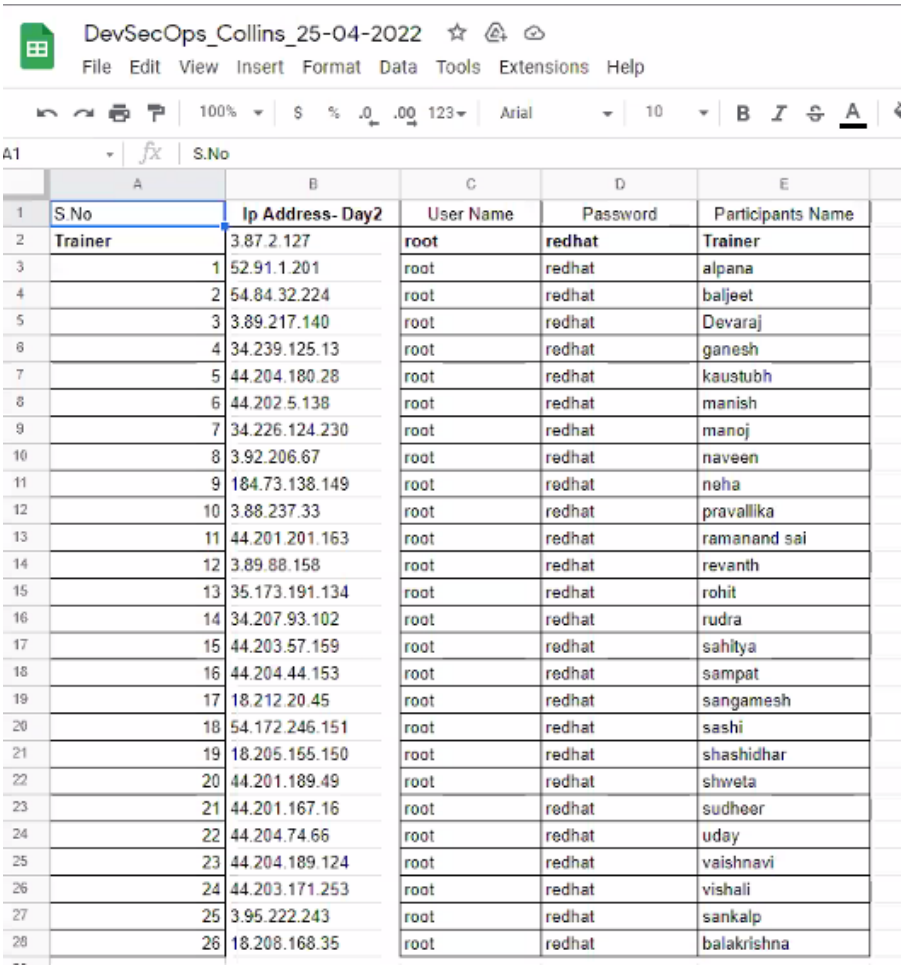
DevSecOps Training

DAY – 2

Today's IP Address details:


DevSecOps_Collins_25-04-2022

File Edit View Insert Format Data Tools Extensions Help

A1 | S.No

	A	B	C	D	E
1	S.No	Ip Address- Day2	User Name	Password	Participants Name
2	**Trainer**	3.87.2.127	**root**	**redhat**	**Trainer**
3	1	52.91.1.201	root	redhat	alpana
4	2	54.84.32.224	root	redhat	baljeet
5	3	3.89.217.140	root	redhat	Devaraj
6	4	34.239.125.13	root	redhat	ganesh
7	5	44.204.180.28	root	redhat	kaustubh
8	6	44.202.5.138	root	redhat	manish
9	7	34.226.124.230	root	redhat	manoj
10	8	3.92.206.67	root	redhat	naveen
11	9	184.73.138.149	root	redhat	neha
12	10	3.88.237.33	root	redhat	pravallika
13	11	44.201.201.163	root	redhat	ramanand sai
14	12	3.89.88.158	root	redhat	revanth
15	13	35.173.191.134	root	redhat	rohit
16	14	34.207.93.102	root	redhat	rudra
17	15	44.203.57.159	root	redhat	sahitya
18	16	44.204.44.153	root	redhat	sampat
19	17	18.212.20.45	root	redhat	sangamesh
20	18	54.172.246.151	root	redhat	sashi
21	19	18.205.155.150	root	redhat	shashidhar
22	20	44.201.189.49	root	redhat	shweta
23	21	44.201.167.16	root	redhat	sudheer
24	22	44.204.74.66	root	redhat	uday
25	23	44.204.189.124	root	redhat	vaishnavi
26	24	44.203.171.253	root	redhat	vishali
27	25	3.95.222.243	root	redhat	sankalp
28	26	18.208.168.35	root	redhat	balakrishna

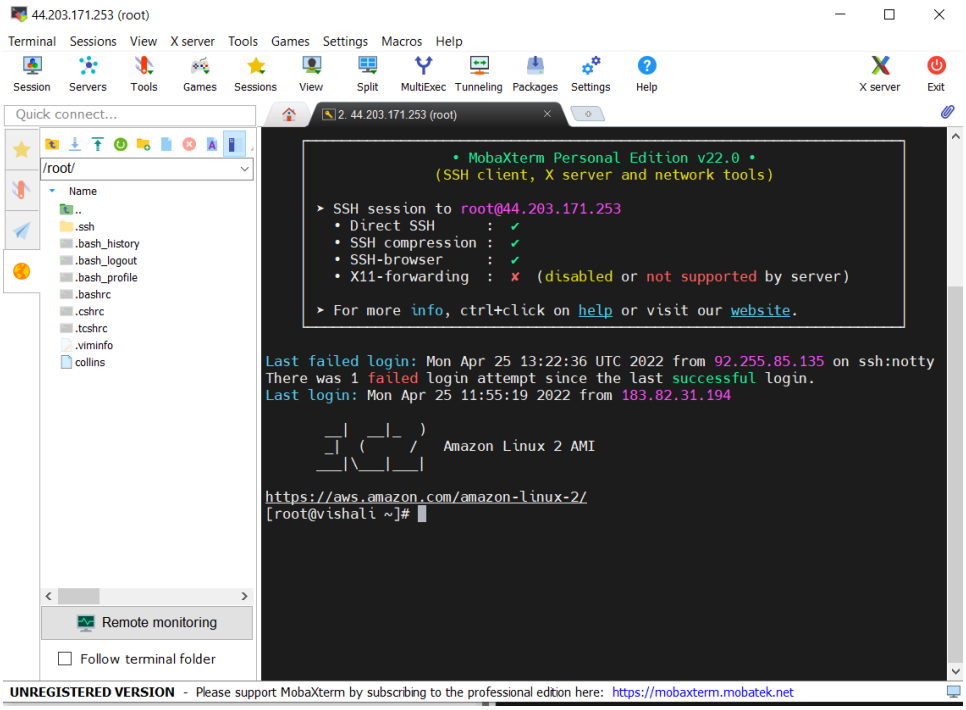
Details of my machine:

IP Address: 44.203.171.253

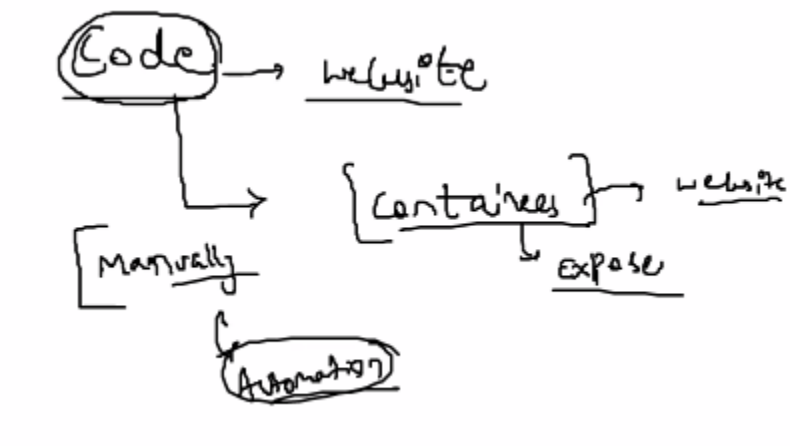
Username: root

Password: redhat

Login to MobaXterm:



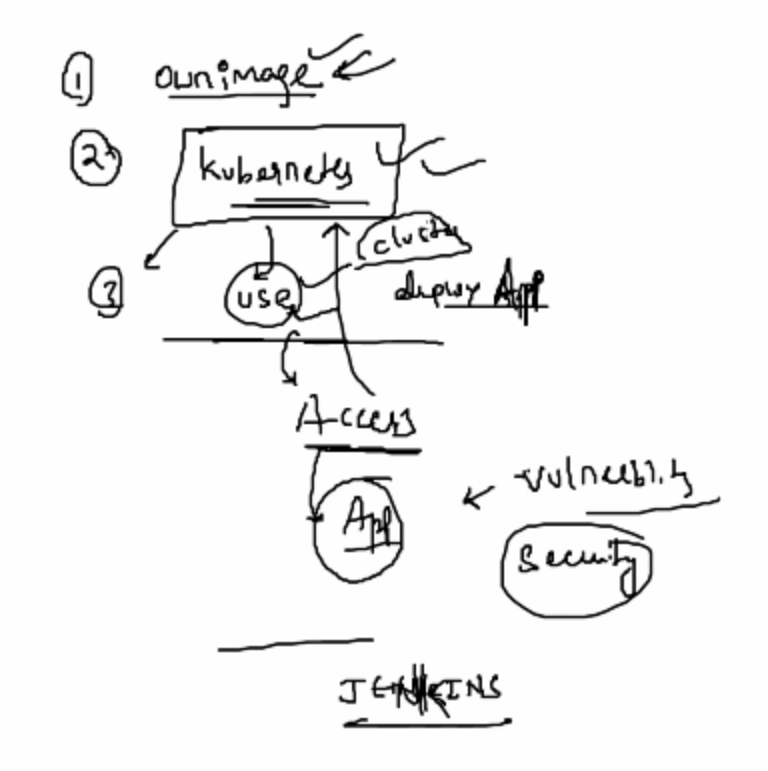
Yesterday we did the process manually



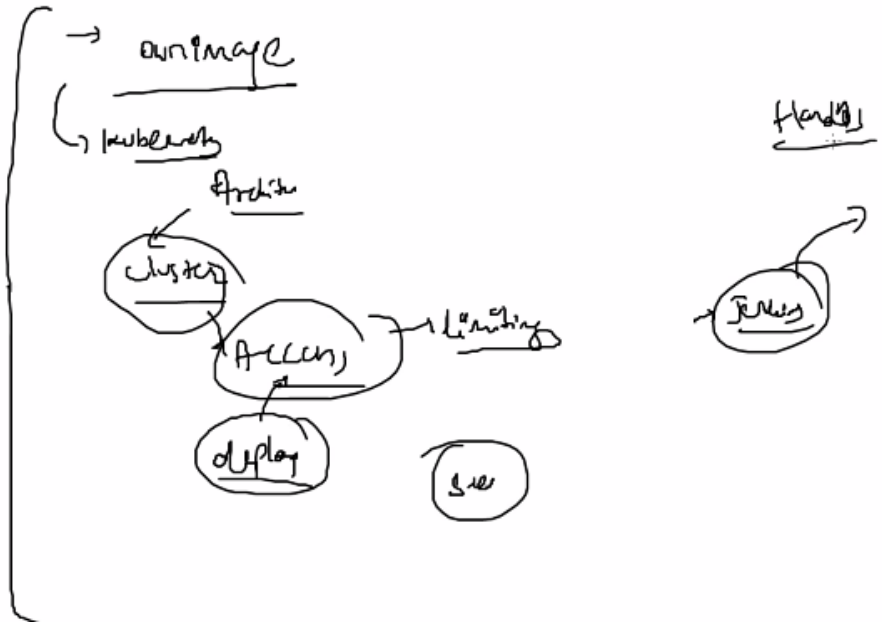
Now, we will learn to automate

Agenda for today:

- How to build the image
- Kubernetes (docker doesn't know how to manage, so we will learn this)
- How to use Kubernetes – Using this we can easily deploy our application
- How to create and access of the kubernetes cluster
- Create an application with your image – how to check the security /enhance the security/ vulnerability/ what things we should take care of while deploying the application
- Jenkins



Kuberenetes -> Create cluster -> Access -> Deploy -> Security

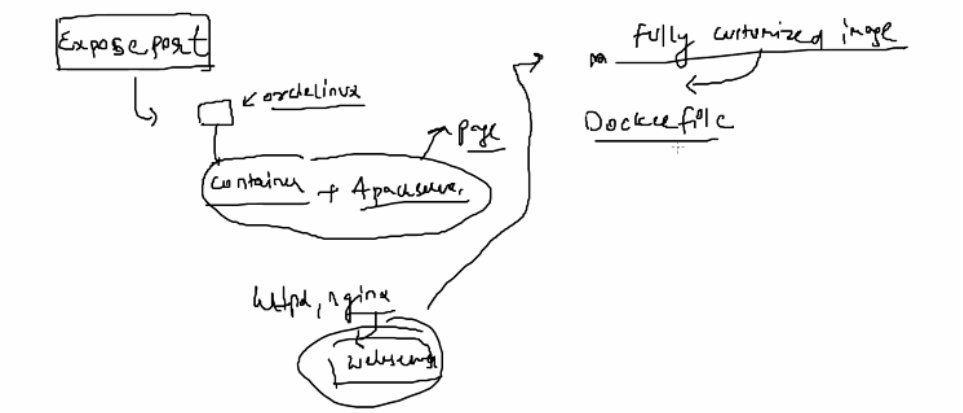


Expose port number:

We were using image oraclelinux image -> create container + Configure Apache server -> Access page.

Httpd, nginx -> These images already have the Apache server configured -> Webserver

These are fully customized image -> Dockerfile

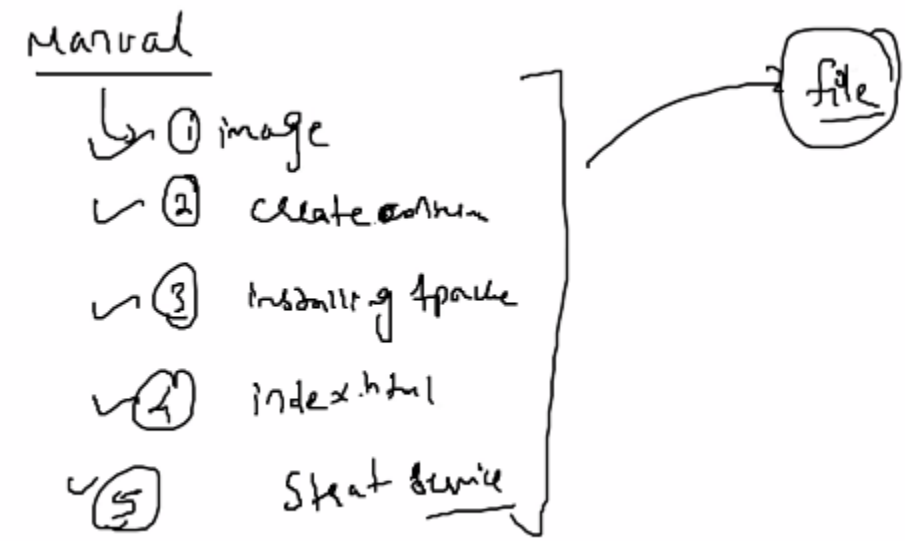


If you want to write something inside the dockerfile, what language you follow?

Programmatic language? NO

Declarative language? Yes, somewhat!

It is based on instruction.



This can be automated when we write into the Dockerfile.

Instruction:

1. FROM
2. RUN
3. RUN
4. Parent process – when the container is created by the image - Starting of the service – CMD

Parent process can’t be given by RUN.

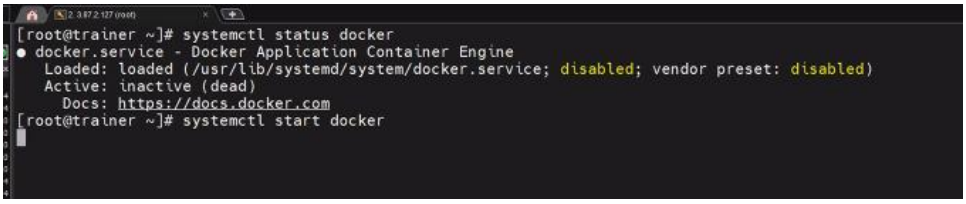
Httpd – what is the parent process?

This is the parent process - Httpd –DFOREGROUND -> will start the service

Suggestion:

Try to create the dockerfile in the empty folder.

Check for the status of docker, if its inactive -> start the docker



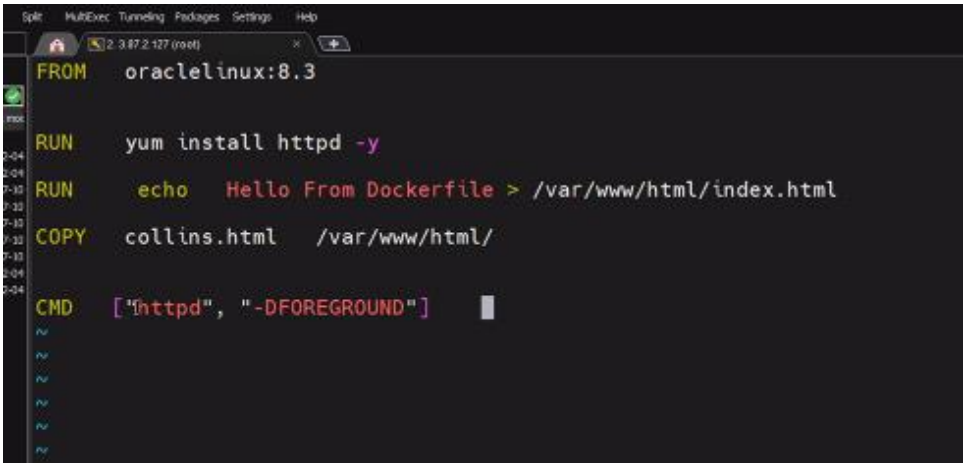
Create the docker file:

Suggestion: D - CAPS

Dockerfile:

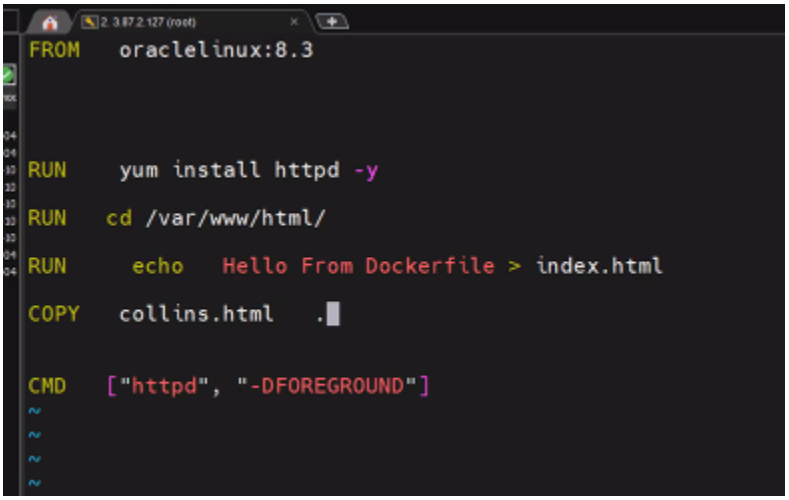
1. Mention the base image - FROM
2. Install the apache server - RUN
3. Create the index.html - RUN echo

4. Copy command will allow to copy local html file into the docker container location.
5. Start the service – cmd



To make it efficient:

Use cd



Use WORKDIR instead of the cd – better!

```
FROM    oraclelinux:8.3



RUN     yum install httpd -y

WORKDIR /var/www/html/

RUN      echo   Hello From Dockerfile > index.html

COPY    collins.html    .

CMD    ["httpd", "-DFOREGROUND"]
~
~
~
~
```

