



FleetIQ Developer Guide

Amazon GameLift Servers



Version

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon GameLift Servers: FleetIQ Developer Guide

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Amazon GameLift Servers FleetIQ?	1
How FleetIQ works	2
Amazon GameLift Servers FleetIQ logic	2
Key resources and components	5
Game architecture	7
Supplementing on-premises hosting	7
Life of a game server group	9
Life of a game server	11
Spot balancing process	13
Best practices	15
Amazon GameLift Servers FleetIQ features	18
Pricing for Amazon GameLift Servers FleetIQ	19
Setting Up	20
Supported software	20
Set up your AWS account	21
Create an AWS account	21
Manage user permissions for Amazon GameLift Servers FleetIQ	23
Create IAM roles for cross-service interaction	28
Preparing games for FleetIQ	35
Integration steps	35
Manage game server groups	37
Create a game server group	38
Update a game server group	39
Track game server group instances	39
Integrate a game server	39
Register game servers	39
Update game server status	40
Deregister game servers	41
Integrate a Game Client	41
Let Amazon GameLift Servers FleetIQ choose a game server	42
Choose your own game server	42
Monitor with CloudWatch	44
Security with FleetIQ	47
Amazon GameLift Servers FleetIQ reference	48

Service API reference (AWS SDK)	48
Amazon GameLift Servers FleetIQ API actions	48
Available programming languages	50
Release notes and SDK versions	50
All Amazon GameLift Servers guides	50
AWS Glossary	51

What is Amazon GameLift Servers FleetIQ?

Amazon GameLift Servers FleetIQ optimizes the use of low-cost Amazon Elastic Compute Cloud (Amazon EC2) Spot Instances for cloud-based game hosting. With Amazon GameLift Servers FleetIQ, you can work directly with your hosting resources in Amazon EC2 and Amazon EC2 Auto Scaling while taking advantage of Amazon GameLift Servers optimizations to deliver inexpensive, resilient game hosting for your players. Amazon EC2 Spot Instances, although offered at steep discounts, are not generally viable for game hosting because availability fluctuates and there is the potential for [interruptions](#). Amazon GameLift Servers FleetIQ significantly mitigates these limitations, making the use of low-cost Spot Instances viable for game hosting.

FleetIQ optimizations are also available when using Amazon GameLift Servers to manage your game hosting. For information on Amazon GameLift Servers hosting options, see the [Amazon GameLift Servers Developer Guide](#).

The Amazon GameLift Servers FleetIQ game hosting solution is designed for game developers who:

- Have existing AWS deployments or want to use Amazon EC2 directly rather than through the fully managed Amazon GameLift Servers service. Amazon GameLift Servers FleetIQ works with EC2 Auto Scaling groups that you manage in your AWS account, giving you full access to your EC2 instances and groups. You can also integrate with other AWS services, including Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and AWS Shield Advanced.
- Have existing on-premises game hosting and want to extend capacity to the cloud. With Amazon GameLift Servers FleetIQ, you can build a hybrid deployment system that uses your on-premises capacity and incrementally adds AWS cloud capacity as needed.

Ready to start working with Amazon GameLift Servers FleetIQ?

- Learn how to use Amazon GameLift Servers FleetIQ for your game by taking the course [Using Amazon Amazon GameLift Servers FleetIQ for Game Servers](#) on AWS Skill Builder. For an overview of related courses, see the [Game Tech Learning Plan](#). Some courses are available in different languages.
- Follow the instructions in [Amazon GameLift Servers FleetIQ integration steps](#).

How Amazon GameLift Servers FleetIQ works

The Amazon GameLift Servers FleetIQ solution is a game hosting layer that supplements the full set of computing resource management tools that you get with Amazon EC2 and Auto Scaling. In addition to offering a slate of features specific to game hosting, Amazon GameLift Servers FleetIQ provides an extra layer of logic that makes it possible to use low-cost Spot Instances for game hosting. This solution lets you directly manage your Amazon EC2 and Auto Scaling resources and integrate as needed with other AWS services.

When using Amazon GameLift Servers FleetIQ, you prepare to launch Amazon EC2 instances as usual: make an Amazon Machine Image (AMI) with your game server software, create an Amazon EC2 launch template, and define configuration settings for an Auto Scaling group. However, instead of creating an Auto Scaling group directly, you create a Amazon GameLift Servers FleetIQ game server group with your Amazon EC2 and Auto Scaling resources and configuration. This action prompts Amazon GameLift Servers FleetIQ to create both a game server group and a corresponding Auto Scaling group. The game server group is linked to and manages certain aspects of the Auto Scaling group.

After the Auto Scaling group is created, you have full access to your Amazon EC2 and Auto Scaling resources. You can change the configuration of your Auto Scaling groups, add multi-level scaling policies or load balancers, and integrate with other AWS services. You can connect directly to instances in the group. As part of its optimization logic, Amazon GameLift Servers FleetIQ also makes periodic updates to certain Auto Scaling group properties. You can track the availability status of all instances deployed by the Auto Scaling group.

You can temporarily suspend Amazon GameLift Servers FleetIQ activity for a game server group at any time. You also have the option to delete a game server group but retain the corresponding Auto Scaling group.

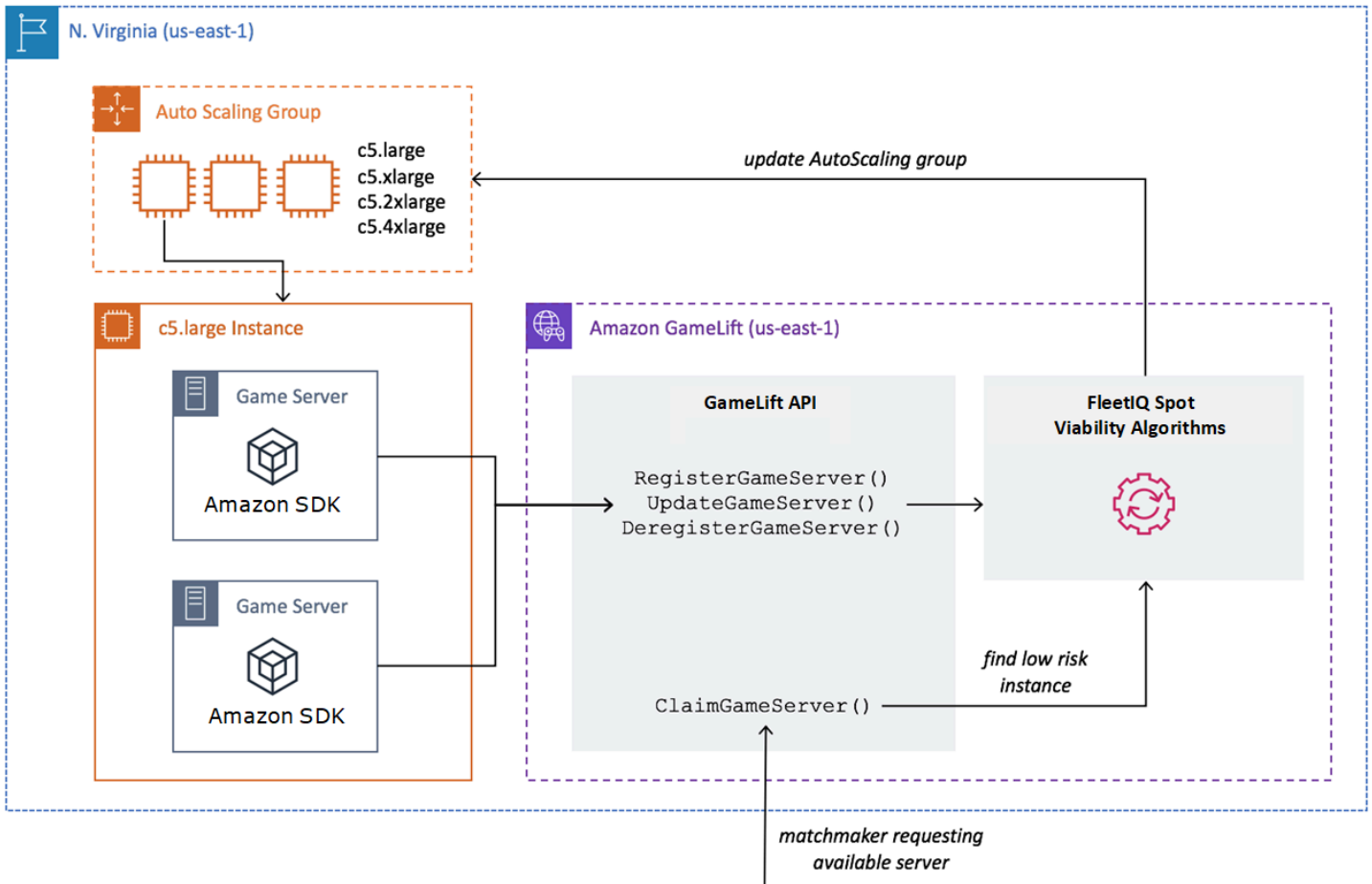
Topics

- [Amazon GameLift Servers FleetIQ logic](#)
- [Key resources and components](#)

Amazon GameLift Servers FleetIQ logic

The following diagram illustrates the role of Amazon GameLift Servers FleetIQ when it is working with Amazon EC2 for game hosting. Its primary goal is to locate the *best* possible game server to host a game session and give players an optimal gameplay experience. Amazon GameLift

Servers FleetIQ defines the *best* resources as those that deliver the highest game hosting viability for the lowest cost. Amazon GameLift Servers FleetIQ approaches this goal in two key ways: first by allowing only viable instance types in the Auto Scaling group, and second by placing new game sessions effectively across the group's available resources.



Fill Auto Scaling group with optimal instance types

The job of the Auto Scaling group is to launch new instances and retire old instances, maintaining a collection of hosting resources and scaling it to meet your player demand. To do this, the Auto Scaling group relies on a list of your desired instance types. The job of Amazon GameLift Servers FleetIQ is to continually check the viability of these desired instance types and update the list for the Auto Scaling group. This process is called instance balancing. It ensures that instances in the Auto Scaling group are continually refreshed so that only currently viable instance types are used at all times.

Amazon GameLift Servers FleetIQ affects how the Auto Scaling group selects optimal instance types in the following ways:

- **It determines usage of Spot and/or On-Demand Instances.** An Amazon GameLift Servers FleetIQ game server group is configured with a balancing strategy, which influences how the Auto Scaling group uses Spot and/or On-Demand Instances. Spot Instances have lower costs due to fluctuating availability and potential [interruptions](#), limitations that Amazon GameLift Servers FleetIQ minimizes for game server hosting. On-Demand instances are more expensive but offer more reliable availability when you need it.
- **It limits new instances to launch on viable instance types only.** A Amazon GameLift Servers FleetIQ game server group maintains a master list of your desired instance types, The instance balancing process continually evaluates each desired instance type on the list for game hosting viability, using a prediction algorithm that looks at the instance type's recent availability and interruption rate. As a result of this evaluation, Amazon GameLift Servers FleetIQ continually updates the Auto Scaling group's list of desired instance types to include only currently viable instances types.
- **It flags existing instances that are non-viable instance types.** Amazon GameLift Servers FleetIQ identifies existing instances in an Auto Scaling group that are currently non-viable instance types. These instances are flagged as *draining*, which means they are terminated and replaced with new instances. For instances that have game server protection turned on, termination is postponed until any active game sessions end normally.

