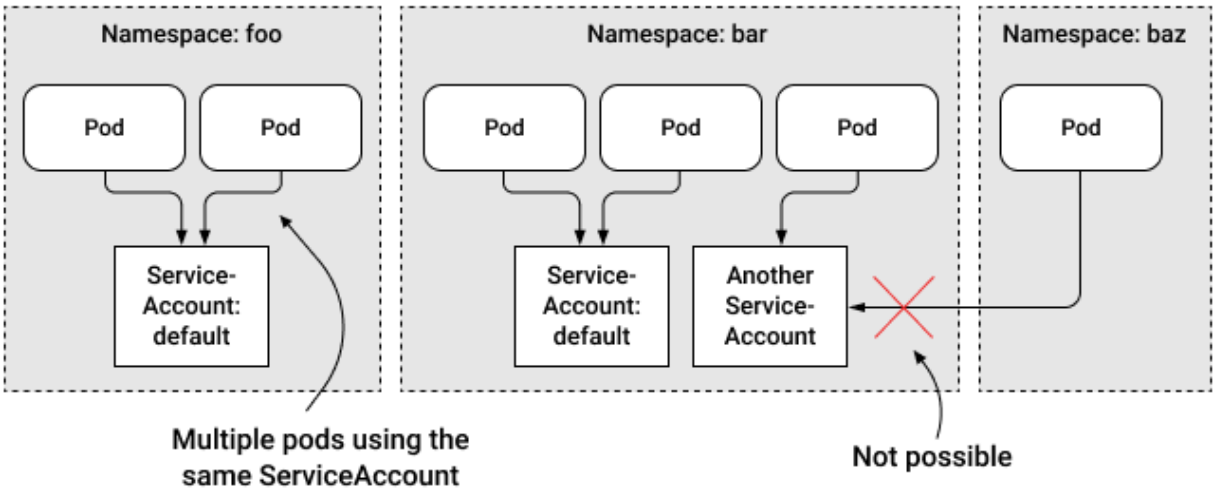
K8S 38: Kubernetes. Resources. RBAC

Описание:

- Role Based Access Control (RBAC) with Kubernetes
- Effective RBAC - Jordan Liggitt, Red Hat

ДляRBAC — это предоставление определенных прав соответствующим аккаунтам (User Account, Service Account). Если описывать простым языком, то в K8S существуют поды, которым соответствуют определенные SA. Этим SA может соответствовать описание разрешенных действий (к примеру, create/get/update/patch/delete) над рядом сущностей (это могут быть другие pods/rs/deployments или определенные объекты или процессы, как pod/exec, и т.д.).
начала рассмотрим объект ServiceAccount. Это внутренний объект kubernetes, формируемый по маске system:serviceaccount::. API-сервер передает это имя пользователя плагинам авторизации, которые сопоставляют его с группой, получая список доступных прав. По умолчанию уже есть несколько serviceaccount, в default есть в каждом namespace kubectl get sa



Данный аккаунт автоматически монтируется в каждый pod /var/run/secrets/kubernetes.io/serviceaccount/token. Внутри пода можно делать запросы к API server с этим токеном. Однако данный аккаунт ограничен своим namespace, получить информацию о ресурсах другого namespace под этим аккаунтом невозможно.
Есть два типа ролей: Role — указанный класс описывает права только в рамках указанного namespace. ClusterRole — описывает права в рамках всего кластера. Сначала разберемся с Role. Посмотрим пример описания роли:

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

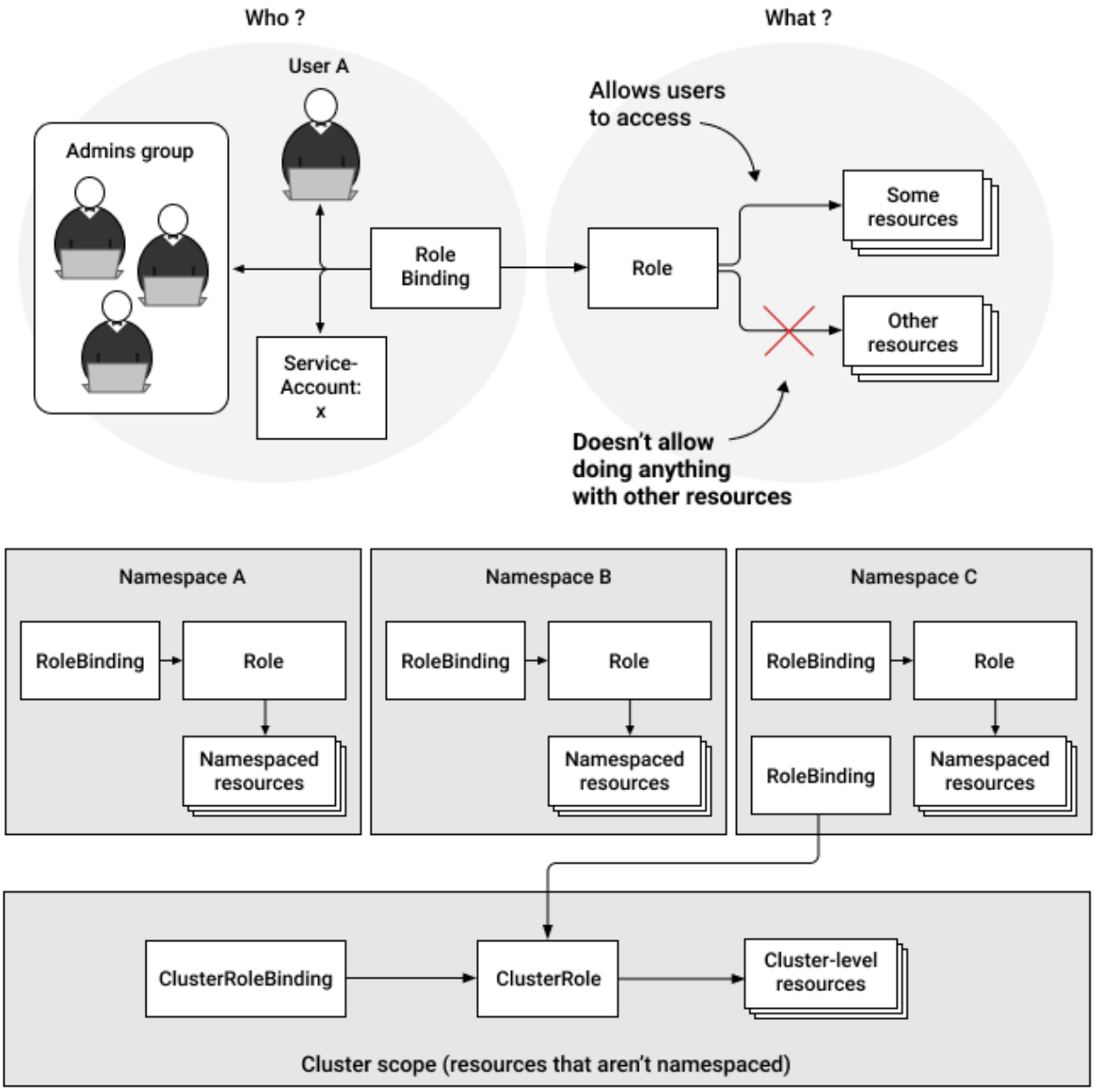
```
  namespace: lfs158
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

Данное описание создаст роль в namespace lfs158, и в нем разрешается доступ к операциям get/watch/list над ресурсами pods в api-group 'core'.
Кластерная роль описывается похожим образом, но не указывает namespace, так как распространяется на весь кластер.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  # "namespace" omitted since ClusterRoles are not namespaced
  name: secret-reader
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
```

HTTP method	Verb for sibgle resource	Verb for collection
GET, HEAD	get (and watch for watching)	list (and watch)
POST	create	n/a
PUT	update	n/a
PATCH	patch	n/a
DELETE	delete	deletecollection

