Kubernetes 101 Tutorial - 1

05/21/2023

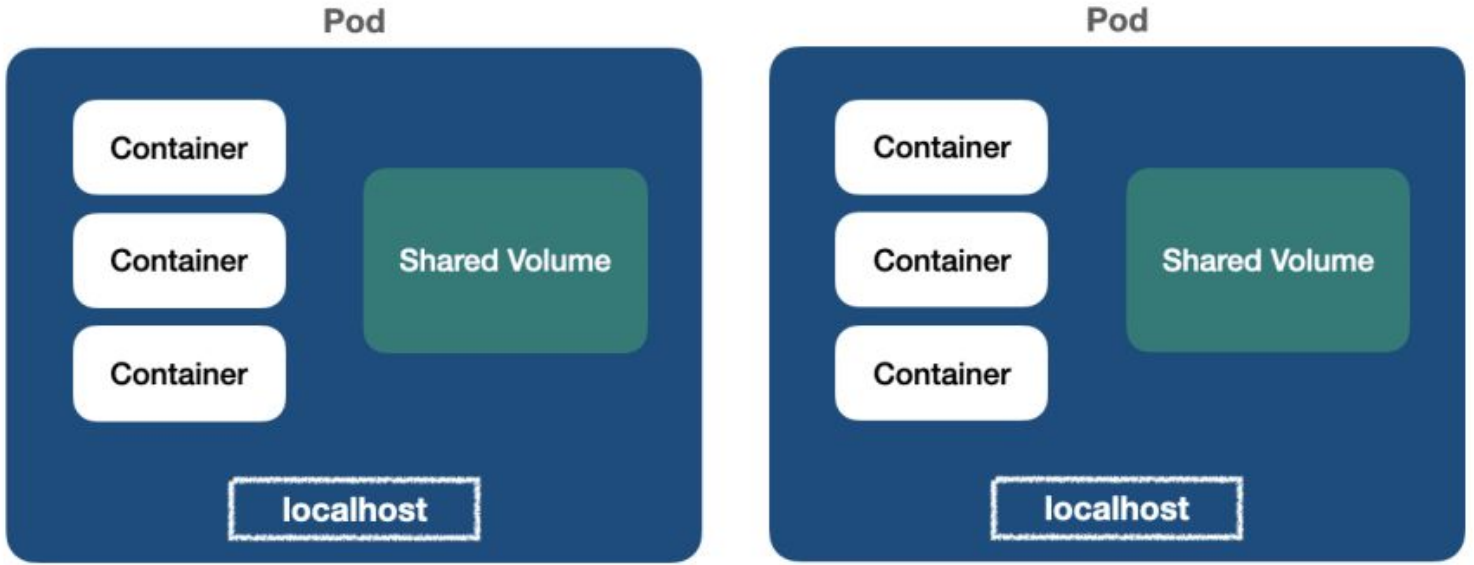
A wrapper for containers

In Kubernetes, we aren't able to create single containers directly. Instead, for the better, we can wrap containers into a single unit which comprises:

- **a specification**: where multiple containers can use the same specification as deployable units
- **a shared storage**: they can use a shared storage so the same volumes are mounted across multiple containers
- **a single network**: containers under the same wrapper can share a single network, so they can communicate to each other

