



A guide to Amazon GameLift and game servers

CONTENTS



- 03 > **Executive summary**
- 05 > **Introduction**
- 06 > **Server hosting**
- 08 > **Amazon GameLift**
- 18 > **Amazon GameLift FleetIQ**
- 23 > **Amazon GameLift FlexMatch**
- 28 > **Servers and network**
- 34 > **Hosting options**
- 39 > **Next steps**

Amazon Web Services (AWS) offers a variety of options for hosting your game servers so you can choose the right tool for your needs. With AWS, you can host game servers directly on virtual machines (VMs) with your custom orchestration, choose a fully managed global game server hosting solution, or select only the features you need, including matchmaking, game session management, and low-cost Spot Instances for game servers.

Amazon Web Services

Amazon GameLift: Deploy, operate, and scale dedicated low-cost servers in the cloud for session-based and persistent world multiplayer games. Built on the AWS Global Cloud Infrastructure, this solution helps deliver high-performing, reliable, low-cost game servers while scaling your resource usage to meet player demand.

Amazon GameLift FleetIQ: Gain more control over game server deployment and game session management, an ideal choice if you want to run game servers in containers. This solution also optimizes low-cost Spot Instances to deliver up to a 70 percent discount compared to On-Demand prices.

Amazon GameLift FlexMatch: Customize key aspects of the matchmaking process to fit your game, including the latency-based and skill-based matching algorithm and building custom rule sets.

AWS Global Accelerator: Improve availability and traffic performance for your game by as much as 60 percent with simplified traffic routing.

Custom hosting options: If you need full control over how your game servers and game sessions are hosted, AWS provides a number of services, including:

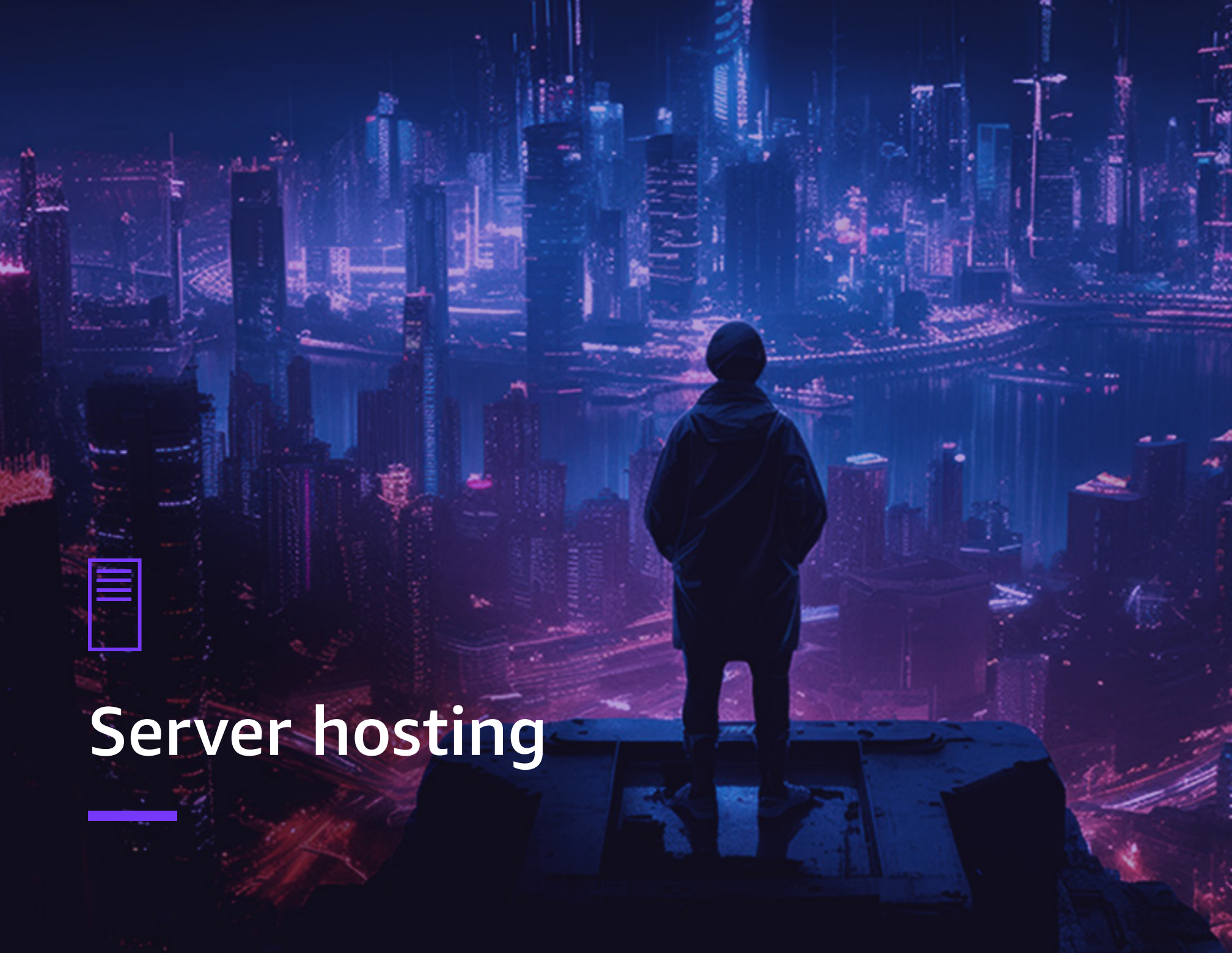
- **Amazon Elastic Compute Cloud (Amazon EC2):** Run your own orchestration for game servers that use On-Demand Instances or Amazon EC2 Spot Instances
- **Amazon Elastic Container Service (Amazon ECS):** Deploy multiplayer game servers using Docker or Windows containers with a managed service
- **Amazon Elastic Kubernetes Service (Amazon EKS):** Deploy multiplayer game servers using a managed Kubernetes service
- **AWS Fargate:** Use containers to run your servers without having to manage infrastructure

When gaming, players want to be fully immersed in their favorite worlds without interruption. To make this happen, even during peak hours, you need a game server hosting solution that can ramp up and down quickly and deploy globally.

Game server hosting is critical to your overall backend architecture for session-based and persistent world multiplayer games, so you need the right tools for the task. And since there's no single solution that fits every scenario, AWS has several options, including:

- Using a fully managed global server hosting solution
- Hosting game servers directly on VMs with your custom orchestration
- Choosing only the features you need, including matchmaking, game session management, and low-cost Spot Instances for game servers

Plus, we offer pay-as-you-go pricing with no upfront costs or long-term commitments. Our offerings use Amazon EC2 for web-scale cloud computing resources and automatic scaling to manage your hosting capacity. You can make use of the AWS Global Infrastructure to deploy your resources in a number of AWS Regions and AWS Local Zones close to the players and scale based on the needs in each of these locations.