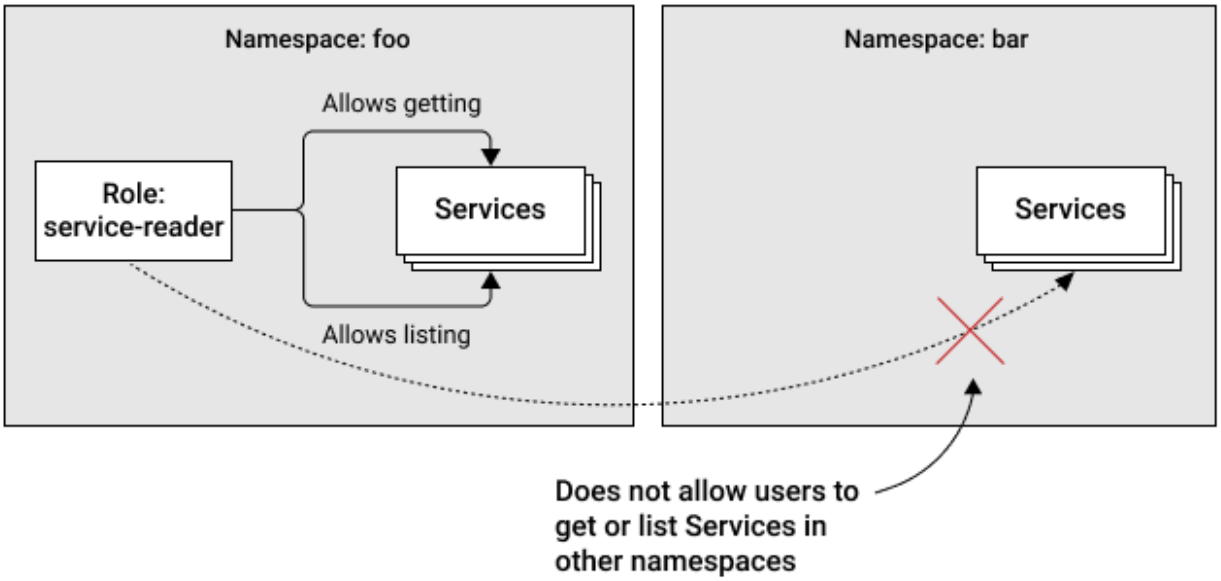
Роль прикрепляется при помощи RoleBindindg и ClusterRoleBinding. Отличия понятны из названия. В первом случае связывание роли и sa происходит в рамках одного namespace. Во втором — в рамках кластера.



Пример привязки RoleBinding

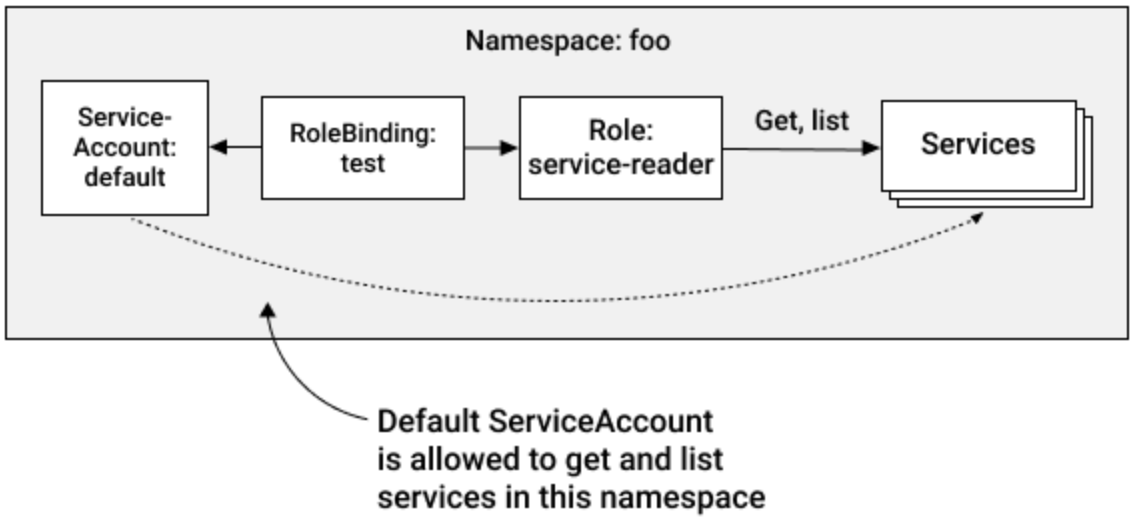
```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: foo
  name: service-reader
rules:
- apiGroups: [""]
  verbs: ["get", "list"]
  resources: ["services"]
```

Данная роль будет создана в namespace foo и разрешит только два действия с сервисами в данном namespace.



