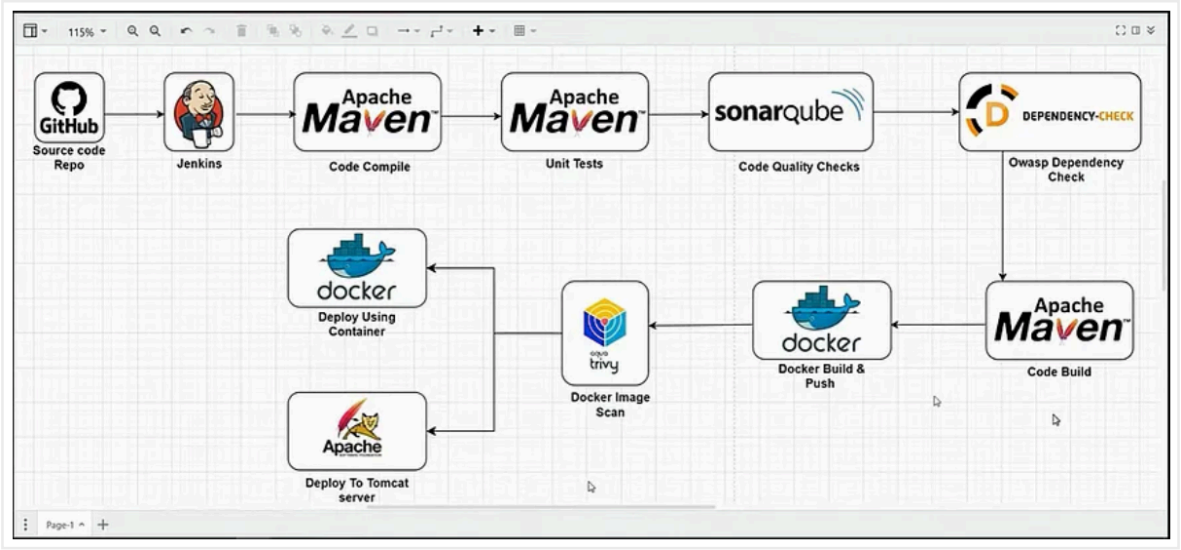
Pet Clinic - DevSecOps CI/CD for Java Based App on Tomcat Server

https://medium.com/cloud-native-daily/10-stages-real-world-ci-cd-devsecops-pipeline-for-deployment-of-petclinic-application-f95431bf940



Step 0 — Create Java Spring Project

- Use Spring Community sample pet clinic project and clone it `git clone https://github.com/spring-projects/spring-petclinic.git`
    - **Actually, the raw spring-petclinic project is unable to pass the mvn test. Please use the PetClinic.zip file provided by the instructor that has made the necessary modifications.**

- **Create a New Repository on GitHub**
    1. Go to GitHub and log in.
    2. Click the "+" icon in the top-right corner and select "New repository."
    3. Name your repository (e.g., PetClinic), add a description if you want, and choose the visibility (public or private).
    4. Do not initialize with a README, .gitignore, or license (since you already have those from the cloned repo). Click "Create repository."

- **Set Up Authentication for GitHub (Choose 1 of the**

methods below)

- **If you're using SSH:**
- Create SSH Key

```
# Generate new SSH Key
ssh-keygen -t rsa -b 4096 -C
"your_email@example.com" -f ~/.ssh/github_id_rsa

# Add new SSH Key to ssh-agent
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/github_id_rsa

# Copy the SSH Key to clipboard
cat ~/.ssh/github_id_rsa.pub
```

- Go to GitHub, navigate to "Settings" > "SSH and GPG keys," and click "New SSH key." Choose Authenticating Key. Paste the key and give it a title.

- Configure SSH Config File to use github_id_rsa for GitHub

```
vim ~/.ssh/config
Add:
Host github.com
  HostName github.com
  User git
  IdentityFile ~/.ssh/github_id_rsa
```

- **If you're using HTTPS and a personal access token:**
  - Go to GitHub, navigate to "Settings" > "Developer settings" > "Personal access tokens."
  - Click "Generate new token," provide a note, select scopes (repo, workflow, etc.), and generate the token. Copy it.

- **Push to Your GitHub Repository**
  - **If you're using SSH:**

```
# Set the new remote URL using SSH

echo "# PetClinic" >> README.md
git init
git add .
git commit -m "Initial commit"
git branch -M main
```

```
git remote add origin https://github.com/your-
username/PetClinic.git
git remote set-url origin git@github.com:your-
username/PetClinic.git
git push -u origin main
```

- **If you're using HTTPS and a personal access token:**

```
# Set the new remote URL using HTTPS
echo "# PetClinic" >> README.md
git init
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin https://github.com/your-
username/PetClinic.git
git remote set-url origin https://github.com/your-
username/PetClinic.git
git push -u origin main

# you'll be prompted to enter your GitHub username
and the personal access token you generated earlier.
```

Step 1 — Use an Ubuntu VM **Instance**

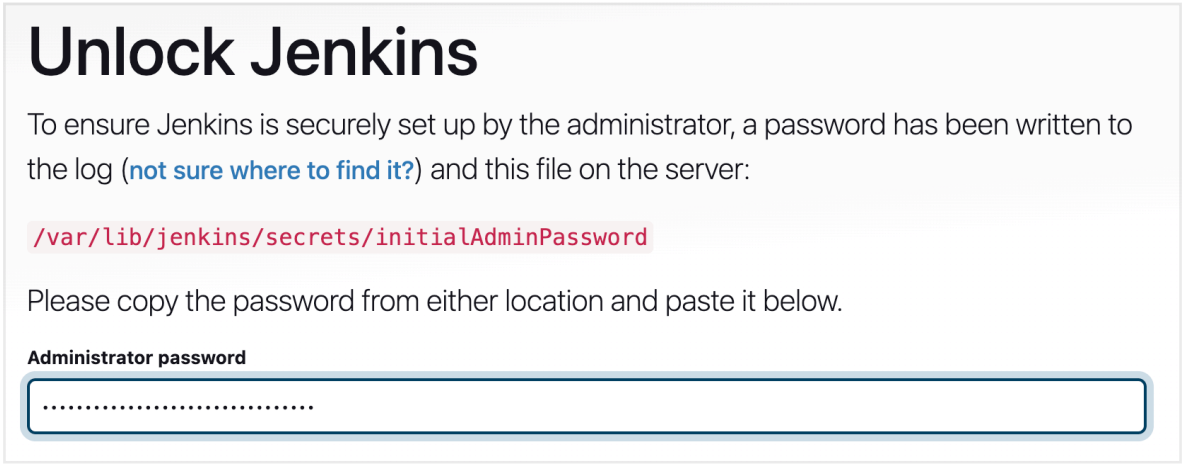
Step 2 — Install Jenkins, Docker and Trivy. Create a Sonarqube Container using Docker.

- Login to the VM

- Install Jenkins

```
sudo apt-get update

curl -fsSL https://pkg.jenkins.io/debian-stable/
jenkins.io-2023.key | sudo tee \
    /usr/share/keyrings/jenkins-keyring.asc > /dev/
null

echo deb [signed-by=/usr/share/keyrings/jenkins-
keyring.asc] \
    https://pkg.jenkins.io/debian-stable binary/ |
sudo tee \
    /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update
sudo apt install -y openjdk-17-jdk

sudo apt install -y openjdk-17-jre

sudo apt install -y jenkins

sudo systemctl enable jenkins
sudo systemctl start jenkins


sudo systemctl status jenkins
sudo cat  /var/lib/jenkins/secrets/
initialAdminPassword
```

```
ubuntu@ip-10-0-14-196:~$ sudo cat  /var/lib/jenkins/secrets/initialAdminPassword
ab85357abcc04c5eaffb1a0e7f7499c7
```



    - Install suggested plugins
    - Create First Admin User

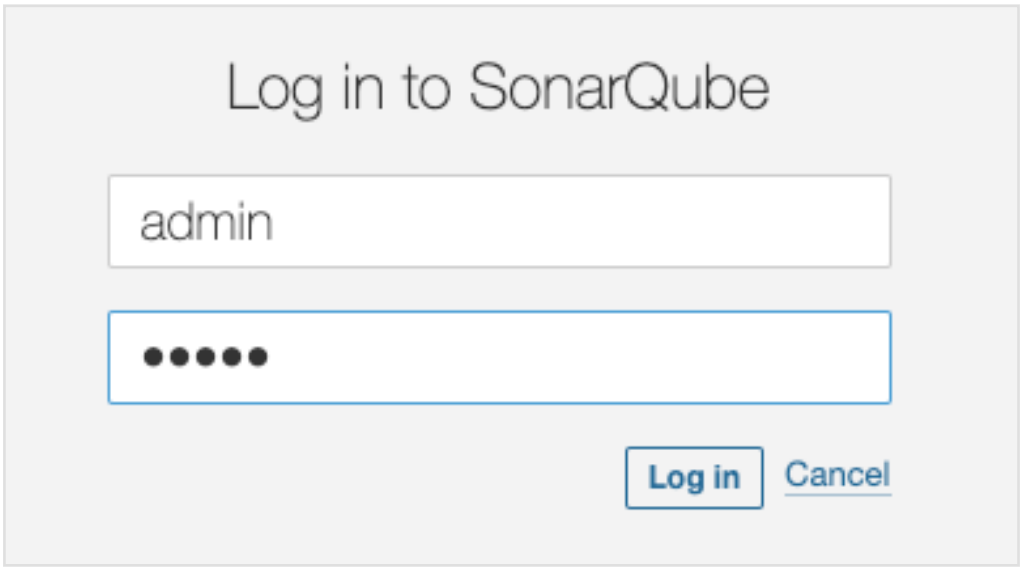
- Install Docker

```
sudo apt-get update
sudo apt-get install docker.io -y
sudo usermod -aG docker $USER
sudo chmod 777 /var/run/docker.sock
sudo docker ps
```

