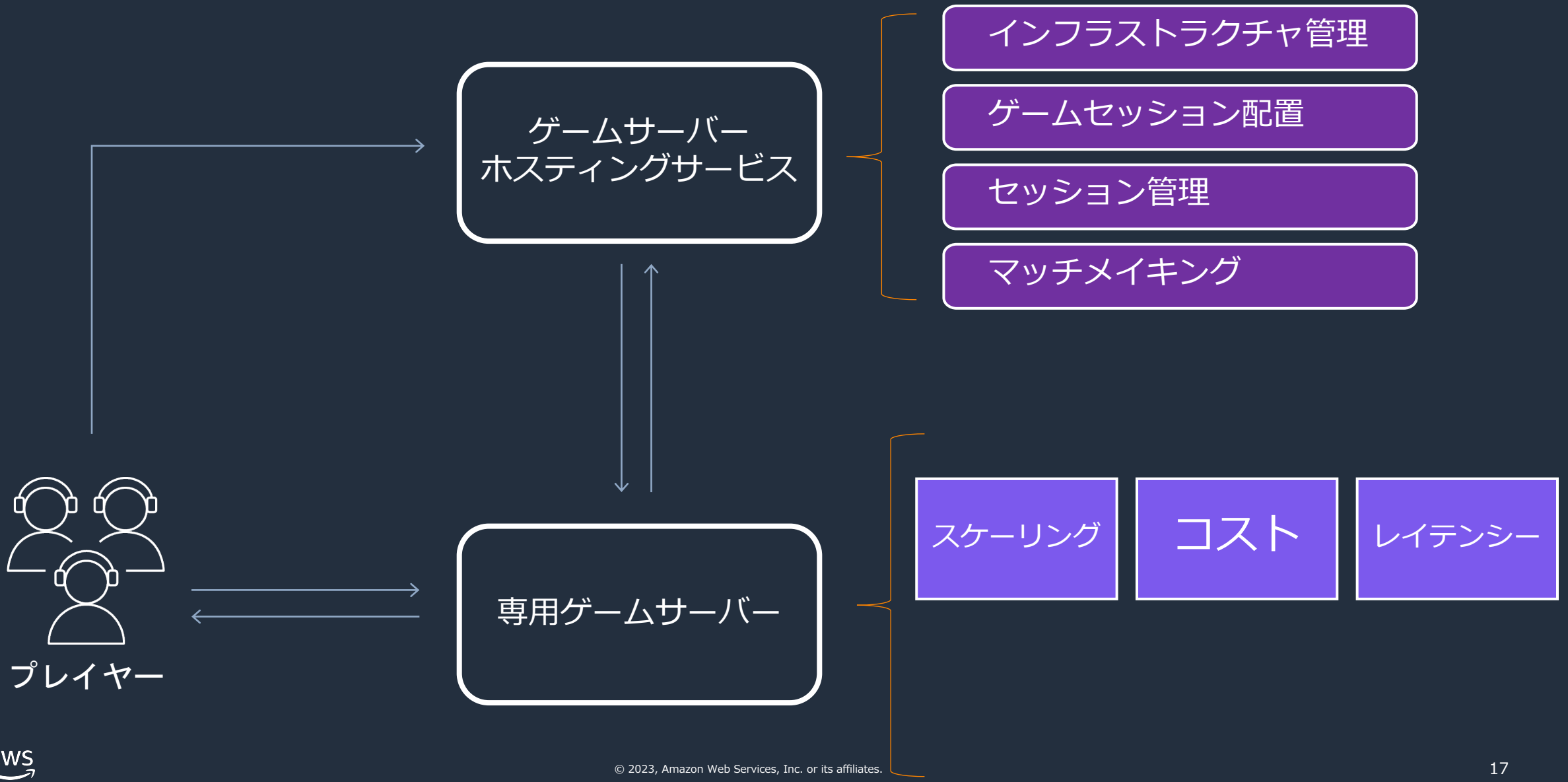
ゲームセッションやそこに接続しているプレイヤーの情報（プレイヤーセッション）を管理する

マッチメイキング

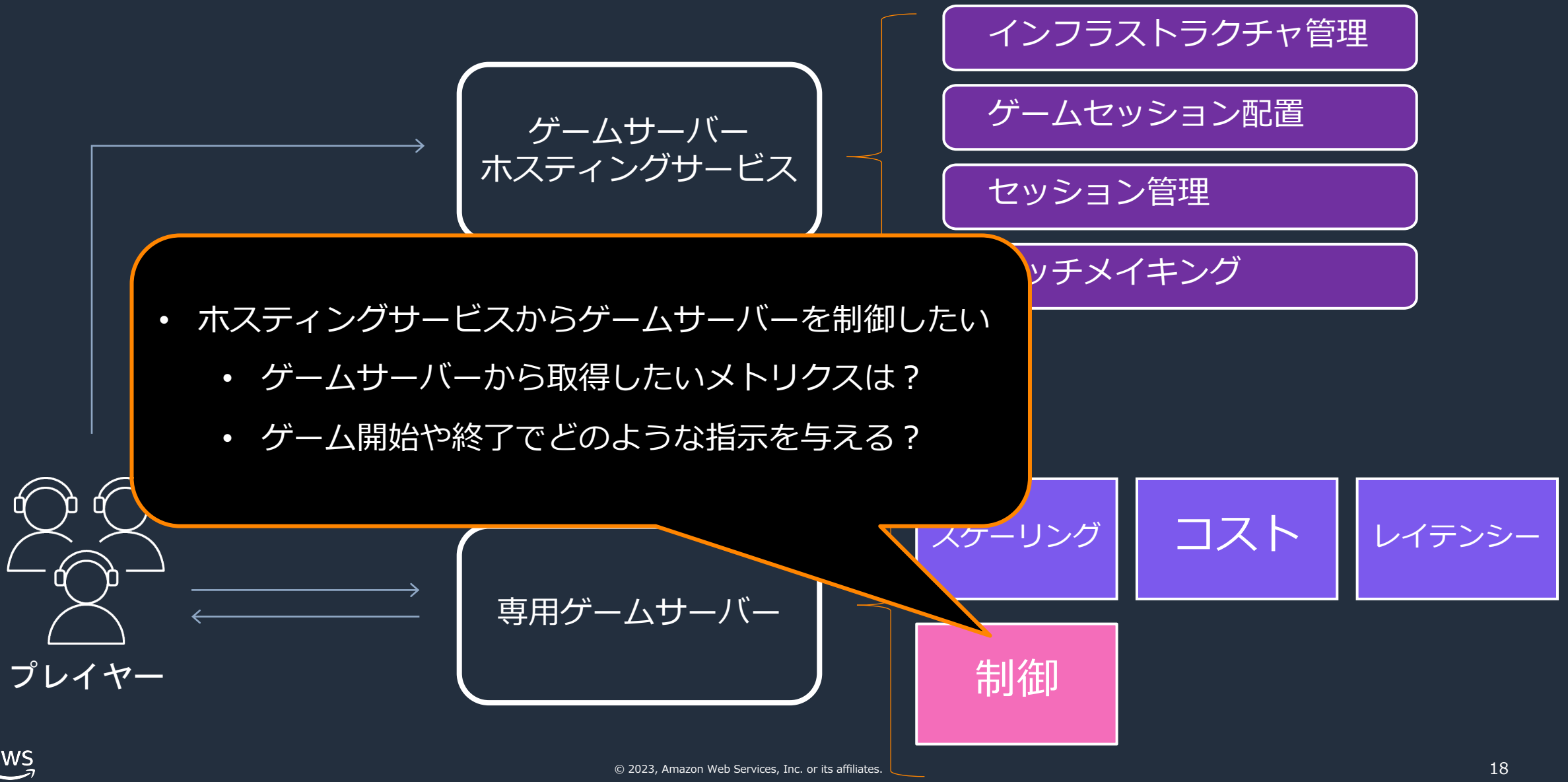
同じゲームセッションで同時にプレイするプレイヤーをグルーピングする

※ゲームセッション：プレイヤーが同時に接続して進行しているゲーム

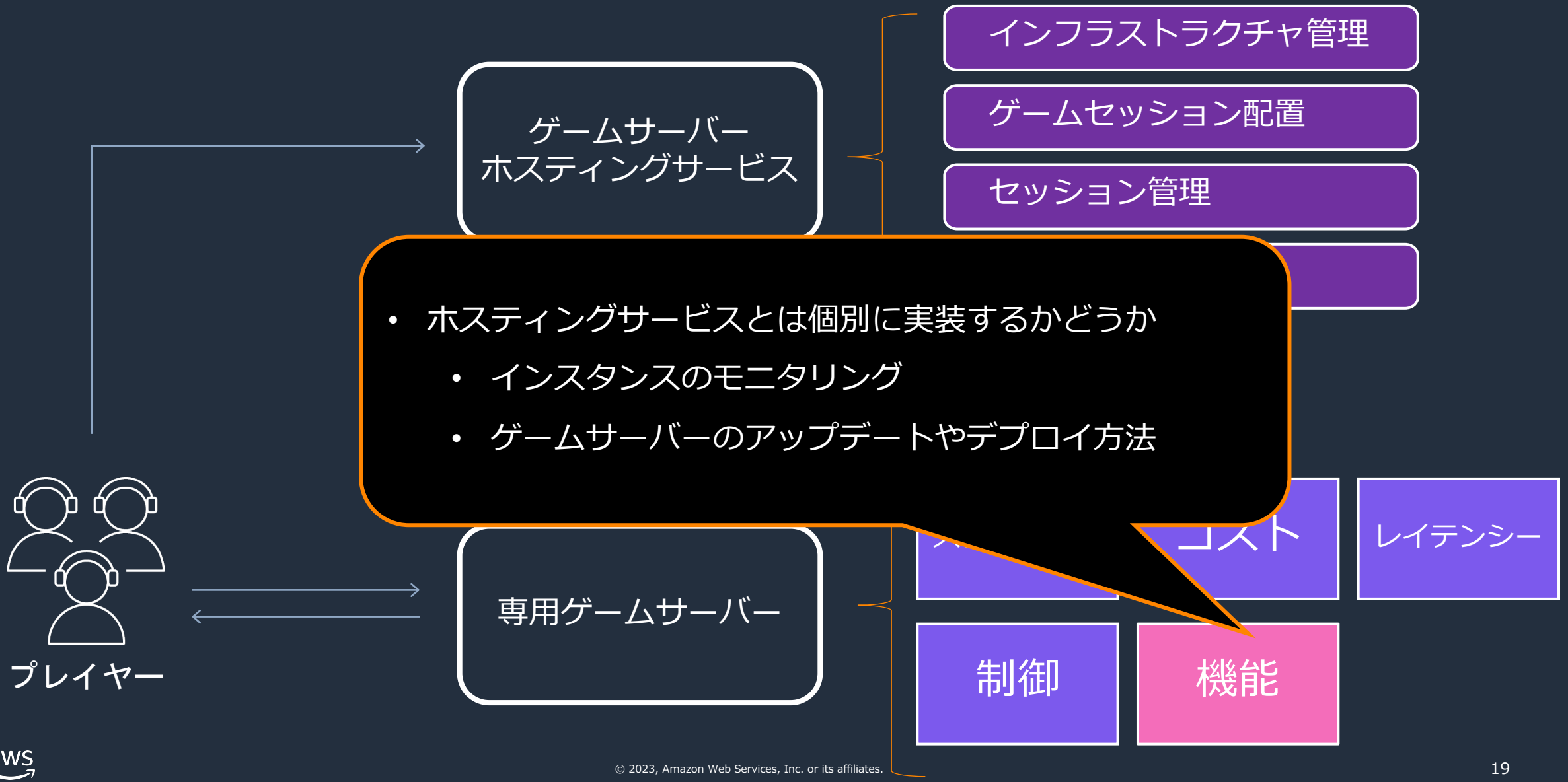
ゲームサーバーをホストするための主な機能



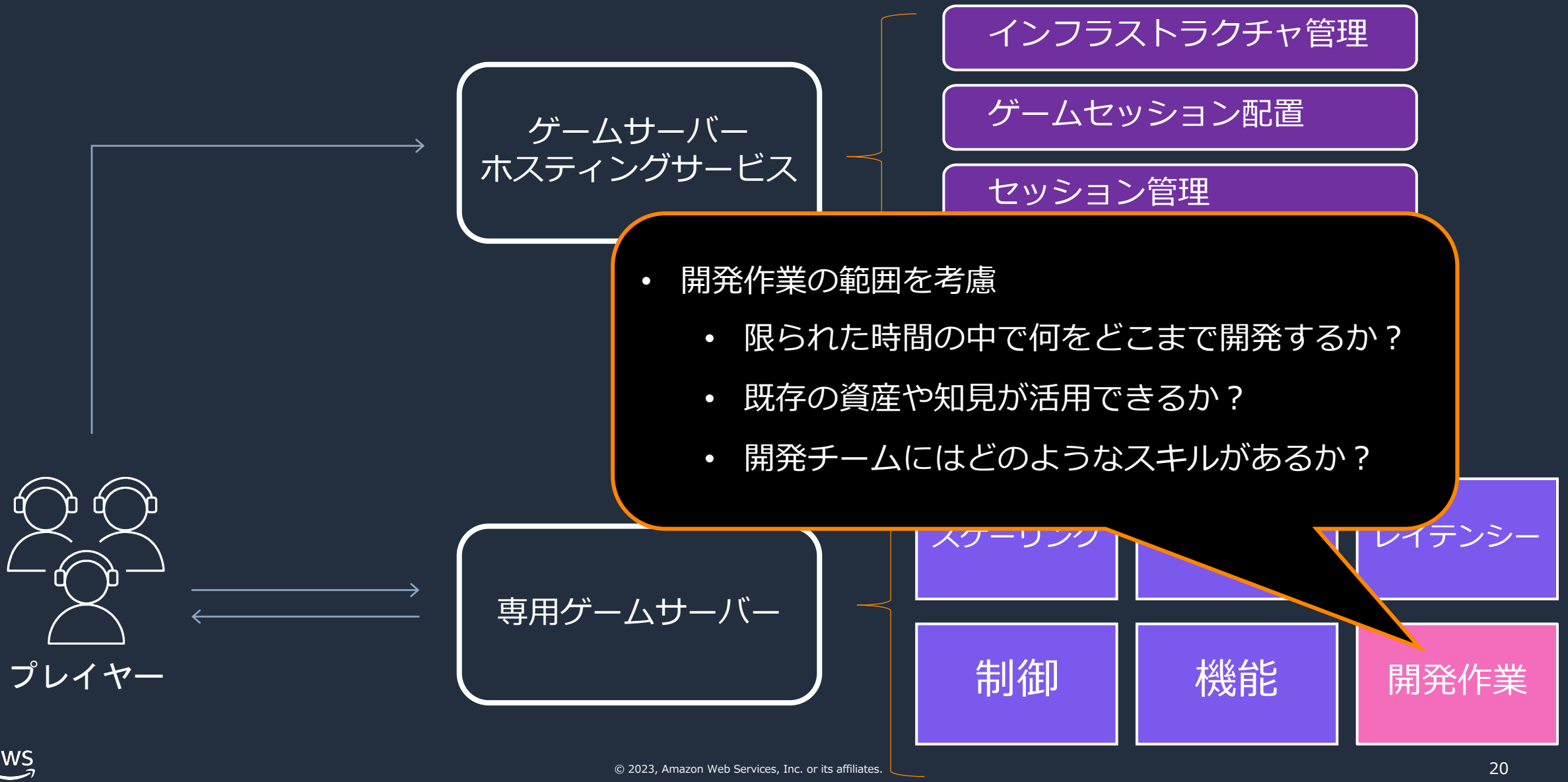
ゲームサーバーの設計時に考慮する必要のある要件



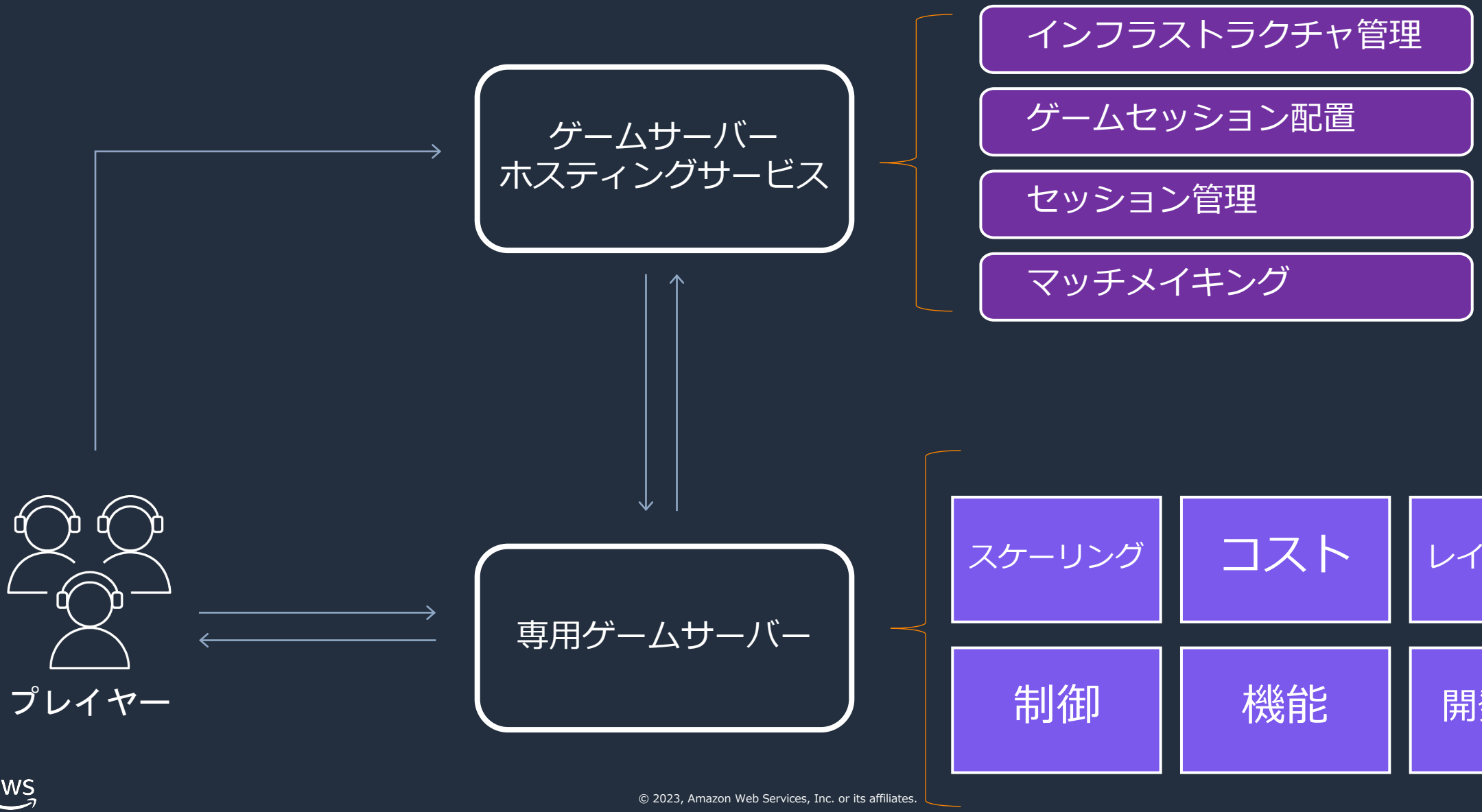
ゲームサーバーの設計時に考慮する必要のある要件



ゲームサーバーの設計時に考慮する必要のある要件



専用ゲームサーバーホスティングの課題



アジェンダ

1. マルチプレイヤーゲームにおける専用ゲームサーバーホスティングの課題
2. Amazon GameLift の概観
3. GameLift Managed Hosting の仕組み
4. GameLift Managed Hosting のリソース
5. GameLift ゲームサーバーのライフサイクル
6. 料金情報など
7. まとめ

Amazon GameLift とは

セッションベースのオンラインマルチプレイヤーゲームを提供する専用ゲームサーバーの
デプロイ・操作・スケーリングを制御するマネージドサービス

Amazon GameLift

グローバル展開と
世界中へのゲーム配置

高可用性

低レイテンシー

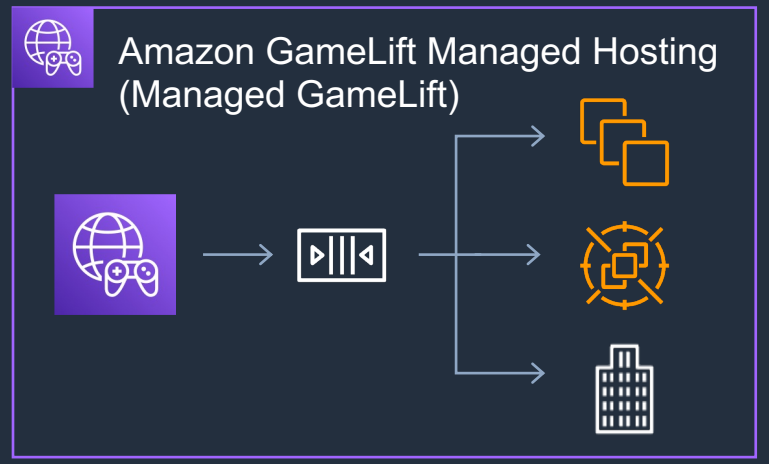
DDoS 攻撃からの保護

柔軟性

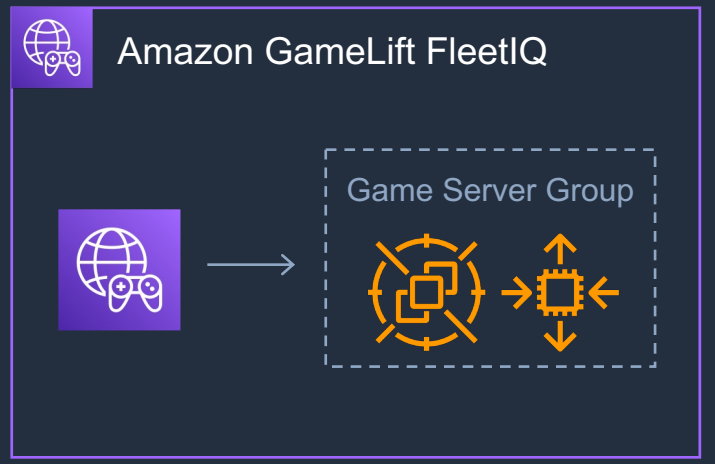
従量制料金

GameLift の概観

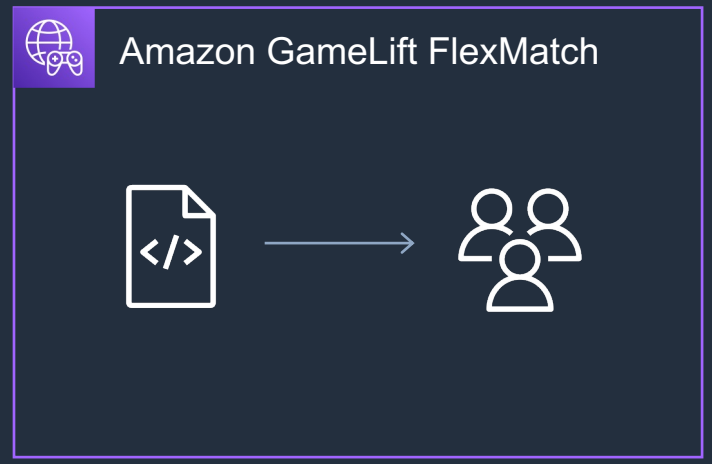
要件に応じて選択可能な複数のオプションを提供



- ゲームサーバーを実行するフリートをデプロイ・管理
- プレイヤーを最適なゲームセッションに配置
- ゲームサーバープロセスの他にゲームセッション及びプレイヤーセッションの情報を管理



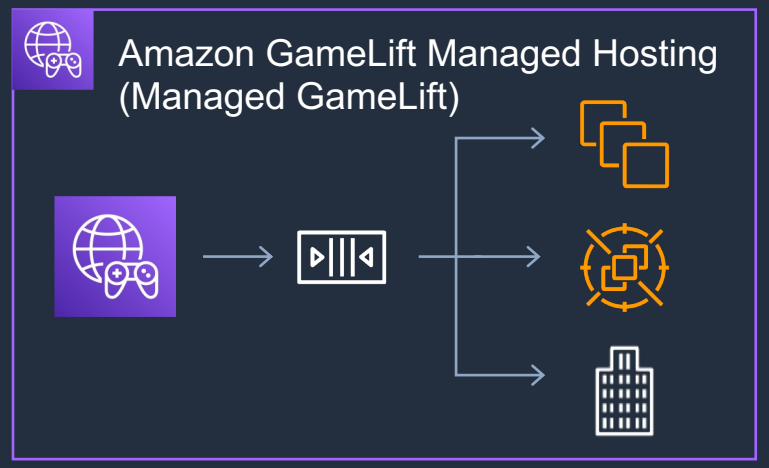
- ゲームサーバーのホスティング向けに EC2 スポットインスタンスの利用を最適化
- 中断率が低いスポットインスタンス上のゲームサーバーを検索しプレイヤーに提供
- ゲームサーバーの稼働率を追跡しインスタンス数を増減



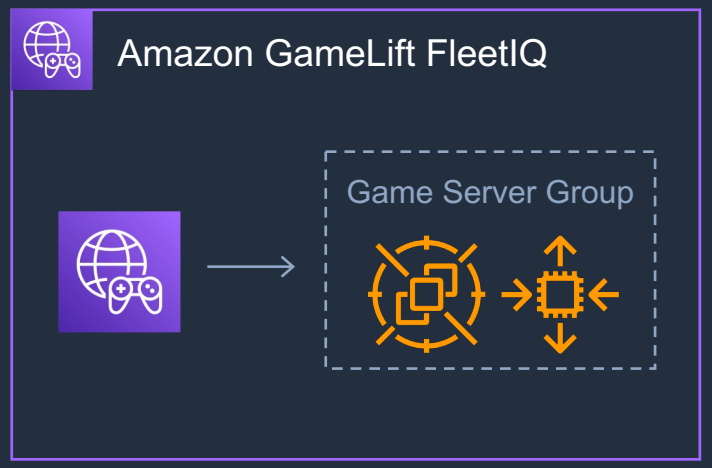
- マルチプレイヤーゲーム向けのカスタマイズ可能なマッチメイキングサービス
- ゲーム内のデータに基づいてマッチングアルゴリズムを調整

GameLift の概観

要件に応じて選択可能な複数のオプションを提供



- インフラストラクチャ管理
- ゲームセッション配置
- セッション管理
- マッチメイキング

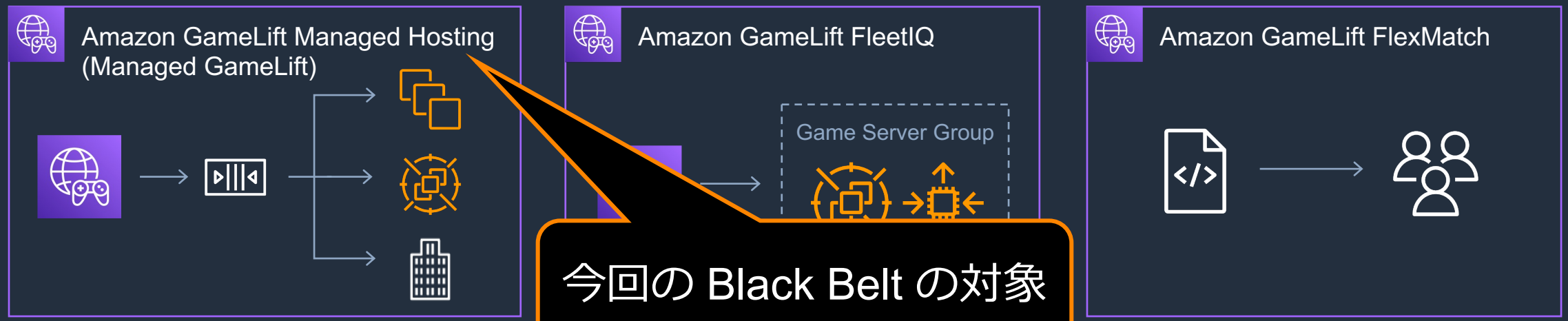


- インフラストラクチャ管理
- ゲームセッション配置
- セッション管理
- マッチメイキング

- インフラストラクチャ管理
- ゲームセッション配置
- セッション管理
- マッチメイキング

GameLift の概観

要件に応じて選択可能な複数のオプションを提供

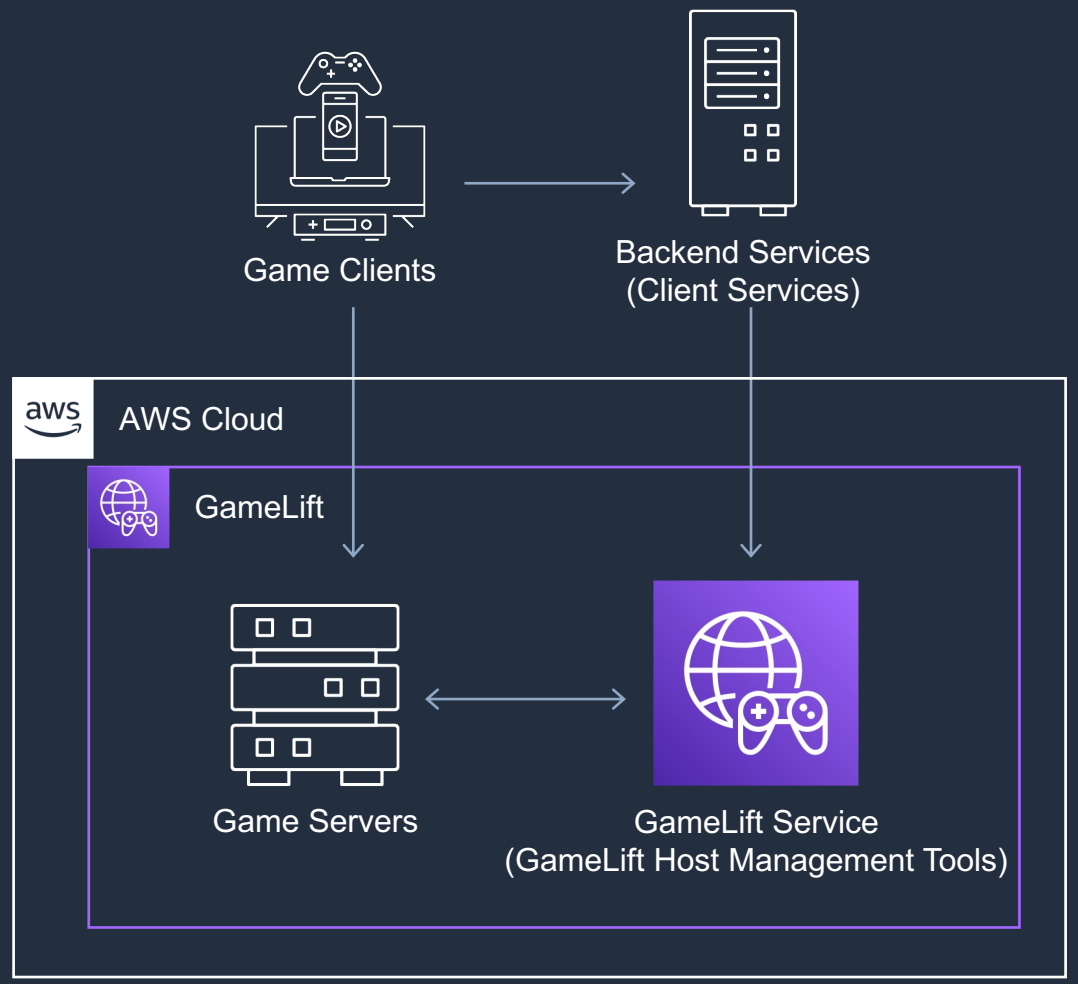


Amazon GameLift Managed Hosting (Managed GameLift)	Amazon GameLift FleetIQ	Amazon GameLift FlexMatch
インフラストラクチャ管理	インフラストラクチャ管理	インフラストラクチャ管理
ゲームセッション配置	ゲームセッション配置	ゲームセッション配置
セッション管理	セッション管理	セッション管理
マッチメイキング	マッチメイキング	マッチメイキング

