Expose an App in Kubernetes (K8s)

Kubernetes cluster

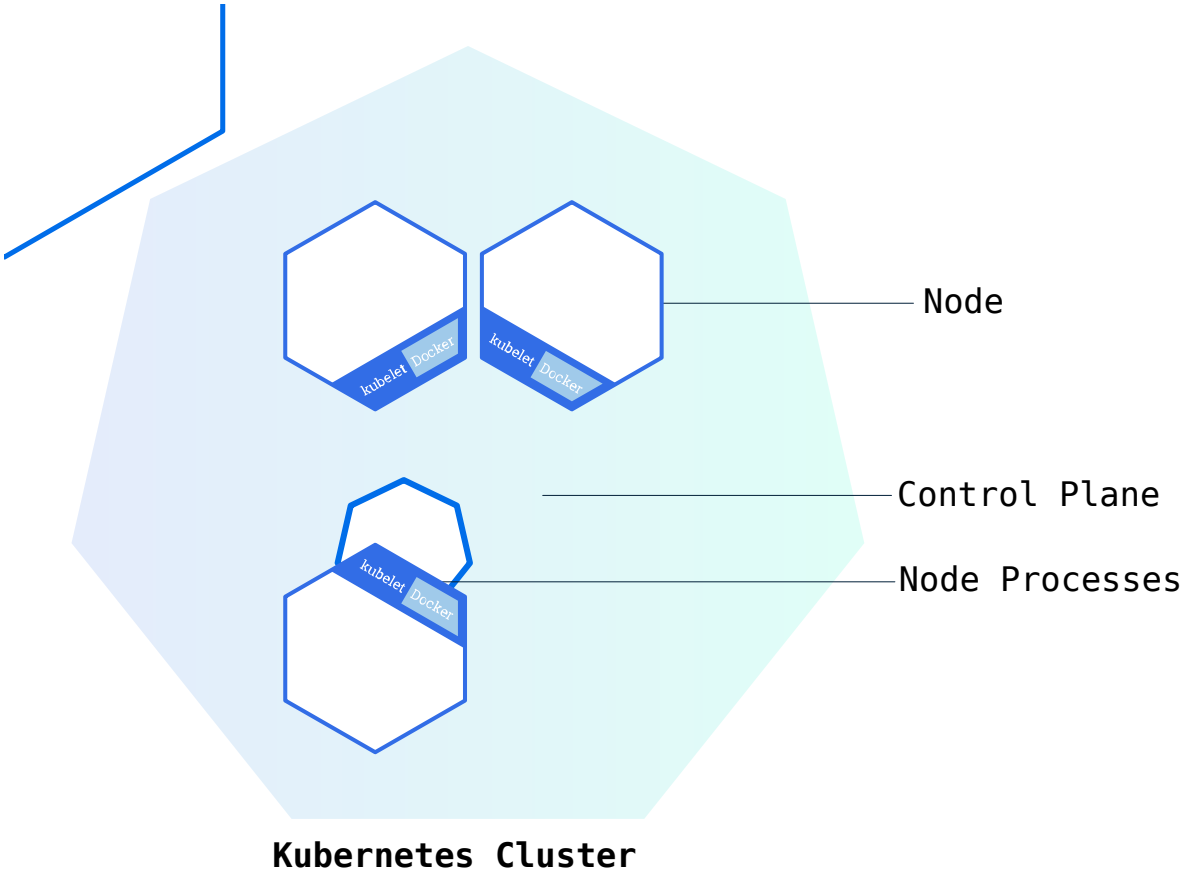
A Kubernetes cluster should have a minimum of three nodes because if one node goes down, both an etcd member and a control plane instance are lost, and redundancy is compromised. You can mitigate this risk by adding more control plane nodes.

Minikube

Minikube is a lightweight Kubernetes implementation that creates a VM on your local machine and deploys a simple cluster containing only one node.

