Introduction to Kubernetes

Dimitris Armenatzoglou
Senior DevOps Engineer
@ Smart Reporting

Some linguistics first… What does “Kubernetes” means?

Koo-ber-ne-teez

From the Greek:
verb: ***Κυβερνώ*** - noun: ***Κυβερνήτης***
To govern - ***governor***

Kee-ver-nee-tees

The same etymological root for the words: ***Cyber***, ***cybernetics***

It also means:
Captain, pilot or the helmsman of a ship

What is Kubernetes?

https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/:

Kubernetes is a **portable**, **extensible**, open-source **platform** for managing **containerized workloads** and **services**, that facilitates both **declarative** configuration and **automation**.

https://www.redhat.com/en/topics/containers/what-is-kubernetes:
Kubernetes (also known as k8s or "kube") is an open source **container orchestration platform** that automates many of the manual processes involved in **deploying**, **managing**, and **scaling containerized applications.**

Why “K8S”?:
https://en.wikipedia.org/wiki/Kubernetes#cite_note-4

Kubernetes Features

- Orchestrate containers on multiple hosts.
- Autoscaling
- Stateless and stateful applications
- Service discovery
- Extensible
- Rolling updates and Rollbacks
- Resource monitoring and Self-healing
- Storage and networking orchestration
- Native load-balancing
- ...

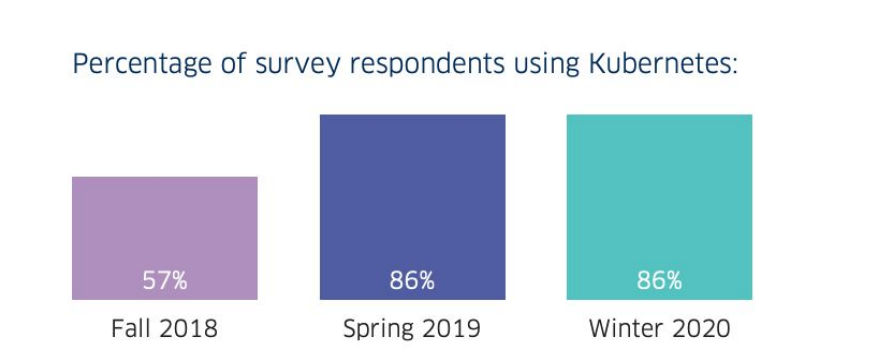
Most importantly:

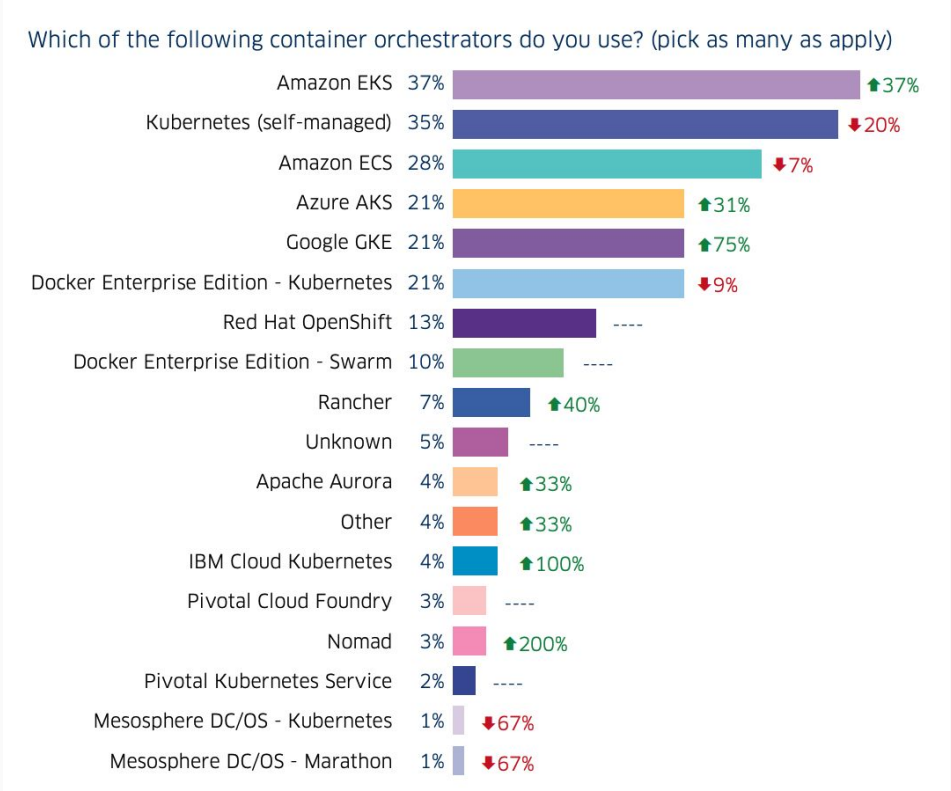
One common interface - the same API - for running containerized applications **across any cloud or any bare-metal environments**.