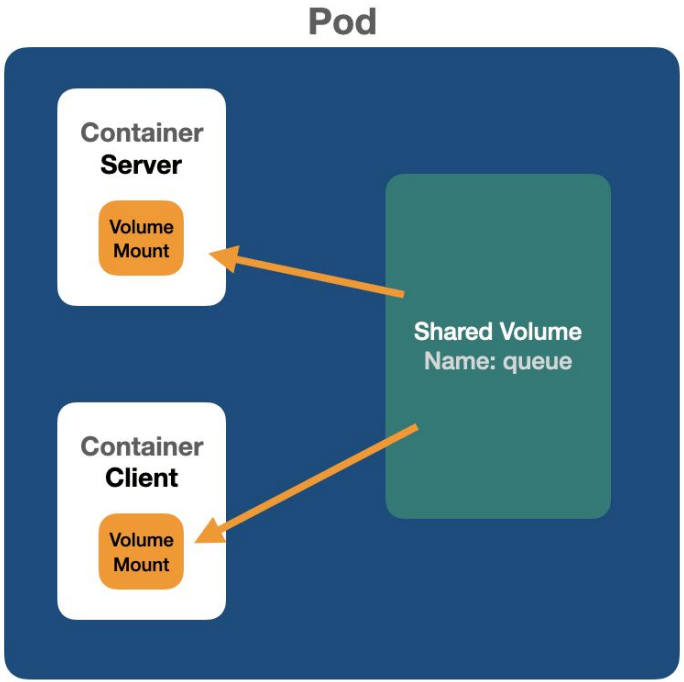
Pods

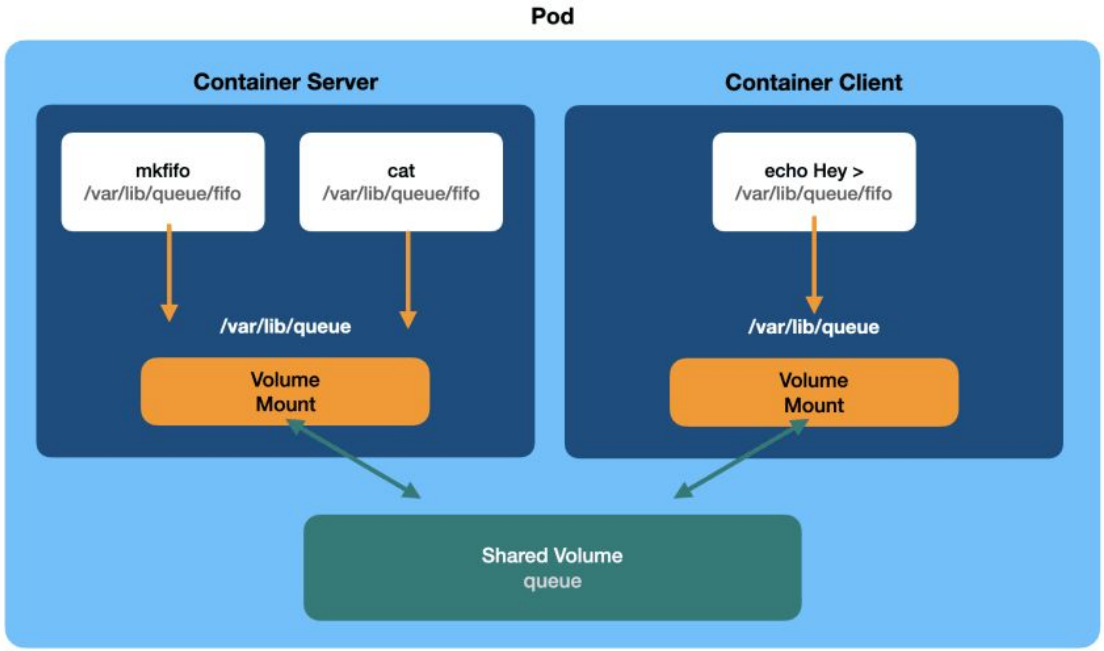
the **smallest deployable unit** you can create and manage in Kubernetes. Within Pods, we can group multiple containers that should communicate to each other somehow, either using the **same network or through shared volumes**.



