**4. Deploy Ingress default backend and verify and clean-up**

--- **Reference** - <https://github.com/stacksimplify/aws-eks-kubernetes-masterclass/tree/master/08-NEW-ELB-Application-LoadBalancers/08-02-ALB-Ingress-Basics>

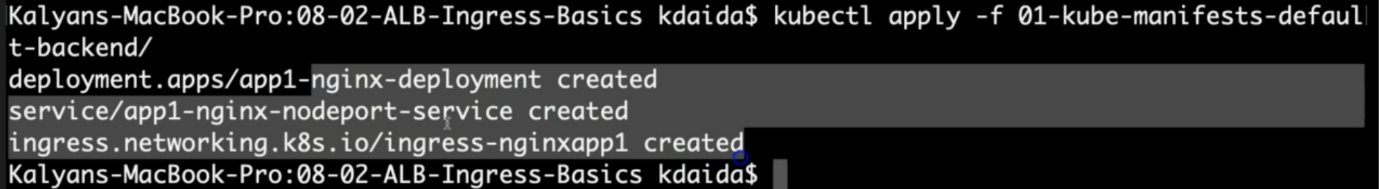
--- **note** – in previous lesion, we have successfully created ingress manifest with default backend. Now we are going to deploy it and test it.

**# Change Directory**

--- **cd 08-02-ALB-Ingress-Basics**

**# Deploy kube-manifests**

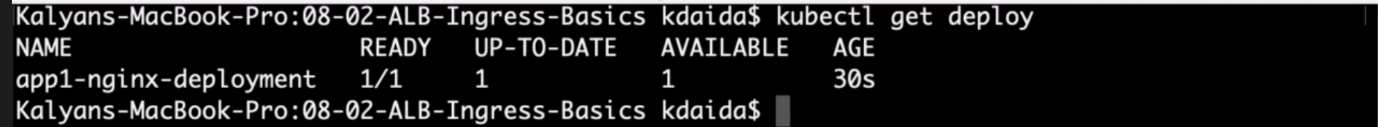
--- **kubectl apply -f 01-kube-manifests-default-backend/**



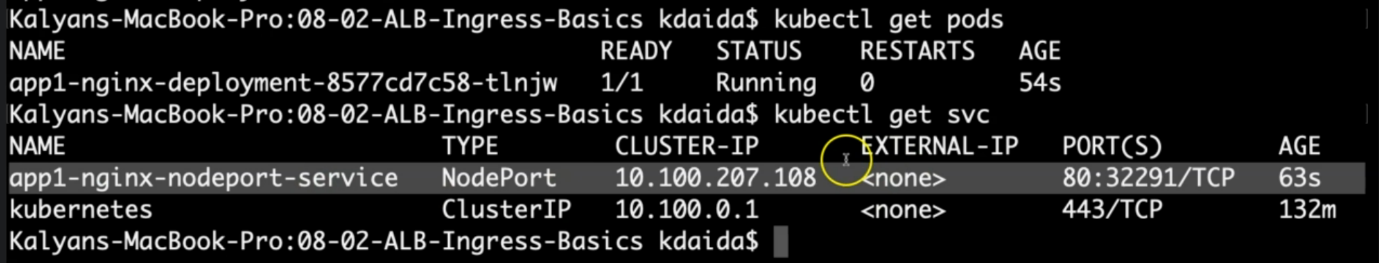
--- **note** – it created 3 things for us, app1-nginx-deployment, app1-nginx-nodeport-service, ingress-nginxapp1.

