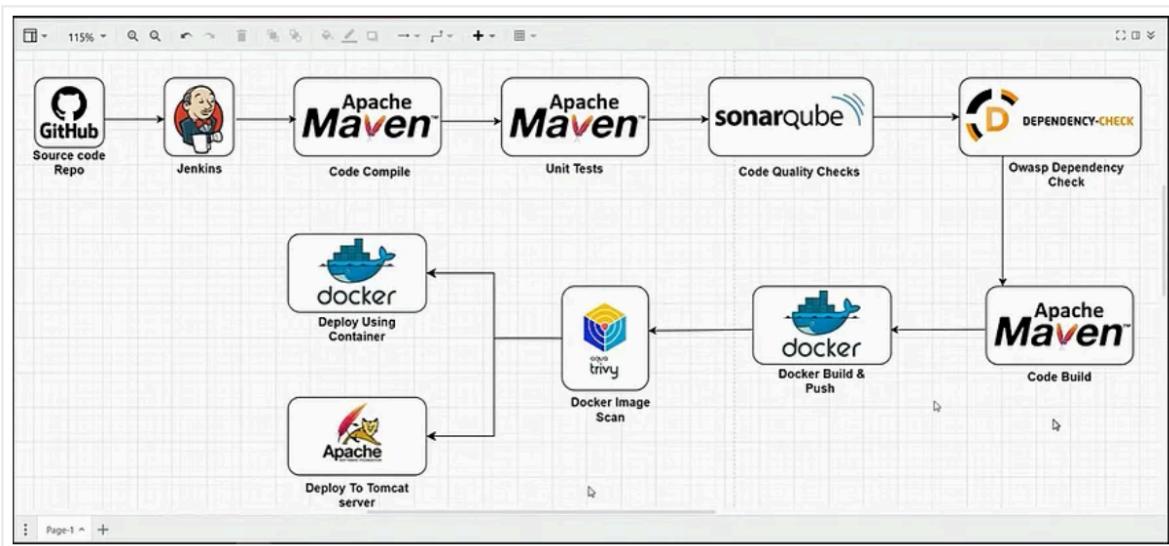


Pet Clinic - DevSecOps CI/CD for Java Based App on Tomcat Server

<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
```

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

.....

- 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

Log in to SonarQube

[Log in](#) [Cancel](#)

- Update password

Update your password

This account should not use the default password.

Enter a new password

All fields marked with * are required

Old Password *

New Password *

Confirm Password *

[Update](#)

- 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/
```

JDK installations

Add JDK

≡ JDK

Name

jdk17

JAVA_HOME

/usr/lib/jvm/java-17-openjdk-amd64/



Install automatically



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

Maven installations

Add Maven

☰ Maven

Name

maven3



Install automatically ?

