GitHub Repo: <https://github.com/lerndevops/educka>

setup hostname:

sudo hostnamectl set-hostname master-node

Install Documents:

<https://github.com/lerndevops/educka/tree/master/1-intall>

Install & setup kubernetes cluster:

<https://github.com/lerndevops/educka/blob/master/1-intall/install-kubernetes-on-ubuntu-debian.md>

39 sudo hostnamectl set-hostname master-node

40 exit

41 clear

42 docker -v

43 kubeadm version -o short

44 kubelet --version

45 kubectl version --short --client

46 clear

47 sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installDocker.sh -P /tmp

48 sudo chmod 755 /tmp/installDocker.sh

49 sudo bash /tmp/installDocker.sh

50 docker -v

51 sudo bash /tmp/installDocker.sh

52 docker -v

53 sudo systemctl restart docker.service

54 sudo systemctl status docker.service

55 sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installCRIDockerd.sh -P /tmp

56 sudo chmod 755 /tmp/installCRIDockerd.sh

57 sudo bash /tmp/installCRIDockerd.sh

58 sudo systemctl restart cri-docker.service

59 sudo systemctl status cri-docker.service

60 sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installK8S.sh -P /tmp

61 sudo chmod 755 /tmp/installK8S.sh

62 sudo bash /tmp/installK8S.sh

65 docker -v

66 cri-dockerd --version

67 kubeadm version -o short

68 sudo chmod 755 /tmp/installK8S.sh

69 sudo bash /tmp/installK8S.sh # if this script fails follow below

if you sitll see the older kube version follow below steps again

70 sudo apt-mark unhold kubeadm kubectl kubelet

71 sudo bash /tmp/installK8S.sh

labsuser@master-node:~$ kubeadm version -o short

v1.26.3

labsuser@master-node:~$ kubelet --version

Kubernetes v1.26.3

labsuser@master-node:~$ kubectl version --short --client

Flag --short has been deprecated, and will be removed in the future. The --short output will become the default.

Client Version: v1.26.3

Kustomize Version: v4.5.7

labsuser@master-node:~$ docker -v

Docker version 23.0.2, build 569dd73

labsuser@master-node:~$ cri-dockerd --version

cri-dockerd 0.3.0 (0de30fc)

## Initialize kubernetes Master Node

sudo kubeadm init --cri-socket unix:///var/run/cri-dockerd.sock --ignore-preflight-errors=all

sudo mkdir -p $HOME/.kube

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

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

84 kubectl get nodes

## install networking driver -- Weave/flannel/canal/calico etc...

## below installs calico networking driver

kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.24.1/manifests/calico.yaml

# Validate: kubectl get nodes

labsuser@master-node:~$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

master-node Ready control-plane 4m18s v1.26.3

once the master node is ready / do below on both worker nodes

39 sudo hostnamectl set-hostname worker-node01

40 exit

41 sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installDocker.sh -P /tmp

42 sudo chmod 755 /tmp/installDocker.sh

43 sudo bash /tmp/installDocker.sh

44 ## Install CRI-Docker

45 sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installCRIDockerd.sh -P /tmp

46 sudo chmod 755 /tmp/installCRIDockerd.sh

47 sudo bash /tmp/installCRIDockerd.sh

48 sudo systemctl restart cri-docker.service

49 sudo wget https://raw.githubusercontent.com/lerndevops/labs/master/scripts/installK8S.sh -P /tmp

50 sudo chmod 755 /tmp/installK8S.sh

51 sudo bash /tmp/installK8S.sh

52 sudo apt-mark unhold kubeadm kubectl kubelet

53 sudo bash /tmp/installK8S.sh

54 clear

55 docker -v

56 cri-dockerd --version

57 kubeadm version -o short

58 kubelet --version

59 kubectl version --client --short

60 sudo kubeadm join 172.31.51.197:6443 --cri-socket unix:///var/run/cri-dockerd.sock --token 7t92l9.vsb5aaistejxj79l --discovery-token-ca-cert-hash sha256:1dddeda8e260744c7054877c2582134bbdc1f7ffefd8d9cb4e8a0fd4a55b4b13

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

107 clear

108 kubectl get pods

109 kubectl run webpod --image=nginx

110 kubectl run webpod1 --image=nginx

111 kubectl run dbpod --image=mong

112 kubectl get pods -o wide

113 kubectl describe pod dbpod

114 kubectl get pods -o wide

115 kubectl logs webpod

116 kubectl exec -it webpod -- /bin/bash

119 kubectl delete pod dbpod

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

126 kubectl get namespaces

127 kubectl describe pod webpod

128 clear

129 kubectl get namespaces

130 kubectl create namespace teama

131 kubectl create namespace teamb

132 kubectl create namespace qa

133 kubectl create namespace uat

134 kubectl create namespace frontend

135 kubectl create namespace backend

136 kubectl get namespaces

137 clear

138 kubectl get pods -o wide

139 kubectl get pods -o wide --namespace default

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

141 kubectl get pods -o wide --namespace teama

142 kubectl get pods -o wide --namespace teamb

143 clear

144 kubectl run pod1 --image=httpd

145 kubectl get pods -o wide --namespace default

146 kubectl run pod1 --image=httpd --namespace teama

147 kubectl run pod1 --image=httpd --namespace teamb

148 kubectl run pod1 --image=httpd --namespace uat

149 kubectl describe pod pod1

150 kubectl describe pod pod1 --namespace uat

151 kubectl delete pod pod1

152 kubectl get pods -o wide --namespace default

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

154 kubectl run pod1 --image=httpd --namespace teamb

155 kubectl get pods -o wide -A

156 kubectl get pods -o wide -n default

**DAY2**

175 kubectl api-resources

176 clear

177 kubectl get pods -o wide

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

179 kubectl get pods -o wide --show-labels -A

180 clear

181 kubectl get pods -o wide --show-labels -A -l run

182 kubectl get pods -o wide --show-labels -A -l run=pod1

183 kubectl get pods -o wide --show-labels --all-namespaces -l run=pod1

184 vi pody.yaml

185 clear

186 ls -l

187 kubectl apply -f pody.yaml

188 kubectl get pods -o wide

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

190 kubectl describe pod pody

in any yaml 99.99% of the time keys are pre defined by the technology (kubernetes)

