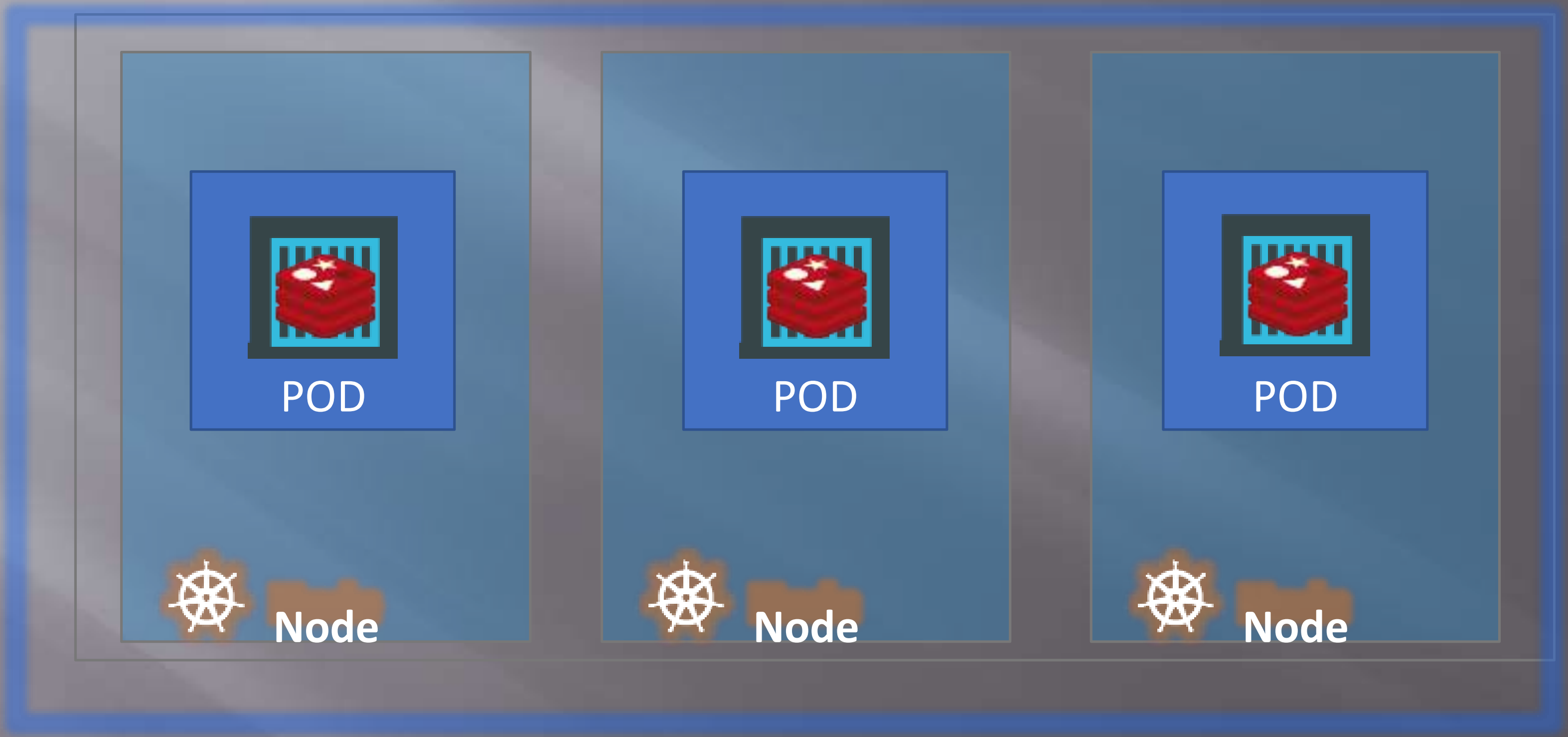
POD

Ravindra Kudache

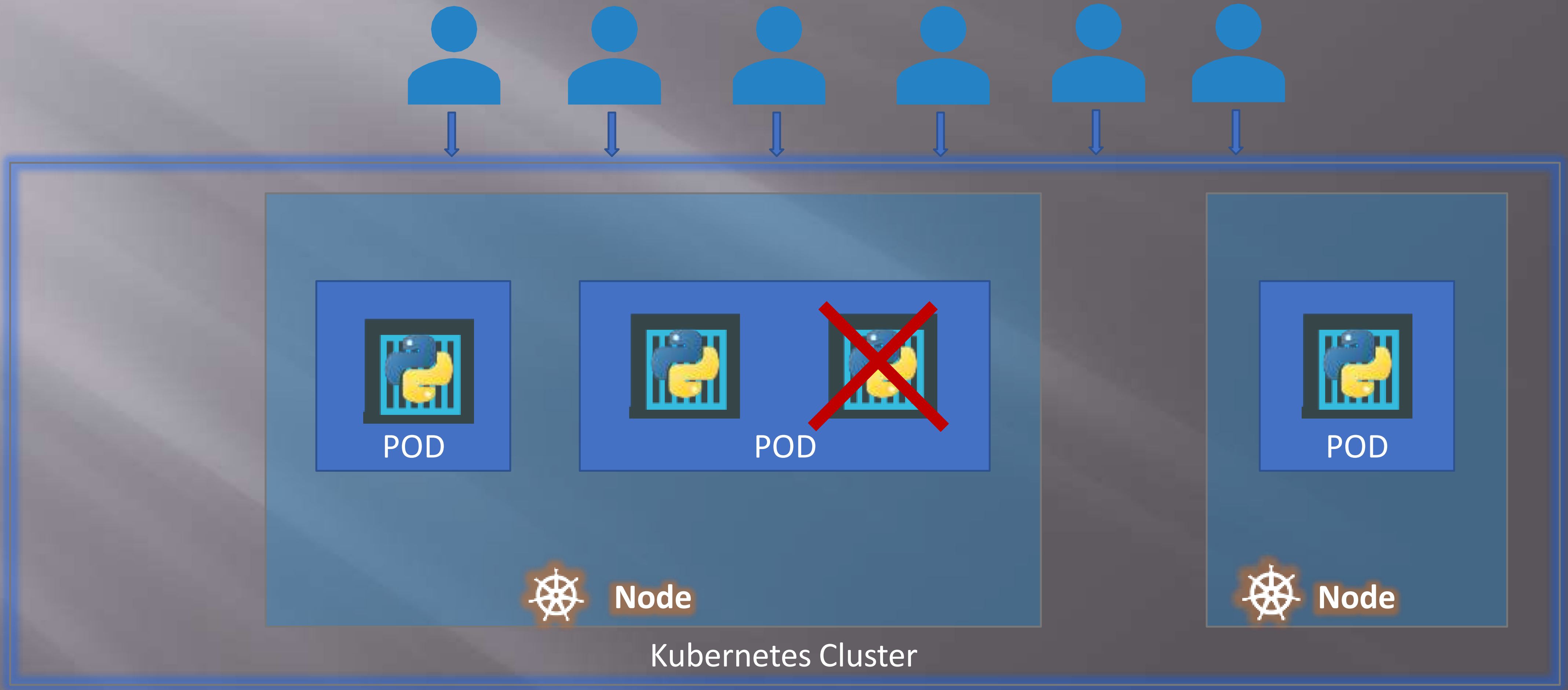
POD



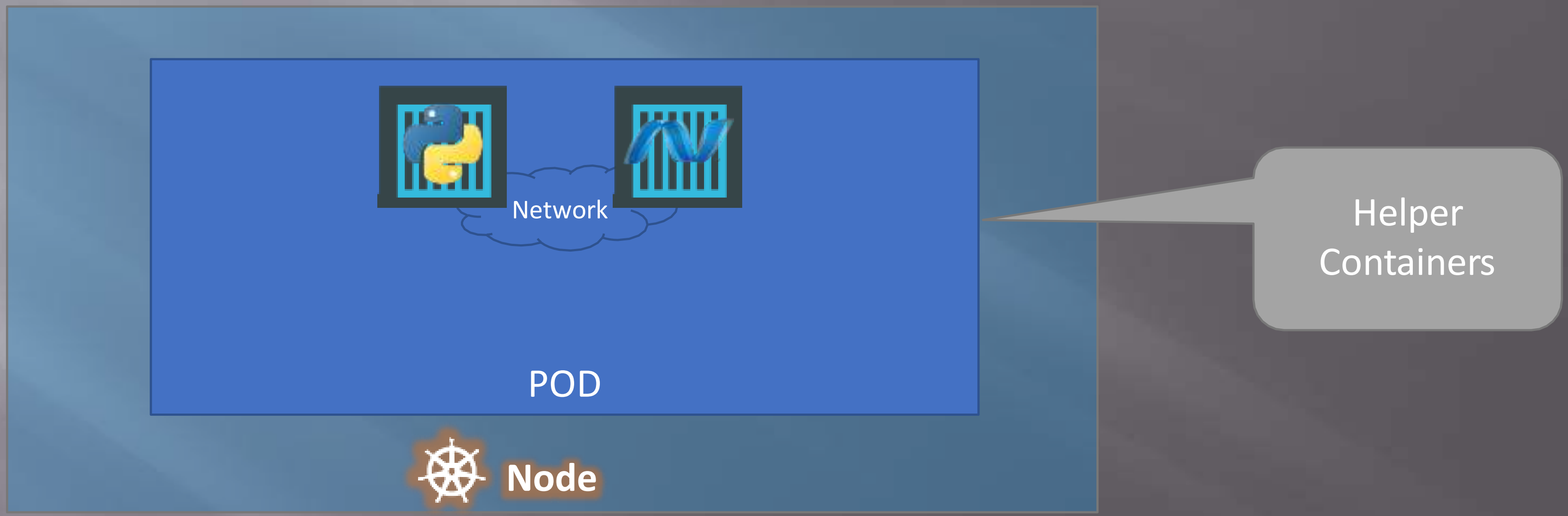
POD

- Smallest thing we can deployed in pod kubernets
- We cannot deploy individual container by themselves as we could with Docker, compose or web app.
- POD keep those container put together which are tightly coupled with each other
- Pod help put one or more container put together

POD



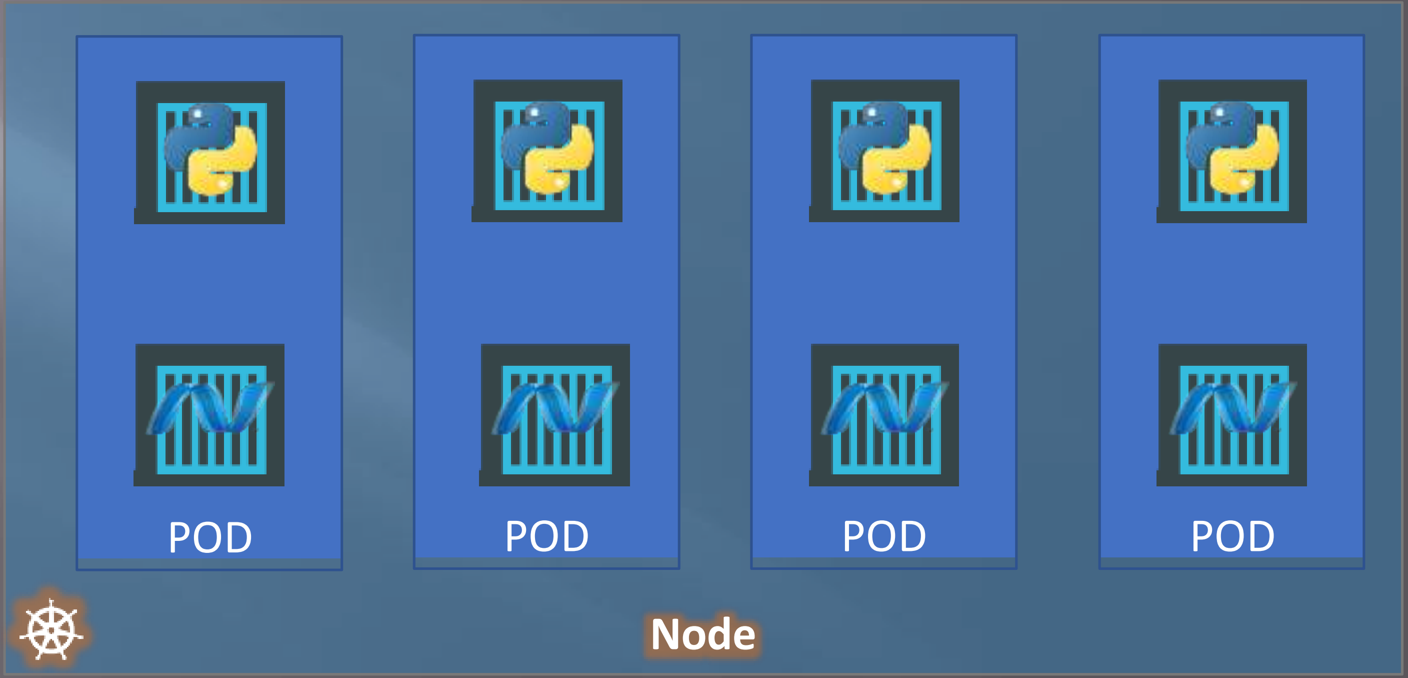
Multi-Container PODs



PODs Again!

```
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

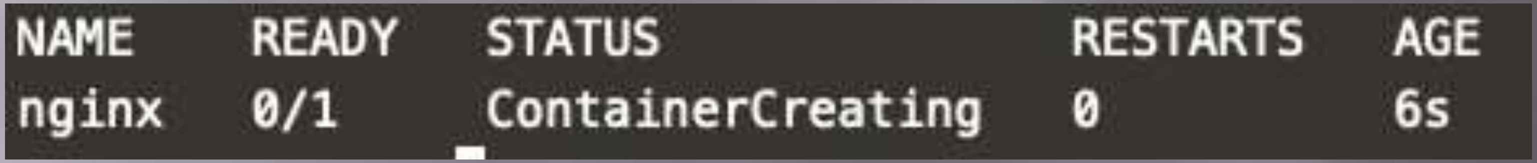


Note: I am avoiding networking and load balancing details to keep explanation simple.

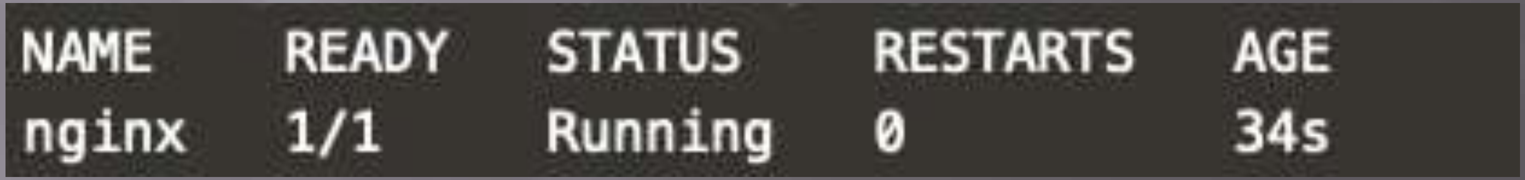
kubectl

```
• kubectl run nginx--image nginx
```

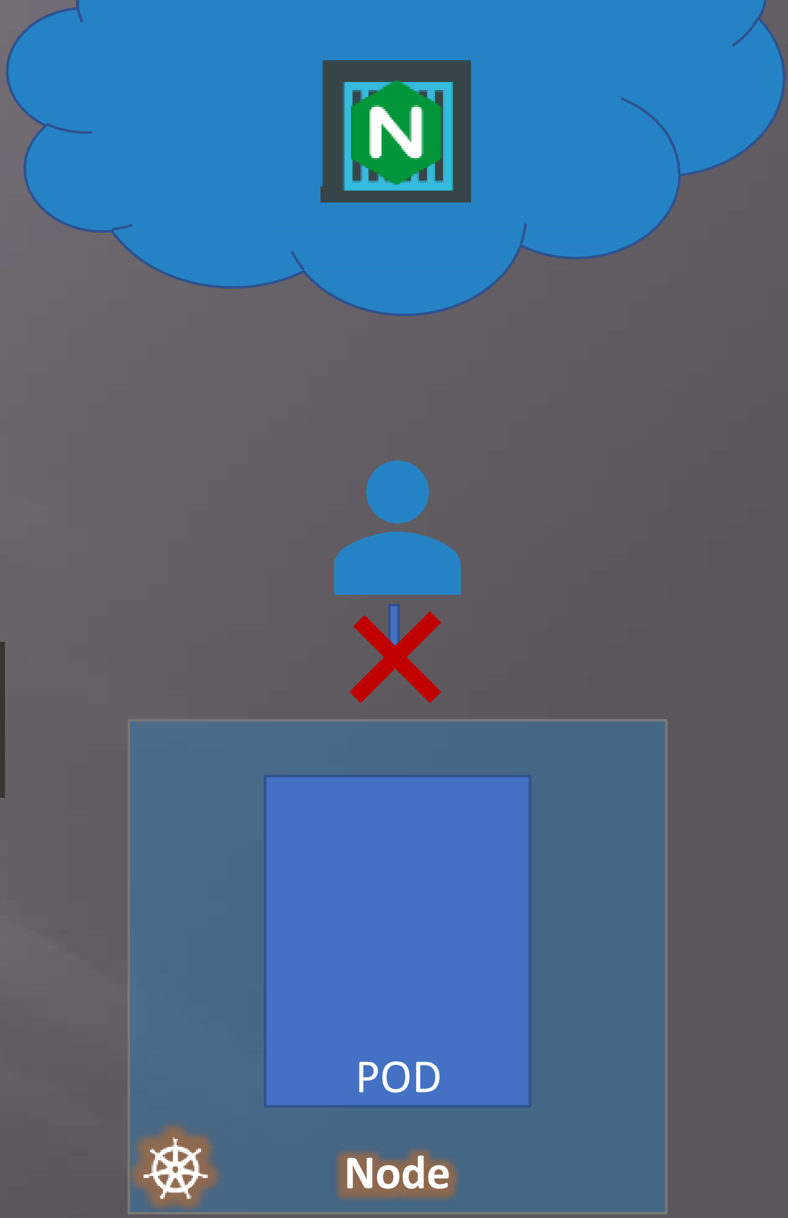
```
kubectl get pods
```



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

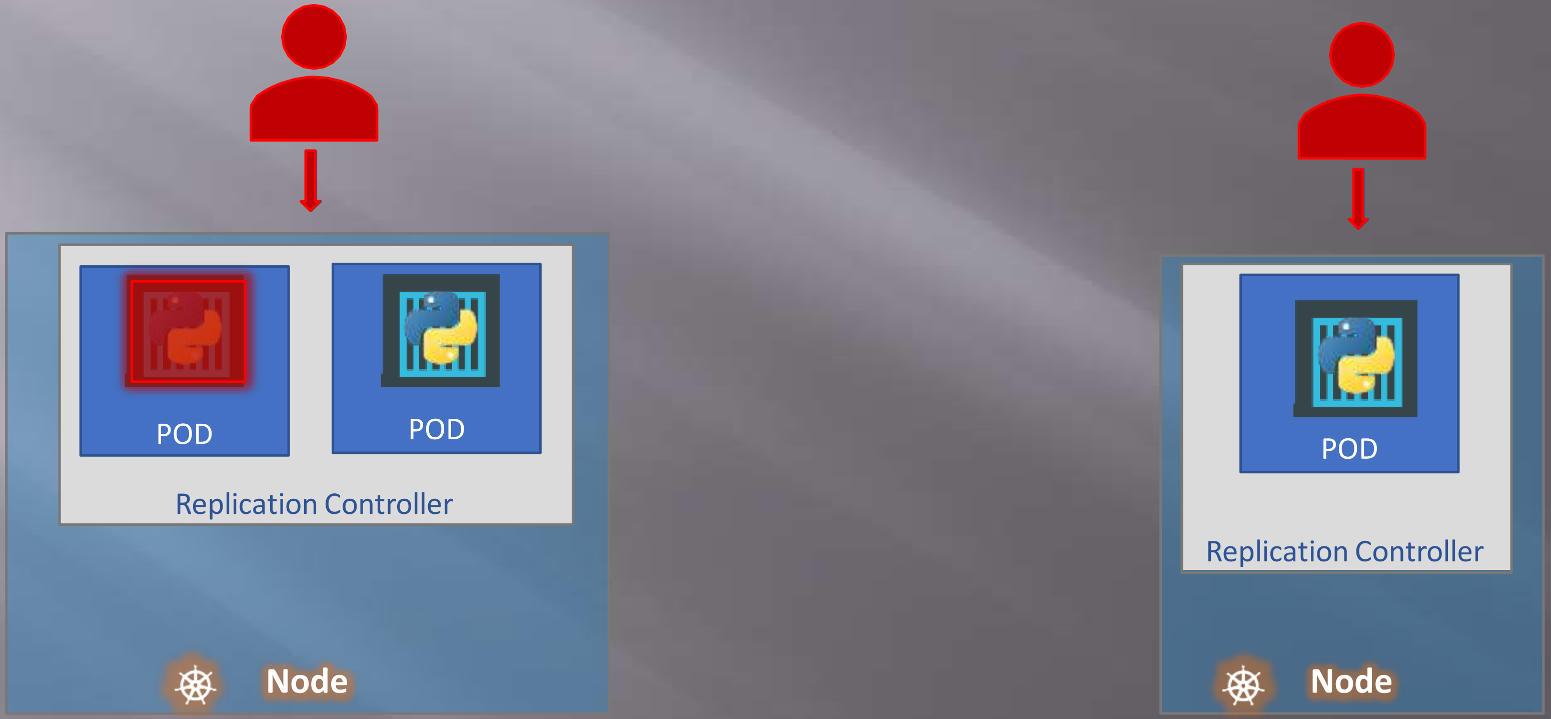


```
NAME     READY   STATUS    RESTARTS   AGE
nginx    1/1     Running   0          34s
```

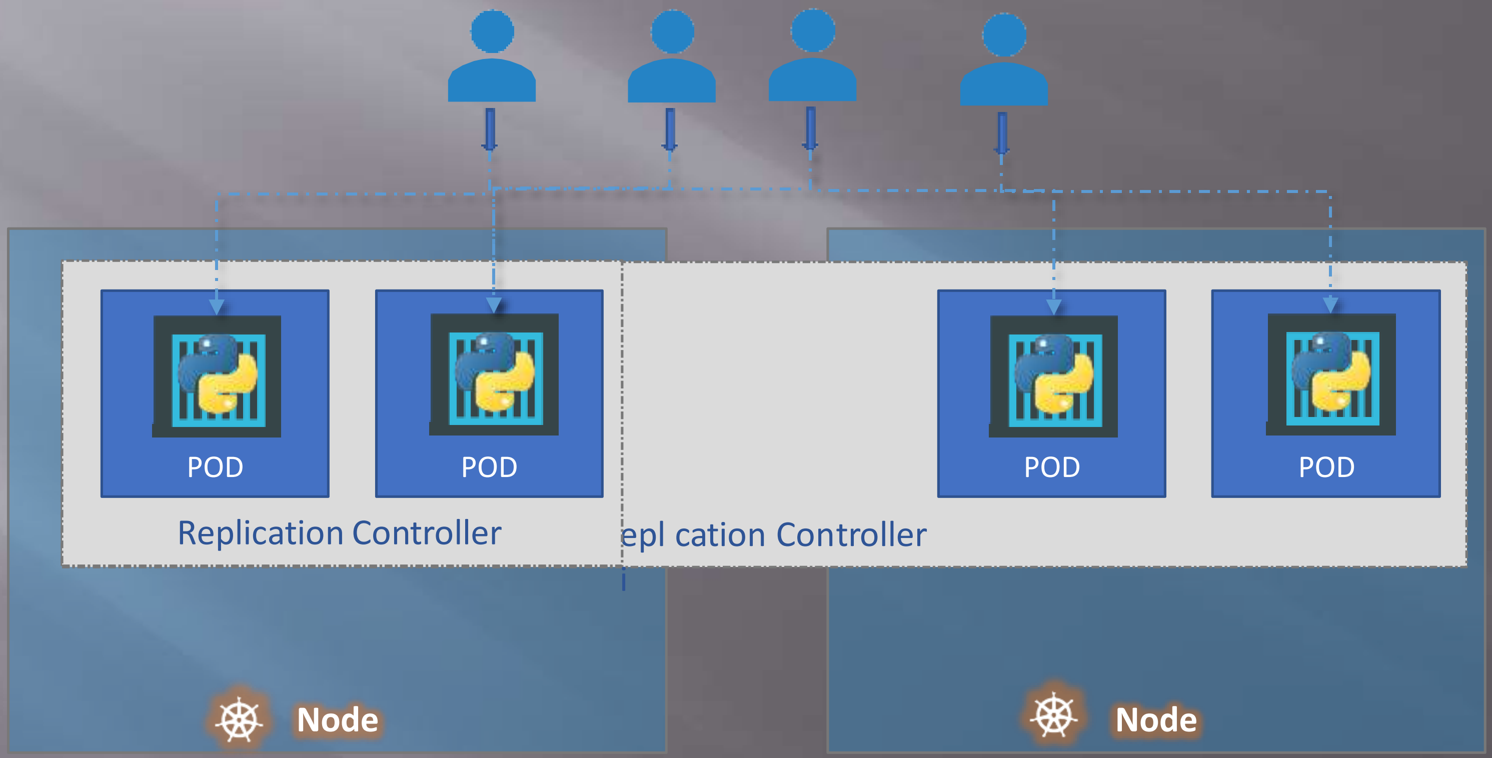


Replication Controller

High Availability



Load Balancing & Scaling



- Replication Controller

Replica Set

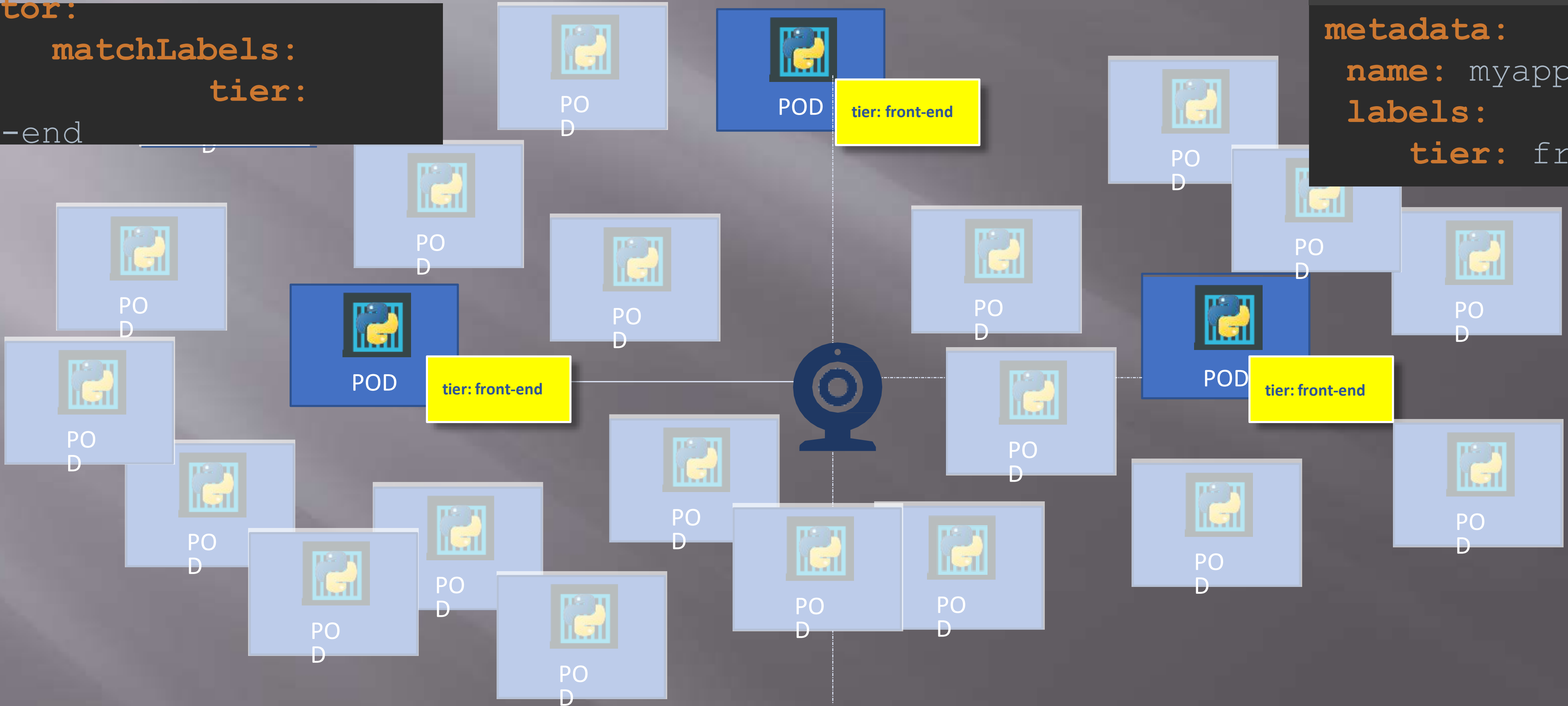
Labels and Selectors

replicaset-definition.yml

```
selector:
        matchLabels:
                tier:
front-end
```

pod-definition.yml

```
metadata:
  name: myapp-pod
  labels:
     tier: front-end
```



Deployment

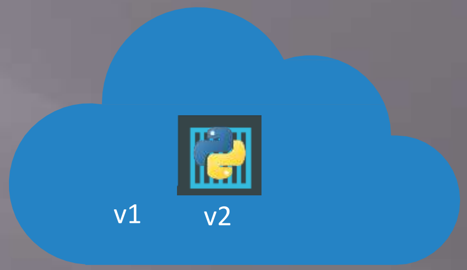
A Kubernetes deployment specifies the application’s life cycle, including the pods assigned to the app. It provides a way to communicate your desired state to Kubernetes deployments, and the controller works on changing the present state into your desired state.

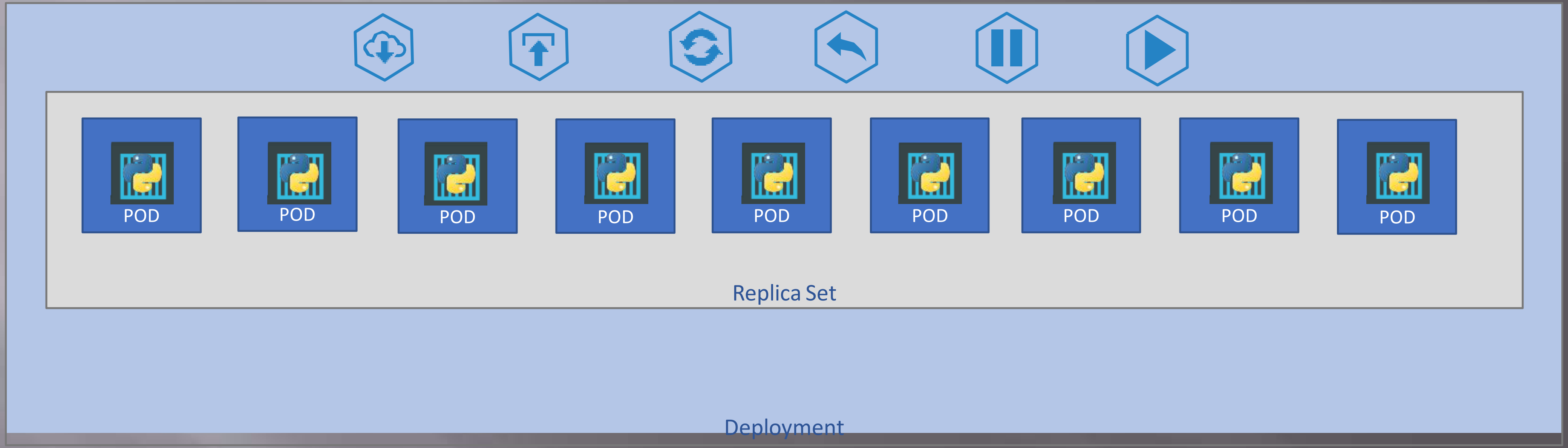
In simple terms, a Kubernetes deployment is a tool that manages the performance and specifies the desired behavior or traits of a pod.

Administrators and IT professionals use deployments to communicate what they want from an application. After this, Kubernetes takes all the necessary steps to create the desired state of the application.

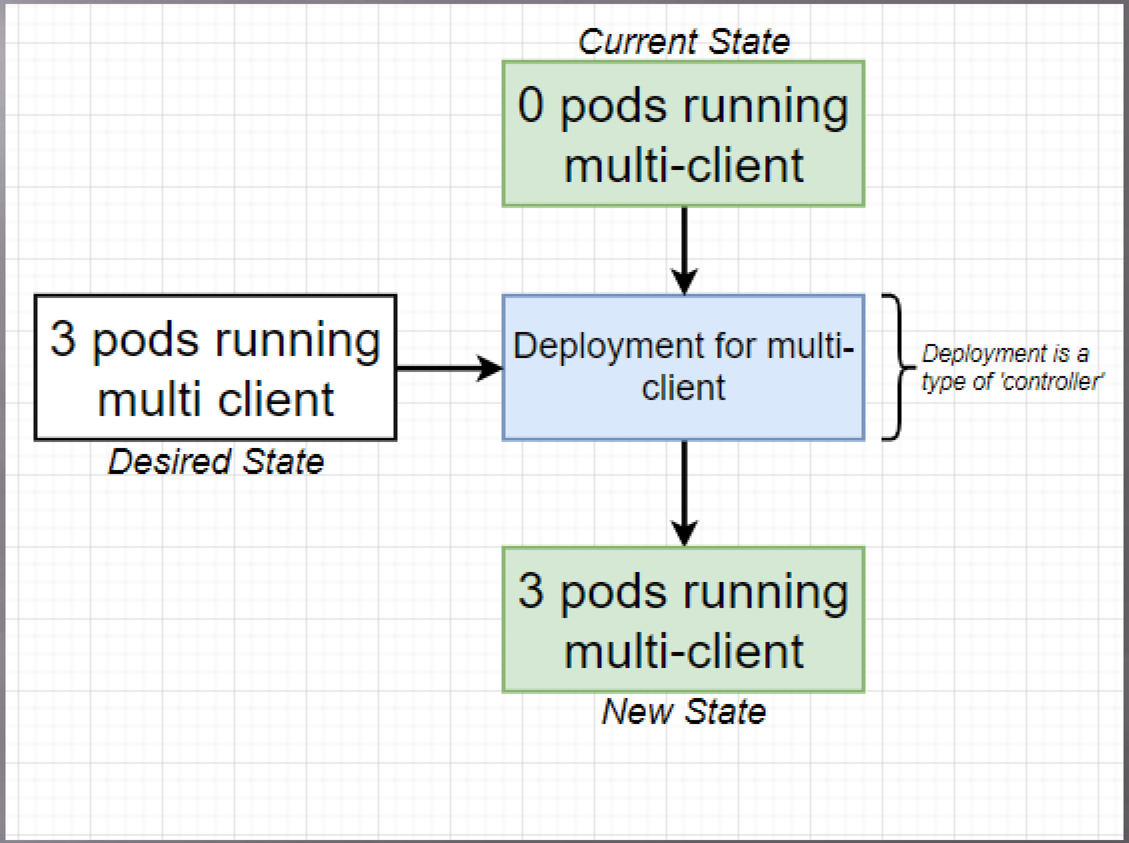
For example, Kubernetes deployments can be used to roll out a ReplicaSet to create pods and check their health to see if they are working optimally.

Deployment


v1
v2


POD
POD
POD
POD
POD
POD
POD
POD
POD
Replica Set
Deployment

Deployment


Current State
0 pods running multi-client
3 pods running multi client
Desired State
Deployment for multi-client
Deployment is a type of 'controller'
3 pods running multi-client
New State

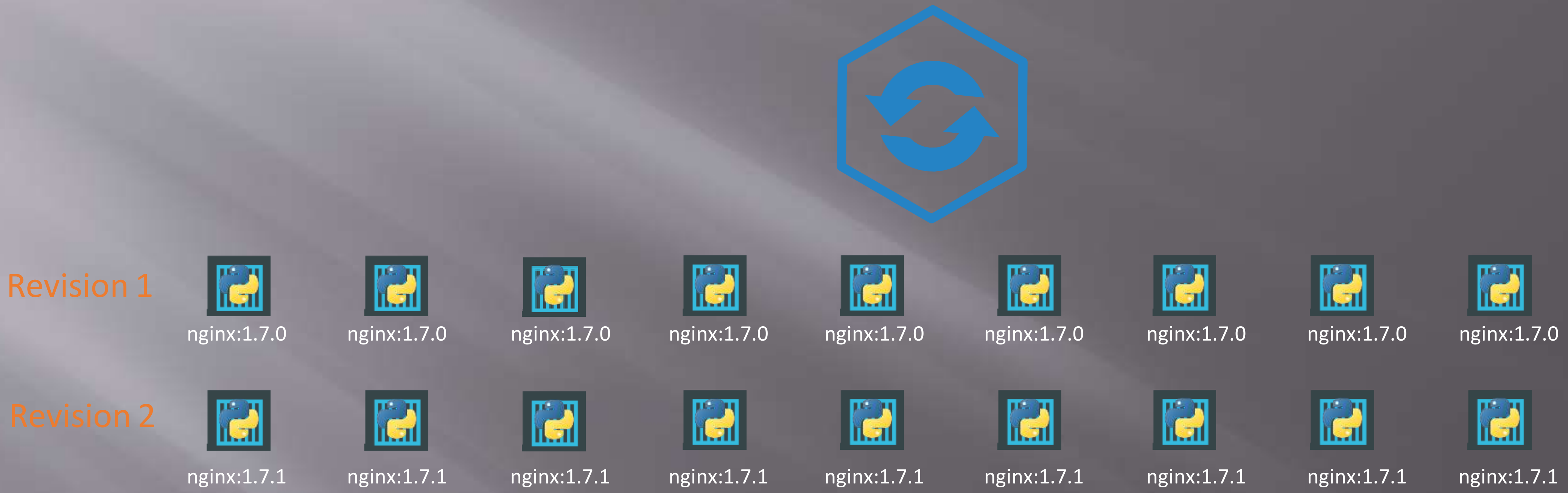
commands

```
> 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
```

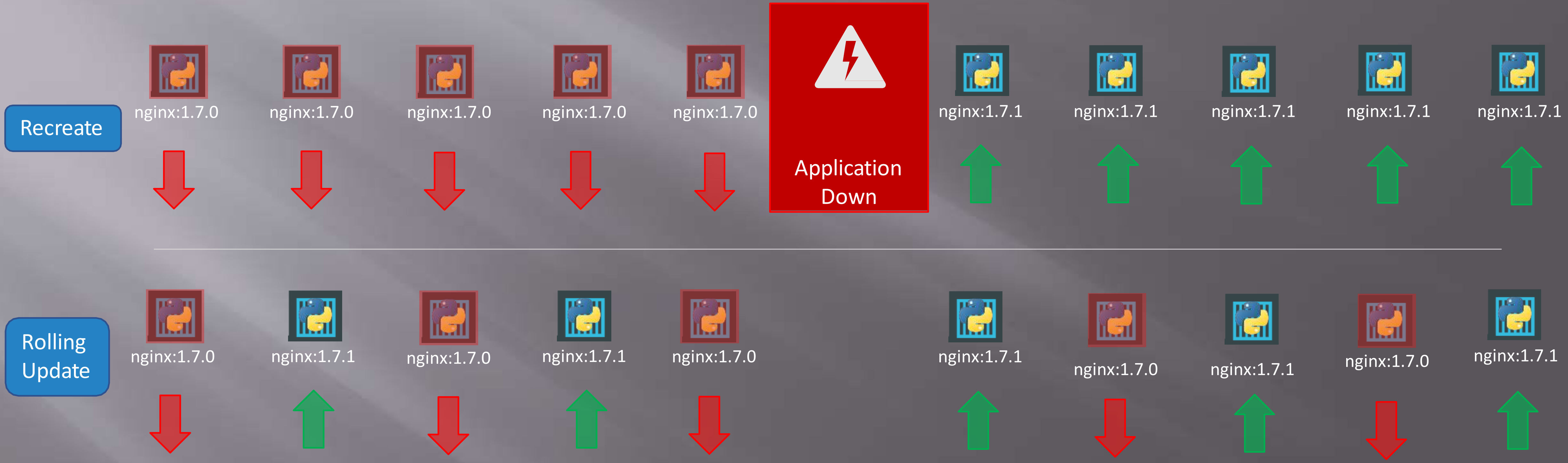
Deployment

Updates and Rollback

Rollout and Versioning



Deployment Strategy



Kubectl apply

```
> kubectl apply -f deployment-definition.yml
```

```
deployment "myapp-deployment" configured
```

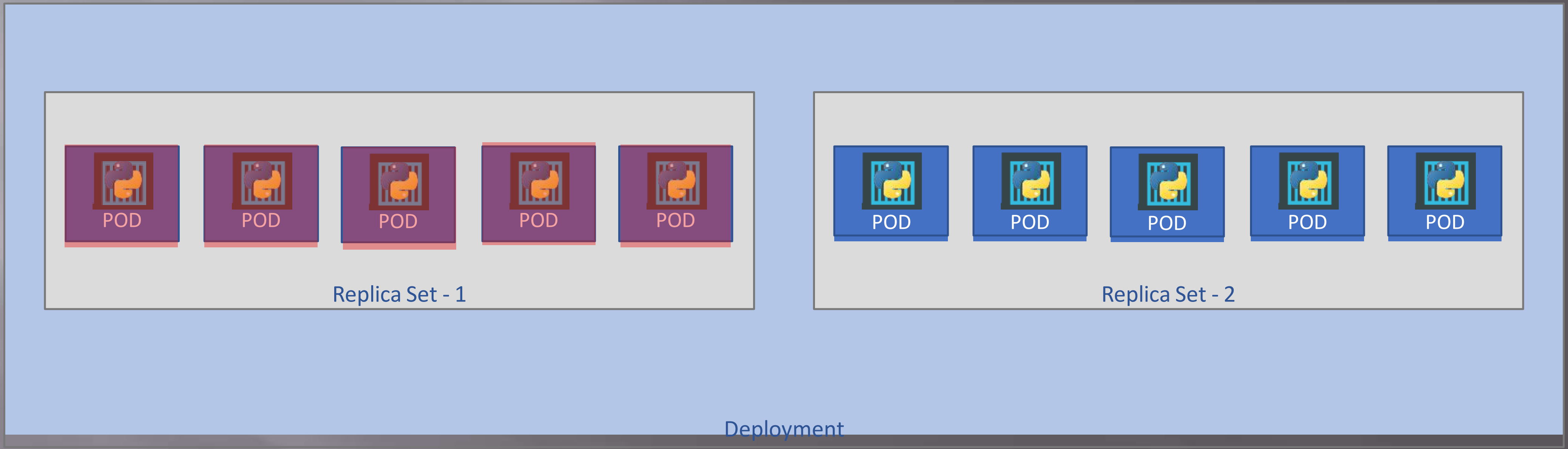
```
>  kubectl set image deployment/myapp-deployment \
                 nginx=nginx:1.9.1
```

```
deployment "myapp-deployment" image is updated
```

deployment-definition.yml

```
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
```

Upgrades

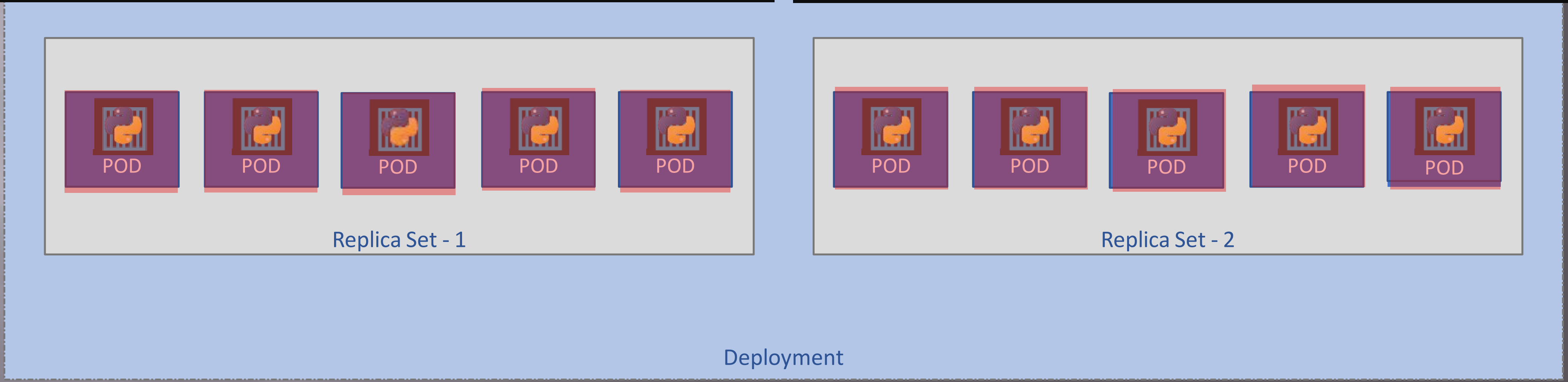


```
> kubectl get replicasets
NAME                        DESIRED    CURRENT    READY     AGE
myapp-deployment-67c749c58c 0          0          0         22m
myapp-deployment-7d57dbdb8d 5          5          5         20m
```

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
```



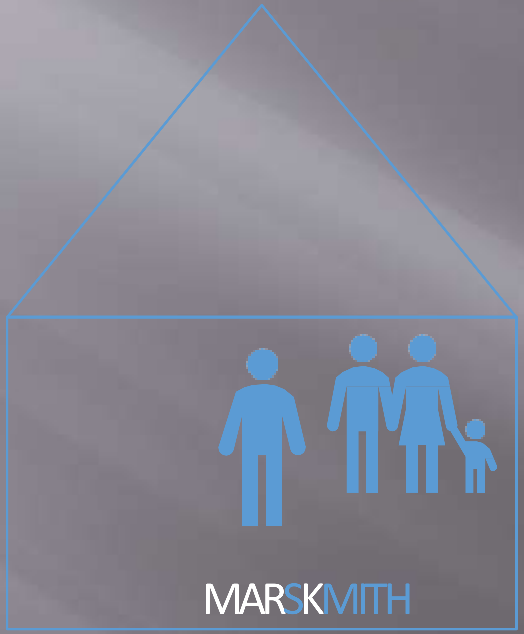
```
> kubectl rollout undo deployment/myapp-deployment
deployment "myapp-deployment" rolled back
```

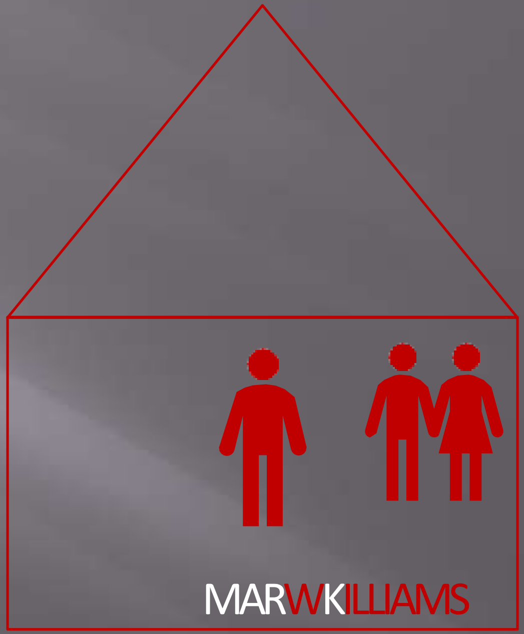
kubectl run

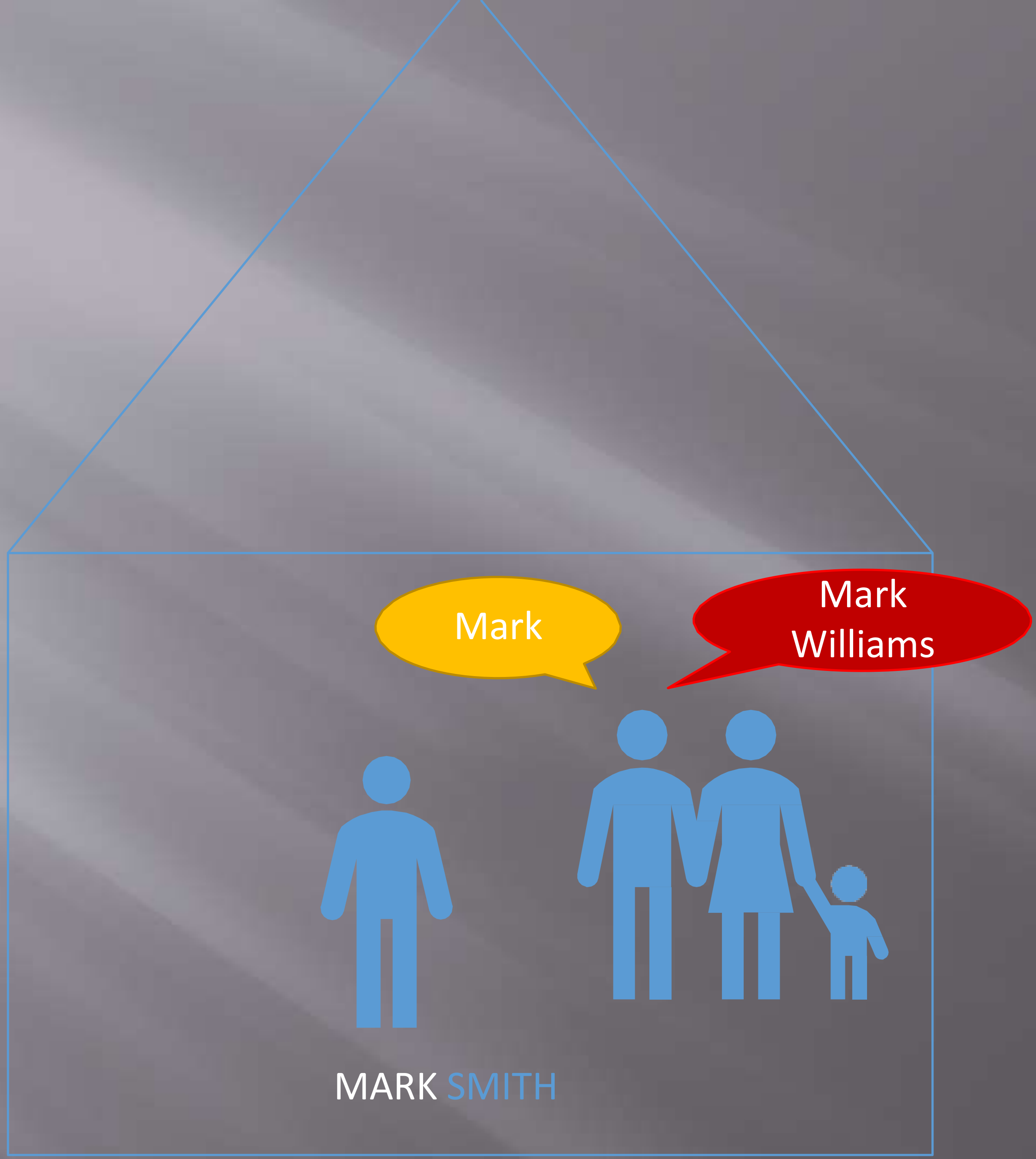
```
> kubectl run nginx --image=nginx
```

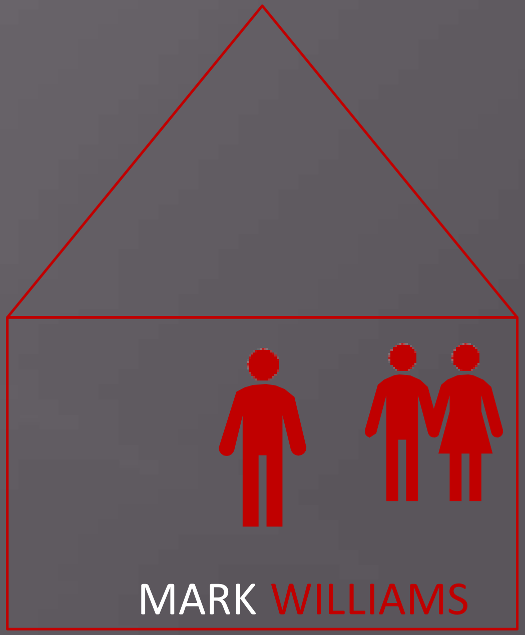
```
deployment "nginx" created
```

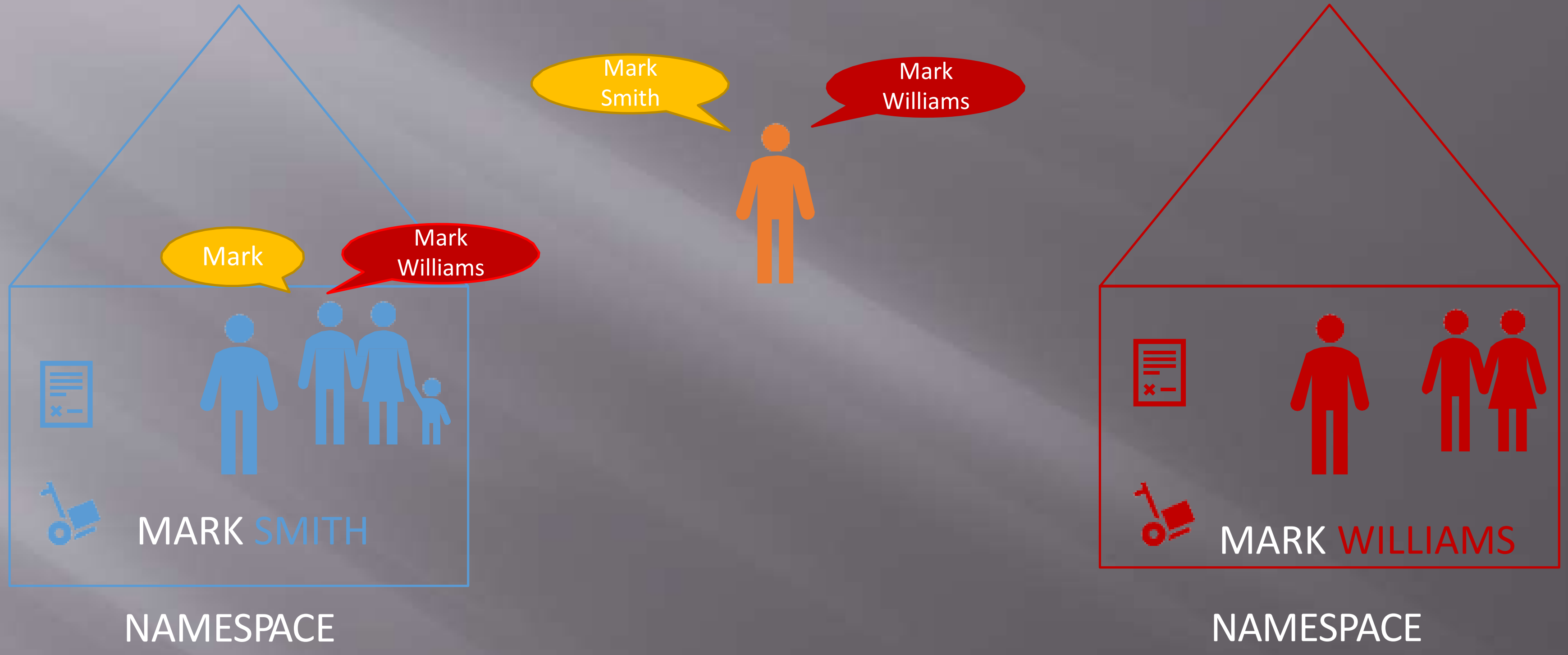
Namespaces

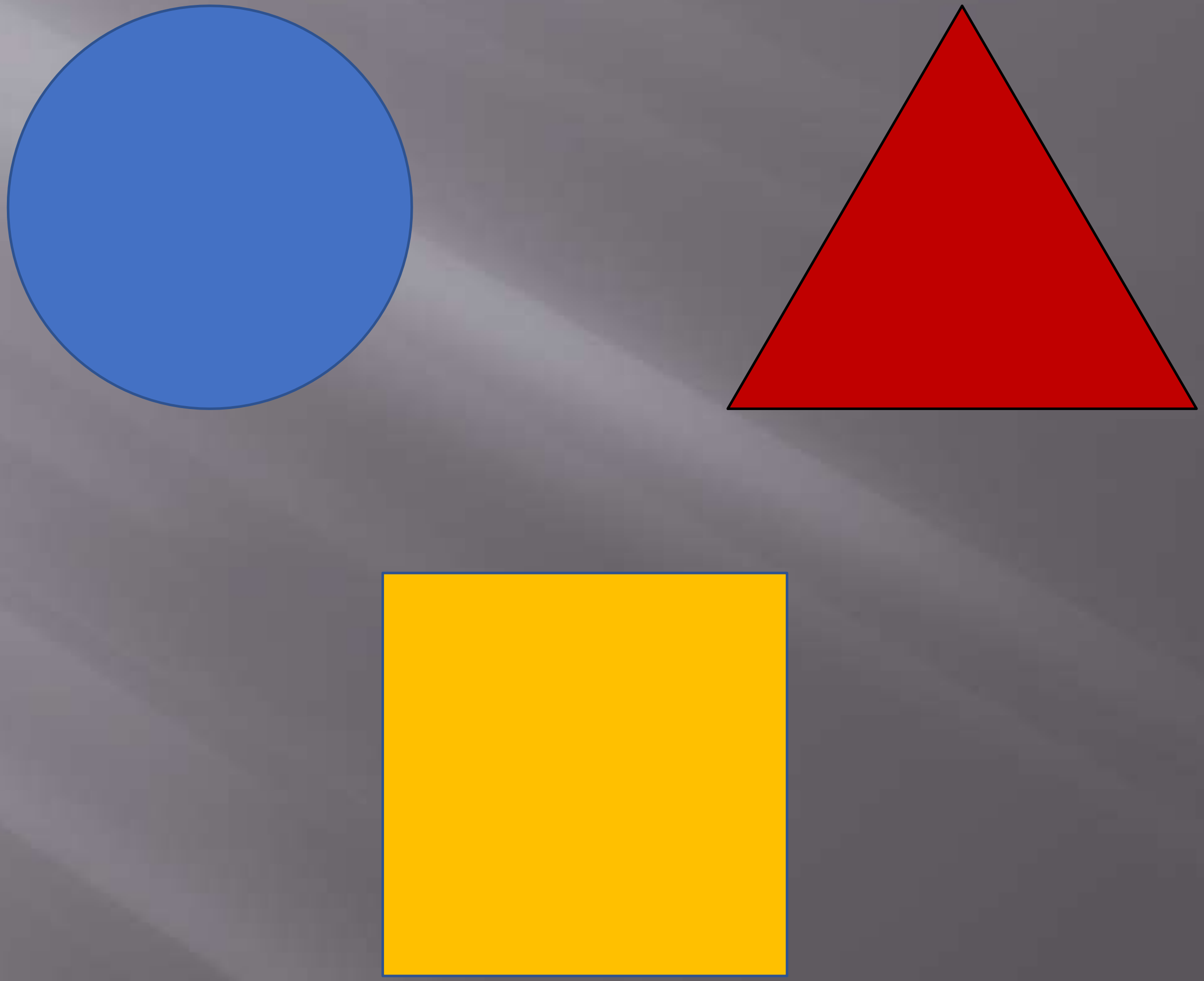

MARSKMITH

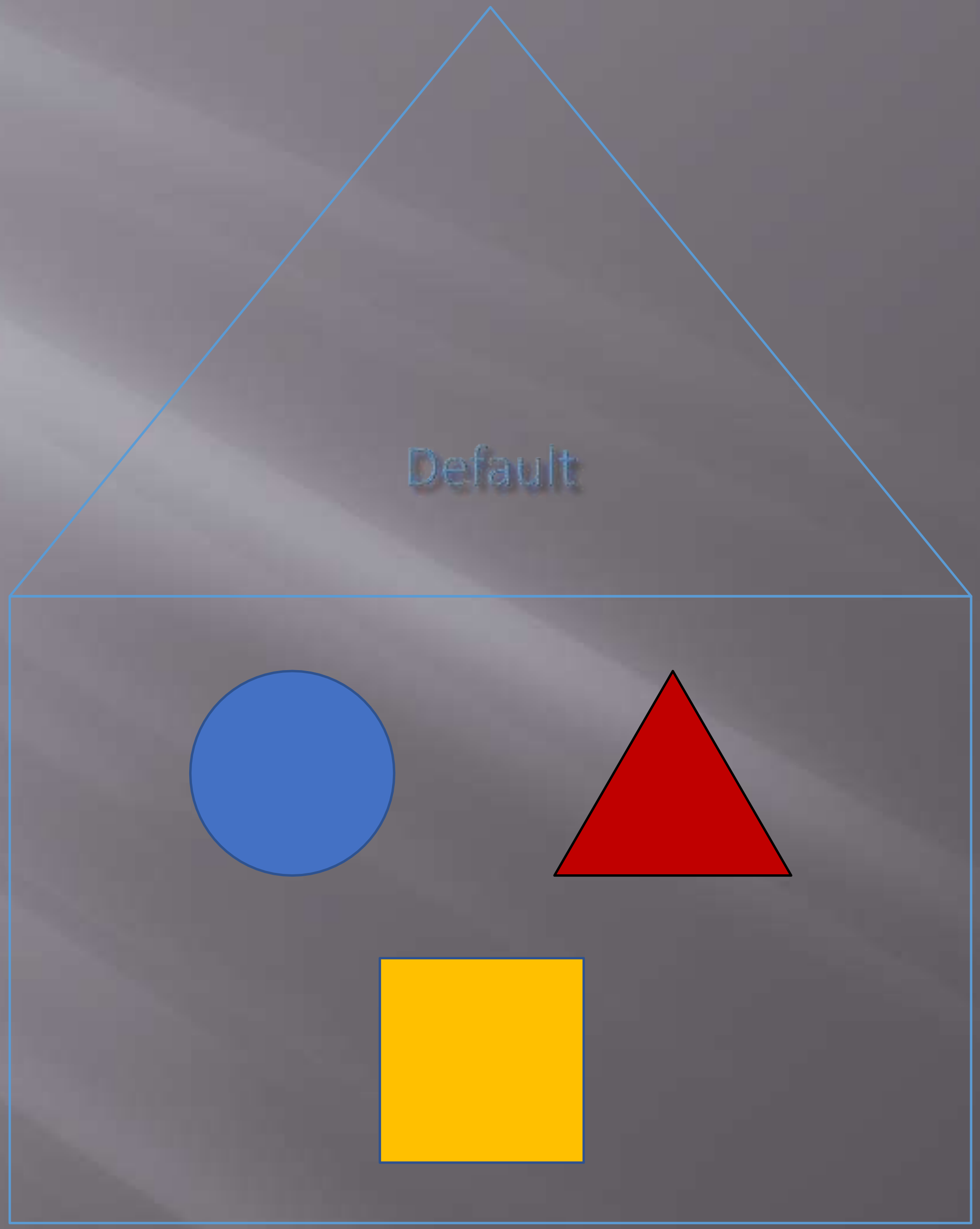