アジェンダ

1. マルチプレイヤーゲームにおける専用ゲームサーバーホスティングの課題
2. Amazon GameLift の概観
3. GameLift Managed Hosting の仕組み
4. GameLift Managed Hosting のリソース
5. GameLift ゲームサーバーのライフサイクル
6. 料金情報
7. まとめ

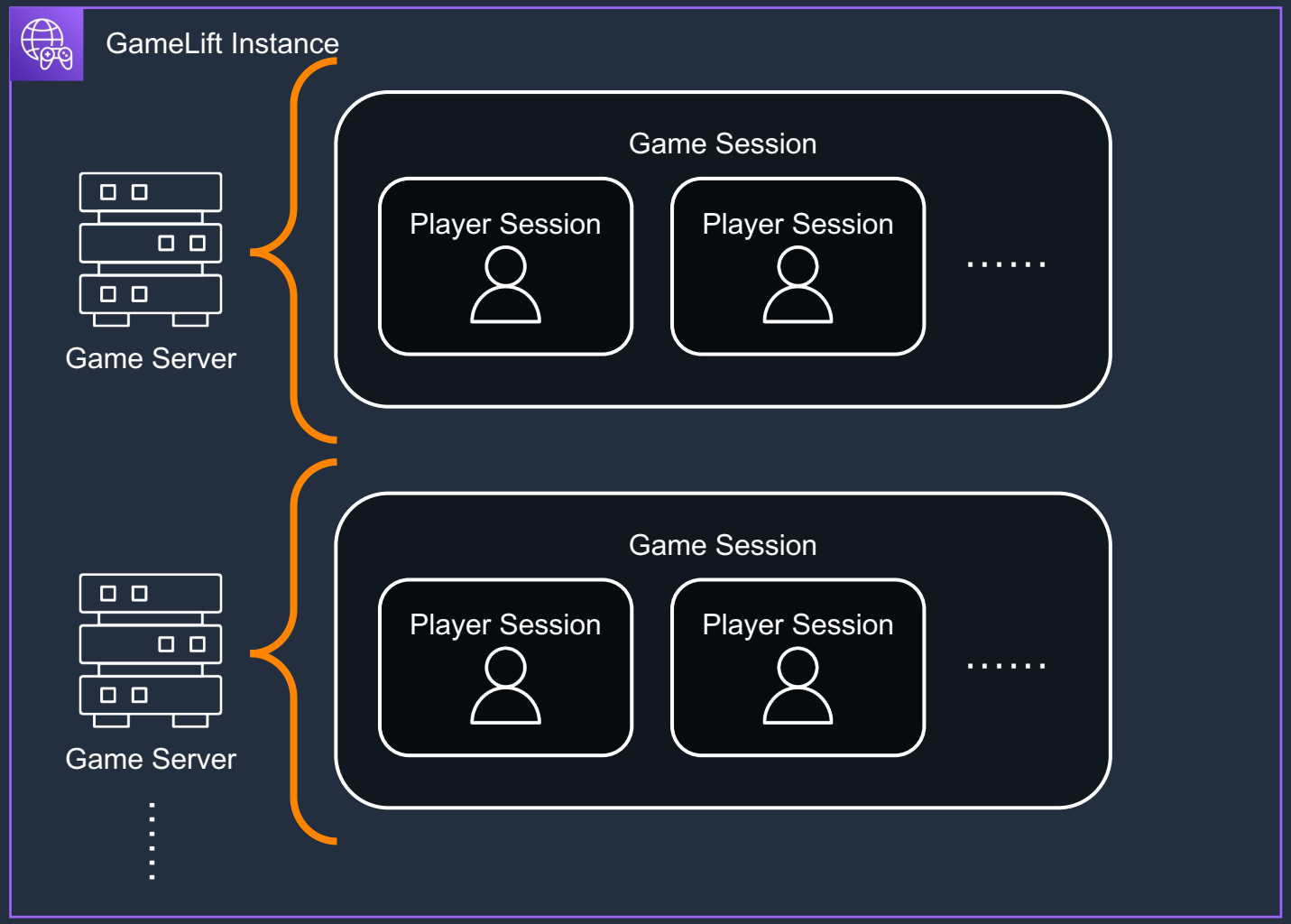
GameLift を使用するゲームのアーキテクチャ概観



用語	説明
ゲームクライアント	プレイヤーのデバイスで実行されるゲームのソフトウェア
バックエンドサービス（クライアントサービス）	AWS SDK 経由で GameLift API を呼び出すなど GameLift に関連する処理を行うサービス ゲームクライアントから GameLift サービスに API リクエストしたい際はバックエンドサービスを中継させる
GameLift サービス（GameLift ホスト管理ツール）	GameLift のリソースを管理するサービス リクエストされた GameLift API の処理や、ゲームサーバーと情報をやりとり
ゲームサーバー	GameLift で管理される EC2 インスタンスや自身のハードウェアなどのコンピューター（フリート）上で稼働するソフトウェア

ゲームセッションとプレイヤーセッションの管理

GAMELIFT ではゲームサーバープロセス、ゲームセッション、プレイヤーセッションを管理
プレイヤーはゲームサーバー上に作成されたゲームセッションにプレイヤーセッションを作成して接続



- リソースの関係
  - 1 インスタンスにつき
    最大 50 サーバープロセス稼働
  - 1 ゲームサーバープロセスにつき
    1 ゲームセッションが作成される
  - 1 ゲームセッションにつき
    最大 200 プレイヤーセッションまで作成可能

GameLift による専用ゲームサーバーのホスティング方法

ゲームサーバーのバイナリを実行するカスタムゲームサーバーと
軽量スクリプトを実行するリアルタイムサーバーの 2 種類がある

カスタムゲームサーバー

- Amazon GameLift Server SDK に対応する言語（C#, C++, Go）を使用して
ゲームサーバーのビルドバイナリを開発
- パッケージ化したビルドを GameLift に
アップロードしフリートをデプロイ
- 詳細なゲームロジックを備え
完全にカスタマイズさせた
本格的なゲームサーバーとして最適

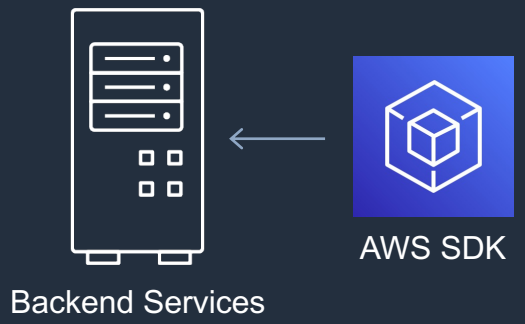
リアルタイムサーバー

- Node.js ベースの JavaScript で
ゲームサーバーのスクリプトを作成
- 作成したスクリプトをアップロードし、
フリートをデプロイ
- 複雑さを必要とせず
簡単で迅速にゲームを起動させたい
軽量のゲームサーバーとして最適

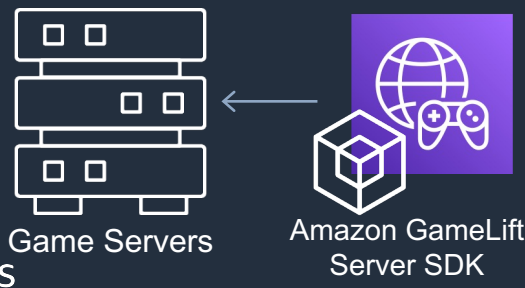
カスタムゲームサーバー – 統合方法

バックエンドサービスとゲームサーバーで専用の SDK を組み込む
ゲームクライアントは自前の方法でゲームサーバーに接続

- バックエンドサービスを経由して GameLift サービスにリクエスト
- バックエンドサービスから受け取った情報を使用して
ゲームサーバーに直接接続
- 通信ライブラリを導入しゲームサーバーとの通信ロジックを実装



- AWS SDK を組み込み GameLift API を呼び出す
- GameLift サービスから取得した情報を
ゲームクライアントに引き渡す



- Amazon GameLift Server SDK を組み込んだビルドを開発し
GameLift サービスとやりとり
- 通信ライブラリを導入しゲームクライアントとの通信ロジックを実装
- Linux/Windows のどちらかの OS を対象にビルドを出力

カスタムゲームサーバー – Amazon GameLift Server SDK

カスタムゲームサーバーに統合する SDK
GAMELIFT サービスとやりとりを行うために必要な関数群を提供

- C# （.NET 4.6, .NET 6）
- C++
- C++ for Unreal Engine
- Go

ゲームエンジンに SDK を統合して
カスタムゲームサーバーを開発

- Unity
- Unreal Engine
- Open 3D Engine (O3DE)

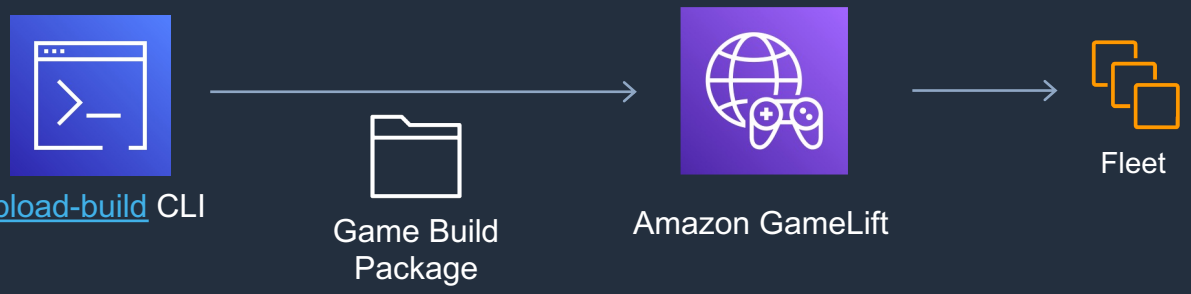
https://docs.aws.amazon.com/ja_jp/GameLift/latest/developerguide/GameLift-supported.html

カスタムゲームサーバー – フリートのデプロイ

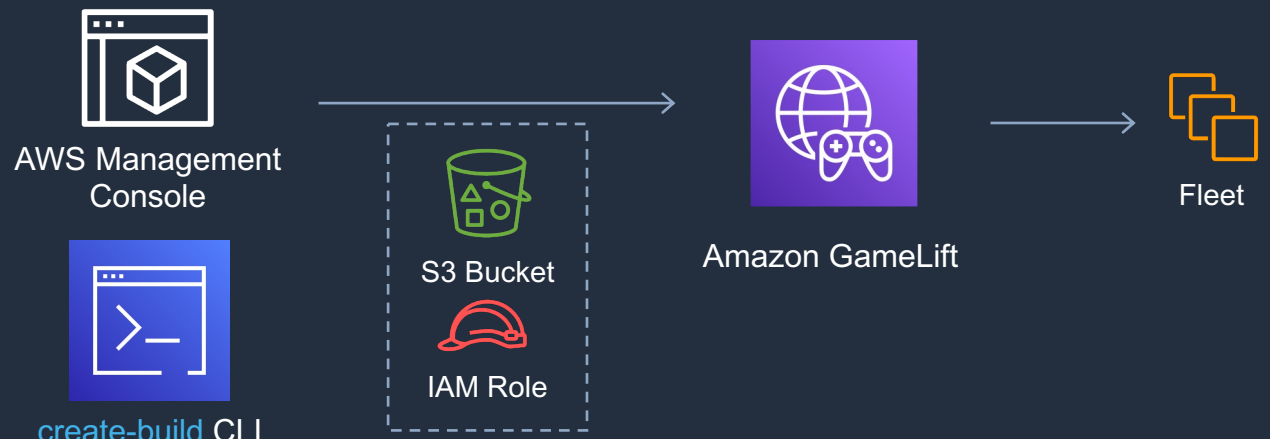
AWS CLI でローカルからビルドをアップロード、または AMAZON S3 バケット内のビルドを指定
ビルドを更新する場合も新規にフリートをデプロイする必要がある

新規 / 更新：ローカルからアップロード

推奨

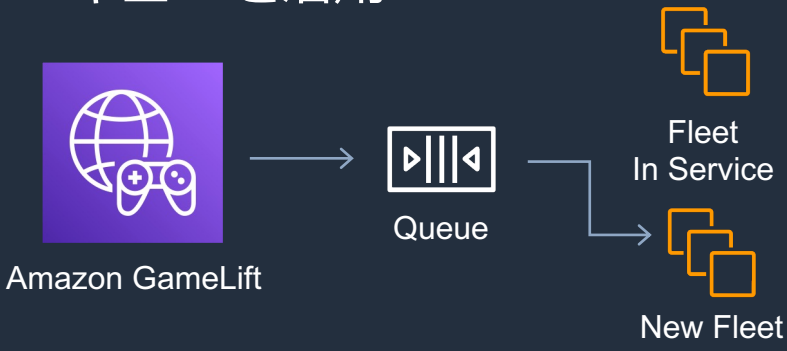


新規 / 更新：S3 バケット内のビルドと IAM ロールを指定

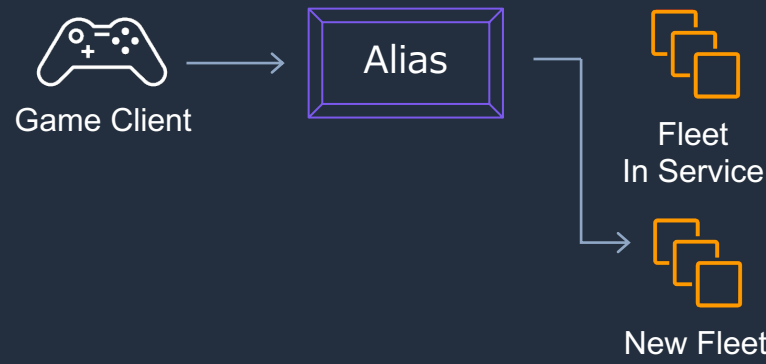


更新したビルドをデプロイする場合は

- キューを活用



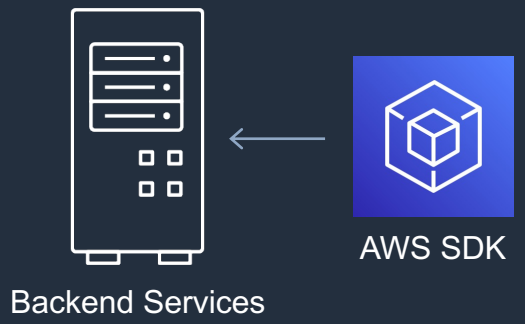
- エイリアスを活用



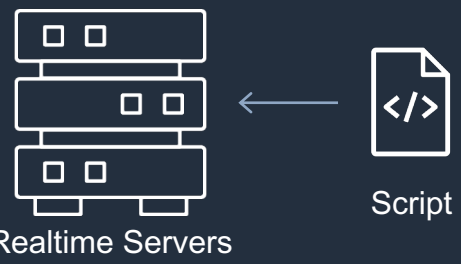
リアルタイムサーバー – 統合方法

ゲームクライアントとクライアントサービスで専用の SDK を組み込む
リアルタイムサーバーはスクリプトを開発

- バックエンドサービスを経由して GameLift サービスにリクエスト
- バックエンドサービスから受け取った情報を引数に、
  Realtime Client SDK(C#)の Connect() でリアルタイムサーバーに接続
- アクションやコールバックを使用してカスタムロジックを実装可能



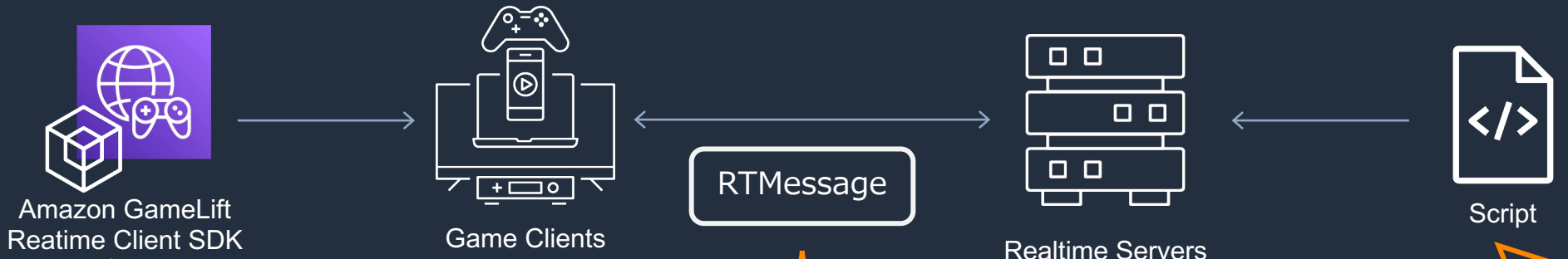
- AWS SDK を組み込み GameLift API を呼び出す
- GameLift サービスから取得した情報を
  ゲームクライアントに引き渡す



- Node.js ベースのスクリプトを作成し
  必要に応じて GameLift サービスやゲームクライアントとやりとり
- インターフェースやコールバックを使用して
  カスタムロジックを実装可能

リアルタイムサーバー – ネットワークスタック

ゲームクライアント/リアルタイムサーバー間でメッセージをやりとり
オペレーションコードとペイロードを使用してロジックを分岐処理



下記情報をメッセージに含める

- ゲームクライアント/リアルタイムサーバーが特定のイベントやアクションを識別するためのオペレーションコード
- オペレーションコードに関連する追加データ（ペイロード）
- 送信先のプレイヤーもしくはグループ
- 通信に使用するプロトコル

opCode

payload

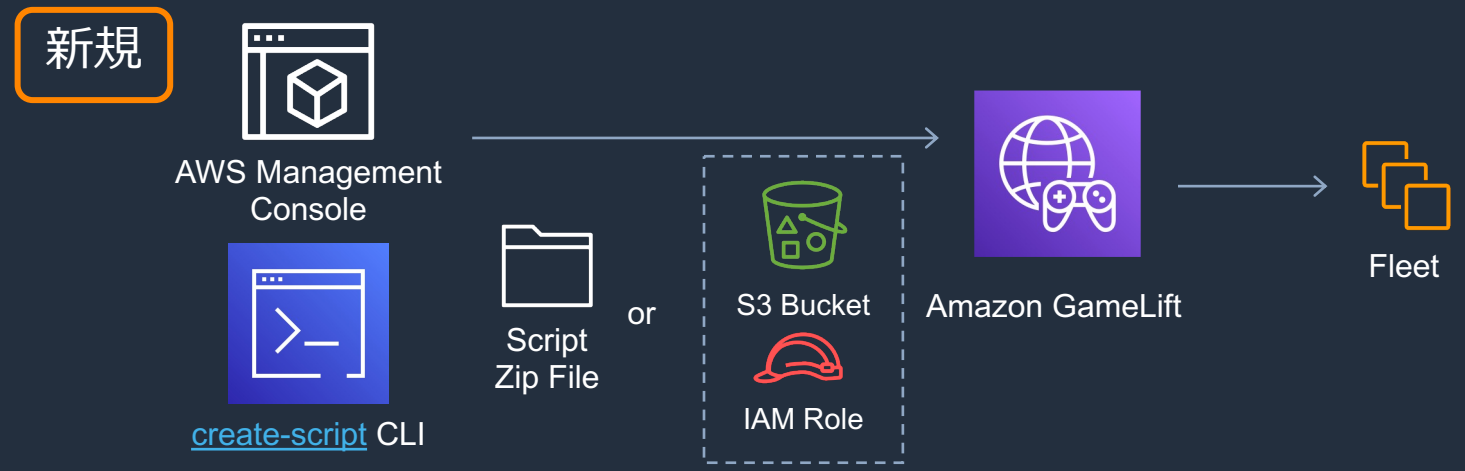
targetPlayer / targetGroup

deliveryIntent (TCP, UDP)

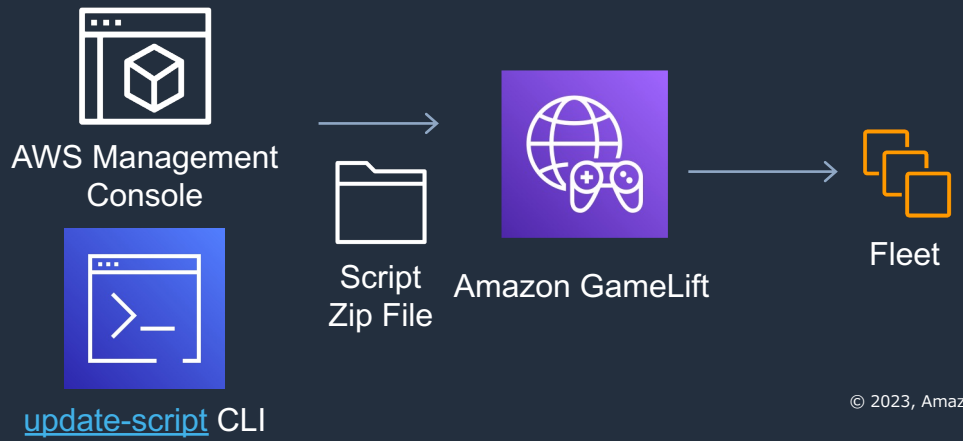
- 送信先のプレイヤーもしくはグループにメッセージを中継
- 必要に応じてコールバックでイベント駆動型のロジックを追加
  - 例：メッセージ受信時にオペレーションコードを識別して分岐処理

リアルタイムサーバー – フリートのデプロイ

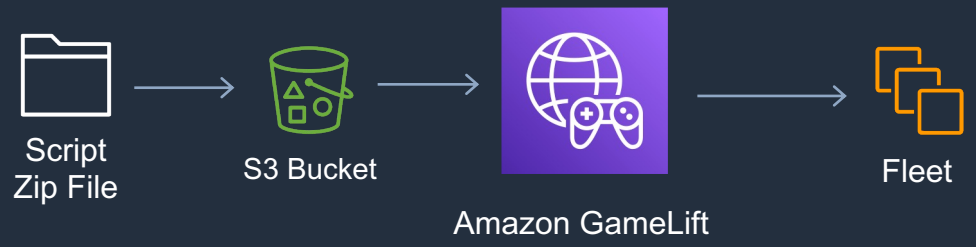
AWS CLI またはコンソール上でローカルまたは AMAZON S3 内のスクリプトを指定
スクリプトを更新する場合にフリートのデプロイは不要



更新：コンソールか CLI で指定



更新：S3 バケットにアップロード



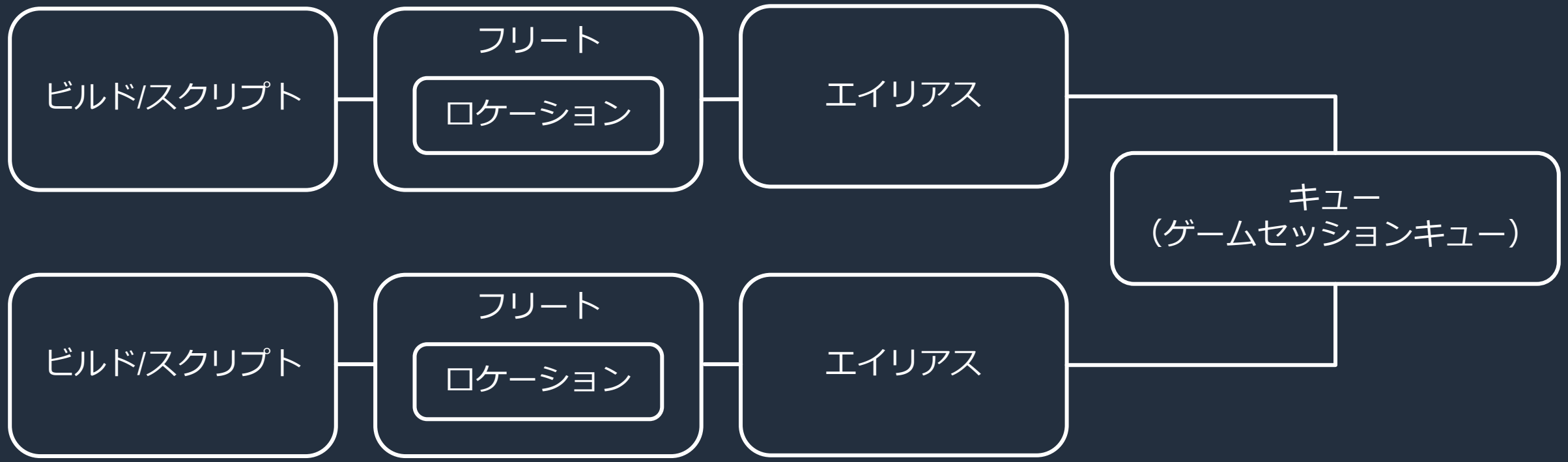
カスタムゲームサーバーとリアルタイムサーバーの主な違い

項目	カスタムゲームサーバー	リアルタイムサーバー
サポートされる言語	・ゲームクライアント：主要なゲームプラットフォームで対応する言語 ・ゲームサーバー：C#, C++, Go	・ゲームクライアント：C# ・リアルタイムサーバー：Node.js + JavaScript
OS	Linux, Windows	Linux
ゲームサーバーのロジック実装	・ゲームロジックを持たせたステートフルなゲームサーバーとして実装 ・Server SDK で GameLift サービスとの連携を処理	・ステートレスな中継サーバーとしても、ゲームロジックを持たせたステートフルなゲームサーバーとしても稼働可能 ・コールバックを用いたイベント駆動型の処理 ・オペレーションコードに基づく分岐処理
通信処理	・通信ライブラリやプロトコルを自前で選択して実装	・ゲームクライアント/リアルタイムサーバーでやりとりするためのネットワークスタックを提供 ・メッセージごとに TCP か UDP を選択
EC2 ポート設定（インバウンド通信に対するアクセス許可）	・プロトコルのタイプ、ポート範囲、IP アドレス範囲を指定	・GameLift サービスが自動で設定
ゲームサーバーの更新	・更新したビルドをアップロードし、フリートを新規にデプロイ	・更新したスクリプトで既存のスクリプトを置き換え ・GameLift サービスが既存のフリートに更新したスクリプトをデプロイ ・以降の新しいゲームセッションは更新されたスクリプトを使用（実行中のゲームセッションは使用しない）

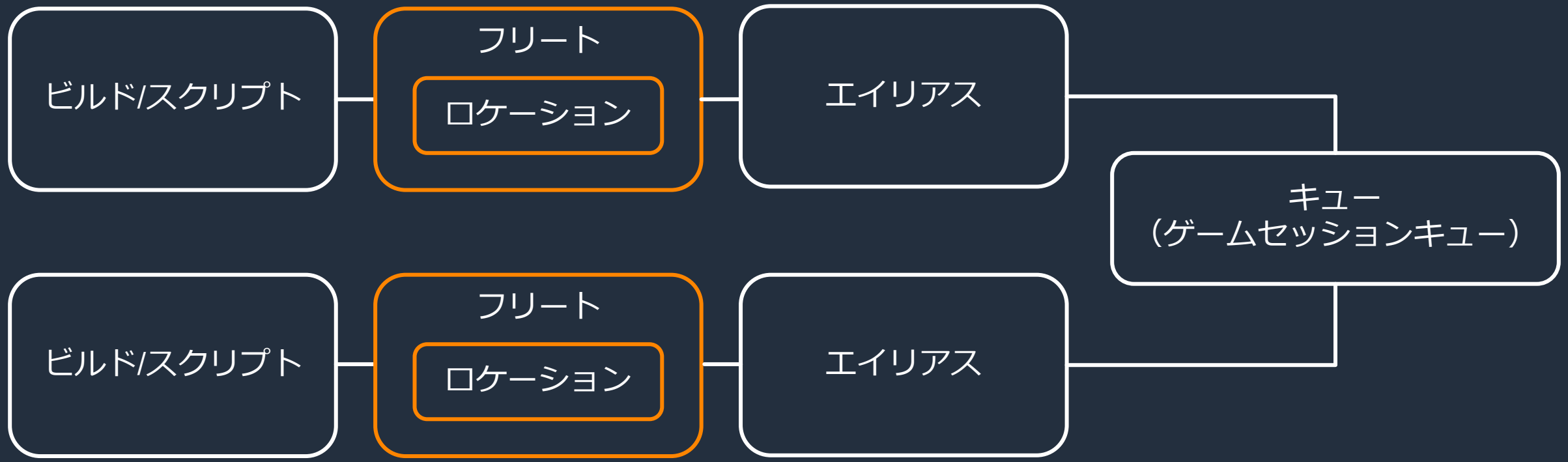
アジェンダ

1. マルチプレイヤーゲームにおける専用ゲームサーバーホスティングの課題
2. Amazon GameLift の概観
3. GameLift Managed Hosting の仕組み
4. GameLift Managed Hosting のリソース
5. GameLift ゲームサーバーのライフサイクル
6. 料金情報
7. まとめ

GameLift ホスティングリソース



GameLift ホスティングリソース



フリート

ゲームサーバーを実行してゲームセッションをホスティングするコンピューティングリソース
GAMELIFT が管理する EC2 インスタンスか自身のハードウェアを登録して作成

Compute type

Select a compute type to create a GameLift fleet.