90% of the time you write your own values

10% chances where both key & value both are pre defined

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

kubectl run podx --image=nginx --namespace default -l app=frontend

kind: Pod # kubectl api-resources

apiVersion: v1

metadata:

name: pody

namespace: default

labels: # are mandatory # like tags # helps to identify a group - if you do not assign a label to pod kube will assing one

app: frontend # here both key & value are your choice

env: prod

team: teama

spec:

containers:

- name: cont1 # the name can be anything

image: docker.io/library/nginx:latest

#- name: cont2

# image: tomcat

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

195 cat pody.yaml

196 kubectl get pods pody

197 kubectl get pods pody -o yaml

198 clear

199 kubectl get pods pody -o json

200 clear

201 kubectl get pods -o wide

202 kubectl run hellopod --image=nginx --namespace default -l app=test --dry-run=client -o yaml

203 kubectl run hellopod --image=nginx --namespace default -l app=test --dry-run=client -o json

204 kubectl run hellopod --image=nginx --namespace default -l app=test --dry-run=client -o json > hellopod.json

205 ls -l

206 cat hellopod.json

207 kubectl apply -f hellopod.json

​​ 210 vi multi1.yaml

211 kubectl apply -f multi1.yaml

212 kubectl get pods -o wide

213 kubectl describe pod multi-cont-pod

214 vi multi2.yaml

215 kubectl apply -f multi2.yaml

216 kubectl get pods -o wide

217 clear

218 kubectl get pods -o wide

219 kubectl logs podx

220 kubectl logs multi-cont-pod2 -c cont1

221 kubectl logs multi-cont-pod2 -c cont2

222 kubectl logs multi-cont-pod2 -c cont3

223 kubectl exec -it multi-cont-pod2 -c cont1 -- /bin/bash

224 kubectl exec -it multi-cont-pod2 -c cont2 -- /bin/bash

225 kubectl exec -it multi-cont-pod2 -c cont3 -- /bin/bash

226 clear

227 kubectl get pods -o wide

228 curl 192.168.87.202:80

229 curl 192.168.87.202:8080

kind: Pod # kubectl api-resources

apiVersion: v1

metadata:

name: multi-cont-pod

namespace: default

labels: # are mandatory # like tags # helps to identify a group - if you do not assign a label to pod kube will assing one

app: frontend # here both key & value are your choice

env: prod

team: teama

spec:

restartPolicy: Always

containers:

- name: cont1 # the name can be anything

image: docker.io/library/nginx:latest

- name: cont2

image: tomcat

- name: cont3

image: ubuntu

kind: Pod # kubectl api-resources

apiVersion: v1

metadata:

name: multi-cont-pod2

namespace: default

labels: # are mandatory # like tags # helps to identify a group - if you do not assign a label to pod kube will assing one

app: frontend # here both key & value are your choice

env: prod

team: teama

spec:

restartPolicy: Always

containers:

- name: cont1 # the name can be anything

image: docker.io/library/nginx:latest # 80

- name: cont2

image: tomcat # 8080

- name: cont3

image: ubuntu

command: ["bash", "-c", "sleep 6000"]

231 kubectl apply -f https://raw.githubusercontent.com/lerndevops/educka/master/2-pods/multi/sidecar.yml

232 kubectl get pods -o wide

233 curl 192.168.158.8:80

237 kubectl get pods -o wide

kind: Pod # kubectl api-resources

apiVersion: v1

metadata:

name: init-cont-pod1

namespace: default

labels: # are mandatory # like tags # helps to identify a group - if you do not assign a label to pod kube will assing one

app: frontend # here both key & value are your choice

env: prod

team: teama

spec:

restartPolicy: Always

initContainers:

- name: initcont1 # the name can be anyting

image: ubuntu

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

- name: initcont2

image: ubuntu

command: ["bash", "-c", "git clone https://github.com/appconfig"]

containers: # app containers

- name: cont1 # the name can be anything

image: docker.io/library/nginx:latest # 80

240 vi init1.yaml

241 kubectl apply -f init1.yaml

242 watch kubectl get pods -o wide

243 kubectl get pods -o wide

244 kubectl describe pod init-cont-pod1

247 kubectl get pods -o wide -n kube-system

248 ls

249 cat pody.yaml

250 clear

251 kubectl get pods -o wide

252 kubectl delete pod pody-worker-node01

253 kubectl get pods -o wide

254 kubectl get pods -n kube-system

255 kubectl get pods -n kube-system -o wide

256 sudo ls -l /etc/kubernetes/manifests/

257 ls -l

258 sudo cp pody.yaml /etc/kubernetes/manifests/

259 kubectl get pods -o wide

On worker node

17 sudo su -

3 systemctl status kubelet

4 cd /var/lib/kubelet/

5 ls -

6 ls -l

7 vi config.yaml

8 clear

9 cd /etc/kubernetes/manifests

10 ls -l

11 vi pody.yaml

12 ls -l

13 more pody.yaml

14 vi pody.yaml

15 mv pody.yaml /tmp

16 ls -l

18 history

<https://github.com/lerndevops/educka/blob/master/2-pods/__lab.md>

**DAY3**

**25 kubectl api-resources**

**26 cat pody.yaml**

**27 clear**

**28 vi rs.yaml**

**29 kubectl apply -f rs.yaml**

**30 vi rs.yaml**

**31 kubectl apply -f rs.yaml**

**32 kubectl get replicaset -o wide**

**33 kubectl get rs -o wide**

**34 kubectl get pods --show-labels**

**35 kubectl delete pod pyapp-8dxr2 pyapp-ccg62 pyapp-sfm8k pyapp-szlnv**

**36 kubectl get rs -o wide**

**37 kubectl get pods --show-labels**

**38 kubectl scale replicaset pyapp --replicas 10**

**39 kubectl get rs -o wide**

**40 kubectl get pods --show-labels**

**41 kubectl scale replicaset pyapp --replicas 3**

**42 kubectl get pods --show-labels**

**43 kubectl get rs -o wide**

**44 kubectl delete replicaset pyapp**

**45 kubectl get rs -o wide**

**46 kubectl get pods --show-labels**

**47 kubectl apply -f rs.yaml**

**kind: ReplicaSet # kubectl api-resources**

**apiVersion: apps/v1**

**metadata:**

**name: pyapp**

**namespace: default**

**#labels: # are optional**

**spec:**

**replicas: 4 # the total number of pods to be created / if we do not write replicas here it will create 1 one pod & manages it**