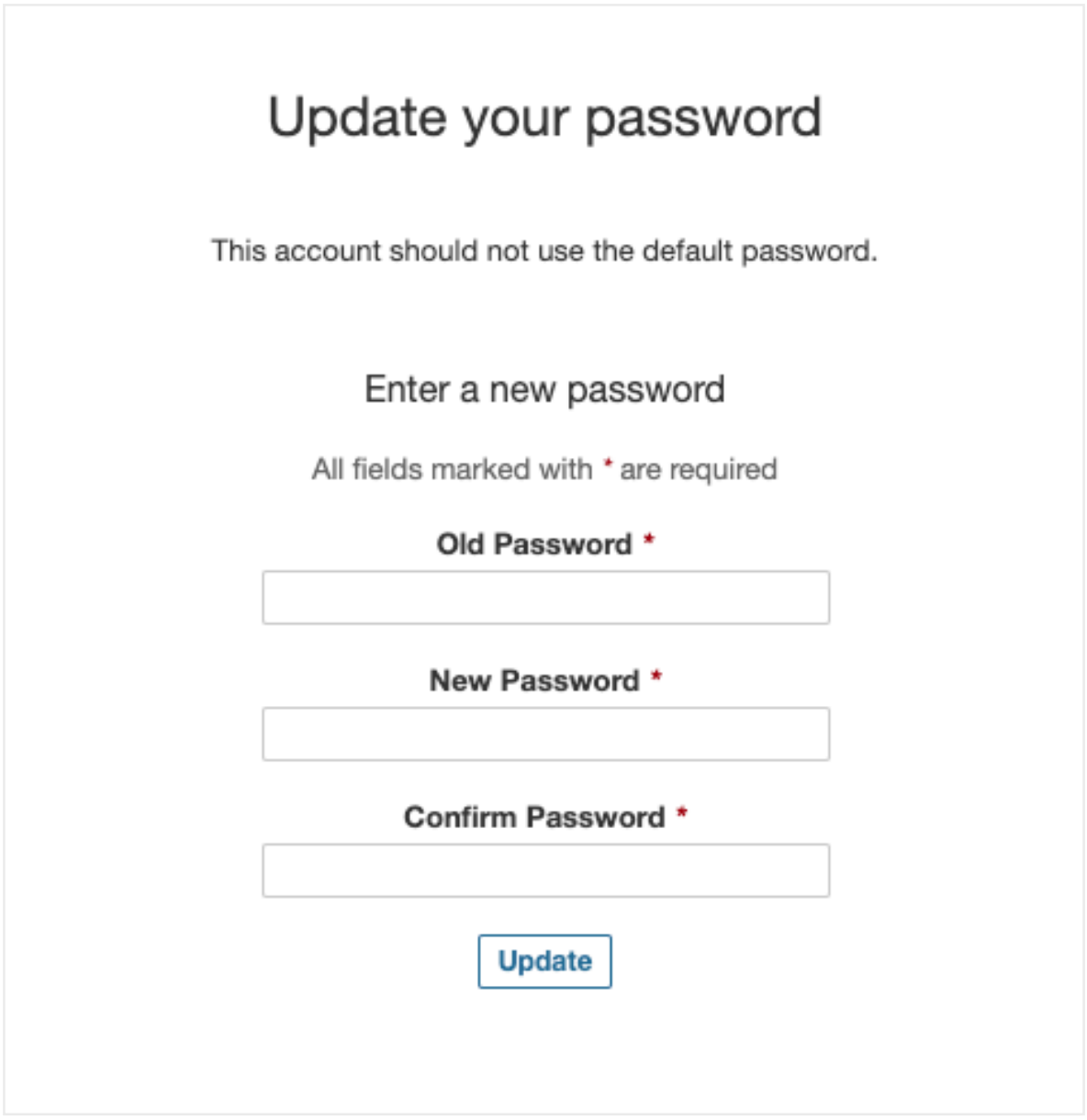
- Add Sonarqube container
    - Ensure Custom TCP port 9000 has been added to Security Group Inbound Rule for Source = My IP

```
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

    - Navigate to publicIP:9000
        - user = admin, password = admin



- Update password



- Install Trivy

```
sudo apt-get install wget apt-transport-https gnupg
```

```
lsb-release -y

wget -qO - https://aquasecurity.github.io/trivy-
repo/deb/public.key | \
  gpg --dearmor | sudo tee /usr/share/keyrings/
trivy.gpg > /dev/null

echo "deb [signed-by=/usr/share/keyrings/trivy.gpg]
\
  https://aquasecurity.github.io/trivy-repo/deb $
(lsb_release -sc) main" | \
  sudo tee -a /etc/apt/sources.list.d/trivy.list

sudo apt-get update

sudo apt-get install trivy -y
```

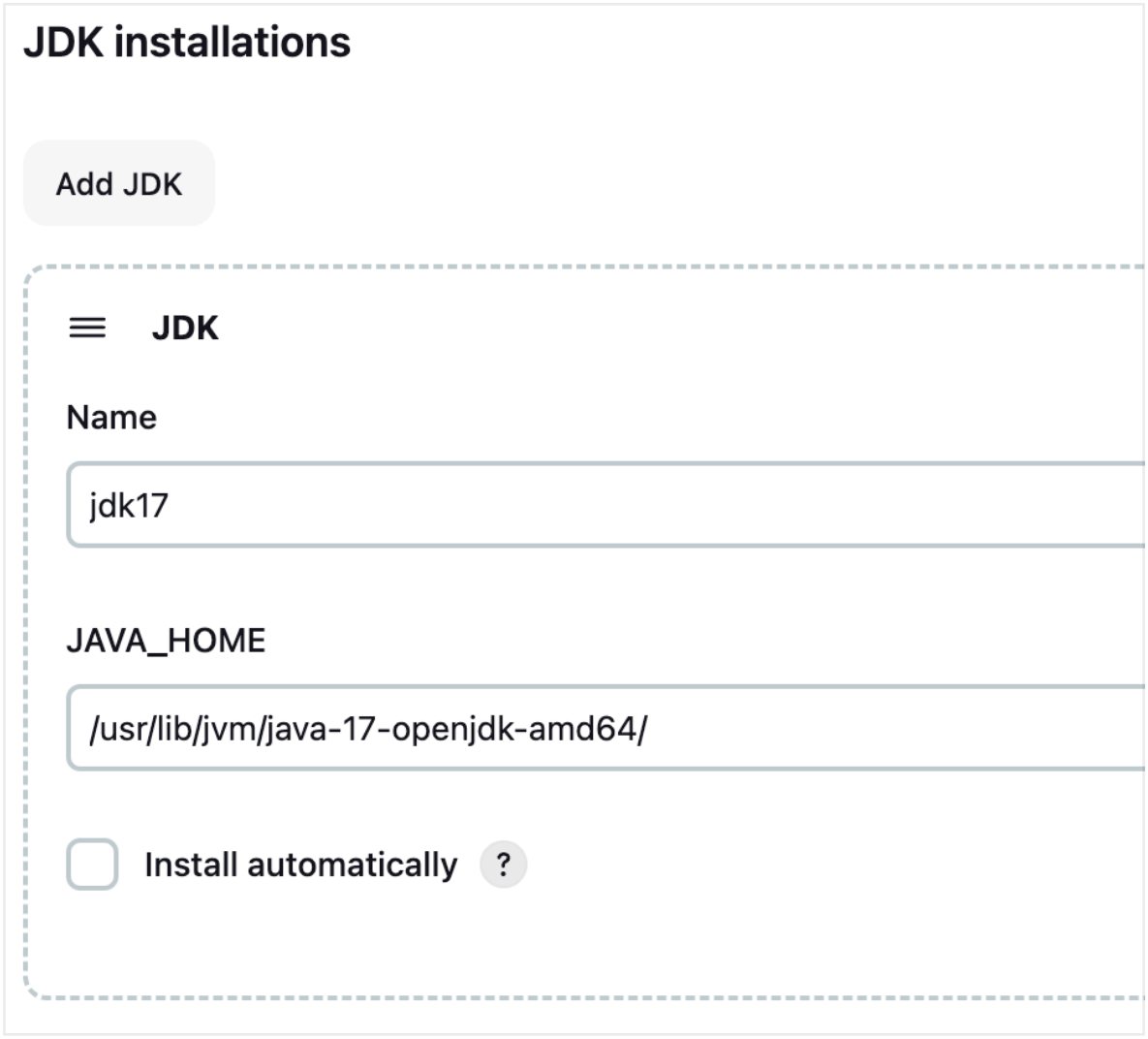
Step 3 — Install Plugins like JDK, Sonarqube Scanner, Maven, OWASP Dependency Check,

- From Jenkins => Goto Manage Jenkins →Plugins → Available Plugins
- Install following plugins
  - Eclipse Temurin Installer (Install without restart)
  - SonarQube Scanner (Install without restart)

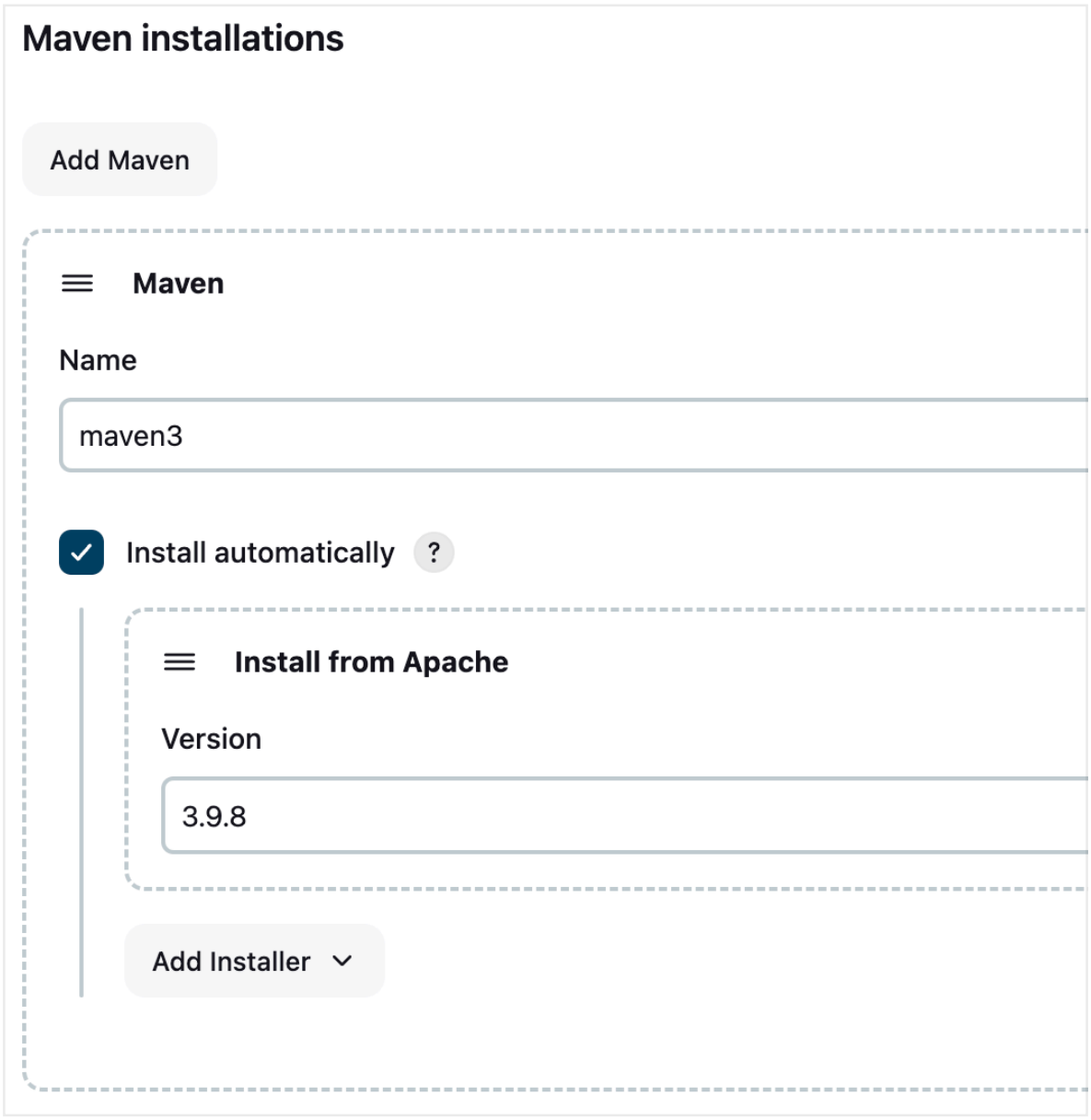
- Configure Java and Maven in Global Tool Configuration
  - Goto Manage Jenkins → Tools → Install JDK and Maven3 → Click on Apply and Save
  - Add Java
    - Name = jdk17
    - For Java_Home, run the following to get path

