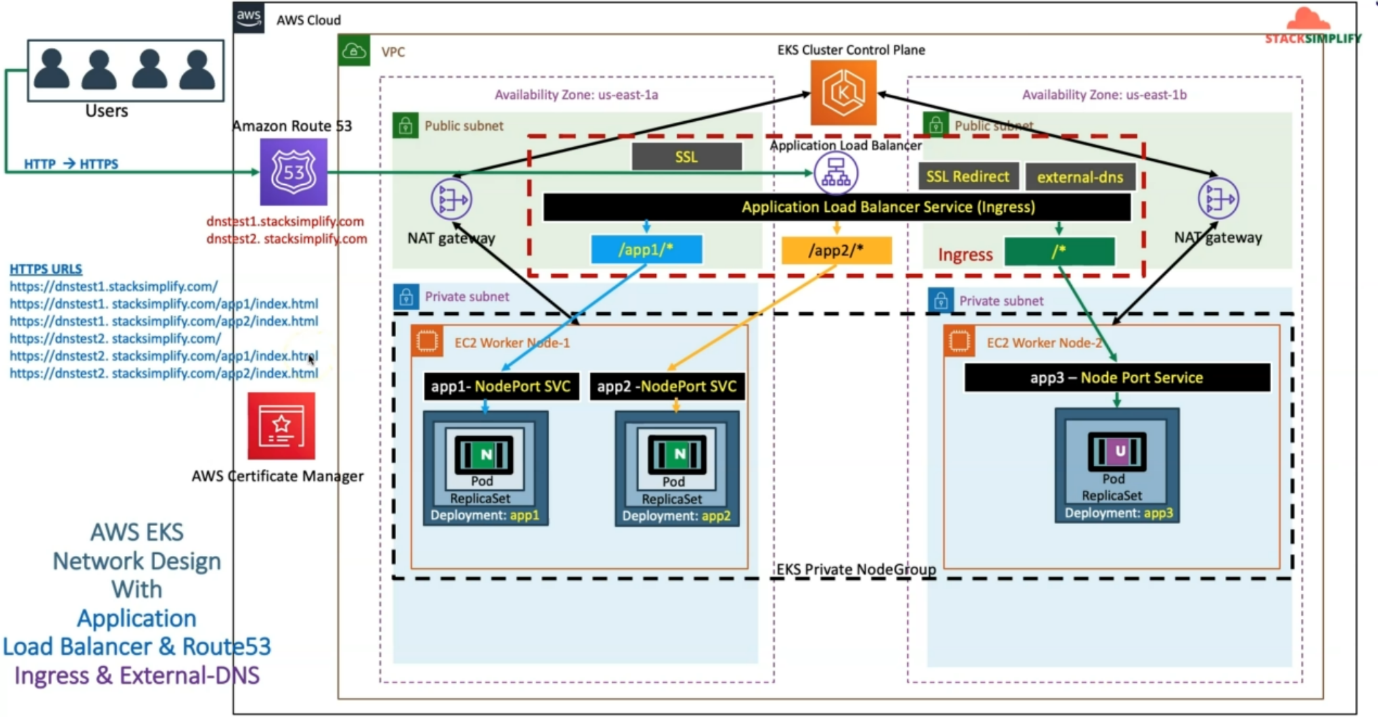
**5. Ingress Service Demo with External DNS**

--- Reference - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/08-NEW-ELB-Application-LoadBalancers/08-07-Use-ExternalDNS-for-k8s-Ingress>

--- in this lesion, we are going to add that external DNS annotation for our Kubernetes ingress service and then test it and then see that whether our DNS records got automatically created by external DNS in Route 53 hosted zones. So, let's see the network design once before moving on.



--- in the ingress service, we are going to add the external DNS related annotations and after that, it is going to automatically add that DNS names in the AWS Route 53 hosted zones, and that will

be able to test those DNS names by accessing them directly in the browser.

--- now let's go back to our GitHub repository

**Update Ingress manifest by adding External DNS Annotation**

--- Added annotation with two DNS Names

1. dnstest901.kubeoncloud.com
2. dnstest902.kubeoncloud.com

--- Once we deploy the application, we should be able to access our Applications with both DNS Names.

