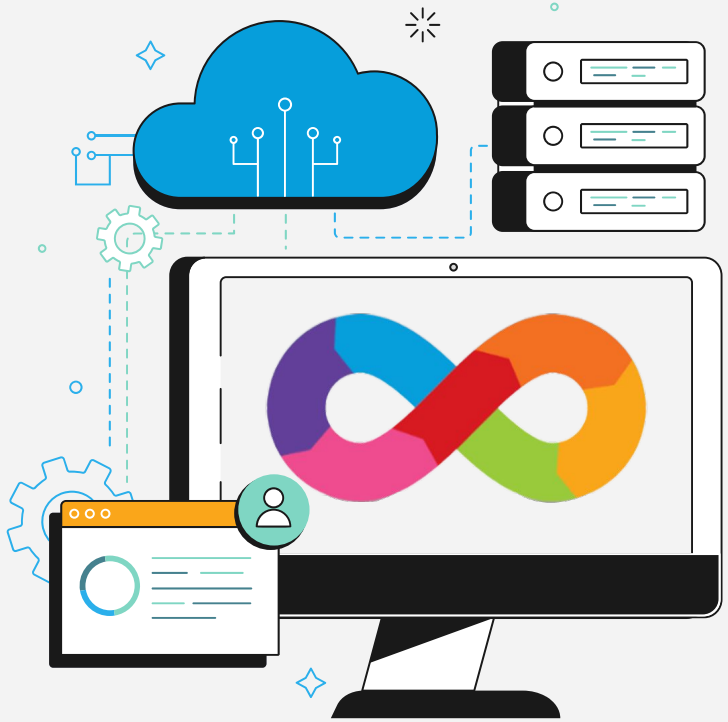
Introduction to DevOps

@ IBA - SMCS

Week 08 - Part 2
K8s Deep Dive



Obaid ur Rehman
Software Architect / Engineering Manager @ Folio3

Kubernetes - Deep Dive

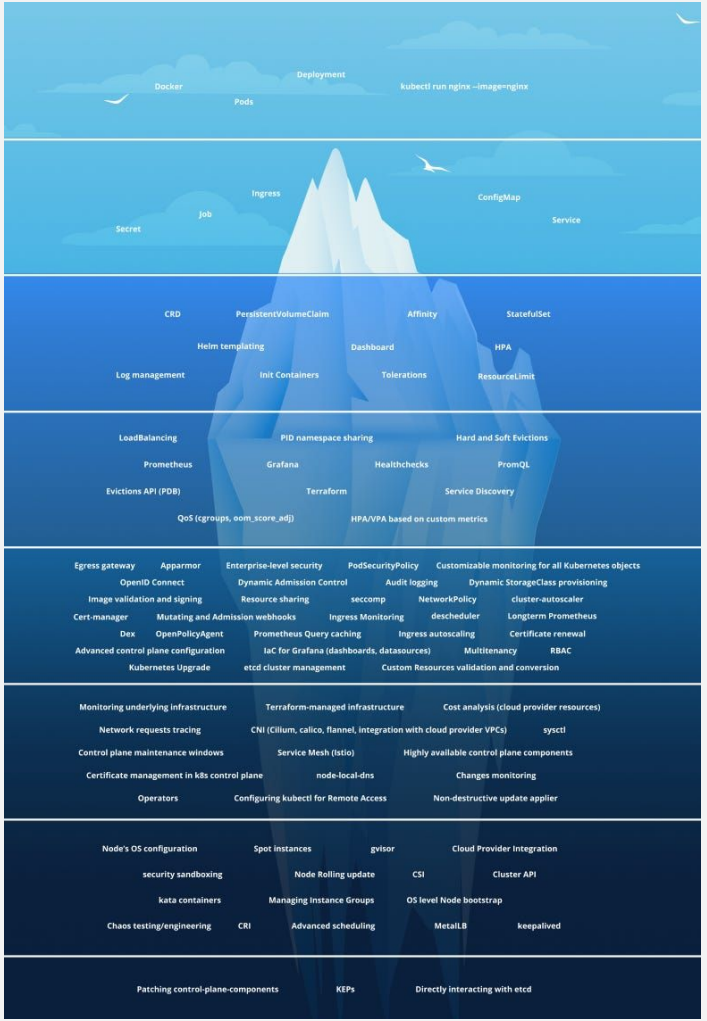
- Jobs & Cron Jobs
- DaemonSet, ReplicaSet, StatefulSet & Deployment
- Ingress
- Persistent Volumes & Volume Claims.
- Memory & CPU Limit on Pods

Kubernetes - Deep Dive

- Horizontal Pod AutoScaling
- Init Containers
- Affinity, Taint & Tolerations
- Probes: Startup, Readiness, liveness probe.
- Rolling Updates

Motivation

Kubernetes is easy 😬



Source: https://anjulsahu.substack.com/p/devops-roadmap-2022-340934d360f9

Job

Kubernetes jobs create one or more pods that will run until execution successfully terminates for a specific number of pods.
Once the task assigned to a job completes without any error, the task(job) stops running. In case of a failure, the job attempts to retry until all pods run successfully. This can be configured.

