Docker Swarm

Docker Swarm is a container orchestration tool that allows you to manage a cluster of Docker nodes as a single logical system. It provides several benefits, such as scalability, high availability, load balancing, and simplified deployment. Here are some use cases and examples of how Docker Swarm can be utilized:

1. High Availability Web Application

Use Case: Deploying a web application that requires high availability and redundancy.

Example:

- Create a Swarm cluster with multiple manager and worker nodes.
- Deploy a replicated service for the web application.
- Docker Swarm ensures that if one node fails, another node takes over, maintaining the application's availability.

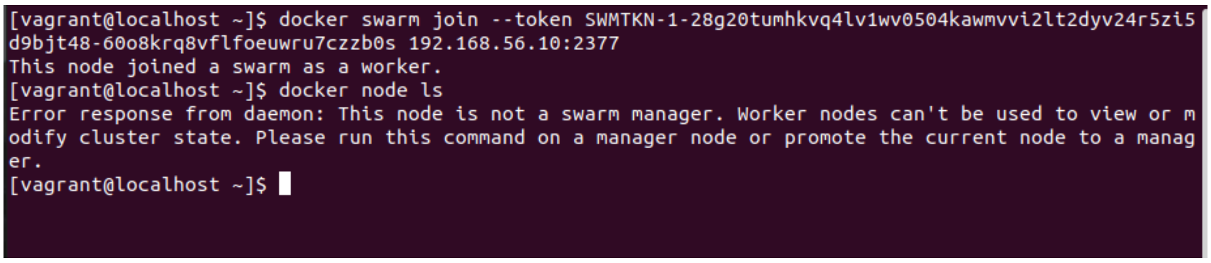
Steps:

Initialize Swarm:

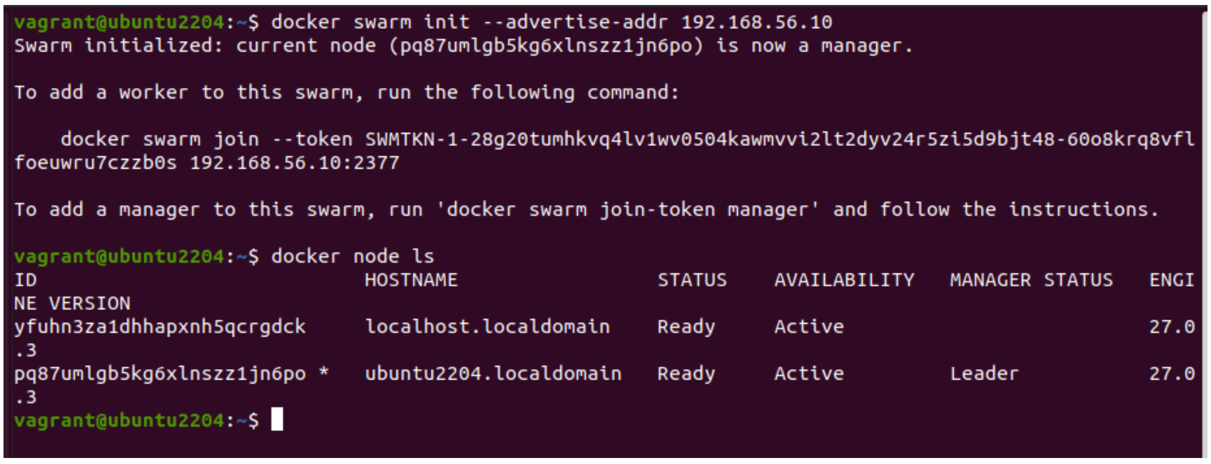
```
docker swarm init --advertise-addr <MANAGER-IP>
```

Add Worker Nodes: On each worker node:

```
docker swarm join --token <WORKER-TOKEN> <MANAGER-IP>:2377
```



```
[vagrant@localhost ~]$ docker swarm join --token SWMTKN-1-28g20tumhkvq4lv1wv0504kawmvvi2lt2dyv24r5zi5
d9bjt48-60o8krq8vflfoeuwru7czzb0s 192.168.56.10:2377
This node joined a swarm as a worker.
[vagrant@localhost ~]$ docker node ls
Error response from daemon: This node is not a swarm manager. Worker nodes can't be used to view or m
odify cluster state. Please run this command on a manager node or promote the current node to a manag
er.
[vagrant@localhost ~]$
```



