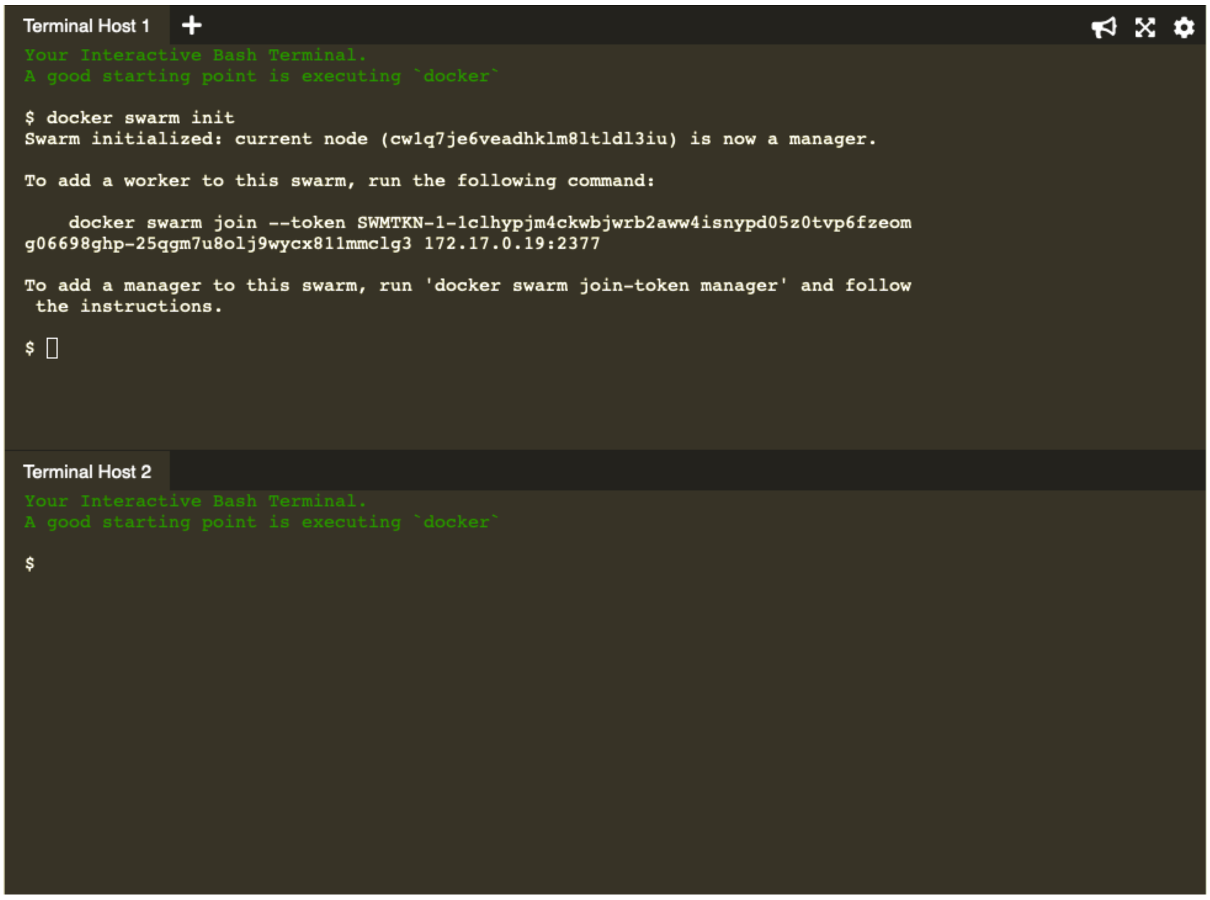
Name – Devashish Choudhary
Roll. No – R171218122
SAP ID – 500070510
DevOps Batch-2 (6th Semester)
Subject – Application Containerization

Experiment- ***Create Swarm Cluster***

- Initialize the Swarm Cluster into one of the terminal or virtual machine by using the following command.

```
$ docker swarm init
```



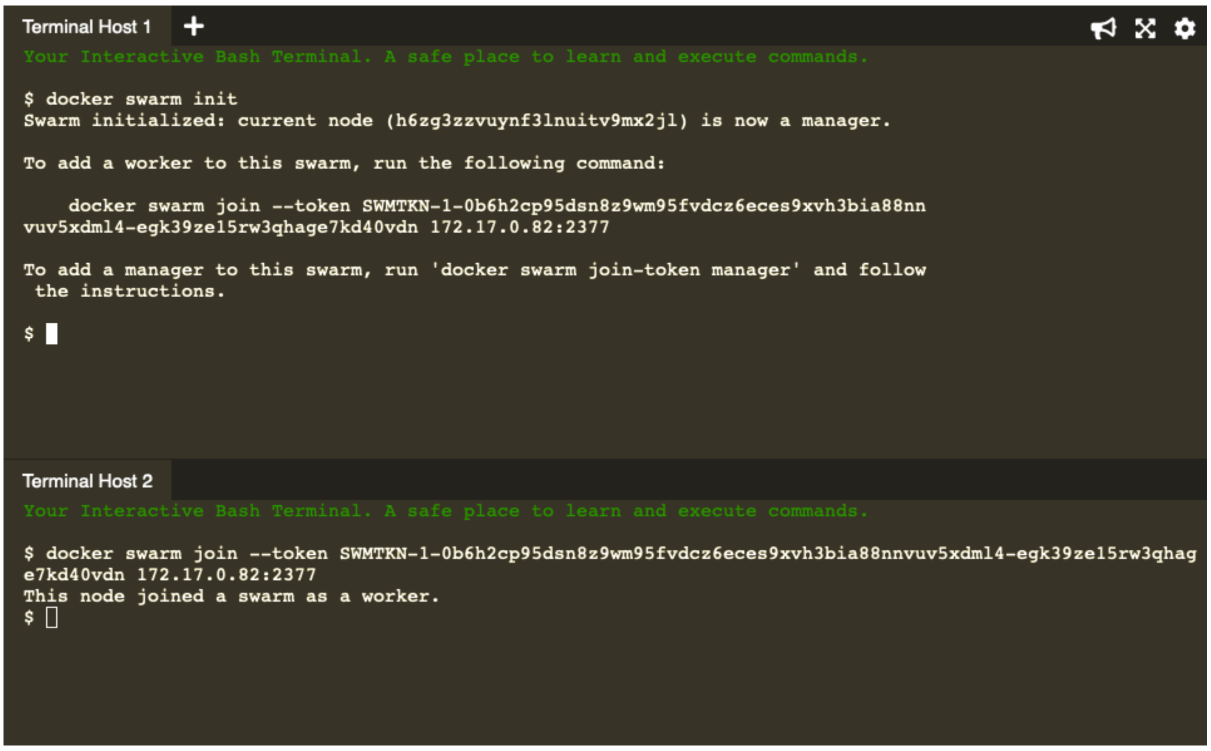
```
Your Interactive Bash Terminal.
A good starting point is executing `docker`

$ docker swarm init
Swarm initialized: current node (cwlq7je6veadhklm8ltldl3iu) is now a manager.

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

    docker swarm join --token SWMTKN-1-1clhypjm4ckwbjwrb2aww4isnypd05z0tvp6fzeom
g06698ghp-25qgm7u8olj9wycx811mmclg3 172.17.0.19:2377

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

$
```

```
Your Interactive Bash Terminal.
A good starting point is executing `docker`

$
```

