|  |
| --- |
| OAMLab |
| 运维实验· k8s的api的简单调用 |
| 部署、api调用 |

|  |
| --- |
| 曾祥林，839225516@qq.com  2023-4-15  基于GPL v3 |

目录

[一、 硬件环境 2](#_Toc132227445)

[1 、Minikube硬件环境 2](#_Toc132227446)

[1.1 最低配置 2](#_Toc132227447)

[2 、操作系统及软件版本 2](#_Toc132227448)

[2.1 操作系统 2](#_Toc132227449)

[2.2 Kubenetes 2](#_Toc132227450)

[二、 操作系统及软件版本 2](#_Toc132227451)

[1 、操作系统 2](#_Toc132227452)

[1.1 Centos 2](#_Toc132227453)

[2 、Kubenetes 2](#_Toc132227454)

[2.1 Kubenetes 2](#_Toc132227455)

[三、 设置k8s playground 3](#_Toc132227456)

[1 、安装docker-ce 3](#_Toc132227457)

[15 、安装minikube kubectl 并启动k8s 1.16.15 3](#_Toc132227458)

[四、 实验 4](#_Toc132227459)

[1 、了解K8s资源及其在API中的组织形式 4](#_Toc132227460)

[1.1 查看API资源 4](#_Toc132227461)

[1.2 查看所有的API路径 5](#_Toc132227462)

[1.3 使用非安全端口(http)，测试api 8](#_Toc132227463)

[2 、使用curl调用k8s的api 9](#_Toc132227464)

[2.1 获取k8s的api的地址 9](#_Toc132227465)

[2.2 客户端验证api 服务器的ca证书 9](#_Toc132227466)

[2.3 验证客户端证书 10](#_Toc132227467)

[2.3.1 使用客户端证书对请求进行身份验证 10](#_Toc132227468)

[2.3.2 curl 使用该证书请求k8s的api 10](#_Toc132227469)

[2.4 使用serviceaccount token请求k8s的api 11](#_Toc132227470)

[2.5 使用useraccount来访问 11](#_Toc132227471)

[2.6 从pod内部调用k8s的api 11](#_Toc132227472)

[3 、实验结束，清理环境 11](#_Toc132227473)

[五、 其他 12](#_Toc132227474)

[1 、无 12](#_Toc132227475)

# 硬件环境

## 、Minikube硬件环境

* 1. 最低配置

最低配置2核CPU 最少2GB空闲内存 20GB 的可用磁盘空间

## 、操作系统及软件版本

* 1. 操作系统

Centos 7.9 x86 64bit

* 1. Kubenetes

Kubenetes 1.16.15

# 操作系统及软件版本

## 、操作系统

* 1. Centos

Centos 7.9 x86 64bit

## 、Kubenetes

* 1. Kubenetes

Kubenetes 1.16.15

# 设置k8s playground

## 、安装docker-ce

|  |
| --- |
| 1. sudo sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-\* 2. sudo sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-\* 3. sudo dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo 4. dnf list docker-ce --showduplicates | sort -r 5. sudo dnf -y install docker-ce --nobest 6. sudo systemctl start docker 7. sudo docker version --format {{.Server.Os}}-{{.Server.Version}} 8. sudo dnf install python3 -y # 用于解析json |

## 、安装minikube kubectl 并启动k8s 1.16.15

|  |
| --- |
| curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 \  && chmod +x minikube  curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl \  && chmod +x kubectl  cp minikube kubectl /usr/local/bin  minikube start --registry-mirror=https://registry.docker-cn.com --kubernetes-version=1.16.15 |

