亚马逊云科技

FleetIQ Developer Guide

Amazon GameLift

Version

Amazon GameLift: FleetIQ Developer Guide

Table of Contents

What is Amazon GameLift FleetIQ?

Amazon GameLift FleetIQ optimizes the use of low-cost Amazon Elastic Compute Cloud (Amazon EC2) Spot Instances for cloud-based game hosting. With Amazon GameLift FleetIQ, you can work directly with your hosting resources in Amazon EC2 and Amazon EC2 Auto Scaling while taking advantage of Amazon GameLift 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 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 to manage your game hosting. For information on Amazon GameLift hosting options, see the Amazon GameLift Developer Guide.

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

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

Ready to start working with Amazon GameLift FleetIQ?

- Learn how to use Amazon GameLift FleetIQ for your game by taking the course Using Amazon Amazon GameLift FleetIQ for Game Servers on Amazon 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 FleetIQ integration steps.

How Amazon GameLift FleetIQ works

The Amazon GameLift 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 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 Amazon services.

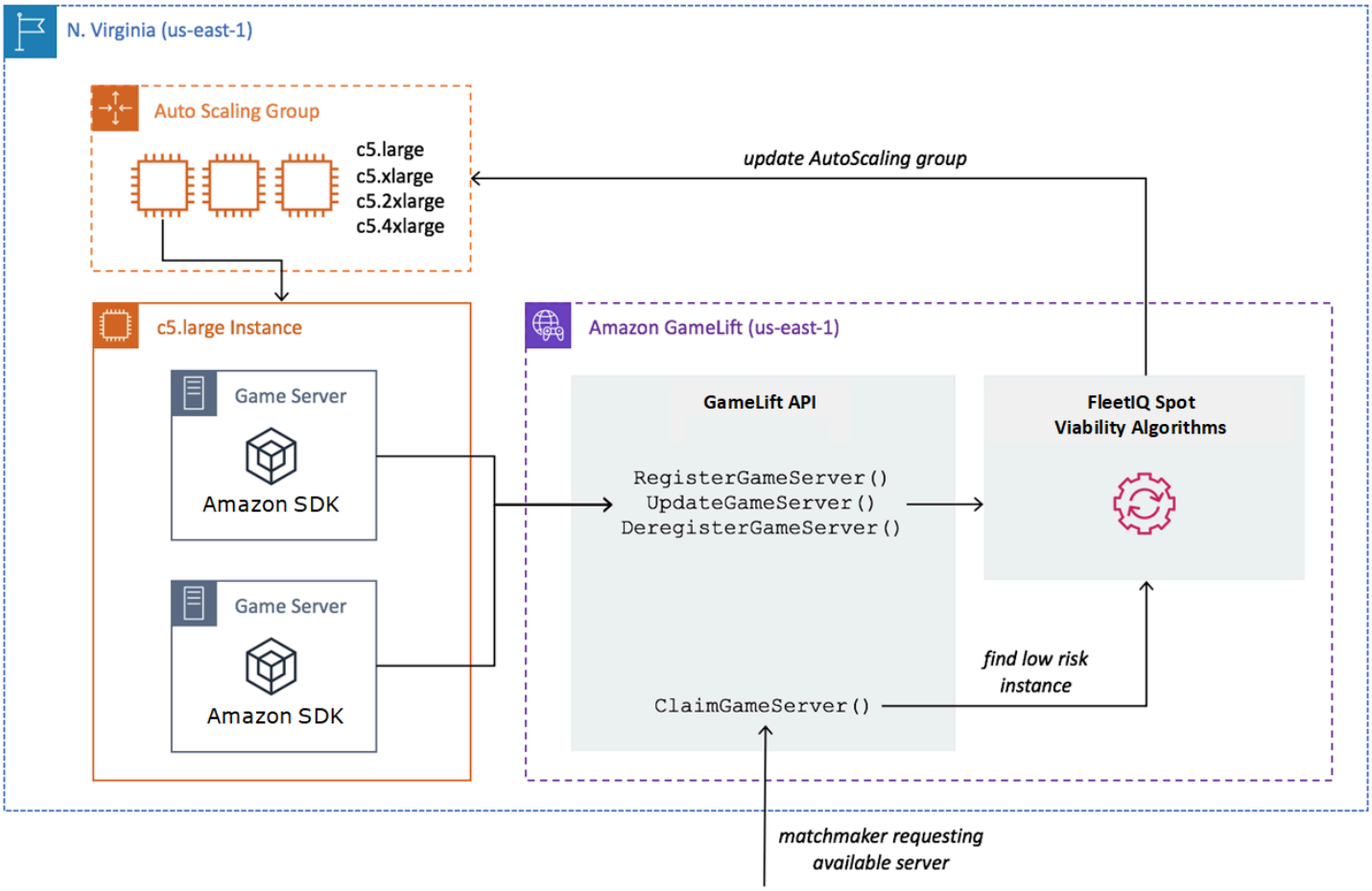
When using Amazon GameLift FleetIQ, you prepare to launch Amazon EC2 instances as usual: make an Amazon Machine Image (AMI) with your game server software, create an Amazon EC2 launch template, and define configuration settings for an Auto Scaling group. However, instead of creating an Auto Scaling group directly, you create a Amazon GameLift FleetIQ game server group with your Amazon EC2 and Auto Scaling resources and configuration. This action prompts Amazon GameLift 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 Amazon services. You can connect directly to instances in the group. As part of its optimization logic, Amazon GameLift 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 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.