Creating a pod

kubectl run <container-name> --image=<some_image>



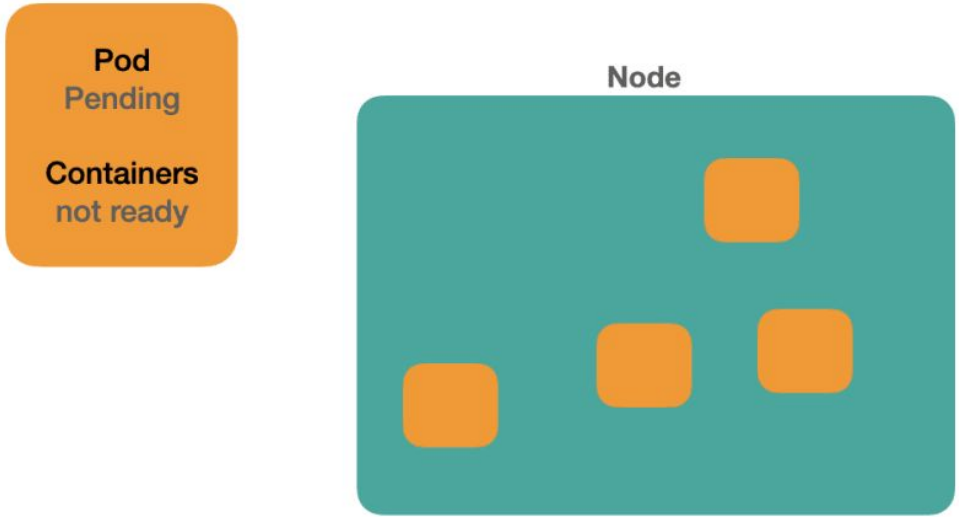


YAML

Introduction: YAML tutorial: Get started in 5 minutes (educative.io)

Pod Lifecycle - Pending

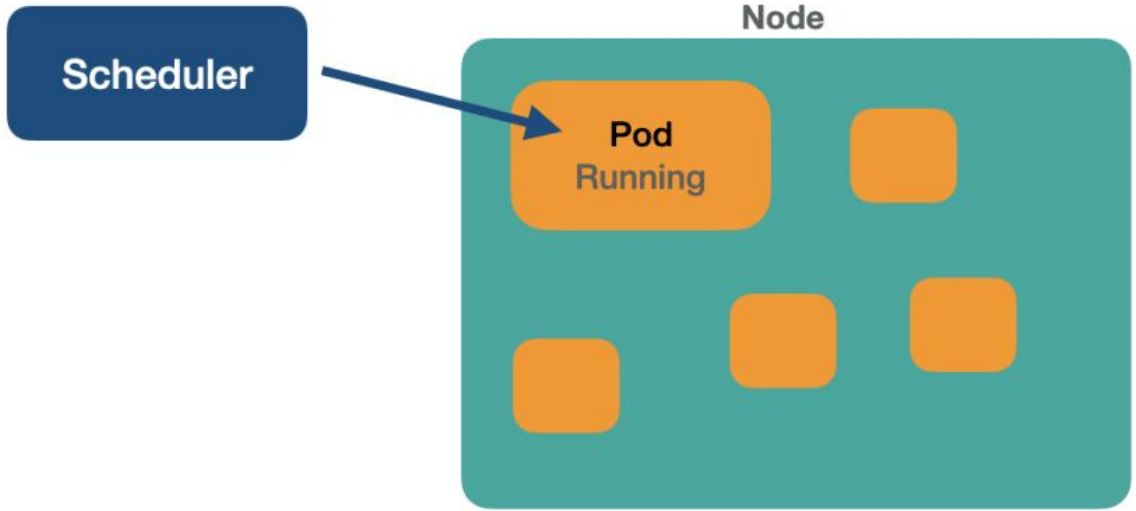
It's when a Pod is **accepted by the cluster** but its *containers are not ready yet*. The Pod is **NOT yet scheduled** to any Node.



Pod Lifecycle - Running

All **containers are created** and the **Pod has been scheduled** to a Node.

At least one of the containers are still running or being started.

