13 docker -v

14 docker version

15 ps -ef| grep docker

16 docker version

17 service docker status

18 docker ps

19 service docker stop

20 docker ps

21 service docker status

22 service docker start

23 docker ps

24 cd /var/lib/docker/

31 docker image ls

32 docker pull tomcat

33 docker pull tomcat:8.5.55

34 docker pull tomcat:8.5.55777

35 docker image ls

36 docker pull nginx

37 docker pull centos

38 docker pull alpine

39 clear

40 docker image ls

41 docker image inspect nginx:latest

42 clear

43 docker image ls

44 docker image rm tomcat

45 docker image ls

48 docker image ls

49 docker run tomcat:8.5.55

50 clear

51 docker ps ## this command to see running cont on your systems

52 docker ps

53 docker ps -a ## to see running & exited cont on your system

54 docker run -d tomcat:8.5.55

55 docker ps

56 docker run -d tomcat:8.5.55

57 docker ps

58 docker run nginx

59 docker run -d nginx

60 docker ps

61 docker contianer inspect 1128bfc5c346

62 docker container inspect 1128bfc5c346

63 docker ps

64 docker ps -a

65 docker inspect 1128bfc5c346

66 clear

67 docker ps

68 docker stop 1128bfc5c346 ca3417e634b4

69 docker ps -a

70 docker start 1128bfc5c346 ca3417e634b4

71 docker ps -a

72 docker restart 1128bfc5c346 ca3417e634b4

73 docker kill 1128bfc5c346 ca3417e634b4

74 docker ps -a

75 docker start 1128bfc5c346 ca3417e634b4

76 docker ps -a

77 docker rm 1128bfc5c346

78 docker rm --force 1128bfc5c346 f97df6f2bf9b 5512eeb80284

79 docker ps -a

83 docker exec -it 16759f4829a1 bash

84 docker ps

85 docker exec -it 112189ff154a bash

86 docker ps

87 docker exec 112189ff154a ls -l

88 docker exec 112189ff154a cat docker-entrypoint.sh

89 docker exec 112189ff154a touch /tmp/naresh.txt

90 ll /tmp/naresh.txt

91 docker exec 112189ff154a ls -l /tmp

92 clear

93 history

94 docker ps

95 docker logs 112189ff154a

96 docker logs --follow 112189ff154a

97 clear

98 docker ps

99 docker stats 112189ff154a

100 free -m

101 docker stats

102 docker ps

103 docker run -d --name naresh nginx

104 docker ps

105 docker rename confident\_jemison harini

106 docker ps

107 clear

108 docker ps

109 docker top e83b62976ede

110 docker top 5b12c024a657

111 top

112 docker top e83b62976ede

115 docker ps

116 docker stats

117 docker top 59b1142ef73e

118 docker top ca3417e634b4

119 clear

120 docker ps

121 docker image ls

122 docker run -d centos

123 docker ps

124 docker ps -a

125 docker top ca3417e634b4

126 docker top centos

127 docker top 48d429e691e3

128 docker image inspect ca3417e634b4

129 docker image inspect tomcat:8.5.55

130 clear

131 docker image inspect centos

132 clear

133 docker run -d centos sleep 30

134 docker ps

135 docker ps -a

136 docker start d759ad762d92

137 docker ps -a

138 docker inspect d759ad762d92

139 clear

140 docker image ls

141 docker ps

142 docker run -d tomcat:8.5.55 sleep 10

143 docker ps

144 docker image ls

145 docker image inspect tomcat:8.5.55

146 clear

147 docker image inspect centos | grep -i exposed

148 docker image inspect tomcat:8.5.55 | grep -i exposed

149 docker run -d tomcat:8.5.55 "sleep 10 ; ls -l"

150 docker run -d tomcat:8.5.55 "sleep 10 ; ls"

151 clear

152 docker ps

153 docker inspect e83b62976ede

154 docker ps

155 docker inspect 59b1142ef73e

156 curl 172.17.0.4:80

157 clear

158 docker ps

159 docker run -d -P nginx

160 docker ps

161 netstat -nap | grep 49153

162 netstat -nap | grep 80

163 clear

164 docker ps

165 docker run -d -p 1234:80 nginx

166 docker ps

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

**how to build docker images ?**

**there are two methods**

**manual process — creating an image from container changes**

**take base image ( can be any image from docker hub ) — tomcat:8.5.40**

**run it as cont — docker run -d tomcat:8.5.40**

**get inside the cont — docker exec -it contid bash**

**## inside cont do whatever changes we want**

**deploy an app to tomcat**

**install something like vi editor**

**come out of cont — always use “ctrl pq”**

**ensure the cont is in running state — docker ps**

**commit the cont changes to prepare a new image**

**— docker commit -m “any msg” contid newimagename:tag**

**Within the cont**

1 ls -l

2 cd webapps/

3 ls -l

4 mkdir myapp

5 cd myapp/

6 ls -l

7 touch index.html

8 ls -l

9 vi index.html

10 apt-get update

11 apt-get install -y vim

12 vi index.html

170 docker pull tomcat:8.5.40

171 docker image ls

173 docker ps

174 docker rm -f `docker ps -qa`

175 clear

176 docker ps

177 docker run -d tomcat:8.5.40

178 docker ps

179 docker exec -it e0666930eb90 bash

180 docker ps

181 docker image ls

182 docker commit -m "deployed myapp" e0666930eb90 myapp:v1

183 docker image ls

184 docker run -d -P myapp:v1

187 docker ps

**automated process — we use a script called Dockerfile**

**— we use pre defined dockerfile insturctions given by docker**

**ex: FROM / RUN / CMD etc..**

**— everyline in dockerfile must start with dockerfile instruction**

**— Every Dockerfile instruction is a single word & must be written in capital letters**

**— dockerfile is plain text file also without any ext**

**vi Dockerfile**

**FROM tomcat:8.5.40**

**RUN mkdir /usr/local/tomcat/webapps/thapp**

**RUN touch /usr/local/tomcat/webapps/thapp/index.html**

**RUN apt-get update**

**RUN apt-get install -y vim**

**RUN echo “hi there, Welcome to th app” > /usr/local/tomcat/webapps/thapp/index.html**

**how to execute the dockerfile script**

**docker build —file Dockerfile —tag newimage:tag . (context (a path on your vm where file exist )**

193 vi Dockerfile

194 clear

195 ls -l

196 pwd

197 docker build --file Dockerfile --tag thapp:v1

198 docker build --file Dockerfile --tag thapp:v1 /root

199 clear

200 docker image ls

201 docker run -d -P thapp:v1

202 docker ps

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

root@thmaster:~# cat myapdfile

FROM ubuntu:18.04

RUN apt-get -y update

RUN apt-get -y install openjdk-8-jdk

RUN mkdir /opt/tomcat

COPY tomcat-8.5.37.tar.gz /tmp

RUN cd /tmp && tar xvfz tomcat-8.5.37.tar.gz

RUN cp -Rv /tmp/apache-tomcat-8.5.37/\* /opt/tomcat/

RUN rm -r /tmp/tomcat-8.5.37.tar.gz

COPY myapp.war /opt/tomcat/webapps

EXPOSE 8080

CMD /opt/tomcat/bin/catalina.sh run

205 vi myapdfile

206 clear

207 ls -l

208 cat myapdfile

209 wget https://github.com/lerndevops/educka/raw/master/docker/code/tomcat-8.5.37.tar.gz

