Docker+Kubernetes Administration

Activity guide

Release 1.15 rev56

Component Soft Ltd.

January 27, 2020

This instructional program, including all material provided herein, is supplied without any guarantees from Component Soft Ltd. Component Soft Ltd. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

Component Soft Ltd. accepts no liability for any claims, demands, losses, damages, costs or expenses suffered or incurred howsoever arising from or in connection with the use of this courseware. All trademarks are the property of their respective owners.

Contents

Lab 1: Introduction to containers

Task 1: Health check

Perform basic health check on your local docker and kubernetes installation.

- On your **lab_machine**, check whether your nodes are running

```
root@lab_machine $> /labfiles/os_nodes list
Id     Name                              State
----------------------------------------------------
1      master1                           running
2      worker1                           running
3      worker2                           running
4      worker3                           running
5      node1                             running
```

- Autoaccept the ssh-keys of the nodes

```
root@lab_machine $> ssh-keyscan -H node1  >> ~/.ssh/known_hosts
# node1 SSH-2.0-OpenSSH_6.6.1
# node1 SSH-2.0-OpenSSH_6.6.1
root@lab_machine $>
```

Task 2: Understand linux namespaces

In this task we introduce a few commands that can be useful to understand how namespaces are working.

- List the namespaces of a process

  Each process has all namespaces listed in directory /**proc**/<**PID**>/**ns**:

```
root@lab_machine $> ls -l /proc/1/ns
total 0
lrwxrwxrwx 1 root root 0 May 23 10:13 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 root root 0 May 23 10:13 mnt -> mnt:[4026531840]
lrwxrwxrwx 1 root root 0 May 23 10:13 net -> net:[4026531956]
lrwxrwxrwx 1 root root 0 May 23 10:13 pid -> pid:[4026531836]
lrwxrwxrwx 1 root root 0 May 23 10:13 user -> user:[4026531837]
lrwxrwxrwx 1 root root 0 May 23 10:13 uts -> uts:[4026531838]
root@lab_machine $>
```

- Execute a command in a new namespace

```
root@lab_machine $> unshare --net bash
root@lab_machine $> ls -l /proc/$$/ns/net
lrwxrwxrwx 1 root root 0 May 23 13:01 /proc/18189/ns/net -> net:[4026532484]
root@lab_machine $> ip a
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
root@lab_machine $> ip link add ns-test type dummy
root@lab_machine $> ip a
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ns-test: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether ce:df:bb:7a:e8:90 brd ff:ff:ff:ff:ff:ff
root@lab_machine $>
```

- Execute a command in the namespace of an other command

First we need the PID of the command to whose namespace we want to attach (we assume that we are in the shell that we have started using unshare)

```
root@lab_machine $> echo $$
18189
```

Run the following commands in a **new** terminal

```
root@lab_machine $> nsenter -t 18189 --net bash
root@lab_machine $> ip a
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ns-test: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether ce:df:bb:7a:e8:90 brd ff:ff:ff:ff:ff:ff
root@lab_machine $>
```

Exit the new shell and check the network links

```
root@lab_machine $> exit
exit
root@lab_machine $> ip a | head
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: br_management: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
       qlen 1000
    link/ether fe:00:27:00:01:01 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.1/24 brd 10.10.10.255 scope global br_management
       valid_lft forever preferred_lft forever

root@lab_machine $>
```

Close the second terminal, and make sure that the shell in the current terminal can see the system's network interfaces (if needed exit the shell that was started with unshare).

```
root@lab_machine $> ip a | head
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ns-test: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether ba:3e:1b:13:67:14 brd ff:ff:ff:ff:ff:ff
root@lab_machine $> exit
exit
root@lab_machine $> ip a | head
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: br_management: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
       qlen 1000
    link/ether fe:00:27:00:01:01 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.1/24 brd 10.10.10.255 scope global br_management
       valid_lft forever preferred_lft forever

root@lab_machine $>
```

Task 3: Managing docker containers

In this task we are presenting basic docker functionality. All commands run on **node1** node.

- Docker is preinstalled but not started. Start docker engine.

```
root@node1:~# systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/
lib/systemd/system/docker.service.
root@node1:~# systemctl start docker
root@node1:~# systemctl status docker
docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disab
led)
   Active: active (running) since Thu 2018-10-04 11:52:04 UTC; 1min 19s ago
     Docs: https://docs.docker.com
 Main PID: 1131 (dockerd)
    Tasks: 16
   Memory: 138.9M
   CGroup: /system.slice/docker.service
           +-1131 /usr/bin/dockerd
           +-1135 docker-containerd --config /var/run/docker/containerd/containerd.toml

Oct 04 11:52:03 node1 dockerd[1131]: time="2018-10-04T11:52:03.819209682Z" level=info
Oct 04 11:52:03 node1 dockerd[1131]: time="2018-10-04T11:52:03.876573864Z" level=info
Oct 04 11:52:03 node1 dockerd[1131]: time="2018-10-04T11:52:03.912044932Z" level=info
Oct 04 11:52:03 node1 dockerd[1131]: time="2018-10-04T11:52:03.912554509Z" level=info
Oct 04 11:52:04 node1 dockerd[1131]: time="2018-10-04T11:52:04.057821919Z" level=info
Oct 04 11:52:04 node1 dockerd[1131]: time="2018-10-04T11:52:04.088479578Z" level=info
Oct 04 11:52:04 node1 dockerd[1131]: time="2018-10-04T11:52:04.107661757Z" level=info
Oct 04 11:52:04 node1 dockerd[1131]: time="2018-10-04T11:52:04.107784765Z" level=info
Oct 04 11:52:04 node1 dockerd[1131]: time="2018-10-04T11:52:04.122436555Z" level=info
Oct 04 11:52:04 node1 systemd[1]: Started Docker Application Container Engine.
Hint: Some lines were ellipsized, use -l to show in full.
root@node1:~# docker version
Client:
 Version:      18.03.1-ce
 API version:  1.37
 Go version:   go1.9.5
 Git commit:   9ee9f40
 Built:        Thu Apr 26 07:20:16 2018
 OS/Arch:      linux/amd64
 Experimental: false
 Orchestrator: swarm

Server:
 Engine:
  Version:      18.03.1-ce
  API version:  1.37 (minimum version 1.12)
  Go version:   go1.9.5
  Git commit:   9ee9f40
  Built:        Thu Apr 26 07:23:58 2018
  OS/Arch:      linux/amd64
  Experimental: false
root@node1:~#
```

- Run nginx webserver in a container (use -d to run on background)

```
root@node1:~# docker container run -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
802b00ed6f79: Pull complete
5291925314b3: Pull complete
bd9f53b2c2de: Pull complete
Digest: sha256:9ad0746d8f2ea6df3a17ba89eca40b48c47066dfab55a75e08e2b70fc80d929e
Status: Downloaded newer image for nginx:latest
7e6c31223e908621c40bb90a3c759fdf424228fb84928a4b5a62f6bffee93abb
root@node1:~#
```

Notice the image was pulled from docker hub as it was the first use of that image. Container has no name, we can refer it by its ID (see the last line).

- Check running containers

```
root@node1:~# docker container ps
CONTAINER ID        IMAGE        COMMAND                   CREATED            STATUS
7e6c31223e90        nginx        "nginx -g 'daemon of..." 1 minutes ago      Up 1 minute
root@node1:~# ps -ef | grep [n]ginx
root      1471  1459  0 11:58 ?        00:00:00 nginx: master process nginx -g daemon of
101       1492  1471  0 11:58 ?        00:00:00 nginx: worker process
root@node1:~#
```

- Run interactivelly busybox image in a container with name bbox

```
root@node1:~# docker container run -ti --name bbox busybox
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
90e01955edcd: Pull complete
Digest: sha256:2a03a6059f21e150ae84b0973863609494aad70f0a80eaeb64bddd8d92465812
Status: Downloaded newer image for busybox:latest
/ #
```

- Keep bbox running and check all containers from **new** terminal window

```
root@node1:~# docker container ps
CONTAINER ID        IMAGE          COMMAND                  CREATED           STATUS
165aaae10370        busybox        "sh"                     2 minutes ago     Up 2 minutes
7e6c31223e90        nginx          "nginx -g 'daemon of..." 2 minutes ago     Up 2 minutes
root@node1:~#
```

- Check isolation of containers - you cannot see nginx from bbox
    - Display all processes in bbox and exit from its shell
    - Check that processes outside the container are not listed

```
/ # hostname
165aaae10370
/ # ps -ef
PID   USER     TIME  COMMAND
    1 root      0:00 sh
    5 root      0:00 ps -ef
/ # exit
root@node1:~#
```

- Busybox container is no longer nunning (exited), but still exists (filesystem is preserved and mounted). Check list of all containers.

```
root@node1:~# docker container ps
CONTAINER ID        IMAGE          COMMAND                  CREATED           STATUS
7e6c31223e90        nginx          "nginx -g 'daemon of..." 5 minutes ago     Up 5 minutes
root@node1:~# docker container ps -a
CONTAINER ID        IMAGE          COMMAND                  CREATED           STATUS
165aaae10370        busybox        "sh"                     6 minutes ago     Exited (0) 2
7e6c31223e90        nginx          "nginx -g 'daemon of..." 5 minutes ago     Up 5 minutes
root@node1:~#
```

- Execute `ls /etc/nginx` from inside of nginx container

```
root@node1:~# docker exec -ti 7e6c31223e90 ls /etc/nginx
conf.d                   koi-utf  mime.types  nginx.conf   uwsgi_params
fastcgi_params           koi-win  modules     scgi_params  win-utf
root@node1:~#
```

- Remove both containers

```
root@node1:~# docker container stop 7e6c31223e90
7e6c31223e90
root@node1:~# docker container rm 7e6c31223e90 bbox
7e6c31223e90
bbox
root@node1:~#
```

Task 4: Managing images

- List localy available images

```
root@node1:~# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              be1f31be9a87        10 minutes ago      109MB
busybox             latest              59788edf1f3e        10 minutes ago      1.15MB
root@node1:~#
```

- We want to run **curl** web client inside **ubuntu** container, but it is not installed there. Install it and commit the container as new ubuntu_curl image.

```
root@node1:~# docker container run -ti --name ubu ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
124c757242f8: Pull complete
9d866f8bde2a: Pull complete
fa3f2f277e67: Pull complete
398d32b153e8: Pull complete
afde35469481: Pull complete
Digest: sha256:de774a3145f7ca4f0bd144c7d4ffb2931e06634f11529653b23eba85aef8e378
Status: Downloaded newer image for ubuntu:latest
root@e00354a60608:/# curl
bash: curl: command not found
root@e00354a60608:/# apt-get update && apt-get install -y curl
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
<output omitted>

root@e00354a60608:/# curl
curl: try 'curl --help' or 'curl --manual' for more information
root@e00354a60608:/# exit
root@node1:~# docker container commit ubu ubuntu_curl
sha256:7e80268206e78bb7c237bc20aeb8f0caf5456be470a301195db297b0f6326b25
root@node1:~# docker image list
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu_curl         latest              7e80268206e7        24 seconds ago      140MB
nginx               latest              be1f31be9a87        2 hours ago         109MB
busybox             latest              59788edf1f3e        2 hours ago         1.15MB
ubuntu              latest              cd6d8154f1e1        4 weeks ago         84.1MB
root@node1:~#
```

- Check that containers created from new image already contain curl

```
root@node1:~# docker container run -ti --rm ubuntu_curl
root@af730503d465:/# curl
curl: try 'curl --help' or 'curl --manual' for more information
root@af730503d465:/# exit
exit
root@node1:~#
```

- Use **docker build** to create your own IRC chat server image
  - based on **ubuntu**
  - **ircd-irc2** package should be installed
  - default container command: **/usr/sbin/ircd -t**
  - image should expose **port 6667**

```
root@node1:~# mkdir build
root@node1:~# cd build
root@node1:~/build# vi Dockerfile
root@node1:~/build# cat Dockerfile
FROM ubuntu
RUN apt-get update && apt-get install -y ircd-irc2
CMD /usr/sbin/ircd -t
EXPOSE 6667
root@node1:~/build# docker build -t chatserver .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM ubuntu
 ---> cd6d8154f1e1
<output omitted>


Step 4/4 : EXPOSE 6667
 ---> Running in d50be06661be
Removing intermediate container d50be06661be
 ---> db3c62ce3030
Successfully built db3c62ce3030
Successfully tagged chatserver:latest
root@node1:~/build# cd
root@node1:~#
```

- Test chat server - run a container with chat server - install chat client **irssi** on node1 - try to connect

```
root@node1:~# docker container run -d --rm --name mychat chatserver
87e05cddcbb713b007eee803d078285154a03ec319322340596ab54f268fc658
root@node1:~# docker container exec 87e05 grep 87e05 /etc/hosts
172.17.0.3    87e05cddcbb7
root@node1:~# yum install -y irssi
Loaded plugins: fastestmirror, priorities, remove-with-leaves
Loading mirror speeds from cached hostfile
<output omitted>

Installed:
  irssi.x86_64 0:0.8.15-16.el7

Complete!
```

```
root@node1:~# irssi -c 172.17.0.3
<output omitted>

13:29 -!- -                          [ Debian GNU/Linux ]
13:29 -!- - |------------------------------------------------------------------------|
13:29 -!- - | This is Debian's default IRCd server configuration for irc2.11. If you |
13:29 -!- - | see this and if you are the server administrator, just edit ircd.conf  |
13:29 -!- - | and ircd.motd in /etc/ircd.                                            |
13:29 -!- - |                                     Martin Loschwitz, 1st January 2005 |
13:29 -!- - |------------------------------------------------------------------------|
13:29 -!- End of MOTD command.
13:29 !irc.localhost Server is currently in split-mode.
13:29 -!- Mode change [+i] for user root
 [13:29] [root(+i)] [1:172 (change with ^X)]
[(status)] /help
<output omitted>

[(status)] /quit
root@node1:~#
```

Task 5: Managing volumes

- Create a directory on `host` system (node1) and make it accessible from busybox container
  - create ./data directory
  - run a new busybox container (–rm) and use ./data as external volume
  - create a new file on volume from inside of the container
  - exit from container
  - check the file from host system

```
root@node1:~# mkdir ./data
root@node1:~# docker container run -ti --rm -v ~/data:/DATA busybox
/ # ls /DATA
/ # cp /etc/hosts /DATA
/ # ls /DATA
hosts
/ # exit
root@node1:~# ls data
hosts
root@node1:~#
```

- Create a volume and share it between containers
  - create a named volume **myvol1**
  - in the first window run busybox mounting myvol1 under /b_data

```
root@node1:~# docker volume create myvol1
myvol1
root@node1:~# docker volume ls
DRIVER              VOLUME NAME
local               myvol1
root@node1:~# docker container run -ti --rm -v myvol1:/b_data busybox
/ # df -h / /b_data
Filesystem                Size      Used Available Use% Mounted on
/dev/mapper/docker-252:1-9332531-c9aaa7b998d531cb56657682ed60c6aa8ab27b758c02707acccd39
                         10.0G     33.9M     10.0G   0% /
/dev/vda1                 8.0G      2.7G      5.3G  34% /b_data
/ #
```

- in the second window run ubuntu mounting myvol1 under /u_data
- in ubuntu container create new file /u_data/text

```
root@node1:~# docker container run -ti --rm -v myvol1:/u_data ubuntu
root@39f858cb7974:/# df -h / /u_data
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/docker-252:1-9332531-51e31b2985880844c0235c84f17bb475070dd89d5d67ee9684fd146
                           10G  107M  9.9G   2% /
/dev/vda1                 8.0G  2.7G  5.3G  34% /u_data
root@39f858cb7974:/# ls /u_data
root@39f858cb7974:/# echo 'Hello docker!' > /u_data/text
root@39f858cb7974:/# exit
```

- in busybox container display the file

```
/ # ls /b_data
text
/ # cat /b_data/text
Hello docker!
/ # exit
root@node1:~#
```

Task 6: Docker networking labs

- Explore container networking
  - run a new busybox container (–rm, -ti)
  - check IP address and network interfaces from inside of the container

```
root@node1:~# docker container run --rm -ti busybox
/ # route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.17.0.1      0.0.0.0         UG    0      0        0 eth0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 eth0
/ # ip a s
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
36: eth0@if37: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
    link/ether 02:42:ac:11:00:05 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.5/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
/ #
```

Container uses **eth0** for outgoing trafic and route it to **172.17.0.1**. Note, that eth0 is one end of an ethernet pair. The ethernet pair (virtual patch cable) interfaces have indexes 36 and 37.

- check the other end of `veth` pair from host side

  Keep container running and continue on host in new terminal window

```
root@node1:~# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group d
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAU
    link/ether 08:00:27:00:01:05 brd ff:ff:ff:ff:ff:ff
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAU
    link/ether 02:42:fe:29:c3:e3 brd ff:ff:ff:ff:ff:ff
11: veth04d7be1@if10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master do
    link/ether 7a:29:c8:3c:4c:80 brd ff:ff:ff:ff:ff:ff link-netnsid 0
35: veth943b579@if34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master do
    link/ether ce:2d:2c:5c:22:9f brd ff:ff:ff:ff:ff:ff link-netnsid 2
37: veth6bb2aba@if36: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master do
    link/ether d6:0c:9e:20:6d:91 brd ff:ff:ff:ff:ff:ff link-netnsid 3
root@node1:~# brctl show | grep -C10 veth6bb2aba
bridge name  bridge id               STP enabled     interfaces
docker0             8000.0242fe29c3e3       no              veth04d7be1
                                                        veth6bb2aba
                                                        veth943b579
root@node1:~#
```

The local end of veth pair 36-37 is interface **veth6bb2aba**, connected to linux bridge **docker0**.

- check docker0 IP address and routing table
- check iptables NAT rule

```
root@node1:~# ip a s docker0
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:fe:29:c3:e3 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:feff:fe29:c3e3/64 scope link
       valid_lft forever preferred_lft forever
root@node1:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.10.10.1      0.0.0.0         UG    0      0        0 eth0
10.10.10.0      0.0.0.0         255.255.255.0   U     0      0        0 eth0
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
root@node1:~# iptables -t nat -vnL POSTROUTING
Chain POSTROUTING (policy ACCEPT 412 packets, 26913 bytes)
```

```
 pkts bytes target     prot opt in     out     source               destination
   42  2599 MASQUERADE  all  --  *      !docker0  172.17.0.0/16        0.0.0.0/0
root@node1:~#
```

Outgoing packets arrive from veth6bb2aba to docker0. Then they are routed via eth0 gateway 10.10.10.1. Outgoing packets are NATed in POSTROUTING chain of `nat` table of iptables.

- run ping to Google from inside of the container and keep it pinging

```
/ # ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=120 time=5.198 ms
64 bytes from 8.8.8.8: seq=1 ttl=120 time=5.553 ms
<output omitted>
```

- check the packet on host system before and after routing

  run tcpdump in node1 terminal window:

```
root@node1:~# tcpdump -ni docker0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on docker0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:17:21.347406 IP 172.17.0.5 > 8.8.8.8: ICMP echo request, id 1792, seq 143, length 64
18:17:21.352769 IP 8.8.8.8 > 172.17.0.5: ICMP echo reply, id 1792, seq 143, length 64
^C
root@node1:~# tcpdump -ni eth0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:17:56.357444 IP 10.10.10.55 > 8.8.8.8: ICMP echo request, id 1792, seq 178, length 64
18:17:56.362540 IP 8.8.8.8 > 10.10.10.55: ICMP echo reply, id 1792, seq 178, length 64
^C
root@node1:~#
```

Notice, the packets are originated from container IP as seen on internal docker0, but they are already NATed to host IP (node1 10.10.10.55) when they are transmitted to real network via eth0.

- Stop ping (^C).

