BATCH : B107 AWS-DevOps

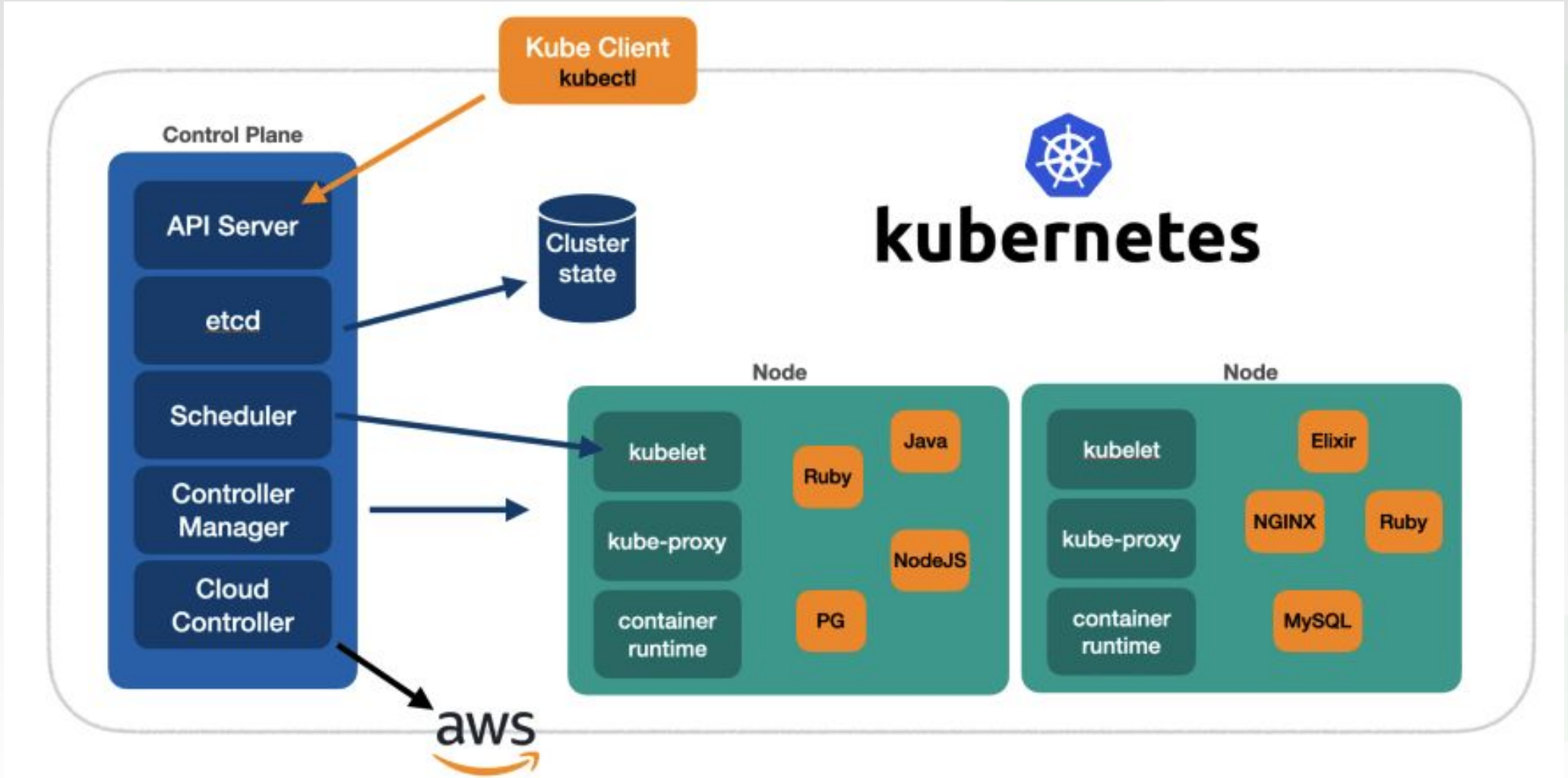
LESSON : **Kubernetes**

DATE : 20.05.2023

SUBJECT : **Kubernetes Objects**

ZOOM GİRİŞLERİNİZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ

Kubernetes


Kube Client
kubectl
Control Plane
API Server
etcd
Scheduler
Controller Manager
Cloud Controller
Cluster state
kubernetes
Node
kubelet
kube-proxy
container runtime
Ruby
Java
NodeJS
PG
Node
kubelet
kube-proxy
container runtime
Elixir
NGINX
Ruby
MySQL
aws

Important Terms

- In robotics and automation, **a control loop** is a non-terminating loop that regulates the state of a system. Thermostat in a room can be an example. When you set the temperature, that's telling the thermostat about your **desired state**. The actual room temperature is the **current state**.

- In Kubernetes, **controllers are control loops that watch the state of your cluster**. Each controller tries to move the current cluster state closer to the desired state.

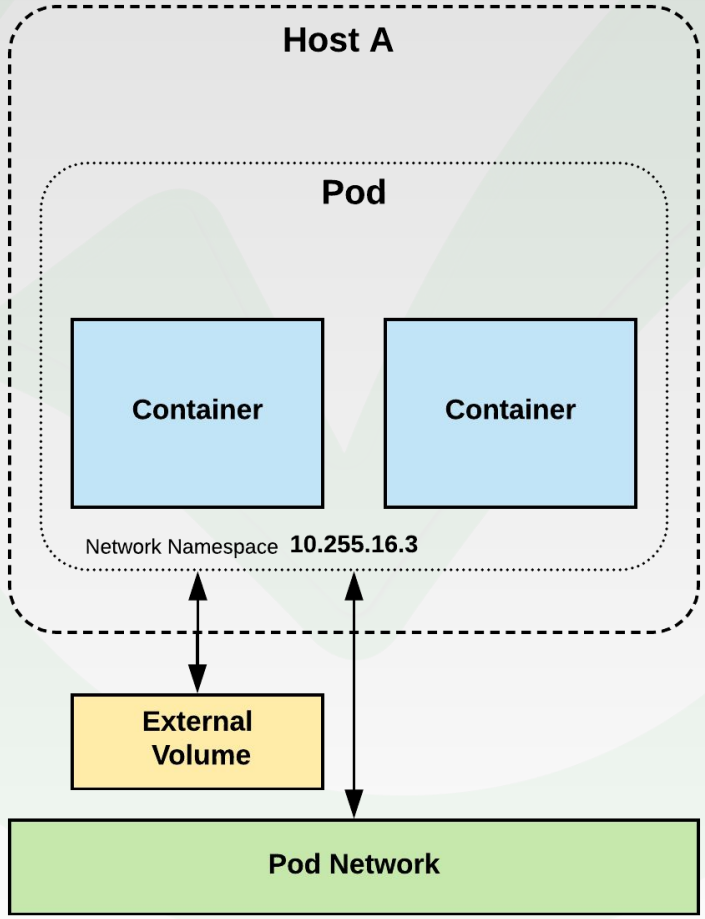
- **A workload is an application** running on Kubernetes. Whether your workload is **a single component or several** that work together, on Kubernetes you run it inside a set of pods.