```
readlink -f /usr/bin/java | sed "s:bin/java::"
```

```
ubuntu@ip-10-0-14-196:~$ readlink -f /usr/bin/java | sed "s:bin/java::"
/usr/lib/jvm/java-17-openjdk-amd64/
```

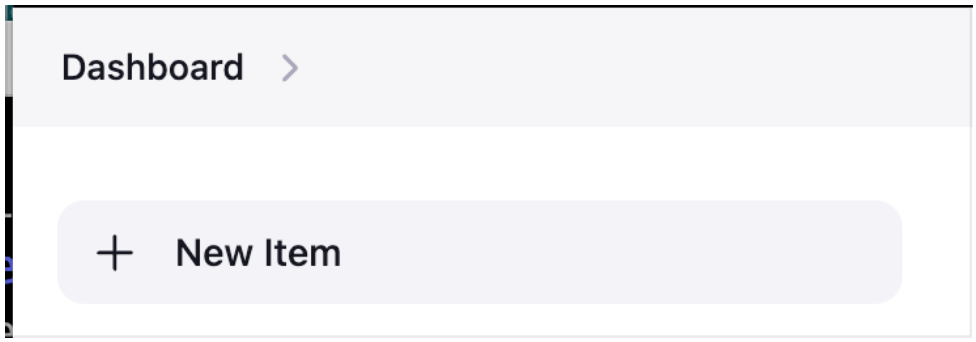


- Add Maven installation (Not Maven configuration)
- Name = maven3



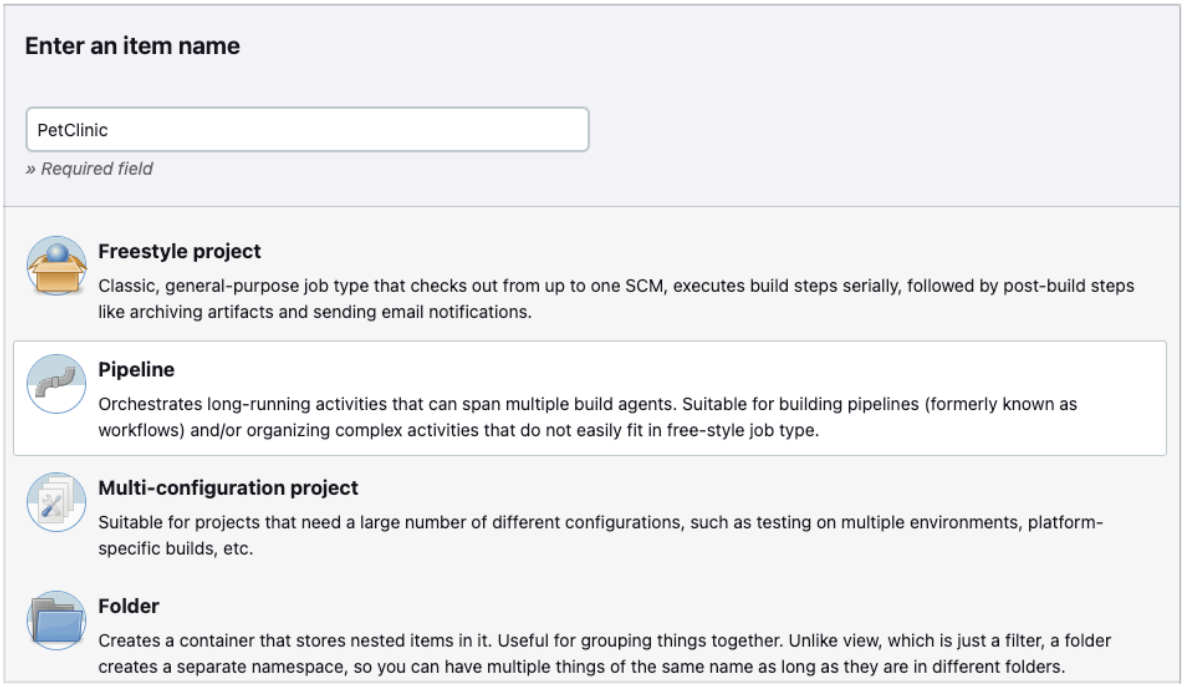
    - Apply and Save

- Test using Jenkins Declarative Pipeline
    - Dashboard → New item



    - Give item name
    - Select Pipeline
    - make sure the jdk and maven matches the name you

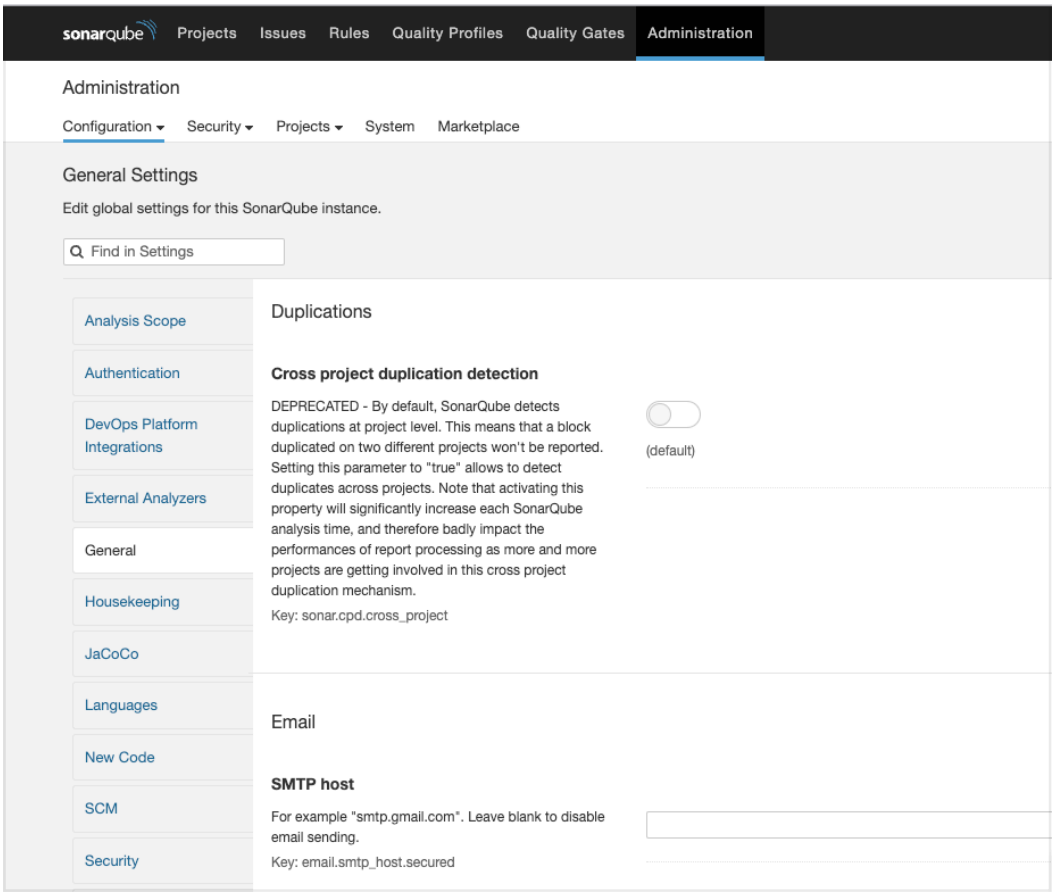
declared earlier above
- Reference **Step 4 — Create a Pipeline Project in Jenkins using Declarative Pipeline** for detailed instructions on creating and running a Pipeline

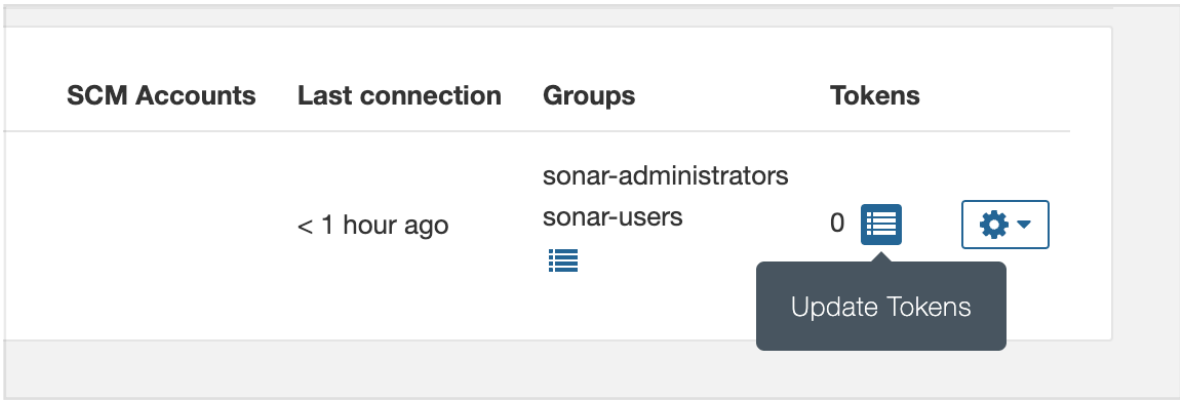