- run nginx container and make it accessible

  – run nginx and publish exposed ports

```
root@node1:~# docker container run -d --name myweb -P nginx
465d34918e275630d54f2e05c372cfc83e8541b0f0c811a13886d36d881e8b54
root@node1:~# docker port myweb
80/tcp -> 0.0.0.0:32768
root@node1:~#
```

- check access from lab_machine

  Open a new lab_machine terminal and try to access the container:

```
root@lab_machine $> curl node1:32768
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<output omitted>

</body>
</html>
root@lab_machine $>
```

Task 7: Docker logging

- Check docker logging
  - display nginx log

```
root@node1:~# docker logs myweb
10.10.10.1 - - [04/Oct/2018:18:23:14 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.29.0"
root@node1:~#
```

Task 8: Cleanup

- Remove all containers (-f forces to stop them if still running)

```
root@node1:~# docker container rm -f $(docker container ps -aq)
465d34918e27
e17a71455ad2
28e454118a25
87e05cddcbb7
root@node1:~# docker container ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
root@node1:~#
```

- Remove all localy stored images

```
root@master1 $> docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
chatserver          latest              0c2a72f28407        6 hours ago         128MB
<none>              <none>              db3c62ce3030        6 hours ago         128MB
ubuntu_curl         latest              7e80268206e7        6 hours ago         140MB
nginx               latest              be1f31be9a87        2 days ago          109MB
busybox             latest              59788edf1f3e        2 days ago          1.15MB
ubuntu              latest              cd6d8154f1e1        4 weeks ago         84.1MB
root@node1:~# docker image rm $(docker image ls -q)
Untagged: chatserver:latest
Deleted: sha256:0c2a72f28407f030f62c6a99938f7d8bd625124e6a470d6bda71276fd1c0c3f9
Deleted: sha256:a3e8bcb3311d9cbb7c86ee8fbbf717bc9d57a2d8e6b827657cfd9ead7e8dd2a6
Deleted: sha256:bca05e3b98f67ca3042ce9472b6008dace3639f63b47026bebecff2a701f3db1
Deleted: sha256:d15aa8add0ee2f59980f5e7192bff6e4cdf6468753c2788b25b38f4bdeb7d3c0
```

```
Deleted: sha256:db3c62ce30309508bc07b9c73b81868167af4cdbe05644349c0f8f3bac89e88a
<output omitted>

root@master1 $> docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
root@node1:~#
```

- Remove all docker volumes

```
root@node1:~# docker volume rm $(docker volume ls -q)
myvol1
root@node1:~# docker volume ls
DRIVER              VOLUME NAME
root@node1:~# rm -rf ~/data
root@node1:~#
```

Lab 2: Health check the Kubernetes environment

Note:

If you use remote lab environment, and previously you made docker exercises, please run change_course.sh script on your lab machine.

```
root@lab_machine$> /root/change_course.sh

To which course do you want to change?

... omitted lines ...

root@lab_machine$>
```

Task 1: Health check

Perform basic health check on your local kubernetes installation.

- On your **lab host**, check whether your nodes are running

```
root@lab_machine $> /labfiles/os_nodes list
Id    Name                               State
----------------------------------------------------
28    master1                            running
29    worker1                            running
30    worker2                            running
31    worker3                            running
```

- Autoaccept the ssh-keys of the nodes

```
root@lab_machine $> for i in master1 worker{1,2,3}; do
> ssh-keyscan -H $i  >> ~/.ssh/known_hosts
> done
# master1 SSH-2.0-OpenSSH_6.6.1
# master1 SSH-2.0-OpenSSH_6.6.1
# worker1 SSH-2.0-OpenSSH_6.6.1
# worker1 SSH-2.0-OpenSSH_6.6.1
# worker2 SSH-2.0-OpenSSH_6.6.1
# worker2 SSH-2.0-OpenSSH_6.6.1
# worker3 SSH-2.0-OpenSSH_6.6.1
# worker3 SSH-2.0-OpenSSH_6.6.1
root@lab_machine $>
```

- Check whether VMs have the same system date and time (1-2 seconds of difference is OK).

```
root@lab_machine $> for i in master1 worker{1,2,3}
> do
> echo -en "$i :"; ssh $i 'date'
> done
master1 :Warning: Permanently added the ECDSA host key for IP address
'10.10.10.51' to the list of known hosts.
Tue May 23 07:19:09 UTC 2017
worker1 :Warning: Permanently added the ECDSA host key for IP address
'10.10.10.52' to the list of known hosts.
Tue May 23 07:19:09 UTC 2017
worker2 :Warning: Permanently added the ECDSA host key for IP address
'10.10.10.53' to the list of known hosts.
Tue May 23 07:19:09 UTC 2017
worker3 :Warning: Permanently added the ECDSA host key for IP address
'10.10.10.54' to the list of known hosts.
Tue May 23 07:19:09 UTC 2017
```

Lab 3: Accessing the kubernetes API

Task 1: Browse the kubernetes API

- Check the help page of the kubectl command

```
root@master1 $> kubectl --help
kubectl controls the Kubernetes cluster manager.

Find more information at https://github.com/kubernetes/kubernetes.

Basic Commands (Beginner):
  create         Create a resource by filename or stdin
  expose         Take a replication controller, service, deployment or pod and expose
                 it as a new Kubernetes Service
  run            Run a particular image on the cluster
  set            Set specific features on objects

Basic Commands (Intermediate):
  explain        Documentation of resources
  get            Display one or many resources
 ...
  proxy          Run a proxy to the Kubernetes API server
 ...
 Other Commands:
   api-resources  Print the supported API resources on the server
   api-versions   Print the supported API versions on the server, in the form of
   "group/version"
   config         Modify kubeconfig files
   plugin         Provides utilities for interacting with plugins.
   version        Print the client and server version information

 Usage:
   kubectl [flags] [options]

 Use "kubectl <command> --help" for more information about a given command.
 Use "kubectl options" for a list of global command-line options (applies to all commands).

root@master1 $>
```

- Start the kubectl proxy command to have a proxy to the API

```
root@master1 $> kubectl proxy &
[1] 17409
Starting to serve on 127.0.0.1:8001
root@master1 $>
```

- Use curl to access the API

```
root@master1 $> curl localhost:8001/api
{
  "kind": "APIVersions",
  "versions": [
    "v1"
  ],
  "serverAddressByClientCIDRs": [
    {
      "clientCIDR": "0.0.0.0/0",
      "serverAddress": "10.10.10.51:6443"
    }
  ]
}
root@master1 $> curl localhost:8001/api/v1
{
  "kind": "APIResourceList",
  "groupVersion": "v1",
  "resources": [
    {
      "name": "bindings",
      "singularName": "",
      "namespaced": true,
      "kind": "Binding",
      "verbs": [
        "create"
      ]
    },
    {
      "name": "componentstatuses",
      "singularName": "",
      "namespaced": false,
      "kind": "ComponentStatus",
      "verbs": [
        "get",
        "list"
      ],
      "shortNames": [
        "cs"
      ]
    },
    {
      "name": "configmaps",
      "singularName": "",
      "namespaced": true,
      "kind": "ConfigMap",
      "verbs": [
        "create",
        "delete",
        "deletecollection",
        "get",
        "list",
        "patch",
        "update",
        "watch"
      ],
      "shortNames": [
        "cm"
      ],
      "storageVersionHash": "qFsyl6wFWjQ="
    },
...
    {
      "name": "services",
      "singularName": "",
      "namespaced": true,
      "kind": "Service",
      "verbs": [
        "create",
        "delete",
        "get",
        "list",
        "patch",
        "update",
        "watch"
      ],
      "shortNames": [
        "svc"
      ],
      "categories": [
        "all"
      ],
      "storageVersionHash": "0/CO1lhkEBI="
    },
    {
      "name": "services/proxy",
      "singularName": "",
      "namespaced": true,
      "kind": "ServiceProxyOptions",
      "verbs": [
        "create",
        "delete",
        "get",
        "patch",
        "update"
      ]
    },
    {
      "name": "services/status",
      "singularName": "",
      "namespaced": true,
      "kind": "Service",
      "verbs": [
        "get",
        "patch",
        "update"
      ]
    }
```

```
  ]
}
root@master1 $>
```

- Use the kubectl api-versions command to verify the supported API groups

```
root@master1 $> kubectl api-versions
admissionregistration.k8s.io/v1beta1
apiextensions.k8s.io/v1beta1
apiregistration.k8s.io/v1
apiregistration.k8s.io/v1beta1
apps/v1
apps/v1beta1
apps/v1beta2
authentication.k8s.io/v1
authentication.k8s.io/v1beta1
authorization.k8s.io/v1
authorization.k8s.io/v1beta1
autoscaling/v1
autoscaling/v2beta1
autoscaling/v2beta2
batch/v1
batch/v1beta1
certificates.k8s.io/v1beta1
coordination.k8s.io/v1
coordination.k8s.io/v1beta1
events.k8s.io/v1beta1
extensions/v1beta1
networking.k8s.io/v1
networking.k8s.io/v1beta1
node.k8s.io/v1beta1
policy/v1beta1
rbac.authorization.k8s.io/v1
rbac.authorization.k8s.io/v1beta1
scheduling.k8s.io/v1
scheduling.k8s.io/v1beta1
storage.k8s.io/v1
storage.k8s.io/v1beta1
v1
root@master1 $>
```

- Use curl to see the content of the rbac.authorization.k8s.io/v1beta1 API group

```
root@master1 $>curl localhost:8001/apis/rbac.authorization.k8s.io/v1beta1
{
  "kind": "APIResourceList",
  "apiVersion": "v1",
  "groupVersion": "rbac.authorization.k8s.io/v1beta1",
  "resources": [
    {
      "name": "clusterrolebindings",
      "namespaced": false,
      "kind": "ClusterRoleBinding",
      "verbs": [
        "create",
        "delete",
        "deletecollection",
        "get",
        "list",
        "patch",
        "update",
        "watch"
      ]
    },
    {
      "name": "clusterroles",
      "namespaced": false,
      "kind": "ClusterRole",
      "verbs": [
        "create",
        "delete",
        "deletecollection",
        "get",
        "list",
        "patch",
        "update",
        "watch"
      ]
    },
    {
      "name": "rolebindings",
      "namespaced": true,
      "kind": "RoleBinding",
      "verbs": [
        "create",
        "delete",
        "deletecollection",
        "get",
        "list",
        "patch",
        "update",
        "watch"
      ]
    },
    {
      "name": "roles",
      "namespaced": true,
      "kind": "Role",
      "verbs": [
        "create",
        "delete",
        "deletecollection",
        "get",
        "list",
        "patch",
        "update",
        "watch"
      ]
    }
  ]
}
root@master1 $>
```

Task 2: Use RBAC to controll access to the API

- Review the content of the role.yaml file from the /labfiles/k8s/Accessing_API directory. The file defines a ClusterRole which can access the get, watch and list verbs of the Pods and Services objects of the core API. Observe that there is no namespace metadata field for this object.

```
root@master1 $> cat role.yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods","services"]
  verbs: ["get", "watch", "list"]

root@master1 $>
```

- Review the content of the binding.yaml file from the /labfiles/k8s/Accessing_API directory. The file defines a RoleBinding in the default namespace. This is binding the ClusterRole *pod-reader* to the *user2* user. Observe the namespace value in the metadata field.

```
root@master1 $> cat binding.yaml
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: binding1
  namespace: default
subjects:
- kind: User
  name: user2
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io

root@master1 $>
```

- Review the content of the clbinding.yaml file from the /labfiles/k8s/Accessing_API directory. The file defines a ClusterRoleBinding. This is binding the *pod-reader* ClusterRole to the *user1* user.

```
root@master1 $> cat clbinding.yaml
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: clbinding1
subjects:
- kind: User
  name: user1
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

```
root@master1 $>
```

- Verify that the users *user1* and *user2* cannot access the pods in the cluster.

```
root@master1 $> kubectl --as user1 get pod
Error from server (Forbidden): pods is forbidden: User "user1" cannot list resource "po
ds" in API group "" in the namespace "default"
root@master1 $> kubectl --as user2 get pod
Error from server (Forbidden): pods is forbidden: User "user2" cannot list resource "po
ds" in API group "" in the namespace "default"
root@master1 $>
```

- create the ClusterRole, and the two bindings.

```
root@master1 $> kubectl create -f role.yaml
clusterrole.rbac.authorization.k8s.io/pod-reader created
root@master1 $>
root@master1 $> kubectl create -f binding.yaml
rolebinding.rbac.authorization.k8s.io/binding1 created
root@master1 $>
root@master1 $> kubectl create -f clbinding.yaml
clusterrolebinding.rbac.authorization.k8s.io/clbinding1 created
root@master1 $>
```

- List the pods from the default namespace as *user1* and *user2*.

```
root@master1 $> kubectl --as user2 get pod
No resources found.

root@master1 $> kubectl create -f ubuntu.yaml
pod/ubuntu created

root@master1 $> kubectl --as user2 get pod
NAME      READY     STATUS    RESTARTS   AGE
ubuntu    1/1       Running   0          35s

root@master1 $> kubectl --as user1 get pod
NAME      READY     STATUS    RESTARTS   AGE
ubuntu    1/1       Running   0          1m
root@master1 $>
```

- List the pods from all the namespaces as *user1* and *user2*

```
root@master1 $> kubectl --as user1 get pod --all-namespaces
NAMESPACE     NAME                                      READY   STATUS    RESTARTS   AGE
default       ubuntu                                    1/1       Running   0          17m
kube-system   coredns-78fcdf6894-pplls                  1/1       Running   1          1d
kube-system   coredns-78fcdf6894-r4qpl                  1/1       Running   1          1d
kube-system   etcd-master1                              1/1       Running   2          1d
kube-system   kube-apiserver-master1                    1/1       Running   2          1d
kube-system   kube-controller-manager-master1           1/1       Running   2          1d
kube-system   kube-proxy-5mm7p                          1/1       Running   1          5h
kube-system   kube-proxy-jjr28                          1/1       Running   1          1d
kube-system   kube-proxy-qbhhg                          1/1       Running   1          5h
kube-system   kube-proxy-vttb6                          1/1       Running   1          5h
```

```
kube-system   kube-scheduler-master1                  1/1       Running   2          1d
kube-system   kubernetes-dashboard-6948bdb78-nj2nq    1/1       Running   1          5h
kube-system   weave-net-fjkr9                         2/2       Running   4          5h
kube-system   weave-net-nmcb4                         2/2       Running   3          5h
kube-system   weave-net-nmklx                         2/2       Running   2          5h
kube-system   weave-net-nwhdg                         2/2       Running   3          5h


root@master1 $> kubectl --as user2 get pod --all-namespaces
Error from server (Forbidden): pods is forbidden: User "user2" cannot list resource "po
ds" in API group "" at the cluster scope
root@master1 $>
```

Task 3: Cleanup

- Delete the objects created during this lab.

```
root@master1 $> kubectl delete pod ubuntu
pod "ubuntu" deleted
root@master1 $>
root@master1 $> kubectl delete clusterrolebinding clbinding1
clusterrolebinding.rbac.authorization.k8s.io "clbinding1" deleted
root@master1 $>
root@master1 $> kubectl delete rolebinding binding1
rolebinding.rbac.authorization.k8s.io "binding1" deleted
root@master1 $>
root@master1 $> kubectl delete clusterrole pod-reader
clusterrole.rbac.authorization.k8s.io "pod-reader" deleted
root@master1 $>
```

Lab 4: Kubernetes workloads

Task 1: Pod operations

- Create a manifest file for a pod that uses the *busybox:1.27* image and it runs the *sleep 3600* command

```
apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  containers:
  - name: bbox
    image: busybox:1.27
    command:
      - sleep
    args:
      - "3600"
```

- Create a Pod in the default namespace using this template

```
root@master1 $> kubectl create -f pod1.yaml
pod/pod1 created
root@master1 $>

root@master1 $> kubectl get pod
NAME     READY    STATUS    RESTARTS   AGE
pod1     1/1      Running   0          36s
root@master1 $>
```

- Create a new namespace named test-ns, and create a pod in it using the same template

```
root@master1 $> kubectl create namespace test-ns
namespace/test-ns created
root@master1 $> kubectl --namespace=test-ns create -f pod1.yaml
pod/pod1 created
root@master1 $> kubectl --namespace=test-ns get pod
NAME     READY    STATUS    RESTARTS   AGE
pod1     1/1      Running   0          12s
root@master1 $>
```

- Get more information about a pod.

```
root@master1 $> kubectl get pod pod1 -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2019-07-11T21:09:32Z"
  name: pod1
  namespace: default
  resourceVersion: "47006"
  selfLink: /api/v1/namespaces/default/pods/pod1
  uid: f2f04108-10a4-4f91-89f2-2d43335123c0
spec:
  containers:
  - args:
    - "3600"
    command:
    - sleep
    image: busybox:1.27
    imagePullPolicy: IfNotPresent
    name: bbox
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-j5bjs
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: worker3
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  volumes:
  - name: default-token-j5bjs
    secret:
      defaultMode: 420
      secretName: default-token-j5bjs
status:
  conditions:
  ...
  hostIP: 10.10.10.54
  phase: Running
  podIP: 10.36.0.1
  qosClass: BestEffort
  startTime: "2019-07-11T21:09:32Z"

root@master1 $>

root@master1 $> kubectl get pod -o wide
NAME READY  STATUS   RESTARTS  AGE   IP          NODE     NOMINATED NODE   READINESS GATES
pod1 1/1    Running  0         6m33s 10.36.0.1   worker3  <none>           <none>
root@master1 $>

root@master1 $> kubectl describe pod pod1
Name:          pod1
Namespace:     default
Priority:      0
Node:          worker3/10.10.10.54
Start Time:    Thu, 11 Jul 2019 21:09:32 +0000
Labels:        <none>
Annotations:   <none>
Status:        Running
IP:            10.36.0.1
Containers:
  bbox:
    Container ID:  docker://01b82f337109121b156a004b2d3b6545573ea6ce360251d70701cc1a34a5fe29
    Image:         busybox:1.27
    Image ID:      docker-pullable://busybox@sha256:bbc3a03235220b170ba48a157dd097dd1379299370
    Port:          <none>
    Host Port:     <none>
    Command:
      sleep
    Args:
      3600
    State:          Running
      Started:      Thu, 11 Jul 2019 21:09:36 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-j5bjs (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-j5bjs:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-j5bjs
    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  13m   default-scheduler    Successfully assigned default/pod1 to worker3
  Normal  Pulling    13m   kubelet, worker3     Pulling image "busybox:1.27"
  Normal  Pulled     13m   kubelet, worker3   Successfully pulled image "busybox:1.27"
  Normal  Created    13m   kubelet, worker3   Created container bbox
  Normal  Started    13m   kubelet, worker3   Started container bbox
```

- Execute a shell in the pod and check its hostname, IP address, network connectivity, and the running processes

```
root@master1 $> kubectl exec -ti pod1 sh
pod1@/ $> hostname
pod1
pod1@/ $>
pod1@/ $> ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
108: eth0@if109: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1376 qdisc noqueue
    link/ether 22:90:ed:e8:8b:12 brd ff:ff:ff:ff:ff:ff
    inet 10.31.0.2/12 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::2090:edff:fee8:8b12/64 scope link
       valid_lft forever preferred_lft forever
pod1@/ $>
pod1@/ $> ping -c1 kubernetes.io
PING kubernetes.io (23.236.58.218): 56 data bytes
64 bytes from 23.236.58.218: seq=0 ttl=39 time=154.143 ms

--- kubernetes.io ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 154.143/154.143/154.143 ms
pod1@/ $>
pod1@/ $> ps -ef
PID   USER     TIME   COMMAND
  1 root       0:00 sleep 3600
  5 root       0:00 sh
 15 root       0:00 ps -ef
pod1@/ $>
```

- Identify the node on which the pod is running and simulate a failure by pausing the corresponding VM. Check what is happenning with the pod.

Note:
Please be aware that the time required for detecting node failures and evict pods from a failed node depends on the following parameters:

- for the `kubelet` there is the `--node-status-update-frequency` that specifies how often the kubelet posts node status to master.
- for the `controller manager` we have the `--node-monitor-period` that specifies the period for syncing NodeStatus in NodeController, the `--node-monitor-grace-period` represents the amount of time which we allow running Node to be unresponsive before marking it unhealthy, and the `--pod-eviction-timeout` that specifies the grace period for deleting pods on failed nodes.

The default values for these parameters results a total timeout of 340 seconds for a pod to be deleted from a failed node. Please consider this timeout for the next exercises. In our environment the parameters of the controller manager can be altered by editing the `/etc/kubernetes/manifests/kube-controller-manager.yaml` file and addig the `--node-monitor-grace-period=20s` and `--pod-eviction-timeout=60s` fields to the command argument list.

```
root@master1 $> kubectl get pod pod1 -o wide
NAME   READY   STATUS    RESTARTS   AGE   IP          NODE    ...
pod1   1/1     Running   10         10h   10.36.0.1   worker3 ...
root@master1 $>

root@master1 $> exit
logout
Connection to master1 closed.
root@lab_machine $>
root@lab_machine $> virsh list
 Id    Name                           State
----------------------------------------------------
  1     master1                        running
  2     worker1                        running
  3     worker2                        running
  4     worker3                        running

root@lab_machine $> virsh suspend worker3
Domain worker1 suspended

root@lab_machine $> ssh master1

root@master1 $> kubectl get node
NAME      STATUS     ROLES    AGE   VERSION
 master1   Ready      master   19h   v1.15.0
 worker1   Ready      <none>   18h   v1.15.0
 worker2   Ready      <none>   18h   v1.15.0
 worker3   NotReady   <none>   18h   v1.15.0

root@master1 $> kubectl exec -ti pod1 sh
Error from server: error dialing backend: dial tcp 10.10.10.54:10250: connect: no route
to host

root@master1 $> kubectl get pod
NAME      READY     STATUS         RESTARTS    AGE
pod1      1/1       Terminating    0           17m
```

- Resume the VM and check the status of the pod again

```
root@master1 $> exit
logout
Connection to master1 closed.
root@lab_machine $> virsh resume worker3
Domain worker1 resumed

root@lab_machine $> ssh master1

root@master1 $> kubectl get node
NAME      STATUS    AGE       VERSION
master1   Ready    master   19h   v1.15.0
worker1   Ready    <none>   19h   v1.15.0
worker2   Ready    <none>   18h   v1.15.0
worker3   Ready    <none>   18h   v1.15.0

root@master1 $> kubectl get pod
No resources found.
```

- Delete the pod from the test-ns namespace, and the test-ns namespace also.

