POD Restart Policy

Never >>

Always >>

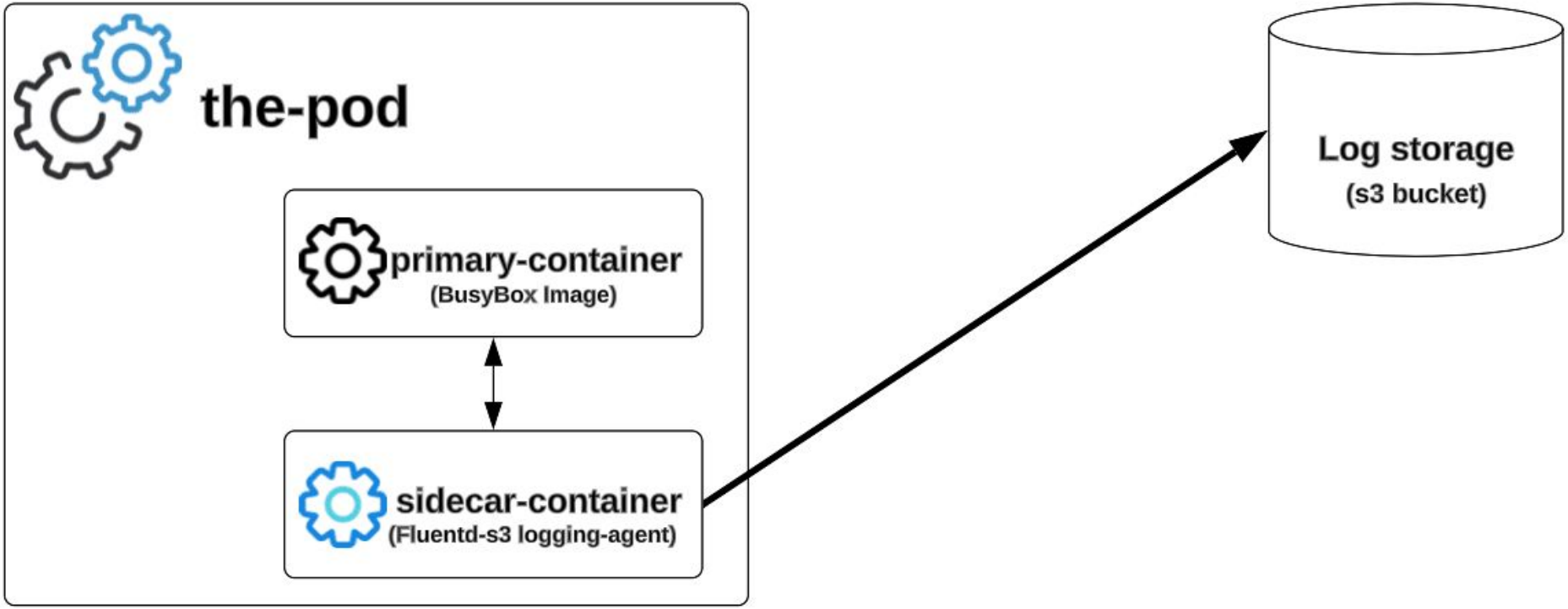
OnFailure >>

SideCar Container

Sidecar containers are the secondary containers that run along with the main application container within the same Pod.

A sidecar container can collect logs generated by the primary containers and forward them to a centralized logging system.

It can also capture metrics and monitoring data from the primary containers and send them to a monitoring system or dashboard.

