CKA v1.26

1、Task 1 of 17　　问题权重： **4%**

设置配置环境：

```
[candidate@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 ReadWriteOnce。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：

```
candidate@node-1:~$ kubectl config use-context hk8s
candidate@node-1:~$ vim 1.yaml
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-config
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 1Gi
  hostPath:
    path: /srv/app-config

candidate@node-1:~$ kubectl apply -f 1.yaml
candidate@node-1:~$ kubectl get pv
```

2、Task 2 of 17　　问题权重： **7%**

此项目无需要更改配置环境。但是在执行此项目之前，请确保您已返回 base 节点：

```
[candidate@mk8s-master-0]$ exit
```

Task

1 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

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。

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

答案：

```
candidate@node-1:~$ sudo -i
root@node-1:~# etcdctl --help        //查看证书参数
root@node-1:~# export "ETCDCTL_API=3"
root@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
```

```
root@node-1:~# systemctl stop etcd
root@node-1:~# mv /var/lib/etcd/ /var/lib/etcd.bake
root@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 --data-dir="/var/lib/etcd/"
root@node-1:~# systemctl start etcd
root@node-1:~# systemctl enable etcd
```

3、Task 3 of 17　问题权重： **7%**

设置配置环境：

```
[candidate@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 `/hello` 上公开 service `hello`

可以使用以下命令检查 service `hello` 的可用性，该命令应返回 `hello`：

```
[candidate@node-1]$ curl -kL <INTERNAL_IP>/hello
```

答案：

```
candidate@node-1:~$ kubectl config use-context k8s
candidate@node-1:~$ kubectl create ingress pong -n ing-internal --rule='/hello=hello:5678'
  --annotation='nginx.ingress.kubernetes.io/rewrite-target=/'
```

4、Task 4 of 17　问题权重： **7%**

设置配置环境：

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

Task

请重新配置现有的 deployment `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service，以公开容器端口 `http`。

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

答案：

```
candidate@node-1:~$ kubectl config use-context k8s
candidate@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
candidate@node-1:~$ kubectl expose deployment front-end --type=NodePort --port=80 --name=front-end-svc
```

5、Task 5 of 17　问题权重： **7%**

设置配置环境：

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

Task

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

确保新的 `NetworkPolicy` 允许 **namespace** `internal` 中的 Pods 连接到namespace `fubar` 中的 Pods 端口 `9200/tcp`。

进一步确保新的 `NetworkPolicy`：

- **不允许**对没有在监听端口 `9200/tcp` 的 **Pods** 的访问。
- **不允许**非来自 **namespace** `internal` **中的Pods 的访问**

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

答案：

```
candidate@node-1:~$ kubectl config use-context hk8s
candidate@node-1:~$ kubectl get ns internal --show-labels
NAME       STATUS   AGE   LABELS
internal   Active   28m   kubernetes.io/metadata.name=internal
candidate@node-1:~$ vim 5.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-port-from-namespace
  namespace: fubar
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: internal
    ports:
    - protocol: TCP
      port: 9200
candidate@node-1:~$ kubectl apply -f 5.yaml
```

6、Task 6 of 17　　问题权重： **4%**

设置配置环境：

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

Task

通过 pod label `name=overloaded-cpu`，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)。

答案：

```
candidate@node-1:~$ kubectl config use-context k8s
candidate@node-1:~$ kubectl top pod -l name=overloaded-cpu --sort-by=cpu          //找到 cpu 使用最大的 pod 的名字
candidate@node-1:~$ echo "pod_name" > /opt/KUTR000401/KUTR00401.txt
```

7、Task 7 of 17　　问题权重： **4%**

设置配置环境：

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

Task

将名为 `ek8s-node-1` 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：

```
candidate@node-1:~$ kubectl config use-context ek8s
candidate@node-1:~$ kubectl cordon ek8s-node-1
candidate@node-1:~$ kubectl drain ek8s-node-1 --ignore-daemonsets --force
```

8、Task 8 of 17　问题权重： **13%**

设置配置环境：

```
[candidate@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：

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

可使用以下命令在该 node 上获取更高权限：

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

答案：

```
candidate@node-1:~$ kubectl config use-context wk8s
candidate@node-1:~$ ssh wk8s-node-0
candidate@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
root@wk8s-node-0:~# exit
```

9、Task 9 of 17　问题权重： **7%**

设置配置环境：

```
candidate@node-1:~$ kubectl config use-context mk8s
```

Task

现有的 Kubernetes 集群正在运行版本 `1.26.1`。仅将 **master 节点上**的所有 Kubernetes 控制平面和节点组件升级到版本 `1.26.1`。

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：

```
[candidate@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：

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

另外，在 master 节点上升级 `kubelet` 和 `kubectl`

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：

```
candidate@node-1:~$ kubectl config use-context mk8s
candidate@node-1:~$ kubectl get nodes
NAME           STATUS   ROLES    AGE    VERSION
mk8s-master-0  Ready    master   60d    v1.26.0
mk8s-node-0    Ready    node     60d    v1.26.0
```

```
candidate@node-1:~$ kubectl cordon mk8s-master-0
candidate@node-1:~$ kubectl drain mk8s-master-0 --ignore-daemonsets
candidate@node-1:~$ ssh mk8s-master-0
candidate@mk8s-master-0:~$ sudo -i
root@mk8s-master-0:~# apt-get update
root@mk8s-master-0:~# apt-cache policy kubectl
root@mk8s-master-0:~# apt-get install kubelet=1.26.1-00 kubeadm=1.26.1-00 -y
root@mk8s-master-0:~# systemctl enable kubelet
root@mk8s-master-0:~# systemctl restart kubelet
root@mk8s-master-0:~# kubeadm upgrade apply v1.26.1 --etcd-upgrade=false
root@mk8s-master-0:~# kubectl uncordon mk8s-master-0
root@mk8s-master-0:~# exit
candidate@mk8s-master-0:~$ exit
candidate@node-1:~$ kubectl get nodes
NAME            STATUS   ROLES    AGE   VERSION
mk8s-master-0   Ready    master   60d   v1.26.1
mk8s-node-0     Ready    master   60d   v1.26.0
```