Amazon GameLift FleetIQ logic

The following diagram illustrates the role of Amazon GameLift 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 FleetIQ defines the *best* resources as those that deliver the highest game hosting viability for the lowest cost. Amazon GameLift 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 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 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.** A Amazon GameLift 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 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 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 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 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 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 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 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 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 FleetIQ. Amazon GameLift 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 FleetIQ uses the following decision-making process to optimize placement with viable low-cost Spot Instances:

1. Where possible, Amazon GameLift 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 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 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 Amazon account before you set up your game hosting resources with Amazon GameLift 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 for Linux Instances. 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.
- **Amazon IAM role.** An IAM role defines a set of permissions that allow limited access to Amazon resources. A trusted entity, such as another Amazon service, can assume the role and inherit its permissions. When using Amazon GameLift FleetIQ, you must provide an IAM role with a

managed policy that allows Amazon GameLift FleetIQ to create and access Auto Scaling groups and EC2 instance resources in your Amazon account. IAM roles are not Region-specific.

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

- **GameLift game server group**. A game server group contains configuration settings that define how Amazon GameLift FleetIQ works with a corresponding Auto Scaling group to deliver low-cost game hosting. Game server groups are Region-specfic. When you create a game server group in a Region, a new Auto Scaling group is automatically created in your Amazon 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 FleetIQ.
- **GameLift game server.** A game server resource represents a game execution that is running on an instance associated with a Amazon GameLift FleetIQ game server group. This resource is created when a game server registers with Amazon GameLift FleetIQ and identifies the game server group it belongs to. Amazon GameLift 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 FleetIQ resources. They are created in your Amazon 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 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 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 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.

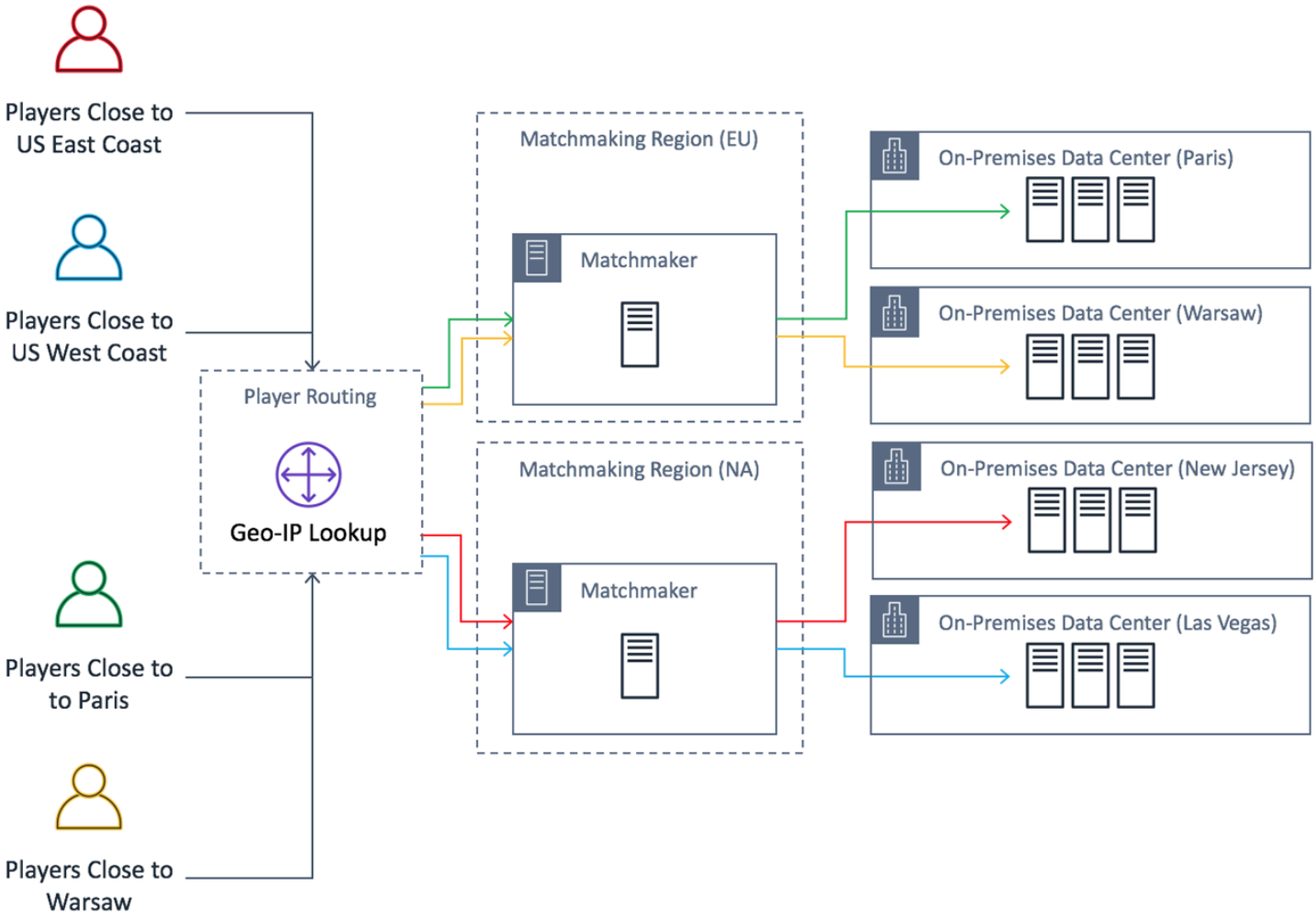
Game architecture with Amazon GameLift FleetIQ

Supplementing on-premises hosting

Amazon GameLift 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 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.