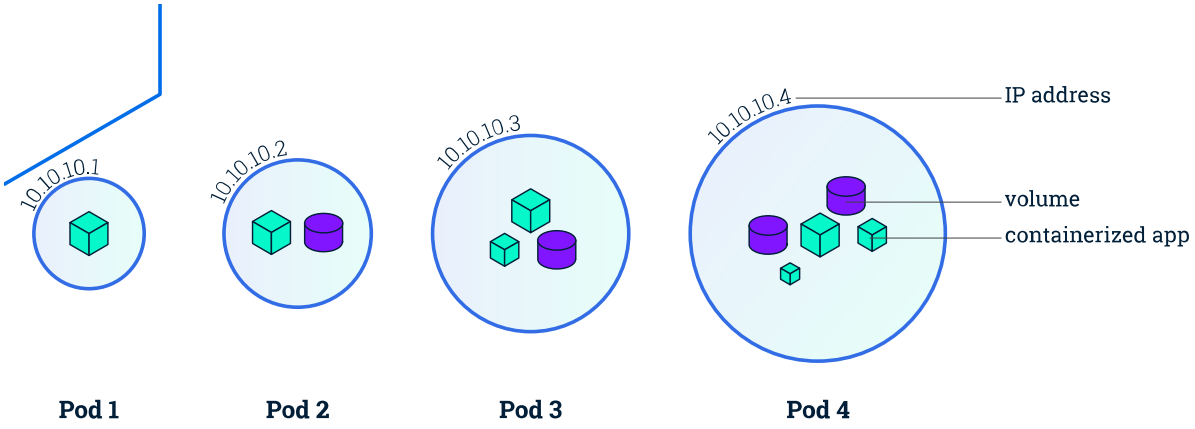
Deployment

Once you have a running Kubernetes cluster, you can deploy your containerized applications on top of it by creating a Kubernetes Deployment. The Deployment instructs Kubernetes how to create and update instances of your application.



Pods

When you create a Deployment, Kubernetes creates a Pod to host your application instance. A Pod always runs on a Node and represents a group of one or more application containers. Some of them share resources like storage (as Volumes), Networking (as a unique cluster IP address) and Information on how to run each container (as container image version or configurations).



Nodes

A node is a VM or a physical computer that serves as a worker machine in a Kubernetes cluster. Every Kubernetes Node runs at least:

- Kubelet, a process responsible for communication between the Kubernetes control plane and the Node.
- A container runtime (like Docker) responsible for pulling the container image from a registry, unpacking the container, and running the application.

Kubernetes Service

- A Service is an abstraction that allows pods to die and replicate in Kubernetes without impacting your application.
- A Service routes traffic across a set of Pods. Discovery and routing among dependent Pods (such as the frontend and backend components in an application) are handled by Kubernetes Services.

Purpose of a Kubernetes Service

Although each Pod has a unique IP address, those IPs are not exposed outside the cluster without a Service. Services allow your applications to receive traffic. Services can be exposed in different ways by specifying a type in the spec of the Service:

- **ClusterIP (default)** - Exposes the Service on an internal IP only reachable from within the cluster.
- **NodePort** - Exposes the Service on the same port of each selected Node in the cluster using NAT.

Makes a Service accessible from outside the cluster using `<NodeIP>:<NodePort>`

- **LoadBalancer** - Creates an external load balancer in the current cloud (if supported) and assigns a fixed, external IP to the Service.
- **ExternalName** - Maps the Service to the contents of the `externalName` field (e.g. foo.bar.example.com), by returning a CNAME record with its value. No proxying of any kind is set up.

Service type – NodePort

When you are using a Kubernetes cluster on premises (i.e. on your development machine with

Minikube), NodePort is a common service type used to expose your services. NodePort relies on the underlying hosts upon which you run Kubernetes to be accessible on your local network, and exposes the service definition through a high-numbered port on all of Kubernetes cluster nodes.

These services are exactly like the default ClusterIP services, with the exception that they have a type of NodePort. To create such a service with the expose command, add `–type=Nodeport` as option to your command.

