LinuxFoundation.CKA

Exam Code:	CKA
Exam Name:	Certified Kubernetes Administrator (CKA) Program Exam
Certification Provider:	Linux Foundation

NEW QUESTION: 1
Set CPU and memory requests and limits for existing pod name
"nginx-prod".
Set requests for CPU and Memory as 100m and 256Mi respectively
Set limits for CPU and Memory as 200m and 512Mi respectively
A. kubectl get po
kubectl set resources po nginx-prod --
limits=cpu=200m,memory=512Mi --requests=cpu=100m,memory=256Mi
//Verify
kubectl top po
kubectl describe po nginx-prod
B. kubectl get po
kubectl set resources po nginx-prod --
limits=cpu=200m,memory=512Mi --requests=cpu=100m,memory=256Mi
//Verify
kubectl describe po nginx-prod
Answer: A

NEW QUESTION: 2
Deploy a pod with image=redis on a node with label disktype=ssd
A. // Get list of nodes
kubectl get nodes
//Get node with the label disktype=ssd
kubectl get no -l disktype=ssd
// Create a sample yaml file
kubectl run node-redis --generator=run-pod/v1 --image=redis --dry
run -o yaml > test-redis.yaml
// Edit test-redis.yaml file and add nodeSelector
vim test-redis.yaml
apiVersion: v1

- name: node-redis
image: redis
imagePullPolicy: IfNotPresent
kubectl apply -f test-redis.yaml
/ // Verify
K kubectl get po -o wide
B. // Get list of nodes
kubectl get nodes
//Get node with the label disktype=ssd
kubectl get no -l disktype=ssd
// Create a sample yaml file
kubectl run node-redis --generator=run-pod/v1 --image=redis --dry
run -o yaml > test-redis.yaml
// Edit test-redis.yaml file and add nodeSelector
vim test-redis.yaml
apiVersion: v1
kind: Pod
metadata:
name: redis
spec:
nodeSelector:
disktype: ssd
containers:
- name: node-redis
image: redis
imagePullPolicy: IfNotPresent
kubectl apply -f test-redis.yaml
/ // Verify
K kubectl get po -o wide
Answer: B

NEW QUESTION: 3
Update the deployment with the image version 1.16.1 and verify the image and check the rollout history
Answer:
kubectl set image deploy/webapp nginx=nginx:1.16.1 kubectl describe deploy webapp | grep Image kubectl rollout history deploy webapp

NEW QUESTION: 4
Get all the pods with label "env"
Answer:

kubectl get pods -L env

NEW QUESTION: 5

Scale the deployment from 5 replicas to 20 replicas and verify

Answer:

kubectl scale deploy webapp --replicas=20 kubectl get deploy webapp kubectl get po -l app=webapp

NEW QUESTION: 6

Get list of PVs and order by size and write to file - /opt/pvlist.txt

Answer:

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

NEW QUESTION: 7

Create a job named "hello-job" with the image busybox which echos "Hello I'm running job"

A. kubectl create job hello-job --image=busybox --dry-run -o yaml
-- echo "Hello I'm running job" > hello-job.yaml
kubectl create -f hello-job.yaml
//Verify Job
kubectl get po
kubectl logs hello-job-*

B. kubectl create job hello-job --image=busybox --dry-run -o yaml
-- echo "Hello I'm running job" > hello-job.yaml
kubectl create -f hello-job.yaml
//Verify Job
kubectl get job
kubectl get po
kubectl logs hello-job-*

Answer: B

NEW QUESTION: 8

Print pod name and start time to "/opt/pod-status" file

Answer:

kubect1 get pods -o=jsonpath='{range .items[*]}{.metadata.name}{"\t"}{.status.podIP}{"\n"}{end}'

NEW QUESTION: 9

Annotate the pod with name=webapp

A. kubectl annotate pod nginx-dev-pod name=webapp
kubectl annotate pod nginx-prod-pod name=webapp
// Verify

```
kubectl describe po nginx-dev-pod | grep -i annotations
kubectl describe po nginx-prod-pod | grep -i annotations
```
B. kubectl annotate pod nginx-dev-pod name=webapp
```
kubectl annotate pod nginx-prod-pod name=webapp
// Verify
kubectl describe po nginx-dev-pod | grep -i annotations
```
Answer: A

