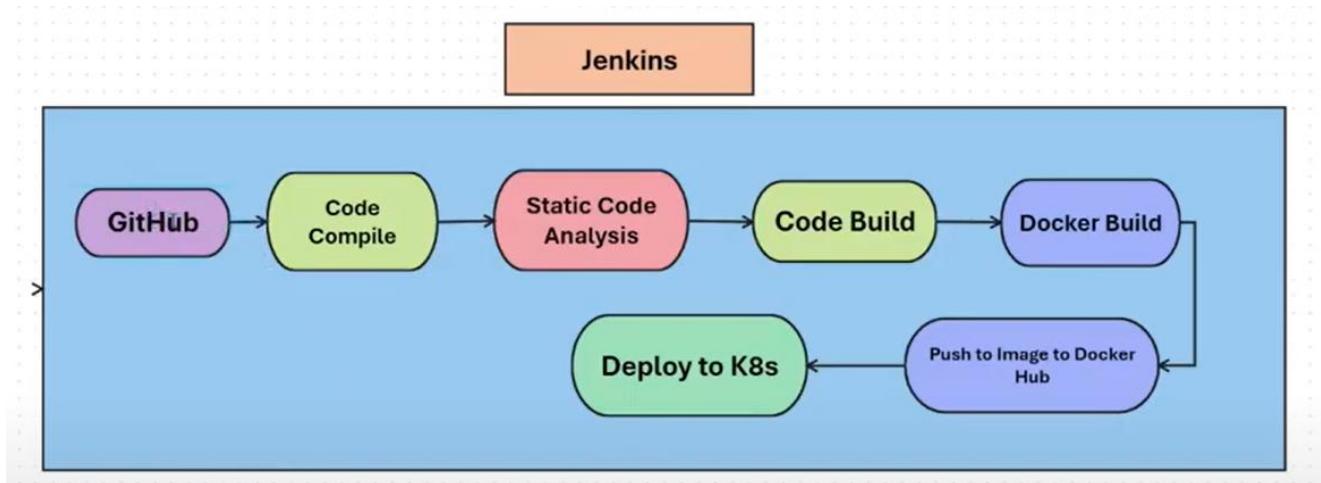


CI/CD Pipeline Implementation using Jenkins with Kubernetes, SonarQube, Docker, Maven



Initially, we will checkout the code in GitHub and then we will compile the code using Maven. Then we use SonarQube for static code analysis, then we build the code using Maven, then we will use Docker to build the image, once the image is built, we will push it into the Docker hub. From the Docker hub, we will be deploying it on the Kubernetes environment.

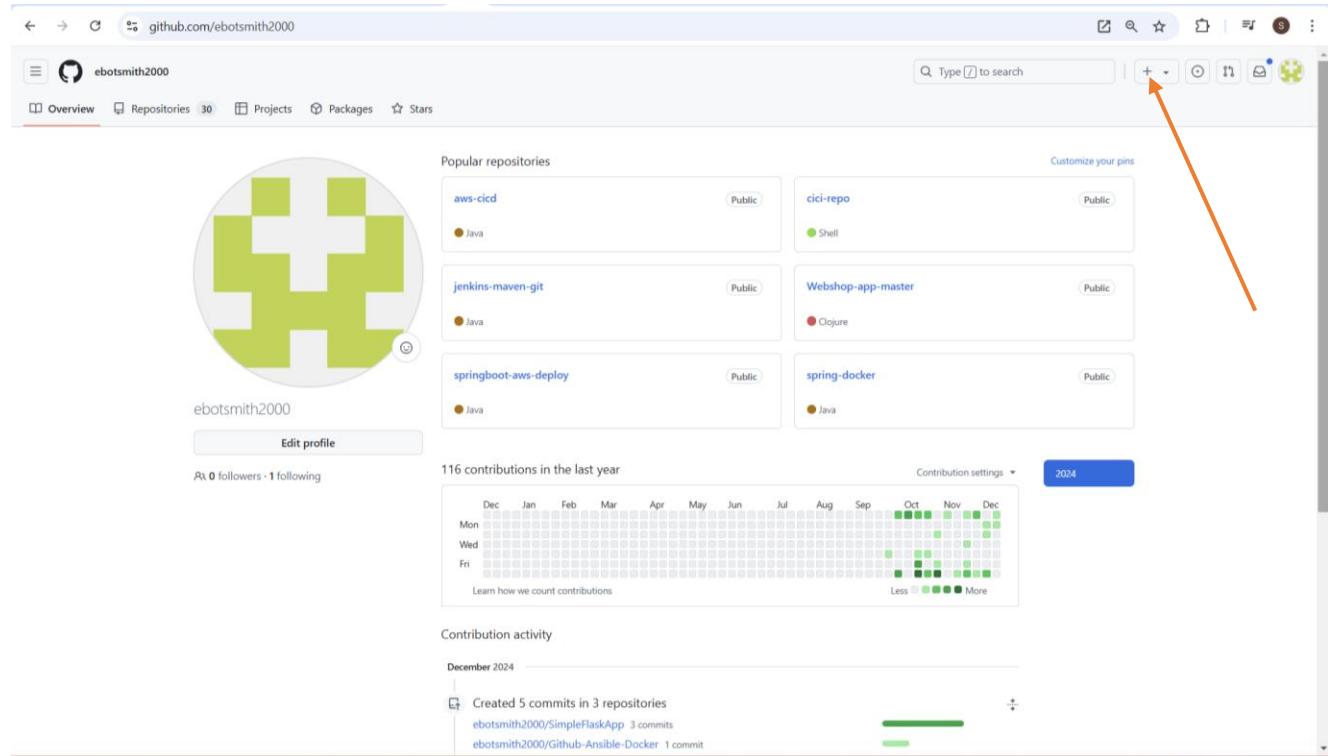
We will see all the plugins installation of all the tools and how the application runs. We will follow these steps:

STEPS:

- **Step 1:** Create a GitHub repository and upload the project files to it
- **Step 2:** Create EC2 instance that will serve as Jenkins Server
- **Step 3:** Add Port 8080 in the inbound rule of the Instance for Jenkins
- **Step 4:** Connect to the Jenkins server
- **Step 5:** Install Java JDK 17
- **Step 6:** Install Jenkins
- **Step 7:** Access Jenkins GUI on browser
- **Step 8:** Create and Configure Pipeline on Jenkins
- **Step 9:** Install Pipeline Stage View plugin
- **Step 10:** Install and configure Maven with Jenkins
- **Step 11:** Install and Configure SonarQube with Jenkins
- **Step 12:** Configure Maven for Code Build
- **Step 13:** Install Docker and Build Docker Image
- **Step 14:** Push the Image to Docker Hub
- **Step 15:** Deploy Image to Kubernetes
- **Step 16:** Automating the process

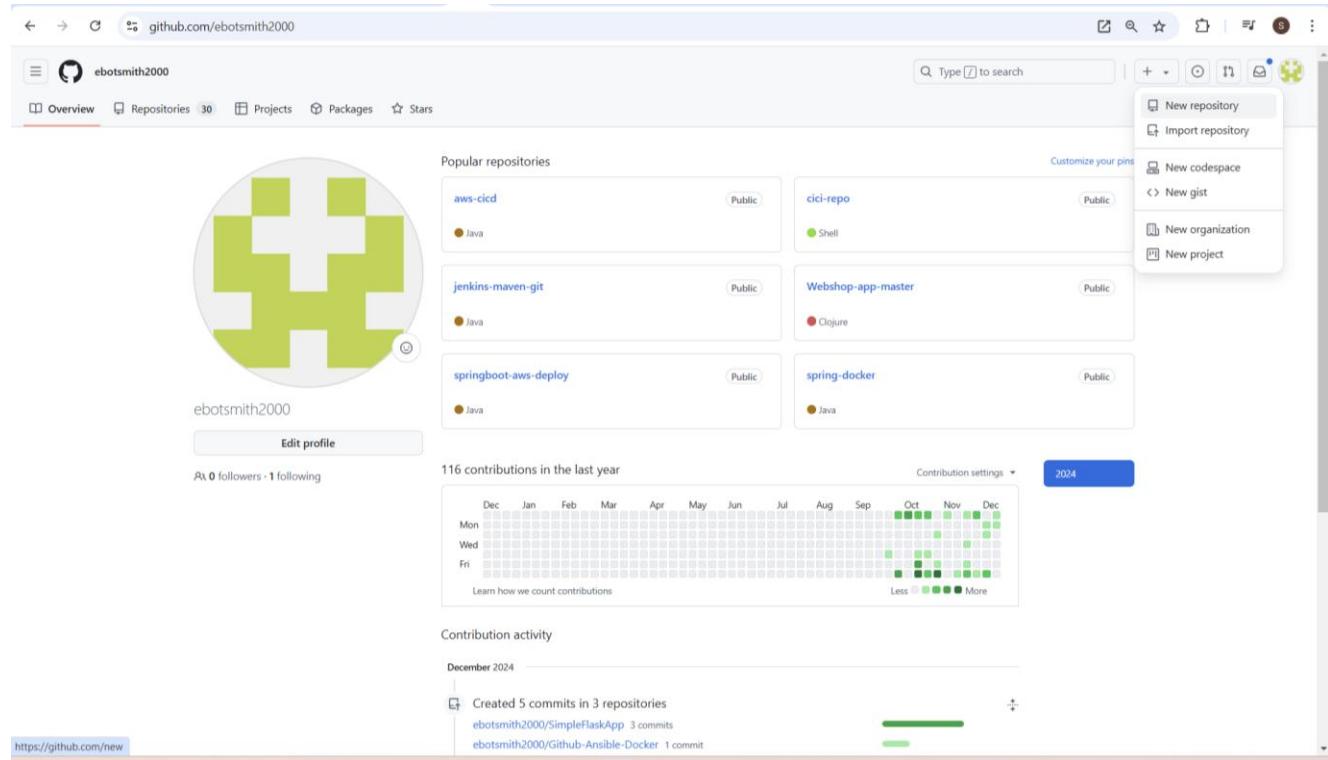
Step 1: Create a GitHub repository and upload the project files to it

Now, we have to create a repository on GitHub and upload our project files from our laptop to the GitHub repository.



A screenshot of a GitHub user profile page for 'ebotsmith2000'. The profile picture is a green and white pixelated logo. The top navigation bar shows 'Overview' (highlighted in red), 'Repositories 30', 'Projects', 'Packages', and 'Stars'. A search bar says 'Type / to search'. On the right, there's a '+ Add repository' button with an orange arrow pointing to it. Below the search bar is a 'Popular repositories' section with cards for 'aws-cicd', 'jenkins-maven-git', 'springboot-aws-deploy', 'cici-repo', 'Webshop-app-master', and 'spring-docker'. The 'Public' status is shown for each. To the right is a 'Customize your pins' section. Below that is a 'Contribution activity' section for December 2024, showing commits made to 'SimpleFlaskApp' and 'Github-Ansible-Docker'. A 'Contribution settings' dropdown is open.

We will start by creating the repository. Click on “+”



A screenshot of the same GitHub user profile page as above, but now the '+ Add repository' button is expanded, showing a dropdown menu with options: 'New repository', 'Import repository', 'New codespace', 'New gist', 'New organization', and 'New project'. The rest of the page content is identical to the first screenshot.

Select “**New Repository**”. Enter the name of the repository, we will call it “**jenkins-pipeline**”

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk ().*

Owner *



Repository name *

/ jenkins-pipeline

jenkins-pipeline is available.

Great repository names are short and memorable. Need inspiration? How about [fictional-winner](#) ?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

You are creating a public repository in your personal account.

Create repository

Click on “**Create Repository**”

The screenshot shows a GitHub repository page for 'jenkins-pipeline'. At the top, there are buttons for 'Pin', 'Unwatch 1', 'Fork 0', and 'Star 0'. Below the header, there are sections for 'Set up GitHub Copilot' and 'Add collaborators to this repository'. A 'Quick setup' section provides instructions for creating a new repository or pushing an existing one from the command line, including sample Git commands.

Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.
[Get started with GitHub Copilot](#)

Add collaborators to this repository
Search for people using their GitHub username or email address.
[Invite collaborators](#)

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/ebotsmith2000/jenkins-pipeline.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# jenkins-pipeline" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/ebotsmith2000/jenkins-pipeline.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/ebotsmith2000/jenkins-pipeline.git
git branch -M main
git push -u origin main
```

We have to navigate to the folder in our Local machine where we have our project files. The files are in this path: C:\DevOps-Projects\Jenkins\jenkins-pipeline

Open PowerShell

A screenshot of a Windows PowerShell window titled 'Windows PowerShell'. The window shows the standard PowerShell welcome message and a command prompt at PS C:\Users\ebots>.

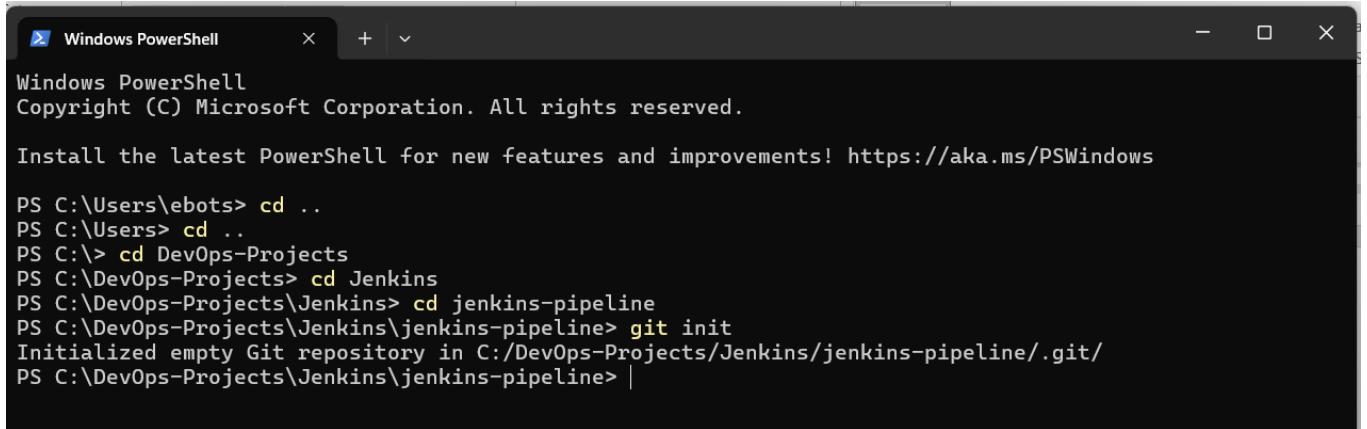
We now navigate to the folder

A screenshot of a Windows PowerShell window titled 'Windows PowerShell'. The window shows the user navigating through directory levels: starting at C:\Users\ebots, moving up to .., then to DevOps-Projects, then to Jenkins, and finally to jenkins-pipeline. The final prompt is PS C:\DevOps-Projects\Jenkins\jenkins-pipeline>.

We are now in the repository in our local machine where the project files are stored

Initialize the repository in the local machine by using the command:

```
git init
```



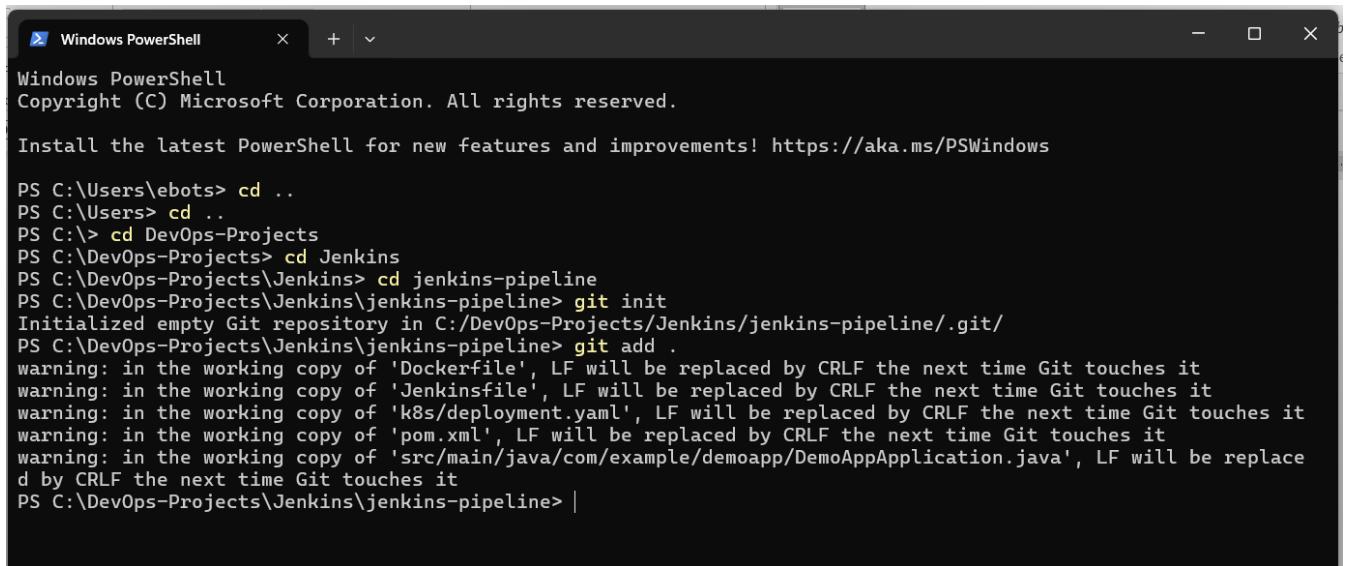
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ebots> cd ..
PS C:\Users> cd ..
PS C:\> cd DevOps-Projects
PS C:\DevOps-Projects> cd Jenkins
PS C:\DevOps-Projects\Jenkins> cd jenkins-pipeline
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git init
Initialized empty Git repository in C:/DevOps-Projects/Jenkins/jenkins-pipeline/.git/
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> |
```

We stage the files by using the command:

```
git add .
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ebots> cd ..
PS C:\Users> cd ..
PS C:\> cd DevOps-Projects
PS C:\DevOps-Projects> cd Jenkins
PS C:\DevOps-Projects\Jenkins> cd jenkins-pipeline
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git init
Initialized empty Git repository in C:/DevOps-Projects/Jenkins/jenkins-pipeline/.git/
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git add .
warning: in the working copy of 'Dockerfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Jenkinsfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'k8s/deployment.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'pom.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main/java/com/example/demoapp/DemoAppApplication.java', LF will be replaced by CRLF the next time Git touches it
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> |
```

We check the status of the files by using the command:

```
git status
```

```
Windows PowerShell x + v
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ebots> cd ..
PS C:\Users> cd ..
PS C:\> cd DevOps-Projects
PS C:\DevOps-Projects> cd Jenkins
PS C:\DevOps-Projects\Jenkins> cd jenkins-pipeline
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git init
Initialized empty Git repository in C:/DevOps-Projects/Jenkins/jenkins-pipeline/.git/
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git add .
warning: in the working copy of 'Dockerfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Jenkinsfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'k8s/deployment.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'pom.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main/java/com/example/demoapp/DemoAppApplication.java', LF will be replaced by CRLF the next time Git touches it
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file: Dockerfile
  new file: Jenkinsfile
  new file: k8s/deployment.yaml
  new file: pom.xml
  new file: src/main/java/com/example/demoapp/DemoAppApplication.java

PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> |
```

You can see the files that have been staged but not committed. Let us now commit the files by using the command:

```
git commit -m "Initial commit"
```

```
Windows PowerShell x + v
Initialized empty Git repository in C:/DevOps-Projects/Jenkins/jenkins-pipeline/.git/
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git add .
warning: in the working copy of 'Dockerfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Jenkinsfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'k8s/deployment.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'pom.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main/java/com/example/demoapp/DemoAppApplication.java', LF will be replaced by CRLF the next time Git touches it
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file: Dockerfile
  new file: Jenkinsfile
  new file: k8s/deployment.yaml
  new file: pom.xml
  new file: src/main/java/com/example/demoapp/DemoAppApplication.java

PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git commit -m "Initial commit"
[master (root-commit) eae78a2] Initial commit
 5 files changed, 122 insertions(+)
 create mode 100644 Dockerfile
 create mode 100644 Jenkinsfile
 create mode 100644 k8s/deployment.yaml
 create mode 100644 pom.xml
 create mode 100644 src/main/java/com/example/demoapp/DemoAppApplication.java
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> |
```

Let us add the files to our remote repository by using the command:

```
git remote add origin https://github.com/ebotsmith2000/jenkins-pipeline.git
```

```

Windows PowerShell
warning: in the working copy of 'Dockerfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Jenkinsfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'k8s/deployment.yaml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'pom.xml', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main/java/com/example/demoapp/DemoAppApplication.java', LF will be replaced by CRLF the next time Git touches it
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file: Dockerfile
  new file: Jenkinsfile
  new file: k8s/deployment.yaml
  new file: pom.xml
  new file: src/main/java/com/example/demoapp/DemoAppApplication.java

PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git commit -m "Initial commit"
[master (root-commit) eae78a2] Initial commit
  5 files changed, 122 insertions(+)
  create mode 100644 Dockerfile
  create mode 100644 Jenkinsfile
  create mode 100644 k8s/deployment.yaml
  create mode 100644 pom.xml
  create mode 100644 src/main/java/com/example/demoapp/DemoAppApplication.java
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git remote add origin https://github.com/ebotsmith2000/jenkins-pipeline.git
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline>

```

Let us push the files to our master branch in the GitHub repository by using the command:

```
git push -u origin master
```

```

Windows PowerShell
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file: Dockerfile
  new file: Jenkinsfile
  new file: k8s/deployment.yaml
  new file: pom.xml
  new file: src/main/java/com/example/demoapp/DemoAppApplication.java

PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git commit -m "Initial commit"
[master (root-commit) eae78a2] Initial commit
  5 files changed, 122 insertions(+)
  create mode 100644 Dockerfile
  create mode 100644 Jenkinsfile
  create mode 100644 k8s/deployment.yaml
  create mode 100644 pom.xml
  create mode 100644 src/main/java/com/example/demoapp/DemoAppApplication.java
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git remote add origin https://github.com/ebotsmith2000/jenkins-pipeline.git
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline> git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 16 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (14/14), 2.01 KiB | 228.00 KiB/s, done.
Total 14 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/ebotsmith2000/jenkins-pipeline.git
 * [new branch] master -> master
branch 'master' set up to track 'origin/master'.
PS C:\DevOps-Projects\Jenkins\jenkins-pipeline>

```

Refresh the GitHub page

The screenshot shows a GitHub repository named 'jenkins-pipeline'. The repository is public and contains several files: k8s, src/main/java/com/example/demoapp, Dockerfile, Jenkinsfile, and pom.xml. All files were committed by 'ebotsmith2000' 3 minutes ago. The README section is empty, and there are suggested workflows for publishing Java packages.

You can see that we have uploaded our project file from the local machine to the GitHub repository.

STEP 2: Create EC2 instance that will serve as Jenkins Server

We will create an AWS Ubuntu Instance. Go to EC2 Dashboard

The screenshot shows the AWS EC2 Instances dashboard. A list of instances is displayed, including:

- Jenkins Apache Server
- Linux Instance (Currently Stopping)
- Docker-Instance
- Sonarque-Instance
- Ansible Manager Node 1
- Ansible Manager Node 2
- Jenkins-Master
- Docker
- Jenkins
- Jenkins-Instance
- Jenkins Server
- Ansible Manager Node 3

Click on “Launch Instance”

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

e.g. My Web Server

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents

Quick Start



Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI

ami-0453ec754f44f9a4a (64-bit (x86), uefi-preferred) / ami-0ed83e7a78a23014e (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Let us give the instance the name “My Server”

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

My Server

Add additional tags

Scroll down to “Application and OS Images (Amazon Machine Images)” and select “ubuntu”

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents

Quick Start



Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

ami-0e2c8caa4b6378d8c (64-bit (x86)) / ami-0932ff346ea84d48 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible



Description

Ubuntu Server 24.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble image

Architecture

64-bit (x86)

AMI ID

ami-0e2c8caa4b6378d8c

Username

ubuntu

Verified provider

Scroll down to “Instance Type” and select “t2.large”

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.large

Family: t2 2 vCPU 8 GiB Memory Current generation: true

On-Demand Windows base pricing: 0.1208 USD per Hour

On-Demand RHEL base pricing: 0.1216 USD per Hour On-Demand SUSE base pricing: 0.1928 USD per Hour

On-Demand Ubuntu Pro base pricing: 0.0963 USD per Hour

On-Demand Linux base pricing: 0.0928 USD per Hour

All generations

[Compare instance types](#)

[Additional costs apply for AMIs with pre-installed software](#)

Scroll down to “Key Pair”

▼ Key pair (login) [Info](#)

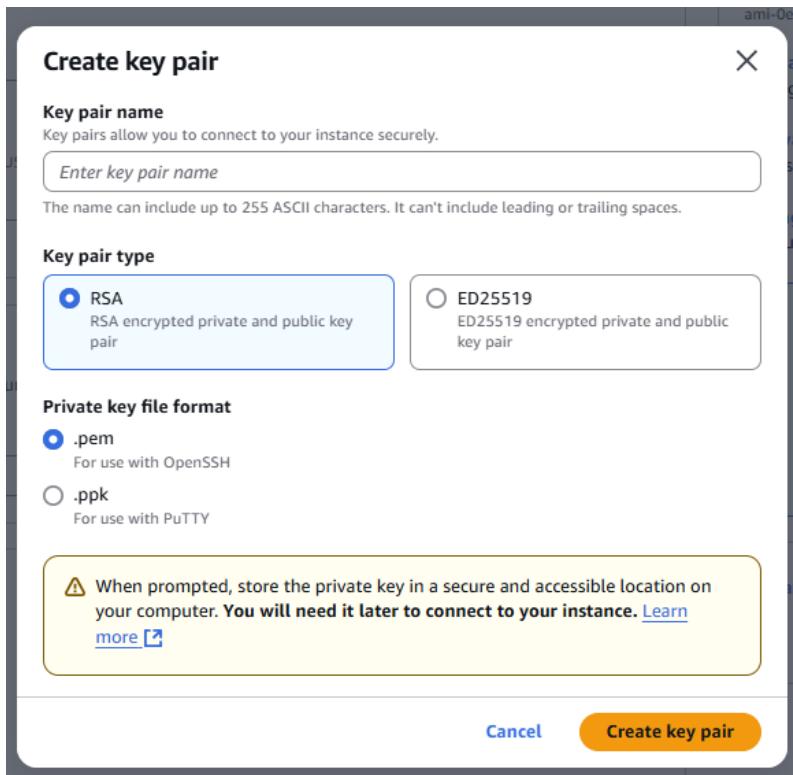
You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

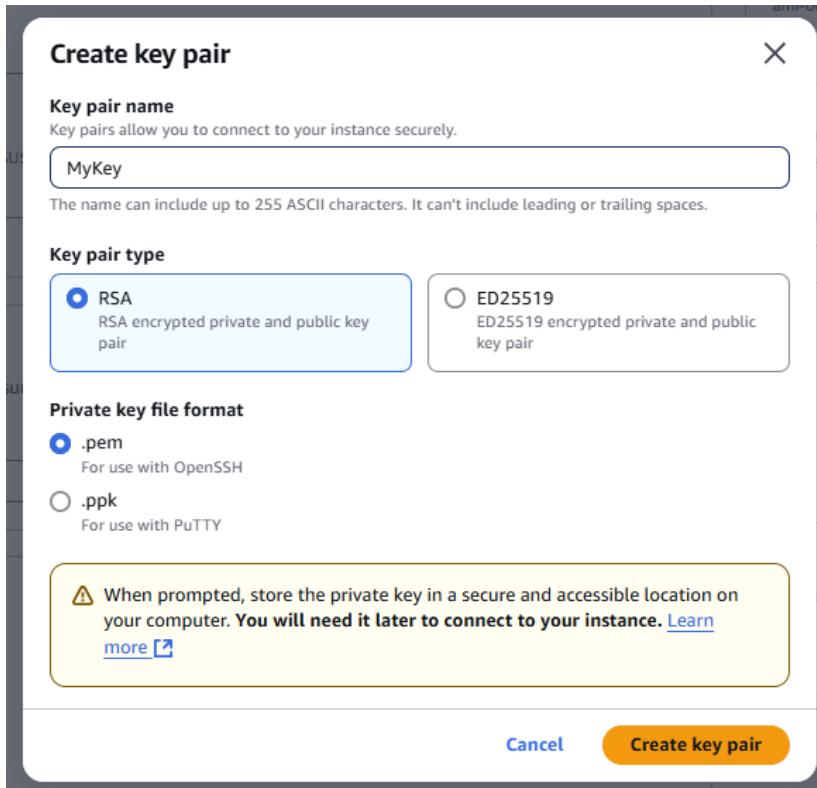
Select

[Create new key pair](#)

Click on “Create new key Pair”



I will use “MyKey” for the Key Pair Name”



Click on “Create Key Pair”

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

MyKey

 [Create new key pair](#)

Scroll down to “**Network Settings**”

▼ Network settings [Info](#)

[Edit](#)

Network | [Info](#)

vpc-07caa33c21359a185 | Default

Subnet | [Info](#)

No preference (Default subnet in any availability zone)

Auto-assign public IP | [Info](#)

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) | [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

[Create security group](#)

[Select existing security group](#)

We'll create a new security group called 'launch-wizard-54' with the following rules:

Allow SSH traffic from

Helps you connect to your instance

Anywhere

0.0.0.0/0

▼

Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server

Allow HTTP traffic from the internet

To set up an endpoint, for example when creating a web server

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

X

Scroll down to “**Configure Storage**” and make the size **30GiB**

▼ Configure storage [Info](#)

[Advanced](#)

1x

30

GiB

gp3

▼

Root volume 3000 IOPS (Not encrypted)

ⓘ Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

X

[Add new volume](#)

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

ⓘ Click refresh to view backup information

The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

C

0 x File systems

[Edit](#)

Click on “**Launch Instance**”

SUCCESS Successfully Initiated launch of instance (i-03a828fc255dc93a5)

▶ Launch log

Next Steps

Q What would you like to do next with this instance, for example "create alarm" or "create backup"

1 2 3 4

Create billing and free tier usage alerts To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds. Create billing alerts	Connect to your instance Once your instance is running, log into it from your local computer. Connect to instance Learn more	Connect an RDS database Configure the connection between an EC2 instance and a database to allow traffic flow between them. Connect an RDS database Create a new RDS database Learn more	Create EBS snapshot policy Create a policy that automates the creation, retention, and deletion of EBS snapshots. Create EBS snapshot policy Learn more	Manage detailed monitoring Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period. Manage detailed monitoring	Create Load Balancer Create a application, network gateway or classic Elastic Load Balancer. Create Load Balancer
Create AWS budget AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location. Create AWS budget	Manage CloudWatch alarms Create or update Amazon CloudWatch alarms for the instance. Manage CloudWatch alarms	Disaster recovery for your instances Recover the instances you just launched into a different Availability Zone or a different Region using AWS Elastic Disaster Recovery (DRS). Disaster recovery for your instances	Monitor for suspicious runtime activities Amazon GuardDuty enables you to continuously monitor for malicious runtime activity and unauthorized behavior, with near real-time visibility into on-host activities occurring across your Amazon EC2 workloads. Monitor for suspicious runtime activities	Get instance screenshot Capture a screenshot from the instance and view it as an image. This is useful for troubleshooting an unreachable instance. Get instance screenshot	Get system log View the instance's system log to troubleshoot issues. Get system log

View all instances

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Click on “View All Instances”

Instances (15) Info												
Find Instance by attribute or tag (case-sensitive) All states ▾ Last updated less than a minute ago Connect Instance state ▾ Actions ▾ Launch instances 1												
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP			
Jenkins-Slave	i-005f479e0b9eef67d	Stopped	t2.medium	-	View alarms +	us-east-1c	-	-	-			
ubuntu server	i-031dc15af78ea7d9f	Stopped	t2.micro	-	View alarms +	us-east-1c	-	-	-			
Jenkins Apache Server	i-0e2a96e685323145e	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1c	ec2-54-90-230-90.com...	54.90.230.90	-			
Linux Instance	i-01708070299757dbf	Stopped	t2.micro	-	View alarms +	us-east-1c	-	-	-			
My Server	i-03a828fc255dc93a5	Running	t2.large	2/2 checks passed	View alarms +	us-east-1c	ec2-54-172-236-64.co...	54.172.236.64	-			
Docker-Instance	i-0d1ca9ccaa920cd66	Stopped	t2.medium	-	View alarms +	us-east-1b	-	-	-			
Sonarqube-Instance	i-0f5aa01b266a57896	Stopped	t2.medium	-	View alarms +	us-east-1b	-	-	-			
Ansible Manager Node 1	i-0da2349a28f2cad66	Stopped	t2.micro	-	View alarms +	us-east-1b	-	-	-			
Ansible Manager Node 2	i-03c1dc52d19d09230	Stopped	t2.micro	-	View alarms +	us-east-1b	-	-	-			
Jenkins-Master	i-082196dded5d6d146	Stopped	t2.medium	-	View alarms +	us-east-1c	-	-	-			
Docker	i-048b155bf92e0917	Stopped	t2.medium	-	View alarms +	us-east-1c	-	-	-			
Jenkins	i-0b5feaae92ca531e6	Stopped	t2.medium	-	View alarms +	us-east-1c	-	-	-			
Jenkins-Instance	i-075f6173379758560	Stopped	t2.medium	-	View alarms +	us-east-1b	-	-	-			

Select an instance

STEP 3: Add Port 8080 in the inbound rule of the Instance for Jenkins

Click on the “Instance ID” of the instance we just created

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like Dashboard, EC2 Global View, Events, Instances, Images, Elastic Block Store, Network & Security, Load Balancing, Auto Scaling, and CloudShell/Feedback. The main area displays the 'Instance summary for i-03a828fc255dc93a5 (My Server)'. The 'Details' tab is active. Key details include:

- Public IPv4 address:** 54.172.236.64 | open address
- Instance state:** Running
- Private IP DNS name (IPv4 only):** ip-172-31-55.ec2.internal
- Instance type:** t2.large
- VPC ID:** vpc-07caa3c21359a185 (Default)
- Subnet ID:** subnet-0eacd54e97941a44a
- Instance ARN:** arn:aws:ec2:us-east-1:510786428272:instance/i-03a828fc255dc93a5

On the right, other tabs like Connect, Instance state, Actions, and a detailed view of the instance's configuration (AMI ID, Stop protection, Instance auto-recovery, etc.) are visible.

Click on “Security” tab

This screenshot is identical to the previous one, showing the AWS EC2 Instances page for instance i-03a828fc255dc93a5. The 'Security' tab is now active. The interface remains the same, displaying the instance summary and various configuration details. The right side of the screen shows the security-related configuration, including IAM Role, IMDSv2, Operator, and the Security tab itself.

Click on “Security Group”

sg-06628fbe687715248 - launch-wizard-24

Actions ▾

Details			
Security group name launch-wizard-24	Security group ID sg-06628fbe687715248	Description launch-wizard-24 created 2024-12-14T14:15:54.726Z	VPC ID vpc-07caa33c21359a185
Owner 510786428272	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

Inbound rules Outbound rules Sharing - new VPC associations - new Tags

Inbound rules (1)					
<input type="text"/> Search					
<input type="checkbox"/>	Name	▼	Security group rule ID	▼	IP version
<input type="checkbox"/>	-		sgr-0b90f69132aed42c2	IPv4	Type
				SSH	Protocol
				TCP	Port range
					22

Click on “Edit Inbound Rules”

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules <small>Info</small>					
Security group rule ID	Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>
sgr-0b90f69132aed42c2	<input type="button" value="SSH"/>	<input type="button" value="TCP"/>	<input type="button" value="22"/>	<input type="button" value="Custom"/>	<input type="text"/> 0.0.0.0/ <input type="button" value="X"/>
<input type="button" value="Add rule"/>					

Add rule Cancel Preview changes Save rules

Click on “Add Rule”

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules <small>Info</small>					
Security group rule ID	Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>
sgr-0b90f69132aed42c2	<input type="button" value="SSH"/>	<input type="button" value="TCP"/>	<input type="button" value="22"/>	<input type="button" value="Custom"/>	<input type="text"/> 0.0.0.0/ <input type="button" value="X"/>
-	<input type="button" value="Custom TCP"/>	<input type="button" value="TCP"/>	<input type="button" value="0"/>	<input type="button" value="Custom"/>	<input type="text"/> <input type="button" value="X"/>
<input type="button" value="Add rule"/>					

Add rule Cancel Preview changes Save rules

Add Port 8080

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0b90f69132aed42c2	SSH	TCP	22	Custom	Q, 0.0.0.0/0
-	Custom TCP	TCP	8080	Anyw...	Jenkins, 0.0.0.0/0

Add rule Delete

Cancel Save rules

Click on “Save Rules”

Inbound security group rules successfully modified on security group (sg-06628fbe687715248 | launch-wizard-24)

Details

Security group name launch-wizard-24	Security group ID sg-06628fbe687715248	Description launch-wizard-24 created 2024-12-14T14:15:54.726Z	VPC ID vpc-07caa33c21359a185
Owner 510786428272	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules Outbound rules Sharing - new VPC associations - new Tags

Inbound rules (2)

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-0b90f69132aed42c2	IPv4	SSH	TCP	22
-	sgr-056cc839e43c75dea	IPv4	Custom TCP	TCP	8080

Port 8080 has been added for Jenkins

STEP 4: Connect to the server

You can see the instance we just created. Select the instance

Instances (1/4) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
My Server	i-0ce9d13dd85b2daec	Running	t2.large	2/2 checks passed	View alarms +	us-east-1b	ec2-3-87-67-132.cc
Docker-Instance	i-0d1ca9ccaa920cd66	Stopped	t2.medium	-	View alarms +	us-east-1b	-
Sonarqube-Instance	i-0f5aa01b266a57896	Stopped	t2.medium	-	View alarms +	us-east-1b	-
Jenkins-Instance	i-07576173379758560	Stopped	t2.medium	-	View alarms +	us-east-1b	-

and click on “Connect”

The screenshot shows the 'Connect to instance' page for an EC2 instance. The 'SSH client' tab is selected. The instance ID is listed as 'i-03a828fc255dc93a5 (My Server)'. Below it, there's a numbered list of steps to connect via SSH:

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is MyKey.pem.
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "MyKey.pem"
4. Connect to your instance using its Public DNS:
ec2-54-172-236-64.compute-1.amazonaws.com

Below the steps, there's an example command:

```
ssh -i "MyKey.pem" ubuntu@ec2-54-172-236-64.compute-1.amazonaws.com
```

A note at the bottom states: "Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username."

