KODEKLOUD

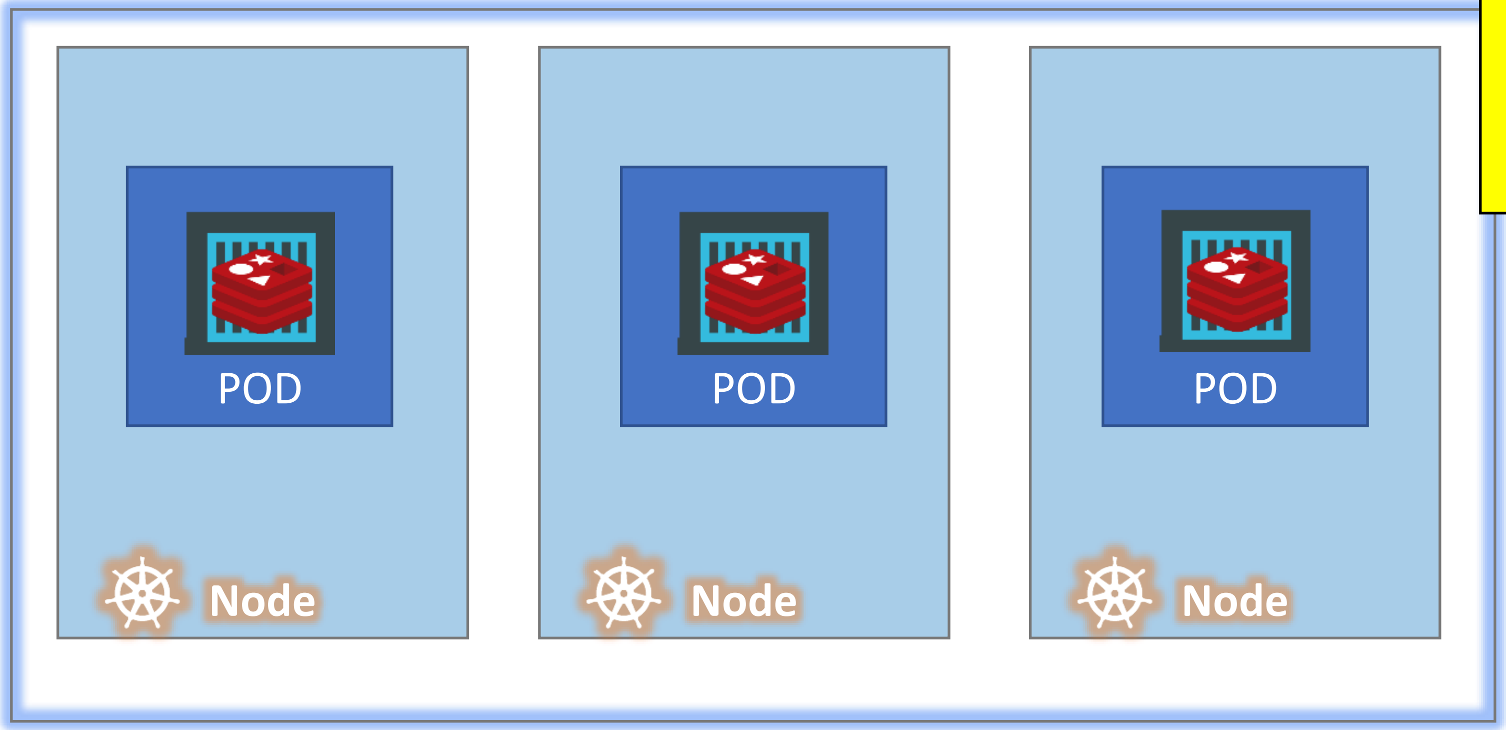
POD

Assumptions

Docker Image

Kubernetes Cluster

POD

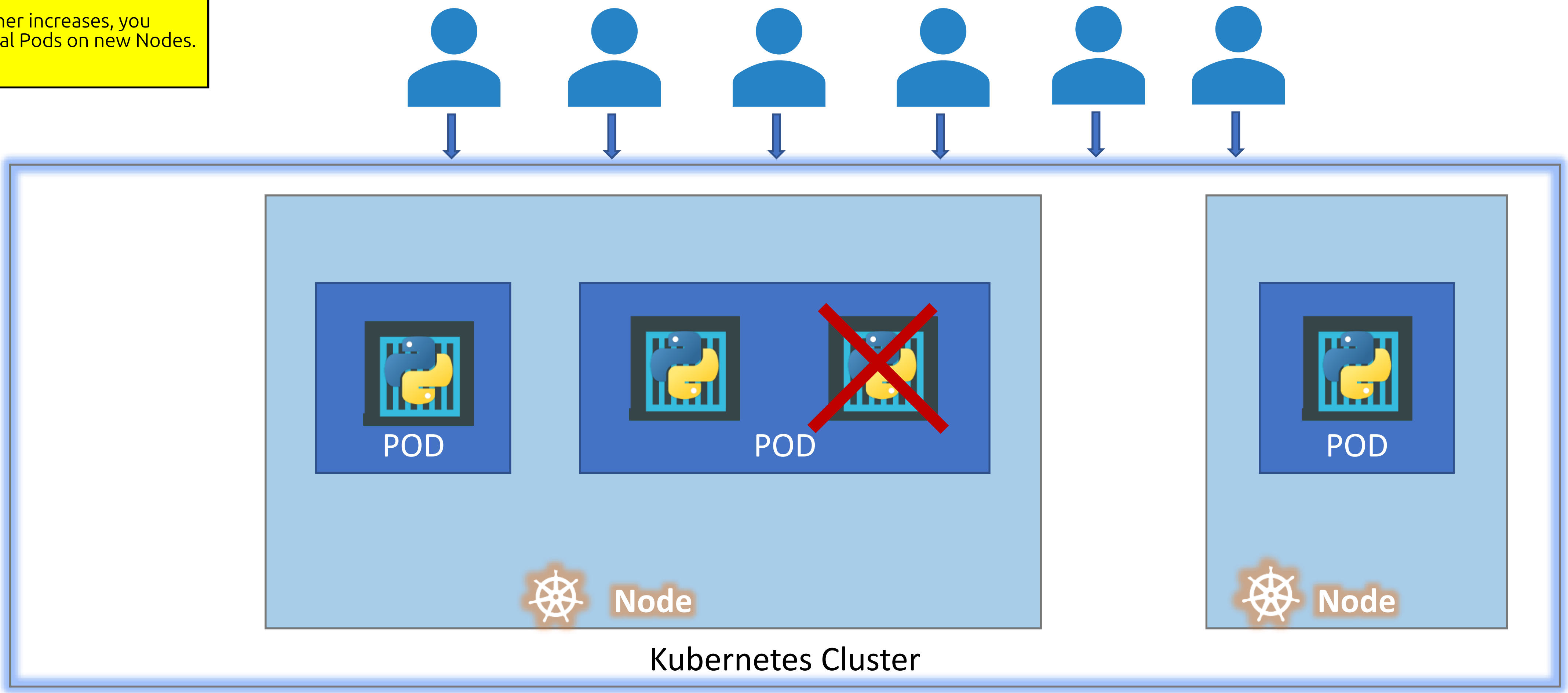


A Pod is the smallest object you can create in a kubernetes cluster.
A Pod is a single instance of your application.

POD

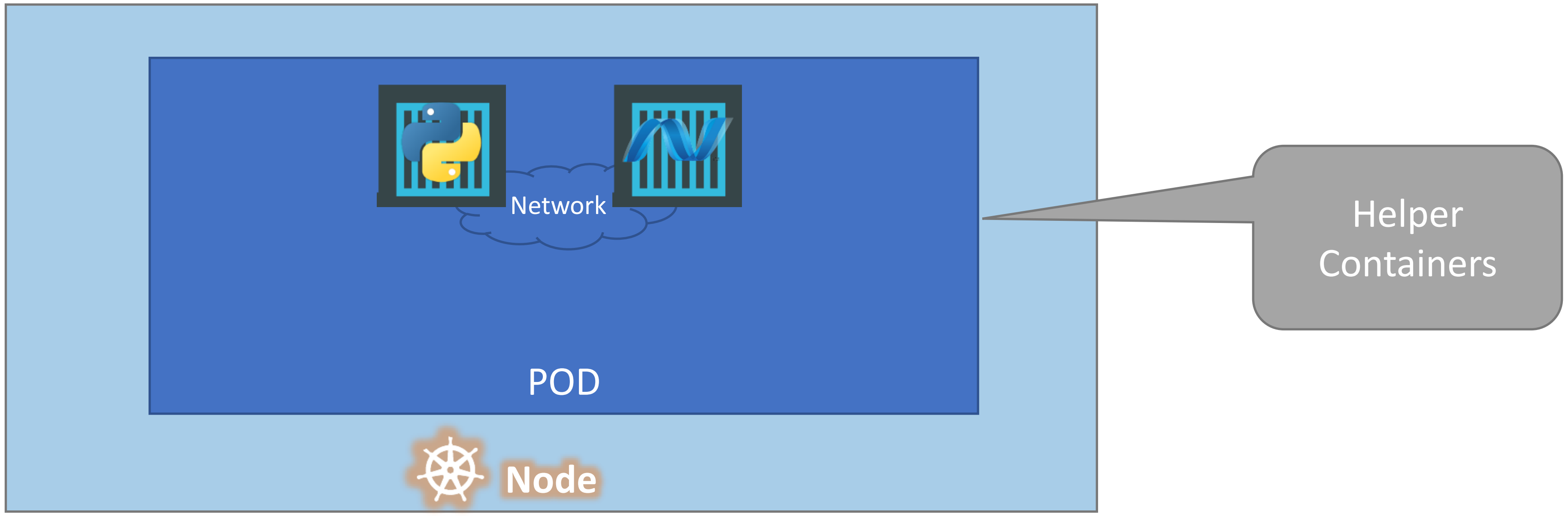
When scaling for more load, you create more Pods, not more instances of your app in the same Pod!

When load further increases, you deploy additional Pods on new Nodes.



Multi-Container PODs

The two containers can communicate using localhost, since they live on the same Pod.

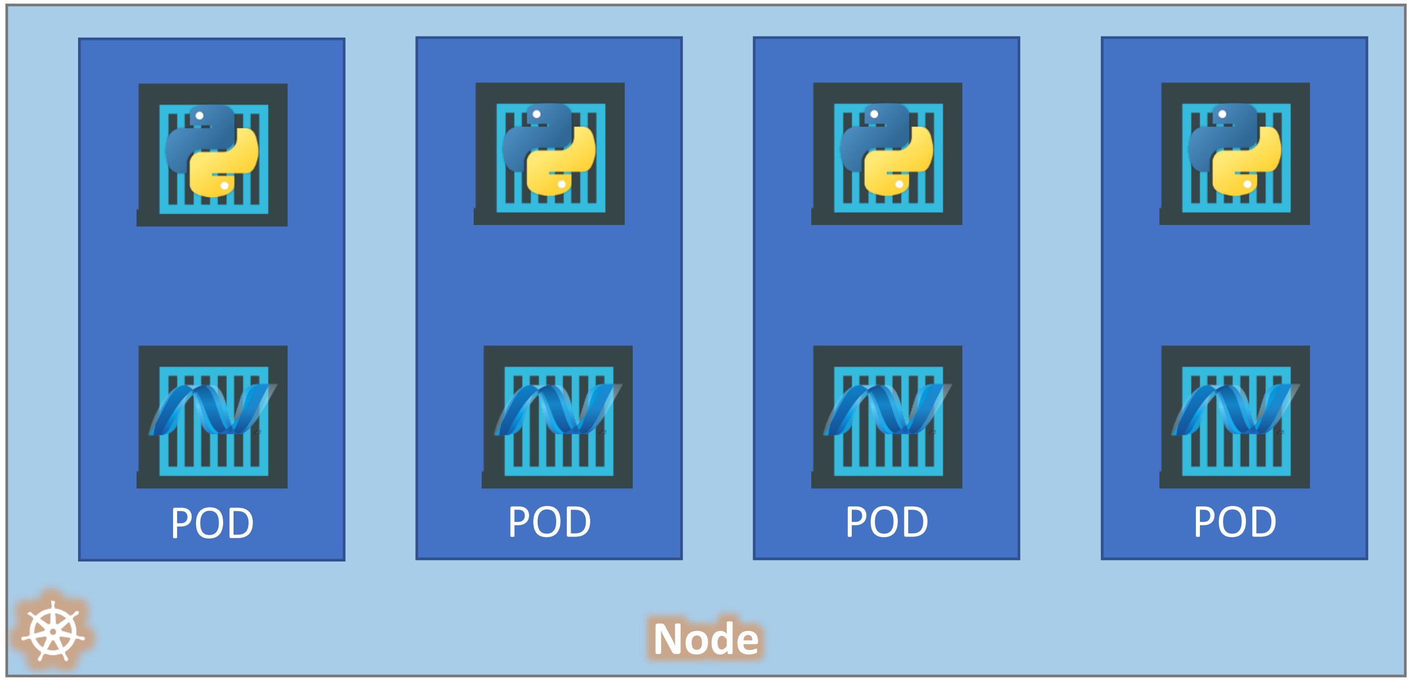


PODs Again!

manually scaling by running more containers and helper containers and linking them together, it gets difficult to manage. Kubernetes solves this issue.

```
docker run python-app
docker run python-app
docker run python-app
docker run python-app
docker run helper –link app1
docker run helper –link app2
docker run helper –link app3
docker run helper –link app4
```

App	Helper	Volume
Python1	App1	Vol1
Python2	App2	Vol2



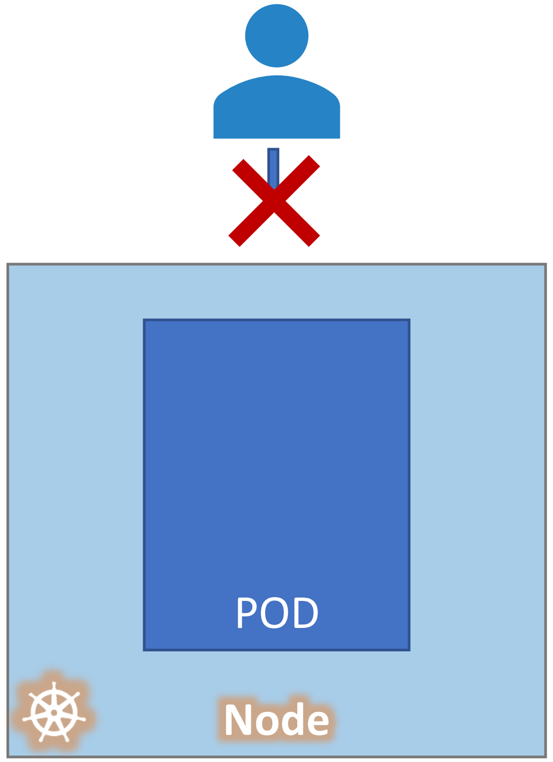
Note: I am avoiding networking and load balancing details to keep explanation simple.

kubectl

- `kubectl run nginx --image nginx`

```
kubectl get pods
NAME     READY    STATUS              RESTARTS    AGE
nginx    0/1      ContainerCreating   0           6s
```

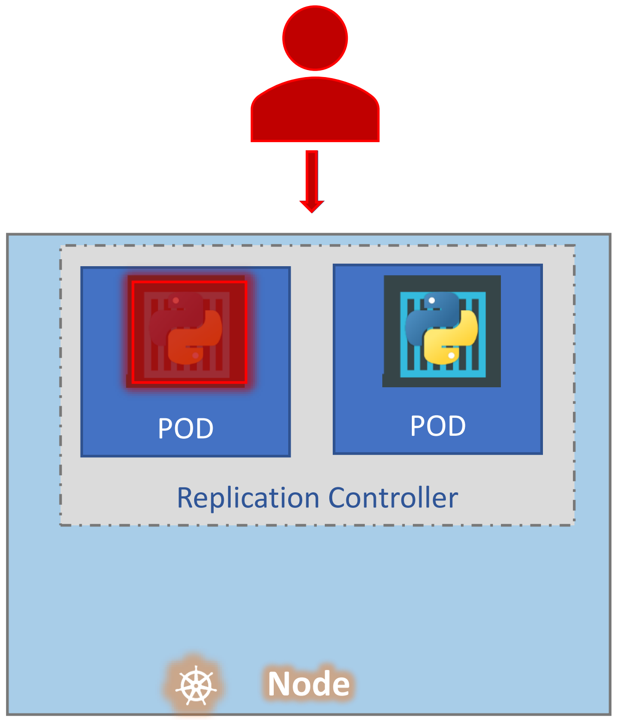
```
NAME     READY    STATUS     RESTARTS    AGE
nginx    1/1      Running    0           34s
```

