Securing AWS EKS Kubernetes Ingress gateway resource with Trusted TLS certificate using Cert-manager-Atlas Issuer

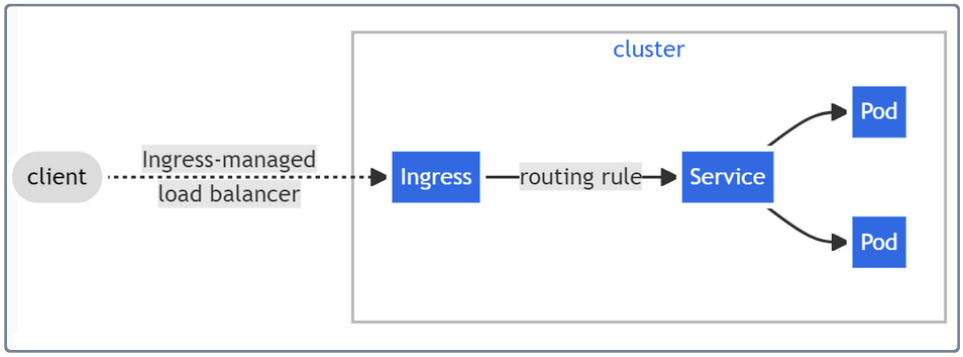
Pre-requisites:

1. AWS Account
2. Nginx Ingress
3. One Valid Domain Name
4. Kops and Kubectl
5. Helm Package Manager
6. Cert-manager & its CRD's
7. Cert-manager-Atlas Plugin

What is Ingress?

Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.

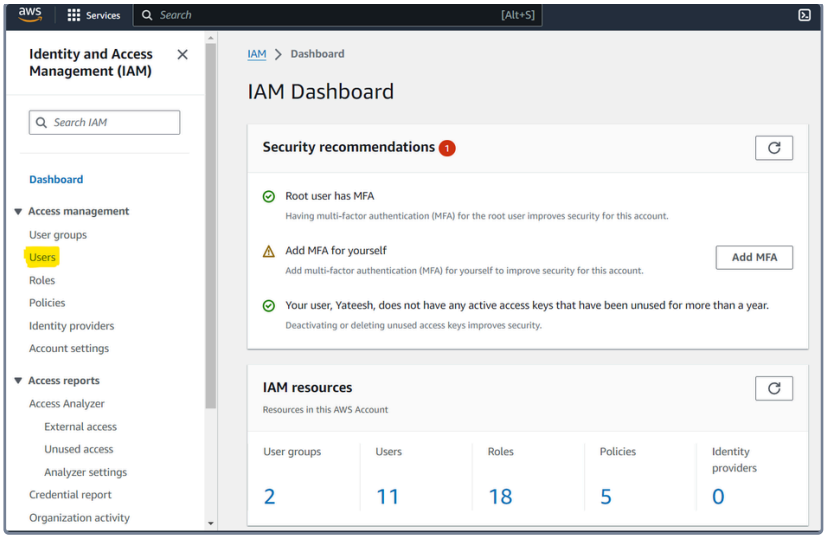
Here is a simple example where an Ingress sends all its traffic to one Service:



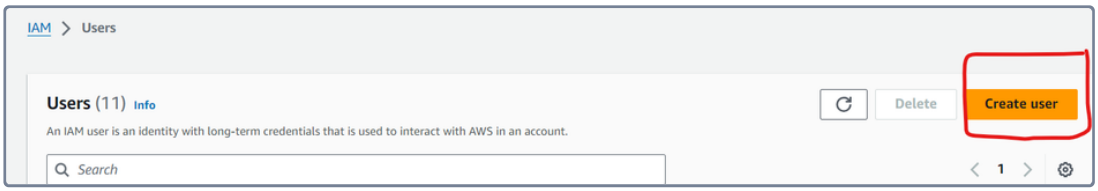
All the below mentioned steps has been executed over Linux (amd64)