Kubernetes Market Adoption

Kubernetes dominates the containers market.

According to the StackRox survey “The state of Container and Kubernetes Security Winter 2020”
86% of the responders use Kubernetes in production.





https://security.stackrox.com/state-of-containers-and-kubernetes-security-report-winter-2020.html

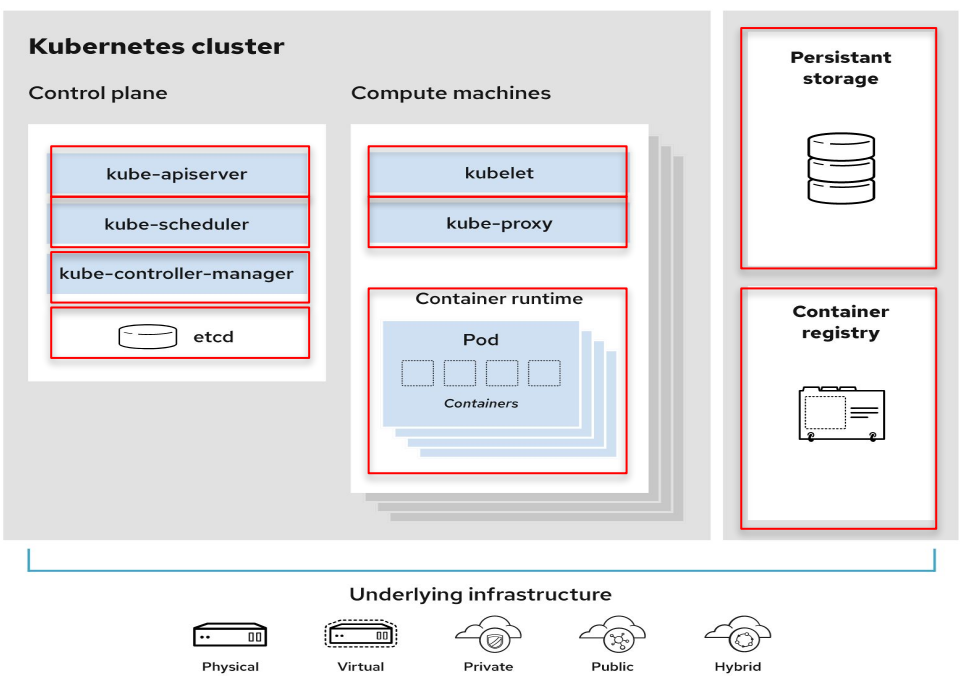
Kubernetes Architecture

API-Server:
The interface to interact with the cluster. REST.

Scheduler:
Schedules a POD to an appropriate Kubernetes node.

Controller-manager:
Runs controller processes. E.g. Node-controller, replication controller, Endpoints controller etc.

etcd:
Stores cluster state and configuration



Persistent Storage:
nfs, efs, ebs, glusterfs etc.

Container Registry:
Harbor, Nexus, Docker Hub etc.

Kubelet:
It ensures that a set of PodSpecs is running healthy as containers on a node.

Kube-Proxy:
A network proxy running on every node. Allows network communication to the Pods from inside or outside the cluster

Container Runtime:
The engine which enables the cluster to run containers. E.g. Docker, rkt, containerd etc.

