**Beginning**

kubectl apply -f https://docs.projectcalico.org/v3.14/manifests/calico.yaml

Each question during this exam must be completed on a designated cluster/configuration context. To minimize switching, the questions are grouped so that all questions on a given cluster appear consecutively.

There are six clusters made up varying numbers of containers, as follows:

* k8s – 1 master, 2 worker
* hk8s – 1 master, 2 worker
* bk8s – 1 master, 1 worker
* wk8s – 1 master, 2 worker
* ek8s – 1 master, 2 worker
* ik8s – 1 master, 1 base node

At the start of the each question you’ll be provided with the command to ensure you are on the correct cluster e.g..

Set configuration context: $ kubectl config use-context k8s

nodes making up each cluster can be reached via ssh, using a command such as the following:

$ ssh k8s-node-0

Elevated privileges can be assumed on any node with the following command: $ sudo -i

When you have finished working on a node, you should return to the base node (with hostname node-1) before attempting any further questions. Nested ssh is not supported.

You can use kubectl and the appropriate context to work on any cluster from the base node. When connected to a cluster member via ssh, you will only be able to work on that particular cluster via kubectl.

**Q1**

Set configuration context: $ kubectl config use-context k8s

Monitor the logs of pod foo and:

* Extract the log lines corresponding to error unable-to-access-website
* Write them to /opt/KULM00201/foo

**ANS:** $ kubectl logs foo|grep “unable-to-access-website” > /opt/KULM00201/foo

*Use below for practice*

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

*mkdir /opt/KULM00201/*

*kubectl logs foo|grep start > /opt/KULM00201/foo*

**Q2**

Set configuration context: $ kubectl config use-context k8s

List all persistent volumes sorted by name, saving the full kubectl output to /opt/KUCC00102/pv\_list. Use kubectl’s own functionality for sorting the output, and do not manipulate it any further.

**ANS**: $ kubectl get pv --sort-by=.metadata.name > /opt/KUCC00102/pv\_list

**Q3**

Set configuration context: $ kubectl config use-context k8s

Ensure a single instance of pod nginx is running on each node of the Kubernetes cluster where nginx also represents the image name which has to be used. Do not override any taints currently in place

Use **DaemonSet** to complete this task and use ds-kusc00201 as DaemonSet name.

**ANS**

|  |  |
| --- | --- |
| 1. kubectl create deploy nginx --image=nginx --dry-run=client -o yaml > DeamonSet.yaml 2. Change the Kind to DeamonSet and remove replicas  |  | | --- | | apiVersion: apps/v1  kind: DaemonSet  metadata:  labels:  app: nginx  name: ds-kusc00201  spec:  selector:  matchLabels:  app: nginx  template:  metadata:  labels:  app: nginx  spec:  containers:  - image: nginx  name: nginx |  1. Kubectl create -f DeamonSet.yaml |

**Q4**

Set configuration context: $ kubectl config use-context k8s

Perform the following tasks

* Add an init container to lumpy-koala (which has been defined in spec file /opt/KUCC00108/pod-spec-KUCC00108.yaml)
* The init container should create an empty file named /workdir/eager.txt
* If /workdir/eager.txt is not detected, the pod should exit
* Once the spec file has been updated with the init container definition, the pod should be created

Ans

|  |
| --- |
| apiVersion: v1  kind: Pod  metadata:  name: lumpy-koala  labels:  name: lumpy-koala  spec:  containers:  - name: base-container  image: busybox  command: ['sh', '-c', 'if [ -f /workdir/eager.txt ]; then sleep 99999; else exit; fi']  volumeMounts:  - name: workdir-volume  mountPath: /workdir  initContainers:  - name: init-container  image: busybox:1.28  command: ['sh','-c','mkdir -p /workdir; touch /workdir/eager.txt']  volumeMounts:  - name: workdir-volume  mountPath: /workdir  volumes:  - name: workdir-volume  emptyDir: {} |

**Q5**

Set configuration context: $ kubectl config use-context k8s

Create a pod named kucc4 with a single app container for each of the following images running inside (there may be between 1 and 4 images specified): nginx + redis + memcached .

