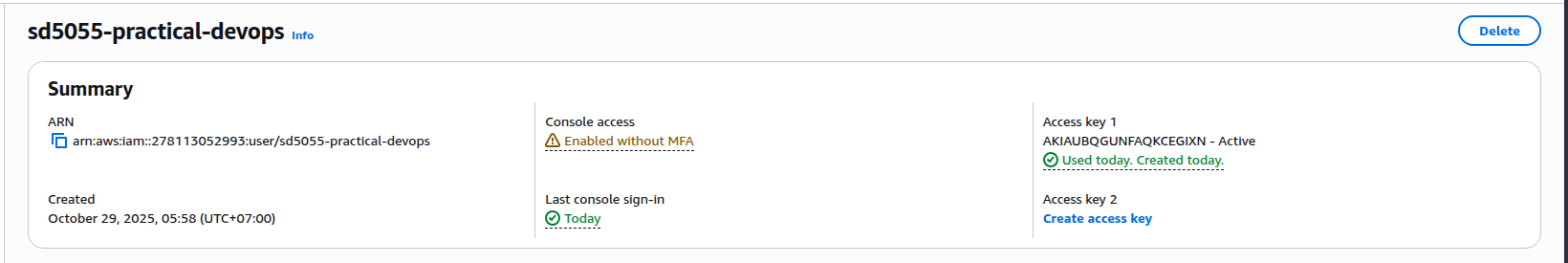
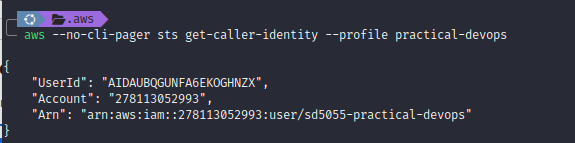
Practical DevOps with AWS infrastructure

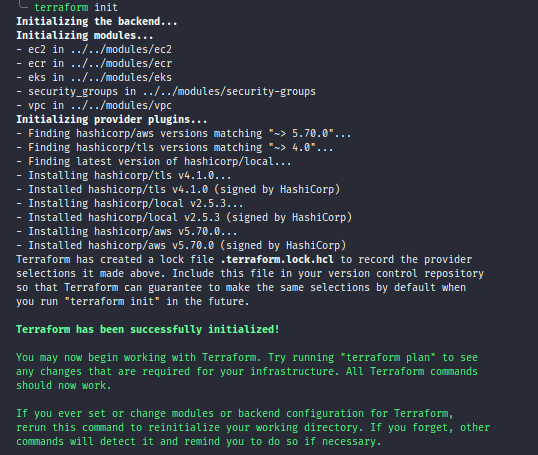
## Create AWS infrastructure with Terraform

Build an AWS infrastructure with Terraform through this repo: <https://github.com/tuankiet1709/sd5055_aws_infrastructure>

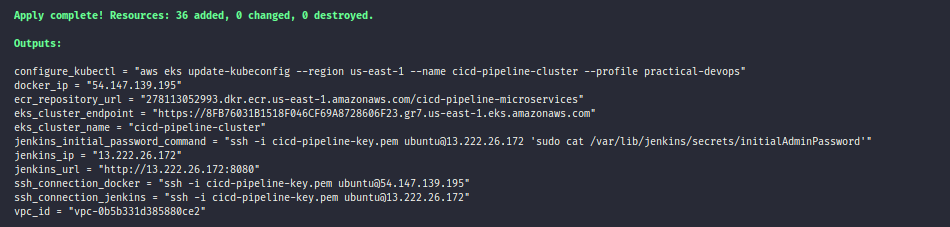
* First, I created IAM user for practical devops traininig (SD5055-practical-devops)
* Then, set IAM user to aws cli local with access key and secret access key



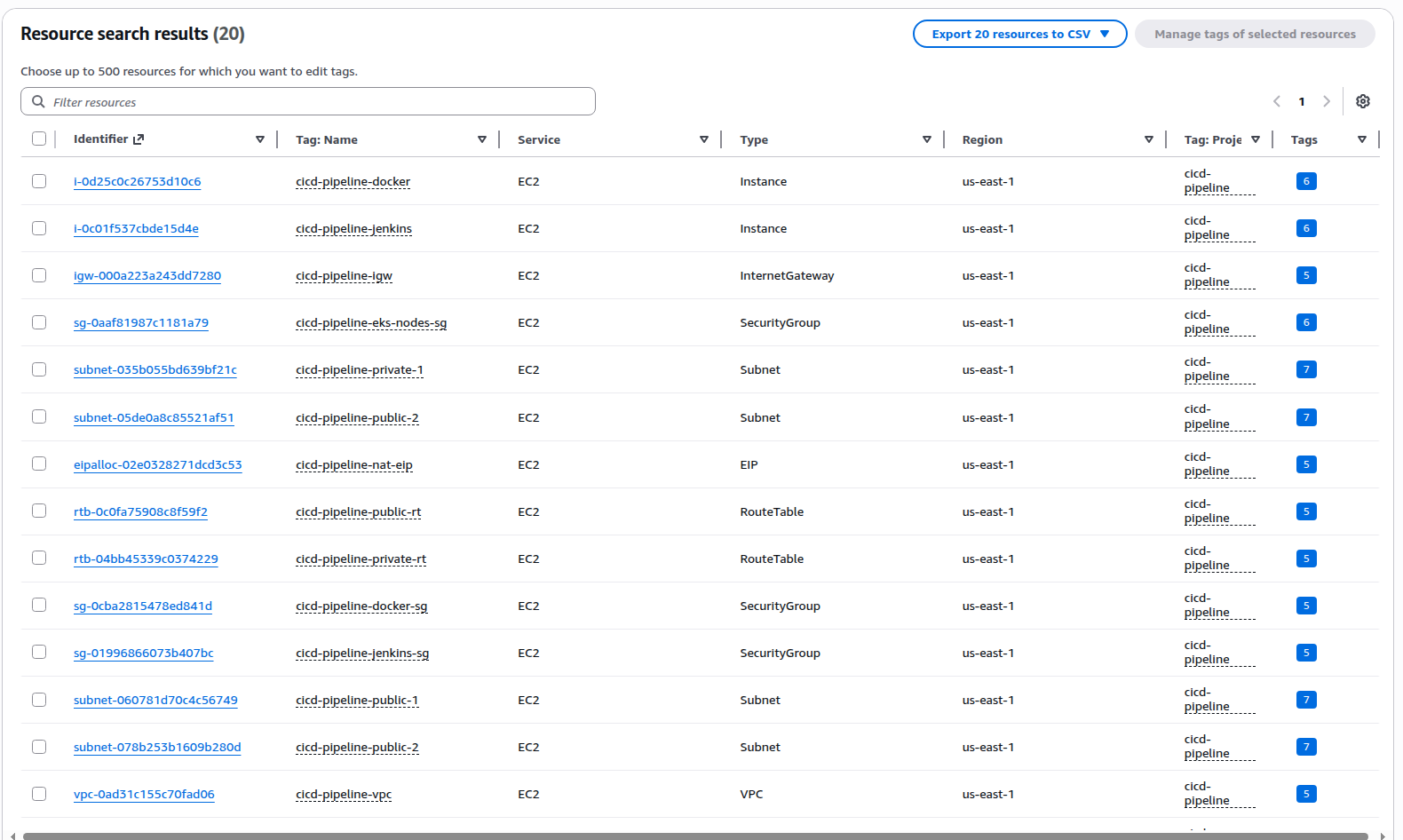
* We will need to run `terraform init` to init the terraform infrastructure



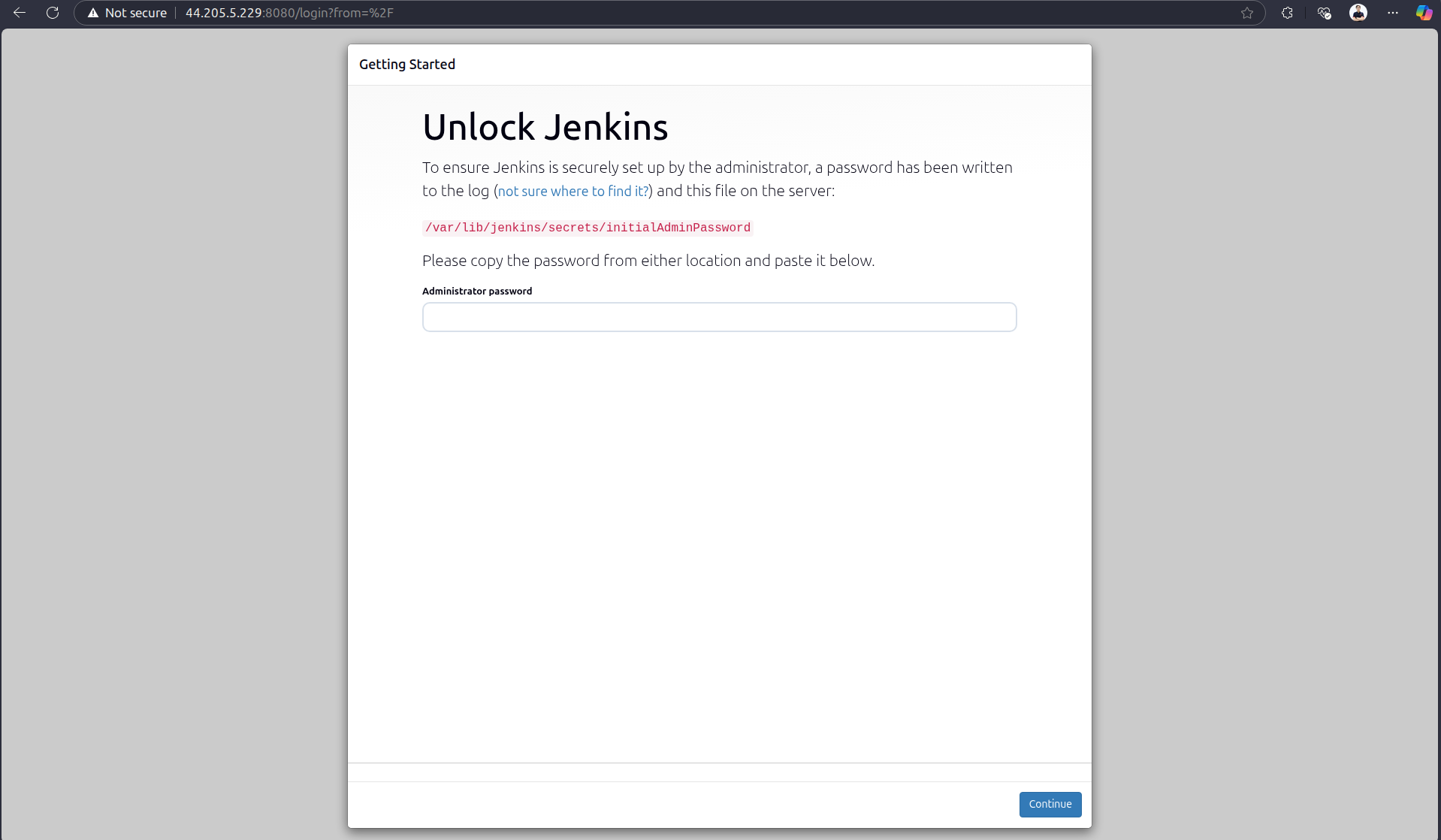
* After initializing terraform successfully, we can use `terraform apply` to create the aws services that we defined in terraform



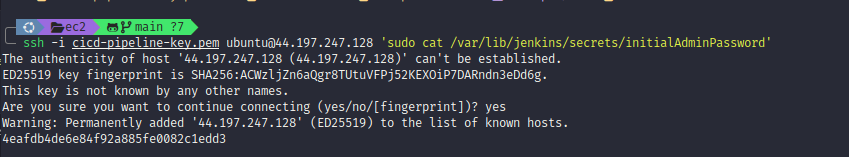
* After applying terraform successfully, we can see all resources we defined on AWS



