sudo  -s

  15  cd

   16  mkdir docker

   17  cd docker/

   18  vi app.py

from flask import Flask

import os

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

    return ('\nHello from Container World! \n\n')

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(host="0.0.0.0", port=8080, debug=True)

   19  vi Dockerfile

FROM ubuntu:20.04

RUN apt update && apt install python3 -y && apt install python3-flask -y

COPY app.py /tmp

EXPOSE 8080

CMD ["python3", "/tmp/app.py"]

   20  docker build -t first:1.0 .

   21  docker images

   22  docker ps -a

   23  docker run -d --name myfirst -p 8000:8080 first:1.0

   24  docker ps

   25  curl localhost:8000

   26  docker logs myfirst

Kubernetes cluster setup link

<https://github.com/rskTech/k8s_material/blob/master/01_Creating_and_Configuring_a_Kubernetes_Cluster.pdf>

   37  kubectl get nodes

   38  kubectl get po -o wide

   39  kubectl get po -o wide -n kube-system

   40  alias k=kubectl

   41  k get nodes

   42  k api-resources

   43  k get ns

   44  k get po

   45  k get po -n kube-system

   46  k get po -n kube-system -o wide

   47  k run first --image=rajendrait99/first:1.0 --port=8080

 k run first --image=rajendrait99/first:1.0 --port=8080 –dry-run=client -o yaml > pod.yaml

   48  k get po

   49  k get po -o wide

   50  k describe po first

   51  k logs first

   52  k exec -it first bash

Deployment:

   69  k get po -o wide

   70  k get po

   71  k delete po first

   72  k get po

   73  k delete po second

   74  k create deploy mydeploy --image=rajendrait99/first:1.0 --port=8080

   75  k get deploy

   76  k get rs

   77  k get po

   78  k get all

   79  k delete po mydeploy-55dc84985c-hr84r

   80  k get all

   81  k delete deploy mydeploy

   82  k get all

   83  k create deploy mydeploy --image=rajendrait99/first:1.0 --port=8080 --dry-run=client -o yaml > deploy.yaml

   84  vi deploy.yaml

   85  k apply -f deploy.yaml

   86  k get all

Manual scaling

   88  k get all

   89  k scale deploy mydeploy --replicas=10   (option1 to scale using imperative command)

   90  k get all

   91  k scale deploy mydeploy --replicas=2

   92  k get all

   93  vi deploy.yaml     (option2 Modify yaml and then apply)

   94  k apply -f deploy.yaml

   95  k get all

   96  k get all -o wide

   97  k get all

   98  k edit deploy mydeploy  (option3 modify live object)

   99  k get all

Horizontal POD autoscaler

  103  k get po -o wide

  104  k get all

  105  k scale deploy mydeploy --replicas=2

  106  k get all

  107  k autoscale deploy mydeploy --cpu-percent=80 --min=2 --max=10

  108  k get hpa

Day3:

Zero down time upgrade:

k delete hpa mydeploy

  122  k delete deploy mydeploy

  123  k create deploy mydeploy --image=nginx:latest --port=80

  124  k get all

  125  k scale deploy mydeploy --replicas=20

  126  k get all

  127  k describe po mydeploy-5b477bf648-z75xc

  128  k rollout history deploy mydeploy

  129  k rollout status deploy mydeploy

  130  k get all

  131  k set image deploy mydeploy nginx=nginx:1.8.1 --record

  132  k rollout history deploy mydeploy

  133  k get all

  134  k rollout status deploy mydeploy

  135  k get all

  136  k describe po mydeploy-6b7b846487-xnb9x

  137  k rollout history deploy mydeploy

  138  k set image deploy mydeploy nginx=nginx:1.9.1 --record

  139  k rollout history deploy mydeploy

  140  k get all

  141  k rollout status deploy mydeploy

  142  k get all

  143  k describe po mydeploy-5f69579954-zpbs6

  144  k rollout history deploy mydeploy

  145  k get all

  147  k rollout undo deploy mydeploy --to-revision=2

  148  k rollout history deploy mydeploy

Service:

  154  git clone https://github.com/rskTech/serviceDemo.git

  155  cd serviceDemo/

  156  ls

  157  cd build/

  158  ls

  159  vi app.py

  160  cd ..