NEW QUESTION: 10
List the nginx pod with custom columns POD_NAME and POD_STATUS
Answer:
```
kubectl get po -o=custom-columns="POD_NAME:.metadata.name,
POD_STATUS:.status.containerStatuses[].state"
```

NEW QUESTION: 11
Label a node as app=test and verify
Answer:
```
kubectl label node node-name app=test // Verify kubectl get no -show-labels kubectl get no
-l app=test
```

NEW QUESTION: 12
Create a Pod nginx and specify a CPU request and a CPU limit of 0.5 and 1 respectively.
A. // create a yml file
```
kubectl run nginx-pod --image=nginx --restart=Never --dry-run -
o yaml > nginx-pod.yml
// add the resources section and create
vim nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
resources:
requests:
cpu: "0.5"
limits:
```

cpu: "1"
restartPolicy: Always
kubectl apply -f nginx-pod.yaml
// verify
kubectl top pod
B. // create a yml file
kubectl run nginx-pod --image=nginx --restart=Never --dry-run -
o yaml > nginx-pod.yml
// add the resources section and create
vim nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
resources:
requests:
cpu: "0.4"
limits:
cpu: "1"
restartPolicy: Always
kubectl apply -f nginx-pod.yaml
// verify
kubectl top pod
Answer: A

NEW QUESTION: 13
Create the service as type NodePort with the port 32767 for the nginx pod with the pod selector app: my-nginx
Answer:
kubectl run nginx --image=nginx --restart=Never -- labels=app=nginx --port=80 --dry-run -o yaml > nginx-pod.yaml

NEW QUESTION: 14
List all the pods that are serviced by the service "webservice" and copy the output in /opt/ $USER/webservice.targets Note: You need to list the endpoints

Answer:
kubectl descrive svc webservice | grep -i "Endpoints" > /opt/$USER/webservice.targets
kubectl get endpoints webservice > /opt/$USER/webservice.targets

NEW QUESTION: 15
Evict all existing pods from a node-1 and make the node unschedulable for new pods.
A. kubectl get nodes
kubectl drain node-1 #It will evict pods running on node-1 to
other nodes in the cluster
kubectl cordon node-1 # New pods cannot be scheduled to the
node
// Verify
kubectl get no
When you cordon a node, the status shows "SchedulingDisabled"
B. kubectl get nodes
kubectl drain node-1 #It will evict pods running on node-1 to
other nodes in the cluster
// Verify
kubectl get no
When you cordon a node, the status shows "SchedulingDisabled"
Answer: A

NEW QUESTION: 16
List all configmap and secrets in the cluster in all namespace and write it to a
file /opt/configmap-secret
Answer:
kubectl get configmap,secrets --all-namespaces > /opt/configmap-secret // Verify
Cat /opt/configmap-secret

NEW QUESTION: 17
Check the Image version of nginx-dev pod using jsonpath
Answer: A

kubect1 get po nginx-dev -o jsonpath='{.spec.containers[].image}{"\n"}'

NEW QUESTION: 18

Create a redis pod, and have it use a non-persistent storage

Note: In exam, you will have access to kubernetes.io site,

Refer : https://kubernetes.io/docs/tasks/configure-pod-container/configurevolume-storage/

A. apiVersion: v1
kind: Pod
metadata:
name: redis
spec:
containers:
- name: redis
image: redis
volumeMounts:
- containerPort: 6379
volumes:
- name: redis-storage
emptyDir: {}

B. apiVersion: v1
kind: Pod
metadata:
name: redis
spec:
containers:
- name: redis
image: redis
volumeMounts:
- name: redis-storage
mountPath: /data/redis
ports:
- containerPort: 6379
volumes:
- name: redis-storage
emptyDir: {}

Answer: B

NEW QUESTION: 19

List pod logs named "frontend" and search for the pattern "started" and write it to a file "/opt/error-logs"

Answer:

Kubectl logs frontend | grep -i "started" > /opt/error-logs

NEW QUESTION: 20

Create a Pod with three busy box containers with commands "ls; sleep 3600;", "echo Hello World; sleep 3600;" and "echo this is the third container; sleep 3600" respectively and check the status