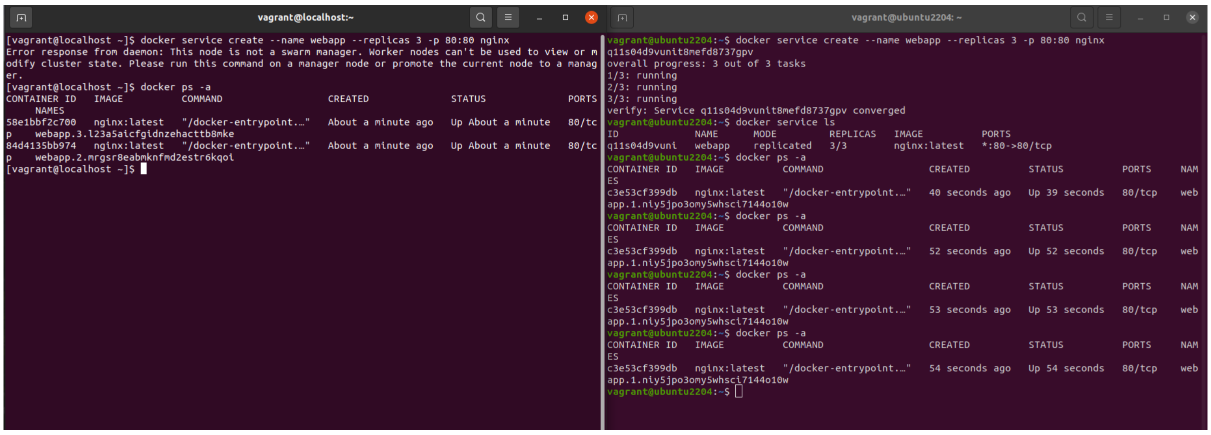
```
vagrant@ubuntu2204:~$ docker swarm init --advertise-addr 192.168.56.10
Swarm initialized: current node (pq87umlgb5kg6xlnszz1jn6po) is now a manager.

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

    docker swarm join --token SWMTKN-1-28g20tumhkvq4lv1wv0504kawmvvi2lt2dyv24r5zi5d9bjt48-60o8krq8vfl
foeuwru7czzb0s 192.168.56.10:2377

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

vagrant@ubuntu2204:~$ docker node ls
ID                            HOSTNAME                 STATUS    AVAILABILITY   MANAGER STATUS   ENGI
NE VERSION
yfuhn3za1dhhapxnh5qcrgdck     localhost.localdomain    Ready     Active                          27.0
.3
pq87umlgb5kg6xlnszz1jn6po *   ubuntu2204.localdomain   Ready     Active         Leader           27.0
.3
vagrant@ubuntu2204:~$
```

Deploy a Web Application:

```
docker service create --name webapp --replicas 3 -p 80:80 nginx
```

Check Service Status:

```
docker service ls
```



vagrant@localhost:~

```
[vagrant@localhost ~]$ docker service create --name webapp --replicas 3 -p 80:80 nginx
Error response from daemon: This node is not a swarm manager. Worker nodes can't be used to view or m
odify cluster state. Please run this command on a manager node or promote the current node to a manag
er.
[vagrant@localhost ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS
     NAMES
58e1bbf2c700   nginx:latest   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tc
p    webapp.3.l23a5aicfgidnzehacttb8mke
84d4135bb974   nginx:latest   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tc
p    webapp.2.mrgsr8eabmknfmd2estr6kqoi
[vagrant@localhost ~]$
```

vagrant@ubuntu2204: ~

```
vagrant@ubuntu2204:~$ docker service create --name webapp --replicas 3 -p 80:80 nginx
q11s04d9vunit8mefd8737gpv
overall progress: 3 out of 3 tasks
1/3: running
2/3: running
3/3: running
verify: Service q11s04d9vunit8mefd8737gpv converged
vagrant@ubuntu2204:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE          PORTS
q11s04d9vuni   webapp    replicated   3/3        nginx:latest   *:80->80/tcp
vagrant@ubuntu2204:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAM
ES
c3e53cf399db   nginx:latest   "/docker-entrypoint.…"   40 seconds ago   Up 39 seconds   80/tcp    web
app.1.niy5jpo3omy5whsci7144o10w
vagrant@ubuntu2204:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAM
ES
c3e53cf399db   nginx:latest   "/docker-entrypoint.…"   52 seconds ago   Up 52 seconds   80/tcp    web
app.1.niy5jpo3omy5whsci7144o10w
vagrant@ubuntu2204:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAM
ES
c3e53cf399db   nginx:latest   "/docker-entrypoint.…"   53 seconds ago   Up 53 seconds   80/tcp    web
app.1.niy5jpo3omy5whsci7144o10w
vagrant@ubuntu2204:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAM
ES
c3e53cf399db   nginx:latest   "/docker-entrypoint.…"   54 seconds ago   Up 54 seconds   80/tcp    web
app.1.niy5jpo3omy5whsci7144o10w
vagrant@ubuntu2204:~$
```

2. Continuous Integration/Continuous Deployment (CI/CD) Pipeline

Use Case: Automating the deployment of applications with a CI/CD pipeline.