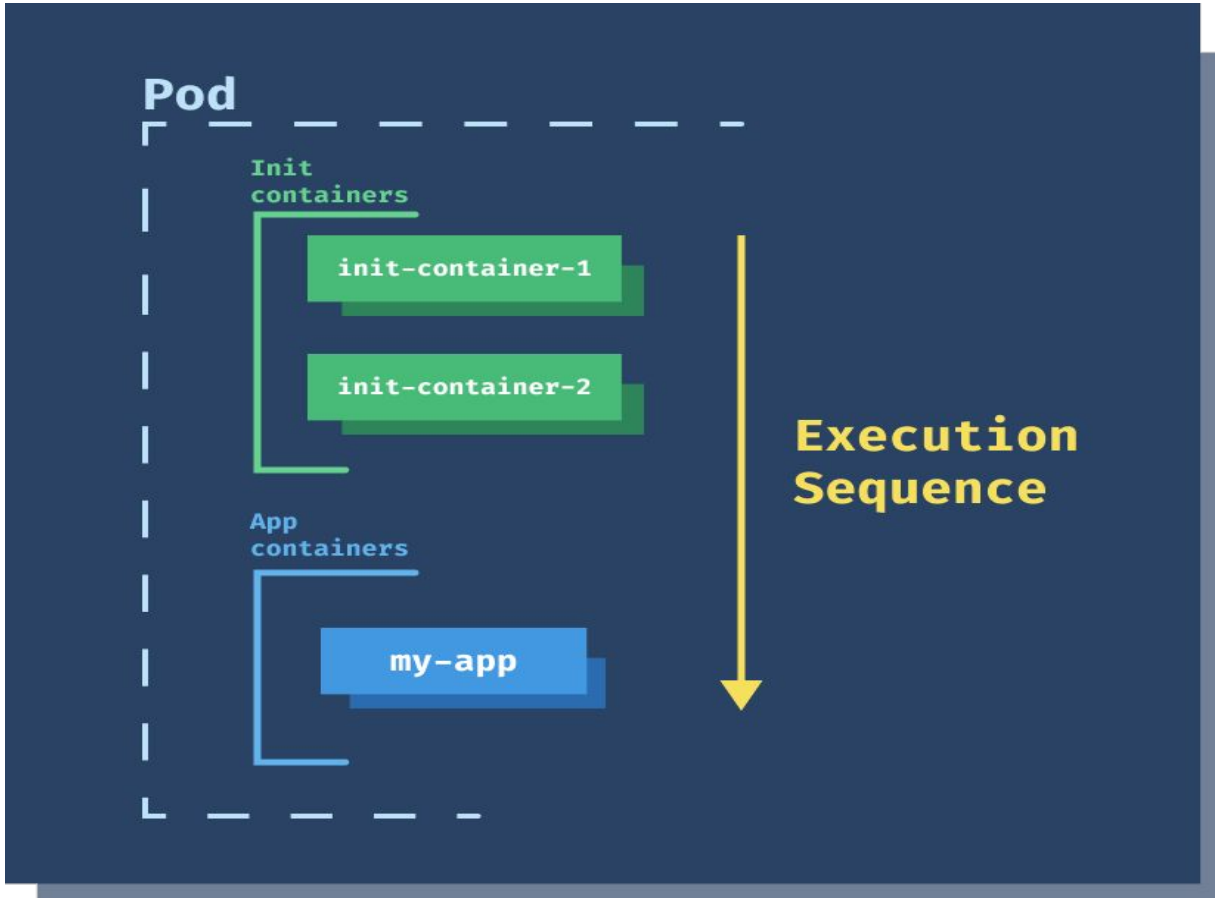

the-pod
primary-container
(BusyBox Image)
sidecar-container
(Fluentd-s3 logging-agent)
Log storage
(s3 bucket)

initContainer

initContainer

specialized containers that run before app containers in a Pod.

Init containers can contain utilities or setup scripts not present in an app image. You can specify init containers in the Pod specification alongside the containers array (which describes app containers)


Pod
Init containers
init-container-1
init-container-2
App containers
my-app
Execution Sequence

Kubernetes service

In Kubernetes, a Service is a method for exposing a network application that is running as one or more Pods in your cluster.

Since pods are ephemeral, a service enables a group of pods, which provide specific functions (web services, image processing, etc.) to be assigned a name and unique IP address (clusterIP). As long as the service is running that IP address, it will not change. Services also define policies for their access.