As the Auto Scaling group launches and retires instances, it maintains a collection that is optimized for game hosting even as the availability of low-cost Spot Instance types fluctuates. Balancing activity takes place on game server groups with active instances only. Learn more about how this process works in [Spot balancing process](#).

Place game sessions effectively

Amazon GameLift Servers FleetIQ tracks all active game servers in the game server group and uses this information to determine the best placement for new game sessions and players.

To enable Amazon GameLift Servers FleetIQ to track game servers, your game server software must report its status. Your custom AMI controls how new game servers processes are started and stopped on each instance. When a new game server is started, it registers with Amazon GameLift Servers FleetIQ, indicating that it is ready to host a game session. After registering, the game server periodically reports its health and whether it is currently hosting a game session. When the game server shuts down, it de-registers with Amazon GameLift Servers FleetIQ.

To start a new game session, your game client (or matchmaker or other client service) sends a request for a game server to Amazon GameLift Servers FleetIQ. Amazon GameLift Servers FleetIQ locates an available game server, claims it for the new game session, and responds with the game server ID and connection info. Your game then prompts the game server to update its status and start a new game session for incoming players.

When selecting a game server to host a new game session, Amazon GameLift Servers FleetIQ uses the following decision-making process to optimize placement with viable low-cost Spot Instances:

1. Where possible, Amazon GameLift Servers FleetIQ places new game sessions on instances that are already hosting other game sessions. By packing (but not overloading) some instances and keeping others idle, the Auto Scaling group is able to quickly scale down idle instances when they're not needed, which lowers hosting costs.
2. Amazon GameLift Servers FleetIQ ignores instances that are flagged as *draining*, that is, not viable for game hosting. These instances are kept running only to support existing game sessions. They can't be used for new game sessions unless no other game servers are available.
3. Amazon GameLift Servers FleetIQ identifies all available game servers that are running on viable instances.

You can turn on game session protection for a game server group to prevent the Auto Scaling group from terminating instances with actively running game sessions.

Key resources and components

Create the following resources in your AWS account before you set up your game hosting resources with Amazon GameLift Servers FleetIQ. As a best practice, develop and test your game server deployment with these resources before using them through a game server group.

- **Amazon Machine Image (AMI).** An AMI is a template for a specific software configuration that you want to launch with your Amazon EC2 instances. For game hosting, your AMI includes an operating system, your game server binaries or container, and other runtime software that your game server requires. For more information about creating an AMI, refer to [Amazon Machine Images](#) in the Amazon EC2 User Guide. AMIs are Region-specific. You can copy an AMI from one Region to another, as described in [Copying AMIs](#) in the *Amazon EC2 User Guide*.
- **Amazon EC2 launch template.** A launch template provides instructions for launching and managing instances in an Auto Scaling group. It specifies an AMI, provides a list of suitable instance types, and sets network, security, and other properties. For more information about

creating a launch template, see [Launching an Instance from a Launch Template](#) in the *Amazon EC2 User Guide*. Launch templates are Region-specific.

- **AWS IAM role.** An IAM role defines a set of permissions that allow limited access to AWS resources. A trusted entity, such as another AWS service, can assume the role and inherit its permissions. When using Amazon GameLift Servers FleetIQ, you must provide an IAM role with a managed policy that allows Amazon GameLift Servers FleetIQ to create and access Auto Scaling groups and EC2 instance resources in your AWS account. IAM roles are not Region-specific.

Amazon GameLift Servers FleetIQ manages the following resources directly and has direct authority over them.

- **Amazon GameLift Servers game server group.** A game server group contains configuration settings that define how Amazon GameLift Servers FleetIQ works with a corresponding Auto Scaling group to deliver low-cost game hosting. Game server groups are Region-specific. When you create a game server group in a Region, a new Auto Scaling group is automatically created in your AWS account in the same Region. The game server group is linked to the Auto Scaling group and has access (by assuming the IAM role) to manage and modify some of its settings. A game server group is a long-lived resource; developers should expect to create them infrequently. A game server group is also a functional grouping resource for game servers that are hosted on instances in the Auto Scaling group and registered with Amazon GameLift Servers FleetIQ.
- **Amazon GameLift Servers game server.** A game server resource represents a game execution that is running on an instance associated with a Amazon GameLift Servers FleetIQ game server group. This resource is created when a game server registers with Amazon GameLift Servers FleetIQ and identifies the game server group it belongs to. Amazon GameLift Servers FleetIQ tracks the utilization status and claim status of each registered game server, which enables it to monitor game server availability. Game servers are Region-specific in that they are associated with a Region-specific game server group. When your game requests a new game server, it specifies the game server group and Region.

These resources are created through Amazon GameLift Servers FleetIQ resources. They are created in your AWS account and you have full control of them.

- **Amazon EC2 Auto Scaling group.** An Auto Scaling group launches and manages a collection of EC2 instances, and automatically scales group capacity. With Amazon GameLift Servers FleetIQ, there is a one-to-one relationship between the game server group and the Auto Scaling group. While you can update all settings for an Auto Scaling group, Amazon GameLift Servers FleetIQ

periodically overrides and updates certain settings as part of its logic to balance Spot Instances for game hosting viability. For more information, see [AutoScalingGroup](#) in the *Amazon EC2 Auto Scaling User Guide*. Auto Scaling groups are Region-specific; they are created in the same Region as the game server group.

- **Amazon EC2 Instance.** An instance is a virtual server in the cloud. Instance types have specific hardware configurations that specify compute, memory, disk, and network resources. They are typically launched by an Auto Scaling group with an AMI. Instances can be Spot or On-Demand, depending on availability. With Amazon GameLift Servers FleetIQ, instances run one or multiple game server processes, each of which can host multiple game sessions. Instances are Region-specific in that they are associated with a Region-specific Auto Scaling group.

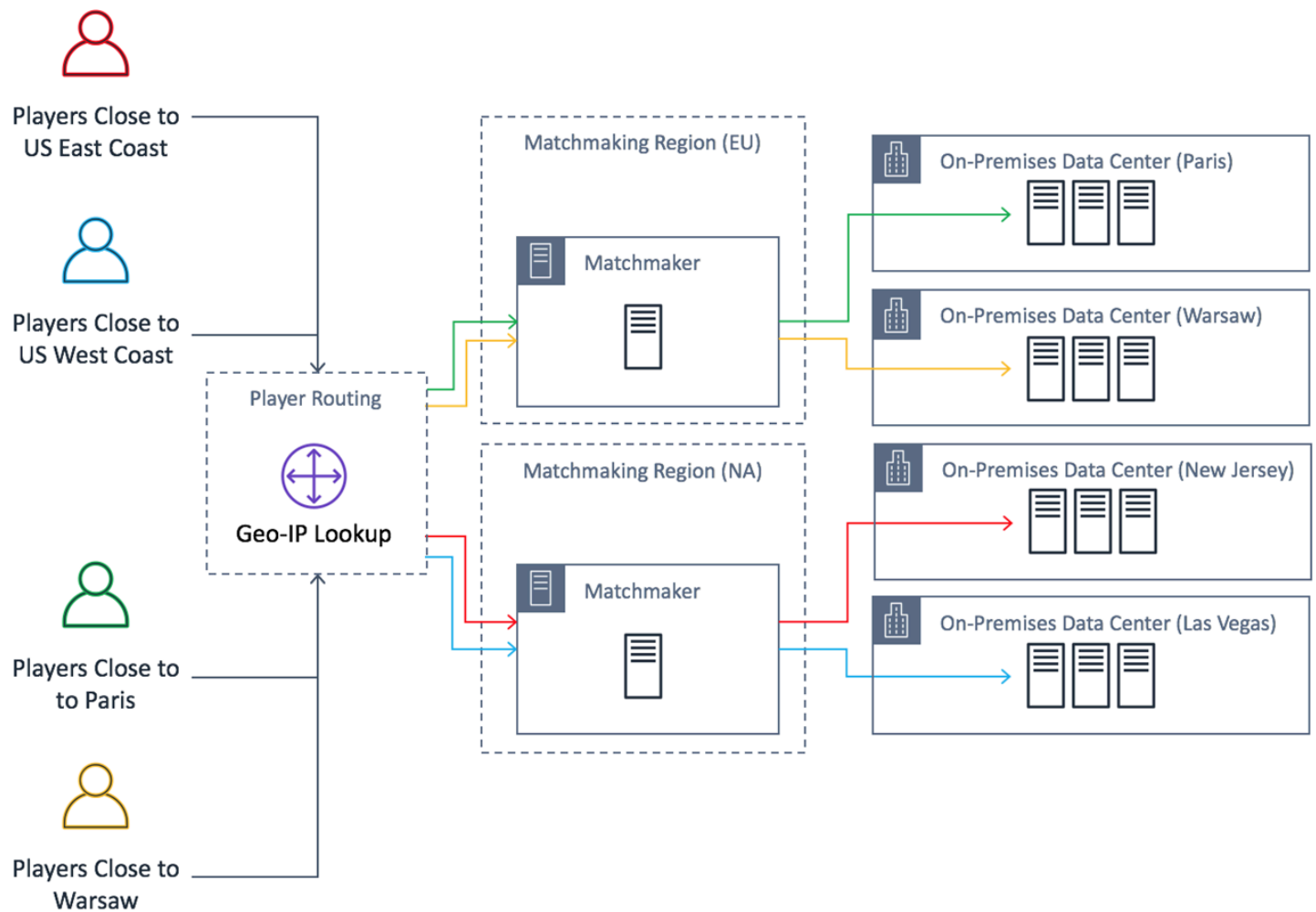
Game architecture with Amazon GameLift Servers FleetIQ

Supplementing on-premises hosting

Amazon GameLift Servers FleetIQ is designed to reuse your existing game backend, including any player geo-IP routing, matchmaking, or lobby services that you might already have in place. The following example illustrates how Amazon GameLift Servers FleetIQ might fit into an existing on-premises deployment.

