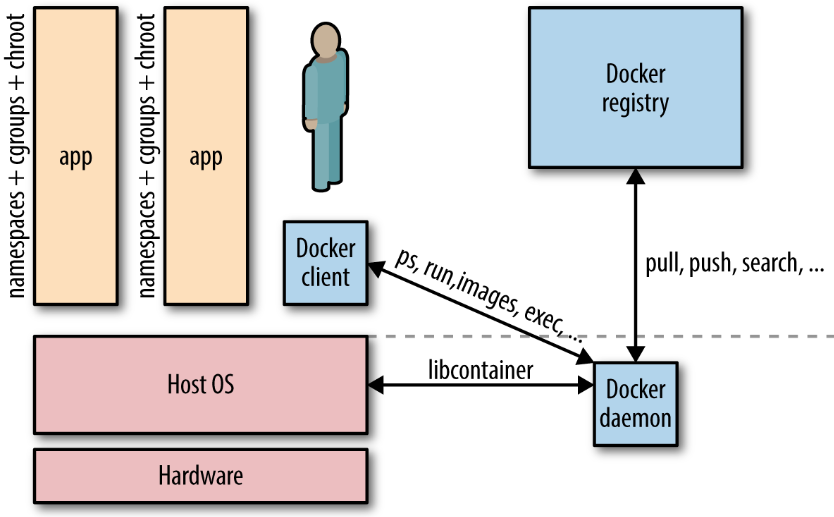
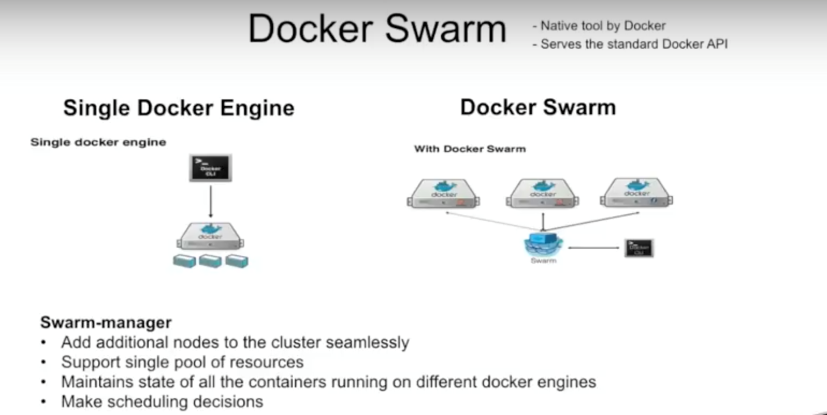
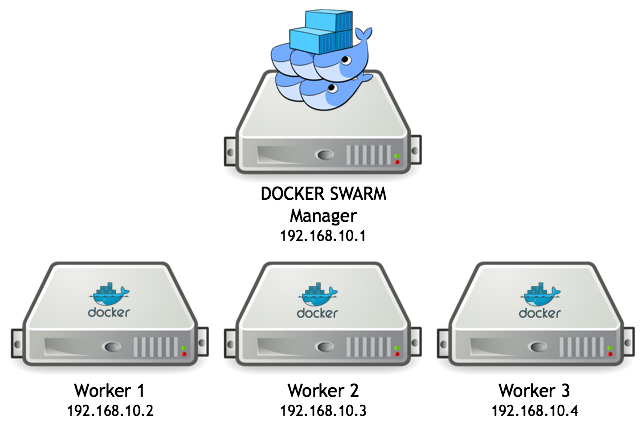
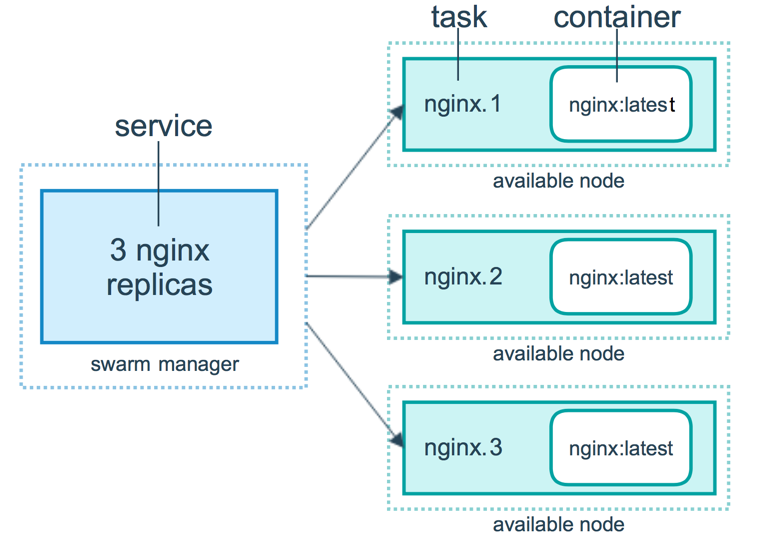
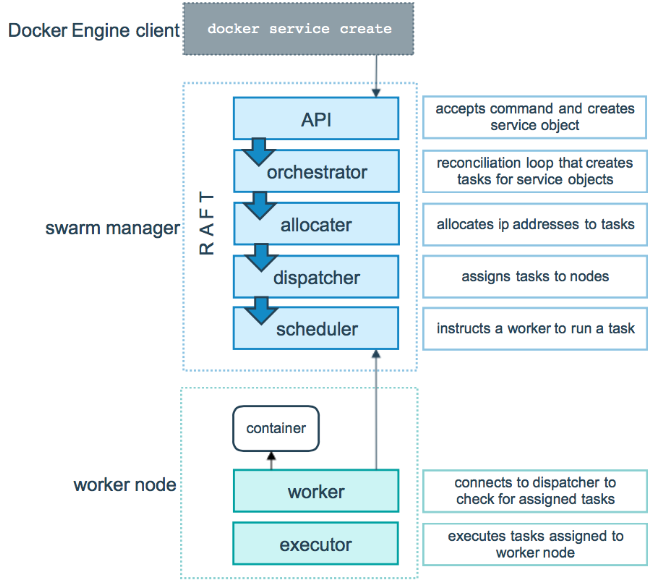
Domain3 – Orchestration.

