

This member-only story is on us. [Upgrade](#) to access all of Medium.

✦ Member-only story

# Practice Enough With These 150 Questions for the CKAD Exam

Exercises get you ready for the Certified Kubernetes Application Developer exam



Bhargav Bachina · [Follow](#)

Published in [Bachina Labs](#) · 21 min read · Nov 11, 2019



2.3K



54

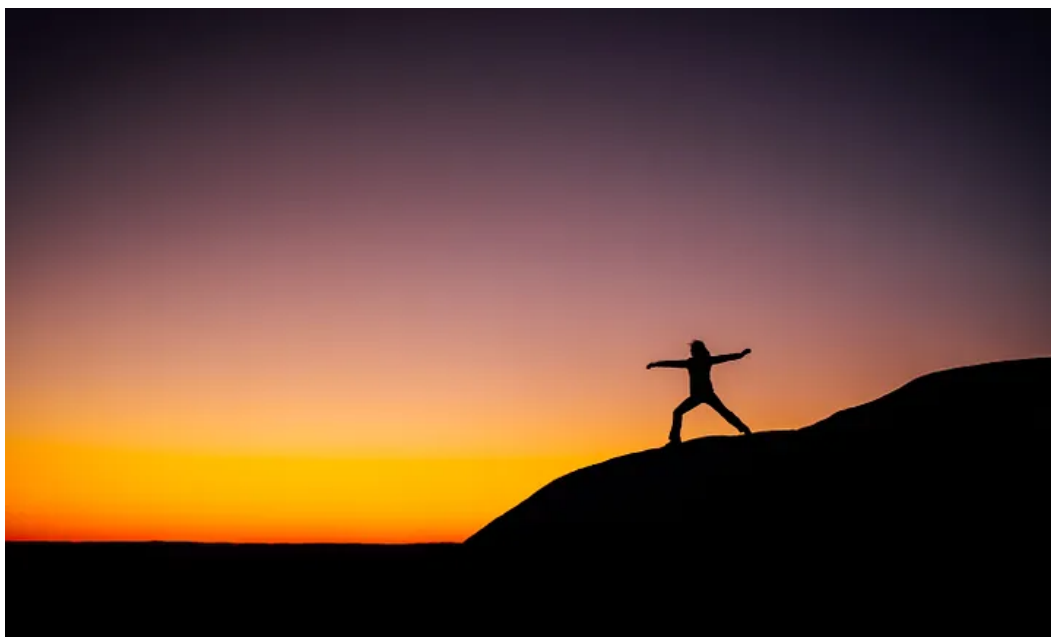


Photo by [Tim Foster](#) on [Unsplash](#)

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. The CNCF/Linux Foundation offers this performance-based exam which targets the developer aspect of kubernetes skills such as deploying apps, configuring apps, rolling out the application, creating persistent volumes, etc.

Since this exam is performance-based rather than just multiple choice questions just knowing the concepts are not enough, we need a lot of practice before the exam. This article helps you understand, practice and get you ready for the exam.

We are not going to discuss any concepts here, rather, I just want to create a bunch of practice questions for the CKAD exam based on the curriculum provided [here](#).

- *Core Concepts (13%)*
- *Multi-Container Pods (10%)*
- *Pod Design (20%)*
- *State Persistence (8%)*
- *Configuration (18%)*
- *Observability (18%)*
- *Services and Networking (13%)*

## **Core Concepts (13%)**

Practice questions based on these concepts

- Understand Kubernetes API Primitives
- Create and Configure Basic Pods

### **1. *List all the namespaces in the cluster***

```
kubectl get namespaces  
kubectl get ns
```

### **2. *List all the pods in all namespaces***

```
kubectl get po --all-namespaces
```

### *3. List all the pods in the particular namespace*

```
kubectl get po -n <namespace name>
```

### *4. List all the services in the particular namespace*

```
kubectl get svc -n <namespace name>
```

### *5. List all the pods showing name and namespace with a json path expression*

```
kubectl get pods -o=jsonpath="{.items[*]['metadata.name'],  
'metadata.namespace']}"
```

### *6. Create an nginx pod in a default namespace and verify the pod running*

```
// creating a pod  
kubectl run nginx --image=nginx --restart=Never  
  
// List the pod  
kubectl get po
```

### *7. Create the same nginx pod with a yaml file*

```
// get the yaml file with --dry-run flag  
kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml >  
nginx-pod.yaml  
  
// cat nginx-pod.yaml  
apiVersion: v1  
kind: Pod  
metadata:  
  creationTimestamp: null  
  labels:  
    run: nginx  
  name: nginx  
spec:  
  containers:  
  - image: nginx  
    name: nginx  
    resources: {}  
    dnsPolicy: ClusterFirst  
    restartPolicy: Never  
status: {}
```

```
// create a pod
kubectl create -f nginx-pod.yaml
```

***8. Output the yaml file of the pod you just created***

```
kubectl get po nginx -o yaml
```

***9. Output the yaml file of the pod you just created without the cluster-specific information***

```
kubectl get po nginx -o yaml --export
```

***10. Get the complete details of the pod you just created***

```
kubectl describe pod nginx
```

***11. Delete the pod you just created***

```
kubectl delete po nginx
kubectl delete -f nginx-pod.yaml
```

***12. Delete the pod you just created without any delay (force delete)***

```
kubectl delete po nginx --grace-period=0 --force
```

***13. Create the nginx pod with version 1.17.4 and expose it on port 80***

```
kubectl run nginx --image=nginx:1.17.4 --restart=Never --port=80
```

***14. Change the Image version to 1.15-alpine for the pod you just created and***

**verify the image version is updated**

```
kubectl set image pod/nginx nginx=nginx:1.15-alpine
kubectl describe po nginx
// another way it will open vi editor and change the version
kubectl edit po nginx
kubectl describe po nginx
```

**15. Change the Image version back to 1.17.1 for the pod you just updated and observe the changes**

```
kubectl set image pod/nginx nginx=nginx:1.17.1
kubectl describe po nginx
kubectl get po nginx -w # watch it
```

**16. Check the Image version without the describe command**

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

**17. Create the nginx pod and execute the simple shell on the pod**

```
// creating a pod
kubectl run nginx --image=nginx --restart=Never
// exec into the pod
kubectl exec -it nginx /bin/sh
```