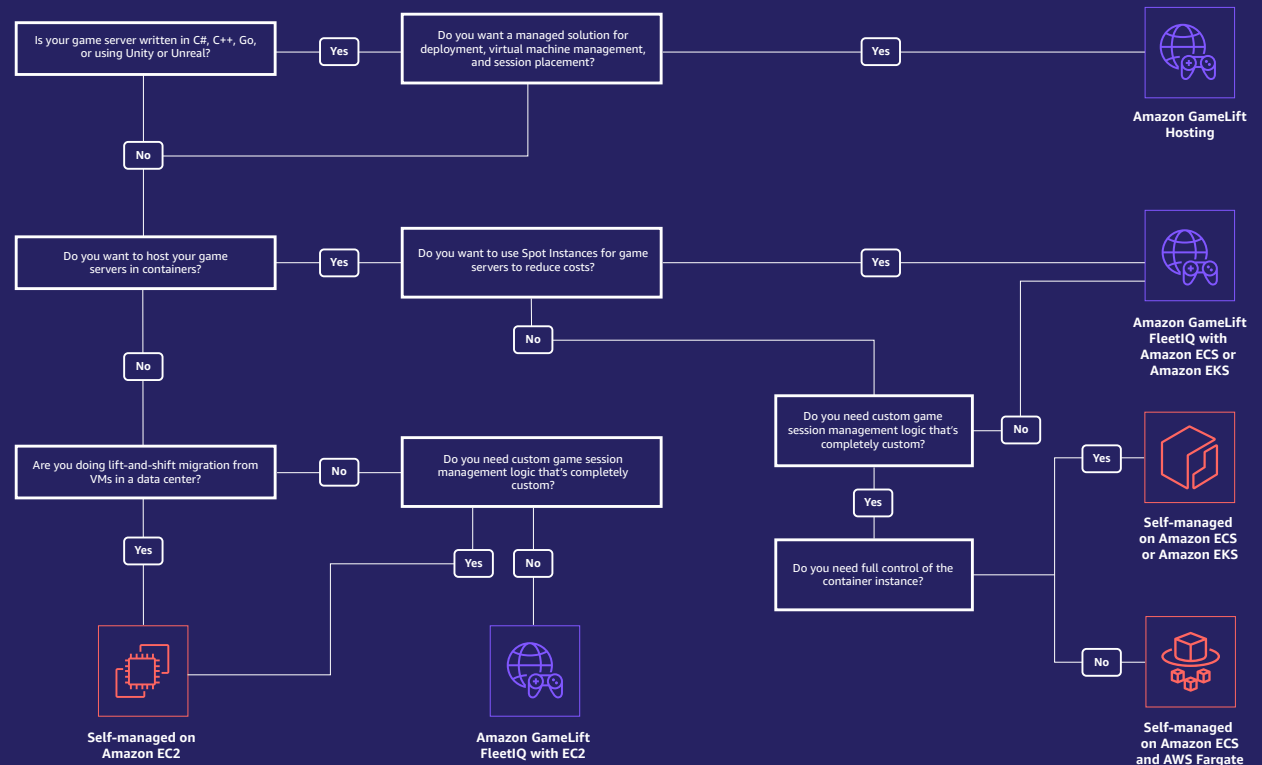
Server hosting



Selecting ideal server hosting options

There are several factors to consider when selecting your game server hosting option. This decision tree can help you navigate the different options on Amazon GameLift and AWS in general. Along with the options outlined here, you can use Amazon GameLift Realtime Servers when your game client uses C# (typically Unity) and you want a more managed server solution with only your custom game server functionality written in Node.js on the server side.

- Amazon GameLift hosting
- Self-managed on Amazon EC2
- Amazon GameLift FleetIQ with EC2
- Self-managed on Amazon ECS and AWS Fargate





Amazon GameLift

Fully managed Amazon GameLift

Amazon GameLift has a fully managed option for deploying, operating, and scaling session-based and persistent world multiplayer game servers. You provide the game server build and configure the service to your needs. This option adds a layer of convenience because it replaces the work required to host custom game servers, including:

- Buying and setting up hardware
- Managing availability and security
- Tracking performance

Plus, automatic scaling capabilities help you avoid paying for more resources than you need while ensuring you have game sessions available for new players to join with minimal waiting time.



CUSTOMER STORY: GUNGHO ONLINE

“With Amazon GameLift, we can provide smooth gameplay experiences with dedicated game server hosting. [Amazon] GameLift handles all of the management to deploy and manage game servers at a low cost, and [Amazon] GameLift FlexMatch automatically matches players for optimum latency during peak times. In our latest game, Ninjala, we’ve seen low rates up to 100 ms.”

Takanori Kikuchi, CTO, GungHo Online Entertainment, Inc.



Fully managed Amazon GameLift—let's talk specifics

SERVER LOCATION

Set up Amazon GameLift fleets to host your game server builds on a fleet of VMs. Each fleet has a home AWS Region, but you can add any of the other 24 supported AWS Regions and AWS Local Zones as locations within the fleet, letting you manage a global deployment with a single Amazon GameLift fleet. In addition to managed locations, you can add any custom hardware, such as your on-premises location, anywhere to extend your global reach even further.

SERVER RELIABILITY

Your fleet uses either Spot Instances or On-Demand Instances. Spot Instances (based on Amazon EC2 Spot Instances) provide a lower cost but have an increased

probability of game session interruption if the capacity is needed back for on-demand use. Amazon GameLift, however, has additional safeguards that make game session interruptions rare, and fleets with Spot Instances are a good choice for most games. Typically, you use a combination of Spot fleets and On-Demand fleets to have a failover mechanism to On-Demand when the Spot capacity is not available. If you have long-running game sessions spanning across hours or even days, it's better to choose On-Demand Instances.

PLAYER CAPACITY

A fleet can have many instances, each one capable of hosting multiple, simultaneous game sessions. You can pack up to 50 game sessions on a single instance and choose between a variety of instance types to find the perfect combination for you. You can add or remove instances from your fleet as needed and use automatic scaling to adjust as player demands shift.

NECESSARY GAME RESOURCES

Amazon GameLift supports a range of operating systems and a wide variety of instance types. The instance type determines the kind of computing hardware used, including processing power, memory, and networking capacity. The solution also supports a number of Amazon EC2 instance types and recently added support for ARM-based AWS Graviton3 instances that provide the best price for cloud workloads and reduce carbon footprint up to 60 percent. See [How to host your Unreal Engine game for under \\$1 per player](#) for a deeper dive into cost optimization.

Key components

You can use fully managed Amazon GameLift on a number of abstraction layers, and you can choose from managing sessions at the fleet level to globally managed matchmaking across fleets. The key components for Amazon GameLift-based end-to-end solutions include:

Game backend service: In addition to Amazon GameLift, you need to host a backend service for your game. The backend will use Amazon GameLift application programming interfaces (APIs) to initiate matchmaking and request game session placement. Popular options for game backend hosting include serverless backends with [Amazon API Gateway](#) and [AWS Lambda](#) or container-based backends with [Elastic Load Balancing \(ELB\)](#) and [Amazon ECS](#). To store your player data, you can use a range of database options, including [Amazon DynamoDB](#) for a fully managed NoSQL database and [Amazon Relational Database Service \(Amazon RDS\)](#) for a managed SQL database. [Guidance for Custom Game Backend Hosting](#) is a best-practice reference implementation for secure and scalable game backend services.

