Automating SSL in GKE using Cert-Manager

**We will first deploy a basic nginx app and configure an ingress to serve it over http then we will install cert-manager and use it to generate/automate ssl.

1. Deploy nginx and serve over HTTP

create a demo deployment

```
kubectl create deploy demo --image=nginx
```

expose demo deployment

```
kubectl expose deployment demo --name=demo-svc --port=80
```

create a global static ip

```
gcloud compute addresses create demo-ip --global
```

Or we can create the global static ip using UI

create an ingress

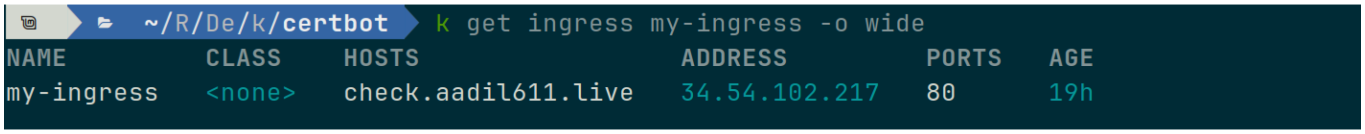
my-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    kubernetes.io/ingress.class: gce
    kubernetes.io/ingress.allow-http: "true"
    # name of our global ip to be used by this ingress
    kubernetes.io/ingress.global-static-ip-name: demo-ip
spec:
  rules:
    - host: check.aadil611.live
      http:
        paths:
          - path: /
            pathType: Prefix
```