MARWKILLIAMS

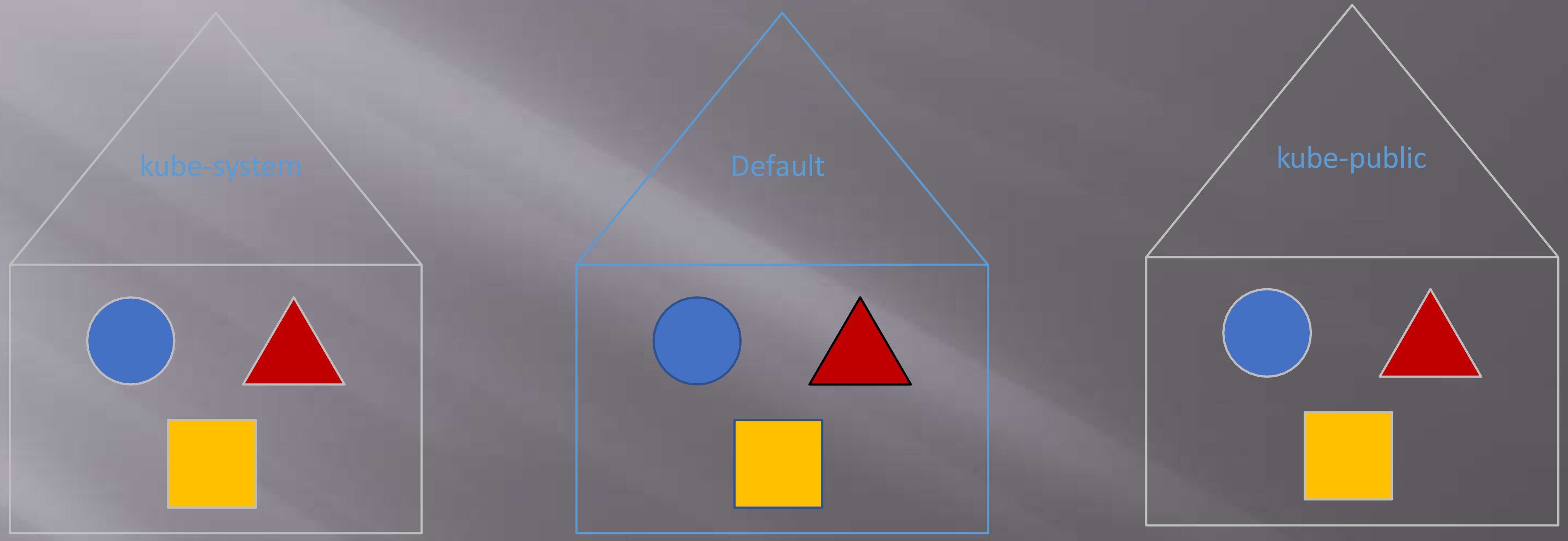

Mark
Mark Williams
MARK SMITH


MARK WILLIAMS

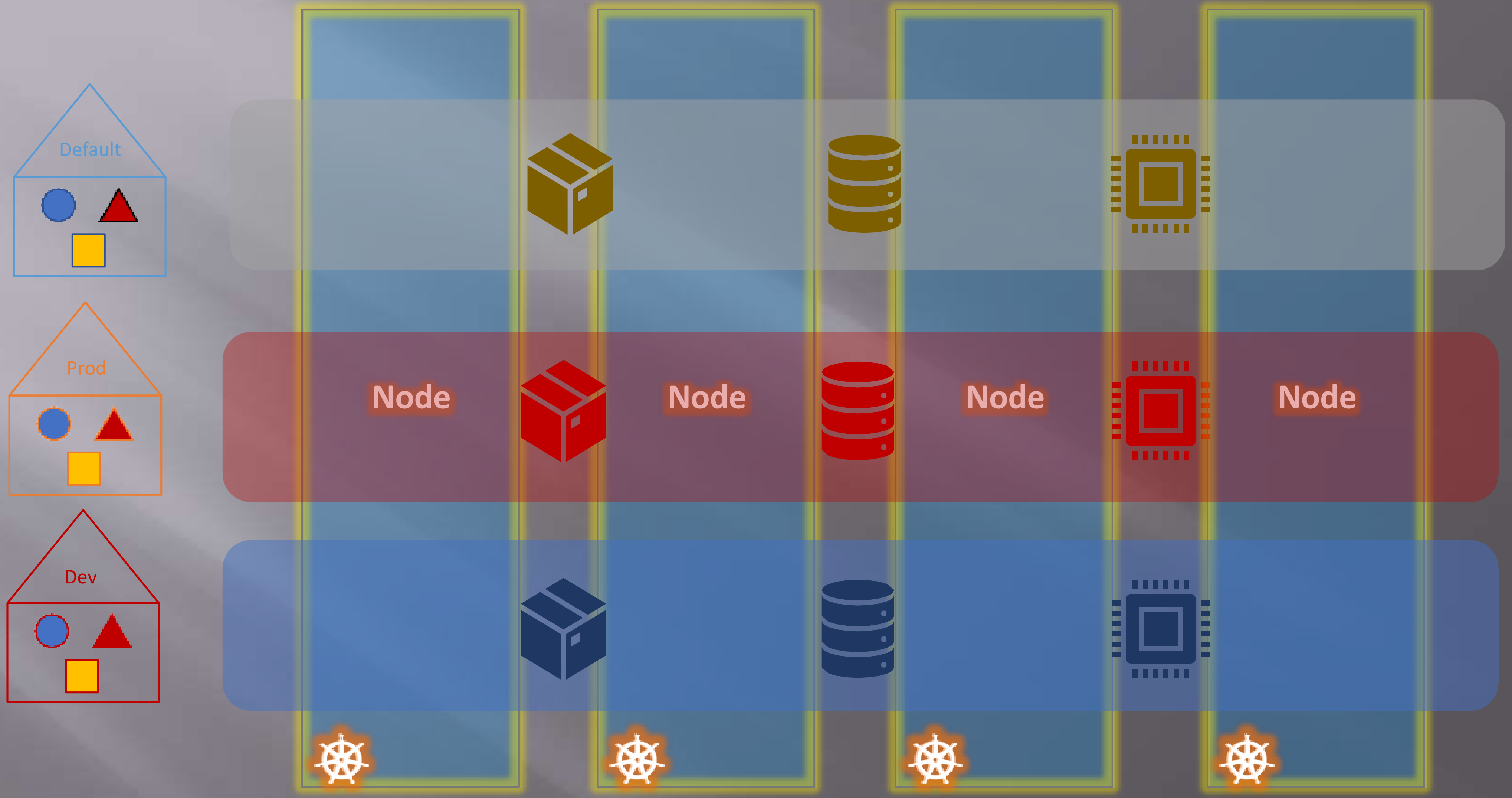

Mark
Mark Williams
MARK SMITH
NAMESPACE
Mark Smith
Mark Williams
MARK WILLIAMS
NAMESPACE



