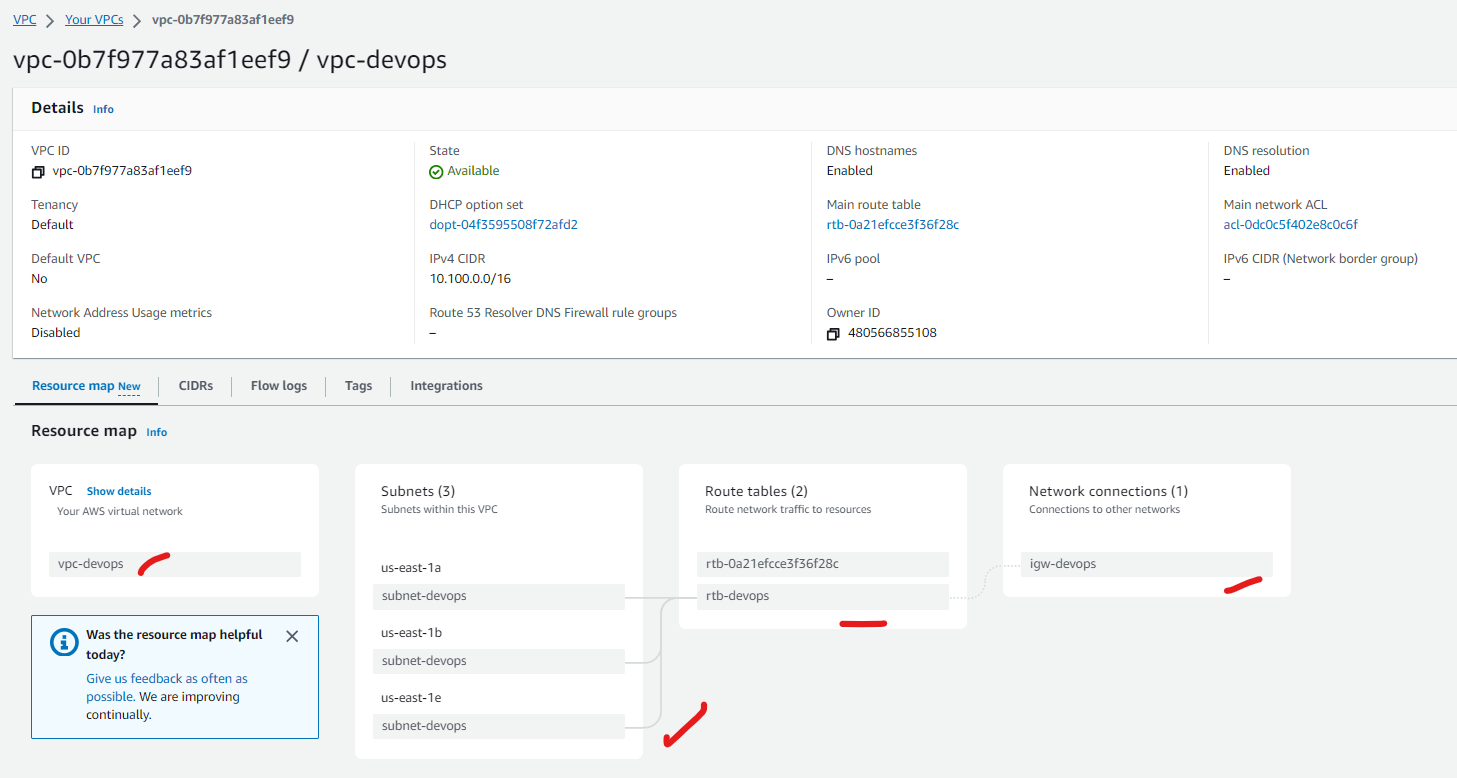
**Practical DevOps– Assignment on AWS**

1. **Setting up a CI/CD Pipeline and deploying applications on AWS EKS**

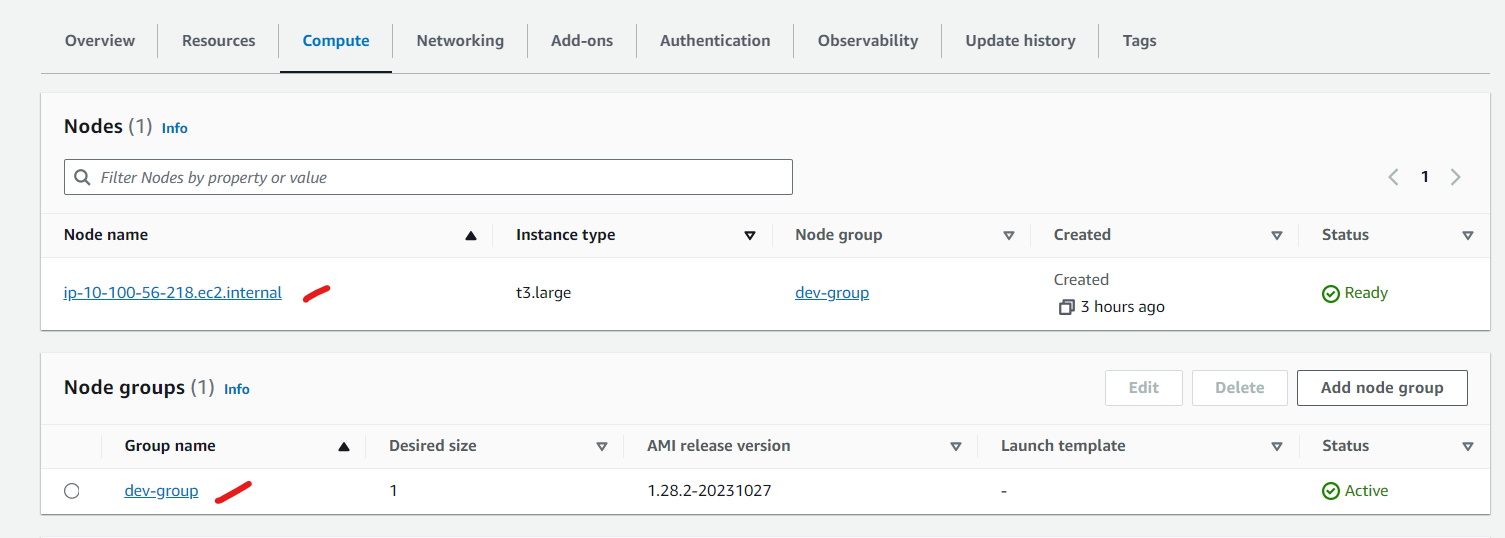
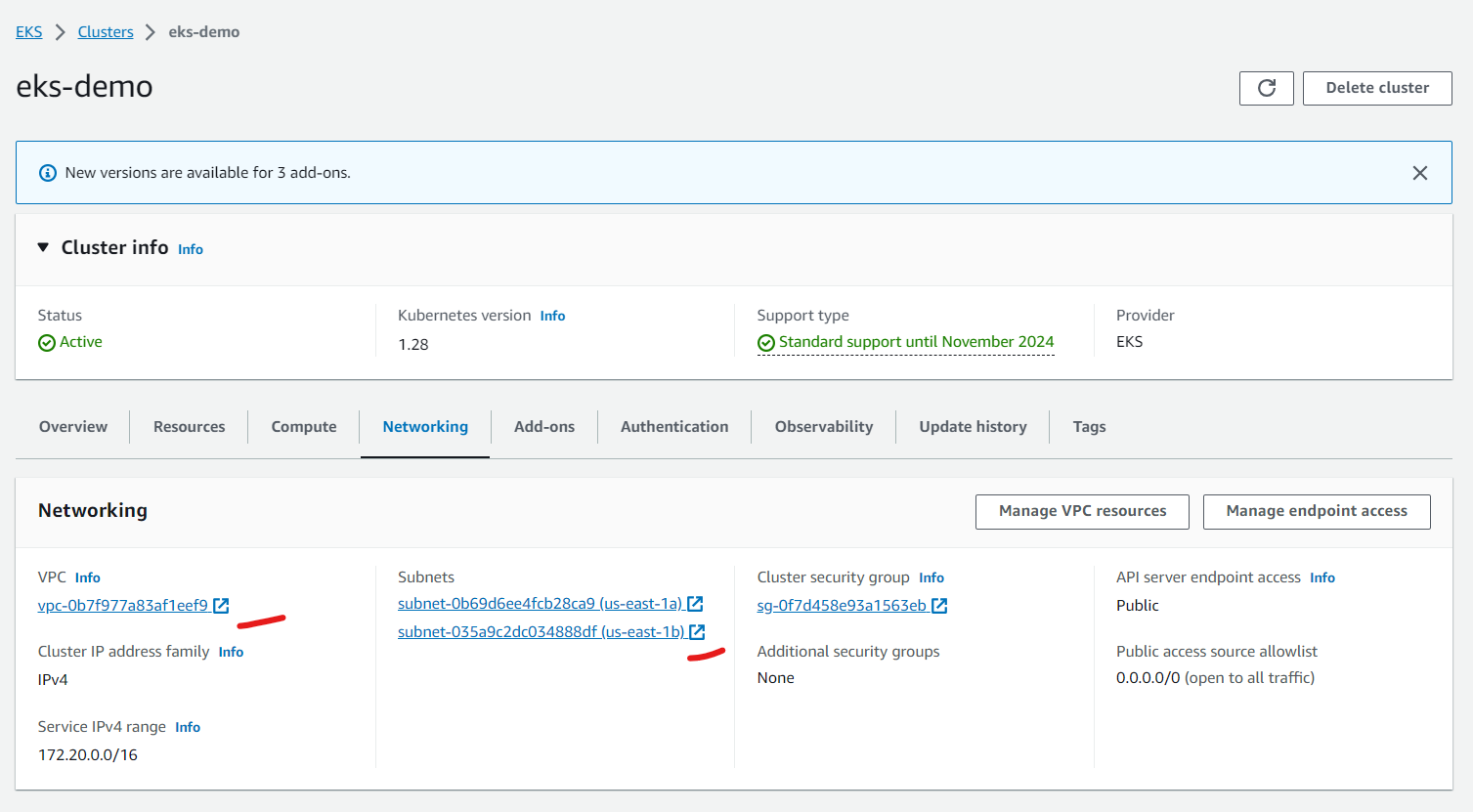
**Step 1: Create a terraform file to provision and manage resources on AWS**