Use-case:

- Run a one time task to take backup.
- Install anything in cluster.

Job

- Kubectl apply - f .
- Kubectl get jobs
- Kubectl logs jobs/pi
- kubectl delete cronjob hello

https://kubernetes.io/docs/concepts/workloads/controllers/job/

CronJob

If you need to start jobs at a specific time in the future, or you want to run them in a repetitive pattern at specific intervals, you should consider using a **CronJob**.

Use-case:

- Run a daily task to take backup.
- Run a task (like sending daily emails at 12.00 midnight).

Daemonset, ReplicaSet, Deployment & StatefulSet

A Pod is the smallest deployable unit in Kubernetes. Pods hold the containers for an application.

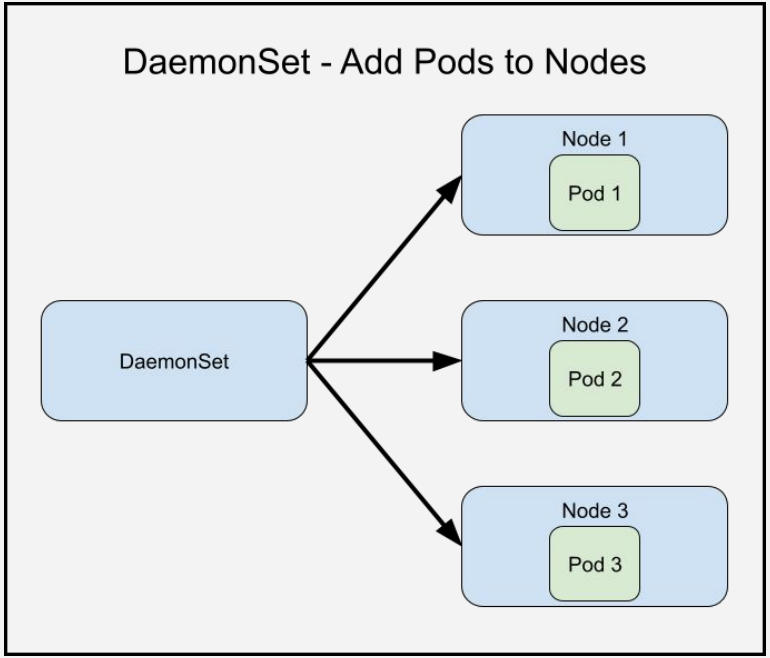
To help deploy Pods, Kubernetes provides these four different options. Let take a look at them one by one.

DaemonSet

Daemonset ensures that the pod runs on all the nodes of the cluster. If a node is added/removed from a cluster, DaemonSet automatically adds/deletes the pod.

Use-case:

- Monitoring Exporters: You would want to monitor all the nodes of your cluster so you will need to run a monitor on all the nodes of the cluster.
- Logs Collection Daemon: You would want to export logs from all nodes so you would need a DaemonSet of log collector like Fluentd to export logs from all your nodes.



ReplicaSet

A ReplicaSet is a Kubernetes object that ensures that a specified number of replicas of a pod are running at any given time.

🔴Not a preferred way, unless the use-case.

https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/

Deployment

A Deployment is a Kubernetes object that manages a set of identical pods, ensuring that a specified number of replicas of the pod are running at any given time.

It provides a declarative approach to managing Kubernetes objects, allowing for automated **rollouts** and **rollbacks** of containerized applications.

🟢Preferred way

ReplicaSet V/S Deployment

Deployment: High-level abstractions that manage replica sets.
It provides additional features such as rolling updates, rollbacks, and versioning of the application.

