source <(kubectl completion bash)

●1.Set configuration context $kubectl config use-context **k8s**

Monitor the logs of Pod **foobar** and Extract log lines corresponding to error ***unable-to-access-website*** Write them to /opt/KULM00201/foobar

监控 foobar Pod 的日志，提取 ***unable-to-access-website*** 相应的行写入到/opt/KULM00201/foobar 文件中

解答：

#kubectl logs foobar | grep '*unable-to-access-website*' > /opt/KULM00201/foobar

备注：

撤销 taint

kubectl taint node vms51.rhce.cc node-role.kubernetes.io/master-

●2.Set configuration context $kubectl config use-context **k8s**

List all PVs sorted by **name**,saving the full kubectl output to **/opt/KUCC0010/my_volumes**.

Use kubectl own functionally for sorting the output, and do not manipulate it any further.

使用 name 排序列出所有的 PV，把输出内容存储到/opt/KUCC0010/my_volumes 文件中

使用 kubectl own 对输出进行排序，并且不再进一步操作它。

解答：

#kubectl get pv --all-namespaces --sort-by={.metadata.name} > /opt/KUCC0010/my_volumes

●3.Set configuration context $kubectl config use-context **k8s**

Ensure a single instance of Pod **nginx** is running on each node of the Kubernetes cluster where **nginx** also represents the image name which has to be used. Do no override any taints currently in place.

Use **Daemonset** to complete this task and use **ds.kusc00201** as Daemonset name.

确保在 kubectl 集群的每个节点上运行一个 Nginx Pod。其中 Nginx Pod 必须使用 Nginx 镜像。不要覆盖当前环境中的任何 traints。

使用 Daemonset 来完成这个任务，Daemonset 的名字使用 ds.kusc00201。

引用：Concepts->Workloads->Controllers->DaemonSet

解答：

#cat ds.kusc00201.yaml

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: ds.kusc00201
  namespace: default
  labels:
    k8s-app: ds.kusc00201
spec:
  selector:
    matchLabels:
      name: ds.kusc00201
  template:
    metadata:
      labels:
        name: ds.kusc00201
    spec:
      containers:
      - name: nginx
        image: nginx
```

#kubectl apply -f ds.kusc00201

●4.Set configuration context $kubectl config use-context **k8s**
Perform the following tasks
Add an init container to **lumpy-koala**(which has been defined in spec file /opt/kucc00100/pod-spec-KUCC00100.yaml)
The init container should create an empty file named **/workdir/calm.txt**. If /workdir/calm.txt is not detected, the Pod should exit Once the spec file has been updated with the init container definition, the Pod should be created.
添加一个 initcontainer 到 lumpy-koala
这个 initcontainer 应该创建一个名为/workdir/calm.txt 的空文件，如果/workdir/calm.txt 没有被检测到，这个 Pod 应该更新 spec 文件并退出，这个 Pod 应该被创建
引用：Task->Configure Pod and Containers->Configure Pod Initialization(Create a Pod that has an Init Container)
Task->Configure Pod and Containers->Configure Liveness and Readiness Probes(Define a liveness command)
解答：
基础环境：

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: nginx
    volumeMounts:
    - name: workdir
      mountPath: /workdir
    livenessProbe:
      exec:
        command:
        - cat
        - /workdir/calm.txt
  initContainers:
  - name: install
    image: busybox
    command:
    - touch
    - /workdir/calm.txt
    volumeMounts:
    - name: workdir
      mountPath: /workdir
  volumes:
  - name: workdir
    emptyDir: {}
#kubectl apply -f   pod-basic.yaml
```

备注：
1.在外层容器上挂载目录，不然无法识别，
2.添加 livenessprobe 监测文件是否存在信息
3.添加 initContainer
●5.Set configuration context $kubectl config use-context **k8s**
Create a pod named **kucc4** with a single container for each of the following images running inside(there may be between 1 and 4 images specified):nginx +redis+Memcached+consul

创建一个名为 kucc4 的 Pod,其中内部运行着 nginx+redis+memcached+consul 4 个容器
引用：Concepts->Workloads->Pods->Pod Overview
找到创建 pod 的例子，修改添加 container 的内容
解答：

```
#cat kucc4.yaml
apiVersion: v1
kind: Pod
metadata:
  name: kucc4
  labels:
    app: kucc4
spec:
  containers:
  - name: nginx
    image: nginx
  - name: redis
    image: redis
  - name: memcached
    image: memcached
  - name: consul
    image: consul
#kubectl apply   -f   kucc4.yaml
```

