Docker-Kubernetes-Foundations
Linux Namespaces

Exercise-1

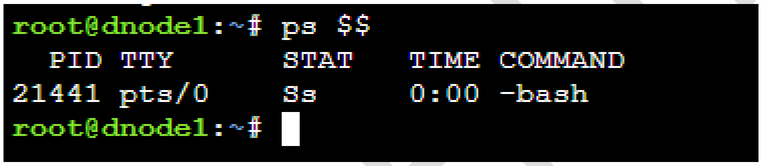
PID Namespaces

NOTE: *Perform the below steps on the dnode1 node*

1. PID of current bash process

- Login to a new terminal in **dnode1** node and verify the PID of current bash process in global NS

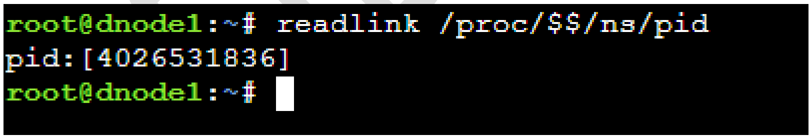
```
ps $$
```



```
root@dnode1:~# ps $$
  PID TTY      STAT   TIME COMMAND
21441 pts/0    Ss     0:00 -bash
root@dnode1:~#
```

- Read the PID namespace of the bash process

```
readlink /proc/$$/ns/pid
```



```
root@dnode1:~# readlink /proc/$$/ns/pid
pid:[4026531836]
root@dnode1:~#
```

2. New PID namespace for a bash process

- Enter the superuser mode

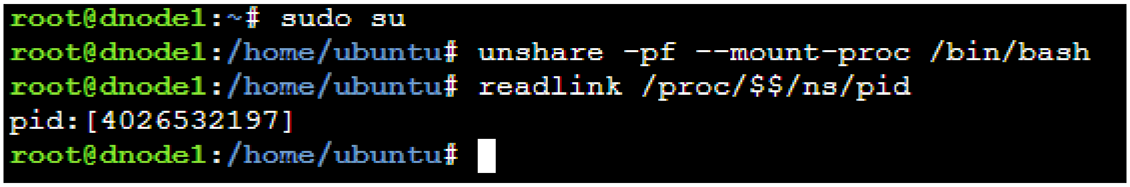
```
sudo su
```

- Create a new PID namespace for a bash process

```
unshare -pf --mount-proc /bin/bash
```

- Read the PID namespace of the bash process

```
readlink /proc/$$/ns/pid
```

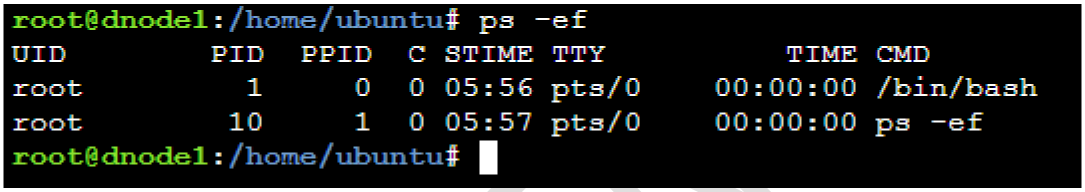


```
root@dnode1:~# sudo su
root@dnode1:/home/ubuntu# unshare -pf --mount-proc /bin/bash
root@dnode1:/home/ubuntu# readlink /proc/$$/ns/pid
pid:[4026532197]
root@dnode1:/home/ubuntu#
```

3. Running processes

- Verify the current running processes in new PID NS

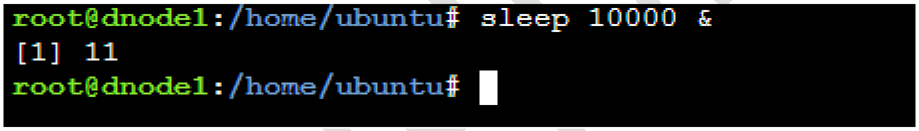
```
ps -ef
```



```
root@dnode1:/home/ubuntu# ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 05:56 pts/0    00:00:00 /bin/bash
root          10     1  0 05:57 pts/0    00:00:00 ps -ef
root@dnode1:/home/ubuntu#
```

- Create new process in this new bash

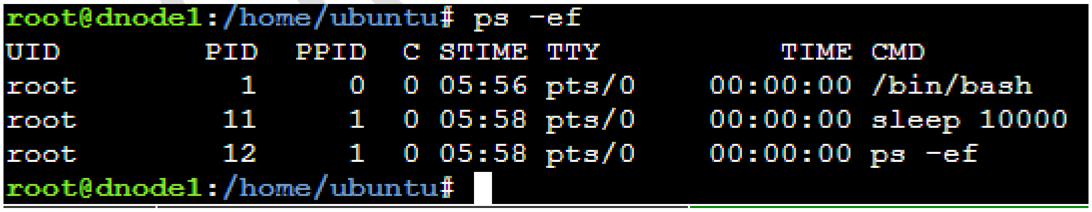
```
sleep 10000 &
```



```
root@dnode1:/home/ubuntu# sleep 10000 &
[1] 11
root@dnode1:/home/ubuntu#
```

- Verify the current running processes in new PID NS

```
ps -ef
```

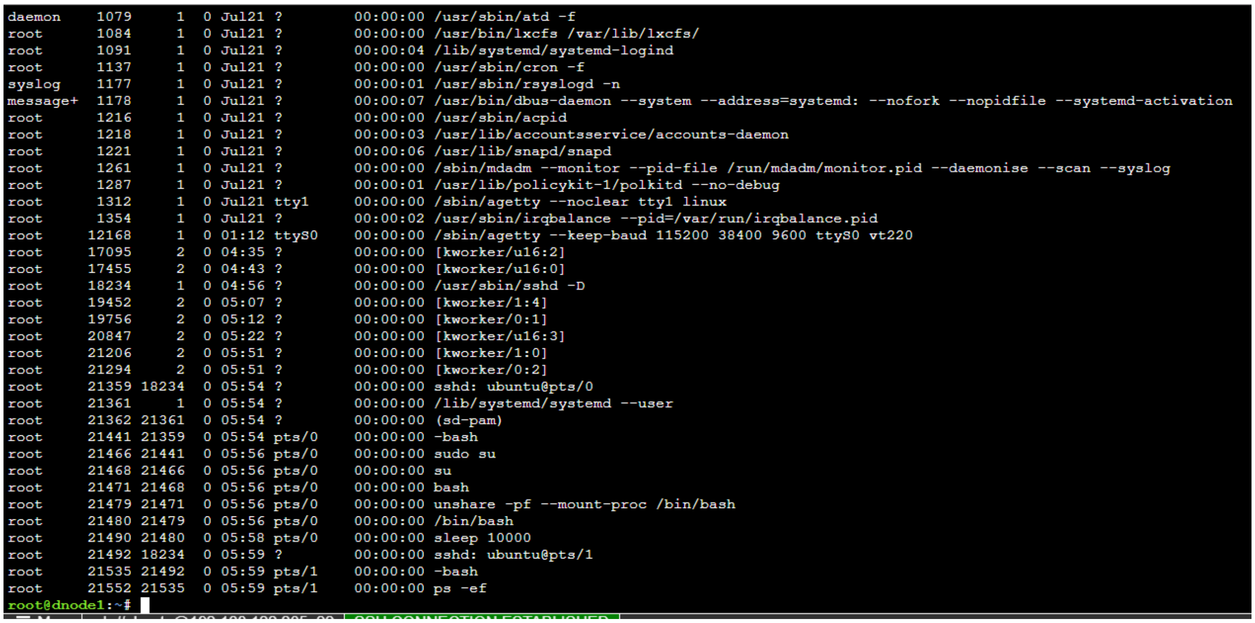


```
root@dnode1:/home/ubuntu# ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 05:56 pts/0    00:00:00 /bin/bash
root          11     1  0 05:58 pts/0    00:00:00 sleep 10000
root          12     1  0 05:58 pts/0    00:00:00 ps -ef
root@dnode1:/home/ubuntu#
```

Observe the Parent and Child Process ID in new PID NS

- Open another SSH session for **dnode1**, verify the current running processes in global PID NS

```
ps -ef
```



```
daemon    1079     1  0 Jul21 ?        00:00:00 /usr/sbin/atd -f
root      1084     1  0 Jul21 ?        00:00:00 /usr/bin/lxcfs /var/lib/lxcfs/
root      1091     1  0 Jul21 ?        00:00:04 /lib/systemd/systemd-logind
root      1137     1  0 Jul21 ?        00:00:00 /usr/sbin/cron -f
syslog    1177     1  0 Jul21 ?        00:00:01 /usr/sbin/rsyslogd -n
message+  1178     1  0 Jul21 ?        00:00:07 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
root      1216     1  0 Jul21 ?        00:00:00 /usr/sbin/acpid
root      1218     1  0 Jul21 ?        00:00:03 /usr/lib/accountsservice/accounts-daemon
root      1221     1  0 Jul21 ?        00:00:06 /usr/lib/snapd/snapd
root      1261     1  0 Jul21 ?        00:00:00 /sbin/mdadm --monitor --pid-file /run/mdadm/monitor.pid --daemonise --scan --syslog
root      1287     1  0 Jul21 ?        00:00:01 /usr/lib/policykit-1/polkitd --no-debug
root      1312     1  0 Jul21 tty1     00:00:00 /sbin/agetty --noclear tty1 linux
root      1354     1  0 Jul21 ?        00:00:02 /usr/sbin/irqbalance --pid=/var/run/irqbalance.pid
root     12168     1  0 01:12 ttyS0    00:00:00 /sbin/agetty --keep-baud 115200 38400 9600 ttyS0 vt220
root     17095     2  0 04:35 ?        00:00:00 [kworker/u16:2]
root     17455     2  0 04:43 ?        00:00:00 [kworker/u16:0]
root     18234     1  0 04:56 ?        00:00:00 /usr/sbin/sshd -D
root     19452     2  0 05:07 ?        00:00:00 [kworker/1:4]
root     19756     2  0 05:12 ?        00:00:00 [kworker/0:1]
root     20847     2  0 05:22 ?        00:00:00 [kworker/u16:3]
root     21206     2  0 05:51 ?        00:00:00 [kworker/1:0]
root     21294     2  0 05:51 ?        00:00:00 [kworker/0:2]
root     21359 18234  0 05:54 ?        00:00:00 sshd: ubuntu@pts/0
root     21361     1  0 05:54 ?        00:00:00 /lib/systemd/systemd --user
root     21362 21361  0 05:54 ?        00:00:00 (sd-pam)
root     21441 21359  0 05:54 pts/0    00:00:00 -bash
root     21466 21441  0 05:56 pts/0    00:00:00 sudo su
root     21468 21466  0 05:56 pts/0    00:00:00 su
root     21471 21468  0 05:56 pts/0    00:00:00 bash
root     21479 21471  0 05:56 pts/0    00:00:00 unshare -pf --mount-proc /bin/bash
root     21480 21479  0 05:56 pts/0    00:00:00 /bin/bash
root     21490 21480  0 05:58 pts/0    00:00:00 sleep 10000
root     21492 18234  0 05:59 ?        00:00:00 sshd: ubuntu@pts/1
root     21535 21492  0 05:59 pts/1    00:00:00 -bash
root     21552 21535  0 05:59 pts/1    00:00:00 ps -ef
root@dnode1:~#
```

Observe the Process ID for bash and sleep process in global and new PID NS

Exercise-2

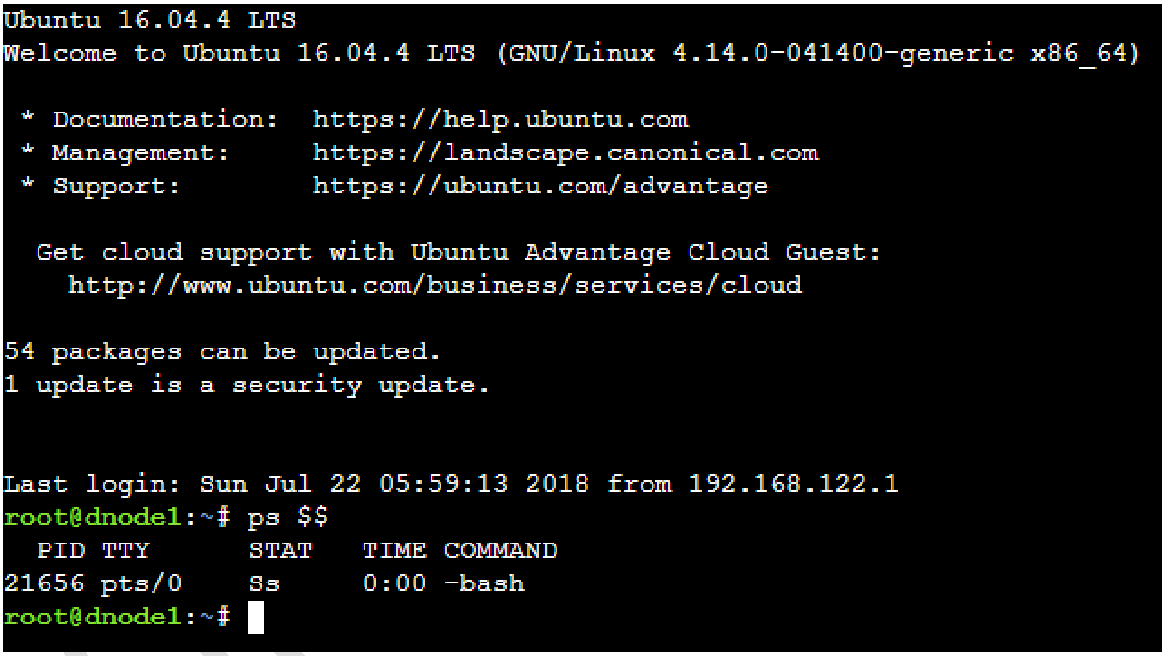
UTS namespace

NOTE: *Perform the below steps on the dnode1 node*

1.PID of current bash process

- Login to a new terminal in **dnode1** node and verify the PID of current bash process in global NS

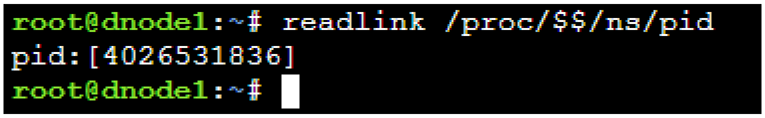
```
ps $$
```



```
Ubuntu 16.04.4 LTS
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.14.0-041400-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

54 packages can be updated.
1 update is a security update.


Last login: Sun Jul 22 05:59:13 2018 from 192.168.122.1
root@dnode1:~# ps $$
  PID TTY      STAT   TIME COMMAND
21656 pts/0    Ss     0:00 -bash
root@dnode1:~# 
```

- Read the PID namespace of the bash process

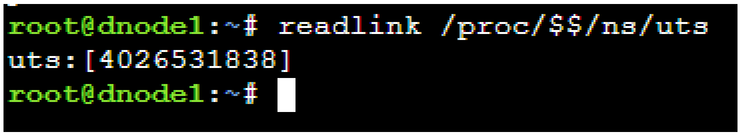
```
readlink /proc/$$/ns/pid
```



```
root@dnode1:~# readlink /proc/$$/ns/pid
pid:[4026531836]
root@dnode1:~# 
```

- Read the UTS namespace of the bash process

```
readlink /proc/$$/ns/uts
```



```
root@dnode1:~# readlink /proc/$$/ns/uts
uts:[4026531838]
root@dnode1:~#
```

- Verify the current hostname of the device

```
hostname
```

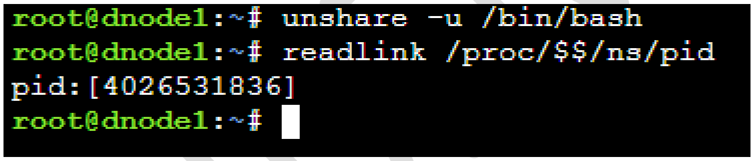
2. New UTS namespace for a bash process

- Create a new UTS namespace for a bash process

```
unshare -u /bin/bash
```

- Read the PID namespace of the bash process

```
readlink /proc/$$/ns/pid
```



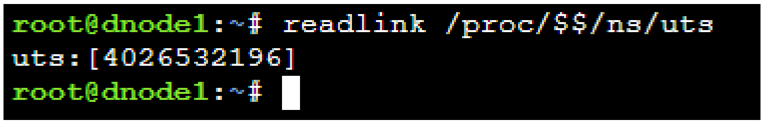
```
root@dnode1:~# unshare -u /bin/bash
root@dnode1:~# readlink /proc/$$/ns/pid
pid:[4026531836]
root@dnode1:~#
```

- Verify the new bash process PID

```
ps $$
```

- Read the UTS namespace of the bash process

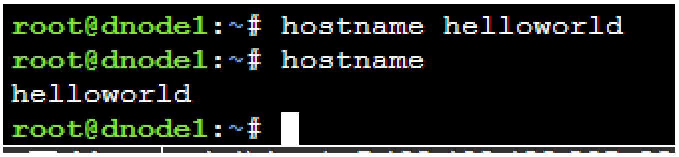
```
readlink /proc/$$/ns/uts
```



```
root@dnode1:~# readlink /proc/$$/ns/uts
uts:[4026532196]
root@dnode1:~#
```

- Verify that UTS inode is different but PID inode is same when compared to global namespace
- Change the hostname in this new UTS shell

```
hostname helloworld
```



```
root@dnode1:~# hostname helloworld
root@dnode1:~# hostname
helloworld
root@dnode1:~#
```

- Display the changed hostname

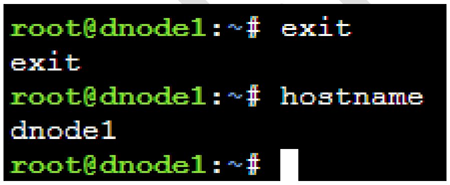
```
hostname
```

- Exit the UTS namespace

```
exit
```

- Verify the hostname of the device is unchanged

```
hostname
```



```
root@dnode1:~# exit
exit
root@dnode1:~# hostname
dnode1
root@dnode1:~#
```

Exercise-3

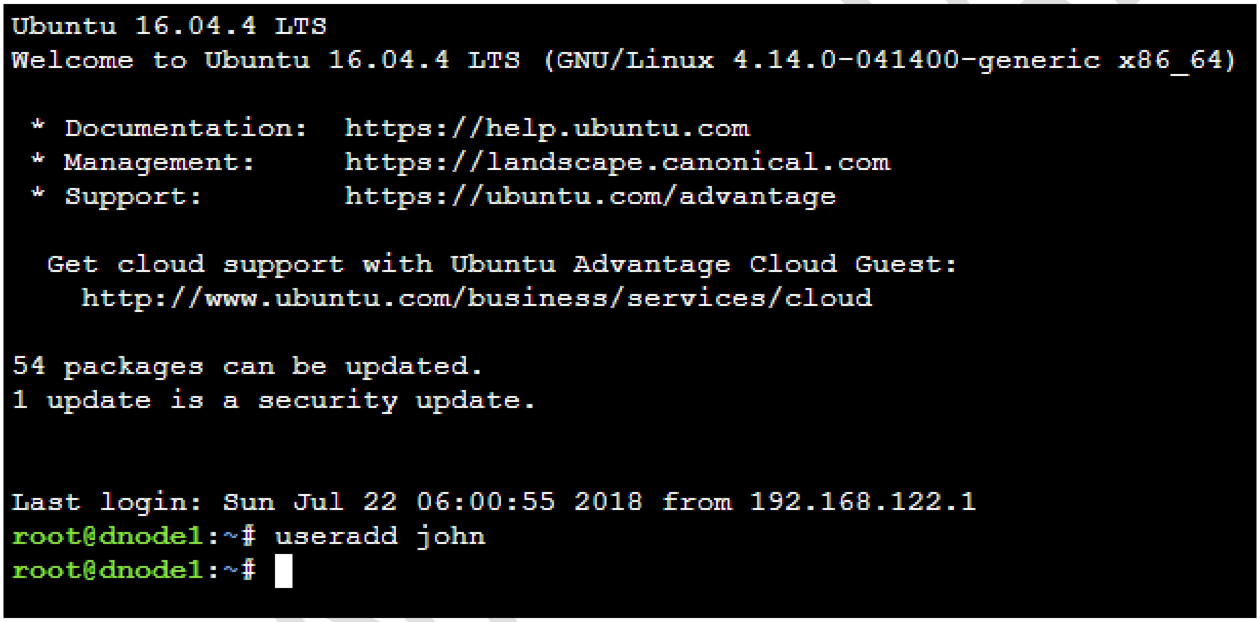
User Namespace

NOTE: *Perform the below steps on the dnode1 node*

Create a new user

- Login to a new terminal in dnode1 node and create a new user named john

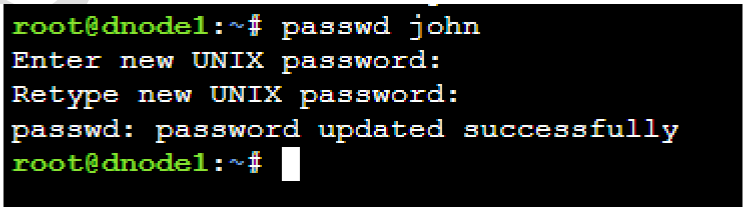
```
useradd john
```



```
Ubuntu 16.04.4 LTS
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.14.0-041400-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

54 packages can be updated.
1 update is a security update.


Last login: Sun Jul 22 06:00:55 2018 from 192.168.122.1
root@dnode1:~# useradd john
root@dnode1:~# 
```

- Assign Password to john

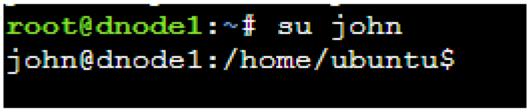
```
passwd john
```



```
root@dnode1:~# passwd john
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@dnode1:~# 
```

- Switch user to john

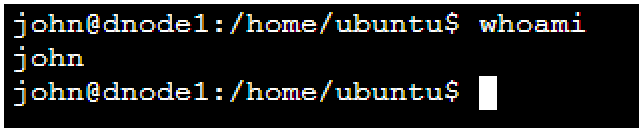
```
su john
```



```
root@dnode1:~# su john
john@dnode1:/home/ubuntu$
```

- Verify the user for current process

```
whoami
```

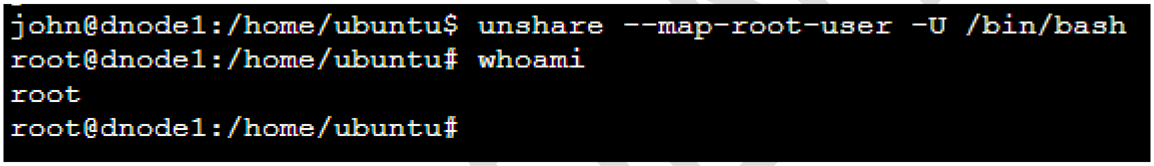


```
john@dnode1:/home/ubuntu$ whoami
john
john@dnode1:/home/ubuntu$
```

Create new User namespace

- Create new User namespace

```
unshare --map-root-user -U /bin/bash
```



```
john@dnode1:/home/ubuntu$ unshare --map-root-user -U /bin/bash
root@dnode1:/home/ubuntu# whoami
root
root@dnode1:/home/ubuntu#
```

- Verify the user for current process

```
whoami
```

Verify the output is root

Exercise-4

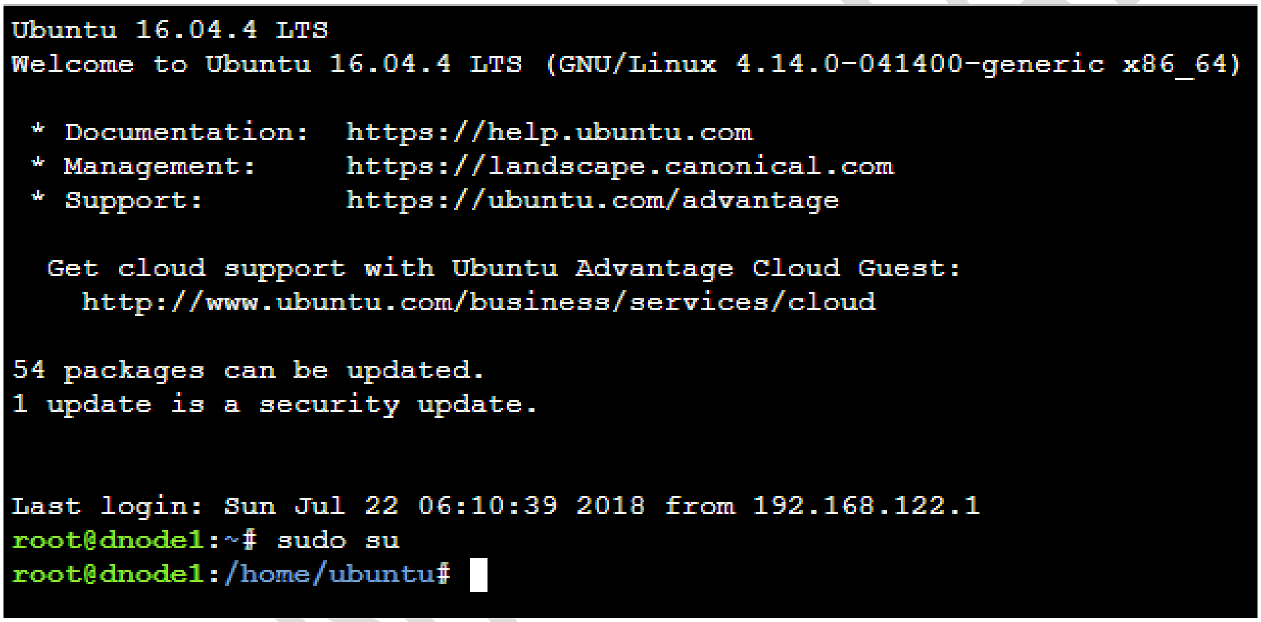
Mount Namespaces

NOTE: *Perform the below steps on the dnode1 node*

Create a new mount space, mount point and verify them

- Enter the superuser mode

```
sudo su
```



```
Ubuntu 16.04.4 LTS
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.14.0-041400-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

54 packages can be updated.
1 update is a security update.


Last login: Sun Jul 22 06:10:39 2018 from 192.168.122.1
root@dnode1:~# sudo su
root@dnode1:/home/ubuntu#
```

- Login to a new terminal in dnode1 node and create new mount namespace

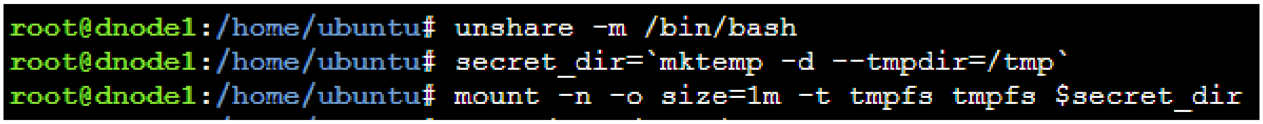
```
unshare -m /bin/bash
```

- Create a new directory. Please observe that a new directory is created under /tmp with random ID.

```
secret_dir=`mktemp -d --tmpdir=/tmp`
```

- Creating a new mount point for above created directory

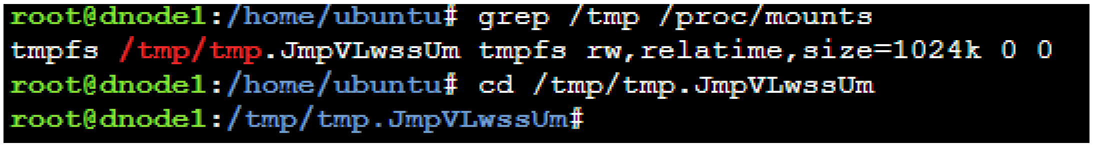
```
mount -n -o size=1m -t tmpfs tmpfs $secret_dir
```



```
root@dnode1:/home/ubuntu# unshare -m /bin/bash
root@dnode1:/home/ubuntu# secret_dir=`mktemp -d --tmpdir=/tmp`
root@dnode1:/home/ubuntu# mount -n -o size=1m -t tmpfs tmpfs $secret_dir
```

- Check the available mount points

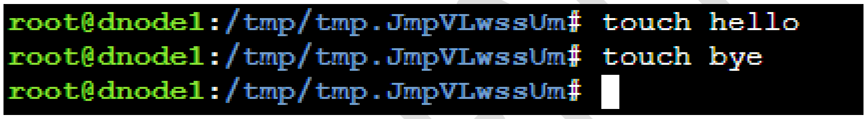
```
grep /tmp /proc/mounts
```



```
root@dnode1:/home/ubuntu# grep /tmp /proc/mounts
tmpfs /tmp/tmp.JmpVLwssUm tmpfs rw,relatime,size=1024k 0 0
root@dnode1:/home/ubuntu# cd /tmp/tmp.JmpVLwssUm
root@dnode1:/tmp/tmp.JmpVLwssUm#
```

- Change to the new directory (replace tmpID with actual directory name created above) and create 2 files in new directory

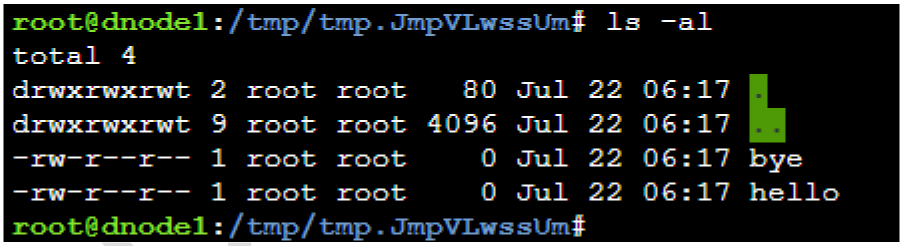
```
cd /tmp/tmpID

touch hello

touch bye
```



```
root@dnode1:/tmp/tmp.JmpVLwssUm# touch hello
root@dnode1:/tmp/tmp.JmpVLwssUm# touch bye
root@dnode1:/tmp/tmp.JmpVLwssUm#
```

- List the files created

```
ls -al
```



```
root@dnode1:/tmp/tmp.JmpVLwssUm# ls -al
total 4
drwxrwxrwt 2 root root   80 Jul 22 06:17 .
drwxrwxrwt 9 root root 4096 Jul 22 06:17 ..
-rw-r--r-- 1 root root    0 Jul 22 06:17 bye
-rw-r--r-- 1 root root    0 Jul 22 06:17 hello
root@dnode1:/tmp/tmp.JmpVLwssUm#
```

- Open another session to dnode1 and change directory within /tmp to newly created directory
- List the files

```
ls -al
```

Verify we cannot see any files that were created earlier

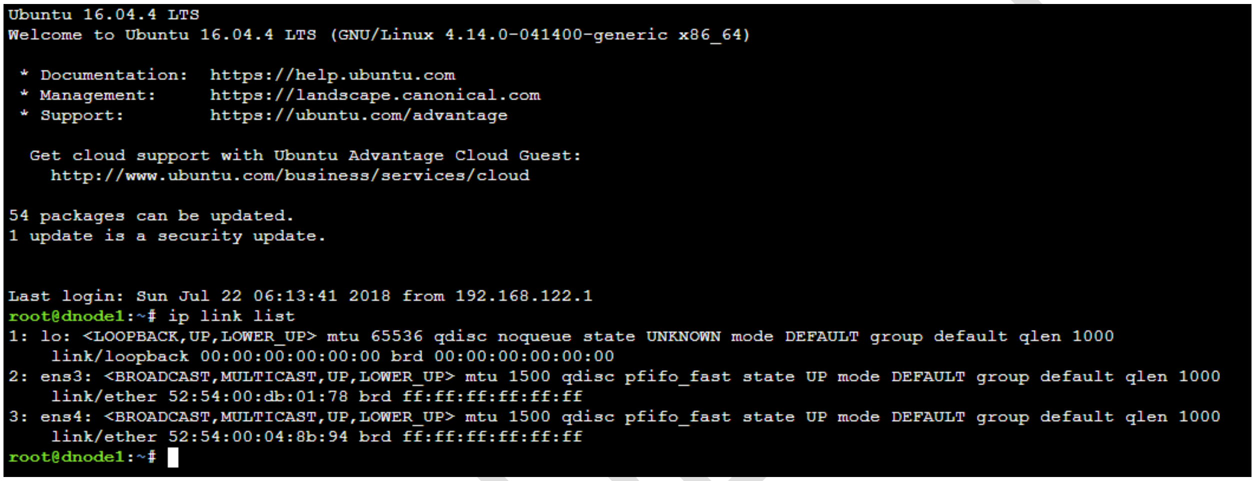
Exercise-5

Net Namespaces

NOTE: *Perform the below steps on the dnode1 node*

- Login to a new terminal in dnode1 node and verify the current network interfaces

```
ip link list
```



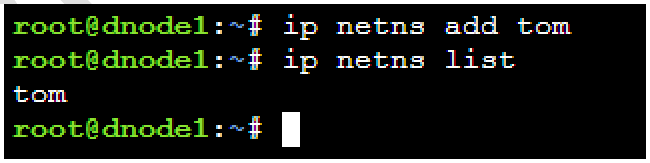
```
Ubuntu 16.04.4 LTS
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.14.0-041400-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

54 packages can be updated.
1 update is a security update.


Last login: Sun Jul 22 06:13:41 2018 from 192.168.122.1
root@dnode1:~# ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 52:54:00:db:01:78 brd ff:ff:ff:ff:ff:ff
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 52:54:00:04:8b:94 brd ff:ff:ff:ff:ff:ff
root@dnode1:~#
```

- Create a new net namespace

```
ip netns add tom
```

- Verify the namespace created

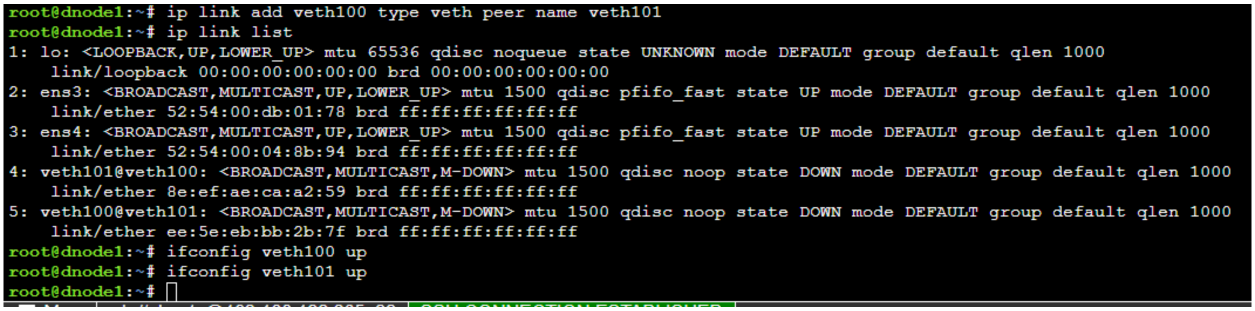
```
ip netns list
```



```
root@dnode1:~# ip netns add tom
root@dnode1:~# ip netns list
tom
root@dnode1:~#
```

- Create a virtual Ethernet pair

```
ip link add veth100 type veth peer name veth101

ip link list

ifconfig veth100 up

ifconfig veth101 up
```



```
root@dnode1:~# ip link add veth100 type veth peer name veth101
root@dnode1:~# ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 52:54:00:db:01:78 brd ff:ff:ff:ff:ff:ff
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 52:54:00:04:8b:94 brd ff:ff:ff:ff:ff:ff
4: veth101@veth100: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 8e:ef:ae:ca:a2:59 brd ff:ff:ff:ff:ff:ff
5: veth100@veth101: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether ee:5e:eb:bb:2b:7f brd ff:ff:ff:ff:ff:ff
root@dnode1:~# ifconfig veth100 up
root@dnode1:~# ifconfig veth101 up
root@dnode1:~# 
```

- Move the veth101 interface into tom namespace

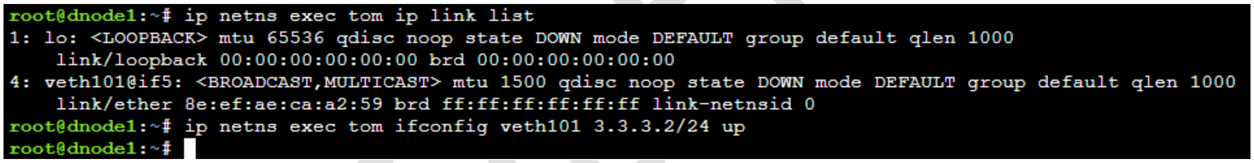
```
ip link set veth101 netns tom
```

- Verify the interfaces in tom namespace

```
ip netns exec tom ip link list
```

- Assign IP address to veth101 and make it up

```
ip netns exec tom ifconfig veth101 3.3.3.2/24 up
```



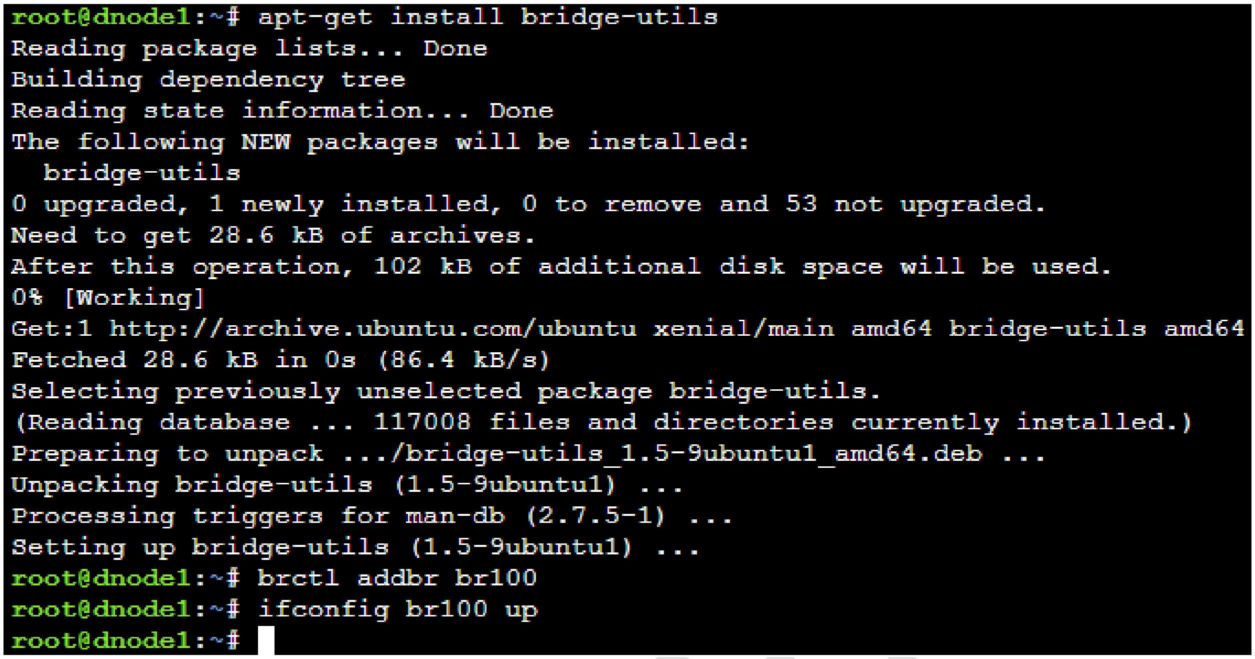
```
root@dnode1:~# ip netns exec tom ip link list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
4: veth101@if5: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 8e:ef:ae:ca:a2:59 brd ff:ff:ff:ff:ff:ff link-netnsid 0
root@dnode1:~# ip netns exec tom ifconfig veth101 3.3.3.2/24 up
root@dnode1:~# 
```

- Install bridge-utils package

```
apt-get install bridge-utils
```

- Create a new linux bridge br100 and make it up

```
brctl addbr br100

ifconfig br100 up
```



```
root@dnode1:~# apt-get install bridge-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  bridge-utils
0 upgraded, 1 newly installed, 0 to remove and 53 not upgraded.
Need to get 28.6 kB of archives.
After this operation, 102 kB of additional disk space will be used.
0% [Working]
Get:1 http://archive.ubuntu.com/ubuntu xenial/main amd64 bridge-utils amd64
Fetched 28.6 kB in 0s (86.4 kB/s)
Selecting previously unselected package bridge-utils.
(Reading database ... 117008 files and directories currently installed.)
Preparing to unpack .../bridge-utils_1.5-9ubuntu1_amd64.deb ...
Unpacking bridge-utils (1.5-9ubuntu1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up bridge-utils (1.5-9ubuntu1) ...
root@dnode1:~# brctl addbr br100
root@dnode1:~# ifconfig br100 up
root@dnode1:~#
```

- Attach veth100 interface to bridge br100

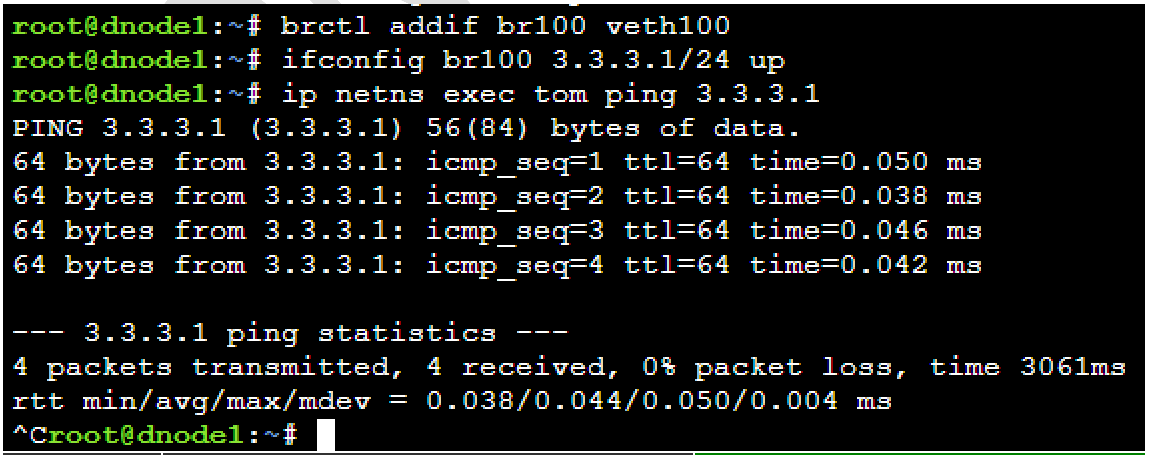
```
brctl addif br100 veth100
```

- Assign IP address to interface br100 in global namespace

```
ifconfig br100 3.3.3.1/24 up
```

- Verify the network connectivity from tom and global namespace

```
ip netns exec tom ping 3.3.3.1
```



```
root@dnode1:~# brctl addif br100 veth100
root@dnode1:~# ifconfig br100 3.3.3.1/24 up
root@dnode1:~# ip netns exec tom ping 3.3.3.1
PING 3.3.3.1 (3.3.3.1) 56(84) bytes of data.
64 bytes from 3.3.3.1: icmp_seq=1 ttl=64 time=0.050 ms
64 bytes from 3.3.3.1: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from 3.3.3.1: icmp_seq=3 ttl=64 time=0.046 ms
64 bytes from 3.3.3.1: icmp_seq=4 ttl=64 time=0.042 ms

--- 3.3.3.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3061ms
rtt min/avg/max/mdev = 0.038/0.044/0.050/0.004 ms
^Croot@dnode1:~#
```

Exercise-6

Multiple Namespaces (PID and UTS)

NOTE: *Perform the below steps on dnode1*

- Enter the superuser mode

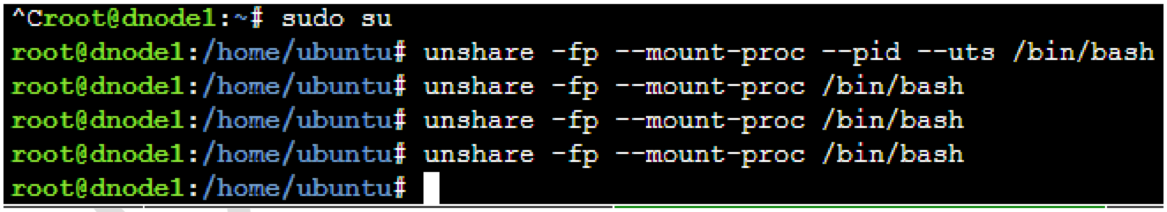
```
sudo su
```

- Create bash process with multiple namespaces

```
unshare -fp --mount-proc --pid --uts /bin/bash
```

- Verify the PID and UTS namespaces associated with this process ((Refer to previous exercises for commands)
- Exit from this namespace
- Let us try namespace within namespace for PID

```
unshare -fp --mount-proc /bin/bash

unshare -fp --mount-proc /bin/bash

unshare -fp --mount-proc /bin/bash
```



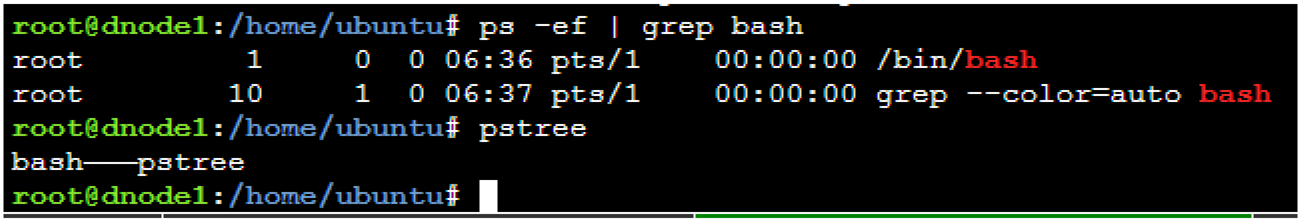
```
^Croot@dnode1:~# sudo su
root@dnode1:/home/ubuntu# unshare -fp --mount-proc --pid --uts /bin/bash
root@dnode1:/home/ubuntu# unshare -fp --mount-proc /bin/bash
root@dnode1:/home/ubuntu# unshare -fp --mount-proc /bin/bash
root@dnode1:/home/ubuntu# unshare -fp --mount-proc /bin/bash
root@dnode1:/home/ubuntu# 
```

- Verify the total processes created in global namespace

```
ps -ef | grep bash
```

- Observe child parent-child relationship among 3 levels of namespaces within them

```
pstree
```



```
root@dnode1:/home/ubuntu# ps -ef | grep bash
root         1      0  0 06:36 pts/1    00:00:00 /bin/bash
root        10      1  0 06:37 pts/1    00:00:00 grep --color=auto bash
root@dnode1:/home/ubuntu# pstree
bash───pstree
root@dnode1:/home/ubuntu#
```

Linux cgroups

Exercise-7

Explore command to limit cpu usage of process

NOTE: *Perform the below exercises on "dnode1" through SSH and also on "desktop1" through VNC*

- Install prime number generator for generating huge cpu usage

```
apt-get update

apt install libmath-prime-util-gmp-perl
```

- Verify the primes command between 1 100

```
primes 1 100
```

- Generate prime number for large set and redirect to null

```
primes 0 9999999999 > /dev/null &
```

- Open VNC session to the "desktop1" machine. Please check Cluster Access -> VNC Access to get more details.
- Open a terminal within VNC desktop1 machine. It will be located at the top of the VNC session.
- Get the ens3 interface IP of the **dnode1**

```
ifconfig ens3
```

- SSH from the terminal opened in VNC session.

```
ssh ubuntu@192.168.122.205
```

Password: ubuntu

- On "dnode1" terminal (opened in VNC terminal), and switch to root user:

```
sudo su
```

- Issue top command to see the CPU usage of processes

```
top
```

- Verify that perl/primes command is consuming ~100% CPU
- Keep the VNC session to the "desktop1" machine running. We will use it for later exercises.
- On dnode1, bring the process into foreground

```
fg
```

(and then press ctrl +c)

