## What is container orchestration?

Container orchestration is the process of automating the networking and management of [containers](https://aws.amazon.com/containers/) so you can deploy applications at scale. Containerization bundles an application’s code with all the files and libraries it needs to run on any infrastructure. Microservices architectures can have hundreds, or even thousands, of containers as applications grow and become more complex. Container orchestration tools aim to simplify container infrastructure management by automating their complete lifecycle—from provisioning and scheduling to deployment and deletion. Organizations can benefit from containerization at scale without incurring additional maintenance overheads.

Why is container orchestration necessary?

Containers have become the standard unit of computing for cloud-native applications. Cloud providers offer virtual server instances for running all sorts of computing workloads, and are a perfect fit for container-based workloads. The only requirement to be able to run containers is that the server itself runs a containerization service like Docker. [Docker](https://aws.amazon.com/docker/) is an open source tool for packaging software and associated libraries, system tools, code, and runtime into a container. It is a lightweight solution for running and managing a few containers on a single server instance, but scaling becomes a challenge.

Before managed container orchestration platforms existed, organizations used complex scripting to manage container deployment, scheduling, and deletion across multiple machines. Maintaining these scripts created challenges, like version control, and setup was difficult to scale. Container orchestration automates and resolves these complexities, removing challenges associated with manual management.

Container orchestration use cases

Container orchestration tools become necessary when you have to:

* Manage and scale containers across a number of instances.
* Run many different containerized applications.
* Run different versions of applications (for example, test and production across CI/CD) at once.
* Ensure app service continuity in case of a server failure by running multiple duplicate instances (replicas) of a container.
* Run multiple instances of an app across multiple different geographical regions.
* Maximize usage of multiple server instances for budgeting purposes.
* Run large containerized applications comprised of thousands of different microservices.

What are the benefits of container orchestration?

Managing complex container architectures without a container orchestration solution can be difficult. Container orchestration manages container creation, configuration, scheduling, deployment, and deletion. It also supports:

* Application load balancing and traffic management.
* App service continuity across containers.
* Security across containerization.
* Container status monitoring.
* Resourcing containers from underlying server or instance resources.

The following are more benefits of container orchestrations.

Built-in resilience

Simple containerization services typically will not restart a container if it goes offline. Similarly, if the machine that a container is running on goes down, the container won’t be restarted when the machine restarts. Container orchestration solutions can ensure that containers are automatically restarted or that more than one version is running at all times in case of machine failure.

Enhanced performance

One of the biggest benefits of container orchestration is that it automates the scalability, availability, and performance of containerized apps. You can configure container orchestration tools to scale based on demand, network availability, and infrastructure restrictions. The container orchestration solution can monitor performance across the container network and automatically reconfigure containers for optimal performance.

Resource optimization

Underlying servers and instances cost money to run and must be used efficiently for cost optimization. Container orchestration allows organizations to maximize the usage of each available instance, as well as instantiate on-demand instances if resources run out. This leads to cost savings in infrastructure.

How does container orchestration work?

Containers are self-contained Linux-based applications or microservices bundled with all the libraries and functions they need to run on almost any type of machine. Container orchestration works by managing containers across a group of server instances (also called *nodes*). A group of nodes that runs interconnected containers is called a *cluster*.

Container orchestration requires, first, an underlying containerization solution running on every node in the cluster—typically, this will be Docker. The nodes must also run the orchestration tool. A designated master node, with a control plane, is the controller of the orchestration solution itself. The administrator of the solution uses a GUI or command-line controller on the master node to manage and monitor the container orchestration tool.

Creation and scheduling

The container orchestration solution reads a declarative configuration file, written in YAML or JSON, to learn the specific required state of the system. Using the information specified in the file, the tool:

1. Obtains container images from a container registry.
2. Provisions the containers with their individual requirements.
3. Determines the networking required between the containers.

The tool then schedules and deploys the multi-container application across the cluster. This best fit between nodes and containers is determined by the container orchestration tool, rather than specified in the configuration file. The tool selects the actual node to run each container based on the node’s resource constraints, such as CPU, memory, and so on, as well as the defined container requirements.

Management

Once the containers are running across the cluster, the orchestration tool manages overall system health to ensure it remains in the specified performance state. This may include:

* Resource allocation across containers.
* Deploying containers to new nodes, or deleting containers.
* Load balancing of traffic to the application.

A container orchestration solution manages the lifecycle of containers to optimize and secure large, complex multi-container workloads and environments. It can manage as many containerized applications as an organization requires. Running multiple master nodes for high availability and fault tolerance is typical under higher organizational demands.