Cancel

Select the “EC2 Instance Connect” tab

The screenshot shows the 'Connect to instance' page for an EC2 instance. The 'EC2 Instance Connect' tab is selected. The instance ID is listed as 'i-03a828fc255dc93a5 (My Server)'. Under 'Connection Type', the 'Public IPv4 address' option is selected, showing '54.172.236.64'. There are also options for 'Connect using EC2 Instance Connect' and 'IPv6 address'. The 'Username' field is set to 'ubuntu'. A note at the bottom states: "Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username."

Cancel

Connect

Click on “Connect”

```
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1021-aws x86_64)
```

```
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
```

```
System information as of Sat Jan 18 15:19:03 UTC 2025
```

System load: 0.0	Processes: 120
Usage of /: 6.0% of 28.02GB	Users logged in: 0
Memory usage: 2%	IPv4 address for enX0: 172.31.44.137
Swap usage: 0%	

```
Expanded Security Maintenance for Applications is not enabled.
```

```
0 updates can be applied immediately.
```

```
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
```

```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.
```

```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

```
ubuntu@ip-172-31-44-137:~$
```

i-0201fc151bea82ac9 (My Server)

Public IPs: 54.198.237.25 Private IPs: 172.31.44.137

You can see that we are connected to the server.

STEP 5: Install Java JDK 17

We will run these commands:

```
sudo apt update  
sudo apt install openjdk-17-jre
```

```
Get:36 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [114 kB]
Get:37 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [7216 B]
Get:38 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [800 kB]
Get:39 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [171 kB]
Get:40 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.0 kB]
Get:41 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [13.5 kB]
Get:42 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:43 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [12.4 kB]
Get:44 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Get:45 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Get:46 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [356 B]
Fetched 30.5 MB in 5s (682 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done

The following additional packages will be installed:
adwaita-icon-themes alsa-alsaconf at-spi2-common at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-service fontconfig fontconfig-config fonts-dejavu-core
fontconfig-extra fonts-dejavu-mono gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme humanity-icon-themes java-common libasound2-dbg libatk-bridge2.0-0*64
libatk-wrapper-gtk libatk-wrapper-gtk2 libatk2.0-0*64 libatk2.0-0*64 libatk2.0-0*64 libatk-wai-client3 libbahvi-common3 libcairo-ro libcupsp2t64 libdatatrel libdbconf1
libdeflate0 libdrm-amdgpu libdrm-intel libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libgail-common libgail18t64 libgd-pixbuf2-0.0-hin libgd-pixbuf2.0-common libgf7 libgl1
libgl1-amber2 libgl1-mesa libglapi-mesa libglvnd0 libglx-mesa libglx0 libglrapi2-3 libgtk2.0-0*64 libgtk2.0-bin libgtk2.0-common libharfbuzz0b libice0 libjbig0 libjpeg-turbo8 libjpegs8
liblcms2-2 liblirc4 liblwm17e64 libpango-1.0-0 libpangocairo-1.0-0 libpangofc2-1.0-0 libpicaace0 libpcsc-lite1 libpixman-1.0 librsvg2-2 librsvg2-common libsharpay0 libsm6 libthai-data libthai0
libxcb-xfixes0 libxcompositelib libxcursor1 libxdamage1 libxfixes3 libxt2 libxinerama1 libxkbfile1 libxmu libxpms4 libxrandr2 libxrender1 libxshmfence1 libxt64 libxtst6 libxv1 libxf86dg1
libxf86vml mesa-vulkan-drivers openjdk-17-jre-headless session-migration ubuntu-mono x11-common x11-utils
Suggested packages:
default-jre alsamixer libasound2-plugins cups-common gvfs liblcms2-utils pscod librsvg2-bin libsass-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic
mesa-utils
Recommended packages:
libxt
The following NEW packages will be installed:
adwaita-icon-themes alsa-alsaconf at-spi2-common at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-service fontconfig fontconfig-config fonts-dejavu-core
fonts-dejavu-extras fonts-dejavu-mono gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme humanity-icon-themes java-common libasound2-dbg libatk-bridge2.0-0*64
libatk-wrapper-gtk libatk-wrapper-gtk2 libatk2.0-0*64 libatk2.0-0*64 libatk2.0-0*64 libatk-wai-client3 libbahvi-common3 libcairo-ro libcupsp2t64 libdatatrel libdbconf1
libdeflate0 libdrm-amdgpu libdrm-intel libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libgail-common libgail18t64 libgd-pixbuf2-0.0-hin libgd-pixbuf2.0-common libgf7 libgl1
libgl1-amber2 libgl1-mesa libglapi-mesa libglvnd0 libglx-mesa libglx0 libglrapi2-3 libgtk2.0-0*64 libgtk2.0-bin libgtk2.0-common libharfbuzz0b libice0 libjbig0 libjpeg-turbo8 libjpegs8
liblcms2-2 liblirc4 liblwm17e64 libpango-1.0-0 libpangocairo-1.0-0 libpangofc2-1.0-0 libpicaace0 libpcsc-lite1 libpixman-1.0 librsvg2-2 librsvg2-common libsharpay0 libsm6 libthai-data libthai0
libtiff6 libxf86vml mesa-vulkan-drivers openjdk-17-jre-headless session-migration ubuntu-mono x11-common x11-utils
libxt
libxtf
libxf86vml mesa-vulkan-drivers openjdk-17-jre-headless session-migration ubuntu-mono x11-common x11-utils
0 upgraded, 117 newly installed, 0 to remove and 1 not upgraded.
Need to get 515 MB of new packages.
After this operation, 508 MB of additional disk space will be used.
Do you want to continue? [Y/n] 
```

Type "Y" and press ENTER

```
Adding debian:ePKI_Root_Certification_Authority.pem
Adding debian:emSign_ECC_Root_CA_-_C3.pem
Adding debian:emSign_ECC_Root_CA_-_G3.pem
Adding debian:emSign_Root_CA_-_C1.pem
Adding debian:emSign_Root_CA_-_G1.pem
Adding debian:vTrus_ECC_Root_CA.pem
Adding debian:vTrus_Root_CA.pem
done.

Setting up openjdk-17-jre:amd64 (17.0.13+11-2ubuntu1~24.04) ...
Processing triggers for libc-bin (2.39-0ubuntu8.3) ...
Processing triggers for libgdk-pixbuf2.0-0:amd64 (2.42.10+dfsg-3ubuntu3.1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-44-137:~$ █
```

Verify Java is Installed by running the command:

```
java -version
```

```
ubuntu@ip-172-31-44-137:~$ java -version
openjdk version "17.0.13" 2024-10-15
OpenJDK Runtime Environment (build 17.0.13+11-Ubuntu-2ubuntu124.04)
OpenJDK 64-Bit Server VM (build 17.0.13+11-Ubuntu-2ubuntu124.04, mixed mode, sharing)
ubuntu@ip-172-31-44-137:~$ █
```

STEP 6: Install Jenkins

Now, let us proceed to install Jenkins, Run the command:

```
curl -fsSL https://pkg.jenkins.io/debian/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 binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

```
ubuntu@ip-172-31-44-137:~$ curl -fsSL https://pkg.jenkins.io/debian/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 binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Ign:2 https://pkg.jenkins.io/debian binary/ InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:5 https://pkg.jenkins.io/debian binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian binary/ Release.gpg [833 B]
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:8 https://pkg.jenkins.io/debian binary/ Packages [67.1 kB]
Fetched 70.0 kB in 0s (170 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
 net-tools
The following NEW packages will be installed:
 jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 1 not upgraded.
Need to get 94.1 MB of archives.
After this operation, 96.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] █
```

Type “Y” and press ENTER

```

Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Ign:2 https://pkg.jenkins.io/debian binary/ InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:5 https://pkg.jenkins.io/debian binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian binary/ Release.gpg [833 B]
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:8 https://pkg.jenkins.io/debian binary/ Packages [67.1 kB]
Fetched 70.0 kB in 0s (170 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  net-tools
The following NEW packages will be installed:
  jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 1 not upgraded.
Need to get 94.1 MB of archives.
After this operation, 96.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 net-tools amd64 2.10-0.1ubuntu4 [204 kB]
Get:2 https://pkg.jenkins.io/debian binary/ jenkins 2.493 [93.9 MB]
Fetched 94.1 MB in 1min 5s (1441 kB/s)
Selecting previously unselected package net-tools.
(Reading database ... 85305 files and directories currently installed.)
Preparing to unpack .../net-tools_2.10-0.1ubuntu4_amd64.deb ...
Unpacking net-tools (2.10-0.1ubuntu4) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../archives/jenkins_2.493_all.deb ...
Unpacking jenkins (2.493) ...
Setting up net-tools (2.10-0.1ubuntu4) ...
Setting up jenkins (2.493) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-44-137:~$ █

```

i-0201fc151bea82ac9 (My Server)

PublicIPs: 54.198.237.25 PrivateIPs: 172.31.44.137

CloudShell [Feedback](#)

Let us verify if Jenkins is installed or not by using the command:

`sudo systemctl status jenkins`

```

ubuntu@ip-172-31-44-137:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-01-18 15:26:38 UTC; 59s ago
       Main PID: 4420 (java)
          Tasks: 49 (limit: 9507)
        Memory: 969.9M (peak: 981.8M)
           CPU: 16.952s
          CGroup: /system.slice/jenkins.service
                  └─4420 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Jan 18 15:26:34 ip-172-31-44-137 jenkins[4420]: b1bf5ae1d744108a2e1aeb76ben08a
Jan 18 15:26:34 ip-172-31-44-137 jenkins[4420]: Jenkins Continuous Integration Server
Jan 18 15:26:34 ip-172-31-44-137 jenkins[4420]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Jan 18 15:26:34 ip-172-31-44-137 jenkins[4420]: ****
Jan 18 15:26:38 ip-172-31-44-137 jenkins[4420]: 2025-01-18 15:26:38.245+0000 [id=31]      INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
Jan 18 15:26:38 ip-172-31-44-137 jenkins[4420]: 2025-01-18 15:26:38.273+0000 [id=23]      INFO    hudson.lifecycle.Lifecycle$OnReady: Jenkins is fully up and running
Jan 18 15:26:38 ip-172-31-44-137 systemd[1]: Started jenkins.service Jenkins Continuous Integration Server.
Jan 18 15:26:38 ip-172-31-44-137 jenkins[4420]: 2025-01-18 15:26:38.643+0000 [id=48]      INFO    h.a.DownloadService$Downloadable$load: Obtained the updated data file for hudson.tasks.Maven.MavenInfo
Jan 18 15:26:38 ip-172-31-44-137 jenkins[4420]: 2025-01-18 15:26:38.644+0000 [id=48]      INFO    hudson.util.Retrier$Start: Performed the action check updates server successfully at the attempt #1
lines 1-20/20 (END)

```

You can see that Jenkins is active and running. So, we have successfully installed Jenkins.

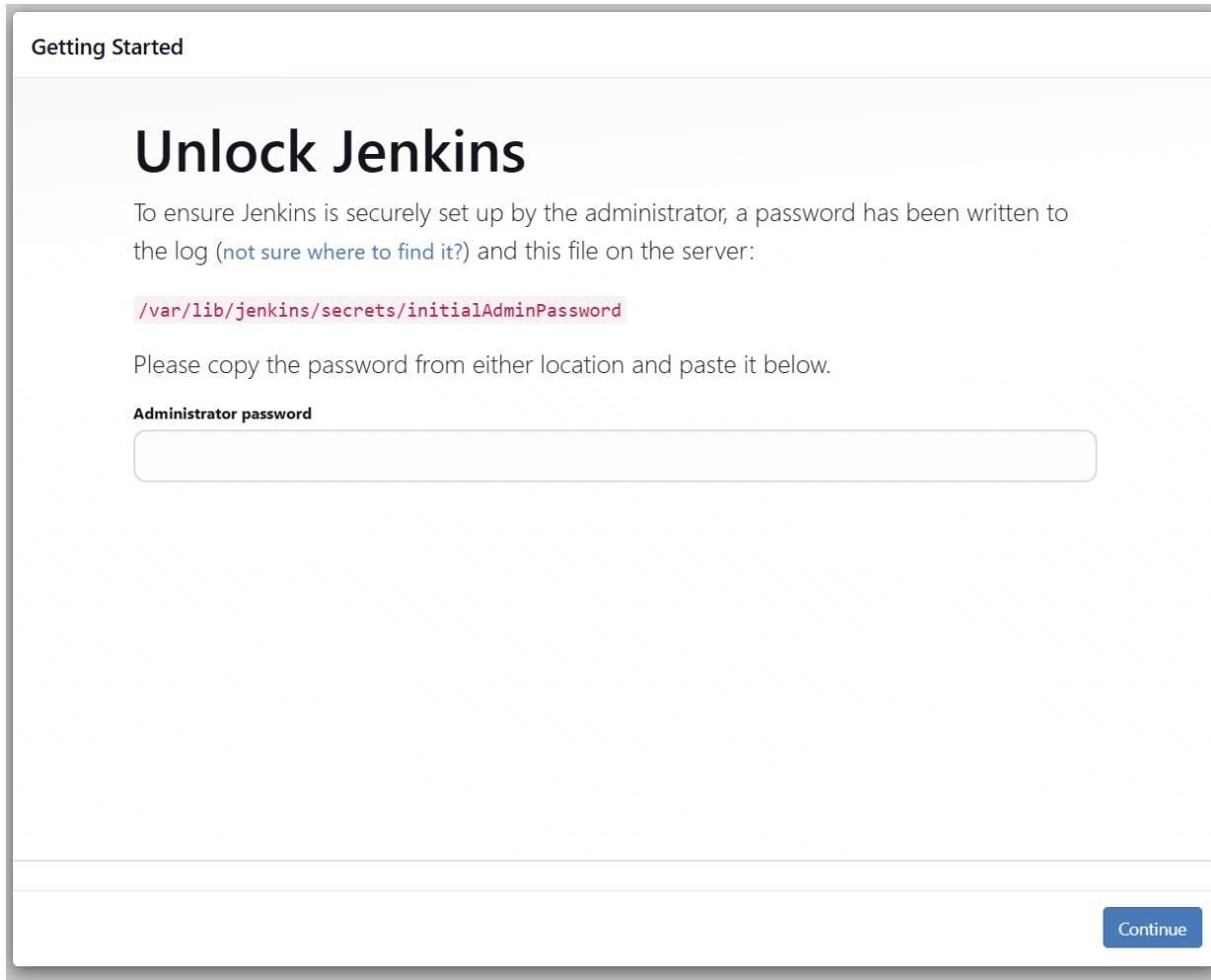
Return to the command mode by typing **CTRL+C**

STEP 7: Access Jenkins GUI on Browser

Now, let us access Jenkins GUI by using this one the browser:

http://<Public IPv4 Address>:8080

That is **http://54.198.237.25:8080**



To get the password you have to run the command:

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

```
ubuntu@ip-172-31-31-55:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
fda0aaa5049a49e6aa453c6645e3c2e8
ubuntu@ip-172-31-31-55:~$ █
```

The password is **fda0aaa5049a49e6aa453c6645e3c2e8**

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

.....

Continue

Click on “Continue”

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.

Click on “Install Suggested Plugins”

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.489

Skip and continue as admin

Save and Continue

Enter the following:

Username: **admin**

Password: **admin**

Full name: **Sidney Ebot**

and Email address: **ebotsmith@gmail.com**

Getting Started

Create First Admin User

Username

admin

Password

.....

Confirm password

.....

Full name

Sidney Ebot

E-mail address

ebotsmith@gmail.com

Jenkins 2.489

Skip and continue as admin

Save and Continue

Click on “Save and Continue”

Getting Started

Instance Configuration

Jenkins URL:

http://54.172.236.64:8080/

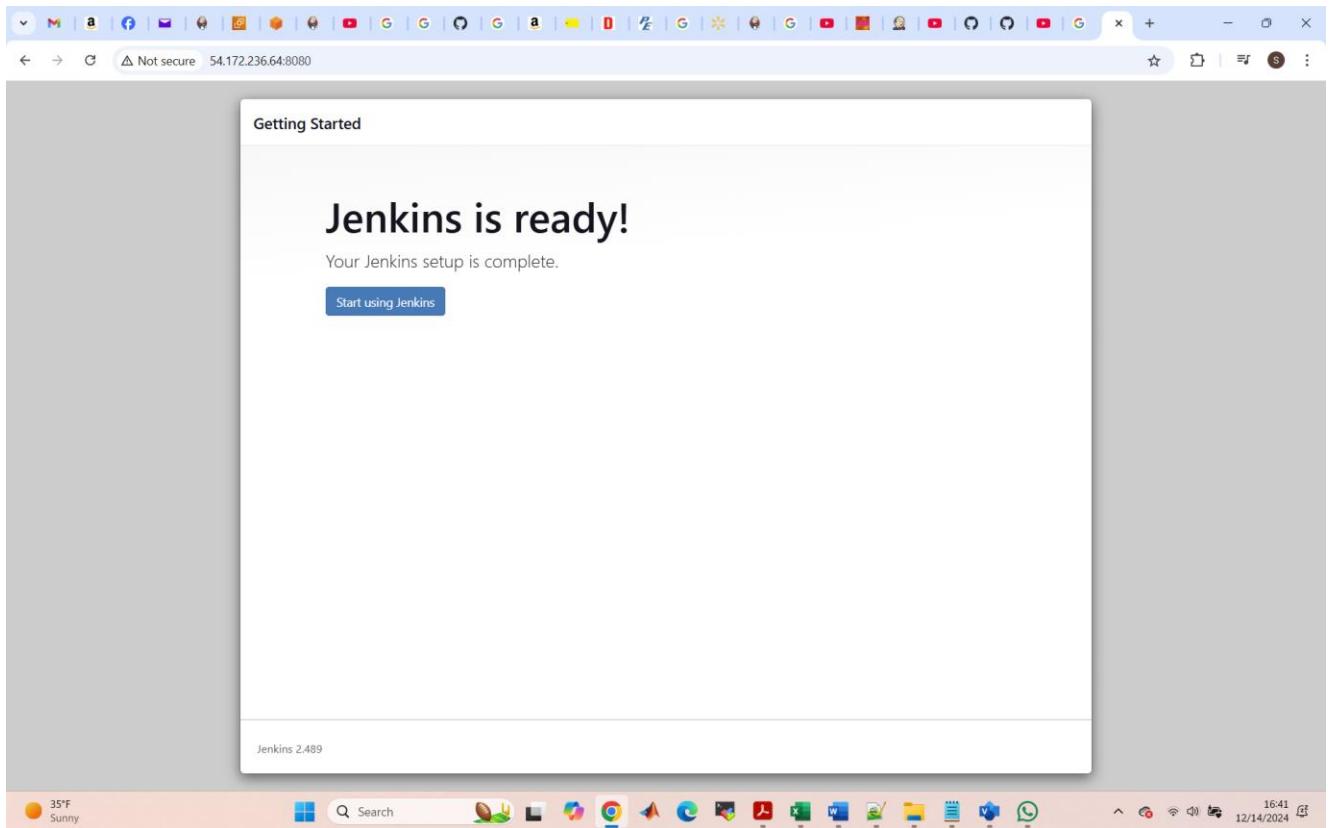
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Not now

Save and Finish

Then click on “**Save and Finish**”



Then start Jenkins by clicking on “**Start using Jenkins**”

The screenshot shows the Jenkins dashboard at the URL <http://54.172.236.64:8080>. The top navigation bar includes links for 'Dashboard', 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. On the left, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0/2). The main content area features a 'Welcome to Jenkins!' message, a 'Start building your software project' section with a 'Create a job' button and a '+' icon, and a 'Set up a distributed build' section with options for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. The bottom right corner shows 'REST API' and 'Jenkins 2.489'.

Jenkins is ready.

STEP 8: Create and Configure Pipeline on Jenkins

Now for our code compile we need maven. On the Jenkins GUI

The screenshot shows the Jenkins dashboard again, identical to the one above. A blue arrow points from the text 'Click on "Create a Job" or "New Item"' to the 'New Item' link in the left sidebar. A red arrow points from the same text to the '+ Create a job' button in the 'Start building your software project' section.

Click on “Create a Job” or “New Item”

New Item

Enter an item name

» This field cannot be empty, please enter a valid name

Select an item type

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.

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.

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.

Multibranch Pipeline

OK

We will give the item the name “**demo**” and select “**Pipeline**”

New Item

Enter an item name

demo

Select an item type

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.

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.

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.

Multibranch Pipeline Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

Click on “OK”

The screenshot shows the Jenkins configuration page for a job named "demo". The left sidebar has tabs for "General", "Build Triggers", "Advanced Project Options", and "Pipeline". The "General" tab is selected. The main area shows a "Description" text input field and a list of build options under "Plain text Preview". The "Enabled" switch is turned on. At the bottom are "Save" and "Apply" buttons.

Click on “Pipeline” on the left-hand side

The screenshot shows the Jenkins configuration page for a job named "demo". The left sidebar has tabs for "General", "Build Triggers", "Advanced Project Options", and "Pipeline". The "Pipeline" tab is selected. The main area shows a "Definition" section with a "Pipeline script" input field containing "Pipeline script". Below it is a "Script" code editor with a single line of Groovy code: "1 try sample Pipeline...". A blue arrow points from the "Pipeline" tab in the sidebar to the "Pipeline" section in the main area. Another orange arrow points from the "try sample Pipeline..." dropdown menu in the "Definition" section to the "HelloWorld" option.

On “Definition”, select “Pipeline Script” and on “Try Sample Pipeline”, select “HelloWorld”

The screenshot shows the Jenkins Pipeline configuration page for a project named 'demo'. The 'Pipeline' tab is selected in the sidebar. The main area displays a Groovy script for defining a pipeline:

```
1 pipeline {  
2     agent any  
3     stages {  
4         stage('Hello') {  
5             steps {  
6                 echo 'Hello World'  
7             }  
8         }  
9     }  
10 }  
11  
12 }
```

Below the script, there is a dropdown menu set to 'Hello World'. A checkbox labeled 'Use Groovy Sandbox' is checked. At the bottom are 'Save' and 'Apply' buttons.

Right-click on “Pipeline Syntax”

The screenshot shows the same Jenkins Pipeline configuration page as above, but with a context menu open over the 'Pipeline Syntax' link at the bottom of the script editor. The menu options include:

- Open link in new tab
- Open link in new window
- Open link in incognito window
- Save link as...
- Copy link address
- AdBlock — block ads across the web >
- Get image descriptions from Google >
- Inspect

Select “Open Link in new Tab”

This screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. On the left, there's a sidebar with links like 'Snippet Generator', 'Declarative Directive Generator', 'Declarative Online Documentation', 'Steps Reference', 'Global Variables Reference', 'Online Documentation', 'Examples Reference', and 'IntelliJ IDEA GDSL'. The main area has a title 'Overview' and a description of the snippet generator. Below that, under 'Steps', a 'Sample Step' is selected: 'archiveArtifacts: Archive the artifacts'. The configuration panel shows 'archiveArtifacts' with a dropdown for 'Files to archive' and an 'Advanced' button. At the bottom is a 'Generate Pipeline Script' button and a large text area for the generated script.

Under “**Sample Step**” in “**Steps**”, select “**git:Git**”. On “**Repository URL**” copy the **HTTP URL** of your **GitHub repository**.

This screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface with the 'git:Git' step selected. The configuration panel shows 'git' with a dropdown for 'Repository URL' containing 'https://github.com/ebotsmith2000/CICD-Pipeline.git', 'Branch' set to 'master', and 'Credentials' set to '- none -'. There are also checkboxes for 'Include in polling?' and 'Include in changelog?'. The 'Generate Pipeline Script' button is visible at the bottom.

Scroll down to “**Generate Pipeline Script**”

Click on “Generate Pipeline Script”

The screenshot shows a web-based interface for managing a CI/CD pipeline. The URL in the address bar is `54.172.236.64:8080/job/demo/pipeline-syntax/`. The page title is "Pipeline Syntax". The main form has the following fields:

- Branch**: A dropdown menu set to "master".
- Credentials**: A dropdown menu set to "- none -".
- + Add**: A button to add new credentials.
- Include in polling?**: A checked checkbox.
- Include in changelog?**: A checked checkbox.

At the bottom left is a blue button labeled "Generate Pipeline Script". Below it is a code editor containing the following Groovy script:

```
git 'https://github.com/ebotsmith2000/CICD-Pipeline.git'
```

Copy the generated code and paste in the line for “step” in “Script” under “Pipeline”

Dashboard > demo > Configuration

Advanced ▾

Configure

Pipeline

Definition

Pipeline script

```

1 pipeline {
2     agent any
3
4     stages {
5         stage('Git Checkout') {
6             steps {
7                 git 'https://github.com/ebotsmith2000/CICD-Pipeline.git'
8             }
9         }
10    }
11 }
12

```

Hello World

Use Groovy Sandbox ?

Pipeline Syntax

Save **Apply**

Click on “Save”

← → ⌂ Not secure 54.172.236.64:8080/job/demo/

Jenkins

Dashboard > demo >

Status demo Add description

</> Changes

Build Now

Configure

Delete Pipeline

Stages

Rename

Pipeline Syntax

Builds No builds

REST API Jenkins 2.489

Click on “Build Now”

The screenshot shows a Jenkins pipeline console output for a build named '#1'. The output is as follows:

```
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/demo
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Git Checkout)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/ebotsmith2000/CICD-Pipeline.git
> git init /var/lib/jenkins/workspace/demo # timeout=10
Fetching upstream changes from https://github.com/ebotsmith2000/CICD-Pipeline.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/ebotsmith2000/CICD-Pipeline.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/ebotsmith2000/CICD-Pipeline.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 641c2b201f0e1efdf1d15a5fc3a9ae6744b654c (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 641c2b201f0e1efdf1d15a5fc3a9ae6744b654c # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git checkout -b master 641c2b201f0e1efdf1d15a5fc3a9ae6744b654c # timeout=10
Commit message: "Initial commit"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

You can see that the build is successful

STEP 9: Install Pipeline Stage View Plugin

Now, click on “Dashboard”

The screenshot shows the Jenkins dashboard at the URL 54.172.236.64:8080. The top navigation bar includes links for 'Not secure', 'Relaunch to update', and user 'Sidney Ebott'. The main dashboard features a 'Build History' section with a 'demo' job listed, showing its last success at 37 min ago. Below it are sections for 'Build Queue' (empty) and 'Build Executor Status' (0/2). On the left, there's a sidebar with 'New Item', 'Manage Jenkins' (which is currently selected), and 'My Views'. A central summary table provides quick stats: 1 success, 1 warning, 1 demo job, last success at 37 min ago, last failure N/A, and last duration 6.5 sec.

Click on “Manage Jenkins”

The screenshot shows the 'Manage Jenkins' page at the URL 54.172.236.64:8080/manage/. The top navigation bar includes links for 'Not secure', 'Relaunch to update', and user 'Sidney Ebott'. The main content area is titled 'Manage Jenkins' with a sub-section 'System Configuration'. It features several configuration items: 'System' (Configure global settings and paths), 'Tools' (Configure tools, their locations and automatic installers), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand), 'Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), 'Appearance' (Configure the look and feel of Jenkins), 'Security' (Secure Jenkins; define who is allowed to access/use the system), 'Credentials' (Configure credentials), and 'Credential Providers' (Configure the credential providers and types). A search bar at the top right allows users to search for specific settings.

Click on “Plugins”

The screenshot shows the Jenkins Plugin Manager interface. At the top, there's a navigation bar with links for Dashboard, Manage Jenkins, and Plugins. Below this is a sidebar with options: Updates (selected), Available plugins (highlighted in grey), Installed plugins, Advanced settings, and Download progress. The main content area has a search bar at the top right. In the center, there's a message: "No updates available" with a shopping cart icon. A note below says: "Disabled rows are already upgraded, awaiting restart. Shaded but selectable rows are in progress or failed." There are also small icons for a refresh and a help button.

Click on “Available Plugins”

This screenshot shows the Jenkins Plugin Manager after clicking on "Available plugins". The sidebar now shows "Available plugins" as the selected tab. The main content area displays a list of available plugins. Each plugin entry includes a checkbox, the plugin name, its version, a brief description, and the date it was released. A search bar is at the top right, and an "Install" button is visible. The list includes:

Install	Name	Released
<input type="checkbox"/>	JavaMail API 1.6.2-10 Library plugins (for use by other plugins)	6 mo 24 days ago
<input type="checkbox"/>	Command Agent Launcher 116.vd85919c54a_d6 Agent Management	1 mo 2 days ago
<input type="checkbox"/>	Oracle Java SE Development Kit Installer 80.v8a_dee33ed6f0 Allows the Oracle Java SE Development Kit (JDK) to be installed via download from Oracle's website.	4 mo 10 days ago
<input type="checkbox"/>	Pipeline: REST API 2.34 User Interface	1 yr 1 mo ago
<input type="checkbox"/>	Pipeline: Stage View 2.34 User Interface	1 yr 1 mo ago
<input type="checkbox"/>	JSch dependency 0.2.16-86.v42e010d9484b_... Library plugins (for use by other plugins) Miscellaneous	11 mo ago

Search for “View”

The screenshot shows the Jenkins plugin manager interface. On the left, a sidebar has 'Available plugins' selected. In the main area, the 'Pipeline: Stage View' plugin is listed with its checkbox checked. The plugin details show it's a 'User Interface' plugin released 1 year 1 month ago. A note indicates it's up for adoption. Other plugins like 'Favorite' and 'Job DSL' are also listed.

Install	Name	Released
<input type="checkbox"/>	Pipeline: Stage View 2.34 User Interface	1 yr 1 mo ago
<input type="checkbox"/>	Favorite 2.221.v19ca_666b_62f5 List view columns User Interface Authentication and User Management Miscellaneous	4 mo 9 days ago
<input type="checkbox"/>	Job DSL 1.90 Build Tools	1 mo 15 days ago
<input type="checkbox"/>	Dashboard View 2.521.v339b_a_f4d8da_8 User Interface	1 mo 3 days ago
<input type="checkbox"/>	built-on-column 1.4 List view columns	1 yr 8 mo ago

Select “Pipeline Stage View”

This screenshot is identical to the one above, showing the Jenkins plugin manager with the 'Pipeline: Stage View' plugin selected for installation. The plugin details and other available plugins are the same.

Install	Name	Released
<input checked="" type="checkbox"/>	Pipeline: Stage View 2.34 User Interface	1 yr 1 mo ago
<input type="checkbox"/>	Favorite 2.221.v19ca_666b_62f5 List view columns User Interface Authentication and User Management Miscellaneous	4 mo 9 days ago
<input type="checkbox"/>	Job DSL 1.90 Build Tools	1 mo 15 days ago
<input type="checkbox"/>	Dashboard View 2.521.v339b_a_f4d8da_8 User Interface	1 mo 3 days ago
<input type="checkbox"/>	built-on-column 1.4 List view columns	1 yr 8 mo ago

and click on “Install”

The screenshot shows the Jenkins plugin manager interface. The left sidebar has links for Updates, Available plugins, Installed plugins, Advanced settings, and Download progress (which is currently selected). The main area lists various Jenkins components with green checkmarks indicating success. A note at the bottom says "you can start using the installed plugins right away".

Component	Status
GITHUB Branch Source	Success
Pipeline: GitHub Groovy Libraries	Success
Pipeline Graph Analysis	Success
Metrics	Success
Pipeline Graph View	Success
Git	Success
EDDSA API	Success
Trilead API	Success
SSH Build Agents	Success
Matrix Authorization Strategy	Success
PAM Authentication	Success
LDAP	Success
Email Extension	Success
Mailer	Success
Theme Manager	Success
Dark Theme	Success
Loading plugin extensions	Success
Pipeline: REST API	Success
Pipeline: Stage View	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

REST API Jenkins 2.489

Click on “Dashboard”

The screenshot shows the Jenkins dashboard. On the left, there are links for New Item, Build History, Manage Jenkins, and My Views. The main area displays a table of pipelines. One pipeline named "demo" is listed with a green checkmark icon, a yellow sun icon, the name "demo", the last success time (50 min #1), and the last failure time (N/A). Below the table, there are sections for Build Queue (No builds in the queue) and Build Executor Status (0/2).

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	demo	50 min #1	N/A	6.5 sec

Icon: S M L

Click on the pipeline “demo”

The screenshot shows the Jenkins Pipeline Status page for the 'demo' pipeline. On the left, there's a sidebar with various actions: Status (highlighted), Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. Below this is a 'Builds' section with a 'Filter' input, showing a single build entry: '#1 Dec 14 18:28 No Changes'. The main area has tabs for 'Stage View' and 'Permalinks'. Under 'Stage View', it says 'Average stage times: (Average full run time: ~6s)' and shows a single stage named 'Git Checkout' with a duration of '3s'. Under 'Permalinks', there's a list of links to recent builds.

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Stages

Rename

Pipeline Syntax

Stage View

Average stage times:
(Average full run time: ~6s)

#1	Dec 14 18:28	No Changes
Git Checkout		
3s		

Permalinks

- Last build (#1), 50 min ago
- Last stable build (#1), 50 min ago
- Last successful build (#1), 50 min ago
- Last completed build (#1), 50 min ago

Builds

Filter

December 14, 2024

#1 11:28 PM

If you see, you are getting a clean view. First stage is to get checkout, so that is done.

STEP 10: Install and Configure Maven with Jenkins for Code Compile

The next step is Code compile using Maven. Let us install Maven. Go to this link:

<https://github.com/gashok13193/DevOps-Docs>

Click on “maven.md” under “Maven”

Installing Maven on Linux/Ubuntu

Go to the URL: <https://maven.apache.org/download.cgi> Copy the link for the "Binary tar.gz archive" file. Then run the following commands to download and untar it.

Step 1: Download the Maven Binaries

```
wget https://dlcdn.apache.org/maven/maven-3/3.9.9/binaries/apache-maven-3.9.9-bin.tar.gz --no-check-certificate
tar -xvf apache-maven-3.9.9-bin.tar.gz
sudo mv apache-maven-3.9.9 /opt/
```

Step 2: Setting M2_HOME and Path Variables

Add the following lines to the user profile file (.profile).

```
M2_HOME='/opt/apache-maven-3.9.9'
PATH="$M2_HOME/bin:$PATH"
export PATH
```

Relaunch the terminal or execute `source .profile` to apply the changes.

Step 3: Verify the Maven installation

Execute `mvn -version` command and it should produce the following output.

