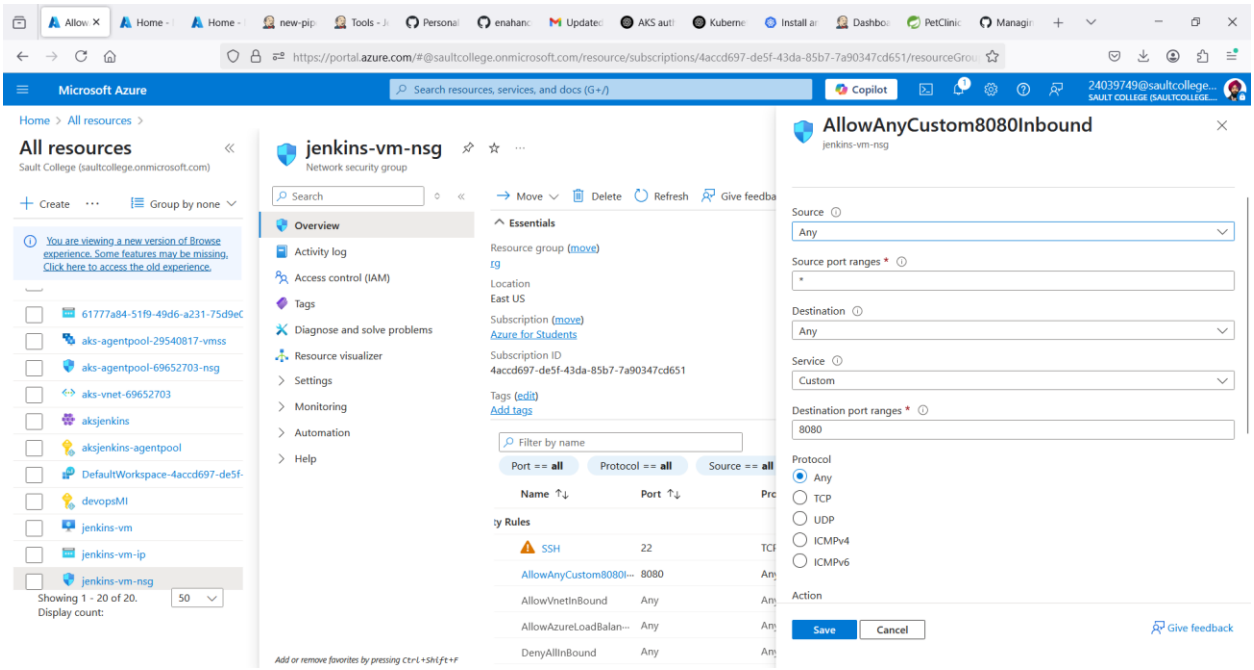
Jenkins Pipeline for Spring Boot Java Application

Contents

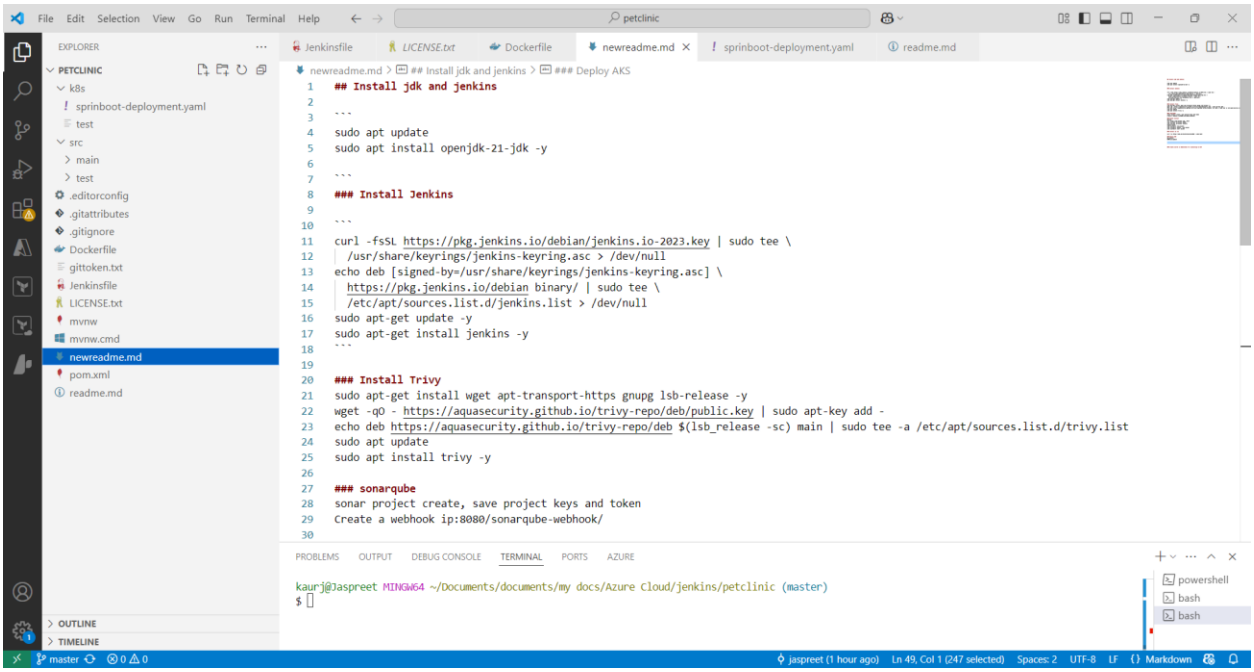
Pre-requisite

Create a vm, install java-jdk, install Jenkins, trivy. Install stage view plugin in jenkins.

Open port 8080

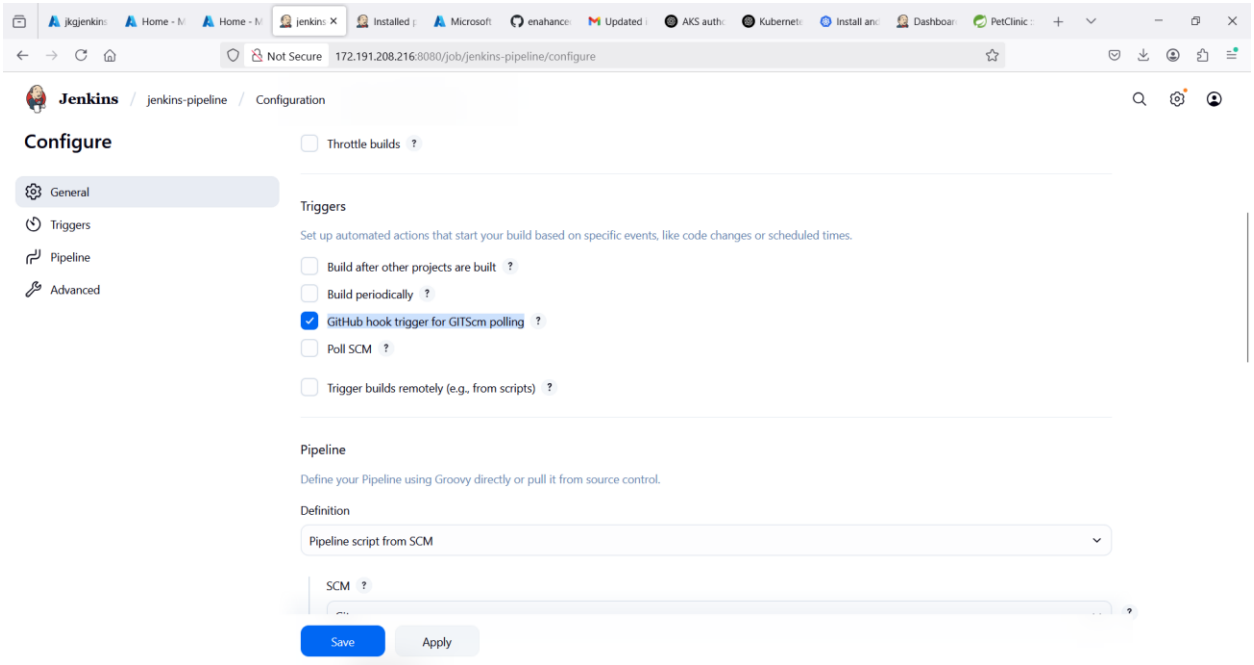


In local pc, clone the petclinic project. Open in vscode

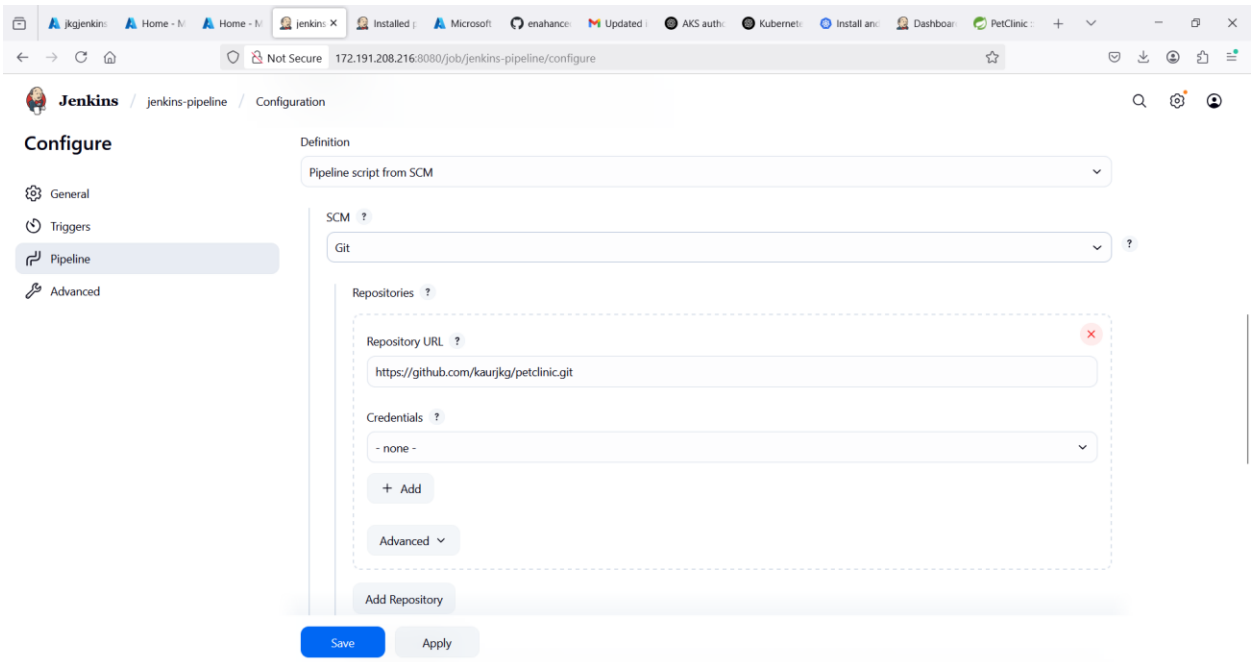


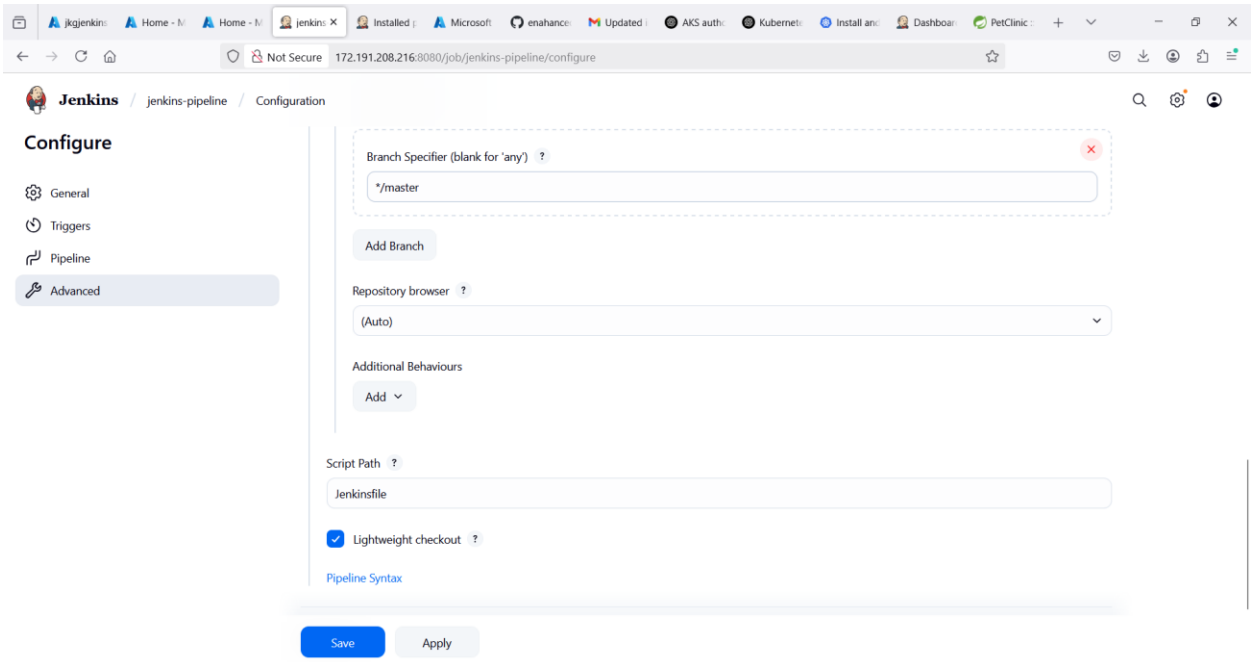
Initialize Jenkins pipeline

Create a Jenkins pipline. Set the pipeline trigger to GitHub hook trigger for GITScm polling