- You can use **workload resources** that manage a set of pods on your behalf. These **resources configure controllers** that make sure the right number of the right kind of pod are running, to match the state you specified.

- This **desired state** is mentioned in the **spec field**.

Pods

- Atomic unit or **smallest "*unit of work*"** of Kubernetes.
- Pods contain **one or MORE containers** that share volumes, a network, and are a part of a single context.
- Pods are used in two main ways:
  - Pods that run a single container
  - Pods that run multiple containers that need to work together
- Pods are virtual machines.
- **Nodes are physical machines.**



Pod Template

- Workload Controllers manage instances of Pods based off a provided template.
- Pod Templates are Pod specs with limited metadata.
- Controllers use Pod Templates to make actual pods.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-example
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx
```

```yaml
template:
  metadata:
    labels:
      app: nginx
  spec:
    containers:
    - name: nginx
      image: nginx
```

ReplicaSet

- Primary method of **managing pod replicas and their lifecycle**.
- Includes their **scheduling, scaling, and deletion**.
- Their job is simple: **Always ensure the desired number of pods are running**.

ReplicaSet

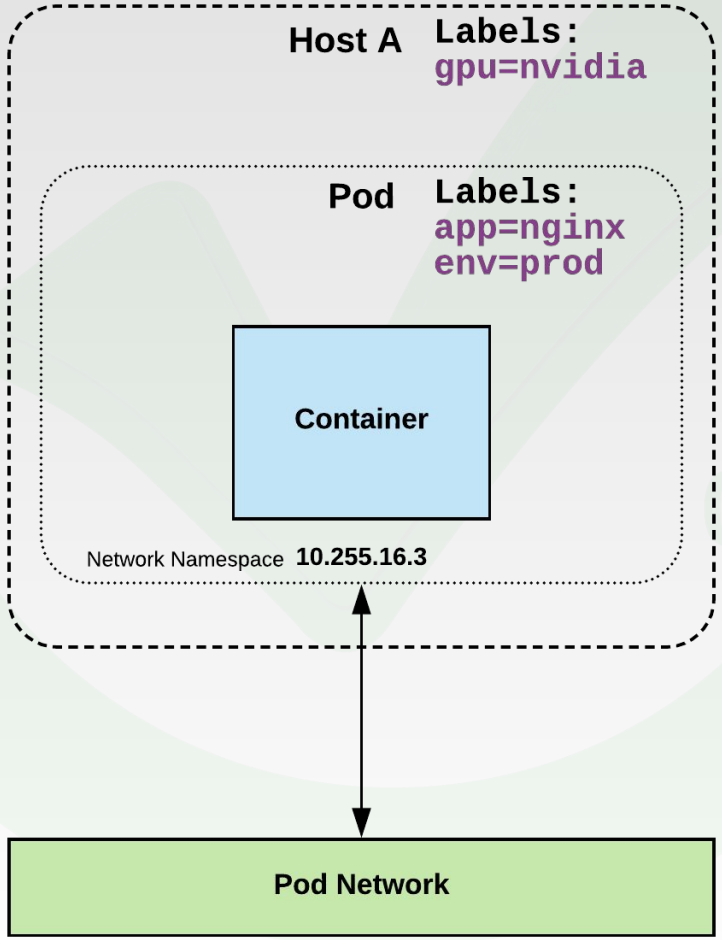
```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: rs-example
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  template:
    <pod template>
```

- `replicas:` The desired number of instances of the Pod.
- `selector:` The label selector for the ReplicaSet will manage ALL Pod instances that it targets; whether it's desired or not.

Labels

- **key-value pairs** that are used to **identify, describe and group** together related sets of objects or resources.
- pods and nodes can be labeled
- NOT characteristic of uniqueness.
- Have a strict syntax with a slightly limited character set.

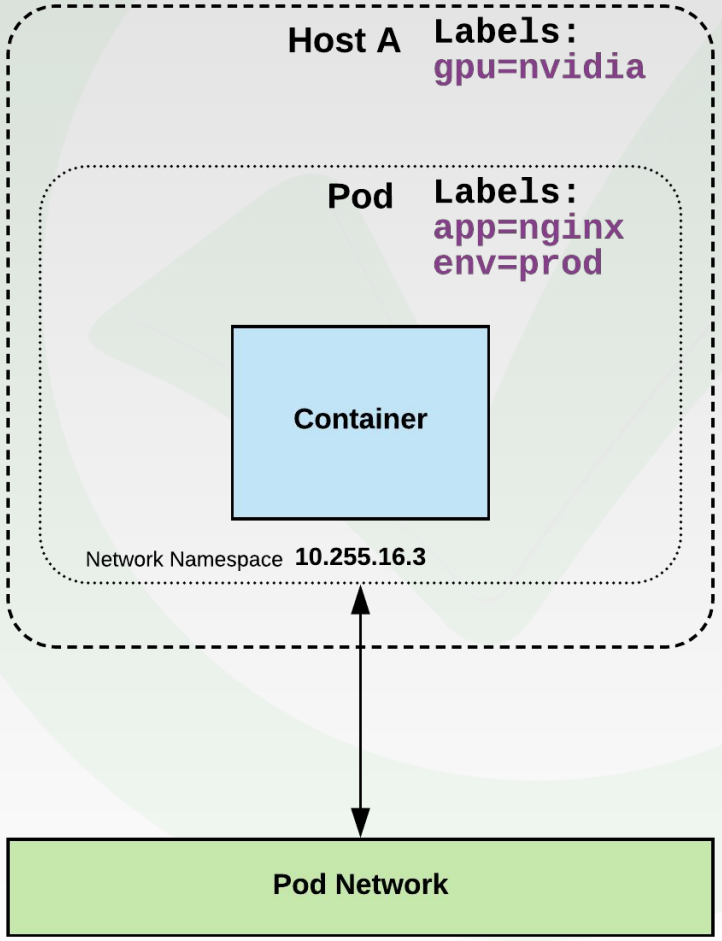
* https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#syntax-and-character-set



Label Example

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-label-example
  labels:
    app: nginx
    env: prod
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    ports:
    - containerPort: 80
```

```
kubectl label pod nginx app=nginx
kubectl label node host1 gpu=nvidia
```