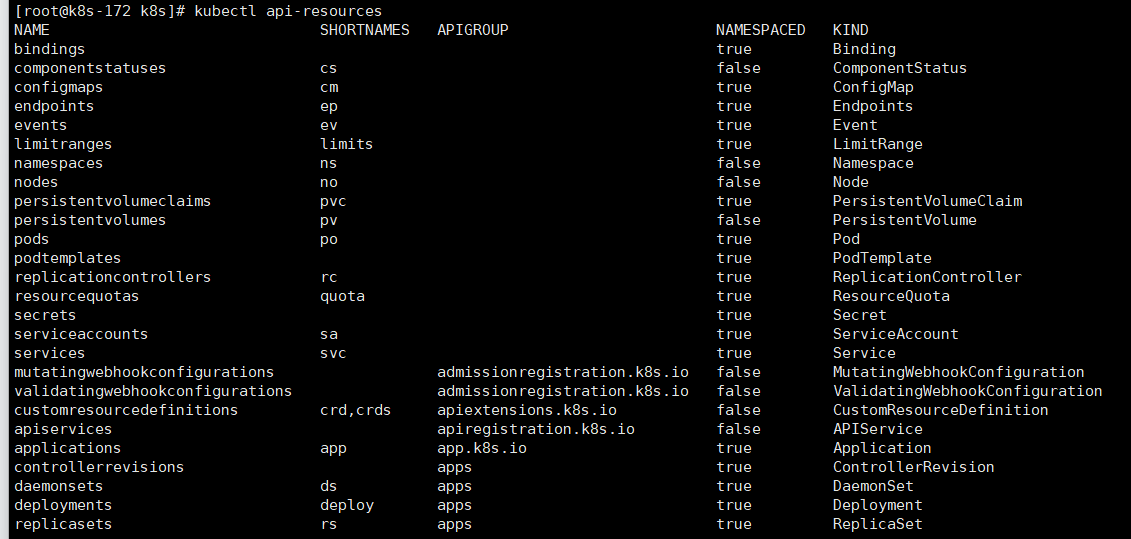
# 实验

## 、了解K8s资源及其在API中的组织形式

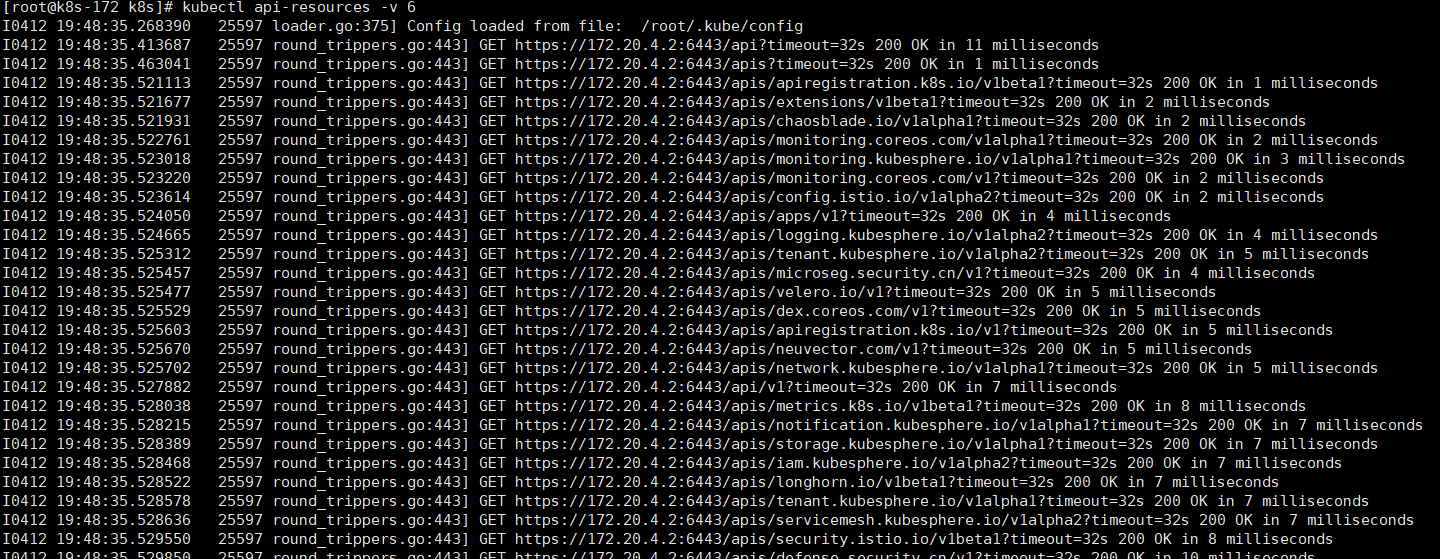
Group(API组)、Version(API版本)和Resource(API资源类型)组成

