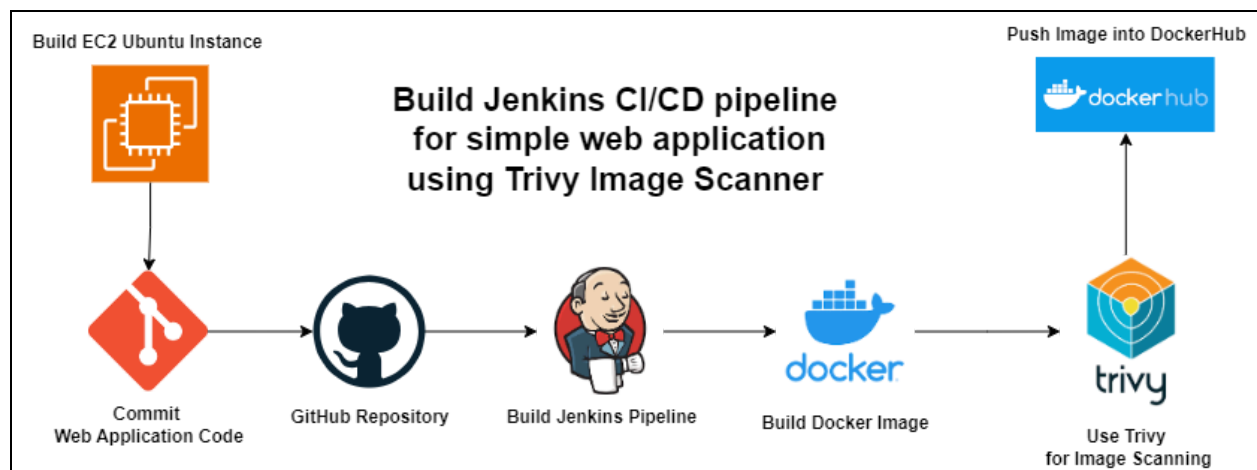


DevOps Project: Build Jenkins CI/CD pipeline for simple web application

Table of contents

Project Flow.....	0
Tech Stack.....	0
Step 1: Set Up AWS EC2 Instance.....	1
Step 2: Install Jenkins on EC2 Instance.....	1
Step 3: Set Up GitHub Repository.....	5
Step 4: Install Plugins and Tools in JenkinsInstall Plugins.....	6
Step 5: Add Jenkins Credentials.....	8
Step 6: Configure Jenkins Pipeline.....	8
Step 7: Run the Pipeline.....	11
Step 8: Verify Deployment.....	12

