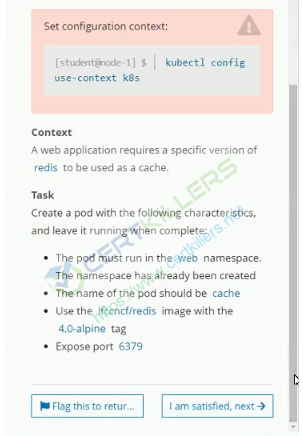
1



Ans: -

#kubectl create namespace web

#Kubectl run cache --image=lfccncf/redis:4.0-alpine -n web

Kubectl expose pod/cache --port=6379 -n web

2 –



Ans:

<https://kubernetes.io/docs/concepts/configuration/secret/>

# kubectl create secret generic another-secret --from-literal=key1=value4

# vi pod.yaml

**apiVersion**: v1

**kind**: Pod

**metadata**:

**name**: nginx-secret

**spec**:

**containers**:

- **name**: nginx

**image**: nginx

**env**:

- **name**: BEST\_VARIABLE

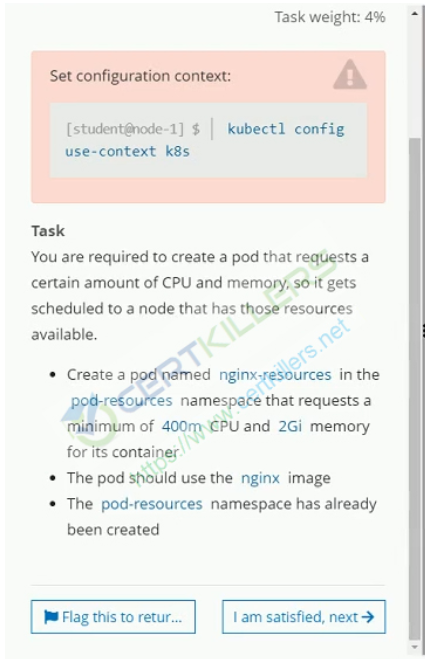
**valueFrom**:

**secretKeyRef**:

**name**: another-secret

**key**: key1

# kubectl create –f pod.yaml

3-

Ans:

**apiVersion**: v1

**kind**: Pod

**metadata**:

**name**: nginx-resources

**namespace**: pod-resources

**spec**:

**containers**:

- **name**: nginx

**image**: nginx

**resources**:

**requests**:

**memory**: "2Gi"

**cpu:** “400m”

4-



Ans:

# kubectl create cm some-config --from-literal=key4=value1

#

# vi nginx-configmap.yaml

**apiVersion**: v1

**kind**: Pod

**metadata**:

**name**: nginx-configmap

**spec**:

**containers**:

- **name**: nginx

**image**: nginx

**volumeMounts**:

- **name**: foo

**mountPath**: "/some/path"

**readOnly**: **true**

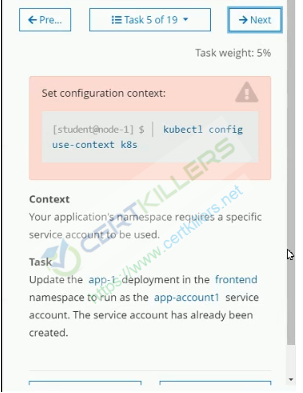
**volumes**:

- **name**: foo

**configMap**:

**name**: some-config

5-



Ans:

# kubectl edit deployment/app-1 -n frontend

**apiVersion**: extensions/v1beta1

**kind**: Deployment

**metadata**:

**name**: app-1

**namespace**: frontend

**spec**:

**replicas**: 1

**template**:

**metadata**:

**labels**:

**app**: nginx

**spec**:

**containers**:

- **name**: nginx

**image**: nginx

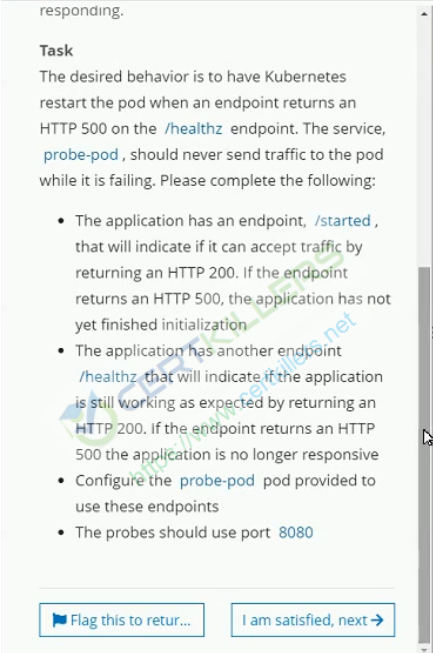
**ports**:

- **containerPort**: 80

**serviceAccountName**: app-account-1

6-

-



Ans:

<https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/>

**apiVersion**: v1

**kind**: Pod

**metadata**:

**labels**:

**test**: liveness

**name**: liveness-http

**spec**:

**containers**:

- **name**: liveness

**image**: k8s.gcr.io/liveness

**args**:

- /server

**livenessProbe**:

**httpGet**:

**path**: /healthz

**port**: 8080

**httpHeaders**:

- **name**: Custom-Header

**value**: Awesome

**initialDelaySeconds**: 3

**periodSeconds**: 3

**readinessProbe**:

**httpGet**:

**path**: /started

**port**: 8080

**httpHeaders**:

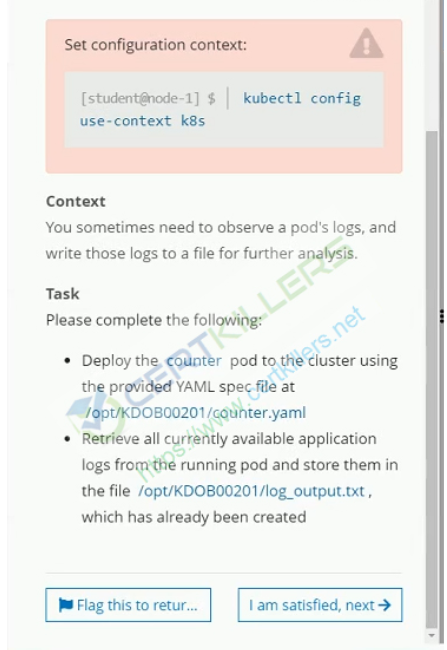
- **name**: Custom-Header

**value**: Awesome

**initialDelaySeconds**: 3

**periodSeconds**: 3

7-

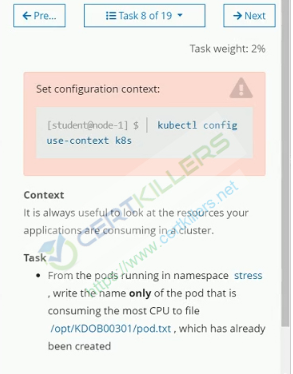


Ans:

#kubectl create –f /opt/KDOB00201/counter.yaml

#kubectl logs <pod\_name> > /opt/KDOB00201/log\_output.txt

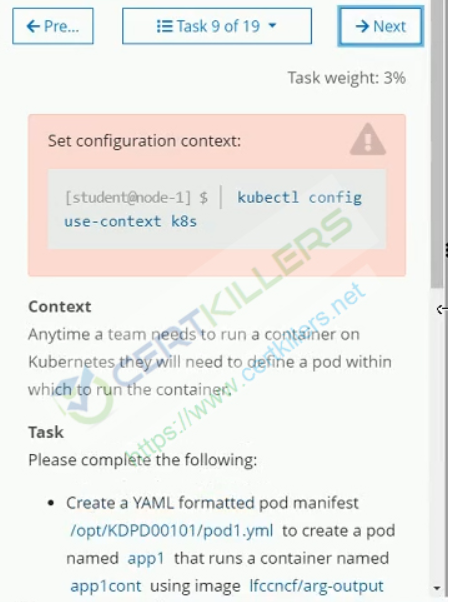
8-

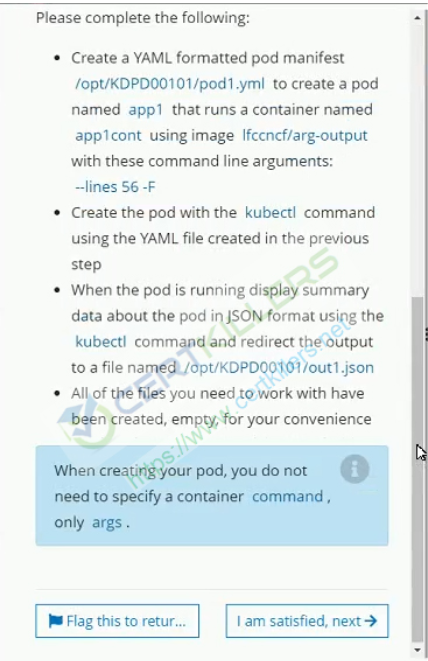


Ans:

#kubectl top pod -n stress --no-headers –sort-by=cpu| head -1 |awk ‘{print $1}’ > /opt/KDOB00301/pod.txt

9-





Ans:

# vi /opt/KDPD00101/pod1.yaml

**apiVersion**: v1

**kind**: Pod

**metadata**:

**name**: app1

**labels**:

**purpose**: demonstrate-command

**spec**:

**containers**:

- **name**: app1cont

**image**: lfccncf/arg-output

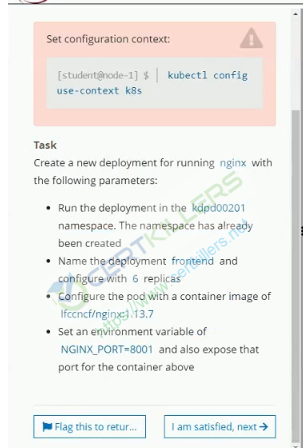
**args**: ["--line", "56", “-F”]

**restartPolicy**: OnFailure

# kubectl –f apply /opt/KDPD00101/pod1.yaml

# kubectl get pod app1 -o json > /opt/KDPD00101/out1.json

10-



Ans:

# kubectl create namespace kdpd00201

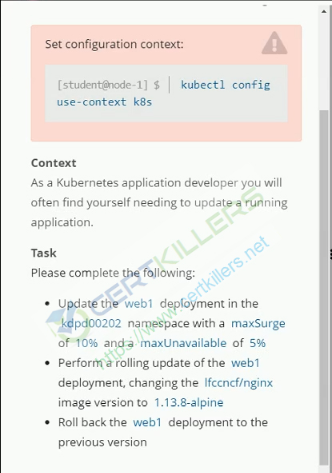
# kubectl create deployment frontend --image=lfccncf/nginx:1.13.7 -n kdpd00201 ;

# kubectl scale --replicas=6 deployment/frontend -n kdpd00201 ;

#kubectl set env deployment frontend NGINX\_PORT=8001 -n kdpd00201

# kubectl expose deployment frontend --port=8001 -n kdpd00201

11-



Ans:

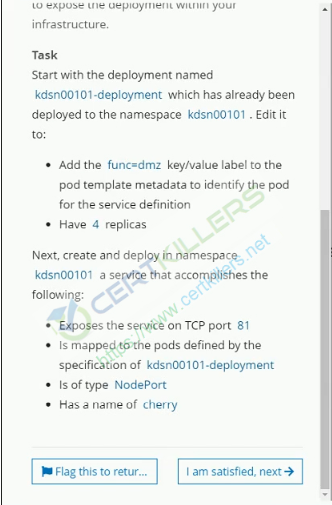
# kubectl edit deployment/web1 -n kdpd00202 // change the value for maxSurge and maxUnavailable

# kubectl set image deployment/web1 nginx=lfccncf/nginx:1.3.8-alpine -n kdpd00202

# kubectl rollout undo deployment/web1 -n kdpd00202

12-





#kubectl create namespace kdsn00101

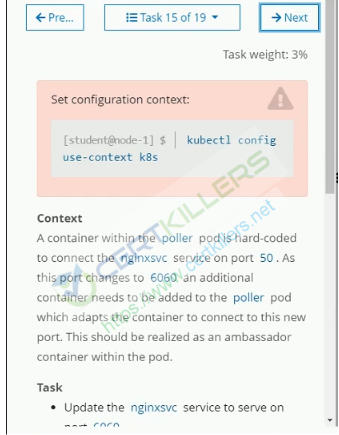
#kubectl create deployment kdsn00101-deployment -n kdsn00101

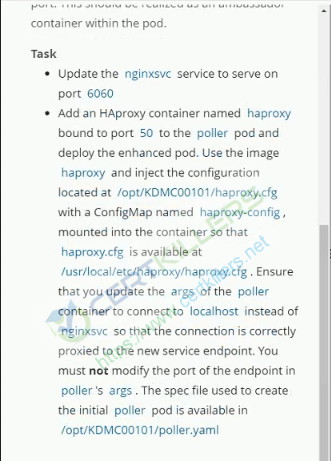
#kubectl label deployment/kdsn00101-deployment func=dmz --overwrite -n kdsn00101

#kubectl scale --replicas=4 deployment/kdsn00101-deployment -n kdsn00101 ;

#kubectl expose deployment/kdsn00101-deployment --name=cherry --port=81 --type=NodePort -n kdsn00101

13-





#kubectl edit svc nginxsvc // change the port in service

# create a configmap

kubectl create configmap haproxy-config --from-file=/opt/KDMC00101/haproxy.cfg

Add a container in /opt/KDMC00101/poller.yaml

**apiVersion: v1**

**kind: Pod**

**metadata:**

**creationTimestamp: null**

**labels:**

**run: nginx**

**name: haproxy**

**spec:**

**volumes:**

**- name: test**

**configMap:**

**name: haproxy-config**

**containers:**

**- image: haproxy**

**name: haproxy**

**resources: {}**

**ports:**

**- containerPort: 50**

**protocol: TCP**

**volumeMounts:**

**- name: test**

**mountPath: /usr/local/etc/haproxy**

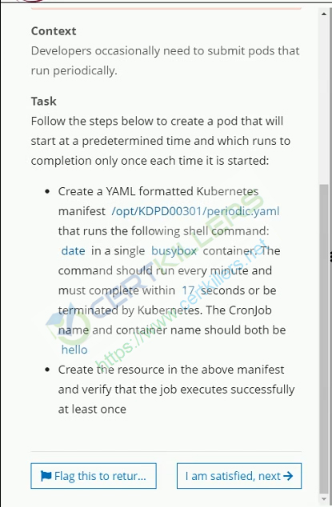
**dnsPolicy: ClusterFirst**

**restartPolicy: Always**

**status: {}**

14–





<https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/>

Ans:

**apiVersion**: batch/v1beta1

**kind**: CronJob

**metadata**:

**name**: hello

**spec**:

**schedule**: "\*/1 \* \* \* \*"

**jobTemplate**:

**spec**:

**activeDeadlineSeconds: 17**

**template**:

**spec**:

**containers**:

- **name**: hello

**image**: busybox

**args**:

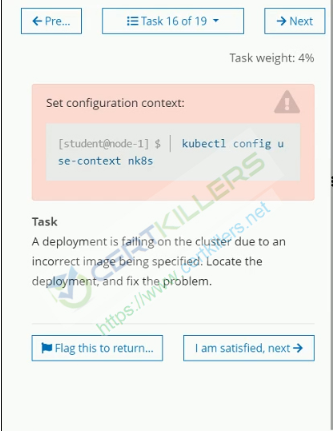
- /bin/sh

- -c

- date;

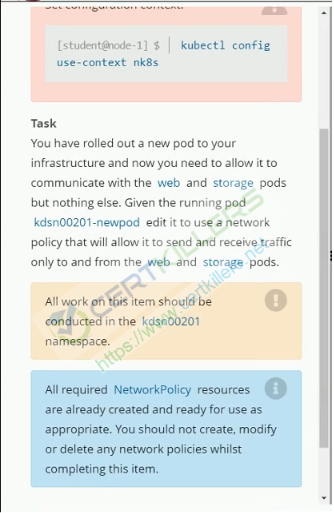
**restartPolicy**: OnFailure

15–



Ans: get deployment in all namespace edit that pod

16 –



Ans:

<https://kubernetes.io/docs/tasks/administer-cluster/declare-network-policy/>

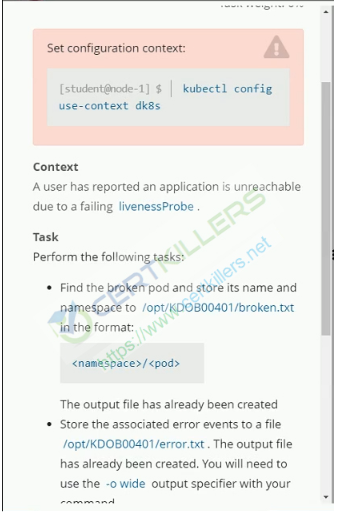
**k label pod kdsn00201-newpod role=web**

**k label pod web role=access-web**

**k label pod kdsn00201-newpod role=storage**

**k label pod storage role=access-storage**

17–



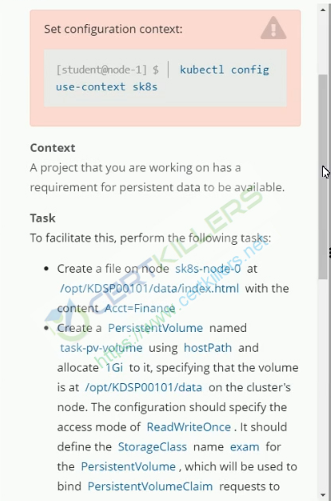


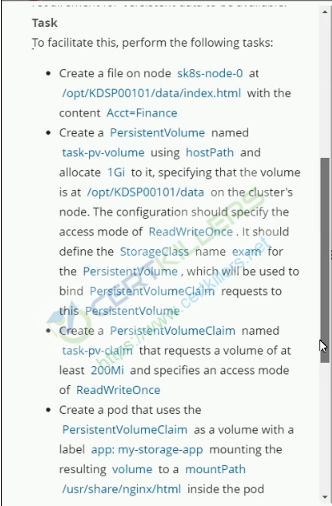
# kubectl get pod -A --field-selector status.phase!=Running -o=jsonpath='{range .items[\*]}{.metadata.namespace}/{.metadata.name}{"\n"}' > /opt/KDOB00401/broken.txt

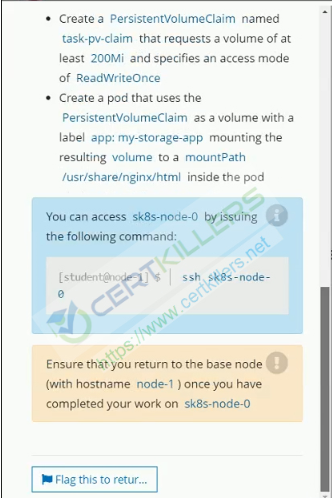
#kubectl get events -A -o wide | grep error > /opt/KDOB00401/error.txt

# to fix the issue we need to check error details

18 –







Ans:

# ssh sk8-node-0

# echo Acct=Finance > /opt/KDSP00101/data/index.html

# Now on master

# vi pv.yaml

**apiVersion**: v1

**kind**: PersistentVolume

**metadata**:

**name**: task-pv-volume

**spec**:

**capacity**:

**storage**: 1Gi

**accessModes**:[‘ReadWriteOnce’]

**storageClassName**: exam

**hostPath**:

path: /opt/KDSP00101/data

# vi pvc.yaml

**apiVersion**: v1

**kind**: PersistentVolumeClaim

**metadata**:

**name**: task-pv-claim

**spec**:

**accessModes**:

- ReadWriteOnce

**resources**:

**requests**:

**storage**: 200Mi

**storageClassName**: exam

vi pod.yaml

**apiVersion**: v1

**kind**: Pod

**metadata**:

**name**: mypod

labels:

app: my-storage-app

**spec**:

**containers**:

- **name**: myfrontend

**image**: nginx

**volumeMounts**:

- **mountPath**: "/usr/share/nginx/html"

**name**: task-pv-claim

**volumes**:

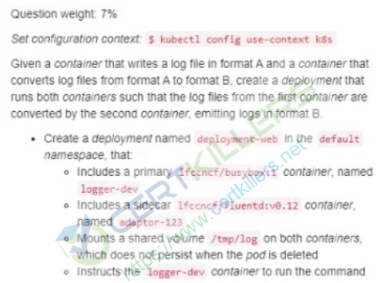
- **name**: task-pv-claim

**persistentVolumeClaim**:

**claimName**: task-pv-claim

exit

19-





Ans:

#kubectl create –f /opt/KDMC00102/fluented-configmap.yaml

# Create a deployment file like below

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**creationTimestamp: null**

**labels:**

**app: deployment-web**

**name: deployment-web**

**spec:**

**replicas: 1**

**selector:**

**matchLabels:**

**app: deployment-web**

**strategy: {}**

**template:**

**metadata:**

**creationTimestamp: null**

**labels:**

**app: deployment-web**

**spec:**

**volumes:**

**- name: vol1**

**emptyDir: {}**

**- name: vol2**

**configMap:**

**name: my-config**

**containers:**

**- image: ifccnf/busybox:1**

**name: logger-dev**

**command: ["/bin/sh"]**

**args: [ "-c", “while true; do echo i luv cncf” >> /tmp/logs/input.log; sleep 10; done]**

**resources: {}**

**volumeMounts:**

**- name: vol1**

**mountPath: /tmp/log**

**- image: ifccnf/fluentd:v0.12**

**name: adaptor-123**

**command: ["/bin/sh"]**

**args: [ "-c", “tail -f /tmp/log/input.log >> /tmp/log/output.log”; sleep 10; done]**

**volumeMounts:**

**- name: vol1**

**mountPath: /tmp/log**

**- name: vol2**

**mountPath: /fluentd/etc**

**status: {}**