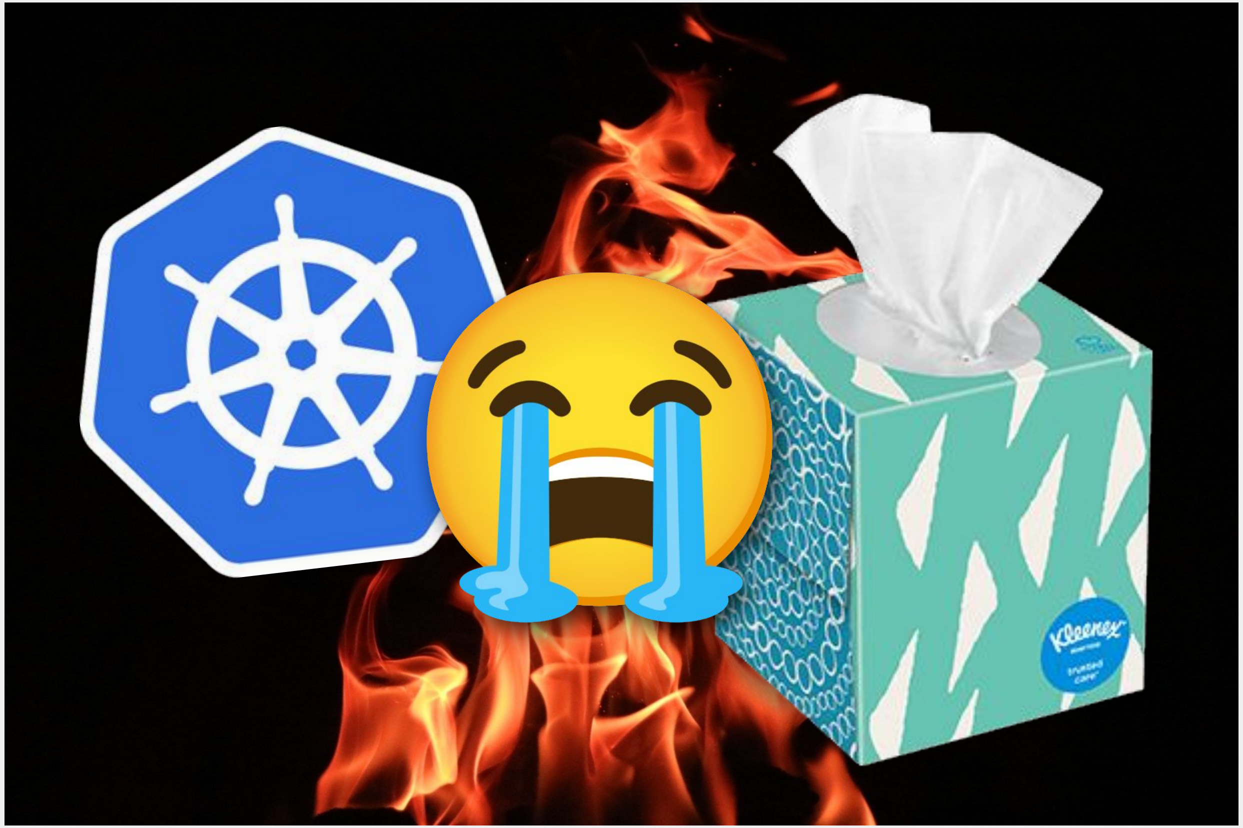
leigh capili

@capileigh

stealthybox

< Performance Art >

@capileigh stealthybox



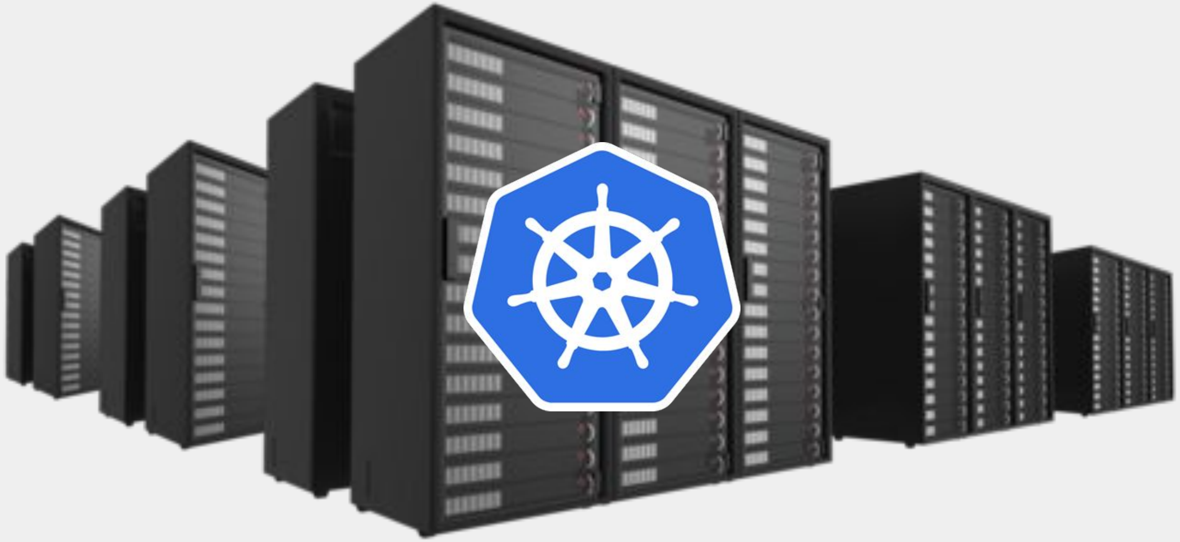
@capileigh stealthybox

Can I get some access?

