**2. Implement SSL Discovery Host Demo**

--- Reference - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/08-NEW-ELB-Application-LoadBalancers/08-10-Ingress-SSL-Discovery-host>

--- let’s go to the github repository and implement ssl discovery with host.

**Introduction**

--- Automatically discover SSL Certificate from AWS Certificate Manager Service using spec.rules.host

--- In this approach, with the specified domain name if we have the SSL Certificate created in AWS Certificate Manager, that certificate will be automatically detected and associated to Application Load Balancer.

--- We don't need to get the SSL Certificate ARN and update it in Kubernetes Ingress Manifest

--- Discovers via Ingress rule host and attaches a cert for app102.stacksimplify.com or \*.stacksimplify.com to the ALB

**Discover via Ingress "spec.rules.host"**

--- **04-ALB-Ingress-CertDiscovery-host.yml**

# Annotations Reference: https://kubernetes-sigs.github.io/aws-load-balancer-controller/latest/guide/ingress/annotations/

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  name: ingress-certdiscoveryhost-demo

  annotations:

    # Load Balancer Name

    alb.ingress.kubernetes.io/load-balancer-name: certdiscoveryhost-ingress

    # Ingress Core Settings

    #kubernetes.io/ingress.class: "alb" (OLD INGRESS CLASS NOTATION - STILL WORKS BUT RECOMMENDED TO USE IngressClass Resource)

    alb.ingress.kubernetes.io/scheme: internet-facing

    # Health Check Settings

    alb.ingress.kubernetes.io/healthcheck-protocol: HTTP

    alb.ingress.kubernetes.io/healthcheck-port: traffic-port

    #Important Note:  Need to add health check path annotations in service level if we are planning to use multiple targets in a load balancer

    alb.ingress.kubernetes.io/healthcheck-interval-seconds: '15'

    alb.ingress.kubernetes.io/healthcheck-timeout-seconds: '5'

    alb.ingress.kubernetes.io/success-codes: '200'

    alb.ingress.kubernetes.io/healthy-threshold-count: '2'

    alb.ingress.kubernetes.io/unhealthy-threshold-count: '2'

    ## SSL Settings

    alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}, {"HTTP":80}]'

    #alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:us-east-1:180789647333:certificate/632a3ff6-3f6d-464c-9121-b9d97481a76b

    #alb.ingress.kubernetes.io/ssl-policy: ELBSecurityPolicy-TLS-1-1-2017-01 #Optional (Picks default if not used)

    # SSL Redirect Setting

    alb.ingress.kubernetes.io/ssl-redirect: '443'

    # External DNS - For creating a Record Set in Route53

    external-dns.alpha.kubernetes.io/hostname: default102.stacksimplify.com

spec:

  ingressClassName: my-aws-ingress-class   # Ingress Class

  defaultBackend:

    service:

      name: app3-nginx-nodeport-service

      port:

        number: 80

  rules:

    - host: app102.stacksimplify.com

      http:

        paths:

          - path: /

            pathType: Prefix

            backend:

              service:

                name: app1-nginx-nodeport-service

                port:

                  number: 80

    - host: app202.stacksimplify.com

      http:

        paths:

          - path: /

            pathType: Prefix

            backend:

              service:

                name: app2-nginx-nodeport-service

                port:

                  number: 80

# Important Note-1: In path based routing order is very important, if we are going to use  "/\*", try to use it at the end of all rules.

# 1. If  "spec.ingressClassName: my-aws-ingress-class" not specified, will reference default ingress class on this kubernetes cluster

# 2. Default Ingress class is nothing but for which ingress class we have the annotation `ingressclass.kubernetes.io/is-default-class: "true"`

--- **01-Nginx-App1-Deployment-and-NodePortService.yml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: app1-nginx-deployment

  labels:

    app: app1-nginx

spec:

  replicas: 1

  selector:

    matchLabels:

      app: app1-nginx

  template:

    metadata:

      labels:

        app: app1-nginx

    spec:

      containers:

        - name: app1-nginx

          image: stacksimplify/kube-nginxapp1:1.0.0

          ports:

            - containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

  name: app1-nginx-nodeport-service

  labels:

    app: app1-nginx

  annotations:

#Important Note:  Need to add health check path annotations in service level if we are planning to use multiple targets in a load balancer

    alb.ingress.kubernetes.io/healthcheck-path: /app1/index.html

spec:

  type: NodePort

  selector:

    app: app1-nginx

  ports:

    - port: 80

      targetPort: 80

--- **02-Nginx-App2-Deployment-and-NodePortService.yml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: app2-nginx-deployment

  labels:

    app: app2-nginx

spec:

  replicas: 1

  selector:

    matchLabels:

      app: app2-nginx

  template:

    metadata:

      labels:

        app: app2-nginx

    spec:

      containers:

        - name: app2-nginx

          image: stacksimplify/kube-nginxapp2:1.0.0

          ports:

            - containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

  name: app2-nginx-nodeport-service

  labels:

    app: app2-nginx

  annotations:

#Important Note:  Need to add health check path annotations in service level if we are planning to use multiple targets in a load balancer

    alb.ingress.kubernetes.io/healthcheck-path: /app2/index.html

spec:

  type: NodePort

  selector:

    app: app2-nginx

  ports:

    - port: 80

      targetPort: 80

--- **03-Nginx-App3-Deployment-and-NodePortService.yml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: app3-nginx-deployment

  labels:

    app: app3-nginx

spec:

  replicas: 1

  selector:

    matchLabels:

      app: app3-nginx

  template:

    metadata:

      labels:

        app: app3-nginx

    spec:

      containers:

        - name: app2-nginx

          image: stacksimplify/kubenginx:1.0.0

          ports:

            - containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

  name: app3-nginx-nodeport-service

  labels:

    app: app3-nginx

  annotations:

#Important Note:  Need to add health check path annotations in service level if we are planning to use multiple targets in a load balancer

    alb.ingress.kubernetes.io/healthcheck-path: /index.html

spec:

  type: NodePort

  selector:

    app: app3-nginx

  ports:

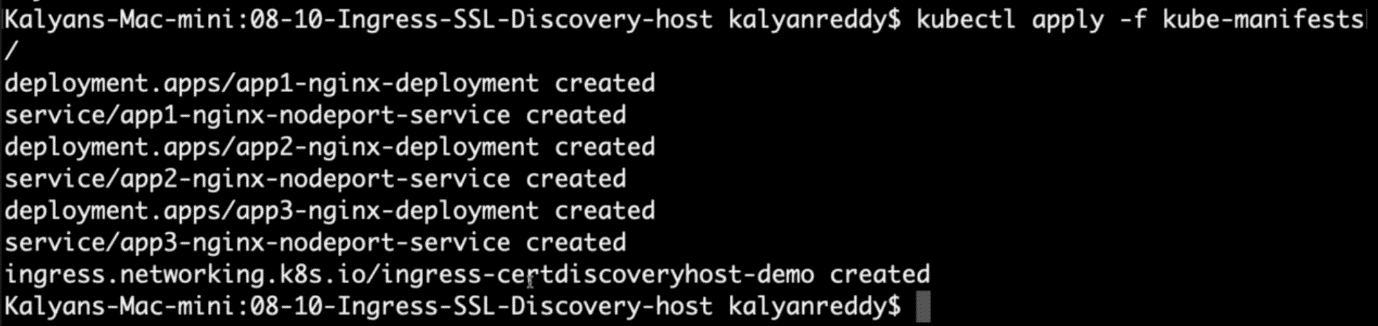
    - port: 80

      targetPort: 80

**Deploy all Application Kubernetes Manifests and Verify**

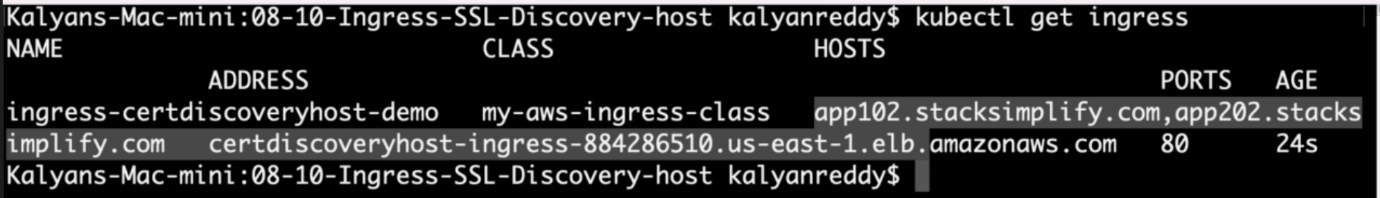
**# Deploy kube-manifests**

--- **kubectl apply -f kube-manifests/**



**# Verify Ingress Resource**

--- **kubectl get ingress**

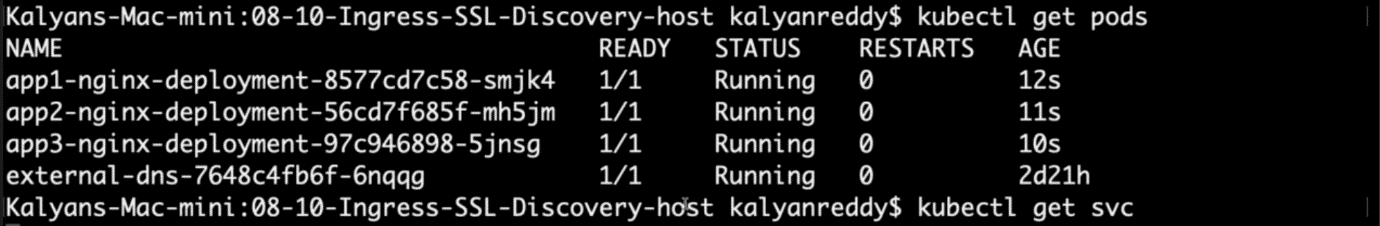


--- **note** – under host, it had already created sub domains. Here you can also find the load balancer or ingress service end point.

