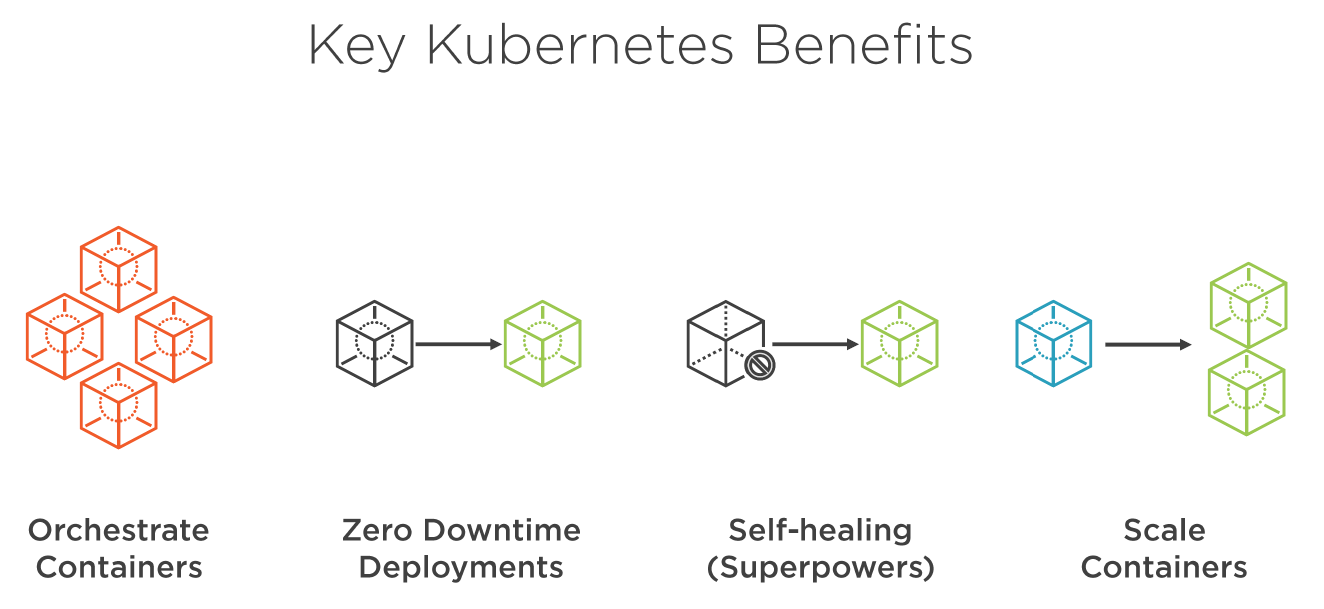


Kubernetes is an open-source container-orchestration system for automating application deployment, scaling, and management of containerized applications. which used to automate container deployment.

Pods are thing but nodes. In pod we can have single or multiple containers. Every pod will have IP, memory, volumes, etc. shared across containers.



