

内部资料！严禁放到互联网传播！！！

内部资料！严禁放到互联网传播！！！

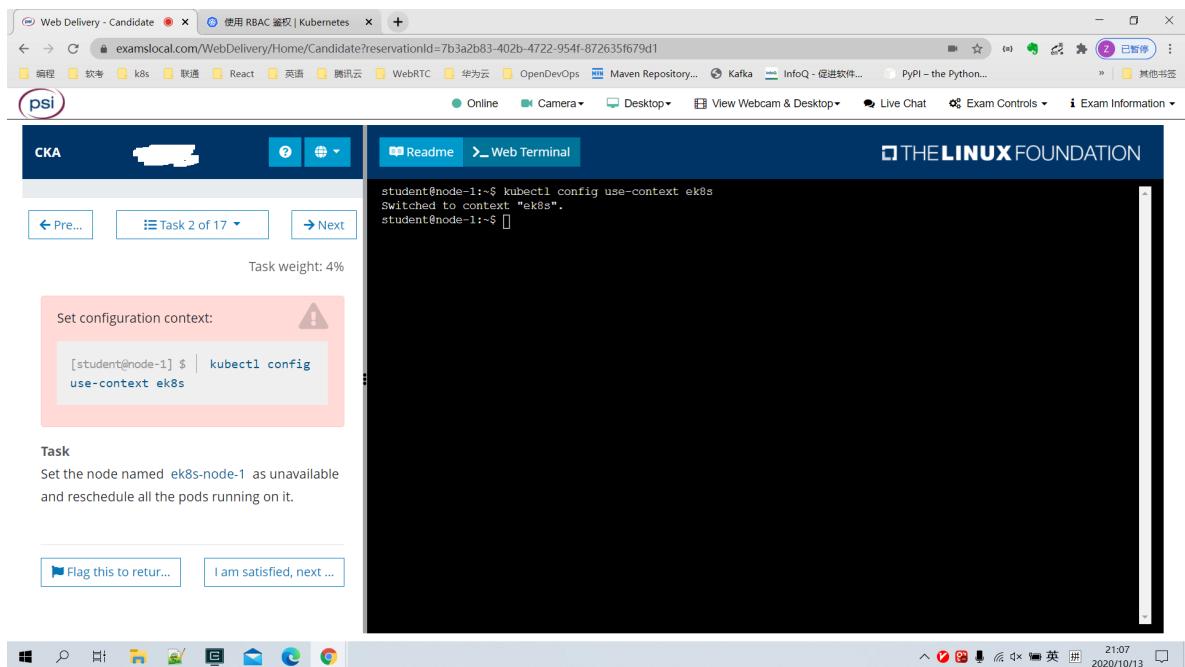
内部资料！严禁放到互联网传播！！！

考生须知

<https://www.cncf.io/certification/cka/>

<https://docs.linuxfoundation.org/tc-docs/certification/tips-cka-and-ckad>

考试窗口



基本点总结

- 考试总时长为2小时，共17道题，有一次补考机会，考试结束后36小时内通过电子邮件发送结果
- 在线考试，只使用chrome浏览器，不会使用到其他软件
- 硬件要求麦克风、摄像头，摄像头可以移动，考前至少提前15分钟进入网站，配合考官对考试环境做检查
- 考试要求房间安静不允许外人进出，咖啡厅以及开放式办公环境不可以。桌面及桌子底下不允许摆放杂物，纸、电子产品、垃圾桶等一律不允许
- 并且网络需要稳定，国外的服务器，国内访问比较慢，建议搭梯子访问<https://www.examslocal.com/ScheduleExam/Home/CompatibilityCheck> 来检测硬件及网络
- 建议使用护照来作为主身份信息，身份证需要配合其他带签名的证件一起才可以
- 除考试窗口外，只允许使用Chrome浏览器额外打开一个窗口，

- <https://kubernetes.io/docs/>
- <https://github.com/kubernetes/>
- <https://kubernetes.io/blog/> 及其子域。

这包括这些页面的所有可用语言翻译（例如<https://kubernetes.io/zh/docs/>）

- 复制和粘贴，不支持Ctrl + C/V
 - 对于Mac: ⌘+ C复制, copy + V粘贴
 - 对于Windows: Ctrl + Insert复制并Shift + Insert粘贴
- 不要用Ctrl + W，会关闭浏览器窗口，即使Ctrl + Alt + W可以代替Ctrl + W
- 考试环境为Kubernetes 1.19版本，主机操作系统为Ubuntu，共有6套环境，通过 `kubectl config use-context xxxx` 进行切换，每道题目开始会提示如何切换环境，默认是在 student@node-1终端，可以通过ssh 来访问集群里的其他节点，通过sudo -i 切换到root用户

考试技巧

- 第一步先使用kubectl命令进行自动补全

```
# 注意source后有空格,注意箭头方向
$ student@node-1# source <(kubectl completion bash)
```

- 每道题开始的时候，一定要根据提示来切换上下文
- 一定要学会使用考试界面带的记事本，因为使用kubectl edit 或者vim的时候非常难用，基本上处于不可用的情况，所以对于从官网粘贴过来的yaml格式，修改的时候一定不要在vim中修改，第一时间粘贴到记事本中，修改好之后，再整体拷贝到shell终端进行后续操作
- 考试时间只能说刚好够用，但绝对不充裕，因此，如果前面的题目没有思路或者卡住了，先进行下一道题目。先把简单的题目分数拿到手
- 考试过程中，如果因为网络问题导致终端断开，重新刷新页面，开启摄像头，打开网页共享即可
- 考试过程中，剩余时间点在：1个小时，30分钟，15分钟，5分钟，考官会给予提醒，通过live chat的方式
- 学会使用界面左侧任务中自带的粘贴，可以省很多时间，点击一下默认就复制好了，直接执行快捷键粘贴即可
- 最快原则：kubectl命令行创建 > 官网粘贴yaml > 手写yaml

考试感悟

- A同学

1. 网络策略题限制内容改变，如增加要求只允许另一个命名空间的pod访问本命名空间的pod。
2. 考试不需要使用vpn，我直接连手机的热点就可以。
3. 其它题目没有变化，少数题目名称和数量内容会改变，仔细看请题目内容按要求作答即可。
4. vim复制很正常，我几乎没用记事本

- B同学

1. 考试前注意网络情况，如果有意外不行花点手机流量
2. 考试最好直接告诉考官你要用中文，
3. 考试的时候按照考官要求检查，检查时间长不影响你考试，考试计时是从你开始考试的时候算起
4. 考试的时候一定要注意切换环境，重点
5. 考试的时候如果出现意外，确保自己能确保的，如service创建后无法curl通，要检查service中是否关联了pod的ip
6. 网断了不要慌，重新连好网络，刷新界面，开放摄像头和桌面，联系考官
7. 心态要稳，天塌了都不是事

