

Управление доступом

User, ServiceAccount, RBAC

Кирилл Касаткин
DevOps-инженер, Renue



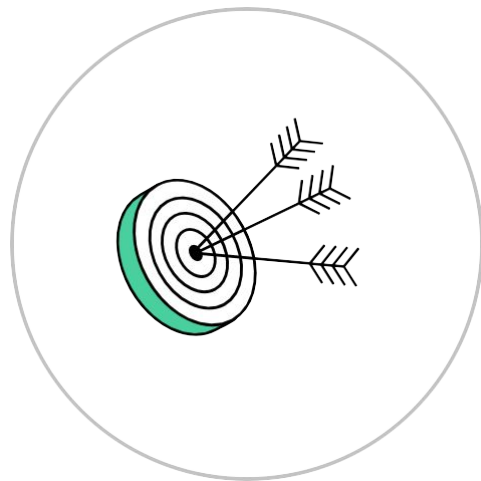
Кирилл Касаткин

DevOps-инженер, Renuе



Цели занятия

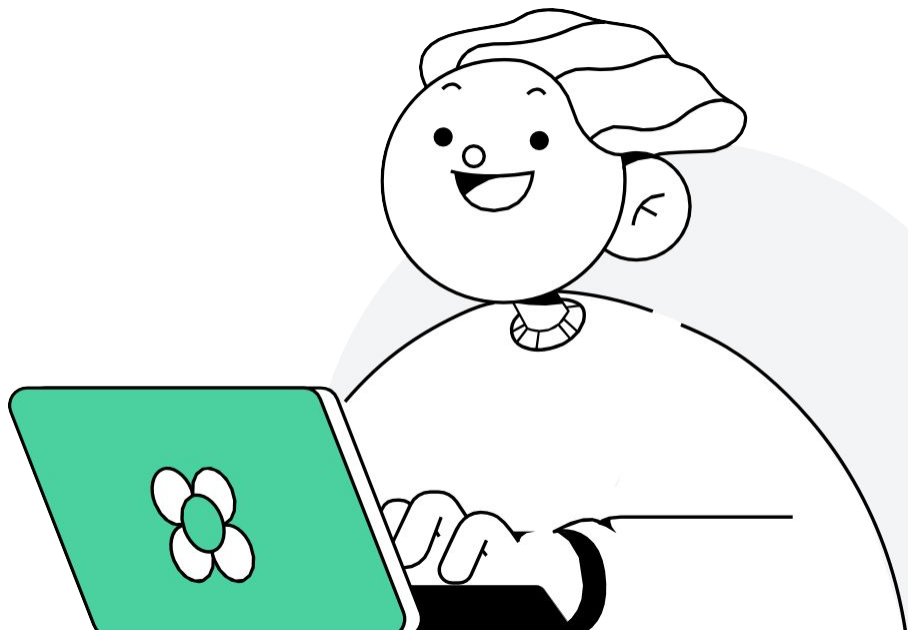
- Узнать:
 - что такое RBAC
 - что такое ServiceAccount и Роли (Role)
- Разобраться с подключением Ролей и пользователей
- Разобрать примеры манифестов объектов K8s



План занятия

- 1 Доступ к API K8s
- 2 RBAC
- 3 Демонстрация работы
- 4 Итоги
- 5 Домашнее задание