**18. Get the IP Address of the pod you just created**

```
kubectl get po nginx -o wide
```

**19. Create a busybox pod and run command ls while creating it and check the logs**

```
kubectl run busybox --image=busybox --restart=Never -- ls
kubectl logs busybox
```

*20. If pod crashed check the previous logs of the pod*

```
kubectl logs busybox -p
```

*21. Create a busybox pod with command sleep 3600*

```
kubectl run busybox --image=busybox --restart=Never -- /bin/sh -c
"sleep 3600"
```

*22. Check the connection of the nginx pod from the busybox pod*

```
kubectl get po nginx -o wide
// check the connection
kubectl exec -it busybox -- wget -o- <IP Address>
```

*23. Create a busybox pod and echo message 'How are you' and delete it manually*

```
kubectl run busybox --image=nginx --restart=Never -it -- echo "How
are you"
kubectl delete po busybox
```

*24. Create a busybox pod and echo message 'How are you' and have it deleted immediately*

```
// notice the --rm flag
kubectl run busybox --image=nginx --restart=Never -it --rm -- echo
"How are you"
```

*25. Create an nginx pod and list the pod with different levels of verbosity*

```
// create a pod
kubectl run nginx --image=nginx --restart=Never --port=80

// List the pod with different verbosity
kubectl get po nginx --v=7
kubectl get po nginx --v=8
kubectl get po nginx --v=9
```

## 26. List the nginx pod with custom columns *POD\_NAME* and *POD\_STATUS*

Top highlight

```
kubectl get po -o=custom-columns="POD_NAME:.metadata.name,
POD_STATUS:.status.containerStatuses[].state"
```

## 27. List all the pods sorted by name

```
kubectl get pods --sort-by=.metadata.name
```

## 28. List all the pods sorted by created timestamp

```
kubectl get pods--sort-by=.metadata.creationTimestamp
```

## Multi-Container Pods (10%)

Practice questions based on these concepts

- Understand multi-container pod design patterns (eg: ambassador, adaptor, sidecar)

## 29. 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

```
// first create single container pod with dry run flag
kubectl run busybox --image=busybox --restart=Never --dry-run -o yaml
-- bin/sh -c "sleep 3600; ls" > multi-container.yaml

// edit the pod to following yaml and create it
kubectl create -f multi-container.yaml
```

```
kubectl get po busybox
```

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: busybox
7    name: busybox
8  spec:
9    containers:
10   - args:
11     - bin/sh
12     - -c
13     - ls; sleep 3600
14     image: busybox
15     name: busybox1
16     resources: {}
17   - args:
18     - bin/sh
19     - -c
20     - echo Hello world; sleep 3600
21     image: busybox
22     name: busybox2
23     resources: {}
24   - args:
25     - bin/sh
26     - -c
27     - echo this is third container; sleep 3600
28     image: busybox
29     name: busybox3
30     resources: {}
31   dnsPolicy: ClusterFirst
32   restartPolicy: Never
33   status: {}
```

multi-container.yaml hosted with ♥ by GitHub

[view raw](#)

multi-container pod

### 30. Check the logs of each container that you just created

```
kubectl logs busybox -c busybox1
kubectl logs busybox -c busybox2
kubectl logs busybox -c busybox3
```

### 31. Check the previous logs of the second container busybox2 if any

```
kubectl logs busybox -c busybox2 --previous
```



**32. Run command `ls` in the third container `busybox3` of the above pod**

```
kubectl exec busybox -c busybox3 -- ls
```

**33. Show metrics of the above pod containers and puts them into the `file.log` and verify**

```
kubectl top pod busybox --containers
// putting them into file
kubectl top pod busybox --containers > file.log
cat file.log
```

**34. Create a Pod with main container `busybox` and which executes this “while true; do echo ‘Hi I am from Main container’ >> /var/log/index.html; sleep 5; done” and with sidecar container with `nginx` image which exposes on port 80. Use `emptyDir` Volume and mount this volume on path `/var/log` for `busybox` and on path `/usr/share/nginx/html` for `nginx` container. Verify both containers are running.**

```
// create an initial yaml file with this
kubectl run multi-cont-pod --image=busybox --restart=Never --dry-run -o yaml > multi-container.yaml

// edit the yml as below and create it
kubectl create -f multi-container.yaml

kubectl get po multi-cont-pod
```

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: multi-cont-pod
7    name: multi-cont-pod
8  spec:
9    volumes:
10   - name: var-logs
11     emptyDir: {}
12   containers:
13   - image: busybox
```

```

14     command: ["/bin/sh"]
15     args: ["-c", "while true; do echo 'Hi I am from Main container' >> /var/log/index
16     name: main-container
17     resources: {}
18     volumeMounts:
19     - name: var-logs
20       mountPath: /var/log
21   - image: nginx
22     name: sidecar-container
23     resources: {}
24     ports:
25     - containerPort: 80
26     volumeMounts:
27     - name: var-logs
28       mountPath: /usr/share/nginx/html
29     dnsPolicy: ClusterFirst
30     restartPolicy: Never
31     status: {}

```

multi-container.yaml hosted with ♥ by GitHub

[view raw](#)

multi-container.yaml

**35. Exec into both containers and verify that main.txt exist and query the main.txt from sidecar container with curl localhost**

```

// exec into main container
kubectl exec -it multi-cont-pod -c main-container -- sh
cat /var/log/main.txt

// exec into sidecar container
kubectl exec -it multi-cont-pod -c sidecar-container -- sh
cat /usr/share/nginx/html/index.html

// install curl and get default page
kubectl exec -it multi-cont-pod -c sidecar-container -- sh
# apt-get update && apt-get install -y curl
# curl localhost

```

## Pod Design (20%)

Practice questions based on these concepts

- Understand how to use Labels, Selectors and Annotations
- Understand Deployments and how to perform rolling updates
- Understand Deployments and how to perform rollbacks
- Understand Jobs and CronJobs

***36. Get the pods with label information***

```
kubectl get pods --show-labels
```

***37. Create 5 nginx pods in which two of them is labeled env=prod and three of them is labeled env=dev***

```
kubectl run nginx-dev1 --image=nginx --restart=Never --labels=env=dev
kubectl run nginx-dev2 --image=nginx --restart=Never --labels=env=dev
kubectl run nginx-dev3 --image=nginx --restart=Never --labels=env=dev

kubectl run nginx-prod1 --image=nginx --restart=Never --
labels=env=prod
kubectl run nginx-prod2 --image=nginx --restart=Never --
labels=env=prod
```

