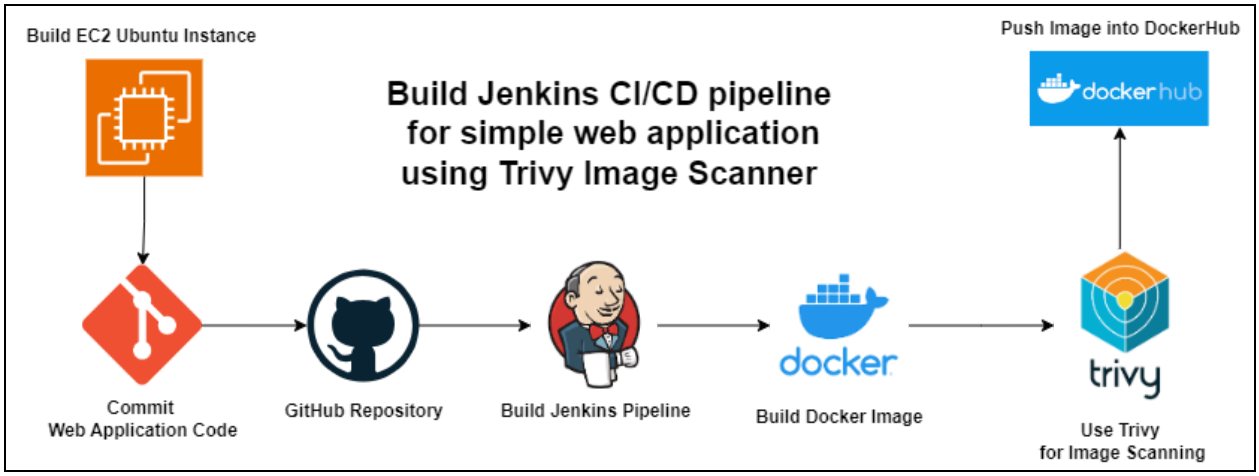
DevOps Project: Build Jenkins CI/CD pipeline for simple web application

Table of contents

Project Flow

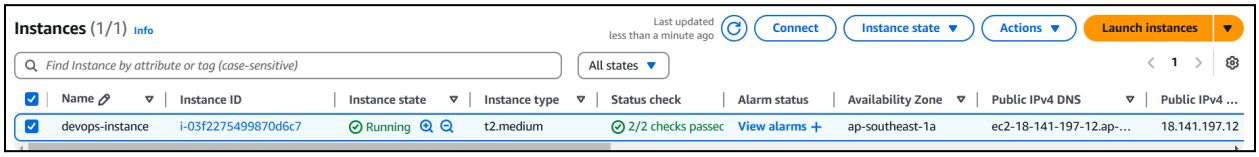


Tech Stack

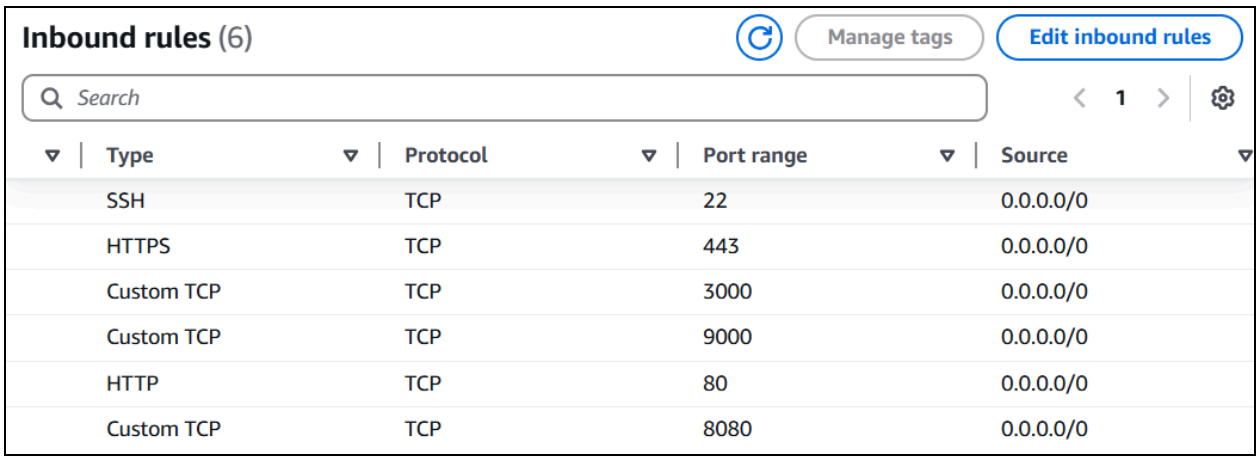
- Containerization: Docker
- Vulnerability scanning: Trivy
- CI/CD service: Jenkins
- Version control and collaboration: GitHub
- Cloud service: AWS EC2