Some k8s key concepts:
PODs, Nodes, Services, Controllers

First things first… Enable Kubernetes on Docker Desktop

Install some k8s tools:

$ brew/yum/apt install kubectl

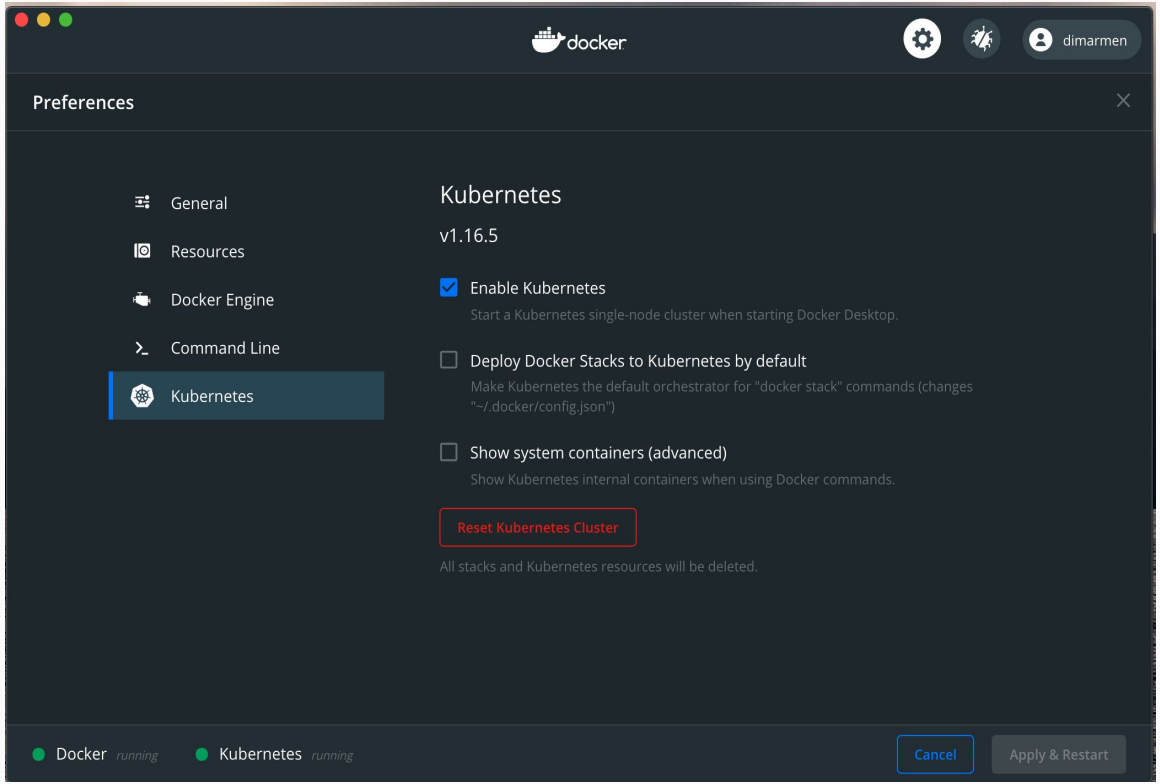
https://kubernetes.io/docs/tasks/tools/install-kubectl/

$ kubectl cluster-info

Workshop Manifests:
https://github.com/DimArmen/kubernetes_demo

Some additional tools that you may find handy in the future:
https://github.com/kubermatic/fubectl
https://github.com/ahmetb/kubectx

Kubernetes on Docker-Desktop:



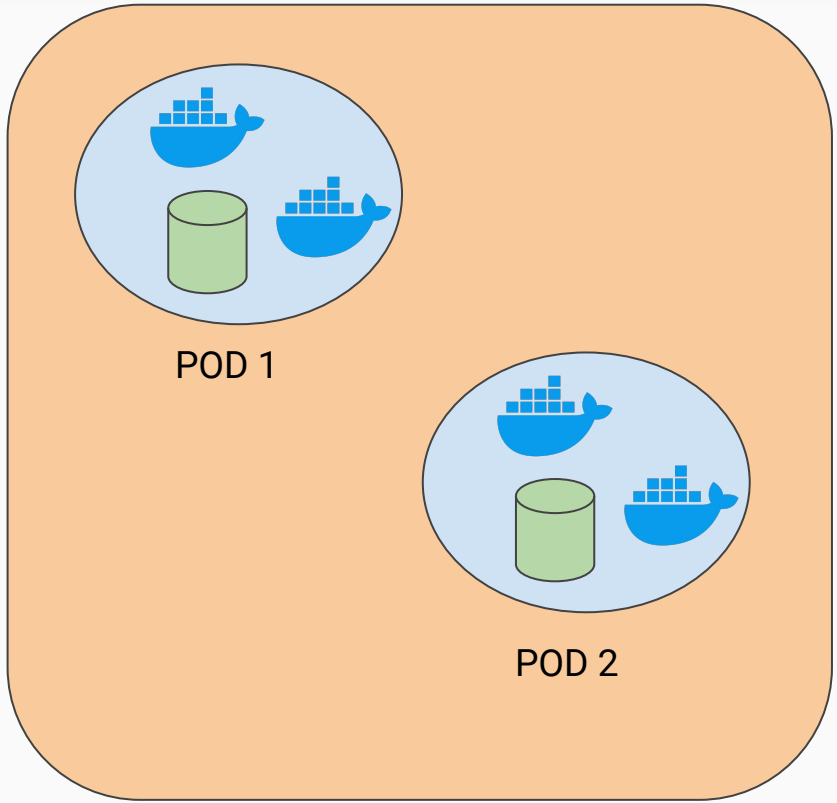
Pods

- The atomic unit of a kubernetes workload. The smallest deployable unit of work.
- A POD is **one or more** containers which form a cohesive instance of a running application.
- The PODS are **ephemeral**.
- Can we deploy Containers on Kubernetes? **Yes!** But **only inside Pods**.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-1
  labels:
    app: web-1
spec:
  containers:
    - name: web
      image: nginx:1.18
      ports:
        - name: web
          containerPort: 80
          protocol: TCP
```

Pod name and labels that can be used by other controllers to identify this pod

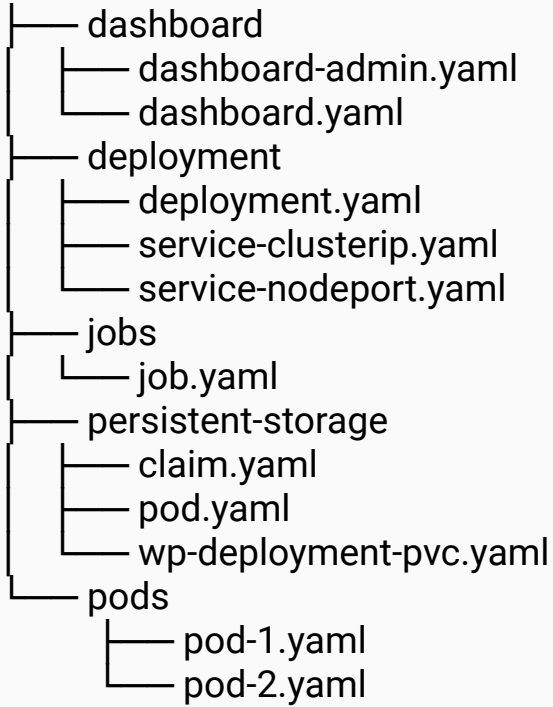
The container name, image and ports



https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.18/

https://kubernetes.io/docs/concepts/workloads/pods/

Create PODs



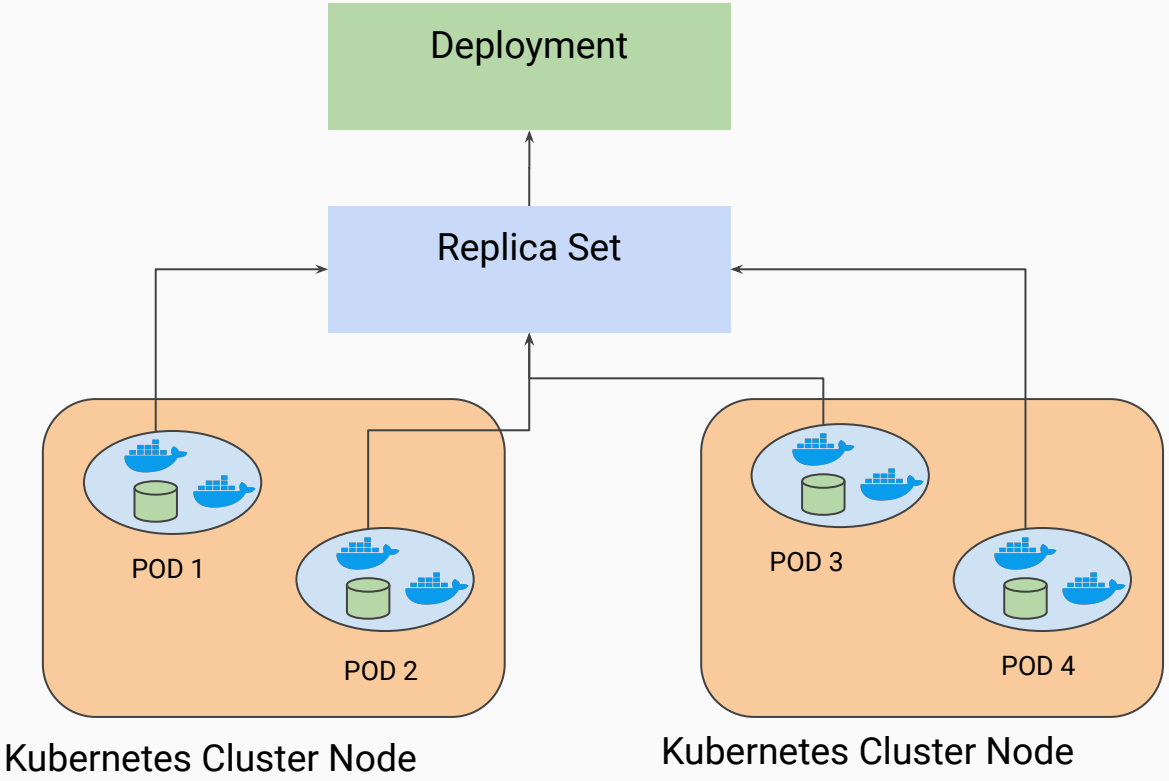