**Horizontal Scaling:**

**Self-Healing:** restarts the containers whenever it fails.

**Fault tolerance:** replace and re schedules whenever node dies.

**Clustering:**

**Zero downtime deployment:**

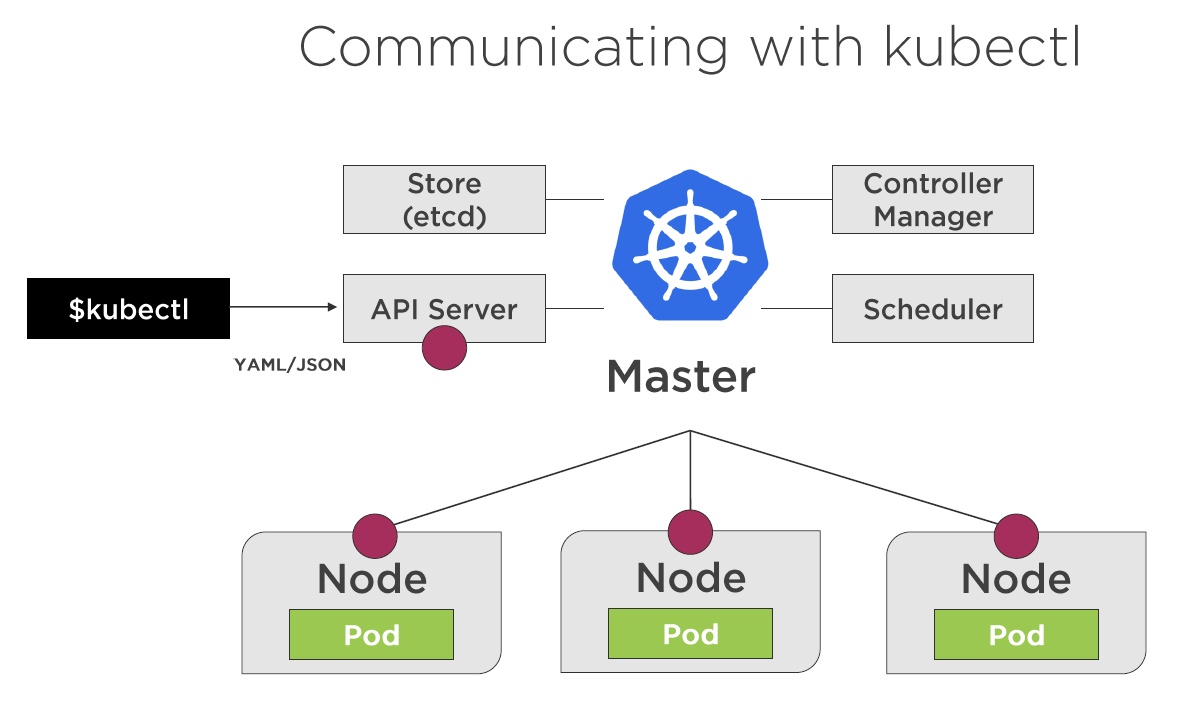
* Rolling update (default)
* Blue-green deployment
* Canary deployment
* Rollbacks

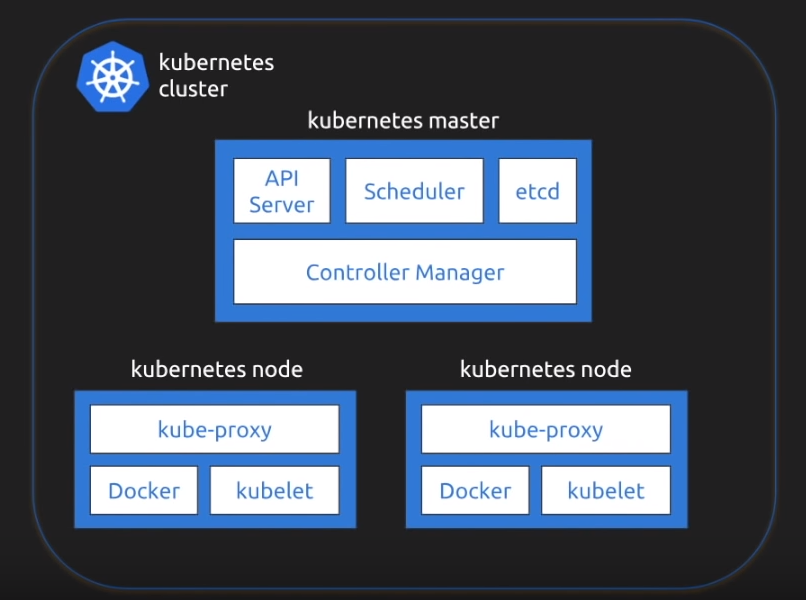
**Scheduling:**

**Load balancing:**

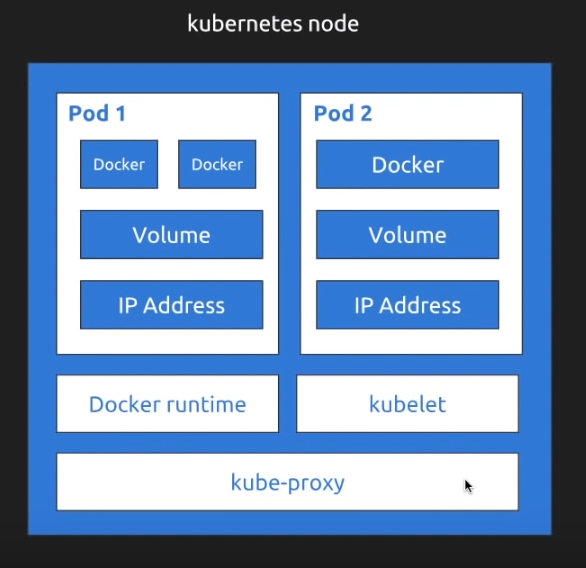
**Automated roll outs and roll backs:** roll out changes to application or config changes to all the instances (for all the container in cluster) with zero down time, it make sure it doesn’t kill all the instances same time. If any failure it’s roll backs.

**Configuration Management:** no need to redeploy.





Kubernetes follows master slave architecture.



Kubernetes node will have one or multiple pods.