A. // first create single container pod with dry run flag
kubectl run busybox --image=busybox --restart=Always --dry-run
-o yaml -- bin/sh -c "sleep 3600; ls" > multi-container.yaml
// edit the pod to following yaml and create it
apiVersion: v1
kind: Pod
metadata:
labels:
run: busybox
name: busybox
spec:
containers:
- args:
- bin/sh
- -c
- ls; sleep 3600
image: busybox
name: busybox-container-1
- args:
- bin/sh
- -c
- echo Hello world; sleep 3600
image: busybox
name: busybox-container-2
- args:
- bin/sh
- -c
- echo this is third container; sleep 3600
image: busybox
name: busybox-container-3
restartPolicy: Always
// Verify
Kubectl get pods

B. // first create single container pod with dry run flag
kubectl run busybox --image=busybox --restart=Always --dry-run

-o yaml -- bin/sh -c "sleep 3600; ls" > multi-container.yaml
// edit the pod to following yaml and create it
apiVersion: v1
kind: Pod
metadata:
labels:
run: busybox
name: busybox
spec:
containers:
- args:
- bin/sh
- -c
- ls; sleep 3600
- echo Hello world; sleep 3600
image: busybox
name: busybox-container-2
- args:
- bin/sh
- -c
- echo this is third container; sleep 3600
image: busybox
name: busybox-container-3
restartPolicy: Always
// Verify
Kubectl get pods
Answer: A

NEW QUESTION: 21
Create an nginx pod with container Port 80 and it should only receive traffic only it checks the endpoint / on port 80 and verify and delete the pod.
A. kubectl run nginx --image=nginx --restart=Never --port=80 --
dry-run -o yaml > nginx-pod.yaml
// add the readinessProbe section and create
vim nginx-pod.yaml
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx

ports:
- containerPort: 60
readinessProbe:
httpGet:
path: /
port: 60
restartPolicy: Never
kubectl apply -f nginx-pod.yaml
// verify
kubectl describe pod nginx | grep -i readiness
kubectl delete po nginx
B. kubectl run nginx --image=nginx --restart=Never --port=80 --
dry-run -o yaml > nginx-pod.yaml
// add the readinessProbe section and create
vim nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
ports:
- containerPort: 80
readinessProbe:
httpGet:
path: /
port: 80
restartPolicy: Never
kubectl apply -f nginx-pod.yaml
// verify
kubectl describe pod nginx | grep -i readiness
kubectl delete po nginx
Answer: B

NEW QUESTION: 22
Check nodes which are ready and print it to a file /opt/nodestatus
A. JSONPATH='{range .items[*]}{@.metadata.name}:{range

```
@.status.conditions[*]}{@.type}={@.status};{end}{end}' \
//Verify
cat /opt/node-status
```

B. JSONPATH='{range .items[*]}{@.metadata.name}:{range

```
@.status.conditions[*]}{@.type}={@.status};{end}{end}' \
&& kubectl get nodes -o jsonpath="$JSONPATH" | grep
"Ready=True" > /opt/node-status
//Verify
cat /opt/node-status
```

Answer: B

NEW QUESTION: 23

Create a file called "config.txt" with two values key1=value1
and key2=value2. Then create a configmap named "keyvalcfgmap" andread data from the file "config.txt" and verify that configmap is created correctly

A. cat >> config.txt << EOF

```
key1=value1
key2=value2
EOF
cat config.txt
// Create configmap from "config.txt" file
kubectl create cm keyvalcfgmap --from-file=config.txt
//Verify
kubectl get cm keyvalcfgmap -o yaml
```

B. cat >> config.txt << EOF

```
key1=value1
key2=value2
EOF
kubectl create cm keyvalcfgmap --from-file=config.txt
//Verify
kubectl get cm keyvalcfgmap -o yaml
```

Answer: A

NEW QUESTION: 24

Resume the rollout of the deployment

Answer:

```
kubectl rollout resume deploy webapp
```

NEW QUESTION: 25

Create a secret mysecret with values user=myuser and password=mypassword

A. kubectl create secret generic my-secret --fromliteral=username=user --from-literal=password=mypassword
// Verify
kubectl get secret --all-namespaces
kubectl get secret generic my-secret -o yaml
B. kubectl create secret generic my-secret --fromliteral=username=user --from-literal=password=mypassword
// Verify
kubectl get secret generic my-secret -o yaml
Answer: A

