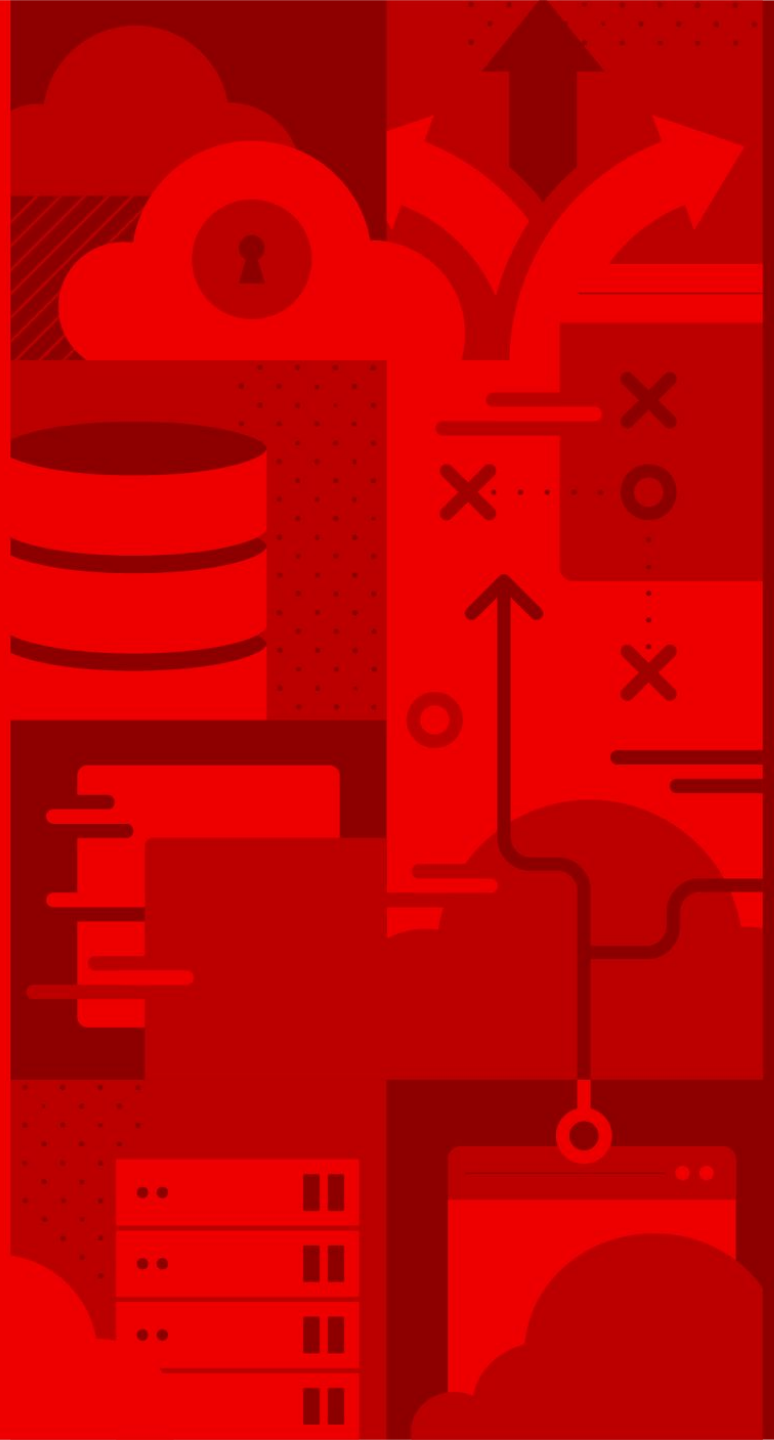
What we'll discuss today

Securing the API

- Understanding API Requests
- Users, Groups, and Service Accounts
- Roles and RoleBindings
- ClusterRoles and ClusterRoleBindings

Source:



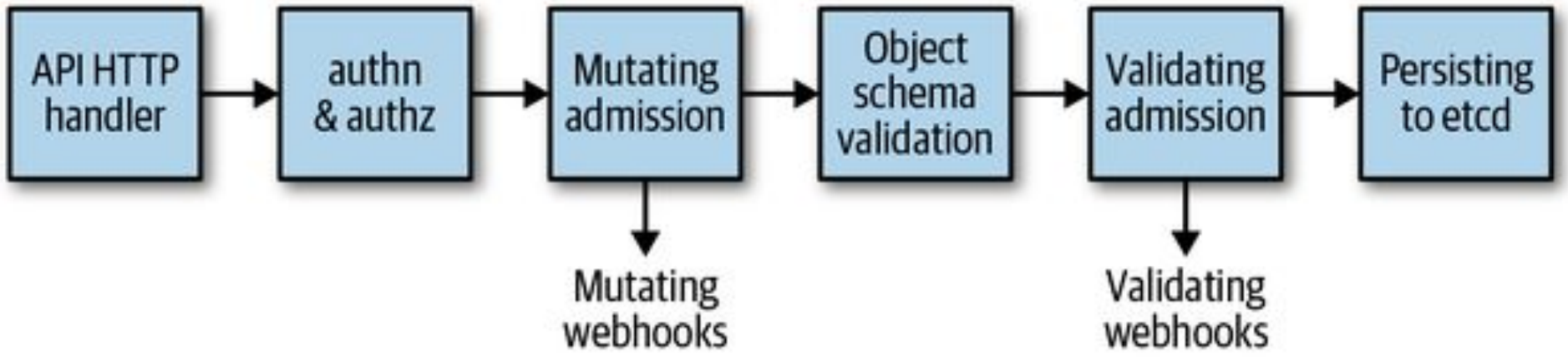
NO ONE **TALKS** TO THE API SERVER FOR **FREE**

Red Hat

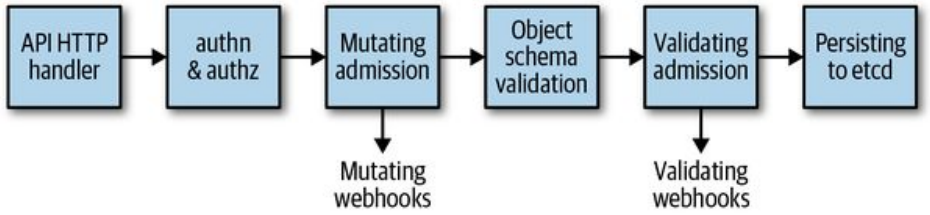
What happens when an HTTP request hits the Kubernetes API?