●6.Set configuration context $kubectl config use-context **k8s**
Schedule a Pod as follows:
Name: nginx-kusc00101
Image: nginx
Node selector: disk=ssd
创建 Pod，名字为 nginx-kusc00101，镜像为 nginx，存放在 label 为 disk=ssd 的 node 上
引用：Concepts->Configuration->Assigning Pods to Nodes
解答：

```
#cat nginx-kusc00101.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-kusc00101
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
  nodeSelector:
     disk: ssd
#kubectl apply -f nginx-kusc00101.yaml
```

备注：
给 node 添加 label

```
kubectl   label   node   vms52.rhce.cc   disk=ssd
```

给 node 取消 label

```
kubectl   label   node   vms52.rhce.cc   disk-
```

●7.Set configuration context $kubectl config use-context **k8s**
Create a deployment as follows
Name: nginx-app
Using container **nginx** with version 1.11.9-alpine
The deployment should contain **3** replicas
Next, deploy the app with new version **1.12.0-**alpine by performing a rolling update and record that update.
Finally,rollback that update to the previous version **1.11.9-**alpine.
创建 deployment
名字为 nginx-app
容器采用 1.11.9 版本的 nginx
这个 deployment 包含 3 个副本
接下来，通过滚动升级的方式更新镜像版本为 1.12.0，并记录这个更新
最后，回滚这个更新到之前的 1.11.9 版本
解答：

```
#kubectl run nginx-app --image=nginx:1.11.9 --replicas=3 --record
#kubectl set image deployment nginx-app nginx-app=nginx:1.12.0 --record
#kubectl rollout history deployment nginx-app
#kubectl rollout undo deployment nginx-app --to-revision=1
```

备注：

```
kubectl rollout pause deployment    nginx-app      暂停 deployment，不记入 history
kubectl rollout resume deployment    nginx-app      恢复 deployment    重新记入 history
```

●8.Set configuration context $kubectl config use-context **k8s**
Create and configure the service **front-end-service**   so it's accessible through **NodePort/ClusterIp** and routes to the existing pod named **front-end**.
创建和配置 service，名字为 front-end-service。可以通过 NodePort/ClusterIp 开访问，并且路由到 front-end 的 Pod 上
解答：

```
#kubectl expose pod front-end    --name=front-end-service --type="NodePort" --port=80
```

●9.Set configuration context $kubectl config use-context **k8s**
Create a Pod as follows:
Name: **jenkins**
Using image: **jenkins**
In a new Kubernetes namespace named **website-frontend**
创建一个 Pod，名字为 Jenkins，镜像使用 Jenkins。在新的 namespace   website-frontend 上创建
引用：Concepts->Workloads->Pods->Pod Overview
解答：

```
#kubectl create ns website-frontend
apiVersion: v1
kind: Pod
metadata:
  name: jenkins
  labels:
    app: jenkins
spec:
  containers:
  - name: jenkins
    image: jenkins
#kubectl apply -f jenkins.yaml -n website-frontend
```

● 10.Set configuration context $kubectl config use-context **k8s**
Create a deployment spec file that will:
Launch **7** replicas of the **redis** image with the **label : app_enb_stage=dev**
Deployment name: **kual00201**
Save a copy of this spec file to **/opt/KUAL00201/deploy_spec.yaml** (or .json)
When you are done,clean up(delete) any new k8s API objects that you produced during this task
创建 deployment 的 spec 文件:
使用 redis 镜像，7 个副本，label 为 app_enb_stage=dev
deployment 名字为 kual00201
保存这个 spec 文件到/opt/KUAL00201/deploy_spec.yaml
完成后，清理(删除)在此任务期间生成的任何新的 k8s API 对象
解答：

```
#kubectl run kual00201 --image=redis --replicas=7    --labels="app_enb_stage=dev" --dry-run    -o yaml >
/opt/KUAL00201/deploy_spec.yaml
```

● 11.Set configuration context $kubectl config use-context **k8s**
Create a file **/opt/KUCC00302/kucc00302.txt** that lists all pods that implement Service **foo** in Namespace **production**.
The format of the file should be one pod name per line.
创建一个文件/opt/KUCC00302/kucc00302.txt ，这个文件列出所有的 service 为 foo ,在 namespace 为 production 的 Pod
这个文件的格式是每行一个 Pod
解答：

```
#kubectl get svc --show-labels -n production
#kubectl get pods -l name=lable-xxx1 -n production | grep -v NAME | awk '{print $1}' >> /opt/KUCC00302/kucc00302.txt
```