Selectors

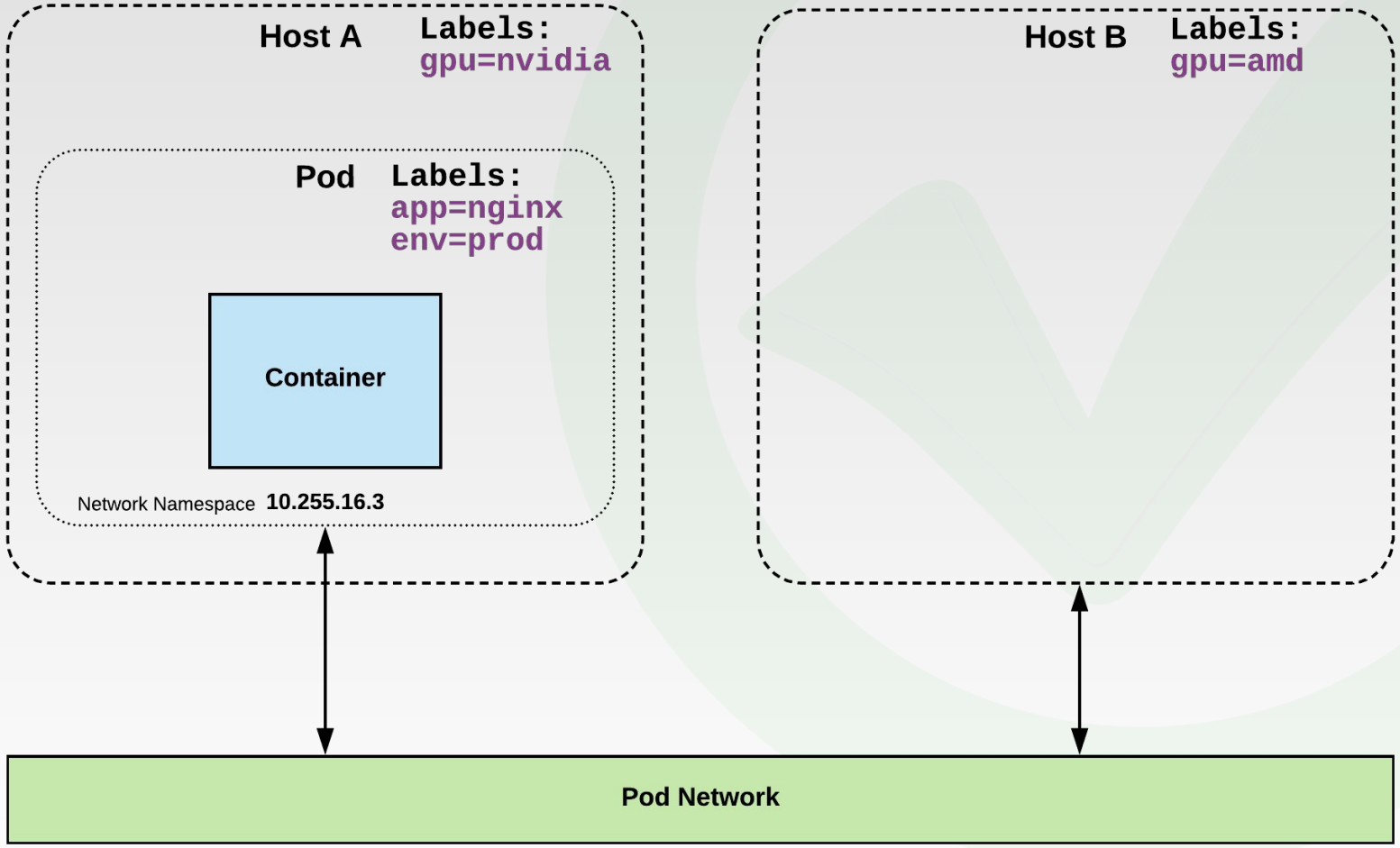
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-label-example
  labels:
    app: nginx
    env: prod
spec:
  containers:
  - name: nginx
    image:
nginx:stable-alpine
    ports:
    - containerPort: 80
  nodeSelector:
    gpu: nvidia
```

Selectors use labels to filter or select objects, and are used throughout Kubernetes.

```yaml
selector:
  matchLabels:
    app: nginx
```

Selector Example

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-label-example
  labels:
    app: nginx
    env: prod
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    ports:
    - containerPort: 80
  nodeSelector:
    gpu: nvidia
```



Deployment

- Declarative method of **managing Pods via ReplicaSets.**
- Provide **rollback functionality and update control.**
- Updates are managed through the pod-template-hash label.
- Each iteration creates a unique label that is assigned to both the ReplicaSet and subsequent Pods.

Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-example
spec:
  replicas: 3
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    <pod template>
```

- **`revisionHistoryLimit:` The number of previous iterations of the Deployment to retain.**
- **`strategy:` Describes the method of updating the Pods based on the `type`. Valid options are `Recreate` or `RollingUpdate`.**
  - `Recreate:` All existing Pods are killed before the new ones are created.
  - `RollingUpdate:` Cycles through updating the Pods according to the parameters: `maxSurge` and `maxUnavailable`.
    - maxUnavailable: how many pods can be unavailable at most during update
    - maxSurge: how many pods can exceed the desired number of Pods during update

Namespaces