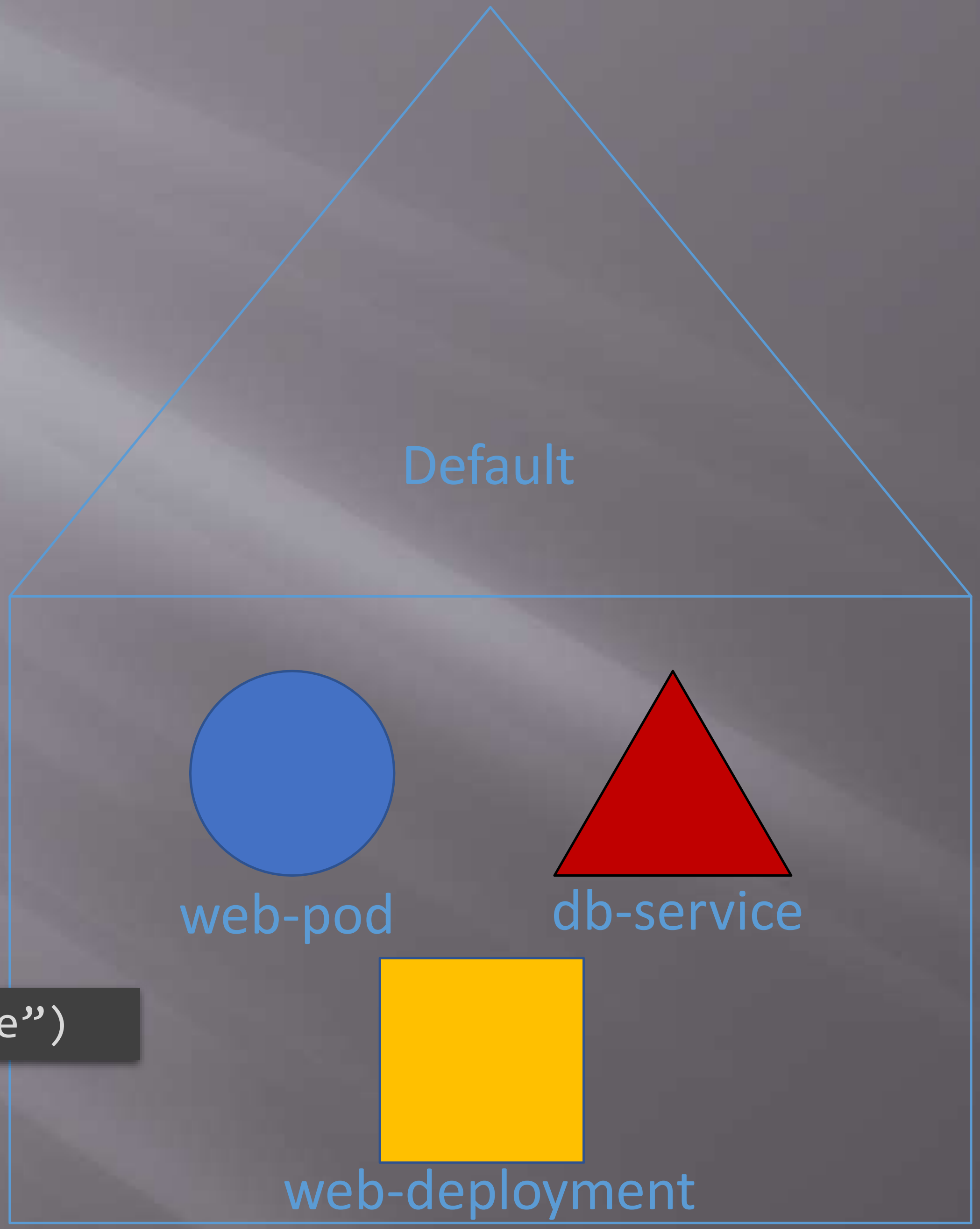

Default


kube-system
Default
kube-public

Namespace – Resource Limits

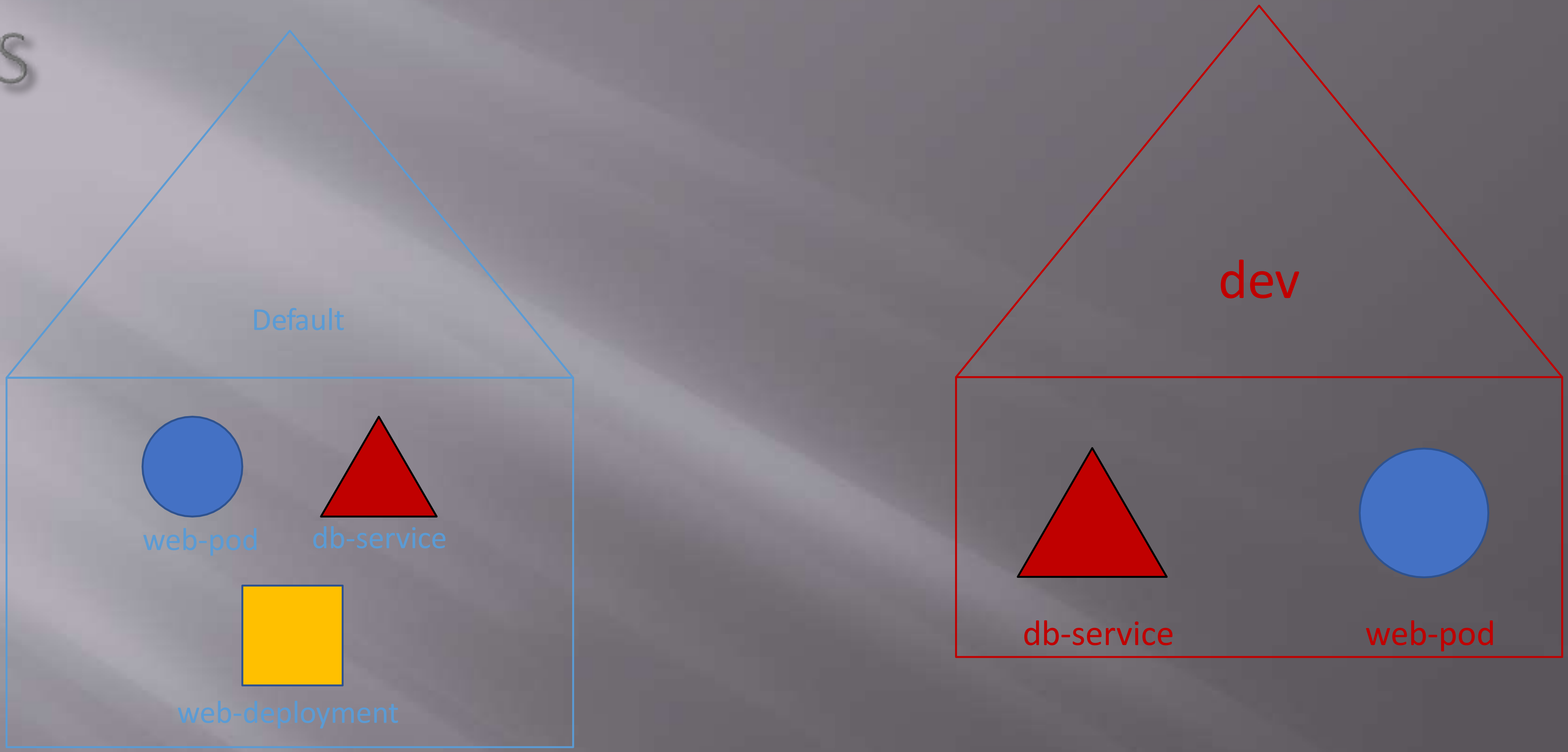


DNS



```
mysql.connect("db-service")
```

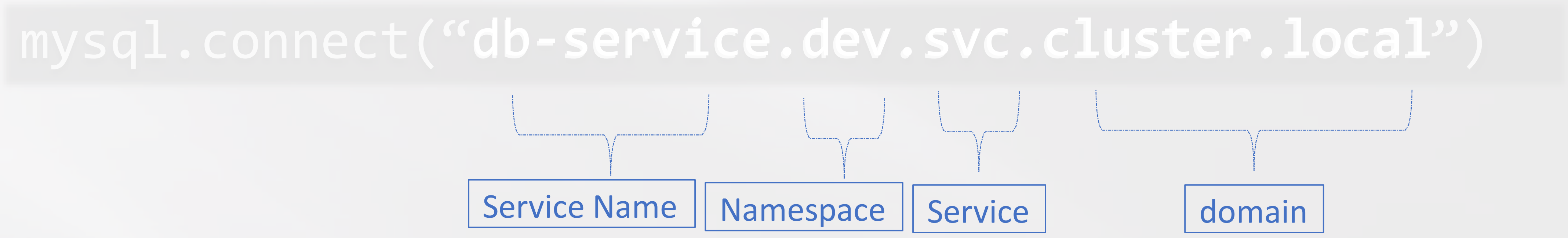
DNS



```
mysql.connect(“db-service”)
```

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

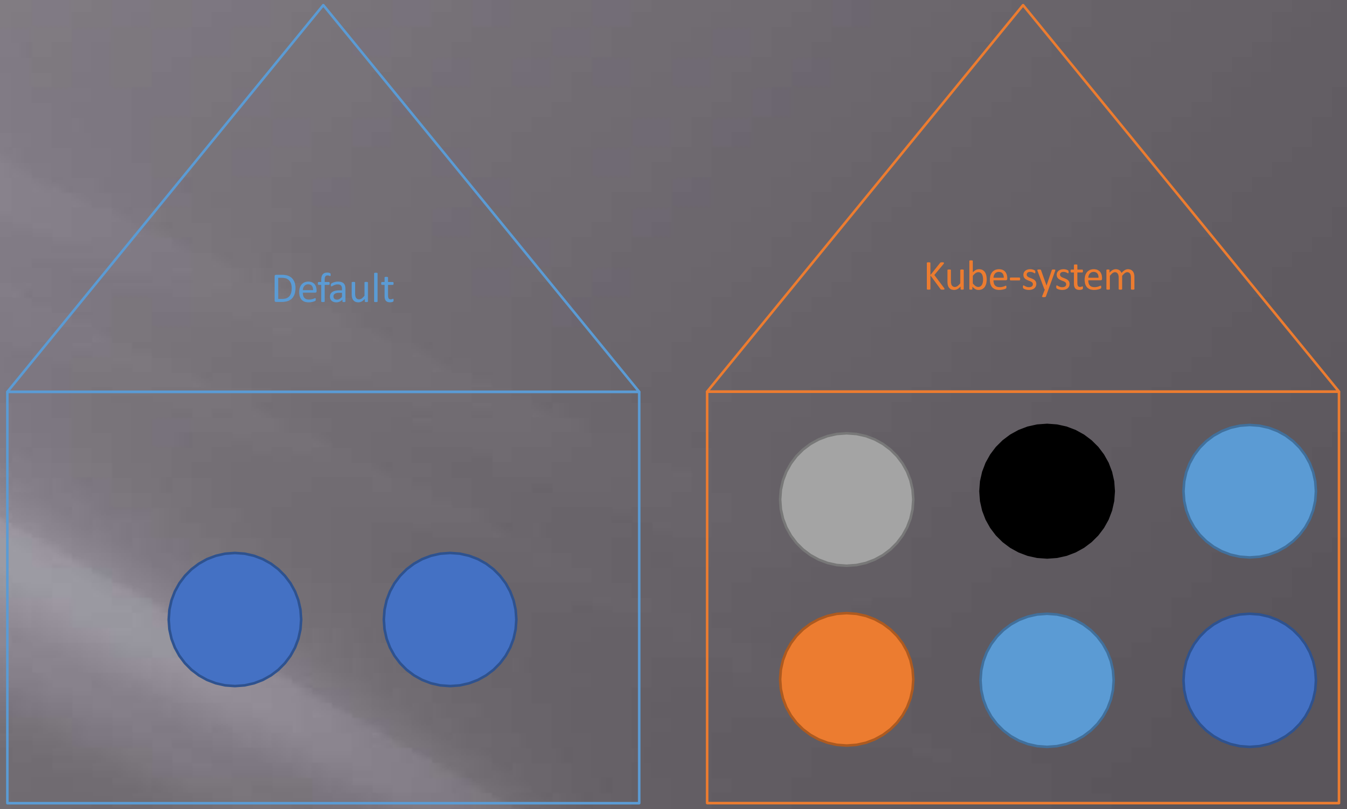
DNS



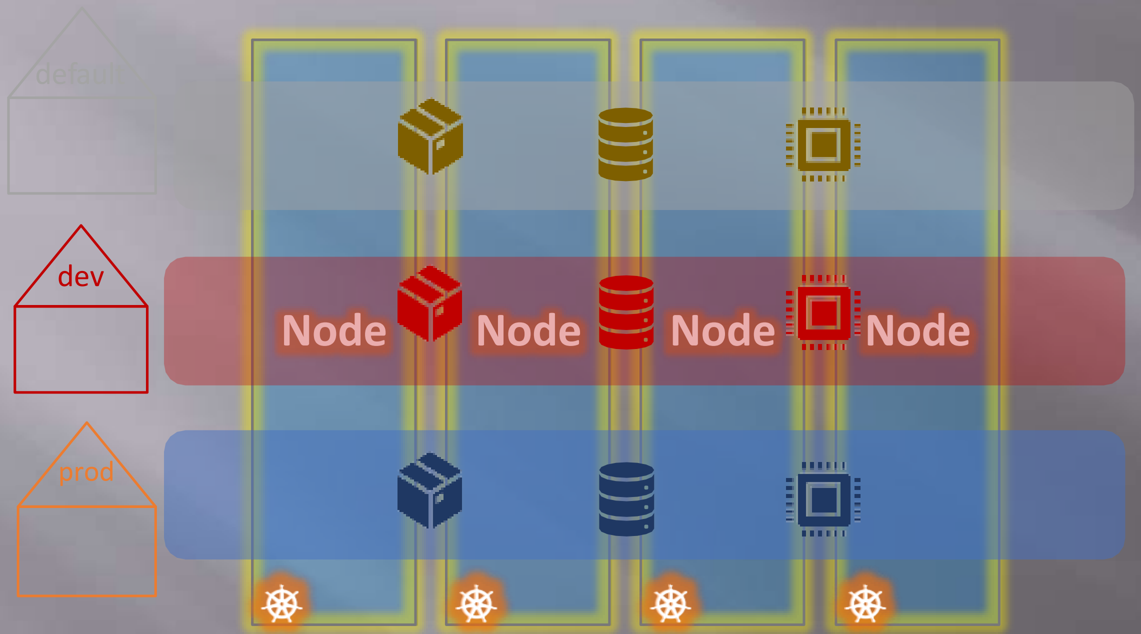
```
mysql.connect(“db-service.dev.svc.cluster.local”)
```

```
> 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
```



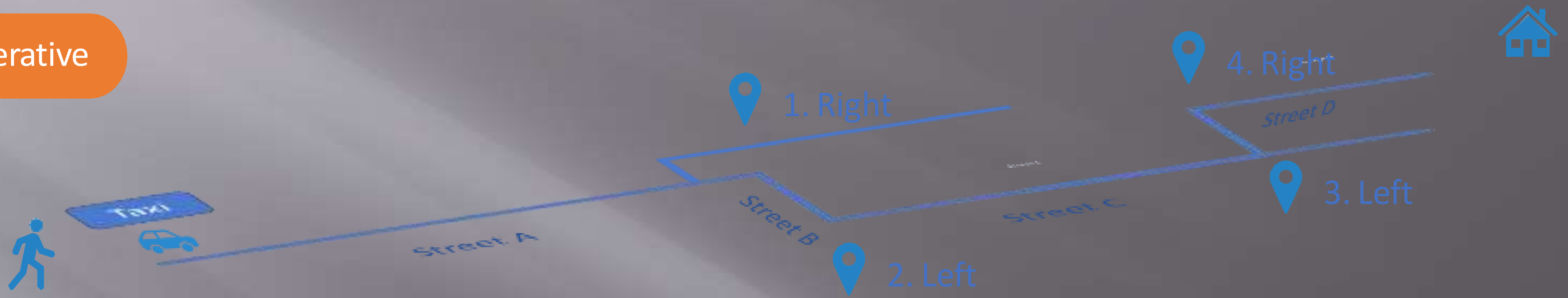
Resource Quota



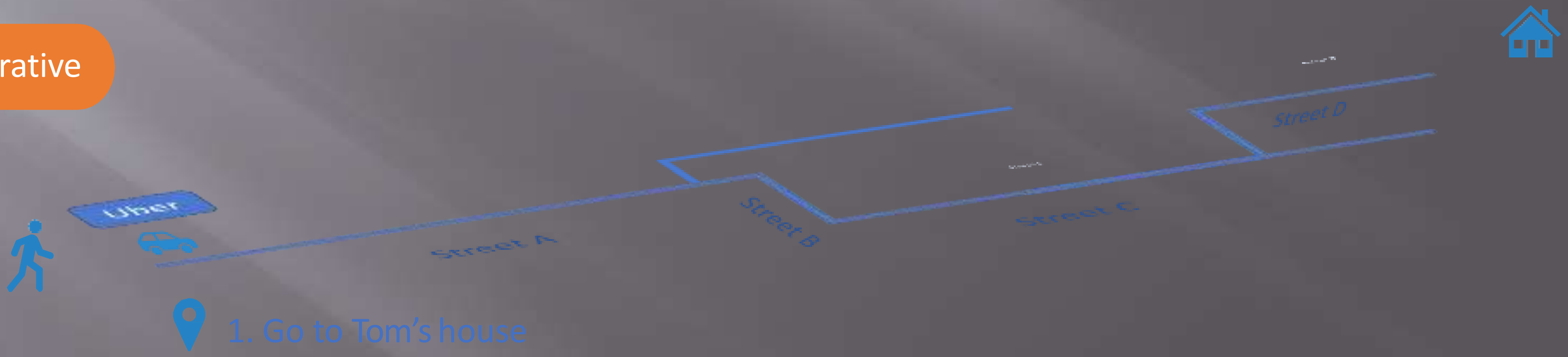
```
> kubectl create -f compute-quota.yaml
```

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
```

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

```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
```

nginx.yaml

```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

```yaml
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

```
> 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"
```

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

```
apiVersion: v1
kind: Pod

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

END