AWS for Games

"Build the next great gaming experience"

Eren Akbaba

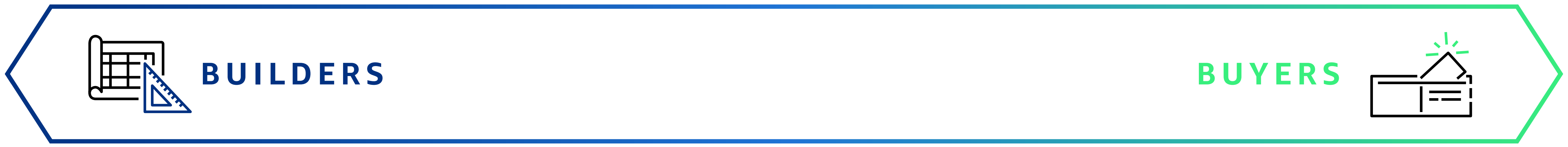
AWS for Games

AWS for Games solution areas

BUILD	RUN		GROW		
Cloud game development	Game servers	Game security	LiveOps	Game analytics	AI & ML

Solutions for builders and buyers

AWS FOR GAMES PROVIDES SOLUTIONS FOR OUR GAMES CUSTOMERS WHEREVER THEY ARE IN THE BUILDER–BUYER CONTINUUM



Native AWS services	Game-specific AWS services	AWS Solutions & Quick Starts	AWS ProServe & GSIs	AWS Partner Solutions & AWS Marketplace

AWS for Games solution areas

BUILD	RUN		GROW		
Cloud game development	Game servers	Game security	LiveOps	Game analytics	AI & ML

USE CASES AND SOLUTIONS

Workstations Build pipelines Version control 3D world building	Hosting session-based games Global game infrastructure	Defend against DDOS attacks Protect against data breaches	Game Services for LiveOps	Centralized game analytics	Community health/toxicity Smart acquisition and retention

AWS for Games solution library

BUILD

Cloud game development

RUN

Game servers

Game security

GROW

LiveOps

Game analytics

AI & ML

AWS SERVICES
Purpose-built cloud products

AWS SOLUTIONS
Ready-to-deploy solutions assembling AWS Services, code, and configurations

PARTNER SOLUTIONS
Software, SaaS, of managed services from AWS Partners

GUIDANCE
Prescriptive architectural diagrams, sample code, and technical support

AWS for Games launch APN partners

ACCELBYTE AccelerateXR AMD AppsFlyer AUTODESK BEAMABLE Cockroach Labs

CODE WIZARDS Couchbase CROWDSTRIKE databricks epam EPIC GAMES GameAnalytics

Globant honeycomb.io JET BRAINS JFrog LACEWORK logz.io NVIDIA

PARSEC PERFORCE revolgy six nines spectrum labs sumo logic TensorWorks

teradici EPIC GAMES ONLINE SERVICES Heroic Labs INCREDIBUILD NASUNI New Relic slalom TEKsystems Global Services

Amazon and AWS Game Services

Amazon GameLift
Dedicated game server hosting solution for multiplayer games

GameLift FlexMatch
Matchmaking for multi-player games

Open 3D Engine (O3DE)
AAA-capable, cross-platform, open-source game engine available under an Apache 2.0 license

Amazon Twitch
145M monthly unique visitors,
25.6M daily unique visitors,
1.6B monthly live hours

Amazon Luna
Game streaming service that allows you to play games on devices you already own

Amazon Prime Gaming
Activate 150M+ Prime members in your game

Overview

Amazon GameLift

Deploy, operate, and scale dedicated, low-cost servers for session-based, multiplayer games

Low-cost game servers

Low latency

High availability

Flexibility

Overview

Amazon GameLift

Multiplayer game server entry points

Fully-managed GameLift

Fully managed, vertically integrated game server stack for dedicated game servers

GameLift Realtime

Lightweight, ready-to-go game servers well-suited for mobile games

GameLift FleetIQ

Server management layer to help developers optimize costs on Amazon EC2 with Spot

GameLift Anywhere

Session management layer to help developers looking for compute flexibility **(Beta)**

Overview

Amazon GameLift

Select the features you need

31 Regions

Fully managed game servers hosted across AWS Regions on multiple Availability and Local Zones ensuring availability and scalability

Instances

Scale automatically based on player demand with 60+ instance types supported on Windows or Linux

SDKs

C++ and C# SDKs to develop GameLift-enabled multiplayer game servers, game clients and game services

Game Engines

Support for Unity and Unreal Engines with regular SDK updates for engine versions, binary plugins, and CloudFormation integrations

Amazon GameLift

A dedicated game server hosting solution that deploys, operates, and scales cloud servers for multiplayer games.

Low cost game servers

Scale game servers automatically based on player demand and leverage low cost Spot Instances for short-lived game sessions

Low Latency

Deploy a single fleet globally and leverage built-in latency-based matchmaking

High Availability

Fully managed game servers hosted across AWS Regions on multiple Availability Zone, as well as Local Zones, providing availability and scalability

Flexibility

Select the features you need. Use the built-in matchmaking or build your own. Select between fully managed and self-managed hosting

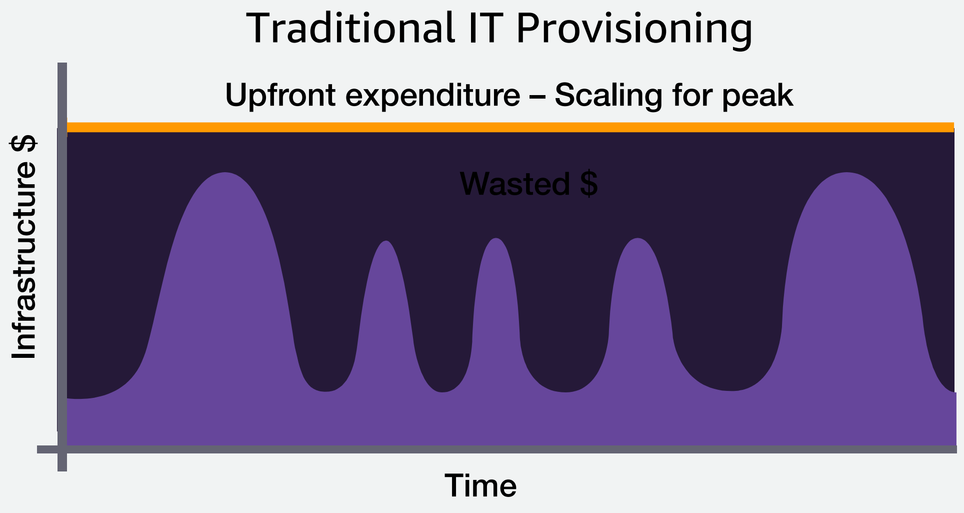
Amazon GameLift

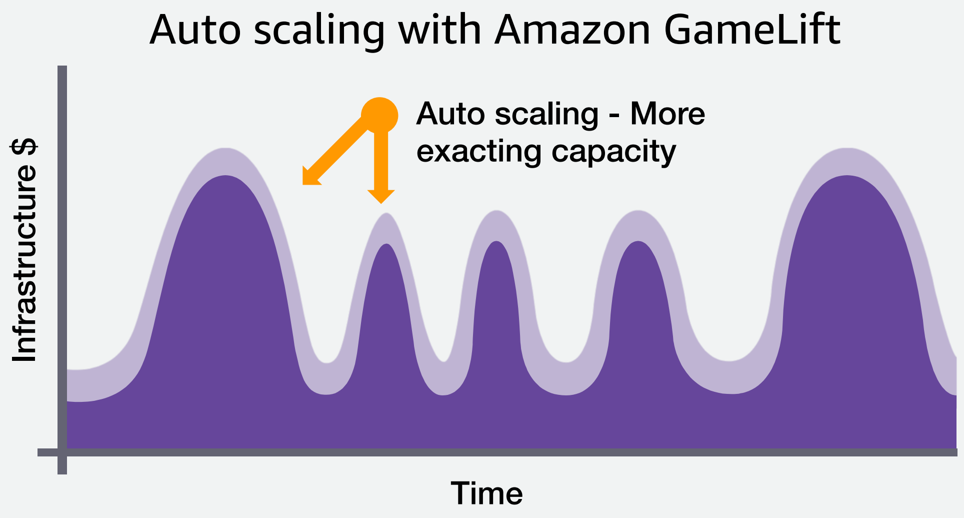
Infrastructure Management

- Auto-scaling
- High-availability (Multi-AZ)
- Automatic cross-region failover

Session Management

- Player & Game sessions
- Worldwide placement
- Matchmaking (FlexMatch)





Two service options for hosting

Fully-managed Amazon GameLift

- Fully managed game server hosting
- Upload your build once, deploy globally
- Managed latency-based session placement
- Supports on-demand and spot fleets for cost optimization with spot viability algorithm
- Supports C++, C#, Unity & Unreal
- Native integration with GameLift FlexMatch for matchmaking