Managed EC2

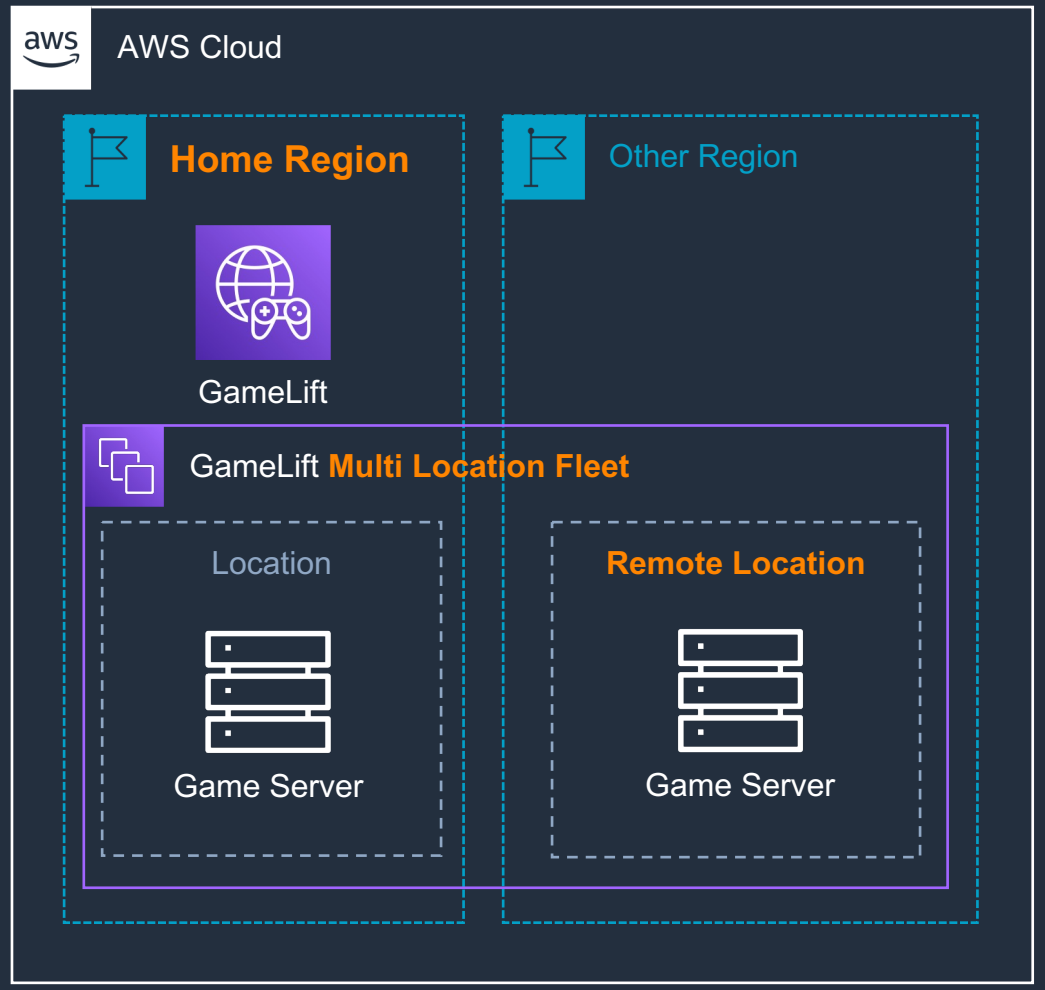
Run your game servers on AWS hardware managed by GameLift. GameLift manages your auto-scaling and process management.

Anywhere

Run your game servers on hardware you provide and manage. You can use on-premises capacity close to your players. You can also use Anywhere fleets for faster iteration and debugging on your local hardware.

フリート - ロケーション

フリートのデプロイ先となる場所
GAMELIFT は拠点となるホームリージョンから複数の地域（リモートロケーション）にデプロイ可能



用語	説明
ホームリージョン	GameLift を操作しているリージョン
リモートロケーション	ホームリージョンとは異なる場所でフリートをデプロイする場所 GameLift がサポートされている Local Zones も指定できる
マルチロケーションフリート	ホームリージョンに加えて、リモートロケーションにインスタンスをデプロイ ロケーションごとにキャパシティの管理、ゲームセッション配置が可能

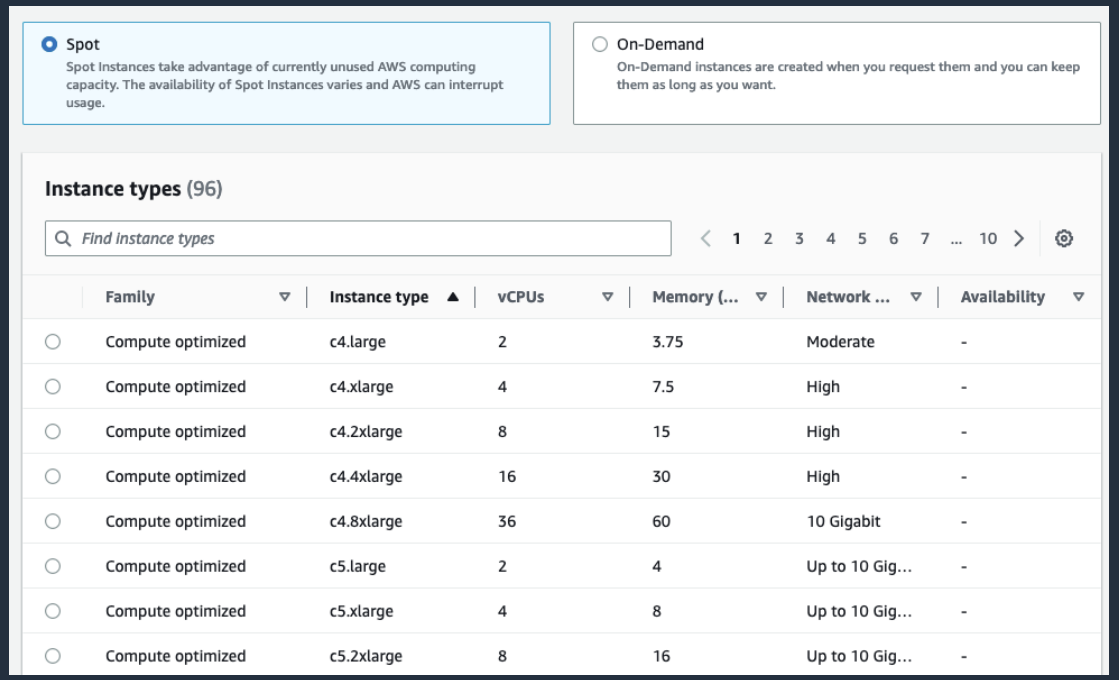
- 東京リージョンから GameLift を操作し
  フリートのデプロイ先に大阪リージョンも加える場合
  - ホームリージョン → 東京リージョン
  - リモートロケーション → 大阪リージョン

Managed EC2 フリート – 主な設計要素

GAMELIFT が管理する EC2 インスタンスでゲームサーバーを稼働させるフリート
カスタムゲームサーバーのビルドまたはリアルタイムサーバーのスクリプトをデプロイ

- リージョン
- ロケーション
- バイナリ型（ビルド, スクリプト）
- インスタンスの詳細
  - フリートタイプ（オンデマンド, スポット）
  - インスタンスタイプ
  - OS（Windows, Amazon Linux）
- フリートの詳細
  - TLS 証明書の生成有無
  - インスタンスロール
  - CloudWatch メトリクスグループ
- ランタイム
  - 起動パス、起動パラメータ、同時プロセス数
- ゲームセッション
  - 各インスタンスでの同時起動数・頻度
  - プレイヤーに対する作成制限
  - ゲームセッション保護ポリシーの有無
- （カスタムゲームサーバーの場合）EC2 ポート設定
- フリートのキャパシティ
  - 手動による調整
  - スケーリングポリシーによる AutoScaling（ターゲットベース, ルールベース）

Managed EC2 フリート – インスタンスの詳細



フリートタイプ

スポット, オンデマンド

OS

Linux, Windows

インスタンスタイプ

C6i instance　C6a instance　C5d instance

M5 instance　M5a instance

R5 instance　R5a instance

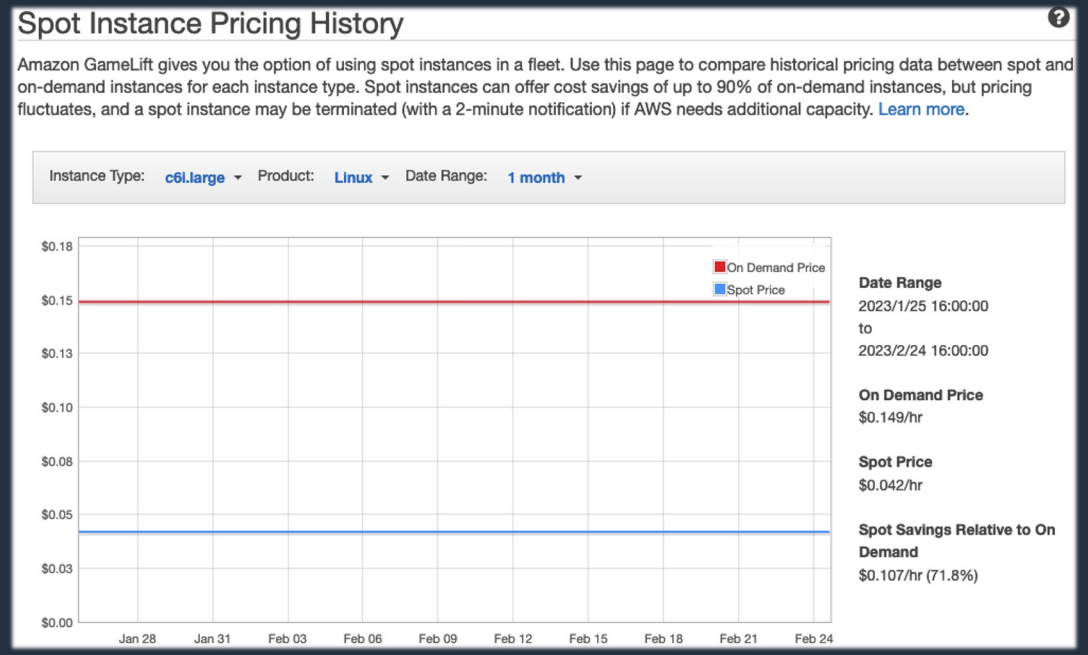
etc.

Amazon Elastic Block Store (Amazon EBS)

汎用 (SSD) ストレージ
50 GB

Managed EC2 フリート – スポットインスタンスの活用

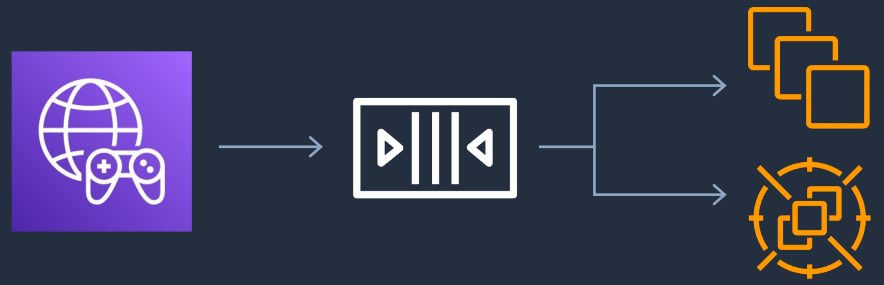
オンデマンドと同様のパフォーマンスで最大90%のコストを削減してゲームサーバーを稼動



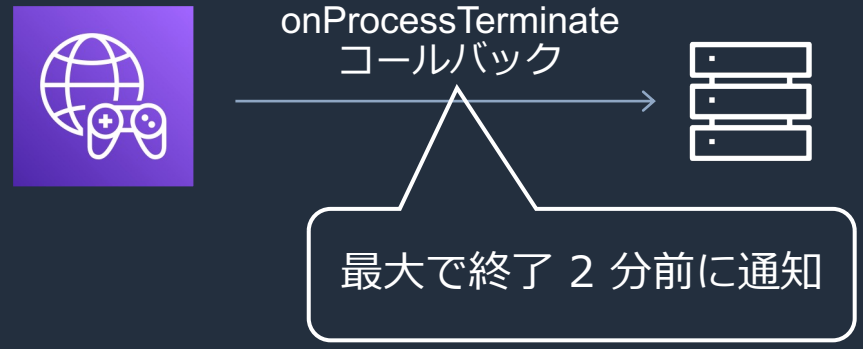
https://us-east-1.console.aws.amazon.com/GameLift/home?region=us-east-1#/spot-history

スポットインスタンスを上手に活用するには

- キューを使用してゲームセッションを配置

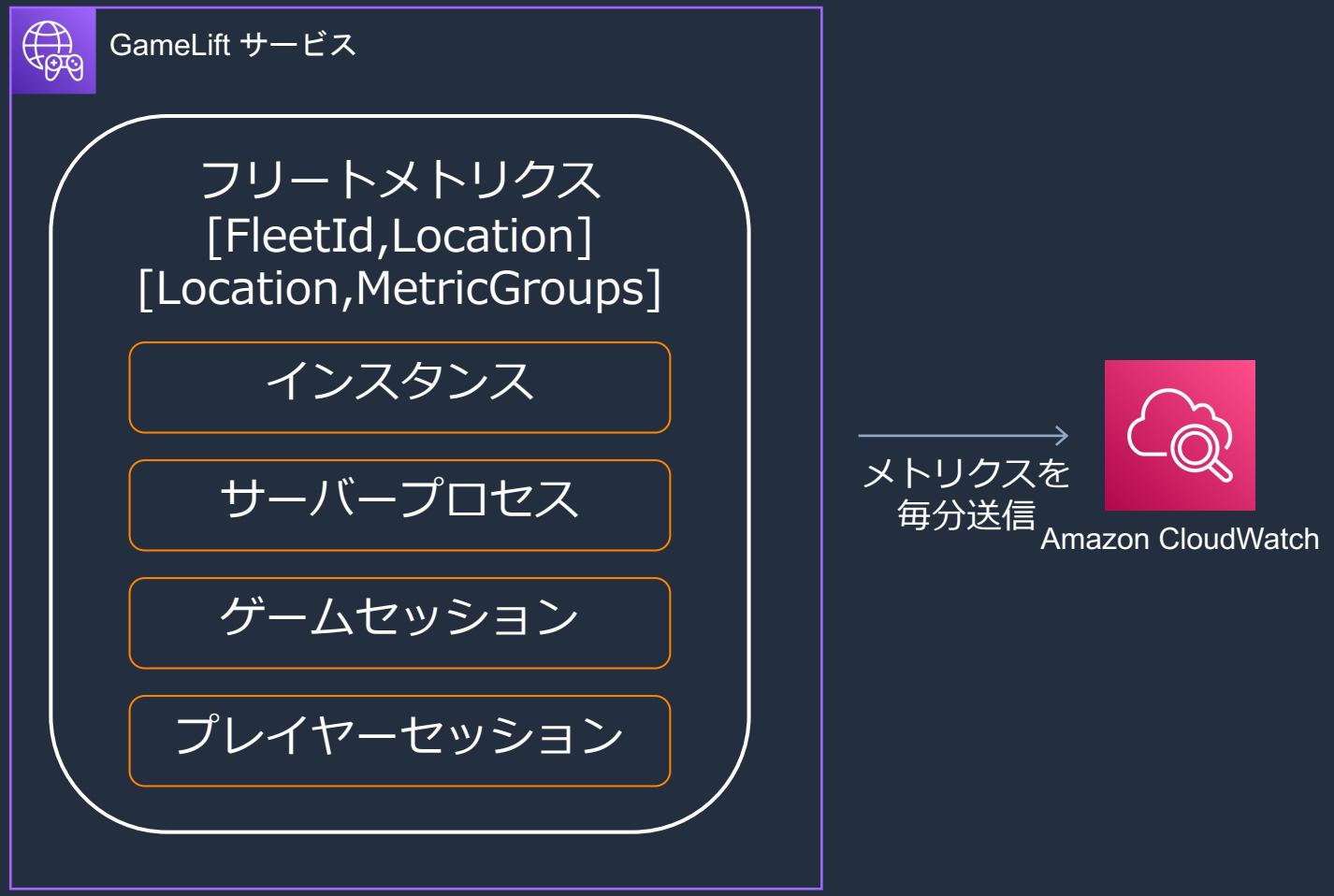


- ゲームサーバーで
  スポットの中断を処理できるようにする



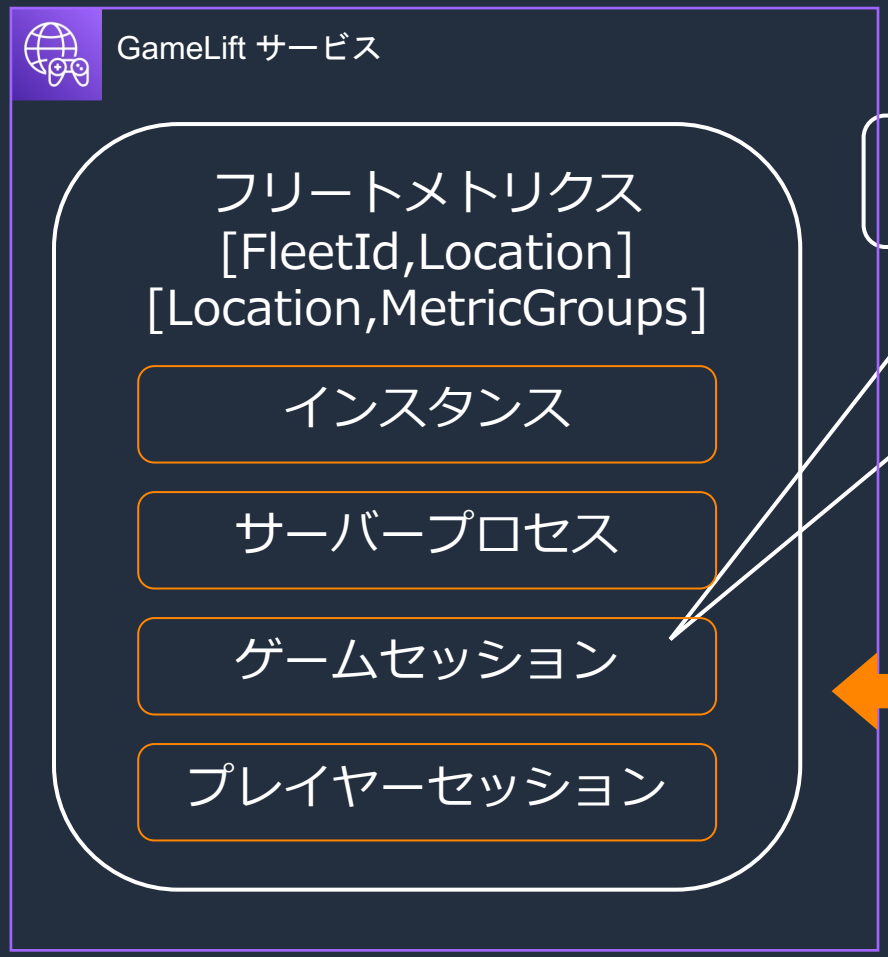
Managed EC2 フリート – CloudWatch メトリクス

フリート上で稼働しているリソースにおけるメトリクスをリアルタイムに追跡
メトリクスグループの指定で複数のフリートのメトリクスをフィルタして集計可能



Managed EC2 フリート – キャパシティのスケーリング

インスタンスをスケーリングしてコスト削減とプレイヤー体験をバランシング
手動で調整するかフリートメトリクスに基づく2種類の AUTO SCALING を使用



ターゲットベース

PercentAvailableGameSessions メトリクス

- フリートに維持させたいバッファのサイズを指定
  - 対象のフリートで使用可能なゲームセッションの割合を追跡
- 多くのゲームでシンプルで効果的に機能

メトリクスを毎分送信

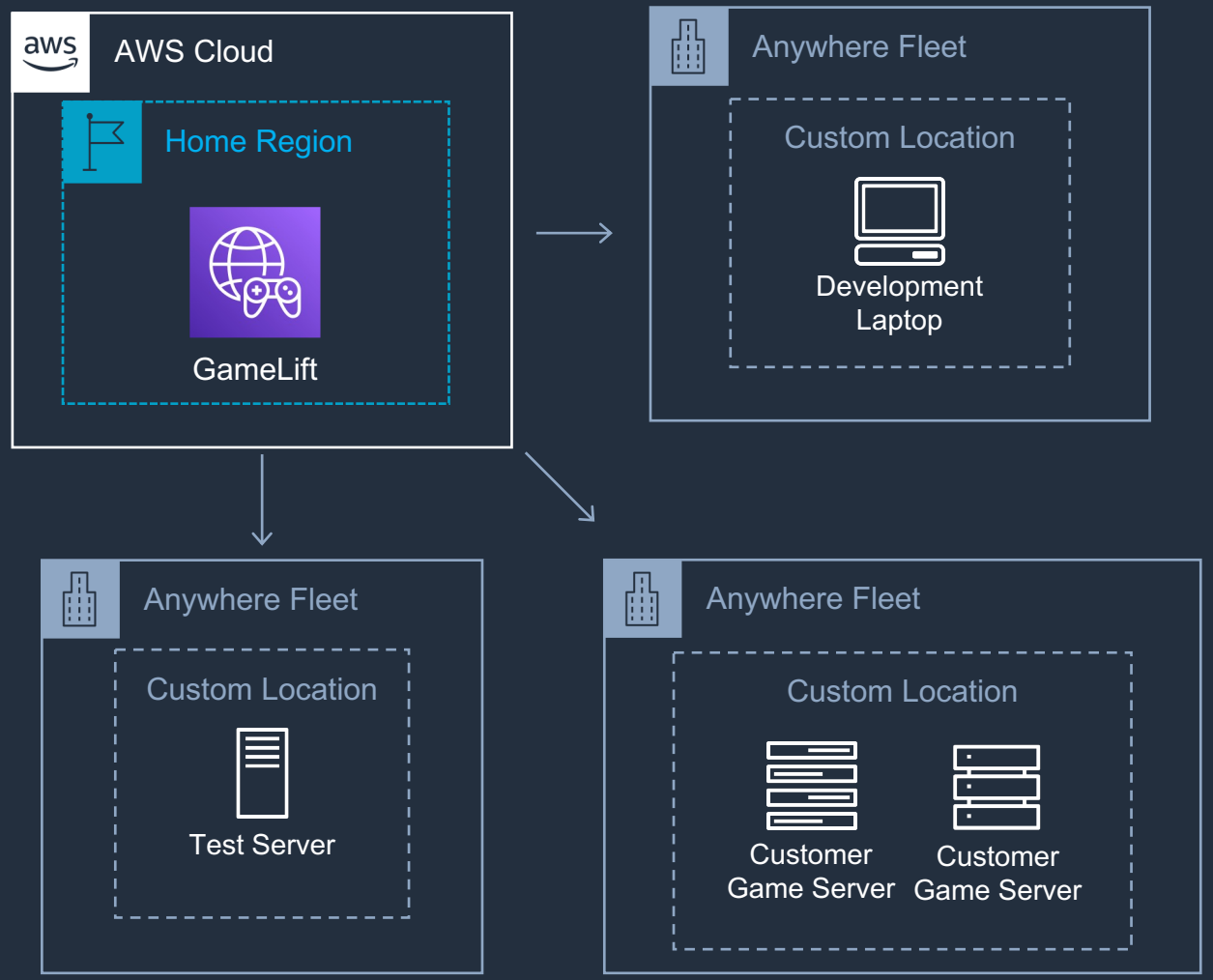
Amazon CloudWatch

ルールベース

- フリートメトリクスを用いた評価式を作成してきめ細かく制御
- 特別な状況に対処するためターゲットベースの補助として有効

Anywhere フリート

GAMELIFT に自身のハードウェア上のセッションを管理させるフリート
セッション管理を基盤となるコンピューティングリソースから切り離す



ゲーム開発の迅速化

- GameLift のゲームセッション配置や
  セッション管理の仕組みを活用しながら
  テストと反復処理を手元の開発機ですぐに実施

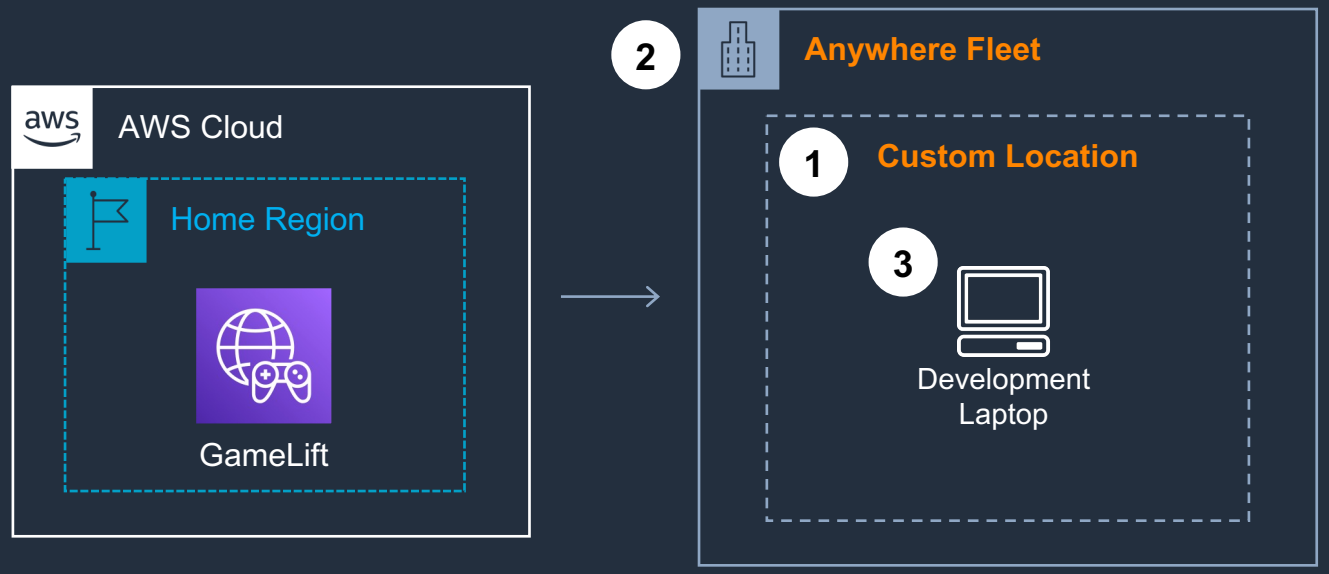
ハイブリッドのサーバー管理

- クラウドまたはオンプレミスでホストされている
  専用ゲームサーバーを 1 か所から
  全て運用・デプロイ・スケール

サーバー運用の合理化

- サーバーインフラストラクチャを
  単一のゲームサーバーオーケストレーション
  レイヤーに統合
- コストと運用の複雑さを軽減

Anywhere フリート – 作成手順

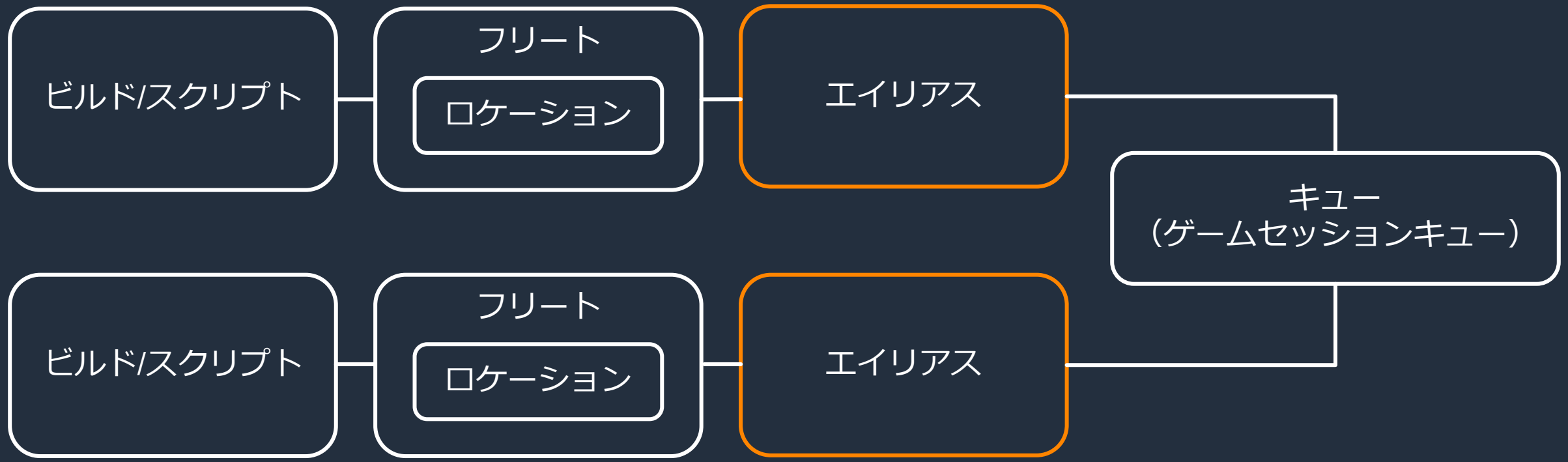


① カスタムロケーションを作成
- ゲームを実行するために使用するハードウェアの場所をラベリング
- 名前は custom- で始める必要あり

② カスタムロケーションを対象にAnywhere フリートを作成

③ Anywhere フリートを対象にGameLift に管理させるハードウェアを登録
- RegisterCompute (CLI / API)を使用
- IP アドレスや ComputeName を指定

