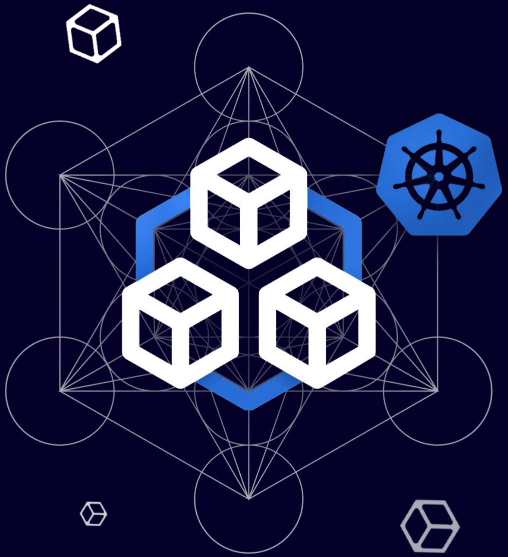
platformer™

Understanding Kubernetes

Service Discovery & Ingresses

Binura Gunasekara

SE @Platformer
Google Certified Cloud Architect
Kubernetes Certified Administrator



platformer.cloud

Kubernetes doesn’t solve all the world’s ills - @jbeda

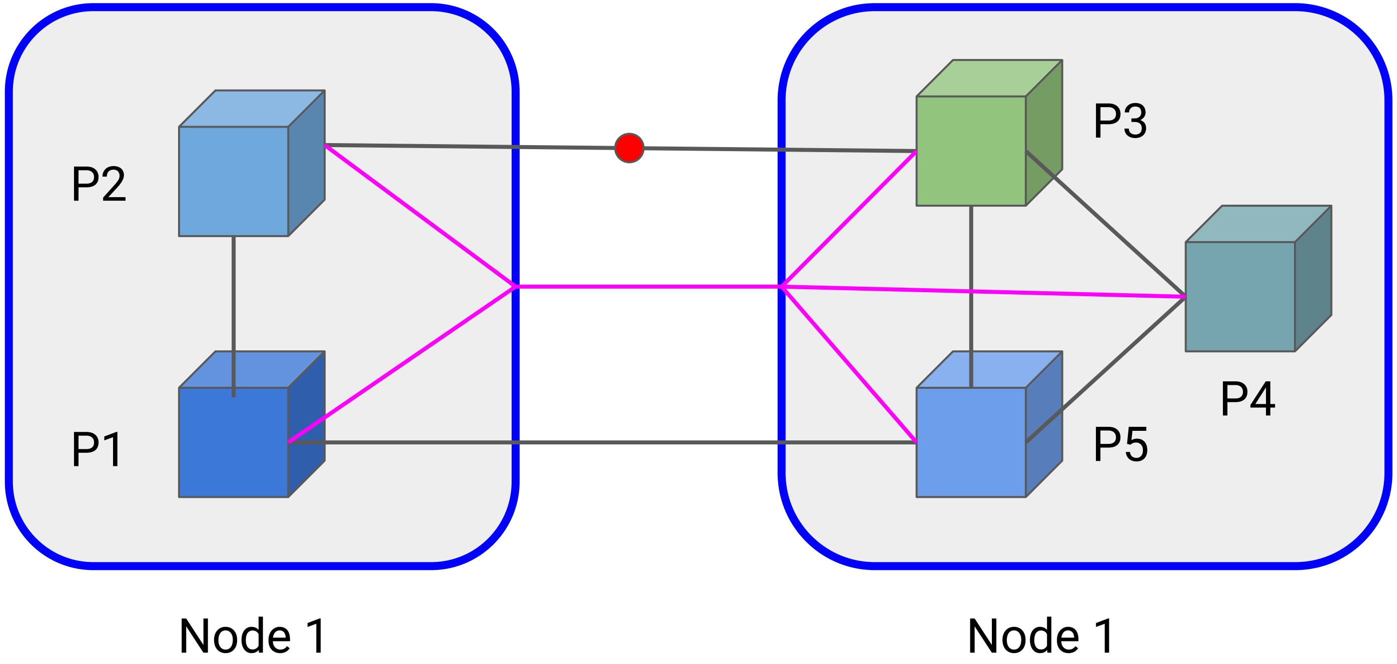
The **3 Commandments** of the k8s Network Model

platformer™

- All Pods can communicate with all other Pods without using *network address translation* (NAT).
- All Nodes can communicate with all Pods without NAT.
- The IP that a Pod sees itself as is the same IP that others see it as.

platformer™

Conceptually...