☰ Install from Apache

Version

3.9.8

Add Installer ▾

○ Apply and Save

- Test using Jenkins Declarative Pipeline
 - Dashboard → New item

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

Enter an item name

PetClinic

» Required field

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.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

```
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

The screenshot shows the SonarQube Administration interface under the General Settings tab. On the left is a sidebar with links: Analysis Scope, Authentication, DevOps Platform Integrations, External Analyzers, General, Housekeeping, JaCoCo, Languages, New Code, SCM, and Security. The main content area has a search bar at the top. Below it, the 'Duplications' section contains the 'Cross project duplication detection' setting, which is described as deprecated and set to detect duplicates at the project level. A toggle switch is shown as off (disabled), with '(default)' written next to it.

SCM Accounts	Last connection	Groups	Tokens
	< 1 hour ago	sonar-administrators sonar-users	0 Update Tokens

- Copy the token string for next step.

Generate Tokens

Name	Expires in
Enter Token Name	30 days

New token "T1" has been created. Make sure you copy it now, you won't be able to see it again!

[Copy](#) squ_6f377c9b8b5d38ea4db3a3e41458549b886955b4

Name	Type	Project	Last use	Created	Expiration
T1	User		Never	June 25, 2024	July 25, 2024

[Revoke](#)

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

Credentials

T	P	Store ↓	Domain
---	---	---------	--------

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

Add credentials

Icon:

New credentials

Kind

- ✓ Username with password
- GitHub App
- SSH Username with private key
- Secret file
- Secret text**
- Certificate

Username ?

New credentials

Kind

Secret text

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Secret

.....

ID ?

sonar-token

Description ?

sonar-token

Create

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

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as envir

Environment variables

SonarQube installations

List of SonarQube installations

Name

- 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

The screenshot shows the Jenkins 'Tools' configuration page under 'Manage Jenkins'. The title is 'SonarQube Scanner installations'. A button 'Add SonarQube Scanner' is visible. A new configuration entry for 'sonar-scanner' is being created, indicated by a dashed border. The 'Name' field contains 'sonar-scanner'. The 'Install automatically' checkbox is checked. Under 'Install from Maven Central', the 'Version' is set to 'SonarQube Scanner 6.1.0.4477'. A 'Add Installer' dropdown menu is shown.

- 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

The screenshot shows the SonarQube interface with the 'Projects' tab selected. A search bar at the top right contains the placeholder 'Search for projects...'. Below it, a table displays the Petclinic project's analysis results:

Petclinic		Passed
Bugs	4	(B)
Vulnerabilities	0	(A)
Hotspots Reviewed	0.0%	(E)
Code Smells	74	(A)
Coverage	0.0%	(O)
Duplications	11.0%	(O)
Lines	15k	(M) CSS

On the left, there are filters for Quality Gate (Passed, Failed), Reliability (A rating, B rating), and a search bar for 'My Favorites'.

○
○

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

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

The Jenkins dashboard features a 'Dashboard' header with a 'New Item' button. Below are three main links:

- Build History
- Manage Jenkins
- My Views

Enter an item name

PetClinic

» Required field



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.



Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



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.

The pipeline should have the following Pipeline script so far:

Pipeline script