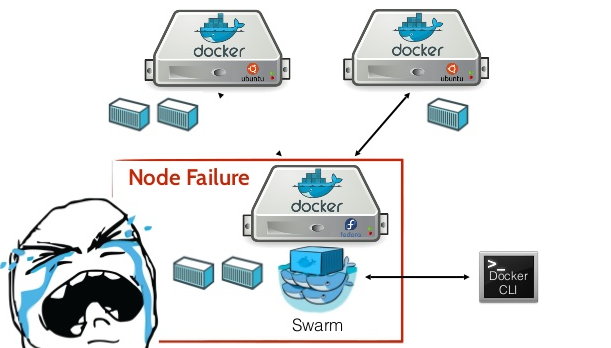


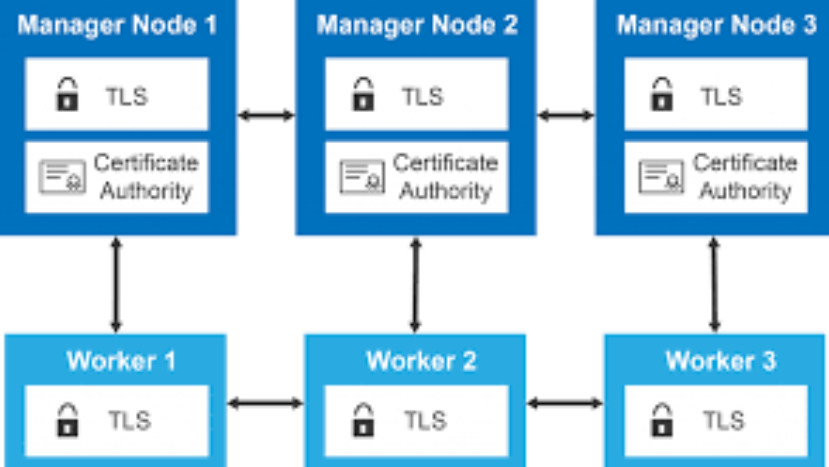












Create a swarm

After you complete the [tutorial setup](https://docs.docker.com/engine/swarm/swarm-tutorial/) steps, you’re ready to create a swarm. Make sure the Docker Engine daemon is started on the host machines.

1. Open a terminal and ssh into the machine where you want to run your manager node. This tutorial uses a machine named manager1. If you use Docker Machine, you can connect to it via SSH using the following command:

$ docker-machine ssh manager1

1. Run the following command to create a new swarm:

docker swarm init --advertise-addr <MANAGER-IP>

**Note**: If you are using Docker for Mac or Docker for Windows to test single-node swarm, simply run docker swarm init with no arguments. There is no need to specify --advertise-addr in this case. To learn more, see the topic on how to [Use Docker for Mac or Docker for Windows](https://docs.docker.com/engine/swarm/swarm-tutorial/#use-docker-for-mac-or-docker-for-windows) with Swarm.

