Authorization

Node

Part of system node groups

name prefixed with system node

kubeAPI

Kubelet

Read & Write requests

User

Handled by specialized authorizer

Node Authorizer

ABAC

→ Associating group of users with set of permissions

Ex:

dev-user → can view PODs
create PODs
delete PODs

RBAC

→ Creating role with set of permissions

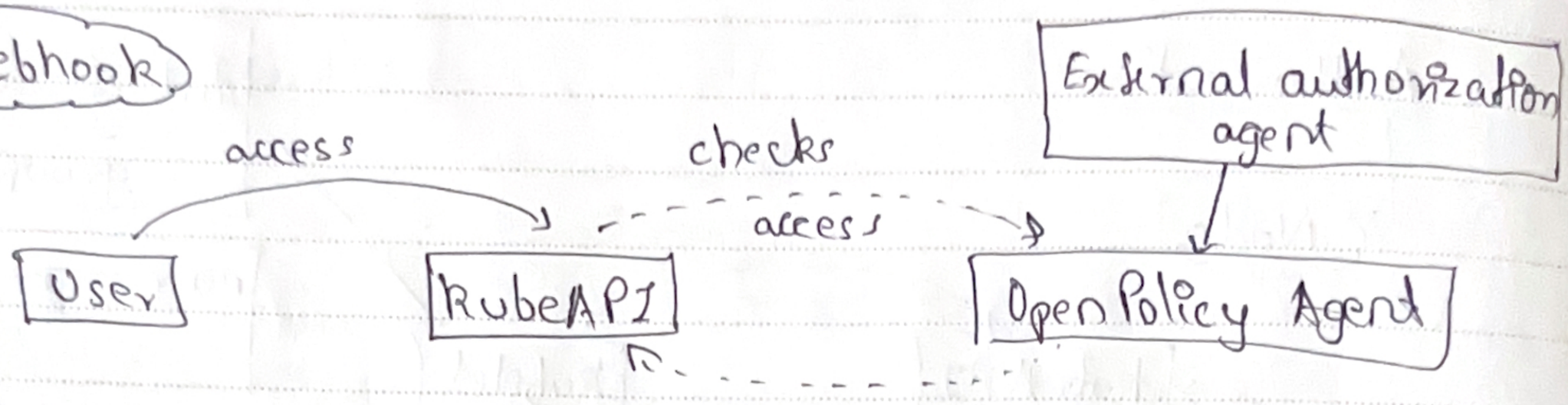
Ex: dev-user1

dev-user2

→ can view POD
create POD
delete POD

Developer

Webhook



Other modes
- always allow → By default always allow is set
- always deny

RBAC

Role Object

kubectl create role <name>
--resource: --verb -n

role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role                    // ClusterRole for cluster based roles
metadata:
    name: developer
rules:
- apiGroups: [" "]
  resources: ["pods"]         // "nodes" for clusterRole
  verbs: ["list", "get", "create", "update", "delete"]
```

To assign role to the user → RoleBinding

resourceName: ["blue", "black"] → access to particular pods instead of all pods

kubectl create rolebinding <name> --role=<?> --user=<?>
-n <namespace>

role-binding.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding          // ClusterRoleBinding for clusterRoles
metadata:
  name: dev-user-developer-binding
subjects:
- kind: User
  name: dev-user
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role               // ClusterRole
  name: developer
  apiGroup: rbac.authorization.k8s.io
```

User-Details

Details of Role creater

Check Access → for user

Ex:

kubectl auth can-i create deployments
kubectl auth can-i delete nodes

kubectl auth can-i create deployments --as dev-use
" " " " pods " "

→ For admins testing of roles

Admission-controllers

- Better security measures
- Perform additional operations before pod gets created