Here are the following steps for your to secure your Ingress with GlobalSign's Trusted TLS certificate using Cert-manager-Atlas Plugin

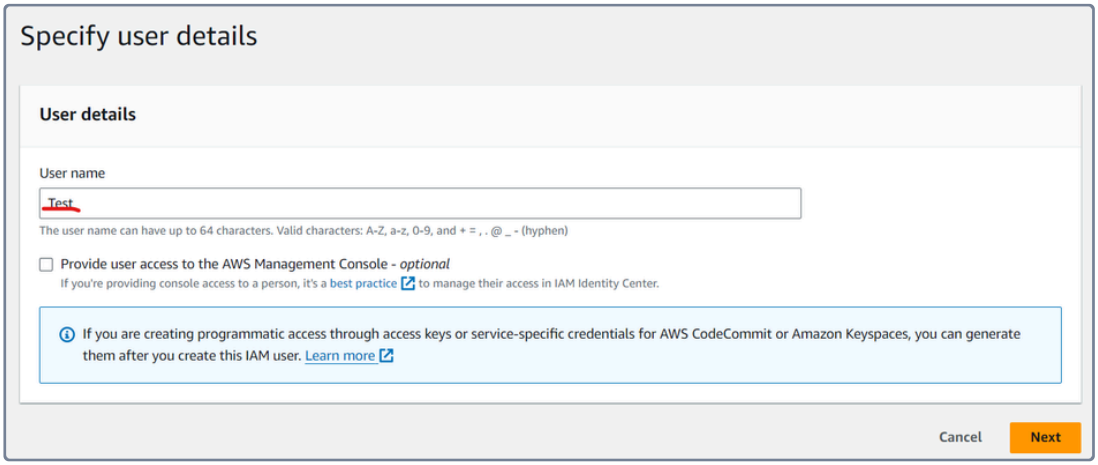
1. **AWS Ubuntu EC2 Instance - Use the AWS documentation for creating Ubuntu Instance**
2. **Create A User in IAM**
   a. Go to IAM Console and select Users



b. Click on create user



c. Give a name to the user

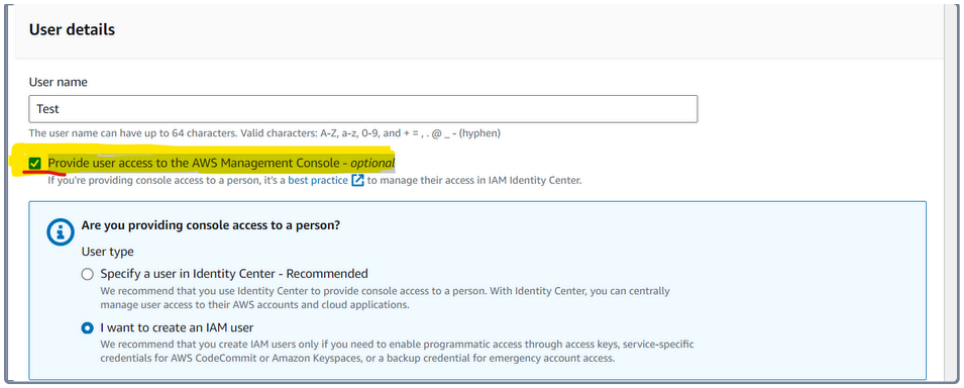


d. Fill the checkbox of "Provide user Access to the AWS Management Console"

i. Select "I want to create an IAM user "

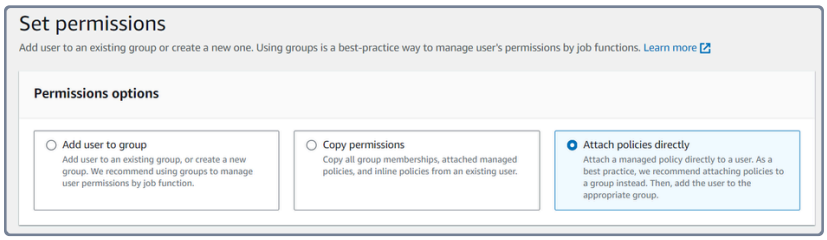
ii. In Console Password, choose "Autogenerate Password" or "Custom Password" based on your choice.

iii. Click "Next" in the bottom right corner.



