**1、Task 1 of 17**　问题权重：**4%**

**设置配置环境：**

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

**Context**

为部署管道创建一个新的 **ClusterRole** 并将其绑定到范围为特定的 **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**,将新的 ClusterRole **deployment-clusterrole** 绑定到新的 ServiceAccount **cicd-token**。

**答案：**

```
[student@node-1]$ kubectl config use-context k8s
[student@node-1]$ source <(kubectl completion bash)
[student@node-1]$ kubectl create clusterrole  deployment-clusterrole
--resource="deployment,statefulSet,daemonSet"  --verb="create"
[student@node-1]$ kubectl create sa cicd-token -n app-team1
[student@node-1]$  kubectl create clusterrolebinding ckapass --clusterrole=deployment-clusterrole
--serviceaccount="app-team1:cicd-token"
[student@node-1]$ kubectl describe clusterrolebindings ckapass
Name:         ckapass
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  deployment-clusterrole
Subjects:
  Kind           Name        Namespace
  ----           ----        ---------
  ServiceAccount  cicd-token  app-team1
```

**2、Task 2 of 17**　问题权重：**4%**

**设置配置环境：**

```
[student@node-1]$ kubectl config use-context ek8s
```

**Task**

将名为 **ek8s-node-0** 的 **node** 设置为不可用，并重新调度该 **node** 上的所有运行的 **pods**

**答案：**

```
[student@node-1]$ kubectl config use-context ek8s
[student@node-1]$ kubectl cordon ek8s-node-0
[student@node-1]$ kubectl drain ek8s-node-0  --ignore-daemonsets
```

**3、Task 3 of 17**　问题权重：**7%**

**设置配置环境：**

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

**Task**

现有的 Kubernetes 集群正在运行版本 1.20.0。仅将**主节点上**的所有 Kubernetes 控制平面和节点组件升级到版本 1.20.5

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，container 管理器，CNI 插件，DNS 服务或任何其他插件。

**答案:**

```
[student@node-1]$ kubectl config use-context mk8s
[student@node-1]$ kubectl get  nodes
NAME            STATUS   ROLES    AGE   VERSION
mk8s-master-0   Ready    master   60d   v1.20.0
mk8s-node-0     Ready    node   60d     v1.20.0

[student@node-1]$ kubectl cordon mk8s-master-0
[student@node-1]$ kubectl drain mk8s-master-0 --ignore-daemonsets
[student@node-1]$ ssh mk8s-master-0
[student@mk8s-master-0]$ sudo -i
[root@mk8s-master-0]# apt-cache policy kubectl
[root@mk8s-master-0]# apt-get install kubectl=1.20.5-00 kubelet=1.20.5-00 kubeadm=1.20.5-00
[root@mk8s-master-0]# systemctl enable kubelet
[root@mk8s-master-0]# systemctl restart kubelet
[root@mk8s-master-0]# kubeadm upgrade apply v1.20.5
[root@mk8s-master-0]# kubectl uncordon k8s-master
[root@mk8s-master-0]$ exit
[student@mk8s-master-0]$ exit
[student@node-1]$ kubectl get  nodes
NAME            STATUS   ROLES    AGE   VERSION
mk8s-master-0   Ready    master   60d   v1.20.5
mk8s-node-0     Ready    master   60d   v1.20.0
```

**4、Task 4 of 17**　　问题权重：**7%**

此项目无需要更改配置环境

**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**/data/backup/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。
然后还原位于**/srv/data/etcd-snapshot-previous.db** 的先前快照。

提供了以下 TLS 证书和秘钥，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**

**答案:**

```
[student@node-1]$ export "ETCDCTL_API=3"
[student@node-1]$ etcdctl  --help    //查看证书参数
[student@node-1]$ 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

[student@node-1]$ export "ETCDCTL_API=3"
[student@node-1]$ 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 restore
```

/srv/data/etcd-snapshot-previous.db

5、Task 5 of 17　问题权重：7%

**设置配置环境：**

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

Task

创建一个名为 allow-port-from-namespace 的新 NetworkPolicy,以允许现有 namespace my-app 中的 Pods 连接到同一 namespace 中其他 Pods 的端口 9200.