Container processes need to bind to different ports within a pod.

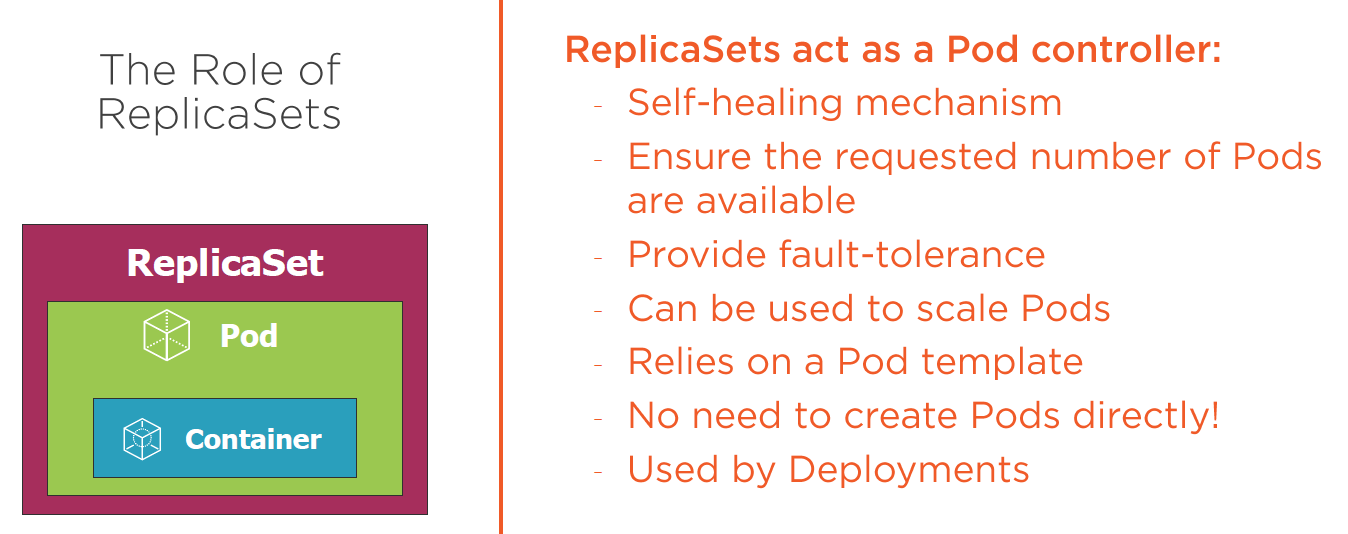
Kubernetes relies on Probes to determine the health of pod container. A probe is a diagnostic performed periodically by kubelet on a container. Two type of probes.

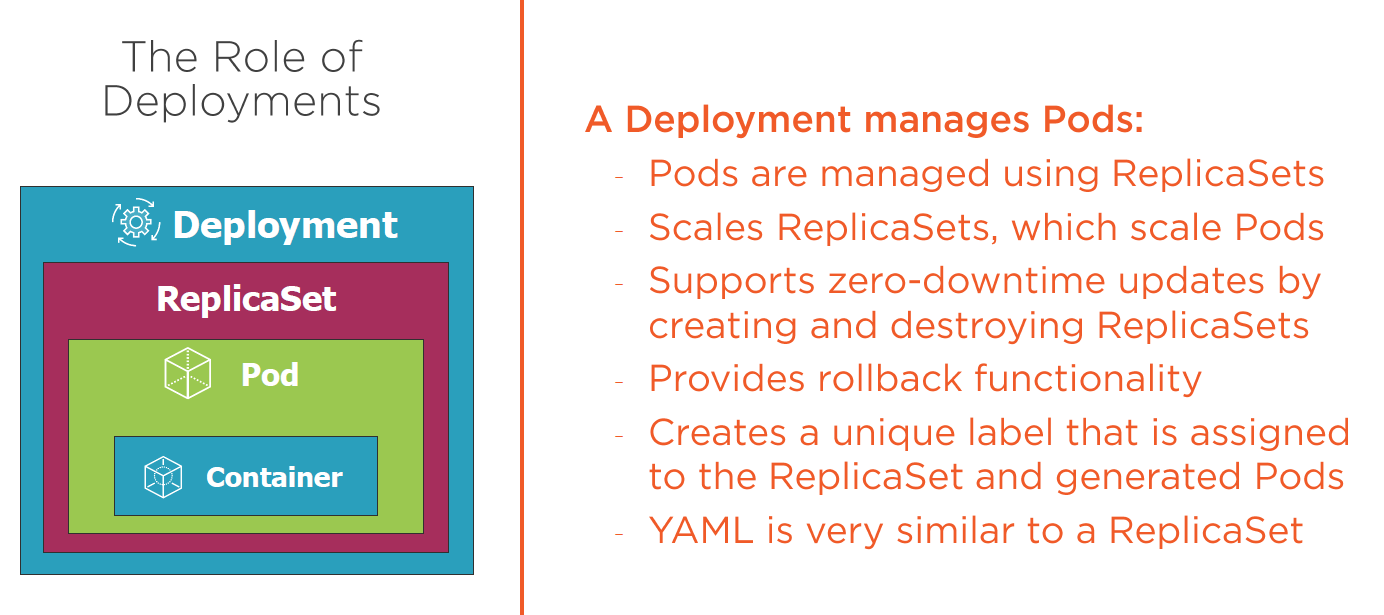
1. Liveness probe: used to determine if pod is healthy and running as expected.
2. Readiness probe: used to determine if pod should receive requests.