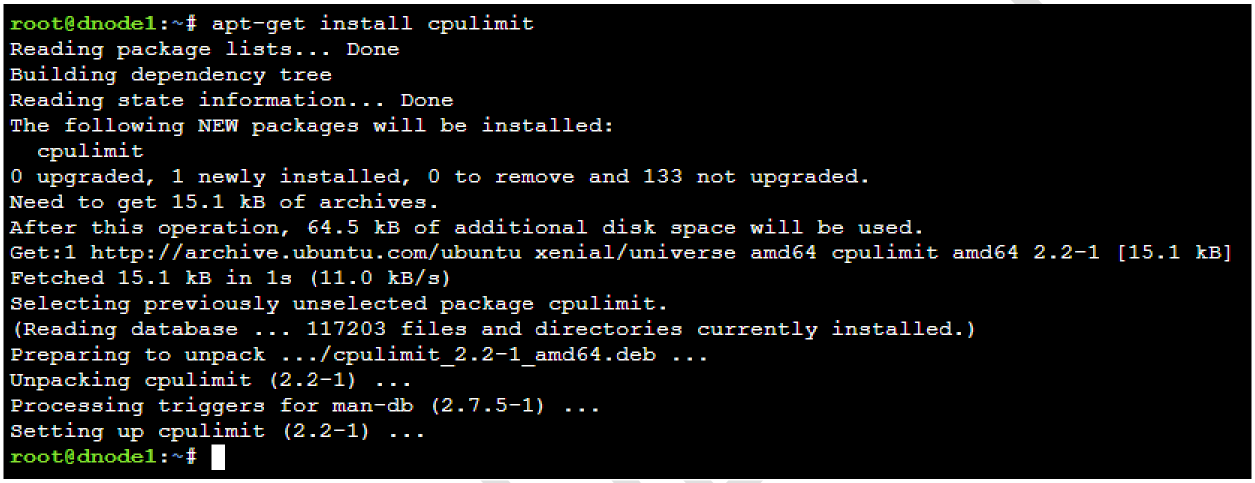
Exercise-8

Use cpulimit command to restrict the CPU usage of a process

NOTE: *Perform below exercise on dnode1*

- Install cpulimit command

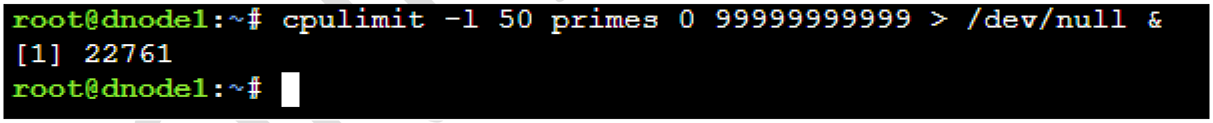
```
apt-get install cpulimit
```



```
root@dnode1:~# apt-get install cpulimit
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  cpulimit
0 upgraded, 1 newly installed, 0 to remove and 133 not upgraded.
Need to get 15.1 kB of archives.
After this operation, 64.5 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu xenial/universe amd64 cpulimit amd64 2.2-1 [15.1 kB]
Fetched 15.1 kB in 1s (11.0 kB/s)
Selecting previously unselected package cpulimit.
(Reading database ... 117203 files and directories currently installed.)
Preparing to unpack .../cpulimit_2.2-1_amd64.deb ...
Unpacking cpulimit (2.2-1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up cpulimit (2.2-1) ...
root@dnode1:~#
```

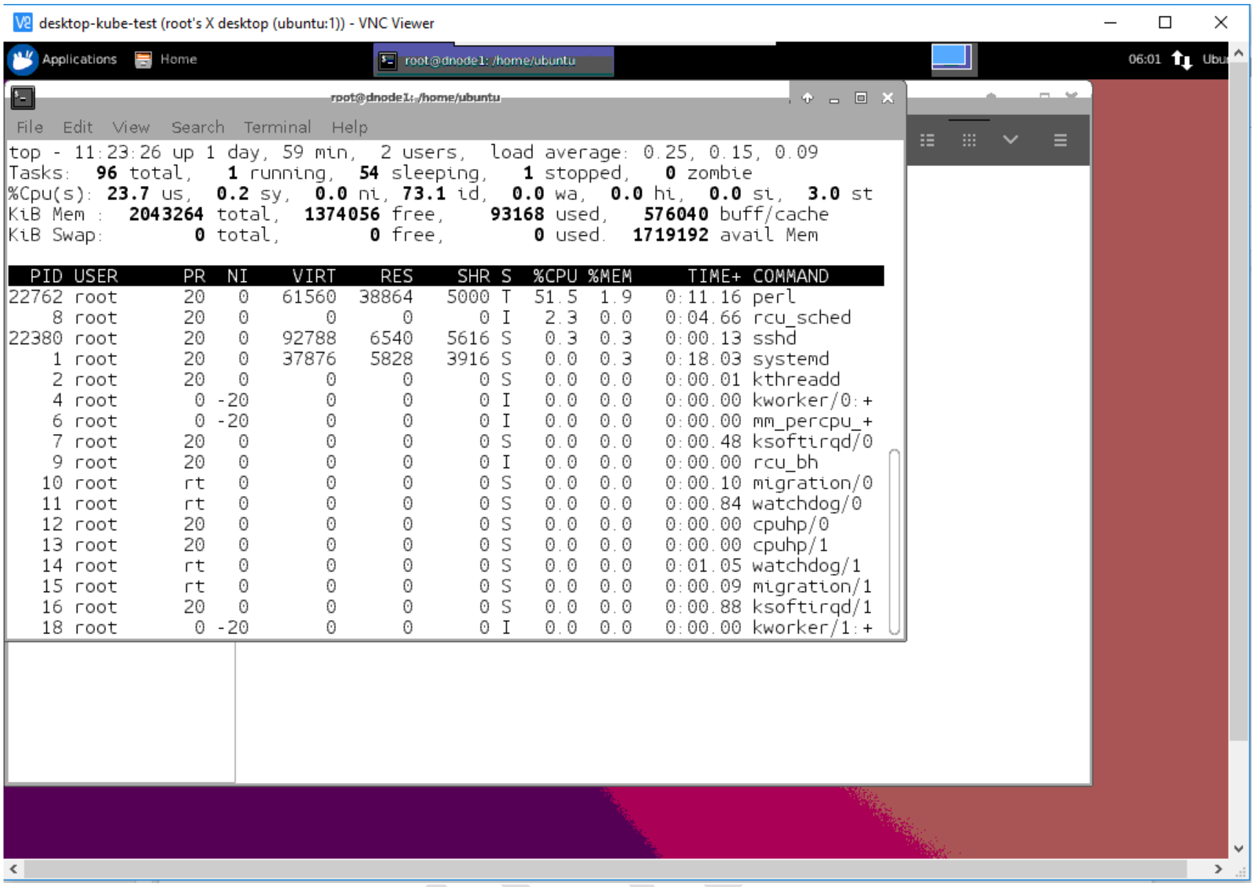
- Execute primes command with cpulimit at the beginning

```
    cpulimit -l 50 primes 0 99999999999 > /dev/null &
```



```
root@dnode1:~# cpulimit -l 50 primes 0 99999999999 > /dev/null &
[1] 22761
root@dnode1:~#
```

- From VNC, observe from top command that process is limited to 50% of CPU


desktop-kube-test (root's X desktop (ubuntu:1)) - VNC Viewer
Applications
Home
root@dnode1: /home/ubuntu
06:01
root@dnode1: /home/ubuntu
File Edit View Search Terminal Help

```
top - 11:23:26 up 1 day, 59 min,  2 users,  load average: 0.25, 0.15, 0.09
Tasks:  96 total,   1 running,  54 sleeping,   1 stopped,   0 zombie
%Cpu(s): 23.7 us,  0.2 sy,  0.0 ni, 73.1 id,  0.0 wa,  0.0 hi,  0.0 si,  3.0 st
KiB Mem :  2043264 total,  1374056 free,    93168 used,   576040 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  1719192 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
22762 root      20   0   61560  38864   5000 T  51.5  1.9   0:11.16 perl
    8 root      20   0       0      0      0 I   2.3  0.0   0:04.66 rcu_sched
22380 root      20   0   92788   6540   5616 S   0.3  0.3   0:00.13 sshd
    1 root      20   0   37876   5828   3916 S   0.0  0.3   0:18.03 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kthreadd
    4 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/0:+
    6 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 mm_percpu_+
    7 root      20   0       0      0      0 S   0.0  0.0   0:00.48 ksoftirqd/0
    9 root      20   0       0      0      0 I   0.0  0.0   0:00.00 rcu_bh
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.10 migration/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.84 watchdog/0
   12 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/0
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/1
   14 root      rt   0       0      0      0 S   0.0  0.0   0:01.05 watchdog/1
   15 root      rt   0       0      0      0 S   0.0  0.0   0:00.09 migration/1
   16 root      20   0       0      0      0 S   0.0  0.0   0:00.88 ksoftirqd/1
   18 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/1:+
```

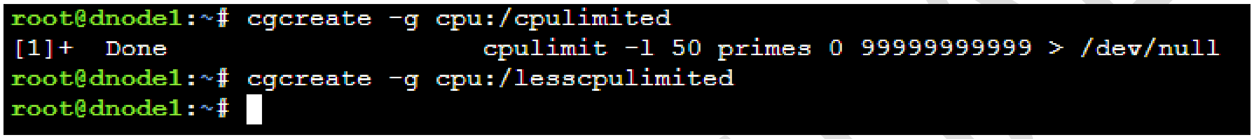
Exercise-9

Use cgroups to limit the cpu usage of a process

NOTE: *Perform below exercise on dnode1*

- Create 2 cgroups cpulimited and lesscpulimited

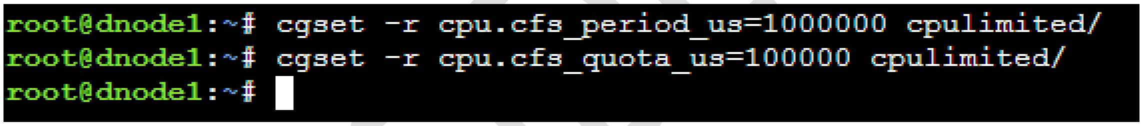
```
cgcreate -g cpu:/cpulimited

cgcreate -g cpu:/lesscpulimited
```



```
root@dnode1:~# cgcreate -g cpu:/cpulimited
[1]+  Done                    cpulimit -l 50 primes 0 99999999999 > /dev/null
root@dnode1:~# cgcreate -g cpu:/lesscpulimited
root@dnode1:~#
```

- Limit the cpu usage to 10% for cpulimited

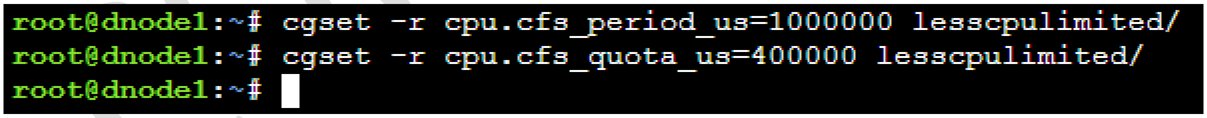
```
cgset -r cpu.cfs_period_us=1000000 cpulimited/

cgset -r cpu.cfs_quota_us=100000 cpulimited/
```



```
root@dnode1:~# cgset -r cpu.cfs_period_us=1000000 cpulimited/
root@dnode1:~# cgset -r cpu.cfs_quota_us=100000 cpulimited/
root@dnode1:~#
```

- Limit the cpu usage to 40% for lesscpulimited group

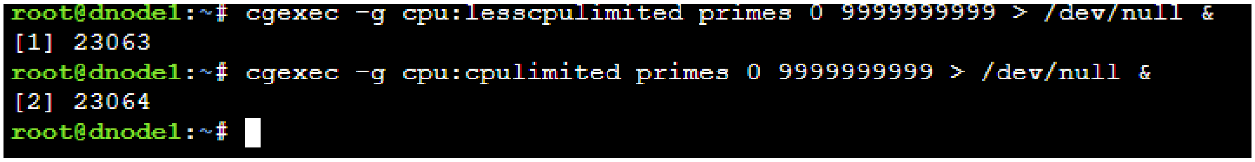
```
cgset -r cpu.cfs_period_us=1000000 lesscpulimited/

cgset -r cpu.cfs_quota_us=400000 lesscpulimited/
```



```
root@dnode1:~# cgset -r cpu.cfs_period_us=1000000 lesscpulimited/
root@dnode1:~# cgset -r cpu.cfs_quota_us=400000 lesscpulimited/
root@dnode1:~#
```

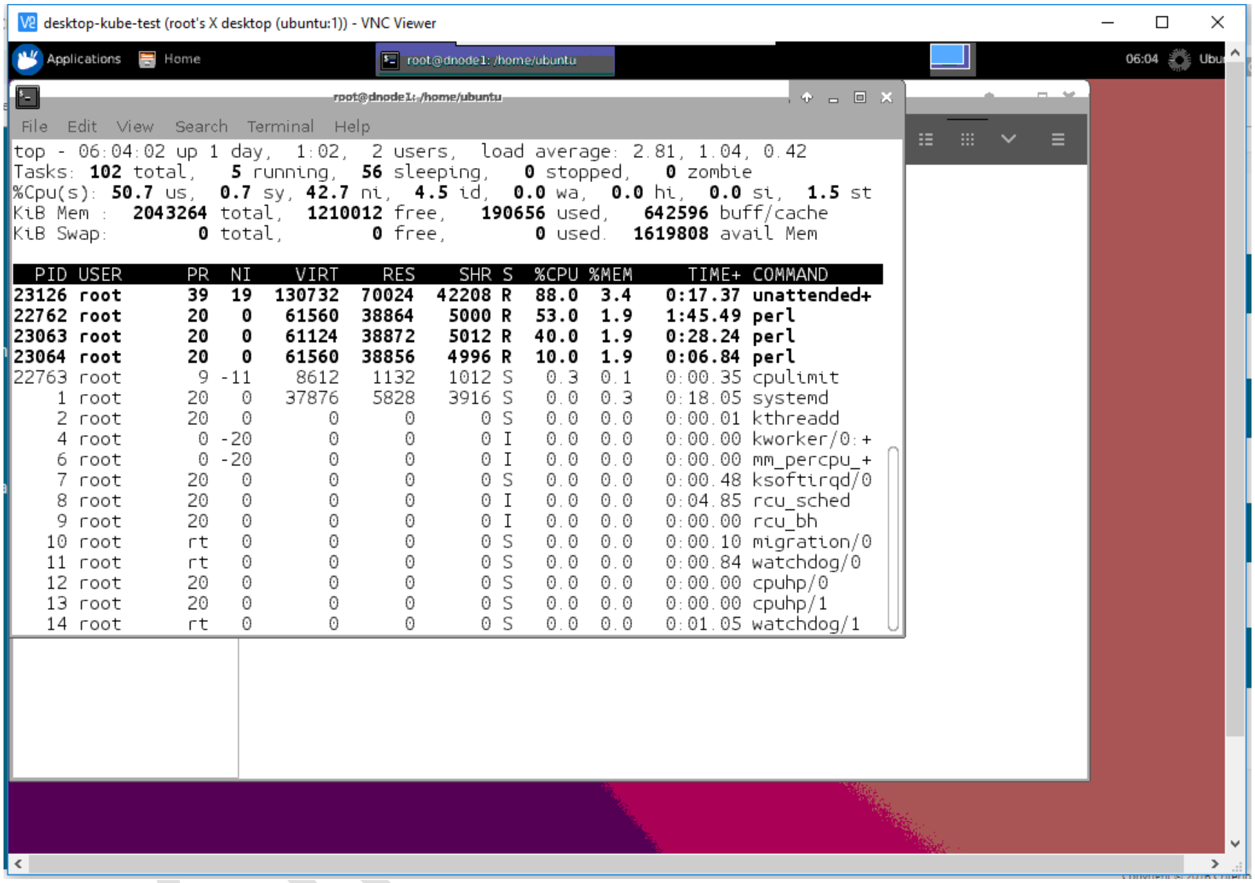
- Run the primes process with these new cgroups

```
cgexec -g cpu:lesscpulimited primes 0 9999999999 > /dev/null &

cgexec -g cpu:cpulimited primes 0 9999999999 > /dev/null &
```



```
root@dnode1:~# cgexec -g cpu:lesscpulimited primes 0 9999999999 > /dev/null &
[1] 23063
root@dnode1:~# cgexec -g cpu:cpulimited primes 0 9999999999 > /dev/null &
[2] 23064
root@dnode1:~# 
```

- From VNC, observe from top command that 2 processes are limited to 40% and 10% of CPU respectively



desktop-kube-test (root's X desktop (ubuntu:1)) - VNC Viewer
Applications
Home
root@dnode1: /home/ubuntu
06:04
root@dnode1: /home/ubuntu
File Edit View Search Terminal Help

```
top - 06:04:02 up 1 day,  1:02,  2 users,  load average: 2.81, 1.04, 0.42
Tasks: 102 total,   5 running,  56 sleeping,   0 stopped,   0 zombie
%Cpu(s): 50.7 us,  0.7 sy, 42.7 ni,  4.5 id,  0.0 wa,  0.0 hi,  0.0 si,  1.5 st
KiB Mem :  2043264 total,  1210012 free,   190656 used,   642596 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  1619808 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
23126 root      39  19  130732  70024  42208 R  88.0  3.4   0:17.37 unattended+
22762 root      20   0   61560  38864   5000 R  53.0  1.9   1:45.49 perl
23063 root      20   0   61124  38872   5012 R  40.0  1.9   0:28.24 perl
23064 root      20   0   61560  38856   4996 R  10.0  1.9   0:06.84 perl
22763 root       9 -11    8612   1132   1012 S   0.3  0.1   0:00.35 cpulimit
    1 root      20   0   37876   5828   3916 S   0.0  0.3   0:18.05 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kthreadd
    4 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/0:+
    6 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 mm_percpu_+
    7 root      20   0       0      0      0 S   0.0  0.0   0:00.48 ksoftirqd/0
    8 root      20   0       0      0      0 I   0.0  0.0   0:04.85 rcu_sched
    9 root      20   0       0      0      0 I   0.0  0.0   0:00.00 rcu_bh
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.10 migration/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.84 watchdog/0
   12 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/0
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/1
   14 root      rt   0       0      0      0 S   0.0  0.0   0:01.05 watchdog/1
```

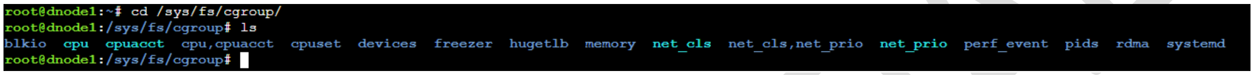
Execise-10

Explore the cgroup filesystem

NOTE: *Perform the below exercise on dnode1*

- All the cgroup controllers will be under /sys/fs/cgroup/

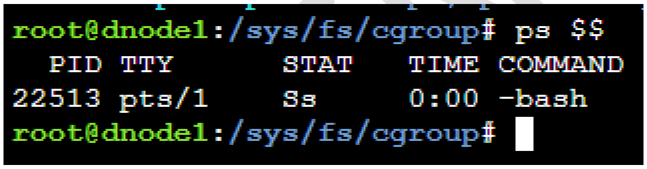
```
cd /sys/fs/cgroup/

ls
```



```
root@dnode1:~# cd /sys/fs/cgroup/
root@dnode1:/sys/fs/cgroup# ls
blkio  cpu  cpuacct  cpu,cpuacct  cpuset  devices  freezer  hugetlb  memory  net_cls  net_cls,net_prio  net_prio  perf_event  pids  rdma  systemd
root@dnode1:/sys/fs/cgroup# 
```

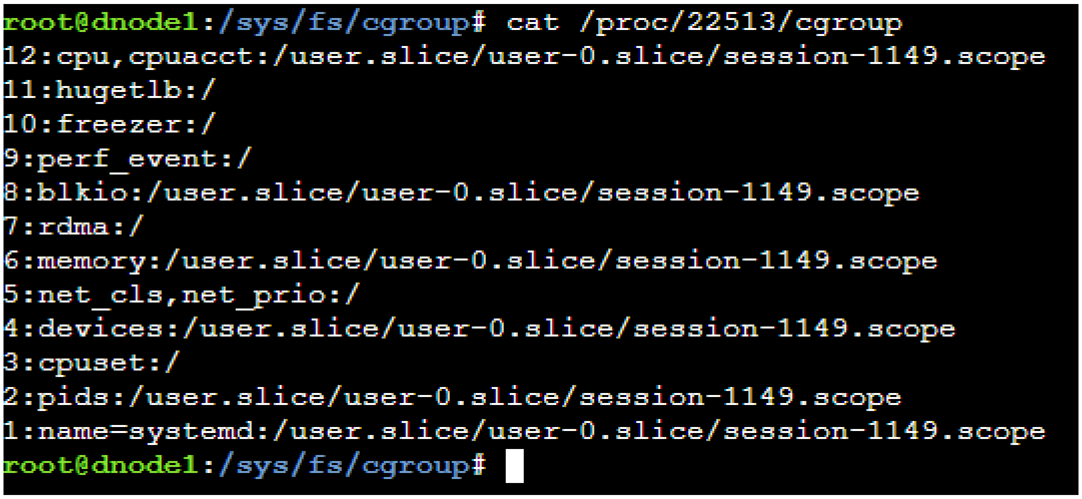
- Verify the default cpu, memory values for bash process

```
ps $$
```



```
root@dnode1:/sys/fs/cgroup# ps $$
  PID TTY      STAT   TIME COMMAND
22513 pts/1    Ss     0:00 -bash
root@dnode1:/sys/fs/cgroup# 
```

```
   cat /proc/<PID>/cgroup
```



```
root@dnode1:/sys/fs/cgroup# cat /proc/22513/cgroup
12:cpu,cpuacct:/user.slice/user-0.slice/session-1149.scope
11:hugetlb:/
10:freezer:/
9:perf_event:/
8:blkio:/user.slice/user-0.slice/session-1149.scope
7:rdma:/
6:memory:/user.slice/user-0.slice/session-1149.scope
5:net_cls,net_prio:/
4:devices:/user.slice/user-0.slice/session-1149.scope
3:cpuset:/
2:pids:/user.slice/user-0.slice/session-1149.scope
1:name=systemd:/user.slice/user-0.slice/session-1149.scope
root@dnode1:/sys/fs/cgroup# 
```

(Verify the cgroup hierarchy followed for this process)

- Get the process ID from above 2 prime process (re-run if closed)
- Verify their cgroup file under /proc/PID/cgroup using their actual process IDs

Docker Part-I

Exercise-11

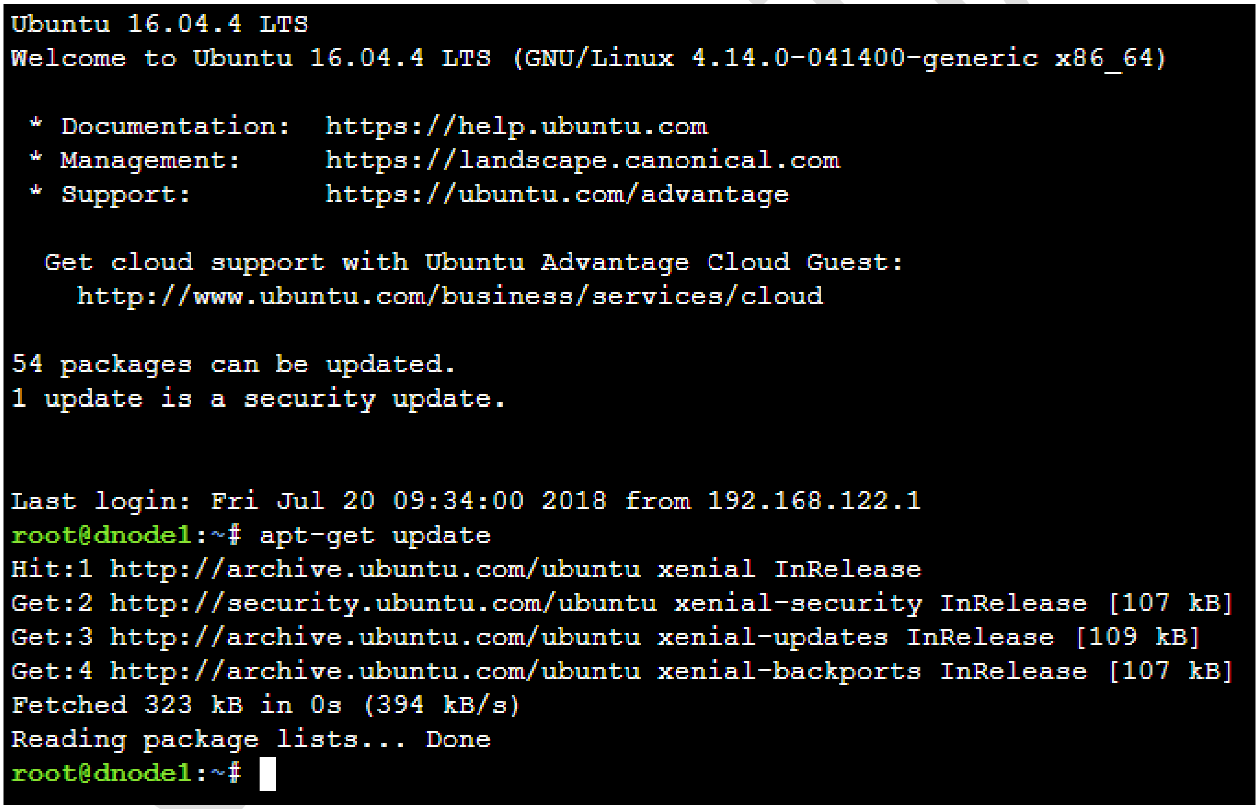
Docker Installation

NOTE: *Perform the below exercise on dnode1*

Setup the repository

- Update the apt package index:

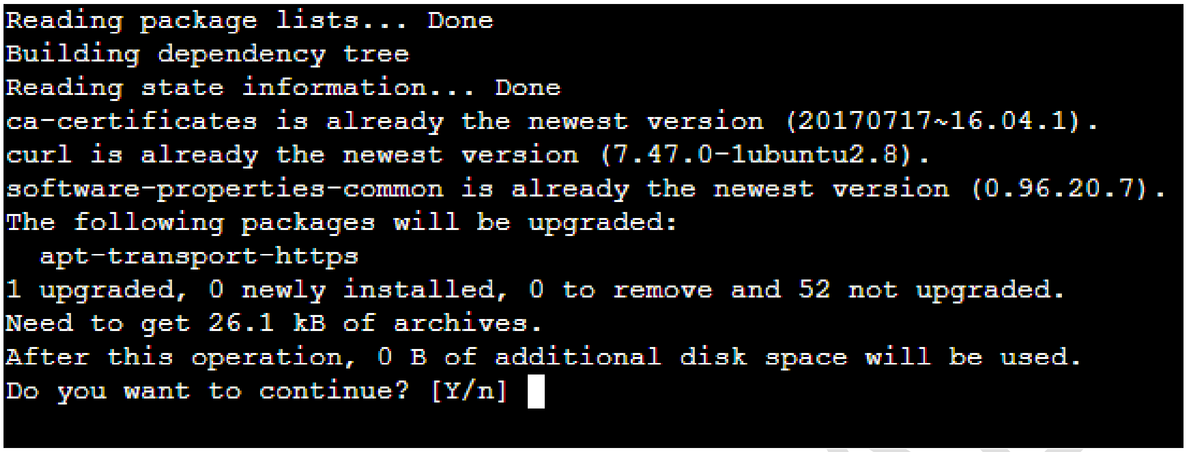
```
apt-get update
```



```
Ubuntu 16.04.4 LTS
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.14.0-041400-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

54 packages can be updated.
1 update is a security update.


Last login: Fri Jul 20 09:34:00 2018 from 192.168.122.1
root@dnode1:~# apt-get update
Hit:1 http://archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [107 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:4 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Fetched 323 kB in 0s (394 kB/s)
Reading package lists... Done
root@dnode1:~#
```

- Install packages to allow apt to use a repository over HTTPS:

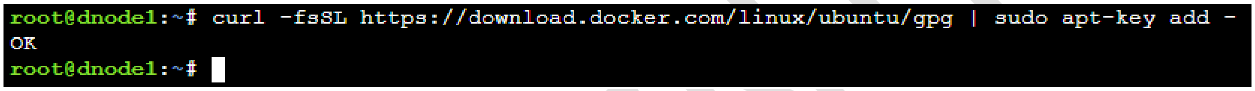
```
apt-get install  apt-transport-https ca-certificates curl software-properties-common
```



```
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20170717~16.04.1).
curl is already the newest version (7.47.0-1ubuntu2.8).
software-properties-common is already the newest version (0.96.20.7).
The following packages will be upgraded:
  apt-transport-https
1 upgraded, 0 newly installed, 0 to remove and 52 not upgraded.
Need to get 26.1 kB of archives.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n]
```

- Add Dockers official GPG key:

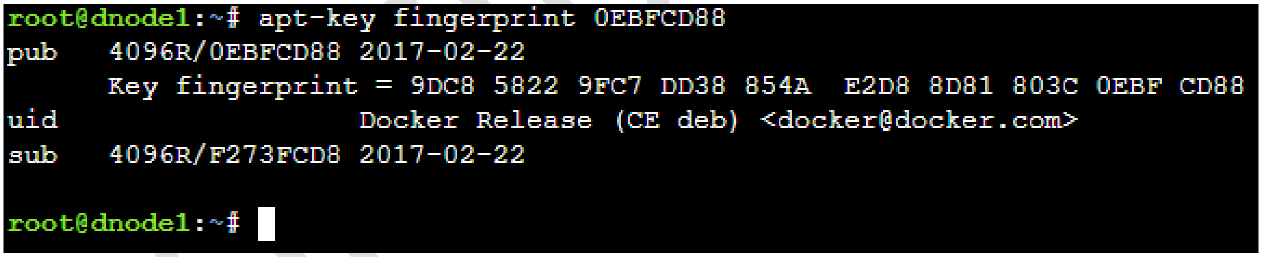
```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```



```
root@dnode1:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
OK
root@dnode1:~#
```

- Verify that you now have the key with the fingerprint 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88, by searching for the last 8 characters of the fingerprint.

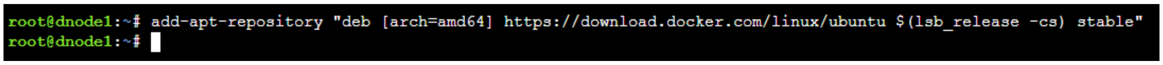
```
apt-key fingerprint 0EBFCD88
```



```
root@dnode1:~# apt-key fingerprint 0EBFCD88
pub   4096R/0EBFCD88 2017-02-22
      Key fingerprint = 9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid                  Docker Release (CE deb) <docker@docker.com>
sub   4096R/F273FCD8 2017-02-22

root@dnode1:~#
```

- Use the following command to set up the stable repository.

```
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
```

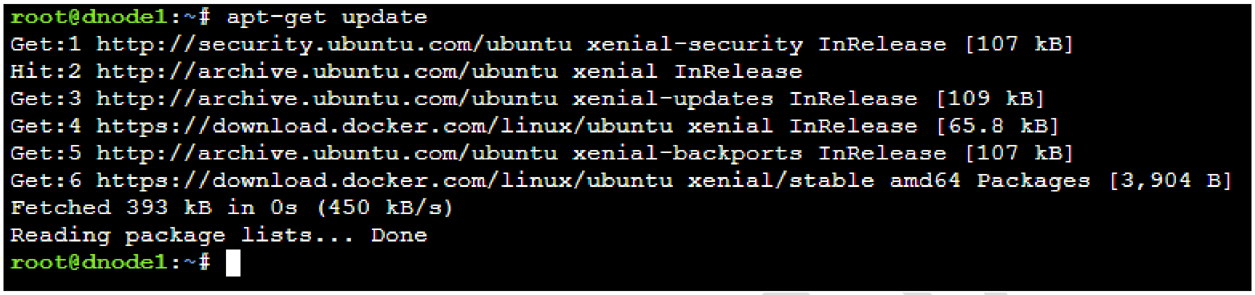


```
root@dnode1:~# add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
root@dnode1:~#
```

Install Docker CE

- Update the apt package index.

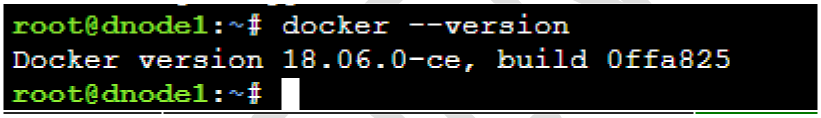
```
apt-get update
```



```
root@dnode1:~# apt-get update
Get:1 http://security.ubuntu.com/ubuntu xenial-security InRelease [107 kB]
Hit:2 http://archive.ubuntu.com/ubuntu xenial InRelease
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:4 https://download.docker.com/linux/ubuntu xenial InRelease [65.8 kB]
Get:5 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Get:6 https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages [3,904 B]
Fetched 393 kB in 0s (450 kB/s)
Reading package lists... Done
root@dnode1:~#
```

- Install the latest version of Docker CE

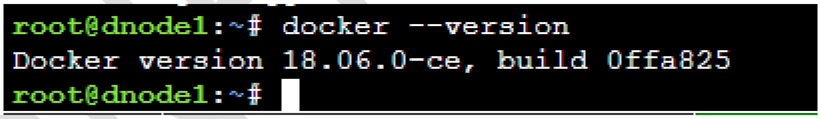
```
apt-get install docker-ce
```



```
root@dnode1:~# docker --version
Docker version 18.06.0-ce, build 0ffa825
root@dnode1:~#
```

- Verify docker version

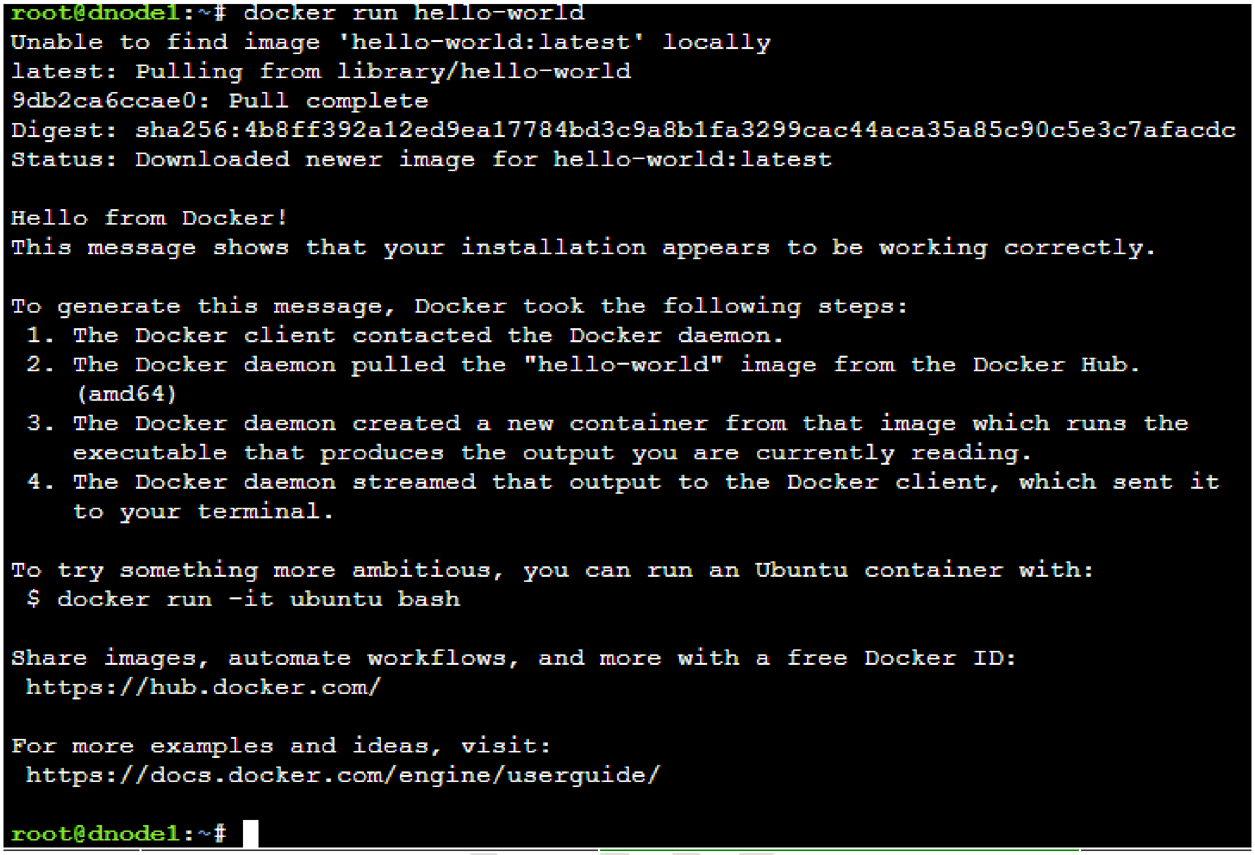
```
docker --version
```



```
root@dnode1:~# docker --version
Docker version 18.06.0-ce, build 0ffa825
root@dnode1:~#
```

- Verify that Docker CE is installed correctly by running the hello-world image.

```
docker run hello-world
```



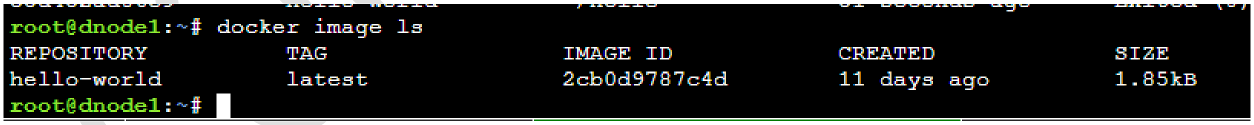
```
root@dnode1:~# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
9db2ca6ccae0: Pull complete
Digest: sha256:4b8ff392a12ed9ea17784bd3c9a8b1fa3299cac44aca35a85c90c5e3c7afacdc
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/

root@dnode1:~# 
```

- List all the docker containers

```
docker ps -a
```

- List all the docker images

```
docker image ls
```



```
root@dnode1:~# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              2cb0d9787c4d        11 days ago         1.85kB
root@dnode1:~# 
```

- Hello-world program is a basic scratch image. Check the source code

https://github.com/docker-library/hello-world/blob/master/hello.c

- Docker Inspect returns low level information on docker objects

```
docker inspect <container ID>
```

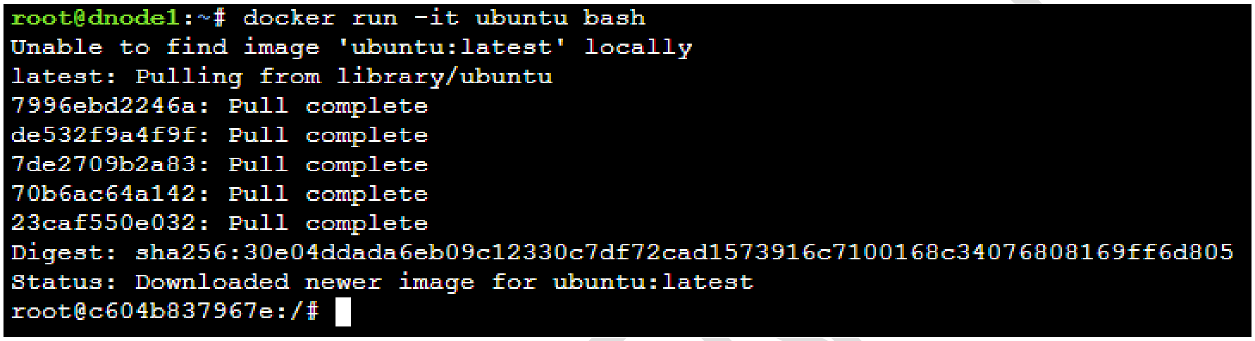
Exercise-12

Bring up Ubuntu Docker Container

NOTE: *Perform the below exercise on dnode1*

- Bring up Ubuntu container

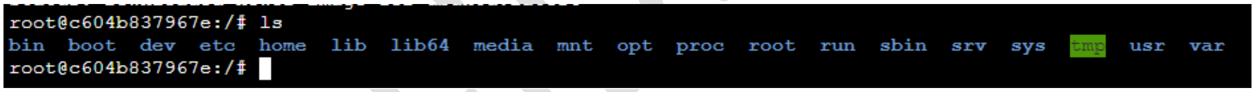
```
docker run -it ubuntu bash
```



```
root@dnode1:~# docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
7996ebd2246a: Pull complete
de532f9a4f9f: Pull complete
7de2709b2a83: Pull complete
70b6ac64a142: Pull complete
23caf550e032: Pull complete
Digest: sha256:30e04ddada6eb09c12330c7df72cad1573916c7100168c34076808169ff6d805
Status: Downloaded newer image for ubuntu:latest
root@c604b837967e:/#
```

- List the directories

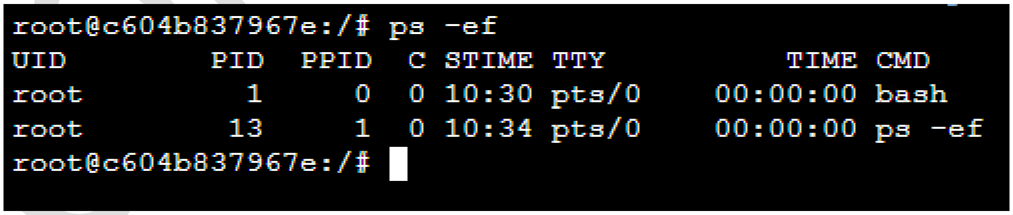
```
ls
```



```
root@c604b837967e:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@c604b837967e:/#
```

- List the processes running

```
ps -ef
```

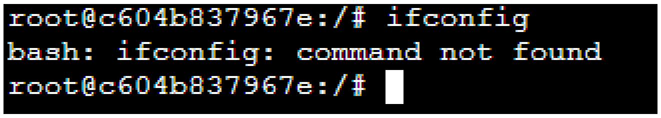


```
root@c604b837967e:/# ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 10:30 pts/0    00:00:00 bash
root          13     1  0 10:34 pts/0    00:00:00 ps -ef
root@c604b837967e:/#
```

- Check the Interfaces

```
ifconfig
```

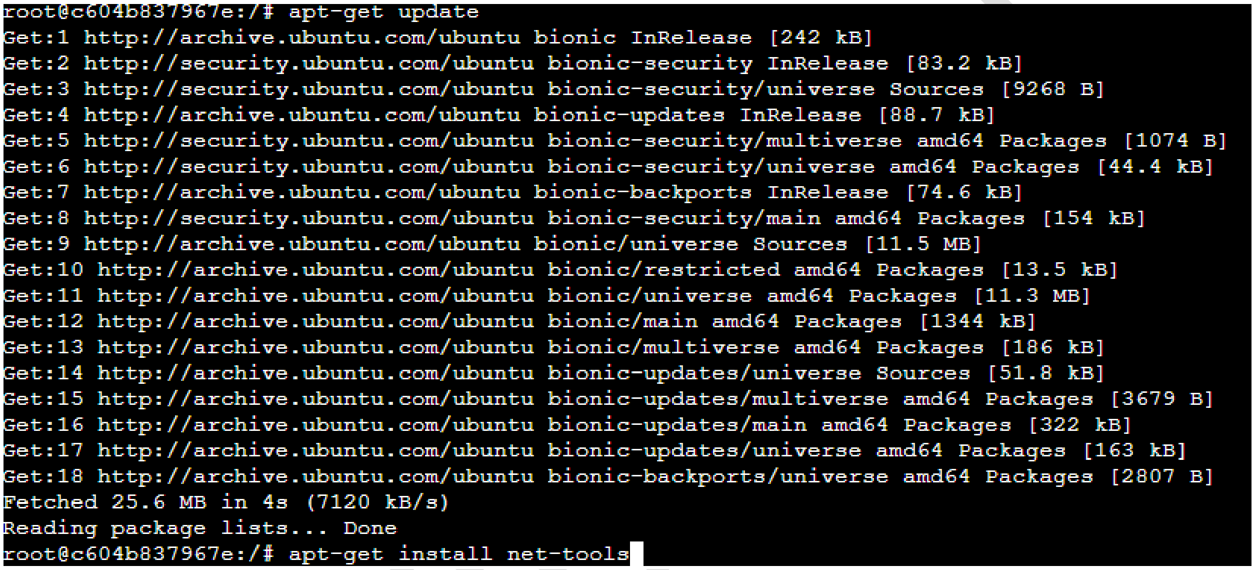
(You will see it says command is not available.)