Maintainer: (optional)

```
FROM    oraclelinux:8.3

MAINTAINER  mayank

RUN     yum install httpd -y

WORKDIR  /var/www/html/

RUN      echo   Hello From Dockerfile > index.html

COPY    collins.html    .

CMD    ["httpd", "-DFOREGROUND"]
~
~
~
~
~
```

Expose:

```
FROM    oraclelinux:8.3

MAINTAINER  mayank

RUN     yum install httpd -y

WORKDIR  /var/www/html/

RUN      echo   Hello From Dockerfile > index.html

COPY    collins.html    .

EXPOSE 80


CMD    ["httpd", "-DFOREGROUND"]
~
~
```

Save and verify:

```
[root@trainer docker]# cat Dockerfile
FROM   oraclelinux:8.3

MAINTAINER  mayank

RUN    yum install httpd -y

WORKDIR  /var/www/html/

RUN     echo   Hello From Dockerfile > index.html

COPY   collins.html   .


EXPOSE 80



CMD   ["httpd", "-DFOREGROUND"]
```

Delete the image:

```
[root@trainer docker]# docker rm -f test2
test2
[root@trainer docker]# docker rm -f test2
```

To get all the container id:

```
[root@trainer docker]# docker ps -aq
3833168a4aa2
54c6dce468ca
645a1ab87aed
81b9dbcd8219
5afd963ce3ef
a48f731ef3d6
6837ab21239b
fa51e54d324b
df5ab63ec788
3d3fd9b51510
0b0f24cfdd47
f042ec8ee747
6fd4f7f62915
[root@trainer docker]#
```

All the container in one go – removed

```
[root@trainer docker]# docker rm -f  $(docker ps -aq)
3833168a4aa2
54c6dce468ca
645a1ab87aed
81b9dbcd8219
5afd963ce3ef
a48f731ef3d6
6837ab21239b
fa51e54d324b
df5ab63ec788
3d3fd9b51510
0b0f24cfdd47
f042ec8ee747
6fd4f7f62915
[root@trainer docker]#
```

Delete one image:

```
[root@trainer docker]# docker images
REPOSITORY                TAG       IMAGE ID       CREATED         SIZE
nginx                     latest    fa5269854a5e   5 days ago      142MB
httpd                     latest    c30a46771695   5 days ago      144MB
mdhack/mayanknginximage   latest    b7307762462d   8 weeks ago     142MB
hello-world               latest    feb5d9fea6a5   7 months ago    13.3kB
centos                    latest    5d0da3dc9764   7 months ago    231MB
oraclelinux               8.3       816d99f0bbe8   12 months ago   224MB
mdhack/myapache           latest    ea36c5376ba4   20 months ago   166MB
mdhack/myserver           latest    e8ce7504414a   20 months ago   350MB
[root@trainer docker]# docker rmi  -f  hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:10d7d58d5ebd2a652f4d93fdd86da8f265f5318c6a73cc5b6a9798ff6d2b2
Deleted: sha256:feb5d9fea6a5e9606aa995e879d862b825965ba48de054caab5ef356dc6b3412
Deleted: sha256:e07ee1baac5fae6a26f30cabfe54a36d3402f96afda318fe0a96cec4ca393359
```

All the docker images – delete at once

```
[root@trainer docker]# docker images -q
fa5269854a5e
c30a46771695
b7307762462d
5d0da3dc9764
816d99f0bbe8
ea36c5376ba4
e8ce7504414a
[root@trainer docker]# docker rmi -f $(docker images -q)
```

Build the docker image from the dockerfile:

```
[root@trainer docker]# docker build   -t mayankimage   .
Sending build context to Docker daemon  2.048kB
Step 1/8 : FROM   oraclelinux:8.3
8.3: Pulling from library/oraclelinux
dd34f38d274c: Pull complete
Digest: sha256:af3182ee6c1e56f18fc1fecaf638da57d7c47233862e5c32f5ad723a6ab4c6db
Status: Downloaded newer image for oraclelinux:8.3
 ---> 816d99f0bbe8
Step 2/8 : MAINTAINER  mayank
 ---> Running in 0a2b11cae54f
Removing intermediate container 0a2b11cae54f
 ---> 47d53b1c921a
Step 3/8 : RUN     yum install httpd -y
 ---> Running in 883ae83e478e
Oracle Linux 8 BaseOS Latest (x86_64)           76 MB/s |  43 MB     00:00
Oracle Linux 8 Application Stream (x86_64)      74 MB/s |  33 MB     00:00
Last metadata expiration check: 0:00:09 ago on Tue Apr 26 04:46:49 2022.
Dependencies resolved.
================================================================================
 Package             Arch    Version                                 Repository         Size
================================================================================
Installing:
 httpd               x86_64  2.4.37-43.0.3.module+el8.5.0+20624+5d3b49d0.3 ol8_appstream      1.4 M
Installing dependencies:
```

Now, the problem is we didn't create collins.html - create it

```
[root@trainer docker]# docker build   -t mayankimage   .
Sending build context to Docker daemon  3.072kB
Step 1/8 : FROM   oraclelinux:8.3
 ---> 816d99f0bbe8
Step 2/8 : MAINTAINER  mayank
 ---> Using cache
 ---> 47d53b1c921a
Step 3/8 : RUN    yum install httpd -y
 ---> Using cache
 ---> 7b3eaa838885
Step 4/8 : WORKDIR  /var/www/html/
 ---> Using cache
 ---> ae009bc9afb9
Step 5/8 : RUN     echo   Hello From Dockerfile > index.html
 ---> Using cache
 ---> b1a7d8609eed
Step 6/8 : COPY   collins.html   .
 ---> 9ed27d68595f
Step 7/8 : EXPOSE 80
 ---> Running in 9d00b77583cb
Removing intermediate container 9d00b77583cb
 ---> a40858ca4bf2
Step 8/8 : CMD   ["httpd", "-DFOREGROUND"]
 ---> Running in 6dd1e6fd689a
Removing intermediate container 6dd1e6fd689a
 ---> 739677877265
Successfully built 739677877265
Successfully tagged mayankimage:latest
[root@trainer docker]# dock
```

Why the size has been increased? Since it is on top of oracle linux we added multiple layers

```
[root@trainer docker]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
mayankimage   latest    739677877265   8 seconds ago    393MB
oraclelinux   8.3       816d99f0bbe8   12 months ago    224MB
[root@trainer docker]#
```

Added multiple layers

```
[root@trainer docker]# docker history   oraclelinux:8.3
IMAGE          CREATED        CREATED BY                                      SIZE      COMMENT
816d99f0bbe8   12 months ago  /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>      12 months ago  /bin/sh -c #(nop) ADD file:8d6d5e7607cbe7af8…   224MB
[root@trainer docker]# docker history   mayankimage:latest
IMAGE          CREATED              CREATED BY                                      SIZE      COMMENT
739677877265   46 seconds ago       /bin/sh -c #(nop)  CMD ["httpd" "-DFOREGROUN…   0B
a40858ca4bf2   46 seconds ago       /bin/sh -c #(nop)  EXPOSE 80                    0B
9ed27d68595f   46 seconds ago       /bin/sh -c #(nop) COPY file:ef9587d7f4c82229…   21B
b1a7d8609eed   About a minute ago   /bin/sh -c echo   Hello From Dockerfile > in…   22B
ae009bc9afb9   About a minute ago   /bin/sh -c #(nop) WORKDIR /var/www/html/        0B
7b3eaa838885   About a minute ago   /bin/sh -c yum install httpd -y                 169MB
47d53b1c921a   About a minute ago   /bin/sh -c #(nop)  MAINTAINER mayank            0B
816d99f0bbe8   12 months ago        /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>      12 months ago        /bin/sh -c #(nop) ADD file:8d6d5e7607cbe7af8…   224MB
[root@trainer docker]#
```

Create the container:

```
[root@trainer docker]# docker images
REPOSITORY    TAG       IMAGE ID       CREATED              SIZE
mayankimage   latest    739677877265   About a minute ago   393MB
oraclelinux   8.3       816d99f0bbe8   12 months ago        224MB
[root@trainer docker]# docker run  -itd   --name newcont1 -p 3333:80     mayankimage
3fcb615f00ce99d4f44af8ad112b5926aa5324247db1b8365edaadf31185fd42
[root@trainer docker]#
```

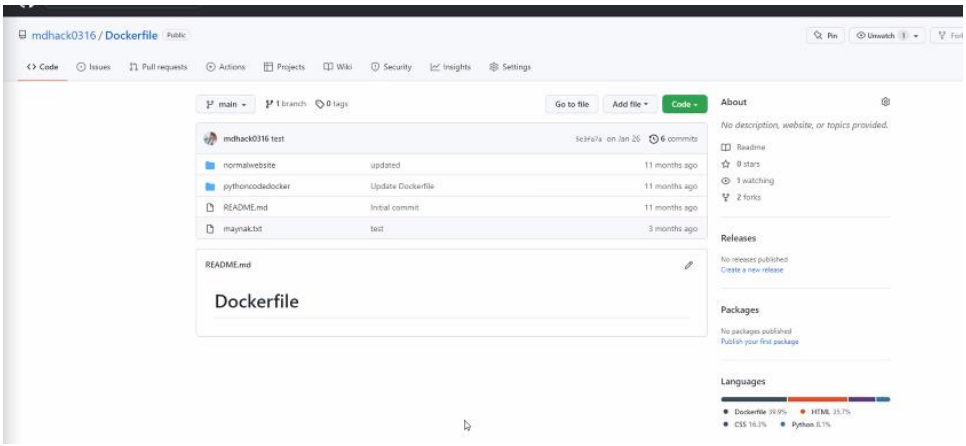
Now, you can check the website.

You will see the message written inside the index.html

If someone didn't mention the port number – random port number will be assigned!

```
[root@trainer docker]# docker run  -itd    -P      mayankimage
1737c454adcabd5318795f1bb9bd4d4be11b07aa44637456e090bb3aebb295da
[root@trainer docker]# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS
          NAMES
1737c454adca   mayankimage    "httpd -DFOREGROUND"   2 seconds ago   Up 1 second    0.0.0.0:49153->80/tcp, :::49153->8
0/tcp   jolly_mendeleev
3fcb615f00ce   mayankimage    "httpd -DFOREGROUND"   2 minutes ago   Up 2 minutes   0.0.0.0:3333->80/tcp, :::3333->80/
tcp     newcont1
[root@trainer docker]#
```

GitHub:



Here, we have a dockerfile

For downloading this repository, we need to install git

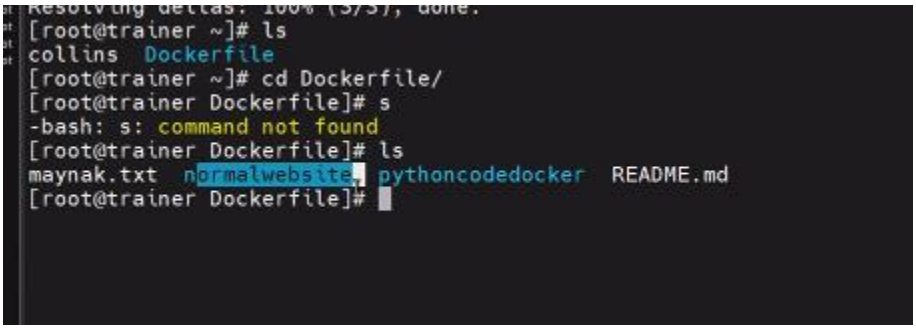
Git install - yum install git -y

git

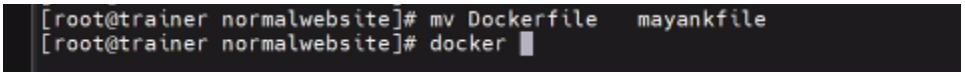
```
[root@vishali ~]# yum install git -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 3.7 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package git.x86_64 0:2.32.0-1.amzn2.0.1 will be installed
--> Processing Dependency: perl-Git = 2.32.0-1.amzn2.0.1 for package: git-2.32
-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core-doc = 2.32.0-1.amzn2.0.1 for package: git-
32.0-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core = 2.32.0-1.amzn2.0.1 for package: git-2.32
-1.amzn2.0.1.x86_64
```

Git clone:

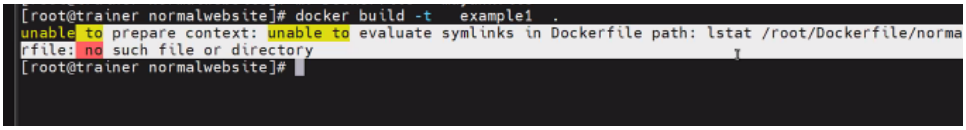
Check for the files inside it:



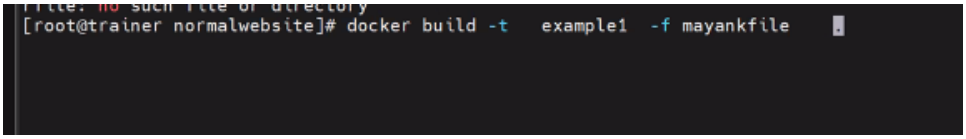
Change the name of the docker file



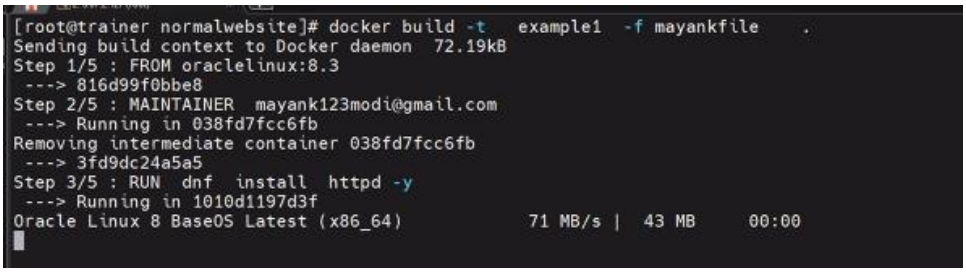
Dockerfile – not found



If you want to change the docker file name, then specify the –f tag:



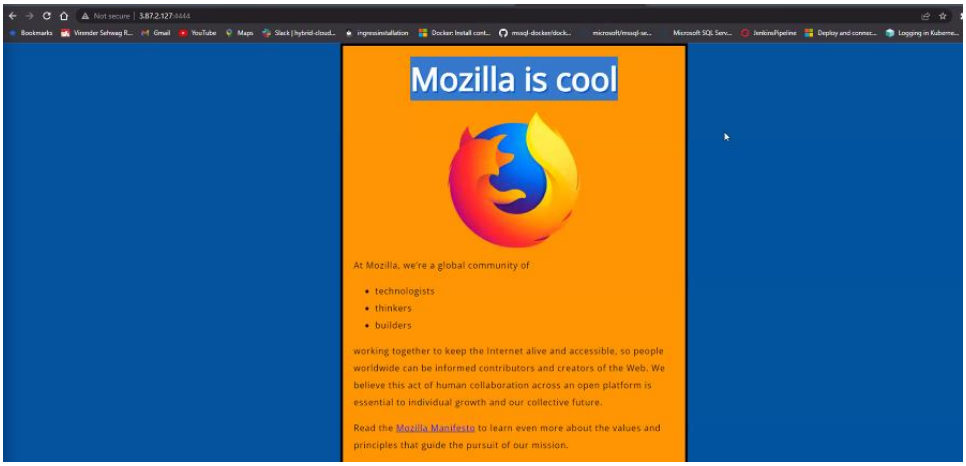
Build the image:



Run the container:

```
[root@trainer normalwebsite]# docker run -itd -p 4444:80  example1
4ea14fe899c89fe56a0d36574306d00765d6f9ddcc7c8f665643158210a3616d
[root@trainer normalwebsite]# vim
```

Check the website:



Verify the index.html:

```
[root@trainer normalwebsite]# vim index.html
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
    <link href="http://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet"
    <link href="styles/style.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <h1>Mozilla is cool</h1>
    <img src="images/firefox-icon.png" alt="The Firefox logo: a flaming fox surrou

    <p>At Mozilla, we're a global community of</p>

    <ul> <!-- changed to list in the tutorial -->
      <li>technologists</li>
      <li>thinkers</li>
      <li>builders</li>
    </ul>

    <p>working together to keep the Internet alive and accessible, so people worldw
nd creators of the Web. We believe this act of human collaboration across an open p
 growth and our collective future.</p>

    <p>Read the <a href="https://www.mozilla.org/en-US/about/manifesto/">Mozilla Ma
ut the values and principles that guide the pursuit of our mission.</p>
  </body>
</html>
~
~
~
~
~
~
~
:q1
```

Change the content inside index.html:

Open the container in exec mode.

```
[root@trainer normalwebsite]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS
          NAMES
4ea14fe899c8   example1       "httpd -DFOREGROUND"   42 seconds ago   Up 40 seconds   0.0.0.0:4444->80/tcp, :::4444->8
0/tcp     crazy_spence
1737c454adca   mayankimage    "httpd -DFOREGROUND"   7 minutes ago    Up 6 minutes    0.0.0.0:49153->80/tcp, :::49153-
>80/tcp   jolly_mendeleev
3fcb615f00ce   mayankimage    "httpd -DFOREGROUND"   9 minutes ago    Up 8 minutes    0.0.0.0:3333->80/tcp, :::3333->8
0/tcp     newcont1
[root@trainer normalwebsite]# docker exec -it crazy_spence  bash
[root@4ea14fe899c8 /]# cd /var/www/html/
[root@4ea14fe899c8 html]# vi index.html
[root@4ea14fe899c8 html]#
```

Change the content here:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
    <link href="http://fonts.googleapis.com/css?family=Open+Sans" rel="styleshe
    <link href="styles/style.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <h1>XC is cool</h1>
    <img src="images/firefox-icon.png" alt="The Firefox logo: a flaming fox sur

    <p>At Mozilla, we re a global community of</p>

    <ul> <!-- changed to list in the tutorial -->
      <li>technologists</li>
      <li>thinkers</li>
      <li>builders</li>
    </ul>

    <p>working together to keep the Internet alive and accessible, so people wo
nd creators of the Web. We believe this act of human collaboration across an op
 growth and our collective future.</p>

    <p>Read the <a href="https://www.mozilla.org/en-US/about/manifesto/">Mozill
ut the values and principles that guide the pursuit of our mission.</p>
  </body>
</html>
~
~
~
~
~
```

Now you can see the content change in the website:

To see **what containers are running:**

```
[root@trainer normalwebsite]# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS
                NAMES
4ea14fe899c8   example1       "httpd -DFOREGROUND"     About a minute ago   Up About a minute   0.0.0.0:4444->80/tcp, ::
:4444->80/tcp      crazy_spence
1737c454adca   mayankimage    "httpd -DFOREGROUND"     8 minutes ago        Up 7 minutes        0.0.0.0:49153->80/tcp, :
::49153->80/tcp    jolly_mendeleev
3fcb615f00ce   mayankimage    "httpd -DFOREGROUND"     10 minutes ago       Up 9 minutes        0.0.0.0:3333->80/tcp, ::
:3333->80/tcp      newcont1
[root@trainer normalwebsite]#
```

Python code:

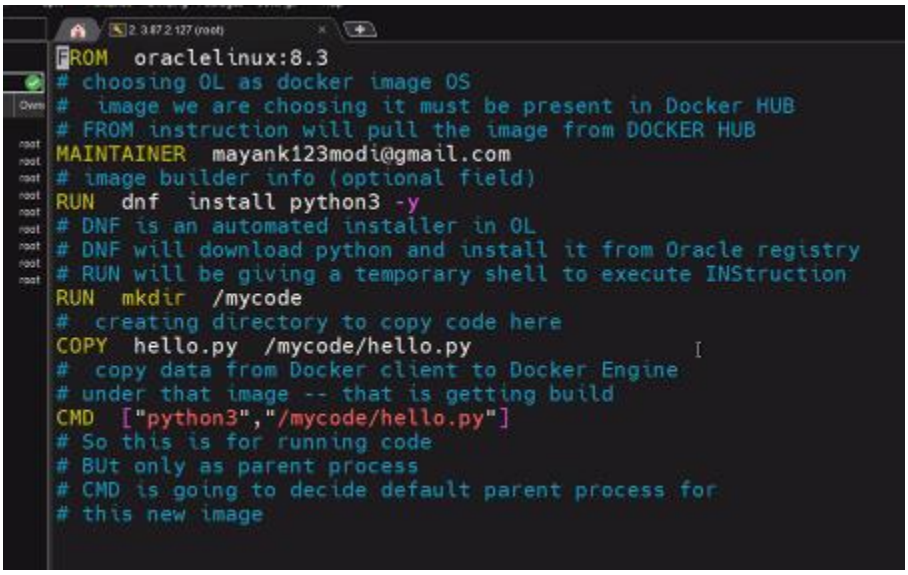
```
[root@trainer pythoncodedocker]# cat hello.py
import time

while True:
    print("Hello all , welcome to python..!!")
    time.sleep(3)
    print("Welcome to Mayank Containers")
    time.sleep(2)
    print("Welcome to Containers ..!!")
    print("_____________________")
    time.sleep(3)

[root@trainer pythoncodedocker]# ls
Dockerfile  hello.py
[root@trainer pythoncodedocker]#
```

What is it trying to do? Change/Print the statement after few seconds

Understand the dockerfile:



```
FROM  oraclelinux:8.3
# choosing OL as docker image OS
#  image we are choosing it must be present in Docker HUB
# FROM instruction will pull the image from DOCKER HUB
MAINTAINER  mayank123modi@gmail.com
# image builder info (optional field)
RUN  dnf  install python3 -y
# DNF is an automated installer in OL
# DNF will download python and install it from Oracle registry
# RUN will be giving a temporary shell to execute INStruction
RUN  mkdir  /mycode
#  creating directory to copy code here
COPY  hello.py  /mycode/hello.py
#  copy data from Docker client to Docker Engine
# under that image -- that is getting build
CMD  ["python3","/mycode/hello.py"]
# So this is for running code
# BUt only as parent process
# CMD is going to decide default parent process for
# this new image
```

Build the dockerfile:

```
[root@trainer pythoncodedocker]# vim Dockerfile
[root@trainer pythoncodedocker]# docker build -t python .
Sending build context to Docker daemon  3.584kB
Step 1/6 : FROM  oraclelinux:8.3
 ---> 816d99f0bbe8
Step 2/6 : MAINTAINER  mayank123modi@gmail.com
 ---> Using cache
 ---> 3fd9dc24a5a5
Step 3/6 : RUN  dnf  install python3 -y
 ---> Running in eead6edaf1cc
Oracle Linux 8 BaseOS Latest (x86_64)           80 MB/s |  43 MB     00:00
```

To run the container in the foreground:

```
[root@trainer pythoncodedocker]# docker run -it python
Hello all , welcome to python..!!
Welcome to Mayank Containers
Welcome to Containers ..!!

