

Deploying Java Application using Jenkins CI/CD Pipeline

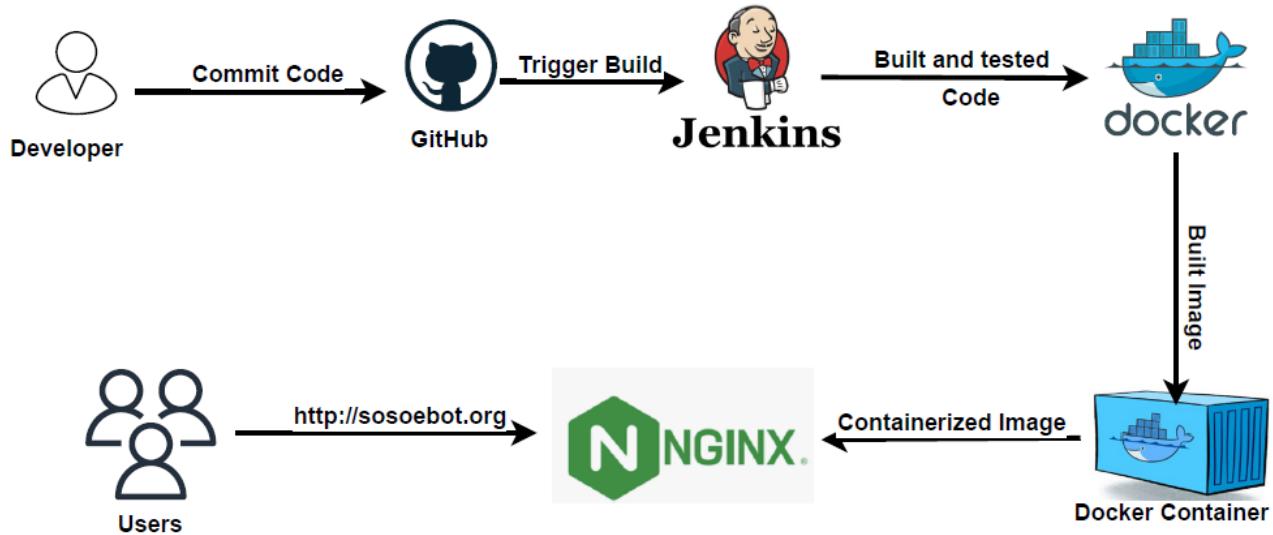
In this comprehensive guide, we dive into the world of building, testing, and deploying Java applications using CI/CD pipelines. Whether you are a beginner or an experienced developer looking to streamline your workflow, this series has something for you.

In these tutorials, we will cover:

- Setting up a Java development environment
- Integrating version control with Git
- Automating builds using Maven
- Writing effective unit tests with JUnit
- Configuring CI/CD pipelines with tools like Jenkins
- Pipeline Scripting
- Deploying your Java application to various environments (development)

By the end of this tutorial, you'll have a solid understanding of how to leverage CI/CD to automate repetitive tasks, ensure code quality, and deliver software more reliably. Join us on this journey to mastering CI/CD for Java applications.

Architecture



The developer writes the code and pushes it to **GitHub**. When the code is pushed, it will trigger **Jenkins**, the CI/CD tool that will pick up the code from GitHub and build it using **Maven**. Maven will build the application and create a file in the format war or jar file.

Maven is a build automation tool primarily used for Java projects. It simplifies the process of building and managing Java projects by providing a uniform build system, project management, and dependency management.

After building the code, it will be sent to Docker. Docker will build a docker image and containerize it. Then containerized image will be deployed to the EC2 instance, that is our server. From this server the users can access the application through the internet.

TOOLS

This project will be executed using these tools:

- Maven: Build tool
- GitHub: SCM
- Jenkins: CI/CD
- Docker: Container
- Nginx: Webserver
- SSL: Cerbot

IMPLEMENTATION

We will now implement the above architecture following these steps:

1. Launch an ubuntu EC2 instance that will serve as our server

- Name: Jenkins-Server
- Enable Port 22 for SSH
- Enable Port 43 for HTTPS
- Enable Port 80 for HTTP
- Add Port 8080 for Jenkins
- Add Port 9000 for Reverse Proxy
- SSH Connect to the server

2. Install JDK and Maven on EC2 server

3. Install Docker on EC2 server

4. Install Jenkins on EC2 server

5. Setup Docker in Jenkins

6. Access Jenkins on Browser

7. Create Pipeline on Jenkins

- Test “Hello World” pipe line

8. Install and Configure JDK on Jenkins

- Install Java JDK plugins on Jenkins
- Install Java JDK as a tool on Jenkins

9. Install and Configure Maven on Jenkins

- Install Maven plugins on Jenkins
- Install Maven as a tool on Jenkins

10. Install and Configure Docker on Jenkins

- Install Docker plugins on Jenkins
- Install Docker as a tool on Jenkins

11. Add Stages to the pipeline and build:

- Git Checkout Stage
- Code Compile Stage
- Maven Build Application
- Docker Build & Push
- Deploy

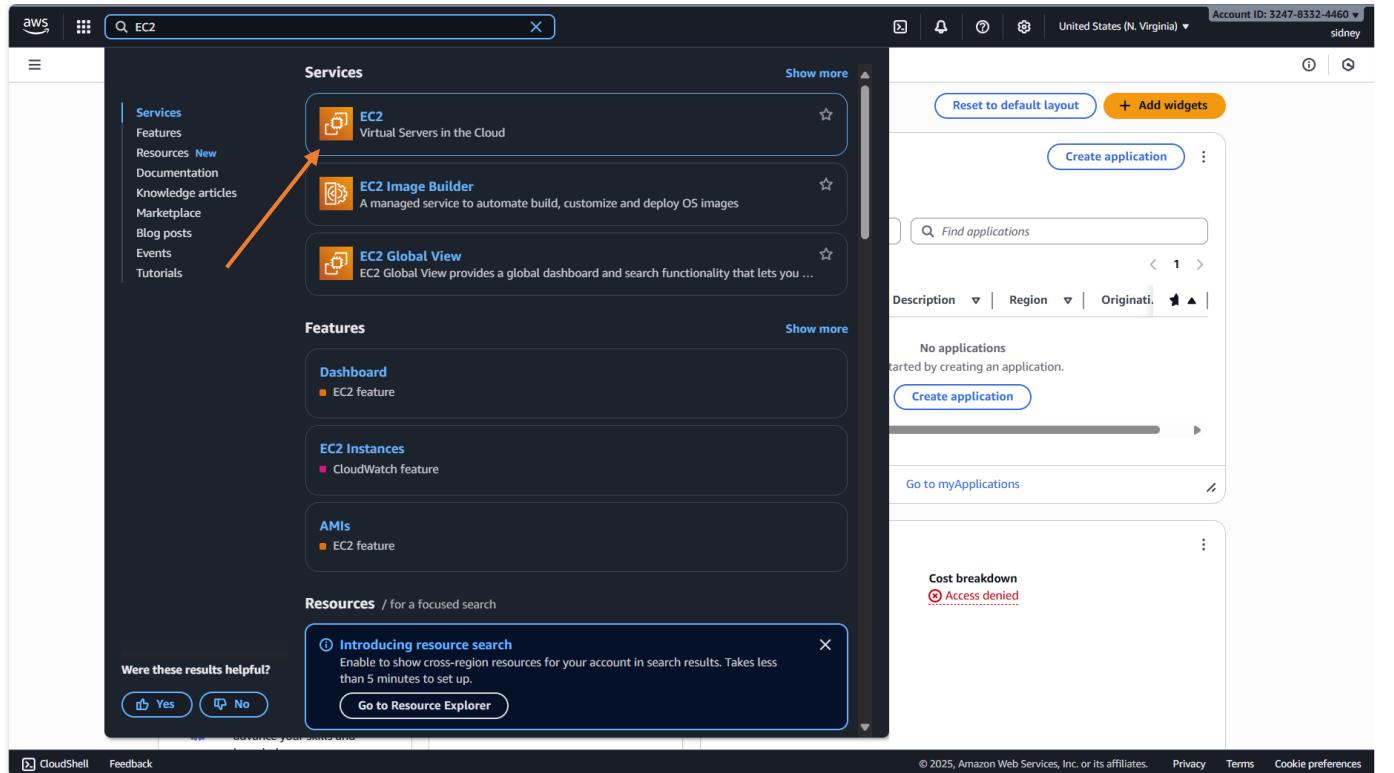
12. Install and configure Nginx and Certbot on EC2 server for Reverse Proxy

STEP 1: Launch an Ubuntu EC2 instance that will serve as a server

We will launch an EC2 instance and later on SSH connect to it to install our dependencies.

Part 1: Launch an Ubuntu EC2 instance

Go to AWS Management console and search for “EC2” Instance.



Click on “EC2”

AWS | Search [Alt+S] | Account ID: 3247-8332-4460 | sidney

EC2

- Dashboard
- EC2 Global View
- Events
- Instances**
 - Instances
 - Instance Types
 - Launch Templates
 - Spot Requests
 - Savings Plans
 - Reserved Instances
 - Dedicated Hosts
 - Capacity Reservations
- Images**
 - AMIs
 - AMI Catalog
- Elastic Block Store**
 - Volumes
 - Snapshots
 - Lifecycle Manager
- Network & Security**
 - Security Groups
 - Elastic IPs
 - Placement Groups
 - Key Pairs
 - Network Interfaces

Compute

Amazon Elastic Compute Cloud (EC2)

Create, manage, and monitor virtual servers in the cloud.

Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest compute platform, with over 600 instance types and a choice of the latest processors, storage, networking, operating systems, and purchase models to help you best match the needs of your workload.

Launch a virtual server

- Launch instance
- View dashboard
- Get started walkthroughs
- Get started tutorial

Benefits and features

EC2 offers ultimate scalability and control

Fully resizable compute capacity to support virtually any workload. This service is best if you want:

- Highest level of control of the entire technology stack, allowing full integration with all AWS services
- Wide variety of server size options
- Wide availability of operating systems to choose from including Linux, Windows, and macOS
- Global scalability

[Find out more about EC2](#)

Use cases

Additional actions

- View running instances
- Migrate a server

Pricing (US)

- EC2 pricing options
- Use the AWS pricing calculator
- Manage budgets

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Click on “Launch Instance”

AWS | Search [Alt+S] | Account ID: 3247-8332-4460 | sidney

EC2 > Instances > Launch an 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

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

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

Recent AMIs

- Amazon Linux
- macOS
- Ubuntu
- Windows
- Red Hat
- SUSE Linux
- Debian

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI
 ami-0ca32bbc84273381 (64-bit (x86), uefi-preferred) / ami-0aa7db6294d00216f (64-bit (Arm), uefi)
 Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Amazon Linux 2023 (kernel-6.1) is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Summary

Number of instances | Info

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.8.2...[read more](#)
 ami-0ca32bbc84273381

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet. Data transfer charges are not included as part of the free tier allowance.

Cancel

Launch instance

Preview code

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

We will call the instance “jenkins-server”

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Name and tags' step, a 'Name' field contains 'jenkins-server'. A link 'Add additional tags' is visible.

Scroll down to “AMI”, select “ubuntu”

The screenshot shows the 'Application and OS Images (Amazon Machine Image)' section. Under the 'Quick Start' tab, the 'Ubuntu' icon is selected. A search bar at the top says 'Search our full catalog including 1000s of application and OS images'. To the right, a button 'Browse more AMIs' is shown with the note 'Including AMIs from AWS, Marketplace and the Community'.

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type
ami-0360c520857e3138f (64-bit (x86)) / ami-026fcdd88446aa0bf (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

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	AMI ID	Publish Date	Username
64-bit (x86)	ami-0360c520857e3138f	2025-08-21	ubuntu

Free tier eligible ▾

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

The screenshot shows the 'Instance type' section. The 't2.medium' option is selected, showing details: Family: t2, 2 vCPU, 4 GiB Memory, Current generation: true. Pricing: On-Demand Ubuntu Pro base pricing: 0.0499 USD per Hour, On-Demand Linux base pricing: 0.0464 USD per Hour, On-Demand RHEL base pricing: 0.0752 USD per Hour, On-Demand Windows base pricing: 0.0644 USD per Hour, On-Demand SUSE base pricing: 0.1464 USD per Hour. A link 'Additional costs apply for AMIs with pre-installed software' is present. Buttons for 'All generations' and 'Compare instance types' are also shown.

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”

Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair

Private key file format

.pem For use with OpenSSH

.ppk For use with PuTTY

⚠️ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more ↗](#)

[Cancel](#) [Create key pair](#)

Give the key pair a name, I will call it “jenkins-key”

Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA RSA encrypted private and public key pair

ED25519 ED25519 encrypted private and public key pair

Private key file format

.pem For use with OpenSSH

.ppk For use with PuTTY

⚠️ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more ↗](#)

[Cancel](#) [Create key pair](#)

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**

jenkins-key

[Create new key pair](#)

Part 2: Scroll down to “Network Settings”. Select “Allow HTTP traffic from the internet”, “Allow HTTP traffic from the internet” and “Allow SSH traffic from”.

▼ Network settings [Info](#)

[Edit](#)

[Network](#) | [Info](#)

vpc-0128e9209eaef1c37

[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-5' 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 the end

▼ Configure storage [Info](#)

[Advanced](#)

1x GiB Root volume, 3000 IOPS, Not encrypted

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

[Firewall \(security group\)](#)

New security group

[Storage \(volumes\)](#)

1 volume(s) - 8 GiB

[Add new volume](#)

The selected AMI contains instance store volumes, however the instance does not allow any instance store volumes. None of the instance store volumes from the AMI will be accessible from the instance

[Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage \(or t3.micro where t2.micro isn't available\) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet. Data transfer charges are not included as part of the free tier allowance.](#)

[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.

0 x File systems

[Edit](#)

► Advanced details [Info](#)

[Cancel](#)

[Launch instance](#)

[Preview code](#)

Click on “Launch Instance”

A screenshot of the AWS EC2 Instances launch success page. At the top, there's a green success banner stating "Successfully initiated launch of instance (i-04d141c8588651c4d)". Below it, a "Launch log" button is visible. A red arrow points from the text "Click on 'Instances'" in the previous slide to the "EC2 > Instances" breadcrumb navigation link. The main area shows "Next Steps" with several cards: "Create billing and free tier usage alerts", "Connect to your instance", "Connect an RDS database", "Create EBS snapshot policy", "Manage detailed monitoring", "Create Load Balancer", "Create AWS budget", and "Manage CloudWatch alarms". Each card has a corresponding "Create" or "Manage" button.

Click on “Instances”

A screenshot of the AWS EC2 Instances list page. The left sidebar shows navigation links for EC2, Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, and Network Interfaces. The main content area shows a table titled "Instances (1) Info". It lists one instance: "jenkins-server" (Instance ID: i-04d141c8588651c4d), which is "Running" (Status check: Initializing). A red arrow points from the text "You can see that our just created instance is initializing. Wait for it to pass the “2/2 check”" in the previous slide to the "Status check" column of the table.

You can see that our just created instance is initializing. Wait for it to pass the “2/2 check”

The instance has passed the “2/2 check”

Part 3: Add Port 8080 for Jenkins

We will add ports 8080 for Jenkins as follows

Click on the “Security” tab

AWS Search [Alt+S] Account ID: 3247-8332-4460 sidney

EC2 Instances

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive) All states ▾

Name Instance ID Instance state Instance type Status check Alarm status Availability Zone

jenkins-server i-04d141c8588651c4d Running t2.medium 2/2 checks passed View alarms us-east-1d

i-04d141c8588651c4d (jenkins-server)

Details | Status and alarms | Monitoring | **Security** | Networking | Storage | Tags

Security details

IAM Role - Owner ID 324783324460 Launch time Sun Sep 07 2025 22:30:43 GMT-0400 (Eastern Daylight Time)

Security groups sg-05531be1e137790f3 (launch-wizard-14)

Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups
-	sgr-081af6ed4d3645aac	443	TCP	0.0.0.0/0	launch-wizard-14
-	sgr-02af2d3d2e8c50747	22	TCP	0.0.0.0/0	launch-wizard-14
-	sgr-06b79ef5cbce6330d	80	TCP	0.0.0.0/0	launch-wizard-14

Outbound rules

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Click on “Security Groups” url

AWS Search [Alt+S] Account ID: 3247-8332-4460 sidney

EC2 Security Groups > sg-05531be1e137790f3 - launch-wizard-14 Actions

sg-05531be1e137790f3 - launch-wizard-14

Details

Security group name launch-wizard-14 Security group ID sg-05531be1e137790f3 Description launch-wizard-14 created 2025-09-08T02:29:42.361Z VPC ID vpc-0128e9209eaef1c37

Owner 324783324460 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)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-081af6ed4d3645aac	IPv4	HTTPS	TCP	443	0.0.0.0/0
-	sgr-02af2d3d2e8c50747	IPv4	SSH	TCP	22	0.0.0.0/0
-	sgr-06b79ef5cbce6330d	IPv4	HTTP	TCP	80	0.0.0.0/0

Manage tags | Edit inbound rules

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Click on “Edit Inbound Rules”

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-081af6ed4d3645aac	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-02af2d3d2e8c50747	SSH	TCP	22	Custom	0.0.0.0/0
sgr-06b79ef5cbce6330d	HTTP	TCP	80	Custom	0.0.0.0/0

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.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-081af6ed4d3645aac	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-02af2d3d2e8c50747	SSH	TCP	22	Custom	0.0.0.0/0
sgr-06b79ef5cbce6330d	HTTP	TCP	80	Custom	0.0.0.0/0
-	Custom TCP	TCP	0	Custom	0.0.0.0/0

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-081af6ed4d3645aac	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-02af2d3d2e8c50747	SSH	TCP	22	Custom	0.0.0.0/0
sgr-06b79ef5cbce6330d	HTTP	TCP	80	Custom	0.0.0.0/0
-	Custom TCP	TCP	8080	Custom	0.0.0.0/0

Add rule

Cancel Preview changes Save rules

For source, click on the drop down and select “Anywhere-IPv4”

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-081af6ed4d3645aac	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-02af2d3d2e8c50747	SSH	TCP	22	Custom	0.0.0.0/0
sgr-06b79ef5cbce6330d	HTTP	TCP	80	Custom	0.0.0.0/0
-	Custom TCP	TCP	8080	Anywhere	0.0.0.0/0

Add rule

Save rules

Then click on “Save Rules”

Inbound rules (4)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-081af6ed4d3645aac	IPv4	HTTPS	TCP	443	0.0.0.0/0
-	sgr-02af2d3d2e8c50747	IPv4	SSH	TCP	22	0.0.0.0/0
-	sgr-06b79ef5cbce6330d	IPv4	HTTP	TCP	80	0.0.0.0/0
-	sgr-03bf01a10367bf92c	IPv4	Custom TCP	TCP	8080	0.0.0.0/0

Part 4: Add Port 9000 for Reverse Proxy

Now let us go to our EC2 server and add Port 9000

Instances (1/1) Info

i-04d141c8588651c4d (jenkins-server)

Details | Status and alarms | Monitoring | **Security** | Networking | Storage | Tags

Instance summary

Instance ID: i-04d141c8588651c4d
Public IPv4 address: 3.89.24.67 [open address]
Instance state: Running
Private IP DNS name (IPv4 only): ip-172-31-25-193.ec2.internal
Instance type: t2.medium
VPC ID: vpc-0128e9209eaef1c37
Subnet ID: subnet-09e0df5f5814abf3
Instance ARN: arn:aws:ec2:us-east-1:324783324460:instance/i-04d141c8588651c4d

Public IPv4 addresses: 172.31.25.193
Public DNS: ec2-3-89-24-67.compute-1.amazonaws.com [open address]

Elastic IP addresses: -
AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Auto Scaling Group name: -
Managed: false

Click on “Security Tab”

Instances (1/1) Info

i-04d141c8588651c4d (jenkins-server)

Details | Status and alarms | Monitoring | **Security** | Networking | Storage | Tags

Security details

IAM Role: -
Owner ID: 324783324460
Launch time: Sun Sep 07 2025 22:30:45 GMT-0400 (Eastern Daylight Time)

Security groups: sg-05531be1e137790f3 (launch-wizard-14)

Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups
-	sgr-081af6ed4d3645aac	443	TCP	0.0.0.0/0	launch-wizard-14
-	sgr-02af2d3d2e8c50747	22	TCP	0.0.0.0/0	launch-wizard-14
-	sgr-06b79ef5cbce6330d	80	TCP	0.0.0.0/0	launch-wizard-14
-	sgr-03bf01a10367bf92c	8080	TCP	0.0.0.0/0	launch-wizard-14

Click on the Security Group

Inbound security group rules successfully modified on security group (sg-05531be1e137790f3 | launch-wizard-14)

sg-05531be1e137790f3 - launch-wizard-14

Details

Security group name launch-wizard-14	Security group ID sg-05531be1e137790f3	Description launch-wizard-14 created 2025-09-08T02:29:42.361Z	VPC ID vpc-0128e9209eaef1c37
Owner 324783324460	Inbound rules count 4 Permission entries	Outbound rules count 1 Permission entry	

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

Inbound rules (4)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-081af6ed4d3645aac	IPv4	HTTPS	TCP	443	0.0.0.0/0
-	sgr-02af2d3d2e8c50747	IPv4	SSH	TCP	22	0.0.0.0/0
-	sgr-06b79ef5cbce6530d	IPv4	HTTP	TCP	80	0.0.0.0/0
-	sgr-03bf01a10367bf92c	IPv4	Custom TCP	TCP	8080	0.0.0.0/0

Manage tags Edit inbound rules

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Click on “Edit Inbound Rules”

Edit inbound rules Info

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

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-081af6ed4d3645aac	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-02af2d3d2e8c50747	SSH	TCP	22	Custom	0.0.0.0/0
sgr-06b79ef5cbce6530d	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-03bf01a10367bf92c	Custom TCP	TCP	8080	Custom	0.0.0.0/0

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.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-081af6ed4d3645aac	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-02af2d3d2e8c50747	SSH	TCP	22	Custom	0.0.0.0/0
sgr-06b79ef5cbce6330d	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-03bf01a10367bf92c	Custom TCP	TCP	8080	Custom	0.0.0.0/0
-	Custom TCP	TCP	0	Custom	0.0.0.0/0

Add rule

Cancel **Preview changes** **Save rules**

Add Port 9000

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-081af6ed4d3645aac	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-02af2d3d2e8c50747	SSH	TCP	22	Custom	0.0.0.0/0
sgr-06b79ef5cbce6330d	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-03bf01a10367bf92c	Custom TCP	TCP	8080	Custom	0.0.0.0/0
-	Custom TCP	TCP	9000	Custom	0.0.0.0/0

Add rule

Cancel **Preview changes** **Save rules**

Click on the drop down and select “Anywhere-IPv4”

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-081af6ed4d3645aac	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-02af2d3d2e8c50747	SSH	TCP	22	Custom	0.0.0.0/0
sgr-06b79ef5cbce6330d	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-03bf01a10367bf92c	Custom TCP	TCP	8080	Custom	0.0.0.0/0
-	Custom TCP	TCP	9000	Anywhere...	0.0.0.0/0

Add rule

Cancel **Preview changes** **Save rules**

Click on “Save Rules”

The screenshot shows the AWS EC2 Security Groups page. A green success message at the top states: "Inbound security group rules successfully modified on security group (sg-05531be1e137790f3 | launch-wizard-14) Details". The main section displays the security group details for "sg-05531be1e137790f3 - launch-wizard-14". Key information includes:

- Security group name:** launch-wizard-14
- Security group ID:** sg-05531be1e137790f3
- Description:** launch-wizard-14 created 2025-09-08T02:29:42.361Z
- VPC ID:** vpc-0128e9209eaef1c37
- Owner:** 324783324460
- Inbound rules count:** 5 Permission entries
- Outbound rules count:** 1 Permission entry

The "Inbound rules" tab is selected, showing a table with 5 rows of inbound rules. The columns are: Name, Security group rule ID, IP version, Type, Protocol, Port range, and Source. The rules are:

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-080d0f96dd0f0f0b9	IPv4	Custom TCP	TCP	9000	0.0.0.0/0
-	sgr-081af6ed4d3645aac	IPv4	HTTPS	TCP	443	0.0.0.0/0
-	sgr-02af2d3d2e8c50747	IPv4	SSH	TCP	22	0.0.0.0/0
-	sgr-06b79ef5cbce6330d	IPv4	HTTP	TCP	80	0.0.0.0/0
-	sgr-03bf01a10367bf92c	IPv4	Custom TCP	TCP	8080	0.0.0.0/0

At the bottom of the page, there are links for CloudShell, Feedback, and navigation icons.

PART 4: SSH Connect to the EC2 instance

Now, let us connect to our EC2 instance through SSH

The screenshot shows the AWS EC2 Instances page. On the left, the navigation menu is visible. In the center, the "Instances (1) Info" section displays a single instance named "jenkins-server" with the following details:

- Instance ID:** i-093e6e5bae1cd3540
- Instance state:** Running
- Instance type:** t2.medium
- Status check:** 2/2 checks passed
- Availability Zone:** us-east-1d

An orange arrow points to the instance ID "i-093e6e5bae1cd3540". At the top right, there are buttons for Connect, Instance state, Actions, and Launch instances. Below the instance details, a "Select an instance" dropdown is shown.

Select the instance

Instances (1/1) Info

Connect

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
jenkins-server	i-04d141c8588651c4d	Running	t2.medium	2/2 checks passed	View alarms +	us-east-1d

i-04d141c8588651c4d (jenkins-server)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary

Instance ID	i-04d141c8588651c4d	Public IPv4 address	3.89.24.67 open address
IPv6 address	-	Instance state	Running
Hostname type	IP name: ip-172-31-25-193.ec2.internal	Private IP DNS name (IPv4 only)	ip-172-31-25-193.ec2.internal
Answer private resource DNS name	IPv4 (A)	Instance type	t2.medium
Auto-assigned IP address	3.89.24.67 [Public IP]	VPC ID	vpc-0128e9209eaef1c37
IAM Role	-	Subnet ID	subnet-09e0df5f5814abf3
IMDSv2	Required	Instance ARN	arn:aws:ec2:us-east-1:524783324460:instance/i-04d141c8588651c4d

Private IPv4 addresses
172.31.25.193

Public DNS
ec2-3-89-24-67.compute-1.amazonaws.com | open address

Elastic IP addresses
-

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.

Learn more

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Click on “Connect”

EC2 > Instances > i-04d141c8588651c4d > Connect to instance

Connect info

Connect to an instance using the browser-based client.

EC2 Instance Connect Session Manager **SSH client** EC2 serial console

Instance ID
i-04d141c8588651c4d (jenkins-server)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is jenkins-key.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
chmod 400 "jenkins-key.pem"
4. Connect to your instance using its Public DNS:
ec2-3-89-24-67.compute-1.amazonaws.com

Example:
ssh -i "jenkins-key.pem" ubuntu@ec2-3-89-24-67.compute-1.amazonaws.com

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

Copy this command and open PowerShell

```
ssh -i "jenkins-key.pem" ubuntu@ec2-3-89-24-67.compute-1.amazonaws.com
```

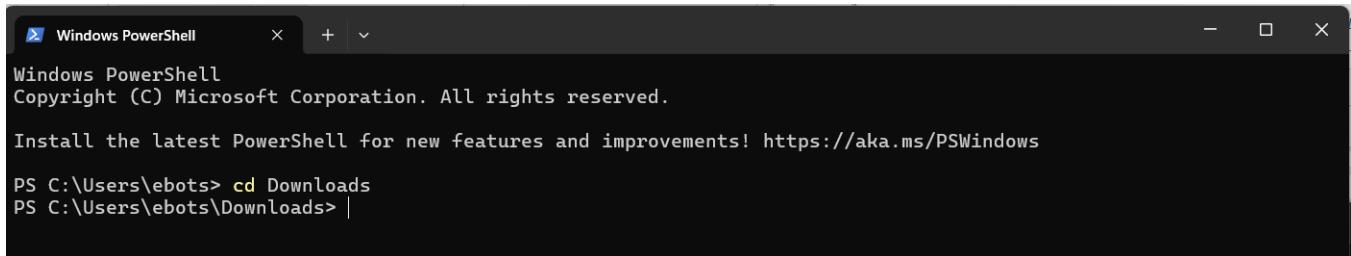
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

PS C:\Users\ebots> |
```

Navigate to the “Downloads” folder where our “.pem” file is stored by using the command:

```
cd Downloads
```



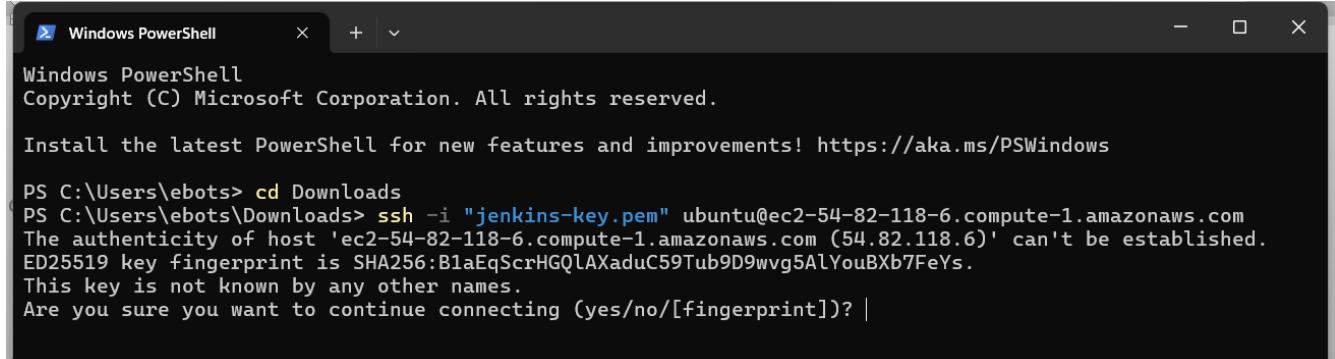
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

PS C:\Users\ebots> cd Downloads
PS C:\Users\ebots\Downloads> |
```

Then run the copied command:

```
ssh -i "jenkins-key.pem" ubuntu@ec2-54-82-118-6.compute-1.amazonaws.com
```

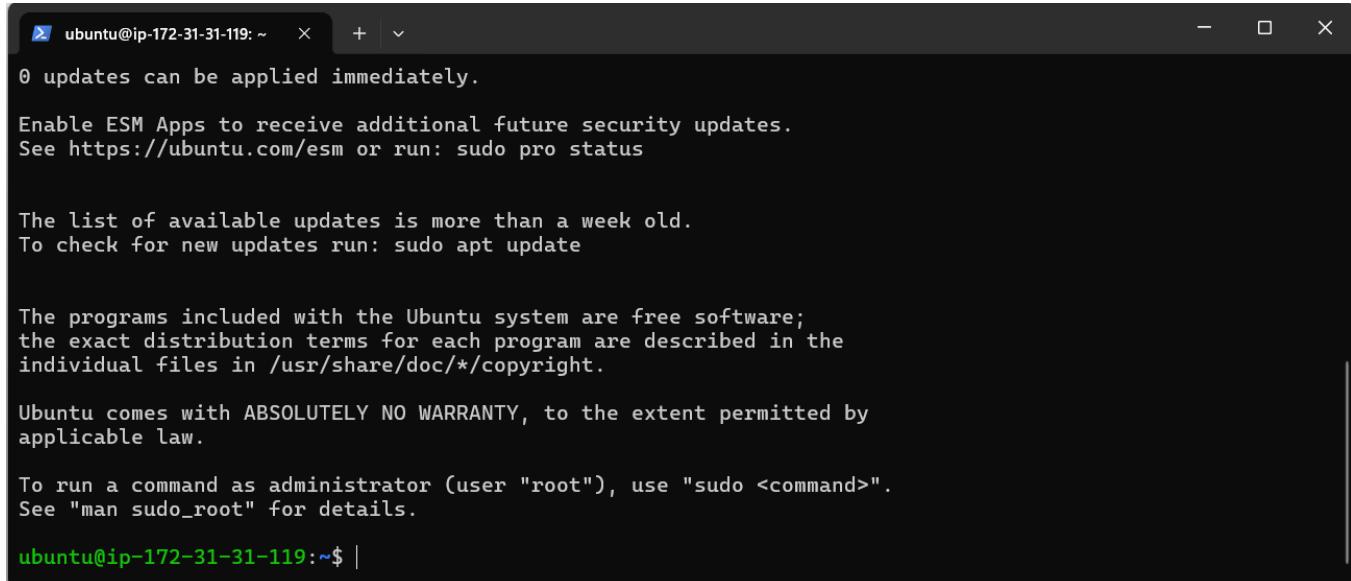


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

PS C:\Users\ebots> cd Downloads
PS C:\Users\ebots\Downloads> ssh -i "jenkins-key.pem" ubuntu@ec2-54-82-118-6.compute-1.amazonaws.com
The authenticity of host 'ec2-54-82-118-6.compute-1.amazonaws.com (54.82.118.6)' can't be established.
ED25519 key fingerprint is SHA256:B1aEqScrHGQlAXaduC59Tub9D9wvg5AlYouBXb7FeYs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? |
```

Type “**yes**” and press ENTER



```
ubuntu@ip-172-31-31-119: ~  +  ▾
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 list of available updates is more than a week old.
To check for new updates run: sudo apt update

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.