```
apiVersion: v1
kind: Namespace
metadata:
  name: prod
  labels:
    app: MyBigWebApp
```

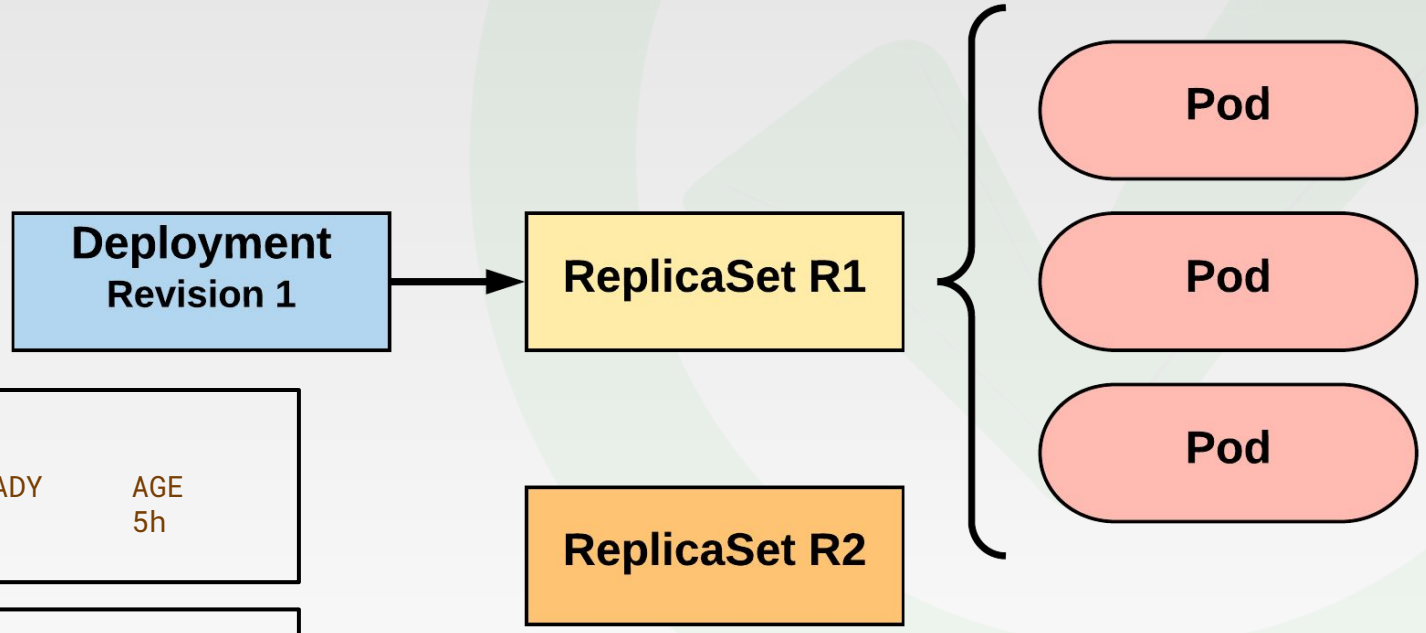
- Namespaces are a logical cluster or environment, and are the primary method of partitioning a cluster or scoping access.
- Namespaces are a way to divide cluster resources between multiple users.

```
$ kubectl get ns --show-labels
NAME          STATUS    AGE       LABELS
default       Active    11h       <none>
kube-public   Active    11h       <none>
kube-system   Active    11h       <none>
prod          Active    6s        app=MyBigWebApp
```

Rolling Update Deployment

Updating pod template generates a new **ReplicaSet** revision.

R1 pod-template-hash:
`676677fff`
R2 pod-template-hash:
`54f7ff7d6d`




```
$ kubectl get replicaset
NAME                        DESIRED   CURRENT   READY     AGE
mydep-6766777fff            3         3         3         5h
```

```
$ kubectl get pods
NAME                        READY     STATUS    RESTARTS   AGE
mydep-6766777fff-9r2zn      1/1       Running   0          5h
mydep-6766777fff-hsfz9      1/1       Running   0          5h
mydep-6766777fff-sjxhf      1/1       Running   0          5h
```

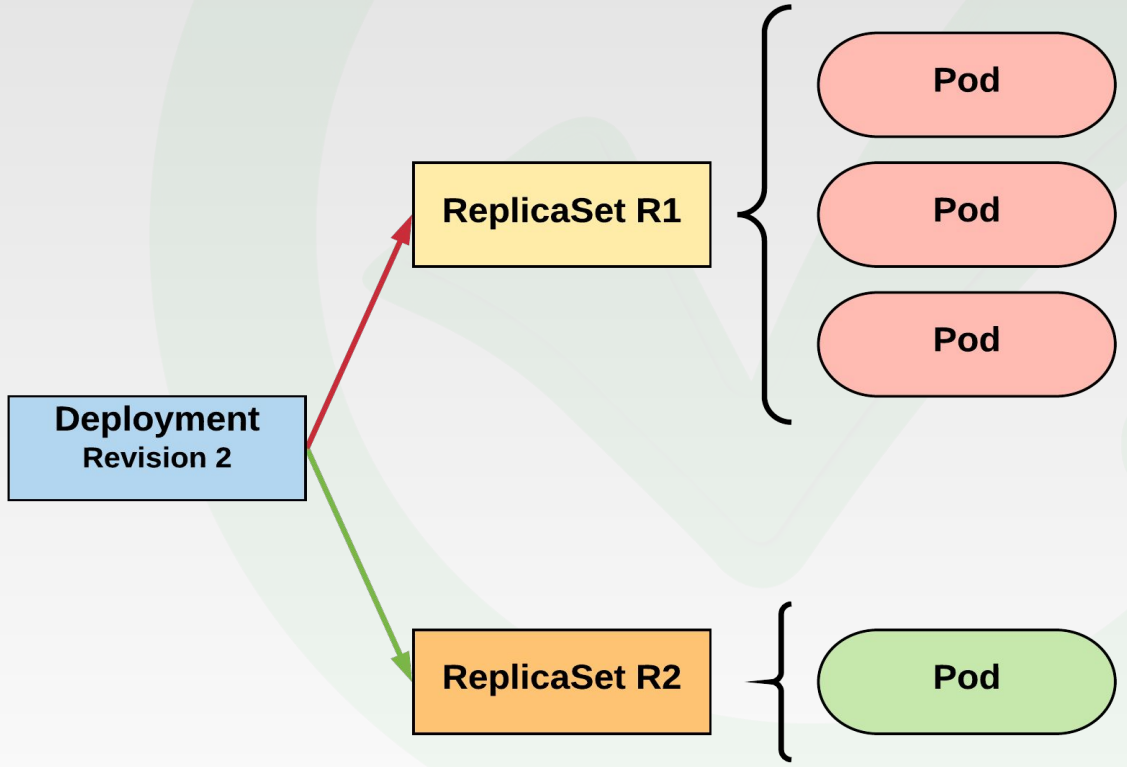
Rolling Update Deployment

New **ReplicaSet** is initially scaled up based on `maxSurge`.

R1 pod-template-hash:
`676677fff`
R2 pod-template-hash:
`54f7ff7d6d`

