Service Management

Creating a Service

- Containers can be deployed to the swarm in much the same way as containers are run on a single host.
- A service is created and the image that should be used to deploy the container is specified. Additionally, port mappings and the number of replica containers to deploy across the swarm can be specified as required.
- The following example shows how to create a service consisting of three replica containers deployed within the swarm. The service is given a name that makes it easy to identify. Port mappings are defined. In this example, the service uses the nginxdemos/hello image:

```
docker service create --replicas 3 --name hello -p 80:80 nginxdemos/hello
```

> You can check which services are running on a swarm at any time:

```
docker service ls
```

Inspect a Service

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
```

Tips:

- 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": {}
    }
}
]
```

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

```
[manager1]$ docker service ps helloworld

NAME                                    IMAGE   NODE     DESIRED STATE  CURRENT
helloworld.1.8p1vev3fq5zm0mi8g0as41w35  alpine  worker2  Running        Running
```

Scale the Service

- Once you have deployed a service to a swarm, you are ready to use the Docker CLI to scale the number of containers in the service. Containers running in a service are called “tasks.”
- Run the following command to change the desired state of the service running in the swarm:

```
$ docker service scale <SERVICE-ID>=<NUMBER-OF-TASKS>
```

> Example:

```
$ docker service scale helloworld=5

helloworld scaled to 5
```

- Run `docker service ps <SERVICE-ID>` to see the updated task list:

```
$ docker service ps helloworld

NAME                                    IMAGE   NODE      DESIRED STATE  CURRENT
helloworld.1.8p1vev3fq5zm0mi8g0as41w35  alpine  worker2   Running        Running
helloworld.2.c7a7tcdq5s0uk3qr88mf8xco6  alpine  worker1   Running        Running
helloworld.3.6crl09vdcalvtfehfh69ogfb1  alpine  worker1   Running        Running
helloworld.4.auky6trawmdlcne8ad8phb0f1  alpine  manager1  Running        Running
helloworld.5.ba19kca06l18zujfwxyc5lkyn  alpine  worker2   Running        Running

helloworld scaled to 5
```

Tips:

- You can see that swarm has created 4 new tasks to scale to a total of 5 running instances of Alpine Linux. The tasks are distributed between the three nodes of the swarm. One is running on manager1.

Delete the Service Running on the Swarm

- Run docker service rm helloworld to remove the helloworld service.

```
$ docker service rm helloworld

helloworld
```

> Run docker `service inspect <SERVICE-ID>` to verify that the swarm manager removed the service. The CLI returns a message that the service is not found:

```
$ docker service inspect helloworld
[]
Error: no such service: helloworld
```

> Even though the service no longer exists, the task containers take a few seconds to clean up. You can use `docker ps` on the nodes to verify when the tasks have been removed.

```
$ docker ps

    CONTAINER ID        IMAGE               COMMAND                  CREATED
    db1651f50347        alpine:latest       "ping docker.com"        44 minutes
    43bf6e532a92        alpine:latest       "ping docker.com"        44 minutes
    5a0fb65d8fa7        alpine:latest       "ping docker.com"        44 minutes
    afb0ba67076f        alpine:latest       "ping docker.com"        44 minutes
    688172d3bfaa        alpine:latest       "ping docker.com"        45 minutes

$ docker ps
   CONTAINER ID        IMAGE               COMMAND             CREATED
```

Complementary Interactive Lesson about Service Management