```
pipeline {
    agent any

    tools{
        jdk 'jdk17'
        maven 'maven3'
    }

     stages{

        stage("Git Checkout"){
            steps{
                git branch: 'main', changelog:
false, poll: false, url: 'https://github.com/
wikenhub/PetClinic.git'
            }
        }

        stage("Compile"){
            steps{
                sh "mvn clean compile"
            }
        }
```

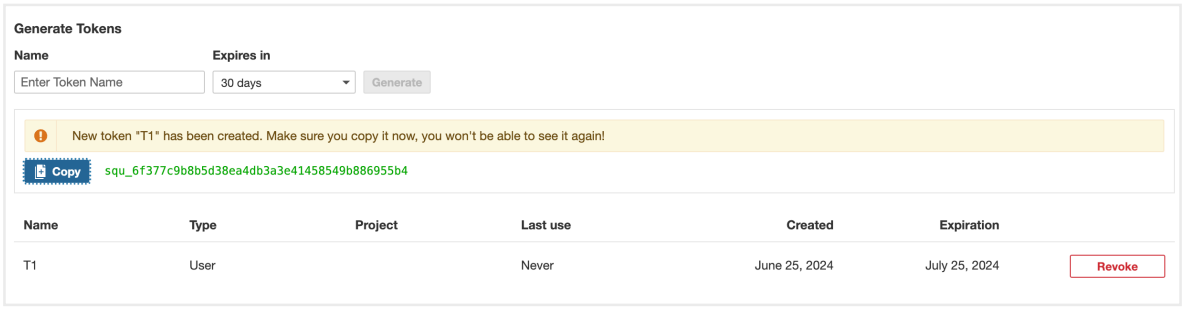
```
            stage("Test Cases"){
                steps{
                    sh "mvn test"
                }
            }
        }
}
```

- **Configure Sonar Server in Manage Jenkins**
  - Goto URL = publicIP:9000
  - Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Toke

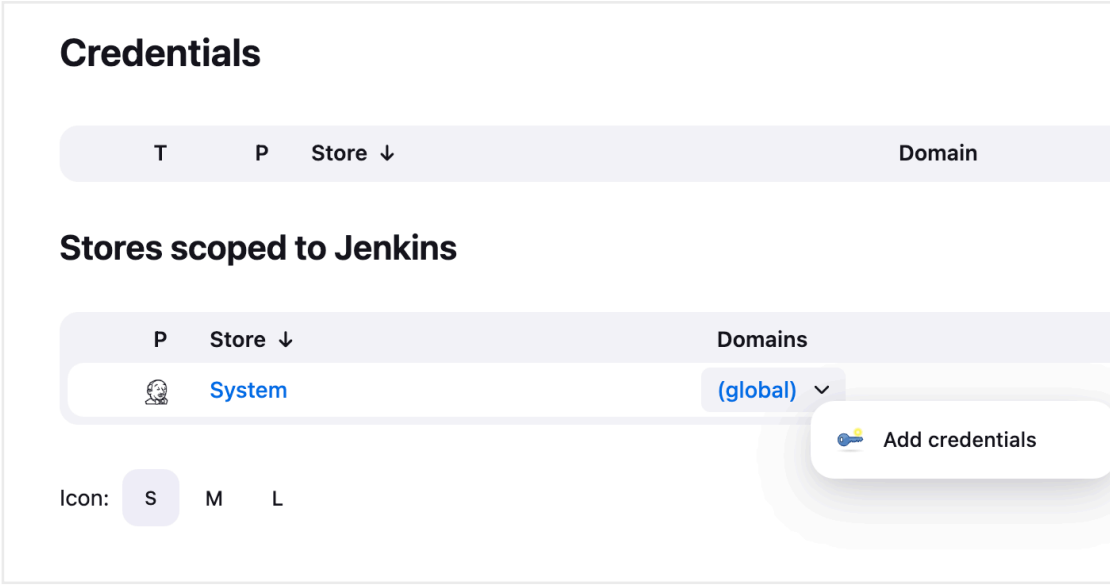




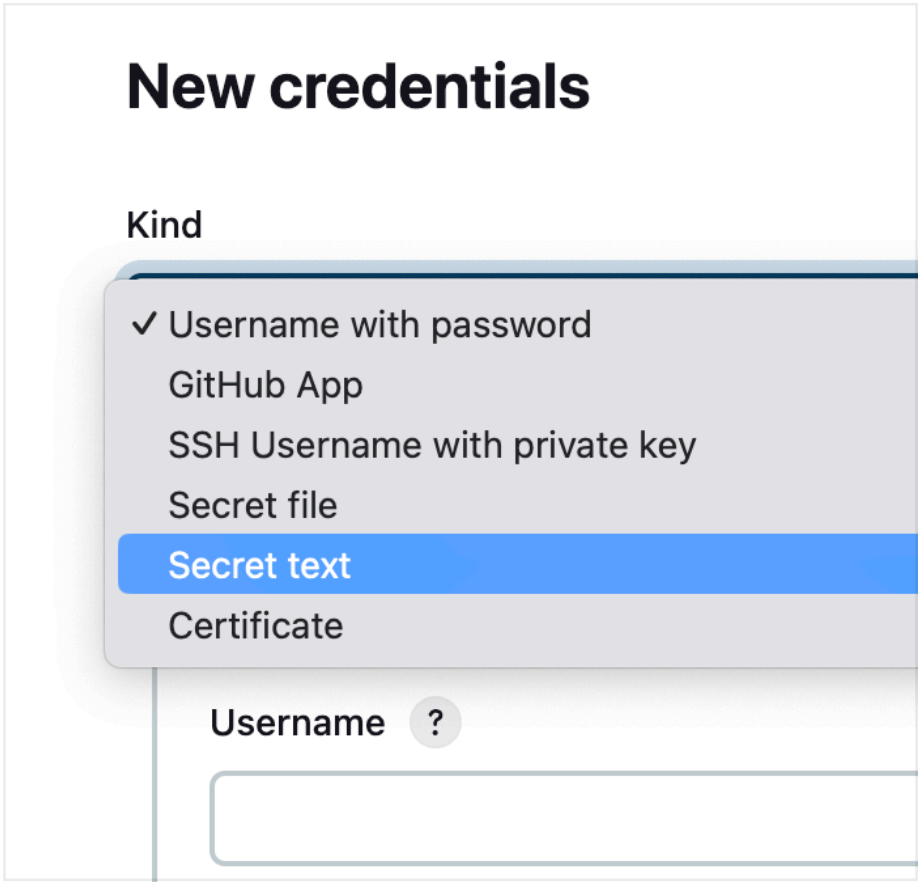
- Copy the token string for next step.