**# Verify Apps**

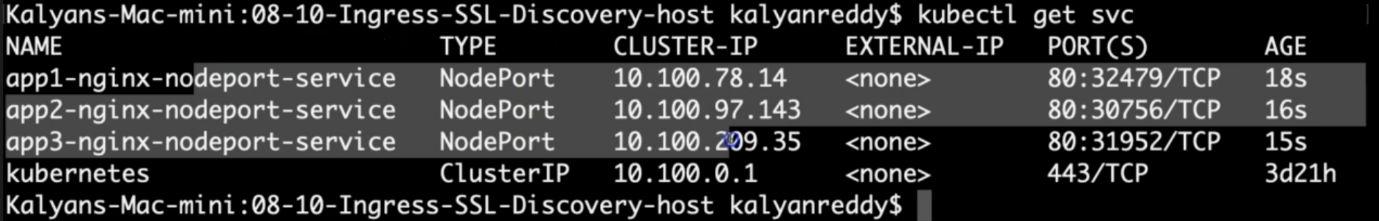
--- **kubectl get deploy**

--- **kubectl get pods**



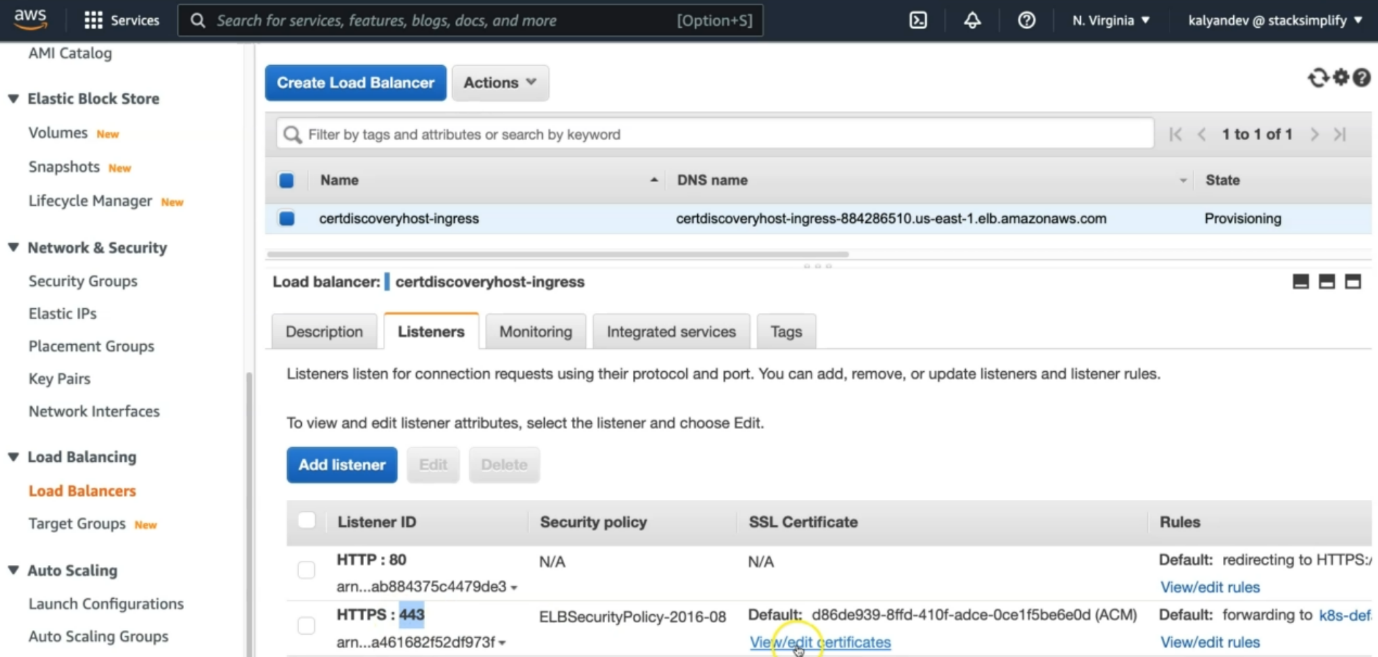
**# Verify NodePort Services**

--- **kubectl get svc**

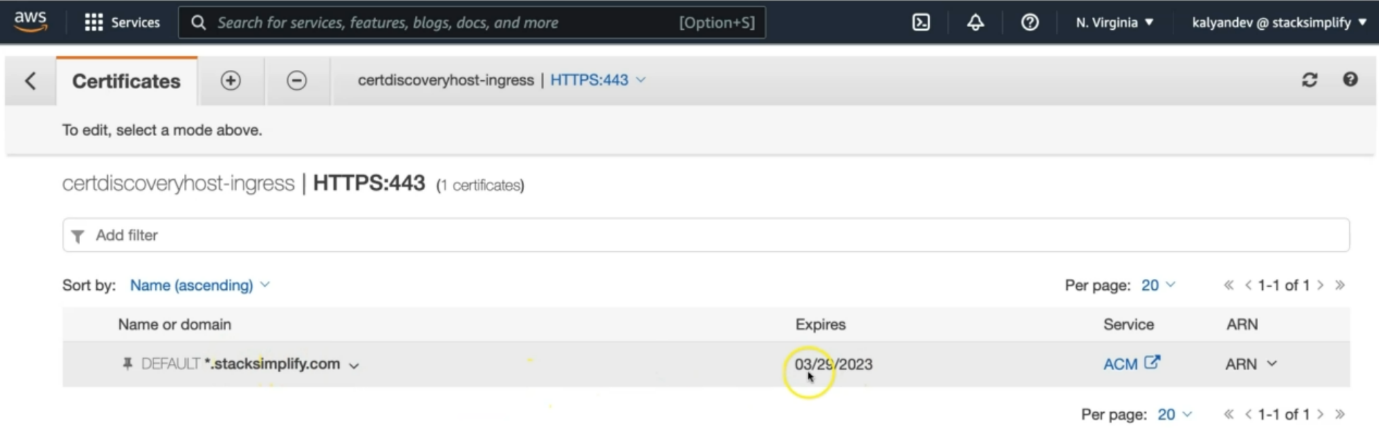


**Verify Load Balancer & Target Groups**

--- Load Balancer - Listeners (Verify both 80 & 443)



--- click on view/edit certificates.



--- it is successfully added the \*.stacksimplify.com certificate.

--- Load Balancer - Rules (Verify both 80 & 443 listeners)

--- Target Groups - Group Details (Verify Health check path)

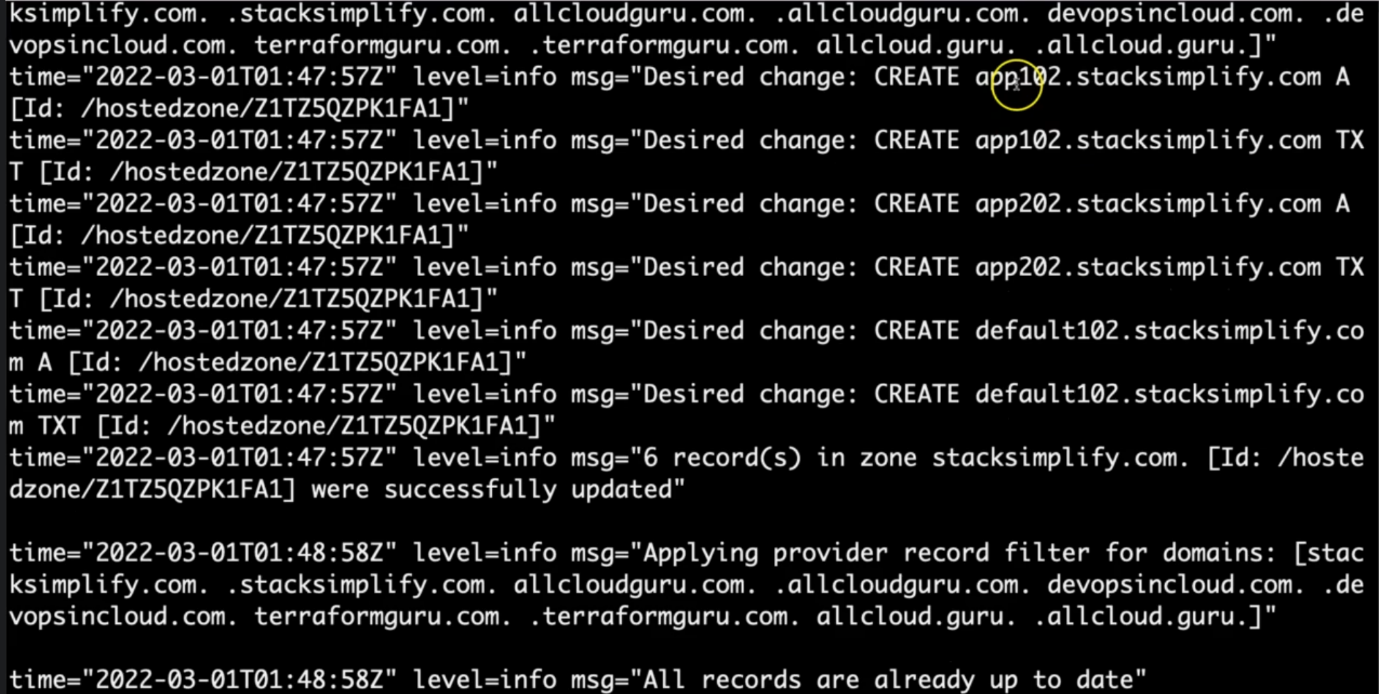
--- Target Groups - Targets (Verify all 3 targets are healthy)