--- File Name: 04-ALB-Ingress-SSL-Redirect-ExternalDNS.yml

    # External DNS - For creating a Record Set in Route53

    external-dns.alpha.kubernetes.io/hostname: dnstest901.stacksimplify.com, dnstest902.stacksimplify.com

--- **note** - In your case it is going to be, replace yourdomain with your domain name.

dnstest901.yourdoamin.com

dnstest902.yourdoamin.com

--- **File Name: 04-ALB-Ingress-SSL-Redirect-ExternalDNS.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-externaldns-demo

  annotations:

    # Load Balancer Name

    alb.ingress.kubernetes.io/load-balancer-name: externaldns-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/d86de939-8ffd-410f-adce-0ce1f5be6e0d

    #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: dnstest901.stacksimplify.com, dnstest902.stacksimplify.com

spec:

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

  defaultBackend:

    service:

      name: app3-nginx-nodeport-service

      port:

        number: 80

  rules:

    - http:

        paths:

          - path: /app1

            pathType: Prefix

            backend:

              service:

                name: app1-nginx-nodeport-service

                port:

                  number: 80

          - path: /app2

            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:

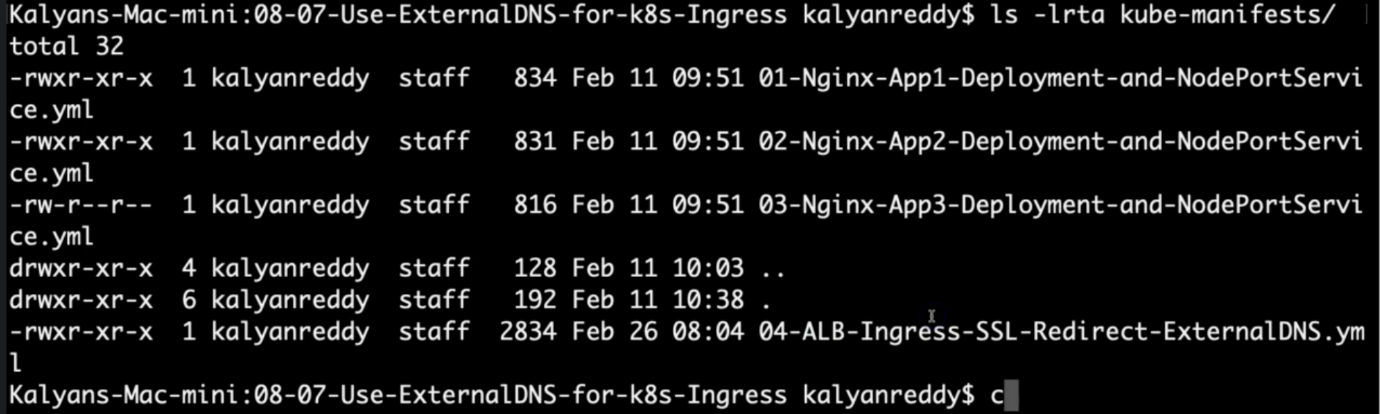
    - port: 80

      targetPort: 80

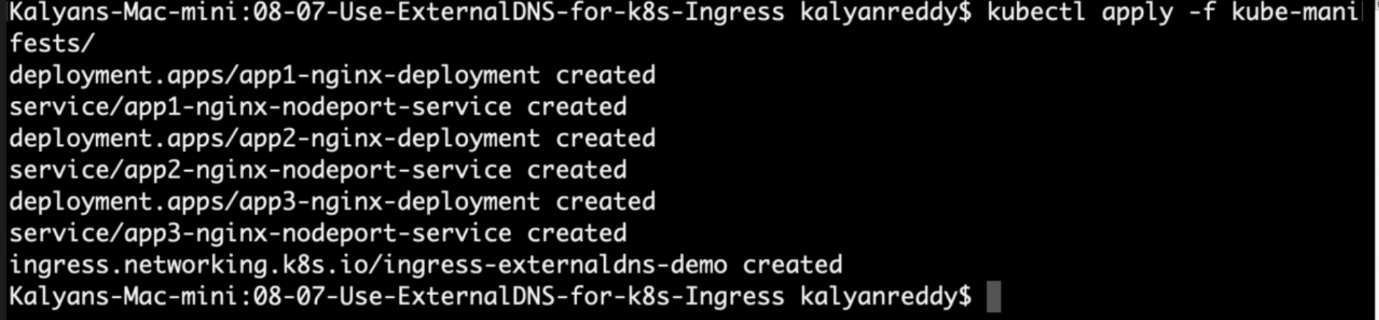
**Deploy all Application Kubernetes Manifests**

**# Deploy kube-manifests**

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



--- just to check that all the files are there.