How the Basics of RBAC Scale for Organizations

@capileigh stealthybox

@capileigh stealthybox



@capileigh stealthybox

Deploy Apps

Setup Networks

Manage Lots of Servers

Enforce Policy

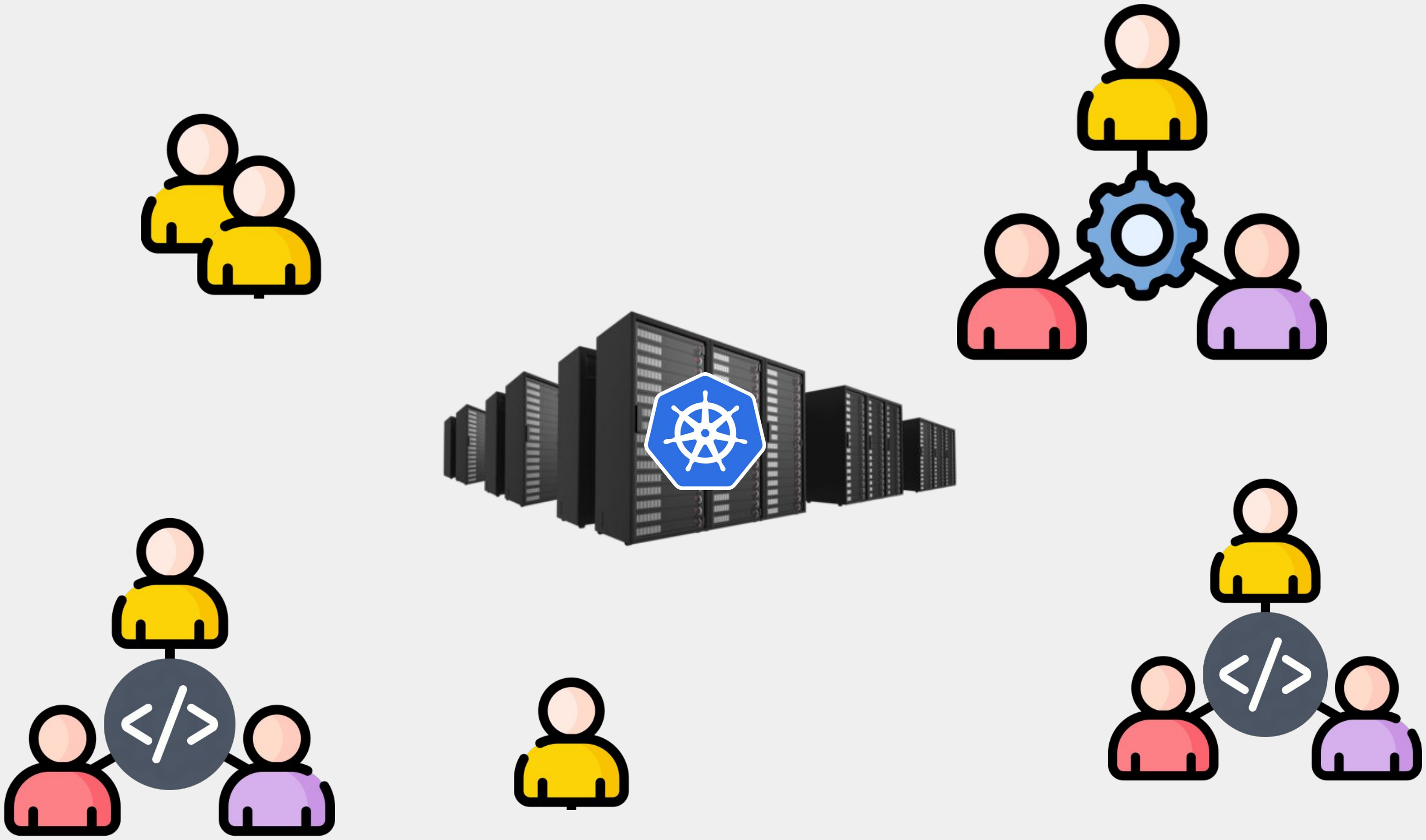
Deployment

Service

Node

NetworkPolicy

@capileigh stealthybox



@capileigh stealthybox

RBAC

Role-Based Access Control

@capileigh stealthybox

When we **login** to a computer
the computer needs to

1. know **who** we are
2. grant what we have **access** to

1. know who we are

Authentication

2. grant what we have access to

Authorization

1. know who we are

Authe N tication

2. grant what we have access to

Authori Z ation

1. know who we are

AuthN

2. grant what we have access to

AuthZ

Role-Based Access Control

is

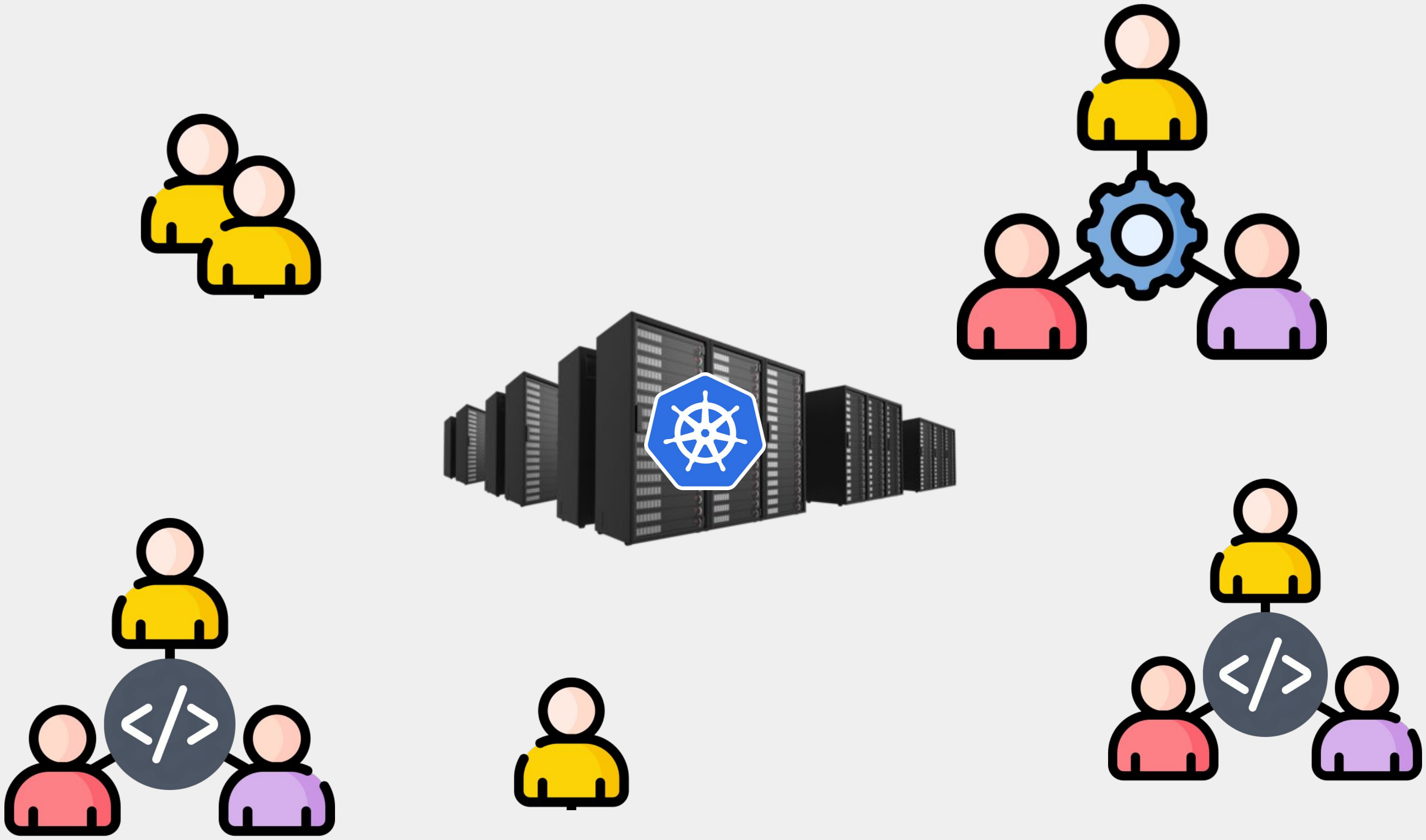
Authorization

What are *people or apps* able to access?