Project Flow

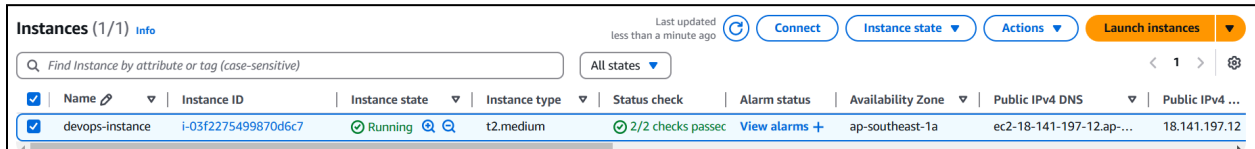


Tech Stack

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

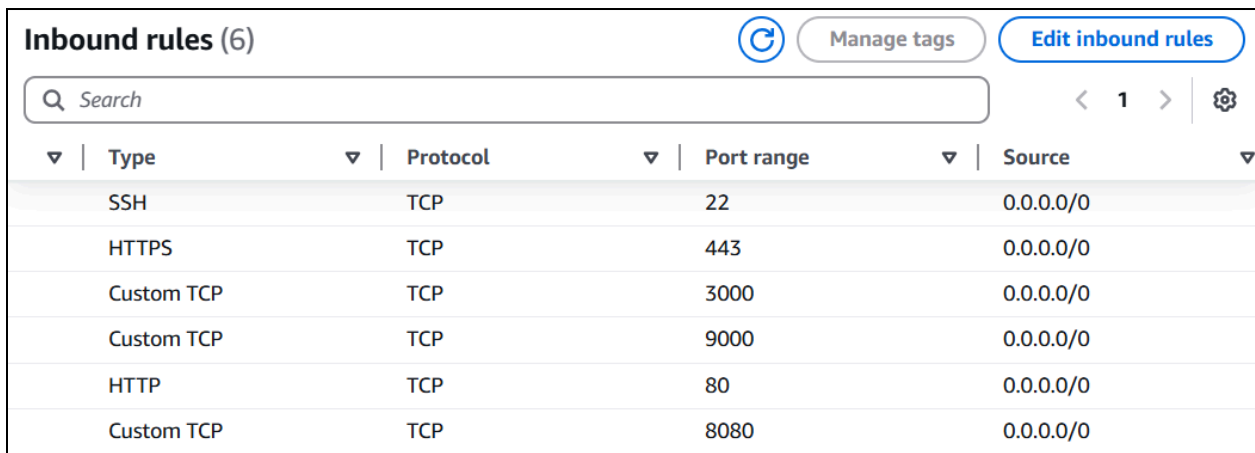
Step 1: Set Up AWS EC2 Instance

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



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
devops-instance	i-03f2275499870d6c7	Running	t2.medium	2/2 checks passed	View alarms	ap-southeast-1a	ec2-18-141-197-12.ap-...	18.141.197.12

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 Docker (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.

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

✓ Folders	✓ OWASP Markup Formatter	⌚ Build Timeout	⌚ Credentials Binding	** Ionicons API Folders OWASP Markup Formatter ** ASM API ** JSON Path API ** Structs ** Pipeline: Step API
⌚ Timestamp	⌚ 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	

The screenshot shows the 'Getting Started' page of Jenkins with the title 'Create First Admin User'. It contains four input fields: 'Username' with the value 'administrator', 'Password' with masked characters '*****', 'Confirm password' with masked characters '*****', and 'Full name' with the value 'administrator'.

