1. **Manage K8s Application**

* Repository:
* MSA: [Felipe-2-2-5/MSA (github.com)](https://github.com/Felipe-2-2-5/MSA)
* CI/CD: [Felipe-2-2-5/sd5294\_devops\_ci\_cd (github.com)](https://github.com/Felipe-2-2-5/sd5294_devops_ci_cd)

1. **Provision AWS resources**

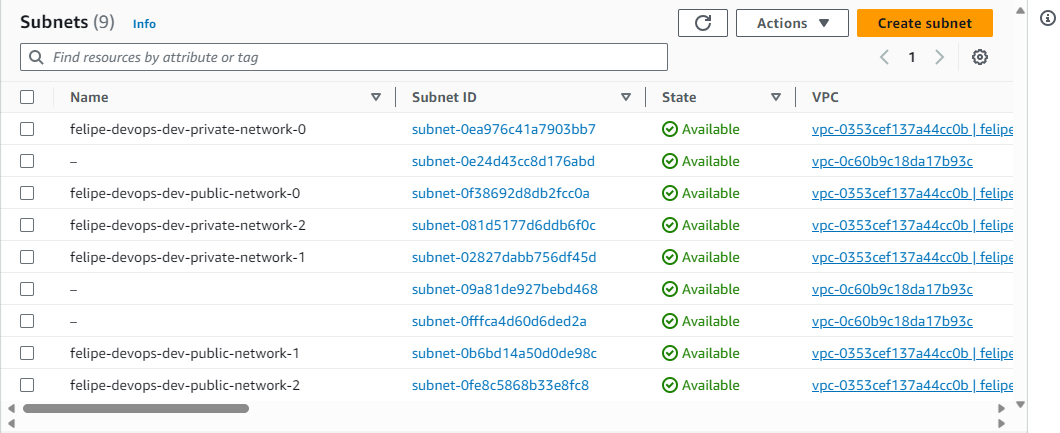
* **EK must be created in High Availability (Multi-AZs)**
* **Create your VPC and use it for worker nodes**

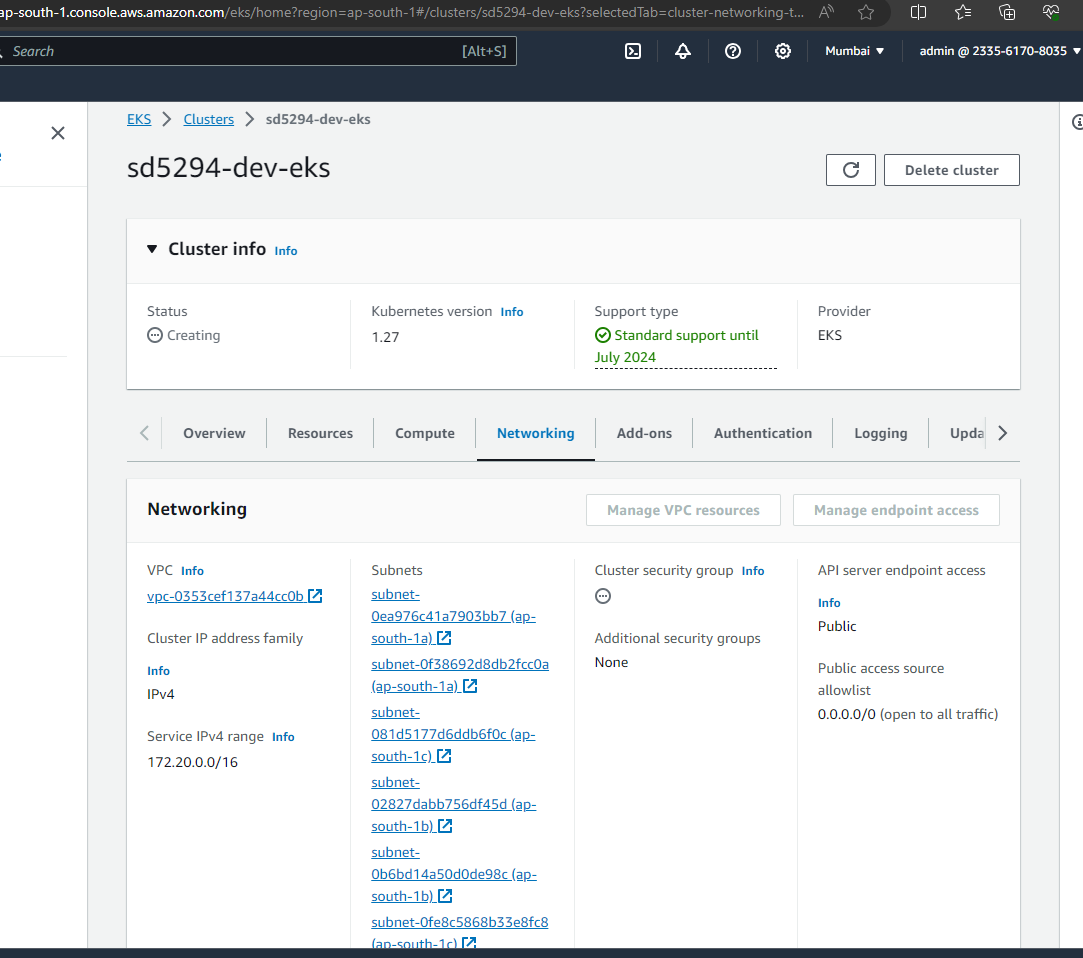
If we want the worker nodes and control plane to be in different VPCs, we should use Infrastructure as Code (IaC) tools like AWS CloudFormation or Terraform, or we can use the AWS CLI. At this point, we need to establish VPC peering and set up an AWS Transit Gateway so that the control plane can communicate with the worker nodes.

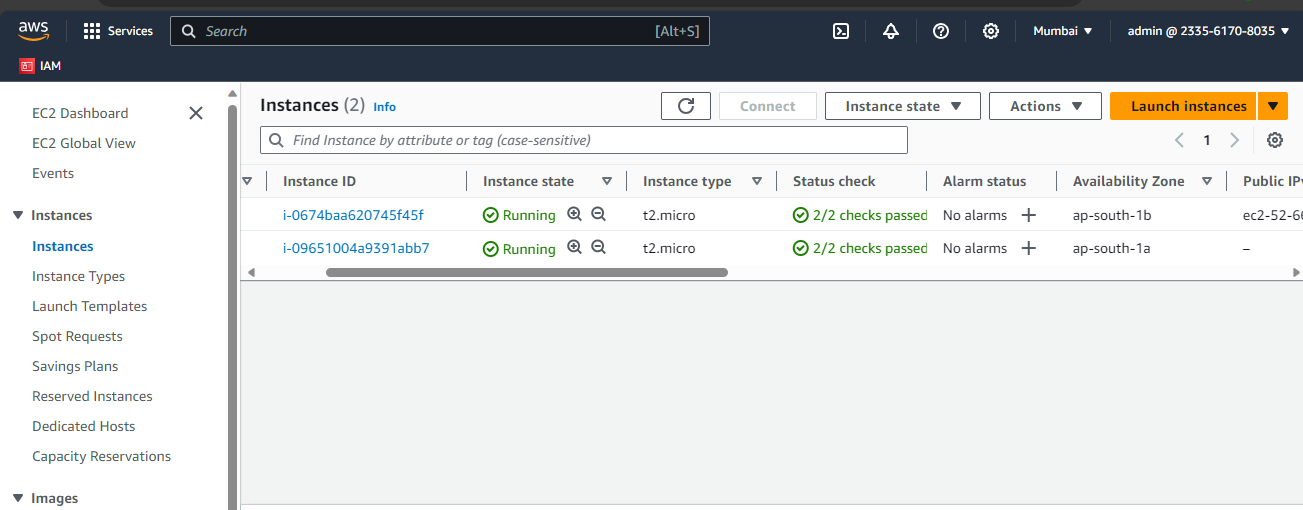
Clone Infrastructure code and deploy aws resouce using terraform (VPC, EC2, EKS)