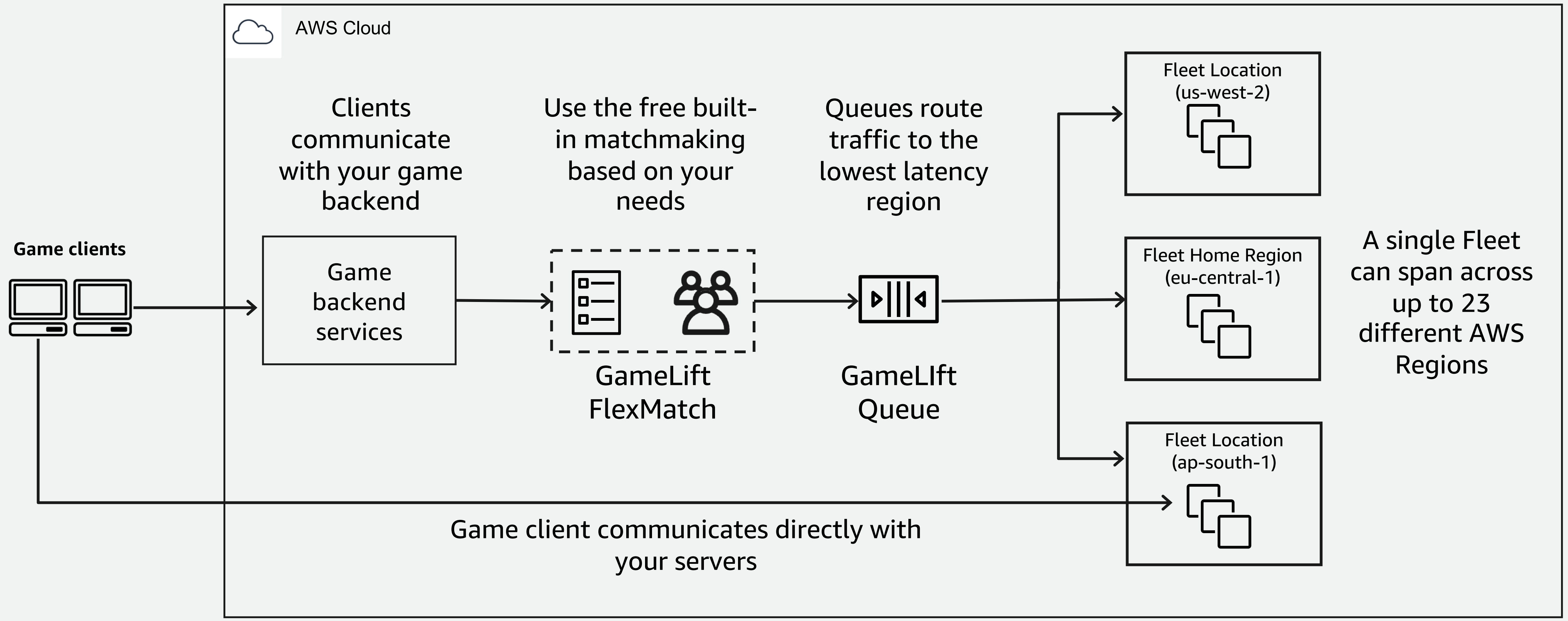
Fully Managed

Amazon GameLift FleetIQ

- Flexible: Hosted on EC2 on your AWS account
- Simple API layer for game session management
- Use your existing tools and software for deploying and running game servers
- Access GameLift Spot viability algorithm independent of other GameLift features
- AWS SDK on the server side: use the programming language of your choice

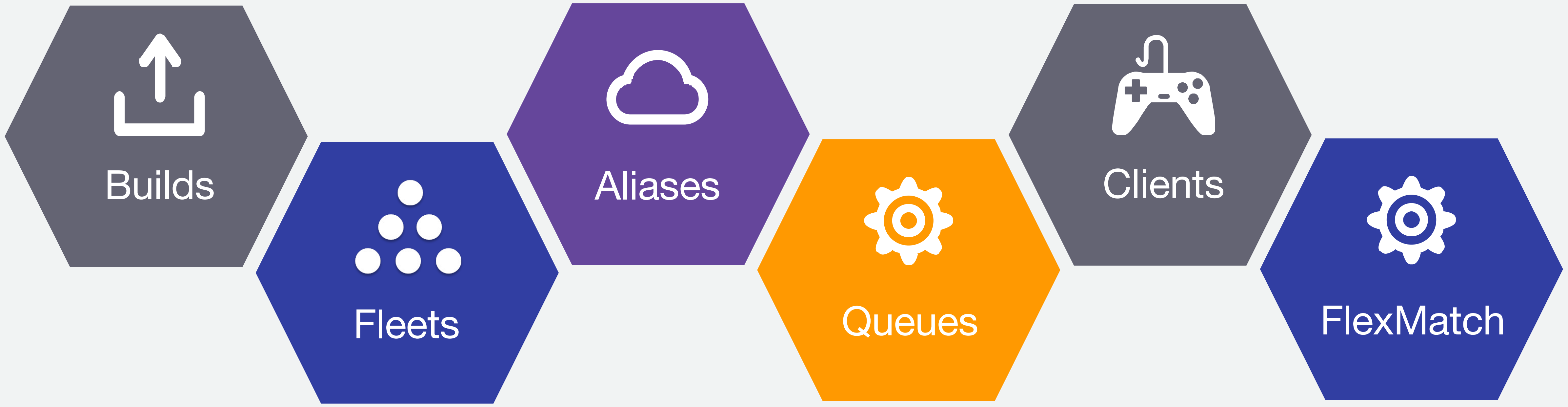
More Flexibility

End-to-End High Level Integration Flow



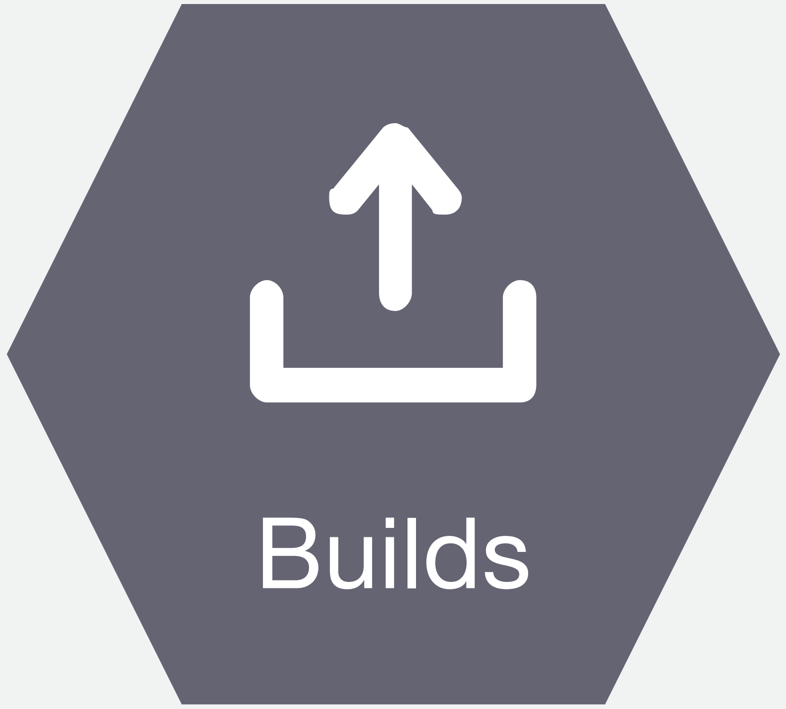
Game clients
AWS Cloud
Clients communicate with your game backend
Game backend services
Use the free built-in matchmaking based on your needs
GameLift FlexMatch
Queues route traffic to the lowest latency region
GameLIft Queue
Fleet Location (us-west-2)
Fleet Home Region (eu-central-1)
Fleet Location (ap-south-1)
A single Fleet can span across up to 23 different AWS Regions
Game client communicates directly with your servers

Amazon GameLift Components

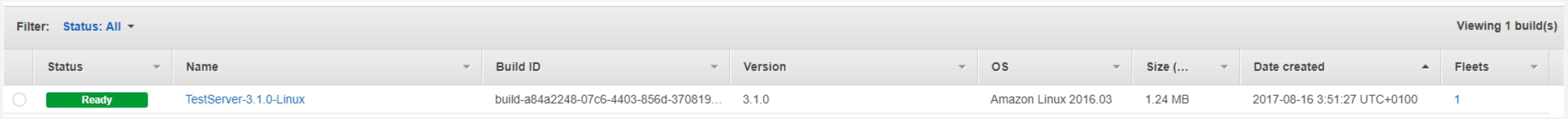


Builds Explained

- Upload via AWS CLI or SDK
- Store multiple builds
- Define Install script for custom instance configuration
- Supported Operating Systems:
    - Amazon Linux
    - Windows
- Server SDKs:
    - C++
    - C#
    - Unreal Plugin (SDK)
    - Unity Plugin (Fully-featured)



Build Dashboard



Filter: Status: All

Viewing 1 build(s)

	Status	Name	Build ID	Version	OS	Size (...	Date created	Fleets
○	Ready	TestServer-3.1.0-Linux	build-a84a2248-07c6-4403-856d-370819...	3.1.0	Amazon Linux 2016.03	1.24 MB	2017-08-16 3:51:27 UTC+0100	1

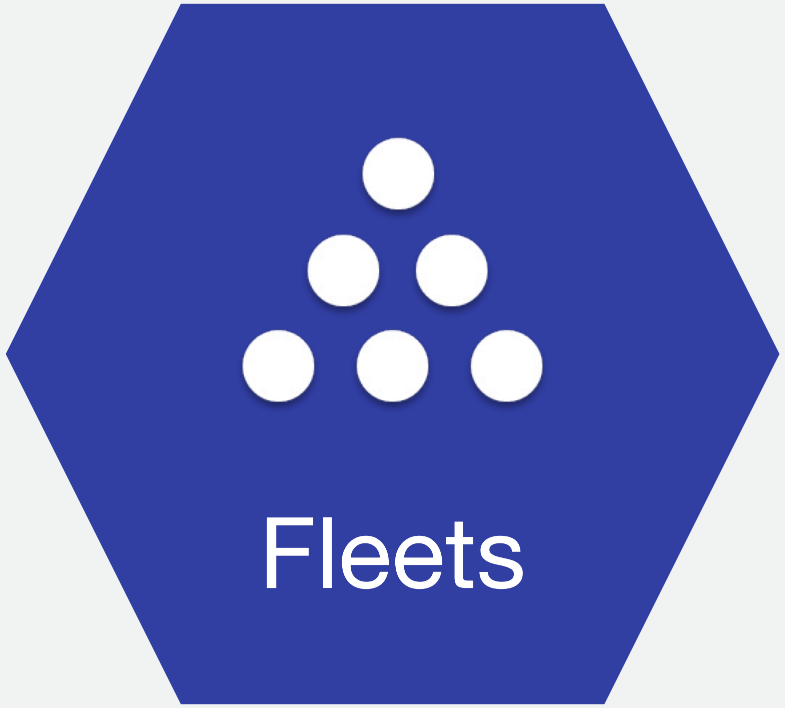
Access key information about builds stored in Amazon GameLift:

- Status
- Version
- OS
- Size
- Number of fleets using this build

Fleets Explained

Fleets represent the deployed state of your game build. They enable you to define how your build should be hosted within Amazon GameLift including:

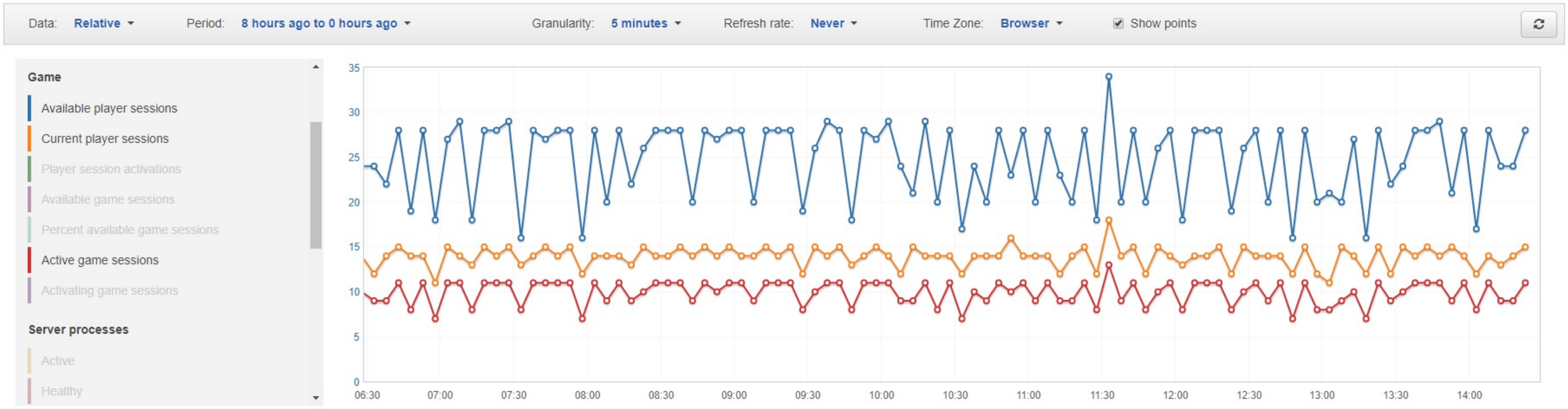
- Instance Type
- Home Region and additional Regions (multi-region Fleets)
- Scaling Policies
- Capacity Limits
- Number of processes per instance (up to 50)



Fleet Metrics

- Game/Player Sessions
- Game Server Processes

- Number of Instances
- Instance Performance
- Scaling Limits


Data: Relative ▾ Period: 8 hours ago to 0 hours ago ▾ Granularity: 5 minutes ▾ Refresh rate: Never ▾ Time Zone: Browser ▾ ✓ Show points

Game
- Available player sessions
- Current player sessions
- Player session activations
- Available game sessions
- Percent available game sessions
- Active game sessions
- Activating game sessions

Server processes
- Active
- Healthy

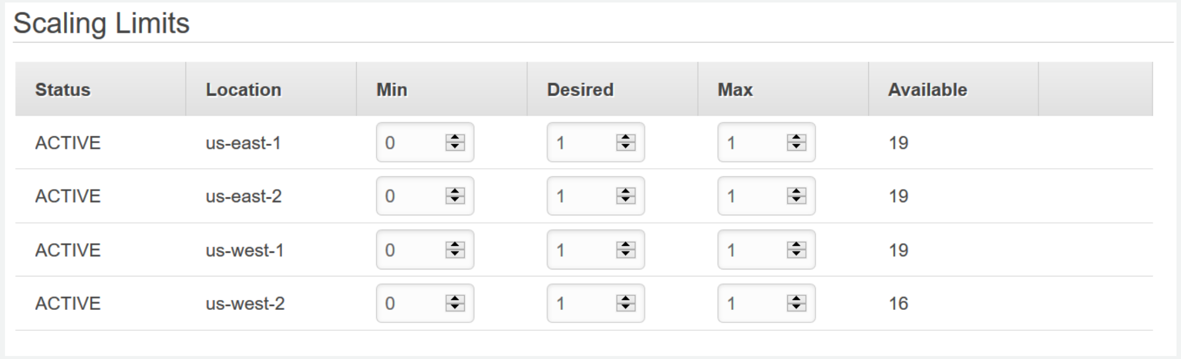
Target tracking auto-scaling

Amazon GameLift can automatically maintain a defined level of session availability for your game servers.

GameLift determines the exact number of server instances to add or remove as it tracks toward a target buffer percentage.

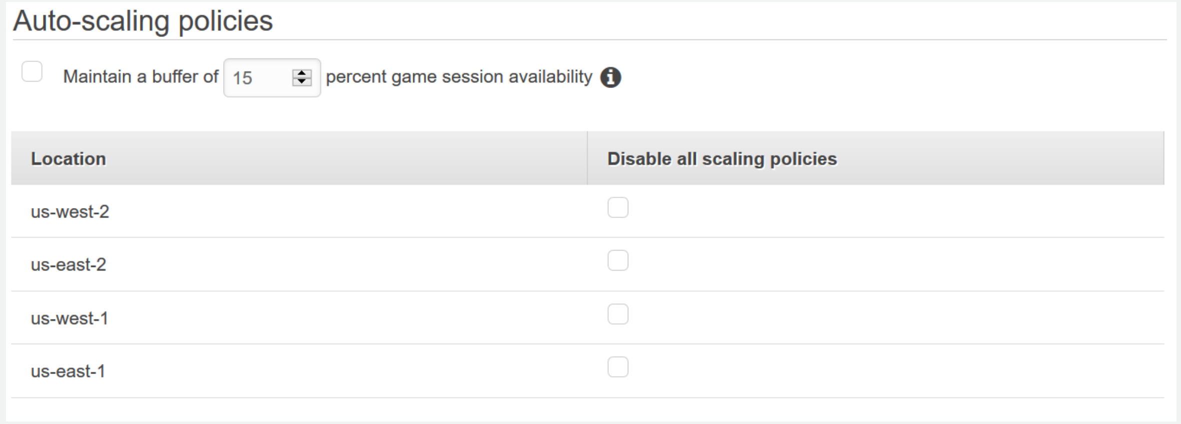
When player demand for a game is rising, new server instances are automatically provisioned

When demand subsides, server instances are automatically scaled down (with protection on running game sessions, when enabled on Fleet level)



Scaling Limits

Status	Location	Min	Desired	Max	Available
ACTIVE	us-east-1	0	1	1	19
ACTIVE	us-east-2	0	1	1	19
ACTIVE	us-west-1	0	1	1	19
ACTIVE	us-west-2	0	1	1	16



Auto-scaling policies

☐ Maintain a buffer of 15 percent game session availability

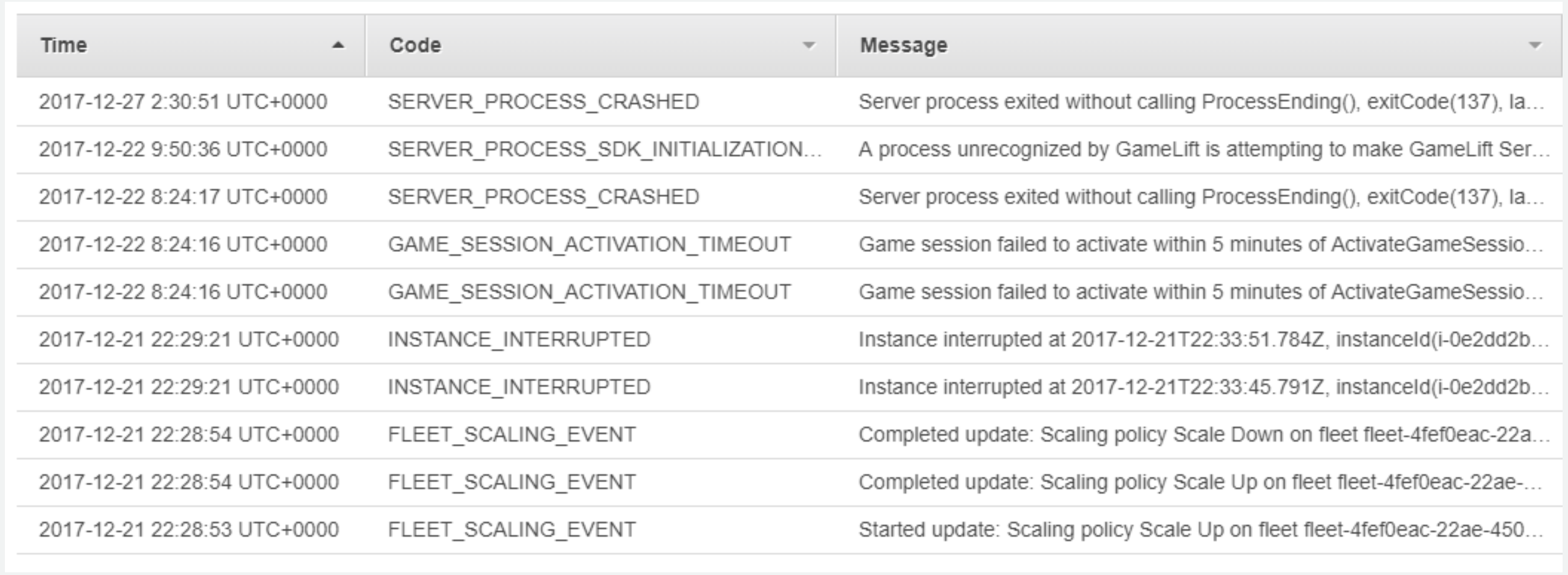
Location	Disable all scaling policies
us-west-2	☐
us-east-2	☐
us-west-1	☐
us-east-1	☐

Fleet Events

See events that take place in the fleet at the game session or instance level.