```
$ mvn -version
Apache Maven 3.6.3 (ceceddd343002696d0abb50b32b541b8a6ba2883f)
Maven home: /opt/apache-maven-3.6.3
Java version: 13.0.1, vendor: Oracle Corporation, runtime: /opt/jdk-13.0.1
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-47-generic", arch: "amd64", family: "unix"
$
```

Run this command to install Maven:

```
 wget https://dlcdn.apache.org/maven/maven-3/3.9.9/binaries/apache-maven-3.9.9-bin.tar.gz --no-check-certificate
 tar -xvf apache-maven-3.9.9-bin.tar.gz
 sudo mv apache-maven-3.9.9 /opt/
```

```
apache-maven-3.9.9/lib/org.eclipse.sisu.plexus-0.9.0.M3.jar
apache-maven-3.9.9/lib/jansi-2.4.1.jar
apache-maven-3.9.9/lib/maven-resolver-named-locks-1.9.22.jar
apache-maven-3.9.9/lib/maven-plugin-api-3.9.9.jar
apache-maven-3.9.9/lib/wagon-http-shared-3.5.3.jar
apache-maven-3.9.9/lib/httpclient-4.5.14.jar
apache-maven-3.9.9/lib/maven-resolver-transport-file-1.9.22.jar
apache-maven-3.9.9/lib/plexus-component-annotations-2.1.0.jar
apache-maven-3.9.9/lib/maven-resolver-connector-basic-1.9.22.jar
apache-maven-3.9.9/lib/plexus-utils-3.5.1.jar
ubuntu@ip-172-31-31-55:~$
```

Then run the next part of the code to set the maven path:

```
M2_HOME='/opt/apache-maven-3.9.9'
PATH="$M2_HOME/bin:$PATH"
export PATH
```

```
apache-maven-3.9.9/lib/maven-settings-builder-3.9.9.jar
apache-maven-3.9.9/lib/failureaccess-1.0.2.jar
apache-maven-3.9.9/lib/plexus-sec-dispatcher-2.0.jar
apache-maven-3.9.9/lib/maven-shared-utils-3.4.2.jar
apache-maven-3.9.9/lib/guava-33.2.1-jre.jar
apache-maven-3.9.9/lib/wagon-file-3.5.3.jar
apache-maven-3.9.9/lib/maven-model-builder-3.9.9.jar
apache-maven-3.9.9/lib/maven-slf4j-provider-3.9.9.jar
apache-maven-3.9.9/lib/plexus-interpolation-1.27.jar
apache-maven-3.9.9/lib/maven-resolver-transport-http-1.9.22.jar
apache-maven-3.9.9/lib/maven-model-3.9.9.jar
apache-maven-3.9.9/lib/plexus-xml-3.0.1.jar
apache-maven-3.9.9/lib/maven-resolver-spi-1.9.22.jar
apache-maven-3.9.9/lib/maven-compat-3.9.9.jar
apache-maven-3.9.9/lib/guice-5.1.0.jar
apache-maven-3.9.9/lib/commons-codec-1.17.1.jar
apache-maven-3.9.9/lib/javax.inject-1.jar
apache-maven-3.9.9/lib/org.eclipse.sisu.plexus-0.9.0.M3.jar
apache-maven-3.9.9/lib/jansi-2.4.1.jar
apache-maven-3.9.9/lib/maven-resolver-named-locks-1.9.22.jar
apache-maven-3.9.9/lib/maven-plugin-api-3.9.9.jar
apache-maven-3.9.9/lib/wagon-http-shared-3.5.3.jar
apache-maven-3.9.9/lib/httpclient-4.5.14.jar
apache-maven-3.9.9/lib/maven-resolver-transport-file-1.9.22.jar
apache-maven-3.9.9/lib/plexus-component-annotations-2.1.0.jar
apache-maven-3.9.9/lib/maven-resolver-connector-basic-1.9.22.jar
apache-maven-3.9.9/lib/plexus-utils-3.5.1.jar
ubuntu@ip-172-31-31-55:~$ M2_HOME='/opt/apache-maven-3.9.9'
PATH="$M2_HOME/bin:$PATH"
export PATH
ubuntu@ip-172-31-31-55:~$
```

Verify the Maven installation by using the command:

```
mvn -version
```

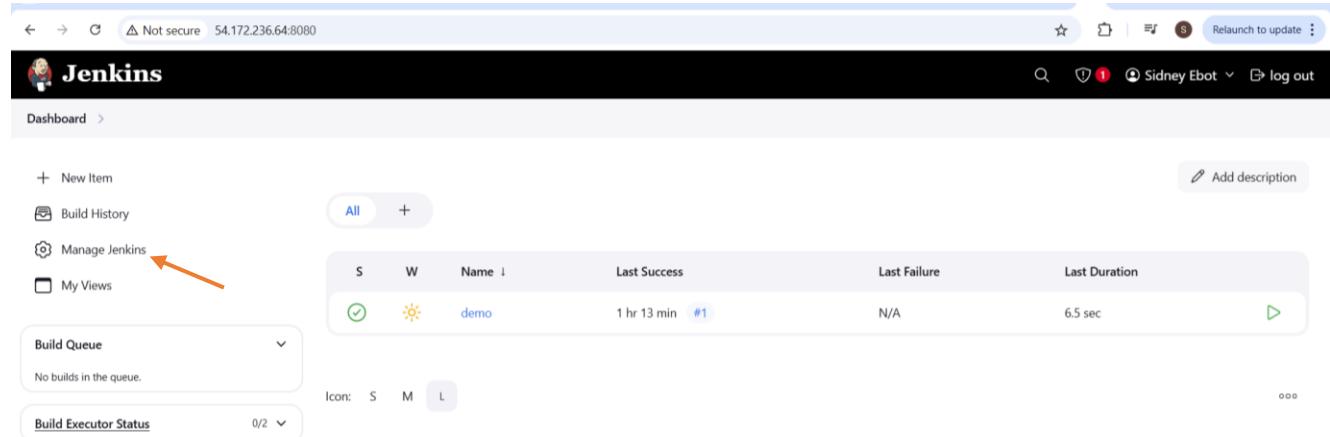
```
ubuntu@ip-172-31-31-55:~$ mvn -version
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: /opt/apache-maven-3.9.9
Java version: 17.0.13, vendor: Ubuntu, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-1018-aws", arch: "amd64", family: "unix"
ubuntu@ip-172-31-31-55:~$
```

Maven version 3.9.9 has been installed.

PART 1: Configure Maven with Jenkins

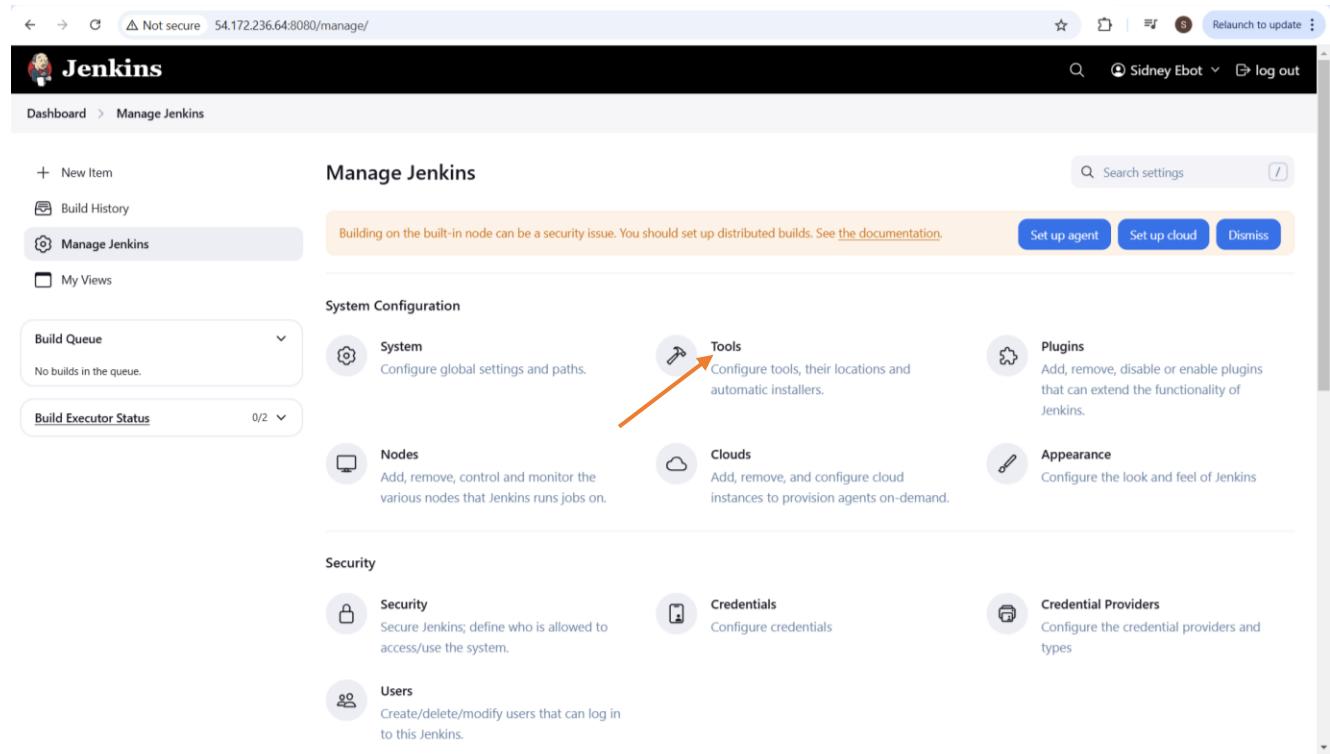
Now, let us configure Maven with Jenkins

Go to “Dashboard”



The screenshot shows the Jenkins dashboard at 54.172.236.64:8080. The left sidebar has links for 'New Item', 'Build History', 'Manage Jenkins' (which is highlighted with an orange arrow), and 'My Views'. The main area shows a build history table with one entry: 'demo' (Status: Green, Last Success: 1 hr 13 min ago, Last Failure: N/A, Last Duration: 6.5 sec). Below the table are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0/2).

Click on “Manage Jenkins”



The screenshot shows the 'Manage Jenkins' page at 54.172.236.64:8080/manage/. The left sidebar has links for 'New Item', 'Build History', 'Manage Jenkins' (which is highlighted with an orange arrow), and 'My Views'. The main area is titled 'Manage Jenkins' and contains several configuration sections: 'System Configuration' (with 'System' and 'Nodes' options), 'Security' (with 'Security', 'Users', and 'Groups' options), and 'System' (with 'Tools', 'Clouds', 'Credentials', 'Users', and 'Groups' options). A callout box highlights the 'Tools' section with the text 'Configure tools, their locations and automatic installers.' An orange arrow points to the 'Tools' link.

Click on “Tools”

The screenshot shows the Jenkins 'Tools' configuration page. At the top, there are sections for 'Maven Configuration' and 'Git installations'. Under 'Git installations', a new entry named 'Default' is being created, with fields for 'Name' and 'Save' and 'Apply' buttons. A dropdown menu is open on the right side of the browser window.

Duplicate the Tab

The screenshot shows the same Jenkins 'Tools' configuration page as above. A context menu is open on the right side of the browser window, with the 'Duplicate' option highlighted by a red arrow. The menu also includes options like 'New tab to the right', 'Add tab to reading list', 'Add tab to new group', 'Move tab to new window', 'Organize similar tabs', 'Reload', 'Close', 'Close other tabs', and 'Close tabs to the right'.

Select “Duplicate”

Tools

Maven Configuration

Default settings provider
Use default maven settings

Default global settings provider
Use default maven global settings

JDK installations

Add JDK

Git installations

Git

Name
Default

Save Apply

PART 2: Install Plugins

Click on “Manage Jenkins”

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

System Configuration

New Item Build History My Views

Build Queue No builds in the queue. **Build Executor Status** 0/2

Nodes Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

System Configure global settings and paths.

Tools Configure tools, their locations and automatic installers.

Clouds Add, remove, and configure cloud instances to provision agents on-demand.

Plugins Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Appearance Configure the look and feel of Jenkins.

Security

Users Create/delete/modify users that can log in to this Jenkins.

Credentials Configure credentials.

Credential Providers Configure the credential providers and types.

Click on “Plugins”

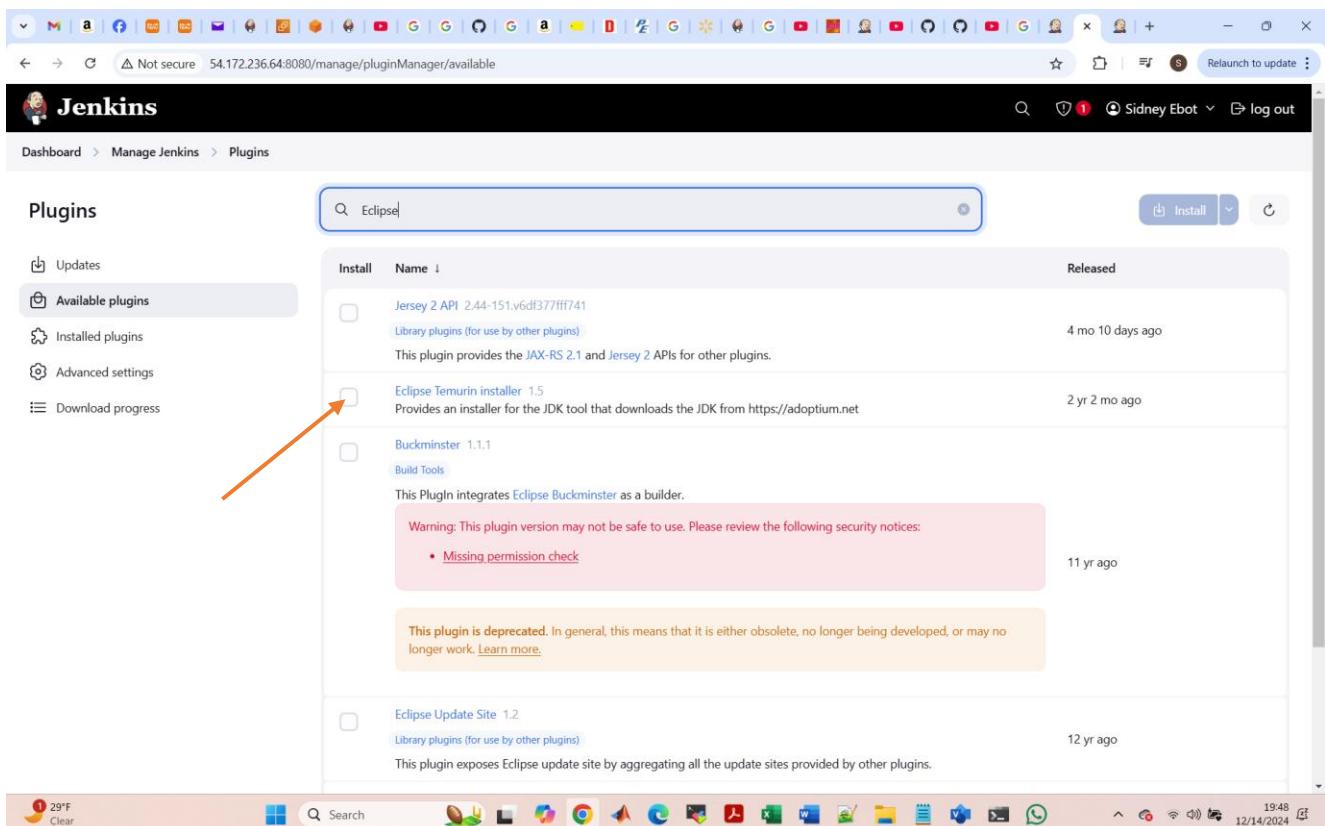
The screenshot shows the Jenkins Plugins page. On the left, a sidebar has tabs for 'Updates', 'Available plugins' (which is selected and highlighted in grey), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main content area has a search bar at the top right labeled 'Search plugin updates'. Below it, a message says 'No updates available' with a checkmark icon. A note at the bottom states: 'Disabled rows are already upgraded, awaiting restart. Shaded but selectable rows are in progress or failed.' An orange arrow points from the text 'Click on “Available Plugins”' to the 'Available plugins' tab in the sidebar.

Click on “Available Plugins”

The screenshot shows the Jenkins Plugins page with the 'Available plugins' tab selected. The main area displays a table of available plugins. The columns are 'Install', 'Name ↓', and 'Released'. A search bar at the top of the table is highlighted with a blue border and an orange arrow pointing to it. The table lists several plugins:

Install	Name ↓	Released
<input type="checkbox"/>	JavaMail API 1.6.2-10 Library plugins (for use by other plugins) This plugin provides the JavaMail API for other plugins.	6 mo 25 days ago
<input type="checkbox"/>	Command Agent Launcher 116.vd85919c54a_d6 Agent Management Allows agents to be launched using a specified command.	1 mo 2 days ago
<input type="checkbox"/>	Oracle Java SE Development Kit Installer 80.v8a_dee33ed6f0 Allows the Oracle Java SE Development Kit (JDK) to be installed via download from Oracle's website.	4 mo 10 days ago
<input type="checkbox"/>	JSch dependency 0.2.16-86.v42e010cf9484b_58 Library plugins (for use by other plugins) Miscellaneous Jenkins plugin that brings the JSch library as a plugin dependency, and provides an SSHAuthenticatorFactory for using JSch with the ssh-credentials plugin.	11 mo ago
<input type="checkbox"/>	SSH server 3.330.vc866a_8389b_58 Adds SSH server functionality to Jenkins, exposing CLI commands through it.	6 mo 8 days ago
<input type="checkbox"/>	Authentication Tokens API 1.119.v50285141b_7e1 This plugin provides an API for converting credentials into authentication tokens in Jenkins.	5 mo 9 days ago
<input type="checkbox"/>	Javadoc 280.v050b_5c849f69 This plugin adds Javadoc support to Jenkins.	4 mo 10 days ago

Search for “Eclipse”



Jenkins

Dashboard > Manage Jenkins > Plugins

Plugins

Search: Eclipse

Install Name Released

Install	Name	Released
<input type="checkbox"/>	Jersey 2 API 2.44-151.v6df377fff741	4 mo 10 days ago
<input type="checkbox"/>	Eclipse Temurin installer 1.5	2 yr 2 mo ago
<input type="checkbox"/>	Buckminster 1.1.1	11 yr ago
<input type="checkbox"/>	Eclipse Update Site 1.2	12 yr ago

This plugin provides the JAX-RS 2.1 and Jersey 2 APIs for other plugins.

Provides an installer for the JDK tool that downloads the JDK from https://adoptium.net

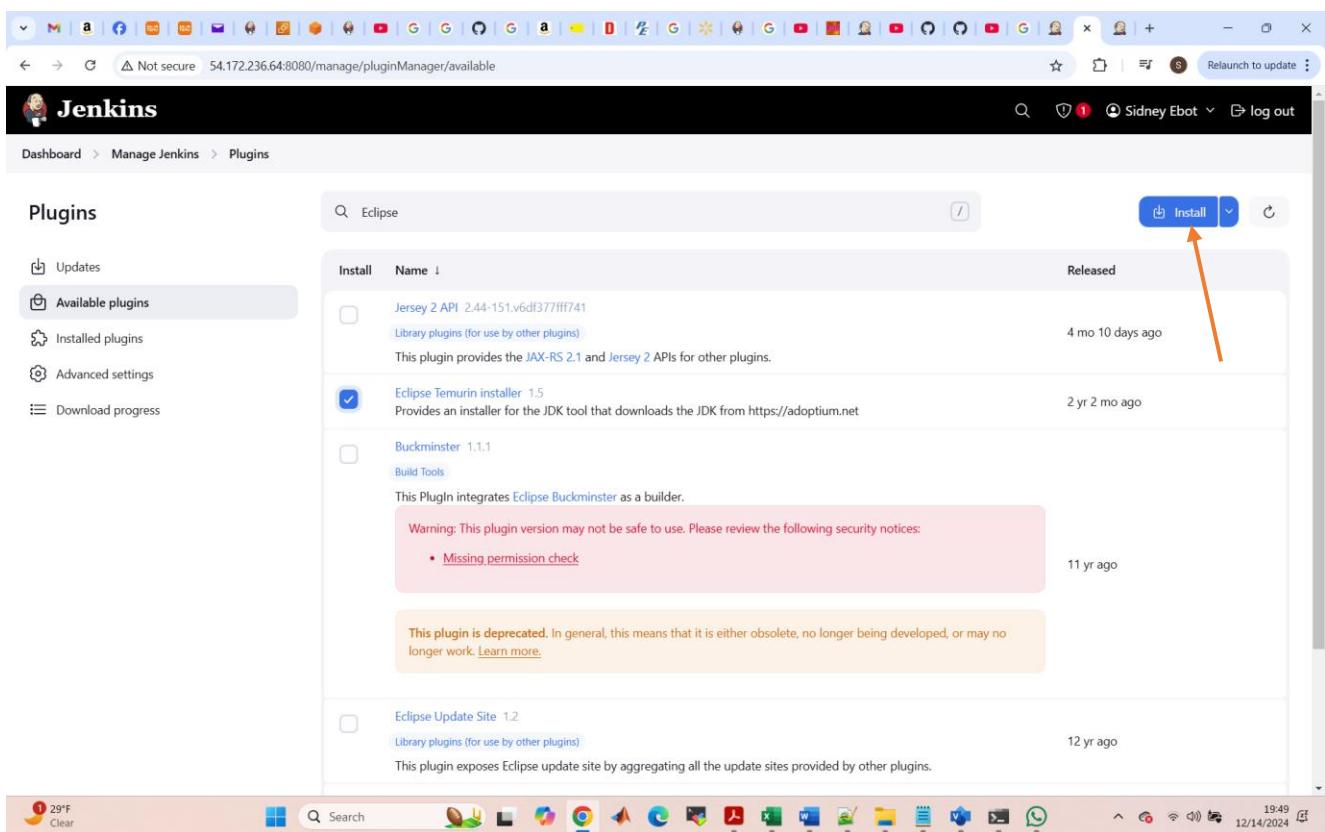
Warning: This plugin version may not be safe to use. Please review the following security notices:

- Missing permission check

This plugin is deprecated. In general, this means that it is either obsolete, no longer being developed, or may no longer work. [Learn more](#).

29°F Clear Search 19:48 12/14/2024

Select “Eclipse Temurin Installer”



Jenkins

Dashboard > Manage Jenkins > Plugins

Plugins

Search: Eclipse

Install Name Released

Install	Name	Released
<input type="checkbox"/>	Jersey 2 API 2.44-151.v6df377fff741	4 mo 10 days ago
<input checked="" type="checkbox"/>	Eclipse Temurin installer 1.5	2 yr 2 mo ago
<input type="checkbox"/>	Buckminster 1.1.1	11 yr ago
<input type="checkbox"/>	Eclipse Update Site 1.2	12 yr ago

Provides an installer for the JDK tool that downloads the JDK from https://adoptium.net

Warning: This plugin version may not be safe to use. Please review the following security notices:

- Missing permission check

This plugin is deprecated. In general, this means that it is either obsolete, no longer being developed, or may no longer work. [Learn more](#).

29°F Clear Search 19:49 12/14/2024

Click on “Install”

Not secure 54.172.236.64:8080/manage/pluginManager/updates/ Relaunch to update

Dashboard > Manage Jenkins > Plugins

Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress**

Metrics	Status
Pipeline Graph View	Success
Git	Success
EDDSA API	Success
Trilead API	Success
SSH Build Agents	Success
Matrix Authorization Strategy	Success
PAM Authentication	Success
LDAP	Success
Email Extension	Success
Mailer	Success
Theme Manager	Success
Dark Theme	Success
Loading plugin extensions	Success
Pipeline: REST API	Success
Pipeline: Stage View	Success
Loading plugin extensions	Success
JavaMail API	Success
Eclipse Temurin installer	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

REST API Jenkins 2.489

It has been installed. Also search for “**Maven**”

Jenkins

Dashboard > Manage Jenkins > Plugins

Plugins

Available plugins

Search: Maven

Install

Install	Name	Released
<input type="checkbox"/>	Maven Integration 3.24	Build Tools 1 mo 22 days ago
<input type="checkbox"/>	Config File Provider 980.v88956a_a_5d6a_d	Groovy-related External Site/Tool Integrations Maven 1 mo 23 days ago
<input type="checkbox"/>	Jira 3.13	External Site/Tool Integrations Maven jira 9 mo 5 days ago
<input type="checkbox"/>	Pipeline Maven Integration 1469.v15ca_a_b_90b_44	pipeline Maven 19 days ago
<input type="checkbox"/>	Artifactory 4.0.8	pipeline This plugin allows your build jobs to deploy artifacts and resolve dependencies to and from Artifactory, and then have them linked to the build job that created them. The plugin includes a vast collection of features, including a rich pipeline API library and release management for Maven and Gradle builds with Staging and Promotion. 5 mo 6 days ago

Select “**Maven Integration**”

Jenkins

Dashboard > Manage Jenkins > Plugins

Plugins

Available plugins

Search: Maven

Install	Name	Released
<input checked="" type="checkbox"/>	Maven Integration 3.24	1 mo 22 days ago
<input type="checkbox"/>	Config File Provider 980.v88956a_a_5d6a_d	1 mo 23 days ago
<input type="checkbox"/>	Jira 3.13	9 mo 5 days ago
<input type="checkbox"/>	Pipeline Maven Integration 1469.v15ca_a_b_90b_44	19 days ago
<input type="checkbox"/>	Artifactory 4.0.8	5 mo 6 days ago

Click on “Install”

Not secure 54.172.236.64:8080/manage/pluginManager/updates/

Dashboard > Manage Jenkins > Plugins

Plugins

Available plugins

Plugin	Status
Filebeat API	Success
SSH Build Agents	Success
Matrix Authorization Strategy	Success
PAM Authentication	Success
LDAP	Success
Email Extension	Success
Mailer	Success
Theme Manager	Success
Dark Theme	Success
Loading plugin extensions	Success
Pipeline: REST API	Success
Pipeline: Stage View	Success
Loading plugin extensions	Success
JavaMail API	Success
Eclipse Temurin installer	Success
Loading plugin extensions	Success
Javadoc	Success
JSch dependency	Success
Maven Integration	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

The installation is done, refresh the previous page of “tools”

Tools

Maven Configuration

Default settings provider

Use default maven settings



?

Default global settings provider

Use default maven global settings



?

JDK installations

Add JDK

Git installations

Git

Name

Default



Save

Apply

PART 3: Configure Maven Installation on Jenkins

Scroll down to “Maven Installation”

Not secure 54.172.236.64:8080/manage/configureTools/

Dashboard > Manage Jenkins > Tools

git

Install automatically ?

Add Git

Gradle installations

Add Gradle

Ant installations

Add Ant

Maven installations

Add Maven

Save Apply

Jenkins 2.489

Under “Maven Installations”, click on “Add Maven”

Maven installations

[Add Maven](#)

≡ Maven

Name

 Required Install automatically ?

≡ Install from Apache

Version

3.9.9

[Add Installer ▾](#)[Add Maven](#)[Save](#)[Apply](#)Give it name as “**maven3**”

Maven installations

[Add Maven](#)

≡ Maven

Name

maven3

 Install automatically ?

≡ Install from Apache

Version

3.9.9

[Add Installer ▾](#)[Add Maven](#)[Save](#)[Apply](#)Then scroll up to “**JDK Installations**”

Dashboard > Manage Jenkins > Tools

Use default maven global settings

JDK installations

Add JDK

Git installations

Git

Name: Default

Path to Git executable: git

Install automatically

Add Git

Save Apply

Click on “Add JDK”

Not secure 54.172.236.64:8080/manage/configureTools/ Relaunch to update

Dashboard > Manage Jenkins > Tools

Use default maven global settings

JDK installations

Add JDK

JDK

Name

● Required

JAVA_HOME

Install automatically

Add JDK

Git installations

Save Apply

Make the naming and selection as shown below:

Use the name “jdk”, check the “Install Automatically” and click on the drop down on “Add Installer” and select “Install from Adoptium.net” then select “Jdk version 17”

The screenshot shows the Jenkins 'Tools' configuration page under 'Manage Jenkins'. A new 'JDK installations' entry is being added with the name 'jdk'. The 'Install automatically' checkbox is checked. The 'Add Installer' dropdown is open, showing the 'Install from adoptium.net' option, which is highlighted with a red border. An orange arrow points to the 'Save' button at the bottom of the page.

Then click on “Save”

The screenshot shows the Jenkins 'Manage Jenkins' dashboard. The 'Manage Jenkins' link in the sidebar is highlighted with a red arrow. The main area displays the 'System Configuration' section with links to 'System', 'Tools', 'Plugins', 'Nodes', 'Clouds', 'Appearance', 'Security', 'Credentials', 'Users', and 'Credential Providers'.

Then, go to “Dashboard” again

The screenshot shows the Jenkins dashboard. On the left, there are links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. Below these are sections for 'Build Queue' (empty) and 'Build Executor Status' (0/2). The main area displays a table with columns: S (Status), W (Work), Name (demo), Last Success (3 hr 6 min #1), Last Failure (N/A), and Last Duration (6.5 sec). An orange arrow points to the 'Name' column of the 'demo' row.

Click on the pipeline “demo”

The screenshot shows the 'demo' pipeline details. The top navigation bar shows 'Dashboard > demo >'. The left sidebar has options: Status (selected), Changes, Build Now, Configure (highlighted with an orange arrow), Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. The right panel shows the 'Stage View' with a single stage named 'Git Checkout' which took 3s. A summary at the bottom states 'Average stage times: (Average full run time: ~6s)'.

Status

</> Changes

▷ Build Now

⚙ Configure

Delete Pipeline

🔍 Full Stage View

☰ Stages

✍ Rename

ⓘ Pipeline Syntax

demo

Stage View

Git Checkout

3s

Average stage times:
(Average full run time: ~6s)

#1	Dec 14 18:28	No Changes
----	-----------------	------------

3s

Permalinks

- Last build (#1), 3 hr 6 min ago
- Last stable build (#1), 3 hr 6 min ago
- Last successful build (#1), 3 hr 6 min ago
- Last completed build (#1), 3 hr 6 min ago

The screenshot shows the 'Builds' page for the 'demo' pipeline. It features a search bar with 'Filter' and a date range from 'December 14, 2024'. A specific build is highlighted: '#1 11:28 PM'. A dropdown arrow is shown next to the build entry.

Click on “Configure”

Dashboard > demo > Configuration

Configure

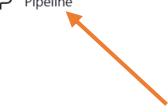
General

Enabled

General

Build Triggers

Advanced Project Options

Pipeline 

Description

Plain text [Preview](#)

- Discard old builds [?](#)
- Do not allow concurrent builds
- Do not allow the pipeline to resume if the controller restarts
- GitHub project
- Pipeline speed/durability override [?](#)
- Preserve stashes from completed builds [?](#)
- This project is parameterized [?](#)
- Throttle builds [?](#)

Save **Apply**

Click on “**Pipeline**” on the left-hand side to go to the code section of the pipeline or scroll down to “**Pipeline**”. Copy the stage

Dashboard > demo > Configuration

Configure

Pipeline 

Definition

Pipeline script

Script [?](#)

```

1 * pipeline {
2     agent any
3
4     stages {
5         stage('Hello') {
6             steps {
7                 git 'https://github.com/ebotsmith2000/jenkins-pipeline.git'
8             }
9         }
10    }
11 }
12

```

Use Groovy Sandbox [?](#)

[Pipeline Syntax](#)

Advanced

Advanced [▼](#)

Save **Apply**

and paste it to have a second stage.

Dashboard > demo > Configuration

Advanced

Configure

General

Build Triggers

Advanced Project Options

Pipeline

Pipeline

Definition

Pipeline script

```

1 pipeline {
2   agent any
3
4   stages {
5     stage("Git Checkout") {
6       steps {
7         git 'https://github.com/ebotsmith2000/CICD-Pipeline.git'
8       }
9     }
10    stage("Git Checkout") {
11      steps {
12        git 'https://github.com/ebotsmith2000/CICD-Pipeline.git'
13      }
14    }
15  }
16 }
17

```

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

We have to configure the Maven. Before that we have installed two tools (**Maven and Java**), the tools will be configured with names as stated in tools part of the code in the “Pipeline”

Pipeline

Definition

Pipeline script

```

1 pipeline {
2   agent any
3
4   tools{
5     jdk 'jdk'
6     maven 'maven3'
7   }
8
9   stages {
10    stage('Git Checkout') {
11      steps {
12        git 'https://github.com/ebotsmith2000/end-to-end.git'
13      }
14    }
15    stage('Code Compile') {
16      steps {
17        sh 'mvn clean compile'
18      }
19    }
20  }
21 }
22

```

Use Groovy Sandbox ?

Pipeline Syntax

Save

Apply

The code on the pipeline will be like below

```

pipeline {
    agent any

    tools{
        jdk 'jdk'
        maven 'maven3'
    }

    stages {
        stage('Git Checkout') {
            steps {
                git 'https://github.com/ebotsmith2000/jenkins-pipeline.git'
            }
        }
        stage('Code Compile') {
            steps {
                sh 'mvn clean compile'
            }
        }
    }
}

```

Click on “Save”

The screenshot shows the Jenkins Pipeline demo status page. On the left, there's a sidebar with options: Status (highlighted), Changes, Build Now (with an orange arrow pointing to it), Configure, Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. The main area is titled "Stage View" and shows a single stage named "Git Checkout". It indicates "Average stage times: (Average full run time: ~6s)". Below the stage view is a "Permalinks" section with a list of build links. At the bottom left is a "Builds" section showing a single build from December 14, 2024, at 11:28 PM.

Status

- </> Changes
- ▷ Build Now
- ⚙ Configure
- trash Delete Pipeline
- 🔍 Full Stage View
- 🔗 Stages
- ✍ Rename
- ⓘ Pipeline Syntax

Stage View

Average stage times:
(Average full run time: ~6s)

Git Checkout	
#1	Dec 14 18:28
No Changes	3s

Permalinks

- Last build (#1), 3 hr 21 min ago
- Last stable build (#1), 3 hr 21 min ago
- Last successful build (#1), 3 hr 21 min ago
- Last completed build (#1), 3 hr 21 min ago

Builds

December 14, 2024

#1 11:28 PM

Let us try to build the code again by clicking on “Build Now”

← → ⌂ Not secure 54.172.236.64:8080/job/demo/2/console

Dashboard > demo > #2

```

Progress (2): 310/334 kB | 177/192 kB
Progress (2): 310/334 kB | 192 kB
Progress (2): 326/334 kB | 192 kB
Progress (2): 334 kB | 192 kB

Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/4.0.0/plexus-utils-4.0.0.jar (192 kB at 8.7 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/qdox/qdox/2.0.3/qdox-2.0.3.jar (334 kB at 13 MB/s)
[INFO] Recompiling the module because of changed source code.
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file with javac [debug target 11] to target/classes
[WARNING] system modules path not set in conjunction with -source 11
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.124 s
[INFO] Finished at: 2024-12-15T02:52:26Z
[INFO] -----
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

The build is successful, it has compiled the code and everything is ready. Now, let us click on “**demo**”

Dashboard > demo >

Declarative: Tool Install	Git Checkout	Code Compile
22s	566ms	6s
22s	566ms	6s

Stage View

Average stage times:
(Average full run time: ~31s)

#2
Dec 14 21:51 No Changes

Permalinks

- Last build (#2), 3 min 0 sec ago
- Last stable build (#2), 3 min 0 sec ago
- Last successful build (#2), 3 min 0 sec ago
- Last completed build (#2), 3 min 0 sec ago

Builds

- Today: #2 2:51 AM
- December 14, 2024: #1 11:28 PM

Here you can see the stages “**Git Checkout**”, “**Code Compile**” and an additional stage called “**Declarative Tool Install**” for the installed tools. Now, we are done with Code Compile.

STEP 11: Install and Configure SonarQube with Jenkins for Static Code Analysis

Now, we have to do “Static Code Analysis” using SonarQube.

PART 1: Install SonarQube on Server

Go to the instance terminal and run this command to download the package:

```
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-10.2.1.78527.zip
ubuntu@ip-172-31-31-55:~$ wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.0.65466.zip
--2024-12-15 03:01:46-- https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.0.65466.zip
Resolving binaries.sonarsource.com (binaries.sonarsource.com)... 99.84.188.45, 99.84.188.106, 99.84.188.21, ...
Connecting to binaries.sonarsource.com (binaries.sonarsource.com)|99.84.188.45|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 293899334 (280M) [binary/octet-stream]
Saving to: 'sonarqube-9.9.0.65466.zip'

sonarqube-9.9.0.65466.zip                                              100%[=====] 293899334/293899334
2024-12-15 03:01:49 (83.4 MB/s) - `sonarqube-9.9.0.65466.zip' saved [293899334/293899334]
ubuntu@ip-172-31-31-55:~$ █
```

The package has been downloaded.

