1. Deploy a pod named nginx-pod using the nginx:alpine image.

kubectl run nginx-pod --image=nginx:alpine

1. Deploy a messaging pod using the redis:alpine image with the labels set to tier=msg.

kubectl run messaging --image=redis:alpine --labels=tier=msg

1. Create a namespace webengg

kubectl create ns webengg

1. Get the list of nodes and copy it to /opt/output\_nodes.json format.

kubectl get nodes -o json > /opt/outputs/nodes-z3444kd9.json

5) Create service messaging-service and expose the pod messaging to port 6379 (refer pod name to 2nd question)

kubectl expose pod messaging --port 6379 --name messaging-service

1. Create a deployment named hr-web-app using the image webapp-color with 2 replicas.

kubectl create deployment hr-web-app --image=kodekloud/webapp-color --replicas=2

1. Create a static pod busybox with sleep command 1000

apiVersion: v1

kind: Pod

metadata:

name: static-busybox

spec:

containers:

- name: busybox

image: busybox

commands:

- "sleep"

- "1000"

8) Create a pod named temp-bus using redis:alpine image in finance namespace.

kubectl run temp-bus --image=redis:alpine -n finance

1. Expose service hr-web-app for the app hr-web-app on port 30082.

kubectl expose deployment hr-web-app --type=NodePort –port=30082 --name hr-web-app --dry-run=client -o yaml > hr-web-app-service.yaml

1. Use JSON path to retrieve node OS information and save it /opt/nodeosinfo.txt

kubectl get nodes -o=jsonpath=’{.items[\*].status.nodeInfo.osImage}’ > /opt/outputs/nodes\_os\_x43kj56.txt

11) Create a Persistent volume with the below spec:

* Volume Name: pv-analytics
* Storage: 100Mi
* Access modes: ReadWriteMany
* Host Path: /pv/data-analytics

apiVersion: v1

kind: PersistentVolume

metadata:

name: pv-analytics

spec:

persistentVolumeReclaimPolicy: Retain

accessModes:

- ReadWriteMany

capacity:

storage: 100Mi

hostPath:

path: /pv/data-analytics

12) Etcd backup to be taken:

ETCDCTL\_API=3 etcdctl version

ETCDCTL\_API=3 etcdctl --endpoints=https://127.0.0.1:2379

--cacert=/etc/kubernetes/pki/etcd/ca.crt

--cert=/etc/kubernetes/pki/etcd/server.crt

--key=/etc/kubernetes/pki/etcd/server.key

snapshot save /opt/etcd-backup.db

13) Create a Pod called redis-storage with image: redis:alpine with a Volume of type emptyDir that lasts for the life of the Pod.

* Pod named 'redis-storage' created
* Pod 'redis-storage' uses Volume type of emptyDir
* Pod 'redis-storage' uses volumeMount with mountPath = /data/redis

apiVersion: v1

kind: Pod

metadata:

name: redis-storage

spec:

containers:

- name: redis

image: redis:alpine

volumeMounts:

- mountPath: /tmp

name: cache-volume

volumes:

- name: cache-volume

emptyDir: {}

14) Create a new pod called super-user-pod with image busybox:1.28. Allow the pod to be able to set system\_time. The container should sleep for 4800 seconds.

* Pod: super-user-pod
* Container Image: busybox:1.28
* SYS\_TIME capabilities for the conatiner?

apiVersion: v1

kind: Pod

metadata:

name: super-user-pod

spec:

containers:

- name: super-user-pod

image: busybox:1.128

command:

- "sleep"

- "4800"

securityContext:

capabilities:

add: ["SYS\_TIME"]

15) A pod definition file is created at /root/CKA/use-pv.yaml. Make use of this manifest file and mount the persistent volume called pv-1. Ensure the pod is running and the PV is bound.

mountPath: /data

persistentVolumeClaim Name: my-pvc

* persistentVolume Claim configured correctly
* pod using the correct mountPath
* pod using the persistent volume claim?

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: my-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 10Mi

######################

apiVersion: v1

kind: Pod

metadata:

labels:

run: use-pv

name: use-pv

spec:

containers:

- image: nginx

name: use-pv

volumeMounts:

- mountPath: /data

name: data-volume

volumes:

- name: data-volume

persistentVolumeClaim:

claimName: my-pvc

16) Create a new deployment called nginx-deploy, with image nginx:1.16 and 1 replica. Record the version. Next upgrade the deployment to version 1.17 using rolling update. Make sure that the version upgrade is recorded in the resource annotation.