Step 1: Set Up AWS EC2 Instance

1. Create EC2 Ubuntu Instance named as **devops-instance**. Select an appropriate instance type (e.g., **t2.medium**).
2. Launch the instance.



3. Configure Security Group: Allow inbound traffic on ports 22 (SSH), 8080 (Jenkins), 3000 (Web Application), HTTP, HTTPS.



Type	Protocol	Port range	Source
SSH	TCP	22	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0
Custom TCP	TCP	3000	0.0.0.0/0
Custom TCP	TCP	9000	0.0.0.0/0
HTTP	TCP	80	0.0.0.0/0
Custom TCP	TCP	8080	0.0.0.0/0

Step 2: Install Jenkins on EC2 Instance

1. **Connect to Instance:**

Use SSH to connect to the newly created EC2 instance.

ssh -i your-key-pair.pem ubuntu@your-ec2-instance-ip

ssh -i "C:\Users\Administrator\OneDrive\Máy tính\newkeypair.pem" ubuntu@ec2-13-212-207-177.ap-southeast-1.compute.amazonaws.com

2. **Install Java and Jenkins (run command):**

Installation Java

sudo apt update

sudo apt install fontconfig openjdk-17-jre

```
java --version
```

Install Jenkins - Long Term Support release version

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
        https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

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-get update

sudo apt-get install jenkins

jenkins --version
```

3. Install Docke (run command):

```
sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee
/etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo
\"$VERSION_CODENAME\") stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null

sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin

sudo usermod -aG docker $USER

sudo systemctl start docker

sudo systemctl enable docker
```

newgrp docker

docker --version.

4. **Install Trivy(run command):**

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

trivy --version

5. **Ensure Docker Permissions for Jenkins (run command):**

sudo usermod -aG docker jenkins

sudo systemctl restart jenkins

sudo systemctl restart docker

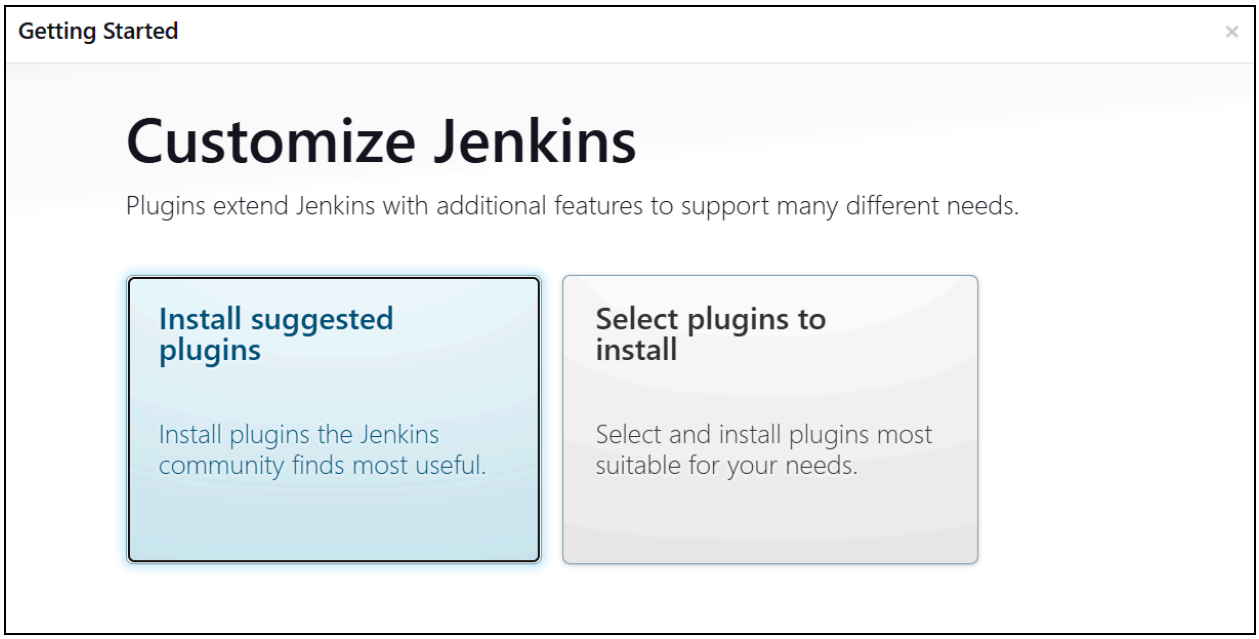
sudo chmod 666 /var/run/docker.sock

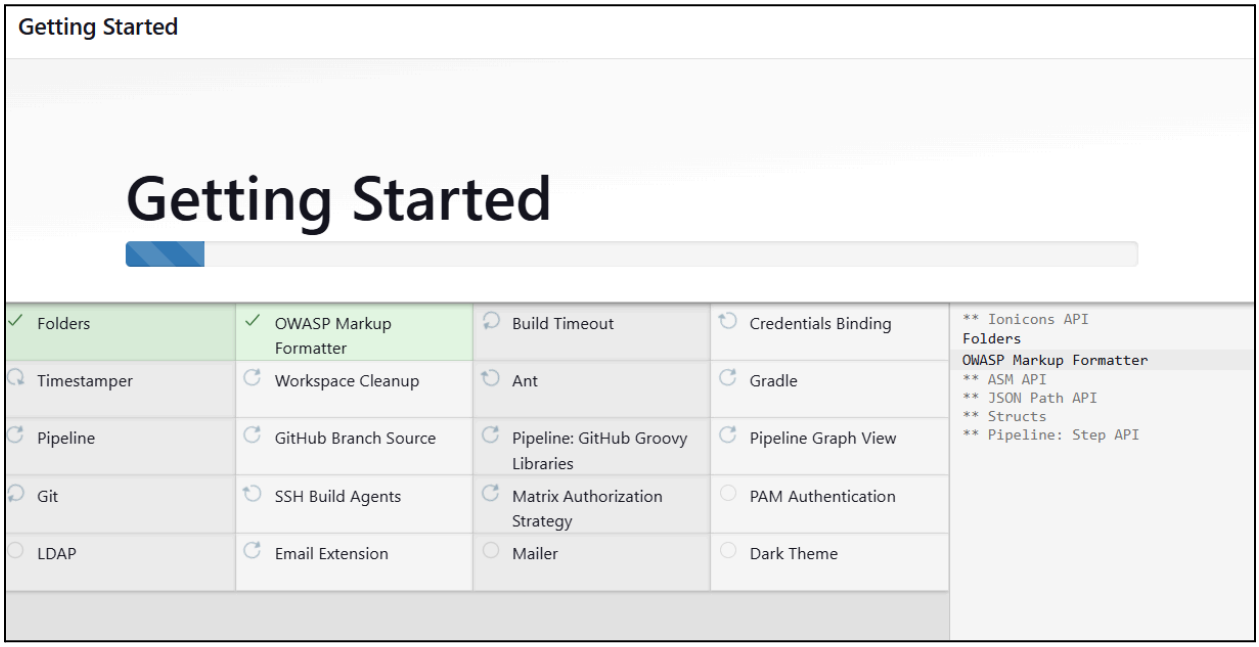
6. **Access Jenkins Dashboard (run command):**

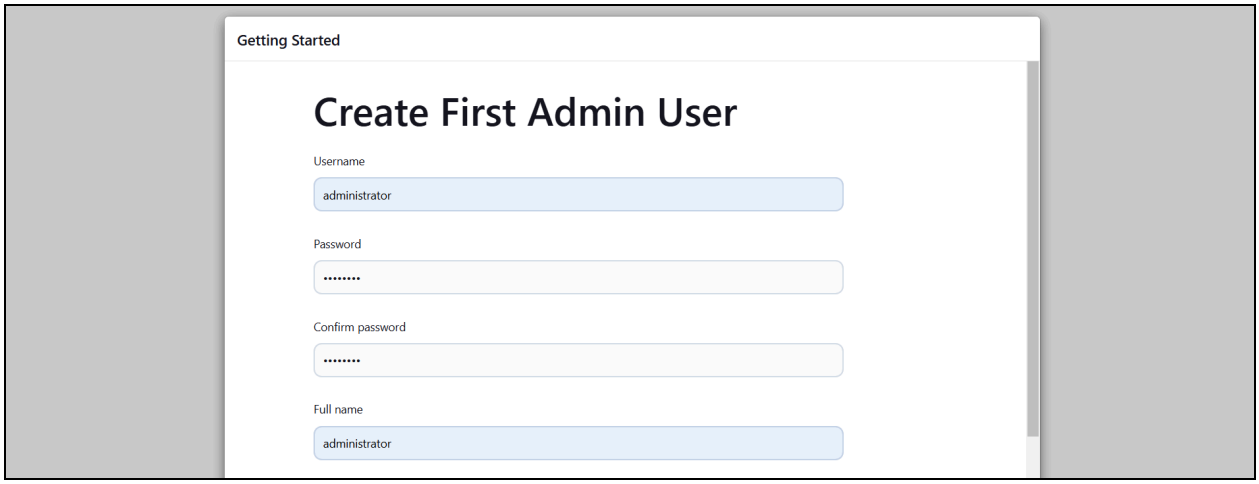
Navigate to http://your-ec2-instance-ip:8080. Unlock Jenkins using the initial admin password found in **/var/lib/jenkins/secrets/initialAdminPassword.**