In the tutorial, the following command creates a swarm on the manager1 machine:

$ docker swarm init --advertise-addr 192.168.99.100

Swarm initialized: current node (dxn1zf6l61qsb1josjja83ngz) is now a manager.

To add a worker to this swarm, run the following command:

docker swarm join \

--token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \

192.168.99.100:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

The --advertise-addr flag configures the manager node to publish its address as 192.168.99.100. The other nodes in the swarm must be able to access the manager at the IP address.

The output includes the commands to join new nodes to the swarm. Nodes will join as managers or workers depending on the value for the --token flag.

Run docker info to view the current state of the swarm:

$ docker info

Containers: 2

Running: 0

Paused: 0

Stopped: 2

...snip...

Swarm: active

NodeID: dxn1zf6l61qsb1josjja83ngz

Is Manager: true

Managers: 1

Nodes: 1

...snip...

1. Run the docker node ls command to view information about nodes:

$ docker node ls

ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS

dxn1zf6l61qsb1josjja83ngz \* manager1 Ready Active Leader

The \* next to the node ID indicates that you’re currently connected on this node.

Docker Engine swarm mode automatically names the node for the machine host name. The tutorial covers other columns in later steps.

Add nodes to the swarm

Once you’ve [created a swarm](https://docs.docker.com/engine/swarm/swarm-tutorial/create-swarm/) with a manager node, you’re ready to add worker nodes.

1. Open a terminal and ssh into the machine where you want to run a worker node. This tutorial uses the name worker1.
2. Run the command produced by the docker swarm init output from the [Create a swarm](https://docs.docker.com/engine/swarm/swarm-tutorial/create-swarm/) tutorial step to create a worker node joined to the existing swarm:

$ docker swarm join \

--token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \

192.168.99.100:2377

This node joined a swarm as a worker.

If you don’t have the command available, you can run the following command on a manager node to retrieve the join command for a worker:

$ docker swarm join-token worker

To add a worker to this swarm, run the following command:

docker swarm join \

--token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \

192.168.99.100:2377

1. Open a terminal and ssh into the machine where you want to run a second worker node. This tutorial uses the name worker2.
2. Run the command produced by the docker swarm init output from the [Create a swarm](https://docs.docker.com/engine/swarm/swarm-tutorial/create-swarm/) tutorial step to create a second worker node joined to the existing swarm:

$ docker swarm join \

--token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-8vxv8rssmk743ojnwacrr2e7c \

192.168.99.100:2377

This node joined a swarm as a worker.

1. Open a terminal and ssh into the machine where the manager node runs and run the docker node ls command to see the worker nodes:

ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS

03g1y59jwfg7cf99w4lt0f662 worker2 Ready Active

9j68exjopxe7wfl6yuxml7a7j worker1 Ready Active

dxn1zf6l61qsb1josjja83ngz \* manager1 Ready Active Leader

The MANAGER column identifies the manager nodes in the swarm. The empty status in this column for worker1 and worker2 identifies them as worker nodes.

Swarm management commands like docker node ls only work on manager nodes.

Deploy a service to the swarm

After you [create a swarm](https://docs.docker.com/engine/swarm/swarm-tutorial/create-swarm/), you can deploy a service to the swarm. For this tutorial, you also [added worker nodes](https://docs.docker.com/engine/swarm/swarm-tutorial/add-nodes/), but that is not a requirement to deploy a service.

1. Open a terminal and ssh into the machine where you run your manager node. For example, the tutorial uses a machine named manager1.
2. Run the following command:

$ docker service create --replicas 1 --name helloworld alpine ping docker.com

9uk4639qpg7npwf3fn2aasksr

* + The docker service create command creates the service.
  + The --name flag names the service helloworld.
  + The --replicas flag specifies the desired state of 1 running instance.
  + The arguments alpine ping docker.com define the service as an Alpine Linux container that executes the command ping docker.com.

1. Run docker service ls to see the list of running services:

$ docker service ls

ID NAME SCALE IMAGE COMMAND

9uk4639qpg7n helloworld 1/1 alpine ping docker.com

Inspect a service on the swarm

Estimated reading time: 2 minutes

When you have [deployed a service](https://docs.docker.com/engine/swarm/swarm-tutorial/deploy-service/) to your swarm, you can use the Docker CLI to see details about the service running in the swarm.

1. If you haven’t already, open a terminal and ssh into the machine where you run your manager node. For example, the tutorial uses a machine named manager1.
2. Run docker service inspect --pretty <SERVICE-ID> to display the details about a service in an easily readable format.