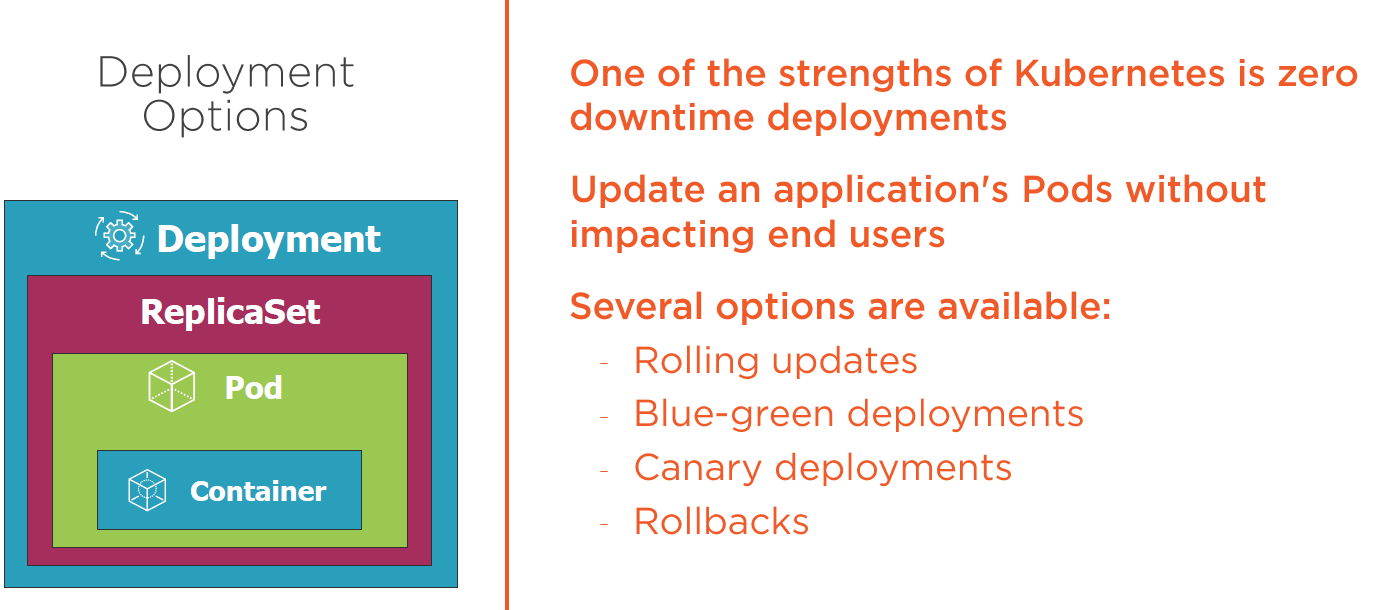
**ReplicaSet:** A ReplicaSet is declarative way to manage pods.