确保新的 NetworkPolicy：

- **不允许**对没有在监听端口 9200 的 Pods 的访问。
- **不允许**不来自 namespace my-app 的 Pods 的访问。

https://kubernetes.io/docs/concepts/services-networking/network-policies/

**答案：**

```
[student@node-1]$ kubectl config use-context hk8s
[student@node-1]$ kubectl label namespaces my-app name=my-app
[student@node-1]$ vim NetworkPolicy.yml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-port-from-namespace
  namespace: my-app
spec:
  podSelector: {}
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          name: my-app
  - ports:
    - protocol: TCP
      port: 9200
    - protocol: UDP
      port: 9200

[student@node-1]$ kubectl apply -f NetworkPolicy.yml
```

6、Task 6 of 17　问题权重：7%

**设置配置环境：**

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

Task

请重新配置现有的部署 front-end 以及添加名为 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，以通过在排定的节点上的 NodePort 来公开各个 Pods

```
[student@node-1]$ kubectl config use-context k8s
[student@node-1]$ kubectl edit deployment  front-end  //添加红色字体部分
spec:
  containers:
  - image: nginx
```

```
    imagePullPolicy: Always
    name: nginx
    ports:
    - containerPort: 80
      name: http
deployment.apps/front-end edited
[student@node-1]$ kubectl expose deployment front-end --type=NodePort --port=80 --name=front-end-svc
```

**7、Task 7 of 17**　问题权重：**7%**

**设置配置环境：**

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

**Task**

如下创建一个新的 **nginx Ingress** 资源：

- 名称：**pong**
- **Namespace: ing-internal**
- 使用服务端口 **5678** 在路径**/hi** 上公开服务 **hi**

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

**答案：**

```
[student@node-1]$ kubectl config use-context k8s
[student@node-1]$ vim  ingress.yml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: pong
  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  ingress.yml
```

**8、Task 8 of 17**　问题权重：**4%**

**设置配置环境：**

**Task**

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

**Task**

将 **deployment** 从 **webserver** 扩展至 **6 pods**

```
[student@node-1]$ kubectl config use-context k8s
[student@node-1]$ kubectl scale deployment webserver --replicas=6
[student@node-1]$ kubectl get pods
```

**9、Task 9 of 17**　问题权重：**4%**

**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 slector：**disk=spinning**

**答案：**

```
[student@node-1]$ kubectl config use-context k8s
[student@node-1]$ kubectl run --image=nginx nginx-kusc00401 -o yaml --dry-run=client > nginx-kusc00401.yml
[student@node-1]$ vim nginx-kusc00401.yml      //修改为如下，并添加红色字体部分
apiVersion: v1
kind: Pod
metadata:
  name: nginx-kusc00401
spec:
  containers:
  - image: nginx
    name: nginx-kusc00401
  nodeSelector:
    disk: spinning
[student@node-1]$ kubectl apply -f nginx-kusc00401.yml
```

**10、Task 10 of 17**　问题权重：**4%**

**设置配置环境：**

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

**Task**

检查有多少 worker 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

**答案：**

```
[student@node-1]$ kubectl config use-context k8s
[student@node-1]$ kubectl get nodes |  grep  Ready  | wc -l      //得到一个数值 N
[student@node-1]$ for x in `kubectl get  nodes | awk 'NR!=1{print $1}'`;do kubectl describe nodes $x | grep  -i
taints | grep NoSchedule ;done
                                                    //数一下行数 M
[student@node-1]$ echo `N-M`  > /opt/KUSC00402/kusc00402.txt
```

**11、Task 11 of 17**　问题权重：**4%**

**设置配置环境：**

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

**Task**

创建一个名为 kucc8 的 pod,在 pod 里面分别为以下每个 images 单独运行一个 app container(可能会有 1-4 个 images)：

**nginx+redis+memcached**

**答案:**

```
[student@node-1]$ kubectl config use-context k8s
[student@node-1]$ kubectl run kucc8  --image=nginx --dry-run=client -o yaml  > kucc8.yaml
[student@node-1]$ vim kucc8.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: kucc8
spec:
  containers:
  - image: nginx
    name: nginx
  - image: redis
    name: redis
  - image: memcached
    name: memcached
[student@node-1]$  kubectl apply -f kucc8.yaml
```

**12、Task 12 of 17**　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 1Gi,访问模式为 ReadWriteMany。volume 类型为 hostPath，位于 /srv/app-config

https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/#create-a-persistentvolume

**答案:**

```
[student@node-1]$ kubectl config use-context hk8s
[student@node-1]$ vim app-config.yml
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-config
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  hostPath:
    path: /srv/app-config

[student@node-1]$ kubectl apply -f  app-config.yml
[student@node-1]$ kubectl get pv
```

**13、Task 13 of 17**　问题权重：**7%**

**设置配置环境:**

```
[student@node-1]$ kubectl config use-context ok8s
```

**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，此 Pod 将作为 volume 挂载到 PersistentVolumeClaim:

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

配置新的 Pod,以对 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**。

**https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-persistent-volume-storage/**

**答案**:

```
[student@node-1]$ kubectl config use-context ok8s
[student@node-1]$ kubectl run web-server --image=nginx  -o yaml  --dry-run=client  >  web-server.yml
[student@node-1]$ vim web-server.yml
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pv-volume
spec:
  storageClassName: csi-hostpath-sc
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Mi
---
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  containers:
  - image: nginx
    name: web-server
    volumeMounts:
    - name: pvc1
      mountPath: /usr/share/nginx/html
  volumes:
  - name: pvc1
    persistentVolumeClaim:
      claimName: pv-volume

[student@node-1]$ kubectl apply  -f web-server.yml --record
[student@node-1]$ kubectl get  pod
[student@node-1]$ kubectl  get pvc
```

```
[student@node-1]$ kubectl edit pvc  pv-volume  --record    //等待以上 pod 和 pvc 都启动起来后，执行此步骤
storage: 10Mi  //改为 70Mi,保存退出后，两分钟后 kubectl get pvc 检查数值是否变为 70Mi
```