**ANS**

|  |  |
| --- | --- |
| 1. kubectl run kucc4 --image=nginx --dry-run=client -o yaml > multi\_cont.yaml 2. Add multiple containers by editing the yaml file  |  | | --- | | apiVersion: v1  kind: Pod  metadata:  labels:  name: kucc4  spec:  containers:  - image: nginx  name: nginx  - image: redis  name: redis  - image: memcached  name: memcached |  1. Create the pod using pods   $ kubectl create –f multi\_cont.yaml |

**Q6**

Set configuration context: $ kubectl config use-context k8s

Schedule a pod as follows:

* Name: nginx-kusc00101
* Image: nginx
* Node selector: disk=spinning

**ANS**

|  |  |
| --- | --- |
| 1. kubectl get nodes –show-labels (check if there are any nodes with disk=spinning label, if not perform step#2) 2. kubectl label node node01 disk=spinning 3. kubectl run nginx-kusc00101 --image=nginx --dry-run=client -o yaml > nodeSelector.yaml 4. add nodeselector to yaml  |  | | --- | | apiVersion: v1  kind: Pod  metadata:  labels:  run: nginx-kusc00101  name: nginx-kusc00101  spec:  containers:  - image: nginx  name: nginx-kusc00101  nodeSelector:  disk: spinning |  1. kubectl create -f nodeSelector.yaml |

**Q7**

Set configuration context: $ kubectl config use-context k8s

Create a deployment as follows:

* Name: nginx-app
* Using container nginx with version 1.10.2-alpine
* The deployment should contain 3 replicas

Next, deploy the application with new version un, by performing a rolling update, and record that update.

Finally, rollback that update to the previous version 1.10.2-alpine.

**ANS**

|  |
| --- |
| 1. kubectl create deployment nginx-app --image=nginx:1.10.2-alpine –save-config 2. kubectl scale deployment nginx-app –replicas=3 3. kubectl **set image** deployment nginx-app nginx=nginx:1.13.0-alpine -**-record** 4. kubectl rollout history deployment nginx-app 5. kubectl rollout **undo** deployment nginx-app |

**Q8**

Set configuration context: $ kubectl config use-context k8s

Create and configure the service front-end-service so it’s accessible through ClusterIP and routes to the existing pod named front-end.

Answer:

|  |
| --- |
| $ kubectl expose pod front-end --name=front-end-service –type=ClusterIP --port=80  *To validate use below commands:*  *kubectl get svc, ep*  *Verify the service using the below curl (ip address from above command)*  *curl http://* |

**Q9**

Set configuration context: $ kubectl config use-context k8s

Create a pod as follows:

* Name: nginx
* Using image: nginx
* In a new Kubernetes namespace named: website-frontend

**ANS**

|  |
| --- |
| 1. Kubectl get namespaces (to check if the namespace already present, if not run step#2) 2. Kubectl create namespace website-frontend 3. kubectl run nginx --image=nginx --namespace=website-frontend (Create PODs) 4. *Verification by namespace*   *kubectl get po -n website-frontend* |

**Q10**

Set configuration context: $ kubectl config use-context k8s

Create a deployment spec file that will:

* Launch 7 replicas of the redis image with the label app\_runtime\_stage=test
* Deployment name: kual00201

Save a copy of this spec file to /opt/KUAL00201/spec\_deploy.yaml (or .json)

When you are done, clean up (delete) any new Kubernetes API object that you produced during this task.

**Answer**

|  |  |
| --- | --- |
| 1. Check the destination directory create   mkdir /opt/KUAL00201/   1. Kubectl create deployment kual00201 –image=redis –dry-run=client -o yaml >> /opt/KUAL00201/spec\_deploy.yaml 2. Vi the yaml file in point#2 and change the labels under spec and Template. 3. Create yaml (refer kubernets.io)  |  | | --- | | apiVersion: apps/v1  kind: Deployment  metadata:  name: kual00201      spec:  replicas: 7  selector:  matchLabels:  app\_runtime\_stage: test  template:  metadata:  labels:  app\_runtime\_stage: test  spec:  containers:  - name: redis  image: redis |  1. $kubectl get deploymets. (**verify**) 2. $kubectl delete deployment kual00201 (**question is only to create file, hence make sure to delelete**) |

**Q11**

Set configuration context: $ kubectl config use-context k8s

Create a file: /opt/KUCC00302/kucc00302.txt that lists all pods that implement service foo in namespace production.

The format of the file should be one pod name per line.

