Drain a node on the swarm

Estimated reading time: 3 minutes

In earlier steps of the tutorial, all the nodes have been running with `ACTIVE` availability. The swarm manager can assign tasks to any `ACTIVE` node, so up to now all nodes have been available to receive tasks.

Sometimes, such as planned maintenance times, you need to set a node to `DRAIN` availability. `DRAIN` availability prevents a node from receiving new tasks from the swarm manager. It also means the manager stops tasks running on the node and launches replica tasks on a node with `ACTIVE` availability.

> **Important**: Setting a node to `DRAIN` does not remove standalone containers from that node, such as those created with `docker run`, `docker-compose up`, or the Docker Engine API. A node's status, including `DRAIN`, only affects the node's ability to schedule swarm service workloads.

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. Verify that all your nodes are actively available.

```
$ docker node ls

ID                            HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
1bcef6utixb0l0ca7gxuivsj0     worker2   Ready   Active
38ciaotwjuritcdtn9npbnkuz     worker1   Ready   Active
e216jshn25ckzbvmwlnh5jr3g *   manager1  Ready   Active        Leader
```

3. If you aren't still running the `redis` service from the rolling update (/engine/swarm/swarm-tutorial/rolling-update/) tutorial, start it now:

```
$ docker service create --replicas 3 --name redis --update-delay 10s redis:3.

c5uo6kdmzpon37mgj9mwglcfw
```

4. Run `docker service ps redis` to see how the swarm manager assigned the tasks to different nodes:

```
$ docker service ps redis

NAME                                    IMAGE        NODE     DESIRED STATE  CURRE
redis.1.7q92v0nr1hcgts2amcjyqg3pq  redis:3.0.6  manager1 Running        Runni
redis.2.7h2l8h3q3wqy5f66hlv9ddmi6  redis:3.0.6  worker1  Running        Runni
redis.3.9bg7cezvedmkgg6c8yzvbhwsd  redis:3.0.6  worker2  Running        Runni
```

   In this case the swarm manager distributed one task to each node. You may see the tasks distributed differently among the nodes in your environment.

5. Run `docker node update --availability drain <NODE-ID>` to drain a node that had a task assigned to it:

```
docker node update --availability drain worker1

worker1
```

6. Inspect the node to check its availability:

```
$ docker node inspect --pretty worker1

ID:                     38ciaotwjuritcdtn9npbnkuz
Hostname:               worker1
Status:
 State:                 Ready
 Availability:          Drain
...snip...
```

   The drained node shows `Drain` for `AVAILABILITY`.

7. Run `docker service ps redis` to see how the swarm manager updated the task assignments for the `redis` service:

```
$ docker service ps redis

NAME                                    IMAGE        NODE      DESIRED STATE
redis.1.7q92v0nr1hcgts2amcjyqg3pq       redis:3.0.6  manager1  Running
redis.2.b4hovzed7id8irg1to42egue8       redis:3.0.6  worker2   Running
 \_ redis.2.7h2l8h3q3wqy5f66hlv9ddmi6   redis:3.0.6  worker1   Shutdown
redis.3.9bg7cezvedmkgg6c8yzvbhwsd       redis:3.0.6  worker2   Running
```

   The swarm manager maintains the desired state by ending the task on a node with `Drain` availability and creating a new task on a node with `Active` availability.

8. Run `docker node update --availability active <NODE-ID>` to return the drained node to an active state:

```
$ docker node update --availability active worker1

worker1
```

9. Inspect the node to see the updated state:

```
$ docker node inspect --pretty worker1

ID:                     38ciaotwjuritcdtn9npbnkuz
Hostname:               worker1
Status:
 State:                 Ready
 Availability:          Active
...snip...
```

   When you set the node back to `Active` availability, it can receive new tasks:

   - during a service update to scale up
   - during a rolling update
   - when you set another node to `Drain` availability
   - when a task fails on another active node

What's next?

Learn how to use a swarm mode routing mesh (/engine/swarm/ingress/).

tutorial (/search/?q=tutorial), cluster management (/search/?q=cluster management), swarm (/search/?q=swarm), service (/search/?q=service), drain (/search/?q=drain)