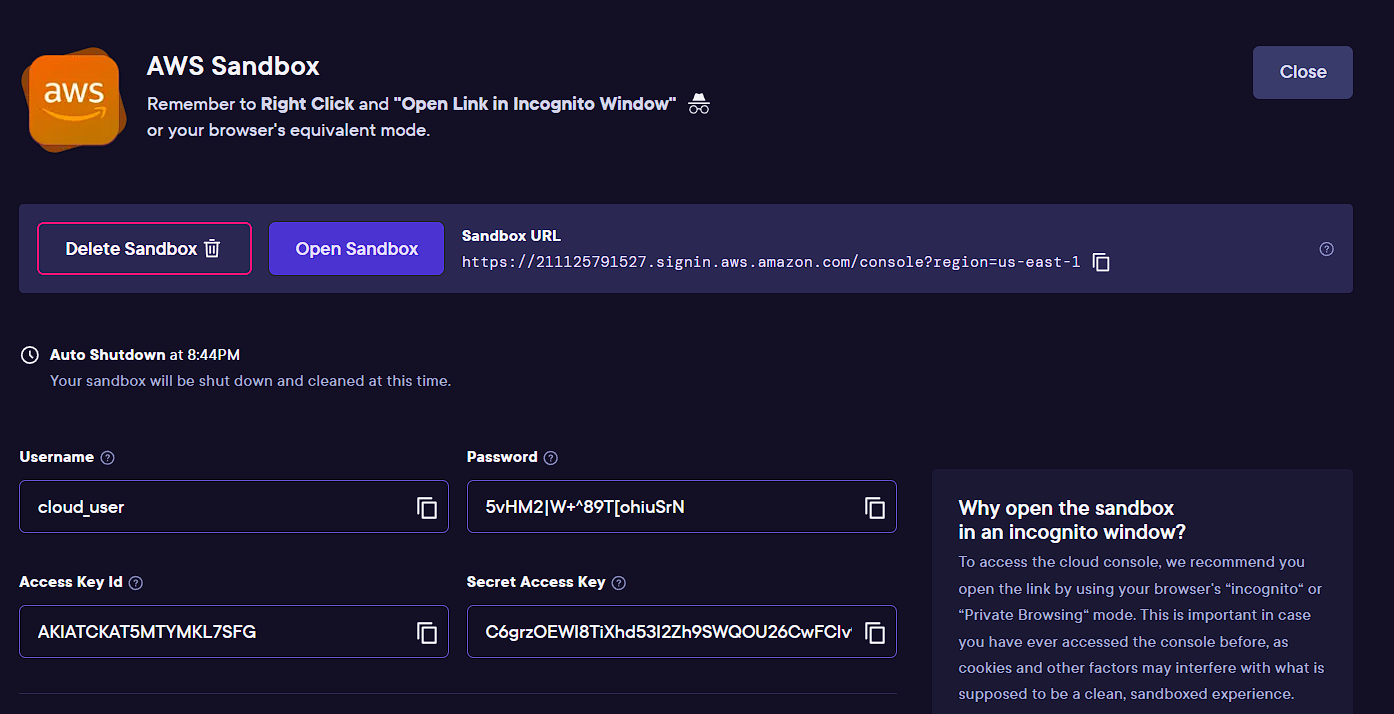
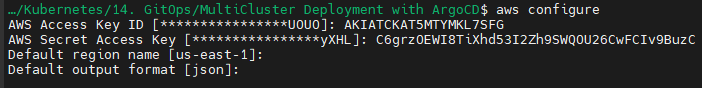
Creating AWS Sandbox:



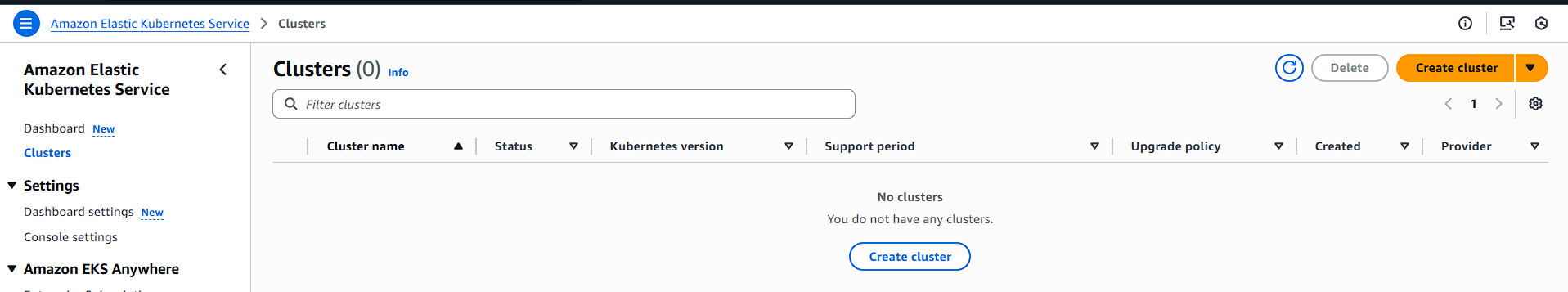
Login to AWS

Also configure aws cli using:

* aws configure

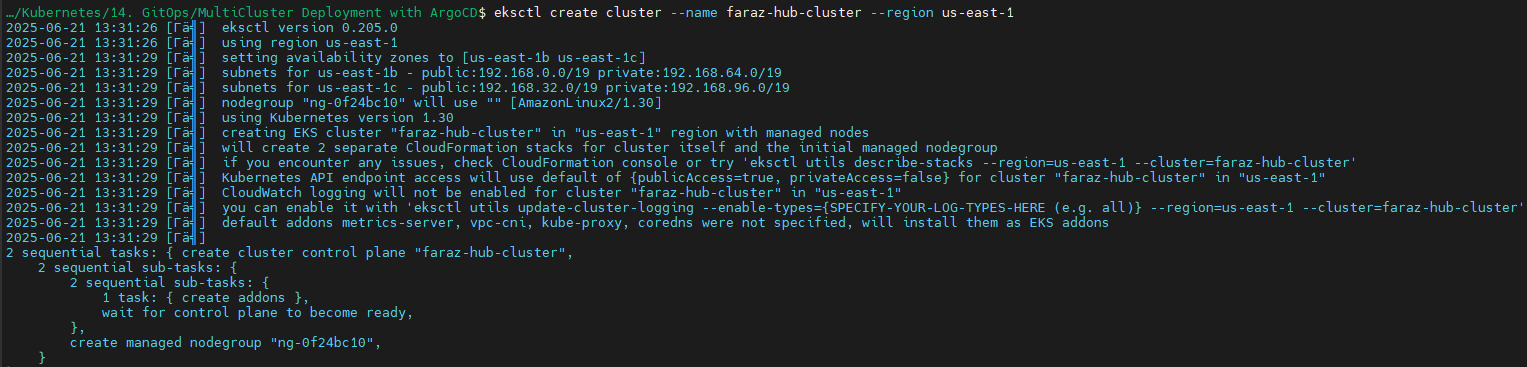


Currently we do not have any cluster:

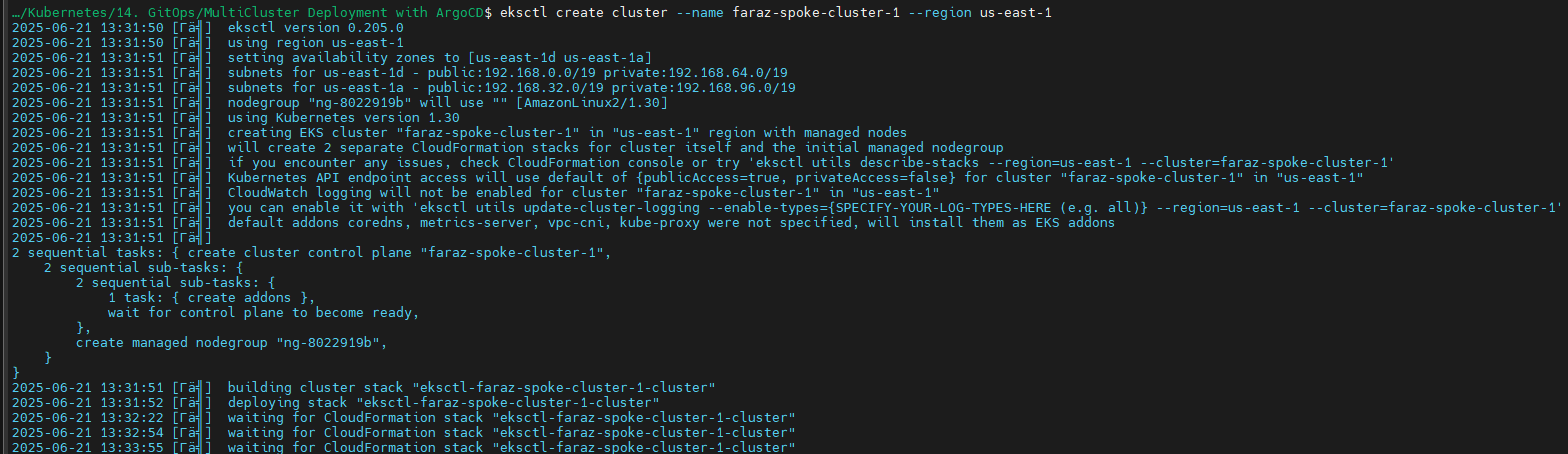


Let’s create our clusters:

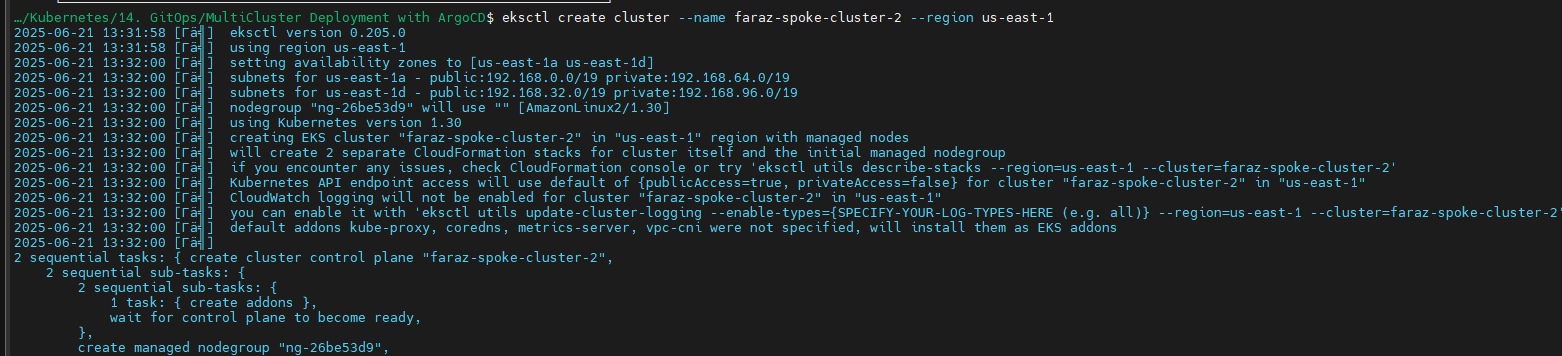
* Since we are demonstrating hub-spoke modes, our architecture will look like:
  + Hub
    - eksctl create cluster --name faraz-hub-cluster --region us-east-1