Run the command:

11

```
ubuntu@ip-172-31-31-55:~$ wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.0.65466.zip
ubuntu@ip-172-31-31-55:~$ wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.0.65466.zip
--2024-12-15 03:01:46-- https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.0.65466.zip
Resolving binaries.sonarsource.com (binaries.sonarsource.com)... 99.84.188.45, 99.84.188.106, 99.84.188.21, ...
Connecting to binaries.sonarsource.com (binaries.sonarsource.com)|99.84.188.45|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 293899334 (280M) [binary/octet-stream]
Saving to: 'sonarqube-9.9.0.65466.zip'

sonarqube-9.9.0.65466.zip                                              100%[=====] 293899334/293899334
2024-12-15 03:01:49 (83.4 MB/s) - `sonarqube-9.9.0.65466.zip' saved [293899334/293899334]
ubuntu@ip-172-31-31-55:~$ ll
total 295940
drwxr-x--- 4 ubuntu ubuntu    4096 Dec 15 03:01 .
drwxr-xr-x  3 root   root    4096 Dec 14 20:10 ../
-rw-r--r--  1 ubuntu ubuntu   801 Dec 15 03:01 .bash_history
-rw-r--r--  1 ubuntu ubuntu  2204 Dec 14 20:53 .bash_logout
-rw-r--r--  1 ubuntu ubuntu  3771 Mar 31 2024 .bashrc
drwxr--r--  2 ubuntu ubuntu  4096 Dec 14 20:53 .cache/
-rw-r--r--  1 ubuntu ubuntu   807 Mar 31 2024 .profile
drwxr--r--  2 ubuntu ubuntu  4096 Dec 14 20:10 .ssh/
-rw-r--r--  1 ubuntu ubuntu  2016 Dec 14 20:53 .sudo_as_admin_successful
-rw-r--r--  1 ubuntu ubuntu 9102945 Aug 17 18:44 sonarqube-war-9.9.0-bin.tar.gz
-rw-r--r--  1 ubuntu ubuntu 293899334 Feb  3 2023 sonarqube-9.9.0.65466.zip
ubuntu@ip-172-31-31-55:~$ █
```

You can see the SonarQube package, we have to unzip the zipped folders. So, we have to run a command to install unzip package.

sudo apt install unzip

```

Saving to: 'sonarqube-9.9.0.65466.zip'
sonarqube-9.9.0.65466.zip          100%[=====] 280.28M 83.1MB/s   in 3.4s
2024-12-15 03:01:49 (83.4 MB/s) - 'sonarqube-9.9.0.65466.zip' saved [293899334/293899334]

ubuntu@ip-172-31-31-55:~$ ll
total 295940
drwxr-x---  4 ubuntu ubuntu    4096 Dec 15 03:01 .
drwxr-xr-x  3 root  root    4096 Dec 14 20:10 ..
-rw-r--r--  1 ubuntu ubuntu     801 Dec 15 01:29 .bash_history
-rw-r--r--  1 ubuntu ubuntu    220 Mar 31 2024 .bash_logout
-rw-r--r--  1 ubuntu ubuntu   3771 Mar 31 2024 .bashrc
drwxr----- 2 ubuntu ubuntu   4096 Dec 14 20:53 .cache/
-rw-r--r--  1 ubuntu ubuntu    807 Mar 31 2024 .profile
drwxr----- 2 ubuntu ubuntu   4096 Dec 14 20:10 .ssh/
-rw-r--r--  1 ubuntu ubuntu    0 Dec 14 21:07 .sudo_as_admin_successful
-rw-rw-r--  1 ubuntu ubuntu  9102945 Aug 17 18:44 apache-maven-3.9.9-bin.tar.gz
-rw-rw-r--  1 ubuntu ubuntu   4096 Feb  3 2023 sonarqube-9.9.0.65466.zip
ubuntu@ip-172-31-31-55:~$ sudo apt install unzip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  zip
The following NEW packages will be installed:
  unzip
0 upgraded, 1 newly installed, 0 to remove and 53 not upgraded.
Need to get 174 kB of archives.
After this operation, 384 kB of additional disk space will be used.
Get:1 https://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 unzip amd64 6.0-28ubuntu4.1 [174 kB]
Fetched 174 kB in 0s (8760 kB/s)
Selecting previously unselected package unzip.
(Reading database... 85359 files and directories currently installed.)
Preparing to unpack .../unzip_6.0-28ubuntu4.1_amd64.deb ...
Unpacking unzip (6.0-28ubuntu4.1) ...
Setting up unzip (6.0-28ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning for broken packages...
Scanning linux images...
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-31-55:~$ 

```

The installation is done, let us now unzip by using the command:

```

unzip sonarqube-10.2.1.78527.zip
Inflating: sonarqube-9.9.0.65466/web/images/samr.png
inflating: sonarqube-9.9.0.65466/web/apple-touch-icon-144x144.png
inflating: sonarqube-9.9.0.65466/web/apple-touch-icon-114x114.png
inflating: sonarqube-9.9.0.65466/web/apple-touch-icon-57x57.png
inflating: sonarqube-9.9.0.65466/web/index.html
  creating: sonarqube-9.9.0.65466/lib/jdbc/
  creating: sonarqube-9.9.0.65466/lib/jdbc/mssql/
inflating: sonarqube-9.9.0.65466/lib/jdbc/mssql/mssql-jdbc-11.2.2.jre17.jar
  creating: sonarqube-9.9.0.65466/lib/jdbc/postgresql/
inflating: sonarqube-9.9.0.65466/lib/jdbc/postgresql/postgresql-42.5.1.jar
  creating: sonarqube-9.9.0.65466/lib/jdbc/h2/
inflating: sonarqube-9.9.0.65466/lib/jdbc/h2/h2-2.1.214.jar
inflating: sonarqube-9.9.0.65466/lib/sonar-shutdowner-9.9.0.65466.jar
  creating: sonarqube-9.9.0.65466/elasticsearch/plugins/
ubuntu@ip-172-31-31-55:~$ 

```

It has extracted. Now, if you run the command:

11

```

ubuntu@ip-172-31-31-55:~$ ll
total 295944
drwxr-x---  5 ubuntu ubuntu    4096 Dec 15 03:11 .
drwxr-xr-x  3 root  root    4096 Dec 14 20:10 ..
-rw-r--r--  1 ubuntu ubuntu     801 Dec 15 01:29 .bash_history
-rw-r--r--  1 ubuntu ubuntu    220 Mar 31 2024 .bash_logout
-rw-r--r--  1 ubuntu ubuntu   3771 Mar 31 2024 .bashrc
drwxr----- 2 ubuntu ubuntu   4096 Dec 14 20:53 .cache/
-rw-r--r--  1 ubuntu ubuntu    807 Mar 31 2024 .profile
drwxr----- 2 ubuntu ubuntu   4096 Dec 14 20:10 .ssh/
-rw-r--r--  1 ubuntu ubuntu    0 Dec 14 21:07 .sudo_as_admin_successful
-rw-rw-r--  1 ubuntu ubuntu  9102945 Aug 17 18:44 apache-maven-3.9.9-bin.tar.gz
drwxr-xr-x 11 ubuntu ubuntu   4096 Feb  3 2023 sonarqube-9.9.0.65466/
-rw-rw-r--  1 ubuntu ubuntu 293899334 Feb  3 2023 sonarqube-9.9.0.65466.zip
ubuntu@ip-172-31-31-55:~$ 

```

You can see that we have the package SonarQube. Navigate to the package SonarQube by using the command:

```
cd sonarqube-10.2.1.78527/
```

```
ubuntu@ip-172-31-31-55:~$ ll
total 295944
drwxr-x--- 5 ubuntu ubuntu 4096 Dec 15 03:11 .
drwxr-xr-x  3 root   root  4096 Dec 14 20:10 ../
-rw-----  1 ubuntu ubuntu 801 Dec 15 01:29 .bash_history
-rw-r--r--  1 ubuntu ubuntu 220 Mar 31 2024 .bash_logout
-rw-r--r--  1 ubuntu ubuntu 3771 Mar 31 2024 .bashrc
drwx----- 2 ubuntu ubuntu 4096 Dec 14 20:53 .cache/
-rw-r--r--  1 ubuntu ubuntu 807 Mar 31 2024 .profile
drwx----- 2 ubuntu ubuntu 4096 Dec 14 20:10 .ssh/
-rw-r--r--  1 ubuntu ubuntu 0 Dec 14 21:07 .sudo_as_admin_successful
-rw-rw-r--  1 ubuntu ubuntu 9102945 Aug 17 18:44 apache-maven-3.9.9-bin.tar.gz
drwxr-xr-x 11 ubuntu ubuntu 4096 Feb  3 2023 sonarqube-9.9.0.65466/
-rw-rw-r--  1 ubuntu ubuntu 293899334 Feb  3 2023 sonarqube-9.9.0.65466.zip
ubuntu@ip-172-31-31-55:~$ cd sonarqube-9.9.0.65466/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466$
```

Run the command:

```
ll
```

```
ubuntu@ip-172-31-31-55:~$ cd sonarqube-9.9.0.65466/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466$ ll
total 128
drwxr-xr-x 11 ubuntu ubuntu 4096 Feb  3 2023 .
drwxr-x---  5 ubuntu ubuntu 4096 Dec 15 03:11 ../
-rw-r--r--  1 ubuntu ubuntu 7651 Feb  3 2023 COPYING
drwxr-xr-x  6 ubuntu ubuntu 4096 Feb  3 2023 bin/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 conf/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 data/
-rw-r--r--  1 ubuntu ubuntu 77588 Feb  3 2023 dependency-license.json
drwxr-xr-x  7 ubuntu ubuntu 4096 Feb  3 2023 elasticsearch/
drwxr-xr-x  4 ubuntu ubuntu 4096 Feb  3 2023 extensions/
drwxr-xr-x  5 ubuntu ubuntu 4096 Feb  3 2023 lib/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 logs/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 temp/
drwxr-xr-x  5 ubuntu ubuntu 4096 Feb  3 2023 web/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466$
```

We have all the SonarQube files in a bin, no matter which application it is, it will always be in the bin.
Navigate to the bin by using the command:

```
cd bin/
```

```
drwxr-xr-x 11 ubuntu ubuntu 4096 Feb  3 2023 ./
drwxr-x---  5 ubuntu ubuntu 4096 Dec 15 03:11 ../
-rw-r--r--  1 ubuntu ubuntu 7651 Feb  3 2023 COPYING
drwxr-xr-x  6 ubuntu ubuntu 4096 Feb  3 2023 bin/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 conf/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 data/
-rw-r--r--  1 ubuntu ubuntu 77588 Feb  3 2023 dependency-license.json
drwxr-xr-x  7 ubuntu ubuntu 4096 Feb  3 2023 elasticsearch/
drwxr-xr-x  4 ubuntu ubuntu 4096 Feb  3 2023 extensions/
drwxr-xr-x  5 ubuntu ubuntu 4096 Feb  3 2023 lib/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 logs/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 temp/
drwxr-xr-x  5 ubuntu ubuntu 4096 Feb  3 2023 web/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466$ cd bin/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin$ █
```

Then run the command:

```
11
```

```
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466$ cd bin/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin$ ll
total 24
drwxr-xr-x  6 ubuntu ubuntu 4096 Feb  3 2023 ./
drwxr-xr-x 11 ubuntu ubuntu 4096 Feb  3 2023 ../
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 linux-x86-64/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 macosx-universal-64/
drwxr-xr-x  3 ubuntu ubuntu 4096 Feb  3 2023 windows-x86-64/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 winsw-license/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin$ █
```

Then navigate to linux-x86-64, using the command:

```
cd linux-x86-64/
```

```
drwxr-xr-x  6 ubuntu ubuntu 4096 Feb  3 2023 web/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466$ cd bin/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin$ ll
total 24
drwxr-xr-x  6 ubuntu ubuntu 4096 Feb  3 2023 ./
drwxr-xr-x 11 ubuntu ubuntu 4096 Feb  3 2023 ../
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 linux-x86-64/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 macosx-universal-64/
drwxr-xr-x  3 ubuntu ubuntu 4096 Feb  3 2023 windows-x86-64/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 winsw-license/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin$ cd linux-x86-64/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin/linux-x86-64$ █
```

Then run the command:

```
11
```

```
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466$ cd bin/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin$ ll
total 24
drwxr-xr-x  6 ubuntu ubuntu 4096 Feb  3 2023 .
drwxr-xr-x 11 ubuntu ubuntu 4096 Feb  3 2023 ..
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 linux-x86-64/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 macosx-universal-64/
drwxr-xr-x  3 ubuntu ubuntu 4096 Feb  3 2023 windows-x86-64/
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 winsw-license/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin$ cd linux-x86-64/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin/linux-x86-64$ ll
total 16
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 .
drwxr-xr-x  6 ubuntu ubuntu 4096 Feb  3 2023 ..
-rwxr-xr-x  1 ubuntu ubuntu 7133 Feb  3 2023 sonar.sh*
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin/linux-x86-64$
```

Run the command:

```
./sonar.sh start
```

```
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 winsw-license/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin$ cd linux-x86-64/
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin/linux-x86-64$ ll
total 16
drwxr-xr-x  2 ubuntu ubuntu 4096 Feb  3 2023 .
drwxr-xr-x  6 ubuntu ubuntu 4096 Feb  3 2023 ..
-rwxr-xr-x  1 ubuntu ubuntu 7133 Feb  3 2023 sonar.sh*
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin/linux-x86-64$ ./sonar.sh start
/usr/bin/java
Starting SonarQube...
Started SonarQube.
ubuntu@ip-172-31-31-55:~/sonarqube-9.9.0.65466/bin/linux-x86-64$
```

So, SonarQube has started.

PART 2: Add Port 9000 to the Instance

By default, SonarQube will be working on **PORT 9000**. We have to add this PORT to the inbound rule of our instance.

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type <small>Info</small>	Protocol <small>Info</small>	Port range	Source <small>Info</small>	Description - optional <small>Info</small>
sgr-0b90f69132aed42c2	SSH	TCP	22	Custom	0.0.0.0/0
sgr-056cc839e43c75dea	Custom TCP	TCP	8080	Custom	Jenkins
-	Custom TCP	TCP	9000	Anyw...	SonarQube

[Add rule](#)

[Cancel](#) [Preview changes](#) [Save rules](#)

Click on “Save Rules”

sg-06628fbe687715248 - launch-wizard-24 [Actions ▾](#)

Details	
Security group name launch-wizard-24	Security group ID sg-06628fbe687715248
Description launch-wizard-24 created 2024-12-14T14:15:54.726Z	VPC ID vpc-07caa33c21359a185
Owner 510786428272	Inbound rules count 3 Permission entries
	Outbound rules count 1 Permission entry

[Inbound rules](#) [Outbound rules](#) [Sharing - new](#) [VPC associations - new](#) [Tags](#)

Inbound rules (3)

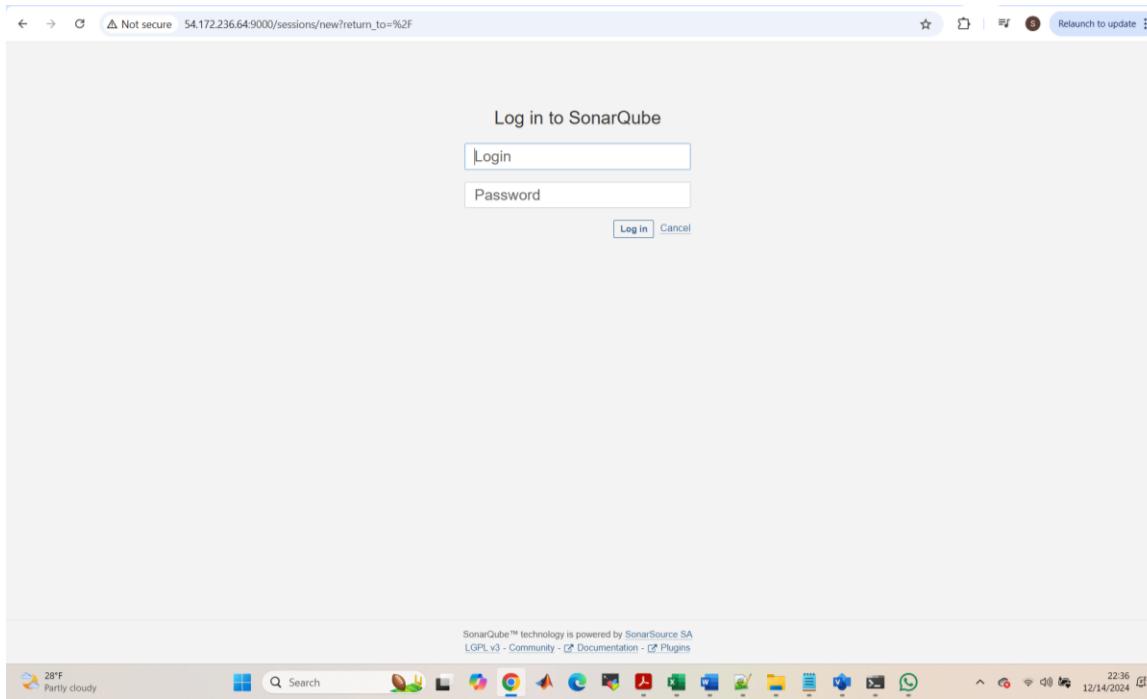
<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-0b90f69132aed42c2	IPv4	SSH	TCP	22
<input type="checkbox"/>	-	sgr-056cc839e43c75dea	IPv4	Custom TCP	TCP	8080
<input type="checkbox"/>	-	sgr-04849706a7f376e9b	IPv4	Custom TCP	TCP	9000

PART 3: Access SonarQube on Browser

So, go to the browser and paste

<http://<Public IPv4 Address>:9000>

That is <http://54.198.237.25:9000>



The default username is “**admin**” and password is “**admin**”. Use this to login

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

We are requested to change the password. I will use “**sosoebot**” as new password

Update your password

This account should not use the default password.

Enter a new password

All fields marked with * are required

Old Password *

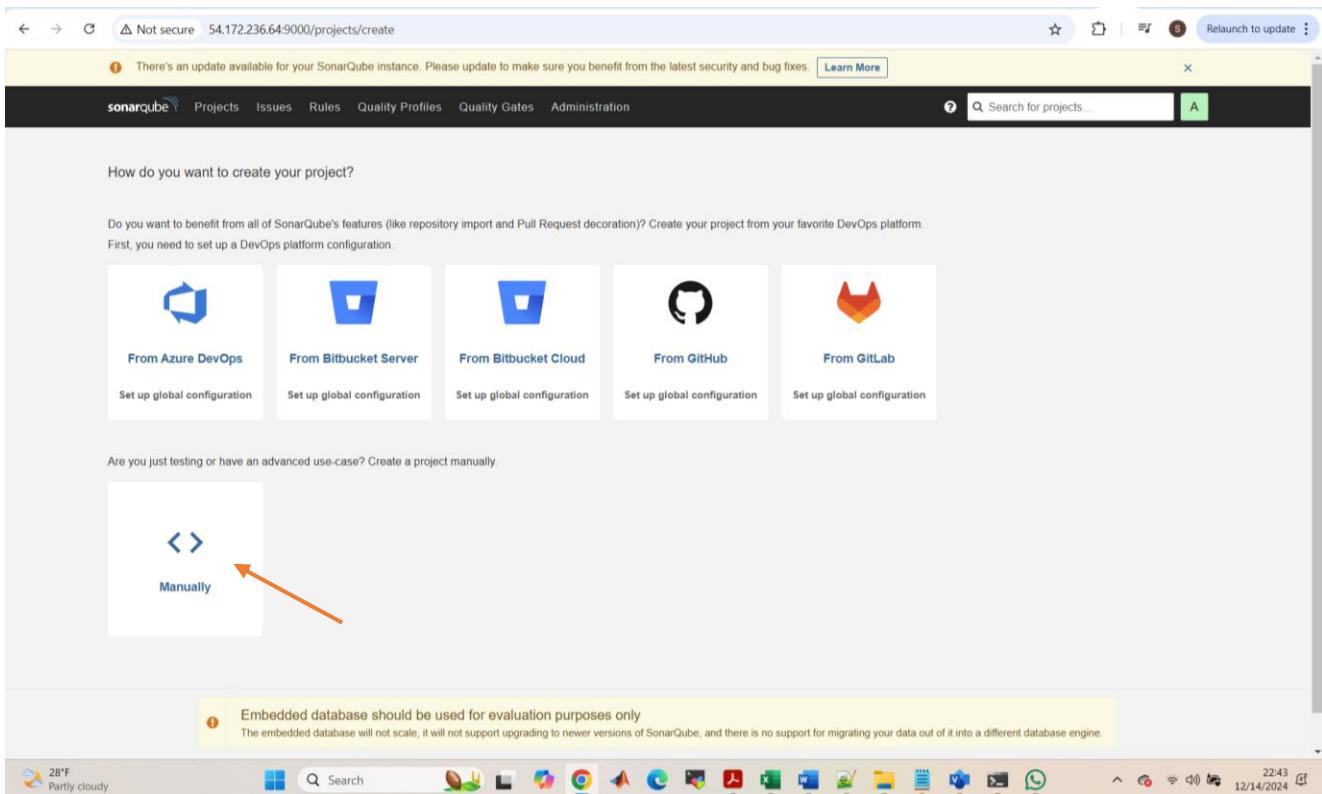
New Password *

Confirm Password *

Update



Click on “**Update**”



The screenshot shows the SonarQube interface for creating a new project. At the top, there's a message about an available update. Below the header, there are five options for importing projects from popular platforms: Azure DevOps, Bitbucket Server, Bitbucket Cloud, GitHub, and GitLab. Underneath these, there's a section for manually creating a project, indicated by a blue icon with two arrows and the word "Manually". A red arrow points to this "Manually" link. At the bottom, there's a note about the embedded database and a system tray at the bottom right showing the date and time.

It is ready, click on “**Manually**”

All fields marked with * are required

Project display name *

Project key *

Main branch name *

main

Set Up

Embedded database should be used for evaluation purposes only
The embedded database will not scale; it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.9 (build 65466) - LGPLv3 - Community - Documentation - Plugins - Web API

28°F Partly cloudy Search 12/14/2024

We will give it the name “new”, key as “new” and branch as “main”

All fields marked with * are required

Project display name *

new

Project key *

new

Main branch name *

main

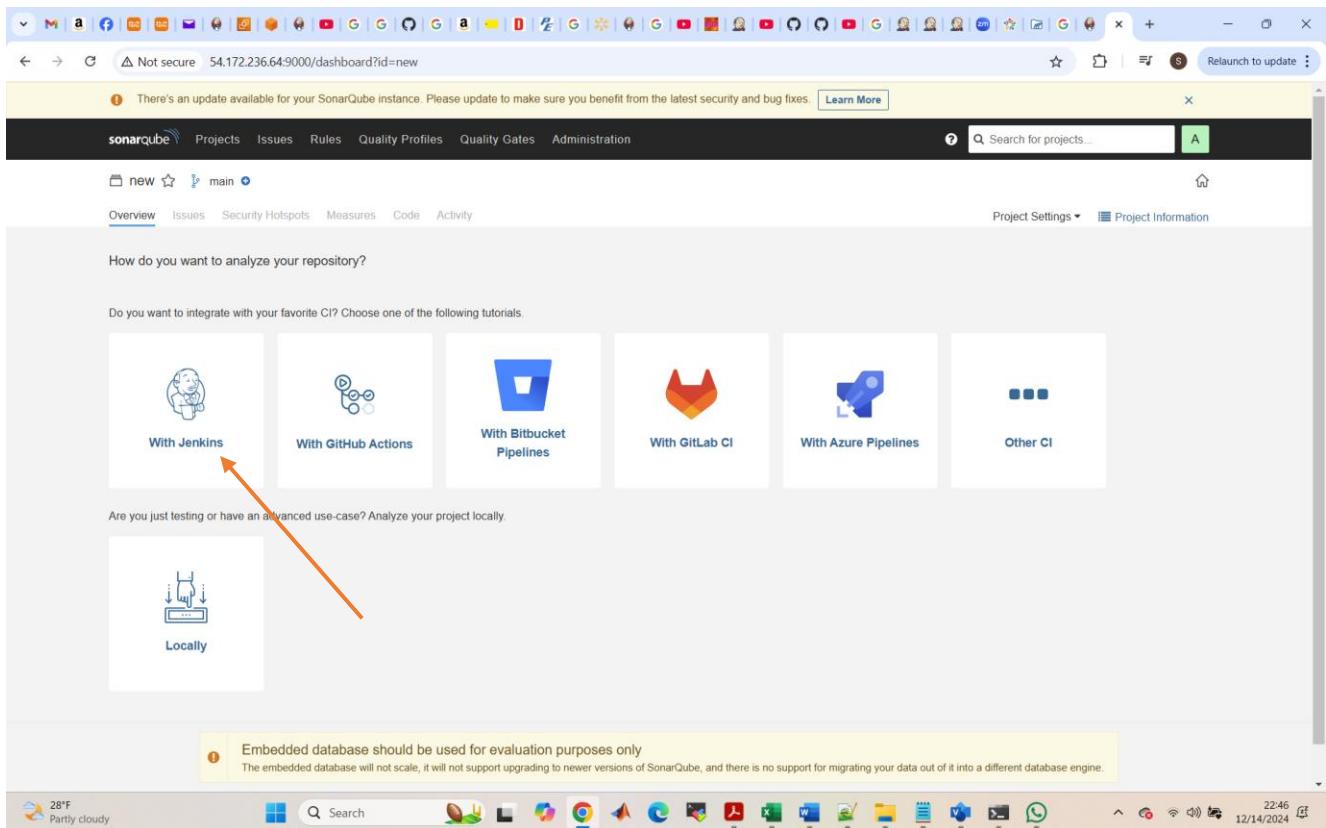
Set Up

Embedded database should be used for evaluation purposes only
The embedded database will not scale; it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.9 (build 65466) - LGPLv3 - Community - Documentation - Plugins - Web API

28°F Partly cloudy Search 12/14/2024

Click on “Set up”



Click on “**With Jenkins**” because we are using Jenkins for CI/CD

The screenshot shows the SonarQube interface after selecting "With Jenkins". The top navigation bar remains the same. The main content area has a heading "Analyze your project with Jenkins" and a sub-section "Select your DevOps platform". Below this, there are four buttons: "Bitbucket Cloud", "Bitbucket Server", "GitHub", and "GitLab". An orange arrow points to the "GitHub" button.

We are using GitHub, so select “**GitHub**”

Analyze your project with Jenkins

Select your DevOps platform GitHub

Prerequisites

To run your project analyses with Jenkins, the following plugins must be [installed](#) and [configured](#):

- SonarQube Scanner plugin for Jenkins - version 2.11 or later

For a step by step guide on installing and configuring those plugins in Jenkins, visit the [Analysis Prerequisites documentation page](#).

We recommend using the configuration in the following steps for the best results, but you can customize it as needed.

Configure Analysis > (arrow)

- 1 Create a Pipeline Job
- 2 Create a GitHub Webhook
- 3 Create a Jenkinsfile
- 4 You're all set!

Click on “Configure Analysis”

Select your DevOps platform GitHub

Prerequisites

1 Create a Pipeline Job

Create a Pipeline in order to automatically analyze your project.

1. From Jenkins' dashboard, click **New Item** and create a **Pipeline Job**.
2. Under **Build Triggers**, choose **Trigger builds remotely**. You must set a unique, secret token for this field.
3. Under **Pipeline**, make sure the parameters are set as follows:
 - **Definition:** Pipeline script from SCM
 - **SCM:** Configure your SCM. Make sure to only build your main branch. For example, if your main branch is called "main", put "*/main" under **Branches to build**.
 - **Script Path:** Jenkinsfile
4. Click **Save**.

Continue > (arrow)

- 2 Create a GitHub Webhook
- 3 Create a Jenkinsfile

Click on “Continue”

SonarQube Projects Issues Rules Quality Profiles Quality Gates Administration ? Search for projects... A

[new](#) [main](#)

Overview Issues Security Hotspots Measures Code Activity Project Settings ▾ Project Information

Analyze your project with Jenkins

Select your DevOps platform GitHub

Prerequisites

1 Create a Pipeline Job

2 Create a GitHub Webhook

Create a Webhook in your repository to trigger the Jenkins job on push. Already have a Webhook configured? [Skip this step](#).

1. Go to the [GitHub Webhook creation page for your repository](#) and enter the following information:
 - URL: Enter the following URL, replacing the values between *** as needed:

```
***JENKINS_SERVER_URL***/job/**JENKINS_JOB_NAME***/build?token=***JENKINS_BUIL
```

Copy
2. Under Which events would you like to trigger this webhook? select **Let me select individual events** and check the following:
 - Pushes
3. Click **Add webhook**.

Continue > (arrow pointing to the button)

3 Create a Jenkinsfile

Click on “Continue” again

SonarQube Projects Issues Rules Quality Profiles Quality Gates Administration ? Search for projects... A

[new](#) [main](#)

Overview Issues Security Hotspots Measures Code Activity Project Settings ▾ Project Information

Analyze your project with Jenkins

Select your DevOps platform GitHub

Prerequisites

1 Create a Pipeline Job

2 Create a GitHub Webhook

3 Create a Jenkinsfile

1. What option best describes your build?

Maven Gradle .NET Other (for JS, TS, Go, Python, PHP, ...) (arrow pointing to the Maven button)

4 You're all set!

! Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.9 (build 65466) - [LGPL v3](#) - [Community](#) - [Documentation](#) - [Plugins](#) - [Web API](#)

Select “Maven”

The screenshot shows the SonarQube dashboard with a search bar containing 'A'. Below it, the 'Project Information' section is visible. Step 3 of the tutorial is displayed, which involves creating a Jenkinsfile. A note says to replace 'Default Maven' with the name given to the Maven tool in Jenkins. A code snippet for a Jenkinsfile is shown, and a 'Copy' button is available. At the bottom, there is a 'Finish this tutorial >' button.

3 Create a Jenkinsfile

1. What option best describes your build?

Maven Gradle .NET Other (for JS, TS, Go, Python, PHP, ...)

2. Create a `Jenkinsfile` file in your repository and paste the following code:

Make sure to replace **Default Maven** with the name you gave to your Maven tool in Jenkins.

```
node {  
    stage('SCM') {  
        checkout scm  
    }  
    stage('SonarQube Analysis') {  
        def mvn = tool 'Default Maven';  
        withSonarQubeEnv() {  
            sh "${mvn}/bin/mvn clean verify sonar:sonar -Dsonar.projectKey=new"  
        }  
    }  
}
```

Copy

Finish this tutorial >

4 You're all set!

Click on “Finish this Tutorial”, And then click on the “A” and select “My Account”

The screenshot shows the SonarQube user profile page for 'Administrator'. The top navigation bar has a green box labeled 'A'. The tabs at the top are Profile (selected), Security, Notifications, and Projects. The 'Security' tab is highlighted with an orange arrow. The main content area displays sections for Login (admin), Groups (sonar-administrators, sonar-users), SCM Accounts (admin), and Preferences. In the Preferences section, there is a 'Enable Keyboard Shortcuts' toggle switch.

A

Administrator

Profile Security Notifications Projects

Login
admin

Groups
sonar-administrators
sonar-users

SCM Accounts
admin

Preferences

Enable Keyboard Shortcuts

Some actions can be performed using keyboard shortcuts. If you do not want to use these shortcuts, you can disable them here (this won't disable navigation shortcuts, which include the arrows, escape, and enter keys). For a list of available keyboard shortcuts, use the question mark shortcut (hit `?` on your keyboard).

Click on “Security”

Tokens

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

Generate Tokens

Name	Type	Expires in
Enter Token Name	Select Token Type	30 days

No tokens

Enter a new password

All fields marked with * are required

Old Password *

New Password *

Confirm Password *

Update

Give a name for the token, I will give it the name “**new**” and for type, select “**Global Analysis token**” and for Expires in, select “**30 days**”.

Tokens

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

Generate Tokens

Name	Type	Expires in
new	Global Analysis Token	30 days

No tokens

Enter a new password

All fields marked with * are required

Old Password *

New Password *

Confirm Password *

Update

Click on “Generate”

The screenshot shows the SonarQube interface with a navigation bar at the top. A green box highlights the "A" icon and the word "Administrator". Below the navigation bar, there's a "Tokens" section. It contains a "Generate Tokens" form with fields for Name (Enter Token Name), Type (Select Token Type), and Expires in (30 days). A "Generate" button is present. A yellow message box displays: "New token 'new' has been created. Make sure you copy it now, you won't be able to see it again!" Below this, a "Copy" button is shown next to the token value: "sqa_e00072dd4f12c9a35b12a22970409c156a3bbdf4". A table lists the generated token: "Name: new, Type: Global, Project: (empty), Last use: Never, Created: December 14, 2024, Expiration: January 12, 2025". A "Revoke" button is at the end of the row. At the bottom, there's a "Enter a new password" section with fields for Old Password and New Password.

Copy the token and save it somewhere: sqa_d2fe69fec824a0b1e5dc5cb671c6389e1bb11488

This will be used for authenticating Jenkins.

PART 4: Install SonarQube Plugin

Now, go back to Jenkins GUI and click on Dashboard

Before you can start using SonarQube in your Jenkins pipeline, you need to install the SonarQube plugin.

- Go to **Manage Jenkins > Manage Plugins > Available.**
- Search for “**SonarQube Scanner**” and install the plugin.

The screenshot shows the Jenkins dashboard. On the left, there are links for "New Item", "Build History", "Manage Jenkins" (which is highlighted with an orange arrow), and "My Views". In the center, there's a table for build status: "S" (Success), "W" (Warning), "Name" (demo), "Last Success" (1 hr 18 min #2), "Last Failure" (N/A), and "Last Duration" (31 sec). At the bottom, there are sections for "Build Queue" (No builds in the queue) and "Build Executor Status" (0/2).

Click on “Manage Jenkins”

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'Build History', 'Manage Jenkins' (which is highlighted), and 'My Views'. The main area is titled 'Manage Jenkins' and contains sections for 'System Configuration' (with 'System', 'Tools', 'Nodes', 'Clouds', and 'Appearance' options), 'Security' (with 'Security' and 'Users' options), and a 'Build Queue' and 'Build Executor Status' summary. At the top right, there are buttons for 'Set up agent', 'Set up cloud', and 'Dismiss'. A message at the top says: 'Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.' Below the main content, there's a system tray showing weather, search, and other icons. The status bar at the bottom right shows the date and time.

Click on “Plugins”

The screenshot shows the Jenkins Plugin Manager page. The sidebar on the left has links for 'Updates' (which is highlighted), 'Available plugins' (with a red arrow pointing to it), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main content area shows a message: 'No updates available' with a checkmark icon. At the bottom, a note says: 'Disabled rows are already upgraded, awaiting restart. Shaded but selectable rows are in progress or failed.' The status bar at the bottom right shows the date and time.

Click on “Available Plugins”

A screenshot of the Jenkins plugin manager interface. On the left, a sidebar shows navigation links: 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area is titled 'Plugins' and contains a search bar with the placeholder 'Search available plugins'. Below the search bar is a table with columns 'Install', 'Name ↓', and 'Released'. The table lists several plugins:

Install	Name ↓	Released
<input type="checkbox"/>	Command Agent Launcher 116.vd85919c54a_d6	1 mo 2 days ago
<input type="checkbox"/>	Agent Management	
	Allows agents to be launched using a specified command.	
<input type="checkbox"/>	Oracle Java SE Development Kit Installer 80.v8a_dee33ed6f0	4 mo 10 days ago
	Allows the Oracle Java SE Development Kit (JDK) to be installed via download from Oracle's website.	
<input type="checkbox"/>	SSH server 3.330.vc866a_8389b_58	6 mo 8 days ago
	Adds SSH server functionality to Jenkins, exposing CLI commands through it.	
<input type="checkbox"/>	Authentication Tokens API 1.119.v50285141b_7e1	5 mo 9 days ago
	This plugin provides an API for converting credentials into authentication tokens in Jenkins.	
<input type="checkbox"/>	Git server 126.v0d945d8d2b_39	6 mo 20 days ago
	git Library plugins (for use by other plugins)	
	Allows Jenkins to act as a Git server.	
<input type="checkbox"/>	Docker Commons 445.v6b_646c962a_94	1 mo 7 days ago
	Library plugins (for use by other plugins) docker	
	Provides the common shared functionality for various Docker-related plugins.	
<input type="checkbox"/>	JavaScript GUI Lib: ACE Editor bundle 1.1	8 yr 9 mo ago
	JavaScript GUI Lib: ACE Editor bundle plugin.	
	<small>This plugin is deprecated. In general, this means that it is either obsolete, no longer being developed, or may no longer work. Learn more</small>	

Search for “sonar”

A screenshot of the Jenkins plugin manager interface after searching for 'sonar'. The search bar at the top contains the text 'sonar'. The main area is titled 'Plugins' and contains a table with columns 'Install', 'Name ↓', and 'Released'. The table lists three plugins:

Install	Name ↓	Released
<input type="checkbox"/>	SonarQube Scanner 2.17.3	26 days ago
	External Site/Tool Integrations Build Reports	
	This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	
<input type="checkbox"/>	Sonar Quality Gates 328.vf4369b_da_d3c2	1 mo 27 days ago
	Library plugins (for use by other plugins) analysis Other Post-Build Actions	
	Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality Gates status is different than "Passed")	
<input type="checkbox"/>	Quality Gates 2.5	8 yr 7 mo ago
	Fails the build whenever the Quality Gates criteria in the Sonar analysis aren't met (the project Quality Gates status is different than "Passed")	
	<small>Warning: This plugin version may not be safe to use. Please review the following security notices:</small>	
	<ul style="list-style-type: none">Credentials transmitted in plain text	
<input type="checkbox"/>	Sonargraph Integration 5.0.2	1 yr 6 mo ago
	External Site/Tool Integrations Build Reports Other Post-Build Actions	
	This plugin integrates Sonargraph functionality into Jenkins, for Sonargraph versions 9 and 10	
<input type="checkbox"/>	CodeSonar 3.5.0	1 yr 1 mo ago
	DevOps Build Notifiers Build Reports Other Post-Build Actions	

Check “SonarQube Scanner”

Dashboard > Manage Jenkins > Plugins

Plugins

Available plugins

Search: sonar

Install	Name	Released
<input checked="" type="checkbox"/>	SonarQube Scanner 2.17.3 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	26 days ago
<input type="checkbox"/>	Sonar Quality Gates 328.vf4369b_da_d3c2 Library plugins (for use by other plugins) analysis Other Post-Build Actions Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality Gates status is different than "Passed") Warning: This plugin version may not be safe to use. Please review the following security notices: • Credentials transmitted in plain text	1 mo 27 days ago
<input type="checkbox"/>	Quality Gates 2.5 Fails the build whenever the Quality Gates criteria in the Sonar analysis aren't met (the project Quality Gates status is different than "Passed")	8 yr 7 mo ago
<input type="checkbox"/>	Sonargraph Integration 5.0.2 External Site/Tool Integrations Build Reports Other Post-Build Actions This plugin integrates Sonargraph functionality into Jenkins, for Sonargraph versions 9 and 10	1 yr 6 mo ago
<input type="checkbox"/>	CodeSonar 3.5.0 DevOps Build Notifiers Build Reports Other Post-Build Actions	1 yr 1 mo ago

Click on “Install”

Dashboard > Manage Jenkins > Plugins

Plugins

Available plugins

Plugin	Status
MATRIX AUTHORIZATION STRATEGY	Success
PAM AUTHENTICATION	Success
LDAP	Success
EMAIL EXTENSION	Success
MAILER	Success
THEME MANAGER	Success
DARK THEME	Success
LOADING PLUGIN EXTENSIONS	Success
Pipeline: REST API	Success
Pipeline: Stage View	Success
LOADING PLUGIN EXTENSIONS	Success
JavaMail API	Success
ECLIPSE TEMURIN INSTALLER	Success
LOADING PLUGIN EXTENSIONS	Success
Javadoc	Success
JSch Dependency	Success
Maven Integration	Success
LOADING PLUGIN EXTENSIONS	Success
SonarQube Scanner	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

PART 5: Add SonarQube Scanner to Jenkins on GUI

- Go to **Manage Jenkins > Tools**.
- Scroll to **SonarQube Scanner Installation** and click **Add SonarQube Scanner**.
- Provide a **name** and **installation method**.

