CKAD_beta_exam

Basic Information:
- total 120 minutes
- total 20 programmatic questions
- questions may have different weights and different difficulity
- exam results will be delivered in three business days, otherwise you have to contact to linux foundation by yourself

Questions:

.bashrc

```
source <(kubectl completion bash)

alias kl="kubectl "
alias klc="kubectl apply -f "
alias klpod="kubectl get pods --all-namespaces -o wide"
alias kldep="kubectl get deployment --all-namespaces -o wide"
alias klall="kubectl get all --all-namespaces -o wide"
```

create namesapce

```
# delete namespace
kubectl delete namespace redis;
kubectl delete namespace pod-resources;
kubectl delete namespace kdpd00201;
kubectl delete namespace kdpd00202;
kubectl delete namespace kdpd00203;

# create namespace
kubectl create namespace redis;
kubectl create namespace pod-resources;
kubectl create namespace kdpd00201;
kubectl create namespace kdpd00202;
kubectl delete namespace kdpd00203;
```

1	Set configuration context: $ kubectl config use-context k8s A web application requires a specific version of Redis to be used as a cache. Create a Pod with the following characteristics, and leave it running when complete: The Pod should run in the redis n amespace. The namespace has already been created The name of the Pod should be re dis-server Use the lfccncf/redis image with the 3.2 tag Expose containerPort 6379 Question weight: 13%	https://kubernetes.io/docs/user-guide/walkthrough/k8s201/ # delete labels, update others

```
$ kubectl create namespace reids # for test

# p1-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: redis-server
  namespace: redis
spec:
  containers:
  - name: redis
    image: lfccncf/redis:3.2
 ports:
    - containerPort: 80
# check
$ kubectl --namespace=redis get po redis-server
NAME           READY     STATUS    RESTARTS   AGE
redis-server   1/1       Running   0          1m
$ kubectl --namespace=redis log redis-server
```

2	Set configuration context: $ kubectl config use-context k8s You are tasked to create a Secret and consume the Secret in a Pod using environment variables as follows: Create a Secret named app-secret with a key/value pair: key2/value1 Start an nginx Pod named nginx-secret using container image nginx , and add an environment variable exposing the value of the secret key key2, using BEST_VARIABLE as the name for the environment variable inside the Pod Question weight: 5%	https://kubernetes.io/docs/tasks/inject-data-application/distribute-credentials-secure/ ``` $ kubectl create secret generic app-secret --from-literal=key2='value1' # p2-pod.yaml apiVersion: v1 kind: Pod metadata: name: nginx-secret spec: containers: - name: envars-test-container image: nginx env: - name: BEST_VARIABLE valueFrom: secretKeyRef: name: app-secret key: key2 # check $ kubectl get po nginx-secret NAME READY STATUS RESTARTS AGE nginx-secret 1/1 Running 0 1m $ kubectl exec -it nginx-secret -- /bin/bash $ printenv | grep BEST_VARIABLE BEST_VARIABLE=value1 ```

3	Set configuration context: $ kubectl config use-context k8s You are required to create a Pod that requests a certain amount of CPU and memory, so it gets scheduled to a node that has those resources available. Create a Pod named nginx-resources in the pod-resources Namespace that requests a minimum of 200m CPU and 2Gi memory for its container The Pod should use the nginx image The pod-resources Namespace has already been created Question weight: 4%	https://kubernetes.io/docs/tasks/configure-pod-container/assign-memory-resource/#specify-a-memory-request-and-a-m

```
$ kubectl create namespace pod-resources # for test

# p3-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-resources
  namespace: pod-resources
spec:
  containers:
  - name: memory-demo-ctr
    image: nginx
    resources:
      requests:
        cpu: "200m"
        memory: "2Gi"
# check
$ kubectl --namespace=pod-resources get po
NAME              READY     STATUS    RESTARTS   AGE
nginx-resources   1/1       Running   0          1m
$ kubectl --namespace=pod-resources describe po
nginx-resources | grep "cpu\|memory"
  memory-demo-ctr:
      cpu:        200m
      memory:     2Gi
```