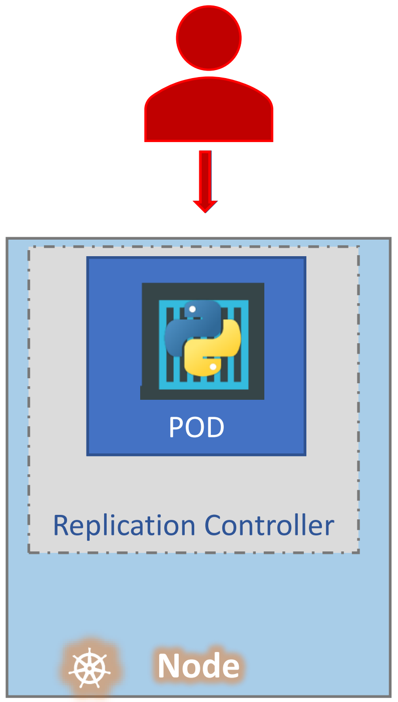


KODEKLOUD

Replication Controller

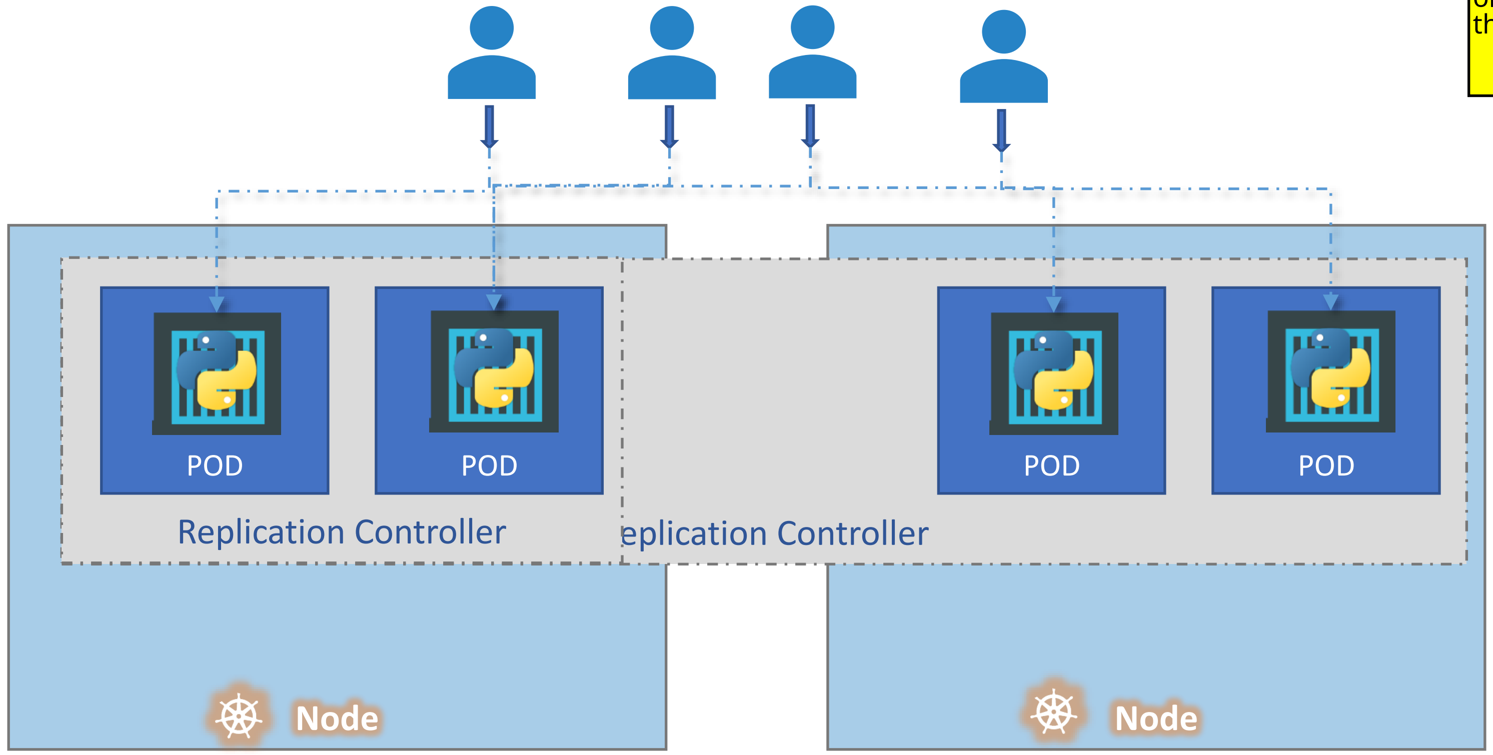
High Availability





Load Balancing & Scaling

Replication Controller helps scaling across multiple nodes. Replication Controller are the older version, ReplicaSets are the new version.



- Replication Controller

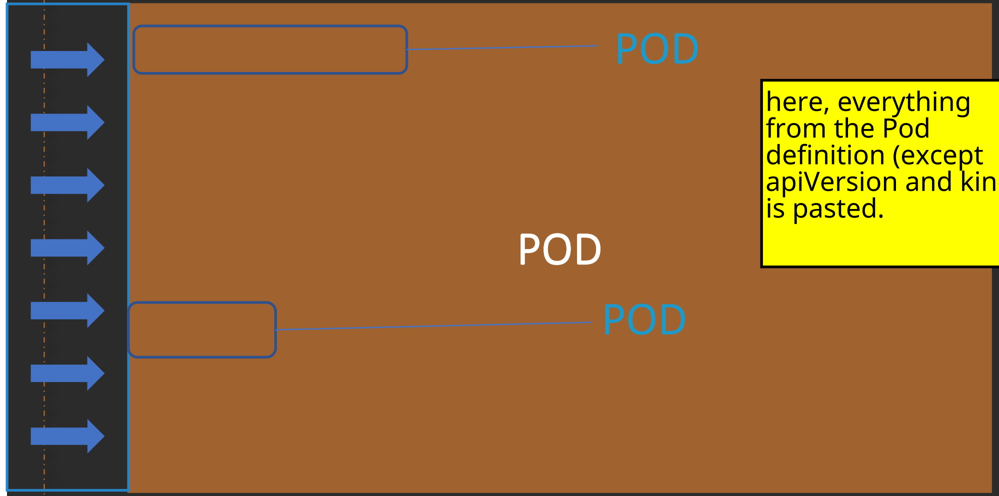
Replica Set

rc-definition.yml

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: myapp-rc
  labels:
      app: myapp
      type: front-end
spec:
  template:

  replicas: 3
```

Replication Controller

Replication Controller



here, everything from the Pod definition (except apiVersion and kind) is pasted.

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

metadata:
  name: myapp-pod
  labels:
     app: myapp
     type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

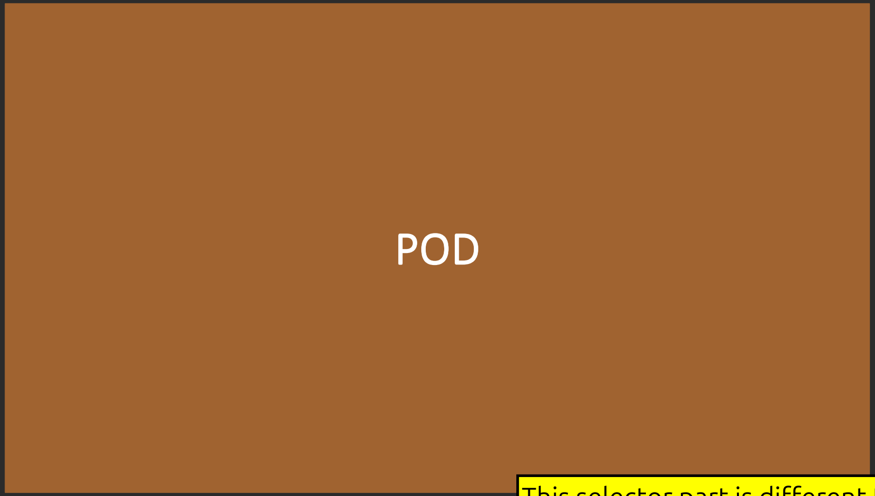
```
• > kubectl create -f rc-definition.yml
replicationcontroller “myapp-rc” created
```

```
> kubectl get replicationcontroller
NAME        DESIRED     CURRENT     READY      AGE
myapp-rc    3           3           3          19s
```

```
> kubectl get pods
NAME              READY      STATUS      RESTARTS    AGE
myapp-rc-4lvk9    1/1        Running     0           20s
myapp-rc-mc2mf    1/1        Running     0           20s
myapp-rc-px9pz    1/1        Running     0           20s
```

replicaset-definition.yml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-repl
  labels:
      app: myapp
      type: front-end
spec:
 template:
```



```yaml
 replicas: 3
 selector:
    matchLabels:
       type: front-end
```

this happens if you forget to use 'v1' instead of 'apps/v1' for ReplicaSet

```
error: unable to recognize "replicaset-
definition.yml": no matches for /, Kind=ReplicaSet
```

This selector part is different to ReplicationController.
Why use selector if we already have specified the Pod definition?
Because ReplicaSets can also manage Pods that were not created as part of the ReplicaSet creation.

pod-definition.yml

KODEKLOUD

```yaml
apiVersion: v1
kind: Pod

  labels:
      app: myapp
      type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

```
• > kubectl create -f replicaset-definition.yml
replicaset "myapp-replicaset" created
```

```
> kubectl get replicaset
NAME              DESIRED    CURRENT     READY       AGE
myapp-replicaset    3          3           3          19s
```

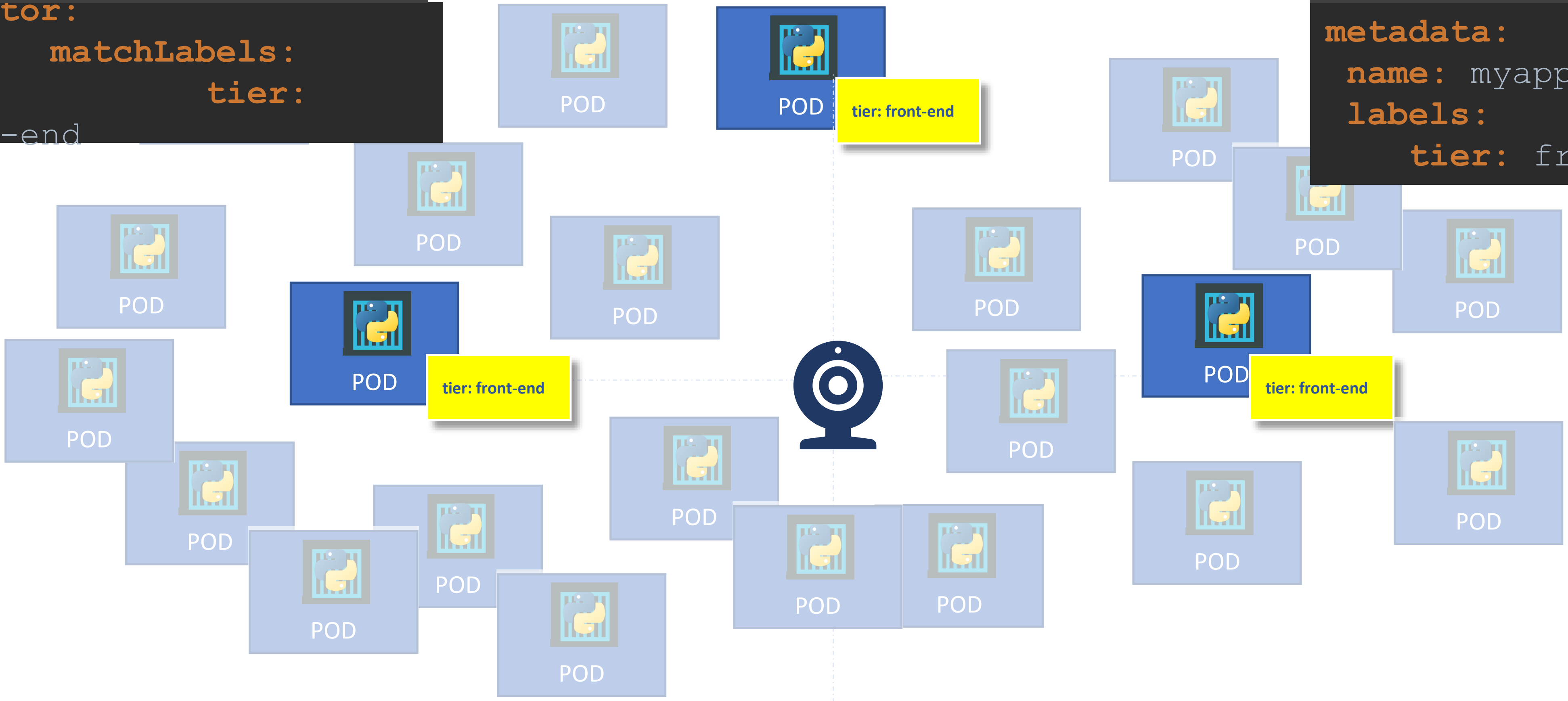
```
> kubectl get pods
NAME                      READY       STATUS     RESTARTS     AGE
myapp-replicaset-9ddl9     1/1        Running    0            45s
myapp-replicaset-9jtpx     1/1        Running    0            45s
myapp-replicaset-hq84m     1/1        Running    0            45s
```

Labels and Selectors

The ReplicaSet is a process that monitors the Pods and deploys new ones if the specified Replicas-number is not met.

replicaset-definition.yml
```
selector:
        matchLabels:
                tier:
front-end
```

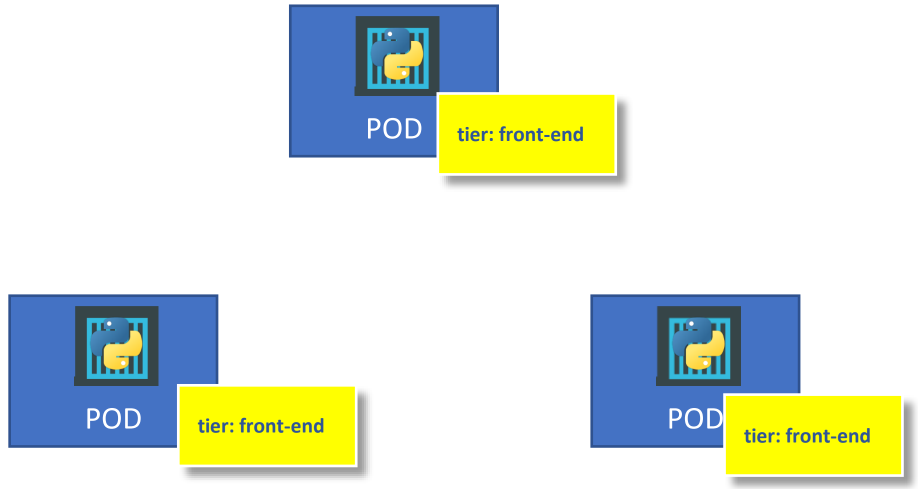
pod-definition.yml
```
metadata:
  name: myapp-pod
  labels:
      tier: front-end
```



replicaset-definition.yml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-replicaset
  labels:
       app: myapp
       type: front-end
spec:
  template:
     metadata:
       name: myapp-pod
       labels:
           app: myapp
           type: front-end
     spec:
       containers:
       - name: nginx-container
         image: nginx
  replicas: 3
  selector:
     matchLabels:
         type: front-end
```

Template



Scale

```
> kubectl replace -f replicaset-definition.yml
```

```
> kubectl scale --replicas=6 -f replicaset-definition.yml
```

```
> kubectl scale --replicas=6 replicaset myapp-replicaset
```

TYPE NAME

replicaset-definition.yml

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-replicaset
  labels:
      app: myapp
      type: front-end
spec:
  template:
     metadata:
       name: myapp-pod
       labels:
          app: myapp
          type: front-end
     spec:
       containers:
       - name: nginx-container
         image: nginx

  replicas: 6
  selector:
     matchLabels:
         type: front-end
```

commands

```
> kubectl create –f replicaset-definition.yml
```

```
> kubectl get replicaset
```

```
> kubectl delete replicaset myapp-replicaset
```

*Also deletes all underlying PODs

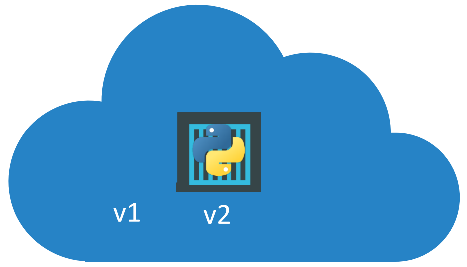
```
> kubectl replace -f replicaset-definition.yml
```