High Level Workflow

- HTTP request is processed by a chain of filters
- Depending on the HTTP path the request is routed to an appropriate handler
- The handler takes the request as well as context and retrieves as well as delivers object from etcd

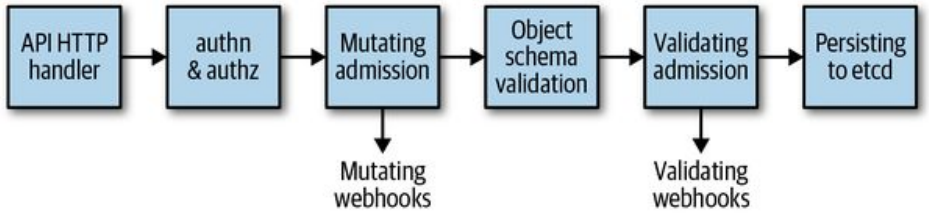

API HTTP handler
authn & authz
Mutating admission
Object schema validation
Validating admission
Persisting to etcd
Mutating webhooks
Validating webhooks

API HTTP handler and authn+authz



Chain of filters that either pass and attach respective information to the context or if filter does not pass return appropriate HTTP response; 401 if user auth failed for example.

Admission, Validation, and Persistence



Admission

- Mutate, validate, or both; mutate can set image pull policy, validation can verify namespace exists before creating object in respective namespace

Validation

- Objects are checked to ensure they adhere to validation logic; i.e. string formats are checked to verify only valid DNS-compatible characters are used

Persistence

- CRUD implementation; optimistic concurrency

Users

Managed by external systems such as Single Sign-On (SSO) as an example; cannot be created, updated, or deleted via the API server.

Groups

Allow granting permissions to more than one user/service account at once. There are several built-in groups that have special meaning.

Service Accounts

Allow an application/service/daemon running inside of a pod to authenticate itself with the API server.

Source:

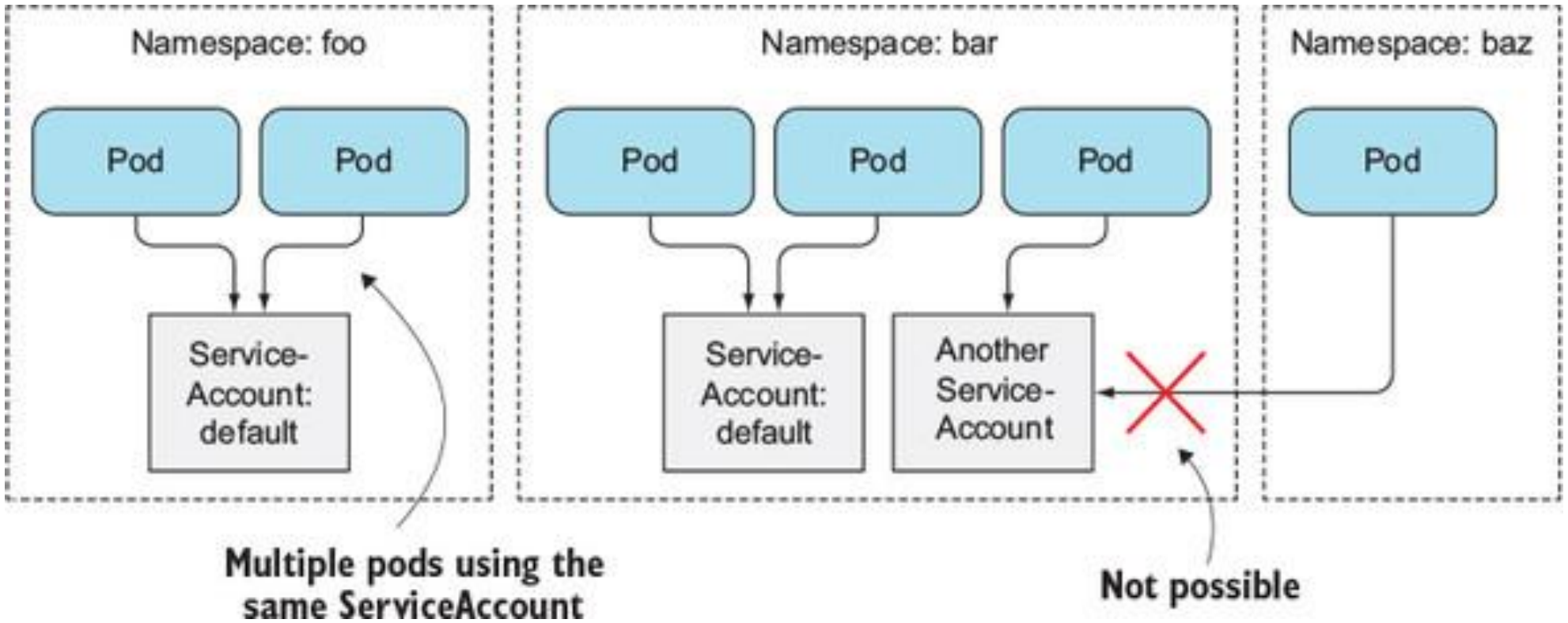
```
apiVersion: v1
kind: Config
preferences: {}

clusters:
- cluster:
    certificate-authority-data: ...
    server: https://lab.tld:6443
  name: lab

users:
- name: admin
  user:
    client-certificate-data: ...
    client-key-data: ...

contexts:
- context:
    cluster: lab
    user: admin
  name: owner
```

```
[root@node-0]# oc config get-contexts
CURRENT   NAME    CLUSTER   AUTHINFO   NAMESPACE
*         owner   lab       admin
          dev     prod      john
```

```
apiVersion: v1
kind: Config
preferences: {}

clusters:
- cluster:
    certificate-authority-data: ...
    server: https://lab.tld:6443
  name: lab
- cluster:
    certificate-authority-data: ...
    server: https://production.tld:6443
  name: prod
...
```

```
apiVersion: v1
kind: Config
preferences: {}

...

users:
- name: admin
  user:
    client-certificate-data: ...
    client-key-data: ...
users:
- name: john
  user:
    client-certificate-data: ...
    client-key-data: ...
...
```