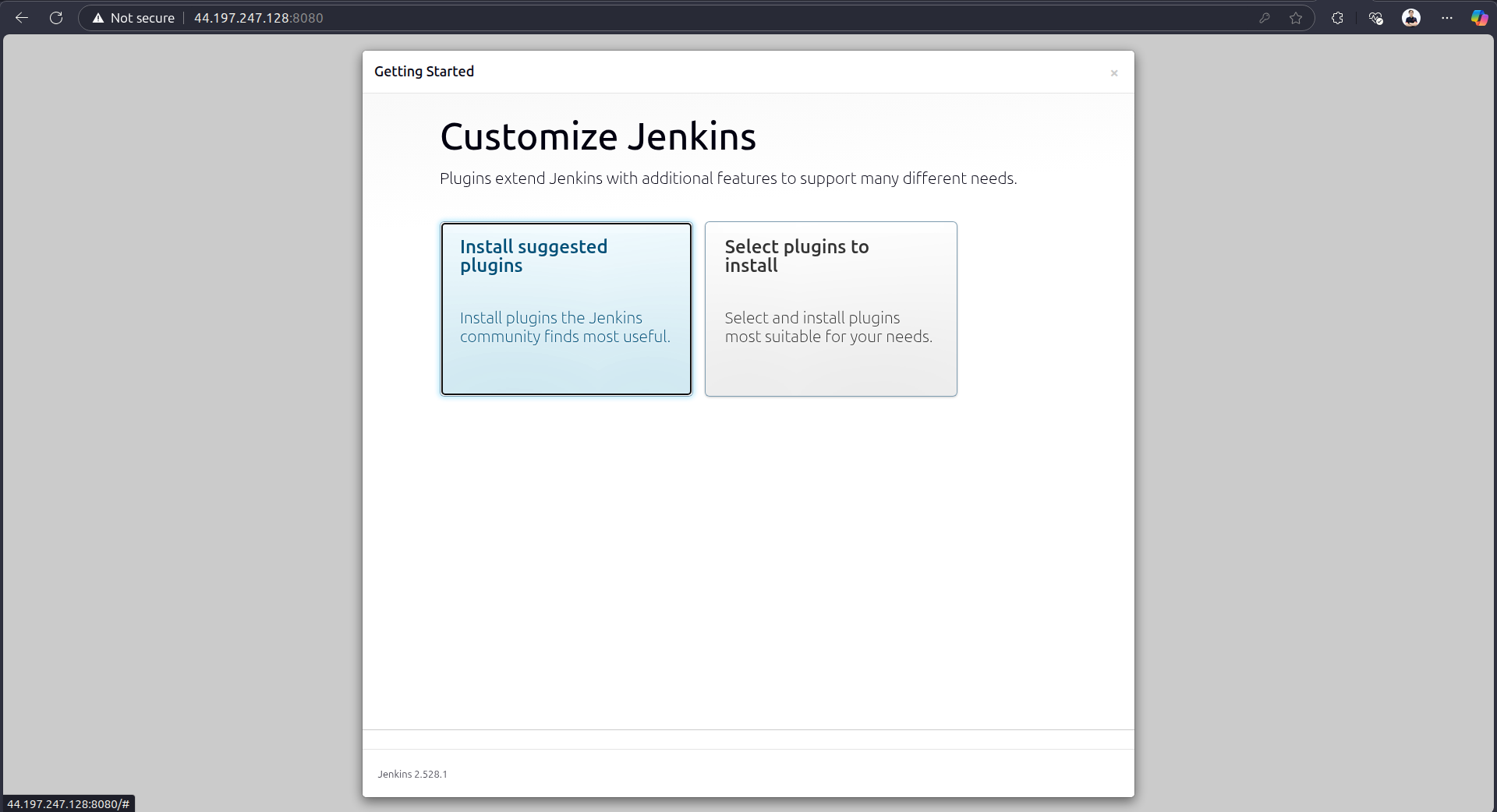
* With terraform, we already created a ec2 with jenkins inside. Now we can access to jenkins



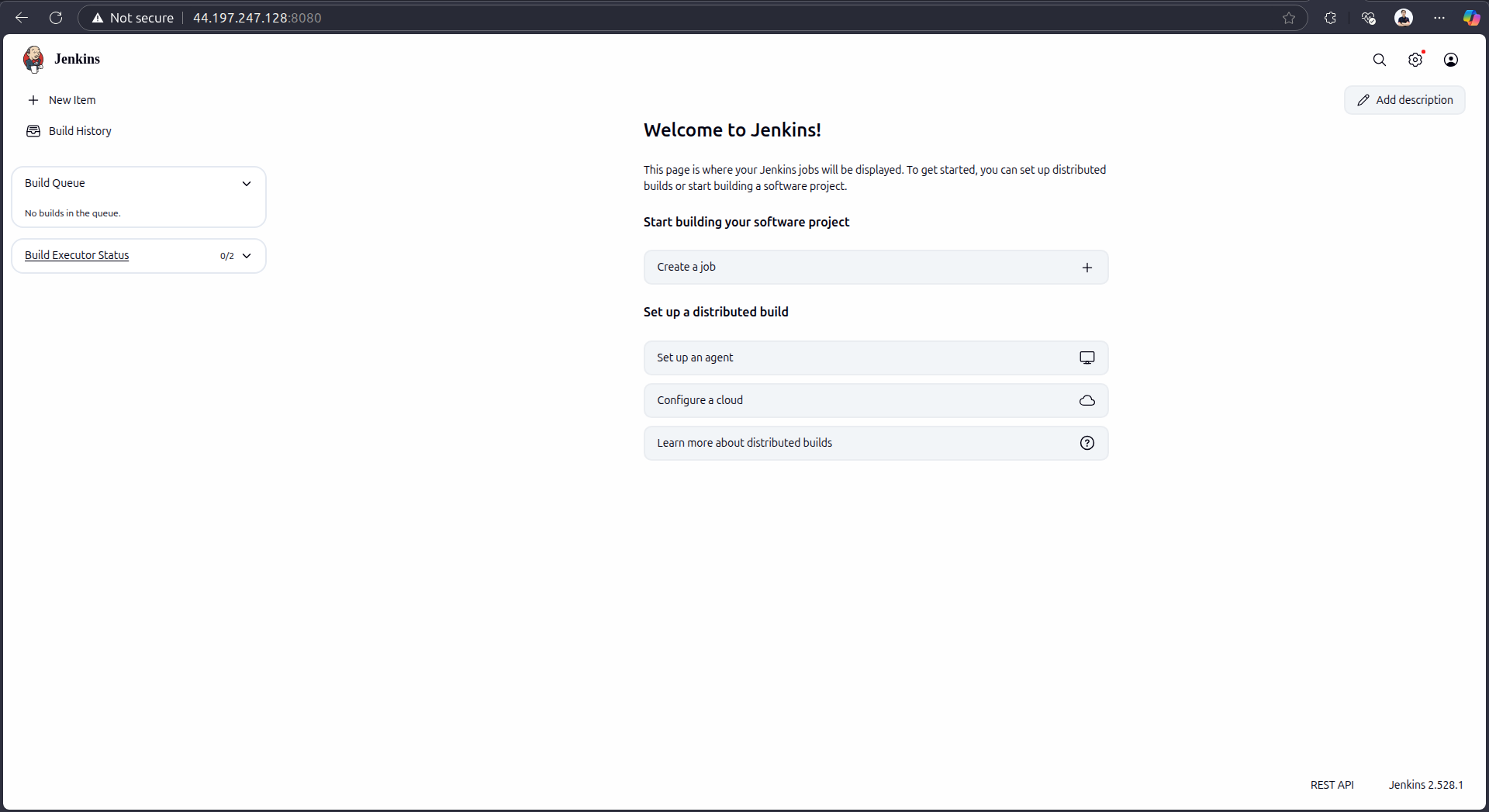
* For the first running, we need to get initialAdminPassword through the path jenkins provided



* Then we can login to the jenkins

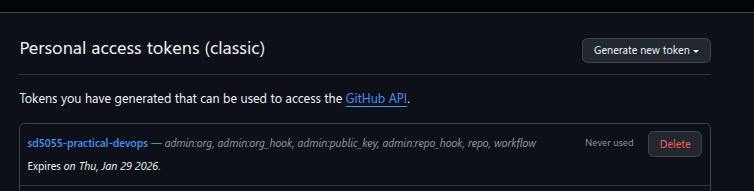
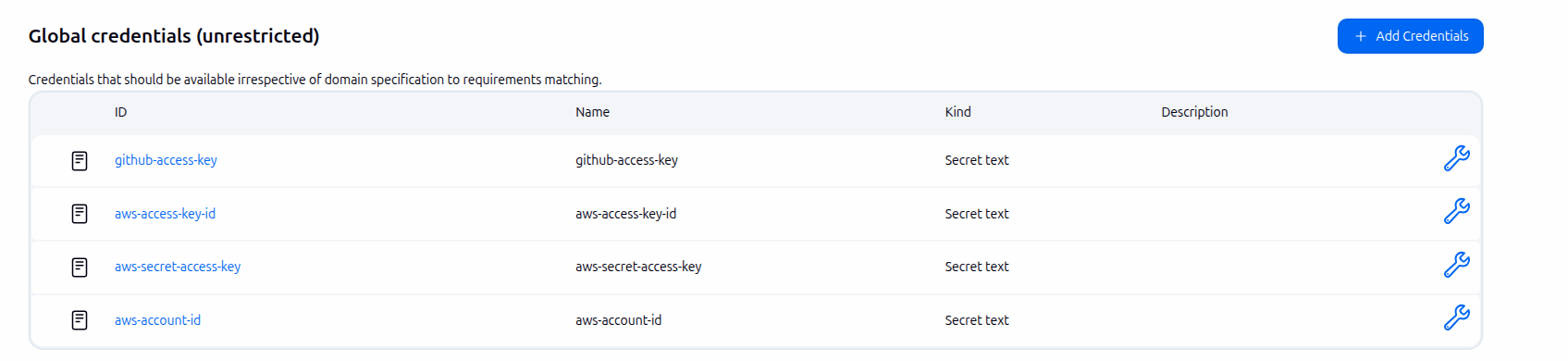
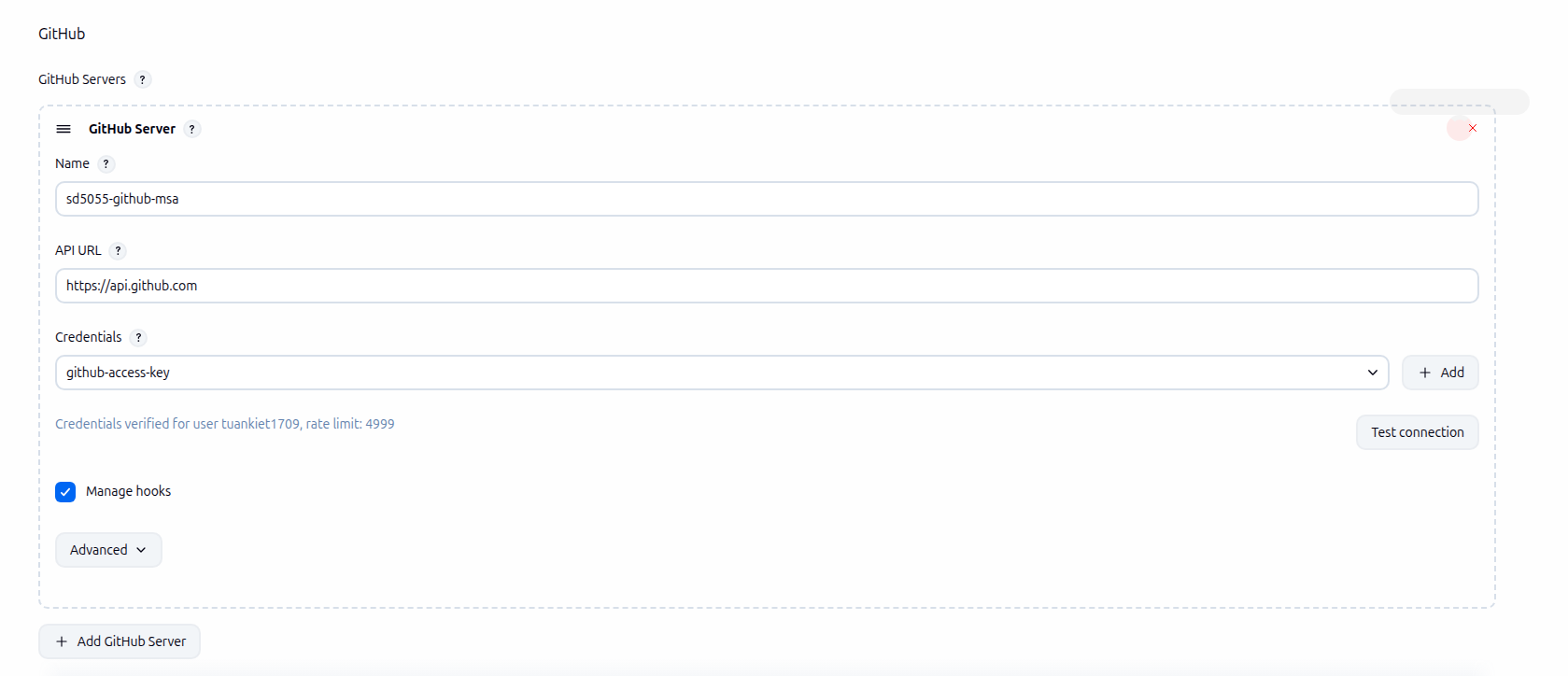