- ***Join the Cluster : -***

To add a worker to this swarm, run the following command to join the node to this swarm.

```
$ docker swarm join --token SWMTKN-1-
0b6h2cp95dsn8z9wm95fvdcz6eces9xvh3bia88nnvuv5xdml4-
egk39ze15rw3qhage7kd40vdn 172.17.0.82:2377
```

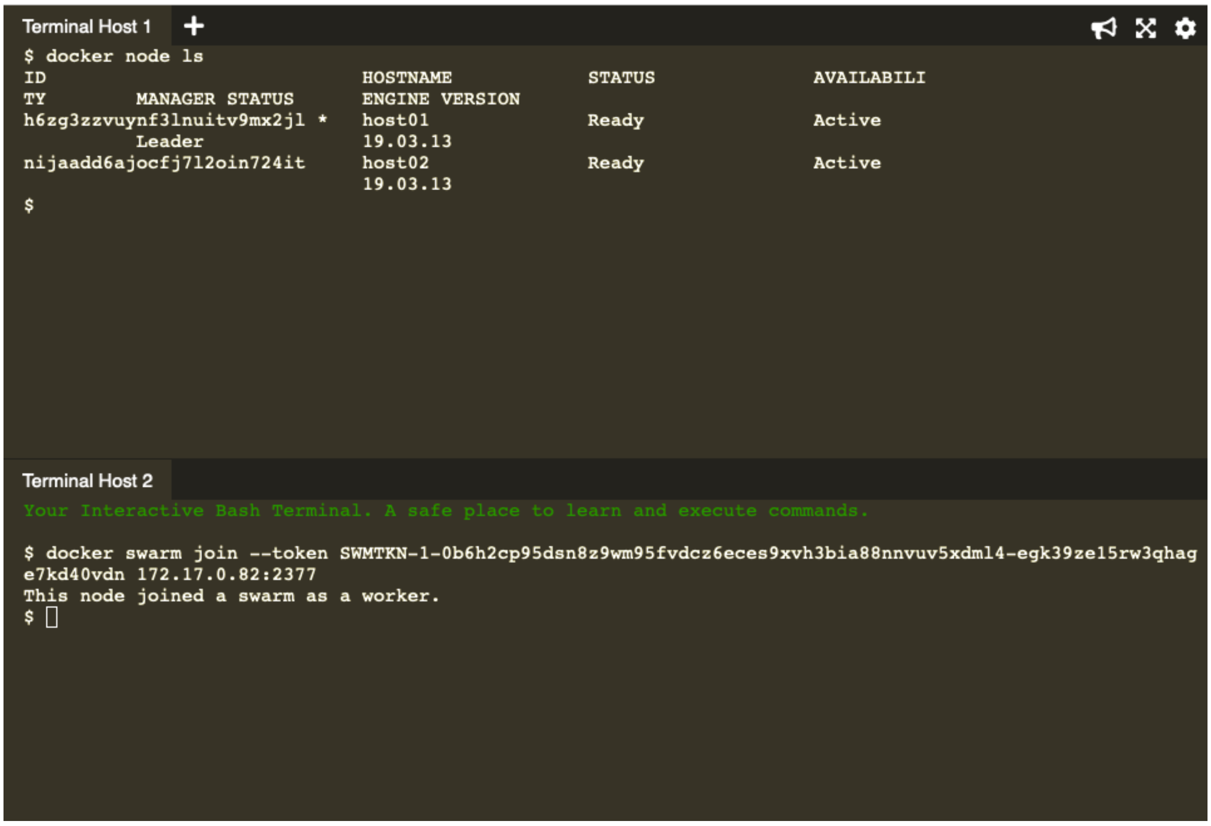


```
Your Interactive Bash Terminal. A safe place to learn and execute commands.

$ docker swarm init
Swarm initialized: current node (h6zg3zzvuynf3lnuitv9mx2jl) is now a manager.

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

    docker swarm join --token SWMTKN-1-0b6h2cp95dsn8z9wm95fvdcz6eces9xvh3bia88nn
vuv5xdml4-egk39ze15rw3qhage7kd40vdn 172.17.0.82:2377

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

$
```

```
Your Interactive Bash Terminal. A safe place to learn and execute commands.

$ docker swarm join --token SWMTKN-1-0b6h2cp95dsn8z9wm95fvdcz6eces9xvh3bia88nnvuv5xdml4-egk39ze15rw3qhag
e7kd40vdn 172.17.0.82:2377
This node joined a swarm as a worker.
$
```

- To see that how many nodes are joined in this Cluster by using the following command.

```
$ docker node ls
```

