**CI/CD for Docker Kubernetes using Jenkins**

**Project Title**

CI/CD for Docker Kubernetes using Jenkins

**Scenario**

The project involves setting up a continuous integration and continuous delivery (CI/CD) pipeline for an application with a microservices architecture. The application’s components are containerized, and developers are constantly making changes to improve development. Consequently, the operations (ops) team frequently needs to adapt to these changes, requiring a continuous build and test process along with regular container builds. Additionally, the ops team must manage regular deployment requests using a Kubernetes cluster to maintain the system.

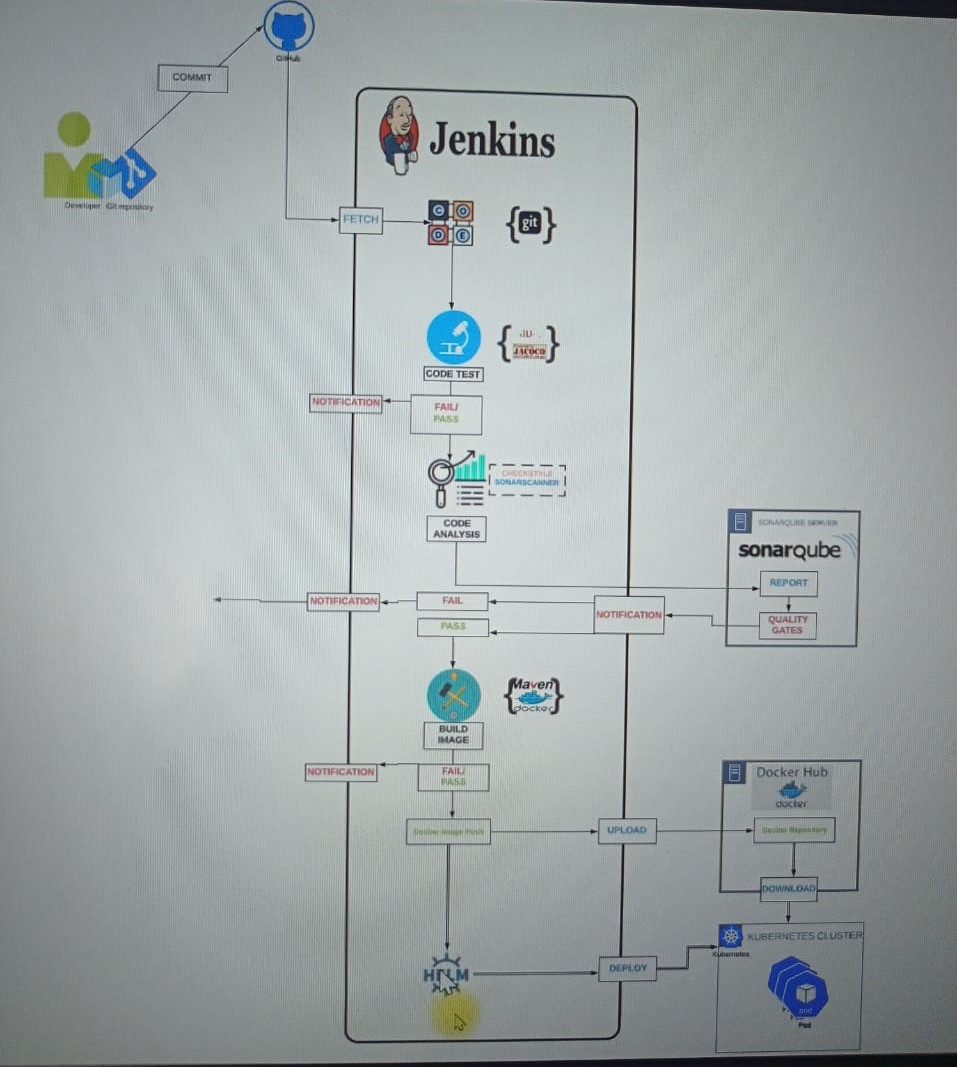
**Problems**

Due to continuous code changes, the ops team receives frequent deployment requests, making it a time-consuming process. The development (dev) team is dependent on the ops team for all deployments, creating a bottleneck and potentially slowing down the overall development and deployment cycle.

**Solution**

To address these challenges, an automated build and release process is required. Implementing a continuous delivery pipeline for Docker containers will streamline the process, allowing for continuous integration, delivery, and deployment. This will reduce the dependency of the dev team on the ops team and speed up the overall workflow.

**Architecture**

****

**Tools**

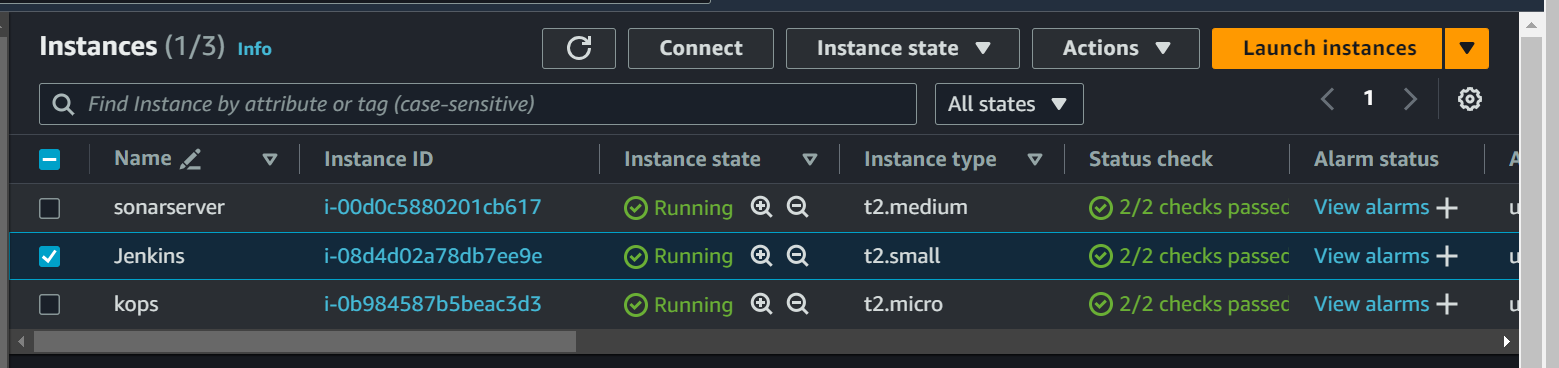
* **Kubernetes:** The container orchestration tool to manage and scale containerized applications.
* **Docker:** Runtime environment to build, run, and test Docker images.
* **Jenkins:** For continuous integration and continuous delivery (CI/CD) to automate the build, test, and deployment processes.
* **Docker Hub:** Container registry to store and distribute Docker images.
* **Helm:** Tool for packaging and deploying applications on the Kubernetes cluster.
* **Git:** Version control system to manage source code and track changes.
* **Maven:** Build tool to automate the build process.
* **SonarQube:** Code analysis server to ensure code quality and security.

**Objective**

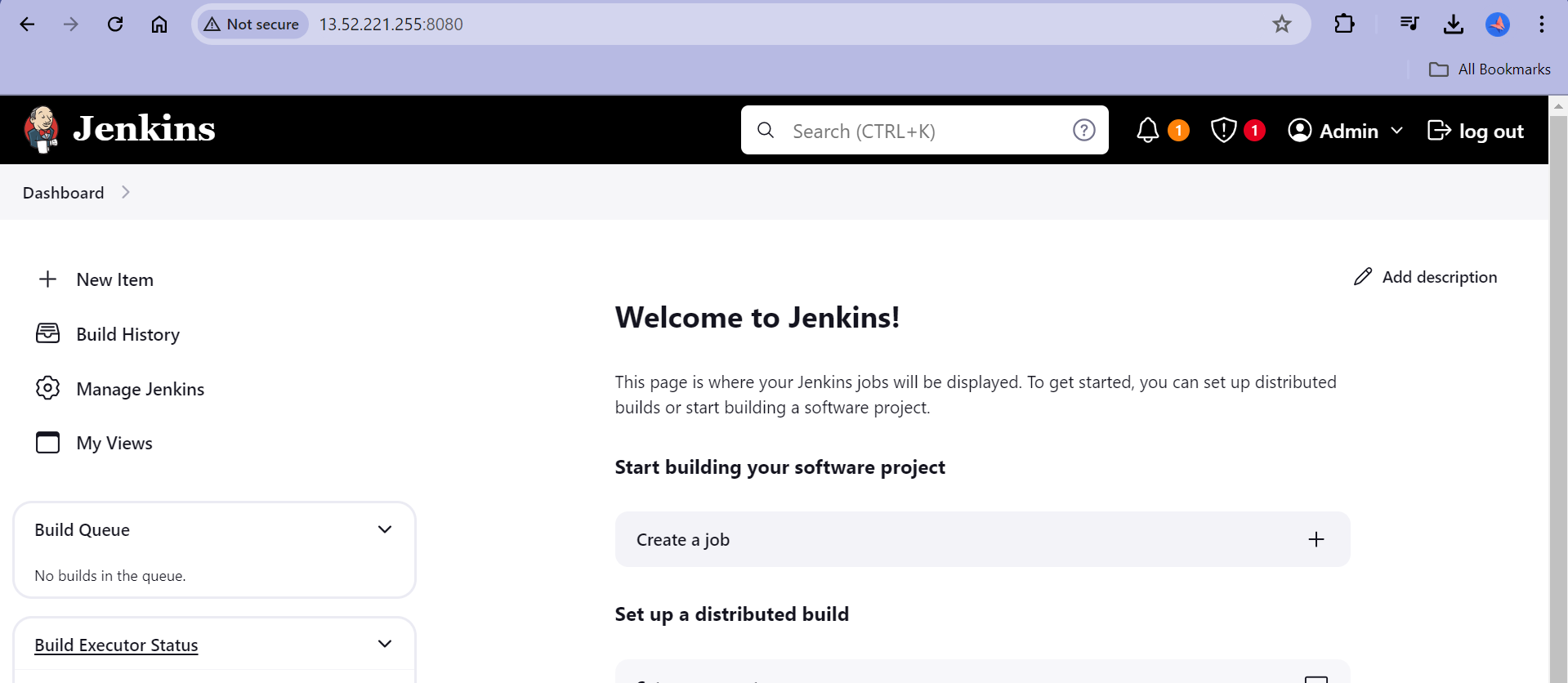
The primary objective is to establish a continuous delivery pipeline for Docker containers, ensuring seamless integration, delivery, and deployment of the application components.