Hello all , welcome to python..!!
```

To run the container in the background:

```
[root@trainer pythoncodedocker]# docker run -itd   python
4dd2762077995c4ce9a2703df0946f27c9f728bd71ad151e8dbb6fe4eb13d640
[root@trainer pythoncodedocker]# docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS          PORTS
          NAMES
4dd276207799   python        "python3 /mycode/hel…"   2 seconds ago    Up 1 second
          awesome_greider
4ea14fe899c8   example1      "httpd -DFOREGROUND"     4 minutes ago    Up 4 minutes    0.0.0.
>80/tcp     crazy_spence
1737c454adca   mayankimage   "httpd -DFOREGROUND"     10 minutes ago   Up 10 minutes   0.0.0.
8->80/tcp   jolly_mendeleev
3fcb615f00ce   mayankimage   "httpd -DFOREGROUND"     12 minutes ago   Up 12 minutes   0.0.0.
>80/tcp     newcont1
```

To check the logs:

```
[root@trainer pythoncodedocker]# docker logs awesome_greider
Hello all , welcome to python..!!
Welcome to Mayank Containers
Welcome to Containers ..!!

Hello all , welcome to python..!!
Welcome to Mayank Containers
Welcome to Containers ..!!

Hello all , welcome to python..!!
[root@trainer pythoncodedocker]#
```

Live log: Mention –f tag

Reference to Docker commands: (Shared by Trainer)

191 systemctl status docker

192 systemctl start docker

193 systemctl status docker

194 docker images

195 docker ps

196 docker ps -a

197 mkdir /docker

198 cd /docker/

199 ls

200 pwd

201 vim Dockerfile

202 cat Dockerfile

203 docker images

204 ls

205 docker ps

206 docker ps -a

207 docker rm -f test2

208 docker ps -a

209 docker rm -f fcf37b509826

210 docker ps -a

```
211  docker ps -aq
212  docker rm -f  $(docker ps -aq)
213  docker ps -a
214  docker images
215  docker rmi  -f  hello-world
216  docker images
217  docker images -q
218  docker rmi -f $(docker images -q)
219  docker images
220  docker ps -a
221  cat Dockerfile
222  #docker build   -t mayankimage
223  l
224  ls
225  docker build   -t mayankimage   .
226  ls
227  vim collins.html
228  docker build   -t mayankimage   .
229  docker images
230  docker history   oraclelinux:8.3
231  docker history   mayankimage:latest
232  docker iamges
233  docker images
234  docker run  -itd   --name newcont1 -p 3333:80     mayankimage
235  cat Dockerfile
236  <<X
237  vim Dockerfile
238  docker images
239  vim Dockerfile
240  docker run  -itd    -P      mayankimage
```

241 docker ps -a

242 cd

243 git

244 yum install git -y

245 git clone https://github.com/mdhack0316/Dockerfile

246 ls

247 cd Dockerfile/

248 s

249 ls

250 cd normalwebsite/

251 ls

252 vim Dockerfile

253 mv Dockerfile mayankfile

254 docker build -t example1 .

255 docker build -t example1 -f mayankfile .

256 docker images

257 docker run -itd -p 4444:80 example1

258 vim index.html

259 docker ps

260 docker exec -it crazy_spence bash

261 docker ps -a

262 cd ..

263 ls

264 cd pythoncodedocker/

265 ls

266 cat hello.py

267 ls

268 vim Dockerfile

269 docker build -t python .

270 docker run -it python

271 docker run -itd python

272 docker ps

273 docker logs awesome_greider

274 docker logs -f awesome_greider

275 ls

276 cd ..

277 ls

TASK: 1

Q.1 create a dockerfile using as base image oraclelinux:8.3 and configure apache server and there should be 3 files index.html training.html trainer.html you need to write something inside all these files. And then create image by ur name and try to run it with any random port no. and verify whether it is accessible or not

Check for the docker service and create a folder docker:

```
[root@vishali ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled;
et: disabled)
   Active: inactive (dead)
     Docs: https://docs.docker.com
[root@vishali ~]# systemctl start docker
[root@vishali ~]# mkdir /docker
[root@vishali ~]# cd /docker/
[root@vishali docker]# ls
[root@vishali docker]# pwd
/docker
[root@vishali docker]# vim Dockerfile
```

Docker file:

```
[root@vishali docker]# cat Dockerfile
FROM oraclelinux:8.3
RUN yum install httpd -y
RUN echo Hello from index file > /var/www/html/index.html
RUN echo Hello from training file > /var/www/html/training.html
RUN echo Hello from trainer file > /var/www/html/trainer.html
EXPOSE 80
CMD ["httpd","-DFOREGROUND"]
[root@vishali docker]#
```


Build the image:

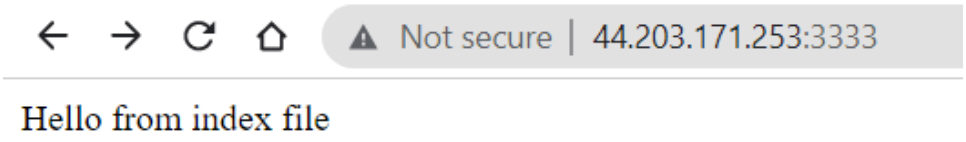
```
[root@vishali docker]# docker build -t vishaliimage .
Sending build context to Docker daemon  2.048kB
Step 1/7 : FROM oraclelinux:8.3
 ---> 816d99f0bbe8
Step 2/7 : RUN yum install httpd -y
 ---> Running in c49fa482ea23
```

Run the container:

```
[root@vishali docker]# docker run -itd --name newcont1 -p 3333:80 vishaliimage
20e3db89d5ae82189dc59dbbab080ef9c3e5a40df38c273355152048bbcfd34e
[root@vishali docker]#
```

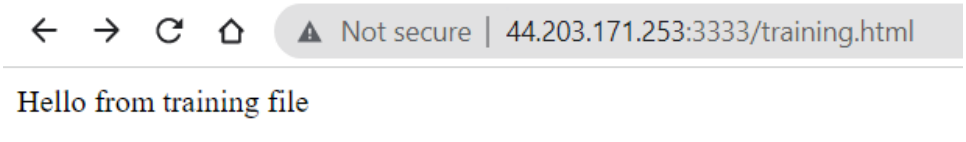
Verification:

http://44.203.171.253:3333/



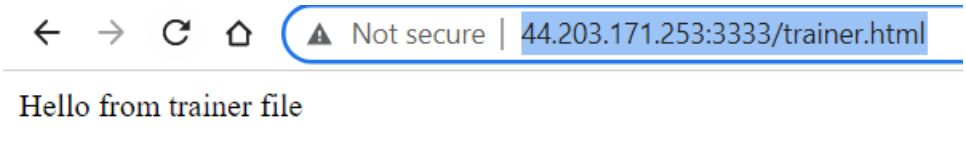
Hello from index file

http://44.203.171.253:3333/training.html



Hello from training file

http://44.203.171.253:3333/trainer.html



Hello from trainer file

Q. 2 Use git command to clone the sae repo that I have did the URL for that is https://github.com/mdhack0316/dockerfile and create multiple imsing existing image and verify whether it is working fine or not

Install the git:

```
[root@vishali ~]# yum install git -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 3.7 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package git.x86_64 0:2.32.0-1.amzn2.0.1 will be installed
--> Processing Dependency: perl-Git = 2.32.0-1.amzn2.0.1 for package: git-2.32
-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core-doc = 2.32.0-1.amzn2.0.1 for package: git-
32.0-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core = 2.32.0-1.amzn2.0.1 for package: git-2.32
-1.amzn2.0.1.x86_64
```

Clone the repo:

```
[root@vishali ~]# git clone https://github.com/mdhack0316/dockerfile
Cloning into 'dockerfile'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 29 (delta 3), reused 19 (delta 1), pack-reused 0
Receiving objects: 100% (29/29), 61.64 KiB | 20.55 MiB/s, done.
Resolving deltas: 100% (3/3), done.
```

Navigate inside normalwebsite:

```
[root@vishali ~]# ls
collins  dockerfile
[root@vishali ~]# cat dockerfile/
cat: dockerfile/: Is a directory
[root@vishali ~]# cd dockerfile/
[root@vishali dockerfile]# ls
maynak.txt  normalwebsite  pythoncodedocker  README.md
[root@vishali dockerfile]# cd normalwebsite/
[root@vishali normalwebsite]# ls
CODE_OF_CONDUCT.md  Dockerfile  images  index.html  LICENSE  READM
```

Understand the docker file:

```
[root@vishali normalwebsite]# cat Dockerfile
FROM oraclelinux:8.3
# Pulling OL docker image from Docker HUB
# If not present in Docker engine
MAINTAINER  mayank123modi@gmail.com
# contact me if any help required
RUN  dnf  install  httpd -y
# DNF is an installer of OL like YUM
COPY  . /var/www/html/
# COpy all the code / data from current location
# to /var/www/html/ in DOcker ENgine
CMD ["httpd","-DFOREGROUND"]
# default parent process is to start webserver
```

Build the image:

```
[root@vishali normalwebsite]# docker build -t imsing .
Sending build context to Docker daemon  72.19kB
Step 1/5 : FROM oraclelinux:8.3
 ---> 816d99f0bbe8
Step 2/5 : MAINTAINER  mayank123modi@gmail.com
 ---> Running in f341ca525e83
Removing intermediate container f341ca525e83
 ---> 5958f6df3c8b
Step 3/5 : RUN  dnf  install  httpd -y
 ---> Running in 03192839250e
Oracle Linux 8 BaseOS Latest (x86 64)            55 MB/s |  43 M
```

Create the container:

```
[root@vishali normalwebsite]# docker run -itd -p 4444:80 imsing
6553dcab51d4e8797569519baa5e3fde1b337237bafeb199d4849e7d9f9db6d3
[root@vishali normalwebsite]#
```

Q.3 use docker.io/library/httpd and change the content of index file by default the content is IT WORKS, but u need to change it and try to run and verify.
Dockerfile:

```
[root@vishali task3]# cat Dockerfile
FROM docker.io/library/httpd
EXPOSE 80
CMD ["httpd","-DFOREGROUND"]
```

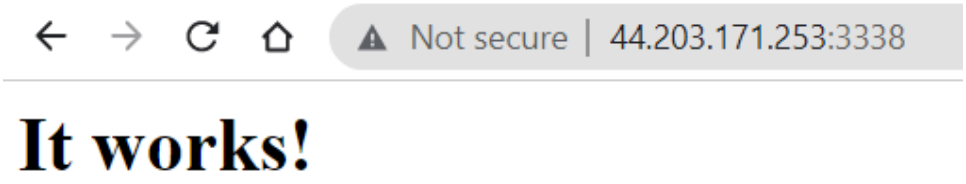
Build the image:

```
[root@vishali task3]# docker build -t httpdimage .
Sending build context to Docker daemon  2.048kB
Step 1/3 : FROM docker.io/library/httpd
 ---> c30a46771695
Step 2/3 : EXPOSE 80
 ---> Running in cbcdd35dd961
Removing intermediate container cbcdd35dd961
 ---> 62df5835537b
Step 3/3 : CMD ["httpd","-DFOREGROUND"]
 ---> Running in f78ba94ada3c
Removing intermediate container f78ba94ada3c
 ---> f078c9ad9969
Successfully built f078c9ad9969
Successfully tagged httpdimage:latest
```

Create the container:

```
[root@vishali task3]# docker run -itd --name httpdcont -p 3338:80 httpdimage
e96f3155a4c20c5875f3d2e9d966995db15ac01e0ba86ffb3407c6a53cb75612
```

Verify the website:



To change the content -get inside the container:

```
[root@vishali task3]# docker exec -it httpdcont bash
root@e96f3155a4c2:/usr/local/apache2# cd /var/www/html/
bash: cd: /var/www/html/: No such file or directory
root@e96f3155a4c2:/usr/local/apache2# ls
bin  build  cgi-bin  conf  error  htdocs  icons  include  logs  modules
```

Problem: Couldn't find out the index.html under the /var/www/html path

How to do we know where index.html file is present for this image?

```
[root@trainer test]# docker run -itd --name check1 httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
1fe172e4850f: Extracting [==================>                                ]  12.78MB/31.38MB
e2fa1fe9b1ec: Download complete
60dd7398e74e: Download complete
ea2ca81c6d4c: Download complete
f646c69a26ec: Download complete
```

Enter inside the container:

```
[root@trainer test]# docker exec -it check1 bash

root@63a7307c2896:/usr/local/apache2#
root@63a7307c2896:/usr/local/apache2#
root@63a7307c2896:/usr/local/apache2# pwd
/usr/local/apache2
root@63a7307c2896:/usr/local/apache2# ls
bin  build  cgi-bin  conf  error  htdocs  icons  include  logs  modules
root@63a7307c2896:/usr/local/apache2#
```

Locate command

```
root@63a7307c2896:/usr/local/apache2# locate index.html
bash: locate: command not found
root@63a7307c2896:/usr/local/apache2# locate index.htm
```

Find command:

```
root@63a7307c2896:/usr/local/apache2# find /   ndex.html
```

```
root@63a7307c2896:/usr/local/apache2# find /   -name  index.html
find: '/proc/10/map_files': Permission denied
find: '/proc/11/map_files': Permission denied
find: '/proc/12/map_files': Permission denied
/usr/local/apache2/htdocs/index.html
root@63a7307c2896:/usr/local/apache2# ca
```

```
root@63a7307c2896:/usr/local/apache2# cat /usr/local/apache2/htdocs/index.htm
<html><body><h1>It works!</h1></body></html>
root@63a7307c2896:/usr/local/apache2#
root@63a7307c2896:/usr/local/apache2#
```

Docker file:

```
FROM    docker.io/library/httpd

RUN     echo myContent > /usr/local/apache2/htdocs/index.html
```

Build the image:

```
[root@trainer test]# docker build -t  newhttpd .
Sending build context to Docker daemon  2.048kB
Step 1/2 : FROM    docker.io/library/httpd
 ---> c30a46771695
Step 2/2 : RUN     echo myContent > /usr/local/apache2/htdocs/index.html
 ---> Running in b966f0068646
Removing intermediate container b966f0068646
 ---> 153fb9ede00c
Successfully built 153fb9ede00c
Successfully tagged newhttpd:latest
```

Create the container:

```
[root@trainer test]# docker run -itd -p 4455:80  newhttpd
3a48d6f592bedca55290bbc74606ea97493a80e6179ae45081d6117b02da24f4
[root@trainer test]# cat
```

Q.4 use nginx image and change the content of this image using docker file. and then try to expose it on any port no. and verify the same

/usr/share/nginx/html/index.html

Find the location:

```
[root@vishali ~]# docker run -itd --name check1 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
1fe172e4850f: Already exists
35c195f487df: Pull complete
213b9b16f495: Pull complete
a8172d9e19b9: Pull complete
f5eee2cb2150: Pull complete
93e404ba8667: Pull complete
Digest: sha256:859ab6768a6f26a79bc42b231664111317d095a4f04e4b6fe79ce37b3d19
Status: Downloaded newer image for nginx:latest
c824441fbb31663ff7fb48e2911f7d60afc77a4ac740b5c109872a67b0bf7e53
```

Enter inside the container:

```
[root@vishali ~]# docker exec -it check1 bash
root@c824441fbb31:/# find / -name index.html
find: '/proc/33/map_files': Permission denied
/usr/share/nginx/html/index.html
root@c824441fbb31:/# ^C
root@c824441fbb31:/# exit
exit
```

Dockerfile:

```
[root@vishali task4]# cat Dockerfile
FROM nginx
RUN echo myContent from nginx > /usr/share/nginx/html/index.html
[root@vishali task4]#
```

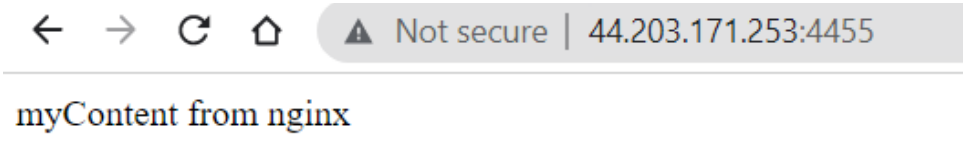
Build the image:

```
[root@vishali task4]# docker build -t newnginx .
Sending build context to Docker daemon  2.048kB
Step 1/2 : FROM nginx
 ---> fa5269854a5e
Step 2/2 : RUN echo myContent from nginx > /usr/share/nginx/html/index.html
 ---> Running in d56b7200aa9b
Removing intermediate container d56b7200aa9b
 ---> 20a73b7c11ab
Successfully built 20a73b7c11ab
Successfully tagged newnginx:latest
```

Run the container:

```
[root@vishali task4]# docker run -itd -p 4455:80 newnginx
2338eda4b3793cef7df7eb410d55292935acf288ce060cec8853bc35aad29de9
```

Verification:



Kubernetes:

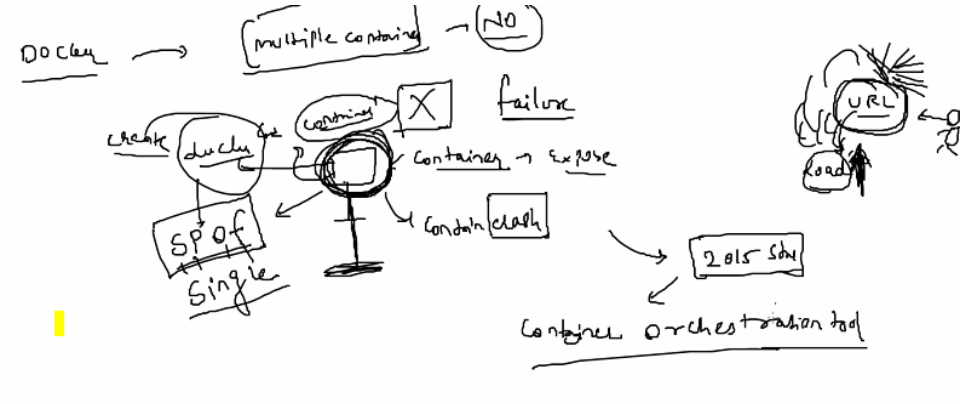
In docker, we can't create multiple containers at once.

Lot of request in the URL– Load increases, time to load increases

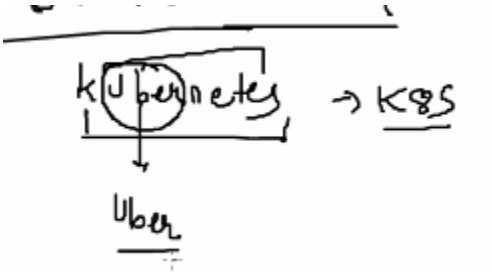
Container may crash -> Machine got crashed

Failure at the this website.

SPOF – Single point of Failure. In production, we can't create it.



Kubernetes – Orchestration Tool



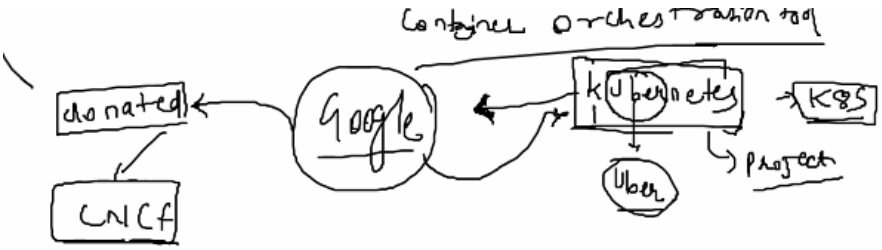
Docker is not supportive at the production.

Kubernetes

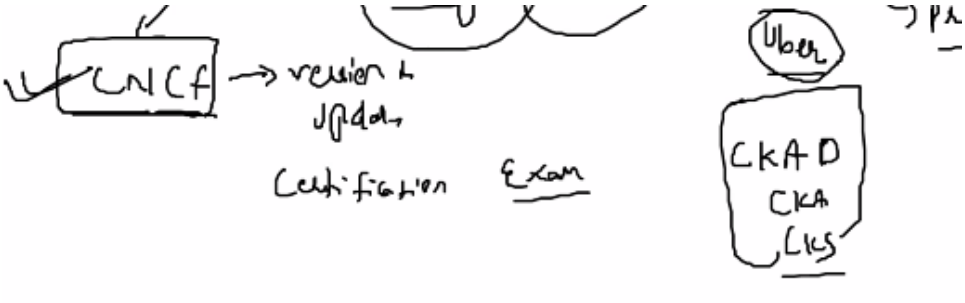
- Avoid SPOF
- Manage load
- Okay to have Multiple point of failures

Google created this project called Kubernetes in 2015 – to manage containers and remove SPOF.

Google donated to CNCF

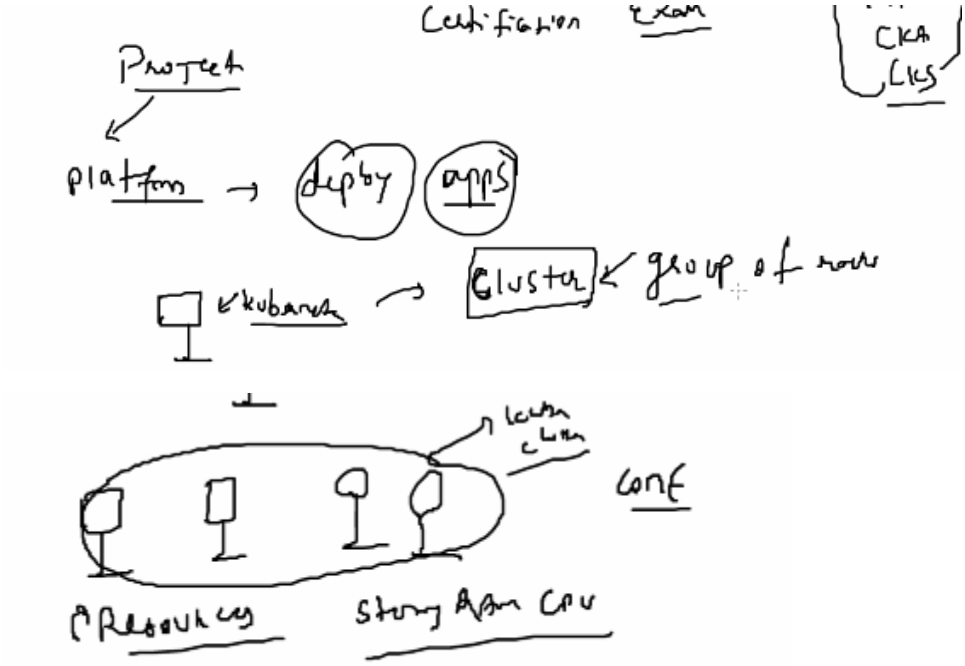


Certification exam – CKAD, CKA and CKS



Kubernetes is a tool where we can deploy our/ any time of application.

Cluster – group of machines working together.



Who will take care of the management?

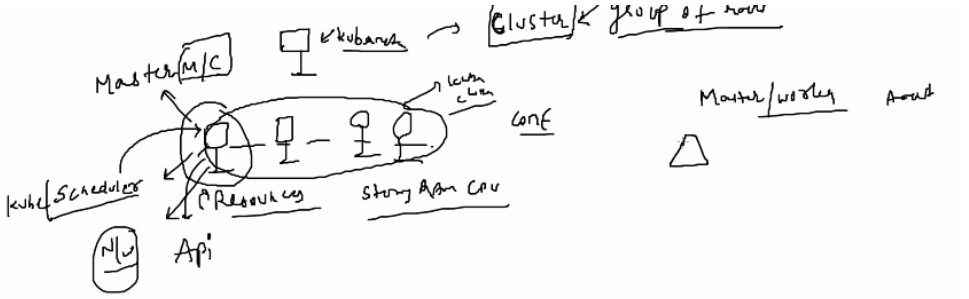
In which system the process run?

All these are taken care of Kubernetes

We will tell this machine is a master.

Who will schedule the process? kube scheduler

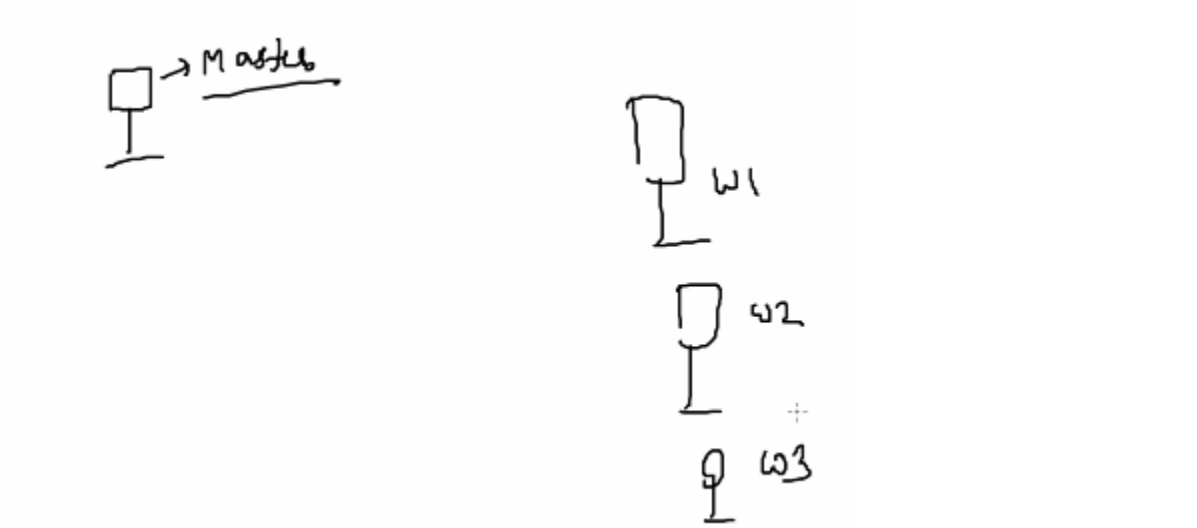
Master/Worker architecture:



Example:

New 4 machines is taken

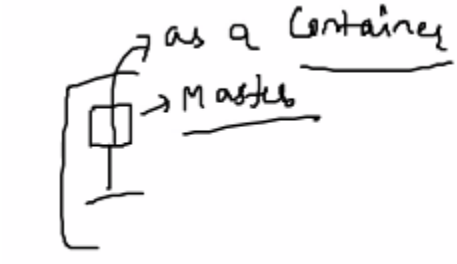
One machine will work as a master machine and remaining machines will be the worker.



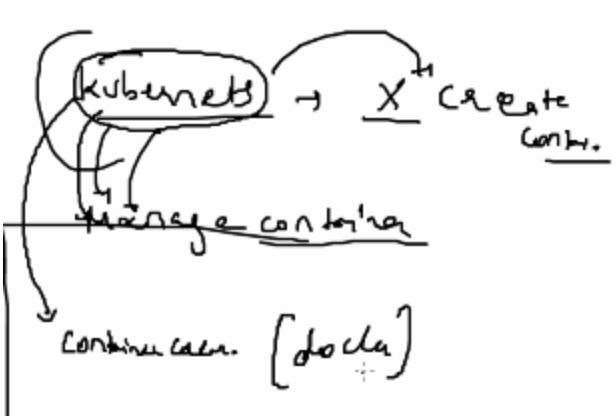
After deploying the application. One is called as master amchine and others are worker machines.

How will the architecture work?

Everything will run on the master machine as a container.

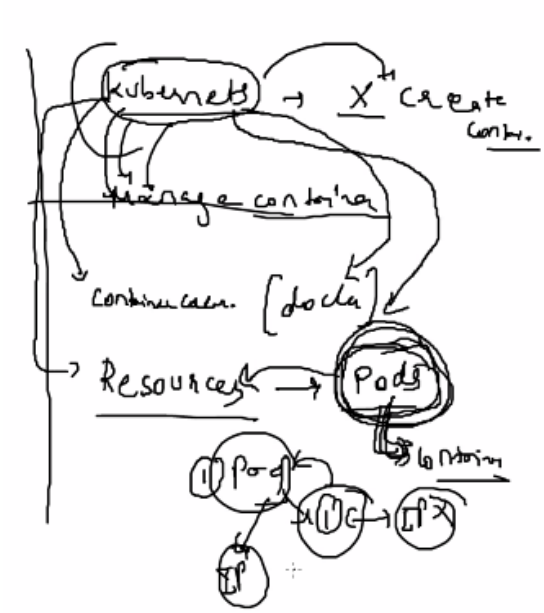


Kubernetes – won't help us to create the container. Only help us in managing the container.



Only docker will create the container. Kuberenetes will have multiple resources inside it. The smallest resources in the kuberenetes are called Pods. It just the virutal entity not a physical one.

Pod – resource help to create a container inside the Pod. It will get the IP address(own)



If I want to create a POD? How will I do it?

Use some commands and request will go to the master components.

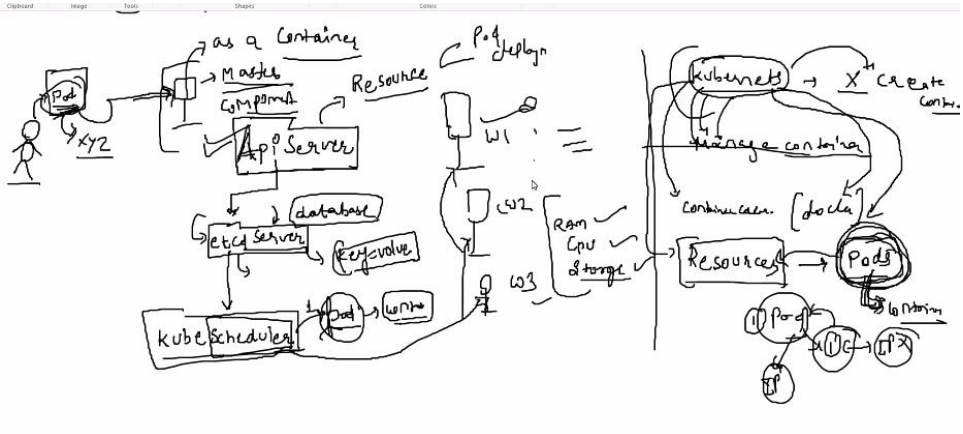
Which components? API server. API server is the one who is providing the resources like deployments and many more.

It will send my resources to new components called – etcd server

Why is my request going to etcd server? Because etcd server is the database of the kubernete's cluster. It has each and single information about the server in the form of key and value pair.

Why is my POD request going to etcd server? It will verify the pod name – already exists/ running or not.

They will again send request to kube scheduler: This will assign task to each machine.



If there is a Pacific Ocean – multiple ships are running? Who will maintain the details of ships? Captain

Like similar, for every machine we will have captain called kubelet.

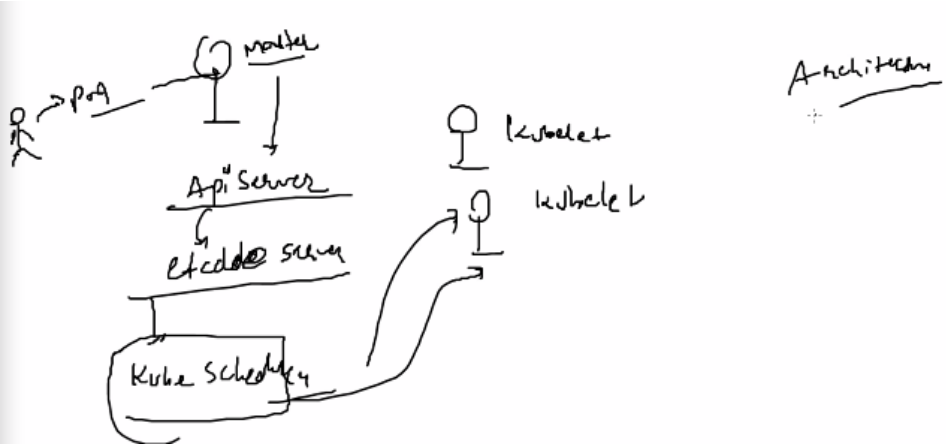
Schedule the tasks randomly in the containers.

If the worker 2 goes down, then what will happen? All the pods will be evited from the worker 2. Pod will create another machine and manages it.

Flow:

User -> Request -> Pod-> Master> API server -> etcd server -> kubescheduler -> Kubelet

Architecture:



On what basis, will the scheduling happen? RAM, storage. RAM will take the top priority.

How to create a Kubernetes cluster?

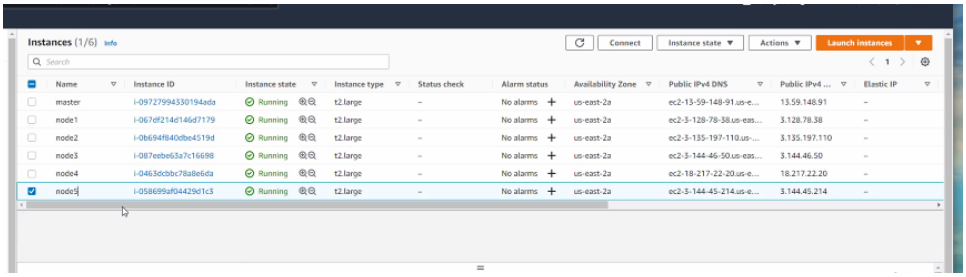
Why am I using AWS? We need multiple machines to create cluster – so using cloud platform.

Creating the machines:

First Create EC2 instance:

- 16GB RAM in the Master
- Configure instance details
- Add storage – 40 GB
- Add tags – no need
- Configure Security group
- Review instance Launch

Change the name to use it later. Node1, node2, node 3.....and so on

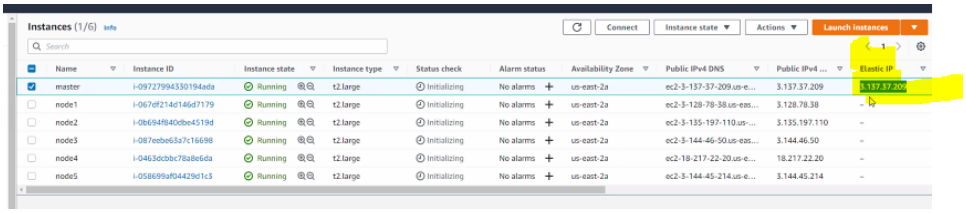


We need to give elastic IP:

What is an Elastic IP?

An Elastic IP address is **a reserved public IP address that you can assign to any EC2 instance in a particular region, until you choose to release it**.

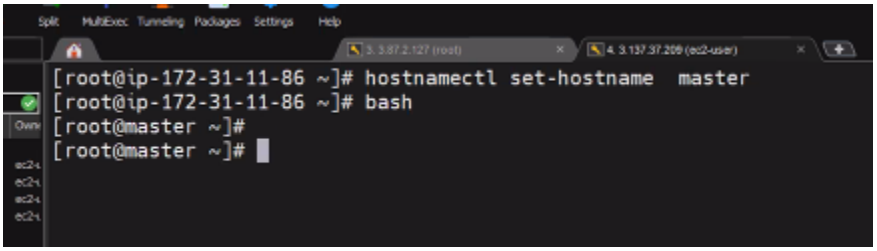
Static IP – Even if you reboot it. This IP will remain same.



Access each machine:

Connect to one machine and Assign as the master

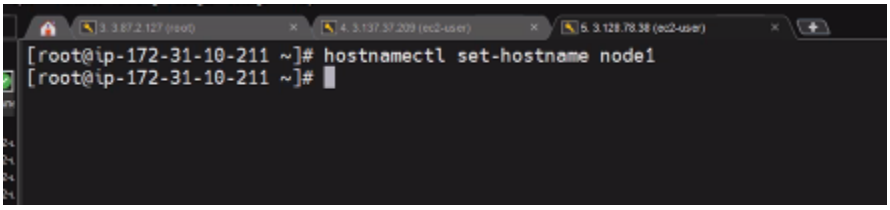
Master:



Set the hostname for reference purposes.

Connect to another machine:

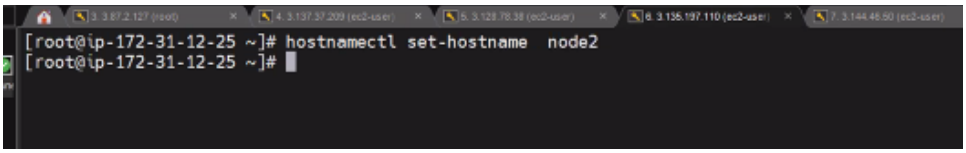
Node1:



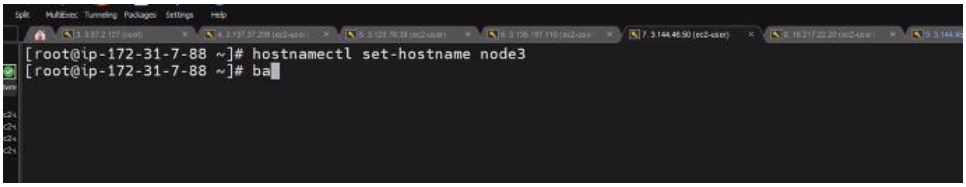
Set the hostname for reference purposes.

Access each machine one by one and give the hostname

Node 2:

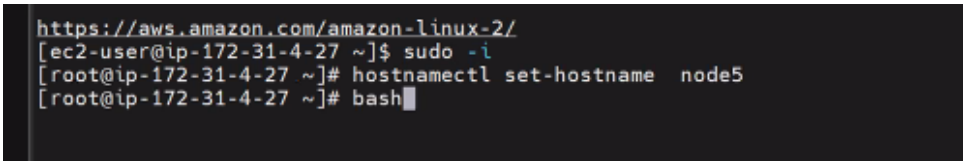


Node 3:



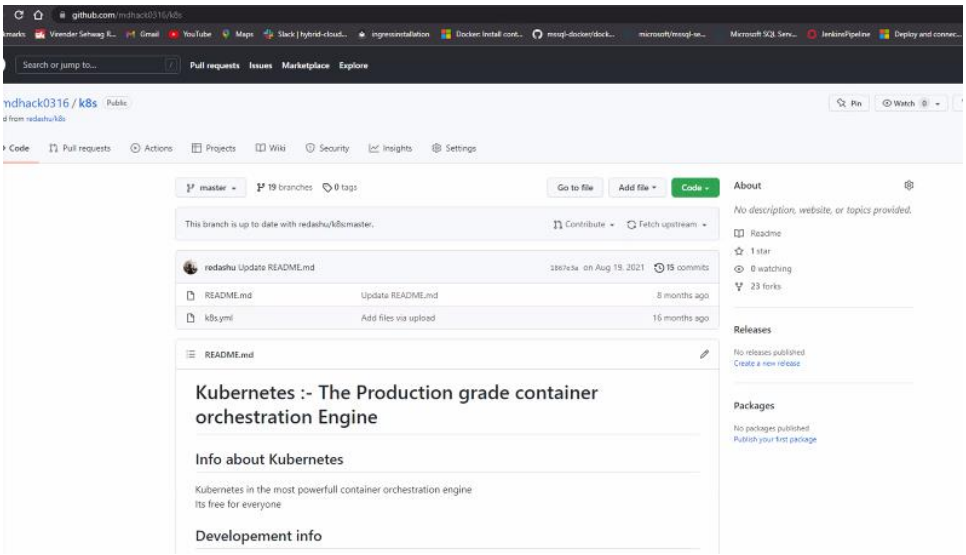
Node 4:

Node 5:

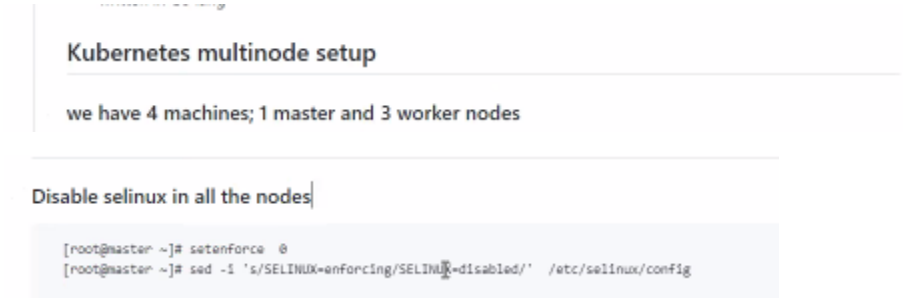


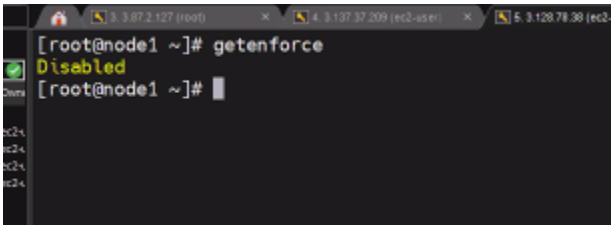
We have total 6 machines. 1 – master machine and other are worker machine.

Go to this link:



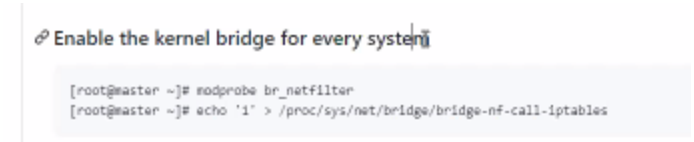
Create a cluster:





Check in every machine whether it is disabled. If not use the above command.

Step 2:

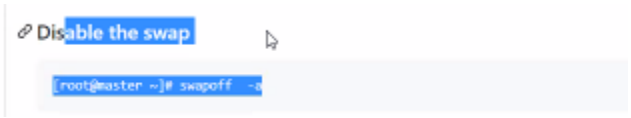


Why do we need this bridge? WE are going to build a kubernetes cluster with 6 machine. Kubernetes system will create a container. We have to take care of networking part. This kernel module will take care of it.