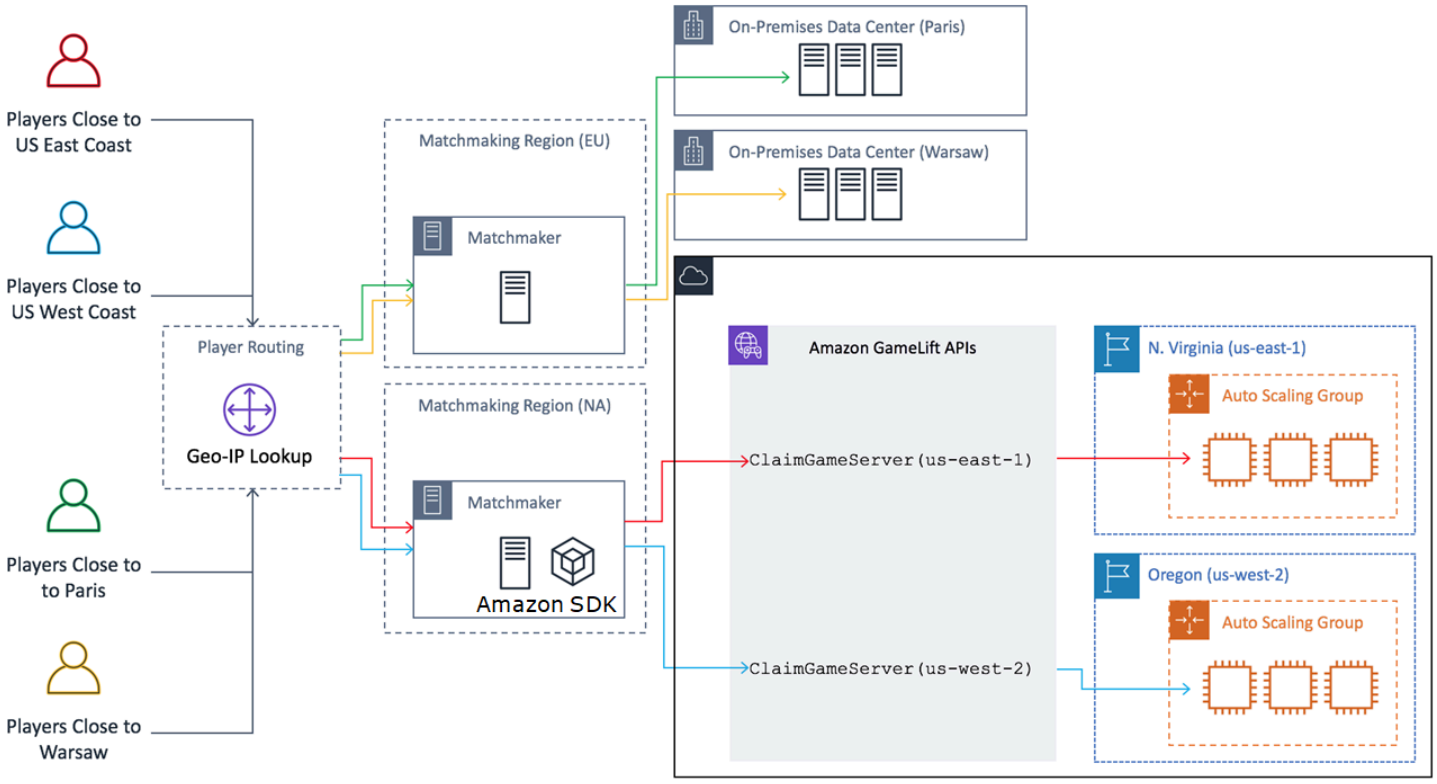


The game developer wants to replace their North America game servers with servers provided by Amazon GameLift FleetIQ. To start, they make minor updates to their game server to enable it for use with Amazon GameLift FleetIQ and then create an Amazon Machine Image (AMI). This image will be installed on 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 FleetIQ game server groups, one for each Amazon 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 FleetIQ which types to use when checking for Spot Instances that are viable for game hosting.

Finally, the developer integrates the Amazon SDK with Amazon GameLift FleetIQ into their North American matchmaker, which calls Amazon GameLift FleetIQ when a new group of players needs server capacity for a game session. Amazon GameLift 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 FleetIQ to find another available game server. Each new game request triggers Amazon GameLift FleetIQ to search for and select game servers with a low chance of interruptions. As a result, Amazon GameLift FleetIQ is constantly redirecting players away from game servers that are not viable for game hosting, even as Spot Instance availability fluctuates over time.