Для привязки роли к аккаунту

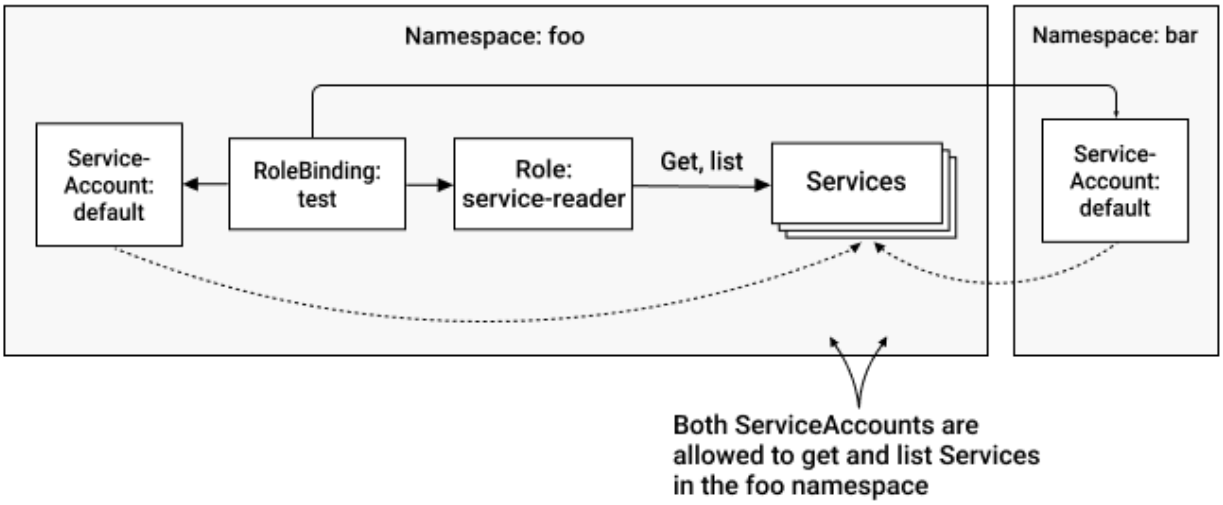
```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: test
  namespace: foo
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: service-reader
subjects:
- kind: ServiceAccount
  name: default
  namespace: foo
```



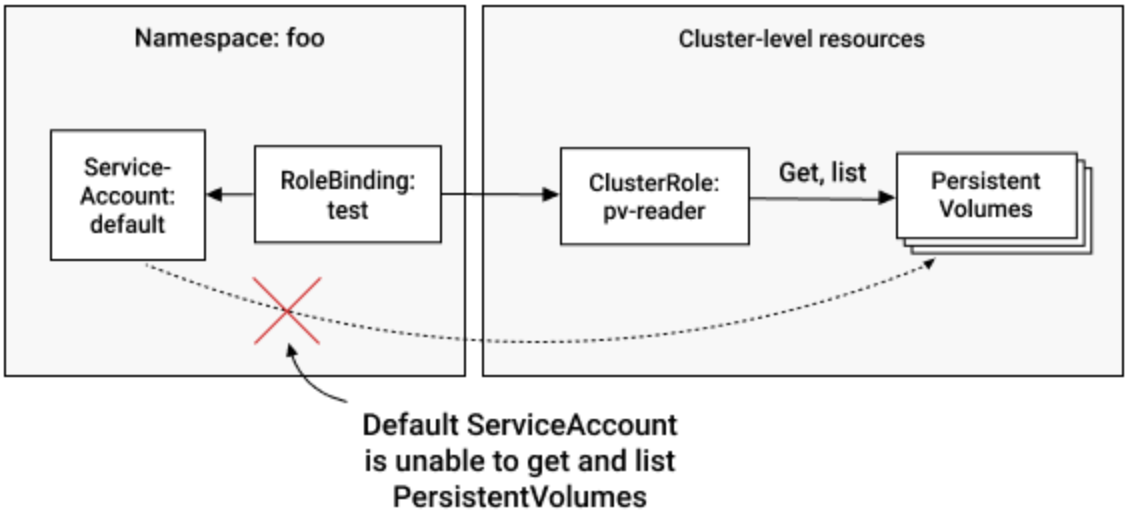
Также роль можно привязать к аккаунту из другого namespace

```
subjects:
- kind: ServiceAccount
  name: default
  namespace: bar
```