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deploy

labels:

app: nginx

spec:

replicas: 1

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.16

ports:

- containerPort: 80

kubectl create deployment nginx-deploy --image=nginx:1.16

kubectl create deployment nginx-deploy --image=nginx:1.16

kubectl apply -f deploy.yaml –record

kubectl create deployment nginx1 --image=nginx:1.16 --replicas=1

kubectl set image deployment/nginx-deploy nginx=1.17 –record

kubectl rollout history deployment nginx-deploy

kubectl rollout history deployment nginx-deploy

17) Create a new user called john. Grant him access to the cluster. John should have permission to create, list, get, update and delete pods in the development namespace . The private key exists in the location: /root/CKA/john.key and csr at /root/CKA/john.csr.

Important Note: As of kubernetes 1.19, the CertificateSigningRequest object expects a signerName.

Please refer the documentation to see an example. The documentation tab is available at the top right of terminal.

* CSR: john-developer Status:Approved
* Role Name: developer, namespace: development, Resource: Pods
* Access: User 'john' has appropriate permissions

Follow the steps under normal user section.

<https://kubernetes.io/docs/reference/access-authn-authz/certificate-signing-requests/>

18) Create a nginx pod called nginx-resolver using image nginx, expose it internally with a service called nginx-resolver-service. Test that you are able to look up the service and pod names from within the cluster. Use the image: busybox:1.28 for dns lookup. Record results in /root/CKA/nginx.svc and /root/CKA/nginx.pod

kubectl run nginx-resolver –image=nginx

kubectl expose nginx-resolver –name=nginx-resolver-service --port=80 --targetPort=80 ---type=ClusterIP

kubectl run test-nslookup --image=nginx --rm -it -- nslookup nginx-service > /root/nginx.svc

kubectl run test-nslookup --image=nginx --rm -it -- nslookup 10-32-0-5.default.pod > /root/nginx.pod

* Pod: nginx-resolver created
* Service DNS Resolution recorded correctly
* Pod DNS resolution recorded correctly

19) Create a static pod on node01 called nginx-critical with image nginx and make sure that it is recreated/restarted automatically in case of a failure.

Use /etc/kubernetes/manifests as the Static Pod path for example.

apiVersion: v1

kind: Pod

metadata:

name: nginx-critical

spec:

containers:

- name: nginx-critical

image: nginx

restartPolicy: OnFailure

ports:

- containerPort: 80

protocol: TCP

20) Create a new service account with the name pvviewer. Grant this Service account access to list all PersistentVolumes in the cluster by creating an appropriate cluster role called pvviewer-role and ClusterRoleBinding called pvviewer-role-binding.

Next, create a pod called pvviewer with the image: redis and serviceAccount: pvviewer in the default namespace.

* ServiceAccount: pvviewer
* ClusterRole: pvviewer-role
* ClusterRoleBinding: pvviewer-role-binding
* Pod: pvviewer
* Pod configured to use ServiceAccount pvviewer ?

kubectl create serviceaccount pvviewer

kubectl create clusterrole pvviewer-role –resources=persistentvolumes –verb=list

kubectl create clusterrolebinding pvviewer-role-binding --clusterrole=pvviewer-role –serviceaccount=default:pvviewer.

apiVersion: v1

kind: Pod

metadata:

name: pvviewer

labels:

run: pvviewer

name: pvviewer

spec:

containers:

* name: pvviewer

image: redis

imagePullPolicy: IfNotPresent

serviceAccountName: pvviewer

21) List the InternalIP of all nodes of the cluster. Save the result to a file /root/CKA/node\_ips.

Answer should be in the format: InternalIP of controlplane<space>InternalIP of node01 (in a single line)

kubectl get nodes -o jsonpath='{.items[\*].status.addresses[?(@.type=="InternalIP")].address}'

22) Create a pod called multi-pod with two containers.