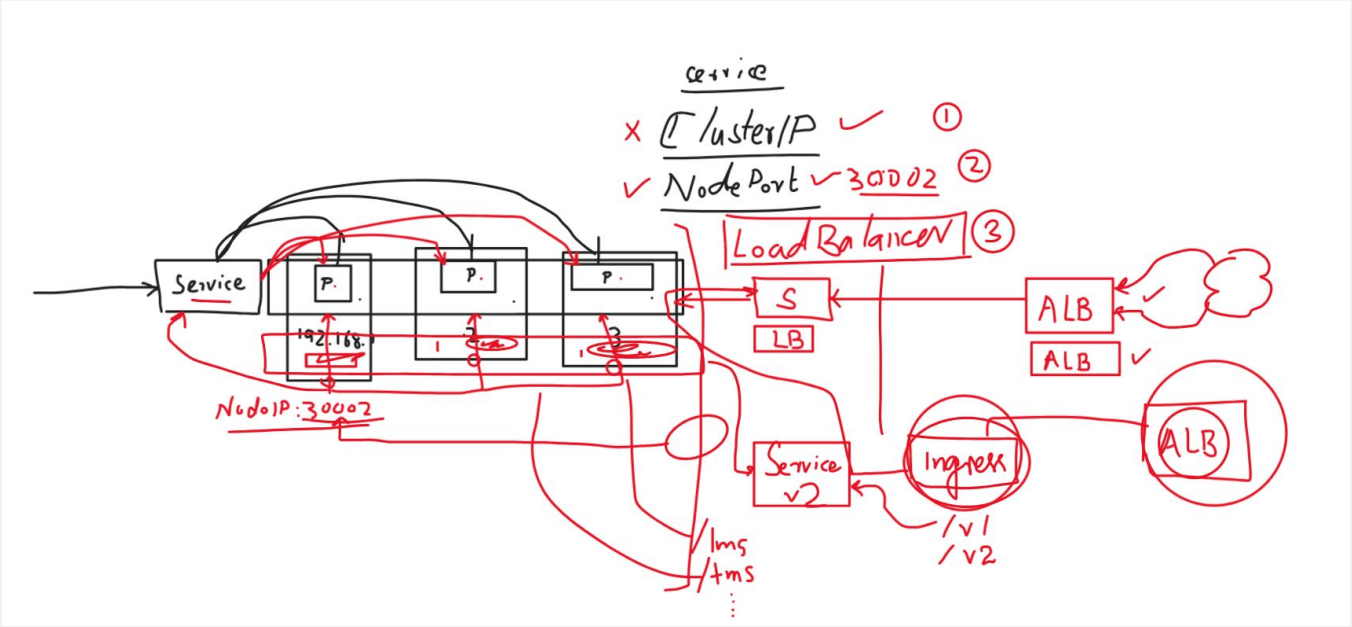
🟢Rolling Updates

ReplicaSet: A lower-level abstraction that manages the desired number of replicas of a pod.
Additionally, it provides basic scaling and self-healing mechanisms.

🔴No Rolling Updates

Ingress

Let's discuss NodePort, ClusterIP & LoadBalancer
Let’s also see how Ingress works.



Persistent Volumes, Persistent Volume Claims

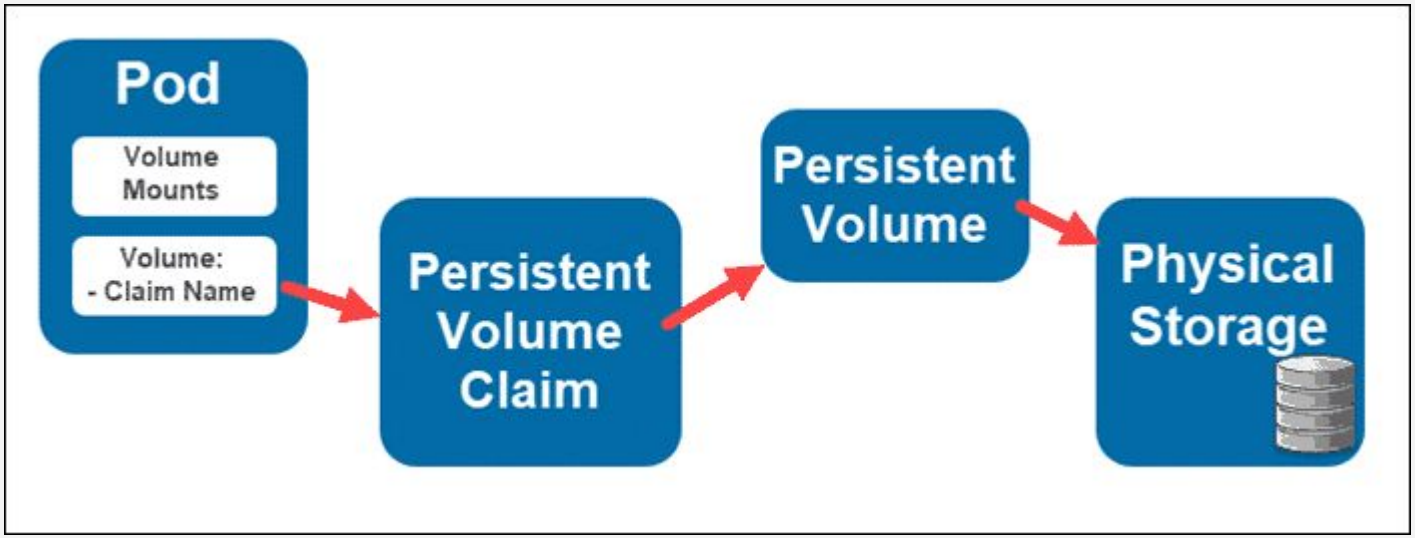
A Kubernetes **persistent volume** (PV) is an object that allows pods to access persistent storage on a storage device, defined via a Kubernetes StorageClass.

PVs must be requested through **persistent volume claims** (PVCs), which are requests for storage. A PVC is essentially a request to mount a PV meeting certain requirements on a pod.