```
$ kubectl get replicaset
NAME                    DESIRED   CURRENT   READY     AGE
mydep-54f7ff7d6d        1         1         1         5s
mydep-6766777fff        2         3         3         5h
```

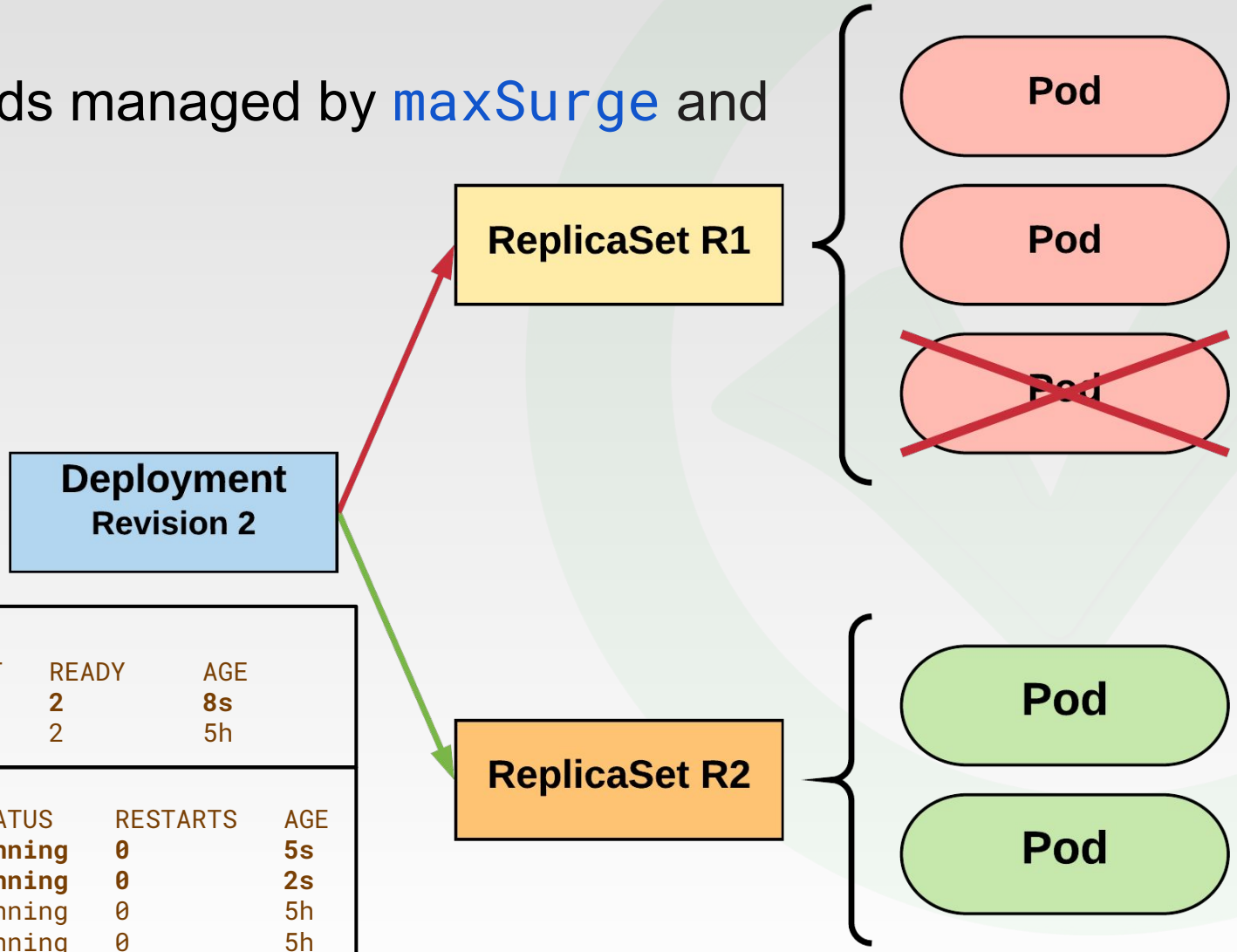
```
$ kubectl get pods
NAME                    READY     STATUS    RESTARTS   AGE
mydep-54f7ff7d6d-9gvll  1/1       Running   0          2s
mydep-6766777fff-9r2zn  1/1       Running   0          5h
mydep-6766777fff-hsfz9  1/1       Running   0          5h
mydep-6766777fff-sjxhf  1/1       Running   0          5h
```



Rolling Update Deployment

Phase out of old Pods managed by `maxSurge` and `maxUnavailable`.

R1 pod-template-hash:
`676677fff`
R2 pod-template-hash:
`54f7ff7d6d`




```
$ kubectl get replicaset
NAME                  DESIRED   CURRENT   READY     AGE
mydep-54f7ff7d6d      2         2         2         8s
mydep-6766777fff      2         2         2         5h
```

