**A diagram of a cluster

AI-generated content may be incorrect.**

**Setup AKS + AGIC**

az aks create \

--name aks-agic \

--resource-group sumi-testing \

--network-plugin azure \

--enable-managed-identity \

--enable-addons ingress-appgw \

--appgw-name myApplicationGateway \

--appgw-subnet-cidr "10.225.0.0/16" \

--node-vm-size Standard\_B2s \

--node-count 1 \

--generate-ssh-keys \

--location eastus

**✅ What This Command Does**

* Deploys AKS with Azure CNI
* Uses Managed Identity
* Integrates Application Gateway Ingress Controller (AGIC)
* Creates a new Application Gateway named myApplicationGateway
* Uses a subnet with CIDR 10.225.0.0/16 for App Gateway (Azure will create this automatically if not exists)

🔍 Optional: Use Existing VNet and Subnet (If needed)

If you already have a VNet and subnet for App Gateway, use:

--appgw-subnet-id "/subscriptions/<sub-id>/resourceGroups/<rg>/providers/Microsoft.Network/virtualNetworks/<vnet>/subnets/<subnet>"

**Let me know if you're using a custom VNet or private AKS cluster, and I’ll tailor the command for that too.**

**🔧 Step-by-Step Next Actions**

**Step 1: Install AGIC (if not already managed by addon)**

Since you used --enable-addons ingress-appgw, AGIC is already deployed. ✅

**To verify:**

**kubectl get pods -n kube-system -l app=ingress-appgw**

You should see a pod like ingress-appgw-deployment-xxx.

**Step 2: Create a Namespace**

**kubectl create namespace ingress-nginx**

**Step 3: Deploy a Sample NGINX App**

# nginx-app.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx

namespace: ingress-nginx

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: nginx-service

namespace: ingress-nginx

spec:

selector:

app: nginx

ports:

- port: 80

targetPort: 80

**kubectl apply -f nginx-app.yaml**

**Step 4: Deploy the Ingress Resource**

# nginx-ingress.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: nginx-ingress

namespace: ingress-nginx

annotations:

kubernetes.io/ingress.class: azure/application-gateway

spec:

rules:

- http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: nginx-service

port:

number: 80

**kubectl apply -f nginx-ingress.yaml**

**Step 5: Get the Application Gateway Public IP**

**Find the App Gateway name:**

**az network application-gateway list -g MC\_sumi-testing\_aks-agic\_eastus -o table**

**Then get its public IP:**

**kubectl get ingress -n ingress-nginx**

**✅ Access your app:**

**Open the Application Gateway Public IP in your browser → You should see NGINX Welcome Page.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

[**https://kubernetes.io/docs/concepts/services-networking/ingress/**](https://kubernetes.io/docs/concepts/services-networking/ingress/)

**https://kubernetes.io/docs/concepts/services-networking/ingress/#examples**

**Ingress :**

Deploy the sample applications

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: app1-nginx-deployment

labels:

app: app1-nginx

spec:

replicas: 1

selector:

matchLabels:

app: app1-nginx

template:

metadata:

labels:

app: app1-nginx

spec:

containers:

- name: app1-nginx

image: stacksimplify/kube-nginxapp1:1.0.0

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: app1-nginx-clusterip-service

labels:

app: app1-nginx

spec:

type: ClusterIP # Change this to LoadBalancer to validate the functionality

selector:

app: app1-nginx

ports:

- port: 80

targetPort: 80

**kubectl apply -f app1\_deployment.yml**

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: app2-nginx-deployment

labels:

app: app2-nginx

spec:

replicas: 1

selector:

matchLabels:

app: app2-nginx

template:

metadata:

labels:

app: app2-nginx

spec:

containers:

- name: app2-nginx

image: stacksimplify/kube-nginxapp2:1.0.0

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: app2-nginx-clusterip-service

labels:

app: app2-nginx

annotations:

spec:

type: ClusterIP # Change this to LoadBalancer to validate the functionality

selector:

app: app2-nginx

ports:

- port: 80

targetPort: 80

**kubectl apply -f app2\_deployment.yml**

<https://github.com/stacksimplify/azure-aks-kubernetes-masterclass/blob/master/10-Ingress-Context-Path-Based-Routing/kube-manifests/04-IngressService-Manifests/01-Ingress-Context-Path-Based-Routing.yml>

kubectl get ingress

**Setup Ingress now**

kubectl create namespace ingress-basic

# Add the official stable repository

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

helm repo update

# Customizing the Chart Before Installing.

helm show values ingress-nginx/ingress-nginx

helm install ingress-nginx ingress-nginx/ingress-nginx \

--namespace ingress-basic \

--set controller.replicaCount=2 \

--set controller.nodeSelector."kubernetes\.io/os"=linux \

--set defaultBackend.nodeSelector."kubernetes\.io/os"=linux \

--set controller.service.externalTrafficPolicy=Local \

--set controller.publishService.enabled=true

**helm install ingress-nginx ingress-nginx/ingress-nginx `**

**--namespace ingress-basic `**

**--set controller.replicaCount=2 `**

**--set "controller.nodeSelector.kubernetes\.io/os=linux" `**

**--set "defaultBackend.nodeSelector.kubernetes\.io/os=linux" `**

**--set controller.service.externalTrafficPolicy=Local `**

**--set controller.publishService.enabled=true**

**kubectl get pods,svc -n ingress-basic**

**You’ll see:**

* **2 ingress-nginx-controller pods**
* **A LoadBalancer service with an EXTERNAL-IP**

# List Services with labels

kubectl get service -l app.kubernetes.io/name=ingress-nginx --namespace ingress-basic

# List Pods

kubectl get pods -n ingress-basic

kubectl get all -n ingress-basic

# Access Public IP

http://<Public-IP-created-for-Ingress>

# Output should be

404 Not Found from Nginx

# Verify Load Balancer on Azure Mgmt Console

**Primarily refer Settings -> Frontend IP Configuration**

**1.Ingress-Basic.yml**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: nginxapp1-ingress-service

#annotations:

#kubernetes.io/ingress.class: nginx

spec:

ingressClassName: nginx

rules:

- http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: app1-nginx-clusterip-service

port:

number: 80

**2.Ingress-Context-Path-Based-Routing.yml**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-cpr

#annotations:

#kubernetes.io/ingress.class: nginx

spec:

ingressClassName: nginx

rules:

- http:

paths:

- path: /app1

pathType: Prefix

backend:

service:

name: app1-nginx-clusterip-service

port:

number: 80

- path: /app2

pathType: Prefix

backend:

service:

name: app2-nginx-clusterip-service

port:

number: 80

- path: /api

pathType: Prefix

backend:

service:

name: springboot-app-service

port:

number: 80

<http://20.43.241.149/app1/index.html>

<http://20.43.241.149/app2/index.html>

**✅ Step 1: Create a DNS Zone in Azure**

A DNS Zone is used to host the DNS records for particular domains.

The domain monitorminds.net may contain several DNS record such as

**mail.monitorminds.net**

**sre.monitorminds.net**

 **Navigate to your Resource Group**

* Go to the Azure portal.
* Click on the appropriate **Resource Group** where you want to manage your DNS.

A screenshot of a chat

AI-generated content may be incorrect.

 **Create a DNS Zone**

* Within the resource group, click **Create > DNS Zone** (you can also search for DNS zone in the top search bar).

A screenshot of a computer

AI-generated content may be incorrect.

* Provide your **custom domain name** (e.g., monitorminds.net) as the DNS zone name.
* Click **Review + Create**, then click **Create** to deploy the DNS zone.

**Copy the Azure Name Servers**

* Once the DNS zone is created, open it.
* You will see a list of **Azure-assigned name servers** (e.g., ns1-01.azure-dns.com, etc.).
* **Copy all the name server entries** — you’ll need them for GoDaddy.

A screenshot of a computer

AI-generated content may be incorrect.

**✅ Step 2: Configure Your Domain on GoDaddy**

1. **Purchase a Domain**
   * Go to [GoDaddy.com](https://www.godaddy.com) and buy your desired domain (e.g., monitorminds.net).

A close-up of a message

AI-generated content may be incorrect.

1. **Update DNS Settings**
   * Log in to your GoDaddy account.
   * Go to **My Products > Domains > DNS Settings** for the purchased domain.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Change Name Servers to Azure DNS**
   * Scroll to the **Name Servers** section.
   * Click **Change** > Select **Custom**.
   * **Replace GoDaddy’s default name servers** with the ones you copied from the Azure DNS zone.
   * Click **Save**.