```
Disabled
[root@node1 ~]# modprobe br_netfilter
```

This will create network in every machine. Run it in every machine.

Step 3:



What is swap memory? In your machine you can have extra RAM. If RAM which are not using given to swap memory. This is not supported in Kubernetes. So, we are disabling it.

```
[root@master ~]# swapoff -a
[root@master ~]#
```

In every machine run this command.

Step 4:

What software we need?

```
[root@master ~]# <<X
>
>    docker -- yes
>   kubeadm  - yes
>  kubelet -- yes
>
> kubectl
```

All these software are required to create the kubernetes cluster.

We can’t install everything directly in linux:

```
[root@master ~]# yum install kubeadm kubelet docker kubectl -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                               | 3.7 kB  00:0
No package kubeadm available.
No package kubelet available.
No package kubectl available.
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 0:20.10.7-5.amzn2 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-20.10.7-5.amzn2.x86_64
--> Processing Dependency: libcgroup >= 0.40.rc1-5.15 for package: docker-20.10.7-5.amzn2.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-20.10.7-5.amzn2.x86_64
--> Processing Dependency: pigz for package: docker-20.10.7-5.amzn2.x86_64
--> Running transaction check
---> Package containerd.x86_64 0:1.4.6-8.amzn2 will be installed
---> Package libcgroup.x86_64 0:0.41-21.amzn2 will be installed
---> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
---> Package runc.x86_64 0:1.0.0-2.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                Arch                Version                  Repository
================================================================================
Installing:
```

Copy this in every machine

```
root@node2 ~]# swapoff -a
root@node2 ~]# cat  <<EOF  >/etc/yum.repos.d/kube.repo
[kube]
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
gpgcheck=0
EOF
root@node2 ~]# yum install kubeadm kubelet docker kubectl -y
```

Then run this command:

```
[root@master ~]# yum install kubeadm kubelet docker kubectl -y
```

```
[root@master ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: https://docs.docker.com
[root@master ~]# systemctl start docker
```

Docker info

```
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splu
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: d71fcd7d8303cbf684402823e425e9dd2e99285d
 runc version: 84113eef6fc27af1b01b3181f31bbaf708715301
 init version: de40ad0
 Security Options:
  seccomp
   Profile: default
```

Kuberentes doesn’t support cgroupfs driver. So, change it to system driver

```
[root@master ~]# cat  <<X  >/etc/docker/daemon.json
> {
>   "exec-opts": ["native.cgroupdriver=systemd"]
> }
>
> X
[root@master ~]# cat /etc/docker/daemon.json
```

```
[root@master ~]# cat /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}

[root@master ~]#
```

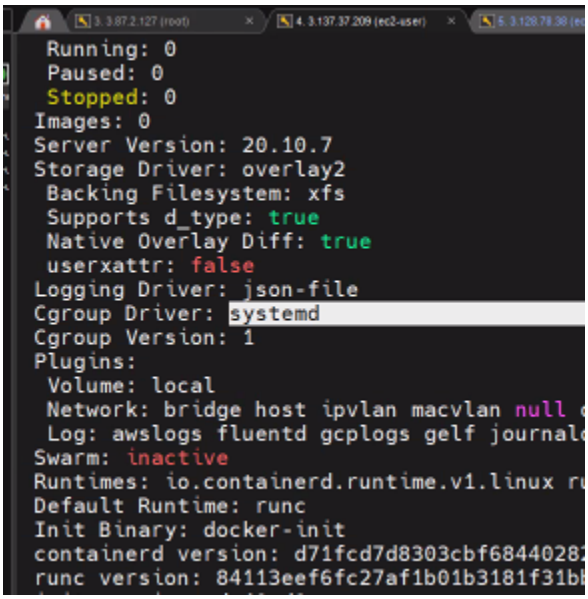
```
[root@master ~]# cat /etc/docker/daemon.json
[root@master ~]# cat /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"]
}

[root@master ~]#
```

Reload the daemon and restart the docker:

```
[root@master ~]# systemctl daemon-reload
[root@master ~]# systemctl restart docker
```

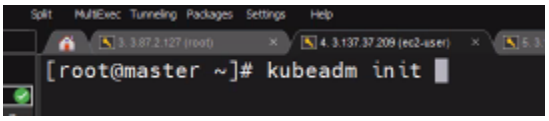
Docker info command:



This command you have to run in every single machine.

Above configuration needs to be completed in every machine.

Now we have to decide which machine is master and worker?



It will be initialized on private IP address – we won't access it

So, mention the Public IP address

When you create a kube cluster - > It will create a pod in the background – Mention the any private IP range for pod.

```
[root@master ~]# kubeadm init  --apiserver-cert-extra-sans=3.137.37.209  --apiserver-advertise-address=0.0.0.0
          --pod-network-cidr=192.168.0.0/24
```

This command will only run on the master machine.

It takes 2 – 3 minutes – depends on the internet speed. Since it downloads multiple images.

```
[root@master ~]# kubeadm init  --apiserver-cert-extra-sans=3.137.37.209  --apiserver-advertise-address=0.0.0.0
          --pod-network-cidr=192.168.0.0/24
[init] Using Kubernetes version: v1.23.6
[preflight] Running pre-flight checks
        [WARNING Service-Docker]: docker service is not enabled, please run 'systemctl enable docker.service'
        [WARNING FileExisting-tc]: tc not found in system path
        [WARNING Hostname]: hostname "master" could not be reached
        [WARNING Hostname]: hostname "master": lookup master on 172.31.0.2:53: no such host
[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 connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
```

Public IP Address:

```
[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 connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kubernetes.default.svc
es.default.svc.cluster.local master] and IPs [10.96.0.1 172.31.11.86 3.137.37.209]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master] and IPs [172.31.11.86 127.0.0.1
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master] and IPs [172.31.11.86 127.0.0.1 ::
```

Created successully:

```
[mark-control-plane] Marking the node master as control-plane by adding the taints [node-role.kubernetes.
Schedule]
[bootstrap-token] Using token: 9hir34.msvosn8vdfkh01p3
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to
rm certificate credentials
[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs fro
otstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certific
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane 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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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 worker nodes by running the following on each as root:

kubeadm join 172.31.11.86:6443 --token 9hir34.msvosn8vdfkh01p3 \
```

Authentication file

```
Your Kubernetes control-plane 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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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 worker nodes by running the following on each as root:

kubeadm join 172.31.11.86:6443 --token 9hir34.msvosn8vdfkh01p3 \
        --discovery-token-ca-cert-hash sha256:d7cd2db64e56153d5be65a0cbd62b530f7190e081012ab20594672d3060f26c0
[root@master ~]#
```

We need to move this file as the kubectl should know. It will only read from specific path.

```
[root@master ~]# kubectl get nodes
The connection to the server localhost:8080 was refused - did you specify the right host or port?
[root@master ~]#
```

```
[root@master ~]# kubectl get nodes
The connection to the server localhost:8080 was refused - did you specify the right host or port?
[root@master ~]# pwd
/root
[root@master ~]# mkdir .kube
[root@master ~]# cp /etc/kubernetes/admin.conf   .kube/config
[root@master ~]# kubectl get nodes
NAME     STATUS     ROLES                  AGE   VERSION
master   NotReady   control-plane,master   97s   v1.23.6
[root@master ~]#
```

No worker machines are added here. To add the worker machine, copy this command in every single machine.

```
Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.11.86:6443 --token 9hir34.msvosn8vdfkh01p3 \
        --discovery-token-ca-cert-hash sha256:d7cd2db64e56153d5be65a0cbd62b530f7190e081012ab2059467[illegible]2d3060f26c0
[root@master ~]# cat /etc/kubernetes/admin.conf
[root@master ~]# cat /etc/kubernetes/admin.conf
apiVersion: v1
```

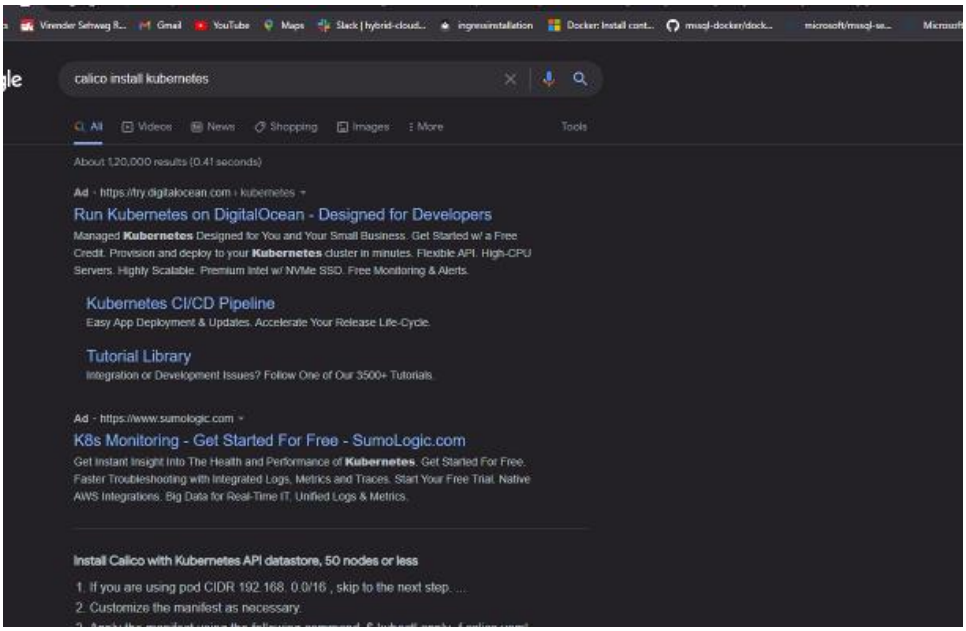
Node 1, node2, …..

```
[root@node1 ~]# kubeadm join 172.31.11.86:6443 --token 9hir34.msvosn8vdfkh01p3 \
>         --discovery-token-ca-cert-hash sha256:d7cd2db64e56153d5be65a0cbd62b530f7190e081012ab20594672d3060f26c0
[preflight] Running pre-flight checks
        [WARNING Service-Docker]: docker service is not enabled, please run 'systemctl enable docker.service'
        [WARNING FileExisting-tc]: tc not found in system path
        [WARNING Hostname]: hostname "node1" could not be reached
        [WARNING Hostname]: hostname "node1": lookup node1 on 172.31.0.2:53: no such host
```

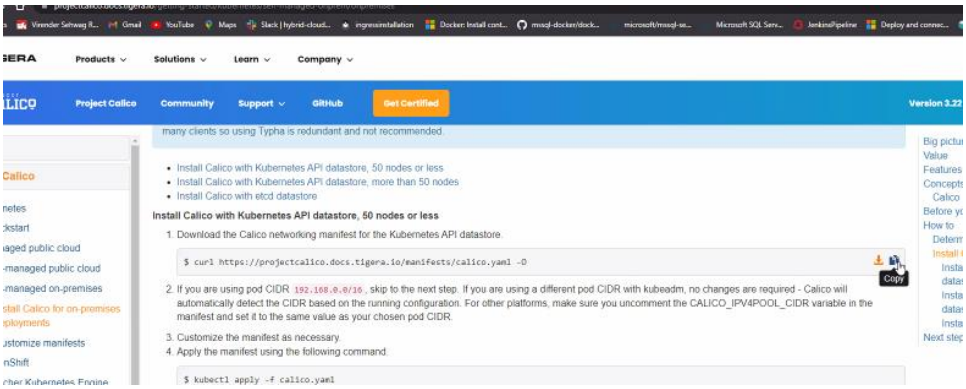
Now cluster is ready!

```
[root@master ~]# kubectl get nodes
NAME     STATUS     ROLES                  AGE     VERSION
master   NotReady   control-plane,master   4m17s   v1.23.6
node1    NotReady   <none>                 29s     v1.23.6
node2    NotReady   <none>                 33s     v1.23.6
node3    NotReady   <none>                 24s     v1.23.6
node4    NotReady   <none>                 21s     v1.23.6
node5    NotReady   <none>                 5s      v1.23.6
[root@master ~]#
```

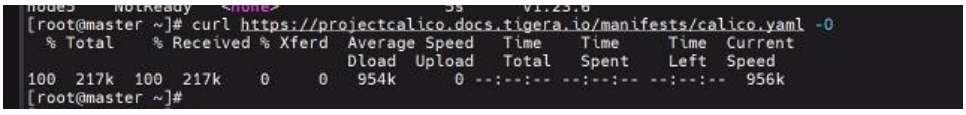
Machine status is not yet ready! Network configuring is not ready – best tool is calico networking – automating make them ready



Copy this command:



Paste it in the master machine:



View the calico.yaml file

Now apply with this kubectl:

```
[root@master ~]# kubectl  apply -f calico.yaml
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
serviceaccount/calico-node created
deployment.apps/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
Warning: policy/v1beta1 PodDisruptionBudget is deprecated in v1.21+, unavailable in v1.25+; use policy/v1 PodDi
onBudget
poddisruptionbudget.policy/calico-kube-controllers created
```

What is –w? K**ubernetes runs your workload by placing containers into Pods to run on Nodes**. A node may be a virtual or physical machine, depending on the cluster. Each node is managed by the control plane and contains the services necessary to run Pods.

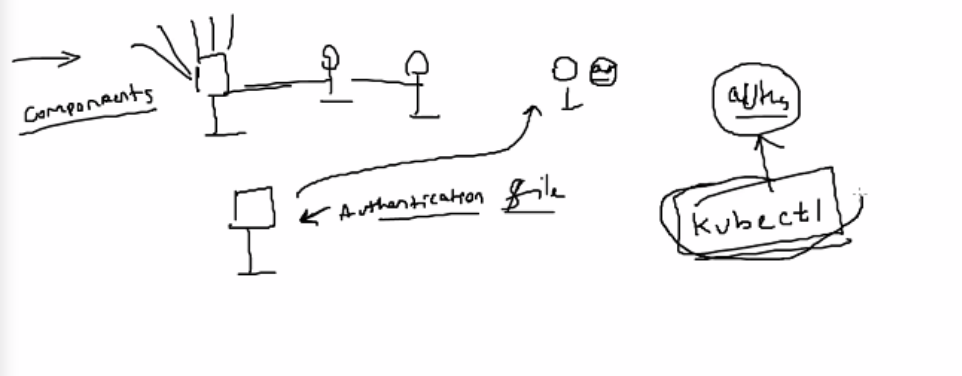
```
[root@master ~]# kubectl get nodes -w
NAME     STATUS     ROLES                  AGE     VERSION
master   NotReady   control-plane,master   5m51s   v1.23.6
node1    NotReady   <none>                 2m3s    v1.23.6
node2    NotReady   <none>                 2m7s    v1.23.6
node3    NotReady   <none>                 118s    v1.23.6
node4    NotReady   <none>                 115s    v1.23.6
node5    NotReady   <none>                 99s     v1.23.6
node3    NotReady   <none>                 2m2s    v1.23.6
node2    Ready      <none>                 2m12s   v1.23.6
node2    Ready      <none>                 2m12s   v1.23.6
```

Now everything is in ready state:

```
[root@master ~]# kubectl get nodes
NAME     STATUS   ROLES                  AGE     VERSION
master   Ready    control-plane,master   6m9s    v1.23.6
node1    Ready    <none>                 2m21s   v1.23.6
node2    Ready    <none>                 2m25s   v1.23.6
node3    Ready    <none>                 2m16s   v1.23.6
node4    Ready    <none>                 2m13s   v1.23.6
node5    Ready    <none>                 117s    v1.23.6
[root@master ~]#
```

Created the cluster but everyone can’t access it. Only the creator can access it for now.

How to access the cluster:



Using kubectl command:

cat <<EOF >/etc/yum.repos.d/kube.repo

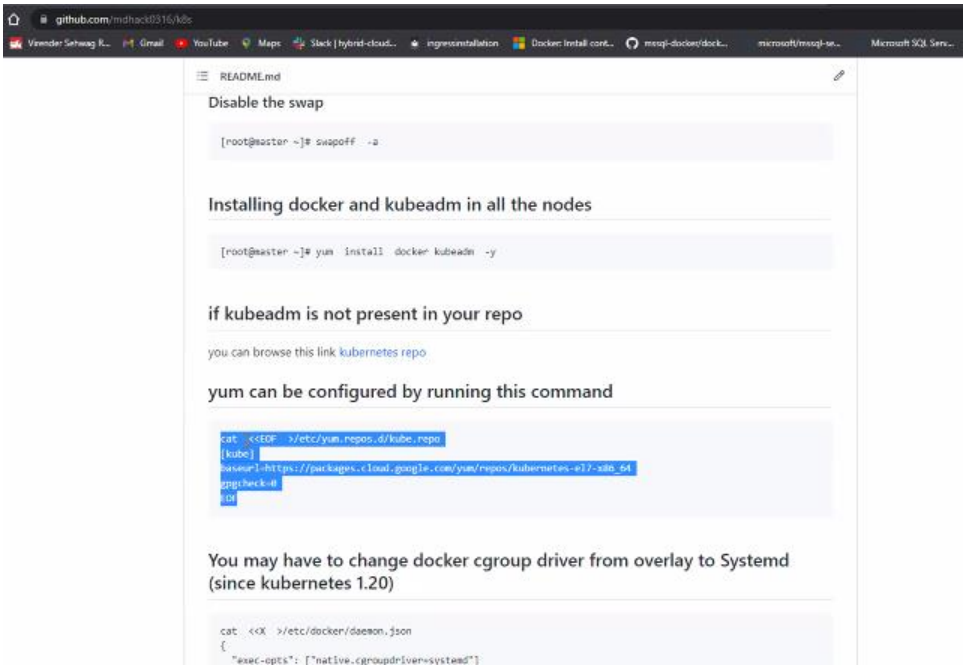
[kube]

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64

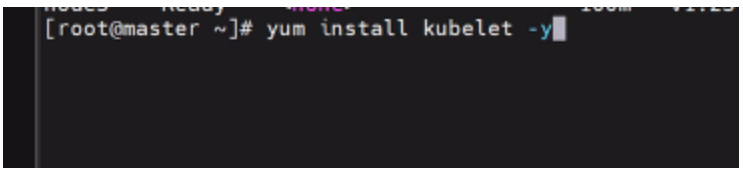
gpgcheck=0

EOF

Link: https://github.com/mdhack0316/k8s



Install kubectl using this command: yum install kubectl -y



Read the authentication file from home locations

```
[root@master ~]# pwd
/root
[root@master ~]# cd .kube/
[root@master .kube]# s
bash: s: command not found
[root@master .kube]# ls
cache  config
[root@master .kube]# cat con
[root@master .kube]# cat config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FU
kFRc0ZBREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJe
```

Mean time creator(trainer) is trying to provide us the authentication file

```
[root@master .kube]# cp config   /var/www/html/
[root@master .kube]# cd
[root@master ~]# chmod -R 777 /var/www/html/
[root@master ~]# cd /var/www/html/
[root@master html]# ls
config
[root@master html]# vim config
```

Help to get you trainer credentials in our machine:

 yum install wget -y ; wget 3.137.37.209/config ; mkdir $HOME/.kube/ ; cp config $HOME/.kube/ ; kubectl get nodes

Verify using this command: kubectl get nodes

```
[root@master html]# kubectl get nodes
NAME      STATUS   ROLES                  AGE    VERSION
master    Ready    control-plane,master   113m   v1.23.6
node1     Ready    <none>                 109m   v1.23.6
node2     Ready    <none>                 109m   v1.23.6
node3     Ready    <none>                 109m   v1.23.6
node4     Ready    <none>                 109m   v1.23.6
node5     Ready    <none>                 109m   v1.23.6
[root@master html]#
```

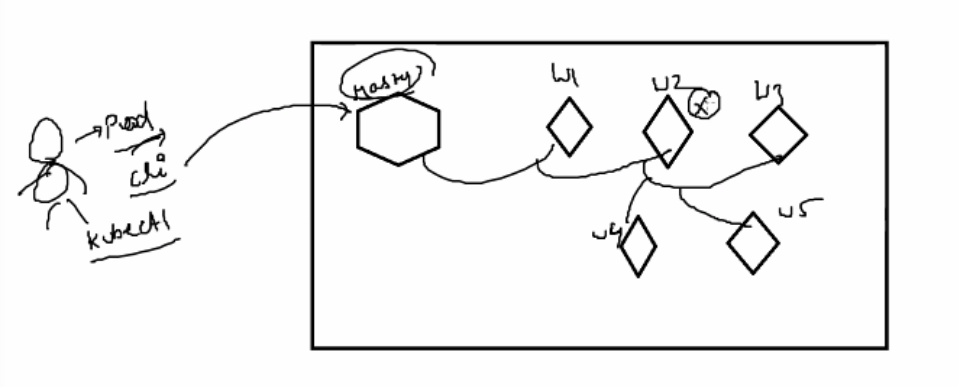
Start the httpd service:

```
[root@master html]# systemctl start httpd
[root@master html]#
```

What is the need of bash completion? Automation

kubectl completion bash > /etc/bash_completion.d/kubectl

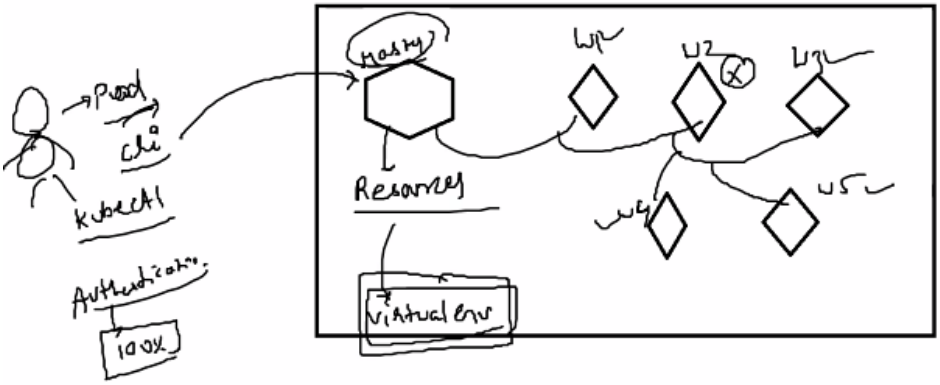
bash



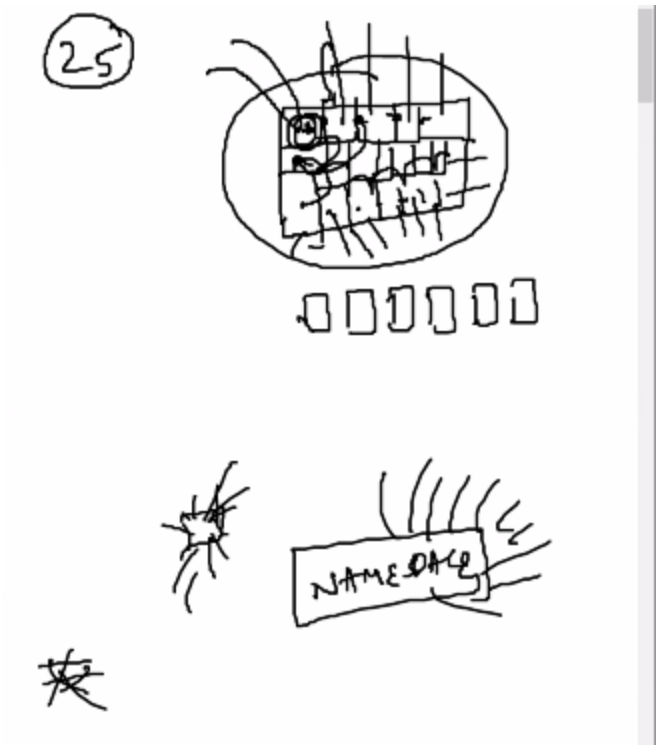
If the worker 2 goes down, where the process or container will go? Anywhere!

In Kubernetes – we have one great resource - what is that?

Virtual environment – whenever we create a cluster – it will be visible to everyone.



Example: Take my home – 25 rooms – you have a lock to lock the rooms. I have participants and assign to each room. Every participant is in each room.

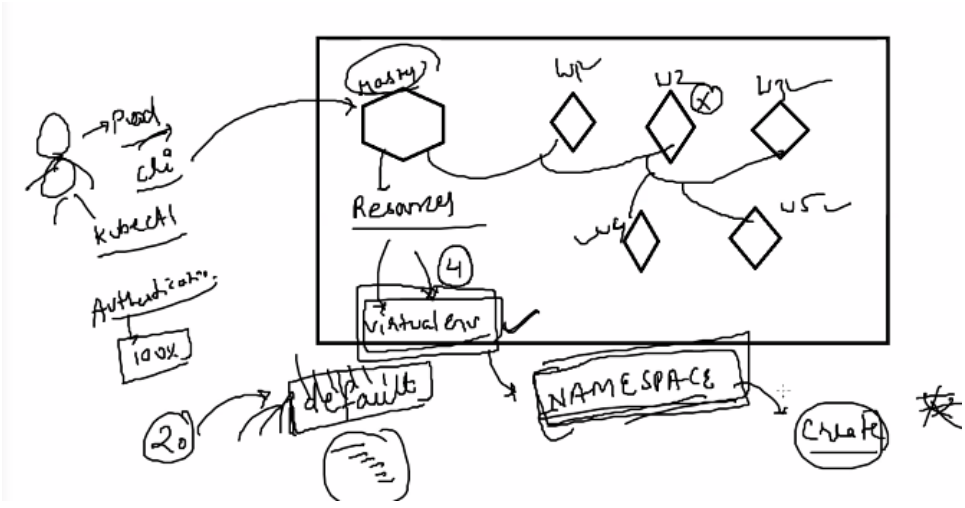


If I don’t lock, each participant can see what they are doing in their room.

If I lock, then each participant can’t see what other participant are doing?
Now, I have given access to everyone where everyone can see what others are doing.

Now I will try to create a personal room – You can create something.

That virtual environment in Kubernetes is called NAMESPACE
If there are multiple developers are there, then there won’t be any mix match in the environment. Everyone will have flexibility to develop their own code in their space.



What NAMESPACE is ? It’s a resources

Can we verify how many resource we have – Total 4

```
[root@master ~]# kubectl get namespace
NAME              STATUS   AGE
default           Active   124m
kube-node-lease   Active   125m
kube-public       Active   125m
kube-system       Active   125m
[root@master ~]#
```

Why default one is taken? The authentication we shared – there is a context name is kuberenetes – this by default using “deafult” as NAMESPACE.

```
[root@master ~]# kubectl get pod
No resources found in default namespace.
[root@master ~]#
```

But now I want to change the NAMESPACE:

```
[root@master ~]# kubectl create namespace mayank
namespace/mayank created
[root@master ~]#
```

Now you can see mayank namespace is created:

```
[root@master ~]# kubectl get ns
NAME              STATUS   AGE
default           Active   126m
kube-node-lease   Active   126m
kube-public       Active   126m
kube-system       Active   126m
mayank            Active   6s
[root@master ~]#
```

Get pods:

```
[root@master ~]# kubectl get pods
No resources found in default namespace.
[root@master ~]#
```

You haven’t told to the context to use mayank as a namespace instead of default

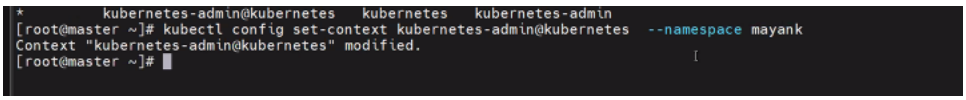
How to change the context?

```
[root@master ~]# kubectl config
current-context  delete-user      get-users        set-cluster      unset
delete-cluster   get-clusters     rename-context   set-context      use-context
delete-context   get-contexts     set              set-credentials  view
[root@master ~]# kubectl config
```

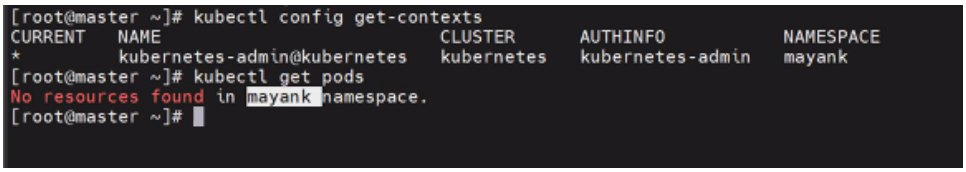
Get context

```
[root@master ~]# kubectl config get-contexts
CURRENT   NAME                          CLUSTER      AUTHINFO           NAMESPACE
*         kubernetes-admin@kubernetes   kubernetes   kubernetes-admin
[root@master ~]#
```

Set context

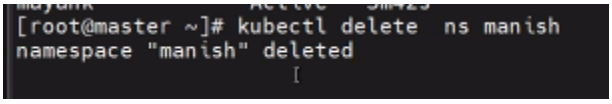


Get context



Get pods to verify it:

Delete the namespace:



Commands shared by trainer:

65 kubectl completion bash > /etc/bash_completion.d/kubectl

66 bash

67 history

68 kubectl get namespace

69 kubectl get pod

70 vim .kube/config

71 kubectl get pod

72 kubectl create namespace mayank

73 kubectl get bs

74 kubectl get ns

75 kubectl get pods

76 bash

77 history

2 kubectl config get-contexts

3 kubectl config set-context kubernetes-admin@kubernetes --namespace mayank

4 kubectl config get-contexts

5 kubectl get pods

6 kubectl get ns

7 kubectl get pods

Smallest resources in the Kubernetes? POD

What pod will create? Container

How to check pods? Kubectl get pods

```
[root@master ~]# kubectl get pods
No resources found in mayank namespace.
[root@master ~]#
```

How to create the pod? Declarative method

CLI – Command line Interface

YAML – Yet another markup language

```
[root@master ~]# kubectl  run   test1  --image httpd
pod/test1 created
[root@master ~]# kube
```

```
[root@master ~]# kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
test1   0/1     ContainerCreating   0          4s
[root@master ~]#
```

