阿里云 × CLOUD NATIVE COMPUTING FOUNDATION

云原生技术公开课

第 27 讲

Kubernetes安全之访问控制

匡大虎（长虑）　阿里巴巴技术专家

关注“阿里巴巴云原生”公众号
获取第一手技术资料

1

Kubernetes API请求
访问控制

2

Kubernetes 认证

3

Kubernetes RBAC

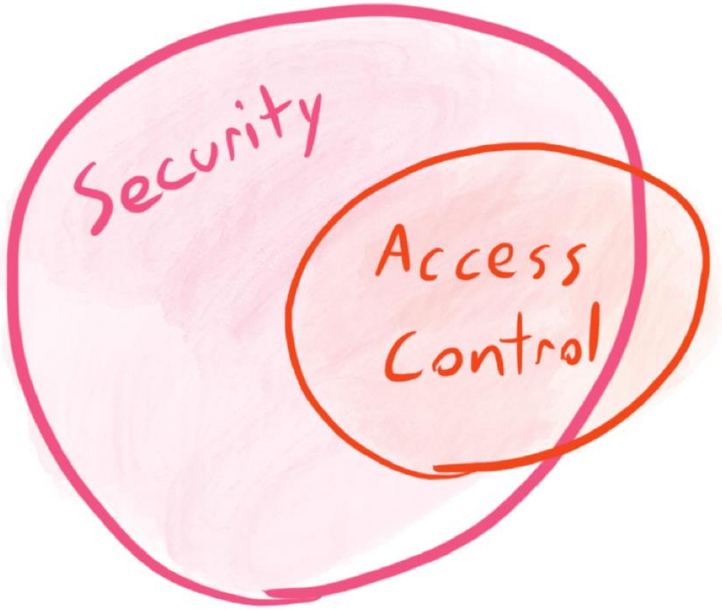
4

Security Context的使用

阿里云
CLOUD NATIVE
COMPUTING FOUNDATION

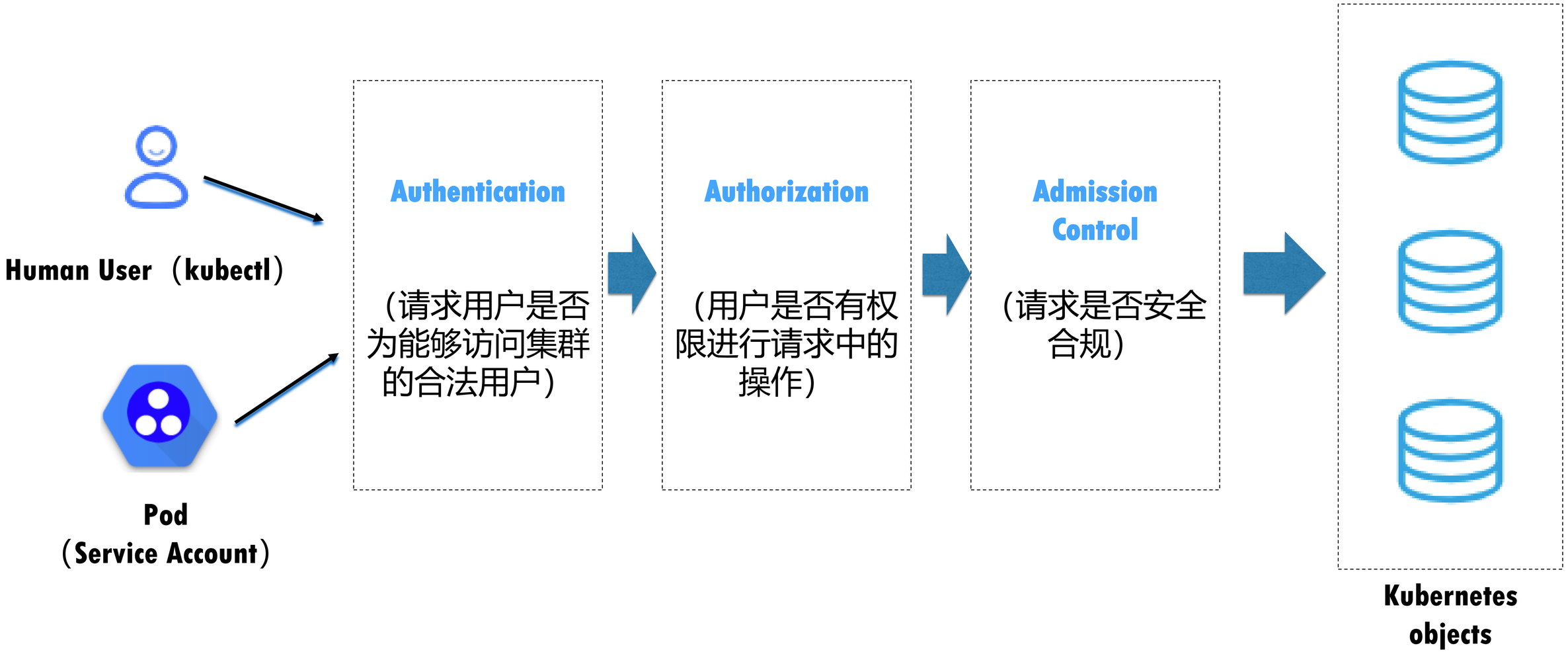
Kubernetes API 请求

- 谁
- 在何种条件下
- 可以对什么资源做什么操作



阿里云 CLOUD NATIVE COMPUTING FOUNDATION

Kubernetes API 请求



阿里云 CLOUD NATIVE COMPUTING FOUNDATION

1

Kubernetes API请求
访问控制

2

Kubernetes 认证

3

Kubernetes RBAC

4

Security Context的使用

阿里云
CLOUD NATIVE
COMPUTING FOUNDATION

Kubernetes中的用户模型

- Kubernetes 没有自身的用户管理能力
- Kubernetes中的用户通常是通过请求凭证设置
  - User=dahu
  - Groups=[“tester”,“developer”]
- Kubernetes支持的请求认证方式主要包括：
  - Basic认证
  - X509证书认证
  - Bearer Tokens (JSON Web Tokens)
    - Service Account
    - OpenID Connect
    - Webhooks

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

x509 证书认证

- 认证机构（CA）

  公钥 /etc/kubernetes/pki/ca.crt　　私钥 /etc/kubernetes/pki/ca.key

- 集群组件间通讯用证书都是由集群根CA签发
- 在证书中有两个身份凭证相关的重要字段：
  - Comman Name(CN): apiserver在认证过程中将其作为用户(user)
  - Organization(O): apiserver在认证过程中将其作为组(group)

```
➜  /tmp openssl x509 -in test1.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 112069 (0x1b5c5)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: O=cb2911270c6bf4e898bcc6b8c65958e68, OU=default, CN=cb2911
        Validity
            Not Before: Aug 27 06:32:00 2019 GMT
            Not After : Aug 26 06:37:54 2022 GMT
        Subject: O=system:masters, OU=, CN=kubernetes-admin
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b8:fb:be:c7:82:cf:09:81:c9:be:d4:b0:94:cc:
                    3d:c5:82:ec:1a:72:54:07:24:75:ab:54:24:fe:15:
                    13:85:2d:ea:15:a4:f3:ab:12:7d:20:ec:9a:14:a6:
```


阿里云 CLOUD NATIVE COMPUTING FOUNDATION

x509 证书认证

每一个**Kubernetes**系统组件都在集群创建时签发了自身对应的客户端证书

组件	**Common Name**	**Organizations**
controller-manger	**system:kube-controller-manager**	
scheduler	**system:kube-scheduler**	
kube-proxy	**system:kube-proxy**	
kubelet	**system:node:$(node-hostname)**	**system:nodes**

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

证书签发API

- Kubernetes提供了证书签发的API certificates.k8s.io/v1beta1
- 客户端将证书的签发请求发送到API server
- 签发请求会以csr资源模型的形式持久化
- 新创建好的csr模型会保持在pending的状态，直到有权限的管理员对其approve
- 一旦csr完成approved，请求对应的证书即被签发

```
cat <<EOF | kubectl apply -f -
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: my-svc.my-namespace
spec:
  request: $(cat server.csr | base64 | tr -d '\n')
  usages:
  - digital signature
  - key encipherment
  - server auth
EOF
```

```
kubectl certificate approve my-svc.my-namespace

kubectl describe csr my-svc.my-namespace

kubectl get csr

kubectl get csr my-svc.my-namespace -o
jsonpath='{.status.certificate}' \    | base64 --decode
> server.crt
```


阿里云 CLOUD NATIVE COMPUTING FOUNDATION

签发用户证书

开发人员

- 生成私钥（可借助openssl等证书工具）

```
openssl genrsa -out test.key 2048
```

- 生成csr

```
openssl req -new -key test.key -out test.csr -subj “/CN=dahu/O=devs”
```

user group

- 通过API创建k8s csr实例或向管理员提交生成的csr文件

集群管理员

- 基于csr文件或实例通过集群ca keypair签发证书，下面是openssl签发示例：

```
openssl x509 -req -in dahu.csr -CA CA_LOCATION/ca.crt -Cakey CA_LOCATION/ca.key -
Cacreateserial -out dahu.crt -days 365
```