```
$ kubectl apply -f pods/pod-1.yaml
$ kubectl exec -it web-1 -- bin/sh
$ kubectl logs (-f) web-1
$ kubectl delete web-1
$ kubectl delete -f pods/pod-1.yaml
```

Deployment

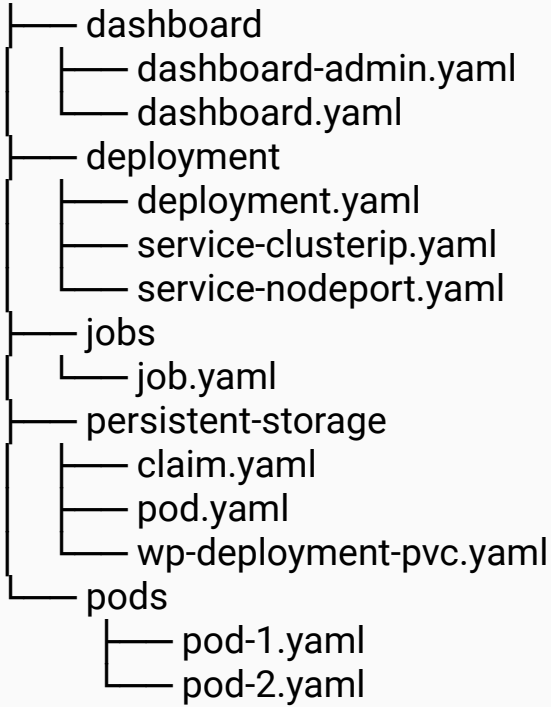
A Deployment Controller provides:

- Declarative updates for Pods via ReplicaSets.
- Update control and rollback functionality
- Uses POD templates to generate the replicaSets

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



Create Deployment




```
$ kubectl apply -f deployments/deployment.yaml
$ kubectl describe deployment hello-kubernetes
$ kubectl rollout history deployment hello-kubernetes
$ kubectl rollout history deployment hello-kubernetes --revision <number>
$ kubectl undo deployment hello-kubernetes
$ kubectl edit deployment hello-kubernetes
```

Service

Kubernetes Services:

- An endpoint used to expose an application running as a set of pods as a network service.
- Provide a reliable network endpoint for routing traffic to pods. **Not Ephemeral.**
- Autodiscovers associated PODS using matchLabels.
- Provides a DNS name such as: **http://<service_name>:<service_port>**

Types of Services:

- ClusterIP
- NodePort
- LoadBalancer