* Refer link : <https://github.com/Chau-NH/sd4322_aws_infrastructure>

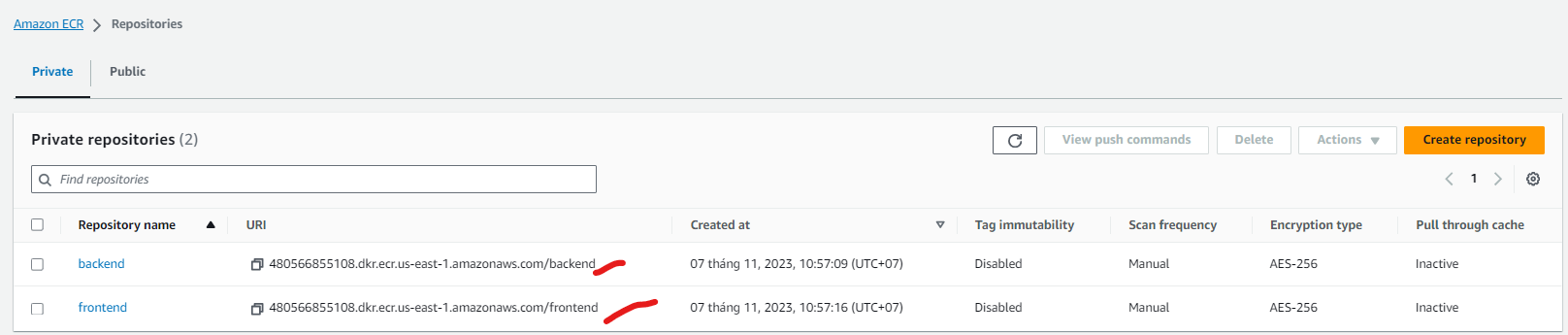
**VPC**



**EKS**

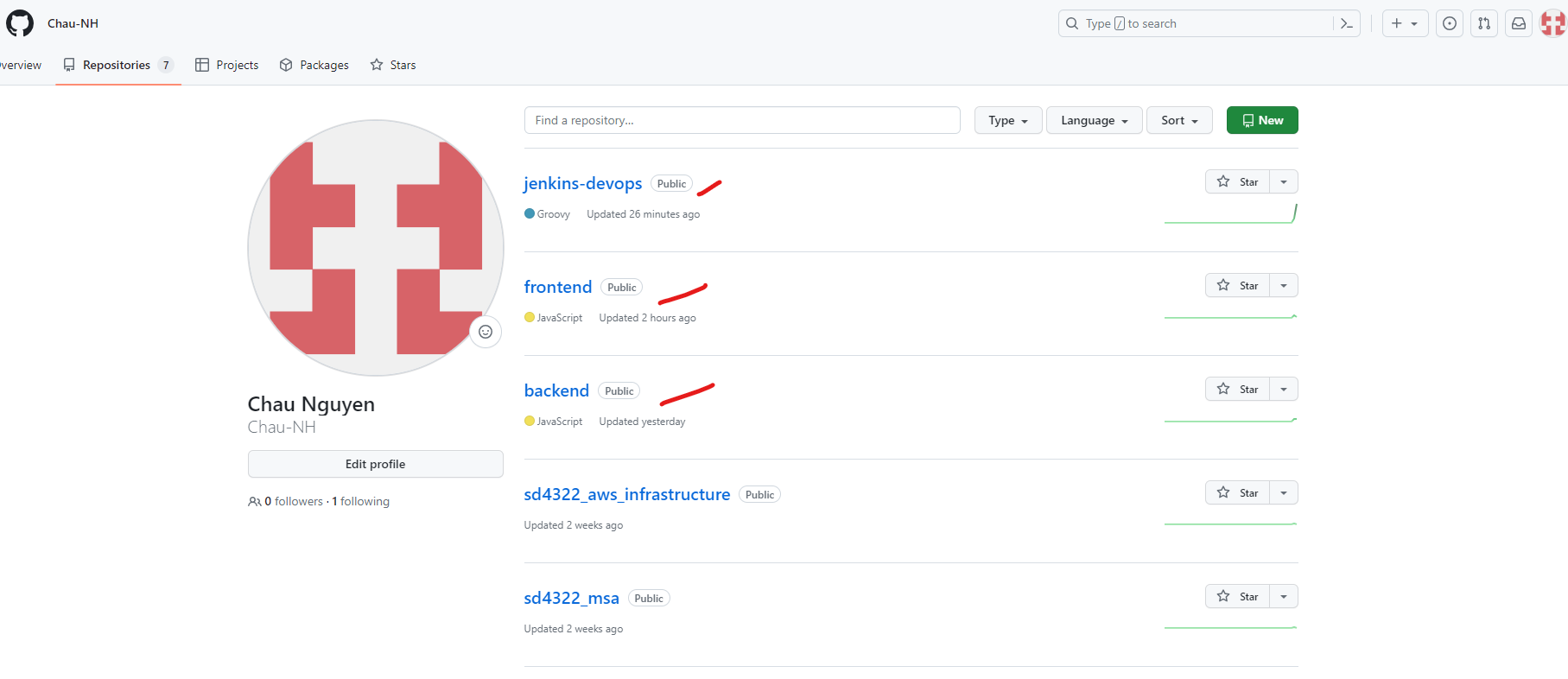


**ECR**

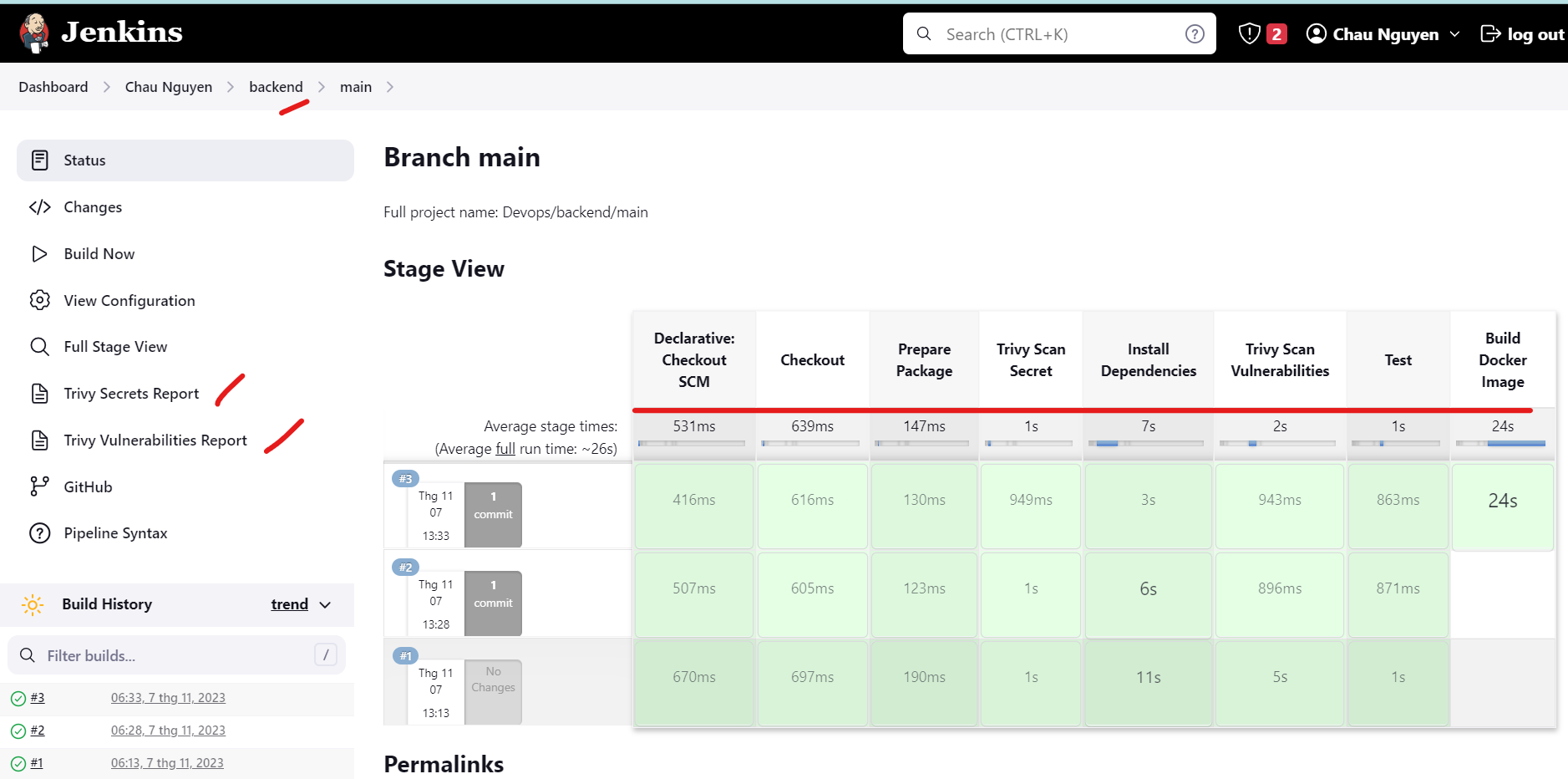


**Step 2: Set up Jenkins for CI process applying Shared Library and Trivy**

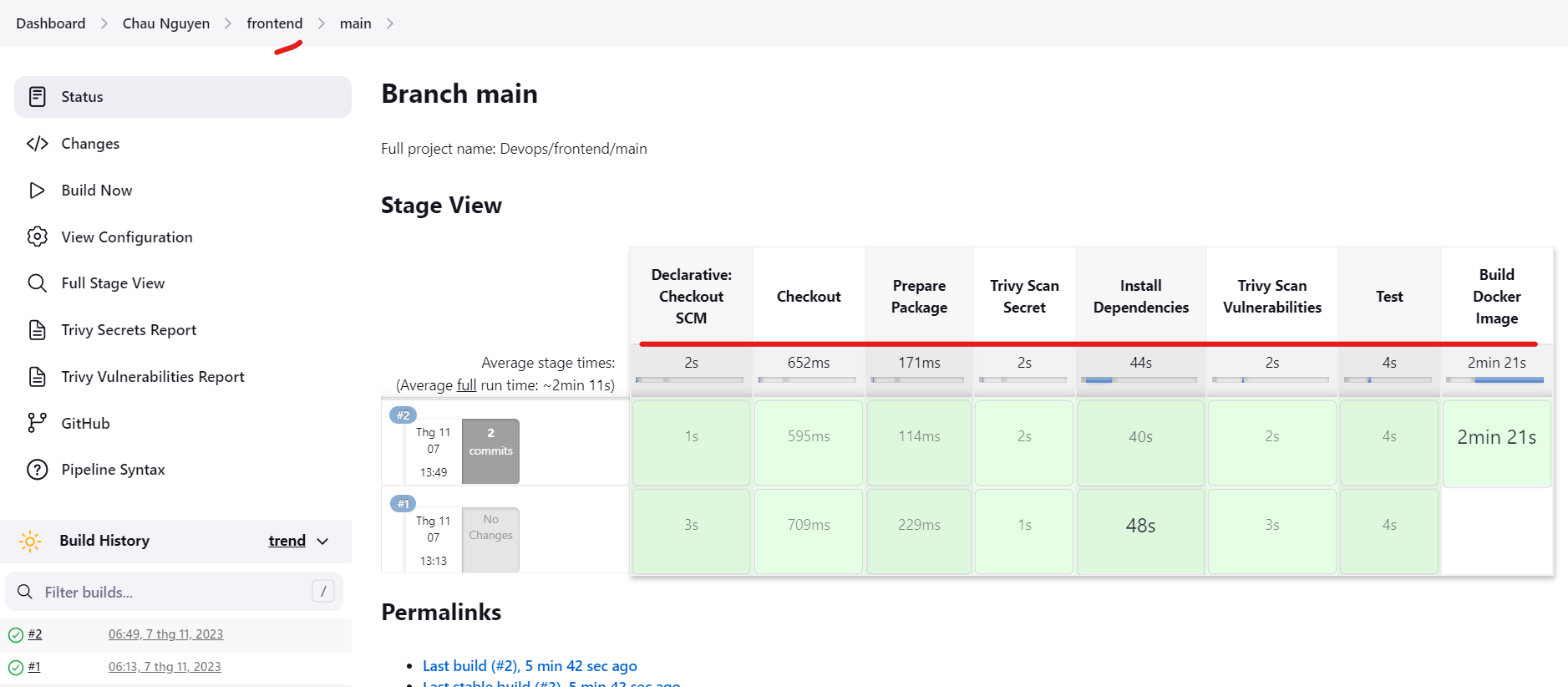
* Source code for micro services: **frontend and backend**
  + <https://github.com/Chau-NH/frontend>
  + <https://github.com/Chau-NH/backend>
* Source code for Jenkins Pipeline**: jenkins-devops** (apply Shared Library)
  + <https://github.com/Chau-NH/jenkins-devops>

