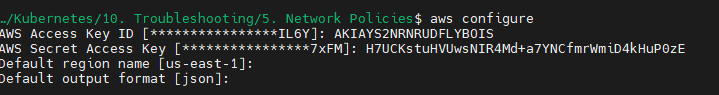
Login to Sandbox and create AWS EKS cluster:

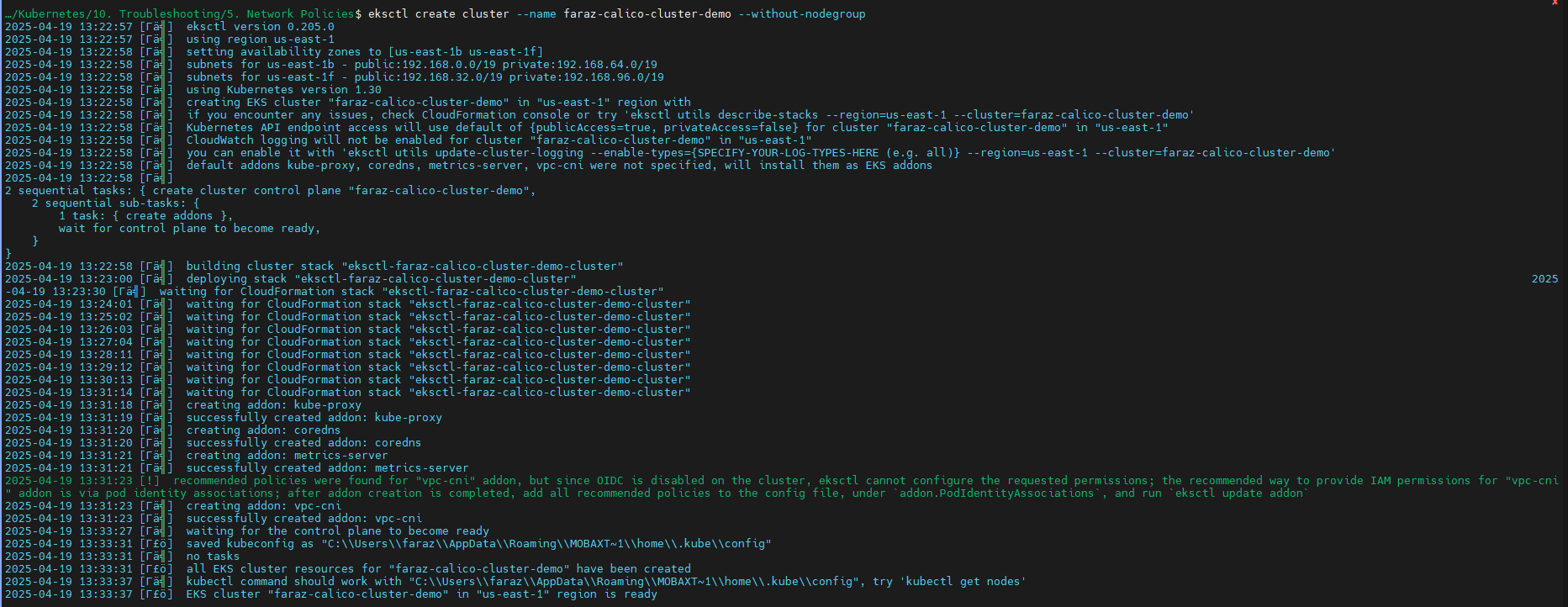
* aws configure



* Use your own Access keys (I have used sandbox access keys which change once session ends)

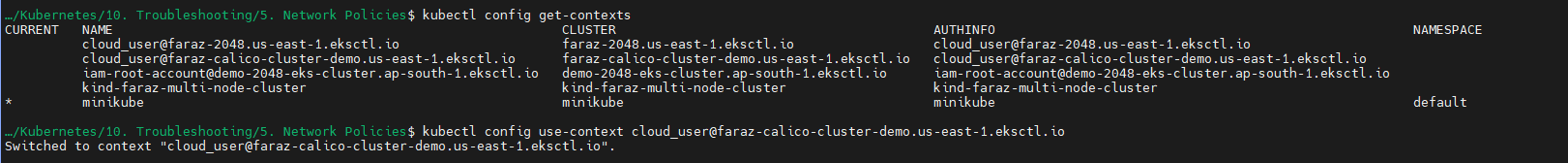
Create your EKS cluster:

* You can refer: <https://docs.tigera.io/calico/latest/getting-started/kubernetes/managed-public-cloud/eks#install-eks-with-calico-networking>

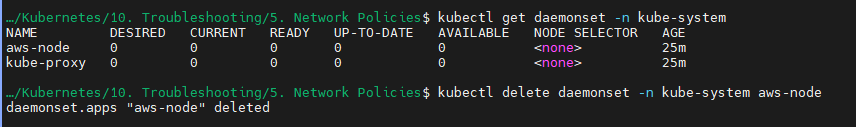


Switch context if you are in minikube/any other config

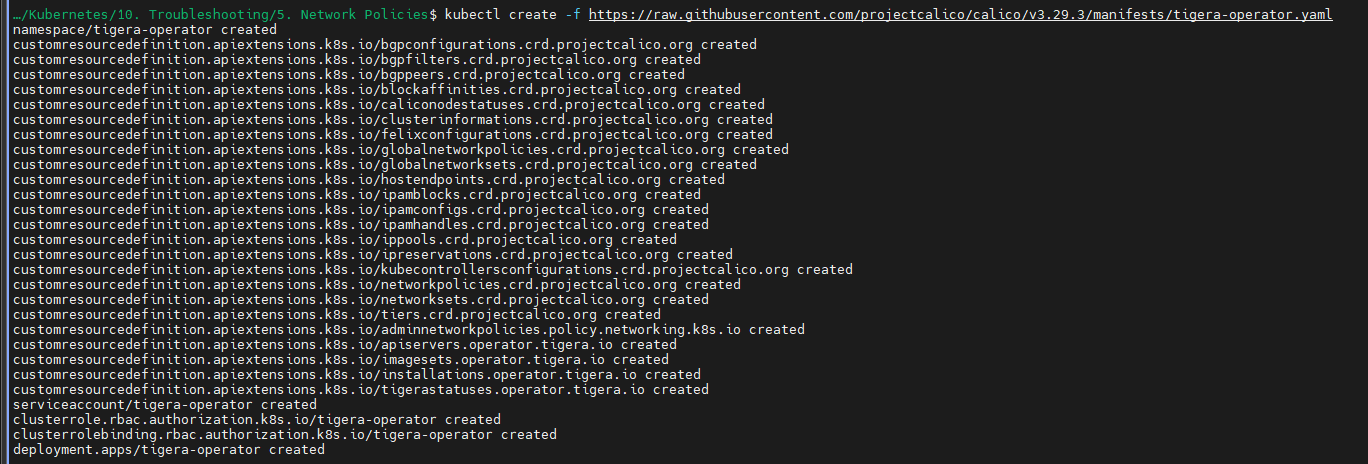
* kubectl config get-contexts
* kubectl config use-context cloud\_user@faraz-calico-cluster-demo.us-east-1.eksctl.io



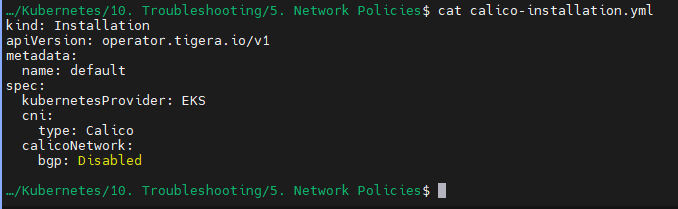
Delete the aws-node daemon set as we are using calico:



Now install calico operator:



* vim calico-installation.yml

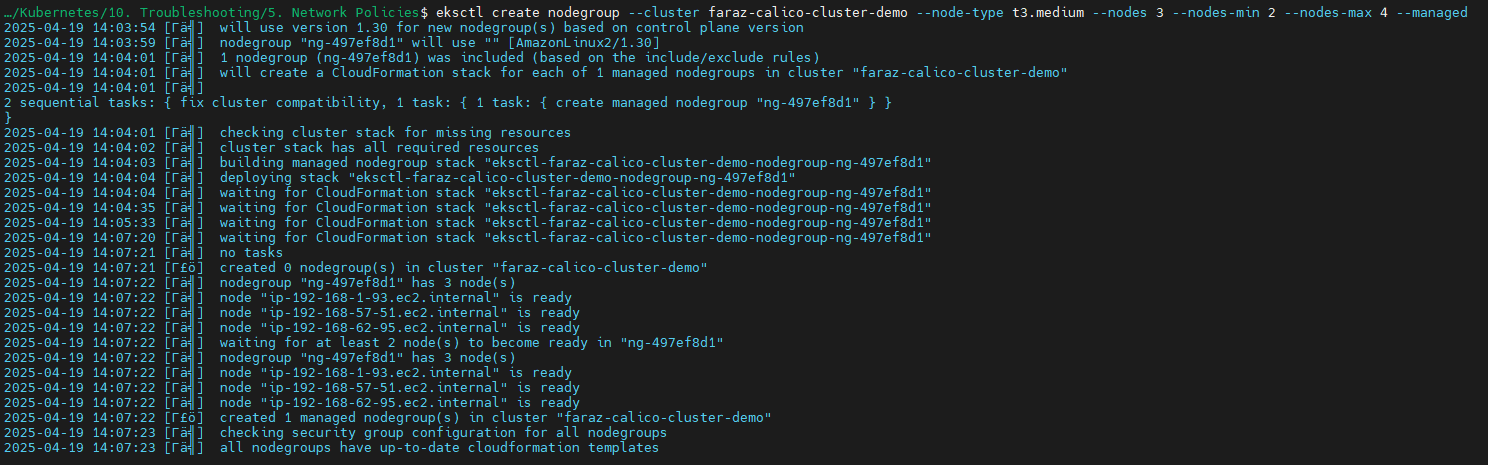


Configure the calico installation



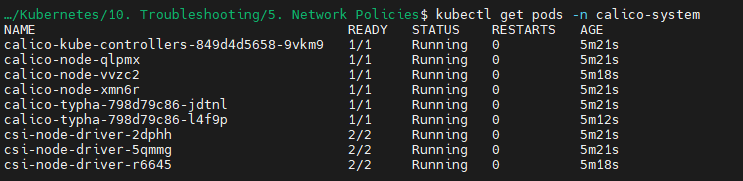
Add nodes to the cluster:

* eksctl create nodegroup --cluster faraz-calico-cluster-demo --node-type t3.medium --nodes 3 --nodes-min 2 --nodes-max 4 --managed



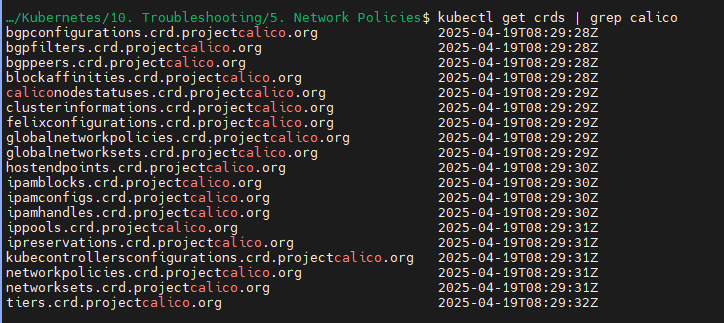
Confirm all your pods are running in calico-system

* kubectl get pods -n calico-system

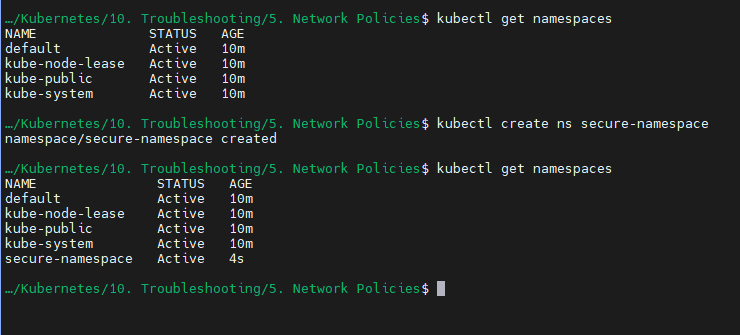


Also your custom resource definition is installed:

* kubectl get crds | grep calico

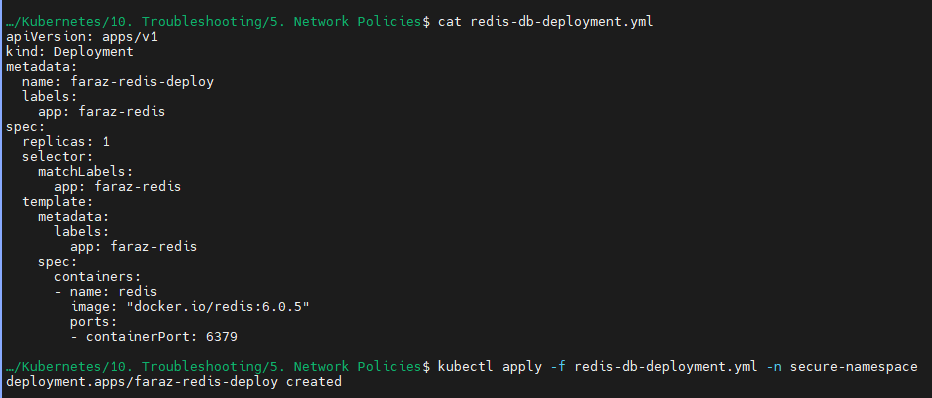


Now create a secure-namespace:

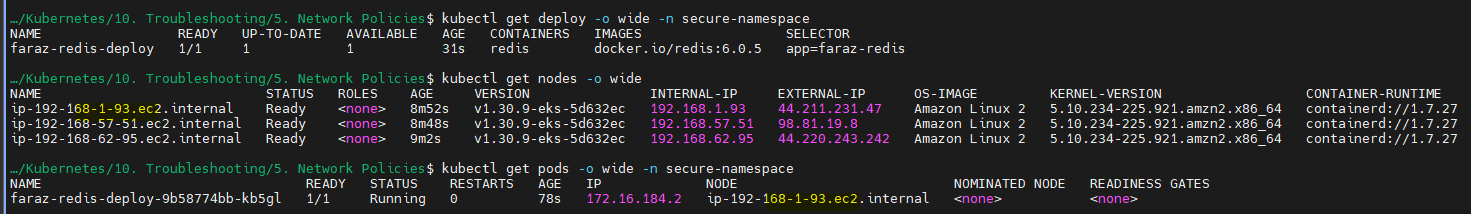


Apply your redis deployment in secure namespace:

* kubectl apply -f redis-db-deployment.yml -n secure-namespace



Validate your deployment:

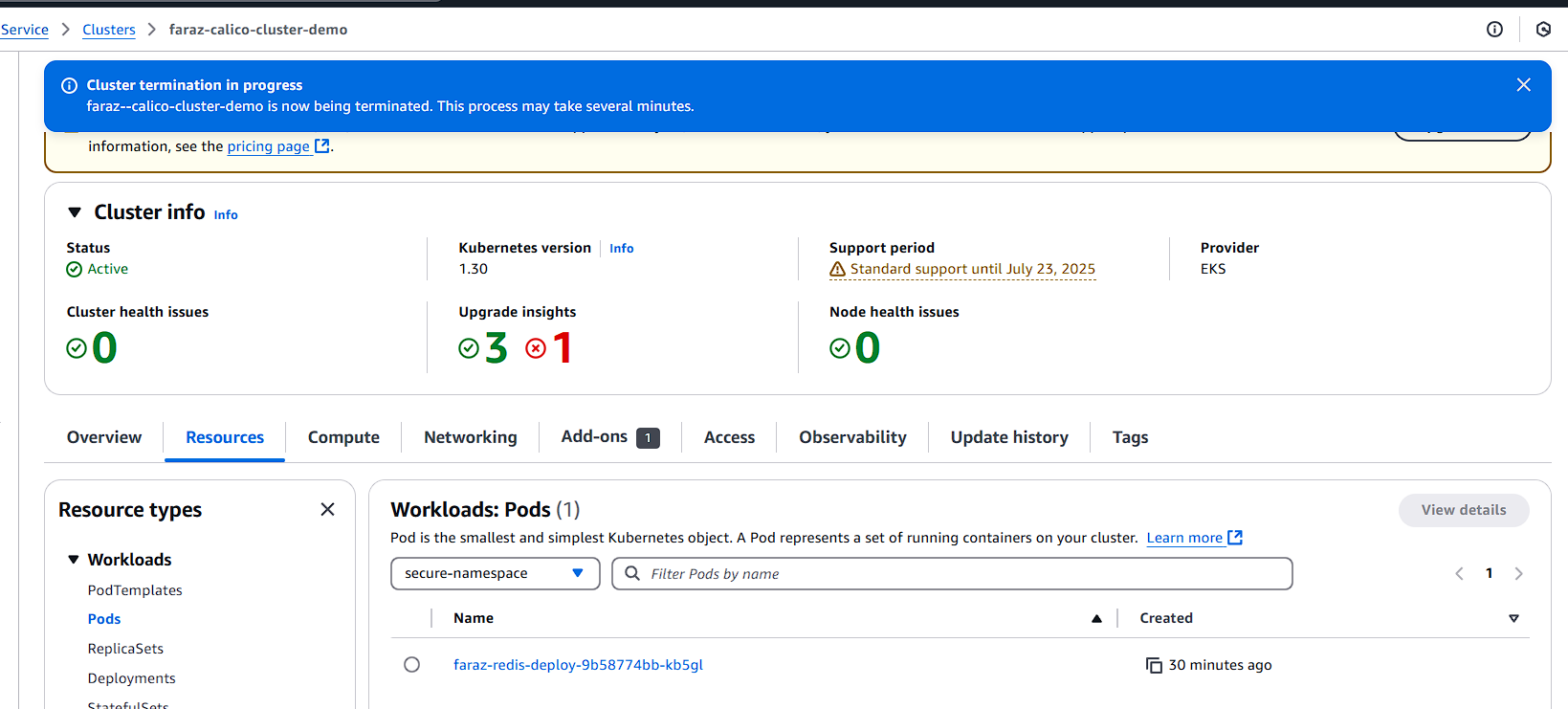


Validate your redis-cli:

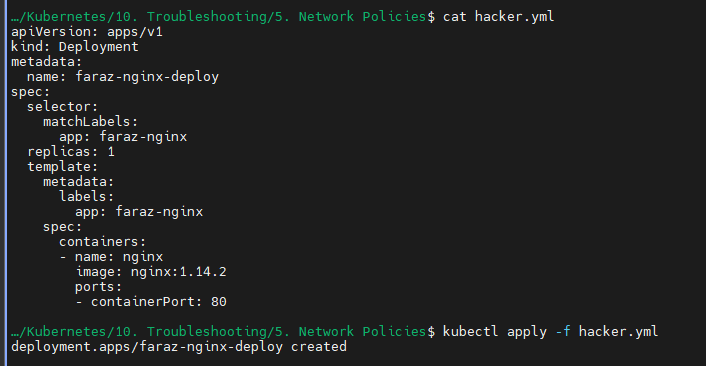
* kubectl exec -it faraz-redis-deploy-9b58774bb-kb5gl -n secure-namespace -- redis-cli



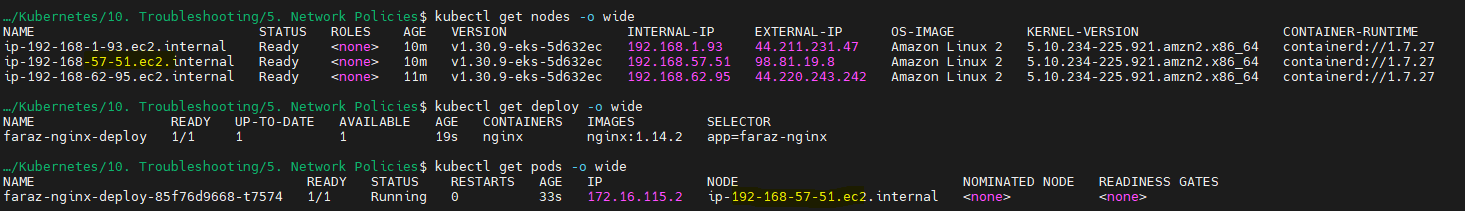
You can also validate on your AWS console:



Now we will create hacker.yml in default namespace:



Validate your deployment:



Now login to your pod in default namespace:

* kubectl exec -it faraz-nginx-deploy-85f76d9668-t7574 -- /bin/bash

Now install redis on this machine by following few of the steps:

<https://redis.io/docs/latest/operate/oss_and_stack/install/archive/install-stack/linux/>