**Deployment:** A deployment is a declarative way to manage Pods using a ReplicaSet.

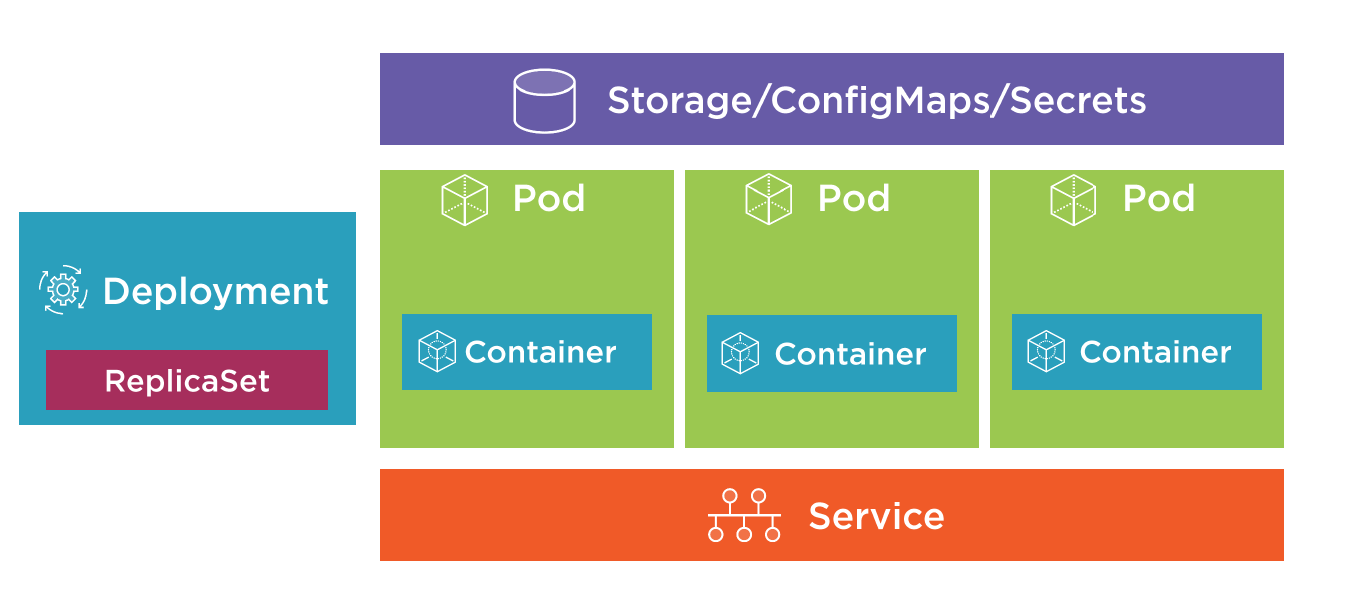
Deployment and ReplicaSet ensure pods stay running and can be used to scale pods.