**GitHub repositories:** 

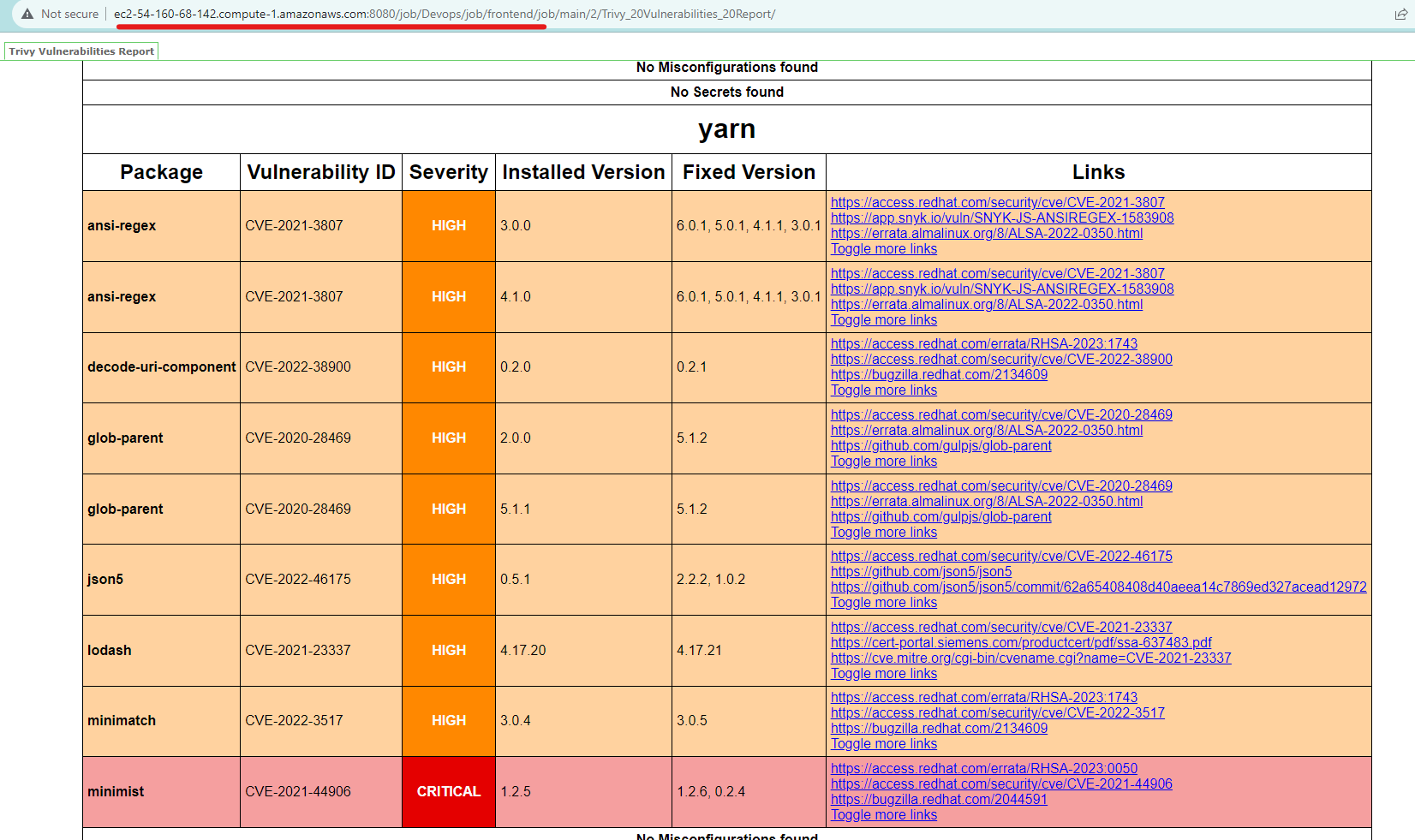
**Backend Pipeline**



**Frontend Pipeline**



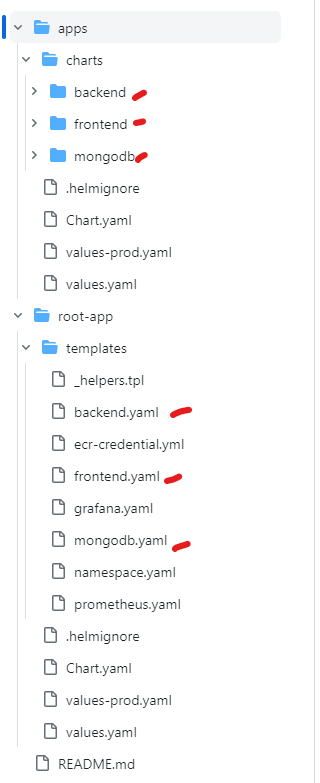
**Trivy security scan result**



1. **Set up ArgoCD for CD process working with Helm charts**

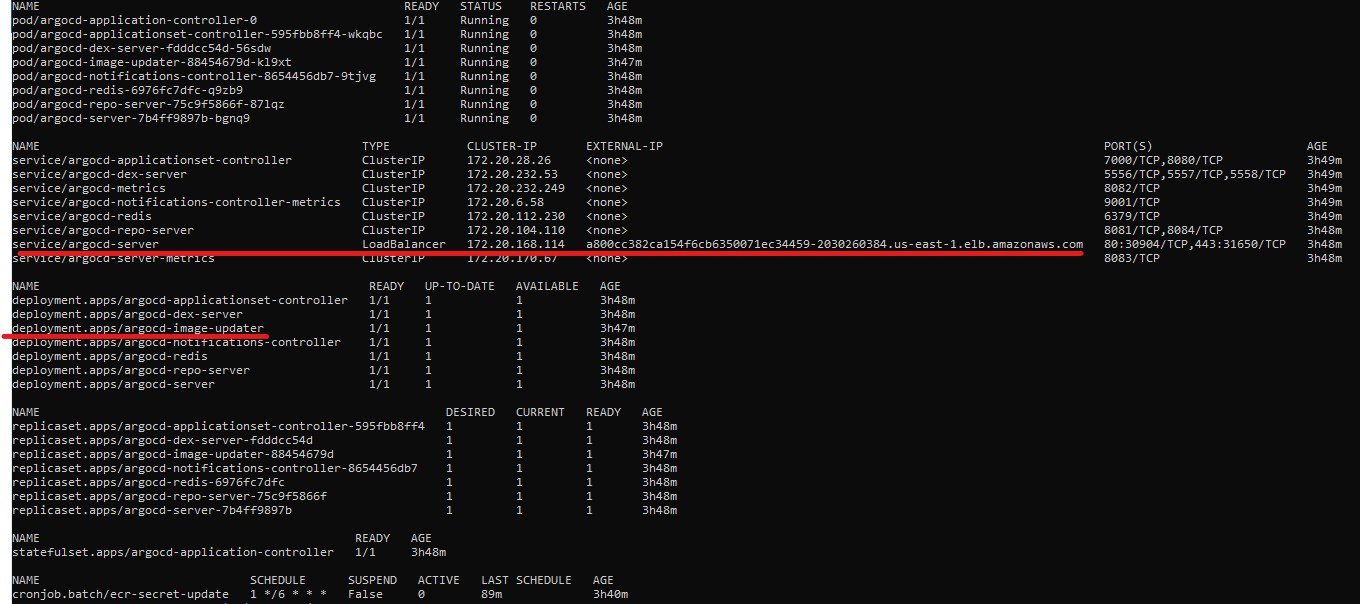
**Step 1: Create a Helm chart repository to manage manifest**

* Source code for Helm chart: <https://github.com/Chau-NH/helm-charts>
* In order to apply **Promotion Environment pattern** we will create values.yaml file for each environment
  + Dev: values-dev.yaml
  + Staging: values-staging.yaml
  + Production: values-prod.yaml

