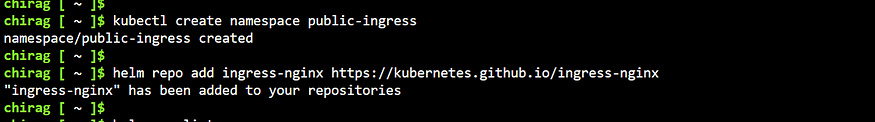
This guide demonstrates the process of setting up the NGINX ingress controller within an Azure Kubernetes Service (AKS) cluster. The ingress controller is specifically configured to utilize a fixed public IP address via an Azure Standard Load Balancer. To handle certificate management, the **cert-manager project is deployed**, automating the generation and setup of **Let’s Encrypt certificates**. Furthermore, it outlines the steps to integrate a custom domain with a certificate, allowing the application to run publicly.

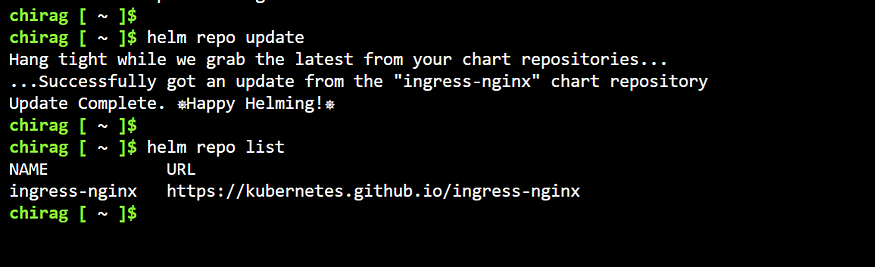
**Steps :**

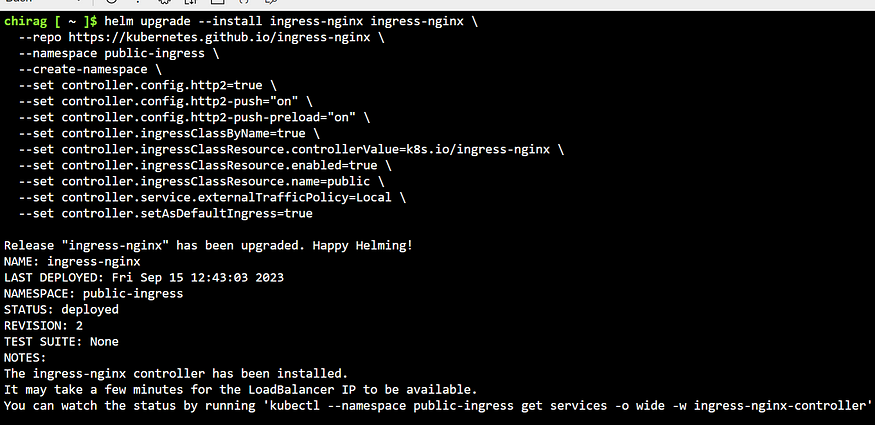
* Create a namespace **public-ingress** for Ingress Controller where all ingress controller-related resources will be created.
* Install cert-manager for SSL certificates in public-ingress namespace using Helm.
* Create a CA cluster issuer for issuing certificates.
* Create first application and service.
* Create Second application and service.
* Setup A Record of domain
* Create an ingress route to configure the rules that route traffic to one of the two applications.
* Verify the automatic created certificate.
* Test the applications using Custom Domain.

**Create an ingress controller**

# Create a namespace for ingress resources  
kubectl create namespace public-ingress  
  
# Add the Helm repository  
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx  
helm repo update  
  
# Use Helm to deploy an NGINX ingress controller  
helm upgrade --install ingress-nginx ingress-nginx \  
 --repo https://kubernetes.github.io/ingress-nginx \  
 --namespace public-ingress \  
 --set controller.config.http2=true \  
 --set controller.config.http2-push="on" \  
 --set controller.config.http2-push-preload="on" \  
 --set controller.ingressClassByName=true \  
 --set controller.ingressClassResource.controllerValue=k8s.io/ingress-nginx \  
 --set controller.ingressClassResource.enabled=true \  
 --set controller.ingressClassResource.name=public \  
 --set controller.service.externalTrafficPolicy=Local \  
 --set controller.setAsDefaultIngress=true

