Chapter 8 Kubernetes Building Blocks

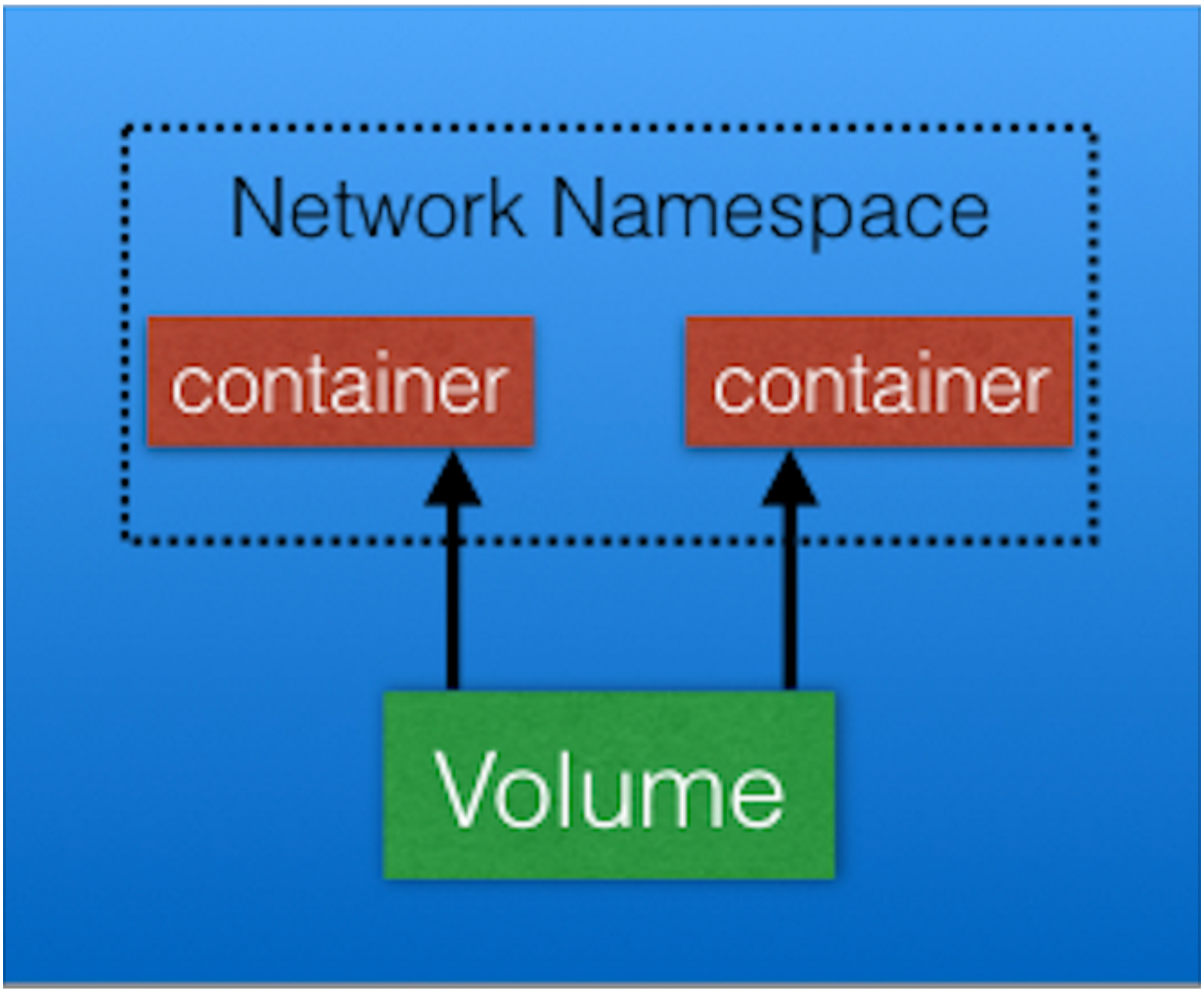
Config

- `spec` has user's intent/desired state
  - API request to create an object must have this section
- K8s system manages the `status` section for objects where it records the actual state of the object
- Request in YAML ➡ `kubectl` ➡ JSON