Click on “Manage Jenkins” again

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'Build History', 'Manage Jenkins' (which is selected and highlighted in blue), and 'My Views'. The main area is titled 'Manage Jenkins' and contains a message about building on the built-in node. Below this, the 'System Configuration' section is visible, featuring several categories: 'System' (with a gear icon), 'Tools' (with a wrench icon, highlighted by an orange arrow), 'Nodes' (with a monitor icon), 'Clouds' (with a cloud icon), 'Plugins' (with a gear icon), 'Appearance' (with a paintbrush icon), and 'Credential Providers' (with a key icon). Each category has a brief description below it.

Click on “Tools” and scroll down to “SonarQube Scanner Installation”

The screenshot shows the 'Tools' configuration page under 'Manage Jenkins'. It includes sections for 'SonarScanner for MSBuild installations' (with an 'Add SonarScanner for MSBuild' button) and 'SonarQube Scanner installations' (with an 'Add SonarQube Scanner' button, highlighted by an orange arrow). Other sections shown are 'Ant installations' (with an 'Add Ant' button) and 'Maven installations' (with a dropdown menu showing 'Edited'). At the bottom are 'Save' and 'Apply' buttons.

Under “SonarQube Scanner Installations”, click on “Add SonarQube Scanner”

Dashboard > Manage Jenkins > Tools

Add SonarScanner for Jenkins

SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner

Name Required X

Install automatically ?

Install from Maven Central

Version X

Add Installer ▼

Add SonarQube Scanner

Save Apply

We will give it the name “**sonar-scanner**”

Dashboard > Manage Jenkins > Tools

SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner

Name X

Install automatically ?

Install from Maven Central

Version X

Add Installer ▼

Add SonarQube Scanner

Ant installations

Save Apply

Click on “**Save**”

The screenshot shows the Jenkins 'Manage Jenkins' page. In the top left, there's a sidebar with links: 'New Item', 'Build History', 'Manage Jenkins' (which is highlighted in blue), and 'My Views'. The main content area has a title 'Manage Jenkins' and a sub-section 'System Configuration'. It contains several configuration items with icons and descriptions:

- System**: Configure global settings and paths.
- Tools**: Configure tools, their locations and automatic installers.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance**: Configure the look and feel of Jenkins.
- Security**: Secure Jenkins; define who is allowed to access/use the system.
- Credentials**: Configure credentials.
- Credential Providers**: Configure the credential providers and types.
- Users**: Create/delete/modify users that can log in to this Jenkins.

A banner at the top right says 'Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).'. There are three buttons at the top right: 'Set up agent', 'Set up cloud', and 'Dismiss'.

PART 6: Configure SonarQube Server on Jenkins

After installing the plugin, you need to configure your SonarQube server in Jenkins.

- Go to **Manage Jenkins -> System**.
- Scroll down to **SonarQube Servers** and click **Add SonarQube**.
- Provide your **SonarQube server URL** and **authentication token**.

Click on “**System**” and scroll down to “**SonarQube Server**”

Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables

SonarQube installations

List of SonarQube installations

Add SonarQube 

Metrics

Access keys ?

Add new access key

Pipeline Speed / Durability

Default Speed / Durability Level ?

None: use pipeline default (Maximum survivability/durability but slowest) 

Writes data with every step, using atomic writes for integrity. Provides maximum ability to retain running pipeline data and resume in the event of a Jenkins failure.

Usage Statistics

Save Apply

Click on “Add SonarQube”

Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables

SonarQube installations

List of SonarQube installations

Name 

! This property is mandatory.

Server URL
Default is <http://localhost:9000>

Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.

- none - 

+ Add

Advanced 

Save Apply

We will give it the name “**SonarQube**”, then copy the IP address from the SonarQube browser

The screenshot shows the SonarQube Security interface. At the top, there's a navigation bar with links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, Profile, Security (which is underlined), Notifications, and Projects. A search bar at the top right contains the placeholder "Search for projects...". On the left, a sidebar for the user "Administrator" is visible. The main content area is titled "Tokens" and contains instructions about using tokens for security instead of user logins. It has a "Generate Tokens" section where a new token named "new" is created with a lifetime of 30 days. A success message says "New token 'new' has been created. Make sure you copy it now, you won't be able to see it again!" followed by a "Copy" button and the token value "sqa_e00072dd4f12c9a35b12a22970409c156a3bbdf4". Below this is a table showing the token details:

Name	Type	Project	Last use	Created	Expiration
new	Global		Never	December 14, 2024	January 12, 2025

At the bottom of the token section is a "Revoke" button. Below the token section is a form for entering a new password, with fields for "Old Password" and "New Password".

Copy the IP address and

The screenshot shows the Jenkins System configuration page under "Manage Jenkins > System". The title is "SonarQube servers". It includes a note that checked boxes will inject SonarQube server configurations as environment variables. There is a checkbox for "Environment variables". Below this is a "SonarQube installations" section with a "List of SonarQube installations" table. The table has one row with the following columns:

Name	Server URL	Server authentication token
SonarQube	http://54.172.236.64:9000/	- none -

An orange arrow points from the "Server URL" field to the "Save" button at the bottom of the form. Another orange arrow points from the "Save" button to the "Save" button in the bottom right corner of the table row.

Paste on the ***authentication token*** Jenkins GUI and click on “Save”

PART 7: Create Server Authentication token

For the “server Authentication Token”, Go to “Manage Jenkins” on another window

The screenshot shows the Jenkins Manage Jenkins interface. In the left sidebar, 'Manage Jenkins' is selected. The main area is titled 'System Configuration' and contains several sections: 'Build Queue' (No builds in the queue), 'Build Executor Status' (0/2), 'System' (Configure global settings and paths), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), 'Tools' (Configure tools, their locations and automatic installers), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand), 'Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), and 'Appearance' (Configure the look and feel of Jenkins). A red arrow points to the 'Credentials' section under the 'Security' heading.

Click on “Credentials”

Credentials

The screenshot shows the Jenkins Credentials page. It displays a table with columns: T, P, Store ↓, Domain, ID, and Name. Below the table, it says 'Stores scoped to Jenkins'. A red arrow points to the 'System' icon in the 'Store' column of the first row, which is labeled '(global)'. There are also icons for S, M, and L.

Click on “System”

System

The screenshot shows the Jenkins System page. It displays a table with columns: Domain ↓ and Description. A single entry is shown: 'Global credentials (unrestricted)' with a description: 'Credentials that should be available irrespective of domain specification to requirements matching.' A red arrow points to the 'Global credentials (unrestricted)' entry. There are also icons for S, M, and L.

Click on “Global Credential”

Global credentials (unrestricted)

+ Add Credentials

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

ID	Name	Kind	Description
This credential domain is empty. How about adding some credentials?			

Icon: S M L

Click on “Add 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 ?

Treat username as secret ?

Password ?

ID ?

Description ?

Create

On “Kind”, select “Secret Text”, on “Secret” paste the key we copied from SonarQube, for “Description” enter “sonar”

New credentials

Kind
Secret text

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

Secret
.....

ID ?

Description ?
sonar

Create

Click on “Create”

Global credentials (unrestricted)

+ Add Credentials

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

ID	Name	Kind	Description
 a3d26812-66ab-46bd-aad7-5696ff5166d4	sonar	Secret text	sonar

Icon: S M L

Go back to the **Jenkins -> System page and refresh it**. On “Server Authentication Token” you will now select “sonar”

Not secure 54.172.236.64:8080/manage/configure

Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables

SonarQube installations

List of SonarQube installations

Name	<input type="text" value="SonarQube"/>	X
Server URL	Default is http://localhost:9000	
	<input type="text" value="http://54.172.236.64:9000/"/>	
Server authentication token	SonarQube authentication token. Mandatory when anonymous access is disabled.	
	<input type="text" value="sonar"/>	
+ Add Advanced ▾		
Save Apply		

Click on “Save”

Jenkins

Dashboard >

+ New Item All + Add description

Build History Manage Jenkins My Views

Build Queue: No builds in the queue. Build Executor Status: 0/2

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	✖	demo	1 hr 46 min #2	N/A	31 sec

Icon: S M L ⚡

Click on the Pipeline “demo”

The screenshot shows the Jenkins interface for a pipeline named "demo". At the top, there's a navigation bar with links for "Dashboard", "Status", "Changes", "Build Now", "Configure", "Delete Pipeline", "Full Stage View", "Stages", "Rename", and "Pipeline Syntax". A red arrow points to the "Configure" link. Below the navigation is a "Stage View" section showing three stages: "Declarative: Tool Install" (22s), "Git Checkout" (566ms), and "Code Compile" (6s). A tooltip indicates an average stage time of 22s. To the right is a "Permalinks" section listing the last four builds: #2 (Dec 14 21:51, No Changes), #1 (Dec 14 11:28 PM), #2 (Dec 14 2:51 AM), and #1 (Dec 14 2024). At the bottom right, there are links for "REST API" and "Jenkins 2.489".

Then click on “Configure”

The screenshot shows the "Configuration" page for the "demo" pipeline. On the left, there's a sidebar with "General", "Build Triggers", "Advanced Project Options", and "Pipeline" (which has a red arrow pointing to it). The main area is titled "General" and contains a "Description" field with a placeholder "Description" and a "Plain text Preview" section. This section includes checkboxes for "Discard old builds", "Do not allow concurrent builds", "Do not allow the pipeline to resume if the controller restarts", "GitHub project", "Pipeline speed/durability override", "Preserve stashes from completed builds", "This project is parameterized", and "Throttle builds". At the bottom are "Save" and "Apply" buttons.

Scroll down to “Pipeline”

The screenshot shows the Jenkins Pipeline configuration page for a project named "demo". The "Pipeline" tab is selected under "Advanced Project Options". The "Definition" dropdown is set to "Pipeline script". The main area contains a code editor with the following Groovy script:

```

6   |       maven 'maven3'
7   |
8   |
9   |
10  | stages {
11  |   stage('Git Checkout') {
12  |     steps {
13  |       git 'https://github.com/ebotsmith2000/CICD-Pipeline.git'
14  |     }
15  |   stage('Code Compile') {
16  |     steps {
17  |       sh 'mvn clean compile'
18  |     }
19  |   }
20  | }
21  |
22  }

```

Below the code editor, there is a checkbox labeled "Use Groovy Sandbox" which is checked. At the bottom are "Save" and "Apply" buttons.

We have configured another tool called “**Sonar-Scanner**” for SonarQube. We have to define it as well but we cannot define it under tools. So, we have to define it under “**environment variable**”. We will give the variable the name “**SCANNER_HOME**”, it will be the HOME path of scanner. There we have to define the tool using the name we gave to the SonarQube Scanner, that is “**sonar-scanner**” under “**tools**”.

The screenshot shows the Jenkins Pipeline configuration page for a project named "demo". The "Pipeline" tab is selected under "Advanced Project Options". The "Definition" dropdown is set to "Pipeline script". The main area contains a code editor with the following Groovy script:

```

1 * pipeline {
2   agent any
3
4   tools{
5     jdk 'jdk'
6     maven 'maven3'
7   }
8   environment{
9     SCANNER_HOME=tool 'sonar-scanner'
10  }
11
12  stages {
13    stage('Git Checkout') {
14      steps {
15        git 'https://github.com/ebotsmith2000/CICD-Pipeline.git'
16      }
17    }

```

Below the code editor, there is a checkbox labeled "Use Groovy Sandbox" which is checked. At the bottom are "Save" and "Apply" buttons.

SonarQube has been defined as an environment variable.

The screenshot shows the Jenkins Pipeline configuration page for a project named "demo". The "Pipeline" tab is selected. In the "Script" section, the following Groovy code is displayed:

```
1+ pipeline {
2   agent any
3
4   tools{
5     jdk 'jdk'
6     maven 'maven3'
7   }
8   environment{
9     SCANNER_HOME=tool 'sonar-scanner'
10  }
11
12  stages {
13    stage('Git Checkout') {
14      steps {
15        git 'https://github.com/ebotsmith2000/CICD-Pipeline.git'
16      }
17    }
18  }
19}
```

Below the script, there is a checked checkbox labeled "Use Groovy Sandbox". At the bottom of the page are "Save" and "Apply" buttons.

Now, let us add the stage for the SonarQube scanner. First, go to the page for “[Pipeline Syntax](#)”

The screenshot shows the Pipeline Syntax Snippet Generator page. The "git" step is selected in the "Sample Step" dropdown. An orange arrow points to this dropdown. The "git" step configuration includes:

- Repository URL: `https://github.com/ebotsmith2000/CICD-Pipeline.git`
- Branch: `master`
- Credentials: `- none -`
- Include in polling?

Refresh the page and under “sample step”, select the item related to SonarQube. In this case it is “**withSonarQubeEnv**”

Snippet Generator

- Declarative Directive Generator
- Declarative Online Documentation
- Steps Reference
- Global Variables Reference
- Online Documentation
- Examples Reference
- IntelliJ IDEA GDSL

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step

withSonarQubeEnv: Prepare SonarQube Scanner environment

withSonarQubeEnv ?

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled. Will default to the one defined in the SonarQube installation.

- none -

+ Add

Generate Pipeline Script

Global Variables

Click on “Generate Pipeline Script”

Snippet Generator

- Declarative Directive Generator
- Declarative Online Documentation
- Steps Reference
- Global Variables Reference
- Online Documentation
- Examples Reference
- IntelliJ IDEA GDSL

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step

withSonarQubeEnv: Prepare SonarQube Scanner environment

withSonarQubeEnv ?

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled. Will default to the one defined in the SonarQube installation.

- none -

+ Add

Generate Pipeline Script

```
withSonarQubeEnv {
    // some block
}
```

We have to copy this and use on the definition of the stage in our configuration of SonarQube

Dashboard > demo > Configuration

Configure

Pipeline

General

Build Triggers

Advanced Project Options

Pipeline

Definition

Pipeline script

```

14    steps {
15        git 'https://github.com/ebotsmith2000/CICD-Pipeline.git'
16    }
17 }
18 stage('Code Compile') {
19     steps {
20         sh 'mvn clean compile'
21     }
22 }
23 stage('SonarQube Analysis') {
24     steps {
25         withSonarQubeEnv {'SonarQube'}
26     }
27 }
28 }
29 }
```

Use Groovy Sandbox

Pipeline Syntax

Save **Apply**

We got the name “**SonarQube**” by going to “**Manage Jenkins**” -> “**Systems**” and scroll down to “**SonarQube Servers**” as shown below.

Dashboard > Manage Jenkins > System >

Disable deferred wipeout on this node

Disk Space Monitoring Thresholds

Environment variables

Tool Locations

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables

SonarQube installations

List of SonarQube installations

Name	<input type="text" value="SonarQube"/>	X
Server URL	Default is http://localhost:9000	
	<input type="text" value="http://54.172.236.64:9000/"/>	
Server authentication token	SonarQube authentication token. Mandatory when anonymous access is disabled.	
	<input type="text" value="sonar"/>	

Save **Apply**

On the “**Configuration**” page and continue

Not secure 54.172.236.64:8080/job/demo/configure

Dashboard > demo > Configuration Advanced ▾

Configure

- General
- Build Triggers
- Advanced Project Options**
- Pipeline

Pipeline

Definition Pipeline script

```

18 +
19 +
20   stage('Code Compile') {
21     steps {
22       sh 'mvn clean compile'
23     }
24   }
25   stage('SonarQube Analysis') {
26     steps {
27       withSonarQubeEnv ('SonarQube'){
28         sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=new \
29           -Dsonar.java.binaries=. \
30           -Dsonar.projectKey=new ...
31       }
32     }
33   }
34 }
```

Use Groovy Sandbox ?

Pipeline Syntax

Save **Apply**

Pipeline

Definition Pipeline script

```

18 +
19 +
20   stage('Code Compile') {
21     steps {
22       sh 'mvn clean compile'
23     }
24   }
25   stage('SonarQube Analysis') {
26     steps {
27       withSonarQubeEnv ('SonarQube'){
28         sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=new \
29           -Dsonar.java.binaries=. \
30           -Dsonar.projectKey=new ...
31       }
32     }
33 }
```

Use Groovy Sandbox ?

Pipeline Syntax

Save **Apply**

```

pipeline {
    agent any

    tools{
        jdk 'jdk'
        maven 'maven3'
    }
    environment{
        SCANNER_HOME= tool 'sonar-scanner'
    }

    stages {
        stage('Git Checkout') {
            steps {
                git 'https://github.com/ebotsmith2000/jenkins-pipeline.git'
            }
        }
        stage('Code Compile') {
            steps {
                sh 'mvn clean compile'
            }
        }
        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('SonarQube'){
                    sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=new \
-Dsonar.java.binaries=. \
-Dsonar.projectKey=new '''
                }
            }
        }
    }
}

```

Click on “Save”

The screenshot shows the Jenkins Pipeline configuration page for a pipeline named "demo". The pipeline script is displayed at the top. Below the script, the "Stage View" section provides a summary of the pipeline's stages and their execution times. The sidebar on the left contains various navigation links and a "Build Now" button, which is highlighted with a red arrow. The bottom left corner shows the build history, and the bottom right corner displays the Jenkins version.

Now, let us build the pipeline by clicking on “Build Now”

The screenshot shows the Jenkins Pipeline interface. On the left, there's a sidebar with options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, SonarQube, Stages, Rename, and Pipeline Syntax. The main area is titled "Stage View" and shows two stages: "Declarative: Tool Install" (11s), "Git Checkout" (532ms), "Code Compile" (5s), and "SonarQube Analysis" (13s). Below this, a table provides detailed timing for each stage across two builds (#5 and #2). The SonarQube Quality Gate section indicates a "Passed" status with "Success" server-side processing. The Builds section lists recent builds: #5 (1:28 PM), #2 (2:51 AM), December 14, 2024, and #1 (11:28 PM).

The build is successful. Go now to SonarQube browser and refresh it. Then click on the “Project” tab

The screenshot shows the SonarQube Project page. The top navigation bar includes Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and a search bar. The main content area displays a summary of the project's quality gate status (Passed) and various metrics: Bugs (0 A), Vulnerabilities (0 A), Hotspots Reviewed (0.0% E), Code Smells (0 A), Coverage (0.0% O), Duplications (0.0% G), and Lines (79 X). On the left, there are filters for Quality Gate (Passed, Failed), Reliability (A-E rating), Security (A-E rating), and Security Review (A-F rating). A note at the bottom states: "Embedded database should be used for evaluation purposes only. The embedded database will not scale; it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine." The footer includes the SonarQube™ technology information and links to Documentation and Plugins.

You can see that it passed and we have scanned 79 lines. Static Code Analysis is done. So, we go to the next step, that is “Code Build”.

STEP 12: Configure Maven for Code Build

To do Code Build, we have to use “**Maven**”. We have already configured Maven. Let us go to the Jenkins GUI and add a stage for “**Code Build**”.

The screenshot shows the Jenkins Pipeline Stage View for the 'demo' pipeline. On the left, there's a sidebar with various pipeline management options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, SonarQube, Stages, Rename, and Pipeline Syntax. The main area is titled 'Stage View' and displays a grid of stages for two builds (#5 and #2). The columns represent different stages: Declarative: Tool Install, Git Checkout, Code Compile, and SonarQube Analysis. Each stage has a color-coded bar indicating its duration: 11s, 532ms, 5s, and 13s respectively. Below the grid, there's a summary of average stage times: 144ms, 498ms, 4s, and 13s. To the right, there's a 'SonarQube Quality Gate' section showing a new build (#5) has passed, with a green 'Passed' button and a 'Success' status message. A 'Permalinks' section lists recent builds: #5 (1:28 PM), #2 (2:51 AM), December 14, 2024, and #1 (11:28 PM).

Click on “Configure”

The screenshot shows the Jenkins Pipeline Configuration page for the 'demo' pipeline. The top navigation bar includes links for Dashboard, demo, Configuration, and other Jenkins global options. The main configuration area is divided into 'Configure' and 'General' tabs. Under 'Configure', there are sections for General, Build Triggers, Advanced Project Options, and Pipeline. The 'General' tab is active, showing an 'Enabled' switch which is turned on. In the 'General' section, there's a 'Description' field with a rich text editor and a 'Plain text' preview link. Below the editor are several configuration options with checkboxes: 'Discard old builds', 'Do not allow concurrent builds', 'Do not allow the pipeline to resume if the controller restarts', 'GitHub project', 'Pipeline speed/durability override', 'Preserve stashes from completed builds', 'This project is parameterized', and 'Throttle builds'. At the bottom of the configuration section are 'Save' and 'Apply' buttons.

Scroll down to “Pipeline”

Dashboard > demo > Configuration

Advanced

Configure

- General
- Build Triggers
- Advanced Project Options**
- Pipeline

Pipeline

Definition

Pipeline script

```

1 pipeline {
2     agent any
3
4     tools{
5         jdk 'jdk'
6         maven 'maven3'
7     }
8     environment{
9         SCANNER_HOME=tool 'sonar-scanner'
10    }
11
12    stages {
13        stage('Git Checkout') {
14            steps {
15                git 'https://github.com/ebotsmith2000/CICD-Pipeline.git'
16            }
17        }
18    }
19
20}

```

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

Let us add another stage for “Code Build”

Dashboard > demo > Configuration

Advanced

Configure

- General
- Build Triggers
- Advanced Project Options**
- Pipeline

Pipeline

Definition

Pipeline script

```

1 pipeline {
2     agent any
3
4     tools{
5         jdk 'jdk'
6         maven 'maven3'
7     }
8     environment{
9         SCANNER_HOME=tool 'sonar-scanner'
10    }
11
12    stages {
13        stage('Git Checkout') {
14            steps {
15                git 'https://github.com/ebotsmith2000/CICD-Pipeline.git'
16            }
17        }
18
19        stage('SonarQube Analysis') {
20            steps {
21                withSonarQubeEnv ('SonarQube'){
22                    sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=new \
23                    -Dsonar.java.binaries=... \
24                    -Dsonar.projectKey=new ...'''
25                }
26            }
27        }
28
29        stage('Code Build') {
30            steps {
31                sh 'mvn clean install'
32            }
33        }
34    }
35
36}
37
38}
39

```

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

Pipeline

Definition

Pipeline script

Script ?

```
23 v
24 v
25 v
26 v
27 v
28 v
29 v
30 v
31 v
32 v
33 v
34 v
35 v
36 v
37 v
38 v
39 v
        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv ('SonarQube'){
                    sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=new \
-Dsonar.java.binaries=. \
-Dsonar.projectKey=new ''
                }
            }
        }
        stage('Code Build') {
            steps {
                sh 'mvn clean install'
            }
        }
    }
```

Use Groovy Sandbox ?

Pipeline Syntax

Save

Apply

```
pipeline {
    agent any

    tools{
        jdk 'jdk'
        maven 'maven3'
    }
    environment{
        SCANNER_HOME= tool 'sonar-scanner'
    }

    stages {
        stage('Git Checkout') {
            steps {
                git 'https://github.com/ebotsmith2000/jenkins-pipeline.git'
            }
        }
        stage('Code Compile') {
            steps {
                sh 'mvn clean compile'
            }
        }
        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('SonarQube'){
                    sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=new \
-Dsonar.java.binaries=. \
-Dsonar.projectKey=new ''
                }
            }
        }
    }
}
```

```

        }
        stage('Code Build') {
            steps {
                sh 'mvn clean install'
            }
        }
    }
}

```

Install will build the code. Click on “Save”

The screenshot shows the Jenkins Pipeline interface. On the left, there's a sidebar with various pipeline management options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, SonarQube, Stages, Rename, and Pipeline Syntax. The main area is titled "Stage View" and displays a table of build stages. The table has four columns: Declarative: Tool Install (11s), Git Checkout (532ms), Code Compile (5s), and SonarQube Analysis (13s). Below the table, two specific builds are shown: #5 (Dec 15 08:28) and #2 (Dec 14 21:51), both labeled "No Changes". To the left of the table, there's a "Builds" section showing a list of recent builds: #5 (1:28 PM), #2 (2:51 AM), and #1 (11:28 PM). A red arrow points from the "Filter" dropdown in this section to the build #5 entry.

Go to “**Workspace**” by clicking on **#5** → Then click on “**Workspace**” and click on “[/var/lib/jenkins/workspace/demo](#) on built-in”

The screenshot shows the Jenkins workspace for build #5. At the top, there's a breadcrumb navigation: Dashboard > demo > #5 > Allocate node : Start > Workspace > Workspace. The main area is titled "Workspace" and shows a file tree. The "Workspace" folder is highlighted with a red arrow. Inside the workspace, there are several files and folders: .git, .scannerwork, k8s, src/main/java/com/example/demoapp, target (also highlighted with a red arrow), Dockerfile, Jenkinsfile, and pom.xml. Below the file tree, there's a table of file details with columns for last modified date, size, and a "View" link. At the bottom right, there's a link to "all files in zip".

Click on “target”

The screenshot shows a Jenkins workspace page. At the top, there's a navigation bar with links: Dashboard > demo > #5 > Allocate node : Start > Workspace > Workspace. Below this is a sidebar with links: Up, Status, Console Output, and Workspace. The main area is titled "Workspace" and shows a file tree under "/ target /". The tree includes "classes/com/example/demoapp", "generated-sources/annotations", and "maven-status/maven-compiler-plugin/compile/default-compile". At the bottom right is a link to download all files in a zip archive.

Here you can see there is **no Jar file**. We will expect a Jar file when the “**Code Build**” is successful. Now let us go back to our pipeline

The screenshot shows a Jenkins pipeline job page for "demo". On the left, there's a sidebar with links: Status, Changes, Build Now (which has an orange arrow pointing to it), Configure, Delete Pipeline, Full Stage View, SonarQube, Stages, Rename, and Pipeline Syntax. The main area is titled "Stage View" and shows a table of stage times for builds #5 and #2. The table has four columns: Declarative: Tool Install, Git Checkout, Code Compile, and SonarQube Analysis. Below this is a "SonarQube Quality Gate" section showing a green "Passed" status. At the bottom, there's a "Builds" section listing recent builds: #5 (1:28 PM), #2 (2:51 AM), December 14, 2024, and #1 (11:28 PM). The status for build #5 is "new | Passed".

Click on “Build Now”, we will get another stage called “Code Build”

The screenshot shows the Jenkins Pipeline Stage View for the 'demo' pipeline. The stage view displays five stages: Declarative: Tool Install, Git Checkout, Code Compile, SonarQube Analysis, and Code Build. The 'Code Build' stage is highlighted in green, indicating success. The 'SonarQube Quality Gate' section shows a green 'Passed' status. The left sidebar includes options like 'Build Now', 'Configure', and 'Delete Pipeline'. The build history on the left shows three successful builds: #8 (Dec 15 09:18), #5 (Dec 15 08:28), and #2 (Dec 14 21:51).

The “Code Build” stage is successful. Now let us check the “Workspace” again

The screenshot shows the Jenkins workspace for build #8. The workspace directory structure is displayed, including 'target', 'classes/com/example/demoapp', 'generated-sources/annotations', 'maven-archiver', and 'maven-status/maven-compiler-plugin/compile/default-compile'. An orange arrow points to the 'target' directory, where the file 'demo-app-1.0-SNAPSHOT.jar' is visible. The file was created on Dec 15, 2024, at 2:18:50 PM, with a size of 2.86 KiB.

You can now see that a Jar file has been created. The “Code Build” is successful.

STEP 13: Install Docker and Build Docker Image

Let us go to the next step which is “**Docker Build**”. If we want to execute Docker, we need to first install Docker to our EC2 instance.

Installation of Docker in Ubuntu:

```
# Add Docker's official GPG key:  
sudo apt-get update  
sudo apt-get install ca-certificates curl  
sudo install -m 0755 -d /etc/apt/keyrings  
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc  
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Copy the above code and paste on the terminal of the instance

```
sudo apt-get update  
sudo apt-get install ca-certificates curl  
sudo install -m 0755 -d /etc/apt/keyrings  
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc  
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
sudo apt-get update  
sudo apt-get install ca-certificates curl  
sudo install -m 0755 -d /etc/apt/keyrings  
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc  
sudo chmod a+r /etc/apt/keyrings/docker.asc  
  
# Add the repository to Apt sources:  
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  
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease  
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]  
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]  
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]  
Ign:5 https://pkg.jenkins.io/debian binary/ InRelease  
Hit:6 https://pkg.jenkins.io/debian binary/ Release  
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]  
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [309 kB]  
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]  
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]  
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]  
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [11.7 kB]  
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]  
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]  
Get:16 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [7188 B]  
Get:17 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.8 kB]  
Get:18 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]  
Get:19 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]  
Fetched 912 kB in 1s (1592 kB/s)  
Reading package lists... Done  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
ca-certificates is already the newest version (20240203).  
ca-certificates set to manually installed.  
curl is already the newest version (8.5.0-2ubuntu10.5).  
curl set to manually installed.  
0 upgraded, 0 newly installed, 0 to remove and 28 not upgraded.  
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease  
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease  
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease  
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]  
Ign:5 https://pkg.jenkins.io/debian binary/ InRelease  
Hit:6 https://pkg.jenkins.io/debian binary/ Release  
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease  
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [16.6 kB]  
Fetched 65.4 kB in 1s (119 kB/s)  
Reading package lists... Done  
ubuntu@ip-172-31-31-55:~$
```

```
# Add the repository to Apt sources:
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
```

```
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
```

A screenshot of a terminal window. At the top, there's a toolbar with icons for AWS Lambda, CloudWatch Metrics, CloudWatch Logs, CloudWatch Metrics Insights, CloudWatch Metrics Dashboards, CloudWatch Metrics Metrics, and CloudWatch Metrics Metrics Insights. Below the toolbar, there's a search bar with the placeholder "Search" and a keybinding "[Alt+S]" in brackets. The main area of the terminal shows the command history and its output.

```
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Ign:5 https://pkg.jenkins.io/debian binary/ InRelease
Hit:6 https://pkg.jenkins.io/debian binary/ Release
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [309 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [11.7 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [7188 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.8 kB]
Get:18 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 kB]
Get:19 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 kB]
Fetched 912 kB in 1s (1592 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.5).
curl set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 28 not upgraded.
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:4 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Ign:5 https://pkg.jenkins.io/debian binary/ InRelease
Hit:6 https://pkg.jenkins.io/debian binary/ Release
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:8 https://download.docker.com/linux/ubuntu/noble/stable amd64 Packages [16.6 kB]
Fetched 65.4 kB in 1s (119 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-31-55:~$ 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
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu noble InRelease
Ign:6 https://pkg.jenkins.io/debian binary/ InRelease
Hit:7 https://pkg.jenkins.io/debian binary/ Release
Reading package lists... Done
ubuntu@ip-172-31-31-55:~$ H
```

i-03a828fc255dc93a5 (My Server)

PublicIPs: 54.172.236.64 PrivateIPs: 172.31.31.55

A screenshot of a terminal window. At the top, there's a toolbar with icons for CloudShell and Feedback. The main area of the terminal shows the command history and its output.

Then run the command:

To Install the latest Version:

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

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

```
ubuntu@ip-172-31-31-55:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 28 not upgraded.
Need to get 124 MB of archives.
After this operation, 445 MB of additional disk space will be used.
Do you want to continue? [y/n] ■
```

Type “Y” and press ENTER

```
Unpacking docker-compose-plugin (2.31.0-1~ubuntu.24.04~noble) ...
Selecting previously unselected package libltdl7:amd64.
Preparing to unpack .../7-libltdl7_2.4.7-7build1_amd64.deb ...
Unpacking libltdl7:amd64 (2.4.7-7build1) ...
Selecting previously unselected package libslirp0:amd64.
Preparing to unpack .../8-libslirp0_4.7.0-1ubuntu3_amd64.deb ...
Unpacking libslirp0:amd64 (4.7.0-1ubuntu3) ...
Selecting previously unselected package slirp4netns.
Preparing to unpack .../9-slirp4netns_1.2.1-1build2_amd64.deb ...
Unpacking slirp4netns (1.2.1-1build2) ...
Setting up docker-buildx-plugin (0.19.2-1~ubuntu.24.04~noble) ...
Setting up containerd.io (1.7.24-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /usr/lib/systemd/system/containerd.service.
Setting up docker-compose-plugin (2.31.0-1~ubuntu.24.04~noble) ...
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.4.0-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.4.0-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.4.0-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.3) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.8.0-1018-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-1020-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...

