LIKE & SHARE

SUBSCRIBE

SSL ON AKS

APPLICATION GATEWAY & LET'S ENCRYPT IN ACTION

- **To set up SSL on AKS using Application Gateway and Let's Encrypt, you need the following prerequisites:**

1. **Azure Prerequisites**
   - An **Azure subscription** with permissions to create resources.
   - An **Azure Kubernetes Service (AKS) cluster** deployed.
   - An **Application Gateway v2 SKU** (required for Ingress Controller).
   - A Public DNS domain (e.g., thejourneyofdevops.com), managed in GoDaddy (or any other DNS provider).
2. **Tools and CLI Setup**
   - **Azure CLI (az): Ensure you have the latest version installed.**
   - **Kubectl**: To interact with the AKS cluster.
   - **Helm**: Required for installing the Application Gateway Ingress Controller (AGIC)**.**
   - **Cert-Manager**: For managing Let's Encrypt SSL certificates.
3. **AKS and AGIC Configuration**
   - Enable **Managed Identity** for AGIC.
   - Deploy AGIC using Helm.

4. **Let's Encrypt & Cert-Manager**
   - Install **Cert-Manager** on AKS.
   - Configure a **ClusterIssuer** for Let's Encrypt.
   - Validate **DNS or HTTP challenge** for certificate issuance.
   - Apply the **Certificate resource** for SSL automation.
5. **DNS Configuration (GoDaddy)**
   - If using **DNS-01 challenge**, configure TXT records in GoDaddy.
   - If using **HTTP-01 challenge**, ensure proper Ingress routing.
6. **Testing & Validation**
   - Ensure SSL is correctly applied to your application.
   - Check **Application Gateway** rules and backend health.
   - **Verify HTTPS traffic using a web browser or curl.**

Let’s Encrypt & Cert-Manager: Detailed Explanation

- Let’s Encrypt provides **free, automated SSL certificates**, and **Cert-Manager** is a Kubernetes-native tool that automates their issuance and renewal.

1. **Install Cert-Manager on AKS:**

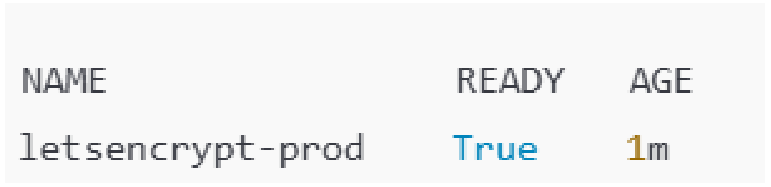
- Cert-Manager is deployed as a Kubernetes controller that automatically provisions TLS certificates from Let’s Encrypt and manages their lifecycle.
  - **Add the Helm Repository** - First, add the Jetstack Helm repository, which maintains Cert-Manager:
    - helm repo add jetstack https://charts.jetstack.io
    - helm repo update
  - **Install Cert-Manager -** Deploy Cert-Manager into your AKS cluster using Helm:
    - helm install cert-manager jetstack/cert-manager --namespace cert-manager --create-namespace --set installCRDs=true
  - **Verify Installation -** Run the following command to check if Cert-Manager pods are running: kubectl get pods -n cert-manager

2. **Configure a ClusterIssuer for Let’s Encrypt** - A **ClusterIssuer** is a Kubernetes resource that defines how certificates should be requested from an external CA (in this case, Let’s Encrypt).
   - **Create a Let's Encrypt ClusterIssuer** –

```yaml
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: your-email@example.com # Replace with your email
    privateKeySecretRef:
      name: letsencrypt-prod
    solvers:
    - http01:
        ingress:
          class: azure/application-gateway
```

server: Specifies Let’s Encrypt production environment.
email: Used for renewal notifications.
privateKeySecretRef: Stores account private key.
http01 solver: Uses HTTP-01 challenge via Azure Application Gateway.

```
kubectl apply -f letsencrypt-clusterissuer.yaml
kubectl get clusterissuer
```



```
NAME               READY   AGE
letsencrypt-prod   True    1m
```

What is an ACME Challenge?

- ACME (**Automated Certificate Management Environment**) is a protocol used by **Let's Encrypt** to automate SSL/TLS certificate issuance. When you request a certificate, Let's Encrypt needs to verify that you **own or control** the domain. This is done using an **ACME challenge.**

  HTTP-01 Challenge (Most Common)

    - Let's Encrypt asks you to place a special file at http://yourdomain.com/.well-known/acme-challenge/
    - It verifies the file via HTTP before issuing the certificate.
    - Works if your website is accessible over HTTP.

  DNS-01 Challenge (Used for Wildcard Domains)