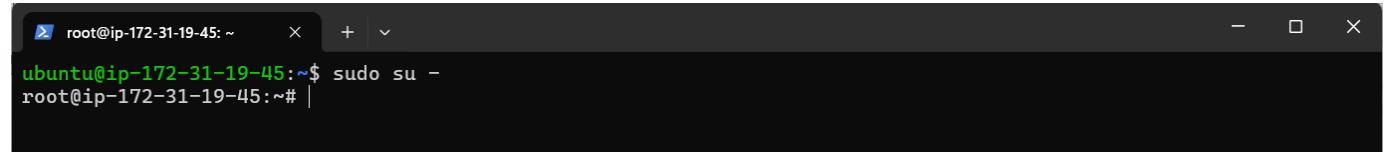
ubuntu@ip-172-31-31-119:~$ |
```

We have successfully SSH connected to our EC2 instance

STEP 2: Install Java on Ubuntu EC2 instance

The next thing is to install Java on our server, since Jenkins is written in Java. To do this we first of all switch to the root user using the command:

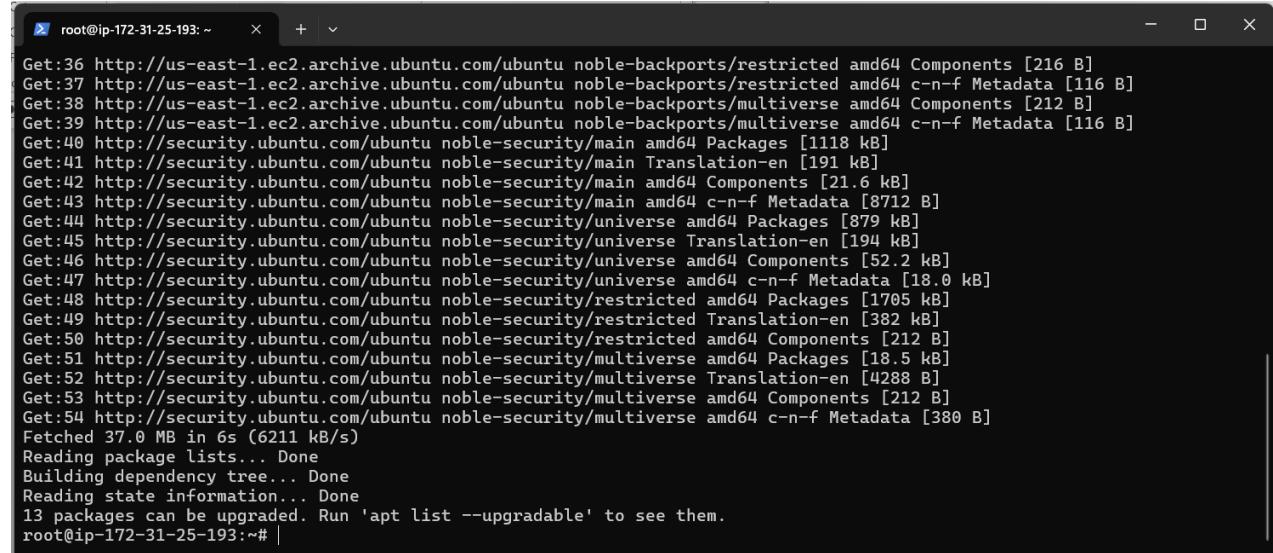
```
sudo su -
```



```
root@ip-172-31-19-45:~$ sudo su -
root@ip-172-31-19-45:~# |
```

Update the packages by running the command:

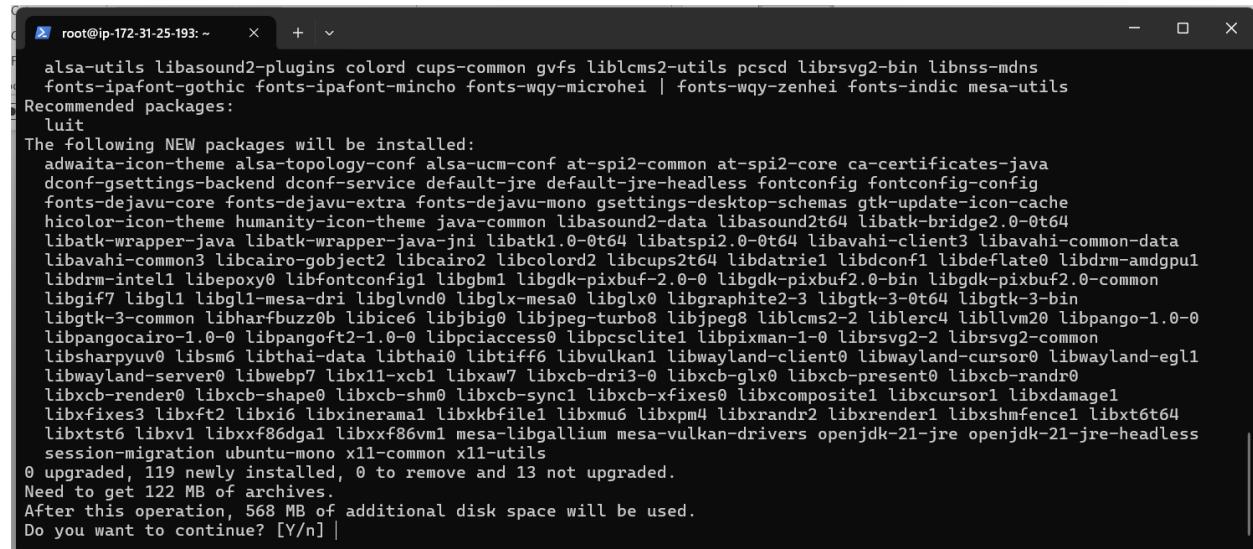
```
apt update -y
```



```
Get:36 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:37 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:38 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:39 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:40 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1118 kB]
Get:41 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [191 kB]
Get:42 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.6 kB]
Get:43 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [8712 B]
Get:44 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [879 kB]
Get:45 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [194 kB]
Get:46 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.2 kB]
Get:47 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [18.0 kB]
Get:48 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [1705 kB]
Get:49 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [382 kB]
Get:50 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [18.5 kB]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [4288 B]
Get:53 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Get:54 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [380 B]
Fetched 37.0 MB in 6s (6211 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
13 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-25-193:~# |
```

Install Java using the command:

```
sudo apt install default-jre
```



```
root@ip-172-31-25-193:~$ sudo apt install default-jre
[sudo] password for root:
Reading package lists...
Building dependency tree...
Reading state information...
The following NEW packages will be installed:
adwaita-icon-theme alsamixer libavahi-client libavahi-common-data
alsa-utils libasound2-plugins colord cups-common gvfs liblcms2-utils pcscd librsvg2-bin libnss-mdns
fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic mesa-utils
Recommended packages:
luit
The following NEW packages will be installed:
adwaita-icon-theme alsamixer libavahi-client libavahi-common-data
alsa-utils libasound2-plugins colord cups-common gvfs liblcms2-utils pcscd librsvg2-bin libnss-mdns
fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic mesa-utils
libatk-bridge2.0 libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0 libatspi2.0-0 libavahi-client3 libavahi-common-data
libavahi-common3 libcairo-gobject2 libcairo2 libcolord2 libcurl2 libdatriel libdconf1 libdeflate0 libdrm-amdgpu1
libdrm-intel1 libepoxy0 libfontconfig1 libgbm1 libgdk-pixbuf2.0-0 libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common
libgif7 libgl1 libgl1-mesa-dri libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libgtk-3-0t64 libgtk-3-bin
libgtk-3-common libharfbuzz0b libice0 libjpeg-turbo8 liblpeg8 liblcms2-2 liblrc4 libllvm20 libpango-1.0-0
libpangocairo-1.0-0 libpangoft2-1.0-0 libpiciaccess0 libpcslite1 libpixman-1-0 librsvg2-2 librsvg2-common
libsharpyuv0 libsm6 libthai-data libthai0 libtiff6 libvulkan1 libwayland-client0 libwayland-cursor0 libwayland-egl1
libwayland-server0 libwebp7 libx11-xcb1 libxaw7 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0
libxcb-render0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libcomposite1 libxcursor1 libxdamage1
libxfixes3 libxf86font1 libxi6 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt64
libxtst6 libxv1 libxxf86dg1 libxxf86vm1 mesa-lib�allium mesa-vulkan-drivers openjdk-21-jre openjdk-21-jre-headless
session-migration ubuntu-mono x11-common x11-utils
0 upgraded, 119 newly installed, 0 to remove and 13 not upgraded.
Need to get 122 MB of archives.
After this operation, 568 MB of additional disk space will be used.
Do you want to continue? [Y/n] |
```

Type "Y" and press Enter

```
root@ip-172-31-25-193: ~      + | 
Adding debian:vTrus_ECC_Root_CA.pem
Adding debian:vTrus_Root_CA.pem
done.
Setting up openjdk-21-jre:amd64 (21.0.8+9~us1~0ubuntu1~24.04.1) ...
Setting up default-jre-headless (2:1.21-75+exp1) ...
Setting up default-jre (2:1.21-75+exp1) ...
Processing triggers for libc-bin (2.39-0ubuntu8.5) ...
Processing triggers for libgdk-pixbuf-2.0-0:amd64 (2.42.10+dfsg-3ubuntu3.2) ...
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.
root@ip-172-31-25-193:~# |
```

Verify if Java has been installed successfully using the command:

```
java -version
```

```
root@ip-172-31-25-193: ~      + | 
Setting up default-jre-headless (2:1.21-75+exp1) ...
Setting up default-jre (2:1.21-75+exp1) ...
Processing triggers for libc-bin (2.39-0ubuntu8.5) ...
Processing triggers for libgdk-pixbuf-2.0-0:amd64 (2.42.10+dfsg-3ubuntu3.2) ...
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.
root@ip-172-31-25-193:~# java -version
openjdk version "21.0.8" 2025-07-15
OpenJDK Runtime Environment (build 21.0.8+9-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 21.0.8+9-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)
root@ip-172-31-25-193:~# |
```

You can see that Java has been installed successfully

STEP 3: Install Docker on EC2 server

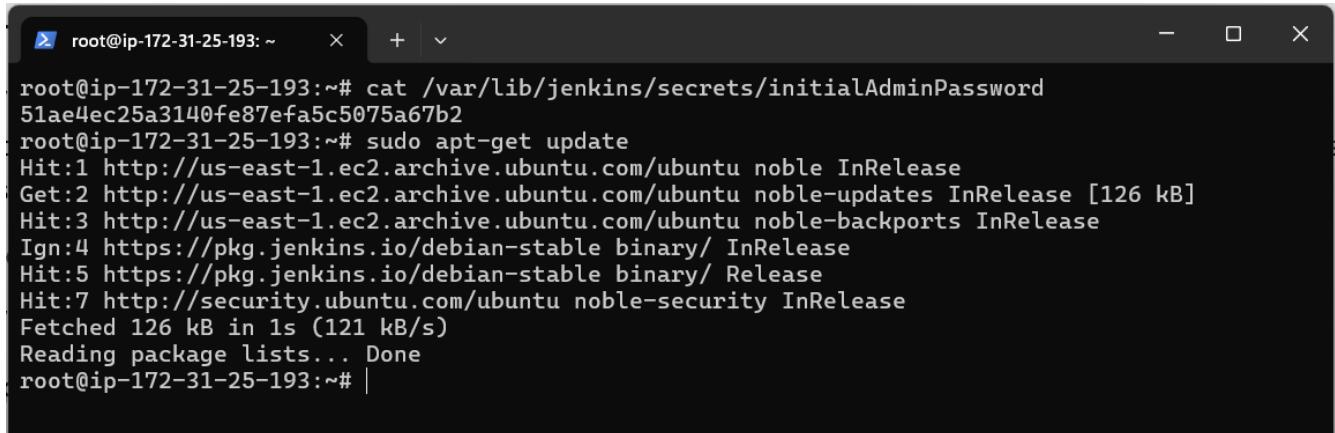
We have to install Docker on our EC2 server. Installing Docker on Ubuntu typically involves setting up Docker's official APT repository and then installing the Docker Engine. The recommended approach is to use the repository to ensure you receive the latest updates and security patches.

Steps to Install Docker Engine on Ubuntu:

We will install Docker engine on our server

Update your package list using the command:

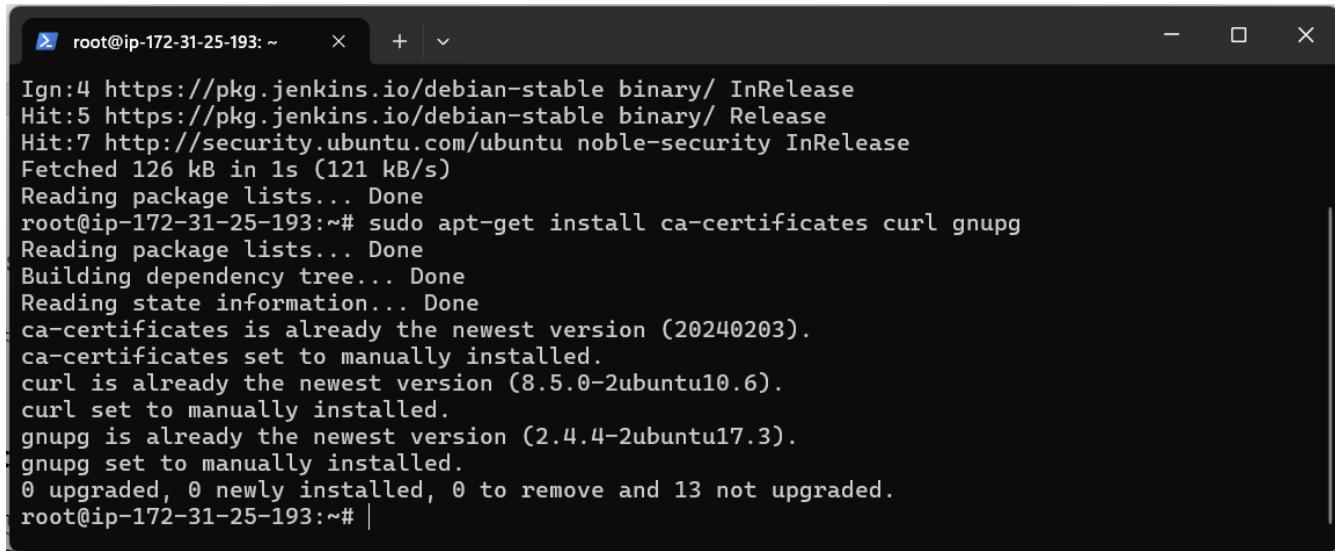
```
sudo apt-get update
```



```
root@ip-172-31-25-193:~# cat /var/lib/jenkins/secrets/initialAdminPassword
51ae4ec25a3140fe87efa5c5075a67b2
root@ip-172-31-25-193:~# 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]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:5 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 126 kB in 1s (121 kB/s)
Reading package lists... Done
root@ip-172-31-25-193:~# |
```

Install necessary packages for adding a repository over HTTPS:

```
sudo apt-get install ca-certificates curl gnupg
```



```
root@ip-172-31-25-193:~# 
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:5 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 126 kB in 1s (121 kB/s)
Reading package lists... Done
root@ip-172-31-25-193:~# sudo apt-get install ca-certificates curl gnupg
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 (8.5.0-2ubuntu10.6).
curl set to manually installed.
gnupg is already the newest version (2.4.4-2ubuntu17.3).
gnupg set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
root@ip-172-31-25-193:~# |
```

Add Docker's official GPG key

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

```
root@ip-172-31-25-193:~ x + | v
Fetched 126 kB in 1s (121 kB/s)
Reading package lists... Done
root@ip-172-31-25-193:~# sudo apt-get install ca-certificates curl gnupg
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 (8.5.0-2ubuntu10.6).
curl set to manually installed.
gnupg is already the newest version (2.4.4-2ubuntu17.3).
gnupg set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
root@ip-172-31-25-193:~# sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/ke
yings/docker.gpg
root@ip-172-31-25-193:~# |
```

Set up the Docker APT repository.

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

```
root@ip-172-31-25-193:~ x + | v
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
gnupg is already the newest version (2.4.4-2ubuntu17.3).
gnupg set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
root@ip-172-31-25-193:~# sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/ke
yings/docker.gpg
root@ip-172-31-25-193:~# echo \
"deb [arch=$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] http
s://download.docker.com/linux/ubuntu \
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
root@ip-172-31-25-193:~# |
```

Update your package list again to include the new Docker repository:

```
sudo apt-get update
```

```
root@ip-172-31-25-193:~# 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
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Ign:6 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:7 https://pkg.jenkins.io/debian-stable binary/ Release
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [30.5 kB]
Fetched 79.3 kB in 1s (151 kB/s)
Reading package lists... Done
root@ip-172-31-25-193:~# |
```

Install Docker Engine, containerd, and Docker Compose:

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

```
root@ip-172-31-25-193:~# Reading package lists... Done
root@ip-172-31-25-193:~# 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 libslirp0 pigz slirp4netns
Suggested packages:
  cgroupfs-mount | cgroup-lite docker-model-plugin
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin libslirp0 pigz slirp4netns
0 upgraded, 9 newly installed, 0 to remove and 13 not upgraded.
Need to get 103 MB of archives.
After this operation, 431 MB of additional disk space will be used.
Do you want to continue? [Y/n] |
```

Type "Y" and press ENTER

```
root@ip-172-31-25-193:~# 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 (2.39-0ubuntu8.5) ...
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.

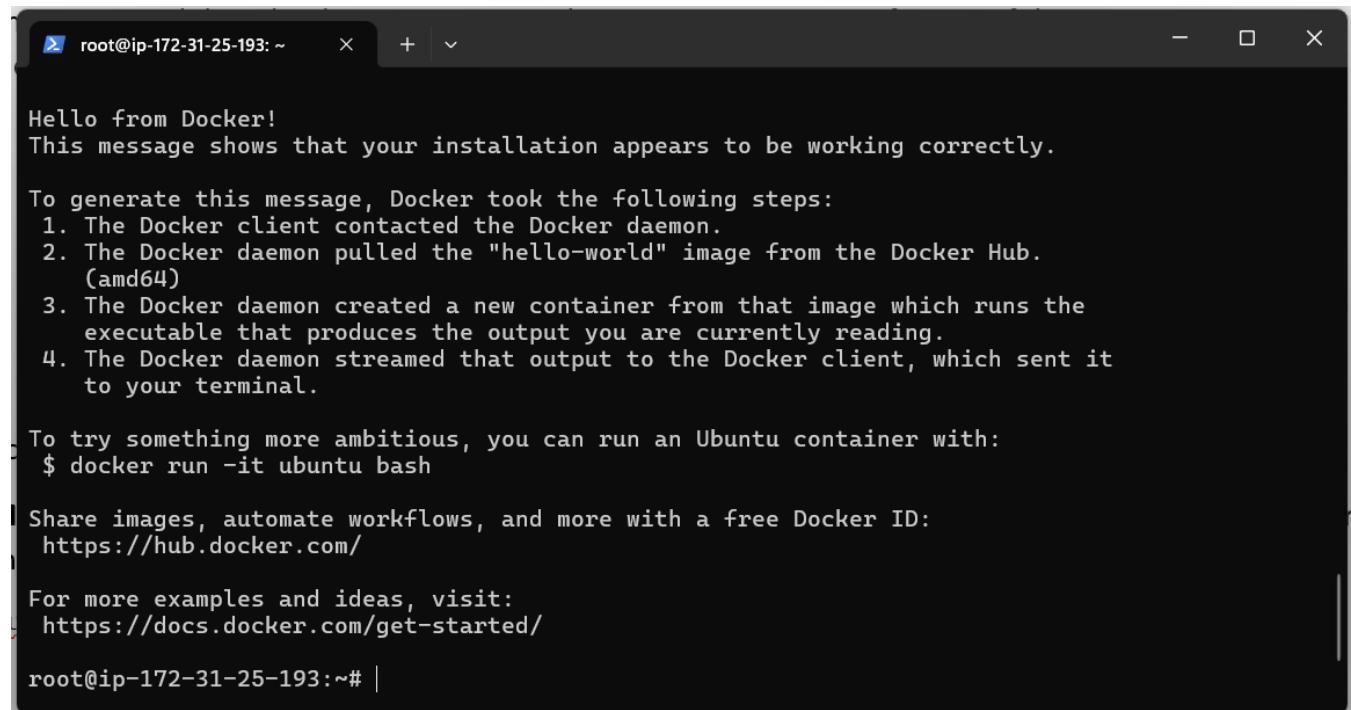
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-25-193:~# |
```

Verify the installation by running the hello-world image:

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

This command downloads a test image and runs it in a container. If successful, it prints a message indicating Docker is working.



```
root@ip-172-31-25-193:~# 
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

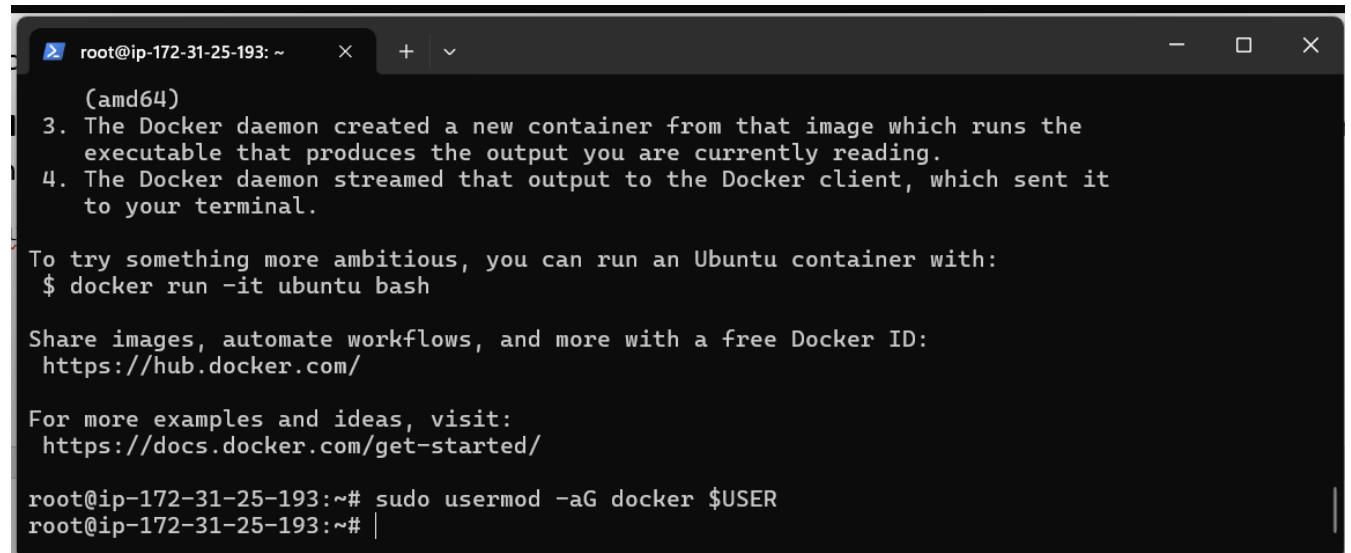
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
root@ip-172-31-25-193:~# |
```

You can see that Docker has been installed successfully.

Manage Docker as a non-root user: To avoid using sudo with every Docker command, add your user to the docker group:

```
sudo usermod -aG docker $USER
```



```
root@ip-172-31-25-193:~# 
(root)
 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

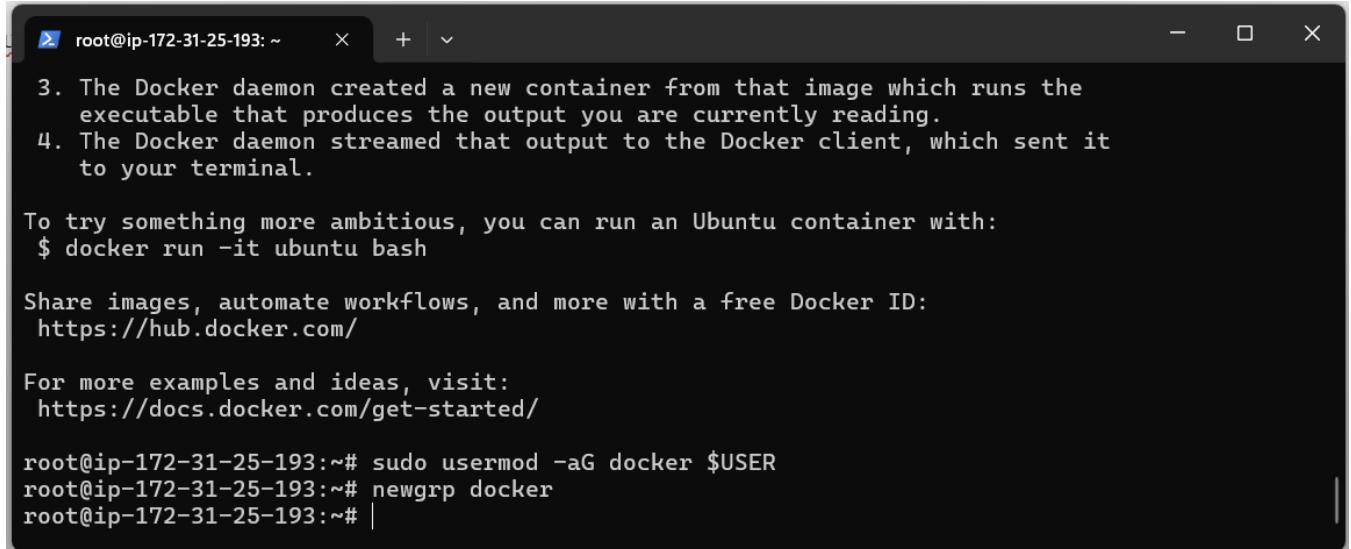
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
root@ip-172-31-25-193:~# sudo usermod -aG docker $USER
root@ip-172-31-25-193:~# |
```

You will need to log out and log back in (or restart your system) for this change to take effect. After logging back in, you can run docker run hello-world without sudo.

Activate a user's membership in the docker group using the command:

```
newgrp docker
```



A terminal window titled "root@ip-172-31-25-193: ~". The window contains the following text:

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

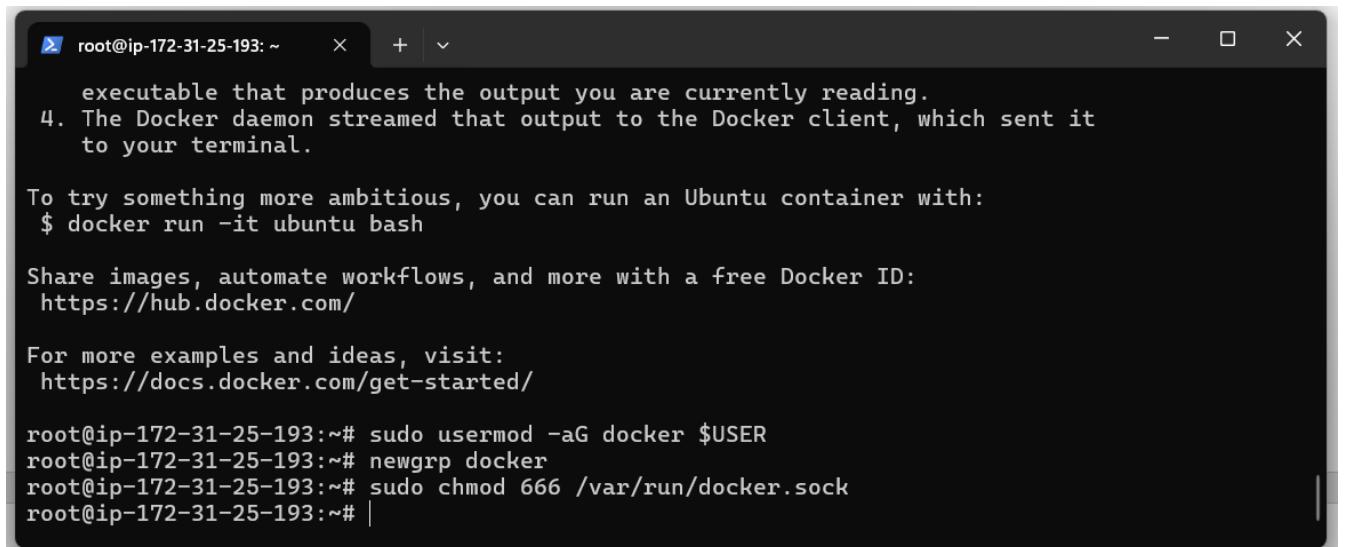
Share images, automate workflows, and more with a free Docker ID:
<https://hub.docker.com/>

For more examples and ideas, visit:
<https://docs.docker.com/get-started/>

```
root@ip-172-31-25-193:~# sudo usermod -aG docker $USER
root@ip-172-31-25-193:~# newgrp docker
root@ip-172-31-25-193:~# |
```

Run the command `sudo chmod 666 /var/run/docker.sock` modifies the permissions of the Docker Unix socket file located at `/var/run/docker.sock`

```
sudo chmod 666 /var/run/docker.sock
```



A terminal window titled "root@ip-172-31-25-193: ~". The window contains the following text:

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/>

```
root@ip-172-31-25-193:~# sudo usermod -aG docker $USER
root@ip-172-31-25-193:~# newgrp docker
root@ip-172-31-25-193:~# sudo chmod 666 /var/run/docker.sock
root@ip-172-31-25-193:~# |
```

Then restart Docker using the command:

```
sudo systemctl restart docker
```

```
root@ip-172-31-25-193: ~ + ^ - □ ×
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/

root@ip-172-31-25-193:~# sudo usermod -aG docker $USER
root@ip-172-31-25-193:~# newgrp docker
root@ip-172-31-25-193:~# sudo chmod 666 /var/run/docker.sock
root@ip-172-31-25-193:~# sudo systemctl restart docker
root@ip-172-31-25-193:~# |
```

Then run the command to verify if docker is working again, but this time we will remove the “sudo”
docker run hello-world

```
root@ip-172-31-22-196: /home ~ + ^ - □ ×
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/

root@ip-172-31-22-196:/home/ubuntu# |
```

You can see that Docker is still working

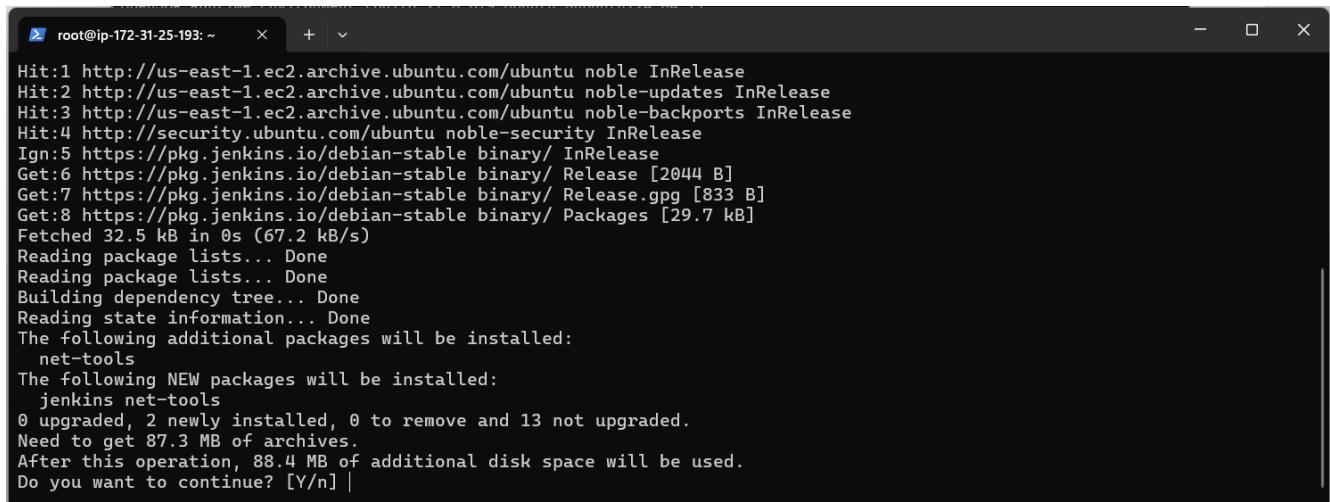
STEP 4: Install Jenkins and Access Jenkins on your browser

In this step, we will install Jenkins and access it on our browser

Install Jenkins on EC2 instance

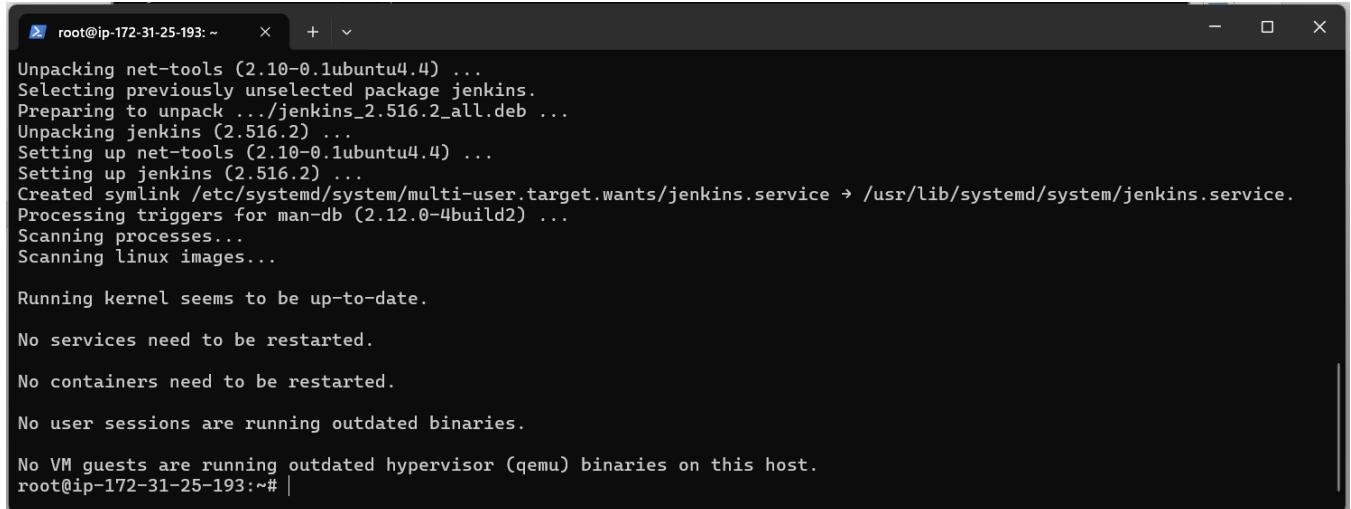
Now, let us install Jenkins. This will be done by running the command:

```
sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/" \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```



```
root@ip-172-31-25-193: ~ + v
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
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:7 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [29.7 kB]
Fetched 32.5 kB in 0s (67.2 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 13 not upgraded.
Need to get 87.3 MB of archives.
After this operation, 88.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] |
```

Type “Y” and press ENTER



```
root@ip-172-31-25-193: ~ + v
Unpacking net-tools (2.10-0.1ubuntu4.4) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.516.2_all.deb ...
Unpacking jenkins (2.516.2) ...
Setting up net-tools (2.10-0.1ubuntu4.4) ...
Setting up jenkins (2.516.2) ...
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.
root@ip-172-31-25-193:~# |
```

Enable Jenkins using the command:

```
sudo systemctl enable jenkins
```

```

root@ip-172-31-25-193:~      + | v
Unpacking jenkins (2.516.2) ...
Setting up net-tools (2.10-0.1ubuntu4.4) ...
Setting up jenkins (2.516.2) ...
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.
root@ip-172-31-25-193:~# sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
root@ip-172-31-25-193:~# |

```

Start Jenkins by using the command:

```
sudo systemctl start jenkins
```

```

root@ip-172-31-25-193:~      + | v
Setting up net-tools (2.10-0.1ubuntu4.4) ...
Setting up jenkins (2.516.2) ...
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.
root@ip-172-31-25-193:~# sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
root@ip-172-31-25-193:~# sudo systemctl start jenkins
root@ip-172-31-25-193:~# |

```

Verify if Jenkins is running by using the command:

```
sudo systemctl status jenkins
```

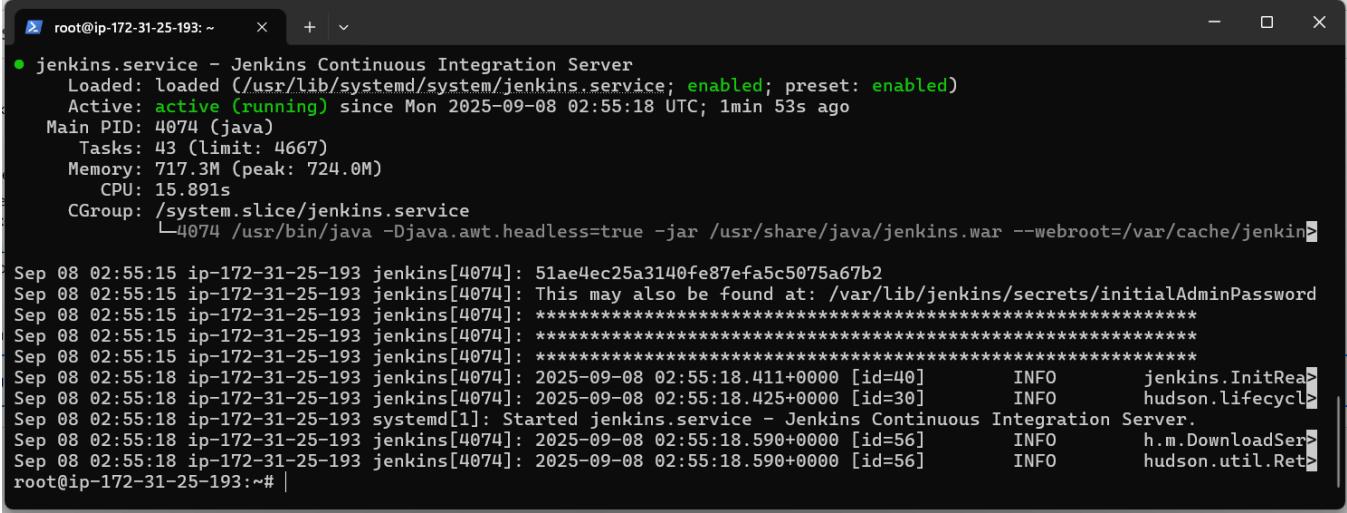
```

root@ip-172-31-25-193:~      + | v
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Mon 2025-09-08 02:55:18 UTC; 1min 53s ago
     Main PID: 4074 (java)
        Tasks: 43 (limit: 4667)
       Memory: 717.3M (peak: 724.0M)
          CPU: 15.891s
         CGroup: /system.slice/jenkins.service
             └─4074 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins

Sep 08 02:55:15 ip-172-31-25-193 jenkins[4074]: 51ae4ec25a3140fe87efa5c5075a67b2
Sep 08 02:55:15 ip-172-31-25-193 jenkins[4074]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 08 02:55:15 ip-172-31-25-193 jenkins[4074]: ****
Sep 08 02:55:15 ip-172-31-25-193 jenkins[4074]: ****
Sep 08 02:55:15 ip-172-31-25-193 jenkins[4074]: ****
Sep 08 02:55:18 ip-172-31-25-193 jenkins[4074]: 2025-09-08 02:55:18.411+0000 [id=40]           INFO      jenkins.InitRea
Sep 08 02:55:18 ip-172-31-25-193 jenkins[4074]: 2025-09-08 02:55:18.425+0000 [id=30]           INFO      hudson.lifecycle
Sep 08 02:55:18 ip-172-31-25-193 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Sep 08 02:55:18 ip-172-31-25-193 jenkins[4074]: 2025-09-08 02:55:18.590+0000 [id=56]           INFO      h.m.DownloadSer
Sep 08 02:55:18 ip-172-31-25-193 jenkins[4074]: 2025-09-08 02:55:18.590+0000 [id=56]           INFO      hudson.util.Ret
Lines 1-20

```

You can see that Jenkins is Active and running. To leave this mode on the terminal, type “q”



```

root@ip-172-31-25-193: ~
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Mon 2025-09-08 02:55:18 UTC; 1min 53s ago
     Main PID: 4074 (java)
        Tasks: 43 (limit: 4667)
       Memory: 717.3M (peak: 724.0M)
          CPU: 15.891s
        CGroup: /system.slice/jenkins.service
                └─4074 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins

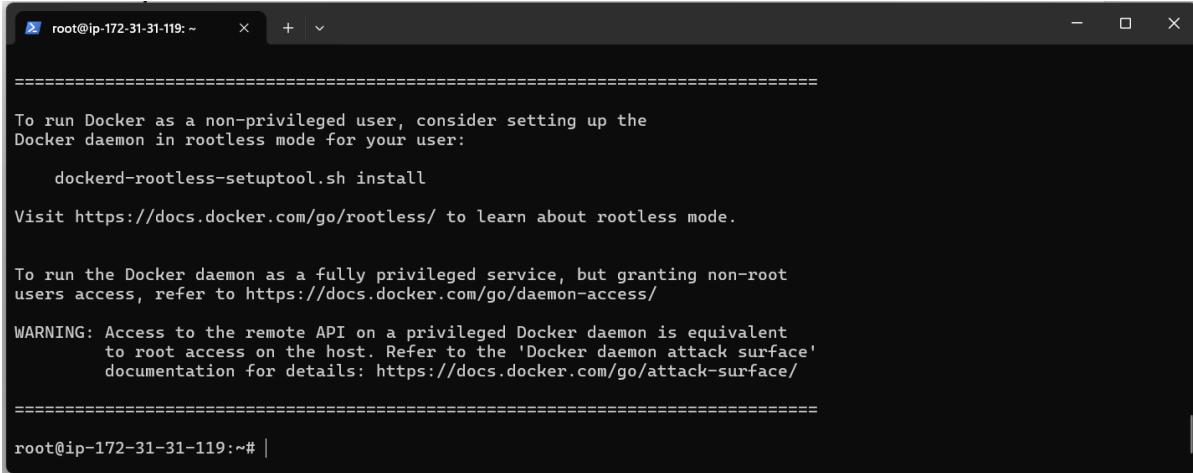
Sep 08 02:55:15 ip-172-31-25-193 jenkins[4074]: 51ae4ec25a3140fe87efa5c5075a67b2
Sep 08 02:55:15 ip-172-31-25-193 jenkins[4074]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 08 02:55:15 ip-172-31-25-193 jenkins[4074]: ****
Sep 08 02:55:15 ip-172-31-25-193 jenkins[4074]: ****
Sep 08 02:55:15 ip-172-31-25-193 jenkins[4074]: ****
Sep 08 02:55:18 ip-172-31-25-193 jenkins[4074]: 2025-09-08 02:55:18.411+0000 [id=40]           INFO      jenkins.InitRea>
Sep 08 02:55:18 ip-172-31-25-193 jenkins[4074]: 2025-09-08 02:55:18.425+0000 [id=30]           INFO      hudson.lifecycle>
Sep 08 02:55:18 ip-172-31-25-193 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Sep 08 02:55:18 ip-172-31-25-193 jenkins[4074]: 2025-09-08 02:55:18.590+0000 [id=56]           INFO      h.m.DownloadSer>
Sep 08 02:55:18 ip-172-31-25-193 jenkins[4074]: 2025-09-08 02:55:18.590+0000 [id=56]           INFO      hudson.util.Ret>
root@ip-172-31-25-193:~# |

```

STEP 5: Setup Docker in Jenkins

We will run this command to add Jenkins user to Docker group on the EC2 server's terminal:

```
curl -fsSL get.docker.com | /bin/bash
```



```

root@ip-172-31-31-119: ~
=====
To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.

To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

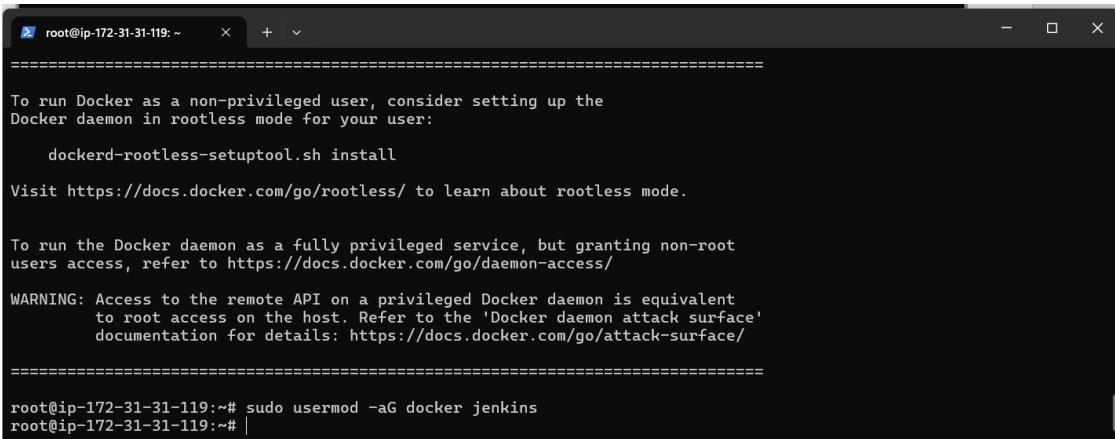
WARNING: Access to the remote API on a privileged Docker daemon is equivalent
to root access on the host. Refer to the 'Docker daemon attack surface'
documentation for details: https://docs.docker.com/go/attack-surface/
=====

root@ip-172-31-31-119:~# |

```

Add Jenkins user to Docker group

```
sudo usermod -aG docker Jenkins
```



```

root@ip-172-31-31-119: ~
=====
To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.

To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
to root access on the host. Refer to the 'Docker daemon attack surface'
documentation for details: https://docs.docker.com/go/attack-surface/
=====

root@ip-172-31-31-119:~# sudo usermod -aG docker jenkins
root@ip-172-31-31-119:~# |

```

Add Ubuntu user to Docker group

```
sudo usermod -aG docker ubuntu
```

```
root@ip-172-31-31-119: ~ + X - □ ×

To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.

To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
to root access on the host. Refer to the 'Docker daemon attack surface'
documentation for details: https://docs.docker.com/go/attack-surface/
=====

root@ip-172-31-31-119:~# sudo usermod -aG docker jenkins
root@ip-172-31-31-119:~# sudo usermod -aG docker ubuntu
root@ip-172-31-31-119:~# |
```

Restart Jenkins using the command:

```
sudo systemctl restart jenkins
```

```
root@ip-172-31-31-119: ~ + X - □ ×

To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.

To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
to root access on the host. Refer to the 'Docker daemon attack surface'
documentation for details: https://docs.docker.com/go/attack-surface/
=====

root@ip-172-31-31-119:~# sudo usermod -aG docker jenkins
root@ip-172-31-31-119:~# sudo usermod -aG docker ubuntu
root@ip-172-31-31-119:~# sudo systemctl restart jenkins
root@ip-172-31-31-119:~# |
```

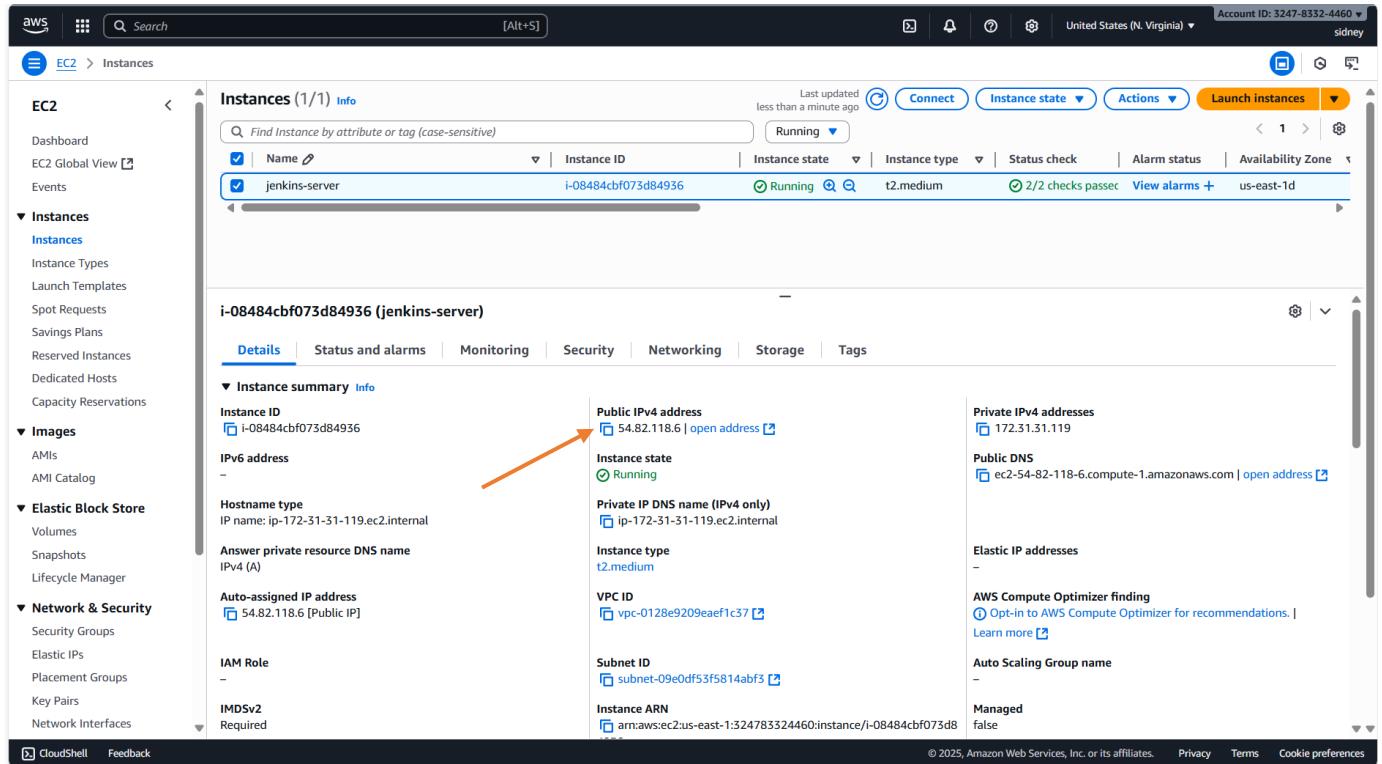
Docker has been setup in Jenkins

STEP 6: Access Jenkins server in browser using EC2 server's public IP

We have to access Jenkins through **Port 8080** on our browser. To do this we have to first grant access to Port 8080 in our EC2 instance Inbound rules.

Now, let us access Jenkins on our browser by using
http://Public IPv4 Address:8080/

Go to our EC2 instance

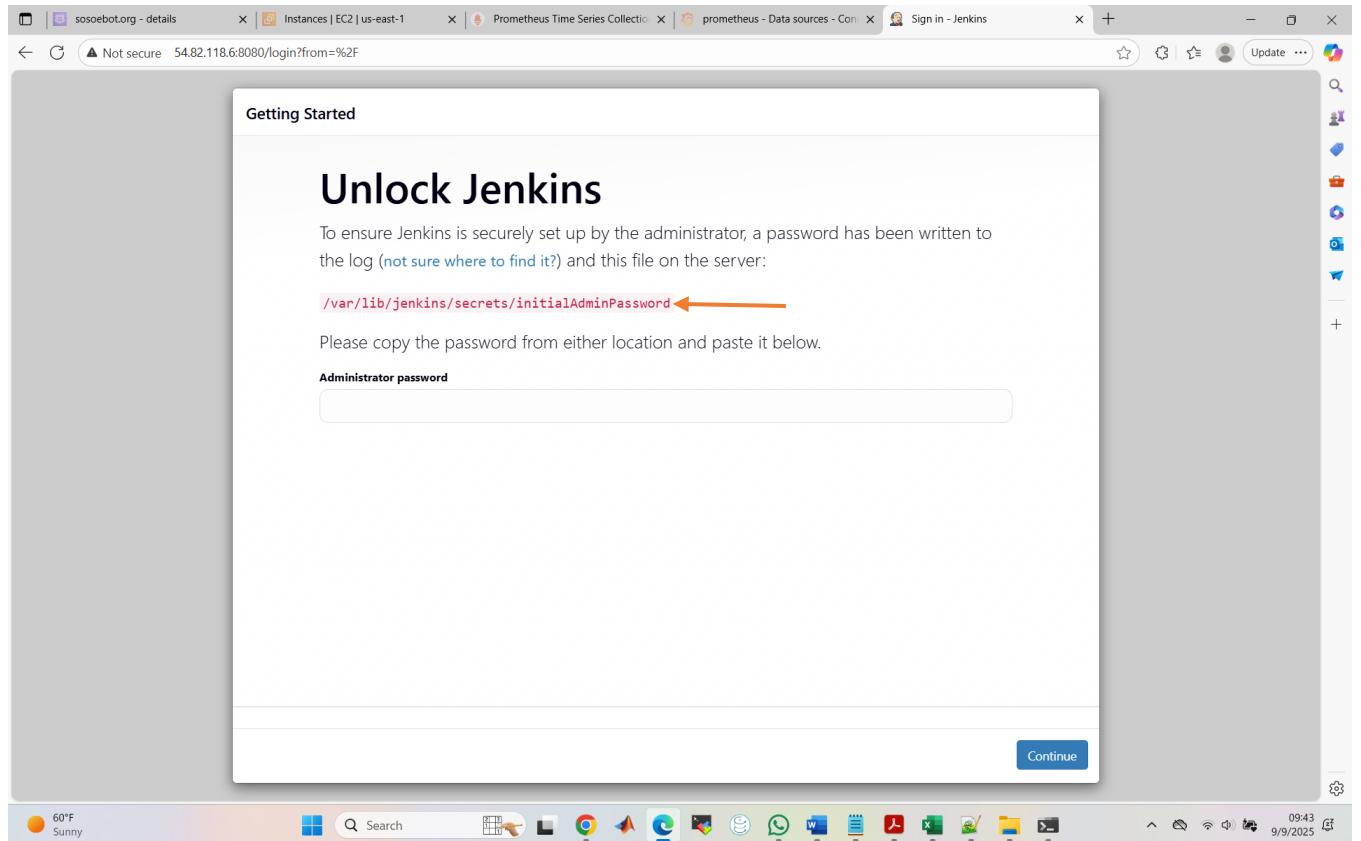


The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with sections like EC2, Dashboard, EC2 Global View, Events, Instances (with sub-options like Instances, Instance Types, Launch Templates, etc.), Images, Elastic Block Store, Network & Security, and more. The main area shows a table of instances with one row selected: "Instances (1/1) Info". The selected instance is "jenkins-server" (Instance ID: i-08484cbf073d84936). The instance status is "Running". Below the table, the instance details are shown under the heading "i-08484cbf073d84936 (jenkins-server)". The "Details" tab is selected. Under "Instance summary", the "Public IPv4 address" is listed as "54.82.118.6" with a link to "open address". Other details include Instance ID (i-08484cbf073d84936), Instance state (Running), Private IP DNS name (ip-172-31-31-119.ec2.internal), Instance type (t2.medium), VPC ID (vpc-0128e9209eaeef1c57), Subnet ID (subnet-09e0df53f5814abf3), and Instance ARN (arn:aws:ec2:us-east-1:324783324460:instance/i-08484cbf073d84936). To the right, there are sections for Private IPv4 addresses (172.31.31.119), Public DNS (ec2-54-82-118-6.compute-1.amazonaws.com), and Elastic IP addresses. At the bottom, there are links for CloudShell, Feedback, and various AWS terms like Opt-in to AWS Compute Optimizer.

The public IP address of our EC2 instance is 3.89.24.67

Paste **http://54.82.118.6:8080**

On our browser



Getting the Jenkins Password

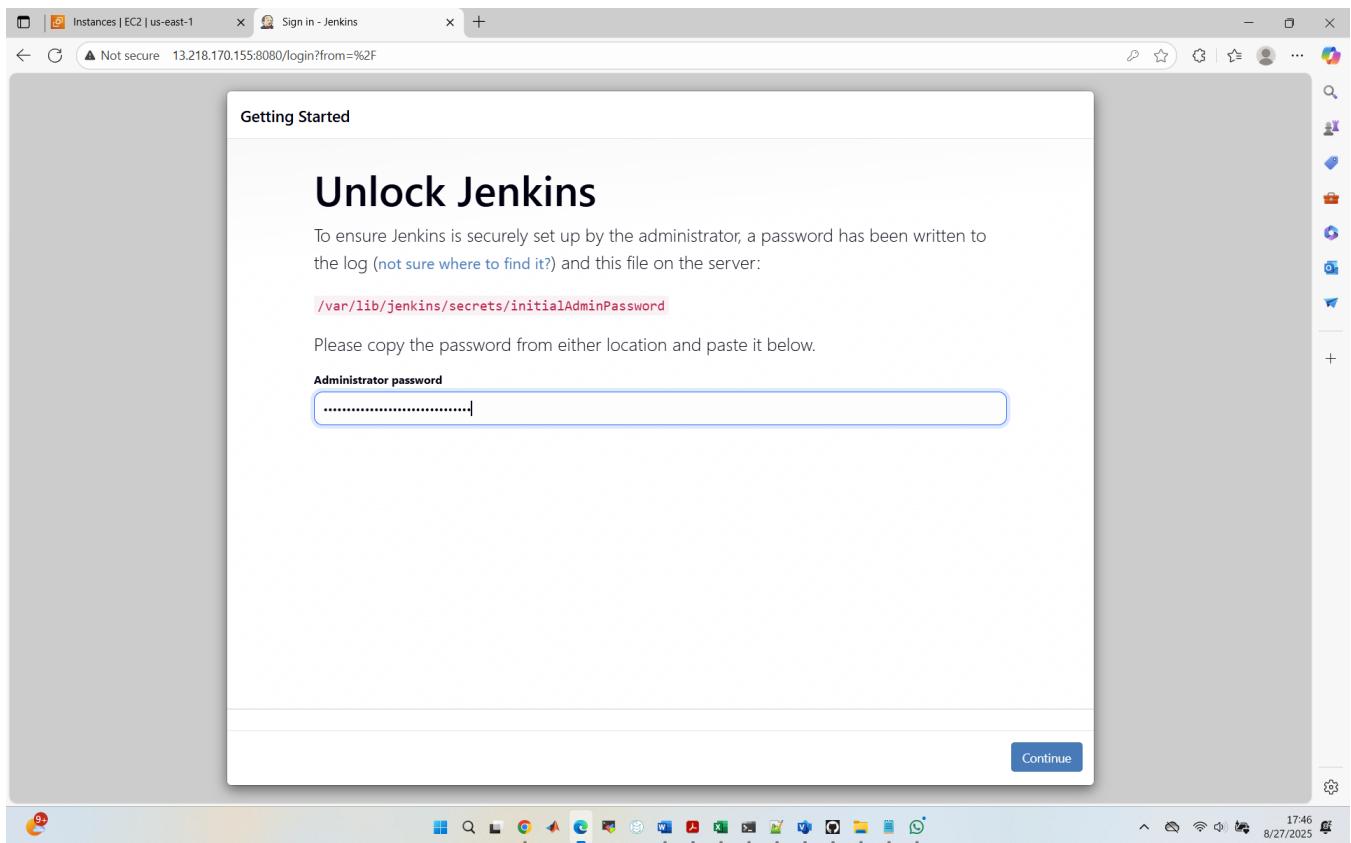
Now, let us get our password by copying the line above and run this command on our EC2 command prompt

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

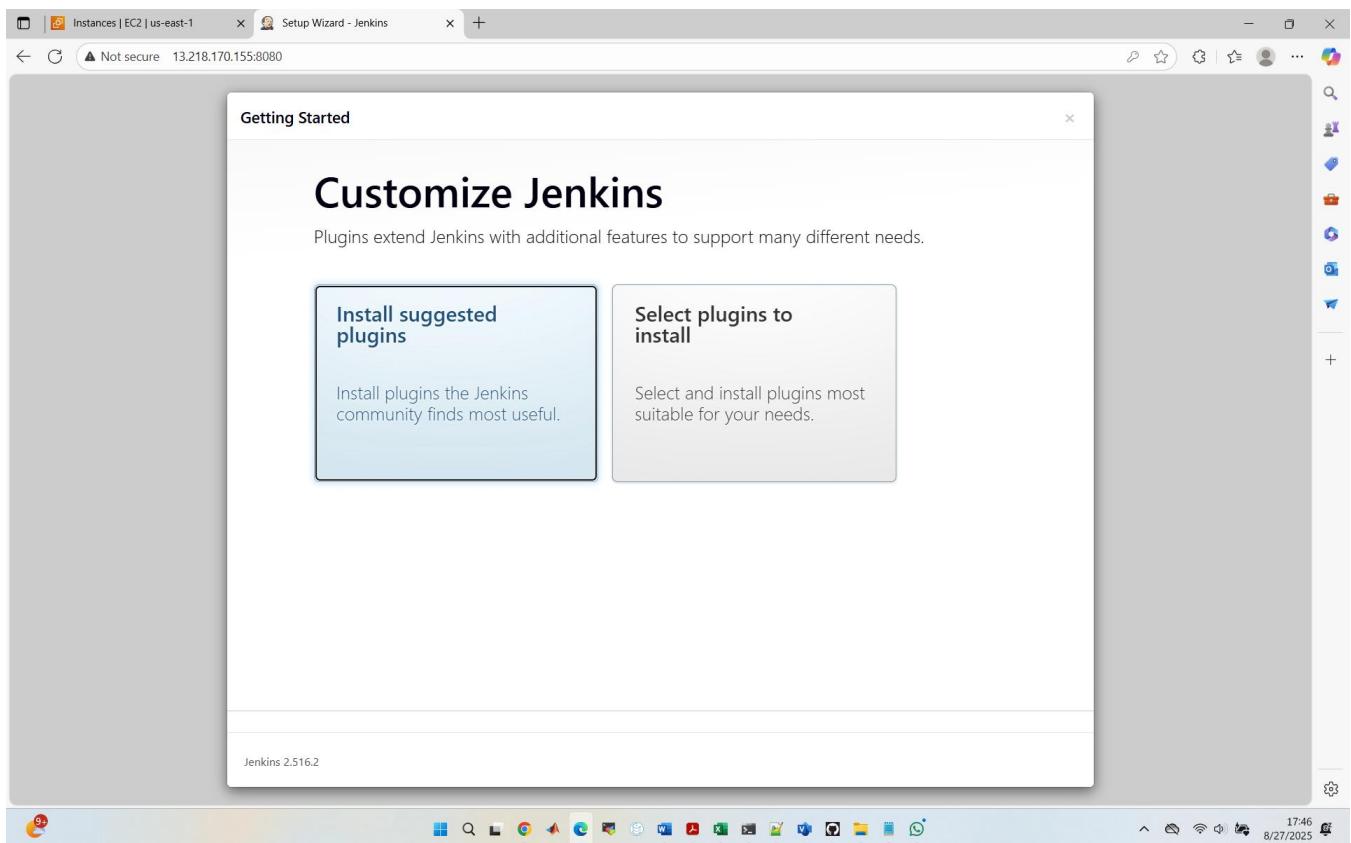
A screenshot of a terminal window. The prompt is "root@ip-172-31-31-119: ~". The command "cat /var/lib/jenkins/secrets/initialAdminPassword" is entered, followed by its output: "930e9e2e4a514caeaff048f6b9b38703". The terminal window has a dark background and light text. The title bar shows the session is for root at ip-172-31-31-119.