**Steps of Implementation**

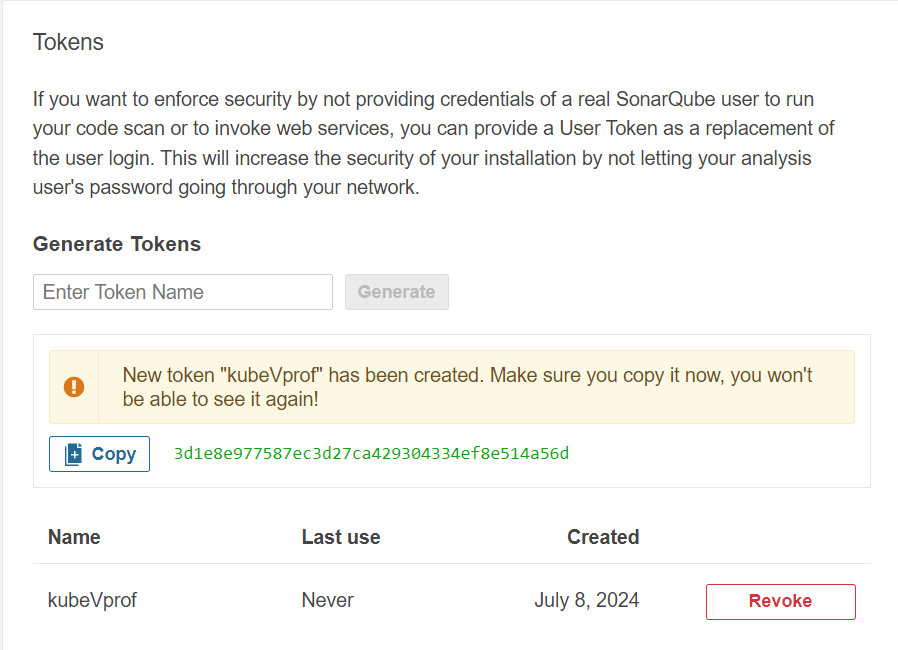
Instances to run : Jenkins, SonarQube and Kubernetes



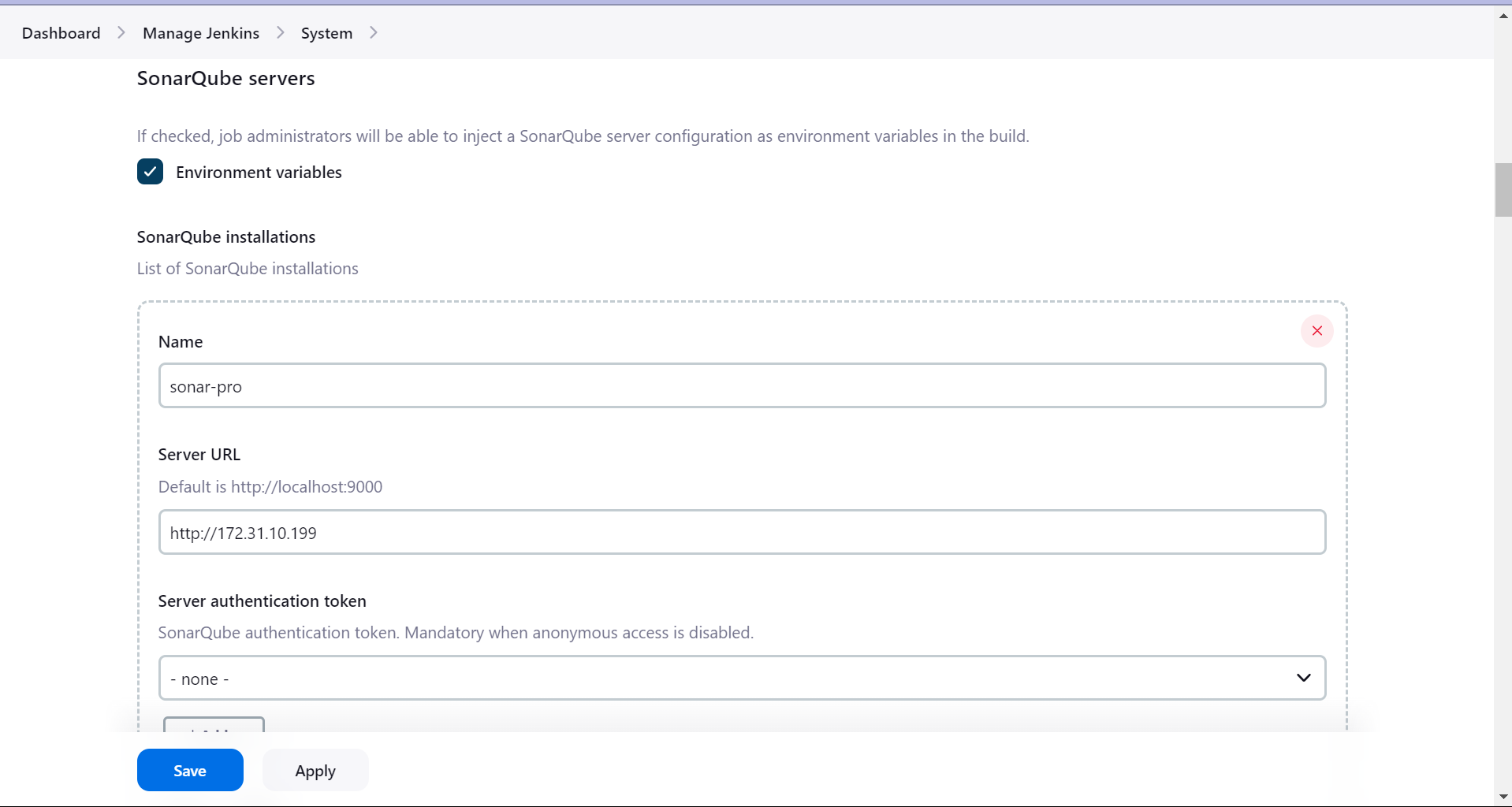
Ssh and login to Jenkins



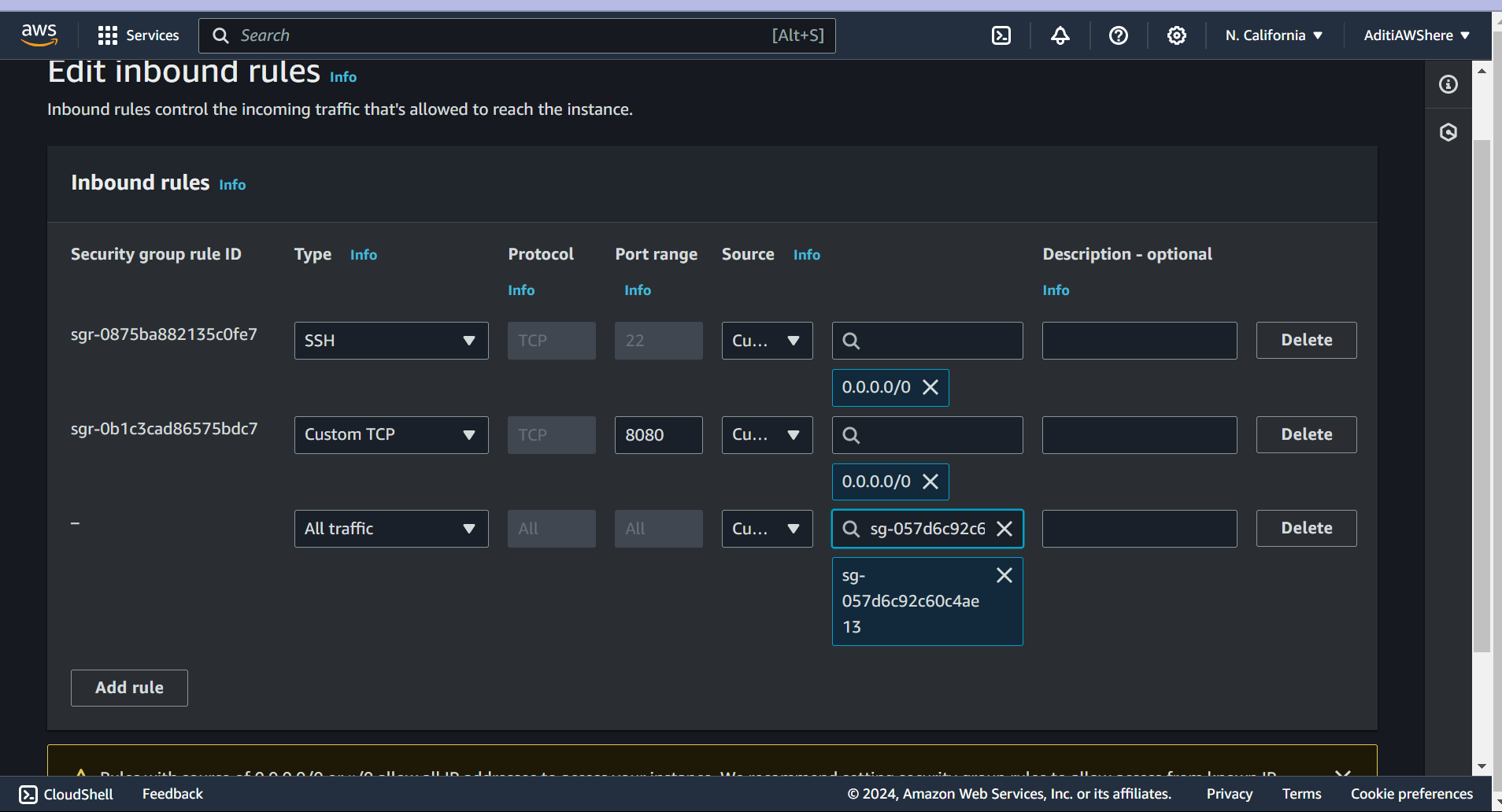
Ssh and login to SonarQube and Generate the token