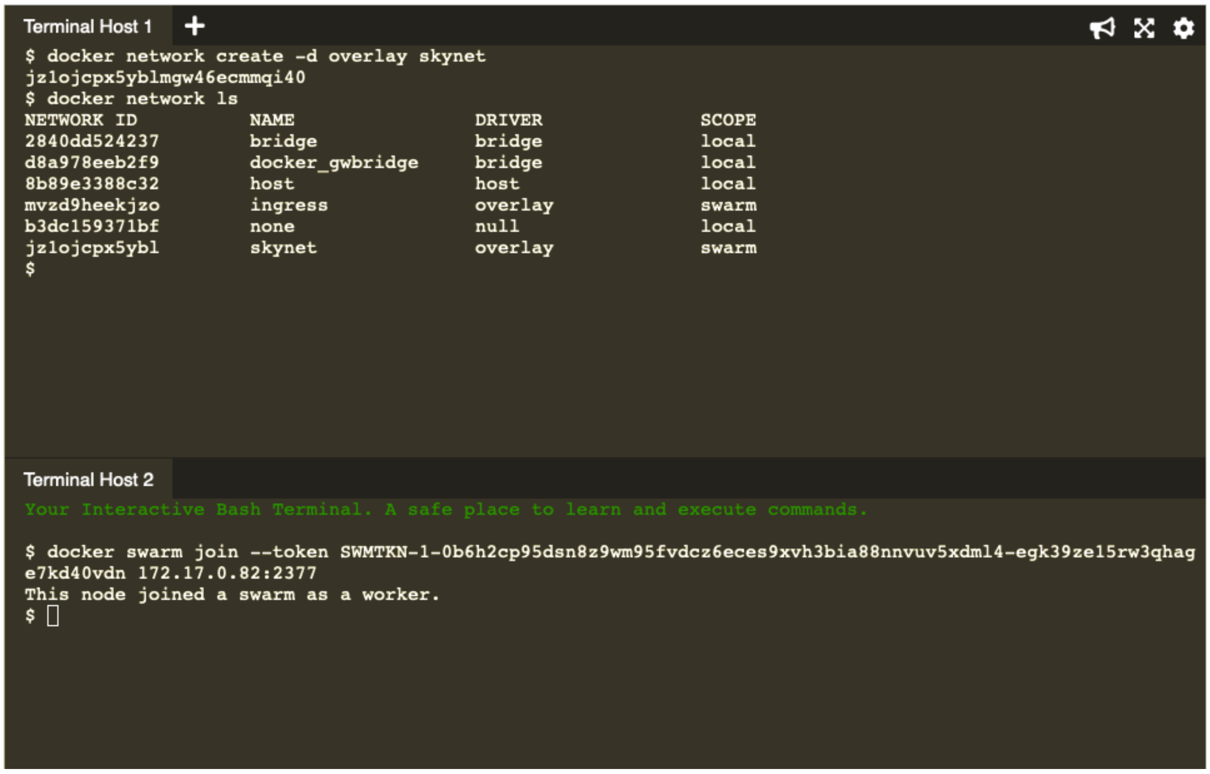


```
$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILI
TY          MANAGER STATUS      ENGINE VERSION
h6zg3zzvuynf3lnuitv9mx2jl *   host01              Ready               Active
            Leader              19.03.13
nijaadd6ajocfj7l2oin724it     host02              Ready               Active
                              19.03.13
$
```

```
Your Interactive Bash Terminal. A safe place to learn and execute commands.

$ docker swarm join --token SWMTKN-1-0b6h2cp95dsn8z9wm95fvdcz6eces9xvh3bia88nnvuv5xdml4-egk39ze15rw3qhag
e7kd40vdn 172.17.0.82:2377
This node joined a swarm as a worker.
$
```

- The following command will create a new overlay network called ***skynet*.** All containers registered to this network can communicate with each other, regardless of which node they are deployed onto.

```
$ docker network create -d overlay Skynet
```



Terminal Host 1

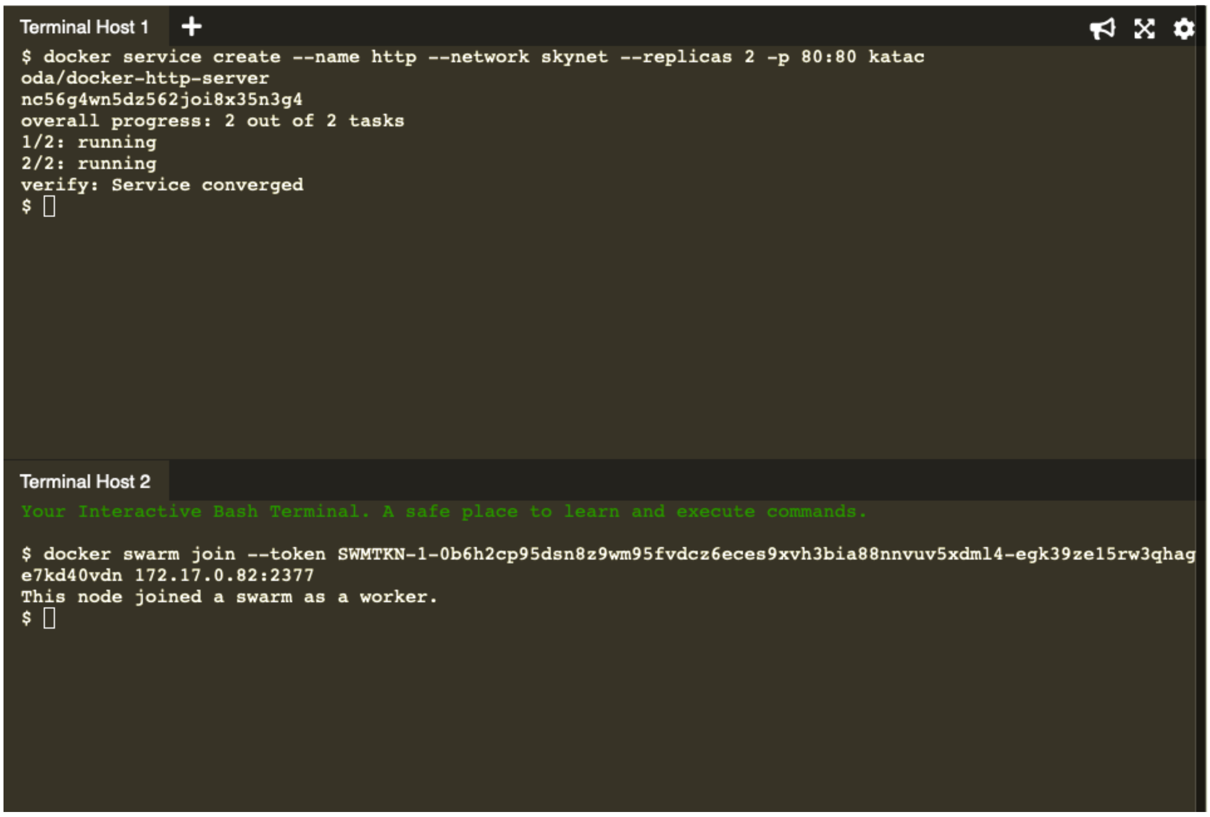
```
$ docker network create -d overlay skynet
jz1ojcpx5yblmgw46ecmmqi40
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
2840dd524237        bridge              bridge              local
d8a978eeb2f9        docker_gwbridge     bridge              local
8b89e3388c32        host                host                local
mvzd9heekjzo        ingress             overlay             swarm
b3dc159371bf        none                null                local
jz1ojcpx5ybl        skynet              overlay             swarm
$
```