Amazon GameLift FlexMatch matchmaking configuration and rule set: If you want to use the matchmaking capabilities of Amazon GameLift, you define a matchmaking configuration and a JSON-document-based rule set and use the matchmaking APIs of Amazon GameLift from your backend service. You can also use your own matchmaker with Amazon GameLift.

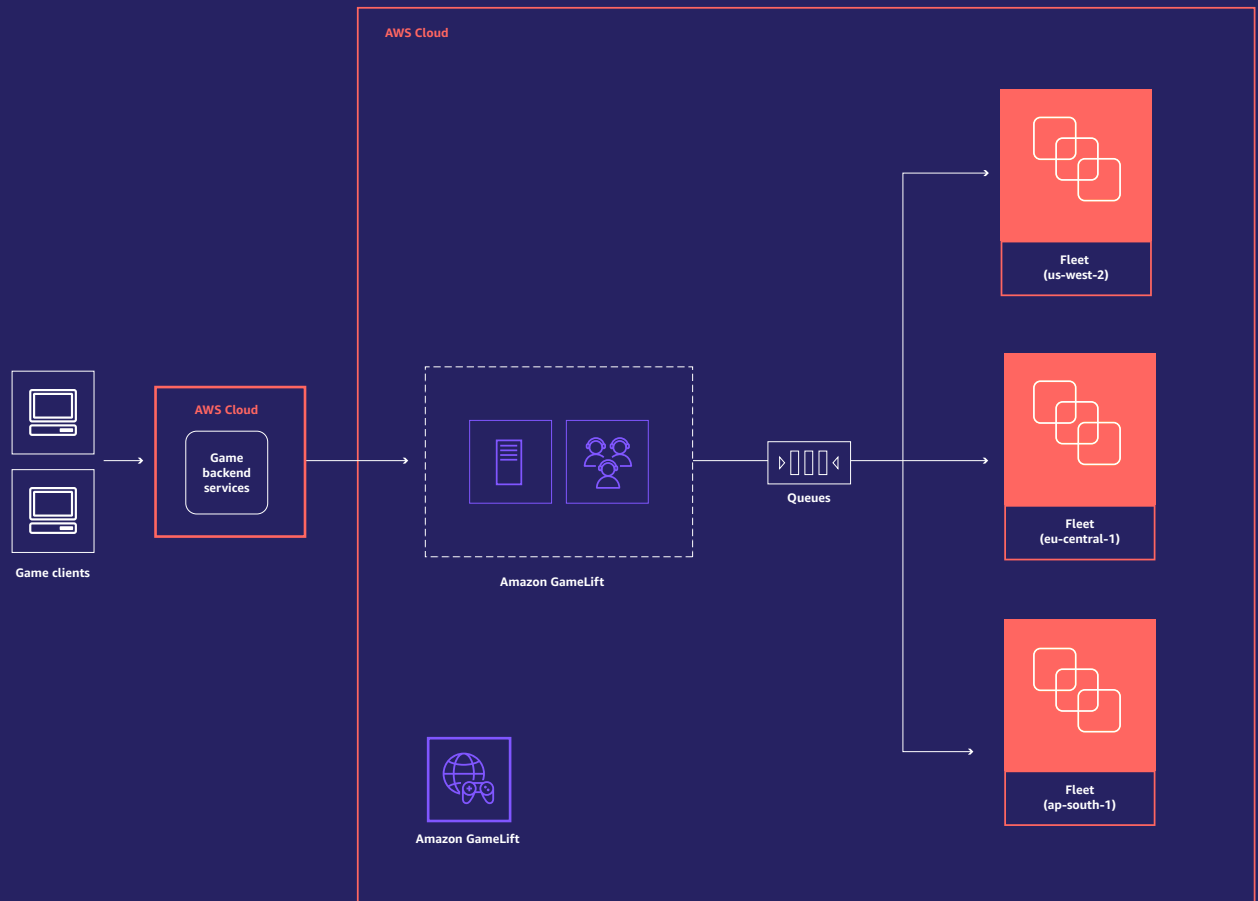
Amazon GameLift queue: To place game sessions across Amazon GameLift fleets globally, you need an Amazon GameLift queue. Within the queue, you define the prioritized order of your locations and the latency requirements. The queue then places game sessions on the lowest cost and lowest latency option across your fleets. Queues can be used with or without a matchmaking configuration. Typically, you would host a multilocation Spot fleet and a failover On-Demand fleet to optimize cost and maximize availability.

Amazon GameLift fleets: This is the actual fleet of Amazon EC2 instances hosted by Amazon GameLift, which runs your [game server build](#) with your defined configuration and scaling logic. A fleet can be used directly, or one or more fleets can be associated with an Amazon GameLift queue.



Amazon GameLift deployment components

Your game clients communicate with a backend service, which then either initiates matchmaking through Amazon GameLift FlexMatch or requests a session placement directly through a queue. In addition to these two options, you can provision sessions directly on a fleet for some special architecture scenarios.