210 wget https://github.com/lerndevops/educka/raw/master/docker/code/myapp.war

211 clear

212 ls -l

213 cat myapdfile

214 docker image ls

215 docker run -d -P tomcat:8.5.55

216 docker ps

217 docker exec -it 92a372b16c04 bash

218 cat myapdfile

219 cat Dockerfile

220 docker build --file myapdfile --tag thapp:v2 /root

221 docker image ls

222 docker run -d -P thapp:v2

223 docker ps

224 cat myapdfile

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

**DAY3**

227 docker image ls

228 docker ps

229 docker login

230 docker push thapp:v2

231 history

232 docker build --file myapdfile --tag lerndevops/thjuly19:v1 /root

233 docker image ls

234 docker tag myapp:v1 lerndevops/thjuly19:v2

235 docker image ls

236 docker push lerndevops/thjuly19:v1

237 docker push lerndevops/thjuly19:v2

238 clear

239 docker pull lerndevops/thjuly19:v1

Volumes:

267 docker run -d -P nginx

268 docker ps

269 docker exec -it f1c4d2bc95ea bash

270 docker ps

271 docker rm -f f1c4d2bc95ea

272 docker ps

273 clear

274 df -h /

275 docker volume ls

276 docker volume create myvol

277 docker volume ls

278 docker volume inspect myvol

279 cd /var/lib/docker/volumes/myvol/\_data

280 ls -l

281 cd

282 docker run -d -P --volume myvol:/usr/share/nginx/html nginx

283 docker ps

284 cd -

285 ls -l

286 echo "hi there, from host vm" > index.html

287 ls -l

288 cat index.html

289 cd

290 docker ps

291 docker rm -f 39277b93e2ab

292 docker ps

293 cd -

294 ls -l

295 cat index.html

296 docker run -d -P --volume myvol:/usr/share/nginx/html nginx

297 docker ps

298 cd

299 mkdir /nginxconf

300 mkdir /nginxlogs

301 docker run -d -P --volume /nginxconf:/usr/share/ngnix/html /nginxlogs:/var/log/nginx nginx

302 docker run -d -P --volume /nginxconf:/usr/share/ngnix/html --volume /nginxlogs:/var/log/nginx nginx

303 docker ps

304 docker inspect 3740101a8ec9

305 clear

306 cd /nginxlogs/

307 ls -l

308 cd /nginxconf/

309 ls -l

310 touch index.html

311 echo "hello from thmaster" > index.html

312 ls -l

313 docker ps

314 docker rm -f 3740101a8ec9

315 docker run -d -P --volume /nginxconf:/usr/share/nginx/html --volume /nginxlogs:/var/log/nginx nginx

316 docker ps

317 cd /nginxlogs/

318 ls -l

319 tail -f access.log

320 ls -l

321 docker ps

322 docker rm -f 86b340a54f4a

323 ls -l

324 cat access.log

Networking in Docker

331 docker network ls

332 docker network inspect bridge

333 ifconfig

334 hostname -i

335 docker network ls

336 clear

337 docker run -d --name cont1 lerndevops/alpine:sleep

338 docker run -d --name cont2 lerndevops/alpine:sleep

339 docker ps

340 docker inspect cont1

341 docker inspect cont2

342 clear

343 docker ps

344 docker exec cont1 ifconfig

345 docker exec cont1 hostname -i

346 docker exec cont2 hostname -i

347 docker exec cont2 ping 172.17.0.2

348 docker exec cont1 ping 172.17.0.3

349 docker network ls

350 clear

351 docker network ls

352 docker network create --help

353 docker network create --driver bridge test

354 docker network ls

355 docker network inspect test

356 docker network create --driver bridge mynet --subnet 192.168.0.0/16 --gateway 192.168.0.1

357 docker network ls

358 docker network inspect mynet

359 clear

360 docker network ls

361 docker run -d --name cont3 --network mynet lerndevops/alpine:sleep

362 docker run -d --name cont4 --network mynet lerndevops/alpine:sleep

363 docker ps

364 docker inspect cont3

365 docker inspect cont4

366 docker exec cont3 ifconfig

367 docker exec cont3 hostname -i

368 docker exec cont4 hostname -i

369 docker exec cont1 hostname -i

370 docker exec cont2 hostname -i

371 docker exec cont4 ping 192.168.0.2

372 docker exec cont3 ping 192.168.0.3

373 docker exec cont3 ping cont4

374 docker exec cont4 ping cont3

375 docker exec cont1 ping cont2

376 docker exec cont1 ping cont4

377 docker exec cont1 ping 192.168.0.3

382 docker network connect mynet cont1

383 docker exec cont1 ifconfig

384 docker exec cont1 ping cont3

385 docker exec cont1 ping cont4

393 docker network ls

394 docker run -d --name cont5 --network none lerndevops/alpine:sleep

395 docker ps

396 docker exec cont5 ifconfig

397 ifconfig

398 clear

399 docker run -d --name cont6 --network host lerndevops/alpine:sleep

400 hostname -i

401 docker exec cont6 hostname -i

402 docker exec cont6 ifconfig

403 clear

404 docker run -d --name cont7 --network host nginx

405 docker ps

406 docker run -d --name cont8 --network host nginx

407 docker ps -a

408 docker logs edd57aaa37ee

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

yaml / yml -- yaml ain't a markup lang -- data serialization lang

yaml is a very human frendly when compared to JSON

in a yaml file we write all the data as key value pairs ( key: value )

there is on thumb rule you need to follow when you write any yaml -- indentation -- spaces

YOU SHOULD NOT USE TAB KEY ON YOUR KEYBOARD WHILE WRITING ANY YAML -- YOU SHOULD USE ONLY SPACE BAR ON YOUR KEYBOARD

yaml is also case sensitive

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

3 data structures are followed in any yaml

scalars

name: naresh

a: 5

b: 56.6

c: true

dict / map

car:

name: bmw

color: red

model: xuv

engine: 2L

list / arry

- bmw

- audi

- skoda

- kia

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

cars:

- bmw:

color: white

model: sadan

engine: 2L

- audi:

color:

- black

- white

- blue

model:

- sadan

- xuv

- hatchback

engine:

- "2L"

- 3L

- 5L

- skoda

- kia

-------------------------------------------------------

411 sudo curl -L https://github.com/docker/compose/releases/download/1.23.2/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

412 sudo chmod +x /usr/local/bin/docker-compose

413 docker-compose -v

414 vi docker-compose.yml

415 clear

416 docker-compose

417 clear

418 ls -l

419 pwd

420 docker-compose up -d

421 vi docker-compose.yml

422 docker-compose up -d

423 vi docker-compose.yml

424 docker-compose up -d

425 docker network ls

426 docker volume ls

427 docker ps

428 docker rm -f `docker ps -qa`

429 docker ps

430 docker-compose up -d

431 docker ps

432 cat docker-compose.yml

433 docker-compose --scale svc2=4 --scale myappsvc=6 up -d

434 docker-compose up --scale svc2=4 --scale myappsvc=6 -d

435 docker ps

436 docker-compose up --scale svc2=2 --scale myappsvc=3 -d

437 docker ps

438 docker-compose down

439 ls -l

440 mv docker-compose.yml mycompose.yml

441 docker-compose up --scale svc2=2 --scale myappsvc=3 -d

442 docker-compose -f mycompose.yml up --scale svc2=2 --scale myappsvc=3 -d

443 docker-compose -f mycompose.yml down

444 docker ps

version: "3.7"

volumes:

myvol:

myvol2:

logs:

networks:

mynet:

test2:

services:

myappsvc: ## this is the name of the service which can be any value of your choice

image: nginx

ports:

- 80

- 443

volumes:

- myvol:/usr/share/nginx/html

- logs:/var/lib/nginx

networks:

- mynet

- test2

svc2:

image: tomcat

ports:

- 8080

volumes:

- myvol2:/usr/share/tomcat/logs

networks:

- mynet

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

***DOCKER SWARM***

*On master node*

546 docker info

547 clear

548 docker swarm init

549 docker node ls

550 hostname -i

551 docker node ls

552 docker node inspect thnode01

*On worker nodes*

Take the join command from your master node and run it on worker nodes : ex like below

docker swarm join --token SWMTKN-1-2lphoskk9t4pkikbsbear69wtv7nww62w60hpohwap6dtiahgf-eowu5moui5ivma3ns4xa4q7x7 10.154.0.5:2377