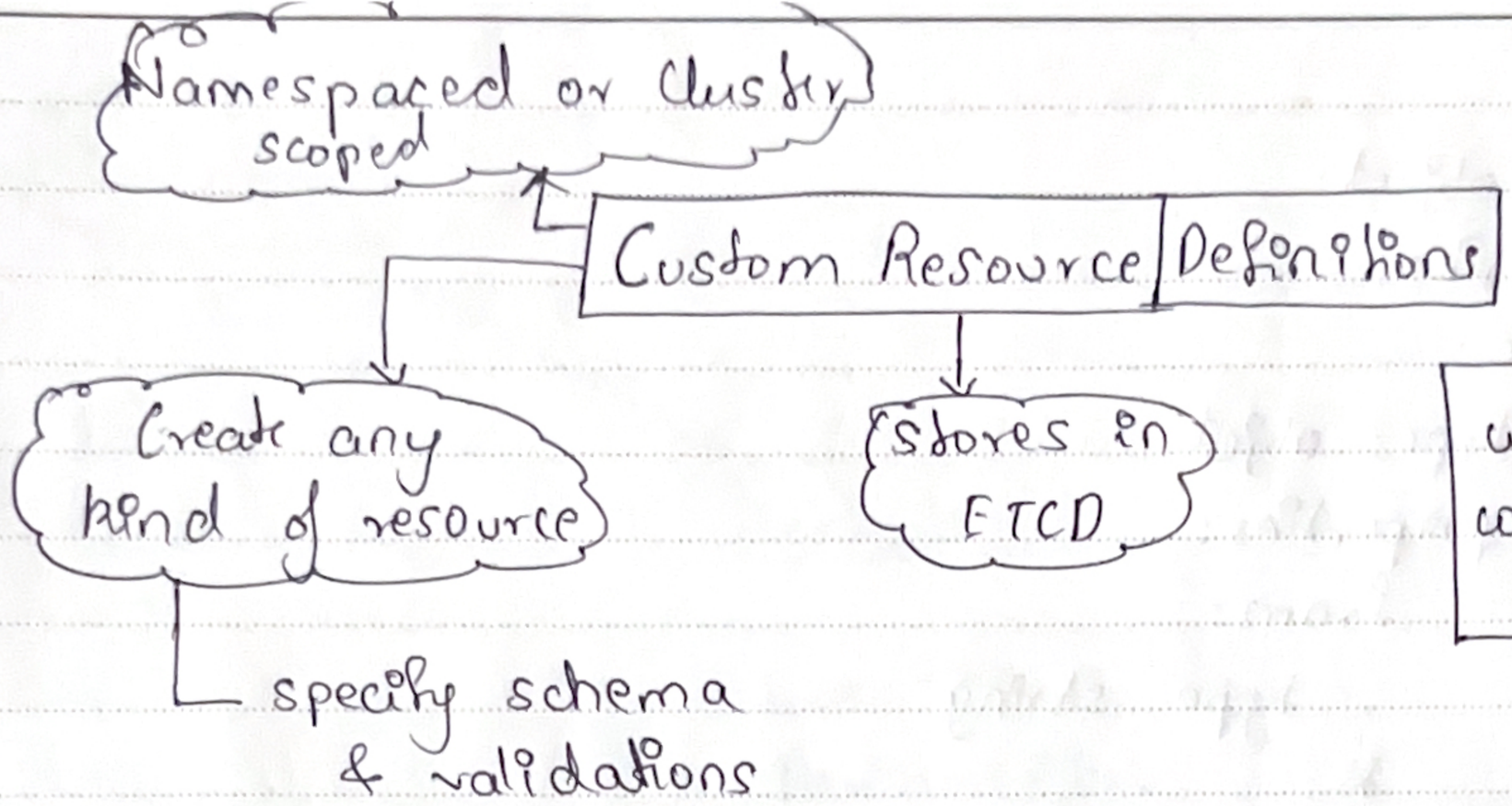
- AlwaysPullImages
- DefaultStorageClass
- EventRateLimit
- NamespaceExists
- many more...

(default) comes with Kubernetes

NamespaceExists → Deprecated (Namespace Lifecycle created)

* To view enabled admission controllers

```
kube-apiserver -h | grep enable-admission-plugins
```



we can create custom controllers to handle custom resources

```
apiVersion : apiextensions.k8s.io/v1
kind : CustomResourceDefinition
metadata:
  name: flightticketts.flights.com
spec:
  scope : Namespaced          ——→ namespaced or scope or not
  group : flights.com         ——— API group
  names:
    kind : FlightTicket
    singular : flightticket
    plural : flightticketts   ——— used by api-resources
    shortNames:
      - ft
  versions:
    - name: v1
      served: true
      storage: true
      schema:                 ——→ define What fields are supported
        openAPIV3Schema:      ——
```