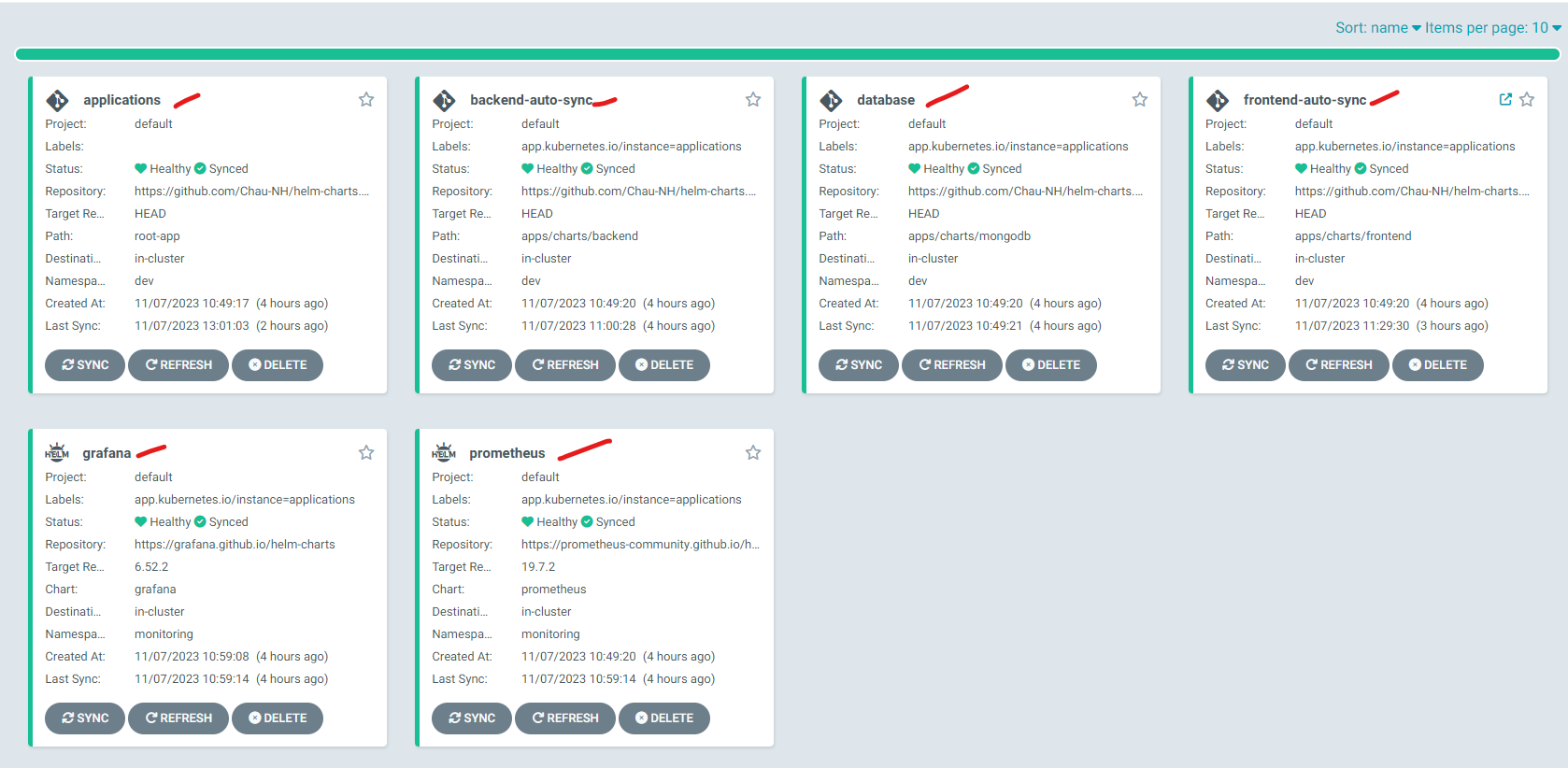
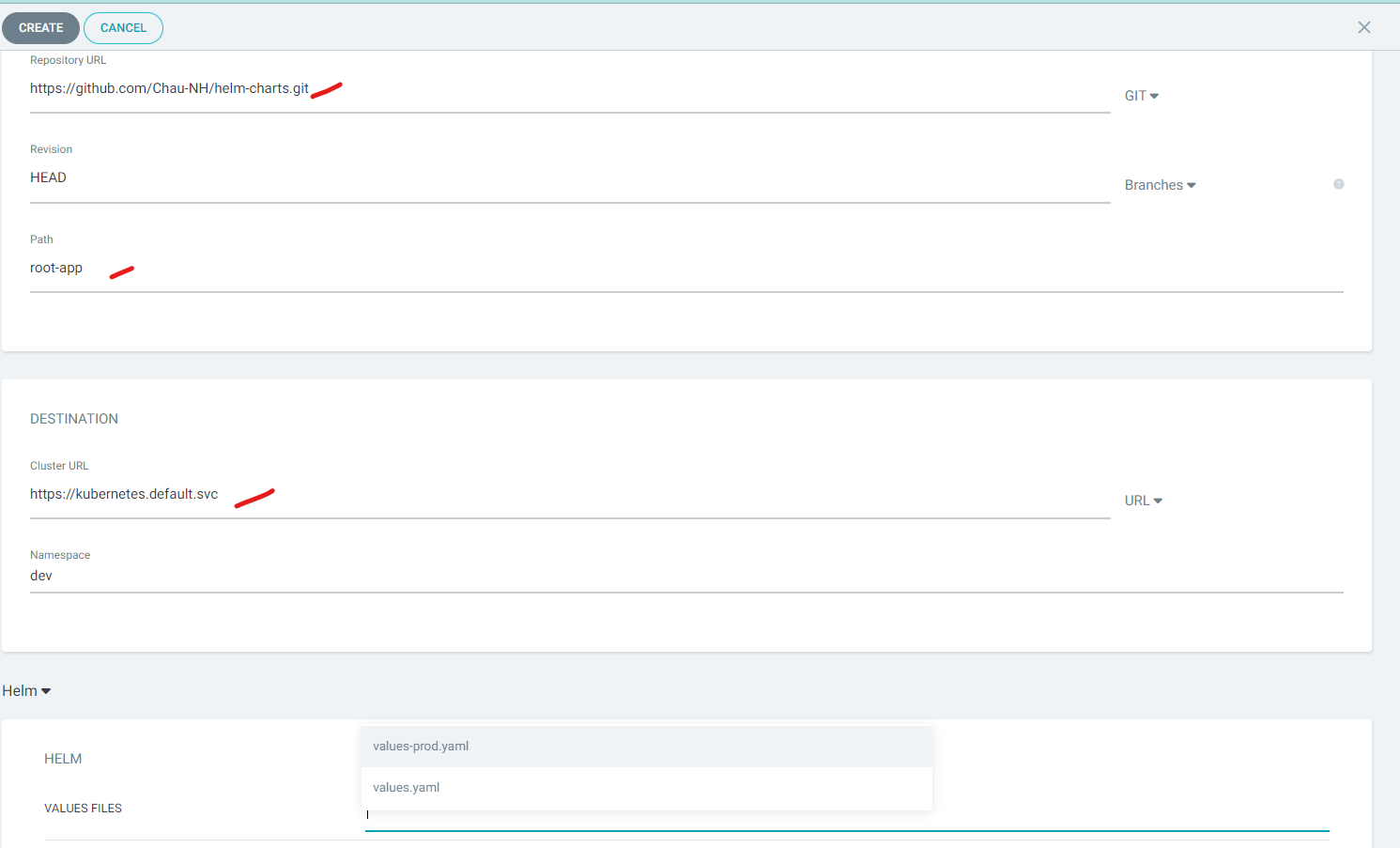


**Step 2: Install ArgoCD and create application with ArgoCD**

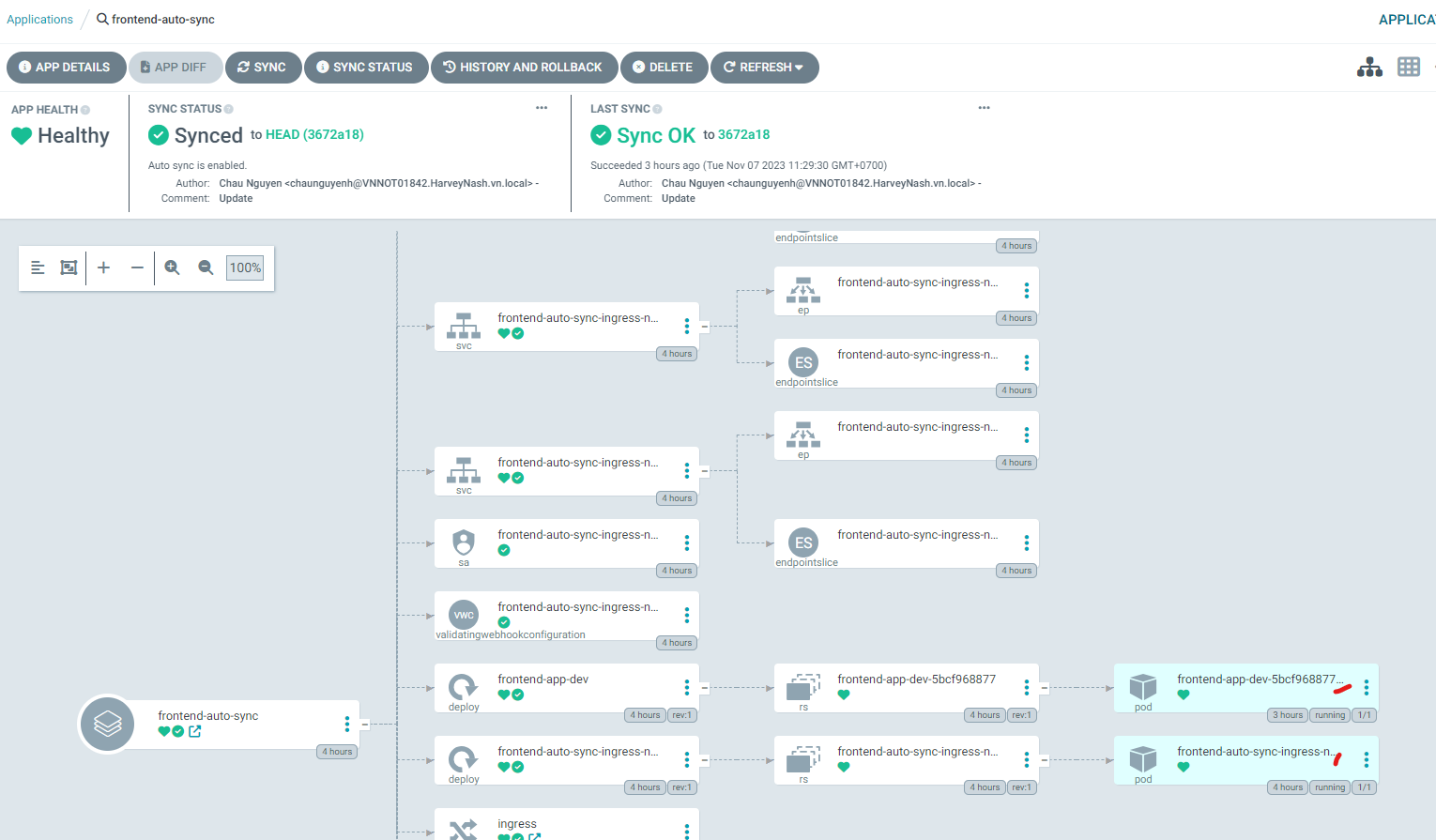
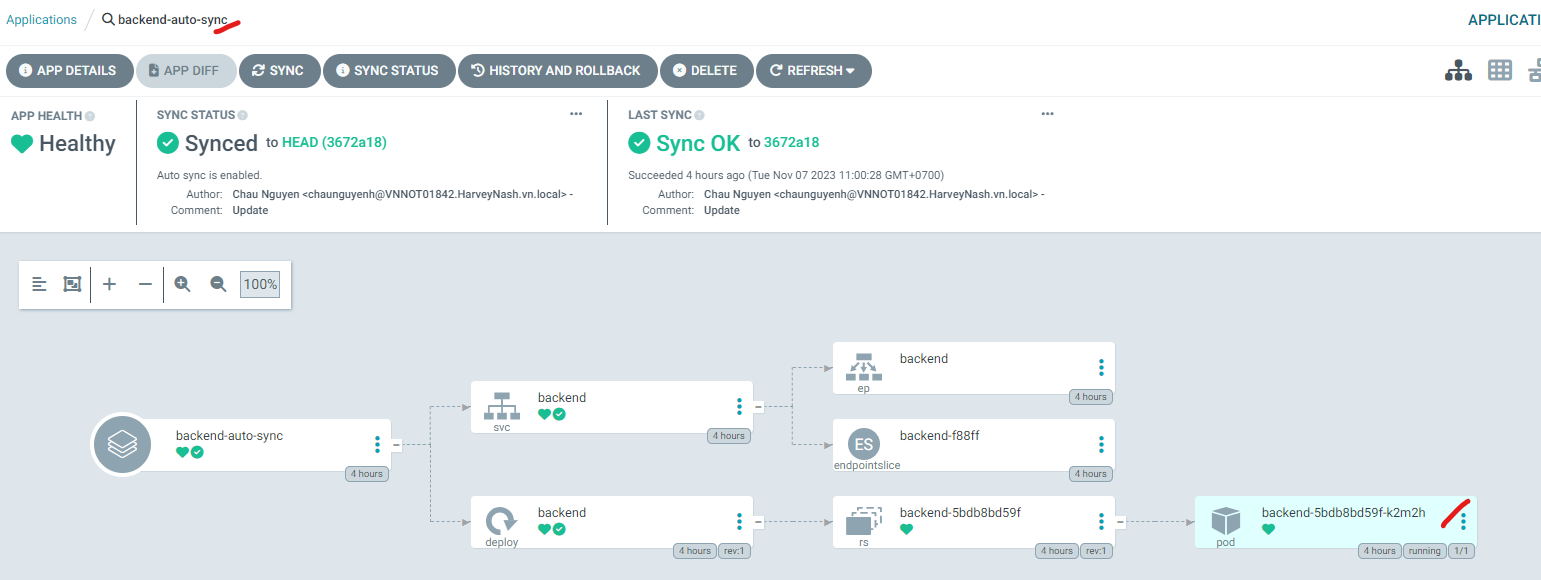
**Result on terminal**