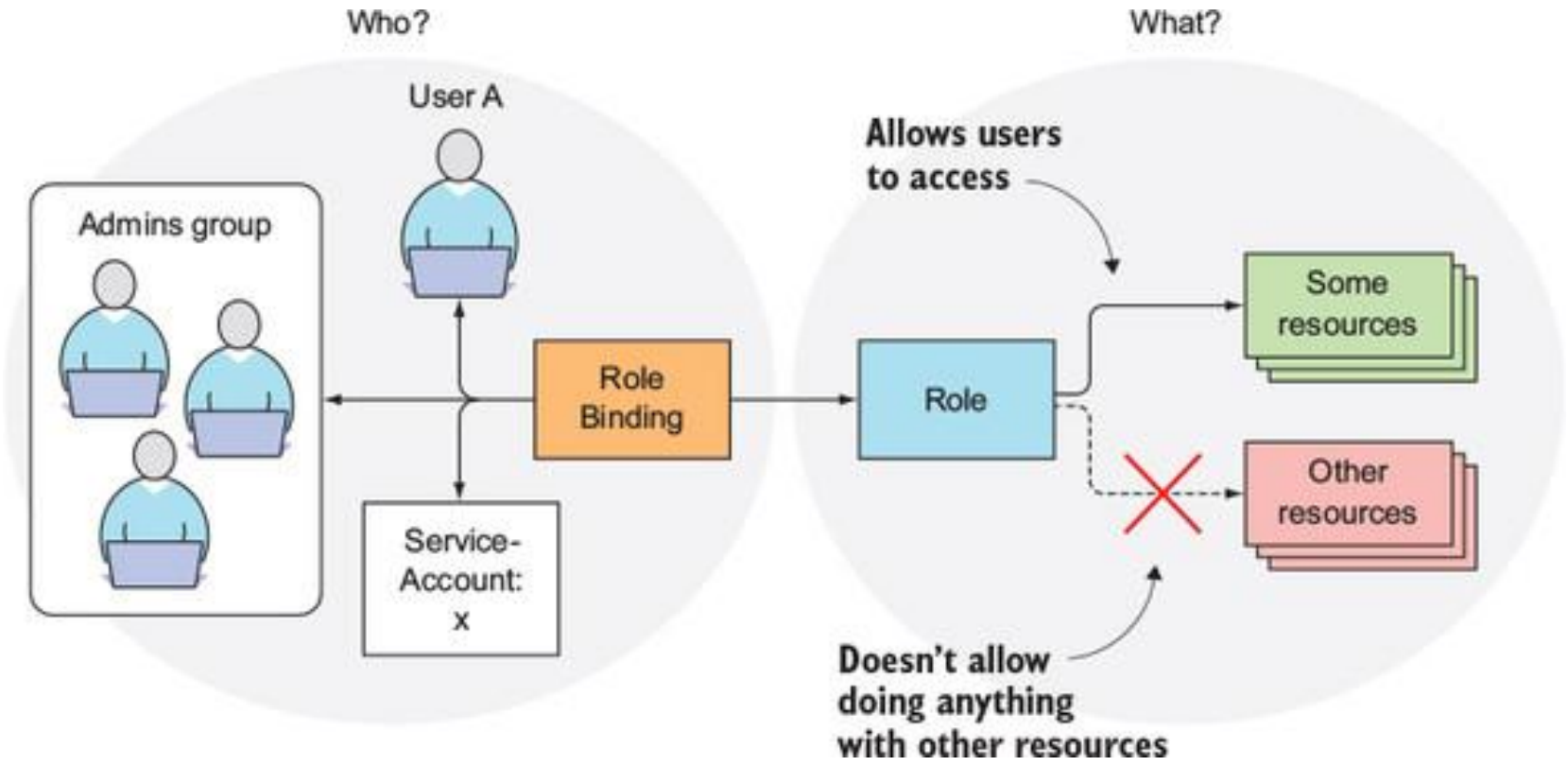
```
apiVersion: v1
kind: Config
preferences: {}

...

contexts:
- context:
    cluster: lab
    user: admin
  name: owner
- context:
    cluster: prod
    user: john
  name: dev
```