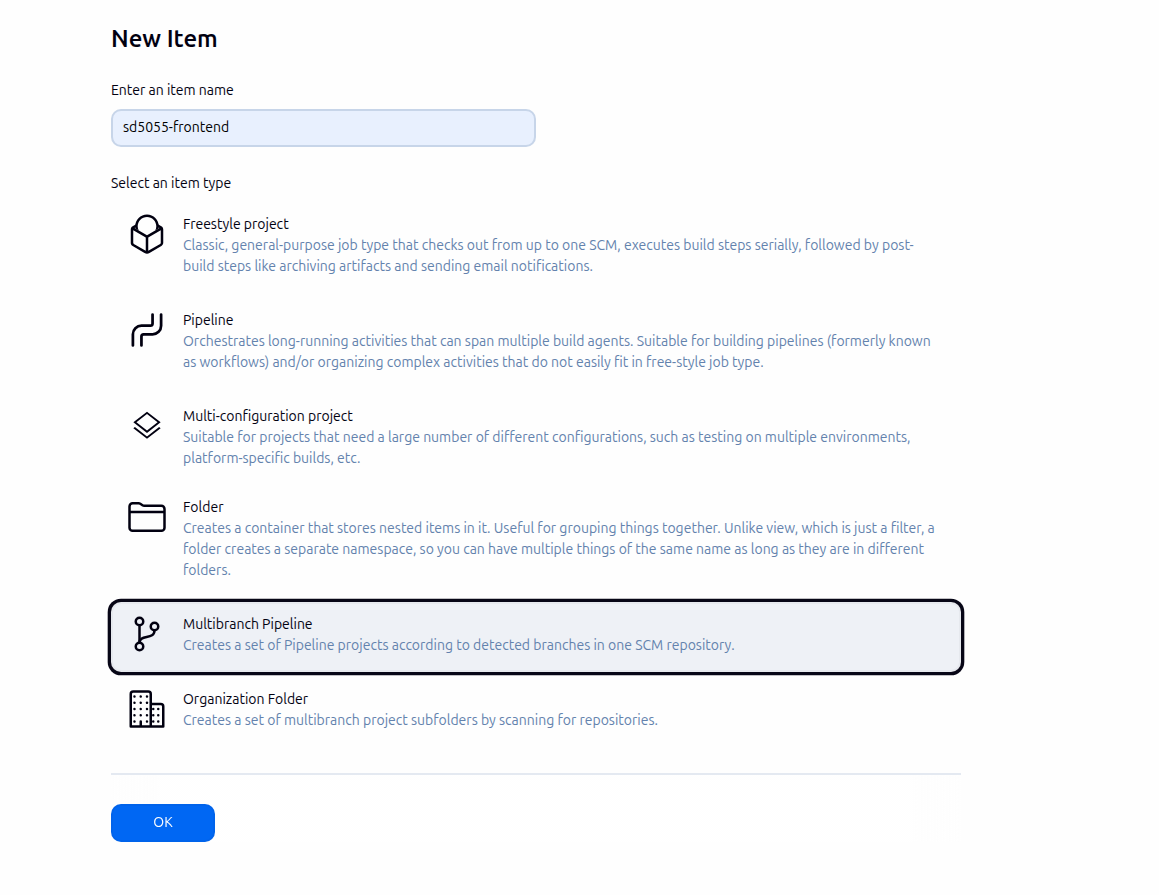
* We can select to install suggested plugins by Jenkins or select plugins you want to install. After installing, we can go to Jenkins dashboard

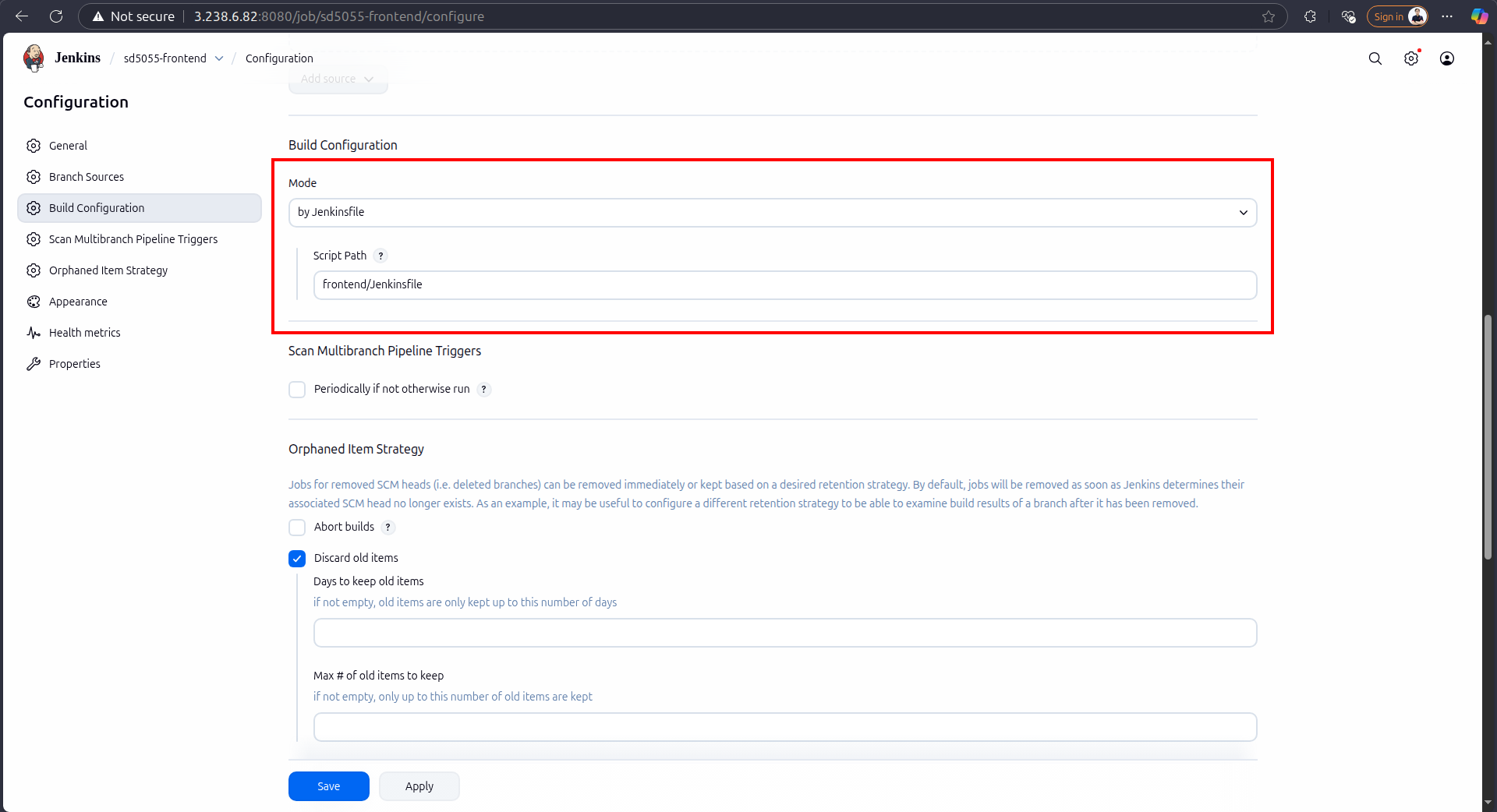
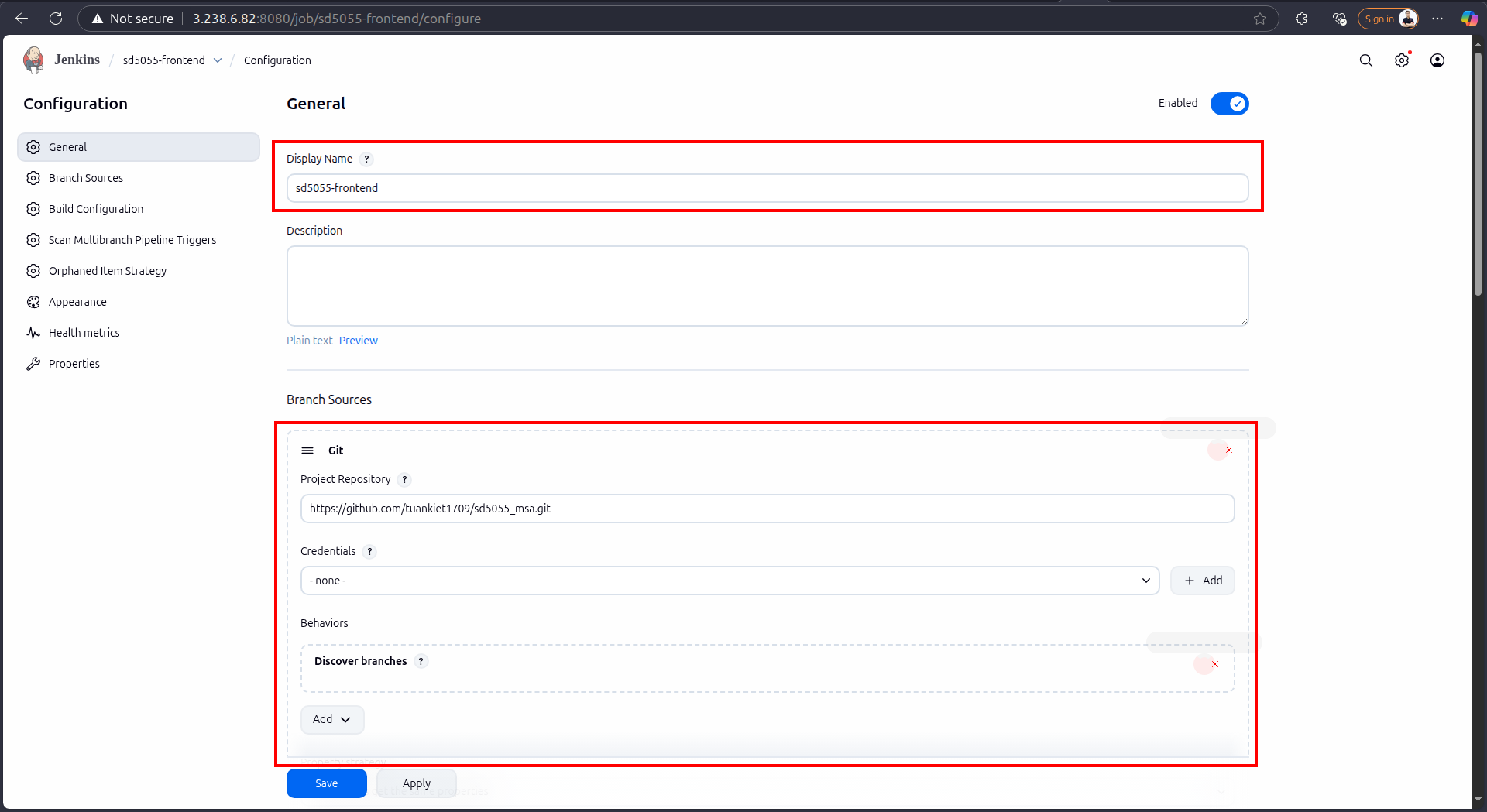


* To run CI/CD, we also need to install some plugins
  + Docker
  + Docker Pipeline
  + Pipeline
  + AWS Steps
  + Crednetials Binding
  + Git