* 1. 查看API资源

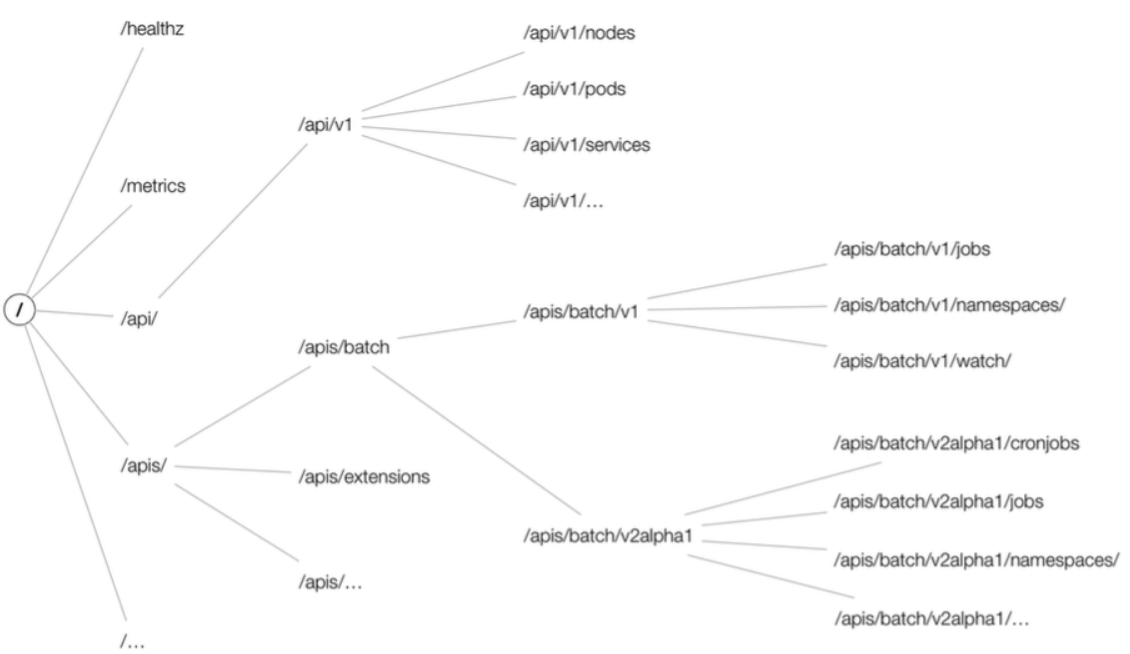
|  |
| --- |
| $ kubectl api-resources |