```
root@master1 $> kubectl delete namespace test-ns
namespace "test-ns" deleted

root@master1 $> kubectl get pod --namespace=test-ns
No resources found.
root@master1 $> kubectl get namespace
NAME          STATUS    AGE
default       Active    1d
kube-public   Active    1d
kube-system   Active    1d
```

Task 2: Replication controller operations

- Create a replication controller that runs *sleep 3600* in a busybox container, and it has 2 replicas

```
root@master1 $> cat rc1.yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: rc1
spec:
  replicas: 2
  selector:
    app: bb
  template:
    metadata:
        name: bbox
        labels:
          app: bb
    spec:
        containers:
        - name: bb
          image: busybox
          command:
```

```
          - sleep
          - "1000"

root@master1 $> kubectl apply -f rc1.yaml
replicationcontroller/rc1 created
root@master1 $> kubectl get rc -o wide
NAME      DESIRED    CURRENT    READY     AGE       CONTAINER(S)    IMAGE(S)    SELECTOR
rc1       2          2          2         1m        bb              busybox     app=bb
root@master1 $>

root@master1 $> kubectl get pod -o wide
NAME         READY    STATUS     RESTARTS   AGE      IP           NODE    ...
rc1-5jvtn    1/1      Running    0          3m58s    10.42.0.1    worker2 ...
rc1-spnc9    1/1      Running    0          3m58s    10.36.0.1    worker3 ...
root@master1 $>
```

- Scale the replication controller to have 3 replicas

```
root@master1 $> kubectl scale replicationcontroller rc1 --replicas=3
replicationcontroller/rc1 scaled
root@master1 $> kubectl get rc -o wide
NAME      DESIRED    CURRENT    READY     AGE       CONTAINER(S)    IMAGE(S)    SELECTOR
rc1       3          3          3         3m       bb              busybox     app=bb
root@master1 $> kubectl get pod -o wide
NAME         READY    STATUS     RESTARTS   AGE      IP           NODE    ...
rc1-5jvtn    1/1      Running    0          6m18s    10.42.0.1    worker2 ...
rc1-7dmvn    1/1      Running    0          19s      10.44.0.2    worker1 ...
rc1-spnc9    1/1      Running    0          6m18s    10.36.0.1    worker3 ...

root@master1 $>
```

- simulate again the failure of a node, and check the status of the pods

```
root@master1 $> exit
logout
Connection to master1 closed.
root@lab_machine $> virsh suspend worker3
Domain worker3 suspended
```

Wait 5 minutes due to the `pod-eviction-timeout` before you continue.

```
root@lab_machine $> ssh master1

root@master1 $>
root@master1 $> kubectl get node
NAME       STATUS      AGE        VERSION
master1    Ready       master     1d         v1.11.0
worker1    Ready       <none>     1d         v1.11.0
worker2    Ready       <none>     1d         v1.11.0
worker3    NotReady    <none>     1d         v1.11.0

root@master1 $>
root@master1 $> kubectl get pod -o wide
NAME         READY    STATUS         RESTARTS    AGE      IP           NODE    ...
rc1-5jvtn    1/1      Running        1           20m      10.42.0.1    worker2 ...
rc1-7dmvn    1/1      Running        0           14m      10.44.0.2    worker1 ...
```

```
rc1-spnc9   1/1     Terminating   0          20m    10.36.0.1   worker3 ...
rc1-t85qb   1/1     Running       0          7m8s   10.42.0.2   worker2 ...
```

- Resume the node. Check the status of the pods once the node is back in the cluster. Delete the replication controller.

```
root@master1 $> exit
logout
Connection to master1 closed.
root@lab_machine $> virsh resume worker3
Domain worker3 resumed

root@lab_machine $> ssh master1

root@master1 $> kubectl get node
NAME           STATUS   ROLES    AGE   VERSION
node/master1   Ready    master   21h   v1.15.0
node/worker1   Ready    <none>   20h   v1.15.0
node/worker2   Ready    <none>   20h   v1.15.0
node/worker3   Ready    <none>   20h   v1.15.0
root@master1 $>
root@master1 $> kubectl get pod -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP          NODE    ...
rc1-5jvtn   1/1     Running   1          24m   10.42.0.1   worker2 ...
rc1-7dmvn   1/1     Running   1          18m   10.44.0.2   worker1 ...
rc1-t85qb   1/1     Running   0          11m   10.42.0.2   worker2 ...

root@master1 $>
root@master1 $> kubectl delete rc rc1
replicationcontroller "rc1" deleted
root@master1 $> kubectl get rc
No resources found.
root@master1 $> kubectl get pod -o wide
No resources found.
```

Task 3: Working with deployments

- Create a deployment that runs 2 replicas of nginx version 1.7.9

```
root@master1 $> cat dep1.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
        labels:
          app: nginx
    spec:
        containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
          - containerPort: 80

root@master1 $> kubectl apply -f dep1.yaml --record
deployment.apps/nginx-deployment created

root@master1 $> kubectl get all
NAME                                    READY   STATUS              RESTARTS   AGE
pod/nginx-deployment-5754944d6c-484tk   0/1     ContainerCreating   0          10s
pod/nginx-deployment-5754944d6c-d5r56   0/1     ContainerCreating   0          10s


NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   21h


NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   0/2     2            0           10s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-5754944d6c   2         2         0       10s
```

- Upgrade the nginx containers to image 1.10.3

```
root@master1 $> kubectl set image deployment/nginx-deployment nginx=nginx:1.10.3 --record
deployment.extensions/nginx-deployment image updated
root@master1 $> kubectl get all
NAME                                    READY   STATUS              RESTARTS   AGE
pod/nginx-deployment-5754944d6c-484tk   1/1     Running             0          115s
pod/nginx-deployment-5754944d6c-d5r56   1/1     Running             0          115s
pod/nginx-deployment-9d4cd975-vg9vs     0/1     ContainerCreating   0          10s


NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   21h


NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   2/2     1            2           115s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-5754944d6c   2         2         2       115s
replicaset.apps/nginx-deployment-9d4cd975     1         1         0       10s

root@master1 $> kubectl get all
NAME                                  READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-9d4cd975-b2sd7   1/1     Running   0          2m3s
pod/nginx-deployment-9d4cd975-vg9vs   1/1     Running   0          2m14s


NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   21h
```

```
NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   2/2     2            2           3m59s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-5754944d6c   0         0         0       3m59s
replicaset.apps/nginx-deployment-9d4cd975     2         2         2       2m14s
```

- Check the rollout history of the deployment, and rollback to the previous version

```
root@master1 $> kubectl rollout history deployment nginx-deployment
deployments "nginx-deployment"
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=dep1.yaml --record=true
2         kubectl set image deployment/nginx-deployment nginx=nginx:1.10.3 --record=true

root@master1 $> kubectl rollout undo deployment nginx-deployment
deployment.extensions/nginx-deployment rolled back

root@master1 $> kubectl get all
NAME                                   READY   STATUS              RESTARTS   AGE
pod/nginx-deployment-5754944d6c-lbz7l   1/1     Running             0          71s
pod/nginx-deployment-5754944d6c-zr2c8   1/1     Running             0          68s
pod/nginx-deployment-9d4cd975-9vpxx     0/1     ContainerCreating   0          2s


NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   21h


NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   2/2     1            2           7m48s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-5754944d6c   2         2         2       7m48s
replicaset.apps/nginx-deployment-9d4cd975     1         1         0       7m37s

root@master1 $> kubectl get all
NAME                                   READY   STATUS        RESTARTS   AGE
pod/nginx-deployment-5754944d6c-lbz7l   0/1     Terminating   0          75s
pod/nginx-deployment-5754944d6c-zr2c8   0/1     Terminating   0          72s
pod/nginx-deployment-9d4cd975-9vpxx     1/1     Running       0          6s
pod/nginx-deployment-9d4cd975-xhkfg     1/1     Running       0          4s


NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   21h


NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   2/2     2            2           7m52s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-5754944d6c   0         0         0       7m52s
replicaset.apps/nginx-deployment-9d4cd975     2         2         2       7m41s

root@master1 $>
```

- Delete the deployment

```
root@master1 $> kubectl delete deployment nginx-deployment
deployment.extensions "nginx-deployment" deleted
root@master1 $> kubectl get all
NAME             CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   10.96.0.1     <none>        443/TCP   1d
root@master1 $>
```

Task 4: Using Jobs

- Create a Job named *connection-test* that start a pod which will send one ICMP Echo Request (ping) to www.google.com. The job should run 10 Pods to successful completion. (Use the busybox image)

```
root@master1 $> cat job1.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: connection-test
spec:
      completions: 10
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/ping
            - -c
            - "1"
            - www.google.com
          restartPolicy: OnFailure

root@master1 $> kubectl apply -f job1.yaml
job.batch/connection-test created
```

- Test the evolution of the job

```
root@master1 $> kubectl get jobs --watch
NAME               DESIRED   SUCCESSFUL   AGE
connection-test   10        0            2s
connection-test   10        1         7s
connection-test   10        2         15s
connection-test   10        3         22s
connection-test   10        4         30s
connection-test   10        5         41s
connection-test   10        6         48s
connection-test   10        7         55s
connection-test   10        8         1m
connection-test   10        9         1m
connection-test   10        10        1m
^C

root@master1 $> kubectl get pod
```

```
NAME                    READY     STATUS      RESTARTS   AGE
connection-test-4dzrx   0/1       Completed   0          54s
connection-test-5lls4   0/1       Completed   0          47s
connection-test-5mxrj   0/1       Completed   0          1m
connection-test-7xllv   0/1       Completed   0          1m
connection-test-9hssg   0/1       Completed   0          34s
connection-test-h4jtl   0/1       Completed   0          40s
connection-test-pv8mc   0/1       Completed   0          1m
connection-test-rb2zl   0/1       Completed   0          1m
connection-test-tf46l   0/1       Completed   0          1m
connection-test-tskbq   0/1       Completed   0          28s
```

- Delete the job, and verify that the Pods were deleted also

```
root@master1 $> kubectl delete job connection-test
job.batch "connection-test" deleted
root@master1 $> kubectl get pod
No resources found.
root@master1 $>
```

Task 5: Using DaemonSets

- Execute one instance of `nginx` on all the worker nodes in the cluster. (Use daemonset for it.)

```
root@master1 $> cat ds1.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: ds-demo
spec:
  selector:
    matchLabels:
      appname: nginx
  template:
    metadata:
      labels:
        appname: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/master

root@master1 $> kubectl apply -f ds1.yaml
daemonset.apps/ds-demo created
```

- Verify the status of the DaemonSet, and that the nginx has been started on all the nodes.

```
root@master1 $> kubectl get ds
NAME      DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
ds-demo   4         4         4         4            4           <none>          44s

root@master1 $> kubectl get pod -o wide
NAME            READY     STATUS    RESTARTS   AGE       IP            NODE    ...
ds-demo-82zzh   1/1       Running   0          1m        10.40.0.5     master1 ...
ds-demo-b7hcr   1/1       Running   0          1m        10.32.0.2     worker1 ...
ds-demo-b829w   1/1       Running   0          1m        10.46.0.1     worker2 ...
ds-demo-x7ddx   1/1       Running   0          1m        10.47.128.2   worker3 ...
```

- Delete the daemon set, and make sure that the Pods have been deleted also.

```
root@master1 $> kubectl delete daemonset ds-demo
daemonset.extensions "ds-demo" deleted
root@master1 $> kubectl get daemonset
No resources found.
root@master1 $> kubectl get pod
No resources found.
root@master1 $>
```

Lab 5: Scheduling and node management

Task 1: Scheduling Pods to nodes

- Label the nodes `worker1` and `worker2` with `ssd=yes`, and the nodes `worker2` and `worker3` with `spinning=yes`

```
root@master1 $>kubectl label node worker1 ssd=yes
node/worker1 labeled
root@master1 $>kubectl label node worker2 ssd=yes
node/worker2 labeled
root@master1 $>kubectl label node worker2 spinning=yes
node/worker2 labeled
root@master1 $>kubectl label node worker3 spinning=yes
node/worker3 labeled
root@master1 $>kubectl get node --show-labels
NAME      STATUS   ROLES    AGE   VERSION   LABELS
master1   Ready    master   32h   v1.15.0   //node-role.kubernetes.io/master=
worker1   Ready    <none>   32h   v1.15.0   //kubernetes.io/os=linux,ssd=yes
worker2   Ready    <none>   32h   v1.15.0   //kubernetes.io/os=linux,spinning=yes,ssd=yes
worker3   Ready    <none>   32h   v1.15.0   //kubernetes.io/os=linux,spinning=yes
```

- Create a deployment named *with-ssd* that will start 2 nginx pods on the nodes that have the label `ssd=yes‘ using the ‘‘nodeSelector`

```
root@master1 $> cat nodeselector.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: with-ssd
spec:
  replicas: 2
  selector:
   matchLabels:
    app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      nodeSelector:
        ssd: "yes"
      containers:
      - name: nginx
```

```
        image: nginx
        ports:
        - containerPort: 80
```

```
root@master1 $> kubectl apply -f nodeselector.yaml
deployment.apps/with-ssd created
```

```
root@master1 $> kubectl get pod -o wide --watch
NAME                        READY   STATUS    RESTARTS   AGE   IP          NODE      //
with-ssd-8644b46976-5wqct   1/1     Running   0          58s   10.42.0.2   worker2   //
with-ssd-8644b46976-fddpw   1/1     Running   0          58s   10.44.0.2   worker1   //
```

- Increase the number of replicas for the deployment and verify that only the `worker1` and `worker2` nodes are used.

```
root@master1 $> kubectl scale deployment with-ssd --replicas=4
deployment.extensions/with-ssd scaled
```

```
root@master1 $> kubectl get pod -o wide
NAME                        READY   STATUS    RESTARTS  AGE   IP          NODE     //
with-ssd-8644b46976-5wqct   1/1     Running   0         24m   10.42.0.2   worker2  //
with-ssd-8644b46976-7qq8f   1/1     Running   0         106s  10.44.0.3   worker1  //
with-ssd-8644b46976-fddpw   1/1     Running   0         24m   10.44.0.2   worker1  //
with-ssd-8644b46976-lzhjb   1/1     Running   0         106s  10.42.0.3   worker2  //
```

- Delete the deployment

```
root@master1 $> kubectl delete -f nodeselector.yaml
deployment.apps "with-ssd" deleted

root@master1 $> kubectl get pod
No resources found.
root@master1 $> kubectl get deployment
No resources found.
```

Task 2: Using affinities

- Create a deployment named *no-spinning* that will run 3 replicas of `redis` on nodes that don't use spinning disks (don't have the `spinning=yes` label).

```
root@master1 $> cat node_affinity.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: no-spinning
spec:
  replicas: 3
  selector:
   matchLabels:
    app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
                 - key: spinning
                   operator: NotIn
                   values:
                    - "yes"
      containers:
      - name: redis
        image: redis
```

```
root@master1 $> kubectl apply -f node_affinity.yaml
deployment.apps/no-spinning created
root@master1 $> kubectl get pod -o wide
NAME                           READY   STATUS    RESTARTS   AGE  IP           NODE    //
no-spinning-7666f45b5f-bltv8   1/1     Running   0          25s  10.44.0.4    worker1 //
no-spinning-7666f45b5f-fksxt   1/1     Running   0          25s  10.44.0.3    worker1 //
no-spinning-7666f45b5f-g4749   1/1     Running   0          25s  10.44.0.2    worker1 //
```

- Create a deployment named *next-to-redis* that will run `nginx` on a node that already runs `redis` (has the label `app=redis`)

```
root@master1 $> cat pod_affinity.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: next-to-redis
spec:
  replicas: 1
  selector:
   matchLabels:
    app: nginx
  template:
    metadata:
        labels:
          app: nginx
    spec:
        affinity:
          podAffinity:
                requiredDuringSchedulingIgnoredDuringExecution:
                - labelSelector:
                        matchExpressions:
                          - key: app
                            operator: In
                            values:
                               - redis
                  topologyKey: kubernetes.io/hostname
        containers:
        - name: nginx
          image: nginx


root@master1 $> kubectl apply -f pod_affinity.yaml
```

```
deployment.apps/next-to-redis created

root@master1 $> kubectl get pod -o wide
NAME                             READY   STATUS    RESTARTS   AGE    IP          NODE    //
next-to-redis-698b8db5dd-4tkj8   1/1     Running   0          12s    10.44.0.5   worker1 //
no-spinning-7666f45b5f-bltv8     1/1     Running   0          3m39s  10.44.0.4   worker1 //
no-spinning-7666f45b5f-fksxt     1/1     Running   0          3m39s  10.44.0.3   worker1 //
no-spinning-7666f45b5f-g4749     1/1     Running   0          3m39s  10.44.0.2   worker1 //
```

- Delete both deployments

```
root@master1 $> kubectl delete -f node_affinity.yaml
deployment.apps "no-spinning" deleted
root@master1 $> kubectl delete -f pod_affinity.yaml
deployment.apps "next-to-redis" deleted
root@master1 $> kubectl get pod -o wide
No resources found.
```

- Create again the *next-to-redis* deployment using the same .yaml file, and check pod status

```
root@master1 $> kubectl apply -f pod_affinity.yaml
deployment.apps/next-to-redis created

root@master1 $> kubectl get pod
NAME                             READY   STATUS     RESTARTS   AGE
next-to-redis-698b8db5dd-bl9sw   0/1     Pending    0          68s

root@master1 $>
```

- Fix the issue to have the pod in running state. (You can watch the events in a different terminal using the `kubectl get pod -o wide --watch` command.)

- Delete the deployments created during the exercise.

Task 3: Pod priorities - OPTIONAL

In this task we are generating a situation where a new pod cannot be scheduled due to lack of resources, and afterwards we will mitigate this situation by adding priorities to the new pod.

> **Note:**
> Please clean up all the workloads you may have created so far, in order to have enough resources available for this task.

- Create a deployment with 3 replicas using the nginx image, and the memory requirement being set to 512MB. Verify that the pods belonging to this deployment are running.

```
root@master1 $> cat prio1.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
   matchLabels:
    app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        resources:
          requests:
             memory: "512Mi"
root@master1 $>
root@master1 $> kubectl apply -f prio1.yaml
deployment.apps/nginx created

root@master1 $> kubectl get pod -o wide
NAME                     READY   STATUS   //   IP          NODE      NOMINATED NODE //
nginx-66499b995f-5q7mx   1/1     Running  //   10.42.0.1   worker2   <none>         //
nginx-66499b995f-fvdlt   1/1     Running  //   10.36.0.1   worker3   <none>         //
nginx-66499b995f-hzjh4   1/1     Running  //   10.44.0.2   worker1   <none>         //
root@master1 $>
```

- Create a new deployment with 3 replicas using the busybox image, and the memory requirement being set to 512MB. Run the `sleep 36000` command in the container. Verify that the pods belonging to this deployment are running.

```
root@master1 $> cat prio2.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bb
spec:
  replicas: 3
  selector:
   matchLabels:
    app: bb
  template:
    metadata:
      labels:
        app: bb
    spec:
      containers:
      - name: bb
        image: busybox
        command:
          - sleep
        args:
          - "36000"
        resources:
          requests:
             memory: "512Mi"

root@master1 $>
root@master1 $> kubectl apply -f prio2.yaml
deployment.apps/bb created
root@master1 $> kubectl get pod -o wide
NAME                     READY   STATUS  //  IP          NODE      NOMINATED NODE //
bb-7577fc9647-2j6v8      1/1     Running //  10.44.0.3   worker1   <none>         //
bb-7577fc9647-cw5rd      1/1     Running //  10.42.0.2   worker2   <none>         //
bb-7577fc9647-kkwdk      1/1     Running //  10.36.0.2   worker3   <none>         //
nginx-66499b995f-5q7mx   1/1     Running //  10.42.0.1   worker2   <none>         //
nginx-66499b995f-fvdlt   1/1     Running //  10.36.0.1   worker3   <none>         //
nginx-66499b995f-hzjh4   1/1     Running //  10.44.0.2   worker1   <none>         //
root@master1 $>
```

Note:
All pods are supposed to be running at this point. If you have Pending pods, then you have already reached the situation where the requirements of the new pod cannot be satisfied.

- increase the number of replicas of the second deployment until you reach the resource limits, and new pods cannot be scheduled.

