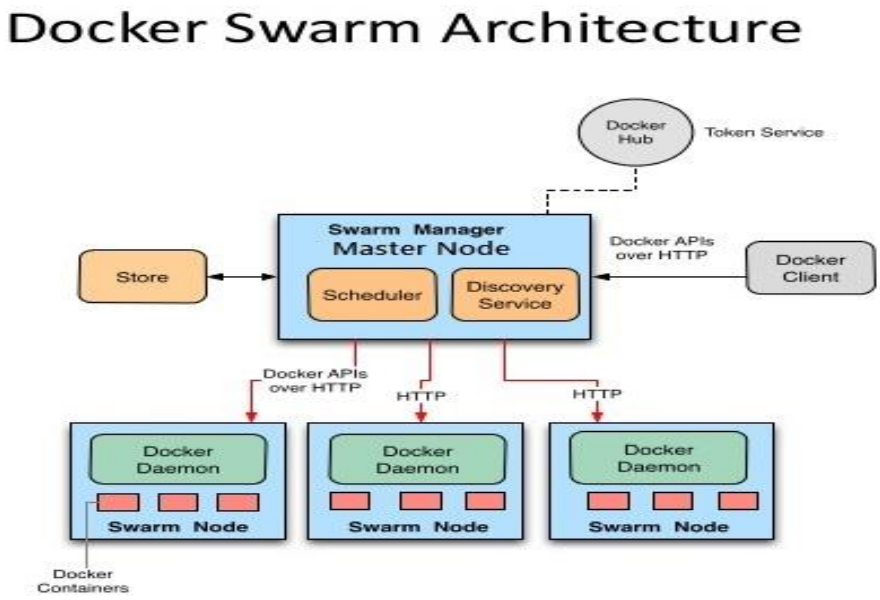
Lab 9 – Docker Swarm

Introduction

Docker Swarm is a clustering and scheduling tool for Docker containers. With Swarm, IT administrators and developers can establish and manage a cluster of Docker nodes as a single virtual system.



1. Create a swarm cluster

1.1 Login as **"root"** user on **aio110** host:

Copy

```
ssh root@aio110
```

1.2 Initialize the swarm or cluster using '**docker swarm init**' command

Run the below command from the manager node(dkmanager) to initialize the cluster.

Syntax:

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

Copy

```
docker swarm init --advertise-addr 10.1.64.134
```

This command will make our node as a manager node and we are also advertising ip address of manager in above command so that slave or worker node can join the cluster.

Sample Output:

```
Swarm initialized: current node (kiejas10cusvnuhit7n8afzy8) is now a
manager.



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



    docker swarm join \

   --token SWMTKN-1-1n7wrakz3m077tdc993hr20brv7ntg6yez6x8sglx8bv82rrig-
cupgbw1aem33fakf7r0cmzgd1 \

   10.1.64.173:2377



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

Note: Take a note of **docker swarm join** command from the above output for the later use!

1.3 Run docker info to view the current state of the swarm:

Copy

```
docker info
```

Output:

```
Containers: 2

Running: 0

Paused: 0

Stopped: 2

  ...snip...

Swarm: active

  NodeID: dxn1zf6l61qsb1josjja83ngz

  Is Manager: true

  Managers: 1

  Nodes: 1

  ...snip...
```

1.4 Run the below command to verify the manager status and to view list of nodes in your cluster:

Copy

```
docker node ls
```

Output:

```
ID                            HOSTNAME                  STATUS
AVAILABILITY        MANAGER STATUS

kiejas10cusvnuhit7n8afzy8 *   aio110   Ready               Active
Leader
```

2. Adding worker node to the existing swarm cluster

Once you've created a swarm with a manager node, you're ready to add worker nodes.

2.1 Open a duplicate session to ssh into the node machine where you want to run a worker node.

Make sure you login as **user0** with the password with `sjc-0901`.

Copy

```
ssh root@aio110
```

2.2 Make sure correct hostname is set:

Copy

```
hostnamectl set-hostname aio110
```

Copy

```
hostname
```

Output:

```
aio110
```

Note: Hostname in the prompt, will change only after **reboot** and we will be reboot the host in a movement.

2.3 Add entries to "***/etc/hosts***" file for local **hostname** resolution.

Copy

```
cat > /etc/hosts <<EOF

172.16.120.12 pod0-master.origin.com

172.16.120.22 pod0-node1.origin.com

10.1.64.1 gw.onecloud   gw
```

```
EOF
```

2.4 Update the system with the latest packages and **reboot** the host to take effect:

Copy