Types of Service

ClusterIP

NodePort

Loadbalancer

ExternalName

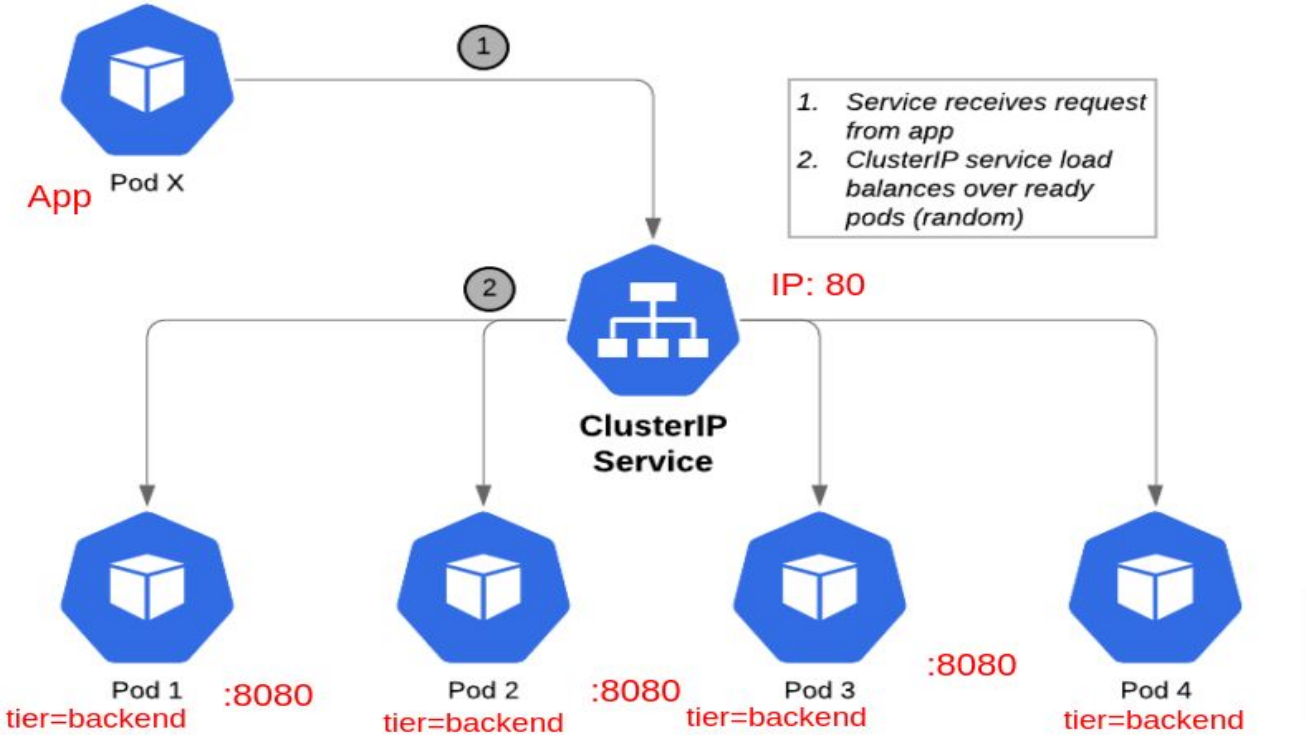
ClusterIP

The ClusterIP provides a load-balanced IP address. One or more pods that match a label selector can forward traffic to the IP address. The ClusterIP service must define one or more ports to listen on with target ports to forward TCP/UDP traffic to containers.

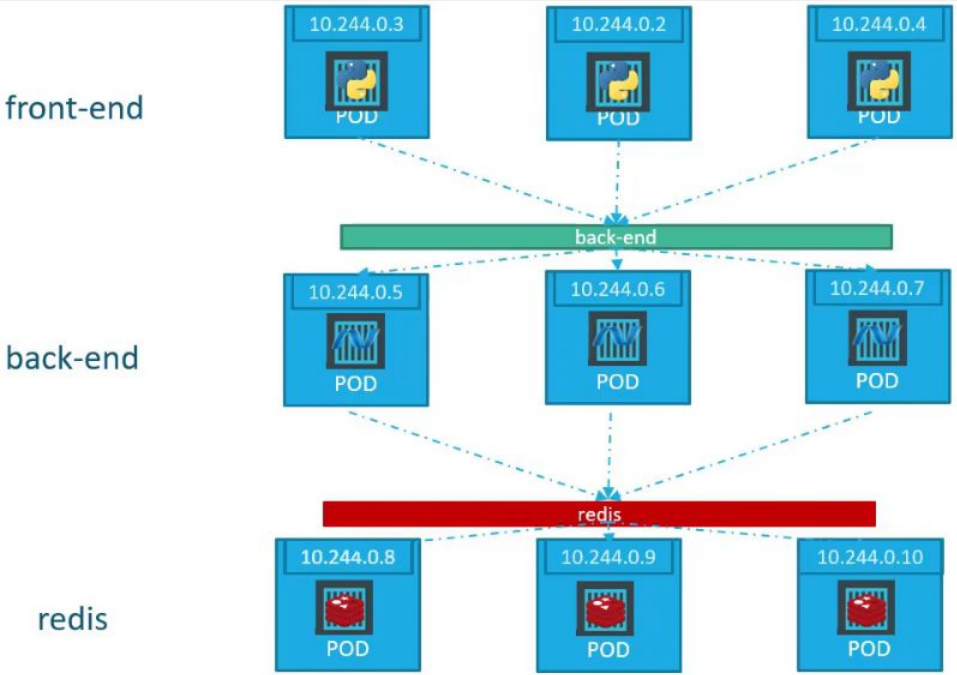
ClusterIP is used to establish connectivity between different applications internally within the Kubernetes providing a stable IP address for communication.