Service type – LoadBalancer

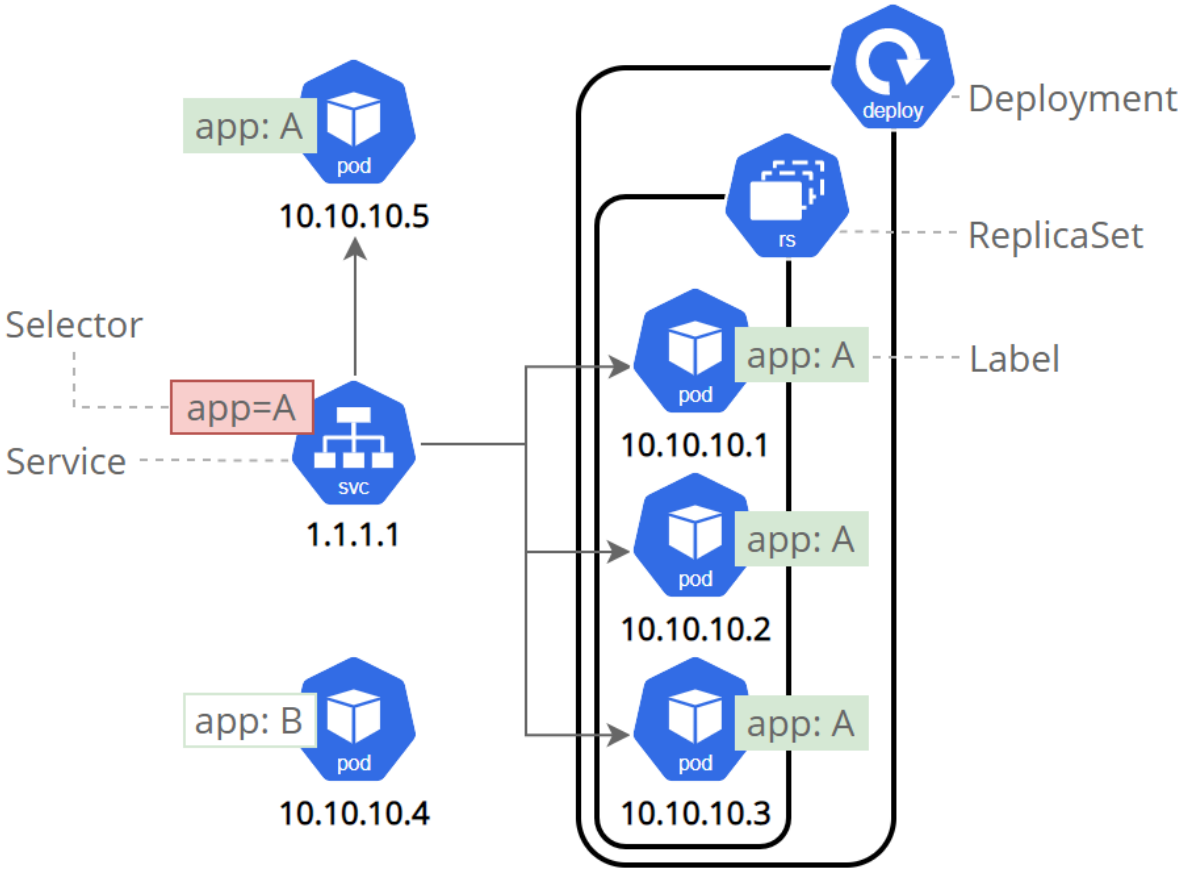
The LoadBalancer service type is not supported in all Kubernetes clusters. It is most commonly used with cloud providers such as Amazon, Google, or Microsoft, and coordinates with the cloud provider's infrastructure to set up an external LoadBalancer that will forward traffic into the service.

How you define these services is specific to your cloud provider, and slightly different between AWS, Azure, and Google. LoadBalancer service definitions may also include recommended annotations to help define how to handle and process SSL traffic. More details about the specifics for each provider can be found in the Kubernetes documentation. The documentation on the LoadBalancer definitions is available at https://kubernetes.io/docs/concepts/services-networking/service/#type-loadbalancer.

Label and Selectors

Services match a set of Pods using labels and selectors, a grouping primitive that allows logical operation on objects in Kubernetes. Labels are key/value pairs attached to objects and can be used in any number of ways:

- Label objects for development, test, and production
- Embed version tags
- Classify an object using tags



From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/en/modul/m321_aws/topics/05?rev=1755113614

Last update: **2025/08/13 21:33**