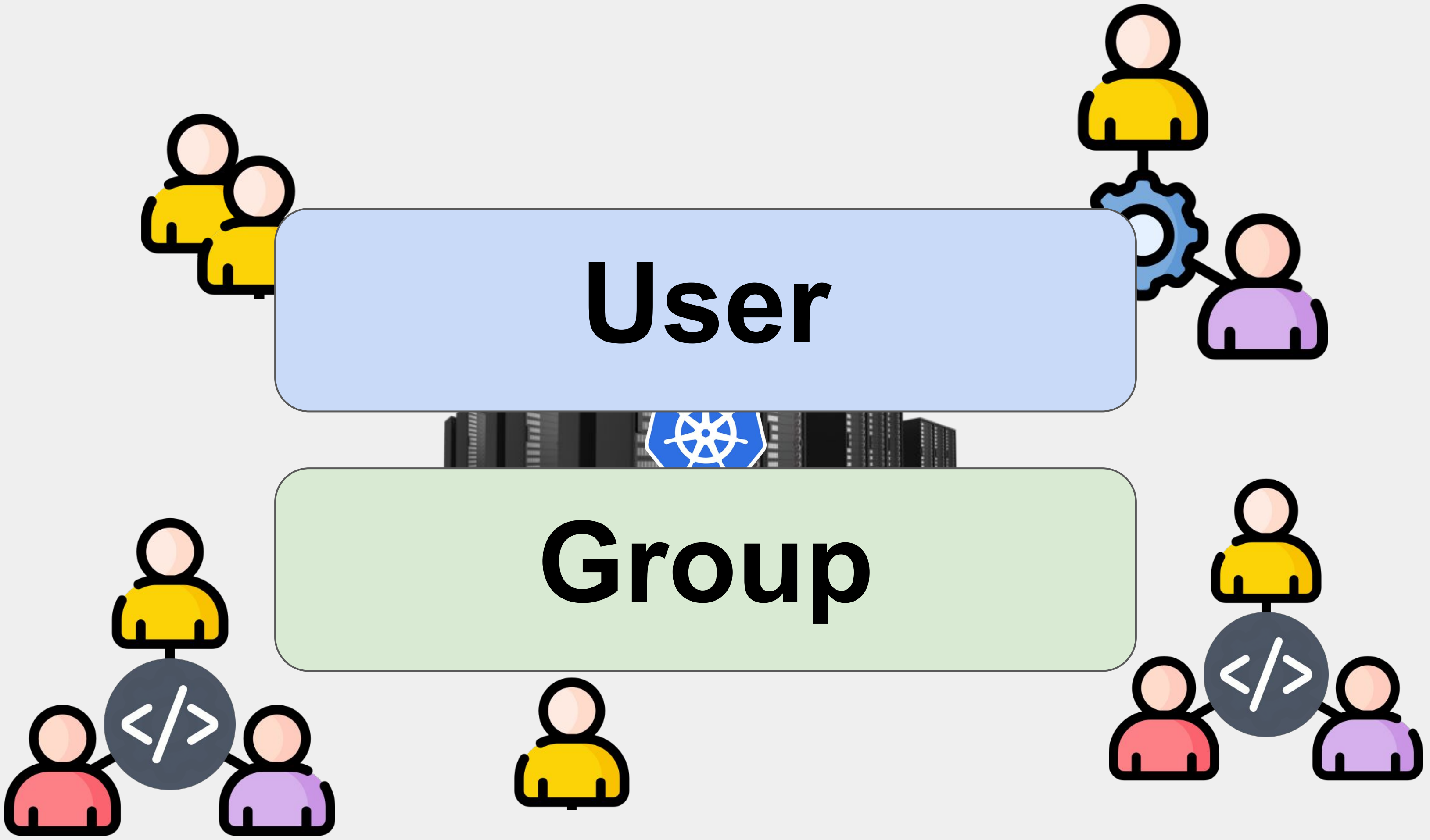
@capileigh stealthybox

What are *people or apps* able to access?

Subjects



@capileigh stealthybox

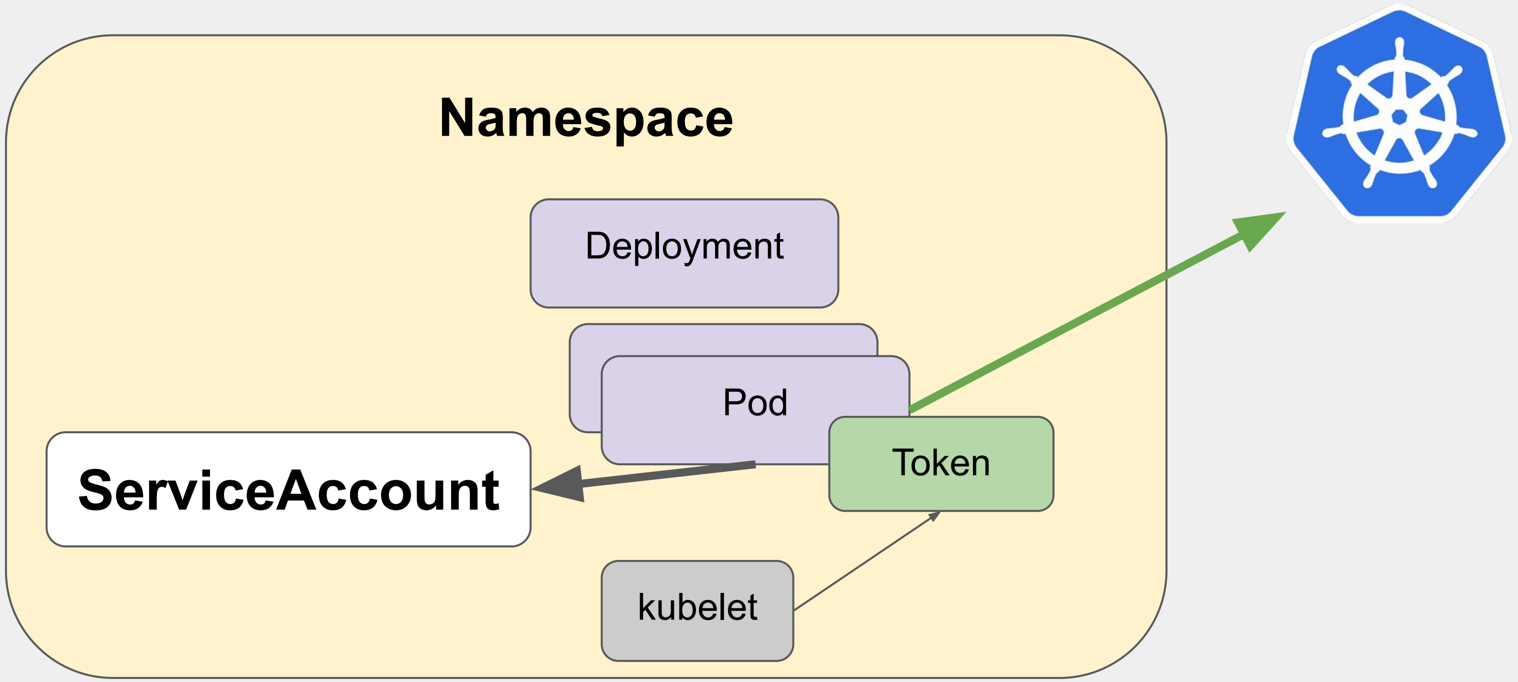


User

Group

@capileigh stealthybox

ServiceAccounts are apps



@capileigh stealthybox

Subjects

People

- User
- Group

Apps

- ServiceAccount

Subjects

Subjects are the **who** in “who has access?”