The password is 930e9e2e4a514caeaff048f6b9b38703

Paste it on the Jenkins browser



Click on “Continue”



Click on “Install Suggested Plugins”

Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.487

[Skip and continue as admin](#)[Save and Continue](#)

Enter the username, Password, Full name and email address.

Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.487

[Skip and continue as admin](#)[Save and Continue](#)

Click on “Save and Continue”

Getting Started

Instance Configuration

Jenkins URL:

http://54.82.118.6:8080/

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.

Jenkins 2.516.2

Not now

Save and Finish



Click on “Save and Finish”

Getting Started

Jenkins is ready!

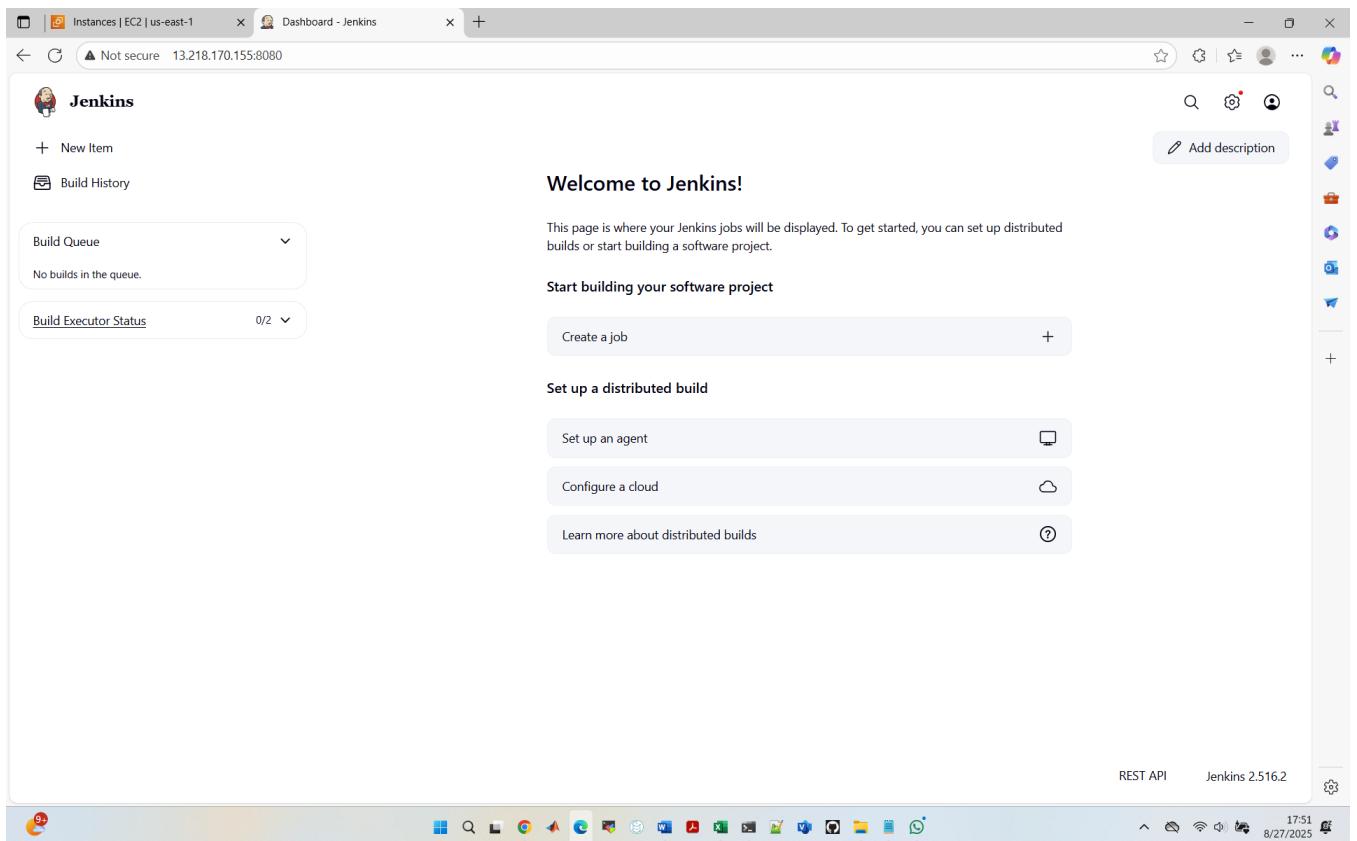
Your Jenkins setup is complete.

Start using Jenkins



Jenkins 2.487

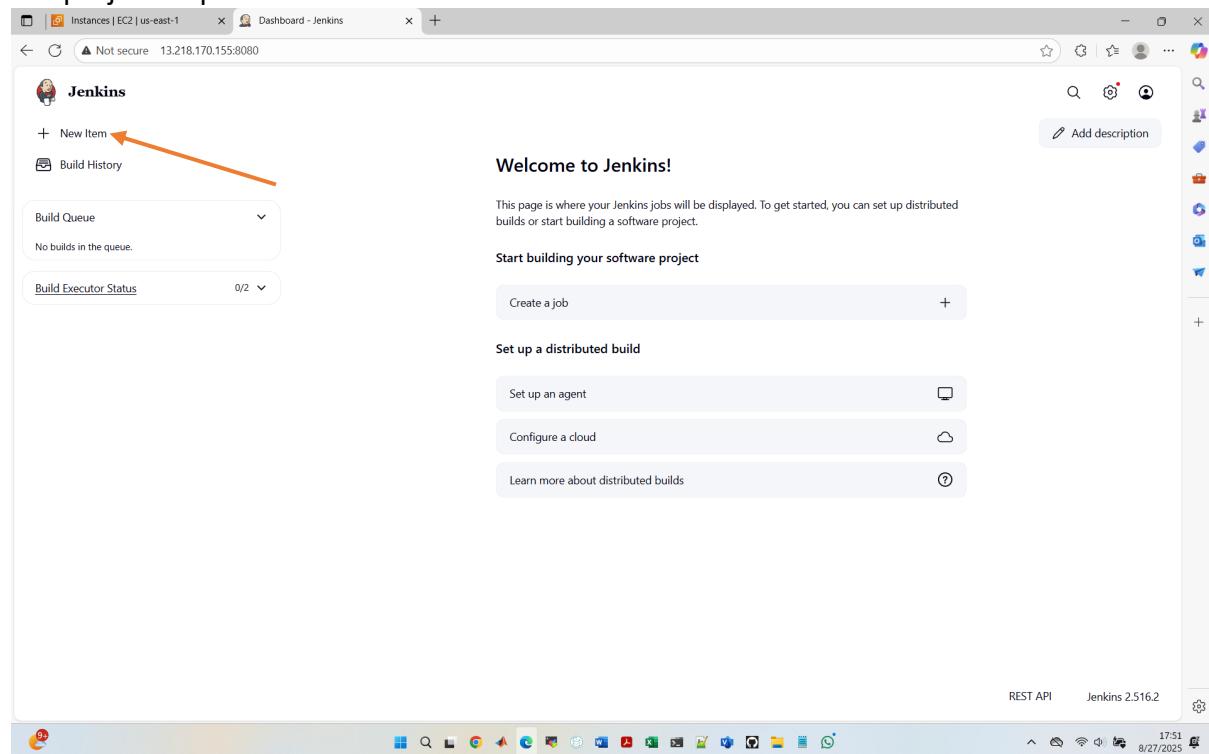
Click on “Start using Jenkins”



We are now on the Jenkins browser.

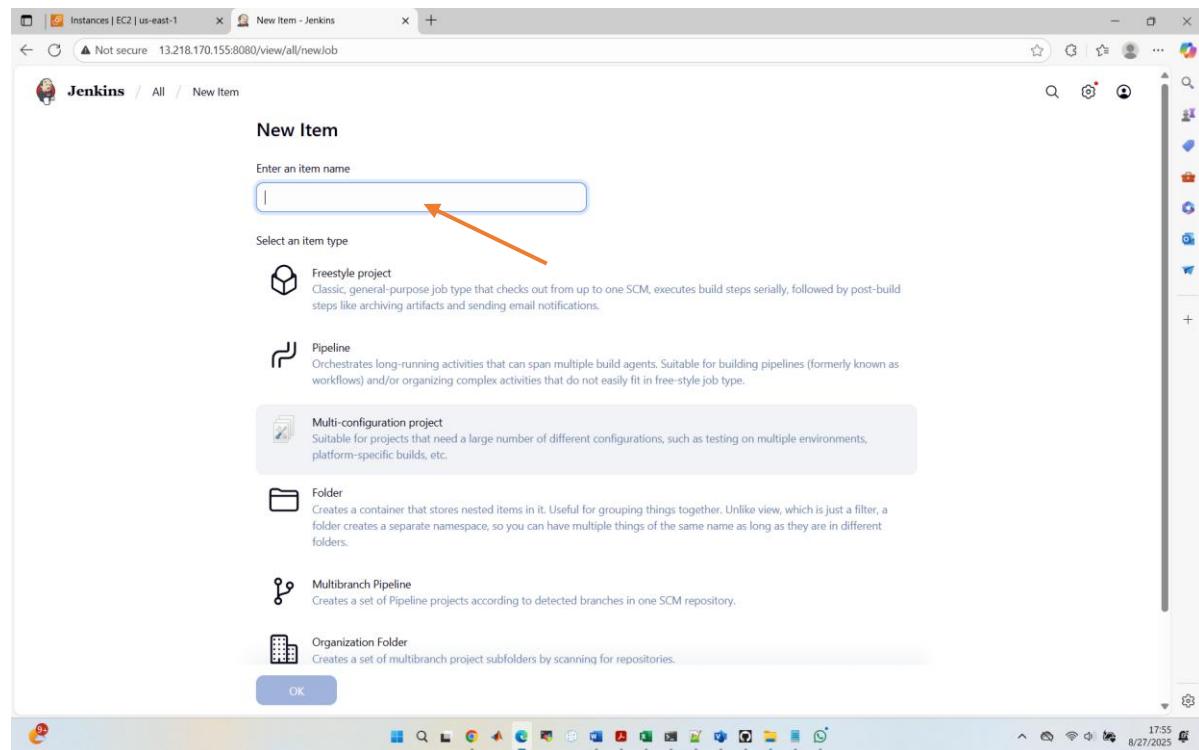
STEP 7: Create Pipeline on Jenkins

We will create a simple “Hello World” pipeline and test. Then we will later modify the pipeline to meet our project requirements.



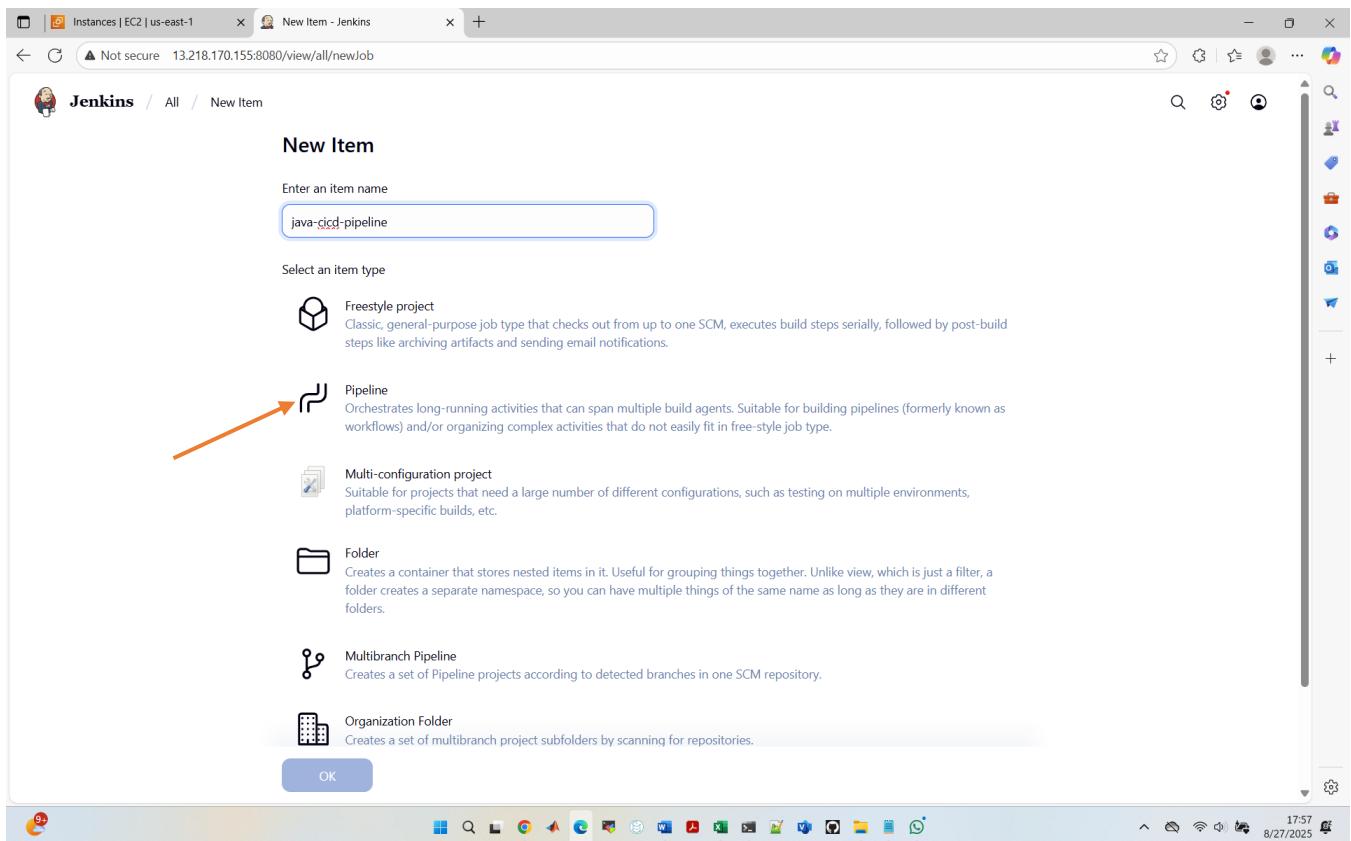
The screenshot shows the Jenkins dashboard. At the top left, there is a link to 'Instances | EC2 | us-east-1'. The main title is 'Dashboard - Jenkins'. Below the title, there is a 'Welcome to Jenkins!' message and a 'Start building your software project' section. On the left, there are links for 'Build History', 'Build Queue' (which says 'No builds in the queue.'), and 'Build Executor Status' (which says '0/2'). In the center, there is a 'Create a job' button with a '+' sign, followed by three sections: 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. At the bottom right, there are links for 'REST API' and 'Jenkins 2.516.2'. The status bar at the bottom shows the date and time: '8/27/2025 17:51'.

Now, let us create our pipeline. Click on “New Item”

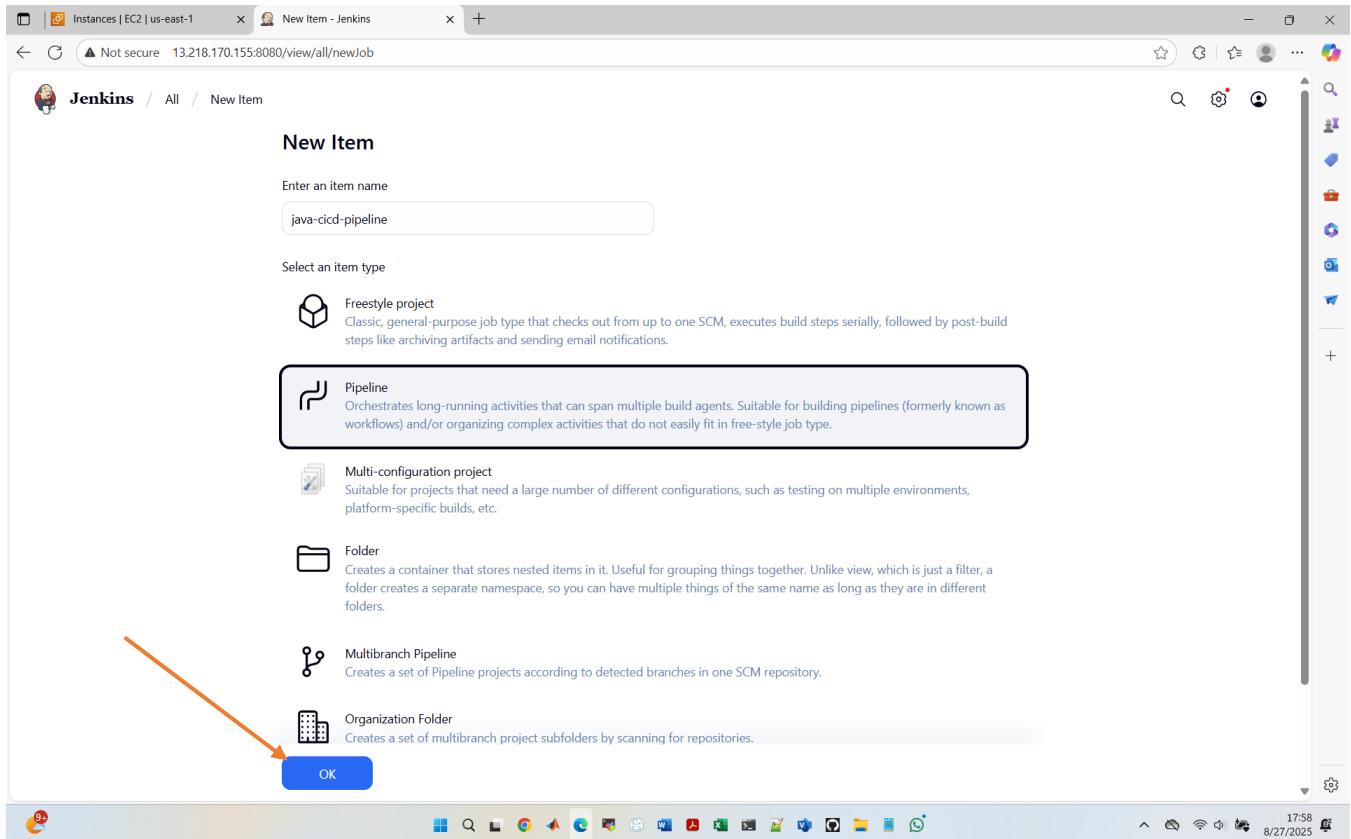


The screenshot shows the 'New Item' creation dialog. The title is 'New Item'. There is a text input field labeled 'Enter an item name' with a placeholder 'Item Name'. Below it, there is a section titled 'Select an item type' with several options: 'Freestyle project', 'Pipeline', 'Multi-configuration project', 'Folder', 'Multibranch Pipeline', and 'Organization Folder'. Each option has a small icon and a brief description. At the bottom of the dialog is an 'OK' button. The status bar at the bottom shows the date and time: '8/27/2025 17:55'.

We have to give our pipeline a name, we will call it “**java-cicd-pipeline**”



Select “Pipeline”



Click on “OK”

The screenshot shows the Jenkins 'General' configuration page for a job named 'java-cicd-pipeline'. The 'Enabled' switch is turned on. The 'Description' field contains the text 'Java Based CICD Pipeline'. Below the description, there are several configuration options: '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 page are 'Save' and 'Apply' buttons.

We can now start to configure our pipeline. For “**Description**”, we will add “**Java Based CICD Pipeline**”

The screenshot shows the Jenkins 'General' configuration page for the same job. The 'Enabled' switch is turned on. The 'Description' field now contains the text 'Java Based CICD Pipeline'. The rest of the configuration options and buttons are identical to the previous screenshot.

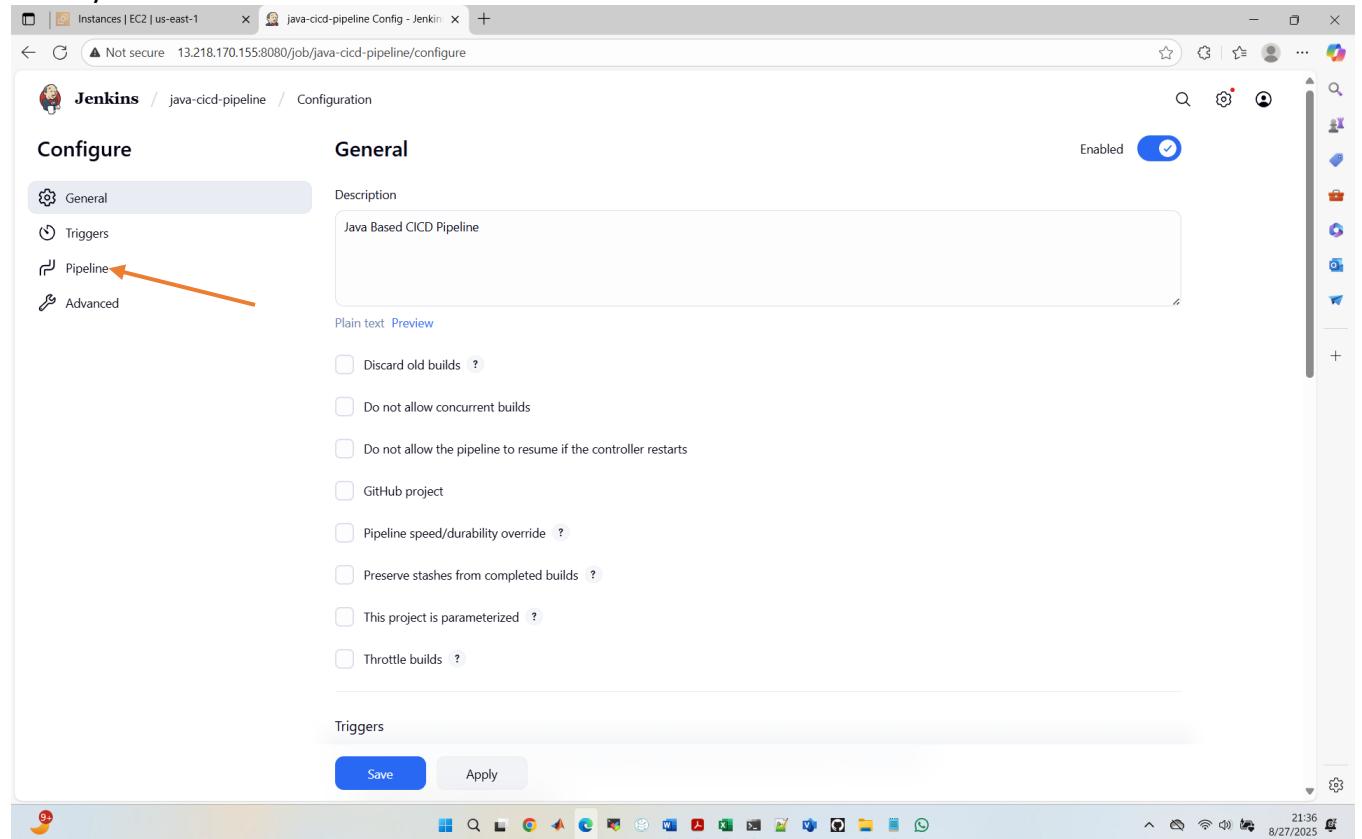
Now we can write the Pipeline. There are two ways to write the pipeline script: **Declarative pipeline**- simplified way Scripted Pipeline and the **Groovy Script**.

Declarative Pipeline: Structured and simplified way to define pipelines in Jenkins. Uses predefined syntax with specific keywords. Great for standardizing pipeline definitions and simpler requirements. Offers less flexibility compared to Scripted Pipeline.

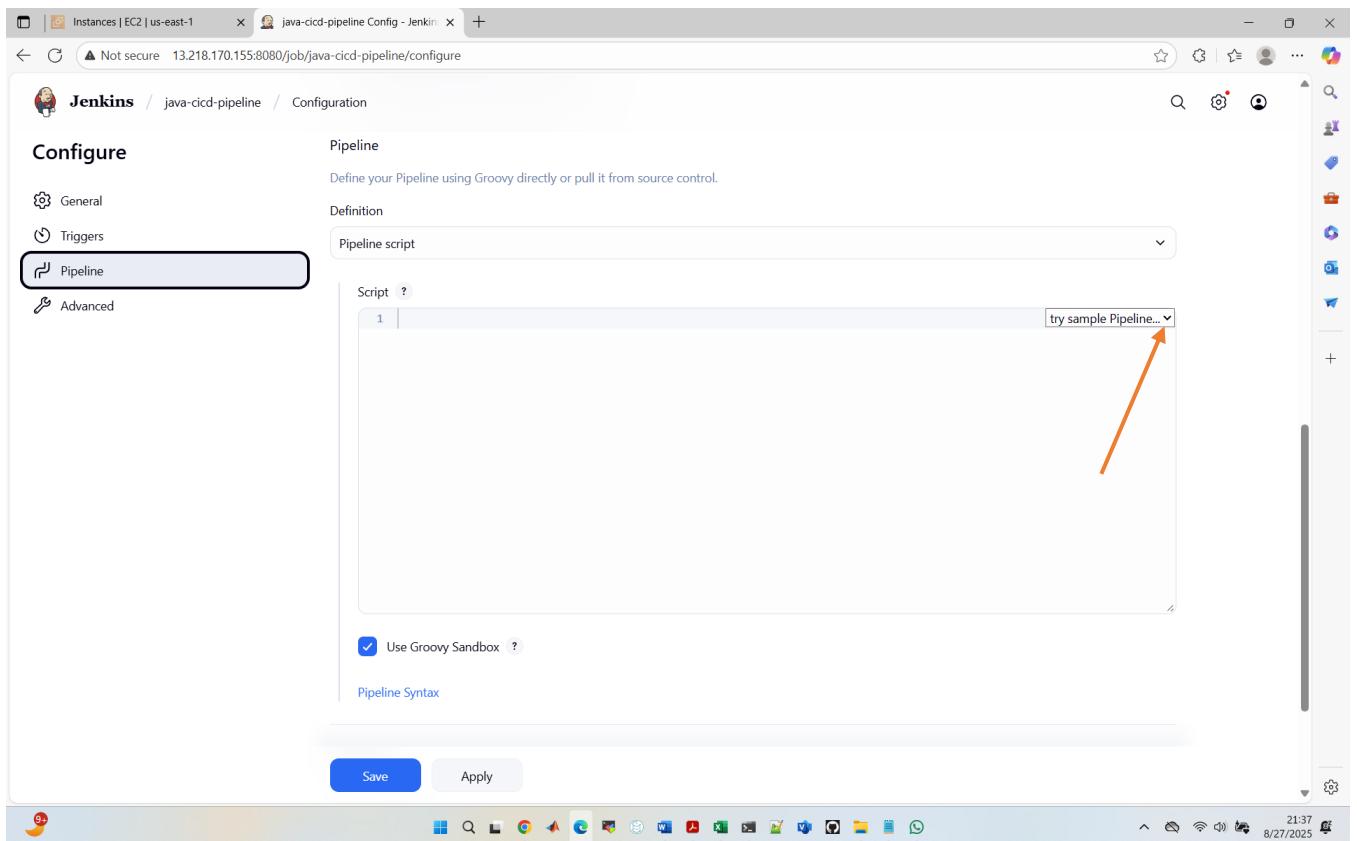
Scripted Pipeline: Allows writing pipelines as Groovy scripts. Provides more flexibility and control with Groovy scripting. Suitable for complex scenarios and advanced control flow. Can be verbose and complex compared to Declarative Pipeline.

In this project, we will be using declarative pipeline, where we are going to use simple English patterns. This will be demonstrated with a simple “Hello World” script.