* Terraform init
* Terraform Plan
* Terraform Apply

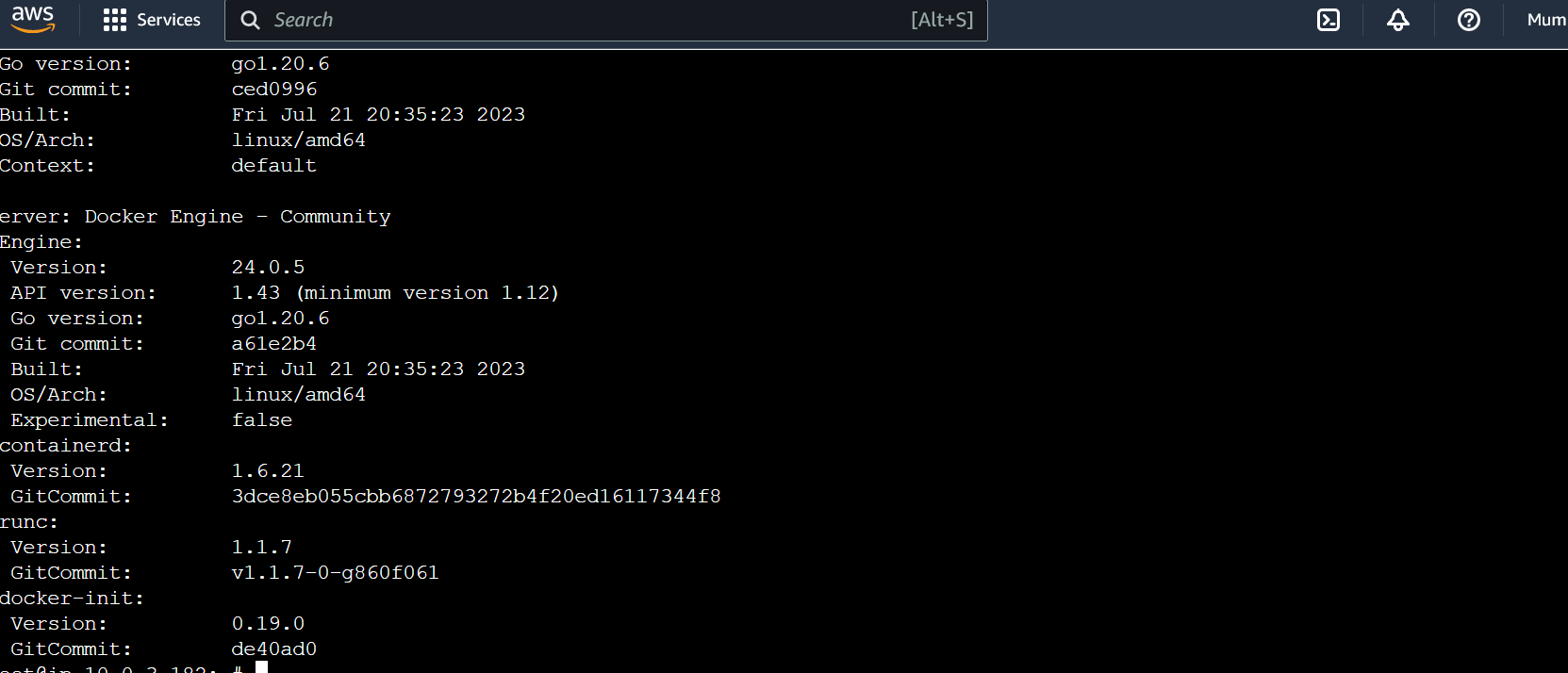
EKS workers nodes via multiple availability zones, cluster use custom VPC instead of default VPC

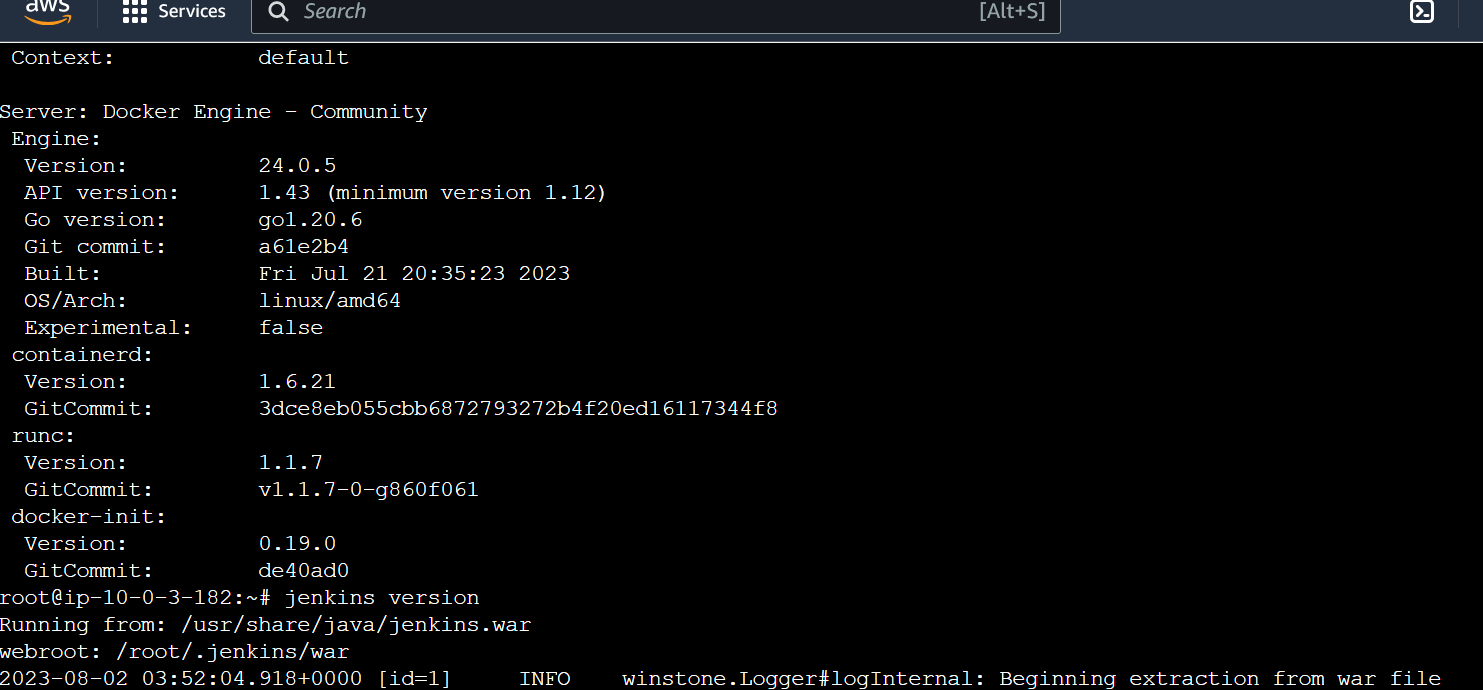


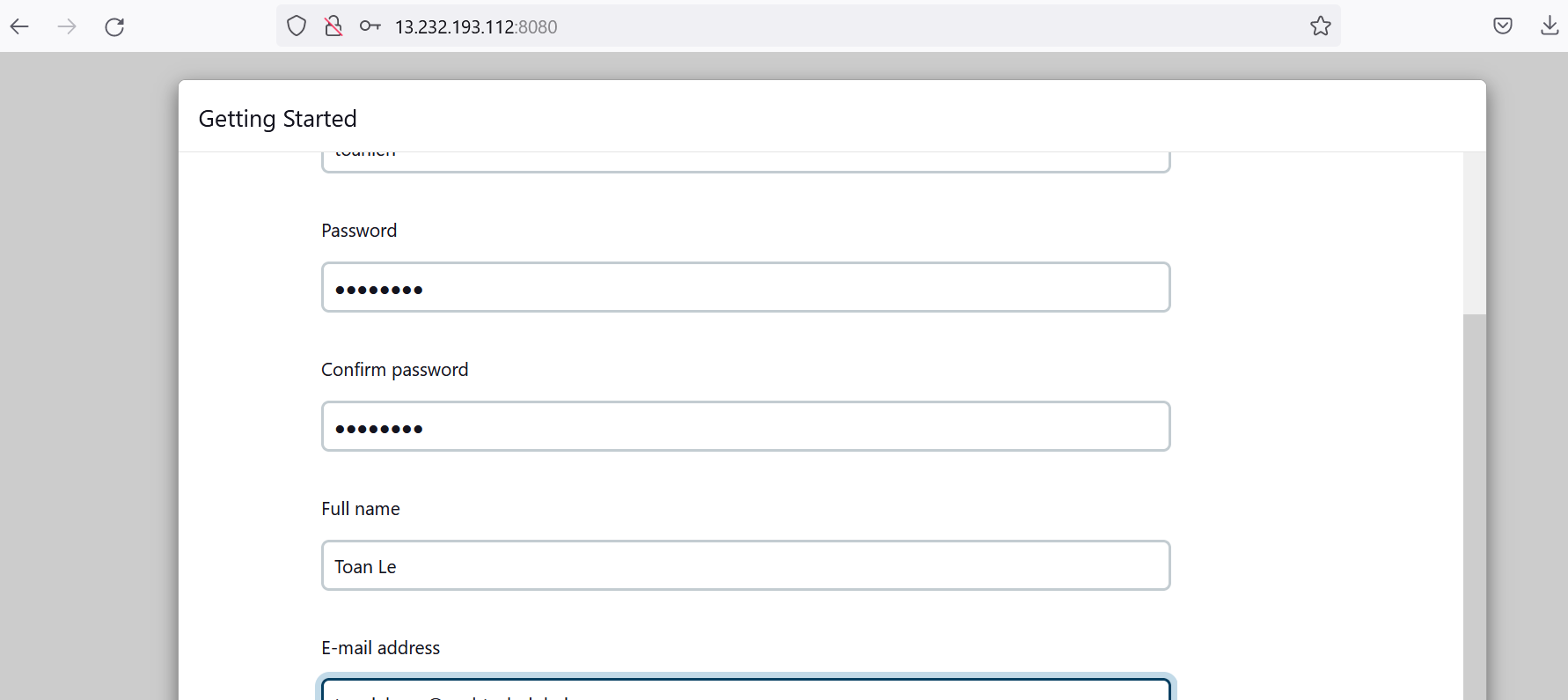




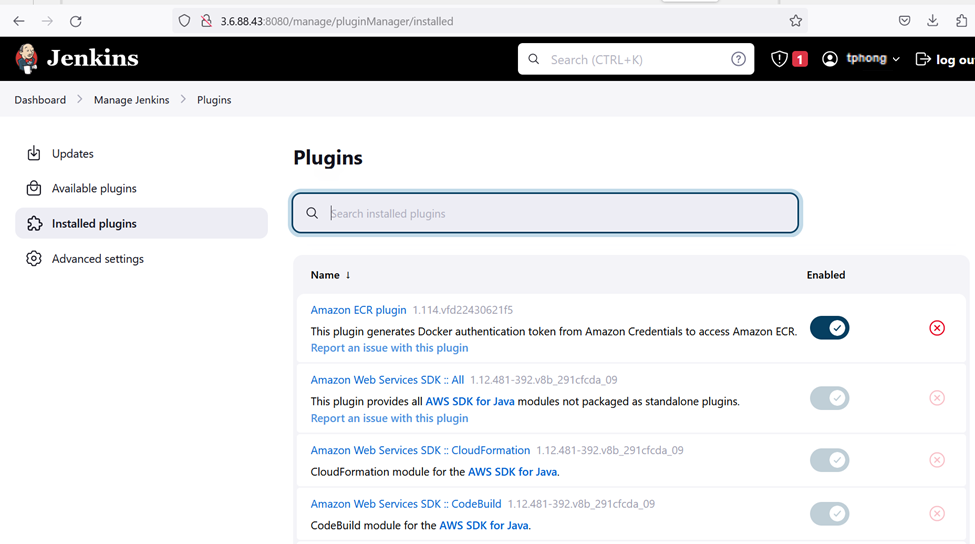
1. **Installation**

* **Install Docker and Jenkins servers on EC2**





Follow prerequisite from <https://github.com/nashtech-garage/devops-ci-cd> to add require plugins and settings