Events reported include:

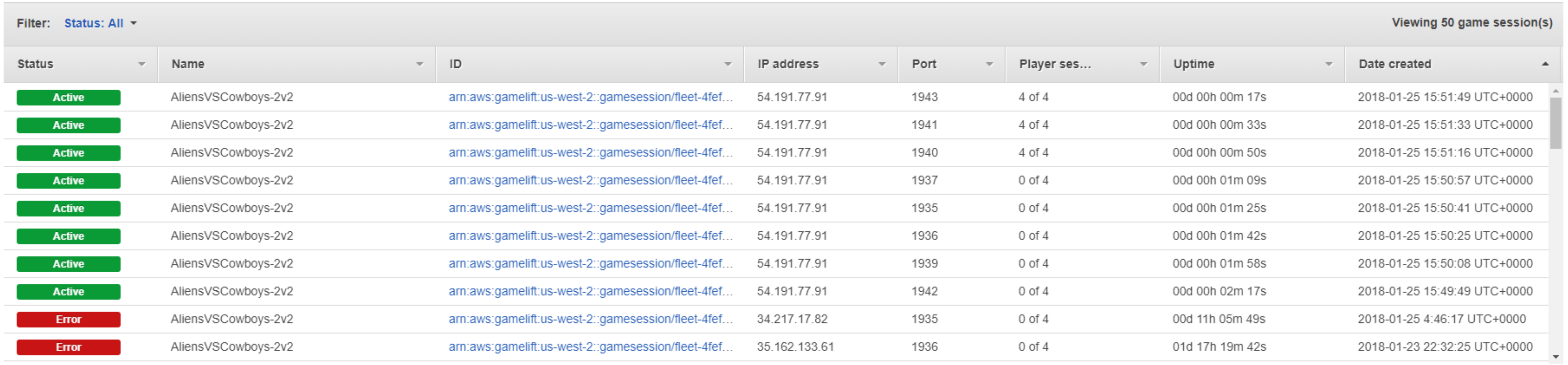
- Scaling
- Information
- Warnings
- Errors
- Crashes



Time	Code	Message
2017-12-27 2:30:51 UTC+0000	SERVER_PROCESS_CRASHED	Server process exited without calling ProcessEnding(), exitCode(137), la...
2017-12-22 9:50:36 UTC+0000	SERVER_PROCESS_SDK_INITIALIZATION...	A process unrecognized by GameLift is attempting to make GameLift Ser...
2017-12-22 8:24:17 UTC+0000	SERVER_PROCESS_CRASHED	Server process exited without calling ProcessEnding(), exitCode(137), la...
2017-12-22 8:24:16 UTC+0000	GAME_SESSION_ACTIVATION_TIMEOUT	Game session failed to activate within 5 minutes of ActivateGameSessio...
2017-12-22 8:24:16 UTC+0000	GAME_SESSION_ACTIVATION_TIMEOUT	Game session failed to activate within 5 minutes of ActivateGameSessio...
2017-12-21 22:29:21 UTC+0000	INSTANCE_INTERRUPTED	Instance interrupted at 2017-12-21T22:33:51.784Z, instanceId(i-0e2dd2b...
2017-12-21 22:29:21 UTC+0000	INSTANCE_INTERRUPTED	Instance interrupted at 2017-12-21T22:33:45.791Z, instanceId(i-0e2dd2b...
2017-12-21 22:28:54 UTC+0000	FLEET_SCALING_EVENT	Completed update: Scaling policy Scale Down on fleet fleet-4fef0eac-22a...
2017-12-21 22:28:54 UTC+0000	FLEET_SCALING_EVENT	Completed update: Scaling policy Scale Up on fleet fleet-4fef0eac-22ae-...
2017-12-21 22:28:53 UTC+0000	FLEET_SCALING_EVENT	Started update: Scaling policy Scale Up on fleet fleet-4fef0eac-22ae-450...

Fleet – Game Sessions

Track running game sessions running in a fleet in real time.

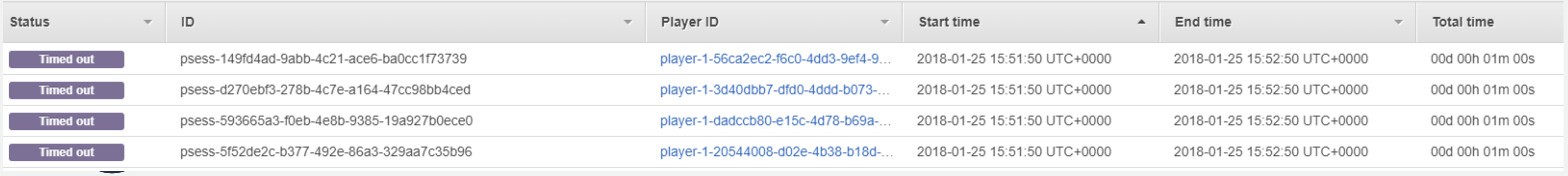


Filter: Status: All

Viewing 50 game session(s)

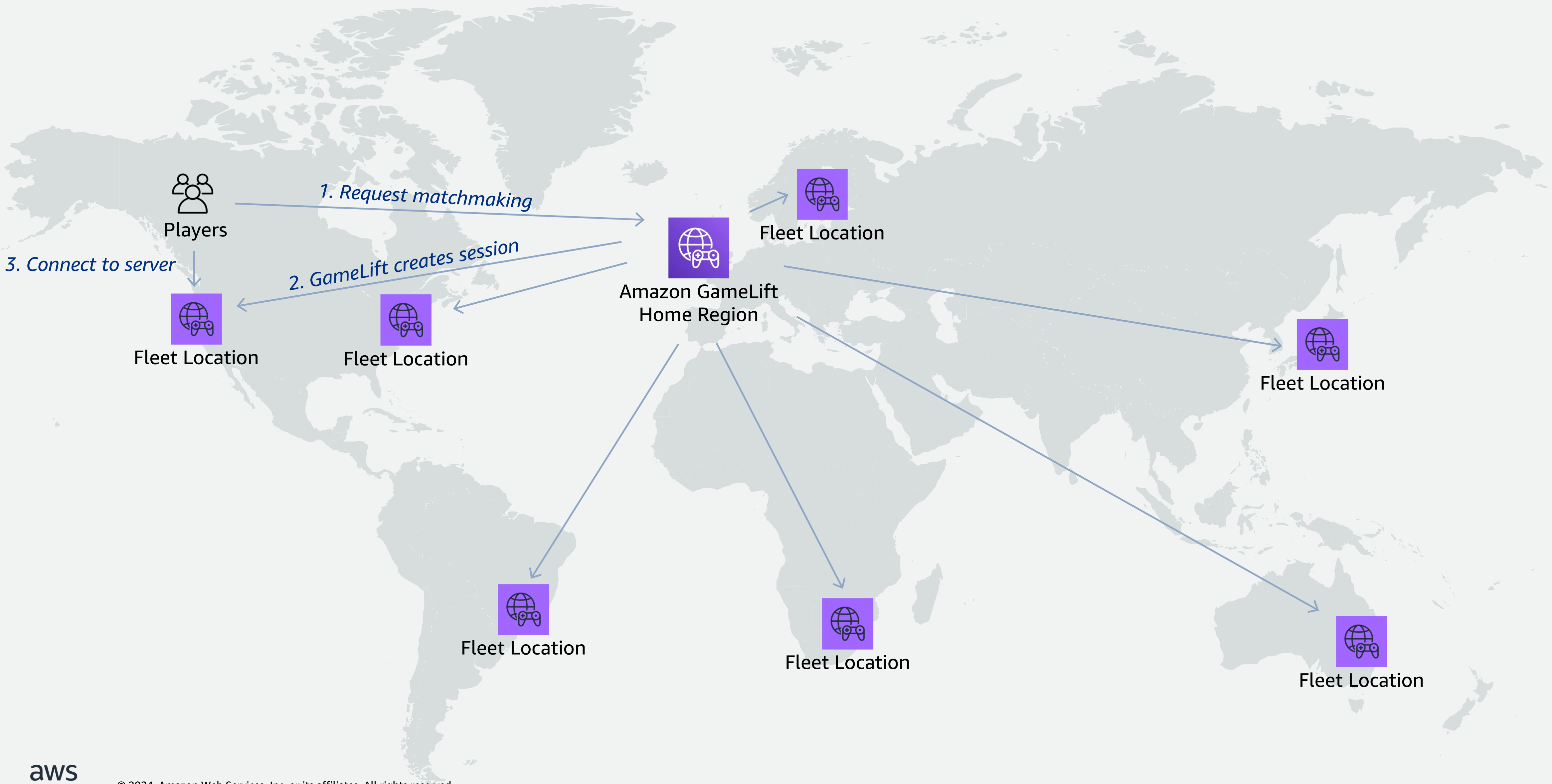
Status	Name	ID	IP address	Port	Player ses...	Uptime	Date created
Active	AliensVSCowboys-2v2	arn:aws:gamelift:us-west-2::gamesession/fleet-4fef...	54.191.77.91	1943	4 of 4	00d 00h 00m 17s	2018-01-25 15:51:49 UTC+0000
Active	AliensVSCowboys-2v2	arn:aws:gamelift:us-west-2::gamesession/fleet-4fef...	54.191.77.91	1941	4 of 4	00d 00h 00m 33s	2018-01-25 15:51:33 UTC+0000
Active	AliensVSCowboys-2v2	arn:aws:gamelift:us-west-2::gamesession/fleet-4fef...	54.191.77.91	1940	4 of 4	00d 00h 00m 50s	2018-01-25 15:51:16 UTC+0000
Active	AliensVSCowboys-2v2	arn:aws:gamelift:us-west-2::gamesession/fleet-4fef...	54.191.77.91	1937	0 of 4	00d 00h 01m 09s	2018-01-25 15:50:57 UTC+0000
Active	AliensVSCowboys-2v2	arn:aws:gamelift:us-west-2::gamesession/fleet-4fef...	54.191.77.91	1935	0 of 4	00d 00h 01m 25s	2018-01-25 15:50:41 UTC+0000
Active	AliensVSCowboys-2v2	arn:aws:gamelift:us-west-2::gamesession/fleet-4fef...	54.191.77.91	1936	0 of 4	00d 00h 01m 42s	2018-01-25 15:50:25 UTC+0000
Active	AliensVSCowboys-2v2	arn:aws:gamelift:us-west-2::gamesession/fleet-4fef...	54.191.77.91	1939	0 of 4	00d 00h 01m 58s	2018-01-25 15:50:08 UTC+0000
Active	AliensVSCowboys-2v2	arn:aws:gamelift:us-west-2::gamesession/fleet-4fef...	54.191.77.91	1942	0 of 4	00d 00h 02m 17s	2018-01-25 15:49:49 UTC+0000
Error	AliensVSCowboys-2v2	arn:aws:gamelift:us-west-2::gamesession/fleet-4fef...	34.217.17.82	1935	0 of 4	00d 11h 05m 49s	2018-01-25 4:46:17 UTC+0000
Error	AliensVSCowboys-2v2	arn:aws:gamelift:us-west-2::gamesession/fleet-4fef...	35.162.133.61	1936	0 of 4	01d 17h 19m 42s	2018-01-23 22:32:25 UTC+0000