```
[root@master ~]# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
test1   1/1     Running   0          15s
```

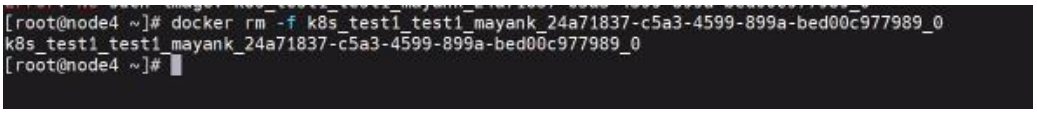
```
[root@master ~]# kubectl get nodes
NAME      STATUS   ROLES                  AGE    VERSION
master    Ready    control-plane,master   136m   v1.23.6
node1     Ready    <none>                 132m   v1.23.6
node2     Ready    <none>                 132m   v1.23.6
node3     Ready    <none>                 132m   v1.23.6
node4     Ready    <none>                 132m   v1.23.6
node5     Ready    <none>                 132m   v1.23.6
[root@master ~]#
```

Which machine it is running?

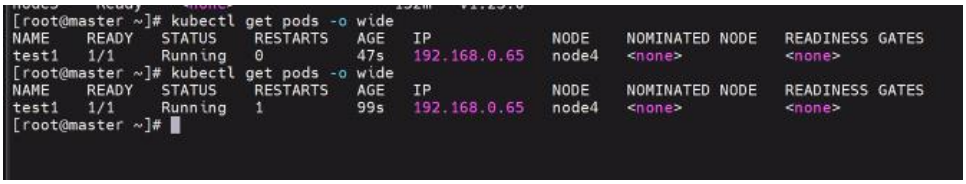
```
[root@master ~]# kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP              NODE    NOMINATED NODE   READINESS GATES
test1   1/1     Running   0          47s   192.168.0.65    node4   <none>           <none>
[root@master ~]#
```

In machine 4?

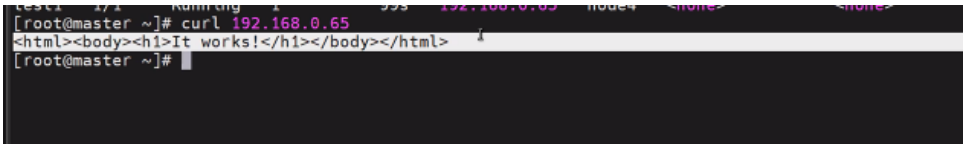
Intentionally delete it



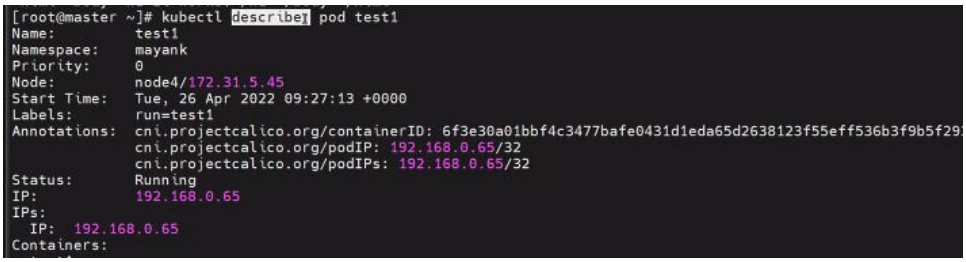
Restart:



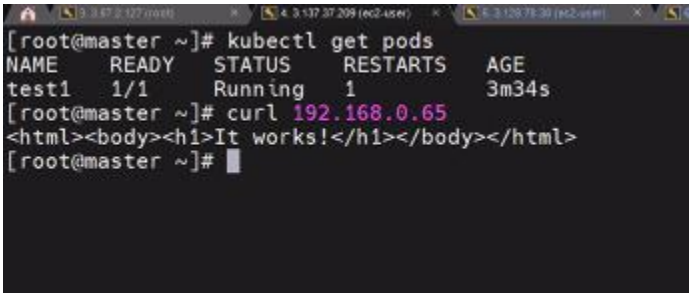
We can’t do this, as it is not in master node. As it is in different network.



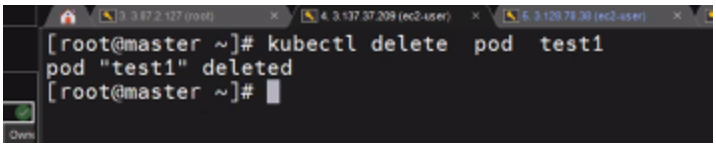
To know the information about the pod:



Verification:

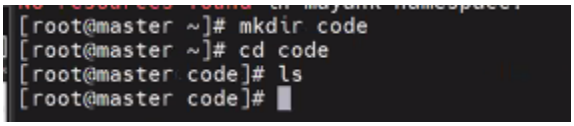


Delete the pod:

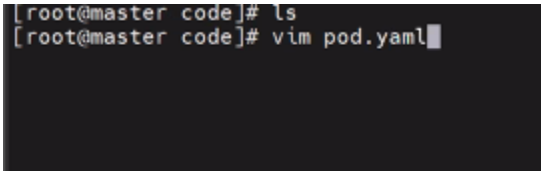


How to create a pod using yaml file?

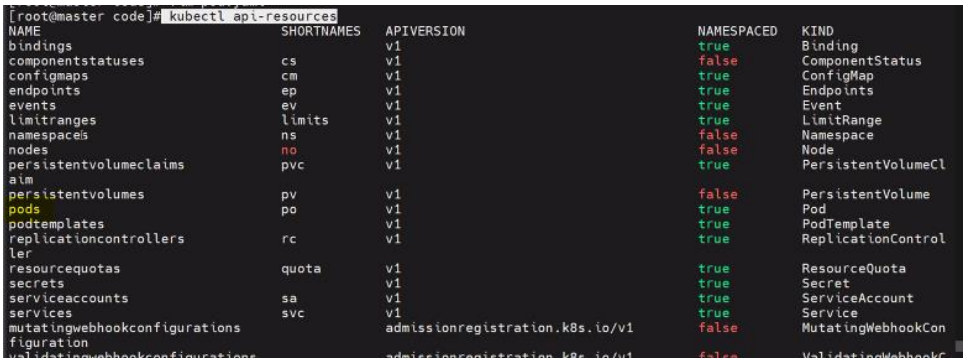
Create a folder



Create and edit the yaml file:



Api-resources:



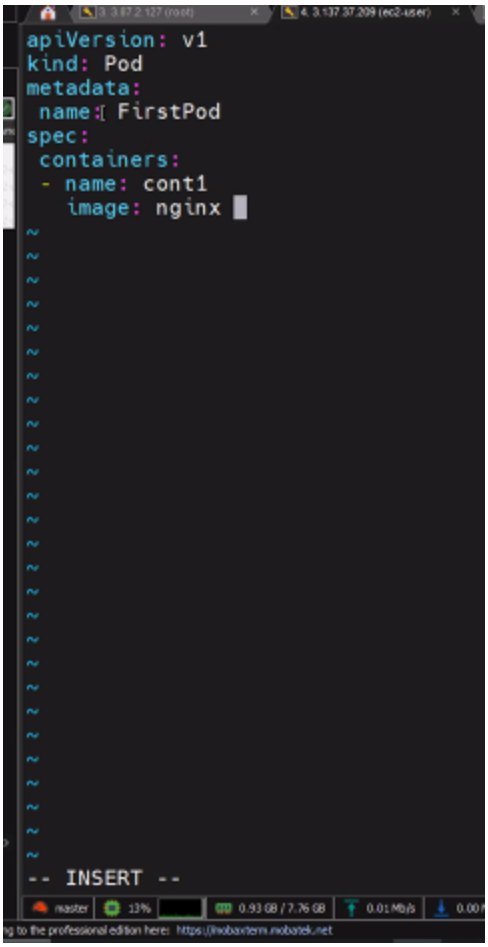
Yaml is a decalartive language:

You can kind the apiversion and kind from above

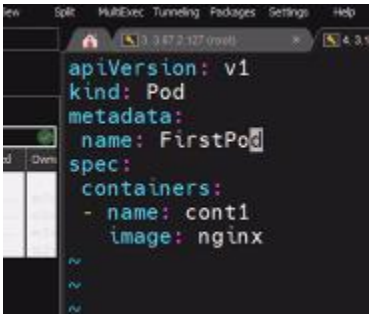
Metadata – some information about the pod

Intention – spacing should be proper – 2 or 3 spaces – same should be followed everywhere

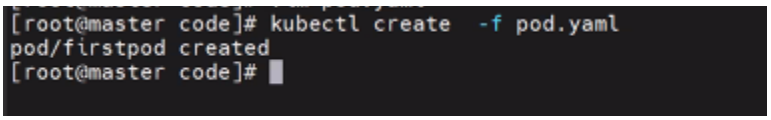
Specification:Write about containers section: name, how many images, We can give multiple values. (hypen is the separator)



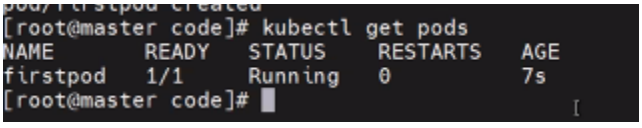
Create the pod using yaml file:



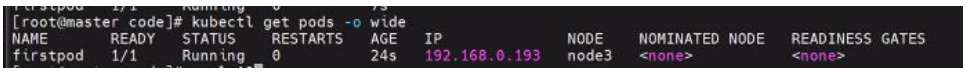
Problem here is the pod name should not be in caps. Make it lower “firstpod”



Get pod:



Get pods wide:

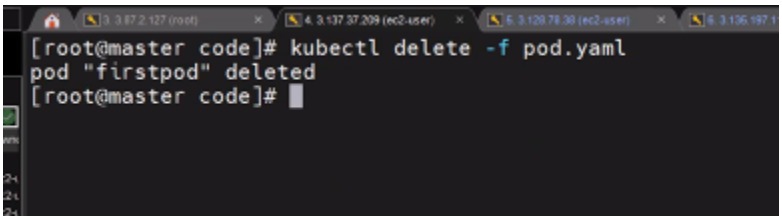


Curl command:

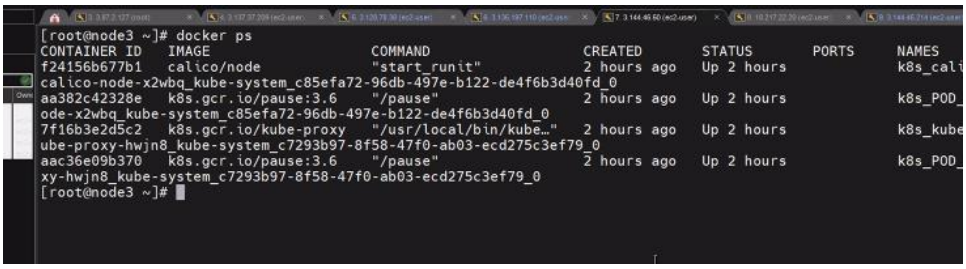
curl 192.168.0.193

You can check by docker ps:

Delete the pod:



In machine 3 – it is not visible:




```
 5  kubectl get pods
 6  kubectl get ns
 7  kubectl  get pods
 8  history
 9  kubectl get ns
10  kubectl delete  ns manish
11  kubectl get ns
12  kubectl get pods
```

```
13  kubectl get namespaces
14  kubectl get pods
15  kubectl  run   test1  --image httpd
16  kubectl get pods
17  kubectl get nodes
18  kubectl get pods -o wide
19  curl 192.168.0.65
20  kubectl describe  pod test1
21  kubectl get pods
22  curl 192.168.0.65
23  kubectl delete  pod  test1
24  kubectl get pods
25  kubectl get pods  --all-namespaces
26  kubectl get pods
27  mkdir code
28  cd code
29  ls
30  vim pod.yaml
31  kubectl api-resources
32  vim pod.yaml
33  kubectl get pods
34  kubectl create  -fpod.yaml
35  kubectl create  -f pod.yaml
36  vim pod.yaml
37  kubectl create  -f pod.yaml
38  kubectl get pods
39  kubectl get pods -o wide
40  curl 192.168.0.193
41  kubectl describe  pod firstpod
42  kubectl delete -f pod.yaml
```

43 history

Yaml file:

```
apiVersion: v1
kind: Pod
metadata:
 name: firstpod
spec:
 containers:
 - name: cont1
   image: nginx
```

TASK:

http://54.159.140.24/kube/task1.php

○	Q.1 Create A namespace with your own name.
○	Q.2 Modify your context to use your name namespace.
○	Q.3 Create a Pod using httpd image and name of POD should be pod1.
○	Q.4 Create a Pod using nginx image and name of POD should be pod2.
○	Q.5 Create a Pod using yaml method , and use nginx image and name of POD should be yamlpod1.
○	Q.6 Create a Pod using yaml method , and use mdhack/myserver image and name of POD should be yamlpod2.

Task:

Question:

Q.1 Create a namespace with your own name.

Get namespace:

```
[root@vishali ~]# kubectl get namespace
NAME              STATUS   AGE
default           Active   153m
kube-node-lease   Active   154m
kube-public       Active   154m
kube-system       Active   154m
manish            Active   3m4s
manoj             Active   6s
mayank            Active   27m
praval            Active   10s
sampat            Active   33s
sankalp           Active   36s
sashi             Active   51s
vaishnavi         Active   48s
```

Get pod:

```
[root@vishali ~]# kubectl get pod
No resources found in default namespace.
```

Create the namespace:

```
[root@vishali ~]# kubectl create namespace vishali
namespace/vishali created
```

Verify it:

```
[root@vishali ~]# kubectl get ns
NAME              STATUS   AGE
alpana            Active   3s
default           Active   154m
kaustubh          Active   28s
kube-node-lease   Active   154m
kube-public       Active   154m
kube-system       Active   154m
manish            Active   4m
manoj             Active   62s
mayank            Active   28m
praval            Active   66s
sampat            Active   89s
sankalp           Active   92s
sashi             Active   107s
shweta            Active   40s
vaishnavi         Active   104s
vishali           Active   9s
```

But pods has got default namespace, we need to change the context

```
[root@vishali ~]# kubectl get pods
No resources found in default namespace.
```

Q.2 Modify your context to use your name namespace.

Get context:

```
[root@vishali ~]# kubectl config get-contexts
CURRENT   NAME                          CLUSTER      AUTHINFO           NAMESPAC
E
*         kubernetes-admin@kubernetes   kubernetes   kubernetes-admin
```

Set the context:

```
[root@vishali ~]# kubectl config set-context kubernetes-admin@kubernetes --names
pace vishali
Context "kubernetes-admin@kubernetes" modified.
```

Again, get the context to verify:

```
[root@vishali ~]# kubectl config get-contexts
CURRENT   NAME                          CLUSTER      AUTHINFO           NAMESPAC
E
*         kubernetes-admin@kubernetes   kubernetes   kubernetes-admin   vishali
```

Get the pods:

```
[root@vishali ~]# kubectl get pods
No resources found in vishali namespace.
[root@vishali ~]# kubectl get ns
NAME              STATUS   AGE
alpana            Active   2m31s
default           Active   157m
kaustubh          Active   2m56s
kube-node-lease   Active   157m
kube-public       Active   157m
kube-system       Active   157m
manish            Active   6m28s
manoj             Active   3m30s
mayank            Active   30m
praval            Active   3m34s
sampat            Active   3m57s
sangamesh         Active   2m22s
sankalp           Active   4m
sashi             Active   4m15s
shweta            Active   3m8s
vaishnavi         Active   4m12s
vishali           Active   2m37s
```

Q.3 Create a Pod using httpd image and name of POD should be pod1.

```
[root@vishali ~]# kubectl run pod1 --image httpd
pod/pod1 created
[root@vishali ~]# kubectl get pods
NAME   READY   STATUS    RESTARTS   AGE
pod1   1/1     Running   0          7s
[root@vishali ~]# kubectl get nodes
NAME     STATUS   ROLES                  AGE    VERSION
master   Ready    control-plane,master   164m   v1.23.6
node1    Ready    <none>                 160m   v1.23.6
node2    Ready    <none>                 160m   v1.23.6
node3    Ready    <none>                 160m   v1.23.6
node4    Ready    <none>                 160m   v1.23.6
node5    Ready    <none>                 159m   v1.23.6
[root@vishali ~]# kubectl get pods -o wide
NAME   READY   STATUS    RESTARTS   AGE   IP              NODE    NOMINATED NODE
   READINESS GATES
pod1   1/1     Running   0          39s   192.168.0.133   node1   <none>
   <none>
```

Q.4 Create a Pod using nginx image and name of POD should be pod2.

```
[root@vishali ~]# kubectl run pod2 --image nginx
pod/pod2 created
[root@vishali ~]# kubectl get pods
NAME   READY   STATUS    RESTARTS   AGE
pod1   1/1     Running   0          4m41s
pod2   1/1     Running   0          10s
[root@vishali ~]# kubectl get pods -o wide
NAME   READY   STATUS    RESTARTS   AGE     IP              NODE    NOMINATED NO
DE   READINESS GATES
pod1   1/1     Running   0          4m58s   192.168.0.133   node1   <none>
     <none>
pod2   1/1     Running   0          27s     192.168.0.201   node3   <none>
     <none>
[root@vishali ~]#
```

Q.5 Create a Pod using yaml method and use nginx image and name of POD should be yamlpod1.

Create a folder:

```
[root@vishali ~]# mkdir code
[root@vishali ~]# cd code
[root@vishali code]# vim pod.yaml
```

Yaml file:

```
[root@vishali code]# cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: yamlpod1
spec:
  containers:
    - name: cont1
      image: nginx
```

Create the pod using that yaml file:

```
[root@vishali code]# kubectl create -f pod.yaml
pod/yamlpod1 created
```

Q.6 Create a Pod using yaml method and use mdhack/myserver image and name of POD should be yamlpod2.

Yaml file:

```
[root@vishali code]# cat pod2.yaml
apiVersion: v1
kind: Pod
metadata:
  name: yamlpod2
spec:
  containers:
    - name: cont2
      image: mdhack/myserver
```

Create the pod using that yaml file:

```
[root@vishali code]# kubectl create -f pod2.yaml
pod/yamlpod2 created
```

Kubectl get pods:

```
[root@vishali code]# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
pod1       1/1     Running   0          15m
pod2       1/1     Running   0          10m
yamlpod1   1/1     Running   0          5m34s
yamlpod2   1/1     Running   0          83s
```

Kubectl get pods –o wide:

```
yamlpod2   1/1     Running   0          83s
[root@vishali code]# kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE     IP              NODE    NOMINATE
D NODE   READINESS GATES
pod1       1/1     Running   0          15m     192.168.0.133   node1   <none>
         <none>
pod2       1/1     Running   0          11m     192.168.0.201   node3   <none>
         <none>
yamlpod1   1/1     Running   0          5m59s   192.168.0.203   node3   <none>
         <none>
yamlpod2   1/1     Running   0          108s    192.168.0.140   node1   <none>
         <none>
[root@vishali code]#
```

If there is no pod entity, what is the other methods? Virtual environment.

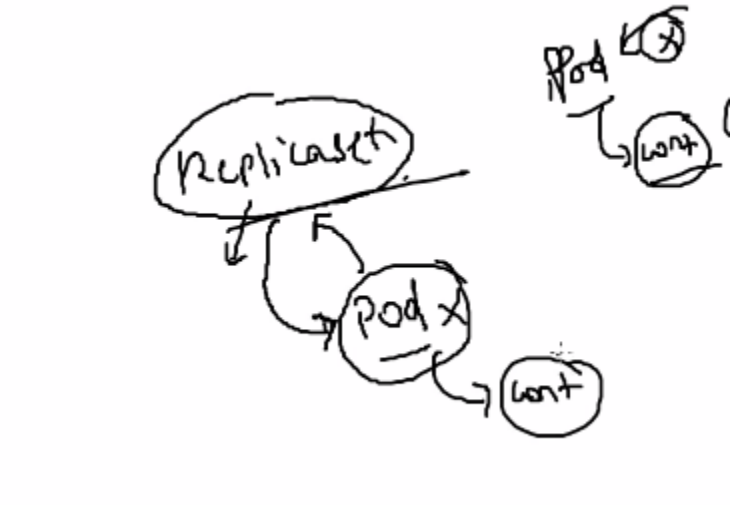
If we create a docker container and delete it? Will it come back? No

Same thing is with pod. It will never come back.

But as the part of high availability part, it should come back. We need to create another feature – resource called deployment. Without down time it can be changed

Pod is an entity, but we are not going to be created automatically.

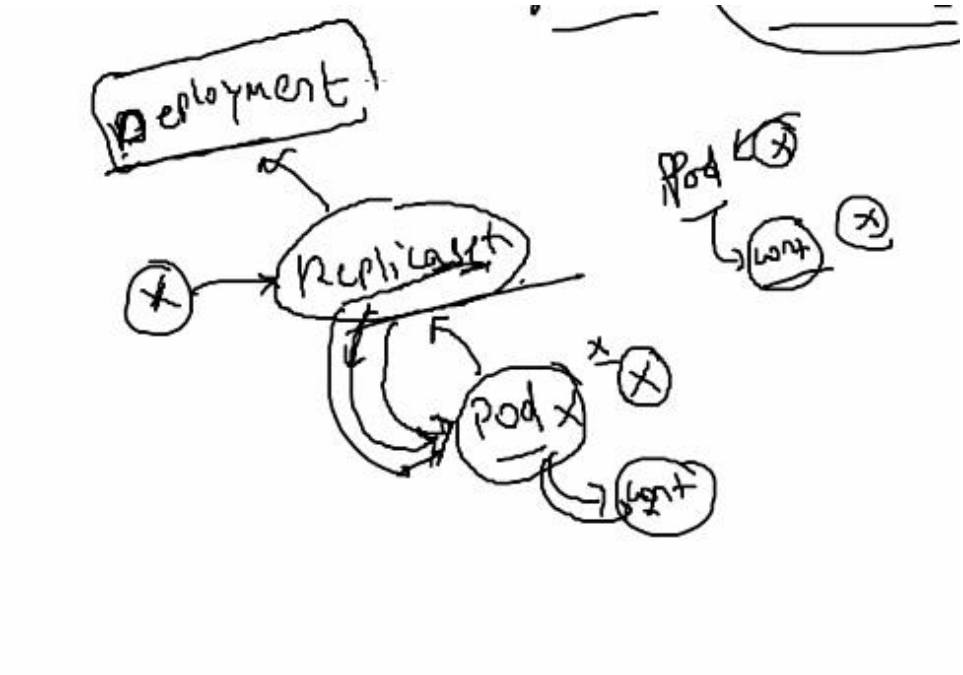
We have replica set – help us to create pods automatically:



Pod will create container.

If you delete the pod, container will delete. What replica will ensure? It will ensure pod will automatically come back.

Replica we can use it but it can't have one feature? What is that?



When deployment is created, then replica set will be created. If the replica is deleted, then deployment is created automatically. You can create n number of replica set. One replica set can have multiple pods.

Deployment: It can be created by cli and yaml.