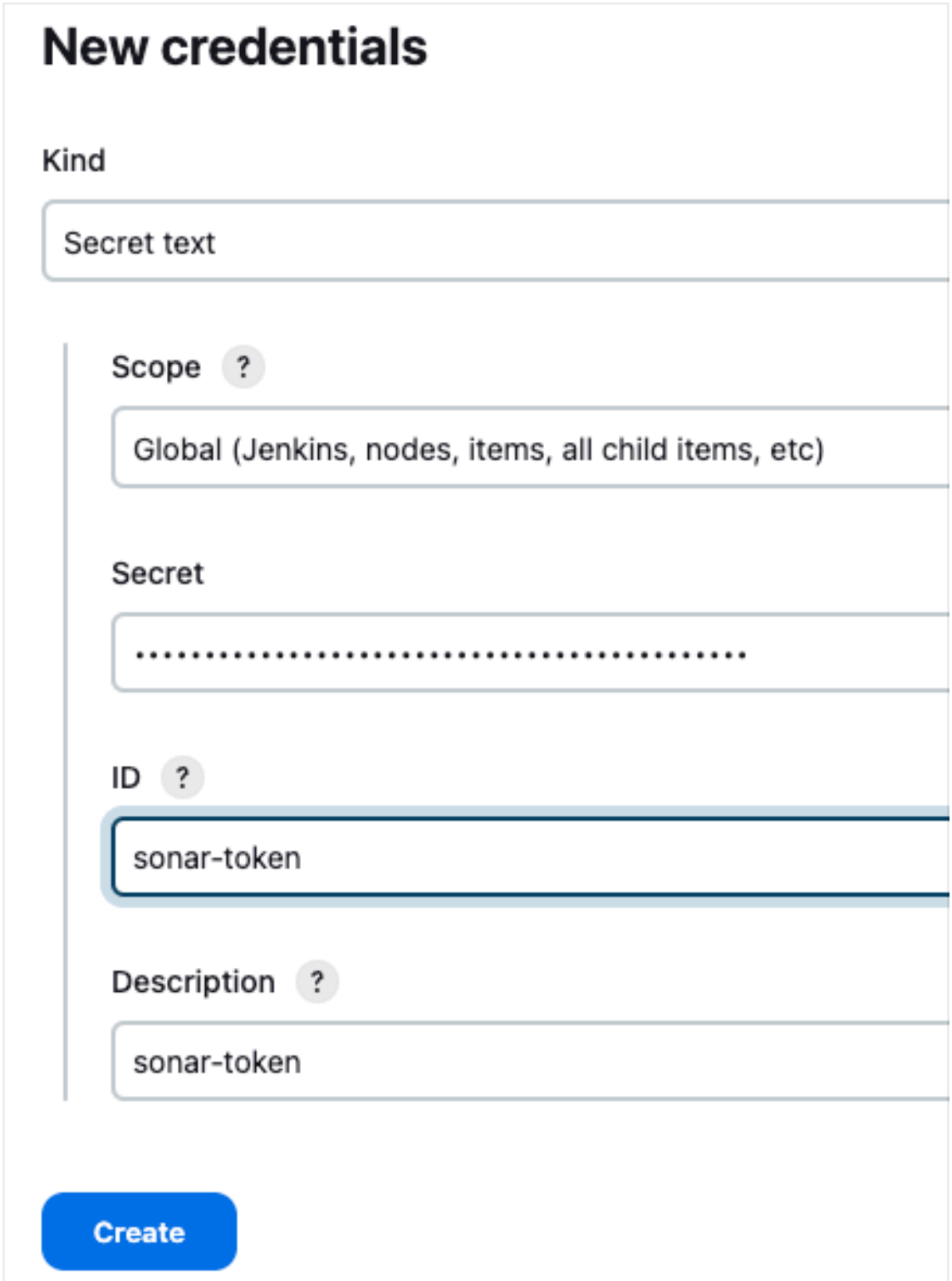


- Dashboard → Manage Jenkins → Credentials → Add Credentials → Select Add Secret Text→

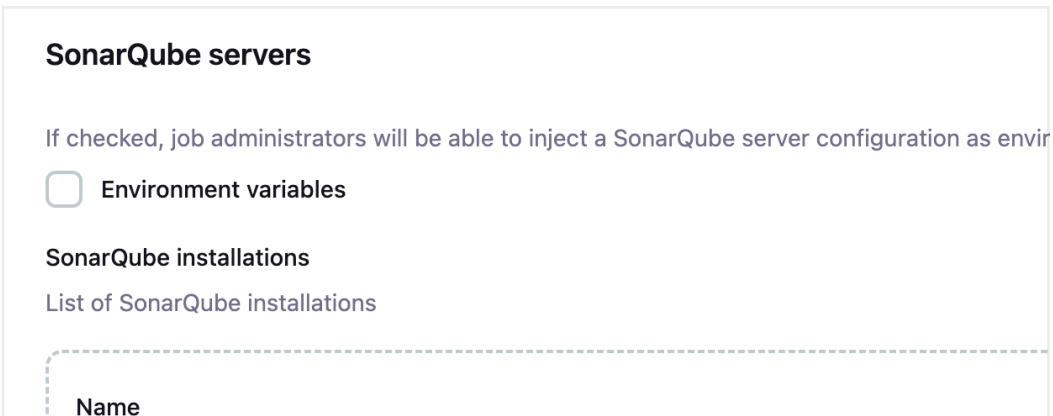


-


New credentials
Kind
✓ Username with password
GitHub App
SSH Username with private key
Secret file
Secret text
Certificate
Username
?



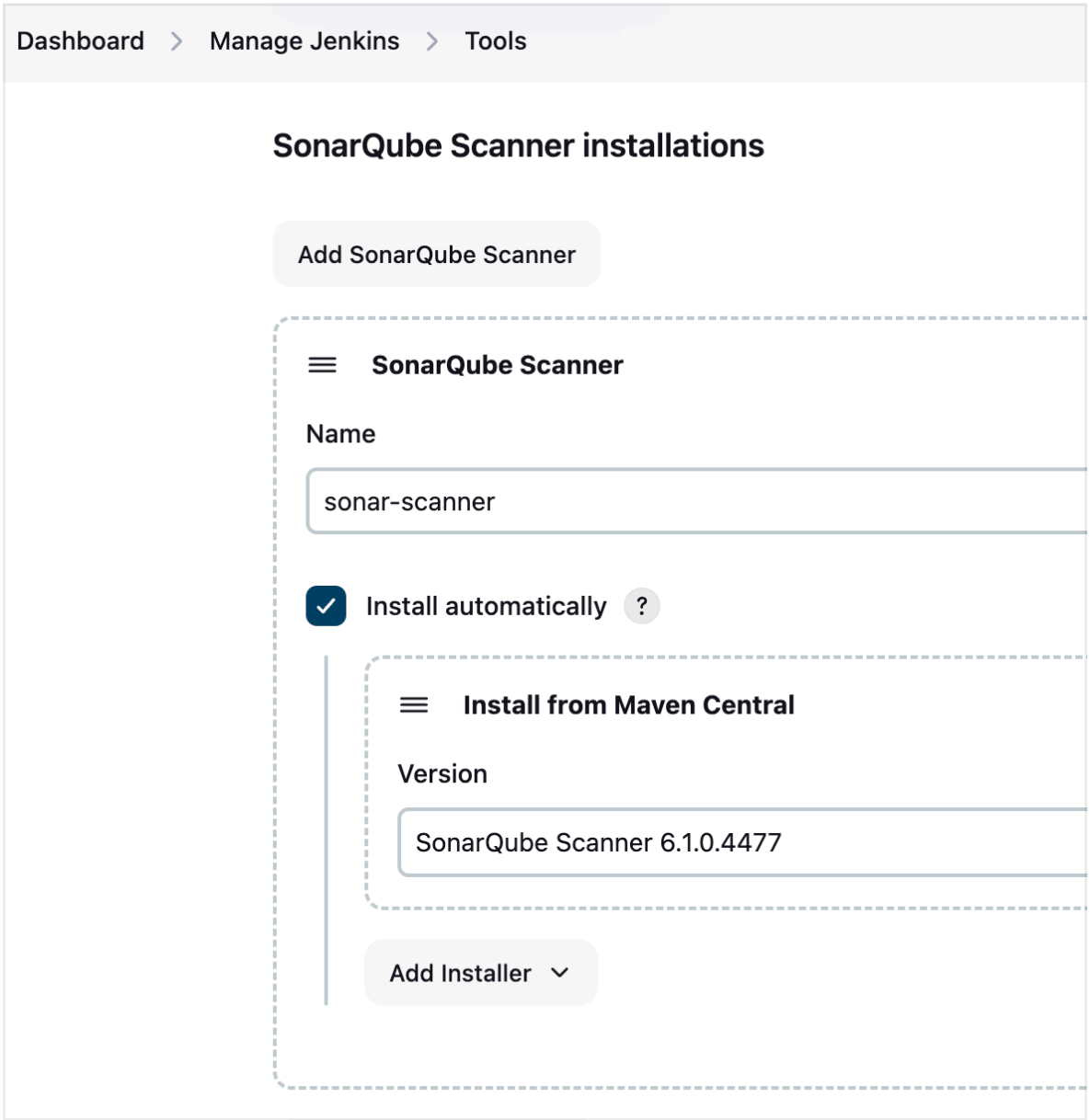
- Dashboard → Manage Jenkins →System -> SonarQube -> Add SonarQube installations



- Apply and Save

Configure System option is used in Jenkins to configure different servers
Global Tool Configuration is used to configure different tools that we install using Plugins

- Dashboard → Manage Jenkins → Tools

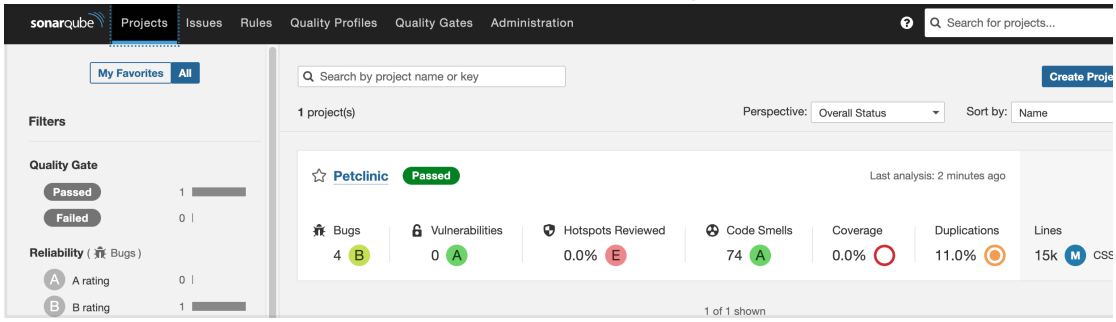


- Apply and Save
- Add Sonarqube Stage in our Pipeline Script - Make sure to change the Git URL to your PetClinic repository.

