Apply rolling updates to a service

Estimated reading time: 4 minutes

In a previous step of the tutorial, you scaled (/engine/swarm/swarm-tutorial/scale-service/) the number of instances of a service. In this part of the tutorial, you deploy a service based on the Redis 3.0.6 container tag. Then you upgrade the service to use the Redis 3.0.7 container image using rolling updates.

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. Deploy your Redis tag to the swarm and configure the swarm with a 10 second update delay. Note that the following example shows an older Redis tag:

   ```
   $ docker service create \
     --replicas 3 \
     --name redis \
     --update-delay 10s \
     redis:3.0.6

   0u6a4s31ybk7yw2wyvtikmu50
   ```

   You configure the rolling update policy at service deployment time.

   The `--update-delay` flag configures the time delay between updates to a service task or sets of tasks. You can describe the time `T` as a combination of the number of seconds `Ts`, minutes `Tm`, or hours `Th`. So `10m30s` indicates a 10 minute 30 second delay.

   By default the scheduler updates 1 task at a time. You can pass the `--update-parallelism` flag to configure the maximum number of service tasks that the scheduler updates simultaneously.

   By default, when an update to an individual task returns a state of `RUNNING`, the scheduler schedules another task to update until all tasks are updated. If, at any time during an update a task returns `FAILED`, the scheduler pauses the update. You can control the behavior using the `--update-failure-action` flag for `docker service create` or `docker service update`.

3. Inspect the `redis` service:

```
$ docker service inspect --pretty redis

ID:             0u6a4s31ybk7yw2wyvtikmu50
Name:           redis
Service Mode:   Replicated
 Replicas:      3
Placement:
 Strategy:          Spread
UpdateConfig:
 Parallelism:   1
 Delay:         10s
ContainerSpec:
 Image:         redis:3.0.6
Resources:
Endpoint Mode:  vip
```

4. Now you can update the container image for `redis`. The swarm manager applies the update to nodes according to the `UpdateConfig` policy:

   ```
   $ docker service update --image redis:3.0.7 redis
   redis
   ```

   The scheduler applies rolling updates as follows by default:

   - Stop the first task.
   - Schedule update for the stopped task.
   - Start the container for the updated task.
   - If the update to a task returns `RUNNING`, wait for the specified delay period then start the next task.
   - If, at any time during the update, a task returns `FAILED`, pause the update.

5. Run `docker service inspect --pretty redis` to see the new image in the desired state:

   ```
   $ docker service inspect --pretty redis

   ID:             0u6a4s31ybk7yw2wyvtikmu50
   Name:           redis
   Service Mode:   Replicated
    Replicas:      3
   Placement:
    Strategy:          Spread
   UpdateConfig:
    Parallelism:   1
    Delay:         10s
   ContainerSpec:
    Image:         redis:3.0.7
   Resources:
   Endpoint Mode:  vip
   ```

The output of `service inspect` shows if your update paused due to failure:

```
$ docker service inspect --pretty redis

ID:             0u6a4s31ybk7yw2wyvtikmu50
Name:           redis
...snip...
Update status:
 State:      paused
 Started:    11 seconds ago
 Message:    update paused due to failure or early termination of task 9p7ith
...snip...
```

To restart a paused update run `docker service update <SERVICE-ID>`. For example:

```
docker service update redis
```

To avoid repeating certain update failures, you may need to reconfigure the service by passing flags to `docker service update`.

6. Run `docker service ps <SERVICE-ID>` to watch the rolling update:

```
$ docker service ps redis

NAME                                   IMAGE        NODE       DESIRED STATE
redis.1.dos1zffgeofhagnve8w864fco      redis:3.0.7  worker1    Running
 \_ redis.1.88rdo6pa52ki8oqx6dogf04fh  redis:3.0.6  worker2    Shutdown
redis.2.9l3i4j85517skba5o7tn5m8g0      redis:3.0.7  worker2    Running
 \_ redis.2.66k185wilg8ele7ntu8f6nj6i  redis:3.0.6  worker1    Shutdown
redis.3.egiuiqpzrdbxks3wxgn8qib1g      redis:3.0.7  worker1    Running
 \_ redis.3.ctzktfddb2tepkr45qcmqln04  redis:3.0.6  mmanager1  Shutdown
```

Before Swarm updates all of the tasks, you can see that some are running `redis:3.0.6` while others are running `redis:3.0.7`. The output above shows the state once the rolling updates are done.

What’s next?

Next, learn about how to drain a node (/engine/swarm/swarm-tutorial/drain-node/) in the swarm.

tutorial (/search/?q=tutorial), cluster management (/search/?q=cluster management), swarm (/search/?q=swarm), service (/search/?q=service), rolling-update (/search/?q=rolling-update)