Kubernetes cluster, ClusterIP service type is the default service in Kubernetes and if no **type** is defined in the definition file, it is assumed that the service is ClusterIP.

Kubernetes Cluster



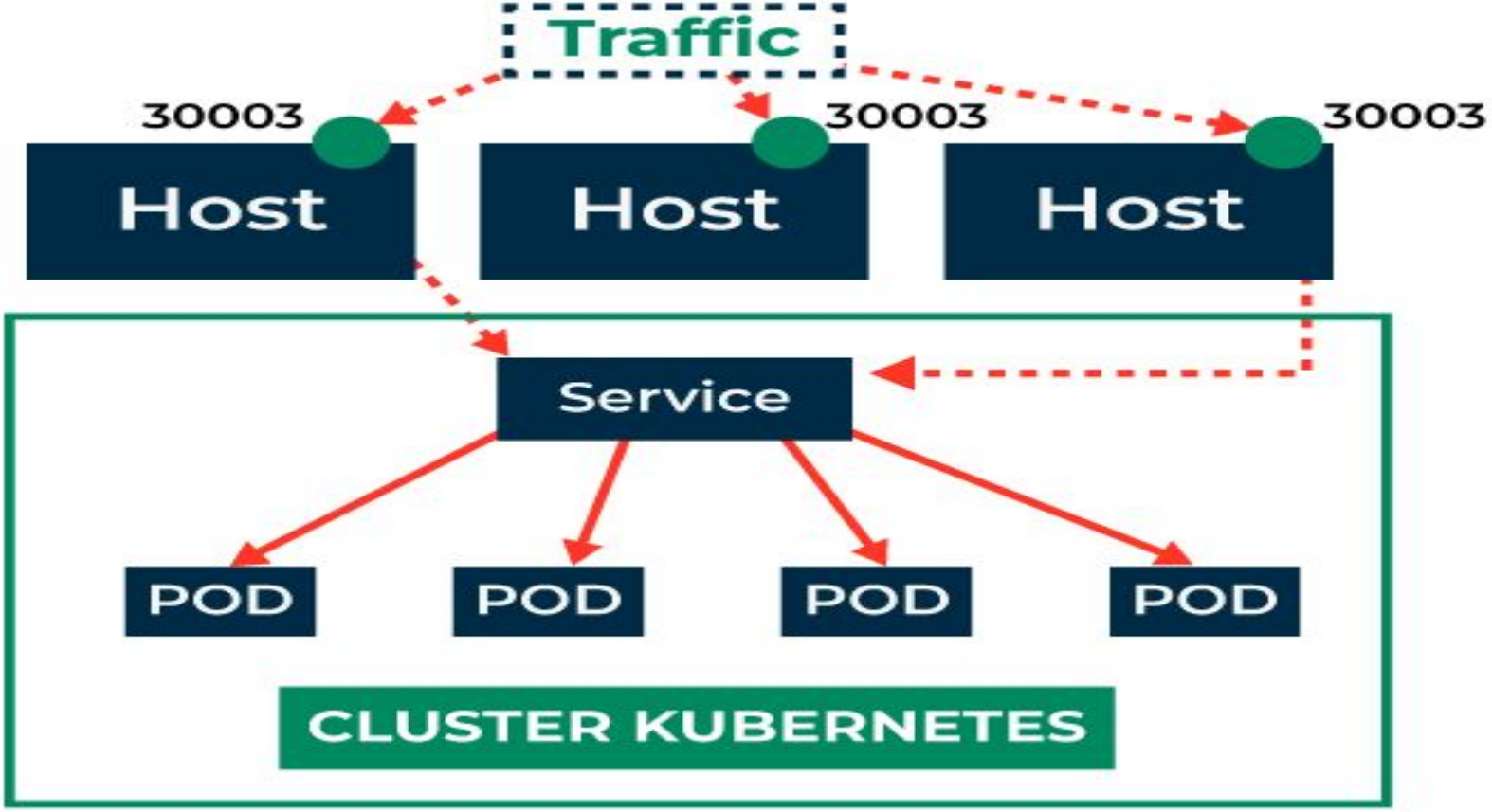
ClusterIP


front-end
back-end
redis
10.244.0.3
10.244.0.2
10.244.0.4
POD
POD
POD
back-end
10.244.0.5
10.244.0.6
10.244.0.7
POD
POD
POD
redis
10.244.0.8
10.244.0.9
10.244.0.10
POD
POD
POD

