# 20. Безопасность в кластере

# Аутентификация

В разрезе аутентификации Kubernetes оперирует не пользовательскими аккаунтами, а служебными УЗ Service Accounts.

```
kubectl create serviceaccount sa1
```

```
kubectl get serviceaccount
```

В качестве аутентификаторов могут выступать:

- имя пользователя и статический пароль;
- имя пользователя и токен;
- сертификат;
- сторонний сервис аутентификации (ldap, kerberos, etc).

## Парольная аутентификация

Что бы указать kube-apiserver.service на необходимость использования статических паролей, необходимо передать в параметрах запуска ключ –basic-auth-file=user-details.csv с файлом, содержащим список вида password, username, userid, group*:

user-details.csv

```
P@ssw0rd1,user1,u0001,group1
P@ssw0rd2,user2,u0002,group2
P@ssw0rd3,user3,u0003,group3
```

Для аутентификации:

```
curl -v -k https://master-node-ip:6443/api/v1/pods -u "user1:P@ssw0rd1"
```

## Аутентификация по токену

Аналогично для файла с токенами –token-auth-file=user-details.csv:

user-token-details.csv

```
9OSR4OYO6QwM7cfg6sKcerCyiVC6xnFZ,user1,u0001,group1
yDLmsJptxat04MUvuFdBw2CFWwv2ogcr,user2,u0002,group2
```

```
IonP8eYFUEfsZJDflyiRVSoMeDJSpBcq,user3,u0003,group3
```

Для аутентификации:

```
curl -v -k https://master-node-ip:6443/api/v1/pods --header "Authorization: Bearer 9OSR4OYO6QwM7cfg6sKcerCyiVC6xnFZ"
```

## Аутентификация по сертификату

Для аутентификации по сертификату, необходимо наличие сертификата подписанного доверенным CA, а так же, что бы в Subject Name был указан аттрибут system:masters

```
openssl genrsa -out admin.key 2048
openssl req -new -key admin.key -subj "/CN=kube-admin/O=system:masters" -out admin.csr
openssl x509 -req -in admin.csr -CA ca.crt -CAkey ca.key -out admin.crt
```

Для аутентификации:

```
curl https://master-node-ip:6443/api/v1/pods --key admin.key --cert admin.crt --cacert ca.crt
```

## Создание сертификата пользователя при помощи Certificate API

```
openssl genrsa -out new_admin.key 2048
openssl req -new -key new_admin.key -subj "/CN=New Admin,O=system:masters" -out new_admin.csr
cat new_admin.csr | base64
```