阿里云 CLOUD NATIVE COMPUTING FOUNDATION

Service Accounts

- Service Account是Kubernetes中唯一能够通过API方式管理的APIServer访问凭证
- 通常用于pod中的业务进程与APIServer的交互
- 当一个namespace创建完成后，会同时在该namespace下生成名为default的一个Service Account和对应的secret实例
- 同样用户也可以通过API创建其他名称的Service Account，并在该namespace下挂载到运行时刻的pod中

```
kubectl get serviceaccounts

kubectl get serviceaccounts/build-robot -o
yaml
kubectl delete serviceaccount/build-robot

kubectl patch serviceaccount default -p
'{"imagePullSecrets": [{"name":
"myregistrykey"}]}'
```

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: build-robot-secret
  annotations: kubernetes.io/service-account.name: build-
robot type: kubernetes.io/service-account-token
EOF
```


阿里云 CLOUD NATIVE COMPUTING FOUNDATION

Service Accounts

service account对应的token会被装载到secret中

```
kubectl get secrets build-robot -o yaml
apiVersion: v1
data:
  ca.crt: $(CA DATA...)
  namespace: dGVzdA==
  token: $(JSON Web Token signed by API server)
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  # ....
```


阿里云 CLOUD NATIVE COMPUTING FOUNDATION

Service Accounts

service account在应用中的挂载方式

```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment-basic
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
      serviceAccountName: build-robot
```

```yaml
spec:
  containers:
  ...
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name:  build-robot-token-s95sq
      readOnly: true
  serviceAccount:  build-robot
  serviceAccountName:  build-robot
  volumes:
  - name:  build-robot-token-s95sq
    secret:
      defaultMode: 420
      secretName:  build-robot-token-s95sq
```


阿里云 CLOUD NATIVE COMPUTING FOUNDATION

生成kubeconfig

在本地进行**kubeconfig**的配置：

- 下载集群**ca**
- 使用**kubectl**添加集群连接信息

```
kubectl config set-cluster sandbox --certificate-authority=ca.pem --embed-certs=true  -
server=https://<目标集群公网地址>:6443
```

- 将新的秘钥信息加入**kubectl**配置中

```
kubectl config set-credentials dahu --client-certificate=dahu.crt --client-key=dahu.key  --
embed-certs=true
```

- 添加新的**context**入口到**kubectl**配置中

```
kubectl config set-context sandbox-dahu --cluster=sandbox --user=dahu
```


阿里云 CLOUD NATIVE COMPUTING FOUNDATION

使用kubeconfig

- 设置KUBECONFIG环境变量

```
export KUBECONFIG_SAVED=$KUBECONFIG
export KUBECONFIG=$KUBECONFIG:config-demo:config-demo-2
kubectl config view
```

- 将$HOME/.kube/config追加到KUBECONFIG环境变量设置中

```
export KUBECONFIG=$KUBECONFIG:$HOME/.kube/config
```

- 多集群config的合并和切换

```
KUBECONFIG=file1:file2:file3 kubectl config view --merge --flatten > ~/.kube/all-config export  (不同config的name需要唯一)