***38. Verify all the pods are created with correct labels***

```
kubectl get pods --show-labels
```

***39. Get the pods with label env=dev***

```
kubectl get pods -l env=dev
```

***40. Get the pods with label env=dev and also output the labels***

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

***41. Get the pods with label env=prod***

```
kubectl get pods -l env=prod
```

***42. Get the pods with label env=prod and also output the labels***

```
kubectl get pods -l env=prod --show-labels
```

***43. Get the pods with label env***

```
kubectl get pods -L env
```

***44. Get the pods with labels env=dev and env=prod***

```
kubectl get pods -l 'env in (dev,prod)'
```

***45. Get the pods with labels env=dev and env=prod and output the labels as well***

```
kubectl get pods -l 'env in (dev,prod)' --show-labels
```

***46. Change the label for one of the pod to env=uat and list all the pods to verify***

```
kubectl label pod/nginx-dev3 env=uat --overwrite  
kubectl get pods --show-labels
```

***47. Remove the labels for the pods that we created now and verify all the labels are removed***

```
kubectl label pod nginx-dev{1..3} env-  
kubectl label pod nginx-prod{1..2} env-  
kubectl get po --show-labels
```

***48. Let's add the label app=nginx for all the pods and verify***

```
kubectl label pod nginx-dev{1..3} app=nginx  
kubectl label pod nginx-prod{1..2} app=nginx  
kubectl get po --show-labels
```

**49. Get all the nodes with labels (if using minikube you would get only master node)**

```
kubectl get nodes --show-labels
```

**50. Label the node (minikube if you are using) nodeName=nginxnode**

```
kubectl label node minikube nodeName=nginxnode
```

**51. Create a Pod that will be deployed on this node with the label nodeName=nginxnode**

```
kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml > pod.yaml
```

```
// add the nodeSelector like below and create the pod  
kubectl create -f pod.yaml
```

```
1  apiVersion: v1  
2  kind: Pod  
3  metadata:  
4    creationTimestamp: null  
5    labels:  
6      run: nginx  
7    name: nginx  
8  spec:  
9    nodeSelector:  
10     nodeName: nginxnode  
11   containers:  
12   - image: nginx  
13     name: nginx  
14     resources: {}  
15     dnsPolicy: ClusterFirst  
16     restartPolicy: Never  
17   status: {}
```

pod.yaml hosted with ❤ by GitHub

[view raw](#)

pod.yaml

**52. Verify the pod that it is scheduled with the node selector**

```
kubectl describe po nginx | grep Node-Selectors
```

**53. Verify the pod nginx that we just created has this label**

```
kubectl describe po nginx | grep Labels
```

**54. Annotate the pods with name=webapp**

```
kubectl annotate pod nginx-dev{1..3} name=webapp  
kubectl annotate pod nginx-prod{1..2} name=webapp
```

**55. Verify the pods that have been annotated correctly**

```
kubectl describe po nginx-dev{1..3} | grep -i annotations  
kubectl describe po nginx-prod{1..2} | grep -i annotations
```

**56. Remove the annotations on the pods and verify**

```
kubectl annotate pod nginx-dev{1..3} name-  
kubectl annotate pod nginx-prod{1..2} name-  
  
kubectl describe po nginx-dev{1..3} | grep -i annotations  
kubectl describe po nginx-prod{1..2} | grep -i annotations
```

**57. Remove all the pods that we created so far**

```
kubectl delete po --all
```

**58. Create a deployment called webapp with image nginx with 5 replicas**

```
kubectl create deploy webapp --image=nginx --dry-run -o yaml >  
webapp.yaml  
  
// change the replicas to 5 in the yaml and create it
```

```
kubectl create -f webapp.yaml
```

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    creationTimestamp: null
5    labels:
6      app: webapp
7    name: webapp
8  spec:
9    replicas: 5
10   selector:
11     matchLabels:
12       app: webapp
13   strategy: {}
14   template:
15     metadata:
16       creationTimestamp: null
17     labels:
18       app: webapp
19     spec:
20       containers:
21       - image: nginx
22         name: nginx
23         resources: {}
24   status: {}
```

webapp.yaml hosted with ♥ by GitHub

[view raw](#)

webapp.yaml

### 59. Get the deployment you just created with labels

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

### 60. Output the yaml file of the deployment you just created

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

### 61. Get the pods of this deployment

```
// get the label of the deployment
kubectl get deploy --show-labels

// get the pods with that label
```

```
kubectl get pods -l app=webapp
```

***62. Scale the deployment from 5 replicas to 20 replicas and verify***

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

***63. Get the deployment rollout status***

```
kubectl rollout status deploy webapp
```

***64. Get the replicaset that created with this deployment***

```
kubectl get rs -l app=webapp
```

***65. Get the yaml of the replicaset and pods of this deployment***

```
kubectl get rs -l app=webapp -o yaml  
kubectl get po -l app=webapp -o yaml
```

***66. Delete the deployment you just created and watch all the pods are also being deleted***

```
kubectl delete deploy webapp  
kubectl get po -l app=webapp -w
```

***67. Create a deployment of webapp with image nginx:1.17.1 with container port 80 and verify the image version***

```
kubectl create deploy webapp --image=nginx:1.17.1 --dry-run -o yaml >  
webapp.yaml
```



```
// add the port section and create the deployment
kubectl create -f webapp.yaml

// verify
kubectl describe deploy webapp | grep Image
```

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    creationTimestamp: null
5    labels:
6      app: webapp
7    name: webapp
8  spec:
9    replicas: 1
10   selector:
11     matchLabels:
12       app: webapp
13   strategy: {}
14   template:
15     metadata:
16       creationTimestamp: null
17     labels:
18       app: webapp
19     spec:
20       containers:
21       - image: nginx:1.17.1
22         name: nginx
23         ports:
24         - containerPort: 80
25         resources: {}
26   status: {}
```

webapp.yaml hosted with ❤ by GitHub

[view raw](#)

webapp.yaml

### 68. Update the deployment with the image version 1.17.4 and verify

```
kubectl set image deploy/webapp nginx=nginx:1.17.4
kubectl describe deploy webapp | grep Image
```

### 69. Check the rollout history and make sure everything is ok after the update

```
kubectl rollout history deploy webapp

kubectl get deploy webapp --show-labels
kubectl get rs -l app=webapp
kubectl get po -l app=webapp
```