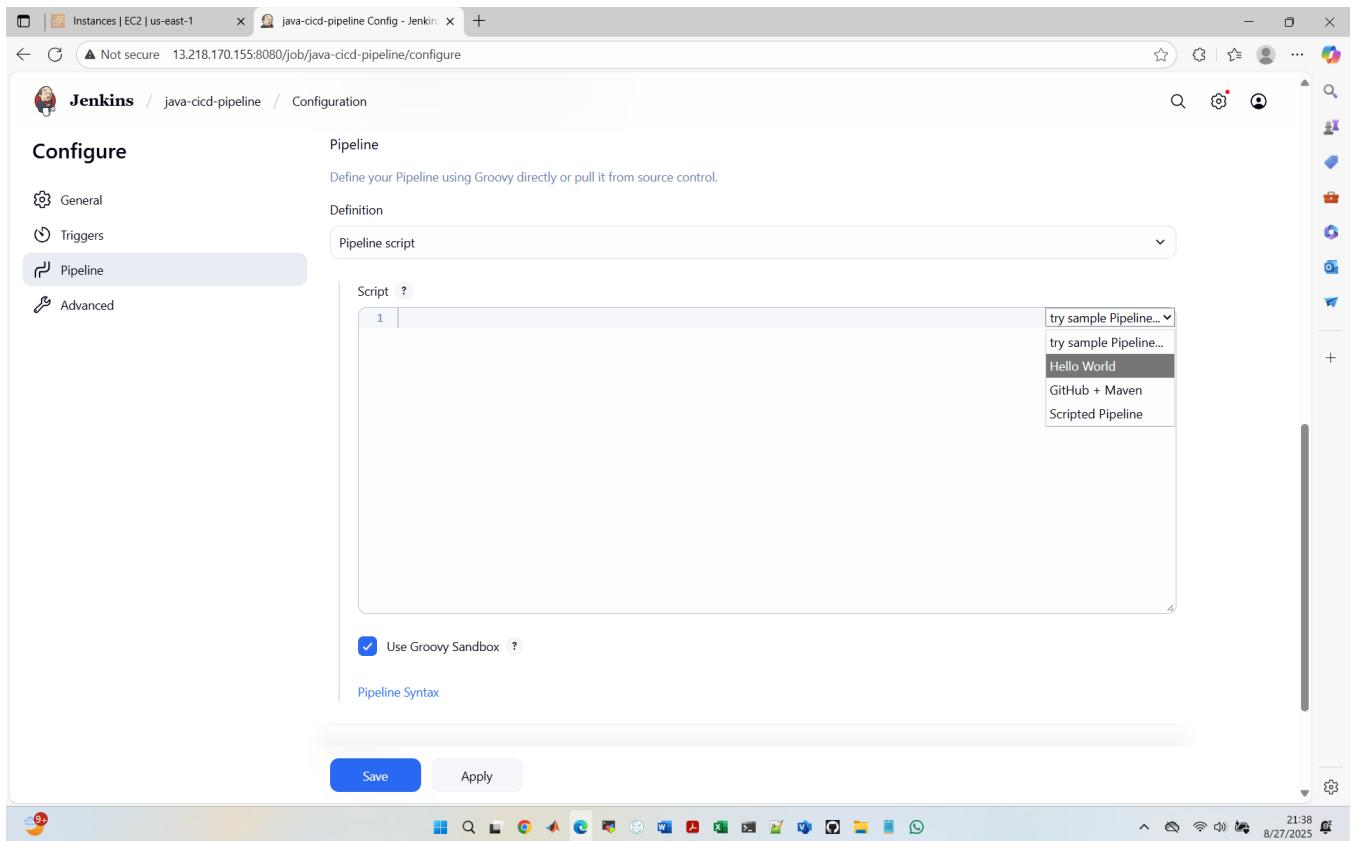
Go to your Jenkins browser



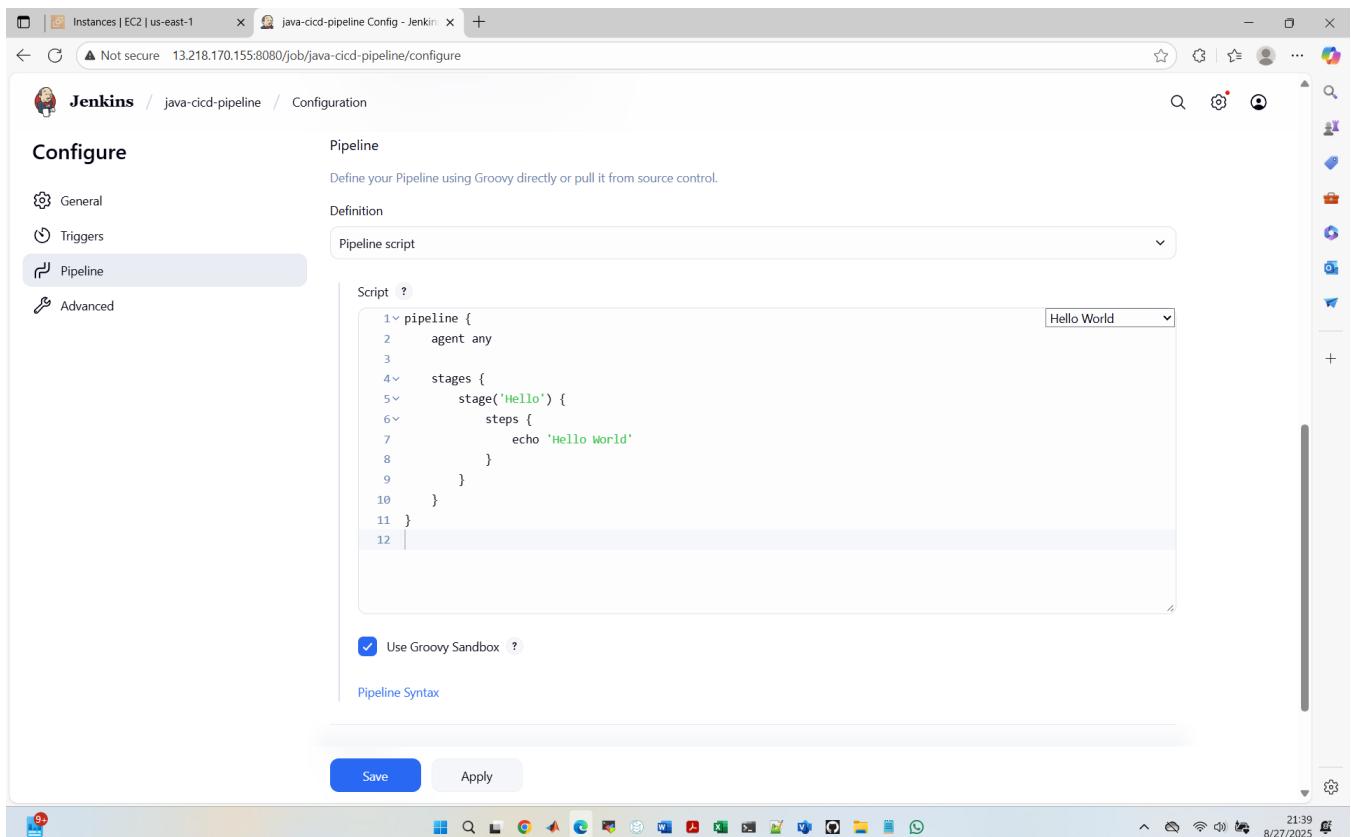
Click on “Pipeline” on the left-hand side



Click on the drop down on “Try sample Pipeline”



Select “Hello World”



Click on “Apply”, followed by clicking on “Save”

Build Now

Configure

Delete Pipeline

Stages

Rename

Pipeline Syntax

No builds

On the left-hand side of the dashboard, there are many options. We will build the pipeline by clicking on “Build Now” and refresh the page.

Jenkins / java-cicd-pipeline

java-cicd-pipeline

Java CI/CD Pipeline

Build Now

Status Changes Build Now Configure Delete Pipeline Stages Rename Pipeline Syntax

Builds

Today #1 4:21 AM

REST API Jenkins 2.516.2

Click on "#1"

Instances | EC2 | us-east-1 | Not secure | 3.94.194.241:8080/job/java-cicd-pipeline/1 - Jenkins

#1 (Aug 28, 2025, 4:21:21 AM)

Started by user admin
Started 1 min 14 sec ago Took 2.6 sec

Console Output

Edit Build Information

Delete build '#1'

Timings

Pipeline Overview

Restart from Stage

Replay

Pipeline Steps

Workspaces

</> No changes.

REST API Jenkins 2.516.2

Then click on "Console Output"