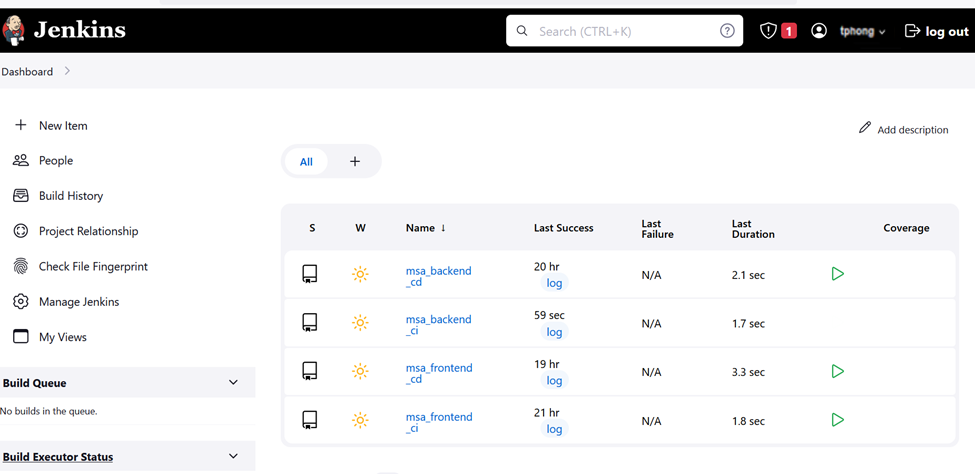
****

****

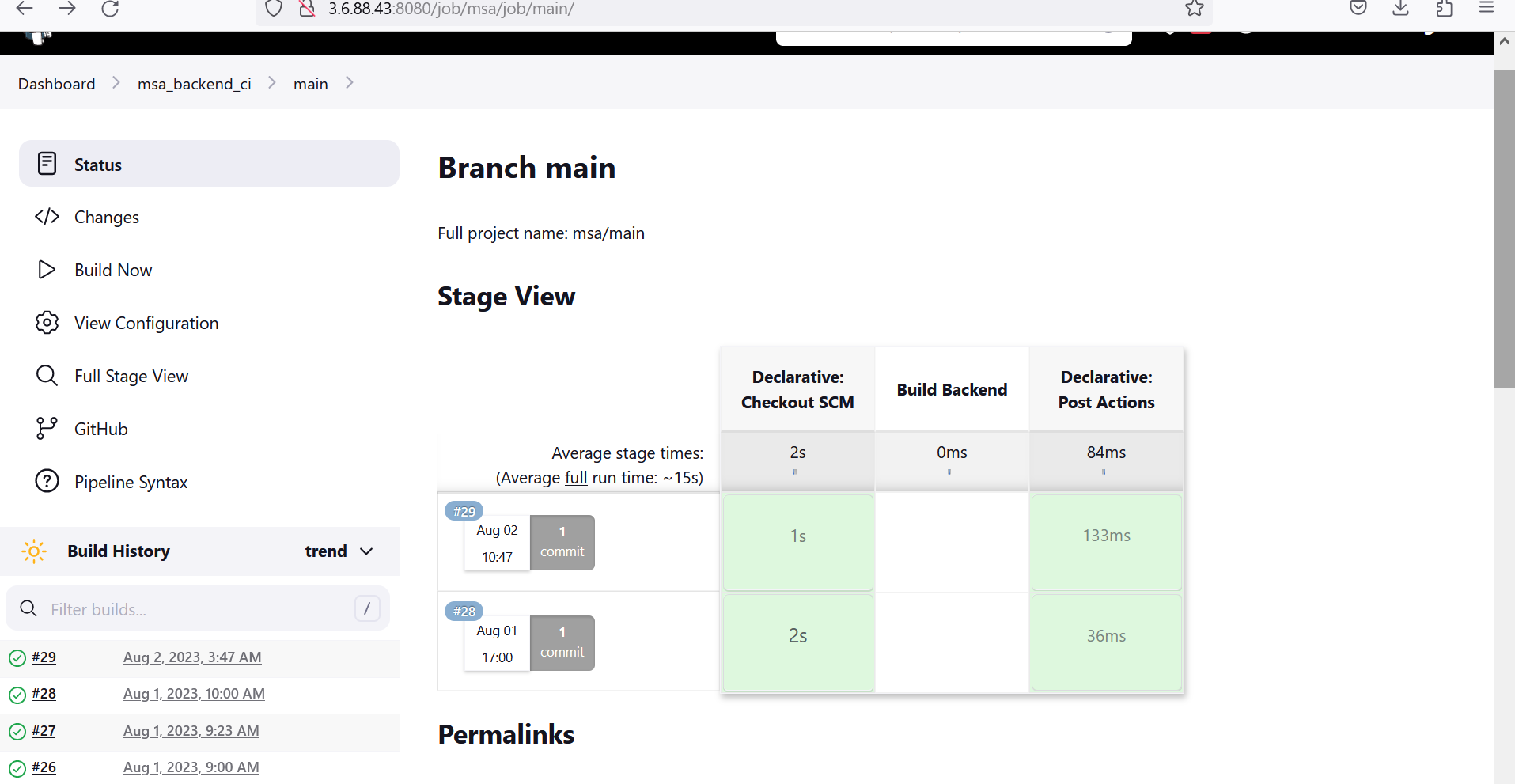
1. **Setup Jenkins pipeline for CI**

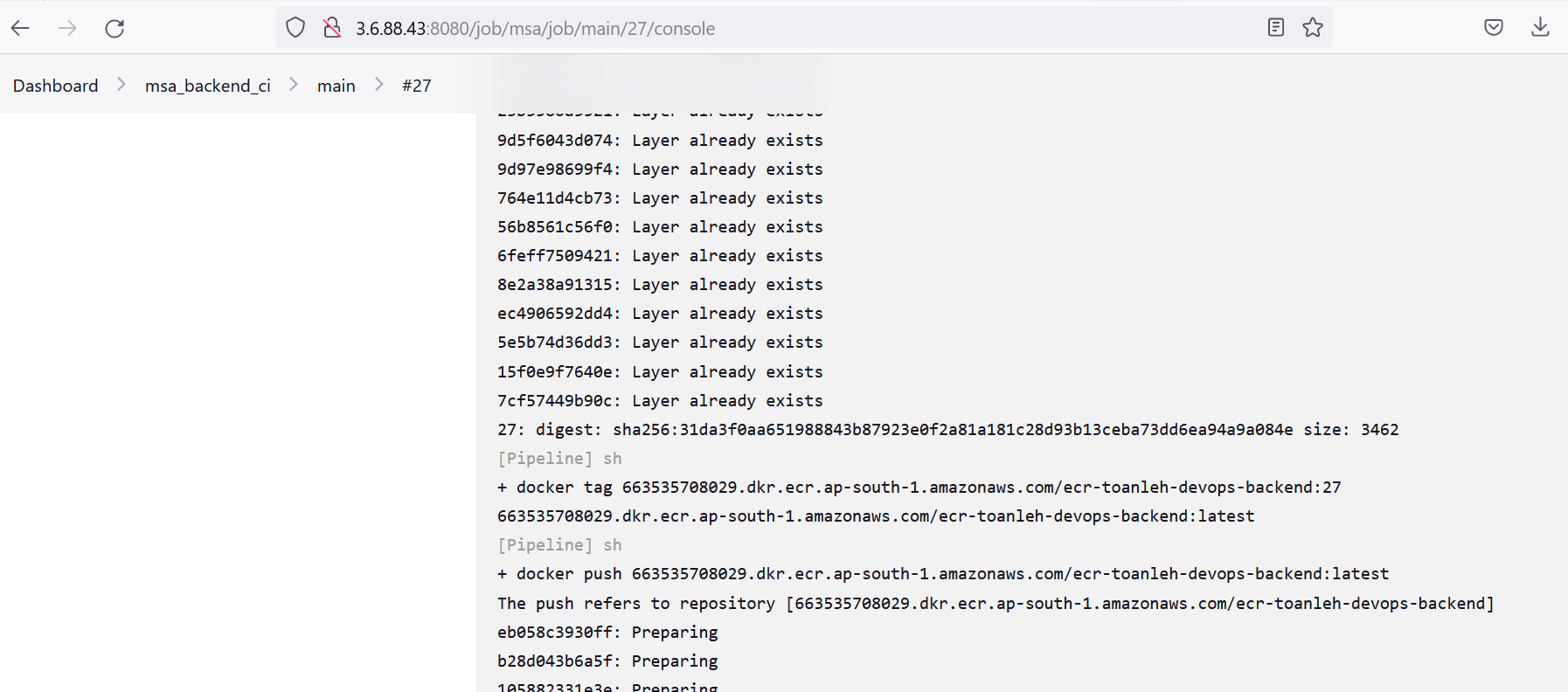
* **Use Trivy in Jenkins CI pipeline**

Set up CI/CD for backend and frontend

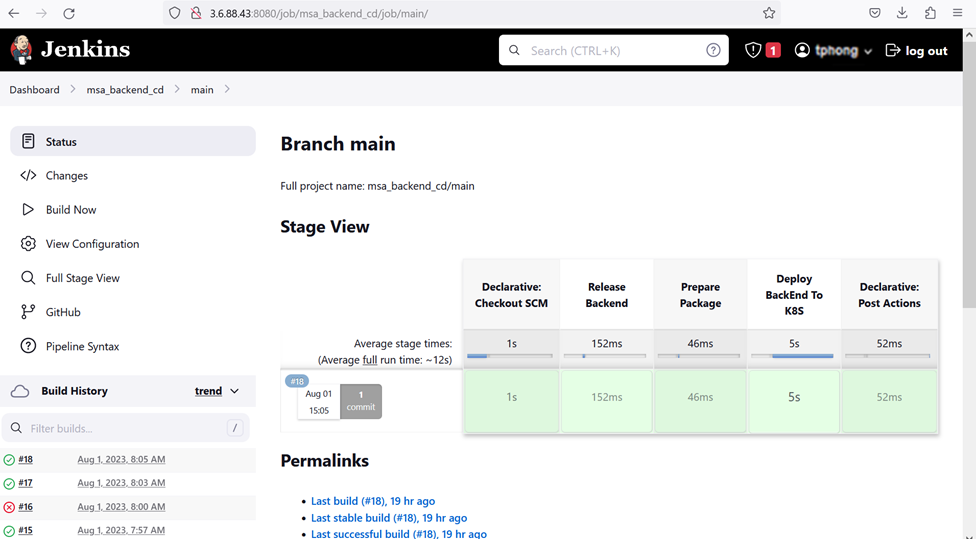
****

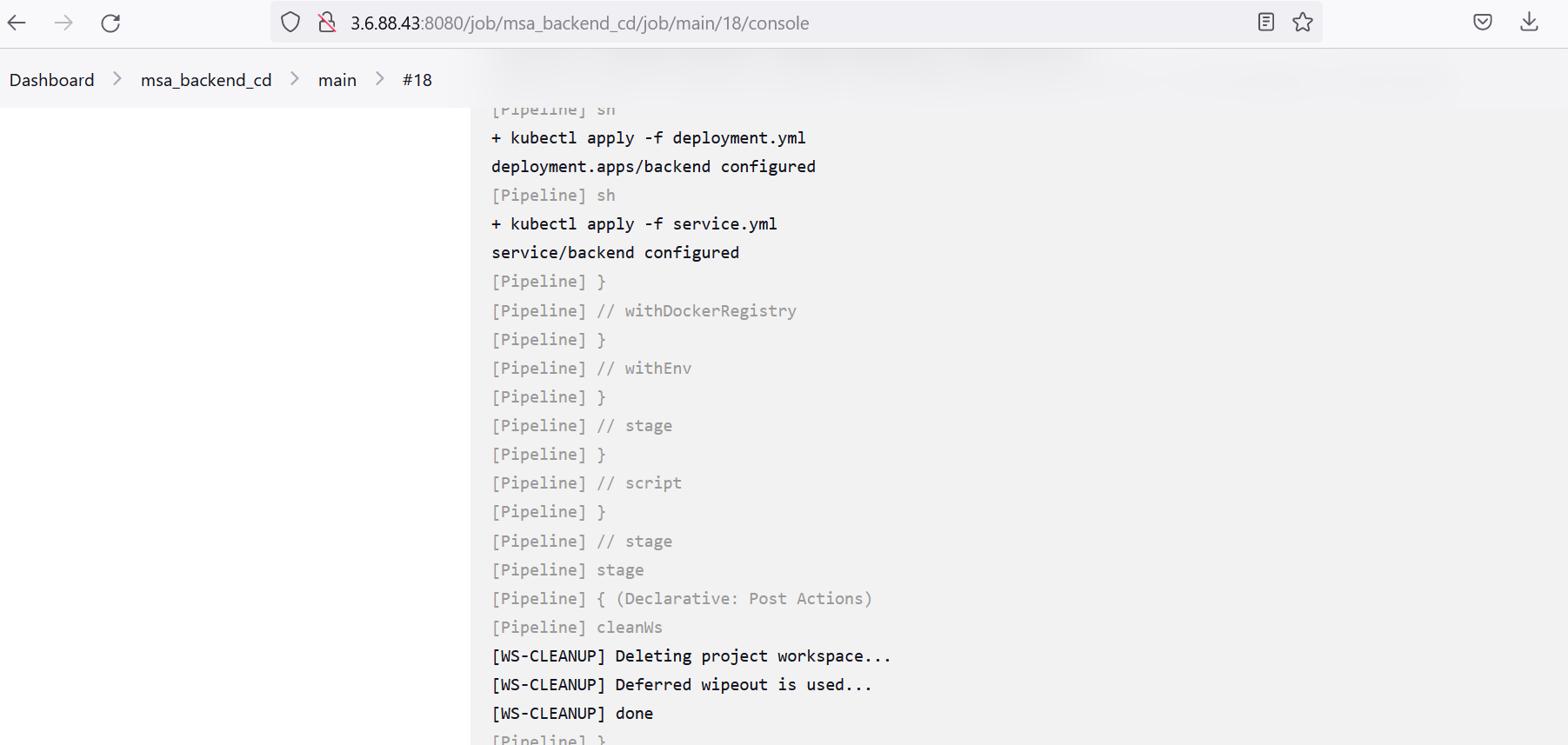