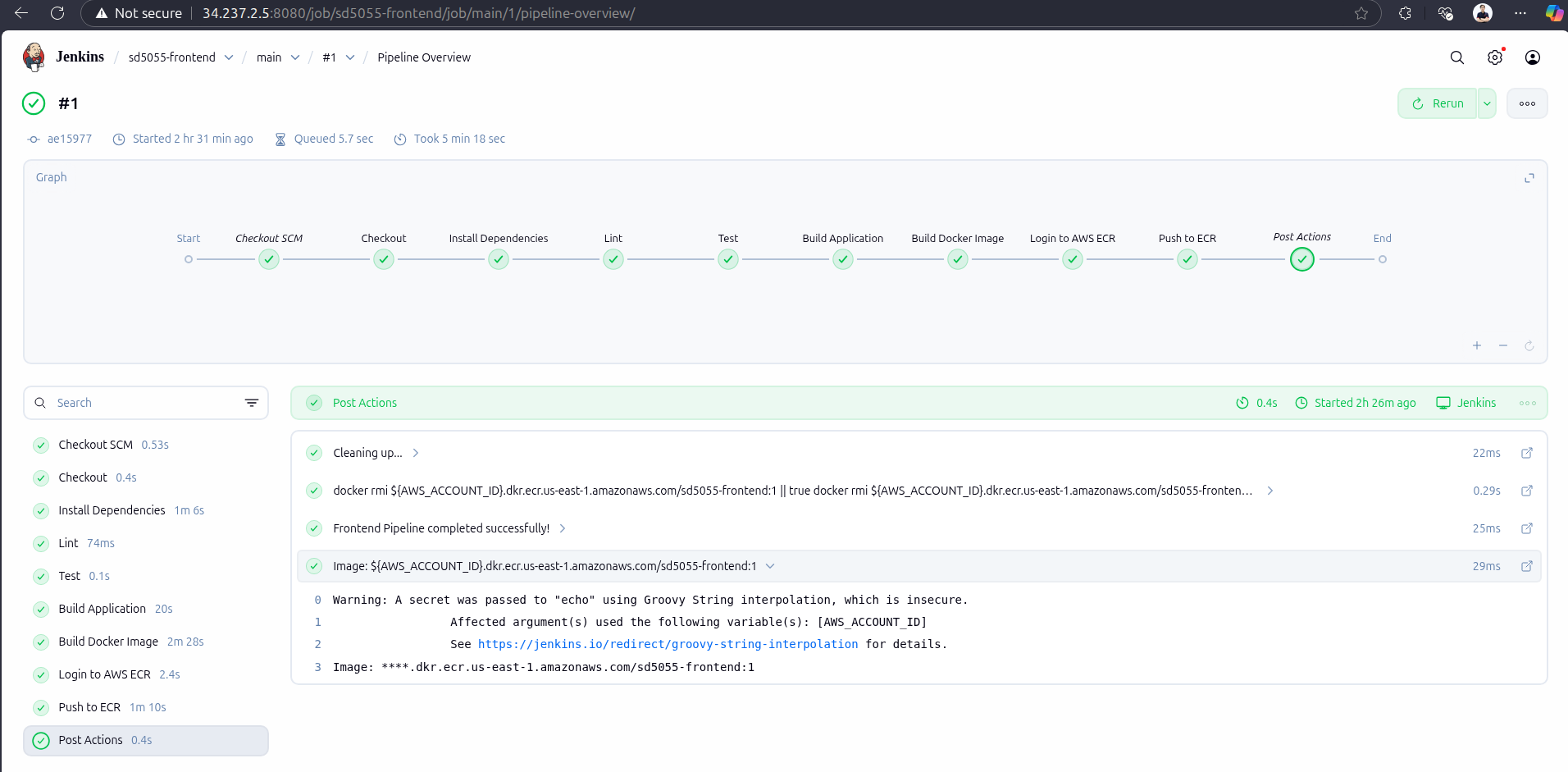
## Deploy MSA application to EKS using Jenkins CI/CD

* We need a MSA repo including frontend and backend: <https://github.com/tuankiet1709/sd5055_msa>
* To run CI on Jenkins, we need to define a Jenkinsfile in BE and FE sources.
* Jenkinsfile will define which agent use to run CI, the stage of CI is a step to install, build, validate, test, and push to ECR
  + <https://github.com/tuankiet1709/sd5055_msa/blob/main/backend/Jenkinsfile>
  + <https://github.com/tuankiet1709/sd5055_msa/blob/main/frontend/Jenkinsfile>
* When we already have a jenkinsfile, we need to config Jenkins to allow access to MSA repository and AWS resources
  + We need to create github access token to allow Jenkins access to msa repo
    - Click on user → Setting → Developer settings → Personal access tokens → Tokens (Classic) → Generate new token (classic) → Put token name and select scopes to allow Jenkins access to github → Create
  + We also need to create github webhook to trigger Jenkins whenever we push new code
    - In sd5055-msa repo → go to Settings → webhooks → Add webhook → we will input jenkins url with /github-webhook (for example: <http://34.237.2.5:8080/github-webhook/>)
  + We need to create some credentials in Jenkins which allow Jenkins access to Resources (Go to Jenkins Credentials → Click on (global) → Add Credentials)
    - **AWS Account ID**: The AWS Account ID
    - **AWS Access Key ID**: The AWS Access Key ID to access aws resources
    - **AWS Secret Access Key**: The AWS Secret Access Key ID to access AWS resources
    - **Github Token Key**: The key to allow Jenkins access to github resources (We already created above)
  + Add Github Server for Jenkins (Go to Jenkins settings → in Github section → Add Github Server with name → Credentials → Select github-access-key

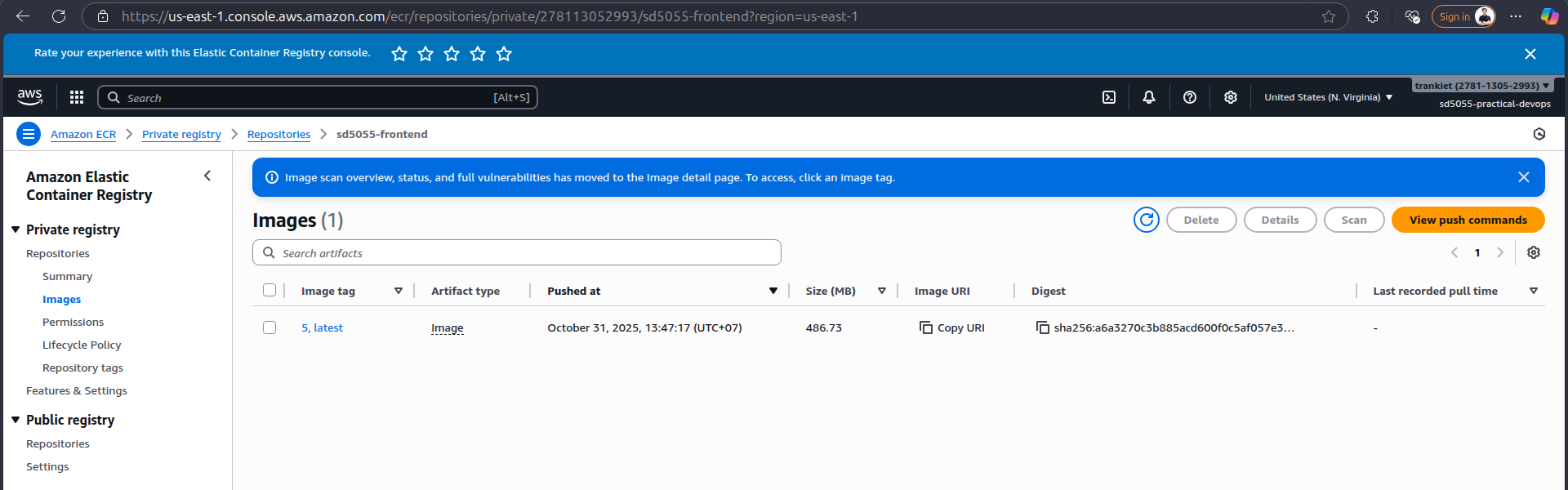
* Next, we will create job to run CI for MSA Application
  1. Frontend
  + Create multibranch pipeline job
  + Configure Jenkins job, point to frontend branch source and jenkinsfile of frontend



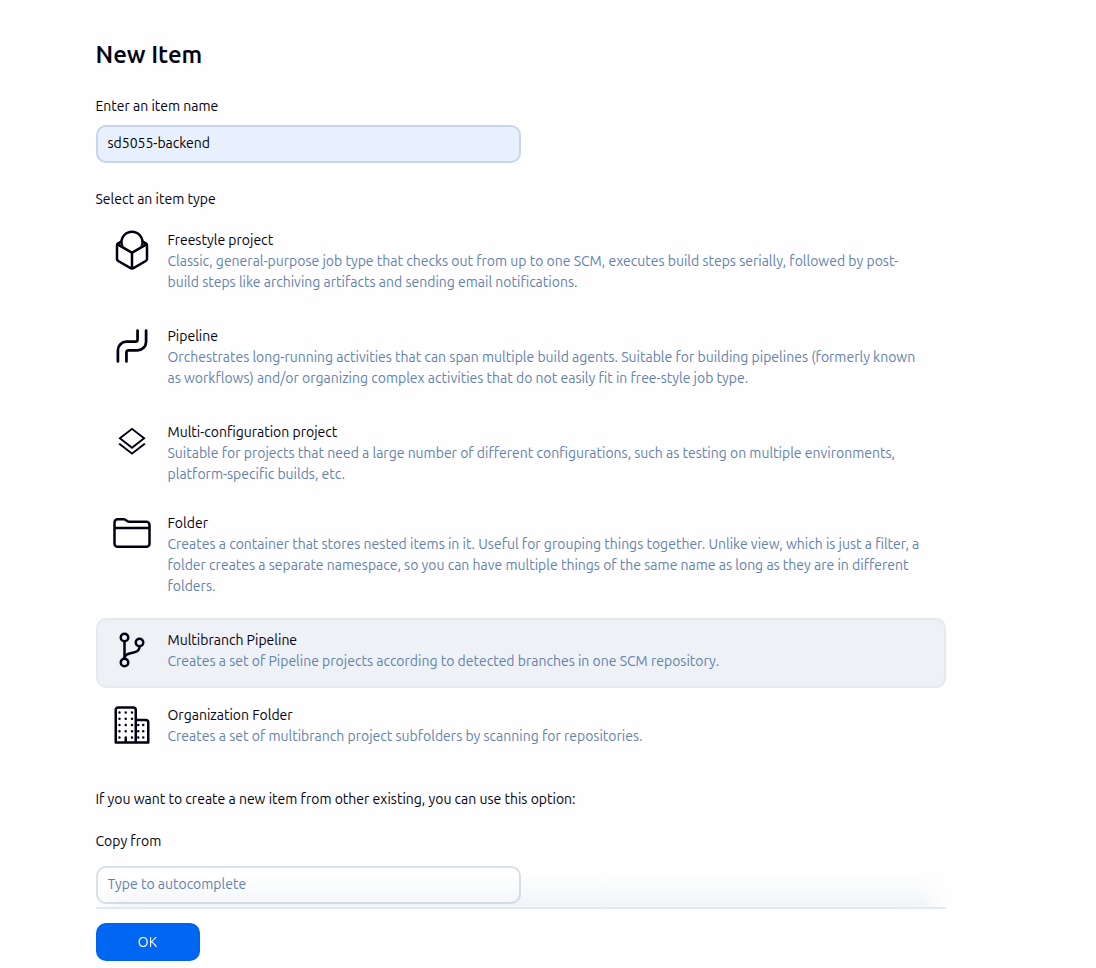
* + After creating, the job will run jenkins based on the jenkins file and source