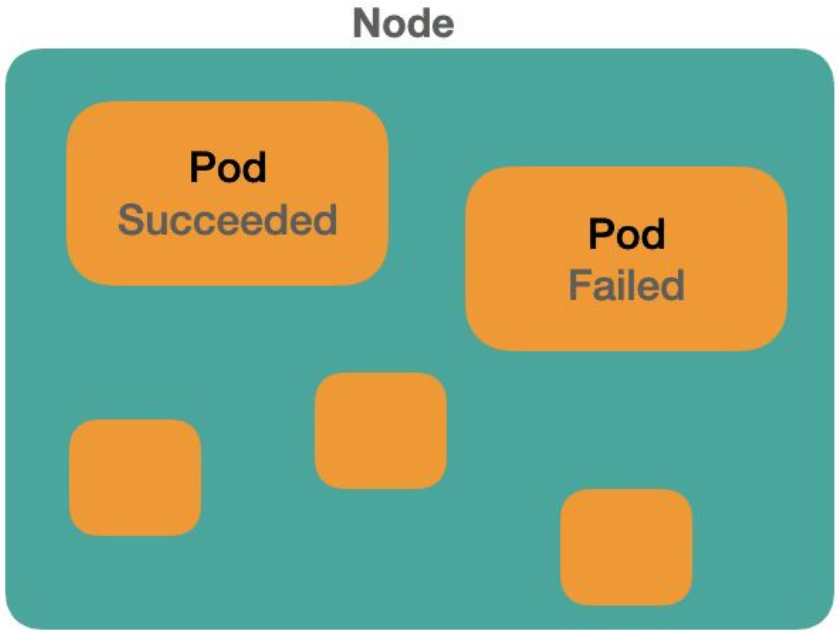


Pod Lifecycle - Terminated / Completed

Indicates that all the containers are `terminated`
(internally by Kubernetes) or `completed`.

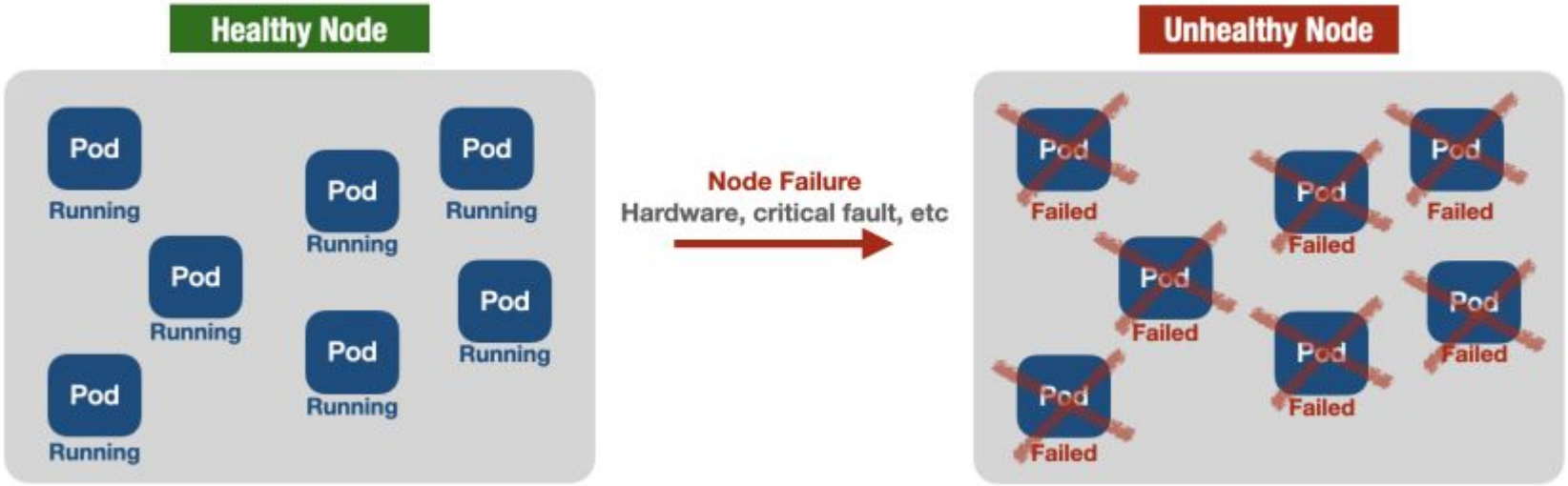
Pod Lifecycle - Succeeded / Failed

If all containers are `Terminated` in success, then the Pod status is `Succeeded`.
But in case all containers have terminated but at least 1 container terminated in failure, the Pod status is `Failed`.



Self healing

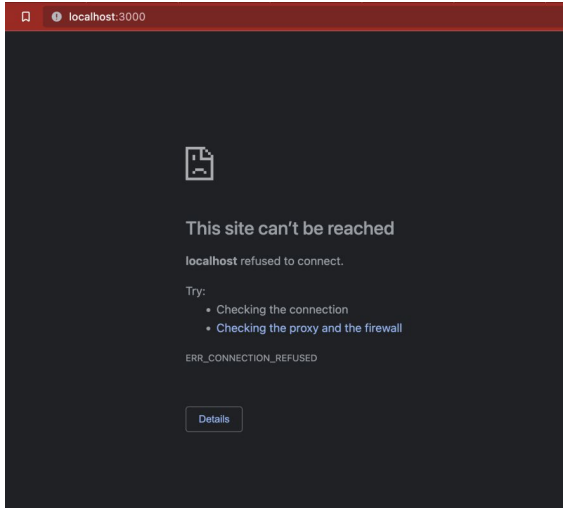
Kubernetes Node: is represented by a virtual machine