```
pipeline {
    agent any
```

```
    tools{
        jdk 'jdk17'
        maven 'maven3'
    }

    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }

     stages{

        stage("Git Checkout"){
            steps{
                git branch: 'main', changelog:
false, poll: false, url: 'https://github.com/
wikenhub/PetClinic.git'
            }
        }

        stage("Compile"){
            steps{
                sh "mvn clean compile"
            }
        }

         stage("Test Cases"){
            steps{
                sh "mvn test"
            }
        }

        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-
scanner -Dsonar.projectName=PetClinic \
                    -Dsonar.java.binaries=. \
                    -Dsonar.projectKey=PetClinic '''
                }
            }
        }
     }
```

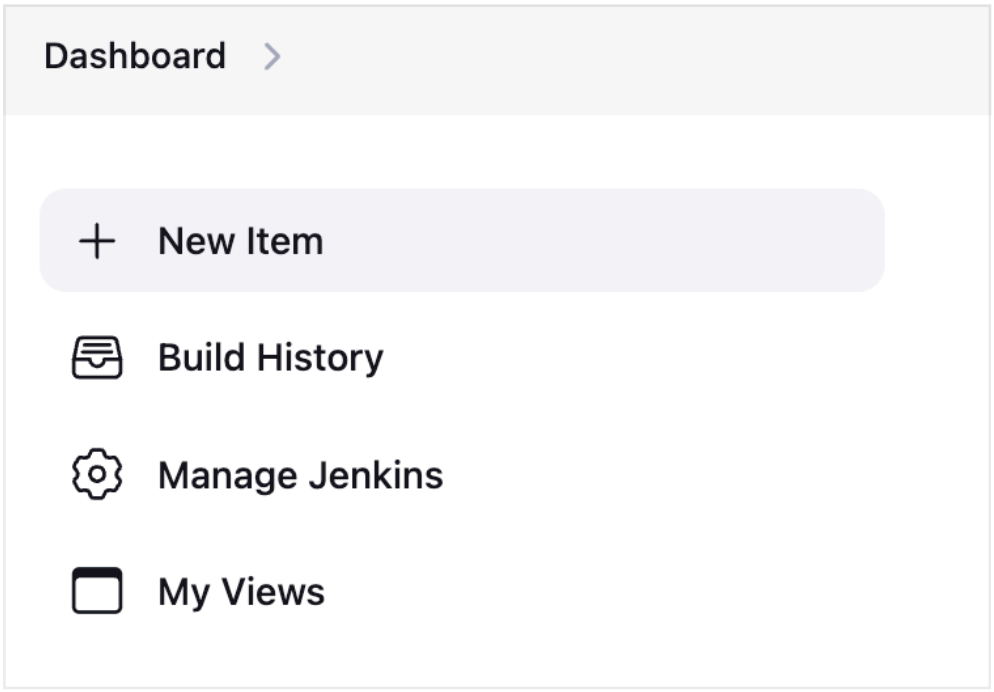
}

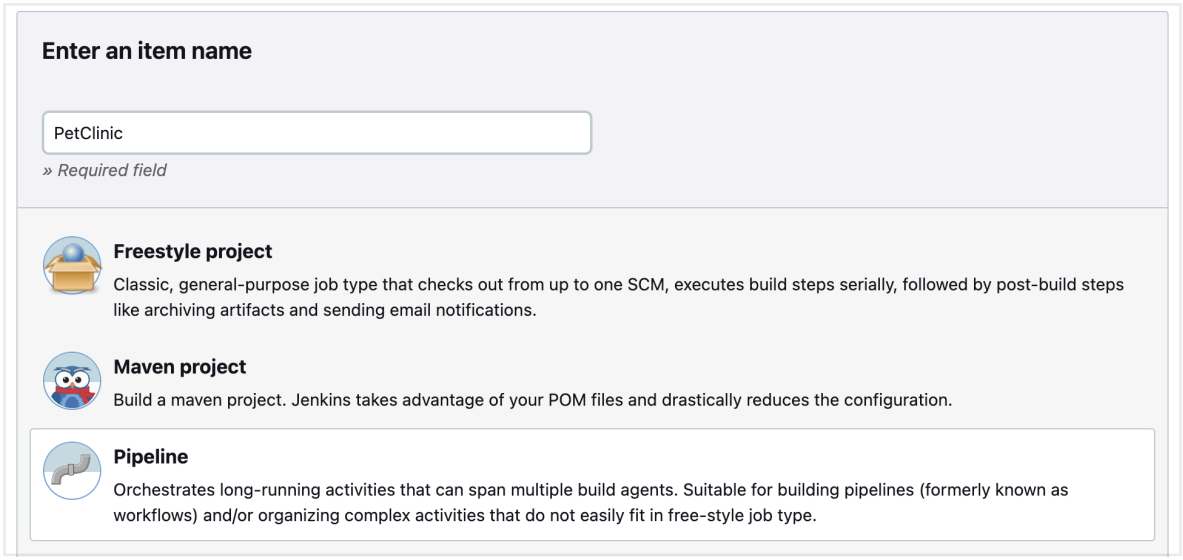
- Run the build and ensure success
- Check from SonarQube Web UI -> Project
- 

-

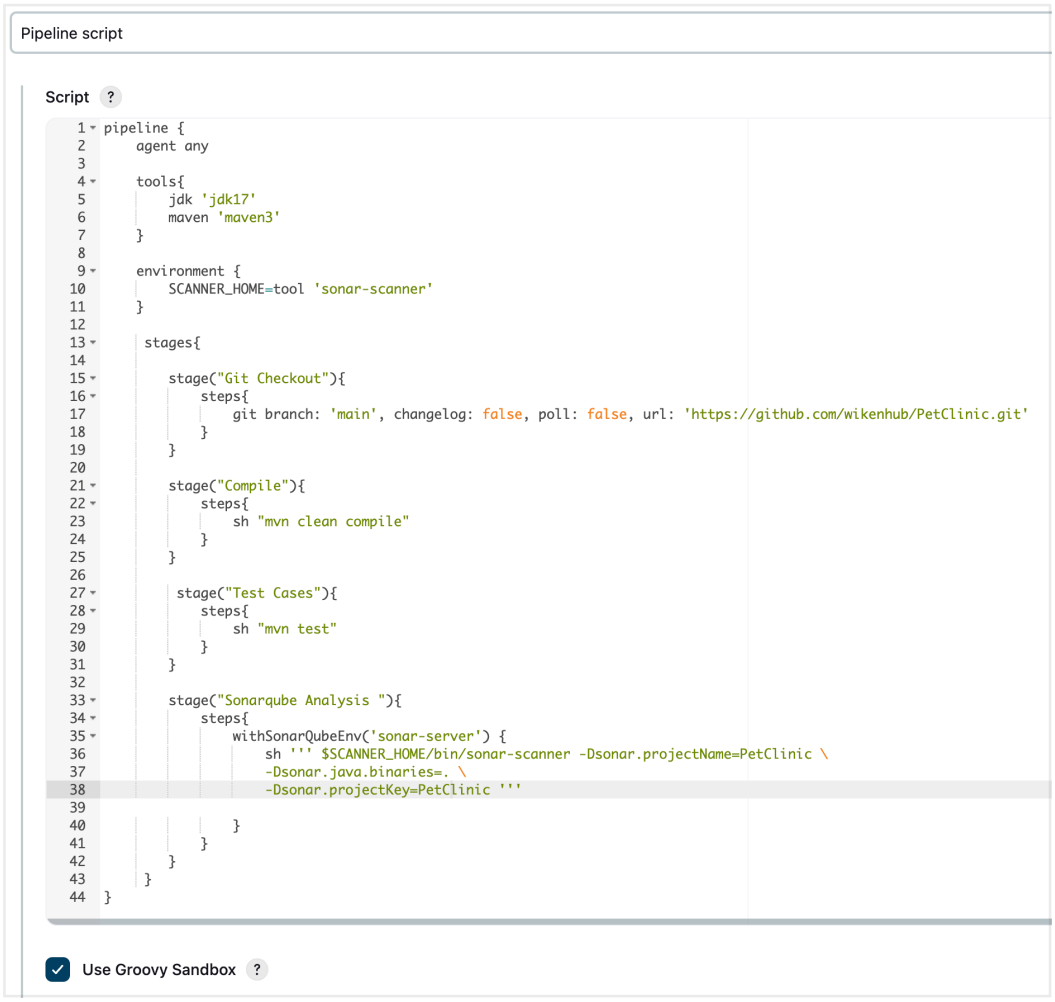
Step 4 — Create a Pipeline Project in Jenkins using Declarative Pipeline

We have been testing JDK, Maven and Sonarqube using a Groovy Pipeline.

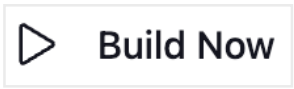




The pipeline should have the following Pipeline script so far:



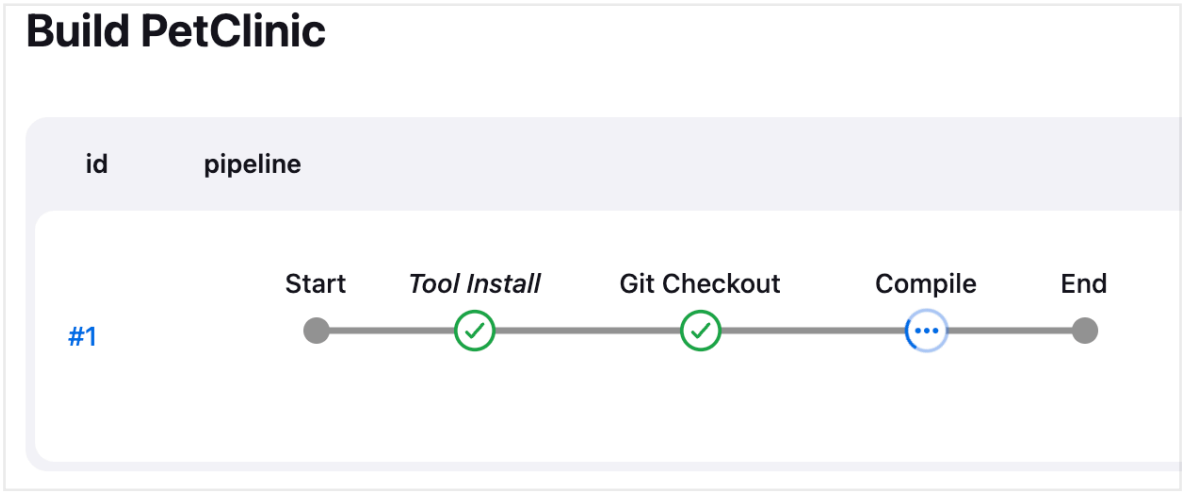
Make sure you have entered the url for your own PetClinic repository in the `Git Checkout` stage.
You can view the stages while building by selecting



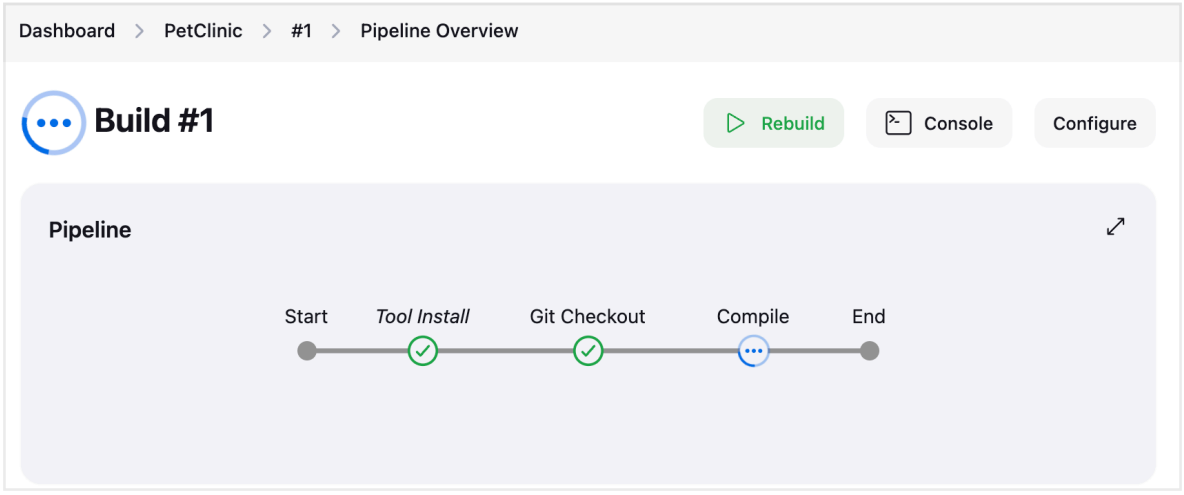
and then