  161  cd deploy/

  162  k get all

  163  alias k=kubectl

  164  k get all

  165  k delete deploy mydeploy

  166  k get all

  167  vi db-pod.yml

  168  k apply -f db-pod.yml

  169  k get po

  170  vi db-svc.yml

  171  vi db-pod.yml

  172  vi db-svc.yml

  173  k apply -f db-svc.yml

  174  k get all

  175  vi web-pod.yaml

  176  k apply -f web-pod.yaml

  177  k get all

  178  vi web-svc.yml

  179  k apply -f web-svc.yml

  180  k get all

  181  k get no -o wide

  182  curl 172.31.41.217:31605/init

  183  curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "1", "user":"John Doe"}' http://172.31.41.217:31605/users/add

  184  curl -i -H "Content-Type: application/json" -X POST -d '{"uid": "2", "user":"Bob"}' http://172.31.41.217:31605/users/add

  185  curl 172.31.41.217:31605/users/1

  186  curl 172.31.41.217:31605/users/2

Creating service using imperative command

  188  k get all

  189  k get no -o wide

  190  curl 172.31.41.217:31605/users/2

  191  vi ../build/app.py

  192  k get all

  193  k delete po  mysql web1

  194  k delete svc mysql web

  195  k get all

  196  k run nginx --image=nginx --port=80

  197  k get po

  198  k expose po nginx --name mysvc --type=NodePort --port=80 --target-port=80

  199  k get all

  200  curl 172.31.41.217:31558

mkdir -p $HOME/.kube

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

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

Day4:

Daemonset

  211  k get po -n kube-system -o wide

  212  k get ds -n kube-system

  213  vi ds.yaml

apiVersion: apps/v1

kind: DaemonSet

metadata:

  name: fluentd-elasticsearch

  labels:

    k8s-app: fluentd-logging

spec:

  selector:

    matchLabels:

      name: fluentd-elasticsearch

  template:

    metadata:

      labels:

        name: fluentd-elasticsearch

    spec:

      containers:

      - name: fluentd-elasticsearch

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

  214  k get ds

  215  k apply -f ds.yaml

  216  k get ds

  217  k delete po nginx

  218  k get ds

  219  k get po

  220  k delete po fluentd-elasticsearch-tx222

  221  k get po -o wide

  222  k delete po fluentd-elasticsearch-lgnxv

  223  k get po -o wide

Scheduling

NodeSelector

  231  k apply -f pod.yaml

  232  vi /etc/kubernetes/admin.conf

  233  vi ~/.kube/config

  234  k get all

  235  k delete ds fluentd-elasticsearch

  236  k get all

  237  k delete svc mysvc

  238  k get all

  239  k get no

  240  k get no --show-labels

  241  k label no worker1 hdd=ssd

  242  k get no --show-labels

  243  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: first

  name: first

spec:

  nodeSelector:

    hdd: ssd

  containers:

  - image: rajendrait99/first:1.0

    name: first

    ports:

    - containerPort: 8080

    resources: {}

  dnsPolicy: ClusterFirst

  restartPolicy: Always

status: {}

  244  k apply -f pod.yaml

  245  k get po -o wide

  246  k delete po first

  247  k get no --show-labels

  248  k label no worker1 hdd-

  249  k get no --show-labels

  250  vi pod.yaml

  251  k apply -f pod.yaml

  252  k get po

  253  k describe po  first

**Node Affinity:**

  260  k explain po

  261  k explain po.spec

  262  k explain po.spec.containers

  263  k explain po.spec.affinity

  264  k explain po.spec.affinity.nodeAffinity

  265  k explain po.spec.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution

  266  k explain po.spec.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms

  267  k explain po.spec.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms.matchExpressions

  268  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: first

  name: first

spec:

  affinity:

    nodeAffinity:

      requiredDuringSchedulingIgnoredDuringExecution:

        nodeSelectorTerms:

          - matchExpressions:

             - key: hdd

               operator: In

               values:

                 - ssd

  containers:

  - image: rajendrait99/first:1.0

    name: first

    ports:

    - containerPort: 8080

    resources: {}

  dnsPolicy: ClusterFirst

  restartPolicy: Always

status: {}

  269  k get no --show-labels

  270  k label no worker1 hdd=ssd

  271  k get no --show-labels

  272  k get po

  273  k delete po first

  274  k apply -f pod.yaml

  275  vi pod.yaml

  276  k apply -f pod.yaml

  277  k explain po.spec.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms.matchExpressions