new_admin_csr.yaml

```
apiVersion: certificates.k8s.io/v1beta1
kind: CertificateSigningRequest
metadata:
  name: New Admin
spec:
  groups:
  - system:masters
  - system:authenticated
  usages:
  - digital signature
  - key encipherment
  - client auth
  request:
```

```
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJQ1dUQ0NBVUVDQVFBd0ZERVNN
QkFHQTFVRUF3d0pUbVYzSUVGa2JXbHVNSUlCSWpBTkJna3Foa2lHOXcwQgpBUUVGQUFPQ0FROEFN
SUlCQ2dLQ0FRRUF3bS94SWphM3JGNjk4aU4xUlpFWEkvYTVteG93V0ZLWW83STBvMFc0ClBueW1K
cVFqZEVIN21VM1d0ZnBPMm9PQWRMSUdTQXVockFpeExmNklNaHU3QzVCMWRuaTBDSFd5dUd
BQ1FI
eXoKNEdjdjRZREVQajVoYkt3Z3ozWkFteG94OHE2VjZCNVNnZ0RvaXo4RytVeDJLSGFFZ3R
jN1NK
ZXVTbStCZ2RuNwpWMldDbVFHaWcrUXRPS0w2L1NBdTJCdG1ML201V05CVm9IdWVmTmZNZEJ
OVWFC
K0dYYURtaWRoZmhnVXl6UVNHCnp4RGgvbTJmZUs1RjNrUXN5T3NCVGdVbElDVG9ydnlZOUR
FTDdF
bXUra0NFT0kzcXhmU1Q0WHNGZDI4aW9LS0cKSEt3c2R1MWE5OUMwVXR6QXRvRzVVbEZBS28
zK2pO
ODBEbWxBS2JReEtlQ3lId0lEQVFBQm9BQXdEUVlKS29aSQpodmNOQVFFTEJRQURnZ0VCQUJ
jUGR6
UjhlRnNqZzlUT3Fsd2I4WkZsd3MwbDNhdG1QdFRSaUZqQnpudjRoem95Ck1BNG9TWkNhdVN
2akdY
Z2VSclpDdG9iZkpPemxRMGI1bTlic2tPNWl3Wnk0WGg5Q3lMTStnTUFkWlhkSk9uaWYKOHN
BYzVq
b1hOY0ZmMDRJOEZtZTZDV0RPVnBEbW1peVM2bFNiVHpwU1NIbEluU3BuV1RRSzFDVGVYeXN
IaXds
SgplVDFKMy9yM0dOQTk0eE1TN3VFTVM2VTR2dW1RcEx2UG9KSW5rL3NMeGRUa1BuTEJmQXl
3N1ZX
V04wNmRGZHJXCllsTkhTcVVBRFBQYWR1TWlsc3lnZmdjR3RMZGZnc2tpNlBjTENrbTBKY2R
INXVq
MytCcWlDaEdlQjZtbTgwQVgKTnJMTHZDOVRkVTBlN0Uvek1KQ1Z0V2lVbGlIOFNET3dyVVl
zaWtJ
    PQotLS0tLUVORCBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0K
```

```
kubectl get csr
kubectl certificate approve new_admin
kubectl get csr new_admin -o yaml
echo "coded certificate" | base64 --decode > new_admin.crt
```

# RBAC

## Создание роли

pod-reader-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
```

```
  name: pod-reader
rules:
- apiGroups: [""] #
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
- apiGroups: [""] #
  resources: ["ConfigMap"]
  verbs: ["get", "create"]
```

## Биндинг роли к конкретному пользователю

pod-reader-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: pod-reader-binding
  namespace: default
subjects:
# You can specify more than one "subject"
- kind: User
  name: jane # "name" is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  # "roleRef" specifies the binding to a Role / ClusterRole
  kind: Role #this must be Role or ClusterRole
  name: pod-reader # this must match the name of the Role or
ClusterRole you wish to bind to
  apiGroup: rbac.authorization.k8s.io
```

## Просмотр сведений RBAC

```
kubectl get roles
kubectl get rolebindings
kubectl describe role developer
kubectl describe rolebinding dev-users-rolebinding
```

## Проверка доступа

```
kubectl auth can-i create deployments
kubectl auth can-i delete nodes
kubectl auth can-i create pods --as sample-user
```

# Кластерные роли

Кластерные роли, в отличие от обычных, не привязаны к ресурсам в определенном namespace.

### Кластерная роль

cluster-role.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-administrator
rules:
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["list","get","create","delete"]
```

### Привязка кластерной роли к пользователю

cluster-role-binding.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
# This cluster role binding allows anyone in the "manager" group to
read secrets in any namespace.
kind: ClusterRoleBinding
metadata:
  name: cluster-admin-role-binding
subjects:
- kind: User
  name: cluster-administrator # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
- kind: Group
  name: cluster-admins # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: cluster-administrator
  apiGroup: rbac.authorization.k8s.io
```

# Service Accounts

По умолчанию в каждом namespace уже существует ServiceAccount с именем default, который

автоматически цепляется каждым создаваемым Pod'ом. При этом, default аккаунт сильно порезан в правах.

Создать ServiceAccount

```
kubectl create serviceaccount test-sa
```

Получить список ServiceAccount'ов

```
kubectl get serviceaccount
```

Вывести полную информацию о ServiceAccount'е

```
kubectl describe serviceaccount test-sa
```

Получить токен безопасности определенного ServiceAccount'а

```
kubectl describe secret test-sa-token-kbbdm
```

Применить ServiceAccount к поду:

pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: simplePod
spec:
  containers:
  - name: simplePod
    image: simplePod
  serviceAccountName: simplePod-sa
```