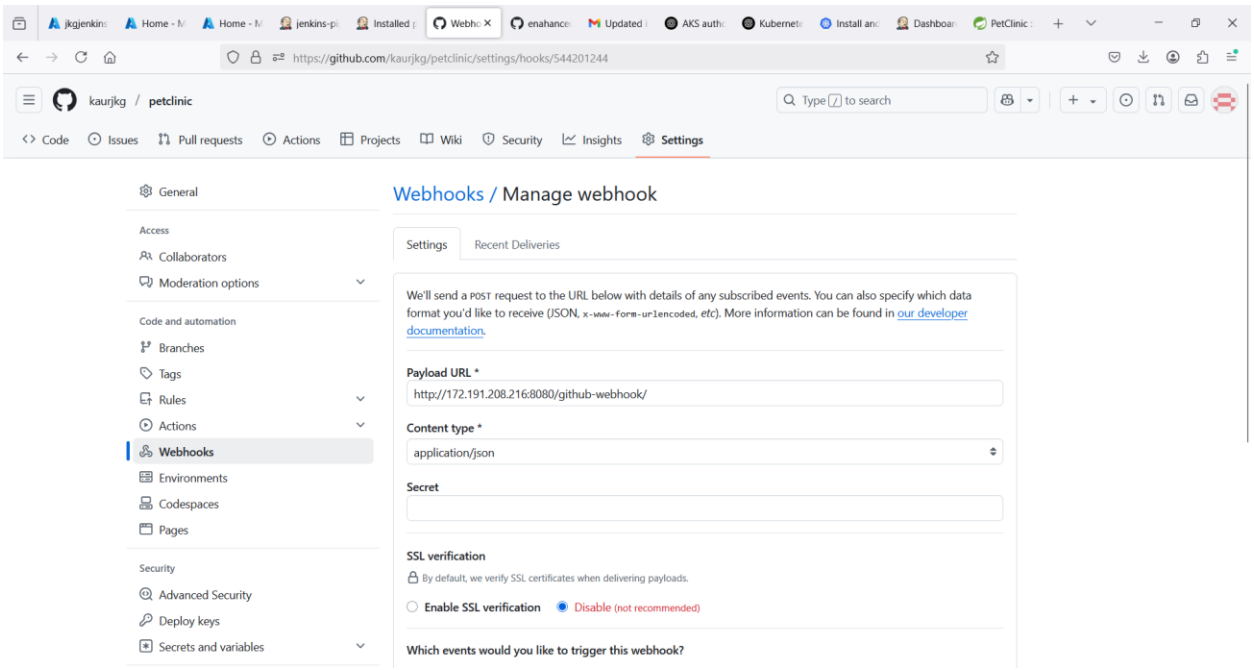


Get the pipeline script from Source code management GIT.



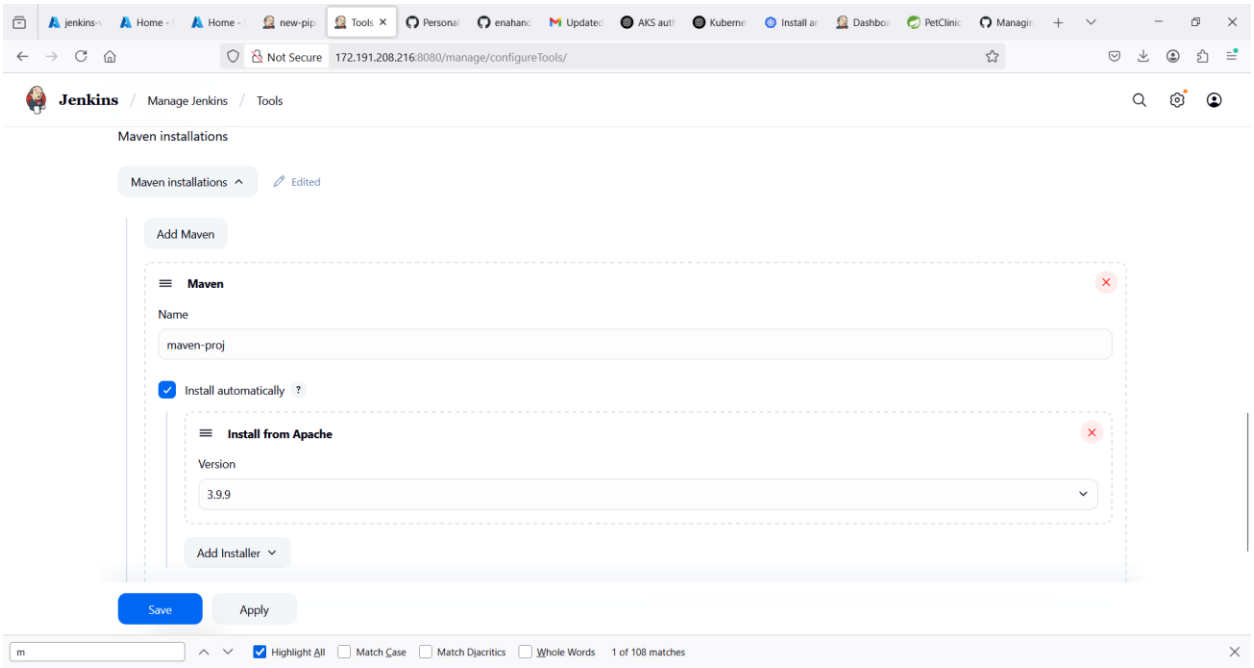


Add github webhook trigger for Jenkins pipeline, so that github can trigger the pipeline when code is pushed to the project repo

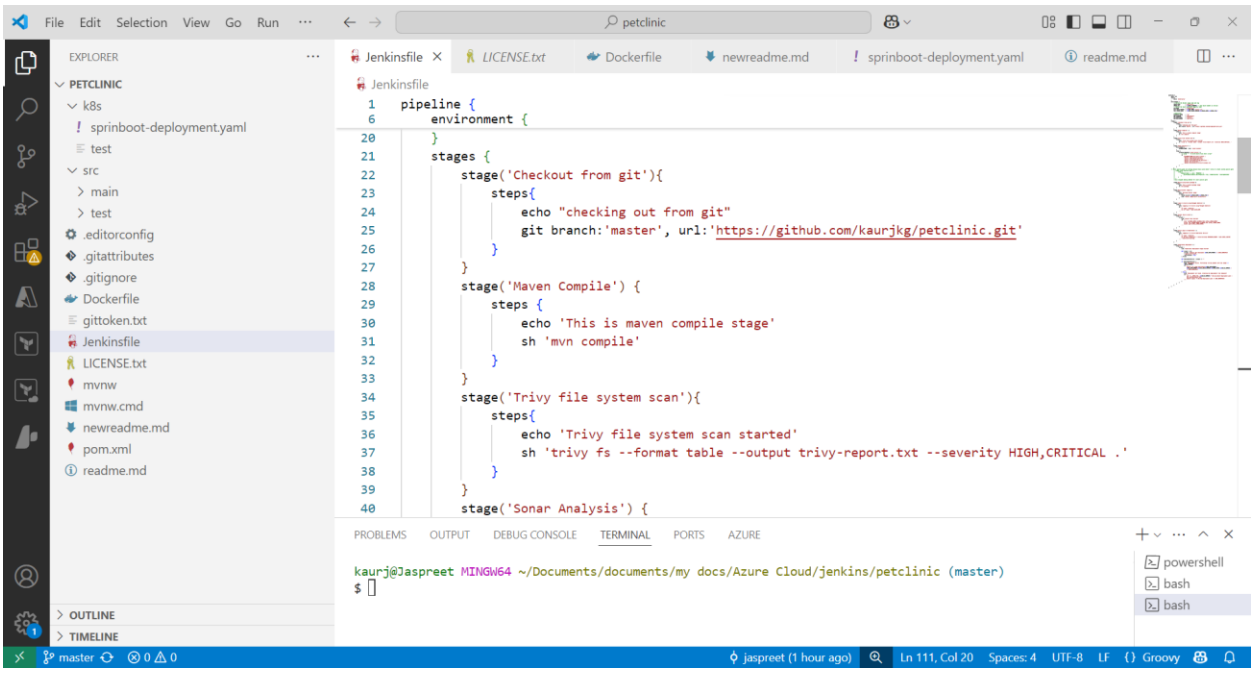


Add stages

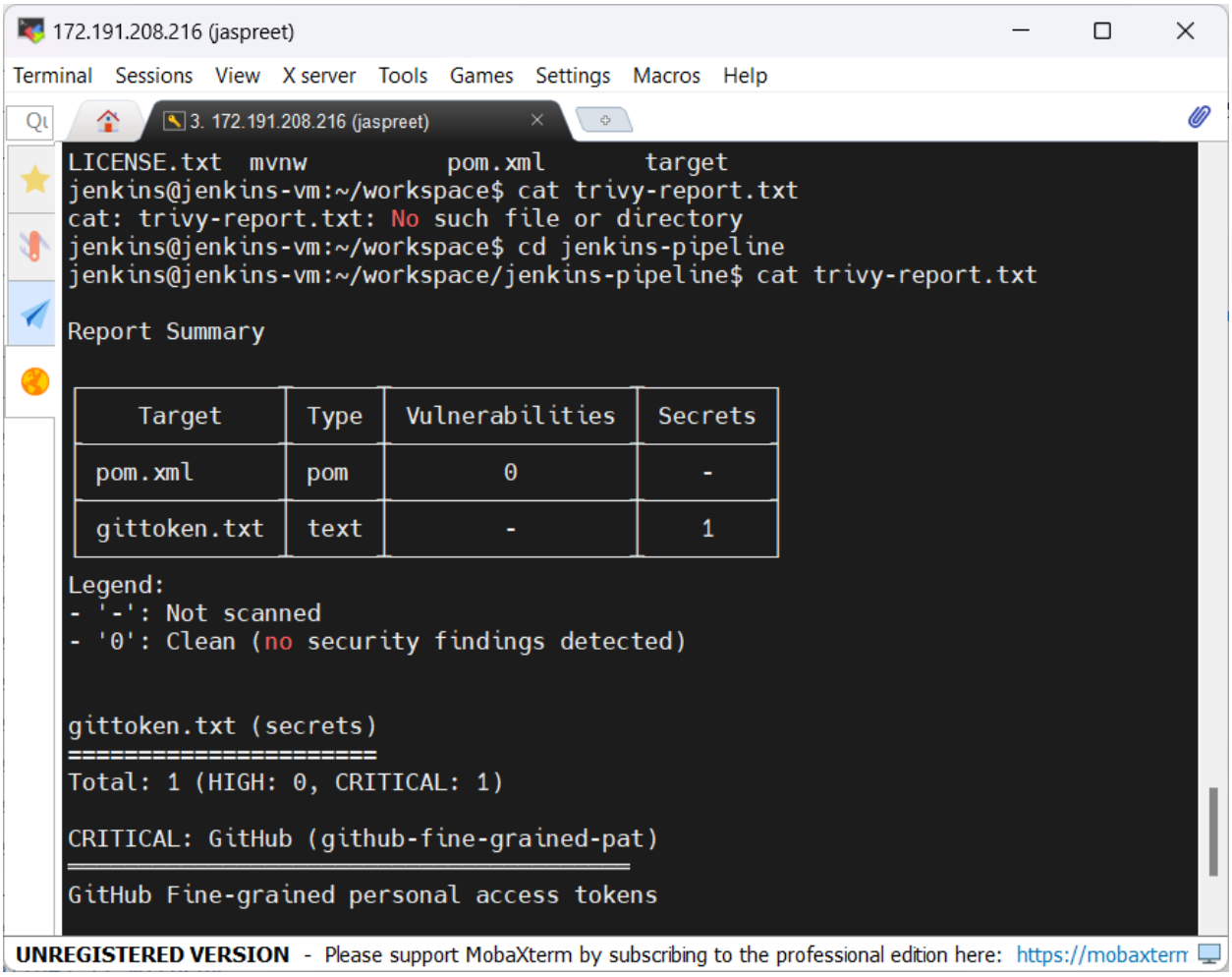
In Jenkins server-> manage Jenkins-> add maven. This name will be used in jenkinsfile pipeline code.



Add stages to clone github code on Jenkins agent (VM) using checkout stage. Add another stage to compile the code using maven. Third stage for trivy which is used to scan code for any security threats.

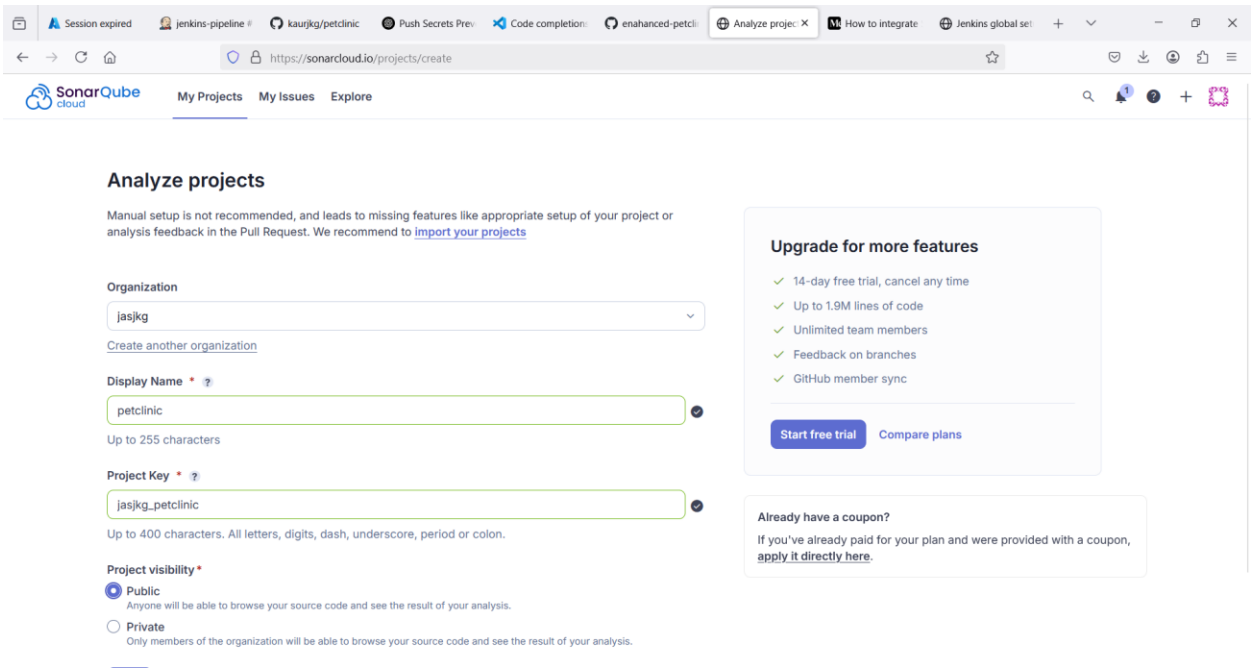


Check trivy output in Jenkins vm agent by logging as sudo su - , su – jenkins.

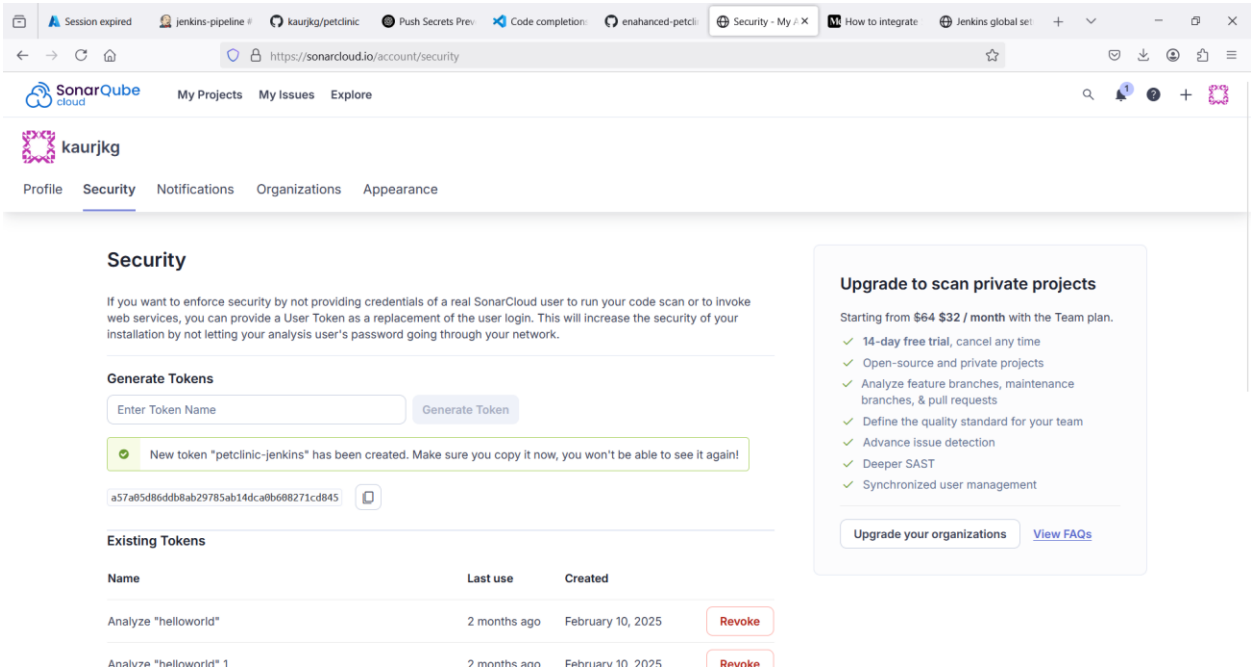


Sonar Integration

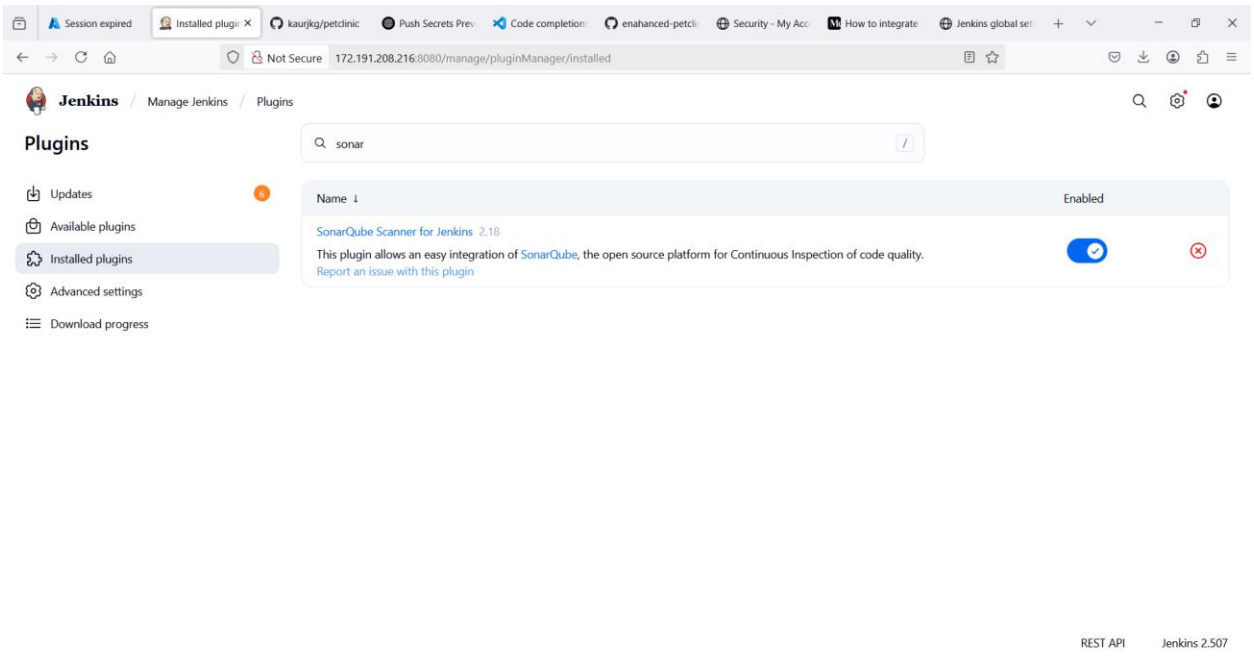
Create Sonarqube project and save information



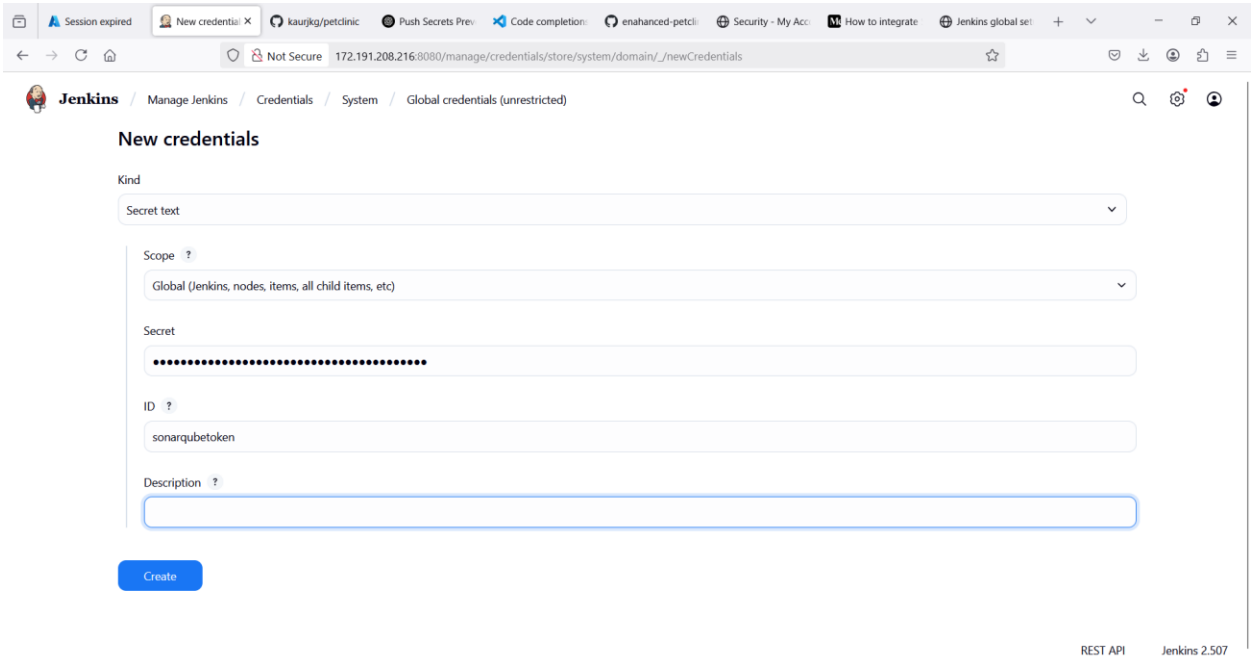
Create a token



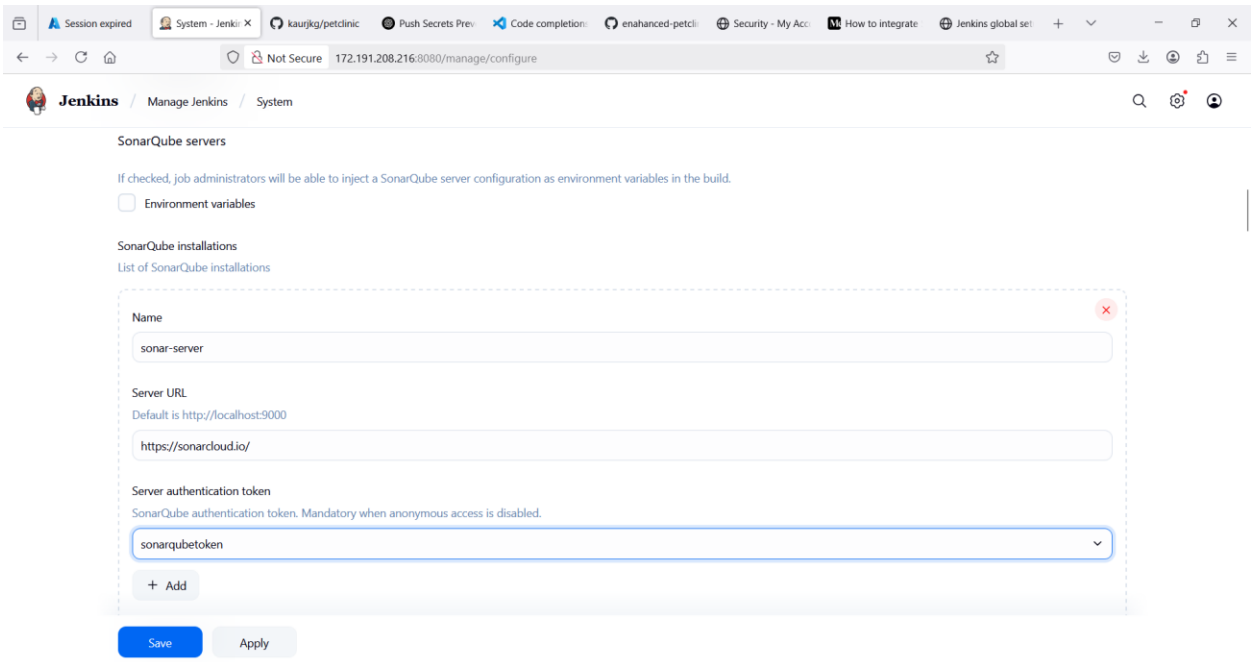
In Jenkins->Manage Jenkins-> Install plugin Sonarqube scanner



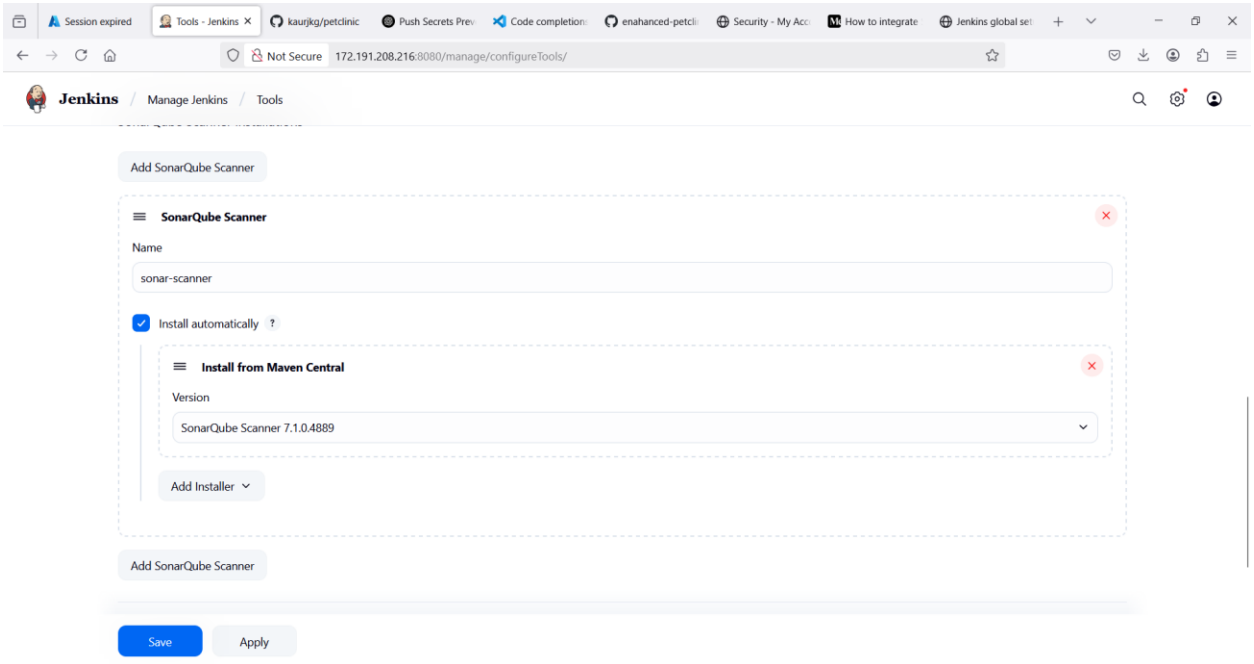
Add token information in Jenkins Global Credentials



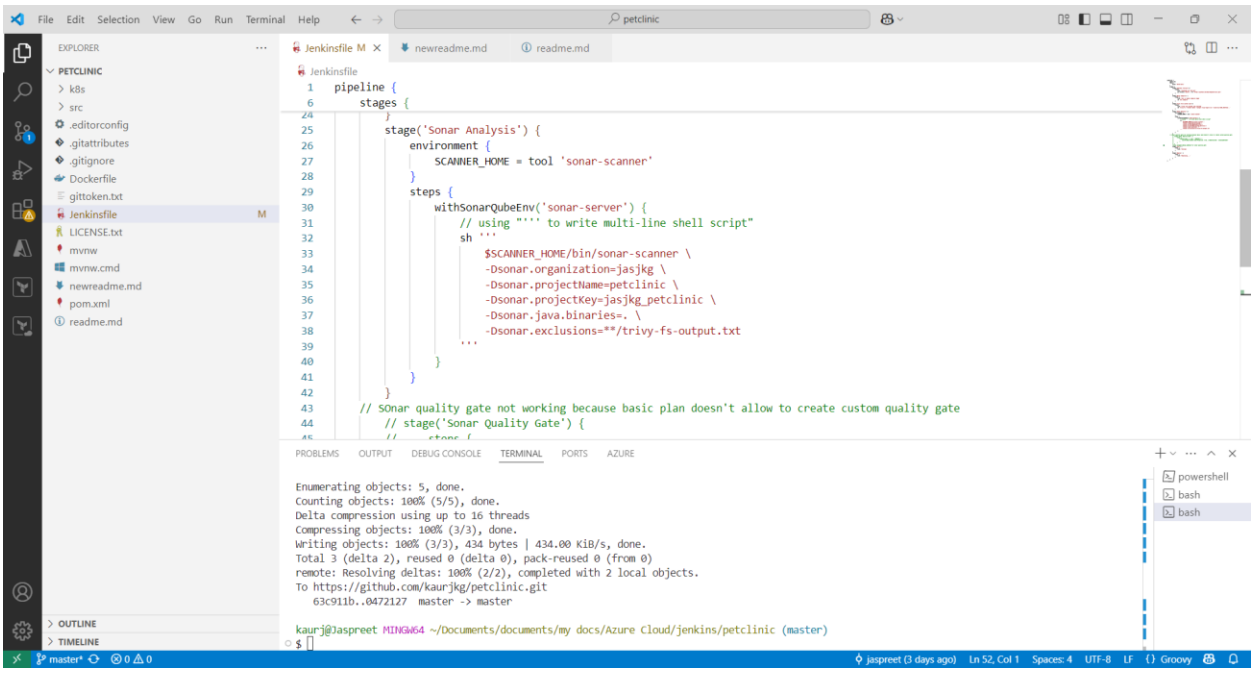
Server Settings in System



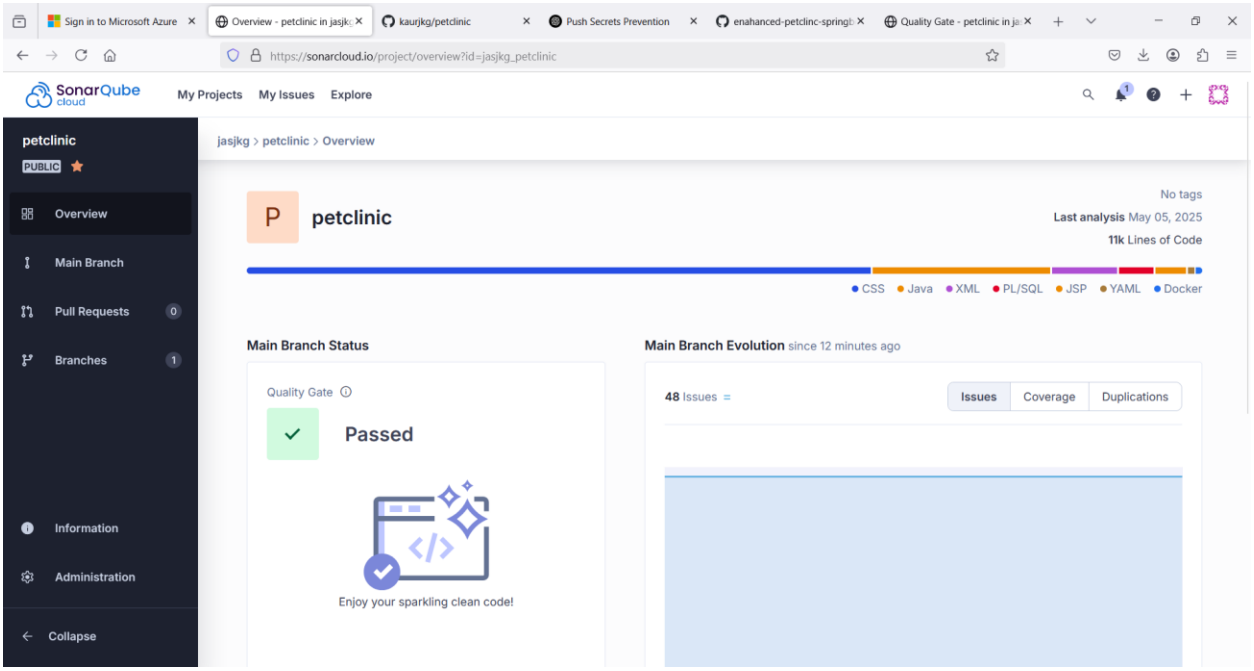
Add sonar scanner client in Tools



Added Pipeline code

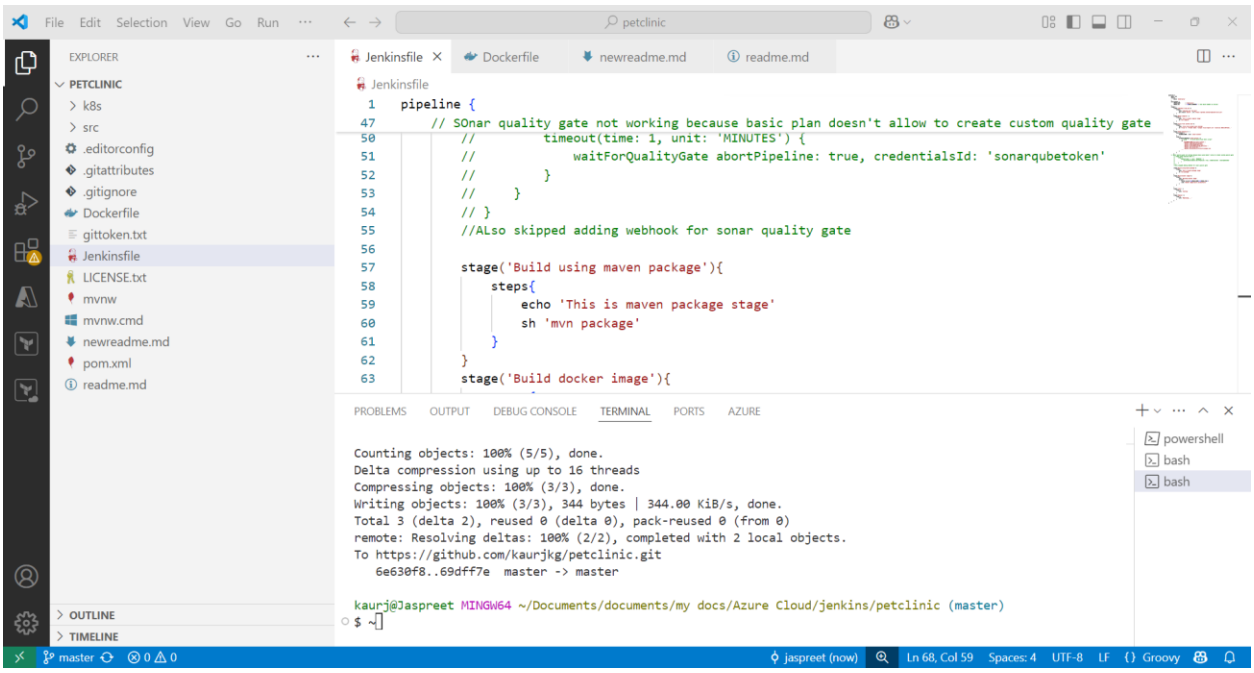


After pipeline ran automatically, sonar tests passed

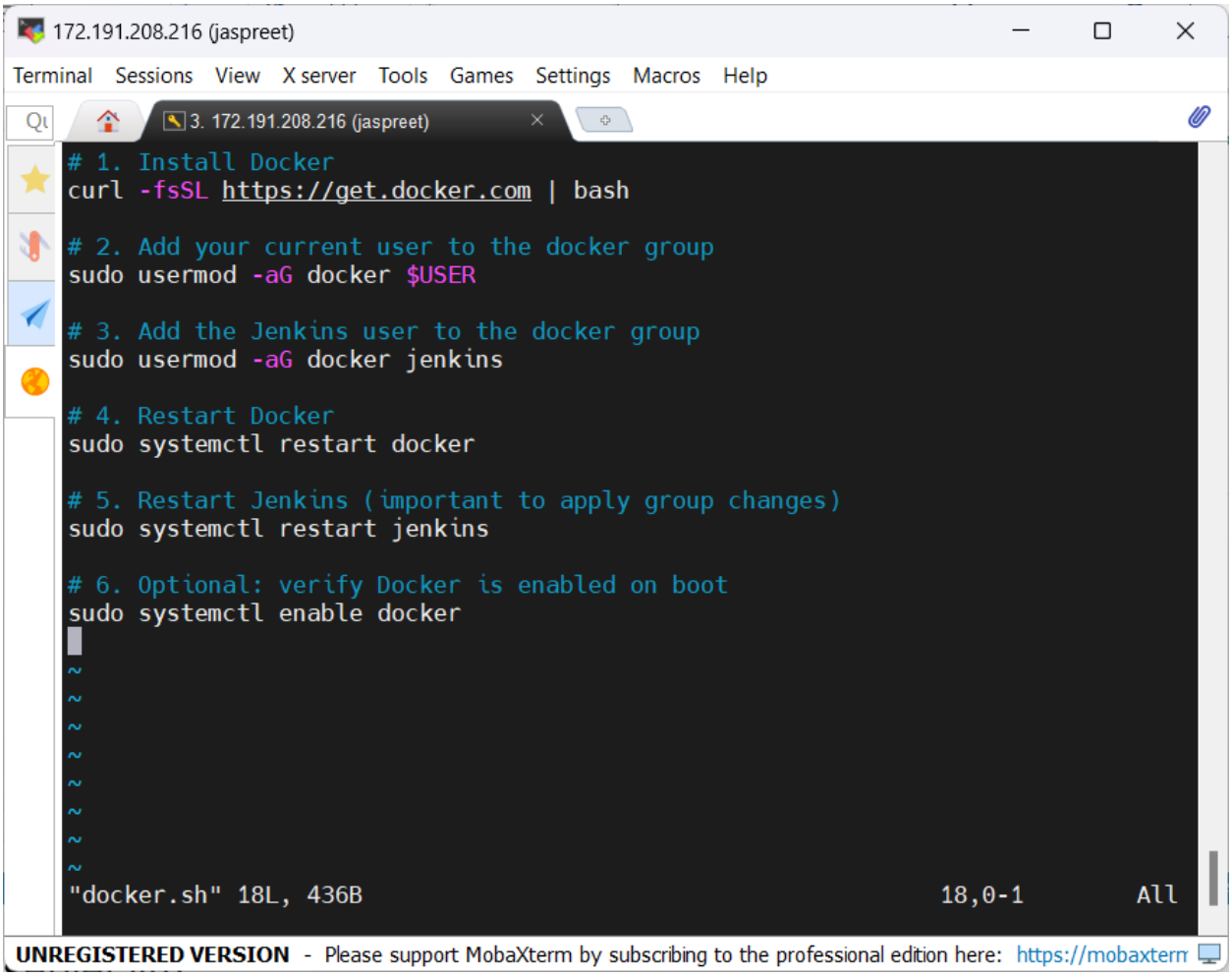


Build JAR and create Docker image

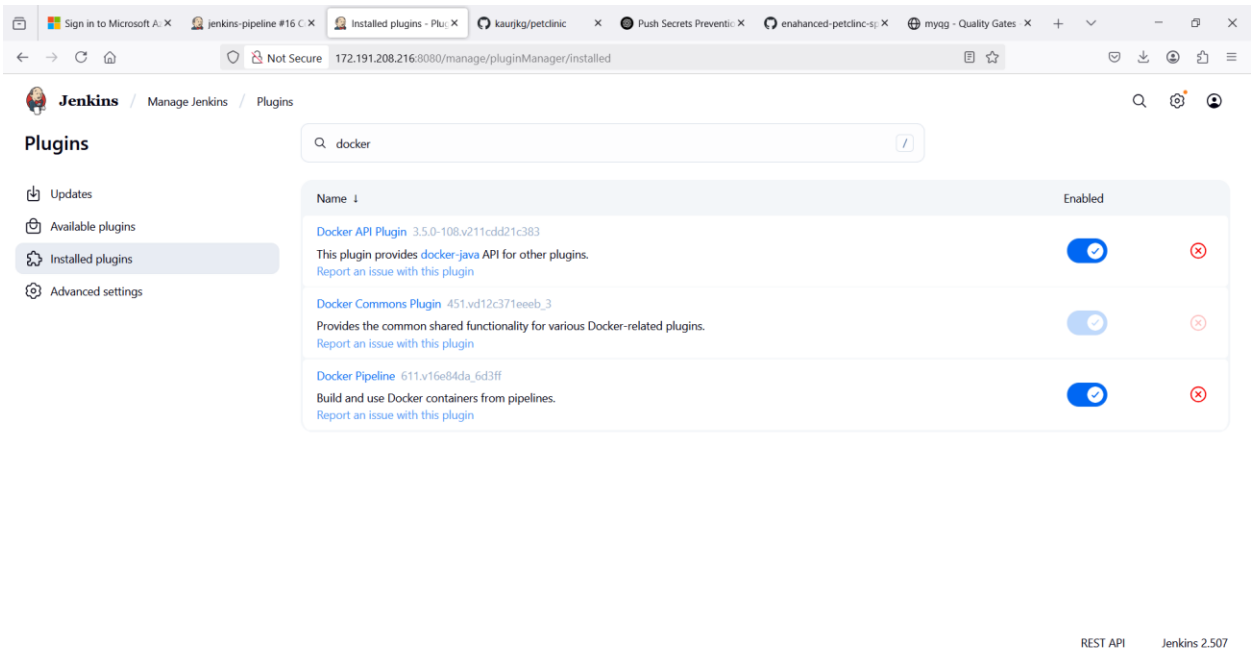
Add maven package step for compiling project and building war file



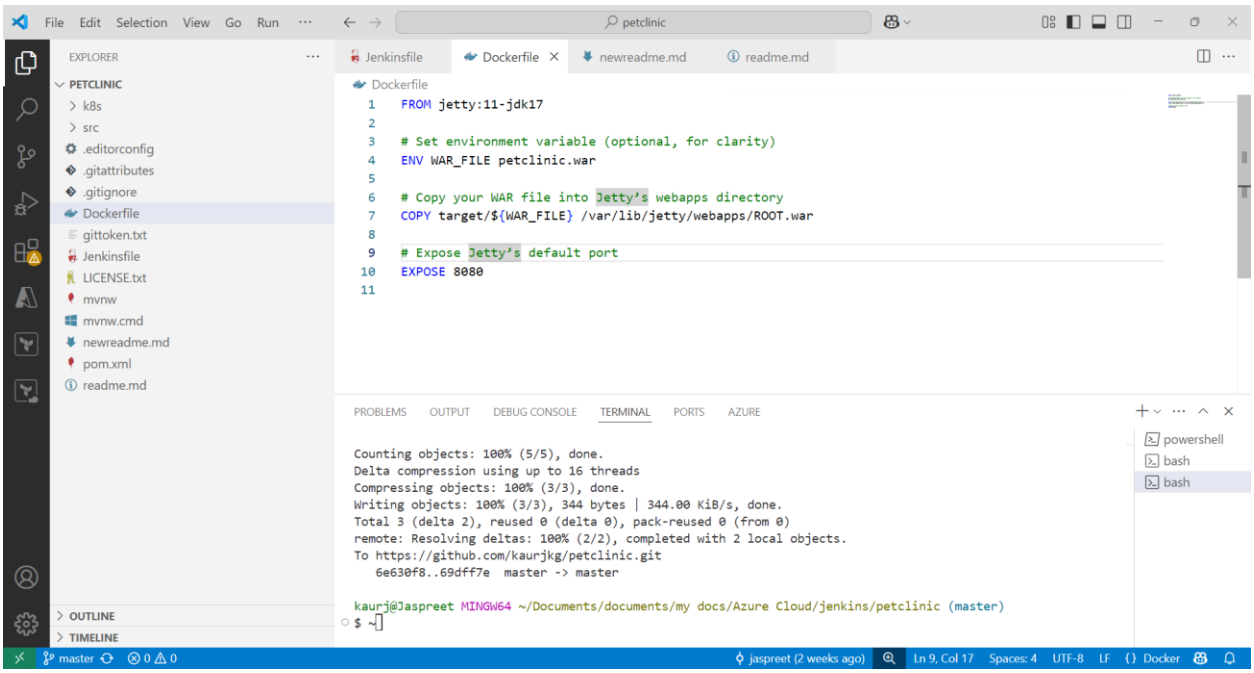
Install Docker in server vm



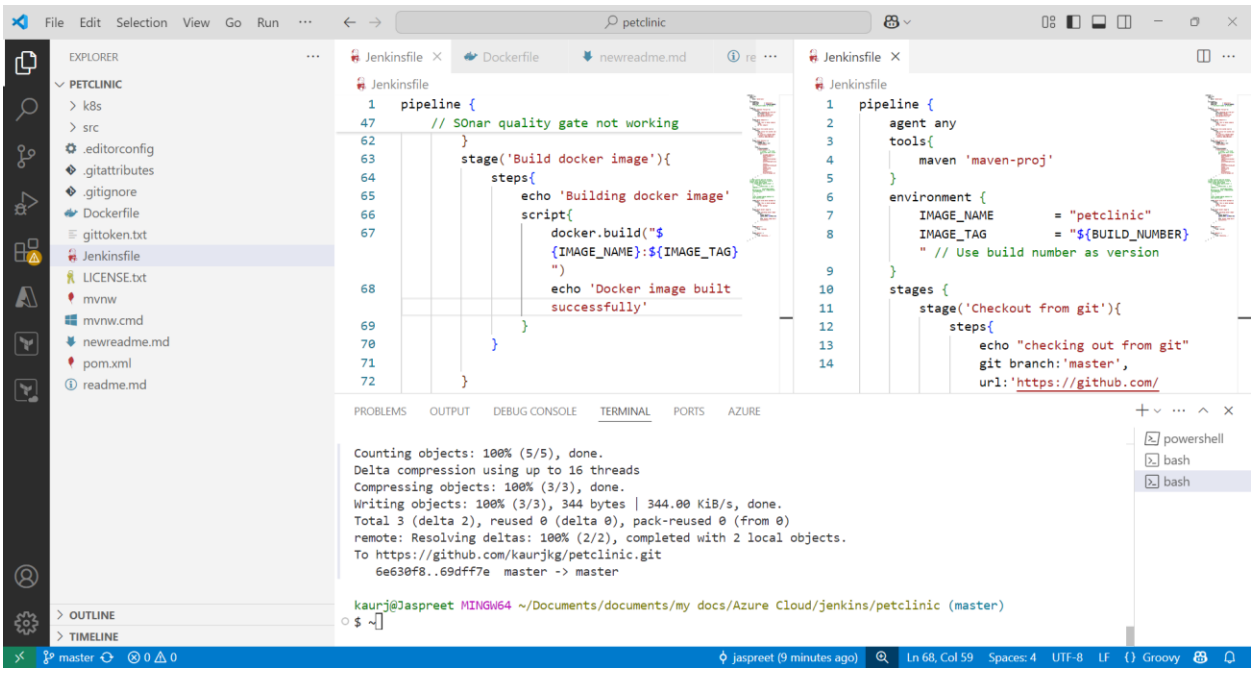
Install docker commons and docker pipeline plugins in Jenkins