***70. Undo the deployment to the previous version 1.17.1 and verify Image has the previous version***

```
kubectl rollout undo deploy webapp  
kubectl describe deploy webapp | grep Image
```

***71. Update the deployment with the image version 1.16.1 and verify the image and also check the rollout history***

```
kubectl set image deploy/webapp nginx=nginx:1.16.1  
kubectl describe deploy webapp | grep Image  
kubectl rollout history deploy webapp
```

***72. Update the deployment to the Image 1.17.1 and verify everything is ok***

```
kubectl rollout undo deploy webapp --to-revision=3  
kubectl describe deploy webapp | grep Image  
kubectl rollout status deploy webapp
```

***73. Update the deployment with the wrong image version 1.100 and verify something is wrong with the deployment***

```
kubectl set image deploy/webapp nginx=nginx:1.100  
kubectl rollout status deploy webapp (still pending state)  
kubectl get pods (ImagePullErr)
```

***74. Undo the deployment with the previous version and verify everything is Ok***

```
kubectl rollout undo deploy webapp  
kubectl rollout status deploy webapp  
kubectl get pods
```

***75. Check the history of the specific revision of that deployment***

```
kubectl rollout history deploy webapp --revision=7
```

***76. Pause the rollout of the deployment***

```
kubectl rollout pause deploy webapp
```

***77. Update the deployment with the image version latest and check the history and verify nothing is going on***

```
kubectl set image deploy/webapp nginx=nginx:latest  
kubectl rollout history deploy webapp (No new revision)
```

***78. Resume the rollout of the deployment***

```
kubectl rollout resume deploy webapp
```

***79. Check the rollout history and verify it has the new version***

```
kubectl rollout history deploy webapp  
kubectl rollout history deploy webapp --revision=9
```

***80. 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***

```
kubectl autoscale deploy webapp --min=10 --max=20 --cpu-percent=85  
kubectl get hpa  
kubectl get pod -l app=webapp
```

***81. Clean the cluster by deleting deployment and hpa you just created***

```
kubectl delete deploy webapp
kubectl delete hpa webapp
```

***82. Create a Job with an image node which prints node version and also verifies there is a pod created for this job***

```
kubectl create job nodeversion --image=node -- node -v
kubectl get job -w
kubectl get pod
```

***83. Get the logs of the job just created***

```
kubectl logs <pod name> // created from the job
```

***84. Output the yaml file for the Job with the image busybox which echos "Hello I am from job"***

```
kubectl create job hello-job --image=busybox --dry-run -o yaml --
echo "Hello I am from job"
```

***85. Copy the above YAML file to hello-job.yaml file and create the job***

```
kubectl create job hello-job --image=busybox --dry-run -o yaml --
echo "Hello I am from job" > hello-job.yaml
kubectl create -f hello-job.yaml
```

***86. Verify the job and the associated pod is created and check the logs as well***

```
kubectl get job
kubectl get po
kubectl logs hello-job-*
```

*87. Delete the job we just created*

```
kubectl delete job hello-job
```

*88. Create the same job and make it run 10 times one after one*

```
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  
kubectl create -f hello-job.yaml
```

hello-job.yaml

**89. Watch the job that runs 10 times one by one and verify 10 pods are created and delete those after it's completed**

```
kubectrl get job -w
kubectrl get po

kubectrl delete job hello-job
```

**90. Create the same job and make it run 10 times parallel**

```
kubectrl 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 parallelism: 10
kubectrl create -f hello-job.yaml
```

```
1  apiVersion: batch/v1
2  kind: Job
3  metadata:
4    creationTimestamp: null
5    name: hello-job
6  spec:
7    parallelism: 10
8    template:
9      metadata:
10       creationTimestamp: null
11     spec:
12       containers:
13       - command:
14         - echo
15         - Hello I am from job
16         image: busybox
17         name: hello-job
18         resources: {}
19         restartPolicy: Never
20     status: {}
```

hello-job.yaml hosted with ♥ by GitHub

[view raw](#)

hello-job.yaml

**91. Watch the job that runs 10 times parallelly and verify 10 pods are created and delete those after it's completed**

```
kubectrl get job -w
kubectrl get po

kubectrl delete job hello-job
```

*92. Create a Cronjob with busybox image that prints date and hello from kubernetes cluster message for every minute*

```
kubectl create cronjob date-job --image=busybox --schedule="*/1 * * * *" -- bin/sh -c "date; echo Hello from kubernetes cluster"
```

*93. Output the YAML file of the above cronjob*

```
kubectl get cj date-job -o yaml
```

*94. Verify that CronJob creating a separate job and pods for every minute to run and verify the logs of the pod*

```
kubectl get job
kubectl get po
kubectl logs date-job-<jobid>-<pod>
```

*95. Delete the CronJob and verify all the associated jobs and pods are also deleted.*

```
kubectl delete cj date-job
// verify pods and jobs
kubectl get po
kubectl get job
```

## State Persistence (8%)

Practice questions based on these concepts

- Understand PersistentVolumeClaims for Storage

***96. List Persistent Volumes in the cluster***

```
kubectl get pv
```

***97. Create a hostPath PersistentVolume named task-pv-volume with storage 10Gi, access modes ReadWriteOnce, storageClassName manual, and volume at /mnt/data and verify***

```
kubectl create -f task-pv-volume.yaml  
kubectl get pv
```

```
task-pv-volume.yaml
```

***98. Create a PersistentVolumeClaim of at least 3Gi storage and access mode***



### *ReadWriteOnce and verify status is Bound*

```
kubectl create -f task-pv-claim.yaml  
kubectl get pvc
```

```
1  apiVersion: v1  
2  kind: PersistentVolumeClaim  
3  metadata:  
4    name: task-pv-claim  
5  spec:  
6    storageClassName: manual  
7    accessModes:  
8      - ReadWriteOnce  
9    resources:  
10     requests:  
11       storage: 3Gi
```

task-pv-claim.yaml hosted with ♥ by GitHub

[view raw](#)

task-pv-claim.yaml

### *99. Delete persistent volume and PersistentVolumeClaim we just created*

```
kubectl delete pvc task-pv-claim  
kubectl delete pv task-pv-volume
```

### *100. Create a Pod with an image Redis and configure a volume that lasts for the lifetime of the Pod*

```
// emptyDir is the volume that lasts for the life of the pod  
kubectl create -f redis-storage.yaml
```