Amazon GameLift FleetIQ life cycles and states

Spot balancing process

Amazon GameLift 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:

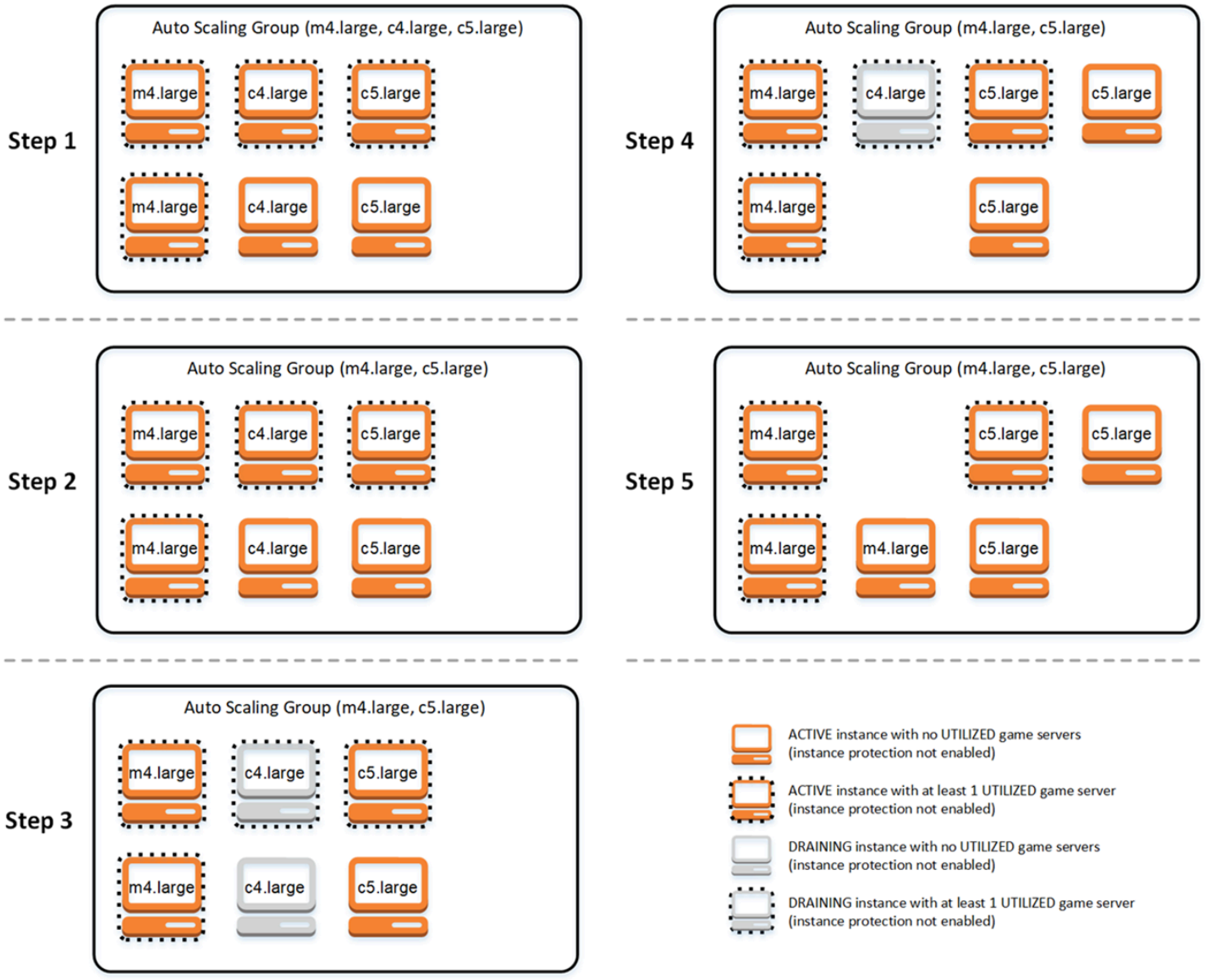
- To constantly refresh the group by only using Spot Instance types that are viable for game hosting.

- To use multiple viable instance types (where possible) in order to reduce the impact of unexpected game server interruptions.

Amazon GameLift 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.