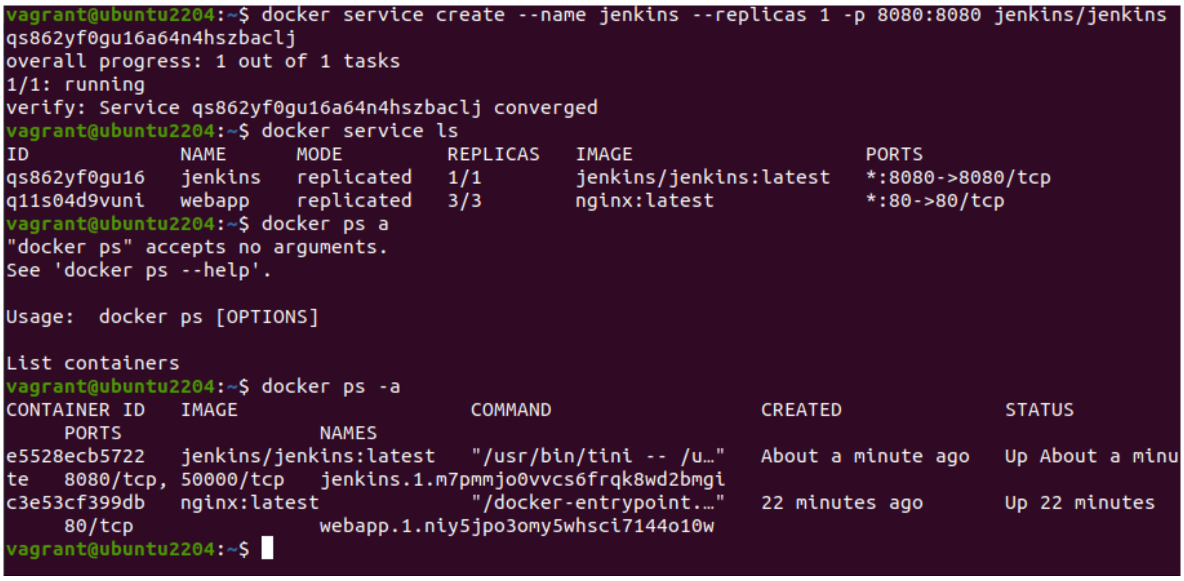
Example:

- Use Docker Swarm to deploy applications automatically when new code is committed.
- Integrate with CI/CD tools like Jenkins, GitLab CI, or GitHub Actions.

Steps:

Initialize Swarm and Deploy Jenkins:

```
docker swarm init
docker service create --name jenkins --replicas 1 -p 8080:8080 jenkins/jenkins
```



```
vagrant@ubuntu2204:~$ docker service create --name jenkins --replicas 1 -p 8080:8080 jenkins/jenkins
qs862yf0gu16a64n4hszbaclj
overall progress: 1 out of 1 tasks
1/1: running
verify: Service qs862yf0gu16a64n4hszbaclj converged
vagrant@ubuntu2204:~$ docker service ls
ID             NAME      MODE         REPLICAS   IMAGE                    PORTS
qs862yf0gu16   jenkins   replicated   1/1        jenkins/jenkins:latest   *:8080->8080/tcp
q11s04d9vuni   webapp    replicated   3/3        nginx:latest             *:80->80/tcp
vagrant@ubuntu2204:~$ docker ps a
"docker ps" accepts no arguments.
See 'docker ps --help'.

Usage:  docker ps [OPTIONS]

List containers
vagrant@ubuntu2204:~$ docker ps -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED              STATUS
     PORTS                 NAMES
e5528ecb5722   jenkins/jenkins:latest   "/usr/bin/tini -- /u…"   About a minute ago   Up About a minu
te   8080/tcp, 50000/tcp   jenkins.1.m7pmmjo0vvcs6frqk8wd2bmgi
c3e53cf399db   nginx:latest             "/docker-entrypoint.…"   22 minutes ago       Up 22 minutes
     80/tcp                webapp.1.niy5jpo3omy5whsci7144o10w
vagrant@ubuntu2204:~$
```

1. **Configure Jenkins to Deploy to Swarm:**
   - Set up Jenkins with necessary plugins for Docker and Docker Swarm.
   - Create a Jenkins pipeline that builds Docker images and deploys them to the Swarm cluster.
2. **Automate Deployment:**
   - Configure Jenkins to trigger builds and deployments on code changes.

3. Load Balancing and Scaling Services

Use Case: Distributing traffic across multiple instances of a service for load balancing and scaling.

Example:

- Deploy a service with multiple replicas.
- Docker Swarm automatically load balances requests across these replicas.