**ANS**

**$ kubectl get svc -n production -o wide**

Example:

k get svc -o wide

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR

foo ClusterIP 10.101.93.135 <none> 80/TCP 8h run=foo-frontend

**$ k get pods -l run=foo-frontend -n production > /opt/KUCC00302/kucc00302.txt**

**Q12**

Set configuration context: $ kubectl config use-context k8s

**Q12.1** Create a Kubernetes secret as follows:

* Name: super-secret
* Username: **swamigal**

**Q12.2** Create a pod named pod-secrets-via-file, using the redis image, which mounts a secret named super-secret at /secrets.

**Q12.3** Create a second pod named pod-secrets-via-env, using the redis image, which exports username as **LAWRENCE**.

**ANS 12.1**

**kubectl create secret generic super-secret --from-literal username=swamigal**

To View Secret:

kubectl get secrets

**ANS 12.2**

|  |  |
| --- | --- |
| **kubectl run pod-secrets-via-file --image=redis --dry-run=client -o yaml > pod-secret-via-file.yaml**   |  | | --- | | apiVersion: v1  kind: Pod  metadata:  creationTimestamp: null  labels:  run: pod-secrets-via-file  name: pod-secrets-via-file  spec:  containers:  - image: redis  name: pod-secrets-via-file  volumeMounts:  - name: foo  mountPath: "/secrets"  readOnly: true  volumes:  - name: foo  secret:  secretName: super-secret |   kubectl describe pod pod-secrets-via-file (verification)  kubectl exec pod-secrets-via-file -it sh  cat /secrets/username |

ANS 12.3

|  |  |  |
| --- | --- | --- |
| k run pod-secrets-via-env --image=redis --dry-run=client -o yaml >pod-secrets-via-env.yaml   |  | | --- | | apiVersion: v1  kind: Pod  metadata:  creationTimestamp: null  labels:  run: pod-secrets-via-env  name: pod-secrets-via-env  spec:  containers:  - image: redis  name: pod-secrets-via-env  env:  - name: LAWRENCE  valueFrom:  secretKeyRef:  name: super-secret  key: usernam | | k describe pod pod-secrets-via-env  k exec pod-secrets-via-env -it sh  echo $LAWRENCE | |

**Q13**

Set configuration context: $ kubectl config use-context k8s

Create a pod as follows:

* Name: non-persistent-redis
* Container image: redis
* Persistent volume with name: app-cache
* Mount path: /data/redis

It should launch in the staging namespace and the volume **must not** be persistent.

Ans:

|  |
| --- |
| Copy yaml from below link  <https://kubernetes.io/docs/tasks/configure-pod-container/configure-volume-storage/>  and Alter to fullfil the requirement |

|  |
| --- |
| apiVersion: v1  kind: Pod  metadata:  name: non-persistent-redis  spec:  containers:  - name: redis  image: redis  volumeMounts:  - name: app-cache  mountPath: /data/redis  volumes:  - name: app-cache  emptyDir: {} |
| kubectl apply -f katrina.yaml **-n staging** |

**Q14**

Set configuration context: $ kubectl config use-context k8s

Scale the deployment guestbook to 3 pods.

ANS: k create deployment guestbook --image=mongo (only practice)

k scale deployment guestbook --replicas=3 (actual answer)

**Q15**

Set configuration context: $ kubectl config use-context k8s

Check to see how many nodes are ready (not including nodes tainted NoSchedule) and write the number to /opt/KUCC00104/kucc00104.txt

ANS:

**kubectl describe nodes |grep -i taints|grep -v NoSchedule|wc -l > /opt/KUCC00104/kucc00104.txt**

[Exam Tip] In exam if you forgot the command, write 2 as answer in the file /opt/KUCC00104/kucc00104.txt

**Q16**

Set configuration context: $ kubectl config use-context k8s

From the pod label **name=cpu-utilizer**, find pods running high CPU workloads and write the name of the pod consuming most CPU to the file /opt/KUTR00102.txt (which already exists)

Ans: $ k top pod l name=cpu-utilizer

kubectl run nginx-new --image=nginx -l name=cpu-utilizer --dry-run=client -o yaml>ngnix2.yaml

[RR] Not correct. The questions ask write the **name of the pod consuming MOST CPU**. Meaning we will need to get the one pod name that is consuming most memory out of all other pods. Few other scenarios given,