```
> kubectl scale –replicas=6 -f replicaset-definition.yml
```

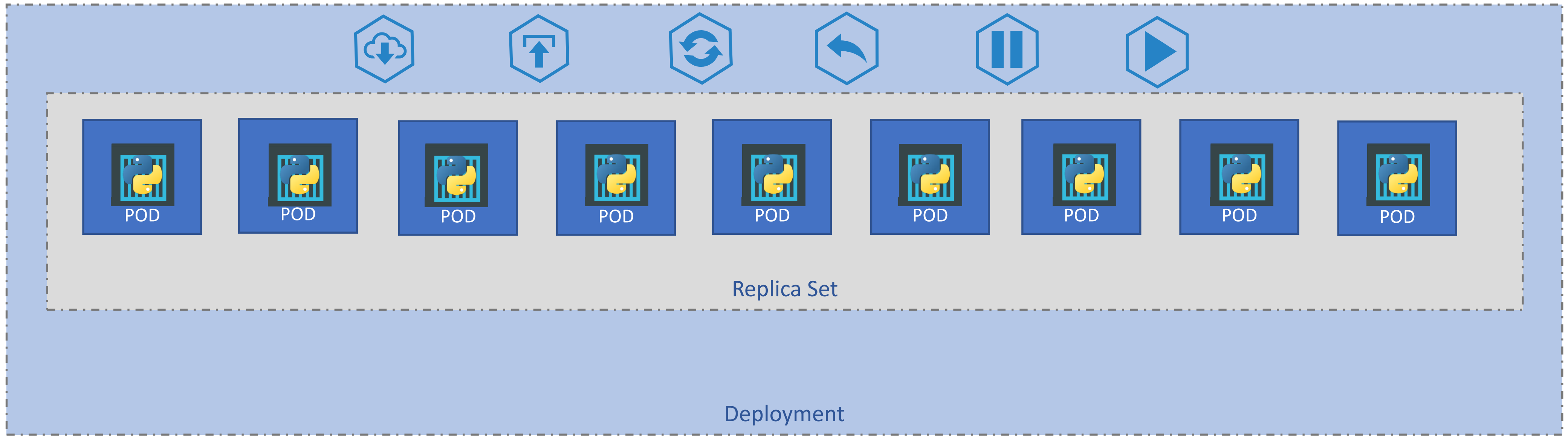
KODEKLOUD

Deployment

Deployment



Deployment takes care of updating the Pod (e.g. with newer image version) and takes care that applications is always available. Done by RollingUpdates.
Can also perform undos (deploying an older version).



Definition

Deployment creates ReplicaSets, ReplicaSets create Pods.

```
> kubectl create -f deployment-definition.yml
deployment "myapp-deployment" created
```

```
> kubectl get deployments
NAME              DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

myapp-deployment  3         3         3            3           21s
```

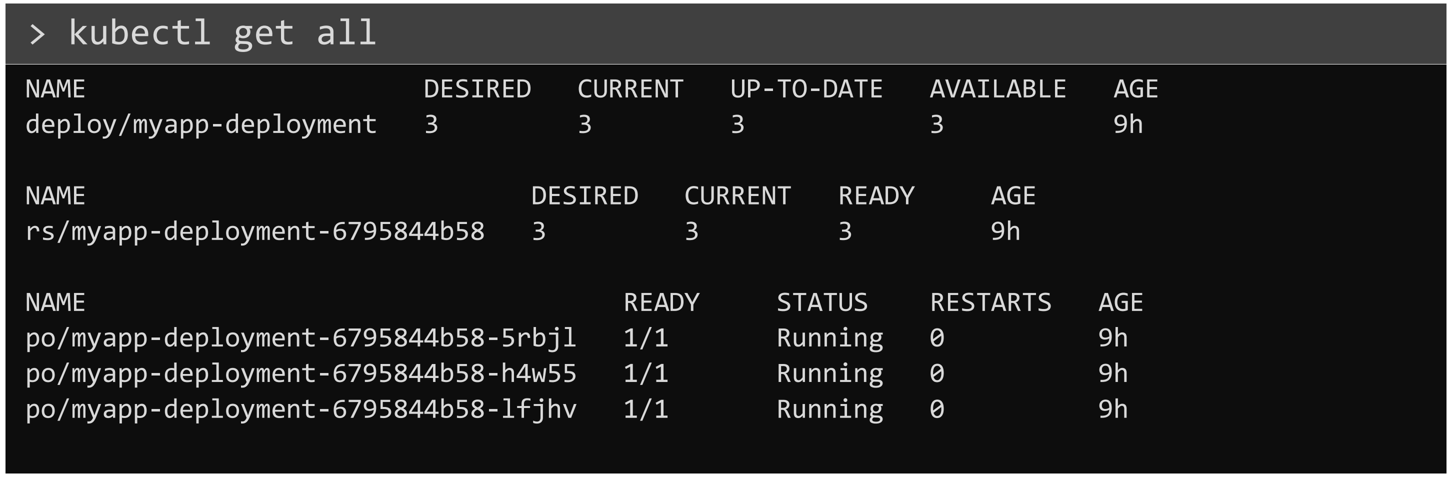
```
> kubectl get replicaset
NAME                         DESIRED   CURRENT   READY   AGE

myapp-deployment-6795844b58  3         3         3       2m
```

```
> kubectl get pods
NAME                               READY    STATUS    RESTARTS   AGE
myapp-deployment-6795844b58-5rbjl  1/1      Running   0          2m
myapp-deployment-6795844b58-h4w55  1/1      Running   0          2m
myapp-deployment-6795844b58-lfjhv  1/1      Running   0          2m
```

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
  labels:
      app: myapp
      type: front-end
spec:
  template:
     metadata:
       name: myapp-pod
       labels:
          app: myapp
          type: front-end
     spec:
       containers:
       - name: nginx-container
         image: nginx

  replicas: 3
  selector:
     matchLabels:
        type: front-end
```

commands



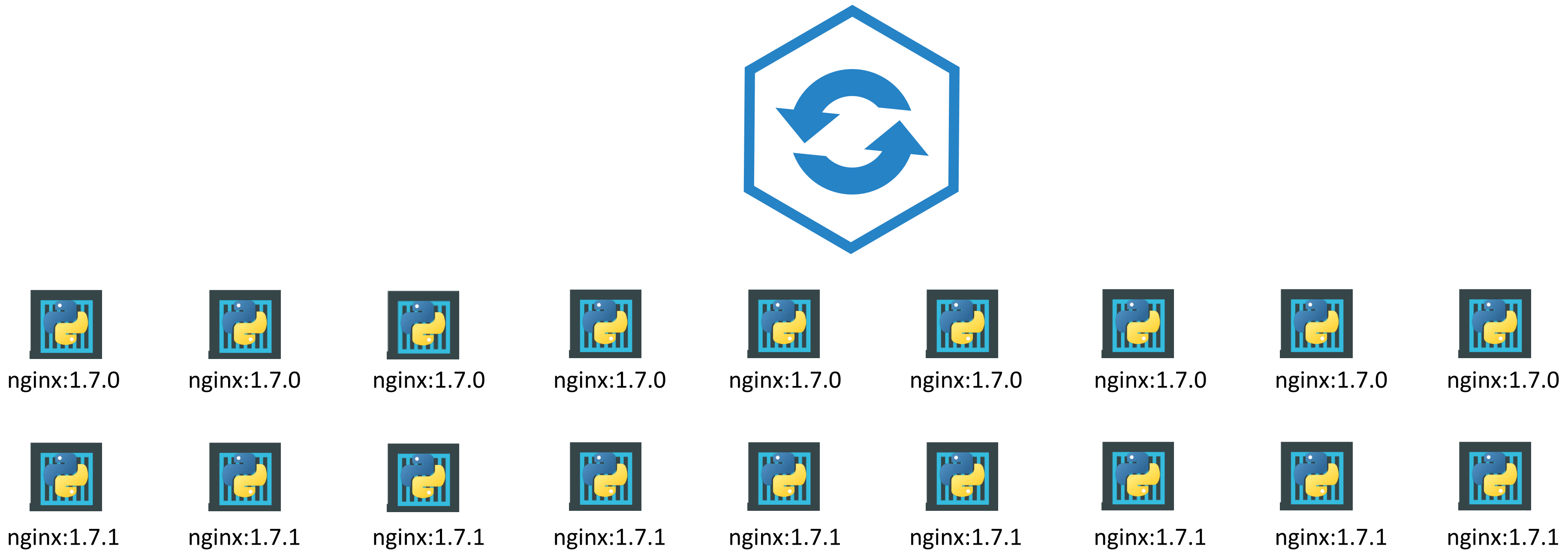
```
> kubectl get all
NAME                       DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE
deploy/myapp-deployment   3          3          3             3            9h

NAME                               DESIRED    CURRENT    READY     AGE
rs/myapp-deployment-6795844b58   3          3          3         9h

NAME                                     READY     STATUS     RESTARTS    AGE
po/myapp-deployment-6795844b58-5rbjl   1/1       Running    0           9h
po/myapp-deployment-6795844b58-h4w55   1/1       Running    0           9h
po/myapp-deployment-6795844b58-lfjhv   1/1       Running    0           9h
```

KODEKLOUD

Deployment

Updates and Rollback

Rollout and Versioning

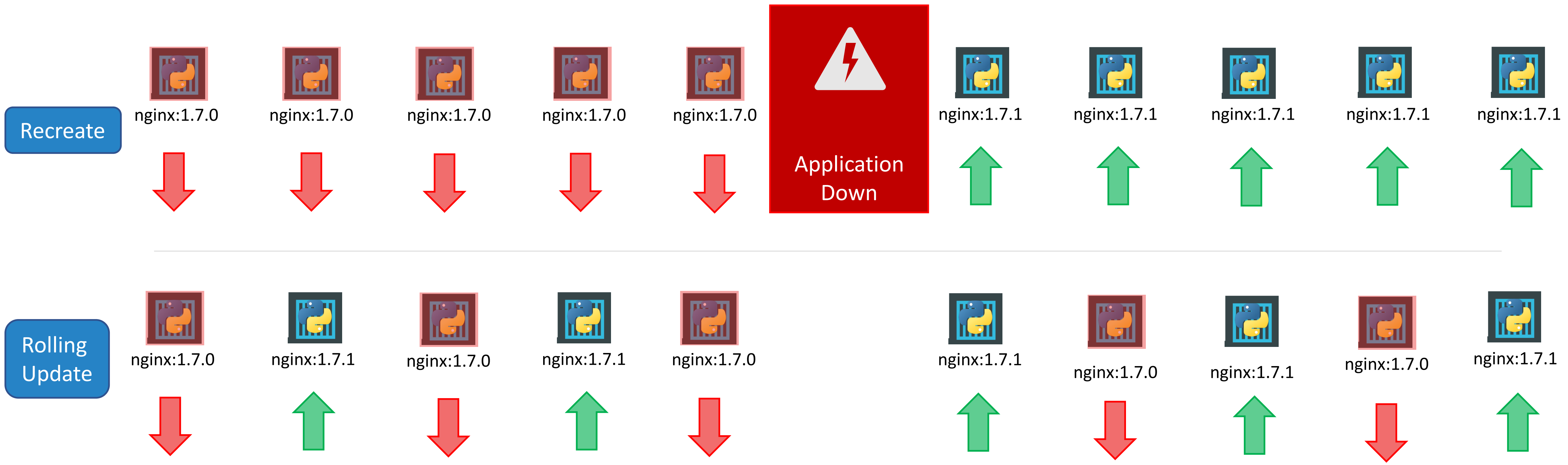


Rollout Command

```
> kubectl rollout status deployment/myapp-deployment
Waiting for rollout to finish: 0 of 10 updated replicas are available...
Waiting for rollout to finish: 1 of 10 updated replicas are available...
Waiting for rollout to finish: 2 of 10 updated replicas are available...
Waiting for rollout to finish: 3 of 10 updated replicas are available...
Waiting for rollout to finish: 4 of 10 updated replicas are available...
Waiting for rollout to finish: 5 of 10 updated replicas are available...
Waiting for rollout to finish: 6 of 10 updated replicas are available...
Waiting for rollout to finish: 7 of 10 updated replicas are available...
Waiting for rollout to finish: 8 of 10 updated replicas are available...
Waiting for rollout to finish: 9 of 10 updated replicas are available...
deployment "myapp-deployment" successfully rolled out
```

```
> kubectl rollout history deployment/myapp-deployment
deployments "myapp-deployment"
REVISION  CHANGE-CAUSE
1         <none>
2         kubectl apply --filename=deployment-definition.yml --record=true
```

Deployment Strategy



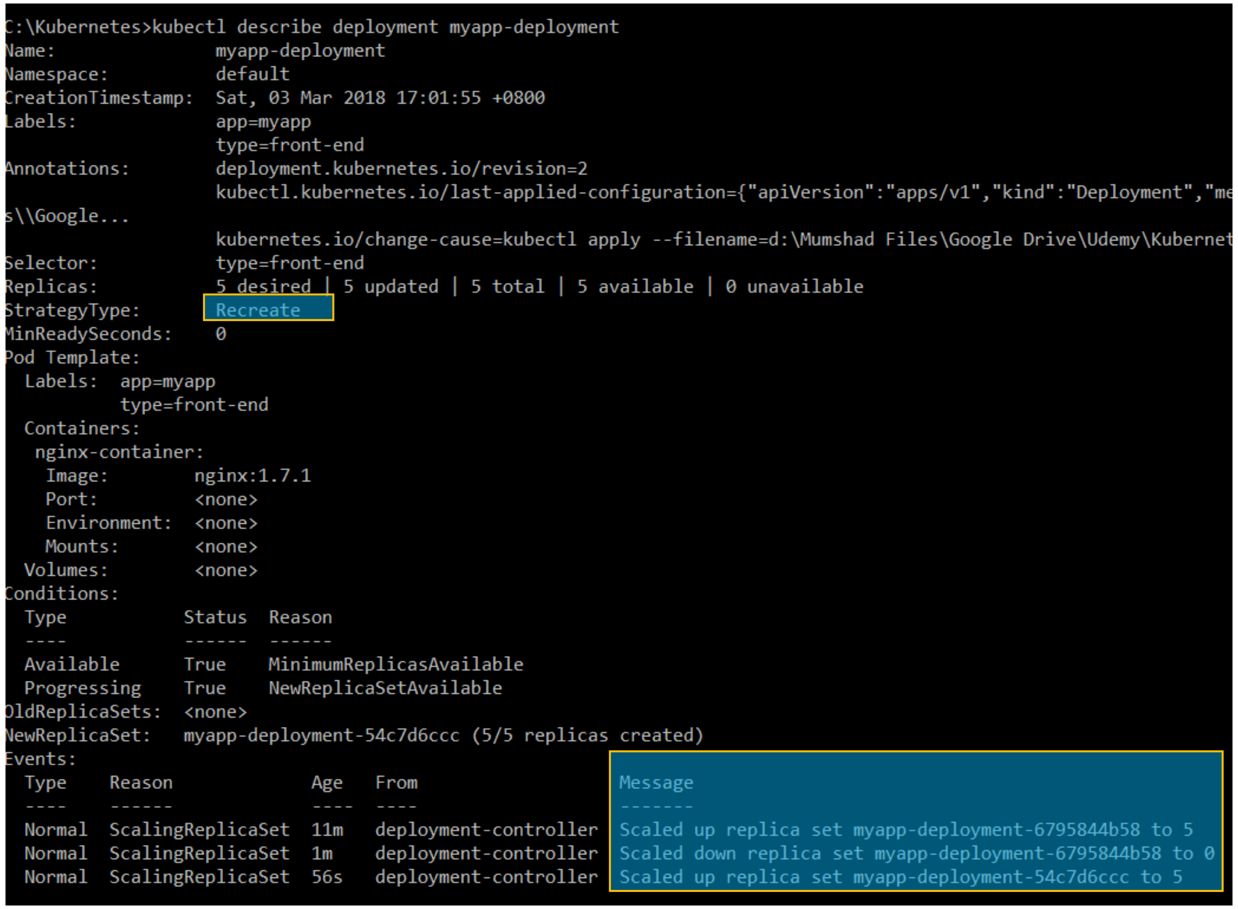
Kubectl apply

```
> kubectl apply -f deployment-definition.yml
deployment "myapp-deployment" configured
```