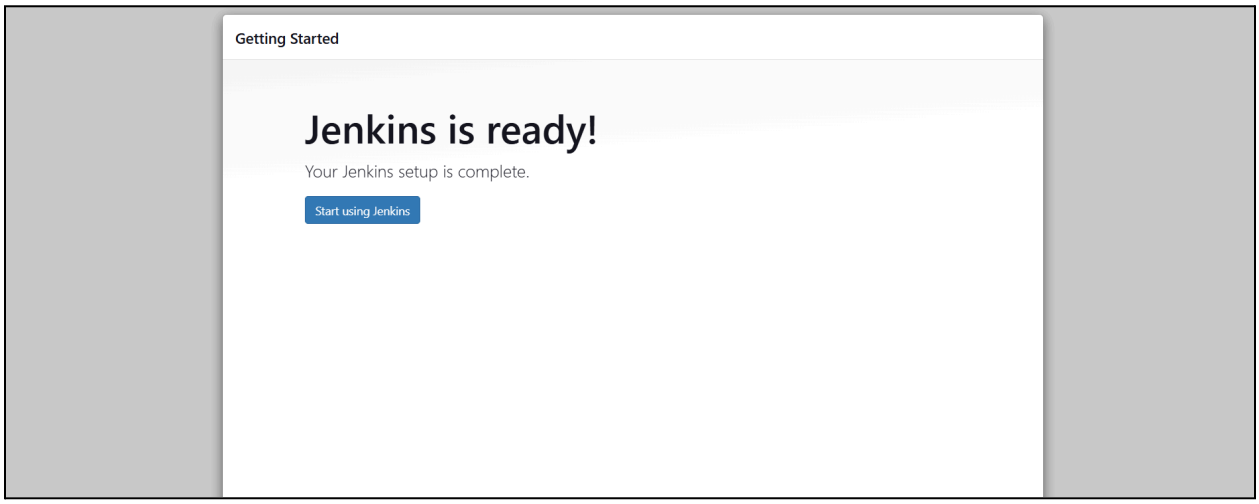
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Follow the setup wizard to install suggested plugins and create an admin user.


Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.


Getting Started
Getting Started
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline Graph View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
Dark Theme
** Ionicons API
Folders
OWASP Markup Formatter
** ASM API
** JSON Path API
** Structs
** Pipeline: Step API





Step 3: Set Up GitHub Repository

Link: https://github.com/nickldn2211/SimpleDevopsProject.git

Upload Project Files: Commit and push the following files to GitHub repository

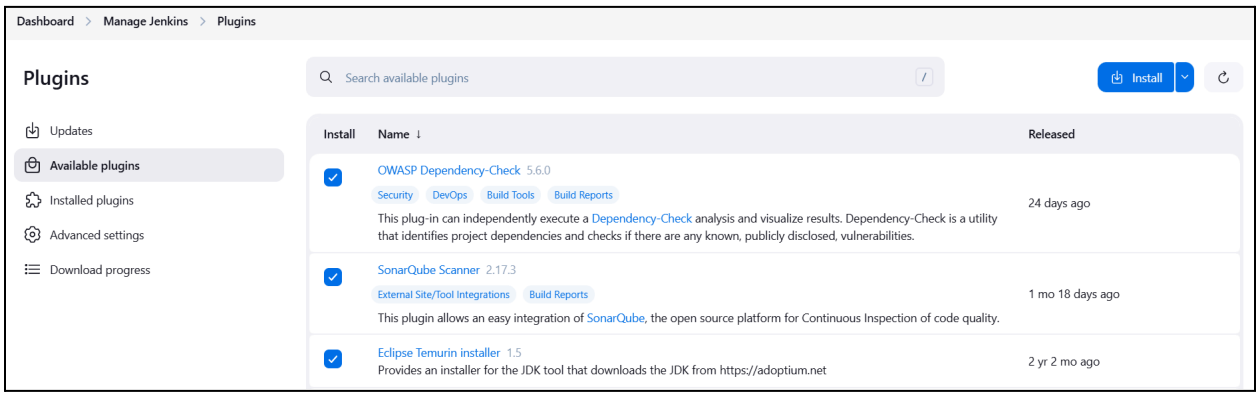
```
> node_modules
v public
  v css
    # styles.css
  v js
    JS scripts.js
  <> index.html
JS app.js