*101. Exec into the above pod and create a file named file.txt with the text 'This is called the file' in the path /data/redis and open another tab and exec again with the same pod and verifies file exist in the same path.*

```
// first terminal
kubectl exec -it redis-storage /bin/sh
cd /data/redis
echo 'This is called the file' > file.txt

//open another tab
kubectl exec -it redis-storage /bin/sh
cat /data/redis/file.txt
```

*102. Delete the above pod and create again from the same yaml file and verifies there is no file.txt in the path /data/redis*

```
kubectl delete pod redis

kubectl create -f redis-storage.yaml
kubectl exec -it redis-storage /bin/sh
cat /data/redis/file.txt // file doesn't exist
```

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

```
kubectl create -f task-pv-volume.yaml
kubectl create -f task-pv-claim.yaml

kubectl get pv
kubectl get pvc
```

**104. Create an nginx pod with containerPort 80 and with a PersistentVolumeClaim `task-pv-claim` and has a mount path `"/usr/share/nginx/html"`**

```
kubectl create -f task-pv-pod.yaml
```

task-pv-pod.yaml

## Configuration (18%)

Practice questions based on these concepts

- Understand ConfigMaps
- Understand SecurityContexts
- Define an application's resource requirements
- Create & Consume Secrets
- Understand ServiceAccounts

*105. List all the configmaps in the cluster*

```
kubectl get cm  
or  
kubectl get configmap
```

*106. Create a configmap called myconfigmap with literal value appname=myapp*

```
kubectl create cm myconfigmap --from-literal=appname=myapp
```

*107. Verify the configmap we just created has this data*

```
// you will see under data  
kubectl get cm -o yaml  
or  
kubectl describe cm
```

*108. delete the configmap myconfigmap we just created*

```
kubectl delete cm myconfigmap
```

*109. Create a file called config.txt with two values key1=value1 and key2=value2 and verify the file*

```
cat >> config.txt << EOF
key1=value1
key2=value2
EOF

cat config.txt
```

***110. Create a configmap named keyvalcfgmap and read data from the file config.txt and verify that configmap is created correctly***

```
kubectl create cm keyvalcfgmap --from-file=config.txt
kubectl get cm keyvalcfgmap -o yaml
```

***111. Create an nginx pod and load environment values from the above configmap keyvalcfgmap and exec into the pod and verify the environment variables and delete the pod***

```
// first run this command to save the pod yaml
kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml >
nginx-pod.yaml

// edit the yaml to below file and create
kubectl create -f nginx-pod.yaml

// verify
kubectl exec -it nginx -- env

kubectl delete po nginx
```

nginx-pod.yml

***112. Create an env file file.env with var1=val1 and create a configmap envcfgmap from this env file and verify the configmap***

```
echo var1=val1 > file.env
cat file.env

kubectl create cm envcfgmap --from-env-file=file.env
kubectl get cm envcfgmap -o yaml --export
```

***113. Create an nginx pod and load environment values from the above configmap envcfgmap and exec into the pod and verify the environment variables and delete the pod***

```
// first run this command to save the pod yml
kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml >
nginx-pod.yml

// edit the yml to below file and create
kubectl create -f nginx-pod.yml

// verify
kubectl exec -it nginx -- env

kubectl delete po nginx
```

nginx-pod.yaml

***114. Create a configmap called cfgvolume with values var1=val1, var2=val2 and create an nginx pod with volume nginx-volume which reads data from this configmap cfgvolume and put it on the path /etc/cfg***

```
// first create a configmap cfgvolume
kubectl create cm cfgvolume --from-literal=var1=val1 --from-
literal=var2=val2

// verify the configmap
kubectl describe cm cfgvolume

// create the config map
kubectl create -f nginx-volume.yml

// exec into the pod
kubectl exec -it nginx -- /bin/sh

// check the path
cd /etc/cfg
ls
```

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: nginx
7    name: nginx
8  spec:
9    volumes:
10   - name: nginx-volume
11     configMap:
12       name: cfgvolume
```

```

13     containers:
14     - image: nginx
15       name: nginx
16       resources: {}
17       volumeMounts:
18       - name: nginx-volume
19         mountPath: /etc/cfg
20     dnsPolicy: ClusterFirst
21     restartPolicy: Never
22     status: {}

```

nginx-volume.yml hosted with ❤ by GitHub

[view raw](#)

nginx-volume.yml

**115. Create a pod called secbusybox with the image busybox which executes command sleep 3600 and makes sure any Containers in the Pod, all processes run with user ID 1000 and with group id 2000 and verify.**

```

// create yml file with dry-run
kubectl run secbusybox --image=busybox --restart=Never --dry-run -o
yaml -- /bin/sh -c "sleep 3600;" > busybox.yml

// edit the pod like below and create
kubectl create -f busybox.yml

// verify
kubectl exec -it secbusybox -- sh
id // it will show the id and group

```

```

1  apiVersion: v1
2  kind: Pod
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: secbusybox
7    name: secbusybox
8  spec:
9    securityContext: # add security context
10     runAsUser: 1000
11     runAsGroup: 2000
12    containers:
13    - args:
14      - /bin/sh
15      - -c
16      - sleep 3600;
17      image: busybox
18      name: secbusybox
19      resources: {}
20      dnsPolicy: ClusterFirst
21      restartPolicy: Never
22      status: {}

```

busybox.yml hosted with ❤ by GitHub

[view raw](#)



busybox.yml

***116. Create the same pod as above this time set the securityContext for the container as well and verify that the securityContext of container overrides the Pod level securityContext.***

```
// create yml file with dry-run
kubectl run secbusybox --image=busybox --restart=Never --dry-run -o
yaml -- /bin/sh -c "sleep 3600;" > busybox.yml

// edit the pod like below and create
kubectl create -f busybox.yml

// verify
kubectl exec -it secbusybox -- sh
id // you can see container securityContext overrides the Pod level
```

busybox.yml

***117. Create pod with an nginx image and configure the pod with capabilities***

***NET\_ADMIN and SYS\_TIME verify the capabilities***

```
// create the yaml file
kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml >
nginx.yml

// edit as below and create pod
kubectl create -f nginx.yml

// exec and verify
kubectl exec -it nginx -- sh
cd /proc/1
cat status

// you should see these values
CapPrm: 00000000aa0435fb
CapEff: 00000000aa0435fb
```

nginx.yml

***118. Create a Pod nginx and specify a memory request and a memory limit of 100Mi and 200Mi respectively.***

```
// create a yml file
kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml >
nginx.yml

