## Deploy nginx ingress using helm chart

To install NGINX Ingress Controller using Helm, you first need to add the NGINX Helm repository:

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

Create nginx ingress namespace

kubectl create namesapce ingress-nginx

Then, you can install the controller with the default values:

helm install my-ingress-nginx ingress-nginx/ingress-nginx --namespace ingress-nginx

Run this command to watch the Load Balancer become available:

kubectl --namespace ingress-nginx get services -o wide -w ingress-nginx-controller

## Deploy nginx ingress using manifest fille

If you prefer using YAML manifests, you can deploy NGINX Ingress Controller directly onto your Kubernetes cluster. You need to apply the necessary YAML files provided by the NGINX Ingress project:

Create ingress nginx namespace:

kubectl create namesapce ingress-nginx

Then install nginx ingress using the kubernetes maintained manifest file:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.10.0/deploy/static/provider/aws/deploy.yaml -n ingress-nginx

This command will create the required Deployments, Services, ConfigMaps, and other resources to run the NGINX Ingress Controller.

two applications to kubernetes cluster:

Create ingress-testing namespace to deploy the testing applciations:

kubectl create namespace ingress-testing

## Deploy Blogging Application:

Run below command to deploy blogging application in ingress-testing namespace:

kubectl -n ingress-testing apply -f https://raw.githubusercontent.com/anveshmuppeda/kubernetes-ingress/main/nginx-ingress/hands-on/blogging-application/deploy.yaml

## Deploy Streaming Application:

Run below command to deploy streaming application in ingress-testing namespace:

kubectl -n ingress-testing apply -f https://raw.githubusercontent.com/anveshmuppeda/kubernetes-ingress/main/nginx-ingress/hands-on/streaming-application/deploy.yaml

Let’s verify the both application pods and respective services using below command, make sure both are up and running:

kubectl get pods,svc -n ingress-testing

kubectl get po,svc -n ingress-testing

NAME READY STATUS RESTARTS AGE

pod/blog-app-678866dcf9-t5cn9 1/1 Running 0 2m6s

pod/stream-app-6fd5c45d5d-s9sxr 1/1 Running 0 5m7s

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

service/blogging-svc ClusterIP 10.245.24.189 <none> 80/TCP 2m6s

service/streaming-svc ClusterIP 10.245.127.41 <none> 80/TCP 5m7s

Now that we’ve successfully deployed both applications, it’s time to make them accessible to the outside world using Ingress. We’ll utilize the default Ingress class, which is our Ingress NGINX, to route traffic from the internet to our target applications.

Here’s how the traffic flow will work:

1. Traffic from the internet will first reach the Ingress NGINX Load Balancer.
2. From there, it will be directed to the Ingress NGINX Controller.
3. Finally, the Ingress NGINX Controller will route the traffic to the respective target applications using their respective Ingress definitions.

Create blog-ingress.yaml using below content:

|  |
| --- |
| apiVersion: networking.k8s.io/v1 |

|  |
| --- |
| kind: Ingress |

|  |
| --- |
| metadata: |

|  |
| --- |
| name: blogging-ingress |

|  |
| --- |
| labels: |

|  |
| --- |
| app: blogging-application |

|  |
| --- |
| annotations: |

|  |
| --- |
| nginx.ingress.kubernetes.io/add-base-url: "true" |

|  |
| --- |
| nginx.ingress.kubernetes.io/rewrite-target: /$1 |

|  |
| --- |
| spec: |

|  |
| --- |
| ingressClassName: nginx |

|  |
| --- |
| rules: |

|  |
| --- |
| - # host: "http://<your-domain.com>" |

|  |
| --- |
| http: |

|  |
| --- |
| paths: |

|  |
| --- |
| path: /blogs/(.\*) |

|  |
| --- |
| pathType: Prefix |

|  |
| --- |
| backend: |

|  |
| --- |
| service: |

|  |
| --- |
| name: blogging-svc |

|  |
| --- |
| port: |

number: 80

annotations: Contains additional configuration options for the Ingress. In this case, it includes annotations specific to the NGINX Ingress Controller.

* nginx.ingress.kubernetes.io/add-base-url: "true": This annotation indicates that the NGINX Ingress Controller should add a base URL to the request.
* nginx.ingress.kubernetes.io/rewrite-target: /$1: This annotation specifies how NGINX should rewrite the target path for incoming requests. The $1 refers to a captured group from the regular expression in the path.

ingressClassName: nginx: Specifies the Ingress class to use for this Ingress. Here, it's set to nginx, indicating that the NGINX Ingress Controller should handle this Ingress.

# host: "http://<your-domain.com>"

Need to apply the yaml fail

kubectl apply -f blog-ingress.yaml -n ingress-testing

**stream-ingress.yaml**

|  |
| --- |
| apiVersion: networking.k8s.io/v1 |

|  |
| --- |
| kind: Ingress |

|  |
| --- |
| metadata: |

|  |
| --- |
| name: streaming-ingress |

|  |
| --- |
| labels: |

|  |
| --- |
| app: streaming-application |

|  |
| --- |
| annotations: |

|  |
| --- |
| nginx.ingress.kubernetes.io/add-base-url: "true" |

|  |
| --- |
| nginx.ingress.kubernetes.io/rewrite-target: /$1 |

|  |
| --- |
| spec: |

|  |
| --- |
| ingressClassName: nginx |

|  |
| --- |
| rules: |

|  |
| --- |
| - # host: "http://<your-domain.com>" |

|  |
| --- |
| http: |

|  |
| --- |
| paths: |

|  |
| --- |
| - path: /streaming/(.\*) |

|  |
| --- |
| pathType: Prefix |

|  |
| --- |
| backend: |

|  |
| --- |
| service: |

|  |
| --- |
| name: streaming-svc |

|  |
| --- |
| port: |

number: 80

kubectl apply -f stream-ingress.yaml -n ingress-testing

verify using the below command:

kubectl get ingress -n ingress-testing

$ kubectl get ingress -n ingress-testing

NAME CLASS HOSTS ADDRESS PORTS AGE

blogging-ingress nginx \* 161.35.240.187 80 34m

streaming-ingress nginx \* 161.35.240.187 80 32m

## Access the blogging and streamin applications using below URL’s

**Blogging application**: http://<loadbalancer-url>/blogs/

**Streaming Application:** http://<loadbalancer-url>/streaming/