The screenshot shows a Jenkins pipeline console output. The pipeline script is as follows:

```
Started by user Sidney Ebot
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/java-cicd-pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello)
[Pipeline] echo
Hello World
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

An orange arrow points from the text "Started by user Sidney Ebot" to the word "Sidney".

Here you can see the user, that is me because I am the one. If the code is pushed from GitHub, the developer who pushed the code names will appear there as the user.

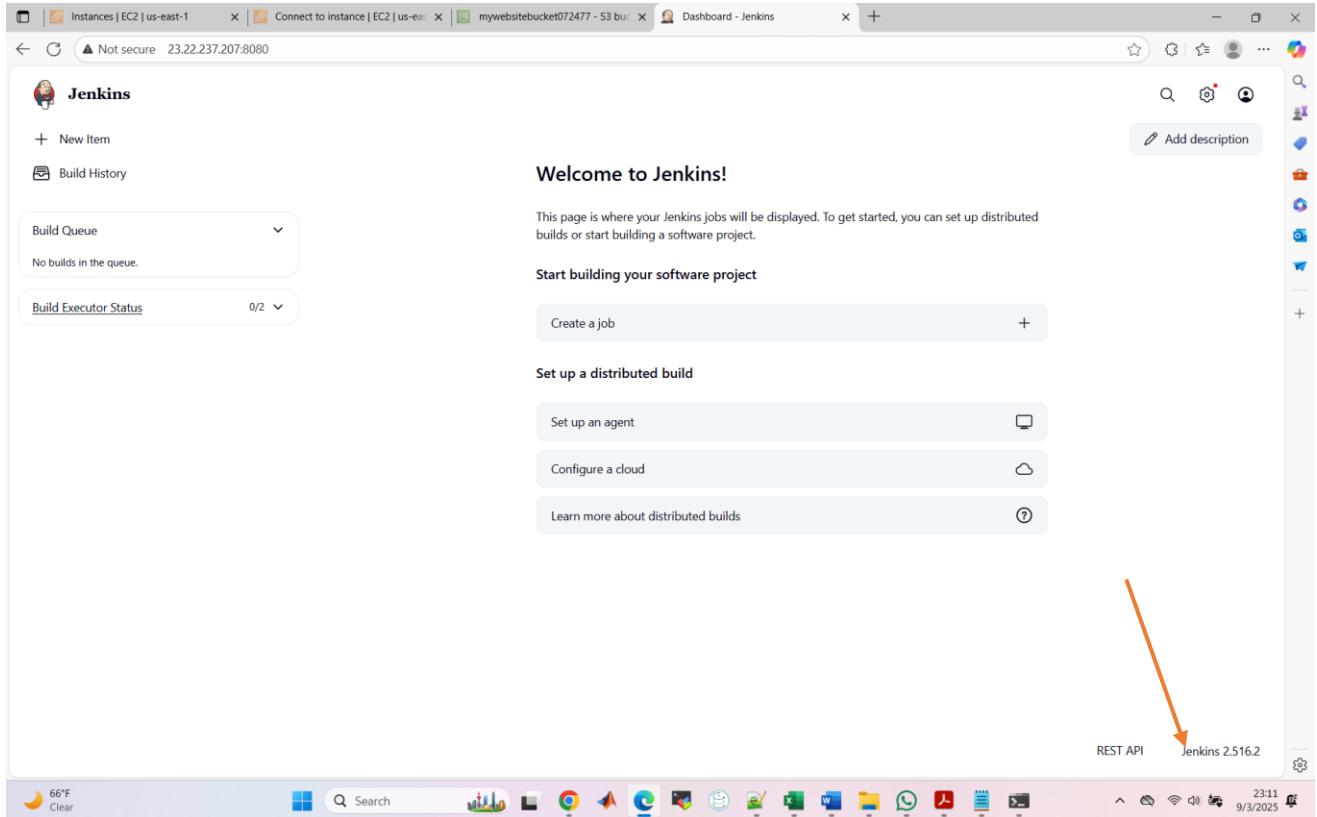
STEP 8: Install and Configure JDK on Jenkins

We will install JDK plugins and configure JDK on Jenkins

PART 1: Install JDK Plugin

We will install the plugin “**Eclipse Temurin Installer**”

Go to “**Manage Jenkins**”



Click on “**Jenkins 2.516.2**”

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job +

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds ?

REST API Jenkins 2.516.2

Select “About Jenkins”

Jenkins

Version 2.516.2

The leading open source automation server which enables developers around the world to reliably build, test, and deploy their software.

Mavenized dependencies Static resources License and dependency information for plugins

Name	Maven ID	License
“Java Concurrency in Practice” book annotations	net.jcip:jcip-annotations:1.0	Creative Commons Attribution License
Annotation Indexer	org.jenkins-ci.annotation-indexer:1.18	MIT License
ANTLR 4 Runtime	org.antlr:antlr4-runtime:4.13.2	BSD-3-Clause
Apache Ant Core	org.apache.ant:ant:1.10.15	The Apache Software License, Version 2.0
Apache Ant Launcher	org.apache.ant:ant-launcher:1.10.15	The Apache Software License, Version 2.0
Apache Commons BeanUtils	commons-beanutils:commons-beanutils:1.11.0	Apache-2.0

Click on “Manage Jenkins”

The screenshot shows the Jenkins 'About Jenkins' page. On the left, there's a sidebar with links like 'System Configuration', 'System', 'Tools', 'Plugins', 'Nodes', 'Clouds', and 'Appearance'. Below that is a navigation menu with 'Security', 'Credentials', 'Credential Providers', and 'Users'. Under 'Status Information', it lists 'System Information' and 'Java Concurrency in Practice' book annotations. The main content area features the Jenkins logo and a table of plugin information:

Maven ID	License
net.jcip:jcip-annotations:1.0	Creative Commons Attribution License
org.jenkins-ci:annotation-indexer:1.18	MIT License
org.antlr:antlr4-runtime:4.13.2	BSD-3-Clause
org.apache.ant:ant:1.10.15	The Apache Software License, Version 2.0
org.apache.ant:ant-launcher:1.10.15	The Apache Software License, Version 2.0
commons-beanutils:commons-beanutils:1.11.0	Apache-2.0

At the bottom, there's a link to 'Plugin Manager'.

Select “Plugins”

The screenshot shows the Jenkins 'Updates - Plugins' page. On the left, there's a sidebar with 'Updates' selected, followed by 'Available plugins', 'Installed plugins', and 'Advanced settings'. A search bar at the top right says 'Search plugin updates'. The main content area shows a message: 'No updates available' with a small lock icon.

Click on “Available Plugins”

The screenshot shows the Jenkins plugin manager interface. The left sidebar has tabs for 'Updates', 'Available plugins' (which is selected), 'Installed plugins', and 'Advanced settings'. The main area is titled 'Plugins' and contains a search bar with placeholder text 'Search available plugins'. A blue 'Install' button is at the top right. Below it is a table with columns: 'Install', 'Name', 'Released', and 'Health'. The table lists several plugins:

Install	Name	Released	Health
<input type="checkbox"/>	PAM Authentication 1.12 Security	6 mo 3 days ago	95
<input type="checkbox"/>	JavaMail API 1.6.2-11 Library plugins (for use by other plugins)	6 mo 13 days ago	96
<input type="checkbox"/>	Command Agent Launcher 123.v37cfdc92ef6 Agent Management	4 mo 26 days ago	100
<input type="checkbox"/>	Oracle Java SE Development Kit Installer 83.v417146707a_3d Allows the Oracle Java SE Development Kit (JDK) to be installed via download from Oracle's website.	7 mo 15 days ago	100
<input type="checkbox"/>	SSH server 3.374.v19b_d59ce6610 Adds SSH server functionality to Jenkins, exposing CLI commands through it.	24 days ago	100
<input type="checkbox"/>	JSch dependency 0.2.16-95.v3eeeb_55fa_b_78 Library plugins (for use by other plugins) Miscellaneous	6 mo 10 days ago	100
<input type="checkbox"/>	Authentication Tokens API 1.144.v5ff4a_5ec5c33 This plugin provides an API for converting credentials into authentication tokens in Jenkins.	29 days ago	100
<input type="checkbox"/>	Javadoc 354.vee1a_660b_4990 This plugin adds Javadoc support to Jenkins.	1 mo 3 days ago	100

At the bottom left, the URL is https://plugins.jenkins.io/javamail-api. The system tray shows a weather icon (65°F) and a date/time stamp (9/3/2025 23:31).

Search for “Eclipse”

The screenshot shows the Jenkins plugin manager interface after a search for 'Eclipse'. The search bar now contains 'Eclipse'. The table in the center lists three results:

Install	Name	Released	Health
<input type="checkbox"/>	Jersey 2 API 2.47-165.ve7809a_3e87e0 Library plugins (for use by other plugins)	2 mo 20 days ago	100
<input checked="" type="checkbox"/>	Eclipse Temurin installer 146.v1898676a_f04e Provides an installer for the JDK tool that downloads the Eclipse Temurin™ build based upon OpenJDK from the AdoptOpenJDK Working Group.	2 mo 1 day ago	100
<input type="checkbox"/>	Buckminster 1.1.1 Build Tools	12 yr ago	11

An orange arrow points to the 'Install' checkbox for the 'Eclipse Temurin installer' row. The table rows contain additional information and warnings:

- The first row for Jersey 2 API includes a note: "This plugin provides the JAX-RS 2.1 and Jersey 2 APIs for other plugins."
- The second row for Eclipse Temurin installer includes a note: "Provides an installer for the JDK tool that downloads the Eclipse Temurin™ build based upon OpenJDK from the AdoptOpenJDK Working Group."
- The third row for Buckminster includes a warning box:
 - 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](#).
 - This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.

At the bottom left, the URL is https://plugins.jenkins.io/eclipse-temurin-installer. The system tray shows a weather icon (65°F) and a date/time stamp (9/3/2025 23:32).

Select “Eclipse Temurin Installer”

Jenkins / Manage Jenkins / Plugins

Eclipse

Install	Name	Released	Health
<input type="checkbox"/>	Jersey 2 API 2.47-165.ve7809a_3e87e0	2 mo 20 days ago	100
<input checked="" type="checkbox"/>	Eclipse Temurin installer 146.v1898676a_f04e	2 mo 1 day ago	100
<input type="checkbox"/>	Buckminster 1.1.1	12 yr ago	11

Eclipse Update Site 1.2

Upcoming Earnings

Search

Downloads

File Explorer

Task List

Terminal

Logs

Build History

Dashboard

Help

23:33 9/3/2025

Click on “Install”

Jenkins / Manage Jenkins / Plugins

Plugins

Plugin	Status
Github	Success
GitHub Branch Source	Success
Pipeline: GitHub Groovy Libraries	Success
Metrics	Success
Pipeline Graph View	Success
Git	Success
EDDSA API	Success
Trilead API	Success
SSH Build Agents	Success
Matrix Authorization Strategy	Success
LDAP	Success
jsoup API	Success
Email Extension	Success
Mailer	Success
Theme Manager	Success
Dark Theme	Success
Loading plugin extensions	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 2.516.2

54°F Clear

Search

File Explorer

Task List

Terminal

Logs

Build History

Dashboard

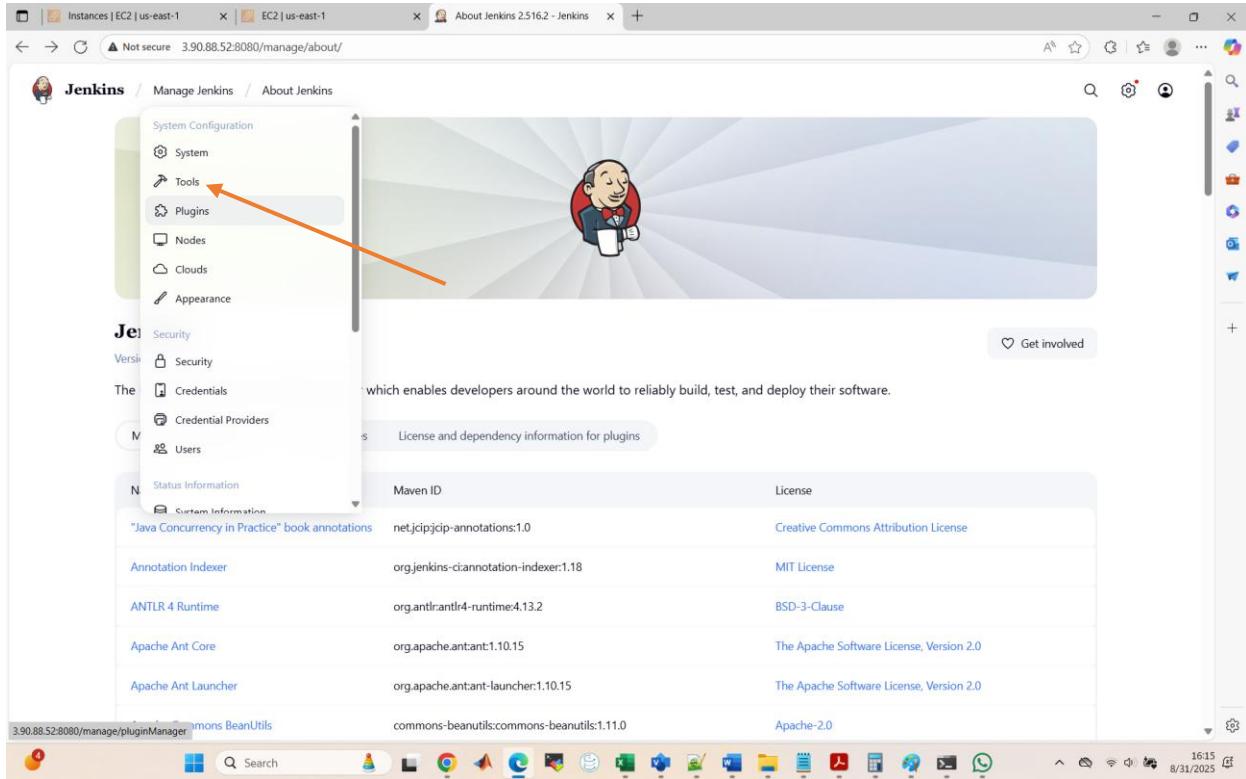
Help

23:59 9/7/2025

PART 2: Configure JDK as a tool on Jenkins

We will now configure JDK on Jenkins. We will use JDK-11 and we will install from adoptium.net.

Click on “Manage Jenkins”



The screenshot shows the Jenkins 'About Jenkins' page. On the left, there is a sidebar with the following navigation options:

- System Configuration
- System
- Tools** (highlighted with an orange arrow)
- Plugins
- Nodes
- Clouds
- Appearance

The main content area features a cartoon character holding a coffee cup. Below the character, there is a section titled "Status Information" which lists various Maven dependencies and their licenses. At the bottom of the page, there is a footer bar with icons for search, refresh, and other system functions.

Maven ID	License
net.jcip:jcip-annotations:1.0	Creative Commons Attribution License
org.jenkins-ci:annotation-indexer:1.18	MIT License
org.antlr:antlr4-runtime:4.13.2	BSD-3-Clause
org.apache.ant:ant:1.10.15	The Apache Software License, Version 2.0
org.apache.ant:ant-launcher:1.10.15	The Apache Software License, Version 2.0
commons-beanutils:commons-beanutils:1.11.0	Apache-2.0

Click on “tools”

Jenkins / Manage Jenkins / 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

Path to Git executable ?

Save Apply

Click on “Add JDK”

Jenkins / Manage Jenkins / Tools

JDK installations

Add JDK

≡ JDK

Name

① Required

JAVA_HOME

Install automatically ?

Add JDK

Git installations

≡ Git

Save Apply

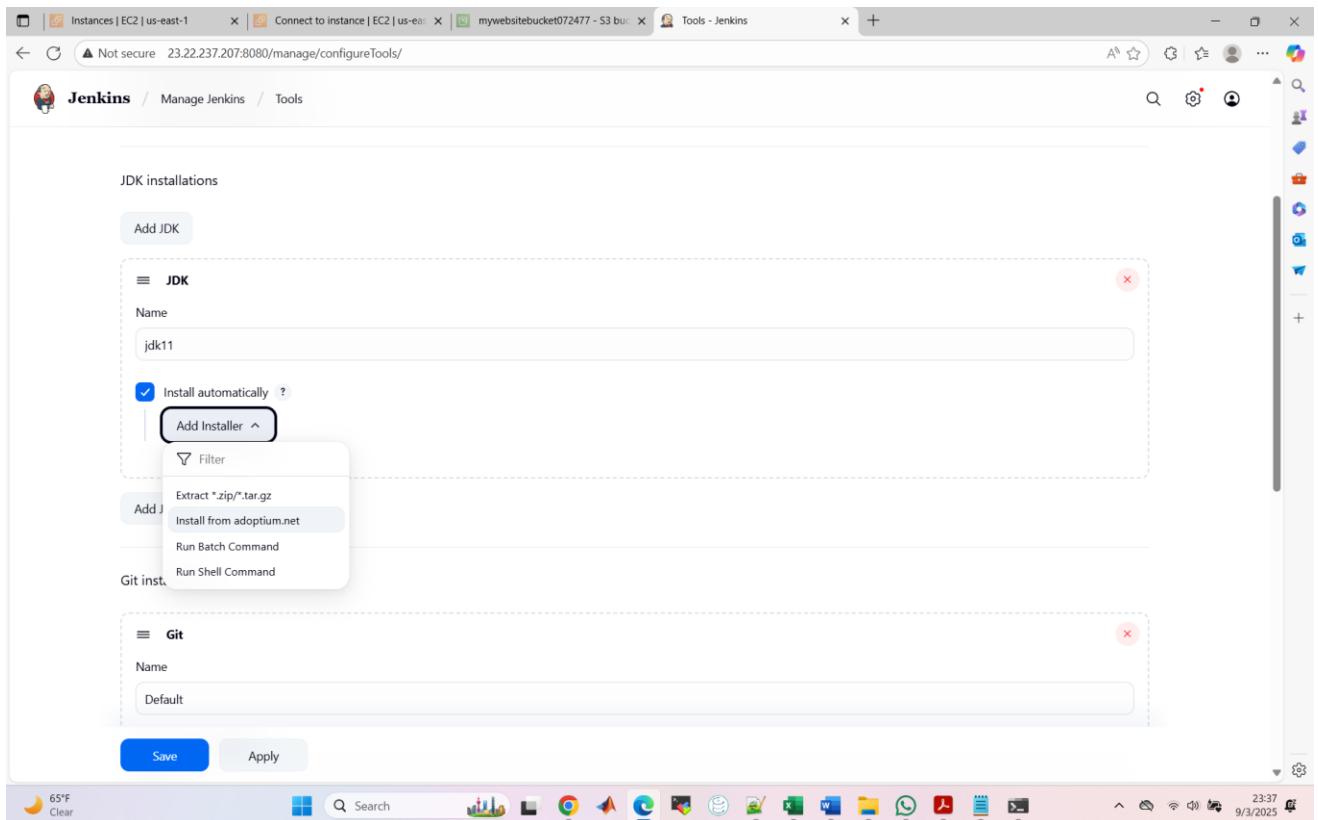
On Name enter “**jdk11**”

The screenshot shows the Jenkins Tools configuration page. In the 'JDK installations' section, there is a form for adding a new JDK. The 'Name' field contains 'jdk11'. Below it, there is a 'JAVA_HOME' field and a checkbox labeled 'Install automatically ?'. A red arrow points to this checkbox. At the bottom of the form are 'Add JDK' and 'Save' buttons.

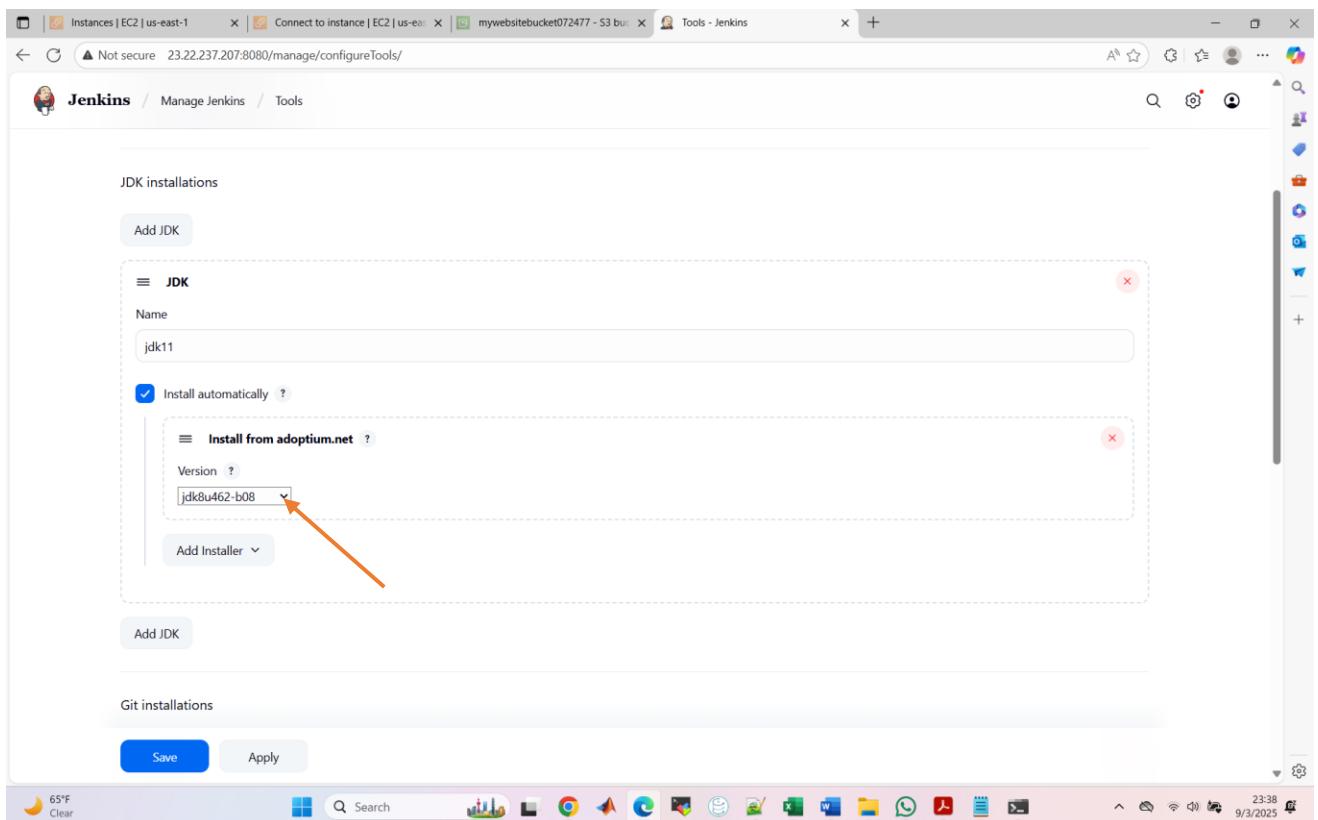
Select “Install Automatically”

The screenshot shows the Jenkins Tools configuration page. In the 'JDK installations' section, the 'Install automatically ?' checkbox is checked. A red arrow points to the 'Add Installer' dropdown menu, which is currently open. The 'Git installations' section is also visible below.

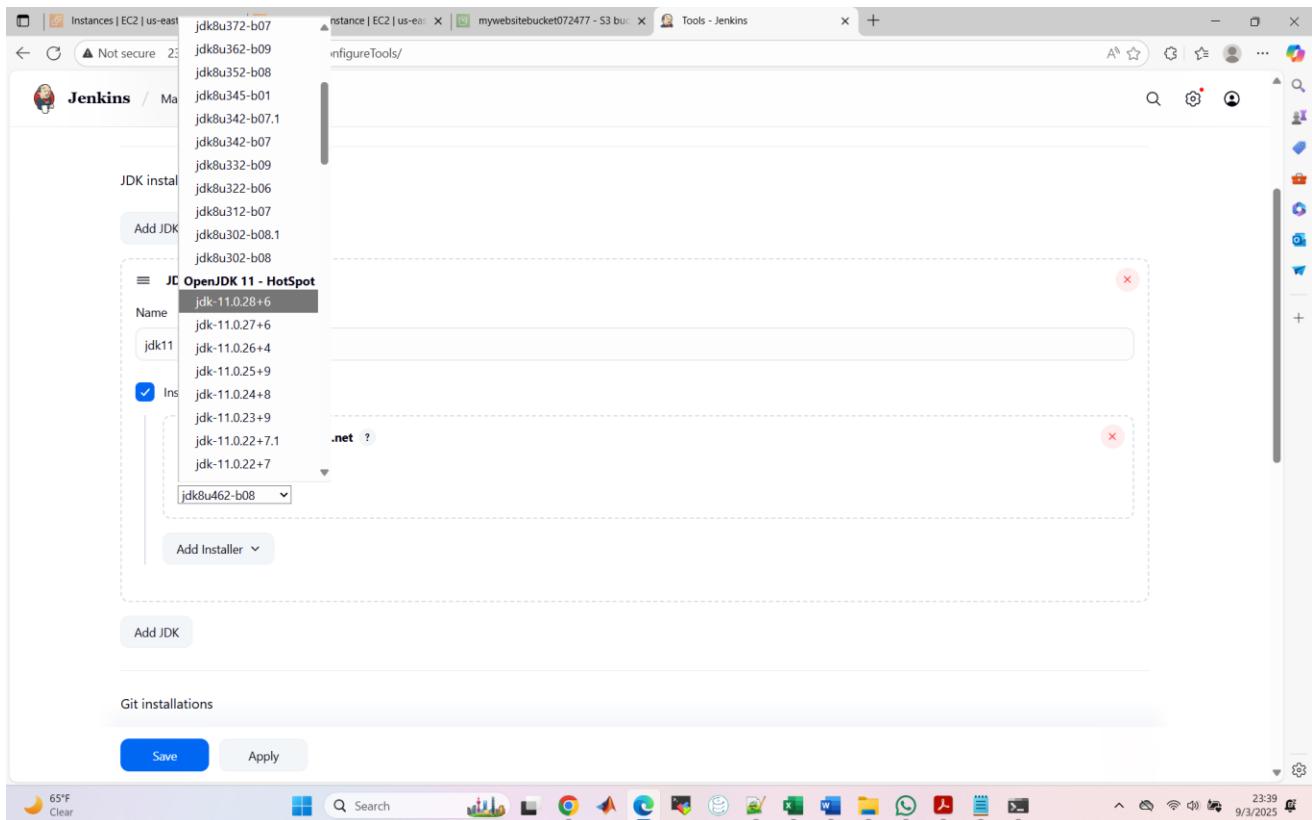
Click on the drop down on “Add Installer”



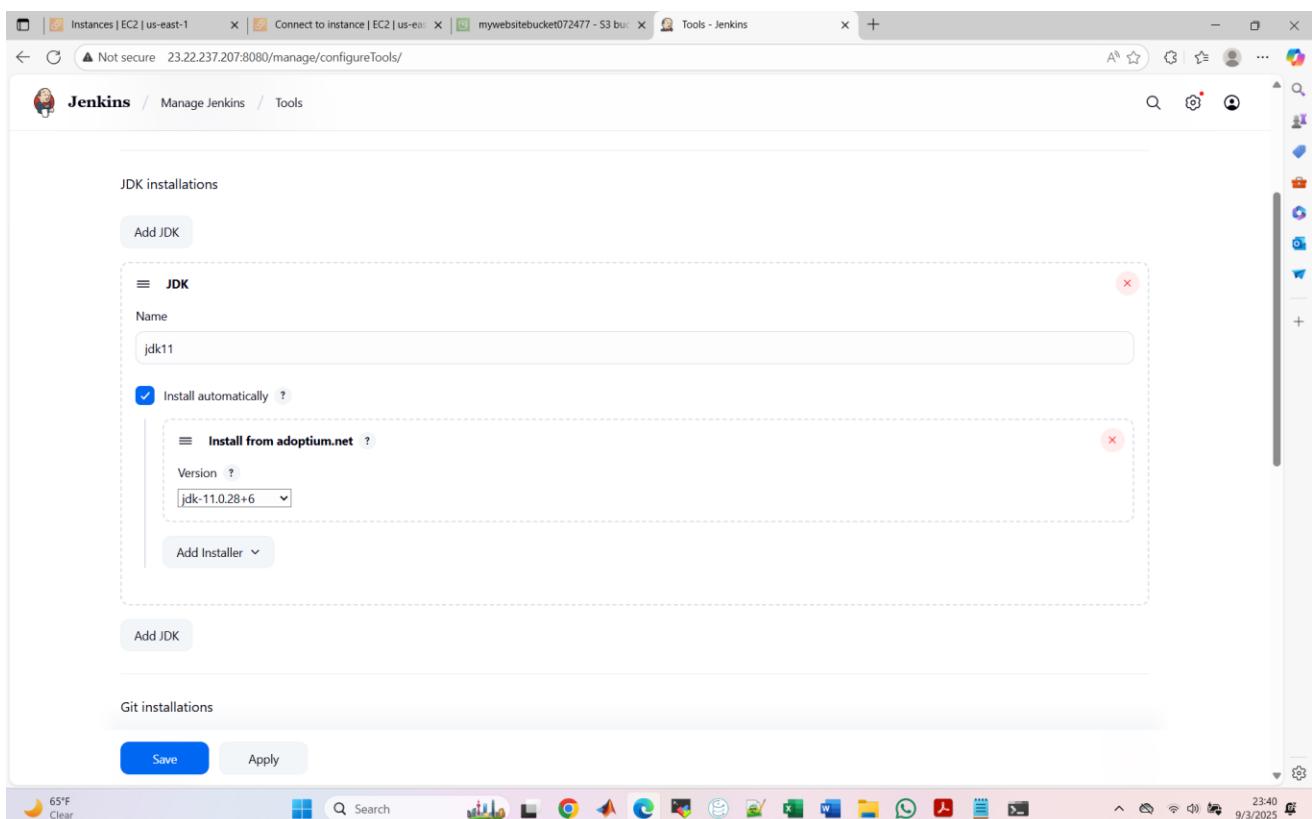
Select “Install from adoptium.net”



Click on the drop down on “Version”



Select version “**jdk-11.0.28+6**”



Click on “**apply**” then “**Save**”

The screenshot shows the Jenkins Manage Jenkins interface. At the top, there are several tabs: 'sosoebot.org - details', 'SecurityGroup | EC2 | us-east-1', 'USCIS Online Account | Welcome', and 'Manage Jenkins - Jenkins'. The main content area has a header 'Manage Jenkins' with a search bar and three buttons: '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](#).'. On the right side, there is a vertical sidebar with icons for various Jenkins features.

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.
- Credential Providers**: Configure the credential providers and types.
- Users**: Create/delete/modify users that can log in to this Jenkins.

Status Information

- System Information**
- System Log**
- Load Statistics**

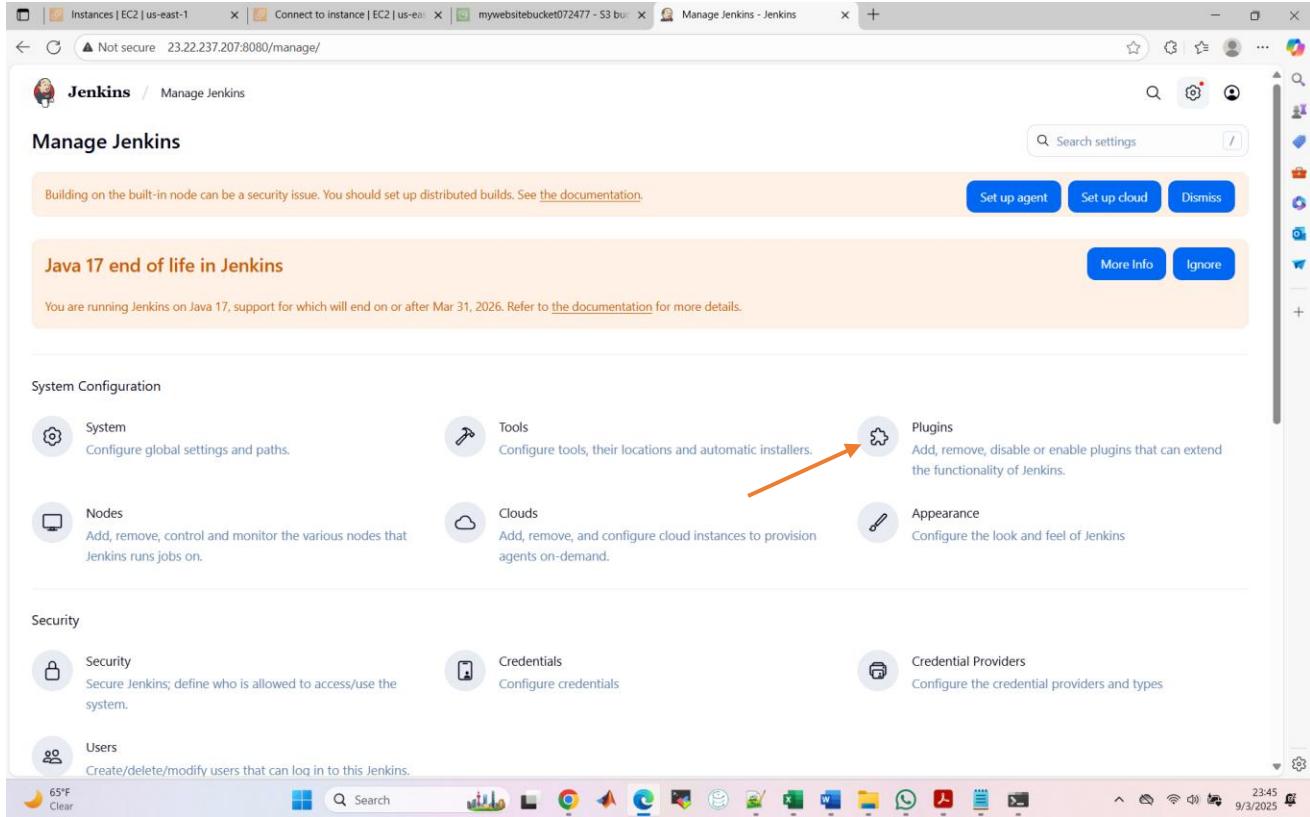
At the bottom, there is a taskbar with various application icons and a system tray showing the date and time (9/8/2025).

STEP 9: Install Maven plugin and Configure Maven on Jenkins

Here we will install Maven plugin and configure Maven on Jenkins.

PART 1: Install the Maven Plugin

We will install the plugin “**Maven Integration**”



The screenshot shows the Jenkins 'Manage Jenkins' interface. At the top, there are several tabs: Instances | EC2 | us-east-1, Connect to instance | EC2 | us-east-1, mywebsitebucket072477 - S3 bucket, and Manage Jenkins - Jenkins. Below the tabs, the URL is 23.22.237.207:8080/manage/. The main content area is titled 'Manage Jenkins' with a sub-section 'Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).'. There are three buttons: Set up agent, Set up cloud, and Dismiss. A message about Java 17 end of life is displayed: 'Java 17 end of life in Jenkins' with 'More Info' and 'Ignore' buttons. The 'System Configuration' section contains icons for System, Tools, Nodes, Clouds, Plugins, Appearance, Security, Credentials, and Credential Providers. The 'Plugins' icon is highlighted with an orange arrow. The bottom of the screen shows a Windows taskbar with various application icons and system status indicators.

Click on “**Plugins**”

The screenshot shows the Jenkins Plugins page. The left sidebar has tabs for 'Updates' (selected), 'Available plugins', 'Installed plugins', and 'Advanced settings'. A search bar at the top right says 'Search plugin updates'. Below it, a message says 'No updates available' with a shopping bag icon.

Click on “Available Plugins”

The screenshot shows the Jenkins Available Plugins page. The left sidebar has tabs for 'Updates' (selected), 'Available plugins' (selected), 'Installed plugins', and 'Advanced settings'. A search bar at the top right says 'Search available plugins' with an 'Install' button. The main area lists various plugins with columns for 'Name', 'Released', and 'Health' (indicated by green circles).

Install	Name	Released	Health
<input type="checkbox"/>	PAM Authentication 1.12 Security	6 mo 3 days ago	95
<input type="checkbox"/>	JavaMail API 1.6.2-11 Library plugins (for use by other plugins)	6 mo 13 days ago	96
<input type="checkbox"/>	Command Agent Launcher 123.v37cfcd92ef67 Agent Management	4 mo 26 days ago	100
<input type="checkbox"/>	Oracle Java SE Development Kit Installer 83.v417146707a_3d Allows the Oracle Java SE Development Kit (JDK) to be installed via download from Oracle's website.	7 mo 15 days ago	100
<input type="checkbox"/>	SSH server 3.374.v19b_d59ce6610 Adds SSH server functionality to Jenkins, exposing CLI commands through it.	24 days ago	100
<input type="checkbox"/>	JSch dependency 0.2.16-95.v3eechb_55fb_b_78 Library plugins (for use by other plugins) Miscellaneous	6 mo 10 days ago	100
<input type="checkbox"/>	Authentication Tokens API 1.144.v5ff4a_5ec5c33 This plugin provides an API for converting credentials into authentication tokens in Jenkins.	29 days ago	100
<input type="checkbox"/>	Javadoc 354.vee1a_660b_4990 This plugin adds Javadoc support to Jenkins.	1 mo 3 days ago	100

Search for “Maven”

The screenshot shows the Jenkins plugin manager interface. A search bar at the top contains the text "maven". Below the search bar is a table listing several Jenkins plugins:

Install	Name	Released	Health
<input type="checkbox"/>	Maven Integration 3.27 Build Tools	23 days ago	100
<input type="checkbox"/>	Config File Provider 994.v3d4a_5fa_f353a_	26 days ago	100
<input type="checkbox"/>	Jira 3.19 External Site/Tool Integrations Maven jira	8 days 6 hr ago	100
<input type="checkbox"/>	Pipeline Maven Integration 1567.vb_2c3a_2116860 pipeline Maven	14 days ago	97
<input type="checkbox"/>	Artifactory 4.0.8 pipeline	1 yr 1 mo ago	82

A tooltip at the bottom left of the table area says "23.22.237.207:8080/manage/pluginManager/available?filter=External Site/Tool Integrations". The browser address bar shows the URL "23.22.237.207:8080/manage/pluginManager/available". The system tray at the bottom indicates it's 65°F, 9/3/2025, and 23:47.

Select “Maven Integration”

The screenshot shows the Jenkins plugin manager interface, similar to the previous one but with a key difference: the "Maven Integration" plugin has its checkbox checked. An orange arrow points from the text "Click on ‘Install’" to the "Install" button, which is now highlighted in blue.

The table of plugins remains the same as in the first screenshot, with the "Maven Integration" row being the most prominent due to the checked checkbox.

Click on “Install”

The screenshot shows the Jenkins 'Manage Jenkins / Plugins' page. The left sidebar lists navigation options: Updates, Available plugins, Installed plugins, Advanced settings, and Download progress. The 'Download progress' option is currently selected. The main content area displays a list of installed and updated plugins, each followed by a green checkmark and the word 'Success'. The list includes: Git, EDDSA API, Trilead API, SSH Build Agents, Matrix Authorization Strategy, LDAP, jsoup API, Email Extension, Mailer, Theme Manager, Dark Theme, Loading plugin extensions, Eclipse Temurin installer, Loading plugin extensions, Dev Tools Symbols API, Javadoc, JSch dependency, Maven Integration, and Loading plugin extensions.

→ [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 2.516.2

PART B: Configure Maven as a tool on Jenkins

We will configure Maven on Jenkins. Go to “Manage Jenkins”

The screenshot shows the Jenkins 'Manage Jenkins' page. The left sidebar has several sections: Plugins, Updates, Available plugins, Installed plugins, Advanced settings, System Configuration (with sub-options: System, Tools, Plugins, Nodes, Clouds, Appearance), Security (with sub-options: Security, Credentials, Credential Providers, Users), and Status Information. The 'Tools' option under 'System Configuration' is currently selected. A large central panel titled 'Load progress' contains the same instructions as the previous screenshot: '→ [Go back to the top page](#)' and '(you can start using the installed plugins right away)', followed by a checkbox for 'Restart Jenkins when installation is complete and no jobs are running'. The bottom status bar shows the URL '23.22.237.207:8080/manage/configureTools' and the Jenkins version 'Jenkins 2.516.2'.

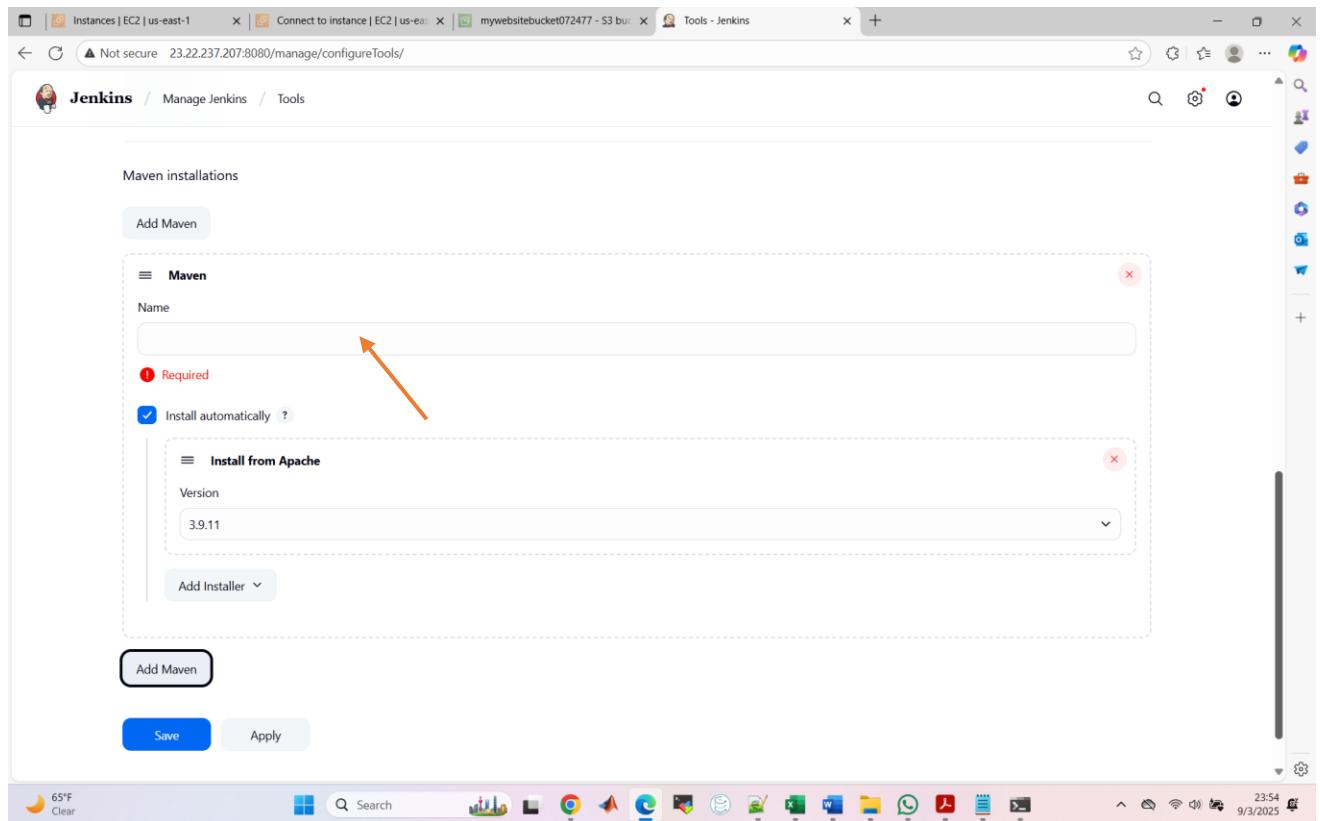
Select “tools”

The screenshot shows the Jenkins 'Tools' configuration page. Under 'Maven Configuration', there are sections for 'Default settings provider' (set to 'Use default maven settings') and 'Default global settings provider' (set to 'Use default maven global settings'). Under 'JDK installations', it says 'JDK installations' and 'Edited'. Under 'Git installations', there is a 'Git' section with a 'Name' field containing 'Default'. Below it is a 'Path to Git executable' field and buttons for 'Save' and 'Apply'. The bottom status bar shows the date as 9/3/2025.

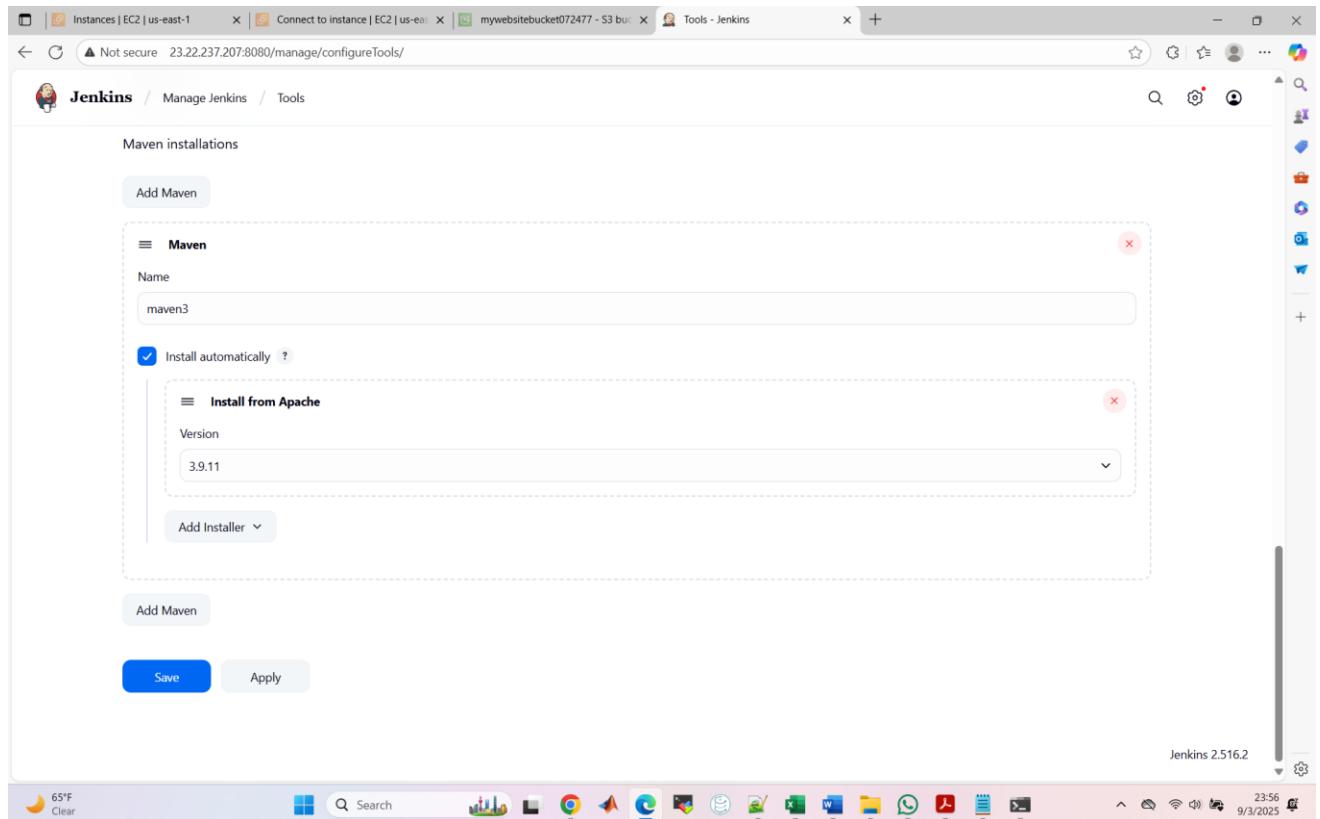
Scroll down to “Maven Installations”

The screenshot shows the Jenkins 'Tools' configuration page. It includes sections for 'Install automatically' (unchecked), 'Add Git', 'Gradle installations' (with 'Add Gradle' button), 'Ant installations' (with 'Add Ant' button), and 'Maven installations' (with 'Add Maven' button). At the bottom are 'Save' and 'Apply' buttons. The bottom status bar shows the Jenkins version as 2.516.2.

Click on “Add Maven”



On name, we will enter “**maven3**”



Click on “Apply” followed by “Save”

The screenshot shows the Jenkins Manage Jenkins page. At the top, there is a warning message: "Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#)." Below this, there are several configuration sections:

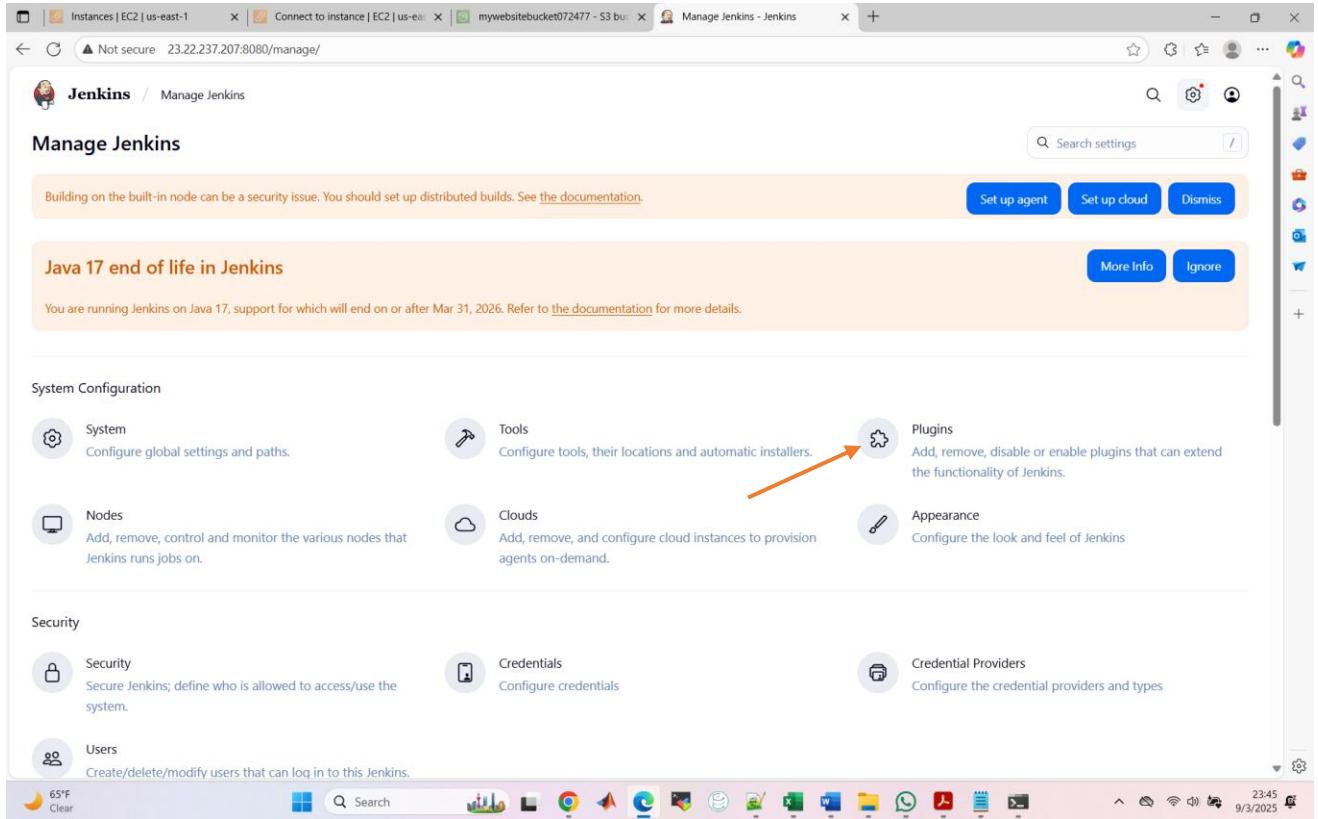
- System Configuration**: Includes links for System (global settings), Tools (configure tools), Plugins (manage plugins), Nodes (control nodes), Clouds (configure clouds), and Appearance (configure look and feel).
- Security**: Includes links for Security (secure Jenkins), Credentials (configure credentials), and Credential Providers (configure providers).
- Users**: Create/delete/modify users.
- Status Information**: Includes links for System Information, System Log, and Load Statistics. It also displays system status (54°F Clear) and a toolbar with various application icons.

STEP 10: Install and Configure Docker on Jenkins

Here we will install “Docker”, “Docker Commons”, “Docker Pipeline” and “Docker API” plugins and configure Dependency Checks on Jenkins.

PART 1: Install all the Plugins

We will install all the docker plugins needed for this project



The screenshot shows the Jenkins 'Manage Jenkins' interface. At the top, there are several tabs: Instances | EC2 | us-east-1, Connect to instance | EC2 | us-east-1, mywebsitebucket072477 - 53 buckets, and Manage Jenkins - Jenkins. Below the tabs, the URL is 23.22.237.207:8080/manage/. The main content area is titled 'Manage Jenkins'. A banner at the top states: 'Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).'. Below this are two buttons: 'Set up agent' and 'Set up cloud', followed by 'Dismiss'. A message about Java 17 end of life is displayed: 'Java 17 end of life in Jenkins' (You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to [the documentation](#) for more details.) with 'More Info' and 'Ignore' buttons. The 'System Configuration' section contains several items: '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). The 'Plugins' item has an orange arrow pointing to it from the left. The bottom of the screen shows a taskbar with various icons and a system tray indicating the date as 9/3/2025 and time as 23:45.

Click on “Plugins”

The screenshot shows the Jenkins Plugins page. The left sidebar has tabs for 'Updates' (selected), 'Available plugins', 'Installed plugins', and 'Advanced settings'. A search bar at the top right says 'Search plugin updates'. Below it, a message says 'No updates available' with a shopping bag icon.

Click on “Available Plugins”

The screenshot shows the Jenkins Available Plugins page. The left sidebar has tabs for 'Updates' (selected), 'Available plugins' (selected), 'Installed plugins', and 'Advanced settings'. A search bar at the top right says 'Search available plugins' with an 'Install' button. The main area lists various available plugins:

Install	Name	Released	Health
<input type="checkbox"/>	PAM Authentication 1.12 Security	6 mo 3 days ago	95
<input type="checkbox"/>	JavaMail API 1.6.2-11 Library plugins (for use by other plugins)	6 mo 13 days ago	96
<input type="checkbox"/>	Command Agent Launcher 123.v37cfcd92ef67 Agent Management	4 mo 26 days ago	100
<input type="checkbox"/>	Oracle Java SE Development Kit Installer 83.v417146707a_3d Allows the Oracle Java SE Development Kit (JDK) to be installed via download from Oracle's website.	7 mo 15 days ago	100
<input type="checkbox"/>	SSH server 3.374.v19b_d59ce6610 Adds SSH server functionality to Jenkins, exposing CLI commands through it.	24 days ago	100
<input type="checkbox"/>	JSch dependency 0.2.16-95.v3e0cb55fb_b_78 Library plugins (for use by other plugins) Miscellaneous	6 mo 10 days ago	100
<input type="checkbox"/>	Authentication Tokens API 1.144.v5ff4a_5ec5c33 This plugin provides an API for converting credentials into authentication tokens in Jenkins.	29 days ago	100
<input type="checkbox"/>	Javadoc 354.vee1a_660b_4990 This plugin adds Javadoc support to Jenkins.	1 mo 3 days ago	100

Search for “Docker”

Jenkins / Manage Jenkins / Plugins

Plugins

Updates Available plugins Installed plugins Advanced settings Download progress

Search: docker

Install	Name	Released	Health
<input type="checkbox"/>	Docker 1274.vc0203fdf2e74	6 mo 4 days ago	100
<input type="checkbox"/>	Docker Commons 457.v0f62a_94f11a_3	3 mo 10 days ago	100
<input type="checkbox"/>	Docker Pipeline 621.va_73f881d9232	3 mo 10 days ago	96
<input type="checkbox"/>	Docker API 3.5.3-122.v156e51f30c0a_	1 mo 6 days ago	100
<input type="checkbox"/>	docker-build-step 2.12	1 yr 3 mo ago	62

This plugin integrates Jenkins with Docker

Provides the common shared functionality for various Docker-related plugins.

Build and use Docker containers from pipelines.

This plugin provides docker-JAVA API for other plugins.

This plugin allows to add various docker commands to your job as build steps.

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

- CSRF vulnerability and missing permission check

Amazon ECR 1.161.v1a_1e8df852d6

Select “Docker”, “Docker Commons”, “Docker Pipeline” and “Docker API”

Jenkins / Manage Jenkins / Plugins

Updates Available plugins Installed plugins Advanced settings Download progress

Search: docker

Install	Name	Released	Health
<input checked="" type="checkbox"/>	Docker 1274.vc0203fdf2e74	6 mo 4 days ago	100
<input checked="" type="checkbox"/>	Docker Commons 457.v0f62a_94f11a_3	3 mo 10 days ago	100
<input checked="" type="checkbox"/>	Docker Pipeline 621.va_73f881d9232	3 mo 10 days ago	96
<input checked="" type="checkbox"/>	Docker API 3.5.3-122.v156e51f30c0a_	1 mo 6 days ago	100
<input type="checkbox"/>	docker-build-step 2.12	1 yr 3 mo ago	62

This plugin integrates Jenkins with Docker

Provides the common shared functionality for various Docker-related plugins.

Build and use Docker containers from pipelines.

This plugin provides docker-JAVA API for other plugins.

This plugin allows to add various docker commands to your job as build steps.

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

- CSRF vulnerability and missing permission check

Amazon ECR 1.161.v1a_1e8df852d6

Click on “Install”

Jenkins / Manage Jenkins / Plugins

Plugins

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

Plugin	Status
Javadoc	Success
JSch dependency	Success
Maven Integration	Success
Loading plugin extensions	Success
SonarQube Scanner	Success
OWASP Dependency-Check	Success
Loading plugin extensions	Success
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 2.516.2

PART B: Configure Docker on Jenkins

We will configure Dependency Check on Jenkins. Go to “Manage Jenkins”

Jenkins / Manage Jenkins / Plugins

System Configuration

- System
- Tools
- Plugins
- Nodes
- Clouds
- Appearance
- Security
- Credentials
- Credential Providers
- Users
- Status Information

Download progress

→ 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

23.22.237.207:8080/manage/pluginManager/updates

23.22.237.207:8080/manage/configureTools

65°F Clear 11:55 9/7/2025 23:53 9/3/2025

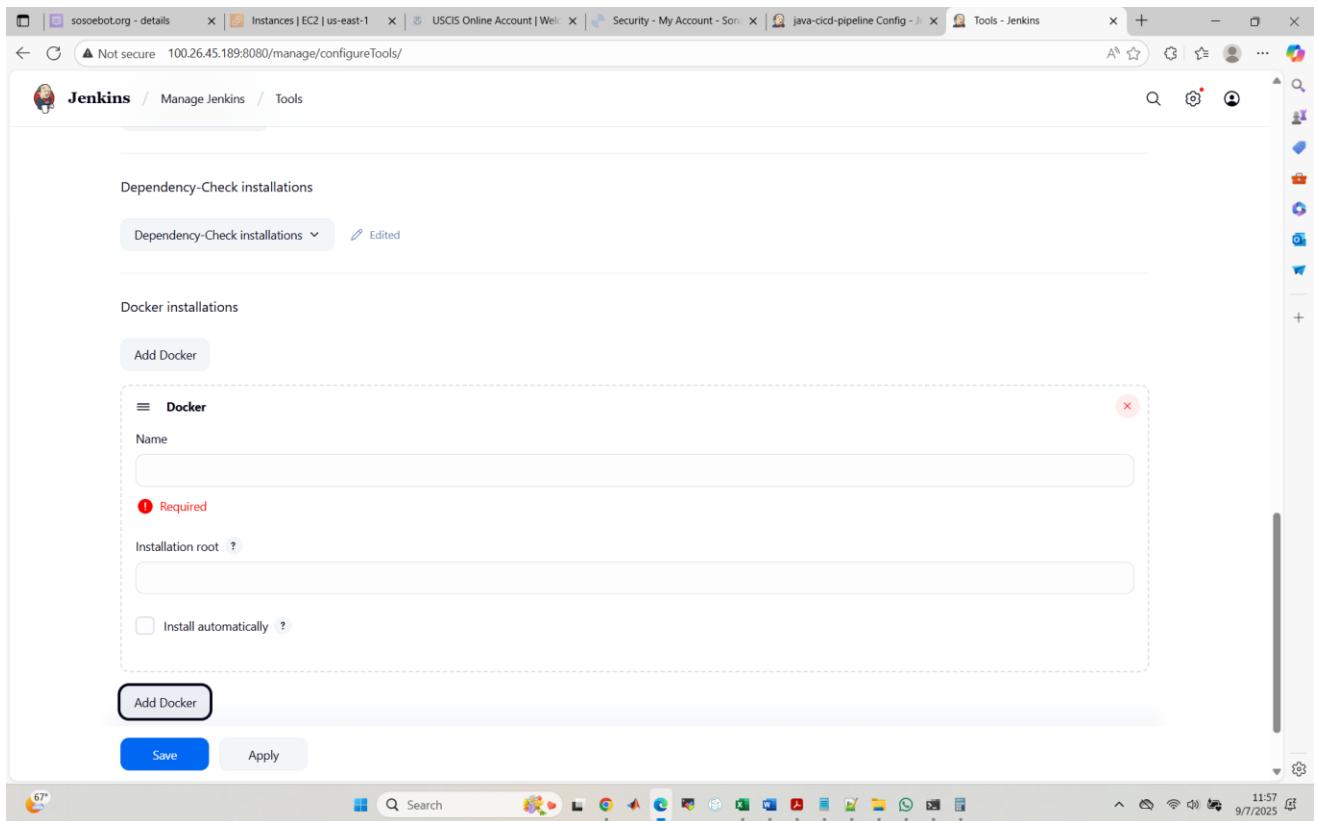
Select “tools”

The screenshot shows the Jenkins 'Tools' configuration page. It includes sections for Maven Configuration, JDK installations, and Git installations. The Git section is expanded, showing a 'Git' configuration panel with fields for Name (set to 'Default') and Path to Git executable. Buttons for 'Save' and 'Apply' are visible at the bottom of this panel.

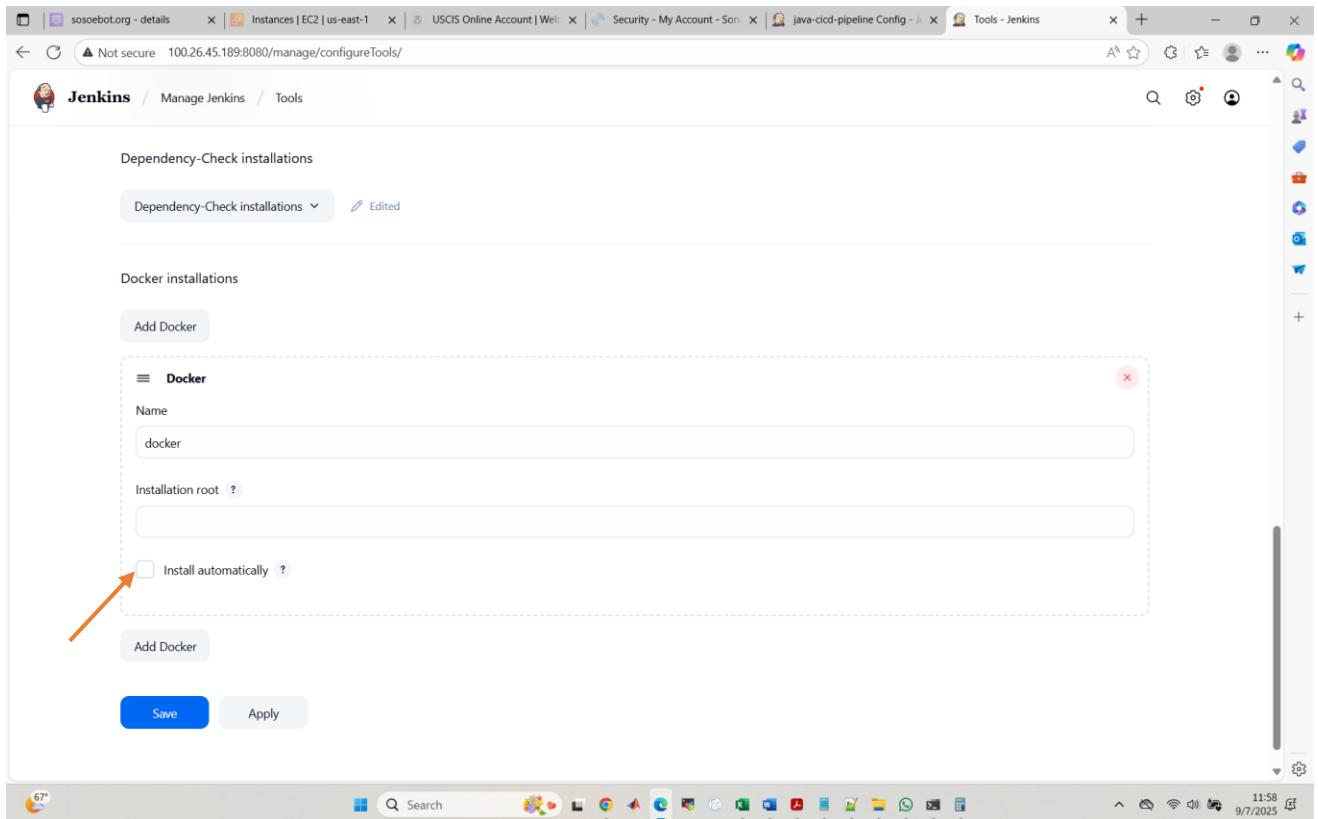
Scroll down to “Docker Installations”

The screenshot shows the Jenkins 'Tools' configuration page with the 'SonarQube Scanner installations' section selected. It includes sections for Ant installations, Maven installations, Dependency-Check installations, and Docker installations. An orange arrow points to the 'Add Docker' button in the Docker installations section. The Jenkins version 'Jenkins 2.516.2' is visible in the bottom right corner of the browser window.

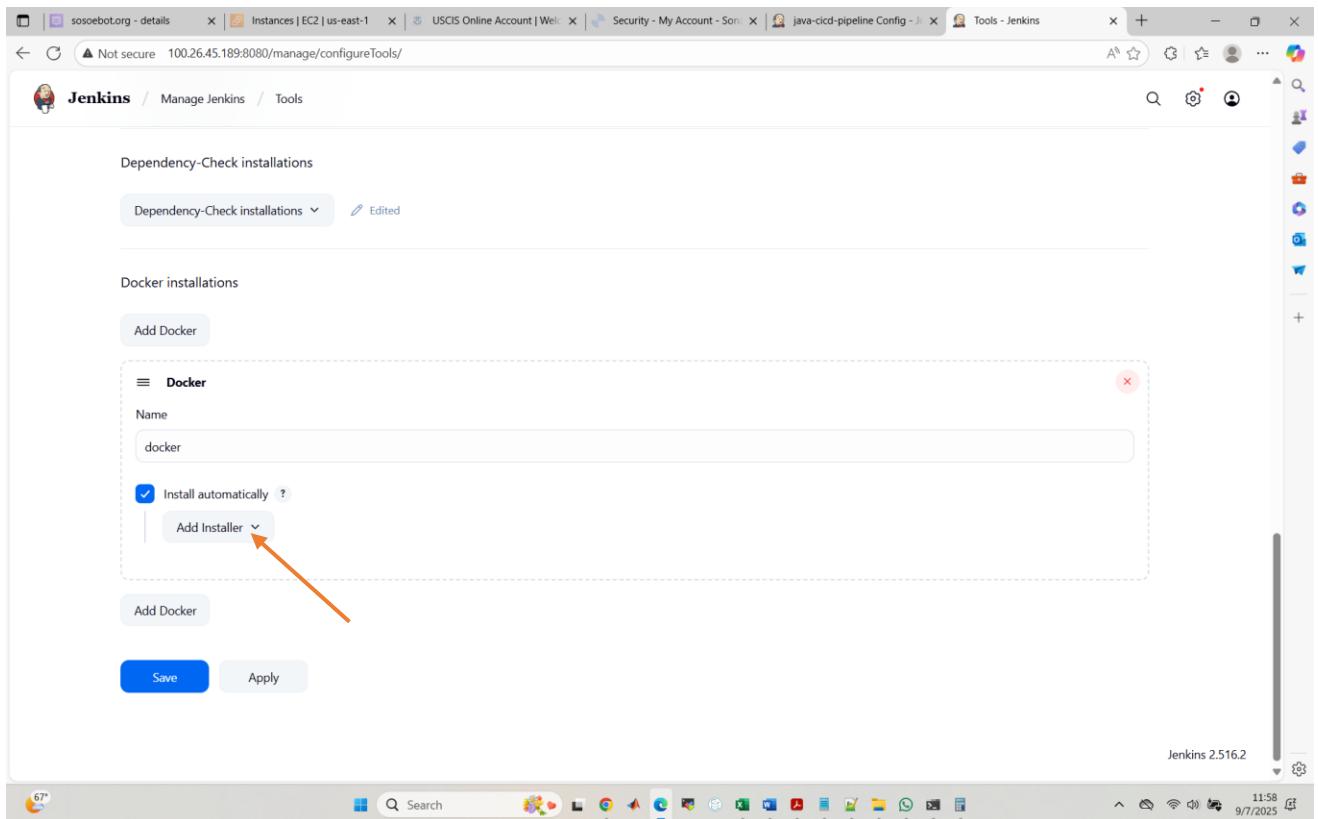
Click on “Add Docker”



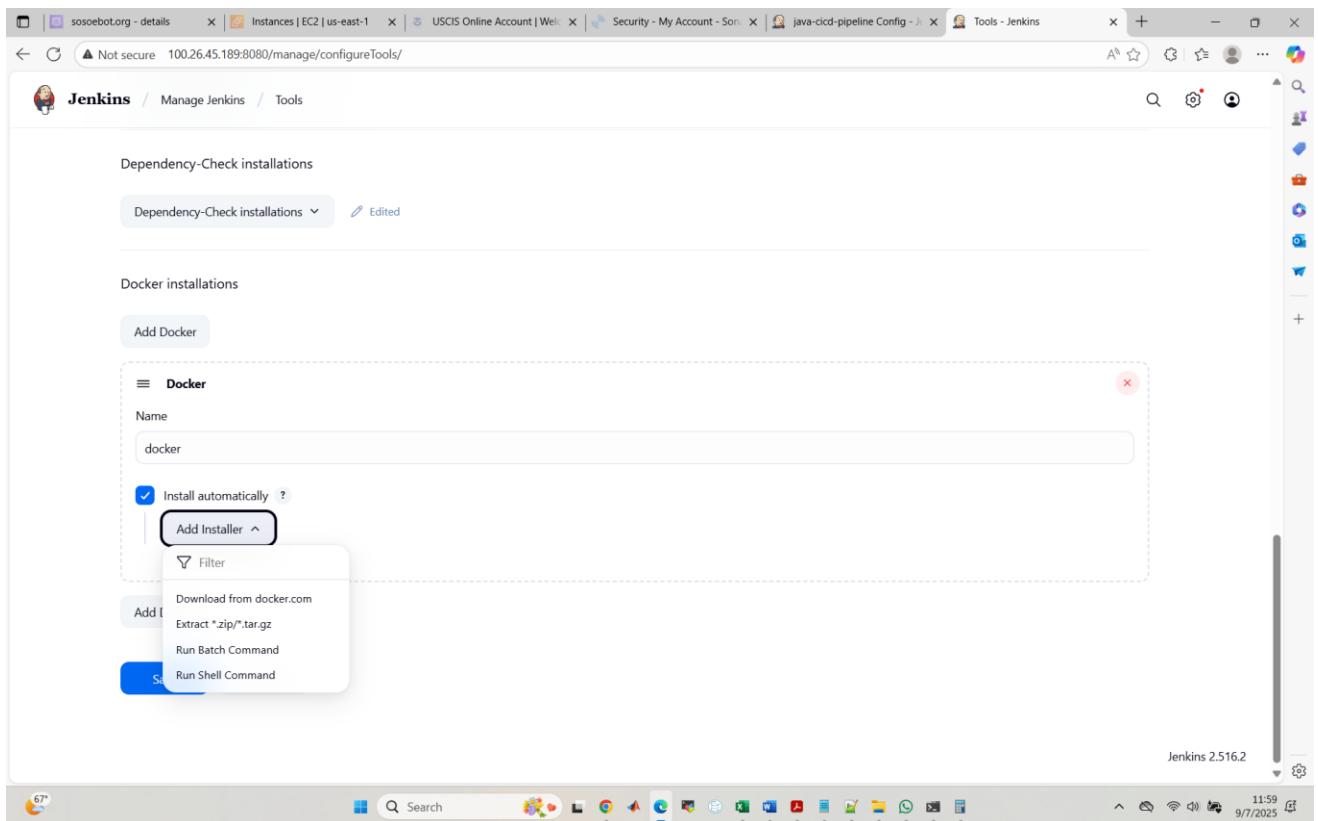
On name, we will enter “**docker**”



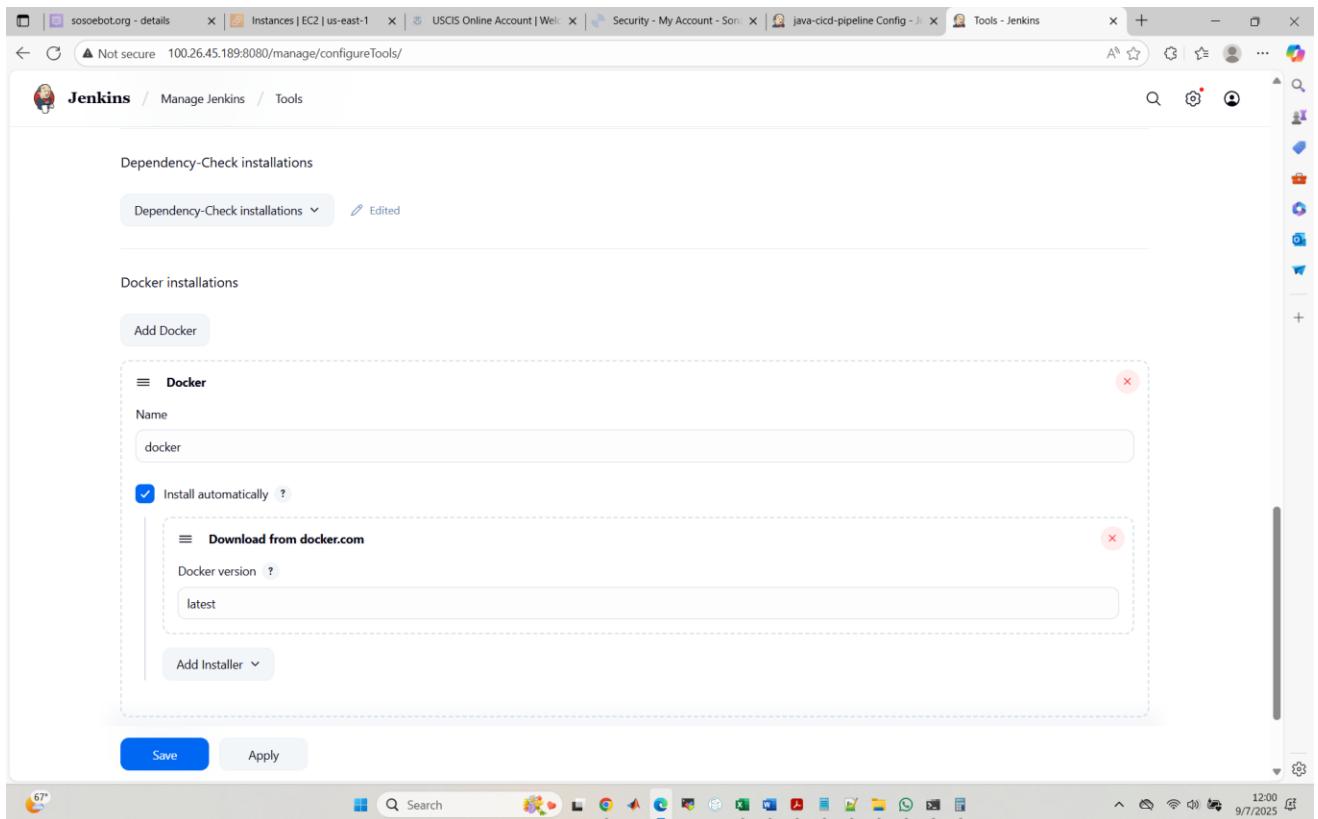
Select “Install Automatically”



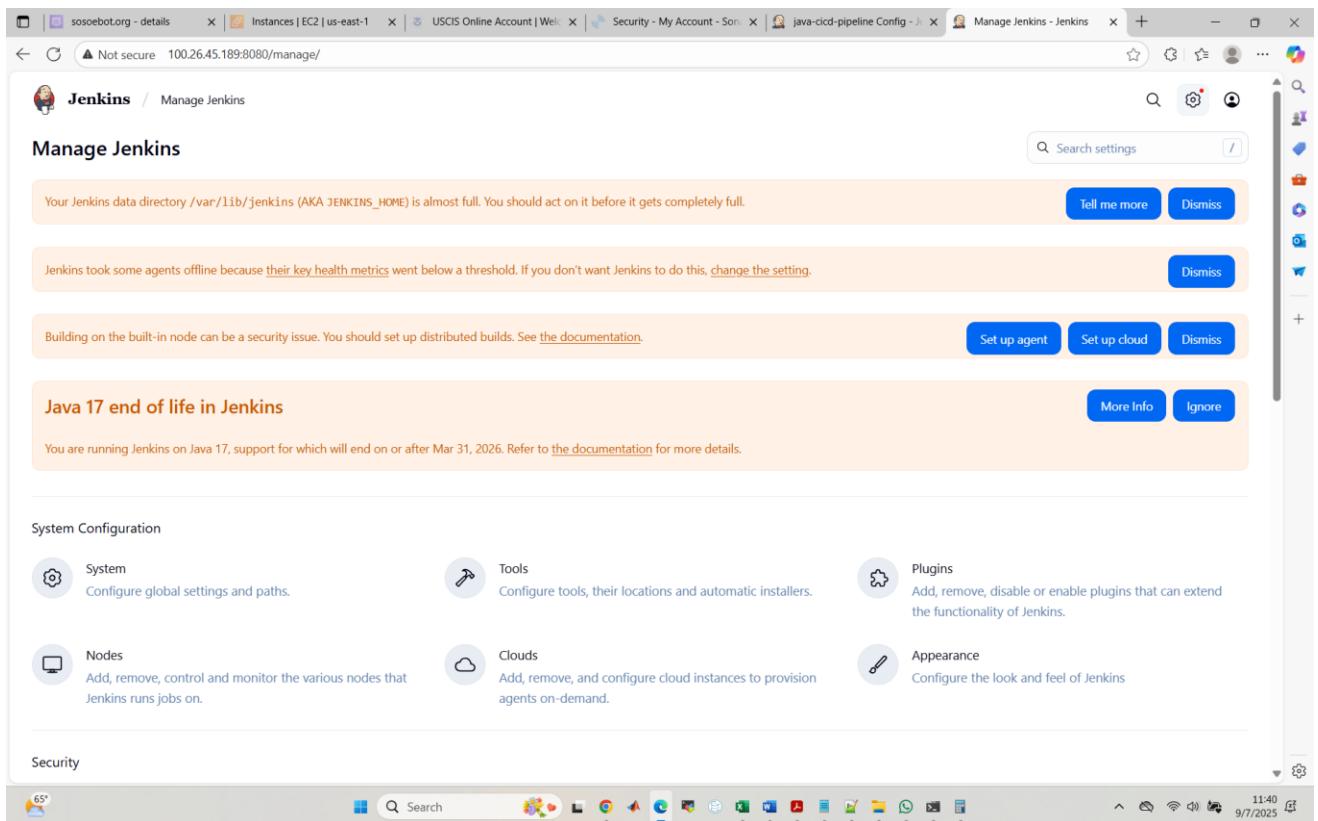
Click on the drop down on “Add Installer”



Select “Download from docker.com” and the Docker version should be “latest”



Click on “Apply” followed by “Save”



Let us install one more plugin called “Pipeline Stage view”. Click on “plugins”

Jenkins / Manage Jenkins / Plugins

Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

No updates available

Disabled rows are already upgraded, awaiting restart. Shaded but selectable rows are in progress or failed.

REST API Jenkins 2.516.2

MIN - CHI Video highlight Search

Then click on “Available Plugins”

Jenkins / Manage Jenkins / Plugins

Available plugins

Search available plugins

Install	Name	Released	Health
<input type="checkbox"/>	Pipeline Graph Analysis 241.vc3d48fb_b_2582	3 mo 10 days ago	100
<input type="checkbox"/>	PAM Authentication 1.12	6 mo 9 days ago	95
<input type="checkbox"/>	JavaMail API 1.6.2-11	6 mo 18 days ago	96
<input type="checkbox"/>	Command Agent Launcher 123.v37cfdc92ef67	5 mo 1 day ago	100
<input type="checkbox"/>	Oracle Java SE Development Kit Installer 83.v417146707a_3d	7 mo 20 days ago	100
<input type="checkbox"/>	SSH server 3.374.v19b_d59ce6610	1 mo 0 days ago	100
<input type="checkbox"/>	Pipeline: REST API 2.38	4 mo 12 days ago	100
<input type="checkbox"/>	Pipeline: Stage View 2.38	4 mo 12 days ago	100

User Interface

MIN - CHI Video highlight Search

Search for “Stage View”

The screenshot shows the Jenkins plugin manager interface. A search bar at the top contains the text "stage view". Below the search bar, a table lists available plugins. The first plugin listed is "Pipeline: Stage View 2.38", which is described as a "User Interface" plugin for Jenkins. It was released 4 months and 12 days ago and has a "Health" status of 100%. An "Install" button is visible next to the plugin name.

Select “Pipeline: Stage View”

The screenshot shows the Jenkins plugin manager interface again, but this time the "Pipeline: Stage View" plugin is selected for installation. The "Install" button is now highlighted in blue, indicating it is the active action. The rest of the interface remains the same, with the search bar containing "stage view" and the table listing the plugin.

Click on “Install”

Jenkins / Manage Jenkins / Plugins

Plugins

Updates Available plugins Installed plugins Advanced settings Download progress

Plugin	Status
Javadoc	Success
JSch dependency	Success
Maven Integration	Success
Loading plugin extensions	Success
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
Pipeline Graph Analysis	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 2.516.2

68°F Sunny

STEP 11: Add Stages to the Pipeline

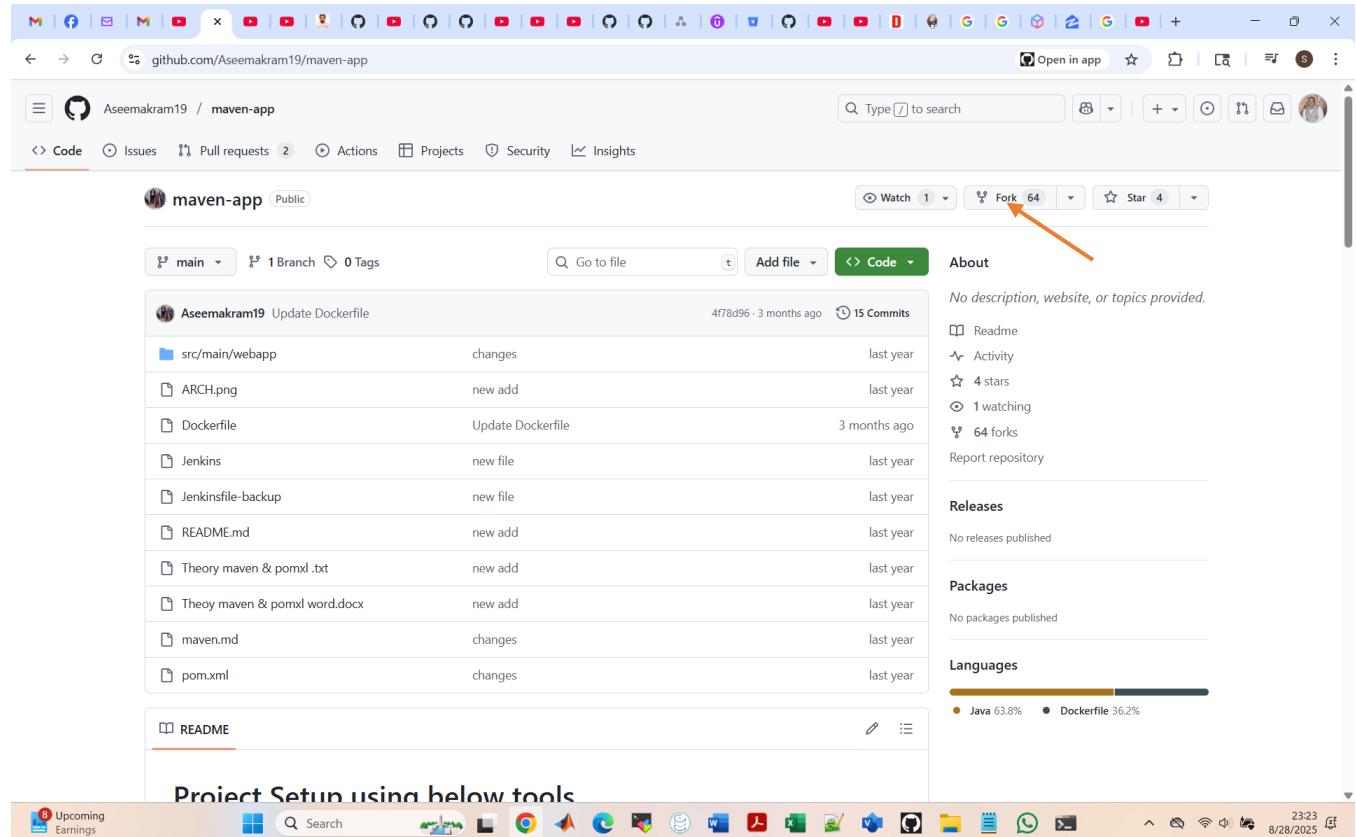
IMPLEMENTATION

Let us now implement our pipeline following our architecture above.

Phase 1: Fork the repository

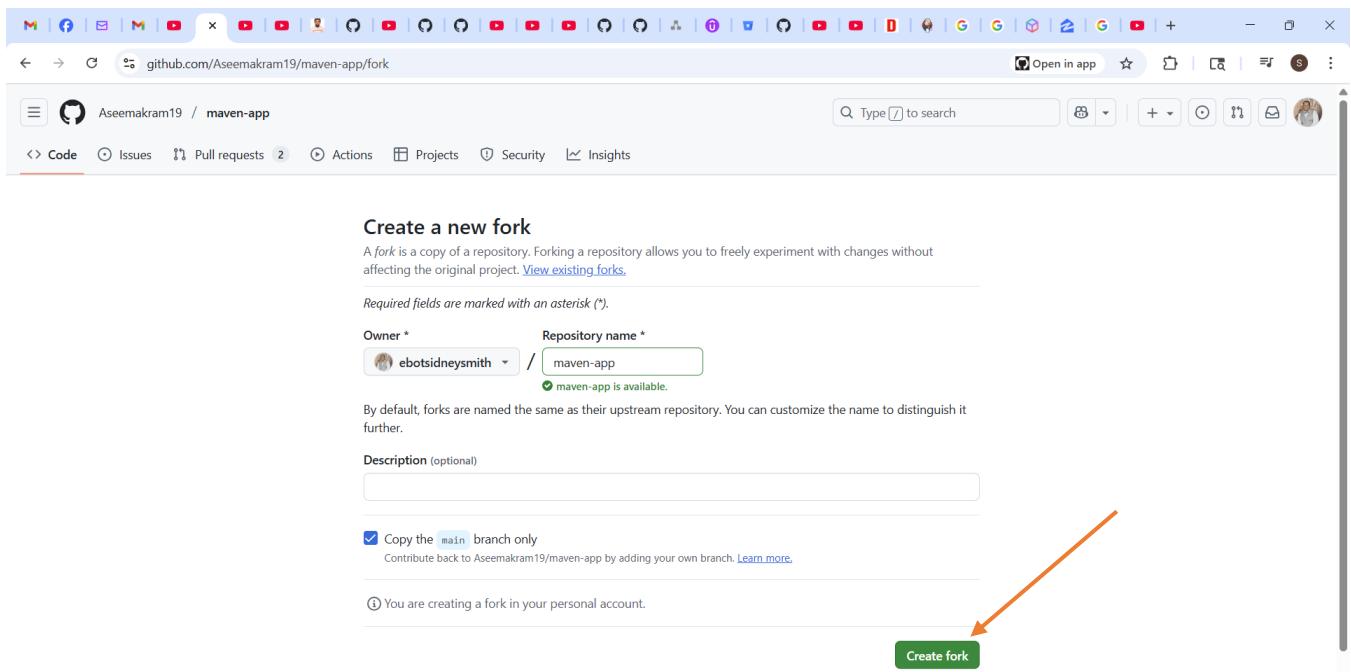
The first stage of our pipeline is to fetch the code from GitHub. Go to the GitHub repository:

<https://github.com/Aseemakram19/maven-app>



A screenshot of a web browser displaying the GitHub repository page for 'maven-app' by 'Aseemakram19'. The repository has 1 branch and 0 tags. The code section shows several commits, including one from 'Aseemakram19' that updated the Dockerfile. The repository has 15 commits in total. The 'About' section indicates no description, website, or topics provided. It shows 4 stars, 1 watching, and 64 forks. The 'Releases' section shows no releases published. The 'Packages' section shows no packages published. The 'Languages' section shows Java at 63.8% and Dockerfile at 36.2%. At the top right, there is a 'Fork' button with the number '64' next to it, which is highlighted with an orange arrow. The browser's address bar shows the URL 'github.com/Aseemakram19/maven-app'.

I will fork this repository by clicking on “Fork”



Click on “Create Fork”

The screenshot shows the forked repository 'ebotsidneysmith/maven-app' on GitHub. It has 1 branch and 0 tags. The 'Code' tab is selected. The repository was forked from 'Aseemkram19/maven-app'. The commit history shows updates to Dockerfile, Jenkinsfile-backup, and README.md. The 'About' section indicates no description, website, or topics provided. The 'Releases' section shows no releases published. The 'Packages' section shows no packages published. The 'Suggested workflows' section includes a 'Clojure' option. The status bar at the bottom shows '59° Partly cloudy' and the date '8/28/2025'.

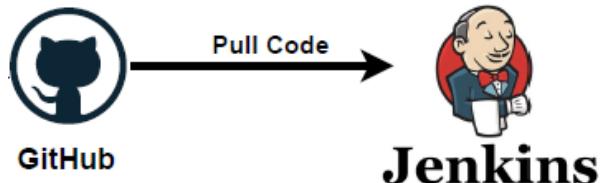
The repository has been forked. Now go to our pipeline code and start modifying it
Head back to Jenkins GUI and click on “Pipeline”

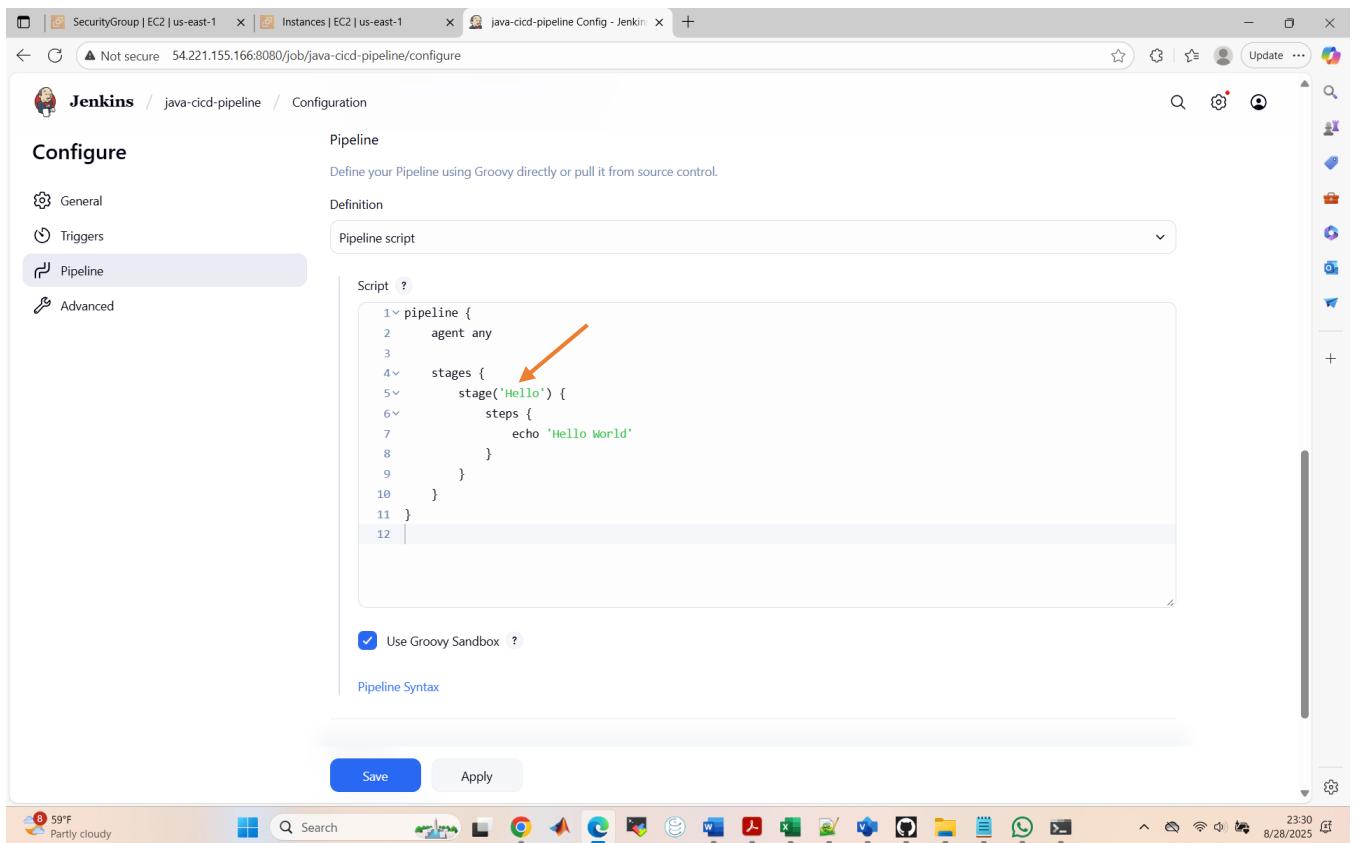
The screenshot shows the Jenkins Pipeline configuration page. The pipeline script is defined as follows:

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

A checkbox labeled "Use Groovy Sandbox" is checked. At the bottom, there are "Save" and "Apply" buttons.

Phase 2: Clone the Repository to Jenkins





Jenkins / java-cicd-pipeline / Configuration

Pipeline

Definition

Pipeline script

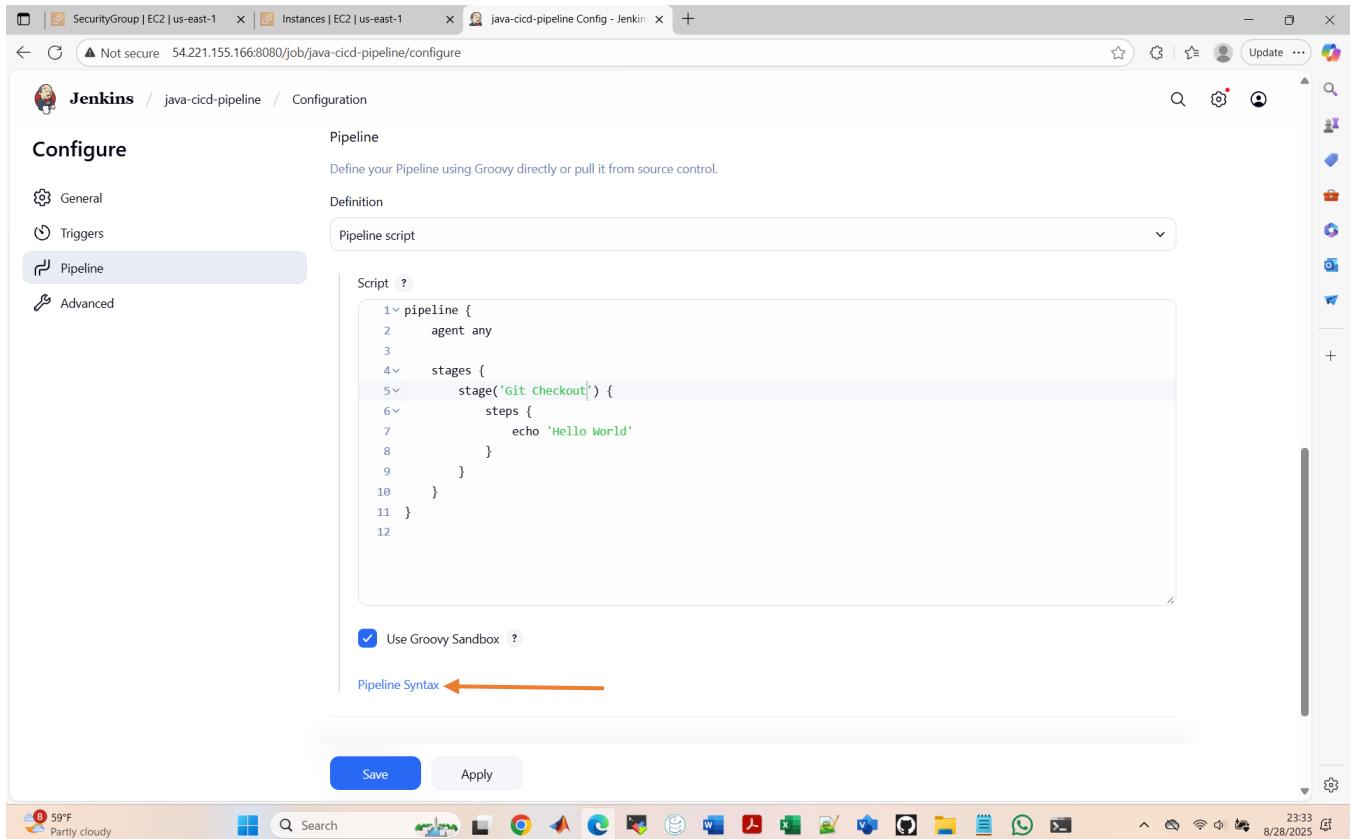
```
Script ?  
1v pipeline {  
2    agent any  
3  
4v   stages {  
5v     stage('Hello') {  
6v       steps {  
7         echo 'Hello World'  
8       }  
9     }  
10   }  
11 }  
12 |
```

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

In the first line on the first stage, change the “Hello” to “Git Checkout”



Jenkins / java-cicd-pipeline / Configuration

Pipeline

Definition

Pipeline script

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

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

Then click on “Pipeline Syntax”, a new window will open

This screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. On the left, a sidebar lists various links: 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 section for 'Steps'. A dropdown menu is open over the 'archiveArtifacts' step, which is described as 'Archive the artifacts'. The dropdown menu has an 'Advanced' button at the bottom. Below the dropdown is a large blue button labeled 'Generate Pipeline Script'. At the bottom of the page, there is a section titled 'Global Variables'.

This screenshot shows the same Jenkins Pipeline Syntax Snippet Generator interface as the previous one, but with a different step selected. The 'git:Git' step is now highlighted with a dark grey background. The dropdown menu for this step is also visible, showing other available steps like 'archiveArtifacts', 'bat', 'build', 'catchError', 'checkout', 'cleanWs', 'deleteDir', 'dir', 'echo', 'emailext', 'emailrecipients', 'error', 'fileExists', 'findBuildScans', 'fingerprint', 'git', 'input', and 'isUnix'. The rest of the interface remains the same, with the sidebar and 'Generate Pipeline Script' button.

Select “git:Git”

This screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. On the left, there's a sidebar with links like '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 sub-section 'Steps'. Under 'Steps', there's a 'Sample Step' dropdown set to 'git: Git'. Below it, there's a configuration form for the 'git' step. The 'Repository URL' field is empty and highlighted with a red arrow. Other fields include 'Branch' (set to 'master'), 'Credentials' (set to '- none -'), and checkboxes for 'Include in polling?' and 'Include in changelog?'. At the bottom is a blue 'Generate Pipeline Script' button.

Copy the URL of the GitHub repository and paste on “Repository URL”

This screenshot shows the same Jenkins Pipeline Syntax Snippet Generator interface as the previous one, but with some changes. The 'Repository URL' field now contains the URL 'https://github.com/ebotsidneysmith/maven-app.git'. The 'Branch' field is also highlighted with a red arrow and set to 'master'. The other fields ('Credentials', 'Include in polling?', and 'Include in changelog?') remain the same as in the first screenshot. The 'Generate Pipeline Script' button is at the bottom.

Change “Master” to “Main” since our code is in the main branch of our repository

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

git: Git

Repository URL ?
https://github.com/ebotsidneysmith/maven-app.git

Branch ?
main

Credentials ?
- none -

+ Add

Include in polling? ?

Include in changelog? ?

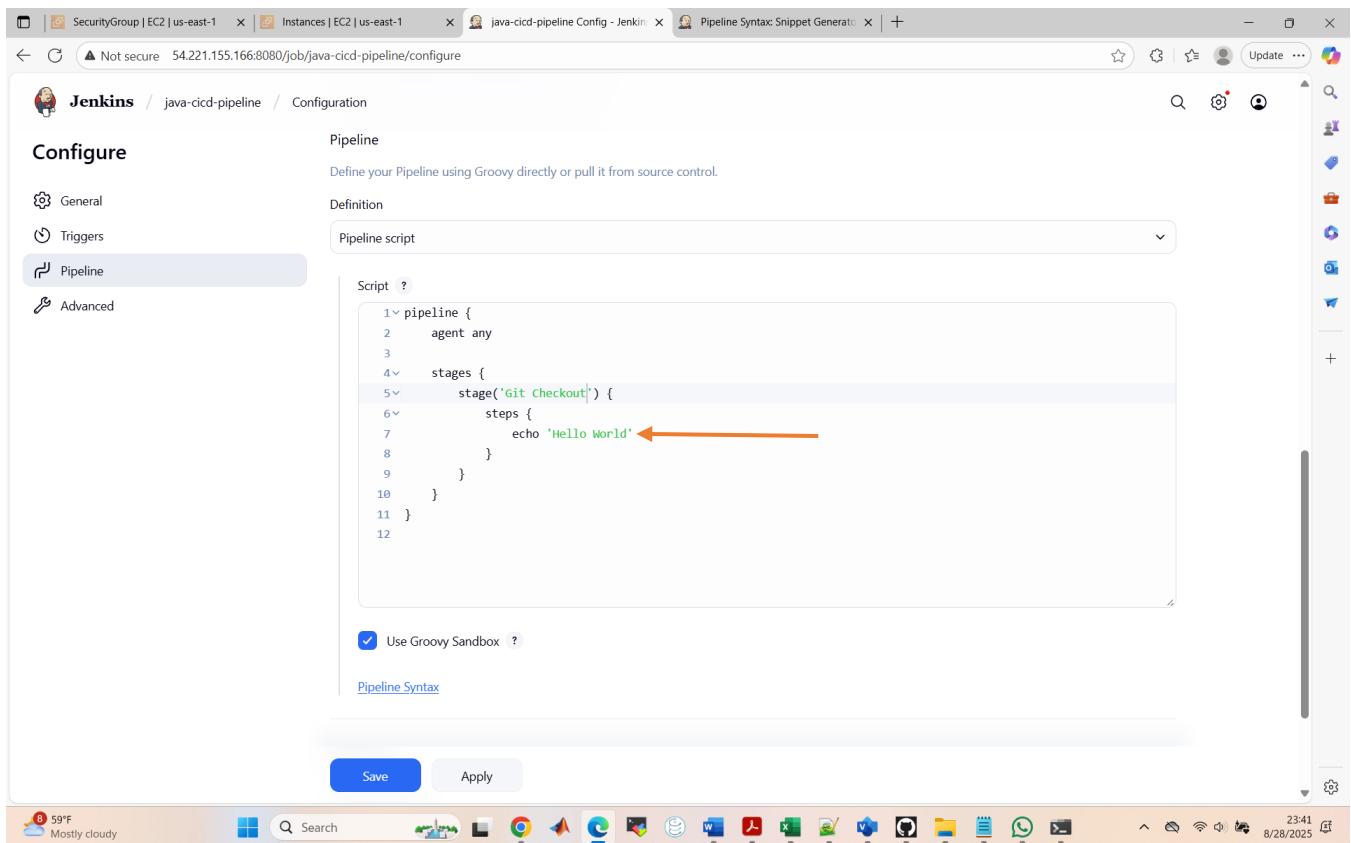
Generate Pipeline Script

Then click on “Generate Pipeline Script”

Generate Pipeline Script