```
yum update -y
```

2.5 Delete the **set_hostname** and **update_hostname** attribute in "**/etc/cloud/cloud.cfg**" file.

Copy

```
sed -i '/set_hostname/d' /etc/cloud/cloud.cfg

sed -i '/update_hostname/d' /etc/cloud/cloud.cfg
```

Note: Above command is used to avoid appending default domain name (".novalocal") to the hostname by cloud_config service as the servers used for this class are actually openstack instances .

2.6 Reboot

Copy

```
reboot
```

Note: Login back as **"root"** user on **aio110**.
`ssh root@aio110`

2.7 Configure **EPEL** repository by installing the below package which provides additional packages for Docker.

EPEL (Extra Packages for Enterprise Linux) is open source and free community based repository project from Fedora team which provides 100% high quality add-on software packages for Linux distribution including **RHEL** (Red Hat Enterprise Linux), **CentOS**, and Scientific **Linux**.

EPEL project is not a part of **RHEL/CentOS** but it is designed for major Linux distributions by providing lots of open source packages like networking, sys admin, programming, monitoring and so on. Most of the epel packages are maintained by Fedora repo.

Copy

```
yum install epel-release -y
```

2.8 Set up docker yum repository:

Copy

```
cat > /etc/yum.repos.d/docker.repo <<EOF

[dockerrepo]

name=Docker Repository

baseurl=https://yum.dockerproject.org/repo/main/centos/7/

enabled=1

gpgcheck=1

gpgkey=https://yum.dockerproject.org/gpg

EOF
```

2.9 Make sure SELinux is disabled:

Security-Enhanced Linux (SELinux) is a **mandatory access control** (**MAC**) security mechanism implemented in the kernel. SELinux was first introduced in **CentOS 4** and significantly enhanced in later CentOS releases.

SELinux has **three basic modes** of operation, of which **Enforcing** is set as the installation default mode.

1. **Enforcing** : The default mode which will enable and enforce the SELinux security policy on the system, denying access and logging actions
2. **Permissive** : In Permissive mode, SELinux is enabled but will not enforce the security policy, only warn and log actions. Permissive mode is useful for troubleshooting SELinux issues.
3. **Disabled** : SELinux is turned off

Note: In our environment “**firewalld**” and “**SELinux**” is **disabled** by default.

Copy

```
sestatus
```

Output:

```
SELinux status:                 disabled
```

2.10: Install **Chrony** and update the **“Chrony” (NTP Server)** configuration to allow connections from our other nodes.

chrony is a versatile implementation of the **Network Time Protocol (NTP)**. It can synchronise the system clock with NTP servers, reference clocks (e.g. GPS receiver), and manual input using wristwatch and keyboard.

Copy

```
yum install chrony -y
```

Copy

```
sed -e '/server/s/^/#/g' -i /etc/chrony.conf

cat >> /etc/chrony.conf <<EOF

server gw.onecloud iburst

Allow 10.1.64.0/24

Allow 10.1.65.0/24

EOF
```

2.11 Next enable, restart and check the status of Chrony service.

Copy

```
systemctl enable chronyd.service

systemctl restart chronyd.service

systemctl status chronyd.service
```

2.12 Run this command to verify the NTP sources:

Copy

```
chronyc sources
```

Sample output:

```
210 Number of sources = 1

MS Name/IP address         Stratum Poll Reach LastRx Last sample

===============================================================================
=========

^? gw.onecloud                   3   6     3     0  -1291us[-1291us]
+/-   90ms
```

2.13 To check current kernel version, run the below command :

Docker requires a **64-bit** installation regardless of user CentOS version. Also, kernel must be **3.10** at minimum, which CentOS 7 runs.

Copy

```
uname -r
```

Output:

```
3.10.0-693.17.1.el7.x86_64
```

Finally, it is recommended that fully update system. Please keep in mind that system should be fully patched to fix any potential kernel bugs. Any reported kernel bugs may have already been fixed on the latest kernel packages.

3. Install Docker

3.1 Install the Docker package.

Copy

```
yum install docker-engine -y
```

Once docker is installed, will need to start the docker daemon.

3.2 Run the below command to start the docker deamon on boot.

Copy

```
systemctl enable docker
```

Output:

```
Created symlink from /etc/systemd/system/multi-
user.target.wants/docker.service to
/usr/lib/systemd/system/docker.service.
```

3.3 Start the Docker daemon.

Copy

```
systemctl start docker
```

3.4 Verify the Status of the Docker.

Copy

```
systemctl status docker
```

Output:

```
● docker.service - Docker Application Container Engine

   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled;
vendor preset: disabled)

   Active: active (running) since Sat 2018-02-17 11:40:44 UTC; 2s ago

     Docs: https://docs.docker.com
```

```
 Main PID: 10508 (dockerd)

   Memory: 14.4M

   CGroup: /system.slice/docker.service

           ├─10508 /usr/bin/dockerd

           └─10514 docker-containerd -l
unix:///var/run/docker/libcontaine...



Feb 17 11:40:41 aio110 systemd[1]: Starting Docker Application
Container.....

Feb 17 11:40:41 aio110 dockerd[10508]: time="2018-02-
17T11:40:41.92854899..."

....

....

Feb 17 11:40:44 aio110 dockerd[10508]: time="2018-02-
17T11:40:44.00961340..."

Hint: Some lines were ellipsized, use -l to show in full.
```

3.5 To add Worker nodes to the swarm or cluster run the command that we get when we initialize the swarm.

Note: Use the command, which we had copied from the step .

Copy

```
docker swarm join \

  --token SWMTKN-1-49nj1cmql0jkz5s954yi3oex3nedyz0fb0xx14ie39trti4wxv-
8vxv8rssmk743ojnwacrr2e7c \

  10.1.64.134:2377
```

Output:

```
This node joined a swarm as a worker.
```

3.6 Go back to a master(**aio110**) host terminal and run the **docker node ls** command to see the **worker nodes**:

Verify the node status from docker manager.

Copy

```
docker node ls
```

Output:

```
ID                            HOSTNAME                  STATUS
AVAILABILITY        MANAGER STATUS

huf0wjif8m7b0qelrond2gd7r     aio110     Ready               Active

kiejas10cusvnuhit7n8afzy8 *   aio110   Ready               Active
Leader
```

At this point of time our docker swarm mode or cluster is up and running with two worker nodes. In the next step we will see how to define a service.

4. Launching service in Docker Swarm mode

In Docker swarm mode containers are replaced with word tasks and tasks (or containers) are launched and deployed as service and Let's assume I want to create a service with the name "webserver" with five containers and want to make sure desired state of containers inside the service is five.

Note: Continue below commands on master host **(aio110)**.

4.1 Run below commands from Docker Manager only.

Copy

```
docker service create -p 80:80 --name webserver --replicas 5 httpd
```

Sample Output:

```
7hqezhyak8jbt8idkkke8wizi

Since --detach=false was not specified, tasks will be created in the
background.

In a future release, --detach=false will become the default.
```

Above command will create a service with name “webserver”, in which desired state of containers or task is 5 and containers will be launched from docker image “httpd”. Containers will be deployed over the cluster nodes i.e dkmanager, workernode .

4.2 List the Docker service with below command

Copy

```
 docker service ls
```

Output:

```
ID                  NAME                MODE                REPLICAS
IMAGE               PORTS

j9sa2jrmp86h        webserver           replicated          0/5
httpd:latest        *:80->80/tcp
```

4.3 Execute the below command to view status of your service “webserver”

Copy

```
docker service ps webserver
```

Output:

```
ID                  NAME                IMAGE               NODE
DESIRED STATE       CURRENT STATE                     ERROR
PORTS
```

```
y46gfpdgbdc1        webserver.1         httpd:latest        aio110
Running             Preparing less than a second ago

w2ba0c0htp98        webserver.2         httpd:latest        aio110
Running             Preparing less than a second ago

lf3bpei7czzz        webserver.3         httpd:latest        aio110
Running             Preparing 16 seconds ago

xkw8djiji8ln        webserver.4         httpd:latest        aio110
Running             Preparing less than a second ago

izq3gwh1aa50        webserver.5         httpd:latest        aio110
Running             Preparing 16 seconds ago
```

4.4 As per above output we can see containers are deployed across the cluster nodes including manager node. Now we can access web page from any of worker node and Docker Manager using the following URLs :

Copy

```
curl http://aio110
```

Output:

```
<html>

<body>

<h1>It works!</h1>

</body>

</html>
```

Copy

```
curl http://aio110
```

Output:

```
<html>

<body>

<h1>It works!</h1>

</body>

</html>
```