Roles

Roles are used to describe **what** should be accessed

Roles

Roles have **rules**
that list what resources and verbs
are allowed

Roles

list allowed API's

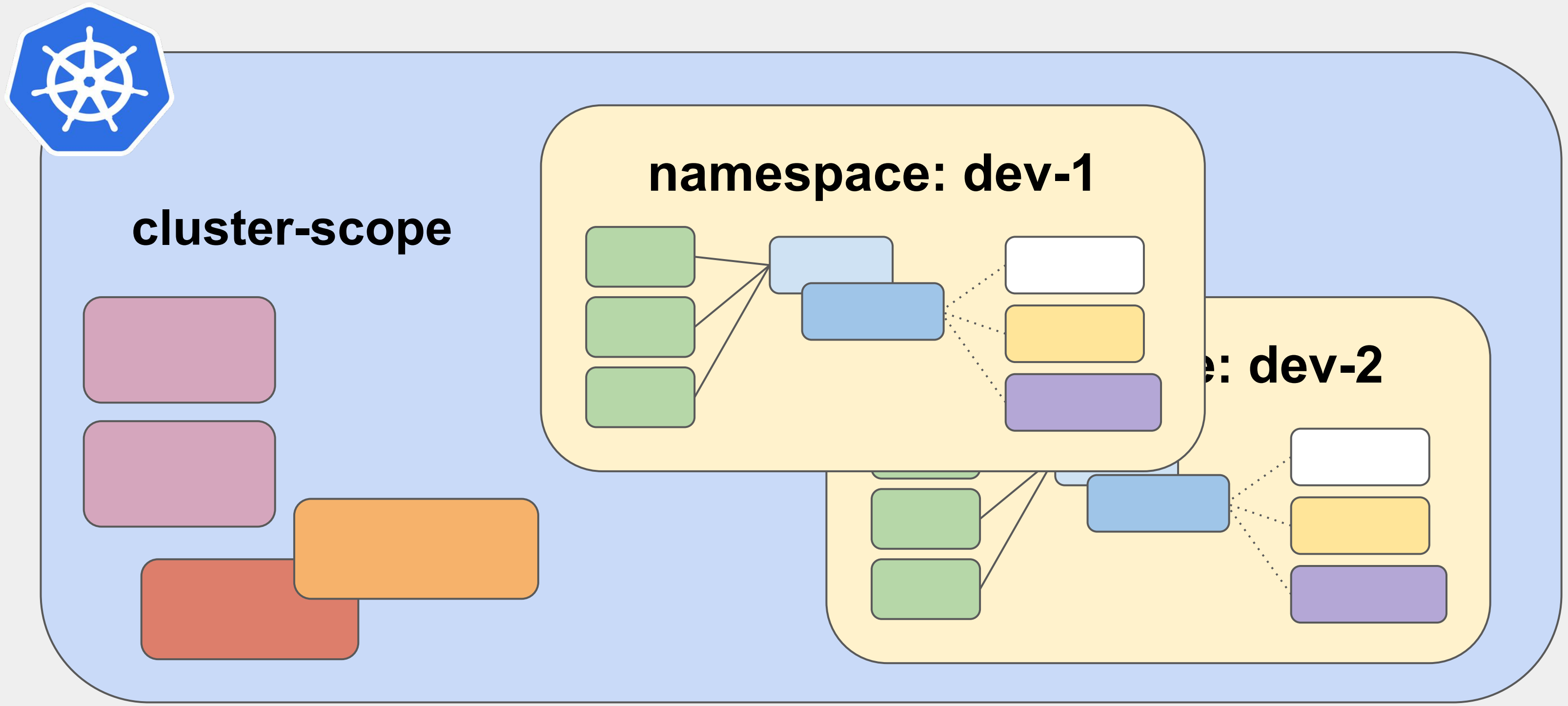
    list verbs allowed for each one

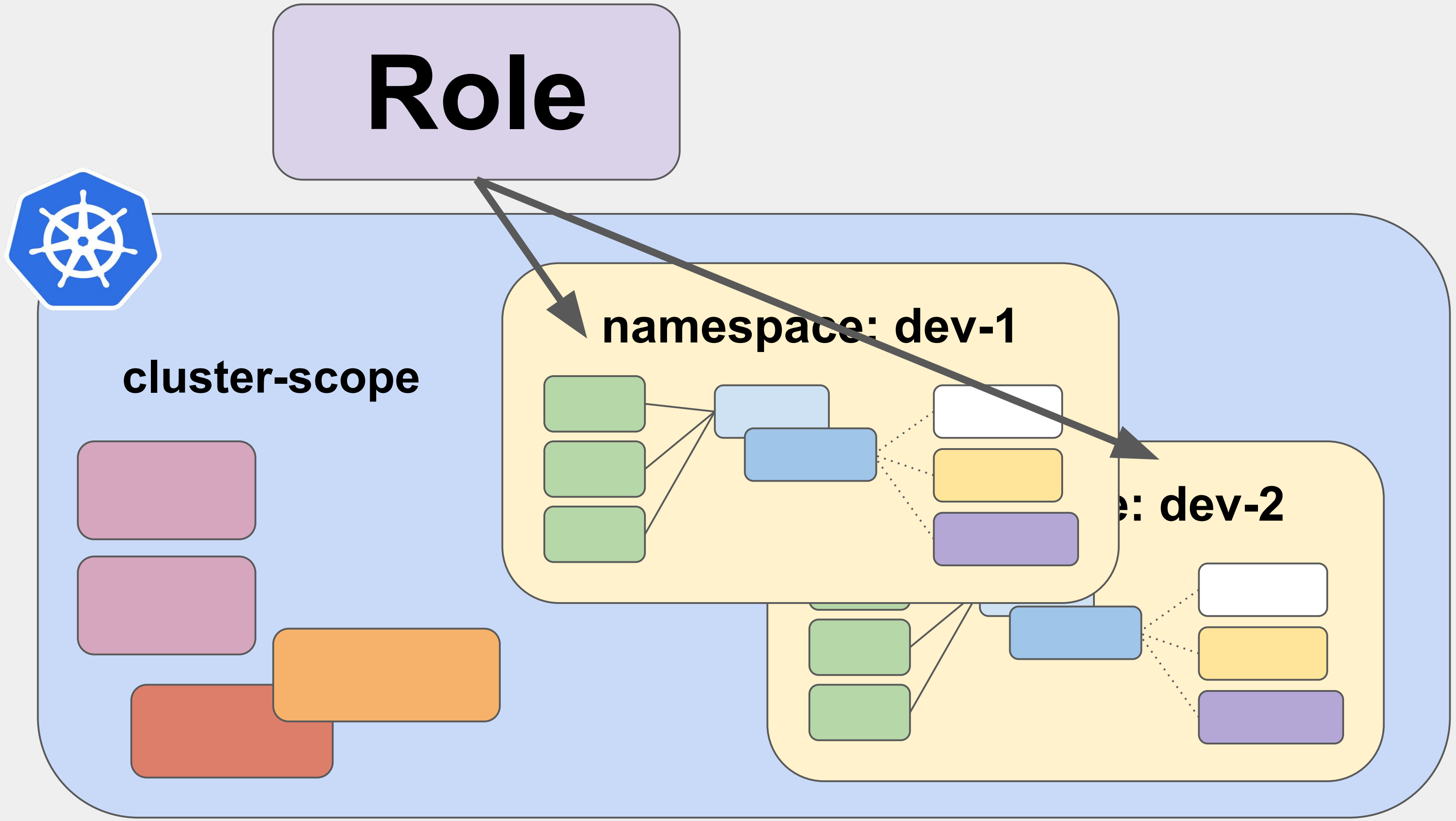
Roles live in a Namespace
Roles list namespaced resources

