**Ingress Controller**

An **Ingress Controller** in Kubernetes is a specialized component that manages Ingress resources to handle external HTTP(S) traffic and route it to the appropriate backend services within a cluster. It acts as a **reverse proxy or load balancer** to enable applications running in Kubernetes to be accessible from outside the cluster

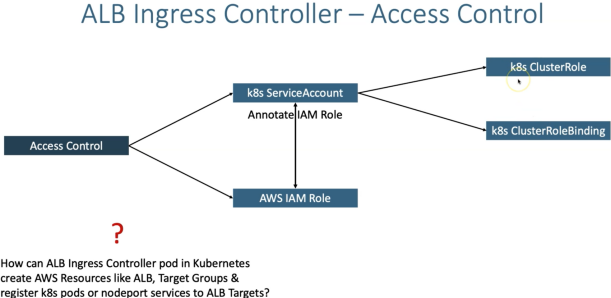
Most Widely used ingress controllers

\* AWS - ALB ingress controller (Application load balancer)

\* Nginx ingress controller

\* HA-proxy ingress

So using ingress contollers kubernetes is going to perform tasks outside the cluster so it would require certain previleges attached we have already discussed in service account section where API to API communiication would be done when certain priveleges are attached to it



We will be annotating our IAM role with already created K8s service account so that when ingress controller pod is deployed it can perform actions outside cluster

Lets see an example of how ingress works using ALBIC ( Application Load Balancer Ingress Controller)

**Official Documentation** :

<https://kubernetes-sigs.github.io/aws-load-balancer-controller/latest/>

**Detailed Steps for installations in cluster** :

[https://github.com/bezardo/aws-eks-kubernetes-masterclass/tree/master/08-NEW-ELB-Application-LoadBalancers/08-01-Load-Balancer-Controller-Install](https://github.com/bezardo/aws-eks-kubernetes-masterclass/tree/master/08-NEW-ELB-Application-LoadBalancers/08-01-Load-Balancer-Controller-Install" \o "https://github.com/bezardo/aws-eks-kubernetes-masterclass/tree/master/08-new-elb-application-loadbalancers/08-01-load-balancer-controller-install" \t "_blank)

**The main purpose of using application load balancers are**

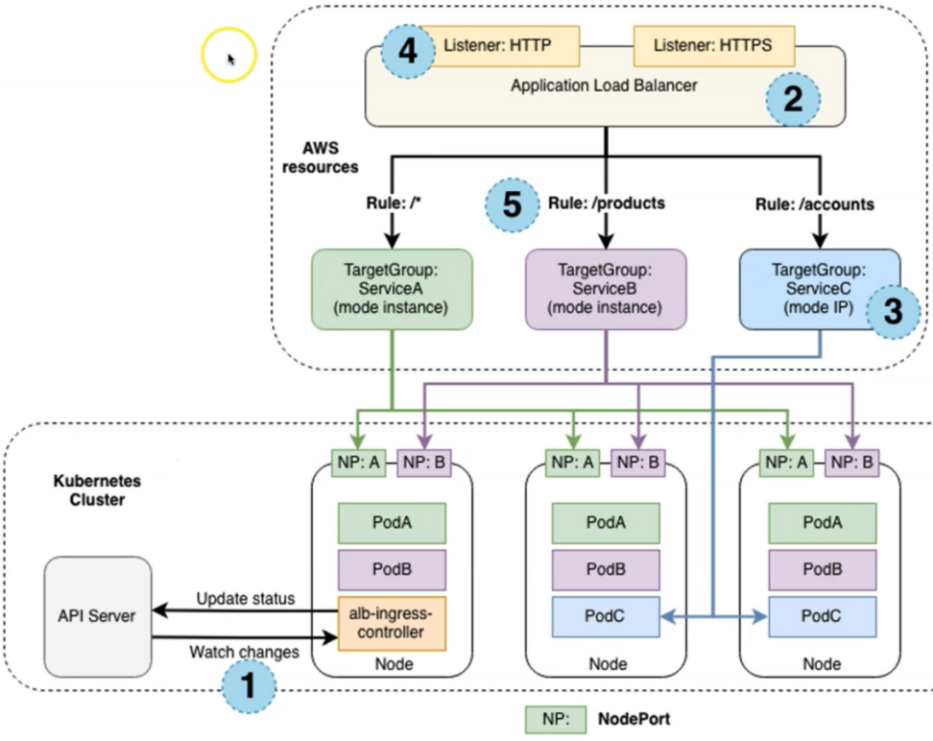
it enables to run our application in layer 7 which handles requests liike http/https , and can handle SSL/TLS certificates , Websocket etc