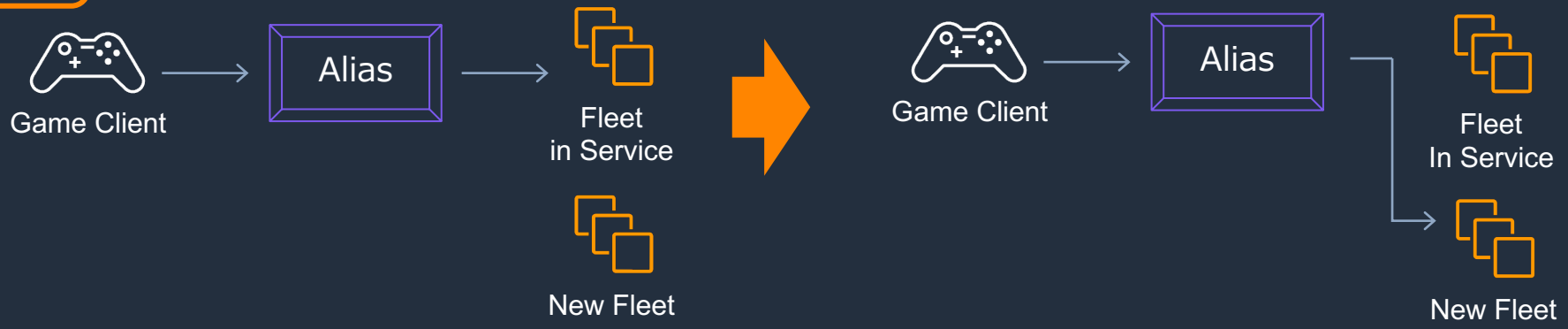
GameLift ホスティングリソース



エイリアス

フリートに割り当てる識別子
フリートを切り替える際のプレイヤートラフィックを制御する

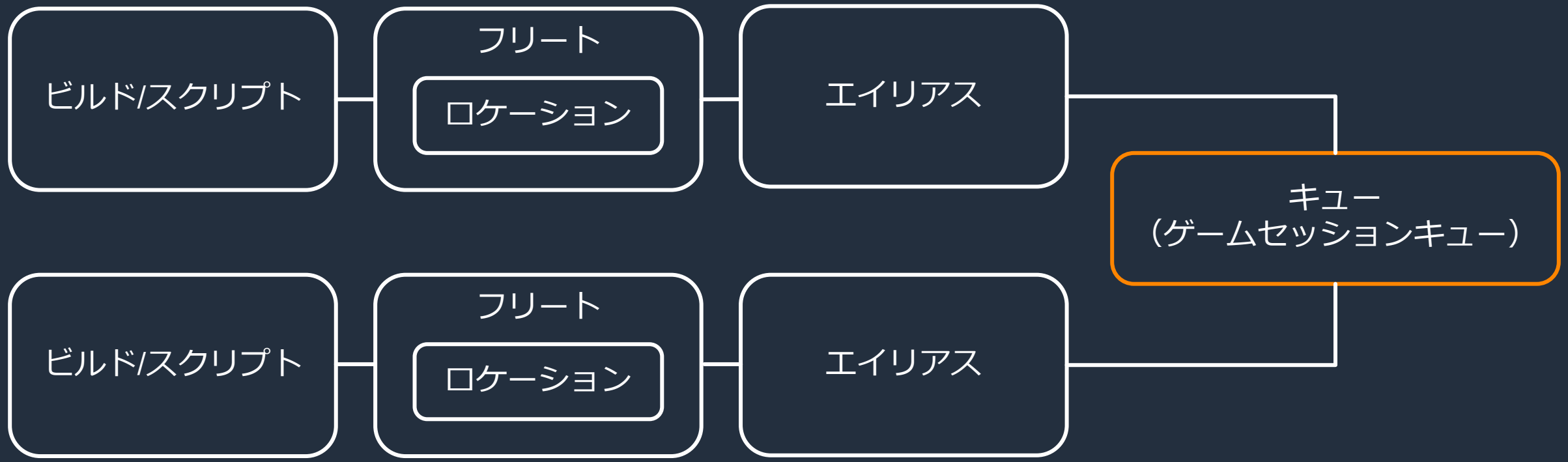
シンプル



ターミナル

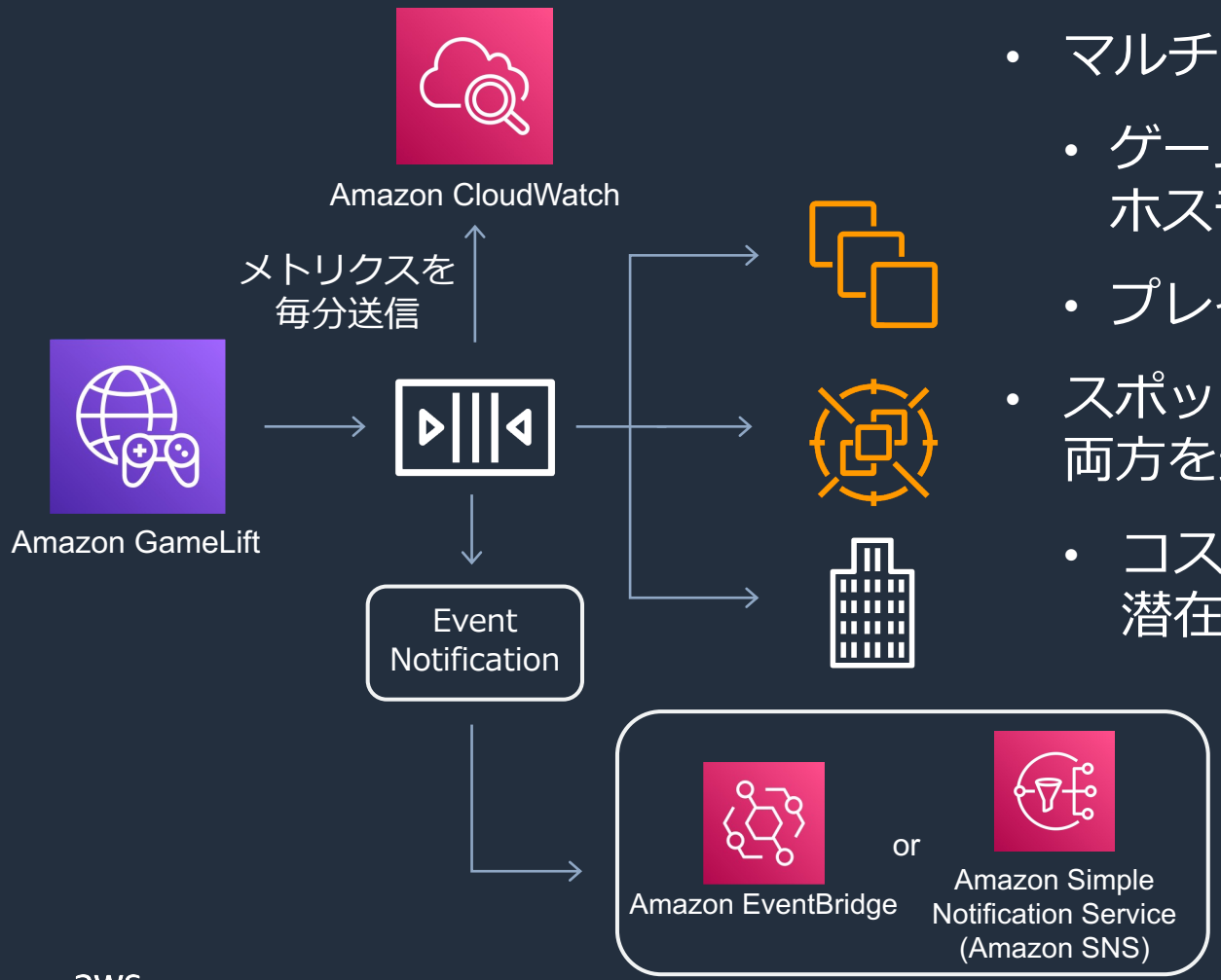
TerminalRoutingStrategyException.message

GameLift ホスティングリソース



キュー（ゲームセッションキュー）

ゲームセッション配置のリクエストに対し
事前定義した優先度に従って最適な場所にゲームセッションを配置するメカニズム



- マルチロケーションフリートを送信先に追加
  - ゲームセッションの配置速度と
    ホスティングの耐障害性を向上
  - プレイヤーのレイテンシーを最小限に抑える
- スポットインスタンスとオンデマンドのフリートの
  両方を追加
  - コストを最適化しつつ
    潜在的なスポット中断の影響を最小限に抑える

キュー – FleetIQ アルゴリズム

可能な限り最適なゲームセッション配置先を検索する

以下のフリートを送信先の候補から除外

- スポット中断率が許容できないフリート
- プレイヤーレイテンシーが
  ポリシーの最大制限を超えている
  リージョンのフリート

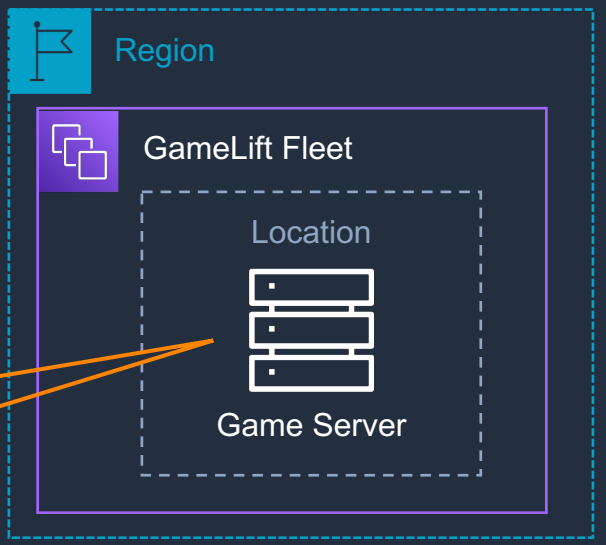
ゲームセッション配置優先順位
（入れ替え可能）

- レイテンシー
- コスト
- 送信先
- ロケーション

リソースがない場合：
優先順位に従って
次の送信先候補となる
フリートに対して評価

選択されたフリートで
新規のゲームセッションがホ
スティングできる
サーバープロセスが利用可能
かどうかを評価

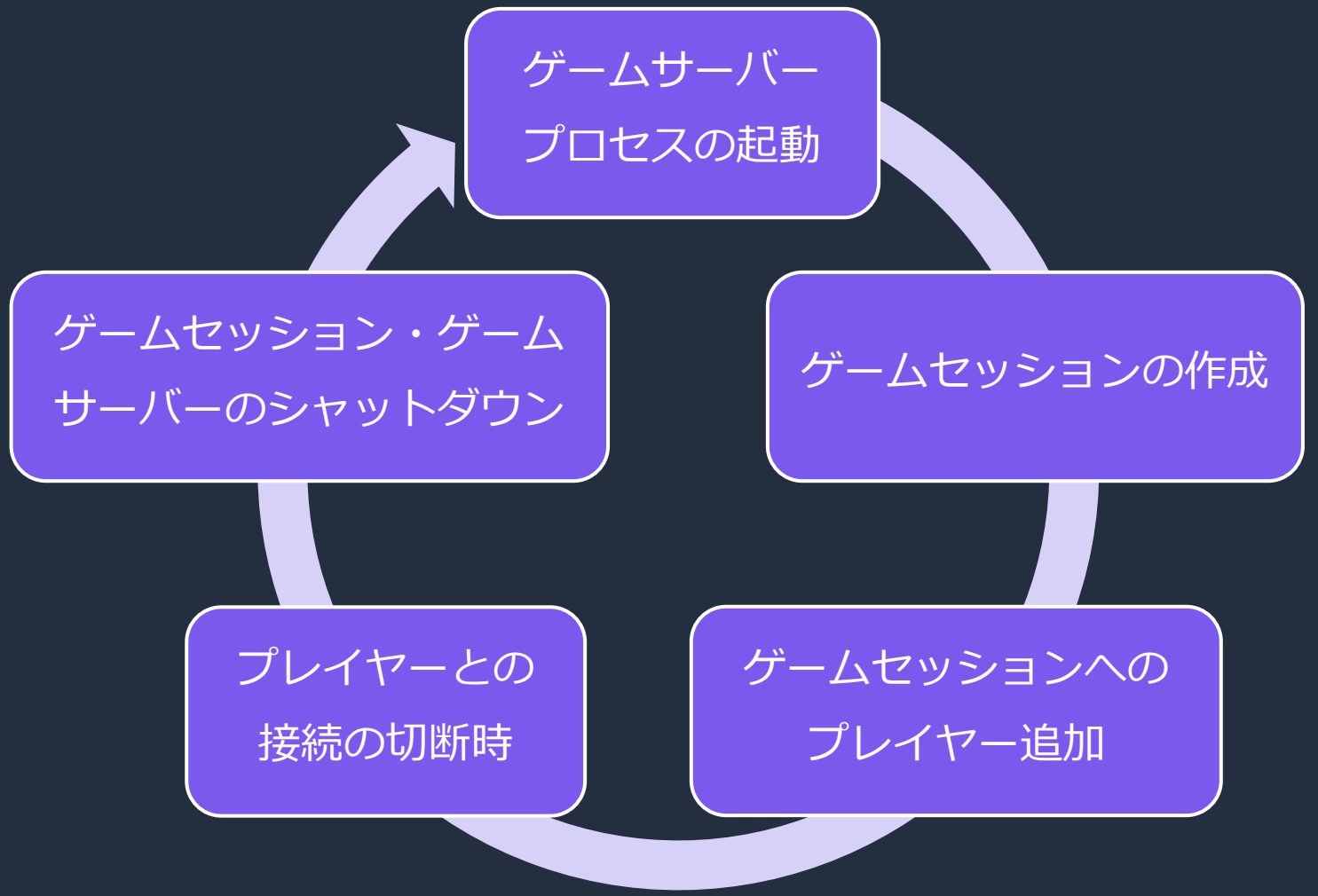
最適なゲームセッション配置先



アジェンダ

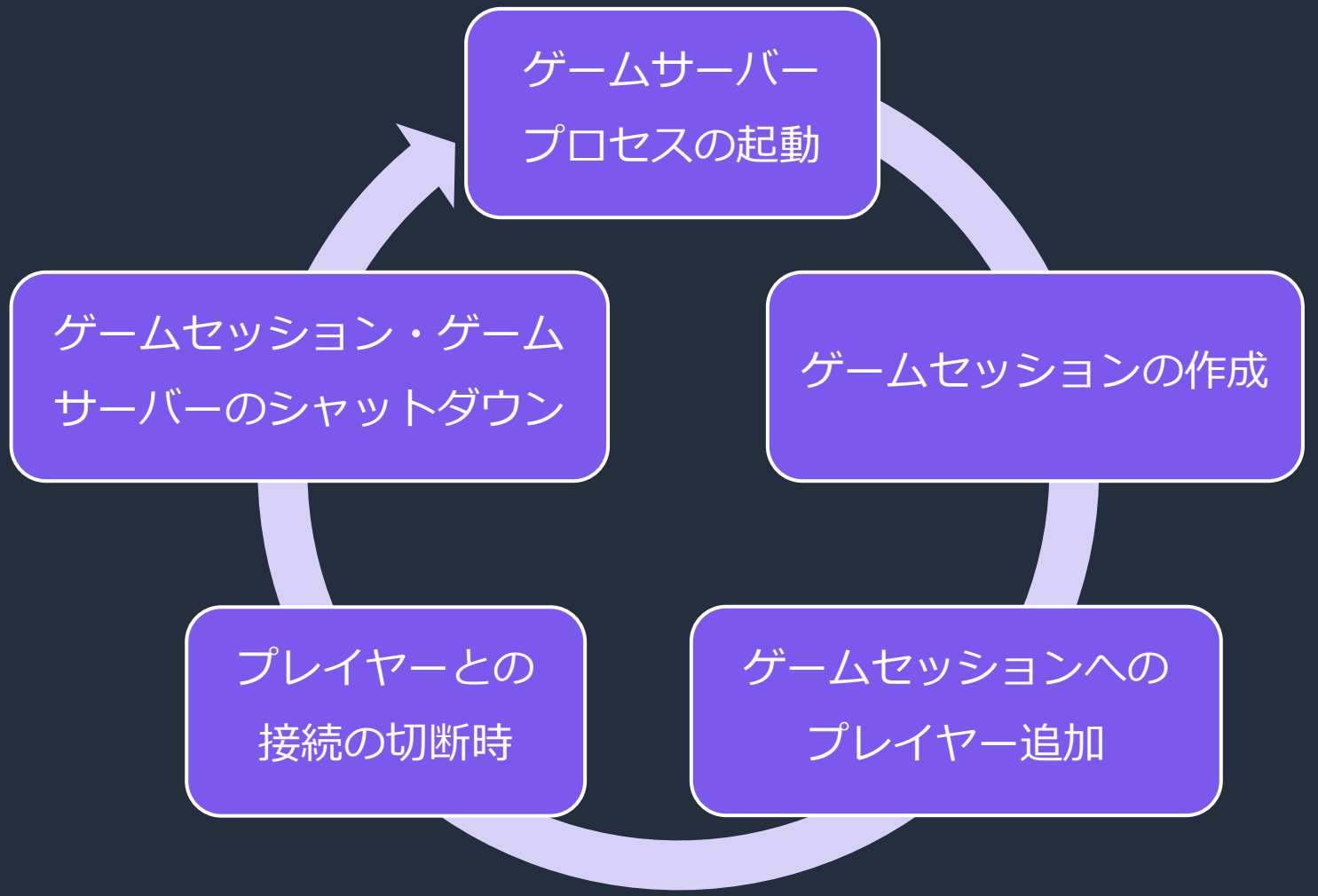
1. マルチプレイヤーゲームにおける専用ゲームサーバーホスティングの課題
2. Amazon GameLift の概観
3. GameLift Managed Hosting の仕組み
4. GameLift Managed Hosting のリソース
5. GameLift ゲームサーバーのライフサイクル
6. 料金情報
7. まとめ

GameLift ゲームサーバーのライフサイクル



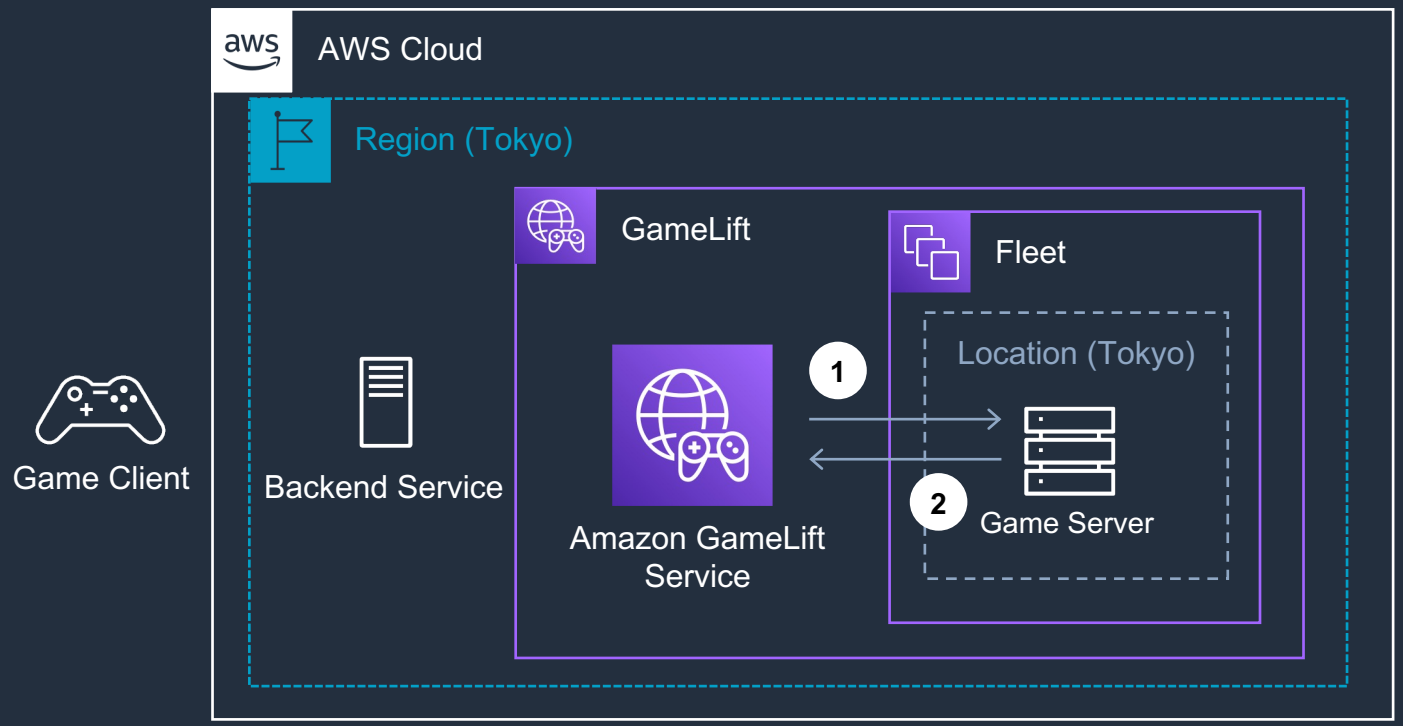
- Managed EC2 フリートへの接続
  - フリートとロケーションを直接指定する場合
  - ゲームセッションキューを利用する場合
- Anywhere フリートへの接続
  - 事前準備として登録したハードウェア上のゲームサーバープロセスを GameLift で認証する
    - フリートとロケーションを直接指定する場合
    - ゲームセッションキューを利用する場合

GameLift ゲームサーバーのライフサイクル



- Managed EC2 フリートへの接続
  - フリートとロケーションを直接指定する場合
  - ゲームセッションキューを利用する場合
- Anywhere フリートへの接続
  - 事前準備として登録したハードウェア上のゲームサーバープロセスを GameLift で認証する
    - フリートとロケーションを直接指定する場合
    - ゲームセッションキューを利用する場合

ゲームサーバープロセスの起動



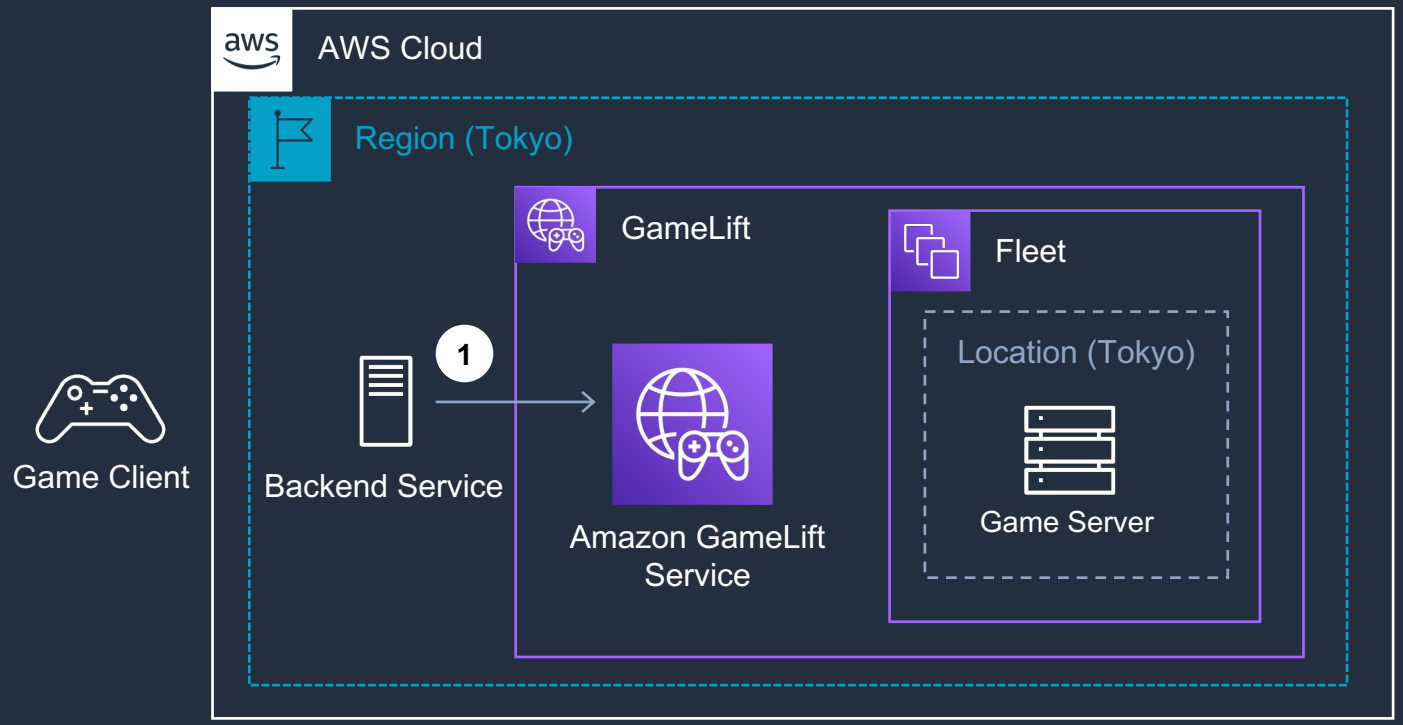
① GameLift サービスが
フリートのゲームサーバープロセスを起動

② ゲームサーバーで InitSDK() を呼び出して
GameLift Server SDK 初期化後、
ProcessReady() を呼び出し、
ゲームセッションをホストする準備が
できた旨を GameLift サービスに通知