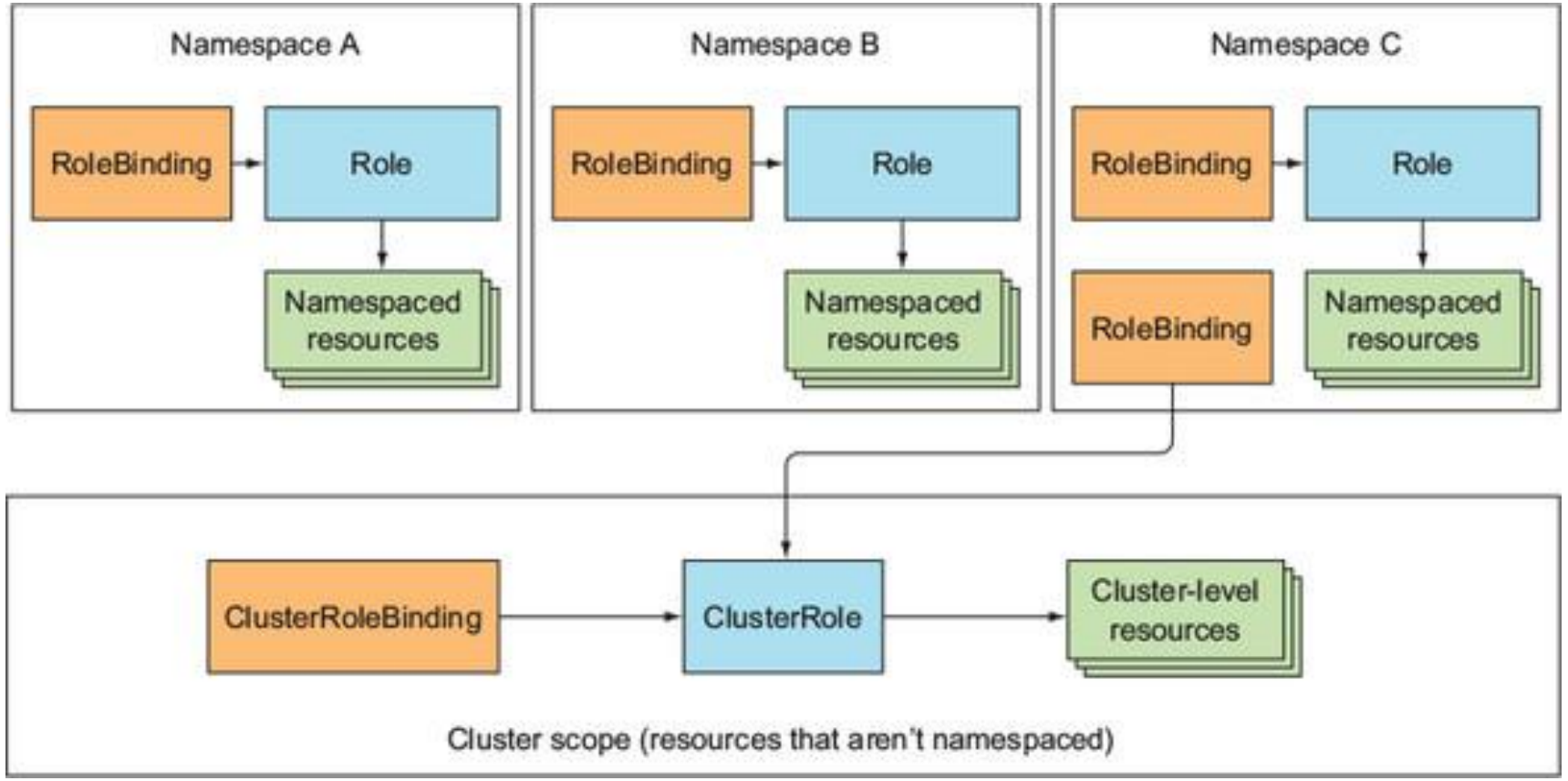
Sources:
https://learning.oreilly.com/library/view/kubernetes-in-action/9781617293726/kindle_split_024.html

Kubernetes API Actions and HTTP Methods

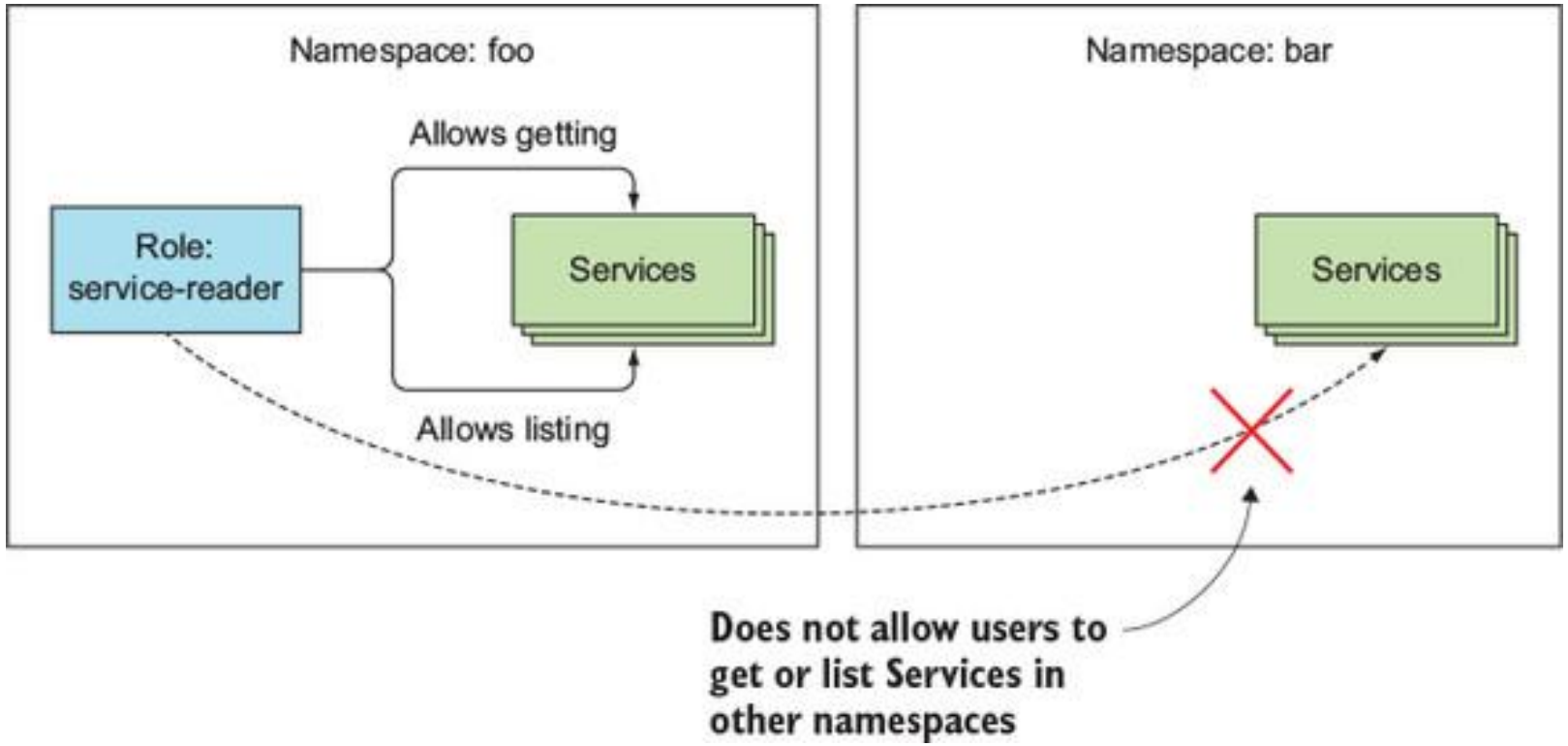
Verb (API)	HTTP Method
get	GET
list	GET
watch	GET
create	POST
update	PUT
patch	PATCH
delete	DELETE
deletecollection	DELETE