Getting Started

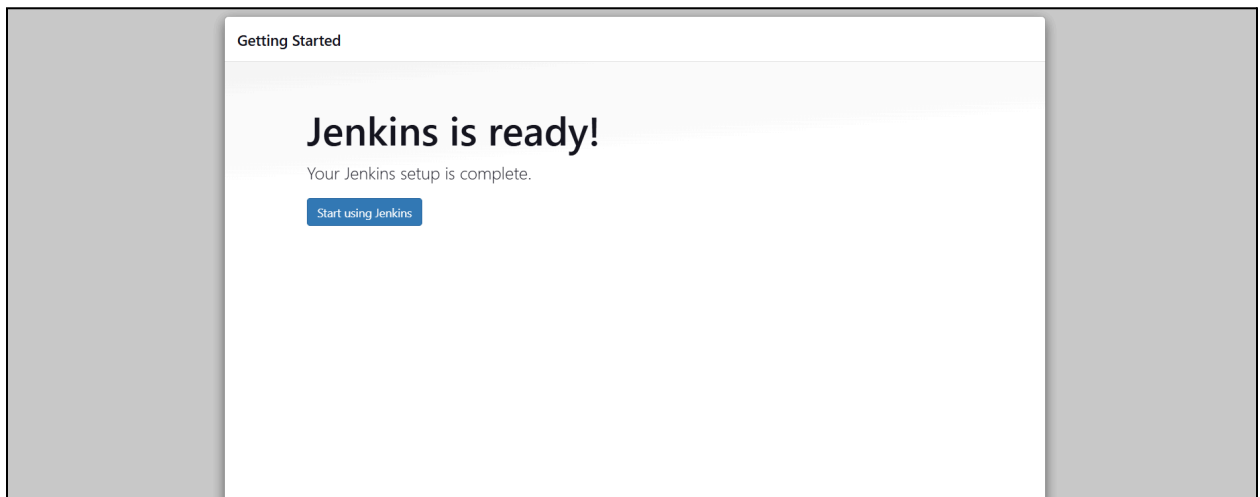
Create First Admin User

Username
administrator

Password

Confirm password

Full name
administrator



Step 3: Set Up GitHub Repository

Link: <https://github.com/nickIdn2211/SimpleDevopsProject.git>

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

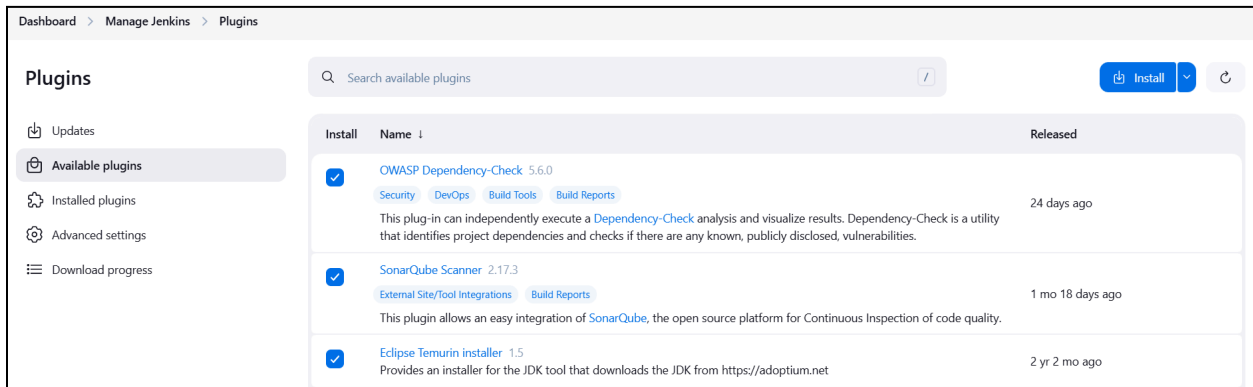


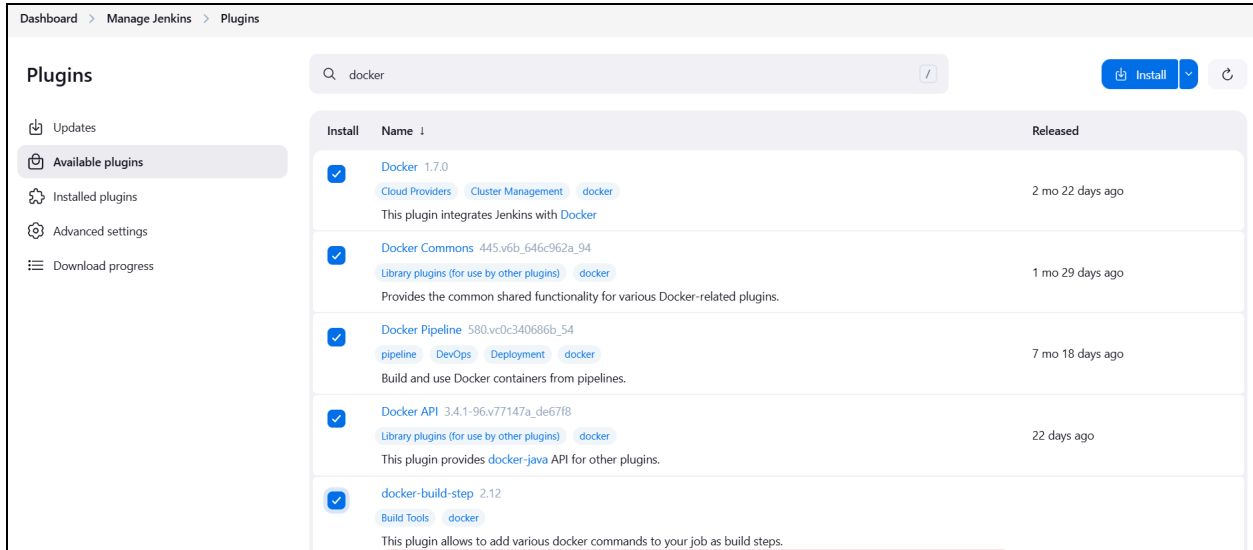
Step 4: Install Plugins and Tools in Jenkins

1. Install Plugins

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

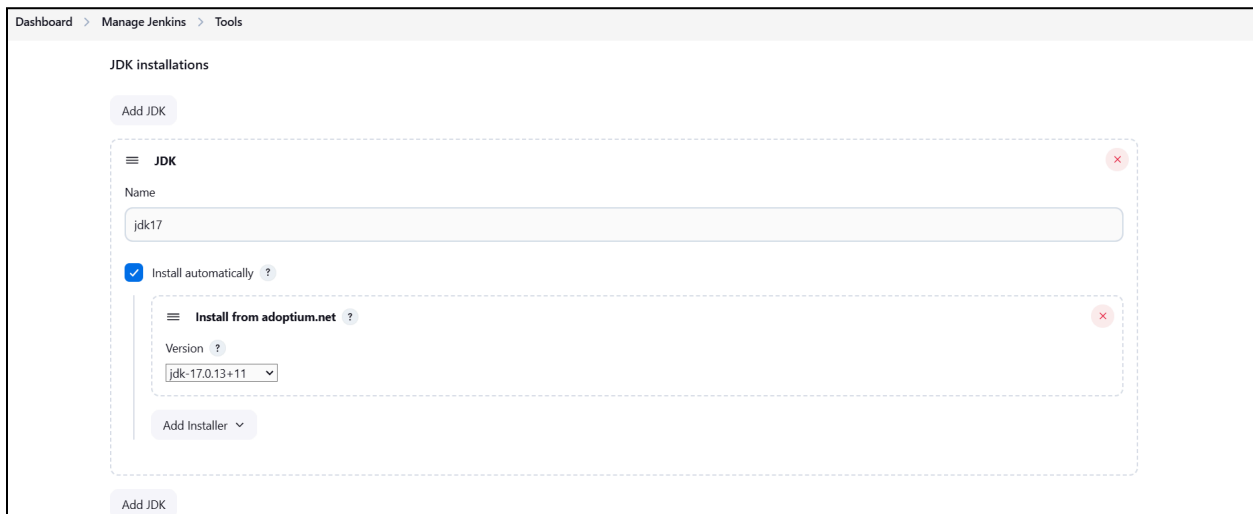
- **Eclipse Temurin Installer**
- **Docker**
- **Docker Commons**
- **Docker Pipeline**
- **Docker API**
- **docker-build-step**