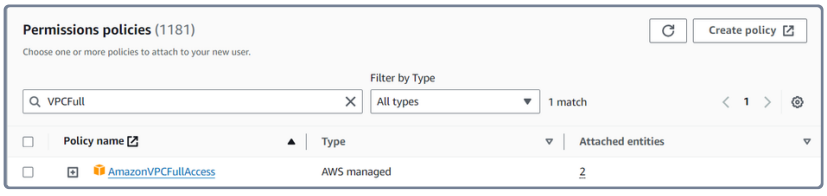
e. In Set Permissions, choose

i. Select "Add User to a group" in case if you already have defined policies for a particular user group, otherwise choose "Attach Policies directly".

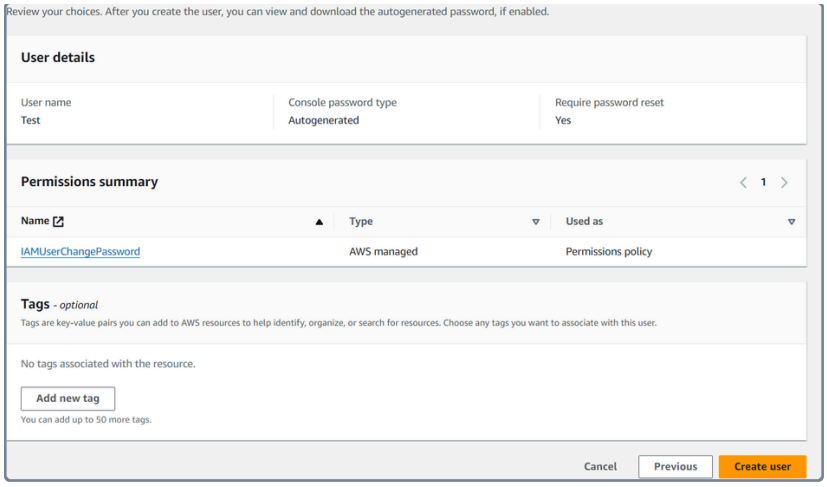


ii. In Permission Policies, provide the following permissions to the user(Note:- You can provide permissions based on your own requirements as this is just for the example purposes.)

1. AdministratorAccess
2. AmazonEC2FullAccess
3. AmazonEKSClusterPolicy
4. AmazonEKSServicePolicy
5. AmazonEventBridgeFullAccess
6. AmazonRoute53FullAccess
7. AmazonVPCFullAccess
8. AWSCloudFormationFullAccess
9. IAMFullAccess
10. Click "Next" in the bottom-right corner

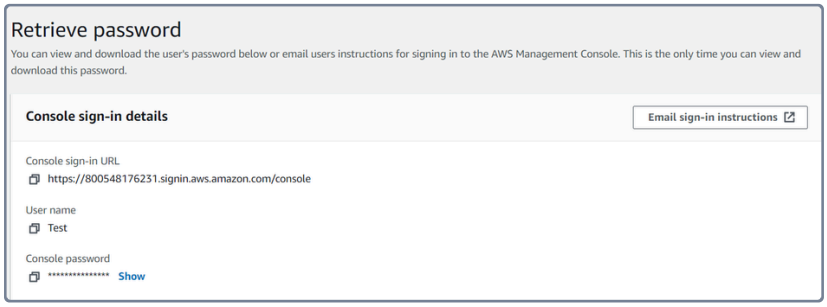


iii. Review your User Permissions and Policies



iv. Select "Create User" in the bottom-right corner and your user will be created.