Create dockerfile in project root folder

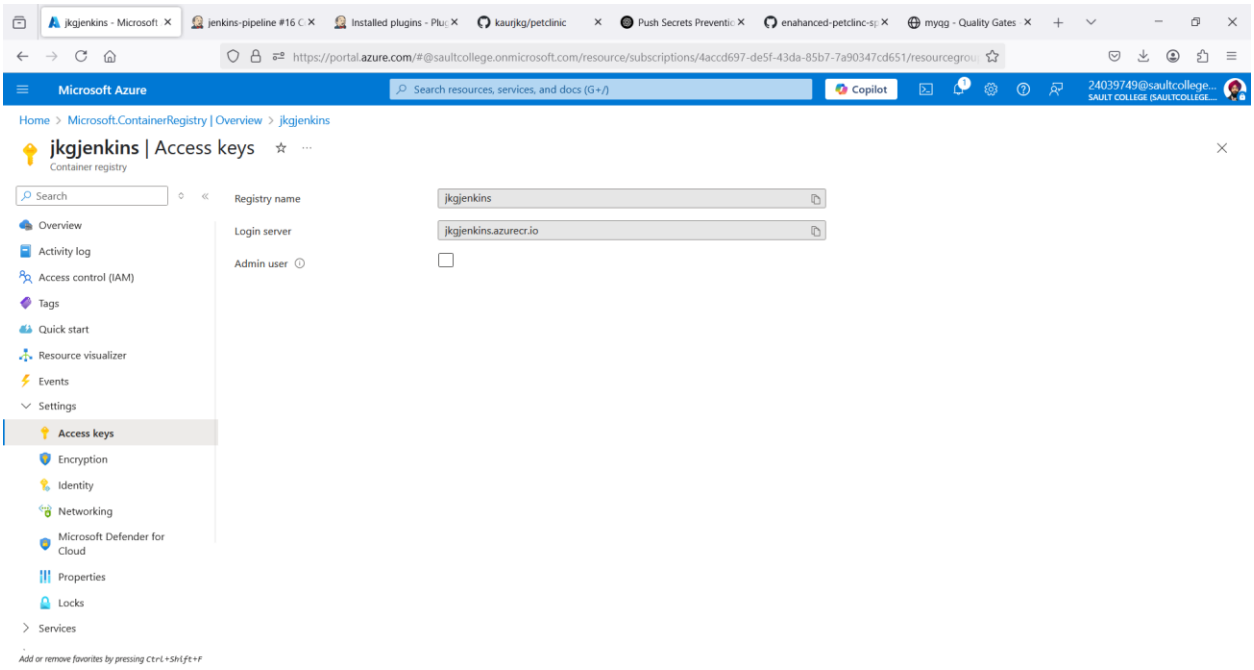


Add docker build script and environment variables for image name and tag name



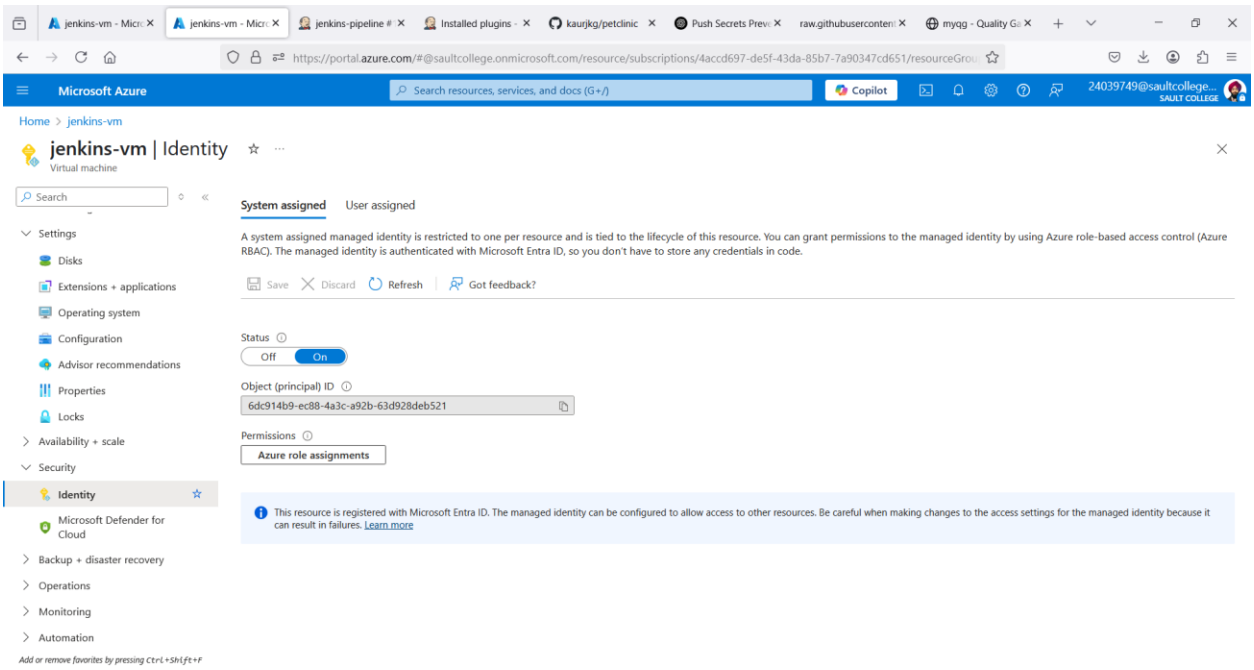
Adding ACR

Azure Container registry with its credentials

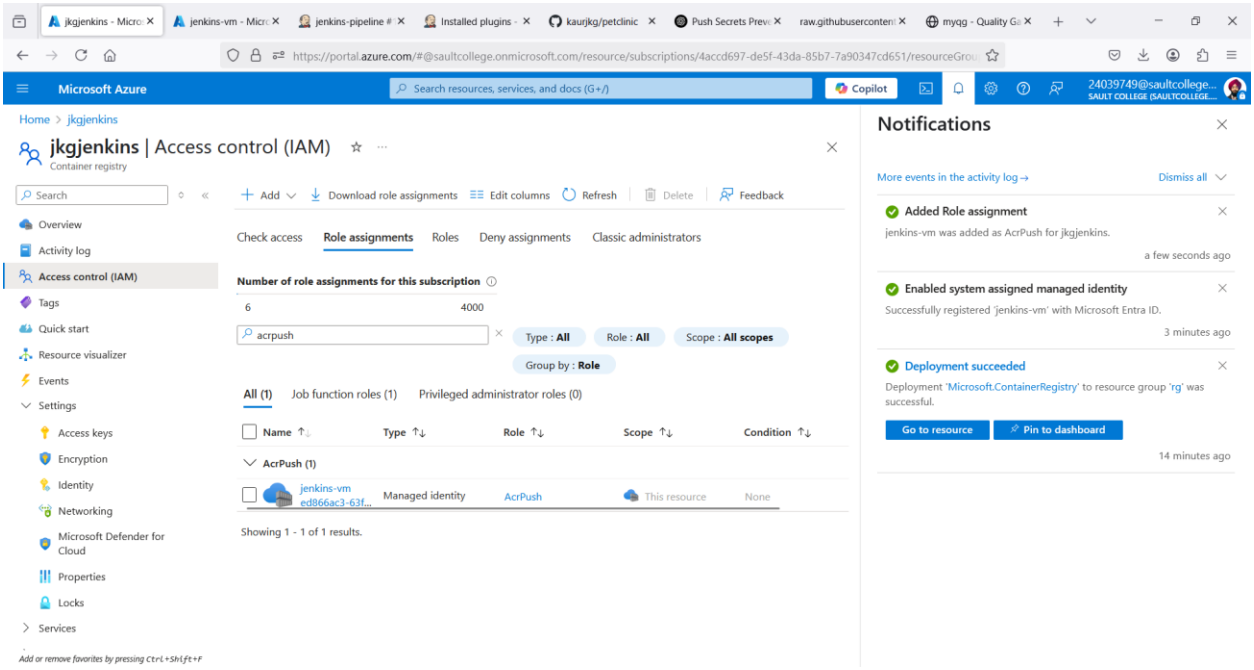


Push a Docker image to Azure Container Registry (ACR) using Managed Identity through the Azure Portal — no secrets or credentials needed

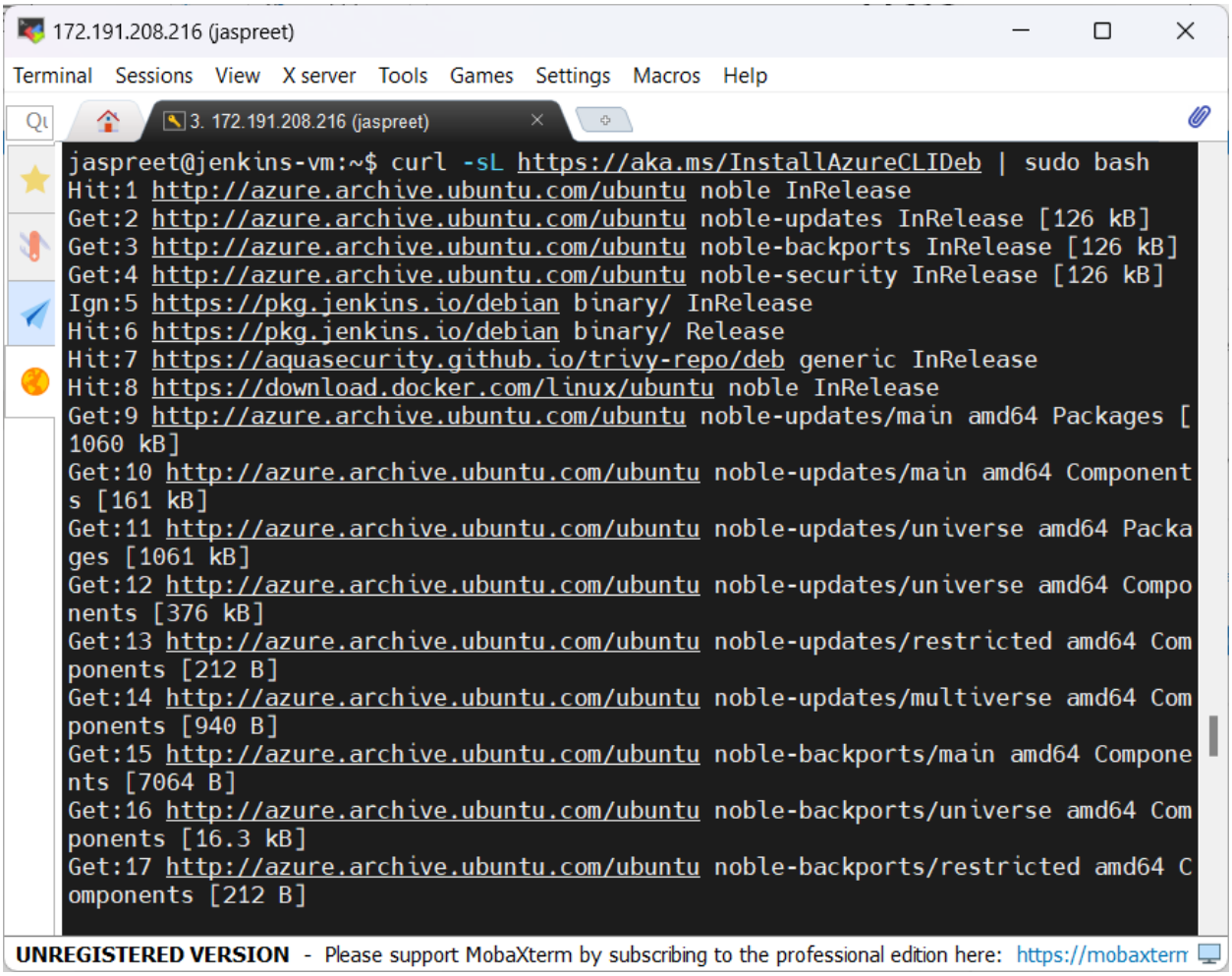
Assign System assigned identity to the Jenkins-vm



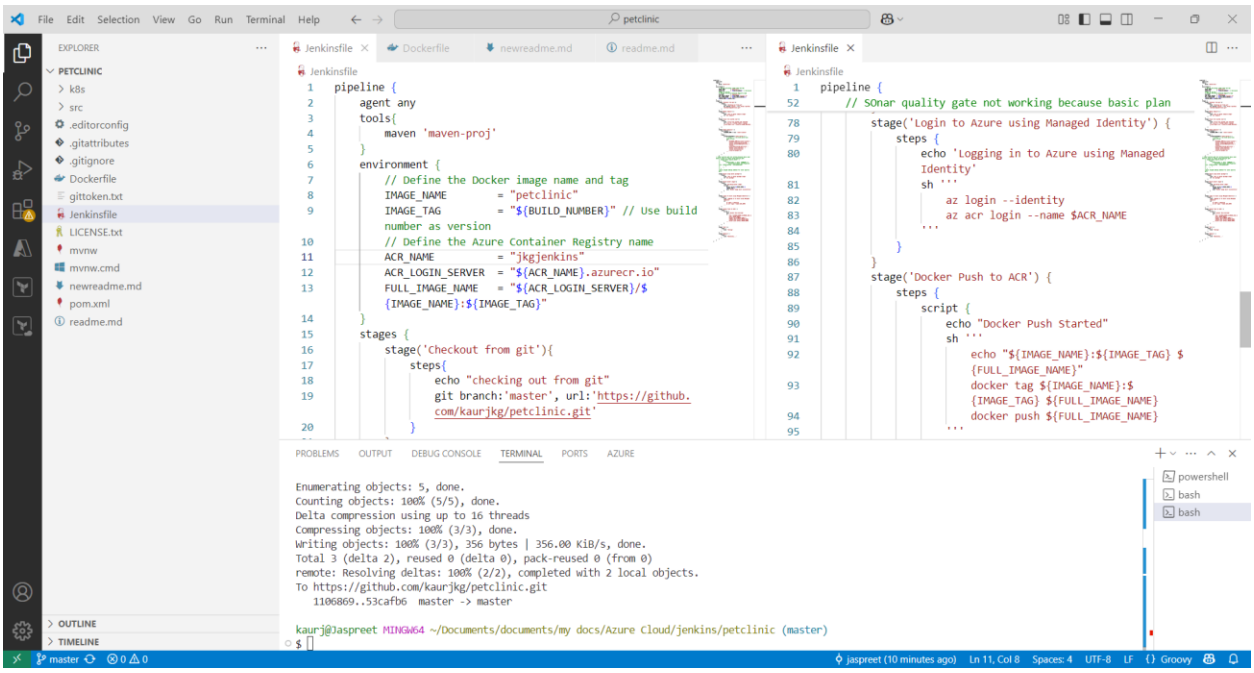
Add acrpush role of Azure container registry to Jenkins-vm