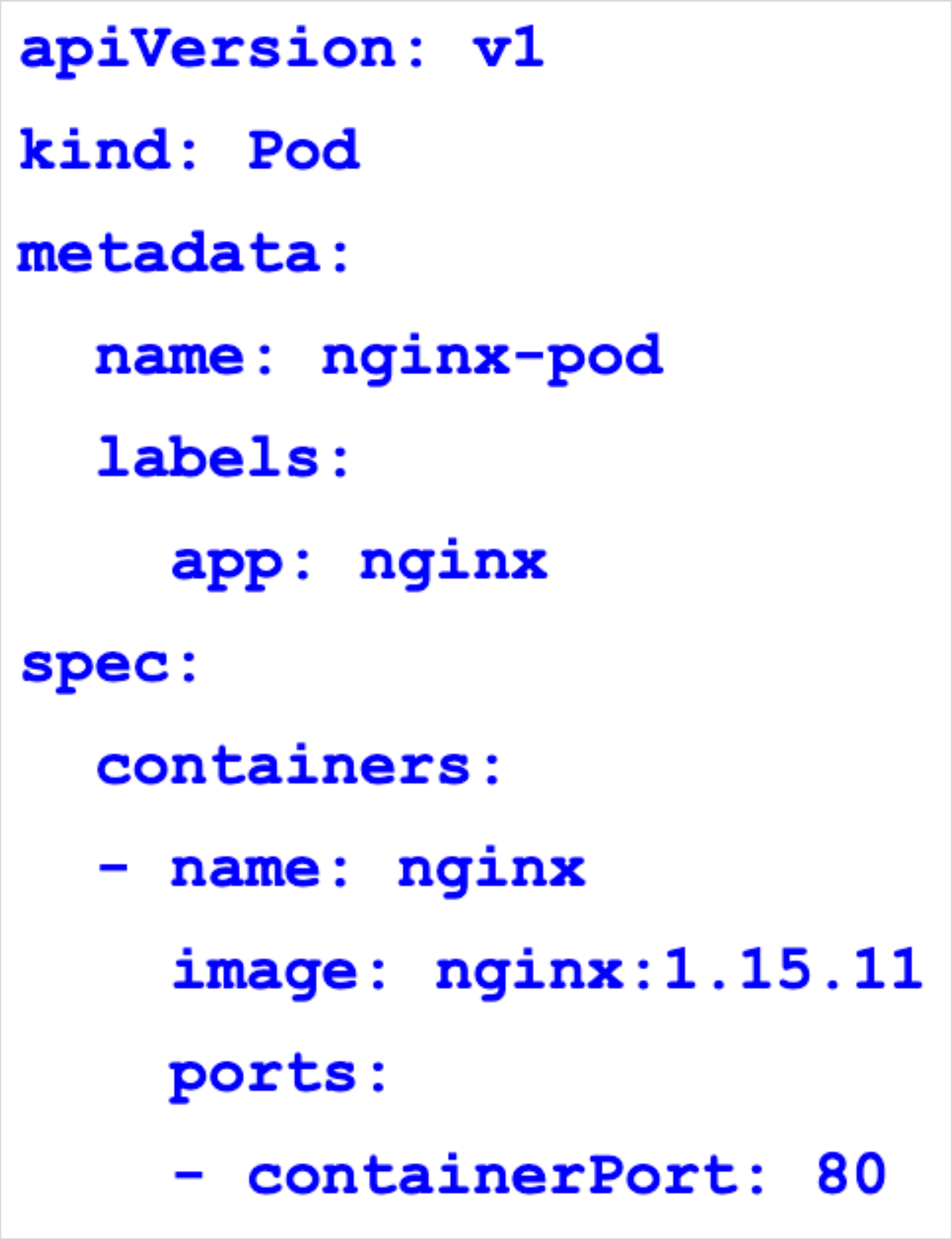
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.11
        ports:
```

- `apiVersion` - API endpoint to connect to
- `kind` - object type (Deployment, Pod, Replicaset, Namespace, Service,...)
- `metadata` - object's basic information (name, labels, namespace,...)
- `spec` - desired state of the object declared in `kind`
- `spec.template` - define spec of Pods

Pods



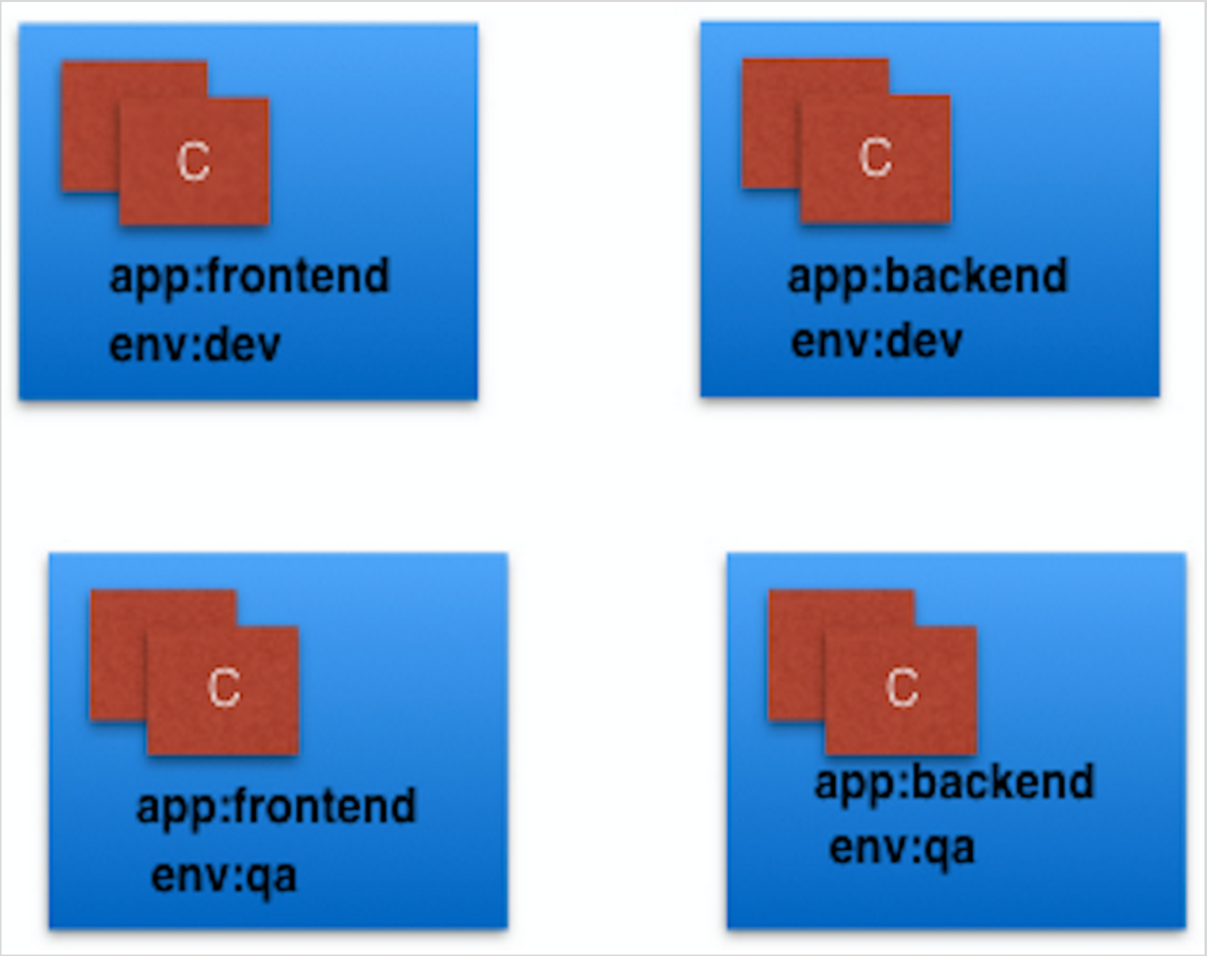
- Logical collection of 1+ containers
- Containers inside a pod
  - are scheduled together on the same host with the Pod
  - share the same network namespace
  - have access to mount the same external storage (volumes)
- Cannot self-heal
- Used with controllers which handles pods' replication, fault tolerance, self-healing, etc.
  - Deployments, ReplicaSets, ReplicationContollers, etc.
  - nested Pod's specification to a controller object using Pod Template



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.15.11
    ports:
    - containerPort: 80
```