KUBECONFIG=~/.kube/all-config

kubectl config get-contexts
kubectl config use-context {your-contexts}
```


阿里云 CLOUD NATIVE COMPUTING FOUNDATION

1

Kubernetes API请求
访问控制

2

Kubernetes 认证

3

Kubernetes RBAC

4

Security Context的使用

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

Kubernetes鉴权—RBAC

谁(Subjects)

对象资源
(API Resources)

操作(Verbs)

Can Bob list pods?

Subject Action Resource

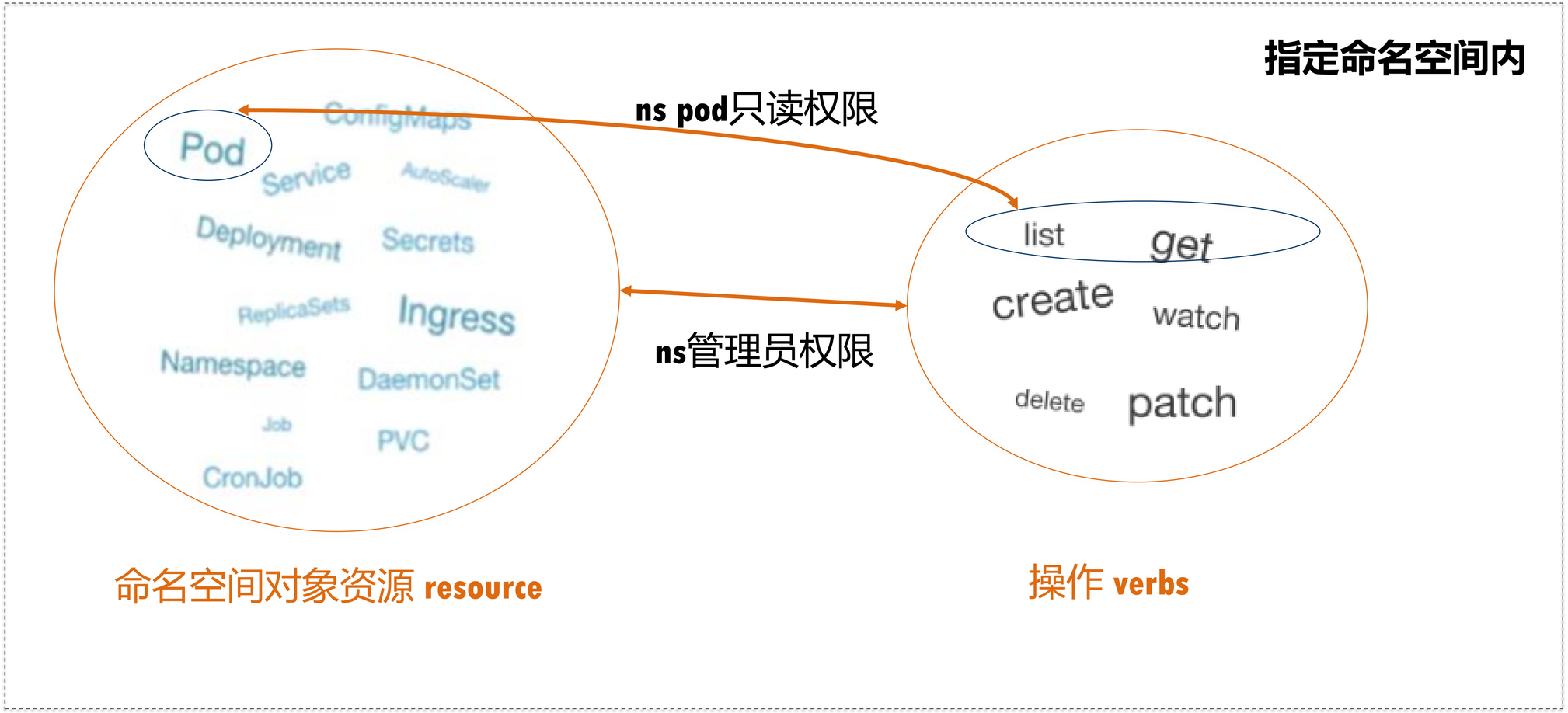
阿里云 CLOUD NATIVE COMPUTING FOUNDATION

RBAC

- 策略包含主体（**subject**）、动作（**verb**）、 资源（**resource**）和命名空间（**namespace**）
  - **User A can create pods in namespace B**
- 默认拒绝所有访问： **deny all**
- **Cannot**：
  - 不能绑定到**namespace**中的一个具体的**object**
  - 不能绑定到指定资源的任意一个**fields**
- **Can**：
  - 可以对**subresources**进行绑定（比如**nodes/status**）

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

RBAC - Role



在指定命名空间上配置角色权限，定义在指定的k8s命名空间资源上用户可以进行哪些操作

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

RBAC - Role

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: pod-access
  namespace: test
rules:
- apiGroups: [""]
  resources:  ["pods", "pods/attach"]
  verbs: ["get", "list", "watch"]
```