Container 1, name: alpha, image: nginx

Container 2: name: beta, image: busybox, command: sleep 4800

* Environment Variables:
* container 1:
* name: alpha
* Container 2:
* name: beta
* Pod Name: multi-pod
* Container 1: alpha
* Container 2: beta
* Container beta commands set correctly.
* Container 1 Environment Value Set
* Container 2 Environment Value Set

apiVersion: v1

kind: Pod

metadata:

name: multi-pod

spec:

containers:

* name: alpha

image: nginx

env:

* name: name

value: alpha

name: beta

image: busybox

command: [“sleep”, “4800”]

env:

* Name: name

Value: beta

23) Create a Pod called non-root-pod , image: redis:alpine

runAsUser: 1000

fsGroup: 2000

24) We have deployed a new pod called np-test-1 and a service called np-test-service. Incoming connections to this service are not working. Troubleshoot and fix it.

Create NetworkPolicy, by the name ingress-to-nptest that allows incoming connections to the service over port 80.

Important: Don't delete any current objects deployed.

Important: Don't Alter Existing Objects!

NetworkPolicy: Applied to All sources (Incoming traffic from all pods)?

NetWorkPolicy: Correct Port?

NetWorkPolicy: Applied to correct Pod?

25) Taint the worker node node01 to be Unschedulable. Once done, create a pod called dev-redis, image redis:alpine, to ensure workloads are not scheduled to this worker node. Finally, create a new pod called prod-redis and image: redis:alpine with toleration to be scheduled on node01.

key: env\_type, value: production, operator: Equal and effect: NoSchedule

Key = env\_type

Value = production

Effect = NoSchedule

pod 'dev-redis' (no tolerations) is not scheduled on node01?

Create a pod 'prod-redis' to run on node01

kubectl taint node node01 env\_type=production:NoSchedule

tolerations:

* Effect: NoSchedule

Key: env\_type

value: production

operator: equal

26) Create a pod called hr-pod in hr namespace belonging to the production environment and frontend tier .

image: redis:alpine

Use appropriate labels and create all the required objects if it does not exist in the system already.

hr-pod labeled with environment production?

hr-pod labeled with tier frontend?

27) A kubeconfig file called super.kubeconfig has been created under /root/CKA. There is something wrong with the configuration. Troubleshoot and fix it.

Fix /root/CKA/super.kubeconfig

28) We have created a new deployment called nginx-deploy. scale the deployment to 3 replicas. Has the replica's increased? Troubleshoot the issue and fix it.

deployment has 3 replicas

Verify if kube-controller.yaml is configured and running properly.

29) . A kubernetes worker node, named wk8s-node-0 is in state NotReady. Investigate why this is the case, and perform any appropriate steps to bring the node to a Ready state, ensureing that any changes are made permanent.

You can assume elevated privileages sudo -i

Ans: ssh to node01

systemctl enable kubelet or systemctl enable --now kubelet

systemctl restart kubelet

systemctl status kubelet

Q30) : From the pod lable name=cpu-utilizer, find pods running high CPU workloads and write the name of the pod consuming most to the file /opt/KUTR00401/KUTROO401.txt (Which is already exist)

 Ans: kubectl top pods -l name=cpu-utilizer

 kubectl top pods -l name=cpu-utilizer --sort-by=cpu --no-headers | cut -f1 -d" "" | head -n1 > /opt/KUTR00401/KUTROO401.txt

Q31) Monitor the logs of pod foo and:

. Extract log lines corrosponding to error " unable-to-access-website"

. Write them to /opt/KUTR00101/f00

Ans: kubectl logs foo | grep unable-to-access-website > /opt/KUTR00101/f00

cat /opt/KUTR00101/f00

 Q32): Without changing its existing containers, an existing Pod needs to be integrated into Kubernetes's built-in logging architecture (eg: kubectl logs). Adding a steaming sidecar container is a good and common way to accomplish this requirment.

 Task: Add a busybox sidecar container to the existing Pod 11-factor-app. The new sidecar container had to run the following command :

 /bin/sh -c tail -n+1 /var/log/11-factor-app.log

 Use a volume mountnamed logs to make the file /var/log/11-factor-app.log available to the sidecar container.

 Don't modify the existing container , Don't modify the path of the log file, both container must be access it at containers must access it at /var/log/110factor-app.log.

Q33) Q 13. Create a new PersistentVolumeClaim:

. Name: pv-volume.