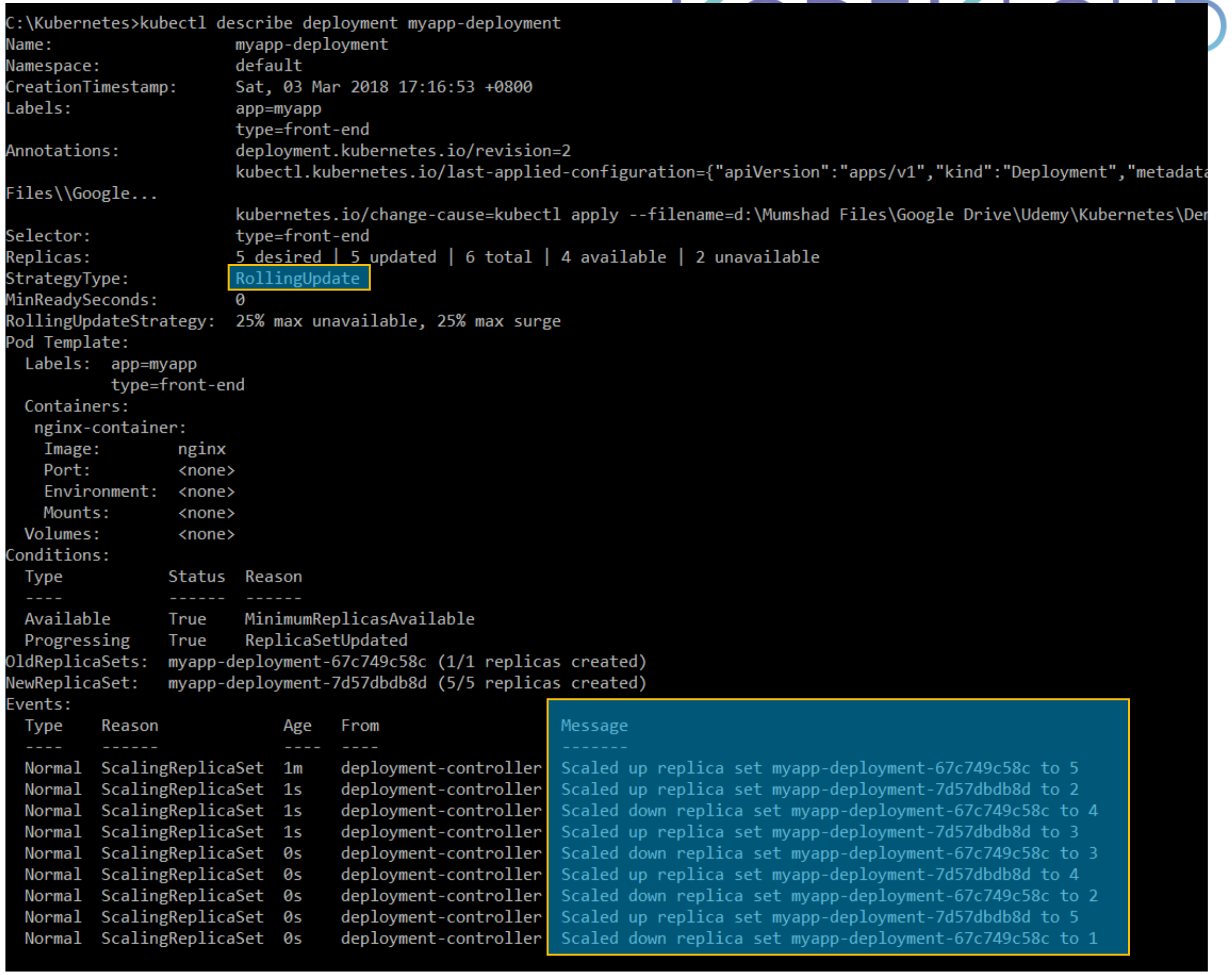
```
> kubectl set image deployment/myapp-deployment \
                nginx=nginx:1.9.1
deployment "myapp-deployment" image is updated
```

deployment-definition.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
  labels:
      app: myapp
      type: front-end
spec:
  template:
    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx:1.7.1

  replicas: 3
  selector:
    matchLabels:
      type: front-end
```



```
C:\Kubernetes>kubectl describe deployment myapp-deployment
Name:                   myapp-deployment
Namespace:              default
CreationTimestamp:      Sat, 03 Mar 2018 17:01:55 +0800
Labels:                 app=myapp
                        type=front-end
Annotations:            deployment.kubernetes.io/revision=2
                        kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"Deployment","me
s\\Google...
                        kubernetes.io/change-cause=kubectl apply --filename=d:\Mumshad Files\Google Drive\Udemy\Kubernet
Selector:               type=front-end
Replicas:               5 desired | 5 updated | 5 total | 5 available | 0 unavailable
StrategyType:           Recreate
MinReadySeconds:        0
Pod Template:
  Labels:  app=myapp
           type=front-end
  Containers:
   nginx-container:
    Image:        nginx:1.7.1
    Port:         <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   myapp-deployment-54c7d6ccc (5/5 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  11m   deployment-controller  Scaled up replica set myapp-deployment-6795844b58 to 5
  Normal  ScalingReplicaSet  1m    deployment-controller  Scaled down replica set myapp-deployment-6795844b58 to 0
  Normal  ScalingReplicaSet  56s   deployment-controller  Scaled up replica set myapp-deployment-54c7d6ccc to 5
```

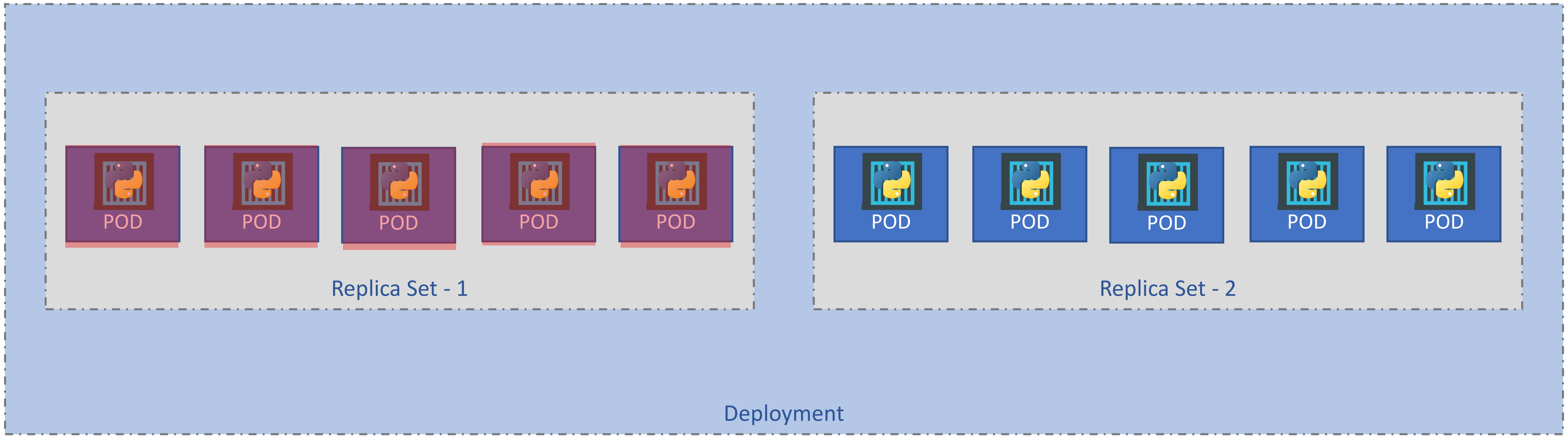


```
C:\Kubernetes>kubectl describe deployment myapp-deployment
Name:                   myapp-deployment
Namespace:              default
CreationTimestamp:      Sat, 03 Mar 2018 17:16:53 +0800
Labels:                 app=myapp
                        type=front-end
Annotations:            deployment.kubernetes.io/revision=2
                        kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"Deployment","metadata
Files\\Google...
                        kubernetes.io/change-cause=kubectl apply --filename=d:\Mumshad Files\Google Drive\Udemy\Kubernetes\De
Selector:               type=front-end
Replicas:               5 desired | 5 updated | 6 total | 4 available | 2 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=myapp
           type=front-end
  Containers:
   nginx-container:
    Image:        nginx
    Port:         <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    ReplicaSetUpdated
OldReplicaSets:  myapp-deployment-67c749c58c (1/1 replicas created)
NewReplicaSet:   myapp-deployment-7d57dbdb8d (5/5 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  1m    deployment-controller  Scaled up replica set myapp-deployment-67c749c58c to 5
  Normal  ScalingReplicaSet  1s    deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 2
  Normal  ScalingReplicaSet  1s    deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 4
  Normal  ScalingReplicaSet  1s    deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 3
  Normal  ScalingReplicaSet  0s    deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 3
  Normal  ScalingReplicaSet  0s    deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 4
  Normal  ScalingReplicaSet  0s    deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 2
  Normal  ScalingReplicaSet  0s    deployment-controller  Scaled up replica set myapp-deployment-7d57dbdb8d to 5
  Normal  ScalingReplicaSet  0s    deployment-controller  Scaled down replica set myapp-deployment-67c749c58c to 1
```

Recreate

RollingUpdate

Upgrades

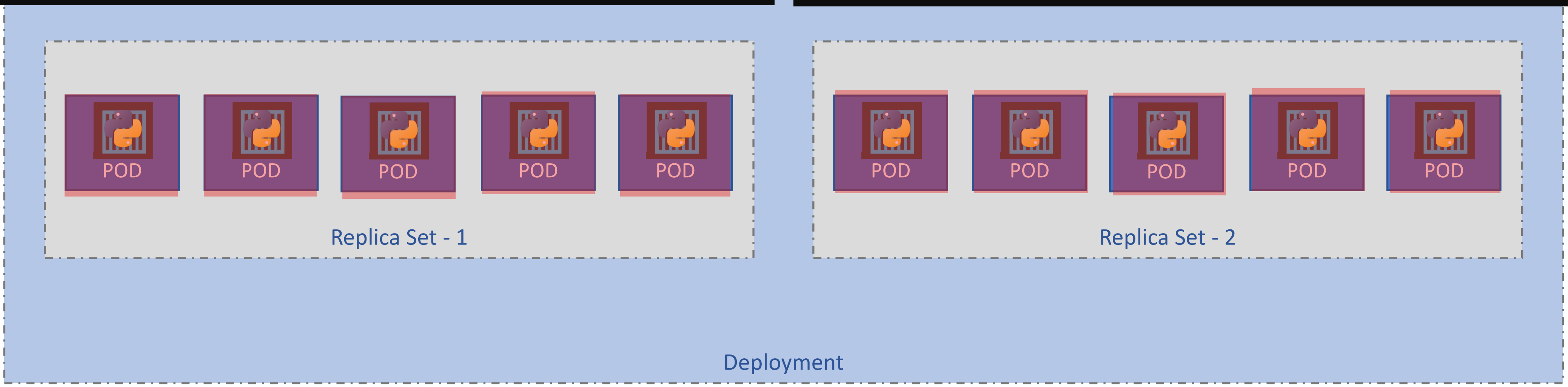


```
> kubectl get replicasets
NAME                        DESIRED    CURRENT    READY     AGE
myapp-deployment-67c749c58c 0          0          0         22m
myapp-deployment-7d57dbdb8d 5          5          5         20m
```

Rollback

```
> kubectl get replicasets
NAME                         DESIRED   CURRENT   READY   AGE
myapp-deployment-67c749c58c  0         0         0       22m
myapp-deployment-7d57dbdb8d  5         5         5       20m
```

```
> kubectl get replicasets
NAME                         DESIRED   CURRENT   READY   AGE
myapp-deployment-67c749c58c  5         5         5       22m
myapp-deployment-7d57dbdb8d  0         0         0       20m
```