|  |
| --- |
| $ kubectl api-resources -v 6 |



Kubernetes里的所有API对象,实际上就可以用如下的树形结构表示出来：



* 1. 查看所有的API路径

|  |
| --- |
| $ kubectl get --raw /  {  "paths": [  "/api",  "/api/v1",  "/apis",  "/apis/",  "/apis/admissionregistration.k8s.io",  "/apis/admissionregistration.k8s.io/v1",  "/apis/admissionregistration.k8s.io/v1beta1",  "/apis/apiextensions.k8s.io",  "/apis/apiextensions.k8s.io/v1",  "/apis/apiextensions.k8s.io/v1beta1",  "/apis/apiregistration.k8s.io",  "/apis/apiregistration.k8s.io/v1",  "/apis/apiregistration.k8s.io/v1beta1",  "/apis/app.k8s.io",  "/apis/app.k8s.io/v1beta1",  "/apis/apps",  "/apis/apps/v1",  "/apis/authentication.k8s.io",  "/apis/authentication.k8s.io/v1",  "/apis/authentication.k8s.io/v1beta1",  "/apis/authorization.k8s.io",  "/apis/authorization.k8s.io/v1",  "/apis/authorization.k8s.io/v1beta1",  "/apis/autoscaling",  "/apis/autoscaling/v1",  "/apis/autoscaling/v2beta1",  "/apis/autoscaling/v2beta2",  "/apis/batch",  "/apis/batch/v1",  "/apis/batch/v1beta1",  "/apis/certificates.k8s.io",  "/apis/certificates.k8s.io/v1beta1",  "/apis/chaosblade.io",  "/apis/chaosblade.io/v1alpha1",  "/apis/cluster.kubesphere.io",  "/apis/cluster.kubesphere.io/v1alpha1",  "/apis/config.istio.io",  "/apis/config.istio.io/v1alpha2",  "/apis/coordination.k8s.io",  "/apis/coordination.k8s.io/v1",  "/apis/coordination.k8s.io/v1beta1",  "/apis/crd.projectcalico.org",  "/apis/crd.projectcalico.org/v1",  "/apis/custom.metrics.k8s.io",  "/apis/custom.metrics.k8s.io/v1beta1",  "/apis/defense.security.cn",  "/apis/defense.security.cn/v1",  "/apis/devops.kubesphere.io",  "/apis/devops.kubesphere.io/v1alpha1",  "/apis/devops.kubesphere.io/v1alpha3",  "/apis/dex.coreos.com",  "/apis/dex.coreos.com/v1",  "/apis/events.k8s.io",  "/apis/events.k8s.io/v1beta1",  "/apis/events.kubesphere.io",  "/apis/events.kubesphere.io/v1alpha1",  "/apis/extensions",  "/apis/extensions/v1beta1",  "/apis/iam.kubesphere.io",  "/apis/iam.kubesphere.io/v1alpha2",  "/apis/install.istio.io",  "/apis/install.istio.io/v1alpha1",  "/apis/logging.kubesphere.io",  "/apis/logging.kubesphere.io/v1alpha2",  "/apis/longhorn.io",  "/apis/longhorn.io/v1beta1",  "/apis/longhorn.rancher.io",  "/apis/longhorn.rancher.io/v1alpha1",  "/apis/metrics.k8s.io",  "/apis/metrics.k8s.io/v1beta1",  "/apis/microseg.security.cn",  "/apis/microseg.security.cn/v1",  "/apis/monitoring.coreos.com",  "/apis/monitoring.coreos.com/v1",  "/apis/monitoring.coreos.com/v1alpha1",  "/apis/monitoring.kiali.io",  "/apis/monitoring.kiali.io/v1alpha1",  "/apis/monitoring.kubesphere.io",  "/apis/monitoring.kubesphere.io/v1alpha1",  "/apis/network.kubesphere.io",  "/apis/network.kubesphere.io/v1alpha1",  "/apis/networking.istio.io",  "/apis/networking.istio.io/v1alpha3",  "/apis/networking.istio.io/v1beta1",  "/apis/networking.k8s.io",  "/apis/networking.k8s.io/v1",  "/apis/networking.k8s.io/v1beta1",  "/apis/neuvector.com",  "/apis/neuvector.com/v1",  "/apis/node.k8s.io",  "/apis/node.k8s.io/v1beta1",  "/apis/notification.kubesphere.io",  "/apis/notification.kubesphere.io/v1alpha1",  "/apis/policy",  "/apis/policy/v1beta1",  "/apis/rbac.authorization.k8s.io",  "/apis/rbac.authorization.k8s.io/v1",  "/apis/rbac.authorization.k8s.io/v1beta1",  "/apis/scheduling.k8s.io",  "/apis/scheduling.k8s.io/v1",  "/apis/scheduling.k8s.io/v1beta1",  "/apis/security.istio.io",  "/apis/security.istio.io/v1beta1",  "/apis/servicemesh.kubesphere.io",  "/apis/servicemesh.kubesphere.io/v1alpha2",  "/apis/storage.k8s.io",  "/apis/storage.k8s.io/v1",  "/apis/storage.k8s.io/v1beta1",  "/apis/storage.kubesphere.io",  "/apis/storage.kubesphere.io/v1alpha1",  "/apis/tenant.kubesphere.io",  "/apis/tenant.kubesphere.io/v1alpha1",  "/apis/tenant.kubesphere.io/v1alpha2",  "/apis/velero.io",  "/apis/velero.io/v1",  "/healthz",  "/healthz/autoregister-completion",  "/healthz/etcd",  "/healthz/log",  "/healthz/ping",  "/healthz/poststarthook/apiservice-openapi-controller",  "/healthz/poststarthook/apiservice-registration-controller",  "/healthz/poststarthook/apiservice-status-available-controller",  "/healthz/poststarthook/bootstrap-controller",  "/healthz/poststarthook/ca-registration",  "/healthz/poststarthook/crd-informer-synced",  "/healthz/poststarthook/generic-apiserver-start-informers",  "/healthz/poststarthook/kube-apiserver-autoregistration",  "/healthz/poststarthook/rbac/bootstrap-roles",  "/healthz/poststarthook/scheduling/bootstrap-system-priority-classes",  "/healthz/poststarthook/start-apiextensions-controllers",  "/healthz/poststarthook/start-apiextensions-informers",  "/healthz/poststarthook/start-kube-aggregator-informers",  "/healthz/poststarthook/start-kube-apiserver-admission-initializer",  "/livez",  "/livez/autoregister-completion",  "/livez/etcd",  "/livez/log",  "/livez/ping",  "/livez/poststarthook/apiservice-openapi-controller",  "/livez/poststarthook/apiservice-registration-controller",  "/livez/poststarthook/apiservice-status-available-controller",  "/livez/poststarthook/bootstrap-controller",  "/livez/poststarthook/ca-registration",  "/livez/poststarthook/crd-informer-synced",  "/livez/poststarthook/generic-apiserver-start-informers",  "/livez/poststarthook/kube-apiserver-autoregistration",  "/livez/poststarthook/rbac/bootstrap-roles",  "/livez/poststarthook/scheduling/bootstrap-system-priority-classes",  "/livez/poststarthook/start-apiextensions-controllers",  "/livez/poststarthook/start-apiextensions-informers",  "/livez/poststarthook/start-kube-aggregator-informers",  "/livez/poststarthook/start-kube-apiserver-admission-initializer",  "/logs",  "/metrics",  "/openapi/v2",  "/readyz",  "/readyz/autoregister-completion",  "/readyz/etcd",  "/readyz/informer-sync",  "/readyz/log",  "/readyz/ping",  "/readyz/poststarthook/apiservice-openapi-controller",  "/readyz/poststarthook/apiservice-registration-controller",  "/readyz/poststarthook/apiservice-status-available-controller",  "/readyz/poststarthook/bootstrap-controller",  "/readyz/poststarthook/ca-registration",  "/readyz/poststarthook/crd-informer-synced",  "/readyz/poststarthook/generic-apiserver-start-informers",  "/readyz/poststarthook/kube-apiserver-autoregistration",  "/readyz/poststarthook/rbac/bootstrap-roles",  "/readyz/poststarthook/scheduling/bootstrap-system-priority-classes",  "/readyz/poststarthook/start-apiextensions-controllers",  "/readyz/poststarthook/start-apiextensions-informers",  "/readyz/poststarthook/start-kube-aggregator-informers",  "/readyz/poststarthook/start-kube-apiserver-admission-initializer",  "/readyz/shutdown",  "/version"  ]  } |