**template: # which pod to be created**

**metadata:**

**#name: hellopod,pod1,ht # kube will generate a random unique name for each pod automatically**

**labels: # are mandatory**

**app: pyapp**

**spec:**

**#restartPolicy: Always**

**#initConainers:**

**containers:**

**- name: cont1**

**image: docker.io/lerndevops/samples:pyapp-v1**

**selector: # is mandatory # which pod to controlled / managed by controller pyapp**

**matchLabels:**

**app: pyapp**

**53 vi ds.yaml**

**54 kubectl apply -f ds.yaml**

**55 kubectl get daemonset -o wide**

**56 kubectl get pods -o wide**

**57 kubectl get nodes -o wide**

**58 kubectl scale daemonset fluentdapp --replicas 6 # expected to fail**

**59 kubectl get ds -n kube-system -o wide**

**60 kubectl get ds**

**61 kubectl get nodes**

**62 kubectl describe node master-node**

**63 clear**

**64 kubectl get ds -n kube-system -o wide**

**65 kubectl get ds**

**66 kubectl describe node master-node | grep -i taints**

**67 kubectl taint node master-node node-role.kubernetes.io/control-plane:NoSchedule- ## delete the condition**

**68 kubectl describe node master-node | grep -i taints**

**69 kubectl get ds**

**70 kubectl get pods -o wide**

**71 kubectl taint node master-node node-role.kubernetes.io/control-plane:NoSchedule ## add the condition back**

**72 kubectl describe node master-node | grep -i taints**

**kind: DaemonSet # kubectl api-resources**

**apiVersion: apps/v1**

**metadata:**

**name: fluentdapp**

**namespace: default**

**#labels: # are optional**

**spec:**

**template: # which pod to be created**

**metadata:**

**#name: hellopod,pod1,ht # kube will generate a randon unique name for each pod automatically**

**labels: # are mandatory**

**app: fluentd**

**spec:**

**terminationGracePeriodSeconds: 0 # default is 30 seconds # 0 is forceful delete immediatly**

**#restartPolicy: Always**

**#initConainers:**

**containers:**

**- name: cont1**

**image: quay.io/fluentd\_elasticsearch/fluentd:v2.5.2**

**selector: # is mandatory # which pod to controlled / managed by controller pyapp**

**matchLabels:**

**app: fluentd**

**82 kubectl get pods -o wide --show-labels**

**83 vi pyapp-svc.yaml**

**84 kubectl apply -f pyapp-svc.yaml**

**85 kubectl get services -o wide**

**86 kubectl get pods -o wide --show-labels -l app=pyapp**

**87 kubectl describe service pyapp**

**88 curl 192.168.158.51:3000 ; echo**

**89 curl 192.168.158.52:3000 ; echo**

**91 curl 10.96.28.97:80 ; echo**

**92 kubectl get pods -o wide --show-labels -l app=pyapp**

**93 kubectl describe service pyapp**

**94 kubectl scale replicaset pyapp --replicas 2**

**95 kubectl get pods -o wide --show-labels -l app=pyapp**

**96 kubectl describe service pyapp**

**97 kubectl scale replicaset pyapp --replicas 6**

**98 kubectl get pods -o wide --show-labels -l app=pyapp**

**99 kubectl describe service pyapp**

**kind: Service**

**apiVersion: v1**

**metadata:**

**name: pyapp**

**namespace: default**

**#labels: # are optional**

**spec:**

**type: ClusterIP # it is an internal virtual load balancer(kube will generate a LB IP) that can forward the req into single/group of pods**

**selector: # helps the services to identify pods in the cluster**

**app: pyapp**

**ports:**

**- name: http # the name can be anything**

**port: 80 # this is the port we will use with the clusterip always to access the app**

**targetPort: 3000 # this is the port of the app inside the cont**

**kind: Service**

**apiVersion: v1**

**metadata:**

**name: pyapp**

**namespace: default**

**#labels: # are optional**

**spec:**

**type: NodePort # it will publish a nodport/hostport/vmport on every node in the cluster then it also create an**

**# internal virutal load balancer(kube will generate a LB IP) that can forward the req into single/group of pods then**

**# it maps the nodeport to clusterip**

**selector: # helps the services to identify pods in the cluster**

**app: pyapp**

**ports:**

**- name: http # the name can be anything**

**port: 80 # this is the port we will use with the clusterip always to access the app**

**targetPort: 3000 # this is the port of the app inside the cont**

**#nodePort: 30001 # this will be published port on every node**

**103 vi np-svc.yaml**

**104 kubectl apply -f np-svc.yaml**

**105 kubectl get services -o wide**

**106 curl 10.99.14.80:80 ; echo**

**kind: Service**

**apiVersion: v1**

**metadata:**

**name: pyapp-lb**

**namespace: default**

**#labels: # are optional**

**spec:**

**type: LoadBalancer # it will publish a nodport/hostport/vmport on every node in the cluster then it also create an**

**# internal virutal load balancer(kube will generate a LB IP) that can forward the req into single/group of pods then**

**# it maps the nodeport to clusterip**

**selector: # helps the services to identify pods in the cluster**

**app: pyapp**

**ports:**

**- name: http # the name can be anything**

**port: 80 # this is the port we will use with the clusterip always to access the app**

**targetPort: 3000 # this is the port of the app inside the cont**

**#nodePort: 30001 # this will be published port on every node**

**112 kubectl get pods -n kube-system -o wide**

**113 vi lb-svc.yaml**

**114 kubectl apply -f lb-svc.yaml**

**115 kubectl get services**

**DAY4**

[**https://github.com/lerndevops/educka/blob/master/3-controllers/deployments/deployment-ex3.yml**](https://github.com/lerndevops/educka/blob/master/3-controllers/deployments/deployment-ex3.yml)

[**https://github.com/lerndevops/educka/blob/master/3-controllers/deployments/rolling-update.md**](https://github.com/lerndevops/educka/blob/master/3-controllers/deployments/rolling-update.md)