```
> kubectl rollout undo deployment/myapp-deployment
deployment “myapp-deployment” rolled back
```

kubectl run

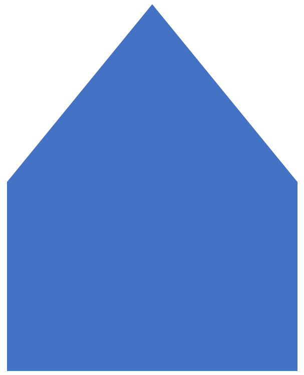
```
> kubectl run nginx --image=nginx
deployment "nginx" created
```

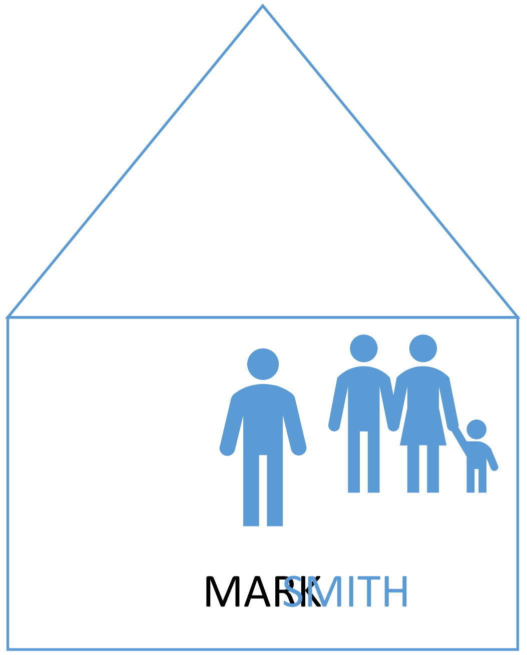
Summarize Commands

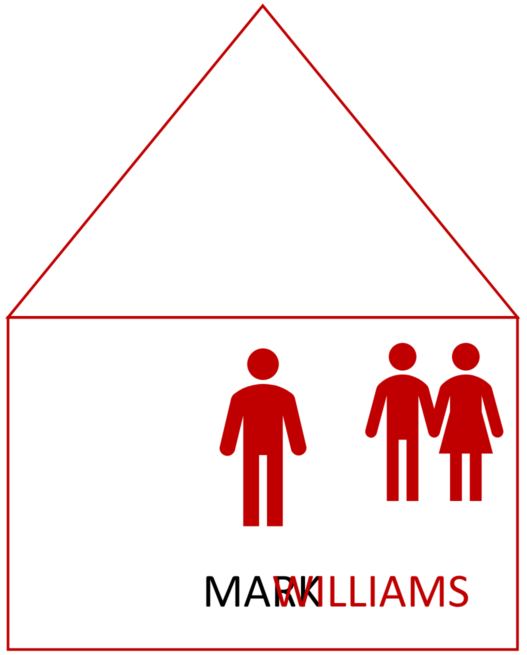
Create	`> kubectl create –f deployment-definition.yml`
Get	`> kubectl get deployments`
Update	`> kubectl apply –f deployment-definition.yml`
	`> kubectl set image deployment/myapp-deployment nginx=nginx:1.9.1`
Status	`> kubectl rollout status deployment/myapp-deployment`
	`> kubectl rollout history deployment/myapp-deployment`
Rollback	`> kubectl rollout undo deployment/myapp-deployment`

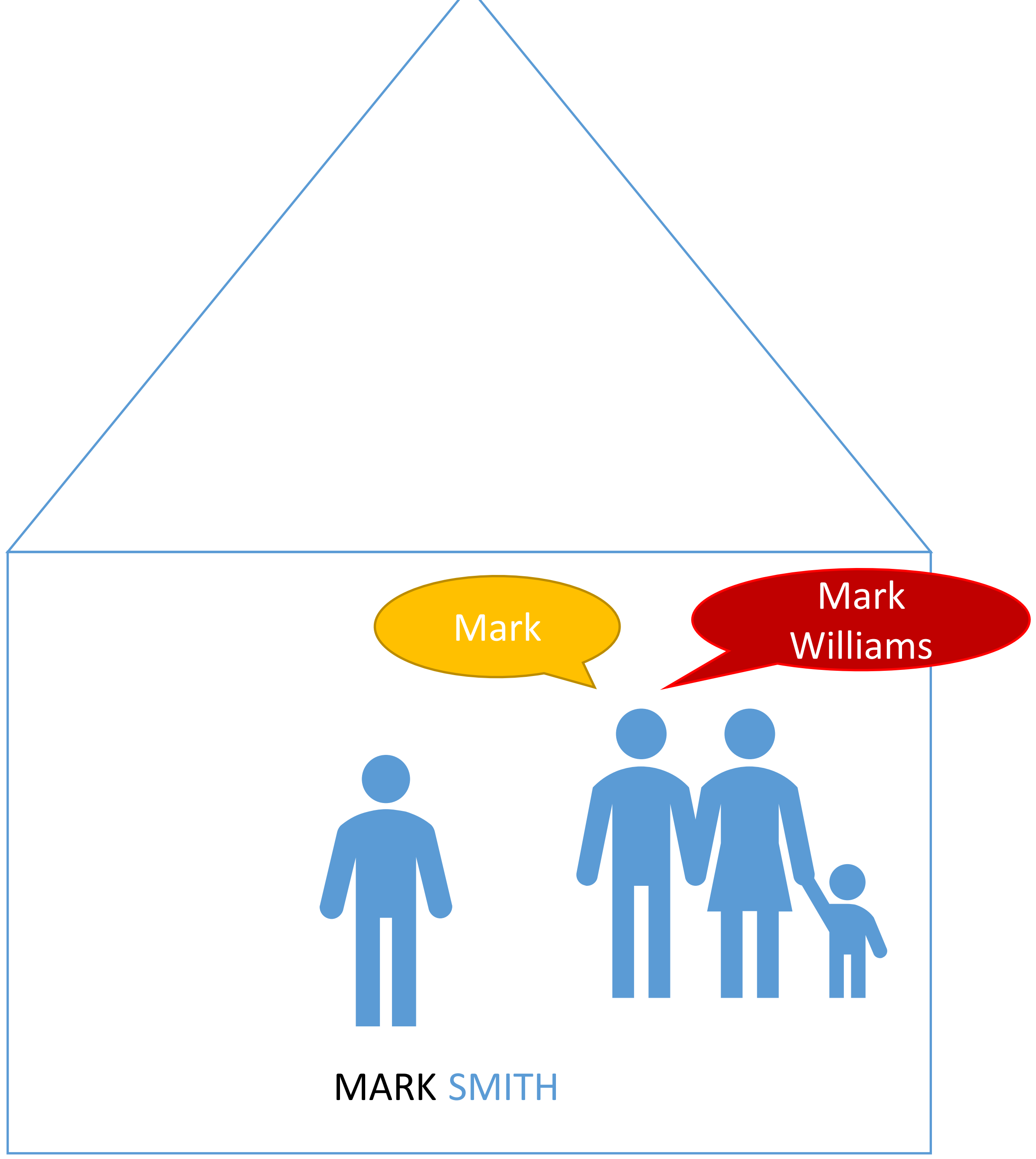
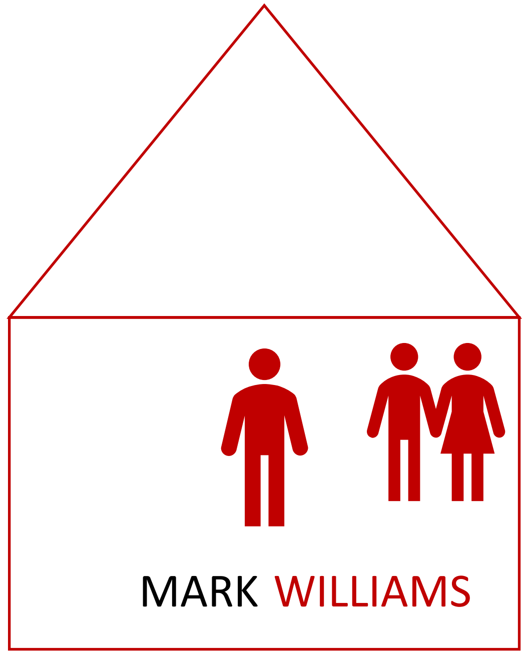
KODEKLOUD

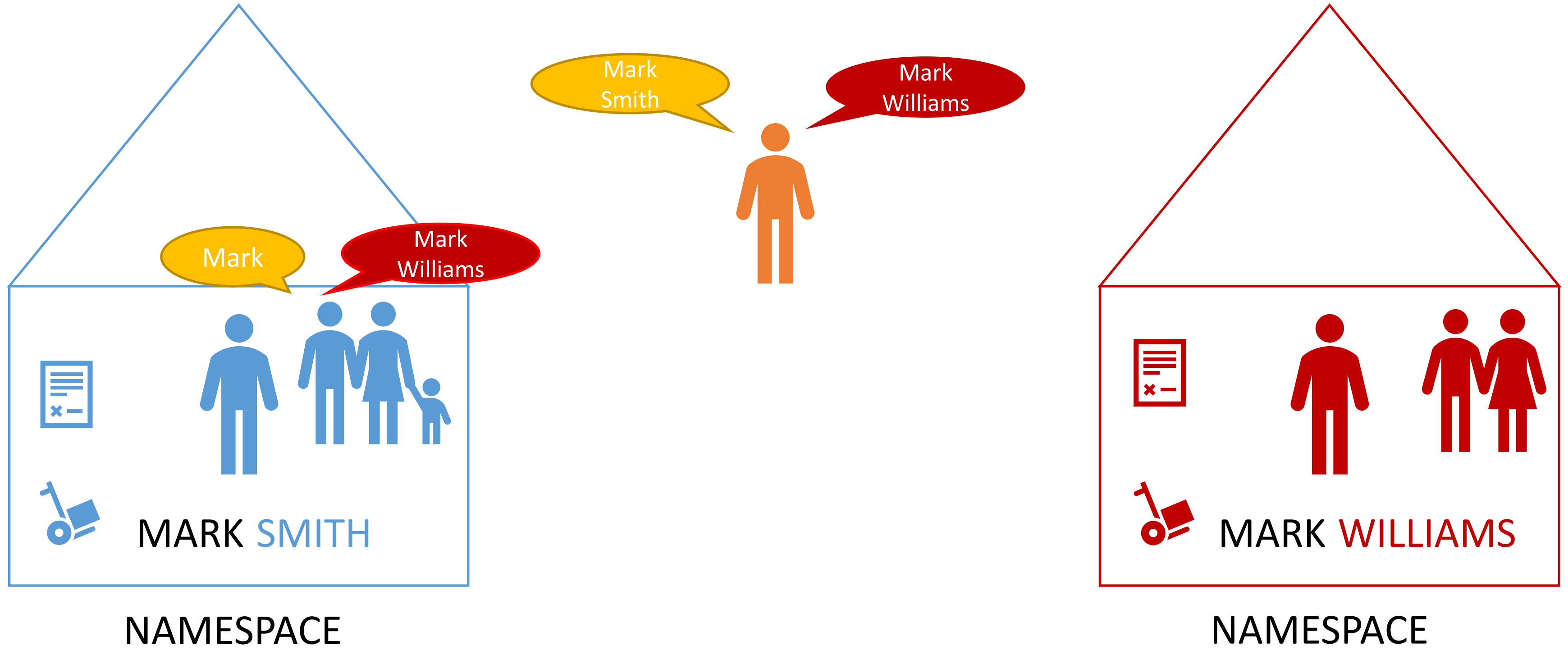