ClusterRoles can list

- namespaced resources
- cluster-scoped resources

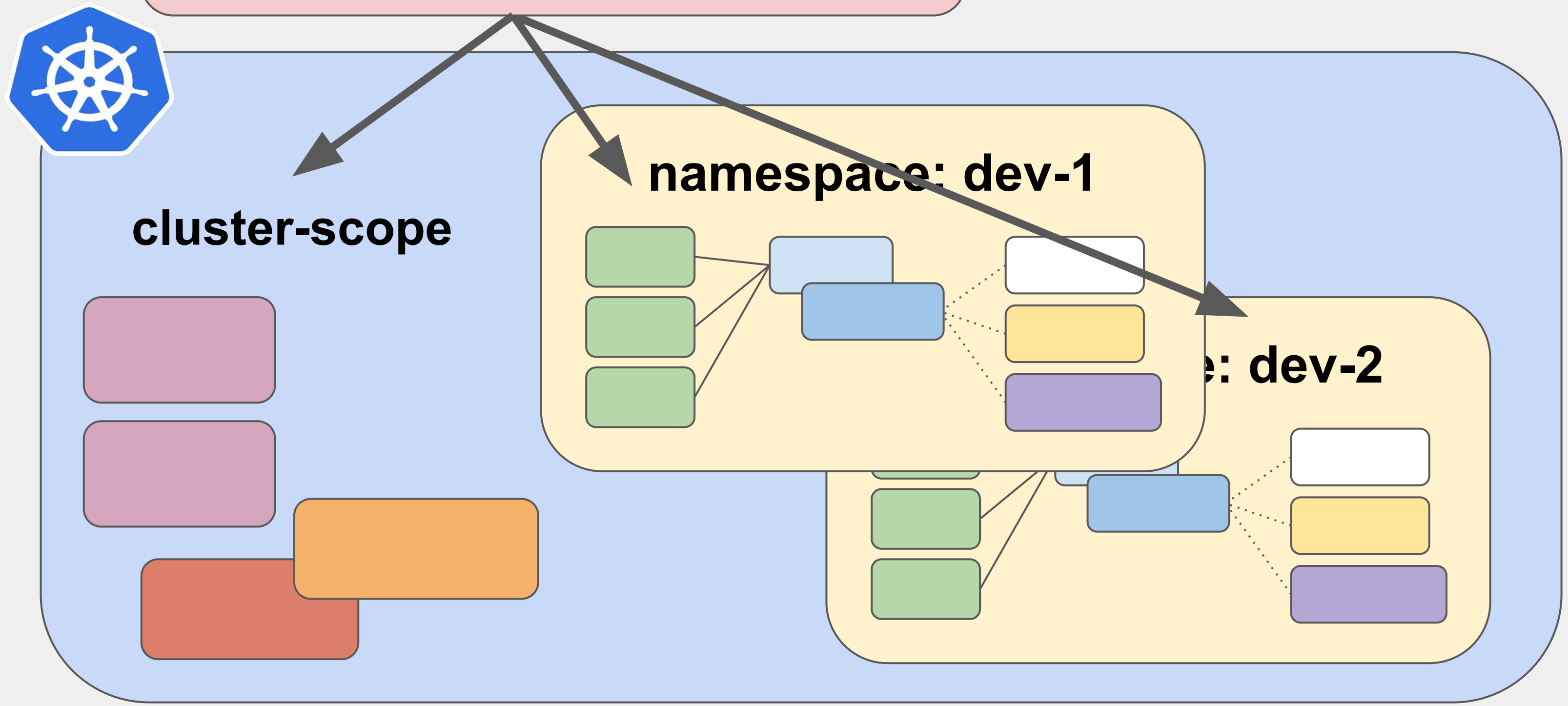
Namespaces





@capileigh stealthybox

ClusterRole

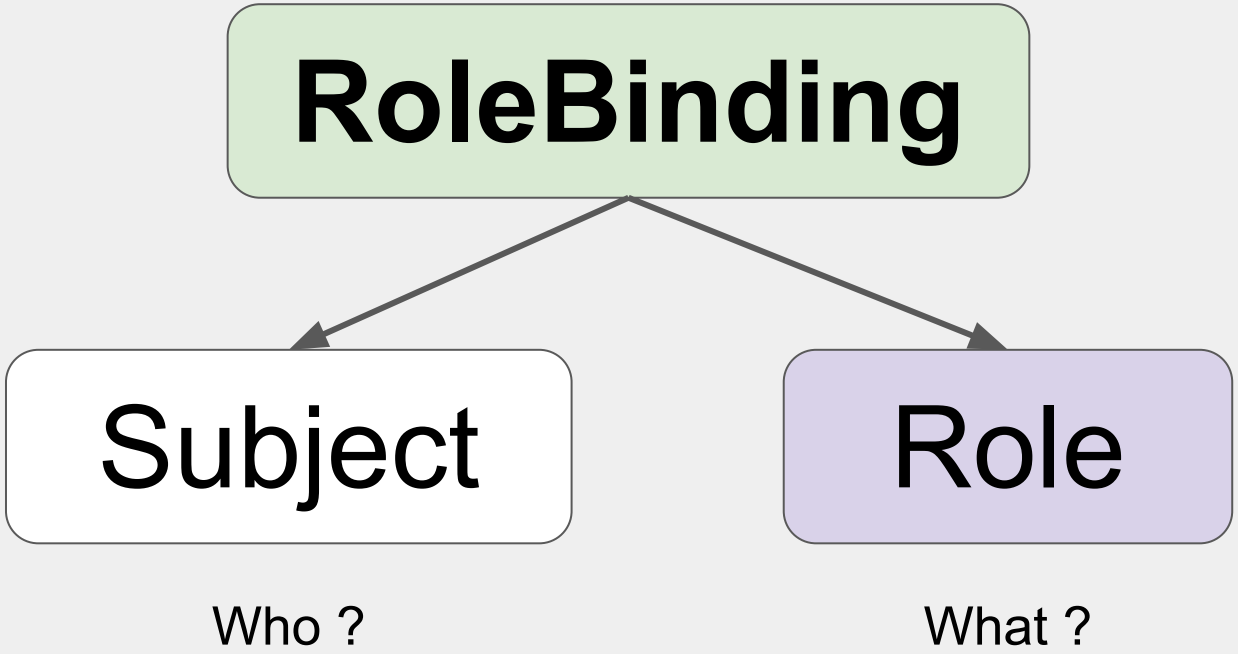


Subject

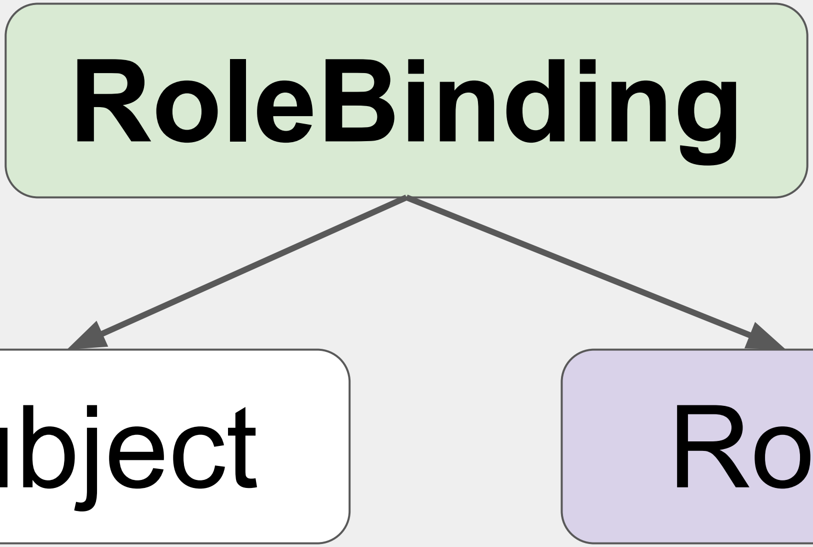
Who

Role

What


RoleBinding
Subject
Role
Who ?
What ?