CUSTOMER STORY: FRAG LAB

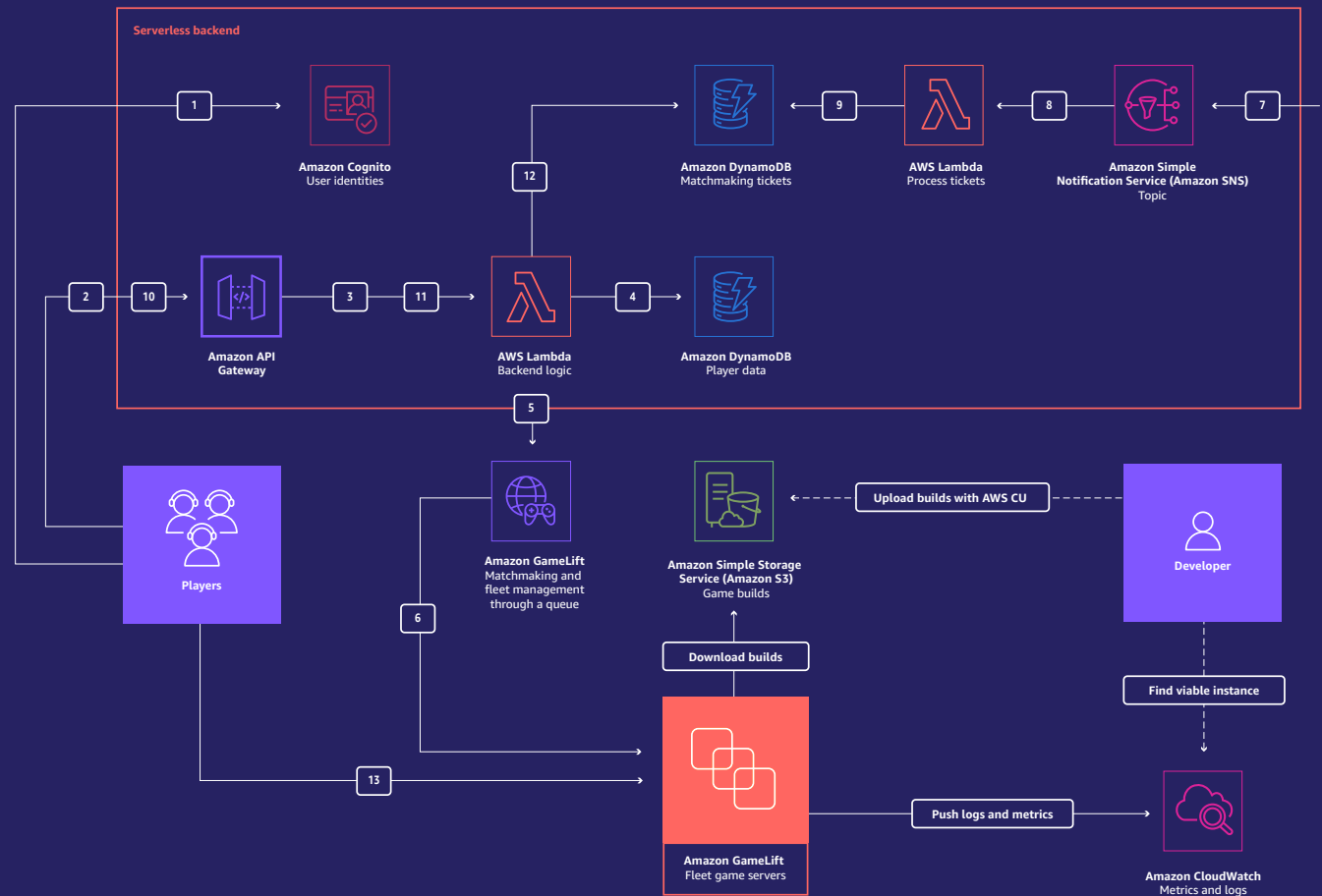
“As part of our partnership with Wargaming to deliver a AAA multiplatform, free-to-play shooter, we want to unify players from all over the world, so providing global availability is in our DNA and is one of our top priorities. That is why Amazon GameLift is one of the top choices for backend servers. Using Amazon GameLift, we were able to build our game on stable, scalable, and readily available servers from all over the world while keeping costs low and ensuring fun, smooth gameplay experiences.”

Sergii Rustamov, Technical Director, Frag Lab



Example architecture

This covers a common pattern in which you host a REST API-based backend with Amazon API Gateway and AWS Lambda, call the Amazon GameLift APIs, and receive Amazon GameLift FlexMatch matchmaking events. Your players communicate with the backend through an identity received from Amazon Cognito and access your game servers directly with TCP or UDP once they've received session information from the backend service. A serverless backend implementation scales with ease based on demand from your first player to millions of players and requires minimal operational overhead.



The following steps cover the solution flow:

- 01 >** Game client requests an Amazon Cognito identity from an Amazon Cognito identity pool. Optionally, this can be connected to external identity providers such as Google, Apple, Xbox Live, and Steam.
- 02 >** The game client receives temporary access credentials and requests a game session through an API hosted with AWS API Gateway by signing the request with these credentials.
- 03 >** AWS API Gateway invokes an AWS Lambda function.
- 04 >** The AWS Lambda function requests player data from the Amazon DynamoDB NoSQL table. The Amazon Cognito identity can be used to securely request the correct player data, since the authenticated identity is provided in the request context data.
- 05 >** With the correct player data for any additional information, such as player skill level, the AWS Lambda function requests a match through Amazon GameLift FlexMatch matchmaking. Amazon GameLift FlexMatch lets you define matchmaking configuration with JSON-based configuration documents. The game client can send latency data against the different Regions in addition to allowing latency-based matchmaking.
- 06 >** Once Amazon GameLift FlexMatch has matched a suitable group of players with accepted latency to a shared Region, it will request a game session placement through an Amazon GameLift queue that has fleets with one or more regional locations registered to it.
- 07 >** Once the session is placed on one of the fleets, an event notification is sent to Amazon Simple Notification Service (Amazon SNS).
- 08 >** An AWS Lambda function will receive the Amazon SNS event and process it.
- 09 >** If the ticket is a MatchmakingSucceeded event, the AWS Lambda function writes the result to Amazon DynamoDB with the server port and IP.
- 10 >** A time-to-live (TTL) value is used to make sure the tickets are deleted from Amazon DynamoDB when they are no longer needed.
- 10 >** The game client makes a signed request to Amazon API Gateway to check the status of the matchmaking ticket on a specific interval.
- 11 >** An AWS Lambda function for checking the matchmaking ticket status is invoked by Amazon API Gateway.
- 12 >** From Amazon DynamoDB, the AWS Lambda function checks if the ticket has already succeeded. In case it has, it will send the IP, port, and player session ID back to the client. Otherwise, it will send a response that the match is not yet ready.
- 13 >** The game client will connect to the game server using TCP or UDP by using the port and IP provided by the backend. It will also send the player session ID to the game server, which can validate it using the Amazon GameLift Server SDK.

CUSTOMER STORY: PANZERDOG

“For Panzerdog, fast, fair matchmaking is a top priority to ensure a great player experience. By using Amazon GameLift, our small team is able to balance their attention to guaranteeing experience and innovating with new technology improvements without requiring additional specialist resources. Also, global availability helps us provide a low latency experience to players wherever they are, while Amazon EC2 Spot fleets enable us to save up to 70% of our game server costs with automatic scaling.”

Roman Malakhov, Head of Studio, Panzerdog





Amazon GameLift FleetIQ

Amazon GameLift FleetIQ

If you need more oversight regarding game server deployment or game session management, Amazon GameLift FleetIQ can help.

Optimize the use of low-cost Spot Instances for cloud-based game server hosting

With this feature, you can work directly with your hosting resources in Amazon EC2 and AWS Auto Scaling to take advantage of Amazon GameLift optimizations and achieve more:

CONTROL

You directly manage Amazon EC2 instances in your AWS account with full control over how the game server's executables are deployed and run. This means that in addition to running directly on Amazon EC2 instances, you can register the fleet of instances to an Amazon ECS or Amazon EKS cluster, host your game servers as containers, and make use of these services' orchestration capabilities.

CONVENIENCE

Amazon GameLift FleetIQ provides lightweight APIs for registering and deregistering your game servers and updating their health and state. You can also use an API to claim a game server from the pool of free registered servers and integrate to your custom backend service.

