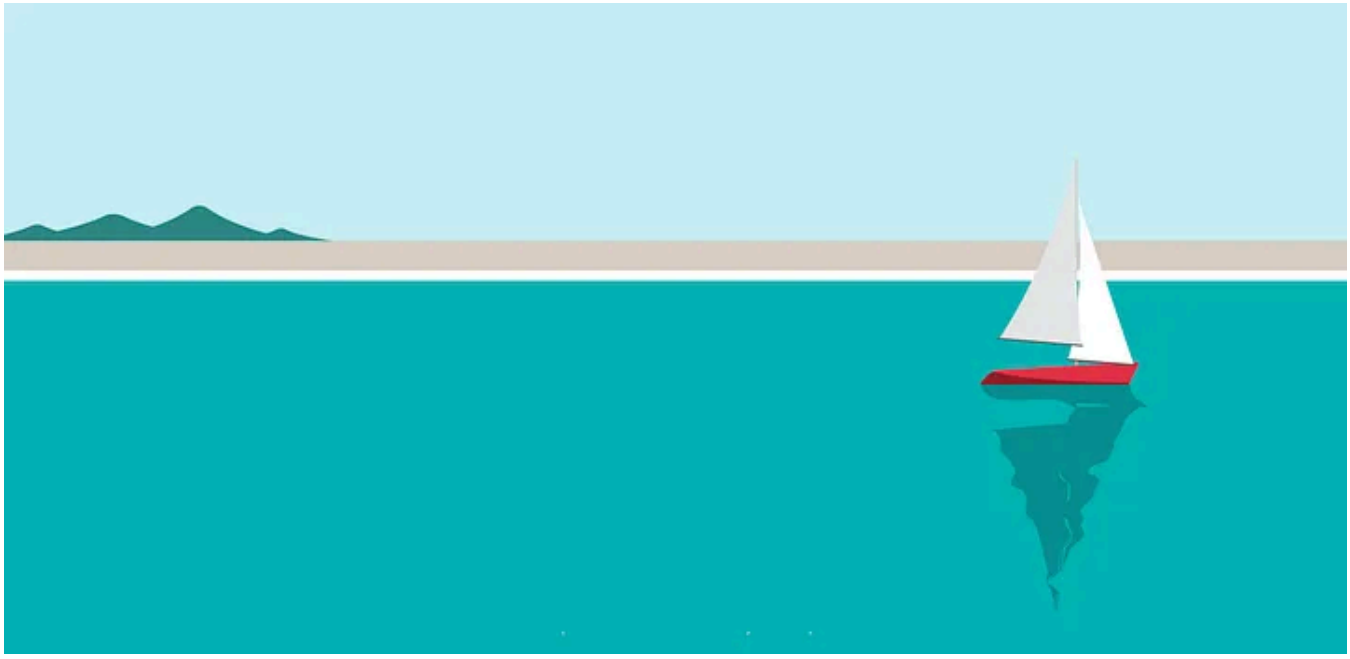
Kubernetes - CKAD preparation

Eric Hemmerlin · Follow
6 min read · Jan 3, 2021

Listen Share More



Let's face it, all these yaml files in Kubernetes can be overwhelming. Here is what helped me during my preparation to pass the CKAD exam : building step by step, a big, giant, Pod yaml file in a specific Namespace, containing a Service Account, an Environment variable, a Port, Resources (limits and requests), a Label, a Command, an ImagePullPolicy, Secrets, ConfigMaps, Volumes and an InitContainer.

```
> alias k=kubectl
```

Namespace

```
> k create ns ckad
> k config set-context --current --namespace=ckad
```

Pod

Generate a Pod with Service Account, Environment variable, Port, Resources, Label, Command, ImagePullPolicy

```
> k run nginx --image=nginx --env=USER=user --port=80 --
serviceaccount=default --limits='cpu=200m,memory=512Mi' --
requests='cpu=200m,memory=512Mi' --labels=app=dev --image-pull-
policy=IfNotPresent --command --dry-run=client -oyaml -- /bin/sh -c
"echo 'Hello Kubernetes!'; sleep 3600" > pod.yaml
```

```
> cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    app: dev
  name: nginx
spec:
  containers:
  - command:
    - /bin/sh
    - -c
    - echo 'Hello Kubernetes!'; sleep 3600
    env:
    - name: USER
      value: user
    image: nginx
    imagePullPolicy: Never
    name: nginx
    ports:
    - containerPort: 80
    resources:
      limits:
        cpu: 200m
        memory: 512Mi
      requests:
        cpu: 200m
        memory: 512Mi
  dnsPolicy: ClusterFirst
```

```
  restartPolicy: Always
  serviceAccountName: default
status: {}
```

```
> k logs nginx
Hello Kubernetes!
```

