PaaS Environnement

Année Universitaire
2021-2022

Goals

The objective of this chapter is to :

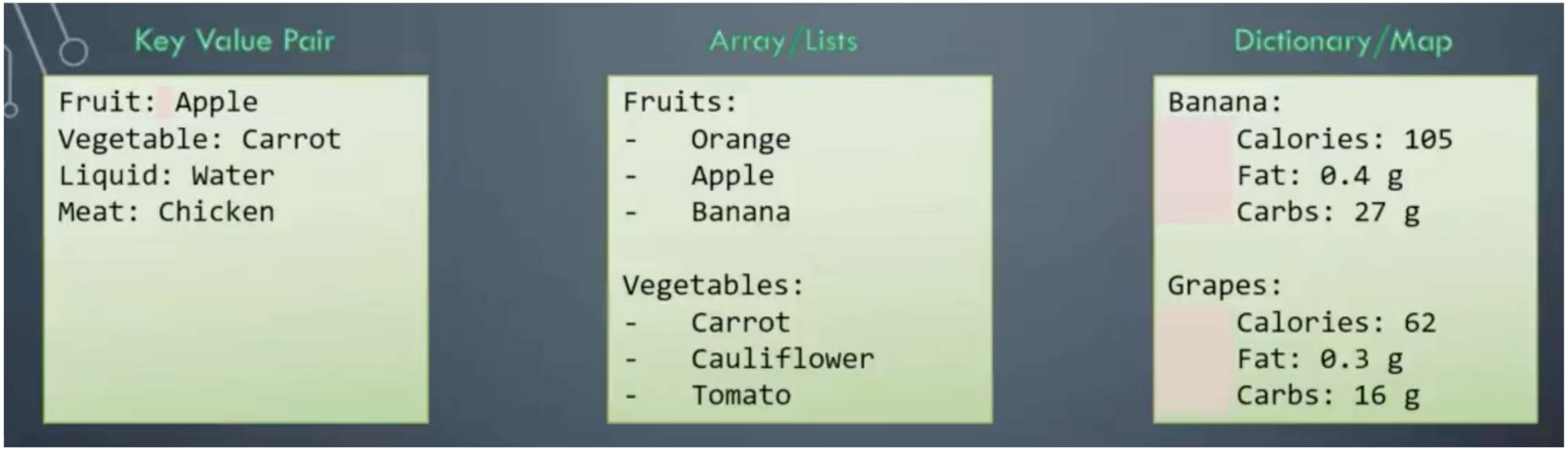
- Understand the basic concepts related to the administration of orchestration solution (Kubernetes).
- Practice some administration operation on Kubernetes cluster.

Outline

1. YAML : Reminder
2. Namespaces
3. Setup Pods with YAML
4. Replica Set
5. Labels & Selectors
6. Deployment
7. Updates & Rollbacks

YAML : Reminder

- **YAML** (a recursive acronym for "YAML Ain't Markup Language") is a human-readable data-serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted.



Key Value Pair

```
Fruit: Apple
Vegetable: Carrot
Liquid: Water
Meat: Chicken
```

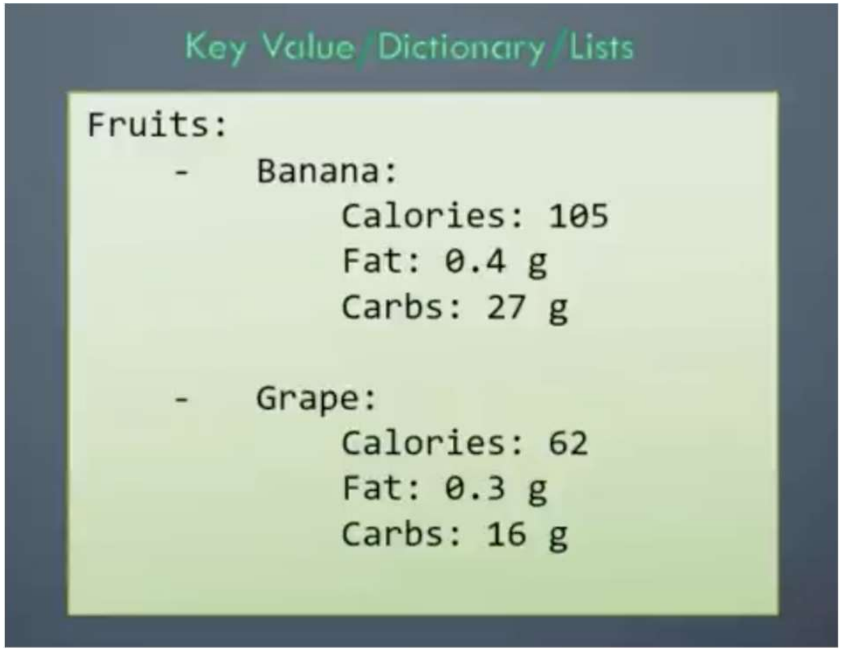
Array/Lists

```
Fruits:
-    Orange
-    Apple
-    Banana

Vegetables:
-    Carrot
-    Cauliflower
-    Tomato
```

Dictionary/Map

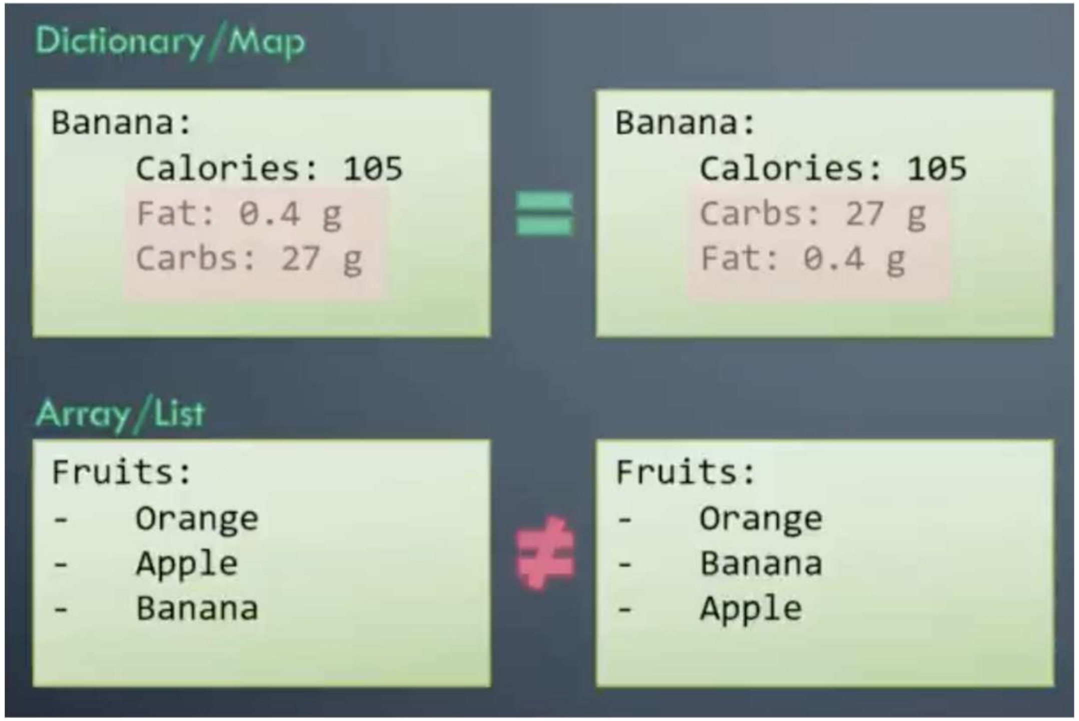
```
Banana:
    Calories: 105
    Fat: 0.4 g
    Carbs: 27 g

Grapes:
    Calories: 62
    Fat: 0.3 g
    Carbs: 16 g
```

YAML : Reminder



Key Value/Dictionary/Lists

```
Fruits:
    -    Banana:
             Calories: 105
             Fat: 0.4 g
             Carbs: 27 g

    -    Grape:
             Calories: 62
             Fat: 0.3 g
             Carbs: 16 g
```



Dictionary/Map

```
Banana:
    Calories: 105
    Fat: 0.4 g
    Carbs: 27 g
```

=

```
Banana:
    Calories: 105
    Carbs: 27 g
    Fat: 0.4 g
```

Array/List

```
Fruits:
-    Orange
-    Apple
-    Banana
```

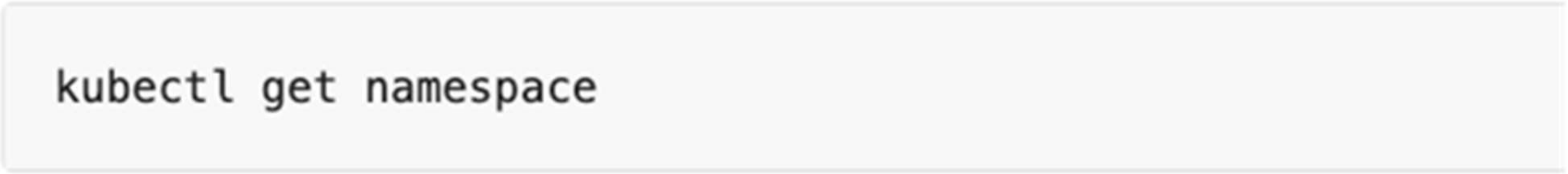
≠

```
Fruits:
-    Orange
-    Banana
-    Apple
```

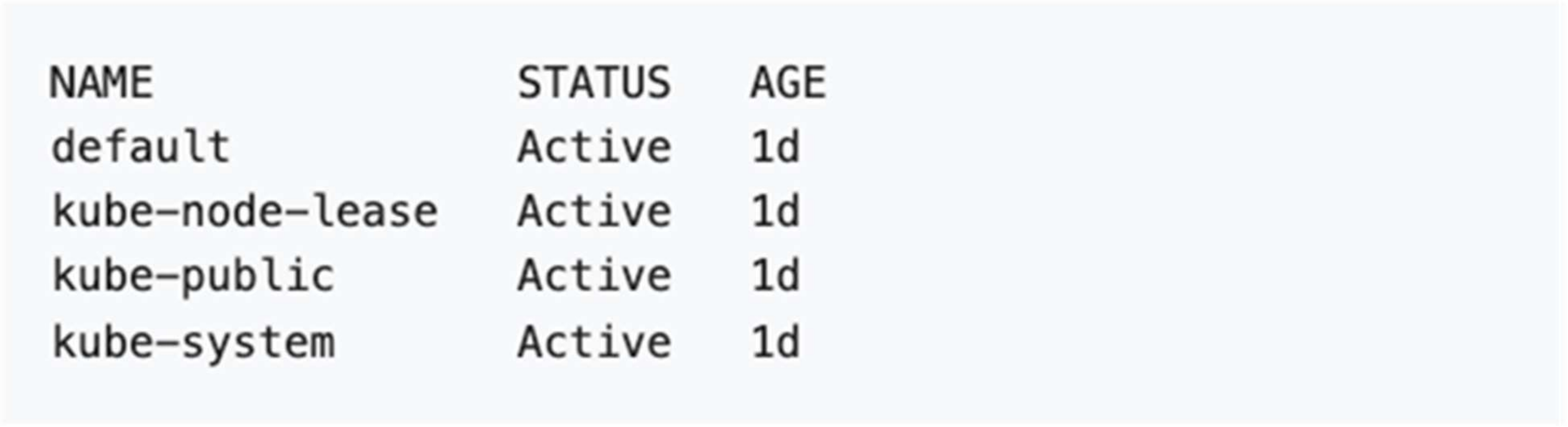
Namespaces

- Namespaces provides a mechanism for isolating groups of resources within a single cluster.
- *Names of resources need to be unique within a namespace, but not across namespaces.*
- Namespaces are intended for use in environments with many users spread across multiple teams, or projects.
- Namespaces are a way to divide cluster resources between multiple users.
- ❖ Les Namespaces fournissent un mécanisme pour isoler des groupes de ressources au sein d'un seul cluster.
- ❖ Les noms des ressources doivent être uniques au sein d'un espace de noms, mais pas à travers les espaces de noms.
- ❖ Les Namespaces sont destinés à être utilisés dans des environnements avec de nombreux utilisateurs répartis entre plusieurs équipes ou projets.
- ❖ Les Namespaces sont un moyen de diviser les ressources du cluster entre plusieurs utilisateurs.

Kuberntes Namespaces



```
kubectl get namespace
```



```
NAME              STATUS   AGE
default           Active   1d
kube-node-lease   Active   1d
kube-public       Active   1d
kube-system       Active   1d
```

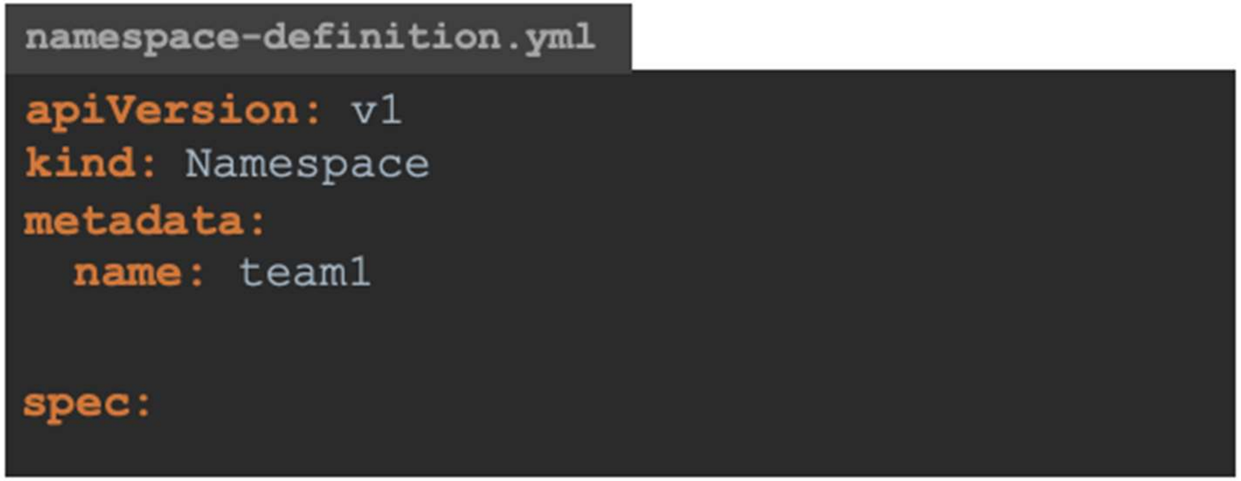
Namespaces

- ***default*** : The default namespace for objects with no other namespace
- ***kube-system*** : The namespace for objects created by the Kubernetes system
- ***kube-public*** : This namespace is created automatically and is readable by all users (including those not authenticated). This namespace is mostly reserved for cluster usage, in case that some resources should be visible and readable publicly throughout the whole cluster.
- ***kube-node-lease*** : This namespace holds Lease objects associated with each node. Node leases allow the kubelet to send heartbeats so that the control plane can detect node failure.

- ❖ default : L'espace de noms par défaut pour les objets sans autre espace de noms.
- ❖ kube-system : L'espace de noms pour les objets créés par le système Kubernetes.
- ❖ kube-public : Cet espace de noms est créé automatiquement et est lisible par tous les utilisateurs (y compris ceux qui ne sont pas authentifiés). Cet espace de noms est principalement réservé à l'utilisation du cluster, au cas où certaines ressources devraient être visibles et lisibles publiquement dans tout le cluster.
- ❖ kube-node-lease : Cet espace de noms contient les objets Lease associés à chaque nœud. Les baux de nœuds permettent au kubelet d'envoyer des battements de cœur afin que le plan de contrôle puisse détecter une défaillance de nœud.

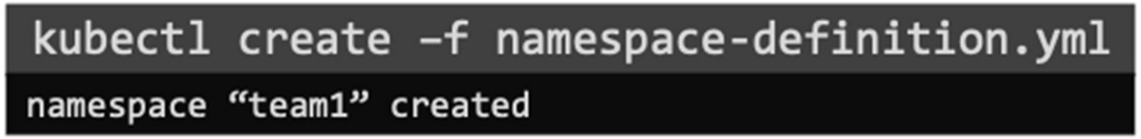
Create namespaces

❖ **First option**



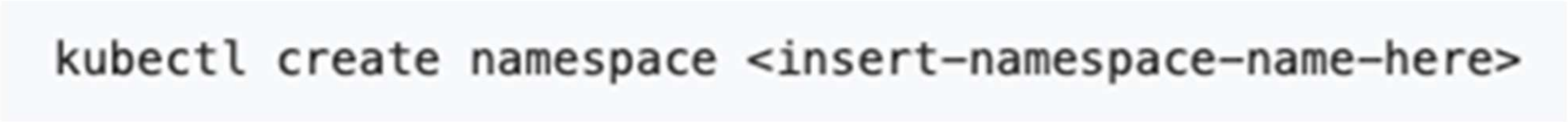
```
namespace-definition.yml
```

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: team1


spec:
```



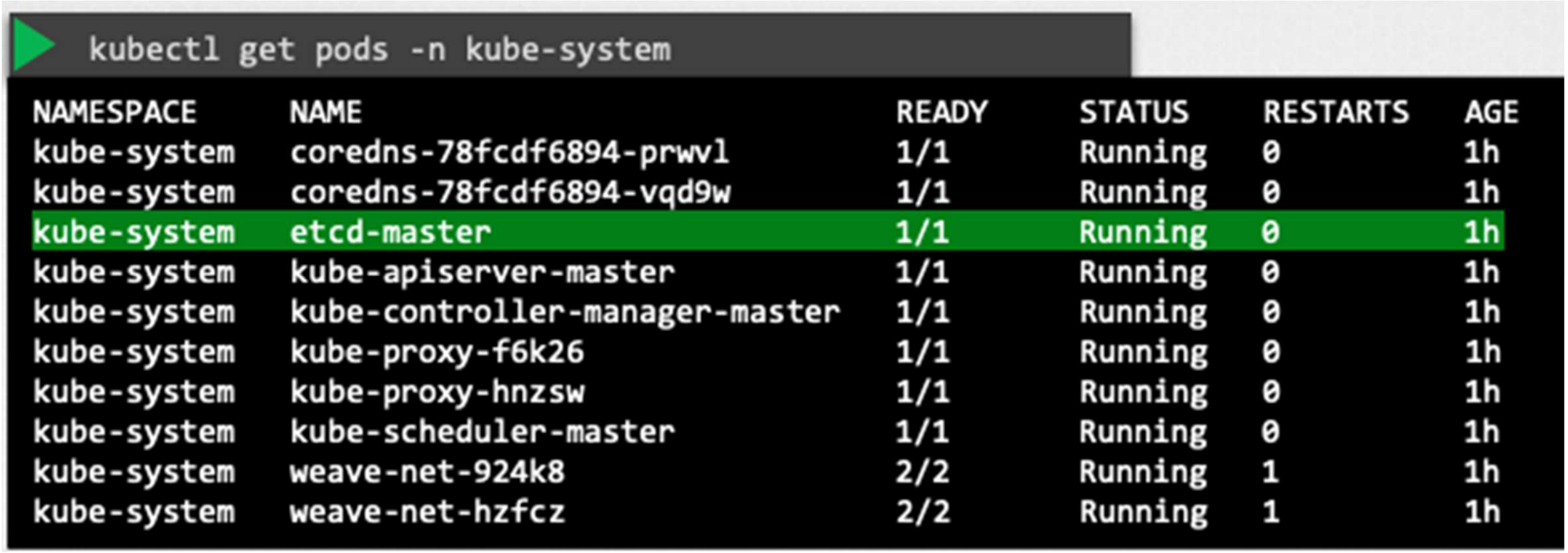
```
kubectl create -f namespace-definition.yml
namespace "team1" created
```

❖ **Second option**



```
kubectl create namespace <insert-namespace-name-here>
```

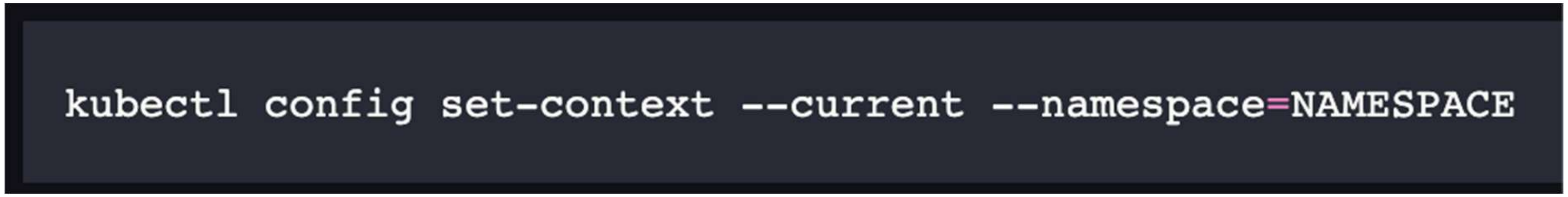
View resources in namespaces



```
kubectl get pods -n kube-system
```

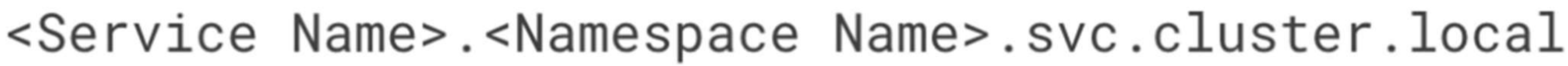
```
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
kube-system   coredns-78fcdf6894-prwvl          1/1     Running   0          1h
kube-system   coredns-78fcdf6894-vqd9w          1/1     Running   0          1h
kube-system   etcd-master                       1/1     Running   0          1h
kube-system   kube-apiserver-master             1/1     Running   0          1h
kube-system   kube-controller-manager-master    1/1     Running   0          1h
kube-system   kube-proxy-f6k26                  1/1     Running   0          1h
kube-system   kube-proxy-hnzsw                  1/1     Running   0          1h
kube-system   kube-scheduler-master             1/1     Running   0          1h
kube-system   weave-net-924k8                   2/2     Running   1          1h
kube-system   weave-net-hzfcz                   2/2     Running   1          1h
```

Setting Default Namespace

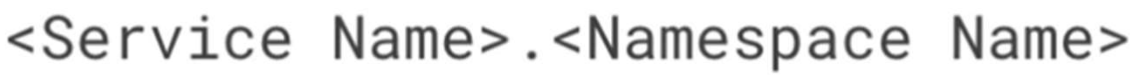


```
kubectl config set-context --current --namespace=NAMESPACE
```

Communication across namespaces



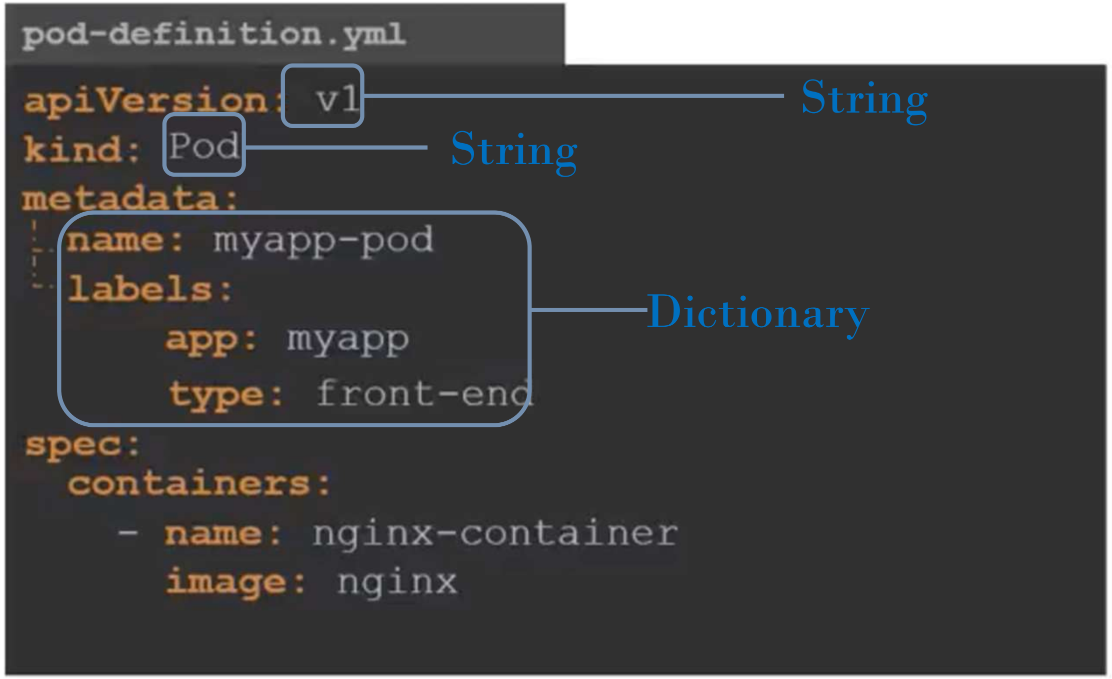
```
<Service Name>.<Namespace Name>.svc.cluster.local
```



```
<Service Name>.<Namespace Name>
```

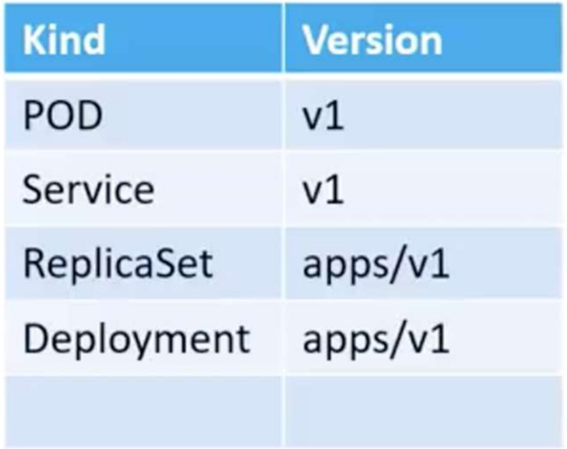
Setup Pods with YAML

- Kubernetes uses YAML files as inputs for creating Kubernetes object like pod, services, deployments, etc.

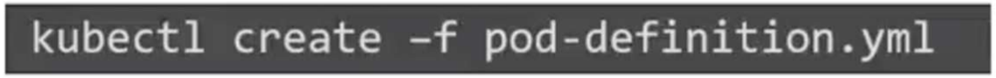


pod-definition.yml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
      app: myapp
      type: front-end
spec:
  containers:
    - name: nginx-container
      image: nginx
```



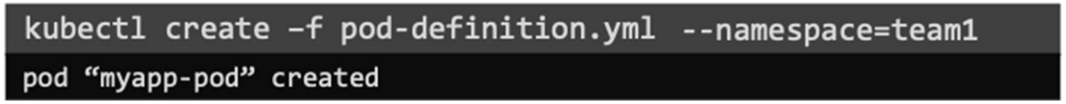
Kind	Version
POD	v1
Service	v1
ReplicaSet	apps/v1
Deployment	apps/v1



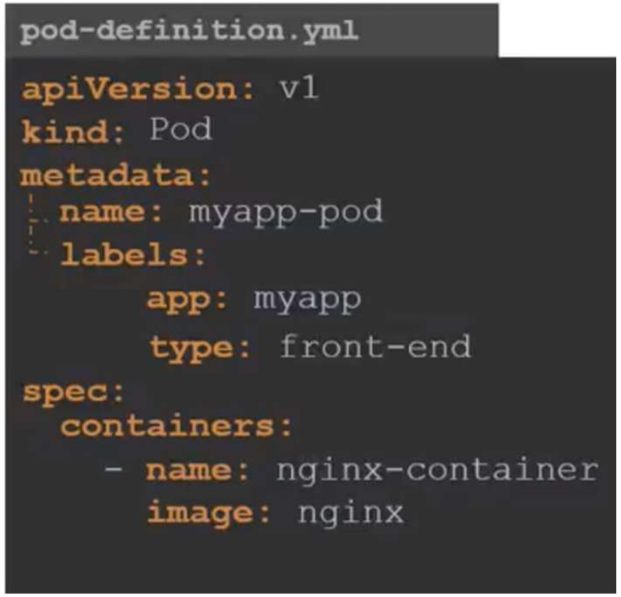
```
kubectl create -f pod-definition.yml
```

Assign Pods to Namespace

- ❖ **First option**

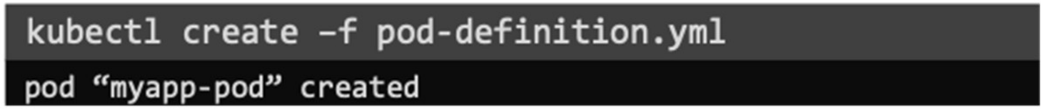


```
kubectl create -f pod-definition.yml --namespace=team1
pod "myapp-pod" created
```

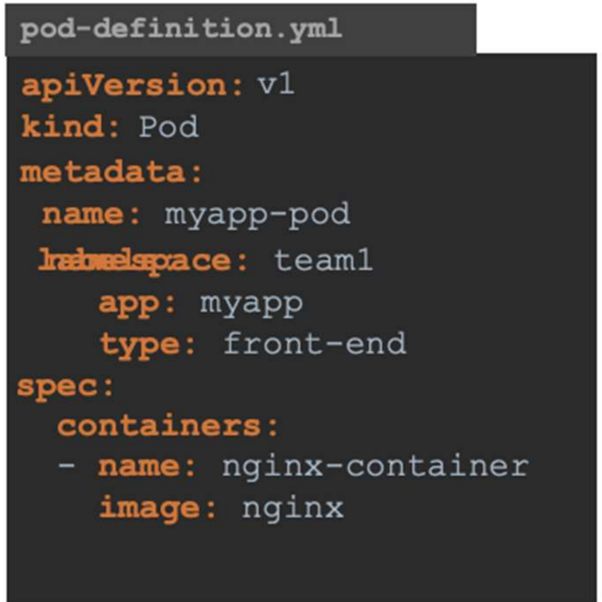


```
pod-definition.yml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
      app: myapp
      type: front-end
spec:
  containers:
    - name: nginx-container
      image: nginx
```

- ❖ **Second option**



```
kubectl create -f pod-definition.yml
pod "myapp-pod" created
```



```
pod-definition.yml
apiVersion: v1
kind: Pod
metadata:
 name: myapp-pod
 namespace: team1
    app: myapp
    type: front-end
spec:
  containers:
  - name: nginx-container
    image: nginx
```

Replication Controllers

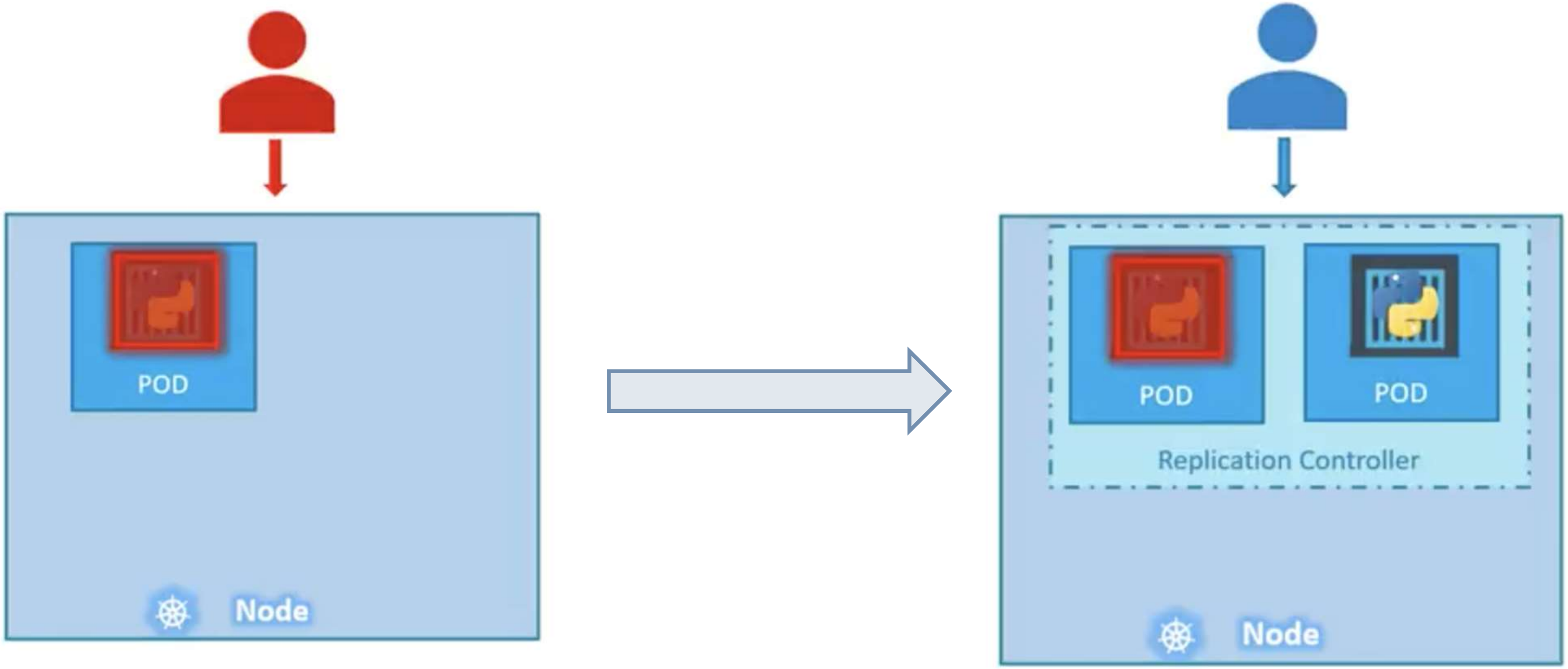
- If pods exit or are deleted, the replication controller acts to instantiate more up to the desired number.
- a replication controller is a Kubernetes object that ensures that a specified number of replicas of a pod are running at all times.
- If there are more running than desired, it deletes as many as necessary to match the number.

- Un replication controller est un objet Kubernetes qui garantit qu'un nombre spécifié de répliques d'un pod est en cours d'exécution en permanence.

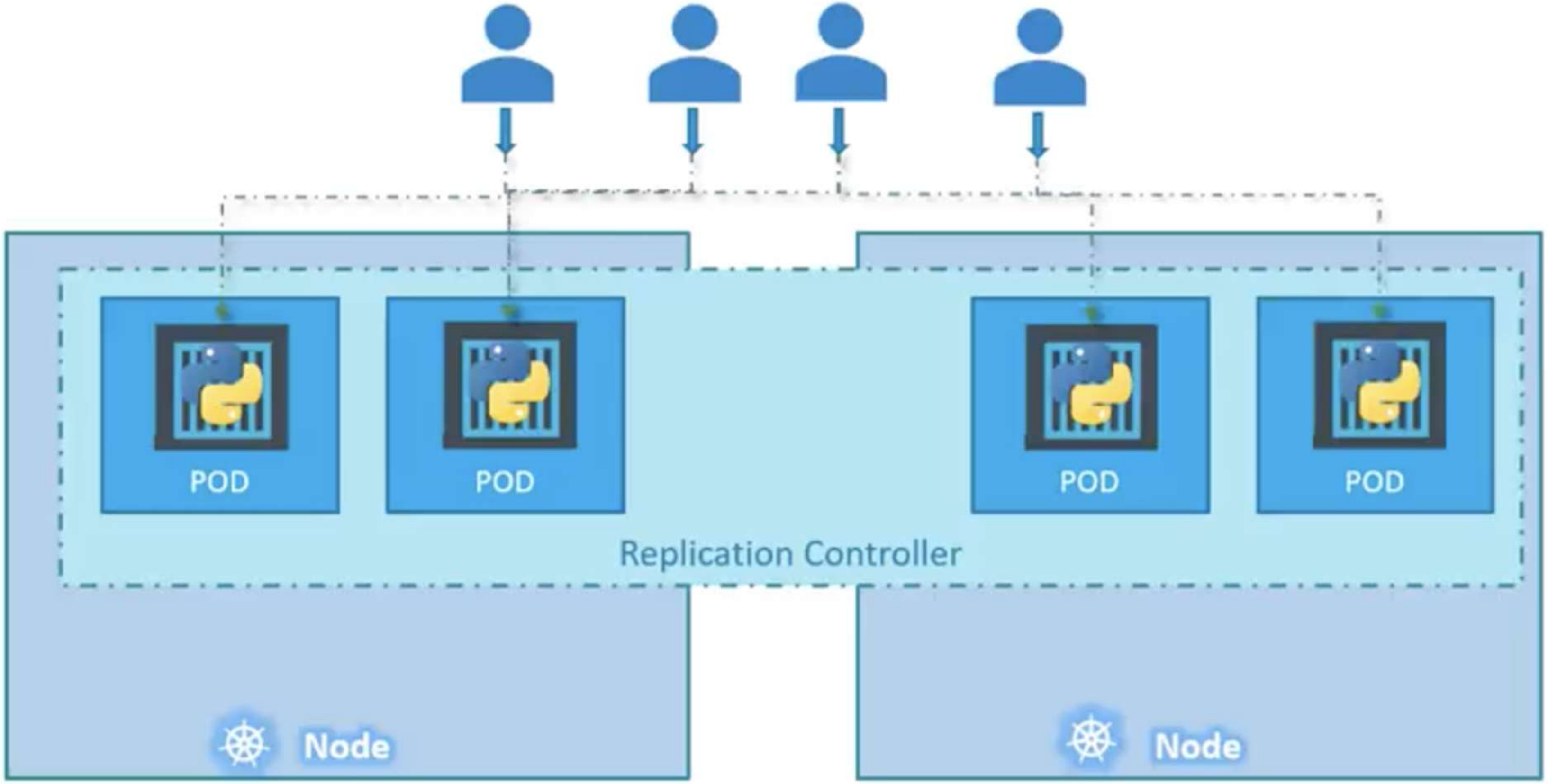
- ❖ Si des pods quittent ou sont supprimés, le contrôleur de réplication agit pour en instancier davantage jusqu'à atteindre le nombre souhaité.

- ❖ S'il y en a plus en cours d'exécution que souhaité, il en supprime autant que nécessaire pour correspondre au nombre désiré.

Replication Controllers : High availability



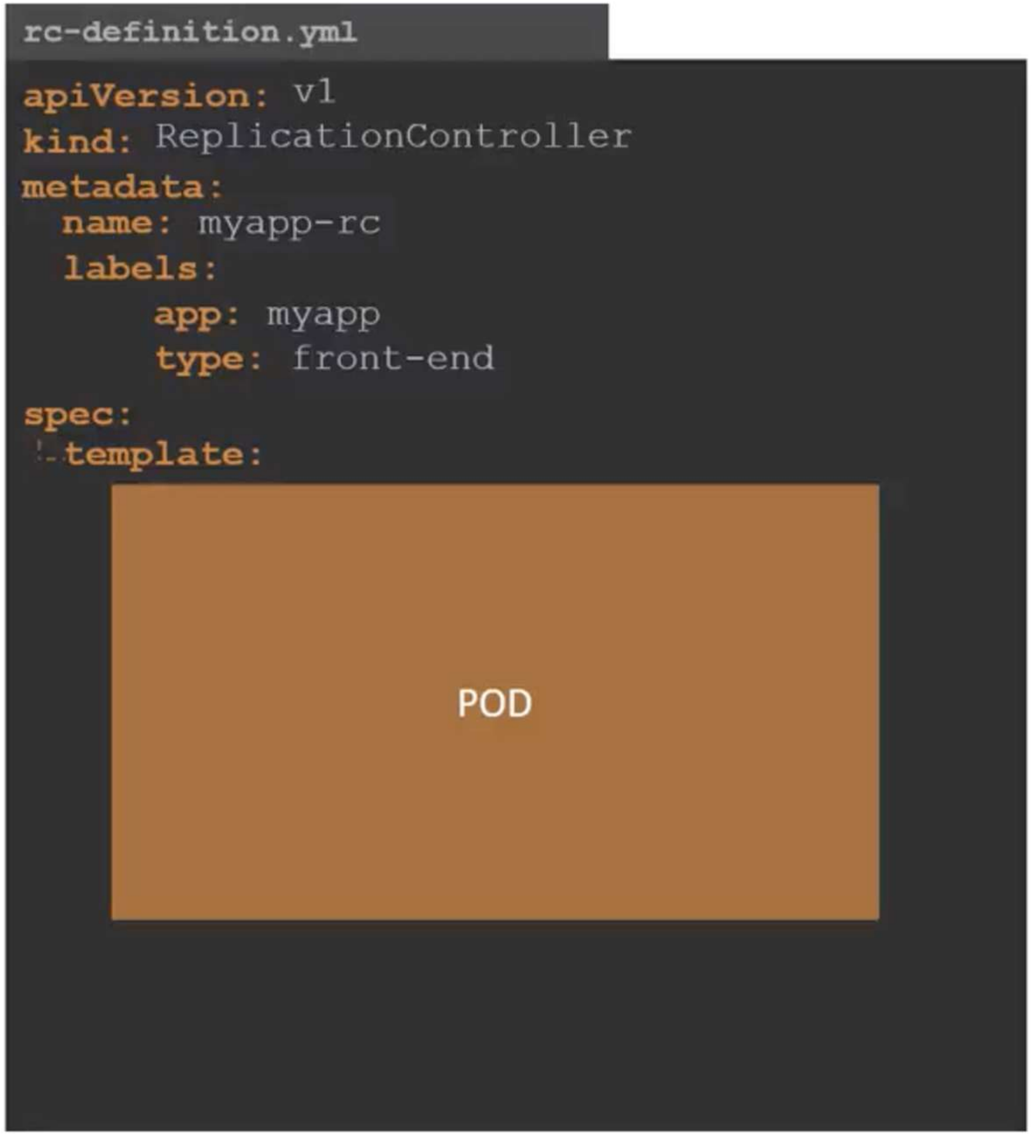
Replication Controllers : Load balancing & Scalling



Setup Replication Controllers

- The definition of a replication controller consists mainly of:
  - The number of replicas desired (which can be adjusted at runtime).
  - A pod definition for creating a replicated pod.
  - ● La définition d'un contrôleur de réplication consiste principalement en :
  - ● Le nombre de répliques souhaitées (qui peut être ajusté à l'exécution).
  - ● Une définition de pod pour créer un pod répliqué.

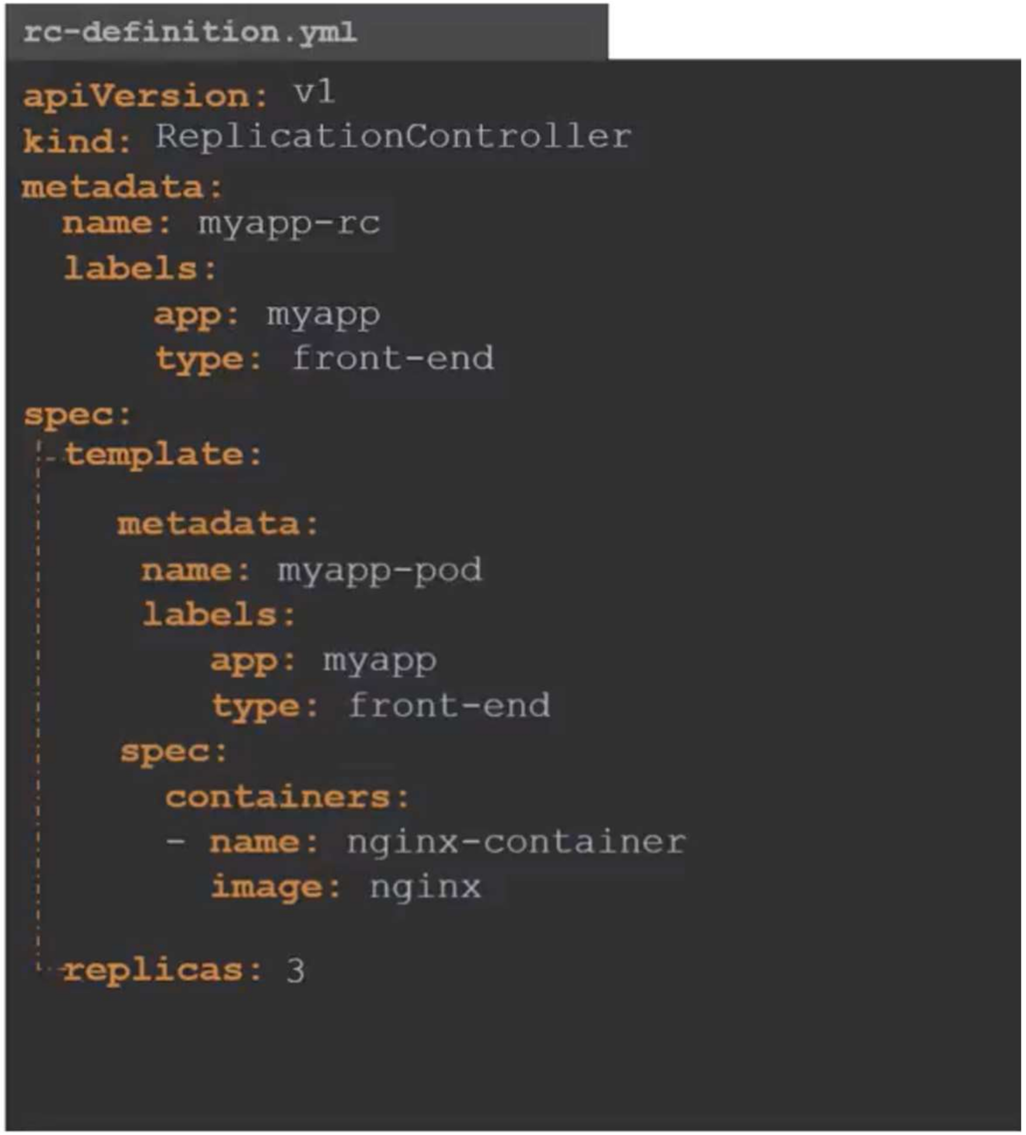
Replication Controllers



rc-definition.yml

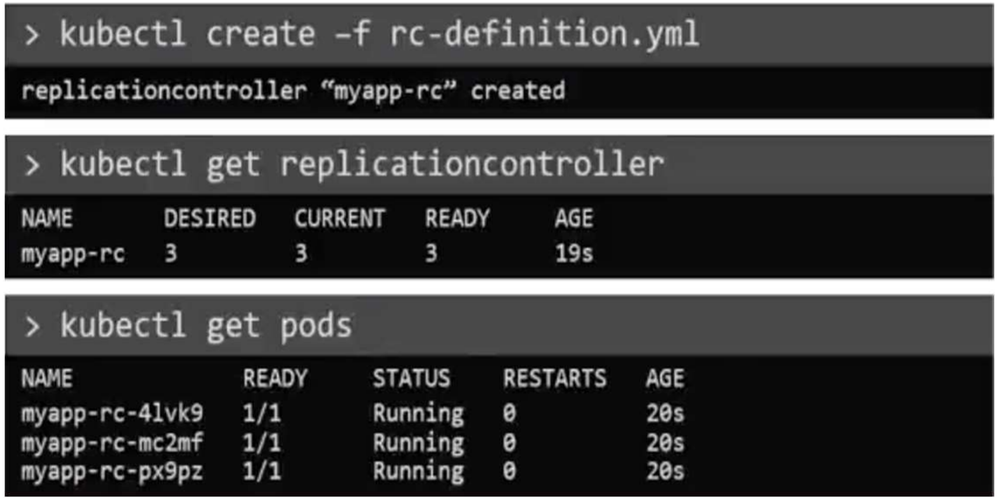
```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: myapp-rc
  labels:
       app: myapp
       type: front-end
spec:
  template:
```

Replication Controllers



rc-definition.yml

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: myapp-rc
  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
```

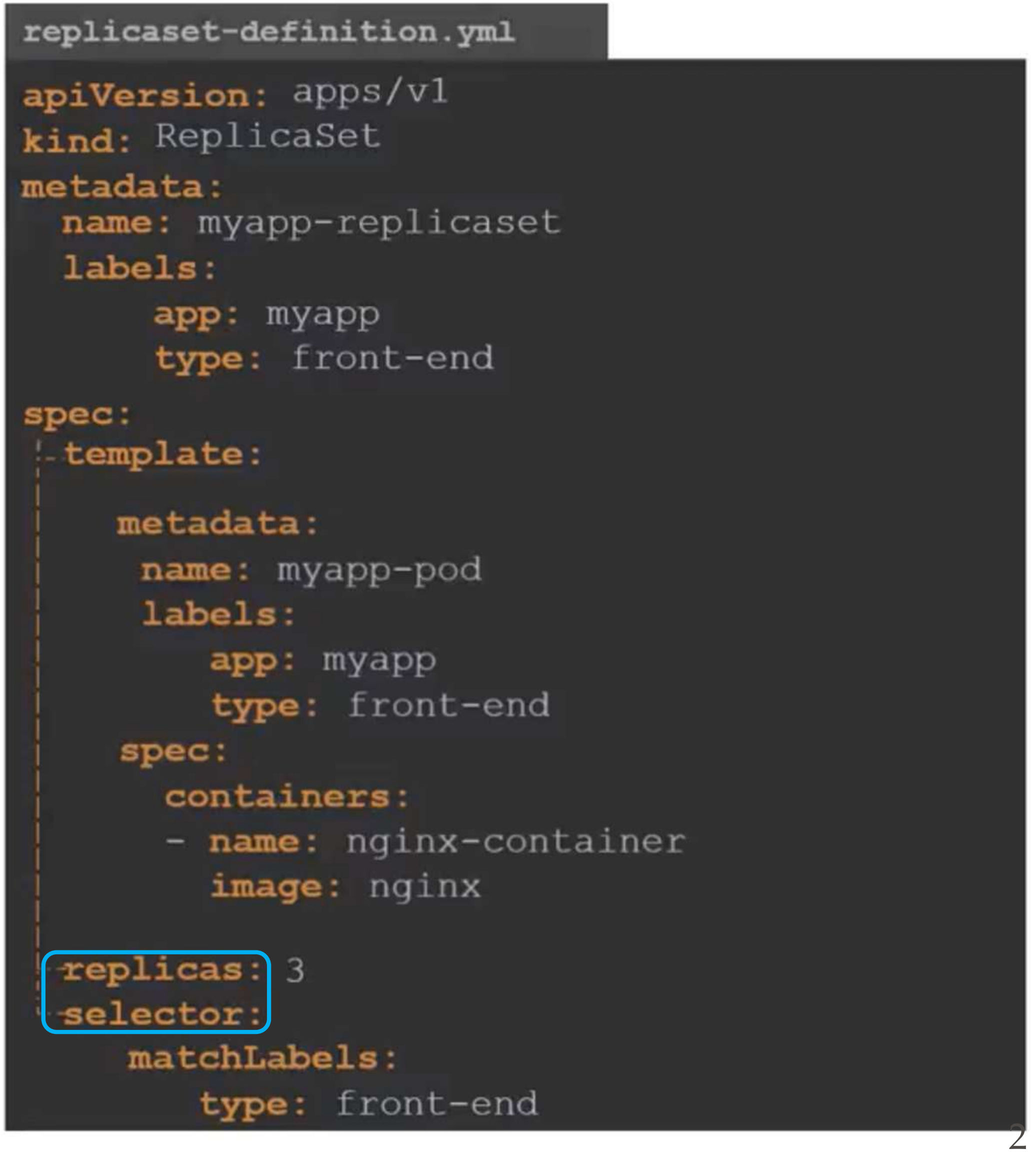