Now

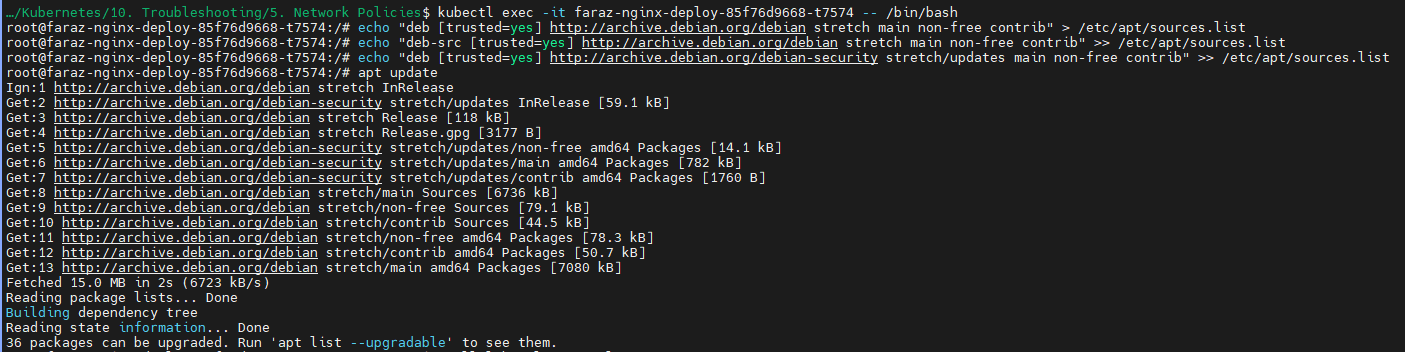
* apt install

[echo "deb [trusted=yes] http://archive.debian.org/debian stretch main non-free contrib" > /etc/apt/sources.list](https://superuser.com/questions/1782162/how-can-i-fix-these-apt-repository-errors)

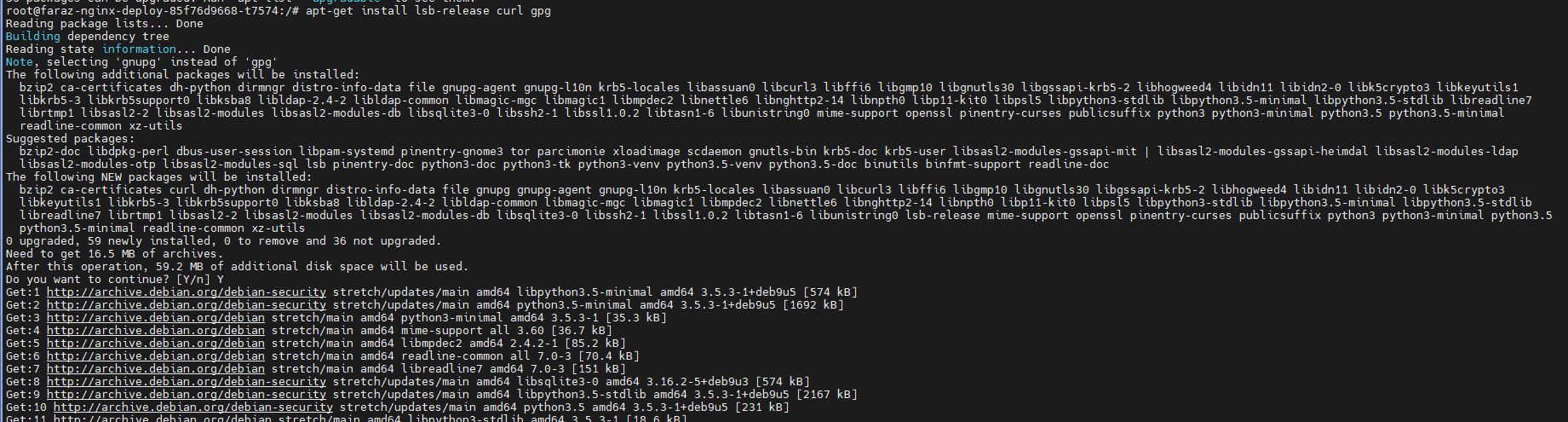
[echo "deb-src [trusted=yes] http://archive.debian.org/debian stretch main non-free contrib" >> /etc/apt/sources.list](https://superuser.com/questions/1782162/how-can-i-fix-these-apt-repository-errors)

[echo "deb [trusted=yes] http://archive.debian.org/debian-security stretch/updates main non-free contrib" >> /etc/apt/sources.list](https://superuser.com/questions/1782162/how-can-i-fix-these-apt-repository-errors)