访问哪些资源

有哪些操作的权限

角色模板中需要指定目标资源的**API group**名称
可通过 **k8s** 官方**API**文档查询

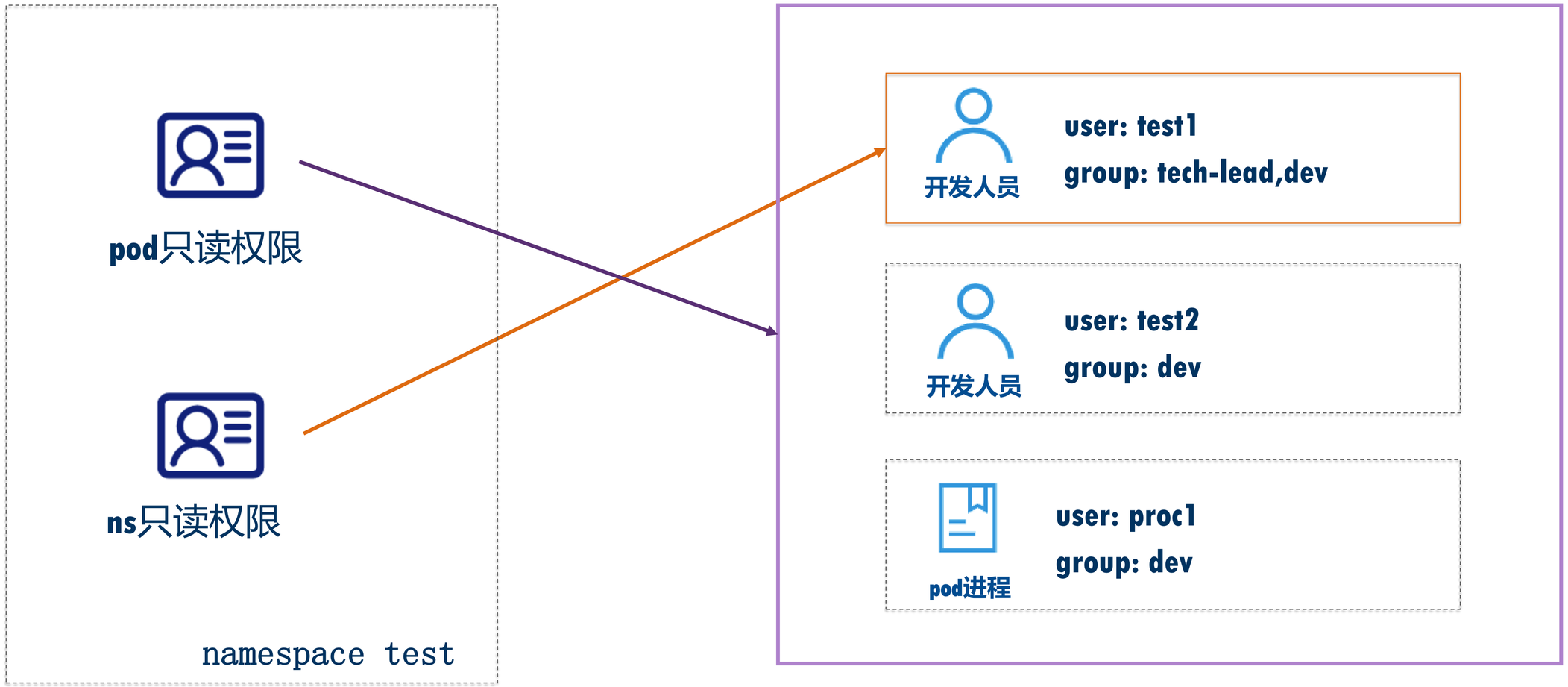
Group	Version	Kind
apps	v1	Deployment

Group	Version	Kind
core	v1	Container

如果**Group**是**core**，在角色模板中的**apiGroups**可置为空

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

RBAC - RoleBinding



阿里云 CLOUD NATIVE COMPUTING FOUNDATION

RBAC - RoleBinding

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: dev-pod-access
  namespace: test
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: pod-access
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: dev
```

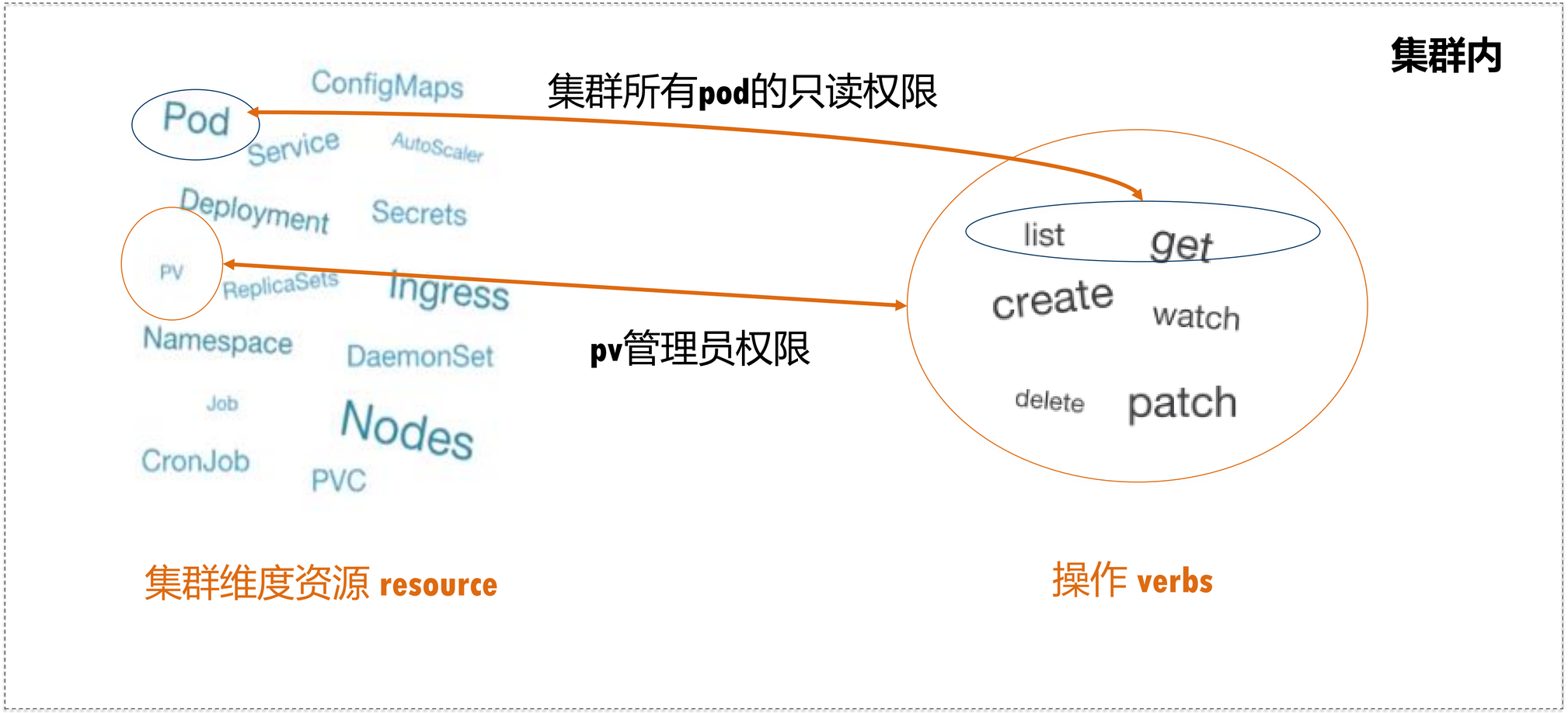
绑定的角色，一个绑定只能指定唯一的Role

绑定谁