558 docker node ls

559 docker service create --name myapp --replicas 7 -p 9080:3000 lerndevops/samplepyapp:v1

560 docker service create --name helloapp --replicas 5 -p 9081:80 nginx

561 docker service ls

562 docker service ps myapp

563 docker ps

564 docker node ps thnode01

565 docker node ps thnode02

566 docker service ls

567 docker service scale myapp=15

568 docker service ps myapp

569 docker service scale myapp=3

570 docker service ps myapp

571 docker service ls

572 docker service rm helloapp

573 docker service ls

**577 docker service ls**

**578 docker service ps myapp**

**579 docker ps**

**580 docker rm -f 81dacfe36bcb**

**581 docker service ls**

**582 docker service ps myapp**

**583 docker node ls**

**584 docker service ps myapp**

**585 docker service ls**

**586 clear**

**587 docker service ls**

**588 docker service scale myapp=6**

**589 docker service ps myapp**

**590 docker service ps myapp | grep -i Running**

**591 docker ps**

**592 docker service ls**

629 docker service l

630 docker service ls

631 docker service rm hellosvc testsvc

632 clear

633 ls -l

634 docker stack deploy -c myapp.yml myapp

635 docker stack ls

636 docker stack ps myapp

637 docker stack services myapp

638 docker service ls

639 docker service ps myapp\_mongo

640 docker inspect s5cbk27vsojg

641 cat myapp.yml

642 clear

643 docker node ls

644 docker node inspect thnode01

645 docker node ls

646 docker node update thnode01 --label-add role=app

647 docker node update thnode02 --label-add role=db

648 docker service ls

649 docker service ps myapp\_springbootapp

650 docker service ls

651 docker service ps myapp\_mongo

652 docker service logs --follow myapp\_springbootapp

653 clear

654 docker service ls

655 docker service scale myapp\_mongo=0

656 docker service ls

657 docker service logs --follow myapp\_springbootapp

658 docker service scale myapp\_mongo=1

root@thmaster:~# cat myapp.yml

version: '3.7'

volumes:

data:

data-bkp:

networks:

springappnet:

services:

springbootapp:

image: lerndevops/springboot-mongo-app:latest

deploy:

replicas: 4

placement:

constraints:

- node.labels.role==app

restart\_policy:

condition: on-failure

resources:

limits:

cpus: "0.2" ## 20% cpu of 1 core cpu

memory: 300M

ports:

- "9090:8080"

networks:

- springappnet

depends\_on:

- mongo

mongo:

image: lerndevops/mongo

deploy:

replicas: 1

placement:

constraints:

- node.labels.role==db

restart\_policy:

condition: on-failure

ports:

- "27017:27017"

networks:

- springappnet

volumes:

- data:/data/db

- data-bkp:/data/bkp

***de init the docker swarm cluster***

*On all worker nodes first run below command*

docker swarm leave --force

*On master node do below*

664 docker node rm thnode02

665 docker node rm thnode01

666 docker node ls

667 docker swarm leave --force

668 docker node ls

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

***KUBERNETES***

673 echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

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

675 sudo apt-get update ; clear

676 sudo apt-get install -y kubelet kubeadm kubectl

677 clear

678 docker -v

679 kubeadm version -o short

680 kubelet --version

681 kubectl version --client --short

701 kubectl get nodes -o wide

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

703 clear

704 kubectl run mypod --image=nginx

705 kubectl get pods

706 kubectl get pods -o wide

707 curl 10.44.0.1:80

708 kubectl describe pod mypod

709 clear

710 kubectl logs mypod

711 kubectl logs --follow mypod

712 clear

713 kubectl get pods

714 kubectl exec -it mypod -- bash

715 kubectl exec -it mypod -- ls -l

716 kubectl exec mypod -- touch /tmp/naresh.txt

717 kubectl exec mypod -- ls -l /tmp

718 kubectl get pods

719 kubectl delete pod mypod

720 kubectl run mypod --image=nginx

721 kubectl run mypod1 --image=nginx

722 kubectl get pods

723 kubectl delete pod mypod --force

743 kubectl get pods

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

745 service kubelet status

746 service docker status

747 history

748 clear

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

750 clear

751 kubectl run nginxpod --image=nginx

752 kubectl get pods -o wide

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

754 docker ps

755 clear

756 kubectl get pods -o wide

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

758 kubectl get namespaces

759 kubectl get pods -o wide --namespace default

760 kubectl get pods -o wide --namespace kube-system

761 kubectl get pods -o wide --all-namespaces ## by default you get cluster-admin access when you create a cluster

762 clear

763 kubectl create namespace teama

764 kubectl create namespace teamb

765 kubectl get ns

766 kubectl run mypod --image=nginx

767 kubectl get pods -n default

768 kubectl get pods -n teama

769 kubectl get pods -n teamb

770 kubectl run mypod --image=nginx -n teama

771 kubectl run mypod --image=nginx -n teamb

772 kubectl get pods -n teama

773 kubectl get pods -n teamb

774 kubectl get pods --all-namesapces

775 kubectl get pods --all-namespaces

787 vi pod.yml

788 clear

789 ls -l

790 kubectl create -f pod.yml

791 kubectl get pods -o wide

792 kubectl get pods --show-labels

793 kubectl get pods -l run

794 kubectl get pods -l team

795 kubectl get pods -l team=thoshiba

796 kubectl get pods -l run

797 kubectl get pods -l run=test

root@thmaster:~# cat pod.yml

kind: Pod ## kubectl api-resources

apiVersion: v1

metadata:

name: testpod

namespace: default

labels: ## defining a label for a is mandatory, with labels we can identify a group of similar pods

team: thoshiba ## here both key & value are your choice

spec: ## is about what is inside the pod

containers:

- name: cont1

image: lerndevops/samplepyapp:v1

ports:

- name: http

containerPort: 3000

- name: https

containerPort: 3443

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

kind: Pod ## kubectl api-resources

apiVersion: v1

metadata:

name: multi-cont-pod2

namespace: default

labels: ## defining a label for a is mandatory, with labels we can identify a group of similar pods

team: thoshiba ## here both key & value are your choice

spec: ## is about what is inside the pod

restartPolicy: Always # Never

containers:

- name: naresh

image: lerndevops/samplepyapp:v1

ports:

- name: http

containerPort: 3000

- name: https

containerPort: 3443

- name: cont2

image: httpd

ports:

- name: http

containerPort: 80

- name: https

containerPort: 443

- name: cont3

image: centos

812 vi multipod.yml

813 clear

814 kubectl create -f multipod.yml

815 kubectl get pods -o wide

816 kubectl describe pod mulit-cont-pod

817 kubectl describe pod multi-cont-pod

818 clear

819 kubectl get pods

820 kubectl logs test

821 kubectl logs multi-cont-pod

822 kubectl logs multi-cont-pod -c cont1

823 kubectl logs multi-cont-pod -c cont2

824 kubectl logs multi-cont-pod -c cont3

825 kubectl exec -it multi-cont-pod -c cont1 -- bash

826 kubectl exec -it multi-cont-pod -c cont2 -- bash

827 kubectl exec -it multi-cont-pod -c cont3 -- bash

828 vi multipod2.yml

829 kubectl create -f multipod2.yml

830 kubectl get pods -o wide

kind: Pod ## kubectl api-resources

apiVersion: v1

metadata:

name: multi-cont-pod2

namespace: default

labels: ## defining a label for a is mandatory, with labels we can identify a group of similar pods

team: thoshiba ## here both key & value are your choice

spec: ## is about what is inside the pod

restartPolicy: Always # Never

containers:

- name: naresh

image: lerndevops/samplepyapp:v1

ports:

- name: http

containerPort: 3000

- name: https

containerPort: 3443

- name: cont2

image: httpd

ports:

- name: http

containerPort: 80

- name: https

containerPort: 443

- name: cont3

image: centos

command: ["bash", "-c", "sleep 3600"]