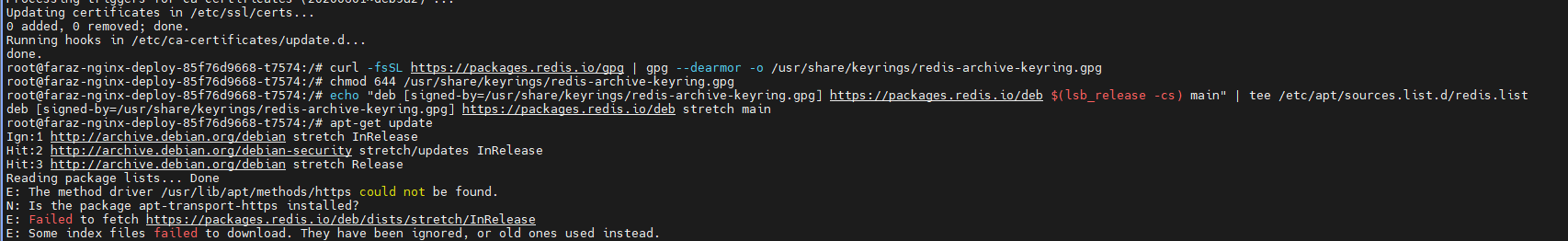
* apt update



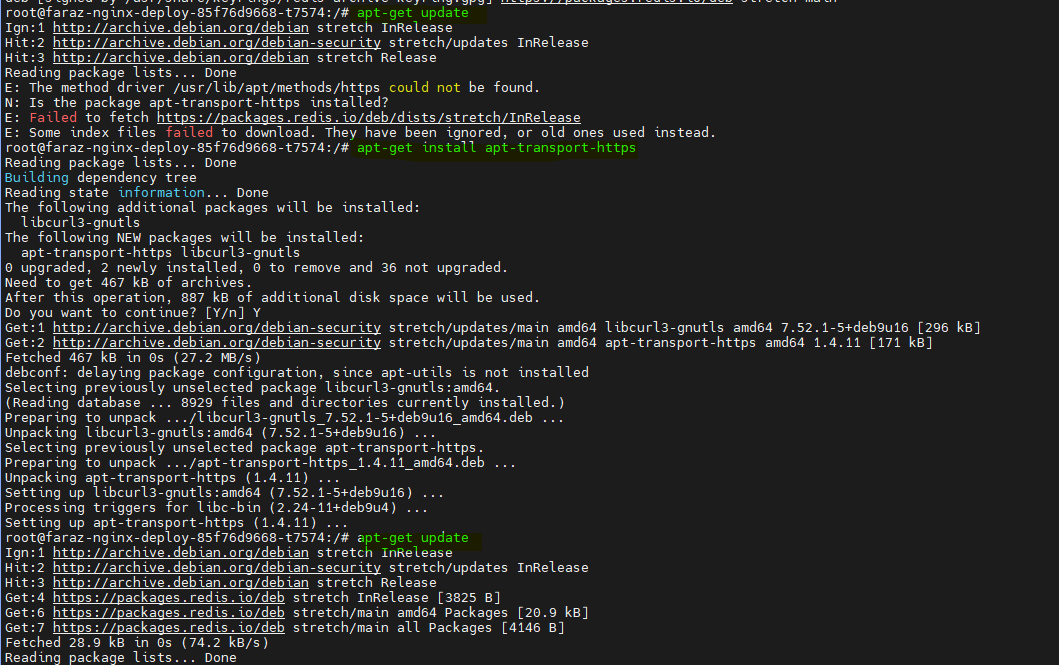
* apt-get install lsb-release curl gpg



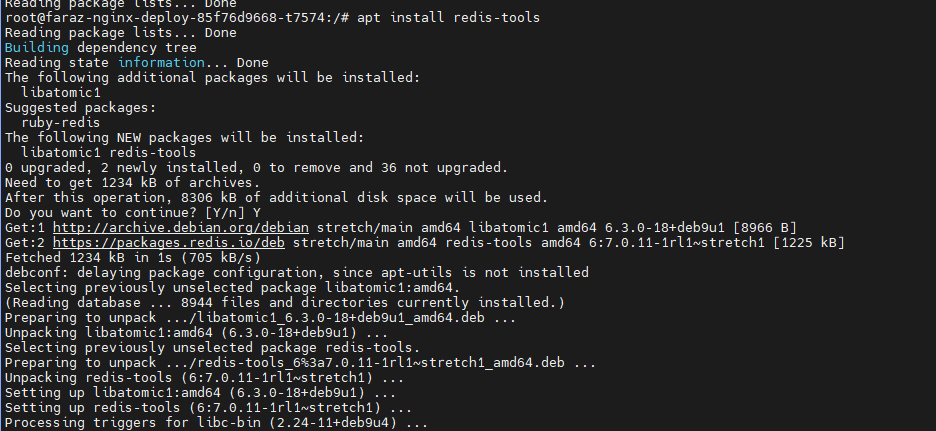
* curl -fsSL https://packages.redis.io/gpg | gpg --dearmor -o /usr/share/keyrings/redis-archive-keyring.gpg
* chmod 644 /usr/share/keyrings/redis-archive-keyring.gpg
* echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg] https://packages.redis.io/deb $(lsb\_release -cs) main" | tee /etc/apt/sources.list.d/redis.list



* apt-get install apt-transport-https
* apt-get update



* **apt install redis-tools**



Now if you see:

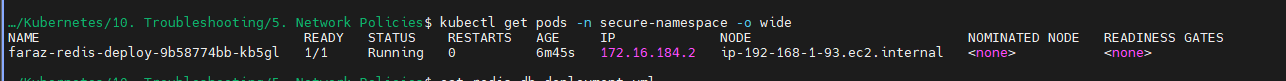
And from my default name space if I try to connect the Pod in secure namespace:

* redis-cli -h 172.16.184.2



I am able to connect from default namespace to secure namespace. Hence, this is a security issue.