v. Retrieve Login URL and Password

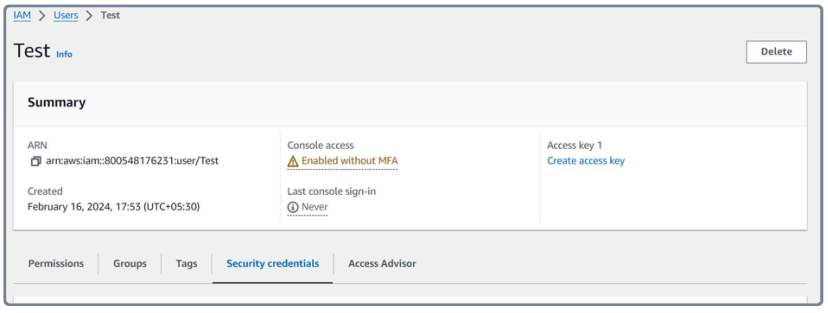


3. **Provide programmatic Access to your user**

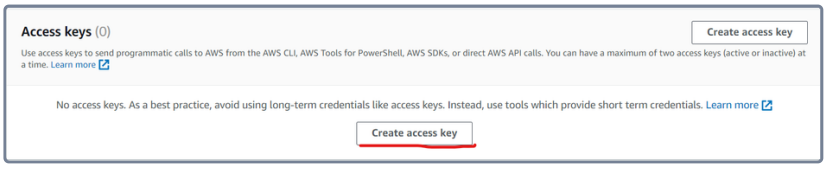
a. Go to IAM and then users again

b. Select your created user

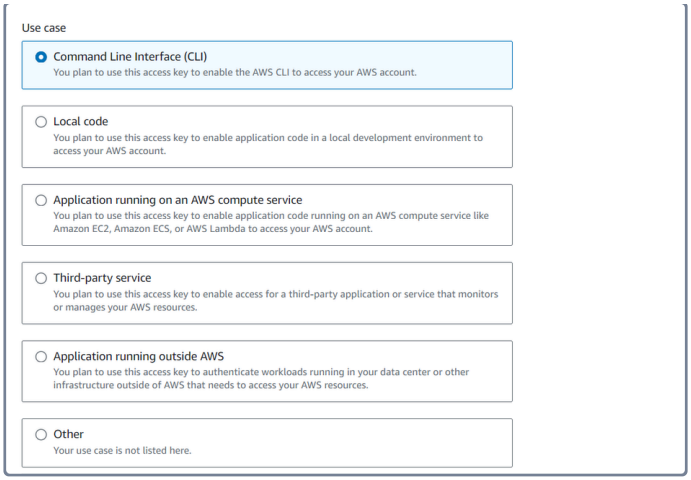
c. Select "Security Credentials"



d. Go to "Access Keys" in your Security Credentials and Choose "Create Access Key"

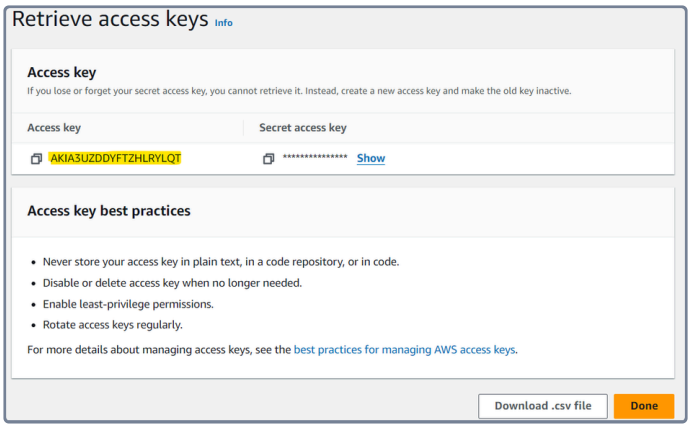


e. Go to the Use Case and select "AWS CLI"



f. Click Next and then "Create Access Key"

g. You will get your Programmatic Access Keys here



4. Connect to your AWS ec2 instance which you have created in the Step-1 (Ref.)
5. Once you are logged in to your instance, then Install the following tools
   a. **Install Unzip**

```
$sudo apt install
```

   b. **Configure AWS CLI**

```
$curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
$unzip awscliv2.zip
$sudo ./aws/install
```