**# Verify k8s Deployment and Pods**

--- **kubectl get deploy**

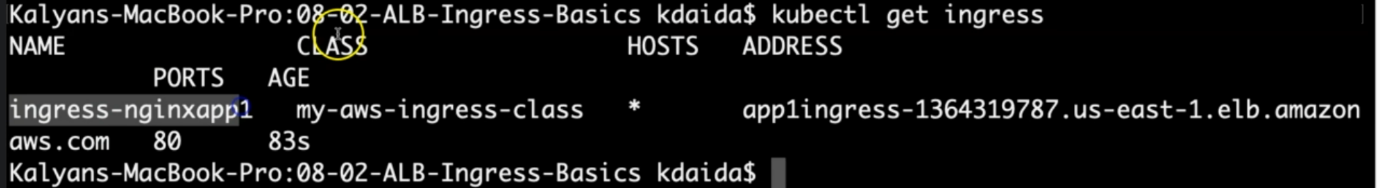


--- **kubectl get pods**



**# Verify Ingress (Make a note of Address field)**

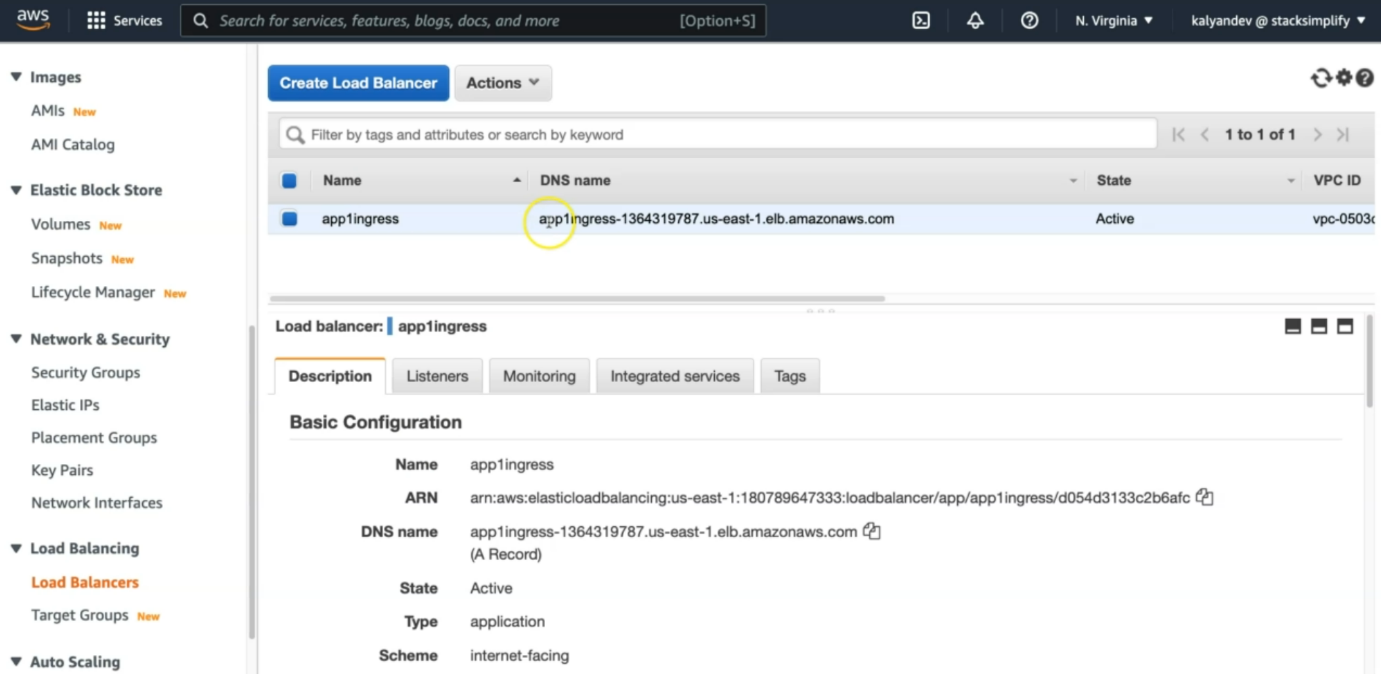
--- **kubectl get ingress**



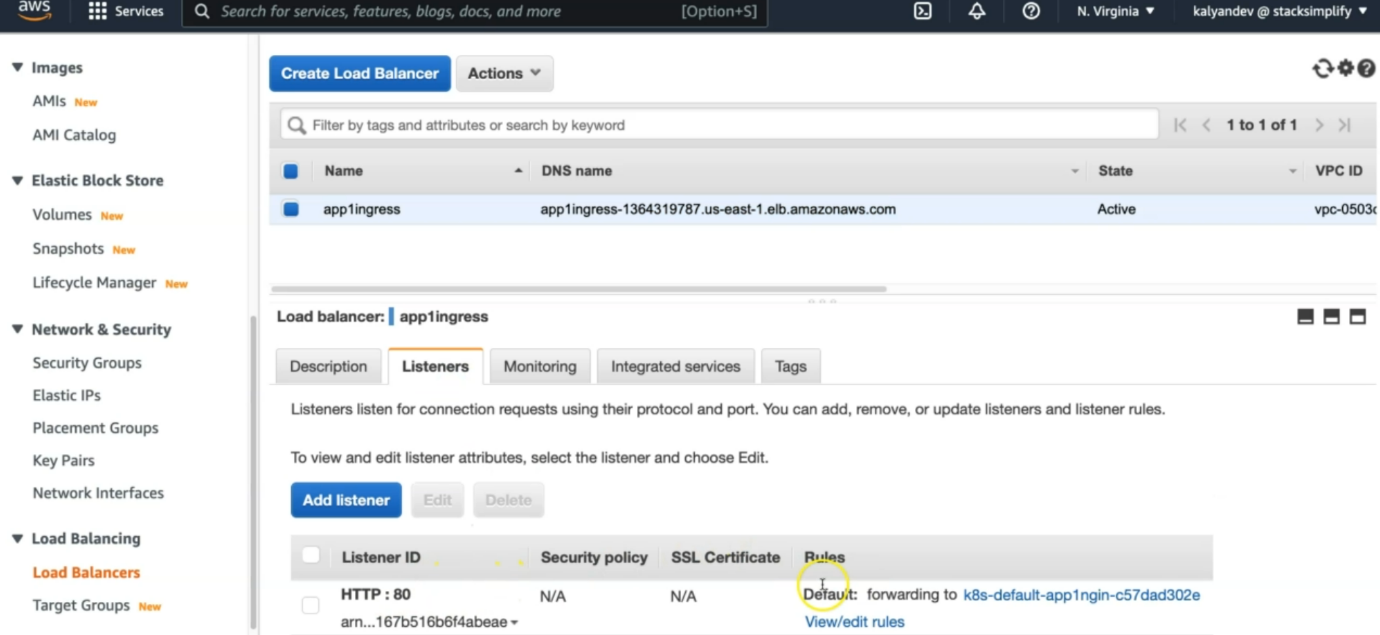
--- **note** - the last one is the name of the load balancer.