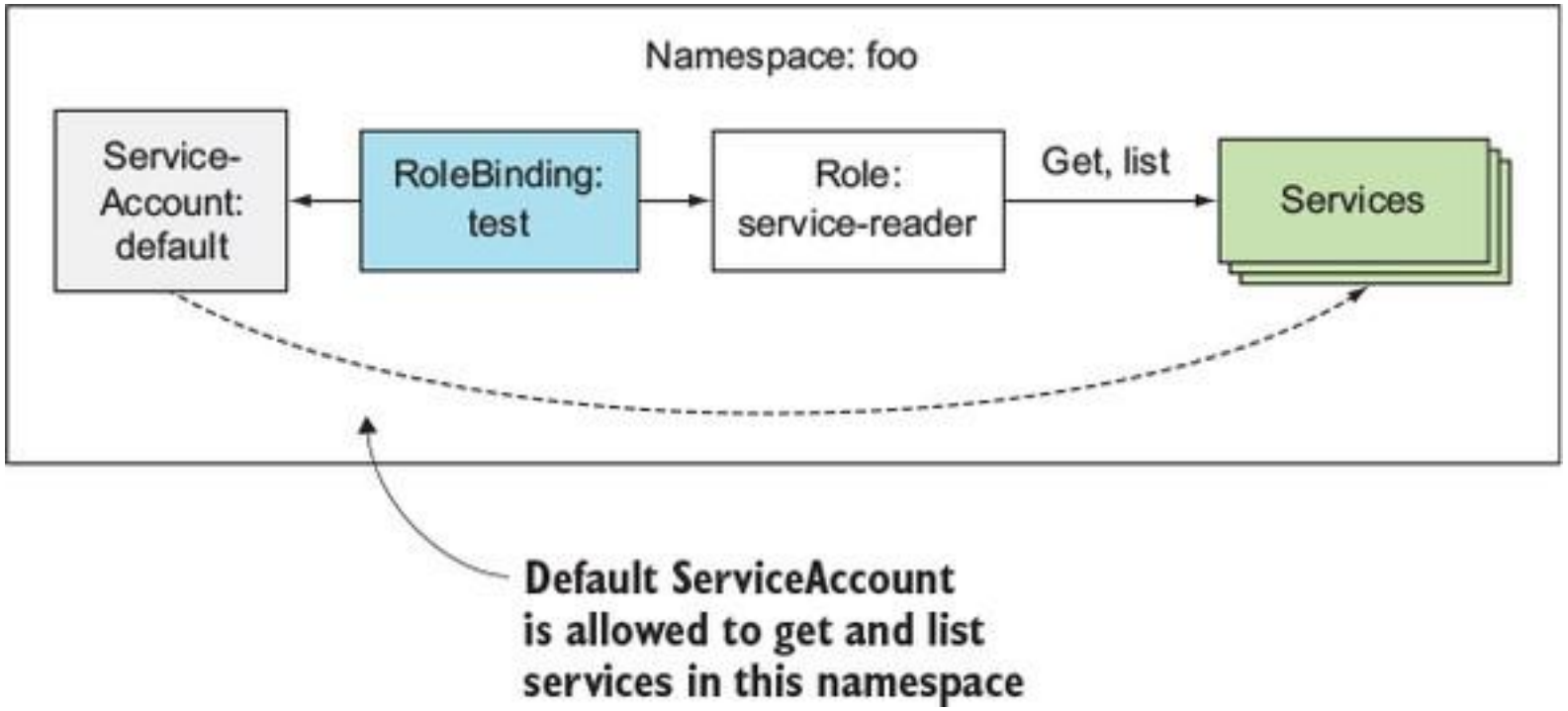
Sources:
https://learning.oreilly.com/library/view/kubernetes-in-action/9781617293726/kindle_split_024.html