```
> kubectl create -f rc-definition.yml
replicationcontroller “myapp-rc” created
```

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

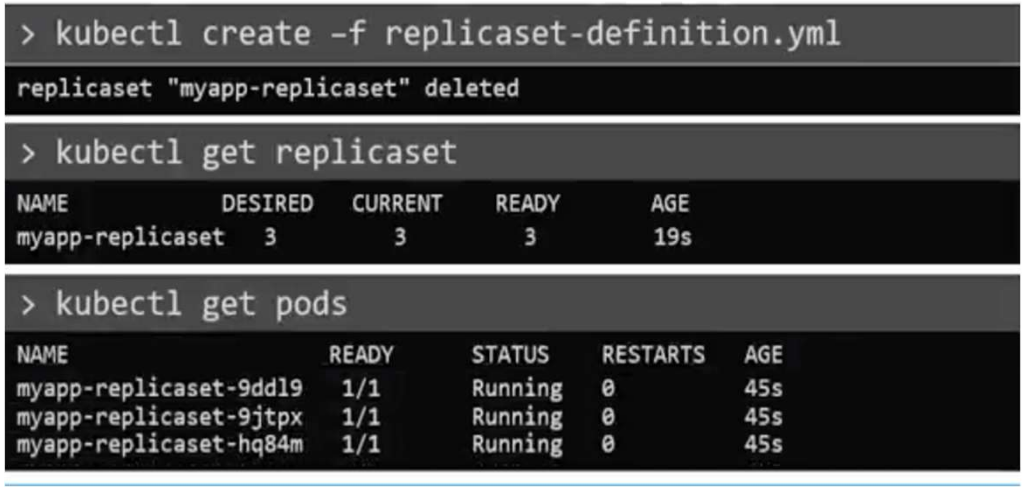
```
> kubectl get pods
NAME                READY     STATUS     RESTARTS    AGE
myapp-rc-4lvk9      1/1       Running    0           20s
myapp-rc-mc2mf      1/1       Running    0           20s
myapp-rc-px9pz      1/1       Running    0           20s
```

ReplicaSet



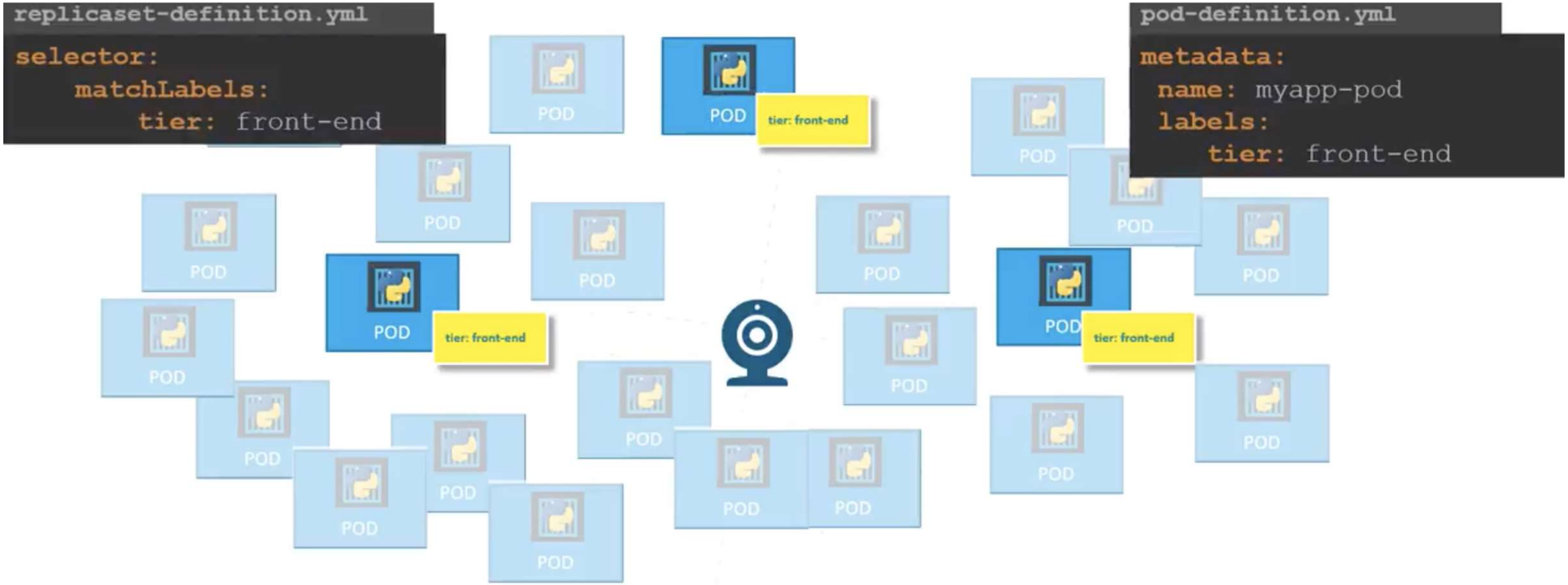
replicaset-definition.yml

```
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
```



```
> kubectl create -f replicaset-definition.yml
replicaset "myapp-replicaset" deleted
> kubectl get replicaset
NAME              DESIRED   CURRENT   READY     AGE
myapp-replicaset   3         3         3         19s
> kubectl get pods
NAME                     READY     STATUS    RESTARTS   AGE
myapp-replicaset-9dd19    1/1       Running   0          45s
myapp-replicaset-9jtpx    1/1       Running   0          45s
myapp-replicaset-hq84m    1/1       Running   0          45s
```

Labels & Selectors



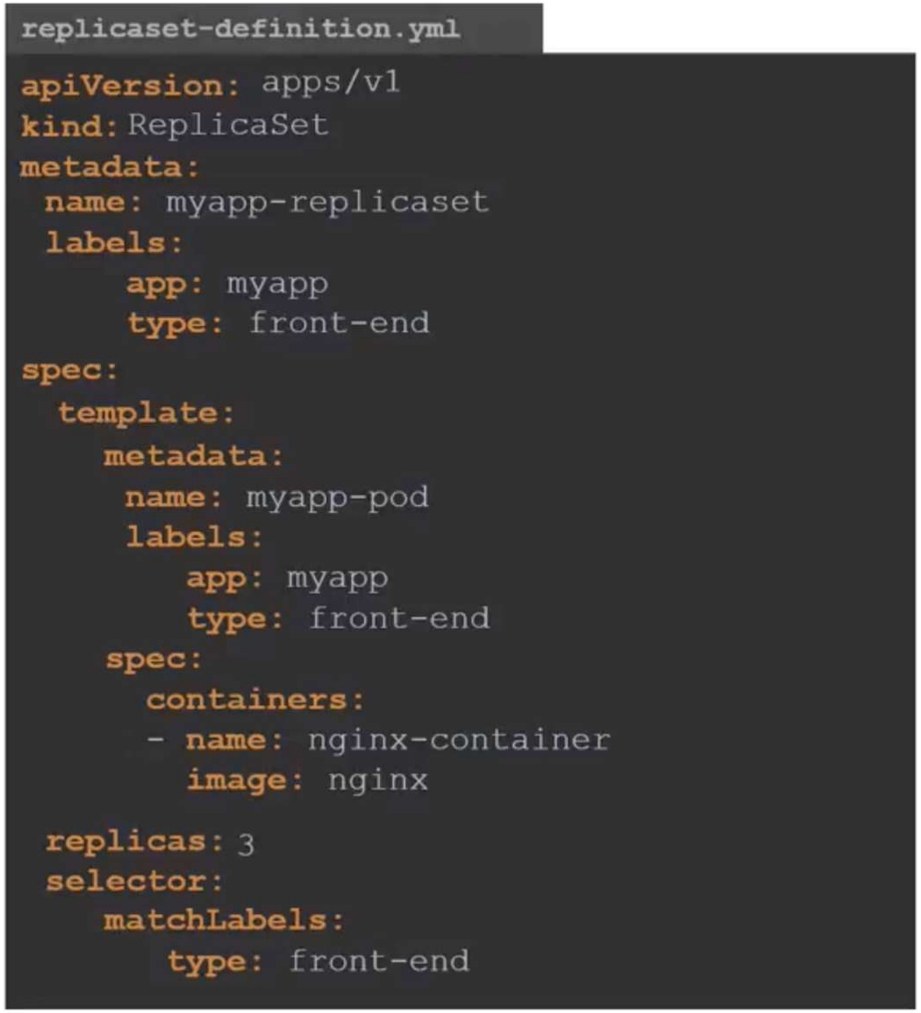
replicaset-definition.yml

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

pod-definition.yml

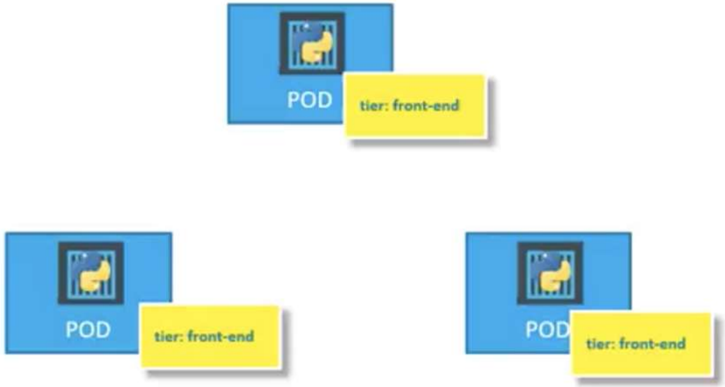
```yaml
metadata:
  name: myapp-pod
  labels:
      tier: front-end
```

Why need we Template section?



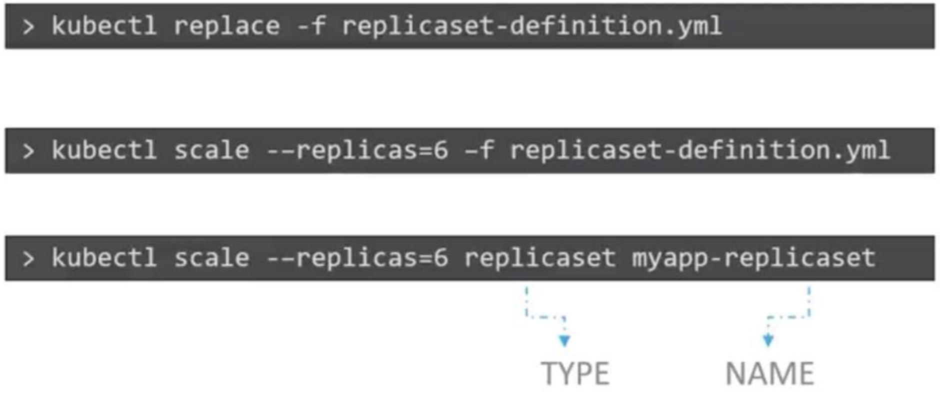
replicaset-definition.yml

```
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
```



Scale ReplicaSet

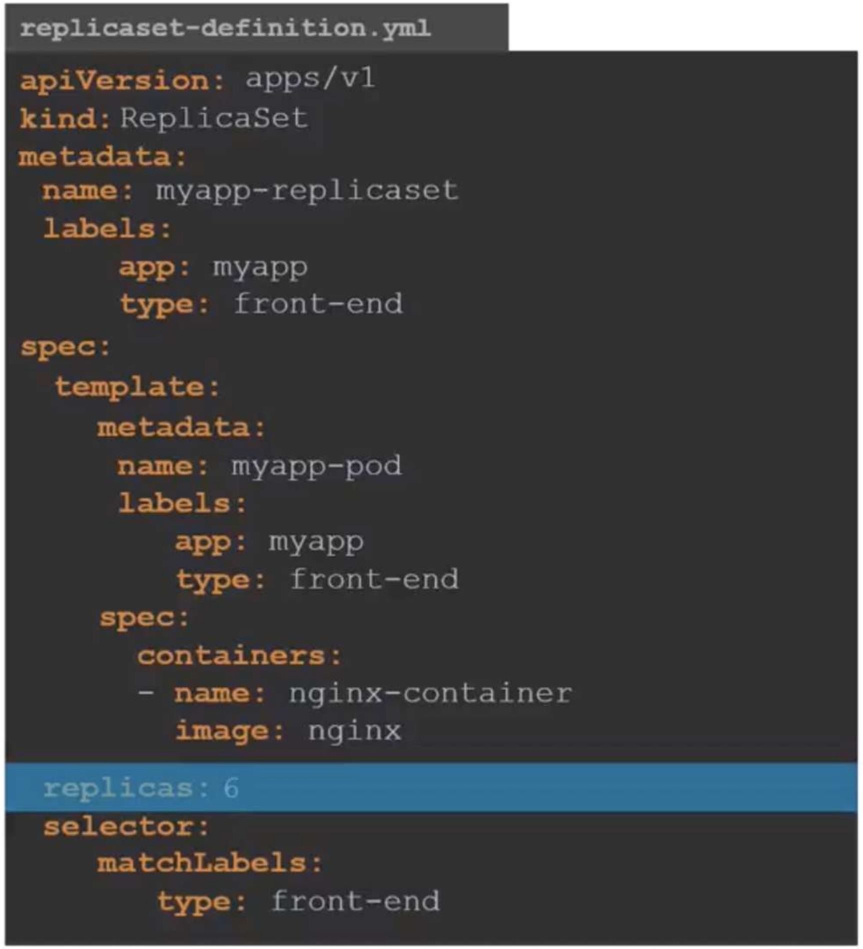
Scale



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

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

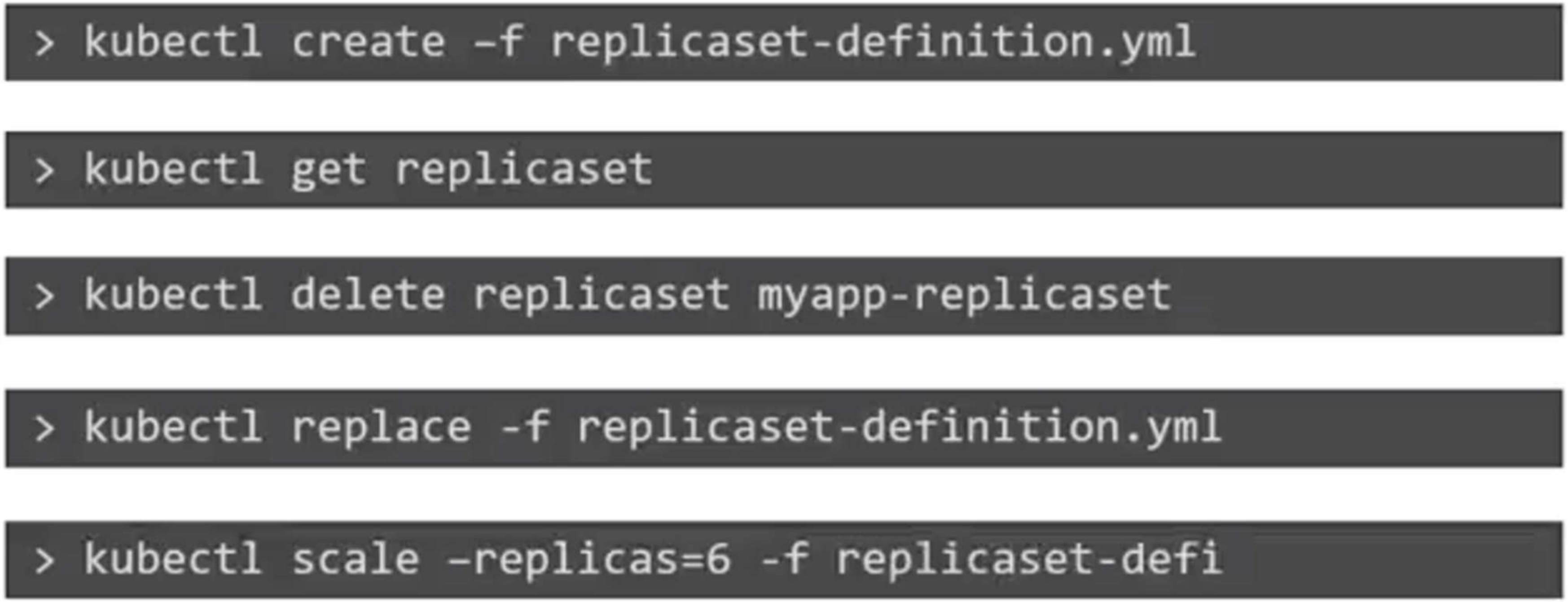
```
> kubectl scale --replicas=6 replicaset myapp-replicaset
```



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
```