Authorize the Subject to the Role's access **rules**



RoleBindings

RoleBinding + Role

    must be within the same namespace

    grants access:

        - *resources within the namespace*

ClusterRoleBinding

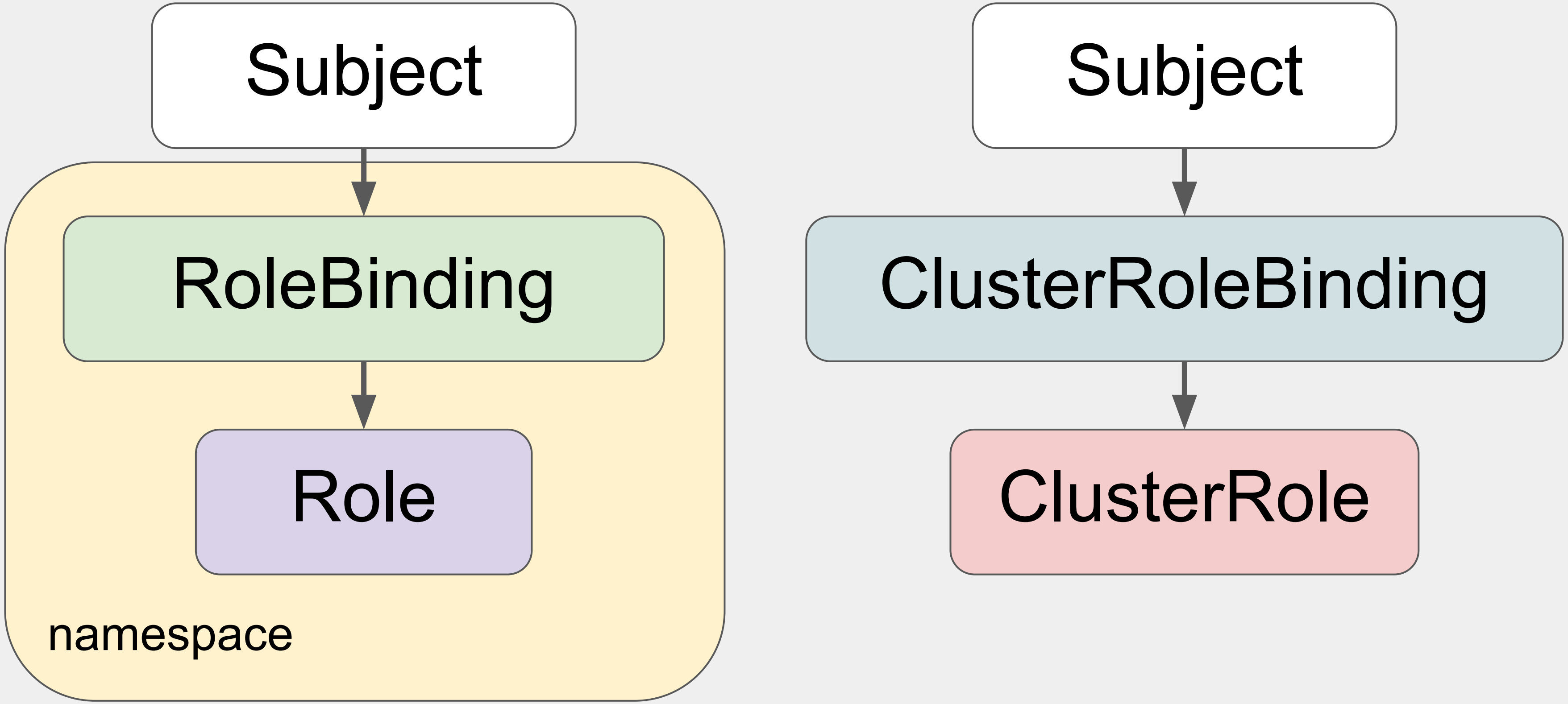
uses only ClusterRoles

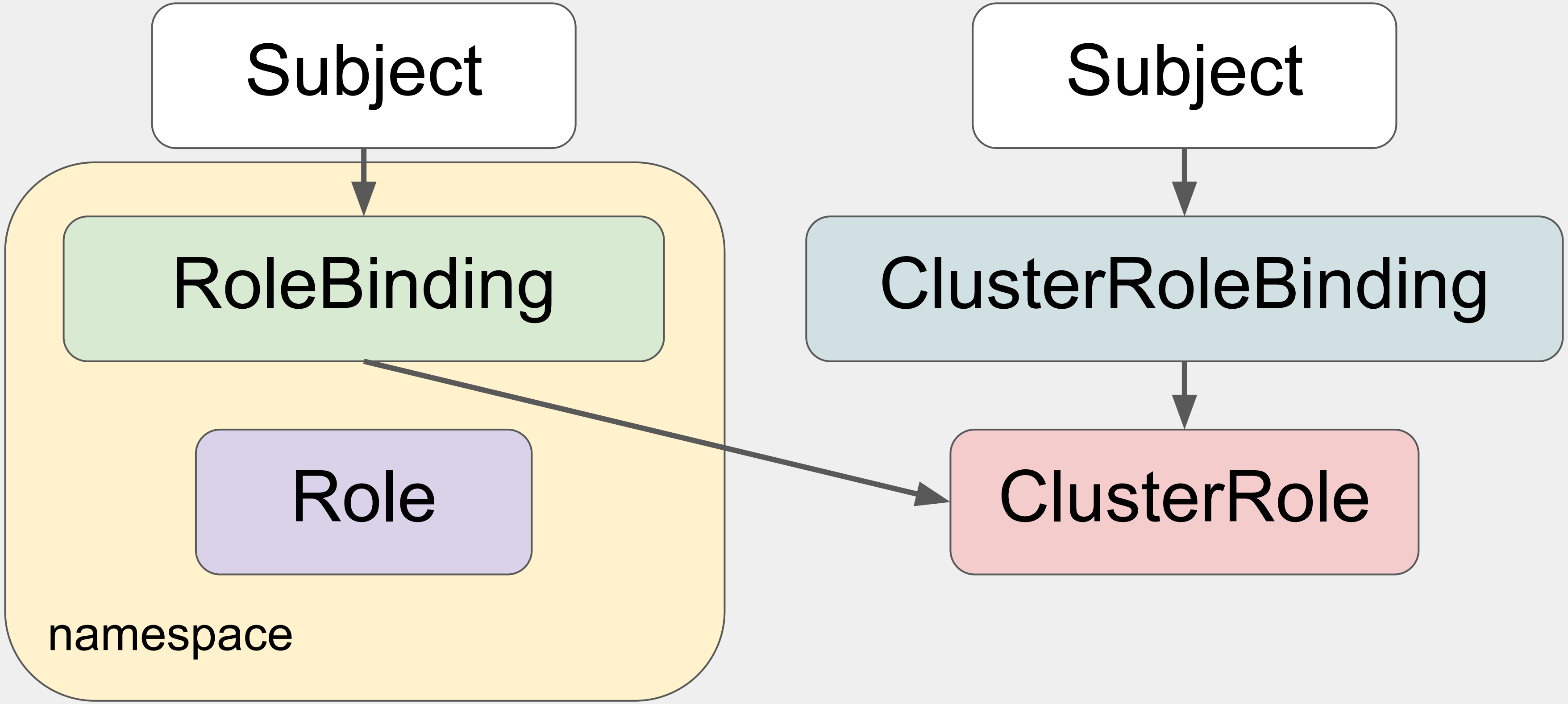
    grants access:

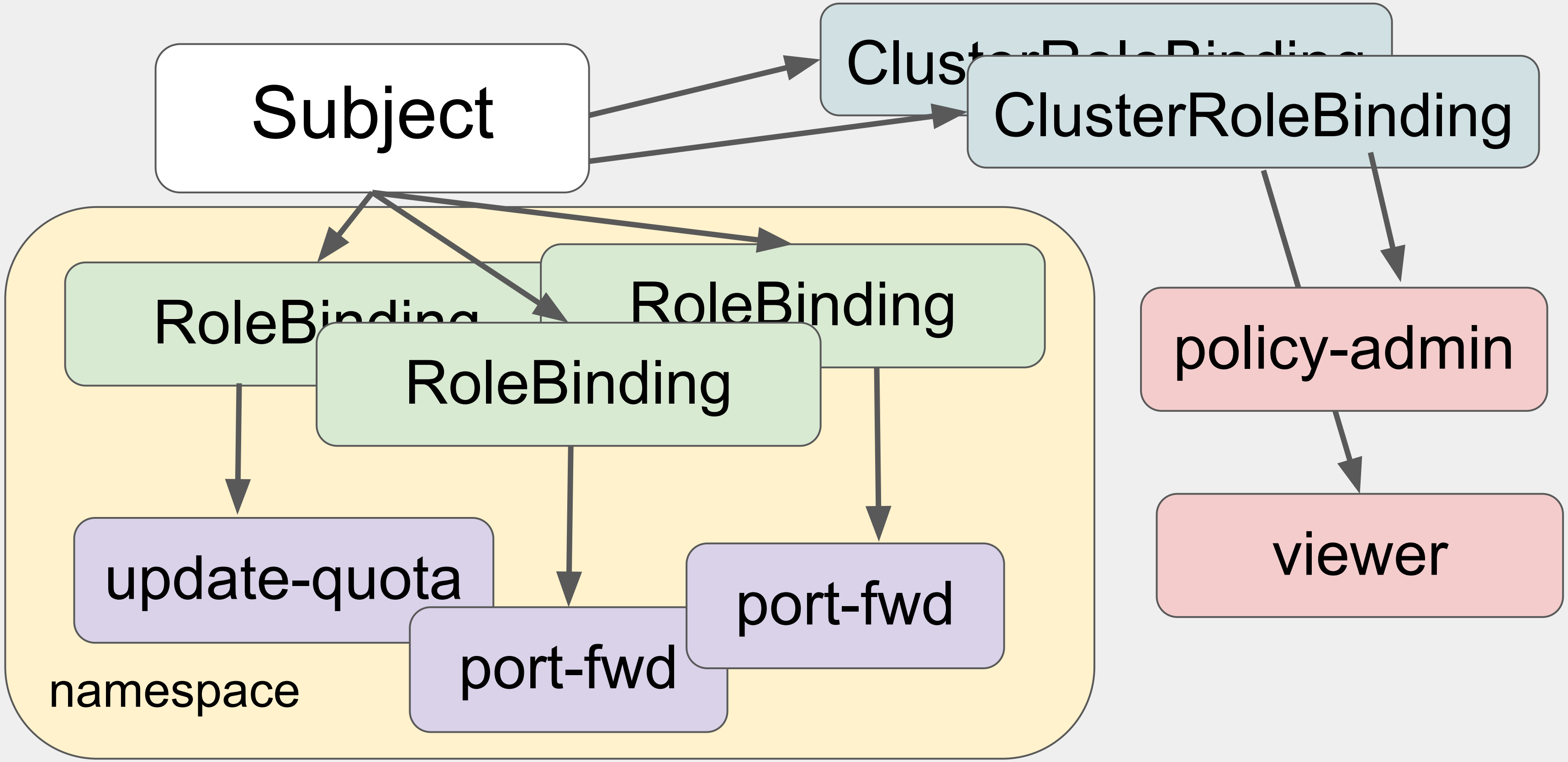
        - *resources across all namespaces*