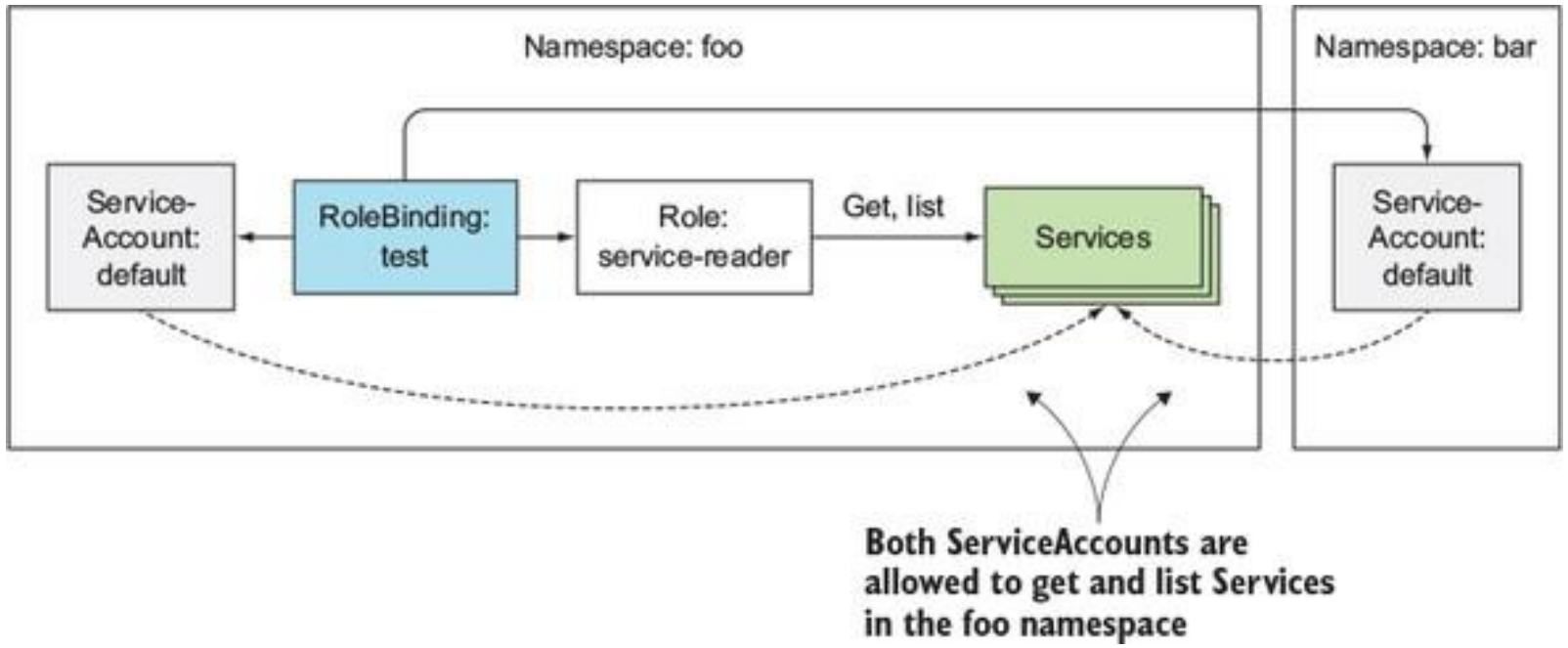
Sources:
https://learning.oreilly.com/library/view/kubernetes-in-action/9781617293726/kindle_split_024.html



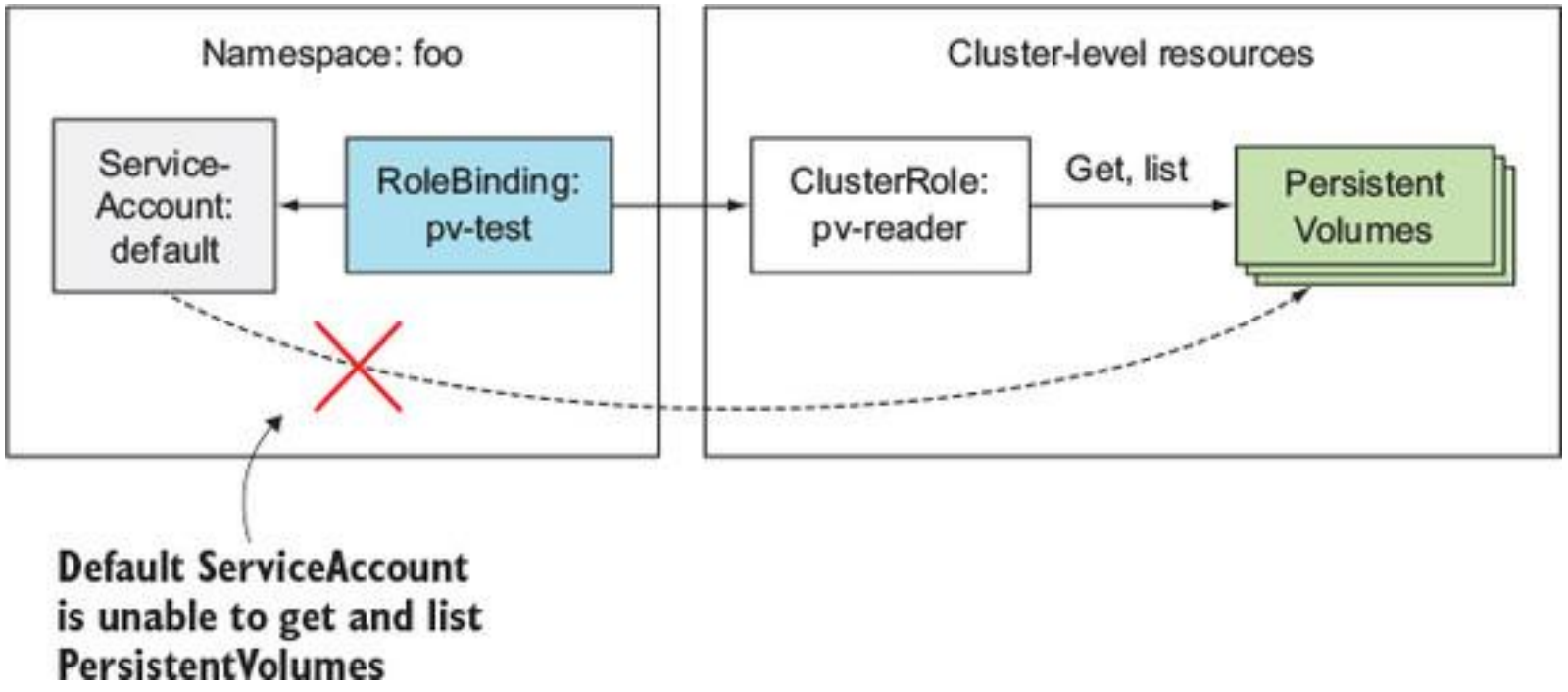
Sources:
https://learning.oreilly.com/library/view/kubernetes-in-action/9781617293726/kindle_split_024.html