To see the details on the helloworld service:

[manager1]$ docker service inspect --pretty helloworld

ID: 9uk4639qpg7npwf3fn2aasksr

Name: helloworld

Service Mode: REPLICATED

Replicas: 1

Placement:

UpdateConfig:

Parallelism: 1

ContainerSpec:

Image: alpine

Args: ping docker.com

Resources:

Endpoint Mode: vip

**Tip**: To return the service details in json format, run the same command without the --pretty flag.

[manager1]$ docker service inspect helloworld

[

{

"ID": "9uk4639qpg7npwf3fn2aasksr",

"Version": {

"Index": 418

},

"CreatedAt": "2016-06-16T21:57:11.622222327Z",

"UpdatedAt": "2016-06-16T21:57:11.622222327Z",

"Spec": {

"Name": "helloworld",

"TaskTemplate": {

"ContainerSpec": {

"Image": "alpine",

"Args": [

"ping",

"docker.com"

]

},

"Resources": {

"Limits": {},

"Reservations": {}

},

"RestartPolicy": {

"Condition": "any",

"MaxAttempts": 0

},

"Placement": {}

},

"Mode": {

"Replicated": {

"Replicas": 1

}

},

"UpdateConfig": {

"Parallelism": 1

},

"EndpointSpec": {

"Mode": "vip"

}

},

"Endpoint": {

"Spec": {}

}

}

]

1. Run docker service ps <SERVICE-ID> to see which nodes are running the service:

[manager1]$ docker service ps helloworld

NAME IMAGE NODE DESIRED STATE LAST STATE

helloworld.1.8p1vev3fq5zm0mi8g0as41w35 alpine worker2 Running Running 3 minutes

In this case, the one instance of the helloworld service is running on the worker2 node. You may see the service running on your manager node. By default, manager nodes in a swarm can execute tasks just like worker nodes.

Swarm also shows you the DESIRED STATE and LAST STATE of the service task so you can see if tasks are running according to the service definition.

1. Run docker ps on the node where the task is running to see details about the container for the task.

**Tip**: If helloworld is running on a node other than your manager node, you must ssh to that node.

[worker2]$docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

e609dde94e47 alpine:latest "ping docker.com" 3 minutes ago Up 3 minutes helloworld.1.8p1vev3fq5zm0mi8g0as41w35

## **List nodes**

To view a list of nodes in the swarm run docker node ls from a manager node:

$ docker node ls

ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS

46aqrk4e473hjbt745z53cr3t node-5 Ready Active Reachable

61pi3d91s0w3b90ijw3deeb2q node-4 Ready Active Reachable

a5b2m3oghd48m8eu391pefq5u node-3 Ready Active

e7p8btxeu3ioshyuj6lxiv6g0 node-2 Ready Active

ehkv3bcimagdese79dn78otj5 \* node-1 Ready Active Leader

The AVAILABILITY column shows whether or not the scheduler can assign tasks to the node:

* Active means that the scheduler can assign tasks to the node.
* Pause means the scheduler doesn’t assign new tasks to the node, but existing tasks remain running.
* Drain means the scheduler doesn’t assign new tasks to the node. The scheduler shuts down any existing tasks and schedules them on an available node.

The MANAGER STATUS column shows node participation in the Raft consensus:

* No value indicates a worker node that does not participate in swarm management.
* Leader means the node is the primary manager node that makes all swarm management and orchestration decisions for the swarm.
* Reachable means the node is a manager node participating in the Raft consensus quorum. If the leader node becomes unavailable, the node is eligible for election as the new leader.
* Unavailable means the node is a manager that can’t communicate with other managers. If a manager node becomes unavailable, you should either join a new manager node to the swarm or promote a worker node to be a manager.

For more information on swarm administration refer to the [Swarm administration guide](https://docs.docker.com/engine/swarm/admin_guide/).

## **Inspect an individual node**

You can run docker node inspect <NODE-ID> on a manager node to view the details for an individual node. The output defaults to JSON format, but you can pass the --pretty flag to print the results in human-readable format. For example:

$ docker node inspect self --pretty

ID: ehkv3bcimagdese79dn78otj5

Hostname: node-1

Joined at: 2016-06-16 22:52:44.9910662 +0000 utc

Status:

State: Ready

Availability: Active

Manager Status:

Address: 172.17.0.2:2377

Raft Status: Reachable

Leader: Yes

Platform:

Operating System: linux

Architecture: x86\_64

Resources:

CPUs: 2

Memory: 1.954 GiB

Plugins:

Network: overlay, host, bridge, overlay, null

Volume: local

Engine Version: 1.12.0-dev