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

kind: Pod ## kubectl api-resources

apiVersion: v1

metadata:

name: init-cont-pod

namespace: default

labels: ## defining a label for a is mandatory, with labels we can identify a group of similar pods

team: thoshiba ## here both key & value are your choice

spec: ## is about what is inside the pod

restartPolicy: Always # Never

containers:

- name: naresh

image: lerndevops/samplepyapp:v1

ports:

- name: http

containerPort: 3000

- name: https

containerPort: 3443

- name: cont2

image: httpd

ports:

- name: http

containerPort: 80

- name: https

containerPort: 443

- name: cont3

image: centos

command: ["bash", "-c", "sleep 3600"]

initContainers:

- name: initcont1

image: alpine

command: ["sh", "-c", "sleep 30"]

- name: inticont2

image: centos

command: ["bash", "-c", "lsss"]

833 vi initpod.yml

834 clear

835 kubectl create -f initpod.yml

836 kubectl get pods -o wide

837 watch kubectl get pods

838 kubectl describe pod init-cont-pod

839 clear

840 cp -p initpod.yml initpod2.yml

841 vi initpod2.yml

842 kubectl create -f initpod2.yml

843 watch kubectl get pods

844 clear

845 kubectl describe pod init-cont-pod2

846 kubectl get pods

847 kubectl edit pod init-cont-pod

kind: Pod ## kubectl api-resources

apiVersion: v1

metadata:

name: init-cont-pod2

namespace: default

labels: ## defining a label for a is mandatory, with labels we can identify a group of similar pods

team: thoshiba ## here both key & value are your choice

spec: ## is about what is inside the pod

restartPolicy: Always # Never

containers:

- name: naresh

image: lerndevops/samplepyapp:v1

ports:

- name: http

containerPort: 3000

- name: https

containerPort: 3443

- name: cont2

image: httpd

ports:

- name: http

containerPort: 80

- name: https

containerPort: 443

- name: cont3

image: centos

command: ["bash", "-c", "sleep 3600"]

initContainers:

- name: initcont1

image: alpine

command: ["sh", "-c", "sleep 30"]

- name: inticont2

image: centos

command: ["bash", "-c", "ls"]

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

861 vi rs1.yml

862 clear

863 kubectl create -f rs1.yml

864 kubectl get replicaset -o wide

865 kubectl get rs -o wide

866 kubectl get pod --show-labels

867 kubectl scale rs rs1 --replicas 10

868 kubectl get pod --show-labels

869 kubectl scale rs rs1 --replicas 5

870 kubectl get pod --show-labels

871 kubectl delete pod test --force

872 kubectl get pods

873 kubectl delete pod rs1-22gmb --force

874 kubectl get pods

875 kubectl delete rs rs1

876 kubectl get pods

root@thmaster:~# cat rs1.yml

kind: ReplicaSet

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: rs1

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 4 # if you dont mention the replicas by default it will create 1 replica

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: app

spec:

restartPolicy: Always

#intiContainers:

containers:

- name: cont1

image: lerndevops/samplepyapp:v2

ports:

- name: http

containerPort: 3000

selector:

matchLabels:

type: app

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

880 kubectl explain pod

881 kubectl explain pod.metadata

882 clear

883 kubectl explain pod.spec

884 clear

885 kubectl explain replicaset.spec

886 kubectl explain replicaset.spec.selector

887 kubectl explain replicaset.spec.template

888 kubectl explain replicaset.spec.template.spec

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

kind: DaemonSet

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: ds1

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

selector:

matchLabels:

type: myapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: myapp

spec:

restartPolicy: Always

#intiContainers:

containers:

- name: cont1

image: lerndevops/samplepyapp:v2

ports:

- name: http

containerPort: 3000

894 kubectl create -f ds1.yml

895 kubectl get ds

896 kubectl get pods -o wide

897 ## daemonset create one replicas per node exactly in cluster

898 kubectl scale ds ds1 --replicas 5

899 kubectl get ds -n kube-system

900 kubectl get ds

901 kubectl get get pods -o wide

902 kubectl get pods -o wide

903 kubectl create -f rs1.yml

904 kubectl get pods -o wide

905 kubectl scale rs rs1 --replicas 30

906 kubectl get pods -o wide

907 clear

908 kubectl get nodes

909 kubectl describe node thmaster

910 clear

911 kubectl describe node thmaster | grep -i taints

912 kubectl describe node thnode01 | grep -i taints

913 kubectl describe node thnode02 | grep -i taints

914 kubectl get ds -o wide

915 kubectl taint node thmaster node-role.kubernetes.io/master:NoSchedule-

916 kubectl get ds -o wide

DAY 6

kind: ReplicaSet

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: rs1

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 4 # if you dont mention the replicas by default it will create 1 replica

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: app

spec:

restartPolicy: Always

#intiContainers:

containers:

- name: cont1

image: lerndevops/samplepyapp:v2

ports:

- name: http

containerPort: 3000

selector:

matchLabels:

type: app

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

kind: Service

apiVersion: v1 ## kubectl api-resources

metadata:

name: svc1

namespace: default

spec:

type: ClusterIP # it acts a internal virtual load balancer & forwards the requests into multiple backend pods

selector:

type: app # this is the labels of the pods which will be the backends,

ports:

- port: 80 ## this is your Virtual LB port

targetPort: 3000 ## this is the port inside cont

939 kubectl get pods -o wide --show-labels

940 kubectl describe pod rs1-4hjqw

941 clear

942 kubectl get pods -o wide --show-labels

943 curl 10.32.0.4:3000 ; echo

944 curl 10.32.0.5:3000 ; echo

945 vi svc1.yml

946 clear

947 kubectl create -f svc1.yml

948 kubectl get services

949 kubectl describe service svc1

950 kubectl get pods -o wide --show-labels

951 curl 10.32.0.4:3000 ; echo

952 curl 10.105.88.126:80 ; echo

953 cat svc1.yml 1

kind: Service

apiVersion: v1 ## kubectl api-resources

metadata:

name: np-svc

namespace: default

spec:

type: NodePort # it publishes a nodeport /hostport on every node in the cluster and also create an

# internal virtual load balancer & forwards the requests into multiple backend pods

selector:

type: app # this is the labels of the pods which will be the backends,

ports:

- port: 80 ## this is your Virtual LB port

targetPort: 3000

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

apiVersion: apps/v1

kind: Deployment

metadata:

name: kubeserve

spec:

replicas: 10

minReadySeconds: 45 # wait for 10 sec before pod is ready going to next

strategy:

type: RollingUpdate

rollingUpdate:

maxUnavailable: 1 # take down 1 pod at a time

maxSurge: 2 # bring one at a time

selector:

matchLabels:

app: kubeserve

template:

metadata:

name: kubeserve

labels:

app: kubeserve

spec:

containers:

- image: leaddevops/kubeserve:v2

name: app

---

kind: Service

apiVersion: v1

metadata:

name: kubeserve-svc

spec:

type: NodePort

ports:

- port: 80

protocol: TCP

targetPort: 80

selector:

app: kubeserve

1008 kubectl create -f dep2.yml

1009 kubectl get deployment

1010 kubectl delete deploy dep1

1011 kubectl delete rs rs1

1012 clear

1013 kubectl get deployment -o wide

1014 kubectl get pods -o wide

1015 kubectl get services

1016 clear

1017 vi dep2.yml

1018 clear

1019 kubectl create -f dep2.yml

1020 kubectl apply -f dep2.yml

1021 cat dep2.yml

1022 clear

1023 kubectl rollout pause deployment kubeserve

1024 kubectl rollout resume deployment kubeserve

1025 kubectl rollout status deployment kubeserve

1026 10 10 10

1027 cat dep2.yml

1028 clear

1029 cat dep2.yml

1030 kubectl set image deployment kubeserve app=leaddevops/kubeserve:v3

1031 clear

1032 kubectl rollout undo deployment kubeserve

1033 kubectl rollout history deployment kubeserve

1034 kubectl rollout undo deployment kubeserver --to-revision 1

1035 kubectl rollout undo deployment kubeserve --to-revision 1

root@thmaster:~# cat dep3.yml

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: dep1

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 4 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: helloapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: helloapp

spec:

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

resources:

requests: ## the min resources that can alloted to the containers

cpu: 100m ## 10% of 1 cpu core on the vm/host ( 1core cpu == 1000 milli cpus )

memory: 128Mi

limits: ## the max resources that a cont can use from the vm/host where it runs

cpu: 200m

memory: 512Mi

env:

- name: JAVA\_HOME

value: /usr/bin/java

- name: username

value: admin

- name: password

value: admin

1062 kubectl create -f dep3.yml

1063 kubectl get pods

1064 kubectl describe pod dep1-74dd48758-bzxqw

1065 clear

1066 kubectl get pod -o wide

1067 kubectl exec dep1-74dd48758-bzxqw -- print env

1068 kubectl exec dep1-74dd48758-bzxqw -- printenv

1069 kubectl exec dep1-74dd48758-bzxqw -- env

1070 env

1071 kubectl exec dep1-74dd48758-bzxqw -- env

1072 cat dep3.yml

***DAY 7***

root@thmaster:~# cat edir-vol-dep.yml

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: edir-vol-dep

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: edirapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: edirapp

spec:

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

volumes:

- name: edirvol # can be any name of your choice

emptyDir: {}

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

resources:

requests: ## the min resources that can alloted to the containers

cpu: 100m ## 10% of 1 cpu core on the vm/host ( 1core cpu == 1000 milli cpus )

memory: 64Mi

limits: ## the max resources that a cont can use from the vm/host where it runs

cpu: 200m

memory: 128Mi

env:

- name: JAVA\_HOME

value: /usr/bin/java

- name: username

value: admin

- name: password

value: admin

volumeMounts:

- name: edirvol

mountPath: /usr/share/nginx/html

- name: cont2

image: centos

command: ["bash", "-c", "echo hi there from cont2 >> /cont2data/index.html ; sleep 3600"]

volumeMounts:

- name: edirvol

mountPath: /cont2data

1100 vi edir-vol-dep.yml

1101 clear

1102 kubectl create -f edir-vol-dep.yml

1103 kubectl get pods -o wide

1104 curl 10.36.0.1:80

1105 kubectl exec -it edir-vol-dep-7c4fcf8f56-ns975 -c cont1 -- bash

1106 kubectl exec -it edir-vol-dep-7c4fcf8f56-ns975 -c cont2 -- bash

1107 clear

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

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: hp-vol-dep

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: hpvolapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: hpvolapp

spec:

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

volumes:

- name: edirvol # can be any name of your choice

emptyDir: {}

- name: hpvol

hostPath:

path: /appdata

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

#resources:

#env:

volumeMounts:

- name: edirvol

mountPath: /usr/share/nginx/html ## this is the path inside cont always

- name: hpvol

mountPath: /var/log/nginx ## this is the path inside cont always

- name: cont2

image: centos

command: ["bash", "-c", "echo hi there from cont2 >> /cont2data/index.html ; sleep 3600"]

volumeMounts:

- name: edirvol

mountPath: /cont2data

1111 vi hp-vol-dep.yml

1112 kubectl create -f hp-vol-dep.yml

1113 clear

1114 kubectl get pods -o wide

1115 curl 10.44.0.2:80

1116 kubectl delete deployment hp-vol-dep

1117 kubectl get pods

-------------------------------------------------------------------------------------------

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: pv-hp-vol-dep

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: pvhpvolapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: pvhpvolapp

spec:

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

volumes:

- name: pvhpvol

persistentVolumeClaim:

claimName: pv1-pvc

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

#resources:

#env:

volumeMounts:

- name: pvhpvol

mountPath: /usr/share/nginx/html ## this is the path inside cont always

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

kind: PersistentVolume

apiVersion: v1

metadata:

name: pv1

spec:

hostPath:

path: /mnt/appdata ## this is the path on vm where the pod is created / running

capacity:

storage: 3Gi

persistentVolumeReclaimPolicy: Retain # Delete

accessModes:

- ReadWriteOnce ## only one node can write / read the data from the mentioned volume ReadWriteMany / ReadOnlyMany

kind: PersistentVolumeClaim

apiVersion: v1

metadata:

name: pv1-pvc

namespace: default

spec:

volumeName: pv1

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 3Gi

1136 vi pv-hp-vol-dep.yml

1137 clear

1138 kubectl create -f pv-hp-vol-dep.yml

1139 kubectl get deployment

1140 kubectl get pods -o wide

1141 kubectl scale deployment pv-hp-vol-dep --replicas 4

1142 kubectl get pods -o wide

1143 kubectl describe pv pv1

1144 clear

1145 kubectl get pods -o wide

1146 curl 10.44.0.4:80

1147 curl 10.44.0.2:80

1148 curl 10.36.0.4:80

------------------------------------------------------------------------------

root@thmaster:~# cat cm.yml

kind: ConfigMap

apiVersion: v1

metadata:

name: testcm

namespace: default

data:

dbname: mydb

dbport: "1521"

root@thmaster:~# cat cm-dep.yml

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: cm-as-vol-dep

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: cmvolapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: cmvolapp

spec:

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

volumes:

- name: cmvol

configMap: ## emptyDir/hostPath/persistentVolumeClaim

name: filecm

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

#resources:

#env:

volumeMounts:

- name: cmvol

mountPath: /appconfig ## this is the path inside cont always

---

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: cm-as-env-vars-dep

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: cmenvapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: cmenvapp

spec:

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

#volumes:

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

#resources:

#env:

envFrom:

- configMapRef:

name: devdbconfig

#volumeMounts:

1199 vi cm.yml

1200 clear

1201 kubectl create -f cm.yml

1202 kubectl get configmap

1203 kubectl get cm

1204 ls -l

1205 cp ds1.yml ds1.xml

1206 kubectl create configmap devdbconfig --from-literal=dbhost=mydevdb --from-literal=dbport=1521

1207 kubectl get cm

1208 kubectl create configmap filecm --from-file=rs1.yml --from-file=ds1.xml

1209 kubectl get cm

1210 kubectl describe cm filecm

1211 clear

1212 history

1213 kubectl get cm

1214 clear

1215 vi cm-dep.yml

1216 clear

1217 kubectl create -f cm-dep.yml

1218 kubectl get pods -o wide

1219 kubectl get cm

1220 kubectl describe cm devdbconfig

1221 kubectl exec cm-as-env-vars-dep-5c4465689b-cfttt -- env

1222 clear

1223 kubectl get pods

1224 kubectl exec -it cm-as-vol-dep-5cb7465cb-kwxwb -- bash

---------------------------------------------------------------------

1229 echo naresh

1230 echo naresh | base64

1231 echo adminpass | base64

1232 ls -l

1233 cat ds1.xml

1234 cat ds1.xml | base64

1235 clear

1236 vi sec.yml

1237 kubectl create -f sec.yml

1238 clear

1239 kubectl get secret

1240 kubectl describe secret test-secret

1241 clear

1242 vi sec-dep.yml

1243 clewar

1244 clear

1245 kubectl create -f sec-dep.yml

1246 vi sec-dep.yml

1247 kubectl apply -f sec-dep.yml

1248 clear

1249 kubectl get pods -o wide

1250 kubectl exec sec-as-env-vars-dep-849bcd868c-4ppzb -- env

1251 clear

1252 kubectl exec -it sec-as-vol-dep-76ccc86cb6-5ndfr -- bash

root@thmaster:~# cat sec-dep.yml

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: sec-as-vol-dep

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: secvolapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: secvolapp

spec:

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

volumes:

- name: secvol

secret: ## emptyDir/hostPath/persistentVolumeClaim/ConfigMap/

secretName: test-secret

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

#resources:

#env:

volumeMounts:

- name: secvol

mountPath: /secconfig ## this is the path inside cont always

---

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: sec-as-env-vars-dep

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: secenvapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: secenvapp

spec:

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

#volumes:

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

#resources:

#env:

envFrom:

- secretRef:

name: test-secret

#volumeMounts:

root@thmaster:~# cat sec.yml

apiVersion: v1

kind: Secret

metadata:

name: test-secret

data:

username: bXktYXBw

password: Mzk1MjgkdmRnN0pi

1257 kubectl create -f https://raw.githubusercontent.com/lerndevops/educka/master/dashboard/dashboard-insecure.yml

1258 kubectl get pods -o wide

1259 kubectl get services

1260 clear

1261 git clone https://github.com/lerndevops/educka

1262 cd educka/monitoring/metrics-server/

1263 ls -l

1264 kubectl apply -f .