- `apiVersion` - v1 for Pod

Labels



- key-value pairs attached to K8s objects
- Organize and select a subset of objects
- Do not provide uniqueness
- Controllers use Labels instead of names/IDs to logically group together decoupled objects

Label Selectors

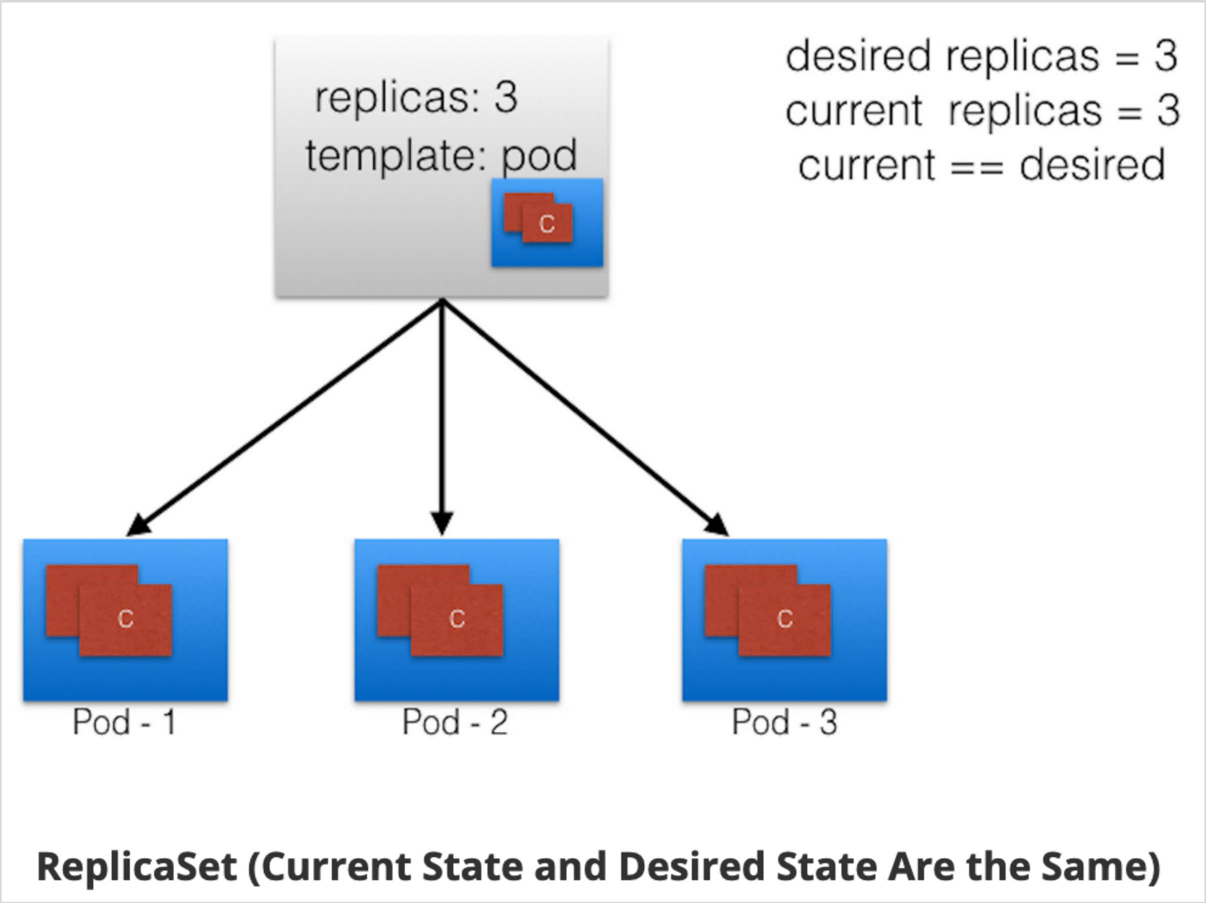
- used to select a subset of objects
- Equality-Based Selectors
  - filtering of objects based on label keys/values
  - `env=dev` (same as `env==dev`)
- Set-Based Selectors
  - filtering of objects based on a set of values
  - `in` `notin` ➜ values
    - `env in (dev,qa)`
  - `exist` `does not exist` ➜ keys
    - `!app`

ReplicationControllers (deprecated)