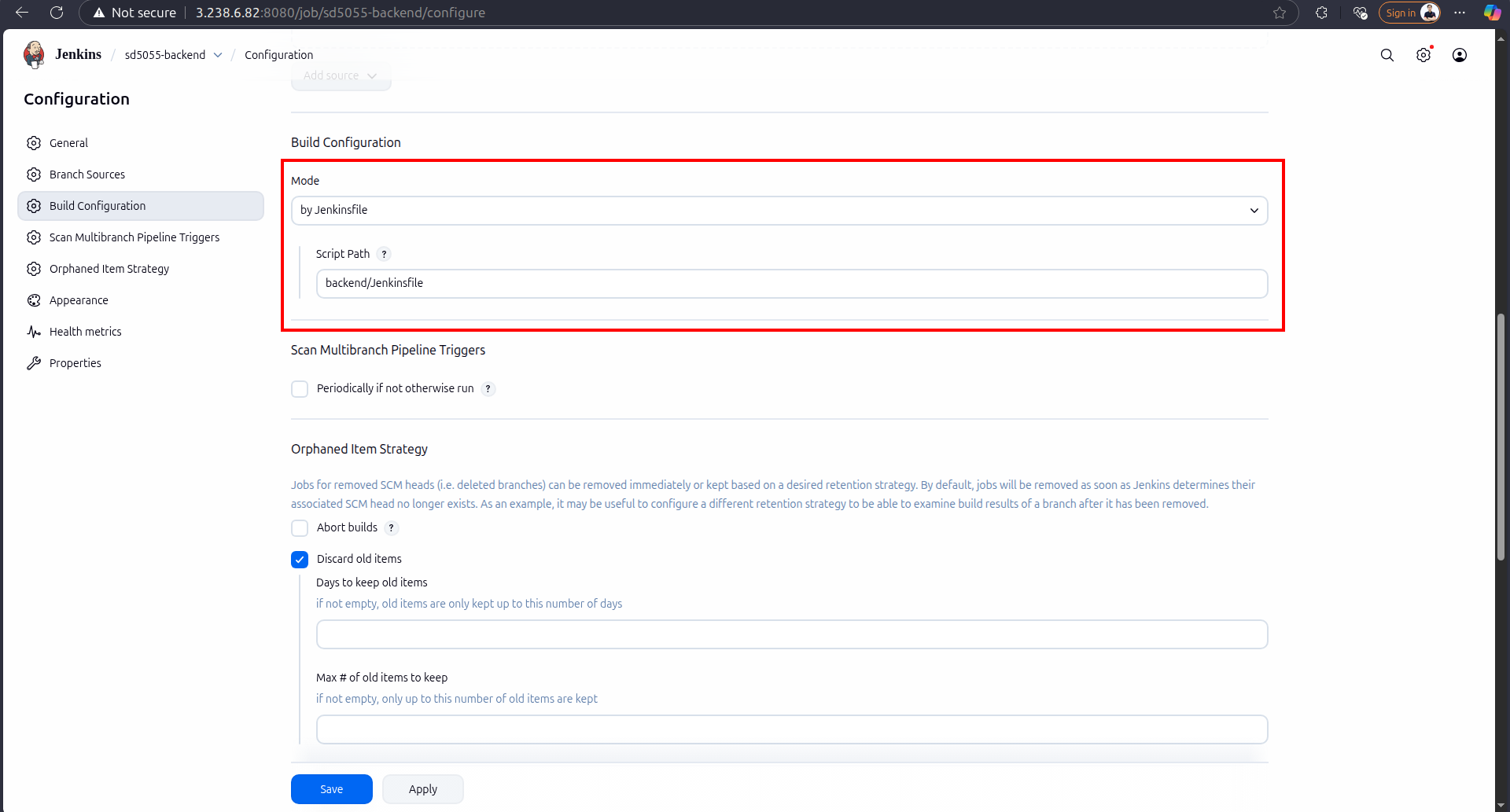
* + Once the CI run successfully, it will push the image to ECR

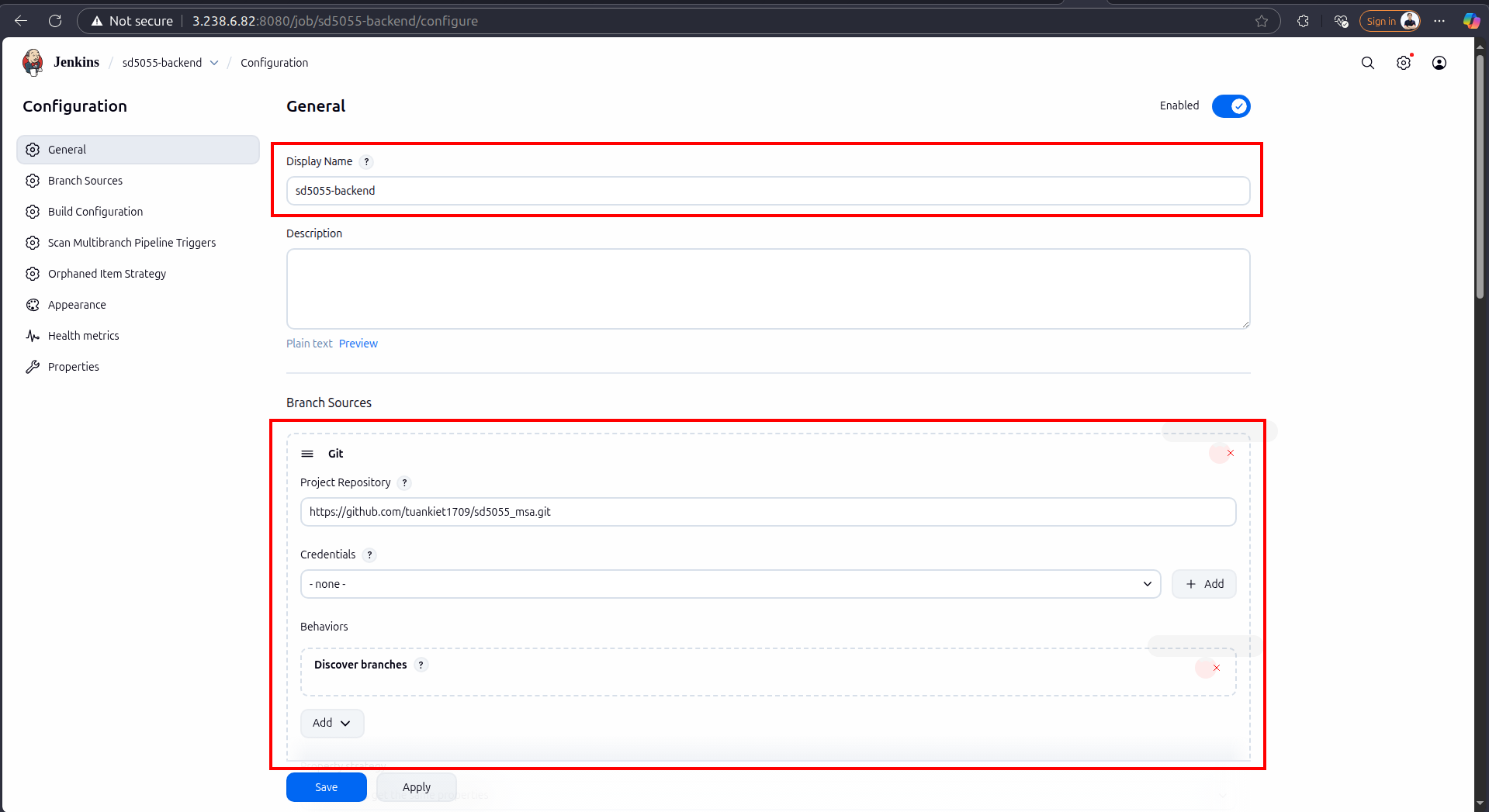


* 1. Backend
  + Setup the same as frontend
  + Create a new multibranch job

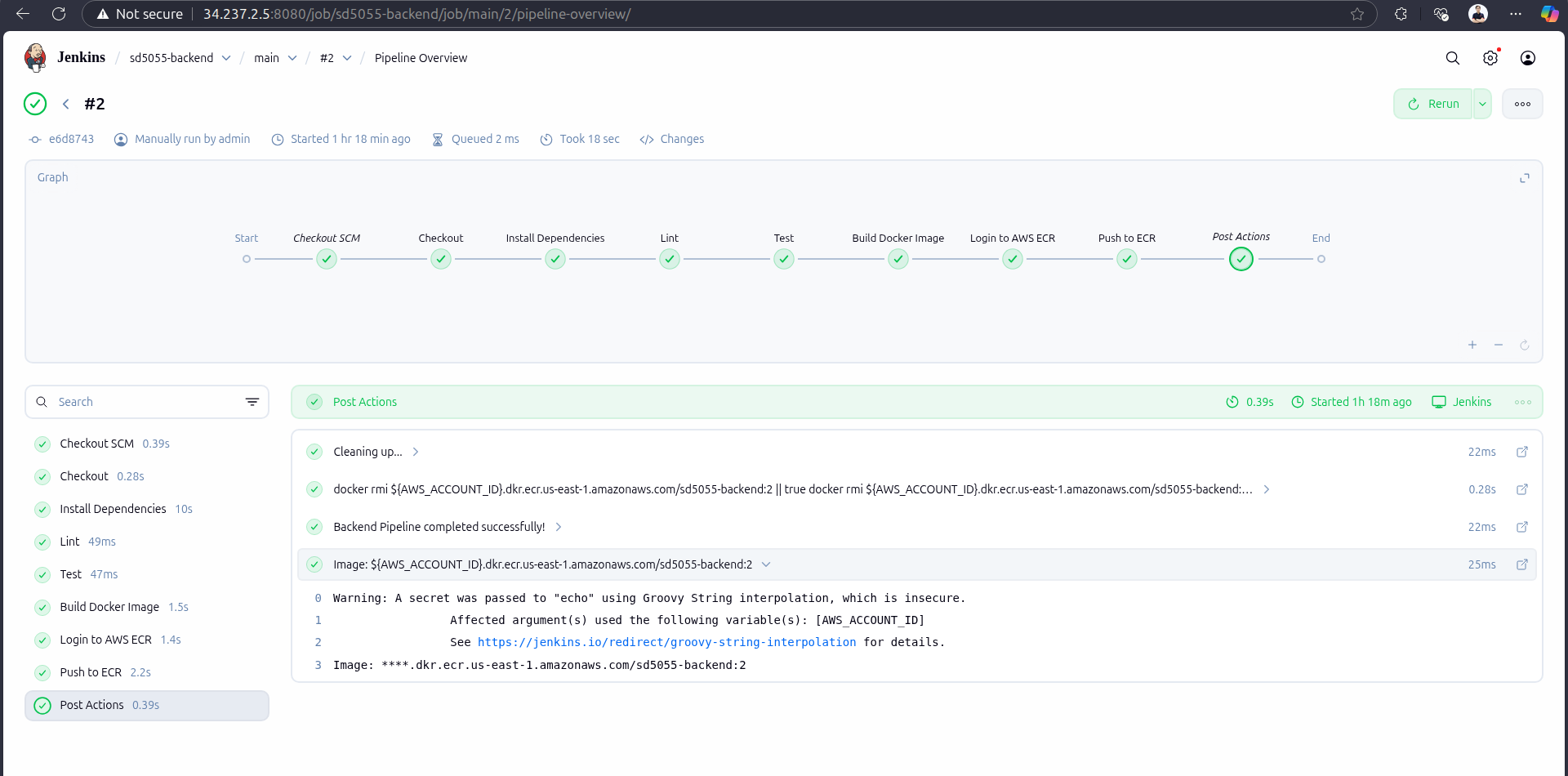


* + Configure Jenkins job, point to backend branch source and jenkinsfile of backend