- C同学

1. 中文考官，说的中文，跟课上说的基本一致，没要信用卡，特别问了他说不用信用卡，身份证即可，其他的就配合他操作即可
2. 开始：环境和题目基本和考题整理的前17T一致，就是部分题的名称有点变动，比如ingress 的换了名称和namespace
3. 其他就是web terminal中复制粘贴还可以，edit编辑也还好，没出现换行的问题，不过大部分还是在笔记本中编辑后copy进去的，可以直接切换root操作，在root用户下可以切换集群，开始在student用户下，etcd还原没权限，就切换到root，发现root用户可以直接切换集群
4. 还有驱逐节点这题，有点变动，需要加--delete-storage-volume 才可以驱逐
6T deployment里面没有打label
kubectl get deployment front-end --show-labels 是空的
svc里面加了 改成了front-end 测试curl是不通的
5. 想到一个很容易搞错的地方，就是在升级完集群后 一般在master节点上，下一题就是备份etcd 很容易就直接在master节点上操作了，因为题目前面标注用当前集群环境，当时在master节点备份的时候发现没有key，有点懵，后来想到升级完没退出到student，退出后又重新操作了一遍

- D同学

1. 考前准备环节
 - 1.1 提前准备好身份证件，信用卡（第一次参加考试怕证件不识别，也准备了护照）
 - 1.2 准备可以翻墙的VPN ExpressVPN 或自己购买阿里云海外服务器（推荐香港）搭建梯子ss 或者v2ray
2. 考试与面试官沟通环节
 - 2.1 距离预约考试前15分钟，进入到会议室（本人考试地点在公司会议室，确保会议室环境整洁，白板没有文字相关内容，桌面没有杂物，提前收拾一下）
 - 2.2 预约考试页面的最后一项按钮会由灰色变为绿色，点击进入PSI考试页面
 - 2.3 检查摄像头和远程桌面确保都是绿色正常状态
 - 2.4 稍等一会儿 面试官会通过livechat发来消息
 - 2.5 面试官第一次发来的消息是英文的，不要着急回复英文 （本人操作：打开了谷歌翻译，试图翻译内容，并用英文回复，在此操作的时候，面试官的提问变成了中文，接下来就是无障碍沟通配合环节了）
3. 考试题目
 - 3.1 注意每题切换上下文，仔细审题，尽可能做题操作环境保持统一用户去完成操作
 - 3.2 题目类型和练习试题一致 可能变化的是相关环境的名字 和端口号
4. 考试意外情况
 - 4.1 沟通检查环境相关情况
 - 4.1.1 （英文沟通），可以提前打开谷歌翻译页面（如果面试官没有切换语言，进行翻译后英文回复）
 - 4.2 网络环境不稳定
 - 4.2.1 可能会出现网络卡顿 打命令反应慢 （检查自己的网络）
 - 4.2.2 server断开重连，不要操作页面标签，等待一下，会重新连接到考试环境
 - 4.3 试题情况
 - 4.3.1 试题的6题 svc front-end试题
添加svc后 curl clusterip 访问不通
curl 127.0.0.1:随机端口不通
解决：

查看svc是否正常把相关pod加入进来

通过其他pod curl clusterip 发现svc可以正常使用

4.3.2 试题16 top命令

题目告知 相关文件目录已经创建，操作时发现目录并没有解决：

手动创建了相关目录和文件，把内容写入文件

5. 考试注意事项

5.1 网络环境确保正常

5.2 身份证，信用卡准备齐全

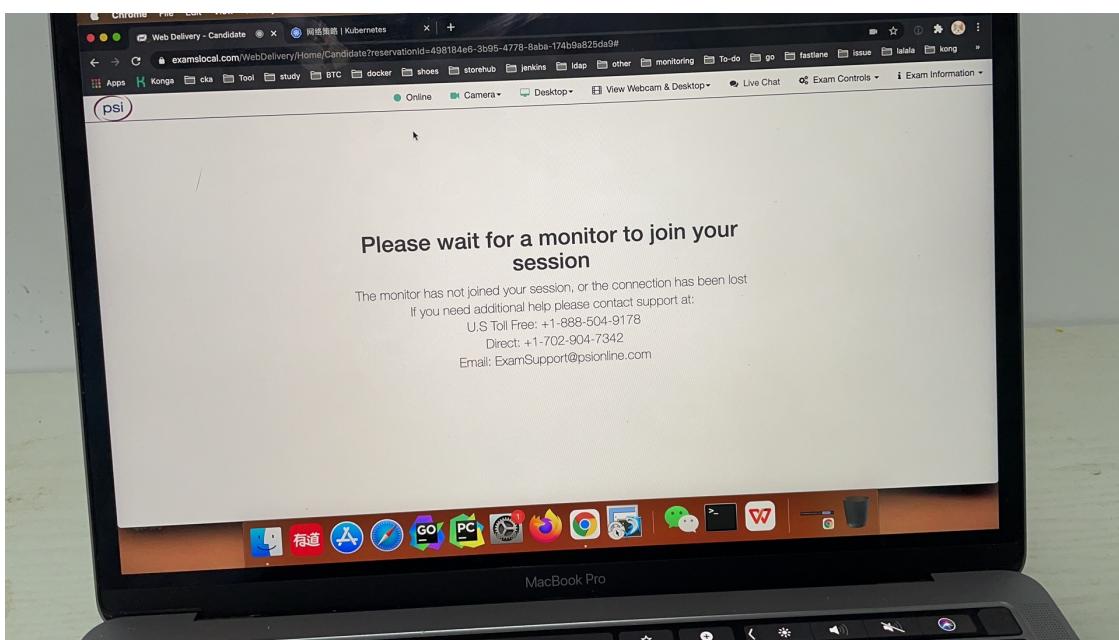
5.3 命令补全技巧可以提升速度

5.4 记事本（本人没有使用到记事本，相关yaml修改都是在vim 完成 set paste）

5.5 仔细审题，时间够用，确保正确率 13% 和 7% 的大题仔细检查

• E同学

考试时没有考官出现



总结：

1. 约考时间和中英文并无直接关系，如需中文考试，直接在开始时询问考官是否可以中文
2. service的题目，通过curl访问不通
3. 网络策略的题目，存在变化，包含两种情况：
 - 只允许同一个命名空间访问
 - 允许另一个命名空间访问
4. 第16题，实际机器并没有对应的目录及文件，需要自行创建
5. 第15题，pod没法直接进行kubectl edit，要先保存出来yaml文件再更改
6. etcd还原要使用root用户
7. student用户和root用户都可以在进行切换集群上下文，但是建议优先使用student操作，如果出现权限问题，再切换到root用户
8. 在使用vim的时候，记得执行 set paste，如果vim好用，则优先vim，否则使用记事本功能
9. 要在最终考试的环境测试好考试网络，比如酒店或者公寓中，网络自身原因，导致连接不上expressVpn

模拟练习

1. RBAC

每个问题的开头都会提供命令以确保使用正确的cluster，例如：

设置配置环境：\$ kubectl config use-context k8s

可使用如下命令通过 ssh 连接到组成每个cluster的node：

\$ ssh k8s-node-0

使用以下命令可在任何node上获取更高权限：

\$ sudo -i

完成某个node上的工作后，应先返回base node（主机名为 node-1），再尝试解答任何其他问题。不支持嵌入式 ssh。

可使用 kubectl 和相应环境配置在base node上访问任何cluster。通过 ssh 连接到 cluster成员上时，通过 kubectl 只能访问该特定cluster。