```
$ kubectl get pods
NAME                        READY     STATUS    RESTARTS   AGE
mydep-54f7ff7d6d-9gvll      1/1       Running   0          5s
mydep-54f7ff7d6d-cqvlq      1/1       Running   0          2s
mydep-6766777fff-9r2zn      1/1       Running   0          5h
mydep-6766777fff-hsfz9      1/1       Running   0          5h
```

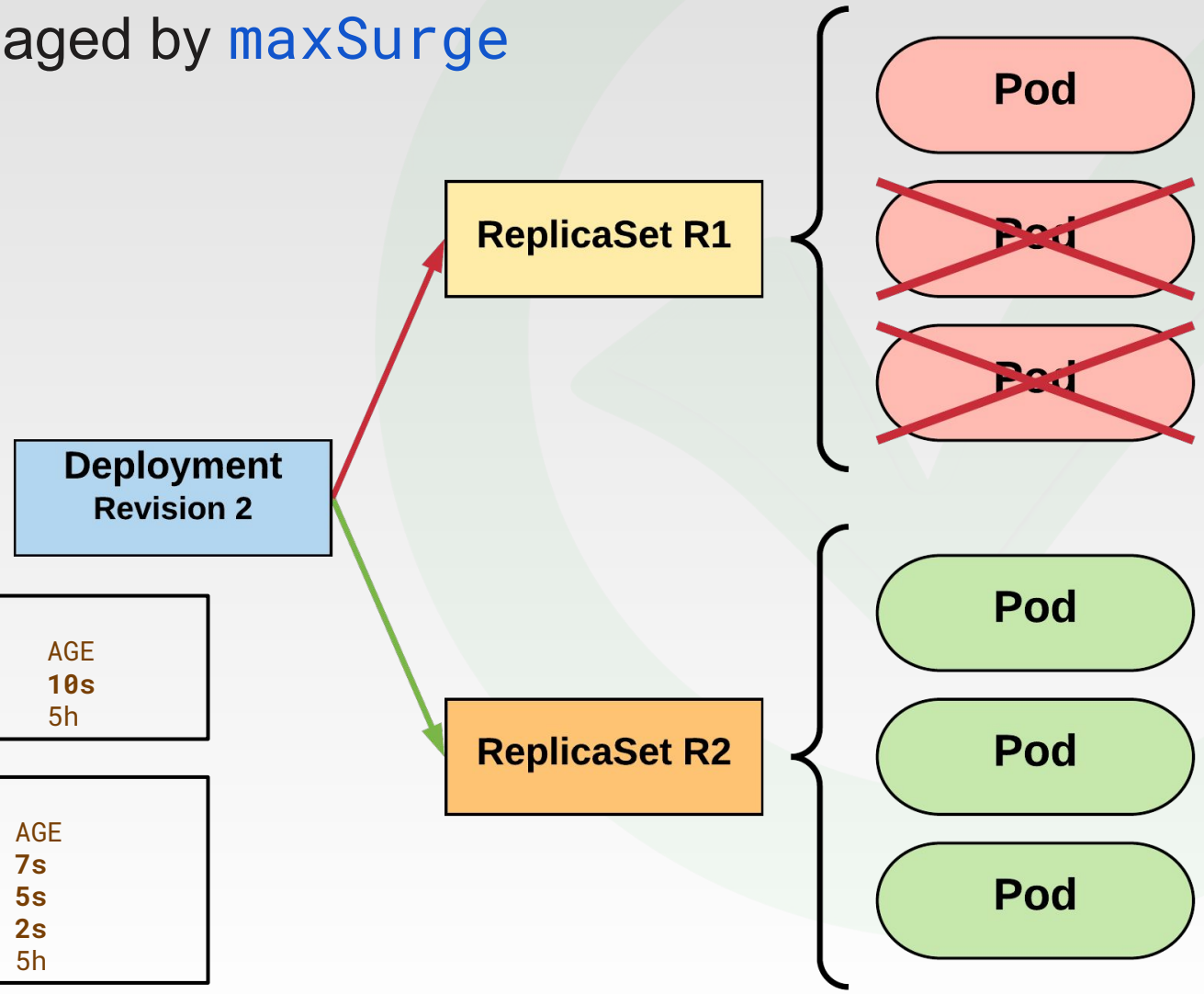
Rolling Update Deployment

Phase out of old Pods managed by `maxSurge` and `maxUnavailable`.

R1 pod-template-hash:
`676677fff`
R2 pod-template-hash:
`54f7ff7d6d`

```
$ kubectl get replicaset
NAME                  DESIRED   CURRENT   READY     AGE
mydep-54f7ff7d6d      3         3         3         10s
mydep-6766777fff      0         1         1         5h
```

```
$ kubectl get pods
NAME                  READY     STATUS    RESTARTS   AGE
mydep-54f7ff7d6d-9gvll   1/1       Running   0          7s
mydep-54f7ff7d6d-cqvlq   1/1       Running   0          5s
mydep-54f7ff7d6d-gccr6   1/1       Running   0          2s
mydep-6766777fff-9r2zn   1/1       Running   0          5h
```



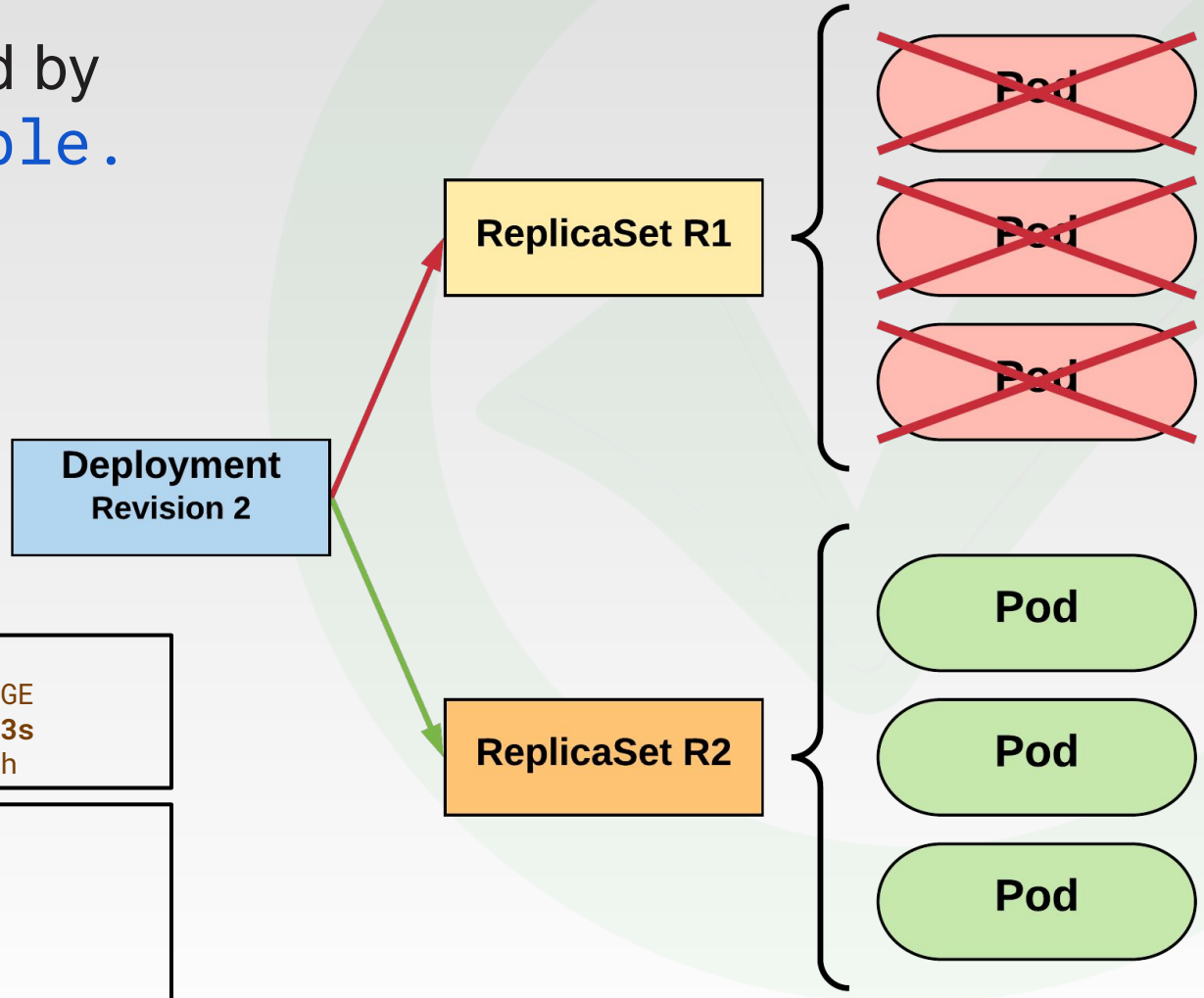
Rolling Update Deployment

Phase out of old Pods managed by `maxSurge` and `maxUnavailable`.

R1 pod-template-hash:
676677fff
R2 pod-template-hash:
54f7ff7d6d