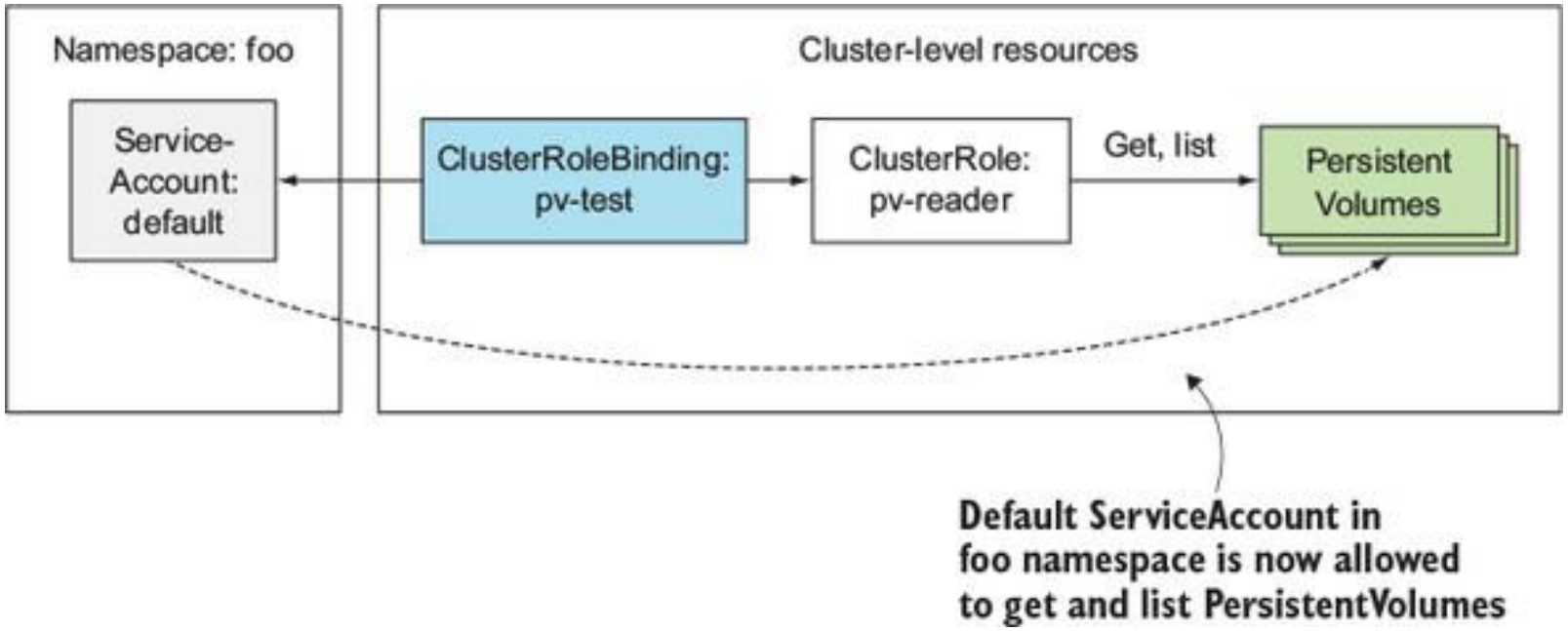
Sources:
https://learning.oreilly.com/library/view/kubernetes-in-action/9781617293726/kindle_split_024.html



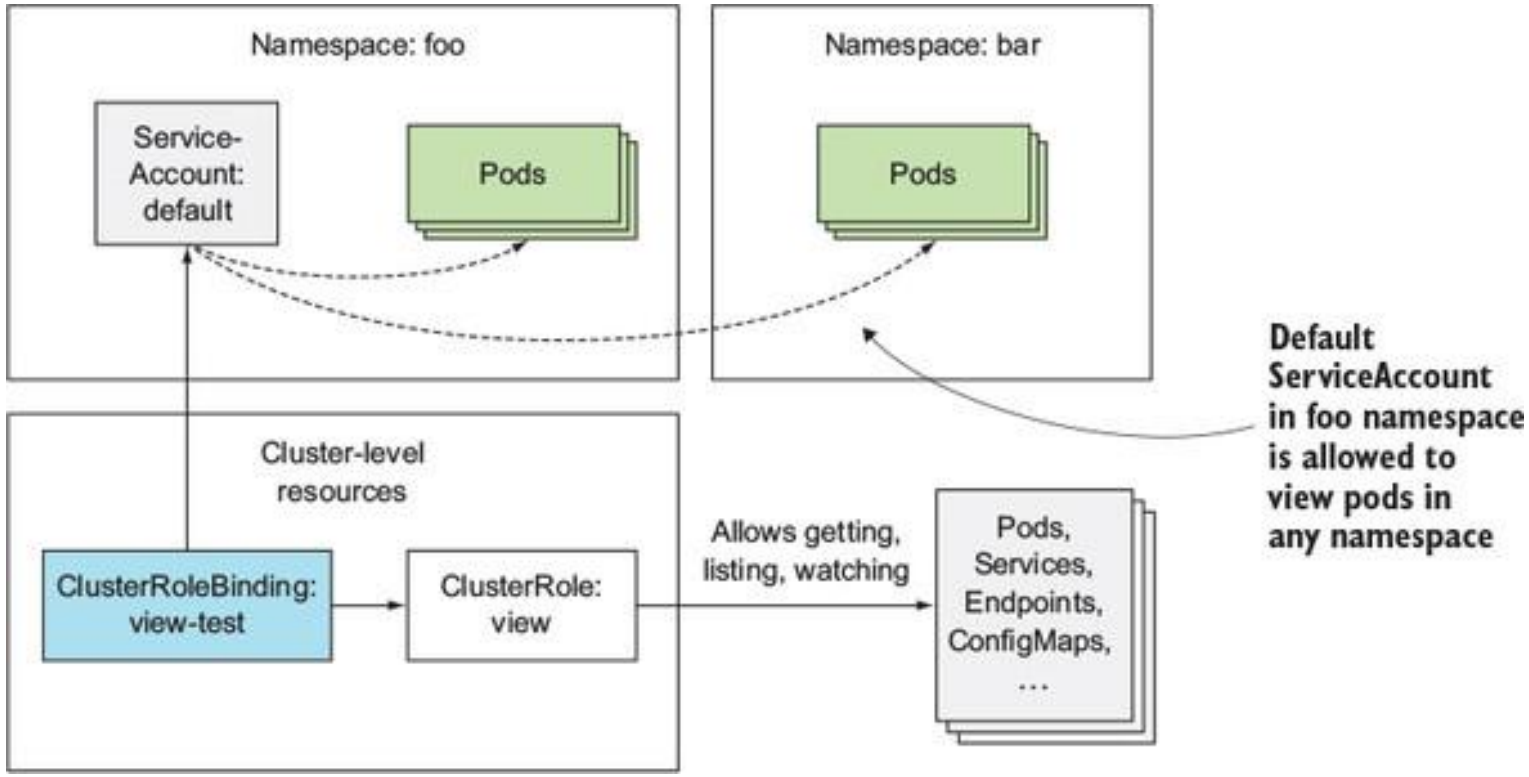
Sources:
https://learning.oreilly.com/library/view/kubernetes-in-action/9781617293726/kindle_split_024.html