Service restarts being deferred:
/etc/needrestart/restart.d/dbus.service
systemctl restart networkd-dispatcher.service
systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
  ubuntu @ session #54: java[8655]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-31-55:~$ ■
```

i-03a828fc255dc93a5 (My Server)

PublicIPs: 54.172.236.64 PrivateIPs: 172.31.31.55

 CloudShell Feedback

Docker has been installed. Let us verify the installation using this command:

```
sudo docker run hello-world
```

```
ubuntu@ip-172-31-31-55:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
clec31eb5944: Pull complete
Digest: sha256:5b3cc85e16e3058003c13b7821318369dad01dac3dbb877aac3c28182255c724
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
 executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
 to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
ubuntu@ip-172-31-31-55:~$ █
```

This confirms that Docker has been installed. But incase we have to use this Docker with Jenkins; we have to give some permissions for Docker and Jenkins.

We will do this by running the command:

```
sudo usermod -aG docker jenkins
```

```
ubuntu@ip-172-31-31-55:~$ sudo usermod -aG docker jenkins
ubuntu@ip-172-31-31-55:~$ █
```

We have given permission. So, if we want to use them, we have to restart Jenkins and Docker, else it won't work.

Let us first restart Jenkins using the command:

```
sudo systemctl restart jenkins
```

```
ubuntu@ip-172-31-31-55:~$ sudo systemctl restart jenkins
█
```

Now, let us restart Docker using the command:

```
sudo systemctl restart docker
```

```
ubuntu@ip-172-31-31-55:~$ sudo systemctl restart jenkins
ubuntu@ip-172-31-31-55:~$ sudo systemctl restart docker
ubuntu@ip-172-31-31-55:~$
```

Verify Docker status using the command:

```
sudo systemctl status docker
```

```
ubuntu@ip-172-31-44-137:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-01-18 18:02:28 UTC; 1min 5s ago
       Docs: https://docs.docker.com
   Main PID: 8995 (dockerd)
      Tasks: 9
        Memory: 22.0M (peak: 22.3M)
         CPU: 32.5ms
      CGroup: /system.slice/docker.service
              └─8995 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jan 18 18:02:27 ip-172-31-44-137 dockerd[8995]: time="2025-01-18T18:02:27.938698546Z" level=info msg="OTEL tracing is not configured, using no-op tracer provider"
Jan 18 18:02:27 ip-172-31-44-137 dockerd[8995]: time="2025-01-18T18:02:27.938825843Z" level=info msg="detected 127.0.0.53 nameserver, assuming systemd-resolved, so using resolv.conf: /run/systemd/resolv.conf"
Jan 18 18:02:27 ip-172-31-44-137 dockerd[8995]: time="2025-01-18T18:02:27.977683254Z" level=info msg="graphdriver using prior storage driver: overlay2"
Jan 18 18:02:27 ip-172-31-44-137 dockerd[8995]: time="2025-01-18T18:02:27.978601276Z" level=info msg="Loading containers: start."
Jan 18 18:02:28 ip-172-31-44-137 dockerd[8995]: time="2025-01-18T18:02:28.078601276Z" level=info msg="Default bridge (dockero) is assigned with an IP address 172.17.0.0/16. Daemon option --bip can be used"
Jan 18 18:02:28 ip-172-31-44-137 dockerd[8995]: time="2025-01-18T18:02:28.339744805Z" level=info msg="Loading containers: done."
Jan 18 18:02:28 ip-172-31-44-137 dockerd[8995]: time="2025-01-18T18:02:28.359297379Z" level=info msg="Docker daemon" commit="38bb94dc containerd-snapshotter=false storage-driver=overlay2 version=27.5.0
Jan 18 18:02:28 ip-172-31-44-137 dockerd[8995]: time="2025-01-18T18:02:28.359382551Z" level=info msg="Daemon has completed initialization"
Jan 18 18:02:28 ip-172-31-44-137 dockerd[8995]: time="2025-01-18T18:02:28.387746628Z" level=info msg="API listen on /run/docker.sock"
Jan 18 18:02:28 ip-172-31-44-137 systemd[1]: Started docker.service - Docker Application Container Engine.
lines 1-32/32 (END)
```

i-0201fc151bea82ac9 (My Server)

PublicIPs: 54.198.237.25 PrivateIPs: 172.31.44.137

Go to the Jenkins GUI and refresh the page

The screenshot shows the Jenkins interface. In the top navigation bar, there is a link labeled "Dashboard". Below the dashboard, the "Workspace" section is displayed, showing a file tree for a build step named "target". The tree includes "classes/com/example/demoapp", "generated-sources/annotations", "maven-archiver", and "demo-app-1.0-SNAPSHOT.jar". A timestamp at the bottom right indicates the file was last modified on Dec 15, 2024, at 2:18:50 PM.

Click on “Dashboard”

The screenshot shows the Jenkins dashboard at 54.172.236.64:8080. The 'Manage Jenkins' link in the left sidebar is highlighted with an orange arrow. The main area displays a build history table for a 'demo' job, showing its last success was 1 hr 8 min ago and its last failure was N/A.

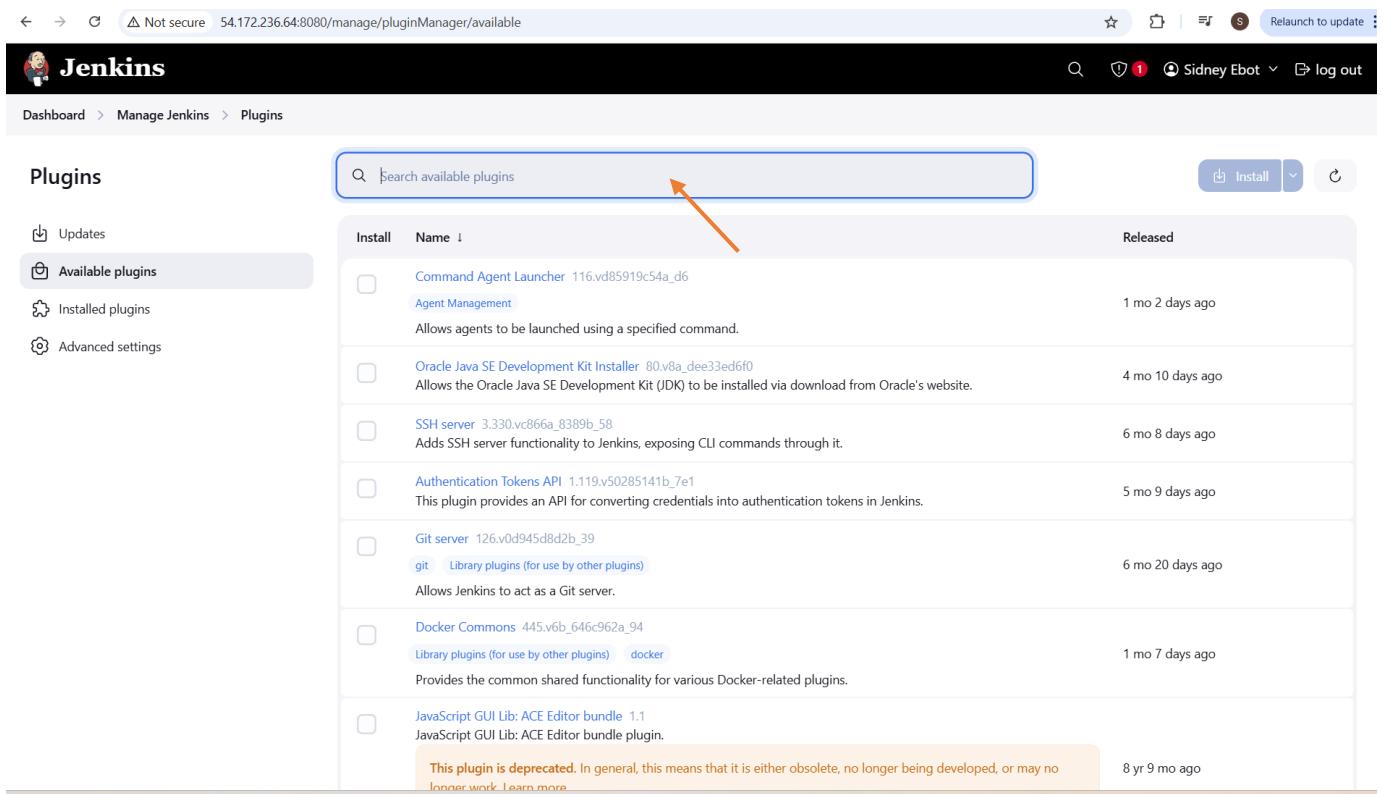
Click on “Manage Jenkins”

The screenshot shows the 'Manage Jenkins' page at 54.172.236.64:8080/manage/. The 'Plugins' section is highlighted with an orange arrow. Other sections shown include System Configuration (System, Tools, Nodes, Clouds), Security (Security, Credentials, Users), and Appearance.

Click on “Plugins”

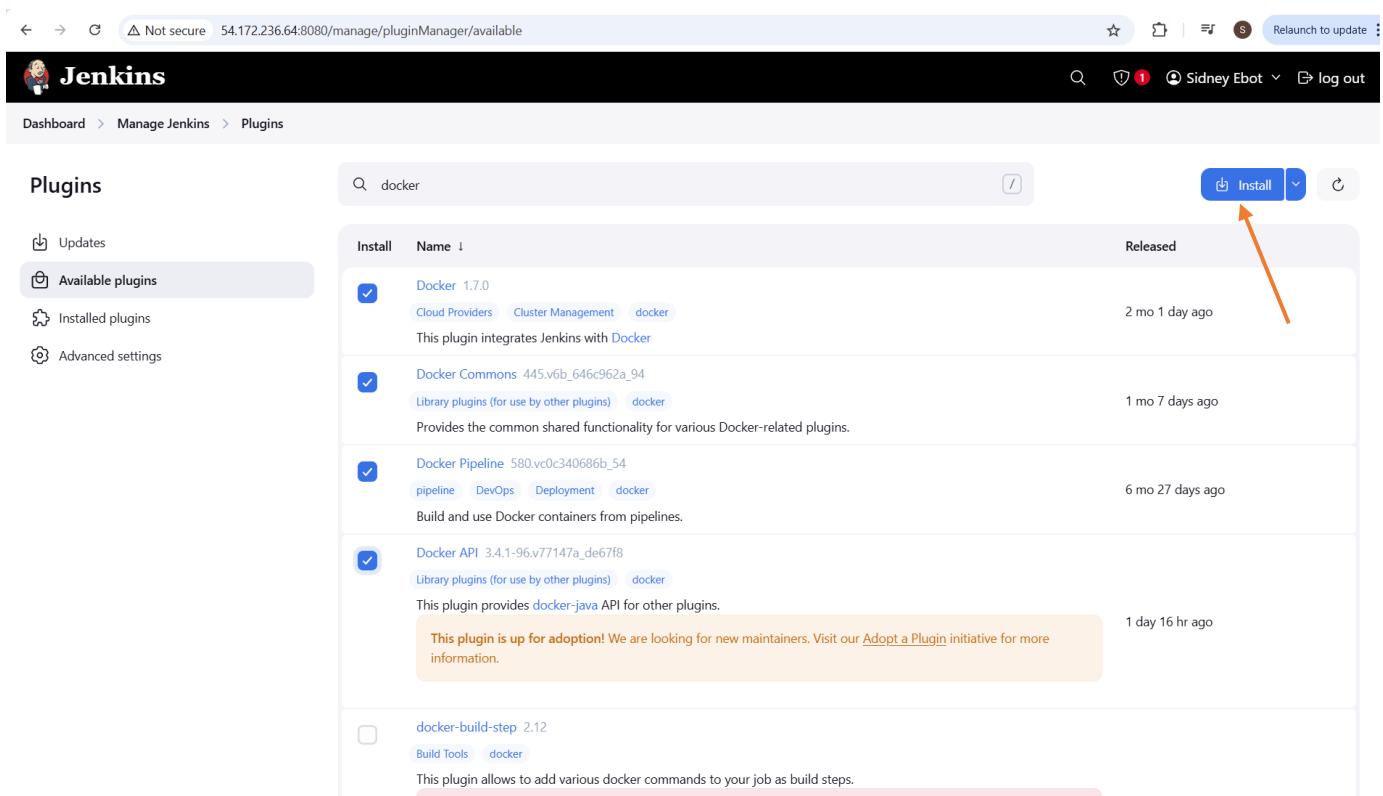
The screenshot shows the 'Plugins' page at 54.172.236.64:8080/manage/plugins. The 'Available plugins' link in the sidebar is highlighted with an orange arrow. The main area shows a message: 'No updates available'.

Click on “Available Plugins”



The screenshot shows the Jenkins Plugin Manager interface. On the left, there's a sidebar with links for Updates, Available plugins (which is selected), Installed plugins, and Advanced settings. The main area has a search bar at the top with the placeholder "Search available plugins". Below the search bar is a table with columns for Install, Name, and Released. The table lists several plugins, including "Command Agent Launcher", "Oracle Java SE Development Kit Installer", "SSH server", "Authentication Tokens API", "Git server", "Docker Commons", and "JavaScript GUI Lib: ACE Editor bundle". An orange arrow points to the search bar.

Search for “Docker”



The screenshot shows the Jenkins Plugin Manager after searching for "Docker". The search bar now contains "docker". In the list, four plugins have checkboxes checked: "Docker", "Docker Commons", "Docker Pipeline", and "Docker API". An orange arrow points to the "Install" button at the top right of the plugin list.

Make the selections as shown above and click on “Install”

Not secure 54.172.236.64:8080/manage/pluginManager/updates/ Relaunch to update

Jenkins

Dashboard > Manage Jenkins > Plugins

Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress

Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Plugin	Status
Cloud Statistics	Success
Authentication Tokens API	Success
Docker Commons	Success
Apache HttpComponents Client 5.x API	Success
Commons Compress API	Success
Docker API	Success
Docker	Success
Docker Commons	Success
Docker Pipeline	Success
Docker API	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

REST API Jenkins 2.489

The plugins have been installed.

Go back to “Manage Jenkins”, then “Tools”

Not secure 54.172.236.64:8080/manage/configureTools/ Relaunch to update

Jenkins

Dashboard > Manage Jenkins > Tools

Tools

Maven Configuration

Default settings provider

Use default maven settings

Default global settings provider

Use default maven global settings

JDK installations

JDK installations

Git installations

Git

Name: Default

Save Apply

Scroll down to “Docker Installations”

Not secure 54.172.236.64:8080/manage/configureTools/

Dashboard > Manage Jenkins > Tools

Add SonarScanner for MSBuild

SonarQube Scanner installations

SonarQube Scanner installations ▾ Edited

Ant installations

Add Ant

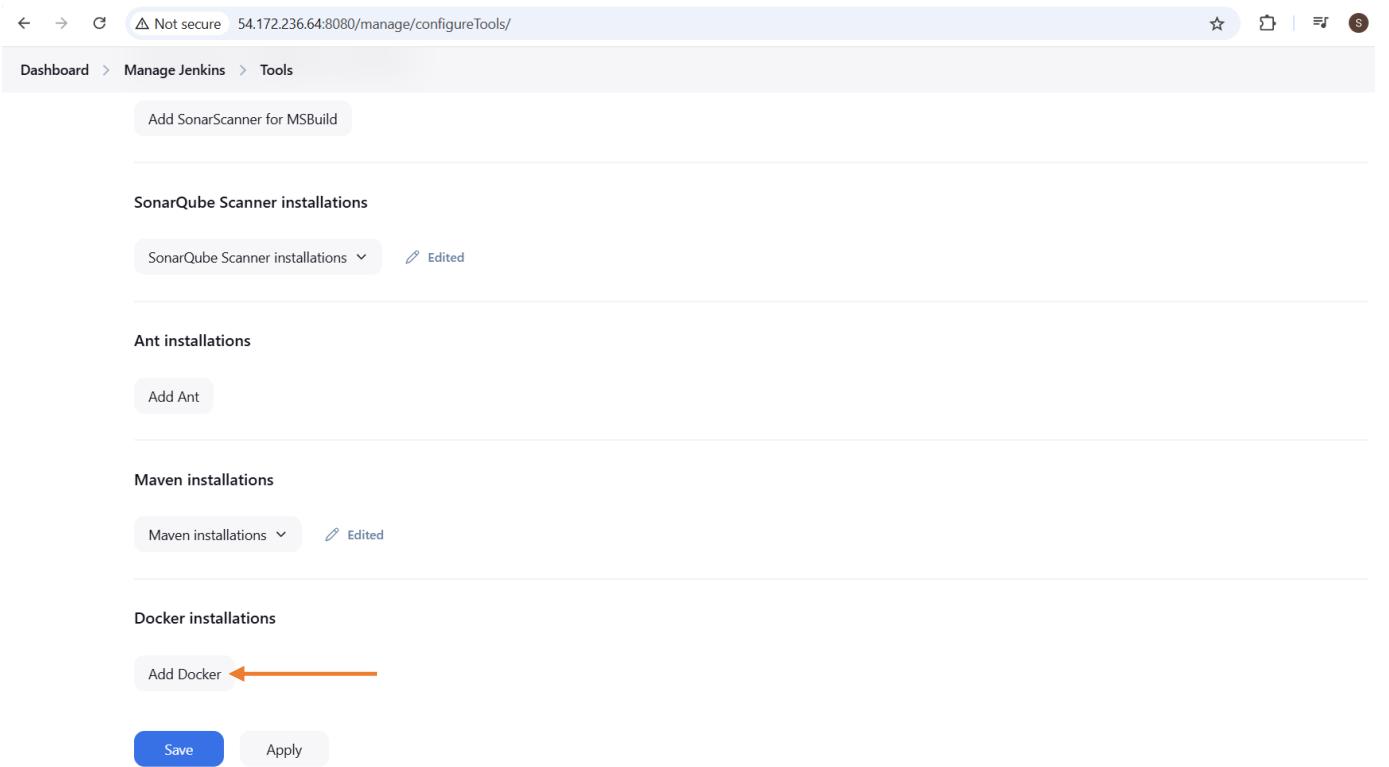
Maven installations

Maven installations ▾ Edited

Docker installations

Add Docker ←

Save Apply



Click on “Add Docker”. Give it the name “docker”, select “Install Automatically” and click on “Add Installer” and choose “Download from Docker.com”

≡ Docker

Name
docker

Install automatically ?

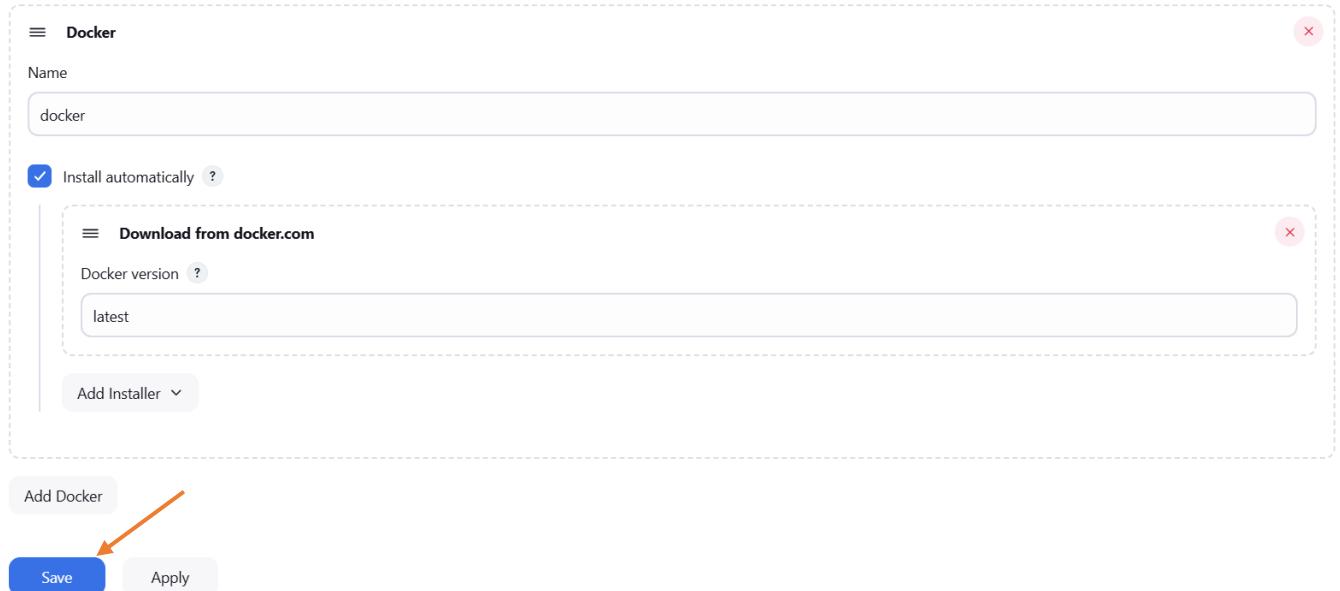
≡ Download from docker.com

Docker version ?
latest

Add Installer ▾

Add Docker ←

Save Apply



Click on “Save”

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

[Set up agent](#) [Set up cloud](#) [Dismiss](#)

System Configuration

- System**: Configure global settings and paths.
- Tools**: Configure tools, their locations and automatic installers.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Appearance**: Configure the look and feel of Jenkins

Security

- Security**: Secure Jenkins; define who is allowed to access/use the system.
- Credentials**: Configure credentials
- Users**: Create/delete/modify users that can log in to this Jenkins.
- Credential Providers**: Configure the credential providers and types

Click on “Dashboard”

[Dashboard](#) >

[New Item](#) [Build History](#) [Manage Jenkins](#) [My Views](#)

[All](#) [+](#)

S	W	Name ↓	Last Success	Last Failure	Last Duration
		demo	1 hr 19 min #8	N/A	24 sec

Icon: S M L

Click on “Manage Jenkins”

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'Build History', 'Manage Jenkins' (which is selected and highlighted in grey), and 'My Views'. The main area is titled 'Manage Jenkins' and contains a message about building on the built-in node. Below this are sections for 'System Configuration' and 'Security'. In the 'System Configuration' section, there are four items: 'System' (Configure global settings and paths), 'Tools' (Configure tools, their locations and automatic installers), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), and 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand). In the 'Security' section, there are three items: 'Security' (Secure Jenkins; define who is allowed to access/use the system), 'Users' (Create/delete/modify users that can log in to this Jenkins), and 'Credentials' (Configure credentials). An orange arrow points from the text 'Click on "Credentials"' in the accompanying instructions to the 'Credentials' link in the 'Security' section.

Click on “Credentials”

The screenshot shows the Jenkins Credentials management page. At the top, it says 'Credentials'. Below that is a table with columns: T, P, Store ↓, Domain, ID, and Name. There is one entry: a file icon, a user icon, the word 'System' (which is highlighted in blue and has an orange arrow pointing to it), '(global)', 'a3d26812-66ab-46bd-aad7-5696ff5166d4', and 'sonar'. Below the table is a section titled 'Stores scoped to Jenkins' with a table showing 'P', 'Store ↓', 'Domains', and an entry for 'System' with '(global)'. At the bottom, there are icons for 'Icon:', 'S', 'M', and 'L'.

Click on “System”

The screenshot shows the Jenkins System page under the Credentials section. A table lists a single entry: "Global credentials (unrestricted)". The table has columns for "Domain" (sorted by name), "Description", and "Icon". The "Icon" column shows icons for Small (S), Medium (M), and Large (L). An orange arrow points to the "Global credentials (unrestricted)" link.

Click on “Global Credential”

The screenshot shows the "Global credentials (unrestricted)" page. It displays a single credential entry with the ID "a3d26812-66ab-46bd-aad7-5696ff5166d4", named "sonar", of type "Secret text", with a description "sonar". A blue button labeled "+ Add Credentials" is visible in the top right corner, with an orange arrow pointing to it.

Click on “Add Credential”

The screenshot shows the "New credentials" creation page. The "Kind" field is set to "Username with password". The "Scope" dropdown is set to "Global (Jenkins, nodes, items, all child items, etc.)". The "Username" field is empty. The "Treat username as secret" checkbox is unchecked. The "Password" field is empty. The "ID" field is empty. The "Description" field is empty. A blue "Create" button is at the bottom left.

Login to the docker hub

The screenshot shows the Docker Hub interface. At the top, there's a navigation bar with links for 'Explore', 'Repositories', 'Organizations', and 'Usage'. A search bar is present with the placeholder 'Search Docker Hub'. Below the navigation is a dropdown menu set to 'ebotsmith' and a search input field. To the right of the search is a button labeled 'Create a repository'. The main area displays a table of repositories owned by 'ebotsmith'. The columns are 'Name', 'Last Pushed', 'Contains', 'Visibility', and 'Scout'. The repositories listed are 'ebotsmith/nodejsapplication' (pushed 13 days ago), 'ebotsmith/springboot-ecs-fargatee' (pushed about 2 months ago), and 'ebotsmith/aws-demo' (pushed 2 months ago). On the right side of the screen, there's a sidebar with a profile picture of a person with an 'E' monogram, the name 'ebotsmith' highlighted with an orange arrow, and links for 'What's new', 'My profile', 'Account settings', and 'Billing'. At the bottom of the sidebar is a 'Sign out' link. A note at the bottom right says 'Create and manage users and grant access to your repositories.' The taskbar at the bottom of the browser window shows various pinned icons.

Copy your username: **ebotsmith**. Paste this user name in the Jenkins GUI and add the Docker Hub password.

The screenshot shows the Jenkins 'Manage Jenkins > Credentials' page. The path is 'Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted)'. A 'Kind' dropdown is set to 'Username with password'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field contains 'ebotsmith'. The 'Password' field contains '*****'. The 'ID' field is empty. The 'Description' field contains 'docker'. At the bottom is a blue 'Create' button with an orange arrow pointing to it.

Click on “Create”

Jenkins

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

Global credentials (unrestricted)

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

ID	Name	Kind	Description
a3d26812-66ab-46bd-aad7-5696ff5166d4	sonar	Secret text	sonar
f04c9509-ba18-4f79-8777-957ebc54d17d	ebotsmith/******** (docker)	Username with password	docker

Icon: S M L

Go to “Pipeline Syntax” page and refresh

Jenkins

Not secure 54.172.236.64:8080/job/demo/pipeline-syntax/

Dashboard > demo > Pipeline Syntax

Snippet Generator

- ① Declarative Directive Generator
- ② Declarative Online Documentation
- ③ Steps Reference
- ④ Global Variables Reference
- ⑤ Online Documentation
- ⑥ Examples Reference
- ⑦ IntelliJ IDEA GDSL

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step

archiveArtifacts: Archive the artifacts

archiveArtifacts ?

Files to archive ?

Advanced ▾

Generate Pipeline Script

Click on the drop down on “**Sample Step**” and select “**withDockerRegistry**” and under Docker installation select “**docker**”

This screenshot shows the Jenkins Pipeline Syntax Snippet Generator. On the left, there's a sidebar with links like 'Snippet Generator', 'Declarative Directive Generator', 'Declarative Online Documentation', 'Steps Reference', 'Global Variables Reference', 'Online Documentation', 'Examples Reference', and 'IntelliJ IDEA GDSL'. The main area is titled 'Overview' and contains a 'Sample Step' section with a dropdown menu showing 'withDockerRegistry: Sets up Docker registry endpoint'. Below it is a configuration form with fields for 'Docker registry URL' (empty), 'Registry credentials' (set to 'ebotsmith/******** (docker)'), and 'Docker installation' (set to 'docker'). At the bottom is a blue 'Generate Pipeline Script' button.

Click on “Generate Pipeline Script”

This screenshot shows the same Jenkins Pipeline Syntax Snippet Generator interface after the 'Generate Pipeline Script' button was clicked. The generated code is displayed in a large text area:

```
// This step should not normally be used in your script. Consult the inline help for details.  
withDockerRegistry(credentialsId: 'f04c9509-ba18-4f79-8777-957ebc54d17d', toolName: 'docker') {  
    // some block  
}
```

A red arrow points from the 'Generate Pipeline Script' button in the previous screenshot to this generated code in the current one. Below the code, there's a 'Global Variables' section with a note about global variables and a link to the 'Global Variables Reference'.

Copy the generated code and add a new stage with it

So, we go back to our Jenkins configuration page

Pipeline

Definition

Pipeline script

Script ?

```
28
29
30
31
32 stage('Code Build') {
33   steps {
34     sh 'mvn clean install'
35   }
36 }
37 stage('Docker Build') {
38   steps {
39     script{
40       withDockerRegistry(credentialsId: 'f04c9509-ba18-4f79-8777-957ebc54d17d', toolName: 'docker') {
41         sh 'docker build -t demo .'
42       }
43     }
44 }
```

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

```
pipeline {
  agent any

  tools{
    jdk 'jdk'
    maven 'maven3'
  }
  environment{
    SCANNER_HOME= tool 'sonar-scanner'
  }

  stages {
    stage('Git Checkout') {
      steps {
        git 'https://github.com/ebotsmith2000/jenkins-pipeline.git'
      }
    }
    stage('Code Compile') {
      steps {
        sh 'mvn clean compile'
      }
    }
    stage('SonarQube Analysis') {
      steps {
        withSonarQubeEnv('SonarQube'){
          sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=new \
-Dsonar.java.binaries=. \
-Dsonar.projectKey=new '''
        }
      }
    }
  }
}
```

```

    }
    stage('Code Build') {
        steps {
            sh 'mvn clean install'
        }
    }
    stage('Docker Build') {
        steps {
            script{
                withDockerRegistry(credentialsId: '7e596026-5cee-4b1f-9d13-97efcaf07959', toolName: 'docker') {
                    sh 'docker build -t demo .'
                }
            }
        }
    }
}

```

Click on “Save”

Let us check if there is any docker image by using the command:

sudo docker images

```

ubuntu@ip-172-31-31-55:~$ sudo systemctl restart jenkins
ubuntu@ip-172-31-31-55:~$ sudo systemctl restart docker
ubuntu@ip-172-31-31-55:~$ sudo docker images
REPOSITORY      TAG          IMAGE ID      CREATED       SIZE
hello-world    latest        d2c94e258dcb   19 months ago  13.3kB
ubuntu@ip-172-31-31-55:~$ 

```

You can see that we have just one docker image called “**hello-world**”

Let us click on “**Build Now**” on the Jenkins GUI

Declarative: Tool Install	Git Checkout	Code Compile	SonarQube Analysis	Code Build	Docker Build
5s	585ms	4s	11s	7s	12s
265ms	831ms	4s	10s	6s	12s
147ms	447ms	4s	10s	9s	
144ms	498ms	4s	13s		
22s	566ms	6s			

Docker build is successful, let us check our docker images again by using the command:

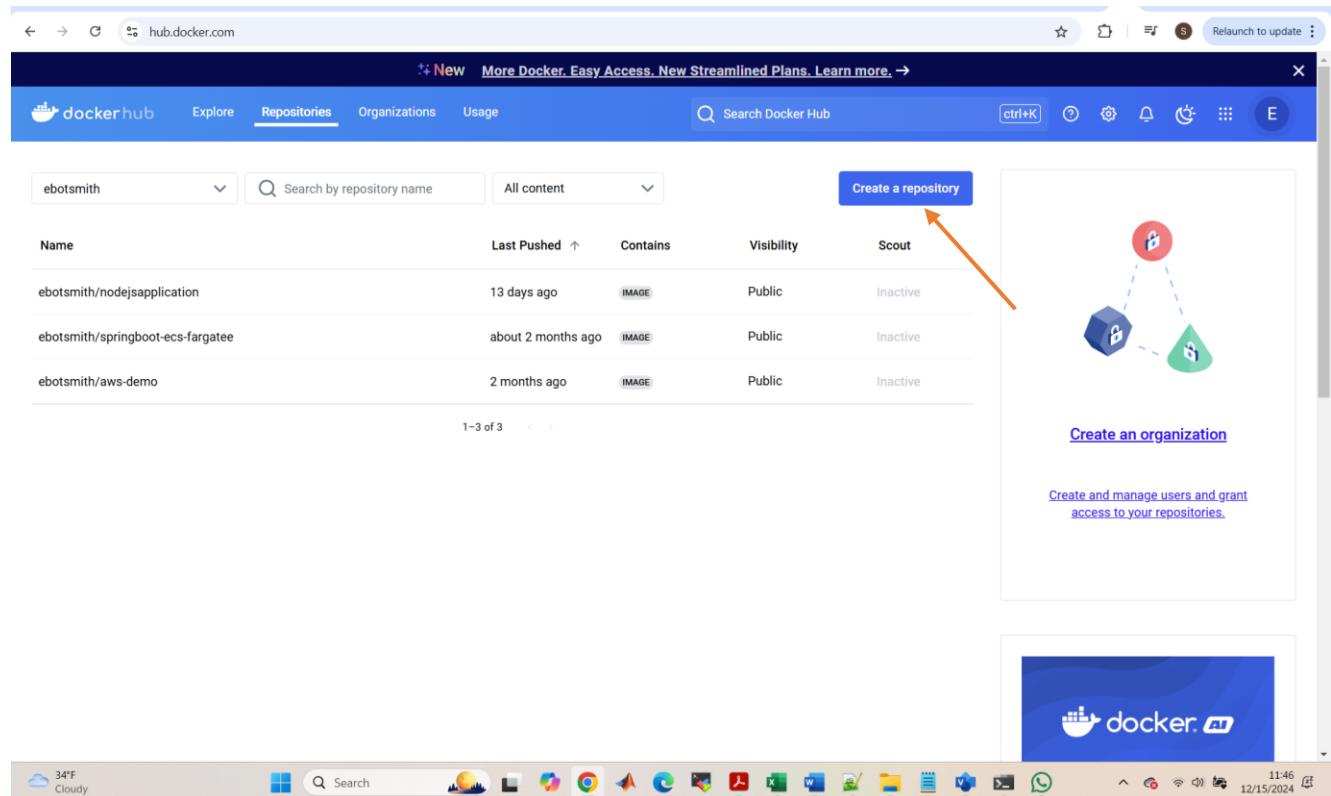
sudo docker images

```
ubuntu@ip-172-31-31-55:~$ sudo docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
demo            latest   bdae5e8b965c  About a minute ago  424MB
hello-world     latest   d2c94e258dcb  19 months ago   13.3kB
ubuntu@ip-172-31-31-55:~$ █
```

You can see that we now have two docker images “**hello-world**” and “**demo**”. So, we have successfully built the docker image of our application. So, the step “**Docker Build**” is done, we have to move to the next step that is “**Push the image to Docker Hub**”

STEP 14: Push the image to Docker Hub

Let us go to the Docker hub and **create a repository**.

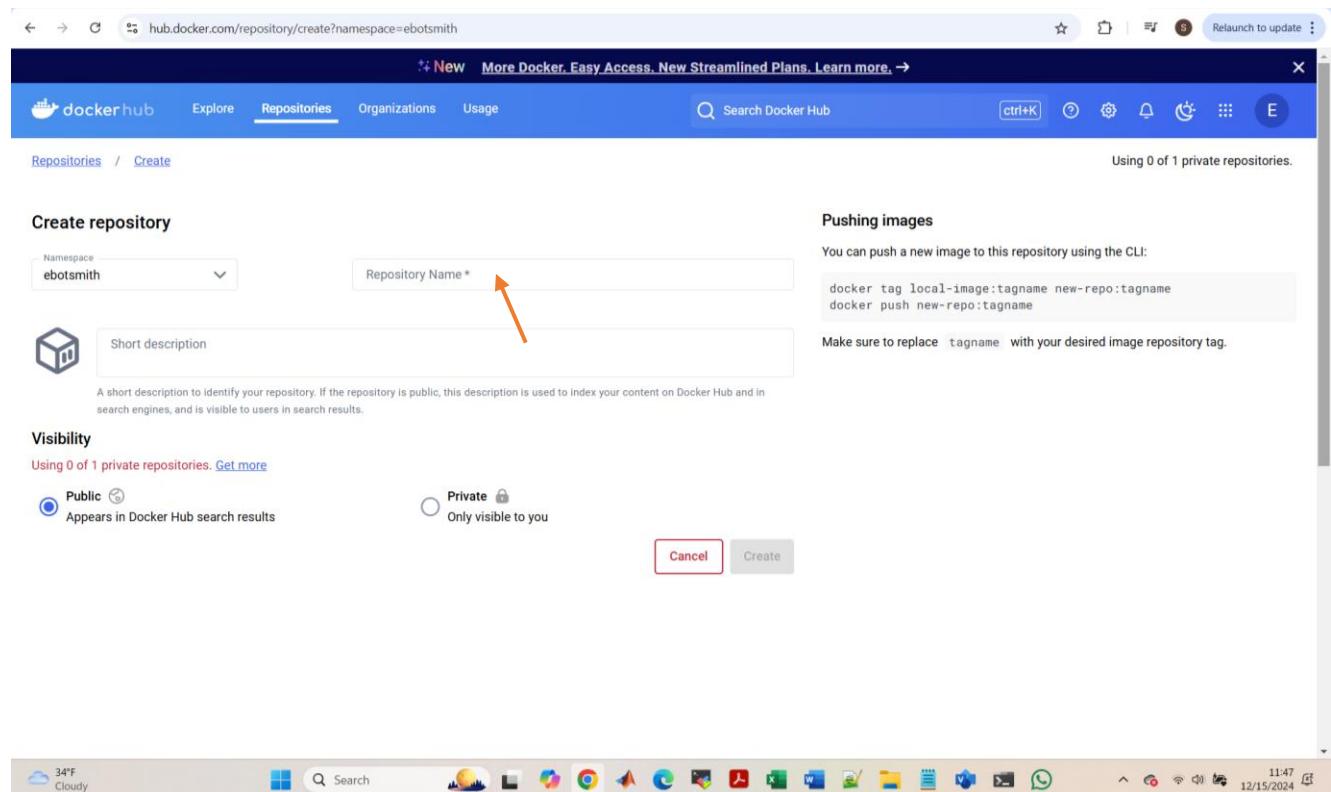


The screenshot shows the Docker Hub interface. At the top, there is a navigation bar with links for 'Explore', 'Repositories' (which is underlined), 'Organizations', and 'Usage'. A search bar is located at the top right. Below the navigation, a dropdown menu shows 'ebotsmith'. To the right of the dropdown is a search bar with placeholder text 'Search by repository name' and a dropdown menu set to 'All content'. On the far right of the header is a 'Create a repository' button, which is highlighted with a red arrow. The main content area displays a table of the user's repositories. The table has columns for 'Name', 'Last Pushed', 'Contains', 'Visibility', and 'Scout'. There are three entries listed:

Name	Last Pushed	Contains	Visibility	Scout
ebotsmith/nodejsapplication	13 days ago	IMAGE	Public	Inactive
ebotsmith/springboot-ecs-fargatee	about 2 months ago	IMAGE	Public	Inactive
ebotsmith/aws-demo	2 months ago	IMAGE	Public	Inactive

At the bottom of the table, it says '1-3 of 3'. To the right of the table, there is a sidebar with icons for creating an organization and managing users. The status bar at the bottom shows the weather as 34°F Cloudy, the date as 12/15/2024, and the time as 11:46.

Click on “Create a Repository”



The screenshot shows the 'Create repository' form on Docker Hub. The URL in the address bar is 'hub.docker.com/repository/create?namespace=ebotsmith'. The form has a 'Namespace' dropdown set to 'ebotsmith' and a 'Repository Name*' input field, which is highlighted with an orange arrow. Below the input field is a 'Short description' text area. To the right of the form, there is a 'Pushing images' section with instructions for using the CLI:

```
docker tag local-image:tagname new-repo:tagname  
docker push new-repo:tagname
```

It also says to replace 'tagname' with your desired image repository tag. At the bottom of the form, there are 'Visibility' options: 'Public' (selected) and 'Private' (radio button). The 'Public' option includes a note that it appears in Docker Hub search results. At the very bottom are 'Cancel' and 'Create' buttons.

Let us give it the name “**demo**”

The screenshot shows the Docker Hub 'Create repository' interface. A red arrow points to the 'Create' button at the bottom right of the form. The 'Repository Name' field contains 'demo'. The 'Namespace' dropdown is set to 'ebotsmith'. The 'Visibility' section shows 'Private' selected. The 'Pushing images' section includes CLI commands for tagging and pushing images.

Click on “**Create**”

The screenshot shows the Docker Hub repository overview for 'ebotsmith/demo'. A red arrow points to the repository name 'ebotsmith/demo'. The 'General' tab is selected. The 'Docker commands' section shows the command 'docker push ebotsmith/demo:tagname'. The 'Tags' section is currently empty. The 'Automated builds' section is available with Pro, Team and Business subscriptions.

Copy this: **ebotsmith/demo**

We have to create a new stage to push the image to Docker hub. So go back to Jenkins GUI

Configure

Average stage times:
Average full run time: ~28s

Declarative: Tool Install	Git Checkout	Code Compile	SonarQube Analysis	Code Build	Docker Build
5s	585ms	4s	11s	7s	12s

Builds

Filter

Today

- #9 4:07 PM
- #8 2:18 PM
- #5 1:28 PM
- #2 2:51 AM

34°F Cloudy

Search

11:42 12/15/2024

Click on “Configure” and scroll down to “Pipeline”

Configure

Pipeline

Definition

Pipeline script

```

1 pipeline {
2     agent any
3     tools{
4         jdk 'jdk'
5         maven 'maven3'
6     }
7     environment{
8         SCANNER_HOME=tool 'sonar-scanner'
9     }
10    stages {
11        stage('Git Checkout') {
12            steps {
13                git 'https://github.com/ebotsmith2000/CICD-Pipeline.git'
14            }
15        }
16    }
17

```

Use Groovy Sandbox

Save Apply

REST API Jenkins 2.489

34°F Cloudy

Search

11:42 12/15/2024

Add another stage

This '**docker tag demo ebotsmith/demo**'. Means we are tagging the image “**demo**” into our Docker hub repository.

Pipeline

Definition

Pipeline script

Script ?