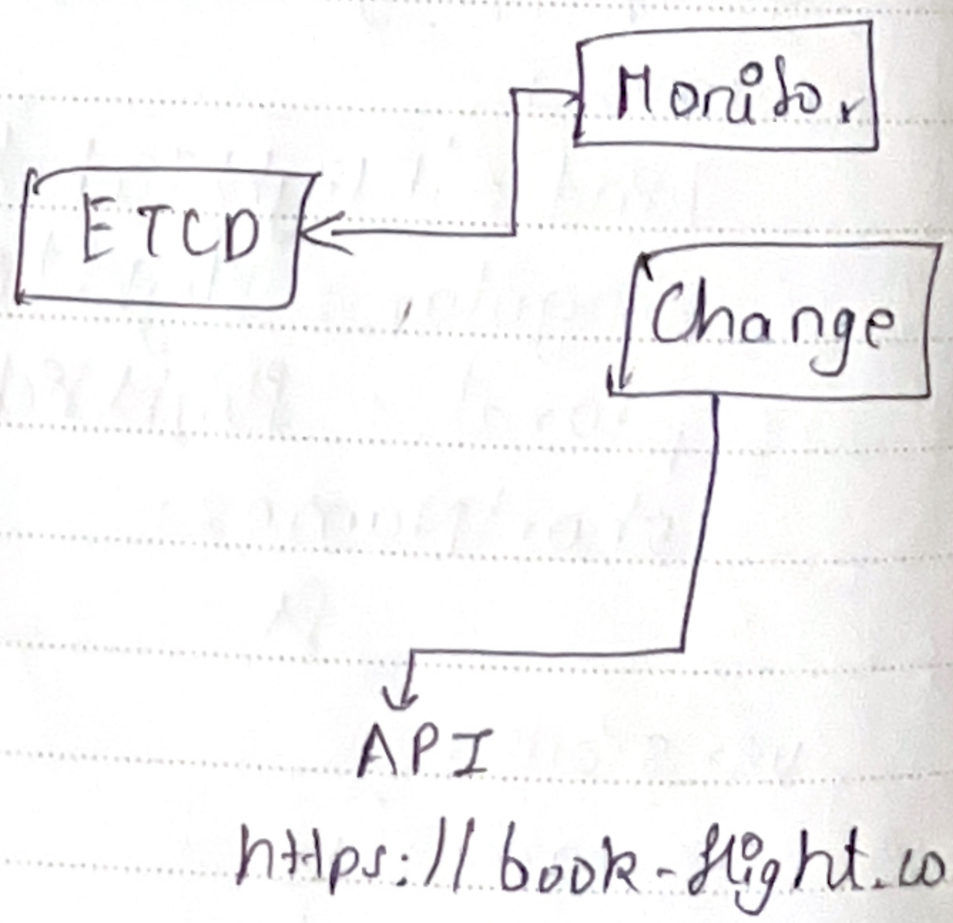
```
type: object
properties:
  spec:
    type: object
    properties:
      from:
        type: string
      to:
        type: string
      number:
        type: integer
        minimum: 1
        maximum: 10
```