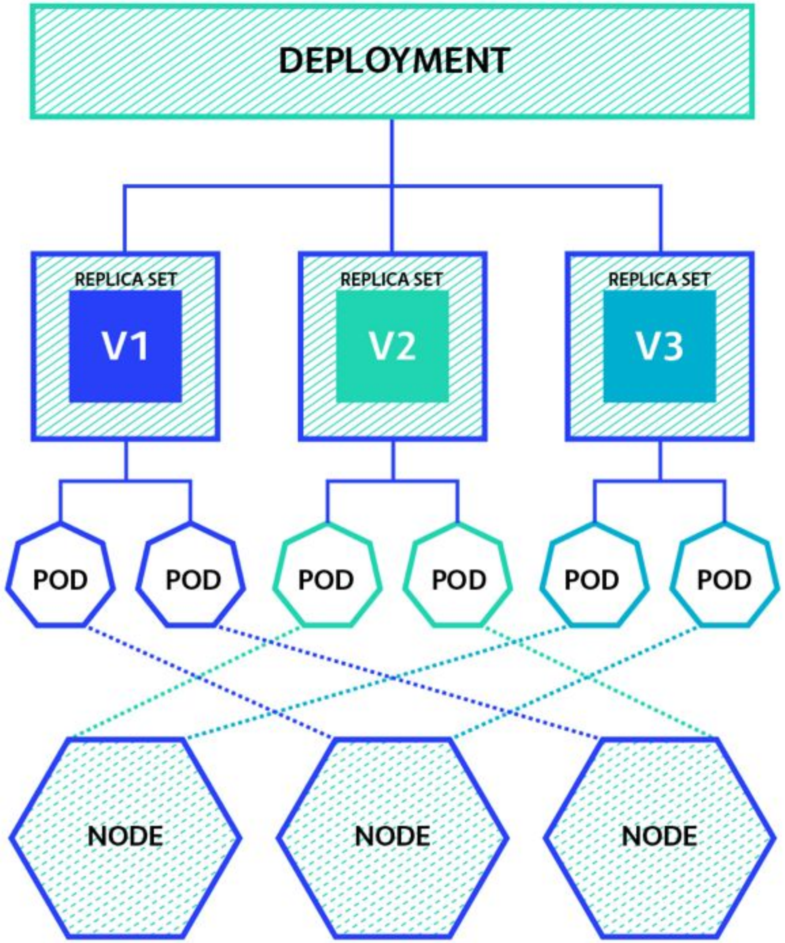


platformer™

Understanding Services

Deployments Refresher

How do you we expose a number of running pods with a single point of entry?



Services

`ClusterIP`:

- Exposes the service on a cluster-internal IP. Choosing this value makes the service only reachable from within the cluster. This is the default `ServiceType`.
- Go-to Service type for internal communication between applications/microservices.

Services

platformer™

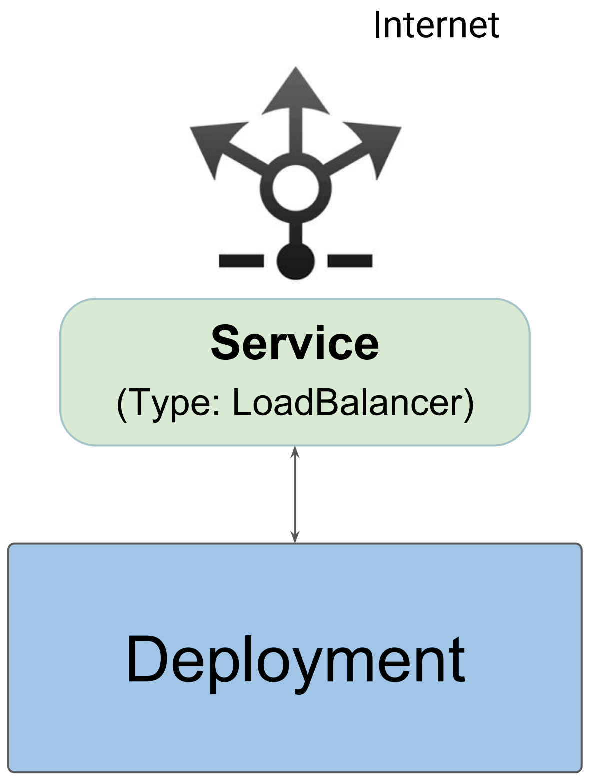
`NodePort`

- Exposes the service on each Node’s IP at a static port (the `NodePort`). A `ClusterIP` service, to which the `NodePort` service will route, is automatically created. You’ll be able to contact the `NodePort` service, from outside the cluster, by requesting `<NodeIP>:<NodePort>`.
- Highly dependent on the Node’s availability. Not recommended for Production