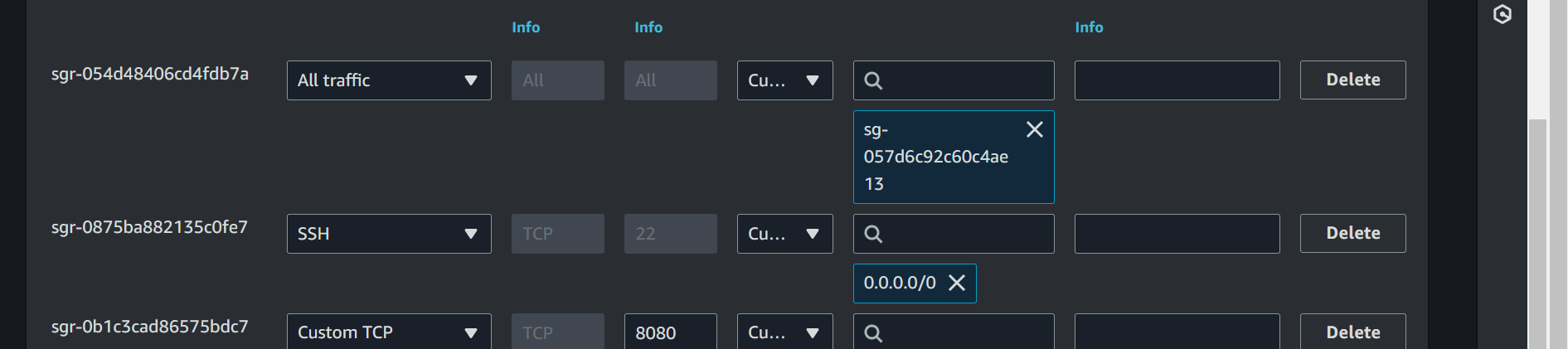


Go to manage Jenkins and add this token

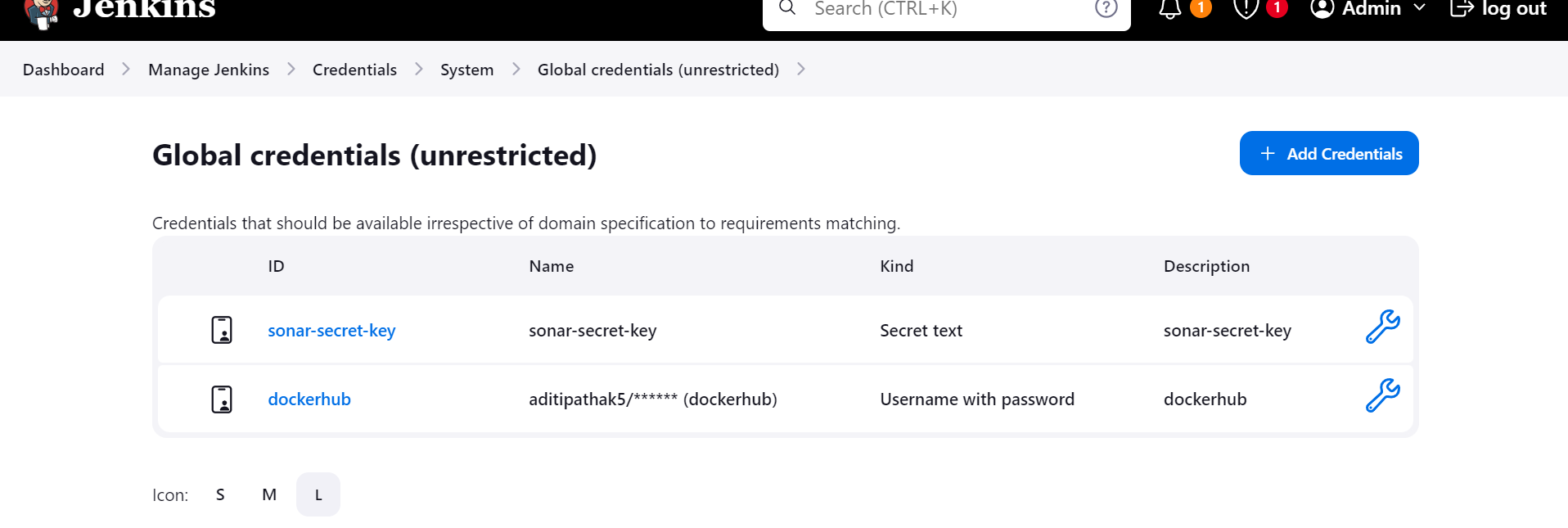


Make sure Jenkins Security Group have access to SonarQube and SonarQube have access to Jenkins



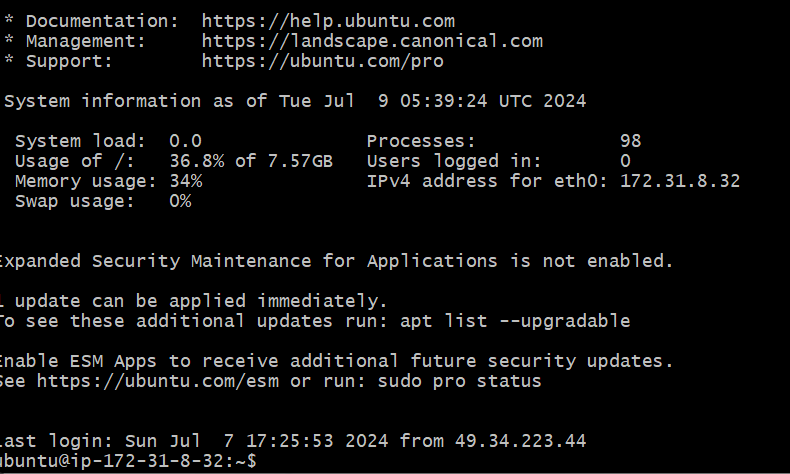


Store details of the Docker Hub account in Jenkins.