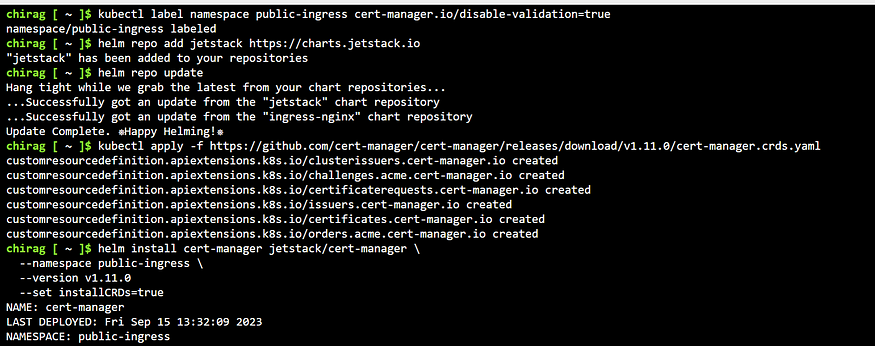






**Install cert-manager**

# Label the cert-manager namespace to disable resource validation  
kubectl label namespace public-ingress cert-manager.io/disable-validation=true  
# Add the Jetstack Helm repository  
helm repo add jetstack https://charts.jetstack.io  
# Update your local Helm chart repository cache  
helm repo update  
# Install CRDs with kubectl  
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.11.0/cert-manager.crds.yaml  
# Install the cert-manager Helm chart  
helm install cert-manager jetstack/cert-manager \  
 --namespace public-ingress \  
 --version v1.11.0

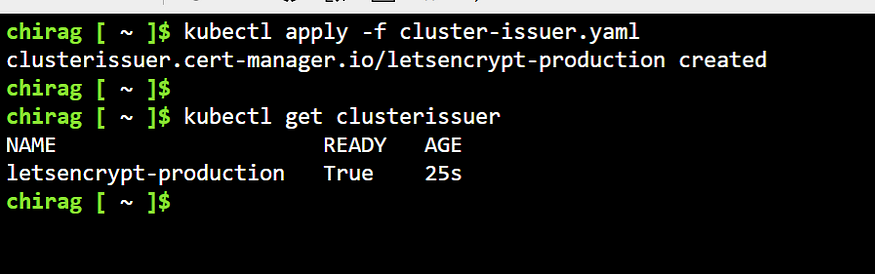


**Create a CA cluster issuer**

# Cluster Issuer  
apiVersion: cert-manager.io/v1  
kind: ClusterIssuer  
metadata:  
 name: letsencrypt-production  
spec:  
 acme:  
 server: https://acme-v02.api.letsencrypt.org/directory  
 email: #Use your mail id  
 privateKeySecretRef:  
 name: letsencrypt-production  
 solvers:  
 - http01:  
 ingress:  
 class: public

To create the issuer, use the kubectl command.

kubectl apply -f cluster-issuer.yaml --namespace public-ingress



**Run demo applications**

Create a *deployment-one.yaml* file and copy in the following example YAML:

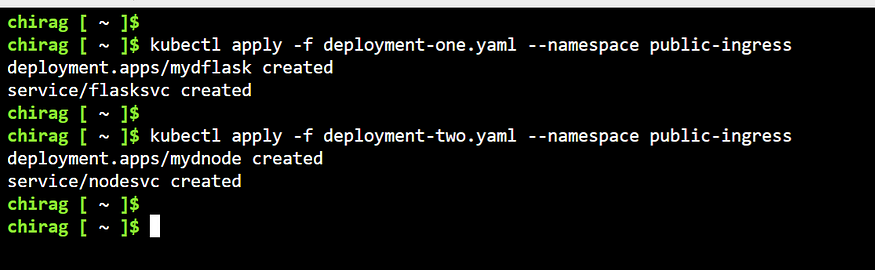
apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: mydflask  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: flaskapp  
 template:  
 metadata:  
 labels:  
 app: flaskapp  
 spec:  
 containers:  
 - name: myflaskc  
 image: chiragnagorijain/myflaskimg  
 ports:  
 - containerPort: 5000  
  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: flasksvc  
spec:  
 type: ClusterIP  
 ports:  
 - port: 80  
 targetPort: 5000  
 selector:  
 app: flaskapp