. Class: csi-hostpath-sc

. Capacity: 10Mi\

[17:48, 19/11/2020] +91 90361 68476: apiVersion: v1

kind: PersistentVolumeClaim

metadata:

  name:pv-volume

spec:

  storageClassName: csi-hostpath-sc

  accessModes:

    - ReadWriteOnce

  resources:

    requests:

      storage: 10Mi

kubectl create -f pv-claim.yaml

kubectl get pvc

[17:49, 19/11/2020] +91 90361 68476: Second create pod and claim the space from PV

Q34) Create a persistent volume with name app-data, of capacity 1Gi and access mode ReadWriteMany, The type of volume is hostPath and its location is /srv/app-data

   Ans:

apiVersion: v1

kind: PersistentVolume

metadata:

  name: app-data

spec:

  storageClassName: manual

  capacity:

    storage: 1Gi

  accessModes:

    - ReadWriteMany

  hostPath:

    path: "/srv/app-data"

kubectl get pv

Q35) Create a pod named kucc4 with single app container for each of the following images running inside (there may be between q and 4 images specified):

nginx + redis + memcached + consul

Ans: kubectl run kucc4 --image-nginx --dry-run -o yaml > 4.yaml

apiVersion: batch/v1

kind: pod

metadata:

  labels:

     run: kucc4

  name: kucc4

spec:

  containers:

  - name: nginx

    image: nginx

  - name: redis

    image: redis

  - name: memcashed

    image: memcashed

  - name: consul

    image: consul

Q36) Check to see how many nodes are ready (not including nodes tainted NoSchedule) and write the number to

/opt/KUSC00402/kusc00402.txt

Ans: kubectl get nodes echo "2" > /opt/KUSC00402/kusc00402.txt

Q37) Schedule a pod as follows:

. Name: nginx-kusc00401

. Image: nginx

. Node Selector: disk=ssd

Ans: kubectl run nginx-kusc00401 --image --dry-run -o yaml > kuse.yaml

Q38) Schedule a pod as follows:

. Name: nginx-kusc00401

. Image: nginx

. Node Selector: disk=ssd

Ans: kubectl run nginx-kusc00401 --image --dry-run -o yaml > kuse.yaml

Q39) Schedule a pod as follows:

. Name: nginx-kusc00401

. Image: nginx

. Node Selector: disk=ssd

Ans: kubectl run nginx-kusc00401 --image --dry-run -o yaml > kuse.yaml

apiVersion: v1

kind: Pod

metadata:

  name: nginx

  labels:

    run: nginx-kusc00401

spec:

  containers:

  - name: nginx

    image: nginx

  nodeSelector:

    disktype: ssd

#kubectl create -f kuse.yaml

Q40) Q 7. Create a new nginx Ingress resource as follows:

  .Name: pong

  .Namespace: in-internal

  .Exposing service service hello on path /hello using service port 5678

  Availablity of sevice can be checked by curl -kL <IN Terminal\_IP>/hello

  Reff URL: [https://kubernetes.io/docs/concepts/services-networking/ingress/](https://urldefense.com/v3/__https:/kubernetes.io/docs/concepts/services-networking/ingress/__;!!OUOUp-U!sfgNvWySIgBe-Wg7dWX4w1P0oEefcZl2p5pj3ukeDGdO3FcX3NxPa_Uw0At8Kj5TBWfU$)

  Ans: vim ingress.yaml

  apiVersion: [networking.k8s.io/v1](https://urldefense.com/v3/__http:/networking.k8s.io/v1__;!!OUOUp-U!sfgNvWySIgBe-Wg7dWX4w1P0oEefcZl2p5pj3ukeDGdO3FcX3NxPa_Uw0At8KmGi9vzz$)

kind: Ingress

metadata:

  name: pong

  namespace: in-internal

  annotations:

[nginx.ingress.kubernetes.io/rewrite-target](https://urldefense.com/v3/__http:/nginx.ingress.kubernetes.io/rewrite-target__;!!OUOUp-U!sfgNvWySIgBe-Wg7dWX4w1P0oEefcZl2p5pj3ukeDGdO3FcX3NxPa_Uw0At8KuZH7HnK$): /

spec:

  rules:

  - http:

      paths:

      - path: /hello

        pathType: Prefix

        backend:

          service:

            name: hello

            port:

              number: 5678

kubectl create -f ingress.yaml

kubectl --namespace=in-internel describe ingress pong

Q41) 6. Reconfigure the existing deployment front-end and add a port specification named http exposing port 80/tcp of the existing container nginx

 create a new service named front-end-svc exposing the container port http.