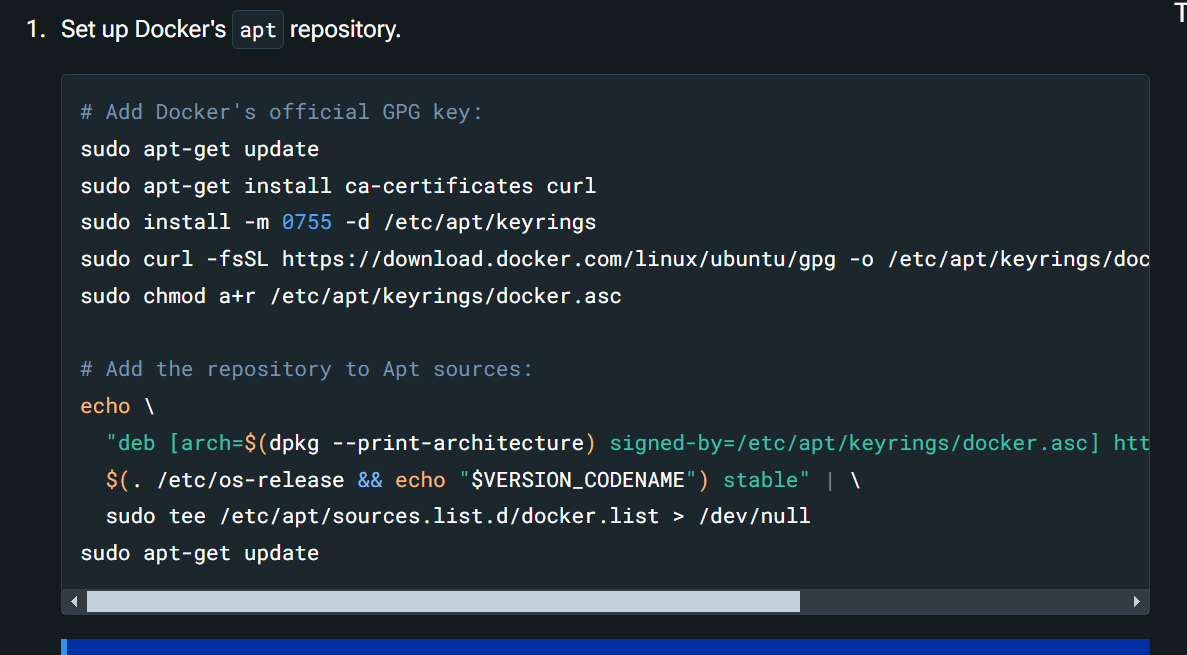


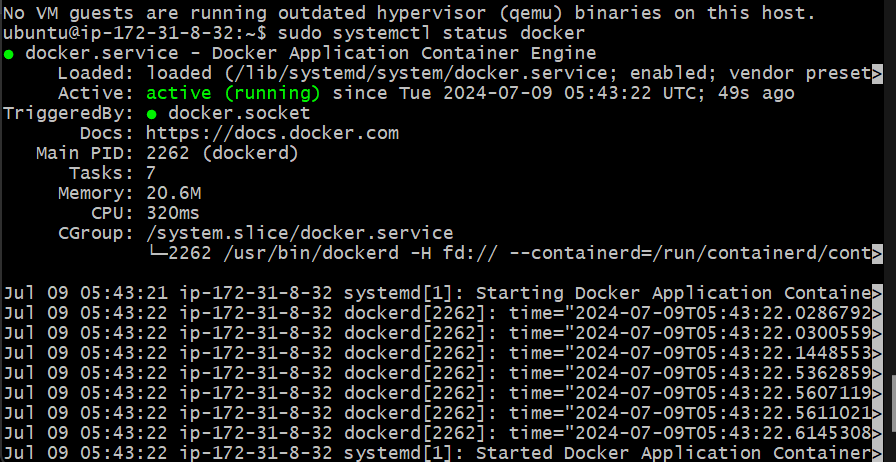
Install Docker Engine on Jenkins Server

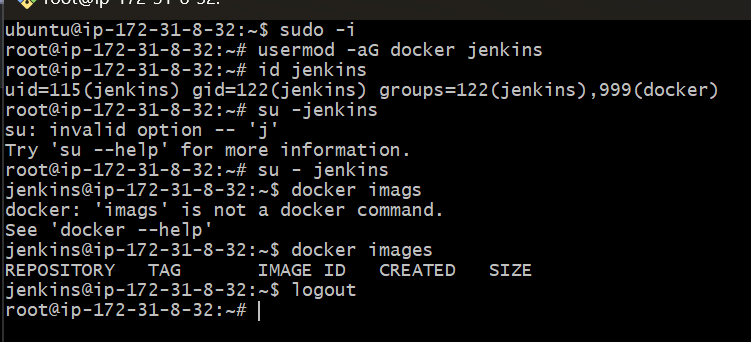
Ssh into Jenkins server



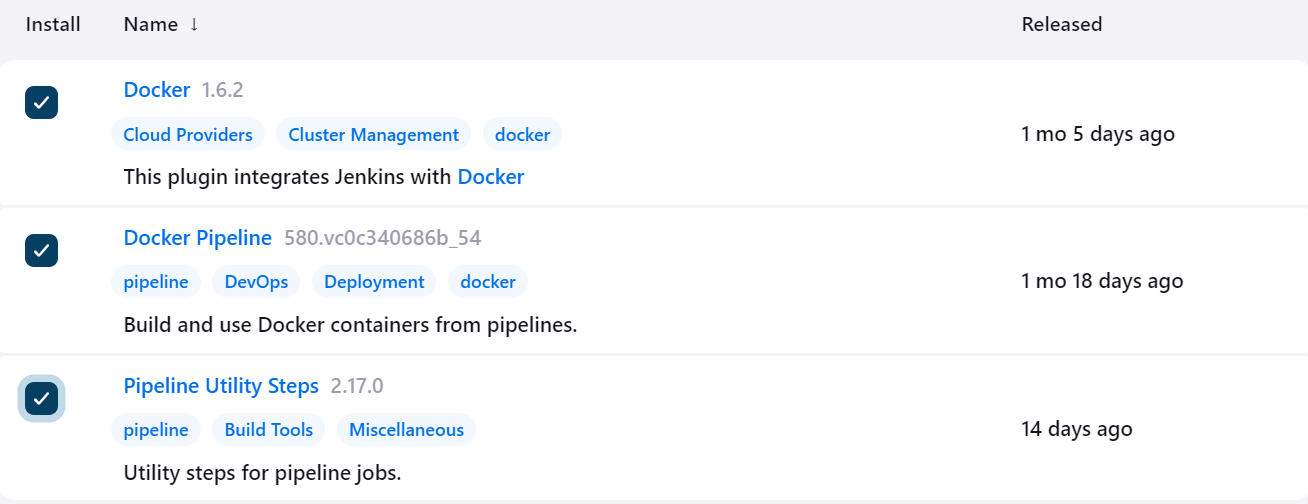
Go to Docker Installation for Ubuntu documentation and follow the steps.







Install some plugins in Jenkins



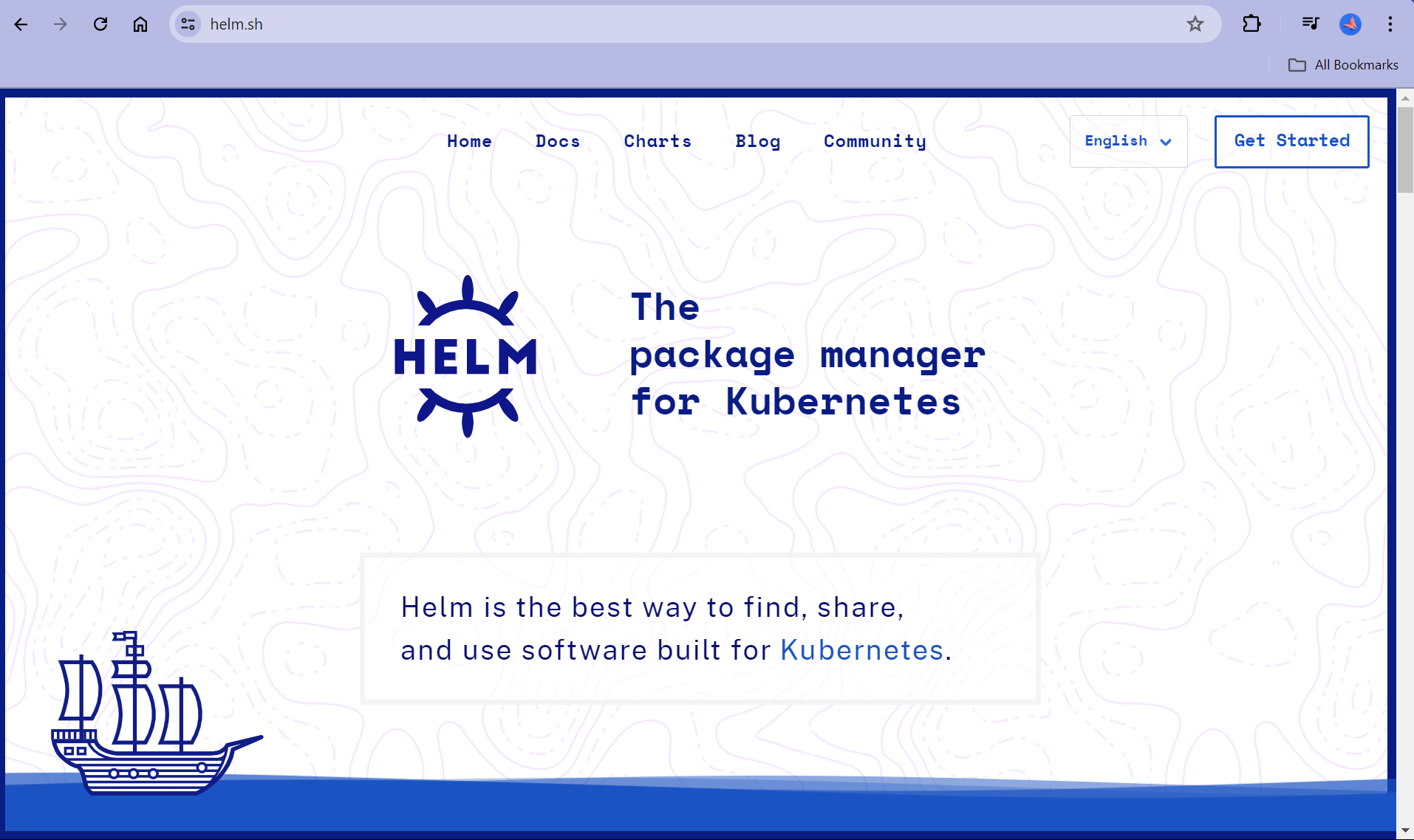
Create a Kubernetes Cluster.