Few sample scenarios related to the Pod Memory and CPU consumption

—————————————————

From the pod label tier=control-plane in all the namespaces , find the pods consuming high CPU & write the name of the pod consuming MOST cpu

k top pod -A -l tier=control-plane --sort-by=cpu | head -2 | tail -1 | awk '{ print $2 }' > /tmp/most-cup-usage-pod-name.txt

—————————————————

From the pod label tier=control-plane in all the namespaces , find the pods consuming high CPU & write the name of the three pods consuming MOST cpu

k top pod -A -l tier=control-plane --sort-by=cpu | head -4 | tail -3 | awk '{ print $2 }' > /tmp/top3-cup-usage-pod-name.txt

—————————————————

From the pod label tier=control-plane in all the namespaces , find the Memory hunging Pods & write the name of the pod consuming MOST memory

k top pod -A -l tier=control-plane --sort-by=memory | head -2 | tail -1 | awk '{ print $2 }' > /tmp/most-memory-hunging-pod-name.txt

————————————————

From the pod label tier=control-plane in all the namespaces , find the Memory hunging Pods & write the name of the top three memory hunging pods

k top pod -A -l tier=control-plane --sort-by=memory | head -4 | tail -3 | awk '{ print $2 }' > /tmp/top3-memory-hunging-pod-name.txt

—————————————————

From the pod label tier=contsrol-plane in all the namespaces , find the top three Memory hunging Pods & write them to file

k top pod -A -l tier=control-plane --sort-by=memory | head -4 > /tmp/top3-memory-hunging-pods.txt

—————————————————

**Q17**

Set configuration context: $ kubectl config use-context k8s

Create a deployment as follow:

* Name: nginx-test
* Exposed via a service nginx-test
* Ensure that the service & pod are accessible via their respective DNS records
* The container(s) within any pod(s) running as a part of this deployment should use the **nginx** image

Next, use the utility nslookup to look up the DNS records of the service & pod and write the output to /opt/KUNW00601/service.dns and /opt/KUNW00601/pod.dns respectively.

**ANS**

|  |
| --- |
| kubectl create deployment nginx-test --image=nginx  kubectl expose deployment.apps/nginx-test --port=80  create a pod (dnsutils) to use nslookup functionality (refere below link for yaml)  <https://kubernetes.io/docs/tasks/administer-cluster/dns-debugging-resolution/>  k exec -it dnsutils -- nslookup nginx-test > /opt/KUNW00601/service.dns  k exec -it dnsutils -- nslookup <pod ip>.default.pod.cluster.local  example : k exec -it dnsutils -- nslookup 10-244-1-6.default.pod.cluster.local  *Note: Default in above is namespace* |

**Q18**

No configuration context change required for this item

Create a snapshot of the etcd instance running at <https://127.0.0.1:2379>, saving the snapshot to the file path /var/lib/backup/etcd-snapshot.db.

The etcd instance is running etcd version 3.3.15.

The following TLS certificate/key are supplied for connecting to the server with etcdct1:

* CA certificate: /opt/KUCM00302/ca.crt
* Client certificate: /opt/KUCM00302/etcd-client.crt
* Client key: /opt/KUCM00302/etcd-client.key

ANS

ETCDCTL\_API=3 etcdctl --endpoints=<https://127.0.0.1:2379> *--cacert=*/opt/KUCM00302/ca.crt

*--cert=*/opt/KUCM00302/etcd-client.crt

*--key=*/opt/KUCM00302/etcd-client.key snapshot save /var/lib/backup/etcd-snapshot.db

**Q19**

Set configuration context: $ kubectl config use-context ek8s

Set the node named ek8s-node-0 as unavailable and reschedule all the pods running on it.

**ANS:**

**Kubectl drain node ek8s-node-0 --ignore-daemonsets**

[Exam Tip] In exam, it might error to ask to use remove local data flag. Please be careful about that. You ensure that you are on the right cluster before attempt this question.

**Q20**

Set configuration context: $ kubectl config use-context wk8s

A kubernetes worker node, named wk8s-node-0 is in state **NotReady**. You are in the L2 Team. The node is generating customer advices. Major incident already kicked off and behind you, your manager is standing and on the phone the incident manager is asking for an ETA ☺ .

Investigate why this is the case, and perform any appropriate steps to bring the node to a Ready state, ensuring that any changes are made permanent.