相应问题中还会提供关于连接cluster nodes的进一步说明。

请使用导航面板开始解答第 1 题。

```
# 设置命令行补全
$ source <(kubectl completion bash)

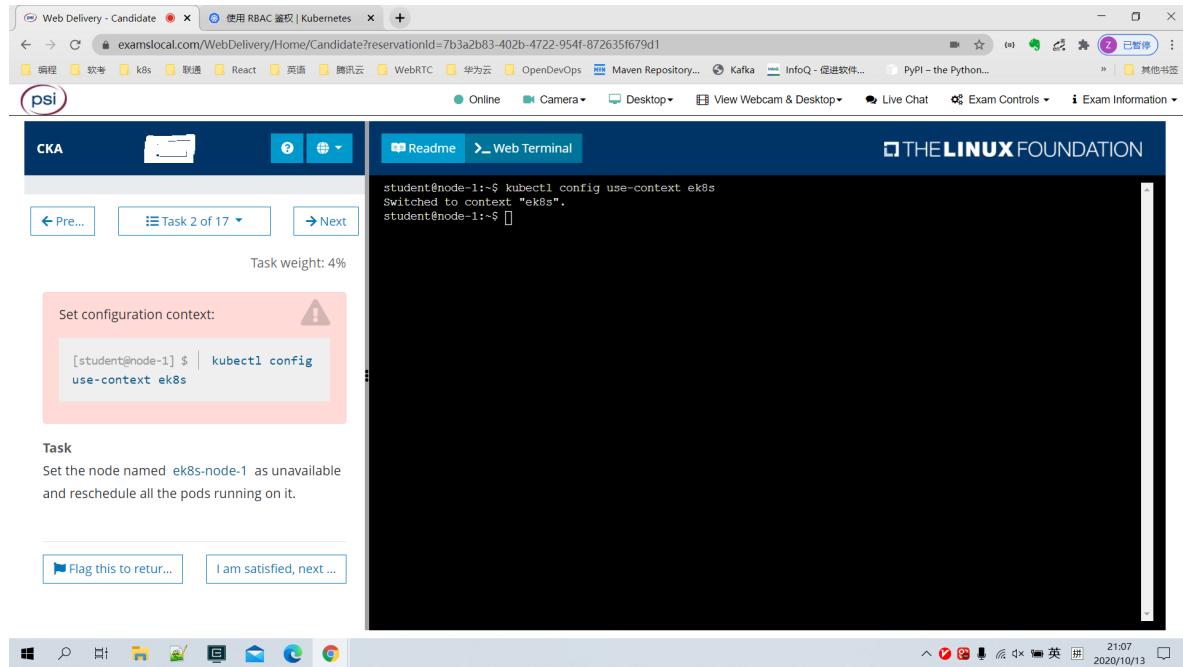
# 选择本题操作集群
$ kubectl config use-context k8s

# 生成资源清单
$ kubectl create clusterrole deployment-clusterrole --verb=create --
resource=deployments,daemonsets,statefulsets
$ kubectl -n app-team1 create serviceaccount cicd-token
$ kubectl -n app-team1 create rolebinding cicd-token-binding --
clusterrole=deployment-clusterrole --serviceaccount=app-team1:cicd-token

# 注意点
```

```
# 审查题目中，明确是要求指定的serviceaccount的权限是在整个集群范围，还是限定于某个命名空间，  
如果是整个集群，则需要使用clusterrolebinding  
$ kubectl create clusterrolebinding cicd-token-binding --clusterrole=deployment-  
clusterrole --serviceaccount=app-team1:cicd-token
```

2. 驱逐节点



```
$ kubectl config use-context ek8s

$ kubectl cordon ek8s-node-1
$ kubectl drain ek8s-node-1 --ignore-daemonsets

# 注意点：
# 1. 一定看好题目中要求驱逐的节点名称
# 2. 使用kubectl drain ek8s-node-1 -h 来查看帮助
```

3. 集群升级

Web Delivery - Candidate 使用 RBAC 鉴权 | Kubernetes

任务

Given an existing Kubernetes cluster running version 1.18.8, upgrade all of the Kubernetes control plane and node components **on the master node only** to version 1.19.0.

You are also expected to upgrade kubelet and kubectl on the master node.

Be sure to drain the master node before upgrading it and uncordon it after the upgrade.

Do not upgrade the worker nodes, etcd, the container manager, the CNI plugin, the DNS service or any other addons.

Flag this to ret... I am satisfied, nex...

THE LINUX FOUNDATION

```
get
Run 'kubectl --help' for usage.
student@node-1:~$ kubectl get nodes
NAME        STATUS   ROLES    AGE   VERSION
ek8s-master-0   Ready    master  42d   v1.19.0
ek8s-node-0     Ready    <none>  42d   v1.19.0
ek8s-node-1     Ready    <none>  42d   v1.19.0
student@node-1:~$ kubectl cordon ek8s-node-1
node/ek8s-node-1 cordoned
student@node-1:~$ kubectl get nodes
NAME        STATUS   ROLES    AGE   VERSION
ek8s-master-0   Ready    master  42d   v1.19.0
ek8s-node-0     Ready    <none>  42d   v1.19.0
ek8s-node-1     Ready,SchedulingDisabled  <none>  42d   v1.19.0
student@node-1:~$ kubectl drain ek8s-node-1 --ignore-daemonsets
error: unable to drain node "ek8s-node-1", aborting command...

There are pending nodes to be drained:
ek8s-node-1
error: cannot delete DaemonSet-managed Pods (use --ignore-daemonsets to ignore): kube-system/kube-flannel-ds-amd64-nl89h, kube-system/kube-proxy-lddw
student@node-1:~$ kubectl drain ek8s-node-1 --ignore-daemonsets
node/ek8s-node-1 already cordoned
WARNING: ignoring DaemonSet-managed Pods: kube-system/kube-flannel-ds-amd64-nl89h, kube-system/kube-proxy-lddw
evicting pod/default/nginx-6cf46d496-5vbr9 evicted
node/ek8s-node-1 evicted
student@node-1:~$ student@node-1:~$
```

Web Delivery - Candidate 使用 RBAC 鉴权 | Kubernetes

设置配置环境:

```
[student@node-1] $ | kubectl config use-context mk8s
```

任务

现有的Kubernetes集群正在运行版本 1.18.8。仅将**主节点上的所有 Kubernetes 控制平面和节点组件升级到版本 1.19.0。**

另外，在**主节点**上升级 kubelet 和 kubectl。

确保在升级之前 drain 主节点，并在升级后 uncordon 主节点。请不要升级工作节点，etcd，容器管理器，CNI插件，DNS服务或任何其他插件。

Flag this to ret... I am satisfied, nex...