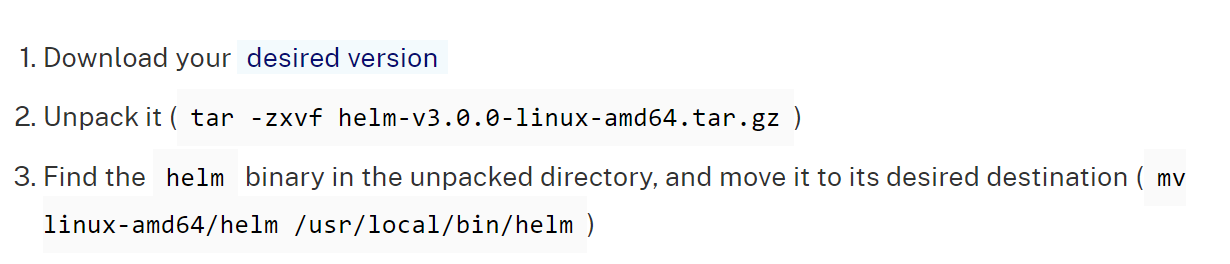
kops create cluster --name=kubevprof.hkhing.xyz --state=s3://vprof-kops-bucket-here --zones=us-west-1b,us-west-1c --node-count=2 --node-size=t3.small --control-plane-size=t3.medium --dns-zone=kubevprof.hkhing.xyz --node-volume-size=8 --control-plane-volume-size=8

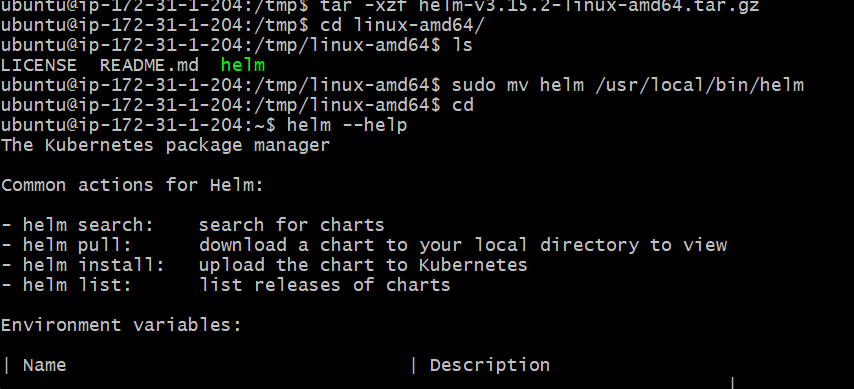
kops update cluster --name=kubevprof.hkhing.xyz --state=s3://vprof-kops-bucket-here --yes --admin

Install Helm:

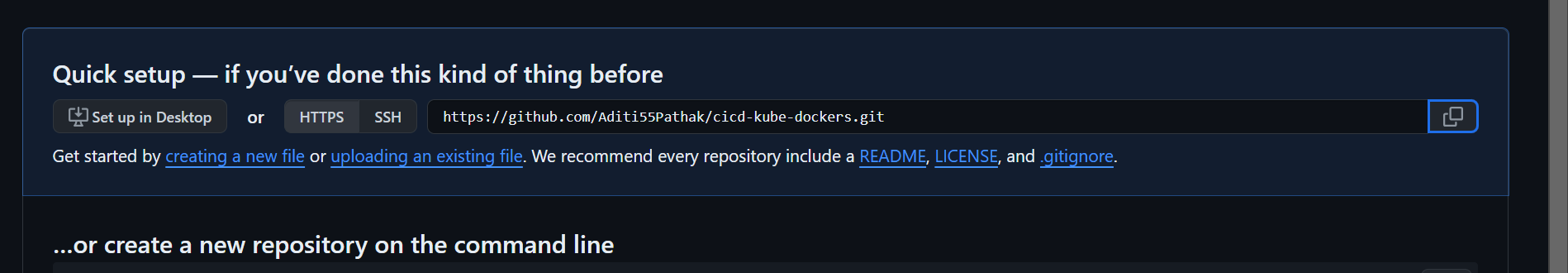
Go to Helm documentation.



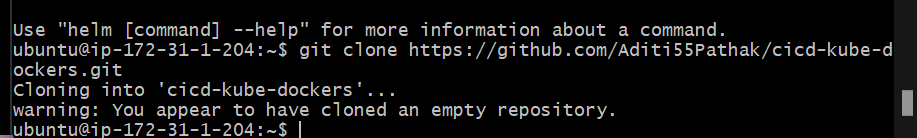




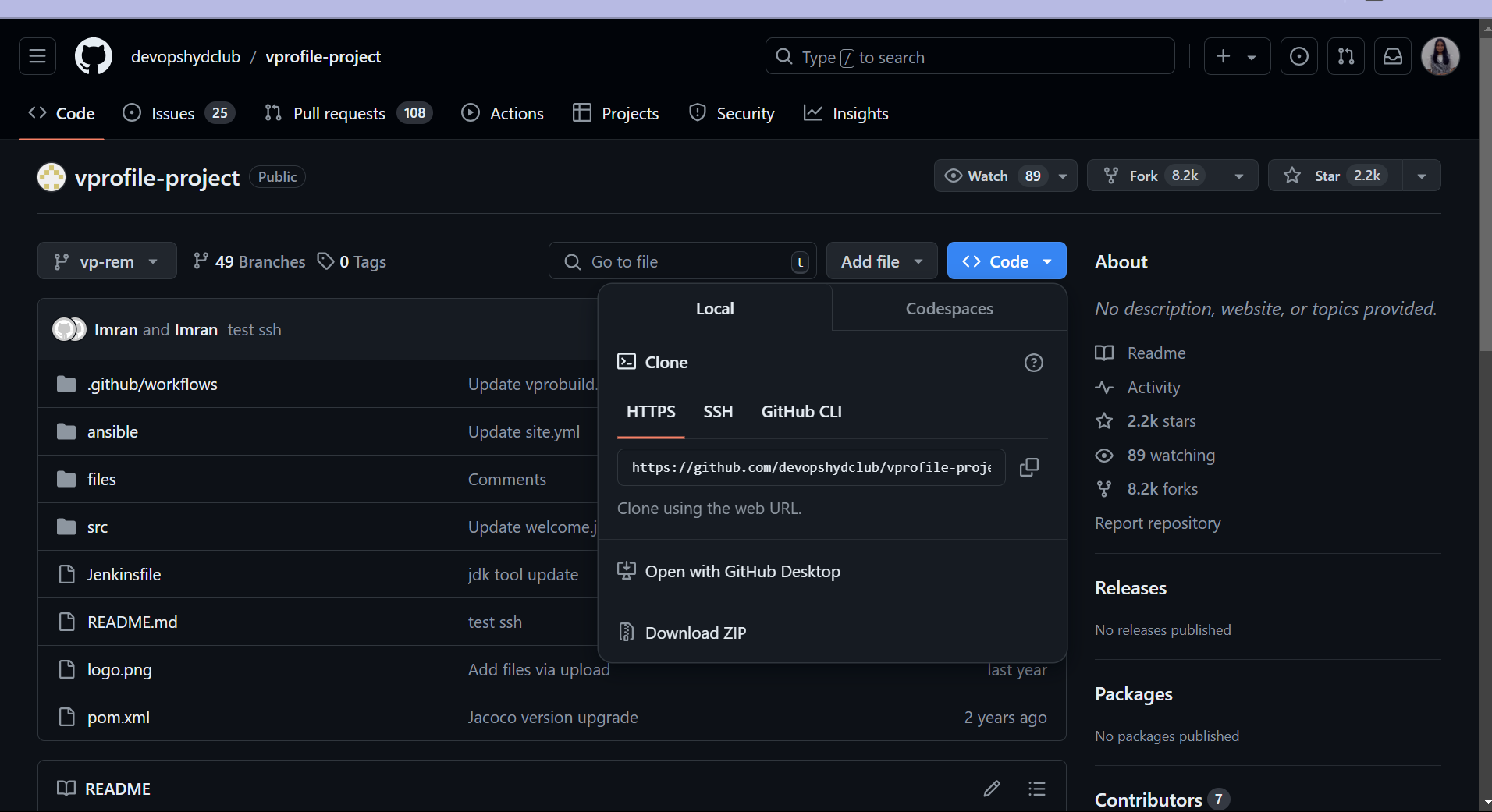
Make your own Git hub Repository

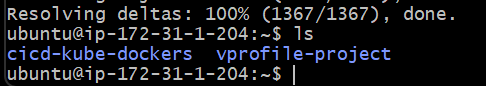


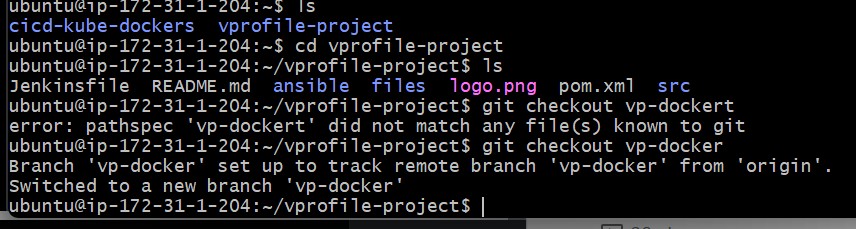
Fetch the URL and clone it.