// add the resources section and create
kubectl create -f nginx.yml

// verify
kubectl top pod
```

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: nginx
7    name: nginx
8  spec:
9    containers:
10   - image: nginx
11     name: nginx
12     resources:
13       requests:
14         memory: "100Mi"
15       limits:
16         memory: "200Mi"
17     dnsPolicy: ClusterFirst
18     restartPolicy: Never
19   status: {}
```

nginx.yml hosted with ❤ by GitHub

[view raw](#)

nginx.yml

***119. Create a Pod nginx and specify a CPU request and a CPU limit of 0.5 and 1 respectively.***

```
// create a yml file
kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml >
nginx.yml

// add the resources section and create
kubectl create -f nginx.yml

// verify
kubectl top pod
```

nginx.yml

***120. Create a Pod nginx and specify both CPU, memory requests and limits together and verify.***

```
// create a yml file
kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml >
nginx.yml

// add the resources section and create
kubectl create -f nginx.yml

// verify
kubectl top pod
```

nginx.yml

*121. Create a Pod nginx and specify a memory request and a memory limit of 100Gi and 200Gi respectively which is too big for the nodes and verify pod fails to start because of insufficient memory*

```
// create a yml file
kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml >
nginx.yml

// add the resources section and create
kubectl create -f nginx.yml

// verify
kubectl describe po nginx // you can see pending state
```

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: nginx
7    name: nginx
8  spec:
9    containers:
10   - image: nginx
11     name: nginx
12     resources:
13       requests:
14         memory: "100Gi"
15         cpu: "0.5"
16       limits:
17         memory: "200Gi"
18         cpu: "1"
19   dnsPolicy: ClusterFirst
```

```
20     restartPolicy: Never
21     status: {}
```

nginx.yml hosted with ❤ by GitHub

[view raw](#)

nginx.yml

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

```
kubectl create secret generic my-secret --from-literal=username=user
--from-literal=password=mypassword
```

**123. List the secrets in all namespaces**

```
kubectl get secret --all-namespaces
```

**124. Output the yaml of the secret created above**

```
kubectl get secret my-secret -o yaml
```

**125. Create an nginx pod which reads username as the environment variable**

```
// create a yaml file
kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml >
nginx.yml

// add env section below and create
kubectl create -f nginx.yml

//verify
kubectl exec -it nginx -- env
```

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: nginx
7    name: nginx
8  spec:
9    containers:
10     - image: nginx
```

```
11     name: nginx
12     env:
13     - name: USER_NAME
14       valueFrom:
15         secretKeyRef:
16           name: my-secret
17           key: username
18     resources: {}
19     dnsPolicy: ClusterFirst
20     restartPolicy: Never
21     status: {}
```

nginx.yml hosted with ❤ by GitHub

[view raw](#)

nginx.yml

### *126. Create an nginx pod which loads the secret as environment variables*

```
// create a yml file
kubectl run nginx --image=nginx --restart=Never --dry-run -o yaml >
nginx.yml

// add env section below and create
kubectl create -f nginx.yml

//verify
kubectl exec -it nginx -- env
```

nginx.yml

***127. List all the service accounts in the default namespace***

```
kubectl get sa
```

***128. List all the service accounts in all namespaces***

```
kubectl get sa --all-namespaces
```

***129. Create a service account called admin***

```
kubectl create sa admin
```

***130. Output the YAML file for the service account we just created***

```
kubectl get sa admin -o yaml
```

***131. Create a busybox pod which executes this command sleep 3600 with the service account admin and verify***

```
kubectl run busybox --image=busybox --restart=Never --dry-run -o yaml  
-- /bin/sh -c "sleep 3600" > busybox.yml
```

```
kubectl create -f busybox.yml
```

```
// verify  
kubectl describe po busybox
```

```
1  apiVersion: v1  
2  kind: Pod  
3  metadata:
```



```
4   creationTimestamp: null
5   labels:
6     run: busybox
7   name: busybox
8   spec:
9     serviceAccountName: admin
10    containers:
11    - args:
12      - /bin/sh
13      - -c
14      - sleep 3600
15      image: busybox
16      name: busybox
17      resources: {}
18    dnsPolicy: ClusterFirst
19    restartPolicy: Never
20    status: {}
```

busybox.yml hosted with ♥ by GitHub

[view raw](#)

busybox.yml

## Observability (18%)

Practice questions based on these concepts

- Understand LivenessProbes and ReadinessProbes
- Understand Container Logging
- Understand how to monitor applications in kubernetes
- Understand Debugging in Kubernetes

*132. Create an nginx pod with containerPort 80 and it should only receive traffic only it checks the endpoint / on port 80 and verify and delete the pod.*

```
kubectl run nginx --image=nginx --restart=Never --port=80 --dry-run -o yml > nginx-pod.yaml
// add the readinessProbe section and create
kubectl create -f nginx-pod.yaml
// verify
kubectl describe pod nginx | grep -i readiness
kubectl delete po nginx
```

```
1  apiVersion: v1
2  kind: Pod
```

```
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: nginx
7    name: nginx
8  spec:
9    containers:
10   - image: nginx
11     name: nginx
12     ports:
13     - containerPort: 80
14     readinessProbe:
15       httpGet:
16         path: /
17         port: 80
18     resources: {}
19   dnsPolicy: ClusterFirst
20   restartPolicy: Never
21  status: {}
```

nginx-pod.yaml hosted with ♥ by GitHub

[view raw](#)

nginx-pod.yaml

**133. Create an nginx pod with containerPort 80 and it should check the pod running at endpoint / healthz on port 80 and verify and delete the pod.**

```
kubectl run nginx --image=nginx --restart=Never --port=80 --dry-run -o yaml > nginx-pod.yaml
```

```
// add the livenessProbe section and create
kubectl create -f nginx-pod.yaml
```

```
// verify
kubectl describe pod nginx | grep -i readiness

kubectl delete po nginx
```

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: nginx
7    name: nginx
8  spec:
9    containers:
10   - image: nginx
11     name: nginx
12     ports:
13     - containerPort: 80
14     livenessProbe:
15       httpGet:
16         path: /healthz
17         port: 80
```

```

18     resources: {}
19     dnsPolicy: ClusterFirst
20     restartPolicy: Never
21     status: {}

```

nginx-pod.yaml hosted with ♥ by GitHub

[view raw](#)

nginx-pod.yaml

**134. Create an nginx pod with containerPort 80 and it should check the pod running at endpoint /healthz on port 80 and it should only receive traffic only it checks the endpoint / on port 80. verify the pod.**