Example

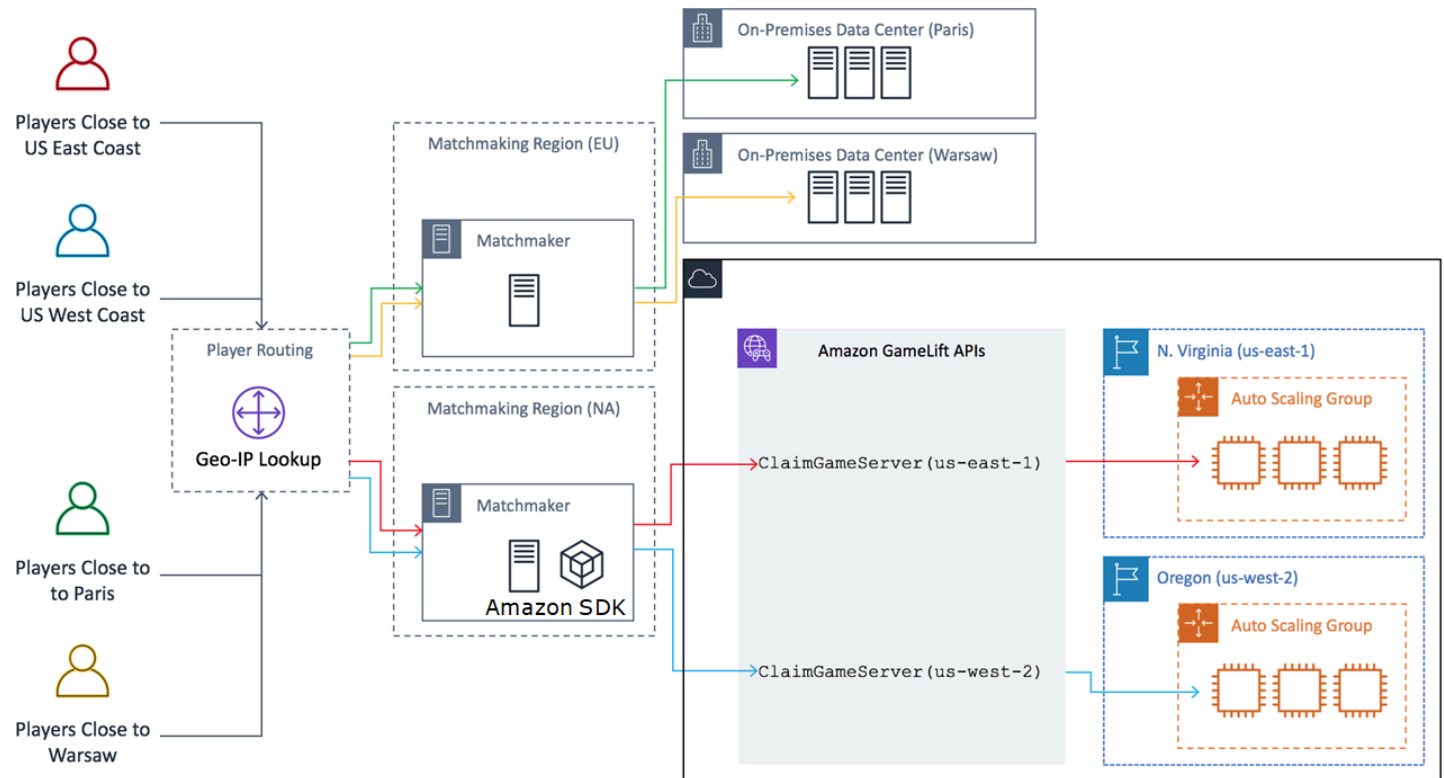
In this example, game hosting is initially handled with four proprietary data centers to host players in North America and Europe. Depending on their approximate physical location, players are routed to one of two regional matchmakers. The matchmakers group players by skill and latency and then place them onto nearby game servers to minimize lag.



The game developer wants to replace their North America game servers with servers provided by Amazon GameLift Servers FleetIQ. To start, they make minor updates to their game server to enable it for use with Amazon GameLift Servers FleetIQ and then create an Amazon Machine Image (AMI). This image will be installed on every EC2 instance that is deployed for the game. The image contains the game server, dependencies, and anything else needed to run game sessions for players.

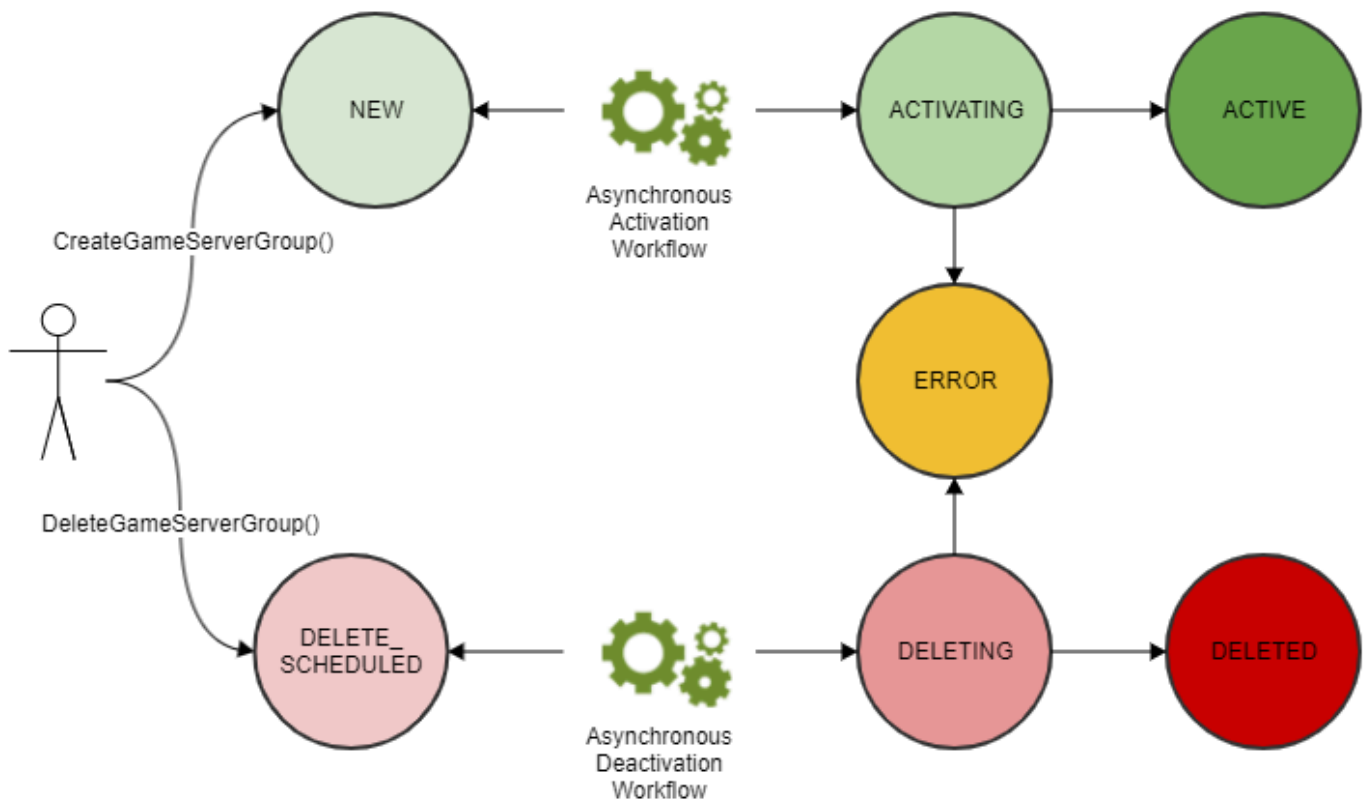
With the AMI ready, the developer creates two Amazon GameLift Servers FleetIQ game server groups, one for each AWS North America Region (`us-east-1` and `us-west-2`). The developer passes in launch template (which provides the AMI), a list of desired instance types, and other configuration settings for the group. The list of desired instance types tells Amazon GameLift Servers FleetIQ which types to use when checking for Spot Instances that are viable for game hosting.

Finally, the developer integrates the AWS SDK with Amazon GameLift Servers FleetIQ into their North American matchmaker, which calls Amazon GameLift Servers FleetIQ when a new group of players needs server capacity for a game session. Amazon GameLift Servers FleetIQ locates a Spot Instance with an available game server, reserves it for the players, and provides server connection information. Players connect to the server, play the game, and disconnect. To start a new game, players re-enter matchmaking, which prompts Amazon GameLift Servers FleetIQ to find another available game server. Each new game request triggers Amazon GameLift Servers FleetIQ to search for and select game servers with a low chance of interruptions. As a result, Amazon GameLift Servers FleetIQ is constantly redirecting players away from game servers that are not viable for game hosting, even as Spot Instance availability fluctuates over time.



Life of a game server group

Game server groups go through the following life cycle, including provisioning and status updates. A game server group is expected to be a long-lived resource.



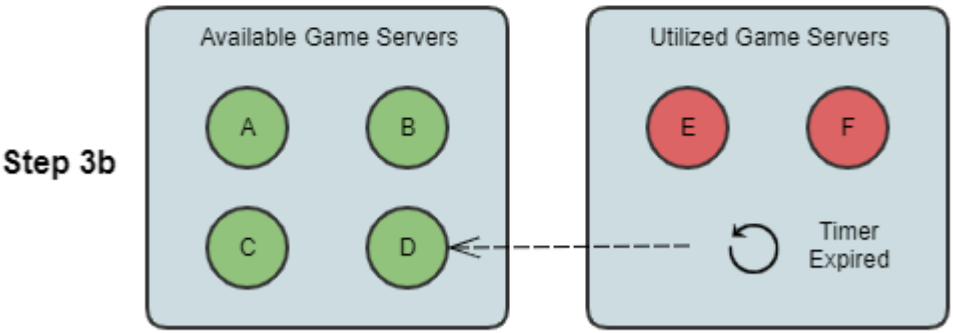
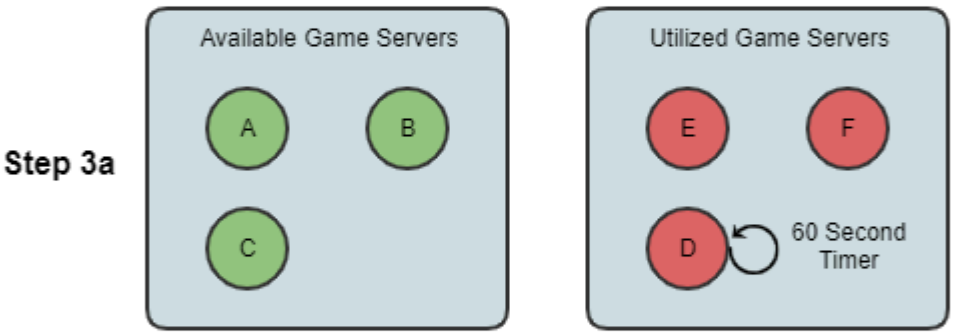
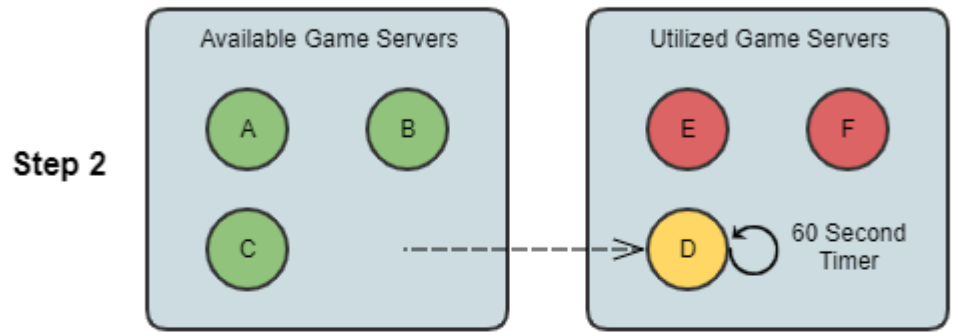
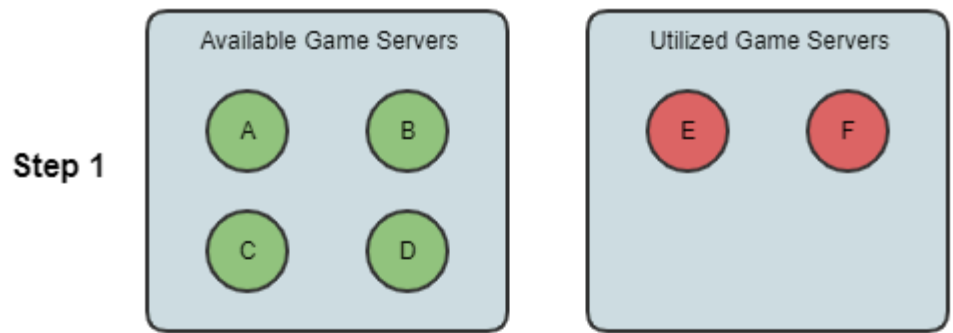
- You create a game server group by calling the Amazon GameLift Servers API `CreateGameServerGroup()` and passing in an EC2 launch template and configuration settings. In response to the call, a new game server group is created and placed in status `NEW`.
- Amazon GameLift Servers FleetIQ activates an asynchronous activation workflow, transitioning the game server group status to `ACTIVATING`. The workflow initiates the creation of underlying resources, including an Amazon EC2 Auto Scaling group and an EC2 instance with the provided AMI.
 - If provisioning fails for any reason, the game server group is placed into status `ERROR`. To get additional error information to help debug the failure cause, call `DescribeGameServerGroup()` on a game server group in an error state.
 - If provisioning succeeds, the game server group is transitioned to status `ACTIVE`. At this point, instances are launched with game servers that register with Amazon GameLift Servers FleetIQ. The group's instance types are periodically evaluated for game hosting viability and balanced as needed. Amazon GameLift Servers FleetIQ also tracks the status of active game servers in the group and responds to requests for game servers.