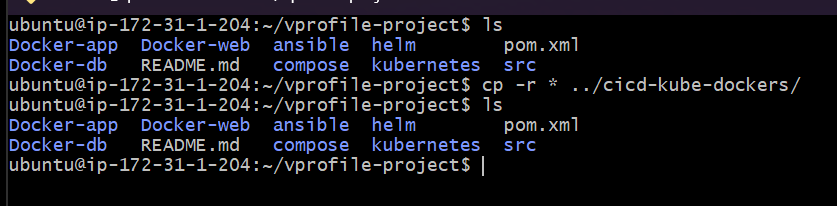
Go to your developer's code and Clone it

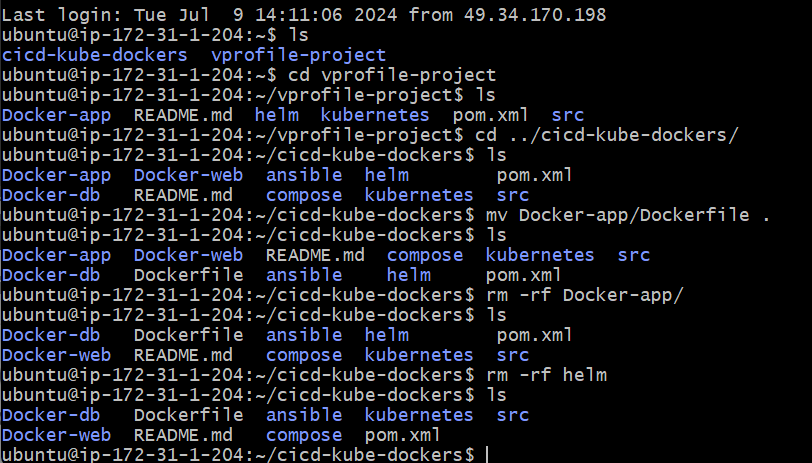


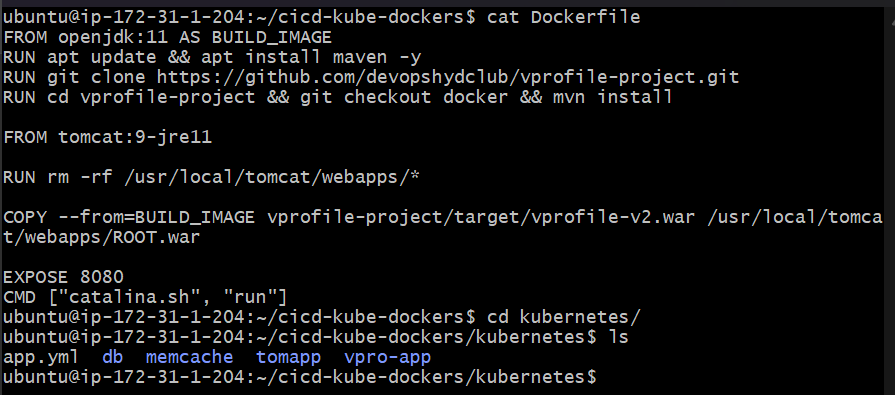


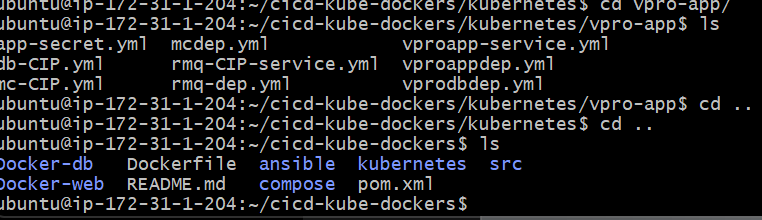


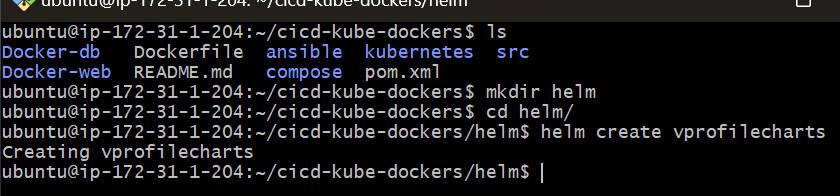
Copy all its content to our repository

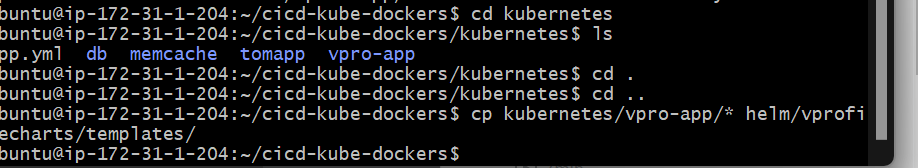


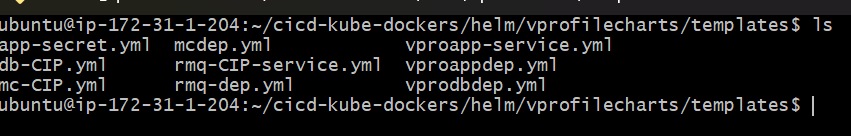