- 以降 GameLift サービスが 60 秒ごとに
onHealthCheck コールバックを呼び出し

ゲームセッションの作成

フリートとロケーションを直接指定して作成する場合

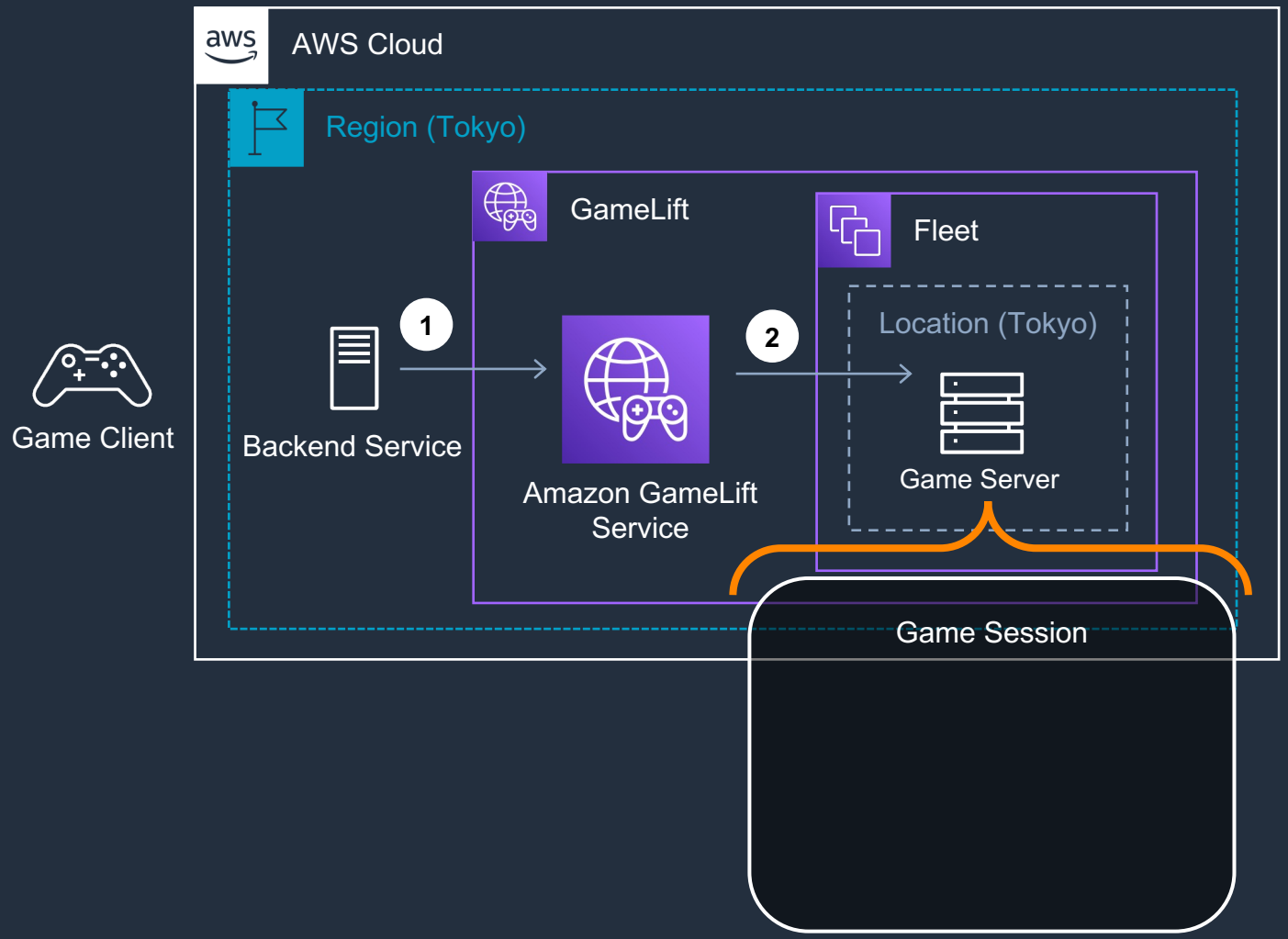


① バックエンドサービスが AWS SDK で CreateGameSession を呼び出す

- 下記を指定
  - 対象のフリートもしくはエイリアス ID
  - 対象のロケーション
  - そのゲームセッションに対して同時に接続可能なプレイヤーセッションの最大数

ゲームセッションの作成

フリートとロケーションを直接指定して作成する場合

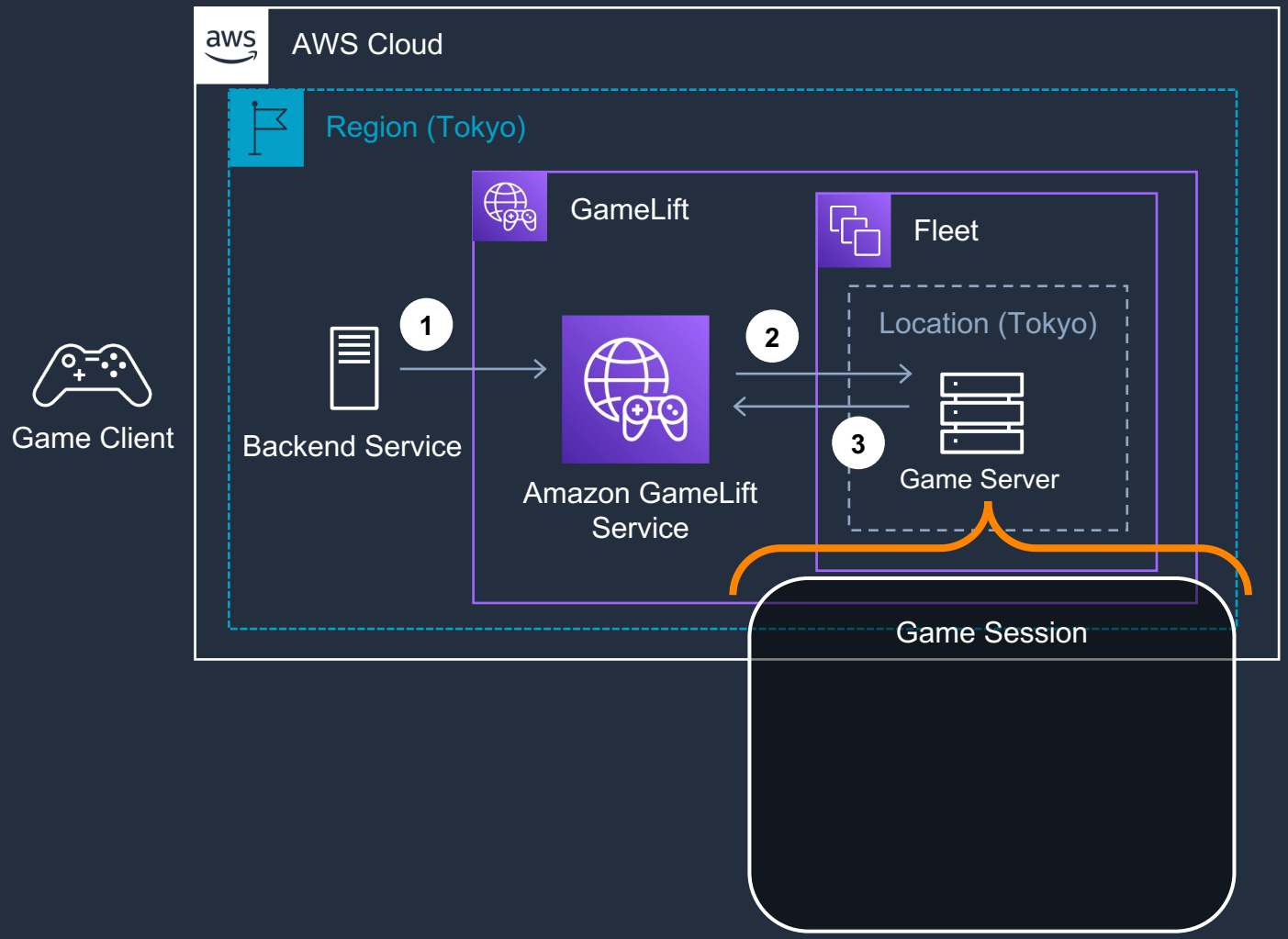


① バックエンドサービスが AWS SDK で CreateGameSession を呼び出す

② GameLift サービスが待機中のゲームサーバープロセスに対しゲームセッションを作成

ゲームセッションの作成

フリートとロケーションを直接指定して作成する場合



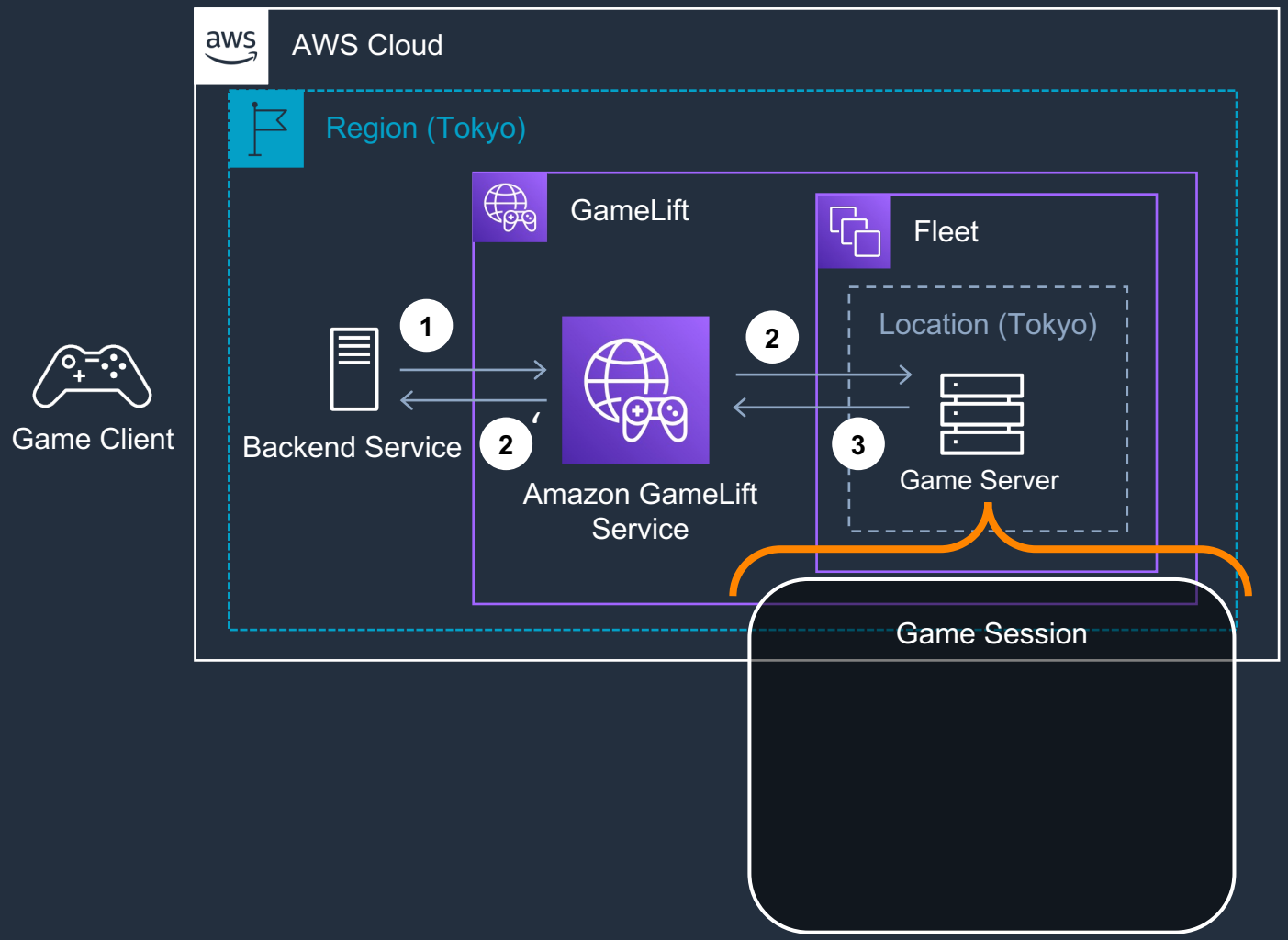
① バックエンドサービスが AWS SDK で CreateGameSession を呼び出す

② GameLift サービスが待機中のゲームサーバープロセスに対しゲームセッションを作成

③ OnStartGameSession コールバックが呼び出される。ゲームサーバーで ActivateGameSession() を呼び出しプレイヤーを受け入れる準備ができたことを GameLift サービスに通知する

ゲームセッションの作成

フリートとロケーションを直接指定して作成する場合



① バックエンドサービスが AWS SDK で CreateGameSession を呼び出す

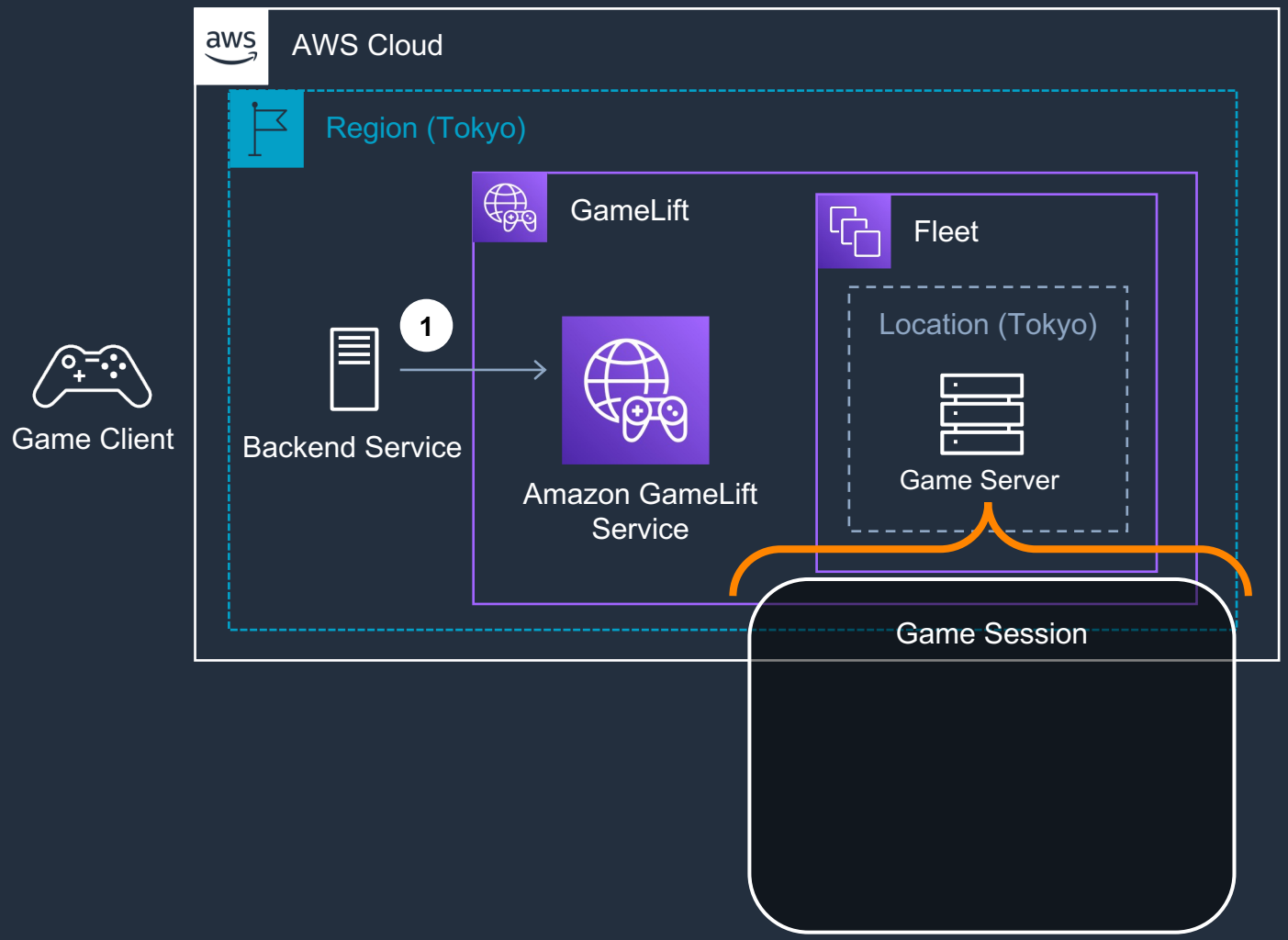
② GameLift サービスが待機中のゲームサーバープロセスに対しゲームセッションを作成

③ OnStartGameSession コールバックが呼び出される。ゲームサーバーで ActivateGameSession() を呼び出しプレイヤーを受け入れる準備ができたことを GameLift サービスに通知する

②' CreateGameSession の返却値としてゲームセッションの情報が呼び出し元に渡される

ゲームセッションへのプレイヤー追加

作成済みのゲームセッションが存在する場合

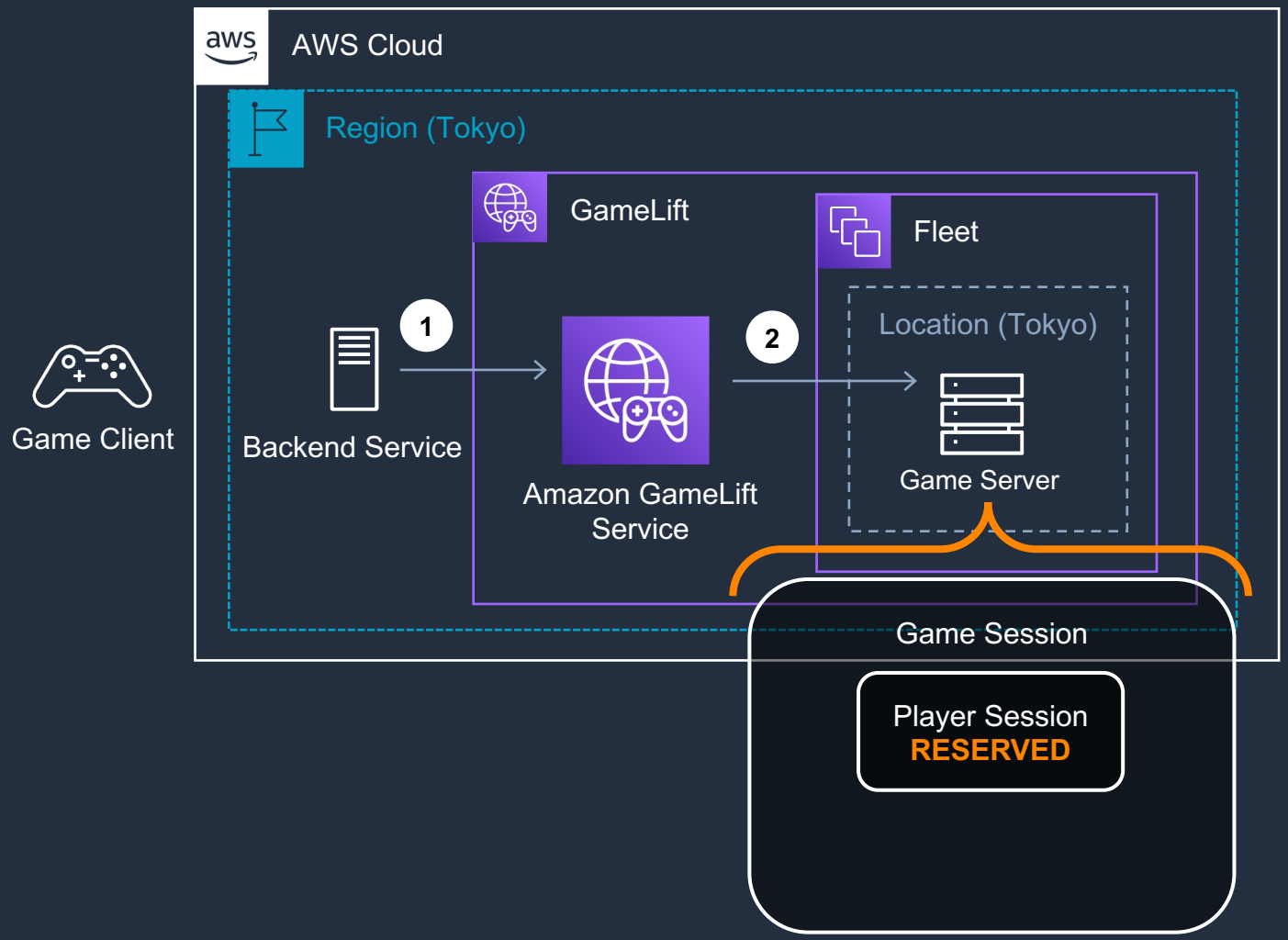


① バックエンドサービスが AWS SDK で作成済みのゲームセッションを対象に CreatePlayerSession もしくは CreatePlayerSessions を呼び出す

- CreatePlayerSession は 1 度に 1 プレイヤーセッションを作成
- CreatePlayerSessions は 1 度に最大 25 プレイヤーセッションを作成可能

ゲームセッションへのプレイヤー追加

作成済みのゲームセッションが存在する場合



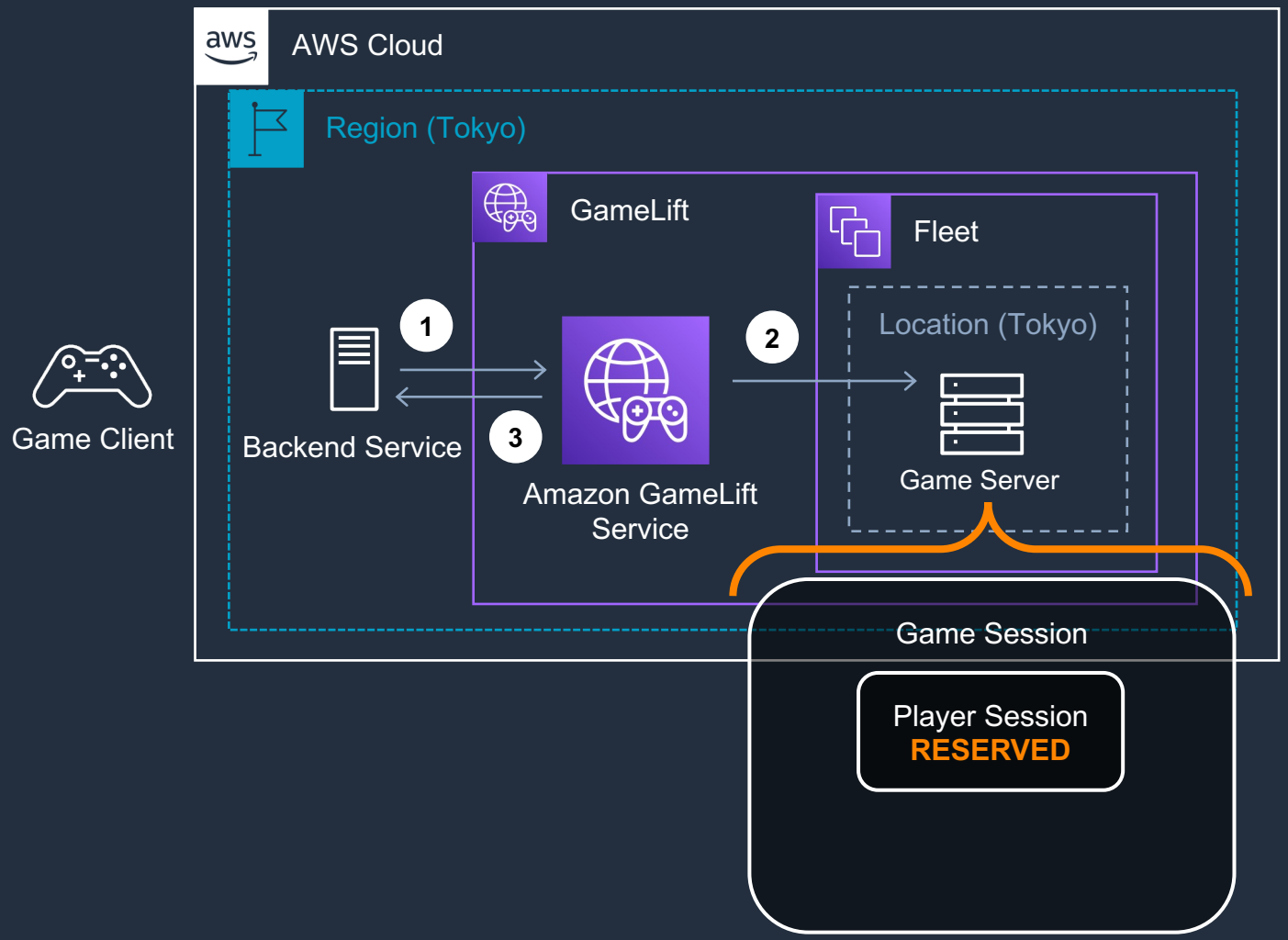
① バックエンドサービスが AWS SDK で作成済みのゲームセッションを対象に CreatePlayerSession もしくは CreatePlayerSessions を呼び出す

② ゲームセッションのステータスを確認しプレイヤーセッションのステータスが “RESERVED” に設定された上で作成される

- プレイヤーを受け入れる予約枠を 60 秒間確保
- 60 秒以内にゲームクライアントから接続がなくステータスが変更されない場合、“TIMEOUT” に設定され予約枠が開放

ゲームセッションへのプレイヤー追加

作成済みのゲームセッションが存在する場合



① バックエンドサービスが AWS SDK で作成済みのゲームセッションを対象に CreatePlayerSession もしくは CreatePlayerSessions を呼び出す

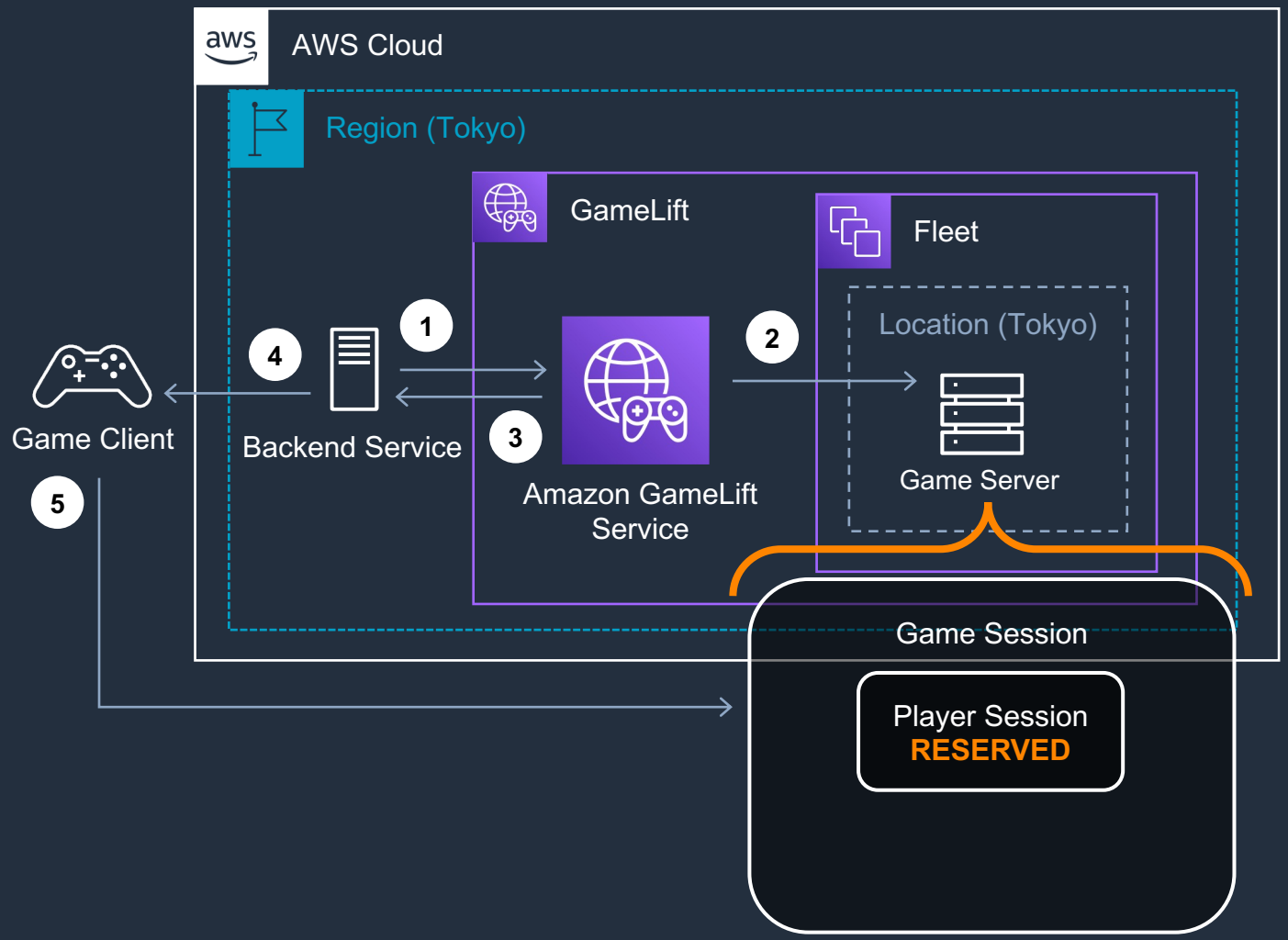
② ゲームセッションのステータスを確認しプレイヤーセッションのステータスが “RESERVED” に設定された上で作成される

③ CreatePlayerSession の返却値として作成されたプレイヤーセッションの情報が呼び出し元に渡される

- 接続先の IP アドレス、ポート番号、DNS 名、PlayerSessionID などが含まれる

ゲームセッションへのプレイヤー追加

作成済みのゲームセッションが存在する場合



① バックエンドサービスが AWS SDK で作成済みのゲームセッションを対象に CreatePlayerSession もしくは CreatePlayerSessions を呼び出す