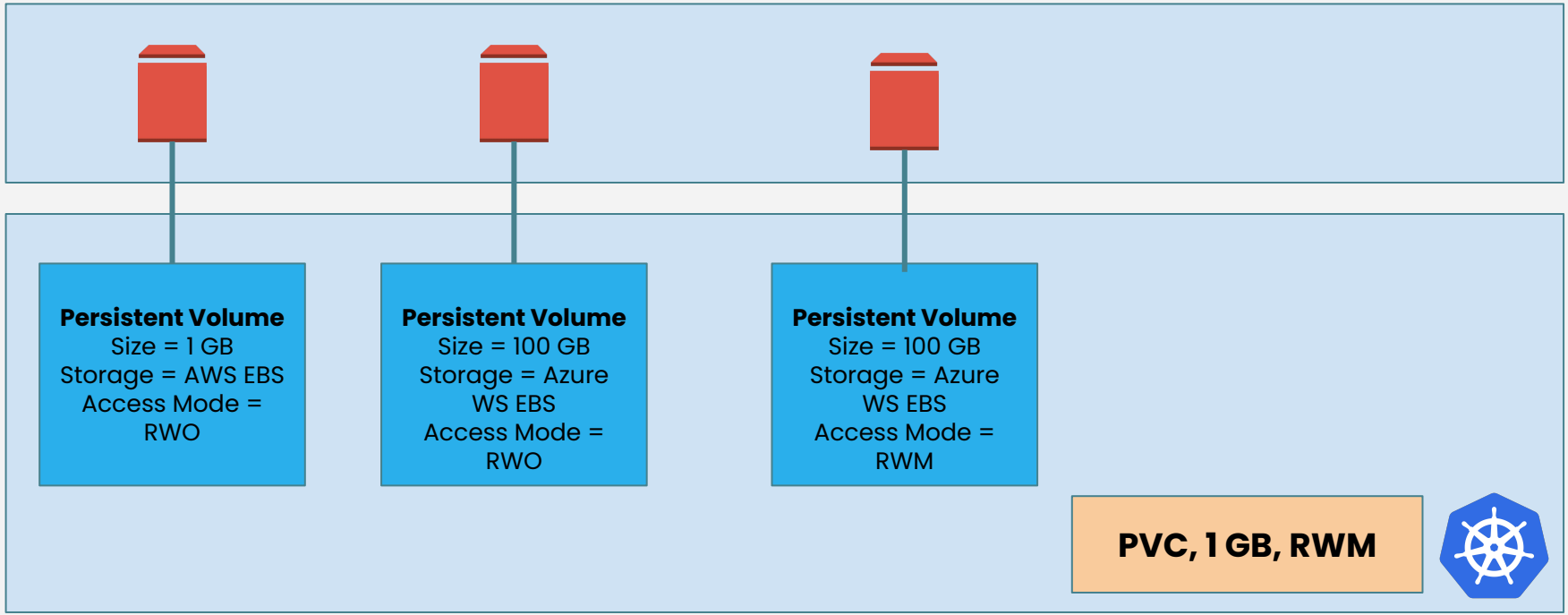
Persistent Volumes, Persistent Volume Claims

- PVs can be provisioned statically or dynamically (We will only cover static)
- Learn how PVC is matched with PV.

Persistent Volumes, Persistent Volume Claims



PV & PVC Binding



Memory & CPU Limit on Pods

By default, pods run with unbounded CPU and memory limits. This means that any pod in the system will be able to consume as much CPU and memory on the node that executes the pod.

When you specify a resource limit for a container, the kubelet enforces those limits so that the running container is not allowed to use more of that resource than the limit you set.

Use-case: You don’t want a rogue container to affect others.