```yaml
apiVersion: v1
kind: Service
metadata:
  name: hello-kubernetes
spec:
  selector:
    name: hello-kubernetes
  type: ClusterIP | NodePort | LoadBalancer
  ports:
  - name: http
    port: 80
    targetPort: 8080
    nodePort: 32000
```

Pods Match labels

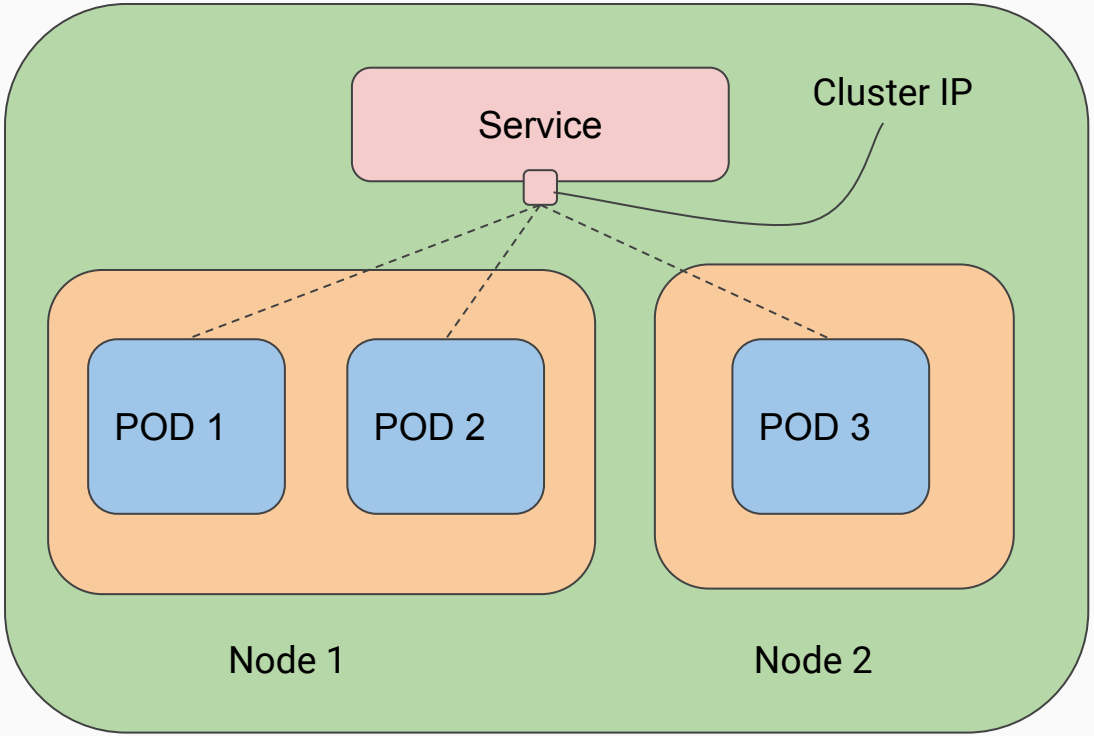
Service Port

Pods Port

If NodePort is used

ClusterIP - Cluster Internal Only

- Cluster **internal only** communication.
- Finds the PODs by using the MatchLabels and groups them in a network service to forward traffic to them



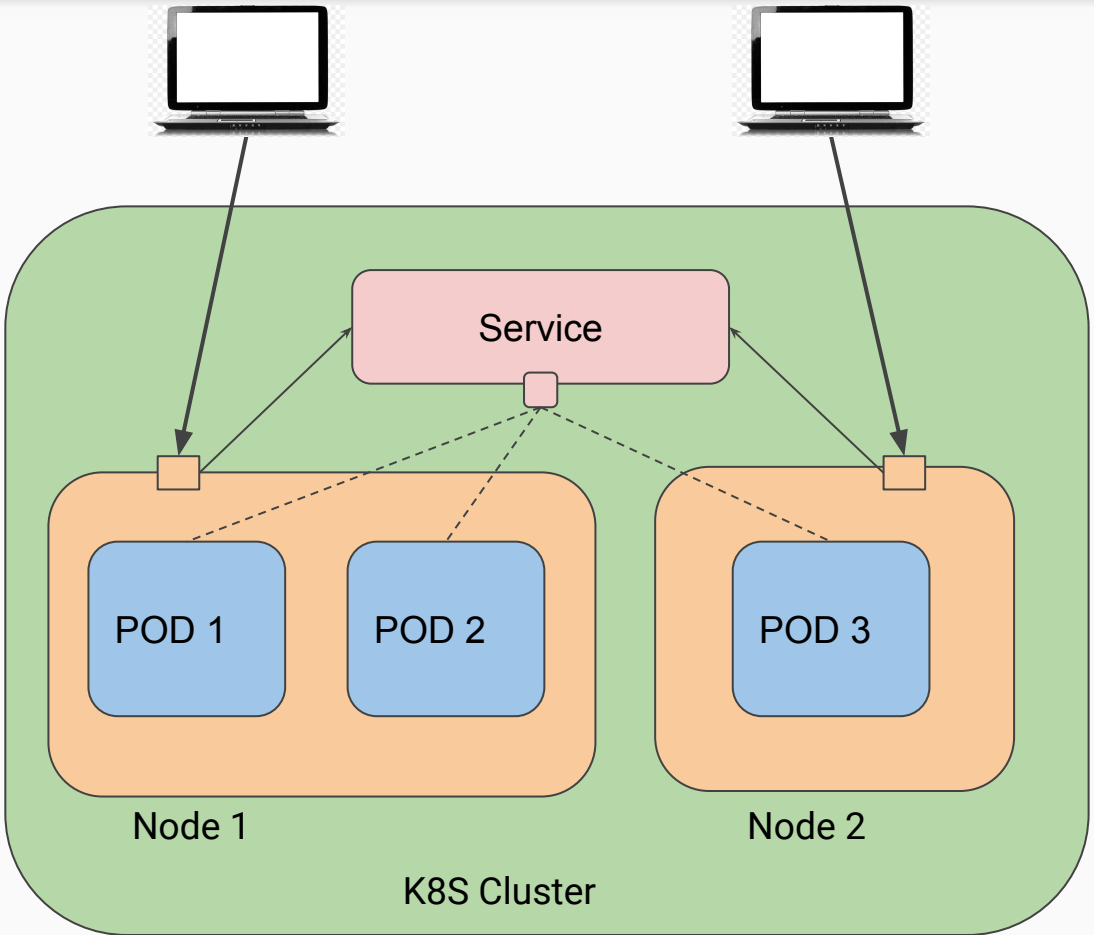