* + Spoke 1
    - eksctl create cluster --name faraz-spoke-cluster-1 --region us-east-1

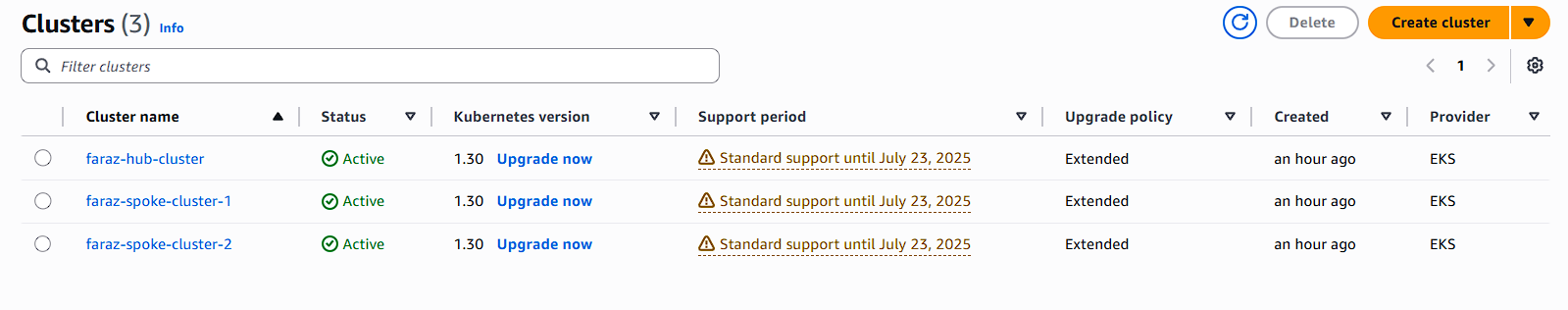


* + Spoke 2
    - eksctl create cluster --name faraz-spoke-cluster-2 --region us-east-1



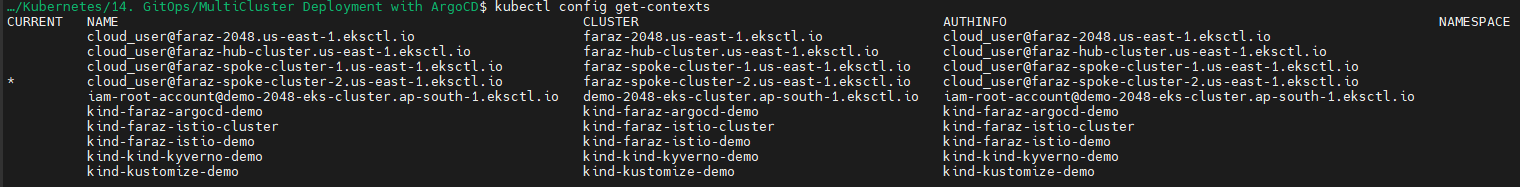
Wait for 10-15 mins for the CloudFormation to create your clusters:

Once created, verify your clusters:

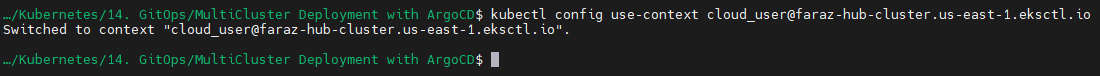


Switch the context to hub:

* kubectl config get-contexts



* kubectl config use-context cloud\_user@faraz-hub-cluster.us-east-1.eksctl.io



Verify your current context:

* kubectl config current-context

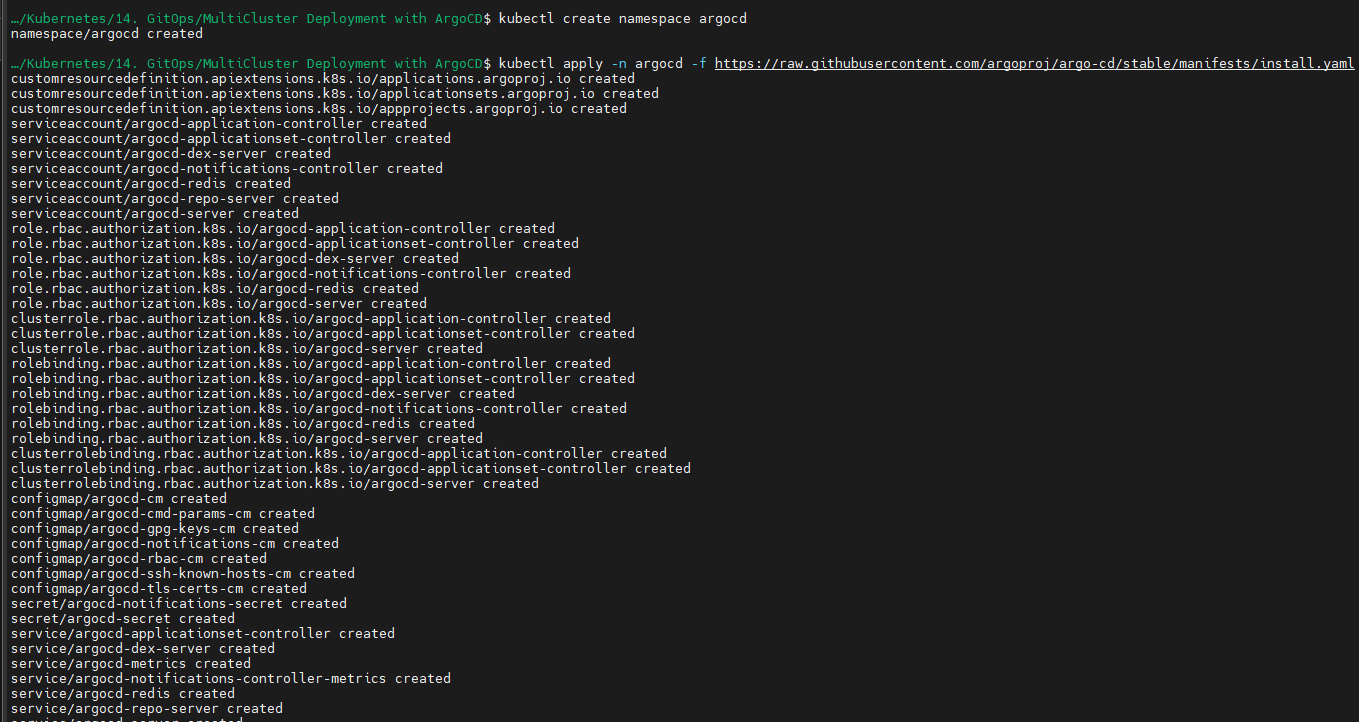


Now We will install ArgoCD in our hub cluster:

Refer: <https://argo-cd.readthedocs.io/en/stable/getting_started/>

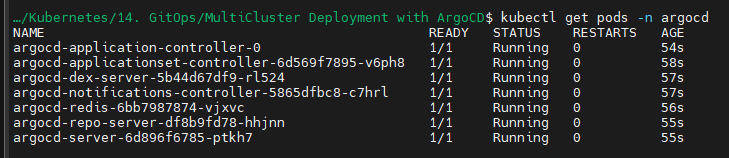
Install ArgoCD in argocd namespace:

* kubectl create namespace argocd
* kubectl apply -n argocd -f <https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml>



Verify your resource creation:

* kubectl get pods -n argocd



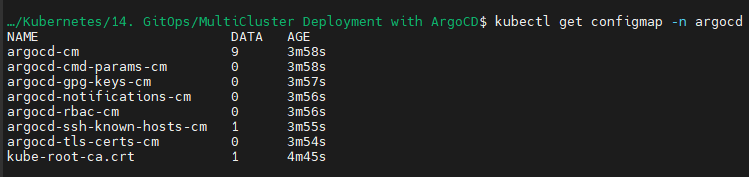
Now we do not want to use TLS/HTTPS and only use HTTP we need to add:

<https://github.com/argoproj/argo-cd/blob/master/docs/operator-manual/argocd-cmd-params-cm.yaml>

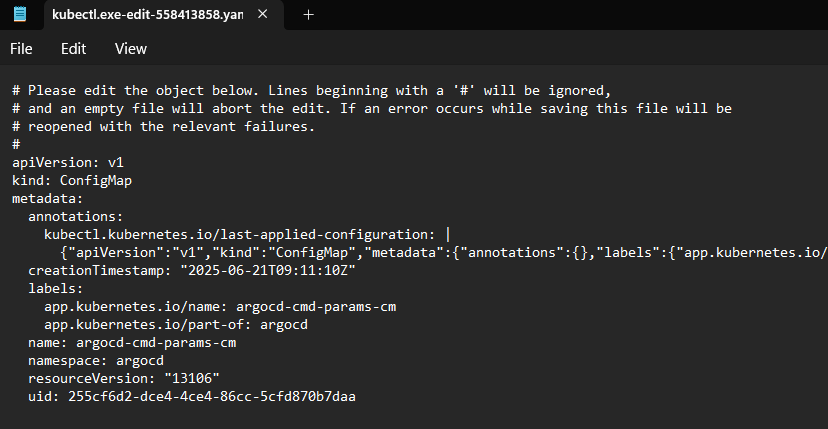


To do that, we will go to configmaps:

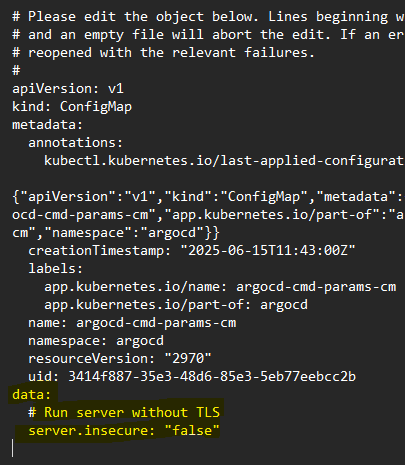
* kubectl get configmaps -n argocd
* kubectl edit cm argocd-cmd-params-cm -n argocd