```
Script ?  
1 * pipeline {  
2     agent any  
3  
4     tools{  
5         jdk 'jdk17'  
6         maven 'maven3'  
7     }  
8  
9     environment {  
10        SCANNER_HOME=tool 'sonar-scanner'  
11    }  
12  
13    stages{  
14  
15        stage("Git Checkout"){  
16            steps{  
17                git branch: 'main', changelog: false, poll: false, url: 'https://github.com/wikenhub/PetClinic.git'  
18            }  
19        }  
20  
21        stage("Compile"){  
22            steps{  
23                sh "mvn clean compile"  
24            }  
25        }  
26  
27        stage("Test Cases"){  
28            steps{  
29                sh "mvn test"  
30            }  
31        }  
32  
33        stage("Sonarqube Analysis ") {  
34            steps{  
35                withSonarQubeEnv('sonar-server') {  
36                    sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=PetClinic \  
37                    -Dsonar.java.binaries=. \  
38                    -Dsonar.projectKey=PetClinic'''  
39                }  
40            }  
41        }  
42    }  
43}  
44}
```



Use Groovy Sandbox ?

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



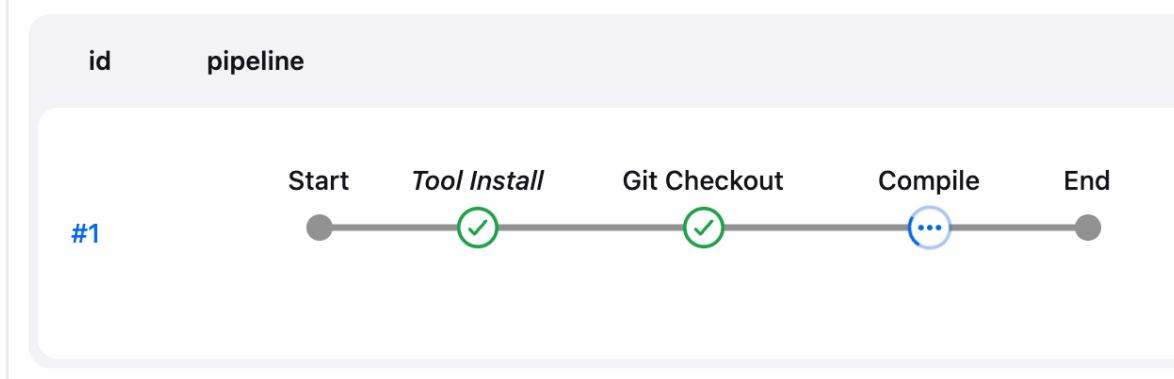
Build Now

and then

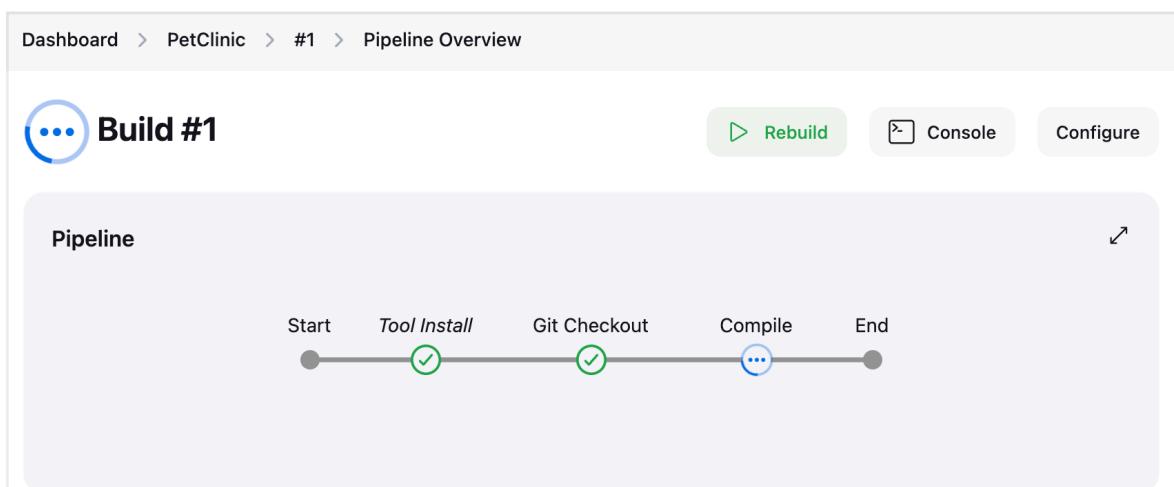


from the left tab

Build PetClinic



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



Build #1

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

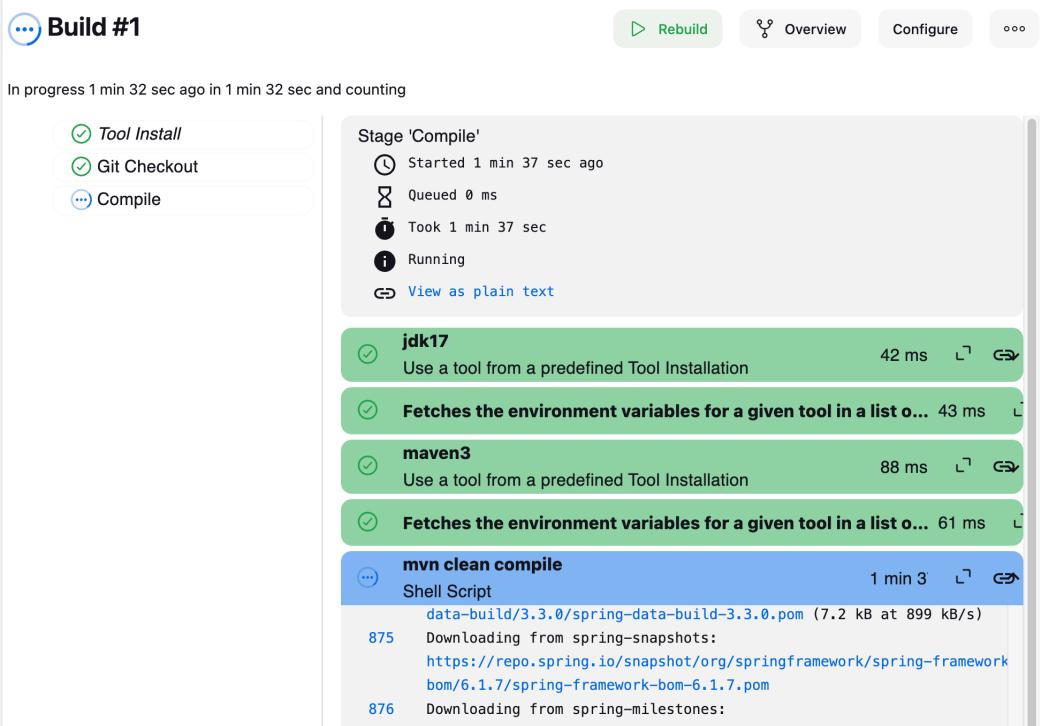
Fetches the environment variables for a given tool in a list o... 43 ms

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 known 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

The screenshot shows the Jenkins Manage Jenkins > Plugins interface. A search bar at the top contains the text "Docker". Below it, a table lists five Docker-related plugins:

Install	Name	Released
<input checked="" type="checkbox"/>	Docker 1.6.2 Cloud Providers Cluster Management docker	21 days ago
<input checked="" type="checkbox"/>	Docker Commons 439.va_3cb_0a_6a_fb_29 Library plugins (for use by other plugins) docker	11 mo ago