THE LINUX FOUNDATION

```
student@node-1:~$ kubectl version
Client Version: version.Info{Major:"1", Minor:"18", GitVersion:"v1.19.0", GitCommit:"e19964183377d0ec2052d1f1fa930c4d7575bd50", GitTreeState:"clean", BuildDate:"2020-08-26T14:30:33Z", GoVersion:"go1.15", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"18", GitVersion:"v1.18.8", GitCommit:"9f2892aab98fe339f3bd70e3c470144299398ace", GitTreeState:"clean", BuildDate:"2020-08-13T16:04:18Z", GoVersion:"go1.13.15", Compiler:"gc", Platform:"linux/amd64"}
student@node-1:~$ kubectl get nodes
NAME        STATUS   ROLES    AGE   VERSION
mk8s-master-0   Ready    master  42d   v1.18.8
mk8s-node-0     Ready    <none>  42d   v1.18.8
student@node-1:~$ kubectl cordon mk8s-master-0
node/mk8s-master-0 cordoned
student@node-1:~$ student@node-1:~$ kubectl drain mk8s-master-0 --ignore-daemonsets
node/mk8s-master-0 already cordoned
error: unable to drain node "mk8s-master-0", aborting command...

There are pending nodes to be drained:
mk8s-master-0
error: cannot delete DaemonSet-managed Pods (use --ignore-daemonsets to ignore): kube-system/kube-flannel-ds-amd64-nd6w1, kube-system/kube-proxy-gj4gf
student@node-1:~$ kubectl drain mk8s-master-0 --ignore-daemonsets
node/mk8s-master-0 already cordoned
WARNING: ignoring DaemonSet-managed Pods: kube-system/kube-flannel-ds-amd64-nd6w1, kube-system/kube-proxy-gj4gf
evicting pod/default/nginx-5d4dcc6d6-zvg1k
evicting pod/kube-system/coredns-6bfff467f8-klbj9
evicting pod/kube-system/coredns-6bfff467f8-9kh82
pod/nginx-5d4dcc6d6-zvg1k evicted
pod/coredns-6bfff467f8-klbj9 evicted
pod|
```

```
$ kubectl config use-context mk8s

# 确认版本为1.18.8，驱逐master节点的pod
$ kubectl get no

# 登录master节点操作
$ ssh mk8s-master-0
$ sudo -i
$ kubectl drain mk8s-master-0 --ignore-daemonsets

# 升级方法一：修改镜像版本升级
$ apt-cache show kubelet | grep 1.19.0
$ apt-get install -y kubelet=1.19.0-00
$ apt-get install -y kubeadm=1.19.0-00
$ apt-get install -y kubectl=1.19.0-00

# static pod路径，默认为/etc/kubernetes/manifests
# 修改/etc/kubernetes/manifests下的 kube-apiserver.yaml, kube-controllermanager.yaml, kube-scheduler.yaml, 将image的版本改为1.19.0
```

```

# 等待kubelet拉起来pod

# 检查是否生效
$ kubectl get no

# 重要！！检测kubeadm、kubectl、kubelet的版本是否正确
$ kubectl version
$ kubeadm version
$ kubelet --version

# 升级方式二：使用kubeadm升级
# 升级kubeadm，一定用apt-get install，不要用apt-get upgrade
$ apt-cache show kubeadm | grep 1.19
$ apt-get install kubeadm=1.19.0-00
$ kubeadm version

# 升级control plan组件
$ kubeadm upgrade apply 1.19.0 --etcd-upgrade=false

# 升级kubectl、kubelet组件，一定用apt-get install，不要用apt-get upgrade
$ apt-cache show kubelet | grep 1.19.0
$ apt-get install kubelet=1.19.0-00
$ apt-get install kubectl=1.19.0-00

# 重要！！检测kubeadm、kubectl、kubelet的版本是否正确
$ kubectl version
$ kubeadm version
$ kubelet --version

# 确认系统组件是否为1.19
$ kubectl get no

# 设置master节点为可调度
$ kubectl uncordon mk8s-master-0

```

4. ETCD备份恢复

The screenshot shows a browser-based application for a Kubernetes exam. At the top, there's a navigation bar with links like 'Readme', 'Web Terminal', 'Exam Information', etc. Below it, a large terminal window displays a Linux shell session:

```

student@node-1:~$ kubectl get nodes
NAME      STATUS    ROLES   AGE     VERSION
mk8s-master-0   Ready, SchedulingDisabled   master   42d   v1.19.0
mk8s-node-0    Ready           <none>   42d   v1.18.8
student@node-1:~$ student@node-1:~$ kubectl uncordon mk8s-master-0
node/mk8s-master-0 uncordoned
student@node-1:~$ student@node-1:~$ kubectl get nodes
NAME      STATUS    ROLES   AGE     VERSION
mk8s-master-0   Ready   master   42d   v1.19.0
mk8s-node-0    Ready   <none>   42d   v1.18.8
student@node-1:~$ student@node-1:~$ 

```

To the left of the terminal, there's a sidebar with the following information:

- CKA** button
- Pr...** button
- Task 4 of 17**
- Next** button
- 问题权重: 7%**
- 此项目无需更改配置环境。**
- Task** section: "首先, 为运行在 <https://127.0.0.1:2379> 上的现有 etcd 实例创建快照并将快照保存到 /data/backup/etcd-snapshot.db。"
- 提示**: "为给定实例创建快照预计能在几秒钟内完成。如果该操作似乎挂起, 则命令可能有问题。用 **CTRL + C** 来取消操作, 然后重试。"
- 然后还原位于**: "/srv/data/etcd-snapshot-previous.db" 的现有先...

```

Readme > Web Terminal THE LINUX FOUNDATION

student@node-1:~$ kubectl get nodes
NAME      STATUS    ROLES   AGE     VERSION
mk8s-master-0   Ready, SchedulingDisabled   master   42d   v1.19.0
mk8s-node-0     Ready    <none>  42d   v1.18.8
student@node-1:~$ kubectl uncordon mk8s-master-0
node/mk8s-master-0 uncordoned
student@node-1:~$ kubectl get nodes
NAME      STATUS    ROLES   AGE     VERSION
mk8s-master-0   Ready   master   42d   v1.19.0
mk8s-node-0     Ready    <none>  42d   v1.18.8
student@node-1:~$ student@node-1:~$ 

```

Flag this to ret... I am satisfied, next...

```

# 和集群升级使用同一个上下文
$ export ETCDCCTL_API=3
$ etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/opt/KUIN00601/ca.crt --
cert=/opt/KUIN00601/etcd-client.crt --key=/opt/KUIN00601/etcd-client.key
snapshot save /data/backup/etcd-snapshot.db

# 还原, 不用加证书, 注意, 还原时一定使用root用户
$ etcdctl snapshot restore /srv/data/etcd-snapshot-previous.db

# 注意, 通常机器中会有etcdctl命令行, 如果所在机器没有, 查找etcd容器, 拷贝一份etcdctl客户端
$ docker ps |grep etcd
$ docker cp 9c52196060cd:/usr/local/bin/etcdctl /usr/bin/

```

5. 网络策略

网络策略NetworkPolicy

Set configuration context:

```
[student@node-1] $ kubectl config use-context hk8s
```

Task

Create a new NetworkPolicy named allow-port-from-namespace that allows Pods in the existing namespace internal to connect to port 8080 of other Pods in the same namespace.

Ensure that the new NetworkPolicy :

- does **not** allow access to Pods not listening on port 8080
- does **not** allow access from Pods not in namespace internal

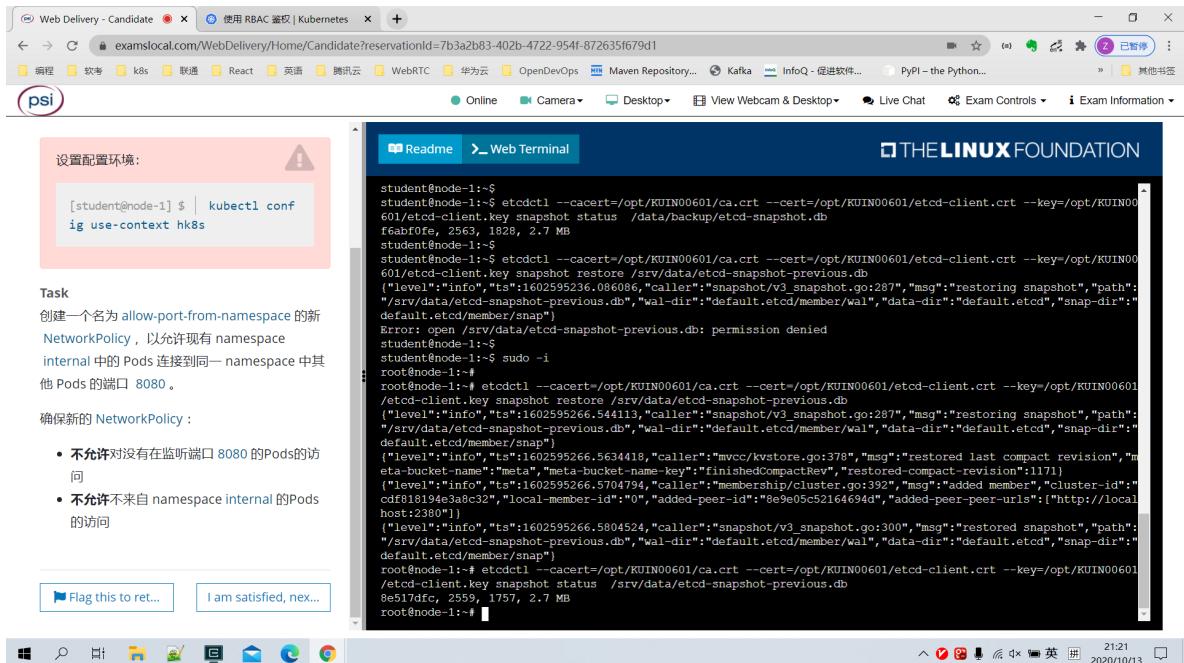
```

Readme > Web Terminal THE LINUX FOUNDATION

root@node-1:~# kubectl config use-context hk8s
Switched to context "hk8s".
root@node-1:~# kubectl get ns
NAME      STATUS   AGE
default   Active   42d
echo      Active   6h21m
external-apps Active   6h21m
internal  Active   6h21m
kube-node-lease Active   42d
kube-public Active   42d
kube-system Active   42d
root@node-1:~# kubectl get ns --show-labels
NAME      STATUS   AGE   LABELS
default   Active   42d   <none>
echo      Active   6h21m  <none>
external-apps Active   6h21m  <none>
internal  Active   6h21m  <none>
kube-node-lease Active   42d   <none>
kube-public Active   42d   <none>
kube-system Active   42d   <none>
root@node-1:~# 

```

Flag this to ret... I am satisfied, next...



```
$ kubectl config use-context hk8s
# 打开保存好的书签: https://kubernetes.io/zh/docs/concepts/services-networking/network-policies/#networkpolicy-resource
```

```
# 允许同一个命名空间下的pod访问
$ cat 5-networkpolicy-same-namespace.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-port-from-namespace
  namespace: internal
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
        - podSelector: {}
      ports:
        - protocol: TCP
          port: 8080
```

```
$ kubectl apply -f 5-networkpolicy-same-namespace.yaml
```

```
# 允许另外default命名空间下的pod访问
$ cat 5-networkpolicy-other-namespace.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-port-from-namespace
  namespace: internal
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
```

```

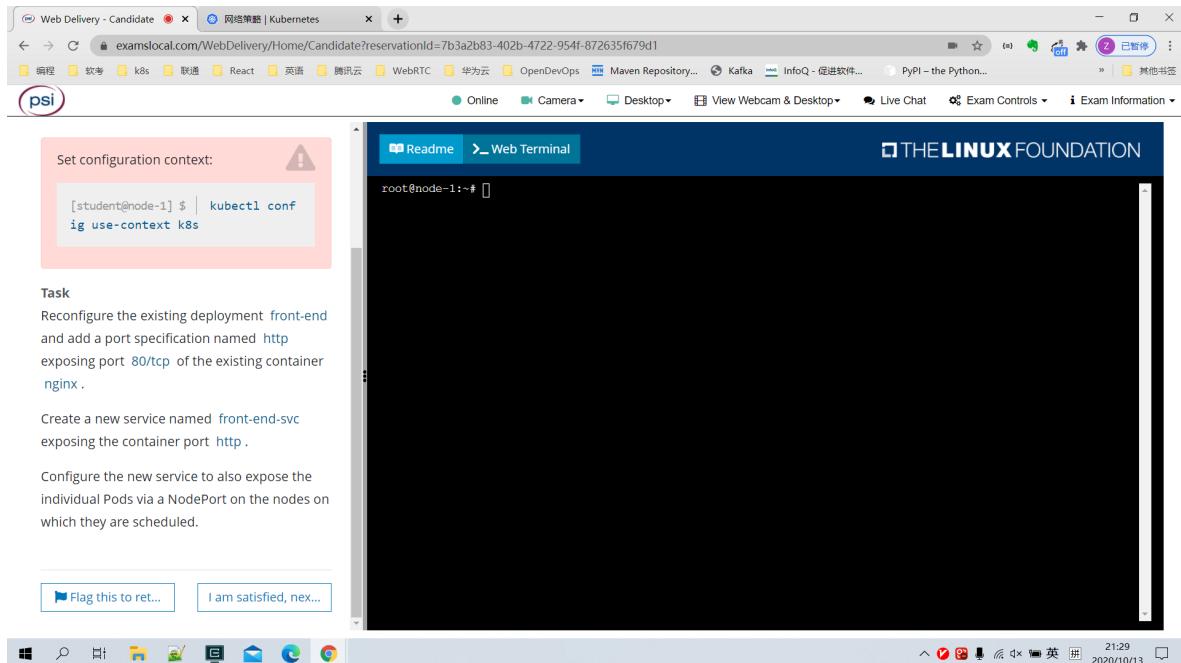
- namespaceSelector:
  matchLabels:
    project: default
ports:
- protocol: TCP
  port: 8080

```

验证，思路：

- # 1. 在internal命名空间创建deployment，名为nginx-app，pod监听在80端口
- # 2. 在default命名空间创建deployment，名为nginx-client
- # 3. 在nginx-client的pod中，访问nginx-app的pod 80端口，默认可以访问成功
- # 4. 添加网络策略1，限制internal命名空间下的pod的80端口，只允许internal命名空间下的pod访问
- # 5. 再次执行步骤3，验证是否还能访问成功
- # 6. 删掉网络策略1，创建网络策略2，允许default命名空间下的pod访问internal命名空间下的pod的80端口
- # 7. 再次执行步骤3，验证是否能访问成功

6. 创建Service



```

$ kubectl config use-context k8s
$ kubectl edit deployment front-end
...
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    ports:
    - containerPort: 80
      name: http
    resources: {}
...
$ kubectl create service nodeport front-end-svc --tcp=80:80 --dry-run=client -
oyaml>6.svc.yaml

```

```

# 6.svc.yaml, 确认selector是否可以和deployment的label对上
$ cat 6.svc.yaml
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: front-end-svc
    name: front-end-svc
spec:
  ports:
  - name: 80-80
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: front-end-svc
    type: NodePort
status:
  loadBalancer: {}

# 查看deployment的label
$ kubectl get deployment front-end --show-labels
front-end   1/1     1           1           5m53s   app=front-end

# 修改6.svc.yaml的selector
$ vi 6.svc.yaml
...

# 查看service, 访问cluster ip和nodeport, 确保结果正确
$ kubectl get svc front-end-svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
front-end-svc   NodePort    10.101.176.80   <none>        80:31493/TCP   9s
$ curl 10.101.176.80
$ curl 127.0.0.1:31493

```

7. Ingress

ingress的使用

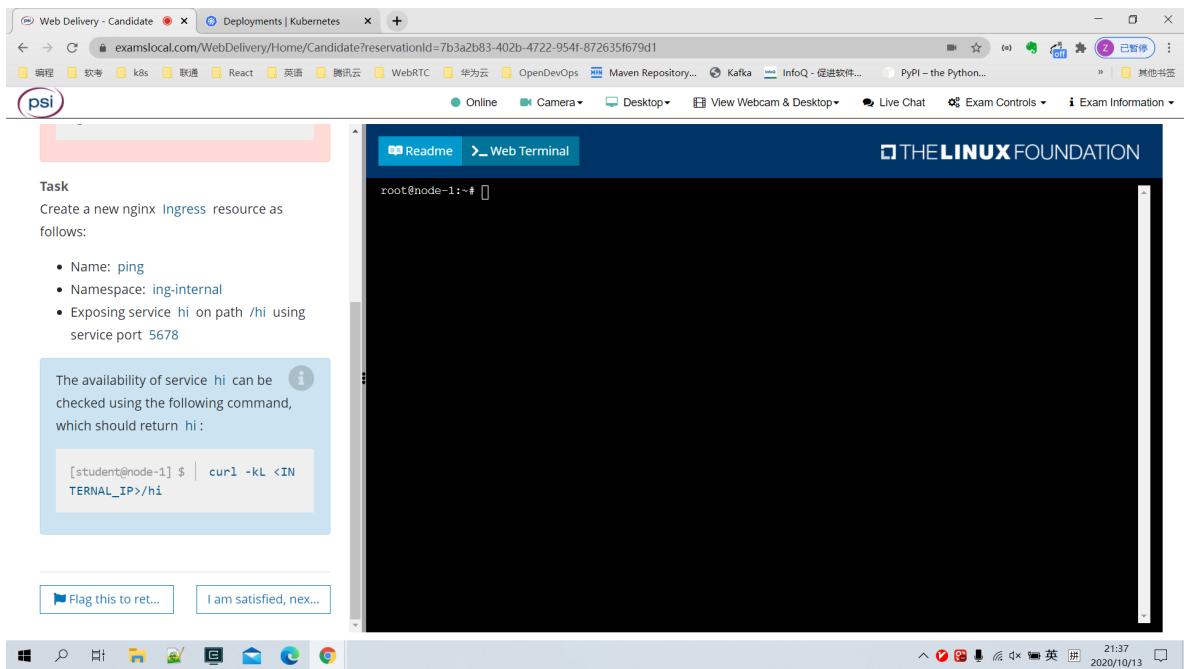
The screenshot shows a web-based exam interface with a task about Ingress. The task instructions are:

Task
Create a new nginx Ingress resource as follows:

- Name: ping
- Namespace: ing-internal
- Exposing service hi on path /hi using service port 5678

A terminal window in the foreground shows the command:

```
[student@node-1] $ | kubectl config use-context k8s
```



收藏: <https://kubernetes.io/zh/docs/concepts/services-networking/ingress/#the-ingress-resource>

```
$ cat 7.ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ping
  namespace: ing-internal
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /hi
        pathType: Prefix
        backend:
          service:
            name: hi
            port:
              number: 5678
$ kubectl apply -f 7.ingress.yaml
```

8. 扩容Deployment

This screenshot shows a web-based interface for a Kubernetes exam. At the top, there's a browser header with tabs for 'Web Delivery - Candidate' and 'Ingress | Kubernetes'. Below the header is a navigation bar with links like '编程', '软考', 'k8s', '联通', 'React', '英语', '腾讯云', 'WebRTC', '华为云', 'OpenDevOps', 'Maven Repository...', 'Kafka', 'InfoQ - 促进软件...', 'PyPI - the Python...', and '其他书签'. The main area has tabs for 'CKA' and 'Online'. A 'Web Terminal' tab is open, showing a terminal session with the following commands and output:

```
root@node-1:~# kubectl config use-context k8s
Switched to context "k8s".
root@node-1:~# kubectl get deployments.apps
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
front-end  2/2     2           2           6h42m
presentation 2/2     2           2           6h41m
root@node-1:~#
root@node-1:~# kubectl scale deployment presentation --replicas=3
deployment.apps/presentation scaled
root@node-1:~# kubectl get deployments.apps
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
front-end  2/2     2           2           6h43m
presentation 3/3     3           3           6h42m
root@node-1:~#
```

The terminal session includes a warning message: 'Set configuration context: [student@node-1] \$ | kubectl config use-context k8s'. Below the terminal, there's a 'Task' section with the instruction 'Scale the deployment presentation to 3 pods.' and two buttons: 'Flag this to return...' and 'I am satisfied, next ...'.

```
$ kubectl scale deployment presentation --replicas=3
```

9. 调度Pod

创建一个名称为 `nginx-kusc00401` 的Pod，使用镜像 `nginx`，并将pod调度到带有 `disk=spinning` 标签的节点

This screenshot shows a web-based interface for a Kubernetes exam. At the top, there's a browser header with tabs for 'Web Delivery - Candidate' and 'Ingress | Kubernetes'. Below the header is a navigation bar with links like '编程', '软考', 'k8s', '联通', 'React', '英语', '腾讯云', 'WebRTC', '华为云', 'OpenDevOps', 'Maven Repository...', 'Kafka', 'InfoQ - 促进软件...', 'PyPI - the Python...', and '其他书签'. The main area has tabs for 'CKA' and 'Online'. A 'Web Terminal' tab is open, showing a terminal session with the command:

```
root@node-1:~#
```

The terminal session includes a warning message: 'Set configuration context: [student@node-1] \$ | kubectl config use-context k8s'. Below the terminal, there's a 'Task' section with the instruction 'Schedule a pod as follows:' and a list:

- Name: `nginx-kusc00401`
- Image: `nginx`
- Node selector: `disk=spinning`

At the bottom, there are two buttons: 'Flag this to return...' and 'I am satisfied, next ...'.

```
$ kubectl run nginx-kusc00401 --restart=Never --image=nginx --dry-run=client -o yaml>9.yaml
```

```
# 编辑pod的yaml，添加nodeSelector内容，注意空格数，生成的restartPolicy: Never 行去掉
$ vim 9.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx-kusc00401
```

```

name: nginx-kusc00401
spec:
  containers:
    - image: nginx
      name: nginx-kusc00401
      resources: {}
  nodeSelector:
    disk: spinning
  dnsPolicy: clusterFirst
status: {}

$ kubectl apply -f 9.yaml

```

10. 统计节点数

The screenshot shows a web browser window titled "Assigning Pods to Nodes | Kube". The main area displays a terminal session with the following command and output:

```

root@node-1:~# kubectl config use-context k8s
Switched to context "k8s".
root@node-1:~# kubectl get nodes
NAME     STATUS   ROLES   AGE     VERSION
k8s-master-0   Ready    master  42d    v1.19.0
k8s-node-0     Ready    <none>  42d    v1.19.0
k8s-node-1     Ready    <none>  42d    v1.19.0
root@node-1:~# kubectl describe nodes | grep -i taint
Taints:        node-role.kubernetes.io/master:NoSchedule
Taints:        <none>
Taints:        <none>
root@node-1:~# echo 2 > /opt/KUSC00402/kusc00402.txt
root@node-1:~# cat /opt/KUSC00402/kusc00402.txt
2
root@node-1:~#

```

Below the terminal, there is a "Task" section with instructions:

Task
Check to see how many nodes are ready (not including nodes tainted `NoSchedule`) and write the number to
`/opt/KUSC00402/kusc00402.txt`.