**134 vi deployment.yaml**

**135 kubectl apply -f deployment.yaml**

**136 kubectl get pods -o wide**

**137 clear**

**138 kubectl get deployment -o wide**

**139 kubectl get replicaset -o wide**

**140 kubectl get pods -o wide**

**141 vi deployment.yaml**

**142 clear**

**143 cat deployment.yaml**

**144 kubectl get services**

**145 cat deployment.yaml**

**146 clear**

**147 kubectl apply -f deployment.yaml**

**148 kubectl rollout status deployment kubeserve**

**149 kubectl get deployment**

**150 vi deployment.yaml**

**151 kubectl apply -f deployment.yaml**

**152 kubectl rollout history deployment kubeserve**

**153 kubectl rollout undo deployment kubeserve**

**154 kubectl get pods -n kube-system**

**155 kubectl rollout undo deployment kubeserve**

**156 kubectl rollout history deployment kubeserve**

**157 vi deployment.yaml**

**158 kubectl rollout undo deployment kubeserve --to-revision 1 --record**

**159 kubectl rollout undo deployment kubeserve --to-revision 1**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: kubeserve**

**spec:**

**replicas: 10**

**revisionHistoryLimit: 30**

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

**strategy:**

**type: RollingUpdate**

**rollingUpdate:**

**maxUnavailable: 1**

**maxSurge: 2**

**selector:**

**matchLabels:**

**app: kubeserve**

**template:**

**metadata:**

**name: kubeserve**

**labels:**

**app: kubeserve**

**spec:**

**containers:**

**- image: leaddevops/kubeserve:v1**

**name: app**

**---**

**kind: Service**

**apiVersion: v1**

**metadata:**

**name: kubeserve-svc**

**spec:**

**type: NodePort**

**selector:**

**app: kubeserve**

**ports:**

**- port: 80**

**targetPort: 80**

**devops@master-node:~$ cat deploy2.yaml**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: mydeployment**

**labels:**

**app: myapp**

**spec:**

**replicas: 3**

**strategy:**

**type: Recreate**

**selector:**

**matchLabels:**

**app: myapp**

**template:**

**metadata:**

**labels:**

**app: myapp**

**spec:**

**containers:**

**- name: myapp-cont**

**image: lerndevops/samplepyapp:v2**

**ports:**

**- containerPort: 3000**

**162 kubectl scale deployment kubeserve --replicas 15**

**163 kubectl get pod**

**164 kubectl scale deployment kubeserve --replicas 5**

**165 vi deploy2.yaml**

**166 kubectl apply -f deploy2.yaml**

**167 kubectl get deployment -o wide**

**168 kubectl describe deployment mydeployment**

**169 vi deploy2.yaml**

**170 clear**

**171 kubectl apply -f deploy2.yaml**

**172 kubectl describe deployment.apps/mydeployment**

**173 vi deploy2.yaml**

**174 kubectl get pods -o wide**

**175 kubectl apply -f deploy2.yaml**

**176 kubectl get pods -o wide**

**177 cat deploy2.yaml**

**180 kubectl apply -f** [**https://raw.githubusercontent.com/lerndevops/educka/master/3-controllers/jobs/etcd-bakup.yaml**](https://raw.githubusercontent.com/lerndevops/educka/master/3-controllers/jobs/etcd-bakup.yaml)

**181 kubectl get deployment**

**182 kubectl get services**

**183 kubectl get job -o wide**

**184 kubectl get pods -o wide**

**185 kubectl logs backup-etcd-job-zxl4q**

**188 kubectl apply -f** [**https://raw.githubusercontent.com/lerndevops/educka/master/3-controllers/cronjobs/etcd-bakup-cronjob.yaml**](https://raw.githubusercontent.com/lerndevops/educka/master/3-controllers/cronjobs/etcd-bakup-cronjob.yaml)

**189 kubectl get cronjob -o wide**

**190 kubectl get jobs -o wide**

**191 kubectl get pods -o wide**

**192 watch kubectl get pod -o wide**

**193 vi cj.yaml**

**194 kubectl apply -f cj.yaml**

**195 kubectl get cronjob -o wide**

**devops@master-node:~$ cat cj.yaml**

**kind: CronJob**

**apiVersion: batch/v1**

**metadata:**

**name: backup-etcd-cornjob**

**namespace: default**

**spec:**

**schedule: "\* \* \* \* \*"**

**suspend: true**

**jobTemplate:**

**spec:**

**backoffLimit: 4**

**template:**

**metadata:**

**labels:**

**app: etcd1**

**spec:**

**restartPolicy: Never**

**volumes:**

**- name: hpvol**

**hostPath:**

**path: /opt/etcd-backup**

**containers:**

**- name: etcd**

**image: lerndevops/samples:etcdctl**

**command: ["sh", "-c", 'ETCDCTL\_API=3 etcdctl --endpoints=etcdserver:2379 snapshot save "etcd-snapshot-latest-`date +"%d-%m-%Y-%H-%M"`.db"']**

**volumeMounts:**

**- name: hpvol**

**mountPath: /opt/etcd-backup**

**202 kubectl run pod1 --image=nginx --namespace default -l env=test --dry-run=client -o yaml**

**203 kubectl run pod1 --image=nginx --namespace default -l env=test --dry-run=client -o json**

**204 kubectl run pod1 --image=nginx --namespace default -l env=test --dry-run=client -o json > pod1.json**

**205 kubectl run pod1 --image=nginx --namespace default -l env=test --dry-run=client -o yaml > pod1.yaml**

**206 kubectl create deployment test --image=nginx**

**207 kubectl get deployment -o wide**

**209 kubectl create deployment test --image=nginx --dry-run=client -o yaml**

**210 kubectl create deployment test --image=nginx --dry-run=client -o yaml > abc.yaml**

**211 vi abc.yaml**

**213 kubectl get pods**

**214 kubectl expose deployment test --name test-svc --type ClusterIP --port 80 --target-port 80**

**215 kubectl expose deployment test --name test-svc --type ClusterIP --port 80 --target-port 80 --dry-run=client -o yaml**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: test2**