****

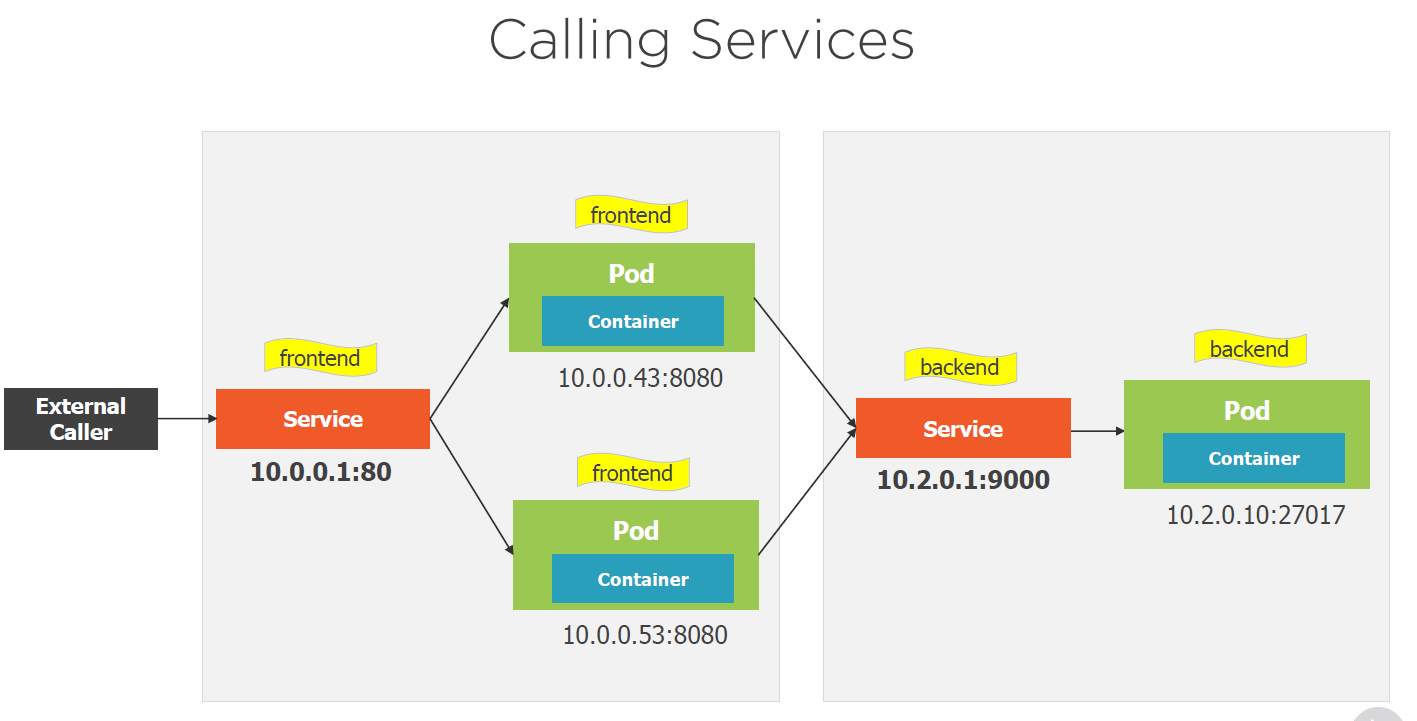
****

****

**Service:**

****

|  |  |
| --- | --- |
|  | Services abstract Pod IP addresses from consumers.  Load balances between pods.  Relies on labels to associate a Service with a pod.  Node’s kube-proxy creates a virtual IP for Services.  Creates endpoints which sit between a service and pod. |



Service Types:

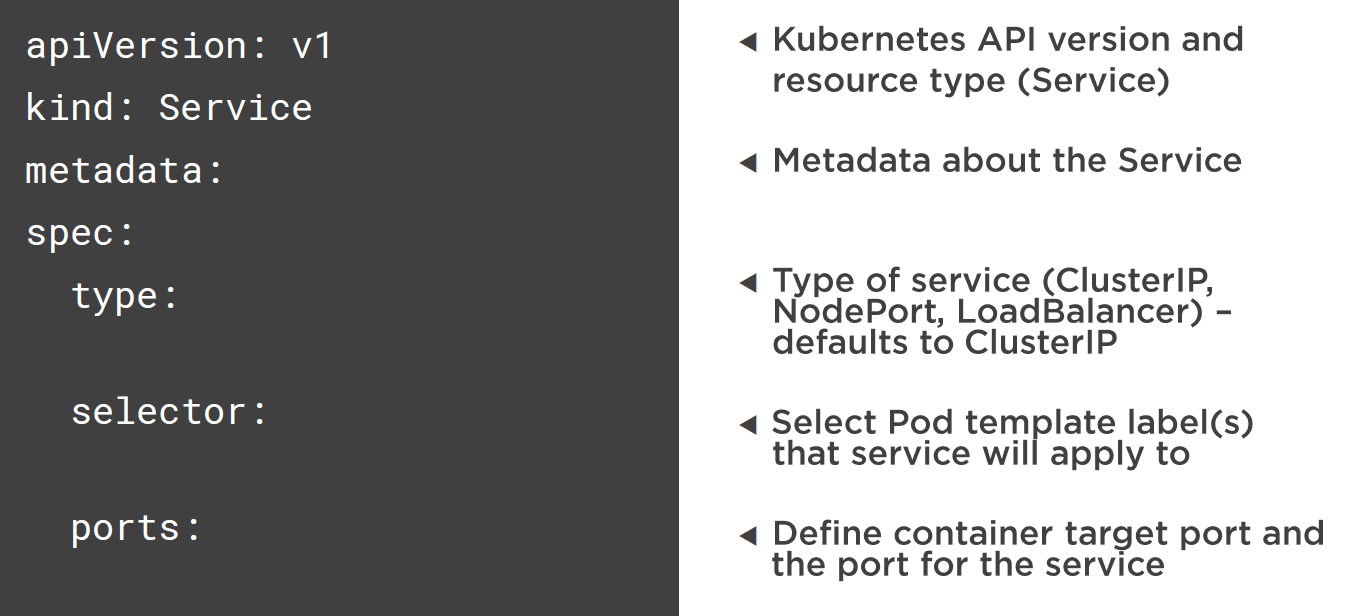
* ClusterIP – Expose the service on a cluster-internal IP (default)
* NodePort – Expose the service on each Node’s IP at a static port.