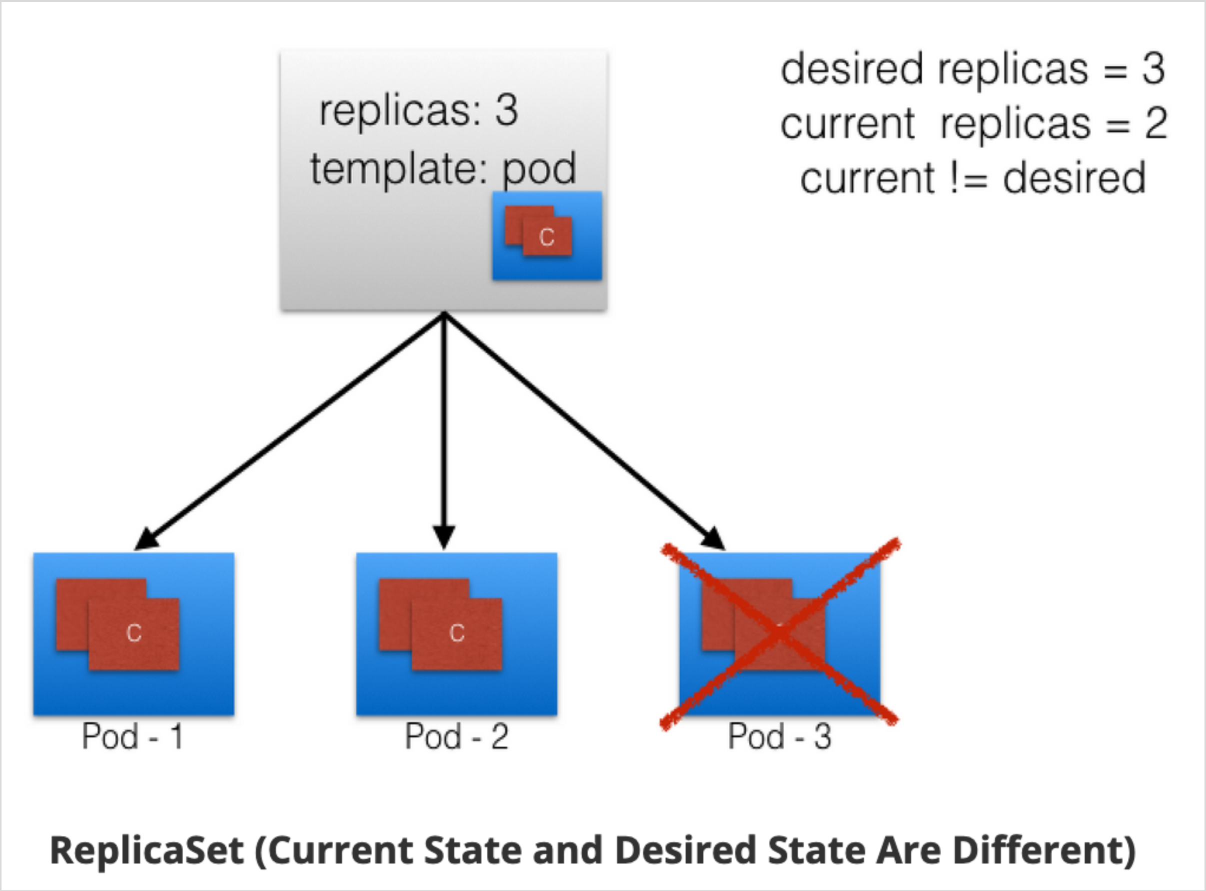
- ensures a specific number of replicas of a Pod is running at any given time
- terminate and create pods to match desired count
- only support equality-based Selectors

ReplicaSets

- next-gen ReplicationController
- support equality- and set-based Selectors
- helps with scaling the number of Pods (using an autoscaler or manually)