Replicaset Commandes



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

```
> kubectl get replicaset
```

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

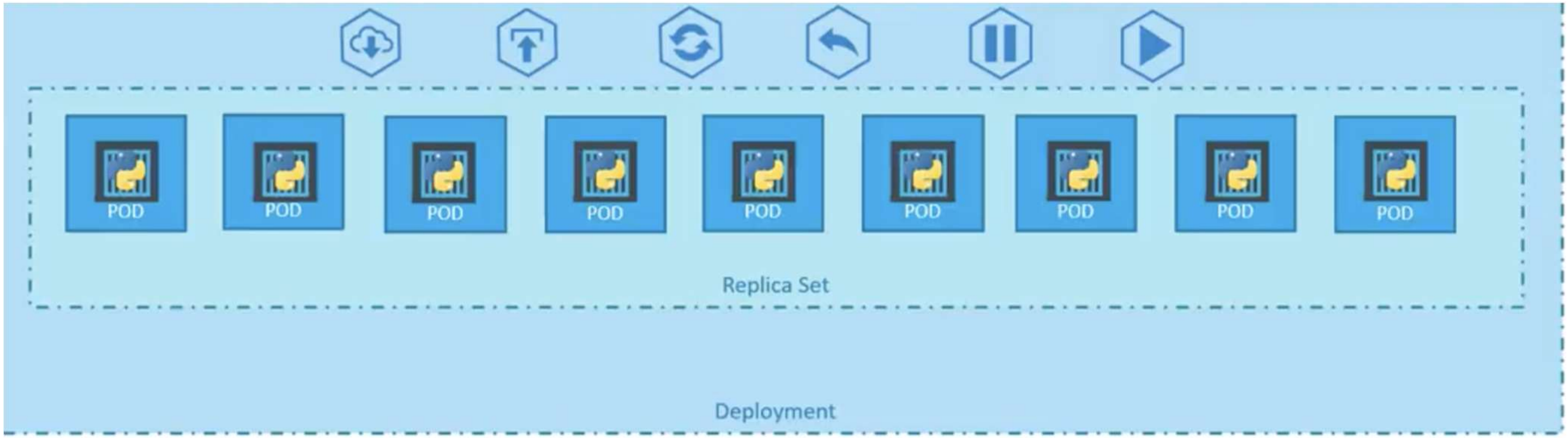
```
> kubectl replace -f replicaset-definition.yml
```

```
> kubectl scale -replicas=6 -f replicaset-defi
```

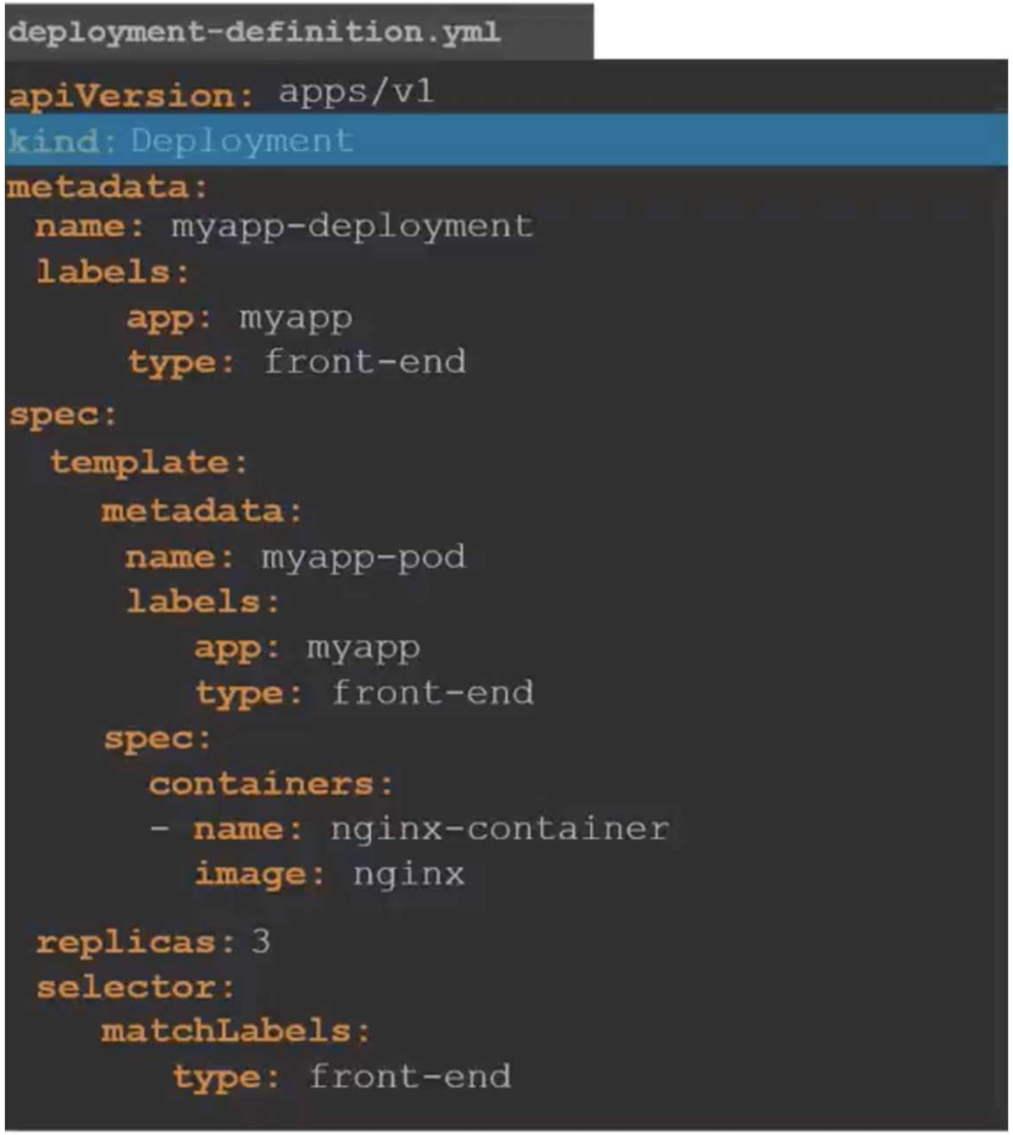
Kubectl scale replicasets   <nom de replicaset> -replicast=6

Deployment

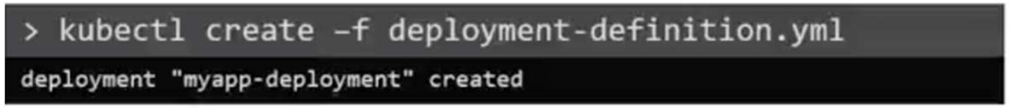
- Kubernetes deployment is Kubernetes resource that object provides in a declarative applications. updates to
- A deployment allows you to describe an application's life cycle, such as which images to use for the app, the number of pods there should be, and the way in which
- Un déploiement Kubernetes est un objet de ressource dans Kubernetes qui permet des mises à jour déclaratives des applications. Un déploiement vous permet de décrire le cycle de vie d'une application, comme les images à utiliser pour l'application, le nombre de pods nécessaires et la manière dont ils doivent être mis à jour.



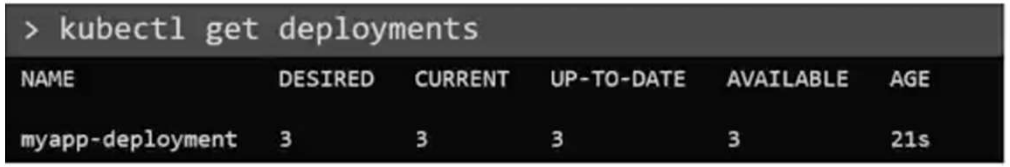
Deployment



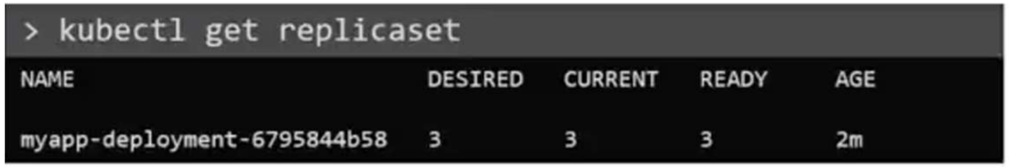
```
deployment-definition.yml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: myapp-deployment
 labels:
     app: myapp
     type: front-end
spec:
  template:
    metadata:
     name: myapp-pod
     labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx
 replicas: 3
 selector:
    matchLabels:
       type: front-end
```



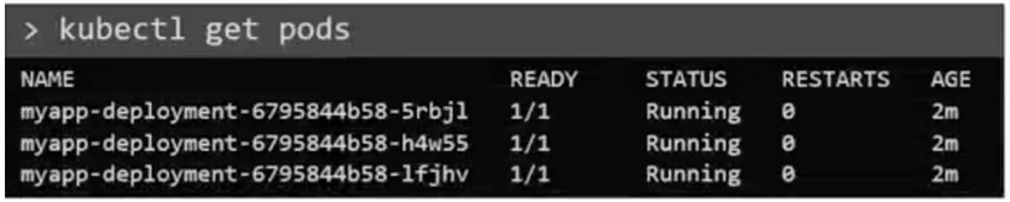
```
> kubectl create -f deployment-definition.yml
deployment "myapp-deployment" created
```



```
> 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
```

Updates & Rollbacks

● Les utilisateurs s'attendent à ce que les applications soient disponibles en tout temps et les développeurs doivent déployer de nouvelles versions plusieurs fois par jour.

● Dans Kubernetes, cela se fait avec des mises à jour progressives. Les mises à jour progressives permettent aux déploiements d'être mis à jour sans interruption en mettant progressivement à jour les instances de Pods avec de nouvelles.