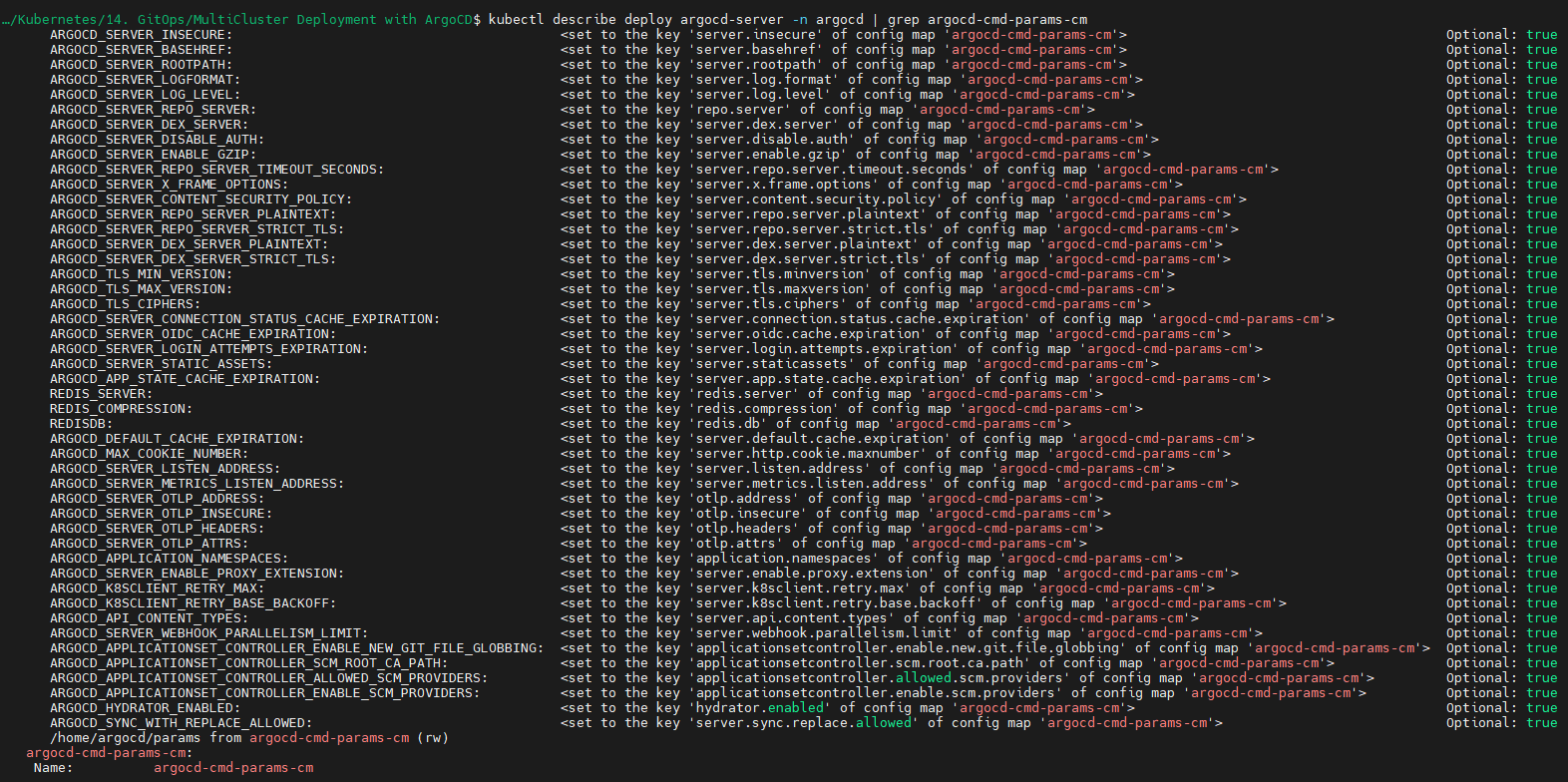


Add this:

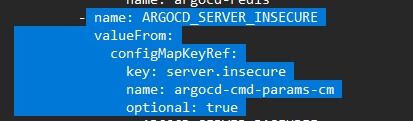


This change will enable our ArgoCD on http as the config map is mounted on argocd-server:

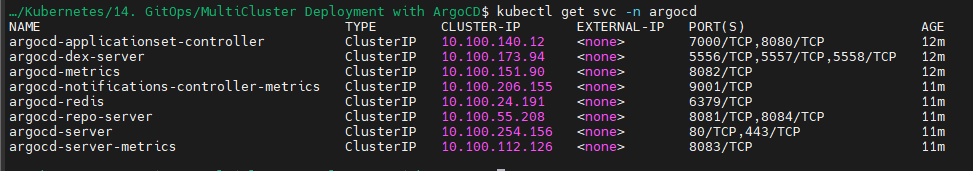
* kubectl describe deploy argocd-server -n argocd | grep argocd-cmd-params-cm



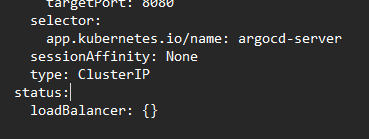
* kubectl edit deploy argocd-server -n argocd

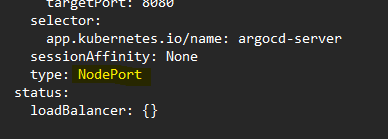


Now, we can either use ingress or we can edit service to NodePort mode:

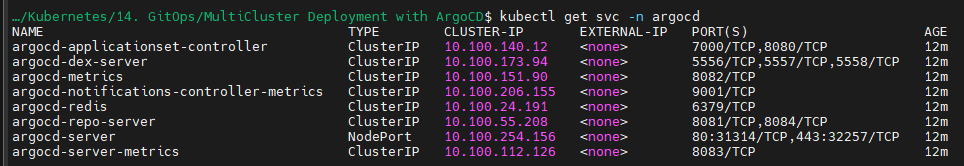


Change the type from cluster IP to Nodeport:

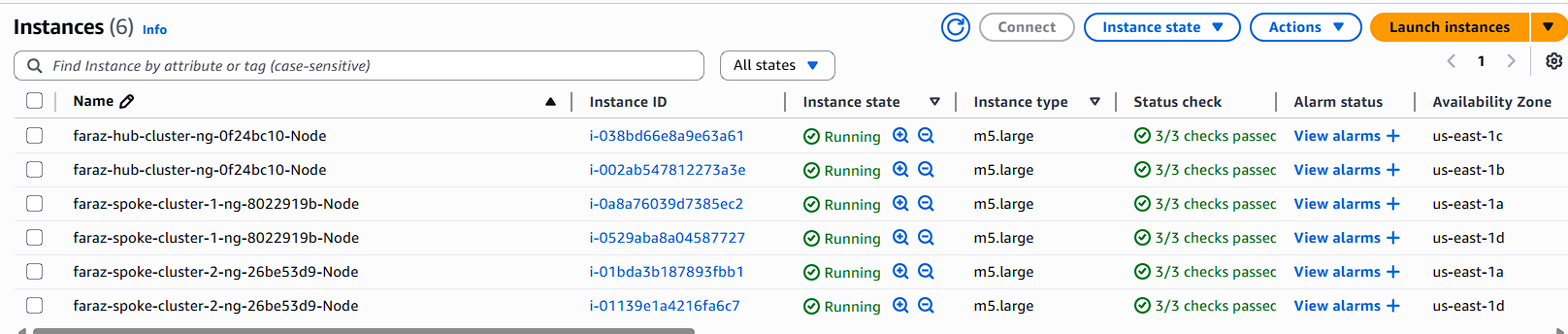




Your service will now be changed to NodePort mode:



Now go to our EC2, EKS is running on EC2 by default:



In your Amazon EKS cluster, you have two nodes. When you expose the ArgoCD server using a **NodePort**, it creates a port on each node that forwards traffic to the relevant pods running the ArgoCD service.

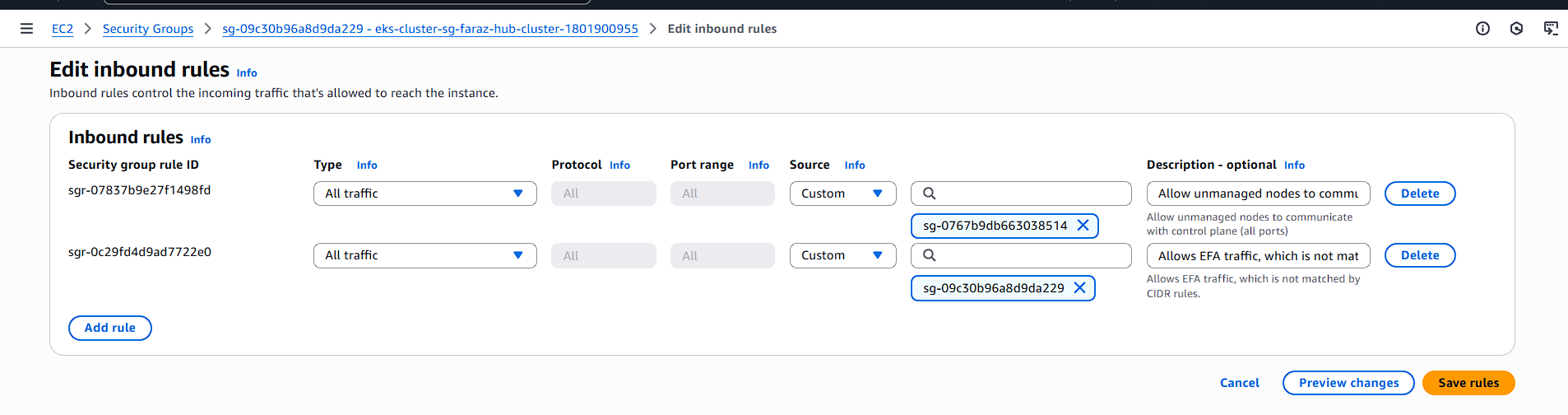
Now, let's say:

* The ArgoCD pod is running on **Node A**.
* You try to access the service via **Node B's** IP address and NodePort.