**Obsevation:**

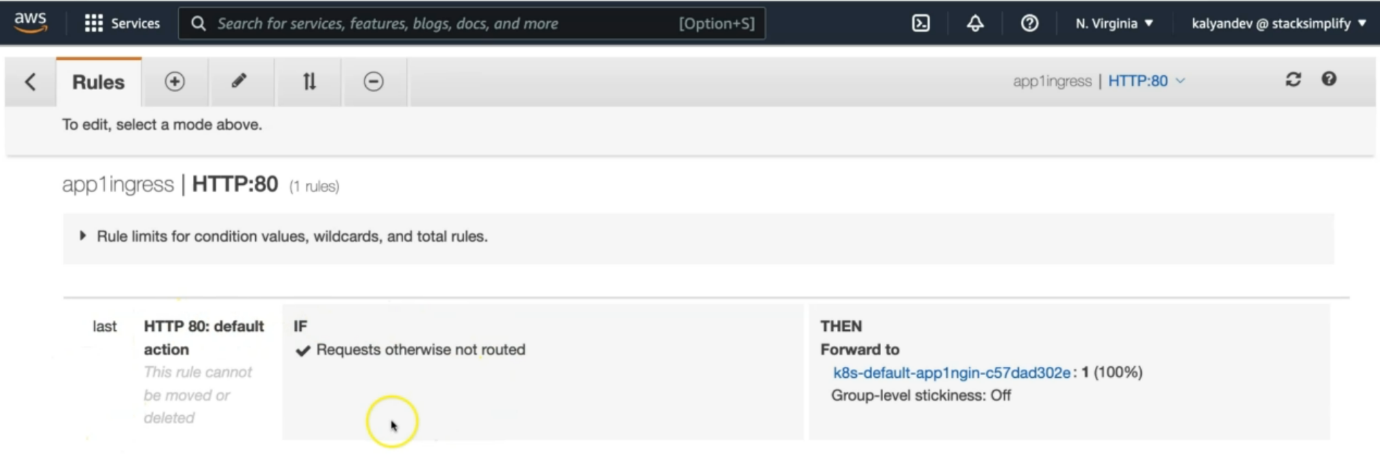
1. Verify the ADDRESS value, we should see something like "app1ingress-1334515506.us-east-1.elb.amazonaws.com"



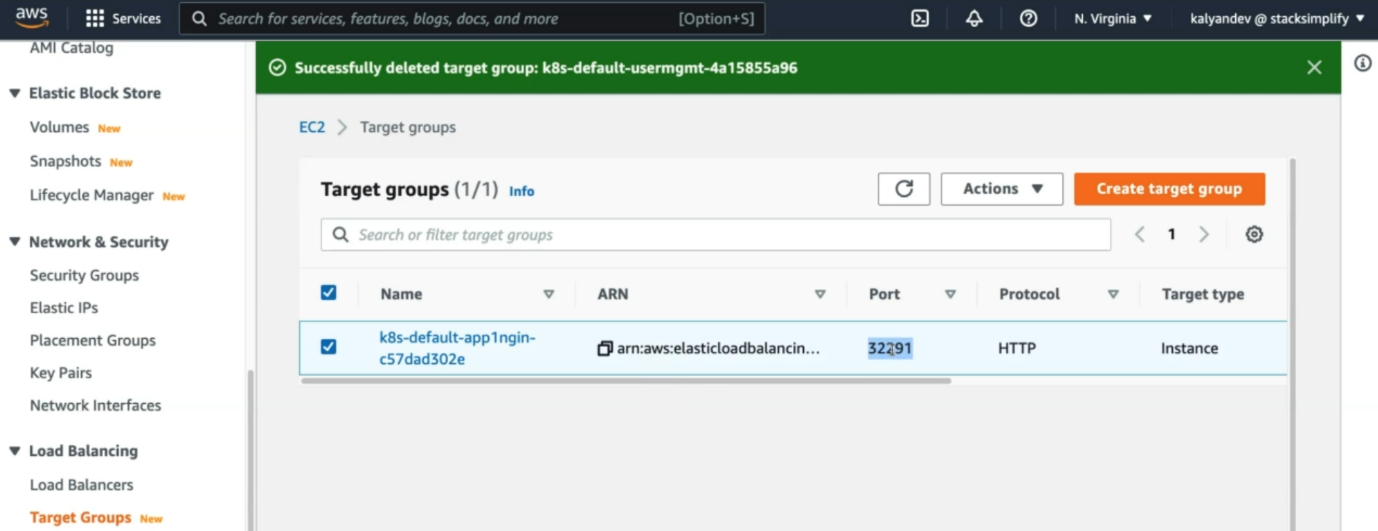
--- **note** - the name of the load balancer is app1ingress and DNS name should be a match.