```
[root@master code]# kubectl get all -n default
NAME           READY   STATUS             RESTARTS   AGE
pod/firstpod   1/1     Running            0          44m
pod/pod1       1/1     Running            0          53m
pod/pod2       0/1     ImagePullBackOff   0          52m
pod/pod22      1/1     Running            0          43m
pod/pod3       0/1     ImagePullBackOff   0          51m
pod/yamlpod1   1/1     Running            0          42m
pod/yamlpod2   1/1     Running            0          41m

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   3h39m
[root@master code]# kubecget ns
bash: kubecget: command not found
```

To know the deployments created:

```
[root@master code]# kubectl get deployments.apps
No resources found in mayank namespace.
[root@master code]#
```

get deployments: (not necessary to use .apps)

```
[root@master code]# kubectl get deployments.apps
No resources found in mayank namespace.
[root@master code]# kubectl get deployments
No resources found in mayank namespace.
[root@master code]#
```

get deployments, pods, replicas

```
[root@master code]# kubectl get deployments,pods,rs
No resources found in mayank namespace.
[root@master code]#
```

get all:

```
[root@master code]# kubectl get all
No resources found in mayank namespace.
[root@master code]#
```

Using CLI:

create deployment:

```
[root@master code]# kubectl create  deployment mayank --image=nginx
deployment.apps/mayank created
[root@master code]#
```

get all:

get all -o wide:

```
[root@master code]# kubectl get all -o wide
NAME                         READY   STATUS    RESTARTS   AGE   IP             NODE    NOMINATED NODE   READINESS GAT
ES
pod/mayank-9c4f4b955-ctdcl   1/1     Running   0          22s   192.168.0.31   node5   <none>           <none>

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES   SELECTOR
deployment.apps/mayank   1/1     1            1           22s   nginx        nginx    app=mayank

NAME                               DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES   SELECTOR
replicaset.apps/mayank-9c4f4b955   1         1         1       22s   nginx        nginx    app=mayank,pod-template-ha
sh=9c4f4b955
[root@master code]#
```

Verify:

```
[root@master code]# curl  192.168.0.31
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
```

Delete the pod – it will create the pod automatically

```
[root@master code]# kubectl delete  pod mayank-9c4f4b955-ctdcl  --force
warning: Immediate deletion does not wait for confirmation that the running resource has been terminate
e may continue to run on the cluster indefinitely.
pod "mayank-9c4f4b955-ctdcl" force deleted
[root@master code]# kubectl get all
NAME                            READY   STATUS              RESTARTS   AGE
pod/mayank-9c4f4b955-lcb48      0/1     ContainerCreating   0          2s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mayank     0/1     1            0           89s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/mayank-9c4f4b955      1         1         0       89s
[root@master code]#
```

verify it:

```
[root@master code]# kubectl get all  -o wide
NAME                            READY   STATUS    RESTARTS   AGE   IP              NODE    NOMINATED NODE   READINESS GA
TES
pod/mayank-9c4f4b955-lcb48      1/1     Running   0          18s   192.168.0.208   node3   <none>           <none>

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE    CONTAINERS   IMAGES   SELECTOR
deployment.apps/mayank     1/1     1            1           105s   nginx        nginx    app=mayank

NAME                                  DESIRED   CURRENT   READY   AGE    CONTAINERS   IMAGES   SELECTOR
replicaset.apps/mayank-9c4f4b955      1         1         1       105s   nginx        nginx    app=mayank,pod-template-h
ash=9c4f4b955
[root@master code]# kubectl delete  rs mayank-9c4f4b955
```

Delete replica – automatically created:

```
[root@master code]# kubectl delete  rs mayank-9c4f4b955
replicaset.apps "mayank-9c4f4b955" deleted
[root@master code]# kubectl get all  -o wide
NAME                            READY   STATUS              RESTARTS   AGE   IP       NODE    NOMINATED NODE   READINESS
 GATES
pod/mayank-9c4f4b955-d5k6x      0/1     ContainerCreating   0          2s    <none>   node5   <none>           <none>

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS   IMAGES   SELECTOR
deployment.apps/mayank     0/1     1            0           2m12s   nginx        nginx    app=mayank

NAME                                  DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES   SELECTOR
replicaset.apps/mayank-9c4f4b955      1         1         0       2s    nginx        nginx    app=mayank,pod-template-ha
sh=9c4f4b955
[root@master code]#
```

Verify:

```
[root@master code]# curl 192.168.0.32
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[root@master code]#
```

get deployment:

```
[root@master code]# kubectl get deployments.apps
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
mayank   1/1     1            1           3m36s
[root@master code]# kubectl edit
```

Edit the deployment:

```
[root@master code]# kubectl edit   deployments.apps  mayank
```

```
[root@master code]# kubectl edit   deployments.apps  mayank
Edit cancelled, no changes made.
[root@master code]#
```

If you want to create a deployment - want to get yaml automatically:

Dry run: Test

```
[root@master code]# kubectl get deployments.apps
[root@master code]# kubectl get deployments.apps
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
mayank   1/1     1            1           3m36s
[root@master code]# kubectl edit   deployments.apps  mayank
Edit cancelled, no changes made.
(reverse-i-search)`dep': kubectl edit   ^Cployments.apps  mayank
[root@master code]# kubectl create  deployment mayank --image=nginx  --dry-run
[root@master code]# kubectl create  deployment mayank --image=nginx  --dry-run
W0426 10:57:32.386642    3726 helpers.go:598] --dry-run is deprecated and can be replaced with
deployment.apps/mayank created (dry run)
[root@master code]#
```

```
[root@master code]# kubectl create  deployment mayank --image=nginx  --dry-run  -o yaml
W0426 10:57:55.199045    4094 helpers.go:598] --dry-run is deprecated and can be replaced with --dry-run=client.
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: mayank
  name: mayank
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mayank
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: mayank
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
[root@master code]#
```

Create the deployment with dry run:

```
[root@master code]# kubectl create  deployment mayank --image=nginx  --dry-run  -o yaml > deployment.yaml
W0426 10:58:13.634753    4405 helpers.go:598] --dry-run is deprecated and can be replaced with --dry-run=client.
[root@master code]# vim de
```

yaml is completely ready:

Yaml file:

How to deploy the yaml file:

create command:

Get all:

```
[root@master code]# kubectl get all
NAME                          READY   STATUS    RESTARTS   AGE
pod/mayank-9c4f4b955-jg5x4    1/1     Running   0          5s

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mayank    1/1     1            1           5s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/mayank-9c4f4b955    1         1         1       5s
[root@master code]#
```

Delete the pod:

```
[root@master code]# kubectl delete  pod mayank-9c4f4b955-jg5x4  -fo
error: the path "o" does not exist
[root@master code]# kubectl delete  pod mayank-9c4f4b955-jg5x4  --force
warning: Immediate deletion does not wait for confirmation that the running resource ha
e may continue to run on the cluster indefinitely.
pod "mayank-9c4f4b955-jg5x4" force deleted
[root@master code]# kubectl delete  pod mayank-9c4f4b955-jg5x4  -fo
error: the path "o" does not exist
[root@master code]# kubectl get all
NAME                          READY   STATUS    RESTARTS   AGE
pod/mayank-9c4f4b955-lnsxs    1/1     Running   0          3s

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mayank    1/1     1            1           37s

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/mayank-9c4f4b955    1         1         1       37s
[root@master code]#
```

Get wide:

```
[root@master code]# kubectl get pods -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP              NODE    NOMINATED NODE   READINESS GATES
mayank-9c4f4b955-lnsxs   1/1     Running   0          40s   192.168.0.33   node5   <none>           <none>
[root@master code]# cur
```

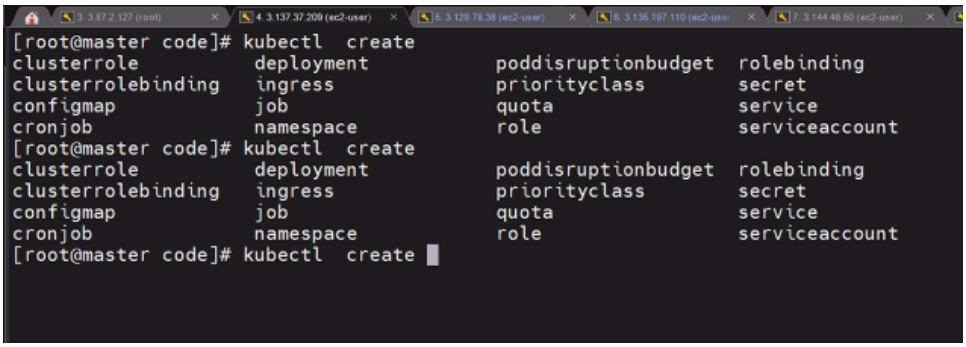
Curl – verify:

```
[root@master code]# curl 192.168.0.33
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[root@master code]#
```

These many things you can create:



Deployment Commands shared by Trainer:

47 kubectl get ns

48 kubectl get pods

49 kubectl get pods -n default

50 kubectl get all -n default

51 kubecget ns

52 kubectl get nsl

53 kubectl get ns

54 kubectl get deployments.apps

55 kubectl get deployments

56 kubectl get deployments,pods,rs

57 kubectl get all

58 kubectl create deployment mayank --image=nginx

59 kubectl get all

60 kubectl get all -o wide

61 curl 192.168.0.31

62 kubectl get all

63 kubectl delete pod mayank-9c4f4b955-ctdcl --force

64 kubectl get all

65 kubectl get all -o wide

66 kubectl delete rs mayank-9c4f4b955

67 kubectl get all -o wide

68 curl 192.168.0.32

```
   69  kubectl get deployments.apps
   70  kubectl edit   deployments.apps  mayank
   71  kubectl create  deployment mayank --image=nginx  --dry-run
   72  kubectl create  deployment mayank --image=nginx  --dry-run  -o yaml
   73  kubectl create  deployment mayank --image=nginx  --dry-run  -o yaml > deployment.yaml
   74  vim deployment.yaml
   75  kubectl create  -f deployment.yaml
   76  kubectl delete  deployments.apps  mayank
   77  kubectl create  -f deployment.yaml
   78  kubectl get all
   79  kubectl delete  pod mayank-9c4f4b955-jg5x4  -fo
   80  kubectl delete  pod mayank-9c4f4b955-jg5x4  --force
   81  kubectl delete  pod mayank-9c4f4b955-jg5x4  -fo
   82  kubectl get all
   83  kubectl get pods -o wide
   84  curl 192.168.0.33
   85  kubectl describe  deployments.apps  mayank
   86  history
```

Yaml file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: mayank
  name: mayank
spec:
  replicas: 1
  selector:
```

```
    matchLabels:
      app: mayank
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: mayank
    spec:
      containers:
      - image: nginx
        name: nginx
        resources: {}
status: {}
```

Task:

http://54.159.140.24/kube/task2.php

Q.1 Create A deployment using CLi Method using mdhack/myapache image and deployment name should be test1.

Create the deployment:

```
[root@vishali ~]# kubectl create deployment test1 --image=mdhack/myapache
deployment.apps/test1 created
```

Verify it:

```
[root@vishali ~]# kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
test1   1/1     1            1           20s
[root@vishali ~]#
```

Q.2 Create A deployment using generating yaml Method using mdhack/myserver image and deployment name should be test2

```
[root@vishali ~]# create deployment test2 --image=mdhack/myserver --dry-run -o y
aml > deployment.yaml
-bash: create: command not found
[root@vishali ~]# ^C
[root@vishali ~]# kubectl create deployment test2 --image=mdhack/myserver --dry-
run -o yaml > deployment.yaml
W0426 11:17:47.731511   10406 helpers.go:598] --dry-run is deprecated and can be
 replaced with --dry-run=client.
[root@vishali ~]# kubectl create -f deployment.yaml
deployment.apps/test2 created
[root@vishali ~]# kubectl get all
NAME                         READY   STATUS    RESTARTS   AGE
pod/pod1                     1/1     Running   0          83m
pod/pod2                     1/1     Running   0          79m
pod/test1-6db788bdd4-mccwb   1/1     Running   0          4m55s
pod/test2-744cdc6c89-dvpgb   1/1     Running   0          11s
pod/yamlpod1                 1/1     Running   0          73m
pod/yamlpod2                 1/1     Running   0          69m

NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/test1   1/1     1            1           4m55s
deployment.apps/test2   1/1     1            1           11s

NAME                               DESIRED   CURRENT   READY   AGE
replicaset.apps/test1-6db788bdd4   1         1         1       4m55s
replicaset.apps/test2-744cdc6c89   1         1         1       11s
[root@vishali ~]#
```

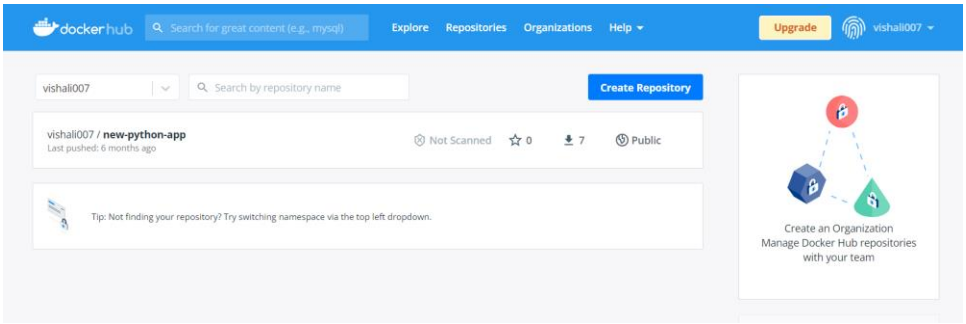
```
[root@vishali ~]# kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE     IP              NO
DE    NOMINATED NODE   READINESS GATES
pod1                     1/1     Running   0          84m     192.168.0.133   no
de1   <none>           <none>
pod2                     1/1     Running   0          80m     192.168.0.201   no
de3   <none>           <none>
test1-6db788bdd4-mccwb   1/1     Running   0          5m51s   192.168.0.150   no
de1   <none>           <none>
test2-744cdc6c89-dvpgb   1/1     Running   0          67s     192.168.0.88    no
de4   <none>           <none>
yamlpod1                 1/1     Running   0          74m     192.168.0.203   no
de3   <none>           <none>
yamlpod2                 1/1     Running   0          70m     192.168.0.140   no
de1   <none>           <none>
[root@vishali ~]#
```

Q.3 Create a dockerfile using oraclelinux:8.3 image configure Apache into it with 3 html files into it. and then upload the newly created image on yourdocker hub account. Use that same image while creating the deployment that u have uploaded.

Q.4 Create a deployment using image quay.io/mayank123modi/simple-webapp After creating this expose it on LoadBalancer by CLI method and check everything.

Post Task:

Q.1 create an account on Dockerhub



Q.2 create your own docker image with having 2 diff html files. And find a way to upload those images on your docker hub account

Edit the dockerfile:

```
[root@vishali /]# cd task3
[root@vishali task3]# ls
Dockerfile
[root@vishali task3]# cat Dockerfile
FROM oraclelinux:8.3
RUN yum install httpd -y
WORKDIR /var/www/html/
RUN echo Hello From Dockerfile > index.html
RUN echo Hello from Training file > training.html
CMD ["httpd","-DFOREGROUND"]
[root@vishali task3]#
```

Docker build:

```
[root@vishali task3]# vim Dockerfile
[root@vishali task3]# docker build -t vishaliimage .
Sending build context to Docker daemon  2.048kB
Step 1/6 : FROM oraclelinux:8.3
8.3: Pulling from library/oraclelinux
dd34f38d274c: Pull complete
Digest: sha256:af3182ee6c1e56f18fc1fecaf638da57d7c47233862e5c32
Status: Downloaded newer image for oraclelinux:8.3
 ---> 816d99f0bbe8
Step 2/6 : RUN yum install httpd -y
 ---> Running in 1eab74182ca4
Oracle Linux 8 BaseOS Latest (x86_64)            42 MB/s |  43
```

Docker images:

```
[root@vishali task3]# docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
vishaliimage   latest    60e49d69ed90   15 seconds ago   393MB
newnginx       latest    20a73b7c11ab   5 hours ago      142MB
httpdimage     latest    0bde37eebb5b   5 hours ago      144MB
nginx          latest    fa5269854a5e   6 days ago       142MB
httpd          latest    c30a46771695   6 days ago       144MB
oraclelinux    8.3       816d99f0bbe8   12 months ago    224MB
```

Docker login: using the command docker login

```
[root@vishali task3]# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you d
have a Docker ID, head over to https://hub.docker.com to create one.
Username: vishali007
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.j
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-sto

Login Succeeded
```

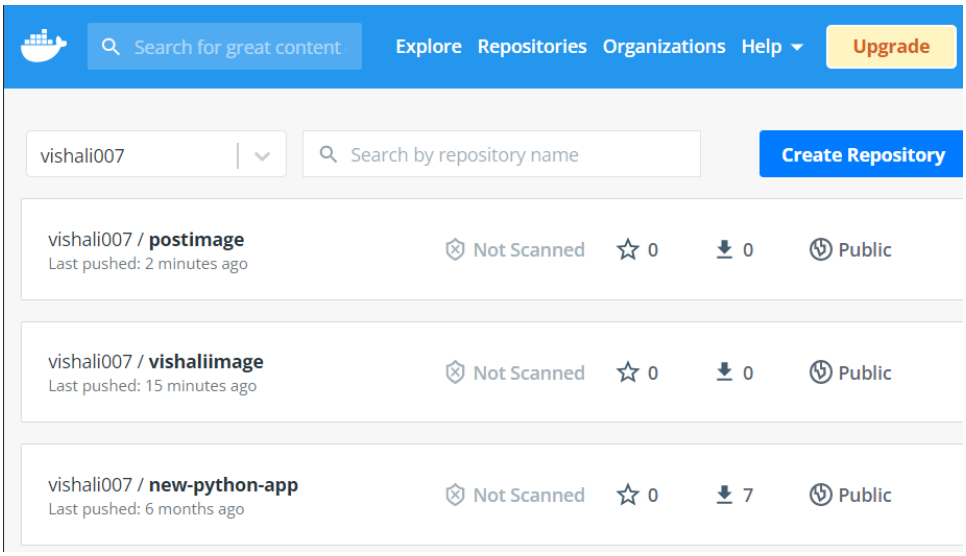
Perform tag and push the image:

docker image tag vishaliimage:latest vishali007/vishaliimage

docker image push vishali007/vishaliimage:1.0

```
[root@vishali task3]# docker image tag vishaliimage:latest vishali007/vishaliima
ge:1.0
[root@vishali task3]# docker image push vishali007/vishaliimage:1.0
The push refers to repository [docker.io/vishali007/vishaliimage]
0b5df366ee68: Pushed
daf56a403ed6: Pushed
ea6f8eb24f4b: Pushed
02a3a073ed48: Mounted from library/oraclelinux
1.0: digest: sha256:ccd6c63949f3aa343685623569aa5d50bdabeea4f836ac7a72eae3f5a8d3
02c5 size: 1156
[root@vishali task3]#
```

Docker images are uploaded in the docker hub:



Q.3 Use the same image that u has uploaded to create deployment in Kubernetes cluster.

Create the deployment:

docker run –itd –name newcont1 vishali007/vishaliimage:1.0

```
[root@vishali task3]# docker run -itd --name newcont1 vishali007/postimage:1.0
83381afa0268c57276c244e039d7fed979b439e06a02474263adab4f5bf21bd8
[root@vishali task3]# kubectl create deployment postimagetask3 --image=vishali00
7/postimage:1.0
deployment.apps/postimagetask3 created
```

Verify the status of the pod:

```
[root@vishali task3]# kubectl get pod
NAME                              READY   STATUS             RESTARTS   AGE
pod1                              1/1     Running            0          125m
pod2                              1/1     Running            0          121m
postimagetask-6f4db4bcd5-vqjd7    0/1     ImagePullBackOff   0          17m
postimagetask3-db9b9dd8d-p8kf7    1/1     Running            0          11m
posttask-59b9548d4c-qhxs7         0/1     ImagePullBackOff   0          21m
posttask3-7ff7f9b799-g22p2        0/1     ImagePullBackOff   0          28m
test1-6db788bdd4-mccwb            1/1     Running            0          47m
test2-744cdc6c89-dvpgb            1/1     Running            0          42m
yamlpod1                          1/1     Running            0          115m
yamlpod2                          1/1     Running            0          111m
[root@vishali task3]#
```

Get the IP details using the –o wide command:

```
[root@vishali task3]# kubectl get pods -o wide
NAME                              READY   STATUS             RESTARTS   AGE     I
P               NODE    NOMINATED NODE   READINESS GATES
pod1                              1/1     Running            0          114m    1
92.168.0.133    node1   <none>           <none>
pod2                              1/1     Running            0          109m    1
92.168.0.201    node3   <none>           <none>
postimagetask-6f4db4bcd5-vqjd7    0/1     ImagePullBackOff   0          6m13s   1
92.168.0.218    node3   <none>           <none>
postimagetask3-db9b9dd8d-p8kf7    1/1     Running            0          21s     1
92.168.0.52     node2   <none>           <none>
posttask-59b9548d4c-qhxs7         0/1     ImagePullBackOff   0          10m     1
92.168.0.50     node2   <none>           <none>
posttask3-7ff7f9b799-g22p2        0/1     ImagePullBackOff   0          16m     1
92.168.0.47     node2   <none>           <none>
test1-6db788bdd4-mccwb            1/1     Running            0          35m     1
92.168.0.150    node1   <none>           <none>
test2-744cdc6c89-dvpgb            1/1     Running            0          30m     1
92.168.0.88     node4   <none>           <none>
yamlpod1                          1/1     Running            0          104m    1
92.168.0.203    node3   <none>           <none>
yamlpod2                          1/1     Running            0          100m    1
92.168.0.140    node1   <none>           <none>
[root@vishali task3]# ^C
```

Pod IP Address: 192.168.0.52

Alternative of how to try these above commands: