

2022年CKA认证题库

2022年CKA认证题库

考试说明:

题库说明:

更新说明:

第一题: 权限控制RBAC

第二题: 设置节点不可用

第三题: 升级 kubeadm

第四题: 备份还原 etcd

第五题: 配置网络策略 NetworkPolicy

第六题: 创建Service

第七题: 按要求创建 Ingress 资源

第八题: 扩容Deployment

第九题: 调度 pod 到指定节点

第十题: 统计ready 状态节点数量

第十一题: 创建多容器的pod

第十二题: 按要求创建PV

第十三题: 创建和使用PVC

第十四题: 监控pod的日志

第十五题: 添加 sidecar 容器并输出日志

第十六题: 查看 cpu 使用率最高的 pod

第十七题: 排查集群中故障节点

附录一、paste模式设置

考试说明:

1. 考试前考官会通过考试系统的共享桌面检查你电脑上的后台进程，通过摄像头确认核实身份信息，至少需要身份证+护照或者信用卡两个证件以及检查你考试周边的环境。由于考试官网在美国，服务器也在美国，导致国内无法打开或者访问系统较慢，因此建议要具备kexue上网条件。考试时浏览器只允许打开两个窗口，一个是考试系统，另外一个可以查阅k8s官方文档，所以建议提前把题库里的每道考题官网链接添加到浏览器书签里，方便快速查找。

- ✿ Ingress--CKA
- ✿ 升级 kubeadm 集群 | Kubernetes
- ✿ 为 Kubernetes 运行 etcd 集群 - Kubernetes
- ✿ sidecar
- ✿ 污点和容忍度 | Kubernetes
- ✿ etcd backup | Kubernetes
- ✿ 为 Kubernetes 运行 etcd 集群 | Kubernetes

2. 更多信息请看中国区官网: <https://training.linuxfoundation.cn/help>
3. 编辑/拷贝YAML文件的时候，建议使用paste模式，解决粘贴乱序问题。（见文档末尾附录）

题库说明:

1. 题库为CKA认证原题题库，每月都会有朋友参加考试，保证题库的准确性。题库永久更新，直到考过为止。最近配置了一套CKA认证的模拟系统，按照考试系统搭建的，已集成题库，可以直接实操练习做题，让你省心省事省力，快速有效的考取CKA认证，需要的朋友可以在淘宝店铺里搜索。

操作系统版本	kubernetes版本
Ubuntu 18.04	1.22

2. 根据店主刷题以及考过的朋友考试经分享，近一年多17道题的考点基本没变，只是改变了题目顺序和部分参数，考试就是题库里的这17道题，熟练练习考过没什么难度。

更新说明:

2021年12月:

1. 第二题 设置节点不可用参数变更: `--delete-emptydir-data`;
2. 第三题 k8s升级版本为从1.22.1到1.22.2;
3. 第四题 新增etcd还原新方法;

2022年1月:

1. 考题无变化;

第一题: 权限控制RBAC

The screenshot shows a task interface for setting up a configuration context. It includes a terminal window with the command `kubectl config use-context k8s` and a detailed task description in Chinese. The task description explains the goal: to create a new `ClusterRole` named `deployment-clusterrole` and bind it to a specific `ServiceAccount` named `cicd-token` in the `app-team1` namespace. The task also lists the resource types that the `ClusterRole` will be allowed to create: `Deployment`, `StatefulSet`, and `DaemonSet`.

设置配置环境:

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

Context

为部署流水线创建一个新的 `ClusterRole` 并将其绑定到特定的 namespace 的特定 `ServiceAccount`。

Task

创建一个名为 `deployment-clusterrole` 且仅允许创建以下资源类型的新 `ClusterRole` :

- `Deployment`
- `StatefulSet`
- `DaemonSet`

在现有的 namespace `app-team1` 中创建一个名为 `cicd-token` 的新 `ServiceAccount`。

限于 namespace `app-team1` 中, 将新的 `ClusterRole` `deployment-clusterrole` 绑定到新的 `ServiceAccount` `cicd-token`。

考题概述:

创建名称 `deployment-clusterrole` 的 `ClusterRole`, 该角色具备创建 `Deployment`、`Statefulset`、`Daemonset` 的权限, 在命名空间 `app-team1` 中创建名称为 `cicd-token` 的 `ServiceAccount`, 绑定 `ClusterRole` 到 `ServiceAccount`, 且限定命名空间为 `app-team1`。

考题解析：

需要熟悉创建 serviceaccount、clusterrole 和 rolebinding 的方法，需要限定在 ns 级别，因此最好使用 rolebinding

参考方法：

Shell | 复制代码

```
1  切换 context
2  kubectl create ns app-team1 （题库练习执行，命名空间考试系统已存在）
3  kubectl create serviceaccount cicd-token -n app-team1
4  kubectl create clusterrole deployment-clusterrole --verb=create --
   resource=deployments,statefulsets,daemonsets
5  kubectl -n app-team1 create rolebinding cicd-clusterrole --
   clusterrole=deployment-clusterrole --serviceaccount=app-team1:cicd-token
```

考点官网链接：<https://kubernetes.io/docs/reference/access-authn-authz/rbac/>

第二题：设置节点不可用

The screenshot displays a task interface for a Kubernetes exam. At the top, there are navigation buttons: '← Previous', 'Task 2 of 17', and '→ Next'. Below these, the task weight is indicated as 'Task weight: 4%' and '问题权重: 4%'. The task is titled 'Set configuration context:' and '设置配置环境:'. The command to be executed is shown in a terminal window: '[student@node-1] \$ kubectl config use-context ek8s'. Below the command, the task description is provided in English: 'Set the node named ek8s-node-1 as unavailable and reschedule all the pods running on it.' and in Chinese: '将名为 ek8s-node-1 的 node 设置为不可用，并重新调度该 node 上所有运行的 pods。'. At the bottom, there are two buttons: 'Flag this to return to la...' and 'I am satisfied, next →'.

考题概述：

设置 ek8s-node-1 节点为不可用、重新调度该节点上的所有 pod

考题解析：

cordon节点，drain 节点，需要忽略 daemonsets 并清除 local-data，否则可能无法驱逐 pod

参考方法：

Shell | 复制代码

- 1 切换 context
- 2 `kubectl cordon ek8s-node-1`
- 3 `kubectl drain ek8s-node-1 --ignore-daemonsets --delete-emptydir-data --force`
- 4 完成后一定要通过 `get nodes` 加以确认

考点官网链接：<https://kubernetes.io/zh/docs/tasks/administer-cluster/safely-drain-node/>

第三题：升级 kubeadm

Task

Given an existing Kubernetes cluster running version 1.20.0, upgrade all of the Kubernetes control plane and node components **on the master node only** to version 1.20.1.

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

You can `ssh` to the master node using:

```
[student@node-1] $ ssh mk8s-master-0
```

You can assume elevated privileges on the master node with the following command:

```
[student@mk8s-master-0] $ sudo -i
```

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

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

Task

现有的 Kubernetes 集群的运行版本 1.22.1。

仅将 **master** 节点上的所有 Kubernetes 控制平面和节点组件升级到版本 1.22.2。

确保在升级之前 drain master 节点，并在升级后 uncordon master 节点。

可使用以下命令通过 `ssh` 连接到 master 节点：

```
[student@node-1] $ ssh mk8s-master-0
```

可使用以下命令在该 master 节点上获取更高权限：

```
[student@mk8s-master-0] $ su do -i
```

另外，需要在 master 节点上升级 `kubelet` 和 `kubctl`。

请不要升级 工作节点，`etcd`，

考题概述：

升级 master 节点为1.22.2，升级前确保drain master 节点，不要升级worker node 、容器 manager、 etcd、 CNI插件、DNS 等内容；

考题解析：

首先 cordon、drain master节点，其次升级 kubeadm 并 apply 到1.22.2版本，升级 kubelet 和 kubctl

参考方法：

Shell | 复制代码

```
1  切换 context
2  kubectl get nodes
3  ssh mk8s-master-0
4  kubectl cordon mk8s-master-0
5  kubectl drain mk8s-master-0 --ignore-daemonsets --force
6  apt-mark unhold kubeadm kubectl kubelet
7  apt-get update && apt-get install -y kubeadm=1.22.2-00 kubelet=1.22.2-00
  kubectl=1.22.2-00
8  apt-mark hold kubeadm kubectl kubelet
9  kubeadm upgrade plan
10 kubeadm upgrade apply v1.22.2 --etcd-upgrade=false
11 systemctl daemon-reload && systemctl restart kubelet
12 //kubectl -n kube-system rollout undo deployment coredns 有些朋友建议rollout
  coredns,
13 kubectl uncordon mk8s-master-0
```

注意：随着K8S版本的更新，升级的版本号也会有变化，只需调整升级命令对应的版本号即可；

考点官网链接：<https://kubernetes.io/zh/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/>

第四题：备份还原 etcd



Task

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 `/srv/data/etcd-snapshot.db`。

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。

然后通过位于 `/data/backup/etcd-snapshot-previous.db` 的先前准备的快照进行还原。

提供了以下 TLS 证书和密钥，以通过 etcdctl 连接到服务器。

- CA 证书: `/opt/KUIN00601/ca.crt`
- 客户端证书: `/opt/KUIN00601/etcd-client.crt`
- 客户端密钥: `/opt/KUIN00601/etcd-client.key`

考题概述：

备份 <https://127.0.0.1:2379> 上的 etcd 数据到 `/var/lib/backup/etcd-snapshot.db`，使用之前的文件 `/data/backup/etcd-snapshot-previous.db` 还原 etcd，使用指定的 `ca.crt`、`etcd-client.crt`、`etcd-client.key`

考题解析：

备份 etcd 到指定目录、从指定备份文件还原 etcd

参考方法：

```
1  备份:
2  ETCDCTL_API=3 etcdctl --endpoints https://172.0.0.1:2379 --
   cacert=/opt/xxx/ca.crt --cert=/opt/xxx/etcd-client.crt --
   key=/opt/xxx/etcd-client.key snapshot save /var/lib/backup/etcd-
   snapshot.db
3  还原:
4  ETCDCTL_API=3 etcdctl --endpoints https://172.0.0.1:2379 --
   cacert=/opt/xxx/ca.crt --cert=/opt/xxx/etcd-client.crt --
   key=/opt/xxx/etcd-client.key snapshot restore /data/backup/etcd-snapshot-
   previous.db
5  还原成功后, 最好通过 get nodes 确定集群状态是正常的
6
7  12月94分通过的朋友, etcd还原分享了一个思路, 经测试可行, 建议使用以下方式还原。(大概的测
   试方法: 可以在etcd备份后单独创建一个pod容器, 然后使用下面方法还原etcd后, get pod可以看
   到备份后创建的pod是没有恢复出来的。 )
8
9  1. 首先先将etcd、api停止了, 移动静态pod文件后, 过了一会容器会自动停止,
10 mv /etc/kubernetes/manifests /etc/kubernetes/manifests.bak
11 2. 备份一下原来etcd的文件夹
12 mv /var/lib/etcd /var/lib/etcd.bak
13 3. 恢复数据
14 ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 snapshot restore
   /data/backup/etcd-snapshot-previous.db --data-dir=/var/lib/etcd
15 4. 启动etcd、api容器, 把静态pod文件夹移回来 过一会就可以启动了
16 mv /etc/kubernetes/manifests.bak /etc/kubernetes/manifests
17 5. 验证集群、pod资源状态
18 kubectl get nodes
19 kubectl get pods
20
```

考点官网链接: <https://kubernetes.io/zh/docs/tasks/administer-cluster/configure-upgrade-etcd/>

第五题：配置网络策略 NetworkPolicy

Task weight: 7%

Set configuration context:

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

Task

Create a new **NetworkPolicy** `allow-port-from-namespace` in the existing namespace `fubar`.

Ensure that the new **NetworkPolicy** allows Pods in namespace `my-app` to connect to port `80` of Pods in namespace `fubar`.

Further ensure that the new **NetworkPolicy**:

- does **not** allow access to Pods, which don't listen on port `80`
- does **not** allow access from Pods, which are not in namespace `my-app`

Task

在现有的 namespace `fubar` 中创建一个名为 `allow-port-from-namespace` 的新 **NetworkPolicy**。

确保新的 **NetworkPolicy** 允许 namespace `corp-net` 中的 Pods 连接到 namespace `fubar` 中的 Pods 的端口 `8080`。

进一步确保新的 **NetworkPolicy**：

- 不允许对没有在监听端口 `8080` 的 Pods 的访问
- 不允许非来自 namespace `corp-net` 中的 Pods 的访问

考题概述：

在命名空间 `fubar` 中创建网络策略 `allow-port-from-namespace`，只允许 ns `my-app` 中的 pod 连上 `fubar` 中 pod 的 80 端口，注意:这里有 2 个 ns，一个为 `fubar`(目标pod的ns)，另外一个为 `my-app`(访问源pod的ns)

考题解析：

复制官网 [services-networking/network-policies](https://kubernetes.io/docs/concepts/services-networking/network-policies) 中的案例，删掉不必要的部分，设置网络策略所属的 ns 为 `fubar`，端口为 80，设置 namespaceSelector 为源ns `my-app` 的 labels

参考方法：

```
1  # kubectl get ns --show-labels
2
3  apiVersion: networking.k8s.io/v1
4  kind: NetworkPolicy
5  metadata:
6    name: allow-port-from-namespace
7    namespace: fubar
8  spec:
9    podSelector:
10     matchLabels: {}
11    policyTypes:
12     - Ingress
13    ingress:
14     - from:
15       - namespaceSelector:
16         matchLabels:
17           kubernetes.io/metadata.name: my-app # my-app命名空间标签
18       podSelector: # 此处podSelector前不要加 - ，加了则表示fubar 中的pod都可以
        访问fubar的80端口
19         matchLabels: {}
20     ports:
21     - protocol: TCP
22       port: 80
```

考点官网链接: <https://kubernetes.io/zh/docs/concepts/services-networking/network-policies/#networkpolicy-resource>

第六题：创建Service

Set configuration context:

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

Task

Reconfigure the existing deployment `front-end` and add a port specification named `http` exposing port `80/tcp` of the existing container `nginx`.

Create a new service named `front-end-svc` exposing the container port `http`.

Configure the new service to also expose the individual Pods via a NodePort on the nodes on which they are scheduled.

Flag this to return to la...

I am satisfied, next →

设置配置环境:

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

Task

请重新配置现有的 deployment `front-end` 并添加名为 `http` 的端口规范来公开现有容器 `nginx` 的端口 `80/tcp`。

创建一个名为 `front-end-svc` 的新服务，以公开容器端口 `http`。

配置此服务，以通过各个 Pods 所在的节点上的 NodePort 来公开它们。

考题概述:

重新配置已有的 deployment `front-end`，添加一个名称为 `http` 的端口，暴露80/TCP，创建名称为 `front-end-svc` 的 service，暴露容器的 `http` 端口，配置service 的类别为NodePort

考题解析:

按照需要edit deploy，添加端口信息，通过 `expose` 命令使用 NodePort 的方式暴露端口

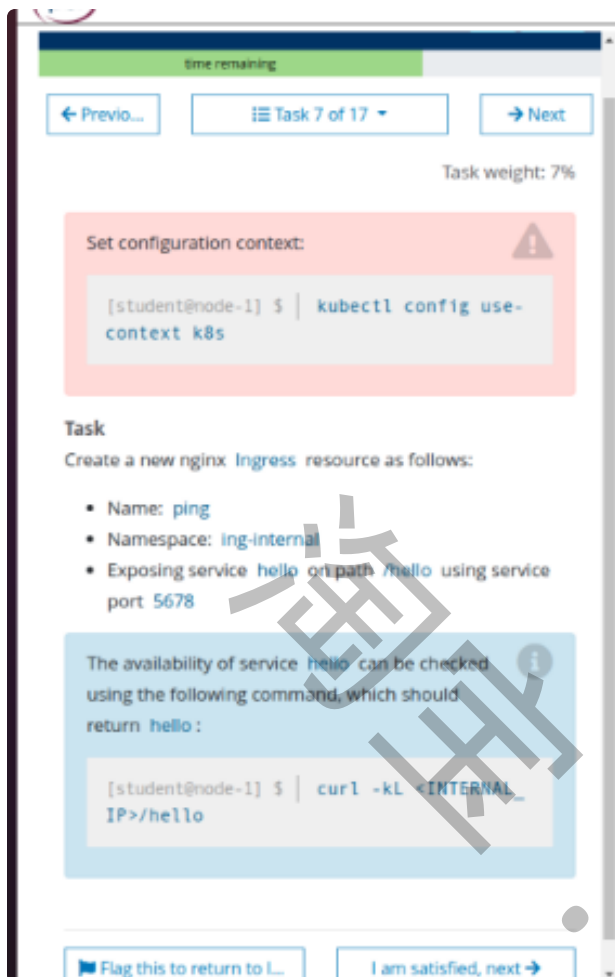
参考方法:

```
1  #题库练习创建deploy例子（考试系统deploy已存在）
2
3  apiVersion: apps/v1
4  kind: Deployment
5  metadata:
6    labels:
7      app: front-end
8      name: front-end
9  spec:
10   replicas: 1
11   selector:
12     matchLabels:
13       app: front-end
14   template:
15     metadata:
16       labels:
17         app: front-end
18     spec:
19       containers:
20       - image: nginx:1.21
21         name: nginx
22         ports:
23         - containerPort: 80
24           name: http
25           protocol: TCP
```

```
1 1) edit front-end , 在containers 中添加如下内容
2 kubectl edit deployment front-end
3 ports:
4 - name: http
5   protocol: TCP
6   containerPort: 80
7
8 2) expose 对应端口
9 kubectl expose deployment front-end --type=NodePort --port=80 --target-
  port=http --name=front-end-svc
10
11 验证:
12 # kubectl get svc
13 NAME                TYPE                CLUSTER-IP          EXTERNAL-IP          PORT(S)
14 front-end-svc        NodePort            10.105.156.36       <none>                80:32518/TCP
15 root@k8s-master:~# curl 10.105.156.36
16
```

考点官网链接: <https://kubernetes.io/docs/concepts/services-networking/service/>

第七题: 按要求创建 Ingress 资源



问题权重: 7%

设置配置环境:

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

Task

如下创建一个新的 nginx Ingress 资源:

- 名称: ping
- Namespace: ing-internal
- 使用 服务 端口 5678 在路径 /hi 上公开服务 hi

可以使用以下命令检查 服务 hi 的可用性, 该命令应返回 hi :

```
[student@node-1] $ | curl -kL
<INTERNAL_IP>/hi
```

考题概述:

创建一个新的 Ingress 资源, 名称 ping, 命名空间 ing-internal, 使用 /hello 路径暴露服务 hello 的 5678 端口

考题解析:

拷贝官文的 yaml 案例, 修改相关参数即可, 设置成功后需要通过 curl -kL <INTERNAL_IP>/hello 来测试

参考方法:

```

1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4    name: ping
5    namespace: ing-internal
6    annotations:
7      nginx.ingress.kubernetes.io/rewrite-target: /
8  spec:
9    rules:
10   - http:
11     paths:
12     - path: /hello
13       pathType: Prefix
14     backend:
15       service:
16         name: hello
17         port:
18           number: 5678
19  创建成功后，通过get ingress 查看ingress的内外IP，然后通过提供的 curl 测试 ingress
    是否正确，正常情况下是会输出 hello 的
20  root@k8s-master:~# kubectl get ingress -n ing-internal
21  NAME      CLASS      HOSTS      ADDRESS      PORTS      AGE
22  ping      <none>     *          192.168.123.151,192.168.123.152      80        109s
23  root@k8s-master:~# curl -kl 192.168.123.151/hello
24  hello

```

考点官网链接: <https://kubernetes.io/zh/docs/concepts/services-networking/ingress/#the-ingress-resource>

第八题：扩容Deployment



考题概述:

扩容 deployment guestbook 为6个pod

考题解析:

调整 replicas 为 6 即可，送分题

参考方法:

- 1 切换 context
- 2
- 3 `kubectl scale deployment --replicas=6 guestbook`

Shell 复制代码

第九题：调度 pod 到指定节点

time remaining

← Previous

Task 9 of 17

→ Next

Task weight: 4%

问题权重: 4%

Set configuration context:

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

Task

Schedule a pod as follows:

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

Flag this to return to la...

I am satisfied, next →

设置配置环境:

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

Task

按如下要求调度一个 pod:

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

考题概述:

创建pod名称nginx-kusc0041, 镜像nginx, 调度该pod到disk=ssd的节点上

考题解析:

拷贝官方案例, 修改下 pod 名称和镜像, 删除多余的部分即可, 送分题

参考方法:

YAML 复制代码

```
1  切换 context
2
3  apiVersion: v1
4  kind: Pod
5  metadata:
6    name: nginx-kusc0041
7  spec:
8    containers:
9      - name: nginx
10        image: nginx
11    nodeSelector:
12      disk: ssd
```

考点官网链接: <https://kubernetes.io/zh/docs/concepts/scheduling-eviction/assign-pod-node/>

第十题：统计ready 状态节点数量

← Previous

Task 10 of 17

→ Next

← Pr... Task 10 of 17 → N...

Task weight: 4% 问题权重: 4%

Set configuration context:

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

设置配置环境:

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

Task

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

Task

检查有多少 nodes 已准备就绪（不包括被打上 Taint: NoSchedule 的节点），并将数量写入 /opt/KUSC00402/kusc00402.txt 。

Flag this to return to la...

I am satisfied, next →

考题概述：

统计ready状态节点 要求不包括NoSchedule的节点

考题解析：

describe node过滤NoSchedule的节点，统计数量输入指定文档即可，送分题

参考方法：

```
1 kubectl describe nodes | grep -i Taints | grep -i -v NoSchedule | wc -l
2 echo number > /path/file
```

考点官网链接：<https://kubernetes.io/zh/docs/concepts/scheduling-eviction/taint-and-toleration/>

第十一题：创建多容器的pod

← Previous

Task 11 of 17

→ Next

Task weight: 4%

Set configuration context:

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

Task

Create a pod named `kucc1` with a single app container for each of the following images running inside (there may be between 1 and 4 images specified): `nginx + redis`.

Flag this to return to la...

I am satisfied, next →

设置配置环境:

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

Task

按如下要求调度一个 Pod:

- 名称: `kucc1`
- app containers: 2
- container 名称/images:
 - `nginx`
 - `memcached`

考点概述:

创建名称为kucc1的pod，pod中运行nginx和redis两个容器

考点解析:

dry-run 一个pod，多追加一个镜像即可，送分题

参考方法:

Shell 复制代码

```
1 kubectl run kucc1 --image=nginx --dry-run=client -oyaml > 11.yaml
```

YAML 复制代码

```

1 # 参考答案
2 apiVersion: v1
3 kind: Pod
4 metadata:
5   name: kucc1
6 spec:
7   containers:
8     - name: nginx
9       image: nginx
10    - name: redis
11      image: redis

```

第十二题: 按要求创建PV

← Previous

Task 12 of 17

→ Next

Task weight: 4%

Set configuration context:

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

Task

Create a persistent volume with name `app-config`, of capacity `2Gi` and access mode `ReadWriteMany`. The type of volume is `hostPath` and its location is `/srv/app-config`.

Flag this to return to later

I am satisfied, next →

← Pr...

Task 12 of 17

→ N...

问题权重: 4%

设置配置环境:

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

Task

创建名为 `app-config` 的 persistent volume, 容量为 `2Gi`, 访问模式为 `ReadWriteOnce`。volume 类型为 `hostPath`, 并且位于 `/srv/app-config`。

考题概述:

创建一个名为`app-config`的PV, PV的容量为`2Gi`, 访问模式为`ReadWriteMany`, volume的类型为`hostPath`, pv映射的`hostPath`为`/srv/app-config`目录

考题解析:

直接从官方拷贝合适的案例, 修改参数, 然后设置 `hostPath`为`/srv/app-config`即可

参考方法:

```
1  切换 context
2
3  apiVersion: v1
4  kind: PersistentVolume
5  metadata:
6    name: app-config
7  spec:
8    capacity:
9      storage: 2Gi
10   accessModes:
11     - ReadWriteMany
12   hostPath:
13     path: /srv/app-config
```

考点官网链接: <https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-persistent-volume-storage/>

第十三题: 创建和使用PVC

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:

- Name: `web-server`
- Image: `nginx`
- Mount path: `/usr/share/nginx/html`

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.

Task

创建一个新的 `PersistentVolumeClaim` :

- 名称: `pv-volume`
- Class: `csi-hostpath-sc`
- 容量: `10Mi`

创建一个新的 Pod, 来将

`PersistentVolumeClaim` 作为 volume 进行挂载:

- 名称: `web-server`
- Image: `nginx`
- 挂载路径: `/usr/share/nginx/html`

配置此新 Pod, 使得对上述 volume 具有 `ReadWriteOnce` 权限。

最后, 使用 `kubectl edit` 或 `kubectl patch` 将 `PersistentVolumeClaim` 的容量扩展为 `70Mi`, 并记录此更改。

考题概述:

使用指定storageclass `csi-hostpath-sc`创建一个名称为`pv-volume`的 pvc, 容量为`10Mi`
创建名称为`web-server`的pod, 将nginx 容器的`/usr/share/nginx/html`目录使用该pvc挂载
将上述pvc的大小从`10Mi`更新为`70Mi`, 并记录本次变更;

考题解析:

根据官方文档拷贝一个PVC, 修改参数, 复制官网nginx 的pod, 然后添加volumeMounts和volume

参考方法:

```
1  切换 context
2
3  # vim 13.1.yaml
4
5  apiVersion: v1
6  kind: PersistentVolumeClaim
7  metadata:
8    name: pv-volume
9  spec:
10   accessModes:
11     - ReadWriteOnce
12   resources:
13     requests:
14       storage: 10Mi
15     storageClassName: csi-hostpath-sc
16
17 vim 13.2.yaml
18 apiVersion: v1
19 kind: Pod
20 metadata:
21   name: web-server
22 spec:
23   containers:
24     - name: nginx
25       image: nginx
26       volumeMounts:
27         - mountPath: "/usr/share/nginx/html"
28           name: mypv
29   volumes:
30     - name: mypv
31       persistentVolumeClaim:
32         claimName: pv-volume
33
34 # kubectl apply -f 13.yaml
35 # kubectl edit pvc pv-volume --record
36 更改 10Mi 为 70Mi
37
```

考点官网链接: <https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-persistent-volume-storage/>

第十四题: 监控pod的日志

[← Previous](#)[Task 14 of 17](#)[→ Next](#)

Task weight: 5%

Set configuration context:

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

Task

Monitor the logs of pod `foobar` and:

- Extract log lines corresponding to error `unable-to-access-website`
- Write them to `/opt/KUTR00101/foobar`

[Flag this to return to la...](#)[I am satisfied, next →](#)

考题概述：

监控foobar pod中的日志

获取包含unable-to-access-website的日志，并将日志写入到/opt/KUTR00101/foobar

考题解析：

kubectl logs 获取日志，通过 grep 进一步获取目标日志，送分题

参考方法：

YAML 复制代码

```
1  切换 context
2
3  kubectl logs foobar | grep unable-to-access-website >
   /opt/KUTR00101/foobar
```

第十五题：添加 sidecar 容器并输出日志

Task weight: 7%

Set configuration context:

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

Context

An existing Pod needs to be integrated into the Kubernetes built-in logging architecture (e.g. `kubectl logs`). Adding a streaming sidecar container is a good and common way to accomplish this requirement.

Task

Add a sidecar container named `sidecar`, using the `busybox` image, to the existing Pod `11-factor-app`. The new sidecar container has to run the following command:

```
/bin/sh -c tail -n+1 -f /var/log/11-factor-app.log
```

Use a Volume, mounted at `/var/log`, to make the log file `11-factor-app.log` available to the sidecar container.

设置配置环境:

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

Context

一个现有的 Pod 需要集成到 Kubernetes 的内置日志记录体系结构中 (例如 `kubectl logs`)。添加 streaming sidecar 容器是实现此要求的一种好方法。

Task

使用 `busybox` image 来将名为 `sidecar` 的 sidecar 容器 添加到现有的 Pod `11-factor-app` 中。新的 sidecar 容器 必须运行以下命令:

```
/bin/sh -c "tail -n+1 -f /var/log/11-factor-app.log"
```

使用挂载在 `/var/log` 的 Volume, 使日志文件 `11-factor-app.log` 可用于 sidecar 容器。

除了添加所需的 volume mount 以外, 请勿更改现有 容器的规范。

考题概述:

添加一个sidecar容器(使用busybox 镜像)到已有的pod 11-factor-app中, 确保sidecar容器能够输出/var/log/11-factor-app.log的信息, 使用volume挂载/var/log目录, 确保sidecar能访问11-factor-app.log 文件

考题解析:

通过 `kubectl get pod -o yaml` 的方法备份原始 pod 信息, 删除旧的pod 11-factor-app copy 一份新 yaml 文件, 添加 一个名称为 sidecar 的容器, 新建 emptyDir 的卷, 确保两个容器都挂载了 /var/log 目录, 新建含有 sidecar 的 pod, 并通过 `kubectl logs` 验证;

参考方法:

- 1 答题思路:
- 2 先用 `kubectl get podname -o yaml > podname.yaml` 获取到yaml文件, 然后删除旧的 pod;
- 3 再重新 copy 一个新的 yaml 添加 sidecar 容器, 并在两个容器中都挂载 `emptyDir` 到 `/var/log`目录, 最后通过 `apply` 生成带 sidecar 的 pod; pod 正常拉起后, 通过 `kubectl logs 11-factor-app sidecar` 确认能正常输入日志即可
- 4
- 5 备份原始pod信息
- 6 `# kubectl get pod 11-factor-app -oyaml > sidecar-new.yaml`
- 7 `# cp sidecar-new.yaml 15.yaml`
- 8 删除pod
- 9 `# kubectl delete pod 11-factor-app` 或者 `kubectl delete -f sidecar-new.yaml`

编辑yaml 添加sidecar容器以及给已有的容器添加挂载:

```
image: busybox
imagePullPolicy: Always
name: count
resources: {}
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
volumeMounts:
- mountPath: /var/run/secrets/kubernetes.io/serviceaccount
  name: kube-api-access-2vzfg
  readOnly: true
- name: varlog
  mountPath: /var/log
- name: sidecar
  image: busybox
  args: [/bin/sh, -c, 'tail -n1 -f /var/log/11-factor-app.log']
  volumeMounts:
  - name: varlog
    mountPath: /var/log
dnsPolicy: ClusterFirst
enableServiceLinks: true
nodeName: k8s-node2
preemptionPolicy: PreemptLowerPriority
priority: 0
restartPolicy: Always
```

1、给已有的容器添加varlog挂载

2、添加sidecar容器

```
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
volumes:
- name: varlog
  emptyDir: {}
- name: kube-api-access-2vzfg
  projected:
    defaultMode: 420
    sources:
    - serviceAccountToken:
        expirationSeconds: 3607
        path: token
    - configMap:
        items:
        - key: ca.crt
```

3、定义varlog存储

```
1 kubectl apply -f 15.yaml
2 # 要保证pod中两个容器都是Running状态
3 root@k8s-master:~# kubectl get po 11-factor-app
4 NAME                READY   STATUS    RESTARTS   AGE
5 11-factor-app        2/2     Running   0           112s
6 kubectl logs 11-factor-app sidecar #验证日志输出
7 0: Thu Dec 23 15:15:50 UTC 2021
8 0: Thu Dec 23 15:15:55 UTC 2021
9 0: Thu Dec 23 15:16:00 UTC 2021
10 0: Thu Dec 23 15:16:05 UTC 2021
11 0: Thu Dec 23 15:16:10 UTC 2021
12 0: Thu Dec 23 15:16:15 UTC 2021
13 0: Thu Dec 23 15:16:20 UTC 2021
14 0: Thu Dec 23 15:16:25 UTC 2021
15 0: Thu Dec 23 15:16:30 UTC 2021
16 0: Thu Dec 23 15:16:35 UTC 2021
17 0: Thu Dec 23 15:16:40 UTC 2021
18 0: Thu Dec 23 15:16:45 UTC 2021
19 0: Thu Dec 23 15:16:50 UTC 2021
20 0: Thu Dec 23 15:16:55 UTC 2021
```

注意：拷贝官网案例，这题要注意复制粘贴的位置，因为有几个朋友反馈添加Sidecar容器后pod启动不了、或者格式有问题。

考题官网书签：<https://kubernetes.io/zh/docs/concepts/cluster-administration/logging/>

第十六题：查看 cpu 使用率最高的 pod



考题概述：

查找label为name=cpu-loader的pod，筛选出cpu负载最高的那个pod，并将名称追加到/opt/KUTR00401/KUTR00401.txt

考题解析：

使用top命令，结合 -l label_key=label_value 和 --sort=cpu 过滤出目标即可

参考方法：

```
1  切换 context
2
3  kubectl top pod -l name=cpu-loader -A --sort-by='cpu'
4  echo podName >> /opt/KUTR00401/KUTR00401.txt
```

Shell

复制代码

第十七题：排查集群中故障节点

← Previo... Task 17 of 17 ▾

Task weight: 13%

Set configuration context:

```
[student@node-1] $ | kubectl config use-context wk8s
```

Task

A Kubernetes worker node, named `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.

You can `ssh` to the failed node using:

```
[student@node-1] $ | ssh wk8s-node-0
```

You can assume elevated privileges on the node with the following command:

```
[student@wk8s-node-0] $ | sudo -i
```

考题概述：

节点wk8s-node-0状态为NotReady，查看原因并恢复其状态为Ready
确保操作为持久的

考题分析：

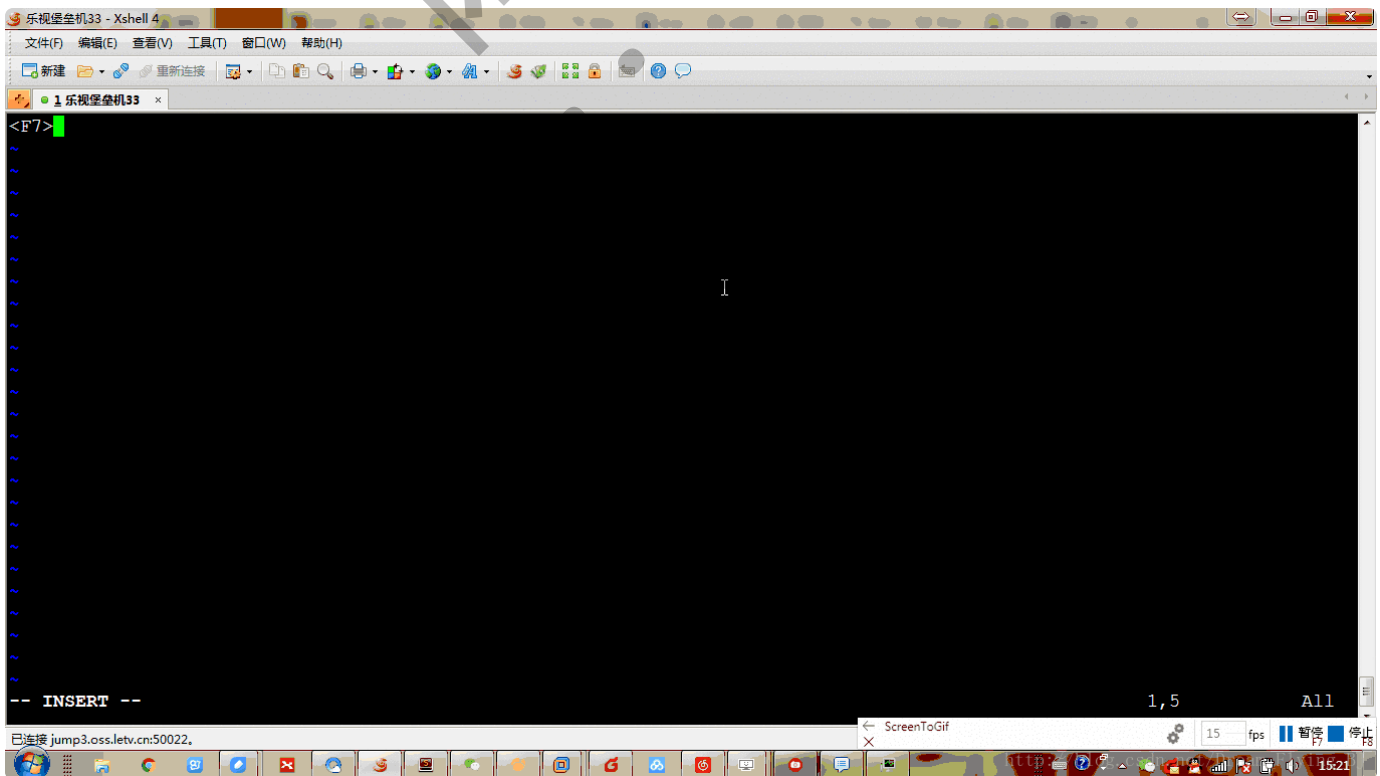
通过get nodes查看异常节点，登录节点查看kubelet等组件的status并判断原因
启动kubelet并enable kubelet即可

参考方法：

```
1  切换 context
2
3  kubectl get nodes
4  ssh wk8s-node-0
5  sudo -i
6  systemctl status kubelet
7  systemctl enable kubelet
8  systemctl restart kubelet
9  systemctl status kubelet
10
11 再次 get nodes, 确保节点恢复 Ready 状态
```

附录一、paste模式设置

先设置paste模式后，再进行粘贴，这样就不用再手动调整格式、缩进问题了。



如果PDF文件不显示动图效果，可以通过这个地址查看：<https://www.yuque.com/docs/share/e0949e2f-b053-4bbf-8a02-d161f3f3b32a?#> 《paste模式设置（动图）》

感兴趣的朋友也可以加Q群，后续题库更新会第一时间通知，并不定时分享各种云原生技术资源，一起学习、一起交流、一起进步；