```
root@master1 $> kubectl scale deployment bb --replicas=4
deployment.extensions/bb scaled
root@master1 $> kubectl get pod
NAME                     READY   STATUS     RESTARTS   AGE
bb-7577fc9647-2j6v8      1/1     Running    0          9m47s
bb-7577fc9647-cw5rd      1/1     Running    0          9m47s
bb-7577fc9647-kkwdk      1/1     Running    0          9m47s
bb-7577fc9647-v7tt5      0/1     Pending    0          8s
nginx-66499b995f-5q7mx   1/1     Running    0          27m
nginx-66499b995f-fvdlt   1/1     Running    0          27m
nginx-66499b995f-hzjh4   1/1     Running    0          27m
root@master1 $> kubectl describe pod bb-7577fc9647-v7tt5
Name:           bb-7577fc9647-v7tt5
Namespace:      default
Priority:       0
Node:           <none>
Labels:         app=bb
                pod-template-hash=7577fc9647
Annotations:    <none>
Status:         Pending
IP:
Controlled By:  ReplicaSet/bb-7577fc9647
Containers:
  bb:
    Image:      busybox
    Port:       <none>
    Host Port:  <none>
    Command:
      sleep
    Args:
      36000
    Requests:
      memory:     512Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-j5bjs (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  default-token-j5bjs:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-j5bjs
    Optional:    false
QoS Class:       Burstable
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
  ----     ------            ----                ----               -------
  Warning  FailedScheduling  24s (x2 over 24s)  default-scheduler  0/4 nodes are availa
  ble: 1 node(s) had taints that the pod didn't tolerate, 3 Insufficient memory.
root@master1 $>
```

- Create a `PriorityClass` named `high-prio` and set its value to `1000`.

```
root@master1 $> kubectl create priorityclass high-prio --value=1000
priorityclass.scheduling.k8s.io/high-prio created
root@master1 $>
```

- Update the definition for the second deployment (bb) to use the afore created priority class, and apply the change.

```
root@master1 $> cat prio3.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bb
spec:
  replicas: 4
  selector:
   matchLabels:
    app: bb
  template:
    metadata:
      labels:
        app: bb
    spec:
      priorityClassName: high-prio
      containers:
      - name: bb
        image: busybox
        command:
          - sleep
        args:
          - "36000"
        resources:
          requests:
             memory: "512Mi"

root@master1 $>
root@master1 $> kubectl apply -f prio3.yaml
deployment.apps/bb configured
root@master1 $>
```

- Watch the changes of the pods after applying the change, and check the final state.

```
root@master1 $> kubectl get pod -o wide -w
NAME                     READY   STATUS       // IP          NODE      NOMINATED NODE //
bb-7577fc9647-4rcs7      1/1     Terminating  // 10.42.0.2   worker2   <none>         //
bb-7577fc9647-grglx      1/1     Terminating  // 10.44.0.3   worker1   <none>         //
...
bb-b6f4f7b86-tskwb       0/1     Pending           // <none>      <none>    worker1   //
nginx-66499b995f-hzjh4   1/1     Terminating       // 10.44.0.2   worker1   <none>    //
nginx-66499b995f-pzgr7   0/1     Pending           // <none>      <none>    <none>    //
nginx-66499b995f-pzgr7   0/1     Pending           // <none>      <none>    <none>    //
nginx-66499b995f-hzjh4   0/1     Terminating       // 10.44.0.2   worker1   <none>    //
nginx-66499b995f-hzjh4   0/1     Terminating       // 10.44.0.2   worker1   <none>    //
nginx-66499b995f-hzjh4   0/1     Terminating       // 10.44.0.2   worker1   <none>    //
bb-b6f4f7b86-tskwb       0/1     Pending           // <none>      worker1   worker1   //
bb-b6f4f7b86-tskwb       0/1     ContainerCreating // <none>      worker1   <none>    //
bb-b6f4f7b86-tskwb       1/1     Running           // 10.44.0.2   worker1   <none>    //

root@master1 $> kubectl get pod -o wide
NAME                     READY   STATUS   IP          NODE      NOMINATED NODE //
bb-b6f4f7b86-fjc77       1/1     Running  10.42.0.2   worker2   <none>         //
bb-b6f4f7b86-fvhrx       1/1     Running  10.44.0.3   worker1   <none>         //
bb-b6f4f7b86-lz75x       1/1     Running  10.36.0.2   worker3   <none>         //
bb-b6f4f7b86-tskwb       1/1     Running  10.44.0.2   worker1   <none>         //
nginx-66499b995f-5q7mx   1/1     Running  10.42.0.1   worker2   <none>         //
nginx-66499b995f-fvdlt   1/1     Running  10.36.0.1   worker3   <none>         //
nginx-66499b995f-pzgr7   0/1     Pending  <none>      <none>    <none>         //
root@master1 $>
```

Observe that the higher priority pod has evicted the lower priority pod in order to be able to start.

- Remove all the deployments created at this task.

Lab 6: Accessing the applications

Task 1: Working with services

- Create an nginx service that has 2 replicas and it is exposing port 80.

```
root@master1 $> kubectl create deployment --image=nginx nginx
deployment.apps/nginx created
root@master1 $> kubectl expose deployment nginx --port=80
service/nginx exposed
root@master1 $> kubectl scale deployment nginx --replicas=2
deployment.extensions/nginx scaled
root@master1 $> kubectl get all
NAME                          READY   STATUS    RESTARTS   AGE
pod/nginx-554b9c67f9-jmrpn    1/1     Running   0          3m31s
pod/nginx-554b9c67f9-zbvbl    1/1     Running   0          25s


NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP    PORT(S)    AGE
service/kubernetes      ClusterIP   10.96.0.1        <none>         443/TCP    27h
service/nginx           ClusterIP   10.106.222.180   <none>         80/TCP     73s


NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx      2/2     2            2           3m31s

NAME                                   DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-554b9c67f9       2         2         2       3m31s

root@master1 $> curl http://10.106.222.180
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {

  ...

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

- Check the endpoints belonging to the service

```
root@master1 $> kubectl get endpoints nginx
NAME     ENDPOINTS                   AGE
nginx    10.36.0.1:80,10.42.0.1:80   4m3s
```

- Increase the number of replicas in the nginx service, and check the endpoints again

```
root@master1 $> kubectl scale deployment nginx --replicas=3
deployment.extensions/nginx scaled
root@master1 $>
root@master1 $> kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
nginx-554b9c67f9-jmrpn   1/1     Running   0          7m8s
nginx-554b9c67f9-qzmln   1/1     Running   0          8s
nginx-554b9c67f9-zbvbl   1/1     Running   0          4m2s
root@master1 $> kubectl get endpoints nginx
NAME     ENDPOINTS                                AGE
nginx    10.36.0.1:80,10.42.0.1:80,10.44.0.2:80   5m26s
```

- Remove the label *app=nginx* from one of the pods, and check what's going on.

```
root@master1 $> kubectl get pod --show-labels
NAME                     READY   STATUS    RESTARTS   AGE   LABELS
nginx-554b9c67f9-jmrpn   1/1     Running   0          17m   app=nginx,pod-template-hash=
554b9c67f9
nginx-554b9c67f9-qzmln   1/1     Running   0          10m   app=nginx,pod-template-hash=
554b9c67f9
nginx-554b9c67f9-zbvbl   1/1     Running   0          14m   app=nginx,pod-template-hash=
554b9c67f9

root@master1 $> kubectl label pod nginx-554b9c67f9-qzmln app-
pod/nginx-554b9c67f9-qzmln labeled
root@master1 $> kubectl get pod --show-labels
NAME                     READY   STATUS    //  LABELS
nginx-554b9c67f9-h2868   1/1     Running //  app=nginx,pod-template-hash=554b9c67f9
nginx-554b9c67f9-jmrpn   1/1     Running //  app=nginx,pod-template-hash=554b9c67f9
nginx-554b9c67f9-qzmln   1/1     Running //  pod-template-hash=554b9c67f9
nginx-554b9c67f9-zbvbl   1/1     Running //  app=nginx,pod-template-hash=554b9c67f9

root@master1 $> kubectl get endpoints nginx
NAME     ENDPOINTS                                AGE
nginx    10.36.0.1:80,10.42.0.1:80,10.44.0.3:80   20m
```

- Start a new busybox container and verify that the nginx service is known.

```
root@master1 $> kubectl create -f pod1.yaml
pod/pod1 created

root@master1 $> kubectl exec -ti pod1 sh
root@pod1 / $> env
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_SERVICE_PORT=443
HOSTNAME=pod1
SHLVL=1
HOME=/root
NGINX_PORT_80_TCP=tcp://10.106.222.180:80
TERM=xterm
```

```
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
NGINX_SERVICE_HOST=10.106.222.180
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
NGINX_PORT=tcp://10.106.222.180:80
NGINX_SERVICE_PORT=80
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_SERVICE_PORT_HTTPS=443
PWD=/
KUBERNETES_SERVICE_HOST=10.96.0.1
NGINX_PORT_80_TCP_ADDR=10.106.222.180
NGINX_PORT_80_TCP_PORT=80
NGINX_PORT_80_TCP_PROTO=tcp

root@pod1 / $> nslookup nginx
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      nginx
Address 1: 10.106.222.180 nginx.default.svc.cluster.local
```

- Create a new service object of type *NodePort* that is exposing port 80 of the nginx pods

```
root@master1 $> kubectl expose deployment nginx --type=NodePort --port=80 --name=nginx-np
service/nginx-np exposed

root@master1 $> kubectl get service
NAME         TYPE         CLUSTER-IP       EXTERNAL-IP    PORT(S)         AGE
kubernetes   ClusterIP    10.96.0.1        <none>         443/TCP         27h
nginx        ClusterIP    10.106.222.180   <none>         80/TCP          25m
nginx-np     NodePort     10.97.54.214     <none>         80:31928/TCP    11s
```

- From your lab machine try to acces the retrieved port using the addresses of your nodes

```
root@lab_machine $> NODES="master1 worker1 worker2 worker3"
root@lab_machine $> for i in $NODES
> do echo ${i}
> curl ${i}:31928|head -5
> echo
> done
master1
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload Upload   Total   Spent    Left  Speed
100   612  100   612    0     0  99674      0 --:--:-- --:--:-- --:--:--   99k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>

worker1
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                               Dload  Upload   Total   Spent    Left  Speed
100   612  100   612    0     0  99029      0 --:--:-- --:--:-- --:--:--   99k
<!DOCTYPE html>
<html>
```

```
<head>
<title>Welcome to nginx!</title>
<style>

worker2
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                               Dload  Upload   Total   Spent    Left  Speed
100   612  100   612    0     0   512k      0 --:--:-- --:--:-- --:--:--  597k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>

worker3
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                               Dload  Upload   Total   Spent    Left  Speed
100   612  100   612    0     0   272k      0 --:--:-- --:--:-- --:--:--  298k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
```

- Delete the nginx, nginx-np services, the nginx deployment, and any pods that may have left over from the exercise

```
root@master1 $> kubectl delete svc nginx-np
service "nginx-np" deleted
root@master1 $> kubectl delete svc nginx
service "nginx" deleted
root@master1 $> kubectl delete deployment nginx
deployment.extensions "nginx" deleted
root@master1 $> kubectl get all
NAME                          READY   STATUS    RESTARTS   AGE
pod/nginx-554b9c67f9-qzmln    1/1     Running   0          25m
pod/pod1                      1/1     Running   0          7m26s


NAME                   TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/kubernetes     ClusterIP   10.96.0.1     <none>        443/TCP   27h

root@master1 $> kubectl delete pod nginx-554b9c67f9-qzmln
pod "nginx-554b9c67f9-qzmln" deleted
root@master1 $> kubectl delete pod pod1
pod "pod1" deleted
```

Task 2: working with Ingress

- Use the *ingr-controller.yaml* file from the `/labfiles/k8s/Accessing_applications` directory to create the ingress controller and the default backend.

```
root@master1 $> kubectl create -f ingr-controller.yaml
clusterrole.rbac.authorization.k8s.io/ingress created
role.rbac.authorization.k8s.io/ingress-ns created
rolebinding.rbac.authorization.k8s.io/ingress-ns-binding created
clusterrolebinding.rbac.authorization.k8s.io/ingress-binding created
deployment.apps/default-http-backend created
service/default-http-backend created
serviceaccount/ingress created
deployment.apps/nginx-ingress-controller created
```

- Create a service that runs two replicas of the ghost image, and exposes port 2368 as NodePort

```
root@master1 $> kubectl create deployment --image=ghost ghost
deployment.apps/ghost created
root@master1 $> kubectl scale deployment ghost --replicas=2
deployment.extensions/ghost scaled
root@master1 $> kubectl expose deployment ghost --port=2368 --type=NodePort
service/ghost exposed
```

- Please check the IP address of the node where the ingress controller is running

```
root@master1 $> kubectl get pod --namespace=kube-system \
> -l k8s-app=nginx-ingress-controller -o wide
NAME                                        READY   STATUS  // IP            NODE    //
nginx-ingress-controller-7f9dfbff87-sg2bc   1/1     Running // 10.10.10.54   worker3 //
```

- Review the ingress_ghost file from your `/labfiles/k8s/Accessing_applications` directory, and create an ingress from it. Adjust the `host` field to the IP identified in the previous step.

```
root@master1 $> cat ingress_ghost.yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ghost
spec:
  rules:
  - host: ghost.10.10.10.54.nip.io
    http:
          paths:
          - backend:
                  serviceName: ghost
                  servicePort: 2368
root@master1 $>

root@master1 $> kubectl apply -f ingress_ghost.yaml
ingress.extensions/ghost created
root@master1 $>

root@master1 $> curl ghost.10.10.10.54.nip.io | sed '10,$d'
% Total     % Received % Xferd  Average Speed    Time     Time      Time  Current
                                                      Dload  Upload    Total    Spent    Left
100 16736  100 16736    0      0    4728      0  0:00:03  0:00:03 --:--:--  4727
<!DOCTYPE html>
<html lang="en">
<head>
```

```
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<title>Ghost</title>
<meta name="HandheldFriendly" content="True" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

- Remove the ingress, and the associated service and deployment.

```
root@master1 $> kubectl delete ingress ghost
ingress.extensions "ghost" deleted
root@master1 $> kubectl delete svc ghost
service "ghost" deleted
root@master1 $> kubectl delete deployment ghost
deployment.extensions "ghost" deleted
root@master1 $> kubectl get all
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   5d
```

Lab 7: Using persistent storage

Task 1: Share a volume in two containers

- Create a pod with 2 busybox containers that are sharing the same volume of *emptyDir* type. The first container should mount it as */data*, while the second as */input*

```
root@master1 $> cat shvol.yaml
apiVersion: v1
kind: Pod
metadata:
  name: shvol1
  namespace: default

spec:
  containers:
    - image: busybox
      command:
            - sleep
            - "3600"
      volumeMounts:
            - mountPath: /data
              name: test
      name: processor
    - image: busybox
      command:
            - sleep
            - "3600"
      volumeMounts:
            - mountPath: /input
              name: test
      name: reader
  volumes:
    - name: test
      emptyDir: {}

root@master1 $> kubectl apply -f shvol.yaml
pod/shvol1 created
root@master1 $> kubectl get pod
NAME      READY     STATUS              RESTARTS   AGE
shvol1    2/2       Running   0          29s
```

- Execute an interactive shell in the containers and test the availability of the mountpoints, and the sharing of data

```
root@master1 $> kubectl exec shvol1 --container=reader -ti sh
shvol1 $> df -h | grep input
/dev/vda1                 8.0G      4.1G      3.9G  51% /input
shvol1 $> echo reader > /input/f1
shvol1 $> cat /input/f1
reader
shvol1 $> exit
root@master1 $>
root@master1 $> kubectl exec shvol1 --container=processor -ti sh
shvol1 $> df -h /data
Filesystem                Size      Used Available Use% Mounted on
/dev/vda1                 8.0G      4.1G      3.9G  51% /data
shvol1 $> cat /data/f1
reader
shvol1 $> echo processor >> /data/f1
shvol1 $> cat /data/f1
reader
processor
shvol1 $> exit
root@master1 $> kubectl exec shvol1 --container=reader -ti cat /input/f1
reader
processor
```

- Verify that gluster is sharing the /vol1, /vol2, and the /vol3 directories on your **lab machine** using gluster volume info.

```
root@lab_machine $> gluster volume info
...

Volume Name: vol1
Type: Distribute
Volume ID: f6efc489-700e-4c28-92d4-c266c6a5b8ed
Status: Started
Snapshot Count: 0
Number of Bricks: 1
Transport-type: tcp
Bricks:
Brick1: 10.10.10.1:/vol1
Options Reconfigured:
nfs.disable: on
transport.address-family: inet

Volume Name: vol2
...
```

- Use the glusterfs-endpoints.yaml file from directory `/labfiles/k8s/Persistent_storage` to create the glusterfs endpoint.

```
root@master1 $> kubectl apply -f glusterfs-endpoints.yaml
endpoints/glusterfs-cluster created

root@master1 $> kubectl get endpoints
NAME                ENDPOINTS          AGE
glusterfs-cluster   10.10.10.1:1       33s
kubernetes          10.10.10.51:6443   5d
```

- Review the content of the gluster-pv.yaml file from directory `/labfiles/k8s/Persistent_storage`, and use it to create the persistent volumes.

```
root@master1 $> cat gluster-pv.yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv0001
  labels:
         type: gluster
spec:
  capacity:
          storage: 15Gi
  accessModes:
          - ReadWriteOnce
  glusterfs:
          endpoints: "glusterfs-cluster"
          path: "/vol1"
          readOnly: false

---

kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv0002
  labels:
         type: gluster
spec:
  capacity:
          storage: 15Gi
  accessModes:
          - ReadWriteOnce
  glusterfs:
        endpoints: "glusterfs-cluster"
        path: "/vol2"
        readOnly: false

---

kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv0003
  labels:
           type: gluster
spec:
  capacity:
          storage: 15Gi
  accessModes:
          - ReadWriteOnce
  glusterfs:
          endpoints: "glusterfs-cluster"
          path: "/vol3"
          readOnly: false

root@master1 $> kubectl apply -f gluster-pv.yaml
persistentvolume/pv0001 created
persistentvolume/pv0002 created
persistentvolume/pv0003 created

root@master1 $> kubectl get pv
NAME      CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS      CLAIM //   AGE
pv0001    15Gi       RWO           Retain          Available         //   25s
pv0002    15Gi       RWO           Retain          Available         //   25s
pv0003    15Gi       RWO           Retain          Available         //   25s
```

- Create a persistent volume claim for 1G of storage capacity. Identify the persistent volume that is backing the claim

```
root@master1 $> cat gluster-pvc.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gluster-pvc1
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi

root@master1 $> kubectl apply -f gluster-pvc.yaml
persistentvolumeclaim/gluster-pvc1 created
root@master1 $> kubectl get pvc
NAME           STATUS    VOLUME    CAPACITY   ACCESSMODES   STORAGECLASS   AGE
gluster-pvc1   Bound     pv0001    15Gi       RWO                          6s
```

- Create a pod that is mounting this volume on /mnt/glusterfs1, and generate some content in it.

```
root@master1 $> cat gluster-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: glusterfs
spec:
  containers:
  - name: glusterfs
    image: nginx
    volumeMounts:
    - mountPath: "/mnt/glusterfs1"
      name: glusterfsvol1
  volumes:
  - name: glusterfsvol1
    persistentVolumeClaim:
          claimName: gluster-pvc1
```

```
root@master1 $> kubectl apply -f gluster-pod.yaml
pod/glusterfs created
root@master1 $> kubectl get pod
NAME        READY     STATUS    RESTARTS   AGE
glusterfs   1/1       Running   0          8s

root@master1 $> kubectl exec -ti glusterfs bash
root@glusterfs $> df -h /mnt/glusterfs1/
Filesystem        Size  Used Avail Use% Mounted on
10.10.10.1:/vol1   122G  13G   105G   11% /mnt/glusterfs1
root@glusterfs $> echo "Hello world!" > /mnt/glusterfs1/f1
```

- Delete the pod, the claim and the persistent volume, and verify that the content is still on the gluster share

```
root@master1 $> kubectl delete pod glusterfs
pod "glusterfs" deleted
root@master1 $> kubectl delete pvc gluster-pvc1
persistentvolumeclaim "gluster-pvc1" deleted
root@master1 $> kubectl delete pv --all
persistentvolume "pv0001" deleted
persistentvolume "pv0002" deleted
persistentvolume "pv0003" deleted
root@master1 $> exit
logout
Connection to master1 closed.
root@lab_machine $> cat /vol1/f1
Hello world!
root@lab_machine $>
```

Task 2: Set the root password for a mysql pod using Secrets

Start a MySQL server and set its root password. The official MySQL image from Docker Hub requires the *MYSQL_ROOT_PASSWORD* environmental variable to be set (see https://hub.docker.com/_/mysql/).

- Create a secret to store the value of the password.

```
root@master1 $> kubectl create secret generic mysql --from-literal=password=TopSecret
secret/mysql created
root@master1 $> kubectl get secret mysql
NAME                  TYPE                                  DATA      AGE
mysql                 Opaque                                1         13s
root@master1 $>
```

- Use this secret in a pod in order to set the value of the MYSQL_ROOT_PASSWORD environmental variable.