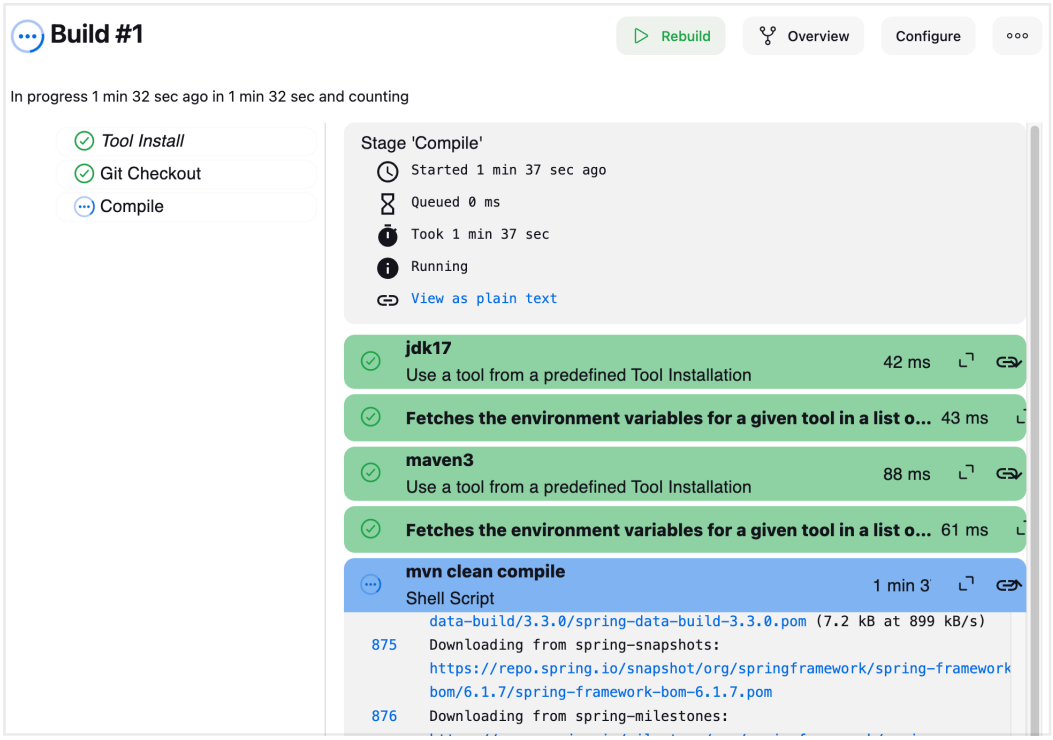
Stages

from the left tab



Click on the build number and then Console from the top menu to see the console output




Build #1
Rebuild
Overview
Configure
In progress 1 min 32 sec ago in 1 min 32 sec and counting
Tool Install
Git Checkout
Compile
Stage 'Compile'
Started 1 min 37 sec ago
Queued 0 ms
Took 1 min 37 sec
Running
View as plain text
jdk17
Use a tool from a predefined Tool Installation
42 ms
Fetches the environment variables for a given tool in a list o... 43 ms
maven3
Use a tool from a predefined Tool Installation
88 ms
Fetches the environment variables for a given tool in a list o... 61 ms
mvn clean compile
Shell Script
1 min 3
data-build/3.3.0/spring-data-build-3.3.0.pom (7.2 kB at 899 kB/s)
875 Downloading from spring-snapshots:
https://repo.spring.io/snapshot/org/springframework/spring-framework
bom/6.1.7/spring-framework-bom-6.1.7.pom
876 Downloading from spring-milestones:

Step 5 — using Trivy

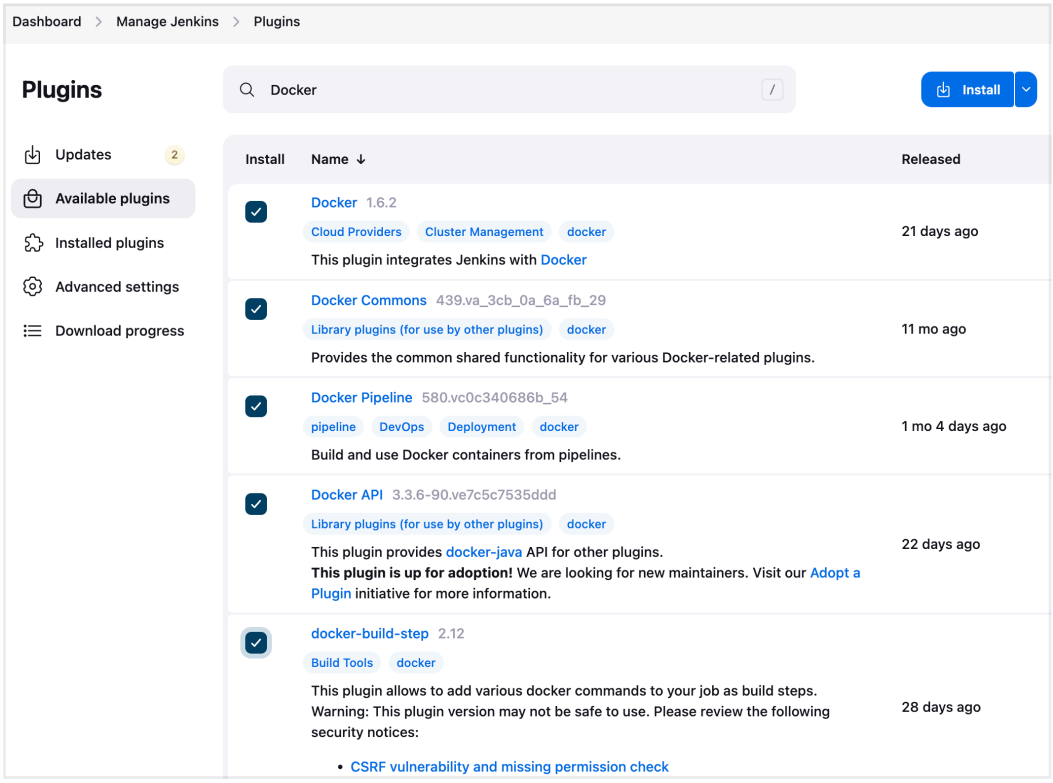
There are multiple alternatives to OWASP Dependency Checking for or scanning vulnerabilities in your Jenkins pipeline, including Snaky, Whitesource Bolt and Trivy. We already have installed Trivy on our EC2 machine so we will choose this alternative.

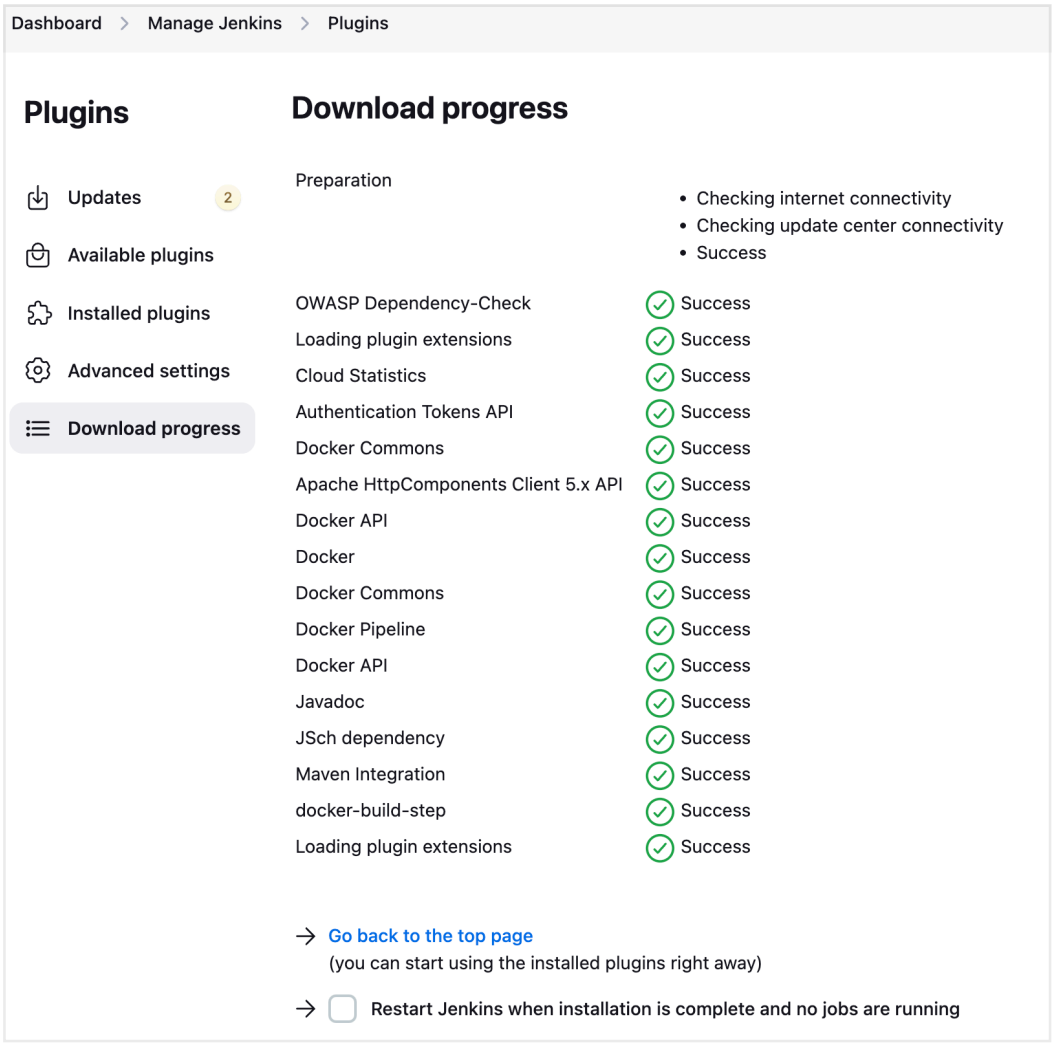
Add the following to our pipeline. It is possible to stop the pipeline if the Trivy status returns an error, however, we will bypass and continue to build. This is because there are know issues with our PetClinic Spring application. In our lab, we will ignore the issues.