Create a *deployment-two.yaml* file and copy in the following example YAML:

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: mydnode  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: nodeapp  
 template:  
 metadata:  
 labels:  
 app: nodeapp  
 spec:  
 containers:  
 - name: mynodec  
 image: chiragnagorijain/mynodeimg:latest  
 ports:  
 - containerPort: 3000  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: nodesvc  
spec:  
 type: ClusterIP  
 ports:  
 - port: 80  
 targetPort: 3000  
 selector:  
 app: nodeapp

Run the two demo applications using kubectl:

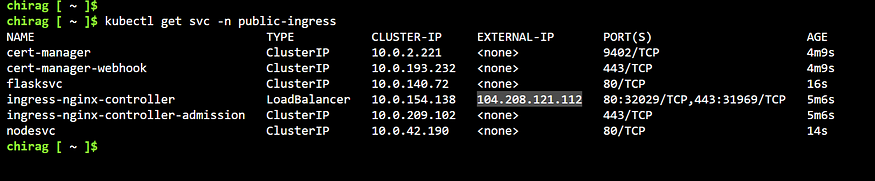
kubectl apply -f deployment-one.yaml --namespace public-ingress  
kubectl apply -f deployment-two.yaml --namespace public-ingress



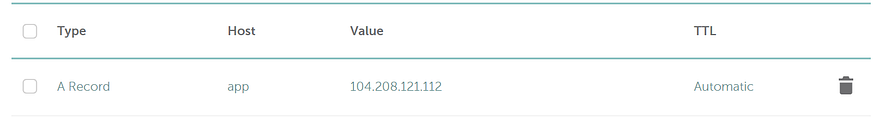
**Create an A record**

A record of custom domain will be the Public IP of Ingress Controller.

kubectl get svc -n public-ingress



Copy this external ip of nginx controller and create A-Record for you domain.

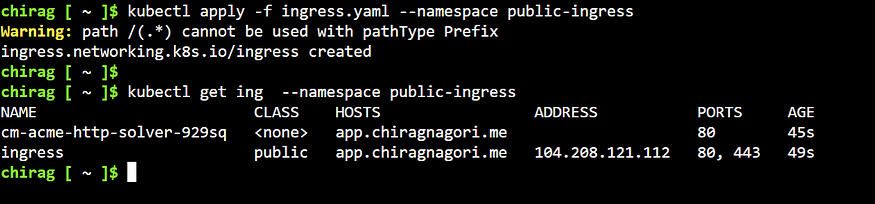


**Create an ingress route**

The ingress resource configures the rules that route traffic to one of the two applications.

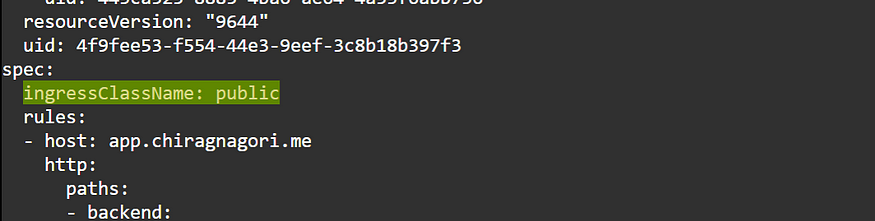
apiVersion: networking.k8s.io/v1  
kind: Ingress  
metadata:  
 name: ingress  
 annotations:  
 cert-manager.io/cluster-issuer: letsencrypt-production  
 nginx.ingress.kubernetes.io/rewrite-target: /$1  
 nginx.ingress.kubernetes.io/use-regex: "true"  
spec:  
 ingressClassName: public  
 tls:  
 - hosts:  
 - app.chiragnagori.me #Use your domain  
 secretName: tls-secret  
 rules:  
 - host: app.chiragnagori.me #Use your domain  
 http:  
 paths:  
 - path: /flaskapp(/|$)(.\*)  
 pathType: Prefix  
 backend:  
 service:  
 name: flasksvc  
 port:  
 number: 80  
 - path: /nodeapp(/|$)(.\*)  
 pathType: Prefix  
 backend:  
 service:  
 name: nodesvc  
 port:  
 number: 80  
 - path: /(.\*)  
 pathType: Prefix  
 backend:  
 service:  
 name: flasksvc  
 port:  
 number: 80

kubectl apply -f ingress.yaml --namespace public-ingress



If you get this acem-solver ingresss the edit this ingress and add ingressClassName in spec section.