可以是**User/Group/ServiceAccount**, 绑定到一个具体的用户，组或**Service Account**

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

RBAC - ClusterRole



在集群所有ns维度下配置角色权限，定义针对所有命名空间范围内的资源用户可以进行哪些操作

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

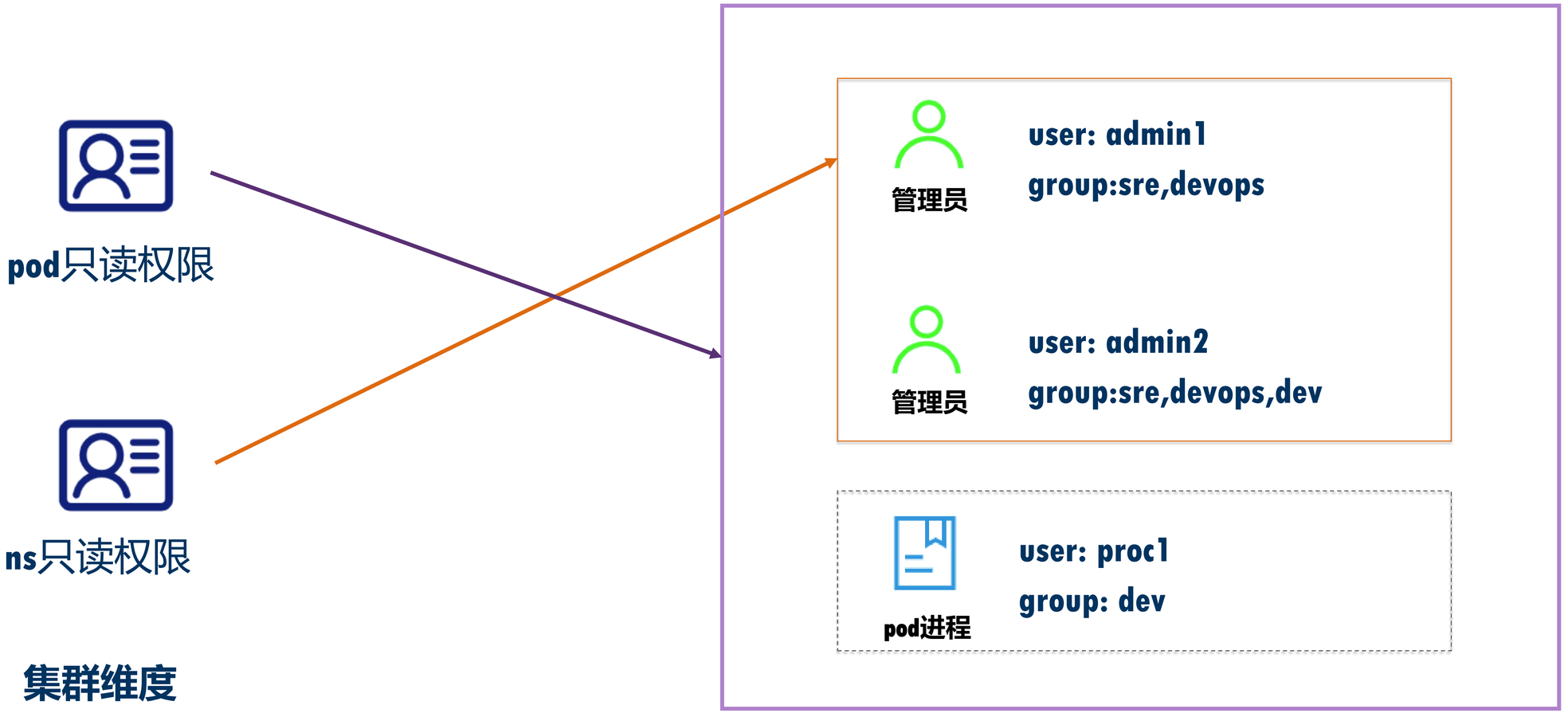
RBAC - ClusterRole

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: pod-access
  namespace: test
rules:
- apiGroups: [""]
  resources:  ["pods", "pods/attach"]
  verbs: ["get", "list", "watch"]
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: all-pod-access
rules:
- apiGroups: [""]
  resources:  ["pods", "pods/attach"]
  verbs: ["get", "list", "watch"]
```