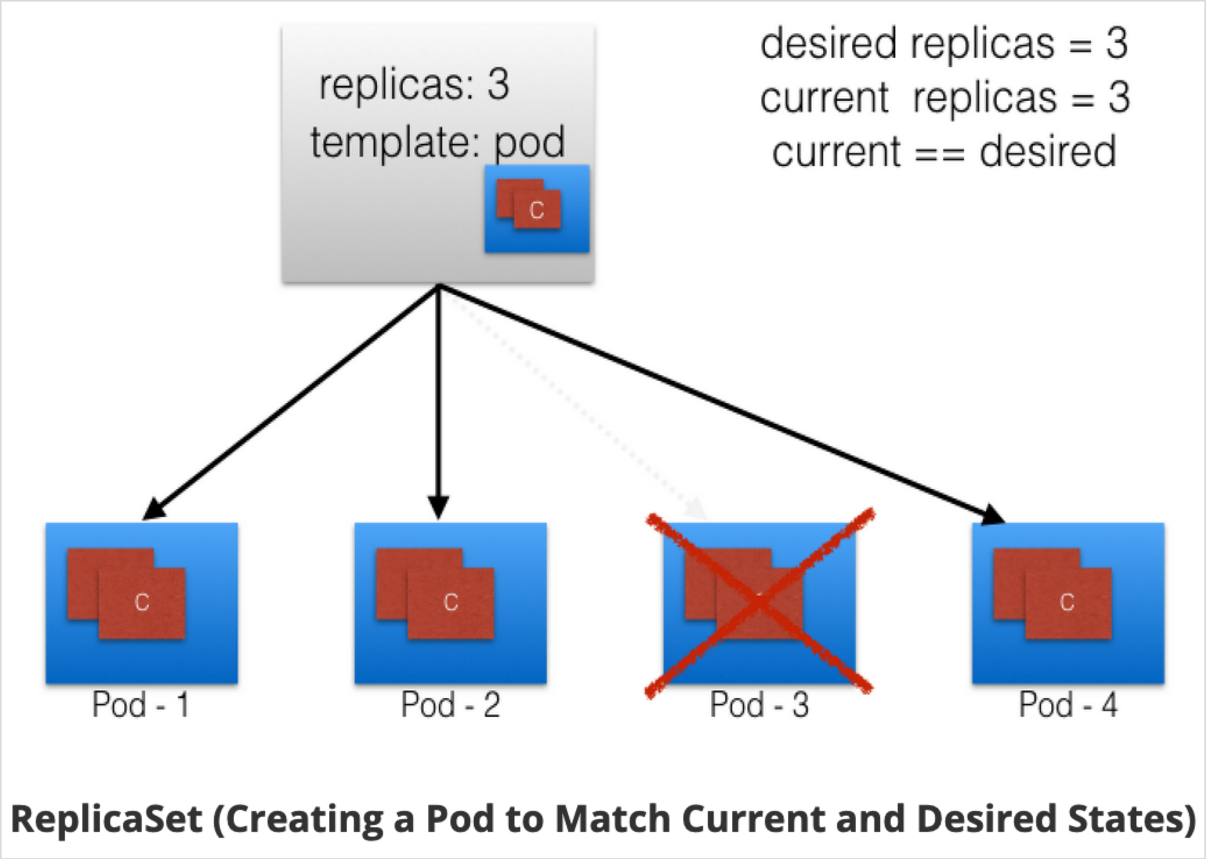


Desired state = 3 replicas



ReplicaSet (Current State and Desired State Are Different)

ReplicaSet will detect the mismatch



ReplicaSet (Creating a Pod to Match Current and Desired States)