Provides use of context based routing and granular rules can be defined for our microservice applications

**Resource manifests we need to implement are**

**IngressClass**

**Ingress**



\* once ALBIC is deployed API server watches for ingress resources and builds the application load balancer , we define the context path rules in ingress manifest

**IngressClass Manifest**

The main purpose os using ingress class are in realtime along with ALB ingress controller there might also be different interal ingress contollers or proxy controllers can be run in same cluster so to define the ingress controller for specific application we define ingressclasses

apiVersion: networking.k8s.io/v1

kind: IngressClass

metadata:

  name: my-aws-ingress-class

  annotations:

    ingressclass.kubernetes.io/is-default-class: "true"

spec:

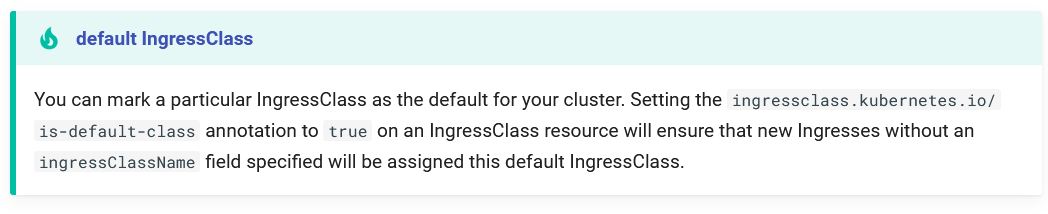
  controller: ingress.k8s.aws/alb

**Additional Note**

You can mark a particular IngressClass as the default for your cluster. Setting the ingressclass.kubernetes.io/is-default-class annotation to true on an IngressClass resource will ensure that new Ingresses without an ingressClassName field specified will be assigned this default IngressClass.

**Reference**:

<https://kubernetes-sigs.github.io/aws-load-balancer-controller/v2.3/guide/ingress/ingress_class/>



**Ingress manifest**

Important compenents in Ingress manifests are : Annotations , ingressClassName , rules

**# Annotations Reference**: <https://kubernetes-sigs.github.io/aws-load-balancer-controller/latest/guide/ingress/annotations/>

If only one application is running in cluster it follows default path /\*

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  name: ingress-nginxapp1

  labels:

    app: app1-nginx

  annotations:

    # Load Balancer Name

    alb.ingress.kubernetes.io/load-balancer-name: app1ingressrules

    # Ingress Core Settings

    alb.ingress.kubernetes.io/scheme: internet-facing

    # Health Check Settings

    alb.ingress.kubernetes.io/healthcheck-protocol: HTTP

    alb.ingress.kubernetes.io/healthcheck-port: ‘80’

    alb.ingress.kubernetes.io/healthcheck-path: /

    alb.ingress.kubernetes.io/healthcheck-interval-seconds: '15'

    alb.ingress.kubernetes.io/healthcheck-timeout-seconds: '5'

    alb.ingress.kubernetes.io/success-codes: '200'

    alb.ingress.kubernetes.io/healthy-threshold-count: '2'

    alb.ingress.kubernetes.io/unhealthy-threshold-count: '2'

spec:

  ingressClassName: my-aws-ingress-class # Ingress Class

  rules:

  - http:

      paths:

      - path: /

        pathType: Prefix

        backend:

          service:

            name: app1-nginx-nodeport-service

            port:

              number: 80

If  "spec.ingressClassName: my-aws-ingress-class" not specified, will reference default ingress class on this kubernetes cluster Default Ingress class is nothing but for which ingress class we have the annotation `ingressclass.kubernetes.io/is-default-class: "true"`

**For MicroService context path based routing**

# Annotations Reference: https://kubernetes-sigs.github.io/aws-load-balancer-controller/latest/guide/ingress/annotations/

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  name: ingress-cpr-demo

  annotations:

    # Load Balancer Name

    alb.ingress.kubernetes.io/load-balancer-name: cpr-ingress

    # Ingress Core Settings

    alb.ingress.kubernetes.io/scheme: internet-facing

    # Health Check Settings

    alb.ingress.kubernetes.io/healthcheck-protocol: HTTP

    alb.ingress.kubernetes.io/healthcheck-port: 80

**#Important Note:  Need to add health check path annotations in service level if we are planning to use multiple targets in a load balancer**

    alb.ingress.kubernetes.io/healthcheck-interval-seconds: '15'

    alb.ingress.kubernetes.io/healthcheck-timeout-seconds: '5'

    alb.ingress.kubernetes.io/success-codes: '200'

    alb.ingress.kubernetes.io/healthy-threshold-count: '2'

    alb.ingress.kubernetes.io/unhealthy-threshold-count: '2'

spec:

  ingressClassName: my-aws-ingress-class   # Ingress Class

  rules:

    - http:

        paths:

          - path: /app1

            pathType: Prefix

            backend:

              service:

                name: app1-nginx-nodeport-service

                port:

                  number: 80

          - path: /app2

            pathType: Prefix

            backend:

              service:

                name: app2-nginx-nodeport-service

                port:

                  number: 80

          - path: /

            pathType: Prefix

            backend:

              service:

                name: app3-nginx-nodeport-service

                port:

                  number: 80

**Important Note** : In path based routing order is very important, if we are going to use  "/\*" (Root Context), try to use it at the end of all rules.

If "spec.ingressClassName: my-aws-ingress-class" not specified, will reference default ingress class on this kubernetes cluster

Default Ingress class is nothing but for which ingress class we have the annotation `ingressclass.kubernetes.io/is-default-class: "true"`

If there are multiple paths we need to define the annotations in nodeport service itself say for example health check path needs to be provided for all the context path so we cannot define them as generic annotation in ingress manifest

**NodePort svc for app 3**

apiVersion: v1

kind: Service

metadata:

  name: app3-nginx-nodeport-service

  labels:

    app: app3-nginx

  annotations:

**#Important Note:  Need to add health check path annotations in service level if we are planning to use multiple targets in a load balancer**

    alb.ingress.kubernetes.io/healthcheck-path: /index.html

spec:

  type: NodePort

  selector:

    app: app3-nginx

  ports:

    - port: 80

      targetPort: 80

**NodePort svc for app 2**

apiVersion: v1

kind: Service

metadata:

  name: app2-nginx-nodeport-service

  labels:

    app: app2-nginx

  annotations:

**#Important Note:  Need to add health check path annotations in service level if we are planning to use multiple targets in a load balancer**

    alb.ingress.kubernetes.io/healthcheck-path: /app2/index.html

spec:

  type: NodePort

  selector:

    app: app2-nginx

  ports:

    - port: 80

      targetPort: 80

**NodePort svc for app 1**

apiVersion: v1

kind: Service

metadata:

  name: app1-nginx-nodeport-service

  labels:

    app: app1-nginx

  annotations:

**#Important Note:  Need to add health check path annotations in service level if we are planning to use multiple targets in a load balancer**

    alb.ingress.kubernetes.io/healthcheck-path: /app1/index.html

spec:

  type: NodePort

  selector:

    app: app1-nginx

  ports:

    - port: 80

      targetPort: 80

**Ingress SSL**

Further if we want to secure the http requests we can register a domain and create a SSL certificate for the domain

## **Pre-requisite - Register a Domain in Route53 (if not exists)**

* Goto Services -> Route53 -> Registered Domains
* Click on **Register Domain**
* Provide **desired domain: somedomain.com** and click on **check** (In my case its going to be stacksimplify.com)
* Click on **Add to cart** and click on **Continue**
* Provide your **Contact Details** and click on **Continue**
* Enable Automatic Renewal
* Accept **Terms and Conditions**
* Click on **Complete Order**

## **Create a SSL Certificate in Certificate Manager**

* Pre-requisite: You should have a registered domain in Route53
* Go to Services -> Certificate Manager -> Create a Certificate
* Click on **Request a Certificate**
  + Choose the type of certificate for ACM to provide: Request a public certificate
  + Add domain names: \*.yourdomain.com
  + Select a Validation Method: **DNS Validation**
  + Click on **Confirm & Request**
* **Validation**
  + Click on **Create record in Route 53**
* Wait for 5 to 10 minutes and check the **Validation Status**

**SSL Settings in ingress manifest**

alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'

alb.ingress.kubernetes.io/certificate-arn: <cert\_arn>

To avoid defining our certificate arn in annotation we can directly specify the hosts in rules

rules:

**- host: app102.stacksimplify.com**

      http:

        paths:

          - path: /

            pathType: Prefix

            backend:

              service:

                name: app1-nginx-nodeport-service

                port:

                  number: 80

In case of multiple context path it would be tedious to define hosts for all so we can define it also with tls

spec:

  ingressClassName: my-aws-ingress-class   # Ingress Class

**tls:**

**- hosts:**

**- "\*.stacksimplify.com"**

  defaultBackend:

    service:

      name: app3-nginx-nodeport-service

      port:

        number: 80

  rules:

    - http:

        paths:

          - path: /app1

            pathType: Prefix

            backend:

              service:

                name: app1-nginx-nodeport-service

                port:

                  number: 80

Run the manifests again now we have to create a DNS record in R53 manually with our new registerd domain and provide the alias target to our ELB ALB