        - *resources at cluster-scope*

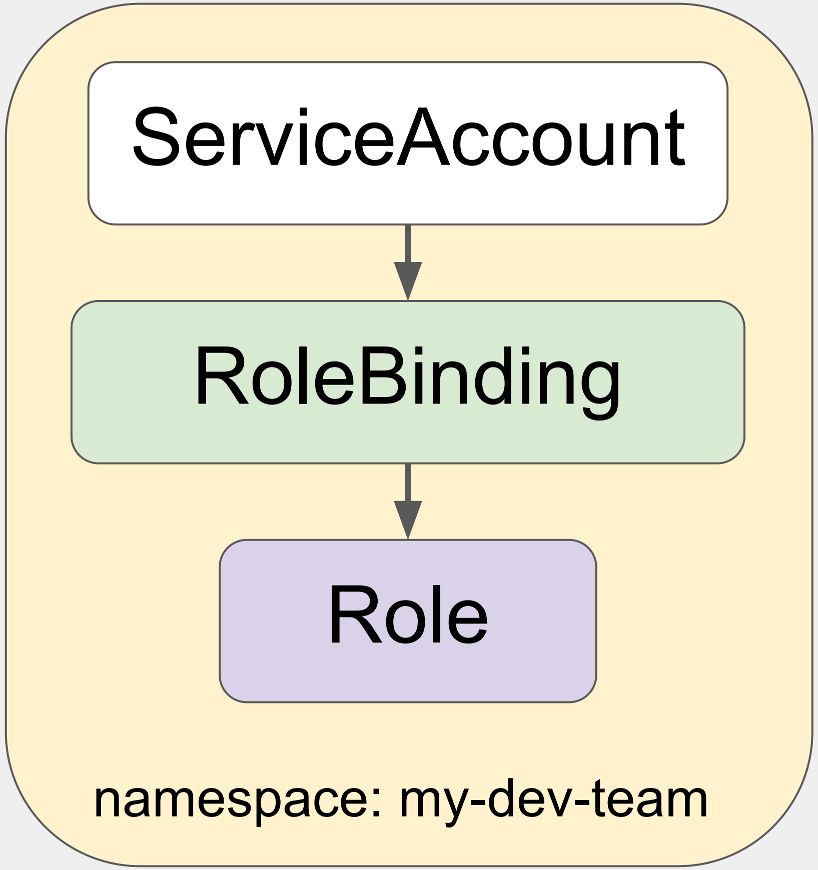
Misconceptions

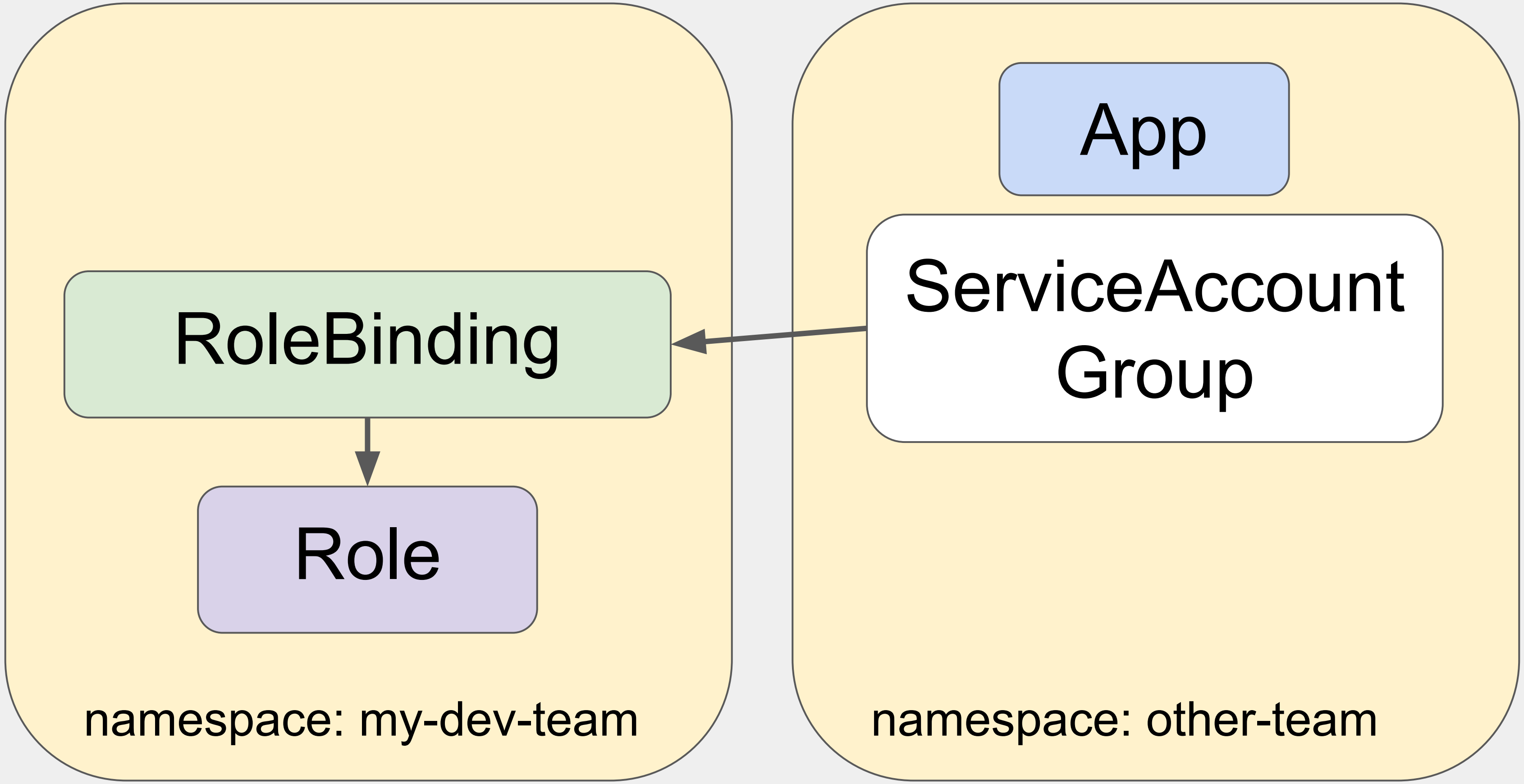

Subject
RoleBinding
Role
namespace
Subject
ClusterRoleBinding
ClusterRole


Subject
RoleBinding
Role
namespace
Subject
ClusterRoleBinding
ClusterRole



@capileigh stealthybox


ServiceAccount
RoleBinding
Role
namespace: my-dev-team


App
ServiceAccount Group
RoleBinding
Role
namespace: my-dev-team
namespace: other-team

ServiceAccounts are just **Users** with formatted names

```
system:serviceaccount:{{namespace}}:{{name}}
```

@capileigh stealthybox

Service Accounts also get **Groups**

```
system:serviceaccounts
system:serviceaccounts:{{namespace}}
```

Underused Features

resourceNames

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
 namespace: default
 name: configmap-updater
rules:
  verbs: ["update", "get"]
- apiGroups: [""]
  resources: ["configmaps"]
  resourceNames: ["my-configmap"]
```

@capileigh stealthybox

create with resourceNames (only works /w server-side apply)

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
 namespace: default
 name: configmap-updater
rules:
  verbs: ["create", "update", "get"]
- apiGroups: [""]
  resources: ["configmaps"]
  resourceNames: ["my-configmap"]
```

@capileigh stealthybox

ClusterRole Aggregation

sub-resources

namespaces/**finalize**
namespaces/**status**
nodes/**proxy**
nodes/**status**
persistentvolumeclaims/**status**
persistentvolumes/**status**
pods/**attach**
pods/**binding**
pods/**eviction**
pods/**portforward**
pods/**proxy**
pods/**status**
replicationcontrollers/**scale**
replicationcontrollers/**status**
resourcequotas/**status**
serviceaccounts/**token**
services/**proxy**
services/**status**

weird, virtual verbs

bind

escalate

impersonate

Impersonation

RBAC & your teams

@capileigh stealthybox

Authentication

Plain Kubernetes is ready for your **Apps**,
but it's **not ready** to authenticate **People**.

Integrate your cluster's login with what you **already use** for Identity

I don't recommend using Certificate auth or ServiceAccounts outside the cluster for general usage by People

Use the “**bootstrap**” **ClusterRoles.**

Use groups!

Try to use an IdP that supports groups.

/w OIDC this is the "groups claim"

⚒️

Tools -> Habits

📆

Use declarative configs

GitOps your Access Control

Transparency
Onboarding
Org Changes
Emergencies

Use declarative configs

GitOps your Access Control

Make sure you have proper process for changing and protecting these configs

Use declarative configs

GitOps your Access Control

“Who should access what” is some of the most important IP in your org.

Consider running automated **integration tests** that use **impersonation**

People (Users/Groups)
- have the access they need
- can't access things they shouldn't

App ServiceAccounts
- have the access they need
  - - within their Namespace
  - - in other needed Namespaces or cluster-scope
- can't access things they shouldn't

Consider running automated **integration tests** that use **impersonation**

```
kubectl auth can-i  \
    --as  user_or_svc_account
    --as-group  group
```

Limitations

list & **watch**
allow getting of every object.

You can't use resourceNames
to **filter** lists.

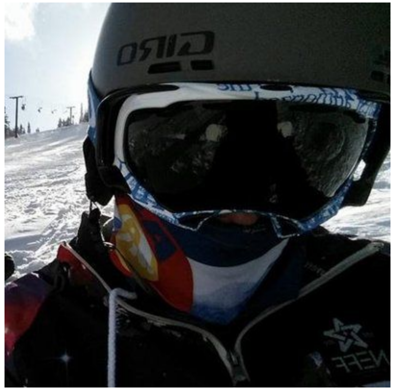
This is problematic for things like
Namespaces if you need secrecy

You can't use resourceNames
to **filter** lists.

If you need this consider
virtualizing the k8s API /w **vclusters**
or an **k8s API server proxy** like Clastix

resourceNames + the create verb only works with server-side apply

(Maybe you could use another Namespace?
Also see hierarchical namespace controller)



@capileigh

stealthybox