Secrets

Secret from literal

```
> k create secret generic secret-literal --from-
literal=username=user --from-literal=password=pass
```

```
> k get secret secret-literal -oyaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-literal
  namespace: ckad
type: Opaque
data:
  password: cGFzcw==
  username: dXNlcg==
```

Secret from env file

```
> vi secret.env
```

Secret env file

```
> cat secret.env
MYSQL_USER=1234
MYSQL_PASS=azer
```

```
> k create secret generic secret-env-file --from-env-file=secret.env
```

```
> k get secret secret-env-file -oyaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-env-file
  namespace: ckad
type: Opaque
data:
```

```
  MYSQL_PASS: YXplcg==
  MYSQL_USER: MTIzNA==
```

Secret from file

```
> vi secret.file
secret.code.true=true
secret.code.false=false

> k create secret generic secret-file --from-file=secret-
file=secret.file --from-file=secret.file

> k get secret secret-file -oyaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-file
  namespace: ckad
type: Opaque
data:
  secret-file:
c2VjcmV0LmNvZGUudHJ1ZT10cnVlCnNlY3JldC5jb2RlLmZhbHNlPWZhbHNlCg==
  secret.file:
c2VjcmV0LmNvZGUudHJ1ZT10cnVlCnNlY3JldC5jb2RlLmZhbHNlPWZhbHNlCg==
```

Pod using all these secrets

```
> vi pod.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    app: dev
  name: nginx
spec:
  containers:
  - command:
    - /bin/sh
    - -c
    - echo 'Hello Kubernetes!'; sleep 3600
    env:
    - name: USER
      value: user
    - name: SECRET_USERNAME
      valueFrom:
        secretKeyRef:
          name: secret-literal
          key: username
    - name: SECRET_PASSWORD
```

```
        valueFrom:
          secretKeyRef:
            name: secret-literal
            key: password
    envFrom:
    - secretRef:
        name: secret-env-file
    image: nginx
    imagePullPolicy: IfNotPresent
    name: nginx
    ports:
    - containerPort: 80
    resources:
      limits:
        cpu: 200m
        memory: 512Mi
      requests:
        cpu: 200m
        memory: 512Mi
    volumeMounts:
    - name: secret-file
      mountPath: "/etc/secret"
      readOnly: true
  volumes:
  - name: secret-file
    secret:
      secretName: secret-file
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  serviceAccountName: default
status: {}
```

```
> k apply -f pod.yaml
```

```
> k exec nginx -- env
SECRET_USERNAME=user
SECRET_PASSWORD=pass
MYSQL_PASS=azer
MYSQL_USER=1234
USER=user
```

```
> k exec nginx -- ls /etc/secret
secret-file
secret.file
```

```
> k exec nginx -- more /etc/secret/secret-file
/etc/secret/secret.file
::::::::::::::
/etc/secret/secret-file
::::::::::::::
secret.code.true=true
secret.code.false=false
::::::::::::::
/etc/secret/secret.file
::::::::::::::
```

```
secret.code.true=true
secret.code.false=false
```

```
> k exec nginx -- touch /etc/secret/file
touch: cannot touch '/etc/secret/file': Read-only file system
command terminated with exit code 1
```

ConfigMap

ConfigMap from literal

```
> k create configmap configmap-literal --from-
literal=special.how=very --from-literal=special.type=charm
```

```
> k get cm configmap-literal -oyaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: configmap-literal
  namespace: ckad
data:
  special.how: very
  special.type: charm
```

ConfigMap from env file

```
> k create configmap configmap-env-file --from-env-
file=configmap.env
```

```
> k get cm configmap-env-file -oyaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: configmap-env-file
  namespace: ckad
data:
  allowed: '"true"'
  lives: "3"
```

ConfigMap from file

```
> k create cm configmap-file --from-file=configmap-
file=configmap.file --from-file=configmap.file
```

```
> k get cm configmap-file -oyaml
apiVersion: v1
kind: ConfigMap
```

```
metadata:
  name: configmap-file
  namespace: ckad
data:
  configmap-file: |
    color.good=purple
    color.bad=yellow
  configmap.file: |
    color.good=purple
    color.bad=yellow
```

Pod using all these ConfigMaps

```
> vi pod.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    app: dev
  name: nginx
spec:
  containers:
  - command:
    - /bin/sh
    - -c
    - echo 'Hello Kubernetes!'; sleep 3600
    env:
    - name: USER
      value: user
    - name: SECRET_USERNAME
      valueFrom:
        secretKeyRef:
          name: secret-literal
          key: username
    - name: SECRET_PASSWORD
      valueFrom:
        secretKeyRef:
          name: secret-literal
          key: password
    - name: CONFIGMAP_SPECIAL_HOW
      valueFrom:
        configMapKeyRef:
          name: configmap-literal
          key: special.how
    - name: CONFIGMAP_SPECIAL_TYPE
      valueFrom:
        configMapKeyRef:
          name: configmap-literal
          key: special.type
    envFrom:
    - secretRef:
```

```
          name: secret-env-file
      - configMapRef:
          name: configmap-env-file
      image: nginx
      imagePullPolicy: IfNotPresent
      name: nginx
      ports:
      - containerPort: 80
      resources:
        limits:
          cpu: 200m
          memory: 512Mi
        requests:
          cpu: 200m
          memory: 512Mi
      volumeMounts:
      - name: secret-file
        mountPath: "/etc/secret"
        readOnly: true
      - name: configmap-file
        mountPath: "/etc/configmap"
        readOnly: true
    volumes:
    - name: secret-file
      secret:
        secretName: secret-file
    - name: configmap-file
      configMap:
        name: configmap-file
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    serviceAccountName: default
  status: {}
```

```
> k apply -f pod.yaml
```

```
> k exec nginx -- env
MYSQL_PASS=azer
MYSQL_USER=1234
USER=user
SECRET_USERNAME=user
SECRET_PASSWORD=pass
CONFIGMAP_SPECIAL_HOW=very
CONFIGMAP_SPECIAL_TYPE=charm
lives=3
allowed="true"
```

```
> k exec nginx -- ls /etc/configmap
configmap-file
configmap.file
```

```
> k exec nginx -- more /etc/configmap/configmap-file
/etc/configmap/configmap.file
::::::::::::::
/etc/configmap/configmap-file
::::::::::::::
```

```
color.good=purple
color.bad=yellow
::::::::::::::
/etc/configmap/configmap.file
::::::::::::::
color.good=purple
color.bad=yellow
```

Volumes

HostPath volume

```
> mkdir /tmp/ckad
> echo 'Hello from Kubernetes storage' > /tmp/ckad/message.txt
```

```
> vi hostpath-storage.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: hostpath-pv
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Mi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/tmp/ckad"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: hostpath-pvc
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Mi
```

```
> k get pv,pvc
NAME                              CAPACITY   ACCESS MODES   RECLAIM
POLICY   STATUS     CLAIM               STORAGECLASS
persistentvolume/hostpath-pv      10Mi       RWO            Retain
Bound      ckad/hostpath-pvc   manual

NAME                                   STATUS   VOLUME        CAPACITY
ACCESS MODES   STORAGECLASS
```

```
persistentvolumeclaim/hostpath-pvc   Bound    hostpath-pv   10Mi
RWO            manual
```

Pod mounting this HostPath volume and an emptyDir

```
> vi pod.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    app: dev
  name: nginx
spec:
  containers:
  - command:
    - /bin/sh
    - -c
    - echo 'Hello Kubernetes!'; sleep 3600
    env:
    - name: USER
      value: user
    - name: SECRET_USERNAME
      valueFrom:
        secretKeyRef:
          name: secret-literal
          key: username
    - name: SECRET_PASSWORD
      valueFrom:
        secretKeyRef:
          name: secret-literal
          key: password
    - name: CONFIGMAP_SPECIAL_HOW
      valueFrom:
        configMapKeyRef:
          name: configmap-literal
          key: special.how
    - name: CONFIGMAP_SPECIAL_TYPE
      valueFrom:
        configMapKeyRef:
          name: configmap-literal
          key: special.type
    envFrom:
    - secretRef:
        name: secret-env-file
    - configMapRef:
        name: configmap-env-file
    image: nginx
    imagePullPolicy: IfNotPresent
    name: nginx
    ports:
    - containerPort: 80
```

```
      resources:
        limits:
          cpu: 200m
          memory: 512Mi
        requests:
          cpu: 200m
          memory: 512Mi
      volumeMounts:
      - name: secret-file
        mountPath: "/etc/secret"
        readOnly: true
      - name: configmap-file
        mountPath: "/etc/configmap"
        readOnly: true
      - name: hostpath-storage
        mountPath: "/mnt/hostpath"
      - name: emptydir-storage
        mountPath: "/mnt/emptydir"
    volumes:
    - name: secret-file
      secret:
        secretName: secret-file
    - name: configmap-file
      configMap:
        name: configmap-file
    - name: hostpath-storage
      persistentVolumeClaim:
        claimName: hostpath-pvc
    - name: emptydir-storage
      emptyDir: {}
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    serviceAccountName: default
  status: {}
```

```
> k exec nginx -- ls /mnt
emptydir
hostpath
```

```
> k exec nginx -- ls /mnt/emptydir
```

```
> k exec nginx -- ls /mnt/hostpath
message.txt
```

```
> k exec nginx -- cat /mnt/hostpath/message.txt
Hello from Kubernetes storage
```

Init container

```
> k exec nginx -- curl localhost
Failed to connect to localhost port 80: Connection refused
```

```
> vi pod.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    app: dev
  name: nginx
spec:
  initContainers:
  - name: init
    image: busybox
    command:
    - /bin/sh
    - -c
    - echo 'Hello Kubernetes!'; sleep 2
  containers:
  - env:
    - name: USER
      value: user
    - name: SECRET_USERNAME
      valueFrom:
        secretKeyRef:
          name: secret-literal
          key: username
    - name: SECRET_PASSWORD
      valueFrom:
        secretKeyRef:
          name: secret-literal
          key: password
    - name: CONFIGMAP_SPECIAL_HOW
      valueFrom:
        configMapKeyRef:
          name: configmap-literal
          key: special.how
    - name: CONFIGMAP_SPECIAL_TYPE
      valueFrom:
        configMapKeyRef:
          name: configmap-literal
          key: special.type
    envFrom:
    - secretRef:
        name: secret-env-file
    - configMapRef:
        name: configmap-env-file
    image: nginx
    imagePullPolicy: IfNotPresent
    name: nginx
    ports:
    - containerPort: 80
    resources:
      limits:
        cpu: 200m
        memory: 512Mi
      requests:
```

```
          cpu: 200m
          memory: 512Mi
      volumeMounts:
      - name: secret-file
        mountPath: "/etc/secret"
        readOnly: true
      - name: configmap-file
        mountPath: "/etc/configmap"
        readOnly: true
      - name: hostpath-storage
        mountPath: "/mnt/hostpath"
      - name: emptydir-storage
        mountPath: "/mnt/emptydir"
  volumes:
  - name: secret-file
    secret:
      secretName: secret-file
  - name: configmap-file
    configMap:
      name: configmap-file
  - name: hostpath-storage
    persistentVolumeClaim:
      claimName: hostpath-pvc
  - name: emptydir-storage
    emptyDir: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  serviceAccountName: default
status: {}
```

```
> k exec nginx -- curl localhost
<html>
...
</html>
```

```
> k logs nginx -c init
Hello Kubernetes!
```

Credits

- Image created by rawpixel.com - www.freepik.com

Kubernetes

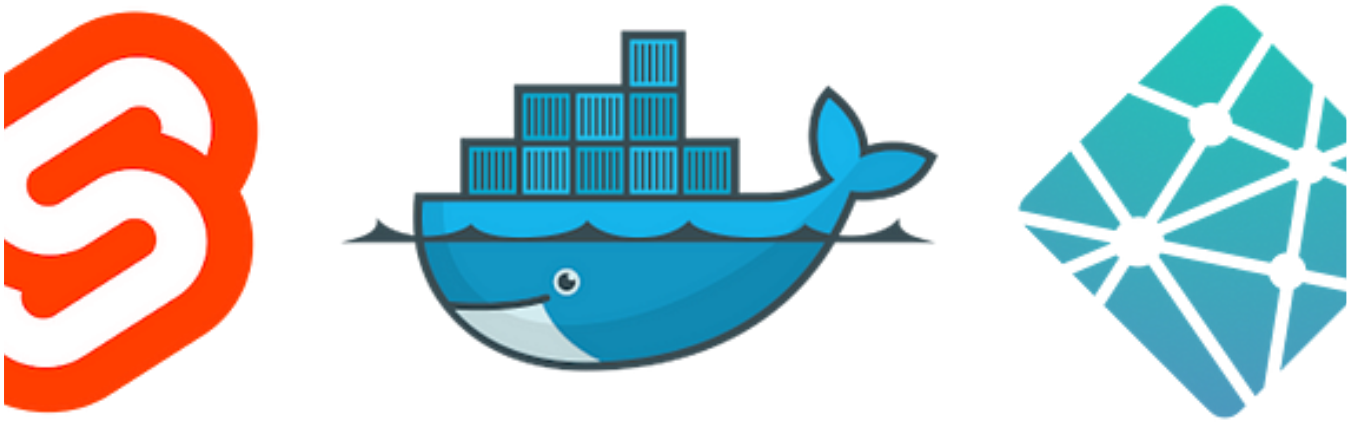
Follow

Written by Eric Hemmerlin

5 Followers

With 10+ years of experience in software engineering and 7+ years as a training manager, I'm a passionate engineer embracing the serverless computing journey.

More from Eric Hemmerlin

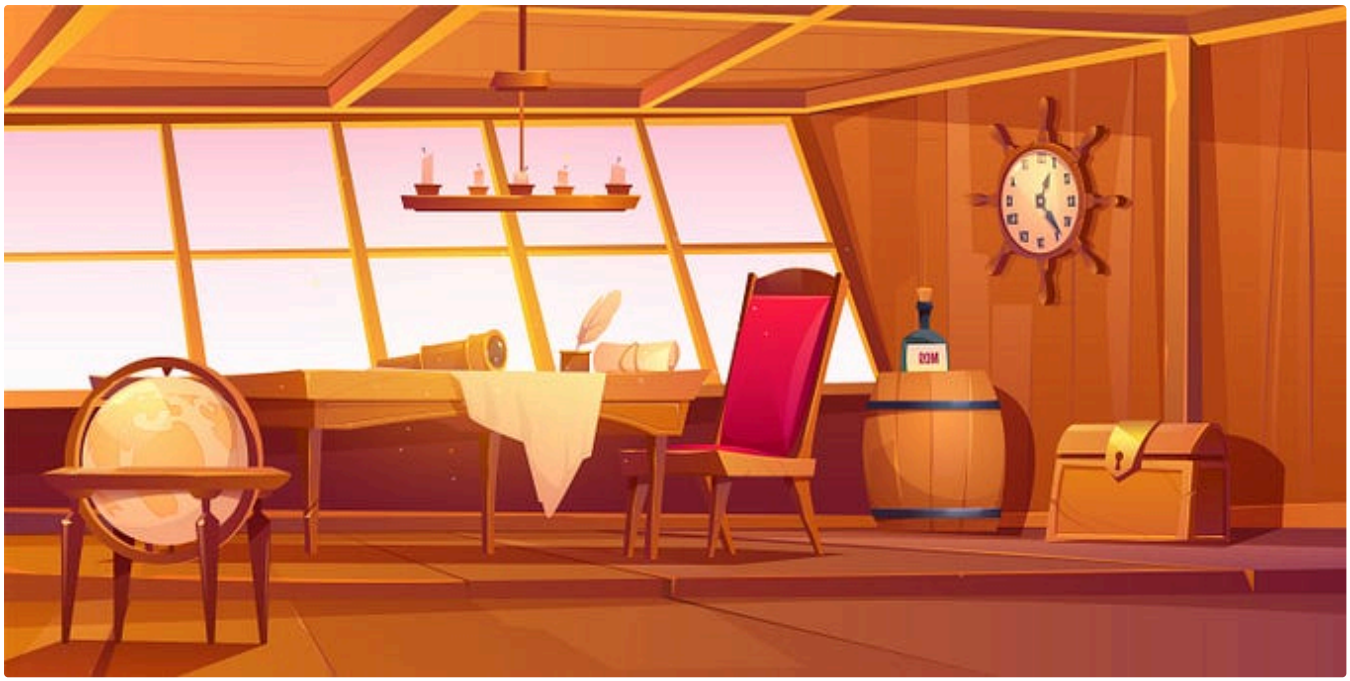


Eric Hemmerlin

Svelte in Docker deployed to Netlify

Develop Svelte in Docker without Node.js installed on your local machine and deploy our Svelte application to Netlify.

Jan 3, 2021 7



Eric Hemmerlin

Kubernetes in Docker on AWS EC2

Using AWS CLI we'll provision an EC2 Linux machine pre-installed with Git, Docker and k3d in order to launch a Kubernetes cluster in...

Jan 16, 2021 11

Eric Hemmerlin

GitLab Auto DevOps on DigitalOcean Kubernetes cluster

Provision a DigitalOcean-managed Kubernetes cluster and deploy a NodeJs application to production using GitLab Auto DevOps with Prometheus...

Oct 3, 2019

See all from Eric Hemmerlin

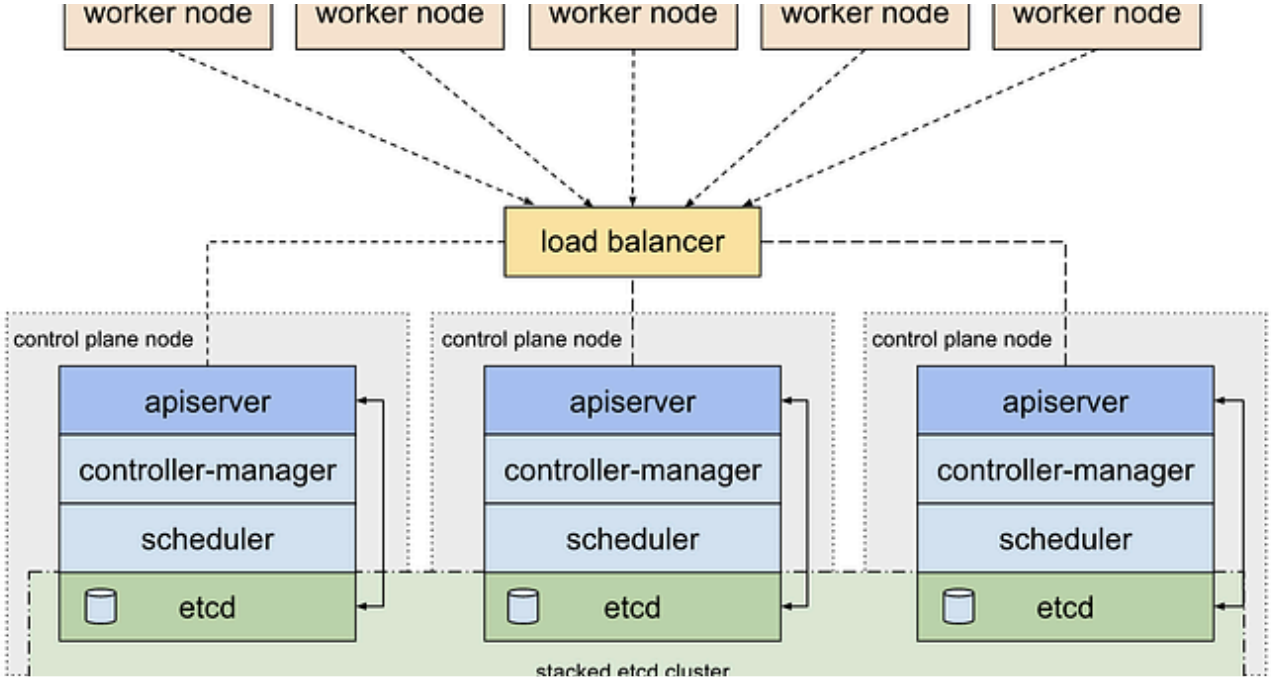
Recommended from Medium

Ben Neighbour in weeklycloud

11 Cool Kubernetes Tools

These tools 10x my productivity

Oct 29 123 1



Mr.PlanB

Maximizing High Availability: Spanning a Kubernetes Cluster Across Multiple Datacenters

Ensuring high availability for critical applications is more important than ever. For enterprises leveraging Kubernetes for their...

Oct 23 24 2

Lists

Natural Language Processing

1792 stories · 1405 saves

The Devops Girl

Comprehensive Guide to RBAC in Kubernetes: Setting Up ClusterRole , ClusterRoleBinding, and AWS IAM...

Learn how to secure your Kubernetes cluster with RBAC, integrate AWS IAM, and manage permissions efficiently. This step-by-step guide...

Jul 28 52

NRT0401

Article 4: Kubernetes Networking: Services, Ingress, and DNS

1. Overview of Kubernetes Networking

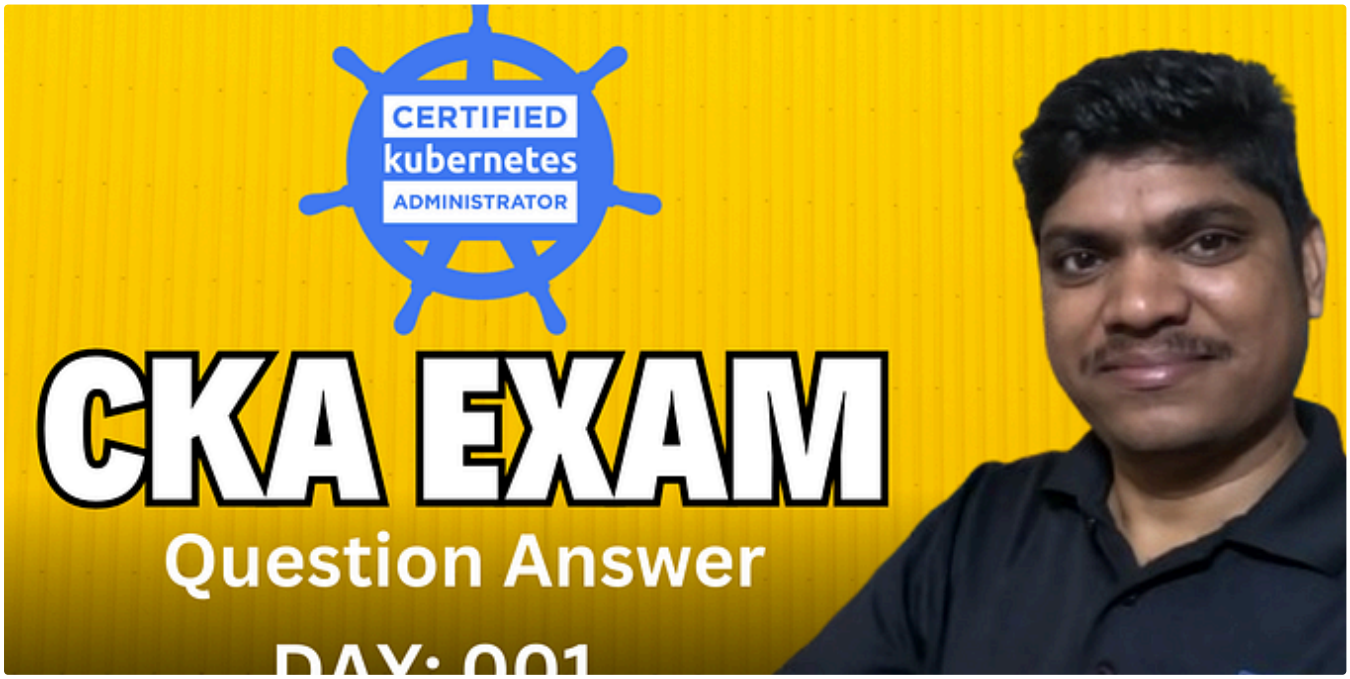
Oct 11 3

Jinha Jeong

Accessing the kubernetes Service in the Default Namespace from Your Pods

Ever notices that kubernetes service is located in the default namespace and wondered about its purpose?

Jun 22 9



Tech Mahato

Day 001 | CKA Exam Preparation | Contexts

Question 1: Contexts

Sep 17

See more recommendations