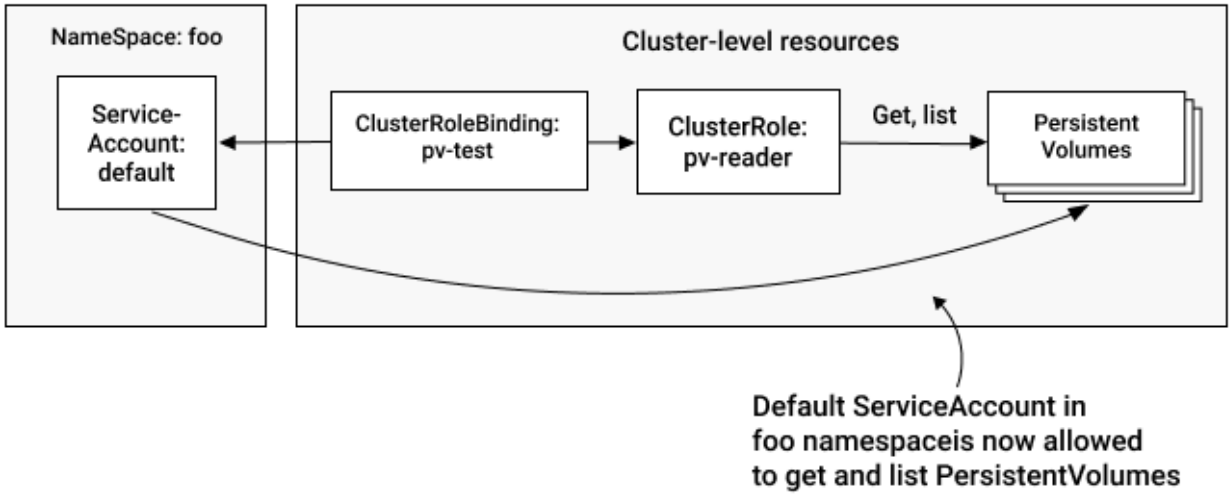
В этом случае sa default из ns bar получит возможность читать сервисы ns foo.



Похожим образом создаются ClusterRoleBinding. Только не указывайте RoleBinding на CLusterRole, это не сработает. ClusterRole — это non-namespaced объект, и привязать его к namespaced ServiceAccount через namespaced RoleBinding не получится.



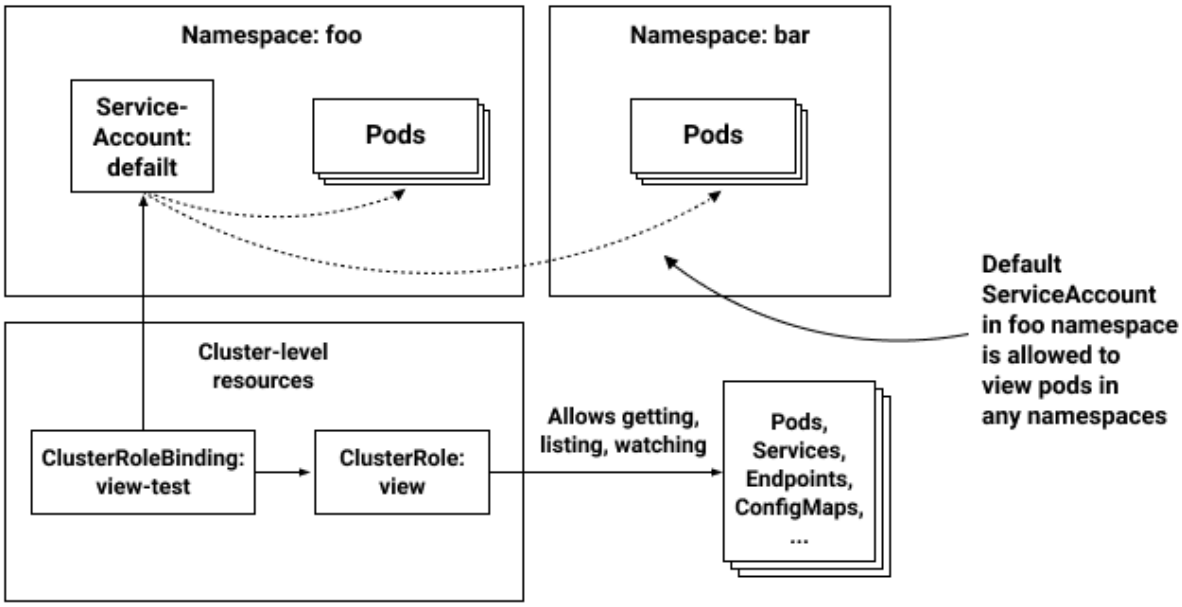
Используйте ClusterRoleBinding.