```
$ kubectl get replicaset
NAME                    DESIRED   CURRENT   READY     AGE
mydep-54f7ff7d6d        3         3         3         13s
mydep-6766777fff        0         0         0         5h
```

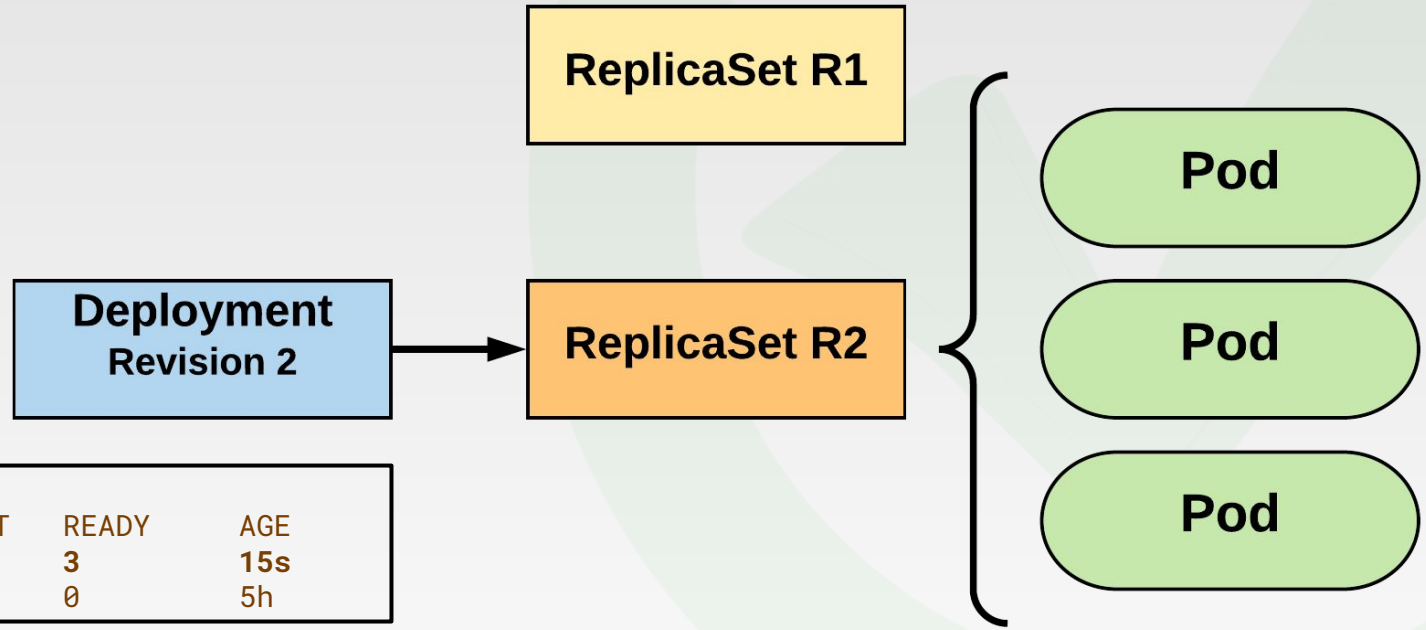
```
$ kubectl get pods
NAME                    READY     STATUS    RESTARTS   AGE
mydep-54f7ff7d6d-9gvll  1/1       Running   0          10s
mydep-54f7ff7d6d-cqvlq  1/1       Running   0          8s
mydep-54f7ff7d6d-gccr6  1/1       Running   0          5s
```



RollingUpdate Deployment

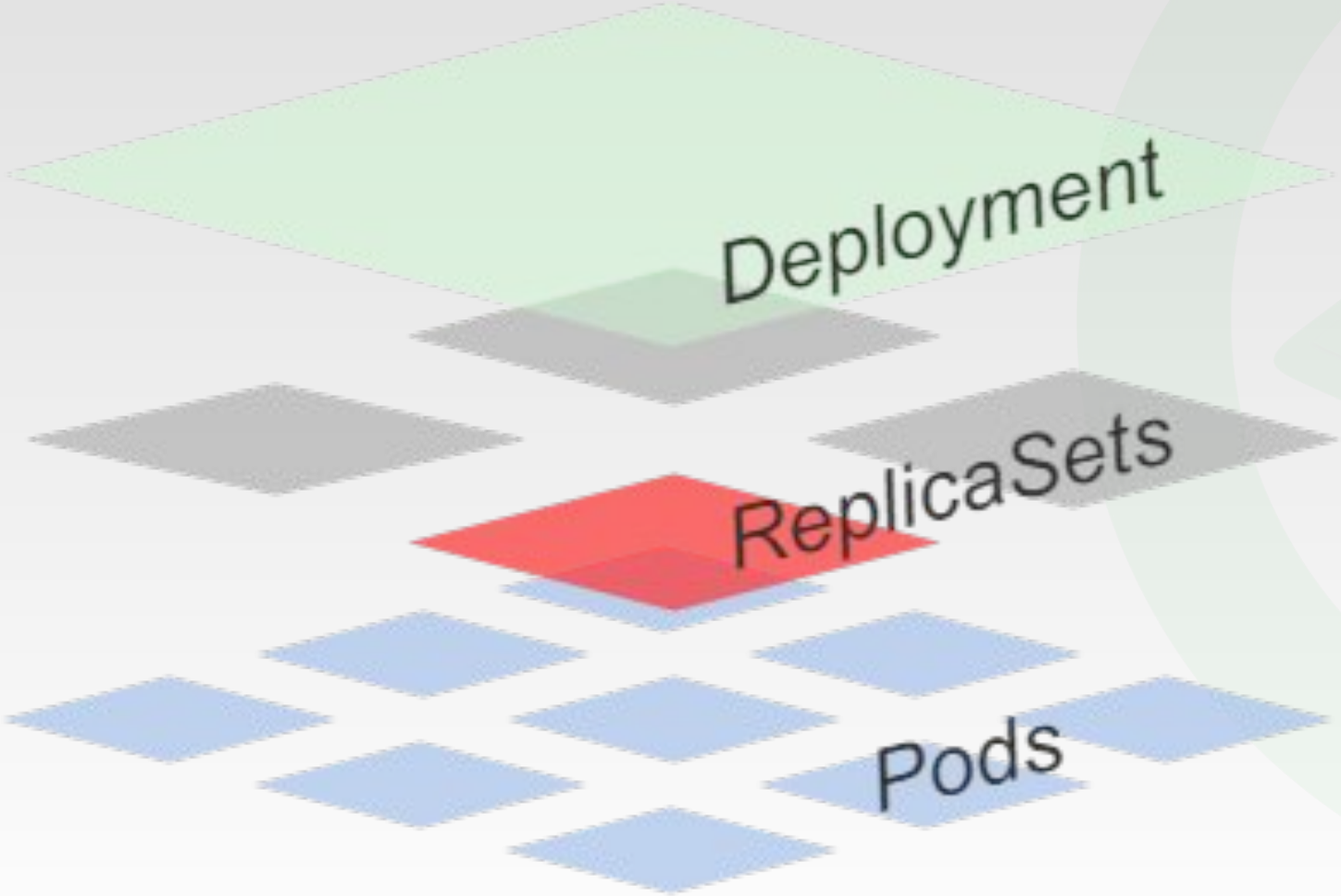
Updated to new deployment revision completed.

R1 pod-template-hash:
676677fff
R2 pod-template-hash:
54f7ff7d6d