Drill down to see player session information.



Status	ID	Player ID	Start time	End time	Total time
Timed out	psess-149fd4ad-9abb-4c21-ace6-ba0cc1f73739	player-1-56ca2ec2-f6c0-4dd3-9ef4-9...	2018-01-25 15:51:50 UTC+0000	2018-01-25 15:52:50 UTC+0000	00d 00h 01m 00s
Timed out	psess-d270ebf3-278b-4c7e-a164-47cc98bb4ced	player-1-3d40dbb7-dfd0-4ddd-b073-...	2018-01-25 15:51:50 UTC+0000	2018-01-25 15:52:50 UTC+0000	00d 00h 01m 00s
Timed out	psess-593665a3-f0eb-4e8b-9385-19a927b0ece0	player-1-dadccb80-e15c-4d78-b69a-...	2018-01-25 15:51:50 UTC+0000	2018-01-25 15:52:50 UTC+0000	00d 00h 01m 00s
Timed out	psess-5f52de2c-b377-492e-86a3-329aa7c35b96	player-1-20544008-d02e-4b38-b18d-...	2018-01-25 15:51:50 UTC+0000	2018-01-25 15:52:50 UTC+0000	00d 00h 01m 00s

Multi-region Fleets enable global deployments



Amazon GameLift Spot Fleets

Spot Fleets allow you to run your game servers in Amazon GameLift at a lower hourly rate. Optimal for shorter game sessions.

- Prices adjust based on demand for instance capacity
- Built-in algorithm estimates the viability to host game servers on Spot
- Linux and Windows instances available
- Instance interruptions can be managed via a notification mechanism (*onProcessTerminate*)
- On-demand Fleets can be used as a failover when spot availability is low

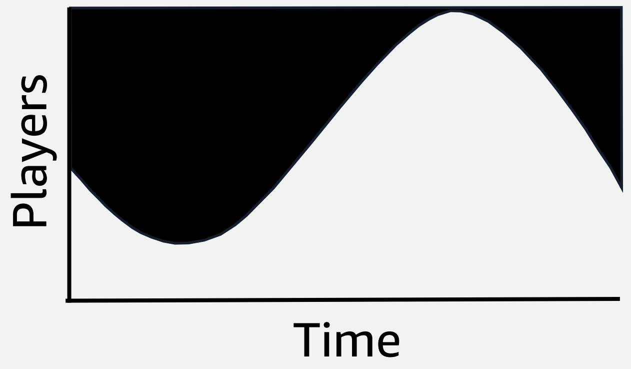
Cost optimization with Fully-Managed GameLift

Target based scaling

Your fleet will always run exactly the amount of instances you need

Define a single configuration for available game sessions

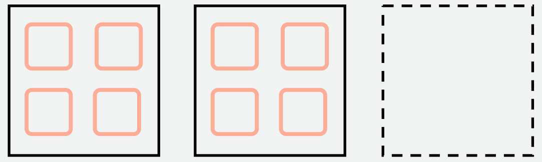
Pay for only the resources you use



Fully utilized servers

Run up to 50 game sessions on a single fleet instance

GameLift will automatically maximize the use of your instances



Spot Instances

Leverage unused EC2 capacity for a high discount on compute

Built-in algorithm checks Spot viability to minimize interruptions

Failover to on-demand fleets when spot availability is low

Optimal for shorter session lengths

Aliases Explained

Aliases allow you to redirect game clients to a fleet that you specify or to notify clients that a fleet is out of service.

There are two types of Alias available:

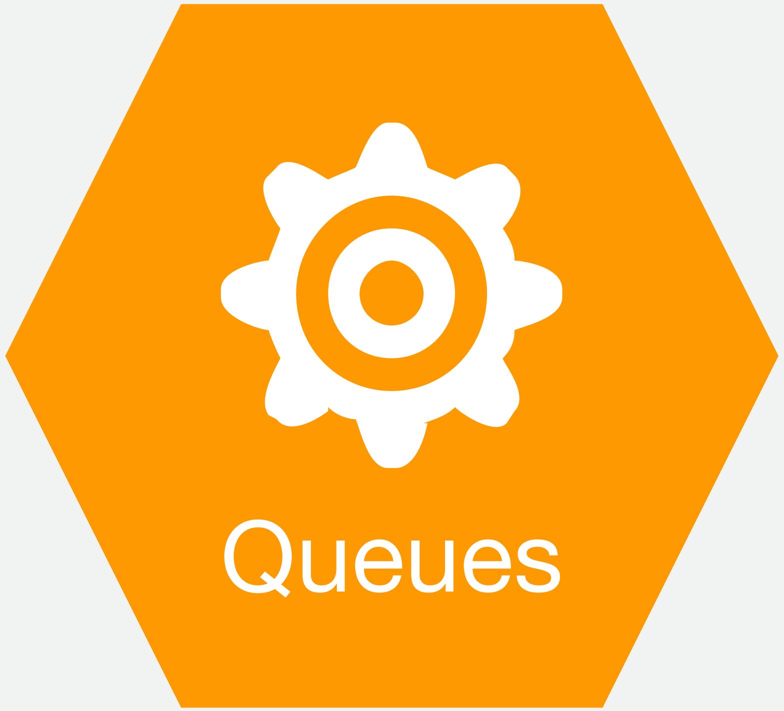
- **Simple** – A simple redirect points to an associated fleet, the fleet an Alias resolves to can be updated at any time
- **Terminal** – This does not resolve to a fleet, instead it passes back a specified message back to the client.

Queues Explained

Queues automate the process of efficiently allocating new game sessions on any fleet in a group.