备注：
● 12.Set configuration context $kubectl config use-context **k8s**
Create a Kubernetes Secret as follows:
Name: **super-secret**
Credential: alice **or** username: bob
Create a Pod named **pod-secrets-via-file** using the **redis** image which mounts a secret named **super-secret** at **/secrets**
Create a second Pod named **pod-secrets-via-env** using the **redis** image, which exports **credential/username** as **TOPSECRET/CREDENTIALS**
备注：Concepts->Configuration->Secrets
解答：

```
# echo -n 'bob' |base64
Ym9i
#cat secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: super-secret
type: Opaque
data:
  username: Ym9i
#cat pod-secret-via-file.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-secret-via-file
spec:
  containers:
  - name: redis
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/secret"
      readOnly: true
  volumes:
  - name: foo
    secret:
      secretName: super-secret
#kubectl apply -f pod-secret-via-file.yaml

#cat pod-secret-via-env.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-secret-via-env
spec:
  containers:
  - name: redis
    image: redis
    env:
      - name: CREDENTIAL
        valueFrom:
          secretKeyRef:
            name: super-secret
            key: username
#kubectl apply -f pod-secret-via-env.yaml
#kubectl get pod
[root@vms51 cka]# kubectl get pod
NAME                  READY   STATUS    RESTARTS   AGE
pod-secret-via-env    1/1     Running   0          2m
pod-secret-via-file   1/1     Running   0          11m
```

● 13.Set configuration context $kubectl config use-context **k8s**

Create a pod as follows:

Name: **non-persistent-redis**

Container image: **redis**

Named-volume with name: **cache-control**

Mount path : **/data/redis**

It should launch in the **pre-prod** namespace and the volume **MUST NOT** be persistent

备注：Concept->Storage->Volumes->emptyDir(Example Pod)

解答：

```
#kubectl create ns pre-prod
#cat non-persistent-redis.yaml
apiVersion: v1
kind: Pod
metadata:
  name: non-persistent-redis
spec:
  containers:
  - image: redis
    name: redis-container
    volumeMounts:
    - mountPath: /data/redis
      name: cache-control
  volumes:
  - name: cache-control
    emptyDir: {}
#kubectl apply -f non-persistent-redis.yaml -n pre-prod
```

●14.Set configuration context $kubectl config use-context **k8s**
Scale the deployment **webserver** to **6** pods
解答：
#kubectl scale deployment webserver --replicas=6
备注：
对接了 heapster，和 HPA 联动后自动弹性伸缩
Kubectl autoscale deployment nginx-app --min=10 --max=15 --cpu-percent=80

●15.Set configuration context $kubectl config use-context **k8s**
Check to see how many **nodes** are ready (not including nodes tainted NoSchedule) and write the number to **/opt/nodenum**
解答：
#kubectl get node | grep -i ready | wc -l

●16.Set configuration context $kubectl config use-context **k8s**
From the Pod label **name=cpu-utilizer**, find pods running **high CPU workloads** and write the name of the Pod consuming **most** CPU to the file **/opt/cpu.txt** (which already exists)
解答：
#kubectl top pod -l name=cpu-utilizer
找到消耗 CPU 最高的 pod
echo 'kusc00201-5tzfk' >> /opt/cpu.txt

●17.Set configuration context $kubectl config use-context **k8s**
Create a deployment as follows
Name: **nginx-dns**
Exposed via a service : **nginx-dns**
Ensure that the service & pod are accessible via their respective DNS records
The contrainer(s) within any Pod(s) running as a part of this deployment should use the **nginx** image
Next, use the utility nslookup to look up the DNS records of the service & pod and write the output to **/opt/service.dns** and **/opt/pod.dns** respectively.
Ensure you use the **busybox:1.28** image (or earlier) for any testing,an the latest release has an upstream bug which impacts the use of nslookup.
创建一个 deployment
名字为:nginx-dns
路由服务名为：nginx-dns
确保服务和 pod 可以通过各自的 DNS 记录访问
容器使用 nginx 镜像
使用 nslookup 工具来解析 service 和 pod 的记录并写入相应的/opt/service.dns 和/opt/pod.dns 文件中
确保你使用 busybox:1.28 的镜像用来测试，最新的版本会影响 nslookup 的使用
备注：Task->Administer a Cluster->Install a Network Policy Provider->Debugging DNS Resolution

解答：

```
#cat busybox1.28.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: busybox28
  name: busybox28
spec:
  containers:
  - image: busybox:1.28
    name: busybox28
    command:
      - sleep
      - "3600000"
#kubectl   apply -f busybox28.yaml
#kubectl run nginx-dns --image=nginx
kubectl expose deployment   nginx-dns   --port=80
# kubectl exec -it busybox28 -- nslookup kubernetes.default     --测试
# kubectl get pod | grep nginx-dns
nginx-dns-8469864c5d-8cpjw     1/1         Running     0              58m
#kubectl exec -it busybox28 -- nslookup nginx-dns-8469864c5d-8cpjw
Server:      10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:        nginx-dns-8469864c5d-8cpjw
Address 1: 101.37.252.74

# kubectl get svc | grep dns
nginx-dns                ClusterIP     10.98.24.207       <none>           80/TCP            58m
[root@vms51 ~]# kubectl exec -it busybox28 -- nslookup   nginx-dns
Server:      10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:        nginx-dns
Address 1: 10.98.24.207 nginx-dns.default.svc.cluster.local
```