Backend CI



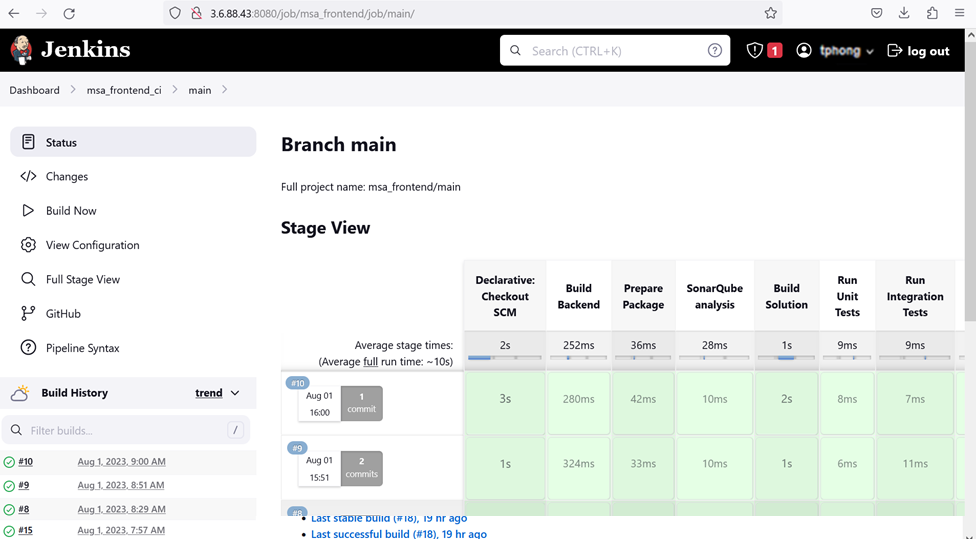


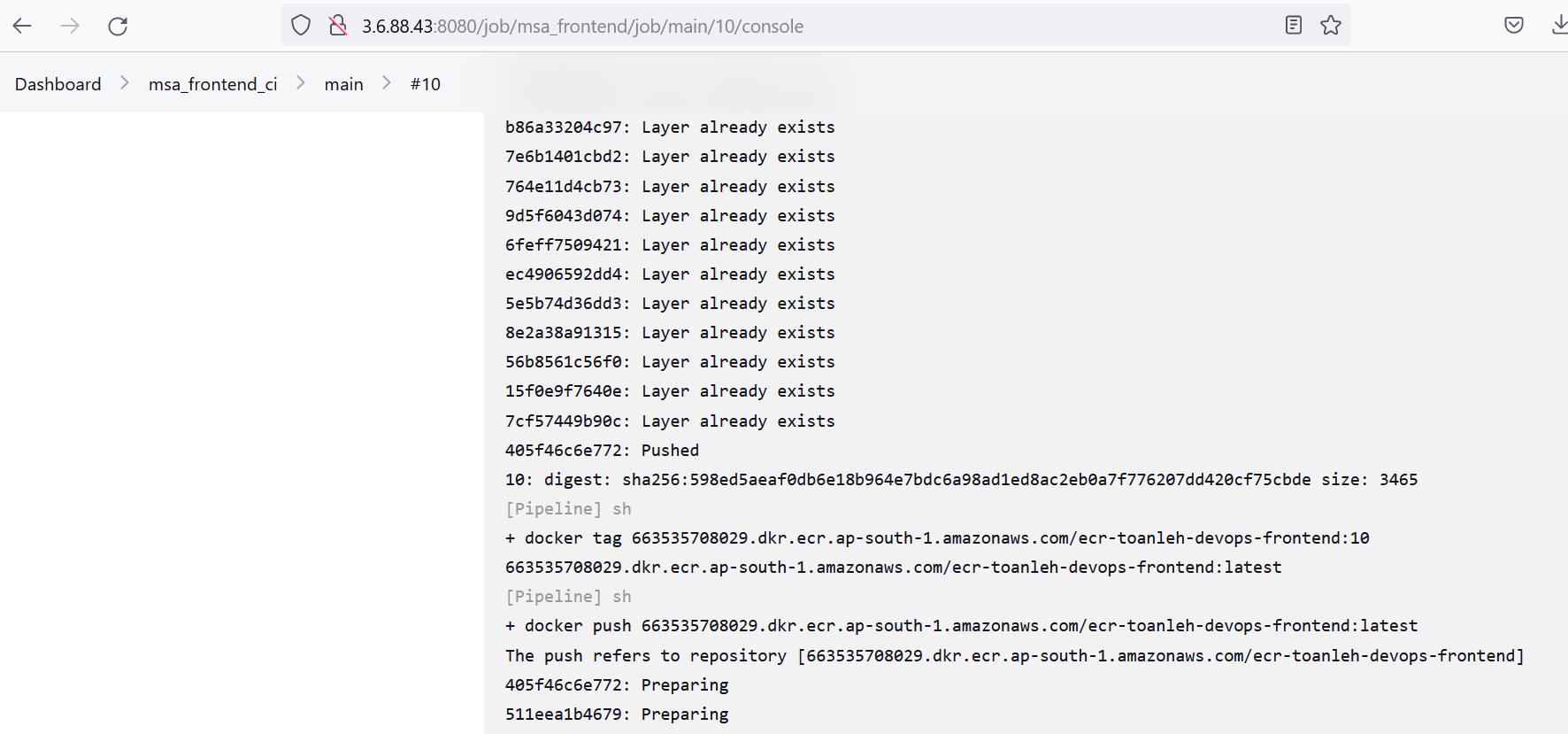
Backend CD



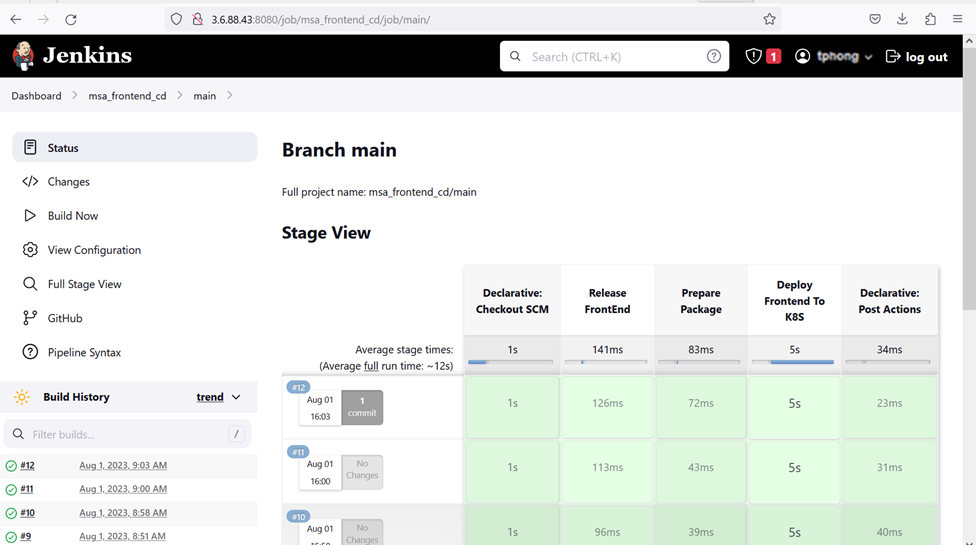


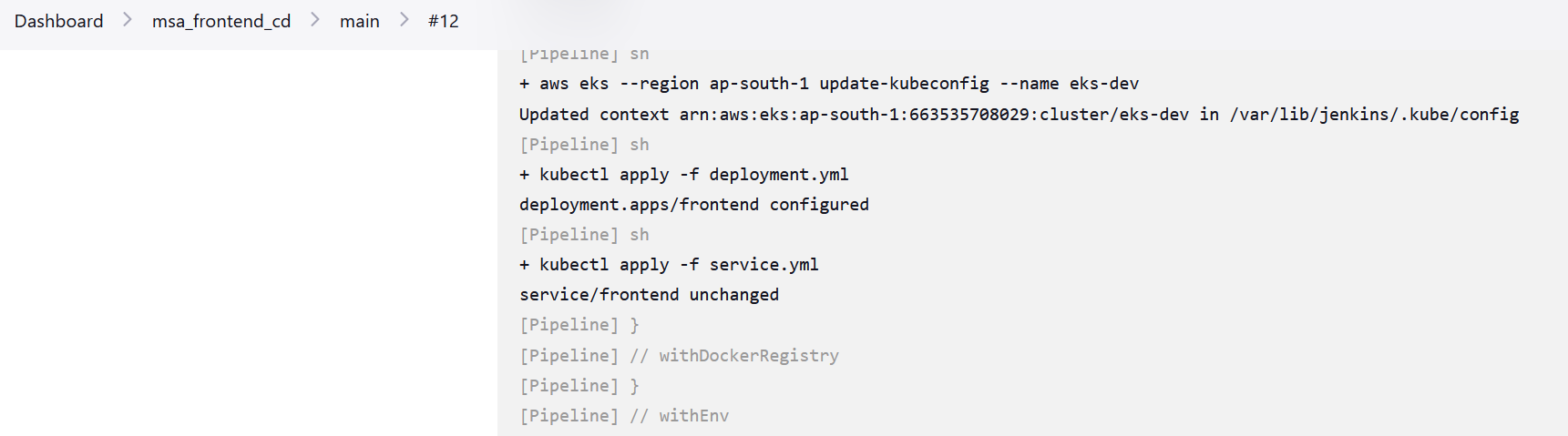
FrontEnd CI

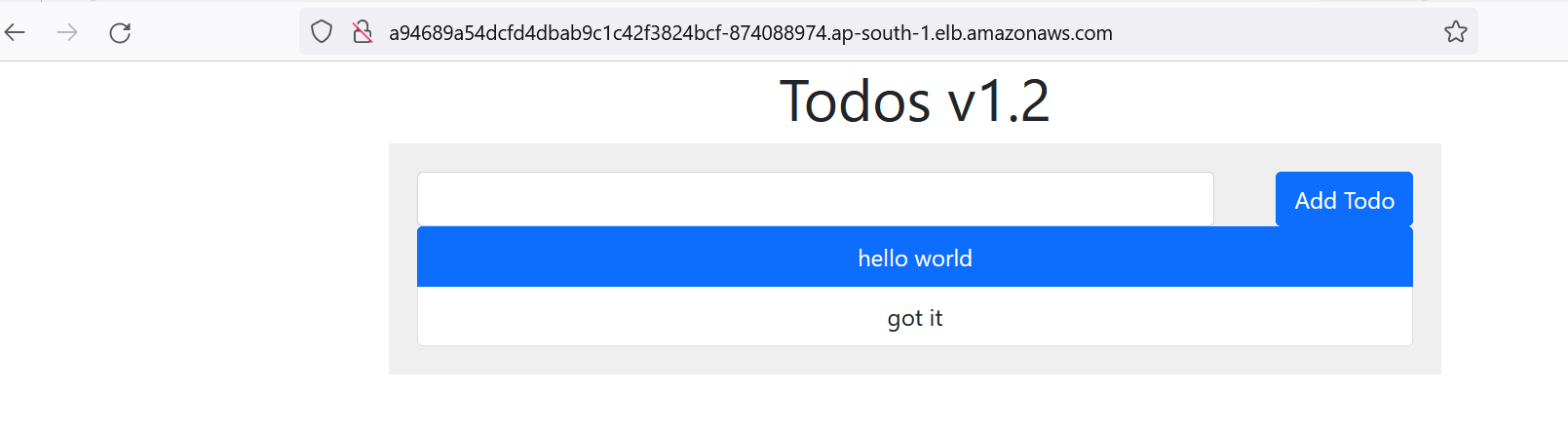




FrontEnd CD

* 



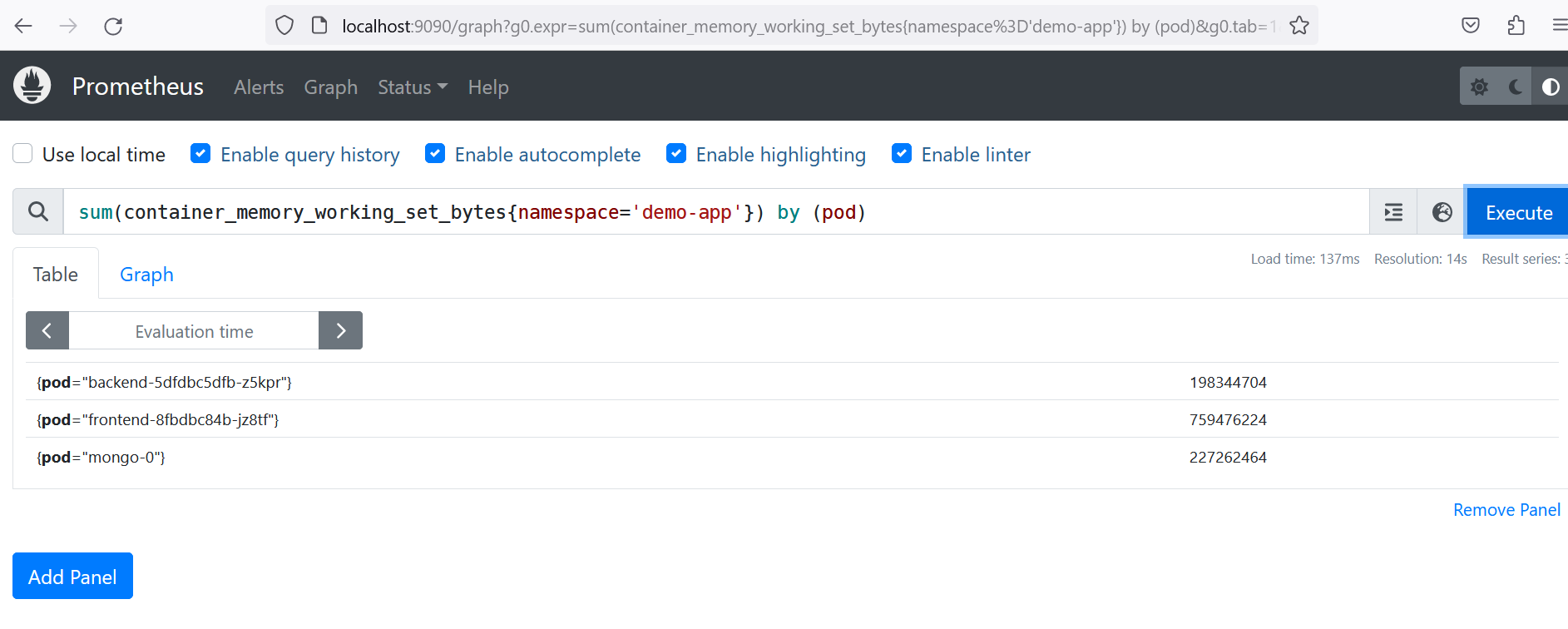


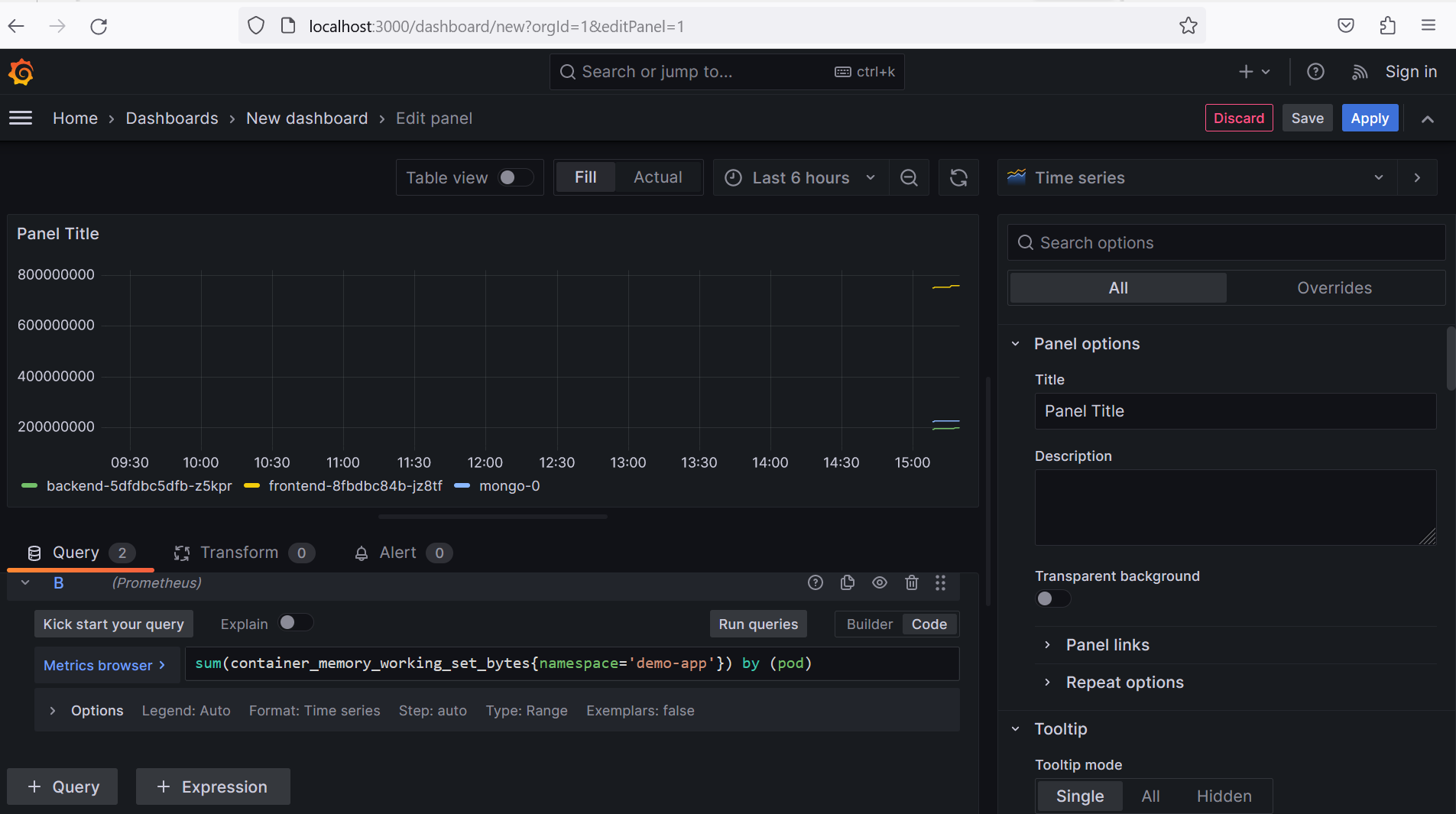
1. **Monitoring**

Use Istio to provide observability metrics, which can be visualized and collected using Grafana and Prometheus.

Install Istio then use istio-ingress-gateway to load balance traffic to our frontend todo app

Install Prometheus and Grafana, ensure those dashboard can visualize our pod (frontend, backend, mongo) traffic