dockerfile
jenkinsfile
{} package-lock.json
{} package.json
```

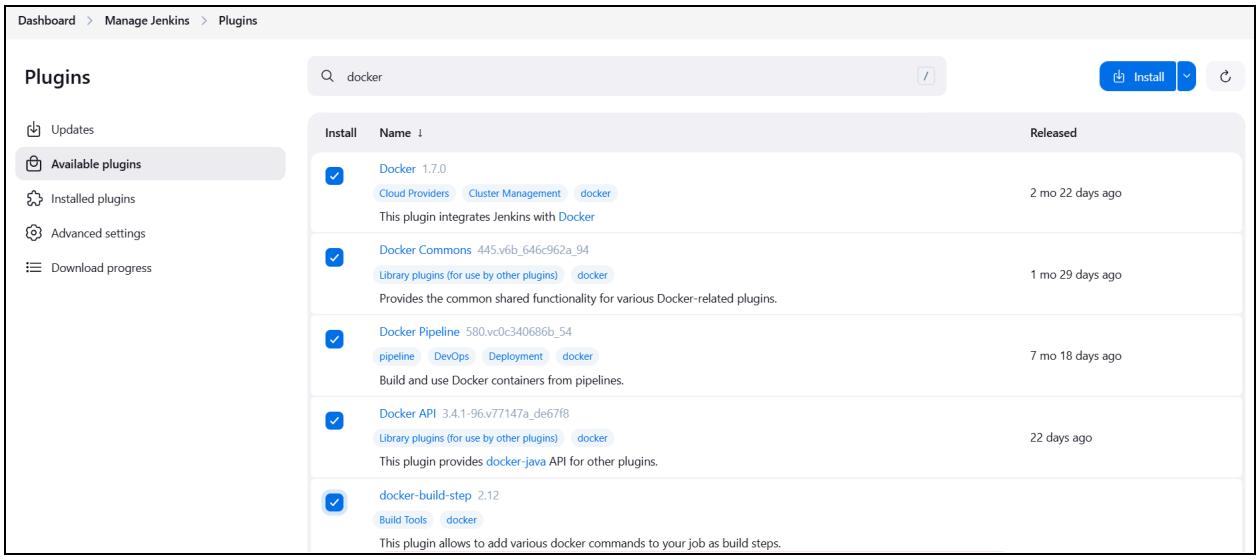
Step 4: Install Plugins and Tools in JenkinsInstall Plugins

1. Install Plugins

Go to Dashboard > Manage Jenkins > Plugins > Available Plugins > Install below plugins (can start using the installed plugins right away)

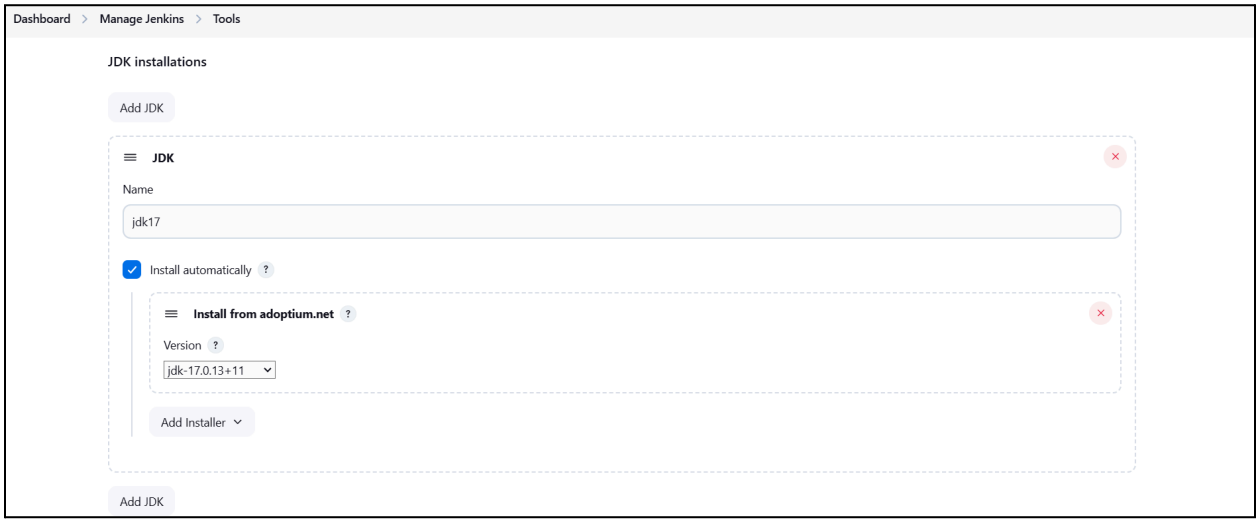
- **Eclipse Temurin Installer**
- **Docker**
- **Docker Commons**
- **Docker Pipeline**
- **Docker API**
- **docker-build-step**





2. Deploy JDK installations

Go to Manage Jenkins > Tools > Install JDK > Choose Install Automatically. Then choose Install from adoptium.net, specify version jdk-17.0.13+11. Name it as jdk17.



3. Deploy Docker installations

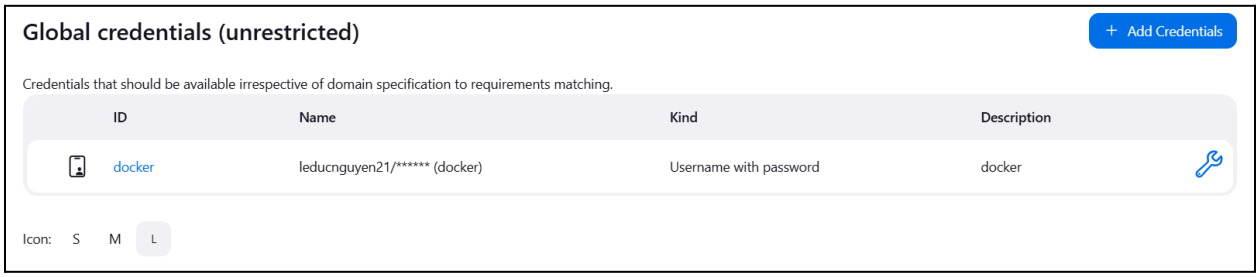