将 nslookup 解析出来的结果写入到相应的文件中

● 18.No configuration context change required for this item

Create a snapshot of the etcd instance running at https://127.0.0.1:2379 saving the snapshot to the file path **/data/backup/etcd-snapshot.db**

The etcd instance is running etcd version 3.2.18

The following TLS certificates/key are supplied for connecting to the server with etcdctl

CA certificate: **/opt/KUCM00302/ca.crt**

Client certificate: **/opt/KUCM00302/etcd-client.crt**

Client key: **/opt/KUCM00302/etcd-client.key**

备注：Task->Administer a Cluster->Install a Network Policy Provider->Operating etcd clusters for Kubernetes

解答：

```
#export ETCDCTL_API=3
#etcdctl --endpoint=https://127.0.0.1 --cert=/opt/KUCM00302/etcd-client.crt --cacert=/opt/KUCM00302/ca.crt
```

--key=/opt/KUCM00302/etcd-client.key snapshot save /data/backup/etcd-snapshot.db

●19.Set configuration context $kubectl config use-context **ek8s**
Set the node labelled with **name=ek8s-node-1** as unavailable and reschedule all the pods running on it
解答：
#kubectl get node --show-labels | grep name=ek8s-node-1
#kubectl drain vms52.rhce.cc

●20.Set configuration context $kubectl config use-context **wk8s**
A Kubernetes worker node,labelled with **name=wk8s-node-0** is in state NotReady.
Investigate why this is the case, and perform any appropriate steps to bring the node to a Ready state,
Ensuring that any changes are made permanent.
Hints:
You can ssh to the failed node using $ssh wk8s-node-0
You can assume elevated privileges on the node with the following command $sudo -i
解答：
是由于 kubelet 没有启动
Kubectl get node 查看一个 node 是 notReady ssh 上去
systemctl status kubelet 发现没有启动
#systemctl start kubelet; systemctl enable kubelet

●21.Set configuration context $kubectl config use-context **wk8s**
Configure the kubelet system managed service,on the node labelled with name=wk8s-node-1, to Launch a Pod containing a single container of image **nginx** named **myservice** automatically. Any spec files required should be placed in the /etc/Kubernetes/manifests directory on the node.
Hints:
You can ssh to the failed node using $ssh wk8s-node-1
You can assume elevated privileges on the node with the following command $sudo -i
备注：Concepts->Workloads->Pods->Pod Overview(Pod Templates)
解答：在标签为 name=wk8s-node-1 的 node 节点上，目录/etc/Kubernetes/manifests 下创建一个 yaml 文件，文件的内容为创建一个 pod。
#cat myservice.yaml
```
apiVersion: v1
kind: Pod
metadata:
  name: myservice
  labels:
    app: myservice
spec:
  containers:
  - name: myservice-container
    image: nginx
```

●22.Set configuration context $kubectl configuse-context **bk8s**
Given a partially-functioning Kubernetes cluser,identify symptoms of failure on the cluter.
Determine the node,the failing service and take actions to bring up the failed service and restore the health of the cluser. Ensure that any changes are made permanently.
The worker node in this cluster is labelled with **name=bk8s-node-0**
Hints:
You can ssh to the relevant nodes using $ssh $(NODE) where $(NODE) is one of bk8s-master-0 or bk8s-node-0

You can assume elevated privileges on any node in the cluster with the following command

$sudo -i

给定一个部分功能正常的 Kubernetes cluser，在 cluter 上存在失败的迹象

确定节点、故障服务，并采取行动启动故障服务并恢复 cluser 的健康状态。确保任何更改都是永久性的。

解答：

是 kube-manager-controller 没有启动，启动就做完了

```
#kubectl get cs
```

能看到 controller manager 没有启动，登陆到 master 上执行以下命令

```
#systemctl start kube-manager-controller.service
```

●23.Set configuration context $kubectl config use-context **hk8s**

Create a persistent volume with name **app-config** of capacity **1Gi** and access mode **ReadWriteOnce**. The type of volume is hostPath and its location is **/srv/app-config**

备注：Concepts->Storage->Persistent Volumes(Persistent Volumes)

解答：

```
#cat pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-config
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
     path: /srv/app-config
#kubectl apply -f pv.yaml
```