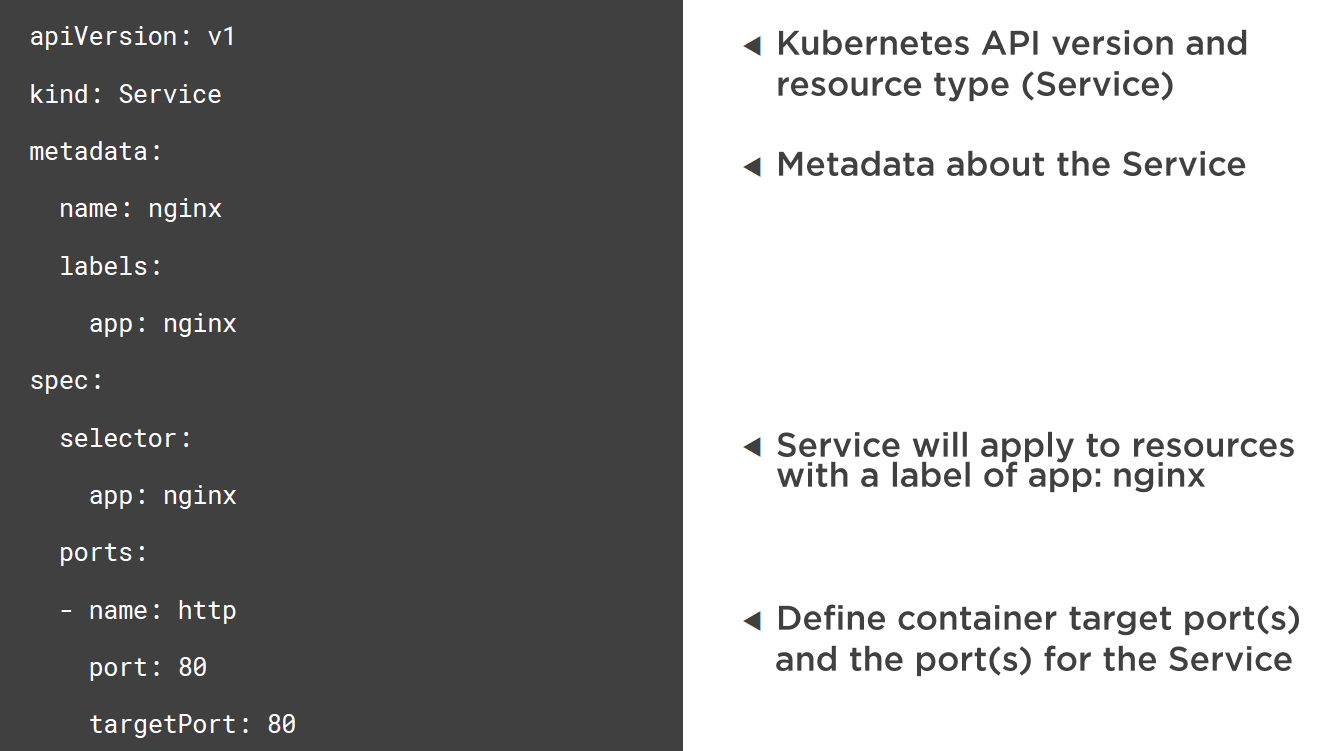
Exposes the service on each node’s IP at static port. Port range (30000 to 32767)

* LoadBalancer – Provision an external IP to act as a load balancer for the service.
* ExternalName – Maps a service to a DNS name.

Used when we talk to external system’s, when external system’s changes IP we won’t get affected.

<https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0>

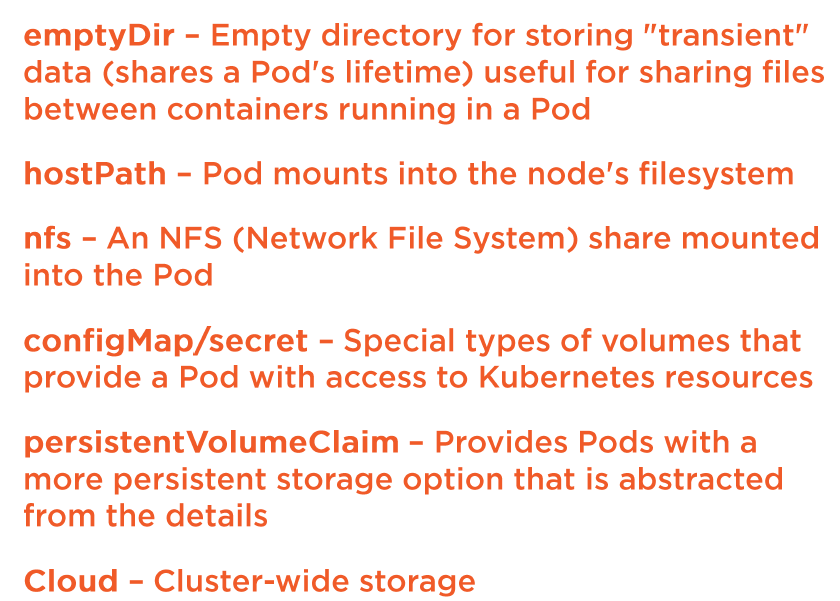




Service and pods are joined together with label’s and selector’s.