6. Now, Configure AWS CLI with the following commands and the programmatic access keys created in Step -3(g)

```
$aws configure
#enter the Access key ID and Secret access key.
#Provide the region details i.e., us-east-1 or any other
#Give output format as "json".
```

```
#Generate public and private keys
$ssh-keygen
```

7. **Install Helm**

```
$curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
$chmod 700 get_helm.sh
$./get_helm.sh
```

8. **Install kubectl and eksctl(tools to manage and interact with the kubernetes cluster)**

   a. Installing latest version of kubectl

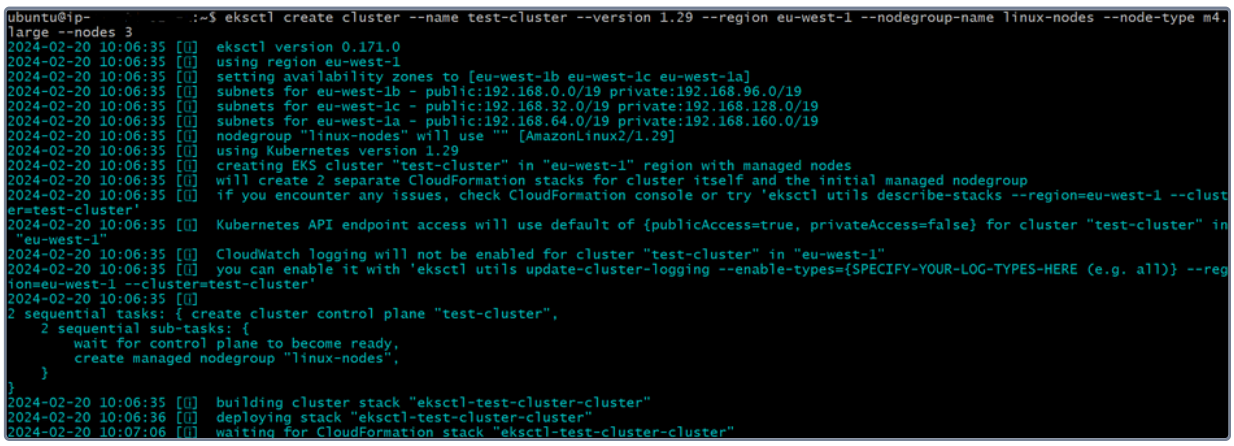
```
$curl -LO"https://dl.k8s.io/release/$(curl -L -shttps://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubec
#make the downloaded file executable
$chmod +x kubectl
#Move the executable to the /usr/local/bin
$sudo mv kubectl /usr/local/bin
```

   b. Installing the latest version of eksctl:

```
#for ARM systems, set ARCH to: arm64, armv6 or armv7
$ARCH=amd64
$PLATFORM=$(uname -s)_$ARCH
$curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_$PLATFORM.tar.gz"
#(Optional) Verify checksum
$curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_checksums.txt" | grep $PLATF
$tar -xzf eksctl_$PLATFORM.tar.gz -C /tmp && rm eksctl_$PLATFORM.tar.gz
$sudo mv /tmp/eksctl /usr/local/bin
```

9. **Creating the cluster with 3 worker node and 1 master node with the below command:**

```
$eksctl create cluster --name test-cluster --version 1.29 --region eu-west-1 --nodegroup-name linux-nodes --no
```



It will take around 10 to 15 mins for cluster to be ready to use. After the said time you can check the status of the cluster by running the below command:

```
$eksctl get cluster
```

```
ubuntu@ip-            :~$ eksctl get cluster
NAME            REGION          EKSCTL CREATED
test-cluster    eu-west-1       True
```

When the cluster is ready with 3 node machines running in eu-west-1 region and 1 master running in eu-west-1 as per the availability zones.