Services

`LoadBalancer`

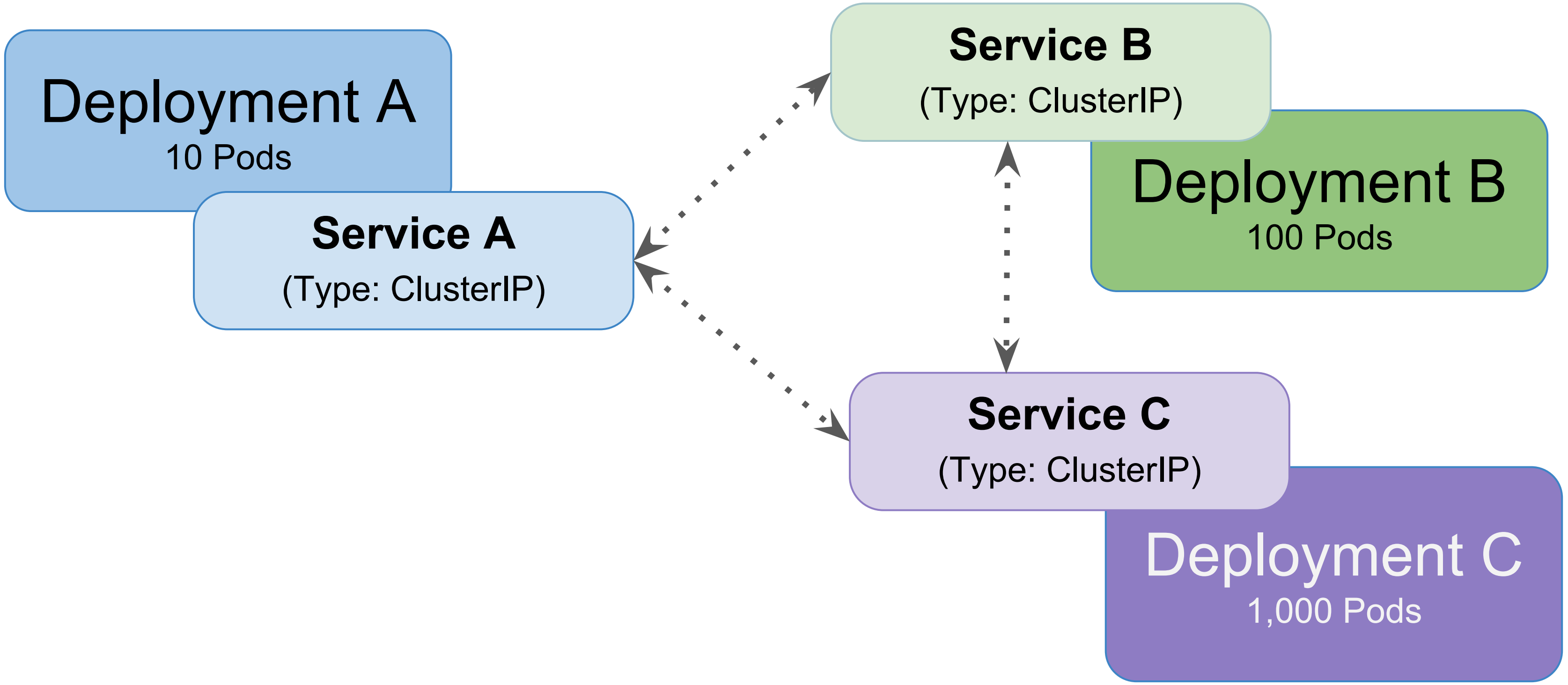
- Exposes the service externally using a **cloud provider's** load balancer. `NodePort` and `ClusterIP` services, to which the external load balancer will route, are automatically created.
- Requires a cloud provider. (There are on-prem solutions but they cannot be requested on demand with a LoadBalancer Type - requires a manual intervention)
- **Is it possible to share a LB between multiple applications? YES - Ingresses!**