Namespaces

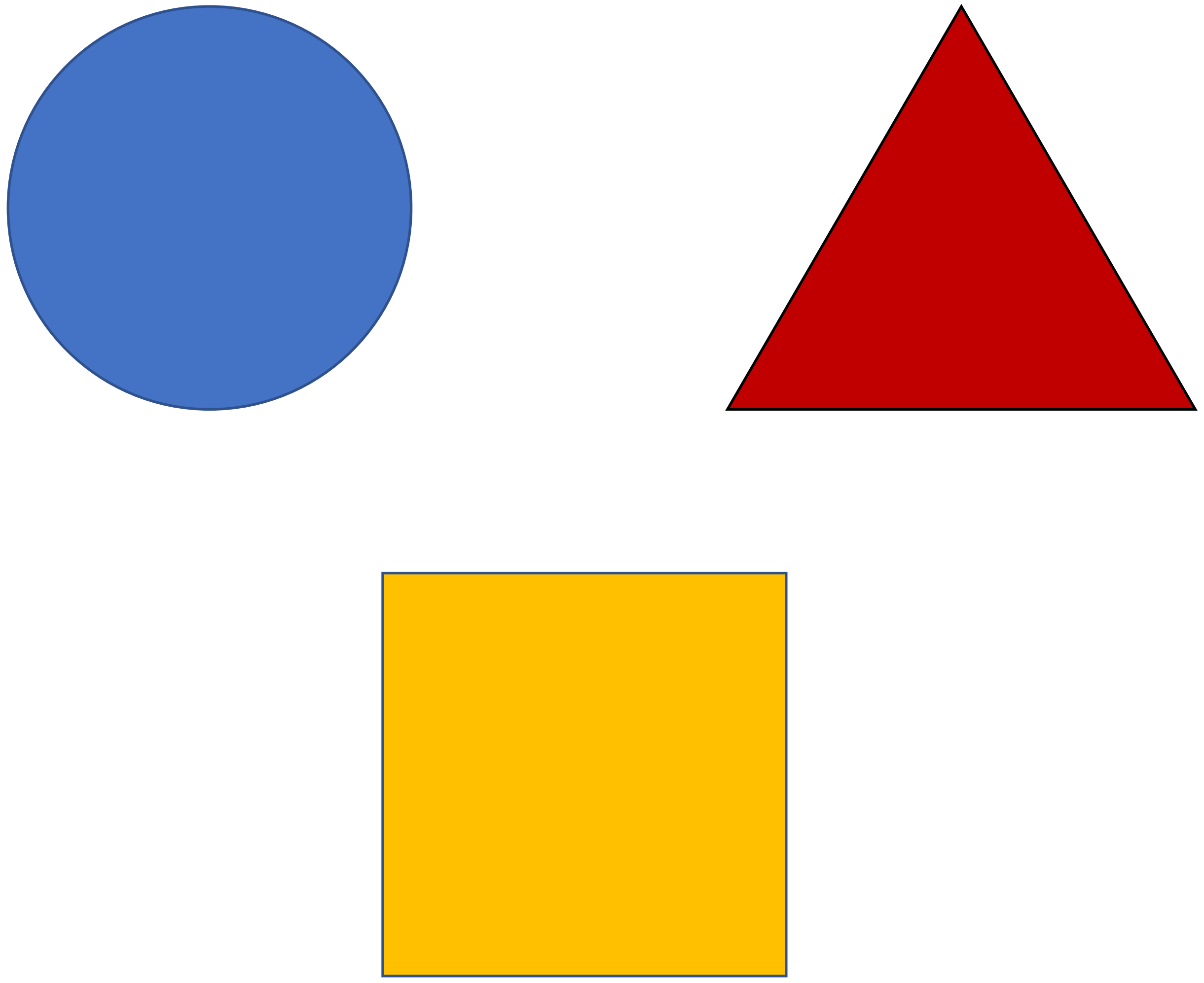


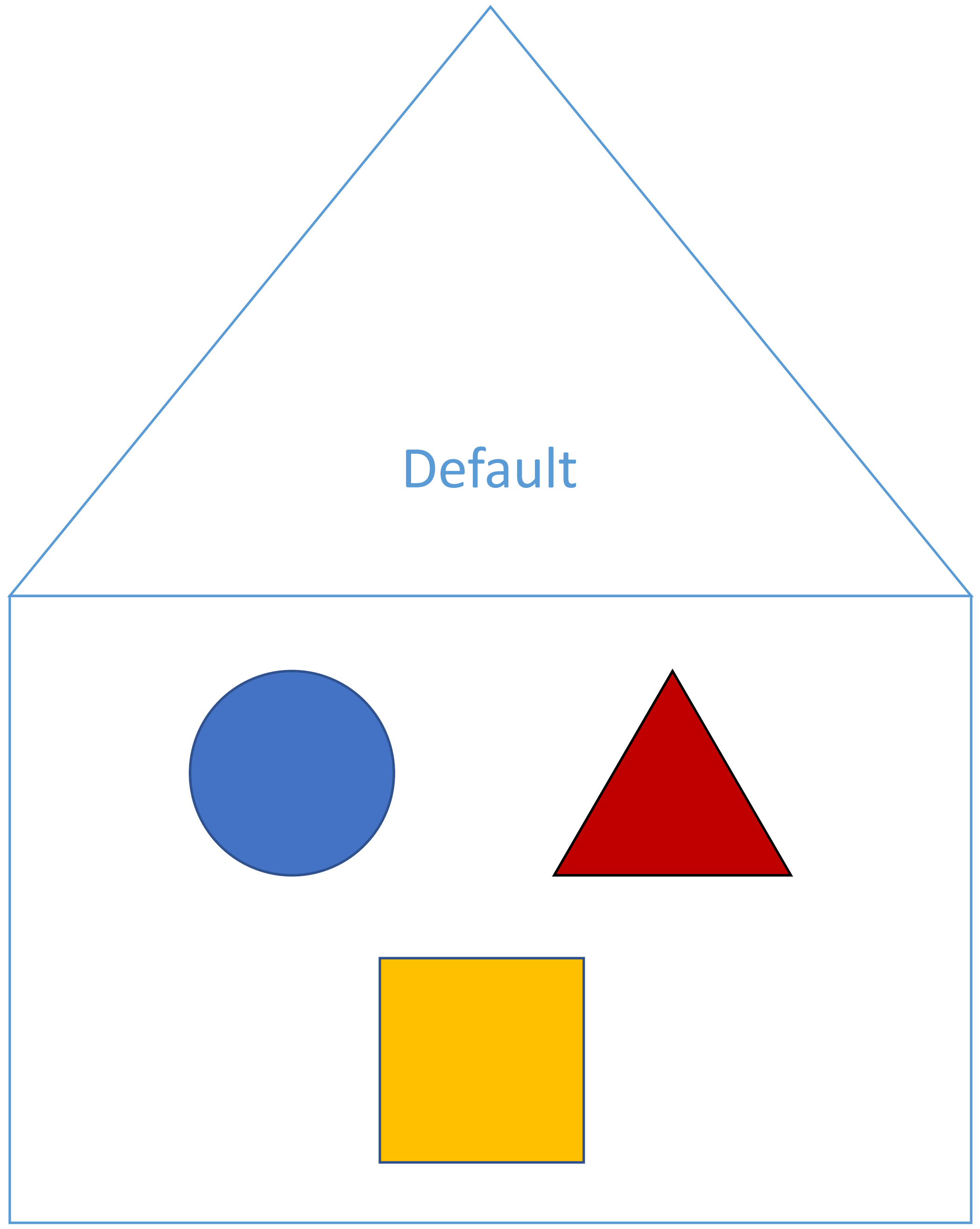

MARK
SMITH


MARK
WILLIAMS


Mark
Mark Williams
MARK SMITH

MARK WILLIAMS

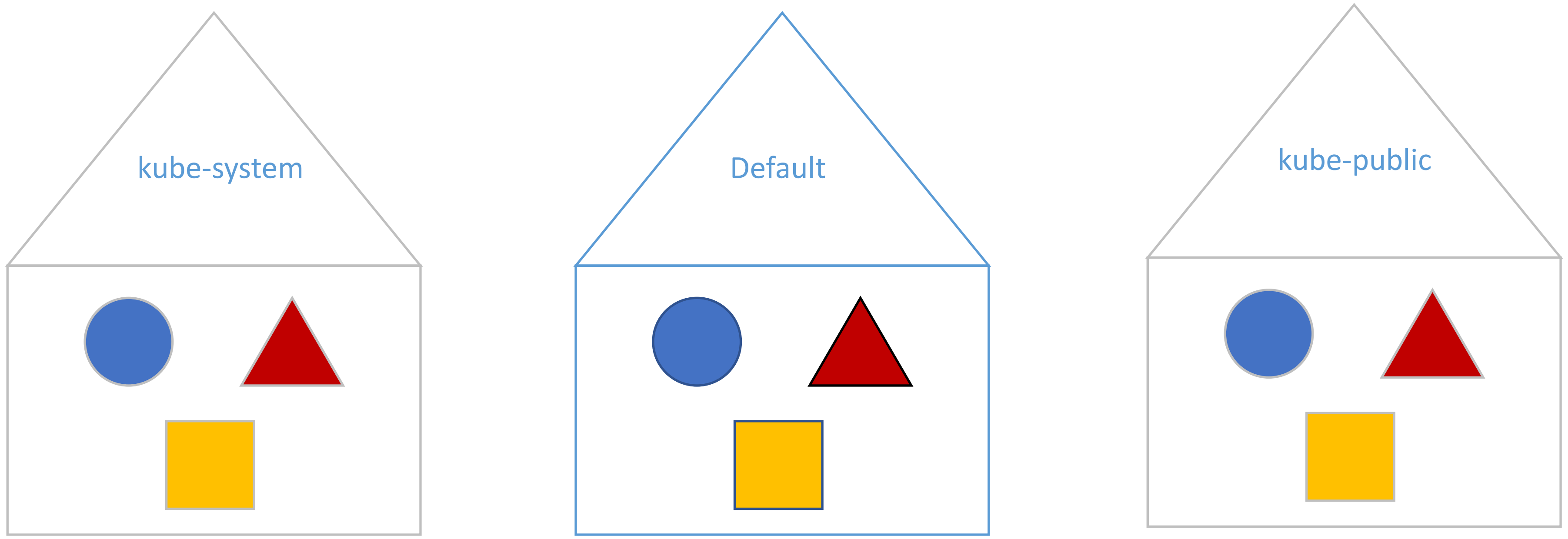

Mark
Smith
Mark
Williams
Mark
Mark
Williams
MARK SMITH
NAMESPACE
MARK WILLIAMS
NAMESPACE



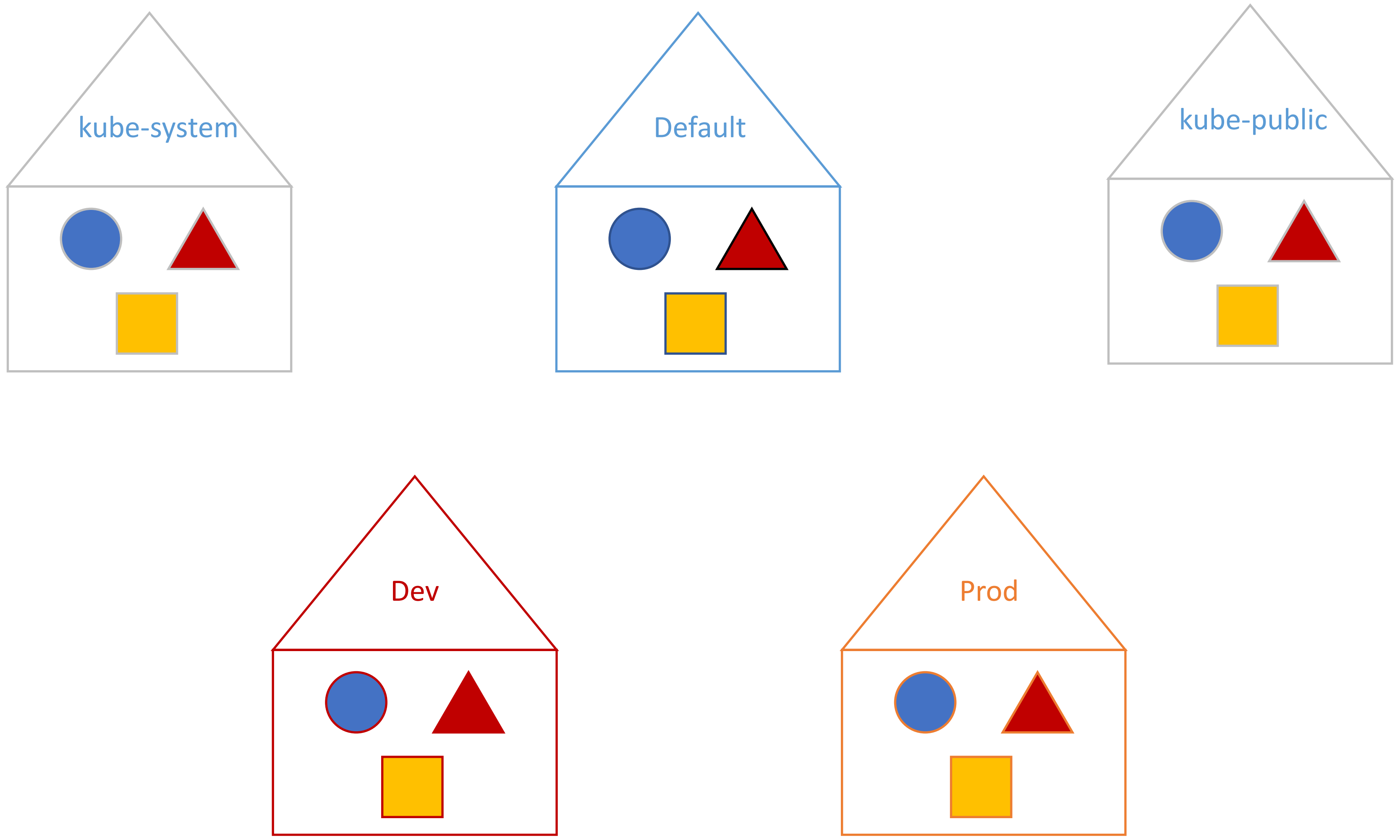

Default

kube-system namespace is created for intern purpose Pods, so that they are isolated from the default namespace.

Here, resources available to all users should be created.

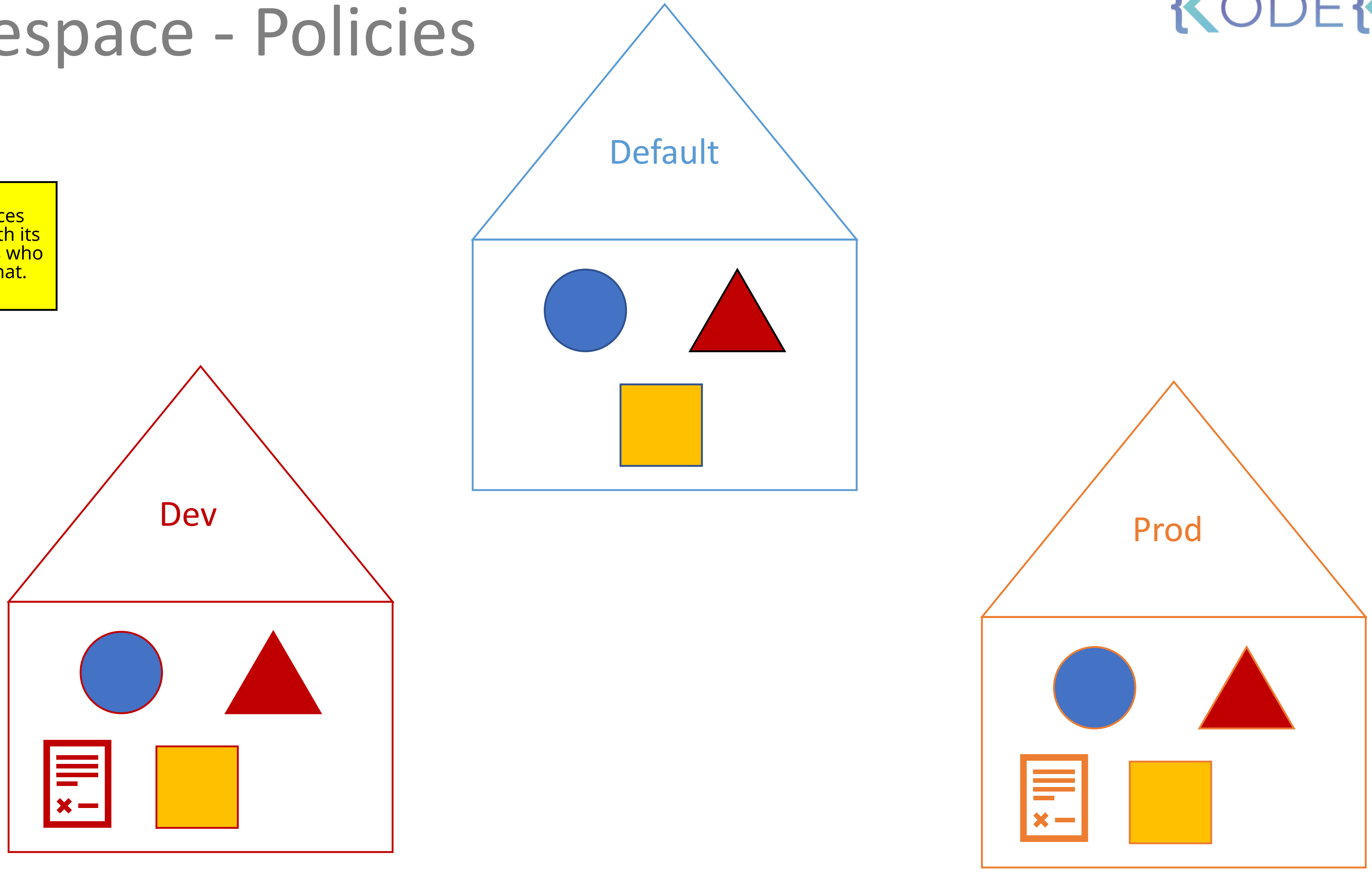


Namespace - Isolation

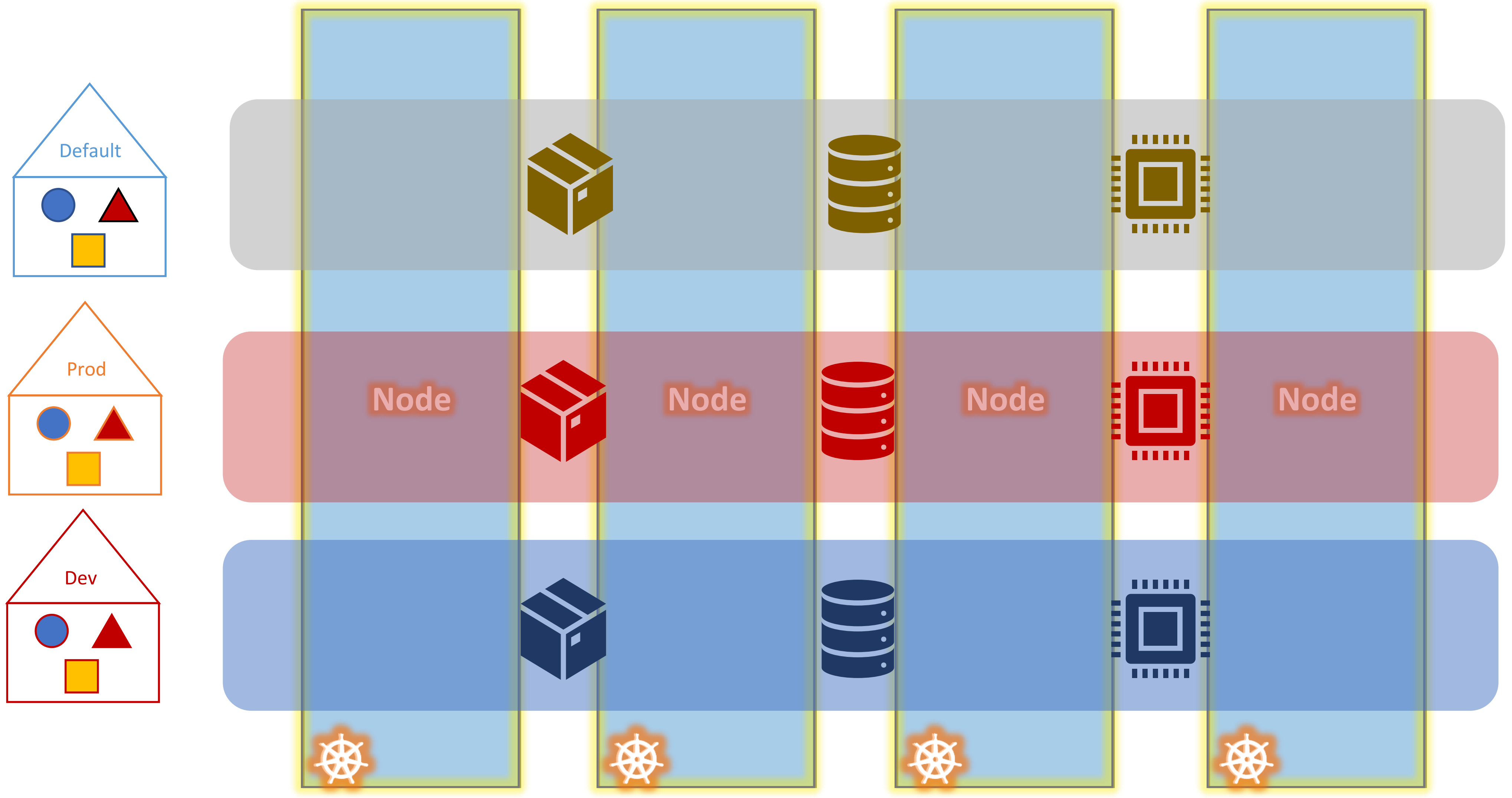


Namespace - Policies

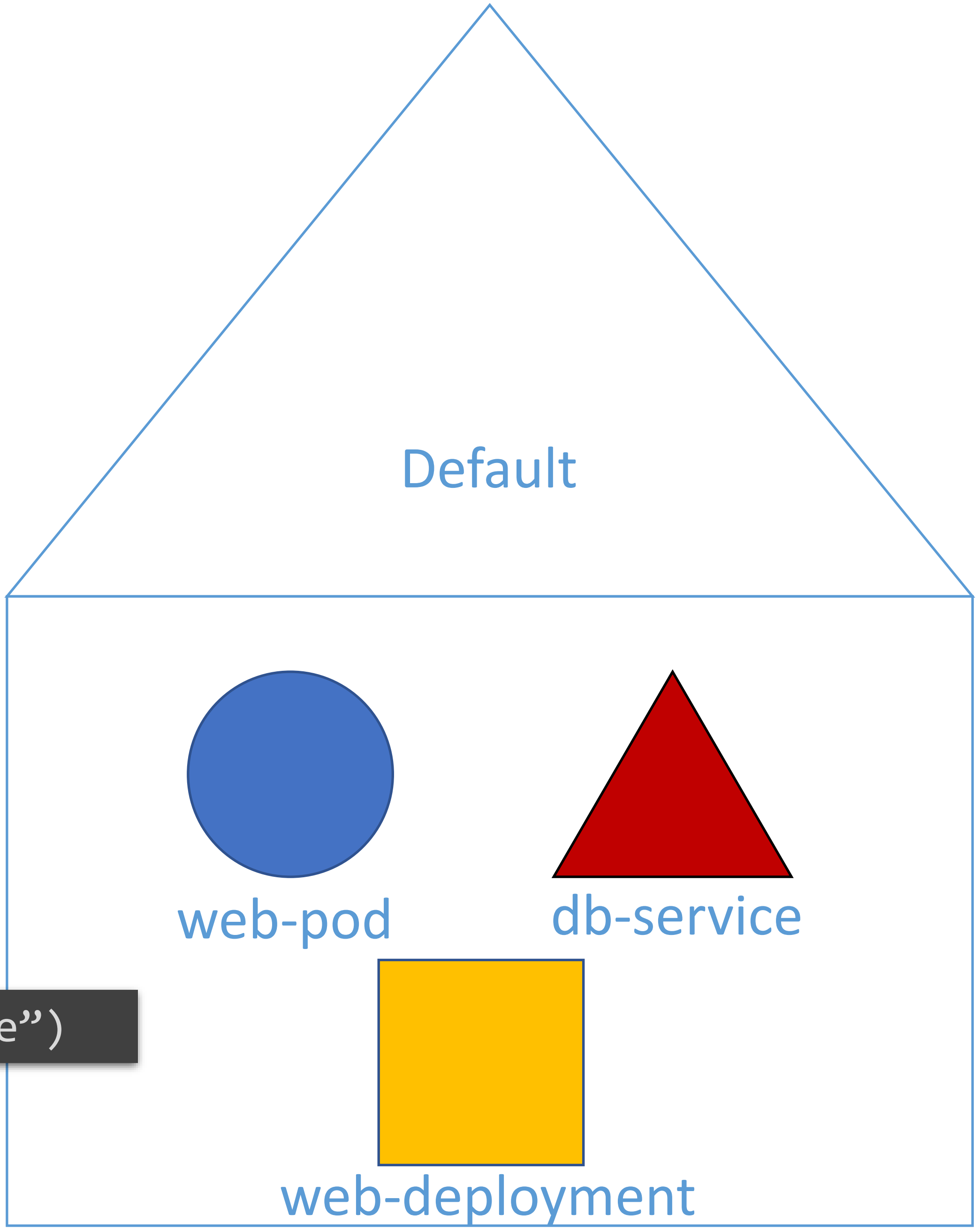
each namespaces comes with its own rules who can do what.



Namespace – Resource Limits

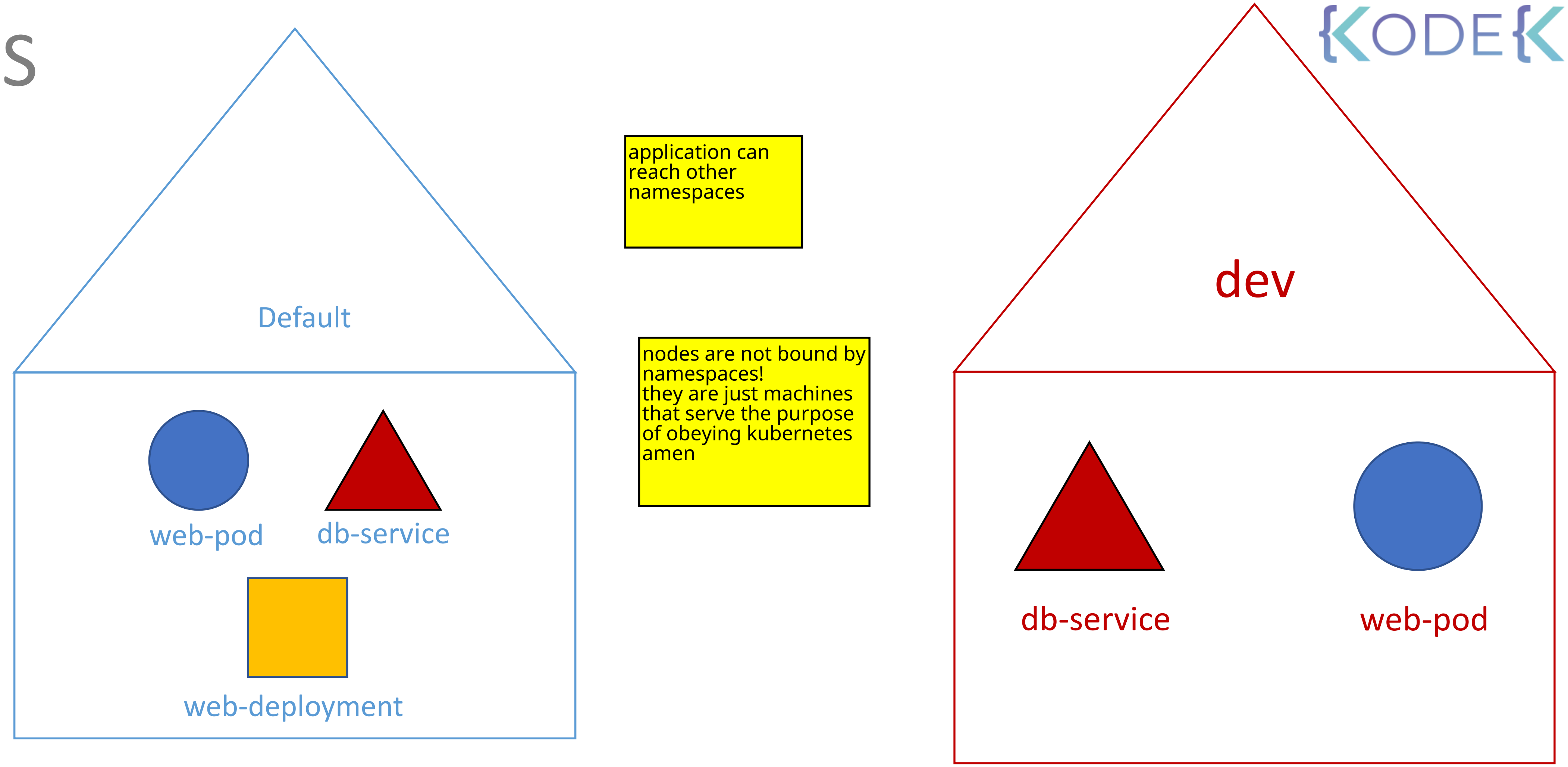


DNS