4	Set configuration context: $ kubectl config use-context k8s You are tasked to create a ConfigMap and consume the ConfigMap in a Pod using a volume mount. Please complete the following: Create a ConfigMap named some-config containing the key/value pair: key3/value4 Start a Pod named nginx-configmap containing a single container using the nginx image, and mount the key you just created into the Pod under directory /some/other/path Question weight: 5%	https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/#add-configmap-data-to-a-specific-pa

```
$ echo "key3=value4" > env.txt
$ kubectl create configmap some-config
--from-env-file=env.txt

# p4-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-configmap
spec:
  containers:
    - name: test-container
      image: nginx
      command: [ "/bin/sh", "-c", "ls /some/other/path/
      volumeMounts:
      - name: config-volume
        mountPath: /some/other/path
  volumes:
    - name: config-volume
      configMap:
        name: some-config
  restartPolicy: Never

# check
$ kubectl get po nginx-configmap
NAME              READY     STATUS      RESTARTS   AGE
nginx-configmap   0/1       Completed   0          57s
$ kubectl logs nginx-configmap
key3
```

5	Set configuration context: $ kubectl config use-context k8s A Pod is running on the cluster but it is not responding. The desired behavior is to have Kubernetes restart the pod when an endpoint returns an HTTP 500 on the /healthz endpoint. The service, liveness-pod, should never send traffic to the Pod while it is failing. Please complete the following: The application has an endpoint, /started, that will indicate if it can accept traffic by returning an HTTP 200. If the endpoint returns an HTTP 500, the application has not yet finished initialization The application has another endpoint /healthz that will indicate if the application is still working as expected by returning an HTTP 200. If the endpoint returns an HTTP 500 the application is no longer responsive Configure the liveness-pod Pod provided to use these endpoints The probes should use port 8080 Question weight: 2%	
6	Set configuration context: $ kubectl config use-context k8s You sometimes need to observe a Pod's logs, and write those logs to a file for further analysis. Please complete the following: Deploy the counterapp Pod to the cluster using the provided YAML spec file at /opt/KDOB00201/counterapp.yaml Retrieve all currently available application logs from the running Pod and store them in the file /opt/KDOB00201/log_output.txt, which has already been created Question weight: 4%	`$ view /opt/KDOB00201/counterapp.yaml` `$ kubectl get po counterapp` `$ kubectl logs -p counterapp > /opt/KDOB00201/log_output.txt` `$ less /opt/KDOB00201/log_output.txt`

7	set configuration context: $ kubectl config use-context k8s Create a new Deployment for running nginx with the following parameters: Run the Deployment in the kdpd00201 Namespace. The Namespace has already been created Name the Deployment frontend and configure with 3 replicas Configure the Pod with a container image of lfccncf/nginx:1.12.2 Set an environment variable of NGINX_PORT=8001 and also expose that port for the container above Question weight: 5%	https://kubernetes.io/docs/user-guide/walkthrough/k8s201/#deployment-management ``` $ kubectl create namespace kdpd00201 # vim p6-deploy.yaml apiVersion: apps/v1 kind: Deployment metadata: name: frontend namespace: kdpd00201 spec: selector: matchLabels: app: nginx replicas: 3 template: metadata: labels: app: nginx spec: containers: - name: nginx image: lfccncf/nginx:1.12.2 env: - name: NGINX_PORT value: "8001" # check $ kubectl --namespace=kdpd00201 get all $ kubectl --namespace=kdpd00201 describe pod/frontend-9bc77c74d-fl267 ```
8	Set configuration context: $ kubectl config use-context k8s As a Kubernetes application developer you will often find yourself needing to update a running application. Please complete the following: Update the web1 Deployment in the kdpd00202 Namespace with a maxSurge of 4 and a maxUnavailable of 2 Perform a rolling update of the web1 Deployment, changing the lfccncf/nginx image version to 1.13 Roll back the web1 Deployment to the previous version Question weight: 6%	