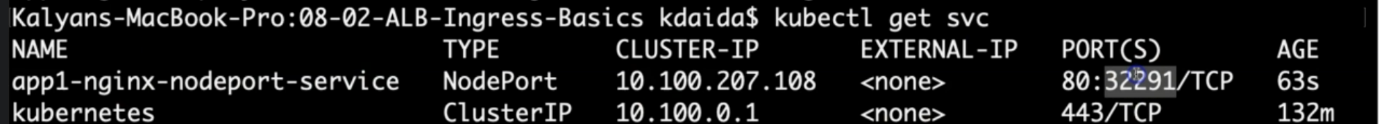
--- **note** - in the listeners, it added HTTP:80 listener. Click on the view/edit rule option.



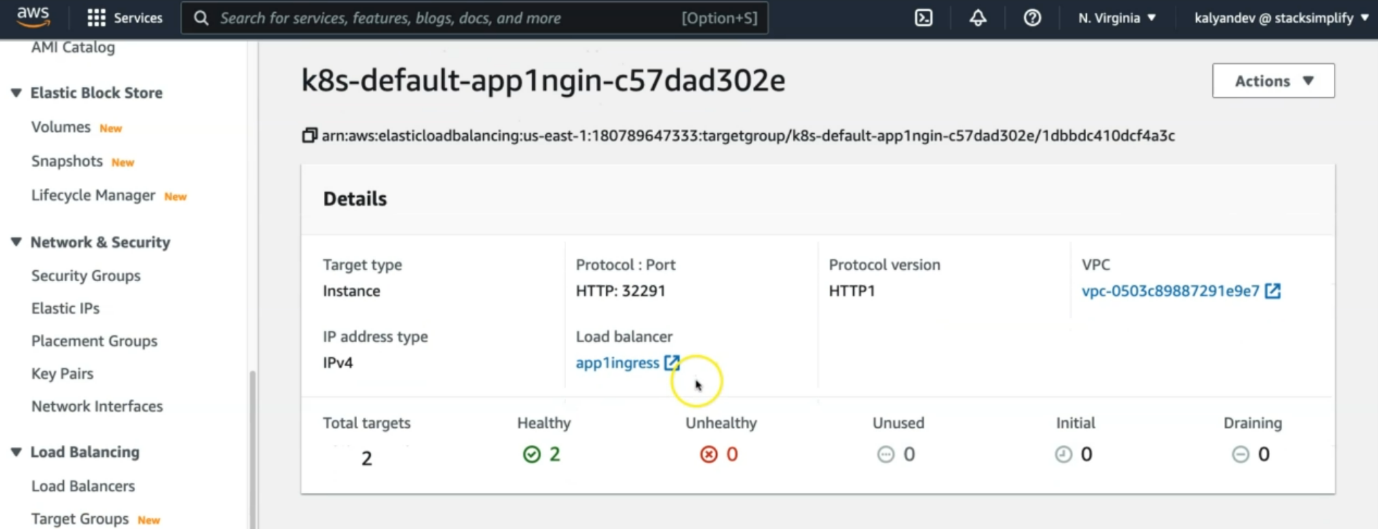
--- **note** – it is added as default backend and it is forwarding traffic to some target group. Go to the target group.