**# Verify Ingress Resource**

--- **kubectl get ingress**

**# 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)

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

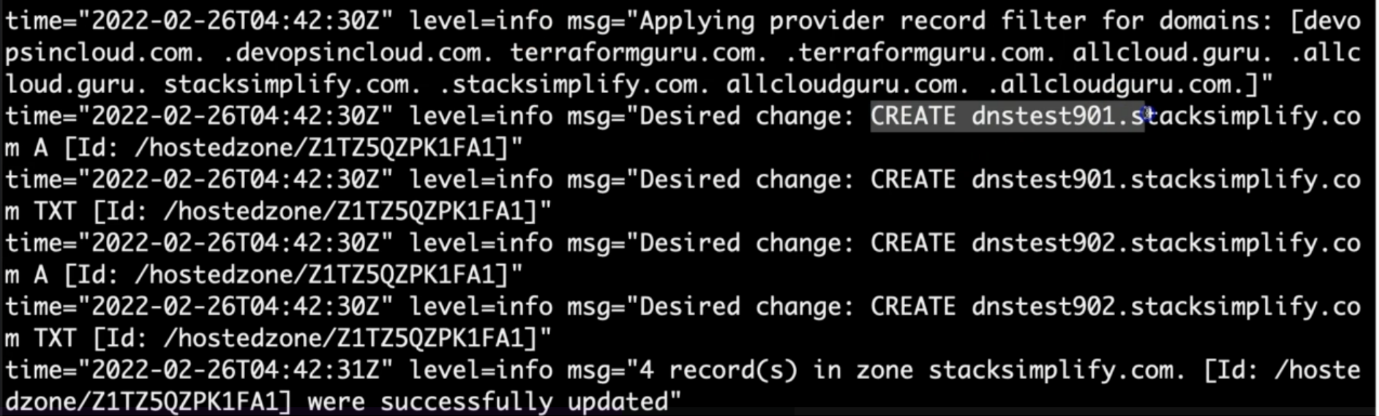
--- Target Groups - Group Details (Verify Health check path)

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

**Verify External DNS Log**

**# Verify External DNS logs**

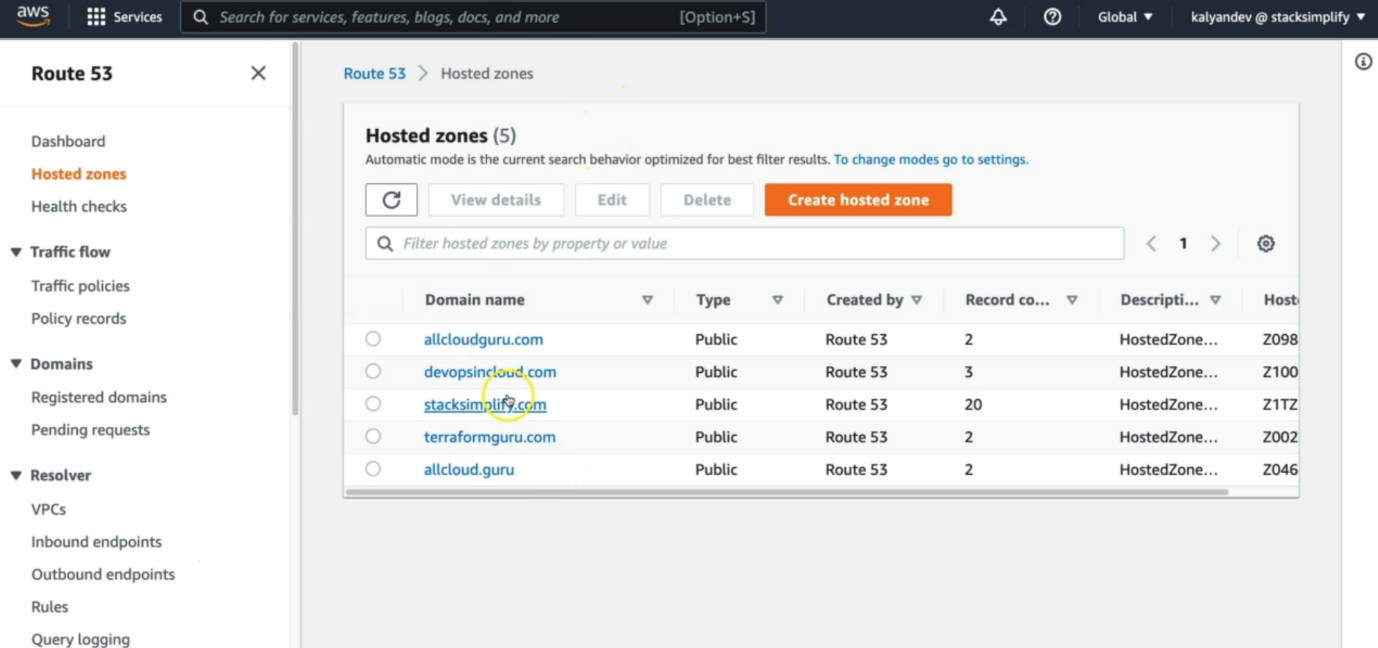
--- **kubectl logs -f $(kubectl get po | egrep -o 'external-dns[A-Za-z0-9-]+')**