Terminal Host 2

```
Your Interactive Bash Terminal. A safe place to learn and execute commands.

$ docker swarm join --token SWMTKN-1-0b6h2cp95dsn8z9wm95fvdcz6eces9xvh3bia88nnvuv5xdml4-egk39ze15rw3qhag
e7kd40vdn 172.17.0.82:2377
This node joined a swarm as a worker.
$ 
```

- Now we are deploying the Docker Image ***katacoda/docker-http-server*****.** We are defining a friendly name of a service called *http* and that it should be attached to the newly created ***skynet*** network.

```
$ docker service create --name http --network skynet --replicas 2 -p 80:80 katacoda/docker-http-server
```



Terminal Host 1

```
$ docker service create --name http --network skynet --replicas 2 -p 80:80 katac
oda/docker-http-server
nc56g4wn5dz562joi8x35n3g4
overall progress: 2 out of 2 tasks
1/2: running
2/2: running
verify: Service converged
$ 
```

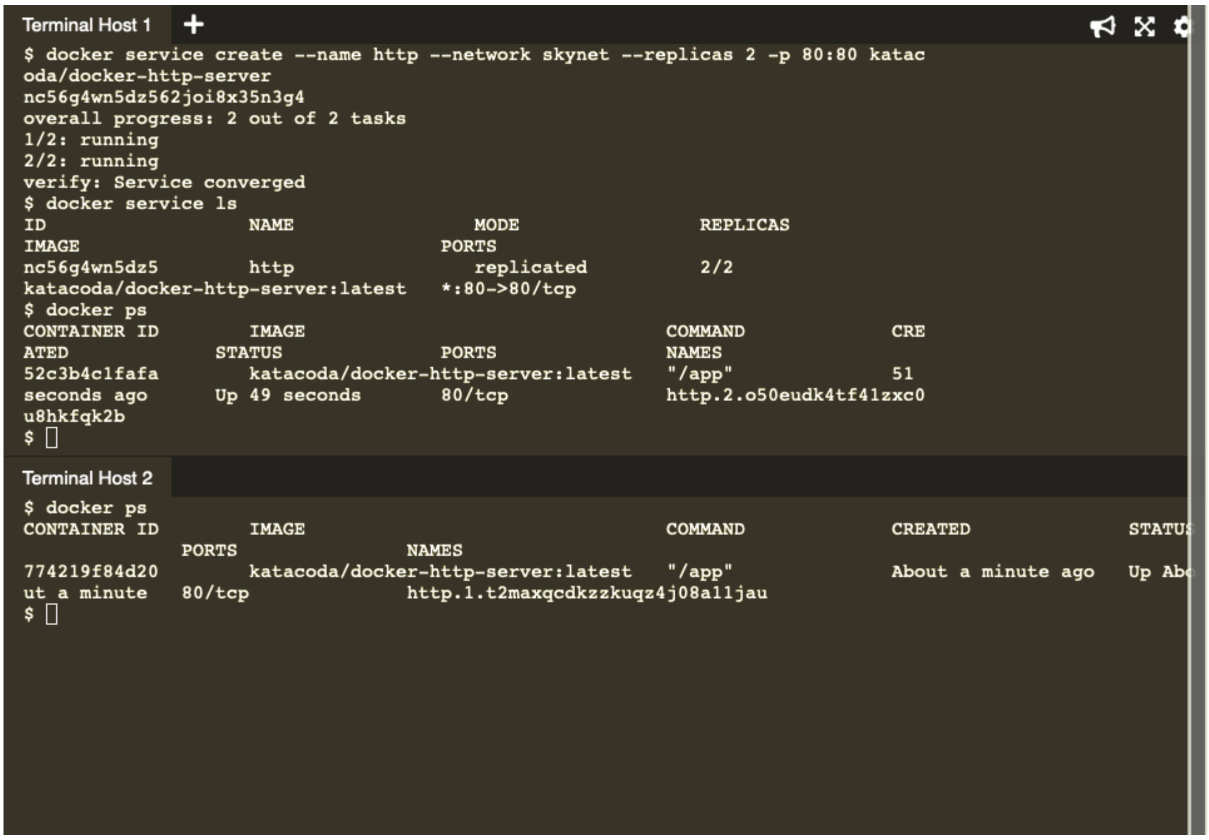
Terminal Host 2

```
Your Interactive Bash Terminal. A safe place to learn and execute commands.

$ docker swarm join --token SWMTKN-1-0b6h2cp95dsn8z9wm95fvdcz6eces9xvh3bia88nnvuv5xdml4-egk39ze15rw3qhag
e7kd40vdn 172.17.0.82:2377
This node joined a swarm as a worker.
$ 
```

- You can view the services running on the cluster using the CLI command.

  ```
  $ docker service ls
  ```

  As containers are started you will see them using the ***docker ps*** command. You should see one instance of the container on each host.