- You remove a game server group by calling `DeleteGameServerGroup()` with the group identifier. This action puts the game server group into status `DELETE_SCHEDULED`. Only game server groups in `ACTIVE` or `ERROR` state can be scheduled for deletion.
- Amazon GameLift Servers FleetIQ activates an asynchronous deactivation workflow in response to the `DELETE_SCHEDULED` status, transitioning the game server group status to `DELETING`. You have the option of deleting just the game server group or delete both the game server group and the linked Auto Scaling group.
 - If deactivation fails for any reason, the game server group is placed into status `ERROR`. To get additional error information to help debug the failure cause, call `DescribeGameServerGroup()` on a game server group in an error state.
 - If deactivation succeeds, the game server group is transitioned to status `DELETED`.

Life of a game server

With Amazon GameLift Servers FleetIQ, game servers go through the following life cycle, including provisioning and status updates. A game server is expected to be a short-lived resource. As a best practice, game servers should be de-registered after the end of a game session rather than being re-used for another game session. This approach helps ensure that available game servers are always running on the lowest-cost resources that are viable for game hosting.

- A game server resource is created when the game server process, running on an instance in a Amazon GameLift Servers FleetIQ-linked Auto Scaling group, calls the Amazon GameLift Servers API `RegisterGameServer()` to notify Amazon GameLift Servers FleetIQ that it is ready to host players and gameplay. A game server has two statuses to track its current availability:
 - Utilization status tracks whether the game server is currently supporting gameplay. This status is initially set to `AVAILABLE`, indicating that it is ready to accept new gameplay. Once the game server is occupied with gameplay, this status is set to `UTILIZED`.
 - Claim status tracks whether the game server is claimed for imminent gameplay. A game server in `CLAIMED` status indicates that it has been temporarily reserved by a game client (or a game service such as a matchmaker). This status prevents Amazon GameLift Servers FleetIQ from providing the same game server to multiple requesters. A game server with a blank claim status is available to be claimed.
- The following diagram illustrates how a game server's utilization status and claim status change over its life span.



-  Utilization Status is AVAILABLE, no Claim Status
-  Utilization Status is AVAILABLE, Claim Status is CLAIMED
-  Utilization Status is UTILIZED, Claim Status can be either

- **Step 1.** A game server group has six registered game servers. Four have a utilization status AVAILABLE (A, B, C, and D), and two are currently UTILIZED (E and F).
- **Step 2.** A game client or matchmaking system calls the Amazon GameLift Servers API `ClaimGameServer()` to request a new game server. This request prompts Amazon GameLift Servers FleetIQ to search for an available game server (D) and set its claim status to CLAIMED for 60 seconds. Amazon GameLift Servers FleetIQ responds to its request with connection information for the game server (IP address and port), as well as other optional game-specific data. Since gameplay has not yet begun on the game server, its utilization status remains AVAILABLE, but it cannot be claimed with another request.
- **Step 3a.** Using the connection information provided, game clients can connect to the game server and initiate gameplay. The game server (D) must be triggered within 60 seconds to change its utilization status to UTILIZED by calling the Amazon GameLift Servers API `UpdateGameServer()`.
- **Step 3b.** If the game server's utilization status is not updated within 60 seconds, the claim timer expires and the claim status is reset to blank. The game server (D) is returned to the pool of available and unclaimed game servers.
- A game server resource is removed after gameplay on the game server is complete and players have disconnected. Before shutting down, the game server process calls the Amazon GameLift Servers API `DeregisterGameServer()` to notify Amazon GameLift Servers FleetIQ of its departure from the game server group's pool of game servers.

Spot balancing process

Amazon GameLift Servers FleetIQ periodically balances the instances in an Auto Scaling group that has Spot Instances. This process is not active with game server groups that use the `ON_DEMAND_ONLY` balancing strategy or do not have any active instances.

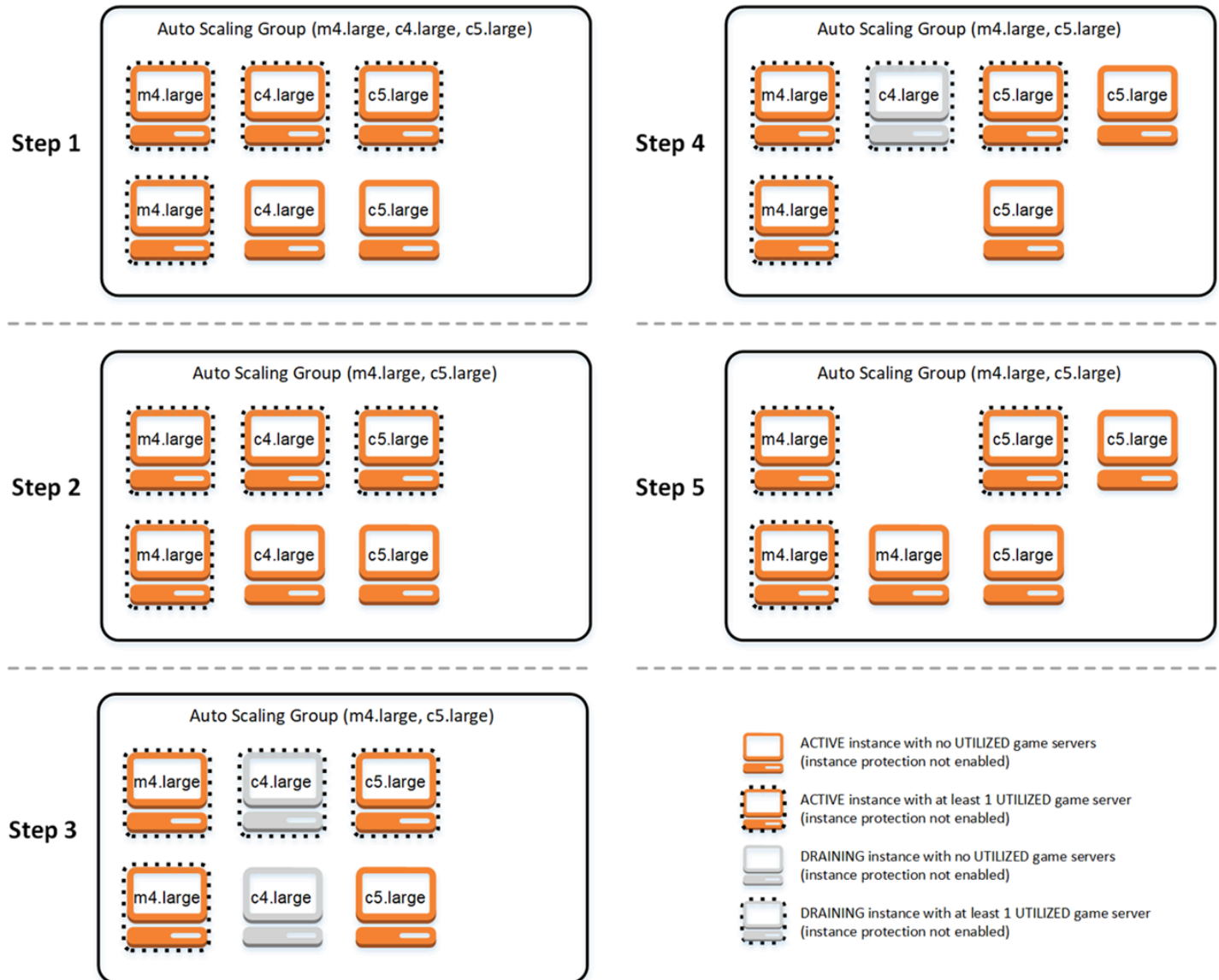
Spot balancing has two key goals:

- To constantly refresh the group by only using Spot Instance types that are viable for game hosting.
- To use multiple viable instance types (where possible) in order to reduce the impact of unexpected game server interruptions.

Amazon GameLift Servers FleetIQ balances by evaluating the group's instance types and removing instances that are more likely to result in game server interruptions. To avoid terminating instances with active gameplay during balancing, best practice is to turn on game server protection for a game server group that's in production.

Example

The following example illustrates how instances in an Auto Scaling group are affected by Spot balancing.



- **Step 1.** Through a game server group, the linked Auto Scaling group is set up to launch instances of types m4.large, c4.large, and c5.large with game server protection enabled. The Auto Scaling group has launched a balanced collection consisting of two Spot Instances of each type. Four

instances have at least one game server in UTILIZED status (shown with a dashed border), while two instances are not currently supporting gameplay.

- **Step 2.** Amazon GameLift Servers FleetIQ evaluates the current game hosting viability of all three instance types. The evaluation determines that the c4.large instance type has an unacceptable potential for game server interruption. Amazon GameLift Servers FleetIQ immediately updates the Auto Scaling group configuration to temporarily remove c4.large from the list of instance types, preventing additional c4.large instances from being launched.
- **Step 3.** Amazon GameLift Servers FleetIQ identifies existing instances of type c4.large and takes actions to remove them from the group. As a first step, all game servers that are running on c4.large instances are flagged as *draining*. Game servers on draining instances can be claimed only as a last resort if no other game servers are available. In addition, an Auto Scaling group with draining instances is triggered to launch new instances to replace them.
- **Step 4.** As new viable instances come online, the Auto Scaling group terminates draining instances. This replacement ensures that the group's desired capacity is maintained. The first instance to be terminated is the c4.large instance with no utilized game servers and game server protection turned off. It is replaced with a new c5.large instance.
- **Step 5.** Draining instances with game server protection continue to run while their game servers are supporting gameplay. When gameplay ends, the remaining c4.large instance is terminated when a new m4.large instance has launched to take its place.