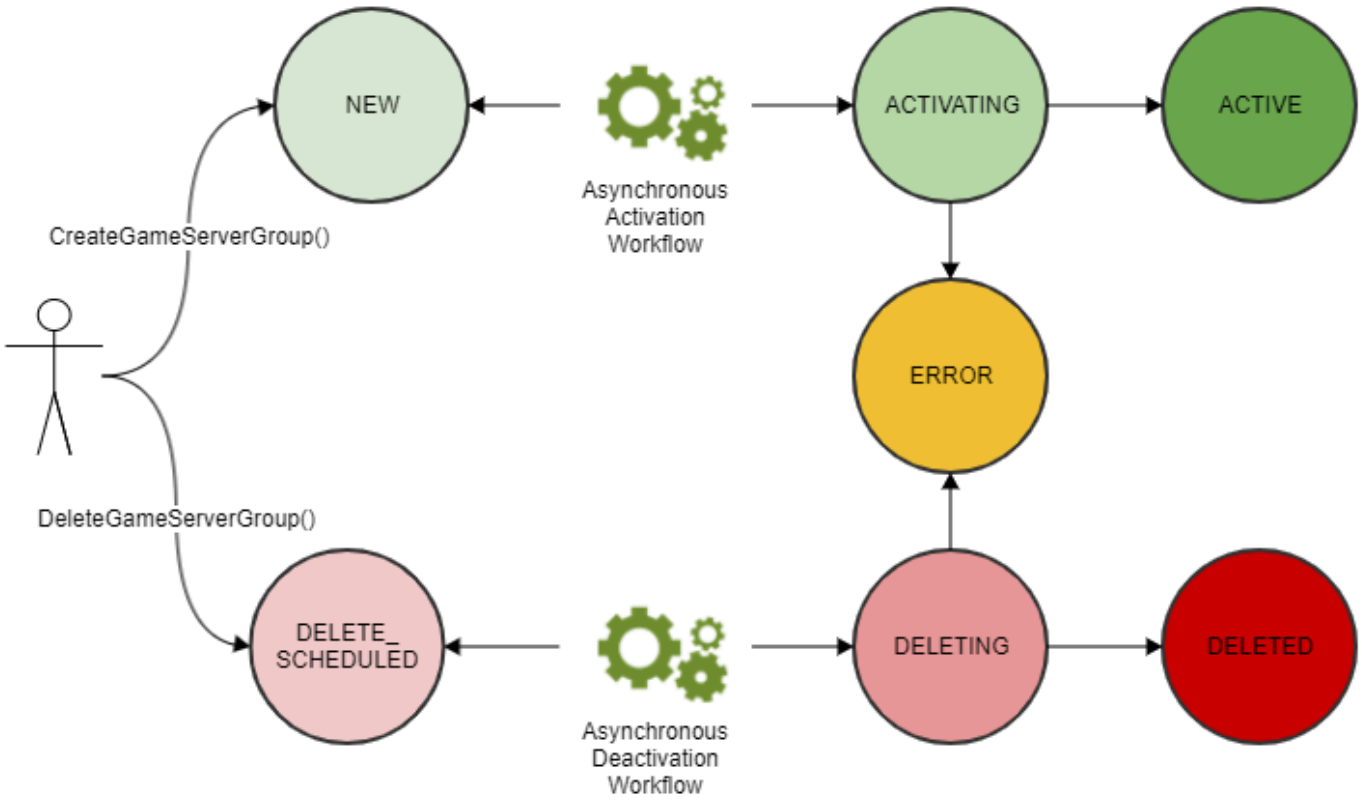


- **Step 1.** Through a game server group, the linked Auto Scaling group is set up to launch instances of types m4.large, c4.large, and c5.large with game server protection enabled. The Auto Scaling group has launched a balanced collection consisting of two Spot Instances of each type. Four instances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

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.



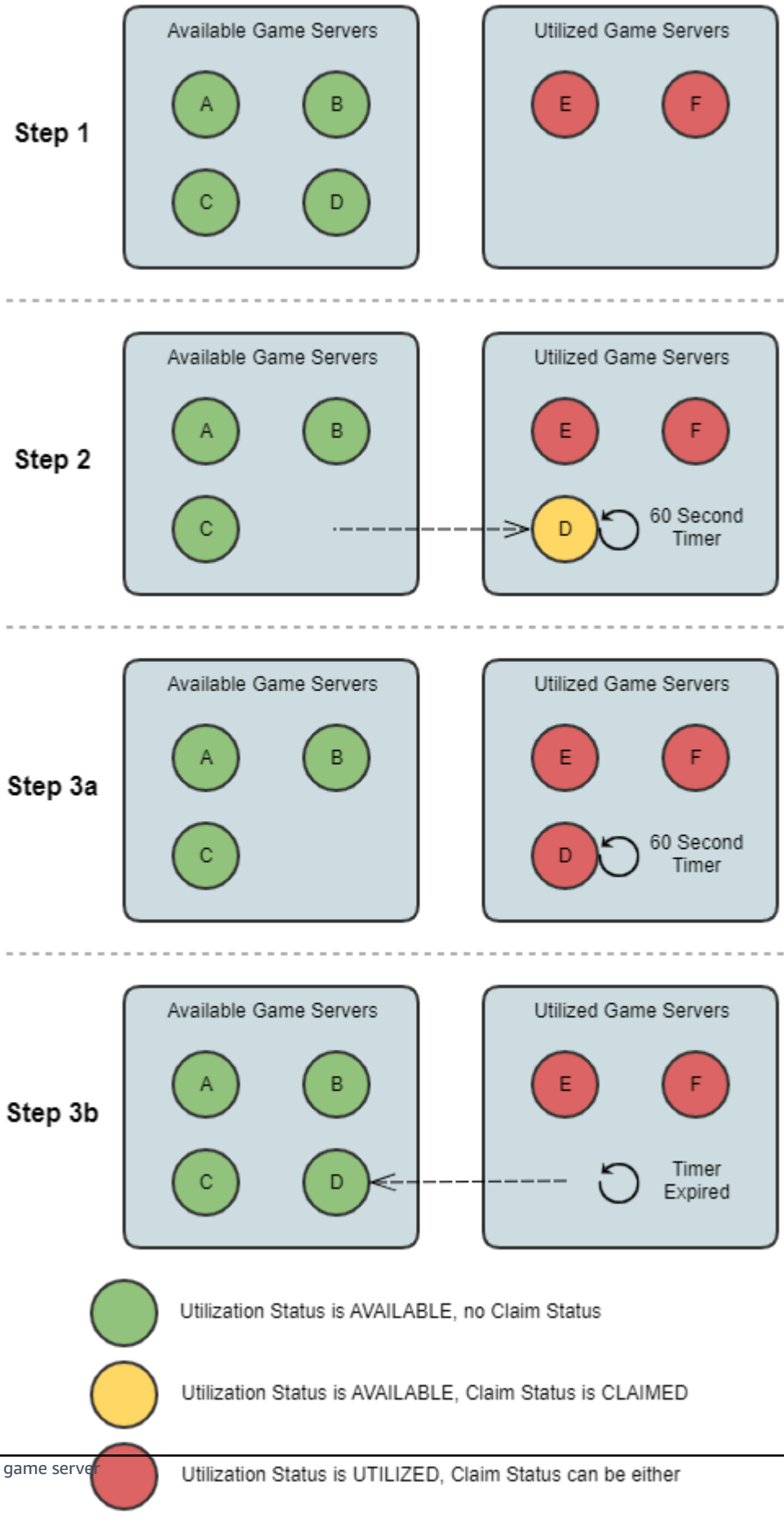
- You create a game server group by calling the Amazon GameLift API `CreateGameServerGroup()` and passing in an EC2 launch template and configuration settings. In response to the call, a new game server group is created and placed in status NEW.
- Amazon GameLift FleetIQ activates an asynchronous activation 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 FleetIQ. The group's instance types are periodically evaluated for game hosting viability and balanced as needed. Amazon GameLift 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 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 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 FleetIQ-linked Auto Scaling group, calls the Amazon GameLift API `RegisterGameServer()` to notify Amazon GameLift 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 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.