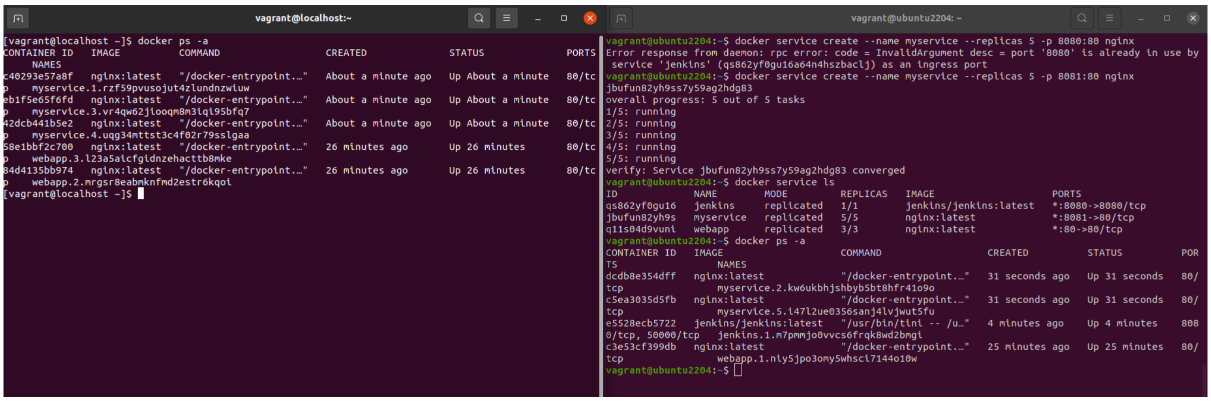
Steps:

Initialize Swarm:

```
docker swarm init
```

Deploy a Service with Load Balancing:

```
docker service create --name myservice --replicas 5 -p 8080:80 nginx
```

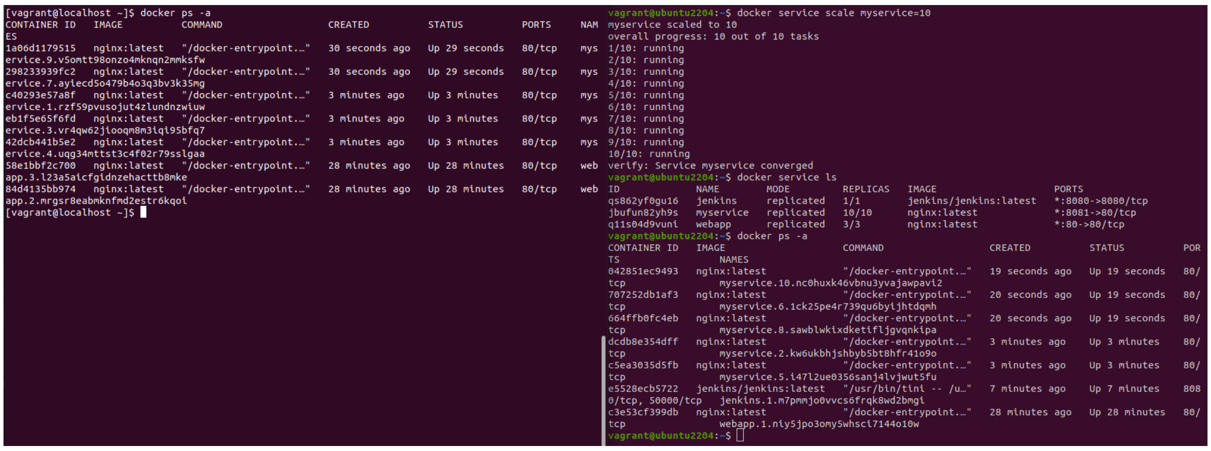


```
[vagrant@localhost ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS
     NAMES
c40293e57a8f   nginx:latest   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tc
p    myservice.1.rzf59pvusojut4zlundnzwiuw
eb1f5e65f6fd   nginx:latest   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tc
p    myservice.3.vr4qw62jiooqm8m3iqi95bfq7
42dcb441b5e2   nginx:latest   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tc
p    myservice.4.uqg34mttst3c4f02r79sslgaa
58e1bbf2c700   nginx:latest   "/docker-entrypoint.…"   26 minutes ago       Up 26 minutes       80/tc
p    webapp.3.l23a5aicfgidnzehacttb8mke
84d4135bb974   nginx:latest   "/docker-entrypoint.…"   26 minutes ago       Up 26 minutes       80/tc
p    webapp.2.mrgsr8eabmknfmd2estr6kqoi
[vagrant@localhost ~]$
```

```
vagrant@ubuntu2204:~$ docker service create --name myservice --replicas 5 -p 8080:80 nginx
Error response from daemon: rpc error: code = InvalidArgument desc = port '8080' is already in use by
 service 'jenkins' (qs862yf0gu16a64n4hszbaclj) as an ingress port
vagrant@ubuntu2204:~$ docker service create --name myservice --replicas 5 -p 8081:80 nginx
jbufun82yh9ss7y59ag2hdg83
overall progress: 5 out of 5 tasks
1/5: running
2/5: running
3/5: running
4/5: running
5/5: running
verify: Service jbufun82yh9ss7y59ag2hdg83 converged
vagrant@ubuntu2204:~$ docker service ls
ID             NAME        MODE         REPLICAS   IMAGE                    PORTS
qs862yf0gu16   jenkins     replicated   1/1        jenkins/jenkins:latest   *:8080->8080/tcp
jbufun82yh9s   myservice   replicated   5/5        nginx:latest             *:8081->80/tcp
q11s04d9vuni   webapp      replicated   3/3        nginx:latest             *:80->80/tcp
vagrant@ubuntu2204:~$ docker ps -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          POR
TS                 NAMES
dcdb8e354dff   nginx:latest             "/docker-entrypoint.…"   31 seconds ago   Up 31 seconds   80/
tcp                myservice.2.kw6ukbhjshbyb5bt8hfr41o9o
c5ea3035d5fb   nginx:latest             "/docker-entrypoint.…"   31 seconds ago   Up 31 seconds   80/
tcp                myservice.5.i47l2ue0356sanj4lvjwut5fu
e5528ecb5722   jenkins/jenkins:latest   "/usr/bin/tini -- /u…"   4 minutes ago    Up 4 minutes    808
0/tcp, 50000/tcp   jenkins.1.m7pmmjo0vvcs6frqk8wd2bmgi
c3e53cf399db   nginx:latest             "/docker-entrypoint.…"   25 minutes ago   Up 25 minutes   80/
tcp                webapp.1.niy5jpo3omy5whsci7144o10w
vagrant@ubuntu2204:~$
```