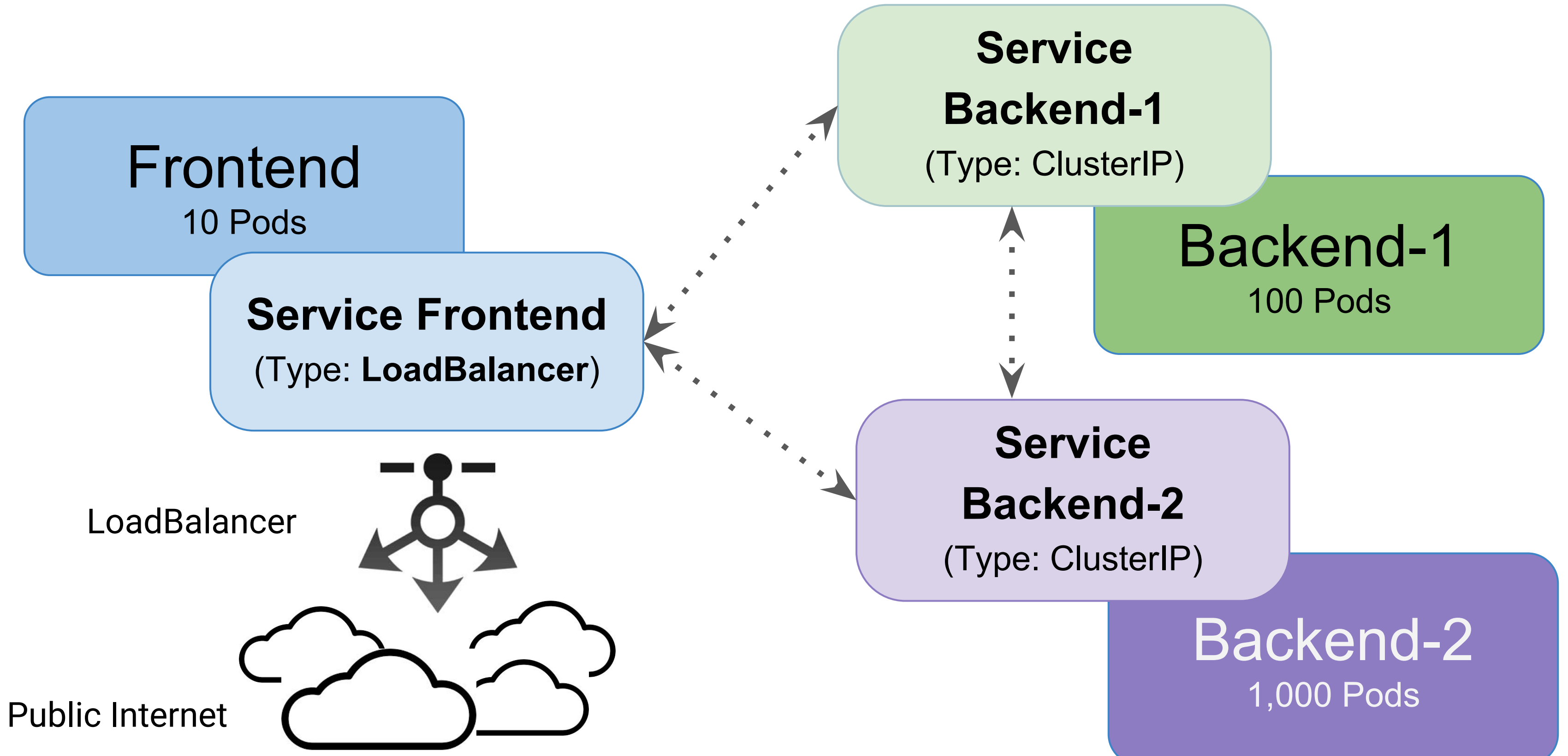
Manages the internal **NodePorts** and **ClusterIPs** as an *abstraction* -

Availability of Nodes is no longer a concern. Kubernetes does the mapping for us.