**labels:**

**app: myapp**

**spec:**

**revisionHistoryLimit: 15**

**replicas: 3**

**strategy:**

**type: Recreate**

**selector:**

**matchLabels:**

**app: myapp**

**template: # what pod to be deployed**

**metadata:**

**labels:**

**app: myapp**

**spec:**

**terminationGracePeriodSeconds: 0**

**restartPolicy: Always**

**#initConatainers:**

**containers:**

**- name: myapp-cont**

**image: lerndevops/samplepyapp:v2**

**ports:**

**- containerPort: 3000**

**env:**

**- name: JAVA\_HOME**

**value: /opt/java**

**- name: DBHOST**

**value: "4.5.6.7"**

**resources:**

**requests: # the min amount of resources guranteed for the cont from vm/worker/node where it runs**

**memory: 128M**

**cpu: 20m # 1 core cpu = 1000 milli cpus**

**limits: # the max amount of resource that a cont can use from the node where it runs**

**memory: 512M**

**cpu: 50m**

**221 vi test2.yaml**

**222 kubectl apply -f test2.yaml**

**223 kubectl get pods -o wide**

**224 kubectl describe pod test2-c5878d9f8-g87l2**

**DAY5**

**Below are Scheduling Techniques which can be used to describe where the pods will be deployed (which worker nodes..)**

**Scheduler component on master nodes decides where to run the pods..**

**Scheduling parameters are written under the pod spec section of the yaml..**

**--- nodeName .. This will deploy the pods to listed nodeName only**

**--- nodeSelector**

**---- taints and tolerations**

**Below yaml is an example of nodeName scheduling.. In this case scheduler component is not involved.. schedule is hardcoded in yaml file.**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: nn-dep**

**labels:**

**app: myapp**

**spec:**

**revisionHistoryLimit: 15**

**replicas: 3**

**strategy:**

**type: Recreate**

**selector:**

**matchLabels:**

**app: myapp**

**template: # what pod to be deployed**

**metadata:**

**labels:**

**app: myapp**

**spec:**

**nodeName: worker-node02**

**terminationGracePeriodSeconds: 0**

**restartPolicy: Always**

**#initConatainers:**

**containers:**

**- name: myapp-cont**

**image: lerndevops/samplepyapp:v2**

**ports:**

**- containerPort: 3000**

**env:**

**- name: JAVA\_HOME**

**value: /opt/java**

**- name: DBHOST**

**value: "4.5.6.7"**

**#resources:**

**198 vi nn-dep.yaml**

**199 kubectl apply -f nn-dep.yaml**

**200 kubectl get pods -o wide**

**201 kubectl scale deployment nn-dep --replicas 10**

**202 kubectl get pods -o wide**

**203 kubectl scale deployment nn-dep --replicas 2**

**204 kubectl get pods -o wide**

**205 kubectl describe pod nn-dep-77d4f7bc4b-cnk2x**

**NodeSelector: scheduling technique.. This will deploy pods to a group of nodes with certain label.**

**We can label nodes and it accepts any key:value pair…. we can also assign multiple lables to nodes.**

**When using nodeselector on nodes with multiple labels, all conditions must be met.. eg. Incase below a node02 is labled for role=app and env=dev**

**212 kubectl get nodes -o wide**

**213 kubectl label node worker-node01 role=app**

**214 kubectl label node worker-node02 role=app**

**215 kubectl label node worker-node02 env=dev**

**217 kubectl label node worker-node01 env=qa**

**218 kubectl describe node worker-node01 | grep -iA 6 labels**

**219 kubectl describe node worker-node02 | grep -iA 6 labels**

**220 clear**

**221 vi ns-dep.yaml**

**222 kubectl apply -f ns-dep.yaml**

**223 kubectl get pods -o wide**

**224 kubectl label node worker-node01 env- # delete the label on the node**

**225 kubectl label node worker-node02 env- # delete the label on the node**

**226 kubectl describe node worker-node01 | grep -iA 6 labels**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: ns-dep**

**labels:**

**app: myapp**

**spec:**

**revisionHistoryLimit: 15**

**replicas: 3**

**#strategy:**

**selector:**

**matchLabels:**

**app: myapp**

**template: # what pod to be deployed**

**metadata:**

**labels:**

**app: myapp**

**spec:**

**#nodeName: worker-node02**

**nodeSelector:**

**role: app**

**terminationGracePeriodSeconds: 0**

**restartPolicy: Always**

**#initConatainers:**

**containers:**

**- name: myapp-cont**

**image: lerndevops/samplepyapp:v2**

**#ports:**

**#env:**

**#resources:**

**Taint-Toleration Scheduling technique:**

**Taint is like a lock, a condition that we apply to node.. when a taint is applied to a node then the node start rejecting the pod requests and its like locking a node from deploying any pod onto it...and the node will not accept requests for pods untill it meets the desired toleration or specific condition.**

**Taints are applied to the nodes in the nodes configuration while tolerations are defined in the pods spec of the yaml file..**

**Taint can be key=value:effect or key:effect**

**Effect is pre defined but key:value can be anything… eg region=us:NoExecute**

**Predefined effects are as below:**

**--- NoSchedule**

**--- NoExecute**

**230 kubectl delete all --all**

**231 clear**

**232 kubectl get nodes -o wide**

**233 kubectl describe node worker-node01 | grep -i taints**

**234 kubectl describe node worker-node02 | grep -i taints**

**235 kubectl describe node master-node | grep -i taints**

**236 kubectl taint node worker-node01 role=db:NoSchedule**

**237 kubectl taint node worker-node02 role=db:NoSchedule**

**238 kubectl create deployment test --image=httpd**

**239 kubectl get pods**

**240 kubectl describe pod test-5788f74bf8-wmn8h**

**241 kubectl describe node worker-node01 | grep -i taints**

**242 kubectl describe node worker-node02 | grep -i taints**

**243 kubectl describe node master-node | grep -i taints**

**244 vi tt-dep1.yaml**

**245 kubectl apply -f tt-dep1.yaml**

**246 kubectl get pods -o wide**

**247 kubectl taint node worker-node01 region=us:NoExecute**

**248 kubectl get pods -o wide**

**249 kubectl taint node worker-node02 region=us:NoExecute**

**250 kubectl get pods -o wide**

**251 vi tt-dep2.yaml**

**252 kubectl apply -f tt-dep2.yaml**

**253 kubectl get pods -o wide**

**254 vi tt-dep3.yaml**

**255 kubectl apply -f tt-dep3.yaml**

**256 kubectl get pods -o wide**

**257 clear**

**258 kubectl get nodes**

**259 kubectl describe node worker-node01 | grep -iA 1 taint s**

**260 kubectl describe node worker-node01 | grep -iA 1 taints**

**261 kubectl describe node worker-node02 | grep -iA 1 taints**

**262 kubectl taint node worker-node01 role=db:NoSchedule- # delete the taint**

**263 kubectl taint node worker-node02 role=db:NoSchedule- # delete the taint**

**264 kubectl taint node worker-node02 region=us:NoExecute- # delete the taint**

**265 kubectl taint node worker-node01 region=us:NoExecute- # delete the taint**

**266 kubectl get pods -o wide**

**# kubectl describe nodes | egrep - i “taints|Name” # this will list all the nodes and the taints applied to them**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: tt-dep1**

