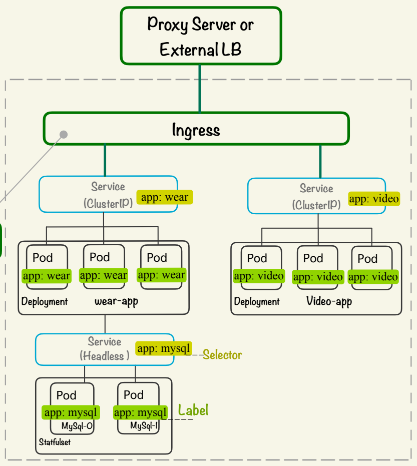
Ingress

Ingress is an API object in Kubernetes that allows access to your Kubernetes services from outside the Kubernetes cluster. It provides load balancing, SSL termination and name-based virtual hosting for your services, In other words, it's a way for your applications to expose URLs to the outside world.

Ingress provides external reachable URLs, SSL termination and name-based virtual hosting to services in the cluster. This means you can route requests to different services based on the request host or path.

Ingress provides layer 7 load balancing. It acts as a reverse proxy and load balances traffic to different services in your Kubernetes cluster

Ingress object allows you to expose multiple services through a single IP address



If you use the LoadBalancer service type, the service is made available to clients outside the cluster through a load balancer. This approach is fine if you only need to expose a single service externally, but it becomes problematic with large numbers of services, since each service needs its own public IP address.Fortunately, by exposing these services through an Ingress object instead, you only need a single IP address.

Ingress consists of two main components:

- **Ingress resource** is a Kubernetes API object that defines the rules for how external traffic should be directed to services within a cluster. The ingress resource specifies the rules for routing traffic based on the host name, path, and other criteria. It also specifies the backend services that should receive the traffic.
- **Ingress controller** is responsible for implementing the rules defined in the ingress resource and handling external traffic based on these rules. Ingress controllers like Nginx use ConfigMaps to store the configuration for the ingress resources and dynamically generate Nginx configuration based on the rules defined in the ingress resource

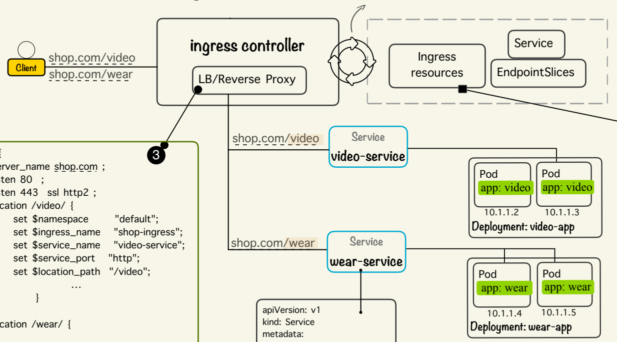
Kubernetes only provides the Ingress resource and needs a separate Ingress Controller to satisfy the Ingress. There are several options available, but for the purpose of this guide, we'll use the Nginx Ingress Controller.Install the Nginx Ingress Controller

In order for the Ingress resource to work, the cluster must have an ingress controller running.Unlike other types of controllers which run as part of the kube-controller-manager binary, Ingress controllers are not started automatically with a cluster. You have to select an Ingress Controller compatible with your setup and start it manually. (The actual implementation of Ingress is done by Ingress Controllers)

How Does an Ingress Controller Work?
Here's a simplified view of how an Ingress Controller works:

1 You define an Ingress Resource in your cluster, which has a set of routing rules associated with it.
2 The Ingress Controller continuously watches for updates to Ingress Resources , Service, and Endpoints or EndpointSlice objects. When it detects a new or modified these objects, the controller is notified. it reads the information in these objects to understand what traffic routing changes it needs to make.
3 The Ingress Controller configures the load balancer to implement the desired traffic routing.

2 The Ingress Controller continuously watches for updates to Ingress Resources, Service, and Endpoints or EndpointSlice objects.



When an ingress resource is created or updated, the ingress controller reads the configuration information from the corresponding ConfigMap and generates the configuration for the load balancer based on the rules defined in the ingress resource. The load balancer configuration is then dynamically updated to reflect the changes in the ingress resource.