Scale the Service:

```
docker service scale myservice=10
```



```
[vagrant@localhost ~]$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAM
ES
1a06d1179515   nginx:latest   "/docker-entrypoint.…"   30 seconds ago   Up 29 seconds   80/tcp    mys
ervice.9.v5omtt98onzo4mknqn2mmksfw
298233939fc2   nginx:latest   "/docker-entrypoint.…"   30 seconds ago   Up 29 seconds   80/tcp    mys
ervice.7.ayiecd5o479b4o3q3bv3k35mg
c40293e57a8f   nginx:latest   "/docker-entrypoint.…"   3 minutes ago    Up 3 minutes    80/tcp    mys
ervice.1.rzf59pvusojut4zlundnzwiuw
eb1f5e65f6fd   nginx:latest   "/docker-entrypoint.…"   3 minutes ago    Up 3 minutes    80/tcp    mys
ervice.3.vr4qw62jiooqm8m3iqi95bfq7
42dcb441b5e2   nginx:latest   "/docker-entrypoint.…"   3 minutes ago    Up 3 minutes    80/tcp    mys
ervice.4.uqg34mttst3c4f02r79sslgaa
58e1bbf2c700   nginx:latest   "/docker-entrypoint.…"   28 minutes ago   Up 28 minutes   80/tcp    web
app.3.l23a5aicfgidnzehacttb8mke
84d4135bb974   nginx:latest   "/docker-entrypoint.…"   28 minutes ago   Up 28 minutes   80/tcp    web
app.2.mrgsr8eabmknfmd2estr6kqoi
[vagrant@localhost ~]$
```

```
vagrant@ubuntu2204:~$ docker service scale myservice=10
myservice scaled to 10
overall progress: 10 out of 10 tasks
1/10: running
2/10: running
3/10: running
4/10: running
5/10: running
6/10: running
7/10: running
8/10: running
9/10: running
10/10: running
verify: Service myservice converged
vagrant@ubuntu2204:~$ docker service ls
ID             NAME        MODE         REPLICAS   IMAGE                         PORTS
qs862yf0gu16   jenkins     replicated   1/1        jenkins/jenkins:latest   *:8080->8080/tcp
jbufun82yh9s   myservice   replicated   10/10      nginx:latest                  *:8081->80/tcp
q11s04d9vuni   webapp      replicated   3/3        nginx:latest                  *:80->80/tcp
vagrant@ubuntu2204:~$ docker ps -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          POR
TS                 NAMES
042851ec9493   nginx:latest             "/docker-entrypoint.…"   19 seconds ago   Up 19 seconds   80/
tcp                myservice.10.nc0huxk46vbnu3yvajawpavi2
707252db1af3   nginx:latest             "/docker-entrypoint.…"   20 seconds ago   Up 19 seconds   80/
tcp                myservice.6.1ck25pe4r739qu6byijhtdqmh
664ffb0fc4eb   nginx:latest             "/docker-entrypoint.…"   20 seconds ago   Up 19 seconds   80/
tcp                myservice.8.sawblwkixdketifljgvqnkipa
dcdb8e354dff   nginx:latest             "/docker-entrypoint.…"   3 minutes ago    Up 3 minutes    80/
tcp                myservice.2.kw6ukbhjshbyb5bt8hfr41o9o
c5ea3035d5fb   nginx:latest             "/docker-entrypoint.…"   3 minutes ago    Up 3 minutes    80/
tcp                myservice.5.i47l2ue0356sanj4lvjwut5fu
e5528ecb5722   jenkins/jenkins:latest   "/usr/bin/tini -- /u…"   7 minutes ago    Up 7 minutes    808
0/tcp, 50000/tcp   jenkins.1.m7pmmjo0vvcs6frqk8wd2bmgi
c3e53cf399db   nginx:latest             "/docker-entrypoint.…"   28 minutes ago   Up 28 minutes   80/
tcp                webapp.1.niy5jpo3omy5whsci7144o10w
vagrant@ubuntu2204:~$
```