```
mysql.connect(“db-service”)
```

DNS



application can reach other namespaces

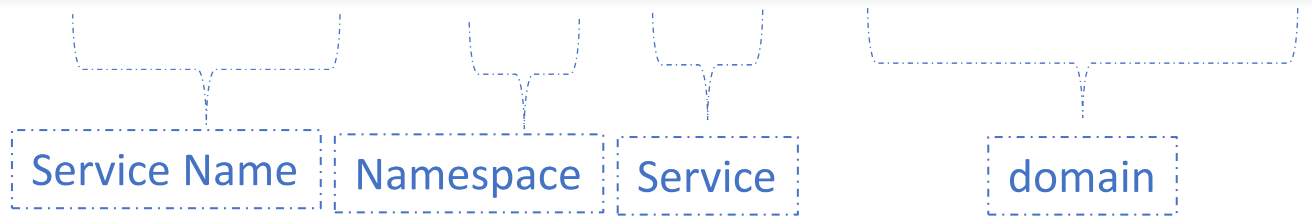
nodes are not bound by namespaces!
they are just machines that serve the purpose of obeying kubernetes
amen

```
mysql.connect(“db-service”)
```

```
mysql.connect(“db-service.dev.svc.cluster.local”)
```

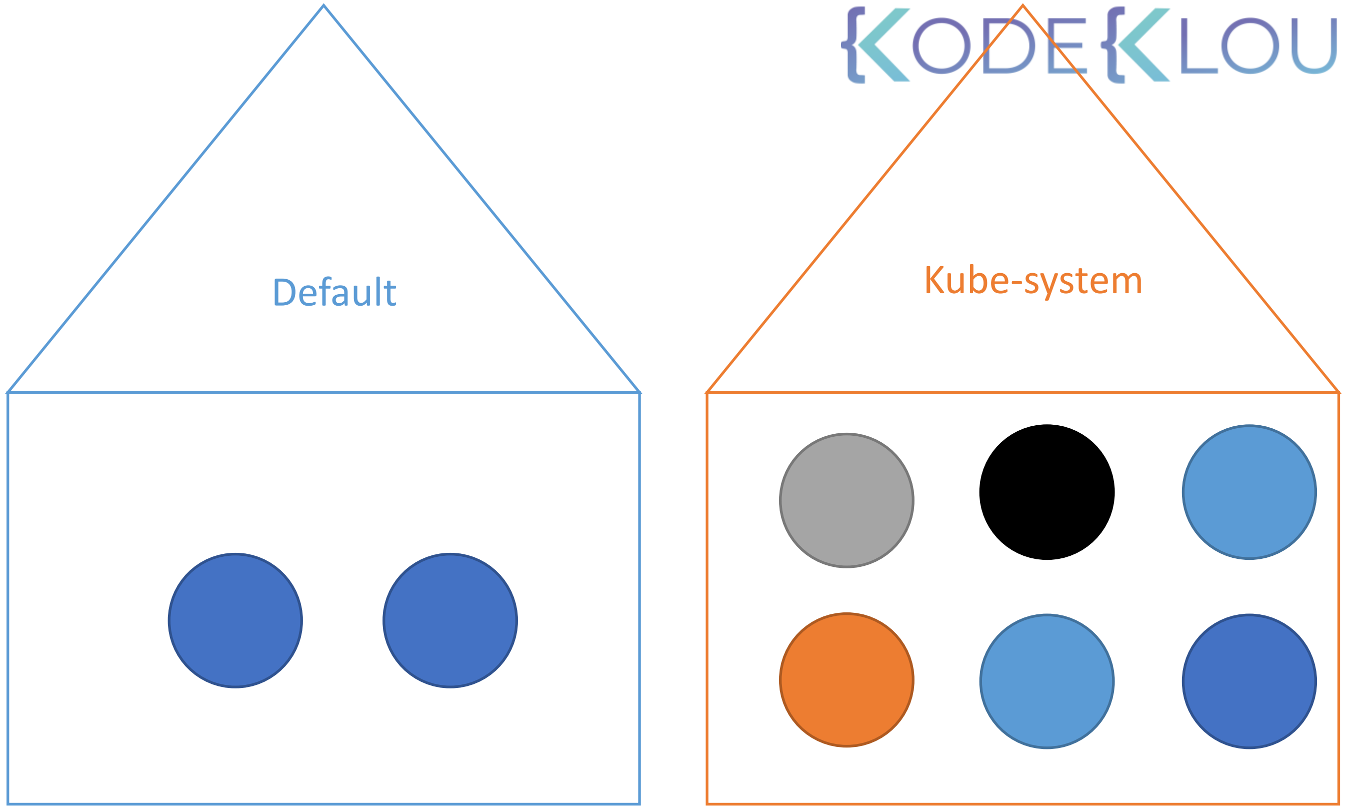
DNS

```
mysql.connect("db-service.dev.svc.cluster.local")
```



```
> kubectl get pods
NAME        READY     STATUS     RESTARTS    AGE
Pod-1       1/1       Running    0           3d
Pod-2       1/1       Running    0           3d
```

```
> kubectl get pods --namespace=kube-system
NAME                                 READY     STATUS     RESTAR
coredns-78fcdf6894-92d52             1/1       Running    7
coredns-78fcdf6894-jx25g             1/1       Running    7
etcd-master                          1/1       Running    7
kube-apiserver-master                1/1       Running    7
kube-controller-manager-master       1/1       Running    7
kube-flannel-ds-amd64-hz4cf          1/1       Running    14
kube-proxy-4b8tn                     1/1       Running    7
kube-proxy-98db4                     1/1       Running    7
kube-proxy-jjrbs                     1/1       Running    7
kube-scheduler-master                1/1       Running    7
```

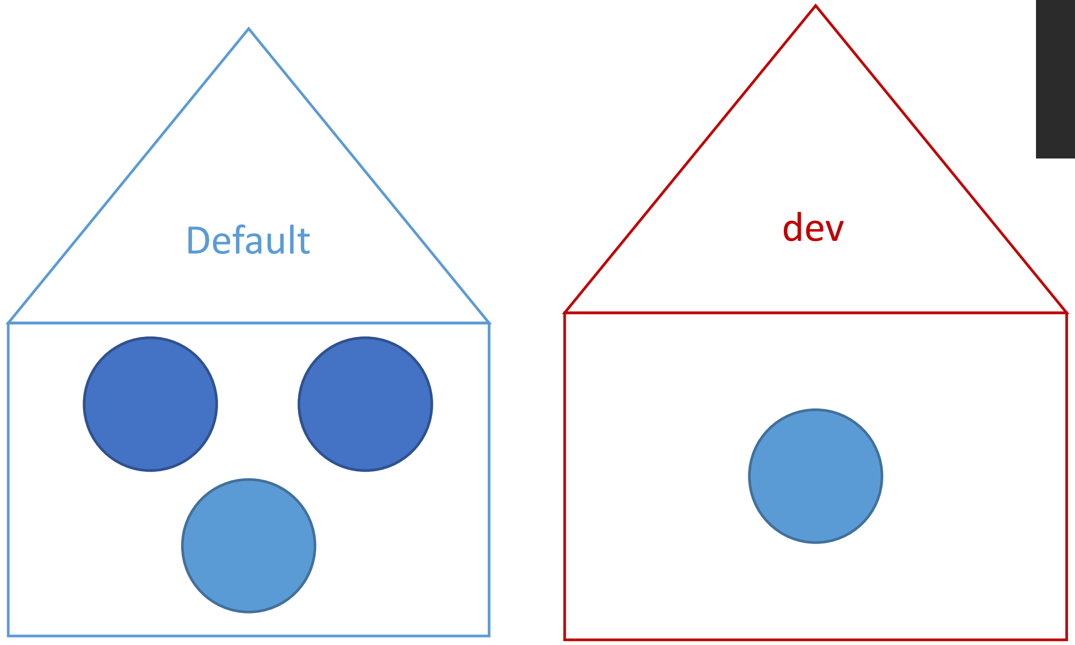


```
> kubectl create -f pod-definition.yml
pod/myapp-pod created
```

```
> kubectl create -f pod-definition.yml --namespace=dev
pod/myapp-pod created
```

pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

metadata:
  name: myapp-pod
  labels:
      app: myapp
      type: front-end
spec:
   containers:
   - name: nginx-container
     image: nginx
```

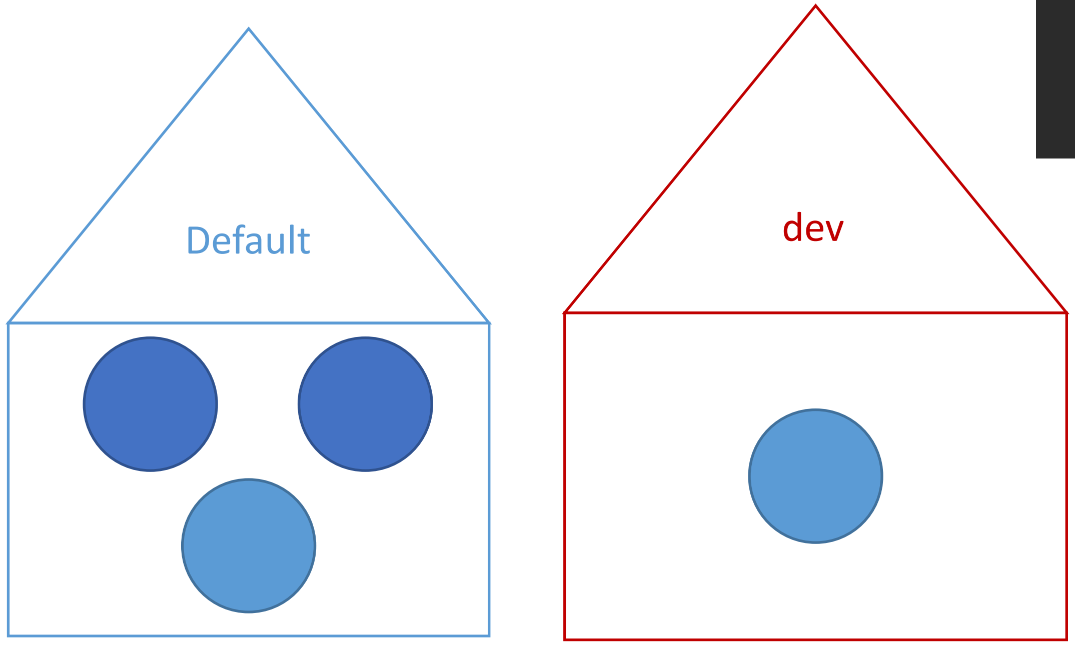


```
> kubectl create -f pod-definition.yml
pod/myapp-pod created
```

```
> kubectl create -f pod-definition.yml --namespace=dev
pod/myapp-pod created
```

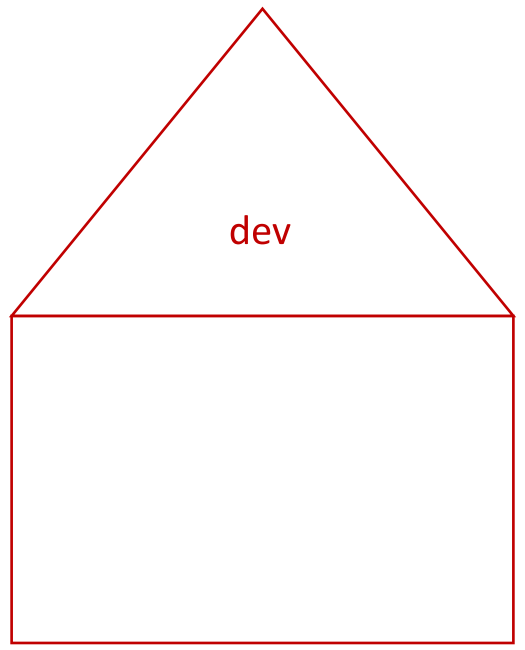
pod-definition.yml

```yaml
apiVersion: v1
kind: Pod

metadata:
  name: myapp-pod
  namespace: dev
  labels:
      app: myapp
      type: front-end
spec:
   containers:
   - name: nginx-container
     image: nginx
```



you can write the namespace in the definition file. this makes sure you always create it in the correct namespace.