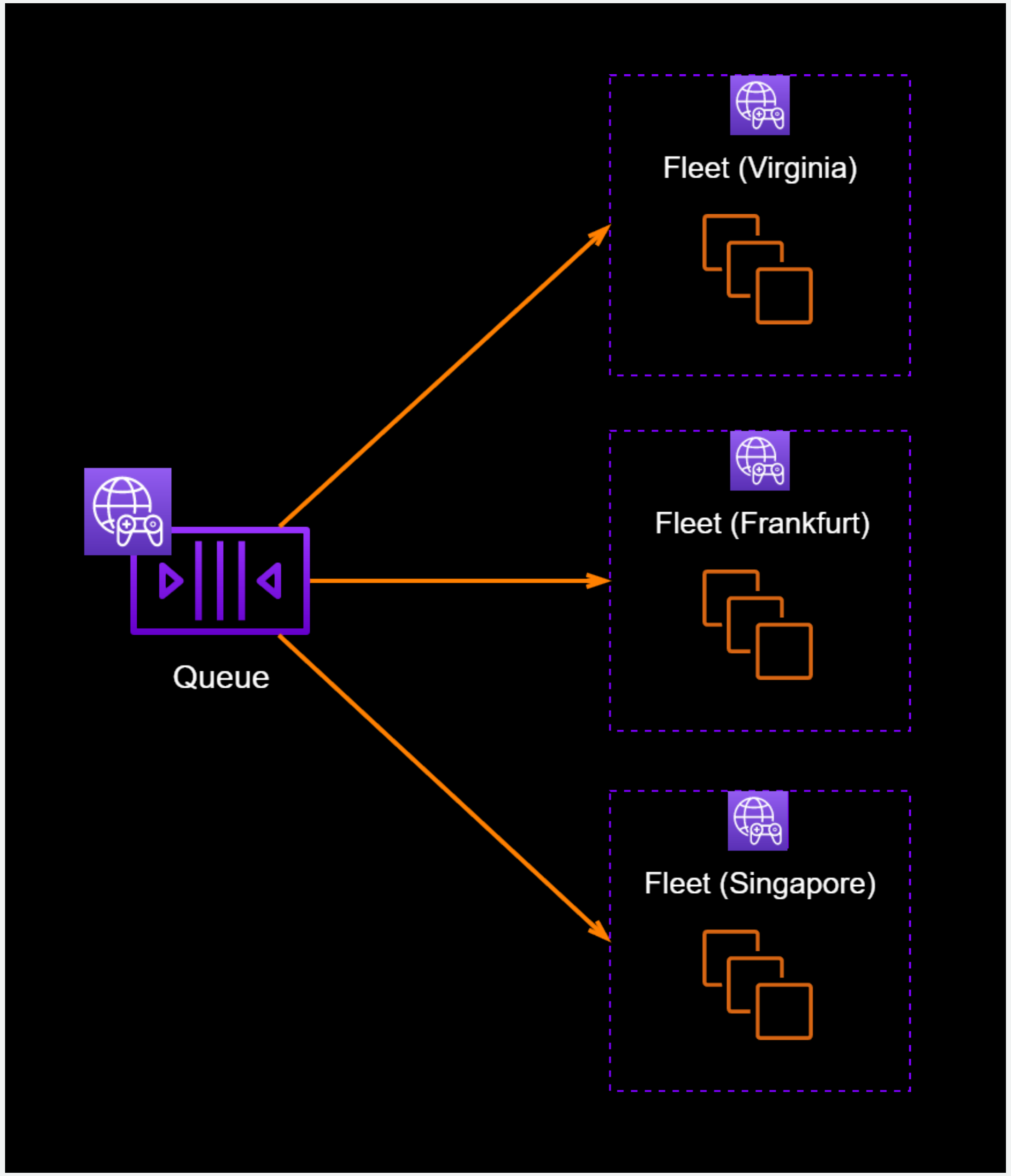
Queues are configurable and allow you to determine several options that are used when Amazon GameLift places a session:

- Queue Timeout
- Member Fleets
- Fleet Priority
- Latency policies



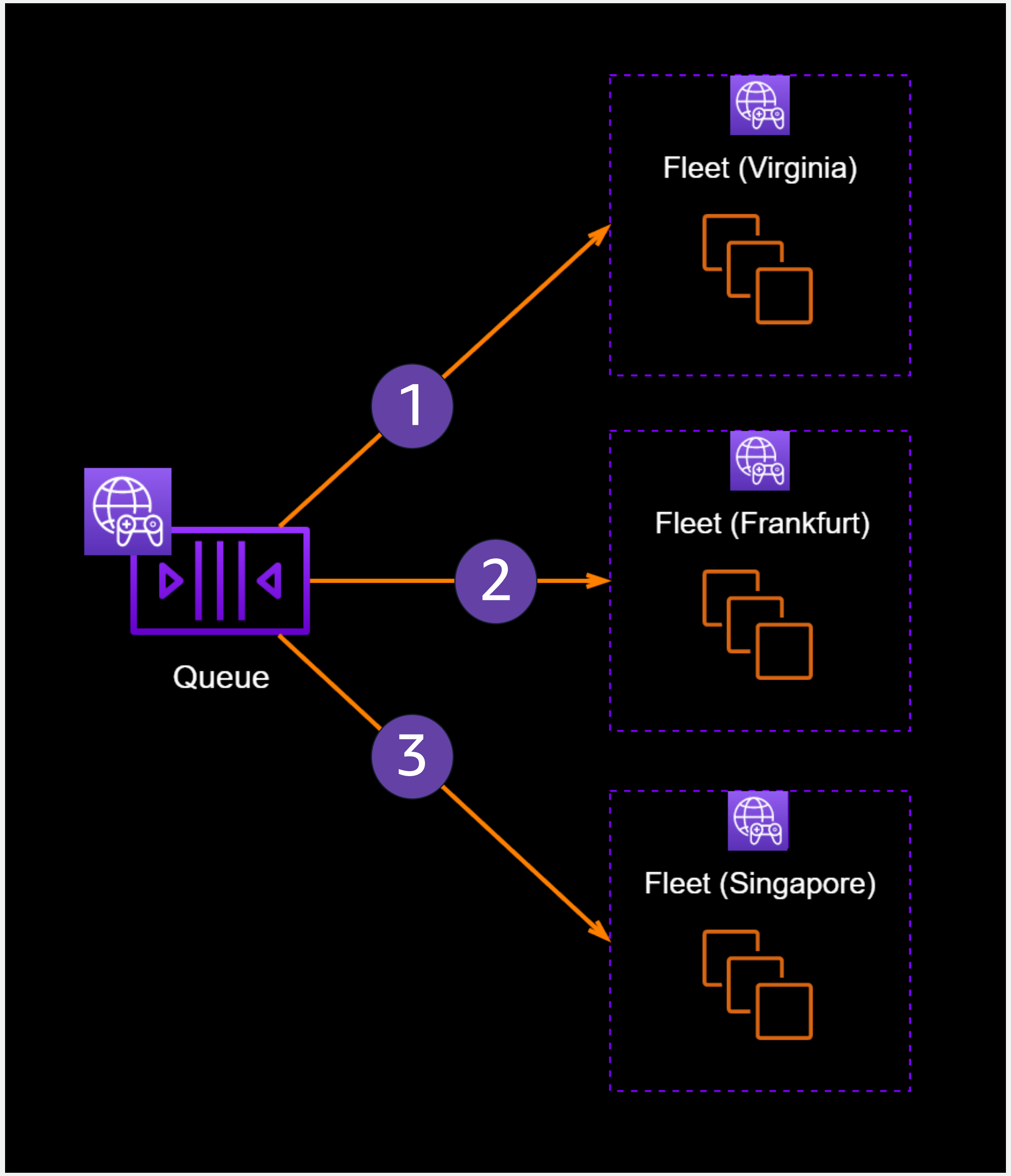
Queues – Member Fleets

- Place Fleets in AWS Regions close to your Players for the best experience
- You can use a single multi-region Fleet to cover all Regions
- 23 Regions supported by Amazon GameLift
- Both Fleets and Aliases can be configured for use by a Queue
- Queues can combine on-demand and spot instances from multiple regions



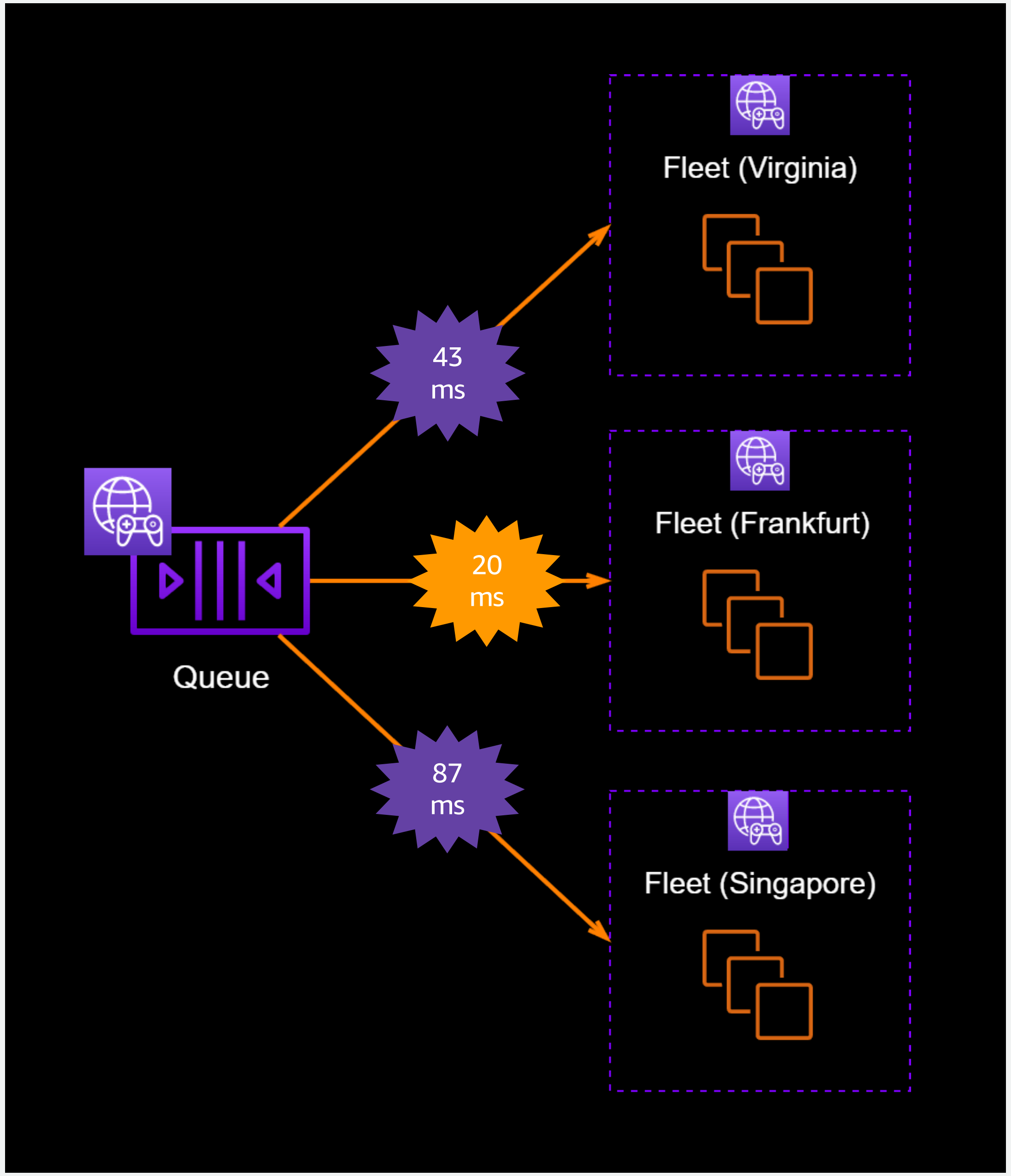
Queues – Fleet Priority

- Prioritize your destinations (Fleets or Aliases) to indicate the preferred fleet order.
- Amazon GameLift uses the priority to locate the best available session, starting with the first priority and moving down.
- When using Spot Instances set this Fleet to a higher priority, then set an On-Demand Fleet below for cases where Spot capacity is not available.