**Volume:**

Volume can be used to hold data and state and exchanges it between pods. Pods can have multiple volumes. Volume types.

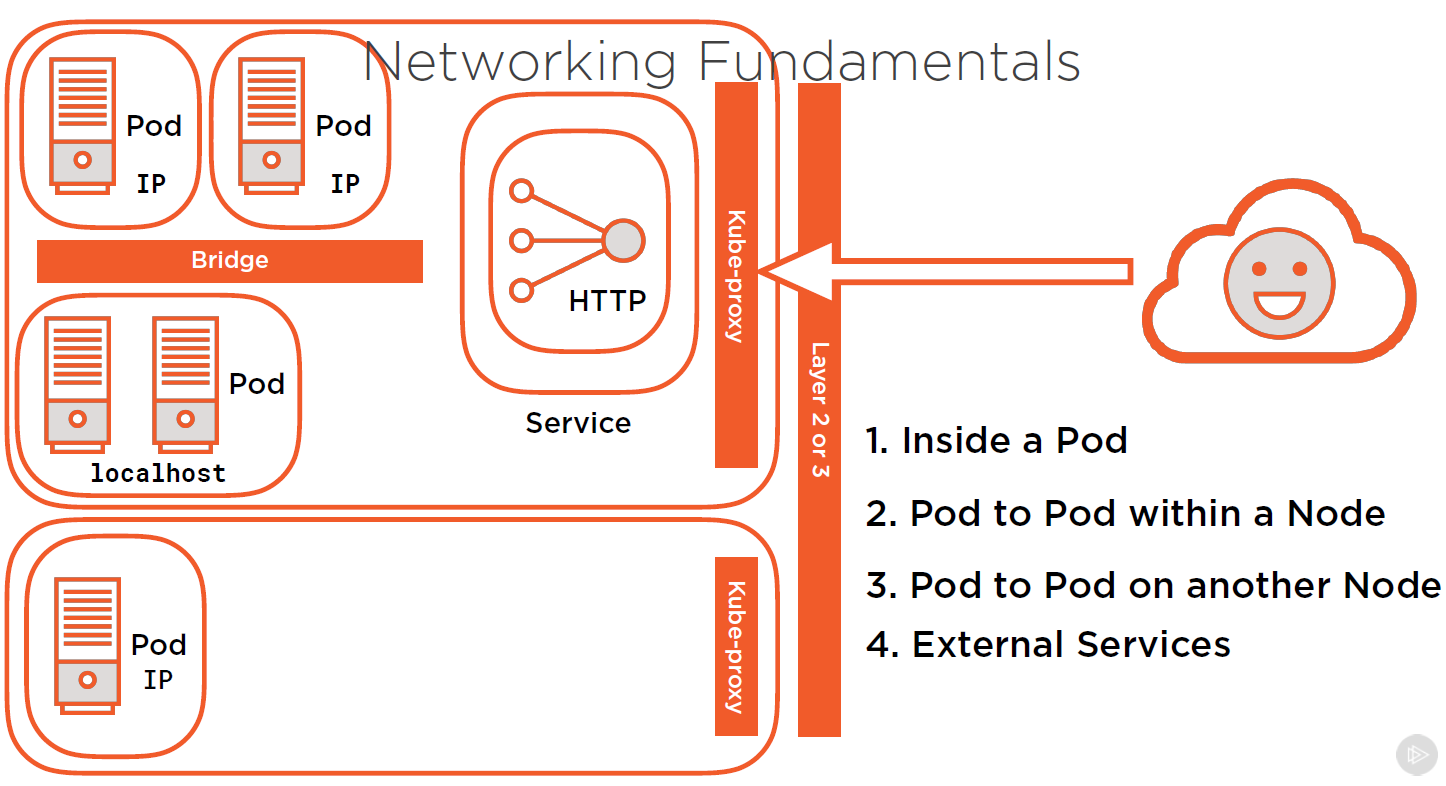


**configMap:** provides way to store configuration information and provide it to containers.



Read this <https://matthewpalmer.net/kubernetes-app-developer/articles/ultimate-configmap-guide-kubernetes.html>

1. Container to container in pod can communicate with localhost
2. Pod to pod in Node can communicate with Bridge using IP address.
3. Pod to pod on another node can communicate with layer 2 or 3 or overlay.
4. External access using port



**Namespace:**