**Result on UI**

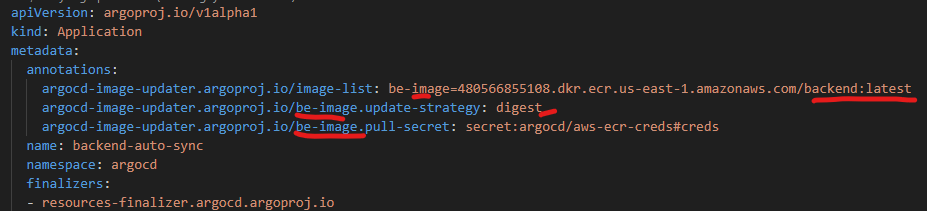
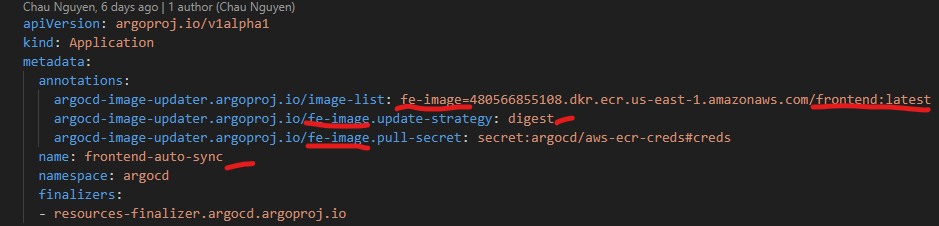


**Backend  
  
Frontend**



**Step 3: Using ArgoCD Image Updater to deploy automatically**

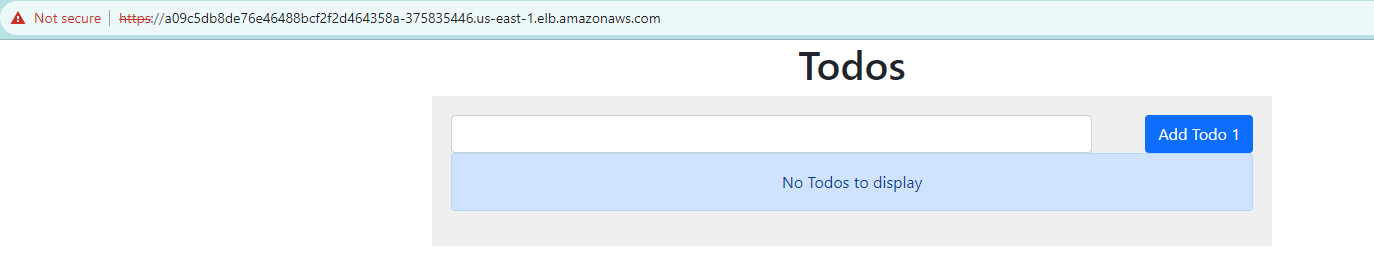
* Apply ArgoCD Image Updater to Backend and Frontend
* We apply **Digest strategy** for ArgoCD Image Updater.



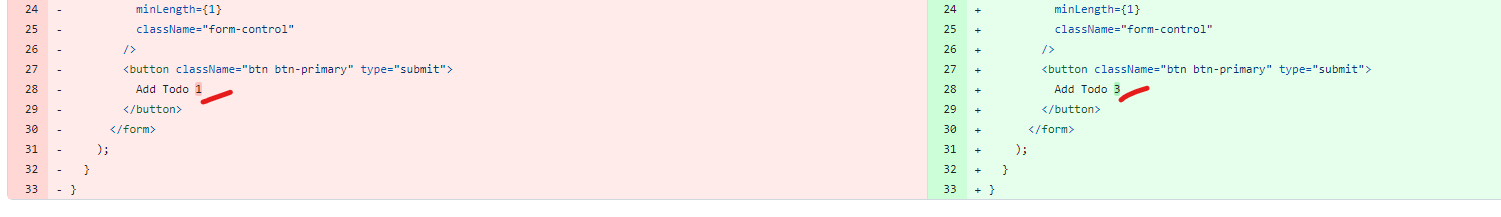
* **Digest strategy** updates to the latest version of a given version (tag), using the tag's SHA digest

**Step 4: Check whether ArgoCD Image Updater is woring or not**

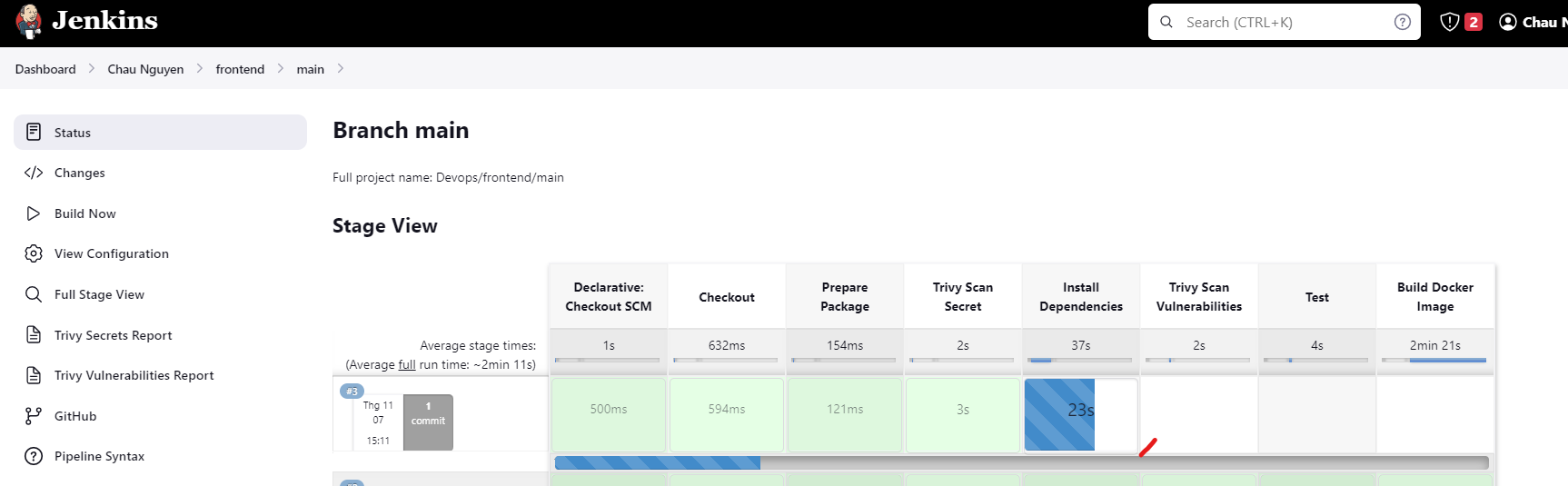
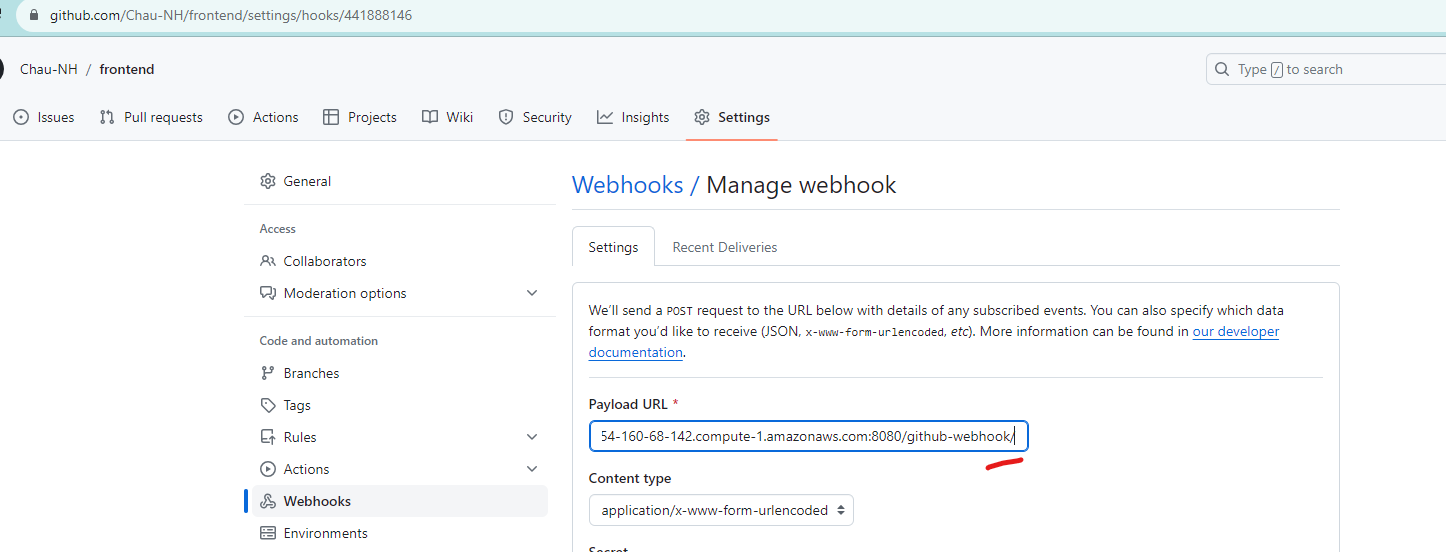
**Go to UI before pushing code**