 Configure the service also expose the individual pods via a NodePort on the nodes on which they are scheduled.

  Ans: kubectl get deployment.apps

  kubectl edit deployment.apps front-end

  Go to container spec > add the following under the name: nginx

  containers:

  - image: nginx:1.14.2

    name: nginx

    ports:

    - containerPort: 80

      name: "http"

      protocol: TCP

  :wq for save and exit

   kubectl describe deployment.apps font-end

   kubectl get svc

   kubectl expose deployment front-end --name=front-end-svc --port=80 --type-NodePort --protocol=TCP

Q42) Q 5: Create a new network policy named allow-port-from-namespace that allows pods in the existing namespace my-app to connect to port 9000 of other pods in the same namespace.

Ensure that the new NetworkPolicy

. does not allow access to pods not listening on port 9000

. does not allow access from Pods not in namespace my-app

Ans:

Referance link: [https://kubernetes.io/docs/concepts/services-networking/network-policies/](https://urldefense.com/v3/__https:/kubernetes.io/docs/concepts/services-networking/network-policies/__;!!OUOUp-U!vmsQ4GDybjrDCEUgH2XX2k4WfQy5npLT6LmxcmvvP1VSA5ZIwi0Tr0uq3eH5INm2qHBx$)

vim netpolicy.yaml

apiVersion: [networking.k8s.io/v1](https://urldefense.com/v3/__http:/networking.k8s.io/v1__;!!OUOUp-U!vmsQ4GDybjrDCEUgH2XX2k4WfQy5npLT6LmxcmvvP1VSA5ZIwi0Tr0uq3eH5INqlyG2m$)

kind: NetworkPolicy

metadata:

  name: tallow-port-from-namespace

  namespace: my-app

spec:

  podSelector: {}

  policyTypes:

  - Ingress

  ingress:

  - from:

    - namespaceSelector:

        matchLabels:

          project: my-app

    ports:

    - protocol: TCP

      port: 9000

kubectl create -f netpolicy.yaml

Q43) Q 4: Create a snapeshote of the existing etcd instance running at [https://127.0.0.1:2379](https://urldefense.com/v3/__https:/127.0.0.1:2379__;!!OUOUp-U!vmsQ4GDybjrDCEUgH2XX2k4WfQy5npLT6LmxcmvvP1VSA5ZIwi0Tr0uq3eH5IAMKGEHQ$), saving the snapshot to /srrv/data/etcd-snapshot.db

Next restore an existing, previous snapshot located at /var/lib/backup/etcd-snapshot-previous.db

CA cert: /op/KUNIN00601/ca.crt

Client crt /op/KUNIN00601/etcd-client.crt

Client key /op/KUNIN00601/etcd-client.key

Ans: ETCDCTL\_API=3 etcdctl --endpoints-[https://127.0.0.1:2379](https://urldefense.com/v3/__https:/127.0.0.1:2379__;!!OUOUp-U!vmsQ4GDybjrDCEUgH2XX2k4WfQy5npLT6LmxcmvvP1VSA5ZIwi0Tr0uq3eH5IAMKGEHQ$) --cacert=/op/KUNIN00601/ca.crt --cert=/op/KUNIN00601/etcd-client.crt --key=/op/KUNIN00601/etcd-client.key snapshot save /srrv/data/etcd-snapshot.db

# verify the snapshot

ETCDCTL\_API=3 etcdctl --write-out=table snapshot status /srrv/data/etcd-snapshot.db

ETCDCTL\_API=3 etcdctl --endpoints-[https://127.0.0.1:2379](https://urldefense.com/v3/__https:/127.0.0.1:2379__;!!OUOUp-U!vmsQ4GDybjrDCEUgH2XX2k4WfQy5npLT6LmxcmvvP1VSA5ZIwi0Tr0uq3eH5IAMKGEHQ$) --cacert=/op/KUNIN00601/ca.crt --cert=/op/KUNIN00601/etcd-client.crt --key=/op/KUNIN00601/etcd-client.key snapshot restore /var/lib/backup/etcd-snapshot-previous.db

export ETCDCTL\_API=3

etcdctl snapshot save --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key --endpoints=127.0.0.1:2379 /opt/etcd-backup.db

Q44) Given an existing kubernetes cluster running version 1.18.8, upgrade all of the kubernetes control plane and node components on the master node only to version 1.19.0

You are expected to upgrade kubelet and kubectl on the master node.