```
git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
```

The script has been generated. Copy the script and go back to our previous Jenkins window



Jenkins / java-cicd-pipeline / Configuration

Pipeline

Definition

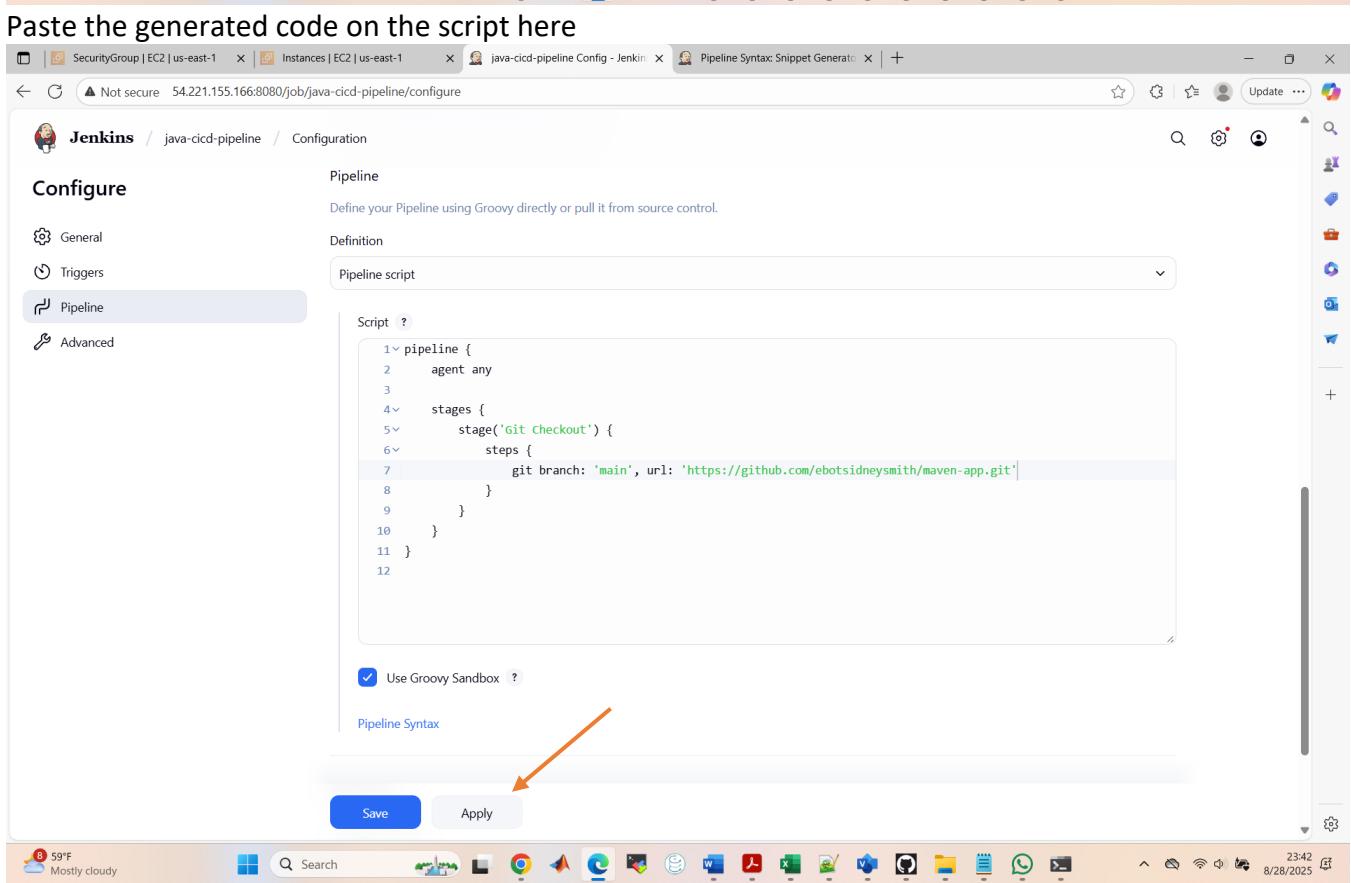
Pipeline script

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

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply



Jenkins / java-cicd-pipeline / Configuration

Pipeline

Definition

Pipeline script