② ゲームセッションのステータスを確認しプレイヤーセッションのステータスが “RESERVED” に設定された上で作成される

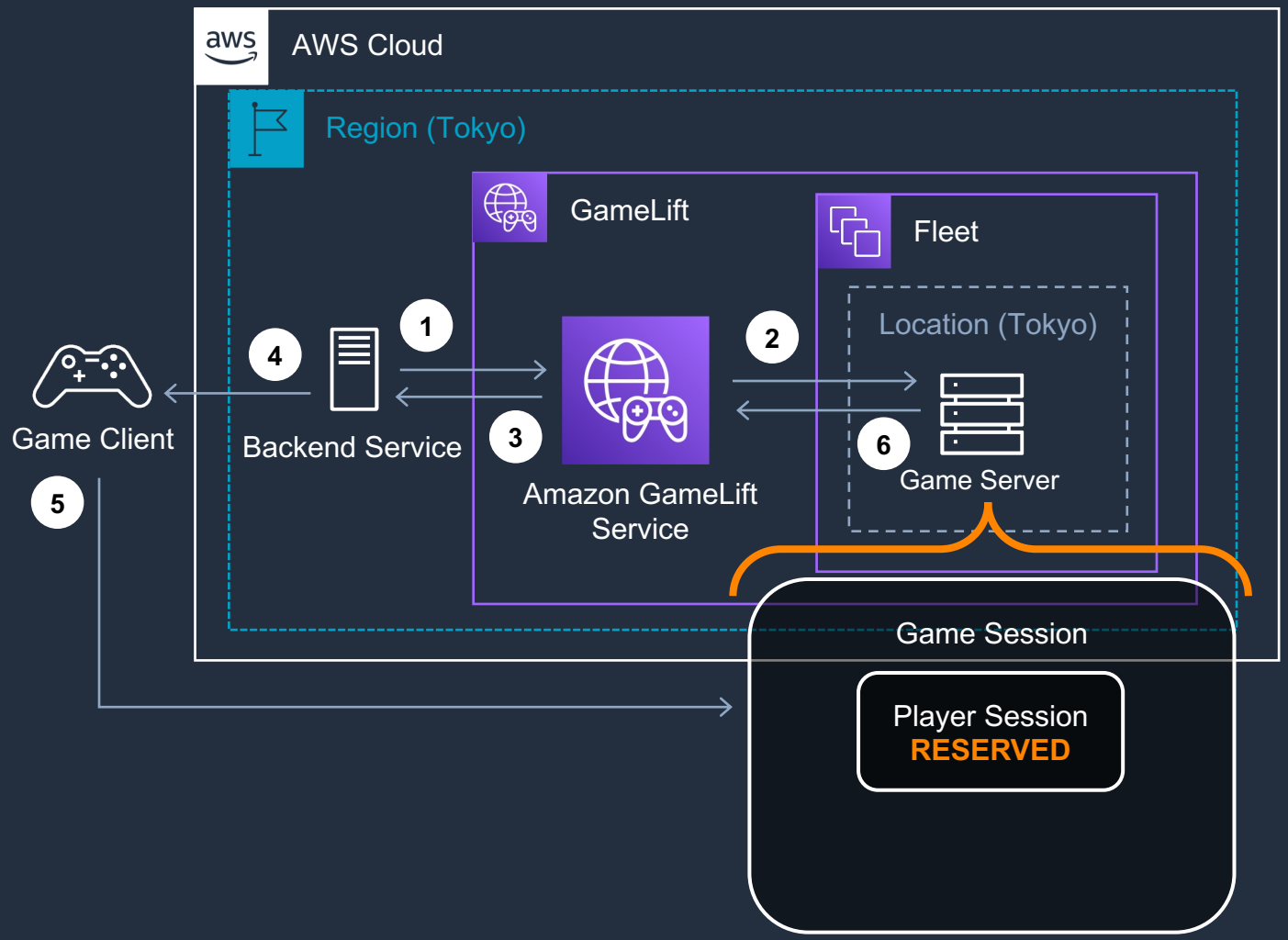
③ CreatePlayerSession の返却値として作成されたプレイヤーセッションの情報が呼び出し元に渡される

④ ゲームクライアントに接続情報を渡す

⑤ 接続情報を使用しゲームサーバープロセスに直接接続する

ゲームセッションへのプレイヤー追加

作成済みのゲームセッションが存在する場合



① バックエンドサービスが AWS SDK で作成済みのゲームセッションを対象に CreatePlayerSession もしくは CreatePlayerSessions を呼び出す

② ゲームセッションのステータスを確認しプレイヤーセッションのステータスが “RESERVED” に設定された上で作成される

③ CreatePlayerSession の返却値として作成されたプレイヤーセッションの情報が呼び出し元に渡される

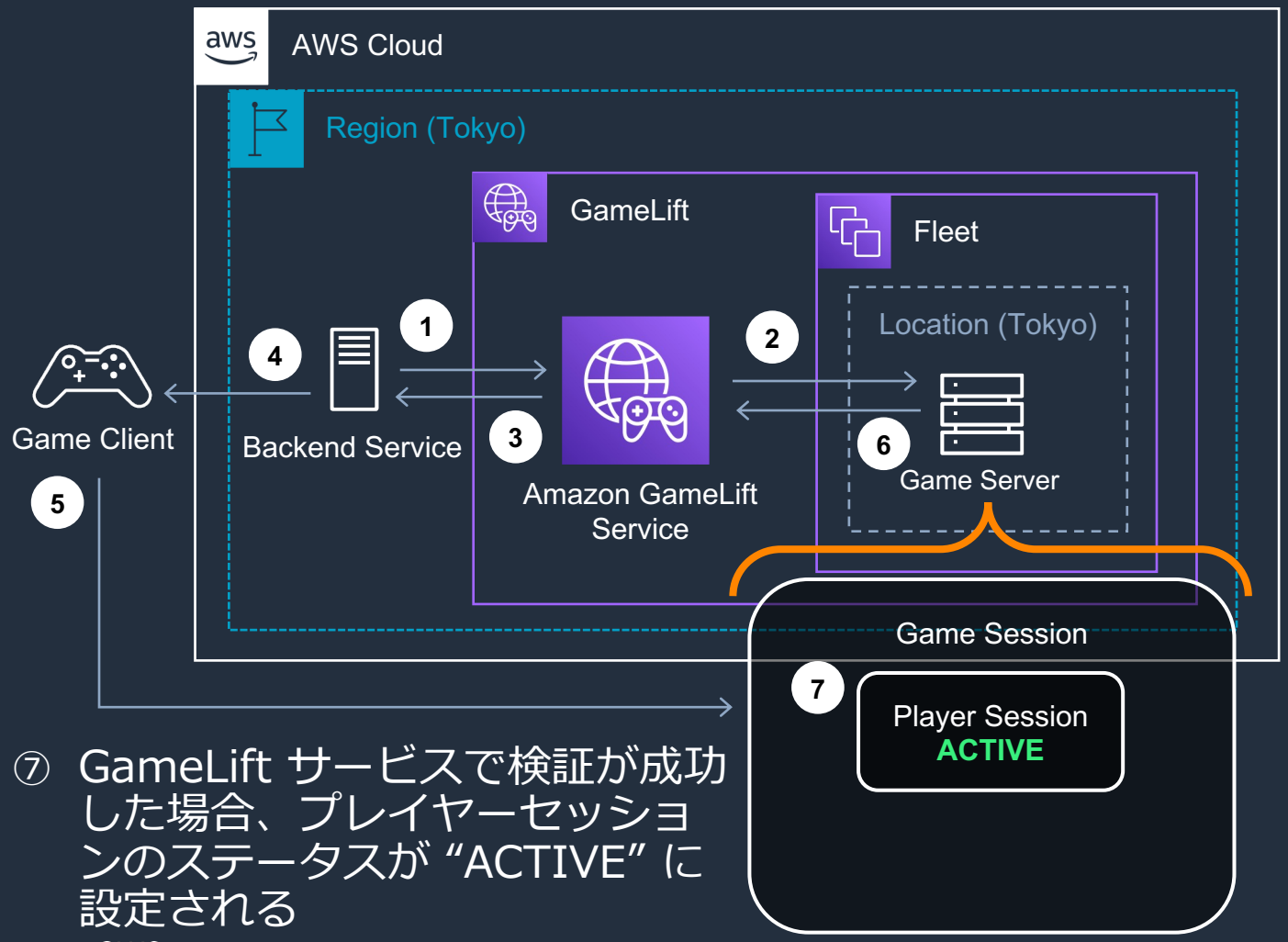
④ ゲームクライアントに接続情報を渡す

⑤ 接続情報を使用しゲームサーバープロセスに直接接続する

⑥ ゲームサーバーが AcceptPlayerSession() を呼び出し、ゲームクライアントから送られた PlayerSessionID を GameLift サービスで検証する

ゲームセッションへのプレイヤー追加

作成済みのゲームセッションが存在する場合



① バックエンドサービスが AWS SDK で作成済みのゲームセッションを対象に CreatePlayerSession もしくは CreatePlayerSessions を呼び出す

② ゲームセッションのステータスを確認しプレイヤーセッションのステータスが “RESERVED” に設定された上で作成される

③ CreatePlayerSession の返却値として作成されたプレイヤーセッションの情報が呼び出し元に渡される

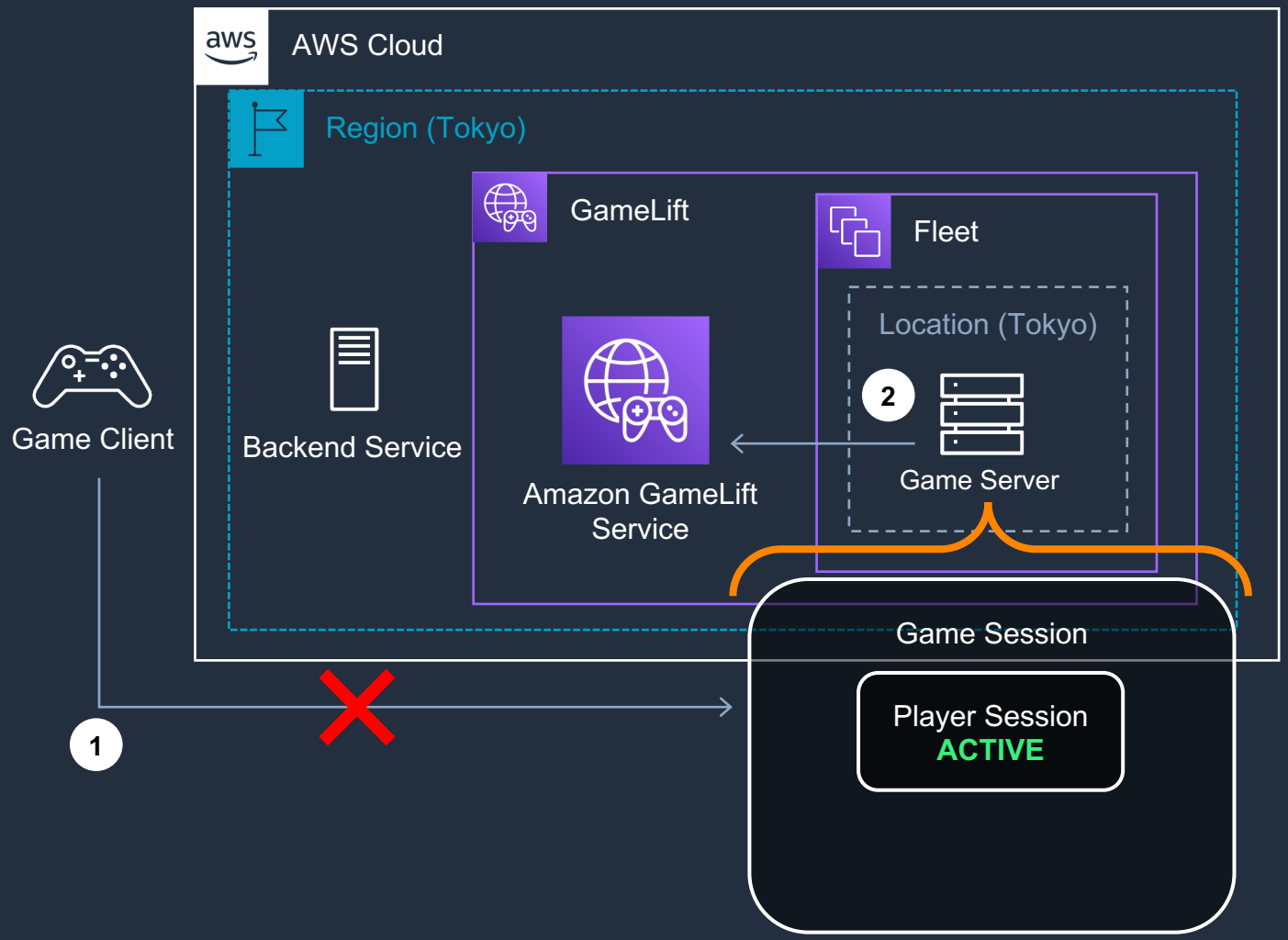
④ ゲームクライアントに接続情報を渡す

⑤ 接続情報を使用しゲームサーバープロセスに直接接続する

⑥ ゲームサーバーが AcceptPlayerSession() を呼び出し、ゲームクライアントから送られた PlayerSessionID を GameLift サービスで検証する

⑦ GameLift サービスで検証が成功した場合、プレイヤーセッションのステータスが “ACTIVE” に設定される

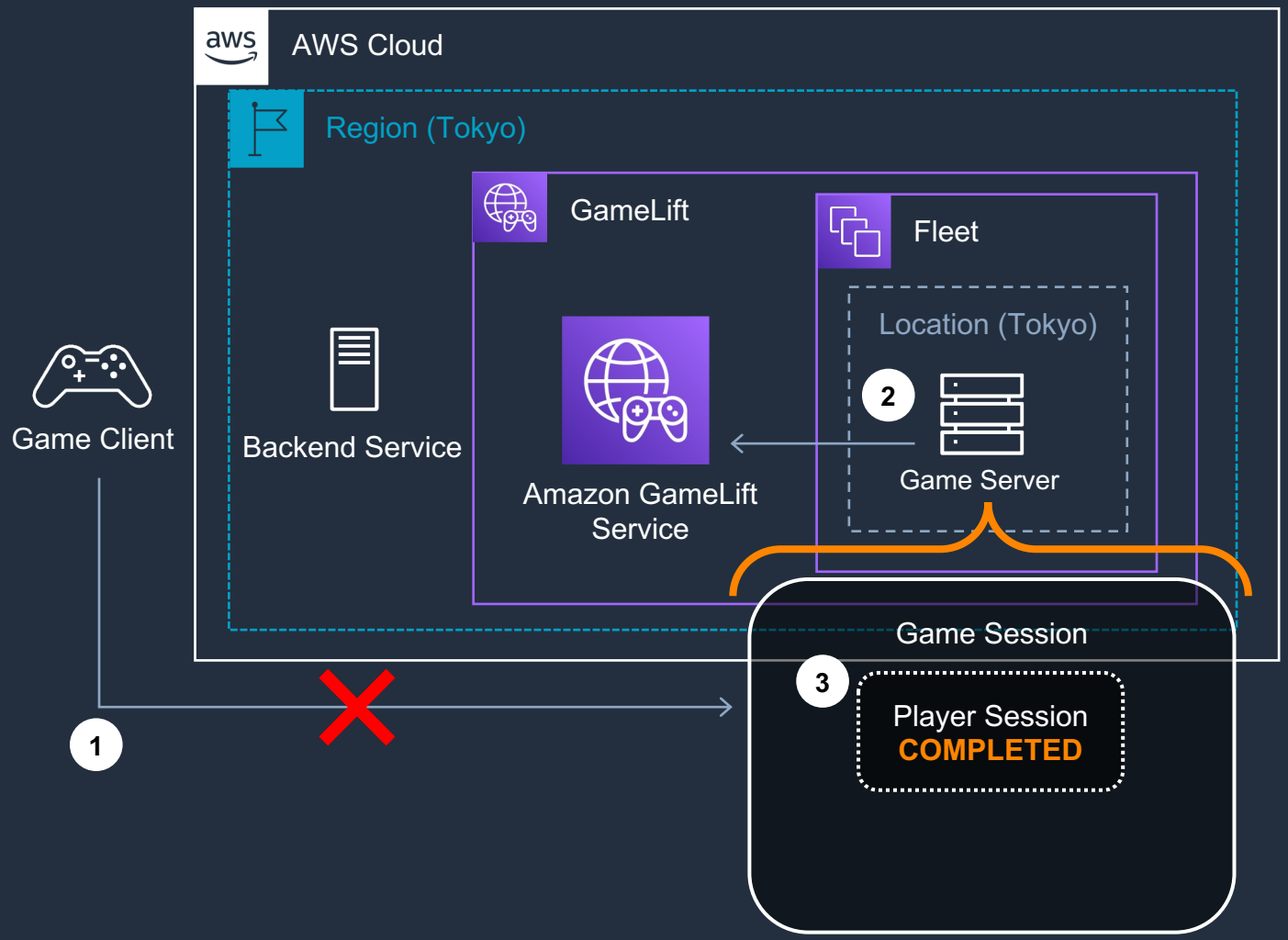
プレイヤーとの接続の切断時



① プレイヤーとゲームサーバーの接続が切断される

② ゲームサーバーで接続が切断された旨を検知し、RemovePlayerSession() を呼び出して GameLift サービスに通知

プレイヤーとの接続の切断時

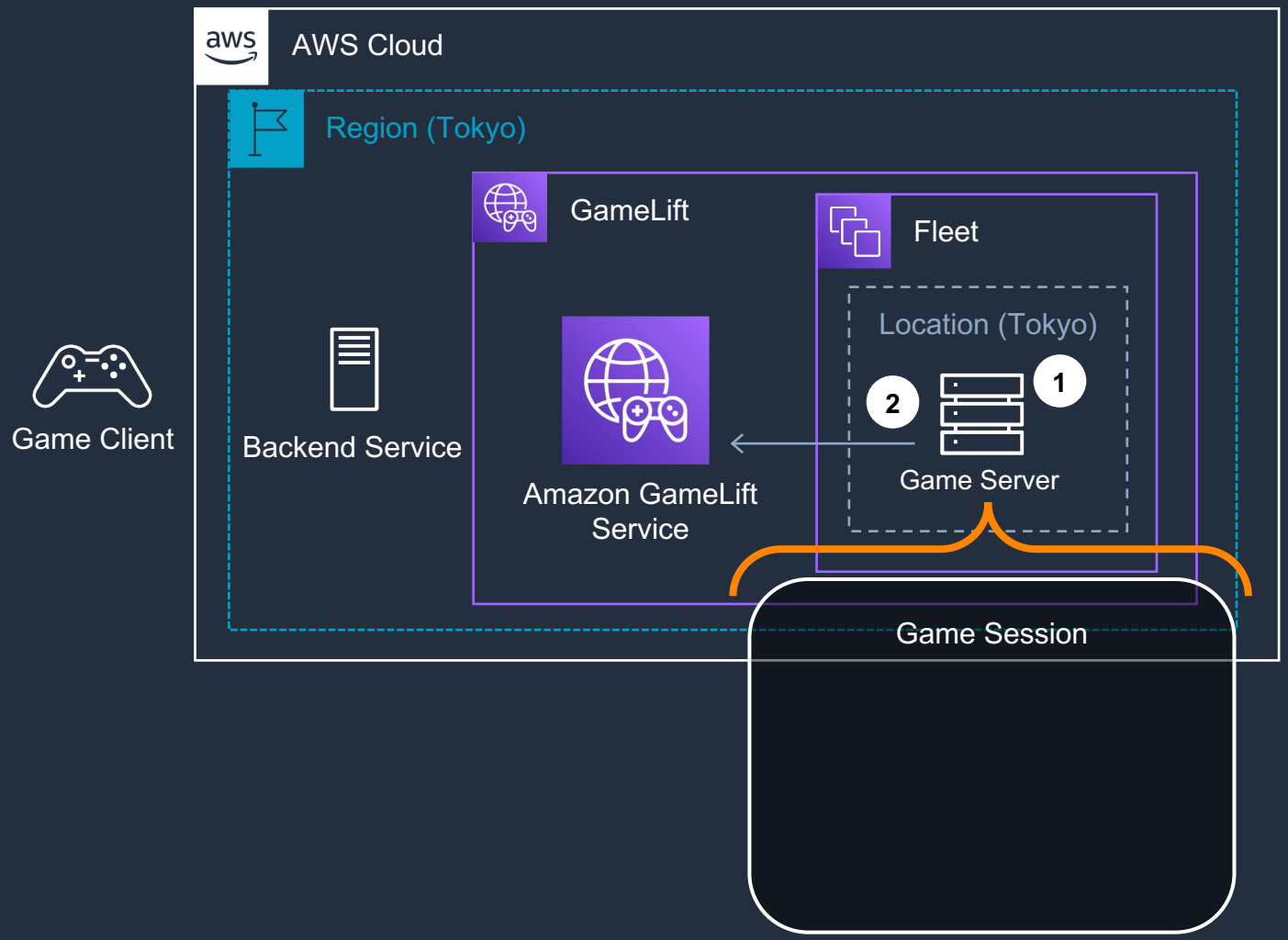


① プレイヤーとゲームサーバーの接続が切断される

② ゲームサーバーで接続が切断された旨を検知し、RemovePlayerSession() を呼び出して GameLift サービスに通知

③ GameLift サービスが対象のプレイヤーセッションのステータスを “COMPLETED” に設定し、プレイヤーセッションの枠を開放する

ゲームセッション・ゲームサーバーのシャットダウン

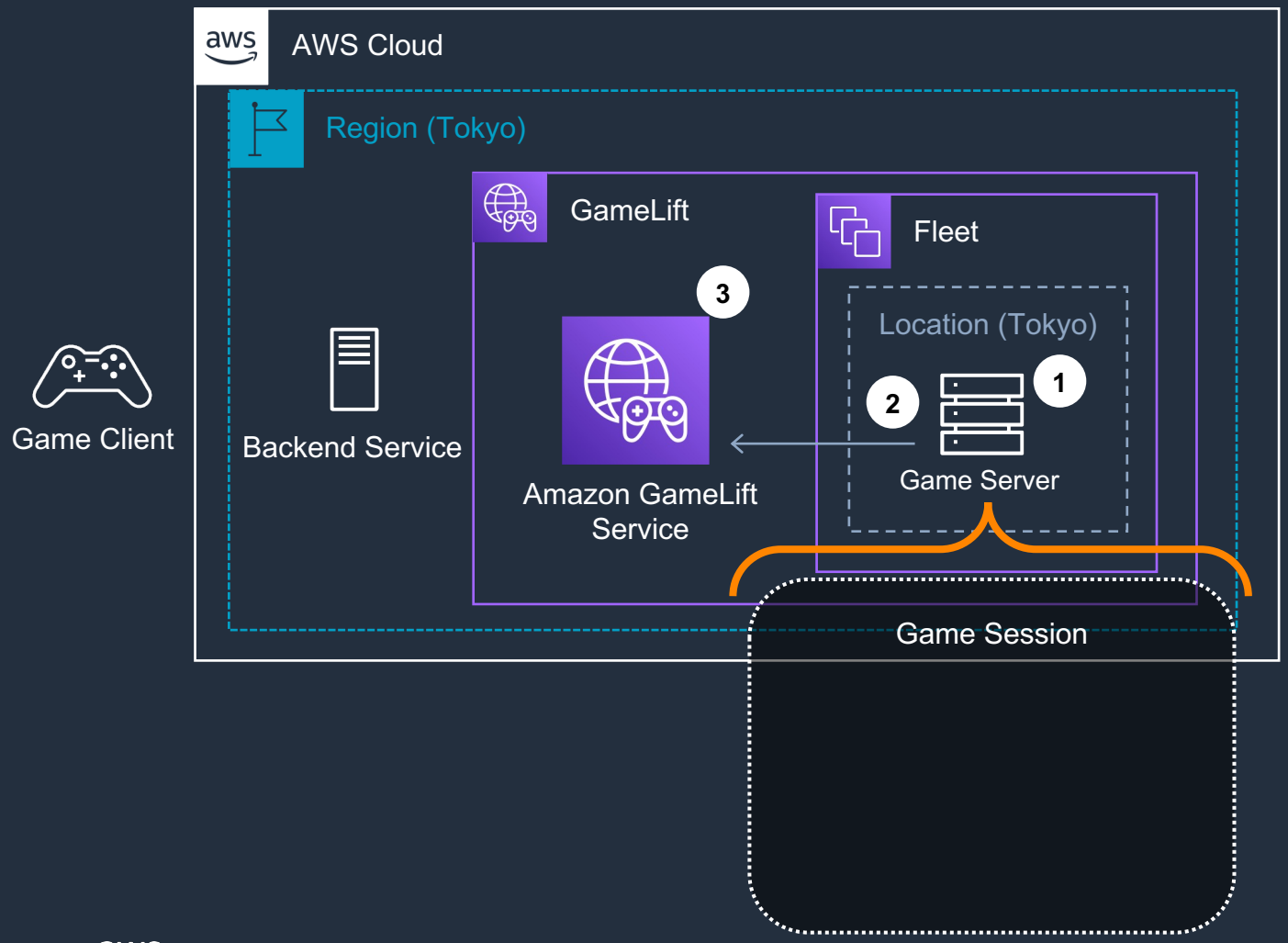


① ゲームサーバーが
ゲームセッションおよびゲームサーバープロセスをシャットダウンする

② ゲームサーバーで ProcessEnding() を呼び出し GameLift サービスにサーバープロセスをシャットダウンする旨を通知

- サーバープロセスのシャットダウン時には Destroy() も呼び出し、
GameLift Server SDK のインスタンスを削除
- ローカルで接続中のエージェントとの通信が閉じる

ゲームセッション・ゲームサーバーのシャットダウン



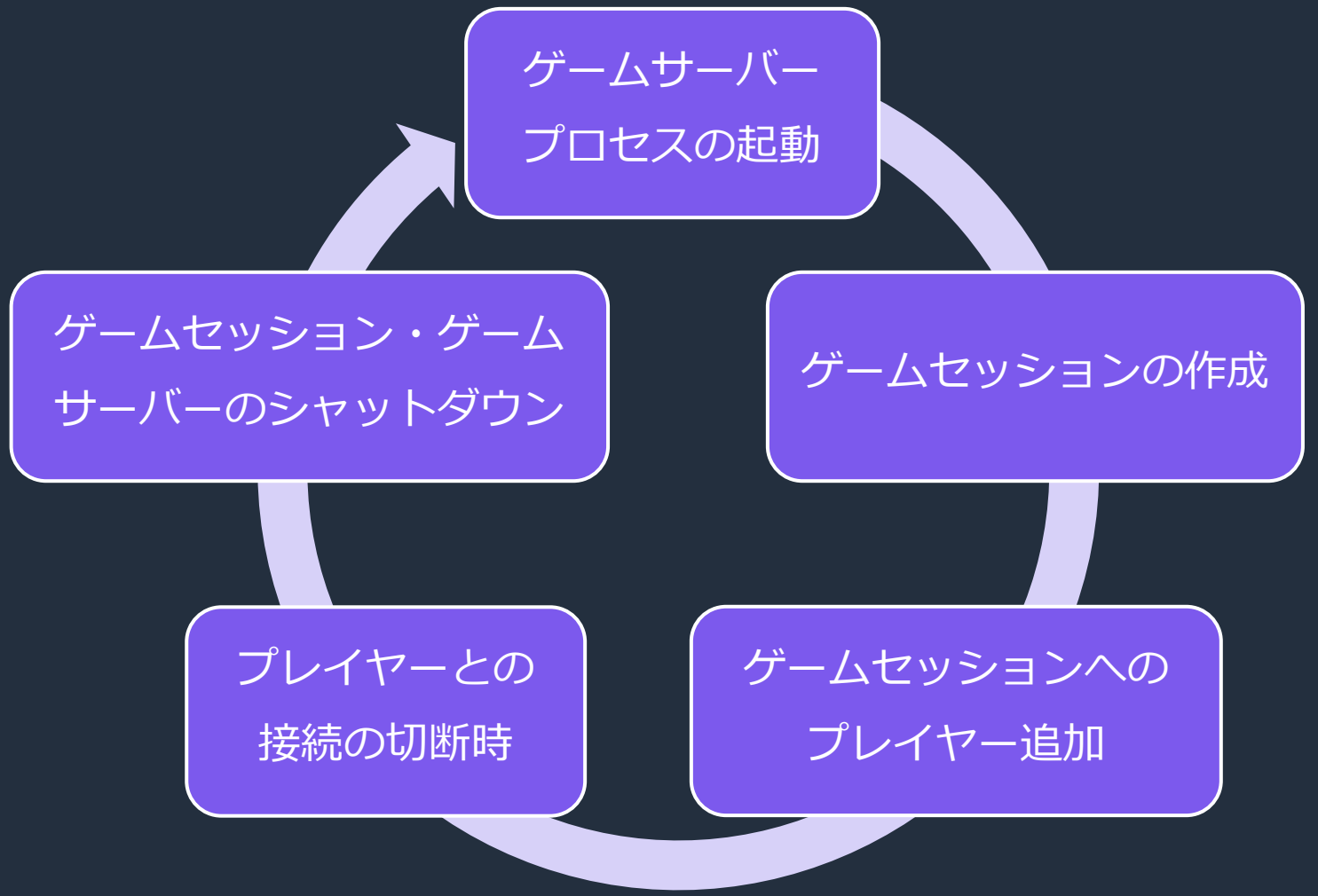
① ゲームサーバーが
ゲームセッションおよびゲームサーバープロセスをシャットダウンする

② ゲームサーバーで ProcessEnding() を呼び出し GameLift サービスにサーバープロセスをシャットダウンする旨を通知

③ GameLift サービスで以下が処理される

- S3 バケットにゲームセッションログを送信
  - 書き出し先は ProcessReady()で定義
  - コンソールか GetGameSessionLogUrl (CLI / API)で取得可能
- ゲームセッションのステータスを “TERMINATED” に設定
- サーバープロセスのステータスを “TERMINATED” に設定
- リソースがリサイクルされる