As a result of this process, the Auto Scaling group maintains its desired capacity while the group balances from using three instance types to two. Amazon GameLift Servers FleetIQ continues to evaluate the original list of instance types for game hosting viability. When c4.large is again considered a viable instance type, the Auto Scaling group is updated to include all three instance types. The group naturally balances over time.

Amazon GameLift Servers FleetIQ best practices

Amazon GameLift Servers FleetIQ is a low-level logic layer that helps you manage Amazon EC2 resources for game hosting. In particular, Amazon GameLift Servers FleetIQ optimizes the use of Spot Instances that are viable for game hosting by minimizing the chance that game sessions might be interrupted. It also provides basic game hosting functionality to track available game servers and route gameplay to low-cost, high-viability game servers.

Amazon GameLift Servers FleetIQ as a standalone feature does not provide advanced features that are offered with the fully managed Amazon GameLift Servers solution, which also uses FleetIQ to

minimize hosting costs. If you need features such as matchmaking, latency-based player routing, game session and player session management, and versioning, take a look at the Amazon GameLift Servers solutions.

Here are some best practices that can help you get the most benefit from Amazon GameLift Servers FleetIQ.

- **Use Amazon GameLift Servers FleetIQ for session-based games.** Amazon GameLift Servers FleetIQ works best when it is constantly directing players onto instances that are least likely to have game session interruptions. Maintaining long-lived sessions interferes with the Amazon GameLift Servers FleetIQ balancing process, which increases the likelihood that games sessions might be interrupted. The ideal workflow is for players to go from matchmaking (or server selection) into gameplay. When the game ends, players go back to matchmaking and are routed to another game server on a new instance. We recommend using Amazon GameLift Servers FleetIQ for games with sessions under two hours.
- **Provide many instance types to choose from.** When you set up a game server group, you provide a list of instance types to be used. The more instance types that you include, the greater flexibility Amazon GameLift Servers FleetIQ has to use Spot Instances with high viability for game hosting. For example, you might list multiple sizes within the same instance family (c5.large, c5.xlarge, c5.2xlarge, c5.4xlarge). With larger instances, you can run more game servers on each instance, potentially lowering costs. With smaller instances, autoscaling can react faster to changes in player demand. Keep in mind that the list of desired instance types is not prioritized—an Auto Scaling group will use a balance of viable instance types to maintain the group's resiliency.
- **Test your game on all instance types.** Ensure that your game server runs properly on every instance type that you configure for your game server group.
- **Use instance capacity weighting.** If you configure your game server group to use a range of instance sizes (such as c5.2xlarge, c5.4xlarge, c5.12xlarge), include capacity weighting information for each instance type. For more information, see [Instance Weighting for Amazon EC2 Auto Scaling](#) in the *Amazon EC2 Auto Scaling User Guide*.
- **Place your game sessions using Amazon GameLift Servers FleetIQ.** When placing groups of players with game servers, use the Amazon GameLift Servers API `ClaimGameServer()`. Amazon GameLift Servers FleetIQ avoids placing players on instances with a higher chance of game session interruptions.
- **Report game server status to Amazon GameLift Servers FleetIQ.** Periodically report server health and utilization status with the Amazon GameLift Servers API `UpdateGameServer()`.

Maintaining accurate game server status helps Amazon GameLift Servers FleetIQ place gameplay more efficiently. It also helps avoid terminating instances with active gameplay during Spot balancing activity.

- **Set up an auto scaling policy.** You can create a target-tracking scaling policy that maintains your hosting capacity based on player utilization and anticipated demand. The Amazon GameLift Servers FleetIQ metric `PercentUtilizedGameServers` is a measure of how much of your hosting capacity is currently in use. Most games want to maintain a buffer of unused game servers so that new players can get into a game quickly. You can create a scaling policy that maintains a certain buffer size, adding or removing instances as player demand fluctuates. For more information, see [Target Tracking Scaling Policies](#) in the Amazon EC2 Auto Scaling User Guide.
- **Use different AWS accounts for development and production environments.** Separating your development and production configurations across accounts can reduce the risk of misconfiguration impacting live players.
- **Enable game session protection for game server groups in production.** To protect your players, turn on game session protection and prevent active game sessions from being terminated early due to scaling or balancing activity.
- **Test your game on EC2 before integrating it with Amazon GameLift Servers FleetIQ.** We recommend getting your game up and running on EC2 and fine-tune your configuration first. You can then create a game server group using the same launch template and AMI.

If you're using Kubernetes, we recommend first getting standard EC2 instances added to your Kubernetes cluster, then create a game server group using the launch template you create for worker nodes in your Kubernetes cluster. If you're using EKS, create your EKS cluster and game server group separately. For the game server group, use the EKS-optimized AMI with the appropriate user data and the launch template configuration used for your EKS integration. See more details about EKS worker nodes and the EKS optimized AMI in the [Amazon EKS-Optimized Linux AMI](#) guide.

- **Use the game server group balancing strategy `ON_DEMAND_ONLY` for reliable game server availability.** With this balancing strategy in force, no Spot Instances are used. This is a useful tool to ensure server availability when you need it most, such as during feature launches or other special events. You can switch a game server group from a Spot to an On-Demand strategy as needed.

Also review these AWS best practices:

- [Best Practices for Amazon EC2](#)
- [Best Practices for Amazon EC2 Auto Scaling](#)

Amazon GameLift Servers FleetIQ features

- **Optimized Spot balancing.** Amazon GameLift Servers FleetIQ periodically evaluates your instance types and replaces Spot Instances that are not considered viable due to a higher potential for game session interruptions. As your EC2 Auto Scaling group retires old instances and starts new ones, the group continually refreshes with instance types that are currently viable for game hosting.
- **Optimum player routing.** Amazon GameLift Servers FleetIQ APIs direct new game sessions onto the most resilient Spot Instances, where they are least likely to be interrupted. In addition, game sessions are packed onto fewer instances, which improves the EC2 Auto Scaling group's ability to scale down unneeded resources and lower hosting costs.
- **Automatic scaling based on player usage.** Amazon GameLift Servers FleetIQ emits game server utilization data as Amazon CloudWatch metrics. You can use these metrics to automatically scale your available hosting resources to track with actual player demand and reduce hosting costs.
- **Direct management of Amazon EC2 instances.** Maintain full control of the EC2 instances and EC2 Auto Scaling groups in your AWS account. This means that you can set up instance launch templates, maintain EC2 Auto Scaling group configurations, and integrate with other AWS services. As part of its Spot balancing activity, Amazon GameLift Servers FleetIQ makes periodic updates to some EC2 Auto Scaling group properties. You can temporarily override these settings or suspend Amazon GameLift Servers FleetIQ activity as needed.
- **Support for multiple game server executable formats.** Amazon GameLift Servers FleetIQ supports all formats that currently run on Amazon EC2, including Windows, Linux, containers, and Kubernetes. See the [Amazon EC2 FAQs](#) for a list of supported operating systems and runtimes.
- **Multiple types of hosting resources.** With Amazon GameLift Servers FleetIQ, you have access to a large range of instance types for game server hosting. (Availability varies by AWS Region.) This means that you can pair your game server with the appropriate mix of CPU, memory, storage, and networking capacity to provide the best possible gaming experience for your players.
- **Worldwide reach.** Amazon GameLift Servers FleetIQ is available in 15 Regions, including in China. With this reach, you can make your game servers available with minimal lag to players, wherever they're located. For a complete list of Regions, see [Amazon GameLift Servers endpoints and quotas](#) in the *AWS General Reference*.

Pricing for Amazon GameLift Servers FleetIQ

Amazon GameLift Servers charges for instances by duration of use and for bandwidth by quantity of data transferred. For a complete list of charges and prices for Amazon GameLift Servers, see [Amazon GameLift Servers Pricing](#).

For information on calculating the cost of hosting your games or matchmaking with Amazon GameLift Servers, see [Generating Amazon GameLift Servers pricing estimates](#), which describes how to use the [AWS Pricing Calculator](#).

Amazon GameLift Servers FleetIQ setting up

The topics in this section help with set up tasks, including how to set up your AWS account for use with Amazon Amazon GameLift Servers FleetIQ service.

Topics

- [Amazon GameLift Servers FleetIQ supported software](#)
- [Set up your AWS account for Amazon GameLift Servers FleetIQ](#)

Amazon GameLift Servers FleetIQ supported software

Amazon GameLift Servers FleetIQ is used to deploy 64-bit, multiplayer game servers, clients, and game services for hosting on Amazon EC2. This solution supports the following environments:

Operating systems for game servers

You can use Amazon GameLift Servers FleetIQ with game servers that run on any of the operating systems supported by EC2. This includes Amazon Linux, Ubuntu, Windows Server, Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Fedora, Debian, CentOS, Gentoo Linux, Oracle Linux, and FreeBSD. See current EC2 features and support at [Amazon EC2 features](#).

Use of containers

If your game server uses containers, Amazon GameLift Servers FleetIQ supports integration with Kubernetes, Amazon Elastic Container Service (Amazon ECS), and Amazon Elastic Kubernetes Service (EKS). See more information at [Containers on AWS](#).

Game development environments

Game clients and servers require some integration to communicate with the Amazon GameLift Servers FleetIQ service. Games make API calls to the AWS SDK. [Download the AWS SDK](#) or [view the Amazon GameLift Servers API reference documentation](#).

The AWS SDK with support for Amazon GameLift Servers is available in the following languages. For information about support for development environments, see the documentation for each language.

- C++ ([SDK docs](#)) ([Amazon GameLift Servers](#))
- Java ([SDK docs](#)) ([Amazon GameLift Servers](#))

- .NET ([SDK docs](#)) ([Amazon GameLift Servers](#))
- Go ([SDK docs](#)) ([Amazon GameLift Servers](#))
- Python ([SDK docs](#)) ([Amazon GameLift Servers](#))
- Ruby ([SDK docs](#)) ([Amazon GameLift Servers](#))
- PHP ([SDK docs](#)) ([Amazon GameLift Servers](#))
- JavaScript/Node.js ([SDK docs](#)) ([Amazon GameLift Servers](#))

Set up your AWS account for Amazon GameLift Servers FleetIQ

To use Amazon GameLift Servers FleetIQ with Amazon EC2, Auto Scaling, and other AWS services, you must set up an AWS account with required access permissions. Complete the following tasks:

- If you don't already have an AWS account to use with Amazon GameLift Servers FleetIQ, create a new one. See [Create an AWS account](#).
- Set Amazon GameLift Servers FleetIQ-specific permissions for users and user groups. See [Manage user permissions for Amazon GameLift Servers FleetIQ](#).
- Create IAM roles to allow Amazon GameLift Servers and your Amazon EC2 resources to interact. See [Create IAM roles for cross-service interaction](#).

Create an AWS account

Create and set up an AWS account to use with Amazon GameLift Servers FleetIQ. There's no charge to create an AWS account.

Topics

- [Sign up for an AWS account](#)
- [Create a user with administrative access](#)

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.

2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform [tasks that require root user access](#).