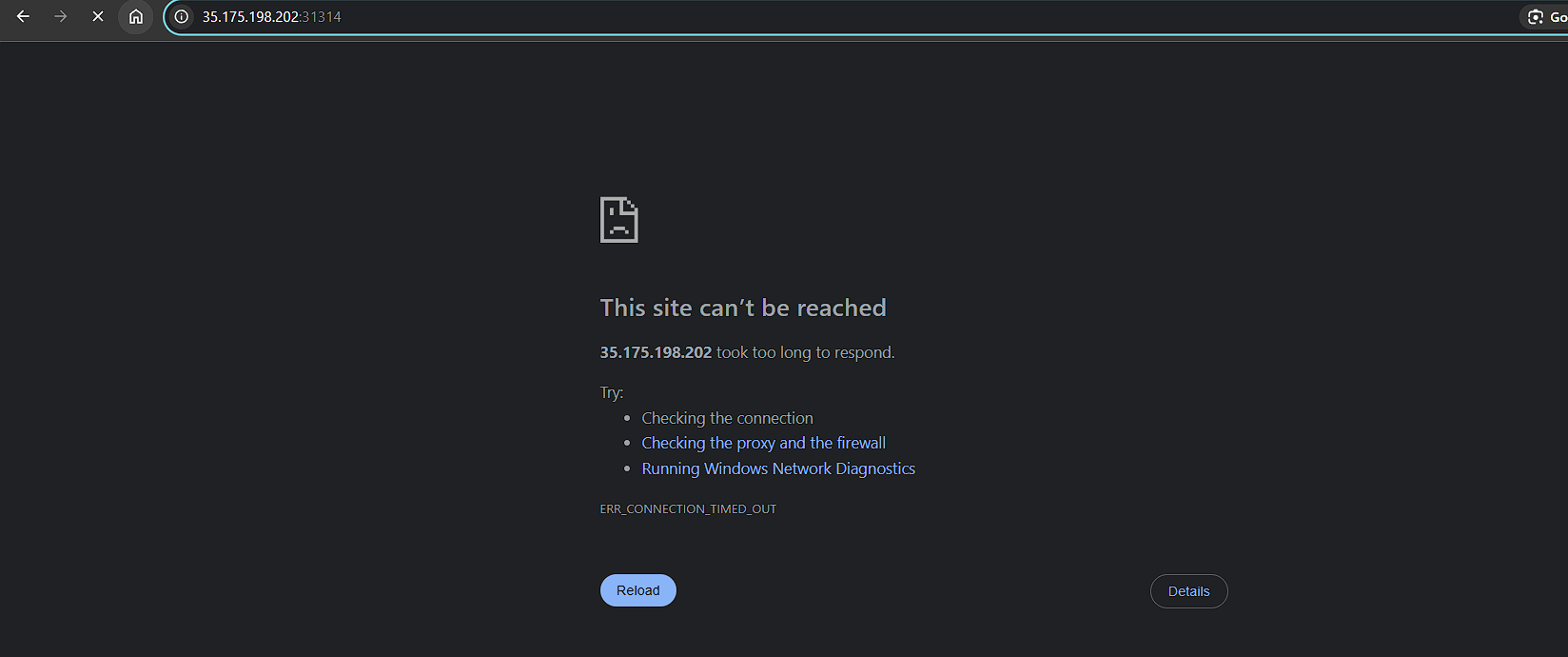
Even though the request is hitting **Node B**, Kubernetes' **Kube-Proxy** ensures that the request is properly routed to the ArgoCD pod running on **Node A**. Kube-Proxy manages this routing mechanism by maintaining the node-to-pod mapping and forwarding traffic efficiently.

This means that regardless of which node's IP you use, Kubernetes will ensure that your request reaches the correct pod, even if it's located on another node. This makes NodePort services highly resilient, especially in multi-node environments.

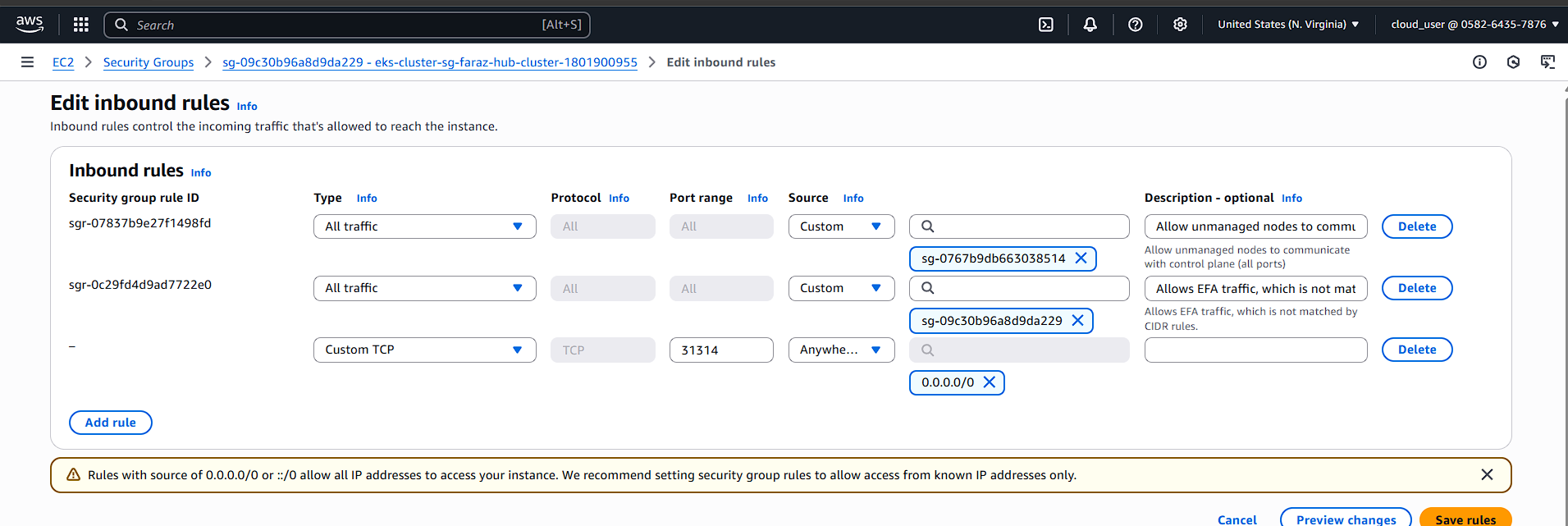
Current SG:



If we try to access, it will fail we need to configure security group:

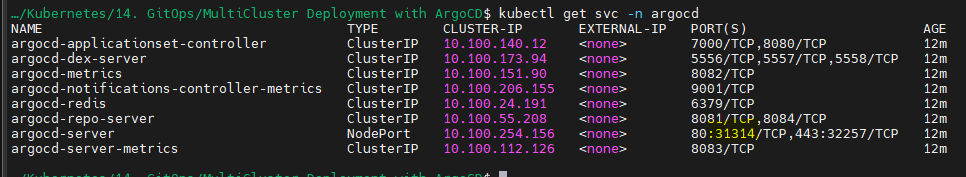


Add this to security group:



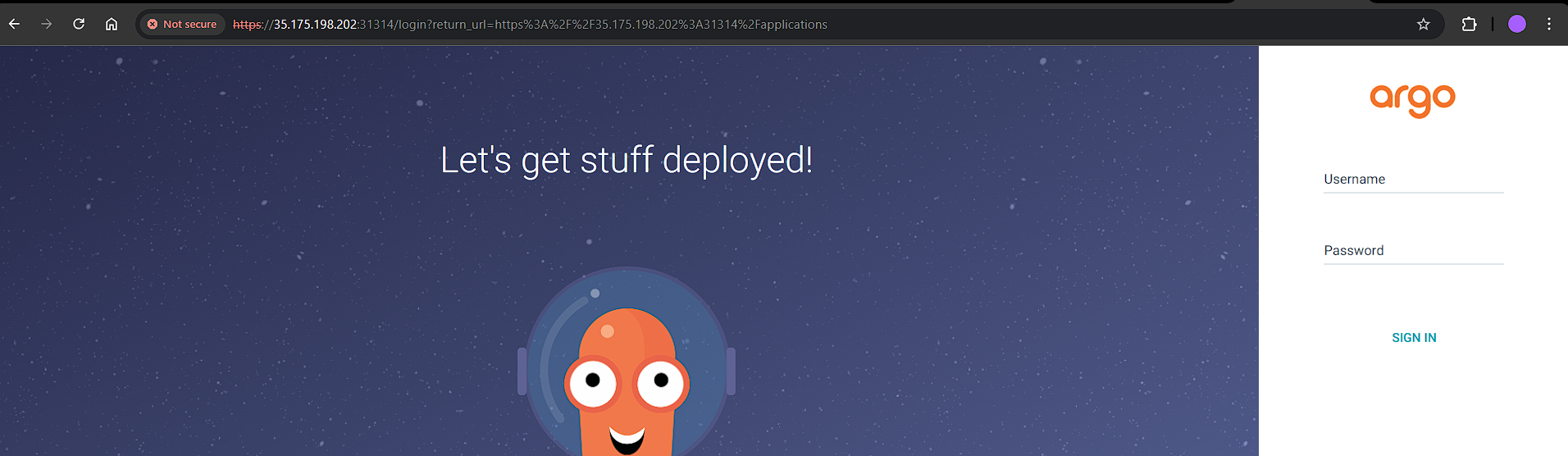
Open Port: 31314

As:



Now you will be able to access ArgoCD on https://<NodeIP/EC2 public IP>:NodePort/

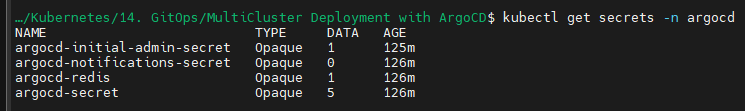
<https://35.175.198.202:31314/>:



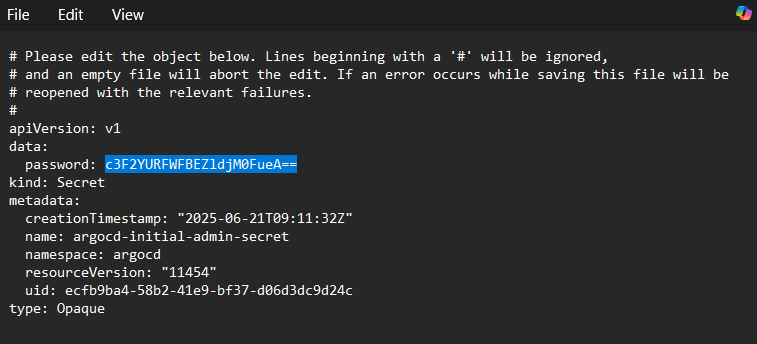
Now at this moment, we have installed, ArgoCD in Hub

Now we will get the secret:

* kubectl get secrets -n argocd



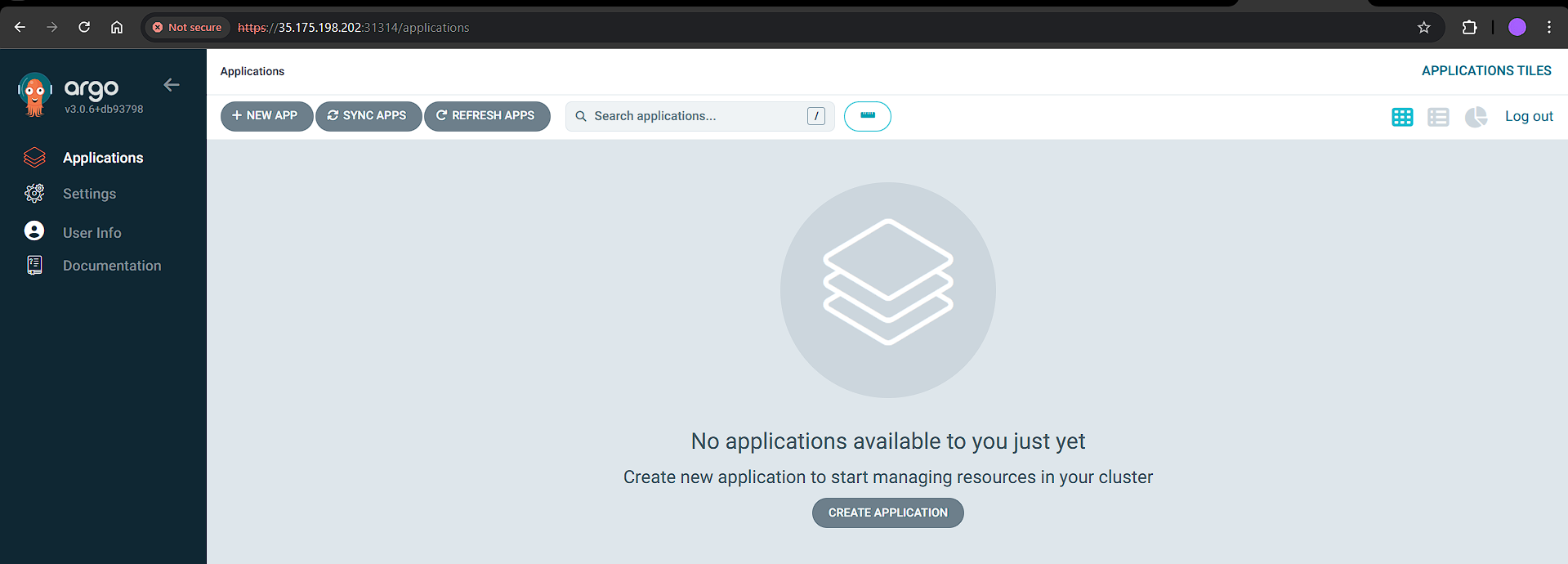
Open the initial admin secret:



Decode the password using base64:

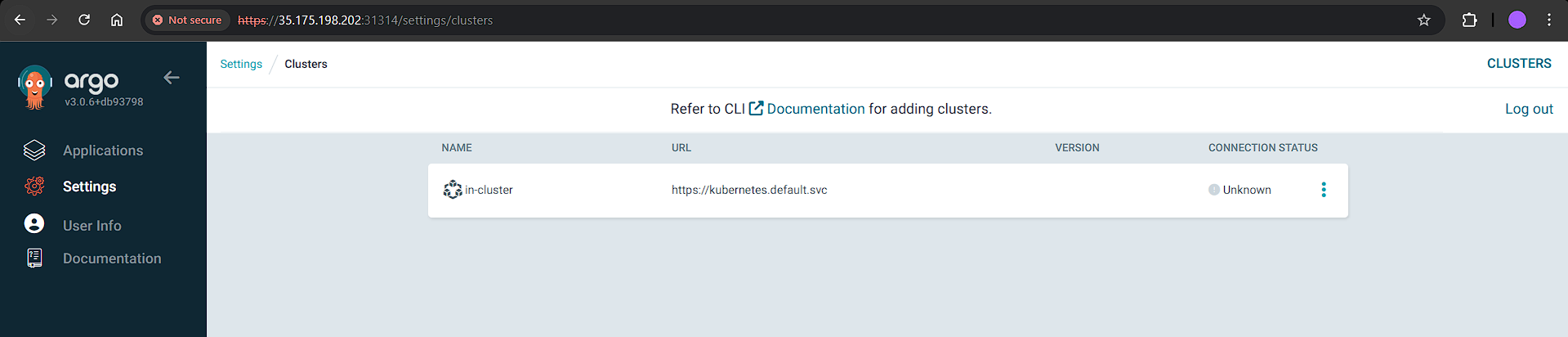


Login to your ArgoCD:

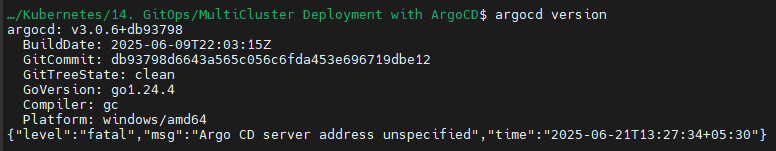


Go to Settings > Clusters:

* In-cluster means the same cluster, ArgoCD is capable enough to deploying applications on the same cluster

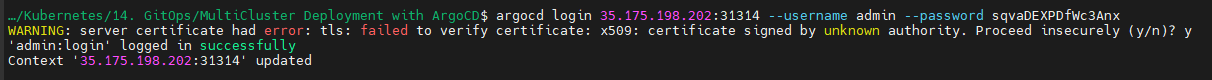


UI of ArgoCD doesn’t allow adding clusters. Hence, install ArgoCD CLI



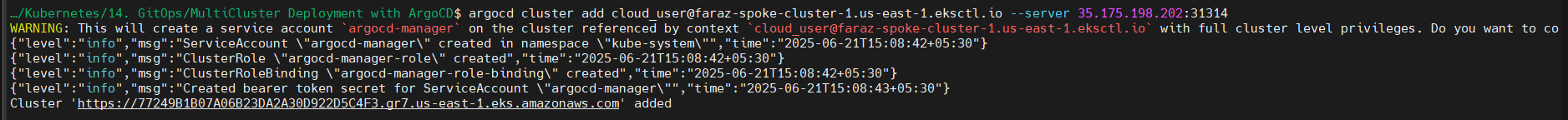
Login to your ArgoCD:

* argocd login 35.175.198.202:31314 --username admin --password sqvaDEXPDfWc3Anx

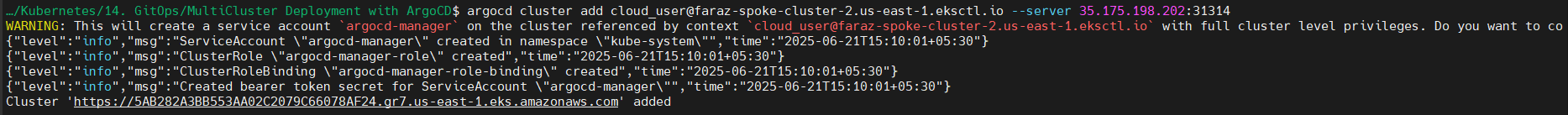


Now we will add our spoke clusters:

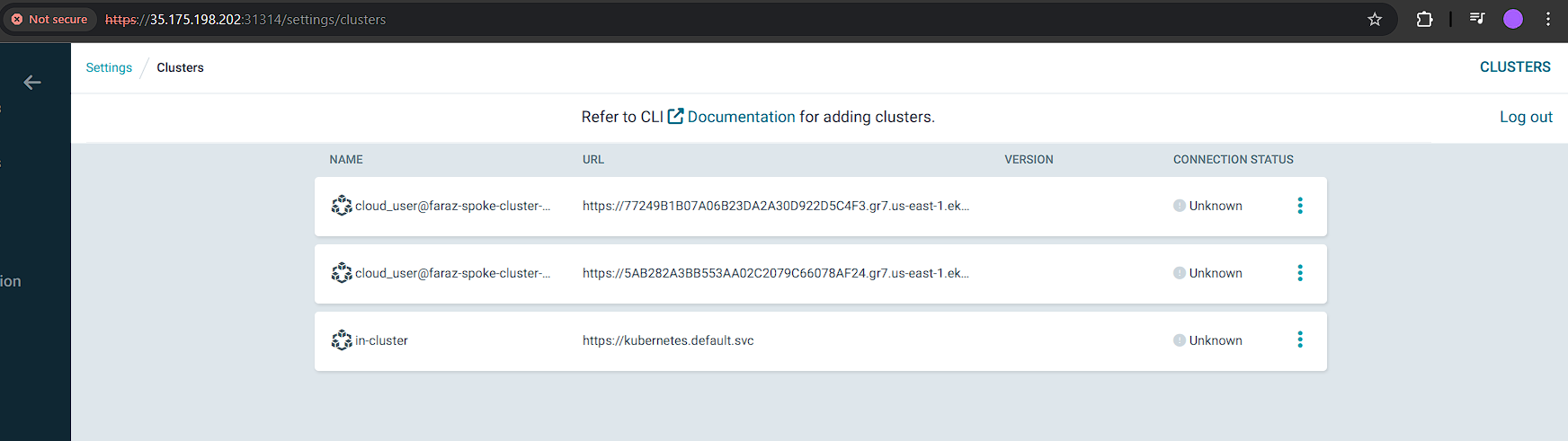
* argocd cluster add cloud\_user@faraz-spoke-cluster-1.us-east-1.eksctl.io --server 35.175.198.202:31314



* argocd cluster add cloud\_user@faraz-spoke-cluster-2.us-east-1.eksctl.io --server 35.175.198.202:31314