```
        stage("Trivy Vulnerability Scan") {
            steps {
                script {
                    def trivyStatus = sh(script:
'trivy fs --severity HIGH,CRITICAL .', returnStatus:
true)
                    if (trivyStatus != 0) {
                        echo "Trivy found
vulnerabilities, but the build will continue."
                    }
                }
            }
        }

        stage("Build"){
            steps{
                sh " mvn clean install"
            }
        }
```

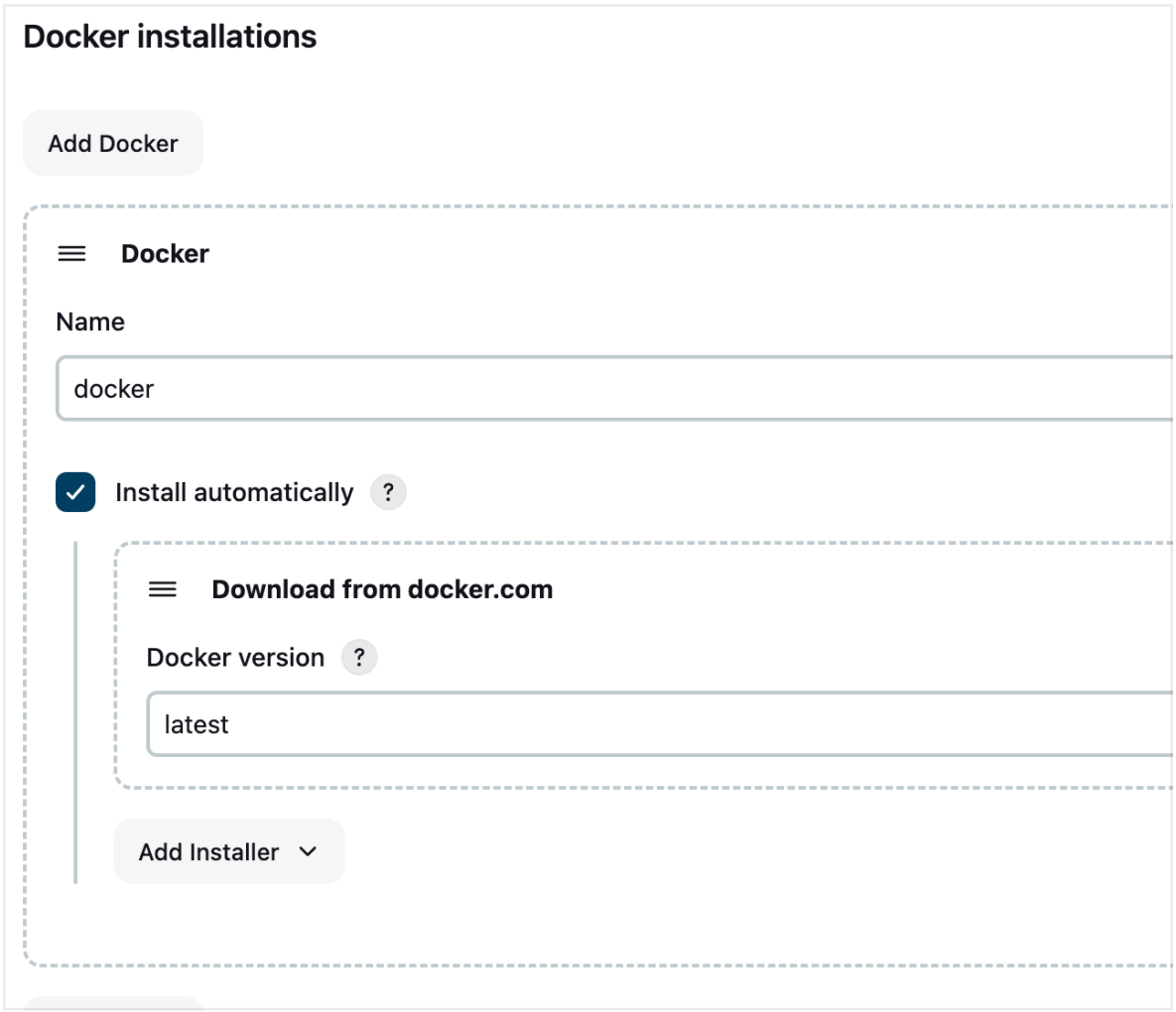
Step 6 — Docker Image Build and Push

- Install the following Docker related plugins
  - Docker
  - Docker Commons
  - Docker Pipeline
  - Docker API
  - docker-build-step



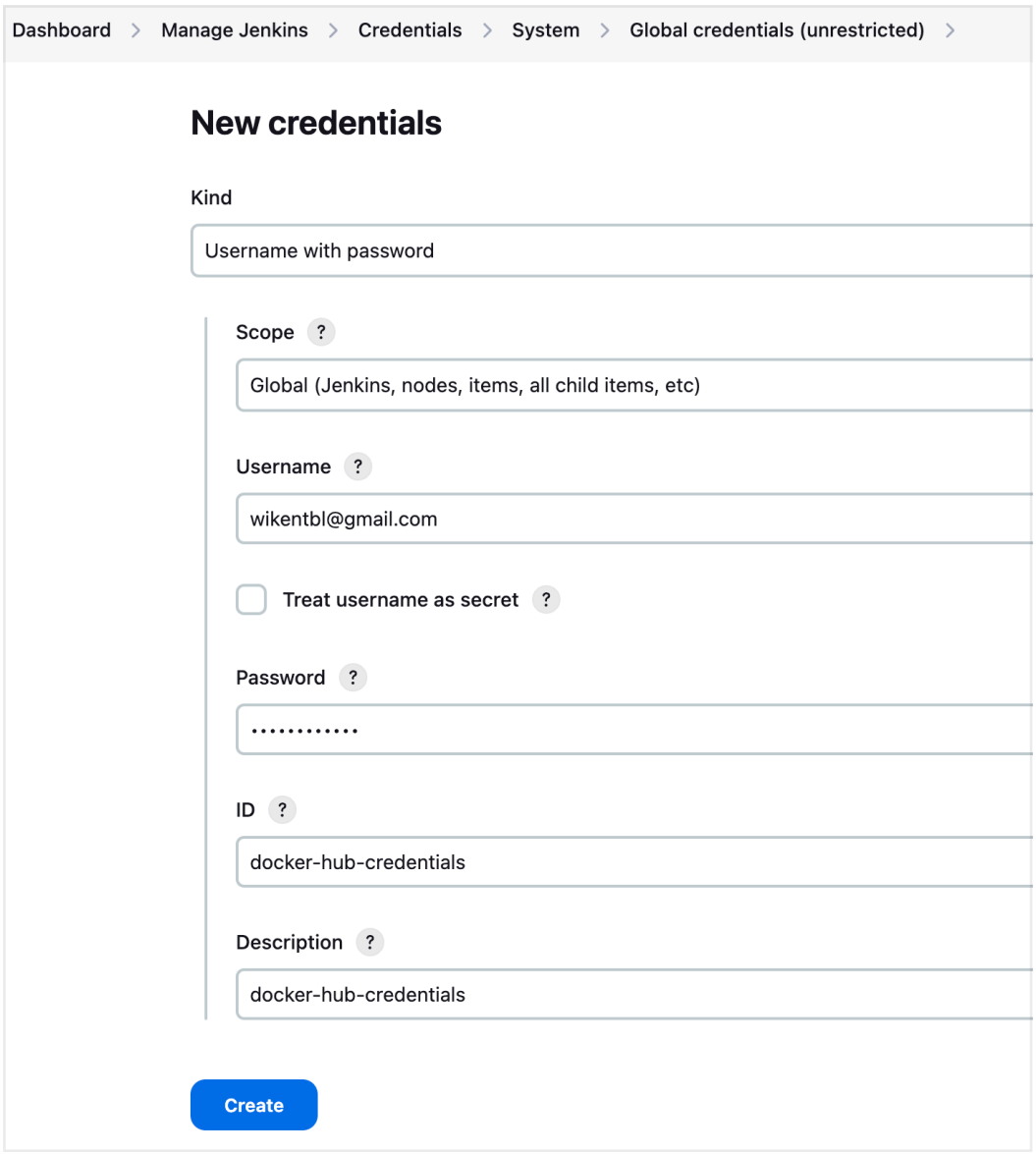


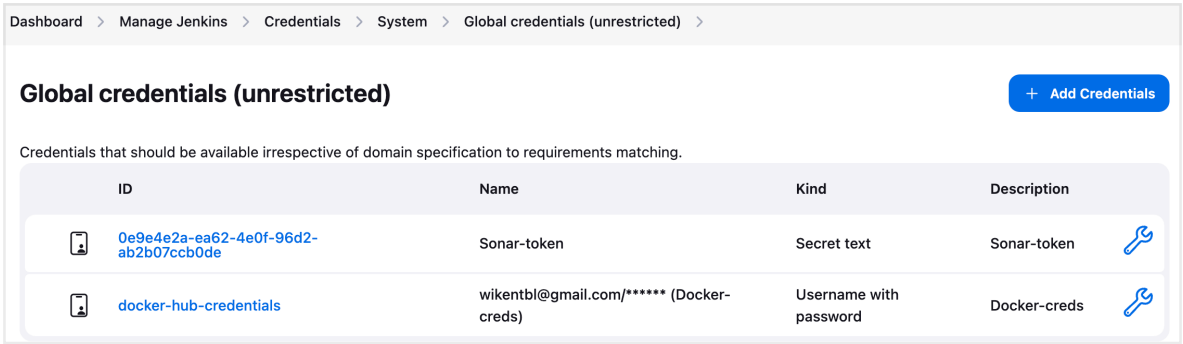
- Add Docker to Tools
  - Dashboard → Manage Jenkins → Tools → Add Docker



    - Apply and Save

- Add DockerHub Username/Password to Global Credentials
    - Dashboard → Manage Jenkins → Credentials
    - We will set the ID = docker-hub-credentials
    - Set description = docker-hub-credentials





- Add Build and Push stage to pipeline script
  - Make sure to replace with your own docker hub account name

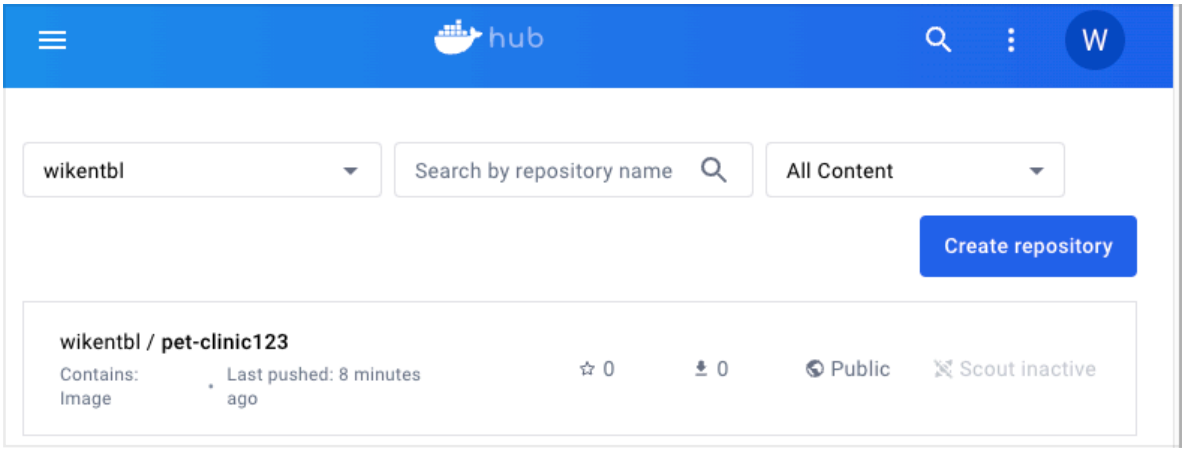
```
stage("Docker Build & Push") {
    steps {
        script {
            def DOCKER_IMAGE_NAME =
```

```
'wikentbl/pet-clinic123:latest'

withDockerRegistry(credentialsId: 'docker-hub-
credentials', toolName: 'docker') {
                        sh "docker build -t
petclinic1 ."
                        sh "docker tag petclinic1 $
{DOCKER_IMAGE_NAME}"
                        sh "docker push $
{DOCKER_IMAGE_NAME}"
                    }
                }
            }
        }
```

- Check that new image has been created in your Ubuntu server
  - Not important but just in case you get confused, please note that the screenshot below shows 2 different images because I ran the Pipeline twice. The second run updated the newer images with tag:latest.

```
ubuntu@ip-10-0-12-221:~$ docker images
REPOSITORY                TAG              IMAGE ID       CREATED          SIZE
petclinic1                latest           925c81c296c7   10 minutes ago   566MB
wikentbl/pet-clinic123    latest           925c81c296c7   10 minutes ago   566MB
wikentbl/pet-clinic123    <none>           ad2bcca082cc   16 minutes ago   566MB
sonarqube                 lts-community    775a3bef255b   7 weeks ago      603MB
openjdk                   8                b273004037cc   23 months ago    526MB
```

- Check to make sure the newly compiled image has been pushed



Step 7 — Check Docker image for vulnerabilities

```
        stage("TRIVY"){
            steps{
                sh " trivy image wikentbl/pet-
clinic123:latest"
            }
        }
```

Step 8 — Deploy image using Docker

- Add deploy stage to Pipeline

```
        stage("Deploy Using Docker") {
            steps {
                script {
                    def DOCKER_IMAGE_NAME =
'wikentbl/pet-clinic123:latest'
                    // Stop and remove any
existing container with the same name
                    // If first run, pet1 will
not exist, so return true regardless.
sh """
                    docker stop pet1 || true
docker rm pet1 || true                        """
// Run the new container                        sh "
docker run -d --name pet1 -p 8082:8080 ${DOCKER_
IMAGE_NAME}"
                }
            }
        }
```

Step 9 — docker check in browser

Step 10 – Terminate the VM Instance