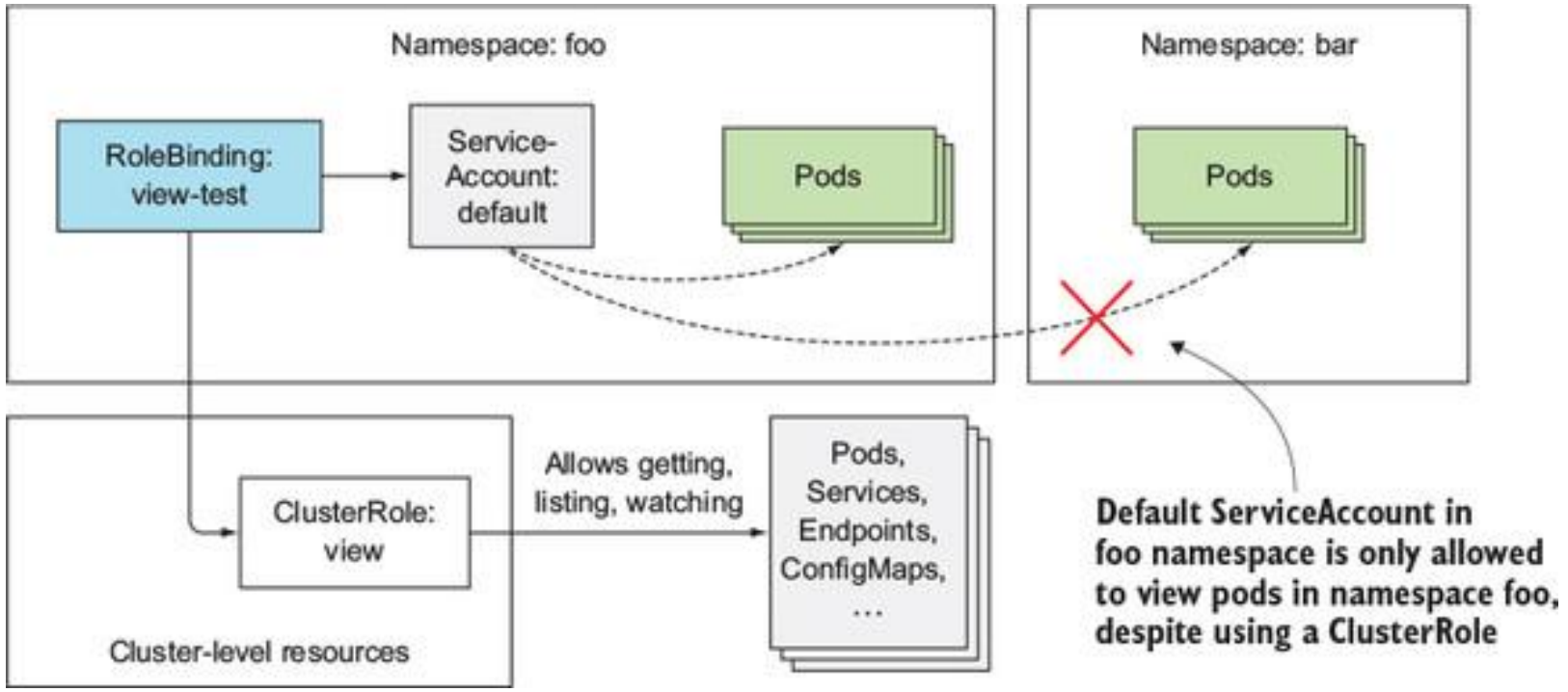
Sources:
https://learning.oreilly.com/library/view/kubernetes-in-action/9781617293726/kindle_split_024.html



Sources:
https://learning.oreilly.com/library/view/kubernetes-in-action/9781617293726/kindle_split_024.html



Sources:
https://learning.oreilly.com/library/view/kubernetes-in-action/9781617293726/kindle_split_024.html



Sources:
https://learning.oreilly.com/library/view/kubernetes-in-action/9781617293726/kindle_split_024.html

Key Takeaways

- Each HTTP request goes through various filters and if they pass additional context is added or if they fail a proper status is returned.
- Your kubeconfig can be used to access multiple clusters at once using what is known as a context and certain access based on the context is given.
- Pods run under the default ServiceAccount, which is created for each namespace automatically, unless changed.
- Roles and ClusterRoles define what actions can be performed on which resources.
- RoleBindings and ClusterRoleBindings bind Roles and ClusterRoles to users, groups, and ServiceAccounts.