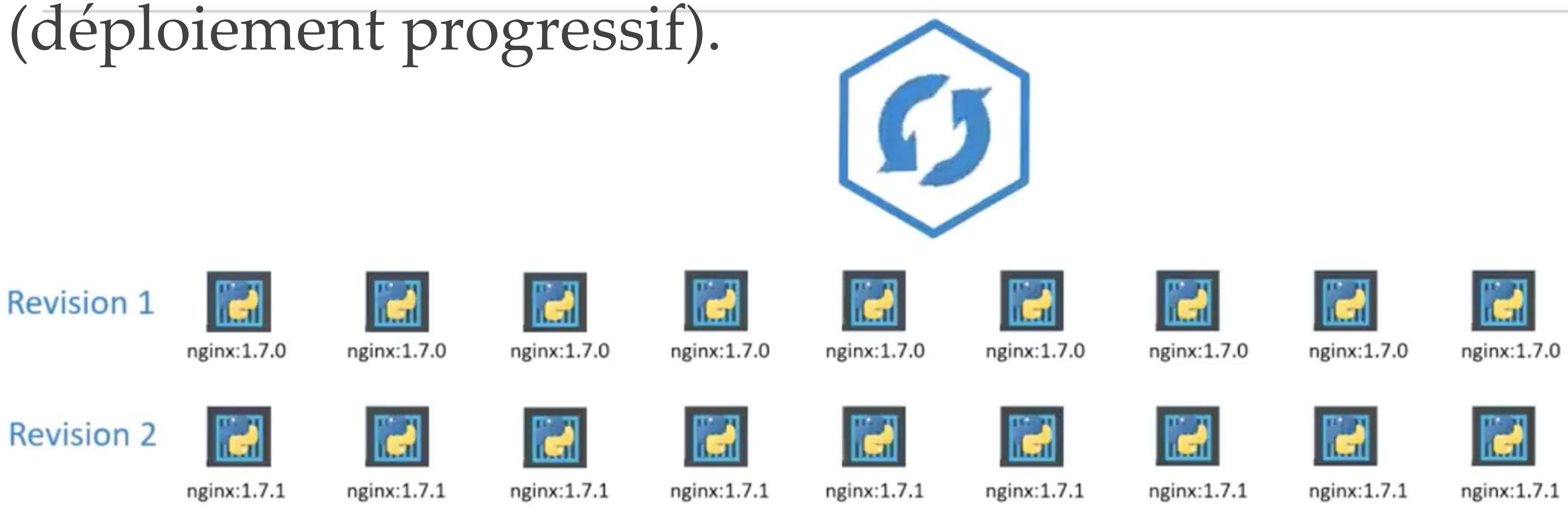
● Les nouveaux Pods seront planifiés sur des nœuds disposant de ressources disponibles.

- Users expect applications to be **available** all the time and developers are expected to deploy **new versions** of them several times a day.
- In Kubernetes this is done with rolling updates. Rolling updates allow Deployments' update to take place with **zero downtime** by **incrementally** updating Pods instances with new ones.
- The new Pods will be scheduled on Nodes with available resources.

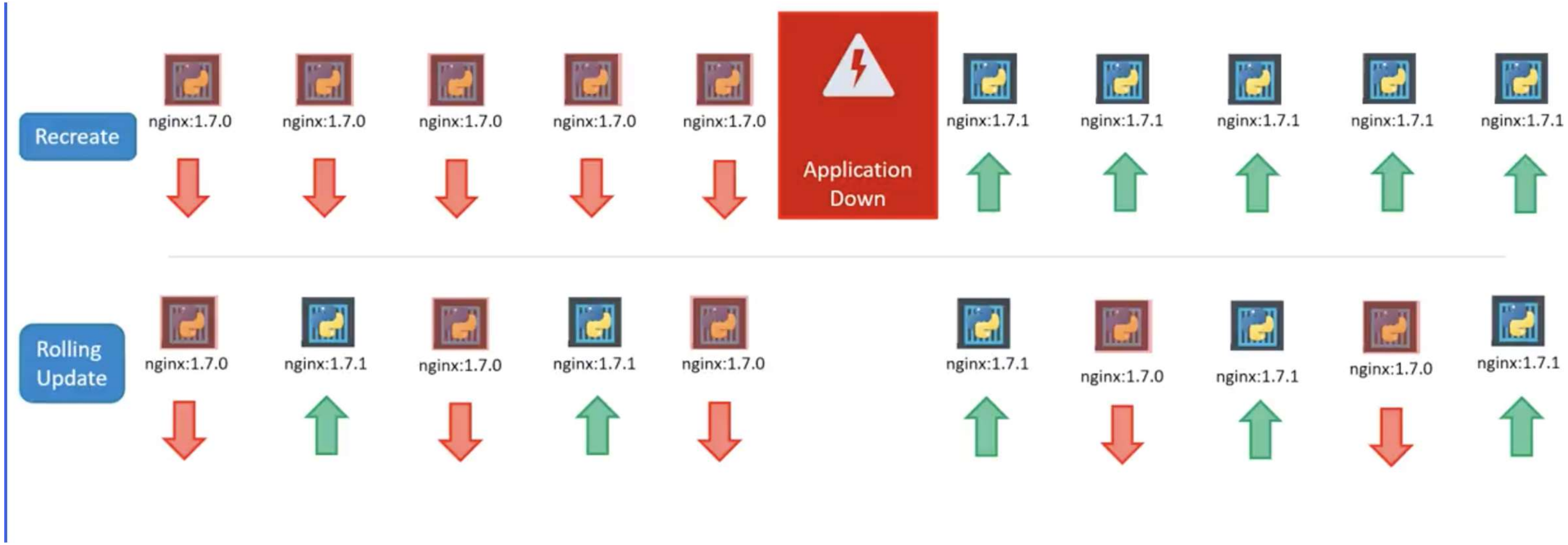
Updates & Rollbacks

- If the application instances are upgrated a new deployment revision is applied (rolloing aout) .

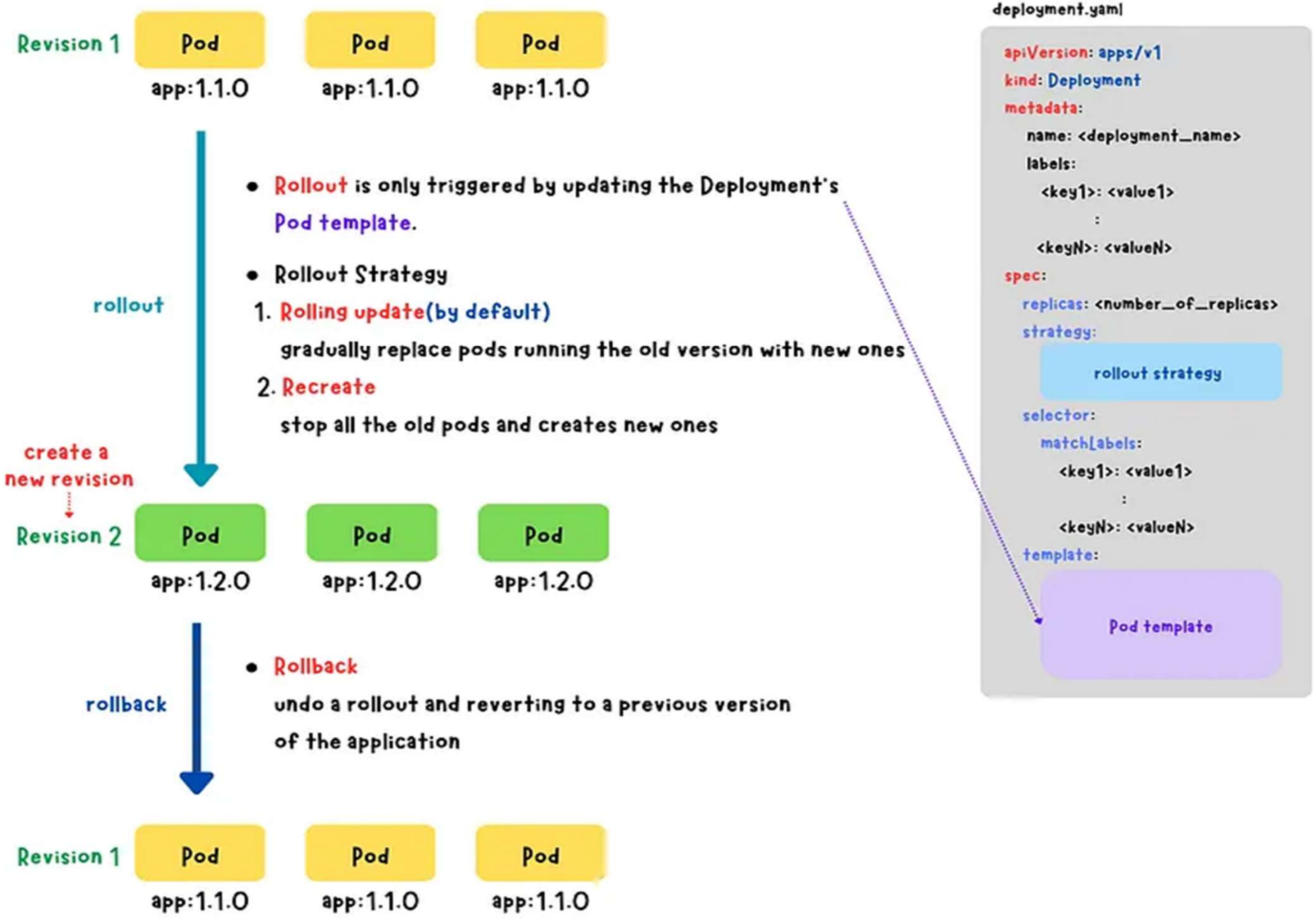
Si les instances de l'application sont mises à niveau, une nouvelle révision du déploiement est appliquée (déploiement progressif).



Deployment Strategy



Kubernetes: Rollout Strategy



- Rollout is only triggered by updating the Deployment's Pod template.
- Rollout Strategy

1. Rolling update(by default)

   gradually replace pods running the old version with new ones

2. Recreate

   stop all the old pods and creates new ones

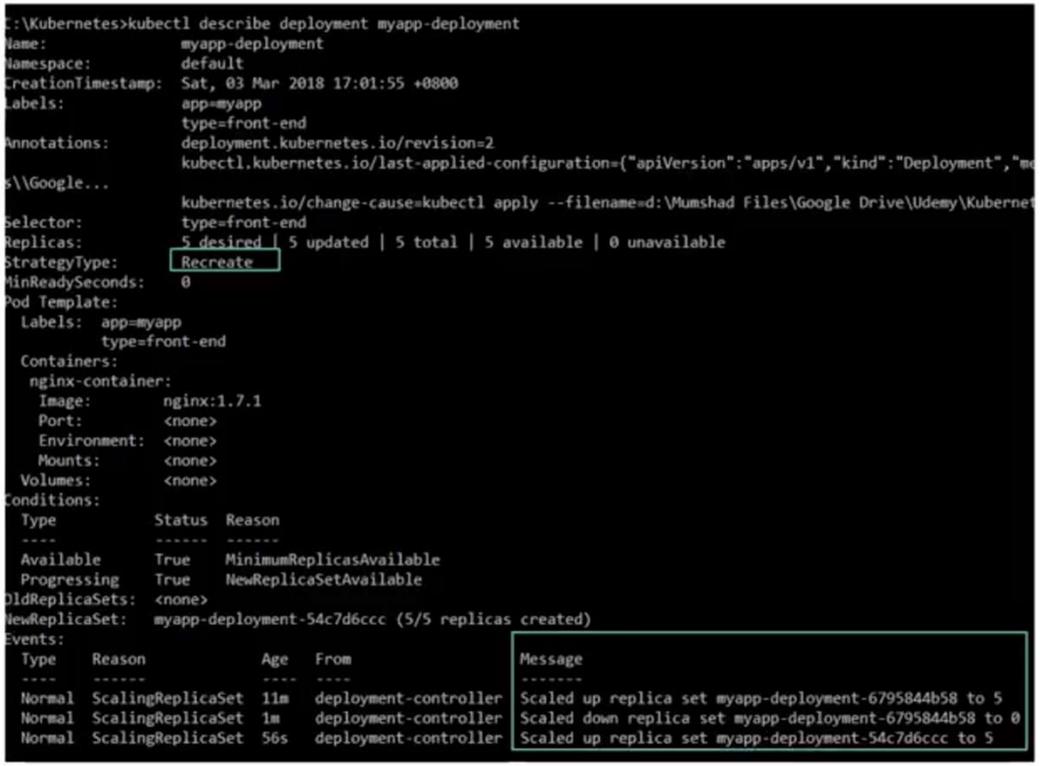
- Rollback

  undo a rollout and reverting to a previous version of the application

deployment.yaml

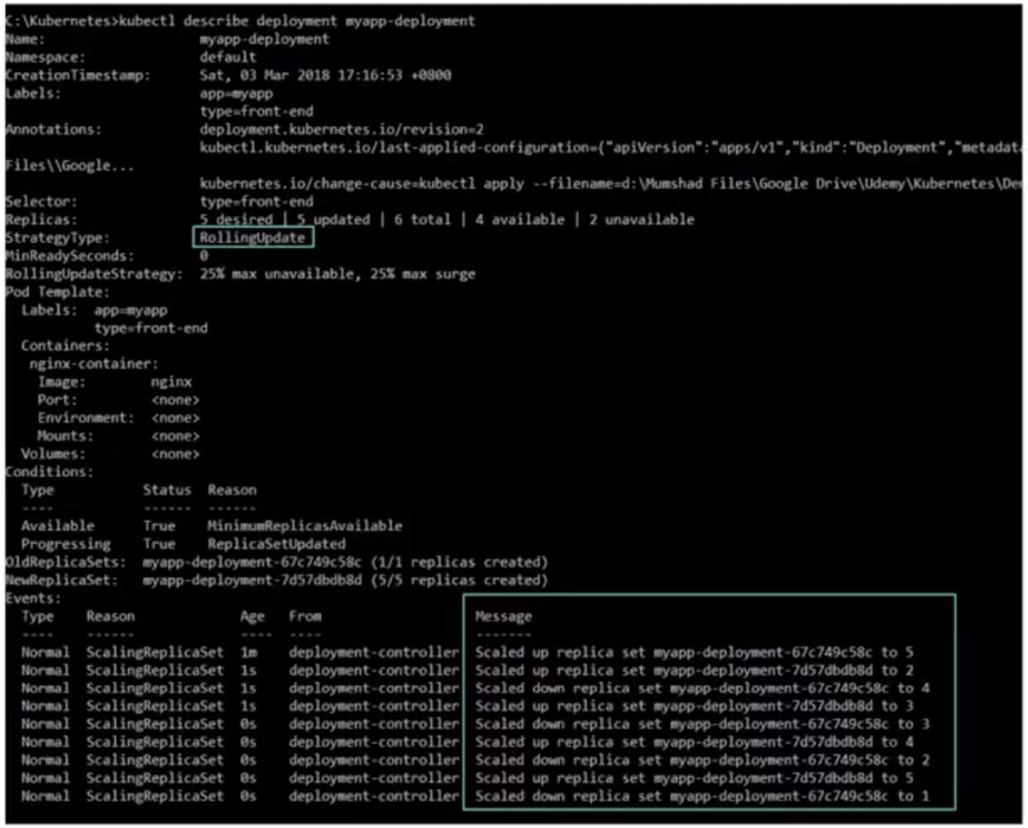
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: <deployment_name>
  labels:
    <key1>: <value1>
         :
    <keyN>: <valueN>
spec:
  replicas: <number_of_replicas>
  strategy:
      rollout strategy
  selector:
    matchLabels:
      <key1>: <value1>
           :
      <keyN>: <valueN>
  template:
      Pod template
```

Recrate Vs Rolling Update