Go to Manage Jenkins > Tools > Docker Installations > Choose Install Automatically. Then choose Download from docker.com, with the latest Docker version. Name it as docker.

Step 5: Add Jenkins Credentials

Go to Manage Jenkins > Credentials > System > Global credentials (unrestricted)

Add credentials for DockerHub using username and password.

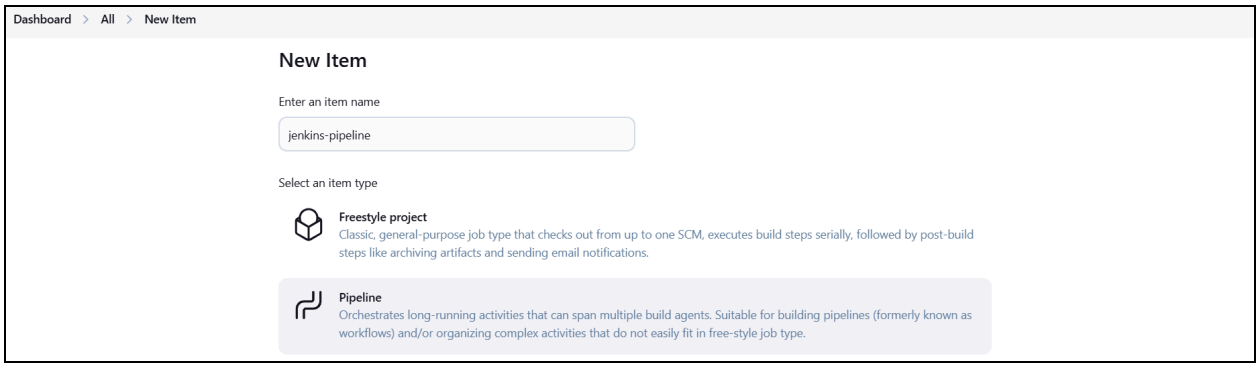
- Enter Docker Hub username and password.
- Give it an ID as **docker**



Step 6: Configure Jenkins Pipeline

Create a New Pipeline Job:

- In the Jenkins dashboard, click on "New Item."
- Enter a name for pipeline **jenkins-pipeline** and select "Pipeline" as the project type.



Step 7: Provide pipeline script in Jenkins pipeline

Go to Dashboard > Open **jenkins-pipeline** > Open Configure > Reach to pipeline section and paste the script as below > Click Apply and Save.