NodePort

NodePort builds on top of the ClusterIP Service and provides a way to expose a group of Pods to the outside world. At the API level, the only difference from the ClusterIP is the mandatory service type which has to be set to `NodePort`, the rest of the values can remain the same.

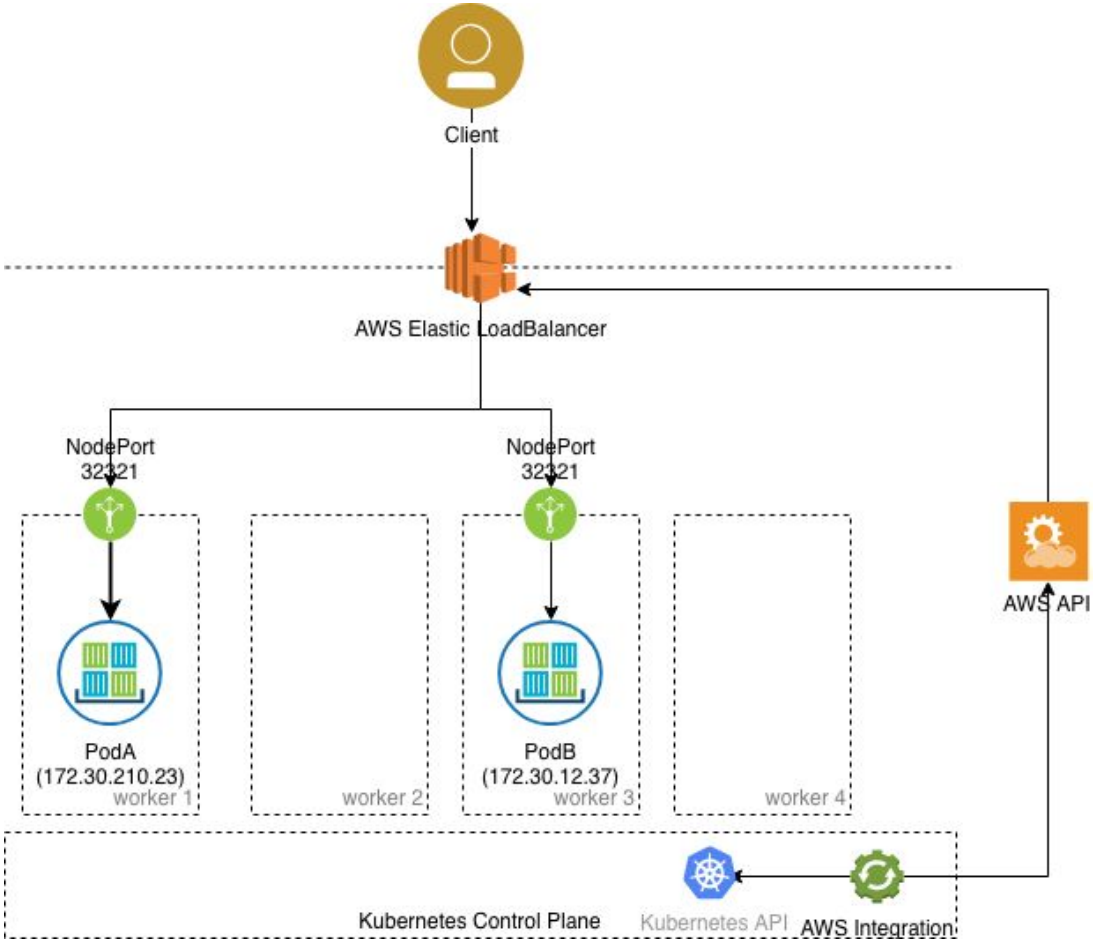
ClusterIP Services allow internal connectivity between different applications within the cluster, providing a stable IP address for communication