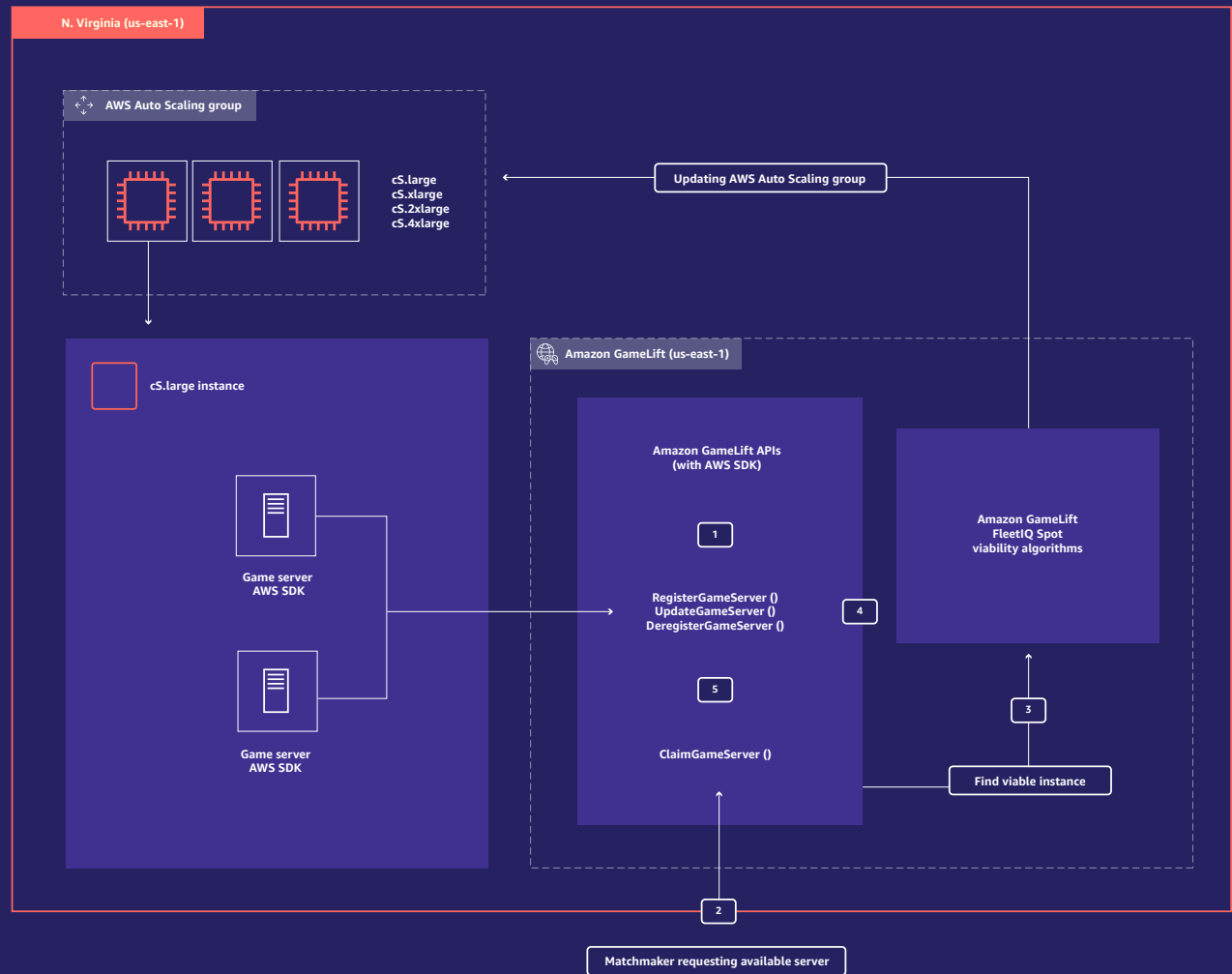
FLEXIBILITY

Amazon GameLift FleetIQ uses the concept of a game server group that creates an AWS Auto Scaling group of Amazon EC2 instances. You can use the built-in functionality to scale the AWS Auto Scaling group of instances based on game session utilization. You can also decide if you want to use a combination of Spot Instances and On-Demand Instances or one or the other for your game servers. Or you can mix different instance types to maximize the availability of Spot Instance capacity.



Example architecture

This covers the scenario in which you host your game servers directly on Amazon EC2 instances deployed through an Amazon GameLift FleetIQ game server group. Your game servers running on these instances and your backend service will communicate with the Amazon GameLift FleetIQ APIs using the AWS SDK of your choice to manage the game sessions. You can implement the backend service as a serverless solution in a similar fashion as the previous example or choose any other hosting option that fits your needs.



The following steps cover the solution flow:

- 01 > The game server process will register to Amazon GameLift FleetIQ by calling RegisterGameServer API. This includes information on which instance ID it's running on, which game server group it's registering to, and the port and IP of the game server.
- 02 > Your backend service calls the ClaimGameServer API. This can be implemented as a serverless backend in a similar fashion as in the previous two example architectures, or it can be any backend implementation of your choice.
- 03 > Amazon GameLift FleetIQ will check the viability of hosting the game session on one of the Spot Instances in the AWS Auto Scaling group. In case the placement is viable, it will claim one of the free servers running on a Spot Instance; otherwise, it will look for an On-Demand Instance to host the session.
- 04 > After the players have connected to the game server, the server will call UpdateGameServer to mark itself as utilized for Amazon GameLift FleetIQ. UpdateGameServer should also be called on 60-second intervals as a health check.
- 05 > The game server will call DeregisterGameServer API after it's done hosting a game session. Typically, you terminate the game server process after this and replace it with a new one that registers itself to Amazon GameLift FleetIQ.

As mentioned before, you can optionally use Amazon ECS or Amazon EKS to host your game servers in containers and still use Amazon GameLift FleetIQ. [This example solution on GitHub](#) covers the scenario of integrating Amazon GameLift FleetIQ with Amazon ECS.

CUSTOMER STORY: ILLFONIC

“Migrating to Amazon GameLift has enabled IllFonic to provide a seamless gaming experience with low latency and better, faster matchmaking while reducing costs and work for its in-house staff. Working on Amazon GameLift has made our small developing company look really good, and it’s making our titles great to play. It definitely shows that a small company like us can compete and punch above its weight.”

Charles Brungardt, CEO and Cofounder, IllFonic





Amazon GameLift FlexMatch

Amazon GameLift FlexMatch

Along with game server hosting, you often need to do matchmaking based on player attributes, location, or other custom rules. You also need to place these matched player groups in available game sessions.

Customize matchmaking within Amazon GameLift for multiplayer games

With Amazon GameLift FlexMatch, you can build a custom set of rules to:

- Define what a multiplayer match looks like for your game
- Evaluate and select compatible players for each match