NEW QUESTION: 26
Create a busybox pod which executes this command sleep 3600 with the service account admin and verify
A. kubectl run busybox --image=busybox --restart=Always --dry-run
-o yaml -- /bin/sh -c "sleep 3600" > busybox.yml
// Edit busybox.yaml file
apiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
labels:
run: busybox
name: busybox
spec:
serviceAccountName: admin
containers:
- args:
- /bin/sh
- -c
- sleep 3600
image: busybox
name: busybox
restartPolicy: Always
// verify
K kubectl describe po busybox
B. kubectl run busybox --image=busybox --restart=Always --dry-run
-o yaml -- /bin/sh -c "sleep 3600" > busybox.yml
// Edit busybox.yaml file
apiVersion: v1
kind: Pod

metadata:
creationTimestamp: null
labels:
run: busybox
name: busybox
spec:
serviceAccountName: admin
containers:
- args:
- /bin/sh
- -c
- sleep 3800
image: busybox
name: busybox
restartPolicy: Always
// verify
K kubectl describe po busybox
Answer: A

NEW QUESTION: 27
Get the number of schedulable nodes and write to a file
/opt/schedulable-nodes.txt
A. kubectl get nodes -o jsonpath="{range
.items[*]}{.metadata.name}
{.spec.taints[?(@.effect=='NoSchedule')].effect}{\"\n\"}{end}"
| awk 'NF==1 {print $0}' > /opt/schedulable-nodes.txt
// Verify
cat /opt/schedulable-nodes.txt
B. kubectl get nodes -o jsonpath="{range
.items[*]}{.metadata.name}
{.spec.taints[?(@.effect=='NoSchedule')].effect}{\"\n\"}{end}"
| awk 'NF==11 {print $0}' > /opt/schedulable-nodes.txt
// Verify
cat /opt/schedulable-nodes.txt
Answer: A

NEW QUESTION: 28
Install a kubernetes cluster with one master and one worker using kubeadm
A. This is a straightforward question, you need to install kubernetes cluster using kubeadm with one master and one worker.
Refer : https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/

B. This is a straightforward question, you need to install kubernetes cluster using kubeadm with one master and one worker.
Installation is considered success once both master and worker
nodes become available.
Refer : https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/
Answer: B

NEW QUESTION: 29
Create an nginx pod and set an env value as 'var1=val1'. Check the env value existence within the pod
A.
```
kubectl run nginx --image=nginx --restart=Never --env=var1=val1
# then
kubectl exec -it nginx -- env
# or
kubectl run nginx --restart=Never --image=nginx --env=var1=val1
-it --rm -- env
```
B.
```
kubectl run nginx --image=nginx --restart=Never --env=var1=val1
# then
kubectl exec -it nginx -- env
# or
kubectl exec -it nginx -- sh -c 'echo $var1'
# or
kubectl describe po nginx | grep val1
# or
kubectl run nginx --restart=Never --image=nginx --env=var1=val1
-it --rm - env
```
Answer: B

NEW QUESTION: 30
Modify "hello-job" and make it run 10 times one after one and 5 times parallelism: 5
A.
```
kubectl create job hello-job --image=busybox --dry-run -o yaml
-- echo "Hello I am from job" > hello-job.yaml
// edit the yaml file to add completions: 10 and
kubectl create -f hello-job.yaml
```
YAML File:
```
apiVersion: batch/v1
kind: Job
metadata:
name: hello-job
spec:
completions: 10
```

parallelism: 5
template:
metadata:
spec:
containers:
- command:
- echo
- Hello I am from job
image: busybox
name: hello-job
restartPolicy: Never

B. kubectl create job hello-job --image=busybox --dry-run -o yaml
-- echo "Hello I am from job" > hello-job.yaml
// edit the yaml file to add completions: 16 and
kubectl create -f hello-job.yaml
YAML File:
apiVersion: batch/v1
kind: Job
metadata:
name: hello-job
spec:
completions: 16
parallelism: 5
template:
metadata:
spec:
containers:
- command:
- echo
- Hello I am from job
image: busybox
name: hello-job
restartPolicy: Never

Answer: A

NEW QUESTION: 31
Create a nginx pod that will be deployed to node with the label "gpu=true"

A. kubectl run nginx --image=nginx --restart=Always --dry-run -o yaml > nodeselector-pod.yaml
// add the nodeSelector like below and create the pod

```
kubectl apply -f nodeselector-pod.yaml
vim nodeselector-pod.yaml
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
nodeSelector:
gpu: true
containers:
- image: nginx
name: nginx
restartPolicy: Always
kubectl apply -f nodeselector-pod.yaml
//Verify
kubectl get no -show-labels
kubectl get po
kubectl describe po nginx | grep Node-Selectors
```