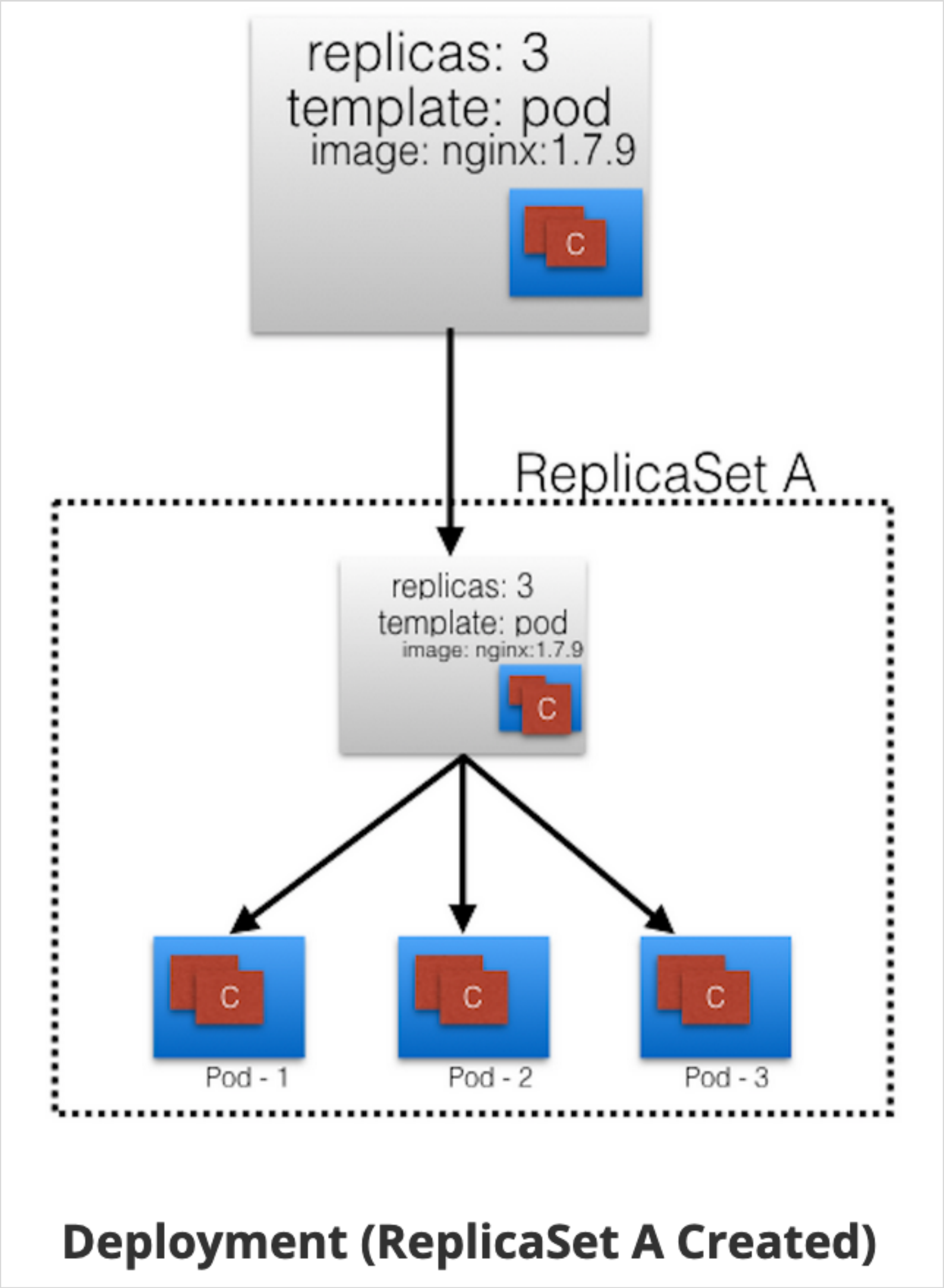
and create an additional Pod

- Deployments manage the creation/deletion/updates of Pods

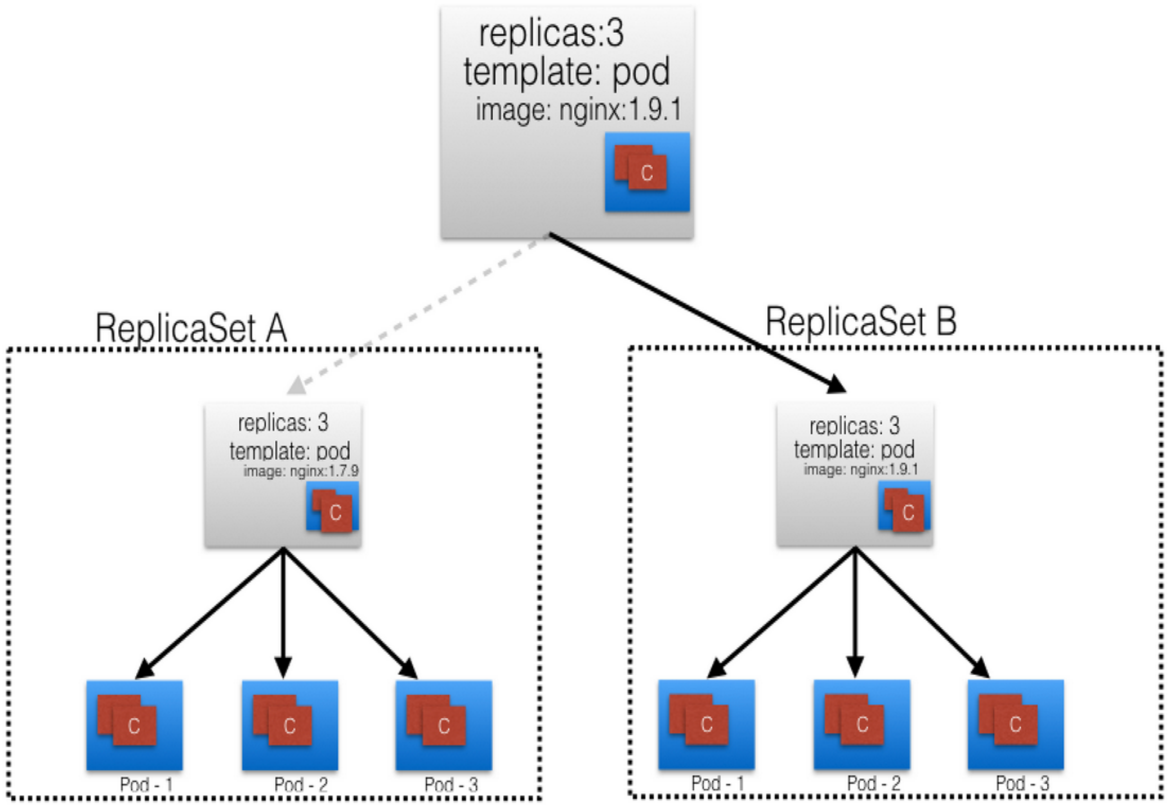
  - automatically creates a ReplicaSet, which creates a Pod
  - no need to manage ReplicaSets and Pods separately because Deployment will manage

Deployments

- provide declarative updates to Pods and ReplicaSets
- DeploymentController is part of the master node's controller manager
  - Checks `current_state == desired_state`
- allows seamless updates `rollouts` and downgrades `rollbacks`
- directly manages its ReplicaSets for scaling