*Нажми на нужный раздел для перехода



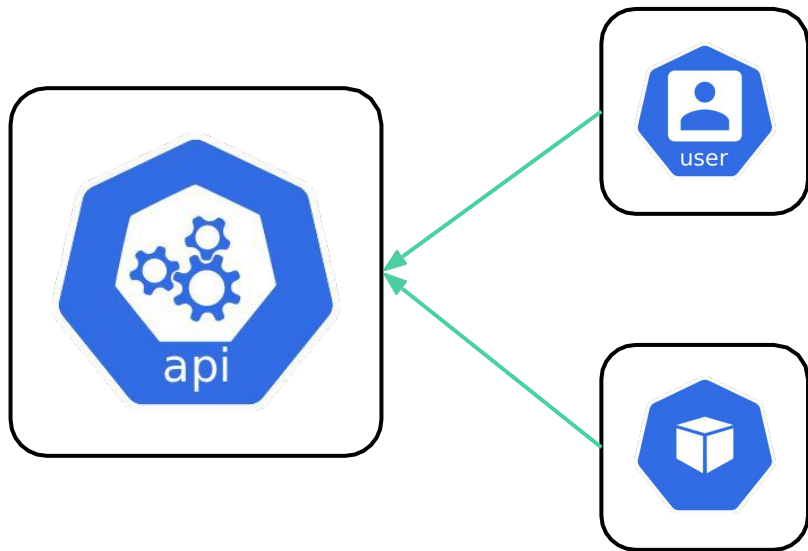
Доступ к API K8s



1

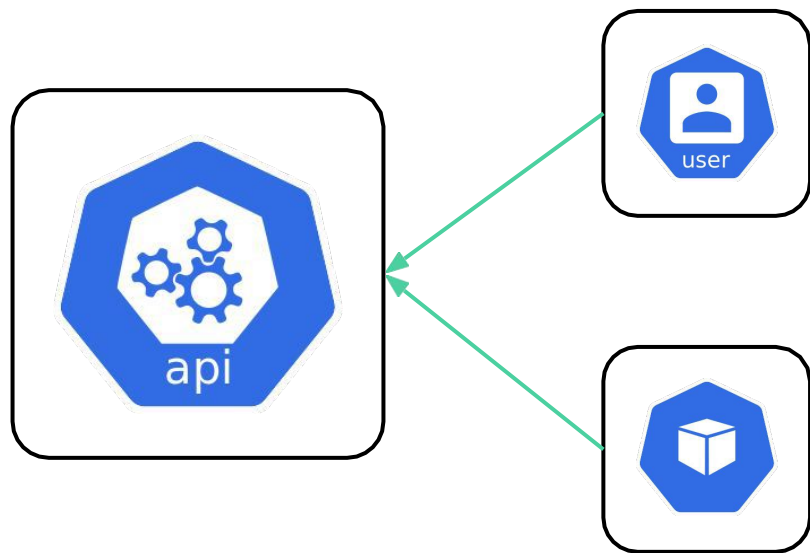
Авторизация и аутентификация

Доступ к API кластера K8s возможен как для пользователей, так и для приложений. При этом K8s не управляет user accounts нативно (нет такого объекта).



Авторизация и аутентификация

Доступ к API кластера K8s возможен как для пользователей, так и для приложений. При этом K8s не управляет user accounts нативно (нет такого объекта).



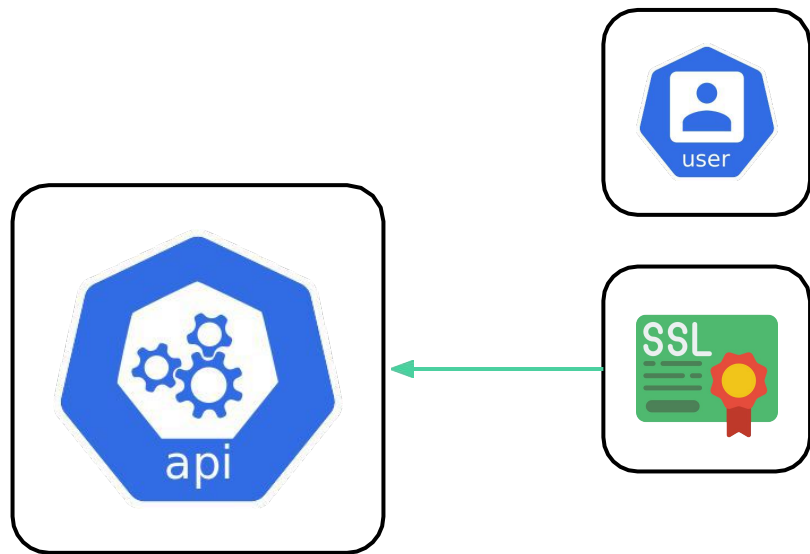
Авторизация пользователей возможна с помощью:

- базовой авторизации
- SSL-сертификатов
- token
- сторонних приложений

Авторизация приложений — token

Авторизация и аутентификация

Доступ к API кластера K8s для пользователя с помощью **SSL**



Особенности:

- можно сгенерировать сертификат и подписать в K8s
- можно использовать объект **CertificateSigningRequest** и `kubectl certificate approve` (например, со стороны админов)

После этого создать config-файл для подключения к кластеру (`kubectl config`)