test file

```
$ kubectl create namespace kdpd00202

# test file
$ vim web1-deploy.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: web1
  namespace: kdpd00202
spec:
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: lfccncf/nginx:1.12.2

# check
$ kubectl --namespace=kdpd00202 get deploy web1
```

```
$ kubectl --namespace=kdpd00202 edit deployment web1
# change maxSurge of 4 and a maxUnavailable of 2
deployment "web1" edited

$ kubectl --namespace=kdpd00202 set image deployment/we
nginx=lfccncf/nginx:1.13
deployment "web1" image updated

$ kubectl --namespace=kdpd00202 get deploy web1 -o wide
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE
CONTAINERS   IMAGES               SELECTOR
web1      3         3         3            3
nginx        lfccncf/nginx:1.13   app=nginx

$ kubectl --namespace=kdpd00202 rollout undo
deployment/web1
deployment "web1" rolled back

$ kubectl --namespace=kdpd00202 get deploy web1 -o wide
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE
CONTAINERS   IMAGES                SELECTOR
web1      3         3         3            3
nginx        lfccncf/nginx:1.12.2   app=nginx
```

9	find maximum cpu usage pod name in a namespace	use top?
10	config service account, and use that account to create a pod	
11	create deployment and service	
12	logs	
13		

14	Given a container that writes a log file in format A and a container that converts log files from format A to format B, create a Deployment that runs both containers such that the log files from the first container are converted by the second container, emitting logs in format B. Create a Deployment named deployment-007 in the default namespace, that: Includes a primary lfccncf/busybox:1 container, named logger-zen Includes a sidecar lfccncf/fluentd:v0.12 container, named adaptor-123 Mounts a shared volume /tmp/log on both containers, which does not persist when the pod is deleted Instructs the logger-zen container to run the command while true; do echo i luv cncf >> /tmp/log/input.log; sleep 10; done, which should output logs to /tmp/log/input.log in plain text format, with example values: i luv cncf i luv cncf i luv cncf The adaptor-123 sidecar container should read /tmp/log/input.log and output the data to /tmp/log/output.* in Fluentd JSON format. Note that no knowledge of Fluentd is required to complete this task: all you will need Question weight: 8%	https://kubernetes.io/docs/concepts/cluster-administration/logging/#sidecar-container-with-a-logging-agent

15	Set configuration context: $ kubectl config use-context k8s Developers occasionally need to submit Pods that run periodically. Follow the steps below to create a Pod that will start at a predetermined time and which runs to completion only once each time it is started: Create a YAML formatted Kubernetes manifest /opt/KDPD00301/periodic.yaml that runs the following shell command: date in a single busybox container . The command should run every minute and must complete within 10 seconds or be terminated by Kubernetes. The CronJob name and container name should both be hello Create the resource in the above manifest and verify that the job executes successfully at least once Question weight: 4%	https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/ ``` # cronjob.yaml apiVersion: batch/v1beta1 kind: CronJob metadata: name: hello spec: activeDeadlineSeconds: 10 # add schedule: "*/1 * * * *" jobTemplate: spec: template: spec: containers: - name: hello image: busybox args: - /bin/sh - -c - date; restartPolicy: OnFailure # check ``` kubectl explain cronjob --api-version=batch/v1beta1