Install azure cli on Jenkins-vm server



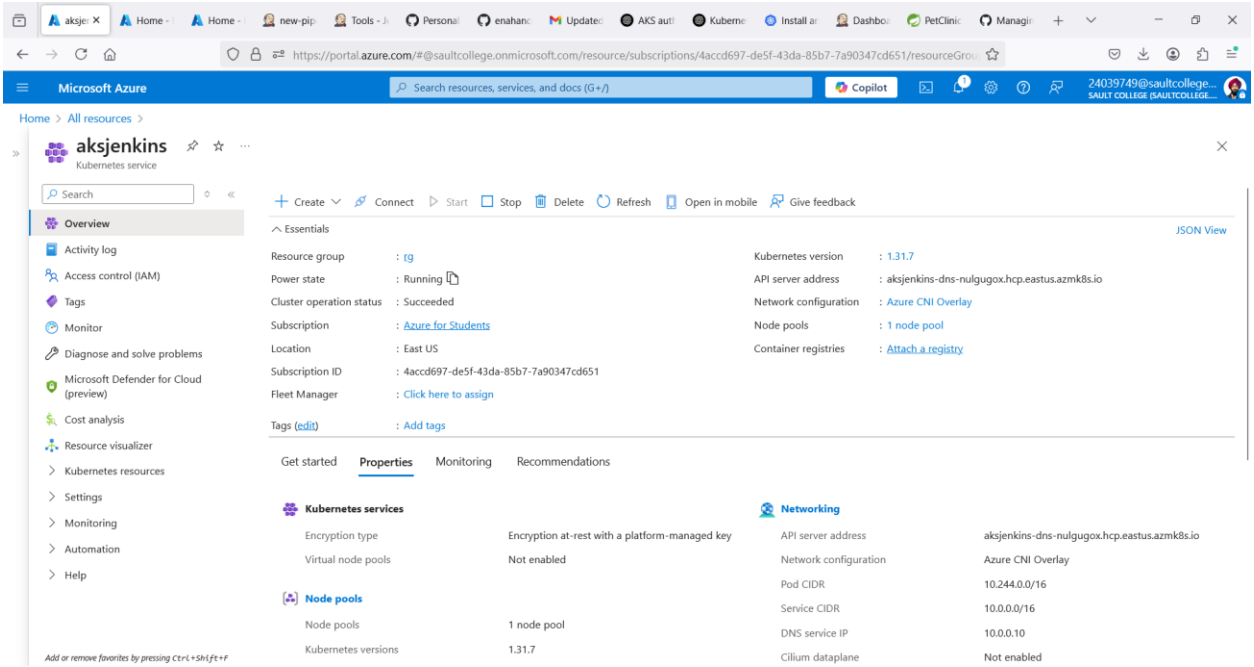
Add az login and docker push stages to jenkinsfile



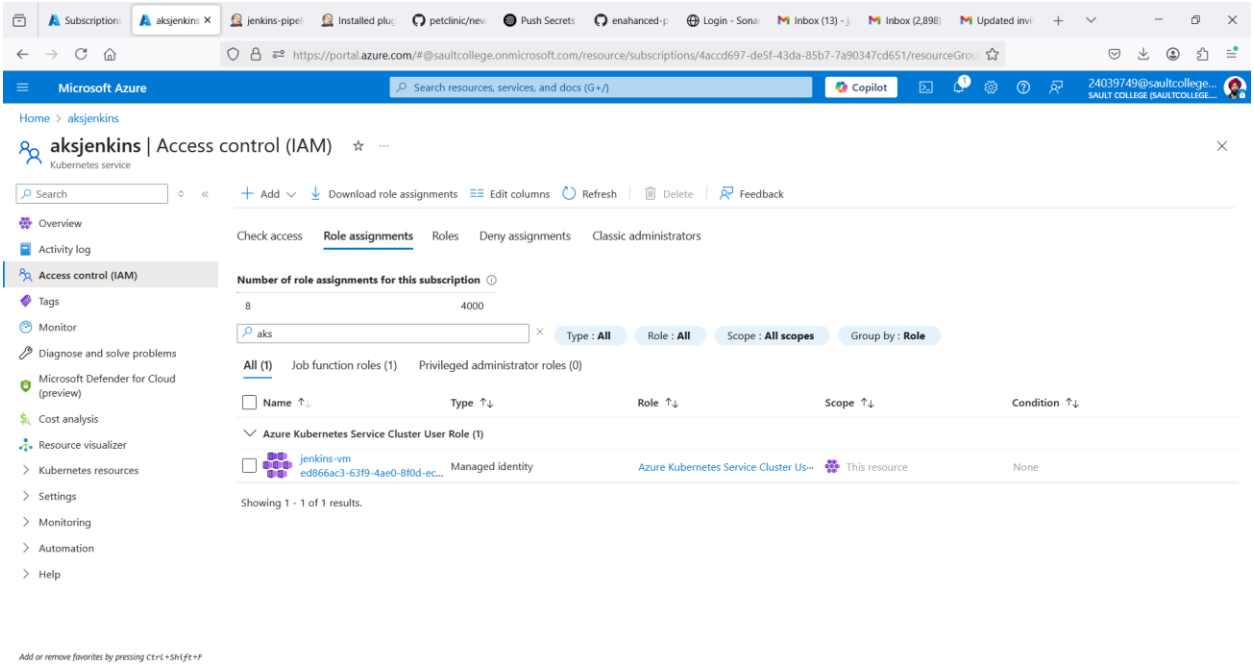
Kubernetes Deployment

Kubernetes login

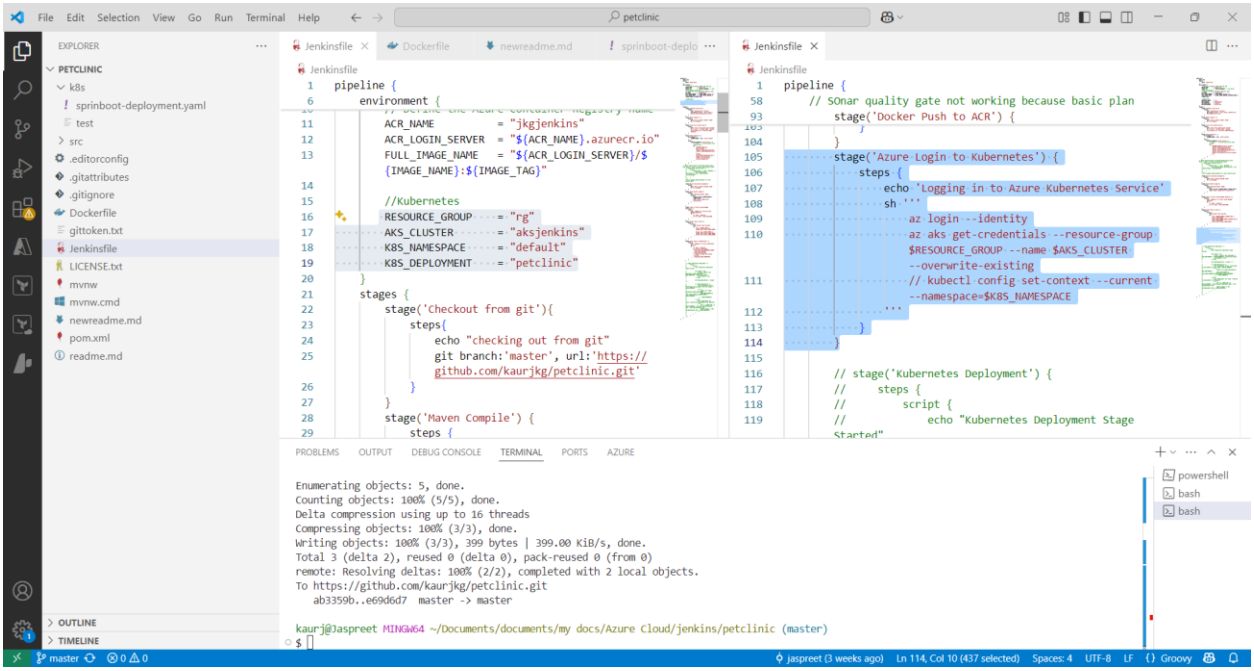
Create aks cluster.



To login to AKS cluster, we need to give the **Managed Identity** the correct role **Azure Kubernetes Service Cluster User Role** on the **AKS resource**, so it can fetch credential.



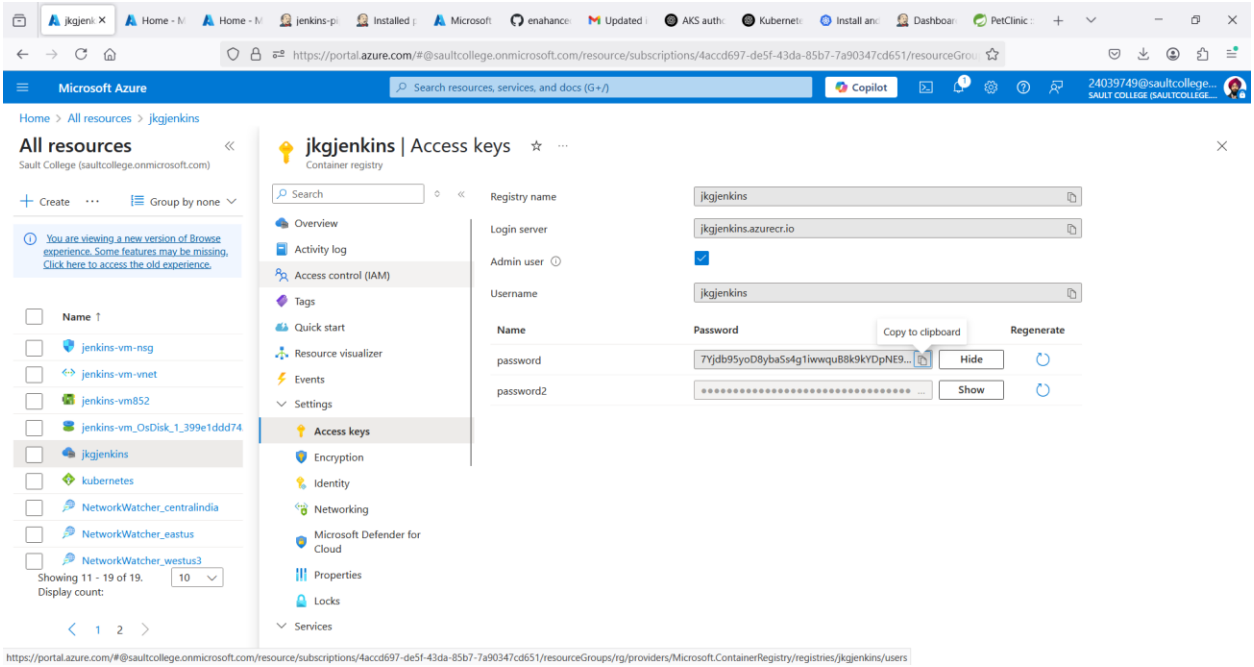
Now add stage for logging to azure for getting aks credentials.