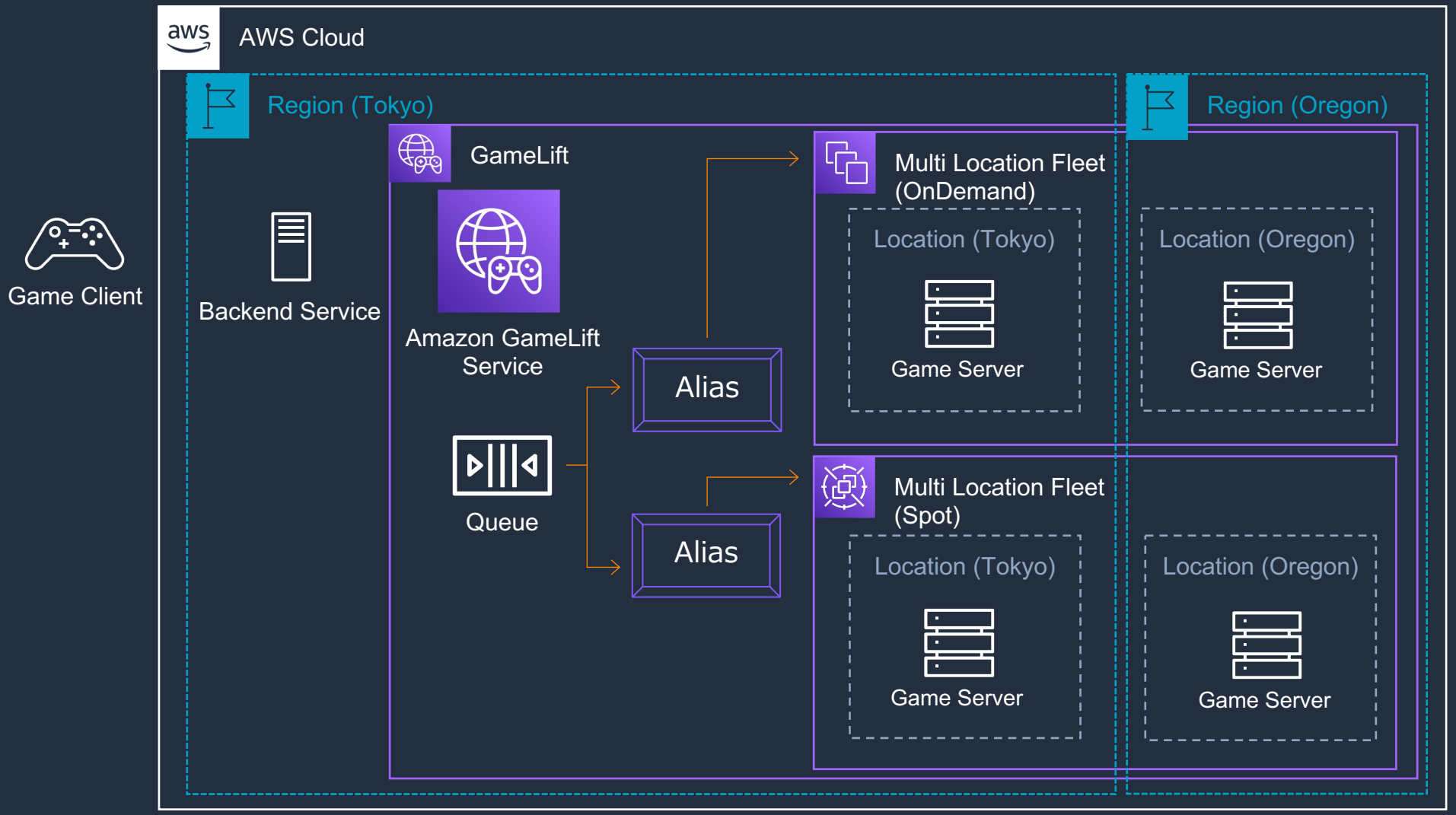
GameLift ゲームサーバーのライフサイクル



- Managed EC2 フリートへの接続
  - フリートとロケーションを直接指定する場合
  - ゲームセッションキューを利用する場合
- Anywhere フリートへの接続
  - 事前準備として登録したハードウェア上のゲームサーバープロセスを GameLift で認証する
    - フリートとロケーションを直接指定する場合
    - ゲームセッションキューを利用する場合

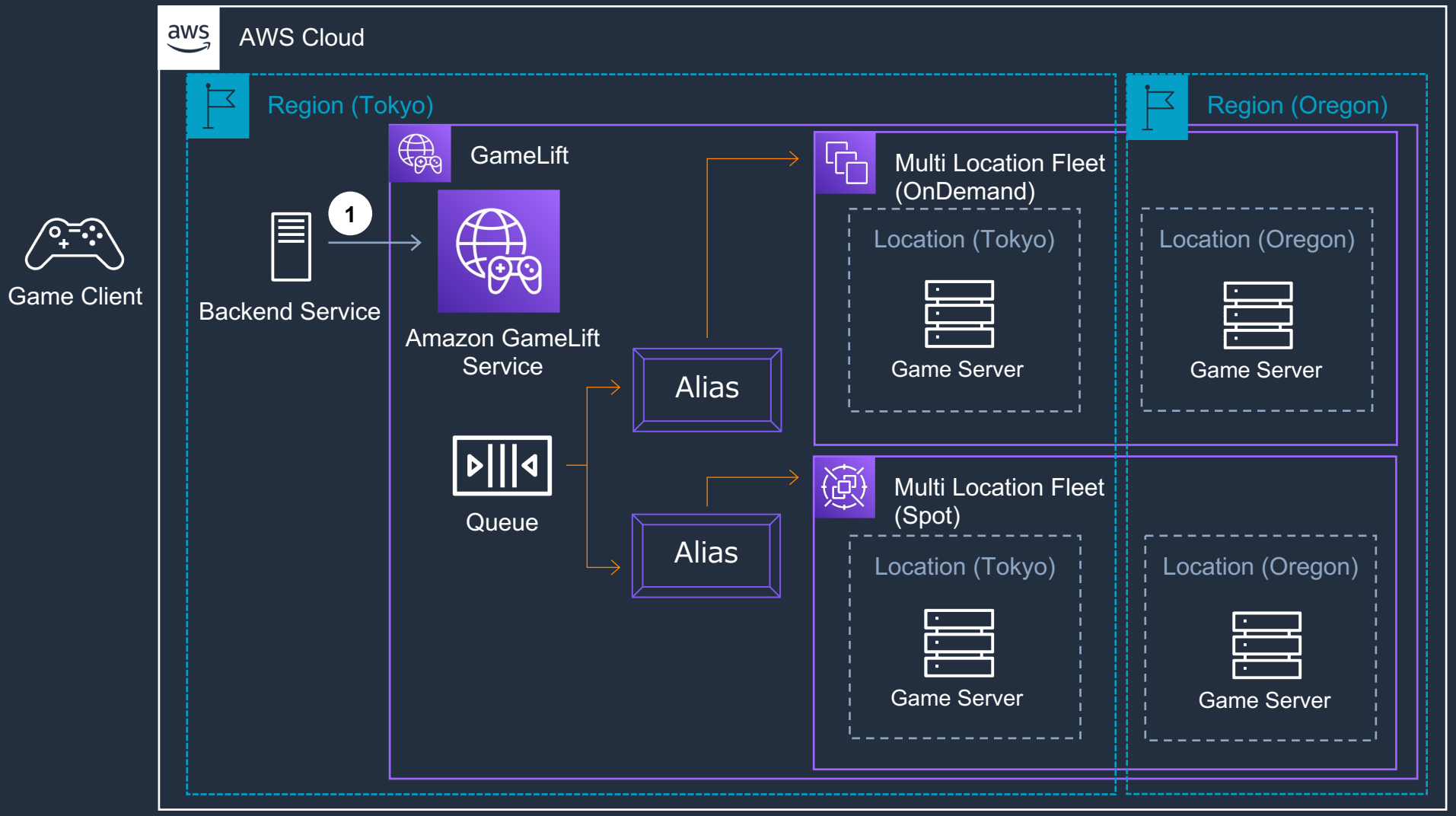
ゲームセッション・プレイヤーセッションの作成

ゲームセッションキューを利用してゲームセッションを配置する場合



ゲームセッション・プレイヤーセッションの作成

ゲームセッションキューを利用してゲームセッションを配置する場合

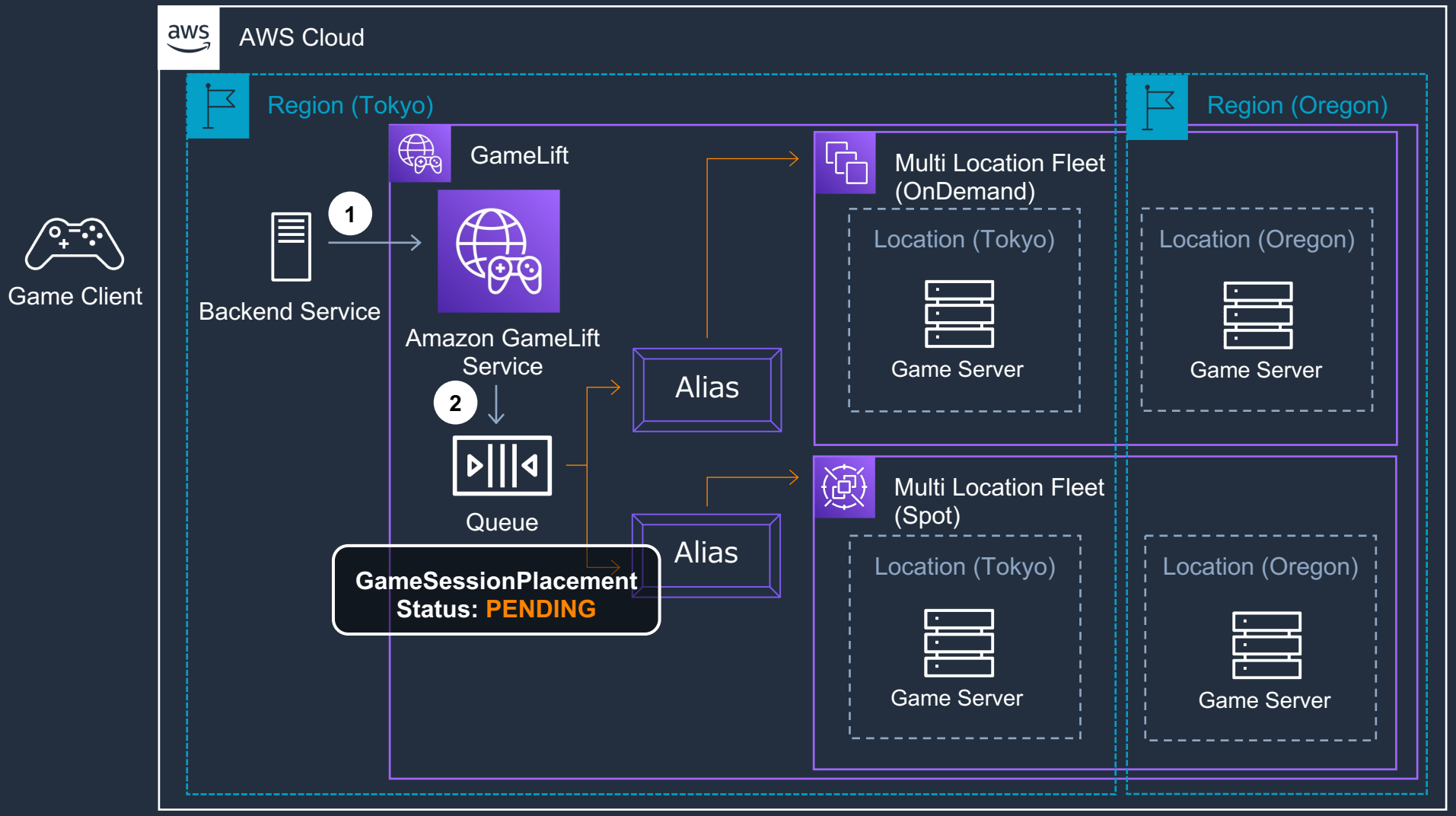


① バックエンドサービスが AWS SDK で StartGameSessionPlacement を呼び出す

- 以下の情報を含める
  - 使用するキューの名前
  - 配置するゲームセッションに対して同時に接続可能なプレイヤーセッションの最大数
  - PlacementId (UUID など)
- プレイヤーレイテンシー情報を含め FleetIQ に参照させる
- プレイヤーセッションを同時に作成可能
- ゲームセッション配置リクエストは非同期で進行

