Access control

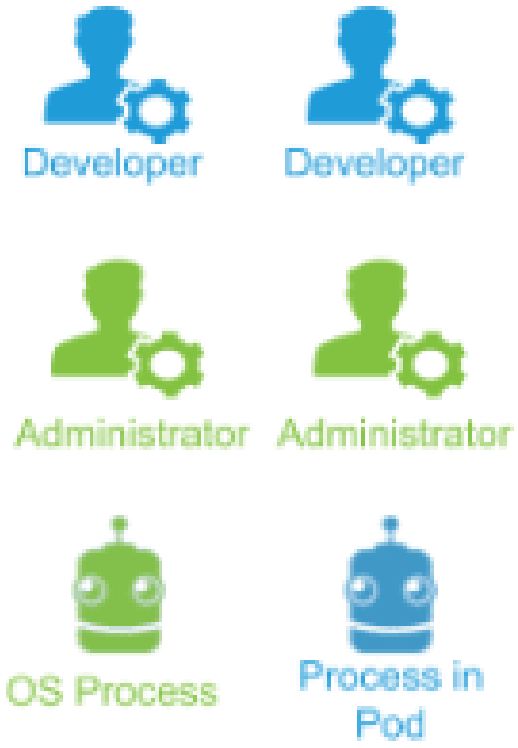
RBAC
role based access control

A Role defines an allowed operation (or set of)

Are assigned to accounts

Role
Consists of 2 things, what object-type and the allowed operation
In Kubernetes terms:

- Verb: get, list, create..
- Nouns: Pods, Volumes, Services..



Subjects

ConfigMaps
Pod
Service
AutoScaler
Deployment
Secrets
Ingress
PV
ReplicaSets
Namespace
DaemonSet
CronJob
Job
PVC
Nodes

API Resources

list
get
create
watch
delete
patch

Operations (Verbs)

RBAC

Roles need to be connected to users/groups

2 types of resources

- Namespace resources
  - RoleBinding
- Cluster resources
  - ClusterRoleBinding (entire cluster)

Role example

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

Role example

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: pod-reader
  namespace: default
subjects:
- kind: User
  name: jane
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role #Role or ClusterRole
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

ClusterRole

```
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"]
```

ClusterRoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: read-secrets-global
subjects:
- kind: Group
  name: manager # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

Role aggregation

Kubernetes comes with default cluster-roles:

- **cluster-admin** — Allows super-user access to perform any action on any resource
- **admin** — Full admin access, intended to be granted within a namespace
- **edit —** Edit access except for editing RBAC
- **view** — Allows view for most of objects

We don’t want to update these, but we can add permissions to them by aggregating.

Role aggregation

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: aggregate-cron-tabs-edit
  labels:
    # Add these permissions to the "admin" and "edit" default roles.
    rbac.authorization.k8s.io/aggregate-to-admin: "true"
    rbac.authorization.k8s.io/aggregate-to-edit: "true"
rules:
- apiGroups: ["stable.example.com"]
  resources: ["crontabs"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

Role aggregation

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: aggregate-cron-tabs-view
  labels:
    # Add these permissions to the "view" default role.
    rbac.authorization.k8s.io/aggregate-to-view: "true"
rules:
- apiGroups: ["stable.example.com"]
  resources: ["crontabs"]
  verbs: ["get", "list", "watch"]
```

Service account

```
kubectl create rolebinding my-sa-view \
  --clusterrole=view \
  --serviceaccount=my-namespace:my-sa \
  --namespace=my-namespace

# if pods don’t have a serviceaccount specified, default is used
```

Users in Kubernetes

2 types:

- service accounts managed by Kubernetes
- normal users

Authentication strategies

- X509 Client Certs
- Static Token File
- Oauth/OpenID provder

X509 Client Certificate

Add the certificate authority to:
--client-ca-file=SOMEFILE in kube-apiserver configuration

For MicroK8s:

```
#kube-apiserver config
sudo vim /var/snap/microk8s/current/args/kube-apiserver
sudo systemctl restart snap.microk8s.daemon -
apiserver.service
```

Static token file

Kube-apiserver configuration:
--token-auth-file=SOMEFILE

File should be CSV, with these columns (min first 3):
token, username, user uuid, groups

Example:
token,user,uid,"group1,group2,group3“

Requests should have ‘Authorization: Bearer <token>’ in the headers

Static Password file

Kube-apiserver config: **--basic-auth-file=SOMEFILE**

CSV file with:
password,user,uid,"group1,group2,group3“

Authorization: Basic BASE64ENCODED(USER:PASSWORD)

Service account tokens

--service-account-key-file
A file containing a PEM encoded key for signing bearer tokens. If unspecified, the API server's TLS private key will be used.

--service-account-lookup
If enabled, tokens which are deleted from the API will be revoked.

Service accounts

```
kubectl create serviceaccount nginx
kubectl get serviceaccounts jenkins -o yaml


apiVersion: v1
kind: ServiceAccount
metadata:
  # ...
secrets:
- name: jenkins-token-1yvwg
```

Service accounts

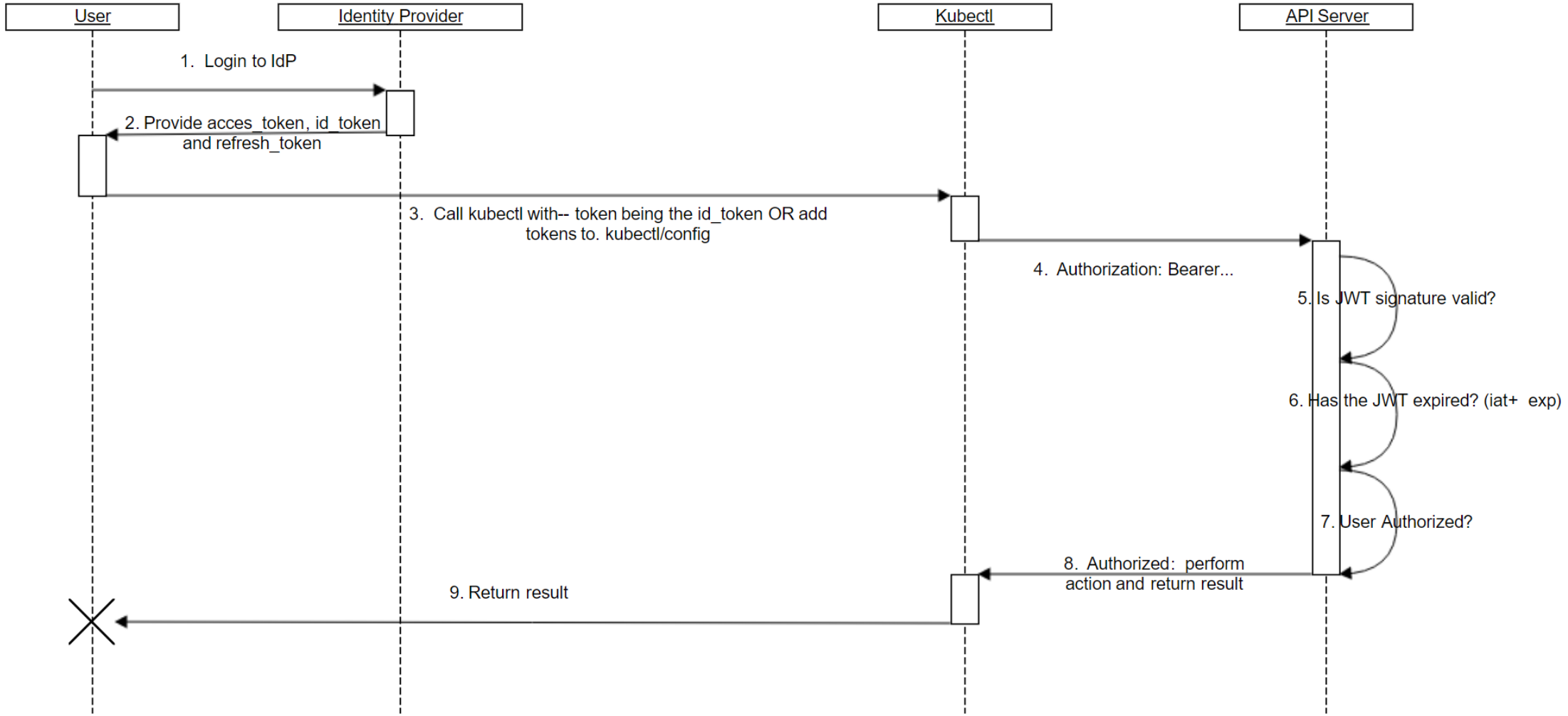
```
kubectl get secret jenkins-token-1yvwg -o yaml