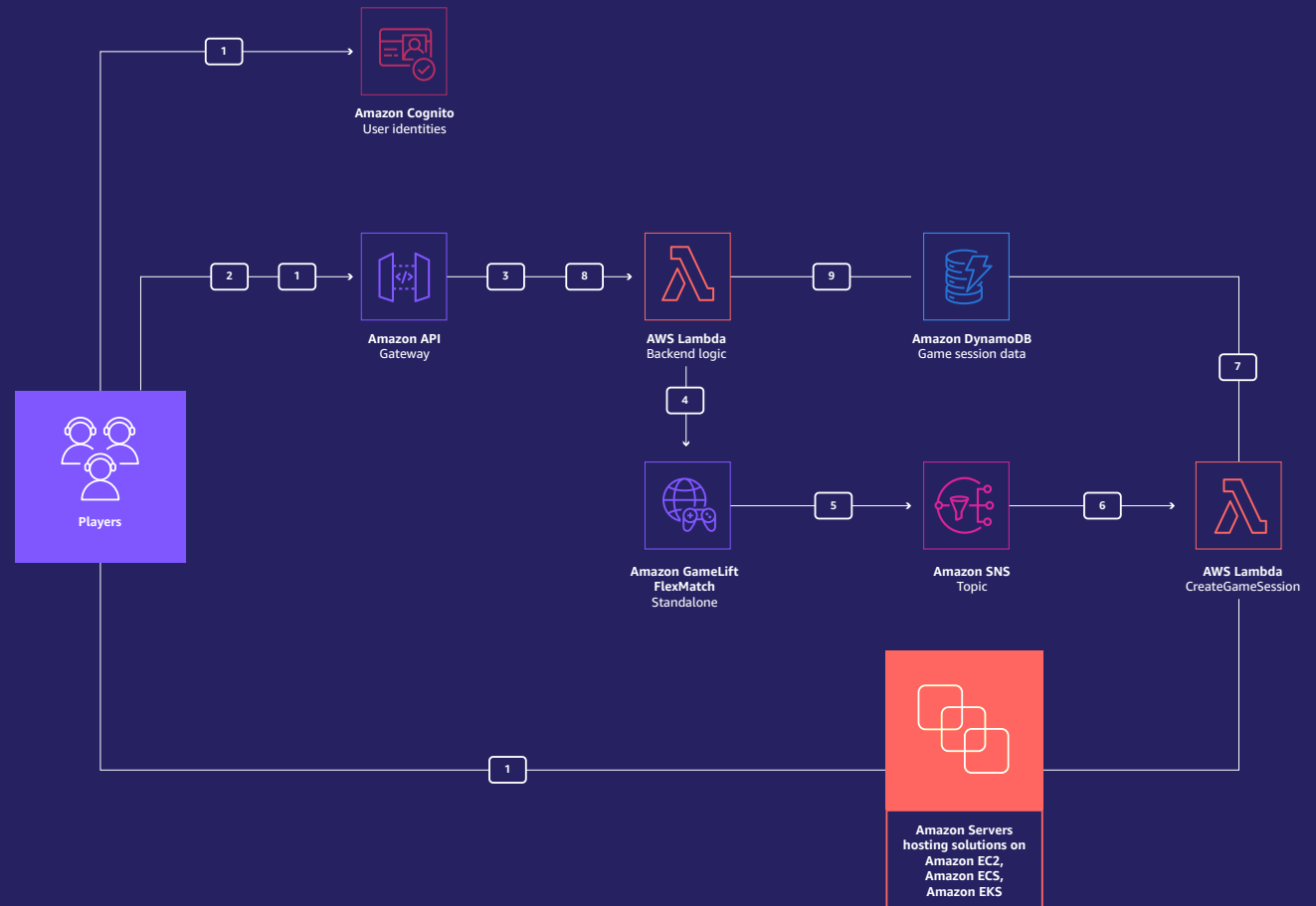
You can also customize key aspects of the matchmaking process, including latency-based and skill-based matching, and build custom rule sets. This solution gives you the flexibility to:

- Set matchmaking priorities depending on your game requirements
- Balance between matchmaking speed and quality of matches (equal player or team skill level, for example)
- Decide how player latency factors into a match, depending on your game's latency requirements

Amazon GameLift FlexMatch is available for use either with fully managed Amazon GameLift (including Realtime Servers) or as a standalone matchmaking service. You can implement Amazon GameLift FlexMatch as a standalone feature with games that use peer-to-peer architecture or host game servers on other cloud compute solutions (including Amazon GameLift FleetIQ).

Example architecture

This example describes a serverless backend solution that makes use of Amazon GameLift FlexMatch in standalone mode. The game servers can be hosted on Amazon EC2, Amazon ECS, or Amazon EKS, either with or without Amazon GameLift FleetIQ.



The following steps cover the solution flow:

- 01 >** Game client requests an Amazon Cognito identity from an Amazon Cognito identity pool. Optionally, this can be connected to external identity providers such as Google, Apple, Xbox Live, and Steam.
- 02 >** The game client receives temporary access credentials and requests a game session through an API hosted with Amazon API Gateway by signing the request with these credentials.
- 03 >** Amazon API Gateway invokes an AWS Lambda function.
- 04 >** The AWS Lambda function calls Amazon GameLift API to initiate matchmaking using Amazon GameLift FlexMatch standalone.
- 05 >** Amazon GameLift FlexMatch will publish to an Amazon SNS topic after the matchmaking has succeeded for a group of players.
- 06 >** An AWS Lambda function will process the event and use your game session management solution to reserve a game session on your server fleet. Game sessions can be managed with Amazon GameLift FleetIQ (in which case you would call ClaimGameServer API at this point) or your custom hosting solution, which typically needs a database for game session and player session management.
- 07 >** The AWS Lambda function writes the game session information (including IP, port, and other info) to Amazon DynamoDB.
- 08 >** The game client requests information on the status of matchmaking through a signed request to Amazon API Gateway.
- 09 >** The AWS Lambda function will check the status of the game session from Amazon DynamoDB. If it's ready, it will return the IP, port, and any additional information, such as your custom player session ID, to the client.
- 10 >** The game client connects directly to the game servers running with your custom hosting solution or Amazon GameLift FleetIQ on Amazon EC2, Amazon EKS, or Amazon ECS.

CUSTOMER STORY: N3TWORK

“We can quickly provision large amounts of server capacity, ensuring we have servers ready to go for anyone around the world. We don’t have to add additional staff to manage infrastructure, and this helps improve the bottom line not just from overhead but also from server hosting by tapping into unused Amazon EC2 Spot Instances at a discount when compared to On-Demand pricing.”

Stephen Detwiler, Former Co-Founder & CTO, N3TWORK





Servers and network

Going global with your game servers and networking

Need to deploy game servers globally to a number of AWS Regions, AWS Local Zones, and possibly custom locations while meeting the varying latency requirements? When using fully managed Amazon GameLift, you can host your game backend services in a single AWS Region together with Amazon GameLift FlexMatch matchmaking and Amazon GameLift queue configurations. You can also deploy game server fleets across any of the supported AWS Regions and AWS

Local Zones globally with ease and extend to any custom locations with Amazon GameLift Anywhere. By sending the latency data from game clients to your backend, you can feed this information to Amazon GameLift FlexMatch and queues, which will then select the most appropriate location for a low-latency player experience.



The benefits of Amazon GameLift Anywhere

A more recent feature of Amazon GameLift is Amazon GameLift Anywhere, which allows you to host additional fleets in any custom location, such as your on-premises infrastructure. And because your local hardware can now be an Amazon GameLift-managed server, you can iterate on the server build in your familiar local desktop environment, and any server errors can materialize in seconds. This can significantly reduce the development time.

Amazon GameLift Anywhere fleets act similar to managed fleets, and you can register both behind a single Amazon GameLift queue. This allows you to balance session placement between managed and custom locations based on player latency. You can also prioritize the fleet order if you want to use your custom hardware first and then extend to the cloud.