Step 1
Available Game Servers
A
B
C
D
Utilized Game Servers
E
F
Step 2
Available Game Servers
A
B
C
Utilized Game Servers
E
F
D
60 Second Timer
Step 3a
Available Game Servers
A
B
C
Utilized Game Servers
E
F
D
60 Second Timer
Step 3b
Available Game Servers
A
B
C
D
Utilized Game Servers
E
F
Timer Expired
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 API `ClaimGameServer()` to request a new game server. This request prompts Amazon GameLift FleetIQ to search for an available game server (D) and set its claim status to CLAIMED for 60 seconds. Amazon GameLift 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 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 API `DeregisterGameServer()` to notify Amazon GameLift FleetIQ of its departure from the game server group's pool of game servers.

Amazon GameLift FleetIQ best practices

Amazon GameLift FleetIQ is a low-level logic layer that helps you manage Amazon EC2 resources for game hosting. In particular, Amazon GameLift 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 FleetIQ as a standalone feature does not provide advanced features that are offered with the fullly managed Amazon GameLift 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 solutions.

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

- **Use Amazon GameLift FleetIQ for session-based games.** Amazon GameLift 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 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 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 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 FleetIQ.** When placing groups of players with game servers, use the Amazon GameLift API `ClaimGameServer()`. Amazon GameLift FleetIQ avoids placing players on instances with a higher chance of game session interruptions.
- **Report game server status to Amazon GameLift FleetIQ.** Periodically report server health and utilization status with the Amazon GameLift API `UpdateGameServer()`. Maintaining accurate game server status helps Amazon GameLift FleetIQ place gameplay more efficiently. It also helps avoid terminating instances with active gameplay during Spot balancing activity.
- **Set up an autoscaling policy.** You can create a target-tracking scaling policy that maintains your hosting capacity based on player utilization and anticipated demand. The Amazon GameLift 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 Amazon 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 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 swtich a game server group from a Spot to an On-Demand strategy as needed.

Also review these Amazon best practices:

- Best Practices for Amazon EC2
- Best Practices for Amazon EC2 Auto Scaling

Amazon GameLift FleetIQ features

- **Optimized Spot balancing.** Amazon GameLift 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 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 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 Amazon Web Services account. This means that you can set up instance launch templates, maintain EC2 Auto Scaling group configurations, and integrate with other Amazon services. As part of its Spot balancing activity, Amazon GameLift FleetIQ makes periodic updates to some EC2 Auto Scaling group properties. You can temporarily override these settings or suspend Amazon GameLift FleetIQ activity as needed.
- **Support for multiple game server executable formats.** Amazon GameLift 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 FleetIQ, you have access to a large range of instance types for game server hosting. (Availability varies by Amazon 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 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 endpoints and quotas in the *Amazon Web Services General Reference*.

Pricing for Amazon GameLift FleetIQ

Amazon GameLift 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, see Amazon GameLift Pricing.

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

Amazon GameLift FleetIQ setting up

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

Topics

Amazon GameLift FleetIQ supported software

Amazon GameLift 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 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 FleetIQ supports integration with Kubernetes, Amazon Elastic Container Service (Amazon ECS), and Amazon Elastic Kubernetes Service (EKS). See more information at Containers on Amazon.

Game development environments

Game clients and servers require some integration to communicate with the Amazon GameLift FleetIQ service. Games make API calls to the Amazon SDK. Download the Amazon SDK or view the Amazon GameLift API reference documentation.