Create Namespace

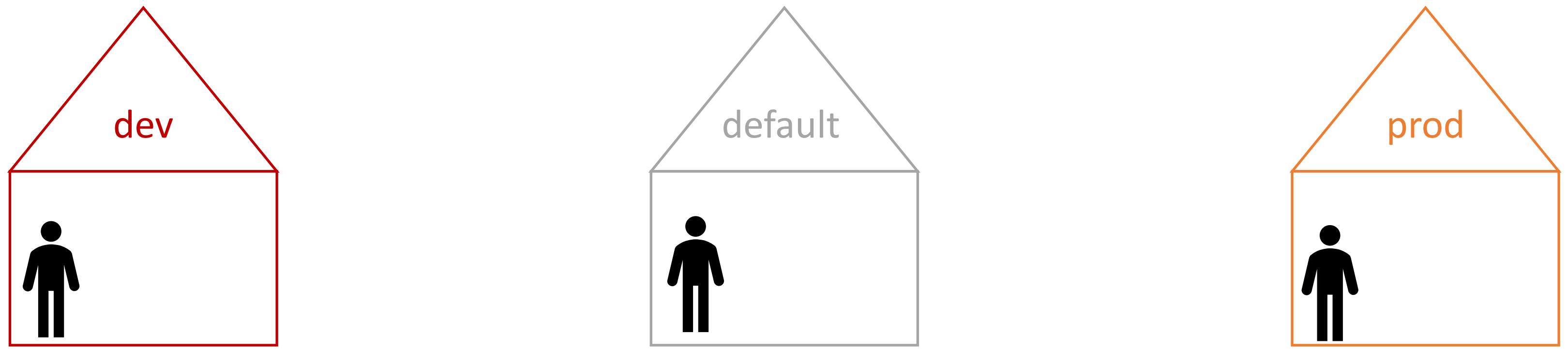


```
namespace-dev.yml
apiVersion: v1
kind: Namespace
metadata:
     name: dev
```

```
> kubectl create –f namespace-dev.yml
namespace/dev created
```

```
> kubectl create namespace dev
namespace/dev created
```

Switch



```
> kubectl get pods --namespace=dev
```

```
> kubectl get pods
```

```
> kubectl get pods --namespace=prod
```

```
> kubectl config set-context $(kubectl config current-context) --namespace=dev
```

```
> kubectl get pods
```

```
> kubectl get pods --namespace=default
```

```
> kubectl get pods --namespace=prod
```

```
> kubectl config set-context $(kubectl config current-context) --namespace=prod
```

```
> kubectl get pods --namespace=dev
```

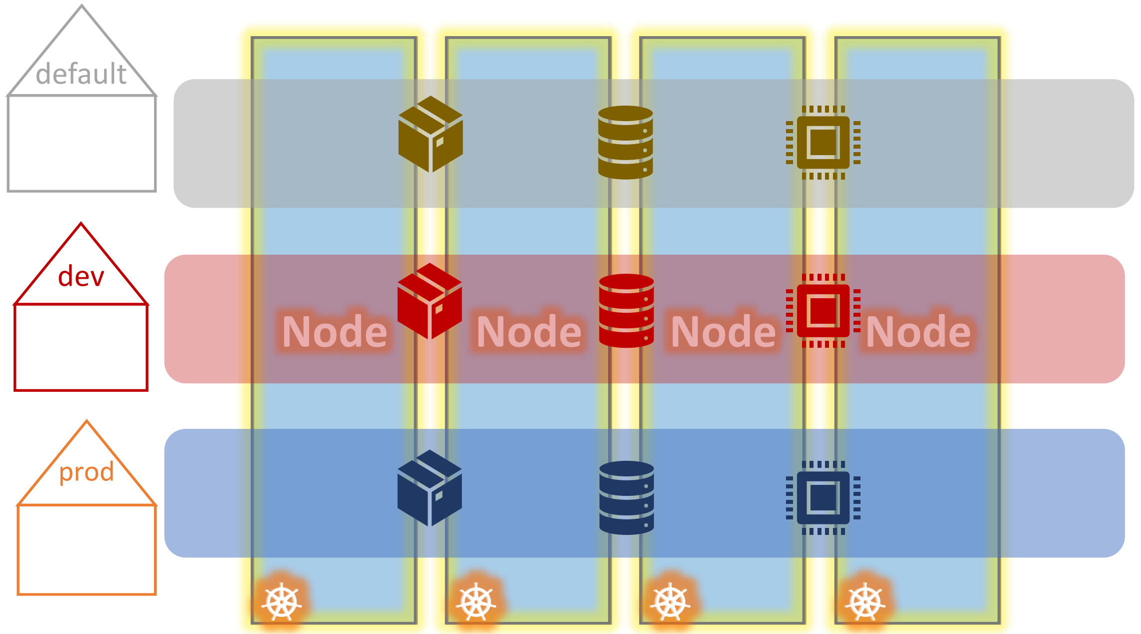
```
> kubectl get pods --namespace=default
```

```
> kubectl get pods
```

```
> kubectl get pods --all-namespaces
> kubectl config set-context $(kubectl config current-context) --namespace=dev
```

Resource Quota

to limit resources in a namespace, you create ResourceQuota.



Compute-quota.yaml

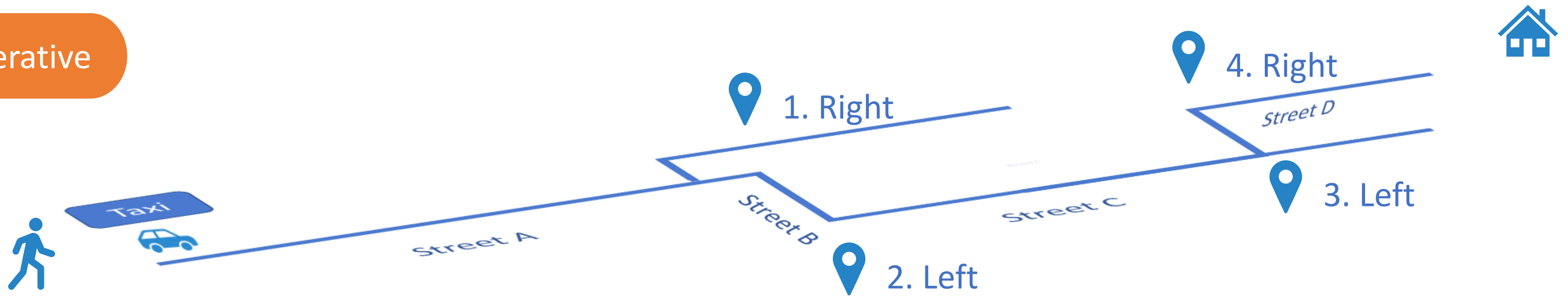
```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
    name: compute-quota
    namespace: dev

spec:
  hard:
    pods: "10"
    requests.cpu: "4"
    requests.memory: 5Gi
    limits.cpu: "10"
    limits.memory: 10Gi
```

```
> kubectl create -f compute-quota.yaml
```

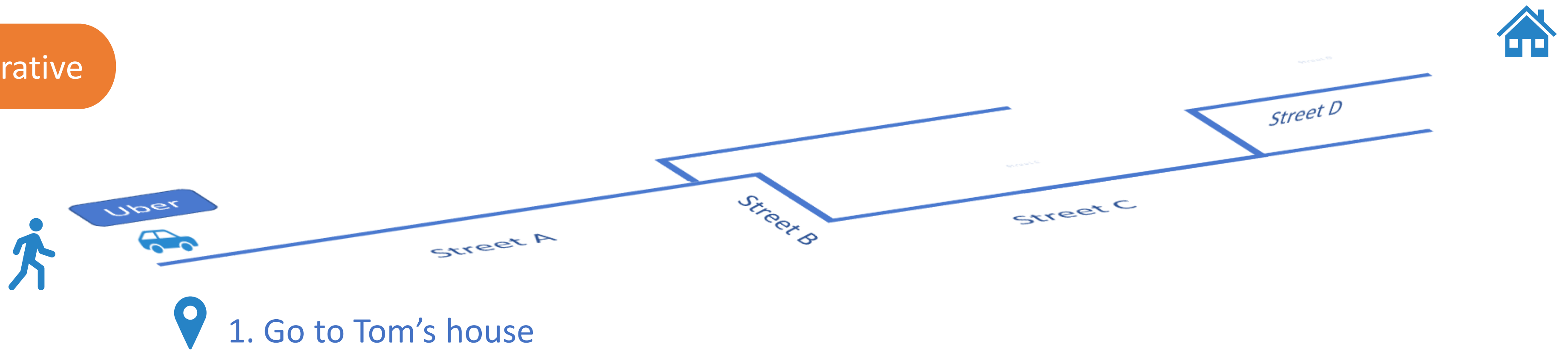
KODEKLOUD

Imperative
vs
Declarative

Imperative



Declarative



Infrastructure as Code

Imperative

```
1. Provision a VM by the name ‘web-server’
2. Install NGINX Software on it
3. Edit configuration file to use port ‘8080’
4. Edit configuration file to web path ‘/var/www/nginx’
5. Load web pages to ‘/var/www/nginx’ from GIT Repo - X
6. Start NGINX server
```

Declarative

```
VM Name: web-server
Package: nnginx:1.18
Port: 8080
Path: /var/www/nginx
Code: GIT Repo - X
```

Kubernetes

Imperative

```
> kubectl run --image=nginx nginx
```

```
> kubectl create deployment --image=nginx nginx
```

```
> kubectl expose deployment nginx --port 80
```

```
> kubectl edit deployment nginx
```

```
> kubectl scale deployment nginx --replicas=5
```

```
> kubectl set image deployment nginx nginx=nginx:1.18
```

```
> kubectl create –f nginx.yaml
```

```
> kubectl replace –f nginx.yaml
```

```
> kubectl delete –f nginx.yaml
```

Declarative

```
> kubectl apply –f nginx.yaml
```

Apply command looks at the existing system and decides what needs to be done to reach the desired state.

Imperative Commands

Create Objects

```
> kubectl run --image=nginx nginx
```

```
> kubectl create deployment --image=nginx nginx
```

```
> kubectl expose deployment nginx --port 80
```

Update Objects

```
> kubectl edit deployment nginx
```

```
> kubectl scale deployment nginx --replicas=5
```

```
> kubectl set image deployment nginx nginx=nginx:1.18
```

Imperative Object Configuration Files

Create Objects

```
> kubectl create –f nginx.yaml
```

Update Objects

```
> kubectl edit deployment nginx
```

nginx.yaml

```
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
     app: myapp
     type: front-end
spec:
   containers:
   - name: nginx-container
     image: nginx
```

Imperative Object Configuration Files

Create Objects

```
> kubectl create -f nginx.yaml
```

Update Objects

```
> kubectl edit deployment nginx
```

The change I perform with kubectl edit is not applied to the local file!

nginx.yaml

```
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

Local file

pod-definition

```
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx:1.18

status:
  conditions:
  - lastProbeTime: null
    status: "True"
    type: Initialized
```

Kubernetes Memory

Imperative Object Configuration Files

Create Objects

```
> kubectl create -f nginx.yaml
```

Update Objects

first edit local file, then use replace. better than kubectl edit

```
> kubectl edit deployment nginx
```

```
> kubectl replace -f nginx.yaml
```

```
> kubectl replace --force -f nginx.yaml
```

```
> kubectl create -f nginx.yaml
Error from server (AlreadyExists): error when creating "nginx.yaml": pods ”myapp-pod" already exists
```

```
> kubectl replace -f nginx.yaml
Error from server (Conflict): error when replacing "nginx.yaml": Operation cannot be fulfilled on pods
”myapp-pod"
```

create failes if Pod exist, replace fails if Pod does not exist - imperative approach is tricky to manage.

nginx.yaml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end-service
spec:
  containers:
  - name: nginx-container
    image: nginx:1.18
```

Declarative

Create Objects

```
> kubectl apply -f nginx.yaml
```

```
> kubectl apply -f /path/to/config-files
```

Update Objects

```
> kubectl apply -f nginx.yaml
```

nginx.yaml

```yaml
apiVersion: v1
kind: Pod

metadata:
 name: myapp-pod
 labels:
    app: myapp
    type: front-end-service
spec:
  containers:
  - name: nginx-container
    image: nginx:1.18
```

Exam Tips

use imperative for simple tasks.

Create Objects

```
> kubectl apply –f nginx.yaml
```

use declarative for more complex stuff.

```
> kubectl run --image=nginx nginx
```

```
> kubectl create deployment --image=nginx nginx
```

```
> kubectl expose deployment nginx --port 80
```

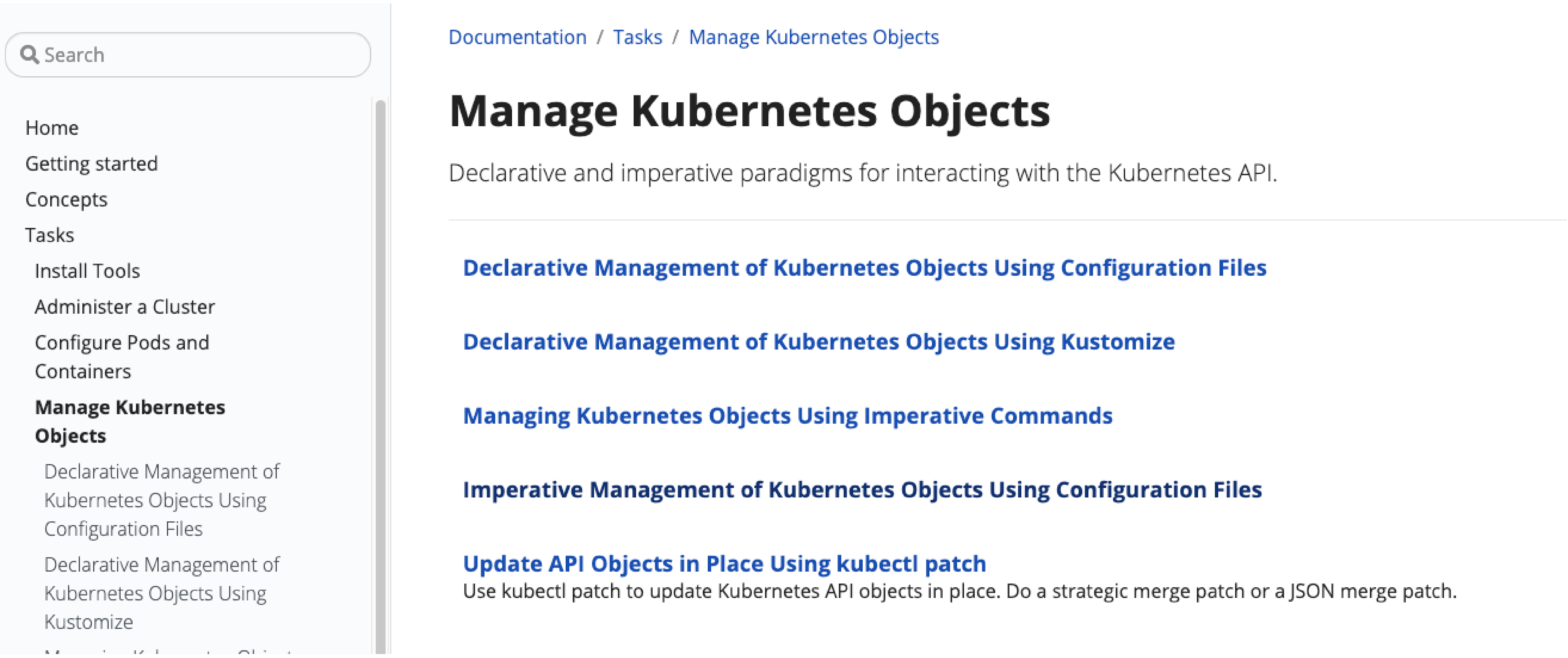
Update Objects

```
> kubectl apply –f nginx.yaml
```

```
> kubectl edit deployment nginx
```

```
> kubectl scale deployment nginx --replicas=5
```

```
> kubectl set image deployment nginx nginx=nginx:1.18
```



Search

Home
Getting started
Concepts
Tasks
Install Tools
Administer a Cluster
Configure Pods and Containers
Manage Kubernetes Objects
Declarative Management of Kubernetes Objects Using Configuration Files
Declarative Management of Kubernetes Objects Using Kustomize

Documentation / Tasks / Manage Kubernetes Objects

Manage Kubernetes Objects

Declarative and imperative paradigms for interacting with the Kubernetes API.

Declarative Management of Kubernetes Objects Using Configuration Files

Declarative Management of Kubernetes Objects Using Kustomize

Managing Kubernetes Objects Using Imperative Commands

Imperative Management of Kubernetes Objects Using Configuration Files

Update API Objects in Place Using kubectl patch
Use kubectl patch to update Kubernetes API objects in place. Do a strategic merge patch or a JSON merge patch.

KODEKLOUD