

# Fleets workloads

Last updated 2025-09-29

IBM Cloud® Code Engine is a fully managed, serverless platform that runs your containerized workloads, including fleet workloads. Learn about running fleets in Code Engine.

## What are fleets?

A fleet, also called a **serverless fleet**, is a Code Engine compute component that runs one or more instances of user code in parallel to complete a large queue of compute-intensive tasks. Fleets can connect to Virtual Private Clouds to securely interoperate with user data and services. Unlike batch jobs, fleets provide dynamic task queuing and single-tenant isolation, and are compatible with both vCPU and GPUs.

## How do fleets work?

Fleets have three principal elements: tasks, instances, and worker nodes. A single task is completed by a single instance of user code, which runs on a worker node. Each worker node can run several tasks concurrently, allowing many tasks to be completed in parallel. When a task is complete, a new instance spins up on the worker node to complete the next available task in the queue. This process repeats on each worker node until all tasks are completed, at which point worker nodes are automatically deprovisioned.

## How are fleets different from jobs?

Unlike batch jobs, which are primarily intended for small to medium sized tasks, fleets are designed to complete large, compute-intensive tasks. Review the table below for specific differences between jobs and fleets.

Characteristic	Fleets	Jobs
Isolation	Single-tenant	Multi-tenant
Task size	Large tasks	Small-to-medium sized tasks
Implementation	Dynamic queue	Static array
Machine configuration	Full control over machine profile	No control over machine profile
VPC connection	Natively connects to existing VPC	

Table 1. Comparing Code Engine Fleets and jobs

## What are the key features of working with Code Engine fleets?


Review the following topics to learn more about working with fleets in Code Engine.

### Automatic scaling

With a fleet, you can take advantage of large scale parallelism to complete large, compute-intensive tasks. Code Engine automatically scales the worker nodes in your fleet to meet resource requirements most efficiently. You can let Code Engine determine the profiles of the workers that are deployed, or you can choose a specific profile, such as a GPU type. In either case, Code Engine automatically scales the number of workers for the greatest level of efficiency.

The number of workers deployed is based on the number of tasks to complete, the resources required for an instance of your code, and the maximum number of concurrent instances you want to run at a time. By adjusting these settings, you can scale your fleet to complete as few as 1 or as many as several million tasks.

For example, imagine you have 4 tasks to complete. Each instance of code to complete a task requires `1 vCPU and 2 GB memory`, for a total requirement of `4 vCPU and 8GB memory`. If you set the maximum number of concurrent instances to 2, Code Engine can deploy 1 worker node with `2 vCPU and 4 GB memory` to run 2 instances at one time. If you set the maximum number of concurrent instances to 4, Code Engine can deploy 2 of the same worker nodes to run all 4 instances at the same time across both workers. In the second scenario, the fleet finishes more quickly but requires more workers to be deployed. Keep in mind that there is an infinite number of combinations for automatic worker node deployment, and that worker nodes of various profiles might be deployed.

 **Note:** Automatic scaling is not available for GPUs. To deploy GPUs for a fleet, you must specify the type of GPU worker to deploy.

### Task and instance specification

When deploying a fleet, users can specify the number of tasks to complete, the number of instances to run at a time, the order in which tasks are completed. Fleets can work on many tasks in parallel by starting multiple instances concurrently. All instances are created with the same amount of vCPU and memory as per the fleet's specification. Additionally users can create a task specification file to provide specific commands and arguments for each task or to create custom task indexes.

### Task storage in COS

You can use a COS bucket to provide the tasks for your fleet by specifying a persistent data store and bucket subpath when you create the fleet. Each object in the COS bucket represents a single task.

### Isolation

Unlike batch jobs, fleets provide single-tenant isolation. Your fleet workers are not shared with other users.

### Retries

Each fleet instance runs to completion. However, in the event of an error, Code Engine restarts the instance. You can limit the maximum number of retries to avoid restarting failed instances.

### Status

A fleet is in the `Running` status when at least one worker node is provisioned. When the fleet has is no longer running, it is in the `Succeeded` status if all tasks completed successfully, or `Failed` if one or more tasks failed. You can also cancel a fleet.

For more information, see [Understanding the status of your fleet](#).

## How can I get started with fleets?

To create and run a simple Code Engine fleet with the `icr.io/codeengine/helloworld` sample image, see [Running your first Code Engine fleet](#).

### Help improve the docs

Something not quite right? Contribute in GitHub

Open doc issue

Edit topic

[Privacy statement](#)

On this page

#### What are fleets?

How do fleets work?

How are fleets different from jobs?

What are the key features of working with Code Engine fleets?

How can I get started with fleets?