```
root@master1 $> cat mysql.yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql
spec:
  containers:
  - image: mysql:5.5
```

```
    name: mysqldb
    env:
    - name: MYSQL_ROOT_PASSWORD
      valueFrom:
         secretKeyRef:
               name: mysql
               key: password

root@master1 $> kubectl apply -f mysql.yaml
pod/mysql created
root@master1 $> kubectl get pod
NAME      READY     STATUS    RESTARTS   AGE
mysql     1/1       Running   0          52s
root@master1 $> kubectl exec -ti mysql bash
root@mysql $> mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.5.56 MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)

mysql> exit
Bye
root@mysql $>
```

Task 3: Use ConfigMap to pass a file to a pod

- Create file name myconf with some arbitrary content, and create a configMap named my-map

```
root@master1 $> echo "this is a demo file" > myconf
root@master1 $> kubectl create configmap my-map --from-file=myconf
configmap/my-map created
root@master1 $> kubectl get configmap my-map -o yaml
apiVersion: v1
data:
  myconf: |
    this is a demo file
```

```
kind: ConfigMap
metadata:
  creationTimestamp: "2019-07-12T15:47:53Z"
  name: my-map
  namespace: default
  resourceVersion: "152884"
  selfLink: /api/v1/namespaces/default/configmaps/my-map
  uid: 656571ea-5498-490e-acca-f63f1fa41feb

root@master1 $>
```

- Create a pod that is using the config map as a volume mounted on /config, and check the content of the /config/myconf file

```
root@master1 $> cat cmap-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: configmap-test
spec:
  containers:
  - image: busybox
    command:
     - sleep
     - "3600"
    volumeMounts:
    - mountPath: /config
      name: map
    name: busy
  volumes:
    - name: map
      configMap:
          name: my-map
root@master1 $> kubectl apply -f cmap-pod.yaml
pod/configmap-test created
root@master1 $> kubectl exec  configmap-test cat /config/myconf
this is a demo file
root@master1 $>
```

- Cleanup: Remove all the resources created in this lab and wait for resource removal.

```
root@master1 $> kubectl delete pod configmap-test
pod "configmap-test" deleted

root@master1 $> kubectl delete pod mysql
pod "mysql" deleted

root@master1 $> kubectl delete pod shvol1
pod "shvol1" deleted

root@master1 $> kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   6d
```

Lab 8: Logging, monitoring kubernetes

Task 1: Investigate the logging in kubernetes

- Create a *logtest* pod using the *nginx* image.

```
root@master1 $> kubectl run --generator=run-pod/v1 --image=nginx logtest
pod/logtest created
```

- Identify the IP Address, and the node for the pod.

```
root@master1 $> kubectl get pod -o wide
NAME       READY    STATUS     RESTARTS    AGE      IP           NODE     //
logtest    1/1      Running    0           3m29s    10.42.0.2    worker2  //
```

- Access the selected nginx instance using curl

```
root@master1 $> curl -s 10.42.0.2|grep title
<title>Welcome to nginx!</title>
root@master1 $>
```

- Check the logs of the selected pod

```
root@master1 $> kubectl logs logtest
10.32.0.1 - - [13/Jul/2019:04:45:50 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.29.0" "-"
```

- On the node where the identified pod runs, check the current system journal file, and verify that the log message is there:

```
root@master1 $> ssh worker2

root@worker2 $> docker ps|grep logtest
ae92e9259743         nginx                           "nginx -g 'daemon of..."    ...
99ffa426fec0         k8s.gcr.io/pause:3.1               "/pause"                 ...


root@worker2 $> docker logs ae92e9259743
10.32.0.1 - - [13/Jul/2019:04:45:50 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.29.0" "-"

root@worker2 $> docker inspect ae92e9259743|grep LogPath
        "LogPath": "/var/lib/docker/containers/ae92e92597433595dee7440dc2...-json.log",
root@worker2 $>cat /var/lib/docker/containers/ae92e92597433595dee7440dc244c8...-json.log
{"log":"10.32.0.1 - - [13/Jul/2019:04:45:50 +0000] \"GET / HTTP/1.1\" 200 612 \"-\" i
\"curl/7.29.0\" \"-\"\n","stream":"stdout","time":"2019-07-13T04:45:50.757613401Z"}
```

- Erase the content of the file, and check the POD's log again

```
root@worker2 $> echo > /var/lib/docker/containers/ae92e92597433595dee7440dc2...-json.log
root@worker2 $> logout
Connection to worker2 closed.
root@master1 $> kubectl logs logtest
failed to get parse function: unsupported log format: "\n"
```

Task 2: Monitoring kubernetes core metrics pipeline

- Install the metrics server

  In the `/labfiles/k8s/Logging_and_monitoring/metrics-server` directory you will find the files needed to install the metrics server:

```
root@master1 metrics-server$> ls
aggregated-metrics-reader.yaml  metrics-server-deployment.yaml
auth-delegator.yaml             metrics-server-service.yaml
auth-reader.yaml                resource-reader.yaml
metrics-apiservice.yaml
root@master1 metrics-server$> cat *yaml | kubectl apply -f -
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
serviceaccount/metrics-server created
deployment.extensions/metrics-server created
service/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
```

```
root@master1 metrics-server$> kubectl -n kube-system get pod -l k8s-app=metrics-server
NAME                                 READY     STATUS     RESTARTS    AGE
metrics-server-68df9fbc9f-qtp        1/1       Running    0           5m
root@master1 metrics-server$> kubectl -n kube-system logs metrics-server-68df9fbc9f-qtp
I0713 04:27:00.069208       1 serving.go:273] Generated self-signed cert (apiserver.loca
l.config/certificates/apiserver.crt, apiserver.local.config/certificates/apiserver.key)
[restful] 2019/07/13 04:27:00 log.go:33: [restful/swagger] listing is available at https
://:443/swaggerapi
[restful] 2019/07/13 04:27:00 log.go:33: [restful/swagger] https://:443/swaggerui/ is ma
pped to folder /swagger-ui/
I0713 04:27:00.484555       1 serve.go:96] Serving securely on [::]:443
```

List the metrics for the nodes:

```
root@master1 metrics-server$> kubectl top node
NAME       CPU(cores)   CPU%      MEMORY(bytes)   MEMORY%
master1    144m         14%       1259Mi          72%
worker1    35m          3%        362Mi           25%
worker2    38m          3%        461Mi           33%
worker3    37m          3%        465Mi           33%
root@master1 metrics-server$>
```

Task 3: Monitoring kubernetes full metrics pipeline (*OPTIONAL*)

- Start a `tunnel` instance from your desktop.

- Create a `NodePort` type service for the kubernetes dashboard.

```
root@master1 $> kubectl --namespace=kube-system get deployment
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
coredns                    2/2     2            2           40h
default-http-backend       1/1     1            1           12h
kubernetes-dashboard       1/1     1            1           39h
metrics-server             1/1     1            1           7m11s
nginx-ingress-controller   1/1     1            1           12h

root@master1 $> kubectl --namespace=kube-system expose deployment kubernetes-dashboard \
> --name=dashboard --type=NodePort
service/dashboard exposed
```

- Identify the port where the dashboard is exposed.

```
root@master1 $> kubectl --namespace=kube-system get svc dashboard
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
dashboard   NodePort   10.102.71.25   <none>        8443:30745/TCP   50m
```

Note:
As of release 1.7 Dashboard no longer has full admin privileges granted by default. For these exercises we will allow full admin priviledges to the dashboard.

- Identify the kubernetes dashboard service account (or create a service account of yourself)

```
root@master1 $> kubectl get serviceaccounts --all-namespaces | grep dashboard
kube-system       kubernetes-dashboard                 1         39h
```

- Create a ClusterRoleBindig that binds the `cluster-admin` ClusterRole to the `kubernetes-dashboard` ServiceAccount.

```
root@master1 $> cat dashboard-rbac.yaml
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: kubernetes-dashboard
  labels:
    k8s-app: kubernetes-dashboard
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: kubernetes-dashboard
  namespace: kube-system
```

```
root@master1 $> kubectl apply -f dashboard-rbac.yaml
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
```

Note:
Similar result can be obtained using the following commands:

```
root@master1 $> kubectl create clusterrolebinding --clusterrole=cluster-admin \
> --serviceaccount=kube-system:kubernetes-dashboard kubernetes-dashboard
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created

root@master1 $> kubectl label clusterrolebinding kubernetes-dashboard \
> k8s-app=kubernetes-dashboard
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard labeled
```

- Retrieve the token associated with the *kubernetes-dashboard* service account

```
root@master1 $> kubectl get serviceaccounts -n kube-system kubernetes-dashboard -o yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{},"labels":
      {"k8s-app":"kubernetes-dashboard"},"name":"kubernetes-dashboard","namespace":"kub
      e-system"}}
  creationTimestamp: "2019-07-11T14:00:22Z"
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kube-system
  resourceVersion: "7441"
  selfLink: /api/v1/namespaces/kube-system/serviceaccounts/kubernetes-dashboard
  uid: 4efef2fa-1be0-4377-bcf8-6767a33aa440
secrets:
- name: kubernetes-dashboard-token-w45j6

root@master1 $> kubectl -n kube-system get secret kubernetes-dashboard-token-w45j6 \
> -o jsonpath='{.data.token}' | base64 -d
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJ
uZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2V
ydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJrdWJlcm5ldGVzLWRhc2hib2FyZC10b2tlbi13NDVqNiIsImt1YmV
ybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJrdWJlcm5ldGVzLWRhc2hib2F
yZCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6IjRlZmVmMmZhLTF
iZTAtNDM3Ny1iY2Y4LTY3NjdhMzNhYTQ0MCIsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDprdWJlLXN5c3R
lbTprdWJlcm5ldGVzLWRhc2hib2FyZCJ9.f8vXT0y7yqLUmp5YA2J5JXA-lo6pZ-zD8exh0Ad69CFEuZ82rP7hco
eP5WicKSJanI1LX1UXVcEIgOe-uz_dLYG6oDZdDk-ge7a_8M1JvOTVI9aFbZPQwlBS358-oURj2-ZfBJYZBa4yAZ
BKZhwZpabPuJjLLPwJzvbH3TAeYba9pwXOtif32r75RyKQZxuyQhX72NHUzVW-Yd3ETrvoI7nzjqeCQmcR3eNWE1
5e5wcWsZo7V_7-z_pX4qQdHFFpMcosVa0llD3XTmLYGPUnrfYKgs5P66qMlePnV6zXoGjm0lw_YdaDjEjGCEnXKW
n86aRAGAUUKh0741bvQNdeNw root@master1 $>
```

- On your remote lab desktop open a browser and connect to the address 10.10.10.51 on the identified port. Accept the certificate. Select “Token” for the authorization, and copy/paste the identified token.

- Install *influxdb*, *grafana*, and *heapster* using the templates from the `/labfiles/k8s/Logging_and_monitoring` directory.

```
root@master1 $> ls
grafana.yaml  heapster_with_RBAC.yaml  influxdb.yaml

root@master1 $> kubectl apply -f influxdb.yaml
deployment.apps/monitoring-influxdb created
service/monitoring-influxdb created

root@master1 $> kubectl apply -f grafana.yaml
deployment.apps/monitoring-grafana created
service/monitoring-grafana created

root@master1 $> kubectl apply -f heapster_with_RBAC.yaml
serviceaccount/heapster created
deployment.apps/heapster created
service/heapster created
clusterrolebinding.rbac.authorization.k8s.io/heapster created
```

Verify the status of the pods created for influxdb, grafana, heapster:

```
root@master1 Logging_and_monitoring$> kubectl -n kube-system get pod -l task=monitoring
NAME                                  READY     STATUS    RESTARTS   AGE
heapster-848fd8b656-h8nbp             1/1       Running   0          4m
monitoring-grafana-c797777db-fbf75    1/1       Running   0          4m
monitoring-influxdb-cf9d95766-bq6bv   1/1       Running   0          4m
root@master1 Logging_and_monitoring$>

root@master1 Logging_and_monitoring$> kubectl -n kube-system logs \
> monitoring-influxdb-cf9d95766-bq6bv | tail
[I] 2019-07-13T07:11:46Z Listening on HTTP:[::]:8086 service=httpd
[I] 2019-07-13T07:11:46Z Starting retention policy enforcement service with check interv
al of 30m0s service=retention
[I] 2019-07-13T07:11:46Z Listening for signals
[I] 2019-07-13T07:11:46Z Storing statistics in database '_internal' retention policy 'mo
nitor', at interval 10s service=monitor
[httpd] 10.42.0.4 - root [13/Jul/2019:07:12:12 +0000] "GET /ping HTTP/1.1" 204 0 "-" "he
apster/v1.5.4" 897af0c3-a53d-11e9-8001-000000000000 367
[I] 2019-07-13T07:13:25Z CREATE DATABASE k8s WITH NAME "default" service=query
[httpd] 10.42.0.4 - root [13/Jul/2019:07:13:25 +0000] "GET /query?db=&q=CREATE+DATABASE+
k8s+WITH+NAME+%22default%22 HTTP/1.1" 200 193 "-" "heapster/v1.5.4" b46b08b2-a53d-11e9-8
002-000000000000 2779
[httpd] 10.42.0.4 - root [13/Jul/2019:07:13:25 +0000] "POST /write?consistency=&db=k8s&p
recision=&rp=default HTTP/1.1" 204 0 "-" "heapster/v1.5.4" b46c17e8-a53d-11e9-8003-00000
0000000 131
[httpd] 10.42.0.4 - root [13/Jul/2019:07:14:25 +0000] "POST /write?consistency=&db=k8s&p
recision=&rp=default HTTP/1.1" 204 0 "-" "heapster/v1.5.4" d82e434b-a53d-11e9-8004-00000
0000000 46
[httpd] 10.42.0.4 - root [13/Jul/2019:07:15:25 +0000] "POST /write?consistency=&db=k8s&p
recision=&rp=default HTTP/1.1" 204 0 "-" "heapster/v1.5.4" fbf19169-a53d-11e9-8005-00000
0000000 46

root@master1 Logging_and_monitoring$>

root@master1 Logging_and_monitoring$> kubectl -n kube-system logs monitoring-grafana-c797777db
t=2019-07-13T07:12:03+0000 lvl=info msg="Created default admin user: admin"
t=2019-07-13T07:12:03+0000 lvl=info msg="Starting plugin search" logger=plugins
```

```
t=2019-07-13T07:12:03+0000 lvl=warn msg="Plugin dir does not exist" logger=plugins dir=/
usr/share/grafana/data/plugins
t=2019-07-13T07:12:03+0000 lvl=info msg="Plugin dir created" logger=plugins dir=/usr/sha
re/grafana/data/plugins
t=2019-07-13T07:12:03+0000 lvl=info msg="Initializing Alerting" logger=alerting.engine
t=2019-07-13T07:12:03+0000 lvl=info msg="Initializing CleanUpService" logger=cleanup
t=2019-07-13T07:12:03+0000 lvl=info msg="Initializing Stream Manager"
t=2019-07-13T07:12:03+0000 lvl=info msg="Initializing HTTP Server" logger=http.server ad
dress=0.0.0.0:3000 protocol=http subUrl= socket=
Connected to the Grafana dashboard.
The datasource for the Grafana dashboard is now set.

root@master1 Logging_and_monitoring$>

root@master1 Logging_and_monitoring$> kubectl -n kube-system logs heapster-848fd8b656-h8nbp |
E0713 07:36:05.003780    1 manager.go:101] Error in scraping containers from kubelet_sum
mary:10.10.10.52:10250: Get https://10.10.10.52:10250/stats/summary/: x509: cannot valid
ate certificate for 10.10.10.52 because it doesn't contain any IP SANs
E0713 07:36:05.018202    1 manager.go:101] Error in scraping containers from kubelet_sum
mary:10.10.10.53:10250: Get https://10.10.10.53:10250/stats/summary/: x509: cannot valid
ate certificate for 10.10.10.53 because it doesn't contain any IP SANs
E0713 07:36:05.021955    1 manager.go:101] Error in scraping containers from kubelet_sum
mary:10.10.10.51:10250: Get https://10.10.10.51:10250/stats/summary/: x509: cannot valid
ate certificate for 10.10.10.51 because it doesn't contain any IP SANs
E0713 07:36:05.029948    1 manager.go:101] Error in scraping containers from kubelet_sum
mary:10.10.10.54:10250: Get https://10.10.10.54:10250/stats/summary/: x509: cannot valid
ate certificate for 10.10.10.54 because it doesn't contain any IP SANs
W0713 07:36:25.000376    1 manager.go:152] Failed to get all responses in time (got 0/4)
E0713 07:37:05.005035    1 manager.go:101] Error in scraping containers from kubelet_sum
mary:10.10.10.51:10250: Get https://10.10.10.51:10250/stats/summary/: x509: cannot valid
ate certificate for 10.10.10.51 because it doesn't contain any IP SANs
E0713 07:37:05.016443    1 manager.go:101] Error in scraping containers from kubelet_sum
mary:10.10.10.53:10250: Get https://10.10.10.53:10250/stats/summary/: x509: cannot valid
ate certificate for 10.10.10.53 because it doesn't contain any IP SANs
E0713 07:37:05.019810    1 manager.go:101] Error in scraping containers from kubelet_sum
mary:10.10.10.52:10250: Get https://10.10.10.52:10250/stats/summary/: x509: cannot valid
ate certificate for 10.10.10.52 because it doesn't contain any IP SANs
E0713 07:37:05.034432    1 manager.go:101] Error in scraping containers from kubelet_sum
mary:10.10.10.54:10250: Get https://10.10.10.54:10250/stats/summary/: x509: cannot valid
ate certificate for 10.10.10.54 because it doesn't contain any IP SANs
W0713 07:37:25.000723    1 manager.go:152] Failed to get all responses in time (got 0/4)

root@master1 Logging_and_monitoring$>
```

Note:
Heapster is complaining because the `kubeadm` installer has not set up IP SANs for the TLS certificates configured for the kubelets. The Subject Alternative Name field lets you specify additional host names (sites, IP addresses, common names, etc.) to be protected by a single SSL Certificate.

Create new certificates for the kubelets, and include the IPs as Subject Alternative Names. To simplify things we will use the `cfssl` and `cfssljson` tools.

```
root@master1 Logging_and_monitoring$> mkdir -p certs&& cd certs
root@master1 certs$> wget -q --timestamping  \
>  https://pkg.cfssl.org/R1.2/cfssl_linux-amd64  \
>  https://pkg.cfssl.org/R1.2/cfssljson_linux-amd64
root@master1 certs$> chmod +x cfssl_linux-amd64 cfssljson_linux-amd64
root@master1 certs$> mv cfssl_linux-amd64 /usr/local/bin/cfssl
root@master1 certs$> mv cfssljson_linux-amd64 /usr/local/bin/cfssljson
root@master1 certs$> cfssl version
Version: 1.2.0
Revision: dev
Runtime: go1.6
root@master1 certs$>
```

Generate a configuration file for the CA. We will use the CA key and CA certificate created at the install of Kubernetes.

```
root@master1 certs$> cat ca-config.json
{
  "signing": {
    "default": {
      "expiry": "8760h"
    },
    "profiles": {
      "kubernetes": {
        "usages": ["signing", "key encipherment", "server auth", "client auth"],
        "expiry": "8760h"
      }
    }
  }
}
```

Create the certificate signing request configuration files for the kubelets(you can find this code in `/labfiles/k8s/Logging_and_monitoring/certs/mkcsr.sh`):

Note:
If you plan to copy-paste the following code snippets please pay attention to the correct line endings! The snippets presented below are provided to you in the /labfiles directory structure.

```
for instance in master1 worker1 worker2 worker3; do
cat > ${instance}-csr.json <<EOF
{
  "CN": "system:node:${instance}",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
     "C": "HU",
     "L": "Budapest",
     "O": "system:nodes",
     "OU": "KBS-10x lab",
     "ST": "Budapest"
    }
  ]
}
EOF
done
```

Generate the certificates for the kubelets (you can find this code in `/labfiles/k8s/Logging_and_monitoring/certs/mkcert.sh`):

```
for instance in master1 worker1 worker2 worker3
do
  IP=$(grep ${instance} /etc/hosts | awk '{print $1}')
  cfssl gencert   -ca=/etc/kubernetes/pki/ca.crt \
     -ca-key=/etc/kubernetes/pki/ca.key  \
     -config=ca-config.json   \
     -hostname=${instance},${IP}   \
     -profile=kubernetes   \
     ${instance}-csr.json | cfssljson -bare ${instance}
done
```

Verify the created files, and make sure that the certificates contains the IPs as SANs:

```
root@master1 certs$> ls -ltrh
total 24K
-rwxr-xr-x 1 root root  373 Oct  5 18:14 mkcert.sh
-rwxr-xr-x 1 root root  339 Oct  5 18:19 mkcsr.sh
-rw-r--r-- 1 root root  233 Oct  5 18:23 ca-config.json
-rw-r--r-- 1 root root  228 Oct  5  2018 worker3-csr.json
-rw-r--r-- 1 root root  228 Oct  5  2018 worker2-csr.json
-rw-r--r-- 1 root root  228 Oct  5  2018 worker1-csr.json
-rw-r--r-- 1 root root  228 Oct  5  2018 master1-csr.json
-rw-r--r-- 1 root root 1.3K Oct  5  2018 master1.pem
-rw------- 1 root root 1.7K Oct  5  2018 master1-key.pem
-rw-r--r-- 1 root root 1.1K Oct  5  2018 master1.csr
-rw-r--r-- 1 root root 1.3K Oct  5  2018 worker3.pem
-rw------- 1 root root 1.7K Oct  5  2018 worker3-key.pem
-rw-r--r-- 1 root root 1.1K Oct  5  2018 worker3.csr
-rw-r--r-- 1 root root 1.3K Oct  5  2018 worker2.pem
-rw------- 1 root root 1.7K Oct  5  2018 worker2-key.pem
-rw-r--r-- 1 root root 1.1K Oct  5  2018 worker2.csr
-rw-r--r-- 1 root root 1.3K Oct  5  2018 worker1.pem
-rw------- 1 root root 1.7K Oct  5  2018 worker1-key.pem
-rw-r--r-- 1 root root 1.1K Oct  5  2018 worker1.csr
root@master1 certs$>

root@master1 certs$> openssl x509 -noout -text -in master1.pem | grep -A1 Alternative
            X509v3 Subject Alternative Name:
                DNS:master1, IP Address:10.10.10.51
root@master1 certs$>
```

Copy the generated certificates and keys to the appropriate locations on the respective nodes, and restart the `kubelet` service on the nodes.