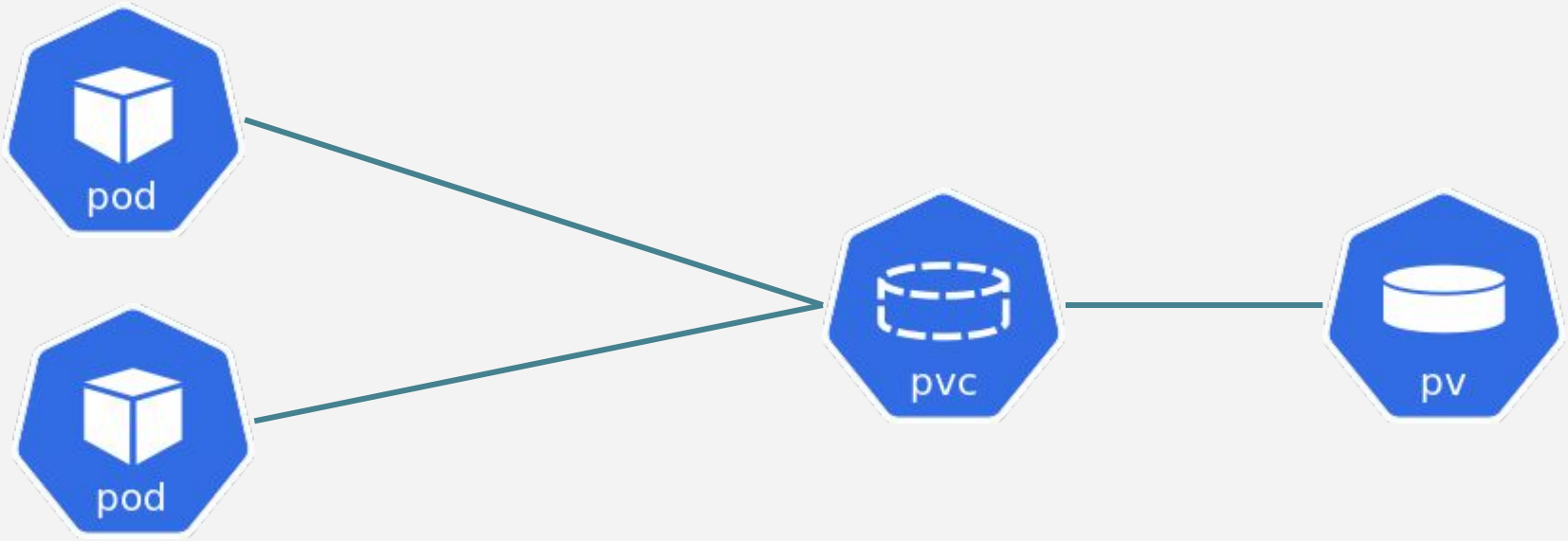
Part II

StatefulSets

Stateful sets are K8s resource that lets you deploy stateful application.

ReCap: Stateful V/s Stateless Applications

Scaling a Stateful App



Statefulset V/s Deployment

Deployment

- Random names for Pods
- Pods create parallelly
- Deleted in Random order

Stateful Sets

- Names are sequential.

- Created in Order
- Deleted in order

Hands-on: Deploying a Redis cluster

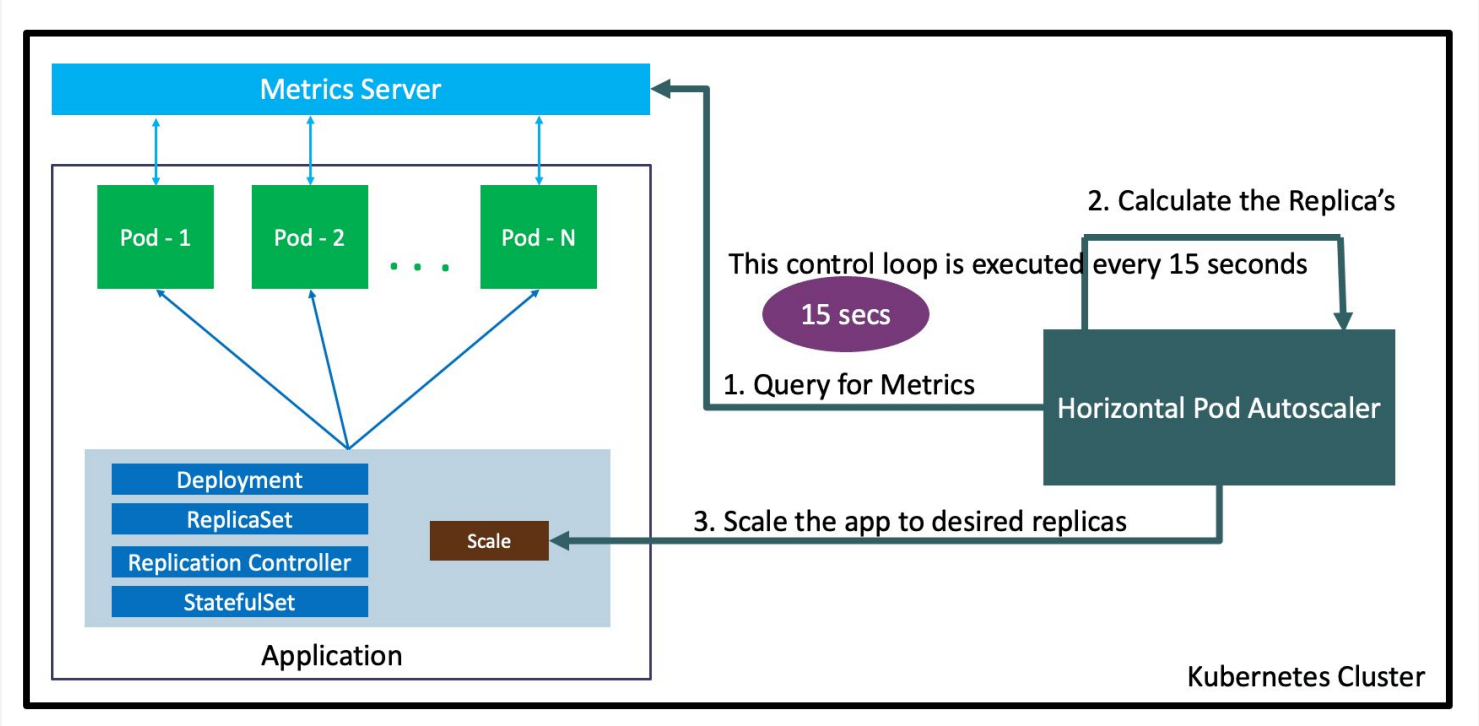
https://github.com/rustudorcalin/deploying-redis-cluster

Horizontal Pod Scaling

A HorizontalPodAutoscaler (HPA for short) automatically updates a workload resource (such as a Deployment or StatefulSet), with the aim of automatically scaling the workload to match demand.