Terminal Host 1

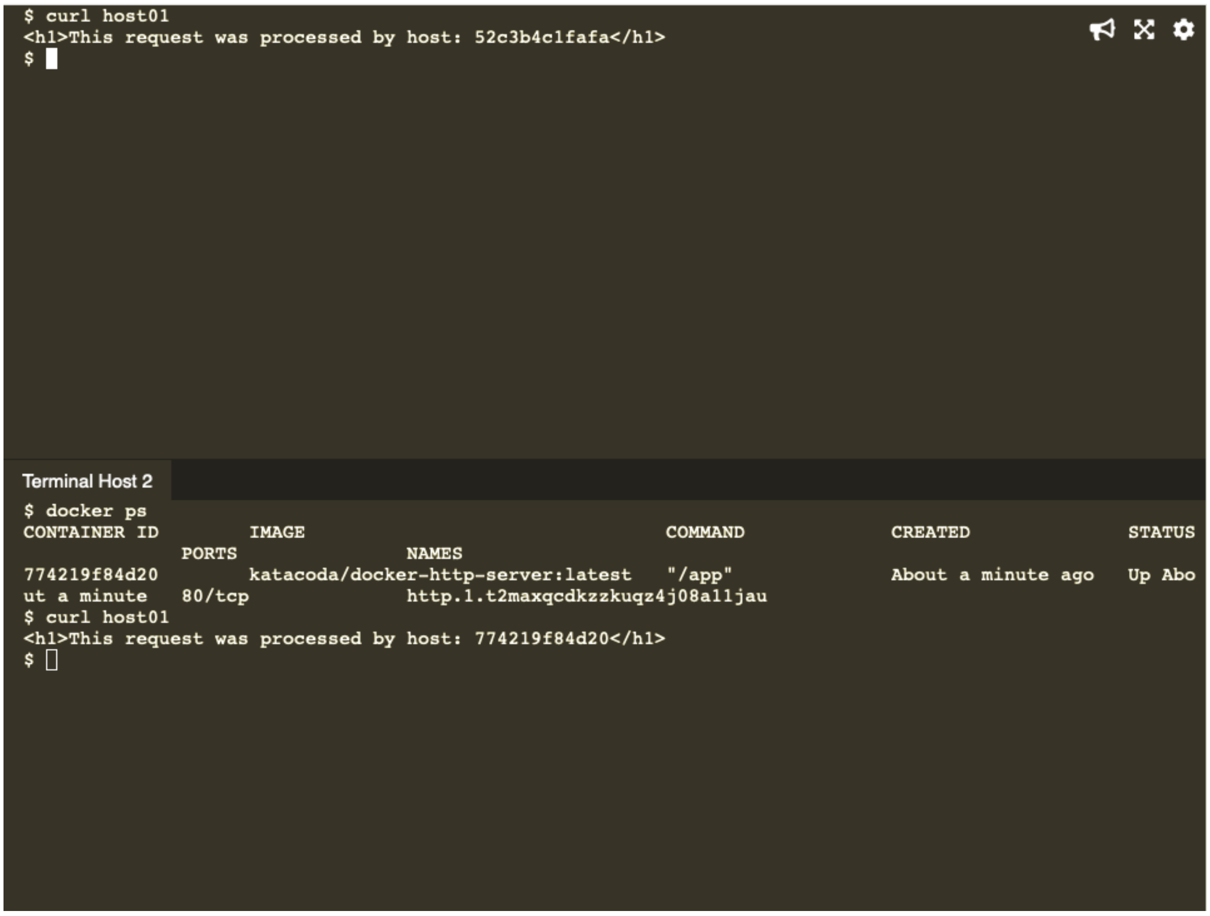
```
$ docker service create --name http --network skynet --replicas 2 -p 80:80 katac
oda/docker-http-server
nc56g4wn5dz562joi8x35n3g4
overall progress: 2 out of 2 tasks
1/2: running
2/2: running
verify: Service converged
$ docker service ls
ID                  NAME                MODE                REPLICAS
IMAGE                              PORTS
nc56g4wn5dz5        http                replicated          2/2
katacoda/docker-http-server:latest   *:80->80/tcp
$ docker ps
CONTAINER ID        IMAGE                                COMMAND             CRE
ATED               STATUS              PORTS               NAMES
52c3b4c1fafa        katacoda/docker-http-server:latest   "/app"              51
seconds ago        Up 49 seconds       80/tcp              http.2.o50eudk4tf41zxc0
u8hkfqk2b
$ 
```

Terminal Host 2

```
$ docker ps
CONTAINER ID        IMAGE                                COMMAND             CREATED             STATU
             PORTS               NAMES
774219f84d20        katacoda/docker-http-server:latest   "/app"              About a minute ago   Up Ab
ut a minute     80/tcp              http.1.t2maxqcdkzzkuqz4j08a1ljau
$ 
```

- If we issue an HTTP request to the public port, it will be processed by the two containers.

  ```
  $ curl host01
  ```



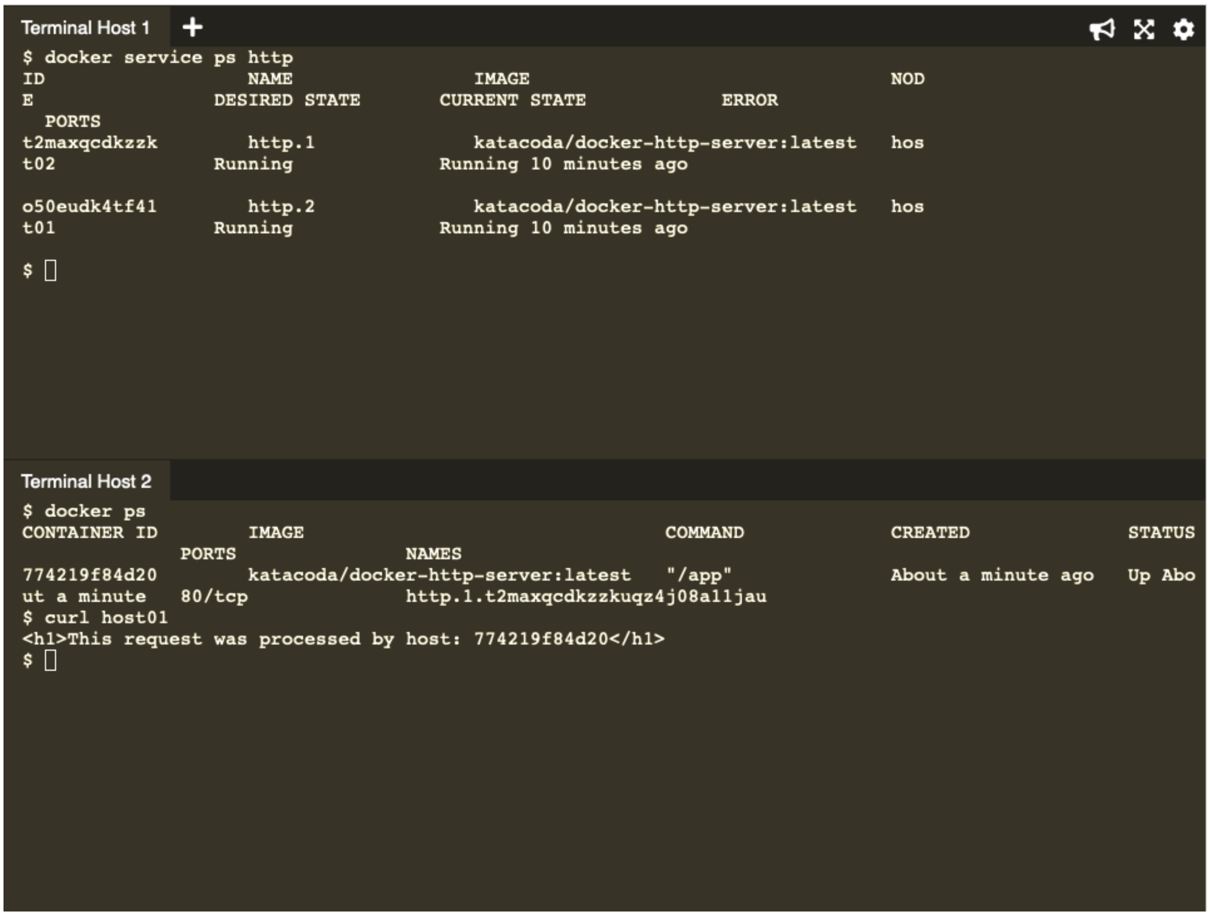
```
$ curl host01
<h1>This request was processed by host: 52c3b4c1fafa</h1>
$
```