**labels:**

**app: myapp**

**spec:**

**revisionHistoryLimit: 15**

**replicas: 3**

**#strategy:**

**selector:**

**matchLabels:**

**app: myapp**

**template: # what pod to be deployed**

**metadata:**

**labels:**

**app: myapp**

**spec:**

**#nodeName: worker-node02**

**#nodeSelector:**

**tolerations:**

**- key: role # role=db:NoSchedule**

**value: db**

**effect: NoSchedule**

**operator: "Equal" # when a taint is defined as key=value:effect then the operator is always Equal**

**- key: node-role.kubernetes.io/master**

**effect: NoSchedule**

**operator: "Exists" # when the taint is defined as key:effect then the operator is always Exists**

**terminationGracePeriodSeconds: 0**

**restartPolicy: Always**

**#initConatainers:**

**containers:**

**- name: myapp-cont**

**image: lerndevops/samplepyapp:v2**

**#ports:**

**#env:**

**#resources:**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: tt-dep3**

**labels:**

**app: myapp**

**spec:**

**revisionHistoryLimit: 15**

**replicas: 3**

**#strategy:**

**selector:**

**matchLabels:**

**app: myapp**

**template: # what pod to be deployed**

**metadata:**

**labels:**

**app: myapp**

**spec:**

**#nodeName: worker-node02**

**#nodeSelector:**

**tolerations:**

**- key: role # role=db:NoSchedule**

**value: db**

**effect: NoSchedule**

**operator: "Equal" # when a taint is defined as key=value:effect then the operator is always Equal**

**- key: node-role.kubernetes.io/master**

**effect: NoSchedule**

**operator: "Exists" # when the taint is defined as key:effect then the operator is always Exists**

**- key: region**

**value: us**

**effect: NoExecute**

**operator: "Equal"**

**terminationGracePeriodSeconds: 0**

**restartPolicy: Always**

**#initConatainers:**

**containers:**

**- name: myapp-cont**

**image: lerndevops/samplepyapp:v2**

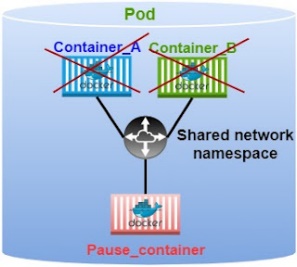
**#ports:**

**#env:**

**#resources:**

**Pause container is the secret container that runs on every pods in kubernetes. And it hold the network namespace for the pod.**

**Kubernetes creates pause containers to acquire the repective pods’s IP address and setup network namespace for all other containers that join that pod…**



**So even if container A and B crash, the network namespace is still there, held by pause container and it keeps it’s IP address whenever a new container within POD is created. containers within a pod share an IP address but run on different ports related to the different applications running inside them.**

**Docker level of commands for creating a pod..**

**174 docker run -d --name cont1 lerndevops/samples:netshoot**

**175 docker inspect cont1**

**176 docker network ls**

**177 docker run -d --name test --network host lerndevops/samples:netshoot**

**178 hostname -i**

**179 docker exec test hostname -i**

**180 docker run -d --name pause k8s.gcr.io/pause:3.6**

**181 docker ps**

**182 docker inspect pause**

**183 docker ps**

**184 docker run -d --name tcont --network container:pause tomcat:latest**

**185 docker run -d --name ncont --network container:pause nginx:latest**

**186 docker inspect pause | grep IPAddr**

**187 docker inspect tcont | grep IPAddr**

**188 docker exec -it tcont /bin/bash**

**189 docker exec ncont /bin/bash**

**190 hostname -i**

**191 docker exec -it ncont /bin/bash**

[**https://github.com/lerndevops/educka/blob/master/6-networking/core-dns/dns-test.md**](https://github.com/lerndevops/educka/blob/master/6-networking/core-dns/dns-test.md)

**Inside the dnstest pod**

**bash-5.1# history**

**1 curl 192.168.87.238:27018**

**2 curl 192.168.87.238:27017**

**3 curl 192.168.87.236:8080**

**4 curl 10.100.39.78:27017**

**5 curl mongo:27017**

**6 exit**

**7 curl**

**8 nslookup 10.100.39.78**

**9 nslookup 10.100.92.36**

**10 nslookup 192.168.87.236**

**11 curl 192-168-87-236.default.pod.cluster.local:8080**

**12 nslookup 192-168-87-236.default.pod.cluster.local**

NAME RESOLUTION CONFIGURATION IN KUBERNETES

**POD to POD communication is always open by default in kubernets via**

**To communicate to a database in pod in kubernets we can direct hit the IP address of the pod hosting database but incase there are multiple database servers/pods, we need a load balancer / cluster IP to communicate**

**Whenver a kubernetes cluster is setup, as as part of setup kube-dns services is created be default that helps in name resolution.**

**Here we use a static service name instead of IP address.**

**Direct communnication is not suggest because of dynamic nature.**

**289 kubectl delete all --all**

**290 clear**

**291 kubectl apply -f https://raw.githubusercontent.com/lerndevops/educka/master/examples/springboot-mongo-app.yml**

**292 kubectl get pod -o wide**

**293 kubectl get services**

**294 kubectl scale deployment mongodb --replicas 0**

**295 kubectl get pod -o wide**

**296 kubectl scale deployment mongodb --replicas 1**

**297 kubectl get pod -o wide**

**298 curl 192.168.87.238:27017**

**299 curl 10.100.39.78:27017**

**300 kubectl exec -it springboot-app-5546cb4889-qx4wd -- /bin/sh**

**301 hostname -i**

**302 ping 10.128.0.3**

**303 ping naresh**

**304 sudo vi /etc/hosts**

**305 cat /etc/hosts**

**306 ping naresh**

**307 ping www.myshop.com**

**308 cat /etc/hosts**

**309 ping worker-node01**

**310 cat /etc/resolv.conf**

**311 nslookup 10.128.0.3**

**312 nslookup worker-node01**

**313 kubectl exec -it springboot-app-5546cb4889-qx4wd -- /bin/sh**

**314 kubectl get services**

**315 kubectl exec -it springboot-app-5546cb4889-qx4wd -- /bin/sh**

**316 kubectl get pods -o wide**

**317 clear**

**318 kubectl run dnstest --image=lerndevops/samples:netshoot**

**319 kubectl get pod -o wide**

**320 kubectl get services**

**321 kubectl exec -it dnstest -- /bin/bash**

**322 history**

**323 kubectl get pod**

**324 kubectl apply -f https://raw.githubusercontent.com/lerndevops/educka/master/examples/springboot-mongo-app.yml -n teama**