--- PRIMARILY VERIFY - CERTIFICATE ASSOCIATED TO APPLICATION LOAD BALANCER

**Verify External DNS Log**

**# Verify External DNS logs**

--- **kubectl logs -f $(kubectl get po | egrep -o 'external-dns[A-Za-z0-9-]+')**



--- it had created, all the subdomain.

**Verify Route53**

--- Go to Services -> Route53

--- You should see Record Sets added for

1. default102.stacksimplify.com
2. app102.stacksimplify.com
3. app202.stacksimplify.com

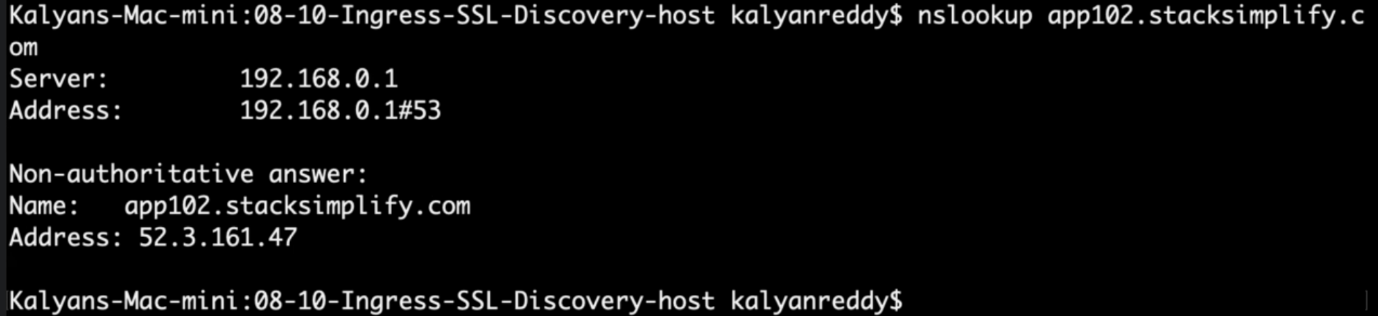
**Access Application using newly registered DNS Name**

--- Perform nslookup tests before accessing Application

--- Test if our new DNS entries registered and resolving to an IP Address

**# nslookup commands**

--- **nslookup default102.stacksimplify.com**



--- it is resolved to an ip address, so this is working.

--- **nslookup app102.stacksimplify.com**

--- **nslookup app202.stacksimplify.com**

**Positive Case: Access Application using DNS domain**

**# Access App1**

--- <http://app102.stacksimplify.com/app1/index.html>

**# Access App2**

--- <http://app202.stacksimplify.com/app2/index.html>

**# Access Default App (App3)**

--- <http://default102.stacksimplify.com>

**Clean Up**

**# Delete Manifests**

--- **kubectl delete -f kube-manifests/**

**## Verify Route53 Record Set to ensure our DNS records got deleted**

--- Go to Route53 -> Hosted Zones -> Records

--- The below records should be deleted automatically

1. default102.stacksimplify.com

2. app102.stacksimplify.com

3.app202.stacksimplify.com

**References**

--- <https://kubernetes-sigs.github.io/aws-load-balancer-controller/v2.4/guide/ingress/cert_discovery/>