Terminal Host 2

```
$ docker ps
CONTAINER ID        IMAGE                               COMMAND             CREATED              STATUS
              PORTS               NAMES
774219f84d20        katacoda/docker-http-server:latest   "/app"              About a minute ago   Up Abo
ut a minute   80/tcp              http.1.t2maxqcdkzzkuqz4j08a11jau
$ curl host01
<h1>This request was processed by host: 774219f84d20</h1>
$
```

- You can view the list of all the tasks associated with a service across the cluster. In this case, each task is a container.

```
$ docker service ps http
```

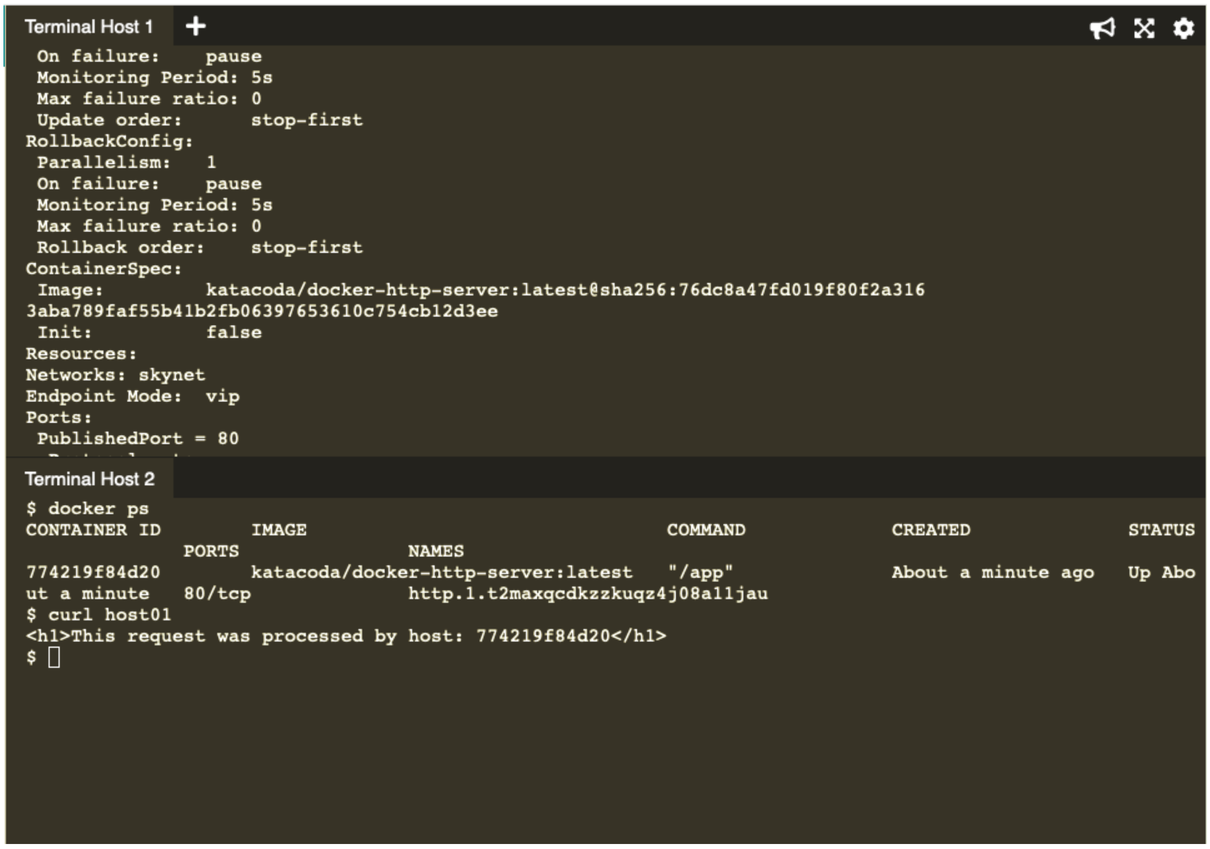


```
$ docker service ps http
ID                  NAME                IMAGE                                 NOD
E                 DESIRED STATE       CURRENT STATE            ERROR
  PORTS
t2maxqcdkzzk        http.1              katacoda/docker-http-server:latest   hos
t02               Running             Running 10 minutes ago

o50eudk4tf41        http.2              katacoda/docker-http-server:latest   hos
t01               Running             Running 10 minutes ago

$ 
```

```
$ docker ps
CONTAINER ID        IMAGE                                COMMAND             CREATED              STATUS
              PORTS               NAMES
774219f84d20        katacoda/docker-http-server:latest   "/app"              About a minute ago   Up Abo
ut a minute   80/tcp              http.1.t2maxqcdkzzkuqz4j08a11jau
$ curl host01
<h1>This request was processed by host: 774219f84d20</h1>
$ 
```