A screenshot of a computer

AI-generated content may be incorrect.

**Validation:**

**nslookup -type=NS monitorminds.net**

Run this from azure cloudshell

A blue screen with white text

AI-generated content may be incorrect.

This means monitorminds.net is successfully resolved with azure DNS.

**Now Let’s add the recordset on DNSZone**

A white background with black text

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Now , we can update the DNS hostname on ingress.yml**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-cpr

#annotations:

#kubernetes.io/ingress.class: nginx

spec:

ingressClassName: nginx

#defaultBackend:

# service:

# name: usermgmt-webapp-clusterip-service

# port:

# number: 80

rules:

- host: "app1.monitorminds.net"

http:

paths:

- path: /app1

pathType: Prefix

backend:

service:

name: app1-nginx-clusterip-service

port:

number: 80

- path: /app2

pathType: Prefix

backend:

service:

name: app2-nginx-clusterip-service

port:

number: 80

- path: /api

pathType: Prefix

backend:

service:

name: springboot-app-service

port:

number: 80

kubectl apply -f ingres.yml

<http://app1.monitorminds.net/app1/index.html>

<http://app1.monitorminds.net/app2/index.html>

<http://app1.monitorminds.net/api/json>

**Another example to use different domains.**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-cpr

#annotations:

#kubernetes.io/ingress.class: nginx

spec:

ingressClassName: nginx

rules:

- host: "app1.monitorminds.net"

http:

paths:

- path: /app1

pathType: Prefix

backend:

service:

name: app1-nginx-clusterip-service

port:

number: 80

- path: /app2

pathType: Prefix

backend:

service:

name: app2-nginx-clusterip-service

port:

number: 80

- host: "springboot.monitorminds.net"

http:

paths:

- path: /api

pathType: Prefix

backend:

service:

name: springboot-app-service

port:

number: 80

====Expose the Endpoint with HTTPS===

**✅ Step 1: Run the OpenSSL Command**

**🔧 For a Wildcard Domain (e.g., \*.monitorminds.net):**

openssl req -x509 -nodes -days 365 \

-newkey rsa:2048 \

-keyout tls.key \

-out tls.crt \

-subj "/CN=\*.monitorminds.net"

🔧 For a Specific Domain (e.g., app.monitorminds.net):

openssl req -x509 -nodes -days 365 \

-newkey rsa:2048 \

-keyout tls.key \

-out tls.crt \

-subj "/CN=app.monitorminds.net"

**📂 Output**

This will create two files in the current directory:

* **tls.key** – Private key
* **tls.crt** – Self-signed certificate

They are valid for **365 days** and **not password-protected** (-nodes).

**🔐 Step 2 (Optional): Create a Kubernetes TLS Secret**

kubectl create secret tls ingress-secret --cert=tls.crt --key=tls.key --namespace ingress-basic

**🌐 Step 3 (Optional): Use with NGINX Ingress**

Add TLS to your Ingress like this:

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-cpr

#annotations:

#kubernetes.io/ingress.class: nginx

spec:

ingressClassName: nginx

tls:

- hosts:

- app1.monitorminds.net

- springboot.monitorminds.net

secretName: ingress-secret

rules:

- host: "app1.monitorminds.net"

http:

paths:

- path: /app1

pathType: Prefix

backend:

service:

name: app1-nginx-clusterip-service

port:

number: 80

- path: /app2

pathType: Prefix

backend:

service:

name: app2-nginx-clusterip-service

port:

number: 80

- host: "springboot.monitorminds.net"

http:

paths:

- path: /api

pathType: Prefix

backend:

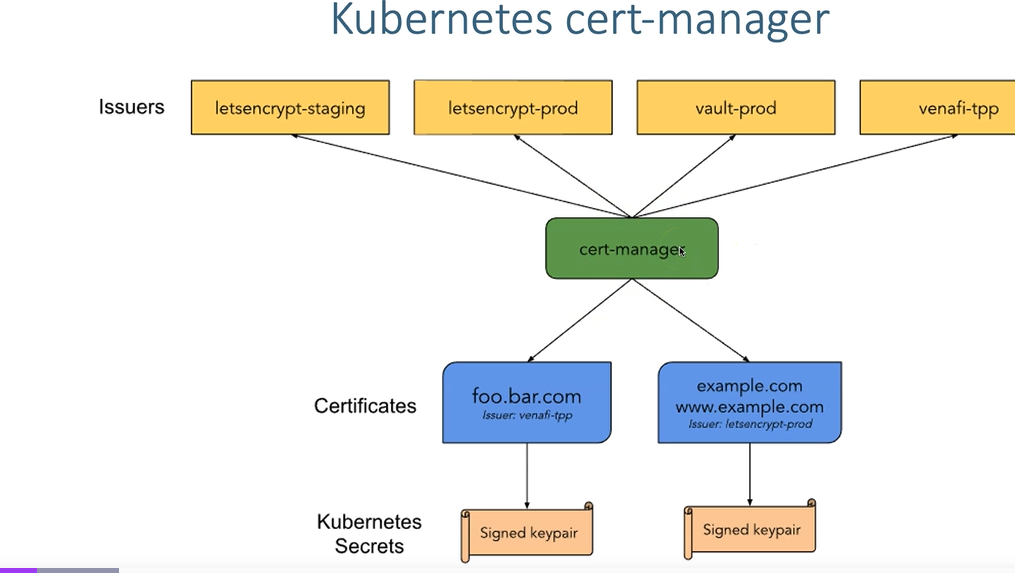
service:

name: springboot-app-service

port:

number: 80

**Expose app on HTTPS with cert-Manager and Let’s Encrypt**



<https://cert-manager.io/docs/>

# Add the Jetstack Helm repository

helm repo add jetstack https://charts.jetstack.io

# Update your local Helm chart repository cache

helm repo update

helm install cert-manager jetstack/cert-manager --namespace ingress-basic --version v1.13.3 --set installCRDs=true

# Verify Cert Manager pods

kubectl get pods --namespace ingress-basic

# Verify Cert Manager Services

kubectl get svc --namespace ingress-basic

kubectl get all --namespace ingress-basic

**Next step is to create the Cluster Issuer**

apiVersion: cert-manager.io/v1

kind: ClusterIssuer

metadata:

name: letsencrypt

spec:

acme:

# You must replace this email address with your own.

# Let's Encrypt will use this to contact you about expiring

# certificates, and issues related to your account.

email: sumanth.krishnamurthi@gmail.com

server: https://acme-v02.api.letsencrypt.org/directory

privateKeySecretRef:

# Secret resource that will be used to store the account's private key.

name: letsencrypt

# Add a single challenge solver, HTTP01 using nginx

solvers:

- http01:

ingress:

ingressClassName: nginx

**kubectl apply -f cluster-issuer.yaml**

**Update the ingress with tls configuration**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-ssl

annotations:

#kubernetes.io/ingress.class: "nginx"

cert-manager.io/cluster-issuer: letsencrypt

spec:

ingressClassName: nginx

tls:

- hosts:

- "app1.monitorminds.net"

secretName: tls-secret

rules:

- host: "app1.monitorminds.net"

http:

paths:

- path: /app1

pathType: Prefix

backend:

service:

name: app1-nginx-clusterip-service

port:

number: 80

- path: /app2

pathType: Prefix

backend:

service:

name: app2-nginx-clusterip-service

port:

number: 80

- path: /api

pathType: Prefix

backend:

service:

name: springboot-app-service

port:

number: 80

kubectl apply -f ingress-https-cert.yaml

kubectl get ingress

kubectl get certificate -A

kubectl get certificate tls-secret -o yaml

kubectl get secret

kubectl get certificate tls-secret -o yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-ssl

annotations:

#kubernetes.io/ingress.class: "nginx"

cert-manager.io/cluster-issuer: letsencrypt

spec:

ingressClassName: nginx

tls:

- hosts:

- "app1.monitorminds.net"

secretName: tls-secret

rules:

- host: "app1.monitorminds.net"

http:

paths:

- path: /app1

pathType: Prefix

backend:

service:

name: app1-nginx-clusterip-service

port:

number: 80

- path: /app2

pathType: Prefix

backend:

service:

name: app2-nginx-clusterip-service

port:

number: 80

- path: /api

pathType: Prefix

backend:

service:

name: springboot-app-service

port:

number: 80

tls:

- hosts:

- app1.monitorminds.net

secretName: tls-secret

<https://learn.microsoft.com/en-us/previous-versions/azure/aks/ingress-tls?tabs=azure-cli>