ClusterIP - Internal Services



Example Design



Service Discovery

There are **2** ways to discover Services

DNS

- A Static DNS is created per each Service. (mapped to the ClusterIP by K8s internally).
- *Eg.* service-a can be reached by another Pod/Node using `http://service-a`
- If in a different namespace, `http://service-a.<namespace>`

Environment Variables

- The Pod must be created after the service.
- The Kubelet adds a set of environment variables pointing to each Service ClusterIP to the Pod.

Eg.

```
SERVICE-A_PORT_80_TCP_ADDR=10.51.250.22
SERVICE-A_PORT_80_TCP_PORT=80
SERVICE-A_PORT_80_TCP_PROTO=tcp
```

Demo Time!

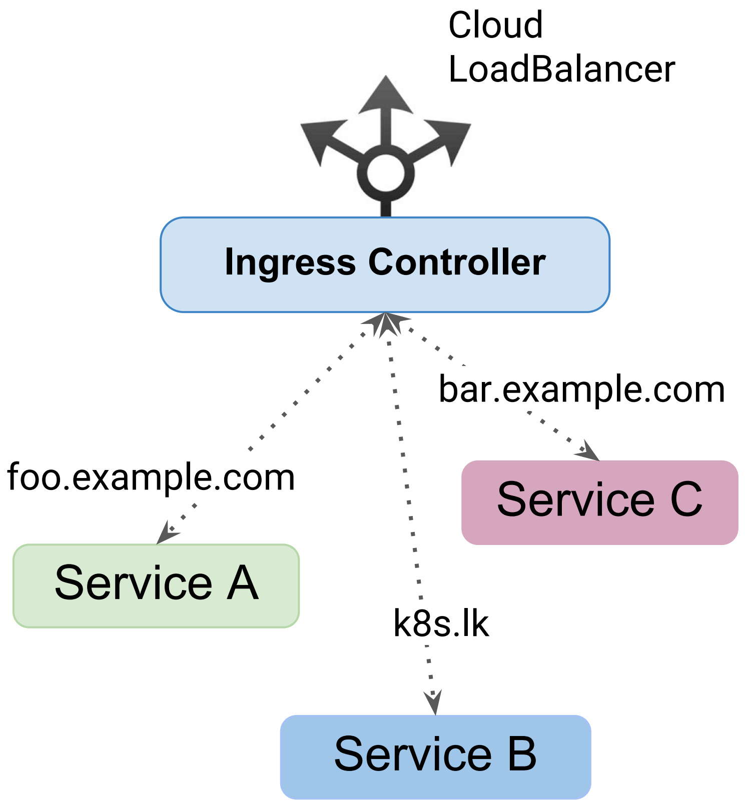
Ingresses

Ingesting Ingresses

- Ingresses expose HTTP/HTTPS routes from outside the cluster, to one or many services running inside the cluster.

- Popular use-cases:
  - Utilize a single Cloud Loadbalancer to serve external traffic to multiple applications.
  - Fan-out ingresses - quite useful when working with microservice-like architectures.