K8S Cluster

NodePort

A NodePort is exposing the Service on each Node.

From **outside the cluste**r the Service is reachable at:

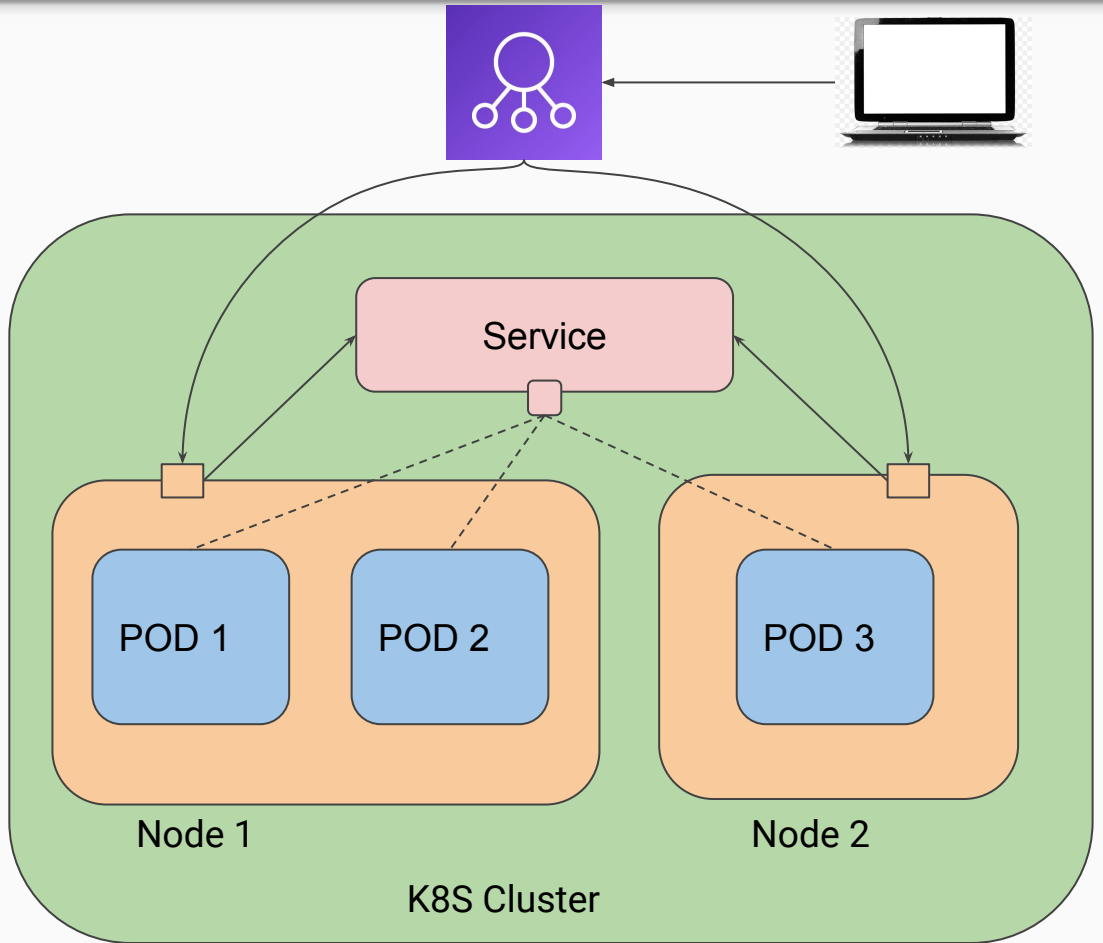
<any_node_IP>:<NodePort>



LoadBalancer

Usually in a Cloud environment where Load Balancers are offered as a service (AWS ALB etc.)

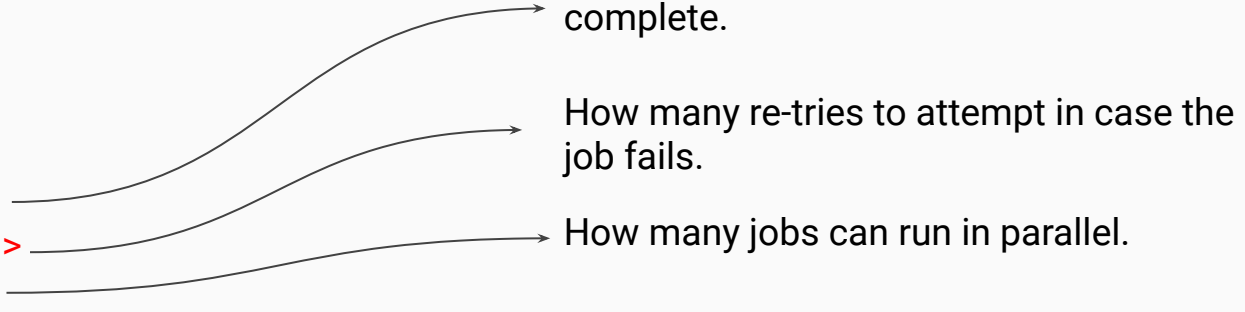
The Kubernetes cloud-controller-manager will automatically create the load balancer resource.



Job

Creates one or more pods and ensures that a specific number of them terminates successfully.