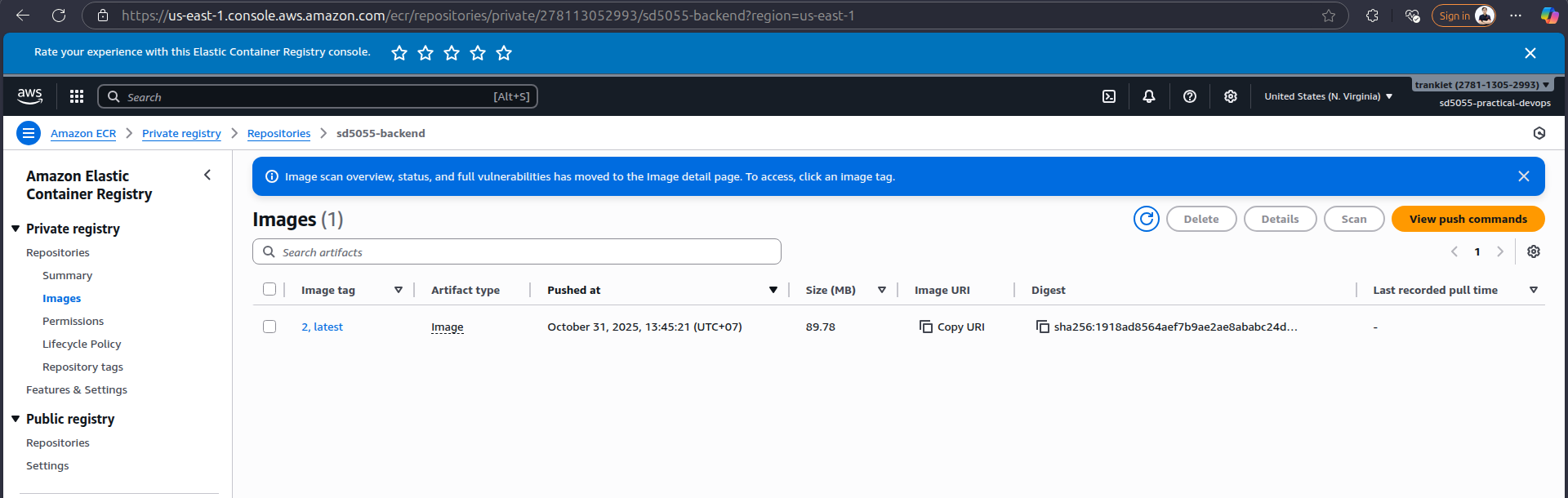


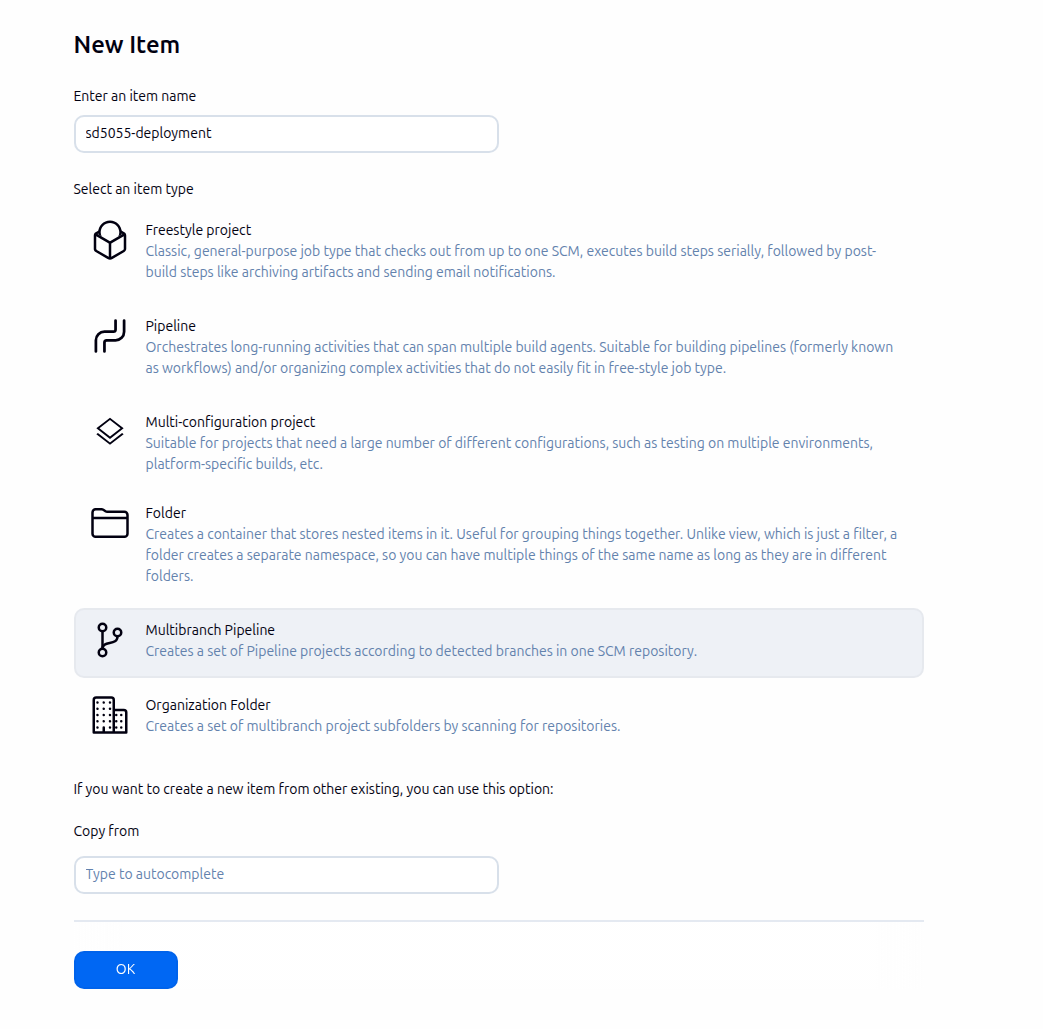


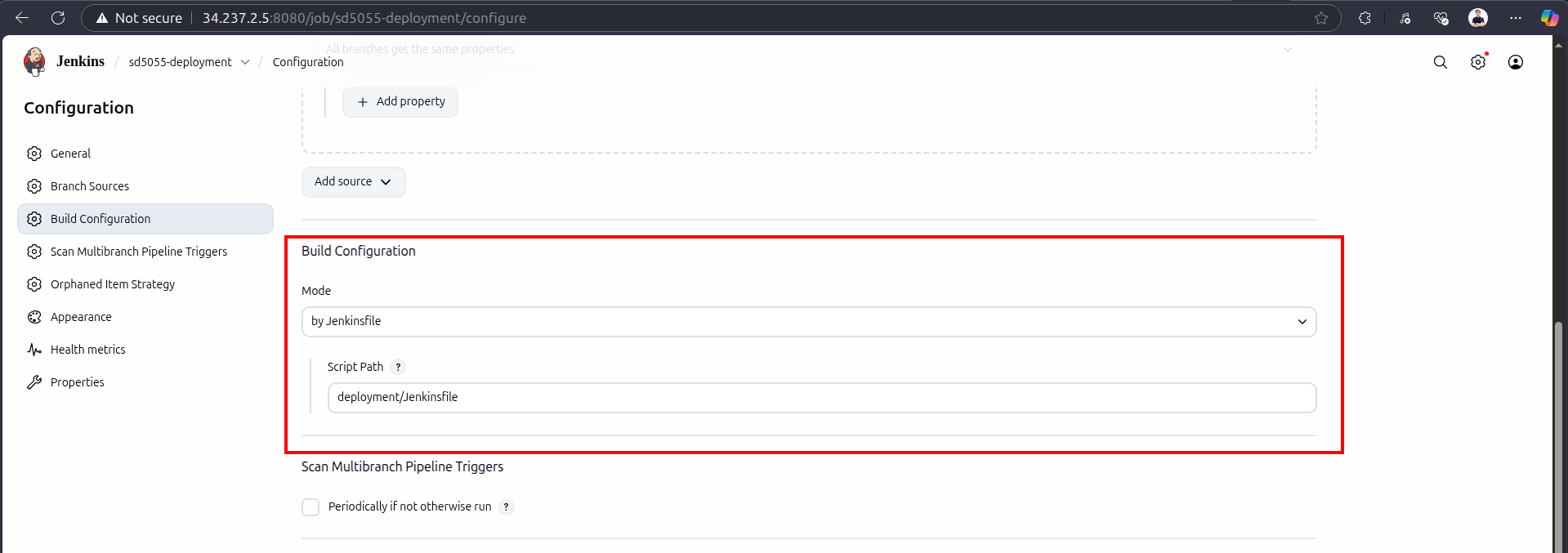
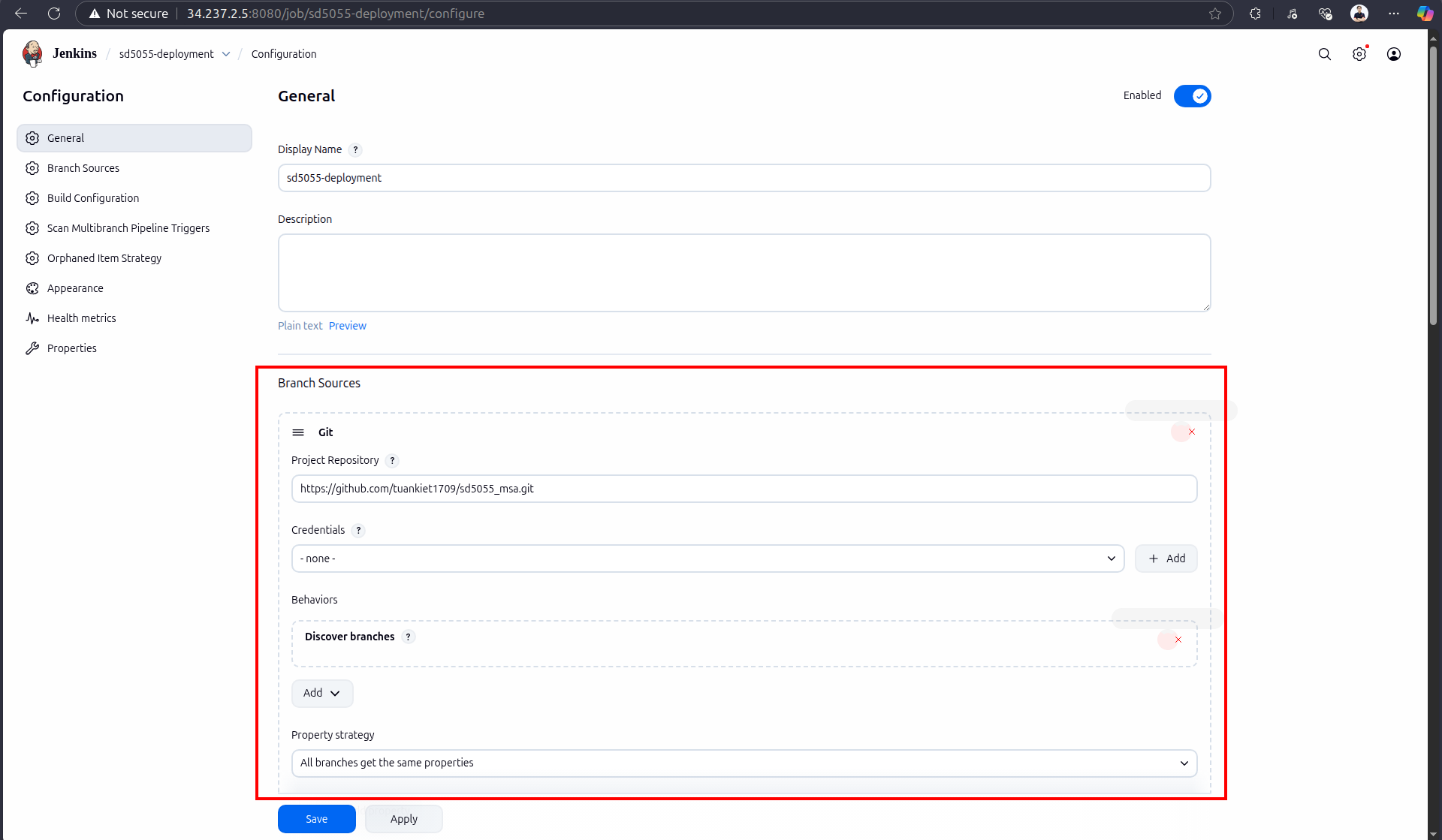
* + After creating, the job will run jenkins based on the jenkins file and source