Пример конфигурации

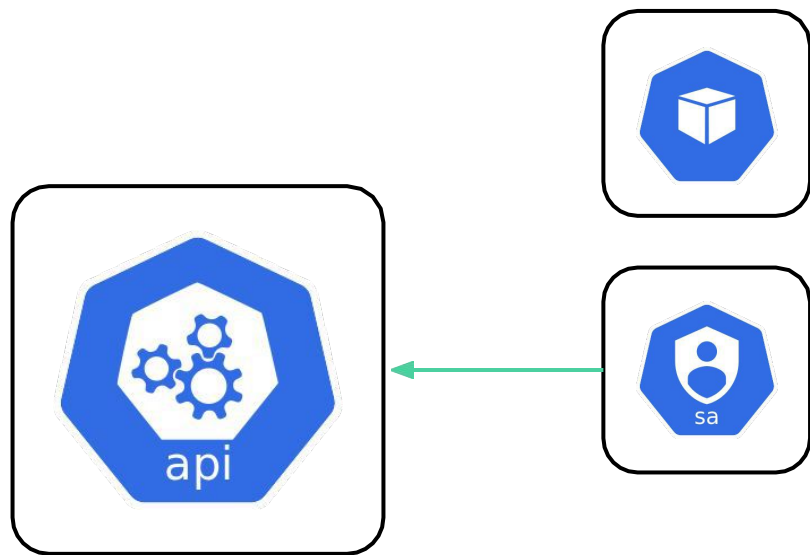
Пример конфигураций CertificateSigningRequest (подробности [здесь](#)):

```
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: ssl-csr
spec:
  groups:
  -system:authenticated
  request: ${BASE64_CSR}
  usages:
  - digital signature
  - key encipherment
  - server auth
  - client auth
```

- `kubectl get csr`
- `kubectl certificate approve ssl-csr`
- `kubectl get csr ssl-csr -o jsonpath={.status.certificate} | base64 --decode > cert.crt`

Авторизация и аутентификация

Доступ к API кластера K8s для Pod осуществляется с помощью **ServiceAccount** (сервис-аккаунт).



Особенности сервис-аккаунта:

- 1 Создается в namespace (по умолчанию default)
- 2 У каждого есть токен
- 3 Монтируется внутри контейнера в

`/var/run/secrets/kubernetes.io/serviceaccount:`

- ca.crt — сертификат CA кластера
- namespace содержит имя namespace, в котором находится Pod
- token содержит токен, используемый для доступа к API кластера

Пример конфигурации

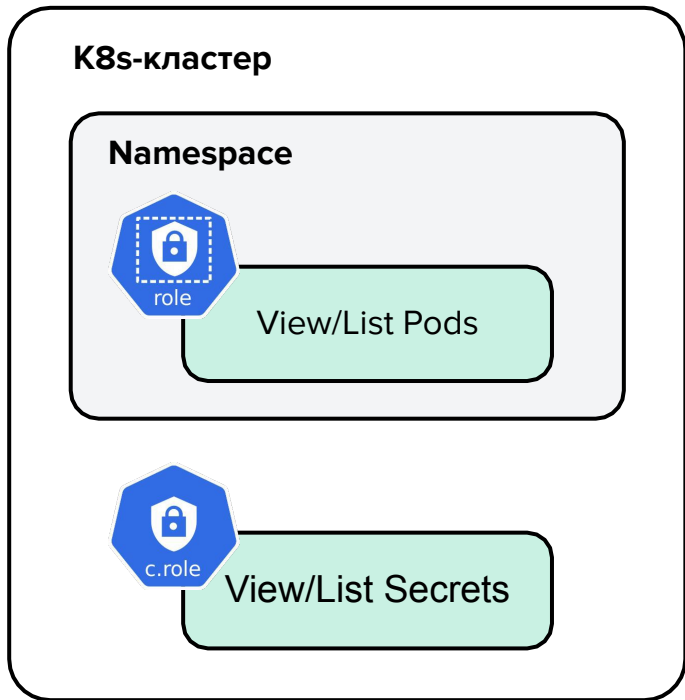
Пример конфигураций ServiceAccount и Pod:

```
apiVersion:v1
kind: ServiceAccount
metadata:
  name: sa-pod
  namespace: my-ns
```

```
apiVersion: v1
kind: Pod
metadata:
  name: pod
  namespace: my-ns
metadata:
  containers:
    - name: busybox
      image: busybox
      command: ['some', 'command']
  serviceAccountName: sa-pod
```

Авторизация и аутентификация

Role и ClusterRole — наборы правил, представляющие набор разрешений.




