1. Deploy Nginx Ingress Controller:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.1/deploy/static/provider/aws/deploy.yaml

1. Deploy cert-manager using following command:

kubectl apply --validate=false -f <https://github.com/jetstack/cert-manager/releases/download/v0.12.0/cert-manager.yaml>

Use below:

kubectl apply -f <https://github.com/jetstack/cert-manager/releases/download/v1.6.1/cert-manager.yaml>

1. Create IAM Policy with name LetsEncrypt-Policy using below json:

BETTER TO GIVE [AmazonRoute53FullAccess](https://console.aws.amazon.com/iam/home#/policies/arn%3Aaws%3Aiam%3A%3Aaws%3Apolicy%2FAmazonRoute53FullAccess) than the below policy.

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"route53:GetChange",

"route53:ChangeResourceRecordSets",

"route53:ListResourceRecordSets"

],

"Resource": [

"arn:aws:route53:::hostedzone/\*",

"arn:aws:route53:::change/\*"

]

},

{

"Sid": "VisualEditor1",

"Effect": "Allow",

"Action": "route53:ListHostedZonesByName",

"Resource": "\*"

}

]

}

1. Create a AWS user called letsencrypt with access and secret key and add above policy.
2. Create a kubectl secret for the secret key using the following command:

kubectl create secret generic acme-route53 --from-literal=secret-access-key=yYwscSJP1SVEr5mn4RICbaxYUmpfGl0ew+8pcn6e -n cert-manager

1. Create a ClusterIssuer using following code:

apiVersion: cert-manager.io/v1

kind: ClusterIssuer

metadata:

name: letsencrypt-prod

namespace: cert-manager

spec:

acme:

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

email: mavrick202@gmail.com

privateKeySecretRef:

name: letsencrypt-prod

solvers:

- selector:

dnsZones:

- "trainingk8s.xyz"

- "telugugcp.xyz"

dns01:

route53:

region: us-east-1

accessKeyID: AKIAQEMIJBYFV3LCQP64

secretAccessKeySecretRef:

name: acme-route53

key: secret-access-key

1. Request for a certificate using the following code.

apiVersion: cert-manager.io/v1

kind: Certificate

metadata:

name: nginx-tls-awscoffee-cert

namespace: default

spec:

secretName: nginx-tls-default

issuerRef:

name: letsencrypt-prod

kind: ClusterIssuer

dnsNames:

- '\*.trainingk8s.xyz'

- '\*.telugugcp.xyz'

acme:

config:

- dns01:

provider: route53

domains:

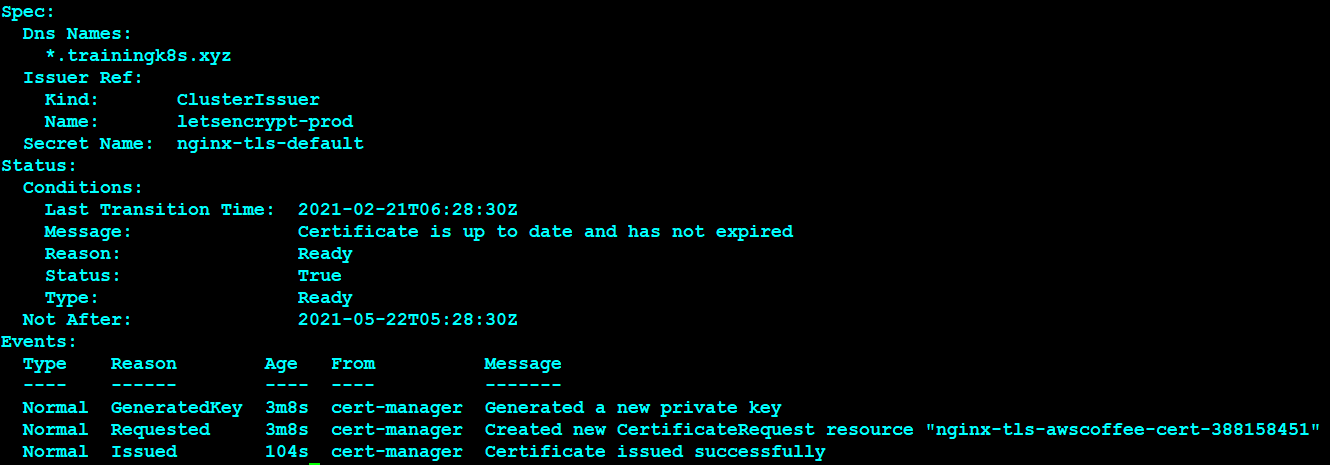
- '\*.trainingk8s.xyz'

