Swarm mode key concepts

Estimated reading time: 4 minutes

This topic introduces some of the concepts unique to the cluster management and orchestration features of Docker Engine 1.12.

What is a swarm?

The cluster management and orchestration features embedded in the Docker Engine are built using swarmkit (https://github.com/docker/swarmkit/). Swarmkit is a separate project which implements Docker's orchestration layer and is used directly within Docker.

A swarm consists of multiple Docker hosts which run in **swarm mode** and act as managers (to manage membership and delegation) and workers (which run swarm services). A given Docker host can be a manager, a worker, or perform both roles. When you create a service, you define its optimal state (number of replicas, network and storage resources available to it, ports the service exposes to the outside world, and more). Docker works to maintain that desired state. For instance, if a worker node becomes unavailable, Docker schedules that node's tasks on other nodes. A *task* is a running container which is part of a swarm service and managed by a swarm manager, as opposed to a standalone container.

One of the key advantages of swarm services over standalone containers is that you can modify a service's configuration, including the networks and volumes it is connected to, without the need to manually restart the service. Docker will update the configuration, stop the service tasks with the out of date configuration, and create new ones matching the desired configuration.

When Docker is running in swarm mode, you can still run standalone containers on any of the Docker hosts participating in the swarm, as well as swarm services. A key difference between standalone containers and swarm services is that only swarm managers can manage a swarm, while standalone containers can be started on any daemon. Docker daemons can participate in a swarm as managers, workers, or both.

In the same way that you can use Docker Compose (/compose/) to define and run containers, you can define and run Swarm service (/engine/swarm/services/) stacks.

Keep reading for details about concepts relating to Docker swarm services, including nodes, services, tasks, and load balancing.

Nodes

A node is an instance of the Docker engine participating in the swarm. You can also think of this as a Docker node. You can run one or more nodes on a single physical computer or cloud server, but production swarm deployments typically include Docker nodes distributed across multiple physical and cloud machines.

To deploy your application to a swarm, you submit a service definition to a manager node. The manager node dispatches units of work called tasks to worker nodes.

Manager nodes also perform the orchestration and cluster management functions required to maintain the desired state of the swarm. Manager nodes elect a single leader to conduct orchestration tasks.

Worker nodes receive and execute tasks dispatched from manager nodes. By default manager nodes also run services as worker nodes, but you can configure them to run manager tasks exclusively and be manager-only nodes. An agent runs on each worker node and reports on the tasks assigned to it. The worker node notifies the manager node of the current state of its assigned tasks so that the manager can maintain the desired state of each worker.

Services and tasks

A service is the definition of the tasks to execute on the manager or worker nodes. It is the central structure of the swarm system and the primary root of user interaction with the swarm.

When you create a service, you specify which container image to use and which commands to execute inside running containers.

In the replicated services model, the swarm manager distributes a specific number of replica tasks among the nodes based upon the scale you set in the desired state.

For global services, the swarm runs one task for the service on every available node in the cluster.

A task carries a Docker container and the commands to run inside the container. It is the atomic scheduling unit of swarm. Manager nodes assign tasks to worker nodes according to the number of replicas set in the service scale. Once a task is assigned to a node, it cannot move to another node. It can only run on the assigned node or fail.

Load balancing

The swarm manager uses ingress load balancing to expose the services you want to make available externally to the swarm. The swarm manager can automatically assign the service a PublishedPort or you can configure a PublishedPort for the service. You can specify any unused port. If you do not specify a port, the swarm manager assigns the service a port in the 30000-32767 range.

External components, such as cloud load balancers, can access the service on the PublishedPort of any node in the cluster whether or not the node is currently running the task for the service. All nodes in the swarm route ingress connections to a running task instance.

Swarm mode has an internal DNS component that automatically assigns each service in the swarm a DNS entry. The swarm manager uses **internal load balancing** to distribute requests among services within the cluster based upon the DNS name of the service.

What's next?

- Read the Swarm mode overview (/engine/swarm/).
- Get started with the Swarm mode tutorial (/engine/swarm/swarm-tutorial/).

docker (/search/?q=docker), container (/search/?q=container), cluster (/search/?q=cluster), swarm mode (/search/?q=swarm mode)