10. **Install cert-manager and its CRD's**

    a. Add and update the Jetstack Helm repository

    ```
    $helm repo add jetstack https://charts.jetstack.io --force-update
    ```

    b. Install the CRD's(Custom Resource Definition) of Certmanager using the following command

    ```
    $kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.13.3/cert-manager.crds.
    ```

    c. Install the Latest cert-manager using helm

    ```
    $helm install cert-manager jetstack/cert-manager --namespace cert-manager --create-namespace --version v1.1
    ```

    d. Now, Install the GlobalSign's Certmanager-Atlas CRD. Once it is installed, then it is ready to handle Atlas Certificate requests.

    ```
    $kubectl apply -f https://github.com/globalsign/atlas-cert-manager/releases/download/v0.0.1/install.yaml
    ```

11. **Label the cert-manager namespace to disable resource validation**

    ```
    $kubectl label namespace cert-manager certmanager.k8s.io/disable-validation=true
    ```

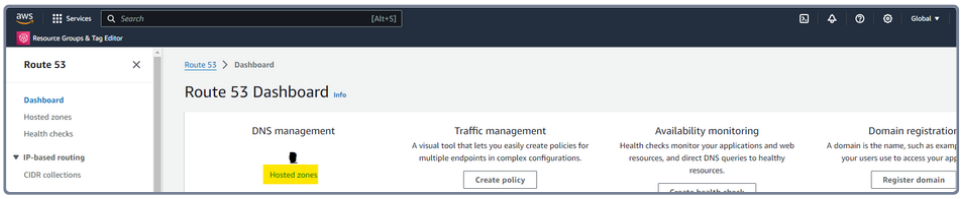
12. **Now, Install Nginx-ingress-controller in namespace cert-manager**

    ```
    $helm upgrade --install ingress-nginx ingress-nginx --repo https://kubernetes.github.io/ingress-nginx --namesp
    $kubectl get svc -n cert-manager
    ```

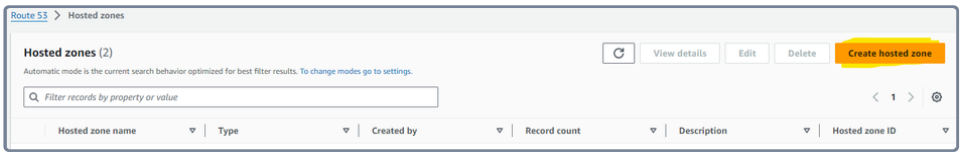
13. **Create A record in your Route 53 to the Hosted Zone for the below created Load Balancer IP(Here the cluster IP is 10.100.96.178)**

```
ubuntu@ip-                :~$ kubectl get svc -n cert-manager
NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP                                                              POR
T(S)                        AGE
cert-manager                         ClusterIP      10.100.241.61    <none>                                                                   940
2/TCP                       94s
cert-manager-webhook                 ClusterIP      10.100.246.181   <none>                                                                   443
/TCP                        94s
ingress-nginx-controller             LoadBalancer   10.100.96.178    a99d49e4a1a62409689a287279f59e79-2014922034.eu-west-1.elb.amazonaws.com   80:
32365/TCP,443:32754/TCP     21s
ingress-nginx-controller-admission   ClusterIP      10.100.219.192   <none>                                                                   443
/TCP                        21s
```