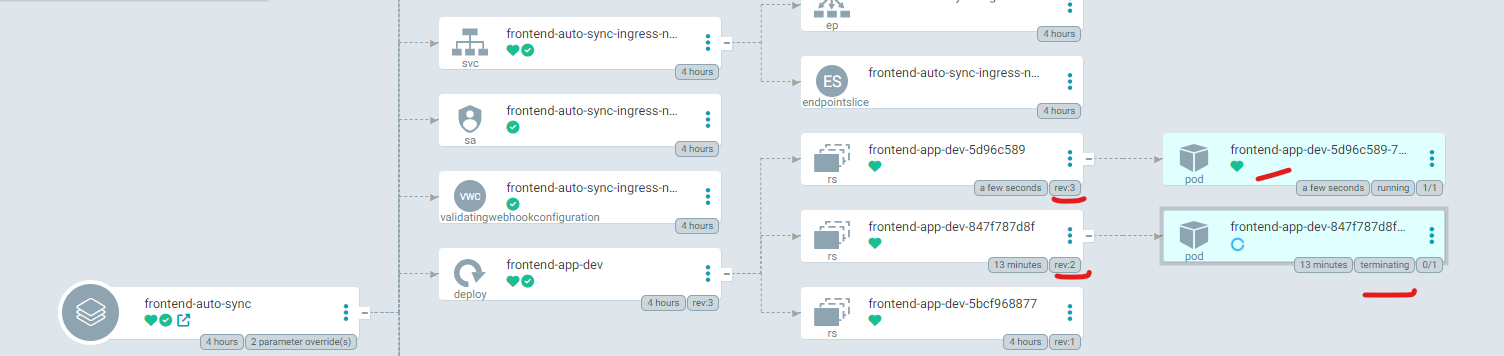
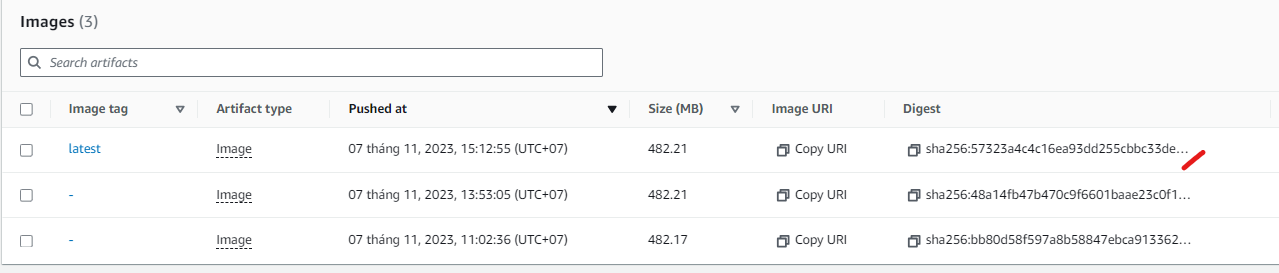
**Push a commit to repo**



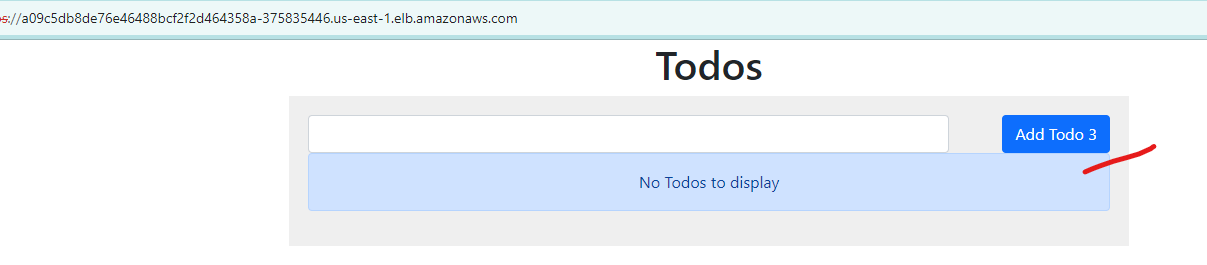
**Jenkins automatically run for every push (Webhook added)**



**ArgoCD automatically run after new Docker image uploaded**

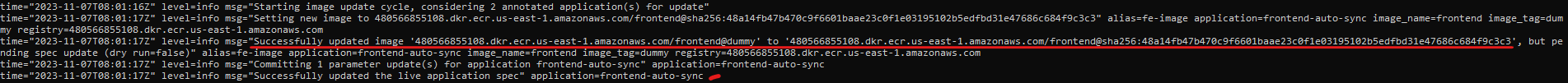


**Go back to UI**



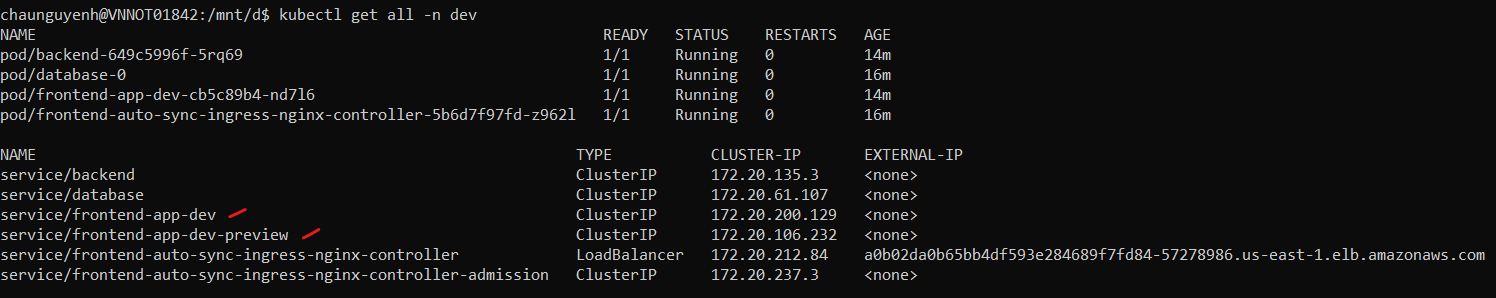
**Check logs of Argocd Image Updater**

Run : kubectl logs deployment.apps/argocd-image-updater -n argocd

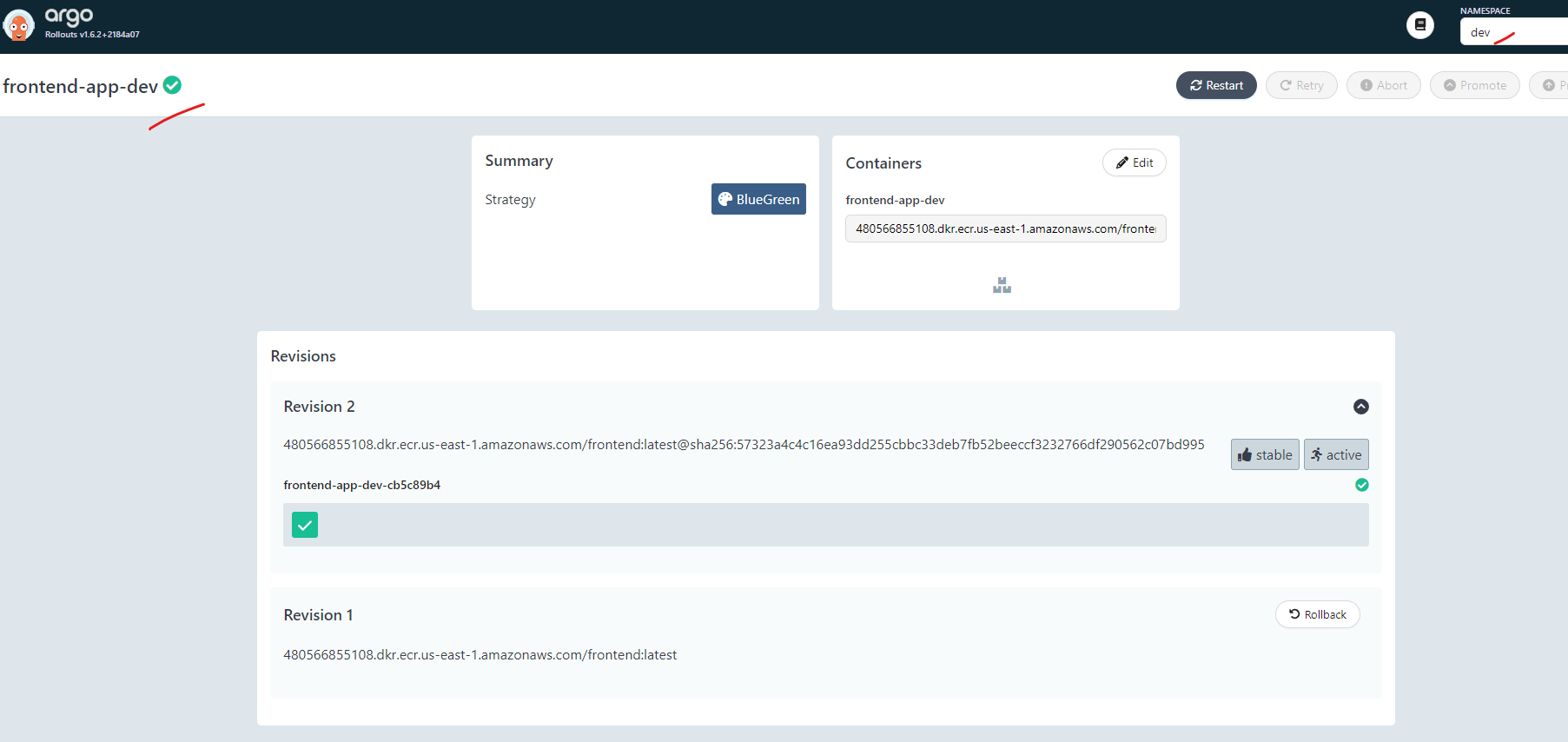
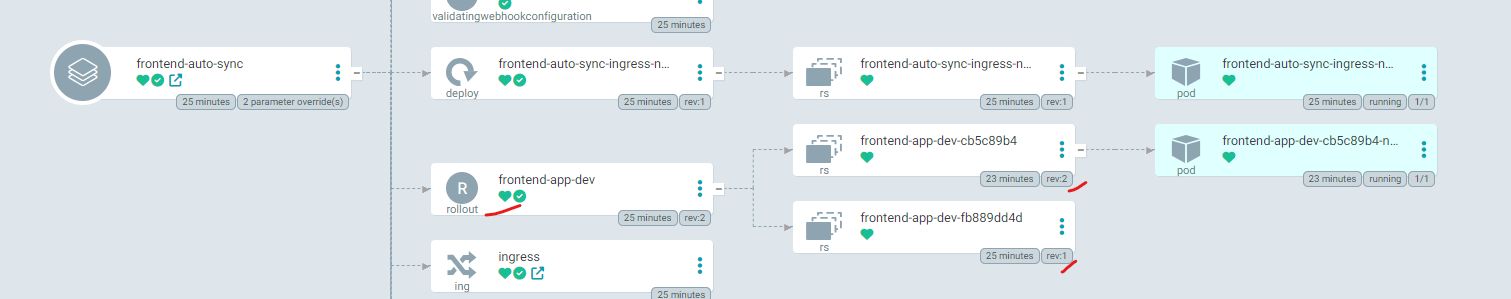
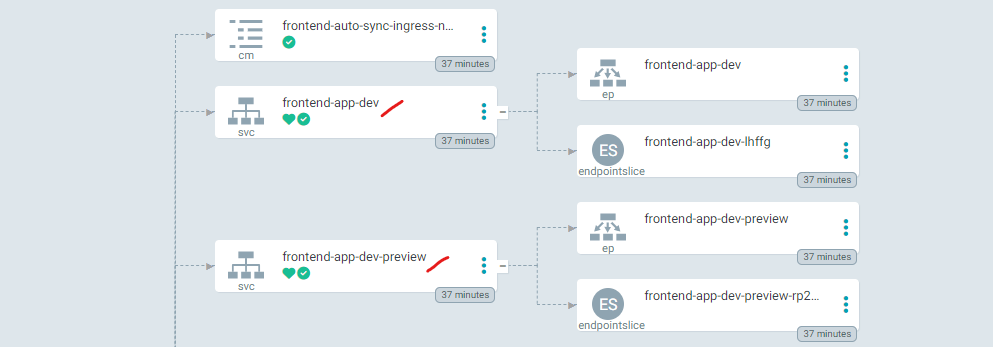