Queues – Player Latency Policies

- Set a maximum acceptable player latency for new game sessions.
- Specify the amount of time to enforce the policy.
- Multiple policies can be specified
- **Note:** This is done also on matchmaking level in FlexMatch, but Queue latency configuration allows you to control overflow to other Regions in case there are no available game sessions in the optimal placement location



Queues – Best Practices

- Use Multi-region Fleets registered to a Queue to deploy globally with a single build upload and Fleet configuration
- Use a Spot Fleet on high priority, and prepare a failover on-demand Fleet
- For disaster recovery, have a backup Queue in a separate Region and plan for failover
- A queue cannot have fleets with different certificate configurations. All fleets in the queue must have TLS certificate generation either enabled or disabled

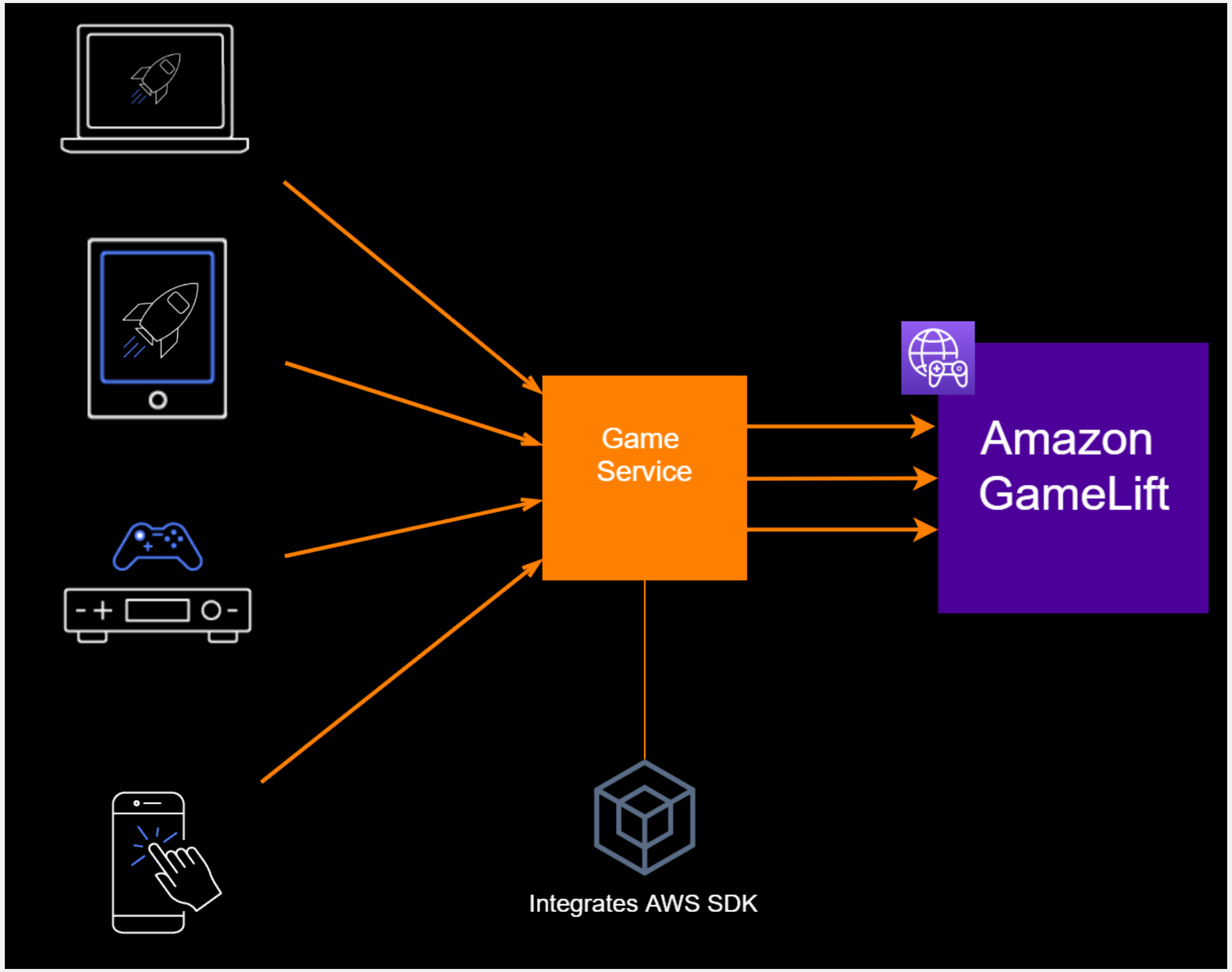
Clients Explained

Amazon GameLift supports any game client or game service that is able to make use of one of the AWS supported SDKs, languages include:

- C++
- C#
- Go
- Python
- JavaScript/Node.js
- Java …

Always build a backend service that calls GameLift APIs instead of the game client!

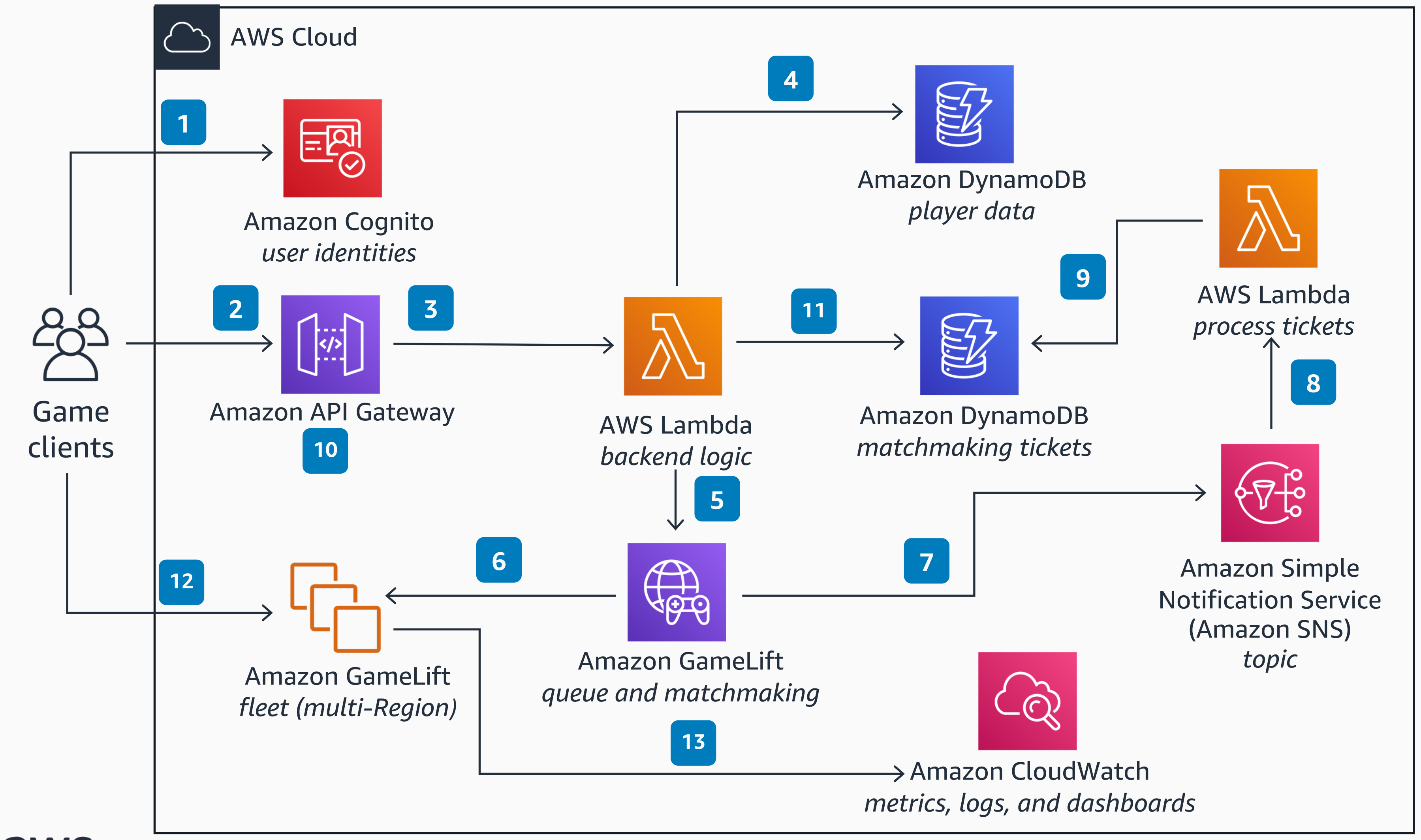
Clients – Best Practices



- Accept game client requests via a central game backend
- Integrate with Amazon GameLift from the Game backend to help isolate clients from change
- Authenticate game clients upon request to the Game Backend
- Capture latency information from Clients by pinging AWS endpoints. This can be for example requests to DynamoDB endpoints available across the regions

Multiplayer Session-based Game Hosting on AWS

Using Amazon GameLift multi-Region fleets and a serverless backend solution to host a session-based multiplayer game.