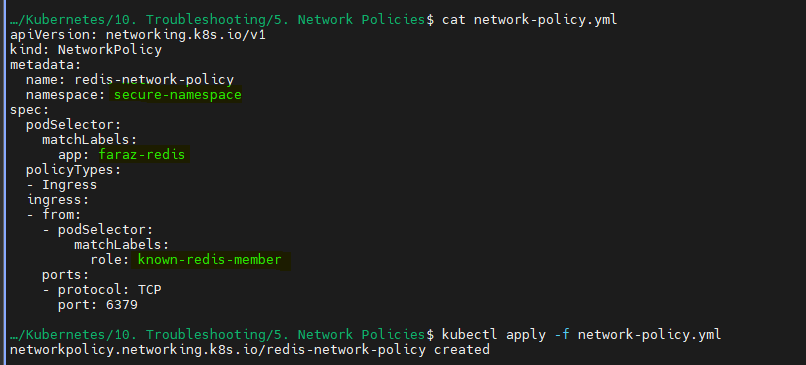
Note: The redis pod in secure name space has IP: 172.16.184.2

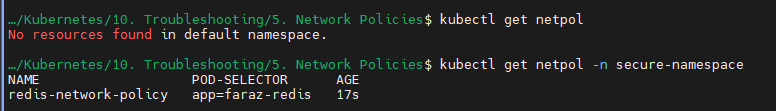


To solve this issue, we may use kubernetes network-policies:

<https://kubernetes.io/docs/concepts/services-networking/network-policies/>

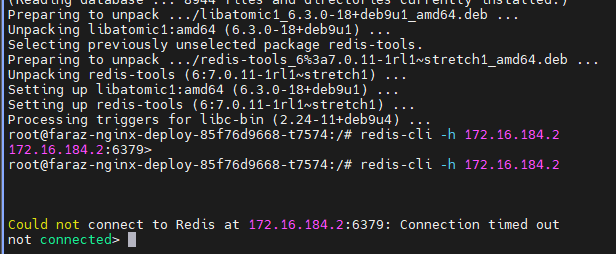
Make your network policy:





If you now try to connect from pod in the default namespace to the pod secure namespace:

You will get the following:



⚠️ Without Calico?

If you define a NetworkPolicy in EKS:

* With Fargate? It gets ignored.
* With EC2, but no Calico? Still ignored.
* With EC2 + Calico? ✅ Enforced correctly.

Calico is what actually enforces Kubernetes NetworkPolicies on your nodes.

Kubernetes supports defining NetworkPolicy objects, but the K8s API server doesn’t enforce them — it’s up to the CNI plugin to implement and apply the rules.