- You can view the details and configuration of a service via

```
$ docker service inspect --pretty http
```

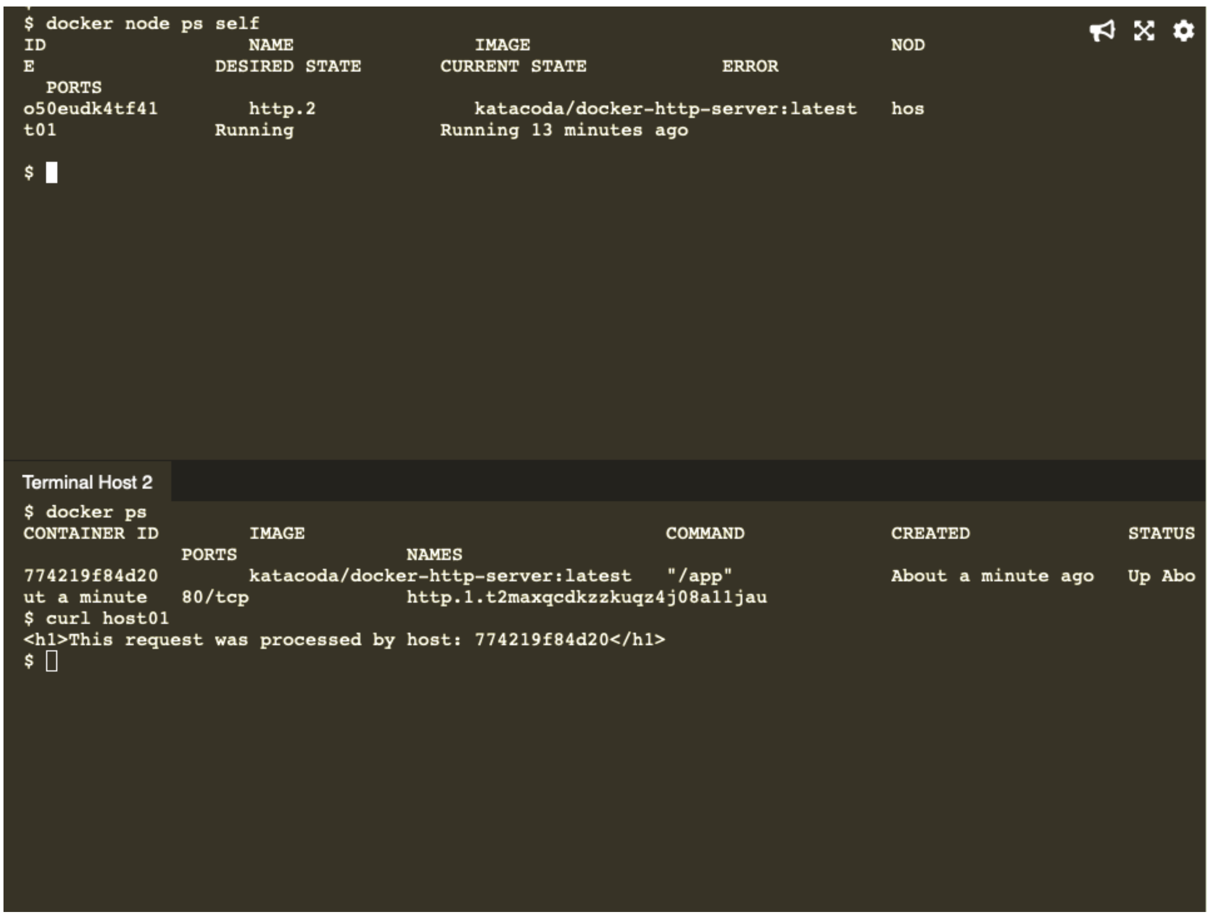


```
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Update order:      stop-first
RollbackConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Rollback order:    stop-first
ContainerSpec:
 Image:         katacoda/docker-http-server:latest@sha256:76dc8a47fd019f80f2a316
3aba789faf55b41b2fb06397653610c754cb12d3ee
 Init:          false
Resources:
Networks: skynet
Endpoint Mode:  vip
Ports:
 PublishedPort = 80
```

```
$ docker ps
CONTAINER ID        IMAGE                                COMMAND             CREATED              STATUS
              PORTS               NAMES
774219f84d20        katacoda/docker-http-server:latest   "/app"              About a minute ago   Up Abo
ut a minute   80/tcp              http.1.t2maxqcdkzzkuqz4j08a1ljau
$ curl host01
<h1>This request was processed by host: 774219f84d20</h1>
$ 
```

- On each node, you can ask what tasks it is currently running. Self refers to the manager node Leader:

```
$ docker node ps self
```



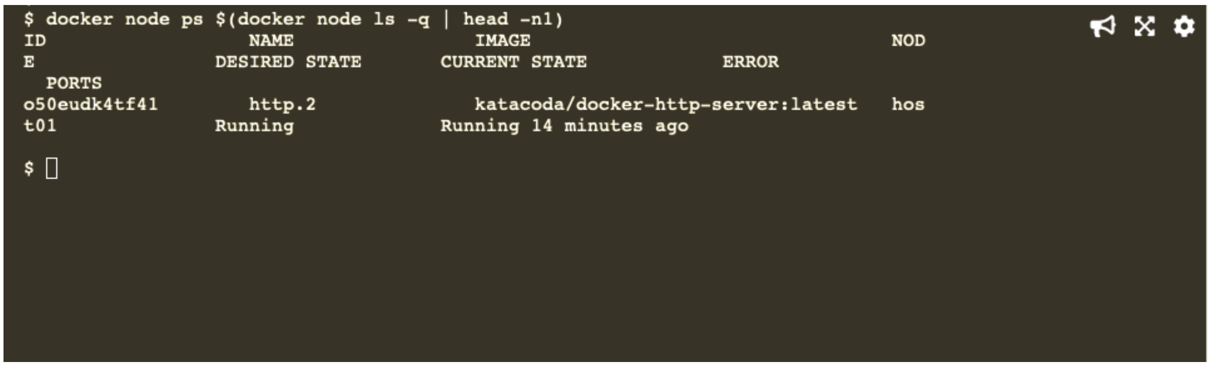
```
$ docker node ps self
ID                  NAME                IMAGE                                NOD
E                 DESIRED STATE       CURRENT STATE            ERROR
  PORTS
o50eudk4tf41        http.2              katacoda/docker-http-server:latest   hos
t01               Running             Running 13 minutes ago

$ 
```