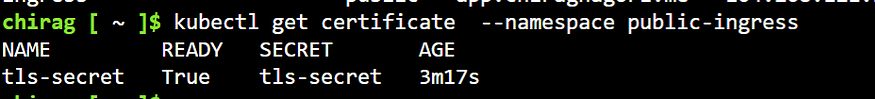




**Verify certificate**

To verify that the certificate was created successfully, use the below command

kubectl get certificate --namespace public-ingress

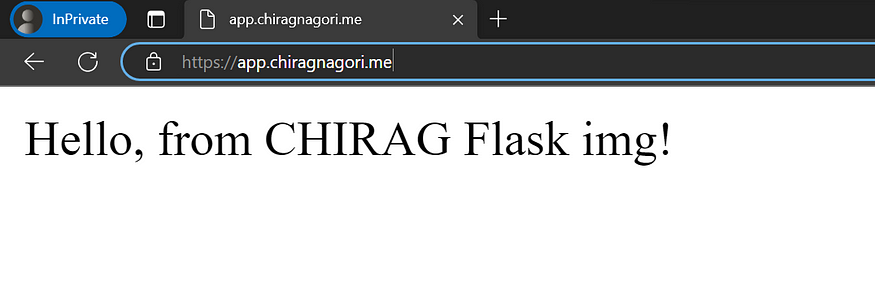


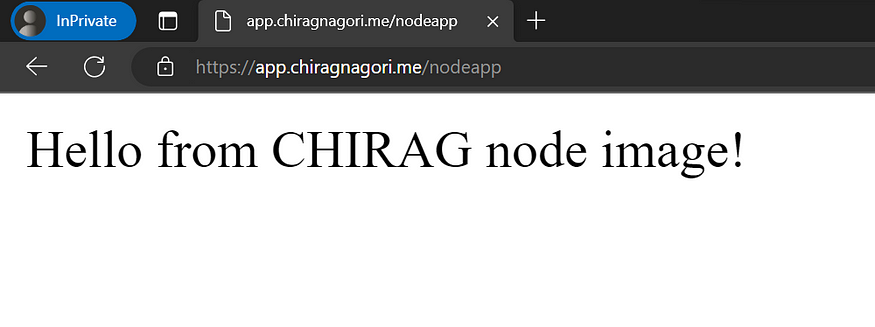
**Test the ingress configuration**

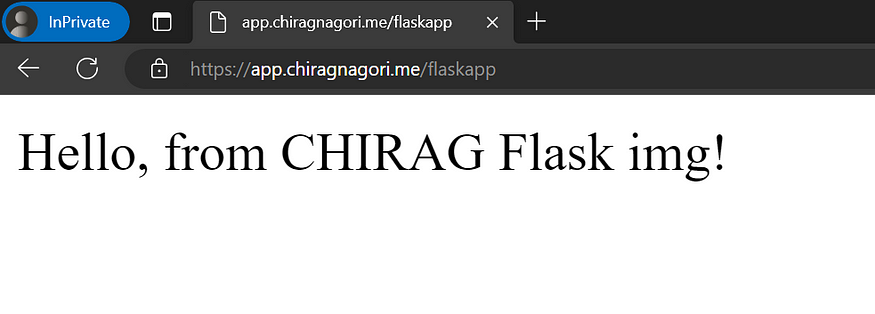
Open a web browser to the FQDN of your Kubernetes ingress controller, such as [*https://*](https://demo-aks-ingress.eastus.cloudapp.azure.com./)app.chiragnagori.me/

[*https://*](https://app.chiragnagori.me/)app.chiragnagori.me/nodeapp

[*https://*](https://app.chiragnagori.me/)app.chiragnagori.me/flaskapp







**Now the applications are secured using TLS certificate and are reachable using the custom domain.**