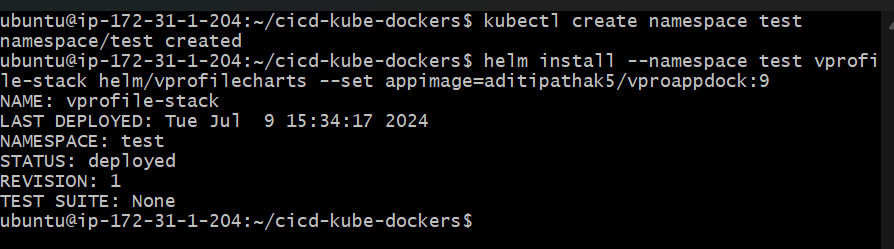


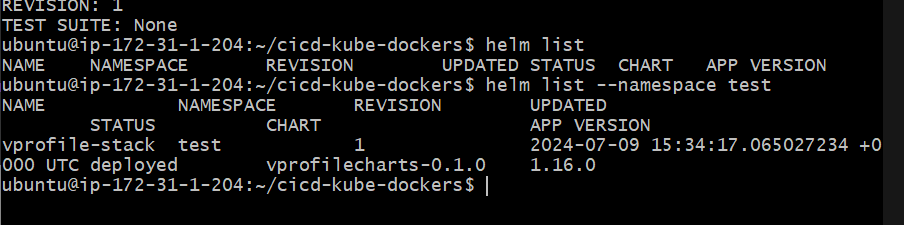


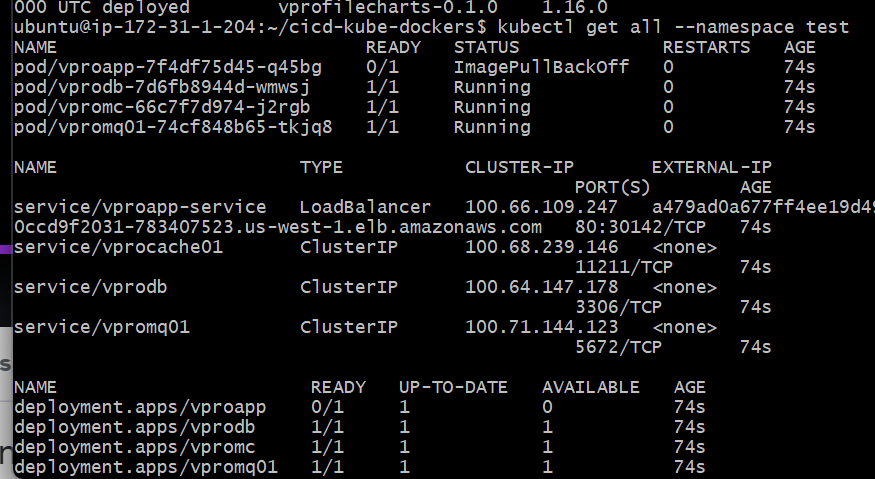


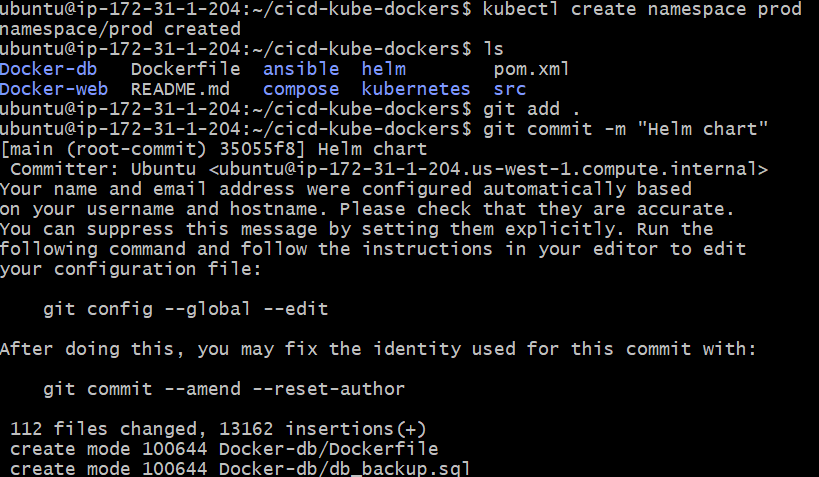


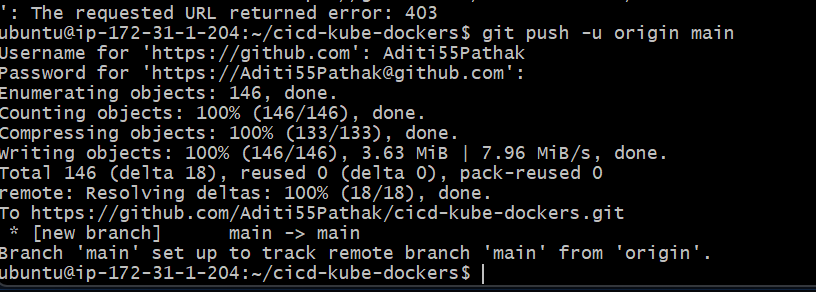






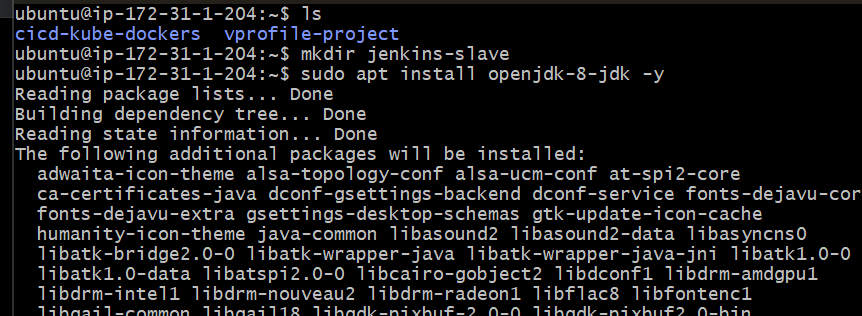


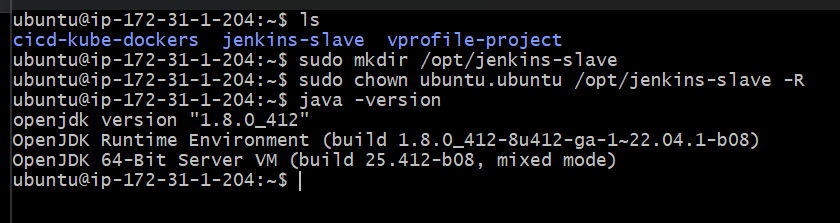




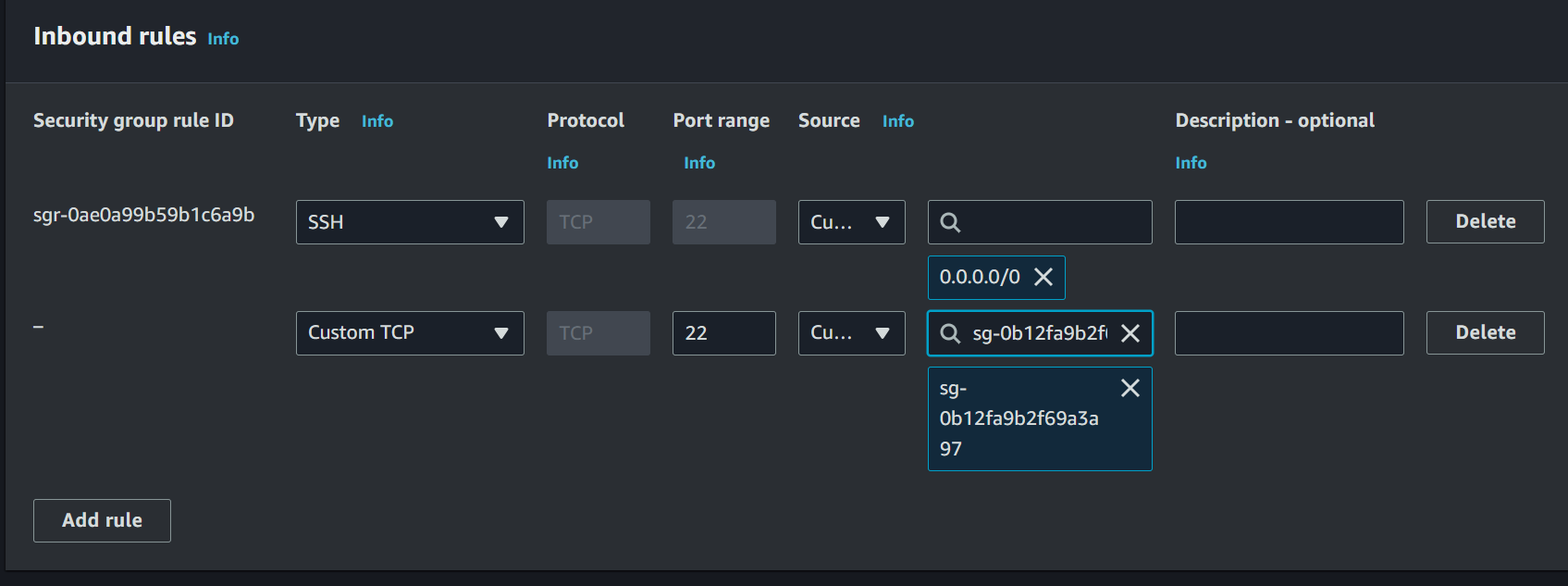
Clone the source code to local repository and write Jenkins File