```
41 42 43 44 45 46 stage('Docker Image Push') { 47   steps { 48     script{ 49       withDockerRegistry(credentialsId: 'f04c9509-ba18-4f79-8777-957ebc54d17d', toolName: 'docker') { 50         sh 'docker tag demo ebotsmith/cicd:$BUILD_ID' 51         sh 'docker push ebotsmith/cicd:$BUILD_ID' 52       } 53     } 54   } 55 } 56 }
```

Use Groovy Sandbox ?

Pipeline Syntax

Save

Apply

```
pipeline {  agent any  tools{    jdk 'jdk'    maven 'maven3'  }  environment{    SCANNER_HOME= tool 'sonar-scanner'  }  stages {    stage('Git Checkout') {      steps {        git 'https://github.com/ebotsmith2000/jenkins-pipeline.git'      }    }    stage('Code Compile') {      steps {        sh 'mvn clean compile'      }    }    stage('SonarQube Analysis') {      steps {        withSonarQubeEnv('SonarQube'){          sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=new \          -Dsonar.java.binaries=. \        }      }    }  }
```

```
        -Dsonar.projectKey=new ''
    }
}
stage('Code Build') {
    steps {
        sh 'mvn clean install'
    }
}
stage('Docker Build') {
    steps {
        script{
            withDockerRegistry(credentialsId: '7e596026-5cee-4b1f-9d13-97efcaf07959', toolName: 'docker') {
                sh 'docker build -t demo .'
            }
        }
    }
}
stage('Docker Image Push') {
    steps {
        script{
            withDockerRegistry(credentialsId: '7e596026-5cee-4b1f-9d13-97efcaf07959', toolName: 'docker') {
                sh 'docker tag demo ebotsmith/demo:$BUILD_ID'
                sh 'docker push ebotsmith/demo:$BUILD_ID'
            }
        }
    }
}
```

Click on “Save”

The screenshot shows the Jenkins Stage View for the 'demo' pipeline. On the left, there's a sidebar with various pipeline management options like 'Changes', 'Build Now' (which has an orange arrow pointing to it), 'Configure', 'Delete Pipeline', etc. Below that is a 'Builds' section showing recent builds (#9, #8, #5, #2). The main area is titled 'Stage View' and displays a grid of stages: Declarative: Tool Install, Git Checkout, Code Compile, SonarQube Analysis, Code Build, and Docker Build. Each stage shows its average time and the current run time for the latest build. A 'SonarQube Quality Gate' section indicates that the build has passed. The bottom of the screen shows the Windows taskbar with various icons.

Then, click on “Build Now” to build

This screenshot shows the Jenkins Stage View after a new build was triggered. The 'Build Now' button is no longer visible in the sidebar. The main grid now includes an additional column for 'Docker Image Push'. The 'SonarQube Quality Gate' section shows the build has passed. The bottom of the screen shows the Windows taskbar.

The image has been pushed. Go now to the docker hub and check the repository if the image is there.

The screenshot shows the Docker Hub repository page for `ebotsmith/cicd`. The repository has one tag, `14`, which was pushed 2 minutes ago. The Docker commands section shows the command `docker push ebotsmith/cicd:tagname`. The `See all` link is highlighted with an orange arrow.

You can now see that the image has been tagged. Click on “See All” to check if the image is ready

The screenshot shows the Docker Hub tags page for `ebotsmith/cicd`. The `14` tag is selected. The tag details show it was last pushed 5 minutes ago by `ebotsmith`. The tag is digest `97027d357c0f`, OS/ARCH is `linux/amd64`, and the last pull was 5 minutes ago. A `Copy` button is available for the tag URL.

The image is ready. So, we have pushed the docker image to Docker hub. Now we have to deploy the image to Kubernetes.

STEP 15: Deploy the Image to Kubernetes.

We have to deploy the image from Docker hub to Kubernetes.

PART 1: Install Kubernetes on Server

To deploy to Kubernetes, we just need to install Kubernetes and its plugins.

First run the command: **cd /home/ubuntu/**

```
ubuntu@ip-172-31-86-141:~$
```

Then run the command:

```
11
```

```
ubuntu@ip-172-31-86-141:~$ ll
total 295936
drwxr-x---  5 ubuntu  ubuntu        4096 Dec 16 21:55 .
drwxr-xr-x  3 root   root         4096 Dec 16 19:41 ../
-rw-r--r--  1 ubuntu  ubuntu       220 Mar 31 2024 .bash_logout
-rw-r--r--  1 ubuntu  ubuntu      3771 Mar 31 2024 .bashrc
drwx----- 2 ubuntu  ubuntu        4096 Dec 16 20:39 .cache/
-rw-r--r--  1 ubuntu  ubuntu       807 Mar 31 2024 .profile
drwx----- 2 ubuntu  ubuntu        4096 Dec 16 19:41 .ssh/
-rw-r--r--  1 ubuntu  ubuntu        0 Dec 16 20:45 .sudo_as_admin_successful
-rw-rw-r--  1 ubuntu  ubuntu    9102945 Aug 17 18:44 apache-maven-3.9.9-bin.tar.gz
drwxr-xr-x 11 ubuntu  ubuntu       4096 Feb  3 2023 sonarqube-9.9.0.65466/
-rw-rw-r--  1 ubuntu  ubuntu  293899334 Feb  3 2023 sonarqube-9.9.0.65466.zip
ubuntu@ip-172-31-86-141:~$
```

To install Kubernetes, open a new vi name Kubeadm.sh using the command:

```
sudo vi kubeadm.sh
```

```
*kubeadm.sh* [New]
```

```
0,0-1     All
```

Paste the below code on the open vi

```
#!/bin/bash

echo ".....-----#####._.-.INSTALLING KUBERNETES-.-
._.#####-----....."

# Update and install dependencies
apt-get update
apt-get install -y curl apt-transport-https ca-certificates gnupg lsb-release

# Configure Kubernetes package repository
KUBE_LATEST=$(curl -L -s https://dl.k8s.io/release/stable.txt | awk 'BEGIN { FS="." } {
printf "%s.%s", $1, $2 }')
mkdir -p /etc/apt/keyrings
curl -fsSL https://pkgs.k8s.io/core:/stable:/${KUBE_LATEST}/deb/Release.key | gpg --dearmor
-o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/${KUBE_LATEST}/deb/ /" >
/etc/apt/sources.list.d/kubernetes.list

# Update package list and install Kubernetes components
apt-get update
apt-get install -y kubelet kubectl kubeadm kubernetes-cni containerd

# Configure containerd
mkdir -p /etc/containerd
containerd config default | sed 's/SystemdCgroup = false/SystemdCgroup = true/' >
/etc/containerd/config.toml
systemctl restart containerd
systemctl enable kubelet

# Initialize Kubernetes cluster
kubeadm reset -f
kubeadm init --pod-network-cidr='10.244.0.0/16' --service-cidr='10.96.0.0/16' --skip-token-
print

# Configure kubectl for the root user
mkdir -p ~/.kube
cp -i /etc/kubernetes/admin.conf ~/.kube/config

# Install a pod network (WeaveNet)
kubectl apply -f "https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-
daemonset-k8s-1.11.yaml"
kubectl rollout status daemonset weave-net -n kube-system --timeout=90s

# Remove taints from the control-plane node to schedule workloads
node=$(kubectl get nodes -o=jsonpath='{.items[0].metadata.name}')
for taint in $(kubectl get node $node -o=jsonpath='{range
.spec.taints[*]}{.key}{":}{.effect}{"-"}{end}')
do
    kubectl taint node $node $taint
done

kubectl get nodes -o wide

echo ".....-----#####._.-.KUBERNETES INSTALLATION COMPLETED-.-
._.#####-----....."
```

```

#!/bin/bash
echo ".....-.- INSTALLING KUBERNETES -.-....."
# Update and install dependencies
apt-get update
apt-get install -y curl apt-transport-https ca-certificates gnupg lsb-release

# Configure Kubernetes package repository
KUBE_LATEST=$(curl -L -s https://dl.k8s.io/release/stable.txt | awk '$BEGIN { FS="." } { printf "%s.%s", $1, $2 }')
curl -fsSL https://pkgs.k8s.io/core/stable/${KUBE_LATEST}/debRelease.key | gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core/stable/${KUBE_LATEST}/deb/ > /etc/apt/sources.list.d/kubernetes.list"

# Update package list and install Kubernetes components
apt-get update
apt-get install -y kubelet kubeadm kubernetes-cni containerd

# Configure containerd
mkdir -p /etc/containerd
containerd config default | sed 's/SystemdCgroup = false/SystemdCgroup = true/' > /etc/containerd/config.toml
systemctl restart containerd
systemctl enable kubelet

# Initialize Kubernetes cluster
kubeadm reset
kubeadm init --pod-network-cidr='10.244.0.0/16' --service-cidr='10.96.0.0/16' --skip-token-print

# Configure kubectl for the root user
mkdir -p ~/.kube
cp -i /etc/kubernetes/admin.conf ~/.kube/config

# Install a pod network (WeaveNet)
kubectl apply -f "https://github.com/weaveworks/weave/releases/download/v2.0.1/weave-daemonset-k8s-1.11.yaml"

# Remove taints from the control-plane node to schedule workloads
node=$(kubectl get nodes --selector=kubernetes.io/os=linux | head -1)
for taint in $(kubectl get node ${node} --export yaml | jq '.status.taints[]|.key|(.effect)=="NoSchedule"|.endT') do
    kubectl taint node ${node} ${taint}
done

kubectl get nodes -o wide
echo ".....-.- KUBERNETES INSTALLATION COMPLETED -.-....."

```

46,136 All

Then to save and quit the vi, type (**:wq**) and press **ENTER**

```

ubuntu@ip-172-31-86-141:~$ ll
total 295936
drwxr-x--- 5 ubuntu ubuntu      4096 Dec 16 21:55 .
drwxr-xr-x  3 root   root      4096 Dec 16 19:41 ..
-rw-r--r--  1 ubuntu ubuntu      220 Mar 31 2024 .bash_logout
-rw-r--r--  1 ubuntu ubuntu     3771 Mar 31 2024 .bashrc
drwx----- 2 ubuntu ubuntu     4096 Dec 16 20:39 .cache/
-rw-r--r--  1 ubuntu ubuntu      807 Mar 31 2024 .profile
drwx----- 2 ubuntu ubuntu     4096 Dec 16 19:41 .ssh/
-rw-r--r--  1 ubuntu ubuntu      0 Dec 16 20:45 .sudo_as_admin_successful
-rw-rw-r--  1 ubuntu ubuntu  9102945 Aug 17 18:44 apache-maven-3.9.9-bin.tar.gz
drwxr-xr-x 11 ubuntu ubuntu     4096 Feb  3 2023 sonarqube-9.9.0.65466/
-rw-rw-r--  1 ubuntu ubuntu 293899334 Feb  3 2023 sonarqube-9.9.0.65466.zip
ubuntu@ip-172-31-86-141:~$ sudo vi kubeadm.sh
ubuntu@ip-172-31-86-141:~$ 
```

To install the script, we first execute permission with the command:

sudo chmod +x kubeadm.sh

```

ubuntu@ip-172-31-86-141:~$ ll
total 295936
drwxr-x--- 5 ubuntu ubuntu      4096 Dec 16 21:55 .
drwxr-xr-x  3 root   root      4096 Dec 16 19:41 ..
-rw-r--r--  1 ubuntu ubuntu      220 Mar 31 2024 .bash_logout
-rw-r--r--  1 ubuntu ubuntu     3771 Mar 31 2024 .bashrc
drwx----- 2 ubuntu ubuntu     4096 Dec 16 20:39 .cache/
-rw-r--r--  1 ubuntu ubuntu      807 Mar 31 2024 .profile
drwx----- 2 ubuntu ubuntu     4096 Dec 16 19:41 .ssh/
-rw-r--r--  1 ubuntu ubuntu      0 Dec 16 20:45 .sudo_as_admin_successful
-rw-rw-r--  1 ubuntu ubuntu  9102945 Aug 17 18:44 apache-maven-3.9.9-bin.tar.gz
drwxr-xr-x 11 ubuntu ubuntu     4096 Feb  3 2023 sonarqube-9.9.0.65466/
-rw-rw-r--  1 ubuntu ubuntu 293899334 Feb  3 2023 sonarqube-9.9.0.65466.zip
ubuntu@ip-172-31-86-141:~$ sudo vi kubeadm.sh
ubuntu@ip-172-31-86-141:~$ sudo chmod +x kubeadm.sh
ubuntu@ip-172-31-86-141:~$ 
```

Run the command

11

```
ubuntu@ip-172-31-86-141:~$ sudo chmod +x kubeadm.sh
ubuntu@ip-172-31-86-141:~$ ll
total 295940
drwxr-x---  5 ubuntu  ubuntu        4096 Dec 16 23:08 .
drwxr-xr-x  3 root   root         4096 Dec 16 19:41 ../
-rw-r--r--  1 ubuntu  ubuntu       220 Mar 31  2024 .bash_logout
-rw-r--r--  1 ubuntu  ubuntu      3771 Mar 31  2024 .bashrc
drwx----- 2 ubuntu  ubuntu      4096 Dec 16 20:39 .cache/
-rw-r--r--  1 ubuntu  ubuntu       807 Mar 31  2024 .profile
drwx----- 2 ubuntu  ubuntu      4096 Dec 16 19:41 .ssh/
-rw-r--r--  1 ubuntu  ubuntu        0 Dec 16 20:45 .sudo_as_admin_successful
-rw-rw-r--  1 ubuntu  ubuntu    9102945 Aug 17 18:44 apache-maven-3.9.9-bin.tar.gz
-rwxr-xr-x  1 root   root      2029 Dec 16 23:08 kubeadm.sh*
drwxr-xr-x 11 ubuntu  ubuntu     4096 Feb  3  2023 sonarqube-9.9.0.65466/
-rw-rw-r--  1 ubuntu  ubuntu  293899334 Feb  3  2023 sonarqube-9.9.0.65466.zip
ubuntu@ip-172-31-86-141:~$ █
```

Run the command to switch to root user:

sudo su

```
ubuntu@ip-172-31-86-141:~$ sudo chmod +x kubeadm.sh
ubuntu@ip-172-31-86-141:~$ ll
total 295940
drwxr-x---  5 ubuntu  ubuntu        4096 Dec 16 23:08 .
drwxr-xr-x  3 root   root         4096 Dec 16 19:41 ../
-rw-r--r--  1 ubuntu  ubuntu       220 Mar 31  2024 .bash_logout
-rw-r--r--  1 ubuntu  ubuntu      3771 Mar 31  2024 .bashrc
drwx----- 2 ubuntu  ubuntu      4096 Dec 16 20:39 .cache/
-rw-r--r--  1 ubuntu  ubuntu       807 Mar 31  2024 .profile
drwx----- 2 ubuntu  ubuntu      4096 Dec 16 19:41 .ssh/
-rw-r--r--  1 ubuntu  ubuntu        0 Dec 16 20:45 .sudo_as_admin_successful
-rw-rw-r--  1 ubuntu  ubuntu    9102945 Aug 17 18:44 apache-maven-3.9.9-bin.tar.gz
-rwxr-xr-x  1 root   root      2029 Dec 16 23:08 kubeadm.sh*
drwxr-xr-x 11 ubuntu  ubuntu     4096 Feb  3  2023 sonarqube-9.9.0.65466/
-rw-rw-r--  1 ubuntu  ubuntu  293899334 Feb  3  2023 sonarqube-9.9.0.65466.zip
ubuntu@ip-172-31-86-141:~$ sudo su
root@ip-172-31-86-141:/home/ubuntu# █
```

Now, let us install kubeadm.sh using the command:

./kubeadm.sh

```
[kubebundle-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.86.141:6443 --token <value withheld> \
    --discovery-token-ca-cert-hash sha256:a36849dbe583253cc288d510f4e1dadcl43c9ac7d7adcf248685cfb91beb833
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
Waiting for daemon set spec update to be observed...
Waiting for daemon set "weave-net" rollout to finish: 0 of 1 updated pods are available...
Waiting for daemon set "weave-net" rollout to finish: 0 of 1 updated pods are available...
daemon set "weave-net" successfully rolled out
node/ip-172-31-86-141 un tainted
NAME           STATUS   ROLES      AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-86-141   Ready    control-plane   23s  v1.32.0  172.31.86.141  <none>        Ubuntu 24.04.1 LTS  6.8.0-1018-aws  containerd://1.7.12
.....-----#-----KUBERNETES INSTALLATION COMPLETED---#-----#
root@ip-172-31-86-141:/home/ubuntu#
```

i-0ce9d13dd85b2daec (My Server)

PublicIPs: 3.87.67.132 PrivateIPs: 172.31.86.141

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

It is done. Now, let us try the command:

kubectl get nodes

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.86.141:6443 --token <value withheld> \
    --discovery-token-ca-cert-hash sha256:a36849dbe583253cc288d510f4e1dadcl43c9ac7d7adcf248685cfb91beb833
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
Waiting for daemon set spec update to be observed...
Waiting for daemon set "weave-net" rollout to finish: 0 of 1 updated pods are available...
Waiting for daemon set "weave-net" rollout to finish: 0 of 1 updated pods are available...
daemon set "weave-net" successfully rolled out
node/ip-172-31-86-141 un tainted
NAME           STATUS   ROLES      AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE          KERNEL-VERSION   CONTAINER-RUNTIME
ip-172-31-86-141   Ready    control-plane   23s  v1.32.0  172.31.86.141  <none>        Ubuntu 24.04.1 LTS  6.8.0-1018-aws  containerd://1.7.12
.....-----#-----KUBERNETES INSTALLATION COMPLETED---#-----#
root@ip-172-31-86-141:/home/ubuntu# kubectl get nodes
NAME           STATUS   ROLES      AGE   VERSION
ip-172-31-86-141   Ready    control-plane   3m44s  v1.32.0
root@ip-172-31-86-141:/home/ubuntu#
```

i-0ce9d13dd85b2daec (My Server)

PublicIPs: 3.87.67.132 PrivateIPs: 172.31.86.141

CloudShell Feedback So, we have one node and it is ready. Let us now run the command:

kubectl get pods

```
root@ip-172-31-86-141:/home/ubuntu# kubectl get pods
No resources found in default namespace.
root@ip-172-31-86-141:/home/ubuntu#
```

No resources found in this default namespace.

Run the command:

```
cd ~
```

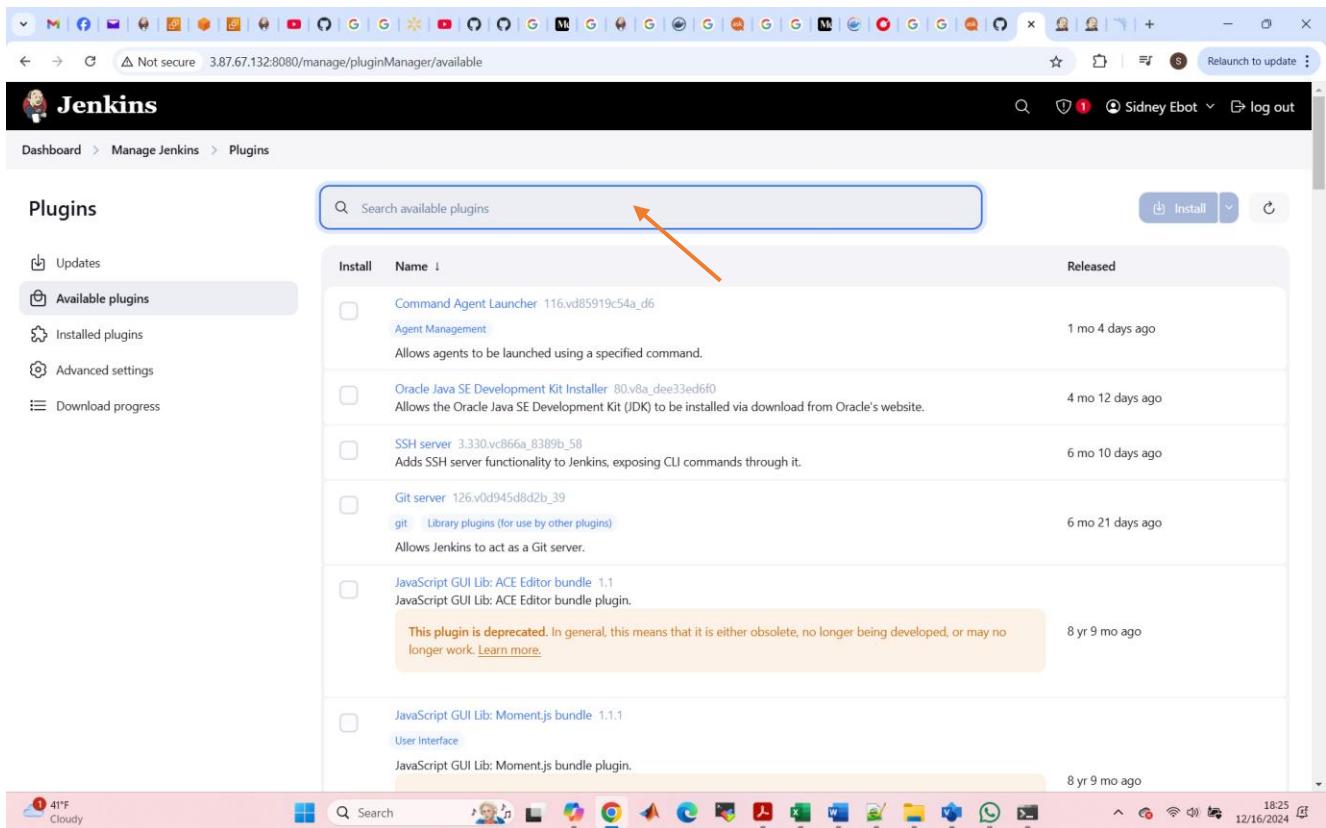
```
root@ip-172-31-86-141:/home/ubuntu# cd ~  
root@ip-172-31-86-141:~# █
```

PART 2: Install Kubernetes Plugins on Jenkins

Now, let us configure the CI/CD pipeline. Go to “Manage Jenkins”

The screenshot shows the Jenkins Manage Jenkins page. At the top, there's a navigation bar with links for 'Dashboard', 'Manage Jenkins' (which is highlighted), and 'My Views'. Below the navigation, there's a search bar and several buttons: 'Set up agent', 'Set up cloud', and 'Dismiss'. A message at the top right says: 'Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).'. The main area is titled 'System Configuration' and contains six sections: 'Build Queue' (No builds in the queue), 'Build Executor Status' (0/2), 'System' (Configure global settings and paths), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), 'Tools' (Configure tools, their locations and automatic installers), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand), 'Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), and 'Appearance' (Configure the look and feel of Jenkins). Below this, there's a 'Security' section with 'Security' (Secure Jenkins; define who is allowed to access/use the system) and 'Users' (Create/delete/modify users that can log in to this Jenkins). The bottom of the screen shows a Windows taskbar with various icons and the date/time '18:24 12/16/2024'.

Go to “Available Plugins”



Jenkins

Dashboard > Manage Jenkins > Plugins

Plugins

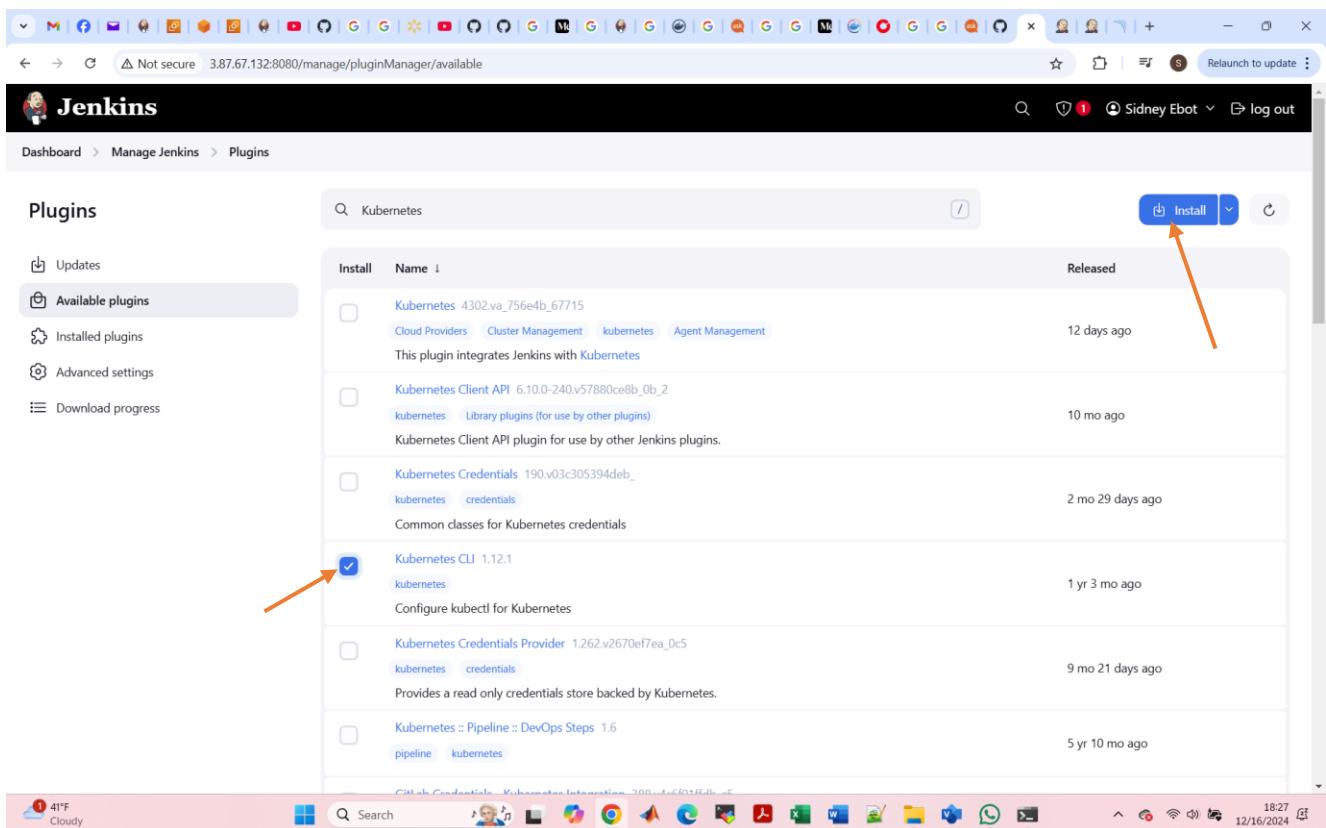
Search available plugins

Install Name Released

- Command Agent Launcher 116.vd85919c54a_d6 1 mo 4 days ago
- Agent Management
- Allows agents to be launched using a specified command.
- Oracle Java SE Development Kit Installer 80.v8a_dee33ed6f0 4 mo 12 days ago
- Allows the Oracle Java SE Development Kit (JDK) to be installed via download from Oracle's website.
- SSH server 3.330.vc866a_8389b_58 6 mo 10 days ago
- Adds SSH server functionality to Jenkins, exposing CLU commands through it.
- Git server 126.v0d945d8d2b_39 6 mo 21 days ago
- git Library plugins (for use by other plugins)
- Allows Jenkins to act as a Git server.
- JavaScript GUI Lib: ACE Editor bundle 1.1 8 yr 9 months ago
- JavaScript GUI Lib: ACE Editor bundle plugin.
- This plugin is deprecated. In general, this means that it is either obsolete, no longer being developed, or may no longer work. [Learn more](#).
- JavaScript GUI Lib: Moment.js bundle 1.1.1 8 yr 9 months ago
- User Interface
- JavaScript GUI Lib: Moment.js bundle plugin.

Cloudy 18:25 12/16/2024

Search for “Kubernetes” and select “Kubernetes CLI”



Jenkins

Dashboard > Manage Jenkins > Plugins

Plugins

Search Kubernetes

Install Name Released

- Kubernetes 4302.va_.756e4b_67715 12 days ago
- Cloud Providers Cluster Management kubernetes Agent Management
- This plugin integrates Jenkins with Kubernetes
- Kubernetes Client API 6.10.0-240.v57880ce8b_0b_2 10 mo ago
- kubernetes Library plugins (for use by other plugins)
- Kubernetes Client API plugin for use by other Jenkins plugins.
- Kubernetes Credentials 190.v03c305394deb_ 2 mo 29 days ago
- kubernetes credentials
- Common classes for Kubernetes credentials
- Kubernetes CLI 1.12.1 1 yr 3 mo ago
- kubernetes
- Configure kubectl for Kubernetes
- Kubernetes Credentials Provider 1.262.v2670ef7ea_0c5 9 mo 21 days ago
- kubernetes credentials
- Provides a read only credentials store backed by Kubernetes.
- Kubernetes :: Pipeline :: DevOps Steps 1.6 5 yr 10 mo ago
- pipeline kubernetes

Cloudy 18:27 12/16/2024

Click on “Install”

Plugins

Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Plugin	Status
Cloud Statistics	Success
Authentication Tokens API	Success
Docker Commons	Success
Apache HttpComponents Client 5.x API	Success
Commons Compress API	Success
Docker API	Success
Docker	Success
Docker Commons	Success
Docker Pipeline	Success
Docker API	Success
Loading plugin extensions	Success
Kubernetes Client API	Success
Kubernetes Credentials	Success
Kubernetes CLI	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

On the Instance terminal, run the command:

11

```
root@ip-172-31-86-141:/home/ubuntu# cd ~
root@ip-172-31-86-141:~# ll
total 40
drwx----- 5 root root 4096 Dec 16 23:41 .
drwxr-xr-x 22 root root 4096 Dec 16 19:41 ../
-rw----- 1 root root 54 Dec 16 23:41 .bash_history
-rw-r--r-- 1 root root 3106 Apr 22 2024 .bashrc
drwxr-xr-x 3 root root 4096 Dec 16 23:13 .kube/
-rw----- 1 root root 20 Dec 16 21:27 .lessht
-rw-r--r-- 1 root root 161 Apr 22 2024 .profile
drwx----- 2 root root 4096 Dec 16 19:41 .ssh/
-rw----- 1 root root 1831 Dec 16 23:08 .viminfo
drwx----- 3 root root 4096 Dec 16 19:41 snap/
root@ip-172-31-86-141:~# █
```

Then run the command:

cd .kube/

```
root@ip-172-31-86-141:/home/ubuntu# cd ~
root@ip-172-31-86-141:~# ll
total 40
drwx----- 5 root root 4096 Dec 16 23:41 .
drwxr-xr-x 22 root root 4096 Dec 16 19:41 ../
-rw----- 1 root root 54 Dec 16 23:41 .bash_history
-rw-r--r-- 1 root root 3106 Apr 22 2024 .bashrc
drwxr-xr-x 3 root root 4096 Dec 16 23:13 .kube/
-rw----- 1 root root 20 Dec 16 21:27 .lessshst
-rw-r--r-- 1 root root 161 Apr 22 2024 .profile
drwx----- 2 root root 4096 Dec 16 19:41 .ssh/
-rw----- 1 root root 1831 Dec 16 23:08 .viminfo
drwx----- 3 root root 4096 Dec 16 19:41 snap/
root@ip-172-31-86-141:~# cd .kube/
root@ip-172-31-86-141:~/ kube#
```

Run the command"

ll

```
root@ip-172-31-86-141:/home/ubuntu# cd ~
root@ip-172-31-86-141:~# ll
total 40
drwx----- 5 root root 4096 Dec 16 23:41 .
drwxr-xr-x 22 root root 4096 Dec 16 19:41 ../
-rw----- 1 root root 54 Dec 16 23:41 .bash_history
-rw-r--r-- 1 root root 3106 Apr 22 2024 .bashrc
drwxr-xr-x 3 root root 4096 Dec 16 23:13 .kube/
-rw----- 1 root root 20 Dec 16 21:27 .lessshst
-rw-r--r-- 1 root root 161 Apr 22 2024 .profile
drwx----- 2 root root 4096 Dec 16 19:41 .ssh/
-rw----- 1 root root 1831 Dec 16 23:08 .viminfo
drwx----- 3 root root 4096 Dec 16 19:41 snap/
root@ip-172-31-86-141:~# cd .kube/
root@ip-172-31-86-141:~/ kube# ll
total 20
drwxr-xr-x 3 root root 4096 Dec 16 23:13 .
drwx----- 5 root root 4096 Dec 16 23:41 ../
drwxr-x--- 4 root root 4096 Dec 16 23:13 cache/
-rw----- 1 root root 5657 Dec 16 23:13 config
root@ip-172-31-86-141:~/ kube#
```

Here, we have the “config” file. This is the file we will use to login to the cluster.

Run the command:

cat config

Copy this and paste on a notepad and save as “**kubeconfig**”

PART 3: Add credentials for Kubernetes

Now go to Jenkins GUI and navigate to “Manage Jenkins”, then to “credentials” to add a credential

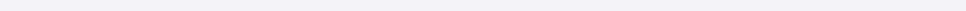
Dashboard > Manage Jenkins > Credentials

Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	189474c2-9633-4663-8ab1-35f87fa4d12a	sonar
		System	(global)	6bf02017-4de0-46f0-a8dc-b4c16bb2e882	ebotsmith/***** (docker)

Stores scoped to Jenkins

P Store ↓ Domains



The screenshot shows the Cloudflare dashboard with the 'System' tab highlighted in blue. An orange arrow points from the bottom-left towards the 'System' tab. The tabs visible are 'System', 'Logs', 'Metrics', and 'Logs & Metrics'.

Click on “System”

System

Domain ↓	Description
 Global credentials (unrestricted)	Credentials that should be available irrespective of domain specification to requirements matching.

Click on “Global Credentials”

Global credentials (unrestricted)



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

ID	Name	Kind	Description
 189474c2-9633-4663-8ab1-35f87fa4d12a	sonar	Secret text	sonar
 6bf02017-4de0-46f0-a8dc-b4c16bb2e882	ebotsmith/******/ (docker)	Username with password	docker

Icon: S M L

Click on “add 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 ?

Treat username as secret ?

Password ?

ID ?

Description ?