<input checked="" type="checkbox"/>	Docker Pipeline 580.vc0c340686b_54 pipeline DevOps Deployment docker	1 mo 4 days ago
<input checked="" type="checkbox"/>	Docker API 3.3.6-90.ve7c5c7535ddd Library plugins (for use by other plugins) docker	22 days ago
<input checked="" type="checkbox"/>	docker-build-step 2.12 Build Tools docker	28 days ago

A sidebar on the left includes links for Updates, Available plugins (which is selected), Installed plugins, Advanced settings, and Download progress.

Plugins

Download progress

	Preparation	
 Updates	OWASP Dependency-Check	 Success
 Available plugins	Loading plugin extensions	 Success
 Installed plugins	Cloud Statistics	 Success
 Advanced settings	Authentication Tokens API	 Success
 Download progress	Docker Commons	 Success
	Apache HttpComponents Client 5.x API	 Success
	Docker API	 Success
	Docker	 Success
	Docker Commons	 Success
	Docker Pipeline	 Success
	Docker API	 Success
	Javadoc	 Success
	JSch dependency	 Success
	Maven Integration	 Success
	docker-build-step	 Success
	Loading plugin extensions	 Success

→ [Go back to the top page](#)
(you can start using the installed plugins right away)

→ Restart Jenkins when installation is complete and no jobs are running

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

Docker installations

Add Docker

☰ Docker

Name

docker

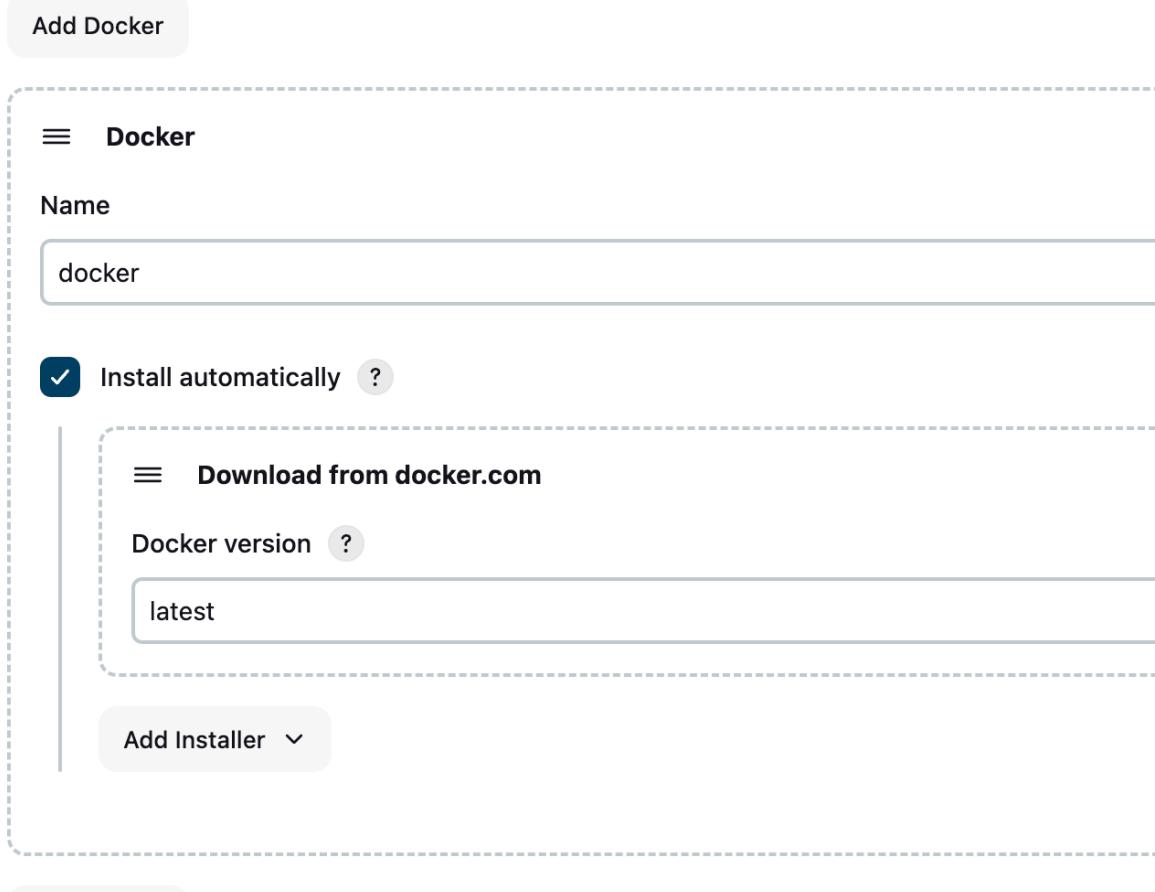
Install automatically ?

☰ Download from docker.com

Docker version ?

latest

Add Installer ▾



- 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

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

wikentbl@gmail.com

Treat username as secret ?

Password ?

.....

ID ?

docker-hub-credentials

Description ?

docker-hub-credentials

Create

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

+ Add Credentials

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

ID	Name	Kind	Description
0e9e4e2a-ea62-4e0f-96d2-ab2b07ccb0de	Sonar-token	Secret text	Sonar-token
docker-hub-credentials	wikentbl@gmail.com/***** (Docker-creds)	Username with password	Docker-creds

- 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 =
```