B. kubectl run nginx --image=nginx --restart=Always --dry-run -o
yaml > nodeselector-pod.yaml

```
// add the nodeSelector like below and create the pod
kubectl apply -f nodeselector-pod.yaml
vim nodeselector-pod.yaml
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
nodeSelector:
gpu: true
yaml
//Verify
kubectl get no -show-labels
kubectl get po
kubectl describe po nginx | grep Node-Selectors
```

Answer: A

NEW QUESTION: 32
Fix a node that shows as non-ready
A. Kubectl get nodes
// Check which node shows a not ready
kubectl describe nodes "node-name"
// Login to the node which shows as not ready and check the
systemctl start kubelet / docker
// Verify
ps -auxxww | grep -i "process-name"
kubectl get nodes
B. Kubectl get nodes
// Check which node shows a not ready
kubectl describe nodes "node-name"
// Login to the node which shows as not ready and check the
process for kubelet, docker , kube-proxy.
// systemctl status kubelet (or) ps -aux | grep -i "processname"
// If the process is not started, then start using
systemctl start kubelet / docker
// Verify
ps -auxxww | grep -i "process-name"
kubectl get nodes
Answer: B

NEW QUESTION: 33
Verify certificate expiry date for ca certificate in /etc/kubernetes/pki
Answer:
openssl x509 -in ca.crt -noout -text | grep -i validity -A 4

NEW QUESTION: 34
// Create a configmap
kubectl create configmap redis-config --from-file=/opt/redisconfig
// Verify
kubectl get configmap redis-config -o yaml
// first run this command to save the pod yml
kubectl run redis-pod --image=redis --restart=Always --dry-run
-o yaml > redis-pod.yml

// edit the yml to below file and create
apiVersion: v1
kind: Pod
metadata:
name: redis
spec:
containers:
- name: redis
image: redis
env:
- name: MASTER
value: "true"
ports:
- containerPort: 6379
volumeMounts:
- mountPath: /redis-master-data
name: data
- mountPath: /redis-master
name: config
volumes:
- name: data
emptyDir: {}
- name: config
configMap:
name: example-redis-config

A. items:
- key: redis-config
path: redis.conf
cf
// // Verify
K kubectl exec -it redis - cat /redis-master-data/redis.conf

B. items:
- key: redis-config
path: redis.conf
cf
kk kubectl apply -f redis-pod.yml
// // Verify
K kubectl exec -it redis - cat /redis-master-data/redis.conf

Answer: B

NEW QUESTION: 35

Pause the rollout of the deployment

Answer:

```
kubectl rollout pause deploy webapp
```

NEW QUESTION: 36

Create a deployment called webapp with image nginx having 5 replicas in it, put the file in /tmp directory with named webapp.yaml

A. //Create a file using dry run command

```
kubectl create deploy --image=nginx --dry-run -o yaml >
/tmp/webapp.yaml
```

// Now, edit file webapp.yaml and update replicas=5

```
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: webapp
name: webapp
spec:
replicas: 5
selector:
matchLabels:
app: webapp
template:
metadata:
labels:
app: webapp
spec:
containers:
- image: nginx
name: nginx
```

Note: Search "deployment" in kubernetes.io site , you will get
the page
https://kubernetes.io/docs/concepts/workloads/controllers/deplo
yment/

// Verify the Deployment

```
kubectl get deploy webapp --show-labels
```

// Output the YAML file of the deployment webapp

```
kubectl get deploy webapp -o yaml
```

B. //Create a file using dry run command

```
kubectl create deploy --image=nginx --dry-run -o yaml >
/tmp/webapp.yaml
```

// Now, edit file webapp.yaml and update replicas=5
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: webapp
name: webapp
spec:
replicas: 5
selector:
matchLabels:
app: webapp
template:
metadata:
labels:
Note: Search "deployment" in kubernetes.io site , you will get
the page
https://kubernetes.io/docs/concepts/workloads/controllers/deplo
yment/
// Verify the Deployment
kubectl get deploy webapp --show-labels
// Output the YAML file of the deployment webapp
kubectl get deploy webapp -o yaml
Answer: A