**Step 5: Blue/Green deployment strategy with Argo Rollouts**

**Apply to Frontend service**

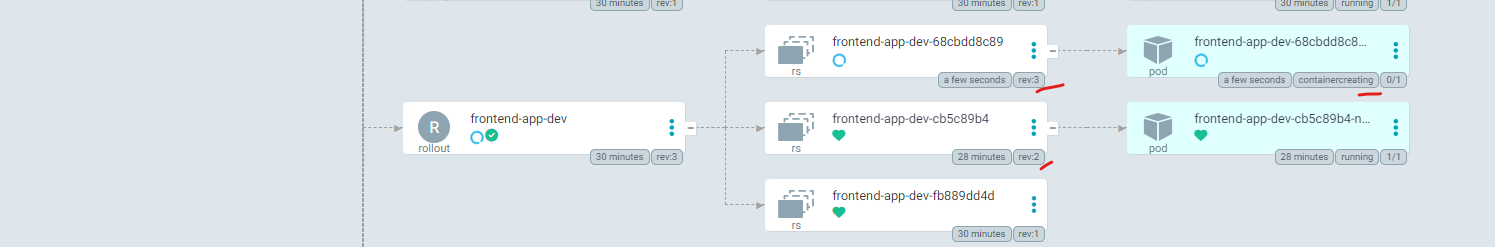


**Check on ArgoCD UI**



**Step 6: Deploy new docker image to check how Blue/Green strategy work**

**Check on ArgoCD UI**

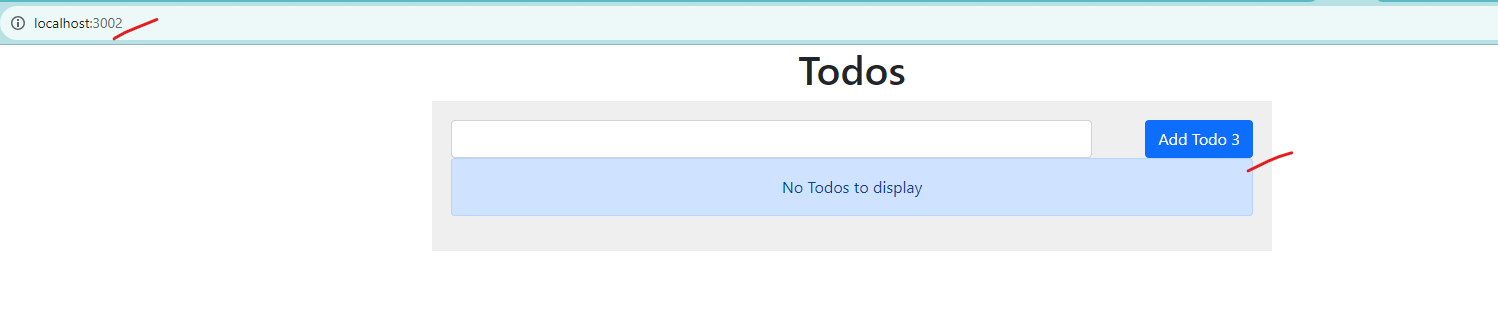
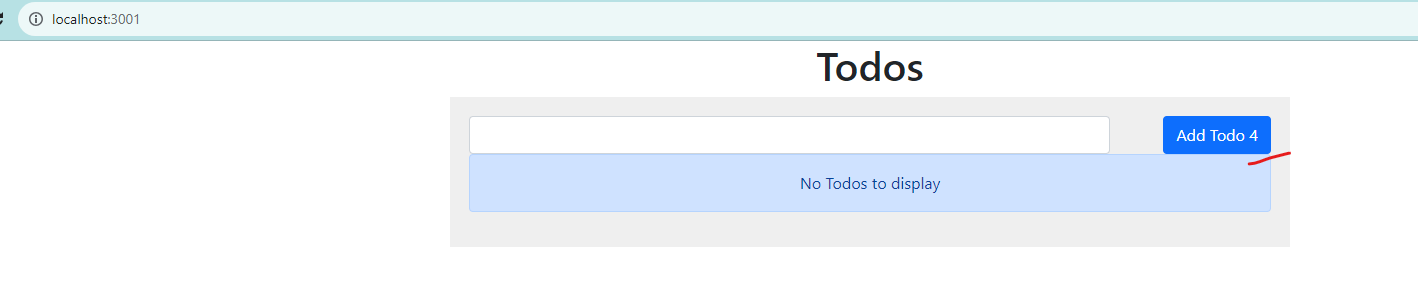


**Check on Argo Rollout Dashboard**

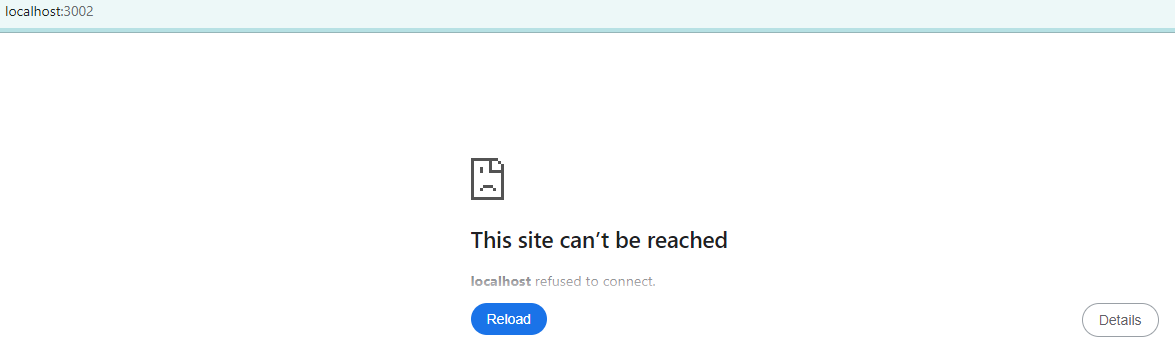
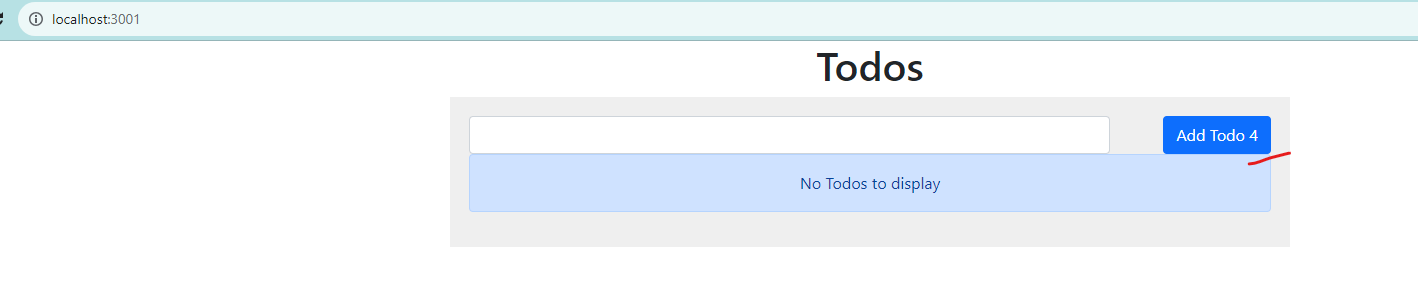
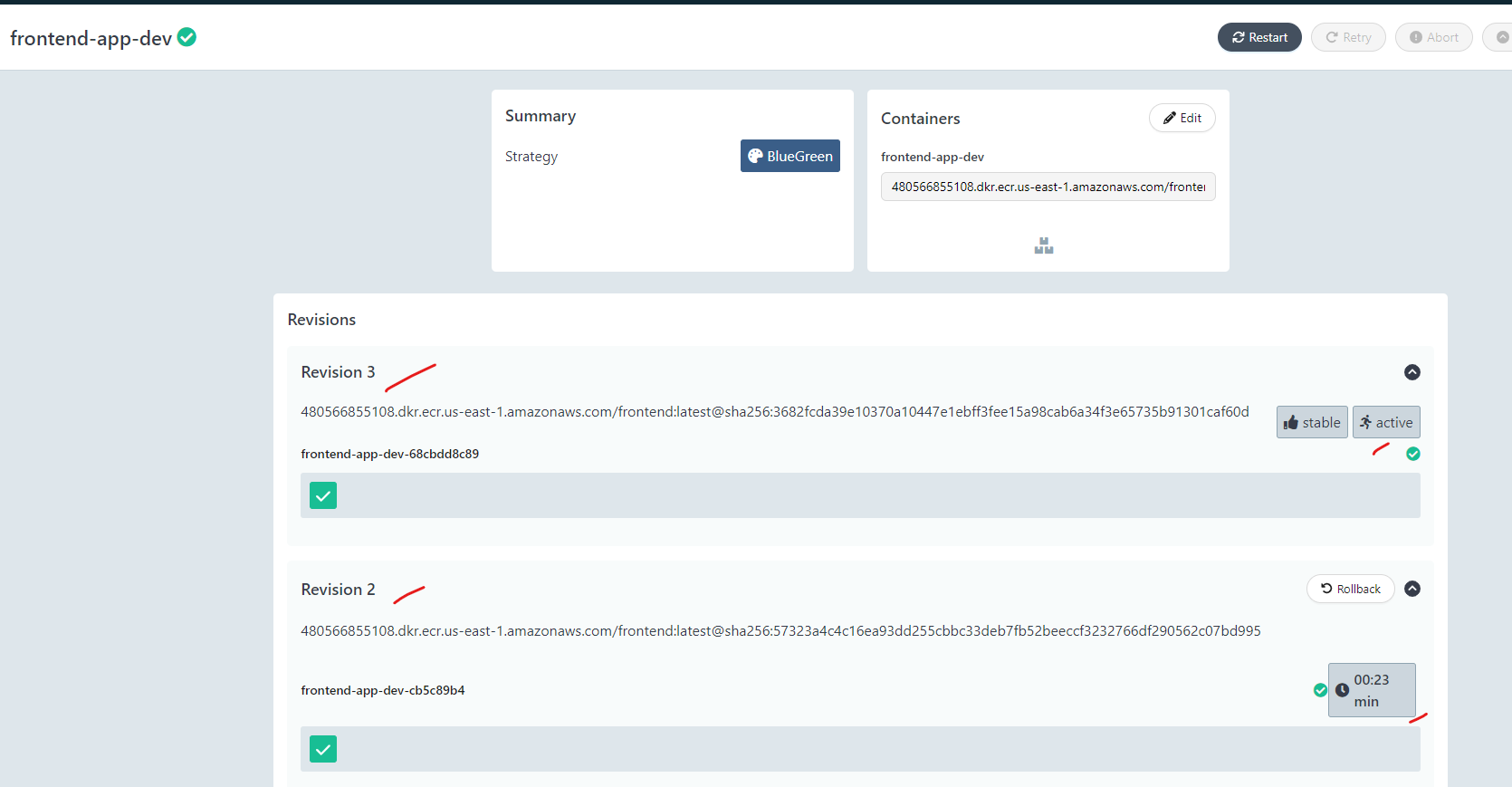