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

- C++ (SDK docs) (Amazon GameLift)

- Java (SDK docs) (Amazon GameLift)
- .NET (SDK docs) (Amazon GameLift)
- Go (SDK docs) (Amazon GameLift)
- Python (SDK docs) (Amazon GameLift)
- Ruby (SDK docs) (Amazon GameLift)
- PHP (SDK docs) (Amazon GameLift)
- JavaScript/Node.js (SDK docs) (Amazon GameLift)

Set up your Amazon account for Amazon GameLift FleetIQ

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

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

Create an Amazon Web Services account

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

Topics

- Sign up for an Amazon Web Services account
- Secure IAM users

Sign up for an Amazon Web Services account

If you do not have an Amazon Web Services account, use the following procedure to create one.

To sign up for Amazon Web Services

1. Open http://www.amazonaws.cn/ and choose **Sign Up**.
2. Follow the on-screen instructions.

Amazon 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 http://www.amazonaws.cn/ and choosing **My Account**.

Secure IAM users

After you sign up for an Amazon Web Services account, safeguard your administrative user by turning on multi-factor authentication (MFA). For instructions, see Enable a virtual MFA device for an IAM user (console) in the *IAM User Guide*.

To give other users access to your Amazon Web Services account resources, create IAM users. To secure your IAM users, turn on MFA and only give the IAM users the permissions needed to perform their tasks.

For more information about creating and securing IAM users, see the following topics in the *IAM User Guide*:

- Creating an IAM user in your Amazon Web Services account
- Access management for Amazon resources
- Example IAM identity-based policies

Manage user permissions for Amazon GameLift FleetIQ

Create additional users or extend Amazon GameLift FleetIQ access permissions to existing users as needed. Users who work with Amazon GameLift 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 Amazon 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 Amazon CloudFormation

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

- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Creating 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 Creating 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*.

Amazon GameLift FleetIQ_policy

```
{
  "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 Amazon CloudFormation

If you use Amazon CloudFormationto manage your game hosting resources, add the Amazon 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 Amazon outside of the Amazon Web Services Management Console. The Amazon APIs and the Amazon Command Line Interface require access keys. Whenever possible, create temporary credentials that consist of an access key ID, a secret access key, and a security token that indicates when the credentials expire.

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

Which user needs programmatic access?	To	By
IAM	Use short-term credentials to sign programmatic requests to the Amazon CLI or Amazon APIs (directly or by using the Amazon SDKs).	Following the instructions in Using temporary credentials with Amazon resources in the *IAM User Guide*.
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the Amazon CLI or Amazon	Following the instructions in Managing access keys for IAM users in the *IAM User Guide*.

Which user needs programmatic access?	To	By
	APIs (directly or by using the Amazon SDKs).	

If you use access keys, see Best practices for managing Amazon access keys.

Create IAM roles for cross-service interaction

In order for Amazon GameLift 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 Amazon account and assigning a set of limited permissions. Each role also sspecifies which services can assume the role.

Set up the following roles:

- Create a role for Amazon GameLift FleetIQ to update your Amazon EC2 resources.
- Create a role for Amazon EC2 resources to communicate with Amazon GameLift FleetIQ.

Create a role for Amazon GameLift FleetIQ

This role allows Amazon GameLift 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 Amazon CLI to create a role for Amazon GameLift 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 Amazon Service and Amazon Managed Policies.

Console

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

1. Open the IAM console and choose **Roles: Create role**.
2. For **Select type of trusted entity**, choose **Amazon service**.

3. For **Choose a use case**, choose **GameLift** from the list of services. Under **Select your use case**, the appropriate Amazon GameLift 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 "Amazon 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 FleetIQ game server groups.

Amazon CLI

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

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

```
{
  "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:

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

   Linux:

```
Amazon 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 FleetIQ game server groups.

3. Use iam attach-role-policy to attach the managed permissions policy "GameLiftGameServerGroupPolicy".

```
Amazon 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 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 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 FleetIQ game server group.

Use the Amazon 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 Amazon Service.

Amazon CLI

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

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

```
{
```

```
  "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:

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

   Linux:

   ```
   Amazon 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.

   ```
   {
       "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:

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

   Linux:

   ```
   Amazon 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.

   ```
   Amazon 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.

   ```
    Amazon 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 FleetIQ

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

- Adapt your game server to communicate with Amazon GameLift 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 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 FleetIQ. If you're looking for the Amazon GameLift managed hosting service, which automates more game hosting processes for you, see the Amazon GameLift Developer Guide.

To get started using Amazon GameLift 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 Amazon account and set up users with Amazon GameLift FleetIQ access.**

   Create a new Amazon Web Services account or choose an existing account to use with Amazon GameLift FleetIQ. Set up users with permissions to manage the Amazon EC2, Auto Scaling, and other Amazon resources used with your game. For detailed instructions, see Set up your Amazon account for Amazon GameLift FleetIQ.
2. **Create IAM roles.**

Create roles that allow Amazon GameLift 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 Amazon SDK and Amazon CLI with Amazon GameLift FleetIQ functionality.**
   - Download the latest version of the Amazon SDK.
   - View the Amazon GameLift API reference documentation.
4. **Prepare your game server for use with Amazon GameLift FleetIQ.**

   Add the Amazon SDK to your game server project and add code to keep Amazon GameLift 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 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 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 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 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 Amazon account that Amazon GameLift FleetIQ can modify. For additional guidance and examples, see Manage Amazon GameLift FleetIQ game server groups.