Create

Click on the drop down on “kind”, select “Secret File”

New credentials

Kind
Secret file

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

File
Choose File No file chosen

ID ?
k8s

Description ?
k8s

Create

Then click on “Choose file” and select the file we just saved, that is “kubeconfig” and in description put “k8s”

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

New credentials

Kind
Secret file

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

File
Choose File kubeconfig.txt

ID ?
k8s

Description ?
k8s

Create

Click on “Create”

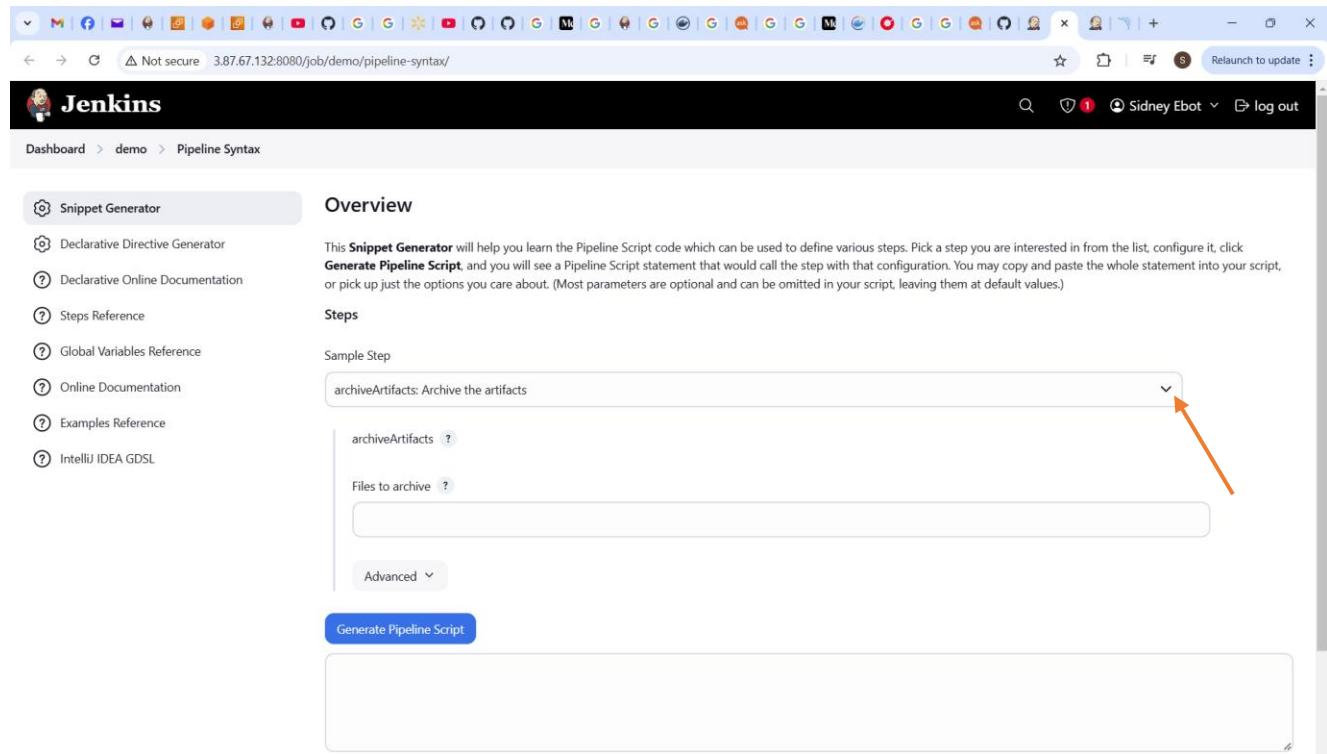
Global credentials (unrestricted)

[+ Add Credentials](#)

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

ID	Name	Kind	Description	
189474c2-9633-4663-8ab1-35f87fa4d12a	sonar	Secret text	sonar	
6bf02017-4de0-46f0-a8dc-b4c16bb2e882	ebotsmith/******** (docker)	Username with password	docker	
a006a966-34b9-41bd-a28d-bec775042789	kubeconfig.txt (k8s)	Secret file	k8s	

Icon: S M L

Go back to the “Pipeline Syntax” and refresh the page


The screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. On the left, there's a sidebar with links like Snippet Generator, Declarative Directive Generator, Declarative Online Documentation, Steps Reference, Global Variables Reference, Online Documentation, Examples Reference, and IntelliJ IDEA GDSL. The main area has a title 'Overview'. It contains a description of the Snippet Generator, a 'Steps' section with a dropdown menu currently showing 'archiveArtifacts: Archive the artifacts', and a 'Generate Pipeline Script' button. Below the button is a large text area where the generated pipeline script is displayed. The status bar at the bottom shows system information like battery level, signal strength, and date/time.

Click on the drop down on “Sample Step”, look for “**withKubeCredentials.....**”, Under “**Credentials**”, select the “**kubeconfig**” file we just saved.

This screenshot shows the Jenkins Pipeline Syntax Snippet Generator. On the left, a sidebar titled "Snippet Generator" lists various documentation links. The main area is titled "Overview" and contains a "Sample Step" section for "withKubeCredentials". The configuration form includes fields for "Credentials to use" (set to "kubeconfig.txt (k8s)"), "Kubernetes API endpoint", and "Cluster name". A note at the bottom states: "This Snippet Generator will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click Generate Pipeline Script, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)"

Scroll down

This screenshot shows the continuation of the Jenkins Pipeline Syntax Snippet Generator. It displays a large, empty text area for the generated pipeline script. Above this area, there is a section for "Certificate of certificate authority" with an "Add" button and a checkbox for "Restrict access to kubeconfig file". At the bottom of the page, under "Global Variables", there is a note: "There are many features of the Pipeline that are not steps. These are often exposed via global variables, which are not supported by the snippet generator. See the Global Variables Reference for details." The Jenkins status bar at the bottom right indicates "Jenkins 2.489".

Click on “Generate Pipeline Script”

Certificate of certificate authority

Add

Restrict access to kubeconfig file

Generate Pipeline Script

```
withKubeCredentials(kubectlCredentials: [[caCertificate: "", clusterName: "", contextName: "", credentialsId: "a006a966-34b9-41bd-a28d-bee775042789", namespace: "", serverUrl: ""]]) {
    // some block
}
```

Global Variables

There are many features of the Pipeline that are not steps. These are often exposed via global variables, which are not supported by the snippet generator. See the [Global Variables Reference](#) for details.

Copy the generated script and use in our next Pipeline stage

Go to Dashboard -> “demo” -> Configure

General

Build Triggers

Advanced Project Options

Pipeline

Definition

Pipeline script

Script ?

```
1 pipeline {
2     agent any
3
4     tools{
5         jdk 'jdk'
6         maven 'maven3'
7     }
8     environment{
9         SCANNER_HOME=tool 'sonar-scanner'
10    }
11
12    stages {
13        stage('Git Checkout') {
14            steps {
15                git 'https://github.com/ebotsmith2000/end-to-end.git'
16            }
17        }
18    }
19}
```

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

Add a new stage to the pipeline script and we will use the copied generated script here. Then modify the script.

But let us first go to our GitHub repository

ebotsmith2000 / end-to-end

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

end-to-end Public

master Branch Tags

Go to file Add file Code

ebotsmith2000 Initial commit 4512f0e - 3 hours ago 1 Commit

k8s Initial commit 3 hours ago

src/main/java/com/example/demoapp Initial commit 3 hours ago

Dockerfile Initial commit 3 hours ago

Jenkinsfile Initial commit 3 hours ago

pom.xml Initial commit 3 hours ago

README

Add a README

Help people interested in this repository understand your project by adding a README.

Add a README

About

No description, website, or topics provided.

Activity 0 stars 1 watching 0 forks

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

Languages

Java 83.8% Dockerfile 16.2%

Suggested workflows

Based on your tech stack

Android CI Configure

Build an Android project with Gradle.

Click on “k8s” folder

ebotsmith2000 / end-to-end

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

master

Go to file

k8s

deployment.yaml

src

Dockerfile

Jenkinsfile

pom.xml

end-to-end / k8s /

ebotsmith2000 Initial commit 4512f0e - 3 hours ago History

Name Last commit message Last commit date

..

deployment.yaml Initial commit 3 hours ago

Click on “deployment.yaml” file

A screenshot of a GitHub repository page for 'ebotsmith2000/end-to-end'. The 'deployment.yaml' file is open in the code editor. A red arrow points to the 'Edit' icon in the top right corner of the code preview area.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo-app
  template:
    metadata:
      labels:
        app: demo-app
    spec:
      containers:
        - name: demo-app
          image: gashok13193/end-to-end:22
          ports:
            - containerPort: 8080
...
apiVersion: v1
kind: Service
metadata:
  name: demo-app
spec:
  selector:
    app: demo-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
      type: NodePort
```

Edit the highlighted line in this script to match with your “**docker Image**”. You will click on “**Edit**”

Open your Docker hub

A screenshot of the Docker Hub website showing a repository named 'ebotsmith/demo'. The 'Tags' tab is selected. A red arrow points to the 'Copy' button next to the tag '15'. The tag details show it was last pushed 7 minutes ago by 'ebotsmith'.

TAG	Digest	OS/ARCH	Last pull	Compressed size
15	0668a9f8f0e6	linux/amd64	7 minutes ago	224.4 MB

Copy this: **ebotsmith/demo:15**

Paste it in the highlighted part of the “**deployment.yaml**” file above

A screenshot of a web browser displaying a GitHub repository page. The URL is `github.com/ebotsmith2000/end-to-end/edit/master/k8s/deployment.yaml`. The page shows a code editor with the file `deployment.yaml` open. The code defines a Kubernetes Deployment and Service. A red arrow points from the bottom right towards the green **Commit changes...** button in the top right corner of the editor. The status bar at the bottom indicates the file is 55% faster with GitHub Copilot.

Then click on “Commit Changes”

A screenshot of a web browser displaying a GitHub repository page. The URL is `github.com/ebotsmith2000/end-to-end/edit/master/k8s/deployment.yaml`. The page shows a code editor with the file `deployment.yaml` open. A modal dialog box titled "Commit changes" is displayed. The "Commit message" field contains the text "Update deployment.yaml". A red arrow points from the bottom right towards the green **Commit changes** button in the bottom right corner of the dialog. The status bar at the bottom indicates the file is 55% faster with GitHub Copilot.

Click on “Commit Changes” again

github.com/ebotsmith2000/end-to-end/blob/master/k8s/deployment.yaml

ebotsmith2000 / end-to-end

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

master Go to file

k8s

- deployment.yaml
- src
- Dockerfile
- Jenkinsfile
- pom.xml

end-to-end / k8s / deployment.yaml

ebotsmith2000 · Update deployment.yaml · 599f86a · 1 minute ago · History

Code Blame 32 lines (32 loc) · 583 Bytes · Code 55% faster with GitHub Copilot

```

1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: demo-app
5 spec:
6   replicas: 1
7   selector:
8     matchLabels:
9       app: demo-app
10    template:
11      metadata:
12        labels:
13          app: demo-app
14        spec:
15          containers:
16            - name: demo-app
17              image: ebotsmith/cicd:7
18            ports:
19              - containerPort: 8080
20 ...
21 apiVersion: v1
22 kind: Service
23 metadata:
24   name: demo-app
25 spec:
26   selector:
27     app: demo-app
28   ports:
29     - protocol: TCP
30       port: 80
31       targetPort: 8080
32   type: NodePort

```

Copy the highlighted part above: **k8s/deployment.yaml**

Then go back to Jenkins GUI to complete the modification of the Pipeline Script

Pipeline

Definition

Pipeline script

Script

```

55
56 }
57 }
58 }
59 }
60 stage('Deploy to k8s') {
61   steps {
62     script{
63       withKubeCredentials(kubectlCredentials: [[credentialsId: 'a006a966-34b9-41bd-a28d-bec775042789']]) {
64       sh 'kubectl apply -f k8s/deployment.yaml'
65     }
66   }
67 }

```

Use Groovy Sandbox

Pipeline Syntax

Save **Apply**

```
pipeline {
  agent any
```

```

tools{
    jdk 'jdk'
    maven 'maven3'
}
environment{
    SCANNER_HOME= tool 'sonar-scanner'
}

stages {
    stage('Git Checkout') {
        steps {
            git 'https://github.com/ebotsmith2000/jenkins-pipeline.git'
        }
    }
    stage('Code Compile') {
        steps {
            sh 'mvn clean compile'
        }
    }
    stage('SonarQube Analysis') {
        steps {
            withSonarQubeEnv('SonarQube'){
                sh '''$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=new \
-Dsonar.java.binaries=. \
-Dsonar.projectKey=new '''
            }
        }
    }
    stage('Code Build') {
        steps {
            sh 'mvn clean install'
        }
    }
    stage('Docker Build') {
        steps {
            script{
                withDockerRegistry(credentialsId: '7e596026-5cee-4b1f-9d13-97efcaf07959', toolName: 'docker')
{
                    sh 'docker build -t demo .'
                }
            }
        }
    }
    stage('Docker Image Push') {
        steps {
            script{
                withDockerRegistry(credentialsId: '7e596026-5cee-4b1f-9d13-97efcaf07959', toolName: 'docker')
{
                    sh 'docker tag demo ebotsmith/demo:$BUILD_ID'
                    sh 'docker push ebotsmith/demo:$BUILD_ID'
                }
            }
        }
    }
    stage('Deploy to k8s') {
        steps {
            script{
                withKubeCredentials(kubectlCredentials: [[caCertificate: '', clusterName: '', contextName: '',
                credentialsId: '10e17ebc-03e7-441a-b922-32530142f75f', namespace: '', serverUrl: '']]) {
                    sh 'kubectl apply -f k8s/deployment.yaml'
                }
            }
        }
    }
}

```

Then click on “Save”

Let us build the pipeline again by clicking on “Build Now”

The screenshot shows the Jenkins interface for a pipeline named 'demo'. In the sidebar, there is a 'Build Now' button with an orange arrow pointing to it. The main area is titled 'Stage View' and displays a grid of stages for five builds (#6 to #10). The columns represent different stages: Declarative: Tool Install, Git Checkout, Code Compile, SonarQube Analysis, Code Build, Docker Build, Docker Image Push, and Deploy to k8s. Build #6 failed at the Docker Build stage. Build #7 failed at the Docker Image Push stage. Build #8 failed at the Deploy to k8s stage. Build #9 failed at the Deploy to k8s stage. Build #10 failed at the Deploy to k8s stage.

	Declarative: Tool Install	Git Checkout	Code Compile	SonarQube Analysis	Code Build	Docker Build	Docker Image Push	Deploy to k8s
#6	3s	574ms	5s	12s	7s	4s	1s	78ms
#7	284ms	786ms	7s	13s	7s	546ms failed	166ms failed	78ms failed
#8	192ms	461ms	4s	10s	6s	1s	2s	
#9	176ms	522ms	4s	10s	6s	11s		
#10	177ms	497ms	4s	11s	10s			
	204ms	591ms	4s	15s				

The build has failed. Let us go to the “Console Output”. Below is the warning message:

Warning: failed to get default registry endpoint from daemon (Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?). Using system default: <https://index.docker.io/v1/>

Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?

It looks like docker is having issues.

SOLUTION

To resolve this

- reboot the EC2 instance.
- Reinstall SonarQube
- Reinstall Docker

The build again

And the build is successful

Then, navigate to the kube folder using the below commands and check if the kube pod is available.

Run the command: **sudo su**

Followed by the command: **cd ~**

Then the command: **cd .kube/**

Now, run the command to get the pods:

kubectl get pods

```
ubuntu@ip-172-31-44-137:~$ sudo su
root@ip-172-31-44-137:/home/ubuntu# cd ~
root@ip-172-31-44-137:~# cd .kube/
root@ip-172-31-44-137:~/._kube# kubectl get pods
NAME                  READY   STATUS        RESTARTS   AGE
demo-app-6c857b5989-m7nrc   0/1   CrashLoopBackOff   7 (88s ago)   12m
root@ip-172-31-44-137:~/._kube#
```

And check the code analysis on SonarQube.

The screenshot shows the SonarQube interface for a single project. On the left, there's a sidebar with 'My Favorites' and 'All' buttons, followed by 'Filters' sections for 'Quality Gate', 'Reliability', and 'Security'. The main area displays a summary for a 'new PUBLIC' project. It shows 1 passed issue and 0 failed issues under 'Quality Gate'. Below this, reliability and security scores are listed. The summary includes metrics for Bugs, Vulnerabilities, Hotspots Reviewed, Code Smells, Coverage, and Duplications. A note at the bottom says 'Last analysis: 2 minutes ago - 80 Lines of Code · YAML, XML, ...'. A status bar at the bottom right indicates '1 project(s) 1 of 1 shown'.

Everything passed

The screenshot shows the SonarQube interface for the 'Issues' tab. The left sidebar has 'My Issues' and 'All' buttons, followed by 'Filters' sections for 'Clean Code Attribute', 'Software Quality', 'Severity', and 'Type'. The main area displays a message 'No Issues. Hooray!' and includes buttons for 'Bulk Change', 'Select issues', 'Navigate to issue', and counters for '0 issues' and '0 effort'. A note at the bottom says 'Embedded database should be used for evaluation purposes only'.

You can see there is no issue

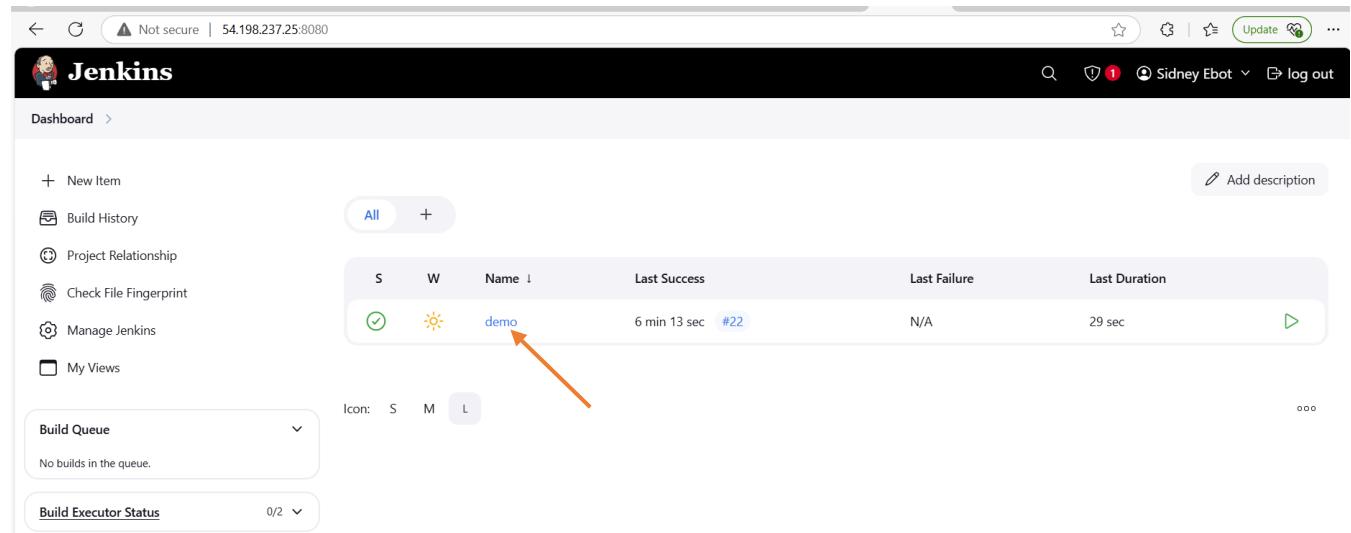
STEP 16: Automating the process

The final step is to automate the pipeline so that whenever any change is made in the GitHub repository and the changes are committed. The Jenkins pipeline is triggered to run automatically.

To achieve this, we have to make two changes. One in Jenkins and the other in GitHub.

PART 1: Modification on Jenkins

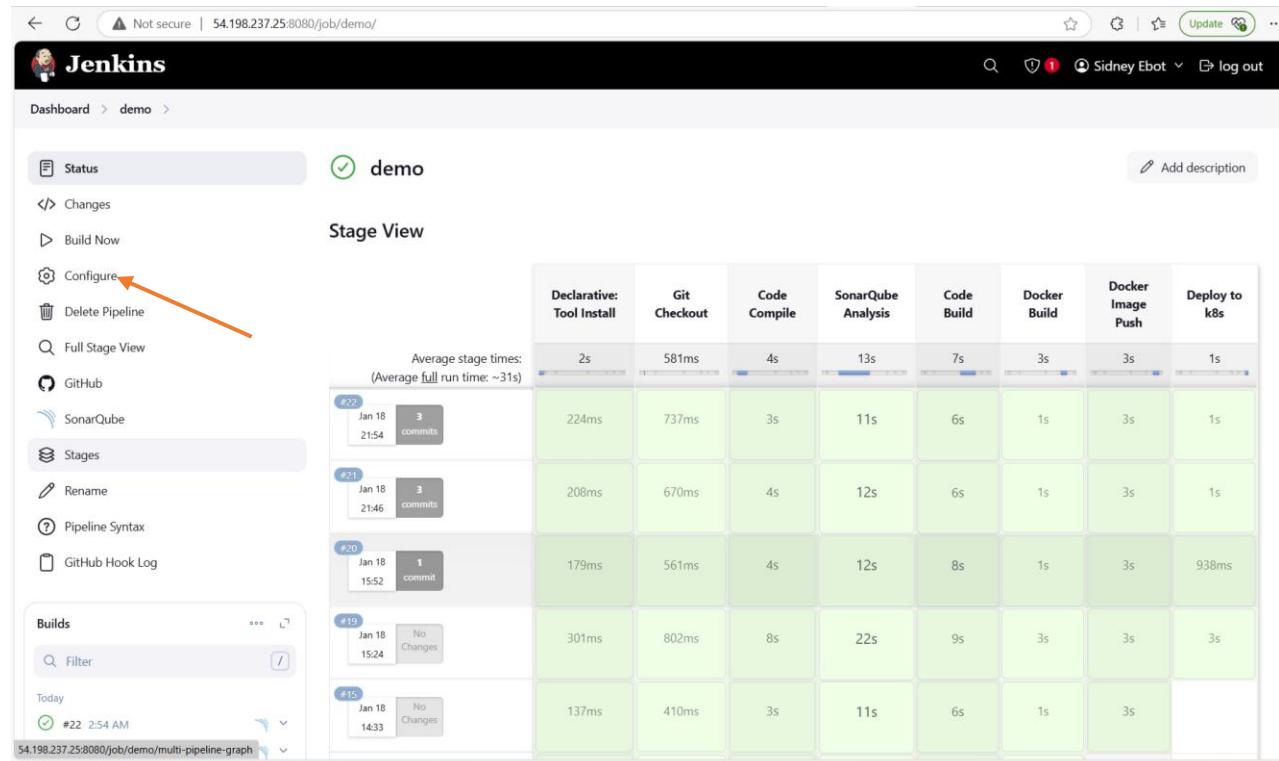
Go to Jenkins Dashboard



The screenshot shows the Jenkins dashboard at 54.198.237.25:8080. The 'demo' pipeline is listed in the main table with a green checkmark icon, a yellow sun icon, and the name 'demo'. An orange arrow points from the text 'Click on the Pipeline "demo"' to the pipeline name in the table.

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	demo	6 min 13 sec #22	N/A	29 sec

Click on the Pipeline “demo”



The screenshot shows the Jenkins pipeline configuration page for 'demo'. On the left, there's a sidebar with options like Status, Changes, Build Now, Configure (which is highlighted with an orange arrow), Delete Pipeline, Full Stage View, GitHub, SonarQube, Stages, Rename, Pipeline Syntax, and GitHub Hook Log. The main area shows the 'Stage View' with a table of stages and their execution times. The table has columns for Declarative: Tool Install, Git Checkout, Code Compile, SonarQube Analysis, Code Build, Docker Build, Docker Image Push, and Deploy to k8s. Each row represents a build (e.g., #22, #21, #20, #19, #18) with its commit details and stage durations.

Declarative: Tool Install	Git Checkout	Code Compile	SonarQube Analysis	Code Build	Docker Build	Docker Image Push	Deploy to k8s
2s	581ms	4s	13s	7s	3s	3s	1s
224ms	737ms	3s	11s	6s	1s	3s	1s
208ms	670ms	4s	12s	6s	1s	3s	1s
179ms	561ms	4s	12s	8s	1s	3s	930ms
301ms	802ms	8s	22s	9s	3s	3s	3s
137ms	410ms	3s	11s	6s	1s	3s	

Click on “Configure”

The screenshot shows the Jenkins configuration interface for a job named "demo". The "General" tab is selected. In the configuration section, there is a checkbox labeled "GitHub project" which is currently unchecked. An orange arrow points to this checkbox. Below the checkbox, there is a field for "Project url" containing the value "https://github.com/ebotsmith2000/jenkins-pipeline/". At the bottom of the page, there are two buttons: "Save" and "Apply".

Under “General”, select “GitHub project” and enter the URL of the GitHub repository

The screenshot shows the Jenkins configuration interface for a job named "demo". The "General" tab is selected. The "GitHub project" checkbox is checked, and its value is highlighted with an orange arrow. The "Project url" field contains the value "https://github.com/ebotsmith2000/jenkins-pipeline/". At the bottom right, there is a "Save" button with an orange arrow pointing to it.

Click on “Save”

Now, we are going to modify the trigger. Click on “Configure” again

Click on “Triggers” on the left-hand side

The screenshot shows the Jenkins 'Configuration' page for a job named 'demo'. The 'Triggers' tab is selected. Under the 'Triggers' heading, there is a sub-section titled 'Set up automated actions that start your build based on specific events, like code changes or scheduled times.' A list of triggers is shown with checkboxes:

- Build after other projects are built
- Build periodically
- Generic Webhook Trigger
- GitHub hook trigger for GITScm polling
- Poll SCM
- Trigger builds remotely (e.g., from scripts)

An orange arrow points to the checkbox for 'GitHub hook trigger for GITScm polling'.

Below the triggers section is a 'Pipeline' section with a 'Definition' dropdown set to 'Pipeline script'. A code editor window shows a Groovy pipeline script:

```
1 * pipeline {
2     agent any
3
4     tools{
5         jdk 'jdk'
6         maven 'maven3'
7     }
8     environment{
9         SCANNER_HOME= tool 'sonar-scanner'
10    }
11}
```

At the bottom of the pipeline editor are 'Save' and 'Apply' buttons.

Under “Triggers”, select “GitHub hook trigger for GITScm polling”

The screenshot shows the same Jenkins 'Configuration' page for the 'demo' job. The 'Triggers' tab is selected. The 'GitHub hook trigger for GITScm polling' checkbox is now checked (indicated by a blue square). All other triggers are unchecked.

Below the triggers section is a 'Pipeline' section with a 'Definition' dropdown set to 'Pipeline script'. A code editor window shows the same Groovy pipeline script as the previous screenshot.

An orange arrow points from the text above to the 'Save' button at the bottom of the pipeline editor.

Click on “Save”

Stage View

Declarative: Tool Install	Git Checkout	Code Compile	SonarQube Analysis	Code Build	Docker Build	Docker Image Push	Deploy to k8s
2s	581ms	4s	13s	7s	3s	3s	1s
224ms	737ms	3s	11s	6s	1s	3s	1s
208ms	670ms	4s	12s	6s	1s	3s	1s
179ms	561ms	4s	12s	8s	1s	3s	938ms
301ms	802ms	8s	22s	9s	3s	3s	3s
137ms	410ms	3s	11s	6s	1s	3s	

PART 2: Modification on GitHub Repository

Now, we have to modify the GitHub repository. Go to the GitHub repository

Settings

About
No description, website, or topics provided.

Releases
No releases published
[Create a new release](#)

Packages
No packages published
[Publish your first package](#)

Languages
Java 83.8% Dockerfile 16.2%

Suggested workflows
Based on your tech stack

Java with Ant
Configure
Build and test a Java project with Apache Ant.

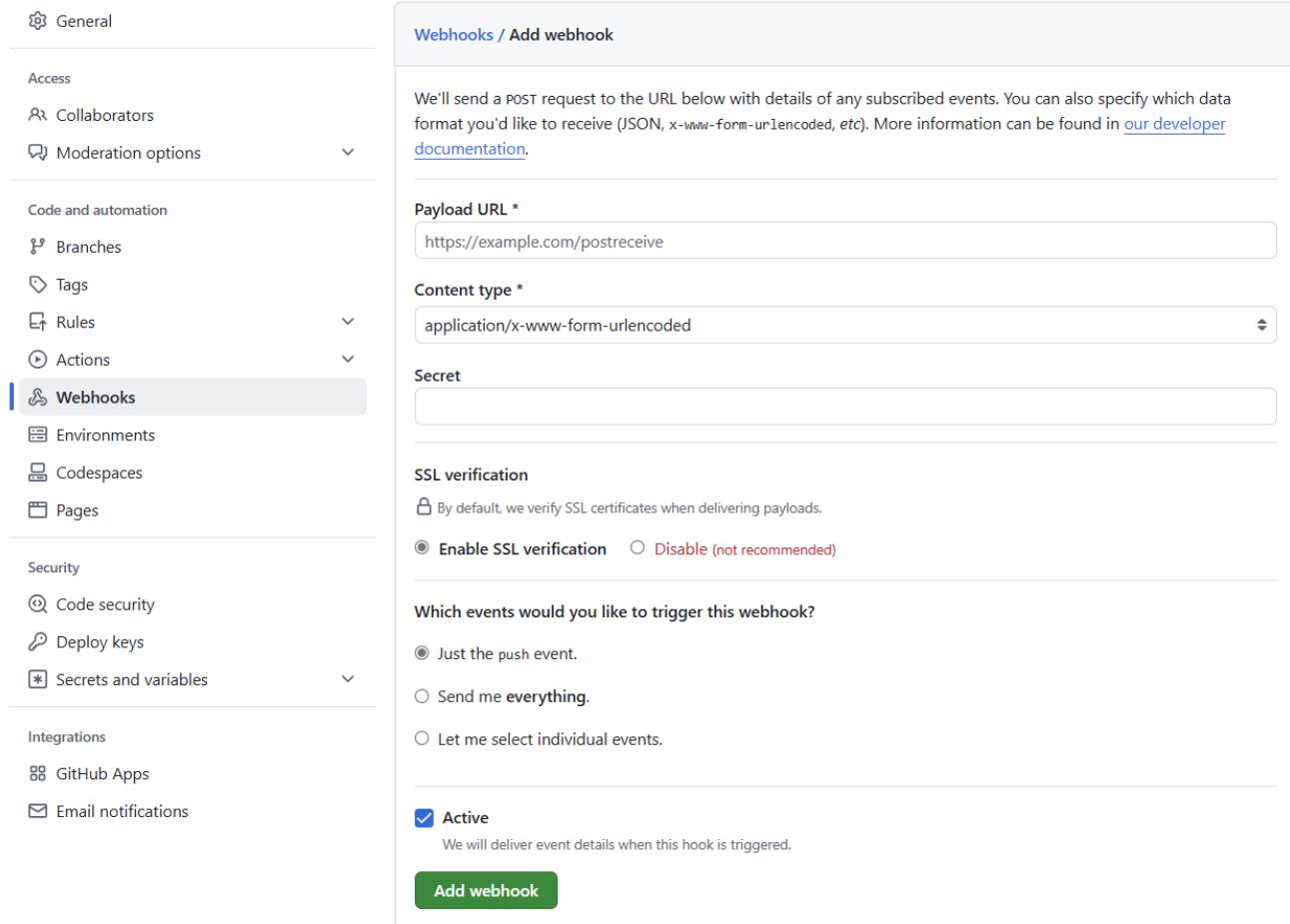
Click on “Settings”

This screenshot shows the 'General' settings page for a GitHub repository named 'jenkins-pipeline'. The left sidebar lists various configuration sections: Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks), Security (Code security, Deploy keys, Secrets and variables), Integrations (GitHub Apps, Email notifications). A red arrow points to the 'Webhooks' link in the 'Code and automation' section. The main content area displays the 'General' settings, including fields for Repository name (jenkins-pipeline) and 'Template repository', and sections for 'Default branch' (set to master) and 'Social preview'.

Click on “webhook” on the left-hand side

This screenshot shows the 'Webhooks' configuration page for the same repository. The left sidebar is identical to the previous screenshot. The main content area is titled 'Webhooks' and contains a brief description: 'Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#)'. A red arrow points to the 'Add webhook' button at the top right of the page.

Click on “Add Webhook”



The screenshot shows the GitHub repository settings interface. On the left, a sidebar lists various configuration sections: General, Access, Collaborators, Moderation options, Code and automation, Branches, Tags, Rules, Actions, Webhooks (which is selected and highlighted in blue), Environments, Codespaces, Pages, Security, Code security, Deploy keys, Secrets and variables, Integrations, GitHub Apps, and Email notifications.

The main content area is titled "Webhooks / Add webhook". It contains the following fields:

- Payload URL ***: https://example.com/postreceive
- Content type ***: application/x-www-form-urlencoded
- Secret**: (empty field)
- SSL verification**:
 - By default, we verify SSL certificates when delivering payloads.
 - Enable SSL verification**
 - Disable (not recommended)**
- Which events would you like to trigger this webhook?**
 - Just the push event.
 - Send me **everything**.
 - Let me select individual events.
- Active**:
We will deliver event details when this hook is triggered.

A green "Add webhook" button is located at the bottom right of the form.

On Payload URL: Enter the Jenkins URL/github-webhook/

That is http://54.198.237.25:8080/github-webhook/

On Content Type: Select **application/json**

The screenshot shows the GitHub settings interface for a repository. On the left, a sidebar lists various configuration sections: General, Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks), Environments, Codespaces, Pages, Security (Code security, Deploy keys, Secrets and variables), and Integrations (GitHub Apps, Email notifications). The 'Webhooks' section is currently selected and highlighted.

The main content area is titled 'Webhooks / Add webhook'. It contains instructions: "We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#)".

Form fields include:

- Payload URL ***:
- Content type ***:
- Secret**: A text input field.

SSL verification:
By default, we verify SSL certificates when delivering payloads.
 Enable SSL verification **Disable (not recommended)**

Which events would you like to trigger this webhook?

- Just the push event.**
- Send me everything.**
- Let me select individual events.**

Active
We will deliver event details when this hook is triggered.

Add webhook

Click on “Add webhook”

The screenshot shows the GitHub repository settings page for a repository named 'jenkins-pipeline'. The left sidebar contains navigation links for General, Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions), Webhooks (selected), Environments, Codespaces, Pages, Security (Code security, Deploy keys, Secrets and variables), Integrations (GitHub Apps, Email notifications), and Help.

Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

This hook has never been triggered.

[Edit](#) [Delete](#)

One webhook is listed:

- <http://54.198.237.25:8080/github-w...> (push)

Now, let us test if the pipeline has been automated.

TESTING

Let me add a new text file called “**test3.txt**” to our GitHub repository. Then commit to see if the pipeline will be triggered automatically.

The screenshot shows the GitHub repository 'jenkins-pipeline' with 1 branch and 0 tags. The commit history is as follows:

- ebotsmith2000 Create test2 (fd690ba · 27 minutes ago)
- k8s Update deployment.yaml (7 hours ago)
- src/main/java/com/example/demoapp Initial commit (12 hours ago)
- Dockerfile Initial commit (12 hours ago)
- Jenkinsfile Initial commit (12 hours ago)
- pom.xml Initial commit (12 hours ago)
- test2 Create test2 (27 minutes ago)

A red arrow points to the 'Add file' button in the top right corner of the commit list.

jenkins-pipeline [Public](#)

[master](#) [1 Branch](#) [0 Tags](#) [Go to file](#) [Add file](#) [Code](#)

Author	Commit Message	Time Ago
ebotsmith2000	Create test2	fd690ba · 27 minutes ago
	k8s	Update deployment.yaml
	src/main/java/com/example/demoapp	Initial commit
	Dockerfile	Initial commit
	Jenkinsfile	Initial commit
	pom.xml	Initial commit
	test2	Create test2

[README](#)

Click on “**Add file**” and select “**create a new file**”

The screenshot shows the GitHub code editor interface. A new file named 'test3' is being created in the 'jenkins-pipeline' repository. The file contains the single line: '1 This is my test for automation'. The GitHub Copilot feature is visible with the message 'Code 55% faster with GitHub Copilot'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings.

Let us add the file called “**test3**”

The screenshot shows the GitHub code editor interface again, this time with an orange arrow pointing to the green 'Commit changes...' button in the top right corner of the commit dialog. The file 'test3' now contains the text '1 This is my test for automation'.

Click on “**Commit Changes**” and we head back to our Jenkins GUI

The screenshot shows the Jenkins pipeline stage view for the 'demo' pipeline. Build #23 is currently running, indicated by a progress bar. The stage view table shows the following data:

Declarative: Tool Install	Git Checkout	Code Compile	SonarQube Analysis	Code Build	Docker Build	Docker Image Push	Deploy to k8s
1s	585ms	4s	13s	7s	3s	3s	1s
212ms	618ms	3s					

A tooltip for the SonarQube analysis step shows the status as 'Success' with a 'Logs' button. The Jenkins sidebar on the left includes links for Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, GitHub, SonarQube, Stages, Rename, Pipeline Syntax, and GitHub Hook Log. The Builds section shows the last two completed builds: #23 at 3:24 AM and #22 at 2:54 AM.

You can see that the Jenkins pipeline has started running automatically

Jenkins

Dashboard > demo >

Status demo Add description

</> Changes

Build Now

Configure

Delete Pipeline

Full Stage View

GitHub

SonarQube

Stages

Rename

Pipeline Syntax

GitHub Hook Log

Builds

Filter

Average stage times:
(Average full run time: ~31s)

	Declarative: Tool Install	Git Checkout	Code Compile	SonarQube Analysis	Code Build	Docker Build	Docker Image Push	Deploy to k8s
#23 Jan 18 22:24	1s	585ms	4s	13s	7s	3s	3s	1s
#22 Jan 18 21:54	212ms	618ms	3s	11s	6s	1s	2s	1s
#21 Jan 18 21:46	224ms	737ms	3s	11s	6s	1s	3s	1s
#20 Jan 18 15:52	208ms	670ms	4s	12s	6s	1s	3s	1s
#19 Jan 18 15:24	179ms	561ms	4s	12s	8s	1s	3s	938ms
No Changes	301ms	802ms	8s	22s	9s	3s	3s	3s

You can see that the pipeline is successful. So, we have made an automated CI/CD pipeline on Jenkins