```
root@c604b837967e:/# ifconfig
bash: ifconfig: command not found
root@c604b837967e:/#
```

- Install relevant packages

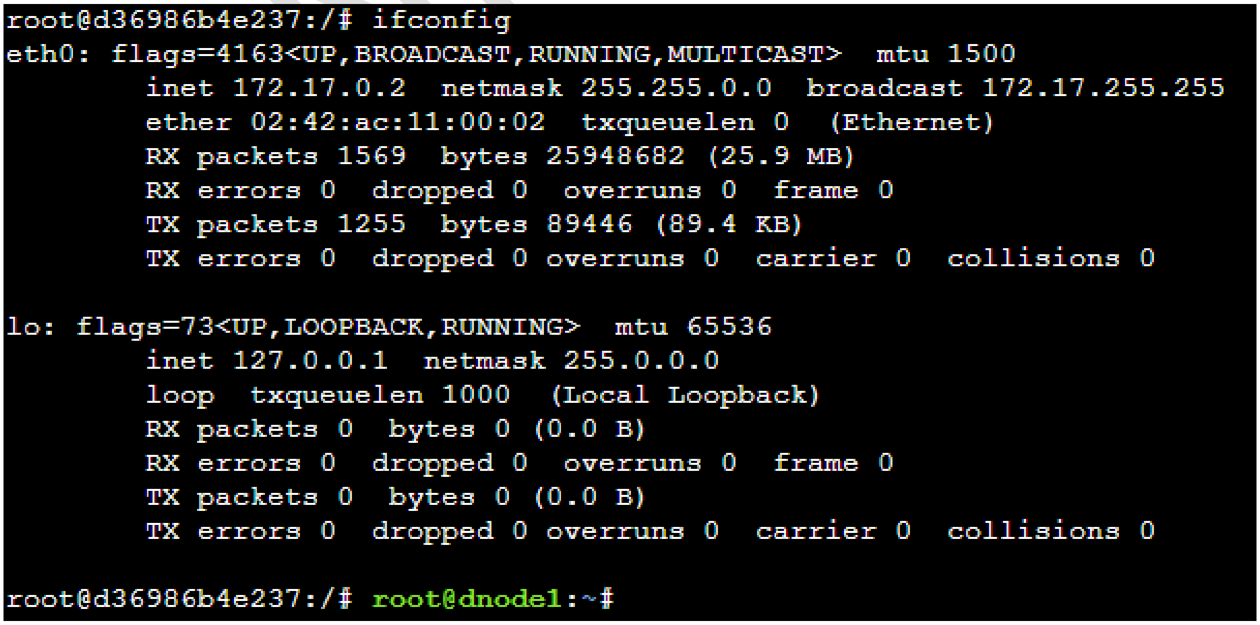
```
apt-get update

apt-get install net-tools
```



```
root@c604b837967e:/# apt-get update
Get:1 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security/universe Sources [9268 B]
Get:4 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [1074 B]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [44.4 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [154 kB]
Get:9 http://archive.ubuntu.com/ubuntu bionic/universe Sources [11.5 MB]
Get:10 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/universe Sources [51.8 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [3679 B]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [322 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [163 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [2807 B]
Fetched 25.6 MB in 4s (7120 kB/s)
Reading package lists... Done
root@c604b837967e:/# apt-get install net-tools
```

- Check the interfaces again

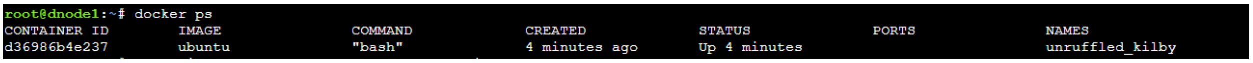
```
ifconfig
```



```
root@d36986b4e237:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 1569  bytes 25948682 (25.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1255  bytes 89446 (89.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@d36986b4e237:/# root@dnode1:~#
```

- SSH into dnode1, i.e, duplicate the browser or get a new dnode1 terminal from access devices. To get container ID use command "docker ps" in dnode1

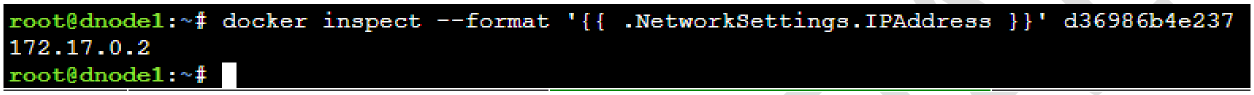
```
docker ps
```



```
root@dnode1:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
d36986b4e237        ubuntu              "bash"              4 minutes ago       Up 4 minutes                            unruffled_kilby
```

- Verify the IP address of container from host

```
docker inspect --format '{{ .NetworkSettings.IPAddress }}' <CONTAINERID>
```



```
root@dnode1:~# docker inspect --format '{{ .NetworkSettings.IPAddress }}' d36986b4e237
172.17.0.2
root@dnode1:~# 
```

- To keep the docker running but detach from it gracefully, press CTRL+p, CTRL+q

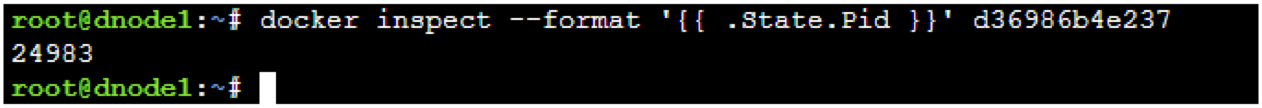
Note: In case you exit the container, all the changes done to it, will be gone.

Execise-13

Verify the PID, NET namespaces and CGROUP of ubuntu-bash process

- Get the PID of the bash process inside Ubuntu container

```
docker inspect --format '{{ .State.Pid }}' <CONTAINERID>
```

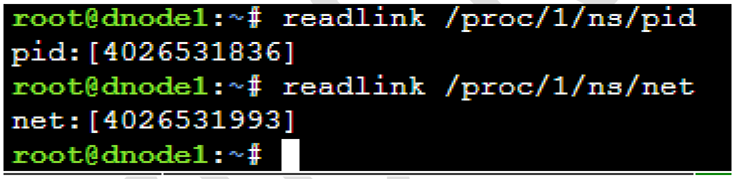


```
root@dnode1:~# docker inspect --format '{{ .State.Pid }}' d36986b4e237
24983
root@dnode1:~#
```

(say the output process number is [PIDbash])

- Verify the PID and NET namespace of INIT process on base node

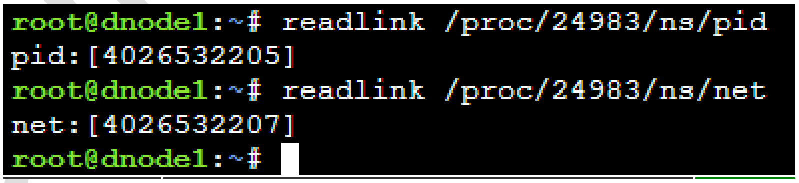
```
readlink /proc/1/ns/pid

readlink /proc/1/ns/net
```



```
root@dnode1:~# readlink /proc/1/ns/pid
pid:[4026531836]
root@dnode1:~# readlink /proc/1/ns/net
net:[4026531993]
root@dnode1:~#
```

- Verify the PID and NET namespace of containers bash process

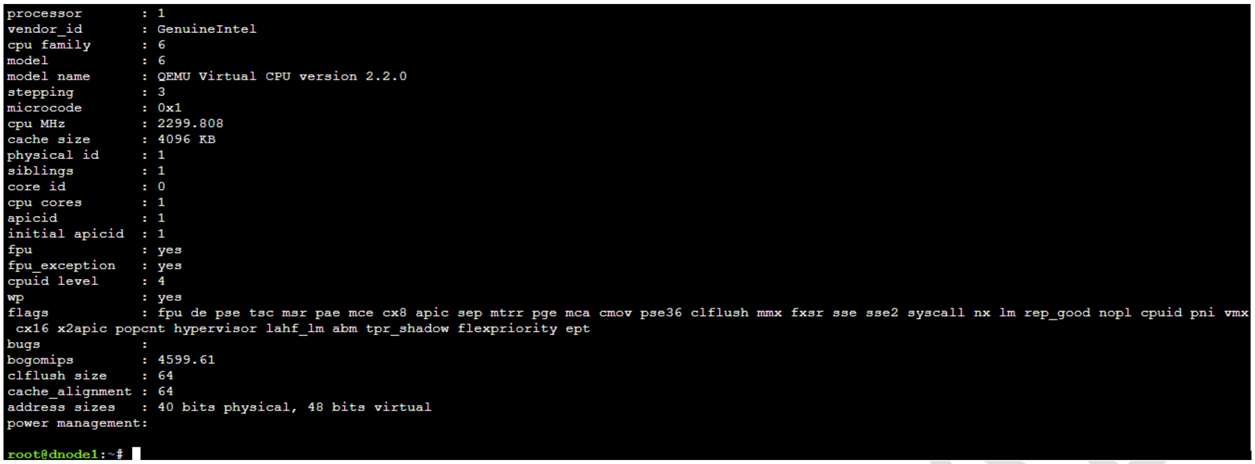
```
readlink /proc/<PIDbash>/ns/pid

readlink /proc/<PIDbash>/ns/net
```



```
root@dnode1:~# readlink /proc/24983/ns/pid
pid:[4026532205]
root@dnode1:~# readlink /proc/24983/ns/net
net:[4026532207]
root@dnode1:~#
```

- Verify the CPUINFO of base node

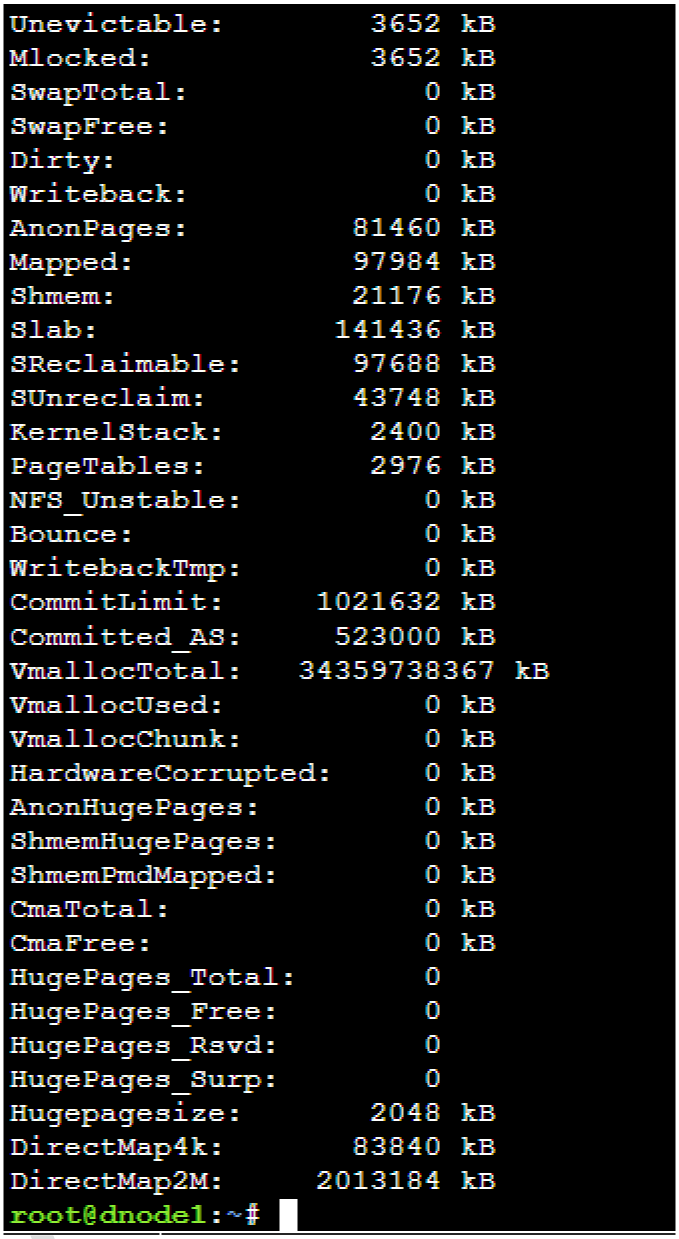
```
cat /proc/cpuinfo
```



```
processor       : 1
vendor_id       : GenuineIntel
cpu family      : 6
model           : 6
model name      : QEMU Virtual CPU version 2.2.0
stepping        : 3
microcode       : 0x1
cpu MHz         : 2299.808
cache size      : 4096 KB
physical id     : 1
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 1
initial apicid  : 1
fpu             : yes
fpu_exception   : yes
cpuid level     : 4
wp              : yes
flags           : fpu de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pse36 clflush mmx fxsr sse sse2 syscall nx lm rep_good nopl cpuid pni vmx
 cx16 x2apic popcnt hypervisor lahf_lm abm tpr_shadow flexpriority ept
bugs            :
bogomips        : 4599.61
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

root@dnode1:~#
```

- Verify the MEMINFO of base node

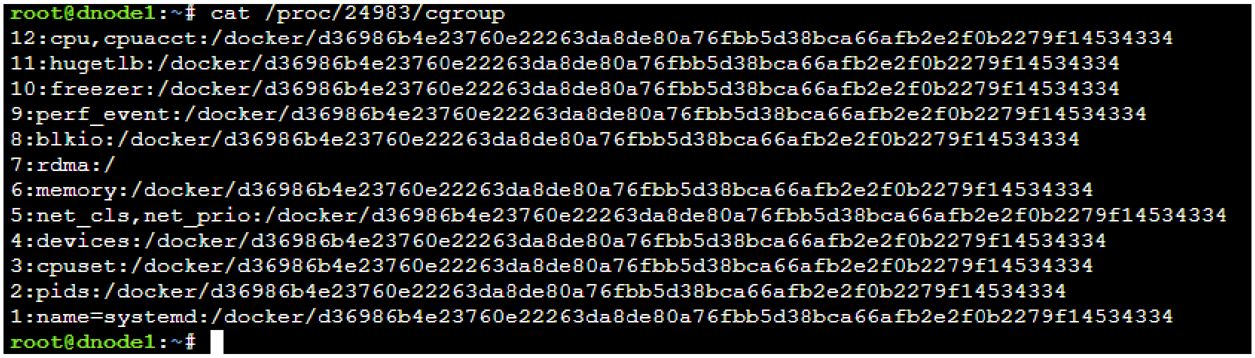
```
cat /proc/meminfo
```



```
Unevictable:        3652 kB
Mlocked:            3652 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:         81460 kB
Mapped:            97984 kB
Shmem:             21176 kB
Slab:             141436 kB
SReclaimable:      97688 kB
SUnreclaim:        43748 kB
KernelStack:        2400 kB
PageTables:         2976 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     1021632 kB
Committed_AS:     523000 kB
VmallocTotal:   34359738367 kB
VmallocUsed:           0 kB
VmallocChunk:          0 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:       83840 kB
DirectMap2M:     2013184 kB
root@dnode1:~#
```

- Verify the CPUSET and MEMORY Hierarchy Path for this process

```
cat /proc/<PIDbash>/cgroup
```

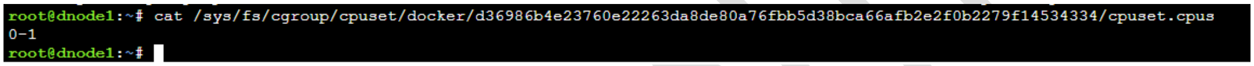


```
root@dnode1:~# cat /proc/24983/cgroup
12:cpu,cpuacct:/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334
11:hugetlb:/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334
10:freezer:/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334
9:perf_event:/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334
8:blkio:/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334
7:rdma:/
6:memory:/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334
5:net_cls,net_prio:/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334
4:devices:/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334
3:cpuset:/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334
2:pids:/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334
1:name=systemd:/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334
root@dnode1:~#
```

(Check the cpuset row for CGROUP Hierarchy location)

- For CPUSET

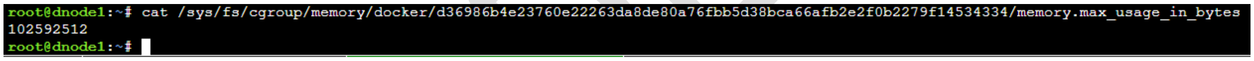
```
cat /sys/fs/cgroup/cpuset/<PATH-TO-CGROUP>/cpuset.cpus
```



```
root@dnode1:~# cat /sys/fs/cgroup/cpuset/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334/cpuset.cpus
0-1
root@dnode1:~#
```

- For memory

```
cat /sys/fs/cgroup/memory/<PATH-TO-CGROUP>/memory.max_usage_in_bytes
```



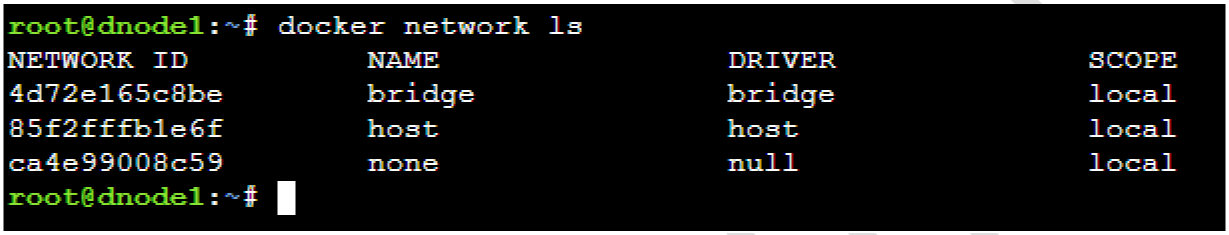
```
root@dnode1:~# cat /sys/fs/cgroup/memory/docker/d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334/memory.max_usage_in_bytes
102592512
root@dnode1:~#
```

Exercise-14

Docker Networking Single Node with user defined network

- Perform below commands on dnode1
- Verify the default docker networks

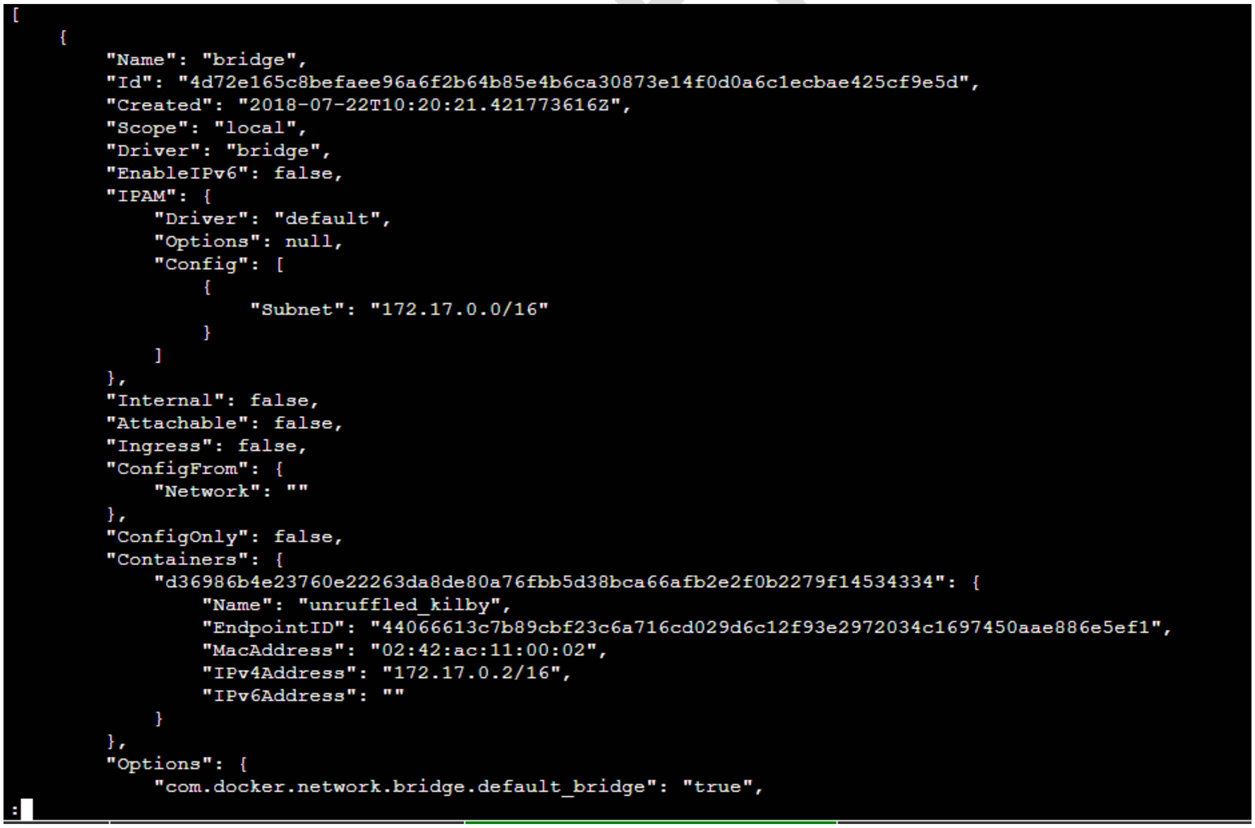
```
docker network ls
```



```
root@dnode1:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
4d72e165c8be        bridge              bridge              local
85f2fffb1e6f        host                host                local
ca4e99008c59        none                null                local
root@dnode1:~#
```

- Inspect the docker default network bridge

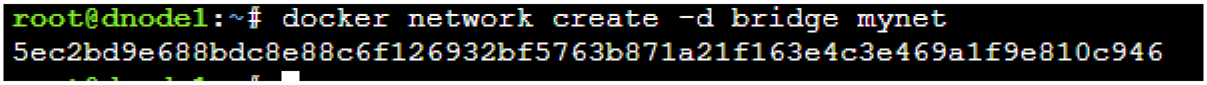
```
docker network inspect bridge
```



```
[
    {
        "Name": "bridge",
        "Id": "4d72e165c8befaee96a6f2b64b85e4b6ca30873e14f0d0a6c1ecbae425cf9e5d",
        "Created": "2018-07-22T10:20:21.421773616Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "d36986b4e23760e22263da8de80a76fbb5d38bca66afb2e2f0b2279f14534334": {
                "Name": "unruffled_kilby",
                "EndpointID": "44066613c7b89cbf23c6a716cd029d6c12f93e2972034c1697450aae886e5ef1",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
:
```

- Create your own bridge network

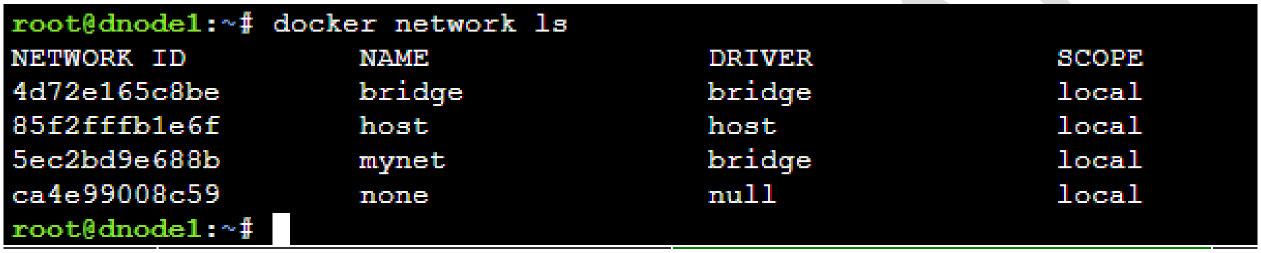
```
docker network create -d bridge mynet
```



```
root@dnode1:~# docker network create -d bridge mynet
5ec2bd9e688bdc8e88c6f126932bf5763b871a21f163e4c3e469a1f9e810c946
```

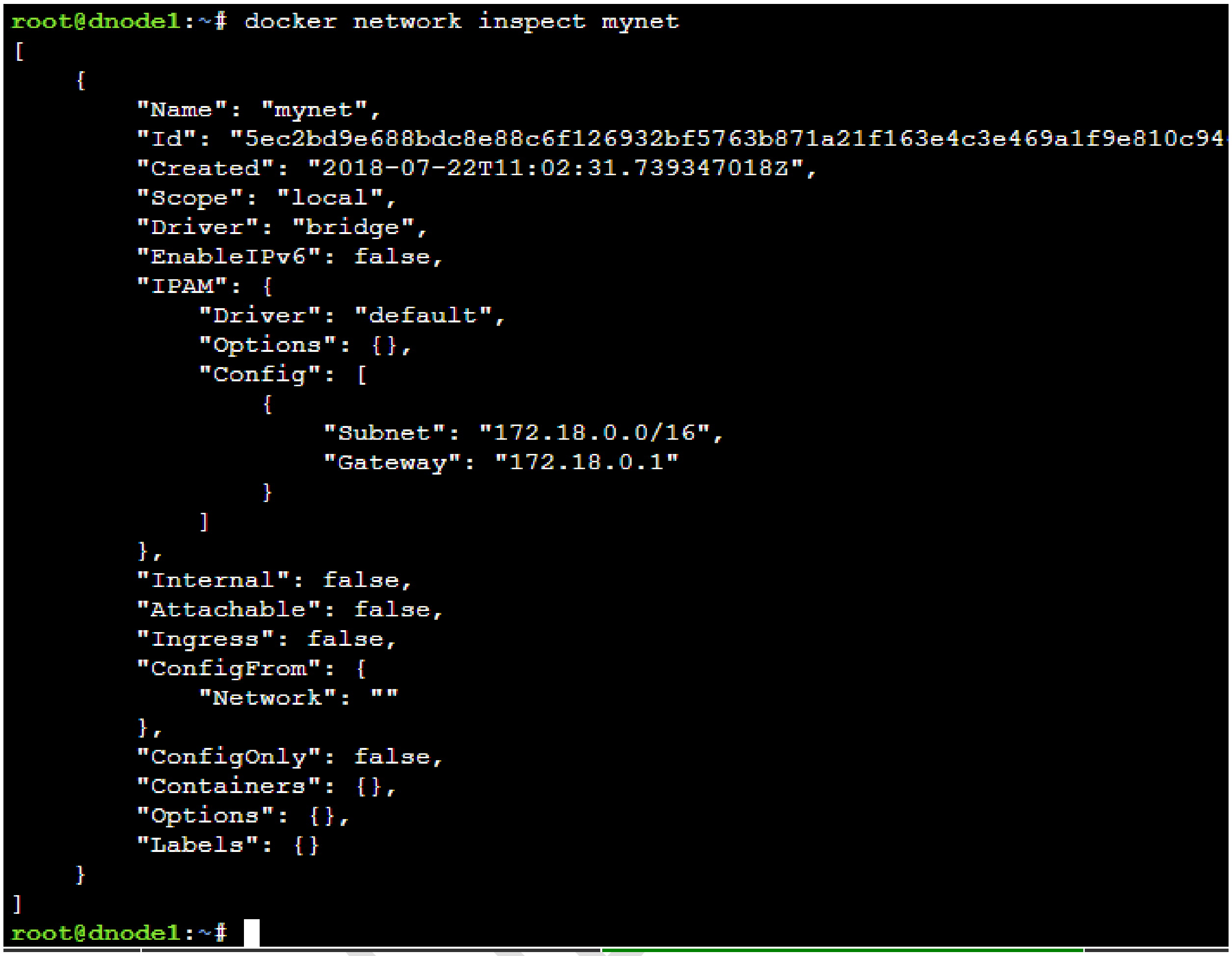
- List the networks and inspect new network

```
docker network ls
```



```
root@dnode1:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
4d72e165c8be        bridge              bridge              local
85f2fffb1e6f        host                host                local
5ec2bd9e688b        mynet               bridge              local
ca4e99008c59        none                null                local
root@dnode1:~#
```

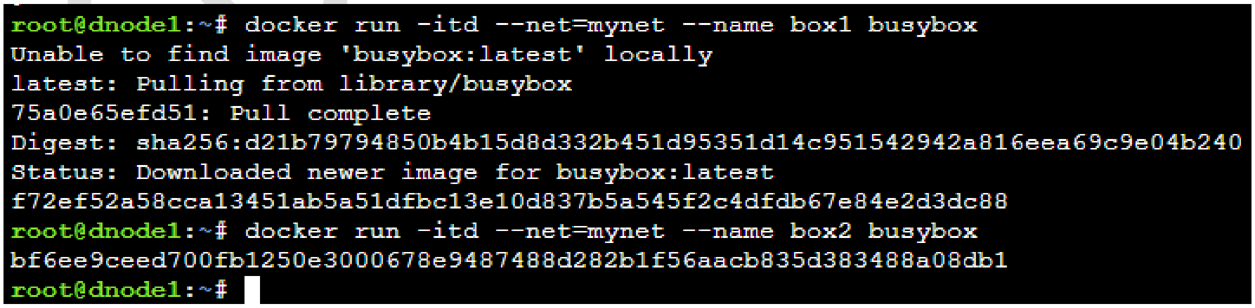
```
docker network inspect mynet
```



```
root@dnode1:~# docker network inspect mynet
[
    {
        "Name": "mynet",
        "Id": "5ec2bd9e688bdc8e88c6f126932bf5763b871a21f163e4c3e469a1f9e810c94
        "Created": "2018-07-22T11:02:31.739347018Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
root@dnode1:~#
```

- Create 2 new docker containers with busybox (Linux image with most of network tools already installed) image

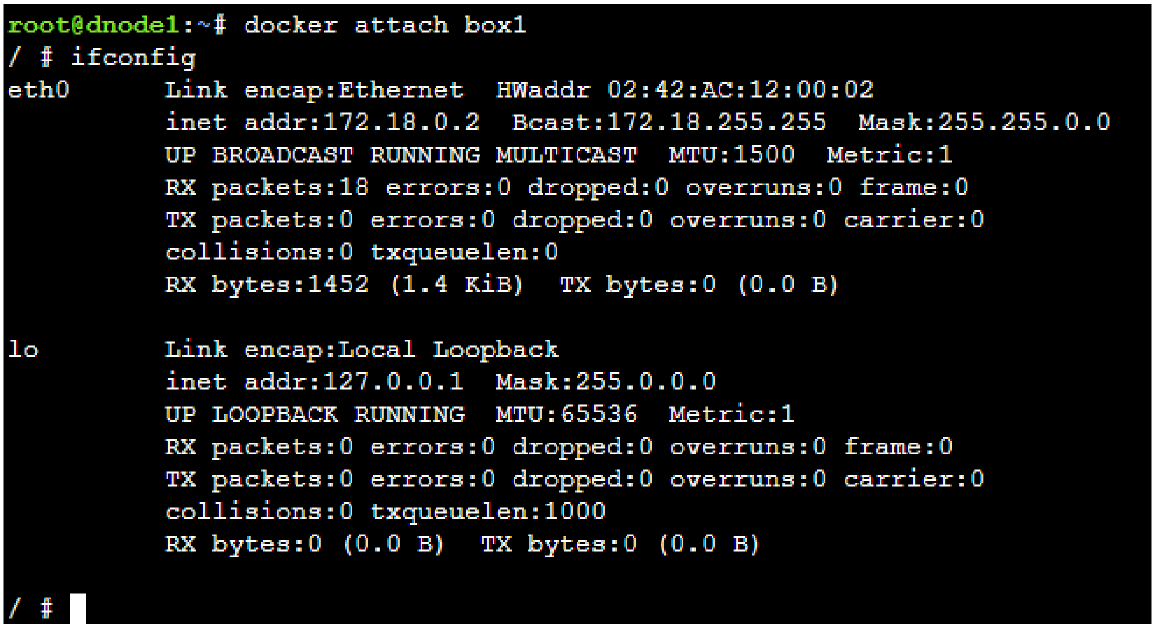
```
docker run -itd --net=mynet --name box1 busybox

docker run -itd --net=mynet --name box2 busybox
```



```
root@dnode1:~# docker run -itd --net=mynet --name box1 busybox
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
75a0e65efd51: Pull complete
Digest: sha256:d21b79794850b4b15d8d332b451d95351d14c951542942a816eea69c9e04b240
Status: Downloaded newer image for busybox:latest
f72ef52a58cca13451ab5a51dfbc13e10d837b5a545f2c4dfdb67e84e2d3dc88
root@dnode1:~# docker run -itd --net=mynet --name box2 busybox
bf6ee9ceed700fb1250e3000678e9487488d282b1f56aacb835d383488a08db1
root@dnode1:~#
```

- Attach to box1 container and observe its IP address (say box1-IP)

```
docker attach box1

ifconfig
```

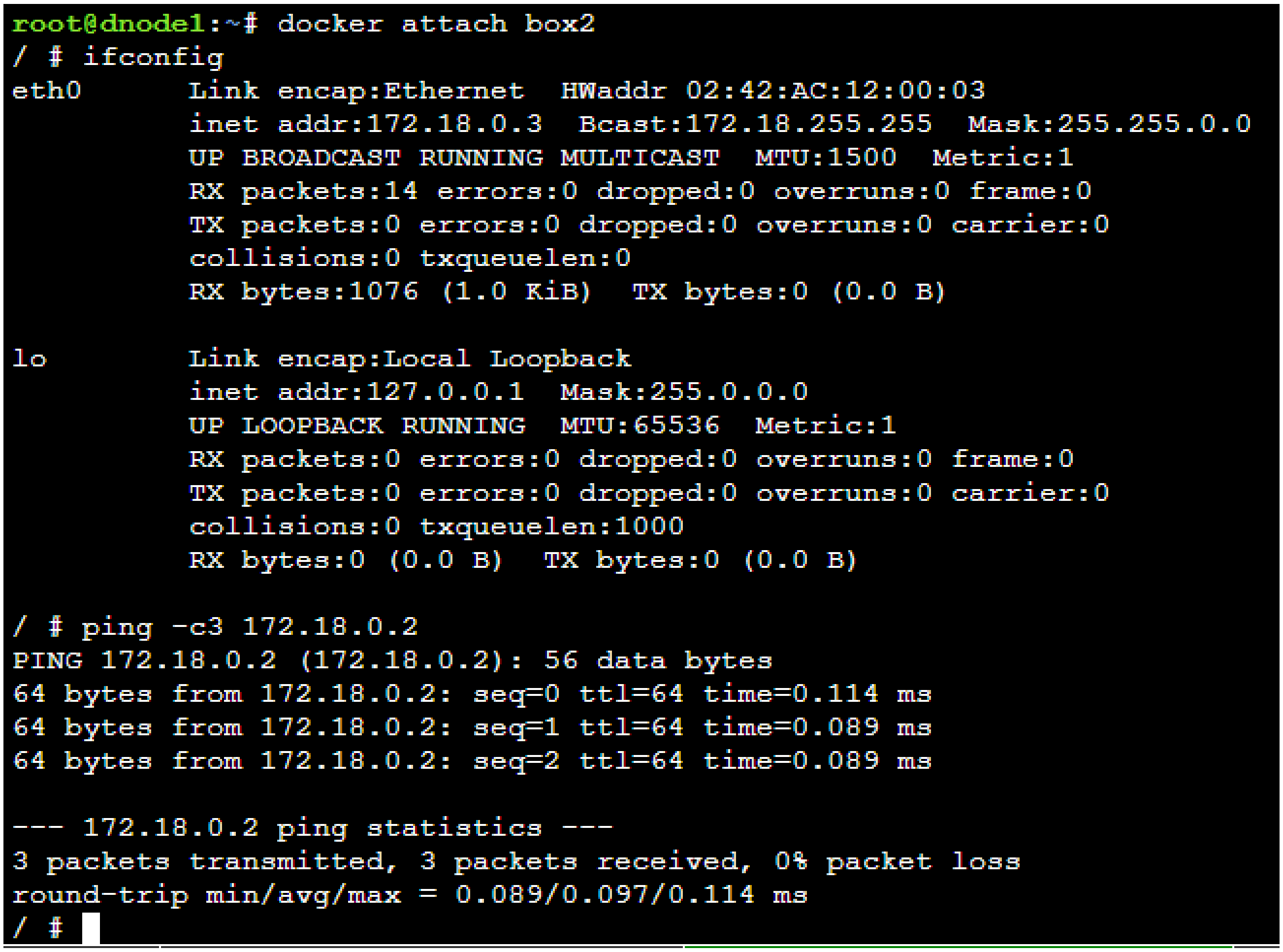


```
root@dnode1:~# docker attach box1
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:12:00:02
          inet addr:172.18.0.2  Bcast:172.18.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1452 (1.4 KiB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/ # 
```

Press Ctrl+p & Ctrl+q to detach from the box1 container

- Attach to box2 container and ping "box1" IP address. Verify it is reachable.

```
docker attach box2

ifconfig

ping <box1-IP>
```



```
root@dnode1:~# docker attach box2
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:12:00:03
          inet addr:172.18.0.3  Bcast:172.18.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1076 (1.0 KiB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/ # ping -c3 172.18.0.2
PING 172.18.0.2 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.114 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.089 ms
64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.089 ms

--- 172.18.0.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.089/0.097/0.114 ms
/ #
```

Exercise-15

Explore Overlay2 storage driver

- Create 2 ubuntu containers
- Inspect details of 2 containers

```
docker inspect <CONTAINERID-1>

docker inspect <CONTAINERID-2>
```

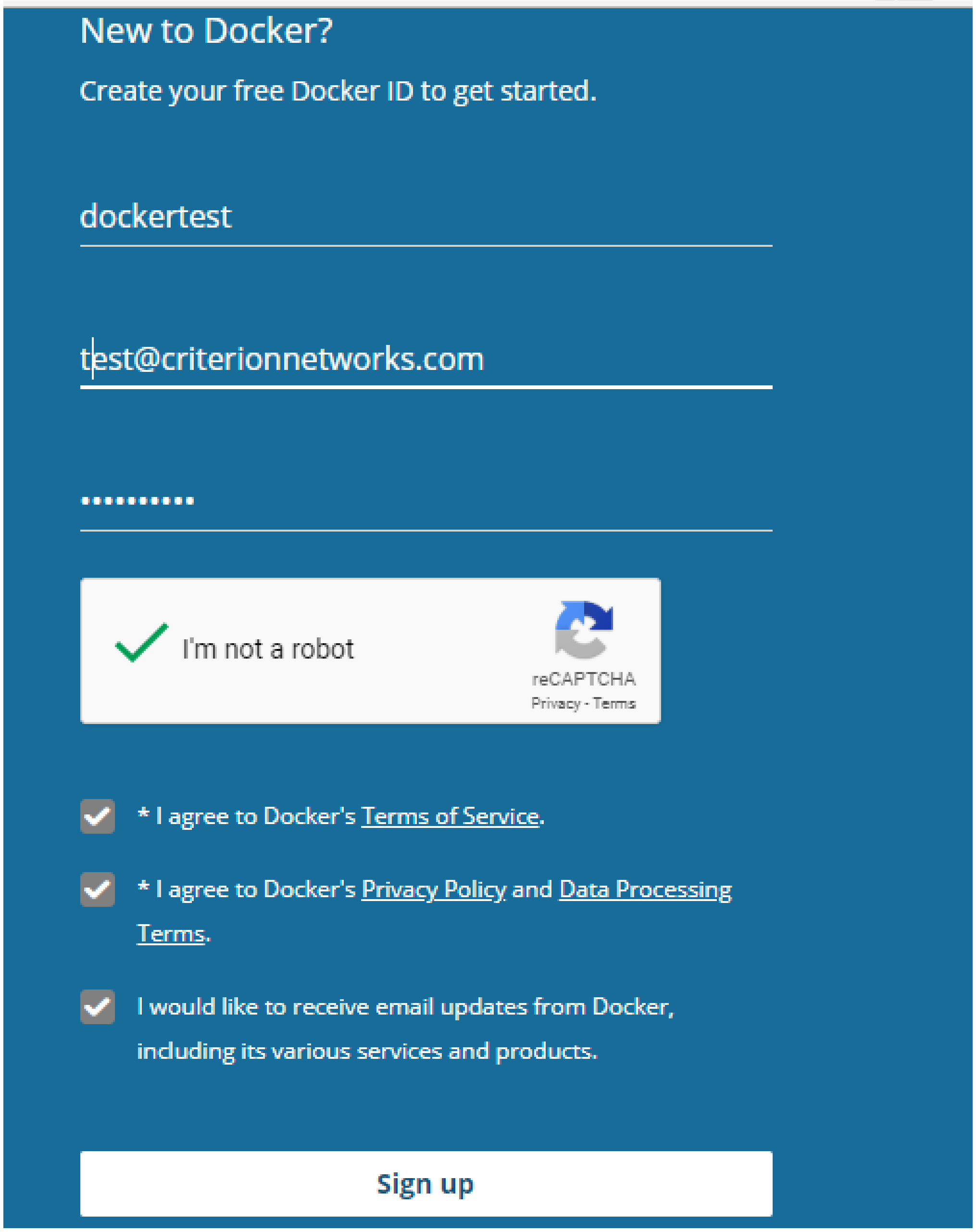
- Verify the Graph Driver details for both containers
- Observe the similarity in both the graph driver outputs
- Observe the directory structure for overlay2 driver i.e lower, diff, merged, work

Docker Part-II

Exercise-16

Sign-up for Docker ID

- Visit https://cloud.docker.com
- Sign up to get a Docker ID
- Verify you are able to login with Docker ID and password



New to Docker?

Create your free Docker ID to get started.

dockertest

test@criterionnetworks.com

••••••••••

☑ I'm not a robot

☑ * I agree to Docker's Terms of Service.

☑ * I agree to Docker's Privacy Policy and Data Processing Terms.

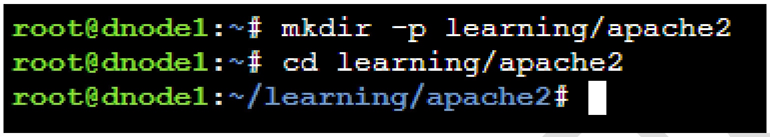
☑ I would like to receive email updates from Docker, including its various services and products.

Exercise-17

Build the Apache2 Web Server Container using DockerFile

- Create a new directory learning/apache2

```
mkdir -p learning/apache2

cd learning/apache2
```



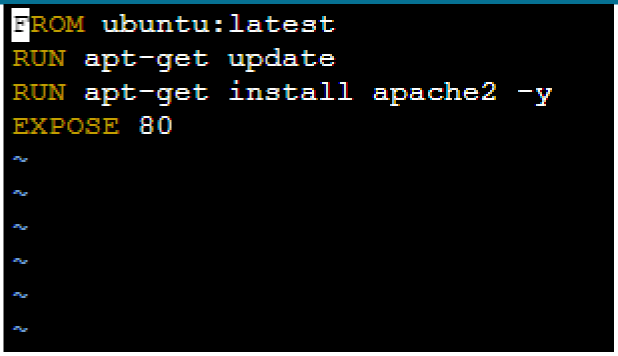
```
root@dnode1:~# mkdir -p learning/apache2
root@dnode1:~# cd learning/apache2
root@dnode1:~/learning/apache2#
```

- Create a new file named Dockerfile

```
vim Dockerfile
```

- Insert following lines in the file, save and close it

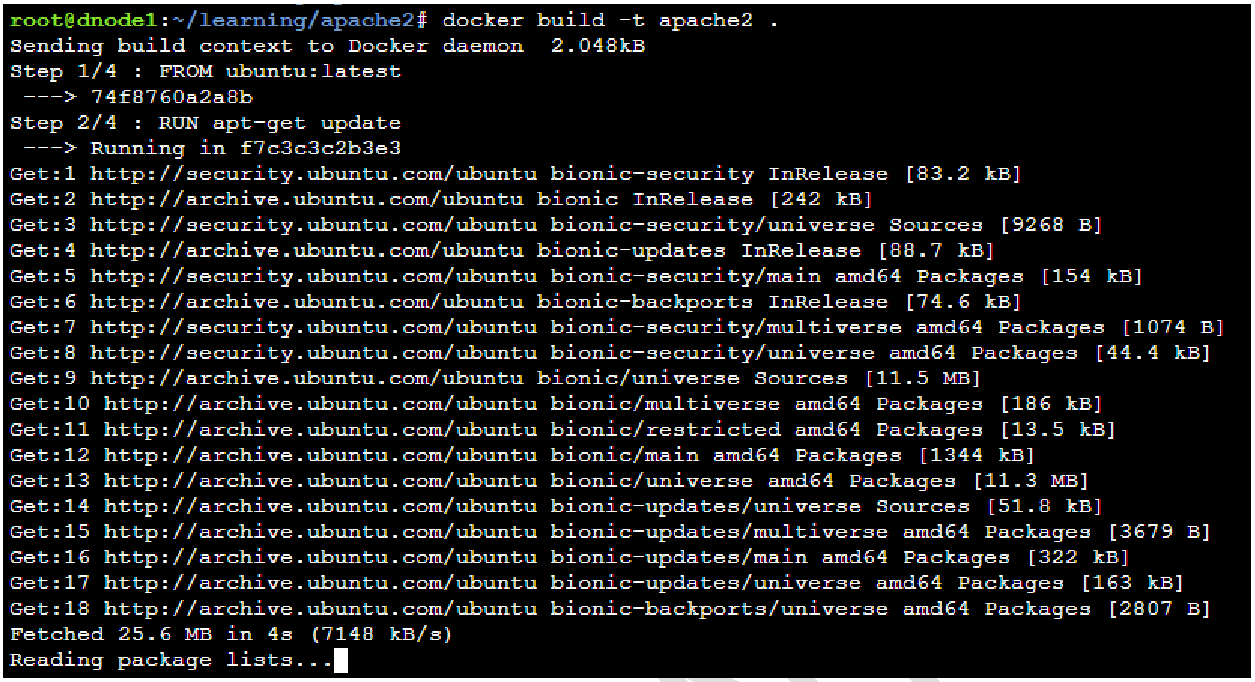
```
FROM ubuntu:14.04

RUN apt-get update

RUN apt-get install apache2 -y

EXPOSE 80
```



```
FROM ubuntu:latest
RUN apt-get update
RUN apt-get install apache2 -y
EXPOSE 80
~
~
~
~
~
~
```

- Build the docker image

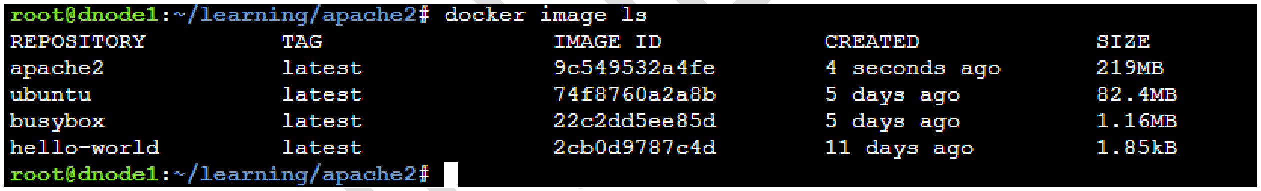
```
docker build -t apache2 .
```



```
root@dnode1:~/learning/apache2# docker build -t apache2 .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM ubuntu:latest
 ---> 74f8760a2a8b
Step 2/4 : RUN apt-get update
 ---> Running in f7c3c3c2b3e3
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security/universe Sources [9268 B]
Get:4 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [154 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:7 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [1074 B]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [44.4 kB]
Get:9 http://archive.ubuntu.com/ubuntu bionic/universe Sources [11.5 MB]
Get:10 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/universe Sources [51.8 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [3679 B]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [322 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [163 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [2807 B]
Fetched 25.6 MB in 4s (7148 kB/s)
Reading package lists...
```

- Verify the image is built

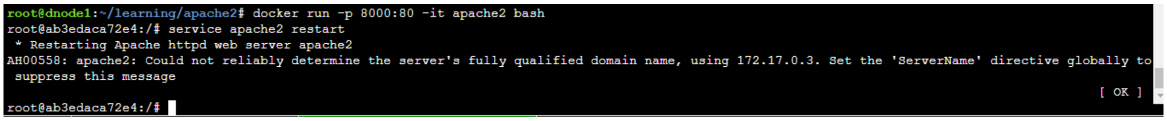
```
docker image ls
```



```
root@dnode1:~/learning/apache2# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
apache2             latest              9c549532a4fe        4 seconds ago       219MB
ubuntu              latest              74f8760a2a8b        5 days ago          82.4MB
busybox             latest              22c2dd5ee85d        5 days ago          1.16MB
hello-world         latest              2cb0d9787c4d        11 days ago         1.85kB
root@dnode1:~/learning/apache2#
```

- Run the docker Image and verify the functionality

```
docker run -p 8000:80 -it apache2 bash

service apache2 restart
```

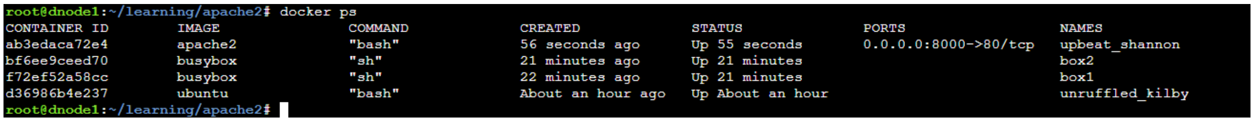


```
root@dnode1:~/learning/apache2# docker run -p 8000:80 -it apache2 bash
root@ab3edaca72e4:/# service apache2 restart
 * Restarting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to
 suppress this message
                                                                                                                                                  [ OK ]
root@ab3edaca72e4:/#
```

NOTE: Do not exit the container and open another dnode1 terminal to view the running apache2 container

- Verify the docker is running at proper port on host dnode1 (The below command has to be performed on the other donode1 terminal.)