1265 cd

1266 clear

1267 kubectl get pods -n kube-system

1268 kubectl top nodes

1269 kubectl top pods

1270 kubectl top nodes

***DAY8 -- Security***

1305 echo $HOME

1306 cd $HOME/.kube

1307 ls -l

1308 vi config

1309 clear

1310 ls -l

1311 kubectl get nodes

1312 rm config

1313 kubectl get nodes

1314 kubectl --kubeconfig=/etc/kubernetes/admin.conf get nodes

1315 kubectl get pods

1316 kubectl --kubeconfig=/etc/kubernetes/admin.conf get pods

1317 cp /etc/kubernetes/admin.conf config

1318 kubectl get pods

1324 mkdir /home/certs

1325 cd /home/certs/

1326 ls -l

1327 openssl genrsa -out user1.key 2048

1328 ls -l

1329 cat user1.key

1330 clear

1331 openssl genrsa -out user2.key 2048

1332 ls -l

1333 openssl req -new -key user1.key -out user1.csr -subj "/CN=user1/O=devops"

1334 ls -l

1335 openssl req -new -key user2.key -out user2.csr -subj "/CN=user2/O=devops"

1336 ls -l

1337 openssl x509 -req -in user1.csr -CA /etc/kubernetes/pki/ca.crt -CAkey /etc/kubernetes/pki/ca.key -CAcreateserial -out user1.crt -days 1000

1338 openssl x509 -req -in user2.csr -CA /etc/kubernetes/pki/ca.crt -CAkey /etc/kubernetes/pki/ca.key -CAcreateserial -out user2.crt -days 1000

1339 ls -l

1340 clear

1341 cp /etc/kubernetes/admin.conf user1.conf

1342 cp /etc/kubernetes/admin.conf user2.conf

1343 ls -l

1344 vi user1.conf

1345 clear

1346 ls -l

1347 cat user1.key

1348 cat user1.key | base64

1349 clear

1350 vi user1.conf

1351 clear

1352 vi user2.conf

1353 clear

1354 vi user1.conf

1355 clear

1356 vi user2.conf

1357 clear

1358 kubectl get nodes

1359 rm /root/.kube/config

1360 kubectl get nodes

1361 kubectl --kubeconfig=/etc/kubernetes/admin.conf get nodes

1362 kubectl --kubeconfig=/home/certs/user1.conf get nodes

1363 kubectl --kubeconfig=/home/certs/user1.conf get pods

1364 kubectl --kubeconfig=/home/certs/user1.conf version --short

1365 kubectl --kubeconfig=/home/certs/user2.conf version --short

1366 kubectl --kubeconfig=/home/certs/user2.conf get pods

root@thmaster:/home/certs# cat user1.conf

apiVersion: v1

clusters:

- cluster:

certificate-authority-data: 

server: https://10.154.0.5:6443

name: kubernetes

contexts:

- context:

cluster: kubernetes

user: user1

name: user1@kubernetes

current-context: user1@kubernetes

kind: Config

preferences: {}

users:

- name: user1

user:

client-certificate: /home/certs/user1.crt

client-key: /home/certs/user1.key

root@thmaster:/home/certs# cat user2.conf

apiVersion: v1

clusters:

- cluster:

certificate-authority-data: 

server: https://10.154.0.5:6443

name: kubernetes

contexts:

- context:

cluster: kubernetes

user: user2

name: user2@kubernetes

current-context: user2@kubernetes

kind: Config

preferences: {}

users:

- name: user2

user:

client-certificate: /home/certs/user2.crt

client-key: /home/certs/user2.key

1376 kubectl --kubeconfig=/etc/kubernetes/admin.conf create namespace teama

1378 kubectl --kubeconfig=/etc/kubernetes/admin.conf create role readonly-role --verb=get,list,watch --resource=pods,deployments,services --namespace=teama

1380 kubectl --kubeconfig=/etc/kubernetes/admin.conf get roles -n teama

1381 kubectl --kubeconfig=/etc/kubernetes/admin.conf describe role readonly-role -n teama

1382 kubectl --kubeconfig=/etc/kubernetes/admin.conf create rolebinding user1-binding --role=readonly-role --user=user1 --namespace=teama

1383 kubectl --kubeconfig=/etc/kubernetes/admin.conf create rolebinding user1-binding --role=readonly-role --user=user1 --namespace=teama --dry-run=client -o yaml

1384 kubectl --kubeconfig=/home/certs/user1.conf get pods

1385 kubectl --kubeconfig=/home/certs/user1.conf get pods -n teama

1386 kubectl --kubeconfig=/home/certs/user1.conf get deployment -n teama

1387 kubectl --kubeconfig=/home/certs/user1.conf get services -n teama

1388 kubectl --kubeconfig=/home/certs/user1.conf get ds -n teama

1389 kubectl --kubeconfig=/home/certs/user1.conf get pvc -n teama

1390 kubectl --kubeconfig=/home/certs/user1.conf get cm -n teama

1396 kubectl explain role

1397 kubectl --kubeconfig=/etc/kubernetes/admin.conf explain role

1398 kubectl --kubeconfig=/etc/kubernetes/admin.conf explain role.rules

1399 kubectl --kubeconfig=/etc/kubernetes/admin.conf explain role.rules.verbs

1400 clear

1401 kubectl --kubeconfig=/etc/kubernetes/admin.conf api-resources

1402 clear

1403 kubectl --kubeconfig=/etc/kubernetes/admin.conf api-resources | grep -i true

1404 kubectl --kubeconfig=/etc/kubernetes/admin.conf api-resources | grep -i false

1405 clear

1406 kubectl --kubeconfig=/etc/kubernetes/admin.conf create clusterrole readonly-cr-role --verb=get,list,watch --resource=nodes,namespaces,pods,services

1407 kubectl --kubeconfig=/etc/kubernetes/admin.conf create clusterrole readonly-cr-role --verb=get,list,watch --resource=nodes,namespaces,pods,services --dry-run=client -o yaml

1408 kubectl --kubeconfig=/etc/kubernetes/admin.conf create cluterrolebinding user2-binding --clusterrole=readonly-cr-role --user=user2

1409 kubectl --kubeconfig=/etc/kubernetes/admin.conf create clusterrolebinding user2-binding --clusterrole=readonly-cr-role --user=user2

1410 kubectl --kubeconfig=/etc/kubernetes/admin.conf create clusterrolebinding user2-binding --clusterrole=readonly-cr-role --user=user2 --dry-run=client -o yaml

1411 kubectl --kubeconfig=/home/certs/user2.conf get pods

1412 kubectl --kubeconfig=/home/certs/user2.conf get pods -n kube-system

1413 kubectl --kubeconfig=/home/certs/user2.conf get pods --all-namespaces

1414 kubectl --kubeconfig=/home/certs/user2.conf get nodes

1415 kubectl --kubeconfig=/home/certs/user2.conf get ns

1416 kubectl --kubeconfig=/home/certs/user2.conf get pv

1417 kubectl --kubeconfig=/home/certs/user2.conf run testpod --image=httpd

1418 kubectl --kubeconfig=/home/certs/user2.conf api-resources

1419 history

1420 kubectl --kubeconfig=/etc/kubernetes/admin.conf get clusterroles

1421 kubectl --kubeconfig=/etc/kubernetes/admin.conf get clusterrolebindings

1422 kubectl --kubeconfig=/etc/kubernetes/admin.conf describe clusterrole cluster-admin