10、Task 10 of 17　问题权重： **4%**

设置配置环境：

```
[candidate@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`

答案：

```
candidate@node-1:~$ kubectl config use-context k8s
candidate@node-1:~$ kubectl logs foo | grep "unable-to-access-website" > /opt/KUTR00101/foobar
```

11、Task 11 of 17　问题权重： **4%**

设置配置环境：

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

Context

为部署流水线创建一个新的 `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：

```
candidate@node-1:~$ kubectl config use-context k8s
candidate@node-1:~$ kubectl create clusterrole  deployment-clusterrole --resource="deployment,statefulSet,daemonSet"
--verb="create"
candidate@node-1:~$ kubectl create sa cicd-token -n app-team1
candidate@node-1:~$ kubectl create rolebinding ckapass --clusterrole=deployment-clusterrole
                  --serviceaccount="app-team1:cicd-token" -n app-team1
```

```
candidate@node-1:~$ kubectl describe rolebindings ckapass -n app-team1
Name:         ckapass
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  deployment-clusterrole
Subjects:
  Kind           Name        Namespace
  ----           ----        ---------
  ServiceAccount  cicd-token  app-team1
```

12、Task 12 of 17　　问题权重： **7%**

设置配置环境：

```
[candidate@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，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/docs/tasks/configure-pod-container/configure-persistent-volume-storage/#create-a-persistentvolumeclaim

答案：

```
candidate@node-1:~$ kubectl config use-context ok8s
candidate@node-1:~$ kubectl run web-server --image=nginx -o yaml --dry-run=client > 12.yml
candidate@node-1:~$ vim 12.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:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
```

```
      claimName: pv-volume
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
```

```
candidate@node-1:~$ kubectl apply -f 12.yml
candidate@node-1:~$ kubectl get pod web-server
candidate@node-1:~$ kubectl get pvc
candidate@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: 70Mi                                               //改为 70Mi,保存退出后，两分钟后 kubectl get pvc 检查数值是否变为 70Mi
```

13、Task 13 of 17　　问题权重： **7%**

设置配置环境：

```
[candidate@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 **legacy-app** 中。新的 sidecar 容器必须运行以下命令：

/bin/sh -c "tail -n+1 -f /var/log/legacy-app.log"

使用挂载在**/var/log** 的 Volume，使日志 **legacy-app.log** 可用于 **sidecar** 容器。

除了添加所需 volume mount 以外，请勿更改现有的容器规范。

答案：

```
candidate@node-1:~$ kubectl config use-context k8s
candidate@node-1:~$ kubectl get pod legacy-app -o yaml > 13.yaml
candidate@node-1:~$ cp 13.yaml 13.yaml.bake
candidate@node-1:~$ vim 13.yaml
```

```
---
apiVersion: v1
kind: Pod
metadata:
  name: legacy-app
spec:
  containers:
  - args:
    - /bin/sh
    - -c
    - |
      i=0; while true; do
        echo "$(date) INFO $i" >> /var/log/legacy-app.log;
        i=$((i+1));
        sleep 1;
      done
    image: busybox
    name: monitor
    volumeMounts:
    - mountPath: /var/log/
```

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

```
candidate@node-1:~$ kubectl delete -f 13.yaml --now
candidate@node-1:~$ kubectl apply -f 15.yaml
candidate@node-1:~$ kubectl logs big-corp-app -c sidecar   //查看日否有日志输出
```

14、Task 14 of 17　　问题权重： **4%**

设置配置环境：

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

Task

按如下要求调度一个 Pod:

- 名称：kucc8
- app containers: 2
- container 名称/images:
  redis
  memcached

答案：

```
candidate@node-1:~$ kubectl config use-context k8s
candidate@node-1:~$ kubectl run kucc8 --image=redis --dry-run=client -o yaml  > 14.yaml
candidate@node-1:~$ vim 14.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: kucc8
spec:
  containers:
  - image: redis
    name: redis
  - image: memcached
    name: memcached
```

```
candidate@node-1:~$ kubectl apply -f 14.yaml
candidate@node-1:~$ kubectl get pod kucc8
```

15、Task 15 of 17　　问题权重： **7%**

设置配置环境：

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

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

答案：

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

16、Task 16 of 17　问题权重： **4%**

设置配置环境：

```
[candidate@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=ssd`

答案：

```
candidate@node-1:~$ kubectl config use-context k8s
candidate@node-1:~$ kubectl run --image=nginx nginx-kusc00401 -o yaml --dry-run=client > 16.yml
candidate@node-1:~$ vim 16.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: ssd

candidate@node-1:~$ kubectl apply -f 16.yml
```

17、Task 17 of 17　问题权重： **4%**

设置配置环境：

Task

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

Task

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

答案：

```
candidate@node-1:~$ kubectl config use-context k8s
candidate@node-1:~$ kubectl scale deployment webserver --replicas=6
candidate@node-1:~$ kubectl get deployments.apps webserver
```