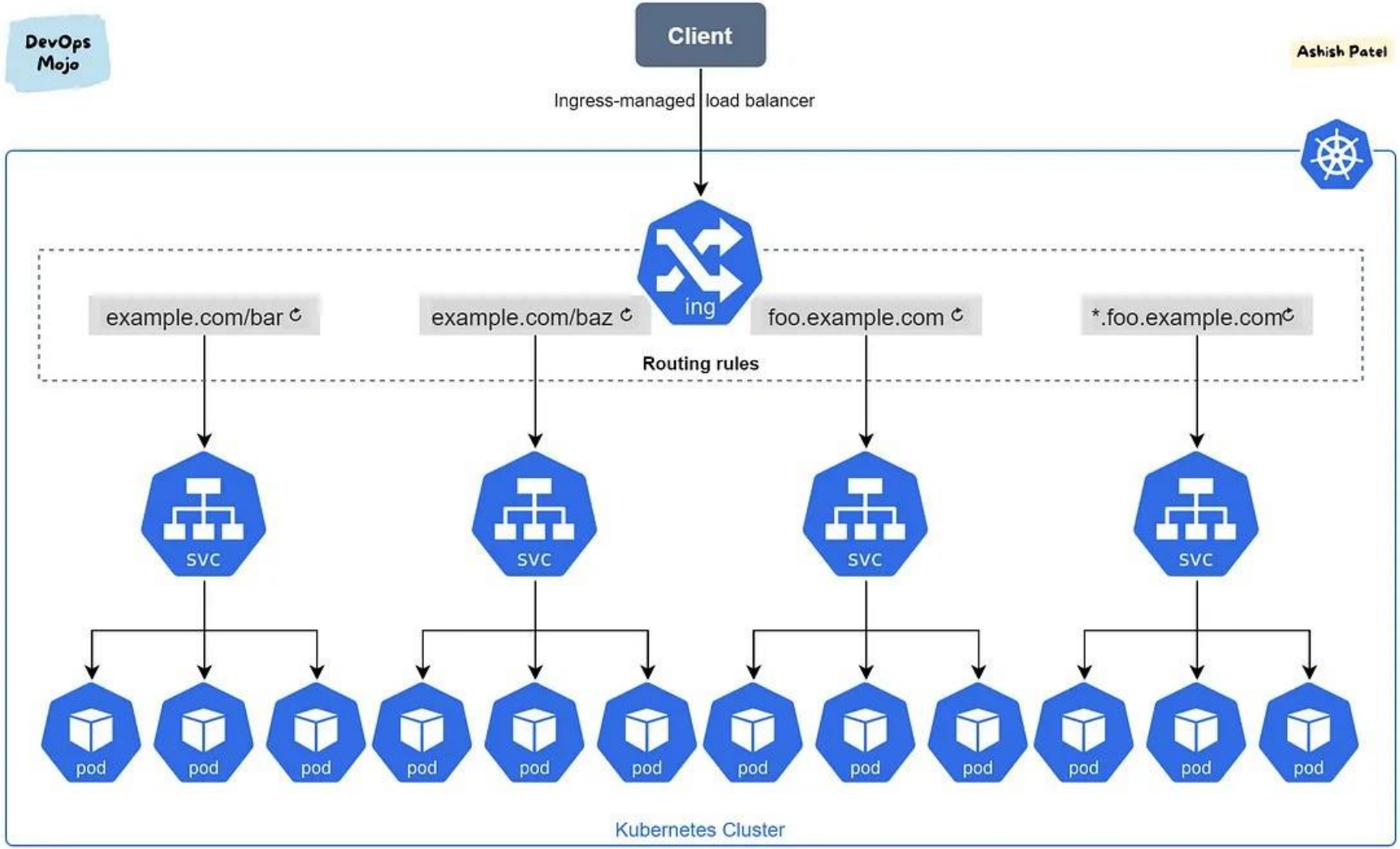

Traffic
30003
30003
30003
Host
Host
Host
Service
POD
POD
POD
POD
CLUSTER KUBERNETES

LoadBalancer

The `LoadBalancer` service type is built on top of `NodePort` service types by provisioning and configuring external load balancers from public and private cloud providers. It exposes services that are running in the cluster by forwarding layer 4 traffic to worker nodes. This is a dynamic way of implementing a case that involves external load balancers and `NodePort` type services.