ゲームセッション・プレイヤーセッションの作成

ゲームセッションキューを利用してゲームセッションを配置する場合

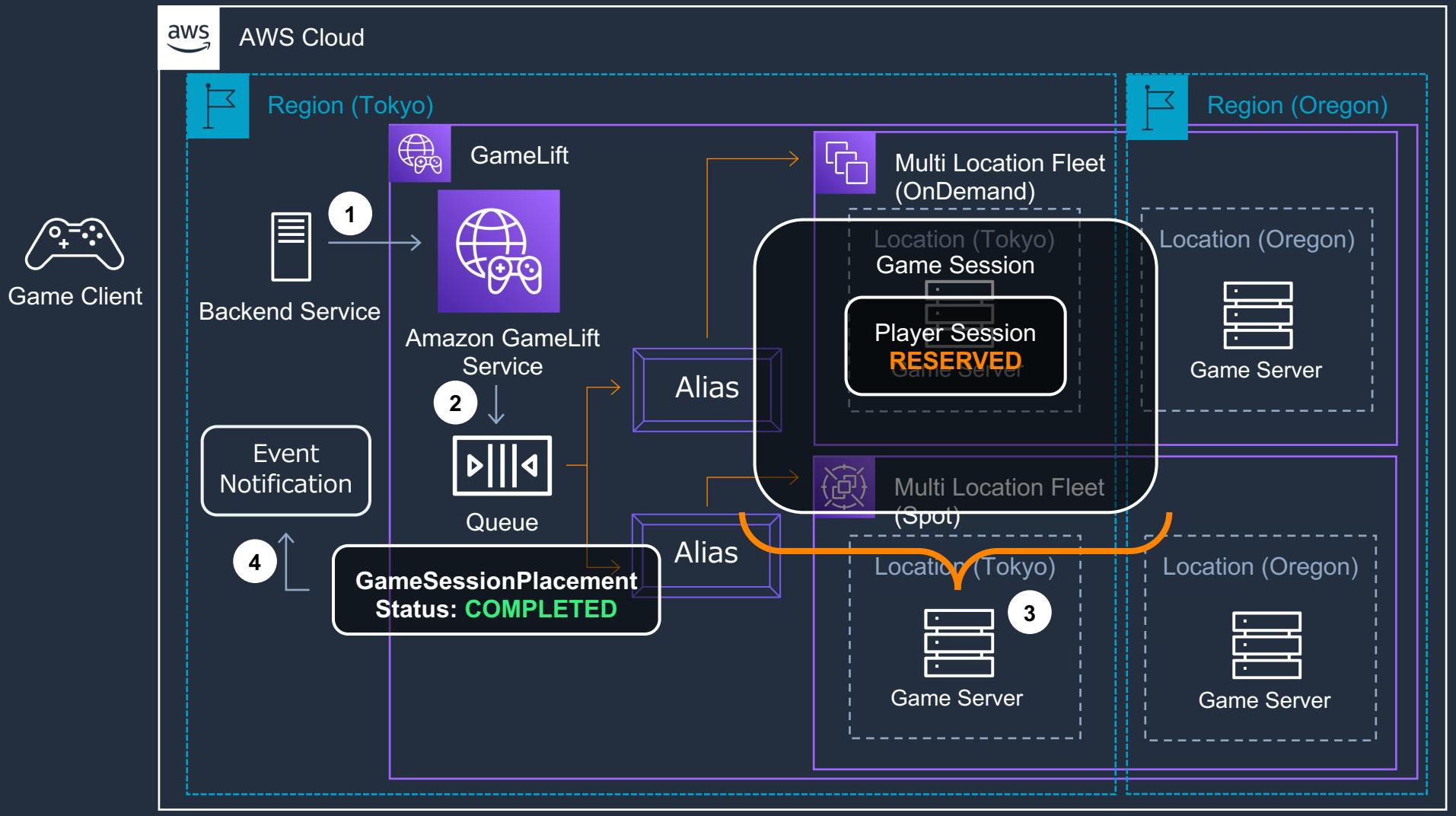


① バックエンドサービスが AWS SDK で StartGameSessionPlacement を呼び出す

② FleetIQ アルゴリズムが最適なゲームセッション配置先を検索

ゲームセッション・プレイヤーセッションの作成

ゲームセッションキューを利用してゲームセッションを配置する場合



① バックエンドサービスが AWS SDK で StartGameSessionPlacement を呼び出す

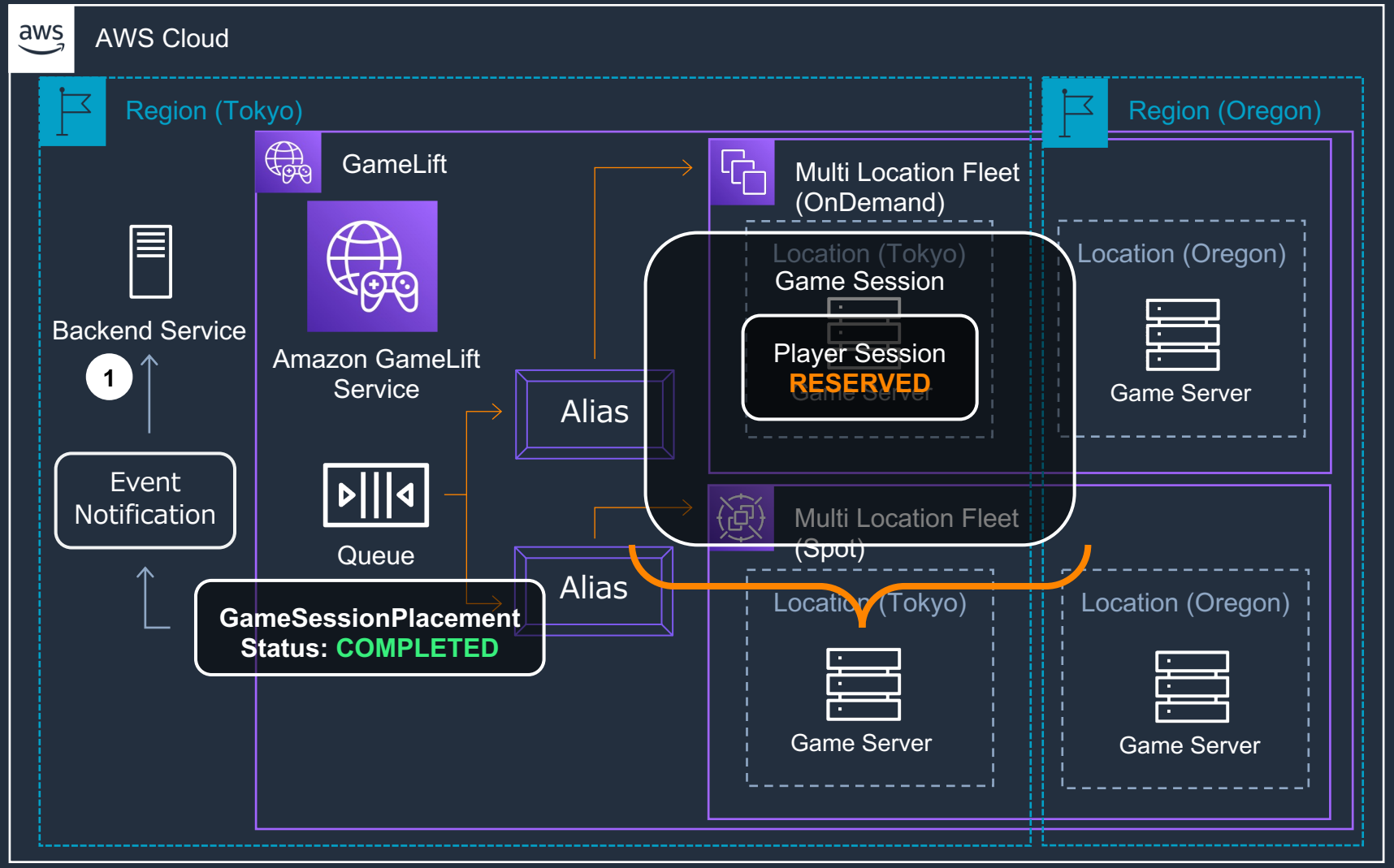
② FleetIQ アルゴリズムが最適なゲームセッション配置先を検索

③ 検索したゲームサーバープロセスでゲームセッションを作成
リクエストにプレイヤーデータを含む場合はプレイヤーセッションも作成

④ ゲームセッション配置リクエストのステータスの変更イベントが通知される

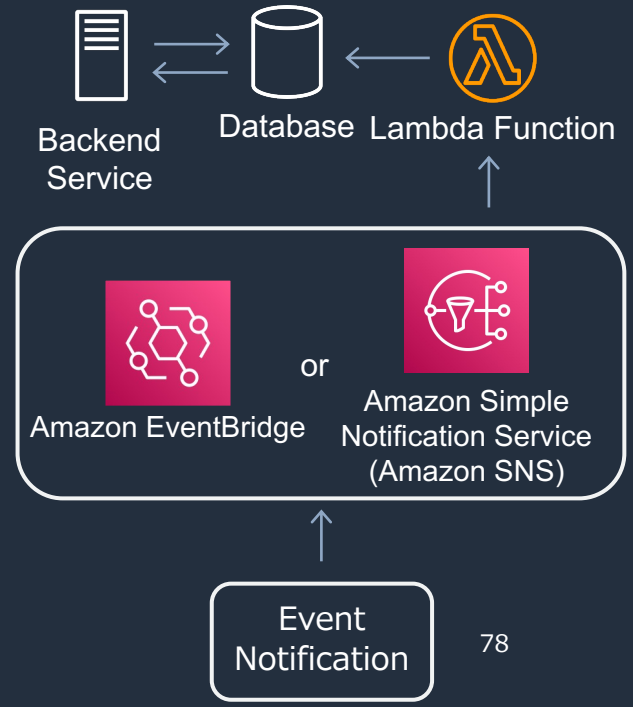
ゲームセッションへのプレイヤー追加

ゲームセッションキューを利用してゲームセッションを配置する場合



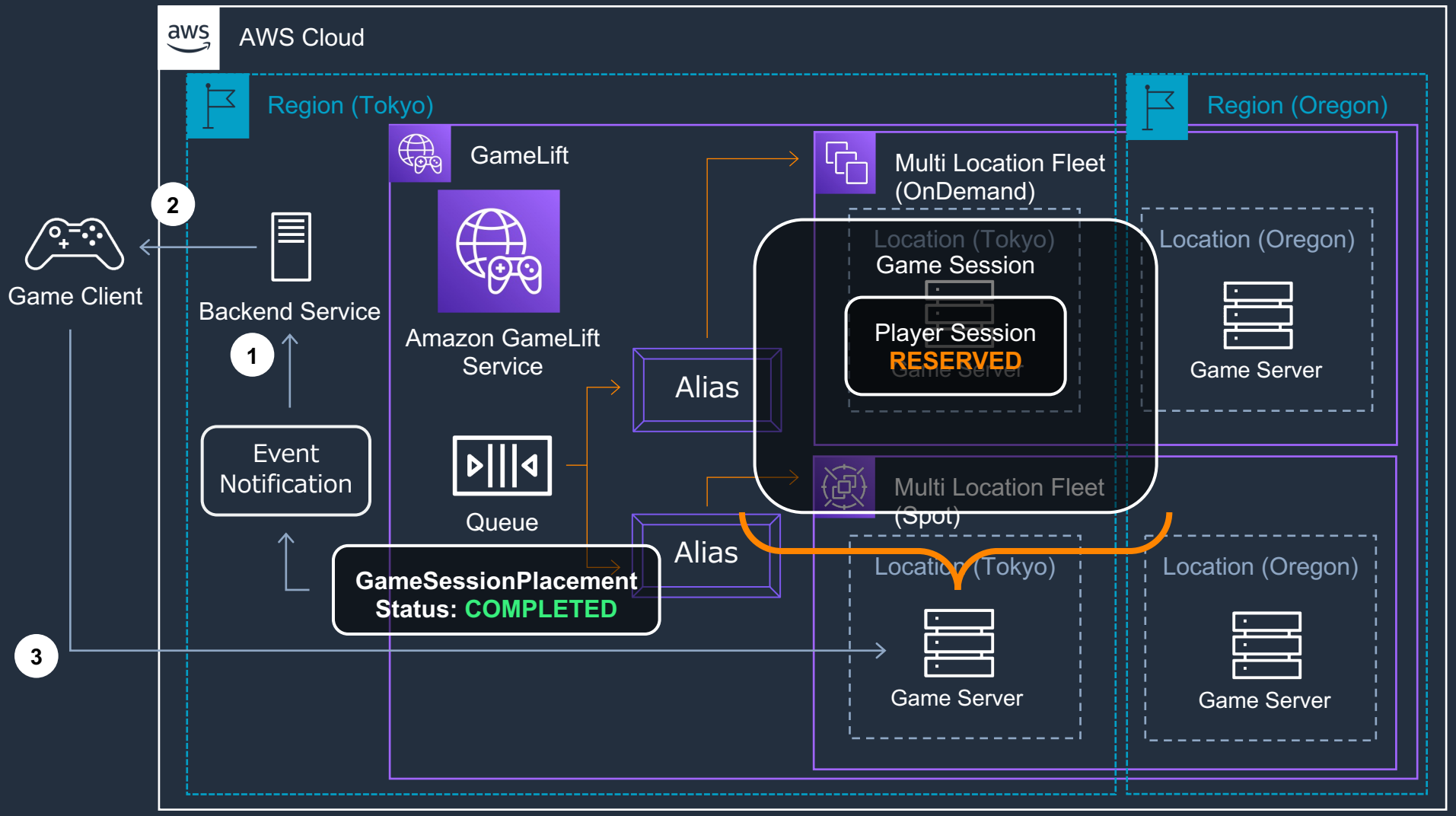
① バックエンドサービスが配置リクエストの結果を受け取る

- イベント通知内容を DB に保存
- バックエンドサービスが DB に格納された情報を参照する



ゲームセッションへのプレイヤー追加

ゲームセッションキューを利用してゲームセッションを配置する場合



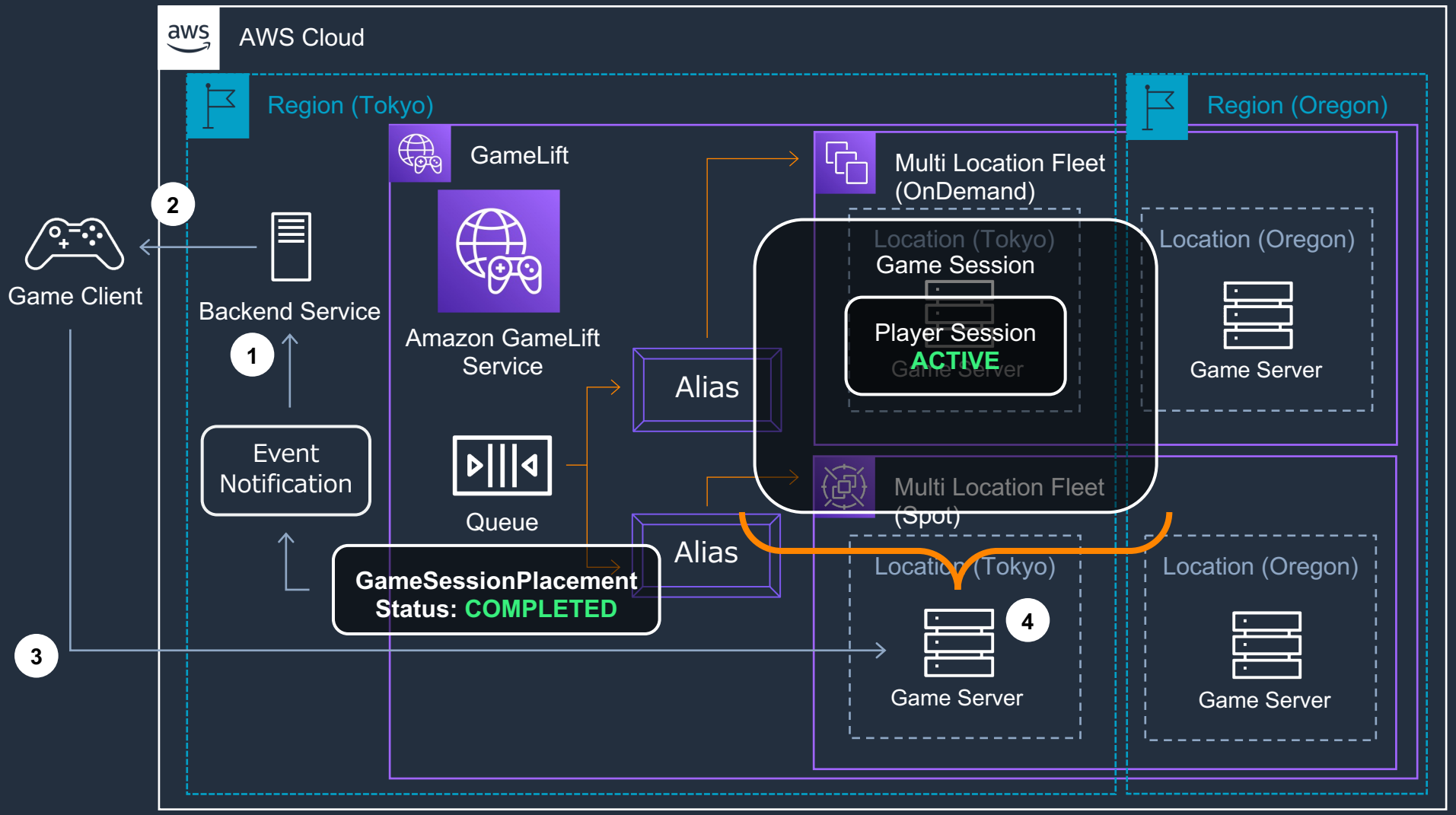
① ゲームクライアントがバックエンドサービスから配置リクエストの結果を受け取る

② ゲームクライアントに接続情報を渡す

③ 接続情報を使用しゲームサーバープロセスに直接接続する

ゲームセッションへのプレイヤー追加

ゲームセッションキューを利用してゲームセッションを配置する場合



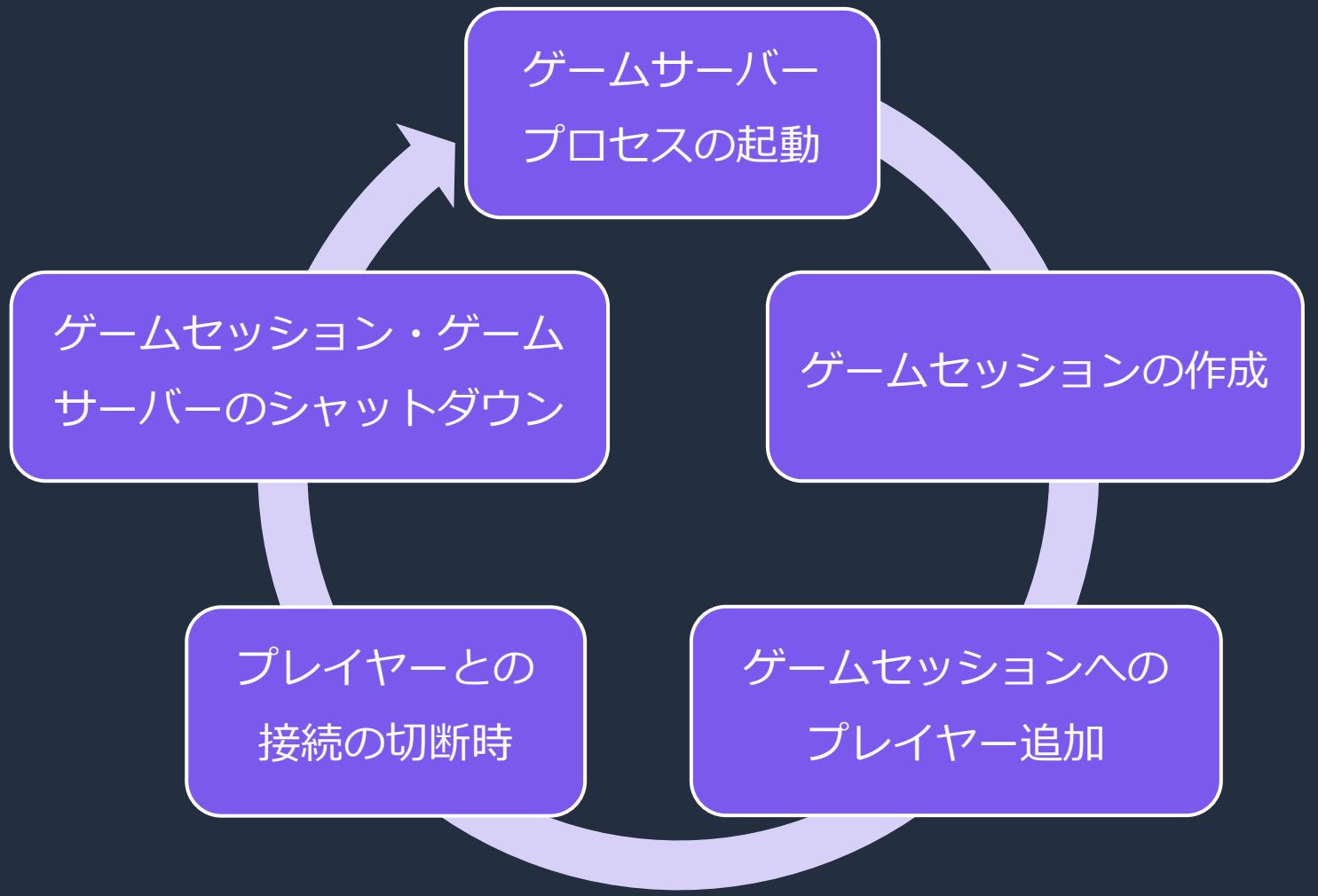
① ゲームクライアントがバックエンドサービスから配置リクエストの結果を受け取る

② ゲームクライアントに接続情報を渡す

③ 接続情報を使用しゲームサーバープロセスに直接接続する

④ ゲームサーバーが AcceptPlayerSession() を呼び出し、GameLift サービスでプレイヤーを検証する
検証後、プレイヤーセッションのステータスが “ACTIVE” に設定される

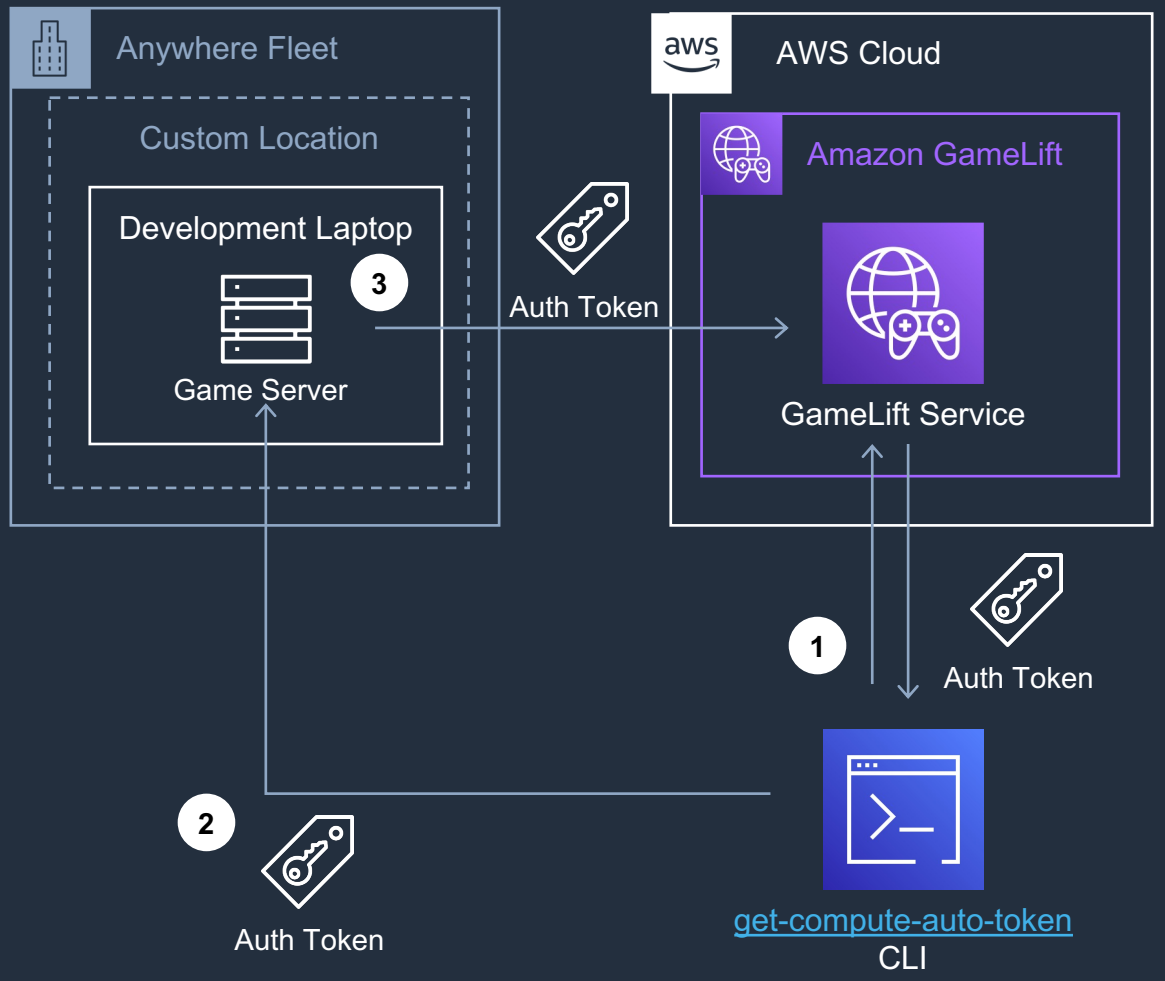
GameLift ゲームサーバーのライフサイクル



- Managed EC2 フリートへの接続
  - フリートとロケーションを直接指定する場合
  - ゲームセッションキューを利用する場合
- Anywhere フリートへの接続
  - 事前準備として登録したハードウェア上のゲームサーバープロセスを GameLift で認証する
    - フリートとロケーションを直接指定する場合
    - ゲームセッションキューを利用する場合

Anywhere フリートのゲームサーバーに接続する

事前準備として登録したハードウェア上のゲームサーバープロセスを GAMELIFT で認証する



① GetComputeAuthToken (CLI / API) を使用して認証トークンを取得

- Anywhere フリート ID と ComputeName を指定

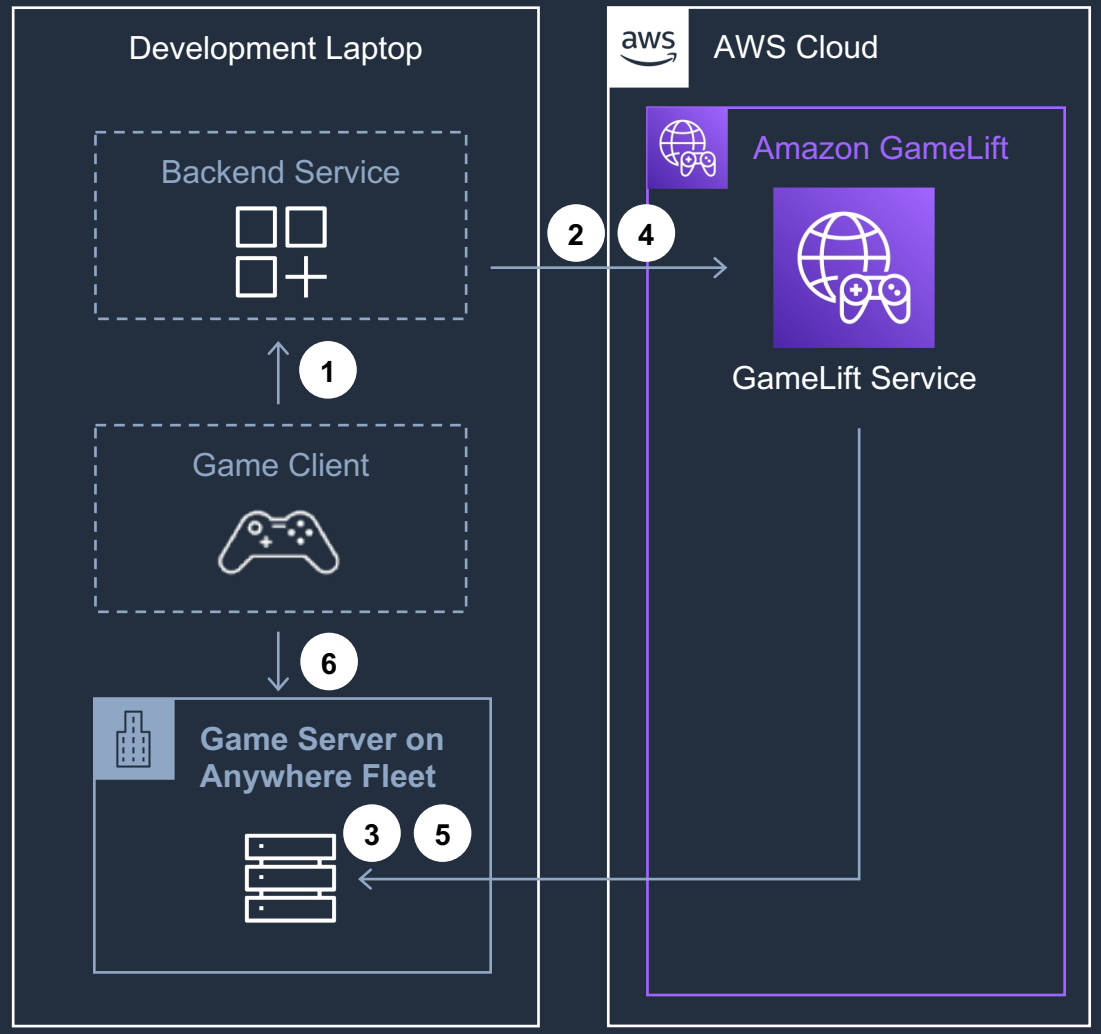
② 取得した認証トークンの情報をゲームサーバープロセスに渡す

③ ゲームサーバープロセス上で GameLift Server SDK v5.X の InitSDK() 関数の引数 ServerParameters に以下のプロパティを設定

プロパティ	説明
webSocketUrl	RegisterCompute の返却値に含まれる WebSocket URL
processId	その Compute 上で一意となる任意の文字列を指定
hostId	RegisterCompute で指定した ComputeName
fleetId	Anywhere フリートの ID
authToken	GetComputeAuthToken で取得した認証トークン

Anywhere フリートのゲームサーバーに接続する

フリートとロケーションを直接指定してゲームセッションを作成する場合



① バックエンドサービスを経由

② AWS SDK で CreateGameSession を Anywhere フリートを対象に呼び出し

③ Anywhere フリートのゲームサーバープロセス上にゲームセッションが作成される

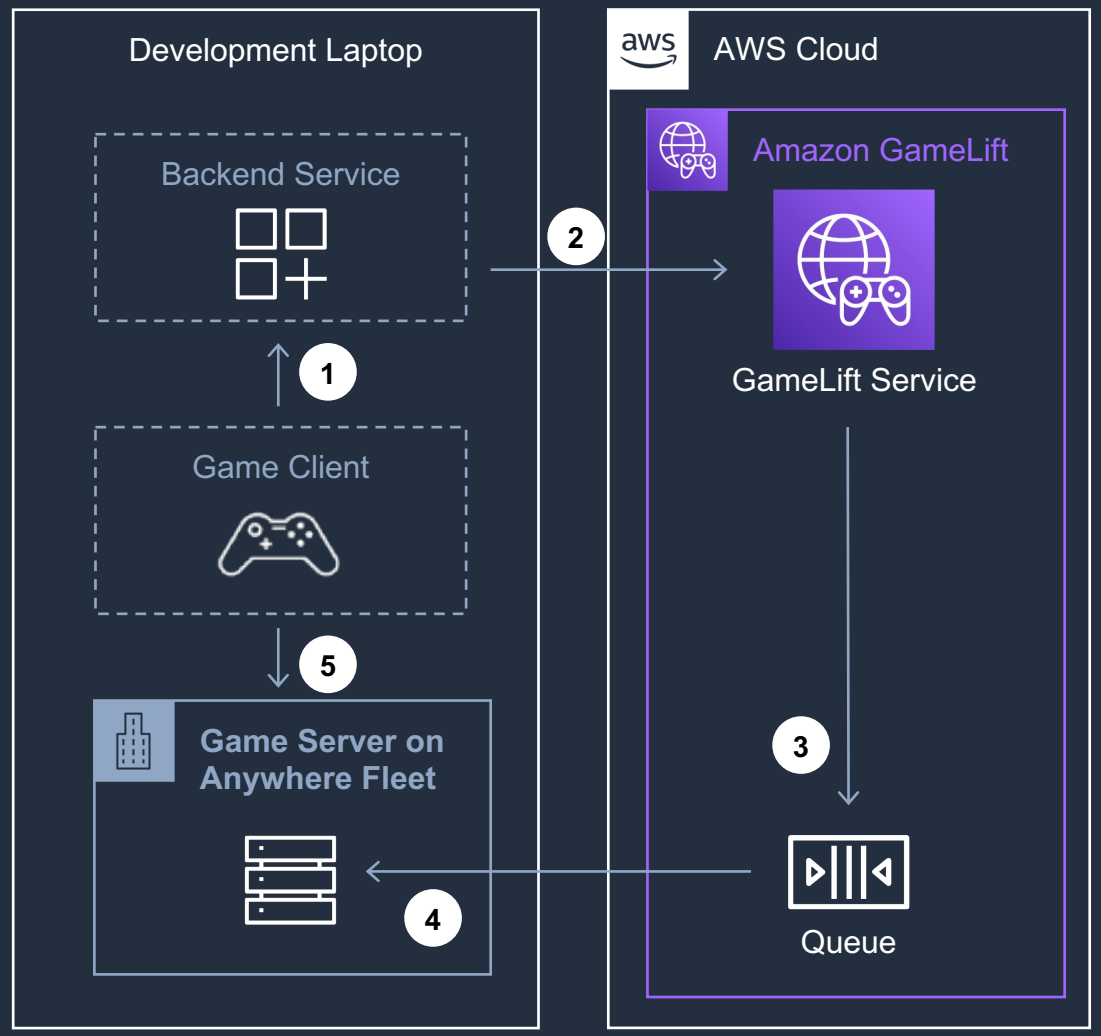
④ AWS SDK で CreatePlayerSession を ③で作成されたゲームセッションを対象に呼び出し

⑤ ③で作成されたゲームセッション上にプレイヤーセッションが作成される

⑥ バックエンドサービスから返却された接続先の情報を使用して Anywhere フリートのゲームサーバーに接続

Anywhere フリートのゲームサーバーに接続する

ゲームセッションキューを利用してゲームセッションを配置する場合



① バックエンドサービスを経由

② AWS SDK で StartGameSessionPlacement を呼び出し

③ FleetIQ アルゴリズムが
最適なゲームセッション配置先を選択

④ ゲームセッションと
プレイヤーセッションが作成される

⑤ ゲームセッションキューの
イベント通知経由で接続先の情報を受け取り
Anywhere フリートのゲームサーバーに接続

アジェンダ

1. マルチプレイヤーゲームにおける専用ゲームサーバーホスティングの課題
2. Amazon GameLift の概観
3. GameLift Managed Hosting の仕組み
4. GameLift Managed Hosting のリソース
5. GameLift ゲームサーバーのライフサイクル
6. 料金情報
7. まとめ

GameLift の料金 – Managed EC2

GAMELIFTインスタンスの稼働時間とデータ転送量に基づく従量制料金

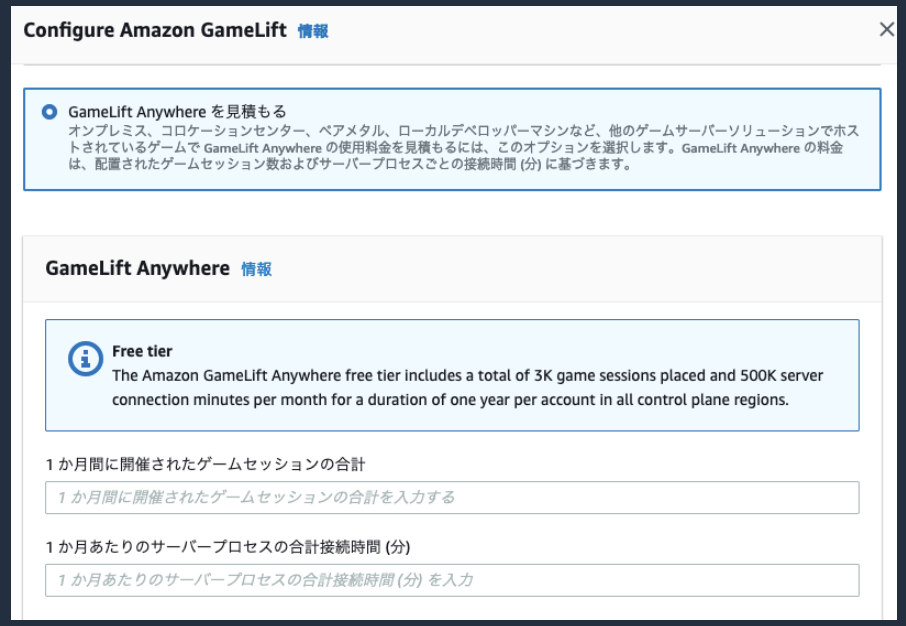
- GameLift インスタンスの稼働時間
  - Auto Scaling とスポットインスタンスでコストを削減
- データ転送量
  - ゲームクライアントとゲームサーバー間のアウトバウンドデータ転送に対して課金
- AWS Pricing Calculator で試算可能
- AWS 無料利用枠 – サインアップ後 12 ヶ月間まで
  - Amazon GameLift 向け
    c5.large オンデマンドインスタンス 125 時間/月
    EBS 汎用 (SSD) ストレージ 50 GB
  - AWS のすべてのサービスを合わせて、
    送信帯域幅 15 GB/月

2023/02 時点
https://aws.amazon.com/jp/GameLift/pricing/

GameLift の料金 – Anywhere

ゲームセッション配置回数とサーバープロセスの接続時間に基づく従量制料金

項目	費用	無料利用枠 – サインアップ後 12 ヶ月間まで
Anywhere フリートへの ゲームセッション配置数	$0.0012	全てのリージョンで合計 3,000 回 / month
Anywhere フリート上の サーバープロセスとの接続時間(分)	$0.000011 / minute	全てのリージョンで計 500,000 分 / month



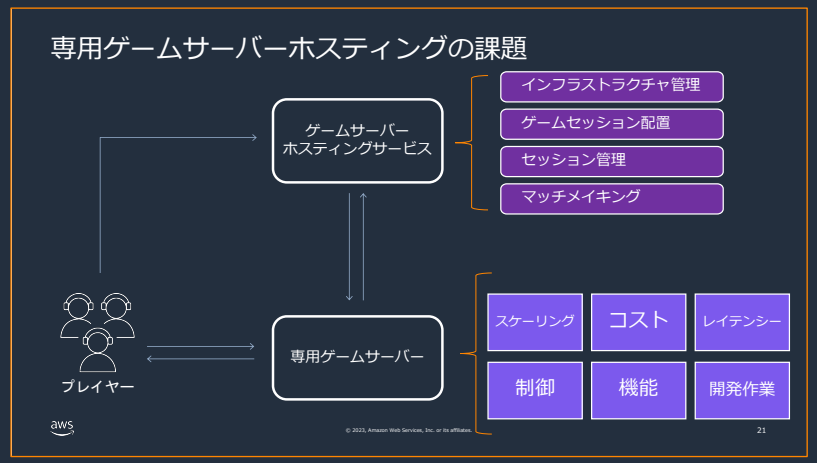
- AWS Pricing Calculator で試算可能
- Anywhere フリートに使用している
コンピューティングリソースの料金は含まない点に注意

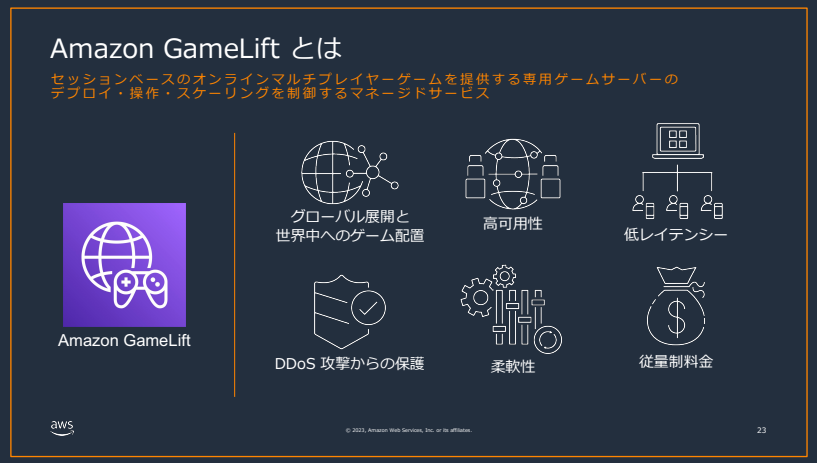
2023/02 時点
https://aws.amazon.com/jp/GameLift/pricing/

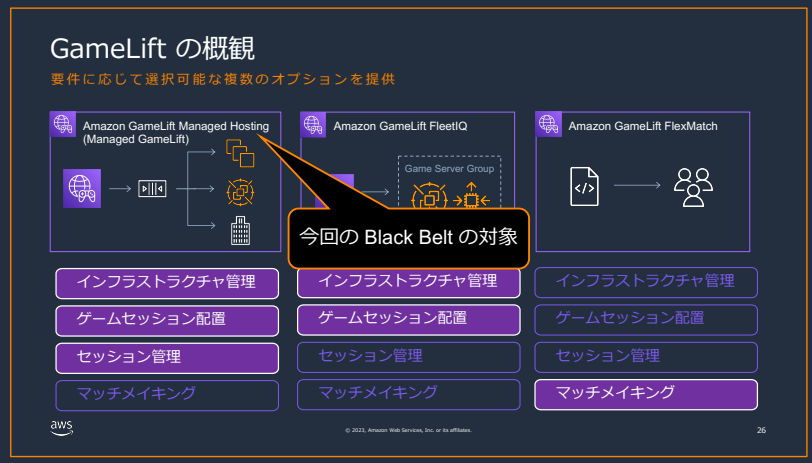
アジェンダ

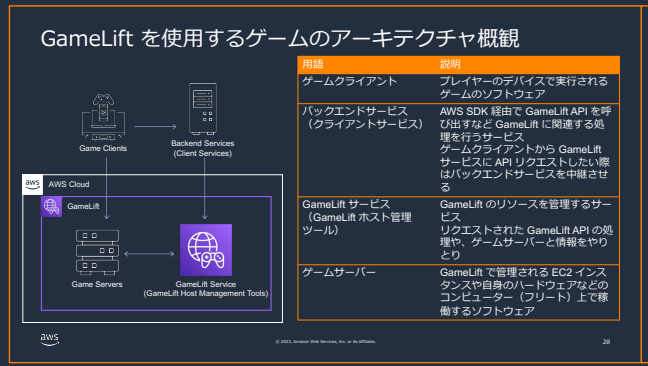
1. マルチプレイヤーゲームにおける専用ゲームサーバーホスティングの課題
2. Amazon GameLift の概観
3. GameLift Managed Hosting の仕組み
4. GameLift Managed Hosting のリソース
5. GameLift ゲームサーバーのライフサイクル
6. 料金情報
7. まとめ

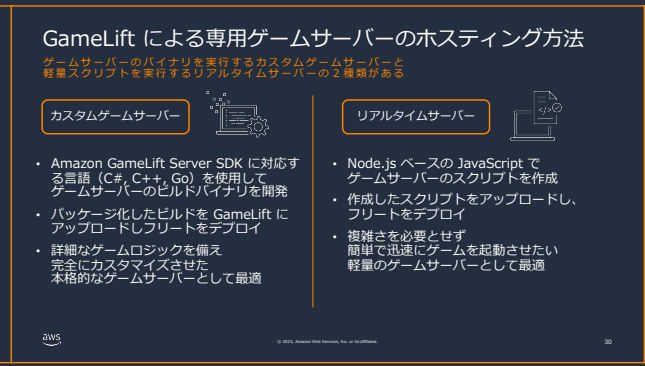
まとめ

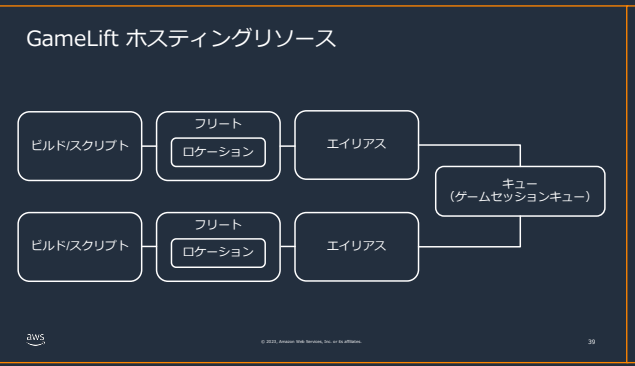


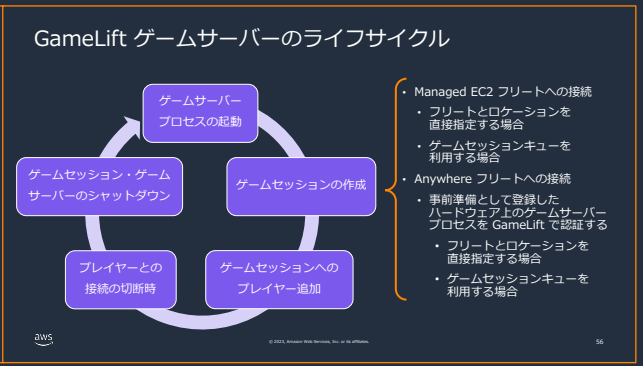












本資料に関するお問い合わせ・ご感想

技術的な内容に関しましては、有料のAWSサポート窓口へ
お問い合わせください

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

料金面でのお問い合わせに関しましては、カスタマーサポート窓口へ
お問い合わせください（マネジメントコンソールへのログインが必要です）

https://console.aws.amazon.com/support/home#/case/create?issueType=customer-service

具体的な案件に対する構成相談は、後述する個別相談会をご活用ください

ご感想はTwitterへ！ハッシュタグは以下をご利用ください
#awsblackbelt

その他コンテンツのご紹介

ウェビナーなど、AWSのイベントスケジュールをご参照いただけます

https://aws.amazon.com/jp/events/

ハンズオンコンテンツ

https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-hands-on/

AWS 個別相談会

AWSのソリューションアーキテクトと直接会話いただけます

https://pages.awscloud.com/JAPAN-event-SP-Weekly-Sales-Consulting-Seminar-2021-reg-event.html

Thank you!