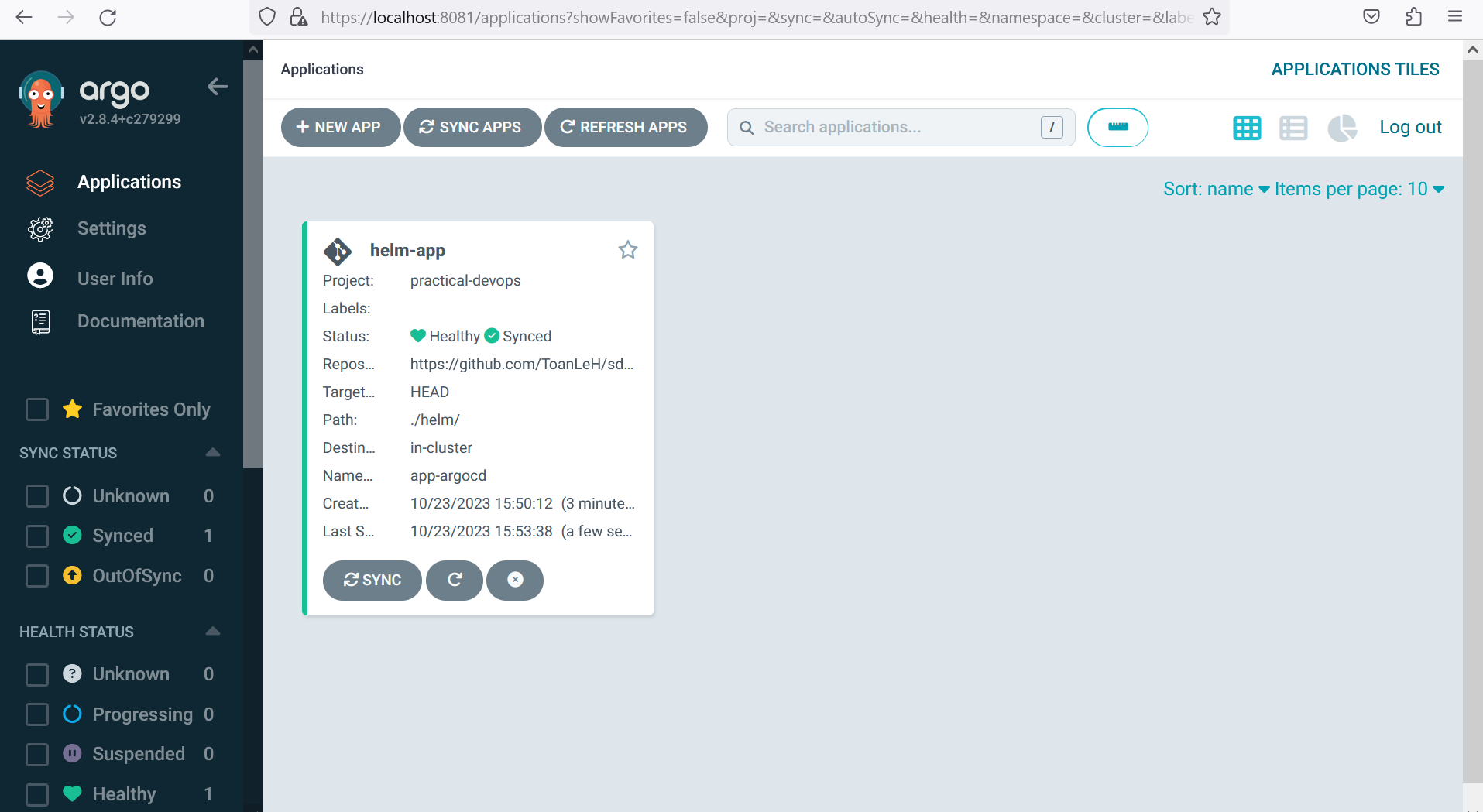
1. **Use GitOps for the CD pipeline**

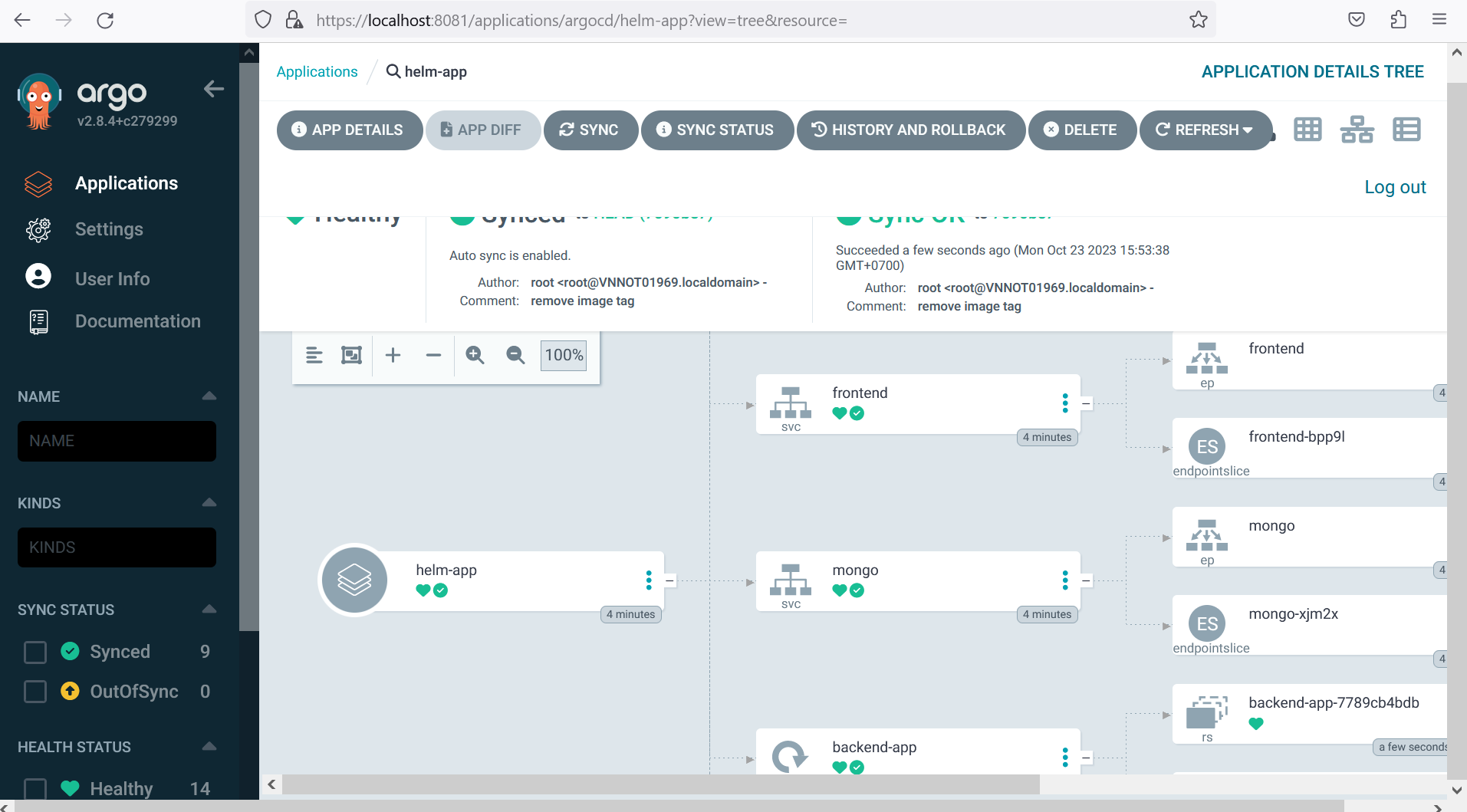
Install ArgoCD and portforward to access UI