```yaml
              backend:
                service:
                  name: demo-svc
                  port:
                    number: 80
```

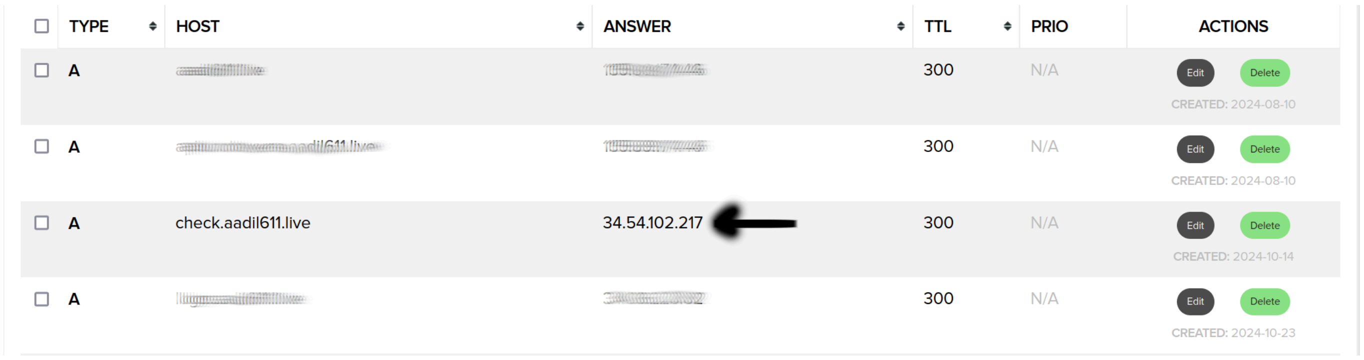
```
k apply -f my-ingress.yaml
```

get ip of our ingress



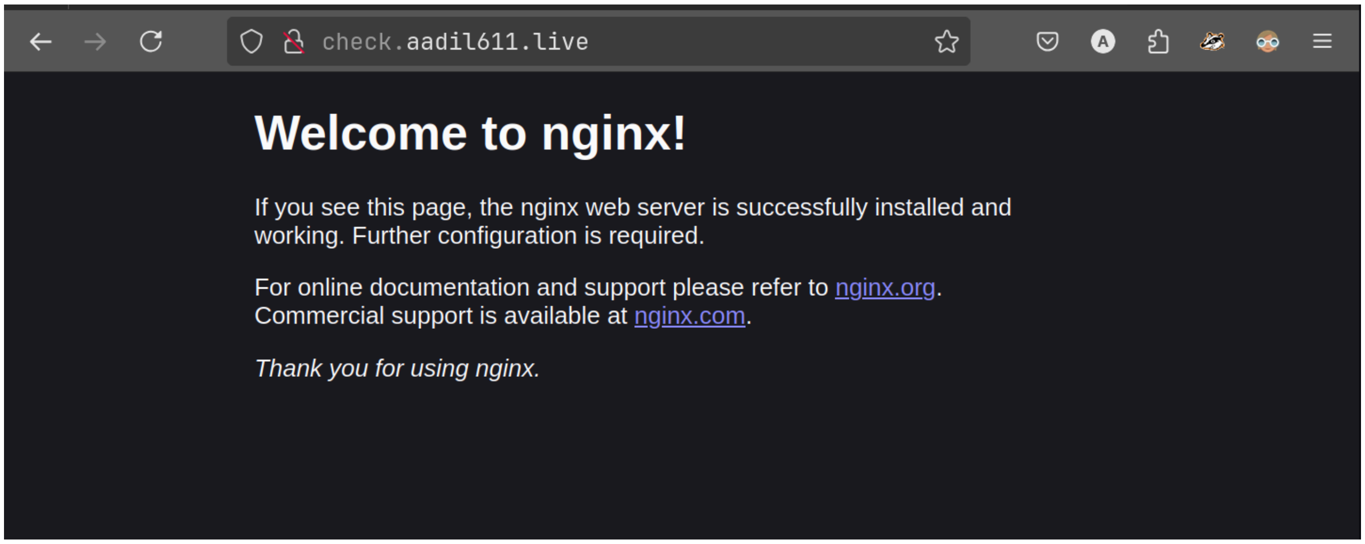
```
~/R/De/k/certbot  k get ingress my-ingress -o wide
NAME         CLASS    HOSTS               ADDRESS         PORTS   AGE
my-ingress   <none>   check.aadil611.live 34.54.102.217   80      19h
```

create a DNS record and map it to ingress ip



☐	TYPE	HOST	ANSWER	TTL	PRIO	ACTIONS
☐	A	[redacted]	[redacted]	300	N/A	Edit Delete CREATED: 2024-08-10
☐	A	[redacted]	[redacted]	300	N/A	Edit Delete CREATED: 2024-08-10
☐	A	check.aadil611.live	34.54.102.217	300	N/A	Edit Delete CREATED: 2024-10-14
☐	A	[redacted]	[redacted]	300	N/A	Edit Delete CREATED: 2024-10-23

check if traffic is being served over our domain



check.aadil611.live

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

even if we try to put https:// before url, it will show us warning and ask to proceed with risk

2. Automating SSL using Cert-Manager

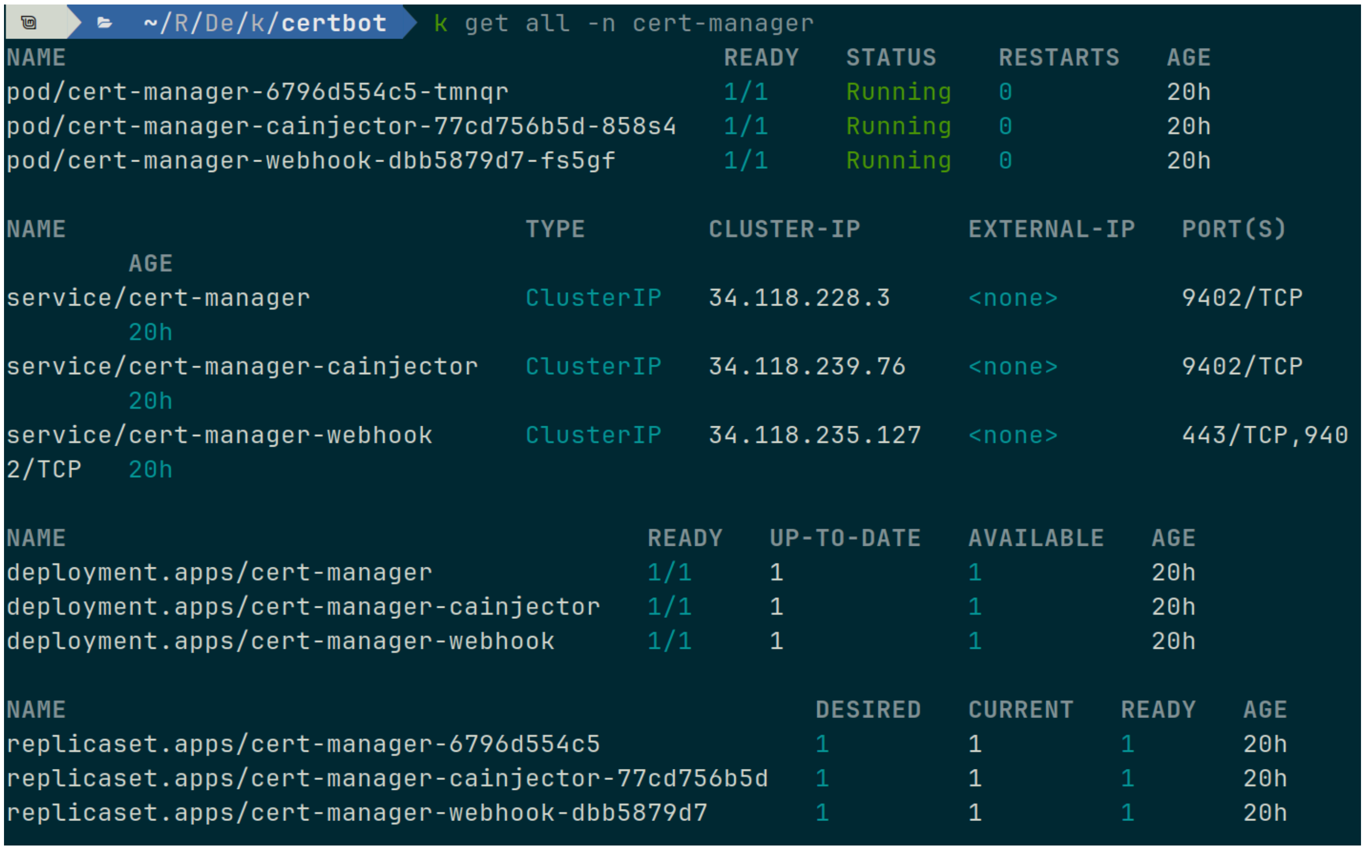
install cert-manager