You ask one of the L1 team member to do this while you are investigating

* ssh to the failed node using: $ ssh wk8s-node-0
* L1 team member assume elevated privileges on the node with the following command: $ sudo -i

ANS

|  |
| --- |
| 1. ssh to node that is NoReady (wk8s-node-1) 2. systemctl status kubelet (check status of kubelet service) 3. ps -aux|grep kubelet (check the status of kubelet process) 4. systemctl start kubelet (if it is not active/not running – start it) 5. systemctl daemon-reload (refresh the status) 6. systemctl enable kubelet. (enable service so that service will start automatically after restart) 7. systemctl status kubelet (check if the service running after restart) 8. Exit from worker node *(Make sure exist only once and check which node after exit)* 9. check if the node status is back to Ready   kubectl get nodes |

**Q21**

Set configuration context: $ kubectl config use-context wk8s

You are working for an event management company. The company is expected to submit a RFP to get a big contract from Hollywood. Your company Sales Manager suddenly asked the IT head that he need a jhakastool webapp that will show Angelina Jolie performing a Turkish belly dance. The IT head came to your desk and said ‘I want to see a pod and the name of the pod is jhakastool’ in the next 1 hour.

Configure the kubelet systemd -managed service, on the node labelled with name=wk8s-node-1, to launch a pod containing a single container of image httpd named jhakastool automatically. Any spec files required should be placed in the /etc/kubernetes/mainfests directory on the node.

You turned to your best colleague/friend to help you and asked your colleague to perform

* ssh to the appropriate node using: $ ssh wk8s-node-1
* you already got the elevated privileges on the node with the following command $ sudo -i

**Answer**

|  |
| --- |
| Create the StaticPod to fulfil the requirement  1. create yaml file in master node with imperative command to create pod requested in question (jhakastool’)  Kubectl run jhakastool –image=nginx –dry-run=client -o yaml > staticpod.yaml  2. remove unwanted tags and copy the yaml conent to nodepad  3. ssh to node wk8s-node-1 (worker node/labeled node as per question)  4. cd /etc/kubernetes/mainfests (worker node)  (*Use copy the contents from (step2))*  5. create yaml file by pasting the pod definition from notepad (step 2)  exit from worker node  6. verify in master node if the static pod created  Kubectl get pods (staticpod) |

**Q22**

No configuration context change required for this item

**Important:** For this item, you will have to ssh to the nodes ik8s-master-0 and ik8s-node-0 and complete all tasks on these nodes. Ensure that you return to the base node (hostname: node-1) when you have completed this item.

As an admin of a small development team, you have been asked to setup a kubernetes cluster to test the feasibility of a new application.

You must use kubeadm to perform this task. Any kubeadm invocations will require the use of the –ignore-preflight-errors=all option.

* Configure the node ik8s-master-0 as a master node.
* Join the node ik8s-node-0 to the cluster.

Docker is already installed on both nodes and apt has been configured so that you can install the required tools.

You must use the kubeadm configuration file located at /etc/kubeadmn.conf when initializing your cluster.

The cluster will be considered complete once both nodes are in a Ready state.

**Answer**

|  |  |
| --- | --- |
| **Master Node Configuration**   |  | | --- | | **1. ssh to master node** ik8s-master-0 **(as per question)**  **2. install kubeadm using below link (installing kubeadm-use bookmarks)**  <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/>  **3. configure cluster (creating cluster with kubeadm-use bookmarks)**  <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/>  **4. Copy the kubeadm join command to nodepad (required for worker to join to cluster)**  **5. configure networks (quickly copy wavenet or calico n/w definition and apply)**  <https://v1-17.docs.kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/#tabs-pod-install-5> | |
| **Worker Node Configuration**   |  | | --- | | **1. ssh to worker node** ik8s-master-0 **(as per question)**  **2. install kubeadm using below link (installing kubeadm-use bookmarks)**  <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/>  3. Join the cluster using command from Master configuration (master config step 4) | |
| **Check the master node if able to get nodes details ik8s-master-0**  **Kubectl get nodes** |

**Q23**

Set configuration context: $kubectl config use-context bk8s

Given a partially-functioning kubernetes cluster, can you identify symptoms of failure on the cluster.

You should determine the node, the failing service and take appropriate actions to bring up the failed services and restore the health of the cluster. You have to ensure that any changes are made permanently.