```
root@master1 certs$> for instance in master1 worker1 worker2 worker3
> do
> scp ${instance}.pem ${instance}:/var/lib/kubelet/pki/kubelet.crt
> scp ${instance}-key.pem ${instance}:/var/lib/kubelet/pki/kubelet.key
> ssh ${instance} systemctl restart kubelet
> done
master1.pem                                                100% 1310     3.6MB/s   00:00
master1-key.pem                                            100% 1679     2.7MB/s   00:00
worker1.pem                                                100% 1310   302.2KB/s   00:00
worker1-key.pem                                            100% 1679   819.4KB/s   00:00
worker2.pem                                                100% 1310   178.4KB/s   00:00
worker2-key.pem                                            100% 1679     3.5MB/s   00:00
worker3.pem                                                100% 1310   324.8KB/s   00:00
worker3-key.pem                                            100% 1675     3.0MB/s   00:00
root@master1 certs$>
```

Verify whether the kubelets are using the new certificates:

```
root@master1 certs$> openssl s_client -connect worker2:10250| openssl x509 -noout -text
...
            X509v3 Subject Alternative Name:
                DNS:worker2, IP Address:10.10.10.53
...
root@master1 certs$>
```

Verify whether the heapster pod can collect the metrics now:

```
root@master1 certs$> kubectl -n kube-system logs heapster-848fd8b656-h8nbp | tail -3
E0713 08:12:05.009433   1 manager.go:101] Error in scraping containers from kubelet_summ
ary:10.10.10.51:10250: request failed - "403 Forbidden", response: "Forbidden (user=syst
em:serviceaccount:kube-system:heapster, verb=get, resource=nodes, subresource=stats)"
E0713 08:12:05.010127   1 manager.go:101] Error in scraping containers from kubelet_summ
ary:10.10.10.53:10250: request failed - "403 Forbidden", response: "Forbidden (user=syst
em:serviceaccount:kube-system:heapster, verb=get, resource=nodes, subresource=stats)"
E0713 08:12:05.014379   1 manager.go:101] Error in scraping containers from kubelet_summ
ary:10.10.10.52:10250: request failed - "403 Forbidden", response: "Forbidden (user=syst
em:serviceaccount:kube-system:heapster, verb=get, resource=nodes, subresource=stats)"

root@master1 certs$>
```

The `system:heapster` ClusterRole associated with the ServiceAccount used by heapster is not allowed to retrieve the `stats` subresource of the `node` resources. We need to adjust the role to allow read access to the stats.

```
root@master1 ~$> kubectl edit clusterrole system:heapster

# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  creationTimestamp: 2018-07-10T13:18:59Z
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: system:heapster
  resourceVersion: "62"
  selfLink: /apis/rbac.authorization.k8s.io/v1/clusterroles/system%3Aheapster
  uid: ce68fba7-8443-11e8-b8b7-080027000101
rules:
- apiGroups:
  - ""
  resources:
  - events
  - namespaces
  - nodes
  - nodes/stats
  - pods
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - extensions
  resources:
  - deployments
  verbs:
  - get
  - list
  - watch
```

Wait a minute and verify whether there are new error messages logged by the heapster pod:

```
root@master1 ~$> sleep 60; date; kubectl -n kube-system logs \
> heapster-848fd8b656-h8nbp | tail -1
Sat Jul 13 08:19:03 UTC 2019
W0713 08:17:25.000346       1 manager.go:152] Failed to get all responses in time (got 0/4)

root@master1 ~$>
```

No new error messages were logged, so we should be able to visualize the data collected by heapster.

- Identify the port where the grafana is exposed on the nodes.

```
root@master1 $> kubectl --namespace=kube-system get svc monitoring-grafana
NAME                 CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
monitoring-grafana   10.96.150.153   <nodes>       80:31520/TCP   15m
```

- Open a new tab in the browser on your remote desktop and connect to the 10.10.10.51 on the identified port. The grafana interface should appear. Here in the **Home** menu select “Pods”, then select the “kube-system” namespace and the monitoring-grafana POD.

- Reload several times the dashboard and grafana tabs in your browser and observe how the graphs are evolving.

- In your terminal list the resource usage of the pods running in the *kube-system* namespace using the *kubectl top* command.

```
root@master1 $> kubectl top pod --namespace=kube-system
NAME                                    CPU(cores)    MEMORY(bytes)
monitoring-grafana-973508798-h0815      0m            11Mi
kube-controller-manager-master1         8m            44Mi
...
root@master1 $>
```

Lab 9: Installing Kubernetes

Task 1: Preparing system for installation

Before start of installation, you have to reset you nodes to initial, uninstalled state.

- Reset nodes by /labfiles/os_nodes script.

```
root@lab_machine $> /labfiles/os_nodes reset_nodes
Domain master1 destroyed

Domain worker1 destroyed

Domain worker2 destroyed

Domain worker3 destroyed

Domain master1 started

Domain worker1 started

Domain worker2 started

Domain worker3 started
```

- Wait a few seconds, and verify whether nodes are in running state

```
root@lab_machine $> /labfiles/os_nodes status
master1      running
worker1      running
worker2      running
worker3      running
node1   shutoff
```

Task 2: Prerequisites - to be verified

Based on kubernetes.io (https://goo.gl/ZhpXGT) you can check a list of prerequisites.

- Verify memory allocation of your nodes

```
root@lab_machine $> echo worker1 worker2 worker3 master1 > nodes
root@lab_machine $> for i in `cat nodes`; do virsh dominfo $i|grep -E 'Name|memory';done
Name:           worker1
Max memory:     1572864 KiB
Used memory:    1572864 KiB
Name:           worker2
Max memory:     1572864 KiB
Used memory:    1572864 KiB
Name:           worker3
Max memory:     1572864 KiB
Used memory:    1572864 KiB
Name:           master1
Max memory:     2097152 KiB
Used memory:    2097152 KiB
```

- Verify the uniqueness of MAC addresses and product UUID-s of nodes.

```
root@lab_machine $> for i in `cat nodes`; do echo "${i}";virsh domiflist $i;done
worker1
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet2      bridge     br_management virtio      08:00:27:00:02:01

worker2
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet1      bridge     br_management virtio      08:00:27:00:03:01

worker3
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet0      bridge     br_management virtio      08:00:27:00:04:01

master1
Interface  Type       Source     Model       MAC
-------------------------------------------------------
vnet3      bridge     br_management virtio      08:00:27:00:01:01

root@lab_machine $> for i in `cat nodes`; do echo "${i}";\
>ssh $i 'cat /sys/class/dmi/id/product_uuid';done
worker1
165F0FC7-5A59-4591-ABB3-806BDDC8B16F
worker2
8B4C43E5-3609-410A-8DBA-EADA78892008
worker3
6AB36F8E-C75A-4620-A6DD-247747B2B1B3
master1
CE6394EC-85E1-4230-A332-E5D20F5E7A97
```

- The swapping has to be disabled, in order to kubelet to work fine.

```
root@lab_machine $> for i in `cat nodes`; do echo "${i}";ssh $i 'swapon -s';done
worker1
worker2
worker3
master1
```

- Verify whether all necessary input ports are open.

```
root@lab_machine $> for i in `cat nodes`; do echo "${i}";ssh $i 'iptables -nvL INPUT';done
worker1
Chain INPUT (policy ACCEPT 41 packets, 8214 bytes)
pkts bytes target     prot opt in     out     source               destination
worker2
Chain INPUT (policy ACCEPT 41 packets, 8214 bytes)
pkts bytes target     prot opt in     out     source               destination
worker3
Chain INPUT (policy ACCEPT 40 packets, 8162 bytes)
pkts bytes target     prot opt in     out     source               destination
master1
Chain INPUT (policy ACCEPT 40 packets, 8162 bytes)
pkts bytes target     prot opt in     out     source               destination
```

Task 3: Installing Docker

Note:
You are able to install maintened version of Docker from CentOS Extras repo, or community "CE" version from docker.com. In this course we are using CE edition.

- Install repo files on all nodes

```
root@lab_machine $> for i in `cat nodes`; do ssh $i \
>'yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo'
> done
Loaded plugins: fastestmirror, priorities, remove-with-leaves
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos
.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
Loaded plugins: fastestmirror, priorities, remove-with-leaves
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos
.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
Loaded plugins: fastestmirror, priorities, remove-with-leaves
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos
.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
Loaded plugins: fastestmirror, priorities, remove-with-leaves
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos
```

```
.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
```

- Install docker-ce package, enable and start docker service

```
root@lab_machine $> for i in `cat nodes`
> do ssh $i 'yum install --setopt=obsoletes=0 -y docker-ce-18.09.7-3.el7 &&\
> systemctl enable docker && systemctl start docker';done
Loaded plugins: fastestmirror, priorities, remove-with-leaves
Loading mirror speeds from cached hostfile
* base: centos.lonyai.com
* epel: mirror.pmf.kg.ac.rs
* extras: centos.lonyai.com
* updates: centos.lonyai.com
Resolving Dependencies
--> Running transaction check
---> Package docker-ce.x86_64 3:18.09.7-3.el7 will be installed
--> Processing Dependency: container-selinux >= 2.9 for package: 3:docker-ce-18.09.7-3.
el7.x86_64
--> Processing Dependency: containerd.io >= 1.2.2-3 for package: 3:docker-ce-18.09.7-3.
el7.x86_64
--> Processing Dependency: docker-ce-cli for package: 3:docker-ce-18.09.7-3.el7.x86_64
--> Running transaction check
---> Package container-selinux.noarch 2:2.99-1.el7_6 will be installed
---> Package containerd.io.x86_64 0:1.2.6-3.3.el7 will be installed
---> Package docker-ce-cli.x86_64 1:18.09.7-3.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
...

Dependency Installed:
  container-selinux.noarch 2:2.99-1.el7_6  containerd.io.x86_64 0:1.2.6-3.3.el7
  docker-ce-cli.x86_64 1:18.09.7-3.el7

Complete!
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr
/lib/systemd/system/docker.service.
Loaded plugins: fastestmirror, priorities, remove-with-leaves
...
```

Task 4: Install kubelet, kubeadm, kubectl

- Create the install_kubeadm.sh file with the content similar to the following example and make it executable:

```
root@lab_machine $> cat install_kubeadm.sh
#!/bin/bash

cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
```

```
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg \
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
setenforce 0
yum install -y kubelet kubeadm kubectl
systemctl enable --now kubelet

root@lab_machine $>
root@lab_machine $> chmod +x install_kubeadm.sh
```

- Copy the script to the nodes, and execute it there:

```
root@lab_machine $> for i in `cat nodes`; do echo "========== $i ========" ;\
> scp install_kubeadm.sh ${i}: ; ssh $i './install_kubeadm.sh'; echo ; done
========== worker1 ========
install_kubeadm.sh                              100%  429   928.4KB/s   00:00
setenforce: SELinux is disabled
Loaded plugins: fastestmirror, priorities, remove-with-leaves
Loading mirror speeds from cached hostfile
* base: centos.lonyai.com
* epel: mirror.pmf.kg.ac.rs
* extras: centos.lonyai.com
* updates: centos.lonyai.com
Retrieving key from https://packages.cloud.google.com/yum/doc/yum-key.gpg
Importing GPG key 0xA7317B0F:
Userid     : "Google Cloud Packages Automatic Signing Key <gc-team@google.com>"
Fingerprint: d0bc 747f d8ca f711 7500 d6fa 3746 c208 a731 7b0f
From       : https://packages.cloud.google.com/yum/doc/yum-key.gpg
Retrieving key from https://packages.cloud.google.com/yum/doc/yum-key.gpg
Importing GPG key 0xA7317B0F:
 Userid     : "Google Cloud Packages Automatic Signing Key <gc-team@google.com>"
 Fingerprint: d0bc 747f d8ca f711 7500 d6fa 3746 c208 a731 7b0f
 From       : https://packages.cloud.google.com/yum/doc/yum-key.gpg
Retrieving key from https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
Resolving Dependencies
--> Running transaction check
---> Package kubeadm.x86_64 0:1.15.0-0 will be installed
--> Processing Dependency: kubernetes-cni >= 0.7.5 for package: kubeadm-1.15.0-0.x86_64
--> Processing Dependency: cri-tools >= 1.11.0 for package: kubeadm-1.15.0-0.x86_64
---> Package kubectl.x86_64 0:1.15.0-0 will be installed
---> Package kubelet.x86_64 0:1.15.0-0 will be installed
--> Processing Dependency: socat for package: kubelet-1.15.0-0.x86_64
--> Running transaction check
---> Package cri-tools.x86_64 0:1.13.0-0 will be installed
---> Package kubernetes-cni.x86_64 0:0.7.5-0 will be installed
---> Package socat.x86_64 0:1.7.3.2-2.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
...

Complete!
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to
/etc/systemd/system/kubelet.service.

========== worker2 ========
...
```

- Verify kubelet service and selinux state on all nodes

```
root@lab_machine $> for i in `cat nodes`; do echo "========== $i ========"
> ssh $i 'systemctl status kubelet'; echo ; done
========== worker1 ========
kubelet.service - kubelet: The Kubernetes Node Agent
Loaded: loaded (/etc/systemd/system/kubelet.service; enabled; vendor preset: disabled)
Drop-In: /etc/systemd/system/kubelet.service.d
                 10-kubeadm.conf
Active: activating (auto-restart) (Result: exit-code) since Tue 2018-07-10 13:15:00 UTC;
5s ago
 Docs: http://kubernetes.io/docs/
Process: 1713 ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS
$KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS (code=exited, status=255)
Main PID: 1713 (code=exited, status=255)

Jul 10 13:15:00 worker1 systemd[1]: kubelet.service: main process exited, code=exited,
status=255/n/a
Jul 10 13:15:00 worker1 systemd[1]: Unit kubelet.service entered failed state.
Jul 10 13:15:00 worker1 systemd[1]: kubelet.service failed.

========== worker2 ========
...

root@lab_machine $> for i in `cat nodes`; do echo "========== $i ========"
> ssh $i 'getenforce'; echo ; done
========== worker1 ========
Disabled

========== worker2 ========
Disabled

========== worker3 ========
Disabled

========== master1 ========
Disabled
```

Task 5: Initialize the master node

- Run *kubeadm init* command on *master1* node to initialize controll plane.

```
root@master1 $> kubeadm init
[init] Using Kubernetes version: v1.15.0
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver.
        The recommended driver is "systemd". Please follow the guide at https://kubernet
        es.io/docs/setup/cri/
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet con
nection
[preflight] You can also perform this action in beforehand using 'kubeadm config images
```

```
pull'
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/ku
beadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Activating the kubelet service

...

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/

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

kubeadm join 10.10.10.51:6443 --token sqdort.gaaxhc1hj1x87pov --discovery-token-ca-cert-
hash sha256:715051ab516ba90d01ed2e35cabc0f67f717778a20c8b2c95eccd77cdf57db44

root@master1 $> mkdir -p $HOME/.kube
root@master1 $> cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@master1 $> chown $(id -u):$(id -g) $HOME/.kube/config
root@master1 $>
```

- At this point the control plane is ready

```
root@master1 $> kubectl get all --namespace=kube-system
NAME                                  READY     STATUS    RESTARTS   AGE
pod/coredns-78fcdf6894-pplls          0/1       Pending   0          18h
pod/coredns-78fcdf6894-r4qpl          0/1       Pending   0          18h
pod/etcd-master1                      1/1       Running   0          18h
pod/kube-apiserver-master1            1/1       Running   0          18h
pod/kube-controller-manager-master1   1/1       Running   0          18h
pod/kube-proxy-5mm7p                  1/1       Running   0          3m
pod/kube-proxy-jjr28                  1/1       Running   0          18h
pod/kube-scheduler-master1            1/1       Running   0          18h

NAME               TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)         AGE
service/kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP   18h

NAME                        DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   ...
daemonset.apps/kube-proxy   2         2         2         2            2           ...

NAME                      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/coredns   2         2         2            0           18h

NAME                                 DESIRED   CURRENT   READY     AGE
replicaset.apps/coredns-78fcdf6894   2         2         0         18h
```

- Install the network add-on

```
root@master1 $> kubectl apply -f \
>"https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.extensions/weave-net created
root@master1 $>
```

Task 6: Join the worker nodes to the cluster

- On each worker node run *kubeadm join* command copied from end of *kubeadm init* output.

```
root@worker1 $> kubeadm join 10.10.10.51:6443 --token USE_THE_OUTPUT_OF_kubeadm_init \
>--discovery-token-ca-cert-hash sha256:USE_THE_OUTPUT_OF_kubeadm_init
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The r
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.15" Confi
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-
[kubelet-start] Activating the kubelet service
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

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

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
root@worker1 $>

root@worker2 $> kubeadm join 10.10.10.51:6443 --token USE_THE_OUTPUT_OF_kubeadm_init \
>--discovery-token-ca-cert-hash sha256:USE_THE_OUTPUT_OF_kubeadm_init
....

root@worker3 $> kubeadm join 10.10.10.51:6443 --token USE_THE_OUTPUT_OF_kubeadm_init \
>--discovery-token-ca-cert-hash sha256:USE_THE_OUTPUT_OF_kubeadm_init
...
```

- Verify node version, pods and daemonsets

```
root@master1 $> kubectl get nodes
NAME       STATUS   ROLES    AGE     VERSION
master1    Ready    master   50m     v1.15.0
worker1    Ready    <none>   14m     v1.15.0
worker2    Ready    <none>   3m49s   v1.15.0
worker3    Ready    <none>   3m32s   v1.15.0

root@master1 $> kubectl get -n kube-system pod,ds -o wide
NAME                                      READY   STATUS   ... IP          NODE     ...
pod/coredns-5c98db65d4-82wf9              1/1     Running  ... 10.32.0.2   master1  ...
pod/coredns-5c98db65d4-n694k              1/1     Running  ... 10.32.0.3   master1  ...
```

```
pod/etcd-master1                      1/1     Running ... 10.10.10.51  master1 ...
pod/kube-apiserver-master1            1/1     Running ... 10.10.10.51  master1 ...
pod/kube-controller-manager-master1   1/1     Running ... 10.10.10.51  master1 ...
pod/kube-proxy-6qt7c                  1/1     Running ... 10.10.10.54  worker3 ...
pod/kube-proxy-fflsq                  1/1     Running ... 10.10.10.51  master1 ...
pod/kube-proxy-n5lj2                  1/1     Running ... 10.10.10.53  worker2 ...
pod/kube-proxy-x4hp5                  1/1     Running ... 10.10.10.52  worker1 ...
pod/kube-scheduler-master1            1/1     Running ... 10.10.10.51  master1 ...
pod/weave-net-jd9km                   2/2     Running ... 10.10.10.51  master1 ...
pod/weave-net-lrkcp                   2/2     Running ... 10.10.10.54  worker3 ...
pod/weave-net-m8jz6                   2/2     Running ... 10.10.10.52  worker1 ...
pod/weave-net-mbb45                   2/2     Running ... 10.10.10.53  worker2 ...

NAME                              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE ...
daemonset.extensions/kube-proxy   4         4         4       4            4         ...
daemonset.extensions/weave-net    4         4         4       4            4         ...
```

Task 7: Install dashboard

- Install *kubernetes-dashboard*, and check it.