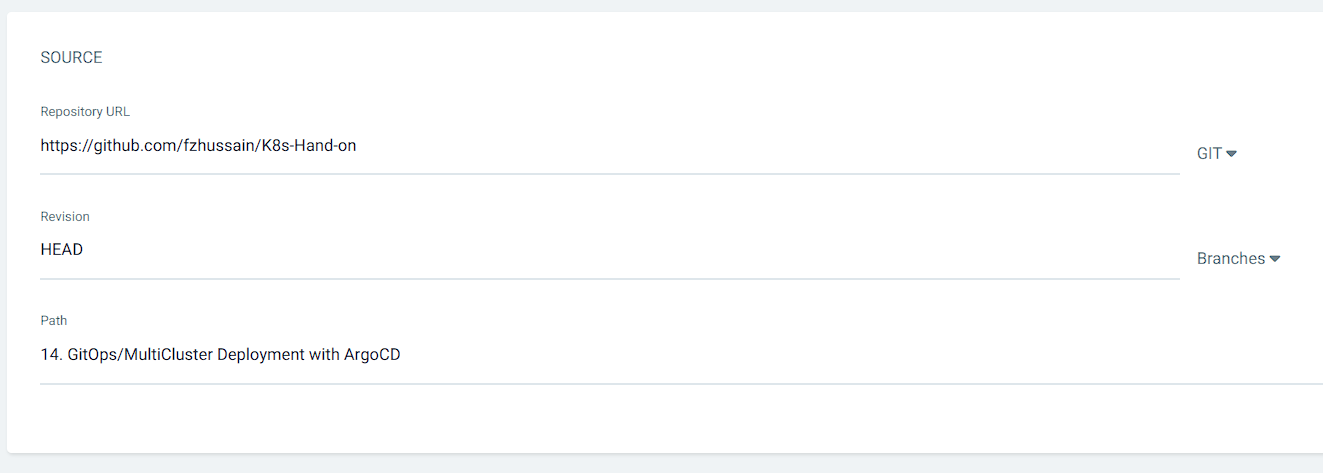


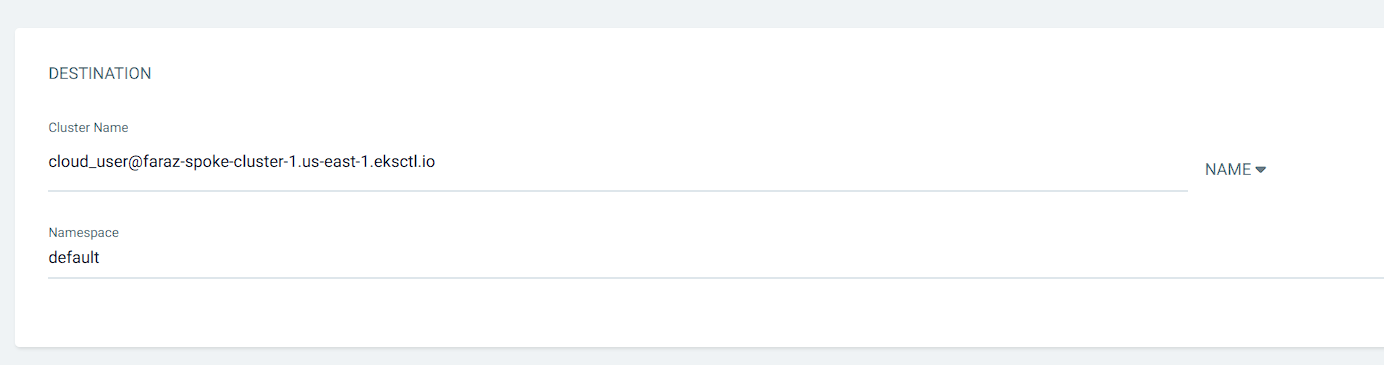
Now we will be able to see our spoke clusters being added to our Hub cluster:



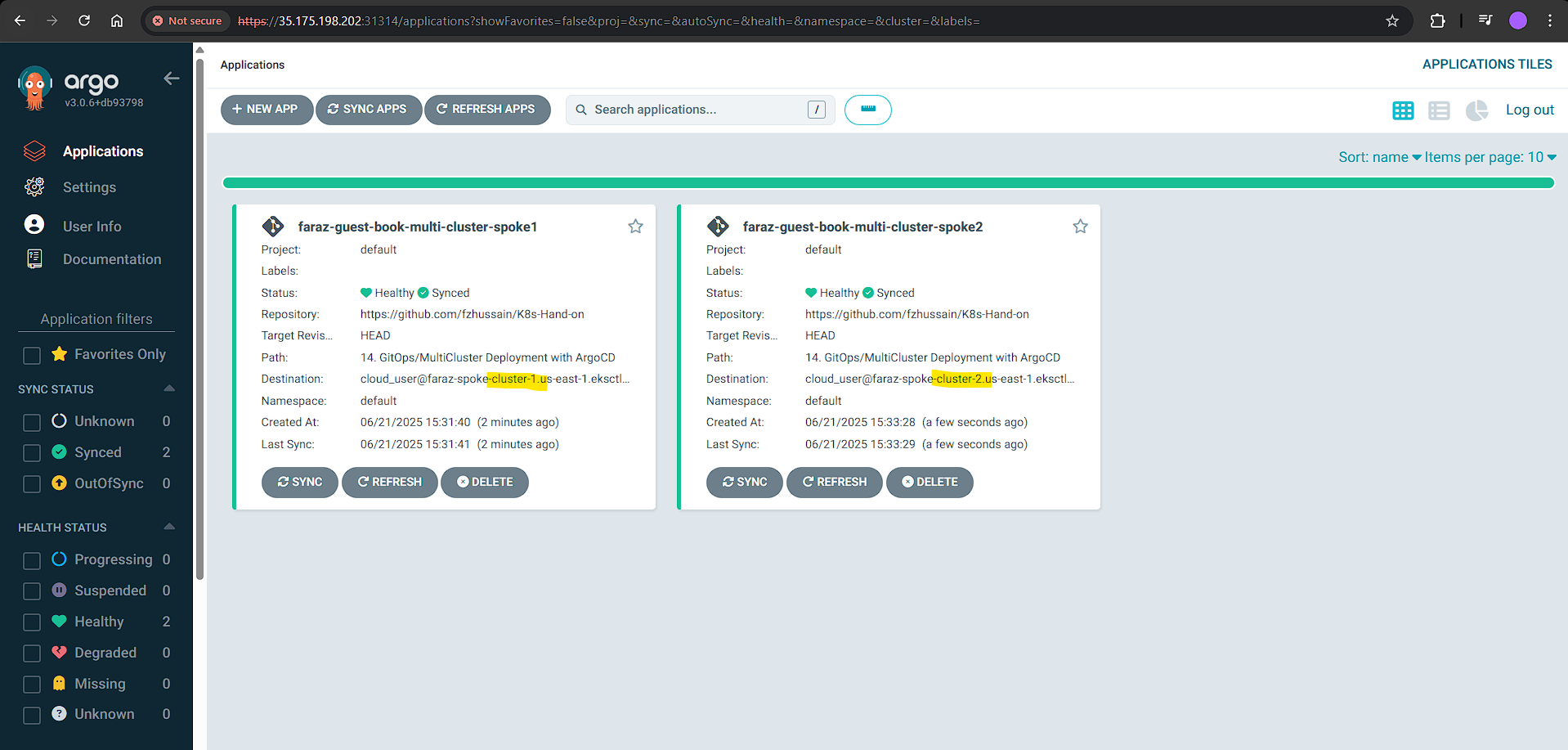
Now we will create our application:



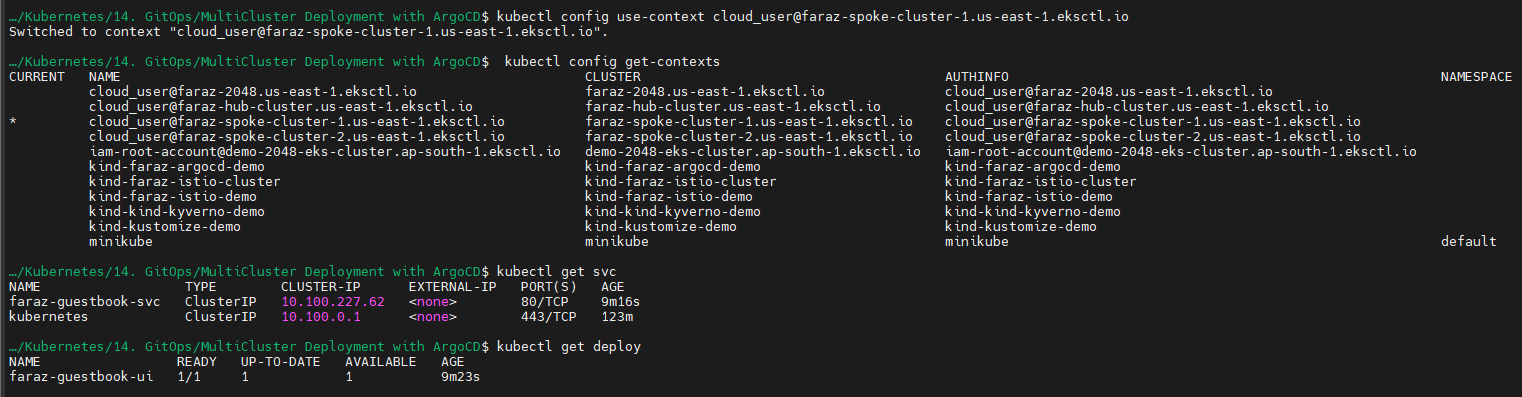


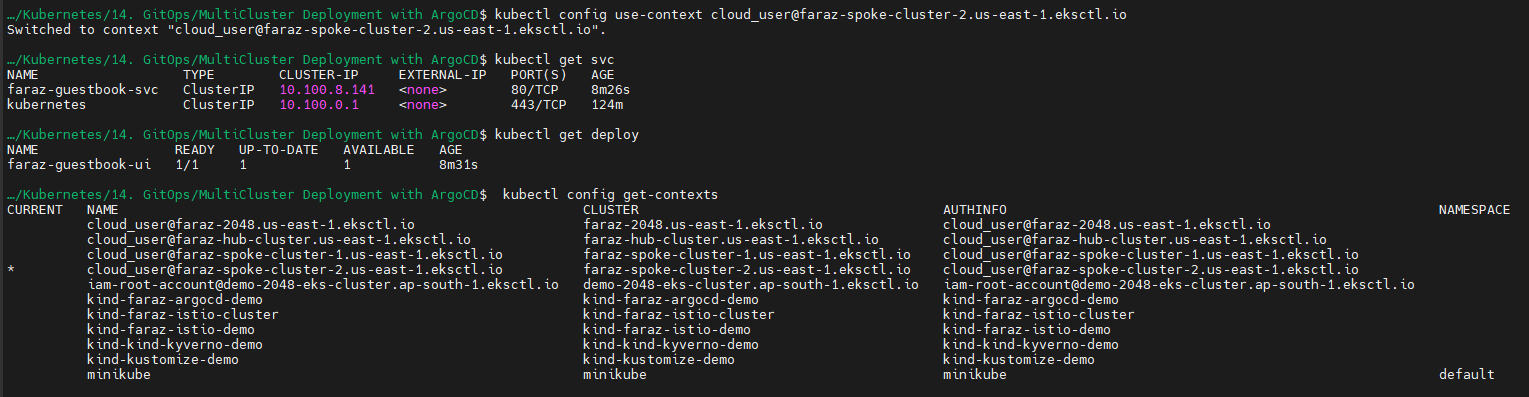


Similarly we will create and deploy in cluster 2:



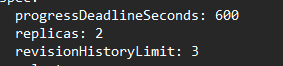
Verify your resource creation:



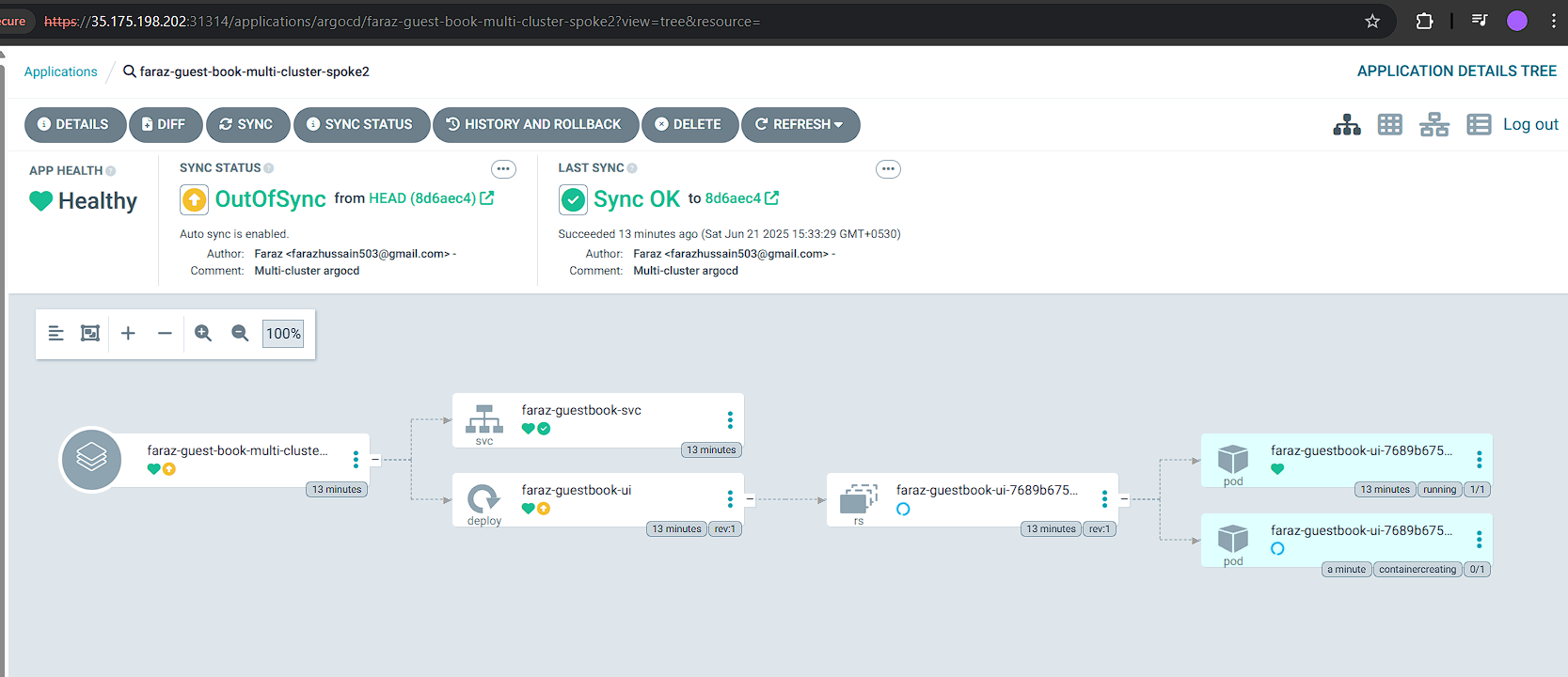


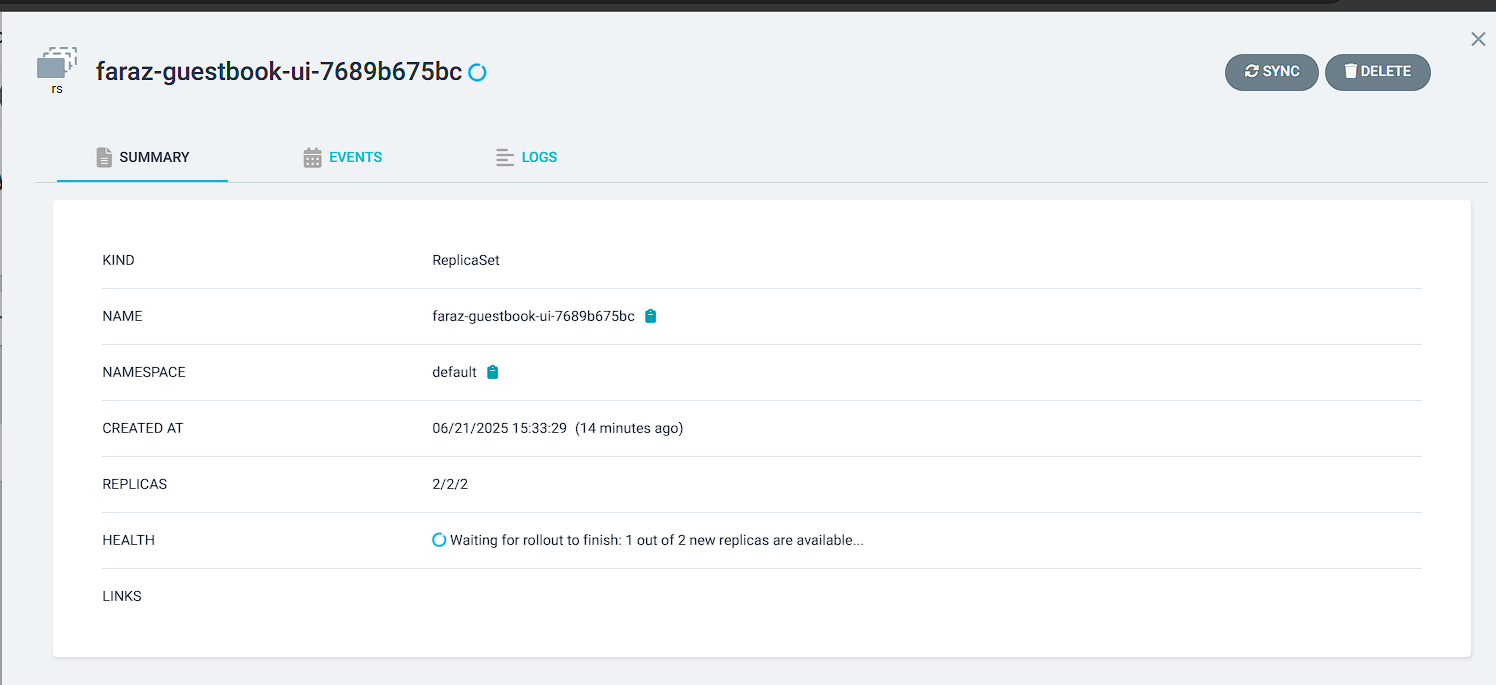
Now let’s edit the replicas:

* kubectl edit deploy faraz-guestbook-ui

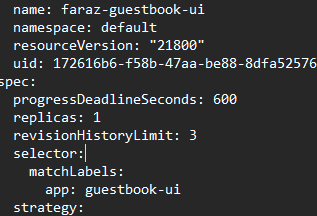


You will notice that we are out of sync:

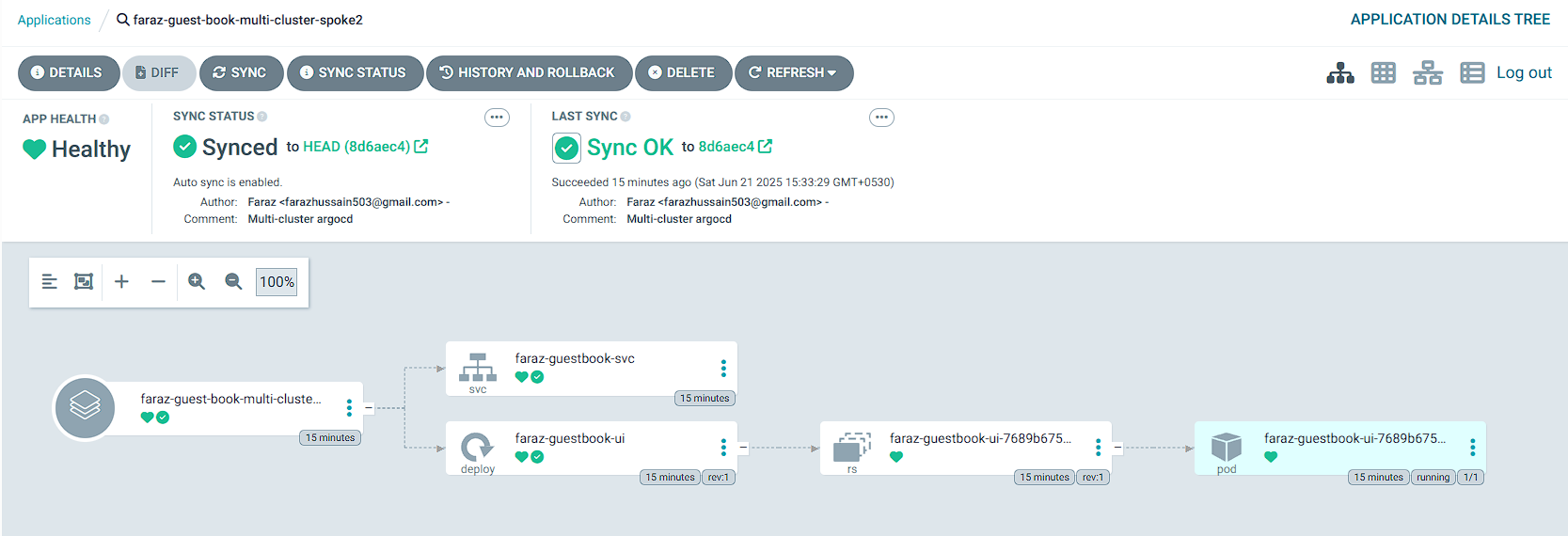




Now let’s edit it back to 1:

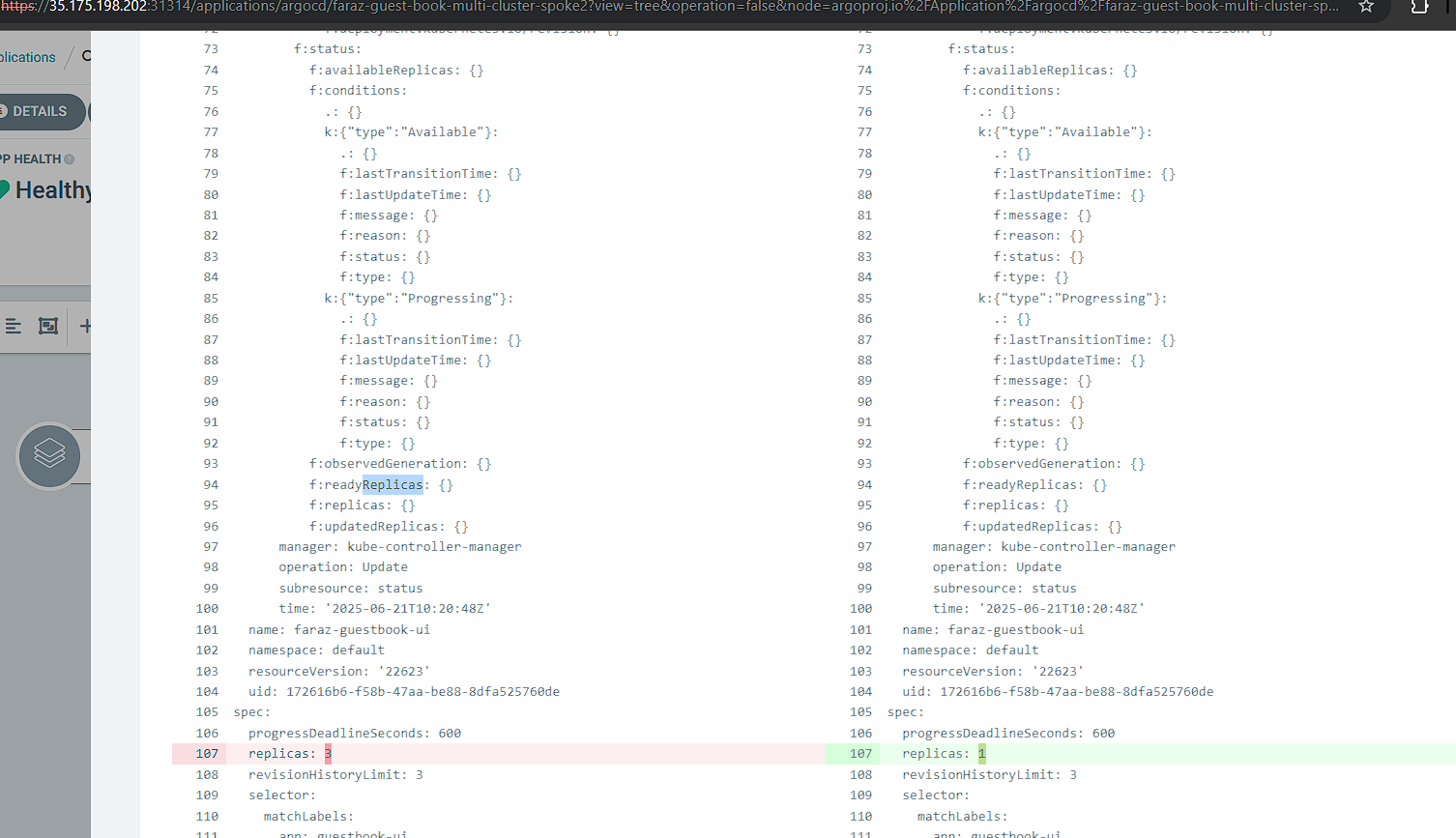


Now it will be synced:

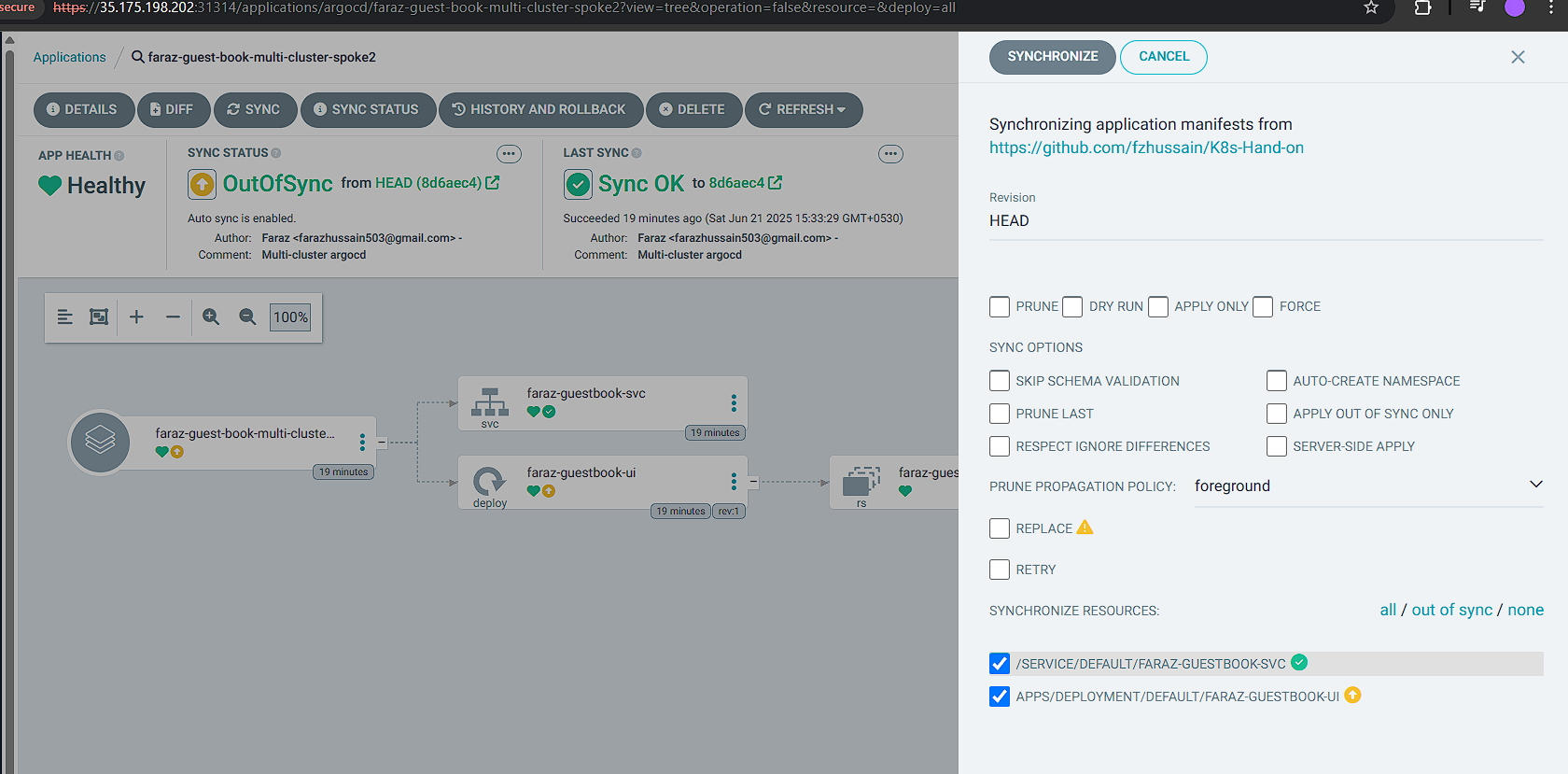


Now let’s change the replicas to 3:

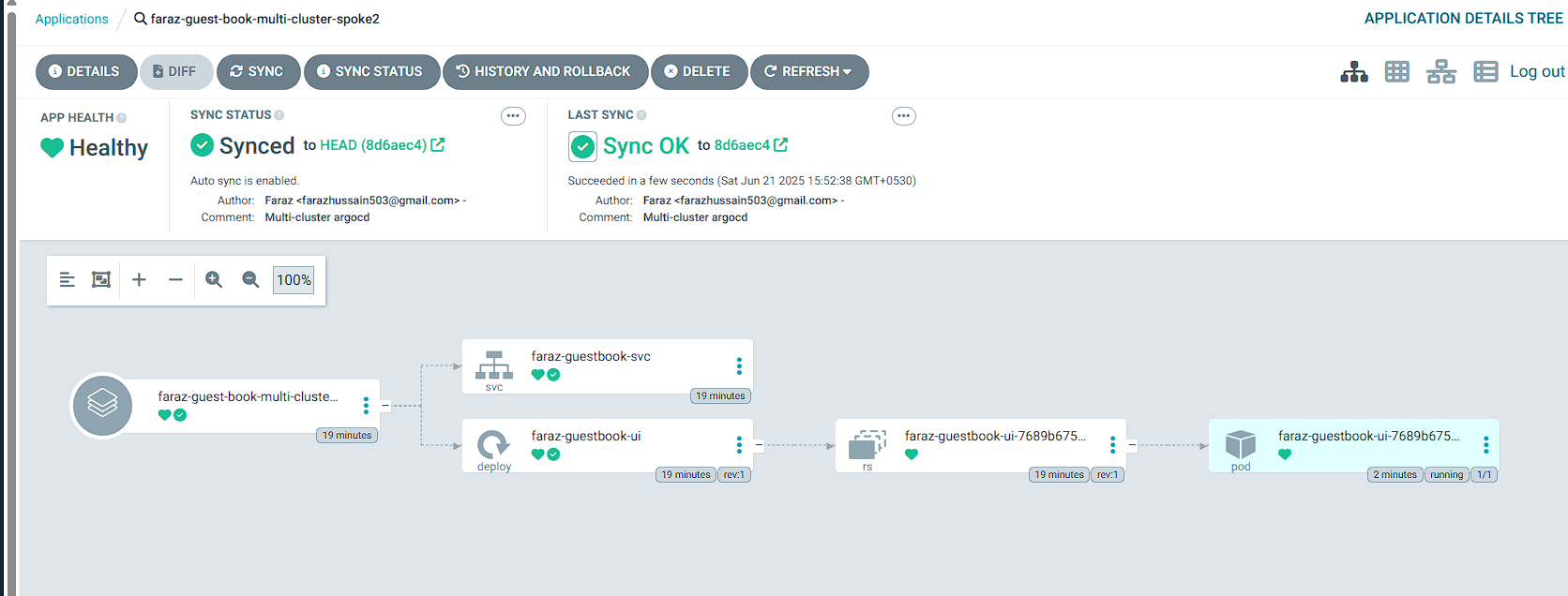
You can also check the difference:



Now if we click on sync:



It will remove those changes which we made manually directly on the cluster:



Is there a way of creating multiple applications at once?  
- Yes, by using application sets.