**325 kubectl get services**

**326 kubectl get services -n teama**

**327 kubectl exec -it dnstest -- /bin/bash**

**DAY6**

**INGRESS**

**Ingress is rules for inbound traffic.. or accessng applicationn from outside world..exposing**

**Ingress controller is also a load balancer that can forward requst into internal multiple kube services(Cluster IP).**

**Ingress always runs inside the kube cluster only.**

**It is like a regular kube pod, running a load balancer process (nginx / haproxy / …)**

**Ingress resource:**

* **It is sa regular kube manifest yaml file**
* **In which we write routing rules & feed it to ingress controller.**

**As a part of ingress controller deployment an ingress class is always created.. which tells what type of ingress controller is used..**

**This is also specified/required as ingressClassName under ingress ules yaml file.**

**Apps.yaml file content**

**apiVersion: v1**

**kind: Service**

**metadata:**

**namespace: default**

**name: nginx-custom-default-backend**

**labels:**

**app.kubernetes.io/name: nginx-custom-default-backend**

**app.kubernetes.io/part-of: ingress-nginx**

**spec:**

**selector:**

**app.kubernetes.io/name: nginx-custom-default-backend**

**app.kubernetes.io/part-of: ingress-nginx**

**ports:**

**- port: 80**

**targetPort: 80**

**name: http**

**---**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**namespace: default**

**name: nginx-custom-default-backend**

**labels:**

**app.kubernetes.io/name: nginx-custom-default-backend**

**app.kubernetes.io/part-of: ingress-nginx**

**spec:**

**replicas: 1**

**selector:**

**matchLabels:**

**app.kubernetes.io/name: nginx-custom-default-backend**

**app.kubernetes.io/part-of: ingress-nginx**

**template:**

**metadata:**

**labels:**

**app.kubernetes.io/name: nginx-custom-default-backend**

**app.kubernetes.io/part-of: ingress-nginx**

**spec:**

**containers:**

**- name: nginx-custom-default-backend**

**image: lerndevops/samples:default-page**

**ports:**

**- containerPort: 80**

**# Setting the environment variable DEBUG we can see the headers sent**

**# by the ingress controller to the backend in the client response.**

**env:**

**- name: DEBUG**

**value: "false"**

**---**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: pyapp**

**spec:**

**replicas: 2**

**selector:**

**matchLabels:**

**app: pyapp**

**template:**

**metadata:**

**labels:**

**app: pyapp**

**spec:**

**containers:**

**- name: pyapp**

**image: lerndevops/samples:pyapp**

**ports:**

**- containerPort: 3000**

**---**

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: pyapp-svc**

**spec:**

**ports:**

**- port: 80**

**targetPort: 3000**

**selector:**

**app: pyapp**

**---**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: unapp**

**spec:**

**replicas: 2**

**selector:**

**matchLabels:**

**app: unapp**

**template:**

**metadata:**

**labels:**

**app: unapp**

**spec:**

**containers:**

**- name: unapp**

**image: lerndevops/samples:unapp**

**ports:**

**- containerPort: 80**

**---**

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: unapp-svc**

**spec:**

**ports:**

**- port: 80**

**targetPort: 80**

**selector:**

**app: unapp**

**---**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: petclinic**

**spec:**

**replicas: 2**

**selector:**

**matchLabels:**

**app: petclinic**

**template:**

**metadata:**

**labels:**

**app: petclinic**

**spec:**

**containers:**

**- name: petclinic**

**image: lerndevops/samples:petclinic**

**ports:**

**- containerPort: 8080**

**---**

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: petclinic-svc**

**spec:**

**ports:**

**- port: 80**

**targetPort: 8080**

**selector:**

**app: petclinic**

**app-ingress-rules.yaml**

**apiVersion: networking.k8s.io/v1**

**kind: Ingress**

**metadata:**

**name: apps-ingress-rule**

**annotations:**

**ingress.kubernetes.io/rewrite-target: /**

**spec:**

**ingressClassName: nginx**

**rules:**

**- http:**

**paths:**

**- path: /**

**pathType: Prefix**

**backend:**

**service:**

**name: nginx-custom-default-backend**

**port:**

**number: 80**

**- path: /unapp**

**pathType: Prefix**

**backend:**

**service:**

**name: unapp-svc**

**port:**

**number: 80**

**- path: /pyapp**

**pathType: Prefix**

**backend:**

**service:**

**name: pyapp-svc**

**port:**

**number: 80**

**- path: /petclinic**

**pathType: Prefix**

**backend:**

**service:**

**name: petclinic-svc**

**port:**

**number: 80**

**421 kubectl apply -f https://raw.githubusercontent.com/lerndevops/educka/master/6-networking/ingress/apps.yaml**

**422 kubectl get services -o wide**

**424 kubectl get deployments**

**425 kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.3.0/deploy/static/provider/baremetal/deploy.yaml**

**426 kubectl get pods -n ingress-nginx**

**427 kubectl get services -n ingress-nginx**

**428 kubectl get ingressclass**

**429 clear**

**430 kubectl get services**

**431 kubectl describe svc nginx-custom-default-backend**

**432 curl 192.168.158.52:80**

**433 curl 10.98.12.211:80**

**434 clear**

**435 kubectl apply -f https://raw.githubusercontent.com/lerndevops/educka/master/6-networking/ingress/app-ingress-rules.yaml**