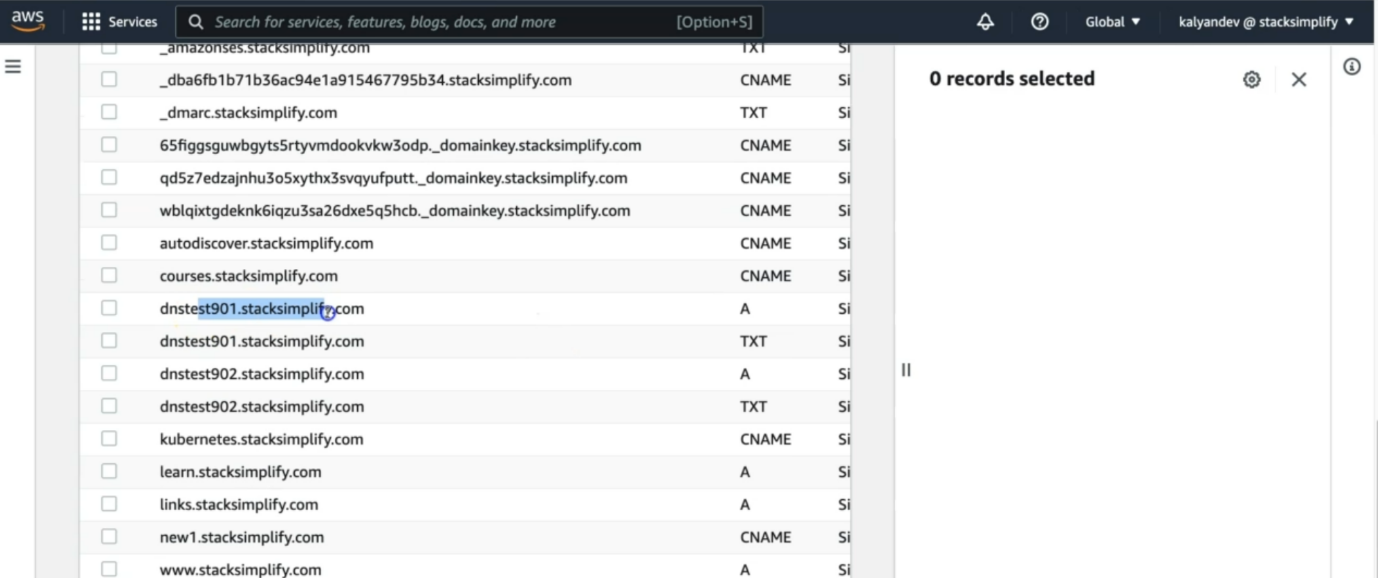
--- **note** – it created the dnstest901.stacksimplify.com and dnstest902.stacksimplify.com. this is our core goal to verify this.

