This article covers the process of configuring [E](https://github.com/kubernetes-sigs/external-dns" \t "_blank)xternalDNS with Azure Kubernetes Service (AKS) and Azure DNS. By deploying and configuring External-DNS and necessary Azure services correctly, you can ensure proper routing from your custom domain into Kubernetes. Ingress object will update the DNS zone with the correct record.

**Steps to be performed:**

* Create a DNS Zone for custom domain.
* Create a namespace ingress-basic for Ingress Controller where all ingress controller related resources will be created.
* Assign managed identity of cluster’s node pools to DNS zone.
* Deploy ExternalDNS in ingress-basic namespace using Helm.
* Install cert-manager for SSL certificates in ingress-basic namespace using Helm.
* Create a CA cluster issuer for issuing certificate.
* Create first application and service.
* Create Second application and service.
* Create an ingress route to configure the host based rules along with DNS record and TLS certificate that route traffic to one of the two applications.
* Verify the automatic created certificate.
* Test the applications using Custom Domain.

**Create a DNS Zone**

# Create a DNS Zone for custom Domain  
az network dns zone create -g aksdemocluster-rg -n mydevsecops.org

**Create an ingress controller**

# Create a namespace for ingress resources  
kubectl create namespace ingress-basic# 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 install ingress-nginx ingress-nginx/ingress-nginx \  
 --namespace ingress-basic \  
 --set controller.replicaCount=2

**Setup Permissions**

# Assign Variables  
tenantid=$(az account show --subscription "Visual Studio Enterprise Subscription" --query tenantId --output tsv)subscriptionid=$(az account show --query id -o tsv)UserClientId=$(az aks show --name [aksdemocluster](https://portal.azure.com/#@shailenderchoudharygmail.onmicrosoft.com/resource/subscriptions/461db24f-c0a7-4a65-aa6b-764717134583/resourceGroups/aksdemocluster-rg/providers/Microsoft.ContainerService/managedClusters/aksdemocluster) --resource-group [aksdemocluster-rg](https://portal.azure.com/#@shailenderchoudharygmail.onmicrosoft.com/resource/subscriptions/461db24f-c0a7-4a65-aa6b-764717134583/resourceGroups/aksdemocluster-rg) --query identityProfile.kubeletidentity.clientId -o tsv)DNSID=$(az network dns zone show --name mydevsecops.org --resource-group aksdemocluster-rg --query id -o tsv)# Assign managed identity of cluster’s node pools DNS Zone Contributor rights on to Custom Domain DNS zone.  
az role assignment create --assignee $UserClientId --role 'DNS Zone Contributor' --scope $DNSID

**Deploy ExternalDNS**

# Add the Helm repository  
helm repo add bitnami <https://charts.bitnami.com/bitnami>  
helm repo update  
  
# Use Helm to deploy an External DNS  
helm install external-dns bitnami/external-dns --namespace ingress-basic --set provider=azure --set txtOwnerId=aksdemocluster --set policy=sync --set azure.resourceGroup=aksdemocluster-rg --set azure.tenantId=$tenantid --set azure.subscriptionId=$subscriptionid --set azure.useManagedIdentityExtension=true --set azure.userAssignedIdentityID=$UserClientId

**Install cert-manager**

# Label the cert-manager namespace to disable resource validation  
kubectl label namespace ingress-basic cert-manager.io/disable-validation=true# Add the Jetstack Helm repository  
helm repo add jetstack [https://charts.jetstack.io#](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/v1.7.1/cert-manager.crds.yaml#](https://github.com/cert-manager/cert-manager/releases/download/v1.7.1/cert-manager.crds.yaml) Install the cert-manager Helm chart  
helm install cert-manager jetstack/cert-manager \  
 --namespace ingress-basic \  
 --version v1.7.1

**Create a CA cluster issuer**

apiVersion: cert-manager.io/v1  
kind: ClusterIssuer  
metadata:  
 name: letsencrypt  
spec:  
 acme:  
 server: <https://acme-v02.api.letsencrypt.org/directory>  
 email: shailender.choudhary@gmail.com  
 privateKeySecretRef:  
 name: letsencrypt  
 solvers:  
 - http01:  
 ingress:  
 class: nginx  
 podTemplate:  
 spec:  
 nodeSelector:  
 "kubernetes.io/os": linux

To create the issuer, use the kubectl command.

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

**Run demo applications**

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

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: aks-helloworld-one  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: aks-helloworld-one  
 template:  
 metadata:  
 labels:  
 app: aks-helloworld-one  
 spec:  
 containers:  
 - name: aks-helloworld-one  
 image: mcr.microsoft.com/azuredocs/aks-helloworld:v1  
 ports:  
 - containerPort: 80  
 env:  
 - name: TITLE  
 value: "Welcome to Azure Kubernetes Service (AKS)"  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: aks-helloworld-one  
spec:  
 type: ClusterIP  
 ports:  
 - port: 80  
 selector:  
 app: aks-helloworld-one

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

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: aks-helloworld-two  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: aks-helloworld-two  
 template:  
 metadata:  
 labels:  
 app: aks-helloworld-two  
 spec:  
 containers:  
 - name: aks-helloworld-two  
 image: mcr.microsoft.com/azuredocs/aks-helloworld:v1  
 ports:  
 - containerPort: 80  
 env:  
 - name: TITLE  
 value: "AKS Ingress Demo"  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: aks-helloworld-two  
spec:  
 type: ClusterIP  
 ports:  
 - port: 80  
 selector:  
 app: aks-helloworld-two

Run the two demo applications using kubectl:

kubectl apply -f aks-helloworld-one.yaml --namespace ingress-basic  
kubectl apply -f aks-helloworld-two.yaml --namespace ingress-basic

**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: hello-world-ingress  
 annotations:  
 kubernetes.io/ingress.class: nginx  
 cert-manager.io/cluster-issuer: letsencrypt  
spec:  
 tls:  
 - hosts:  
 - web1.mydevsecops.org  
 - web2.mydevsecops.org  
 secretName: tls-secret  
 rules:  
 - host: web1.mydevsecops.org  
 http:  
 paths:  
 - path: /  
 pathType: Prefix  
 backend:  
 service:  
 name: aks-helloworld-one  
 port:  
 number: 80- host: web2.mydevsecops.org  
 http:  
 paths:  
 - path: /  
 pathType: Prefix  
 backend:  
 service:  
 name: aks-helloworld-two  
 port:  
 number: 80

Create the ingress resource using the kubectl:

kubectl apply -f hello-world-ingress.yaml --namespace ingress-basic

**Verify certificate**

To verify that the certificate was created successfully, use the kubectl describe certificate tls-secret --namespace ingress-basic command.

**Note**: Wait for few minutes for Txt and A records update in DNS Zone.

Open web1.mydevsecops.org and web2.mydevsecops.org in browser to check 2 different webpage which are being routed from App1 and App2.

For more Detailed Video on Azure Kubernetes Service(AKS) check my YouTube channel:

<https://www.youtube.com/channel/UCkJRkUw2hYSlleTUuHY43lQ>