```
pipeline {
   agent any

   tools {
      jdk 'jdk17'
   }

   environment {
      DOCKER_IMAGE = "leducnguyen21/simpledevopsproject-image"
   }

   stages {
      stage('Clean Workspace') {
         steps {
            cleanWs()
         }
      }

      stage('Clone GitHub Repository') {
         steps {
            git branch: 'main', url:
'https://github.com/nickldn2211/SimpleDevopsProject.git'
         }
      }

      stage('Build Docker Image') {
```

```
        steps {
            script {
                docker.build("${DOCKER_IMAGE}:${env.BUILD_ID}")
            }
        }
    }

    stage('Run Unit Tests') {
        steps {
            script {
                sh 'mvn test'
            }
        }
    }

    stage('Security Scan') {
        steps {
            script {
                sh 'trivy image ${DOCKER_IMAGE}:${env.BUILD_ID}'
            }
        }
    }

    stage('Push Docker Image') {
        steps {
            script {
                docker.withRegistry('https://index.docker.io/v1/', 'dockerhub') {
                    docker.image("${DOCKER_IMAGE}:${env.BUILD_ID}").push()
                }
            }
        }
    }

    stage('Deploy Application') {
        steps {
            script {
                // Stop any running container with the same name (if any)
                sh "docker stop simpledevopsproject || true && docker rm
simpledevopsproject || true"