```
docker ps
```



```
root@dnode1:~/learning/apache2# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                  NAMES
ab3edaca72e4        apache2             "bash"              56 seconds ago      Up 55 seconds       0.0.0.0:8000->80/tcp   upbeat_shannon
bf6ee9ceed70        busybox             "sh"                21 minutes ago      Up 21 minutes                              box2
f72ef52a58cc        busybox             "sh"                22 minutes ago      Up 21 minutes                              box1
d36986b4e237        ubuntu              "bash"              About an hour ago   Up About an hour                           unruffled_kilby
root@dnode1:~/learning/apache2#
```

- Get the IP address of dnode1 for interface "ens3"
- Open VNC on desktop1 machine. Open Firefox web browser
- Perform http request in the web browser

```
http://<IP-ADDR-dnode1-ens3>:8000
```

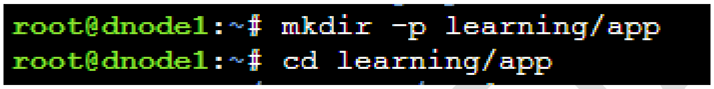
(Verify Apache2 web page is displayed)

Exercise-18

Build an application using Python Run-time

- Create a new directory learning/app

```
mkdir -p learning/app

cd learning/app
```



```
root@dnode1:~# mkdir -p learning/app
root@dnode1:~# cd learning/app
```

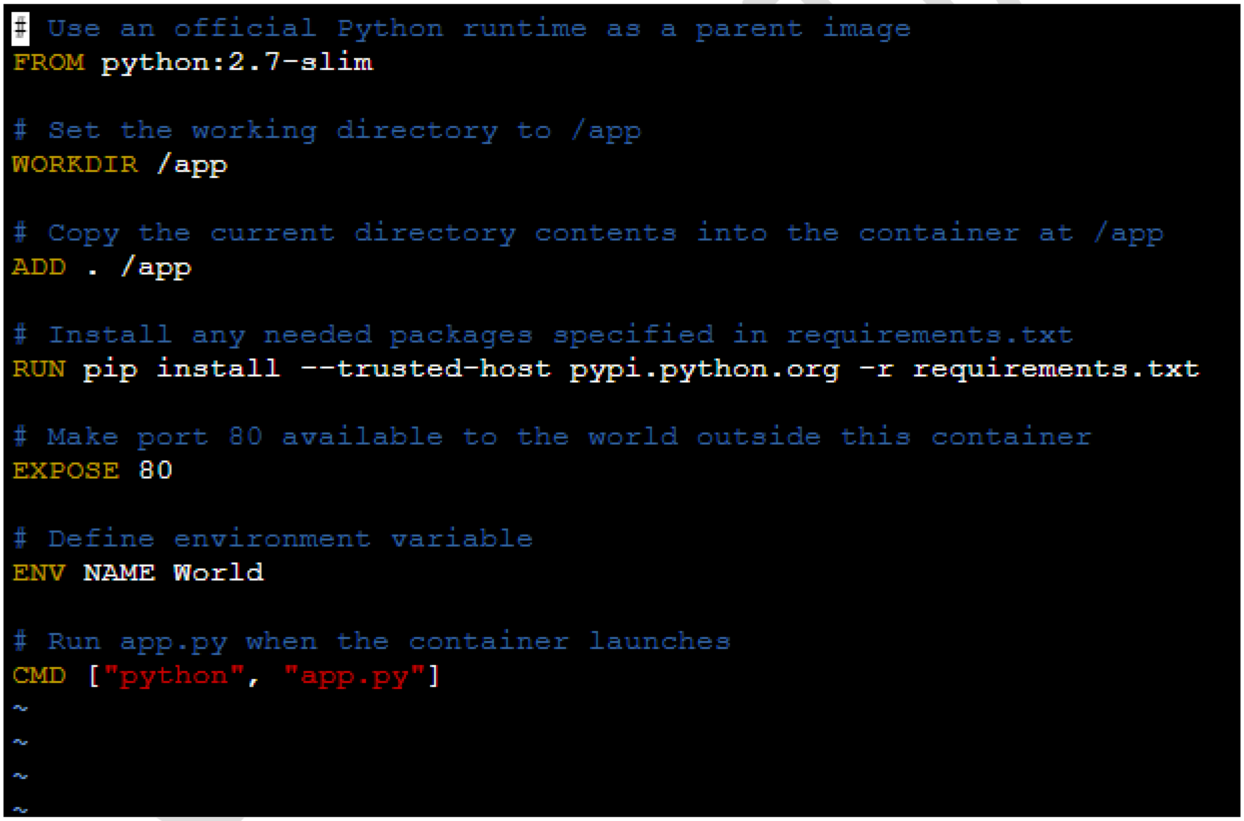
- Create a new file named Dockerfile

```
vim Dockerfile
```

- Insert following lines in the file, save and close it

```
# Use an official Python runtime as a parent image

FROM python:2.7-slim



# Set the working directory to /app

WORKDIR /app



# Copy the current directory contents into the container at /app

ADD . /app



# Install any needed packages specified in requirements.txt

RUN pip install --trusted-host pypi.python.org -r requirements.txt
```

```
# Make port 80 available to the world outside this container

EXPOSE 80



# Define environment variable

ENV NAME World



# Run app.py when the container launches

CMD ["python", "app.py"]
```

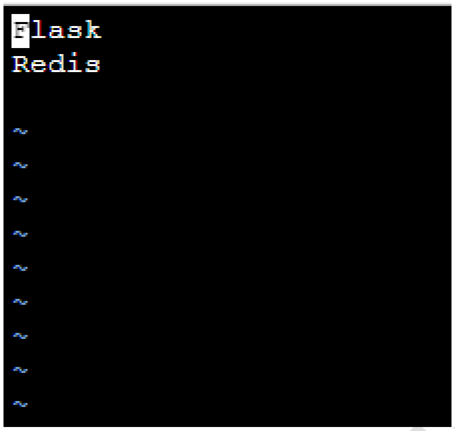


```
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
~
~
~
~
```

- Create a file named requirements.txt

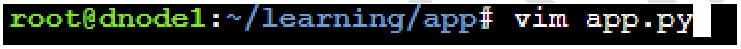
```
vim requirements.txt



Flask
```

```
Redis
```



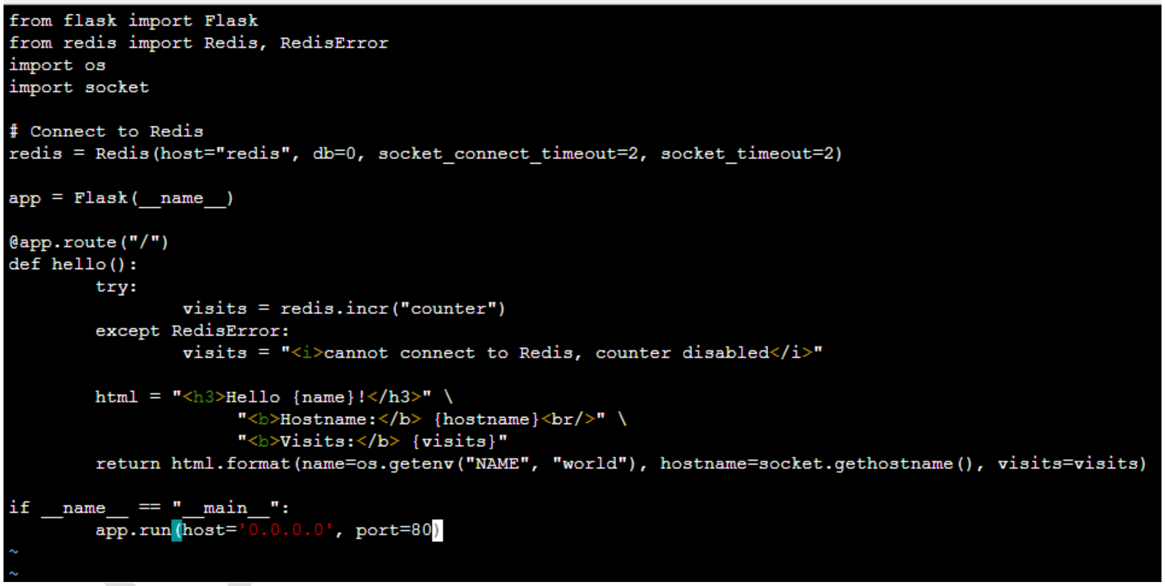
```
Flask
Redis

~
~
~
~
~
~
~
~
~
```

- Create a file named app.py



```
root@dnode1:~/learning/app# vim app.py
```

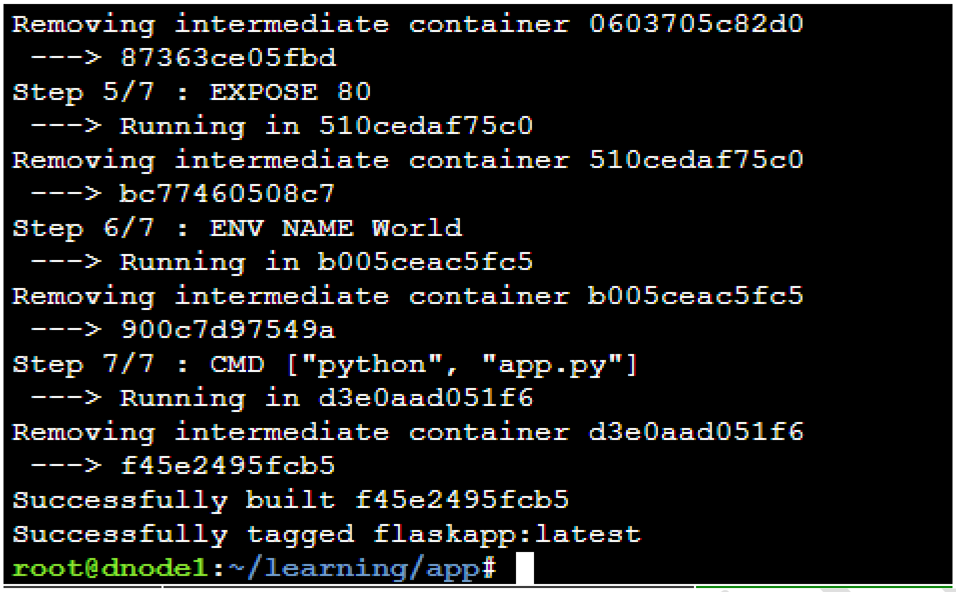
```
from flask import Flask
from redis import Redis, RedisError
import os
import socket

# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

app = Flask(__name__)

@app.route("/")
def hello():
        try:
                visits = redis.incr("counter")
        except RedisError:
                visits = "<i>cannot connect to Redis, counter disabled</i>"


        html = "<h3>Hello {name}!</h3>" \

                     "<b>Hostname:</b> {hostname}<br/>" \

                     "<b>Visits:</b> {visits}"

        return html.format(name=os.getenv("NAME", "world"), hostname=socket.gethostname(), visits=visits)


if __name__ == "__main__":

        app.run(host='0.0.0.0', port=80)
```



```
from flask import Flask
from redis import Redis, RedisError
import os
import socket

# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

app = Flask(__name__)

@app.route("/")
def hello():
        try:
                visits = redis.incr("counter")
        except RedisError:
                visits = "<i>cannot connect to Redis, counter disabled</i>"

        html = "<h3>Hello {name}!</h3>" \
                     "<b>Hostname:</b> {hostname}<br/>" \
                     "<b>Visits:</b> {visits}"
        return html.format(name=os.getenv("NAME", "world"), hostname=socket.gethostname(), visits=visits)

if __name__ == "__main__":
        app.run(host='0.0.0.0', port=80)
~
~
```

- Build the docker image

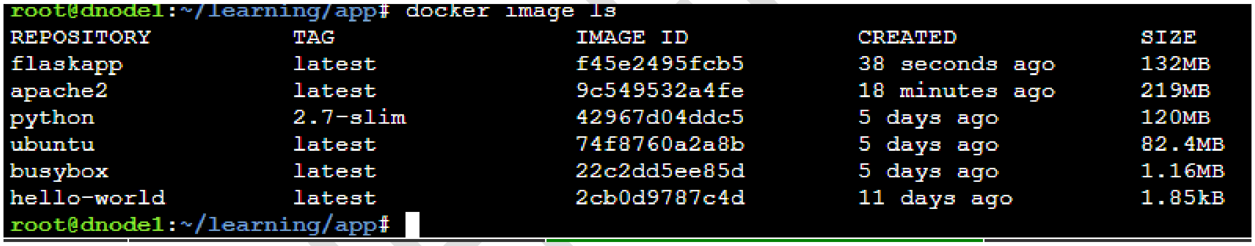
```
docker build -t flaskapp .
```



```
Removing intermediate container 0603705c82d0
 ---> 87363ce05fbd
Step 5/7 : EXPOSE 80
 ---> Running in 510cedaf75c0
Removing intermediate container 510cedaf75c0
 ---> bc77460508c7
Step 6/7 : ENV NAME World
 ---> Running in b005ceac5fc5
Removing intermediate container b005ceac5fc5
 ---> 900c7d97549a
Step 7/7 : CMD ["python", "app.py"]
 ---> Running in d3e0aad051f6
Removing intermediate container d3e0aad051f6
 ---> f45e2495fcb5
Successfully built f45e2495fcb5
Successfully tagged flaskapp:latest
root@dnode1:~/learning/app#
```

- Verify the image is built

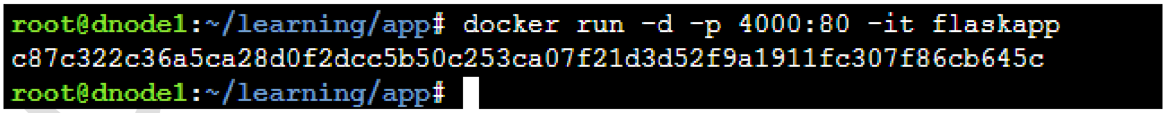
```
docker image ls
```



```
root@dnode1:~/learning/app# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
flaskapp            latest              f45e2495fcb5        38 seconds ago      132MB
apache2             latest              9c549532a4fe        18 minutes ago      219MB
python              2.7-slim            42967d04ddc5        5 days ago          120MB
ubuntu              latest              74f8760a2a8b        5 days ago          82.4MB
busybox             latest              22c2dd5ee85d        5 days ago          1.16MB
hello-world         latest              2cb0d9787c4d        11 days ago         1.85kB
root@dnode1:~/learning/app#
```

- Run the docker Image in detached mode and verify the functionality

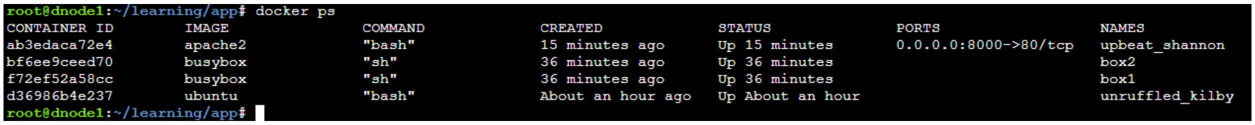
```
docker run -d -p 4000:80 -it flaskapp
```



```
root@dnode1:~/learning/app# docker run -d -p 4000:80 -it flaskapp
c87c322c36a5ca28d0f2dcc5b50c253ca07f21d3d52f9a1911fc307f86cb645c
root@dnode1:~/learning/app#
```

- Verify the docker is running at proper port on host dnode1

```
docker ps
```



```
root@dnode1:~/learning/app# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                  NAMES
ab3edaca72e4        apache2             "bash"              15 minutes ago      Up 15 minutes       0.0.0.0:8000->80/tcp   upbeat_shannon
bf6ee9ceed70        busybox             "sh"                36 minutes ago      Up 36 minutes                              box2
f72ef52a58cc        busybox             "sh"                36 minutes ago      Up 36 minutes                              box1
d36986b4e237        ubuntu              "bash"              About an hour ago   Up About an hour                           unruffled_kilby
root@dnode1:~/learning/app#
```

- Get the IP address of dnode1 for interface "ens3"

- Open VNC on desktop1 machine. Open Firefox web browser
- Perform http request in the web browser

```
http://<IP-ADDR-dnode1>:4000
```

(Verify Flask web page and Redis Error is displayed)

Exercise-19

Push the Docker Images into Docker Hub Registry

- Login into docker hub using Docker ID

```
docker login
```

(Provide your docker username and password to login)

- Tag the image to be pushed into Public Repository
- Format: docker tag imagename username/repository:tag)

NOTE: Replace nareshthukkani with your docker username in command below

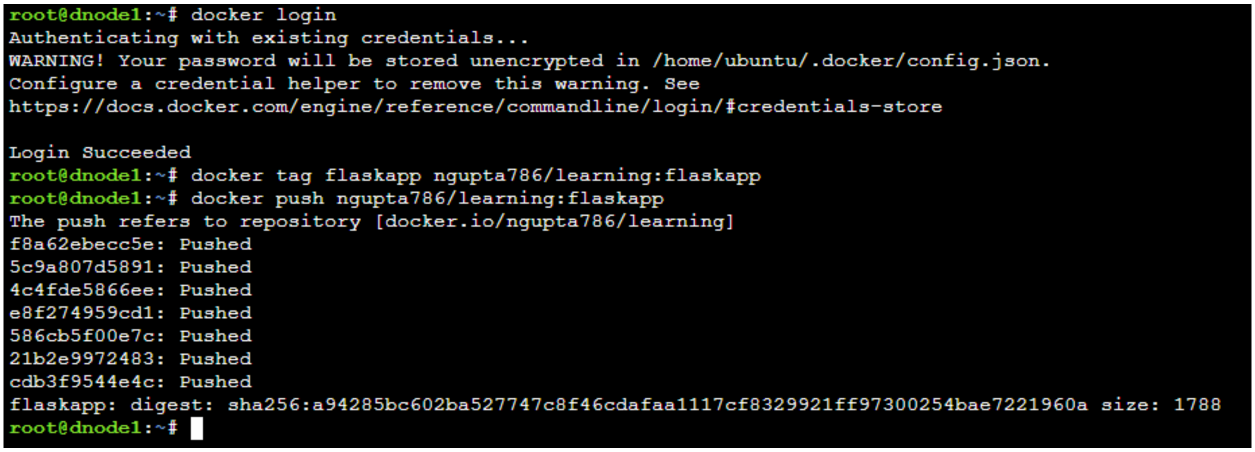
Example:

```
docker tag flaskapp nareshthukkani/learning:flaskapp
```

- Verify docker images. Displays new tag image with same Image ID
- Push the image into Docker Hub

NOTE: Replace nareshthukkani with your username in command below

```
docker push nareshthukkani/learning:flaskapp
```



```
root@dnode1:~# docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@dnode1:~# docker tag flaskapp ngupta786/learning:flaskapp
root@dnode1:~# docker push ngupta786/learning:flaskapp
The push refers to repository [docker.io/ngupta786/learning]
f8a62ebecc5e: Pushed
5c9a807d5891: Pushed
4c4fde5866ee: Pushed
e8f274959cd1: Pushed
586cb5f00e7c: Pushed
21b2e9972483: Pushed
cdb3f9544e4c: Pushed
flaskapp: digest: sha256:a94285bc602ba527747c8f46cdafaa1117cf8329921ff97300254bae7221960a size: 1788
root@dnode1:~#
```

- Login into docker cloud account. Click on Repositories to check the image pushed
- Now, please push apache2 image as well in the above way

Exercise-20

Docker Service using docker-compose

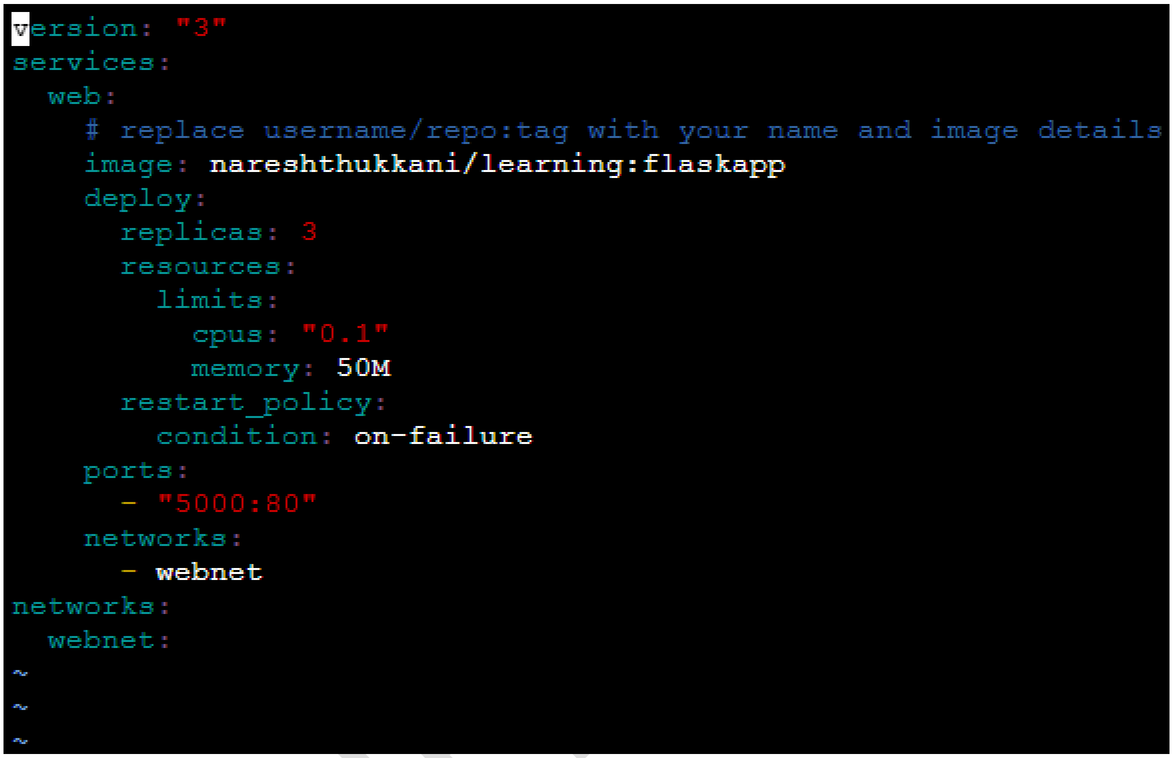
- Perform the below commands on "dnode1" node. We will bring up single node dswarm cluster.
- Create a file named docker-compose.yml

```
vim docker-compose.yml
```

```
root@dnode1:~# vim docker-compose.yml
root@dnode1:~# 
```

- Insert the below lines [replace the "nareshthukkani" with your docker username]

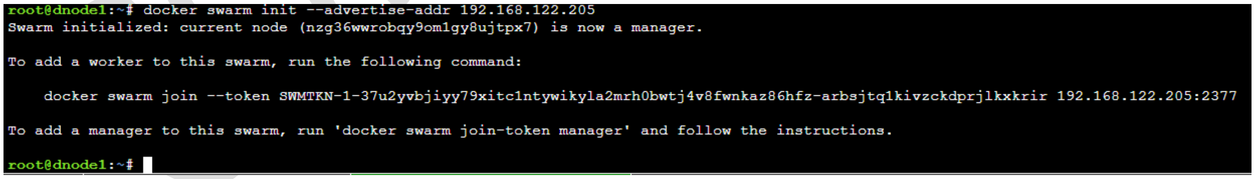
```
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: nareshthukkani/learning:flaskapp
    deploy:
      replicas: 3
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "5000:80"
    networks:
      - webnet

networks:

  webnet:
```



```
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: nareshthukkani/learning:flaskapp
    deploy:
      replicas: 3
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "5000:80"
    networks:
      - webnet
networks:
  webnet:
~
~
~
```

- Initialize the Docker Swarm

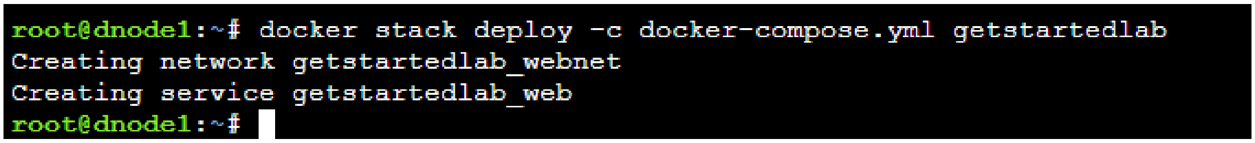
```
    docker swarm init --advertise-addr 192.168.122.205
```



```
root@dnode1:~# docker swarm init --advertise-addr 192.168.122.205
Swarm initialized: current node (nzg36wwrobqy9om1gy8ujtpx7) is now a manager.

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

    docker swarm join --token SWMTKN-1-37u2yvbjiyy79xitc1ntywikyla2mrh0bwtj4v8fwnkaz86hfz-arbsjtq1kivzckdprjlkxkrir 192.168.122.205:2377

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

root@dnode1:~#
```

- Deploy the stack

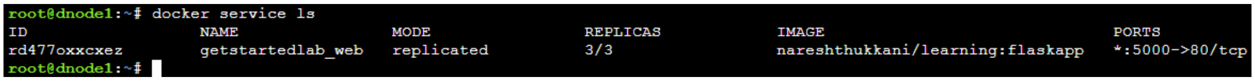
```
docker stack deploy -c docker-compose.yml getstartedlab
```



```
root@dnode1:~# docker stack deploy -c docker-compose.yml getstartedlab
Creating network getstartedlab_webnet
Creating service getstartedlab_web
root@dnode1:~#
```

- Verify the docker service is launched

```
docker service ls
```

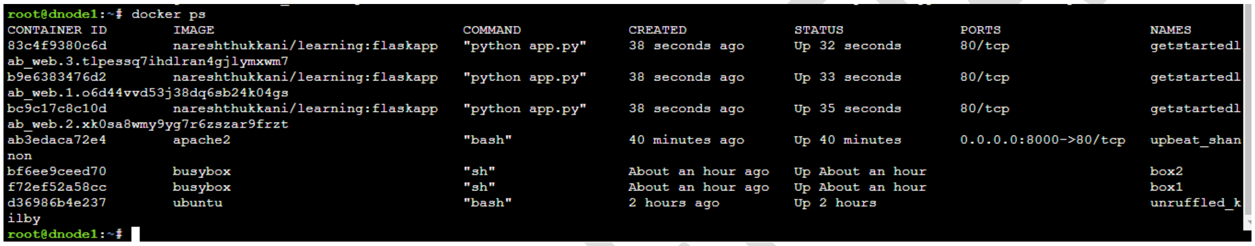


```
root@dnode1:~# docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE                                PORTS
rd477oxxcxez        getstartedlab_web   replicated          3/3                 nareshthukkani/learning:flaskapp     *:5000->80/tcp
root@dnode1:~#
```

NOTE: Fetch the port number exposed in the displayed service which will be used with the advertised address used in the init command.

- Verify the new containers deployed

```
docker ps
```



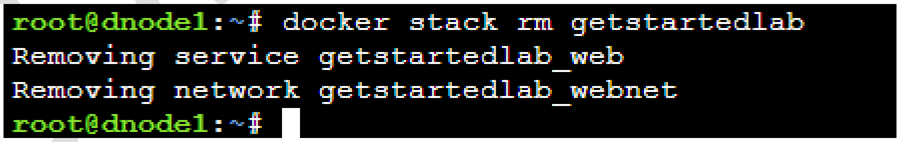
```
root@dnode1:~# docker ps
CONTAINER ID        IMAGE                              COMMAND             CREATED             STATUS              PORTS                  NAMES
83c4f9380c6d        nareshthukkani/learning:flaskapp   "python app.py"     38 seconds ago      Up 32 seconds       80/tcp                 getstartedl
ab_web.3.tlpessq7ihdlran4gjlymxwm7
b9e6383476d2        nareshthukkani/learning:flaskapp   "python app.py"     38 seconds ago      Up 33 seconds       80/tcp                 getstartedl
ab_web.1.o6d44vvd53j38dq6sb24k04gs
bc9c17c8c10d        nareshthukkani/learning:flaskapp   "python app.py"     38 seconds ago      Up 35 seconds       80/tcp                 getstartedl
ab_web.2.xk0sa8wmy9yg7r6zszar9frzt
ab3edaca72e4        apache2                            "bash"              40 minutes ago      Up 40 minutes       0.0.0.0:8000->80/tcp   upbeat_shan
non
bf6ee9ceed70        busybox                            "sh"                About an hour ago   Up About an hour                           box2
f72ef52a58cc        busybox                            "sh"                About an hour ago   Up About an hour                           box1
d36986b4e237        ubuntu                             "bash"              2 hours ago         Up 2 hours                                 unruffled_k
ilby
root@dnode1:~#
```

- Verify from VNC, browse the

```
http://<ip-address-advertised>:<port-fetched-in-service-display-command>
```

- Web Browser repeatedly to this service gives different web page every time
- Change the replicas value in docker-compose.yml to 5 and deploy the stack again
- Verify 5 containers are deployed
- Take down the app

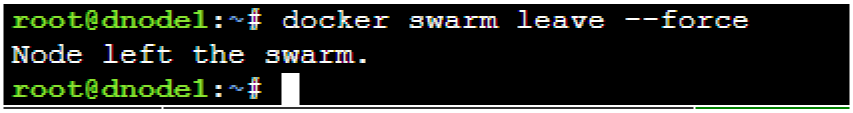
```
docker stack rm getstartedlab
```



```
root@dnode1:~# docker stack rm getstartedlab
Removing service getstartedlab_web
Removing network getstartedlab_webnet
root@dnode1:~#
```

- Take down the swarm

```
docker swarm leave --force
```



```
root@dnode1:~# docker swarm leave --force
Node left the swarm.
root@dnode1:~#
```

Exercise-21

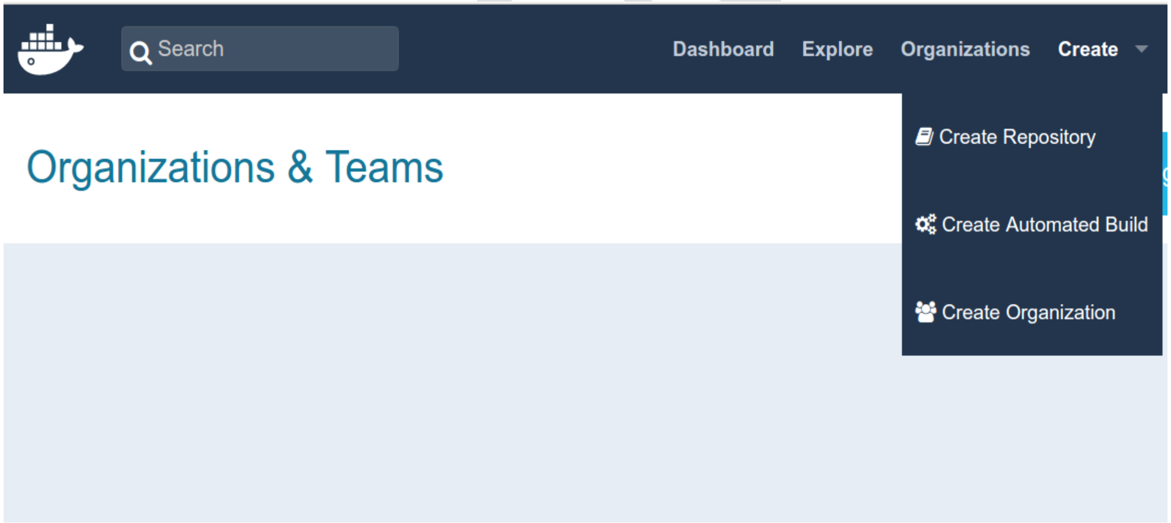
Docker with DevOps

1.Docker Github

- In this exercise we will use github , so you need to have a github account.You can use your existing github account or else create a new one.
- Sign-in to github and open following URL https:github.com/rahulhada/docker-training
- Click on Fork , this will create your own copy in your account
- View all the files present in the keystone and Glance directories
- Now go to http://hub.docker.com and create an account

2.Build Docker Repo

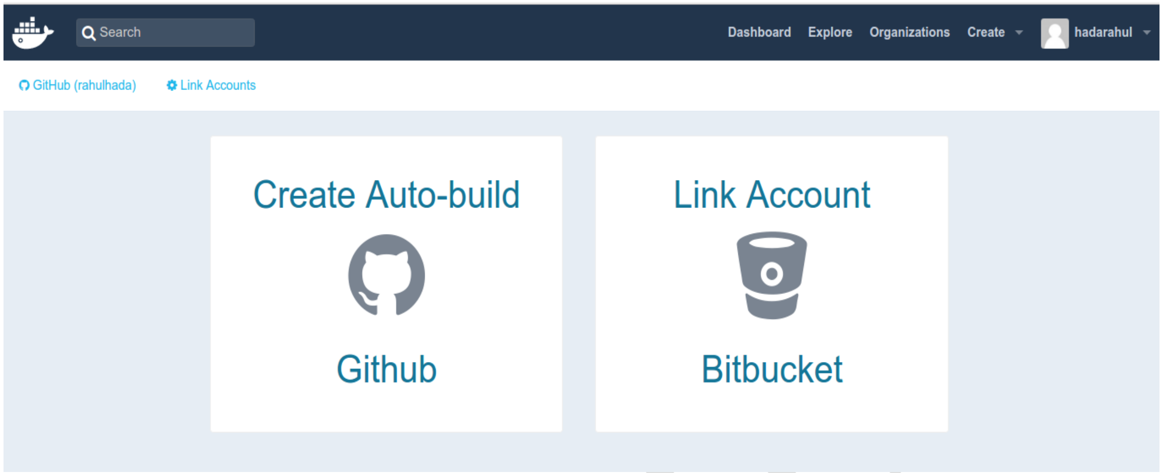
- Now link the github docker-training repo with docker-hub
- Click on Create Automatic Build



Organizations & Teams

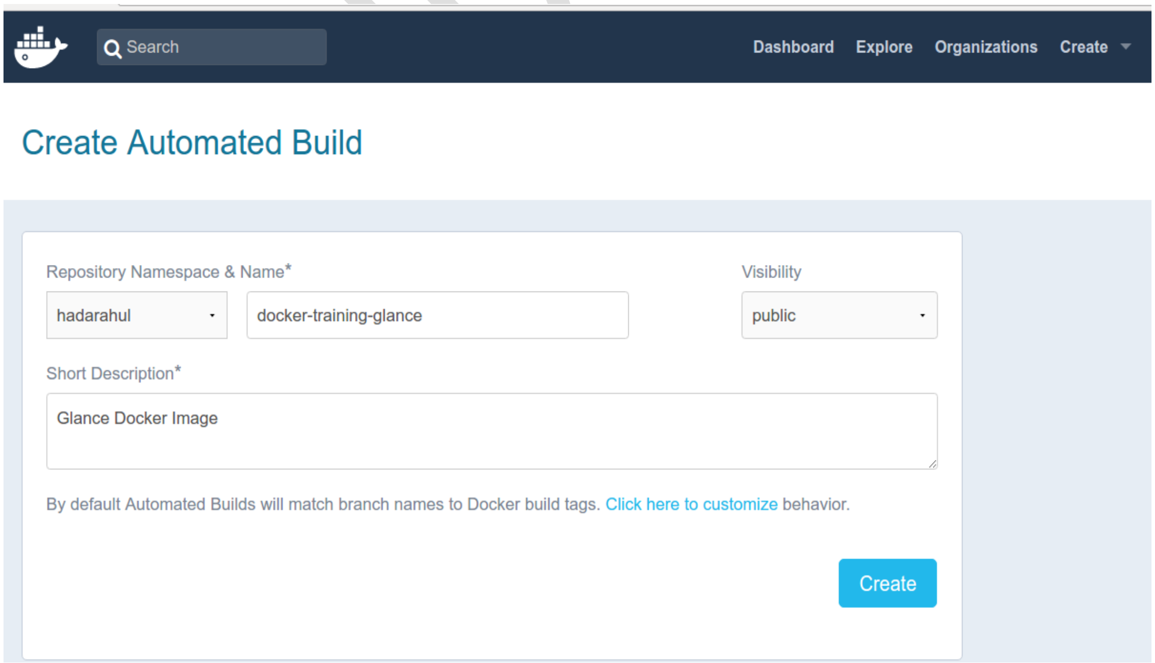
3.Auto-Build for Git

- Create auto-build for github repo



4.Select Docker Training Repo

- Select docker-training repo and give name to your docker image. We can name it as docker-training-glance for glance image and docker-training-keystone for keystone image.
- Click on Create button



Create Automated Build

Repository Namespace & Name*: hadarahul / docker-training-glance

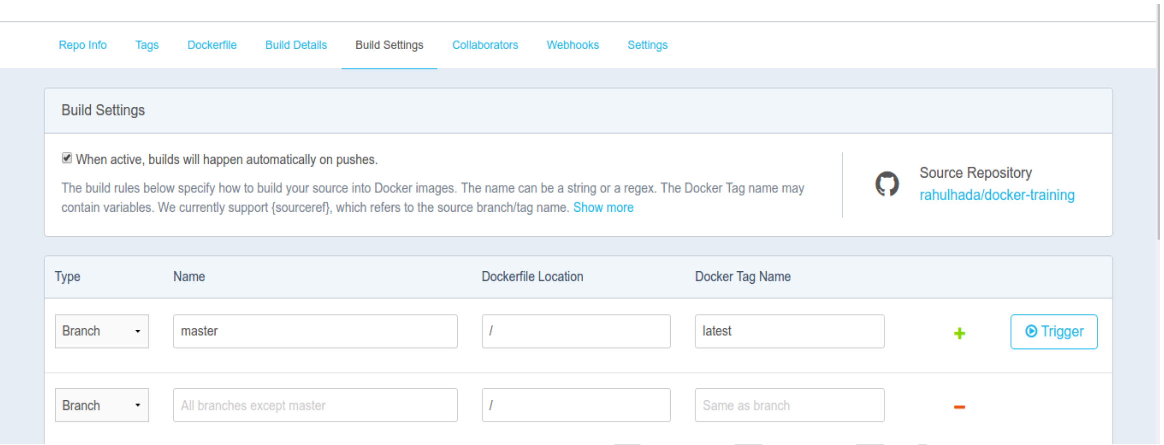
Visibility: public

Short Description*: Glance Docker Image

By default Automated Builds will match branch names to Docker build tags. Click here to customize behavior.

5.Build Setting

- After that click on Build Settings tab



Build Settings

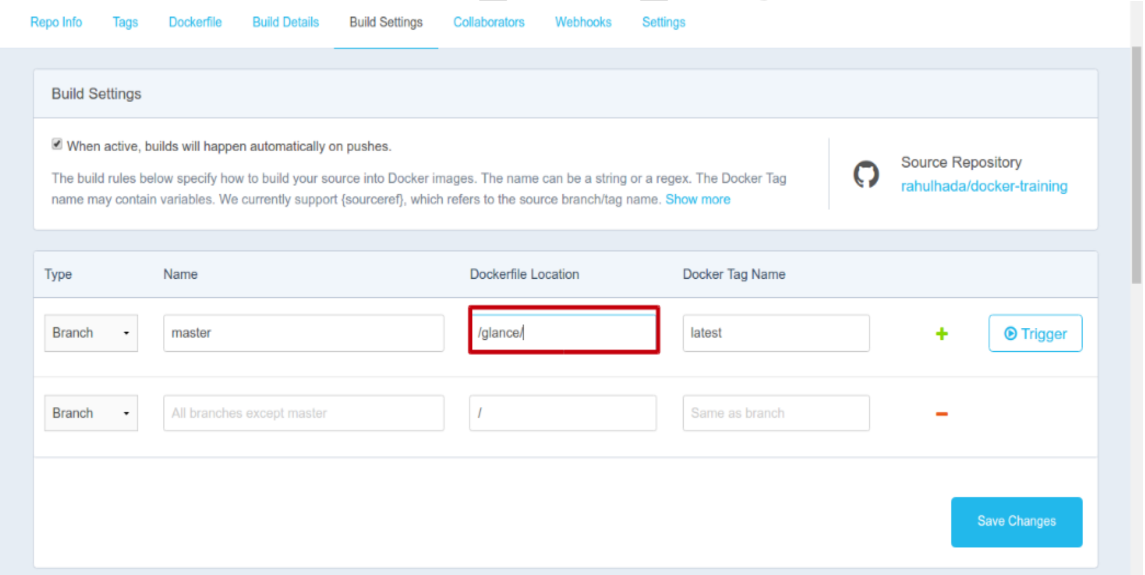
☑ When active, builds will happen automatically on pushes.

The build rules below specify how to build your source into Docker images. The name can be a string or a regex. The Docker Tag name may contain variables. We currently support {sourceref}, which refers to the source branch/tag name. Show more

Source Repository
rahulhada/docker-training

Type	Name	Dockerfile Location	Docker Tag Name
Branch	master	/	latest
Branch	All branches except master	/	Same as branch

- Here we need to give the location of Dockerfile directory. The Dockerfile located in glance and keystone directories.



Build Settings

☑ When active, builds will happen automatically on pushes.

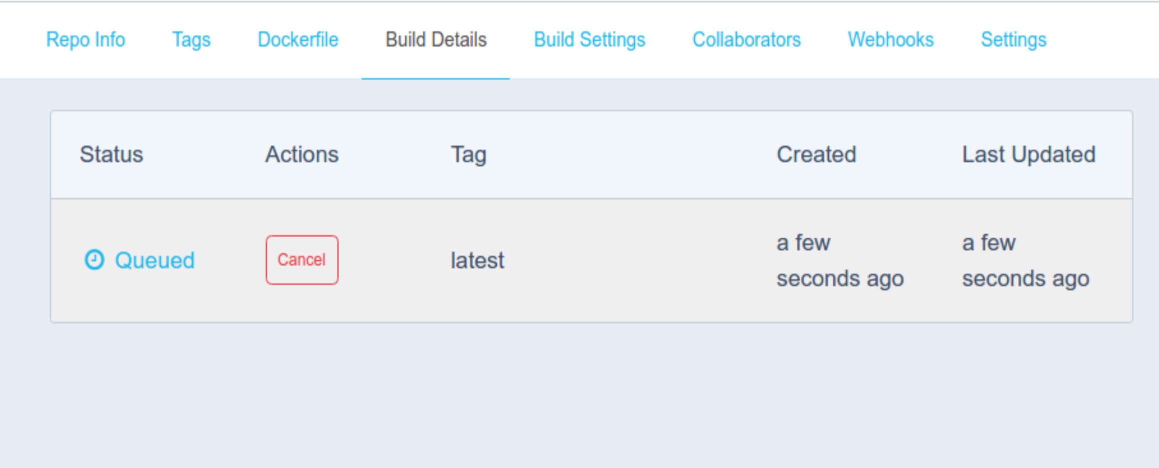
The build rules below specify how to build your source into Docker images. The name can be a string or a regex. The Docker Tag name may contain variables. We currently support {sourceref}, which refers to the source branch/tag name. Show more

Source Repository
rahulhada/docker-training

Type	Name	Dockerfile Location	Docker Tag Name
Branch	master	/glance/	latest
Branch	All branches except master	/	Same as branch

6.Verify Build Status

- Now click on **Save Changes** and then **trigger** to start building. Click on build details tab to see the build status.



Status	Actions	Tag	Created	Last Updated
Queued	Cancel	latest	a few seconds ago	a few seconds ago

- This will take few minutes to build and wait till the status of build is successful. Repeat the steps from 2 to 6 to build keystone image.

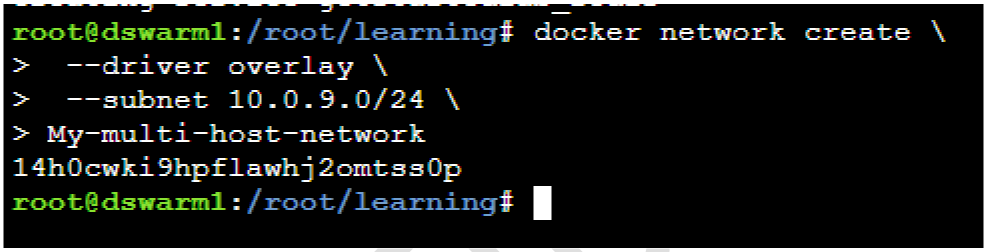
Exercise-22

To Spawn the Mysql Database, Keystone and Glance Container

On dswarm1 node

- Create an overlay network to connect hosted container together

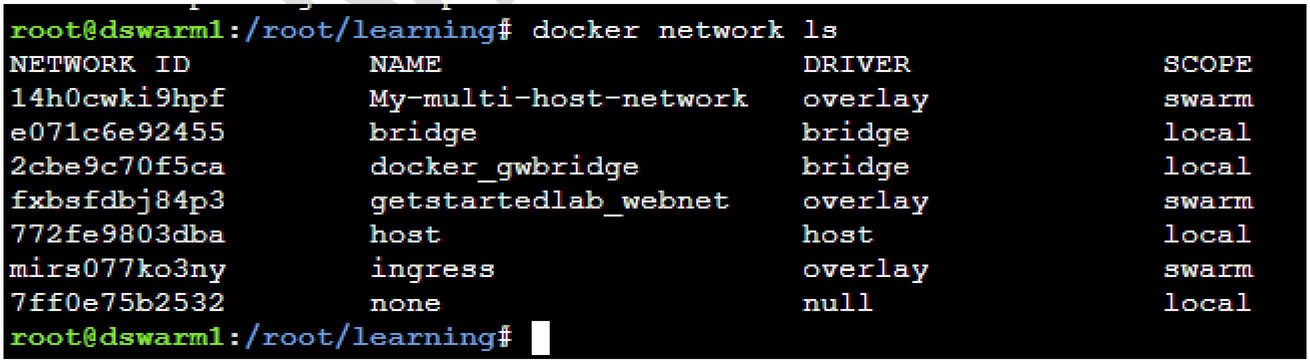
```
docker network create \

 --driver overlay \

 --subnet 10.0.9.0/24 \

My-multi-host-network
```



```
root@dswarm1:/root/learning# docker network create \
>  --driver overlay \
>  --subnet 10.0.9.0/24 \
> My-multi-host-network
14h0cwki9hpflawhj2omtss0p
root@dswarm1:/root/learning#
```

- To verify the created network

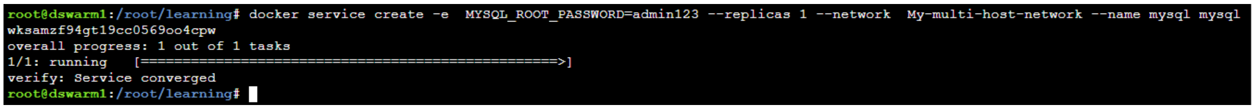
```
docker network ls
```



```
root@dswarm1:/root/learning# docker network ls
NETWORK ID          NAME                    DRIVER              SCOPE
14h0cwki9hpf        My-multi-host-network   overlay             swarm
e071c6e92455        bridge                  bridge              local
2cbe9c70f5ca        docker_gwbridge         bridge              local
fxbsfdbj84p3        getstartedlab_webnet    overlay             swarm
772fe9803dba        host                    host                local
mirs077ko3ny        ingress                 overlay             swarm
7ff0e75b2532        none                    null                local
root@dswarm1:/root/learning#
```

- Create a docker service for mysql

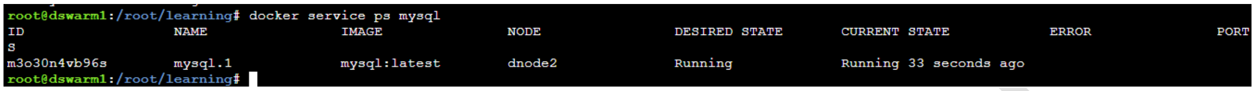
```
docker  service create -e  MYSQL_ROOT_PASSWORD=admin123 --replicas 1 --network
getstartedlab_webnet --name mysql mysql:5.7.23
```



```
root@dswarm1:/root/learning# docker service create -e  MYSQL_ROOT_PASSWORD=admin123 --replicas 1 --network  My-multi-host-network --name mysql mysql
wksamzf94gt19cc0569oo4cpw
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
root@dswarm1:/root/learning# 
```

- Verify mysql service is running and observe the node it is spawned

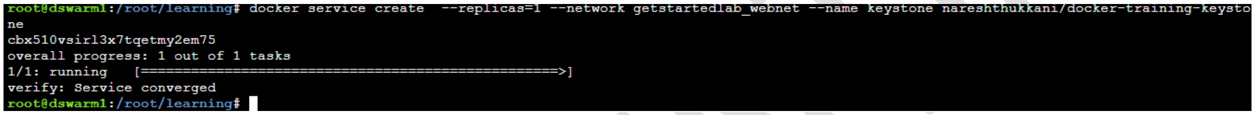
```
docker service ps mysql
```



```
root@dswarm1:/root/learning# docker service ps mysql
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE            ERROR               PORT
S
m3o30n4vb96s        mysql.1             mysql:latest        dnode2              Running             Running 33 seconds ago
root@dswarm1:/root/learning# 
```

- Create a docker service for keystone (replace "nareshthukkani" with your docker username)

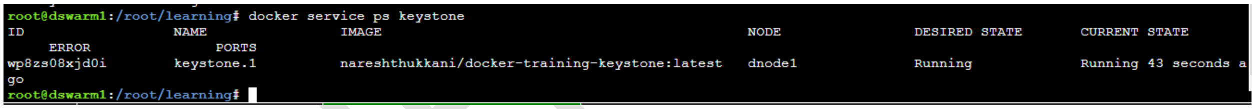
```
docker service create  --replicas=1 --network getstartedlab_webnet --name keys
tone nareshthukkani/docker-training-keystone
```



```
root@dswarm1:/root/learning# docker service create  --replicas=1 --network getstartedlab_webnet --name keystone nareshthukkani/docker-training-keysto
ne
cbx510vsirl3x7tqetmy2em75
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
root@dswarm1:/root/learning# 
```

- Verify keystone service is running and observe node it is spawned

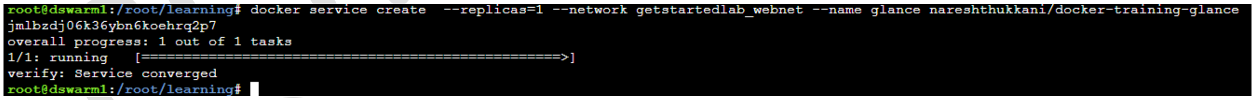
```
docker service ps keystone
```



```
root@dswarm1:/root/learning# docker service ps keystone
ID                  NAME                IMAGE                                            NODE                DESIRED STATE       CURRENT STATE
     ERROR               PORTS
wp8zs08xjd0i        keystone.1          nareshthukkani/docker-training-keystone:latest   dnode1              Running             Running 43 seconds a
go
root@dswarm1:/root/learning# 
```

- Create a docker service for glance (replace "nareshthukkani" with your docker username)

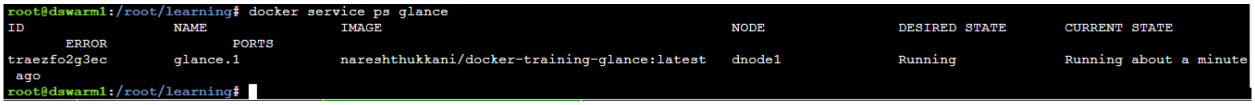
```
docker service create  --replicas=1 --network getstartedlab_webnet --name glan
ce nareshthukkani/docker-training-glance
```



```
root@dswarm1:/root/learning# docker service create  --replicas=1 --network getstartedlab_webnet --name glance nareshthukkani/docker-training-glance
jmlbzdj06k36ybn6koehrq2p7
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
root@dswarm1:/root/learning# 
```

- Verify glance service is running and observe node it is spawned

```
docker service ps glance
```



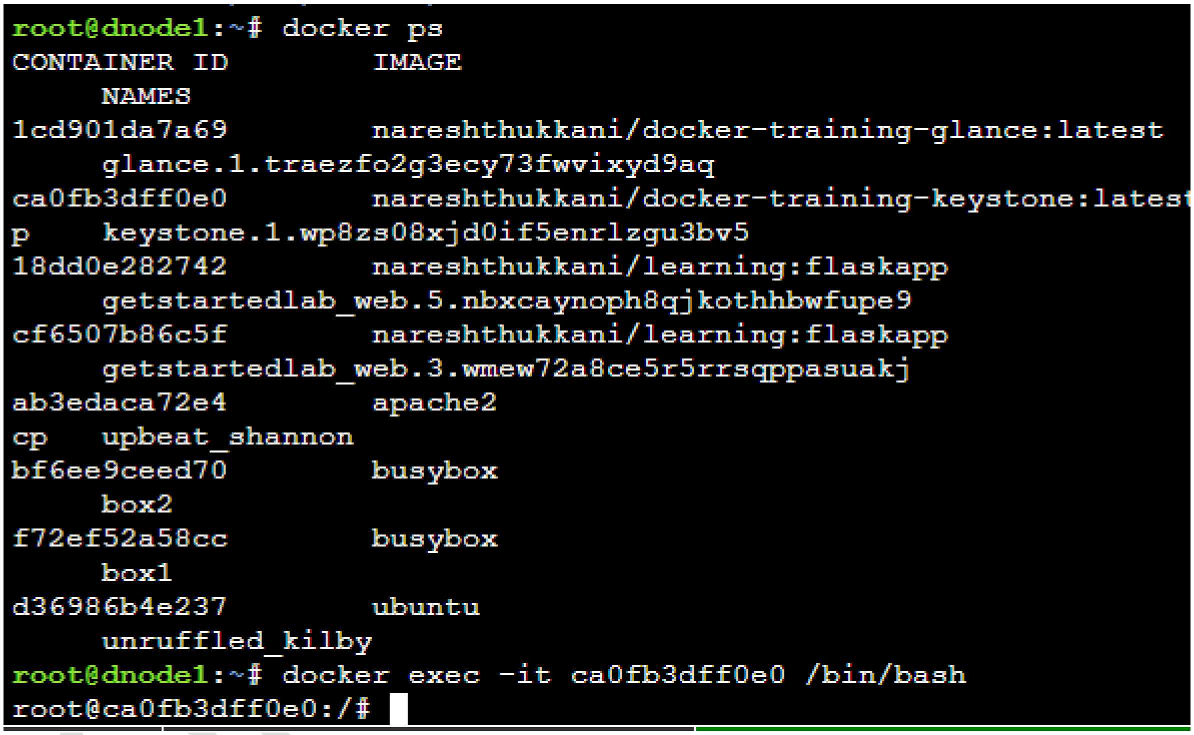
```
root@dswarm1:/root/learning# docker service ps glance
ID                  NAME                IMAGE                                          NODE                DESIRED STATE       CURRENT STATE
       ERROR               PORTS
traezfo2g3ec        glance.1            nareshthukkani/docker-training-glance:latest   dnode1              Running             Running about a minute
 ago
root@dswarm1:/root/learning# 
```

Exercise-23

To configure services and verify functionality

- Fetch the container ID of keystone container and login to the container

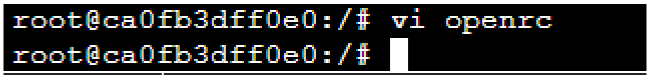
```
docker ps

docker exec -it <KEYSTONE-CONTAINERID> /bin/bash
```



```
root@dnode1:~# docker ps
CONTAINER ID        IMAGE
     NAMES
1cd901da7a69        nareshthukkani/docker-training-glance:latest
     glance.1.traezfo2g3ecy73fwvixyd9aq
ca0fb3dff0e0        nareshthukkani/docker-training-keystone:latest
p    keystone.1.wp8zs08xjd0if5enrlzgu3bv5
18dd0e282742        nareshthukkani/learning:flaskapp
     getstartedlab_web.5.nbxcaynoph8qjkothhbwfupe9
cf6507b86c5f        nareshthukkani/learning:flaskapp
     getstartedlab_web.3.wmew72a8ce5r5rrsqppasuakj
ab3edaca72e4        apache2
cp   upbeat_shannon
bf6ee9ceed70        busybox
     box2
f72ef52a58cc        busybox
     box1
d36986b4e237        ubuntu
     unruffled_kilby
root@dnode1:~# docker exec -it ca0fb3dff0e0 /bin/bash
root@ca0fb3dff0e0:/#
```

- Create openrc file, paste the contents and save the file

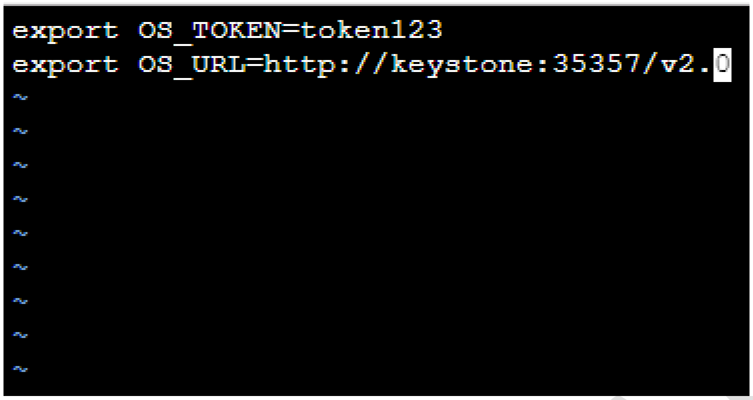
```
vi openrc
```



```
root@ca0fb3dff0e0:/# vi openrc
root@ca0fb3dff0e0:/#
```

- Insert the following lines

```
export OS_TOKEN=token123

export OS_URL=http://keystone:35357/v2.0
```



```
export OS_TOKEN=token123
export OS_URL=http://keystone:35357/v2.0
~
~
~
~
~
~
~
~
~
```

- Source openrc file

```
source openrc
```

- Create an entry for identity service

```
openstack service create --name keystone --description "OpenStack Identity" identity
```

- Create Identity service endpoints

```
openstack endpoint create \

--publicurl http://keystone:5000/v2.0 \

--internalurl http://keystone:5000/v2.0 \

--adminurl http://keystone:35357/v2.0 \

--region RegionOne \

identity
```

- Create admin project

```
openstack project create admin
```

- Create admin user

```
openstack user create --password admin admin
```

- Create role admin

```
openstack role create admin
```

- Assign role admin to user admin in admin project

```
openstack role add --project admin --user admin admin
```

- Create project with name service

```
openstack project create --description "Service Project" service
```

- Create user with name glance

```
openstack user create --password  GLANCE_PASS glance
```

- Assign admin role to glance user in service project

```
openstack role add --project service --user glance admin
```

- Create service with name glance

```
openstack service create --name glance --description "OpenStack Image service"
image
```

- Create glance service endpoints

```
openstack endpoint create \

--publicurl http://glance:9292 \

--internalurl http://glance:9292 \

--adminurl http://glance:9292 \

--region RegionOne \

image
```

- Download an cirros OS image and upload into glance image database
- Login into keystone container and create temporary folder

```
mkdir /tmp/images
```

- Get the cirros image and store it in the temporary folder

```
wget -P /tmp/images http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64
-disk.img
```

- Create environment variable file with name openrc_admin1, paste the contents and save it.