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

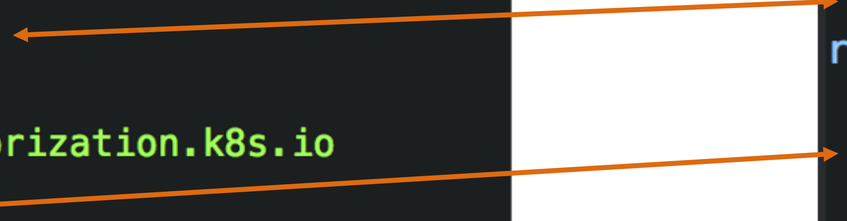
RBAC - ClusterRoleBinding



RBAC - ClusterRoleBinding

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: dev-pod-access
  namespace: test
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: pod-access
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: dev
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: dev-all-pod-access
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: all-pod-access
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: dev
```



阿里云 CLOUD NATIVE COMPUTING FOUNDATION

RBAC - Default ClusterRolebinding

- **system:basic-user**：未认证用户组(**group system:unauthenticated**)的默认角色，不具备任何的操作权限
- **cluster-admin**： **system:masters**组默认的集群角色绑定，通过绑定**cluster-admin clusterrole**，具备集群所有资源的所有操作权限

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 112069 (0x1b5c5)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: O=cb2911270c6bf4e898bcc6b8c65958e68, OU=default, CN=cb2911270c6bf4e898bcc6b8c65958e68
        Validity
            Not Before: Aug 27 06:32:00 2019 GMT
            Not After : Aug 26 06:37:54 2022 GMT
        Subject: O=system:masters, OU=, CN=kubernetes-admin
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b8:fb:be:c7:82:cf:09:81:c9:be:d4:b0:94:cc:
```

- 集群系统组件都有默认的**clusterrolebinding**，包括 **kube-controller-manager, kube-scheduler, kube-proxy……**

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

角色中的verbs如何设置？

kubectl get deploy -w	deployment: get list watch
kubectl delete deploy test	deployment: get delete
kubectl run —image=nginx nginx-test	deployment: create
kubectl edit deploy nginx-test	deployment: get patch
kubectl expose deploy nginx-test —port=80 --target-port=8000	deployment: get service: create
kubectl exec -it pod-test bash	pods: get pods/exec: create