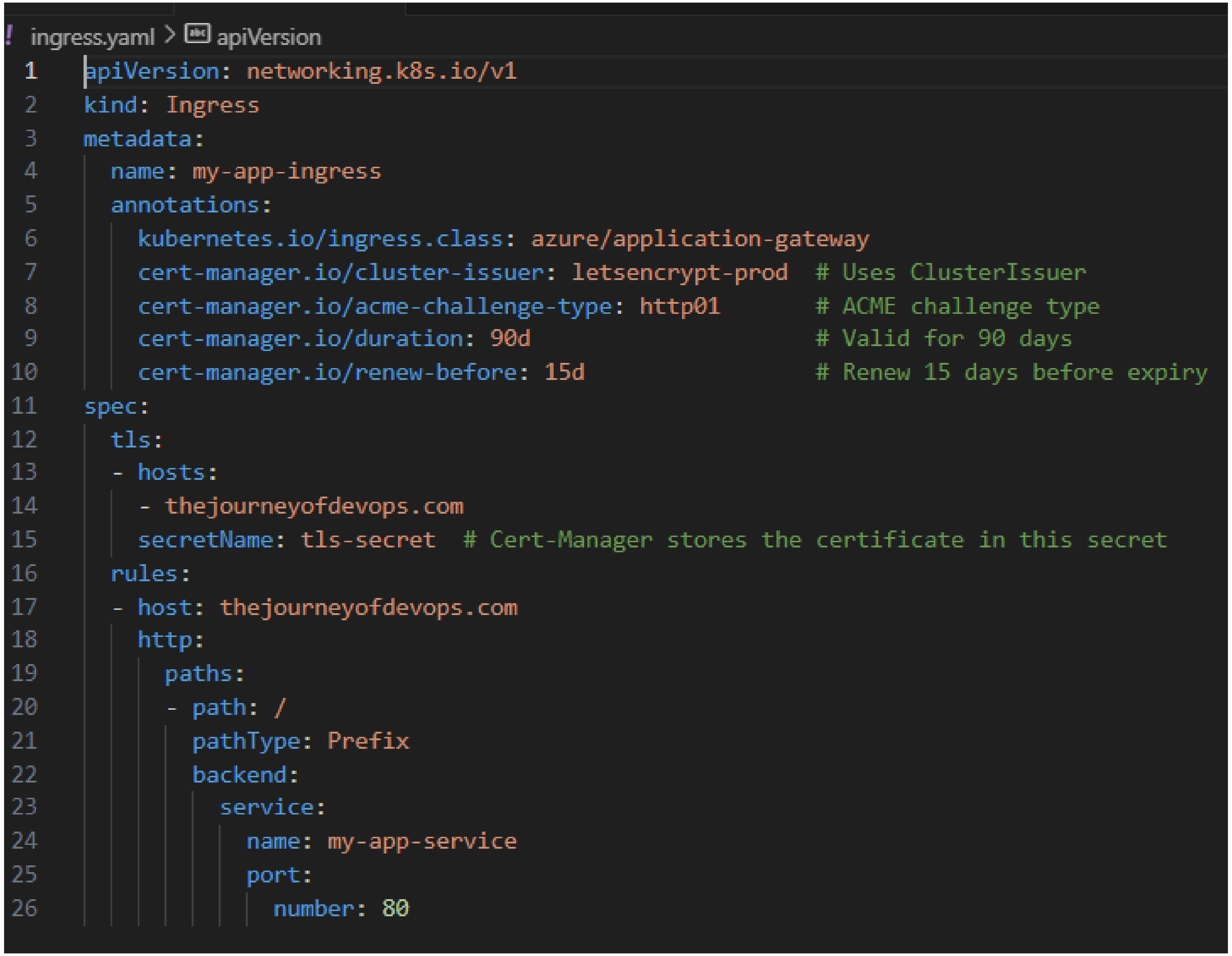
    - Requires adding a special DNS TXT record (_acme-challenge.yourdomain.com).
    - Used for **wildcard certificates** (*.yourdomain.com).
    - Works even if your website isn't online yet.

    Cert-Manager automatically handles these challenges based on your Ingress annotations.

    - If using `http-01`, it creates temporary pods to respond to the challenge.
    - If using `dns-01`, it updates your DNS provider with the required TXT record.

What is ingress-shim in Cert-Manager?

- ingress-shim is a built-in controller in Cert-Manager that automatically creates a Certificate resource when it detects TLS annotations in an Ingress resource.
- **Why is ingress-shim Useful?**
  - Normally, when using Cert-Manager with Let's Encrypt, you need to manually create a Certificate resource to request an SSL certificate.
  - **With ingress-shim**, you can skip that manual step! Instead, you just add annotations in your Ingress, and Cert-Manager will:
  - Automatically generate a Certificate resource.
  - Handle ACME challenges (HTTP-01 or DNS-01) to get the SSL certificate.
  - Store the certificate in a Kubernetes Secret.
  - Renew the certificate before it expires.
- **Even if ingress-shim is enabled, it's best to manually create the Certificate resource to have more control over renewal settings.**
- ACME (**Automated Certificate Management Environment**) is a protocol used by **Let's Encrypt** to automate SSL/TLS certificate issuance. When you request a certificate, Let's Encrypt needs to verify that you **own or control** the domain. This is done using an **ACME challenge.**



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-app-ingress
  annotations:
    kubernetes.io/ingress.class: azure/application-gateway
    cert-manager.io/cluster-issuer: letsencrypt-prod  # Uses ClusterIssuer
    cert-manager.io/acme-challenge-type: http01       # ACME challenge type
    cert-manager.io/duration: 90d                     # Valid for 90 days
    cert-manager.io/renew-before: 15d                 # Renew 15 days before expiry
spec:
  tls:
  - hosts:
    - thejourneyofdevops.com
    secretName: tls-secret  # Cert-Manager stores the certificate in this secret
  rules:
  - host: thejourneyofdevops.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: my-app-service
            port:
              number: 80
```

DEMO

Debug

- •kubectl get certificates -A
- •kubectl describe certificate -n <namespace> <certificate-name>
- •kubectl get challenges -A
- •kubectl describe challenge -n <namespace> <challenge-name>
- •kubectl logs -n cert-manager deploy/cert-manager
- •kubectl get deploy -n cert-manager cert-manager -o yaml | Select-String "ingress-shim“ (If Ingress Shim is enabled, you should see something like: - --controllers=*,ingress-shim)
- •Enable Ingress Shim in Cert-Manager - kubectl edit deploy -n cert-manager cert-manager
- •Find the args section in the cert-manager-controller container and add the following argument:
-   - --controllers=*,ingress-shim