Deploy image from ACR to AKS

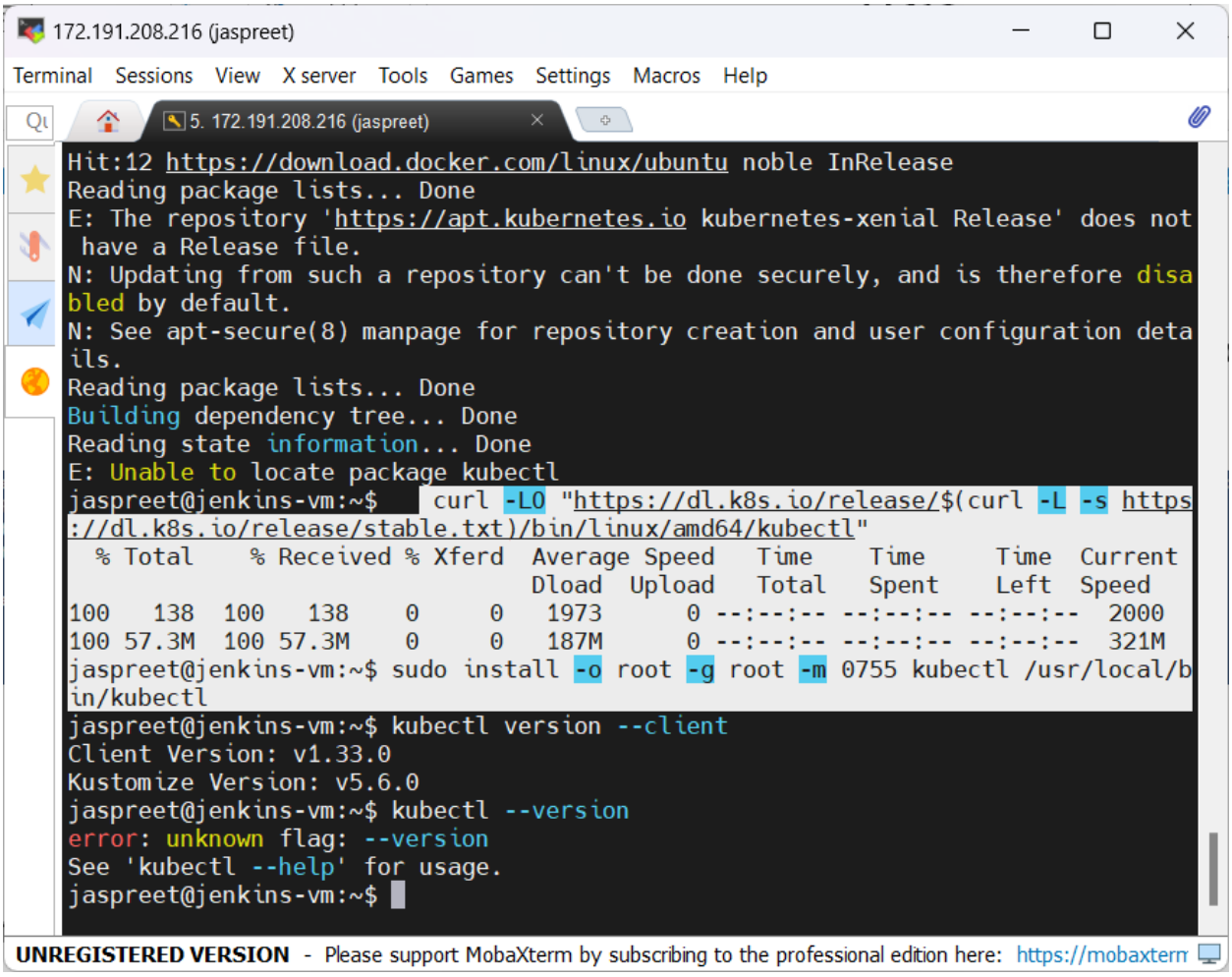
Create a secret to allow your Kubernetes cluster to pull Docker images from your private Azure Container Registry (ACR).

Get the credentials from acr registry

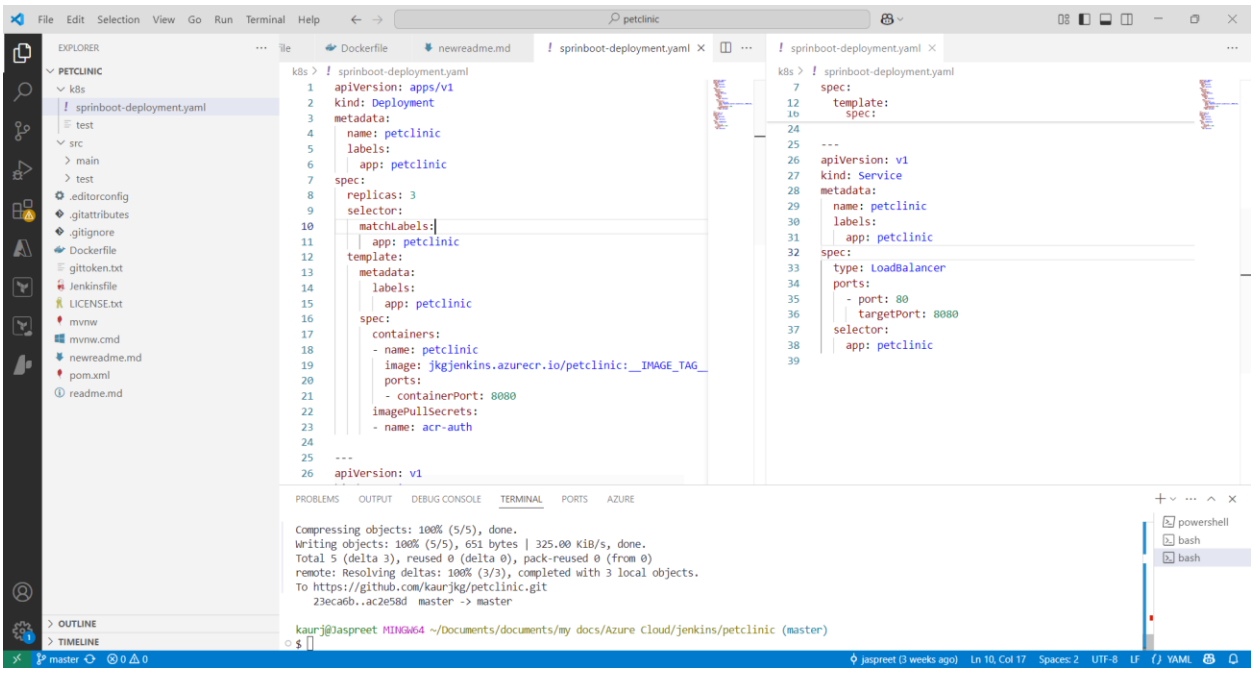



```
PS /home/gurpreet> kubectl create secret docker-registry acr-auth  --docker-server=jkgjenkins.azurecr.io   --docker-username=jkgjenkins   --docker-password=7Yjdb95yoD8ybaSs
4g1iwwquB8k9kYDpNE92/TILtS+ACRArRQ5s   --docker-email=jassu2498@gmail.com
error: failed to create secret secrets "acr-auth" already exists
PS /home/gurpreet> kubectl get secrets
NAME        TYPE                             DATA   AGE
acr-auth    kubernetes.io/dockerconfigjson   1      29m
PS /home/gurpreet>
```

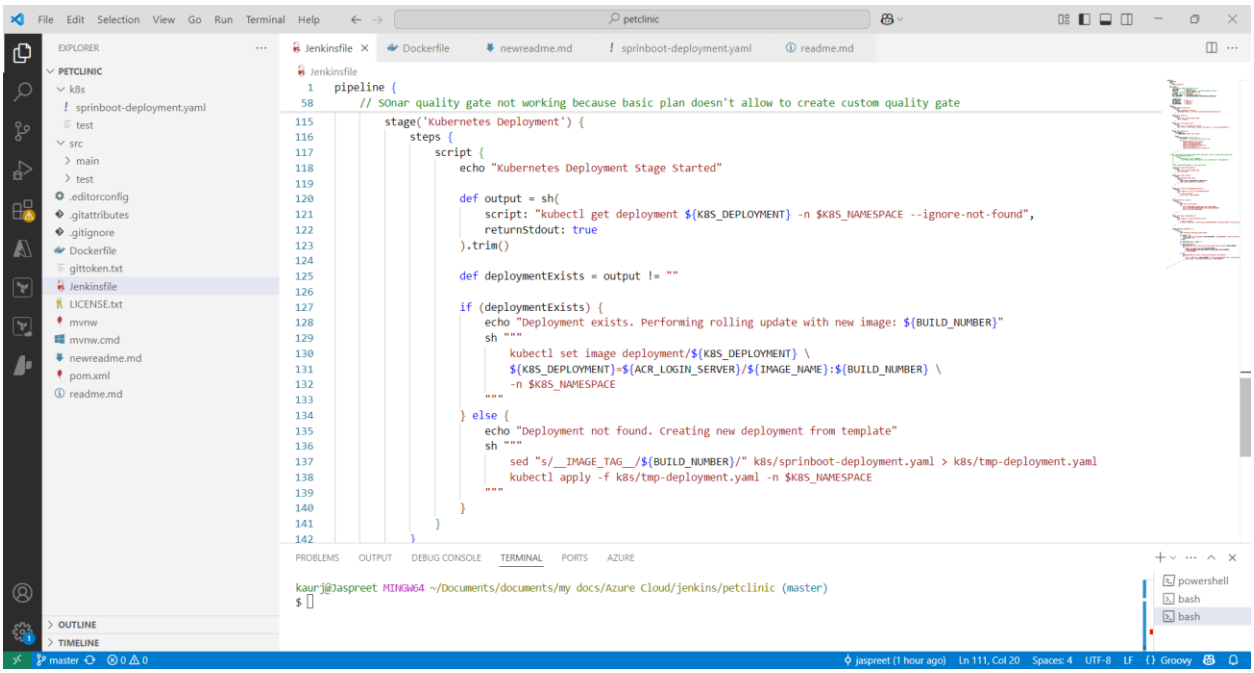
Install kubectl in jenkins vm agent.



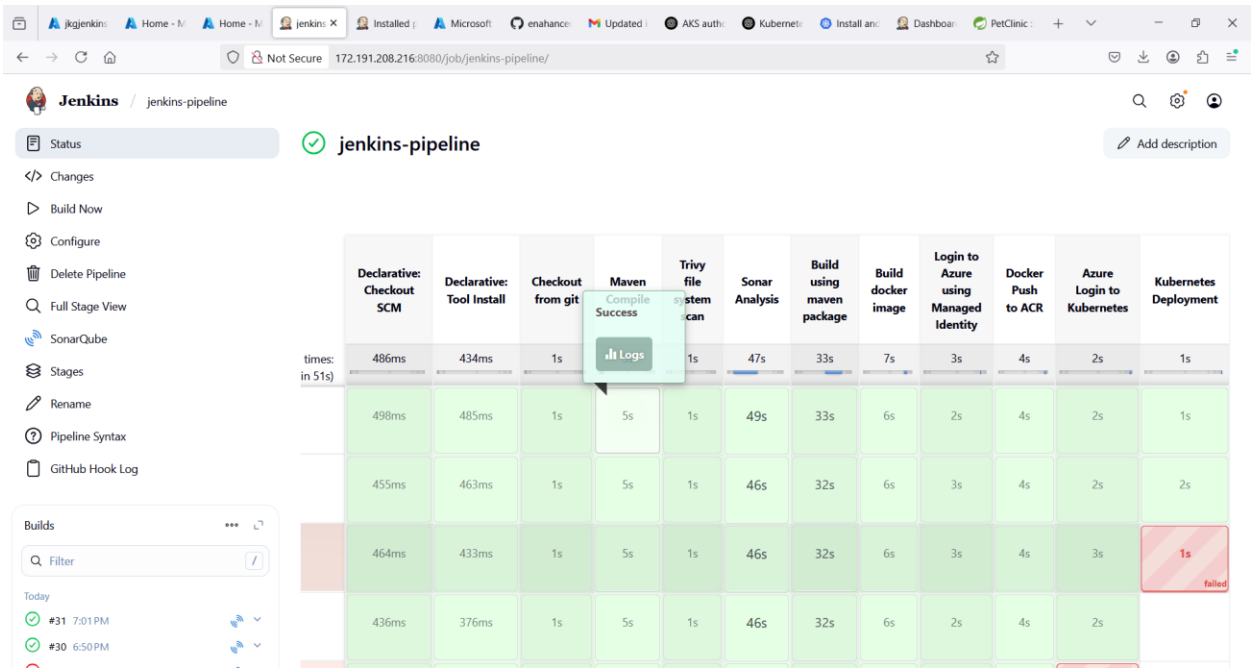
Write the deployment and service yaml.



Write pipeline stage for deployment



After pushing the code, the Jenkins pipeline ran.



Check the pods and service created.

```
PS /home/gurpreet> kubectl get po
NAME                         READY   STATUS    RESTARTS   AGE
petclinic-695c4ff7dc-64npn   1/1     Running   0          57s
petclinic-695c4ff7dc-8m281   1/1     Running   0          52s
petclinic-695c4ff7dc-f5btp   1/1     Running   0          55s
PS /home/gurpreet> kubectl get svc
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)        AGE
kubernetes   ClusterIP      10.0.0.1       <none>          443/TCP        4d17h
petclinic    LoadBalancer   10.0.102.95    134.33.249.25   80:30808/TCP   12m
```

The application is running on External IP provided by the load balancer service.