1 Ingress resource is the YAML configuration that defines the rules for routing traffic.

```
server {
      server_name shop.com ;
      listen 80  ;
      listen 443  ssl http2 ;
      location /video/ {
           set $namespace       "default";
           set $ingress_name    "shop-ingress";
           set $service_name    "video-service";
           set $service_port    "http";
           set $location_path   "/video";
                     ...
                }

      location /wear/ {

           set $namespace       "default";
           set $ingress_name    "shop-ingress";
           set $service_name    "wear-service";
           set $service_port    "http";
           set $location_path   "/wear";
    }       }
```

Nginx.conf

```
apiVersion: v1
kind: Service
metadata:
  name: wear-service
spec:
  selector:
    app: wear
  ports:
   - name: http
     port: 80
     targetPort: 8080
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-path
spec:
  rules:
   - host: shop.com
     http:
       paths:
         - path: /video
           pathType: Prefix
           backend:
             service:
               name: video-service
               port:
                 name: http
         - path: /wear
           pathType: Prefix
           backend:
             service:
               name: wear-service
               port:
                 number: 80
```

The ingress controller uses the service name specified in the ingress rules to lookup the IP addresses of the pods backing that service. It then routes traffic to those pods according to the path matching rules defined in the ingress

Ingress resources define rules for routing HTTP/HTTPS traffic to services in a Kubernetes cluster. They specify path-based rules that map URLs to backend services.

```
kubectl describe ingress ingress-path
Name:              ingress-path
Namespace:          default
Address:
Default backend:   <default>
Rules:
  Host          Path  Backends
  ----          ----  --------
  shop.com
                /video   video-service:80 (10.1.1.2:5678, 10.1.1.3:5678)
                /wear    wear-service:80 (10.1.1.4:5678, 10.1.1.5:5678)
Events:          <none>
```

when configuring an Ingress resource, the "backend" field specifies the service that should receive the forwarded traffic. However, it's important to note that the traffic never directly reaches the service itself. Instead, controller uses service endpoints to route the traffic, not the service.

there is one rule specified for the `shop.com` host, and under that rule, there are two paths (`/video` and `/wear`) defined for routing traffic to their respective backend services.

Ingress controllers often include a default backend component that handles traffic that doesn't match any Ingress rules.

How to customize Nginx Ingress Controller?

Helm Chart Values: If deploying the Ingress controller via Helm chart, you can customize settings by overriding chart values. The Helm chart exposes many config settings as values.
ConfigMap: using a ConfigMap to set global configuration in NGINX, For example, you can specify custom log formats, change timeout values, enable features like GeoIP, etc
Annotation: use this if you want a specific configuration for a particular ingress rule.

How to enable Basic Authentication for an ingress rule in Kubernetes?

This example shows how to add authentication in a Ingress rule using a secret that contains a file generated with htpasswd

1 Generate the base64 encoded user/pass combo:

```
htpasswd -nbm arye Heisenberg | base64
```

2 Convert htpasswd into a secret:

```
kubectl create secret generic shop-basic-auth --from-literal=auth=<base64 output>
```

Or

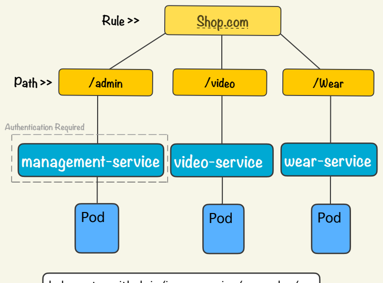
```
apiVersion: v1
kind: Secret
metadata:
  name: shop-basic-auth
  namespace: default
data:
  auth: YXJ5ZTokYXByMSQxbzAzWElTQiRJVFYudWh0dmcuVmV5d0t5a0s1cC4vCgo=
```

Or

```
kubectl create secret generic shop-basic-auth --from-literal=username=arye --from-literal=password=Heisenberg
```

3 Configure the Ingress rule to use the basic authentication secret

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: shop.com-admin
  namespace: default
  annotations:
      kubernetes.io/ingress.class: "nginx"
      nginx.ingress.kubernetes.io/auth-type: basic
      nginx.ingress.kubernetes.io/auth-secret: shop-basic-auth
      nginx.ingress.kubernetes.io/auth-realm: Authentication Required
spec:
  rules:
   - host: shop.com
     http:
       paths:
         - path: /admin
           pathType: Prefix
           backend:
             service:
               name: management-service
               port:
                 name: http
```



kubernetes.github.io/ingress-nginx/examples/
You can find more examples in this link