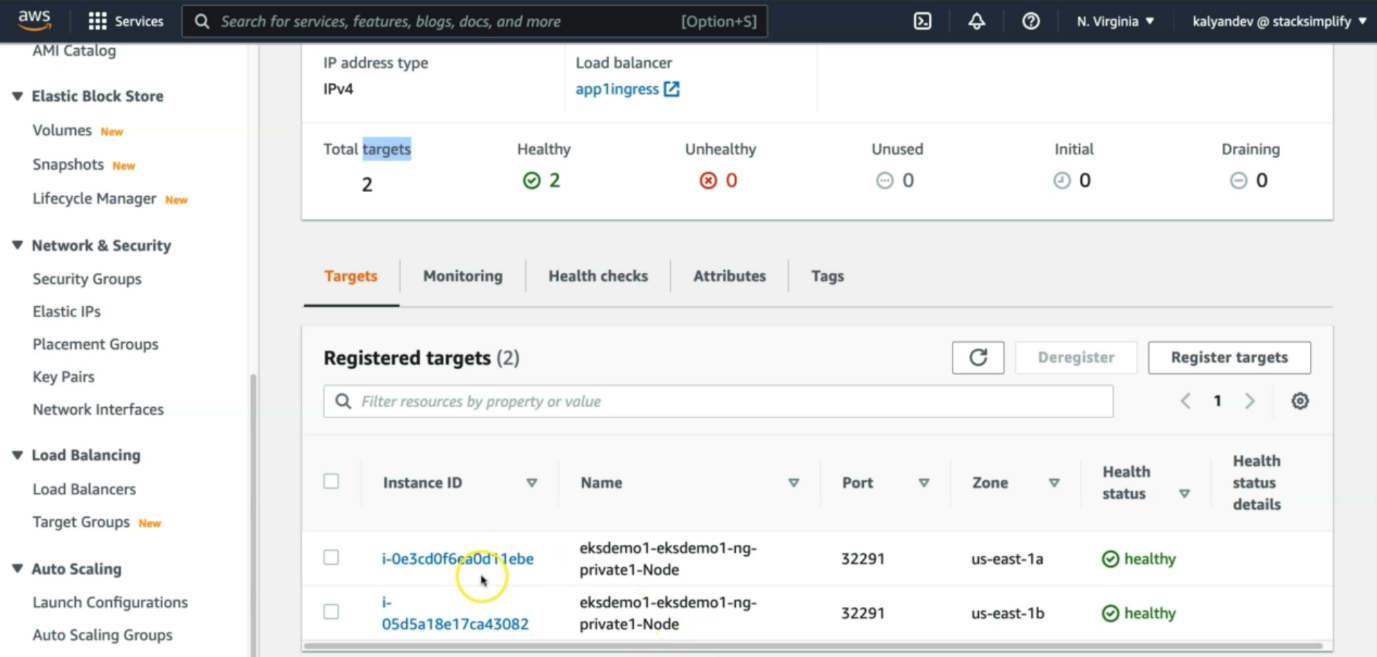
--- the node port is same as CLI. This means is that you node is registered here. The protocol is HTTP, target type instance.



--- let’s go inside of the target group.



--- this one is associated with the load balancer app1ingress and total target is 2 and they are healthy.