**Port forward 2 services to localhost**- kubectl port-forward service/frontend-app-dev 3001:3000 -n dev  
- kubectl port-forward service/frontend-app-dev-preview 3002:3000 -n dev

**Check on UI**  
Service with **port 3001** is showing new versionService with **port 3002** is showing old version

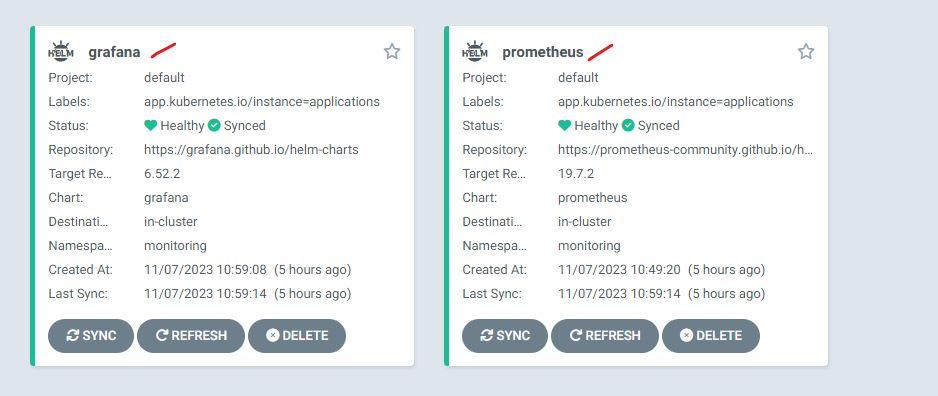


**Execute Promotion on Argo Rollout Dashboard**Service with **port 3001** is showing new version  
  
Service with **port 3002** is not working anymore

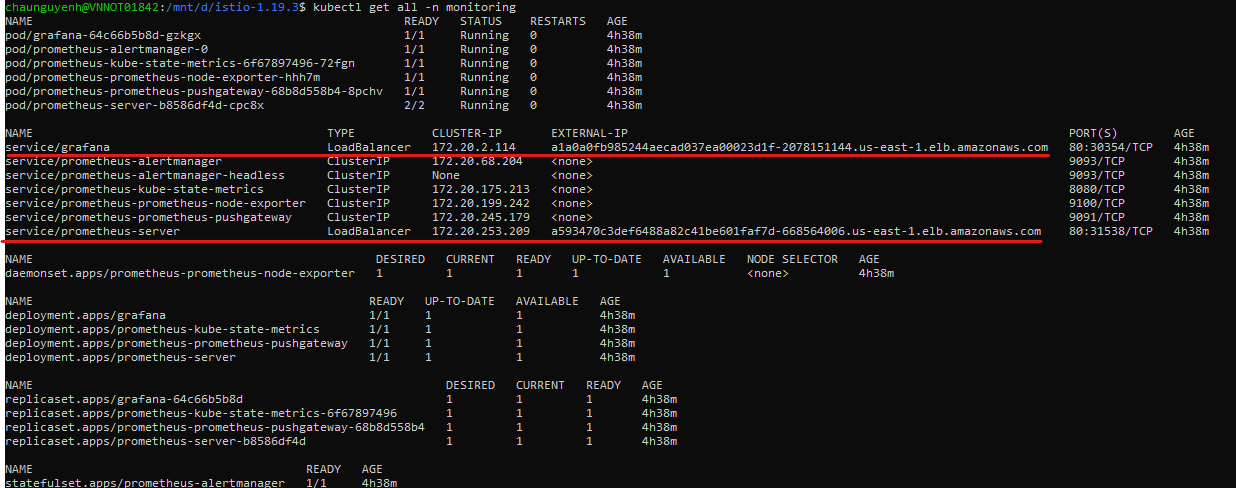


1. **Set up Prometheus and Grafana with Helm charts**

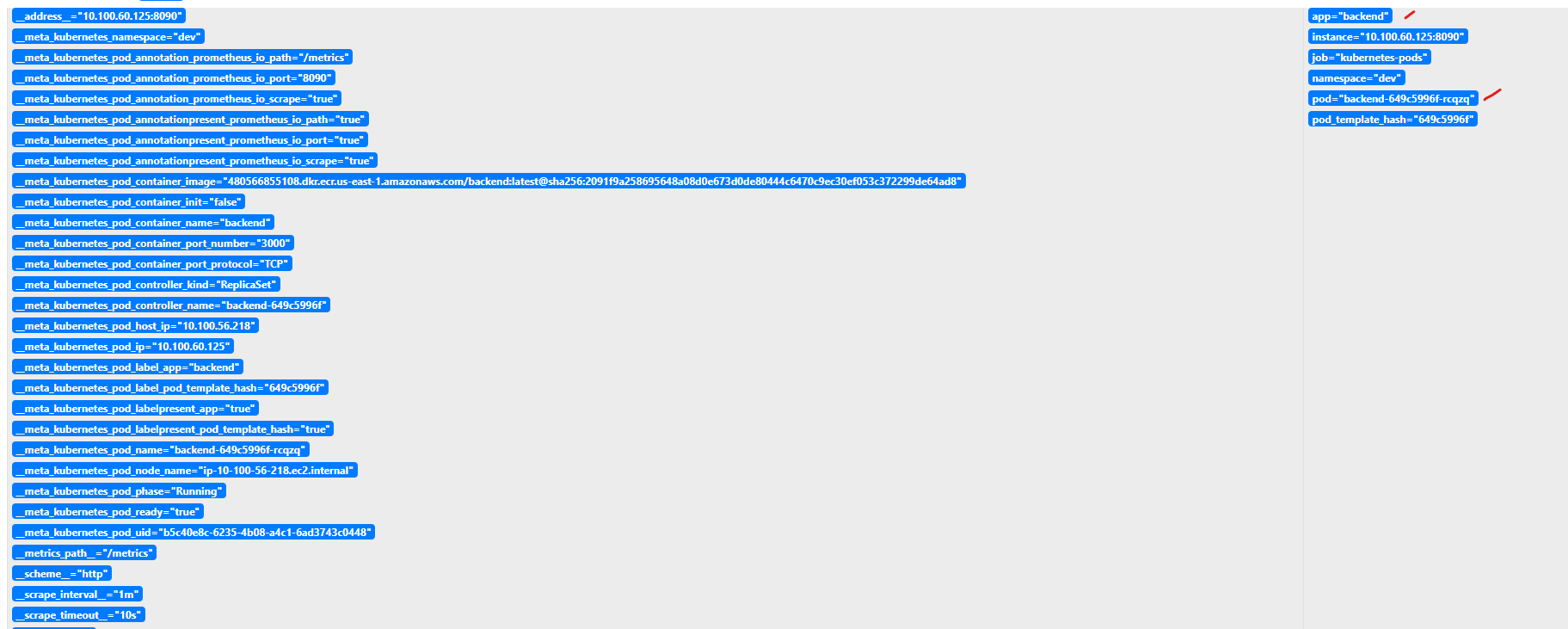
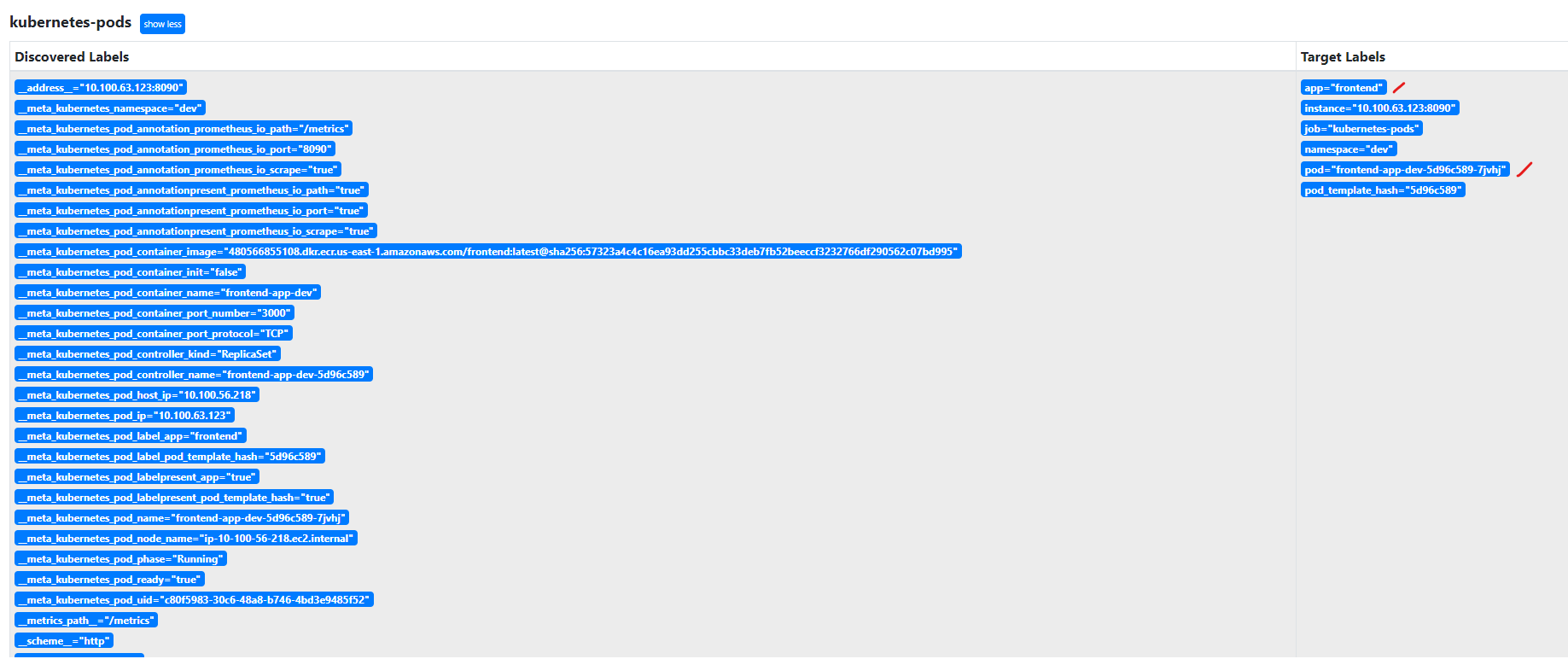
**Argocd UI**



**Ternimal**



**Prometheus UI**



**Grafana UI**