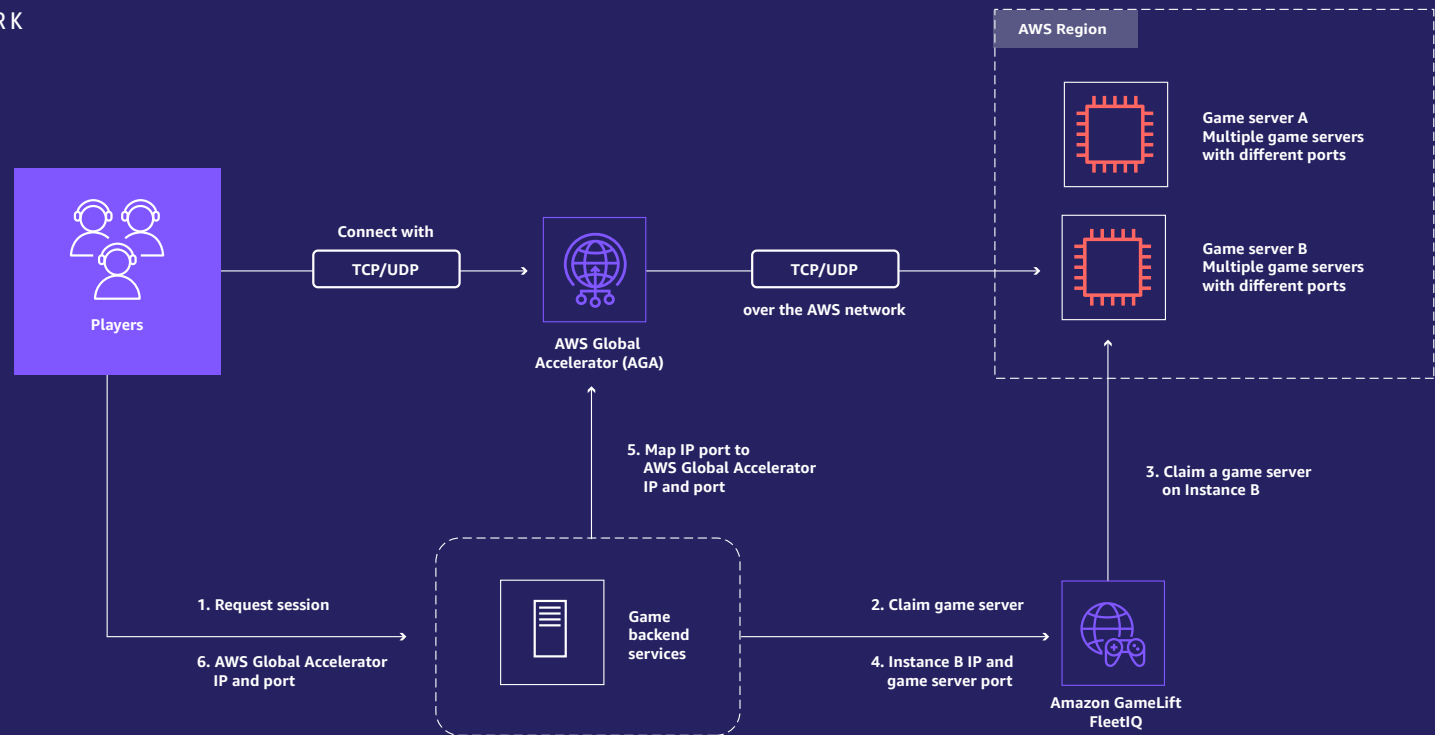
AWS Global Accelerator for game servers

You might need to accelerate the connectivity between your players and game servers even further. Especially for fast-paced games, your players expect a continuously immersive experience. But as the number of concurrent players grows, so does the complexity of your IP management and the likelihood of latency-inducing routing problems. [AWS Global Accelerator](#) simplifies traffic routing and improves availability and performance for your game by as much as 60 percent.

AWS Global Accelerator moves your traffic off the public internet and onto the AWS network through more than 80 edge locations. So, no matter what entry point or server your players are connecting through, AWS Global Accelerator provides two customer-facing static IPs. These static IPs are announced from AWS edge locations around the world through anycast and route traffic back to regional endpoints, such as your Amazon EC2 instances and load balancers, over the Amazon network. This feature helps you scale and reconfigure your endpoints and even handle endpoint failures on the fly without interrupting play or requiring client updates.

Acceleration is supported for both your TCP traffic and your in-game UDP traffic.





Custom routing with AWS Global Accelerator and Amazon GameLift FleetIQ

The AWS Global Accelerator custom routing feature helps you accelerate traffic to your game servers easily and effectively. With a custom routing accelerator, you can direct multiple users to a unique port on your accelerator.

The accelerator will map the port to a specific Amazon EC2 destination with a private IP address, port in single

or multiple AWS Regions, and route the traffic there. This feature helps you integrate AWS Global Accelerator with your application logic, such as matchmaking servers. With custom routing accelerators, you can use AWS Global Accelerator as the single point of entry for your game servers while sending your user traffic to specific Amazon EC2 destinations in any AWS Region. Custom routing can also be used together with Amazon GameLift

FleetIQ. After you've used the Amazon GameLift FleetIQ API to claim a game server on an appropriate instance, you'll be able to use the AWS Global Accelerator API to map that game server IP and port to an AWS Global Accelerator port. You can then start routing accelerated TCP and UDP traffic to your game servers.

CUSTOMER STORY: GAMELOFT

“Since implementing [Amazon] GameLift FleetIQ and [Amazon EC2] Spot [Instances], servers automatically scale with fluctuating player demand, and preliminary estimates already show cost savings of 20%–40% in comparison to On-Demand Instances.”

Nicolae Georgescu, Studio Online Lead, Gameloft





Hosting options



Other game server hosting options

In addition to using the options within the Amazon GameLift services, you might need full control over how your game servers and game sessions are hosted.

Or you might be building a more asynchronous or stateless solution that better fits other architecture models. AWS provides a number of services you can use to build any custom game server hosting solution to match your needs.

KEY OPTIONS



Amazon EC2: Run your own orchestration for game servers that use On-Demand Instances or Amazon EC2 Spot Instances



Amazon ECS: Deploy multiplayer game servers using Docker or Windows containers with a managed service



Amazon EKS: Deploy multiplayer game servers using a managed Kubernetes service



AWS Fargate: Use containers to run your servers without having to manage infrastructure



AWS Lambda: Run event-driven code without having to spin up and manage a server

Virtual machines

Amazon EC2 provides virtual server instances for a variety of platforms in the cloud. The option of bare metal instances provides your games with direct access to the processor and memory of the underlying server. **Amazon EC2 Spot Instances** offer unused Amazon EC2 instances for less than the On-Demand Instance price. This helps you save costs as long as you manage the possible interruption of Spot Instances. To learn more about Amazon EC2, check out the [developer resources](#)

or [get started building with Amazon EC2](#). Learn how **Gearbox Software used Amazon EC2** to support millions of concurrent players and thousands of authentications per second for the worldwide launch of Borderlands 3.

To host game servers on Amazon EC2, you commonly use AWS Auto Scaling groups to scale the fleet of instances based on a custom metric in Amazon

CloudWatch, such as the percentage of utilized game sessions. You would publish this metric directly from your game servers or from your game backend. You would also need to host a backend service and database for game and player sessions.