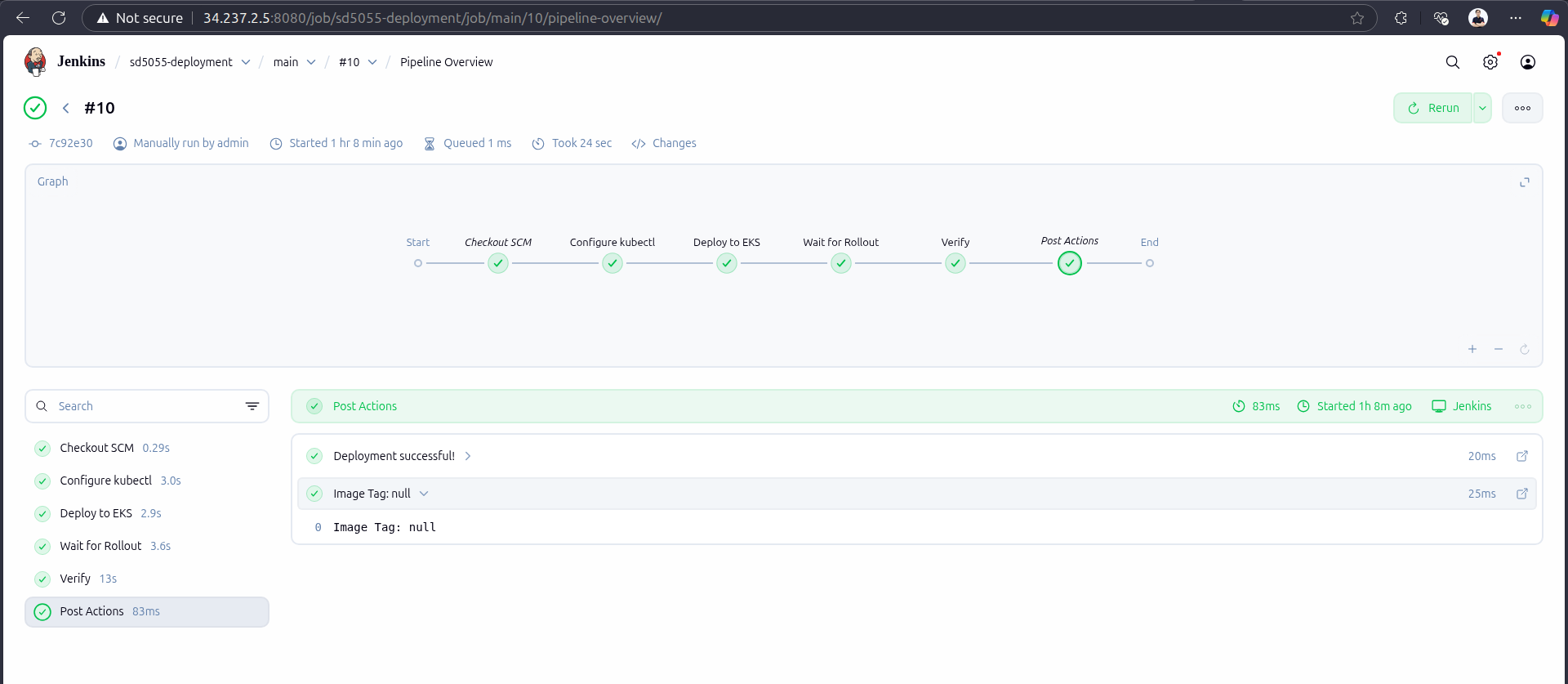
* + Once the CI run successfully, it will push the image to ECR



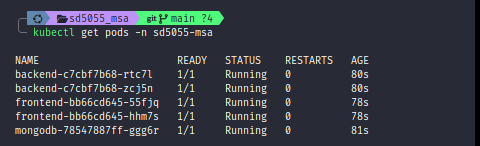
* 1. Deployment
  + Refer to: <https://github.com/tuankiet1709/sd5055_devops_cicd_deployment/tree/main/deployment>
  + Create a new multibranch job
  + Configure Jenkins to point to deployment resource and jenkinsfile

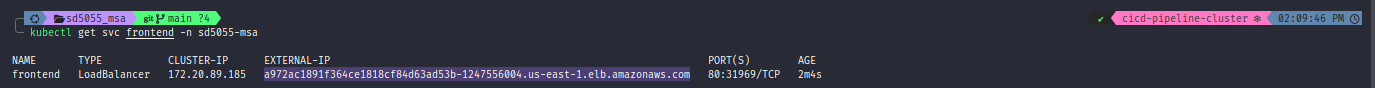
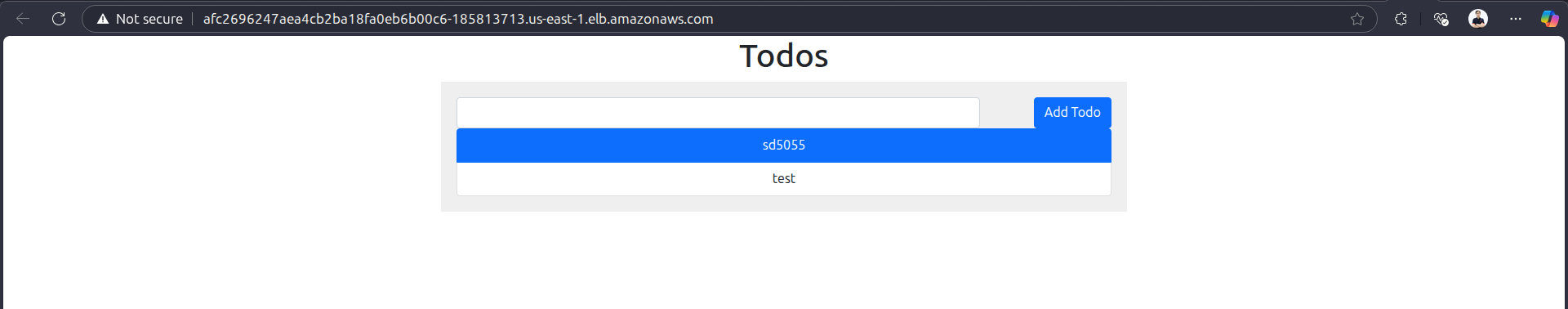
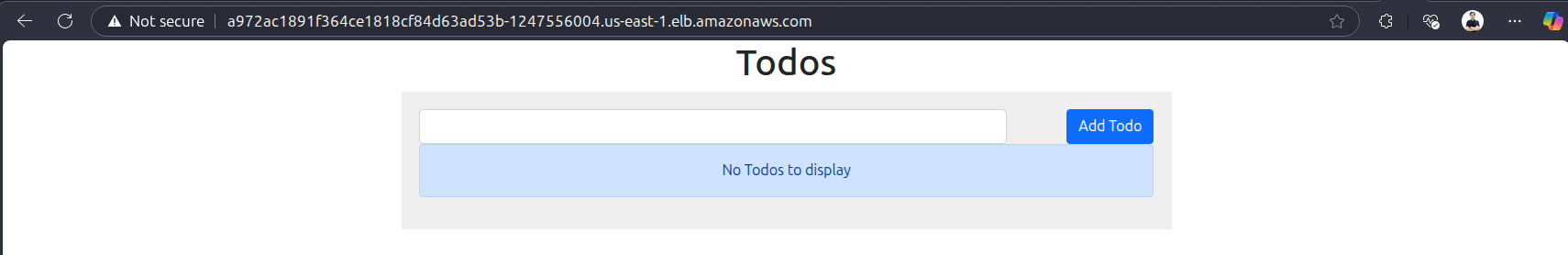


* + After creating, the job will run jenkins based on the jenkins file and source



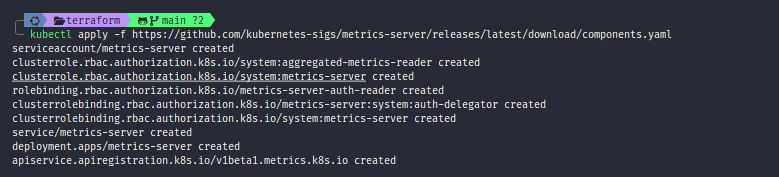
* + Once the Jenkins run successfully, it will deploy backend and frontend from ECR to eks in sd5055-msa namespace
  + Use `kubectl get pods -n sd5055-msa` to get all pods inside namespace



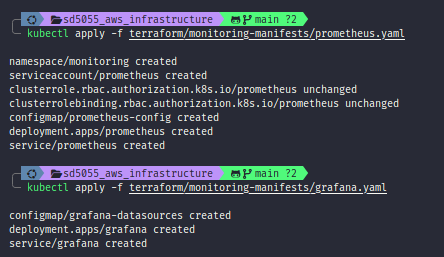
* + Use `kubectl get svc frontend -n sd5055-msa` to get frontend URL from eks
  + We can access to application through the URL we retrieve from EKS
  + The application is working as expected now

## Monitoring with Prometheus and Grafana

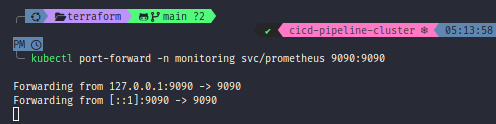
* Define yaml file to deploy Prometheus and Grafana in sd5055\_aws\_infrastructure: https://github.com/tuankiet1709/sd5055\_aws\_infrastructure/tree/main/monitoring
* Deploy metrics-server to eks to monitoring metrics



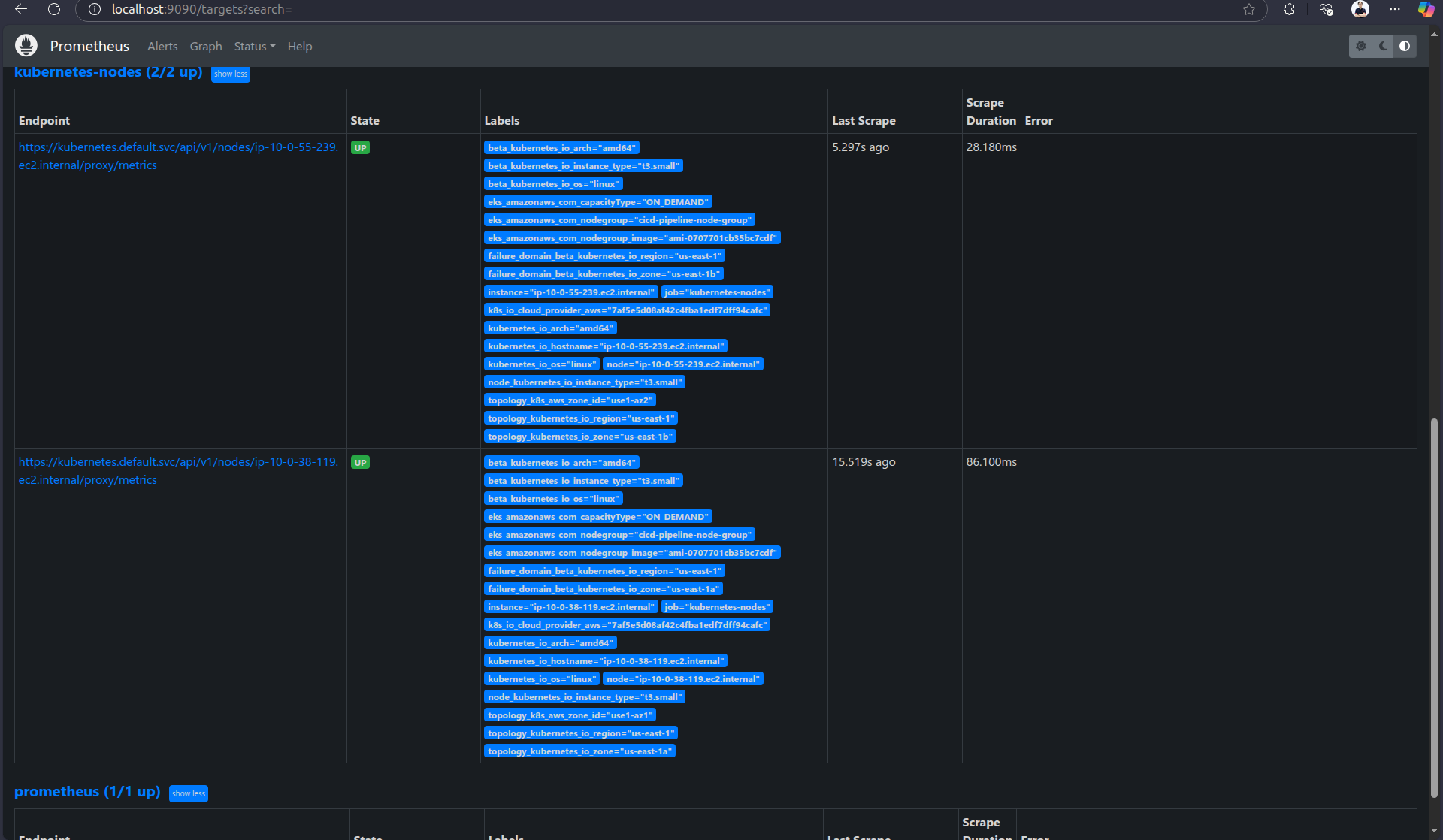
* Deploy grafana and prometheus to eks successfully