```
apiVersion: batch/v1
kind: Job
metadata:
 name: hello
spec:
 template:
   spec:
     containers:
     - name: hello
       image: busybox
       args: [“hello”]
       command: ["echo"]
     restartPolicy: Never
     # completions: <int>
     # backoffLimit: <int>
     # parallelism: <int>
```



Persistent Volumes and Persistent Volume Claims

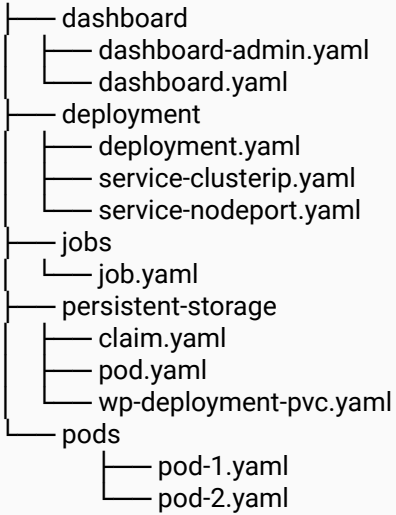
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: app
spec:
  containers:
  - name: app
    image: centos
    command: ["/bin/sh"]
    args: ["-c", "while true; do echo $(date
-u) >> /data/out.txt; sleep 5; done"]
    volumeMounts:
    - name: persistent-storage
    mountPath: /data
  volumes:
  - name: persistent-storage
    persistentVolumeClaim:
      claimName: pvc-example
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-example
spec:
  accessModes:
     - ReadWriteOnce
  storageClassName: hostpath
  resources:
     requests:
       storage: 4Gi
```

Mount the volume specified by “name” at the “mountPath”

Use the PVC as a volume for this POD.

Kubernetes Dashboard



Deploy the Dashboard:

```
$ kubectl apply -f dashboard/dashboard.yaml

$ kubectl apply -f dashboard/dashboard-admin.yaml

$ kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{print $1}')


$ kubectl proxy


The Dashboard is available at:

http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/
```

Thank you!