**436 kubectl get ingress**

**437 kubectl describe ingress apps-ingress-rule**

**451 kubectl run hostnetpod --image=nginx --dry-run=client -o yaml > hostnetpod.yaml**

**452 vi hostnetpod.yaml**

**453 cat hostnetpod.yaml**

**454 kubectl apply -f hostnetpod.yaml**

**455 kubectl get pods -o wide**

**456 history**

**457 kubectl get pods -n kube-system -o wide**

**461 kubectl run dnstest --image=lerndevops/samples:netshoot**

**462 kubectl exec -it dnstest -- /bin/bash**

**463 kubectl exec -it pyapp-f7c59bc88-4m59n -- /bin/bash**

**464 clear**

**465 kubectl delete deployment --all**

**466 kubectl apply -f https://raw.githubusercontent.com/lerndevops/educka/master/examples/springboot-mongo-app.yml**

**467 kubectl get pods**

**468 kubectl get pods --show-labels**

**469 vi netpol.yaml**

**470 kubectl get networkpolicies**

**471 kubectl get netpol**

**472 kubectl apply -f netpol.yaml**

**475 kubectl get netpol**

**476 kubectl describe netpol allow-access-mongo-from-spring-app-only**

**477 kubectl get pod -o wide**

**478 kubectl exec -it springboot-app-5546cb4889-5gxjq -- /bin/bash**

**479 kubectl exec -it springboot-app-5546cb4889-5gxjq -- /bin/sh**

**480 kubectl exec -it dnstest -- /bin/bash**

**kind: NetworkPolicy**

**apiVersion: networking.k8s.io/v1**

**metadata:**

**name: allow-access-mongo-from-spring-app-only**

**namespace: default**

**spec:**

**podSelector: # on which pods the rule / policy should be applied**

**matchLabels:**

**app: mongodb**

**policyTypes:**

**- Ingress # all the incoming traffic to pod with label above**

**#- Egress # all the outgoing traffic from the pod with label above**

**ingress:**

**- from: # from which pod in the cluster should mongo dbpod accept the traffic & block everything else**

**- podSelector:**

**matchLabels:**

**app: myapp**

**#- namespaceSelector:**

**#- ipBlock:**

**# cidr: 0.0.0.0/0**

**482 kubectl get pods**

**483 kubectl get deployment -o wide**

**484 kubectl scale deployment springboot-app --replicas 8**

**485 kubectl delete pod springboot-app-5546cb4889-kfw77**

**486 kubectl scale deployment springboot-app --replicas 2**

**487 kubectl apply -f https://raw.githubusercontent.com/lerndevops/educka/master/3-controllers/statefulsets/statefulset-ex1.yml**

**488 kubectl get sts**

**489 kubectl scale sts db --replicas 12**

**490 kubectl scale sts db --replicas 6**

**491 kubectl delete pod db-3**

**492 kubectl get services**

**493 kubectl delete svc nginx-custom-default-backend**

**494 clear**

**495 kubectl get services**

**496 kubectl get pod**

**497 kubectl exec -it dnstest -- /bin/bash**

**Inside dnstest pod**

**21 nslookup mongo**

**22 nslookup redis**

**23 nslookup 192.168.87.245**

**24 nslookup 192.168.158.62**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: lp-dep**

**labels:**

**app: myapp**

**spec:**

**revisionHistoryLimit: 15**

**replicas: 3**

**#strategy:**

**selector:**

**matchLabels:**

**app: myapp**

**template: # what pod to be deployed**

**metadata:**

**labels:**

**app: myapp**

**spec:**

**#nodeName: worker-node02**

**#nodeSelector:**

**#tolerations:**

**terminationGracePeriodSeconds: 0**

**restartPolicy: Always**

**#initConatainers:**

**containers:**

**- name: myapp-cont**

**image: lerndevops/samples:springboot-app**

**#ports:**

**#env:**

**#resources:**

**#probes -- helps to run a helath check inside the cont & then take some action if failed**

**livenessProbe: # what happens if the live ness probe fails. Ans: the container will be restarted immediatley**

**exec:**

**command: ["curl", "localhost:8080"]**

**initialDelaySeconds: 60**

**periodSeconds: 10**

**timeoutSeconds: 5**

**507 vi lp-dep.yaml**

**508 kubectl delete deployment lp-dep**

**509 kubectl apply -f lp-dep.yaml**

**510 cat lp-dep.yaml**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: lp-dep**

**labels:**

**app: myapp**

**spec:**

**revisionHistoryLimit: 15**

**replicas: 3**

**#strategy:**

**selector:**

**matchLabels:**

**app: myapp**

**template: # what pod to be deployed**

**metadata:**

**labels:**

**app: myapp**

**spec:**

**#nodeName: worker-node02**

**#nodeSelector:**

**#tolerations:**

**terminationGracePeriodSeconds: 0**

**restartPolicy: Always**

**#initConatainers:**

**containers:**

**- name: myapp-cont**

**image: lerndevops/samples:springboot-app**

**#ports:**

**#env:**

**#resources:**

**#probes -- helps to run a helath check inside the cont & then take some action if failed**

**livenessProbe: # what happens if the live ness probe fails. Ans: the container will be restarted immediatley**

**exec:**

**command: ["curl", "localhost:8080"]**

**initialDelaySeconds: 60**

**periodSeconds: 10**

**timeoutSeconds: 5**

**readinessProbe:**

**exec:**

**command: ["curl", "localhost:880"]**

**initialDelaySeconds: 60**

**periodSeconds: 10**

**timeoutSeconds: 5**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: lp-dep**

**labels:**

**app: myapp**

**spec:**

**revisionHistoryLimit: 15**

**replicas: 3**

**#strategy:**

**selector:**

**matchLabels:**

**app: myapp**

**template: # what pod to be deployed**

**metadata:**

**labels:**

**app: myapp**

**spec:**

**#nodeName: worker-node02**

**#nodeSelector:**

**#tolerations:**

**terminationGracePeriodSeconds: 0**

**restartPolicy: Always**

**#initConatainers:**

**containers:**

**- name: myapp-cont**

**image: lerndevops/samples:springboot-app**

**#ports:**

**#env:**

**#resources:**

**#probes -- helps to run a helath check inside the cont & then take some action if failed**

**livenessProbe: # what happens if the live ness probe fails. Ans: the container will be restarted immediatley**

**exec:**

**command: ["curl", "localhost:8080"]**

**initialDelaySeconds: 60**

**periodSeconds: 10**

**timeoutSeconds: 5**

**readinessProbe:**

**# httpGet:**

**# path: /**

**# port: 8080**

**#tcpSocket:**

**#port: 8080**

**exec:**

**command: ["curl", "localhost:880"]**

**initialDelaySeconds: 60**

**periodSeconds: 10**

**timeoutSeconds: 5**

**DAY7**