1423 kubectl --kubeconfig=/etc/kubernetes/admin.conf describe clusterrole admin

1424 history

1425 kubectl --kubeconfig=/etc/kubernetes/admin.conf create clusterrolebinding user2-binding --clusterrole=cluster-admin --group=devops

1426 kubectl --kubeconfig=/etc/kubernetes/admin.conf create clusterrolebinding devops-group-binding --clusterrole=cluster-admin --group=devops

1427 kubectl --kubeconfig=/home/certs/user2.conf run testpod --image=httpd

1428 kubectl --kubeconfig=/home/certs/user2.conf create deployment --image=nginx -n teama

1429 kubectl --kubeconfig=/home/certs/user2.conf create deployment test --image=nginx -n teama

1430 kubectl --kubeconfig=/home/certs/user2.conf delete deploy test -n teama

1431 kubectl --kubeconfig=/home/certs/user1.conf create deployment test --image=nginx

1432 kubectl --kubeconfig=/home/certs/user1.conf get pods --all-namespaces

1435 cp /etc/kubernetes/admin.conf /root/.kube/config

1436 kubectl get serviceaccounts

1437 kubectl get sa

1438 kubectl create sa test

1439 kubectl get sa

1440 kubectl describe sa test

1441 kubectl describe secret test-token-c24x9

1442 kubectl get secret/test-token-c24x9 -o jsonpath='{.data.namespace}'

1443 kubectl get secret/test-token-c24x9 -o jsonpath='{.data.namespace}' | base64 -d

1444 kubectl -n kube-system get secret/test-token-c24x9 -o jsonpath='{.data.token}' | base64 --decode

1445 kubectl get secret/test-token-c24x9 -o jsonpath='{.data.token}' | base64 --decode

1446 kubectl get services

1447 clear

1448 kubectl get cluterrolebindings

1449 kubectl get clusterrolebindings

1450 kubectl delete clusterrolebindings kubernetes-dashboard

1451 history

1453 kubectl --kubeconfig=/etc/kubernetes/admin.conf create clusterrolebinding kube-dash-binding --clusterrole=readonly-cr-role --serviceaccount=default:kubernetes-dashboard

1457 kubectl apply -f https://raw.githubusercontent.com/lerndevops/microservices-demo/master/deploy/kubernetes/complete-demo.yaml

1458 kubectl get pods -n sock-shop

1460 kubectl get pods -n sock-shop

1461 kubectl get svc -n sock-shop

1462 kubectl get pods -n sock-shop

1463 kubectl logs user-846f474c46-cshvr

1464 kubectl logs user-846f474c46-cshvr -n sock-shop

1465 kubectl get pods -n sock-shop

1466 kubectl logs user-db-5f68d7b558-6fvzp -n sock-shop

1467 kubectl get pods -n sock-shop -o wide

DAY9

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: nodename-dep

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: myapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: myapp

spec:

nodeName: thnode01

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

#volumes:

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

#resources:

#env:

#volumeMounts:

---

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: ns-dep1

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: myapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: myapp

spec:

#nodeName: thnode01

nodeSelector:

role: app

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

#volumes:

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

#resources:

#env:

#volumeMounts:

---

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: ns-dep2

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: myapp

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: myapp

spec:

#nodeName: thnode01

nodeSelector:

role: app

env: qa

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

#volumes:

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

#resources:

#env:

#volumeMounts:

1480 kubectl get pods -n sock-shop

1481 kubectl delete all --all -n sock-shop

1482 clear

1483 history

1484 clear

1485 kubectl get nodes

1486 vi nn-dep.yml

1487 clear

1488 kubectl create -f nn-dep.yml

1489 kubectl get pods -o wide

1490 kubectl scale deployment nodename-dep --replicas 6

1491 kubectl get pods -o wide

1492 clear

1493 kubectl get nodes

1494 kubectl describe node thnode01

1495 clear

1496 kubectl label node thnode01 role=app

1497 kubectl label node thnode02 role=app

1498 kubectl label node thnode02 env=dev

1499 kubectl label node thnode02 env=qa

1500 kubectl label node thnode01 env=qa

1501 kubectl describe node thnode01

1502 vi ns-dep1.yml

1503 clear

1504 kubectl create -f ns-dep1.yml

1505 kubectl get pods -o wide

1506 kubectl scale deployment ns-dep1 --replicas 10

1507 kubectl get pods -o wide

1508 vi ns-dep2.yml

1509 kubectl create -f ns-dep2.yml

1510 kubectl get pods -o wide

1511 kubectl scale deployment ns-dep2 --replicas 10

1512 kubectl get pods -o wide

1513 kubectl get nodes

1514 kubectl label node thnode1 abc=bbc

1515 kubectl label node thnode01 abc=bbc

1516 kubectl label node thnode01 abc-

1517 kubectl label node thnode01 role-

1518 kubectl label node thnode01 env-

1519 kubectl label node thnode02 env-

1520 kubectl label node thnode02 role-

1521 kubectl get nodes

1522 kubectl scale deployment ns-dep2 --replicas 12

1523 kubectl get pods -o wide

1524 kubectl describe pod ns-dep2-7c6f58ff58-r5xmh

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: tt-dep1

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: happ

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: happ

spec:

#nodeName: thnode01

#nodeSelector

tolerations:

- key: role

operator: "Equal" # Exists

value: db

effect: NoSchedule

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

#volumes:

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

#resources:

#env:

#volumeMounts:

1527 kubectl get nodes

1528 kubectl describe node thnode01

1529 clear

1530 kubectl describe node thnode01 | grep -i taints

1531 kubectl describe node thnode02 | grep -i taints

1532 kubectl describe node thmaster | grep -i taints

1533 kubectl taint node thmaster role=db:NoSchedule

1534 kubectl taint node thnode01 role=db:NoSchedule

1535 kubectl taint node thnode02 role=db:NoSchedule

1536 kubectl describe node thnode01 | grep -i taints

1537 kubectl describe node thnode02 | grep -i taints

1538 kubectl describe node thmaster | grep -i taints

1539 kubectl get node -o wide

1540 kubectl get pods -o wide

1541 kubectl create deployment testdep --image=httpd

1542 kubectl get pods -o wide

1543 kubectl scale deployment testdep --replicas 4

1544 kubectl get pods -o wide

1545 kubectl describe pod testdep-98868866c-5tbgj

1546 clear

1547 vi tt-dep1.yml

1548 clear

1549 kubectl create -f tt-dep1.yml

1550 vi tt-dep1.yml

1551 clear

1552 kubectl create -f tt-dep1.yml

1553 kubectl get pods -o wide

kind: Deployment

apiVersion: apps/v1

metadata: ## this is replicaset metadata

name: tt-dep2

namespace: default

##labels: labels for controller definition are not mandatory, if you want to write you can write

spec: ## this is replicset spec

replicas: 2 # if you dont mention the replicas by default it will create 1 replica

selector:

matchLabels:

type: happ

template: ## this describes the pod definition that include always metadata & spec of the pod

metadata: ## this is the pod meteadata where we can define the pod labels

#name: test ## the name of the pod randomly created by kube, we need not to define the name

labels:

type: happ

spec:

#nodeName: thnode01

#nodeSelector

tolerations:

- key: role

operator: "Equal" # Exists

value: db

effect: NoSchedule

- key: zone

operator: "Equal"

value: useast

effect: "NoExecute"

restartPolicy: Always

terminationGracePeriodSeconds: 0 ## forceful delete, the default time kube waits before deleting the pod is 30sec