```
vi openrc_admin1
```

- Insert the below lines

```
export OS_PROJECT_DOMAIN_ID=default

export OS_USER_DOMAIN_ID=default

export OS_PROJECT_NAME=admin

export OS_TENANT_NAME=admin

export OS_USERNAME=admin

export OS_PASSWORD=admin

export OS_AUTH_URL=http://keystone:35357/v3
```

- Source the openrc_admin1 file

```
source openrc_admin1
```

- Upload image into glance database with name cirros123

```
glance image-create --debug --disk-format qcow2 --container-format bare --file
/tmp/images/cirros-0.3.4-x86_64-disk.img --name cirros123
```

- Confirm upload of the images and validate attributes on glance service container

```
glance image-list
```

- Exit the Keystone container

```
exit
```

- Login into the glance container

```
docker ps

docker exec -it <GLANCE-CONTAINERID> /bin/bash
```

NOTE: Also take a note of myql container ID which will be used within the glance container

- Verify the image is uploaded into glance file directory

```
cd /var/lib/glance/images

ls al
```

(Observe /etc/glance/glance-api.conf)

- Login into Mysql container, login to mysql process and verify available databases

```
mysql -u root -h <mysql-container-id> -p
```

(Provide password as admin123)

```

    show databases;
```

- Change to keystone database and list the users

```
use keystone;

select * from user;

exit
```

Kubernetes

Exercise-24

Kubernetes Installation, inititalization and join the worker nodes

Kubernetes Installation

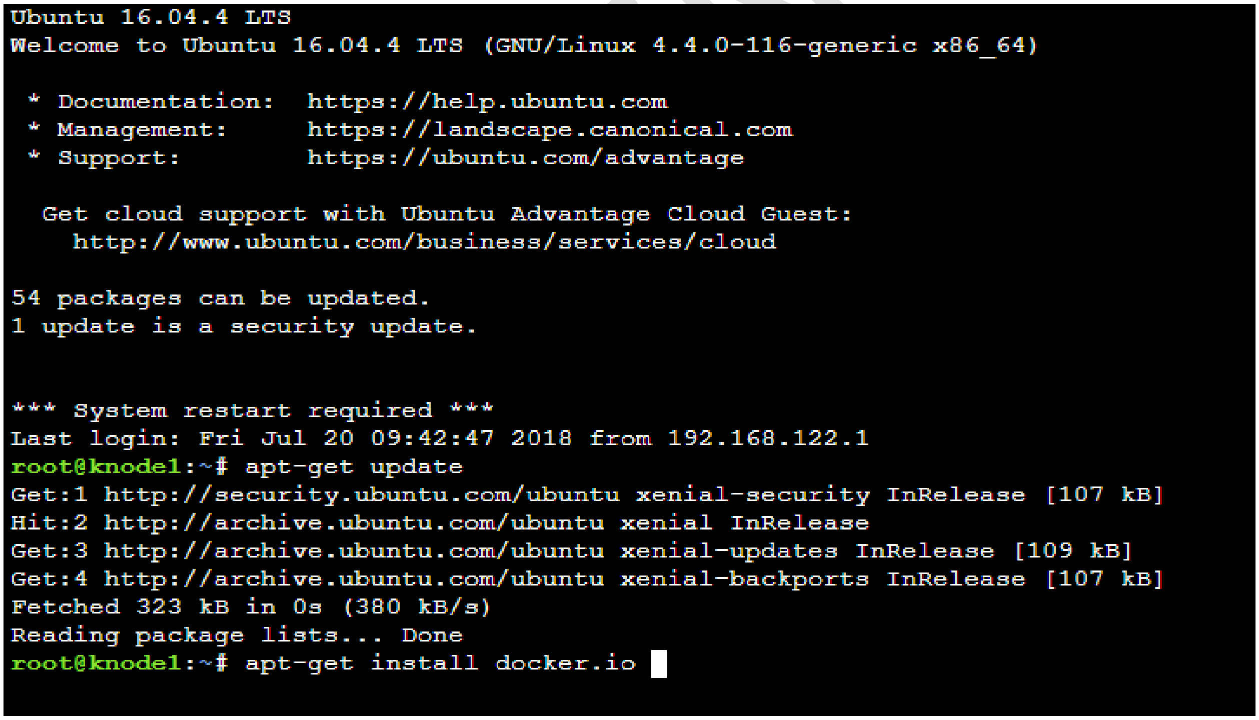
NOTE: *Perform the following commands on ALL 3 nodes (k8s, knode1 and knode2)*

- Update the apt package index:

```
apt-get update
```

- Install the latest version of Docker

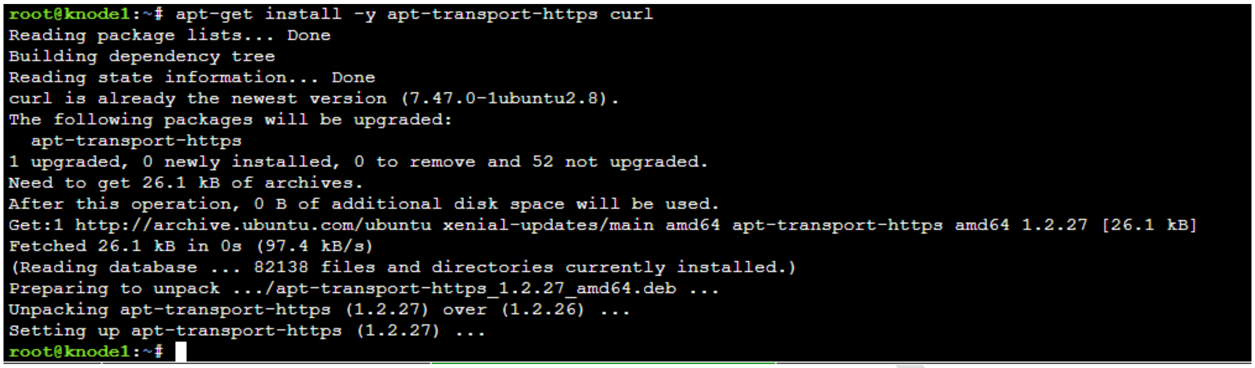
```
apt-get install docker.io
```



```
Ubuntu 16.04.4 LTS
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-116-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

54 packages can be updated.
1 update is a security update.


*** System restart required ***
Last login: Fri Jul 20 09:42:47 2018 from 192.168.122.1
root@knode1:~# apt-get update
Get:1 http://security.ubuntu.com/ubuntu xenial-security InRelease [107 kB]
Hit:2 http://archive.ubuntu.com/ubuntu xenial InRelease
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:4 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Fetched 323 kB in 0s (380 kB/s)
Reading package lists... Done
root@knode1:~# apt-get install docker.io
```

- Install packages to allow apt to use a repository over HTTPS:

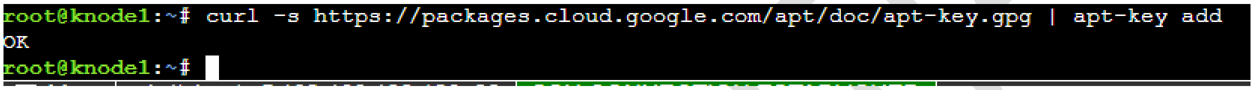
```
apt-get install -y apt-transport-https curl
```



```
root@knode1:~# apt-get install -y apt-transport-https curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
curl is already the newest version (7.47.0-1ubuntu2.8).
The following packages will be upgraded:
  apt-transport-https
1 upgraded, 0 newly installed, 0 to remove and 52 not upgraded.
Need to get 26.1 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu xenial-updates/main amd64 apt-transport-https amd64 1.2.27 [26.1 kB]
Fetched 26.1 kB in 0s (97.4 kB/s)
(Reading database ... 82138 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_1.2.27_amd64.deb ...
Unpacking apt-transport-https (1.2.27) over (1.2.26) ...
Setting up apt-transport-https (1.2.27) ...
root@knode1:~#
```

- Add Kubernetes official GPG key:

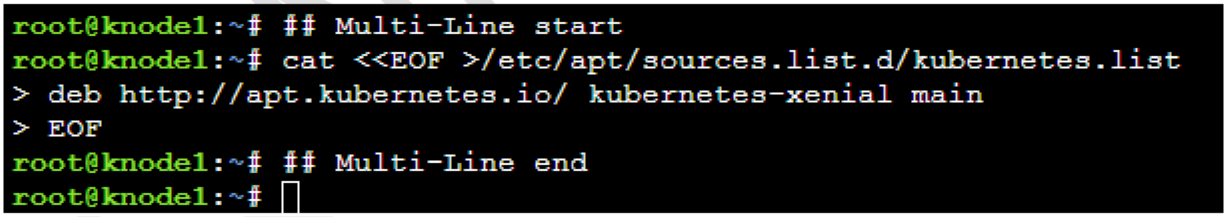
```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
```



```
root@knode1:~# curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add
OK
root@knode1:~#
```

- Add Kubernetes repo to sources.list.d

```
## Multi-Line start

cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

deb http://apt.kubernetes.io/ kubernetes-xenial main

EOF

## Multi-Line end
```

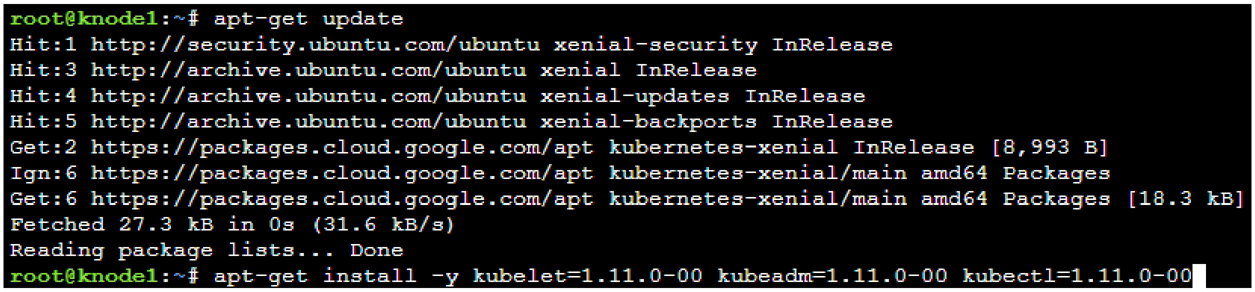


```
root@knode1:~# ## Multi-Line start
root@knode1:~# cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
> deb http://apt.kubernetes.io/ kubernetes-xenial main
> EOF
root@knode1:~# ## Multi-Line end
root@knode1:~#
```

- Update the apt package index and install the kubelet,kubeadm and kubectl packages:

```
apt-get update

apt-get install -y kubelet=1.11.0-00 kubeadm=1.11.0-00 kubectl=1.11.0-00
```

Correct with: apt-get install -y kubelet kubeadm kubectl
Add: swapoff -a
#nano /etc/fstab
Then comment the /swapfile --> #/swapfile

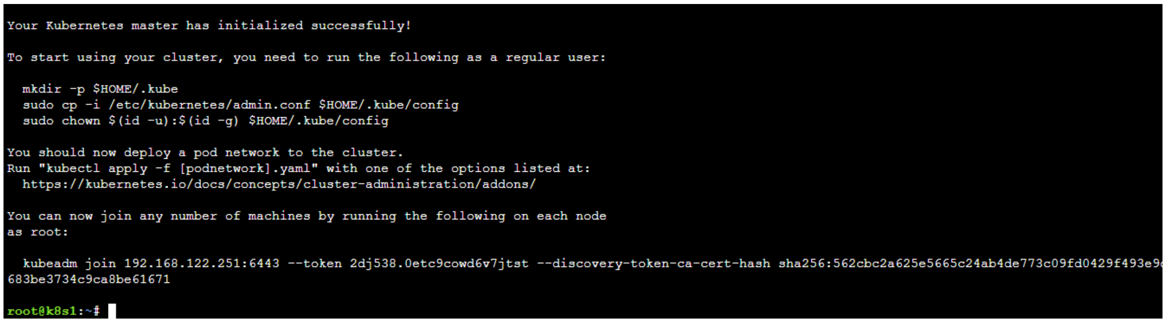


```
root@knode1:~# apt-get update
Hit:1 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu xenial InRelease
Hit:4 http://archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:5 http://archive.ubuntu.com/ubuntu xenial-backports InRelease
Get:2 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8,993 B]
Ign:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packages
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packages [18.3 kB]
Fetched 27.3 kB in 0s (31.6 kB/s)
Reading package lists... Done
root@knode1:~# apt-get install -y kubelet=1.11.0-00 kubeadm=1.11.0-00 kubectl=1.11.0-00
```

Kubernetes Master Initialization ONLY on k8s node

- On **k8s node**, initialize the kubernetes master

```
kubeadm init --kubernetes-version=1.11.0 --ignore-preflight-errors=all
```



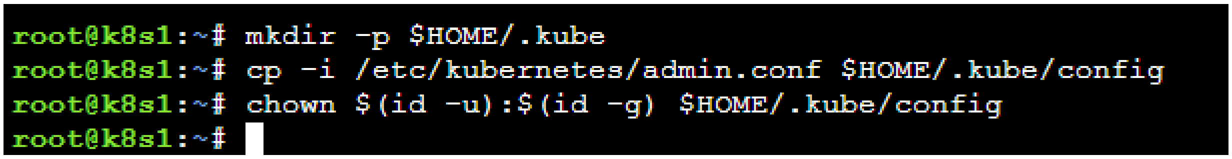
```
Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join 192.168.122.251:6443 --token 2dj538.0etc9cowd6v7jtst --discovery-token-ca-cert-hash sha256:562cbc2a625e5665c24ab4de773c09fd0429f493e9
683be3734c9ca8be61671

root@k8s1:~#
```

NOTE: Please take a note of the command which is the output of the above command, as that will used later for joining worker nodes with master. It will be of the format:

```
kubeadm join --token <token> <master-ip>:<master-port> --discovery-token-ca-ce
rt-hash sha256:<hash>
```

- Prepare system for adding workloads, including the network plugin.

```
mkdir -p $HOME/.kube

cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

chown $(id -u):$(id -g) $HOME/.kube/config
```

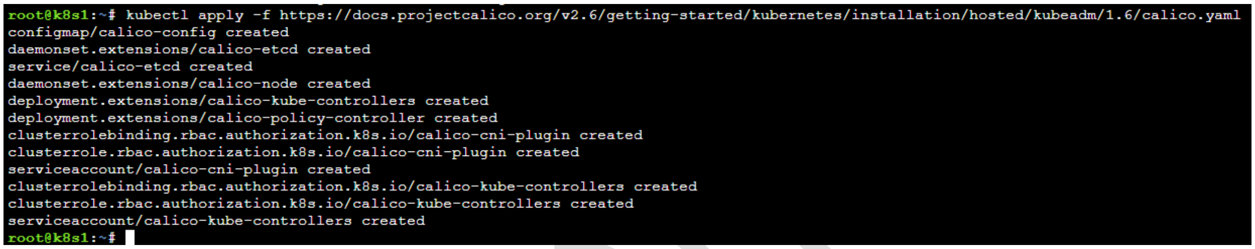


```
root@k8s1:~# mkdir -p $HOME/.kube
root@k8s1:~# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@k8s1:~# chown $(id -u):$(id -g) $HOME/.kube/config
root@k8s1:~#
```

- Install the Calico network plugin:

```
kubectl apply -f https://docs.projectcalico.org/v2.6/getting-started/kubernetes/installation/hosted/kubeadm/1.6/calico.yaml
```

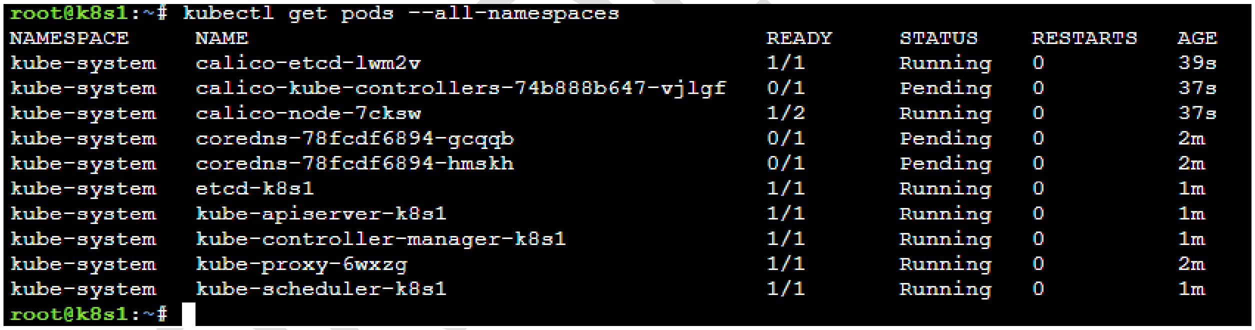
pkumar@k8s-master:~$ kubectl apply -f https://docs.projectcalico.org/v3.14/manifests/calico.yaml



```
root@k8s1:~# kubectl apply -f https://docs.projectcalico.org/v2.6/getting-started/kubernetes/installation/hosted/kubeadm/1.6/calico.yaml
configmap/calico-config created
daemonset.extensions/calico-etcd created
service/calico-etcd created
daemonset.extensions/calico-node created
deployment.extensions/calico-kube-controllers created
deployment.extensions/calico-policy-controller created
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
serviceaccount/calico-cni-plugin created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
root@k8s1:~#
```

- Check to see if the pods are running:

```
kubectl get pods --all-namespaces
```



```
root@k8s1:~# kubectl get pods --all-namespaces
NAMESPACE     NAME                                       READY     STATUS    RESTARTS   AGE
kube-system   calico-etcd-lwm2v                          1/1       Running   0          39s
kube-system   calico-kube-controllers-74b888b647-vjlgf   0/1       Pending   0          37s
kube-system   calico-node-7cksw                          1/2       Running   0          37s
kube-system   coredns-78fcdf6894-gcqqb                   0/1       Pending   0          2m
kube-system   coredns-78fcdf6894-hmskh                   0/1       Pending   0          2m
kube-system   etcd-k8s1                                  1/1       Running   0          1m
kube-system   kube-apiserver-k8s1                        1/1       Running   0          1m
kube-system   kube-controller-manager-k8s1               1/1       Running   0          1m
kube-system   kube-proxy-6wxzg                           1/1       Running   0          2m
kube-system   kube-scheduler-k8s1                        1/1       Running   0          1m
root@k8s1:~#
```

(The pods will start up over a short period of time)

Install Kubernetes dashboard plugin

Before joining the nodes, install the kubernetes dashboard, so that it comes on master node.

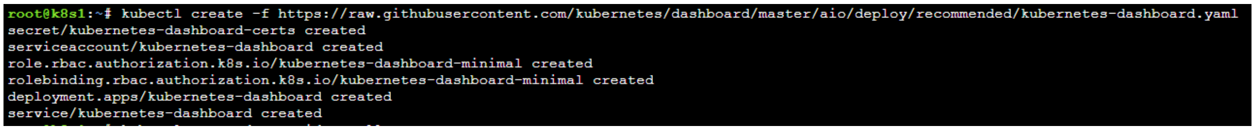
Deploying the dashboard

```
kubectl create -f https://raw.githubusercontent.com/kubernetes/dashboard/master/aio/deploy/recommended/kubernetes-dashboard.yaml
```

If this link doesn't work, try with:

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.4/aio/deploy/recommended.yaml

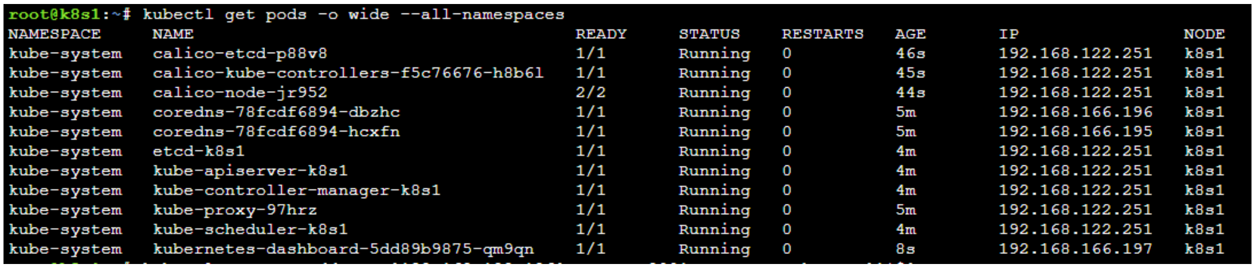
Source: https://github.com/kubernetes/dashboard



```
root@k8s1:~# kubectl create -f https://raw.githubusercontent.com/kubernetes/dashboard/master/aio/deploy/recommended/kubernetes-dashboard.yaml
secret/kubernetes-dashboard-certs created
serviceaccount/kubernetes-dashboard created
role.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
deployment.apps/kubernetes-dashboard created
service/kubernetes-dashboard created
```

Verify that the dashboard pod is in running state and is running on k8s1 node, i.e, master node

```
kubectl get pods -o wide --all-namespaces
```



```
root@k8s1:~# kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                                       READY     STATUS    RESTARTS   AGE       IP                NODE
kube-system   calico-etcd-p88v8                          1/1       Running   0          46s       192.168.122.251   k8s1
kube-system   calico-kube-controllers-f5c76676-h8b61     1/1       Running   0          45s       192.168.122.251   k8s1
kube-system   calico-node-jr952                          2/2       Running   0          44s       192.168.122.251   k8s1
kube-system   coredns-78fcdf6894-dbzhc                   1/1       Running   0          5m        192.168.166.196   k8s1
kube-system   coredns-78fcdf6894-hcxfn                   1/1       Running   0          5m        192.168.166.195   k8s1
kube-system   etcd-k8s1                                  1/1       Running   0          4m        192.168.122.251   k8s1
kube-system   kube-apiserver-k8s1                        1/1       Running   0          4m        192.168.122.251   k8s1
kube-system   kube-controller-manager-k8s1               1/1       Running   0          4m        192.168.122.251   k8s1
kube-system   kube-proxy-97hrz                           1/1       Running   0          5m        192.168.122.251   k8s1
kube-system   kube-scheduler-k8s1                        1/1       Running   0          4m        192.168.122.251   k8s1
kube-system   kubernetes-dashboard-5dd89b9875-qm9qn      1/1       Running   0          8s        192.168.166.197   k8s1
```

To join the worker nodes into Kubernetes master

In all the Worker nodes run:

swapoff -a
#nano /etc/fstab
Then comment the /swapfile --> #/swapfile

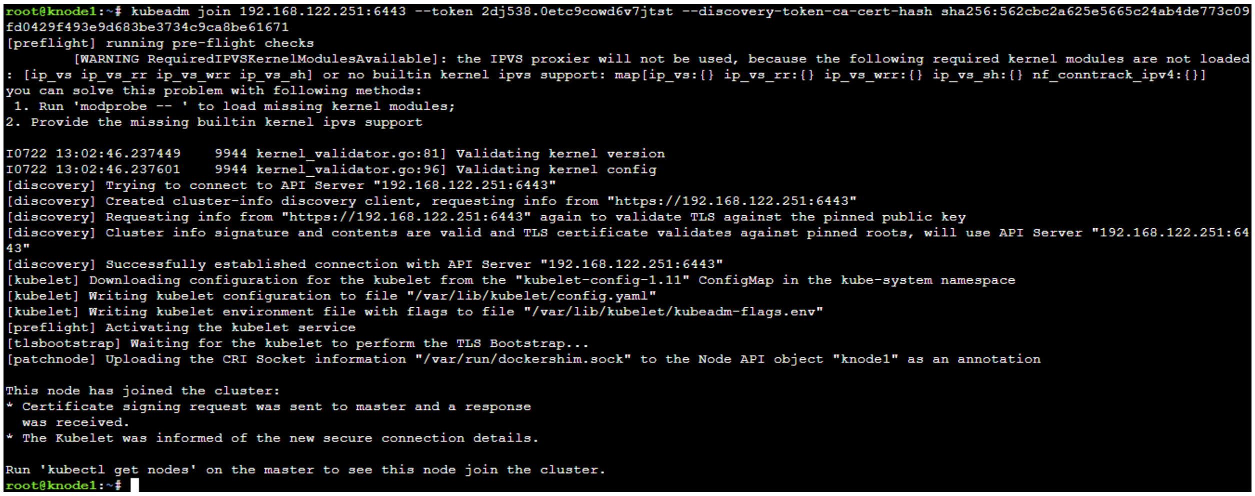
NOTE: On knode1 and knode2, perform following commands to join the Kubernetes cluster

- Run the below command (which was the output of kubeadm init during master initialization)

```
kubeadm join --token <token> <master-ip>:<master-port> --discovery-token-ca-cert-hash sha256:<hash>
```

check with: kubectl get nodes

A few seconds later, you should notice this node in the output from **kubectl get nodes** when run on the master (k8s1 node)



```
root@knode1:~# kubeadm join 192.168.122.251:6443 --token 2dj538.0etc9cowd6v7jtst --discovery-token-ca-cert-hash sha256:562cbc2a625e5665c24ab4de773c09
fd0429f493e9d683be3734c9ca8be61671
[preflight] running pre-flight checks
        [WARNING RequiredIPVSKernelModulesAvailable]: the IPVS proxier will not be used, because the following required kernel modules are not loaded
: [ip_vs ip_vs_rr ip_vs_wrr ip_vs_sh] or no builtin kernel ipvs support: map[ip_vs:{} ip_vs_rr:{} ip_vs_wrr:{} ip_vs_sh:{} nf_conntrack_ipv4:{}]
you can solve this problem with following methods:
 1. Run 'modprobe -- ' to load missing kernel modules;
2. Provide the missing builtin kernel ipvs support

I0722 13:02:46.237449    9944 kernel_validator.go:81] Validating kernel version
I0722 13:02:46.237601    9944 kernel_validator.go:96] Validating kernel config
[discovery] Trying to connect to API Server "192.168.122.251:6443"
[discovery] Created cluster-info discovery client, requesting info from "https://192.168.122.251:6443"
[discovery] Requesting info from "https://192.168.122.251:6443" again to validate TLS against the pinned public key
[discovery] Cluster info signature and contents are valid and TLS certificate validates against pinned roots, will use API Server "192.168.122.251:64
43"
[discovery] Successfully established connection with API Server "192.168.122.251:6443"
[kubelet] Downloading configuration for the kubelet from the "kubelet-config-1.11" ConfigMap in the kube-system namespace
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[preflight] Activating the kubelet service
[tlsbootstrap] Waiting for the kubelet to perform the TLS Bootstrap...
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "knode1" as an annotation

This node has joined the cluster:
* Certificate signing request was sent to master and a response
  was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the master to see this node join the cluster.
root@knode1:~#
```

- Extra commands for help (Do NOT delete node): To delete the node:

```
kubectl drain <node name> --delete-local-data --force --ignore-daemonsets

kubectl delete node <node name>
```

To access the Dashboard:

* http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/ (verify the port 8001 with the kubectl proxy command)
* kubectl create serviceaccount dashboard -n default
* run:

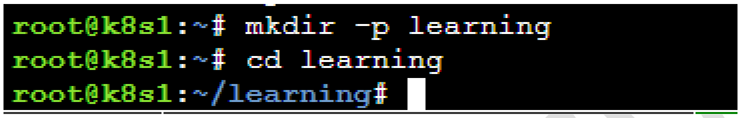
kubectl clusterbinding dashboard-admin -n default \
--clusterrole=cluster-admin \
--serviceaccount=default:dashboard

Exercise-25

Working with Pods

- Create a directory in k8s node and a file pod.yml

```
mkdir -p learning

cd learning
```



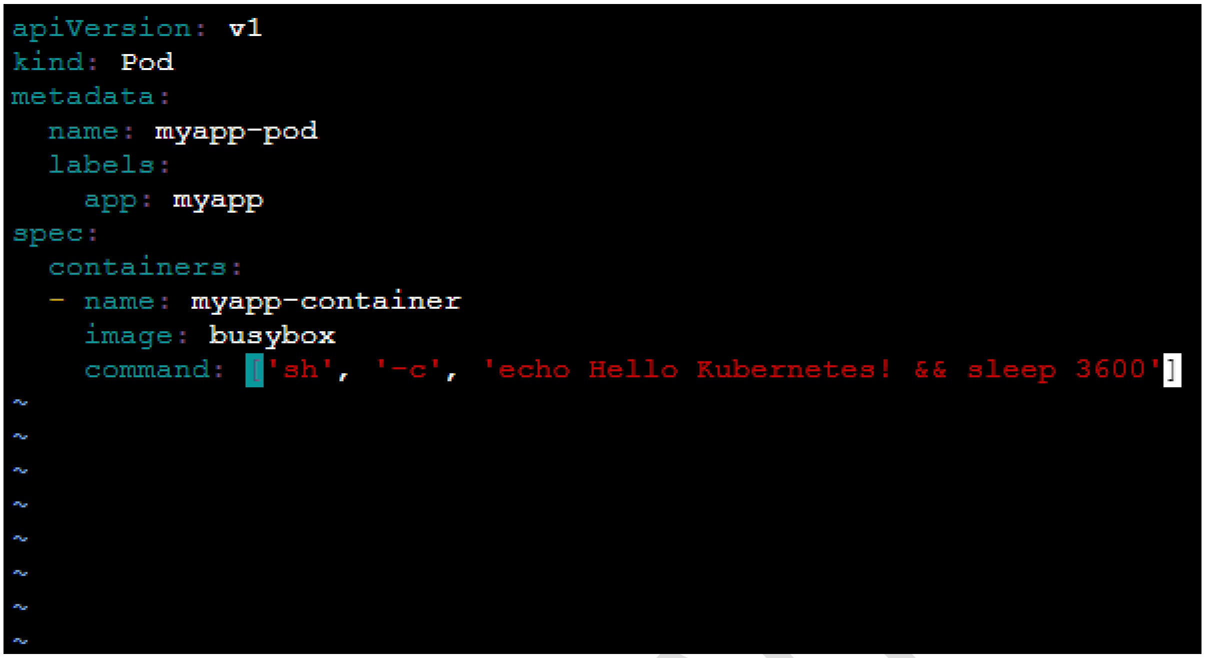
```
root@k8s1:~# mkdir -p learning
root@k8s1:~# cd learning
root@k8s1:~/learning#
```

- Create a file named pod.yml

```
vim pod.yml
```

- Insert the below lines"

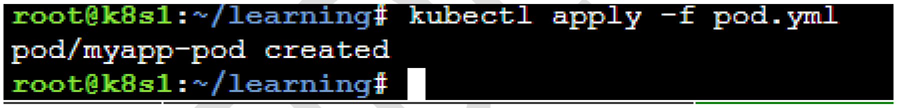
```
apiVersion: v1

kind: Pod

metadata:

  name: myapp-pod

  labels:

    app: myapp

spec:

  containers:

  - name: myapp-container

    image: busybox

    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```



```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
~
~
~
~
~
~
~
~
```

- Create the pod with specified file

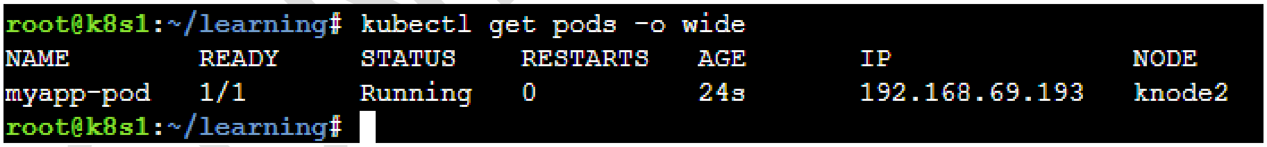
```
kubectl apply -f pod.yml
```



```
root@k8s1:~/learning# kubectl apply -f pod.yml
pod/myapp-pod created
root@k8s1:~/learning#
```

- To introspect pod information

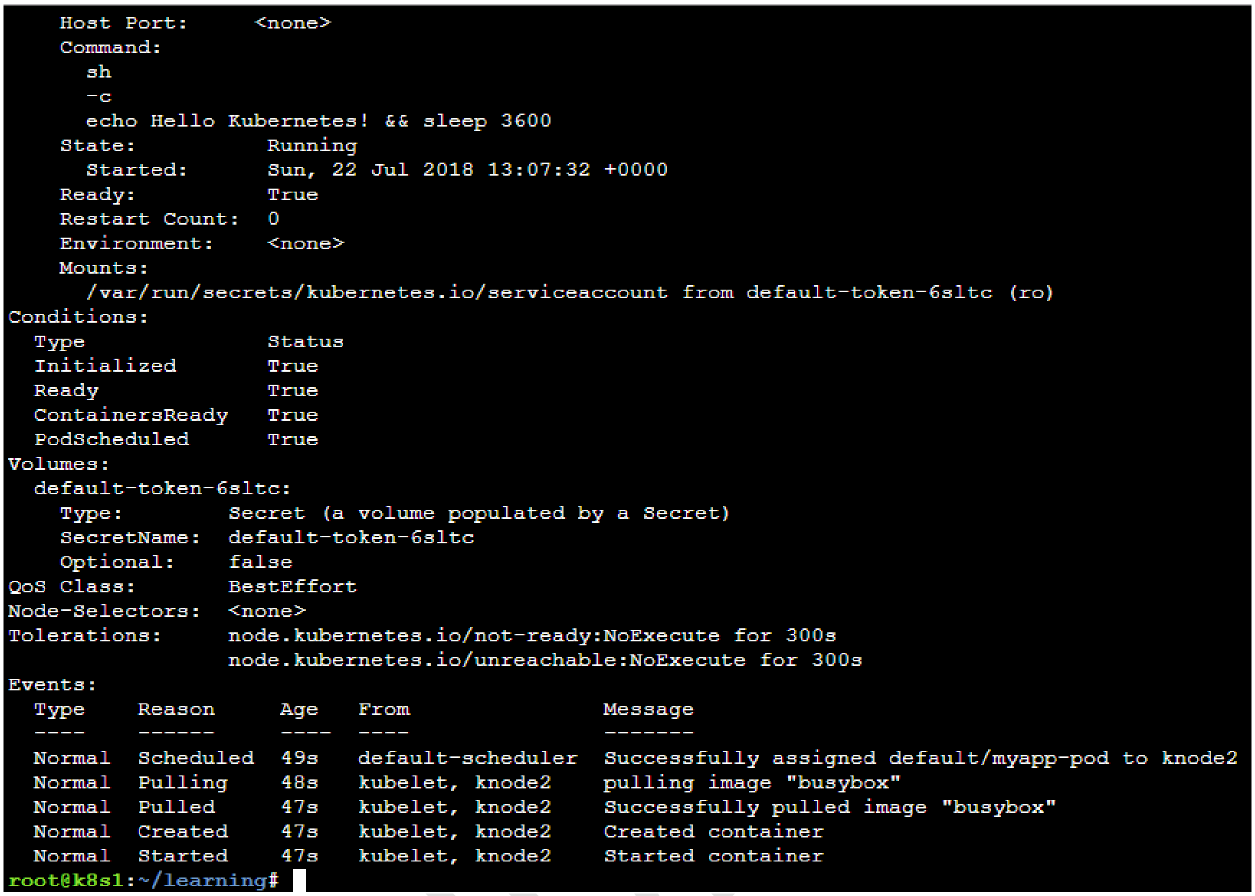
```
kubectl get pods -o wide
```



```
root@k8s1:~/learning# kubectl get pods -o wide
NAME         READY     STATUS     RESTARTS    AGE       IP                NODE
myapp-pod    1/1       Running    0           24s       192.168.69.193    knode2
root@k8s1:~/learning#
```

- Another useful command to check pod details

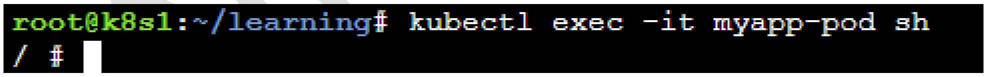
```
kubectl describe pods myapp-pod
```



```
    Host Port:      <none>
    Command:
      sh
      -c
      echo Hello Kubernetes! && sleep 3600
    State:          Running
      Started:      Sun, 22 Jul 2018 13:07:32 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-6sltc (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-6sltc:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-6sltc
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age   From                Message
  ----    ------     ----  ----                -------
  Normal  Scheduled  49s   default-scheduler   Successfully assigned default/myapp-pod to knode2
  Normal  Pulling    48s   kubelet, knode2     pulling image "busybox"
  Normal  Pulled     47s   kubelet, knode2     Successfully pulled image "busybox"
  Normal  Created    47s   kubelet, knode2     Created container
  Normal  Started    47s   kubelet, knode2     Started container
root@k8s1:~/learning#
```

- To enter into container

```
kubectl exec -it myapp-pod sh
```



```
root@k8s1:~/learning# kubectl exec -it myapp-pod sh
/ #
```

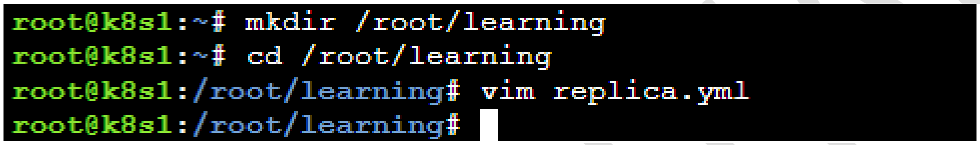
Exercise-26

Working with ReplicaSets

- Create a directory in k8s node and a file replica.yml

- `mkdir /root/learning`
- `cd /root/learning`

```
vim replica.yml
```

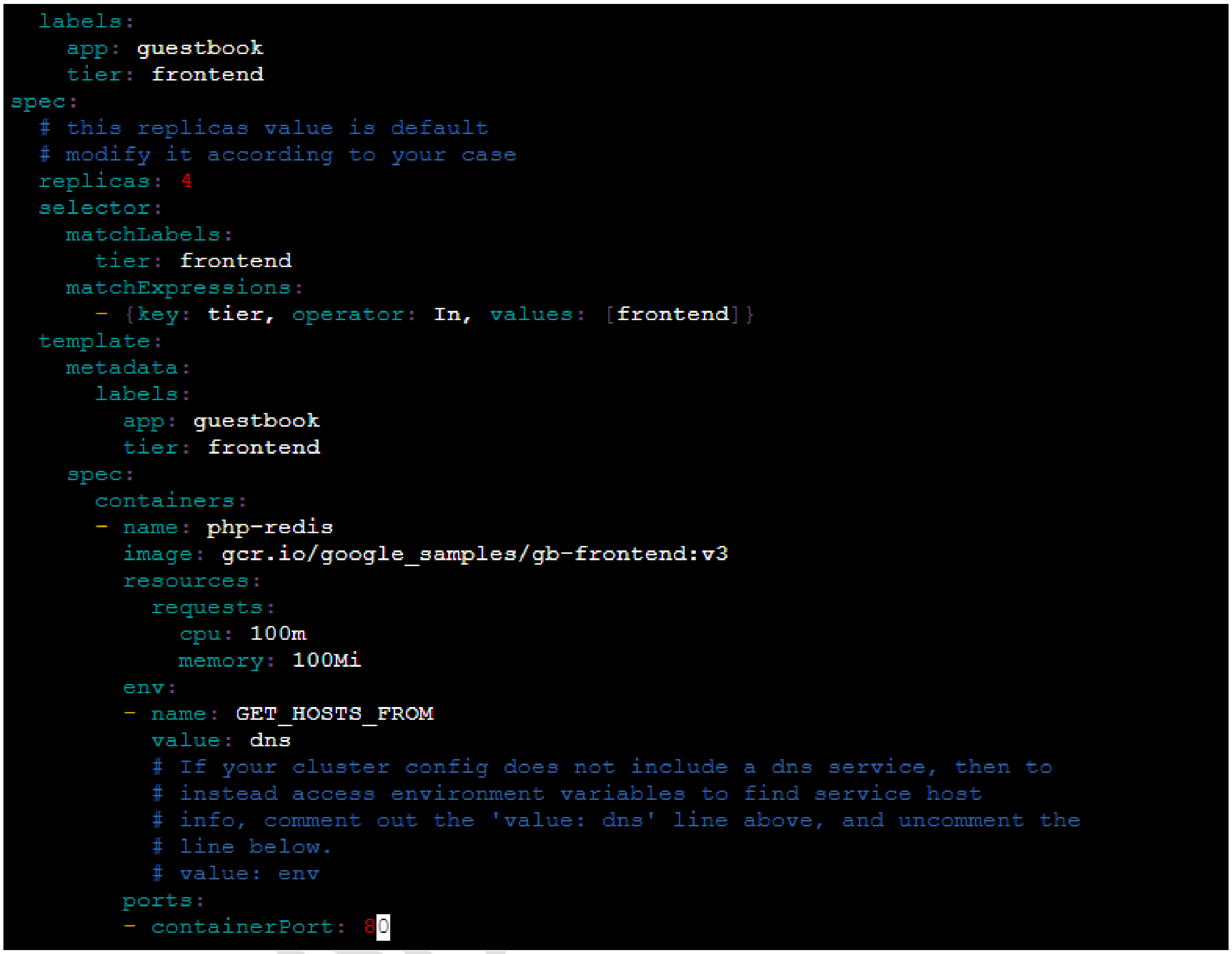


```
root@k8s1:~# mkdir /root/learning
root@k8s1:~# cd /root/learning
root@k8s1:/root/learning# vim replica.yml
root@k8s1:/root/learning#
```

- Insert the below lines

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2

kind: ReplicaSet

metadata:

  name: frontend

  labels:

    app: guestbook

    tier: frontend

spec:

  # this replicas value is default

  # modify it according to your case

  replicas: 4

  selector:

    matchLabels:

      tier: frontend
    matchExpressions:
      - {key: tier, operator: In, values: [frontend]}
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
          # If your cluster config does not include a dns service, then to
          # instead access environment variables to find service host
          # info, comment out the 'value: dns' line above, and uncomment the
          # line below.
          # value: env
        ports:
```

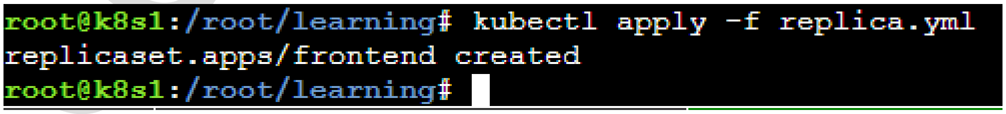
```
        - containerPort: 80
```



```
  labels:
    app: guestbook
    tier: frontend
spec:
  # this replicas value is default
  # modify it according to your case
  replicas: 4
  selector:
    matchLabels:
      tier: frontend
    matchExpressions:
      - {key: tier, operator: In, values: [frontend]}
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
          # If your cluster config does not include a dns service, then to
          # instead access environment variables to find service host
          # info, comment out the 'value: dns' line above, and uncomment the
          # line below.
          # value: env
        ports:
        - containerPort: 80
```

- Create the pod with specified file

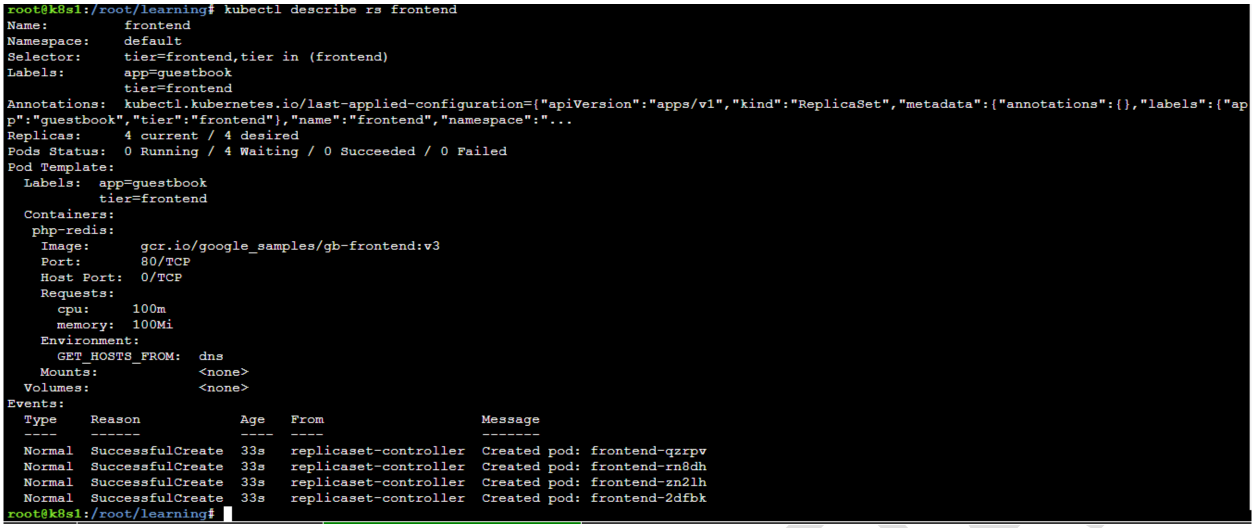
```
kubectl apply -f replica.yml
```



```
root@k8s1:/root/learning# kubectl apply -f replica.yml
replicaset.apps/frontend created
root@k8s1:/root/learning# 
```

- To introspect pod information

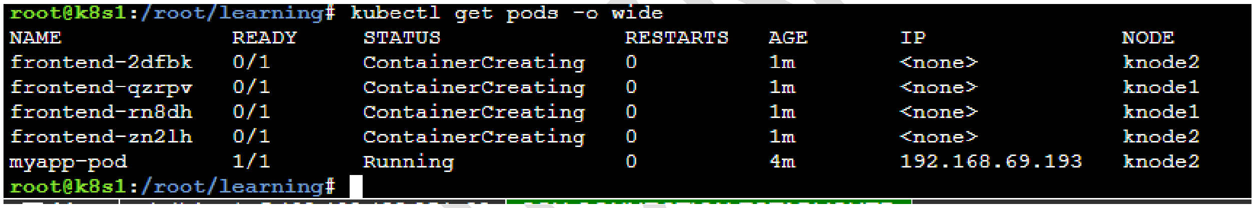
```
kubectl get rs -o json
```



```
root@k8s1:/root/learning# kubectl describe rs frontend
Name:         frontend
Namespace:    default
Selector:     tier=frontend,tier in (frontend)
Labels:       app=guestbook
              tier=frontend
Annotations:  kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"ReplicaSet","metadata":{"annotations":{},"labels":{"ap
p":"guestbook","tier":"frontend"},"name":"frontend","namespace":"...
Replicas:     4 current / 4 desired
Pods Status:  0 Running / 4 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=guestbook
           tier=frontend
  Containers:
   php-redis:
    Image:      gcr.io/google_samples/gb-frontend:v3
    Port:       80/TCP
    Host Port:  0/TCP
    Requests:
      cpu:     100m
      memory:  100Mi
    Environment:
      GET_HOSTS_FROM:  dns
    Mounts:            <none>
  Volumes:             <none>
Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----                   -------
  Normal  SuccessfulCreate  33s   replicaset-controller  Created pod: frontend-qzrpv
  Normal  SuccessfulCreate  33s   replicaset-controller  Created pod: frontend-rn8dh
  Normal  SuccessfulCreate  33s   replicaset-controller  Created pod: frontend-zn2lh
  Normal  SuccessfulCreate  33s   replicaset-controller  Created pod: frontend-2dfbk
root@k8s1:/root/learning#
```

- Another useful command to check pod details

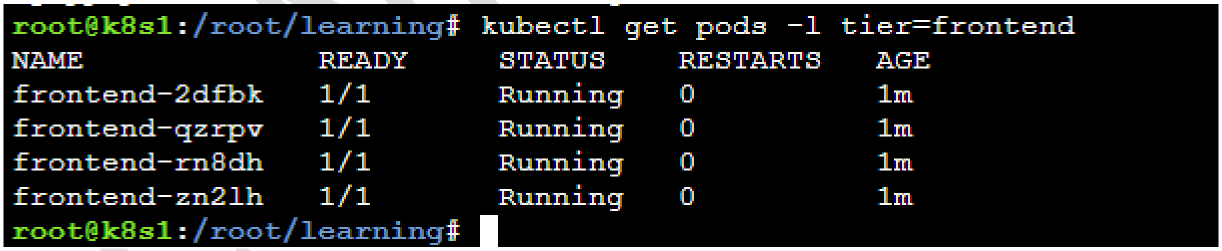
```
kubectl describe rs frontend
```



```
root@k8s1:/root/learning# kubectl get pods -o wide
NAME             READY     STATUS              RESTARTS   AGE       IP               NODE
frontend-2dfbk   0/1       ContainerCreating   0          1m        <none>           knode2
frontend-qzrpv   0/1       ContainerCreating   0          1m        <none>           knode1
frontend-rn8dh   0/1       ContainerCreating   0          1m        <none>           knode1
frontend-zn2lh   0/1       ContainerCreating   0          1m        <none>           knode2
myapp-pod        1/1       Running             0          4m        192.168.69.193   knode2
root@k8s1:/root/learning#
```

- Display all the pod information including their IP addresses

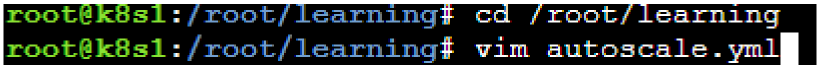
```
kubectl get pods -o wide
```



```
root@k8s1:/root/learning# kubectl get pods -l tier=frontend
NAME             READY     STATUS    RESTARTS   AGE
frontend-2dfbk   1/1       Running   0          1m
frontend-qzrpv   1/1       Running   0          1m
frontend-rn8dh   1/1       Running   0          1m
frontend-zn2lh   1/1       Running   0          1m
root@k8s1:/root/learning#
```

- Display selected pod information with labels

```
kubectl get pods -l tier=frontend
```

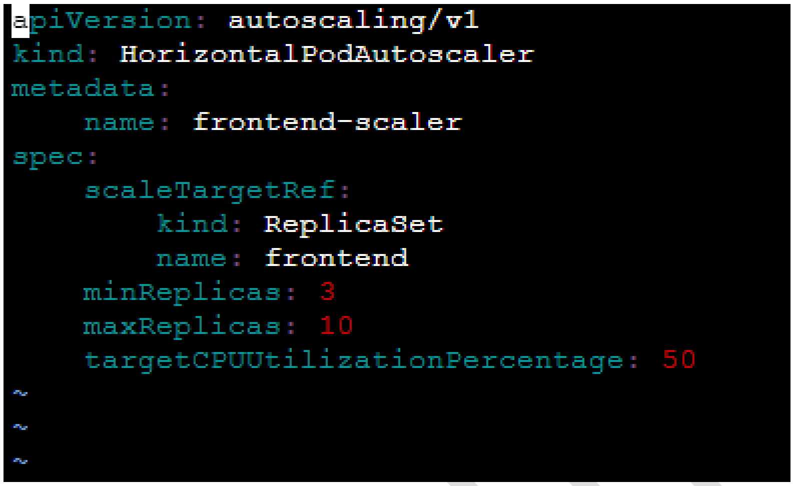


```
root@k8s1:/root/learning# cd /root/learning
root@k8s1:/root/learning# vim autoscale.yml
```

- Autoscale options with ReplicaSets

```
cd /root/learning
```

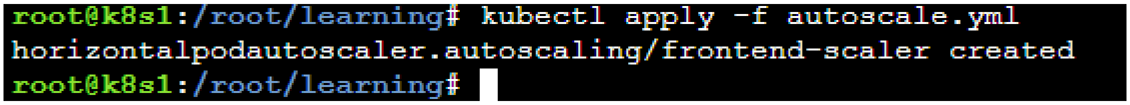
```
vim autoscale.yml
```



```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
    name: frontend-scaler
spec:
    scaleTargetRef:
        kind: ReplicaSet
        name: frontend
    minReplicas: 3
    maxReplicas: 10
    targetCPUUtilizationPercentage: 50
~
~
~
```

- Insert the below lines

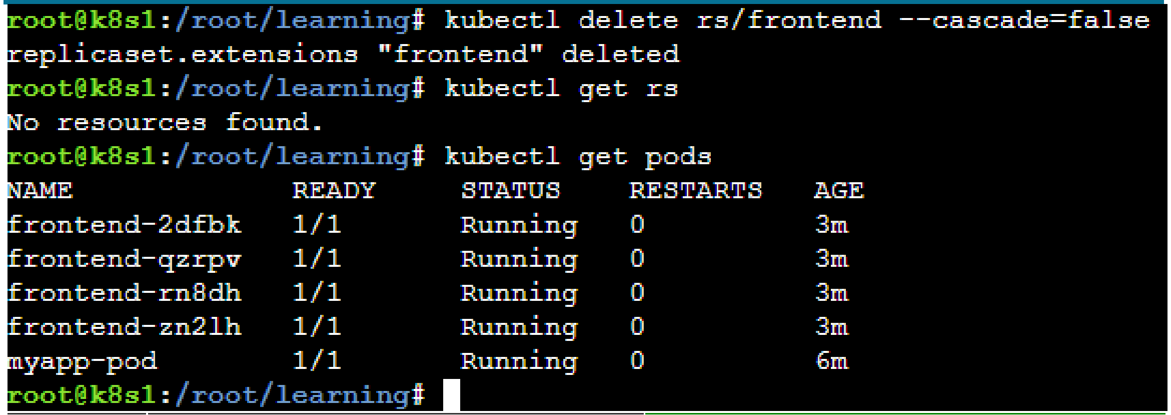
```
apiVersion: autoscaling/v1

kind: HorizontalPodAutoscaler

metadata:

    name: frontend-scaler

spec:

    scaleTargetRef:

        kind: ReplicaSet

        name: frontend

    minReplicas: 3

    maxReplicas: 10

    targetCPUUtilizationPercentage: 50
```



```
root@k8s1:/root/learning# kubectl apply -f autoscale.yml
horizontalpodautoscaler.autoscaling/frontend-scaler created
root@k8s1:/root/learning#
```

- Apply the autoscale file

```
kubectl apply -f autoscale.yml
```



```
root@k8s1:/root/learning# kubectl delete rs/frontend --cascade=false
replicaset.extensions "frontend" deleted
root@k8s1:/root/learning# kubectl get rs
No resources found.
root@k8s1:/root/learning# kubectl get pods
NAME              READY     STATUS    RESTARTS   AGE
frontend-2dfbk    1/1       Running   0          3m
frontend-qzrpv    1/1       Running   0          3m
frontend-rn8dh    1/1       Running   0          3m
frontend-zn2lh    1/1       Running   0          3m
myapp-pod         1/1       Running   0          6m
root@k8s1:/root/learning#
```

- Cascading option (Demonstrates loosely couple nature of replicaset and pods

```
kubectl delete rs/frontend --cascade=false

kubectl get rs

kubectl get pods
```

Exercise-27

Working with Deployments

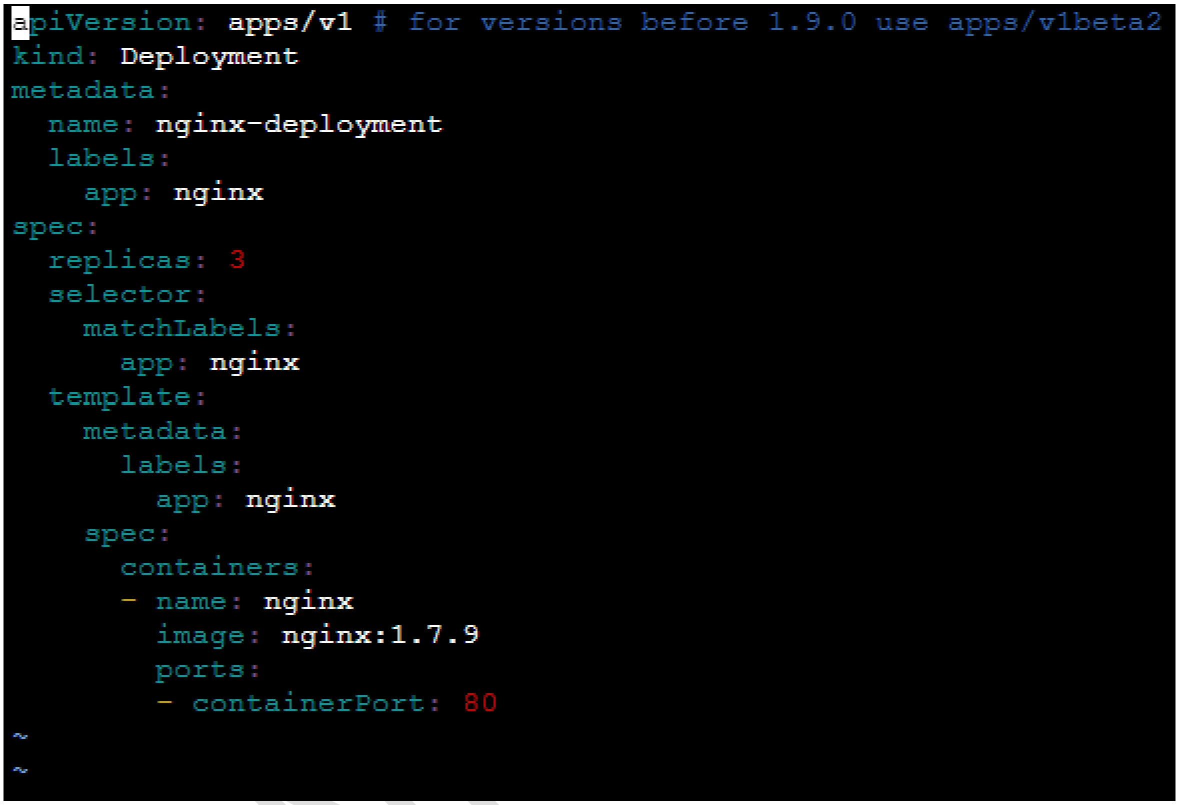
- Create a new file in k8s node with name deploy1.yml

```
cd /root/learning/
vim deploy1.yml
```

- Insert the below lines

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
```

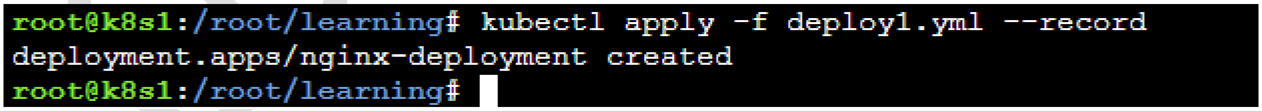
```
        image: nginx:1.7.9

        ports:

        - containerPort: 80
```



```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
~
~
```

- Create the pod with specified file

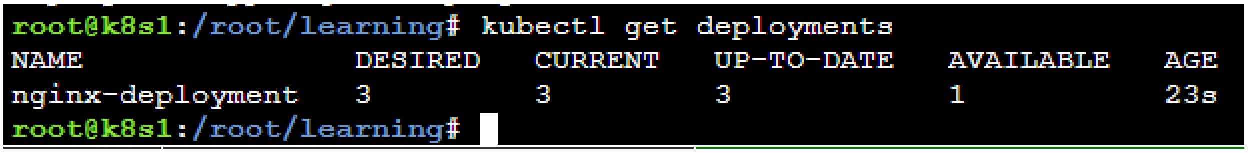
```
kubectl apply -f deploy1.yml --record
```



```
root@k8s1:/root/learning# kubectl apply -f deploy1.yml --record
deployment.apps/nginx-deployment created
root@k8s1:/root/learning#
```

- To introspect pod information

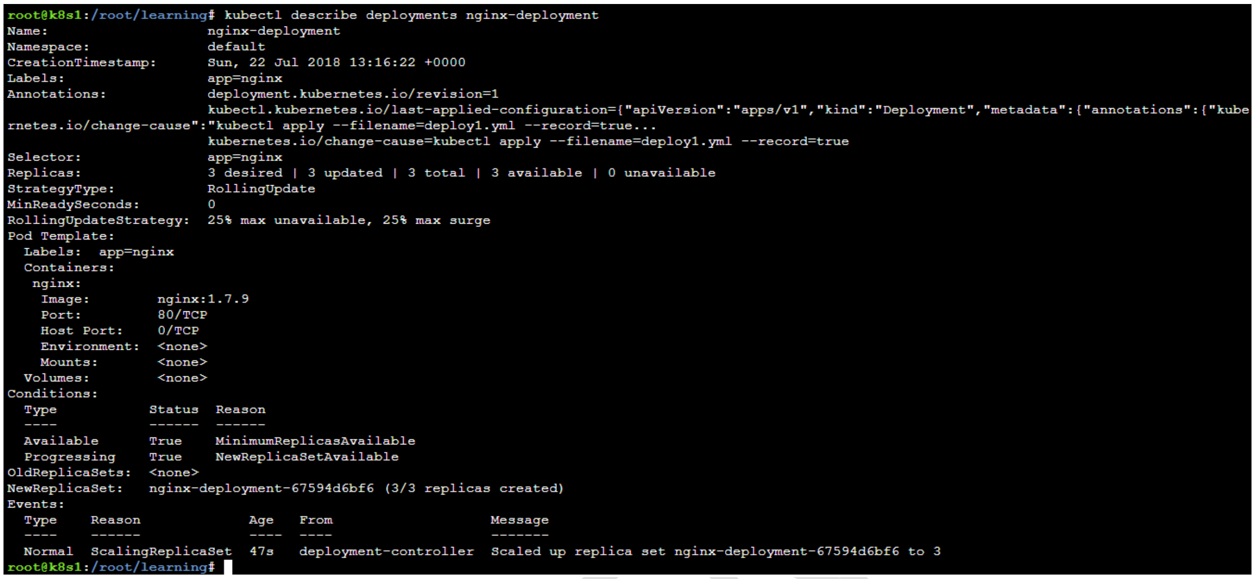
```
kubectl get deployments
```



```
root@k8s1:/root/learning# kubectl get deployments
NAME               DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3         3         3            1           23s
root@k8s1:/root/learning#
```

- Another useful command to check pod details

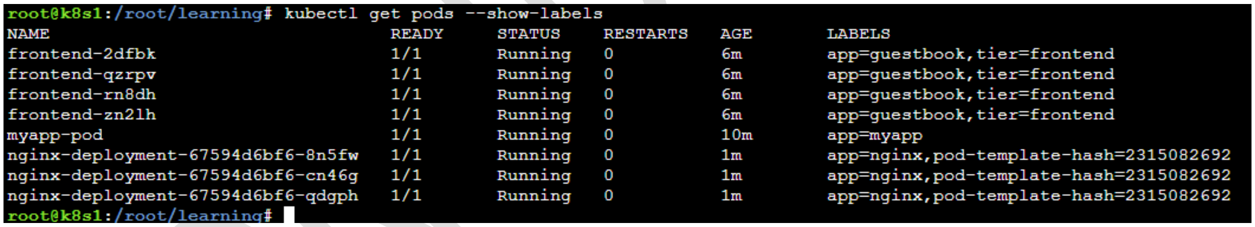
```
kubectl describe deployments nginx-deployment
```



```
root@k8s1:/root/learning# kubectl describe deployments nginx-deployment
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Sun, 22 Jul 2018 13:16:22 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision=1
                        kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{"kube
rnetes.io/change-cause":"kubectl apply --filename=deploy1.yml --record=true...
                        kubernetes.io/change-cause=kubectl apply --filename=deploy1.yml --record=true
Selector:               app=nginx
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.7.9
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   nginx-deployment-67594d6bf6 (3/3 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  47s   deployment-controller  Scaled up replica set nginx-deployment-67594d6bf6 to 3
root@k8s1:/root/learning#
```

- Display all the pod information along with labels

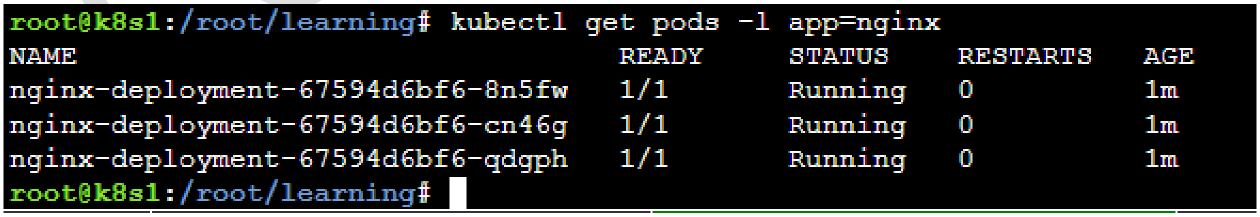
```
kubectl get pods --show-labels
```



```
root@k8s1:/root/learning# kubectl get pods --show-labels
NAME                                READY     STATUS    RESTARTS   AGE       LABELS
frontend-2dfbk                      1/1       Running   0          6m        app=guestbook,tier=frontend
frontend-qzrpv                      1/1       Running   0          6m        app=guestbook,tier=frontend
frontend-rn8dh                      1/1       Running   0          6m        app=guestbook,tier=frontend
frontend-zn2lh                      1/1       Running   0          6m        app=guestbook,tier=frontend
myapp-pod                           1/1       Running   0          10m       app=myapp
nginx-deployment-67594d6bf6-8n5fw   1/1       Running   0          1m        app=nginx,pod-template-hash=2315082692
nginx-deployment-67594d6bf6-cn46g   1/1       Running   0          1m        app=nginx,pod-template-hash=2315082692
nginx-deployment-67594d6bf6-qdgph   1/1       Running   0          1m        app=nginx,pod-template-hash=2315082692
root@k8s1:/root/learning#
```

- Display selected pod information with labels

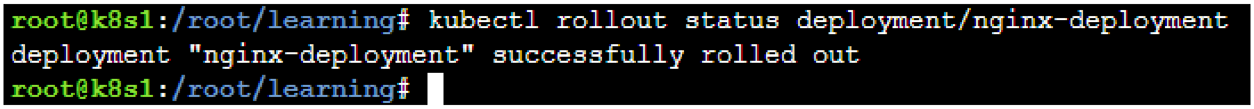
```
kubectl get pods -l app=nginx
```



```
root@k8s1:/root/learning# kubectl get pods -l app=nginx
NAME                                READY     STATUS    RESTARTS   AGE
nginx-deployment-67594d6bf6-8n5fw   1/1       Running   0          1m
nginx-deployment-67594d6bf6-cn46g   1/1       Running   0          1m
nginx-deployment-67594d6bf6-qdgph   1/1       Running   0          1m
root@k8s1:/root/learning#
```

- To check the roll-out status

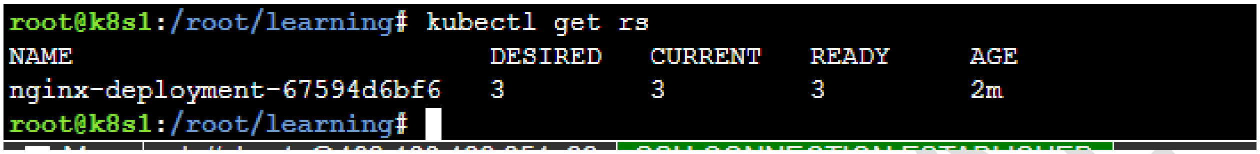
```
kubectl rollout status deployment/nginx-deployment
```



```
root@k8s1:/root/learning# kubectl rollout status deployment/nginx-deployment
deployment "nginx-deployment" successfully rolled out
root@k8s1:/root/learning#
```

- Check the ReplicaSet controllers

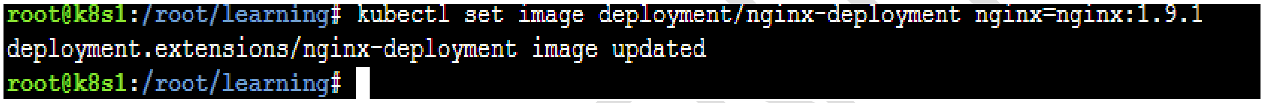
```
kubectl get rs
```



```
root@k8s1:/root/learning# kubectl get rs
NAME                          DESIRED   CURRENT   READY     AGE
nginx-deployment-67594d6bf6   3         3         3         2m
root@k8s1:/root/learning#
```

- To rollout an update to nginx version

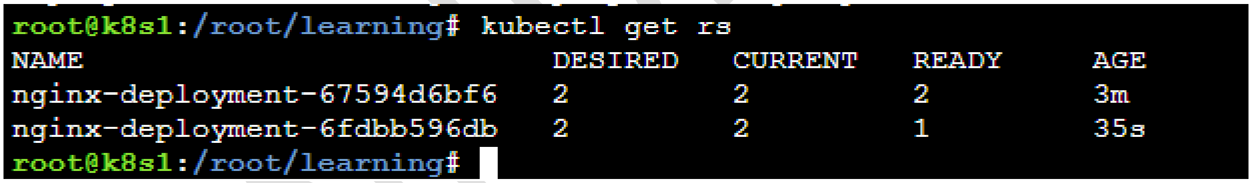
```
kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1
```



```
root@k8s1:/root/learning# kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1
deployment.extensions/nginx-deployment image updated
root@k8s1:/root/learning#
```

- Check the ReplicaSet controllers again

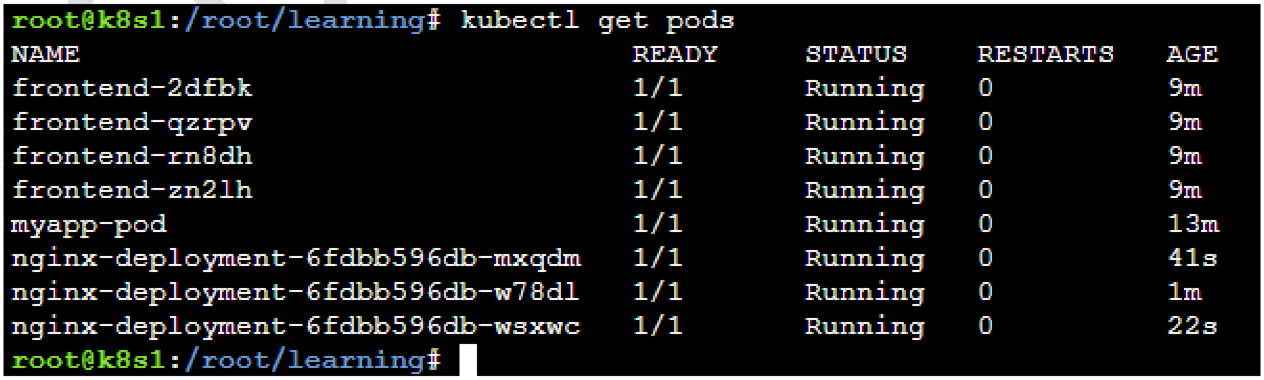
```
kubectl get rs
```



```
root@k8s1:/root/learning# kubectl get rs
NAME                          DESIRED   CURRENT   READY     AGE
nginx-deployment-67594d6bf6   2         2         2         3m
nginx-deployment-6fdbb596db   2         2         1         35s
root@k8s1:/root/learning#
```

- Check the pods and verify it shows new pods

```
kubectl get pods
```



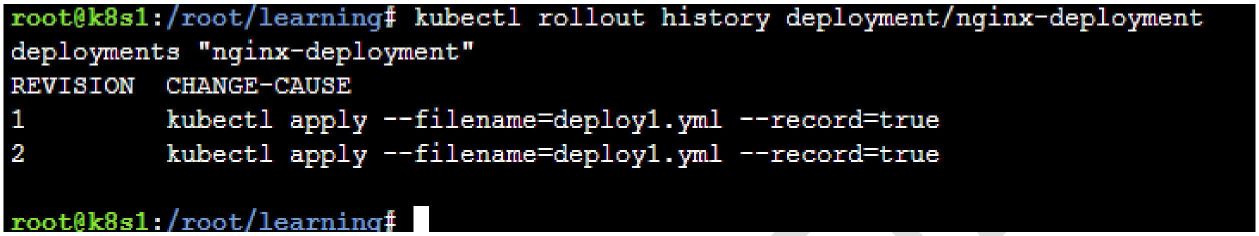
```
root@k8s1:/root/learning# kubectl get pods
NAME                                READY     STATUS    RESTARTS   AGE
frontend-2dfbk                      1/1       Running   0          9m
frontend-qzrpv                      1/1       Running   0          9m
frontend-rn8dh                      1/1       Running   0          9m
frontend-zn2lh                      1/1       Running   0          9m
myapp-pod                           1/1       Running   0          13m
nginx-deployment-6fdbb596db-mxqdm   1/1       Running   0          41s
nginx-deployment-6fdbb596db-w78dl   1/1       Running   0          1m
nginx-deployment-6fdbb596db-wsxwc   1/1       Running   0          22s
root@k8s1:/root/learning#
```

- Check the deployments workflow

```
kubectl describe deployments
```

- Check the rollout history of the deployments

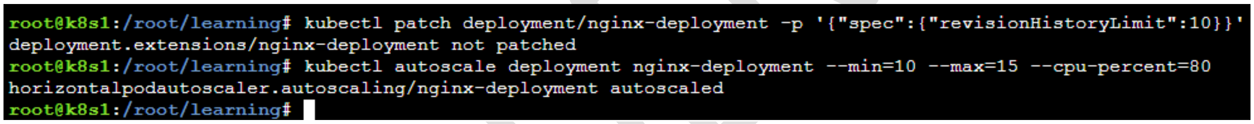
```
kubectl rollout history deployment/nginx-deployment
```



```
root@k8s1:/root/learning# kubectl rollout history deployment/nginx-deployment
deployments "nginx-deployment"
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=deploy1.yml --record=true
2         kubectl apply --filename=deploy1.yml --record=true

root@k8s1:/root/learning#
```

- To change the revision limit

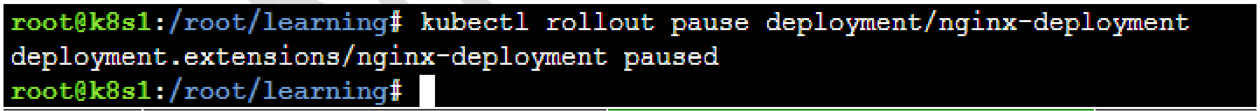
```
kubectl patch deployment/nginx-deployment -p '{"spec":{"revisionHistoryLimit":
10}}'
```



```
root@k8s1:/root/learning# kubectl patch deployment/nginx-deployment -p '{"spec":{"revisionHistoryLimit":10}}'
deployment.extensions/nginx-deployment not patched
root@k8s1:/root/learning# kubectl autoscale deployment nginx-deployment --min=10 --max=15 --cpu-percent=80
horizontalpodautoscaler.autoscaling/nginx-deployment autoscaled
root@k8s1:/root/learning#
```

- Scaling a deployment

```
kubectl autoscale deployment nginx-deployment --min=10 --max=15 --cpu-percent=
80
```



```
root@k8s1:/root/learning# kubectl rollout pause deployment/nginx-deployment
deployment.extensions/nginx-deployment paused
root@k8s1:/root/learning#
```

- Pause a deployment

```
kubectl rollout pause deployment/nginx-deployment
```

- Change the image version names

```
kubectl set image deploy/nginx-deployment nginx=nginx:1.9.4
```

- Resume a deployment

```
kubectl rollout resume deployment/nginx-deployment
```

- Waiting time before declaring as dead

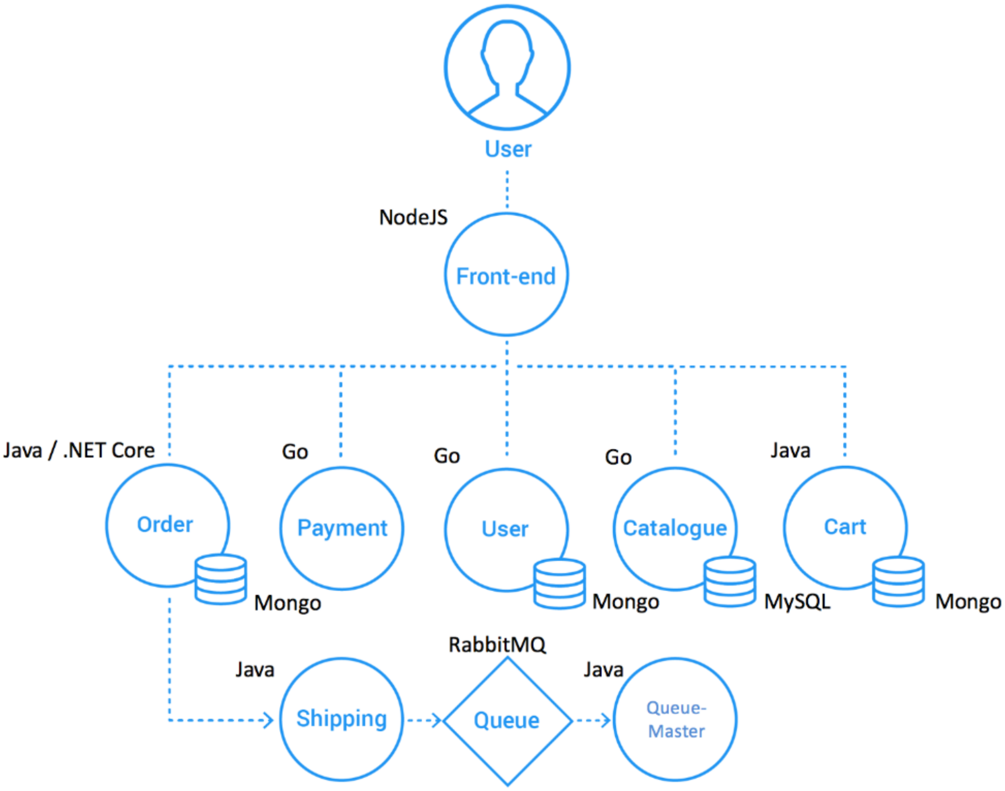
```
kubectl patch deployment/nginx-deployment -p '{"spec":{"progressDeadlineSeconds":600}}'
```

Exercise-28

Objective

In this set of exercises, we will deploy a simple multi-tier application name sock-shop on the kubernetes cluster

Sock-Shop Architecture Overview



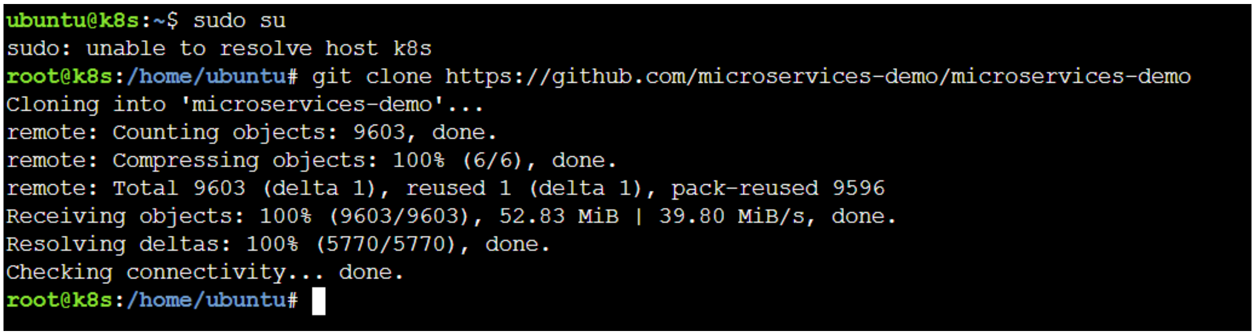
Sock Shop is a multi-tier e-commerce site which is designed to test container based environments. All the components of sock-shop are designed to be containers which can be deployed in our kubernetes cluster.

Clone Sock Shop Deployment Repo

On **k8s1** node

Clone the repository:

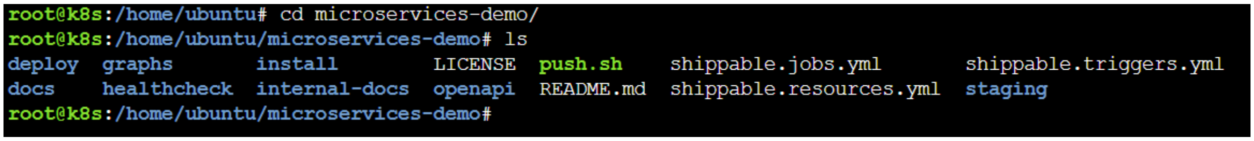
```
cd /home/ubuntu/

git clone https://github.com/microservices-demo/microservices-demo.git
```



```
ubuntu@k8s:~$ sudo su
sudo: unable to resolve host k8s
root@k8s:/home/ubuntu# git clone https://github.com/microservices-demo/microservices-demo
Cloning into 'microservices-demo'...
remote: Counting objects: 9603, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 9603 (delta 1), reused 1 (delta 1), pack-reused 9596
Receiving objects: 100% (9603/9603), 52.83 MiB | 39.80 MiB/s, done.
Resolving deltas: 100% (5770/5770), done.
Checking connectivity... done.
root@k8s:/home/ubuntu# 
```

Go to the repository:

```
cd microservices-demo

ls
```



```
root@k8s:/home/ubuntu# cd microservices-demo/
root@k8s:/home/ubuntu/microservices-demo# ls
deploy  graphs       install        LICENSE  push.sh    shippable.jobs.yml       shippable.triggers.yml
docs    healthcheck  internal-docs  openapi  README.md  shippable.resources.yml  staging
root@k8s:/home/ubuntu/microservices-demo#
```

Create namespace and deploy application

- Clone the repository

```
cd /home/ubuntu/

git clone https://github.com/microservices-demo/microservices-demo.git
```

- Go to the repository:

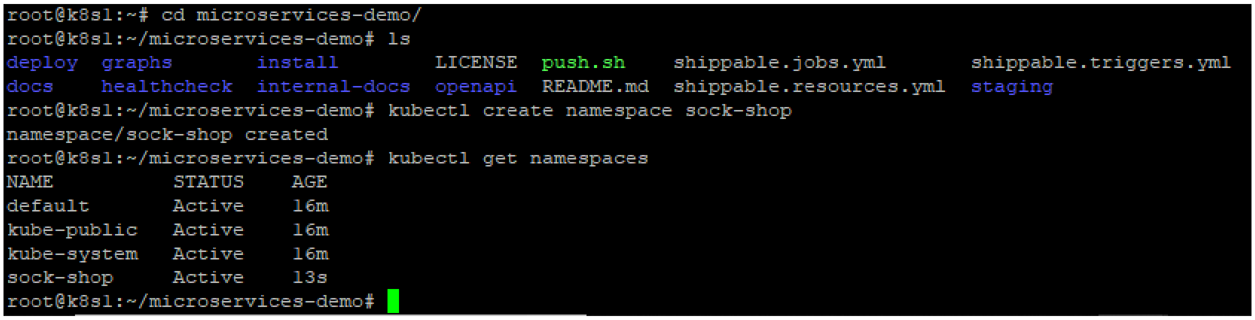
```
cd /home/ubuntu/microservices-demo

ls
```

- Create new namespace sock-shop:

```
kubectl create namespace sock-shop
```

- To verify and list all the namespaces:

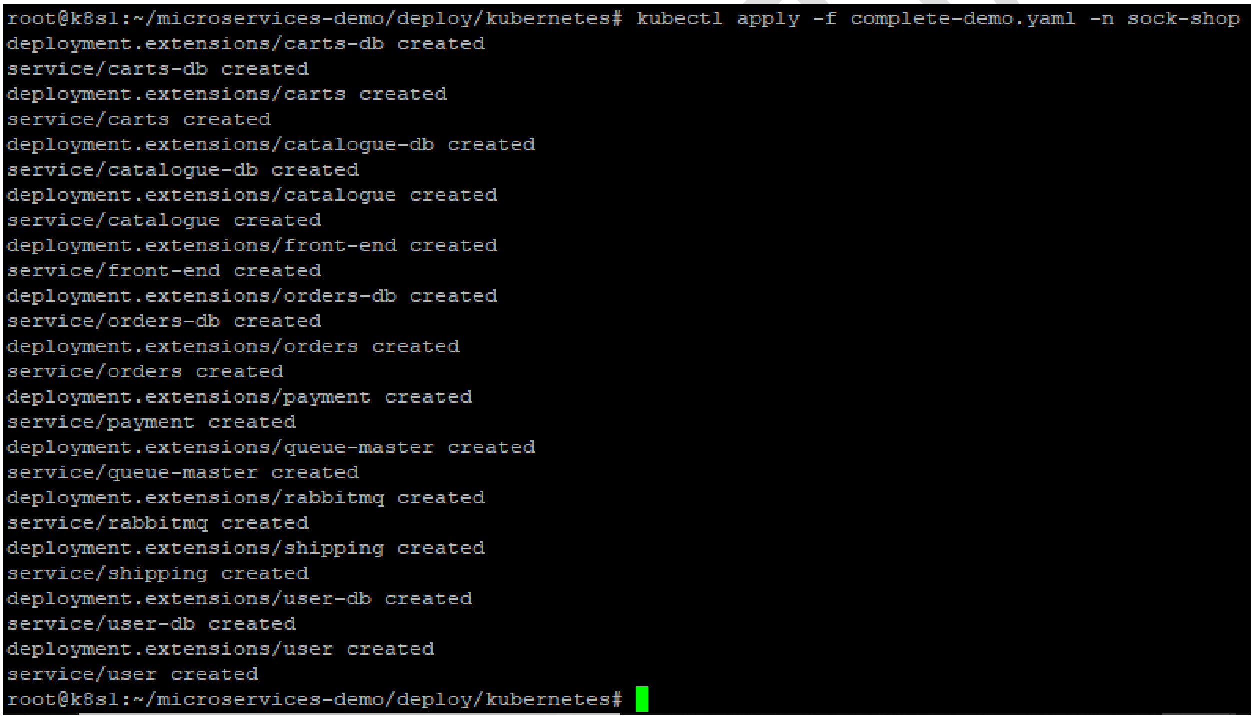
```
kubectl get namespaces
```



```
root@k8s1:~# cd microservices-demo/
root@k8s1:~/microservices-demo# ls
deploy  graphs       install        LICENSE  push.sh    shippable.jobs.yml       shippable.triggers.yml
docs    healthcheck  internal-docs  openapi  README.md  shippable.resources.yml  staging
root@k8s1:~/microservices-demo# kubectl create namespace sock-shop
namespace/sock-shop created
root@k8s1:~/microservices-demo# kubectl get namespaces
NAME          STATUS    AGE
default       Active    16m
kube-public   Active    16m
kube-system   Active    16m
sock-shop     Active    13s
root@k8s1:~/microservices-demo#
```

- Deploy sock-shop application:

```
cd /home/ubuntu/microservices-demo/deploy/kubernetes/

kubectl apply -f complete-demo.yaml -n sock-shop
```

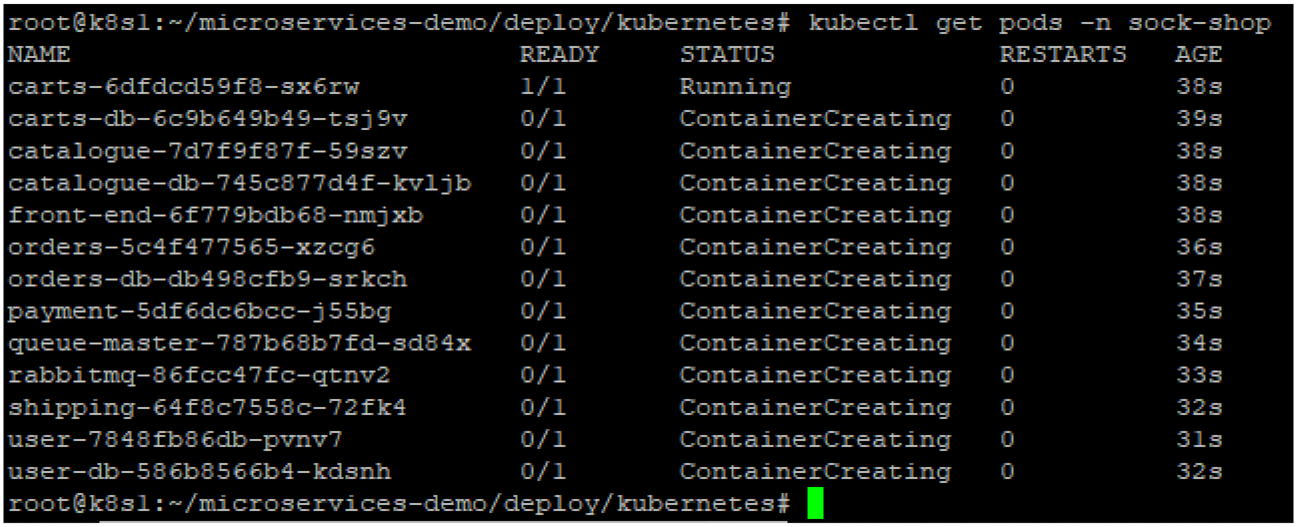


```
root@k8s1:~/microservices-demo/deploy/kubernetes# kubectl apply -f complete-demo.yaml -n sock-shop
deployment.extensions/carts-db created
service/carts-db created
deployment.extensions/carts created
service/carts created
deployment.extensions/catalogue-db created
service/catalogue-db created
deployment.extensions/catalogue created
service/catalogue created
deployment.extensions/front-end created
service/front-end created
deployment.extensions/orders-db created
service/orders-db created
deployment.extensions/orders created
service/orders created
deployment.extensions/payment created
service/payment created
deployment.extensions/queue-master created
service/queue-master created
deployment.extensions/rabbitmq created
service/rabbitmq created
deployment.extensions/shipping created
service/shipping created
deployment.extensions/user-db created
service/user-db created
deployment.extensions/user created
service/user created
root@k8s1:~/microservices-demo/deploy/kubernetes#
```

Verify Pods Status

- To get the details of pods running in sock-shop namespace:

```
kubectl get pods -n sock-shop
```



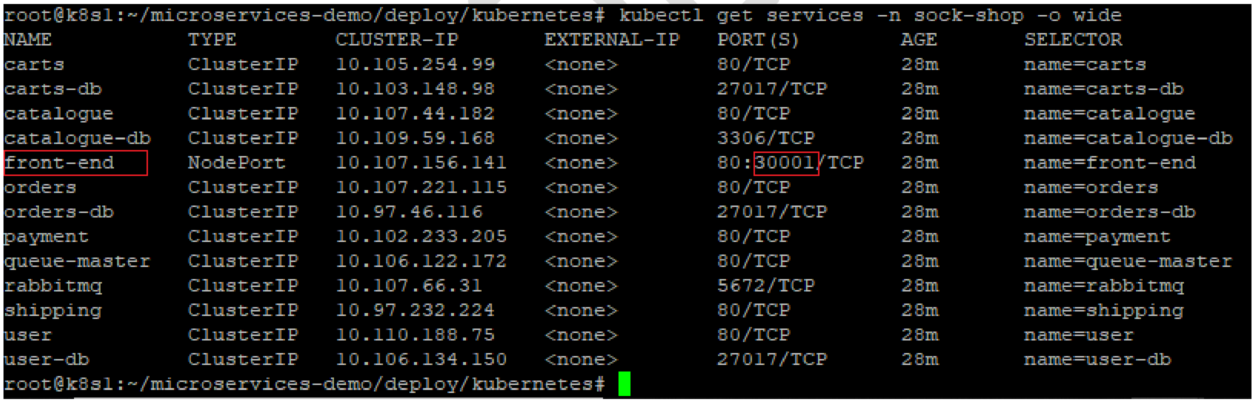
```
root@k8s1:~/microservices-demo/deploy/kubernetes# kubectl get pods -n sock-shop
NAME                                READY     STATUS              RESTARTS   AGE
carts-6dfdcd59f8-sx6rw              1/1       Running             0          38s
carts-db-6c9b649b49-tsj9v           0/1       ContainerCreating   0          39s
catalogue-7d7f9f87f-59szv           0/1       ContainerCreating   0          38s
catalogue-db-745c877d4f-kvljb       0/1       ContainerCreating   0          38s
front-end-6f779bdb68-nmjxb          0/1       ContainerCreating   0          38s
orders-5c4f477565-xzcg6             0/1       ContainerCreating   0          36s
orders-db-db498cfb9-srkch           0/1       ContainerCreating   0          37s
payment-5df6dc6bcc-j55bg            0/1       ContainerCreating   0          35s
queue-master-787b68b7fd-sd84x       0/1       ContainerCreating   0          34s
rabbitmq-86fcc47fc-qtnv2            0/1       ContainerCreating   0          33s
shipping-64f8c7558c-72fk4           0/1       ContainerCreating   0          32s
user-7848fb86db-pvnv7               0/1       ContainerCreating   0          31s
user-db-586b8566b4-kdsnh            0/1       ContainerCreating   0          32s
root@k8s1:~/microservices-demo/deploy/kubernetes#
```

[Here we can see that pods are still building.]