阿里云
CLOUD NATIVE COMPUTING FOUNDATION

1

Kubernetes API请求
访问控制

2

Kubernetes 认证

3

Kubernetes RBAC

4

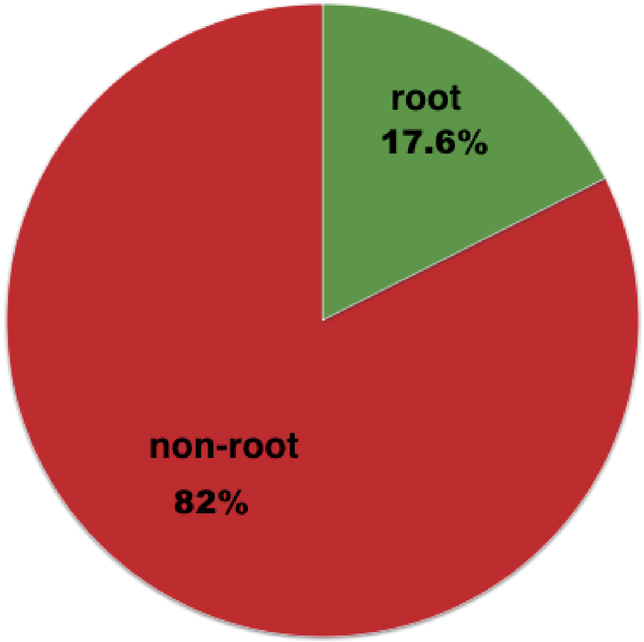
Security Context的使用

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

CVE-2019-5736

- 攻击者可以通过特定的容器镜像或者**exec**操作获取到宿主机**runc**执行时的文件句柄并修改掉**runc**的二进制文件，从而获取到宿主机的**root**执行权限

- 通过将容器设置为非**root**运行模式可以有效的阻止该攻击



most popular apps from docker hub

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

Kubernetes Runtime安全策略

- **遵循权限最小化原则**
- 在**pod**或**container**维度设置**Security Context**
- 使用**Pod Security Policy**
- 可以开启下列**admission controllers**：
  - **“ImagePoliocyWebhook”**：支持对接外部的**webhook**对部署镜像进行校验
  - **“AlwaysPullImages”**: 在多租环境下防止部署镜像被恶意篡改

Security Context Setting	描述
SecurityContext->runAsNonRoot	声明容器以非root用户运行
SecurityContext->Capabilities	控制容器运行时刻的拥有的系统capabilites，可以显示添加或删除
SecurityContext->readOnlyRootFilesystem	控制容器运行时刻是否有文件系统的写权限
PodSecurityContext-> MustRunAsNonRoot	阻止所有以root用户启动的容器

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

Pod Security Policy

控制面	字段名称
已授权容器的运行	`privileged`
为容器添加默认的一组能力	`defaultAddCapabilities`
为容器去掉某些能力	`requiredDropCapabilities`
容器能够请求添加某些能力	`allowedCapabilities`
控制卷类型的使用	`volumes`
主机网络的使用	`hostNetwork`
主机端口的使用	`hostPorts`
主机 PID namespace 的使用	`hostPID`
主机 IPC namespace 的使用	`hostIPC`
主机路径的使用	`allowedHostPaths`
容器的 SELinux 上下文	`seLinux`
用户 ID	`runAsUser`
配置允许的补充组	`supplementalGroups`
分配拥有 Pod 数据卷的 FSGroup	`fsGroup`
必须使用一个只读的 root 文件系统	`readOnlyRootFilesystem`

- 通过在**apiserver**的**admission-plugin**参数中添加**PodSecurityPolicy**开启
- 在集群中创建指定的**PSP**策略实例
- 配置策略与身份（**user/service account**）的**RBAC**策略绑定，注意大多数**pod**中使用的身份都是**serviceaccount**
- 启用**PSP**后**admission**会强制要求**pod**在鉴权后找到至少一个对应的策略实例，因此最好设置一个集群维度的全局策略，同时针对指定**namespace**配置细化策略
- 注意**PSP**策略的使用顺序，当同时有多个策略满足权限绑定关系时：
  - 优先使用非**mutating**，也就是不改变**pod**模型的策略
  - 如果上述条件过滤后仍旧有多个满足策略，通过策略实例**name**进行字母排序选取第一个策略进行校验

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

总结 – 多租安全加固

- **RBAC**和基于**namespace**的软隔离是基本且必要的安全措施
- 使用**PSP(Pod Security Policies)**对**Pod**的安全参数进行校验，同时加固**Pod**运行时刻安全
- 使用**Resource Quota & Limit Range**限制租户的资源使用配额
- 敏感信息保护 **(secret encryption at REST)**
- 在应用运行时刻遵循权限的最小化原则，尽可能缩小**pod**内容器的系统权限
- 使用**NetworkPolicy**进行业务应用间东西向网络流量的访问控制
- **Log everything**
- 对接监控系统，实现容器应用维度的监控

阿里云 CLOUD NATIVE COMPUTING FOUNDATION

谢谢观看

THANK YOU

关注“阿里巴巴云原生”公众号
获取第一手技术资料