                // Run the new Docker container on port 3000
                sh "docker run -d --name simpledevopsproject -p 3000:3000
${DOCKER_IMAGE}:${env.BUILD_ID}"
```

```
            }
        }
    }

    stage('Post-Deployment Health Check') {
        steps {
            script {
                // Check if the application is running
                sh 'curl -f http://localhost:3000 || exit 1'
            }
        }
    }
  }

  post {
    success {
        echo 'Pipeline completed successfully!'
    }
    failure {
        echo 'Pipeline failed!'
    }
  }
}
```

Here is the list of stages in the pipeline:

1. Clean Workspace: Clears the workspace to ensure a clean environment.
1. Clone GitHub Repository: Clones the specified branch of the GitHub repository.
2. Build Docker Image: Uses the Dockerfile to build a Docker image.
3. Run Unit Tests: Executes unit tests using Maven to ensure code quality.
4. Security Scan: Runs a security scan on the Docker image using Trivy.
5. Push Docker Image: Pushes the Docker image to Docker Hub.
6. Deploy Application: Pulls down the Docker image from Docker Hub and runs the web application locally.
7. Post-Deployment Health Check: Verifies the application is running.

Post Actions:

- Success: Echoes a message indicating the pipeline completed successfully.
- Failure: Echoes a message indicating the pipeline failed.

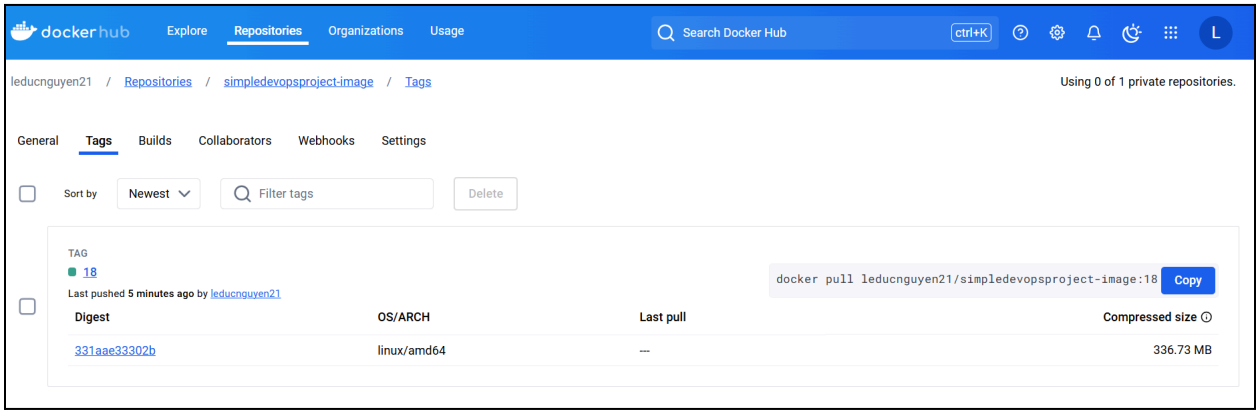
Step 7: Run the Pipeline

Monitor Pipeline Execution:

- In the Jenkins dashboard, go to the pipeline job. Then click Build now to trigger the pipeline.
- Monitor the execution by clicking on the build number and viewing the console output.

Step 8: Verify Deployment

1. Go to Dockerhub and check if the pipeline is published successfully to DockerHub or not.



2. Pull down image from DockerHub

In local machine, run command to pull images to local machine

docker pull leducnguyen21/simpledevopsproject-image:18

docker run -d --name simpledevopsproject-local -p 4000:3000 leducnguyen21/simpledevopsproject-image:18

Enable port 4000 for EC2 instance to access website application.

sgr-0f9ade4e933a0aa0e	4000	TCP	0.0.0.0/0	launch-wizard-3	–

3. Access Web Application:

Open a web browser and navigate to http://your-ec2-instance-ip:4000 to see the website running. In this case, this is a simple website application created for this DevOps project.