- **Role** — используется для предоставления доступа к ресурсам внутри namespace
- **ClusterRole** — используется для предоставления доступа к ресурсам, доступным в пределах всего кластера

RBA

C



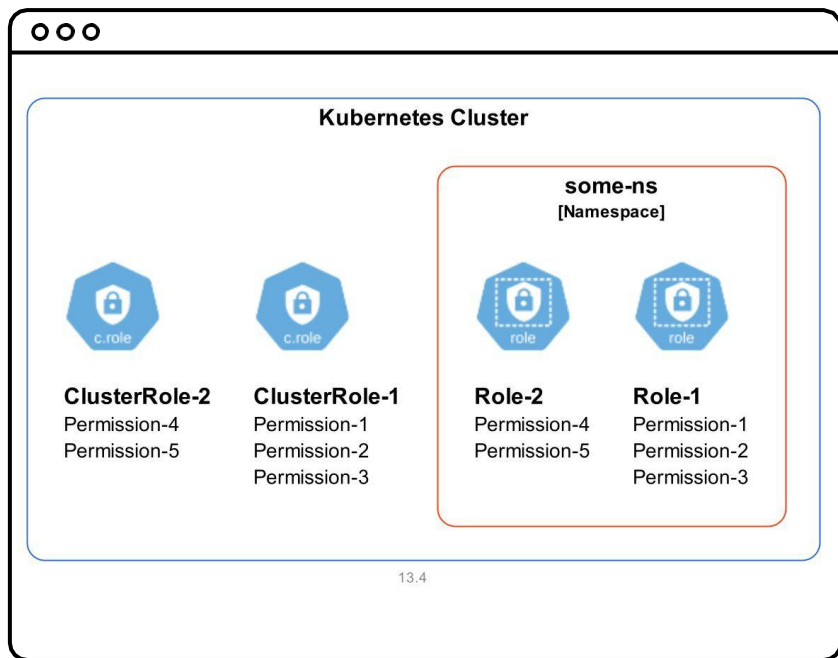
2



**Role-based access control
(RBAC) — управление
доступом, основанное
на ролях**

Роли в RBAC

RBAC можно использовать со всеми ресурсами Kubernetes, которые поддерживают CRUD (create, read, update, delete).



- **Role** — используется для предоставления доступа к ресурсам внутри namespace
- **ClusterRole** — используется для предоставления доступа к ресурсам, доступным в пределах всего кластера

Пример конфигурации

Пример конфигураций Role и ClusterRole:

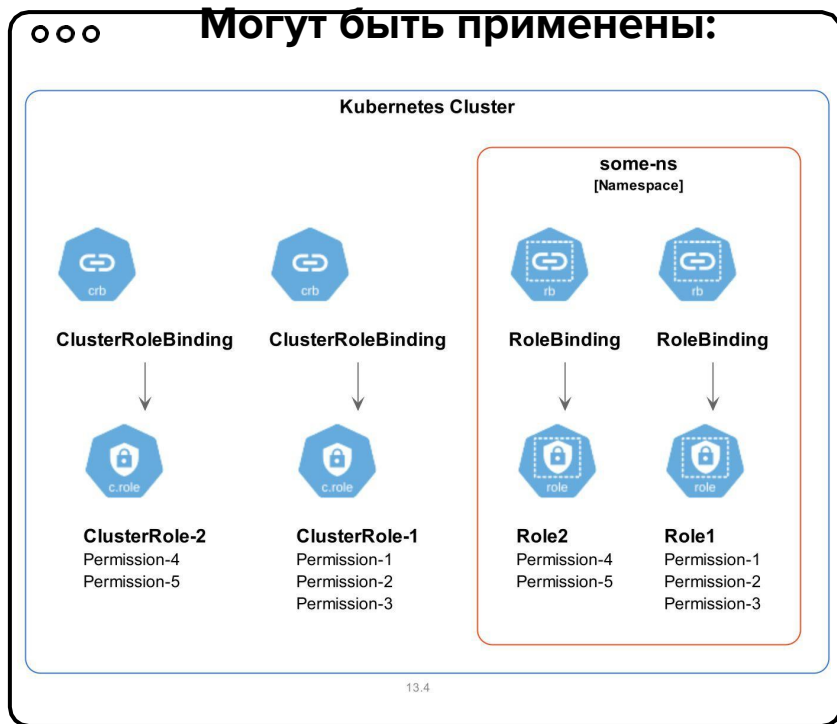
```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: pod-reader
  namespace: my-ns
rules:
- apiGroups: [""]
  resources: ["pods", "pods/log"]
  verbs: ["get", "watch", "list"]
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
```

Список apiGroups можно увидеть командой *kubectl api-resources*

Роли в RBAC

RoleBinding и ClusterRoleBinding — привязка субъекта к Role или ClusterRole.