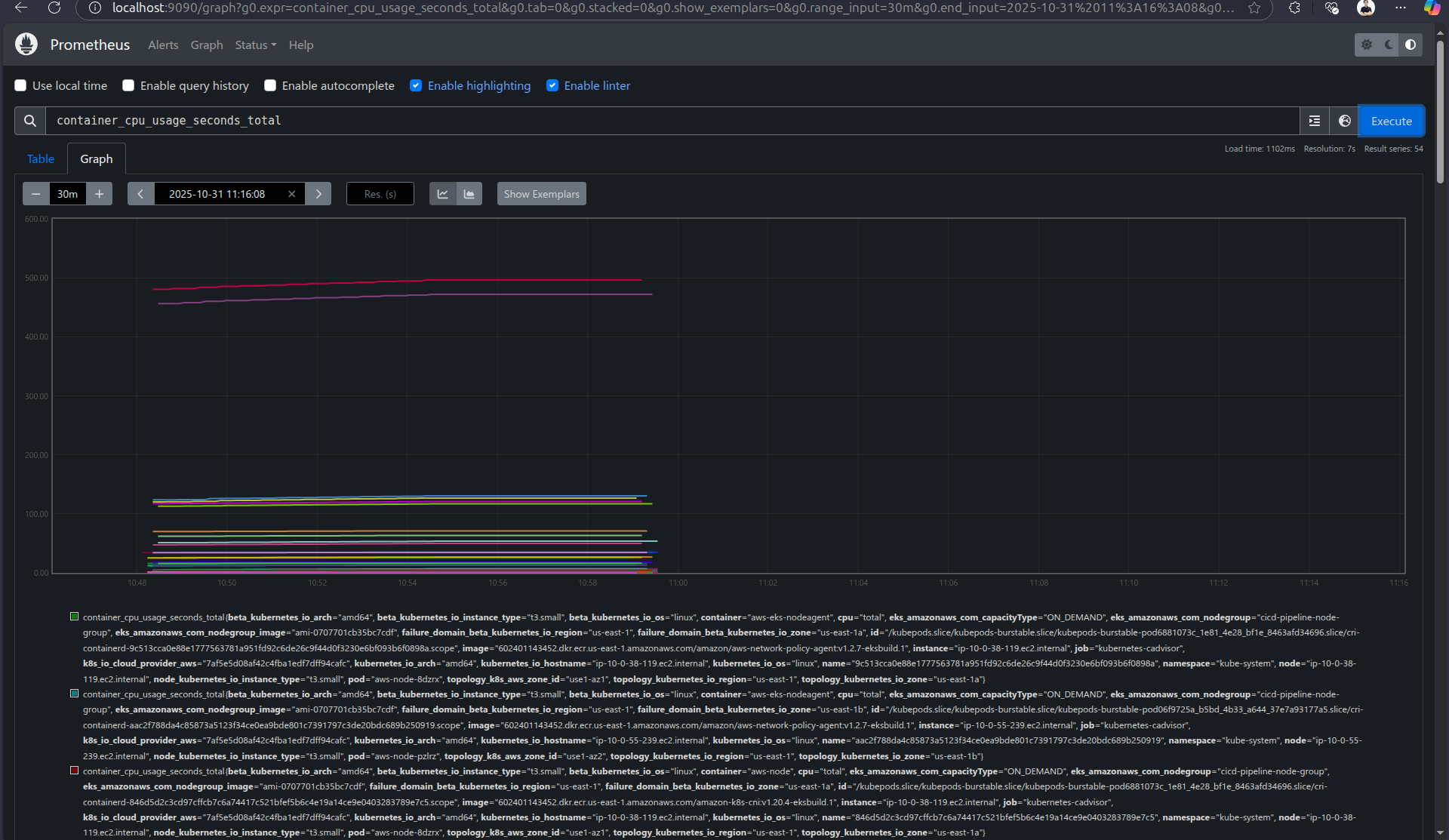


* Forward prometheus port from eks to local

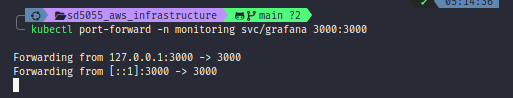


* We can monitoring with prometheus now

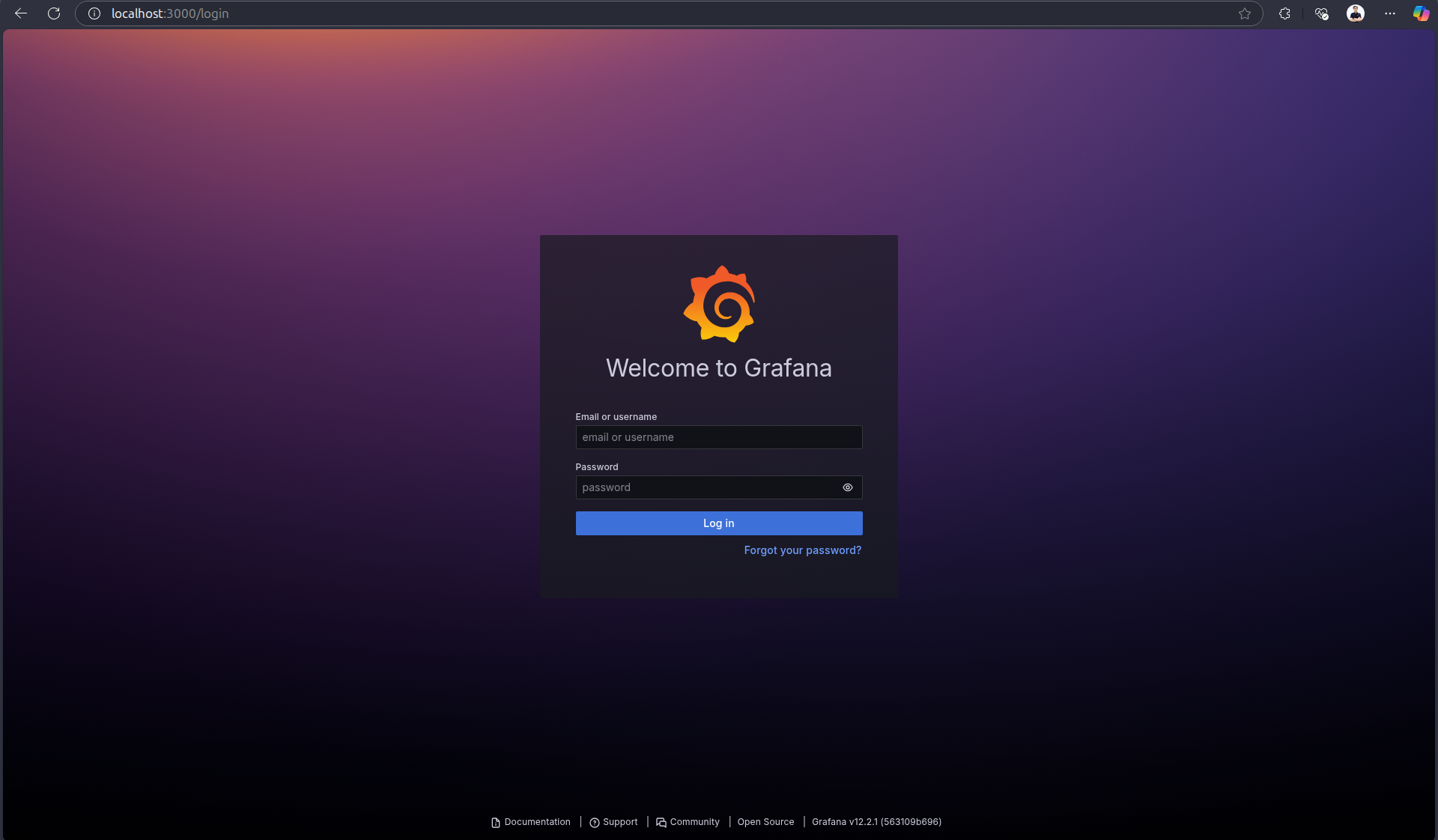




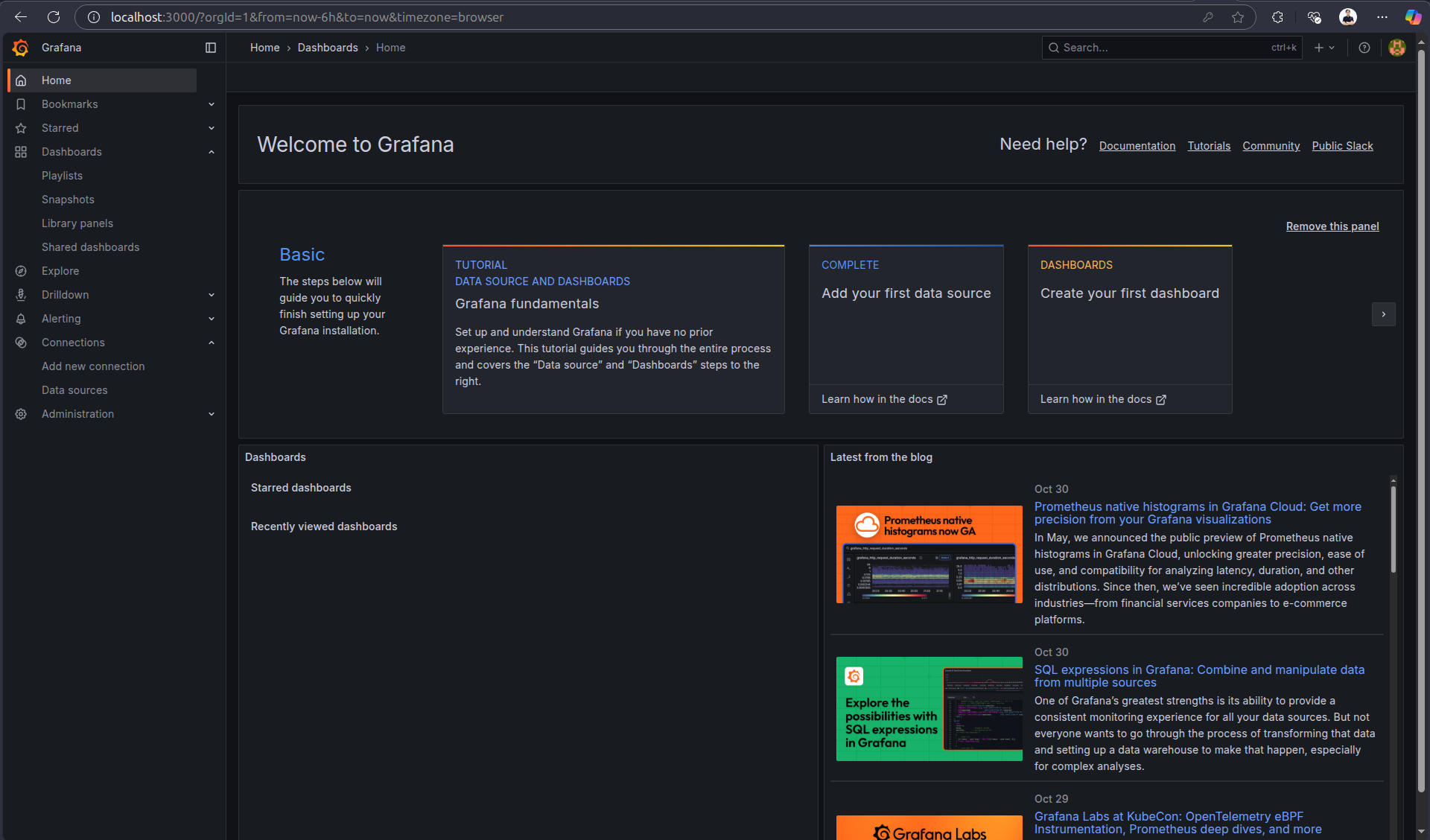
* Forward Grafana port from eks to local



* We can access to Grafana now



* Logged in the Grafana successfully



* We can monitor with Grafana now