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

   Add the Amazon 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 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 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 FleetIQ game server groups

This topic describes the tasks required to set up a Amazon GameLift 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 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 Amazon account to allow GameLift 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 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 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 capcity 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.

```
Amazon 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 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 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.

```
Amazon 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 console, under **Game server groups**, to monitor the status of your game server groups.

Integrate Amazon GameLift FleetIQ into a game server

This topic describes the tasks required to prepare your game server project to communicate with Amazon GameLift FleetIQ. Refer to Amazon GameLift 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 FleetIQ by calling RegisterGameServer(). Registering allows Amazon GameLift 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 FleetIQ with relevant game server data and connection information, including the port and IP address that it uses for inbound client connections.

```
Amazon 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 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.

```
Amazon 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 FleetIQ.

Utilization status

Reporting game server utilization status keeps Amazon GameLift 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 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 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 withing this window. If utilization status is not updated, Amazon GameLift FleetIQ removes the claim, assumes that the game server is available, and may reserve the game server for another client claim request.

```
Amazon 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 FleetIQ using DeregisterGameServer().

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

Integrate Amazon GameLift FleetIQ into a game client

This topic describes the tasks required to prepare your game client or matchmaking service to communicate with Amazon GameLift 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 FleetIQ choose an available game server. This option takes advantage of Amazon GameLift 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").

Let Amazon GameLift FleetIQ choose a game server

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

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

In response to a claim request, Amazon GameLift 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 FleetIQ after players

connect or gameplay starts. This ensures that Amazon GameLift FleetIQ does not provide this game server in response to subsequent requests for game server capacity. Update game server status by calling UpdateGameServer().

```
Amazon 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.

```
Amazon gamelift list-game-servers \
    --game-server-group-name MyLiveGroup \
    --limit 20 \
    --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 FleetIQ does not exercise any of its instance type optimization logic, as described in Amazon GameLift FleetIQ logic.

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

Monitor Amazon GameLift FleetIQ with Amazon CloudWatch

Use Amazon CloudWatch metrics to scale your instance capacity, build operations dashboards, and trigger alarming. Amazon GameLift FleetIQ as a standalone solution emits a set of Amazon CloudWatch metrics to your Amazon 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 at Amazon GameLift metrics.

Metric	Description
`AvailableGameServers`	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. Units: Count Relevant CloudWatch statistics: Sum Dimensions: GameServerGroup
`UtilizedGameServers`	Game servers that are currently occupied with gameplay. This number includes game servers that are in UTILIZED status. Units: Count Relevant CloudWatch statistics: Sum Dimensions: GameServerGroup
`DrainingAvailableGameServers`	Game servers on instances scheduled for terminati on that are currently not supporting gameplay. These game servers are the lowest priority to be claimed in response to a new claim request.

Metric	Description
	Units: Count Relevant CloudWatch statistics: Sum Dimensions: GameServerGroup
DrainingUtilizedGameServers	Game servers on instances scheduled for terminati on that are currently supporting gameplay. Units: Count Relevant CloudWatch statistics: Sum Dimensions: GameServerGroup
PercentUtilizedGameServers	Portion of game servers that are currently supportin g 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. Units: Percent Relevant CloudWatch statistics: Average, Minimum, Maximum Dimensions: GameServerGroup
GameServerInterruptions	Game servers on Spot Instances that were interrupt ed due to limited Spot availability. Units: Count Relevant CloudWatch statistics: Sum Dimensions: GameServerGroup, InstanceType

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

Amazon GameLift FleetIQ reference guides

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

Topics

Amazon GameLift FleetIQ service API reference (Amazon SDK)

This topic provides a task-based list of API actions for Amazon GameLift FleetIQ. The Amazon GameLift FleetIQ service API is packaged into the Amazon SDK in the `aws.gamelift` namespace. Download the Amazon SDK or view the Amazon GameLift API reference documentation.

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

Amazon GameLift FleetIQ API actions

The following operations allow you to manage your Amazon GameLift 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 and 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 a Amazon GameLift 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 – Reinstate 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 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 FleetIQ to remove the game server from the game server group. CLI command: deregister-game-server

Available programming languages

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

- C++ (SDK docs) (Amazon GameLift)
- Java (SDK docs) (Amazon GameLift)
- .NET (SDK docs) (Amazon GameLift)

- Go (SDK docs) (Amazon GameLift)
- Python (SDK docs) (Amazon GameLift)
- Ruby (SDK docs) (Amazon GameLift)
- PHP (SDK docs) (Amazon GameLift)
- JavaScript/Node.js (SDK docs) (Amazon GameLift)

Security with Amazon GameLift FleetIQ

If you're using Amazon GameLift FleetIQ as a standalone feature with Amazon EC2, see Security in Amazon EC2 in the *Amazon EC2 User Guide for Linux Instances*.

Cloud security at Amazon is the highest priority. As an Amazon 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 Amazon and you. For information on how to apply the shared responsibility model when using Amazon GameLift FleetIQ, see Security in Amazon GameLift.

Amazon GameLift FleetIQ release notes and SDK versions

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

Amazon GameLift developer resources

To view all Amazon GameLift documentation and developer resources, see the Amazon GameLift Documentation home page.

Amazon Glossary

For the latest Amazon terminology, see the Amazon glossary in the *Amazon Web Services Glossary Reference*.