* 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.
* info\_outlineSolution

Pods authenticate to the API Server using ServiceAccounts. If the serviceAccount name is not specified, the default service account for the namespace is used during a pod creation.  
  
Reference: https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/

Now, create a service account pvviewer:

kubectl create serviceaccount pvviewer

To create a clusterrole:

kubectl create clusterrole pvviewer-role --resource=persistentvolumes --verb=list

To create a clusterrolebinding:

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

Solution manifest file to create a new pod called pvviewer as follows:

---

apiVersion: v1

kind: Pod

metadata:

labels:

run: pvviewer

name: pvviewer

spec:

containers:

- image: redis

name: pvviewer

# Add service account name

serviceAccountName: pvviewer

* 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)

* info\_outlineSolution

Explore the jsonpath loop.  
kubectl get nodes -o jsonpath='{.items[\*].status.addresses[?(@.type=="InternalIP")].address}' > /root/CKA/node\_ips

* 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
* info\_outlineSolution

Solution manifest file to create a multi-container pod multi-pod as follows:

---

apiVersion: v1

kind: Pod

metadata:

name: multi-pod

spec:

containers:

- image: nginx

name: alpha

env:

- name: name

value: alpha

- image: busybox

name: beta

command: ["sleep", "4800"]

env:

- name: name

value: beta

fo\_outlineQuestion

Create a Pod called non-root-pod , image: redis:alpine  
  
runAsUser: 1000  
  
fsGroup: 2000

* info\_outlineSolution

Solution manifest file to create a pod called non-root-pod as follows:

---

apiVersion: v1

kind: Pod

metadata:

name: non-root-pod

spec:

securityContext:

runAsUser: 1000

fsGroup: 2000

containers:

- name: non-root-pod

image: redis:alpine

Verify the user and group IDs by using below command:

kubectl exec -it non-root-pod -- id

* info\_outlineQuestion
* 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.
* info\_outlineSolution

Solution manifest file to create a network policy ingress-to-nptest as follows:

---

apiVersion: networking.k8s.io/v1

kind: NetworkPolicy

metadata:

name: ingress-to-nptest

namespace: default

spec:

podSelector:

matchLabels:

run: np-test-1

policyTypes:

- Ingress

ingress:

- ports:

- protocol: TCP

port: 80

CheckCompleteIncomplete

* format\_list\_bulletedDetails

Q. 6

* info\_outlineQuestion

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

* info\_outlineSolution

To add taints on the node01 worker node:

kubectl taint node node01 env\_type=production:NoSchedule

Now, deploy dev-redis pod and to ensure that workloads are not scheduled to this node01 worker node.

kubectl run dev-redis --image=redis:alpine

To view the node name of recently deployed pod:

kubectl get pods -o wide

Solution manifest file to deploy new pod called prod-redis with toleration to be scheduled on node01 worker node.

---

apiVersion: v1

kind: Pod

metadata:

name: prod-redis

spec:

containers:

- name: prod-redis

image: redis:alpine

tolerations:

- effect: NoSchedule

key: env\_type

operator: Equal

value: production

To view only prod-redis pod with less details:

kubectl get pods -o wide | grep prod-redis

CheckCompleteIncomplete

* format\_list\_bulletedDetails

Q. 7

* info\_outlineQuestion

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.

* info\_outlineSolution

Create a namespace if it doesn't exist:

kubectl create namespace hr

and then create a hr-pod with given details:

kubectl run hr-pod --image=redis:alpine --namespace=hr --labels=environment=production,tier=frontend

CheckCompleteIncomplete

* format\_list\_bulletedDetails

Q. 8

* info\_outlineQuestion

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

* info\_outlineSolution

Verify host and port for kube-apiserver are correct.  
  
Open the super.kubeconfig in vi editor.  
  
Change the 9999 port to 6443 and run the below command to verify:

kubectl cluster-info --kubeconfig=/root/CKA/super.kubeconfig

CheckCompleteIncomplete

* format\_list\_bulletedDetails

Q. 9

* info\_outlineQuestion

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.

* info\_outlineSolution

Use the command kubectl scale to increase the replica count to 3.

kubectl scale deploy nginx-deploy --replicas=3

The controller-manager is responsible for scaling up pods of a replicaset. If you inspect the control plane components in the kube-system namespace, you will see that the controller-manager is not running.

kubectl get pods -n kube-system

The command running inside the controller-manager pod is incorrect.  
After fix all the values in the file and wait for controller-manager pod to restart.  
  
Alternatively, you can run sed command to change all values at once:

sed -i 's/kube-contro1ler-manager/kube-controller-manager/g' /etc/kubernetes/manifests/kube-controller-manager.yaml

This will fix the issues in controller-manager yaml file.  
  
At last, inspect the deployment by using below command:

kubectl get deploy

CheckCompleteIncomplete

* format\_list\_bulletedDetails

TEST AGAIN