Clues:

* You may ssh to the relevant nodes using $ ssh ${NODE} where ${NODE} is one of bk8s-master-0 or bk8s-node-0
* Assume elevated privileges on any node in the cluster with the following command $ sudo -i

**ANS**

|  |  |  |
| --- | --- | --- |
| |  | | --- | | 1. Kubectl get cs (check status of components) : if fails, system pods are not running  2. check if the yaml present in the default staticpod path / **/etc/kubernetes/manifests**  3. check if the staticPod configuration is correct in file /var/lib/kubelet/config.yaml  If staticPod entry incorrect/missing, correct accordingly, sample provided below | | *More details, refer Ravi CKA Preparation document* | |

**Q24**

Set configuration context: $kubectl config use-context hk8s

Create a persistent volume with name trade-data, of capacity 2Gi and access mode ReadWriteOnce. The type of volume is hostPath and its location is /srv/trade-data.

Ans: As there is no imperative statement for this question, we must do this declaratively.

1. Browse to this link <https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/>.
2. Copy the pods/storage/pv-volume.yaml
3. Change the size, name and volume path
4. Open VI in the exam terminal and paste the yaml
5. Kubectl apply -f filename.yaml

|  |
| --- |
| apiVersion: v1  kind: PersistentVolume  metadata:  name: trade-data  labels:  type: local  spec:  storageClassName: manual  capacity:  storage: 2Gi  accessModes:  - ReadWriteOnce  hostPath:  path: "/srv/trade-data" |

**Q25**

Set configuration context: $ kubectl config use-context k8s

Monitor the logs of pod foo and:

* Extract the log lines corresponding to error file-not-found
* Write them to /opt/KULM00201/foo

**ANS**

|  |
| --- |
| **kubectl logs foo | grep “file-not-found” > /opt/ KULM00201/foo** |

**Q26**

Set configuration context: $ kubectl config use-context k8s

List all persistent volumes sorted by **capacity**, saving the full kubectl output to /opt/KUCC00102/pv\_list. Use kubectl’s own functionality for sorting the output, and do not manipulate it any further.

**ANS**

|  |
| --- |
| **kubectl get pv --sort-by=.spec.capacity.storage > /opt/KUCC001/pv\_list** |

**Q27**

Set configuration context: $ kubectl config use-context k8s

Create and configure the service front-end-service so it’s accessible through NodePort and routes to the existing pod named front-end.

**ANS**

|  |
| --- |
| **kubectl expose pod front-end --name=front-end-service-np --type=NodePort --port=80** |

[ExamTip] perfect but ensure that there is no specific node port given in the question. If the specific node port given then you will need to dry run as yaml and add the specific node port to override the random assigned nodeport

**Q28**

Set configuration context: $ kubectl config use-context k8s

Create a pod as follows:

* Name: mongo
* Using image: mongo
* In a new Kubernetes namespace named: website-backend

**ANS**

|  |
| --- |
| **kubectl create ns website-backend**  **kubectl run mongo --image= mongo --restart=Never --namespace=website-backend**  To verify: k get pods -n website-backend |

**Q29**

Set configuration context: $ kubectl config use-context k8s

Create a deployment spec file that will:

* Launch 3 replicas of the nginx image with the label app\_runtime\_stage=dev
* Deployment name: kual00201

Save a copy of this spec file to /opt/KUAL00201/deployment\_spec.yaml (or .json)

When you are done, clean up (delete) any new Kubernetes API object that you produced during this task.

**Answer**

|  |  |
| --- | --- |
| 1. Check the destination directory create   mkdir /opt/KUAL00201/   1. Kubectl create deployment kual00201 –image=nginx –dry-run=client -o yaml >> /opt/KUAL00201/deployment\_spec.yaml 2. Vi the yaml file in point#2 and change the labels under spec and Template. 3. Create yaml (refer kubernets.io)  |  | | --- | | apiVersion: apps/v1  kind: Deployment  metadata:  name: kual00201  spec:  replicas: 3  selector:  matchLabels:  app\_runtime\_stage: dev  template:  metadata:  labels:  app\_runtime\_stage: dev  spec:  containers:  - name: ngnix  image: ngnix |  1. $kubectl get deploymets 2. $kubectl delete deployment kual00201 *(Make sure* ***delete*** *deployment a question doesn’t need the deployment executed)* |

**Q30**

Set configuration context: $ kubectl config use-context k8s

Create a file: /opt/KUCC00302/kucc00302.txt that lists all pods that implement service baz in namespace development.