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to <https://aws.amazon.com/> and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the [AWS Management Console](#) as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

For help signing in by using root user, see [Signing in as the root user](#) in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

For instructions, see [Enable a virtual MFA device for your AWS account root user \(console\)](#) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

For instructions, see [Enabling AWS IAM Identity Center](#) in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

For a tutorial about using the IAM Identity Center directory as your identity source, see [Configure user access with the default IAM Identity Center directory](#) in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

For help signing in using an IAM Identity Center user, see [Signing in to the AWS access portal](#) in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

For instructions, see [Create a permission set](#) in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

For instructions, see [Add groups](#) in the *AWS IAM Identity Center User Guide*.

Manage user permissions for Amazon GameLift Servers FleetIQ

Create additional users or extend Amazon GameLift Servers FleetIQ access permissions to existing users as needed. Users who work with Amazon GameLift Servers FleetIQ game server groups and the related Amazon EC2 and Auto Scaling services must have permissions to access these services.

As a best practice ([Security best practices in IAM](#)), apply least-privilege permissions for all users. You can set permissions for individual users or user groups and limit user access by service, action, or resource.

Use following instructions to set user permissions based on how you manage the users in your AWS account. If you use IAM users, as a best practice always attach permissions to roles or user groups, not individual users.

- [Permissions syntax for users](#)

- [Additional permissions syntax for use with CloudFormation](#)

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

Create a permission set. Follow the instructions in [Create a permission set](#) in the *AWS IAM Identity Center User Guide*.

- Users managed in IAM through an identity provider:

Create a role for identity federation. Follow the instructions in [Create a role for a third-party identity provider \(federation\)](#) in the *IAM User Guide*.

- IAM users:

- Create a role that your user can assume. Follow the instructions in [Create a role for an IAM user](#) in the *IAM User Guide*.

- (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in [Adding permissions to a user \(console\)](#) in the *IAM User Guide*.

Reference: Amazon GameLift Servers FleetIQ_policy

The following is an example of the Amazon GameLift Servers FleetIQ_policy for your reference:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEquals": {

```

```

        "iam:PassedToService": "gamelift.amazonaws.com"
    }
}
},
{
    "Action":
    [
        "iam:CreateServiceLinkedRole"
    ],
    "Effect": "Allow",
    "Resource": "arn:*:iam::*:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling"
},
{
    "Action":
    [
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:CreateOrUpdateTags",
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:ExitStandby",
        "autoscaling:PutLifecycleHook",
        "autoscaling:PutScalingPolicy",
        "autoscaling:ResumeProcesses",
        "autoscaling:SetInstanceProtection",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling>DeleteAutoScalingGroup"
    ],
    "Effect": "Allow",
    "Resource": "*"
},
{
    "Action":
    [
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeSubnets",
        "ec2:RunInstances",
        "ec2:CreateTags"
    ],
    "Effect": "Allow",
    "Resource": "*"
},
{
    "Action":
    [

```

```

        "events:PutRule",
        "events:PutTargets"
    ],
    "Effect": "Allow",
    "Resource": "*"
  }
]
}

```

Additional permissions for CloudFormation

If you use CloudFormation to manage your game hosting resources, add the CloudFormation permissions to the policy syntax.

```

{
  "Action": [
    "autoscaling:DescribeLifecycleHooks",
    "autoscaling:DescribeNotificationConfigurations",
    "ec2:DescribeLaunchTemplateVersions"
  ]
  "Effect": "Allow",
  "Resource": "*"
}

```

Set up programmatic access for users

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the	Following the instructions for the interface that you want to use.

Which user needs programmatic access?	To	By
	AWS CLI, AWS SDKs, or AWS APIs.	<ul style="list-style-type: none"> For the AWS CLI, see Login for AWS local development in the <i>AWS Command Line Interface User Guide</i>. For AWS SDKs, see Login for AWS local development in the <i>AWS SDKs and Tools Reference Guide</i>.
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	<p>Following the instructions for the interface that you want to use.</p> <ul style="list-style-type: none"> For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the <i>AWS Command Line Interface User Guide</i>. For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the <i>AWS SDKs and Tools Reference Guide</i>.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	<p>Following the instructions in Using temporary credentials with AWS resources in the <i>IAM User Guide</i>.</p>

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. <ul style="list-style-type: none"> • For the AWS CLI, see Authenticating using IAM user credentials in the <i>AWS Command Line Interface User Guide</i>. • For AWS SDKs and tools, see Authenticate using long-term credentials in the <i>AWS SDKs and Tools Reference Guide</i>. • For AWS APIs, see Managing access keys for IAM users in the <i>IAM User Guide</i>.

If you use access keys, see [Best practices for managing AWS access keys](#).

Create IAM roles for cross-service interaction

In order for Amazon GameLift Servers FleetIQ to work with your Amazon EC2 instances and Auto Scaling groups, you must allow the services to interact with each other. This is done by creating IAM roles in your AWS account and assigning a set of limited permissions. Each role also specifies which services can assume the role.

Set up the following roles:

- [Create a role for Amazon GameLift Servers FleetIQ](#) to update your Amazon EC2 resources.
- [Create a role for Amazon EC2](#) resources to communicate with Amazon GameLift Servers FleetIQ.

Create a role for Amazon GameLift Servers FleetIQ

This role allows Amazon GameLift Servers FleetIQ to access and modify your Amazon EC2 instances, Auto Scaling groups, and lifecycle hooks as part of its Spot balancing and automatic scaling activities.

Use the IAM console or the AWS CLI to create a role for Amazon GameLift Servers FleetIQ and attach a managed policy with the necessary permissions. For more information on IAM roles and managed policies, see [Creating a Role for an AWS Service](#) and [AWS Managed Policies](#).

Console

These steps describe how to create a service role with a managed policy for Amazon GameLift Servers using the AWS Management Console.

1. Open the [IAM console](#) and choose **Roles: Create role**.
2. For **Select type of trusted entity**, choose **AWS service**.
3. For **Choose a use case**, choose **GameLift** from the list of services. Under **Select your use case**, the appropriate Amazon GameLift Servers use case is automatically selected. To continue, choose **Next: Permissions**.
4. The list **Attached permissions policies** should contain one policy: **GameLiftGameServerGroupPolicy**. If this policy is not shown, check the filters or use the search feature to add it to the role. You can view a policy's syntax (choose the ► icon to expand), but you cannot change the syntax. When the role is created, you can update the role and attach additional policies to add or remove permissions.

For **Set permissions boundary**, keep the default setting (Create role without a permissions boundary). This is an advanced setting that is not required. To continue, choose **Next: Tags**.

5. **Add tags** is an optional setting for resource management. For example, you might want to add tags to this role to track project-specific resource usage by role. To see more information on tagging for IAM roles and other uses, follow the **Learn more** link. To continue, choose **Next: Review**.
6. On the **Review** page, make the following changes as needed:
 - Enter a role name and optionally update the description.
 - Verify the following:
 - **Trusted entities** is set to "AWS service: gamelift.amazonaws.com". This value must be updated once the role has been created.

- **Policies** includes GameLiftGameServerGroupPolicy.

To complete the task, choose **Create role**.

7. Once the new role has been created, you must manually update the role's trust relationship. Go to the **Roles** page and choose the new role name to open its summary page. Open the **Trust relationships** tab and choose **Edit trust relationship**. In the policy document, update the **Service** property to include `autoscaling.amazonaws.com`. The revised **Service** property should look like this:

```
"Service": [  
  "gamelift.amazonaws.com",  
  "autoscaling.amazonaws.com"  
]
```

To save your change, choose **Update Trust Policy**.

The role is now ready. Take note of the role's ARN value, which is displayed at the top of the role's summary page. You will need this information when setting up Amazon GameLift Servers FleetIQ game server groups.

AWS CLI

These steps describe how to create a service role with a managed policy for Amazon GameLift Servers using the AWS CLI.

1. Create a trust policy file (example: `FleetIQtrustpolicyGameLift.json`) with the following JSON syntax.

JSON

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "Service": [  
          "gamelift.amazonaws.com",  
          "autoscaling.amazonaws.com"  
        ]  
      }  
    }  
  ]  
}
```

```
    ]
  },
  "Action": "sts:AssumeRole"
}
]
```

2. Create a new IAM role with [iam create-role](#) and associate it with the trust policy JSON file that you just created.

Windows:

```
AWS iam create-role --role-name FleetIQ-role-for-GameLift --assume-role-policy-document file://C:\policies\FleetIQtrustpolicyGameLift.json
```

Linux:

```
AWS iam create-role --role-name FleetIQ-role-for-GameLift --assume-role-policy-document file://policies/FleetIQtrustpolicyGameLift.json
```

When the request is successful, the response includes the properties of the newly created role. Take note of the ARN value. You will need this information when setting up Amazon GameLift Servers FleetIQ game server groups.

3. Use [iam attach-role-policy](#) to attach the managed permissions policy "GameLiftGameServerGroupPolicy".

```
AWS iam attach-role-policy --role-name FleetIQ-role-for-GameLift --policy-arn arn:aws:iam::aws:policy/GameLiftGameServerGroupPolicy
```

To verify that the permissions policy is attached, call [iam list-attached-role-policies](#) with the new role's name.

The role is now ready. You can verify that the IAM role is configured correctly by calling [gamelift create-game-server-group](#) with the `role-arn` property set to the new role's ARN value. When the `GameServerGroup` enters `ACTIVE` state, this indicates that Amazon GameLift Servers FleetIQ is able to modify Amazon EC2 and Auto Scaling resources in your account, as expected.

Create a role for Amazon EC2

This role enables your Amazon EC2 resources to communicate with Amazon GameLift Servers FleetIQ. For example, your game servers, which are running on Amazon EC2 instances, need to be able to report health status. Include this role in an IAM instance profile with your Amazon EC2 launch template when creating a Amazon GameLift Servers FleetIQ game server group.

Use the AWS CLI to create a role for Amazon EC2, attach a custom policy with the necessary permissions, and attach the role to an instance profile. For more information, see [Creating a Role for an AWS Service](#).

AWS CLI

These steps describe how to create a service role with custom Amazon GameLift Servers permissions for Amazon EC2 using the AWS CLI.

1. Create a trust policy file (example: `FleetIQtrustpolicyEC2.json`) with the following JSON syntax.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Create a new IAM role with [iam create-role](#) and associate it with the trust policy JSON file that you just created.

Windows:

```
AWS iam create-role --role-name FleetIQ-role-for-EC2 --assume-role-policy-document file://C:\policies\FleetIQtrustpolicyEC2.json
```

Linux:

```
AWS iam create-role --role-name FleetIQ-role-for-EC2 --assume-role-policy-document file://policies/FleetIQtrustpolicyEC2.json
```

When the request is successful, the response includes the properties of the newly created role. Take note of the ARN value. You will need this information when setting up your Amazon EC2 launch template.

3. Create a permissions policy file (example: FleetIQpermissionsEC2.json) with the following JSON syntax.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "gamelift:*",
      "Resource": "*"
    }
  ]
}
```

4. Use [iam put-role-policy](#) to attach the permissions policy JSON file, which you just created, to the new role.

Windows:

```
AWS iam put-role-policy --role-name FleetIQ-role-for-EC2 --policy-name FleetIQ-permissions-for-EC2 --policy-document file://C:\policies\FleetIQpermissionsEC2.json
```

Linux:

```
AWS iam put-role-policy --role-name FleetIQ-role-for-EC2 --policy-name FleetIQ-permissions-for-EC2 --policy-document file://policies/FleetIQpermissionsEC2.json
```

To verify that the permissions policy is attached, call [iam list-role-policies](#) with the new role's name.

5. Create an instance profile with [iam create-instance-profile](#) with the new role for use with Amazon EC2. For more information, see [Managing Instance Profiles](#).