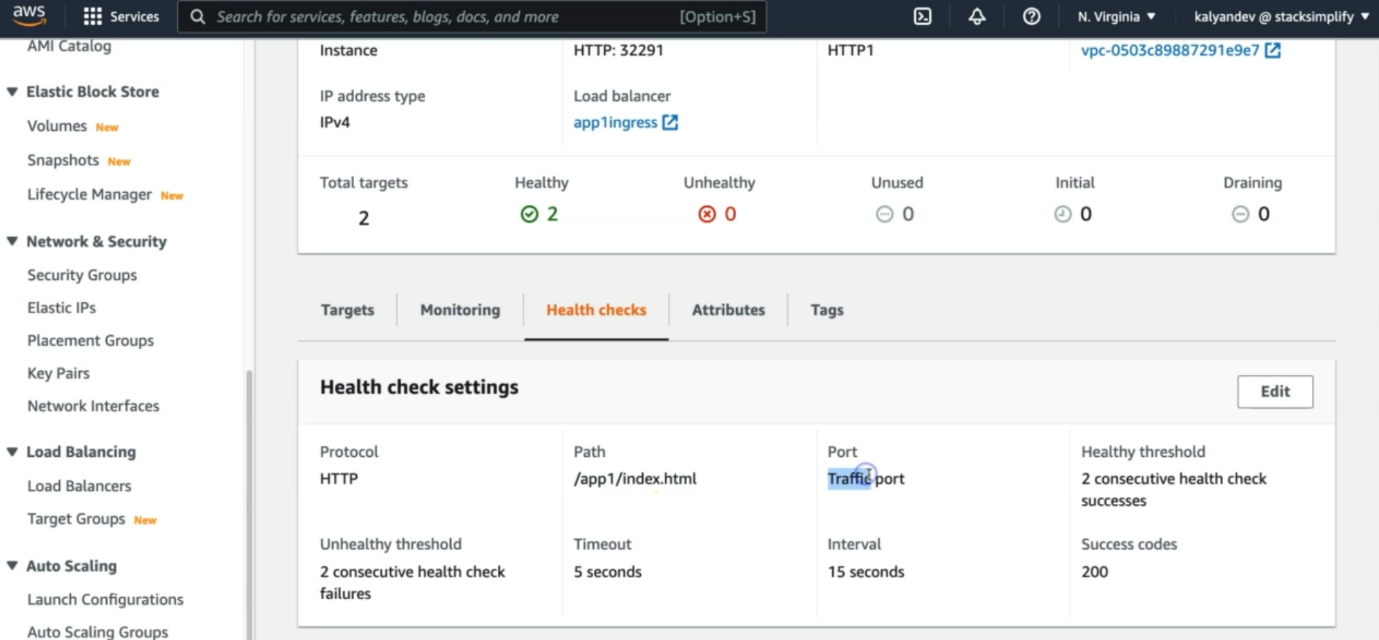


--- both instances registered successfully here. We have total 2 nodes and the both nodes are register here successfully. Whether the pod is present in which node, it does not care, why because your node port service is spread across two nodes. This means that if you have pod in which node is not a consider here.

--- **note** – in your kubernetes cluster if you have 50 nodes then all the 50 nodes will be added here.

--- this is about the traffic instance mode.

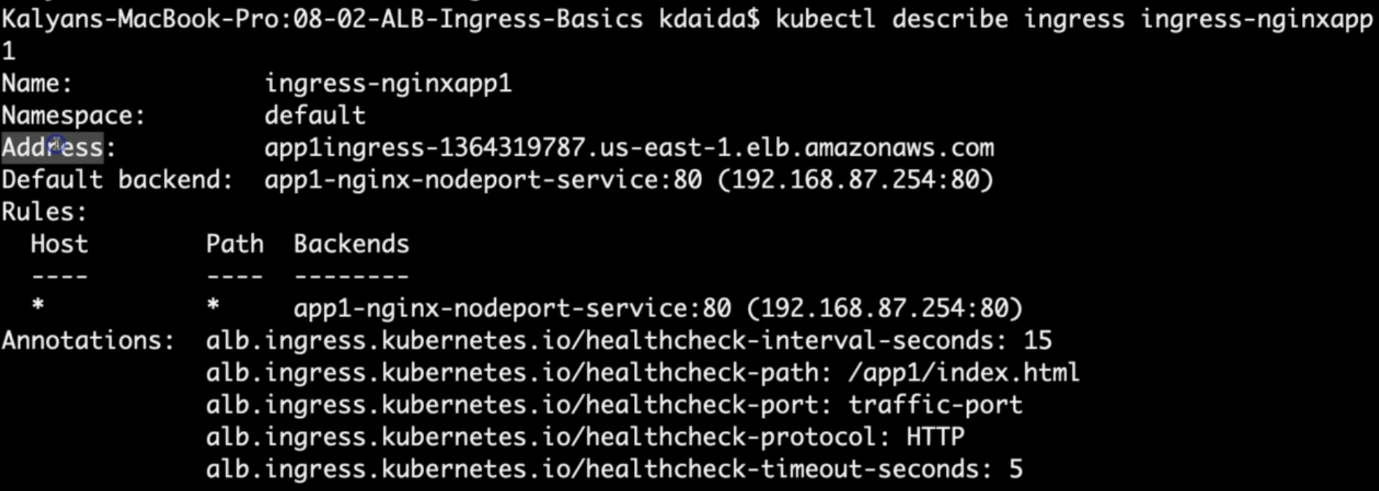
--- go to the health check option to see, what setting we provided.

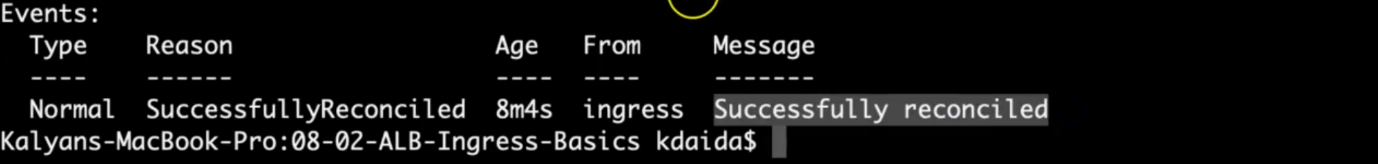


--- protocol is HTTP, path is /app1/index.html, port is traffic port...etc. all these things we mentioned in the default backend manifest.

**# Describe Ingress Controller**

--- **kubectl describe ingress ingress-nginxapp1**





--- note – you can see that there is no error events.