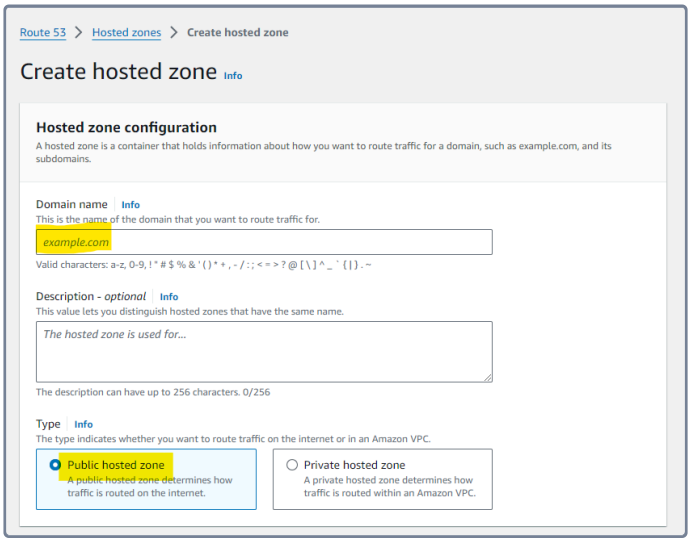
14. **As soon as the ingress-nginx-controller get the EXTERNAL-IP value with extension *.eu-west-1.elb.amazonaws.com, Add this value as A record into hosted zone. It would be in the sync within 60sec.**
    Note:- Before creating the hosted zone kindly make sure you have the valid domain.
15. **Creating A record over AWS Route53**
    a. Go to https://us-east-1.console.aws.amazon.com/route53/v2/home?region=eu-west-1#Dashboard and click on "Hosted zones".



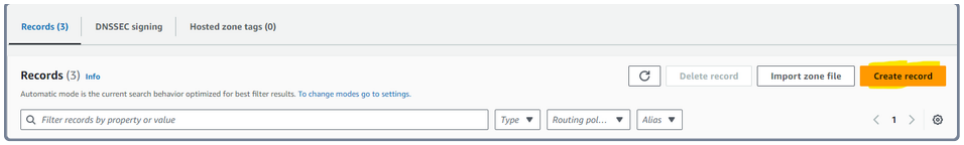
    b. Click on "Create hosted zone"



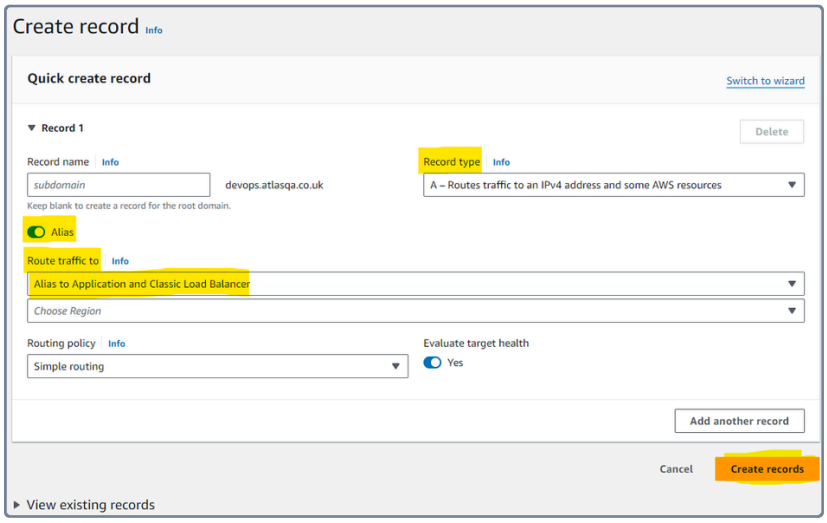
    c. Enter the name followed by your actual domain name and make sure the "Public hosted zone" should be selected:



    d. After creating the hosted zone, you would get sone NS record along with SOA record. Now add the NS records into your domain registrar
    e. After adding the NS into domain registrar your hosted zone is now ready to accept traffic, Now you can create the A record into the hosted zone:
        i. Into your hosted zone, Click on "Create record"



ii. On the next screen make sure that Record Type is "A" and "Alias" are selected. Also make sure that "Route traffic to" "Alias to Application and Classic Load Balancer is selected". After selecting the required fields click on "Create records":



iii. The record would be created and it would take around 60sec to get in the sync.