  278  vi pod.yaml

  279  k apply -f pod.yaml

  280  k get po

  281  k get po -o wide

  283  vi pod.yaml

**PodAffinity**

  286  k explain po.spec.affinity.podAffinity

  287  k explain po.spec.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution

  288  k explain po.spec.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution.labelSelector

  289  k explain po.spec.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution.labelSelector.matchExpressions

  290  k gte po

  291  k get po

  292  k delete po first

  293  vi pod.yaml

  294  k get no --show-lables

  295  k get no --show-labels

  296  vi pod.yaml

  297  k run nginx --image=nginx

  298  k get po

  299  k get po  --show-labels

  300  k label po nginx hdd=ssd

  301  k get po  --show-labels

  302  k get po -o wide

  303  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: first

  name: first

spec:

  affinity:

    podAffinity:

      requiredDuringSchedulingIgnoredDuringExecution:

        - labelSelector:

            matchExpressions:

             - key: hdd

               operator: In

               values:

                 - ssd

          topologyKey: kubernetes.io/hostname

  containers:

  - image: rajendrait99/first:1.0

    name: first

    ports:

    - containerPort: 8080

    resources: {}

  dnsPolicy: ClusterFirst

  restartPolicy: Always

   306  k apply -f pod.yaml

  313  k get po

  314  k get po -o wid

Pod AntiAffinity

  317  k get all

  318  k delete po first

  319  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: first

  name: first

spec:

  affinity:

    podAntiAffinity:

      requiredDuringSchedulingIgnoredDuringExecution:

        - labelSelector:

            matchExpressions:

             - key: hdd

               operator: In

               values:

                 - ssd

          topologyKey: kubernetes.io/hostname

  containers:

  - image: rajendrait99/first:1.0

    name: first

    ports:

    - containerPort: 8080

    resources: {}

  dnsPolicy: ClusterFirst

  restartPolicy: Always

status: {}

  320  k get po -o wide

  321  k apply -f pod.yaml

  322  k get po -o wide

……………………………………………………….

**Day5:**

**Taint and Tolerations**

   30  alias k=kubectl

   31  k get no

   32  k describe no master

   33  k describe no worker1

   34  k describe no worker2

   35  k taint no worker1 hdd=ssd:NoSchedule

   36  k describe no worker1

   37  k get no

   38  k get po

   39  k run nginx --image=nginx --port=80 --dry-run=client -o yaml > pod.yaml

   40  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: nginx

  name: nginx

spec:

  tolerations:

    - key: hdd

      operator: Equal

      value: ssd

      effect: NoSchedule

  containers:

  - image: nginx

    name: nginx

    ports:

    - containerPort: 80

    resources: {}

  dnsPolicy: ClusterFirst

  restartPolicy: Always

status: {}

   41  k apply -f pod.yaml

   42  k get po -o wide

   43  k delete po nginx

   44  k get no

   45  k cordon worker2

   46  k get no

   47  k apply -f pod.yaml

   48  k get po

   49  k describe po nginx

   50  k delete po nginx

   51  vi pod.yaml

   52  k apply -f pod.yaml

   53  k get po -o wide

   56  k get no

   57  k uncordon worker2

   58  k get no

   59  k taint no worker1 hdd-

   60  vi ds.yaml

apiVersion: apps/v1

kind: DaemonSet

metadata:

  name: fluentd-elasticsearch

  labels:

    k8s-app: fluentd-logging

spec:

  selector:

    matchLabels:

      name: fluentd-elasticsearch

  template:

    metadata:

      labels:

        name: fluentd-elasticsearch

    spec:

      tolerations:

      # these tolerations are to have the daemonset runnable on control plane nodes

      # remove them if your control plane nodes should not run pods

      - key: node-role.kubernetes.io/control-plane

        operator: Exists

        effect: NoSchedule

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

        operator: Exists

        effect: NoSchedule

      containers:

      - name: fluentd-elasticsearch

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

   61  k apply -f ds.yaml

   62  k get ds

   63  k get po -o wide

**Job:**

   67  k get job

   68  k create job myjob --image=ubuntu:20.04 --dry-run=client -o yaml -- /bin/sh -c "sleep 10" > job.yaml