1:21.10
1:22.15
1:23.32
1:24.59
1:23.10

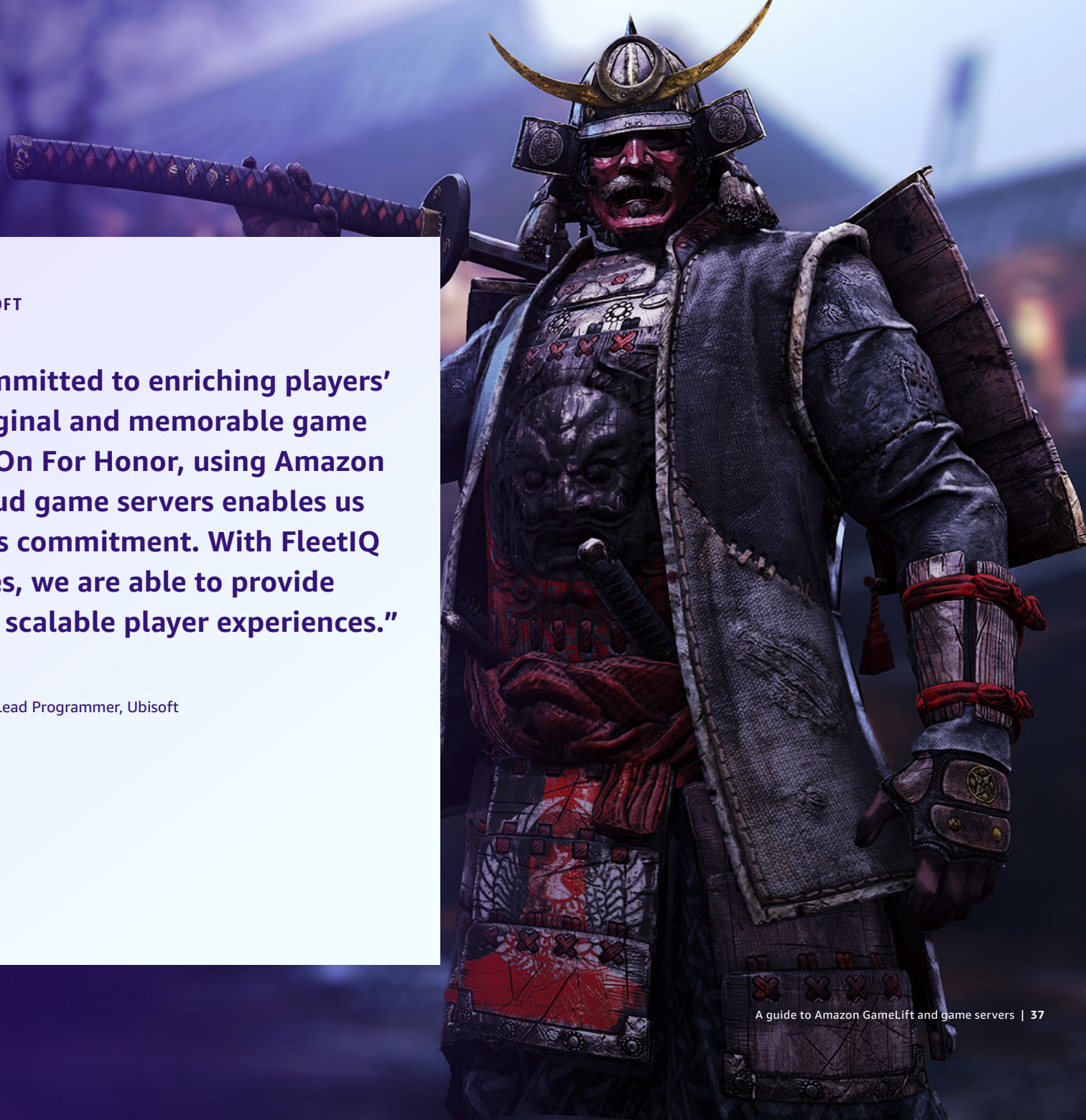


HOSTING OPTIONS

CUSTOMER STORY: UBISOFT

“Ubisoft is committed to enriching players’ lives with original and memorable game experiences. On For Honor, using Amazon GameLift cloud game servers enables us to uphold this commitment. With FleetIQ Spot Instances, we are able to provide seamless and scalable player experiences.”

Laurent Chouinard, Project Lead Programmer, Ubisoft



Containers



Amazon ECS offers highly scalable and fast container orchestration in a fully managed service with support for Docker and Windows containers. Amazon ECS includes multiple scheduling strategies for container deployment for different kinds of game functions—from stateless backend services to game servers. To learn more about Amazon ECS, check out [Amazon ECS documentation](#). To host stateful game servers on Amazon ECS, you need to implement a scheduling service that provisions game server tasks in your Amazon ECS cluster based on the availability of game sessions and integrates to a custom backend.



Amazon EKS is a fully managed Kubernetes service deeply integrated with other AWS services such as serverless compute and automatic scaling. This integration gives you a scalable and highly available control plane over your game servers. To learn more about Amazon EKS, check out the [Amazon EKS user guide](#) or [get started deploying with Amazon EKS](#).

See the [FleetIQ adapter for Agones](#) to learn how to integrate the open-source Agones game server management solution with Amazon EKS and Amazon GameLift FleetIQ.

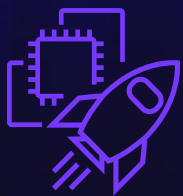
Serverless



AWS Fargate is a serverless compute engine that works with both Amazon ECS and Amazon EKS. It enables you to use containers to run your game servers without having to manage the underlying instances. This takes away much of the infrastructure management, helping you focus on building your game. To learn more about AWS Fargate, check out these [getting started resources](#). Also, check out this [example solution on hosting game servers on AWS Fargate](#) to learn more.



AWS Lambda helps you run code without having to spin up a server or container. Although this typically won't match the need of real-time game session hosting, it's great for turn-based games. For example, [Second Dinner and Nuverse built and scaled Marvel Snap, the 2022 mobile game of the year, using AWS Lambda](#). To learn more about AWS Lambda, check out the [getting started developer guide](#). You can also learn how [Activision Blizzard uses AWS Lambda](#) to deliver personalized experiences to millions of Call of Duty players daily or check out reference architectures for inventory management, [real-time analytics for mobile games](#), and [push notifications](#) using AWS Lambda and other AWS services.



Next steps

NEXT STEPS

Whether you're looking for a fully managed solution with Amazon GameLift or certain features with Amazon GameLift FleetIQ and Amazon GameLift FlexMatch as a standalone service, AWS has what you need. And in case you like to build things from the ground up, you're covered with all layers of compute abstraction, from VMs to serverless.

AWS PARTNERS



VISIT THE GAME SERVERS AND NETWORKING SOLUTION PAGE:

[Game Servers and Networking – AWS for Games](#)

VISIT THE AMAZON GAMESLIFT PRODUCT PAGE:

[Amazon GameLift – AWS for Games](#)

Contact us: **awsforgames@amazon.com**

© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