LoadBalancer service in specially for cloud platforms.

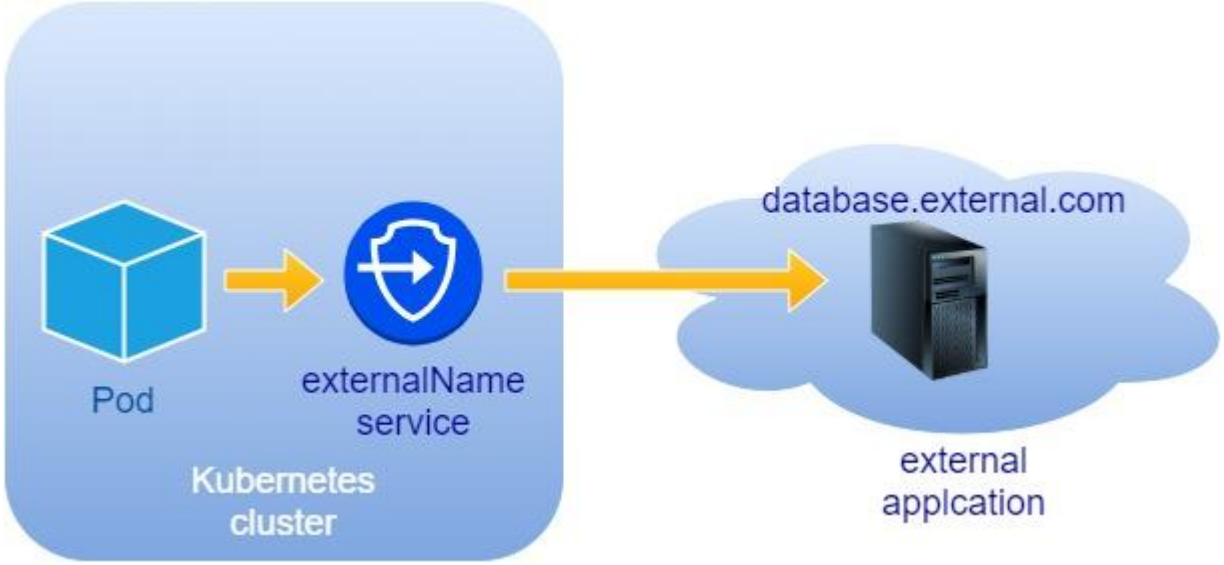

Client
AWS Elastic LoadBalancer
NodePort
32321
NodePort
32321
AWS API
PodA
(172.30.210.23)
worker 1
worker 2
PodB
(172.30.12.37)
worker 3
worker 4
Kubernetes Control Plane
Kubernetes API
AWS Integration


DevOps Mojo
Client
Ashish Patel
Ingress-managed load balancer
ing
example.com/bar
example.com/baz
foo.example.com
*.foo.example.com
Routing rules
SVC
SVC
SVC
SVC
pod
pod
pod
pod
pod
pod
pod
pod
pod
pod
pod
pod
Kubernetes Cluster

ExternalName

The ExternalName service in Kubernetes is a specialized type of Service that allows you to map an in-cluster service name to an external hostname or IP address outside of your Kubernetes cluster. This feature enables seamless integration between your Kubernetes applications and external services or resources, providing a bridge between your containerized workloads and the external environment.

> The ExternalName service is particularly useful in scenarios where you need to access services hosted outside your Kubernetes cluster, migrate legacy applications to Kubernetes while maintaining dependencies on external systems, or create hybrid deployments where some components run in Kubernetes while others run on external systems. By mapping an internal service name to an external resource, you can transparently access external services from within your Kubernetes applications, simplifying the integration process and enabling a smoother transition to a containerized environment.


database.external.com
Pod
externalName
service
Kubernetes
cluster
external
application

Use Cases

The ExternalName service is useful in the following scenarios:

1. Accessing external services: If you need to access a service that is hosted outside your Kubernetes cluster, you can use an ExternalName service to map an internal service name to the external service's hostname or IP address.
2. Migrating to Kubernetes: When migrating an application to Kubernetes, you may have dependencies on external services that cannot be moved to the cluster immediately. The ExternalName service allows you to maintain these dependencies while gradually migrating the application to Kubernetes.
3. Hybrid deployments: If you have a hybrid deployment where some components run in Kubernetes and others run on external systems, you can use ExternalName services to connect the in-cluster components to the external ones.