```
$ kubectl get replicaset
NAME                          DESIRED   CURRENT   READY     AGE
mydep-54f7ff7d6d              3         3         3         15s
mydep-6766777fff              0         0         0         5h
```

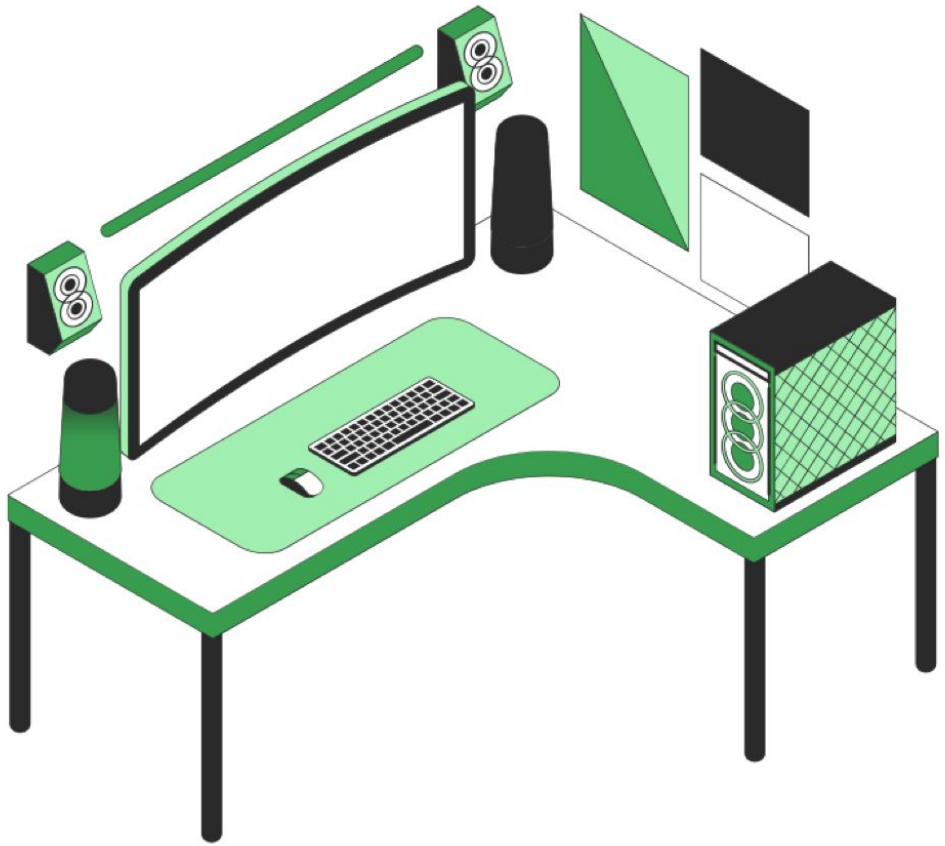
```
$ kubectl get pods
NAME                          READY     STATUS    RESTARTS   AGE
mydep-54f7ff7d6d-9gvll        1/1       Running   0          12s
mydep-54f7ff7d6d-cqvlq        1/1       Running   0          10s
mydep-54f7ff7d6d-gccr6        1/1       Running   0          7s
```


Deployment
ReplicaSets
Pods

Summary

In a K8s cluster:

○ I need to start a containerized app. Solution: use ..	Pod
○ I need to launch a containerized app of 10 replicas. Solution: use ..	ReplicaSet
○ I need to configure updates on a containerized app. Solution: use ..	Deployment
○ I need to make sure a specific group of Pods are run on a specific node.	Labels, Selectors
○ I need to separate resources of Java Dev from the rest of the team.	Namespaces



Do you have any questions?

Send it to us! We hope you learned something new.