**Observation:**

1. Review Default Backend and Rules

**# List Services**

--- kubectl get svc

**# Verify Application Load Balancer using**

Goto AWS Mgmt Console -> Services -> EC2 -> Load Balancers

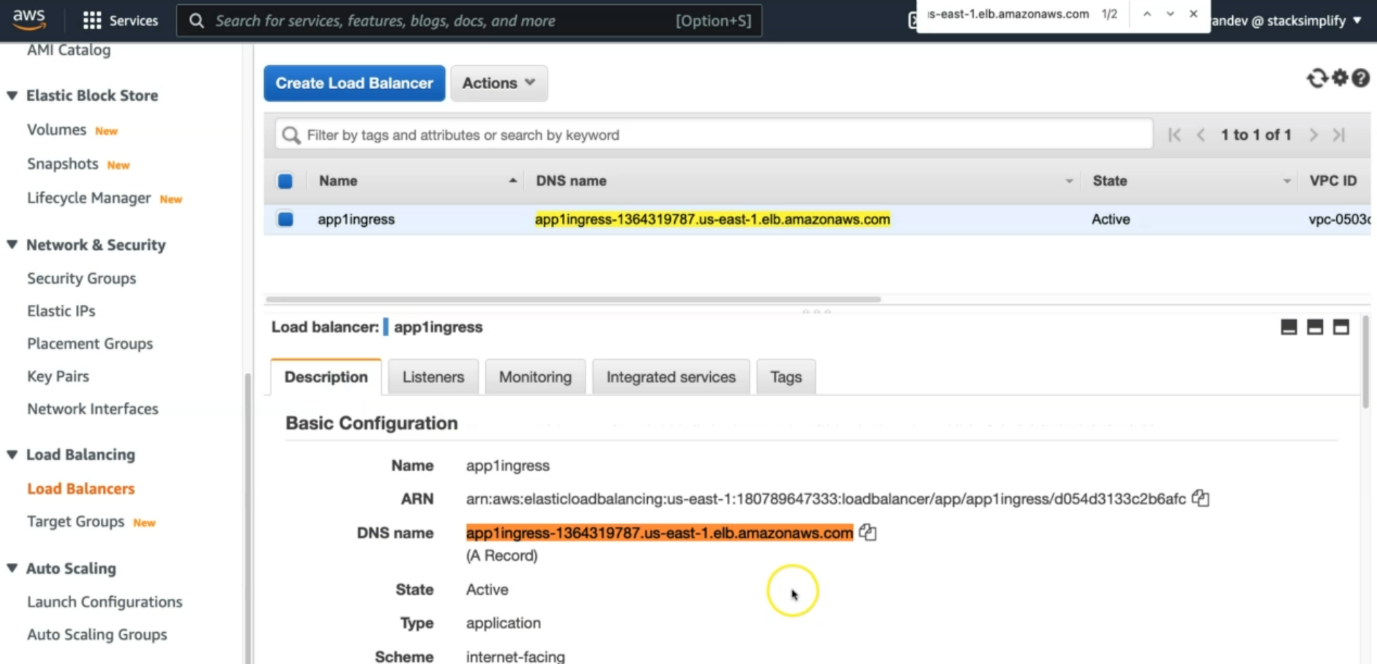
1. Verify Listeners and Rules inside a listener

2. Verify Target Groups

**# Access App using Browser**

--- kubectl get ingress

--- http://<ALB-DNS-URL>



--- the highlighted one is application load balancer DNS.

--- **http://<ALB-DNS-URL>/app1/index.html**

or

--- **http://<INGRESS-ADDRESS-FIELD>**

--- **http://<INGRESS-ADDRESS-FIELD>/app1/index.html**

**# Sample from my environment (for reference only)**

--- **http://app1ingress-154912460.us-east-1.elb.amazonaws.com**

--- **http://app1ingress-154912460.us-east-1.elb.amazonaws.com/app1/index.html**

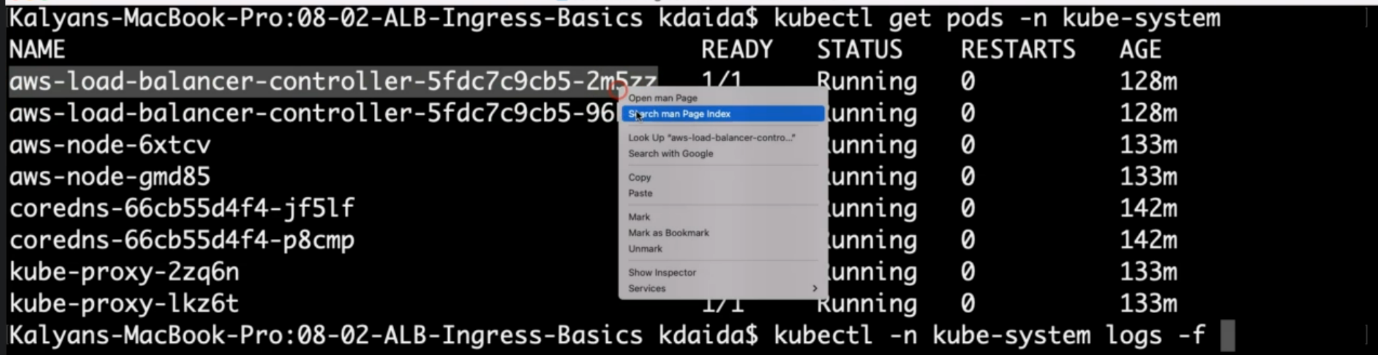
**# Verify AWS Load Balancer Controller logs**