Horizontal scaling means that the response to increased load is to deploy more Pods.

Horizontal Pod Scaling

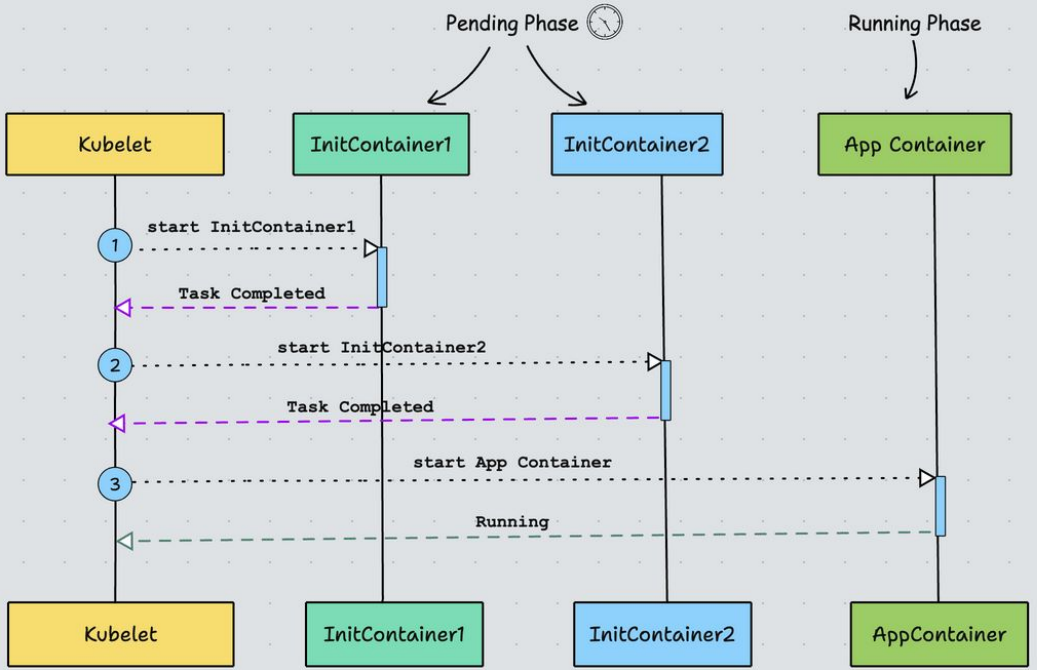


Horizontal Pod Scaling

Please read:
https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

Init Container

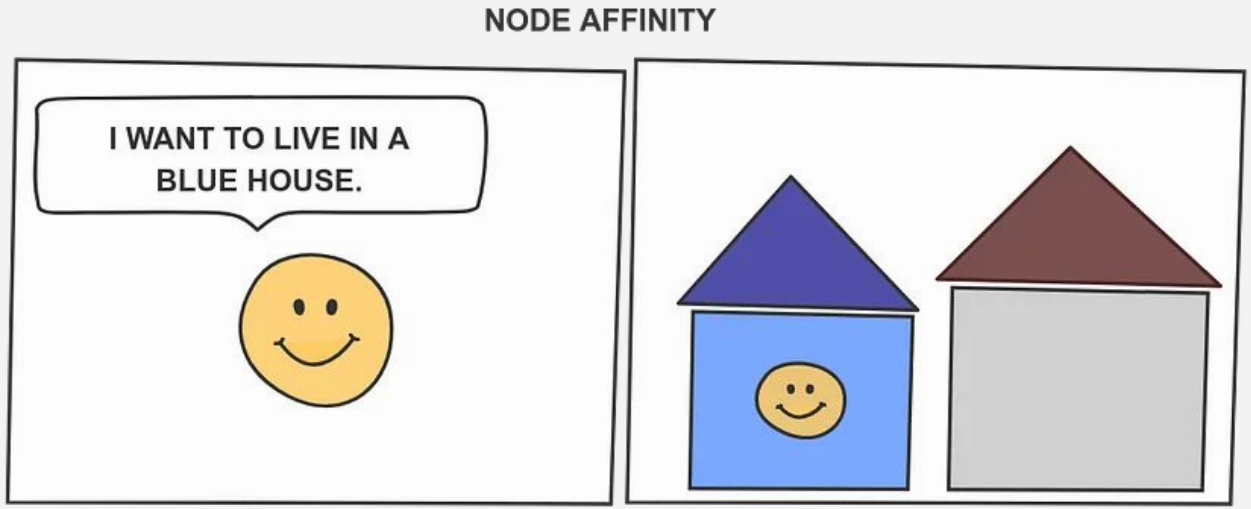
Init Containers are containers that start and run to completion before starting the main containers in the pod. It acts as a preparatory step, allowing us to perform initialization tasks, configure prerequisites, or configure dependencies required by the application in the main containers.