The format of the file should be one pod name per line.

**ANS:**

|  |
| --- |
| **$ kubectl get svc -n development -o wide**  Example:  k get svc -o wide  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR  baz ClusterIP 10.101.93.135 <none> 80/TCP 8h run=baz-frontend  **$ k get pods -l run=baz-frontend -n production > /opt/KUCC00302/kucc00302.txt** |

**Q31**

Set configuration context: $ kubectl config use-context k8s

Q31.1 Create a Kubernetes secret as follows:

* Name: super-secret
* credential: **jhunhunwala**

Q31.2 Create a pod named pod-secrets-via-file, using the redis image, which mounts a secret named super-secret at /secrets.

Q31.3 Create a second pod named pod-secrets-via-env, using the redis image, which exports credential as **PRADIPTA**.

**ANS**

Ans:31.1

Ref: <https://kubernetes.io/docs/tasks/inject-data-application/distribute-credentials-secure/#define-container-environment-variables-using-secret-data>

|  |
| --- |
| **kubectl create secret generic super-secret --from-literal=credential=jhunhunwala**  **kubectl get secret** |

Ans:31.2

|  |  |
| --- | --- |
| Ref the link below and take the yaml temple and create pod: pod-secrets-via-file  <https://kubernetes.io/docs/tasks/inject-data-application/distribute-credentials-secure/#define-container-environment-variables-using-secret-data>   |  | | --- | | apiVersion: v1  kind: Pod  metadata:  name: pod-secrets-via-file  spec:  containers:  - name: redis  image: redis  volumeMounts:  # name must match the volume name below  - name: secret-volume  mountPath: /secrete/  # The secret data is exposed to Containers in the Pod through a Volume.  volumes:  - name: secret-volume  secret:  secretName: super-secret | |
| Verification  kubectl exec pod-secrets-via-file -it sh  cd /secrete/  cat credential |

|  |  |
| --- | --- |
| Ref the link below and take the yaml temple and create pod: pod-secrets-via-env  <https://kubernetes.io/docs/tasks/inject-data-application/distribute-credentials-secure/#define-container-environment-variables-using-secret-data>   |  | | --- | | apiVersion: v1  kind: Pod  metadata:  name: pod-secrets-via-env  spec:  containers:  - name: pod-secrets-via-env  image: redis  env:  - name: PRADIPTA  valueFrom:  secretKeyRef:  name: super-secret  key: credential | |
| Verification  k exec pod-secrets-via-env2 -it -- /bin/sh -c 'echo $PRABHURAJ'  OR  kubectl exec pod-secrets-via-env2 -it sh  ECHO $PRADIPTA |

**Q32**

Set configuration context: $ kubectl config use-context k8s

From the pod label **name=overload-cpu**, find pods running high CPU workloads and write the name of the pod consuming most CPU to the file /opt/KUTR00102.txt (which already exists)

kubectl top pod -| name=overload-cpu - - sort-by=cpu > /opt/KUTR00102.txt

kubectl top pod -| name=overload-cpu - - sort-by=cpu|head -n2|grep -v NAME > /opt/KUTR00102.txt

or

Ans: $ k top pod l name=overload-cpu

[RR] Not correct. The questions ask write the **name of the pod consuming MOST CPU** . Meaning we will need to get the one pod name that is consuming mot memory out of all other pods. Few other scenarios given,

Few sample scenarios related to the Pod Memory and CPU consumption

—————————————————

From the pod label tier=control-plane in all the namespaces , find the pods consuming high CPU & write the name of the pod consuming MOST cpu

k top pod -A -l tier=control-plane --sort-by=cpu | head -2 | tail -1 | awk '{ print $2 }' > /tmp/most-cup-usage-pod-name.txt

—————————————————

From the pod label tier=control-plane in all the namespaces , find the pods consuming high CPU & write the name of the three pods consuming MOST cpu

k top pod -A -l tier=control-plane --sort-by=cpu | head -4 | tail -3 | awk '{ print $2 }' > /tmp/top3-cup-usage-pod-name.txt