- '\*.telugugcp.xyz'

ku apply -f cert.yaml --validate=false

1. Check the certificate status

ku describe certificate nginx-tls-awscoffee-cert



1. Create a Deployment and service using following code

ku run deploy01 --image=sreeharshav/testcontainer:v1

ku expose deployment deploy01 --port=8000 --target-port=80

1. Create ingress as per below. Make sure you change the secret name as you have created.

ku get secrets

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: java-ingress

annotations:

ingress.kubernetes.io/rewrite-target: /

spec:

tls:

- hosts:

- "java.telugugcp.xyz"

secretName: nginx-tls-default

rules:

- host: java.telugugcp.xyz

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: javadeploy

port:

number: 8001

1. Follow steps as mentioned in the article below:

<https://medium.com/cloud-prodigy/configure-letsencrypt-and-cert-manager-with-kubernetes-3156981960d9>

For renew time:

issuerRef:

kind: ClusterIssuer

name: letsencrypt-issuer

renewBefore: 2159h50m0s

**LetsEncrypt Troubleshooting:**

ku describe cert nginx-tls-awscoffee-cert

ku describe certificaterequests.cert-manager.io nginx-tls-awscoffee-cert-388158451

kubectl describe order nginx-tls-awscoffee-cert-388158451-2127603027

kubectl describe challenge nginx-tls-awscoffee-cert-388158451-2127603027-3471110041

About Cert-Manager:

<https://cert-manager.io/docs/#:~:text=cert%2Dmanager%20is%20a%20native,key%20pair%2C%20or%20self%20signed>.

Using Self-Signed Certs with MKCERT

<https://github.com/FiloSottile/mkcert/releases/tag/v1.4.3>

mv mkcert-v1.4.3-linux-amd64 mkcert

chmod 700 mkcert

./mkcert -install

Created a new local CA 💥

The local CA is now installed in the system trust store! ⚡️

./mkcert \*.trainingk8s.xyz

mv \_wildcard.trainingk8s.xyz-key.pem tls2.key

mv \_wildcard.trainingk8s.xyz.pem tls2.crt

kubectl create secret tls nginx-tls-default2 --cert=tls2.crt --key=tls2.key

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: www-ingress

annotations:

ingress.kubernetes.io/rewrite-target: /

spec:

tls:

- hosts:

- www.trainingk8s.xyz

secretName: nginx-tls-default2

rules:

- host: www.trainingk8s.xyz

http:

paths:

- path: /

backend:

serviceName: www-deployment

servicePort: 80

---

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: myapp1-ingress

annotations:

ingress.kubernetes.io/rewrite-target: /

spec:

tls:

- hosts:

- myapp1.trainingk8s.xyz

secretName: nginx-tls-default2

rules:

- host: myapp1.trainingk8s.xyz

http:

paths:

- path: /

backend:

serviceName: myapp1-deployment

servicePort: 80

---

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: myapp2-ingress

annotations:

ingress.kubernetes.io/rewrite-target: /

spec:

tls:

- hosts:

- myapp2.trainingk8s.xyz

secretName: nginx-tls-default2

rules:

- host: myapp2.trainingk8s.xyz

http:

paths:

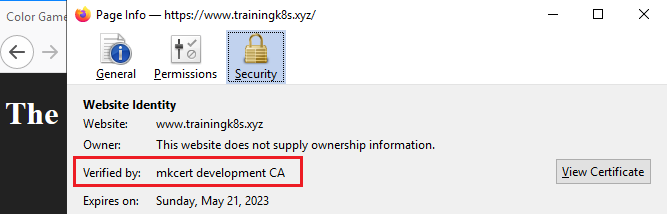
- path: /

backend:

serviceName: myapp2-deployment

servicePort: 80





**TRAEFIK:**

<https://tferdinand.net/en/traefik-2-reverse-proxy-in-kubernetes/>

Moreover, I create a local directory in which I will store my certificates, because Let's Encrypt limits the number of weekly requests for the same certificate