```
root@master1 $> kubectl apply -f https://raw.githubusercontent.com/kubernetes\
>/dashboard/v1.10.1/src/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
root@master1 $>

root@master1 $> kubectl get -n kube-system pod,deployment,svc
NAME                                        READY   STATUS    RESTARTS   AGE
pod/coredns-5c98db65d4-82wf9                1/1     Running   0          88m
pod/coredns-5c98db65d4-n694k                1/1     Running   0          88m
pod/etcd-master1                            1/1     Running   0          87m
pod/kube-apiserver-master1                  1/1     Running   0          87m
pod/kube-controller-manager-master1         1/1     Running   0          87m
pod/kube-proxy-6qt7c                        1/1     Running   0          41m
pod/kube-proxy-fflsq                        1/1     Running   0          88m
pod/kube-proxy-n5lj2                        1/1     Running   0          41m
pod/kube-proxy-x4hp5                        1/1     Running   0          52m
pod/kube-scheduler-master1                  1/1     Running   0          87m
pod/kubernetes-dashboard-7d75c474bb-d8j9j   1/1     Running   0          6m30s
pod/weave-net-jd9km                         2/2     Running   0          56m
pod/weave-net-lrkcp                         2/2     Running   0          41m
pod/weave-net-m8jz6                         2/2     Running   0          52m
pod/weave-net-mbb45                         2/2     Running   1          41m

NAME                                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/coredns                2/2     2            2           88m
deployment.extensions/kubernetes-dashboard   1/1     1            1           6m30s

NAME                           TYPE        CLUSTER-IP     EXTERNAL-IP  PORT(S)      ...
```

```
service/kube-dns               ClusterIP   10.96.0.10     <none>       53/UDP,53/TCP...
service/kubernetes-dashboard   ClusterIP   10.106.16.135  <none>       443/TCP   ...
```

Lab 10: Introduction to Helm

There are no exercises for this module.

Lab 11: Install Helm and Tiller

Task 1: Install Helm

- Install the 2.10 version of the Helm client on your *master1* node.

  Download the desired release of helm:

```
root@master1 $> wget https://storage.googleapis.com/kubernetes-helm/helm-v2.10.0\
> -linux-amd64.tar.gz
--2018-09-11 14:59:31--  https://storage.googleapis.com/kubernetes-helm/helm-v2.
10.0-linux-amd64.tar.gz
Resolving storage.googleapis.com (storage.googleapis.com)... 172.217.21.208, 2a0
0:1450:4001:818::2010
Connecting to storage.googleapis.com (storage.googleapis.com)|172.217.21.208|:44
3... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9207549 (8.8M) [application/x-tar]
Saving to: ‘helm-v2.10.0-linux-amd64.tar.gz’

100%[======================================>] 9,207,549   55.0MB/s   in 0.2s

2018-09-11 14:59:31 (55.0 MB/s) - ‘helm-v2.10.0-linux-amd64.tar.gz’ saved [920
7549/9207549]

root@master1 $>
```

  Uncompress the archive:

```
root@master1 $> tar -zxvf helm-v2.10.0-linux-amd64.tar.gz
linux-amd64/
linux-amd64/LICENSE
linux-amd64/helm
linux-amd64/README.md
root@master1 $>
```

  Move the binary into a directory that is in your *$PATH* and check that it’s accessible.

```
root@master1 $> mv linux-amd64/helm /usr/local/bin/
root@master1 $> helm --help
The Kubernetes package manager

To begin working with Helm, run the 'helm init' command:
       helm init

This will install Tiller to your running Kubernetes cluster.
It will also set up any necessary local configuration.
....
```

Task 2: Install Tiller with default parameters.

- Install Tiller using the *helm init* command.

```
root@master1 $> helm init
Creating /root/.helm
Creating /root/.helm/repository
Creating /root/.helm/repository/cache
Creating /root/.helm/repository/local
Creating /root/.helm/plugins
Creating /root/.helm/starters
Creating /root/.helm/cache/archive
Creating /root/.helm/repository/repositories.yaml
Adding stable repo with URL: https://kubernetes-charts.storage.googleapis.com
Adding local repo with URL: http://127.0.0.1:8879/charts
$HELM_HOME has been configured at /root/.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.

Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated
users' policy.
To prevent this, run `helm init` with the --tiller-tls-verify flag.
For more information on securing your installation see: https://docs.helm.sh/using_helm/
#securing-your-helm-installation
Happy Helming!
root@master1 $>
```

- Check Tiller version

```
root@master1 $>helm version
Client: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
root@master1 $>
```

- Identify the IP and port where Tiller is listening

```
root@master1 $> kubectl -n kube-system get svc
NAME                   TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)         AGE
kube-dns               ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP   63d
kubernetes-dashboard   ClusterIP   10.96.192.170   <none>        443/TCP         62d
tiller-deploy          ClusterIP   10.99.216.58    <none>        44134/TCP       4m

root@master1 $> kubectl -n kube-system get endpoints tiller-deploy
NAME            ENDPOINTS         AGE
tiller-deploy   10.46.0.1:44134   4m
```

- Try to connect directly to the Tiller pod and list the installed charts

```
root@master1 $> helm --host 10.46.0.1:44134 version
Client: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}

root@master1 $>  helm --host 10.46.0.1:44134 list
Error: configmaps is forbidden: User "system:serviceaccount:kube-system:default" cannot
list configmaps in the namespace "kube-system"
```

Note:
Tiller cannot access the API using the default service account in a kubernetes cluster that uses RBAC. We need to configure the proper RBAC settings for Tiller.

Task 3: Install a secured Tiller

- Remove the existing Tiller

  Remove the Tiller service and deployment from kubernetes:

```
root@master1 $> kubectl -n kube-system delete svc tiller-deploy
service "tiller-deploy" deleted

root@master1 $> kubectl -n kube-system delete deployment tiller-deploy
deployment.extensions "tiller-deploy" deleted
```

- Install a secured Tiller that is using TLS for authentication and the *cluster-admin* ClusterRole

  Create the certificate authority and the certificates for Helm and Tiller

```
root@master1 $> openssl genrsa -out ./ca.key.pem 4096
Generating RSA private key, 4096 bit long modulus
.............++
................................++
e is 65537 (0x10001)
root@master1 $> openssl req -key ca.key.pem -new -x509 -days 3650 -sha256 \
>  -out ca.cert.pem -extensions v3_ca
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:HU
State or Province Name (full name) []:Budapest
Locality Name (eg, city) [Default City]:Budapest
Organization Name (eg, company) [Default Company Ltd]:Componentsoft
Organizational Unit Name (eg, section) []:IT
```

```
Common Name (eg, your name or your server's hostname) []:
Email Address []:
root@master1 $>
root@master1 $> openssl genrsa -out ./helm.key.pem 4096
Generating RSA private key, 4096 bit long modulus
...................++
...................................++
e is 65537 (0x10001)
root@master1 $> openssl req -key helm.key.pem -new -sha256 -out helm.csr.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:HU
State or Province Name (full name) []:Budapest
Locality Name (eg, city) [Default City]:Budapest
Organization Name (eg, company) [Default Company Ltd]:Componentsoft
Organizational Unit Name (eg, section) []:IT
Common Name (eg, your name or your server's hostname) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
root@master1 $> openssl x509 -req -CA ca.cert.pem -CAkey ca.key.pem -CAcreateserial \
> -in helm.csr.pem -out helm.cert.pem -days 730
Signature ok
subject=/C=HU/ST=Budapest/L=Budapest/O=Componentsoft/OU=IT
Getting CA Private Key
root@master1 $>
root@master1 $> openssl genrsa -out ./tiller.key.pem 4096
Generating RSA private key, 4096 bit long modulus
.....................................................................................
......................................................++
................................................++
e is 65537 (0x10001)
root@master1 $> openssl req -key tiller.key.pem -new -sha256 -out tiller.csr.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:HU
State or Province Name (full name) []:Budapest
Locality Name (eg, city) [Default City]:Budapest
Organization Name (eg, company) [Default Company Ltd]:Componentsoft
Organizational Unit Name (eg, section) []:IT
Common Name (eg, your name or your server's hostname) []:
Email Address []:

Please enter the following 'extra' attributes
```

```
to be sent with your certificate request
A challenge password []:
An optional company name []:
root@master1 $> openssl x509 -req -CA ca.cert.pem -CAkey ca.key.pem -CAcreateserial \
> -in tiller.csr.pem -out tiller.cert.pem -days 730
Signature ok
subject=/C=HU/ST=Budapest/L=Budapest/O=Componentsoft/OU=IT
Getting CA Private Key
root@master1 $>
```

Create the service account *tiller*, and a cluster role binding that links the *cluster-admin* cluster role to the *tiller* service account.

```
root@master1 $> kubectl create serviceaccount --namespace kube-system tiller
serviceaccount/tiller created
root@master1 $> kubectl create clusterrolebinding tiller-binding \
> --clusterrole=cluster-admin --serviceaccount=kube-system:tiller
clusterrolebinding.rbac.authorization.k8s.io/tiller-binding created
root@master1 $>
```

Initialize the Tiller server using the afore created certificates and service account:

```
root@master1 $> helm init --tiller-tls --tiller-tls-cert tiller.cert.pem \
>  --tiller-tls-key tiller.key.pem --tiller-tls-verify --tls-ca-cert ca.cert.pem \
>  --service-account tiller
Creating /root/.helm
Creating /root/.helm/repository
Creating /root/.helm/repository/cache
Creating /root/.helm/repository/local
Creating /root/.helm/plugins
Creating /root/.helm/starters
Creating /root/.helm/cache/archive
Creating /root/.helm/repository/repositories.yaml
Adding stable repo with URL: https://kubernetes-charts.storage.googleapis.com
Adding local repo with URL: http://127.0.0.1:8879/charts
$HELM_HOME has been configured at /root/.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.
Happy Helming!
root@master1 $>
```

Look for the tiller related resources in the Kubernetes cluster:

```
root@master1 $> kubectl -n kube-system get svc tiller-deploy
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
tiller-deploy   ClusterIP   10.102.213.225   <none>        44134/TCP   13m
root@master1 $> kubectl -n kube-system get deployments tiller-deploy
NAME            DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
tiller-deploy   1         1         1            1           13m
root@master1 $> kubectl -n kube-system get secret tiller-secret
NAME            TYPE      DATA      AGE
tiller-secret   Opaque    3         14m
root@master1 $>
root@master1 $> kubectl describe secret -n kube-system tiller-secret
Name:         tiller-secret
Namespace:    kube-system
```

```
Labels:         app=helm
                name=tiller
Annotations:    <none>

Type:  Opaque

Data
====
tls.key:  3243 bytes
ca.crt:   1972 bytes
tls.crt:  1854 bytes
root@master1 $>
```

Note:
The *tiller-secret* is a Kubernetes secret that contains three objects: the CA certficate, the Tiller private key and the Tiller certificate. These are used by Tiller to create secure connections with the clients, and authenticate them.

- Verify that Helm can communicate to Tiller using a secure connection

```
root@master1 $> helm --tls --tls-ca-cert ca.cert.pem --tls-cert helm.cert.pem \
>  --tls-key helm.key.pem version
Client: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
root@master1 $>
```

- Configure the Helm environment

  Configure the Helm client to use a secure connection to the Tiller server. Helm can receive parameters on the command line, through environmental variables, or by files in the $HELM_HOME directory.

```
root@master1 $> cp ca.cert.pem /root/.helm/ca.pem
root@master1 $> cp helm.cert.pem /root/.helm/cert.pem
root@master1 $> cp helm.key.pem /root/.helm/key.pem
root@master1 $> helm version --tls
Client: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
root@master1 $>
```

Task 4: Delete Tiller

Delete the tiller related objects:

```
root@master1 $> helm version --tls
Client: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
root@master1 $> kubectl -n kube-system get svc tiller-deploy
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)     AGE
tiller-deploy   ClusterIP   10.98.210.176   <none>        44134/TCP   3m
root@master1 $> kubectl -n kube-system get secrets tiller-secret
NAME            TYPE      DATA      AGE
tiller-secret   Opaque    3         4m
root@master1 $> kubectl -n kube-system get deployments tiller-deploy
NAME            DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
tiller-deploy   1         1         1            1           4m
root@master1 $> kubectl -n kube-system delete svc tiller-deploy
service "tiller-deploy" deleted
root@master1 $> kubectl -n kube-system delete deployments tiller-deploy
deployment.extensions "tiller-deploy" deleted
root@master1 $> kubectl -n kube-system delete secrets tiller-secret
secret "tiller-secret" deleted
root@master1 $> kubectl -n kube-system delete clusterrolebindings tiller-binding
clusterrolebinding.rbac.authorization.k8s.io "tiller-binding" deleted
root@master1 $> kubectl -n kube-system delete serviceaccounts tiller
serviceaccount "tiller" deleted
root@master1 $> helm version --tls
Client: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
Error: could not find tiller
```

Note:
If our Tiller deployment works properly, then we can use the *helm reset* command for removing Tiller. This command has options for removing the $HELM_HOME directory as well. Please have a look at *helm help reset*.

Task 5: Update Tiller

- create a default Tiller install

```
root@master1 $> helm init
$HELM_HOME has been configured at /root/.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.

Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated
users' policy.
To prevent this, run `helm init` with the --tiller-tls-verify flag.
For more information on securing your installation see: https://docs.helm.sh/using_helm
/#securing-your-helm-installation
Happy Helming!
root@master1 $>
root@master1 $> helm version
```

```
Client: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b11
5f93090", GitTreeState:"clean"}
root@master1 $> helm list
Error: configmaps is forbidden: User "system:serviceaccount:kube-system:default" cannot
list configmaps in the namespace "kube-system"
```

- Create the RBAC related objects for Tiller and update it:

```
root@master1 $> kubectl create serviceaccount --namespace kube-system tiller
serviceaccount/tiller created
root@master1 $> kubectl create clusterrolebinding tiller-cluster-rule \
> --clusterrole=cluster-admin --serviceaccount=kube-system:tiller
clusterrolebinding.rbac.authorization.k8s.io/tiller-cluster-rule created
root@master1 $> helm init --service-account tiller --upgrade
$HELM_HOME has been configured at /root/.helm.

Tiller (the Helm server-side component) has been upgraded to the current version.
Happy Helming!
```

Note:
Similar results can be achieved by patching the deployment to use the desired service account: .. console:

```
root@master1 $> kubectl patch deployment --namespace kube-system tiller-deploy \
> -p '{"spec":{"template":{"spec":{"serviceAccount":"tiller"}}}}'
deployment.extensions/tiller-deploy patched
```

Lab 12: Using helm

Task 1: Working with repositories

- List the currently configured repositories

```
root@master1 $> helm repo list
NAME            URL
stable          https://kubernetes-charts.storage.googleapis.com
local           http://127.0.0.1:8879/charts
root@master1 $>
```

- Add a new repo (https://charts.bitnami.com/bitnami)

```
root@master1 $> helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
root@master1 $> helm repo list
NAME            URL
stable          https://kubernetes-charts.storage.googleapis.com
local           http://127.0.0.1:8879/charts
bitnami         https://charts.bitnami.com/bitnami
root@master1 $>
```

- Update the repository cache

```
root@master1 $> helm repo update
Hang tight while we grab the latest from your chart repositories...
...Skip local chart repository
...Successfully got an update from the "bitnami" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. * Happy Helming!*
root@master1 $>
```

- Remove the local repository

```
root@master1 $> helm repo remove local
"local" has been removed from your repositories
root@master1 $> helm repo list
NAME            URL
stable          https://kubernetes-charts.storage.googleapis.com
bitnami         https://charts.bitnami.com/bitnami
root@master1 $>
```

Task 2: Managing releases.

- Search for a mariadb chart.

```
root@master1 $> helm search mariadb
NAME                       CHART VERSION   APP VERSION DESCRIPTION
bitnami/mariadb            4.4.2           10.1.36     Fast, reliable, scalable, and
easy to use open-source rel...
bitnami/mariadb-cluster    1.0.1           10.2.14     Chart to create a Highly avai
lable MariaDB cluster
stable/mariadb             4.4.2           10.1.36     Fast, reliable, scalable, and
easy to use open-source rel...
bitnami/phpmyadmin         0.1.10          4.8.2       phpMyAdmin is an mysql admini
stration frontend
stable/phpmyadmin          0.1.10          4.8.2       phpMyAdmin is an mysql admini
stration frontend
root@master1 $>
```

- Read the configurable parameters of the *stable/mariadb* chart.

```
root@master1 $> helm inspect values stable/mariadb
## Bitnami MariaDB image
## ref: https://hub.docker.com/r/bitnami/mariadb/tags/
##
image:
  registry: docker.io
  repository: bitnami/mariadb
  tag: 10.1.36-debian-9
  ## Specify a imagePullPolicy
  ## Defaults to 'Always' if image tag is 'latest', else set to 'IfNotPresent'
  ## ref: http://kubernetes.io/docs/user-guide/images/#pre-pulling-images
  ##
  pullPolicy: IfNotPresent

...

metrics:
  enabled: false
  image:
    registry: docker.io
    repository: prom/mysqld-exporter
    tag: v0.10.0
    pullPolicy: IfNotPresent
  resources: {}
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/port: "9104"

root@master1 $>
```

- Install the mariadb chart with the *master.persistence.enabled=false* and *slave.persistence.enabled=false* parameters and the name of the release being *mydb*.

```
root@master1 $> helm install --name mydb --set master.persistence.enabled=false \
> --set slave.persistence.enabled=false stable/mariadb
NAME:   mydb
LAST DEPLOYED: Mon Sep 17 09:52:23 2018
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Secret
NAME          TYPE    DATA  AGE
mydb-mariadb  Opaque  2     0s

==> v1/ConfigMap
NAME                                DATA  AGE
mydb-mariadb-master-init-scripts  1     0s
mydb-mariadb-master               1     0s
mydb-mariadb-slave                1     0s
mydb-mariadb-tests                1     0s
==> v1/Service
NAME                TYPE       CLUSTER-IP    EXTERNAL-IP  PORT(S)   AGE
mydb-mariadb        ClusterIP  10.96.41.97   <none>       3306/TCP  0s
mydb-mariadb-slave  ClusterIP  10.98.155.29  <none>       3306/TCP  0s

==> v1beta1/StatefulSet
NAME                 DESIRED  CURRENT  AGE
mydb-mariadb-master  1        1        0s
mydb-mariadb-slave   1        1        0s

==> v1/Pod(related)
NAME                   READY  STATUS             RESTARTS  AGE
mydb-mariadb-master-0  0/1    ContainerCreating  0         0s
mydb-mariadb-slave-0   0/1    ContainerCreating  0         0s


NOTES:

...

  3. To connect to slave service (read-only):

      mysql -h mydb-mariadb-slave.default.svc.cluster.local -uroot -p my_database

root@master1 $>
```

- Check the resources created in kubernetes (including the configmaps in the kube-system namespace), and the status of the release.

```
root@master1 $> kubectl get all -l release=mydb
NAME                        READY     STATUS    RESTARTS    AGE
pod/mydb-mariadb-master-0   1/1       Running   0           2h
pod/mydb-mariadb-slave-0    1/1       Running   0           2h

NAME                         TYPE        CLUSTER-IP     EXTERNAL-IP    PORT(S)     AGE
service/mydb-mariadb         ClusterIP   10.96.41.97    <none>         3306/TCP    2h
service/mydb-mariadb-slave   ClusterIP   10.98.155.29   <none>         3306/TCP    2h

NAME                                       DESIRED   CURRENT   AGE
```

```
statefulset.apps/mydb-mariadb-master   1         1         2h
statefulset.apps/mydb-mariadb-slave    1         1         2h
root@master1 $>

root@master1 $> kubectl -n kube-system get configmaps -l OWNER=TILLER
NAME      DATA      AGE
mydb.v1   1         2h
root@master1 $>
```

- Update the mydb release to disable the db replication.

```
root@master1 $> helm upgrade  --set master.persistence.enabled=false \
> --set slave.persistence.enabled=false --set replication.enabled=false \
> mydb stable/mariadb
Release "mydb" has been upgraded. Happy Helming!
LAST DEPLOYED: Mon Sep 17 12:10:22 2018
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/Secret
NAME          TYPE    DATA  AGE
mydb-mariadb  Opaque  1     2h

==> v1/ConfigMap
NAME                         DATA  AGE
mydb-mariadb-init-scripts  1     1m
mydb-mariadb                 1     1m
mydb-mariadb-tests           1     2h

==> v1/Service
NAME          TYPE       CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE
mydb-mariadb  ClusterIP  10.96.41.97  <none>       3306/TCP  2h

==> v1beta1/StatefulSet
NAME          DESIRED  CURRENT  AGE
mydb-mariadb  1        1        1m

==> v1/Pod(related)
NAME            READY  STATUS   RESTARTS  AGE
mydb-mariadb-0  1/1    Running  0         1m

...

  2. To connect to master service (read/write):

      mysql -h mydb-mariadb.default.svc.cluster.local -uroot -p my_database

root@master1 $>

root@master1 $> kubectl get all -l release=mydb
NAME                 READY     STATUS    RESTARTS   AGE
pod/mydb-mariadb-0   1/1       Running   0          17m

NAME                   TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
service/mydb-mariadb   ClusterIP   10.96.41.97   <none>        3306/TCP   2h
```

```
NAME                             DESIRED   CURRENT   AGE
statefulset.apps/mydb-mariadb    1         1         17m
root@master1 $>

root@master1 $> kubectl -n kube-system get configmaps -l OWNER=TILLER
NAME       DATA      AGE
mydb.v1    1         2h
mydb.v2    1         18m
root@master1 $>
```

- Delete the release

```
root@master1 $> helm delete mydb
release "mydb" deleted
root@master1 $> kubectl get all -l release=mydb
No resources found.
root@master1 $> kubectl -n kube-system get configmaps -l OWNER=TILLER
NAME       DATA      AGE
mydb.v1    1         2h
mydb.v2    1         23m
root@master1 $> helm delete mydb --purge
release "mydb" deleted
root@master1 $> kubectl -n kube-system get configmaps -l OWNER=TILLER
No resources found.
root@master1 $>
```

Lab 13: Working with charts.

Task 1: Investigate the directory structure of a chart

- Create a new chart using `helm create`

```
root@master1 ~ $> helm create first
Creating first
```

- List the content of the chart directory, and of the Chart.yaml

```
root@master1 ~ $> ls -R first
first:
charts  Chart.yaml  templates  values.yaml

first/charts:

first/templates:
deployment.yaml  _helpers.tpl  ingress.yaml  NOTES.txt  service.yaml

root@master1 ~ $> cd first/
root@master1 first $> cat Chart.yaml
apiVersion: v1
appVersion: "1.0"
description: A Helm chart for Kubernetes
name: first
version: 0.1.0
root@master1 first $>
```

Task 2: Modify the chart and package it with a new version.

- Change the description of the chart, and afterwards package it with a new version.

  Edit the Chart.yaml file and change the description field.