—————————————————

From the pod label tier=control-plane in all the namespaces , find the Memory hunging Pods & write the name of the pod consuming MOST memory

k top pod -A -l tier=control-plane --sort-by=memory | head -2 | tail -1 | awk '{ print $2 }' > /tmp/most-memory-hunging-pod-name.txt

————————————————

From the pod label tier=control-plane in all the namespaces , find the Memory hunging Pods & write the name of the top three memory hunging pods

k top pod -A -l tier=control-plane --sort-by=memory | head -4 | tail -3 | awk '{ print $2 }' > /tmp/top3-memory-hunging-pod-name.txt

—————————————————

From the pod label tier=control-plane in all the namespaces , find the top three Memory hunging Pods & write them to file

k top pod -A -l tier=control-plane --sort-by=memory | head -4 > /tmp/top3-memory-hunging-pods.txt

**Q33**

Set configuration context: $ kubectl config use-context k8s

Create a deployment as follow:

* Name: nginx-dns
* Exposed via a service: nginx-dns
* Ensure that the service & pod are accessible via their respective DNS records
* The container(s) within any pod(s) running as a part of this deployment should use the **nginx** image

k create deployment nginx-dns --image=nginx

k expose deployment.apps/nginx-dns--port=80

create a pod to use busybox functionality

<https://kubernetes.io/docs/tasks/administer-cluster/dns-debugging-resolution/>

k exec -it busybox -- nslookup nginx-test > /opt/KUNW00601/service.dns

k exec -it busybox -- nslookup <pod ip>.default.pod.cluster.local

example : k exec -it busybox -- nslookup 10-244-1-6.default.pod.cluster.local

[RR] The pod dns should fetch using **k exec -it** busybox **-- nslookup 10-244-1-6.default.pod**

Where **default** is the namespace of the pod.

Next, use the utility nslookup to look up the DNS records of the service & pod and write the output to /opt/KUNW00601/service.dns and /opt/KUNW00601/pod.dns respectively. Ensure you use the busybox: 1.28 image (or earlier) for any testing, as the latest release has an upstream bug which impacts the use of nslookup.

**Q34**

Set configuration context: $ kubectl config use-context wk8s

Configure the kubelet systemd -managed service, on the node labelled with name=wk8s-node-1, to launch a pod containing a single container of image jenkins named jhakastool automatically. Any spec files required should be placed in the /etc/kubernetes/mainfests directory on the node.

You turned to your best colleague/friend to help you and asked your colleague to perform

* ssh to the appropriate node using: $ ssh wk8s-node-1
* you already got the elevated privileges on the node with the following command $ sudo -i

**ANS**

|  |
| --- |
| Create the StaticPod to fulfil the requirement  1. create yaml file in master node with imperative command to create pod requested in question (jhakastool’)  Kubectl run jhakastool –image=nginx –dry-run=client -o yaml > staticpod.yaml  2. remove unwanted tags and copy the yaml conent to nodepad  3. ssh to node wk8s-node-1 (worker node/labeled node as per question)  4. cd /etc/kubernetes/mainfests (worker node)  (*Use copy the contents from (step2))*  5. create yaml file by pasting the pod definition from notepad (step 2)  exit from worker node  6. verify in master node if the static pod created  Kubectl get pods (staticpod) |

**Q35**

Set configuration context: $kubectl config use-context bk8s

Given a partially-functioning kubernetes cluster, can you identify symptoms of failure on the cluster.

You should determine the node, the failing service and take appropriate actions to bring up the failed services and restore the health of the cluster. You have to ensure that any changes are made permanently.

Clues:

* You may ssh to the relevant nodes using $ ssh ${NODE} where ${NODE} is one of bk8s-master-0 or bk8s-node-0
* Assume elevated privileges on any node in the cluster with the following command $ sudo -i

**ANS**

**Refer: Q23**

**Q36**

Set configuration context: $kubectl config use-context hk8s

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: As there is no imperative statement for this question, we must do this declaratively.

1. Browse to this link <https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/>.
2. Copy the pods/storage/pv-volume.yaml
3. Change the size, name and volume path
4. Open VI in the exam terminal and paste the yaml
5. Kubectl apply -f filename.yaml

|  |
| --- |
| apiVersion: v1  kind: PersistentVolume  metadata:  name: app-data  labels:  type: local  spec:  storageClassName: manual  capacity:  storage: 1Gi  accessModes:  - ReadWriteMany  hostPath:  path: "/srv/app-data " |