Source: https://devopscube.com/kubernetes-init-containers/

Affinity

Node affinity is a property of Pods that ***attracts*** them to a set of nodes



Affinity

Use Case:

- A pod might need specialized hardware such as an SSD drive or a GPU
- Pods that communicate frequently might need to be collocated with each other
- Pods that have high computational requirements might need to be confined to specific nodes

Affinity

Use Case:

- A pod might need specialized hardware such as an SSD drive or a GPU
- Pods that communicate frequently might need to be collocated with each other
- Pods that have high computational requirements might need to be confined to specific nodes

Affinity - How does it work

- You can add a label to a node using this command:

  ```
  kubectl label nodes <node-name> <key>=<value>
  ```

Affinity - NodeSelector & affinity

- You define the selector in your pod:

```
spec:
  containers:
  - name: nginx
    image: nginx
  nodeSelector:
    <key>: <value>
```

Affinity - NodeSelector & affinity

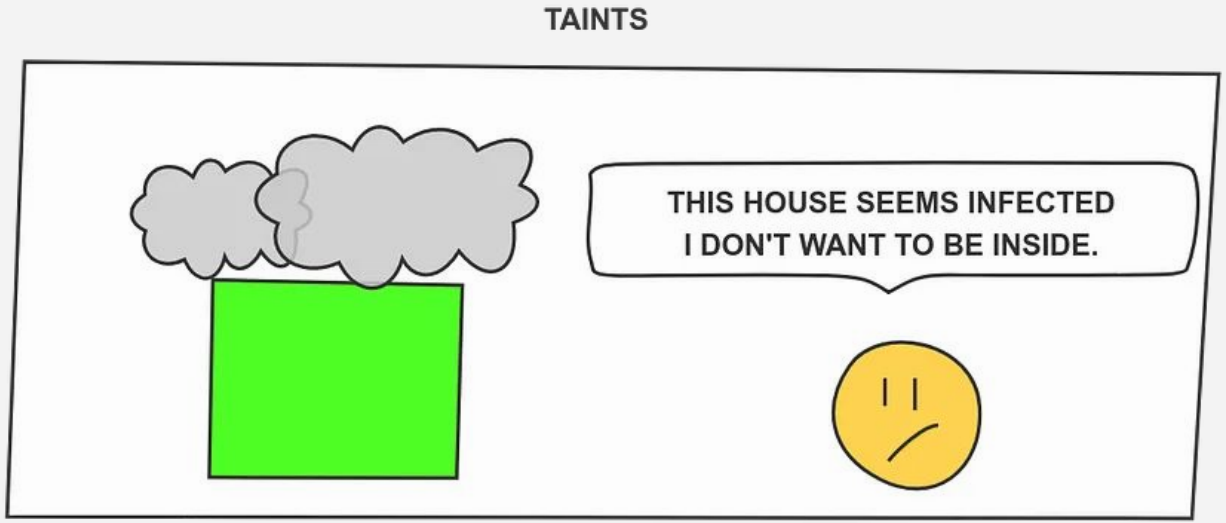
- You define the affinity

```
  affinity:
    nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: disktype
            operator: In
            values:
            - ssd
```

Taint

Taints are the opposite -- they allow a node to repel a set of pods.



Taint

Taint a Node:

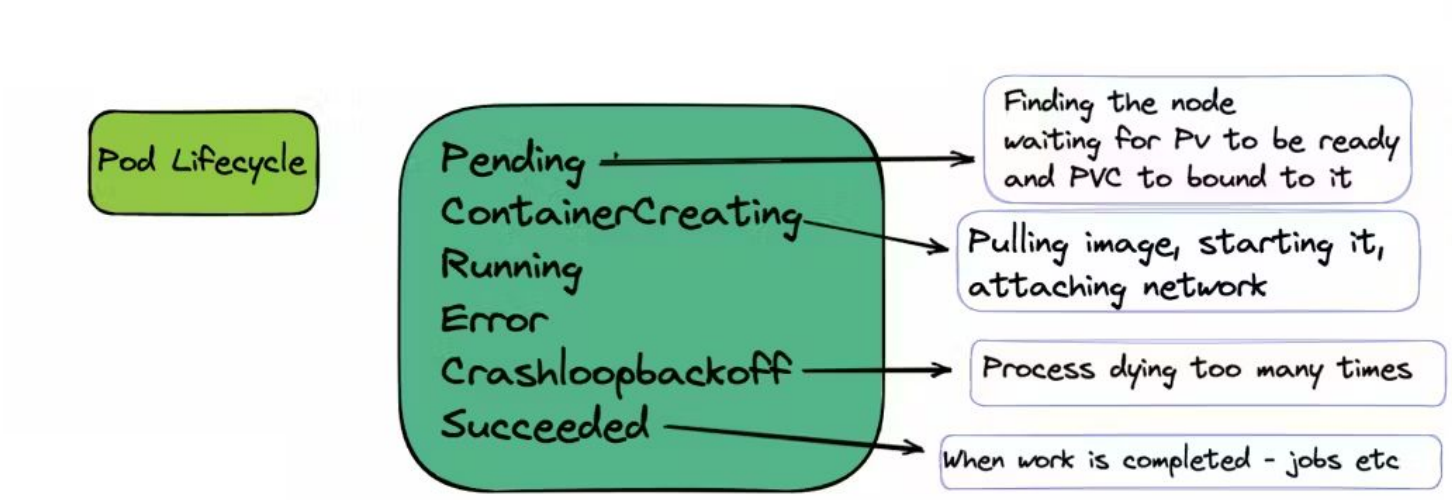
```
kubectl taint nodes node1 key1=value1:NoSchedule
```