```
AWS iam create-instance-profile --instance-profile-name FleetIQ-role-for-EC2
```

When the request is successful, the response includes the properties of the newly created instance profile.

6. Use [iam add-role-to-instance-profile](#) to attach the role to the instance profile.

```
AWS iam add-role-to-instance-profile --role-name FleetIQ-role-for-EC2 --instance-profile-name FleetIQ-role-for-EC2
```

The role and profile is now ready to be used with an Amazon EC2 launch template.

Preparing games for Amazon GameLift Servers FleetIQ

This section covers how to implement your design for hosting games on Amazon EC2 with Amazon GameLift Servers FleetIQ. To get your multiplayer games up and running, you need to do the following:

- Adapt your game server to communicate with Amazon GameLift Servers FleetIQ.
- Create a FleetIQ game server group to deploy your game servers.
- Add functionality to your game client service to request available game servers.

The topics in this section provide detailed information on how to accomplish this work. To get started, see the integration plan, which provides a detailed step-by-step guide.

Topics

- [Amazon GameLift Servers FleetIQ integration steps](#)
- [Manage Amazon GameLift Servers FleetIQ game server groups](#)
- [Integrate Amazon GameLift Servers FleetIQ into a game server](#)
- [Integrate Amazon GameLift Servers FleetIQ into a game client](#)

Amazon GameLift Servers FleetIQ integration steps

This integration plan outlines the key steps to getting your multiplayer games up and running on Amazon EC2 instances with Amazon GameLift Servers FleetIQ. If you're looking for the Amazon GameLift Servers managed hosting service, which automates more game hosting processes for you, see the [Amazon GameLift Servers Developer Guide](#).

To get started using Amazon GameLift Servers FleetIQ, you need to have a working game server that runs in either an on-premises or Amazon EC2 environment. Your game server can be a single process that manages one or multiple game sessions, spawns child processes, or runs inside of a container.

1. Get an [AWS account](#) and set up users with Amazon GameLift Servers FleetIQ access.

Create a new AWS account or choose an existing account to use with Amazon GameLift Servers FleetIQ. Set up users with permissions to manage the Amazon EC2, Auto Scaling, and

other AWS resources used with your game. For detailed instructions, see [Set up your AWS account for Amazon GameLift Servers FleetIQ](#).

2. Create IAM roles.

Create roles that allow Amazon GameLift Servers FleetIQ, Amazon EC2, and Auto Scaling resources to communicate with each other. See [Create IAM roles for cross-service interaction](#) for more details.

3. Get the AWS SDK and AWS CLI with Amazon GameLift Servers FleetIQ functionality.

- [Download the latest version of the AWS SDK](#).
- [View the Amazon GameLift Servers API reference documentation](#).

4. Prepare your game server for use with Amazon GameLift Servers FleetIQ.

Add the AWS SDK to your game server project and add code to keep Amazon GameLift Servers FleetIQ updated with the current status and usage of your game servers. See [the section called "Integrate a game server"](#) for additional guidance and examples. Amazon GameLift Servers FleetIQ uses this information to provide your matchmaking system with a list of viable, unoccupied game servers, and also avoid terminating instances that are currently hosting players during balancing.

5. Create an Amazon EC2 Amazon Machine Image (AMI) with your game server.

Create an AMI with your game server software and with any other runtime assets or configuration settings. For help, see [Amazon Machine Images \(AMI\)](#) in the *Amazon EC2 User Guide*.

6. Create an Amazon EC2 launch template.

Build an Amazon EC2 launch template that uses your custom AMI and defines network and security settings for your hosting resources. The launch template must reference the instance profile that you created (see Step 2) with permissions that allow your game server to communicate with Amazon GameLift Servers FleetIQ. You don't need to include instance types in your launch template, as this is done later. For help, see [Creating a Launch Template](#) in the *Amazon EC2 User Guide*.

Note

Before using a launch template with Amazon GameLift Servers FleetIQ, we highly recommend that you first set up an Auto Scaling group to verify that the template configuration and AMI are deploying properly.

7. Set up Amazon GameLift Servers FleetIQ hosting resources.

In each Region where you want to deploy game servers, create a game server group by calling [CreateGameServerGroup\(\)](#). Pass in the launch template (containing your custom AMI and network and security settings), IAM role, and a list of instance types that your game can run on. This action sets up an Auto Scaling group in your AWS account that Amazon GameLift Servers FleetIQ can modify. For additional guidance and examples, see [Manage Amazon GameLift Servers FleetIQ game server groups](#).

8. Integrate Amazon GameLift Servers FleetIQ into your game client.

Add the AWS SDK to your game client, matchmaker, or other backend component that allocates game server capacity. Depending on your game type, your matchmaker might call [ListGameServers\(\)](#) or [ClaimGameServer\(\)](#) to obtain server capacity and reserve an available game server. For additional guidance and examples, see [Integrate Amazon GameLift Servers FleetIQ into a game client](#).

9. Scale up your Auto Scaling group.

As instances are provisioned in your Auto Scaling group, they launch your game servers. Each game server then registers with Amazon GameLift Servers FleetIQ as available capacity, to be listed or claimed later by your matchmaker.

10. Test your game.

Invoke your matchmaker and call `ClaimGameServer` to request server capacity. Pass the resulting IP and port back to game clients so they can connect to the game server.

Manage Amazon GameLift Servers FleetIQ game server groups

This topic describes the tasks required to set up a Amazon GameLift Servers FleetIQ game server group. Creating a game server group triggers the creation of an EC2 Auto Scaling group with

all the necessary configuration settings, along with configuration to manage Amazon GameLift Servers FleetIQ optimizations for game hosting.

Before you can create a game server group, you must at minimum have the following resources prepared:

- An Amazon EC2 launch template that specifies how to launch Amazon EC2 instances with your game server build. For more information, see [Launching an Instance from a Launch Template](#) in the *Amazon EC2 User Guide*.
- An IAM role that extends limited access to your AWS account to allow Amazon GameLift Servers FleetIQ to create and interact with the Auto Scaling group. For more information, see [Create IAM roles for cross-service interaction](#).

Create a game server group

To create a game server group, call [CreateGameServerGroup\(\)](#). This operation creates both a Amazon GameLift Servers FleetIQ game server group and a corresponding Auto Scaling group. When you create the game server group, you provide game-specific settings for Amazon GameLift Servers FleetIQ, including balancing strategy and instance type definitions. You also provide initial property settings for the Auto Scaling group.

The following example triggers the creation of a `GameServerGroup` that specifies `c4.large` and `c5.large` instance types and limits the group to Spot Instances only, and an Auto Scaling group that uses the specified launch template for deploying instances and manages group capacity within the minimum and maximum settings using a target-tracking automatic scaling policy. After a short provisioning period, an `AutoScalingGroup` resource is created, and the `GameServerGroup` enters an `ACTIVE` state.

```
AWS gamelift create-game-server-group \  
  --game-server-group-name MyLiveGroup \  
  --role-arn arn:aws:iam::123456789012:role/GameLiftGSGRole \  
  --min-size 1 \  
  --max-size 10 \  
  --game-server-protection-policy FULL_PROTECTION \  
  --balancing-strategy SPOT_ONLY \  
  --launch-template LaunchTemplateId=lt-012ab345cde6789ff \  
  --instance-definitions '[{"InstanceType": "c4.large"}, {"InstanceType":  
"c5.large"}]' \  
  --auto-scaling-policy '{"TargetTrackingConfiguration": {"TargetValue": 66}}'
```

Update a game server group

You can update game server group properties that affect how Amazon GameLift Servers FleetIQ manages hosting for game servers, including resource type optimizations. To update these properties, call [UpdateGameServerGroup\(\)](#). After the changes to the game server group take effect, Amazon GameLift Servers FleetIQ may overwrite certain properties in the Auto Scaling group.

For all other Auto Scaling group properties, such as `MinSize`, `MaxSize`, and `LaunchTemplate`, you can modify these directly on the Auto Scaling group.

In the example below, the instance type definitions are updated to switch over to `c4.xlarge` and `c5.xlarge` instances types.

```
AWS gamelift update-game-server-group \  
  --game-server-group-name MyLiveGroup \  
  --instance-definitions '[{"InstanceType": "c4.xlarge"}, {"InstanceType":  
  "c5.xlarge"}]'
```

Track game server group instances

After you create and deploy instances to your game server group and Auto Scaling group, you can track the status of game server instances by calling [DescribeGameServerInstances\(\)](#). You can use this operation to track instance status.. For more information on game server group status, see [Life of a game server group](#).

You can also use the [Amazon GameLift Servers console](#), under **Game server groups**, to monitor the status of your game server groups.

Integrate Amazon GameLift Servers FleetIQ into a game server

This topic describes the tasks required to prepare your game server project to communicate with Amazon GameLift Servers FleetIQ. Refer to [Amazon GameLift Servers FleetIQ best practices](#) for additional guidance.

Register game servers

When a game server process is launched and ready to host live gameplay, it must register with Amazon GameLift Servers FleetIQ by calling [RegisterGameServer\(\)](#). Registering allows Amazon

GameLift Servers FleetIQ to respond to matchmaking systems or other client services when they request information on server capacity or claim a game server. When registering, the game server can provide Amazon GameLift Servers FleetIQ with relevant game server data and connection information, including the port and IP address that it uses for inbound client connections.

```
AWS gamelift register-game-server \  
  --game-server-id UniqueId-1234 \  
  --game-server-group-name MyLiveGroup \  
  --instance-id i-1234567890 \  
  --connection-info "1.2.3.4:123" \  
  --game-server-data "{\"key\": \"value\"}"
```

Update game server status

Once a game server is registered, it should regularly report health and utilization status in order to keep the state of server capacity in sync on Amazon GameLift Servers FleetIQ. Report health and utilization status by calling [UpdateGameServer\(\)](#). In the example below, the game server is reporting that it is healthy and is not currently occupied with hosting players or gameplay.

```
AWS gamelift update-game-server \  
  --game-server-group-name MyLiveGroup \  
  --game-server-id UniqueId-1234 \  
  --health-check HEALTHY \  
  --utilization-status AVAILABLE
```

Health status

If your game server has a mechanism for tracking health status, you can use this mechanism to trigger a game server health update to Amazon GameLift Servers FleetIQ.

Utilization status

Reporting game server utilization status keeps Amazon GameLift Servers FleetIQ informed on which game servers are currently ideal and available for new game sessions. Your game server must have a mechanism that triggers a utilization status update to Amazon GameLift Servers FleetIQ. For example, you might trigger the update when players connect to the game server or when a game session starts.

When starting a game session, client or matchmaking services claim an available game server (by calling [ClaimGameServer\(\)](#)), prompt players to connect to the game server, and trigger the game

server to start gameplay. This process is described in [Integrate Amazon GameLift Servers FleetIQ into a game client](#). A game server "claim" is valid for 60 seconds, and the game server must be able to update utilization status within this window. If utilization status is not updated, Amazon GameLift Servers FleetIQ removes the claim, assumes that the game server is available, and may reserve the game server for another client claim request.