- Requires an **Ingress Controller** and a set of **Ingress Resources.**

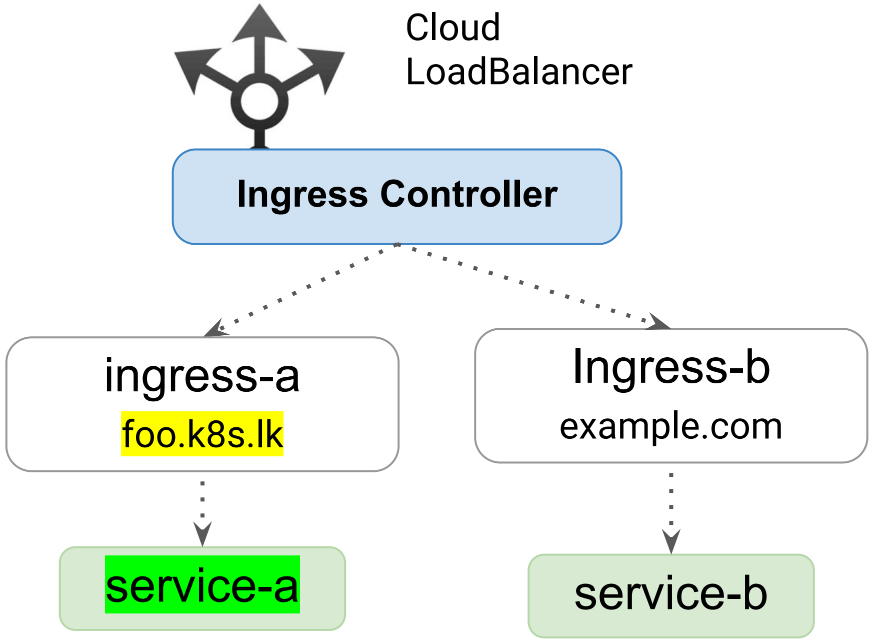


The Ingress Controller

- Acts as a plugin Controller to the **kube-controller-manager.**
- Popular Ingress Controllers - *NGNIX, Traeffic, Istio Gateway* (The Nginx Ingress controller is different to the normal Ngnix pods you've seen today.)
- Installing it is just a matter of **kubectl apply -f.**
- **An <u>Ingress Controller</u> relies on a set of other resources called '<u>Ingresses</u>' to figure out where to route which traffic.**

Single-service Ingress (Resource)

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-a
spec:
  rules:
  - host: foo.k8s.lk
    http:
      paths:
      - path: /
        backend:
          serviceName: service-a
          servicePort: 80
```



How do you point different DNS names at the Ingress Controller?

platformer™

An Ingress Controller is assigned an **external IP** on creation. (This is in fact the Load Balancer's IP address - eg. 98.100.54.19).

Simply add the required CNAMEs to this IP Address. *And remember to make this IP static.*

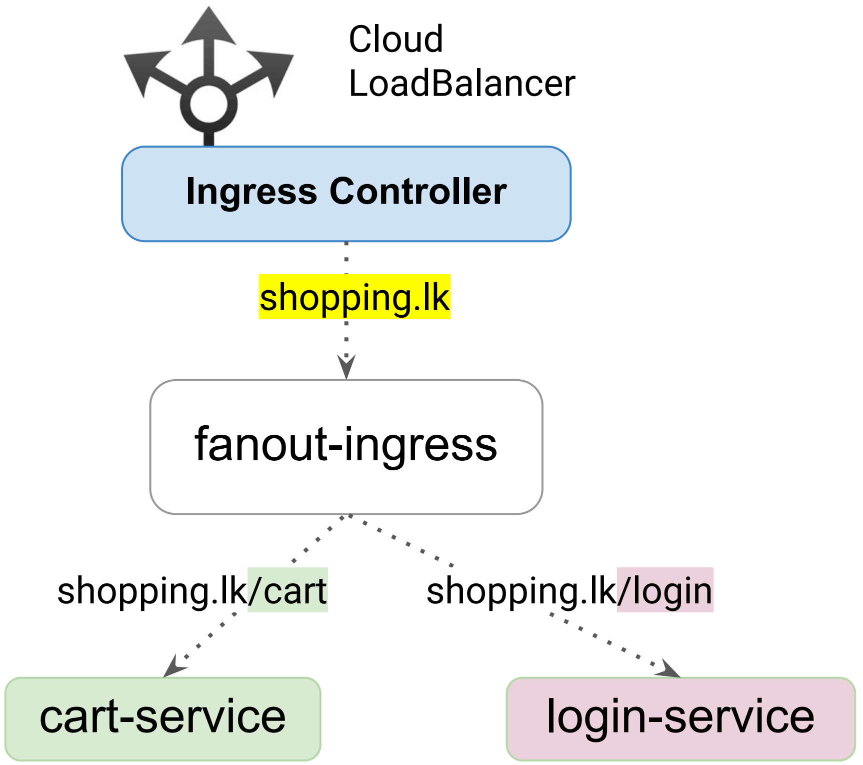
CNAME RECORD	IP ADDRESS
foo.example.com	98.100.54.19
k8s.lk	98.100.54.19
bar.example.com	98.100.54.19

Fan-out Ingress (Resource)

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: fanout-ingress-example
spec:
  rules:
  - host: shopping.lk
    http:
      paths:

      - path: /cart
        backend:
          serviceName: cart-service
          servicePort: 5000

      - path: /login
        backend:
          serviceName: login-service
          servicePort: 8080
```



Thanks!

- Get these slides from **https://github.com/BinuraG**
- Refer to the Docs at **kubernetes.io/docs**

Contact us through platformer.com for consultation.

binura.g@platformer.com