```
kubectl apply -f https://github.com/cert-manager/cert-
manager/releases/download/v1.16.1/cert-manager.yaml
```

> Note: avoid using gke auto-pilot cluster. i faced multiple challenges with auto-pilot cluster while installing,working with cert-manager and could't resolve it so i moved to standard cluster.

confirm the installation

```
kubectl get all -n cert-manager
```



```
 ~/R/De/k/certbot  k get all -n cert-manager
NAME                                          READY   STATUS    RESTARTS   AGE
pod/cert-manager-6796d554c5-tmnqr             1/1     Running   0          20h
pod/cert-manager-cainjector-77cd756b5d-858s4  1/1     Running   0          20h
pod/cert-manager-webhook-dbb5879d7-fs5gf      1/1     Running   0          20h

NAME                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
        AGE
service/cert-manager              ClusterIP   34.118.228.3     <none>        9402/TCP
        20h
service/cert-manager-cainjector   ClusterIP   34.118.239.76    <none>        9402/TCP
        20h
service/cert-manager-webhook      ClusterIP   34.118.235.127   <none>        443/TCP,940
2/TCP   20h

NAME                                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/cert-manager              1/1     1            1           20h
deployment.apps/cert-manager-cainjector   1/1     1            1           20h
deployment.apps/cert-manager-webhook      1/1     1            1           20h

NAME                                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/cert-manager-6796d554c5              1         1         1       20h
replicaset.apps/cert-manager-cainjector-77cd756b5d   1         1         1       20h
replicaset.apps/cert-manager-webhook-dbb5879d7       1         1         1       20h
```

Create an Issuer

issuer.yaml

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: letsencrypt-demo
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: work.aadil611@gmail.com
    privateKeySecretRef:
      name: letsencrypt-demo
    solvers:
      - http01:
          ingress:
            name: my-ingress
```

> this issuer will make a call to server url and generate the ssl (by verifying the challenge over http using our ingress) and store it into a secret named letsencrypt-demo (we don't have to create this secret explicitly).

```
kubectl apply -f issuer.yaml
```

create a secret for storing ssl certificate for specific ingress rules

demo-ssl-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: demo-ssl-secret
type: kubernetes.io/tls
stringData:
  tls.key: ""
  tls.crt: ""
```

```
kubectl apply -f demo-ssl-secret.yaml
```

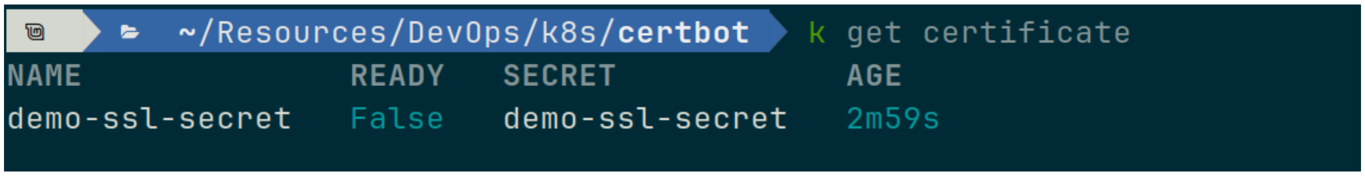
Update Ingress to generate ssl