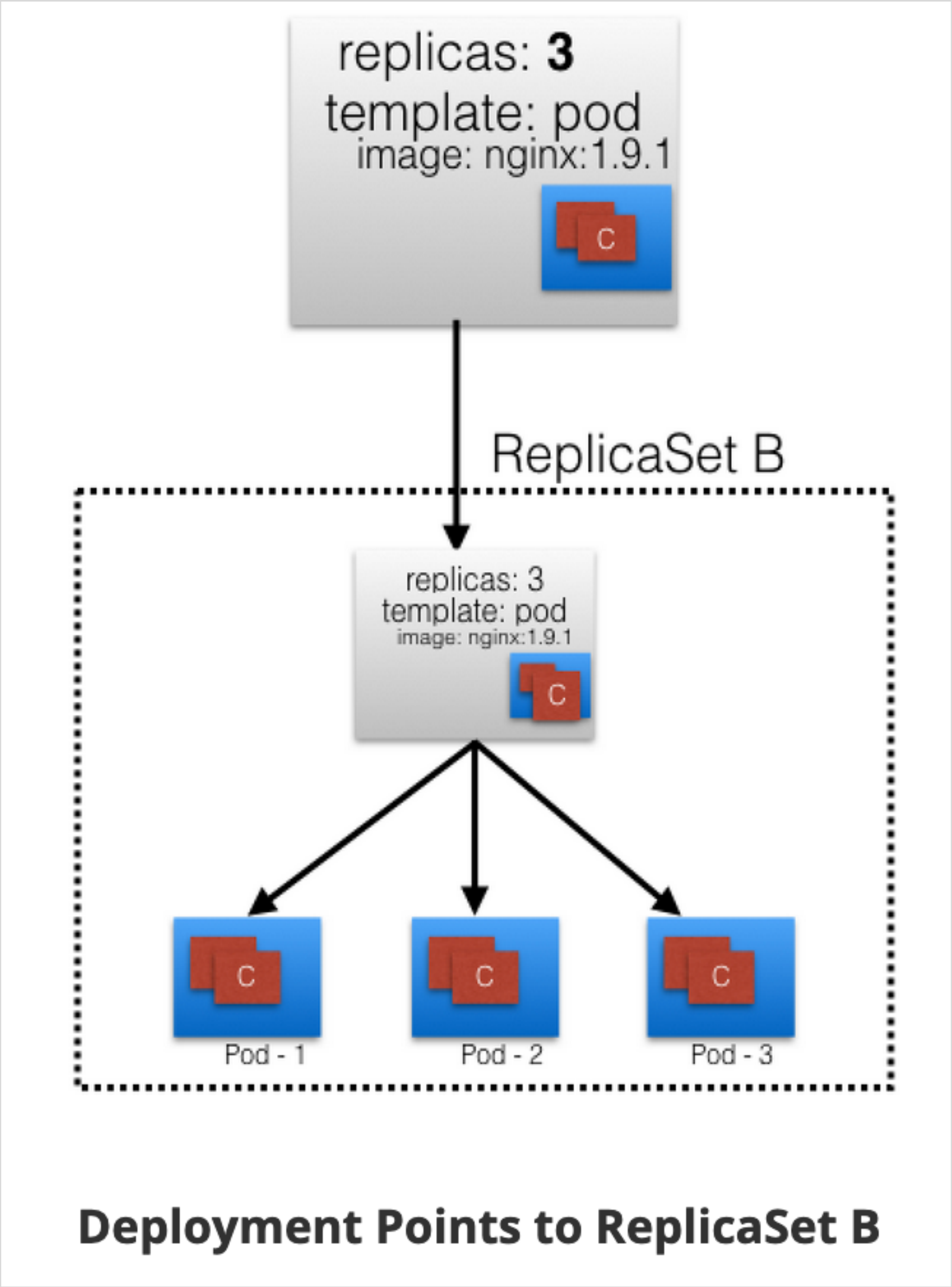
ReplicaSet for image "nginx: 1.7.9" (Revision 1)



Deployment (ReplicaSet B Created)

Rolling update to image "nginx: 1.9.1" (Revision 2)

- Rolling update changes Revision number
- Scaling or labeling deployment do not trigger a rolling update
- Once the rolling update has completed, Deployment will show both ReplicaSets A and B, where A is scaled to 0 pods, and B is scaled to 3 Pods.



Deployment pointing to ReplicaSet 2

Namespaces

- For partitioning a cluster into sub-clusters for multi-user/team use cases

- Names of resources/objects in a Namespace are unique (not across Namespaces in the cluster)
- `kubectl get namespaces`
- Kubernetes creates four default Namespaces:
  - `kube-system`
    - objects created by the K8s system, i.e. control plane agents
  - `default`
    - objects/resources created by admins/devs
    - default connection
  - `kube-public`
    - unsecured, readable by anyone
  - `kube-node-lease`
    - node lease objects used for node heartbeat data

Demo

LinuxFoundationXLFS158x-…
Aug 18, 2019

```
kubectl create deployment mynginx --image=nginx:1.15-alpine
kubectl get deploy,rs,po -l app=mynginx
kubectl scale deploy mynginx --replicas=3
kubectl describe deployment mynginx
kubectl rollout history deploy mynginx
kubectl set image deployment mynginx nginx=nginx:1.16-alpine
kubectl rollout undo deployment mynginx --to-revision=1
```