**14、Task 14 of 17**　问题权重：**5%**

**设置配置环境：**

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

**Task**

监控 **pod foo** 的日志并：

- 提取与错误 **unable-to-access-website** 相对应的日志行
- 将这些日志行写入**/opt/KUTR00101/foobar**

```
[student@node-1]$ kubectl config use-context k8s
[student@node-1]$ kubectl logs foo | grep "unable-to-access-website" > /opt/KULM00201/foobar
```

**15、Task 15 of 17**　问题权重：**7%**

**设置配置环境：**

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

**Context**

在不更改其现有容器的情况下，需要将一个现有的 Pod 集成到 Kubernetes 的内置日志记录体结构中（例如 kubectl logs）。添加 streaming sidecar 容器是实现此要求的一种好方法。

**Task**

将一个 busybox sidecar 容器添加到现有的 Pod big-corp-app。新的 sidecar 容器必须运行以下命令：

**/bin/sh -c tail -n+1 -f /var/log/big-corp-app.log**

使用名为 **logs** 的 **volume mount** 来让文件**/var/log/big-corp-app.log** 可用于 **sidecar** 容器。

不要更改现有容器。不要修改日志文件的路径，两个容器都必须通过/var/log/big-corp-app.log 来访问该文件。

```
[student@node-1]$ kubectl config use-context k8s
[student@node-1]$ kubectl  get pod big-corp-app -o yaml  >  big-corp-app.yml
[student@node-1]$ cp big-corp-app.yml  big-corp-app.yml.bake
[student@node-1]$ vim big-corp-app.yml  //修改去掉 status 和 metadata 中描述状态的信息保必要信息，添加如下内容：
```

**答案：**

```
[student@node-1]$ kubectl config use-context k8s
---
apiVersion: v1
kind: Pod
metadata:
  name: big-corp-app
spec:
  containers:
  - args:
    - /bin/sh
    - -c
    - |
      i=0; while true; do
        echo "$(date) INFO $i" >> /var/log/big-corp-app.log;
        i=$((i+1));
        sleep 1;
      done
    image: busybox
```

```
    name: big-corp-app
    volumeMounts:
    - mountPath: /var/log/
      name: logs

  - image: busybox
    name: sidecar
    command: ["/bin/sh","-c","tail -n+1 -f /var/log/big-corp-app.log"]
    volumeMounts:
    - mountPath: /var/log/
      name: logs
  volumes:
  - name: logs
    emptyDir: {}
```

```
[student@node-1]$ kubectl delete pod big-corp-app
[student@node-1]$ kubectl apply -f big-corp-app.yml
[student@node-1]$ kubectl logs  big-corp-app  -c sidecar    //查看日否有日志输出
```

**16、Task 16 of 17**　问题权重：**5%**

**设置配置环境：**

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

**Task**

**通过 pod label name=cpu-utilizer，**找到运行时占用大量 **CPU** 的 **pod，**并将占用 **CPU** 最高的 **pod** 名称写入文件 **/opt/KUTR000401/KUTR00401.txt**(已存在)。

```
[student@node-1]$ kubectl config use-context k8s
[student@node-1]$ kubectl top pod  -l name=cpu-utilizer | sort -k 2  //找到 cpu 使用最大的 pod 的名字
[student@node-1]$ echo  “pod_name” > /opt/KUTR000401/KUTR00401.txt
```

**17、Task 17 of 17**　问题权重：**13%**

**设置配置环境：**

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

**Task**

名为 wk8s-node-0 的 Kubernetes worker node 处于 NotReady 状态。调查发生这种情况的原因，并采取相应措施将 node 恢复为 Ready 状态，确保所做的任何更改永久有效。

可使用以下命令通过 ssh 连接到故障 node：

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

可使用以下命令在该 node 上获取更高权限：

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

```
[student@node-1]$ kubectl config use-context wk8s
[student@node-1]$ ssh wk8s-node-0
[student@wk8s-node-0]$ sudo -i
[root@wk8s-node-0]# systemctl status kubelet
[root@wk8s-node-0]# systemctl start kubelet
[root@wk8s-node-0]# systemctl enable kubelet
```