At the bottom of the terminal window, there are two buttons: "Flag this to ret..." and "I am satisfied, nex...".

如图所示即可

11. 创建多容器Pod

The screenshot shows a web browser window titled "Assigning Pods to Nodes | Kube". The main area displays a terminal session with the following command and output:

```

root@node-1:~#
root@node-1:~# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
bar           1/1     Running   0          6h49m
big-corp-app  1/1     Running   0          6h48m
cpu-loader-98b9se  1/1     Running   0          6h48m
cpu-loader-ab2d3s  1/1     Running   0          6h48m
cpu-loader-k1pb9a  1/1     Running   0          6h48m
front-end-559b49fccd-kdgx9  1/1     Running   0          19m
front-end-559b49fccd-qqzjm  1/1     Running   0          19m
kucc8         4/4     Running   0          22s
nginx-kusc00401  1/1     Running   0          6m56s
presentation-764c9f588-68hjk  1/1     Running   0          9m21s
presentation-764c9f588-6hjpt  1/1     Running   0          6h51m
presentation-764c9f588-nsjs5  1/1     Running   0          6h51m
root@node-1:~#
root@node-1:~# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
bar           1/1     Running   0          6h49m
big-corp-app  1/1     Running   0          6h48m
cpu-loader-98b9se  1/1     Running   0          6h48m
cpu-loader-ab2d3s  1/1     Running   0          6h48m
cpu-loader-k1pb9a  1/1     Running   0          6h48m
front-end-559b49fccd-kdgx9  1/1     Running   0          19m
front-end-559b49fccd-qqzjm  1/1     Running   0          19m
kucc8         4/4     Running   0          27s
nginx-kusc00401  1/1     Running   0          7m1s
presentation-764c9f588-68hjk  1/1     Running   0          9m26s
presentation-764c9f588-6hjpt  1/1     Running   0          6h51m
presentation-764c9f588-nsjs5  1/1     Running   0          6h51m
root@node-1:~#

```

Below the terminal, there is a "Task" section with instructions:

Task
Create a pod named `kucc8` with a single app container for each of the following images running inside (there may be between 1 and 4 images specified):
`nginx + redis + memcached + consul`.

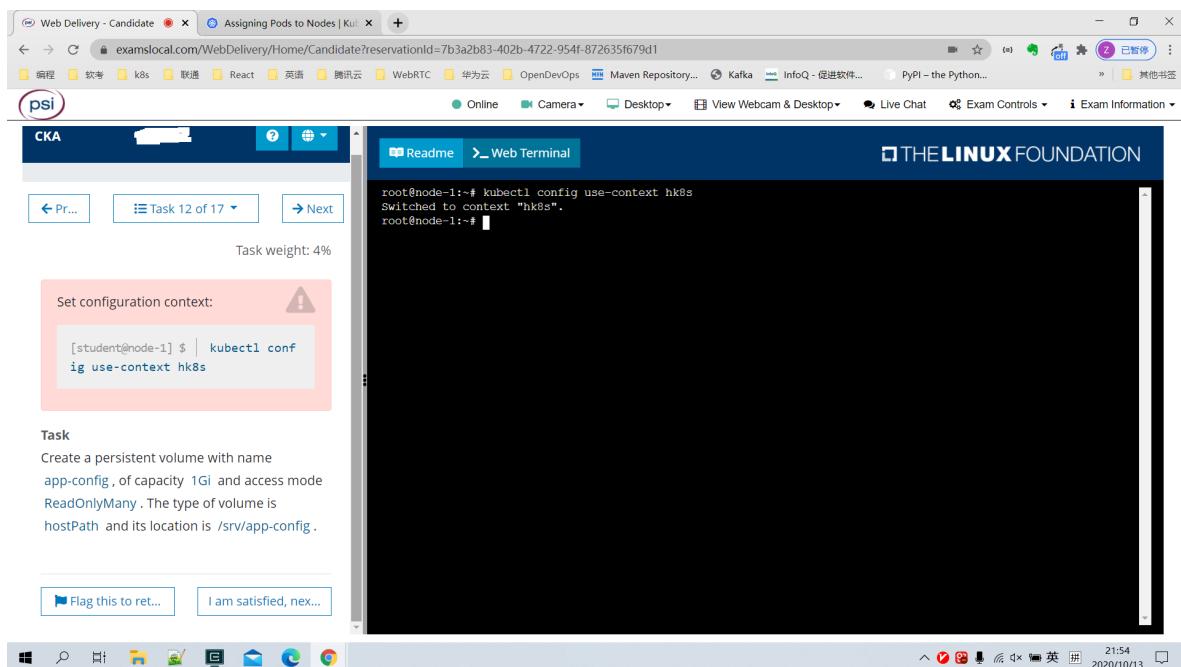
```
$ kubectl run kucc8 --restart=Never --image=nginx --dry-run=client -o yaml>11.pod.yaml
```

编辑yaml，添加另外3个container即可

```
$ cat 11.pod.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: kucc8
  name: kucc8
spec:
  containers:
    - image: nginx
      name: nginx
    - image: redis
      name: redis
    - image: memcached
      name: memcached
    - image: consul
      name: consul
```

```
$ kubectl apply -f 11.pod.yaml
```

12. PV



```
# 打开书签: https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-persistent-volume-storage/#%E5%88%9B%E5%BB%BA-persistentvolume
```

```
$ cat 12.pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-config
  labels:
    type: local
```

```

spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadOnlyMany
  hostPath:
    path: "/srv/app-config"

```

\$ kubectl apply -f 12.pv.yaml

注意点，看清楚题目中的访问模式及大小，若题目中没有要求使用storageClass，需要手动删除从网站中粘贴过来的storageClassName

13. PVC

PVC的创建及挂载

The screenshot shows a web-based interface for a Kubernetes exam. On the left, there is a configuration pane with the following content:

```

CKA [REDACTED]
? ⓘ
Pr... Task 13 of 17 Next
Task weight: 7%
Set configuration context:
[student@node-1] $ | kubectl config use-context ok8s

```

Task:

Create a new PersistentVolumeClaim :

- Name: pv-volume
- Class: csi-hostpath-sc
- Capacity: 10Mi

Create a new Pod which mounts the PersistentVolumeClaim as a volume:

Configure the new Pod to have ReadWriteOnce access on the volume.