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.

Dashboard > Manage Jenkins > Tools

Docker installations

Add Docker

Docker

Name

docker

☒ Install automatically ?

Download from docker.com

Docker version ?

latest

Add Installer

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**

Global credentials (unrestricted) + Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
 docker	leducnguyen21/***** (docker)	Username with password	docker 

Icon: S M L

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.


Dashboard > All > New Item

New Item

Enter an item name

Select an item type

**Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

**Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job 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/nickIdn2211/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 simpledevsproject || true && docker rm
simpledevsproject || true"

            // Run the new Docker container on port 3000
            sh "docker run -d --name simpledevsproject -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:

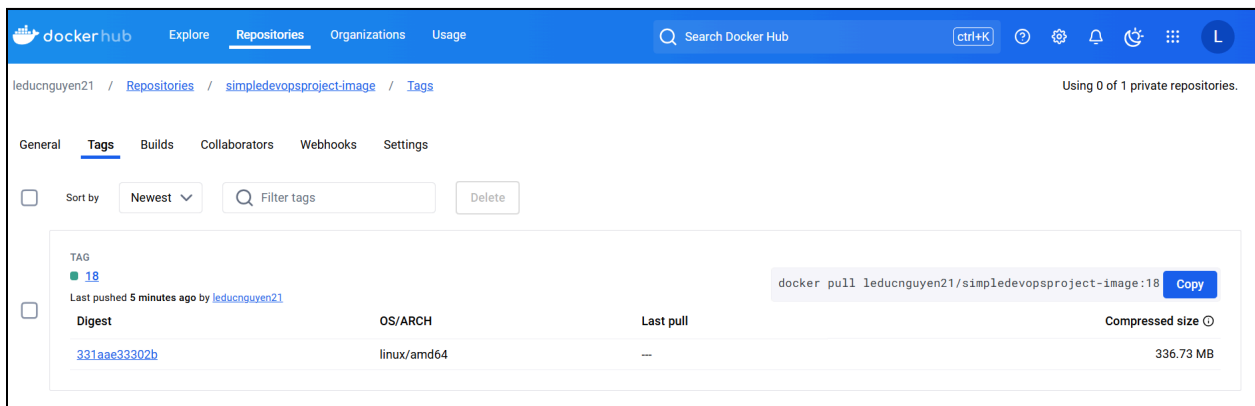
- In the Jenkins dashboard, go to the pipeline job. Then click Build now to trigger the pipeline.
- Monitor the execution by clicking on the build number and viewing the console output.



Build #18 (Jan 8, 2025, 5:24:56 PM) is shown as successful with a green checkmark. It includes buttons for 'Add description' and 'Keep this build forever'. The build was started by user 'administrator' 45 seconds ago and took 32 seconds. A timeline shows 9 ms waiting, 32 sec build duration, and 32 sec total from scheduled to completion. The build is linked to a git repository: <https://github.com/nickldn2211/SimpleDevopsProject.git>, revision 98f5b906fb0141f730d0356c3441c833b4417a29, at refs/remotes/origin/main. A code icon indicates 'No changes'.

Step 8: Verify Deployment

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



The DockerHub page for repository 'simpledevopsproject-image' shows a tag '18' pushed 5 minutes ago by 'leducnguyen21'. The image details table is as follows:

Digest	OS/ARCH	Last pull	Compressed size
331aae33302b	linux/amd64	---	336.73 MB

A 'docker pull leducnguyen21/simpledevopsproject-image:18' command is shown with a 'Copy' button.

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-Of9ade4e933a0aa0e	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.