- на users
- groups
- service accounts

Пример конфигурации

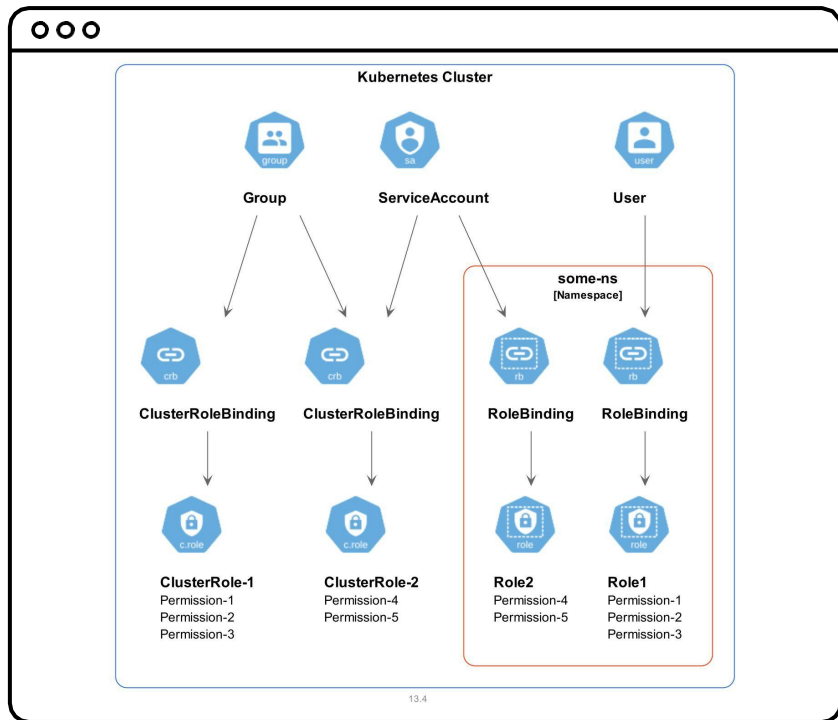
Пример конфигураций Role и ClusterRole:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: pod-reader
  namespace: my-ns
subjects:
- kind: User
  name: dev
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: read-secrets-global
subjects:
- kind: Group
  name: managers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

Роли в RBAC

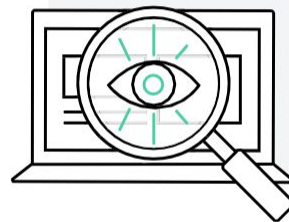
RoleBinding и ClusterRoleBinding могут содержать несколько «пользователей»:



```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: read-secrets-global
subjects:
- kind: Group
  name: managers
  apiGroup: rbac.authorization.k8s.io
- kind: ServiceAccount
  name: sa-secret-reader
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

Демонстрация работы

SSL-авторизация, ServiceAccount, RBAC



Итоги

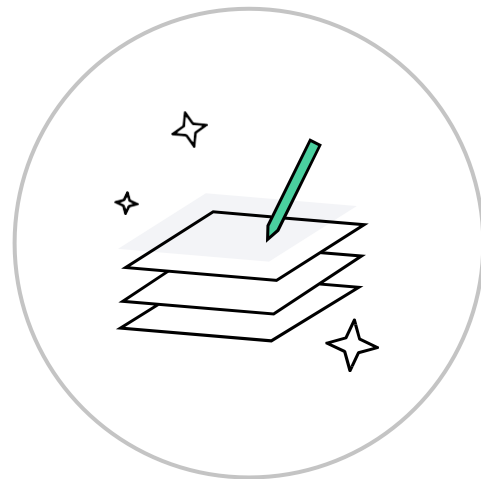
- 1 Узнали, что такое ServiceAccount, Роли и RBAC
- 2 Разобрались с подключением Ролей и пользователей
- 3 Поняли, как можно ограничить права доступа пользователей и приложений к API кластера
- 4 Рассмотрели примеры манифестов объектов K8s
- 5 Попробовали подключиться к кластеру и посмотреть в работе объекты, изученные на занятии



Домашнее задание

Давайте посмотрим ваше [домашнее задание](#)

- 1 Вопросы о домашней работе задавайте в чате группы
- 2 Задачи можно сдавать по частям
- 3 Зачёт по домашней работе ставят после того, как приняты все задачи



**Задавайте вопросы
и пишите отзыв о лекции**