SSH into kops ec2 instance

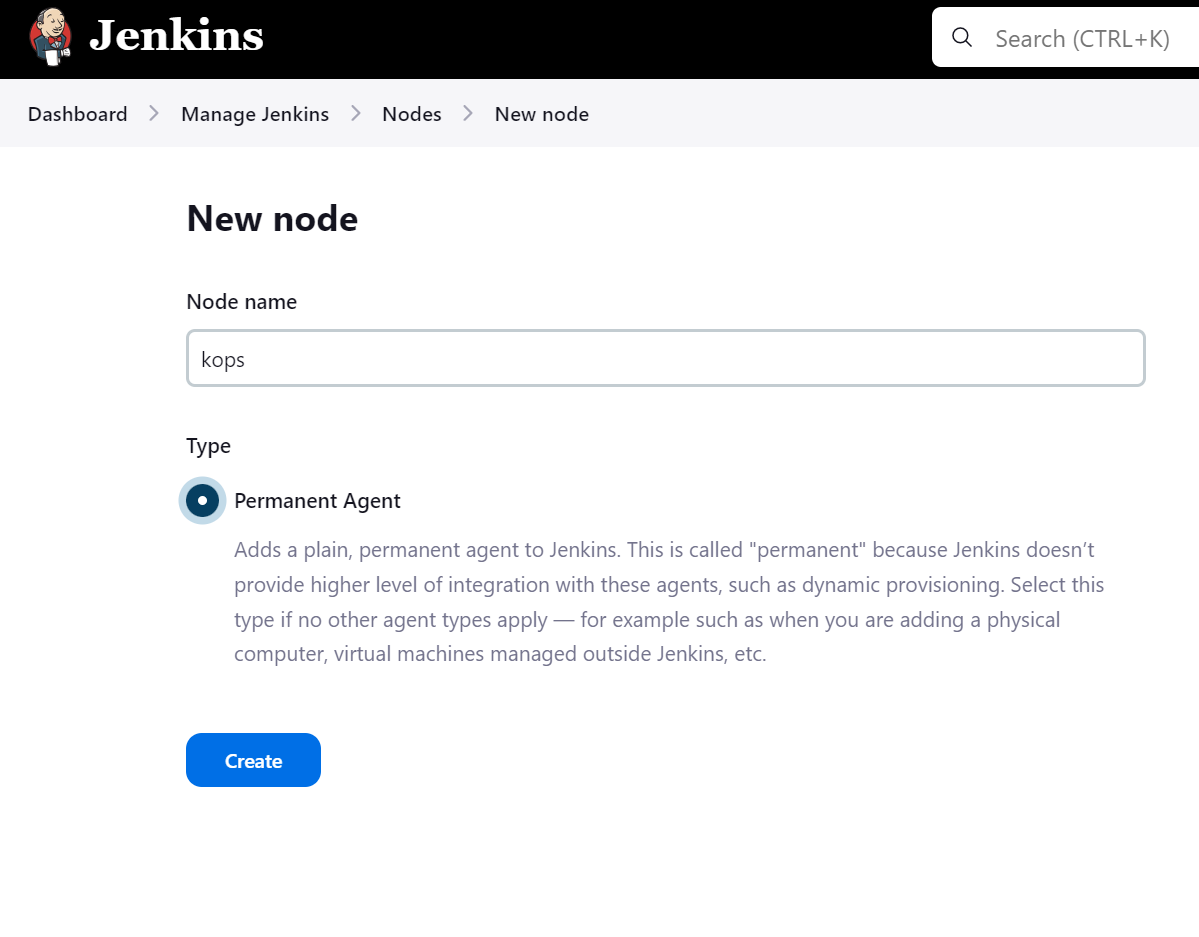




Edit kops Security Group

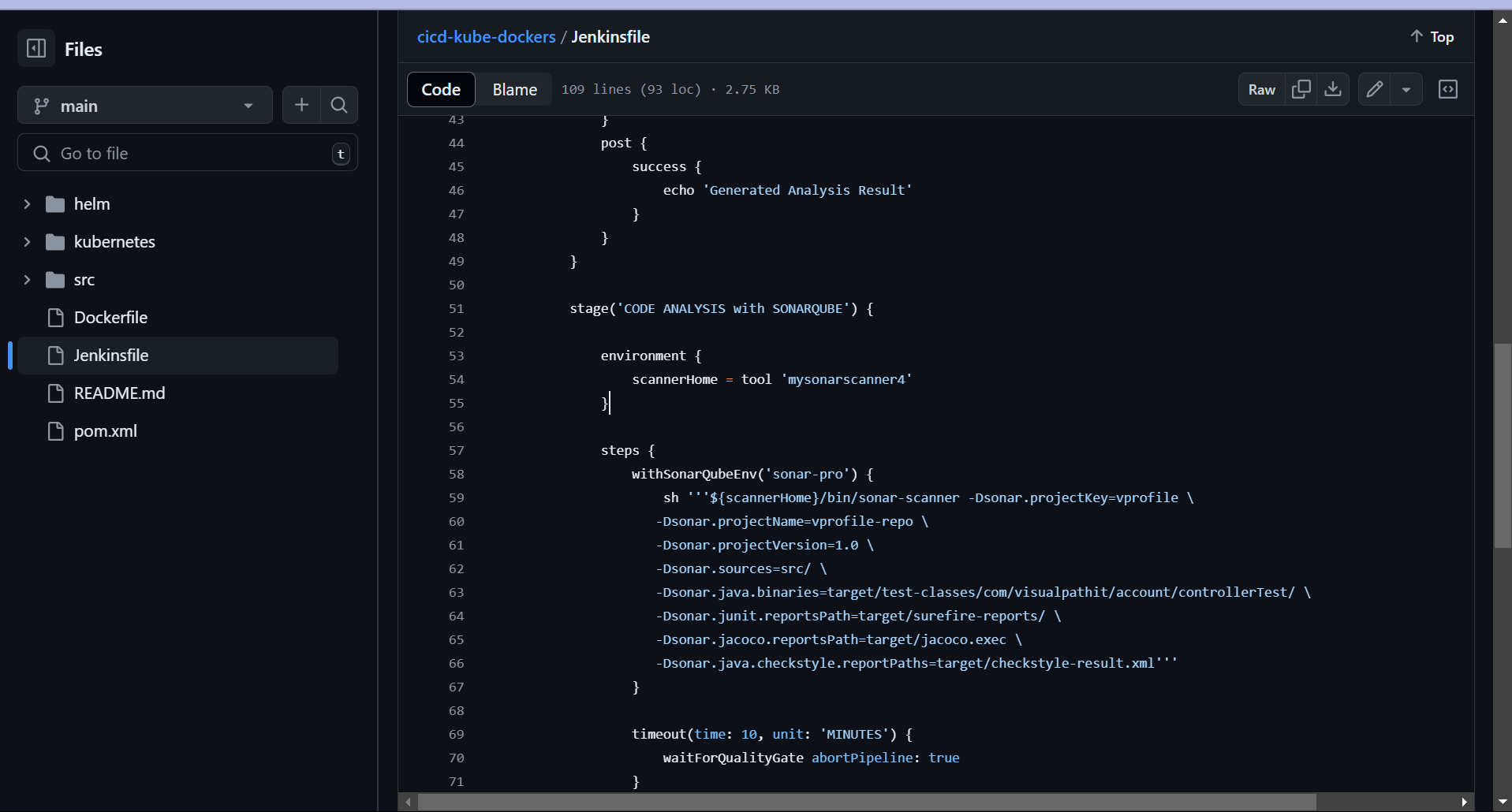


Go to Manage Jenkins-> Manage Nodes and Cloud -> Add new node

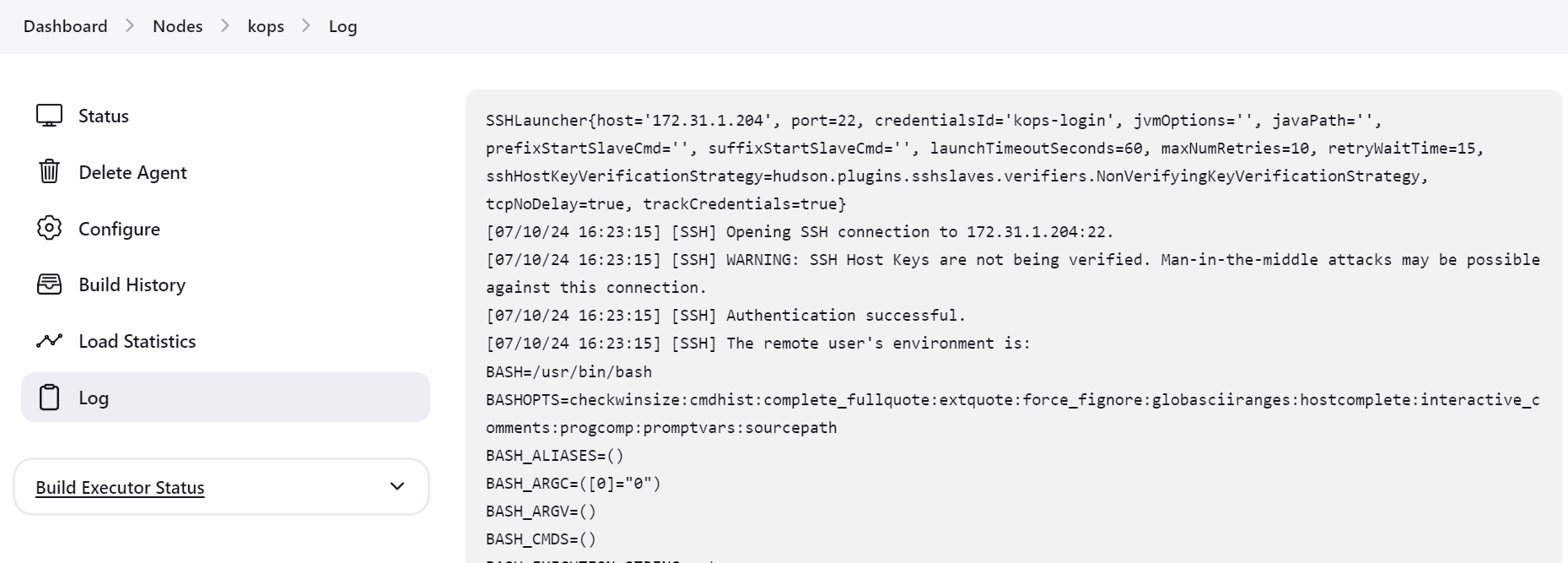


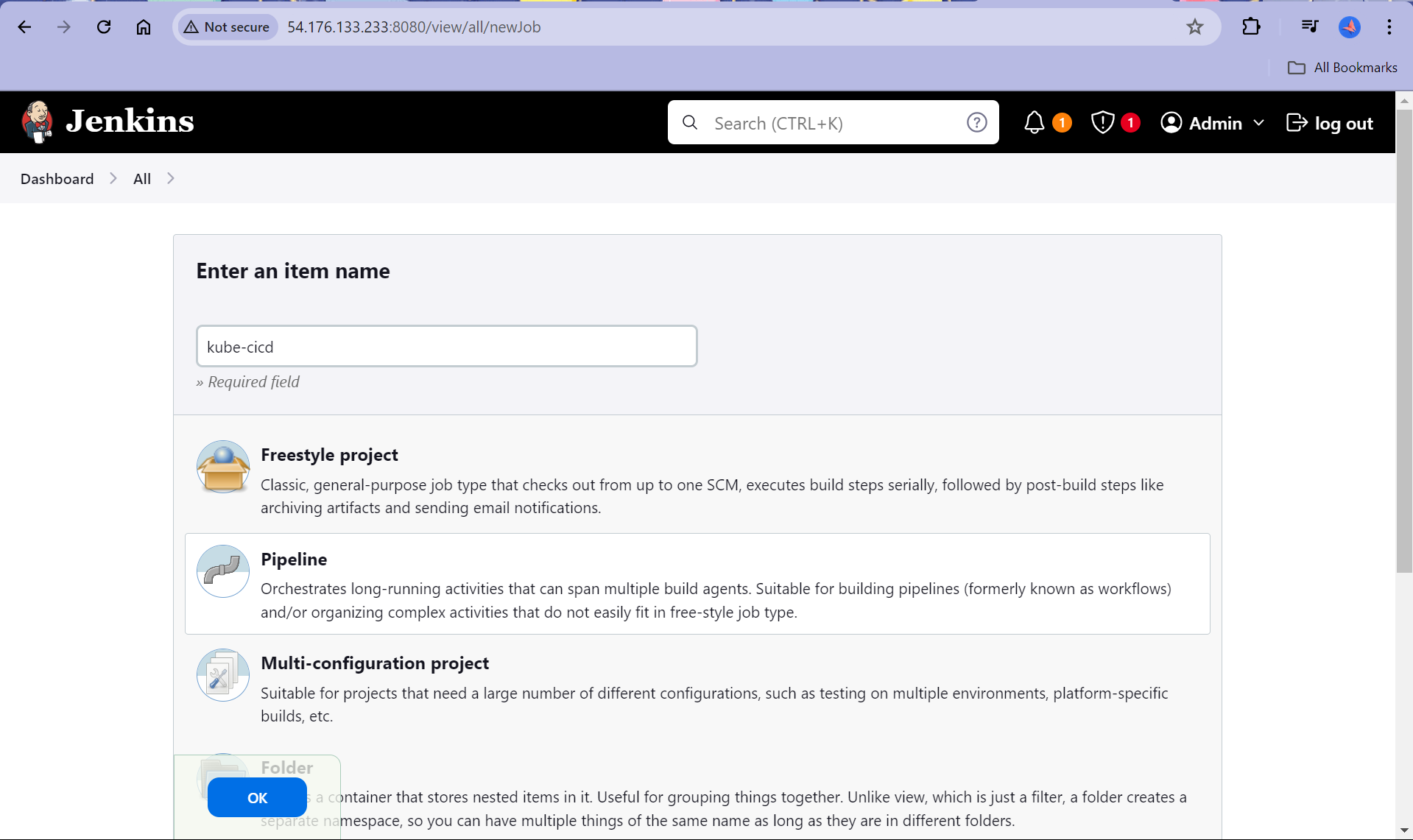


Push Jenkins File to your desired Git Repository



Launch the agent node





Build the pipeline. And You are ready to set.