Terminal Host 2

```
$ docker ps
CONTAINER ID        IMAGE                                COMMAND             CREATED             STATUS
              PORTS               NAMES
774219f84d20        katacoda/docker-http-server:latest   "/app"              About a minute ago  Up Abo
ut a minute   80/tcp              http.1.t2maxqcdkzzkuqz4j08a1ljau
$ curl host01
<h1>This request was processed by host: 774219f84d20</h1>
$ 
```

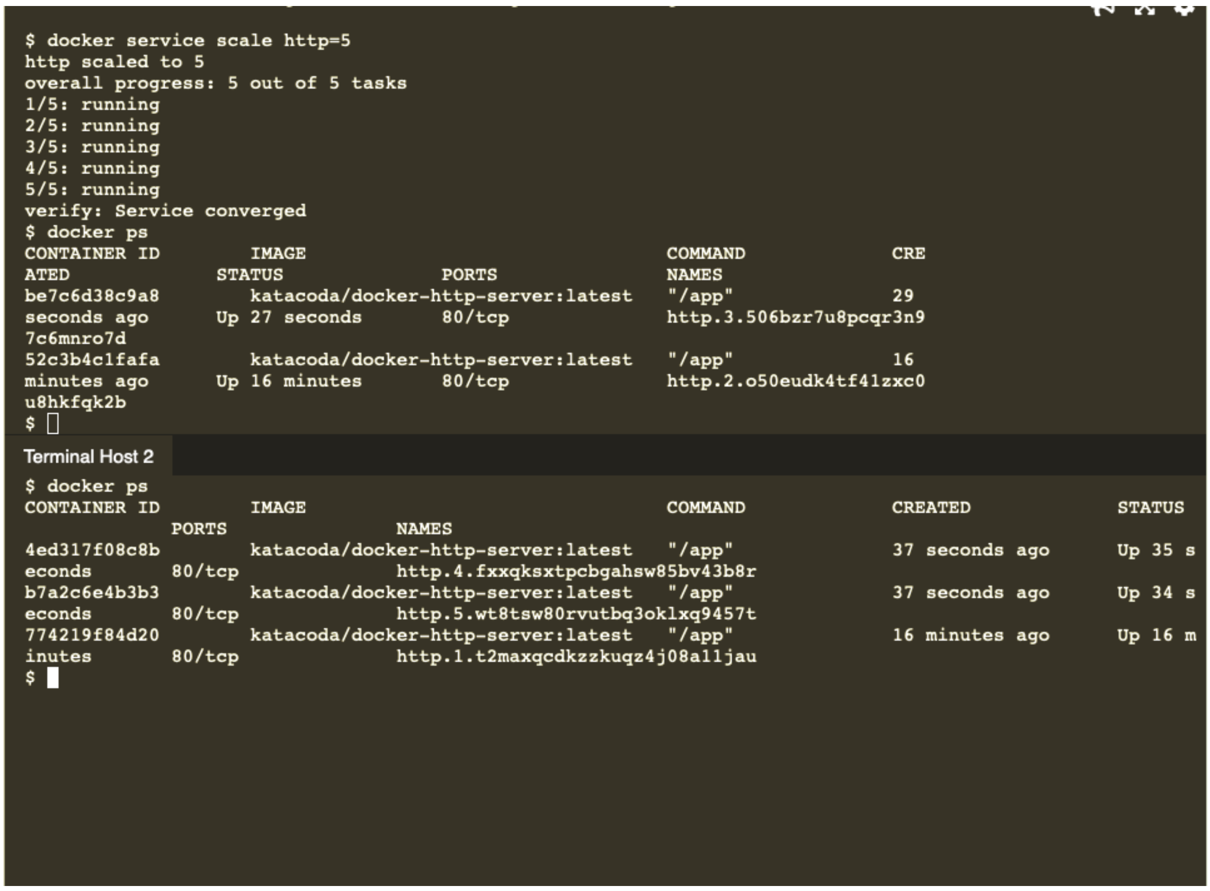
- Using the ID of a node you can query individual hosts.

```
$ docker node ps $(docker node ls -q | head -n1)
```



```
$ docker node ps $(docker node ls -q | head -n1)
ID                  NAME                IMAGE                                 NOD
E                 DESIRED STATE       CURRENT STATE            ERROR
  PORTS
o50eudk4tf41        http.2              katacoda/docker-http-server:latest   hos
t01               Running             Running 14 minutes ago

$ 
```

- The command below will scale our *http* service to be running across five containers.



```
$ docker service scale http=5
http scaled to 5
overall progress: 5 out of 5 tasks
1/5: running
2/5: running
3/5: running
4/5: running
5/5: running
verify: Service converged
$ docker ps
CONTAINER ID        IMAGE                                COMMAND             CRE
ATED             STATUS              PORTS               NAMES
be7c6d38c9a8        katacoda/docker-http-server:latest   "/app"              29
seconds ago      Up 27 seconds       80/tcp              http.3.506bzr7u8pcqr3n9
7c6mnro7d
52c3b4c1fafa        katacoda/docker-http-server:latest   "/app"              16
minutes ago      Up 16 minutes       80/tcp              http.2.o50eudk4tf41zxc0
u8hkfqk2b
$ 
```

Terminal Host 2

```
$ docker ps
CONTAINER ID        IMAGE                                COMMAND             CREATED             STATUS
              PORTS               NAMES
4ed317f08c8b        katacoda/docker-http-server:latest   "/app"              37 seconds ago      Up 35 s
econds        80/tcp              http.4.fxxqksxtpcbgahsw85bv43b8r
b7a2c6e4b3b3        katacoda/docker-http-server:latest   "/app"              37 seconds ago      Up 34 s
econds        80/tcp              http.5.wt8tsw80rvutbq3oklxq9457t
774219f84d20        katacoda/docker-http-server:latest   "/app"              16 minutes ago      Up 16 m
inutes        80/tcp              http.1.t2maxqcdkzzkuqz4j08a1ljau
$ 
```