```

kubectl run nginx --image=nginx --restart=Never --port=80 --dry-run -
o yaml > nginx-pod.yaml

```

```

// add the livenessProbe and readiness section and create
kubectl create -f nginx-pod.yaml

```

```

// verify
kubectl describe pod nginx | grep -i readiness
kubectl describe pod nginx | grep -i liveness

```

```

1  apiVersion: v1
2  kind: Pod
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: nginx
7    name: nginx
8  spec:
9    containers:
10   - image: nginx
11     name: nginx
12     ports:
13       - containerPort: 80
14     livenessProbe:
15       httpGet:
16         path: /healthz
17         port: 80
18     readinessProbe:
19       httpGet:
20         path: /
21         port: 80
22     resources: {}
23     dnsPolicy: ClusterFirst
24     restartPolicy: Never
25     status: {}

```

nginx-pod.yaml hosted with ♥ by GitHub

[view raw](#)

nginx-pod.yaml

**135. Check what all are the options that we can configure with readiness and liveness probes**

```
kubectl explain Pod.spec.containers.livenessProbe
kubectl explain Pod.spec.containers.readinessProbe
```

**136. Create the pod nginx with the above liveness and readiness probes so that it should wait for 20 seconds before it checks liveness and readiness probes and it should check every 25 seconds.**

```
kubectl create -f nginx-pod.yaml
```

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: nginx
7    name: nginx
8  spec:
9    containers:
10   - image: nginx
11     name: nginx
12     ports:
13     - containerPort: 80
14     livenessProbe:
15       initialDelaySeconds: 20
16       periodSeconds: 25
17       httpGet:
18         path: /healthz
19         port: 80
20     readinessProbe:
21       initialDelaySeconds: 20
22       periodSeconds: 25
23       httpGet:
24         path: /
25         port: 80
26     resources: {}
27     dnsPolicy: ClusterFirst
28     restartPolicy: Never
29   status: {}
```

nginx-pod.yaml hosted with ♥ by GitHub

[view raw](#)

nginx-pod.yaml

**137. Create a busybox pod with this command “echo I am from busybox pod;**

*sleep 3600;" and verify the logs.*

```
kubectrl run busybox --image=busybox --restart=Never -- /bin/sh -c
"echo I am from busybox pod; sleep 3600;"

kubectrl logs busybox
```

*138. copy the logs of the above pod to the busybox-logs.txt and verify*

```
kubectrl logs busybox > busybox-logs.txt
cat busybox-logs.txt
```

*139. List all the events sorted by timestamp and put them into file.log and verify*

```
kubectrl get events --sort-by=.metadata.creationTimestamp

// putting them into file.log
kubectrl get events --sort-by=.metadata.creationTimestamp > file.log
cat file.log
```

*140. Create a pod with an image alpine which executes this command "while true; do echo 'Hi I am from alpine'; sleep 5; done" and verify and follow the logs of the pod.*

```
// create the pod
kubectrl run hello --image=alpine --restart=Never -- /bin/sh -c
"while true; do echo 'Hi I am from Alpine'; sleep 5;done"

// verify and follow the logs
kubectrl logs --follow hello
```

*141. Create the pod with this `kubectrl create -f`*

*<https://gist.githubusercontent.com/bbachi/212168375b39e36e2e2984c097167b00/raw/1fd63509c3ae3a3d3da844640fb4cca744543c1c/not-running.yml>. The pod is not in the running state. Debug it.*

```
// create the pod
kubectrl create -f
https://gist.githubusercontent.com/bbachi/212168375b39e36e2e2984c0971
```

[67b00/raw/1fd63509c3ae3a3d3da844640fb4cca744543c1c/not-running.yml](https://gist.githubusercontent.com/bbachi/1f001f10337234d46806929d12245397/raw/84b7295fb077f15de979fec5b3f7a13fc69c6d83/problem-pod.yaml)

```
// get the pod
kubectl get pod not-running
kubectl describe po not-running

// it clearly says ImagePullBackOff something wrong with image
kubectl edit pod not-running // it will open vim editor
or
kubectl set image pod/not-running not-running=nginx
```

**142. This following yaml creates 4 namespaces and 4 pods. One of the pod in one of the namespaces are not in the running state. Debug and fix it.**  
<https://gist.githubusercontent.com/bbachi/1f001f10337234d46806929d12245397/raw/84b7295fb077f15de979fec5b3f7a13fc69c6d83/problem-pod.yaml>.

```
kubectl create -f
https://gist.githubusercontent.com/bbachi/1f001f10337234d46806929d12245397/raw/84b7295fb077f15de979fec5b3f7a13fc69c6d83/problem-pod.yaml

// get all the pods in all namespaces
kubectl get po --all-namespaces

// find out which pod is not running
kubectl get po -n namespace2

// update the image
kubectl set image pod/pod2 pod2=nginx -n namespace2

// verify again
kubectl get po -n namespace2
```

**143. Get the memory and CPU usage of all the pods and find out top 3 pods which have the highest usage and put them into the cpu-usage.txt file**

```
// get the top 3 hungry pods
kubectl top pod --all-namespaces | sort --reverse --key 3 --numeric | head -3

// putting into file
kubectl top pod --all-namespaces | sort --reverse --key 3 --numeric | head -3 > cpu-usage.txt

// verify
cat cpu-usage.txt
```

## Services and Networking (13%)

Practice questions based on these concepts

- Understand Services
- Demonstrate a basic understanding of NetworkPolicies

*144. Create an nginx pod with a yaml file with label my-nginx and expose the port 80*

```
kubectl run nginx --image=nginx --restart=Never --port=80 --dry-run -o yaml > nginx.yaml
```

```
// edit the label app: my-nginx and create the pod  
kubectl create -f nginx.yaml
```

```
1  apiVersion: v1  
2  kind: Pod  
3  metadata:  
4    creationTimestamp: null  
5    labels:  
6      app: my-nginx  
7    name: nginx  
8  spec:  
9    containers:  
10   - image: nginx  
11     name: nginx  
12     ports:  
13     - containerPort: 80  
14     resources: {}  
15   dnsPolicy: ClusterFirst  
16   restartPolicy: Never  
17   status: {}
```

nginx.yaml hosted with ♥ by GitHub

[view raw](#)

nginx.yaml

*145. Create the service for this nginx pod with the pod selector app: my-nginx*

```
// create the below service  
kubectl create -f nginx-svc.yaml
```

```
1  apiVersion: v1  
2  kind: Service  
3  metadata:  
4    name: my-service  
5  spec:  
6    selector:  
7      app: my-nginx  
8    ports:
```

```

9      - protocol: TCP
10     port: 80
11     targetPort: 9376