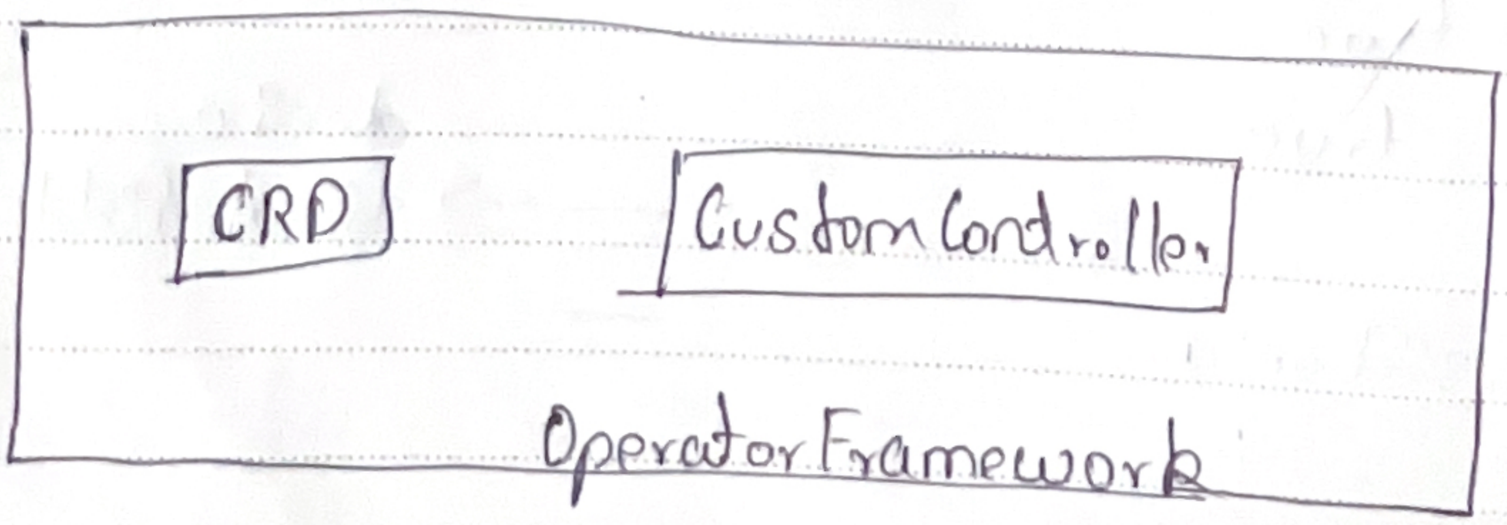
} validations

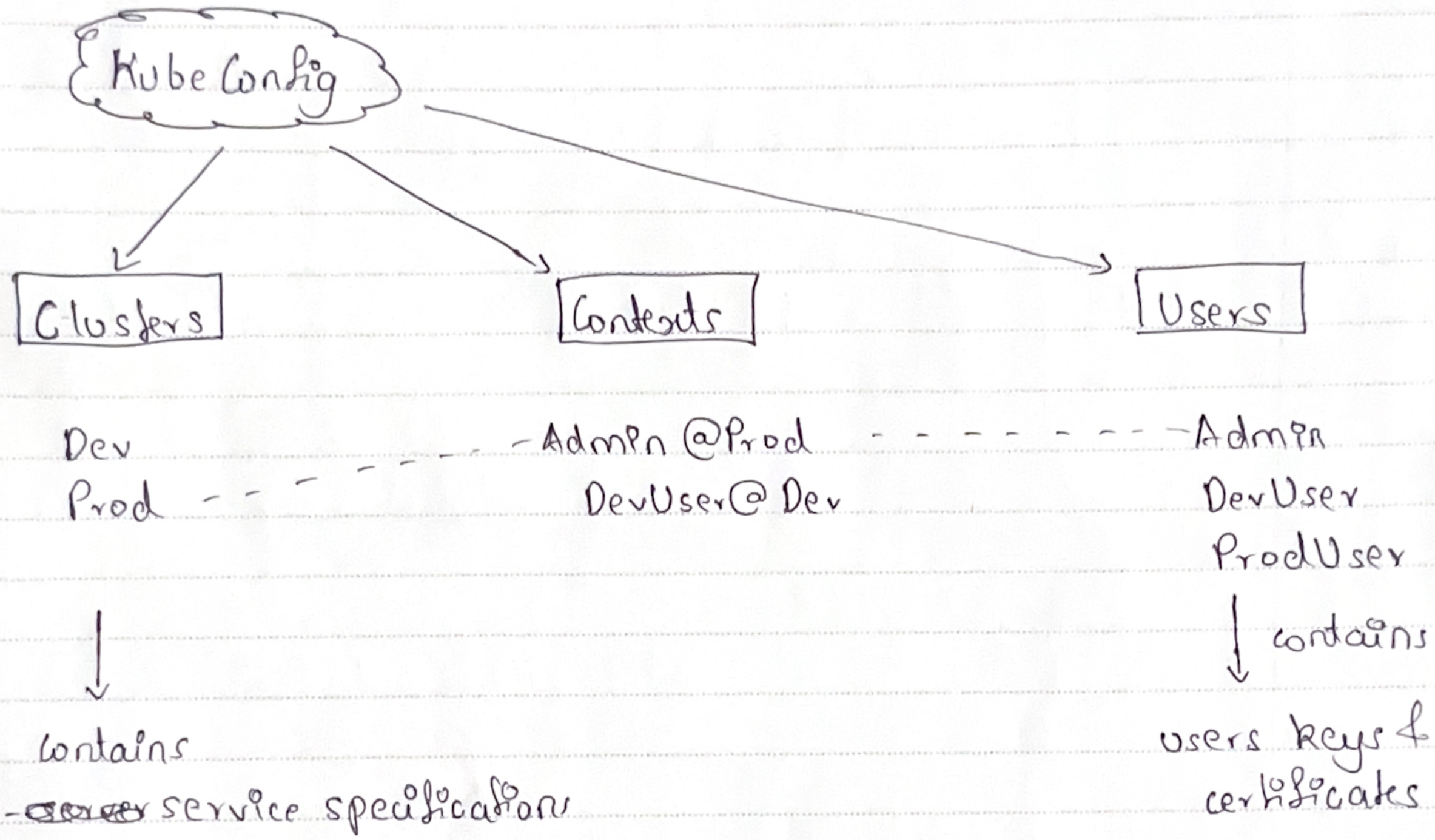
Custom Resource object

```
apiVersion: flights.com/v1
kind: FlightTicket
metadata:
  name: my-flight-ticket
spec:
  from: Bengluru
  to: London
  number: 2
```

Custom Controller







File

```
apiversion : v1
kind : Config  -> current-context : <default>
                                    specify
clusters :
  - name : my-kube
    cluster :
      certificate-authority : ca.crt
      server : https://my-kube : 6443
contexts :
  - name : my-kube-admin @ my-kube
    context :
      cluster : my-kube
      user : my-kube-admin  -> namespace : < >
users :
  - name : my-kube-admin
    user :
      client-certificate : admin.crt
      client-key : admin.key
```

Note:
Don't have to create any objects

for command line
->kubectl config -h

-> We can specify data directly.
certificate-authority-data