/api/<group-name>/<api-version>/<namespace 不一定有>/<Resource>

* 1. 使用非安全端口(http)，测试api

|  |
| --- |
| # 使用localhost:8080 代理k8s api  $ kubectl proxy --port=8080 &  $ kubectl proxy --address='0.0.0.0' --accept-hosts='^\*$' --port=8001  # list 所有 api paths  $ curl http://localhost:8080/  # List known versions of the `core` group  $ curl http://localhost:8080/api  # List known resources of the `core/v1` group  $ curl http://localhost:8080/api/v1  # 查看一个pod资源  $ curl http://localhost:8080/api/v1/namespaces/default/pods/wayne-backend-69b66d7786-lspcn  # List 除了core资源的其它的api groups  $ curl http://localhost:8080/apis  # List known versions of the `apps` group  $ curl http://localhost:8080/apis/apps  # List known resources of the `apps/v1` group  $ curl http://localhost:8080/apis/apps/v1  # 查看一个Deployment resource  $ curl http://localhost:8080/apis/apps/v1/namespaces/default/deployments/wayne-backend |

## 、使用curl调用k8s的api

* 1. 获取k8s的api的地址

|  |
| --- |
| # kubectl cluster-info  Kubernetes master is running at <https://172.20.4.2:6443>  KUBE\_API=$(kubectl config view -o jsonpath='{.clusters[0].cluster.server}') |

* 1. 客户端验证api 服务器的ca证书

|  |
| --- |
| # curl $KUBE\_API/version  curl: (60) Peer's Certificate issuer is not recognized.  More details here: http://curl.haxx.se/docs/sslcerts.html |

k8s使用的是https api接口，Kubernetes API 服务器的 TLS 证书是由curl未知的证书颁发机构 (CA) minikubeCA签名的。由于curl无法信任它，因此请求失败。

默认情况下，curl信任底层操作系统所信任的同一组 CA。例如，在 Ubuntu 或 Debian 上，可以在 找到受信任的 CA 列表/etc/ssl/certs/ca-certificates.crt。显然，minikube不会将其证书添加到此文件中。

minikube将 CA 证书保存到~/.minikube/ca.crt：

k8s集群的ca证书在 /etc/kubernetes/pki/ca.crt

|  |
| --- |
| cat ~/.minikube/ca.crt | openssl x509 -text  Certificate:  Data:  Version: 3 (0x2)  Serial Number: 1 (0x1)  Signature Algorithm: sha256WithRSAEncryption  Issuer: CN = minikubeCA  Validity  Not Before: Dec 15 20:46:36 2021 GMT  Not After : Dec 14 20:46:36 2031 GMT  Subject: CN = minikubeCA  Subject Public Key Info: |

手动指定k8s ca证书来访问api

|  |
| --- |
| $ curl --cacert ~/.minikube/ca.crt $KUBE\_API/version  {  "major": "1",  "minor": "16",  "gitVersion": "v1.16.15",  "gitCommit": "2adc8d7091e89b6e3ca8d048140618ec89b39369",  "gitTreeState": "clean",  "buildDate": "2020-09-02T11:31:21Z",  "goVersion": "go1.13.15",  "compiler": "gc",  "platform": "linux/amd64"  } |

* 1. 验证客户端证书

|  |
| --- |
| curl --cacert ~/.minikube/ca.crt $KUBE\_API/apis/apps/v1/deployments  {  "kind": "Status",  "apiVersion": "v1",  "metadata": {    },  "status": "Failure",  "message": "deployments.apps is forbidden: User \"system:anonymous\" cannot list resource \"deployments\" in API group \"apps\" at the cluster scope",  "reason": "Forbidden",  "details": {  "group": "apps",  "kind": "deployments"  },  "code": 403  } |

与不受保护的/version端点不同，Kubernetes 通常会限制对其 API 端点的访问，从错误消息中可以清楚地看出，该请求已通过身份验证User "system:anonymous"，显然，该用户无权列出部署资源。

失败的请求不包含任何身份验证方式（尽管如此，它已通过身份验证，作为匿名用户），因此我需要提供一些额外的信息才能获得所需的访问级别。

* + 1. 使用客户端证书对请求进行身份验证

用户证书可以使用 kubectl config view 找到

|  |
| --- |
| $ kubectl config view -o jsonpath='{.users[0]}' | python -m json.tool  {  "name": "cluster1",  "user": {  "client-certificate": "/home/vagrant/.minikube/profiles/cluster1/client.crt",  "client-key": "/home/vagrant/.minikube/profiles/cluster1/client.key"  }  } |

* + 1. curl 使用该证书请求k8s的api

|  |
| --- |
| $ curl $KUBE\_API/apis/apps/v1/deployments \  --cacert ~/.minikube/ca.crt \  --cert ~/.minikube/profiles/cluster1/client.crt \  --key ~/.minikube/profiles/cluster1/client.key  {  "kind": "DeploymentList",  "apiVersion": "apps/v1",  "metadata": {  "resourceVersion": "654514"  },  "items": [...]  } |