```
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
```

Recreate

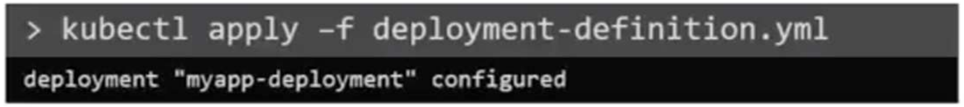


```
C:\Kubernetes>kubectl describe deployment myapp-deployment
Name:                   myapp-deployment
Namespace:              default
CreationTimestamp:      Sat, 03 Mar 2018 17:16:53 +0800
Labels:                 app=myapp
                        type=front-end
Annotations:            deployment.kubernetes.io/revision=2
                        kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"apps/v1","kind":"Deployment","metadat
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
```

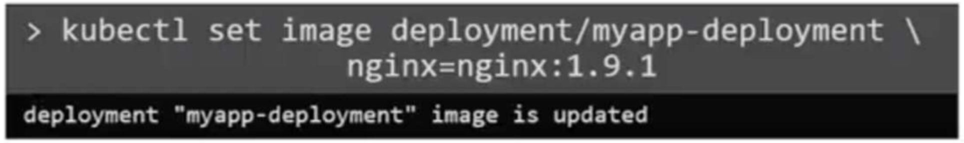
RollingUpdate

Updates

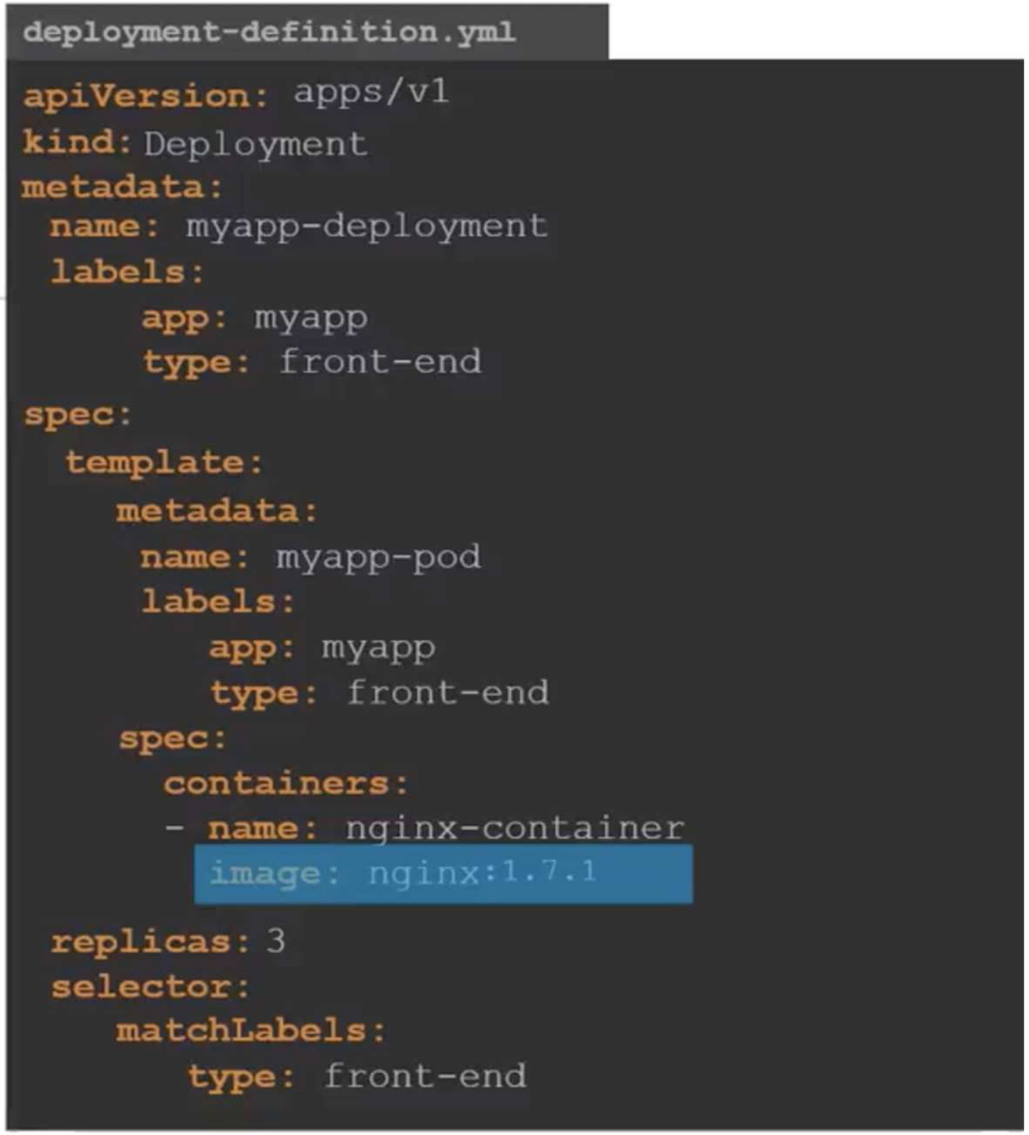
Kubectl apply



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



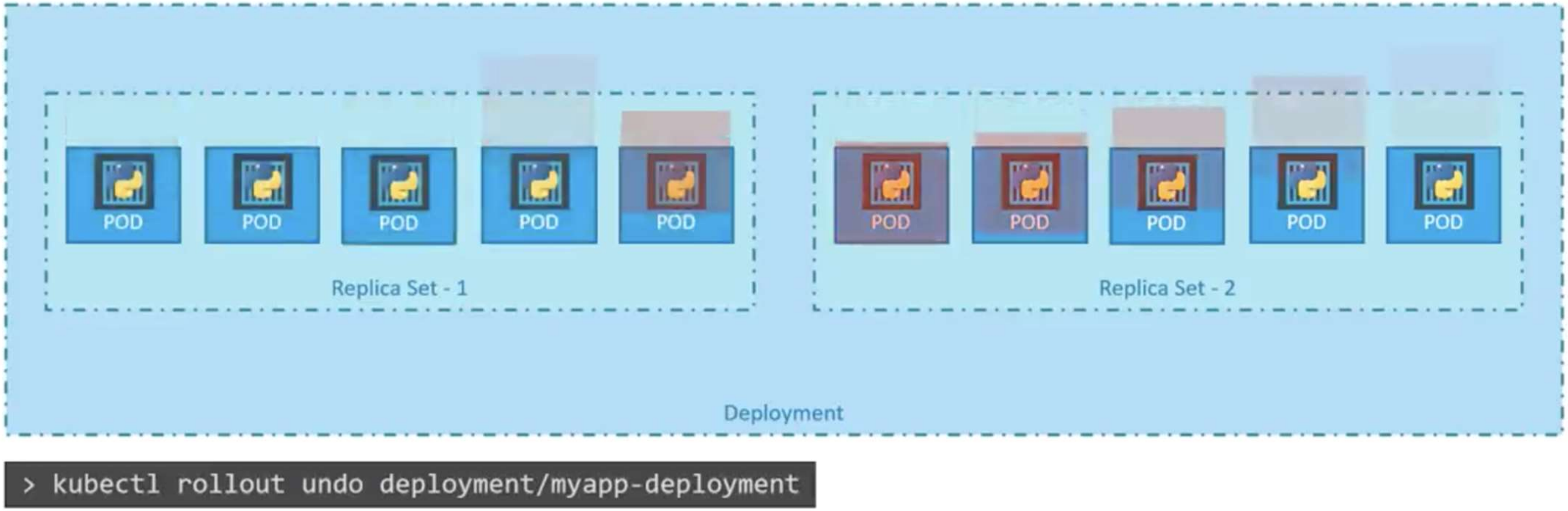
```
> kubectl set image deployment/myapp-deployment \
                nginx=nginx:1.9.1
deployment "myapp-deployment" image is updated
```



deployment-definition.yml

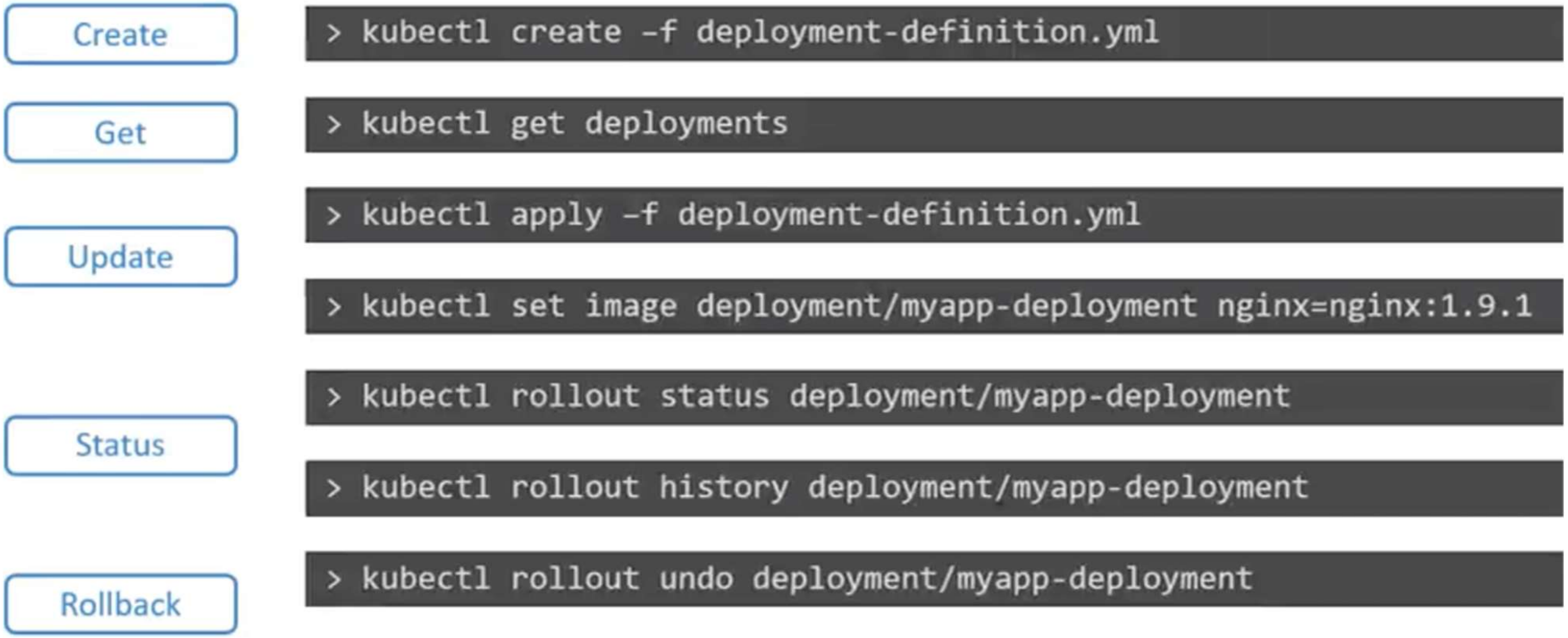
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
  labels:
      app: myapp
      type: front-end
spec:
  template:
    metadata:
      name: myapp-pod
      labels:
          app: myapp
          type: front-end
    spec:
      containers:
      - name: nginx-container
  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

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

Rollback



```
> kubectl rollout undo deployment/myapp-deployment
```

Deployment 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
```

Lab 2 : Play with K8s