#intiContainers:

#volumes:

containers:

- name: cont1

image: nginx

ports:

- name: http

containerPort: 80

#resources:

#env:

#volumeMounts:

1562 vi tt-dep2.yml

1563 clear

1564 kubectl create -f tt-dep2.yml

1565 kubectl get pods -o wide | grep thnode01

1566 kubectl describe pod tt-dep2-ff7685d9f-6cm8c

1567 kubectl get pods -o wide

1568 kubectl get deployment

1569 kubectl scale deployment nodename-dep --replicas 1

1570 kubectl scale deployment ns-dep1 --replicas 1

1571 kubectl scale deployment ns-dep2 --replicas 1

1572 kubectl get pods -o wide

1573 kubectl delete deployment --all

1574 kubectl get pods -o wide

1575 kubectl create -f tt-dep2.yml

1576 kubectl get pods -o wide

1579 kubectl taint node thnode01 role=db:NoSchedule-

1580 kubectl taint node thnode02 role=db:NoSchedule-

1581 kubectl taint node thnode02 zone=useast:NoExecute-

1582 kubectl taint node thnode01 zone=useast:NoExecute-

Networking

1587 kubectl run mypod --image=httpd

1588 kubectl get pods -o wide

1589 kubectl run mypod1 --image=nginx

1590 kubectl get pods -o wide

1591 clear

1592 kubectl get pods -n kube-system

1593 ifconfig

1594 ip route

1595 clear

1596 kubectl create deployment pingtest --image=lerndevops/netshoot:sleep

1597 kubectl scale deployment pingtest --replicas 6

1598 kubectl get pods -o wide

1599 kubebctl exec pingtest-84bc89889b-2nc5t -- ping 10.36.0.5

1600 kubectl exec pingtest-84bc89889b-2nc5t -- ping 10.36.0.5

1601 kubectl exec pingtest-84bc89889b-2nc5t -- ping 10.44.0.3

1609 ping 10.154.0.6

1610 vi /etc/hosts

1611 ping naresh

1612 cat /etc/hosts

1613 clear

1614 cat /etc/hosts

1615 ping thnode01

1616 clear

1617 cat /etc/resolv.conf

1618 clear

1619 kubectl get pods -n kube-system

1620 kubectl get pods -o wide

1621 kubectl get services -o wide

1622 clear

1623 kubectl run dnstest --image=lerndevops/netshoot:sleep

1624 kubectl get pods -o wide

1625 kubectl exec -it -- dnstest

1626 kubectl exec -it dnstest -- bash

1627 kubectl get services -n kube-system

1628 kubectl describe service 10.96.0.10 -n kube-system

1629 kubectl describe service kube-dns -n kube-system

1630 kubectl get pods -o wide -n kube-system | grep -i coredns

1631 kubectl exec -it dnstest -- bash

1632 kubectl get services

1633 kubectl exec -it dnstest -- bash

1634 clear

1635 kubectl get ns

1636 kubectl create ns teamb

1637 kubectl run nginx --image=nginx -n default

1638 kubectl run htpd --image=httpd -n teama

1639 kubectl run pyapp --image=lerndevops/samplepyapp:v1 -n teamb

1640 kubectl get pods -o wide --all-namespaces

1641 vi svc1.yml

1642 kubectl get pods -o wide --all-namespaces --show-labels

1643 vi svc1.yml

1644 kubectl create -f svc1.yml

1645 kubectl get all -n default

1646 kubectl get all -n teama

1647 kubectl get all -n teamb

1648 clear

1649 kubectl exec -it dnstest -- bash

1650 clear

1651 kubectl apply -f https://raw.githubusercontent.com/DevOpsUniversity-DU/Certified-DevOps-Foundation/master/kube/controllers/myapp.yml

1652 kubectl get pods -o wide

1653 kubectl get services

1654 kubectl exec myapp-6498959f5c-8z226 -- ps -ef|grep java

1655 kubectl get pods

1656 kubectl logs mongodb-5d6d78669d-8ftk2

1657\* kubectl get pods -o wid

1658 kubectl logs myapp-6498959f5c-8z226

1659 kubectl get deployment

1660 kubectl scale deployment mongodb --replicas 0

1661 kubectl get deployment

1662 kubectl describe svc mongo

1663 kubectl scale deployment mongodb --replicas 1

1664 kubectl describe svc mongo

root@thmaster:~# cat svc1.yml

kind: Service

apiVersion: v1 ## kubectl api-resources

metadata:

name: svc1

namespace: default

spec:

type: ClusterIP # it acts a internal virtual load balancer & forwards the requests into multiple backend pods

selector:

run: nginx # this is the labels of the pods which will be the backends,

ports:

- port: 80 ## this is your Virtual LB port

targetPort: 80

---

kind: Service

apiVersion: v1 ## kubectl api-resources

metadata:

name: svc1

namespace: teama

spec:

type: ClusterIP # it acts a internal virtual load balancer & forwards the requests into multiple backend pods

selector:

run: htpd # this is the labels of the pods which will be the backends,

ports:

- port: 80 ## this is your Virtual LB port

targetPort: 80

---

kind: Service

apiVersion: v1 ## kubectl api-resources

metadata:

name: svc1

namespace: teamb

spec:

type: ClusterIP # it acts a internal virtual load balancer & forwards the requests into multiple backend pods

selector:

run: pyapp # this is the labels of the pods which will be the backends,

ports:

- port: 80 ## this is your Virtual LB port

targetPort: 3000

root@thmaster:~# kubectl exec -it dnstest -- bash

bash-5.0# history

1 hostname -i

3 cat /etc/resolv.conf

5 nslookup testapp

6 curl svc1

7 nslookup svc1

8 curl svc1.default.svc.cluster.local:80

9 curl svc1.teama.svc.cluster.local:80

10 curl svc1.teamb.svc.cluster.local:80 ; echo

***CleanUP Cluster for Istio Setup:***

1668 kubectl get ns

1669 kubectl delete ns sock-shop teama teamb kubernetes-dashboard

1670 kubectl delete all --all -n default

***Setup Istio:***

1757 git clone https://github.com/lerndevops/educka

1758 cd educka/

1759 git pull

1760 cd istio/

1761 ls -l

1762 kubectl apply -f istio-init-1.10.3.yml

1763 ls -l

1764 kubectl get pods -n istio-system

1765 kubectl

1766 kubectl get pods -n istio-system

1767 ls -l

1768 kubectl apply -f kiali.yml

1769 kubectl apply -f prometheus.yaml

1770 kubectl apply -f grafana.yaml

1771 kubectl apply -f jaeger.yaml

While running above command if you see something like “unable to recognize” error re run the same above one more time to see that goes smooth second run

root@thmaster:~/educka/istio# kubectl get pods -n istio-system

NAME READY STATUS RESTARTS AGE

grafana-56d978ff77-n2q82 1/1 Running 0 4m18s

istio-egressgateway-5547fcc8fc-kgl7n 1/1 Running 0 5m12s

istio-ingressgateway-8f568d595-z44jx 1/1 Running 0 5m12s

istiod-6659979bdf-64ss5 1/1 Running 0 5m12s

kiali-5bb9c9cf49-p2s6d 1/1 Running 0 4m49s

prometheus-8958b965-fqfxz 2/2 Running 0 4m28s

1779 kubectl apply -f 1-fleemanapp-full-stack.yml

1780 kubectl get pods -o wide

1781 vi 1-fleemanapp-full-stack.yml

1782 clear

1783 kubectl get pods

1784 kubectl get ns

1785 kubectl get ns default -o yaml

1786 clear

1787 ls -l

1788 more label-default-namespace.yml

1789 kubectl apply -f label-default-namespace.yml

1790 kubectl get ns default -o yaml

1791 clear

1792 kubectl get pods

1793 kubectl delete pods --all --force

1794 kubectl get pods