Finally, using kubectl edit or kubectl patch expand the PersistentVolumeClaim to a capacity of 70Mi and record that change.

At the bottom, there are two buttons: "Flag this to ret..." and "I am satisfied, nex..."

The right side of the interface contains two terminal panes. The top pane shows the root prompt on node-1: "root@node-1:~#". The bottom pane shows the same root prompt. Both panes have the title "THE LINUX FOUNDATION".

```

# 打开收藏的标签 https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-persistent-volume-storage/#%E5%88%9B%E5%BB%BA-persistentvolume

# 创建pvc
$ vi 13.pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-volume
spec:
  storageClassName: csi-hostpath-sc
  accessModes:
    - ReadwriteOnce
  resources:
    requests:
      storage: 10Mi
$ kubectl apply -f 13.pvc.yaml

# 创建pod
$ vi 13.pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  volumes:
    - name: pv-volume
      persistentVolumeClaim:
        claimName: pv-volume
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: pv-volume
$ kubectl apply -f 13.pod.yaml

# 编辑大小
$ kubectl edit pvc pv-volume --record

```

14. 保存错误日志

kubectl logs命令使用

The screenshot shows a browser window with a navigation bar at the top. Below the bar is a header for 'psi' with tabs for 'CKA' and 'Web Terminal'. The main area has a 'Readme' tab and a 'Web Terminal' tab. The terminal window displays a root shell session on a node. The user runs 'kubectl config use-context k8s', which switches context. Then they run 'kubectl log bar', which fails with an error: 'Error: unknown command "log" for "kubectl"'. They then run 'kubectl logs bar' and pipe its output through 'grep' for 'unable-to-access-website'. The output shows multiple instances of this error. A pink callout box on the left says 'Set configuration context:' with the command '[student@node-1] \$ | kubectl config use-context k8s'.

```
$ kubectl logs bar | grep unable-to-access-website >>/opt/KUTR00101/bar
```

检查一下结果

```
$ cat /opt/KUTR00101/bar
```

15. sidecar容器

sidecar的使用

The screenshot shows a browser window with a navigation bar at the top. Below the bar is a header for 'psi' with tabs for 'CKA' and 'Web Terminal'. The main area has a 'Readme' tab and a 'Web Terminal' tab. The terminal window displays a root shell session on a node. It lists several pods and their status. A sidebar on the right shows a 'Live Chat' window with messages from a user asking about remaining time and a monitor replying about a 15-minute deadline. A pink callout box on the left says 'Kubernetes's built-in logging architecture (e.g., kubectl logs). Adding a streaming sidecar container is a good and common way to accomplish this requirement.' Another callout box says 'Don't modify the existing container. Don't modify the path of the log file, both containers must access it /var/log/big-corp-app.log.'

```
# 打开书签: https://kubernetes.io/zh/docs/concepts/cluster-administration/logging/#%E4%BD%BF%E7%94%A8-sidecar-%E5%AE%B9%E5%99%A8%E5%92%8C%E6%97%A5%E5%BF%97%E4%BB%A3%E7%90%86
```

```
$ kubectl get po big-corp-app -oyaml
apiVersion: v1
kind: Pod
metadata:
```

```

name: big-corp-app
spec:
  containers:
    - name: count
      image: busybox
      args:
        - /bin/sh
        - -c
        - >
          i=0;
          mkdir /var/log;
          while true;
          do
            echo "$(date) INFO $i" >> /var/log/big-corp-app.log;
            i=$((i+1));
            sleep 1;
          done

# 添加sidecar
$ kubectl get po big-corp-app -oyaml >15.pod.yaml
$ vi 15.pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: big-corp-app
spec:
  containers:
    - name: count
      image: busybox
      args:
        - /bin/sh
        - -c
        - >
          i=0;
          mkdir /var/log;
          while true;
          do
            echo "$(date) INFO $i" >> /var/log/big-corp-app.log;
            i=$((i+1));
            sleep 1;
          done
  volumeMounts:
    - name: logs
      mountPath: /var/log
    - name: busybox
      image: busybox
      args: ["/bin/sh", "-c", "tail -n+1 -f /var/log/big-corp-app.log"]
  volumeMounts:
    - name: logs
      mountPath: /var/log
  volumes:
    - name: logs
      emptyDir: {}

# 删除旧的
$ kubectl delete -f 15.pod.yaml
# 创建新的
$ kubectl apply -f 15.pod.yaml

```

16. top命令

kubectl top命令的使用

The screenshot shows a web-based Kubernetes exam interface. On the left, there's a sidebar with navigation links like '编程', '软考', 'k8s', etc. The main area has tabs for 'Readme' and 'Web Terminal'. In the terminal window, the command `$ kubectl top pod -l name=cpu-loader --sort-by='cpu' --no-headers | awk 'NR==1 {print $1}' >/opt/KUTR00401/KUTR00401.txt` is run, and its output is shown in a code block at the bottom:

```
$ kubectl top pod -l name=cpu-loader --sort-by='cpu' --no-headers | awk 'NR==1 {print $1}' >/opt/KUTR00401/KUTR00401.txt
```

17. 集群故障排查

故障排查之节点notready

The screenshot shows a web-based Kubernetes exam interface. On the left, there's a sidebar with navigation links like '编程', '软考', 'k8s', etc. The main area has tabs for 'Readme' and 'Web Terminal'. In the terminal window, the command `kubectl top pod POD_NAME --containers` is run, and its output is shown in a code block at the bottom:

```
kubectl top pod POD_NAME --containers
# Show metrics for the pods defined by label name=myLabel
kubectl top pod -l name=myLabel

Options:
-A, --all-namespaces=false: If present, list the requested object(s) across all namespaces.
Namespace in current context is ignored even if specified with --namespace.
--containers=false: If present, print usage of containers within a pod.
--no-headers=false: If present, print output without headers.
-l, --selector=: Selector (label query) to filter on, supports '=', '==', and '!='.(e.g. -l key1=value1, key2=value2)
--sort-by=: If non-empty, sort pods list using specified field. The field can be either
'cpu' or 'memory'.

Usage:
  kubectl top pod [NAME | -l label] [options]

Use "kubectl options" for a list of global command-line options (applies to all commands).
root@node-1:~# kubectl top pod --sort-by-cpu -l name=cpu-loader
Error: unknown flag: --sort-by-cpu
See 'kubectl top pod --help' for usage.
root@node-1:~# kubectl top pod --sort-by-cpu -l name=cpu-loader
NAME          CPU(cores)   MEMORY(bytes)
cpu-loader-98b9se   64m      0Mi
cpu-loader-k1pb9a   41m      0Mi
cpu-loader-ab2d3s   17m      7Mi
root@node-1:~# echo cpu-loader-98b9se > /opt/KUTR00401/KUTR00401.txt
root@node-1:~# cat /opt/KUTR00401/KUTR00401.txt
cpu-loader-98b9se
root@node-1:~#
```

```
# 连接到NotReady节点
$ ssh wk8s-node-0
# 获取权限
$ sudo -i
# 查看服务是否运行正常
$ systemctl status kubelet
#如果服务非正常运行进行恢复
$ systemctl start kubelet
#设置开机自启
$ systemctl enable kubelet
```