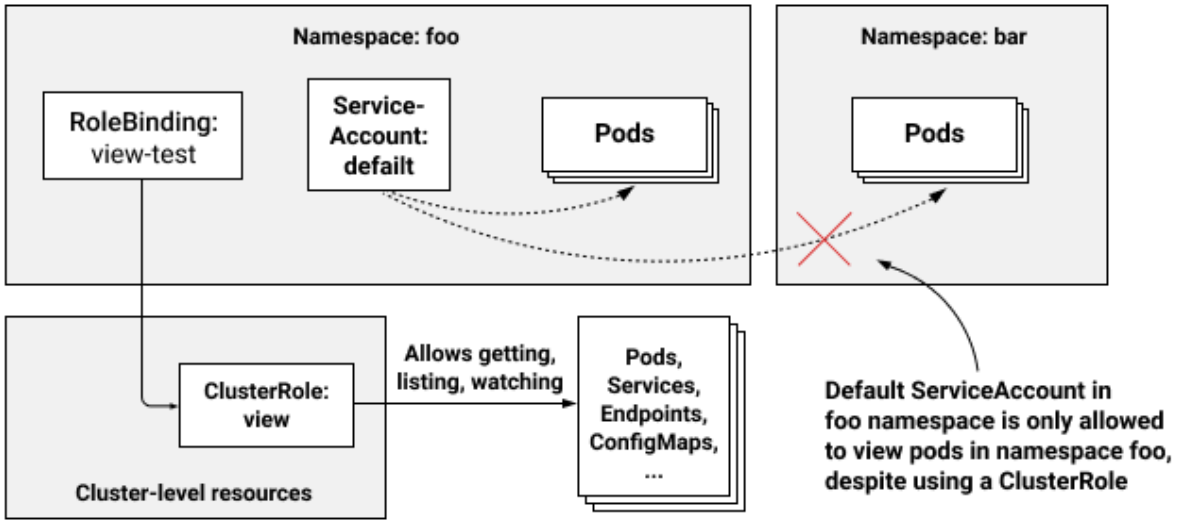


For accessing	Role type to use	Binding type to use
Cluster-level resources (Nodes, PersistentVolumes, ...)	ClusterRole	ClusterRoleBinding
Non-resource URLs (/api, /healtthz, ...)	ClusterRole	ClusterRoleBinding
Namespaced resources in any namespace (and across all namespaces)	ClusterRole	ClusterRoleBinding
Namespaced resources in a specific namespace (reusing the same ClusterRole in multiple namespaces)	ClusterRole	RoleBinding
Namespaced resources in a specific namespace (Role must be defined in each namespace)	Role	RoleBinding

Взгляните на роль kubectl get clusterrole system:discovery -o yaml. Вместо указания объектов доступа прописаны URL. Подобным способом также можно создавать роли и

предоставлять доступ. Только немного изменятся действия — get, post, patch. Конечно, для данной роли есть binding — kubectl get clusterrolebinding system:dicsovery -o yaml. Посмотрите, что в списке субъектов привязки. Group system:authenticated и Group system:unauthenticated. Даже неавторизованный пользователь может получить доступ к этим URL. Проверьте curl -k https://$(minikube ip):8443/api. ClusterRole не обязательно предоставляет доступ к ресурсам кластера, также может описывать namespaced ресурсы. Например, kubectl get clusterrole view -o yaml. Данная кластерная роль ведет себя по-разному в зависимости от привязки. Попробуйте выполнить из пода curl localhost:8001/api/v1/pods или curl localhost:8001/api/v1/namespaces/foo/pods. А теперь привяжите роль kubectl create clusterrolebinding view-test --clusterrole=view --serviceaccount=foo:default. Что вы получите? Попробуйте заменить clusterrolebinding на rolebinding, какой будет результат?





Полезные ссылки:

- Using RBAC authorization

Задание:

1. Получите список ClusterRoles и опишите отличия между ролями edit и admin.
2. Создайте цепочку pod-sa-clusterrolebinging-clusterrole с описанием доступа к endpoints в кластере.
3. Приведите описания созданных сущностей в yaml формате в качестве ответа.