```

nginx-svc.yaml hosted with ❤ by GitHub

[view raw](#)

nginx-svc.yaml

**146. Find out the label of the pod and verify the service has the same label**

```

// get the pod with labels
kubectl get po nginx --show-labels

// get the service and chekc the selector column
kubectl get svc my-service -o wide

```

**147. Delete the service and create the service with kubectl expose command and verify the label**

```

// delete the service
kubectl delete svc my-service

// create the service again
kubectl expose po nginx --port=80 --target-port=9376

// verify the label
kubectl get svc -l app=my-nginx

```

**148. Delete the service and create the service again with type NodePort**

```

// delete the service
kubectl delete svc nginx

// create service with expose command
kubectl expose po nginx --port=80 --type=NodePort

```

**149. Create the temporary busybox pod and hit the service. Verify the service that it should return the nginx page index.html.**


 Search

Write



```

// get the cluster IP from this command
kubectl get svc nginx -o wide

// create temporary busybox to check the nodeport
kubectl run busybox --image=busybox --restart=Never -it --rm -- wget
-o- <Cluster IP>:80

```



### 150. Create a NetworkPolicy which denies all ingress traffic

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

## Conclusion

CKAD is a performance-based exam and it's all about completing 19 questions within 2 hours. We need a lot of practice for it. These 150 questions give you enough practice for the exam. The more you practice the more comfortable you feel during the exam. Practice. Practice. Practice.

[Kubernetes](#)[DevOps](#)[Web Development](#)[Programming](#)[Software Development](#)

## Written by Bhargav Bachina

22K Followers · Editor for Bachina Labs

Entrepreneur | 600+ Tech Articles | Subscribe to upcoming Videos  
<https://www.youtube.com/channel/UCWLSuUuIkLIQvbMHRUfKM-g> |  
<https://www.linkedin.com/in/bachina>

[Follow](#)

## More from Bhargav Bachina and Bachina Labs



 Bhargav Bachina in Bits and Pieces

## How to Implement Idle Timeout in Angular

Learn how to use idle-timeout in your Angular app to improve performance and security

6 min read · Sep 30, 2019



975



13



...



 Bhargav Bachina in Bachina Labs

## 250 Practice Questions For Terraform Associate Certification

Read and Practice these questions before your exam

★ · 47 min read · Jul 13, 2020



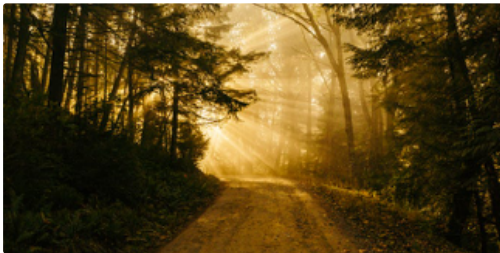
1.7K



39



...



 Bhargav Bachina in Bachina Labs

## React—How To Proxy To Backend Server

Explaining how to configure a proxy for backend API calls with an example.

★ · 6 min read · Jun 14, 2020



713



9



...



 Bhargav Bachina in Bachina Labs

## How To Develop and Build React App With Java Backend

Learn How you develop and build with an example project

★ · 7 min read · Jun 17, 2020



473



9



...

See all from Bhargav Bachina

See all from Bachina Labs

## Recommended from Medium

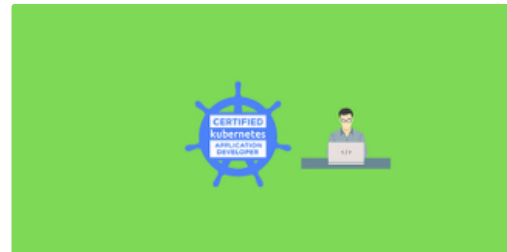


 Shafath Ahmed

### I passed CKAD: Here is what you need to know

I passed CKAD exam on September 2023 after studying about 1 month. In this article, I

8 min read · Sep 5, 2023



 Mohamed BEN HASSINE (teckbootcamps.com)

### CKAD exam with our 2024 Ultimate Guide

Enhance your preparation for the Certified Kubernetes Developer (CKAD) Exam using

8 min read · Sep 25, 2023



## Lists



### General Coding Knowledge

20 stories · 978 saves



### Stories to Help You Grow as a Software Developer

19 stories · 865 saves



### Coding & Development


11 stories · 480 saves



### Leadership

48 stories · 255 saves



 The Optimizer

## How do I pass the Certified Kubernetes Administrator


Hola! 🙋

3 min read · Sep 15, 2023

 25 



 Theophilus Okwugwuni

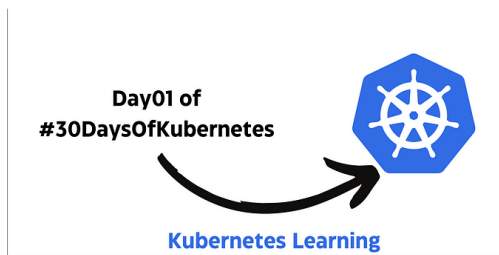
## CKAD Exam Preparation Tips

5 tips to help you ace the Certified Kubernetes Application Developer Exam

3 min read · Sep 9, 2023

 2 



 Aman Pathak in DevOps.dev

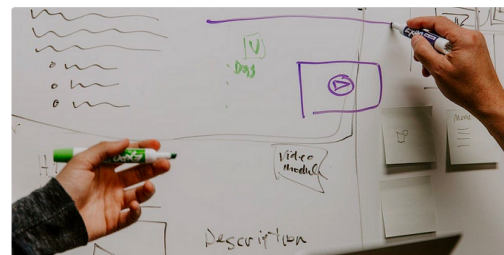
## Unlocking the Power of Kubernetes: Day 01


Introduction: Welcome to Day 01 of the #30DaysOfKubernetes challenge! In this

5 min read · Oct 3, 2023

 469  4





 Carlos Arguelles

## My favorite coding question to give candidates

A coding question, from the viewpoint of an Google/Amazon/Microsoft interviewer

11 min read · Nov 12, 2023

 6.5K  58

[See more recommendations](#)