NEW QUESTION: 37
Create a namespace called 'development' and a pod with image nginx called nginx on this namespace.
Answer:
kubectl create namespace development kubectl run nginx --image=nginx --restart=Never -n development

NEW QUESTION: 38
Apply the autoscaling to this deployment with minimum 10 and maximum 20 replicas and target CPU of 85% and verify hpa is created and replicas are increased to 10 from 1
Answer:
kubectl autoscale deploy webapp --min=10 --max=20 --cpu percent=85 kubectl get hpa
kubectl get pod -l app=webapp

NEW QUESTION: 39

Create PersistentVolume named task-pv-volume with storage 10Gi, access modes ReadWriteMany, storageClassName manual, and volume at /mnt/data and Create a PersistentVolumeClaim of at least 3Gi storage and access mode ReadWriteOnce and verify

A. vim task-pv-volume.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
name: task-pv-volume
labels:
type: local
spec:
storageClassName: manual
capacity:
storage: 10Gi
accessModes:
- ReadWriteMany
hostPath:
path: "/mnt/data"
kubectl apply -f task-pv-volume.yaml
//Verify
kubectl get pv
vim task-pvc-volume.yaml
apiVersion: v1
- ReadWriteMany
resources:
requests:
storage: 3Gi
kubectl apply -f task-pvc-volume.yaml
//Verify
Kuk kubectl get pvc

B. vim task-pv-volume.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
name: task-pv-volume
labels:
type: local
spec:
storageClassName: manual
capacity:

storage: 10Gi
accessModes:
- ReadWriteMany
hostPath:
path: "/mnt/data"
kubectl apply -f task-pv-volume.yaml
//Verify
kubectl get pv
vim task-pvc-volume.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: task-pv-claim
spec:
storageClassName: manual
accessModes:
- ReadWriteMany
resources:
requests:
storage: 3Gi
kubectl apply -f task-pvc-volume.yaml
//Verify
Kuk kubectl get pvc
Answer: B

NEW QUESTION: 40
Get the pods with labels env=dev and env=prod and output the labels as well
Answer:
kubectl get pods -l 'env in (dev,prod)' --show-labels

NEW QUESTION: 41
Create 2 nginx image pods in which one of them is labelled with env=prod and another one labelled with env=dev and verify the same.
A. kubectl run --generator=run-pod/v1 --image=nginx -- labels=env=prod nginx-prod --dry-run -o yaml > nginx-prodpod.yaml Now, edit nginx-prod-pod.yaml file and remove entries like "creationTimestamp: null" "dnsPolicy: ClusterFirst" vim nginx-prod-pod.yaml
apiVersion: v1 kind: Pod metadata:
labels:
env: prod
name: nginx-prod
spec:

```
containers:
- image: nginx
name: nginx-prod
restartPolicy: Always
# kubectl create -f nginx-prod-pod.yaml
kubectl run --generator=run-pod/v1 --image=nginx --
labels=env=dev nginx-dev --dry-run -o yaml > nginx-dev-pod.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
env: dev
name: nginx-dev
spec:
containers:
- image: nginx
name: nginx-dev
restartPolicy: Always
# kubectl create -f nginx-prod-dev.yaml
Verify :
kubectl get po --show-labels
kubectl get po -l env=prod
kubectl get po -l env=dev
```

B. kubectl run --generator=run-pod/v1 --image=nginx -- labels=env=prod nginx-prod --dry-run -o yaml > nginx-prodpod.yaml Now, edit nginx-prod-pod.yaml file and remove entries like "creationTimestamp: null" "dnsPolicy: ClusterFirst" vim nginx-prod-pod.yaml
apiVersion: v1 kind: Pod metadata:

```
labels:
env: prod
name: nginx-prod
spec:
containers:
- image: nginx
name: nginx-prod
restartPolicy: Always
# kubectl create -f nginx-prod-pod.yaml
kubectl run --generator=run-pod/v1 --image=nginx --
labels=env=dev nginx-dev --dry-run -o yaml > nginx-dev-pod.yaml
apiVersion: v1
kind: Pod
metadata:
```

```
- image: nginx
name: nginx-dev
restartPolicy: Always
# kubectl create -f nginx-prod-dev.yaml
```

Verify :

```
kubectl get po --show-labels
kubectl get po -l env=dev
```

Answer: A