Ans: kubectl drain mk8s-master-0 --ignore-daemonsets (if needed add --force --delete-local-data)

On master

root@controlplane:~# kubectl drain controlplane --ignore-daemonsets

root@controlplane:~# apt update

root@controlplane:~# apt-get install kubeadm=1.20.0-00

root@controlplane:~# kubeadm upgrade plan v1.20.0

root@controlplane:~# kubeadm upgrade apply v1.20.0

root@controlplane:~# apt-get install kubelet=1.20.0-00

root@controlplane:~# systemctl daemon-reload

root@controlplane:~# systemctl restart kubelet

root@controlplane:~# kubectl uncordon controlplane

root@controlplane:~# kubectl drain node01 --ignore-daemonsets

On node.

root@node01:~# apt update

root@node01:~# apt-get install kubeadm=1.20.0-00

root@node01:~# kubeadm upgrade node

root@node01:~# apt-get install kubelet=1.20.0-00

root@node01:~# systemctl daemon-reload

root@node01:~# systemctl restart kubelet

root@controlplane:~# kubectl uncordon node01

root@controlplane:~# kubectl get pods -o wide | grep gold (make sure this is scheduled on node)

kubectl get nodes

ssh mk8s-master-0

apt-get update

apt-get install -y kubeadm=1.19.0-00

kubeadm upgrade plan

kubeadm upgrade apply v1.19.0

kubeadm version

sudo kubeadm upgrade node

sudo kubeadm upgrade apply

apt-get install kubelet-1.19.0-00 -y

apt-get install kubectl-1.19.0-00 -y

sudo systemctl daemon-reload

sudo systemctl restart kubelet

kubectl uncordoned mk8s-master-0

kubectl get nodes

Referance: [https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/](https://urldefense.com/v3/__https:/kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/__;!!OUOUp-U!ra_oGnFFPM1wcALY901oEUEjBr7qMzlyYVs7ZR4jIC-hRnhk9rRtG5RNTJV8OXlozzgC$)

Q45) Set a node named eks-node-0 as unavailale and reschedule all the pods running on it.

Ans: kubectl drain eks8-node0 --ignore-daemonsets --force --delete-local-data

kubectl get nodes

kubectl describe node eks8-node0

Q46) Create a new Clusterrole name deployment-clusterrole , which only allows to create the following resources types

deployment,statefulSet,Daemonset

Q47) Create a new Serviceaccount named cicd-tocken in the existing namespace app-team1

bind the new clisterrole deplyment-clusterrole to the new service account cicd-tocken limited to the namespace app-team1

Ans: kubectl create clusterrole deployment-clusterrole --verb=create --resource=Deployment,StatefulSet,DaemonSet

kubectl create sa cicd-tocken --namespace=app-team1

kubectl create clusterrolebinding deployment-bind --clusterrole=deployment-clusterrole --serviceaccount=app-team1:cicd-tocken

Q48) Create a pod called secret-1401 in the admin1401 namespace using the busybox image. The container within the pod should be called secret-admin and should sleep for 4800 seconds.

The container should mount a read-only secret volume called secret-volume at the path /etc/secret-volume. The secret being mounted has already been created for you and is called dotfile-secret.

kubectl run secret-1401 --image=busybox --dry-run=client -oyaml --command -- sleep 4800 > admin.yaml

---

apiVersion: v1

kind: Pod

metadata:

creationTimestamp: null

labels:

run: secret-1401

name: secret-1401

namespace: admin1401

spec:

volumes:

- name: secret-volume

# secret volume

secret:

secretName: dotfile-secret

containers:

- command:

- sleep

args:

- "4800"

image: busybox

name: secret-admin

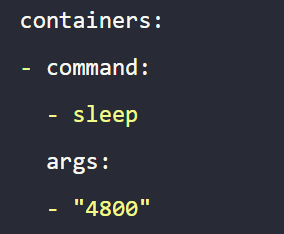
# volumes' mount path

volumeMounts:

- name: secret-volume

readOnly: true

mountPath: "/etc/secret-volume"



command:

- "sleep"

- "4800"

command: [“sleep”,”4800”]

- command:

- sleep

- "1000"

**JSON PATH:**

kubectl get nodes -o=jsonpath='{.items[\*].metadata.name}' > /opt/outputs/node\_names.txt