--- **kubectl get po -n kube-system**

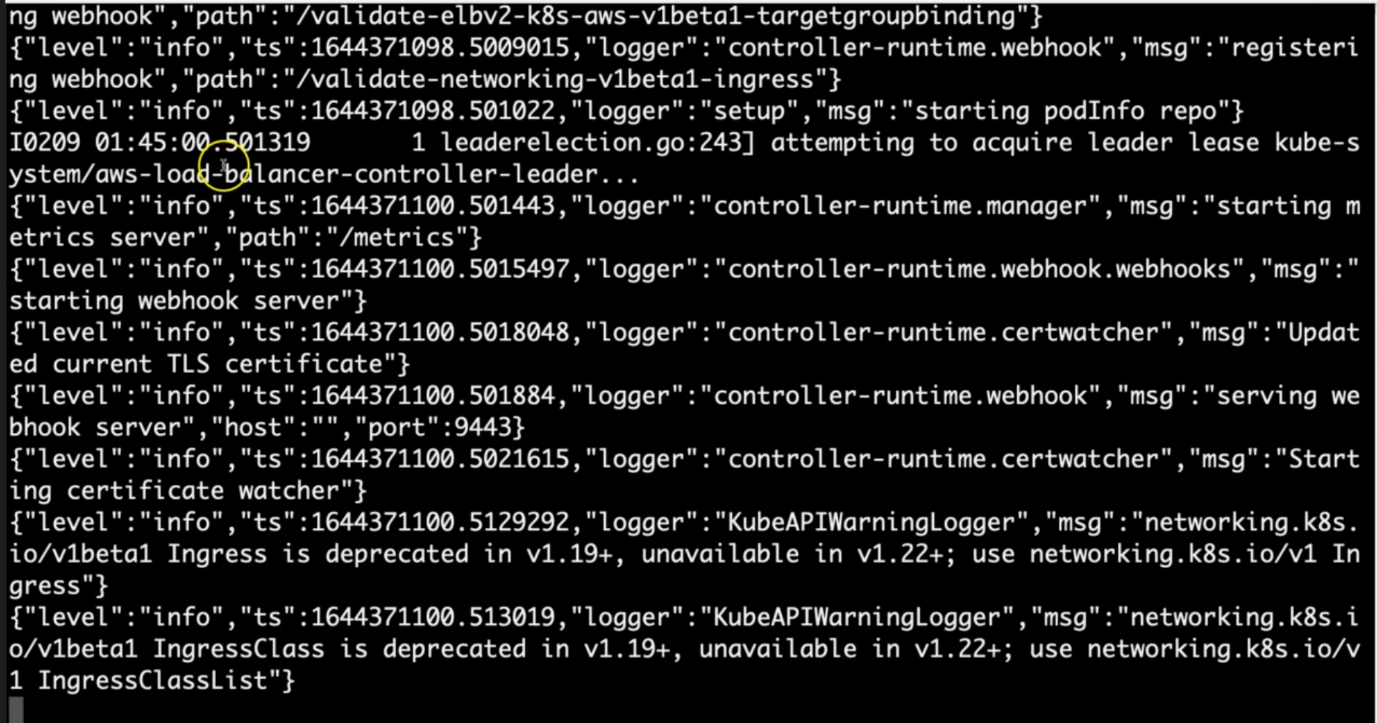
**## POD1 Logs:**

--- **kubectl -n kube-system logs -f <POD1-NAME>**

--- **kubectl -n kube-system logs -f aws-load-balancer-controller-65b4f64d6c-h2vh4**



--- **NOTE** – if your ingress is not working then you can check the logs of aws load balancer related pod. There you will find any error.



--- note – here, you will find any errors if your ingress load balancer is not working.

**##POD2 Logs:**

--- **kubectl -n kube-system logs -f <POD2-NAME>**

--- **kubectl -n kube-system logs -f aws-load-balancer-controller-65b4f64d6c-t7qqb**

**Clean Up**

**# Delete Kubernetes Resources**

--- **kubectl delete -f 01-kube-manifests-default-backend/**