16	Set configuration context: $ kubectl config use-context k8s You have been tasked with scaling an existing Deployment for availability, and creating a Service to expose the Deployment within your infrastructure. Start with the Deployment named kdsn00101-deployment which has already been deployed to the namespace kdsn00101. Edit it to: Add the role=webFrontEnd key/value label to the Pod template metadata to identify the pod for the Service definition Have 4 replicas Next, create and deploy in namespace kdsn00101 a Service that accomplishes the following: Exposes the Service on TCP port 8080 Is mapped to the pods defined by the specification of kdsn00101-deployment Is of type NodePort Has a name of cherry Question weight: 6%	https://kubernetes.io/docs/concepts/services-networking/service/ https://kubernetes.io/docs/concepts/services-networking/connect-applications-service/ ``` kubectl --namespace=kdsn00101 apply -f dep.yaml kubectl --namespace=kdsn00101 edit deployment kdsn00101-deployment role:webFrontEnd # two loc replicas 4 kind: Service apiVersion: v1 metadata: name: cherry # add spec: type: NodePort selector: role: webFrontEnd # add ports: - protocol: TCP nodePort: 30000 port: 8080 targetPort: 80 ``` kubectl --namespace=kdsn00101 apply -f svc.yaml
17	Set configuration context: $ kubectl config use-context nk8s You have rolled out a new Pod to your infrastructure and now you need to allow it to communicate with the www and api pods but nothing else. Given the running pod kdsn00201 -newpod edit it to use a network policy that will allow it to send and receive traffic only to and from the www and api Pods. All work on this item should be conducted in the kdsn00201 nam espace. All required NetworkPolicy resour ces are already created and ready for use as appropriate. You should not create, modify or delete any network policies whilst completing this item. Question weight: 6%	https://kubernetes.io/docs/concepts/services-networking/network-policies/ https://kubernetes.io/docs/tasks/administer-cluster/declare-network-policy/ change pods lables

18	Set configuration context: $ kubectl config use-context dk8s A user has reported an application is unreachable due to a failing livenessProbe. The associated deployment could be running in any of the following namespaces: qa test production alan Perform the following tasks: Find the broken Pod and store its name and namespace to /opt/KDOB00401/broken.txt in the format <namespace>/<pod>. The output file has already been created Store the associated error events to a file /opt/KDOB00401/error.txt. The output file has already been created Fix the issue Question weight: 6%	https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/
19	Set configuration context: $ kubectl config use-context sk8s A project that you are working on has a requirement for persistent data to be available. To facilitate this, perform the following tasks: Create a file on node sk8s-node-0 at /opt/KDSP00101/data/index.html with the content PW=MyStuff! Create a PersistentVolume named task-pv-volume using hostPath and allocate 2Gi to it, specifying that the volume is at /opt/KDSP00101/data on the cluster's Node. The configuration should specify the access mode of ReadWriteOnce. It should define the StorageClass name storage for the PersistentVolume, which will be used to bind PersistentVolumeClaim requests to this PersistentVolume Create a PersistentVolumeClaim named task-pv-claim that requests a volume of at least 100Mi and specifies an access mode of ReadWriteOnce Create a Pod that uses the PersistentVolumeClaim as a volume with a label app: my-storage-app mounting the resulting volume to a mountPath /usr/share/nginx/html inside the Pod Hint: You can access sk8s-node-0 by issuing the following command: ssh sk8s-node-0	https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/

Ensure that you return to the base node (with hostname node-1) once you have completed your work on sk8s-node-0

Question weight: 6%

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: task-pv-volume
  labels:
    type: local
spec:
  storageClassName: k8s #update
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/home/liguohui9527/q19/data" #update

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: task-pv-claim
spec:
  storageClassName: k8s #update
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 200Mi #update

kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
  labels:               #update
    app: my-storage-app #update
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: task-pv-claim
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
```

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

kubectl apply -f pv.yaml;
kubectl get pv task-pv-volume;
kubectl apply -f pvc.yaml;
kubectl get pv task-pv-volume;
kubectl get pvc task-pv-claim;
kubectl apply -f pod.yaml;
kubectl get pod task-pv-pod;
kubectl describe po task-pv-pod;
```

Lessons:
- DO NOT USE WORK COMPUTER! Or at least turn off firewall!
- very practically questions and quite like real-word problem solving
- Time is short and stressfull. If you cannot finish a question within 10 minutes, then just skip to the next one
- create a new folder for each question, and double check your solution with the question before you executing your command
- you can view the kubernetes document during the exam, better to memerize the key operations instead of reading the documents in the exam
- at least 2 ~ 3 months dedicated working hours on kubernetes will help you to have a good understanding on all the concepts and pass the exam