Note: Please check the status after 2 to 3 minutes:]

- You can also get the services running over sock-shop namespaces.

```
kubectl get services -n sock-shop -o wide
```

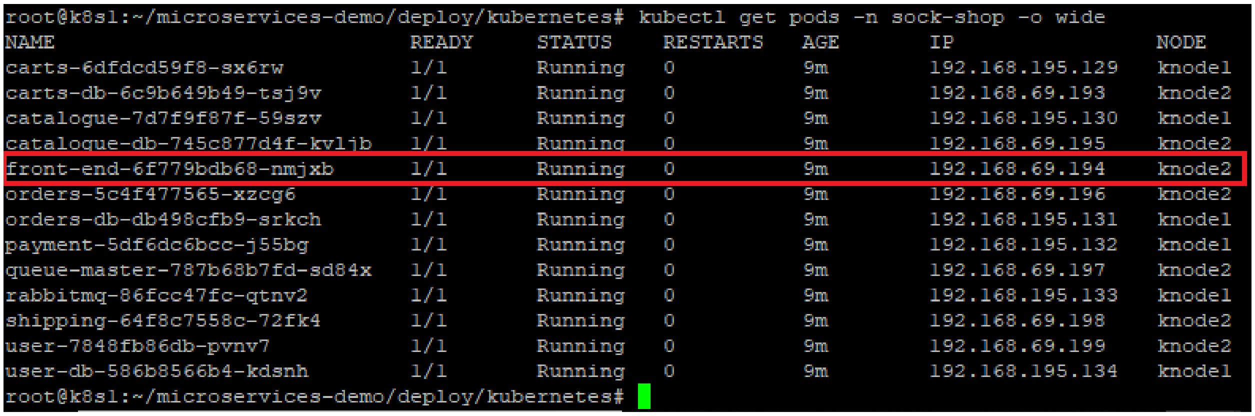


```
root@k8s1:~/microservices-demo/deploy/kubernetes# kubectl get services -n sock-shop -o wide
NAME           TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE       SELECTOR
carts          ClusterIP   10.105.254.99    <none>        80/TCP         28m       name=carts
carts-db       ClusterIP   10.103.148.98    <none>        27017/TCP      28m       name=carts-db
catalogue      ClusterIP   10.107.44.182    <none>        80/TCP         28m       name=catalogue
catalogue-db   ClusterIP   10.109.59.168    <none>        3306/TCP       28m       name=catalogue-db
front-end      NodePort    10.107.156.141   <none>        80:30001/TCP   28m       name=front-end
orders         ClusterIP   10.107.221.115   <none>        80/TCP         28m       name=orders
orders-db      ClusterIP   10.97.46.116     <none>        27017/TCP      28m       name=orders-db
payment        ClusterIP   10.102.233.205   <none>        80/TCP         28m       name=payment
queue-master   ClusterIP   10.106.122.172   <none>        80/TCP         28m       name=queue-master
rabbitmq       ClusterIP   10.107.66.31     <none>        5672/TCP       28m       name=rabbitmq
shipping       ClusterIP   10.97.232.224    <none>        80/TCP         28m       name=shipping
user           ClusterIP   10.110.188.75    <none>        80/TCP         28m       name=user
user-db        ClusterIP   10.106.134.150   <none>        27017/TCP      28m       name=user-db
root@k8s1:~/microservices-demo/deploy/kubernetes#
```

Note: Get the node port for the front-end service as highlighted in below screenshot

- Verify that all the pods came to running state and fetch the node name

```
kubectl get pods -n sock-shop -o wide
```

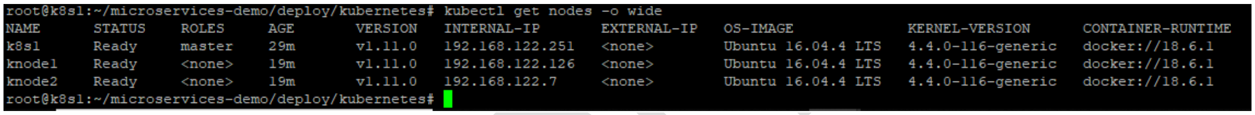


```
root@k8s1:~/microservices-demo/deploy/kubernetes# kubectl get pods -n sock-shop -o wide
NAME                                READY     STATUS    RESTARTS   AGE       IP                NODE
carts-6dfdcd59f8-sx6rw              1/1       Running   0          9m        192.168.195.129   knode1
carts-db-6c9b649b49-tsj9v           1/1       Running   0          9m        192.168.69.193    knode2
catalogue-7d7f9f87f-59szv           1/1       Running   0          9m        192.168.195.130   knode1
catalogue-db-745c877d4f-kvljb       1/1       Running   0          9m        192.168.69.195    knode2
front-end-6f779bdb68-nmjxb          1/1       Running   0          9m        192.168.69.194    knode2
orders-5c4f477565-xzcg6             1/1       Running   0          9m        192.168.69.196    knode2
orders-db-db498cfb9-srkch           1/1       Running   0          9m        192.168.195.131   knode1
payment-5df6dc6bcc-j55bg            1/1       Running   0          9m        192.168.195.132   knode1
queue-master-787b68b7fd-sd84x       1/1       Running   0          9m        192.168.69.197    knode2
rabbitmq-86fcc47fc-qtnv2            1/1       Running   0          9m        192.168.195.133   knode1
shipping-64f8c7558c-72fk4           1/1       Running   0          9m        192.168.69.198    knode2
user-7848fb86db-pvnv7               1/1       Running   0          9m        192.168.69.199    knode2
user-db-586b8566b4-kdsnh            1/1       Running   0          9m        192.168.195.134   knode1
root@k8s1:~/microservices-demo/deploy/kubernetes#
```

Note: Get the node name against the front-end pod

- Get the IP of the above captured node

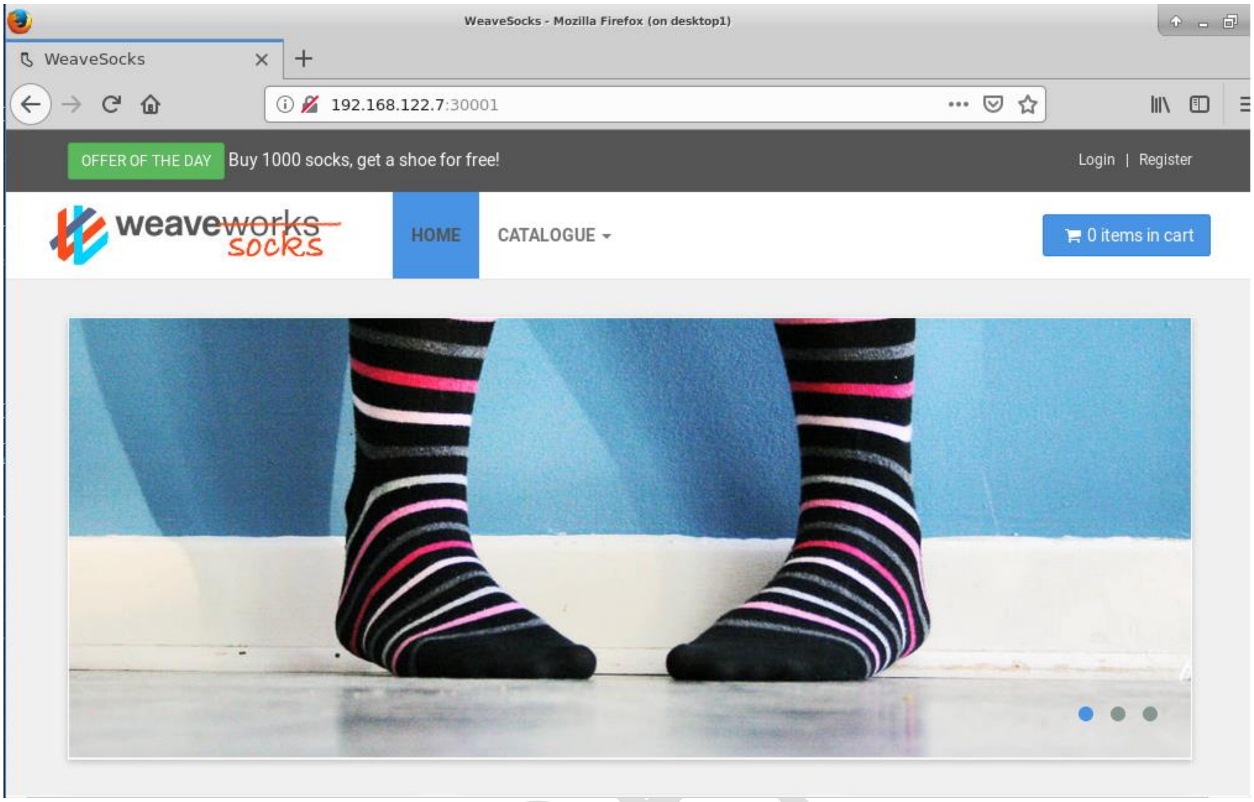
```
kubectl get nodes -o wide
```



```
root@k8s1:~/microservices-demo/deploy/kubernetes# kubectl get nodes -o wide
NAME      STATUS    ROLES     AGE       VERSION   INTERNAL-IP       EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
k8s1      Ready     master    29m       v1.11.0   192.168.122.251   <none>        Ubuntu 16.04.4 LTS   4.4.0-116-generic   docker://18.6.1
knode1    Ready     <none>    19m       v1.11.0   192.168.122.126   <none>        Ubuntu 16.04.4 LTS   4.4.0-116-generic   docker://18.6.1
knode2    Ready     <none>    19m       v1.11.0   192.168.122.7     <none>        Ubuntu 16.04.4 LTS   4.4.0-116-generic   docker://18.6.1
root@k8s1:~/microservices-demo/deploy/kubernetes#
```

- Go to the Desktop VNC session, open the browser and browse the below URL

```
http://<node-ip>:<node-port>
```


WeaveSocks - Mozilla Firefox (on desktop1)
WeaveSocks
192.168.122.7:30001
OFFER OF THE DAY
Buy 1000 socks, get a shoe for free!
Login | Register
weaveworks socks
HOME
CATALOGUE
0 items in cart

Exercise-29

Config Map

Problem Statement

Let’s us consider the use case where some environment variables such as username and password for a database are to be used in multiple replication controller or pod definition files. The username and password value would need to be specified in each of the definition files and if the username and password were to change, all the definition files would need to be updated as well, which could be very tedious.

Alternatively, variable values could be supplied to kubectl when a command is run, which involves specifying command-line flags each time the command is run.

Solution

Here comes the usage of Kubernetes ConfigMap management pattern which is a map of configuration properties that can be used in definition files for pods, replication controllers, and other Kubernetes objects to configure

- environment variables,
- command arguments, and
- configuration files such as key-value pairs in volumes, etc.

A single ConfigMap may package multiple configuration properties as key/value pairs. By creating ConfigMaps, you specify the configuration properties in a single configuration map, which can be updated as required without having to update each of the definition files in which the ConfigMap is used.

Decoupling the containers from the configuration data provides portability of the applications running in the containers.

Exercise-30

Config map use case

Create config file

The following configuration file will create a configuration file that keeps a list of addresses, open a vim editor with a file name **configmap.yaml**

```
cd /home/ubuntu/learning/

vim configmap.yaml
```

Then copy and paste the below configuration in the opened vim editor

```
---

apiVersion: v1

kind: ConfigMap

metadata:

     name: db-config

     namespace: default

data:

     db-ip-addresses: 1.2.3.4,5.6.7.8
```

To create the config use the below command

```
kubectl create -f configmap.yaml
```

Below message will be displayed

configmap/db-config created

The data section contains all the key-value pairs, in this case, just a single pair with a key name of db-ip-addresses. It will be important later when consuming the configmap in a pod.

You can check out the content to make sure configmap is created:

```
kubectl get configmap db-config -o yaml
```

Output will be displayed as below:

```
apiVersion: v1
data:
    db-ip-addresses: 1.2.3.4,5.6.7.8
kind: ConfigMap
metadata:
    creationTimestamp: 2018-11-23T07:32:52Z
    name: db-config
    namespace: default
    resourceVersion: "382141"
    selfLink: /api/v1/namespaces/default/configmaps/db-config
    uid: fc440309-eef1-11e8-b862-5254006dd4e6
```

Consuming a ConfigMap as an environment variable

When we are creating a pod, we can specify a ConfigMap and consume its values.

```
vim mypod1.yaml
```

Go to insert mode (by pressing "i") and copy-paste the below configuration in this file

Here is how to consume our configuration map as an environment variable:

```
apiVersion: v1
kind: Pod
metadata:
    name: some-pod
spec:
    containers:
        - name: some-container
```

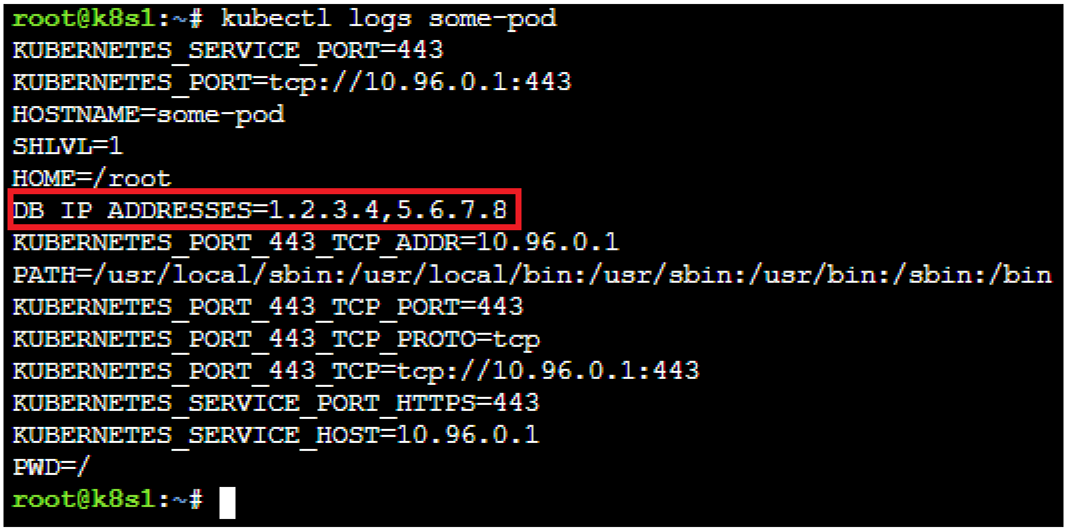
```
             image: busybox

             command: [ "/bin/sh", "-c", "env" ]

             env:

                 - name: DB_IP_ADDRESSES

                     valueFrom:

                         configMapKeyRef:

                             name: db-config

                             key: db-ip-addresses

     restartPolicy: Never
```

Create pod with above config

```
kubectl create -f mypod1.yaml
```

This pod runs the busybox minimal container and executes an env bash command and immediately exists. The db-ip-addresses key from the db-config map is mapped to the DB_IP_ADDRESSES environment variable, and is reflected in the output:

```
kubectl logs some-pod
```



```
root@k8s1:~# kubectl logs some-pod
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT=tcp://10.96.0.1:443
HOSTNAME=some-pod
SHLVL=1
HOME=/root
DB_IP_ADDRESSES=1.2.3.4,5.6.7.8
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_SERVICE_HOST=10.96.0.1
PWD=/
root@k8s1:~#
```

Exercise-31

Starting the Kubernetes UI application

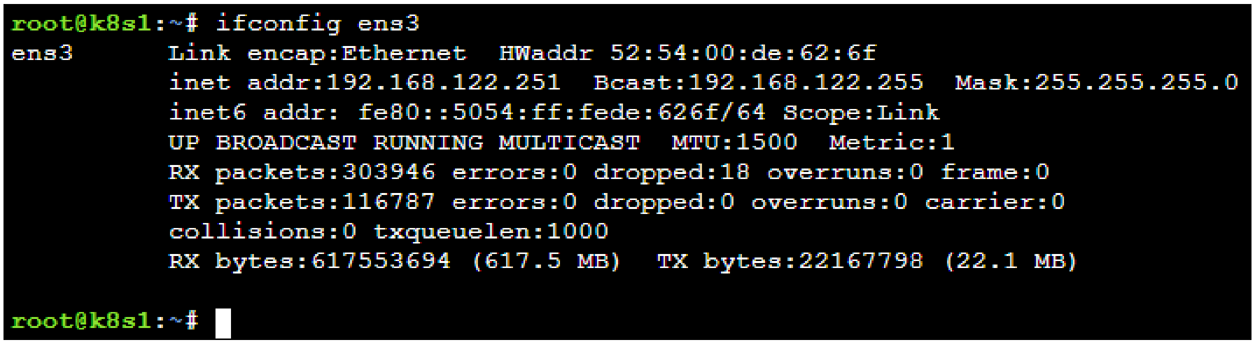
Starting the Dashboard application

[The below commands has to be executed in k8s1 node]

Get the IP of the k8s1 node using below command

```
ifconfig ens3
```

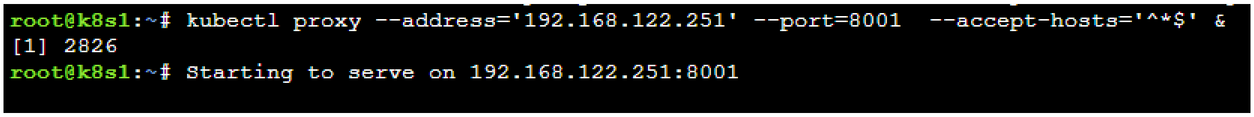
Mostly the IP will be "192.168.122.251", but verify in your setup once.



```
root@k8s1:~# ifconfig ens3
ens3      Link encap:Ethernet  HWaddr 52:54:00:de:62:6f
          inet addr:192.168.122.251  Bcast:192.168.122.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fede:626f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:303946 errors:0 dropped:18 overruns:0 frame:0
          TX packets:116787 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:617553694 (617.5 MB)  TX bytes:22167798 (22.1 MB)

root@k8s1:~# 
```

Start the dashboard application using proxy server commmand

```
kubectl proxy --address='192.168.122.251' --port=8001  --accept-hosts='^*$' &
```



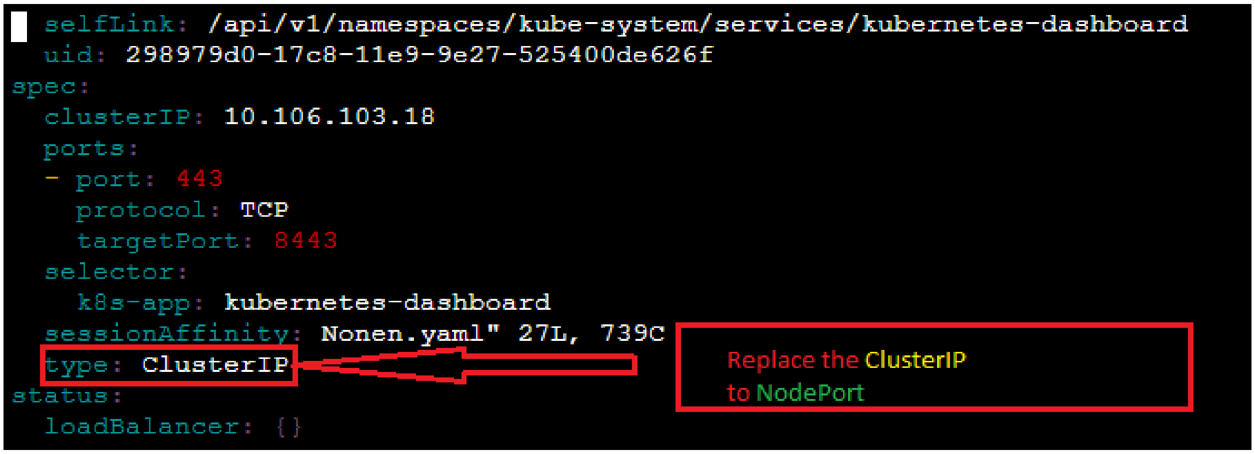
```
root@k8s1:~# kubectl proxy --address='192.168.122.251' --port=8001  --accept-hosts='^*$' &
[1] 2826
root@k8s1:~# Starting to serve on 192.168.122.251:8001
```

NOTE: Check if the IP is same as mentioned in above command, else replace the above mentioned IP with the one which you have obtained using the previous command.

As we will be accessing the dashboard from desktop node, so we have to change the service type to **NodePort** from **ClusterIP**

Edit kubernetes-dashboard service

```
kubectl -n kube-system edit service kubernetes-dashboard
```



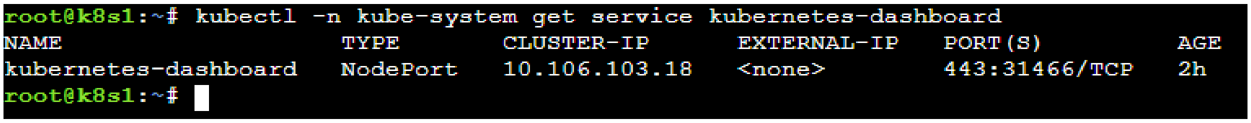
```
  selfLink: /api/v1/namespaces/kube-system/services/kubernetes-dashboard
  uid: 298979d0-17c8-11e9-9e27-525400de626f
spec:
  clusterIP: 10.106.103.18
  ports:
  - port: 443
    protocol: TCP
    targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
  sessionAffinity: Nonen.yaml" 27L, 739C
  type: ClusterIP
status:
  loadBalancer: {}
```

Replace the ClusterIP to NodePort

You should see yaml representation of the service. Change type: ClusterIP to type: NodePort and save file. If it's already changed go to next step.

Next we need to check port on which Dashboard was exposed.

```
kubectl -n kube-system get service kubernetes-dashboard
```



```
root@k8s1:~# kubectl -n kube-system get service kubernetes-dashboard
NAME                   TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)         AGE
kubernetes-dashboard   NodePort   10.106.103.18   <none>        443:31466/TCP   2h
root@k8s1:~#
```

Dashboard has been exposed on port obtained above (HTTPS). Now you can access it from your browser at: https://master-ip:nodeport.

Create service account and cluster tole binding

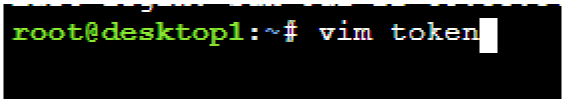
Create cluster admin service account (on *k8s1* node)

```
kubectl create serviceaccount cluster-admin-dashboard-sa

kubectl create clusterrolebinding cluster-admin-dashboard-sa \

    --clusterrole=cluster-admin \

    --serviceaccount=default:cluster-admin-dashboard-sa
```

And then, use the token of just created cluster admin service account.

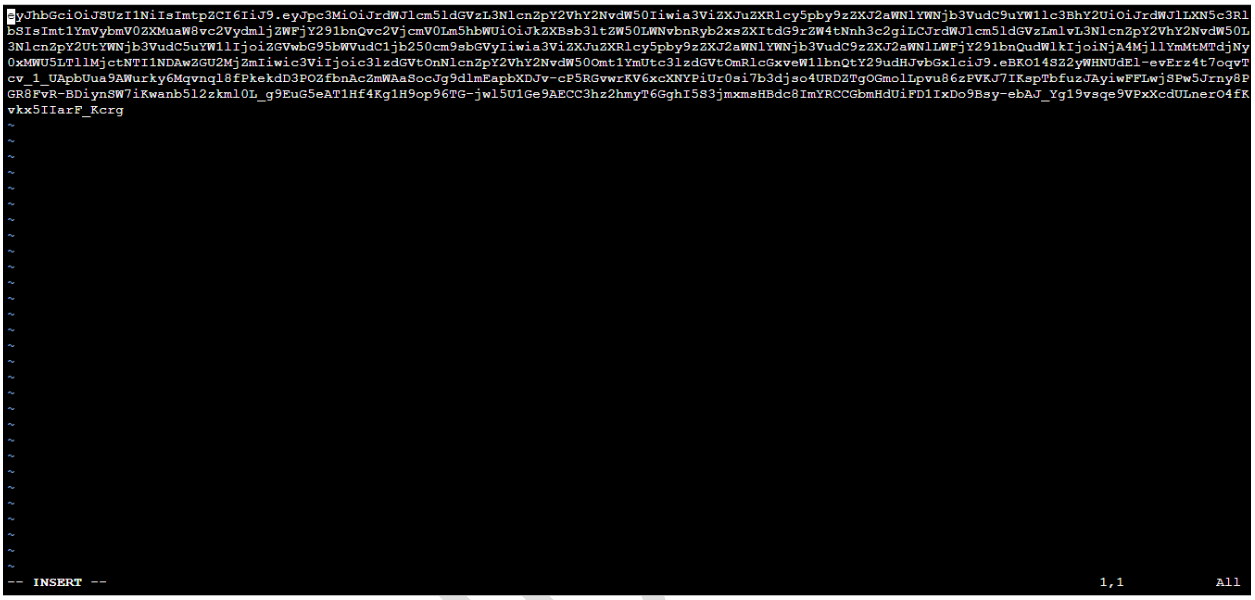
```
kubectl get secret | grep cluster-admin-dashboard-sa

kubectl describe secret <cluster-admin-dashboard-name-obtained-above>
```

Copy this output and open the **desktop node** from the "Access Devices" tab and create a file named token using vim editor



```
root@desktop1:~# vim token
```

Go to the insert mode and paste the token here



```
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3Rl
bSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJkZXBsb3ltZW50LWNvbnRyb2xsZXItdG9rZW4tNnh3c2giLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L
3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiZGVwbG95bWVudC1jb250cm9sbGVyIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQudWlkIjoiNjA4MjllYmMtMTdjNy
0xMWU5LTllMjctNTI1NDAwZGU2MjZmIiwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50Omt1YmUtc3lzdGVtOmRlcGxveW1lbnQtY29udHJvbGxlciJ9.eBKO14SZ2yWHNUdEl-evErz4t7oqvT
cv_1_UApbUua9AWurky6Mqvnql8fPkekdD3POZfbnAcZmWAaSocJg9dlmEapbXDJv-cP5RGvwrKV6xcXNYPiUr0si7b3djso4URDZTgOGmolLpvu86zPVKJ7IKspTbfuzJAyiwFFLwjSPw5Jrny8P
GR8FvR-BDiynSW7iKwanb5l2zkml0L_g9EuG5eAT1Hf4Kg1H9op96TG-jwl5U1Ge9AECC3hz2hmyT6GghI5S3jmxmsHBdc8ImYRCCGbmHdUiFD1IxDo9Bsy-ebAJ_Yg19vsqe9VPxXcdULnerO4fK
vkx5IIarF_Kcrg
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                                                                   1,1           All
```

Exercise-32

Kubernetes Dashboard

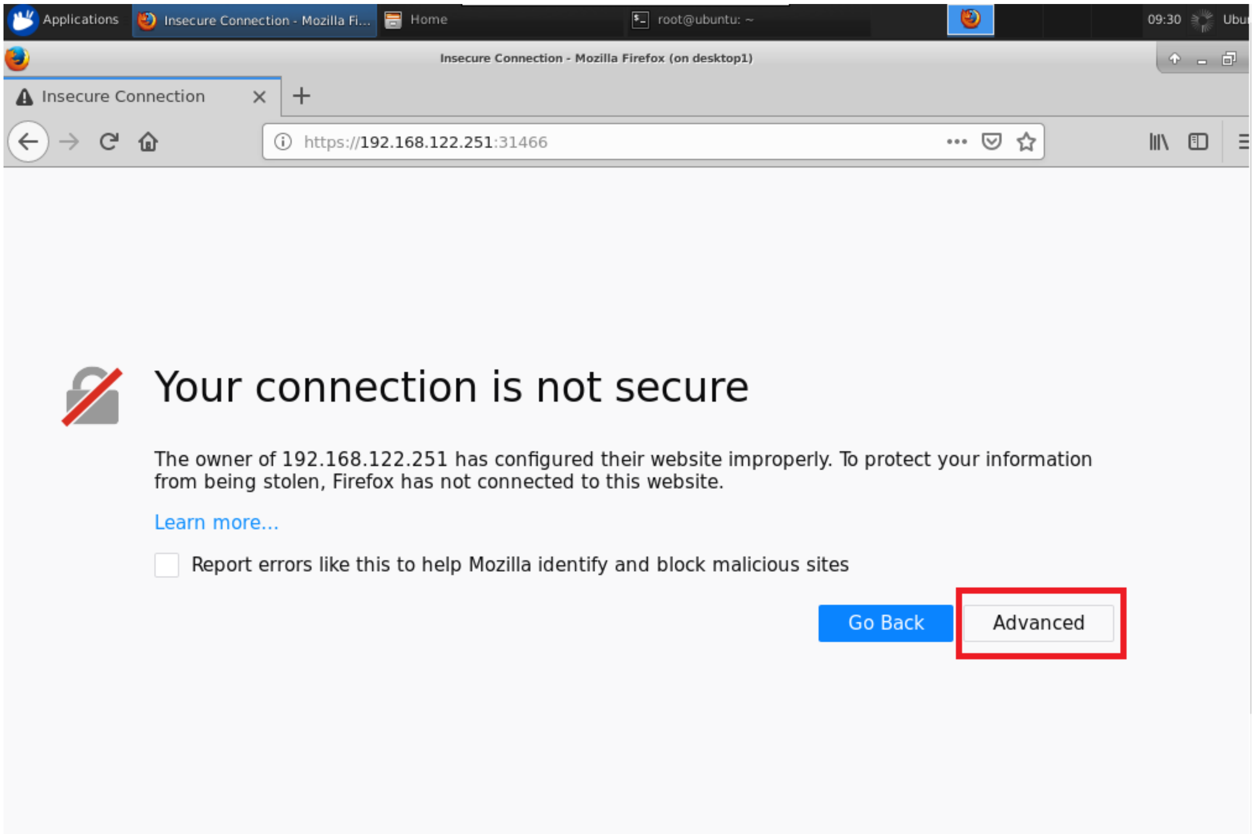
Verify the Dashboard

Go to the desktop's VNC session and open the Mozilla firefox browser and type the below URL

```
https://<k8s1-node-ip>:<kubernetes-dashboard-node-port>
```

NOTE: Kubernetes dashboard node port is the port we have obtained previously.

It will redirect to the below page, click on advanced


Applications
Insecure Connection - Mozilla Fi...
Home
root@ubuntu: ~
09:30
Insecure Connection - Mozilla Firefox (on desktop1)
Insecure Connection
https://192.168.122.251:31466

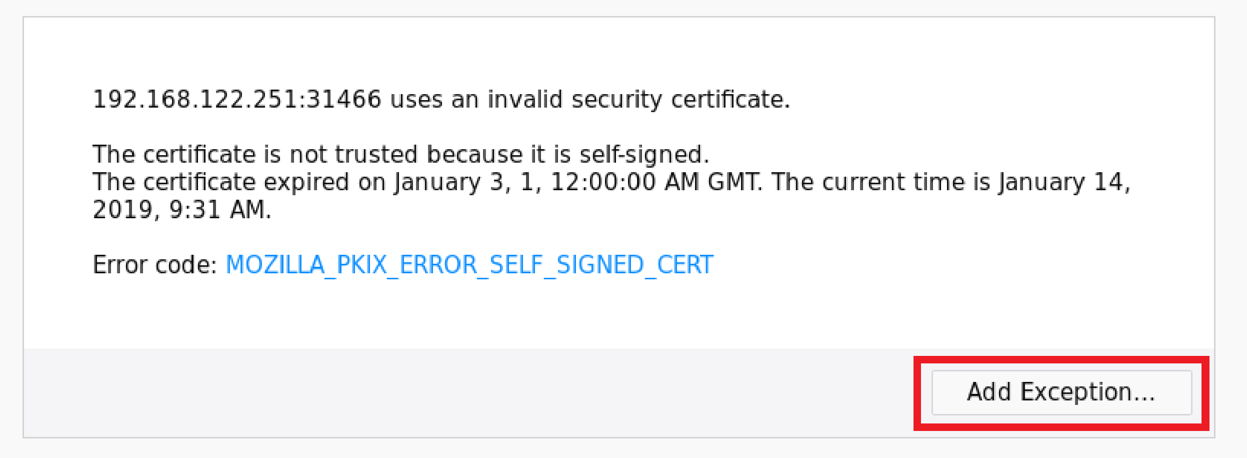
Your connection is not secure

The owner of 192.168.122.251 has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.

Learn more...

□ Report errors like this to help Mozilla identify and block malicious sites

Scroll down and click on "Add Exception"

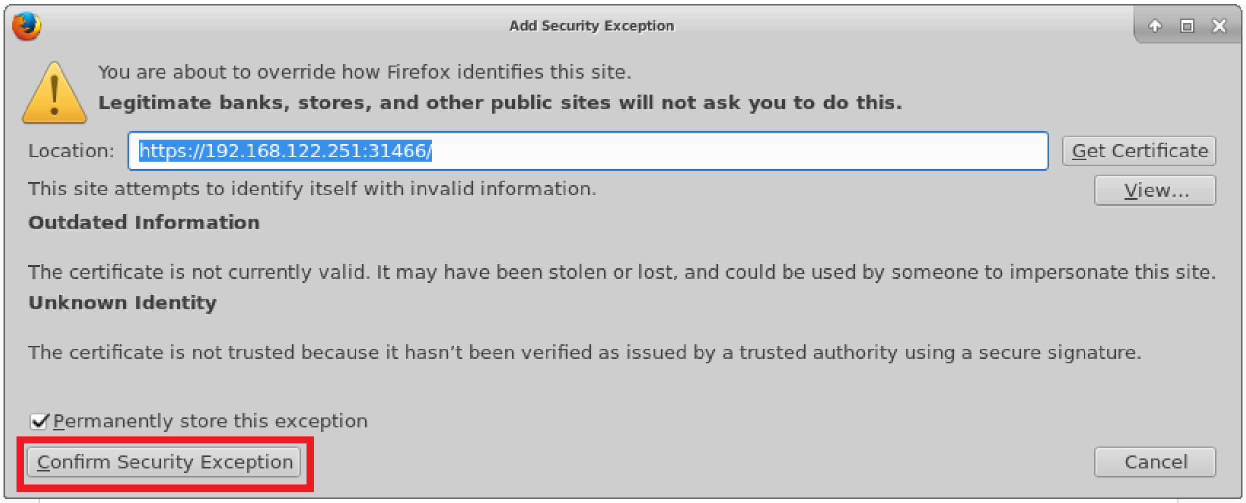


192.168.122.251:31466 uses an invalid security certificate.

The certificate is not trusted because it is self-signed.
The certificate expired on January 3, 1, 12:00:00 AM GMT. The current time is January 14, 2019, 9:31 AM.

Error code: MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT

A window as shown below will pop-up click on "Confirm Security Exception"



Add Security Exception

You are about to override how Firefox identifies this site.
Legitimate banks, stores, and other public sites will not ask you to do this.

Location: https://192.168.122.251:31466/

This site attempts to identify itself with invalid information.

Outdated Information

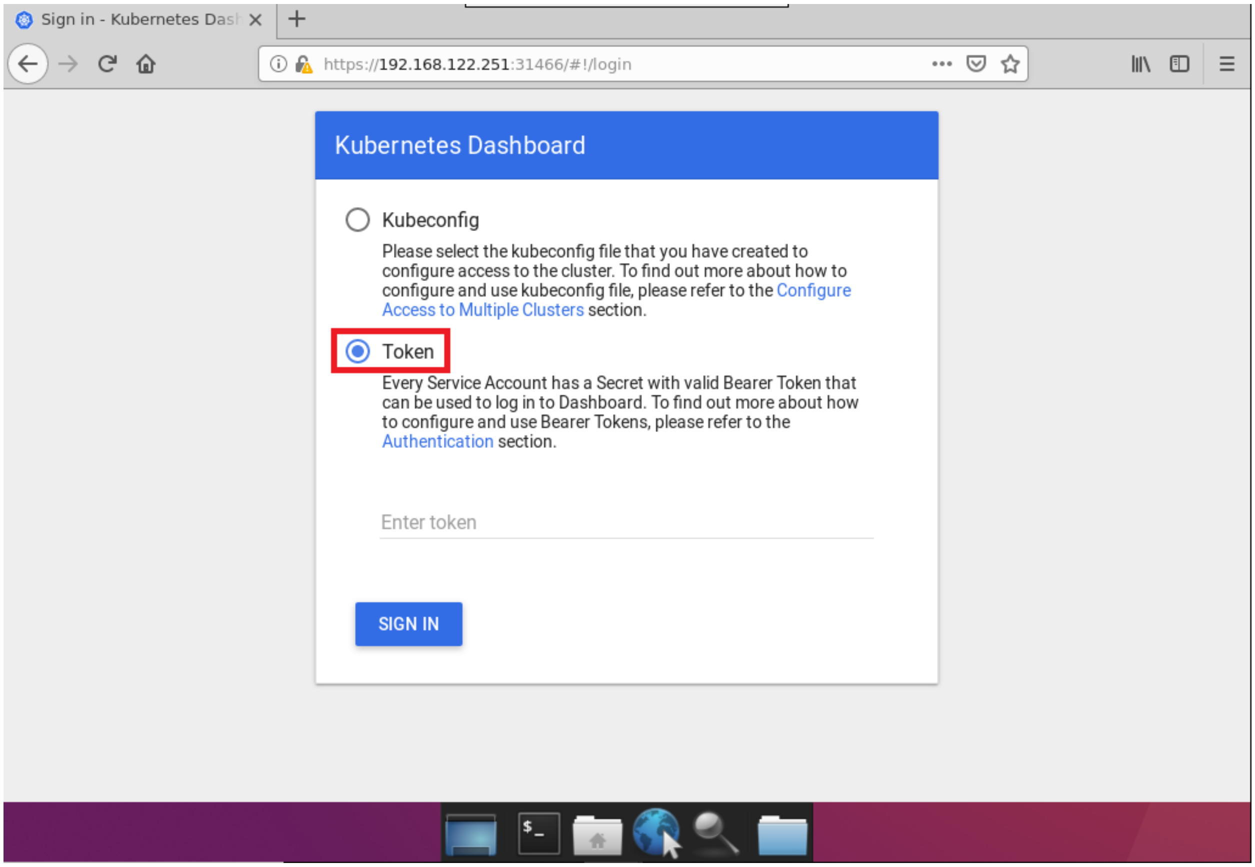
The certificate is not currently valid. It may have been stolen or lost, and could be used by someone to impersonate this site.

Unknown Identity

The certificate is not trusted because it hasn’t been verified as issued by a trusted authority using a secure signature.

☑ Permanently store this exception

The Kubernetes Dashboard's authentication page will open, there select "Token"


Sign in - Kubernetes Dash
https://192.168.122.251:31466/#!/login

Kubernetes Dashboard

○ Kubeconfig

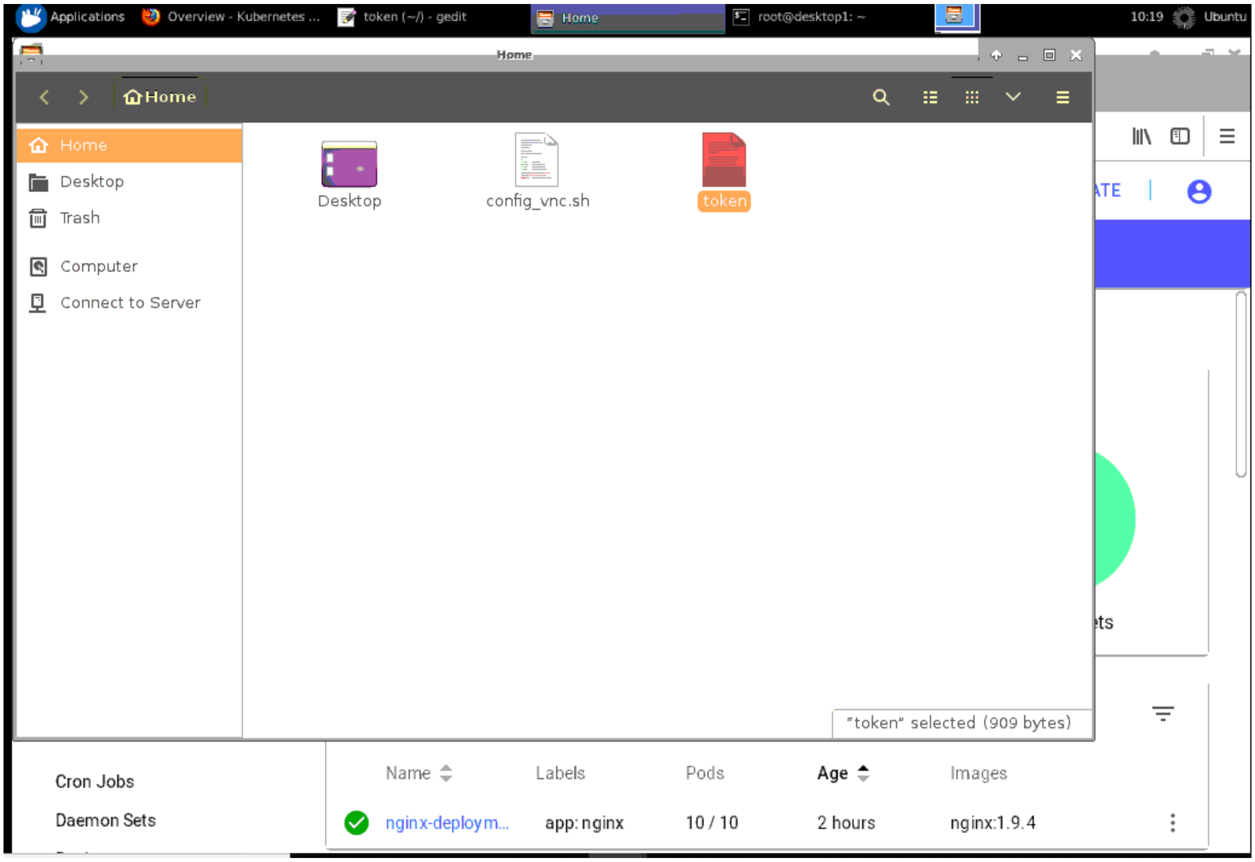
Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.

● Token

Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.

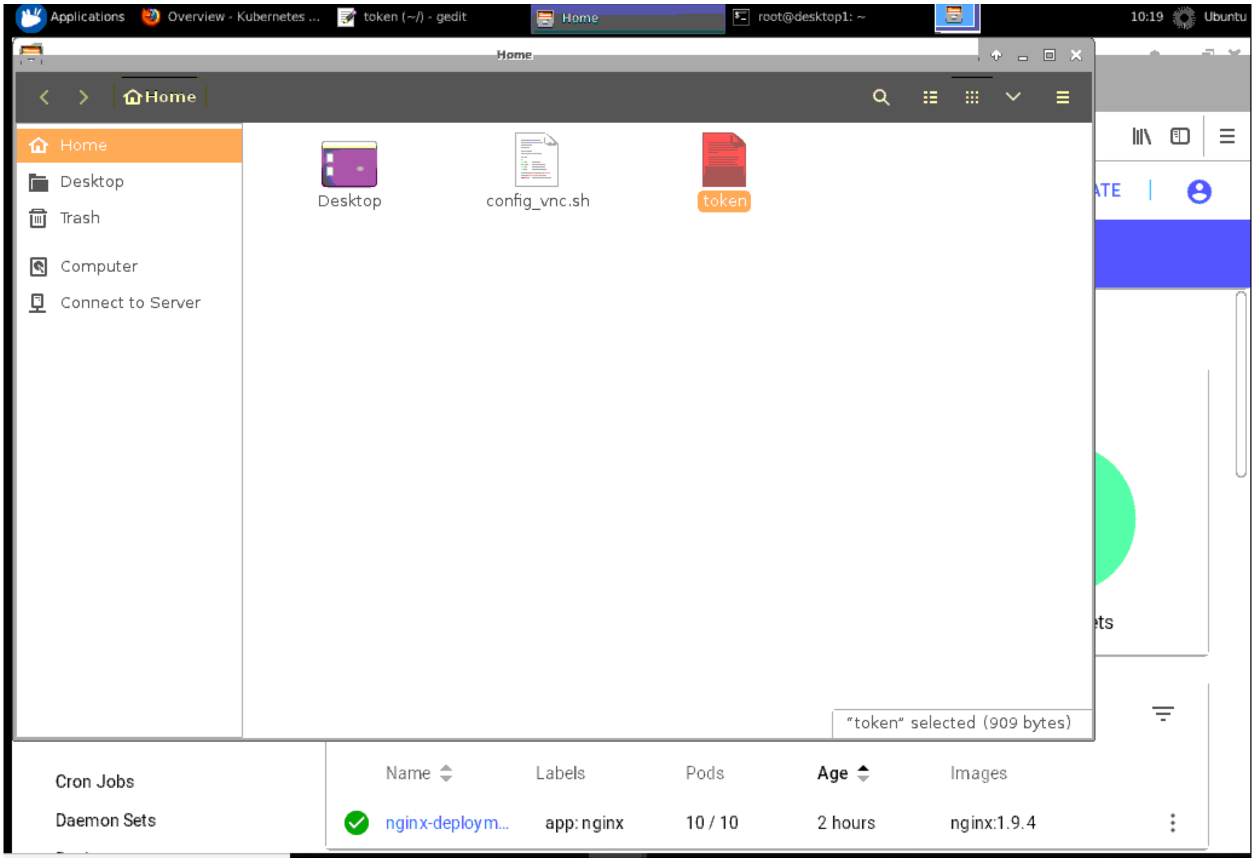
Enter token

Now in the VNC session open the terminal and read the token file which was created earlier in this exercise. Double click on the token file to open it.