1. The game client requests an **Amazon Cognito** identity and temporary AWS credentials.
2. The client signs a matchmaking request to **API Gateway** with the temporary credentials. The request includes client latency information to supported AWS Regions.
3. **API Gateway** calls an **AWS Lambda** function with player identity information.
4. The **Lambda** function gets player skill level from a **DynamoDB** table.
5. The **Lambda** function requests matchmaking from **GameLift FlexMatch** with player skill and latency data.
6. **GameLift FlexMatch** creates a match with multiple players, and a **GameLift** queue allocates a session in a **GameLift** fleet location based on the latency data.
7. **GameLift FlexMatch** publishes an event to **Amazon SNS** on matchmaking success.
8. **Amazon SNS** triggers a subscribed **Lambda** function for ticket processing.
9. The **Lambda** function stores the ticket result in a **DynamoDB** table.
10. The game client polls for matchmaking success on a defined interval from **API Gateway.**
11. The **Lambda** function checks matchmaking information from the **DynamoDB** table and informs the client of a successful match by returning server IP, port, and player session ID.
12. The client connects directly to the server and sends the player session ID. GameLift Server SDK is used to validate the player session.
13. Game servers send logs and metrics to **Amazon CloudWatch** with **CloudWatch** agent.

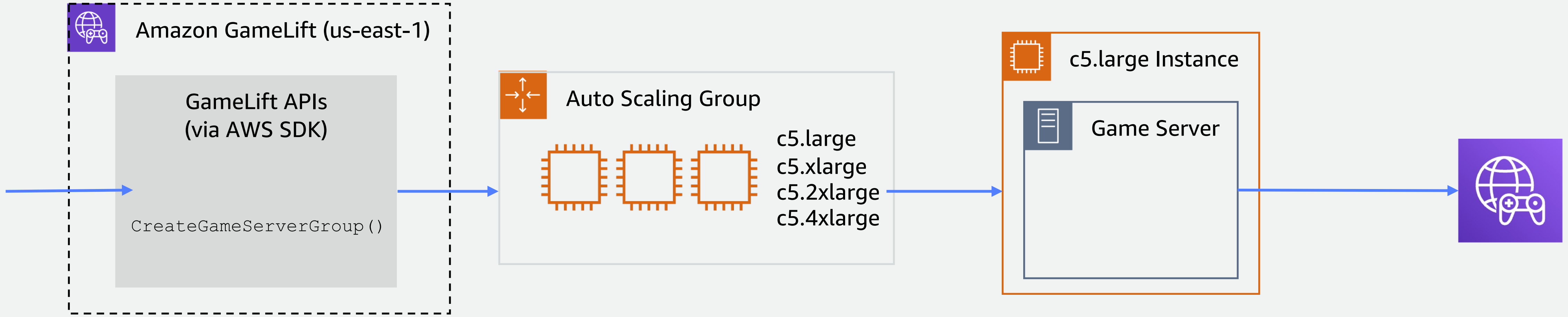
Reviewed for technical accuracy February 23, 2022

AWS Reference Architecture

GameLift FleetIQ benefits

- **More flexibility:** Game Server Group hosted on EC2 on your AWS account
- **Simple API layer** for game session management
- **Manages Fleet scaling** for you
- **Use your existing tools** and software to deploy and manage game server processes
- Access **GameLift Spot viability algorithm** independent of other GameLift features
- AWS SDK on the server side: **use the programming language of your choice**

How GameLift FleetIQ works



Use AWS SDK, CLI or CloudFormation to create a Game Server Group

An EC2 Auto Scaling Group is created with your configuration of instances

FleetIQ automatically scales this and manages spot and on-demand balance based on spot viability

You control how the game server processes are run on the instances

Game Servers report their state to FleetIQ

Your Backend calls `ClaimGameServer()` to request a free session

Matchmaking with Amazon GameLift FlexMatch

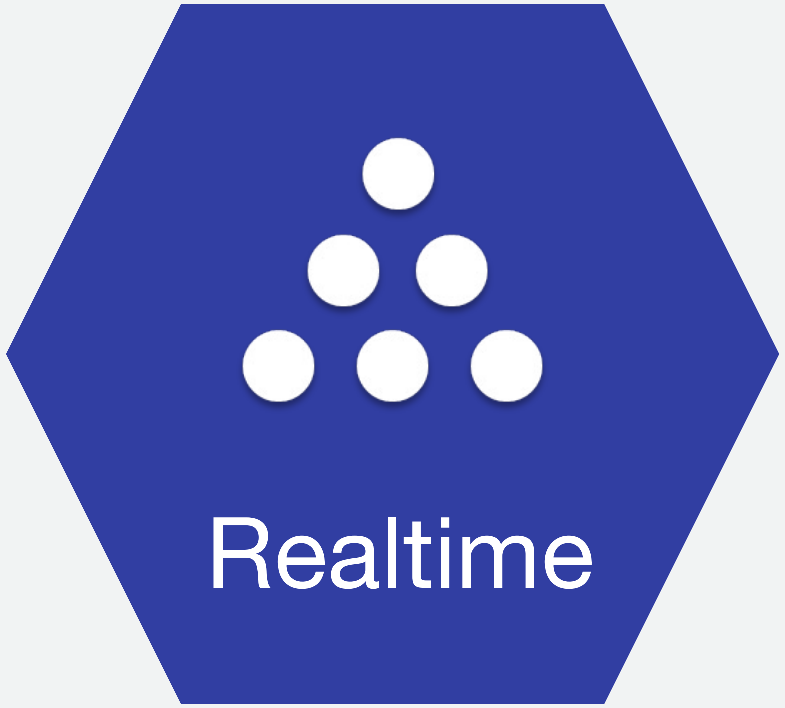
- Fully managed customizable matchmaking
- Works natively with GameLift Hosting, as well as standalone with any game server hosting solution
- Player teams support
- Latency-based matching
- Rule relaxing
- Match acceptance support
- Best region placement
- Player drop in/out support with FlexMatch backfill

```
{
    "name": "aliens_vs_cowboys",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{ "name": "skill", "type": "number", "default": 10 }],
    "teams": [{ "name": "cowboys", "maxPlayers": 8, "minPlayers": 4},
              { "name": "aliens", "maxPlayers": 8, "minPlayers": 4}],
    "rules": [{
        "name": "FairTeamSkill",
        "description": "Avg skill level in team max 10 distance from avg of all",
        "type": "distance",
        "measurements": [ "avg(teams[*].players.attributes[skill])" ],
        "referenceValue": "avg(flatten(teams[*].players.attributes[skill]))",
        "maxDistance": 10
    }, {
        "name": "EqualTeamSizes",
        "description": "The number of players in each team needs to match",
        "type": "comparison",
        "measurements": [ "count(teams[cowboys].players)" ],
        "referenceValue": "count(teams[aliens].players)",
        "operation": "="
    }],
    "expansions": [{
        "target": "rules[FairTeamSkill].maxDistance",
        "steps": [{
            "waitTimeSeconds": 5,
            "value": 50
        }, {
            "waitTimeSeconds": 15,
            "value": 100
        }]
    }]
}
```

GameLift Realtime Explained

The Realtime servers have a managed server logic you can extend with scripts

- Full network stack for game client/server interaction (encrypted TCP and UDP)
- Integrated with the GameLift service
- Live updates to Realtime configurations and server logic
- Flexible control of hosting resources
- Supports Unity/C# on the client side
- Node.js scripts on the server side



AWS FOR GAMES

Best Practices

Amazon GameLift Best Practices (Architecture)

- Don’t call GameLift APIs from the client, use your backend instead and authenticate the players
- Don’t use any of the Describe or Search APIs of GameLift control plane in production -> they are **not** designed for production use
  - Instead, use Queue events and FlexMatch events
- Use the Fleet IAM Role to access any AWS services (has to be assumed separately)
- Review the checklists for different development phases
- When using Spot fleets, always have a backup On-demand fleet as well behind the queue
- Terminate game server processes after each game to reduce any memory leaks etc. Issues. Call *ProcessEnding()* before terminating to immediately get a replacement process
- Only use Amazon Linux 2 OS for maximum performance and security

Amazon GameLift Best Practices (Operations)

- Use CloudWatch Agent or 3rd party tools to collect process metrics, custom metrics and logs → Build relevant global dashboards for multi-region fleets
- For game launch day, pre-warm the fleets and keep a high minimum, and switch to automatic scaling down after successful launch day
- Make sure to request increases to all utilized GameLift APIs prior to launch
  - *StartMatchmaking*, *StartGameSessionPlacement*, *CreateGameSession*, *CreatePlayerSession* etc.
- Make sure to request increases to all instance limits for all utilized instance types and regions
- Monitor critical CloudWatch metrics
  - AvailableGameServers, TicketsFailed, AverageWaitTime, PlacementsFailed/TimedOut …