   69  vi job.yaml

   70  k get all

   71  k delete ds fluentd-elasticsearch

   72  k delete po nginx

   73  k get all

   74  k apply -f job.yaml

   75  watch kubectl get all

   76  vi job.yaml

   77  k delete job myjob

   78  k apply -f job.yaml

   79  watch kubectl get all

   80  k delete job myjob

   81  vi job.yaml

apiVersion: batch/v1

kind: Job

metadata:

  creationTimestamp: null

  name: myjob

spec:

  completions: 10

  parallelism: 3

  template:

    metadata:

      creationTimestamp: null

    spec:

      containers:

      - command:

        - /bin/sh

        - -c

        - sleep 10

        image: ubuntu:20.04

        name: myjob

        resources: {}

      restartPolicy: Never

status: {}

   82  k apply -f job.yaml

   83  watch kubectl get all

**Cronjob  : to create a schedule website: crontab.guru**

  86  k get all

   87  k delete job myjob

   88  k create cj mycj --image=ubuntu:20.04 --schedule="\*/1 \* \* \* \*" --dry-run=client -o yaml -- /bin/sh -c "sleep 10" > cronjob.yaml

   89  vi cronjob.yaml

   90  k get cj

   91  k apply -f cronjob.yaml

   92  watch kubectl get all

**apiVersion: v1**

**kind: Pod**

**metadata:**

**creationTimestamp: null**

**labels:**

**run: pod1**

**name: pod1**

**spec:**

**nodeSelector:**

**kubernetes.io/hostname: master**

**tolerations:**

**- key: node-role.kubernetes.io/control-plane**

**effect: NoSchedule**

**containers:**

**- image: httpd:2.4.41-alpine**

**name: pod1-container**

**resources: {}**

**dnsPolicy: ClusterFirst**

**restartPolicy: Always**

**status: {}**

**Day6:**

**ConfigMap**

  32  alias k=kubectl

   33  k get no

   34  k get cm

   35  k create cm mycm --from-literal=db\_port=8080 --from-literal=db\_host=192.168.1.5

   36  k get cm

   37  k describe cm mycm

   38  k run nginx --image=nginx --port=80 --dry-run=client -o yaml > pod.yaml

   43  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: nginx

  name: nginx

spec:

  containers:

  - image: nginx

    name: nginx

    ports:

    - containerPort: 80

    env:

      - name: DBPORT

        valueFrom:

          configMapKeyRef:

            name: mycm

            key: db\_port

      - name: DBHOST

        valueFrom:

          configMapKeyRef:

            name: mycm

            key: db\_host

    resources: {}

  dnsPolicy: ClusterFirst

  restartPolicy: Always

status: {}

   44  k apply -f pod.yaml

   45  k get po

   46  k exec -it nginx bash

k exec -it nginx -- env

54  vi myconfig.ini

logfile=myapp.log

dbport=8080

dbhost=localhost

dbuser=admin

   55  k create cm mycm1 --from-file=myconfig.ini

   56  k describe cm mycm1

   57  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: nginx

  name: nginx

spec:

  containers:

  - image: nginx

    name: nginx

    ports:

    - containerPort: 80

    env:

      - name: DBPORT

        valueFrom:

          configMapKeyRef:

            name: mycm

            key: db\_port

      - name: DBHOST

        valueFrom:

          configMapKeyRef:

            name: mycm

            key: db\_host

    resources: {}

    volumeMounts:

      - name: myvol

        mountPath: /etc/lala

  dnsPolicy: ClusterFirst

  restartPolicy: Always

  volumes:

    - name: myvol

      configMap:

        name: mycm1

status: {}

   58  k delete po nginx

   59  k appaly -f pod.yaml

   60  k apply -f pod.yaml

   61  k get po

   62  k exec -it bash

   63  k exec -it nginx bash

   64  cat myconfig.ini

   65  vi pod.yaml

**Secret:**

1. **Generic secret**

   80  alias k=kubectl

   81  k get secret

   82  k create secret generic db-secret --from-literal=dbpass=admin

   83  k get secret

   84  k describe secret db-secret

   85  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: nginx

  name: nginx

spec:

  containers:

  - image: nginx

    name: nginx

    ports:

    - containerPort: 80

    env:

      - name: DBPASS

        valueFrom:

          secretKeyRef:

            name: db-secret

            key: dbpass

    resources: {}

  dnsPolicy: ClusterFirst

  restartPolicy: Always

status: {}

   86  k delete po nginx

   87  k apply -f pod.yaml

   88  k get po

   89  k exec -it nginx bash

**Docker registry secret**

   92  kubectl create secret docker-registry docker-secret --docker-email=example@gmail.com --docker-username=dev --docker-password=pass1234 --docker-server=my-registry.example:5000

   93  k get secret

   94  vi pod1.yaml

apiVersion: v1

kind: Pod

metadata:

  name: secret-demo-2

spec:

  containers:

  - name: demo-container

    image: nginx

    envFrom:

    - secretRef:

       name: docker-secret

   95  k delete po nginx

   96  k apply -f pod1.yaml

   97  k get po

   98  k exec -it secret-demo-2 bash

**TLS secrets**

 101  k create secret tls my-tls-secret --cert=/etc/kubernetes/pki/ca.crt --key=/etc/kubernetes/pki/ca.key

  102  k get secret

  103  vi pod2.yaml

apiVersion: v1

kind: Pod

metadata:

  name: secret-demo-3

spec:

  containers:

  - name: demo-container

    image: nginx

    volumeMounts:

      - name: data

        mountPath: /etc/cert-data

  volumes:

  - name: data

    secret:

      secretName: my-tls-secret

  104  k apply -f pod2.yaml

  105  k get po

  106  k exec -it secret-demo-3 bash

**Day7:**

**PersistentVolume & PersistentVolumeClaim**

109  cd

  110  alias k=kubectl

  111  k get no

  112  vi pv.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

  name: mypv

spec:

  accessModes:

    - ReadWriteMany

  storageClassName: normal

  capacity:

    storage: 2G

  hostPath:

    path: /opt

  113  k explain pv.spec

  114  k explain pv.spec.nfs

  115  k explain pv.spec.vsphereVolume

  116  k explain pv.spec.awsElasticBlockStore

  117  vi pv.yaml

  118  k get pv

  119  k apply -f pv.yaml

  120  k get pv

  121  vi pvc.yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  name: mypvc

spec:

  accessModes:

    - ReadWriteMany

  storageClassName: normal

  resources:

    requests:

      storage: 2G

  122  k get pv

  123  k apply -f pvc.yaml

  124  k get pv

  125  k get pvc

  126  vi pvc.yaml

  127  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: nginx

  name: nginx

spec:

  containers:

  - image: nginx

    name: nginx

    ports:

    - containerPort: 80

    resources: {}

    volumeMounts:

      - name: myvol

        mountPath: /etc/lala

  dnsPolicy: ClusterFirst

  restartPolicy: Always

  volumes:

    - name: myvol

      persistentVolumeClaim:

        claimName: mypvc

status: {}

  128  k get po

  129  k delete po secret-demo-2 secret-demo-3

  130  k apply -f pod.yaml

  131  k get po

  132  k exec -it nginx bash

k exec -it nginx -- ls /etc/lala

**Statefulset:**

140  git clone https://github.com/rskTech/k8s\_material.git

  141  cd k8s\_material/statefulset/

  142  vi sfs.yaml

  143  vi sc.yaml

  144  k get all

  145  k get pv

  146  k delete pv mypv

  147  k get po

  148  k delete po nginx

  149  k delete pvc mypvc

  150  k delete pv mypv

  151  k get all

  152  k apply -f sc.yaml

  153  k get sc

  154  k get sts

  155  k apply -f sfs.yaml

  156  k get sts

  157  k get po

  158  k get pvc

  159  vi pv.yaml

  160  k get pvc

  161  k get po

  162  k apply -f pv.yaml

  163  k get po

  164  k get pv

  165  k get pvc

  166  k get po

  167  k get pvc

  168  k get pv

  169  vi pv1.yaml

  170  k apply -f pv1.yaml

  171  k get pv

  172  k get pvc

  173  k get po

  174  vi pv2.yaml

  175  k apply -f pv2.yaml

  176  k get pvc

  177  k get po

  178  k get sts

  179  k get po

  180  k delete po web-0

  181  k get po

  182  k scale sts web --replicas=5

  183  k get po

  184  k scale sts web --replicas=2

  185  k get po

  186  k scale sts web --replicas=3

  187  k get po

  188  k get pv

  189  k get pvc

  190  vi sfs.yaml

**Assignment Q6:**

 202  k delete pv mypv mypv1 mypv2

  203  k delete pvc www-web-0 www-web-1 www-web-2

  204  k get all

  205  cd

  206  vi pv.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

  name: safari-pv

spec:

  accessModes:

    - ReadWriteOnce

  capacity:

    storage: 2G

  hostPath:

    path: /volumes/Data

  207  vi pvc.yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  namespace: project-tiger

  name: safari-pvc

spec:

  accessModes:

    - ReadWriteOnce

  resources:

    requests:

      storage: 2G

  208  k create ns project-tiger

  209  k create deploy safari --image=httpd:2.4.41-alpine -n project-tiger --dry-run=client -o yaml > deploy1.yaml

  210  vi deploy1.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

  creationTimestamp: null

  labels:

    app: safari

  name: safari

  namespace: project-tiger

spec:

  replicas: 1

  selector:

    matchLabels:

      app: safari

  strategy: {}

  template:

    metadata:

      creationTimestamp: null

      labels:

        app: safari

    spec:

      containers:

      - image: httpd:2.4.41-alpine

        name: httpd

        resources: {}

        volumeMounts:

          - name: myvol

            mountPath: /tmp/safari-data

      volumes:

        - name: myvol

          persistentVolumeClaim:

            claimName: safari-pvc

status: {}

  211  k apply -f pv.yaml

  212  vi pv.yaml

  213  k apply -f pv.yaml

  214  k get pv

  215  k apply -f pvc.yaml

  216  k get pv

  217  vi deploy1.yaml

  218  k apply -f deploy1.yaml

  219  k get all -n project-tiger

  220  k get pvc -n project-tiger

  221  vi deploy1.yaml

  222  cat pv.yaml

  223  cat pvc.yaml

  224  cat deploy1.yaml

**Day8:**

**Assigning resources**

   24  alias k=kubectl

   25  k get no

   26  k get po

   29  k run nginx --image=nginx --port=80  --dry-run=client -o yaml > pod.yaml

   30  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: nginx

  name: nginx

spec:

  containers:

  - image: nginx

    name: nginx

    ports:

    - containerPort: 80

    resources:

      requests:

        cpu: 0.1m

        memory: 200M

      limits:

        memory: 400M

        cpu: 0.2m

  dnsPolicy: ClusterFirst

  restartPolicy: Always

status: {}

   31  k apply -f pod.yaml

   32  k get po

   33  k describe po nginx

   34  k get no

   35  k describe no worker1

**ResourceQuota**

   37  cat pod.yaml

   38  k gte po

   39  k get po

   40  k describe po nginx

   41  k get po -o wide

   42  k get po

   43  k delete po nginx

   44  vi rq.yaml

apiVersion: v1

kind: ResourceQuota

metadata:

  name: myrq

  namespace: myns

spec:

  hard:

    cpu: "0.3m"

    memory: 500M

    pods: 10

   45  k get ns

   46  k create ns myns

   47  k get ns

   48  k apply -f rq.yaml

   49  k get quota

   50  k get quota -n myns

   51  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: nginx

  name: nginx1

  namespace: myns

spec:

  containers:

  - image: nginx

    name: nginx

    ports:

    - containerPort: 80

    resources:

      requests:

        cpu: 0.1m

        memory: 200M

      limits:

        memory: 400M

        cpu: 0.2m

  dnsPolicy: ClusterFirst

  restartPolicy: Always

status: {}

   52  k apply -f pod.yaml

   53  k get po

   54  k get po -n myns

   55  k get quota -n myns

   56  vi pod.yaml

   57  k apply -f pod.yaml

**PriorityClass**

   70  k api-resources | grep class

   71  vi pc.yaml

apiVersion: scheduling.k8s.io/v1

kind: PriorityClass

metadata:

  name: high

value: 1000

   72  k apply -f pc.yaml

   73  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: nginx

  name: nginx

spec:

  priorityClassName: high

  containers:

  - image: nginx

    name: nginx

    ports:

    - containerPort: 80

    resources:

      requests:

        cpu: 0.1m

        memory: 400M

      limits:

        memory: 400M

        cpu: 0.2m

  dnsPolicy: ClusterFirst

  restartPolicy: Always

status: {}

   74  k apply -f pod.yaml

   77  k get po

Network Policy

   82  k api-resources

   83  k get all

   84  k delete po nginx

   85  k run db --image=nginx

   88  k delete po frontend

   89  k run frontend --image=nginx

   90  k run other --image=nginx

   91  k get po

   92  k get po -o wide

   93  k exec -it frontend bash

   94  k exec -it other bash

   95  k get po -o wide

   96  vi np.yaml

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

  name: test-network-policy

  namespace: default

spec:

  podSelector:

    matchLabels:

      role: db

  policyTypes:

  - Ingress

  ingress:

  - from:

    - podSelector:

        matchLabels:

          role: frontend

   97  k get po --show-labels

   98  k label po db role=db

   99  k label po frontend role=frontend

  100  vi np.yaml

  101  k get po --show-labels

  102  k get networkpolicy

  103  k apply -f np.yaml

  104  k get po -o wide

  105  k exec -it frontend bash

  106  k exec -it other bash

Day9:

Kubectl config

  112  vi .kube/config

  113  curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.12.0/kind-linux-amd64

  114  chmod +x ./kind

  115  sudo mv ./kind /usr/local/bin/kind

  116  kind

  117  vi config

kind: Cluster

apiVersion: kind.x-k8s.io/v1alpha4

nodes:

- role: control-plane

- role: worker

- role: worker

  118  kin create cluster --config config --name mycluster

  119  kind create cluster --config config --name mycluster

  120  vi .kube/config

  121  docker ps

  122  vi .kube/config

  123  k get no

  124  alias k=kubectl

  125  k get no

  126  k get po

  127  k config -h

  128  k config get-clusters

  129  k config get-users

  131  k config get-contexts

  132  k config use-context kubernetes-admin@kubernetes

  133  k get no

  134  k get po

RBAC:

140  mkdir test

  141  cd test/

  142  openssl genrsa -out john.key 2048

  143  ls

  144  openssl req -new -key john.key -out john.csr -subj "/CN=john/O=examplegroup"

  145  ls

Switch to root user

cp /etc/kubernetes/pki/ca.crt /home/labsuser

Switch to labuser

sudo chmod 777 ca.crt ca.key

  146  openssl x509 -req -in john.csr -CA /etc/kubernetes/pki/ca.crt -CAkey /etc/kubernetes/pki/ca.key  -CAcreateserial -out john.crt

  151  kind delete cluster --name mycluster

  152  k get no

  153  pwd

  154  ls

  155  k config set-credentials john --client-certificate=/root/test/john.crt --client-key=/root/test/john.key

  156  vi .kucon

  157  vi ~/.kube/config

  158  k config set-context mycontext --user=john --cluster=kubernetes

  159  vi ~/.kube/config

  160  vi role.yaml

  161  k apply -f role.yaml

  162  k create rolebinding myrolebinding --role=myrole --user=john

  163  k config use-context mycontext

  164  k get po

  165  k run nginx --image=nginx

  166  k get deploy

  167  k get po -w

  168  vi role.yaml

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

        name: myrole

        namespace: default

rules:

    - apiGroups: [""]

      resources: ["pods"]

      verbs: ["list", "watch", "get"]

  169  k get po

  170  k get deploy

  171  k config get-contexts

  172  k config use-context kubernetes-admin@kubernetes

  173  k get deploy

Cluster maintenance:

  181  k get po -o wide

  182  k create deploy test --image=nginx

  183  k get po -o wide

  184  k scale deploy test --replicas=5

  185  k get po -o wide

  186  k drain worker1

  187  k drain worker1 --force --ignore-daemonsets

  188  k get po -o wide

  189  k get no

  190  k uncordon worker1

  191  k get no

Day 10:

Liveness & Readiness Probe

  193  alias k=kubectl

  194  k get no

  195  cd

  196  k get no

  197  k get all

  198  k delete deploy test

  199  k get all

  200  k delete po frontend

  201  k get all

  202  k explain po.spec.containers.readinessProbe

  203  k run nginx --image=nginx --port=80 --dry-run=client -o yaml > pod.yaml

  204  vi pod.yaml

apiVersion: v1

kind: Pod

metadata:

  creationTimestamp: null

  labels:

    run: nginx

  name: nginx

spec:

  containers:

  - image: nginx

    name: nginx

    ports:

    - containerPort: 80

    resources: {}

    readinessProbe:

      initialDelaySeconds: 5

      periodSeconds: 2

      httpGet:

        path: /

        port: 80

    livenessProbe:

      initialDelaySeconds: 5

      periodSeconds: 2

      httpGet:

        path: /

        port: 80

  dnsPolicy: ClusterFirst

  restartPolicy: Always

status: {}

  205  k apply -f pod.yaml

  206  k get po

  207  k describe po nginx

  208  k delete po nginx

  209  vi pod.yaml

  210  k apply -f pod.yaml

  211  k get po

  212  vi pod.yam

Helm Charts

  219  snap install helm --classic

  220  helm create myapp

  221  ls myapp/

  222  cd myapp/

  223  vi Chart.yaml

apiVersion: v2

name: myapp

description: A Helm chart for Kubernetes

# A chart can be either an 'application' or a 'library' chart.

#

# Application charts are a collection of templates that can be packaged into versioned archives

# to be deployed.

#

# Library charts provide useful utilities or functions for the chart developer. They're included as

# a dependency of application charts to inject those utilities and functions into the rendering

# pipeline. Library charts do not define any templates and therefore cannot be deployed.

type: application

# This is the chart version. This version number should be incremented each time you make changes

# to the chart and its templates, including the app version.

# Versions are expected to follow Semantic Versioning (https://semver.org/)

version: 0.1.0

# This is the version number of the application being deployed. This version number should be

# incremented each time you make changes to the application. Versions are not expected to

# follow Semantic Versioning. They should reflect the version the application is using.

# It is recommended to use it with quotes.

appVersion: "1.16.0"

  224  rm values.yaml

  225  ls charts/

  226  rm -rf charts/

  227  ls

  228  cd templates/

  229  rm -rf \*

  230  ls

  231  cp ~/deploy.yaml .

  232  vi deploy.yaml

  233  vi ../values.yaml    => under myapp folder

app:

  name: myapp

  replicas: 5

  image: rajendrait99/first:1.0

  port: 8080

  type: NodePort

  234  vi deploy.yaml    => under templates folder

apiVersion: apps/v1

kind: Deployment

metadata:

  creationTimestamp: null

  labels:

    app: {{.Values.app.name}}

  name: {{.Values.app.name}}

spec:

  replicas: {{.Values.app.replicas}}

  selector:

    matchLabels:

      app: {{.Values.app.name}}

  strategy: {}

  template:

    metadata:

      creationTimestamp: null

      labels:

        app: {{.Values.app.name}}

    spec:

      containers:

      - image: {{.Values.app.image}}

        name: {{.Values.app.name}}

        ports:

        - containerPort: {{.Values.app.port}}

        resources: {}

status: {}

  235  vi ../values.yaml

  236  vi deploy.yaml

  237  vi service.yaml   => templates folder

apiVersion: v1

kind: Service

metadata:

  name: {{.Values.app.name}}

spec:

  selector:

    app: {{.Values.app.name}}

  ports:

    - port: {{.Values.app.port}}

      targetPort: {{.Values.app.port}}

  type: {{.Values.app.type}}

  238 vi ../values.yaml

  239  ls

  240  vi ../values.yaml

  241  cd ..

  242  ls

  243  cd ..

  244  helm template myapp myapp/

  245  helm install myapp myapp/

  246  k get all

  247  k get no

  248  k get no -o wide

  249  curl 172.31.4.187:31992

  250  helm list

  251  helm uninstall myapp

  252  helm install myapp myapp/

  253  k get all

  254  ls

  255  cp -R myapp/ yourapp

  256  vi yourapp/Chart.yaml

  257  vi yourapp/values.yaml

  258  helm install yourapp yourapp/

  259  k get all

  260  curl 172.31.4.187:31961

  261  curl 172.31.4.187:30648

  262  cd myapp/

  263  ls

  264  cd templates/

  265  ls

  266  cd ..

  267  vi yourapp/values.yaml

  268  helm upgrade yourapp yourapp/

  269  k get all

  270  helm -h

helm uninstall myapp yourapp