apiVersion: v1
data:
  ca.crt: (APISERVER'S CA BASE64 ENCODED)
  namespace: ZGVmYXVsdA==
  token: (BEARER TOKEN BASE64 ENCODED)
kind: Secret
metadata:
  # ...
type: kubernetes.io/service-account-token
```

OpenID Connect Tokens

OpenID Connect is a flavor of OAuth2

- supported by some OAuth2 providers, notably:
- Azure Active Directory,
- Salesforce,
- Google

OpenID Connect Tokens



OpenID Connect

1. Login to your identity provider
2. Your identity provider will provide you with an access_token, id_token and a refresh_token
3. When using kubectl, use your id_token with the --token flag or add it directly to your kubeconfig
4. kubectl sends your id_token in a header called Authorization to the API server
5. The API server will make sure the JWT signature is valid by checking against the certificate named in the configuration
6. Check to make sure the id_token hasn’t expired
7. Make sure the user is authorized
8. Once authorized the API server returns a response to kubectl
9. kubectl provides feedback to the user

Parameter	Description	Example	Required
`--oidc-issuer-url`	URL of the provider which allows the API server to discover public signing keys. Only URLs which use the `https://` scheme are accepted. This is typically the provider's discovery URL without a path, for example "https://accounts.google.com" or "https://login.salesforce.com". This URL should point to the level below .well-known/openid-configuration	If the discovery URL is `https://accounts.google.com/.well-known/openid-configuration`, the value should be `https://accounts.google.com`	Yes
`--oidc-client-id`	A client id that all tokens must be issued for.	kubernetes	Yes
`--oidc-username-claim`	JWT claim to use as the user name. By default `sub`, which is expected to be a unique identifier of the end user. Admins can choose other claims, such as `email` or `name`, depending on their provider. However, claims other than `email` will be prefixed with the issuer URL to prevent naming clashes with other plugins.	sub	No
`--oidc-username-prefix`	Prefix prepended to username claims to prevent clashes with existing names (such as `system:` users). For example, the value `oidc:` will create usernames like `oidc:jane.doe`. If this flag isn't provided and `--oidc-user-claim` is a value other than `email` the prefix defaults to `( Issuer URL )#` where `( Issuer URL )` is the value of `--oidc-issuer-url`. The value `-` can be used to disable all prefixing.	`oidc:`	No
`--oidc-groups-claim`	JWT claim to use as the user's group. If the claim is present it must be an array of strings.	groups	No
`--oidc-groups-prefix`	Prefix prepended to group claims to prevent clashes with existing names (such as `system:` groups). For example, the value `oidc:` will create group names like `oidc:engineering` and `oidc:infra`.	`oidc:`	No
`--oidc-required-claim`	A key=value pair that describes a required claim in the ID Token. If set, the claim is verified to be present in the ID Token with a matching value. Repeat this flag to specify multiple claims.	`claim=value`	No
`--oidc-ca-file`	The path to the certificate for the CA that signed your identity provider's web certificate. Defaults to the host's root CAs.	`/etc/kubernetes/ssl/kc-ca.pem`	No

Setup kubectl to use OpenID Connect

```
kubectl config set-credentials USER_NAME \
   --auth-provider=oidc \
   --auth-provider-arg=idp-issuer-url=( issuer url ) \
   --auth-provider-arg=client-id=( your client id ) \
   --auth-provider-arg=client-secret=( your client secret ) \
   --auth-provider-arg=refresh-token=( your refresh token ) \
   --auth-provider-arg=idp-certificate-authority=( path to cacert ) \
   --auth-provider-arg=id-token=( your id_token )
```

```
apiVersion: audit.k8s.io/v1beta1
kind: Policy
omitStages:
  - "RequestReceived"
rules:
  - level: Request
    users: ["admin"]
    resources:
      - group: ""
        resources: ["*"]
  - level: Request
    user: ["system:anonymous"]
    resources:
      - group: ""
        resources: ["*"]
```