$.prizes[?(@.year == 2014)].laureates[\*].firstname

kubectl get nodes -o=jsonpath='{.items[\*]

kubectl config view --kubeconfig=my-kube-config -o jsonpath="{.users[\*].name}" > /opt/outputs/users.txt

kubectl config view --kubeconfig=my-kube-config -o jsonpath="{.users[\*].name}"/opt/outputs/aws-context-name

kubectl config view --kubeconfig=my-kube-config -o jsonpath="{.contexts[?(@.context.user=='aws-user')].name}" > /opt/outputs/aws-context-name

kubectl get nodes -o=custom-columns

kubectl get nodes -o jsonpath='{.items[\*].status.nodeInfo.osImage}' > /opt/outputs/nodes\_os.txt

kubectl get pv --sort-by=.spec.capacity.storage > /opt/outputs/storage-capacity-sorted.txt

kubectl get pv --sort-by=.spec.capacity.storage -o=custom-columns=NAME:.metadata.name,CAPACITY:.spec.capacity.storage > /opt/outputs/pv-and-capacity-sorted.txt

kubectl get pv -o jsonpath='{.items[\*].spec.capacity.storage}' > /opt/outputs/storage.txt

kubectl get nodes -o=jsonpath='{.items[\*].metadata.name}' > /opt/outputs/nodes.txt

kubectl get nodes -o jsonpath='{.items[\*].status.nodeInfo.osImage}' > /opt/outputs/nodes\_os.txt

kubectl get nodes -o jsonpath='{.items[\*].status.addresses[?(@.type=="ExternalIP")].address}'

kubectl get nodes -o jsonpath='{.items[\*].status.addresses[?(@.type=="InternalIP")].address}'

kubectl get nodes -o=jsonpath='{.items[\*].metadata.name}'

kubectl -n admin2406 get deployment -o custom-columns=DEPLOYMENT:.metadata.name,CONTAINER\_IMAGE:.spec.template.spec.containers[].image,READY\_REPLICAS:.status.readyReplicas,NAMESPACE:.metadata.namespace --sort-by=.metadata.name > /opt/admin2406\_data

kubectl cluster-info --kubeconfig /root/CKA/admin.kubeconfig

kubectl get deployment --sort-by=.metadata.name -n labs -o=custom-columns=DEPLOYMENT:.metadata.name,PORT:.spec.containers[\*].ports.containerPort

kubectl get nodes -o=jsonpath='{.items[\*].metadata.name}' -o=custom-columns=NODENAME:.metadata.name

kubectl get pv --sort-by=.spec.capacity.storage -o=custom-columns=STORAGE:.spec.capacity.storage,ACCESSMODES:.spec.accessModes

kubectl get nodes -o=jsonpath=’{.items[\*].status.nodeInfo.osImage}’

kubectl get nodes -o jsonpath='{.items[\*].status.nodeInfo.osImage}' > /opt/outputs/nodes\_os.txt

kubectl get nodes -o=jsonpath=’{.items[\*].status.nodeInfo.osImage}{“\t”}{.items[\*].status.nodeInfo.osImage}’ -o=custom-columns=OSIMAGE:.status.nodeInfo.osImage,ARCHITECHTURE:.status.nodeInfo.architechture

kubectl run nginx-envpod --image=nginx --restart=Never --env=ENV\_APP=WEBAPP

kubectl autoscale deployment/my-nginx --min=1 --max=3 --cpu-percent=80

kubectl get hpa

kubectl create secret generic app-secret --from-file=app\_secret.properties

kubectl create secret generic dev-db-secret --from-literal=username=devuser

--from-literal=password='S!B\\*d$zDsb'

kubectl create configmap app-config --from-literal=app\_color=blue --fromliteral=app\_type=prod

kubectl create configmap app-config --from-literal=app\_color=blue --fromliteral=app\_type=prod

kubectl run hello --schedule="\*/1 \* \* \* \*" --restart=OnFailure --image=busybox -- /bin/sh -c "date

kubectl run busybox --image=busybox -- /bin/sh -c "sleep 3600"

k get no -o jsonpath="{range .items[\*]}{.metadata.name}

{.spec.taints[?(@.effect=='NoSchedule')].timeAdded} {\"\n\"} {end}" | awk '$2 !=""'

set nu set ic set expandtab set shiftwidth=2 set tabstop=2