1. Cluster upgrade from 1.21.0 to 1.21.1 in control plane only.

2. Create a pv for the given specifications.

Capacity / accessmode/ Storageclass/ EmptyDir

3. Create a pvc and attach to an existing sc and create a pod with created pvc. Then use "kubectl edit" or "kubectl patch" to increase the storage of the PVC from 10Mi to 70Mi and also change it to emptyDir.

Create PVC / Create POD /

4. Write a network policy to allow the ingress from any pods in a given namespace to any pod in another namespace through an specific port.

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

**kind**: NetworkPolicy

**metadata**:

**name**: test-network-policy

**namespace**: Blue

**spec**:

**podSelector**:

**matchLabels**:

**policyTypes**:

- Ingress

**ingress**:

- **from**:

- **namespaceSelector**:

**matchLabels**:

**podSelector**:

**matchLabels**:

**ports**:

- **protocol**: TCP

**port**: 6379

5. Run a multi container pod with 2 containers.

6. Run a pod with one init container having command to write logs in to a file. (Need to use the given config)

7. Create a sa , create a cluster role for creating deployments, statefulsets, daemonsets and create a role binding for a specific namespace.

kubectl create clusterrole --resource ??

8. Get the highest cpu consuming pod and write the name in a file.

9. Get all the nodes available in the cluster without the taint taint=NoSchedule and write the number in a file.

10. Create an ingress resource with one rule one path.

11. For a given deployment, you are asked to edit it and add "tcp 80" port on the container, then expose the container of that deployment as a service.

12. Scale the deployment to 5 replaces and record.

13. Create a side car logging container inside the nginx container pod with a volume mount.

14. Do the ETCD backup and restore.

15. Worker node not working, need to fix the worker node.

16. Create a deployment with given spec

17. Mark the node1 unavailable and reschedule the pods.

------------SHS—

\*\*when required, you can create, delete, edit, etc the existing objects, and if that is NOT allowed, they will indicate that you should not edit, or modify an specific object.

\*\*when required use a particular Namespace, they will indicate, if not indicated, then consider it as default one.

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

- kubeadmin upgrade from 1.20 to 1.20.1 in master node only, not in worker nodes.

- perform an etcd backup and perform backup a restore

\* - In one sidecar application they gave you the name of the one ".log" file and the path where that .log file is located, so you are asked to mounth that file in the pod definition file of the app.

\* - You are asked to monitor the logs of a pod and save in a given directory, only the lines which are matching with some kind of error, for example: "Entry not found"

[example of logs in the excercise:

line1 - Host is not reachable

line2 - Command not found

line3 - Entry not found

line4 - Fatal error ......]

\* - Create a pod runing a single container which uses 4 images: niginx + busybox + redis

\* - From the cluster, get the pods which are having higher CPU utilizacion and save the output in a given directory.

[example of pods and their CPU usage:

name PCU

pod1 0

pod2 45

pod3 88

pod4 0

pod5 0

pod6 23..]

- From the cluster, get all the nodes in Ready state and save the output in a given directory [don't include the nodes tainted or unavailable]

- Mark the node1 unavailable and reeschedule the pods. [the cluster only has master + node1 , so the pods are all in node1]

\* - One excercise asks to mount a PVC in a Pod, and they gave you the specifications of the PVC:

>> class: csi-hostpath-sc

storage: 10Mi

then, they ask you once it's mounted in the Pod, use "kubectl edit" or "kubectl patch" to increase the storage of the PVC from 10Mi to 70Mi and also change it to emptyDir [if I am not wrong], and record. [I guess they refer by "recording" to record the changes made for the PVC]

- in a cluster where there are kinda 5 namespaces: To Create a network policy that allows the ingress of the Pods of the Namespace "A" through an specific port, i.e port "6300" to the applications in namespace "B". Also, the ingress of pods from other namespaces should not be allowed, and ingress from other ports than 6300 should not be allowed.

- Troubleshoot the issue in one cluster [the node1 is Not ready] so fix that.

\* - To create a PV with a given hostpath, namespace and some other given specifications.

- To create a Service account, cluster role and cluster role binding for: daemonset, deployment, statefulset

- Increase the replicas of a given deployment.

- For a given deployment, you are asked to edit it and add "tcp 80" port on the container, then expose the container of that deployment as a service.

- in Some excercise it was requested to create a Nodeport for an existing app to expose it as a service. [don't remember more details]

-----------DES-

1. Update k8 to 1.21.1 from 1.21.0 in cp only

2. Create a pv

3. Create a pvc and attach to an existing sc, create a pod with pic

4. Write a np to allow traffic only to port 8080 from all pods to any pod in a given namespace

5. Run a multi container pod

6. Run a pod with one init container having command “sleep 60” and image busybody

7. Create a sa and associate to a running pod nginx; you may recreate the pod

8. Get the highest cpu consuming pod and write the name in a file

9. Get all the nodes available in the cluster without the taint taint=NoSchedule and write the number in a file

10. Create an ingress resource with one rule two paths ; URLs and paths are given

11. Get the list of pods availble in the cluster, sort them by name and write in a file

12. Create a NodePort svc for the existing pod

13. Scale the deployment to 3 replaces and record

14. Create a side car logging container inside the nginx container pod

15. Fix the given kubeconfig file

16. Create a custom scheduler and schedule a nginx pod using that scheduler in cp node.