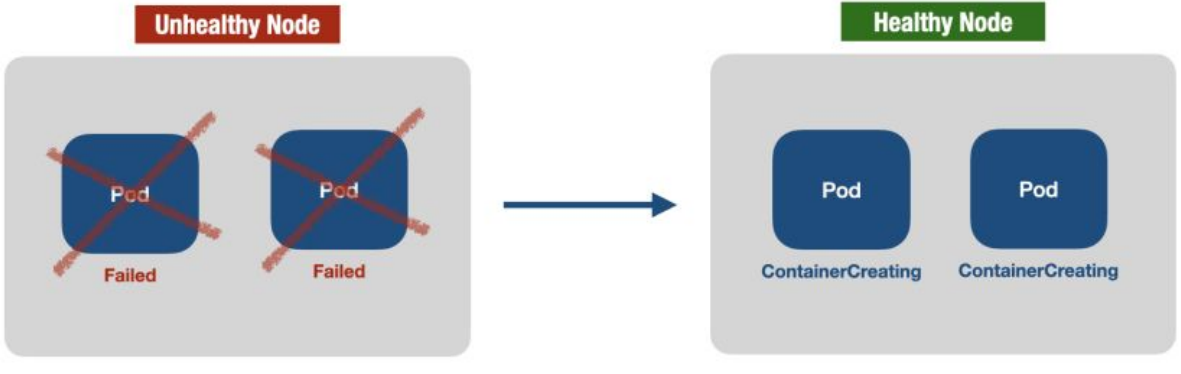


Application Downtime

A new node is required. But provisioning hardware is a *costly* operation, **it takes time**.
Meanwhile, the **Pod remains failed** in the unhealthy node and the application is suffering a **downtime**.
Once the new node *has joined and is ready to accept* new Pods, we can start **all the pods manually** using `kubectl` in the newly healthy node

```
$ kubectl apply -f ./pod.yml
```



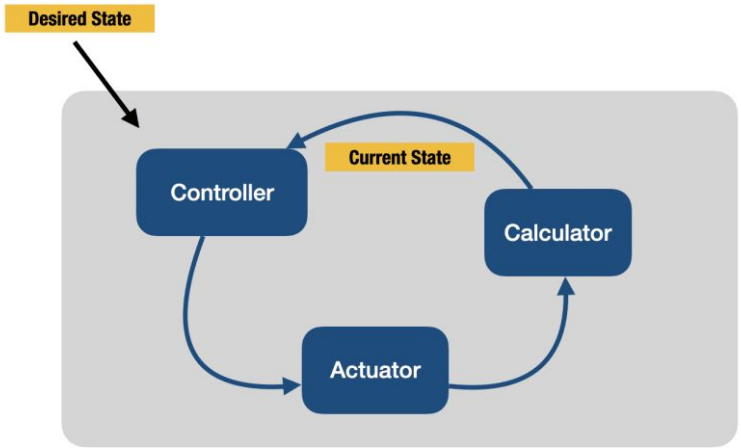


Self-healing system

A system which is capable of **detecting failures** and also *restarting components* or applications **automatically** with no human intervention.

Automation is key. And a potential solution for self-healing comes from **Robotics**.

In Robotics, we usually create a **controller** that gets a **desired state** and, by using some sort of **control loop**, it continuously check if the *current state matches the desired state*, trying to come **closer as much as possible**.