```
kubectl edit ingress my-ingress
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    kubernetes.io/ingress.class: gce
    kubernetes.io/ingress.allow-http: "true"
    kubernetes.io/ingress.global-static-ip-name: demo-ip
    cert-manager.io/issuer: letsencrypt-demo # added line
spec:
```

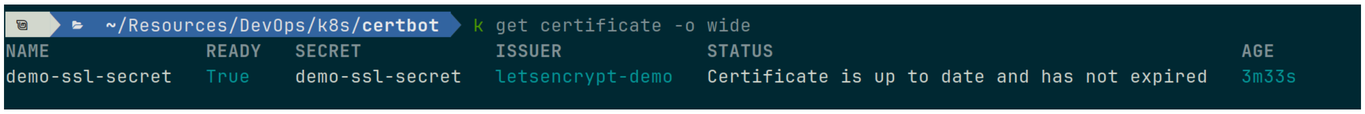
```yaml
  rules:
    - host: check.aadil611.live
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: demo-svc
                port:
                  number: 80

# added
  tls:
    - hosts:
        - check.aadil611.live
      secretName: demo-ssl-secret
```

check status of certificates



```
~/Resources/DevOps/k8s/certbot  k get certificate
NAME              READY   SECRET            AGE
demo-ssl-secret   False   demo-ssl-secret   2m59s
```

its not ready yet. wait for few more minites and check again

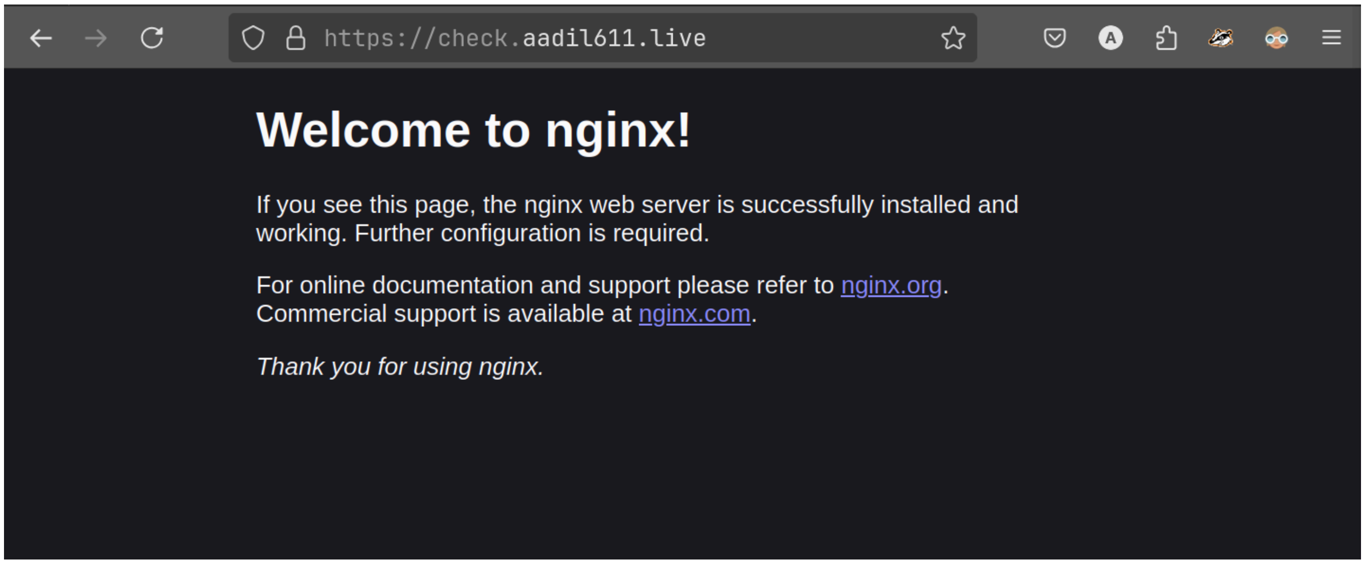


```
~/Resources/DevOps/k8s/certbot  k get certificate -o wide
NAME              READY   SECRET            ISSUER              STATUS                                          AGE
demo-ssl-secret   True    demo-ssl-secret   letsencrypt-demo    Certificate is up to date and has not expired   3m33s
```

we can see that certificate is generated and it's showing ready as *True*

it may take between 5 - 10 minutes

access the url again


https://check.aadil611.live

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

🎉 Hurray! SSL is now live on *check.aadil611.live*

> Note:

- Use Namespaces to separate certificate management for different environments (dev, staging, prod) and prevent conflicts.
- Test with ACME Staging (*server url*: https://acme-staging-v02.api.letsencrypt.org/directory) before production to avoid rate limits; switch to production when stable.
- Monitor Renewals and set alerts for failed renewals, catching issues early.
- Use DNS-01 Challenges for internal services to validate ownership without public exposure.

if we use staging url just for testing, we can create prod issuer and overwrite ingress to use this production issuer

```
kubectl annotate ingress my-ingress cert-manager.io/issuer=letsencrypt-production --overwrite
```