```
AWS gamelift update-game-server \  
  --game-server-group-name MyLiveGroup \  
  --game-server-id UniqueId-1234 \  
  --health-check HEALTHY \  
  --utilization-status UTILIZED
```

Deregister game servers

When a game concludes, the game server must deregister from Amazon GameLift Servers FleetIQ using [DeregisterGameServer\(\)](#).

```
AWS gamelift deregister-game-server \  
  --game-server-group-name MyLiveGroup \  
  --game-server-id UniqueId-1234
```

Integrate Amazon GameLift Servers FleetIQ into a game client

This topic describes the tasks required to prepare your game client or matchmaking service to communicate with Amazon GameLift Servers FleetIQ in order to acquire a game server to host a game session.

Create a method that allows your game client or matchmaker to request a game server resource for players. You have a couple of options for how to do this:

- Have Amazon GameLift Servers FleetIQ choose an available game server. This option takes advantage of Amazon GameLift Servers FleetIQ optimizations to use low-cost Spot Instances and for automatic scaling.
- Request all available game servers and select one to use (often referred to as "list and pick").

Topics

- [Let Amazon GameLift Servers FleetIQ choose a game server](#)

- [Choose your own game server](#)

Let Amazon GameLift Servers FleetIQ choose a game server

To have Amazon GameLift Servers FleetIQ choose an available game server, call [ClaimGameServer\(\)](#) without specifying a game server ID. In this scenario, Amazon GameLift Servers FleetIQ does exercise its logic to find a game server on an instance that is viable for game hosting and optimized for automatic scaling.

```
AWS gamelift claim-game-server \  
  --game-server-group-name MyLiveGroup
```

In response to a claim request, Amazon GameLift Servers FleetIQ identifies the `GameServer` resource, connection information, and game data, which clients can use to connect to the game server. The game server's claim status is set to `CLAIMED` for 60 seconds. Either your game server or client service needs to update the game server's status on Amazon GameLift Servers FleetIQ after players connect or gameplay starts. This ensures that Amazon GameLift Servers FleetIQ does not provide this game server in response to subsequent requests for game server capacity. Update game server status by calling [UpdateGameServer\(\)](#).

```
AWS gamelift update-game-server \  
  --game-server-group-name MyLiveGroup \  
  --game-server-id UniqueId-1234 \  
  --health-check HEALTHY \  
  --utilization-status UTILIZED
```

Choose your own game server

With the "list and pick" method, your game client or matchmaker requests a list of available game servers by calling [ListGameServers\(\)](#). You might want to use game server data to provide additional information that players or your matchmaker can use when selecting a game server. To control how results are returned, you can request paginated results and sort game servers by registration date. The following request returns 20 active and available game servers in the specified game server group, sorted by registration time with the newest game servers listed first.

```
AWS gamelift list-game-servers \  
  --game-server-group-name MyLiveGroup \  
  --limit 20 \  
  --registration-date-order DESC
```

```
--sort-order DESCENDING
```

Based on the list of available game servers, the client or matchmaking service selects a game server and claims it by calling [ClaimGameServer\(\)](#) with the specific game server ID. In this scenario, Amazon GameLift Servers FleetIQ does not exercise any of its instance type optimization logic, as described in [Amazon GameLift Servers FleetIQ logic](#).

```
AWS gamelift claim-game-server \  
  --game-server-group-name MyLiveGroup \  
  --game-server-id UniqueId-1234
```

Monitor Amazon GameLift Servers FleetIQ with Amazon CloudWatch

Use Amazon CloudWatch metrics to scale your instance capacity, build operations dashboards, and trigger alarming. Amazon GameLift Servers FleetIQ as a standalone solution emits a set of Amazon CloudWatch metrics to your AWS account. Also see [Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch](#) in the *Amazon EC2 Auto Scaling User Guide*.

The FleetIQ metrics are listed here. See complete Amazon CloudWatch metric information for Amazon GameLift Servers at [Amazon GameLift Servers metrics](#).

Metric	Description
AvailableGameServers	<p>Game servers that are available to run a game execution and are not currently occupied with gameplay. This number includes game servers that have been claimed but are still in AVAILABLE status.</p> <p>Units: Count</p> <p>Relevant Amazon CloudWatch statistics: Sum</p> <p>Dimensions: GameServerGroup</p>
UtilizedGameServers	<p>Game servers that are currently occupied with gameplay. This number includes game servers that are in UTILIZED status.</p> <p>Units: Count</p> <p>Relevant Amazon CloudWatch statistics: Sum</p> <p>Dimensions: GameServerGroup</p>
DrainingAvailableGameServers	<p>Game servers on instances scheduled for termination that are currently not supporting gameplay. These game servers are the lowest priority to be claimed in response to a new claim request.</p>

Metric	Description
	<p>Units: Count</p> <p>Relevant Amazon CloudWatch statistics: Sum</p> <p>Dimensions: GameServerGroup</p>
DrainingUtilizedGameServers	<p>Game servers on instances scheduled for termination that are currently supporting gameplay.</p> <p>Units: Count</p> <p>Relevant Amazon CloudWatch statistics: Sum</p> <p>Dimensions: GameServerGroup</p>
PercentUtilizedGameServers	<p>Portion of game servers that are currently supporting game executions. This metric indicates the amount of game server capacity that is currently in use. It is useful for driving an Auto Scaling policy that can dynamically add and remove instances to match with player demand.</p> <p>Units: Percent</p> <p>Relevant Amazon CloudWatch statistics: Average, Minimum, Maximum</p> <p>Dimensions: GameServerGroup</p>
GameServerInterruptions	<p>Game servers on Spot Instances that were interrupted due to limited Spot availability.</p> <p>Units: Count</p> <p>Relevant Amazon CloudWatch statistics: Sum</p> <p>Dimensions: GameServerGroup, InstanceType</p>

Metric	Description
InstanceInterruptions	<p>Spot Instances that were interrupted due to limited availability.</p> <p>Units: Count</p> <p>Relevant Amazon CloudWatch statistics: Sum</p> <p>Dimensions: GameServerGroup, InstanceType</p>

Security with Amazon GameLift Servers FleetIQ

If you're using Amazon GameLift Servers FleetIQ as a standalone feature with Amazon EC2, see [Security in Amazon EC2](#) in the *Amazon EC2 User Guide*.

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. For information on how to apply the shared responsibility model when using Amazon GameLift Servers FleetIQ, see [Security in Amazon GameLift Servers](#).

Amazon GameLift Servers FleetIQ reference guides

This section contains reference documentation for use with Amazon GameLift Servers FleetIQ.

Topics

- [Amazon GameLift Servers FleetIQ service API reference \(AWS SDK\)](#)
- [Amazon GameLift Servers FleetIQ release notes and SDK versions](#)
- [Amazon GameLift Servers developer resources](#)

Amazon GameLift Servers FleetIQ service API reference (AWS SDK)

This topic provides a task-based list of API actions for Amazon GameLift Servers FleetIQ. The Amazon GameLift Servers FleetIQ service API is packaged into the AWS SDK in the `aws.gameLift` namespace. [Download the AWS SDK](#) or [view the Amazon GameLift Servers API reference documentation](#).

Amazon GameLift Servers FleetIQ optimizes the use of low-cost Spot Instances for cloud-based game hosting with Amazon EC2. See the [Amazon GameLift Servers Developer Guide](#) for more information on other Amazon GameLift Servers hosting options.

Topics

- [Amazon GameLift Servers FleetIQ API actions](#)
- [Available programming languages](#)

Amazon GameLift Servers FleetIQ API actions

The following operations allow you to manage your Amazon GameLift Servers FleetIQ resources, including game server groups and game servers, in conjunction with Amazon EC2 and Auto Scaling groups.

Manage game server groups

Use these operations to manage your game server deployments with FleetIQ optimizations. A game server group controls how your game server processes are launched on Amazon EC2 instances, sets up an Auto Scaling group, and defines how to apply FleetIQ optimizations.

- [CreateGameServerGroup](#) – Create a new game server group and corresponding Auto Scaling group, and begin launching instances to host your game server. CLI command: [create-game-server-group](#)
- [ListGameServerGroups](#) – Get a list of all game server groups in an Amazon GameLift Servers region. CLI command: [list-game-server-groups](#)
- [DescribeGameServerGroup](#) – Retrieve metadata for a game server group. CLI command: [describe-game-server-group](#)
- [UpdateGameServerGroup](#) – Change game server group metadata. CLI command: [update-game-server-group](#)
- [DeleteGameServerGroup](#) – Permanently remove a game server group and terminate FleetIQ activity for the associated hosting resources. CLI command: [delete-game-server-group](#)
- [ResumeGameServerGroup](#) – Reinststate suspended FleetIQ activity for a game server group. CLI command: [resume-game-server-group](#)
- [SuspendGameServerGroup](#) – Temporarily stop FleetIQ activity for a game server group. CLI command: [suspend-game-server-group](#)

Manage game servers

Use these operations to manage your game server deployments with FleetIQ optimizations. A game server group controls how your game server processes are launched on Amazon EC2 instances, sets up and Auto Scaling group, and defines how to apply FleetIQ optimizations.

- [RegisterGameServer](#) – Call from a new game server to notify Amazon GameLift Servers FleetIQ that the game server is ready to host gameplay. CLI command: [register-game-server-group](#)
- [ListGameServers](#) – Call from a game client service to get a list of all game servers that are currently running in a game server group. CLI command: [list-game-servers](#)
- [ClaimGameServer](#) – Call from a game client service to locate and reserve a game server to host a new game session. CLI command: [claim-game-server](#)
- [DescribeGameServer](#) – Retrieve metadata for a game server. CLI command: [describe-game-server](#)
- [UpdateGameServer](#) – Change game server metadata, health status, or utilization status. CLI command: [update-game-server](#)
- [DeregisterGameServer](#) – Call from a terminating game server to prompt Amazon GameLift Servers FleetIQ to remove the game server from the game server group. CLI command: [deregister-game-server](#)

Available programming languages

The AWS SDK with support for Amazon GameLift Servers is available in the following languages. For information about support for development environments, see the documentation for each language.

- C++ ([SDK docs](#)) ([Amazon GameLift Servers](#))
- Java ([SDK docs](#)) ([Amazon GameLift Servers](#))
- .NET ([SDK docs](#)) ([Amazon GameLift Servers](#))
- Go ([SDK docs](#)) ([Amazon GameLift Servers](#))
- Python ([SDK docs](#)) ([Amazon GameLift Servers](#))
- Ruby ([SDK docs](#)) ([Amazon GameLift Servers](#))
- PHP ([SDK docs](#)) ([Amazon GameLift Servers](#))
- JavaScript/Node.js ([SDK docs](#)) ([Amazon GameLift Servers](#))

Amazon GameLift Servers FleetIQ release notes and SDK versions

The Amazon GameLift Servers release notes provide details about new FleetIQ features, updates, and fixes related to the service. This page also includes Amazon GameLift Servers SDK version history.

Amazon GameLift Servers developer resources

To view all Amazon GameLift Servers documentation and developer resources, see the [Amazon GameLift Servers Documentation](#) home page.

AWS Glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS Glossary Reference*.