Name	Labels	Pods	Age	Images
nginx-deploym...	app: nginx	10 / 10	2 hours	nginx:1.9.4

Copy this token and paste in the Dashboard

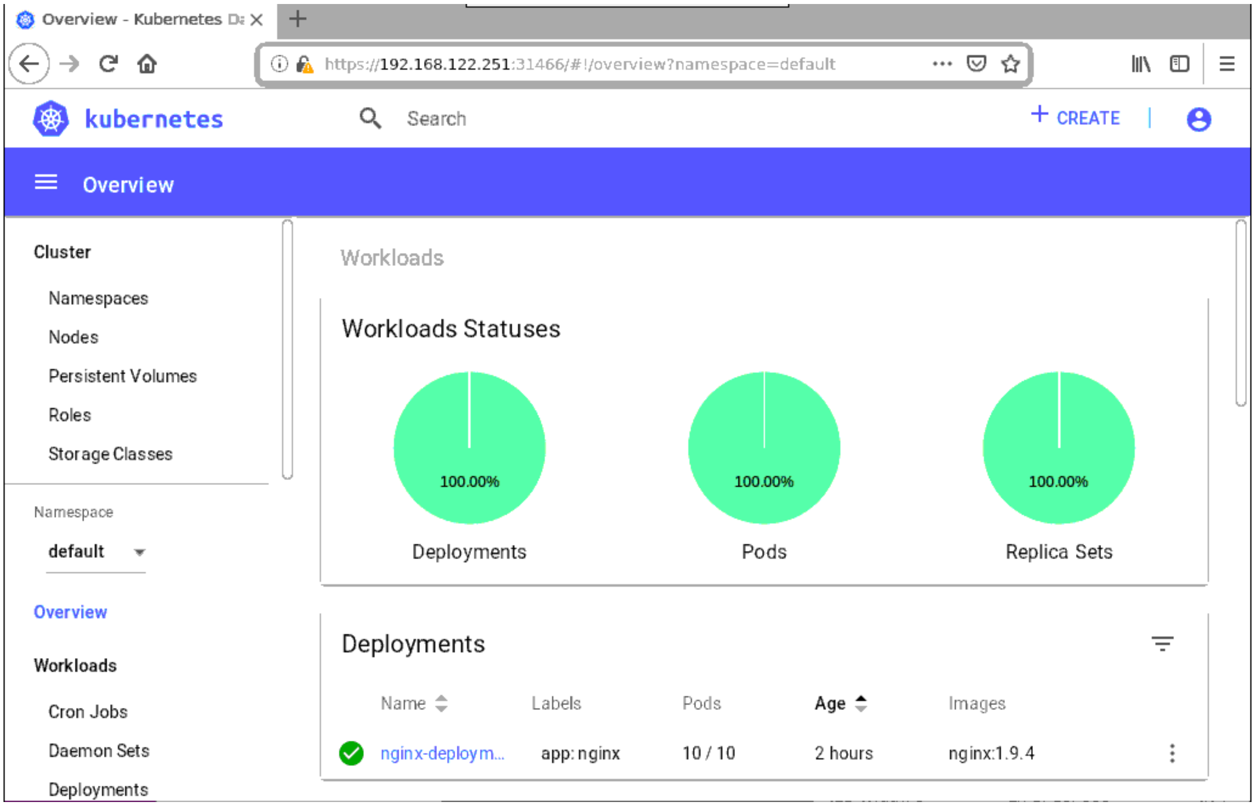


"token" selected (909 bytes)

Name	Labels	Pods	Age	Images
nginx-deploym...	app: nginx	10 / 10	2 hours	nginx:1.9.4

Kubernetes Dashboard overview

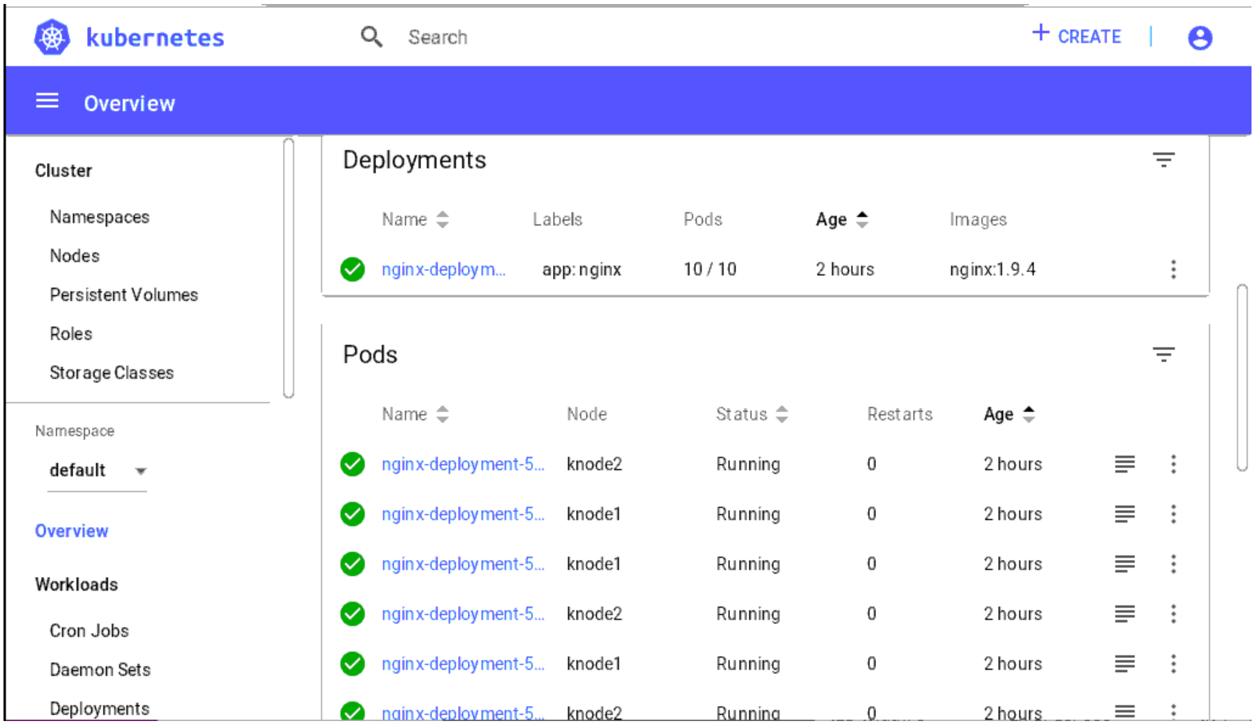
After authentication, dashboard will look like this



Deployments

Name	Labels	Pods	Age	Images
✓ nginx-deploym...	app: nginx	10 / 10	2 hours	nginx:1.9.4

Scrolling down will show you all the deployments, which have been done till now.



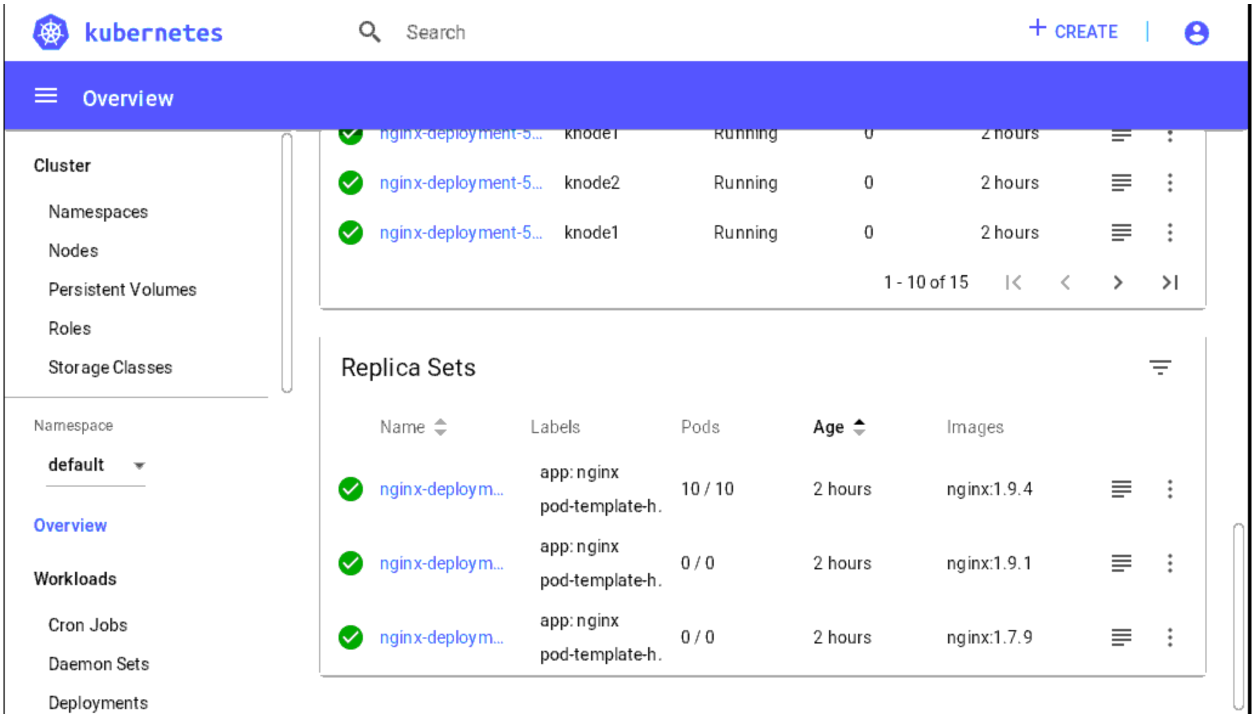
Deployments

Name	Labels	Pods	Age	Images
✓ nginx-deploym...	app: nginx	10 / 10	2 hours	nginx:1.9.4

Pods

Name	Node	Status	Restarts	Age
✓ nginx-deployment-5...	knode2	Running	0	2 hours
✓ nginx-deployment-5...	knode1	Running	0	2 hours
✓ nginx-deployment-5...	knode1	Running	0	2 hours
✓ nginx-deployment-5...	knode2	Running	0	2 hours
✓ nginx-deployment-5...	knode1	Running	0	2 hours
✓ nginx-deployment-5...	knode2	Running	0	2 hours

Further scrolling will show you the replica sets created in previous exercises



✓ nginx-deployment-5...	knode1	Running	0	2 hours
✓ nginx-deployment-5...	knode2	Running	0	2 hours
✓ nginx-deployment-5...	knode1	Running	0	2 hours

1 - 10 of 15

Replica Sets

Name	Labels	Pods	Age	Images
✓ nginx-deploym...	app: nginx pod-template-h.	10 / 10	2 hours	nginx:1.9.4
✓ nginx-deploym...	app: nginx pod-template-h.	0 / 0	2 hours	nginx:1.9.1
✓ nginx-deploym...	app: nginx pod-template-h.	0 / 0	2 hours	nginx:1.7.9

You can further explore the dashboard by verifying the **cluster nodes, namespaces and other side menu options** with the output of the CLI

Docker Part-III(Optional)

Exercise-31

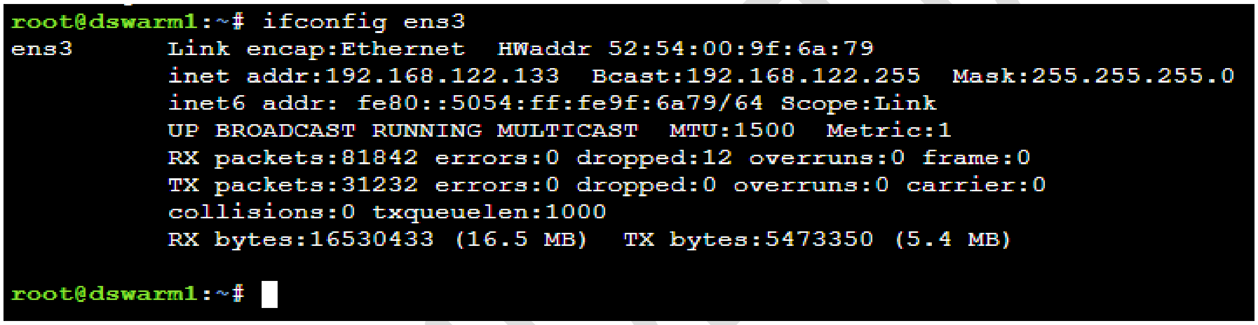
Bring up Docker Swarm Cluster

- Install docker-ce using Exercise-11 on both dnode2 and dswarm

On dswarm1 node:

- Get the IP address of node

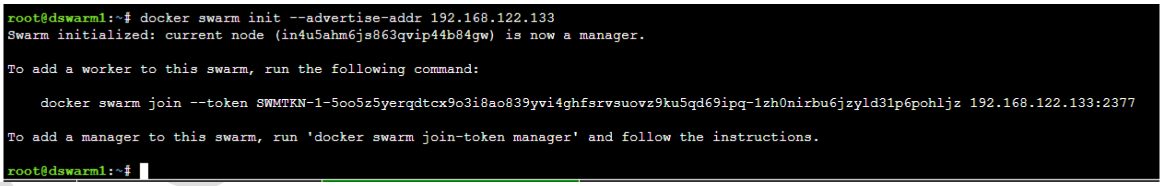
```
ifconfig ens3
```



```
root@dswarm1:~# ifconfig ens3
ens3      Link encap:Ethernet  HWaddr 52:54:00:9f:6a:79
          inet addr:192.168.122.133  Bcast:192.168.122.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe9f:6a79/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:81842 errors:0 dropped:12 overruns:0 frame:0
          TX packets:31232 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16530433 (16.5 MB)  TX bytes:5473350 (5.4 MB)

root@dswarm1:~#
```

- Initialize the docker swarm to advertize this IP address

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



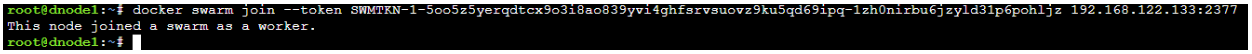
```
root@dswarm1:~# docker swarm init --advertise-addr 192.168.122.133
Swarm initialized: current node (in4u5ahm6js863qvip44b84gw) is now a manager.

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

    docker swarm join --token SWMTKN-1-5oo5z5yerqdtcx9o3i8ao839yvi4ghfsrvsuovz9ku5qd69ipq-1zh0nirbu6jzyld31p6pohljz 192.168.122.133:2377

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

root@dswarm1:~#
```

- Observe the output displayed by above command. We will run this commands on dnode1 and dnode2 to make them part of swarm cluster

On dnode1 node

- Join the dswarm cluster

```
docker swarm join --token <<TOKEN-ID-FROM-ABOVE-OUTPUT>>  <<IP-ADDRESS-OF-DSWARM1>:2377
```

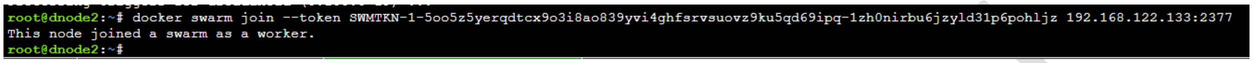


```
root@dnode1:~# docker swarm join --token SWMTKN-1-5oo5z5yerqdtcx9o3i8ao839yvi4ghfsrvsuovz9ku5qd69ipq-1zh0nirbu6jzyld31p6pohljz 192.168.122.133:2377
This node joined a swarm as a worker.
root@dnode1:~#
```

On dnode2 node

- Join the dswarm cluster

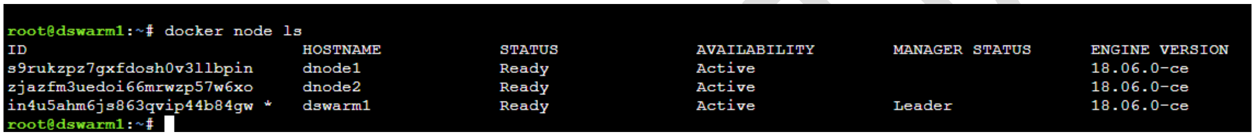
```
docker swarm join --token <<TOKEN-ID-FROM-ABOVE-OUTPUT>>  <<IP-ADDRESS-OF-DSWARM1>:2377
```



```
root@dnode2:~# docker swarm join --token SWMTKN-1-5oo5z5yerqdtcx9o3i8ao839yvi4ghfsrvsuovz9ku5qd69ipq-1zh0nirbu6jzyld31p6pohljz 192.168.122.133:2377
This node joined a swarm as a worker.
root@dnode2:~#
```

- Verify the nodes in the cluster on dswarm1 node

```
docker node ls
```



```
root@dswarm1:~# docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
s9rukzpz7gxfdosh0v3llbpin     dnode1              Ready               Active                                  18.06.0-ce
zjazfm3uedoi66mrwzp57w6xo     dnode2              Ready               Active                                  18.06.0-ce
in4u5ahm6js863qvip44b84gw *   dswarm1             Ready               Active              Leader              18.06.0-ce
root@dswarm1:~#
```

TIP: For debugging in case of any failure, to leave the swarm cluster on a node. [DO NOT LEAVE]

docker swarm leave

Exercise-32

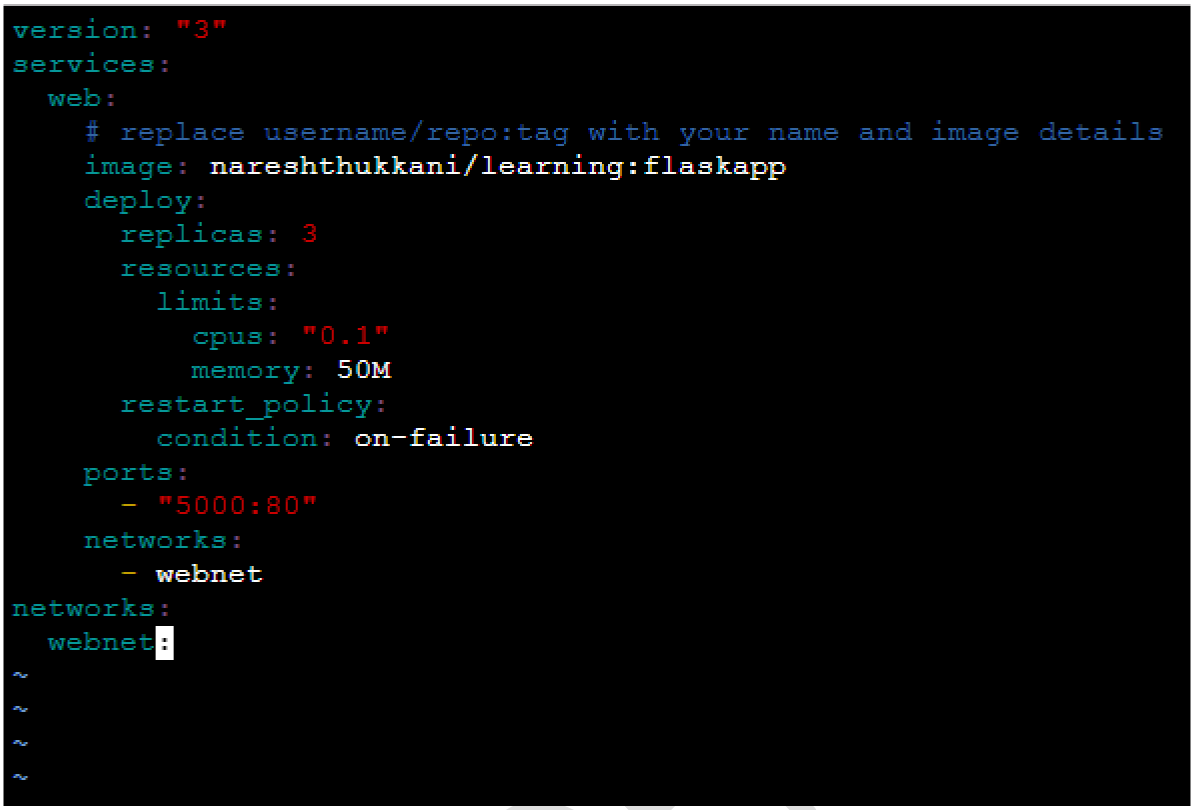
Deploy the App on Docker Swarm Cluster

- On "dswarm1" node, create a file named docker-compose.yml

```
vim docker-compose.yml
```

- Insert the below lines

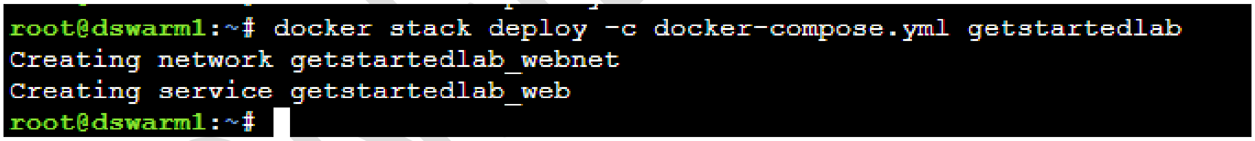
```
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: nareshthukkani/learning:flaskapp
    deploy:
      replicas: 3
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "5000:80"
    networks:
      - webnet
networks:
  webnet:
```



```
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: nareshthukkani/learning:flaskapp
    deploy:
      replicas: 3
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "5000:80"
    networks:
      - webnet
networks:
  webnet:
~
~
~
~
```

- Deploy the stack

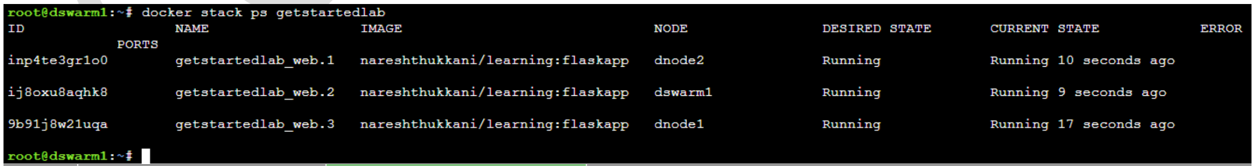
```
docker stack deploy -c docker-compose.yml getstartedlab
```



```
root@dswarm1:~# docker stack deploy -c docker-compose.yml getstartedlab
Creating network getstartedlab_webnet
Creating service getstartedlab_web
root@dswarm1:~#
```

- Verify multiple containers are deployed as per docker-compose.yml

```
docker stack ps getstartedlab
```



```
root@dswarm1:~# docker stack ps getstartedlab
ID                  NAME                  IMAGE                              NODE                DESIRED STATE       CURRENT STATE             ERROR
               PORTS
inp4te3gr1o0        getstartedlab_web.1   nareshthukkani/learning:flaskapp   dnode2              Running             Running 10 seconds ago

ij8oxu8aqhk8        getstartedlab_web.2   nareshthukkani/learning:flaskapp   dswarm1             Running             Running 9 seconds ago

9b91j8w21uqa        getstartedlab_web.3   nareshthukkani/learning:flaskapp   dnode1              Running             Running 17 seconds ago

root@dswarm1:~#
```

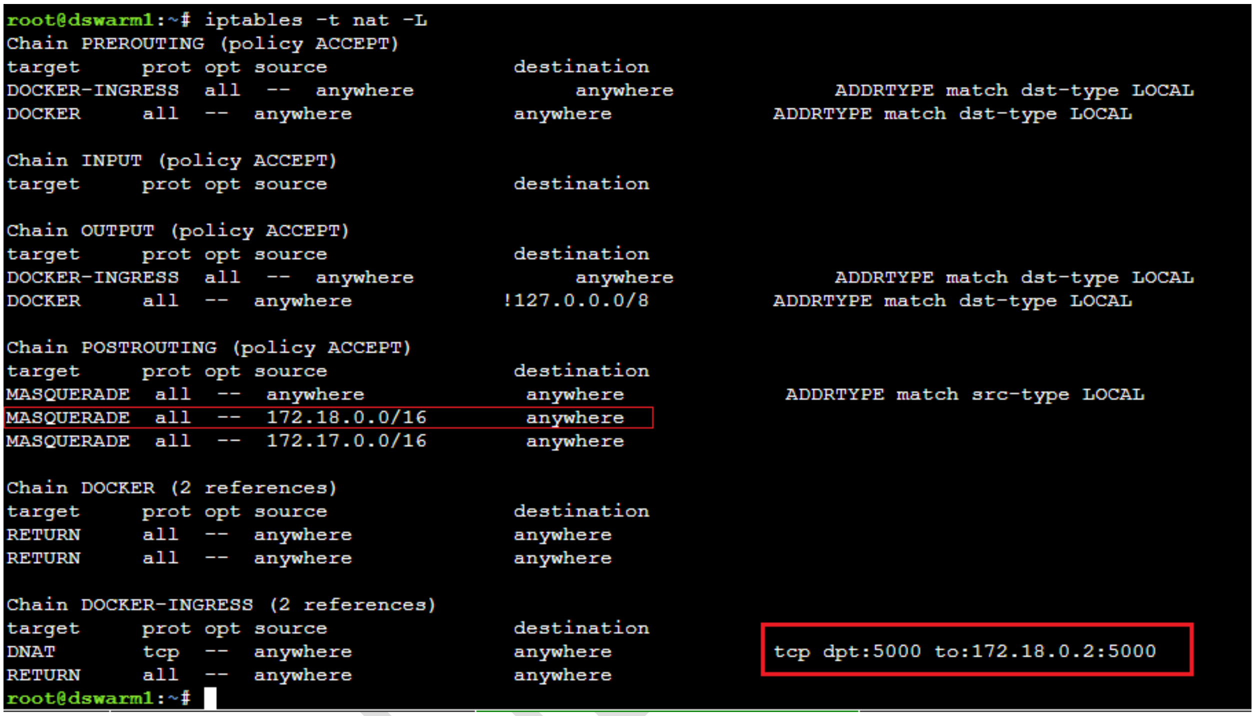
- Verify using VNC, Web browser by pointing at following IP:port

```
http://<IP-dnode1>:5000

http://<IP-dnode2>:5000
```

```
http://<IP-dswarm>:5000
```

- For clients that are OUTSIDE the cluster and accessing the EXPOSED services, Docker swarm uses Routing Mesh to ensure services are reachable from any node. Verify IP Tables rules on all nodes

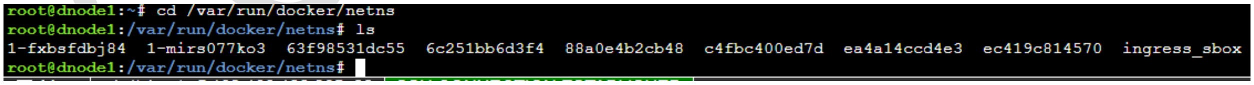
```
    iptables -t nat -L
```



```
root@dswarm1:~# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DOCKER-INGRESS  all  --  anywhere             anywhere             ADDRTYPE match dst-type LOCAL
DOCKER     all  --  anywhere             anywhere             ADDRTYPE match dst-type LOCAL

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DOCKER-INGRESS  all  --  anywhere             anywhere             ADDRTYPE match dst-type LOCAL
DOCKER     all  --  anywhere            !127.0.0.0/8          ADDRTYPE match dst-type LOCAL

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  anywhere             anywhere             ADDRTYPE match src-type LOCAL
MASQUERADE  all  --  172.18.0.0/16        anywhere
MASQUERADE  all  --  172.17.0.0/16        anywhere

Chain DOCKER (2 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain DOCKER-INGRESS (2 references)
target     prot opt source               destination
DNAT       tcp  --  anywhere             anywhere             tcp dpt:5000 to:172.18.0.2:5000
RETURN     all  --  anywhere             anywhere
root@dswarm1:~#
```

- Docker swarm creates hidden containers that does IPVS Linux Kernel based load-balancing.
- On any nodes where service is present, check the network namespaces created.

```
cd /var/run/docker/netns

ls
```

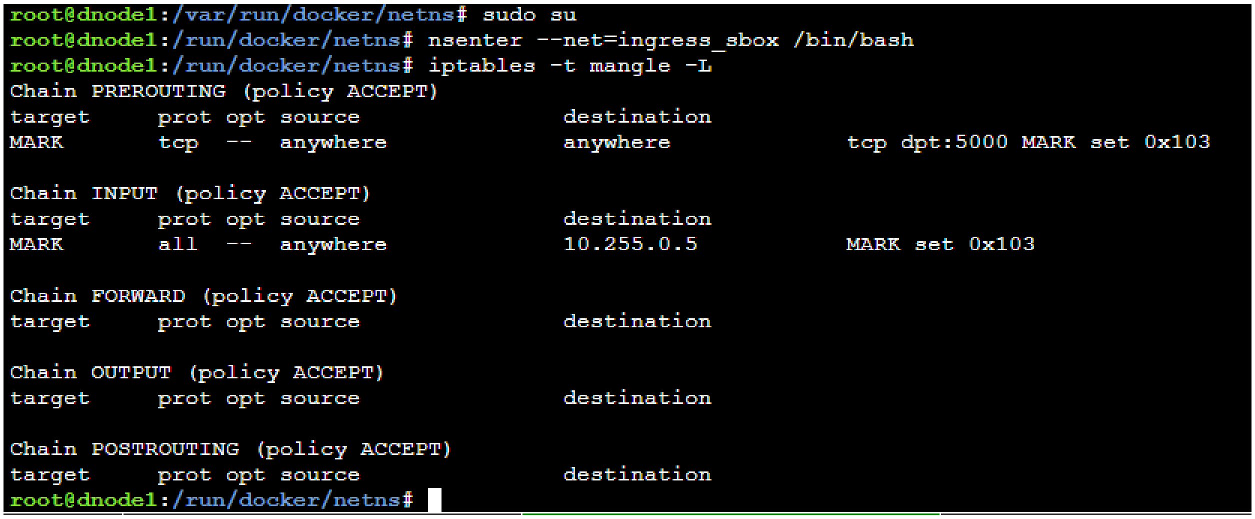


```
root@dnode1:~# cd /var/run/docker/netns
root@dnode1:/var/run/docker/netns# ls
1-fxbsfdbj84  1-mirs077ko3  63f98531dc55  6c251bb6d3f4  88a0e4b2cb48  c4fbc400ed7d  ea4a14ccd4e3  ec419c814570  ingress_sbox
root@dnode1:/var/run/docker/netns#
```

(You will see a network namespace with name ingress_sbox)

- Enter the superuser mode and then execute the command

```
sudo su

nsenter --net=ingress_sbox /bin/bash

iptables -t mangle -L
```

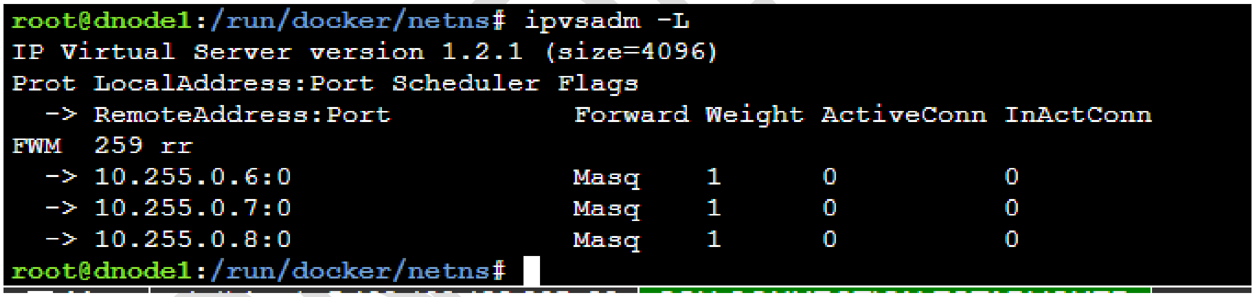


```
root@dnode1:/var/run/docker/netns# sudo su
root@dnode1:/run/docker/netns# nsenter --net=ingress_sbox /bin/bash
root@dnode1:/run/docker/netns# iptables -t mangle -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
MARK       tcp  --  anywhere             anywhere             tcp dpt:5000 MARK set 0x103

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
MARK       all  --  anywhere             10.255.0.5           MARK set 0x103

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
root@dnode1:/run/docker/netns# 
```

(Verify the MARK for published service port)

- Verify the IPVS Load Balancing rules in Round Robin Fashion

```
apt-get install ipvsadm

ipvsadm -L
```



```
root@dnode1:/run/docker/netns# ipvsadm -L
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
FWM  259 rr
  -> 10.255.0.6:0                 Masq    1      0          0
  -> 10.255.0.7:0                 Masq    1      0          0
  -> 10.255.0.8:0                 Masq    1      0          0
root@dnode1:/run/docker/netns# 
```

- For clients that are INSIDE the cluster and accessing other INSIDE services, docker swarm provisions "IPTables" rules and IPVS Load-balancing rules on every container Namespace.
- In order to know the container's network namespace, grep the "SandboxID" from the docker inspect command. Observe the first 12 characters of the SandboxID. This will be the Network namespace ID.

```
docker ps

docker inspect CONTAINERID | grep -i SandboxID
```

- Enter the superuser mode and then execute the command

```
sudo su

nsenter --net=CONTAINER-NETNS-ID /bin/bash
```

```
iptables -t mangle -L
```

(Verify the MARK for published service port)

- Verify the IPVS Load Balancing rules in Round Robin Fashion

```
ipvsadm -L
```

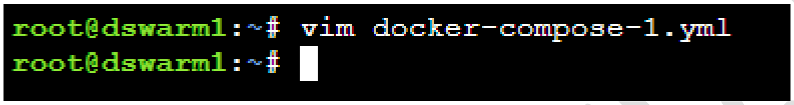
Exercise-33

Docker Stacks with Visualizer

On dswarm1 node

- Create new file that includes a visualizer along with the contents in docker-compose.yml

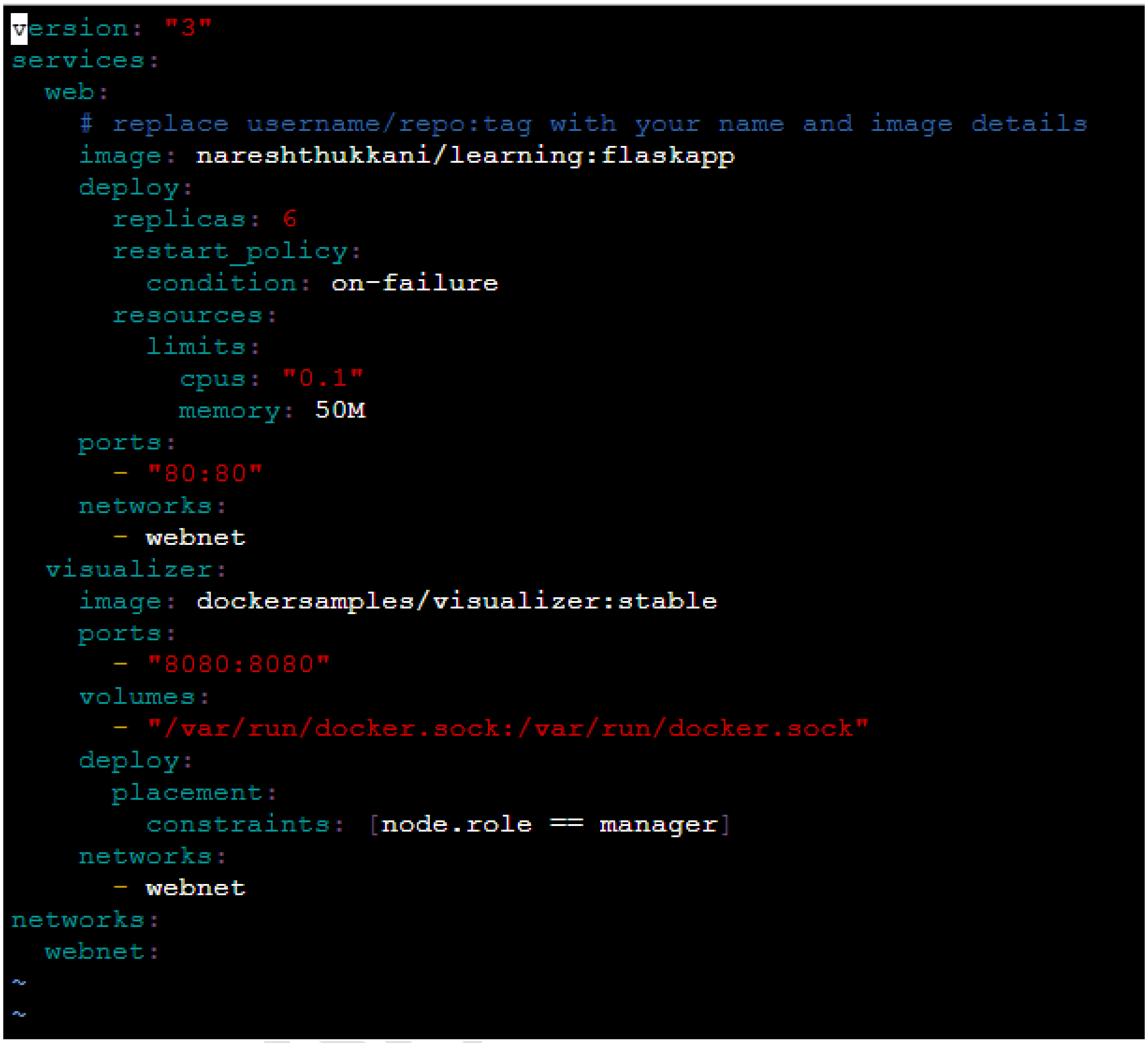
```
vim docker-compose-1.yml
```



```
root@dswarm1:~# vim docker-compose-1.yml
root@dswarm1:~#
```

- Insert the below lines

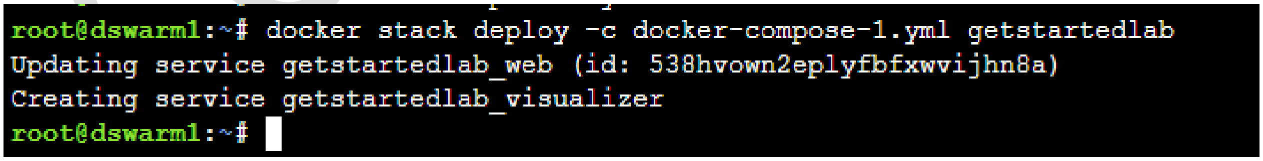
```
version: "3"

services:

  web:

    # replace username/repo:tag with your name and image details

    image: nareshthukkani/learning:flaskapp

    deploy:

      replicas: 6

      restart_policy:

        condition: on-failure

      resources:

        limits:

          cpus: "0.1"

          memory: 50M

    ports:

      - "80:80"

    networks:
      - webnet
  visualizer:
    image: dockersamples/visualizer:stable
    ports:
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]
    networks:
      - webnet
networks:
  webnet:
```



```
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: nareshthukkani/learning:flaskapp
    deploy:
      replicas: 6
      restart_policy:
        condition: on-failure
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
    ports:
      - "80:80"
    networks:
      - webnet
  visualizer:
    image: dockersamples/visualizer:stable
    ports:
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]
    networks:
      - webnet
networks:
  webnet:
~
~
```

- Deploy the stack

```
docker stack deploy -c docker-compose-1.yml getstartedlab
```



```
root@dswarm1:~# docker stack deploy -c docker-compose-1.yml getstartedlab
Updating service getstartedlab_web (id: 538hvown2eplyfbfxwvijhn8a)
Creating service getstartedlab_visualizer
root@dswarm1:~#
```

- Verify visualizer using VNC, Web browser at port [IP-dswarm:5000]
- Visualize the docker containers across multiple nodes

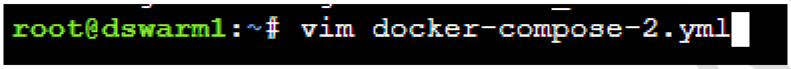
Exercise-34

Docker stacks with Persistent Data

On dswarm1 node

- Create a new file that includes Redis database service along with contents of docker-compose-1.yml

```
vim docker-compose-2.yml
```

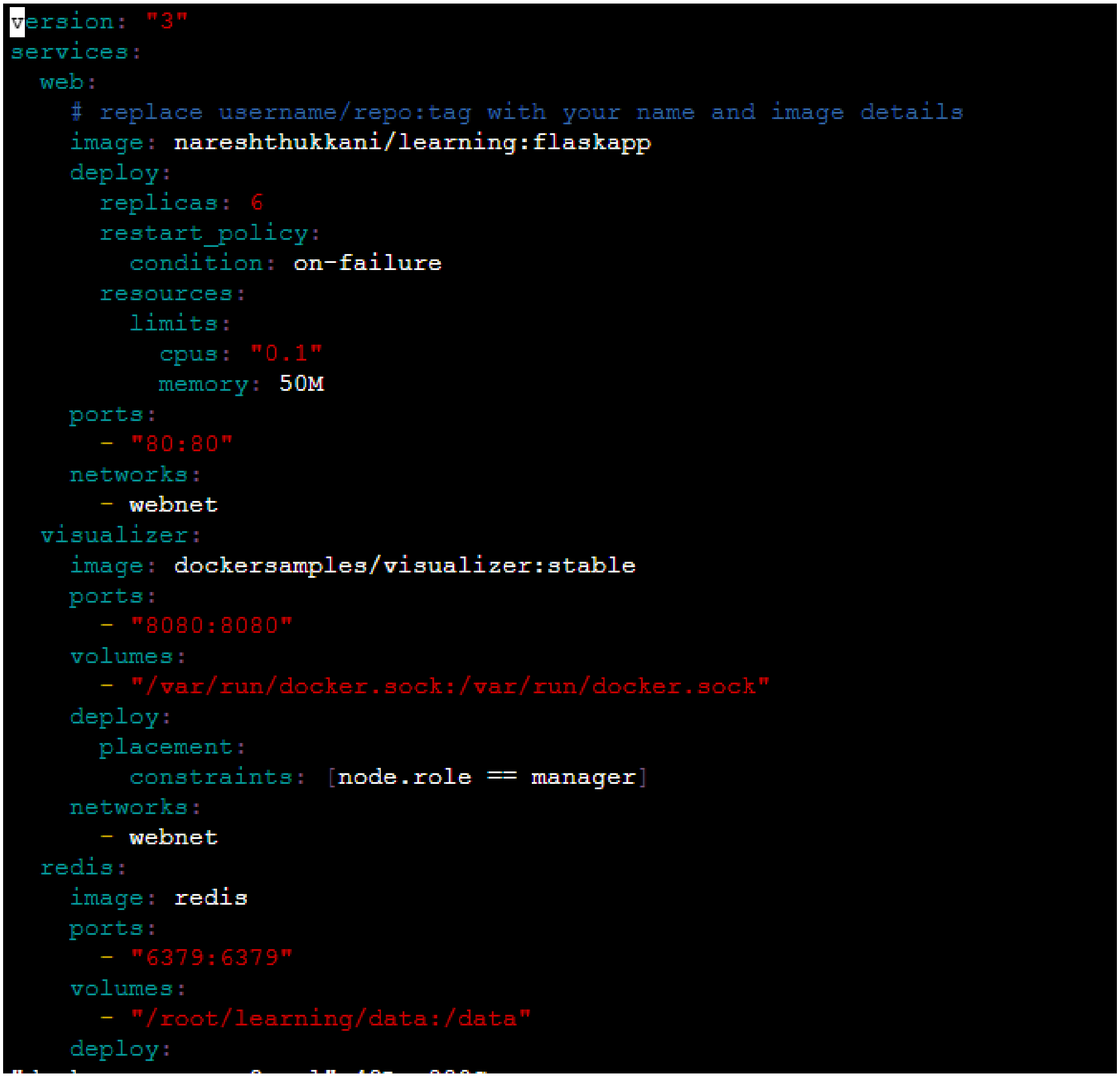


```
root@dswarm1:~# vim docker-compose-2.yml
```

- Insert the below lines

```
version: "3"

services:

  web:

    # replace username/repo:tag with your name and image details

    image: nareshthukkani/learning:flaskapp

    deploy:

      replicas: 6

      restart_policy:

        condition: on-failure

      resources:

        limits:

          cpus: "0.1"

          memory: 50M

    ports:

      - "80:80"

    networks:
      - webnet
  visualizer:
    image: dockersamples/visualizer:stable
    ports:
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]
    networks:
      - webnet
  redis:
    image: redis
    ports:
      - "6379:6379"
    volumes:
      - "/root/learning/data:/data"
    deploy:
      placement:
        constraints: [node.role == manager]
    command: redis-server --appendonly yes
    networks:
      - webnet
```

```
networks:

  webnet:
```



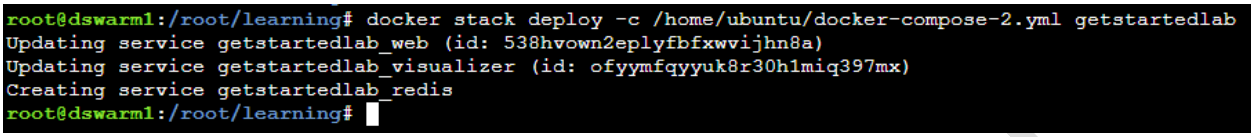
```
version: "3"
services:
  web:
    # replace username/repo:tag with your name and image details
    image: nareshthukkani/learning:flaskapp
    deploy:
      replicas: 6
      restart_policy:
        condition: on-failure
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
    ports:
      - "80:80"
    networks:
      - webnet
  visualizer:
    image: dockersamples/visualizer:stable
    ports:
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    deploy:
      placement:
        constraints: [node.role == manager]
    networks:
      - webnet
  redis:
    image: redis
    ports:
      - "6379:6379"
    volumes:
      - "/root/learning/data:/data"
    deploy:
```

- Create a directory for persistent data under /root/learning path

```
mkdir /root/learning

cd /root/learning

mkdir data
```

- Deploy the stack

```
docker stack deploy -c /home/ubuntu/docker-compose-2.yml getstartedlab
```



```
root@dswarm1:/root/learning# docker stack deploy -c /home/ubuntu/docker-compose-2.yml getstartedlab
Updating service getstartedlab_web (id: 538hvown2eplyfbfxwvijhn8a)
Updating service getstartedlab_visualizer (id: ofyymfqyyuk8r30h1miq397mx)
Creating service getstartedlab_redis
root@dswarm1:/root/learning# 
```

- Verify visualizer using VNC, Web browser at port [IP-dswarm:8080]
- Visualize the docker containers across multiple nodes
- Visualize Redis service is present on dswarm Manager only
- Even though the Redis container terminates, data is persistent in /root/learning/data folder

Miscellaneous

Track Labels

Track labels

Guidance: You will need to create files based on inputs provided here

- The primary, stable release would have a track label with value as stable:

```
name: frontend
        replicas: 3
        ...
        labels:
                app: guestbook
                tier: frontend
                track: stable
        ...
        image: gb-frontend:v3
```

and then you can create a new release of the guestbook frontend that carries the track label with different value (i.e. canary), so that two sets of pods would not overlap:

```
        name: frontend-canary
        replicas: 1
        ...
        labels:
                app: guestbook
                tier: frontend
                track: canary
        ...
        image: gb-frontend:v4
```

The frontend service would span both sets of replicas by selecting the common subset of their labels (i.e. omitting the track label), so that the traffic will be redirected to both applications:

```
        selector:
             app: guestbook
             tier: frontend
```

You can tweak the number of replicas of the stable and canary releases to determine the ratio of each release that will receive live production traffic (in this case, 3:1). Once youre confident, you can update the stable track to the new application release and remove the canary one.

Updating the Labels and annotations

Guidance: Explore below commands and play with deployments

Sometimes existing pods and other resources need to be relabeled before creating new resources. This can be done with kubectl label. For example, if you want to label all your nginx pods as frontend tier, simply run:

```
kubectl label pods -l app=nginx tier=fe
```

Output:

```
pod "my-nginx-2035384211-j5fhi" labeled
pod "my-nginx-2035384211-u2c7e" labeled
pod "my-nginx-2035384211-u3t6x" labeled
```

This first filters all pods with the label app=nginx, and then labels them with the tier=fe. To see the pods you just labeled, run:

```
kubectl get pods -l app=nginx -L tier
```

Output:

```
NAME                        READY     STATUS    RESTARTS   AGE       TIER
my-nginx-2035384211-j5fhi   1/1       Running   0          23m       fe
my-nginx-2035384211-u2c7e   1/1       Running   0          23m       fe
```

```
    my-nginx-2035384211-u3t6x   1/1       Running   0          23m       fe
```

This outputs all app=nginx pods, with an additional label column of pods tier (specified with -L or --label-columns).

Updating annotations

Sometimes you would want to attach annotations to resources. Annotations are arbitrary non-identifying metadata for retrieval by API clients such as tools, libraries, etc. This can be done with kubectl annotate. For example:

```
    kubectl annotate pods my-nginx-v4-9gw19 description='my frontend running nginx'

    kubectl get pods my-nginx-v4-9gw19 -o yaml
```

Output:

```
    apiversion: v1

    kind: pod

    metadata:

        annotations:

            description: my frontend running nginx

    ...
```