A picture containing text, screenshot, software, multimedia software

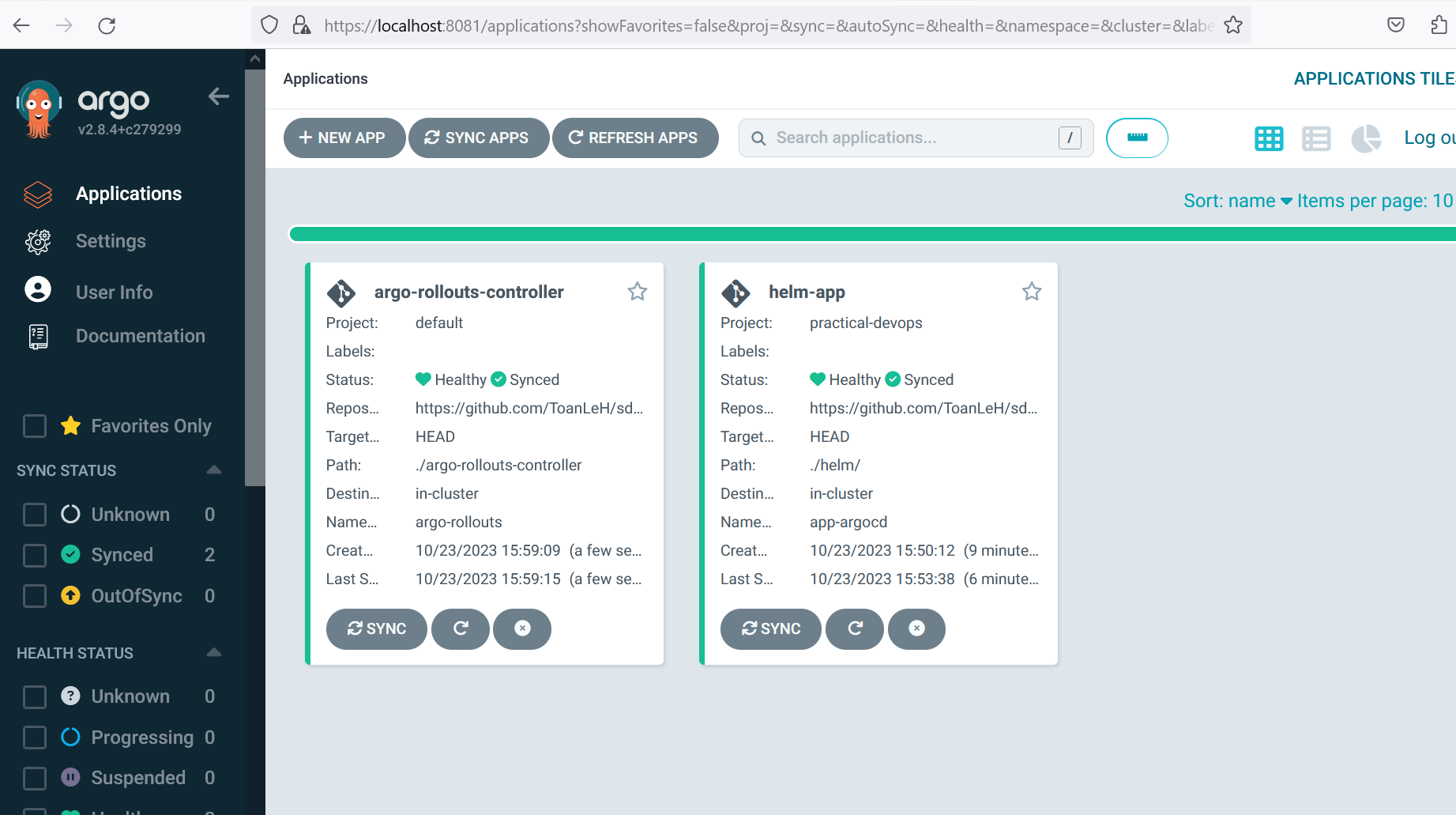
Description automatically generated

Login to ArgoCD and run command to create the helm application

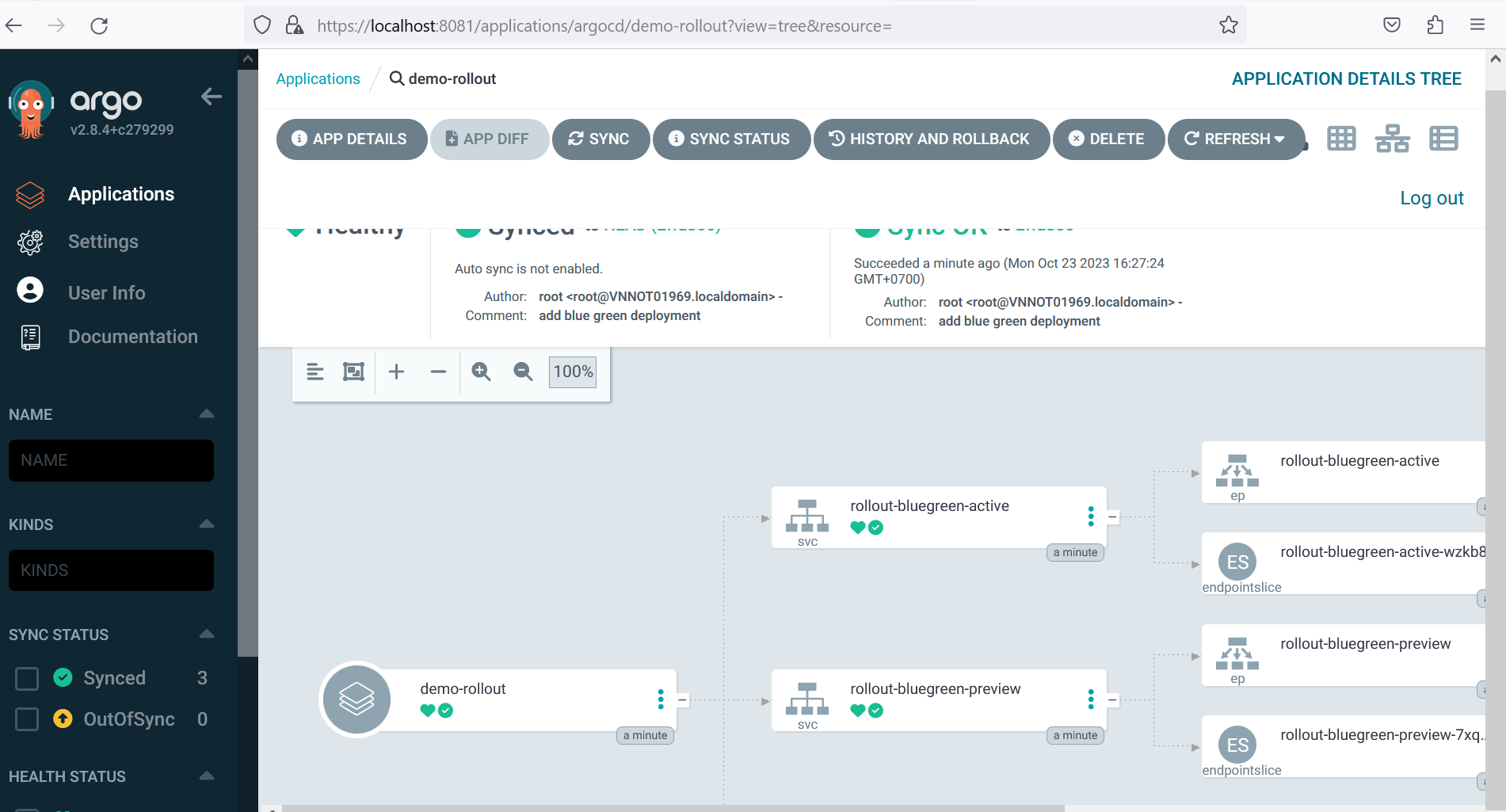




Install the Argo Rollouts Controller



Create first deployment and port forward to see the blue version of our app

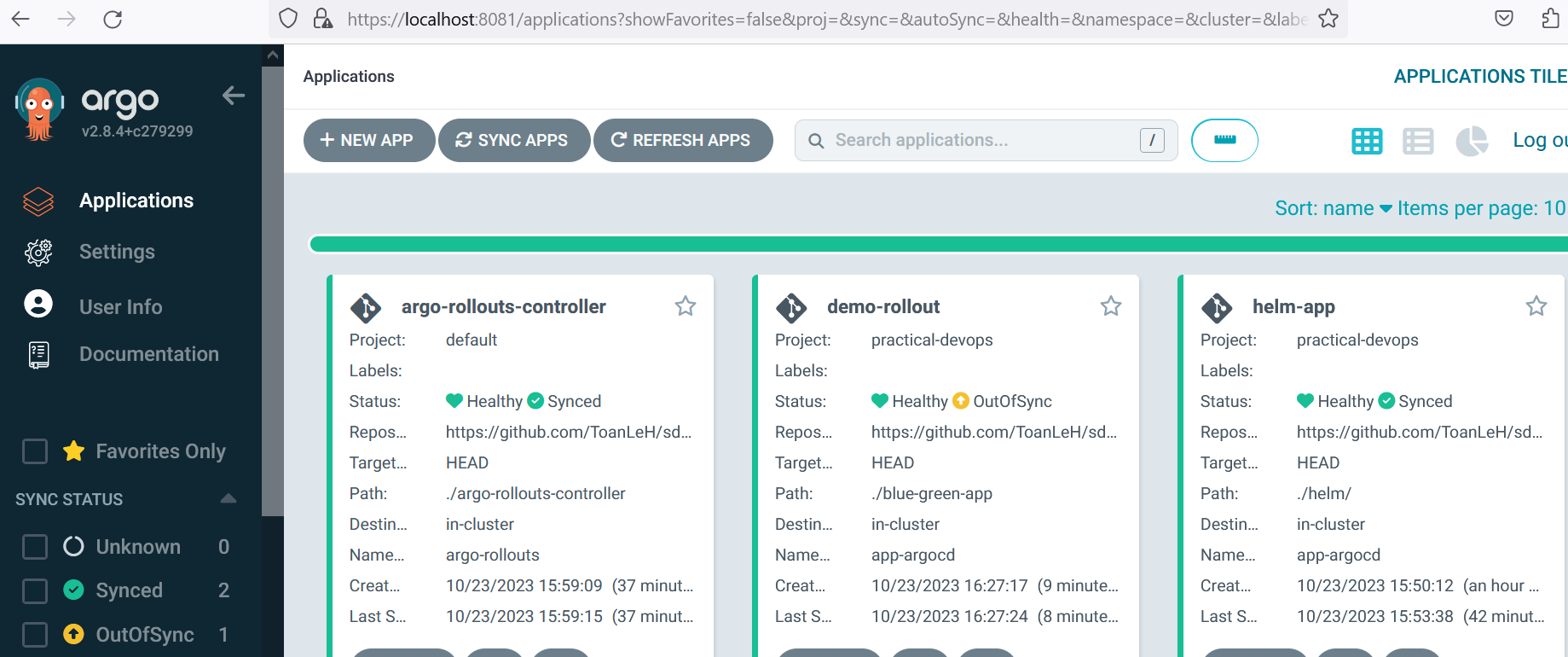




**Blue/Green deployments**

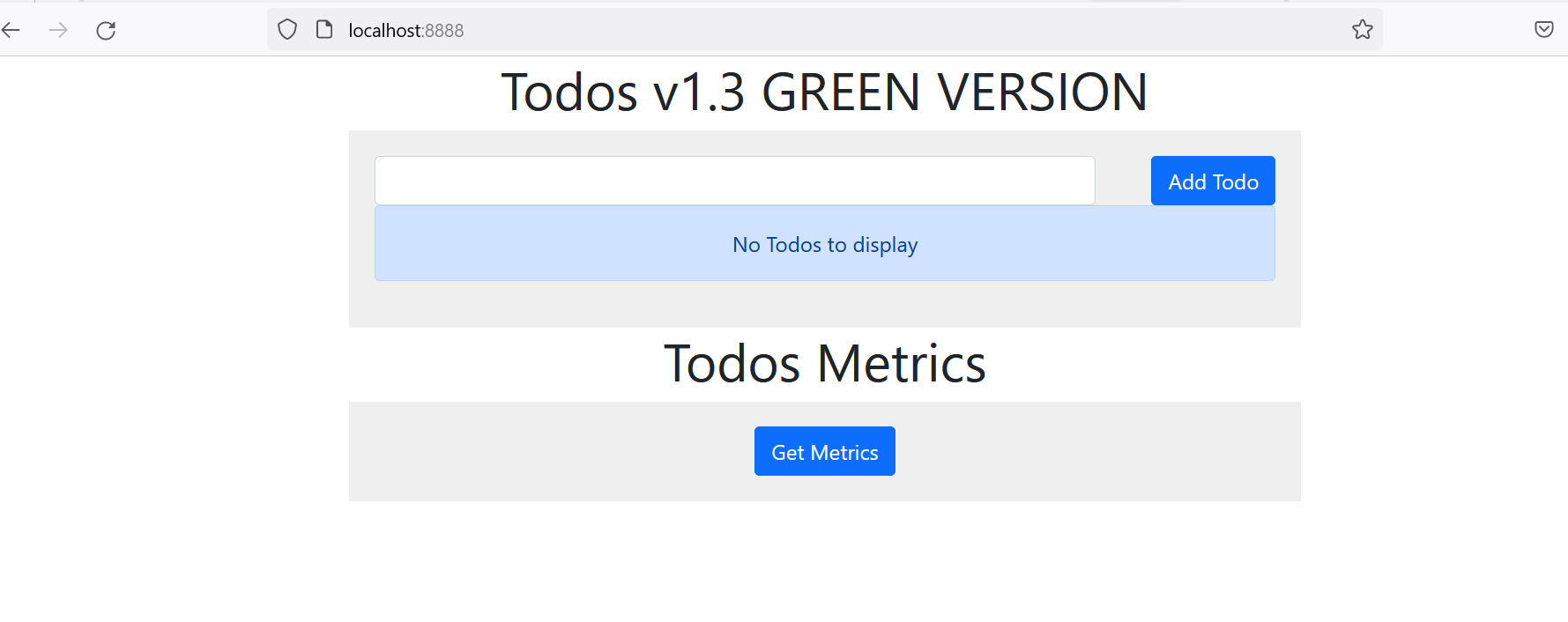
We are now ready to have a blue/Green deployment with the next version. Change the container image of the rollout to the next version with:

Image: 233561708035.dkr.ecr.ap-south-1.amazonaws.com/ecr-sd5294-devops-frontend:green



After you change the image, the following things happen.

* Argo Rollouts creates another replicaset with the new version
* The old version is still there and gets live/active traffic
* ArgoCD will mark the application as out-of-sync
* ArgoCD will also mark the health of the application as "suspended" because we have setup the new color to wait.



Notice that we used the autoPromotionEnabled: true property in the definition of the rollout.