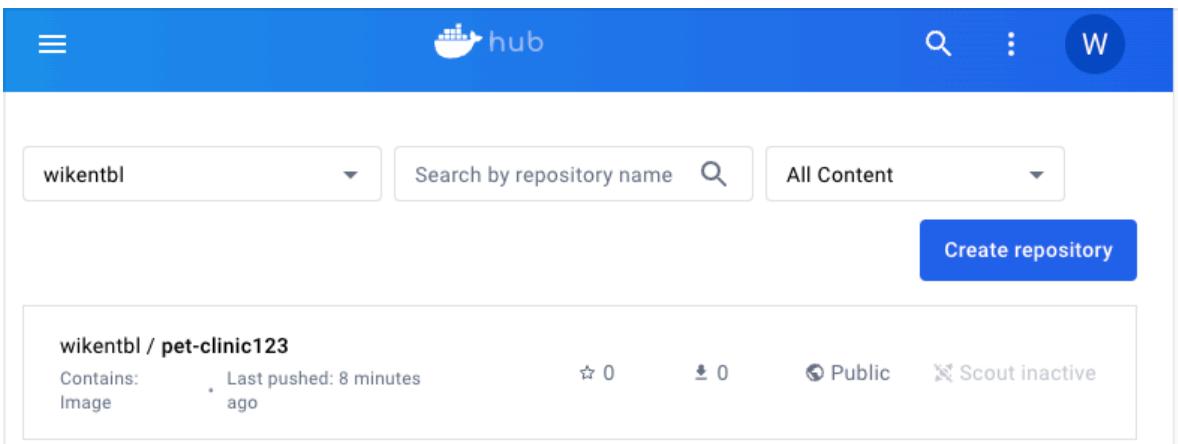
```
'wikentbl/pet-clinic123:latest'

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>   ad2bcc082cc   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
                    docker run -d --name pet1 -p 8082:8080 ${DOCKER_
IMAGE_NAME}"
                }
        }
    }
}
```

Step 9 — docker check in browser

Step 10 – Terminate the VM Instance