Example: a thermostat

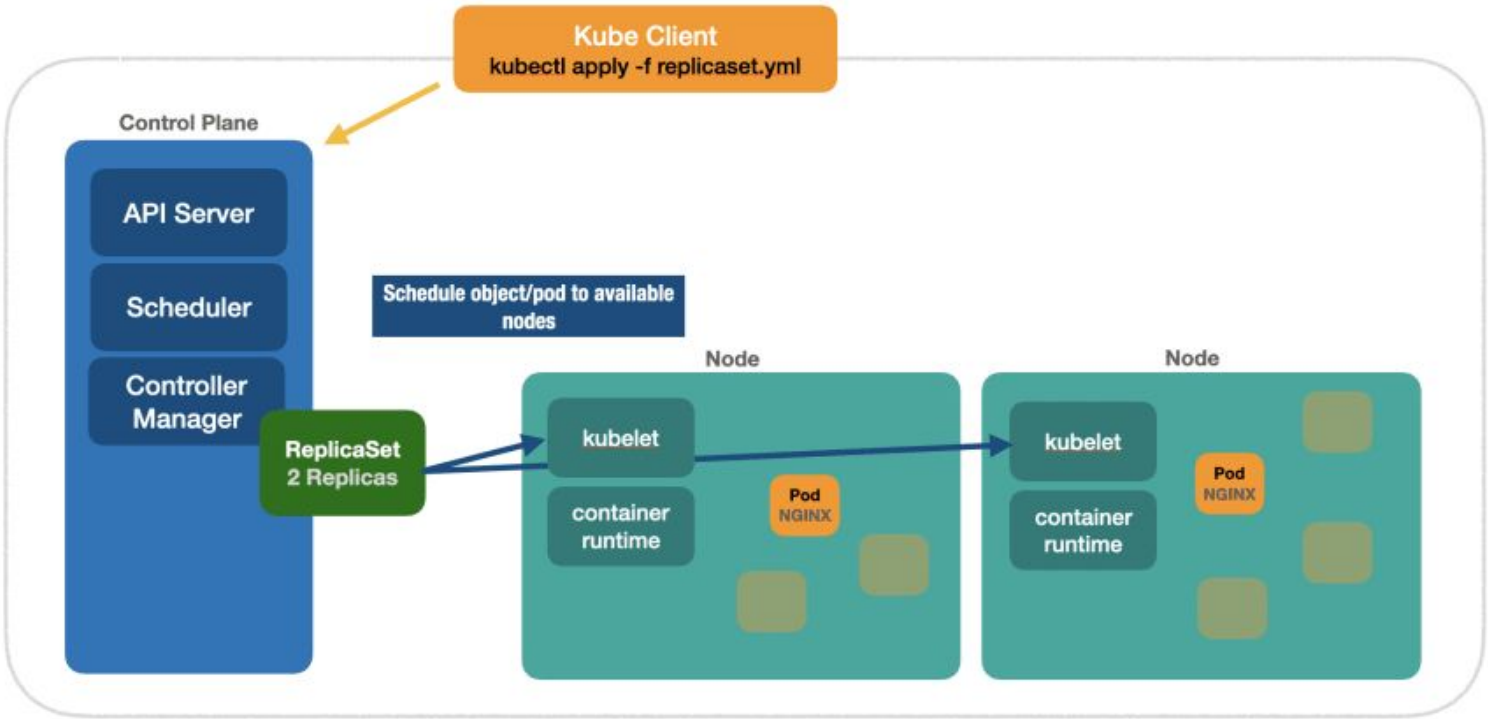
Kubernetes Controllers

Kubernetes controllers are **control loops** that watch the cluster state, then take actions to *match the desired state* as much as possible.
But how do we make use of *controllers*? Kubernetes provides several Workload Resources so we can rely on them to **manage Pods on our behalf**.
Time to explore one of the main *workload resources* that guarantees *self-healing capabilities*, the **ReplicaSet**.

ReplicaSet

```yaml
### The kind of the Kubernetes object
kind: ReplicaSet
apiVersion: apps/v1
metadata:
  name: nginx
spec:
  ### The number of replicas of nginx Pod
  ### The controller will manage the Pods on our behalf
  ### Anytime a Pod goes down, the controller will restart a new one to guarantee that at least 2 nginx Pods are running
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
```

ReplicaSet


Kube Client
kubectl apply -f replicaset.yml
Control Plane
API Server
Scheduler
Controller Manager
ReplicaSet
2 Replicas
Schedule object/pod to available nodes
Node
kubelet
container runtime
Pod
NGINX
Node
kubelet
container runtime
Pod
NGINX

Deleting a pod of ReplicaSet/Deleting ReplicaSet

In case we delete a Pod that was created by a ReplicaSet, the controller will start a **new one automatically.**

But in case we want to delete all Pods of a ReplicaSet, we should delete the `replicaset` instead.

Next

Kubernetes workload resources