4. Microservices Architecture

Use Case: Deploying a microservices-based application with multiple interdependent services.

Example:

- Use Docker Swarm to manage the deployment and scaling of each microservice.
- Ensure communication between services through the Swarm network.

Steps:

Initialize Swarm:

```
docker swarm init
```

Deploy Microservices:

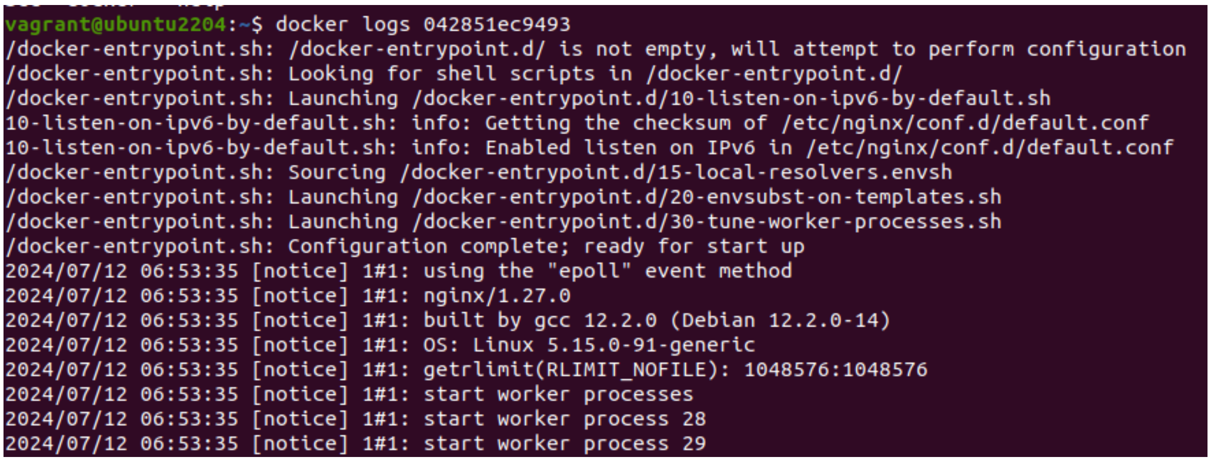
```
docker service create --name service1 --replicas 3 -p 5000:5000
my_microservice1

docker service create --name service2 --replicas 2 -p 5001:5001
my_microservice2
```

1. **Ensure Services Communicate:**
   - Use Docker Swarm's service discovery to enable communication between services using their service names.

Docker Logs

To view the logs of a container, you can use the following command:

docker logs <container_name_or_id>



```
vagrant@ubuntu2204:~$ docker logs 042851ec9493
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/07/12 06:53:35 [notice] 1#1: using the "epoll" event method
2024/07/12 06:53:35 [notice] 1#1: nginx/1.27.0
2024/07/12 06:53:35 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/07/12 06:53:35 [notice] 1#1: OS: Linux 5.15.0-91-generic
2024/07/12 06:53:35 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/07/12 06:53:35 [notice] 1#1: start worker processes
2024/07/12 06:53:35 [notice] 1#1: start worker process 28
2024/07/12 06:53:35 [notice] 1#1: start worker process 29
```

Options

Here are some useful options for the docker logs command:

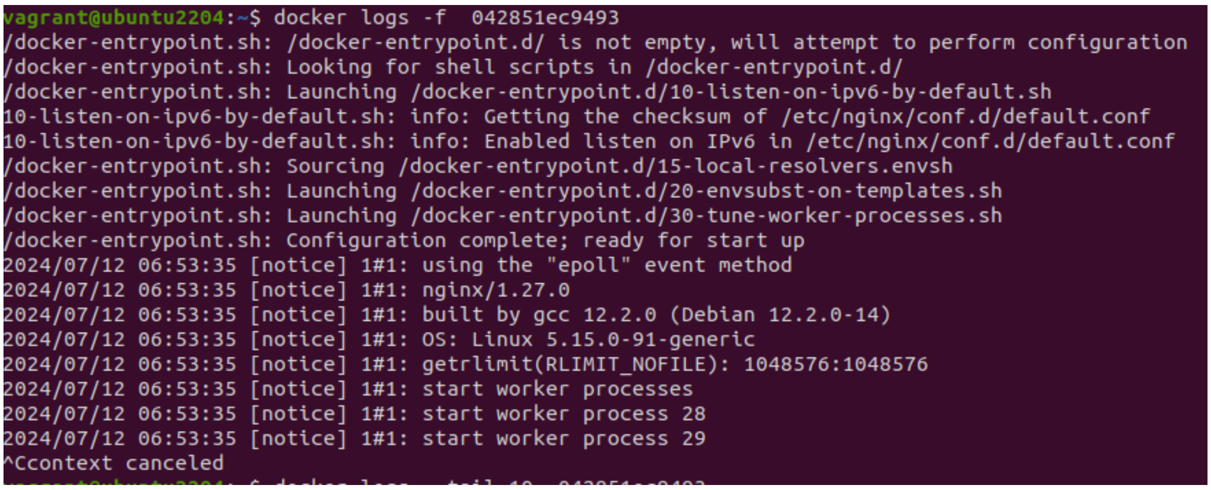
- **-f, --follow**: Follow log output (similar to tail -f).
- **--tail**: Show only the last N lines of log output.
- **-t, --timestamps**: Show timestamps for each log entry.
- **--since**: Show logs since a specific time (e.g., 2022-07-01T13:23:37 or 10m for last 10 minutes).
- **--until**: Show logs up until a specific time.

1. Viewing Logs of a Container

docker logs my_container

2. Following Logs in Real-Time

docker logs -f my_container



```
vagrant@ubuntu2204:~$ docker logs -f  042851ec9493
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/07/12 06:53:35 [notice] 1#1: using the "epoll" event method
2024/07/12 06:53:35 [notice] 1#1: nginx/1.27.0
2024/07/12 06:53:35 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/07/12 06:53:35 [notice] 1#1: OS: Linux 5.15.0-91-generic
2024/07/12 06:53:35 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/07/12 06:53:35 [notice] 1#1: start worker processes
2024/07/12 06:53:35 [notice] 1#1: start worker process 28
2024/07/12 06:53:35 [notice] 1#1: start worker process 29
^Ccontext canceled
```