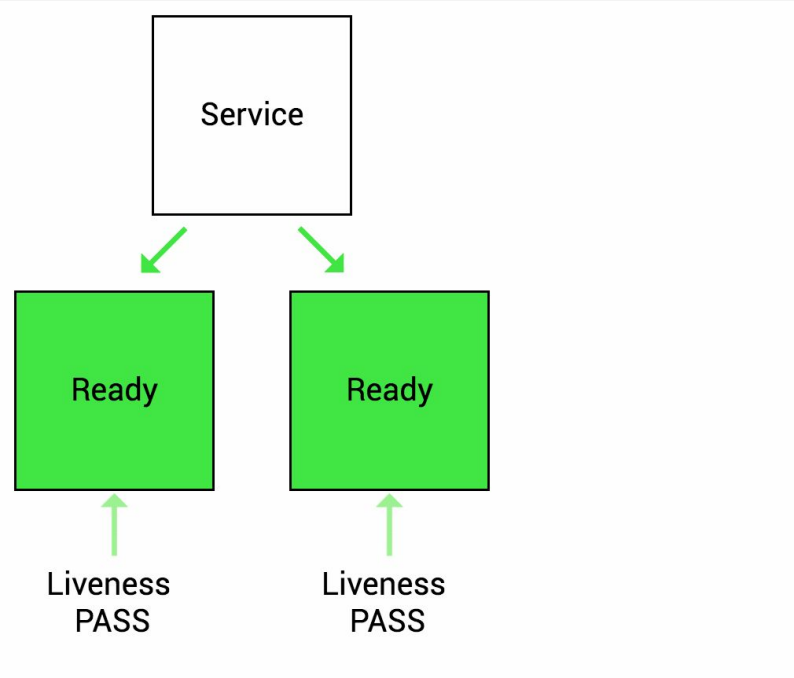
This will make no Pod to schedule on this node UNLESS it has toleration to it.

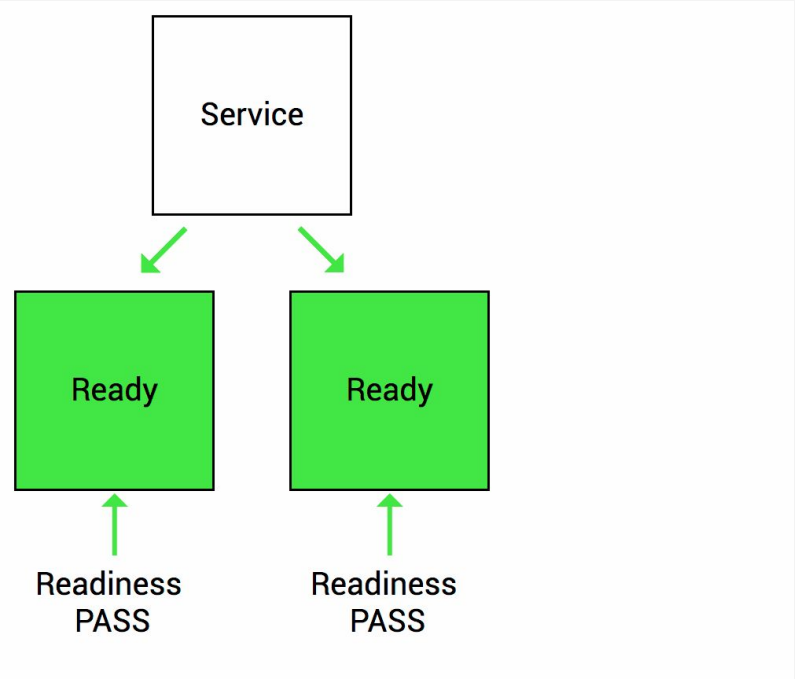
Toleration

Tolerations are used to allow a Pod to be scheduled on nodes with certain conditions that would otherwise prevent it from being scheduled there. These conditions might include node taints, which repel Pods unless the Pods have specific tolerations matching those taints.

Pod Life Cycle




Service
Ready
Ready
Liveness
PASS
Liveness
PASS


Service
Ready
Ready
Readiness
PASS
Readiness
PASS

Rolling Updates

https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/#:~:text=A%20rolling%20update%20allows%20a,before%20removing%20the%20old%20Pods.

End

Q&A

https://github.com/marcel-dempers/docker-development-youtube-series/tree/master/kubernetes