```
root@master1 ~ $> cat first/Chart.yaml
apiVersion: v1
appVersion: "1.0"
description: this is our first chart
name: first
version: 0.1.0
```

Package the chart as version 0.2.0:

```
root@master1 ~ $> helm package --version 0.2.0 first/
Successfully packaged chart and saved it to: /root/first-0.2.0.tgz
```

Verify that the changes are in the package:

```
root@master1 ~ $> helm inspect chart  ./first-0.2.0.tgz
apiVersion: v1
appVersion: "1.0"
description: this is our first chart
name: first
version: 0.2.0
```

Lab 14: Writing chart templates.

Task 1: Use values in the template

- Write a chart that will create ConfigMaps in the Kubernetes cluster. The name of the ConfigMap should be of the form: <release_name>-<release-time>, and it should contain one key named `description` with the value being the description of the chart.

  Create a new chart and remove the content of the template directory and the values file.

```
root@master1 ~ $> helm create cm1
Creating cm1
root@master1 ~ $> rm -f cm1/templates/*

root@master1 ~ $> vi cm1/values.yaml

# Default values for cm1.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.
```

  Create a template file with the following content:

```
root@master1 ~ $> cat cm1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-{{ .Release.Time.Seconds }}
data:
  description: {{ .Chart.Description }}
```

  Try the result of the chart using the `--debug --dry-run` options of `helm install`.

```
root@master1 ~ $> helm install --debug --dry-run --name cm-test cm1/
[debug] Created tunnel using local port: '38001'

[debug] SERVER: "127.0.0.1:38001"

[debug] Original chart version: ""
[debug] CHART PATH: /root/cm1

NAME:   cm-test
REVISION: 1
RELEASED: Wed Oct  3 07:23:17 2018
CHART: cm1-0.1.0
```

```
USER-SUPPLIED VALUES:
{}

COMPUTED VALUES:
{}

HOOKS:
MANIFEST:

---
# Source: cm1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-test-1538551397
data:
  description: A Helm chart for Kubernetes
root@master1 ~ $>
```

- Add a dependency named `child1` to the current chart. The child chart should create a `ConfigMap` having its name set by the `myName` value, and it should define one key named *username* that has the default value *root* if not specified otherwise with the `myUsername` value. Test the chart by providing the afore mentioned parameters in different ways (values file of the child, values file of the parent, command line).

  Create the child chart into the chart directory of the existing one, and remove the default template files created with it. Also clean up its values file.

```
root@master1 ~ $> cd cm1/charts/
root@master1 charts $> helm create child1
Creating child1
root@master1 charts $> rm -f child1/templates/*
root@master1 charts $> echo > child1/values.yaml
```

  Create a template file for the child chart with similar content:

```
root@master1 charts $> cat child1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ default "cm-child" .Values.myName }}
data:
  username: {{ default "root" .Values.myUser.userName }}
```

  Create a values file with the following content:

```
root@master1 charts $> cat child1/values.yaml
myName: child-default
myUser:
  userName: "alice"
  home: "/home/alice/"
```

  Try the chart without providing any parameter on the command line, and identify the values section and the generated manifest.

```
root@master1 charts $> cd
root@master1 ~ $> helm install --debug --dry-run --name dependency-test cm1/
...
COMPUTED VALUES:
child1:
  global: {}
  myName: child-default
  myUser:
    home: /home/alice/
    userName: alice

MANIFEST:

---
# Source: cm1/charts/child1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: child-default
data:
  username: alice
...
```

Note:
Both values were set using the `values.yaml` of the child chart.

Modify the `values.yaml` file of the parent chart to have the following content, and try again the previous command:

```
root@master1 ~ $> cat cm1/values.yaml
child1:
  myName: "set-by-parent-values-file"
root@master1 ~ $> helm install --debug --dry-run --name dependency-test cm1/
...
COMPUTED VALUES:
child1:
  global: {}
  myName: set-by-parent-values-file
  myUser:
    home: /home/alice/
    userName: alice
...
# Source: cm1/charts/child1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: set-by-parent-values-file
data:
  username: alice
...
root@master1 ~ $>
```

Note:
The key set in the `values.yaml` of the parent is transmitted to the child chart.

Test again the chart by setting the `child1.myUser.userName` to empty string from the command line:

```
root@master1 ~ $> helm install --debug --dry-run --name cm-test \
> --set child1.myUser.userName='' cm1/
...
USER-SUPPLIED VALUES:
child1:
  myUser:
    userName: ""

COMPUTED VALUES:
child1:
  global: {}
  myName: set-by-parent-values-file
  myUser:
    home: /home/alice/
    userName: ""
...
# Source: cm1/charts/child1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: set-by-parent-values-file
data:
  username: root
...
root@master1 ~ $>
```

Note:
The `username` will get its value from the `default` function in the template.

- Modify the parent chart to import the `myUser` from `child1` as `myChild1`, and use the `userName` value for a new key named `myChildUser` in its ConfigMap.

  Create a `requirements.yaml` file for the parent chart with the following content:

```
root@master1 ~ $> cat cm1/requirements.yaml
dependencies:
  - name: child1
    version: 0.1.0
    import-values:
      - child: myUser
        parent: myChild1
```

Modify the parent template by adding one line as in the following example:

```
root@master1 ~ $> cat cm1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-{{ .Release.Time.Seconds }}
data:
  description: {{ .Chart.Description }}
  myChildUser: {{ .Values.myChild1.userName }}
```

Test the changes:

```
root@master1 ~ $> helm install --debug --dry-run --name cm-test  cm1/
...
COMPUTED VALUES:
child1:
  global: {}
  myName: set-by-parent-values-file
  myUser:
    home: /home/alice/
    userName: alice
myChild1:
  home: /home/alice/
  userName: alice
...
# Source: cm1/charts/child1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: set-by-parent-values-file
data:
  username: alice
---
# Source: cm1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-test-1538583312
data:
  description: A Helm chart for Kubernetes
  myChildUser: alice
root@master1 ~ $>
```

Task 2: Use the flow control operations

- Verify whether the first character of the `userName` is lower or upper case, and convert it to upper case if needed (you can use the `title` function for this) when setting the `username` key in the child template.

  Modify the child template like this:

```
root@master1 ~ $> cat  cm1/charts/child1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ default "cm-child" .Values.myName }}
data:
  username:  {{ if regexFind "^[a-z]" .Values.myUser.userName }}
{{- title .Values.myUser.userName }}
{{ else }}
{{- .Values.myUser.userName }}
{{ end }}
root@master1 ~ $>
```

Test the new template with different values:

```
root@master1 ~ $> helm install --debug --dry-run --name cm-test  cm1/

COMPUTED VALUES:
child1:
  global: {}
  myName: set-by-parent-values-file
  myUser:
    home: /home/alice/
    userName: alice
myChild1:
  home: /home/alice/
  userName: alice

# Source: cm1/charts/child1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: set-by-parent-values-file
data:
  username:  Alice
...
root@master1 ~ $>
```

```
root@master1 ~ $> helm install --debug --dry-run --name cm-test \
> --set child1.myUser.userName='trudy' cm1/
...
USER-SUPPLIED VALUES:
child1:
  myUser:
    userName: trudy

COMPUTED VALUES:
child1:
  global: {}
  myName: set-by-parent-values-file
  myUser:
    home: /home/alice/
    userName: trudy
myChild1:
  home: /home/alice/
  userName: alice
```

```
HOOKS:
MANIFEST:

---
# Source: cm1/charts/child1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: set-by-parent-values-file
data:
  username: Trudy
...
root@master1 ~ $>
```

- Change the context (the dot) to `myUser` for the child template, and use the value of `home` for a new key named `homeDir` in the child ConfigMap.

  Modify the child template like this:

```
root@master1 ~ $> cat cm1/charts/child1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ default "cm-child" .Values.myName }}
data:
  username:  {{ if regexFind "^[a-z]" .Values.myUser.userName }}
{{- title .Values.myUser.userName }}
{{- else }}
{{- .Values.myUser.userName }}
{{- end }}
  {{- with .Values.myUser }}
  homeDir: {{ .home }}
  {{- end }}
root@master1 ~ $>
```

Test the new template:

```
root@master1 ~ $> helm install --debug --dry-run --name cm-test  cm1/
...
COMPUTED VALUES:
child1:
  global: {}
  myName: set-by-parent-values-file
  myUser:
    home: /home/alice/
    userName: alice
myChild1:
  home: /home/alice/
  userName: alice
...
# Source: cm1/charts/child1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: set-by-parent-values-file
data:
  username:  Alice
  homeDir: /home/alice/
...
```

Lab 15: Helm plugins

Task 1: Create a plugin named list-failed that will list the failed releases.

Create a subdirectory named `list-failed` in the `~/.helm/plugins` directory (in $HELM_HOME/plugins), and create the pugin.yaml file with the following content:

```
root@master1 ~ $> mkdir -p .helm/plugins/list-failed
root@master1 ~ $> cat .helm/plugins/list-failed/plugin.yaml
name: "list-failed"
version: "0.1.0"
usage: "List failed releases"
description: |-
  List the failed releases
ignoreFlags: false
useTunnel: false
command: "$HELM_PLUGIN_DIR/lf.sh"
```

```
root@master1 ~ $> cat .helm/plugins/list-failed/lf.sh
#! /bin/bash

for i in `helm list --failed --short`
do
    echo "========== release name: $i ============"
    helm history $i
    echo
done

root@master1 ~ $> chmod +x .helm/plugins/list-failed/lf.sh
root@master1 ~ $>
```

Introduce an error into the templates created in the previous lab (for example change the key `metadata` to something wrong like `metaadata`). and do an upgrade on the release.

```
root@master1 ~ $> helm install cm-test  cm1/
root@master1 ~ $> cat cm1/templates/cm.yaml
apiVersion: v1
kind: ConfigMap
metaadata:
  name: {{ .Release.Name }}
data:
  description: {{ .Chart.Description }}
  myChildUser: {{ .Values.myChild1.userNames }}

```

```
root@master1 ~ $> helm upgrade -i cm-test cm1/
[debug] Created tunnel using local port: '36675'

[debug] SERVER: "127.0.0.1:36675"

Error: UPGRADE FAILED: Could not get information about the resource: resource name may not be

root@master1 ~ $>
```

Try the new plugin:

```
root@master1 ~ $> helm list-failed
========== release name: cmtest ============
REVISION      UPDATED                  STATUS       CHART      DESCRIPTION
1             Thu Oct  4 11:27:32 2018 SUPERSEDED   cm1-0.1.0  Install complete
2             Thu Oct  4 11:29:27 2018  SUPERSEDED  cm1-0.2.0  Upgrade complete
3             Thu Oct  4 11:30:03 2018  DEPLOYED    cm1-0.2.0  Upgrade complete
4             Thu Oct  4 11:31:33 2018  FAILED      cm1-0.2.0  Upgrade "cmtest" failed: failed
```

Appendinx A: Upgrading kubernetes

- Check the current version of the *kubectl* client, the API server, and the nodes.

```
root@master1 $> kubectl version | cut -b-71
Client Version: version.Info{Major:"1", Minor:"6", GitVersion:"v1.6.3",
Server Version: version.Info{Major:"1", Minor:"6", GitVersion:"v1.6.3",
root@master1 $>
root@master1 $> kubectl get node
NAME      STATUS    AGE       VERSION
master1   Ready     190d      v1.6.3
worker1   Ready     190d      v1.6.3
worker2   Ready     190d      v1.6.3
worker3   Ready     190d      v1.6.3
root@master1 $>
```

- Download the desired version of *kubeadm*. For this exercise we will use v1.8.4.

Note:
The version of the latest stable release can be found at the following address: https://dl.k8s.io/release/stable.txt

```
root@master1 $> export VERSION=v1.8.4
root@master1 $> export ARCH=amd64

root@master1 $> cp /usr/bin/kubeadm /usr/bin/kubeadm.backup
root@master1 $> curl -sSL \
>  https://dl.k8s.io/release/${VERSION}/bin/linux/${ARCH}/kubeadm > /usr/bin/kubeadm
root@master1 $> chmod a+x /usr/bin/kubeadm

root@master1 $> kubeadm version | cut -b -73
kubeadm version: &version.Info{Major:"1", Minor:"8", GitVersion:"v1.8.4",
```

- Disable the swap. The current version of the *kubelet* does not support swapping by default (it can be overridden by a flag). Execute this step on **ALL** the servers.

```
root@master1 $> swapon -s
Filename                              Type            Size    Used    Priority
/dev/vdb1                             partition       5242876 3496    -1
root@master1 $> swapoff /dev/vdb1
root@master1 $> vi /etc/fstab
root@master1 $> grep swap /etc/fstab
# UUID=02b73b2e-9f4f-4313-b4e2-0b009946d409 swap       swap    defaults    0 0
root@master1 $>
```

- Upload configuration about the current state so *kubeadm upgrade* later can know how to configure the upgraded cluster

```
root@master1 $> kubeadm config upload from-flags
[uploadconfig] Storing the configuration used in ConfigMap "kubeadm-config"
in the "kube-system" Namespace
```

- Check which versions are available to upgrade to and validate whether your current cluster is upgradeable

```
root@master1 $> kubeadm upgrade plan
[preflight] Running pre-flight checks
[upgrade] Making sure the cluster is healthy:
[upgrade/health] Checking API Server health: Healthy
[upgrade/health] Checking Node health: All Nodes are healthy
[upgrade/health] Checking Static Pod manifests exists on disk:
   All manifests exist on disk
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with
   'kubectl -n kube-system get cm kubeadm-config -oyaml'
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: v1.6.3
[upgrade/versions] kubeadm version: v1.8.4
[upgrade/versions] Latest stable version: v1.8.4
[upgrade/versions] Latest version in the v1.6 series: v1.6.13


Components that must be upgraded manually after you've upgraded the
control plane with 'kubeadm upgrade apply':
COMPONENT   CURRENT      AVAILABLE
Kubelet     4 x v1.6.3   v1.6.13

Upgrade to the latest version in the v1.6 series:

COMPONENT            CURRENT   AVAILABLE
API Server           v1.6.3    v1.6.13
Controller Manager   v1.6.3    v1.6.13
Scheduler            v1.6.3    v1.6.13
Kube Proxy           v1.6.3    v1.6.13
Kube DNS             1.14.5    1.14.5

You can now apply the upgrade by executing the following command:

      kubeadm upgrade apply v1.6.13

_____________________________________________________________________

Components that must be upgraded manually after you've upgraded the
control plane with 'kubeadm upgrade apply':
COMPONENT   CURRENT      AVAILABLE
Kubelet     4 x v1.6.3   v1.8.4

Upgrade to the latest stable version:

COMPONENT            CURRENT   AVAILABLE
API Server           v1.6.3    v1.8.4
Controller Manager   v1.6.3    v1.8.4
Scheduler            v1.6.3    v1.8.4
```

```
Kube Proxy           v1.6.3    v1.8.4
Kube DNS             1.14.5    1.14.5

You can now apply the upgrade by executing the following command:

      kubeadm upgrade apply v1.8.4

_____________________________________________________________________
```

The *kubeadm* tool can update one minor version at a time, so we need to update our control plane to version 1.7 first, and then to 1.8.

- Upgrade the control plane to version 1.7.2:

```
root@master1 $> kubeadm upgrade apply v1.7.2
[preflight] Running pre-flight checks
[upgrade] Making sure the cluster is healthy:
[upgrade/health] Checking API Server health: Healthy
[upgrade/health] Checking Node health: All Nodes are healthy
[upgrade/health] Checking Static Pod manifests exists on disk:
  All manifests exist on disk
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system
get cm kubeadm-config -oyaml'
[upgrade/version] You have chosen to upgrade to version "v1.7.2"
[upgrade/versions] Cluster version: v1.6.3
[upgrade/versions] kubeadm version: v1.8.4
[upgrade/confirm] Are you sure you want to proceed with the upgrade? [y/N]: y
[upgrade/prepull] Will prepull images for components [kube-apiserver
kube-controller-manager kube-scheduler]

...

[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.7.2". Enjoy!

[upgrade/kubelet] Now that your control plane is upgraded, please proceed
 with upgrading your kubelets in turn.
```

- Make sure that all the system pods are running:

```
root@master1 $> kubectl --namespace=kube-system get pod -o wide
NAME                              READY  STATUS   RESTARTS  AGE   IP            NODE
etcd-master1                      1/1    Running  3         192d  10.10.10.51   master1
kube-apiserver-master1            1/1    Running  0         1m    10.10.10.51   master1
kube-controller-manager-master1   1/1    Running  0         2m    10.10.10.51   master1
kube-dns-3036648637-zmcqt         3/3    Running  0         1m    10.32.0.2     worker1
kube-proxy-lnkrs                  1/1    Running  0         1m    10.10.10.51   master1
kube-proxy-mt1kr                  1/1    Running  0         47s   10.10.10.52   worker1
kube-proxy-p9jh2                  1/1    Running  0         1m    10.10.10.53   worker2
kube-proxy-pgkkz                  1/1    Running  0         1m    10.10.10.54   worker3
kube-scheduler-master1            1/1    Running  0         1m    10.10.10.51   master1
kubernetes-dashboard-2039414953-q 1/1    Running  1         191d  10.47.0.2     worker2
weave-net-4rb6s                   2/2    Running  7         191d  10.10.10.53   worker2
weave-net-86qfc                   2/2    Running  7         191d  10.10.10.52   worker1
weave-net-b5p7l                   2/2    Running  6         191d  10.10.10.51   master1
```

```
weave-net-tlvwm                   2/2    Running  7         191d  10.10.10.54   worker3
```

- Upgrade the control plane to 1.8.4:

```
root@master1 $> kubeadm upgrade apply v1.8.4 --force
[preflight] Running pre-flight checks
[upgrade] Making sure the cluster is healthy:
[upgrade/health] Checking API Server health: Healthy
[upgrade/health] Checking Node health: All Nodes are healthy
[upgrade/health] Checking Static Pod manifests exists on disk:
 All manifests exist on disk
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with
 'kubectl -n kube-system get cm kubeadm-config -oyaml'
[upgrade/version] You have chosen to upgrade to version "v1.8.4"
[upgrade/versions] Cluster version: v1.7.2
[upgrade/versions] kubeadm version: v1.8.4

...

[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.8.4". Enjoy!

[upgrade/kubelet] Now that your control plane is upgraded, please proceed
 with upgrading your kubelets in turn.
```

- Prepare the host for maintenance, by making it unschedulable and evicting the workload:

```
root@master1 $> kubectl drain master1 --ignore-daemonsets
node "master1" cordoned
error: pods not managed by ReplicationController, ReplicaSet, Job, DaemonSet
 or StatefulSet (use --force to override): etcd-master1, kube-apiserver-master1,
 kube-controller-manager-master1, kube-scheduler-master1; pods with local storage
 (use --delete-local-data to override): weave-net-b5p7l
```

- Upgrade the packages on the host using *yum update*

```
root@master1 $> yum update

...

Complete!
```

- Reboot the master node

- When the node is back again, mark it schedulable:

```
root@master1 $>  kubectl uncordon master1
node "master1" uncordoned
root@master1 $> kubectl  get node
NAME       STATUS    ROLES     AGE       VERSION
master1    Ready     master    191d      v1.8.4
worker1    Ready     <none>    191d      v1.6.3
worker2    Ready     <none>    191d      v1.6.3
worker3    Ready     <none>    191d      v1.6.3
```

- Upgrade the packages on the worker nodes one by one using the following procedure:
- marking them unschedulable
- update the packages
- reboot the nodes
- mark the nodes schedulable again

```
root@master1 $> kubectl drain worker1 --ignore-daemonsets
node "worker1" cordoned
error: pods with local storage (use --delete-local-data to override): weave-net-86qfc
root@master1 $> ssh worker1

root@worker1 $> yum update -y

...

Complete!

root@worker1 $> reboot

root@master1 $> kubectl uncordon worker1
node "worker1" uncordoned

root@master1 $> kubectl get node
NAME       STATUS    ROLES     AGE       VERSION
master1    Ready     master    191d      v1.8.4
worker1    Ready     <none>    191d      v1.8.4
worker2    Ready     <none>    191d      v1.6.3
worker3    Ready     <none>    191d      v1.6.3
```

After updating all the other worker nodes we should have the following state:

```
root@master1 $> kubectl get node
NAME       STATUS    ROLES     AGE       VERSION
master1    Ready     master    192d      v1.8.4
worker1    Ready     <none>    191d      v1.8.4
worker2    Ready     <none>    191d      v1.8.4
worker3    Ready     <none>    191d      v1.8.4
```