3. Showing the Last 10 Lines of Logs

```
docker logs --tail 10 my_container
```

4. Showing Logs with Timestamps

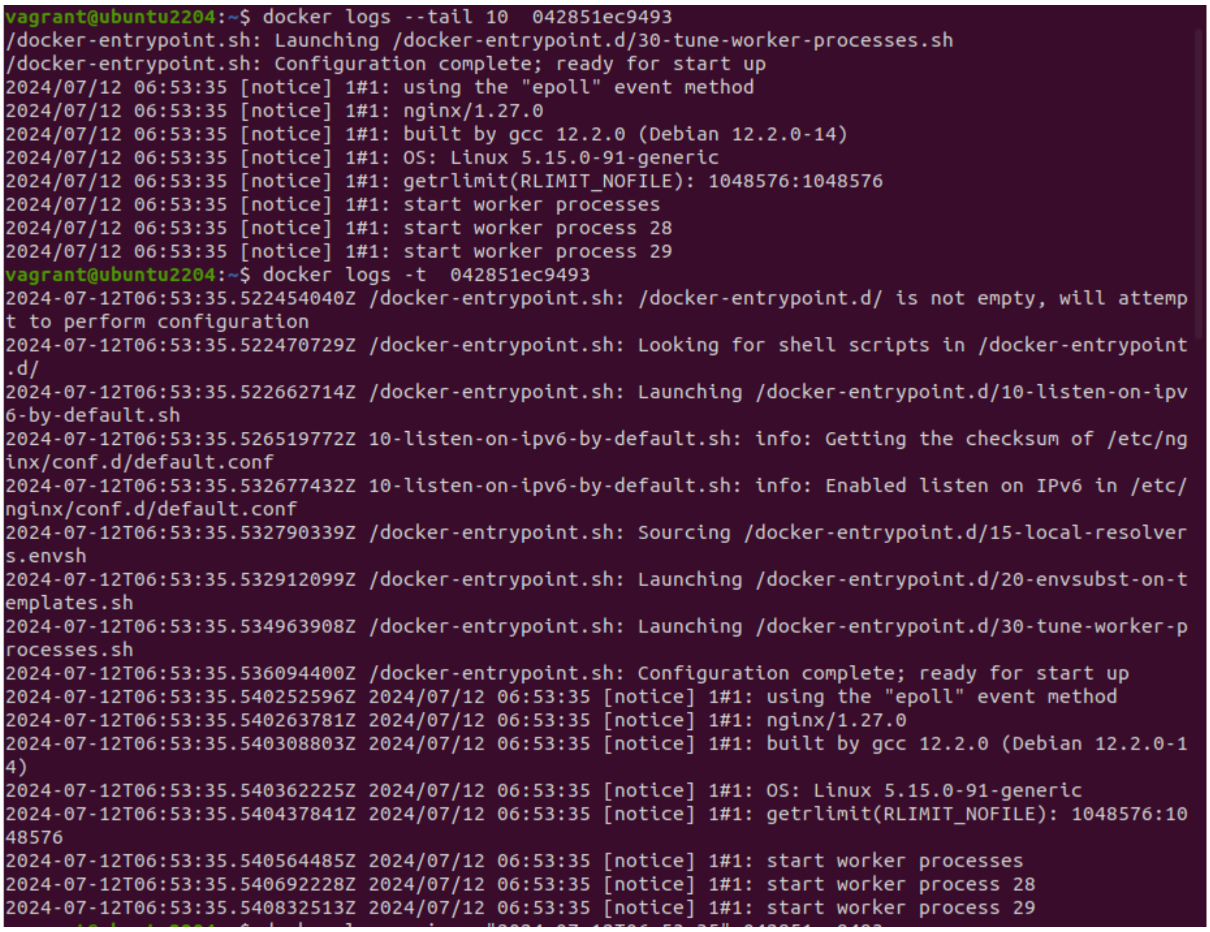
```
docker logs -t my_container
```

5. Showing Logs Since a Specific Time

```
docker logs --since "2023-07-11T15:00:00" my_container
```

6. Combining Options

```
docker logs -f --tail 10 --since "10m" my_container
```



```
vagrant@ubuntu2204:~$ docker logs --tail 10  042851ec9493
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/07/12 06:53:35 [notice] 1#1: using the "epoll" event method
2024/07/12 06:53:35 [notice] 1#1: nginx/1.27.0
2024/07/12 06:53:35 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/07/12 06:53:35 [notice] 1#1: OS: Linux 5.15.0-91-generic
2024/07/12 06:53:35 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/07/12 06:53:35 [notice] 1#1: start worker processes
2024/07/12 06:53:35 [notice] 1#1: start worker process 28
2024/07/12 06:53:35 [notice] 1#1: start worker process 29
vagrant@ubuntu2204:~$ docker logs -t  042851ec9493
2024-07-12T06:53:35.522454040Z /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attemp
t to perform configuration
2024-07-12T06:53:35.522470729Z /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint
.d/
2024-07-12T06:53:35.522662714Z /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv
6-by-default.sh
2024-07-12T06:53:35.526519772Z 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/ng
inx/conf.d/default.conf
2024-07-12T06:53:35.532677432Z 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/
nginx/conf.d/default.conf
2024-07-12T06:53:35.532790339Z /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolver
s.envsh
2024-07-12T06:53:35.532912099Z /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-t
emplates.sh
2024-07-12T06:53:35.534963908Z /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-p
rocesses.sh
2024-07-12T06:53:35.536094400Z /docker-entrypoint.sh: Configuration complete; ready for start up
2024-07-12T06:53:35.540252596Z 2024/07/12 06:53:35 [notice] 1#1: using the "epoll" event method
2024-07-12T06:53:35.540263781Z 2024/07/12 06:53:35 [notice] 1#1: nginx/1.27.0
2024-07-12T06:53:35.540308803Z 2024/07/12 06:53:35 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-1
4)
2024-07-12T06:53:35.540362225Z 2024/07/12 06:53:35 [notice] 1#1: OS: Linux 5.15.0-91-generic
2024-07-12T06:53:35.540437841Z 2024/07/12 06:53:35 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:10
48576
2024-07-12T06:53:35.540564485Z 2024/07/12 06:53:35 [notice] 1#1: start worker processes
2024-07-12T06:53:35.540692228Z 2024/07/12 06:53:35 [notice] 1#1: start worker process 28
2024-07-12T06:53:35.540832513Z 2024/07/12 06:53:35 [notice] 1#1: start worker process 29
```