16. Create GlobalSign Issuer to issue a TLS certificate for your Ingress using the following steps:-
    a. Create a secret to store the GlobalSign's ATLAS account api_key, secrets along with mTLS and private key(You can get these API credentials from GlobalSign's Team)

```
$kubectl create secret generic issuer-credentials --from-literal=apikey=$API_KEY --from-literal=apisecret=$
```

    b. **Create an Issuer of GlobalSign.**

```
cat <<EOF | kubectl apply -f -
apiVersion: hvca.globalsign.com/v1alpha1
kind: Issuer
metadata:
  name: gs-issuer
  namespace: cert-manager
spec:
  authSecretName: "issuer-credentials"
  url: "https://emea.api.hvca.globalsign.com:8443/v2"
EOF
```

    c. Create Certificate Resource with the following Configuration

```
cat <<EOF | kubectl apply -f -
apiVersion: cert-manager.io/v1
kind: Certificate
```

```
metadata:
  name: pki.atlasqa.co.uk
  namespace: cert-manager
spec:
  # Secret names are always required.
  secretName: www.atlasqa.co.uk

  duration: 2160h # 90d
  renewBefore: 360h # 15d
  subject:
  #  organizations:
  #    - jetstack
  # The use of the common name field has been deprecated since 2000 and is
  # discouraged from being used.
  commonName: pki.atlasqa.co.uk
  isCA: false
  privateKey:
    algorithm: RSA
    encoding: PKCS1
    size: 2048
  usages:
    - server auth
    #- client auth
  # At least one of a DNS Name, URI, or IP address is required.
#  dnsNames:
#    -
#www.atlasqa.co.uk
  # Issuer references are always required.
  issuerRef:
    name: gs-issuer
    # We can reference ClusterIssuers by changing the kind here.
    # The default value is Issuer (i.e. a locally namespaced Issuer)
    kind: Issuer
    # This is optional since cert-manager will default to this value however
    # if you are using an external issuer, change this to that issuer group.
    group: hvca.globalsign.com
EOF
```

d. At times the certificate object can take couple of seconds to become READY.

```
ubuntu@ip-    :~$ kubectl apply -f object-cert.yml
certificate.cert-manager.io/pki.atlasqa.co.uk created
ubuntu@ip-    :~$ kubectl get certificate -n cert-manager
NAME               READY   SECRET             AGE
pki.atlasqa.co.uk  True    www.atlasqa.co.uk  4s
```

17. **Securing nginx ingress resource by the below configuration:**

```
cat <<EOF | kubectl apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx
  namespace: cert-manager
  annotations:
```

```
    cert-manager.io/issuer: GS-issuer
    kubernetes.io/ingress.class: nginx
spec:
  tls:
    - hosts:
        - pki.atlasqa.co.uk
      secretName: www.atlasqa.co.uk
  rules:
    - host: pki.atlasqa.co.uk
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: example-service
                port:
                  number: 80
EOF
```

18. **The ingress resource that has been created could take up to 1min to get the load balancer URL as ADDRESS.**

```
ubuntu@ip-          :~$ kubectl apply -f ingress.yml
Warning: annotation "kubernetes.io/ingress.class" is deprecated, please use 'spec.ingressClassName' instead
ingress.networking.k8s.io/nginx-ingress created
ubuntu@ip-          :~$ kubectl get ingress -n cert-manager
NAME            CLASS    HOSTS                ADDRESS                                                                  PORTS     AG
nginx-ingress   <none>   devops.atlasqa.co.uk   a215bae5711314b3cb9a9bd1556950fb-1850122743.us-east-2.elb.amazonaws.com   80, 443   87
```

19. **Attaching the screen shot:**