**SSL redirect**

To redirect https to https request we have to add below annotation in our ingress manifest

**# SSL Redirect Setting**

alb.ingress.kubernetes.io/ssl-redirect: '443'

**Note** : If we want to set custom rules like redirect to a dns , forward to particular target group we can do it via actions annotation : <https://kubernetes-sigs.github.io/aws-load-balancer-controller/latest/guide/ingress/annotations/#actions>

**External-DNS**

From above ingress ssl topic we have manually created the record sets and attached our elb target group we can further instruct our kube api server to create the same with ‘externaldns’ service which are similar to ALBIC where we deploy a pod ‘external-dns’ and provide necesaary service account and role permissions to create and manage DNS settings

**Official documentation** : <https://github.com/kubernetes-sigs/external-dns/blob/master/docs/tutorials/aws.md>

**Installation steps** : <https://github.com/bezardo/aws-eks-kubernetes-masterclass/tree/master/08-NEW-ELB-Application-LoadBalancers/08-06-Deploy-ExternalDNS-on-EKS>

After installation verify the pods and define below annotation in our ingress manifest

external-dns.alpha.kubernetes.io/hostname: dnstest901.stacksimplify.com, dnstest902.stacksimplify.com

Similarly if we are using a classic or network load balancers we can add annotation at service level

apiVersion: v1

kind: Service

metadata:

  name: app1-nginx-loadbalancer-service

  labels:

    app: app1-nginx

  annotations:

    external-dns.alpha.kubernetes.io/hostname: externaldns-k8s-service-demo101.stacksimplify.com

spec:

  type: LoadBalancer

  selector:

    app: app1-nginx

  ports:

    - port: 80

      targetPort: 80

We have seen path based routing below is the example for namebasedVirtual host routing

# Annotations Reference: https://kubernetes-sigs.github.io/aws-load-balancer-controller/latest/guide/ingress/annotations/

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  name: ingress-namedbasedvhost-demo

  annotations:

    # Load Balancer Name

    alb.ingress.kubernetes.io/load-balancer-name: namedbasedvhost-ingress

    # Ingress Core Settings

    alb.ingress.kubernetes.io/scheme: internet-facing

    # Health Check Settings

    alb.ingress.kubernetes.io/healthcheck-protocol: HTTP

    alb.ingress.kubernetes.io/healthcheck-port: traffic-port

    #Important Note:  Need to add health check path annotations in service level if we are planning to use multiple targets in a load balancer

    alb.ingress.kubernetes.io/healthcheck-interval-seconds: '15'

    alb.ingress.kubernetes.io/healthcheck-timeout-seconds: '5'

    alb.ingress.kubernetes.io/success-codes: '200'

    alb.ingress.kubernetes.io/healthy-threshold-count: '2'

    alb.ingress.kubernetes.io/unhealthy-threshold-count: '2'

    ## SSL Settings

    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'

    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-1:180789647333:certificate/632a3ff6-3f6d-464c-9121-b9d97481a76b

    #alb.ingress.kubernetes.io/ssl-policy: ELBSecurityPolicy-TLS-1-1-2017-01 #Optional (Picks default if not used)

    # SSL Redirect Setting

    alb.ingress.kubernetes.io/ssl-redirect: '443'

    # External DNS - For creating a Record Set in Route53

    external-dns.alpha.kubernetes.io/hostname: default101.stacksimplify.com

spec:

  ingressClassName: my-aws-ingress-class   # Ingress Class

  defaultBackend:

    service:

      name: app3-nginx-nodeport-service

      port:

        number: 80

rules:

    - host: app101.stacksimplify.com

      http:

        paths:

          - path: /

            pathType: Prefix

            backend:

              service:

                name: app1-nginx-nodeport-service

                port:

                  number: 80

    - host: app201.stacksimplify.com

      http:

        paths:

          - path: /

            pathType: Prefix

            backend:

              service:

                name: app2-nginx-nodeport-service

                port:

                  number: 80

**Important Note :**

In path based routing order is very important, if we are going to use  "/\*", try to use it at the end of all rules.

If  "spec.ingressClassName: my-aws-ingress-class" not specified, will reference default ingress class on this kubernetes cluster

Default Ingress class is nothing but for which ingress class we have the annotation `ingressclass.kubernetes.io/is-default-class: "true"`

**Other important annotations**

# Target Type: IP

    alb.ingress.kubernetes.io/target-type: ip alb.ingress.kubernetes.io/target-type: instance

# Creates Internal Application Load Balancer

    alb.ingress.kubernetes.io/scheme: internal