* 1. 使用serviceaccount token请求k8s的api

创建serviceaccount并绑定集群角色cluster-admin：

|  |
| --- |
| kubectl create seviceaccount sa-test  kubectl create clusterrolebinding sa-test-cluster-admin --clusterrole='cluster-admin' --serviceaccount=default:sa-test |

获取seviceaccount sa-test 的secret token：

|  |
| --- |
| SERVICE\_ACCOUNT=sa-test# Get the ServiceAccount's token Secret's name  SECRET=$(kubectl get serviceaccount ${SERVICE\_ACCOUNT} -o json | jq -Mr '.secrets[].name | select(contains("token"))')# Extract the Bearer token from the Secret and decode  TOKEN=$(kubectl get secret ${SECRET} -o json | jq -Mr '.data.token' | base64 -d)# Extract, decode and write the ca.crt to a temporary location  kubectl get secret ${SECRET} -o json | jq -Mr '.data["ca.crt"]' | base64 -d > /tmp/ca.crt# Get the API Server location  APISERVER=https://$(kubectl -n default get endpoints kubernetes --no-headers | awk '{ print $2 }')  curl -s $APISERVER/openapi/v2 --header "Authorization: Bearer $TOKEN" --cacert /tmp/ca.crt | less |

* 1. 使用useraccount来访问

创建user sa-test的证书

|  |
| --- |
| openssl genrsa -out sa-test.key 2048  openssl req -new -key sa-test.key -out sa-test.csr -subj "/CN=sa-test"  openssl x509 -req -in sa-test.csr -out sa-test.crt -sha1 -CA ca.crt -CAkey ca.key -CAcreateserial -days 3650 |

创建角色getpods，创建角色绑定user panmeng和role getpods：

|  |
| --- |
| kubectl create role getpods --verb=get --verb=list --resource=pods  kubectl create rolebinding sa-test-getpods --role=getpods --user=sa-test --namespace=default |

验证访问是否正常：

|  |
| --- |
| curl --cert /etc/kubernetes/pki/sa-test.crt -X GET $APISERVER/api/v1/namespaces/default/pods?limit=1 --key /etc/kubernetes/pki/sa-test.key --insecure |

* 1. 从pod内部调用k8s的api

Kubernetes API 服务地址通过环境变量提供给 Pod：

|  |
| --- |
| $ kubectl run -it --image curlimages/curl --restart=Never mypod -- sh  $ env | grep KUBERNETES  KUBERNETES\_SERVICE\_PORT=443  KUBERNETES\_PORT=tcp://10.96.0.1:443  KUBERNETES\_PORT\_443\_TCP\_ADDR=10.96.0.1  KUBERNETES\_PORT\_443\_TCP\_PORT=443  KUBERNETES\_PORT\_443\_TCP\_PROTO=tcp  KUBERNETES\_SERVICE\_PORT\_HTTPS=443  KUBERNETES\_PORT\_443\_TCP=tcp://10.96.0.1:443  KUBERNETES\_SERVICE\_HOST=10.96.0.1 |

pod 通常还会将 Kubernetes CA 证书和服务帐户token放在/var/run/secrets/kubernetes.io/serviceaccount/

|  |
| --- |
| $ curl https://${KUBERNETES\_SERVICE\_HOST}:${KUBERNETES\_SERVICE\_PORT}/apis/apps/v1 \  --cacert /var/run/secrets/kubernetes.io/serviceaccount/ca.crt \  --header "Authorization: Bearer $(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" --cacert /var/run/secrets/kubernetes.io/serviceaccount/ca.crt \  --header "Authorization: Bearer $(cat /var/run/secrets/kubernetes.io/serviceaccount/token)" |

## 、实验结束，清理环境

|  |
| --- |
| minikube stop --profile cluster1  minikube delete --profile cluster1  rm -rf ~/.kube ~/.minikube  # 清除images和卷  docker system prune -af --volumes |

# 其他

## 、无