**Verify Route53**

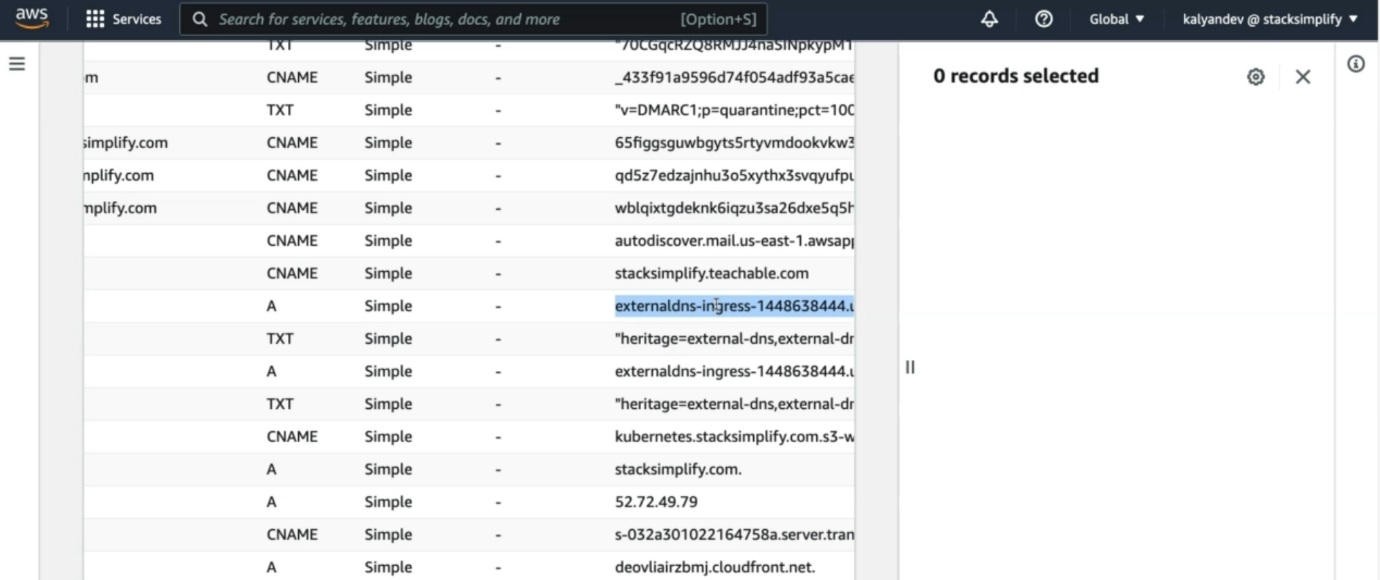
--- Go to Services -> Route53



--- click on our domain stacksimplify.com



--- You should see Record Sets added for dnstest901.stacksimplify.com, dnstest902.stacksimplify.com



--- that domain is associated with externaldns (load balancer end point)

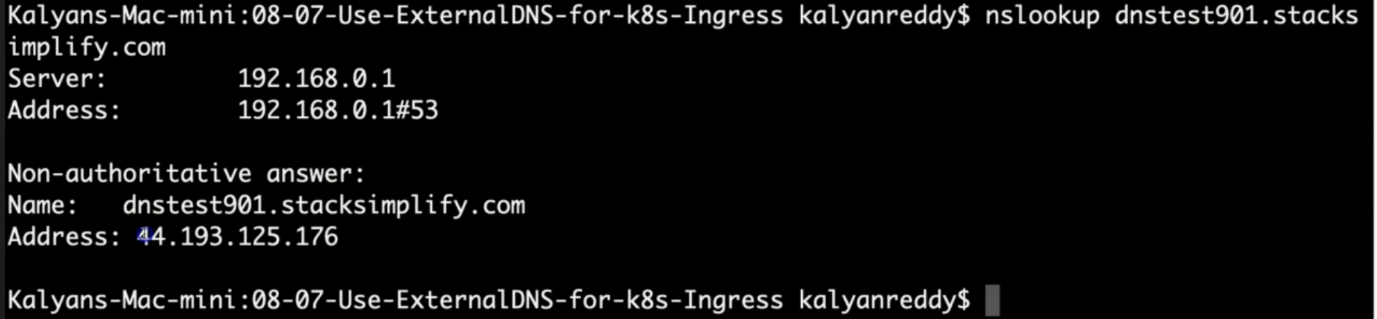
**Access Application using newly registered DNS Name**

--- Perform nslookup tests before accessing Application

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

**# nslookup commands**

--- **nslookup dnstest901.stacksimplify.com**



--- it is resolving to an IP address.

--- **nslookup dnstest902.stacksimplify.com**

**Access Application using dnstest1 domain**

**# HTTP URLs (Should Redirect to HTTPS)**

--- <http://dnstest901.stacksimplify.com/app1/index.html>

--- <http://dnstest901.stacksimplify.com/app2/index.html>

--- <http://dnstest901.stacksimplify.com/>

**Access Application using dnstest2 domain**

**# HTTP URLs (Should Redirect to HTTPS)**

--- <http://dnstest902.stacksimplify.com/app1/index.html>

--- <http://dnstest902.stacksimplify.com/app2/index.html>

--- <http://dnstest902.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. dnstest901.stacksimplify.com

2. dnstest902.stacksimplify.com

**References**

--- <https://github.com/kubernetes-sigs/external-dns/blob/master/docs/tutorials/alb-ingress.md>

--- <https://github.com/kubernetes-sigs/external-dns/blob/master/docs/tutorials/aws.md>