```
1v pipeline {  
2    agent any  
3  
4v   stages {  
5v     stage('Git Checkout') {  
6v       steps {  
7         git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'  
8       }  
9     }  
10   }  
11 }  
12 }
```

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

Click on “Apply”, followed by “Save”

Jenkins / java-cicd-pipeline

java-cicd-pipeline

Java Based CICD Pipeline

Permalinks

- Last build (#1), 1 hr 5 min ago
- Last stable build (#1), 1 hr 5 min ago
- Last successful build (#1), 1 hr 5 min ago
- Last completed build (#1), 1 hr 5 min ago

Builds

Filter

Today

#1 2:38 AM

REST API Jenkins 2.516.2

59°F Mostly cloudy Search 2:43 8/28/2025

Let's test the pipeline. Click on "Build Now"

← C Not secure 54.221.155.166:8080/job/java-cicd-pipeline/ Jenkins / java-cicd-pipeline

java-cicd-pipeline

Java Based CICD Pipeline

Permalinks

- Last build (#1), 1 hr 5 min ago
- Last stable build (#1), 1 hr 5 min ago
- Last successful build (#1), 1 hr 5 min ago
- Last completed build (#1), 1 hr 5 min ago

Builds

Filter

Today

#1 2:38 AM

REST API Jenkins 2.516.2

Refresh the page

The screenshot shows the Jenkins Java CICD Pipeline job page. The pipeline is named "java-cicd-pipeline" and is described as a "Java Based CICD Pipeline". It has four stages: "Git Checkout", "Build", "Test", and "Deploy". The "Git Checkout" stage is currently running, indicated by a green progress bar. The "Build" stage has a green status icon. The "Test" and "Deploy" stages have not yet run. The "Build Now" button is visible on the left sidebar. The "Builds" section shows two completed builds: #2 at 3:44 AM and #1 at 2:38 AM. The Jenkins version is 2.516.2.

Let us build this first stage by clicking on “Build Now” and check the result

The screenshot shows the Jenkins Java CICD Pipeline job page with the "Stage View" section highlighted. The "Git Checkout" stage is shown with a green bar indicating it took 332ms. Below the stage view, the "Builds" section shows three completed builds: #3 at 2:27 PM, #2 at 2:26 PM, and #1 at 2:03 PM. The Jenkins version is 2.516.2.

You can see that the build is successful. Click on the drop down on “#3”

The screenshot shows the Jenkins Java CICD Pipeline status page. The pipeline is named "java-cicd-pipeline". A summary bar indicates a total duration of 332ms. The pipeline consists of two stages: "Git Checkout" and "Build". The "Git Checkout" stage took 332ms. The "Build" stage also took 332ms. Below the summary, there is a "Pipeline Steps" section showing the stages: "Git Checkout", "Build", and "Build". The "Build" stage is expanded, showing the steps: "git", "mvn clean", "mvn test", and "mvn package". The "Build" stage took 45 seconds. The "Build" stage is marked with a green checkmark, indicating success. The "Build" stage has a tooltip: "Build #3: 2:27 PM (45 sec ago)".

Select “Console Output”

The screenshot shows the Jenkins Console Output for build #3. The output log is as follows:

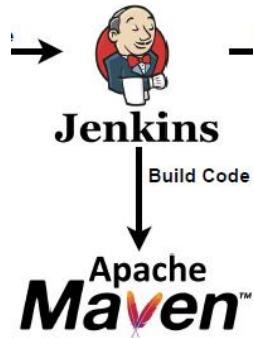
```

Started by user Sidney Ebot
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/java-cicd-pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Git Checkout)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/java-cicd-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/ebotsidneysmith/maven-app.git # timeout=10
Fetching upstream changes from https://github.com/ebotsidneysmith/maven-app.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/ebotsidneysmith/maven-app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 7c550f57df3b49382263d74809414d8251edb936 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 7c550f57df3b49382263d74809414d8251edb936 # timeout=10
> git branch -a -v --no-abbrev # timeout=10
> git branch -D main # timeout=10
> git checkout -b main 7c550f57df3b49382263d74809414d8251edb936 # timeout=10
Commit message: "Update Dockerfile"
> git rev-list --no-walk 7c550f57df3b49382263d74809414d8251edb936 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
```

You can see that the build is successful. Git has cloned the repository. So, we are done with this part of the architecture

Phase 3: Build the code with Maven

The next thing to do is to Build the code using Maven



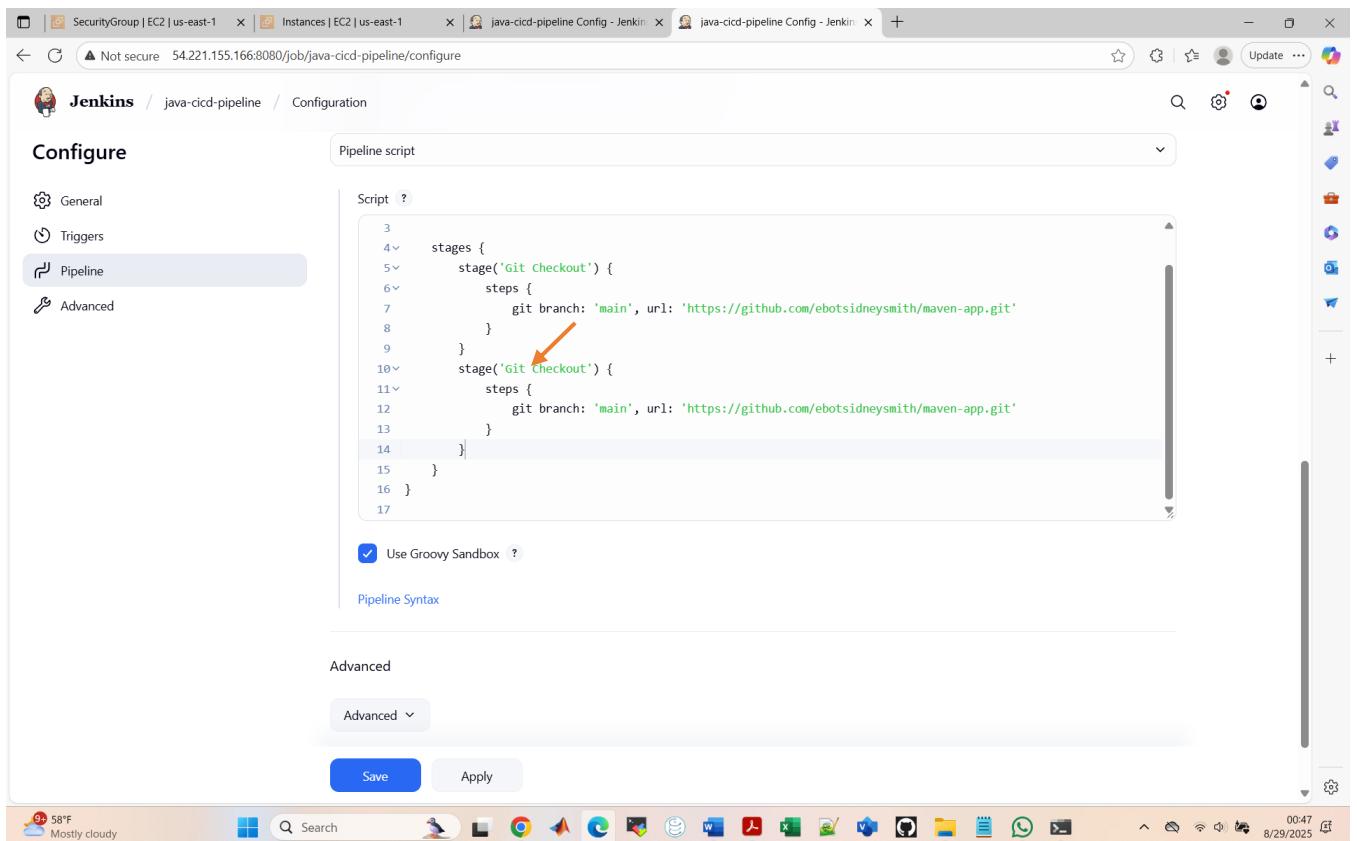
Go back to our pipeline code

Screenshot of the Jenkins Pipeline configuration screen for a job named "java-cicd-pipeline". The left sidebar shows "Configure" and tabs for "General", "Triggers", "Pipeline" (which is selected), and "Advanced". The main area is titled "Pipeline" with the sub-instruction "Define your Pipeline using Groovy directly or pull it from source control.". Below this is a "Definition" section with a "Pipeline script" dropdown set to "Script". The "Script" text area contains the following Groovy code:

```
1< pipeline {
2   agent any
3
4   stages {
5     stage('Git Checkout') {
6       steps {
7         git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
8       }
9     }
10  }
11}
12|
```

Below the script is a checked checkbox for "Use Groovy Sandbox". At the bottom are "Save" and "Apply" buttons.

Copy this stage and paste for the second stage

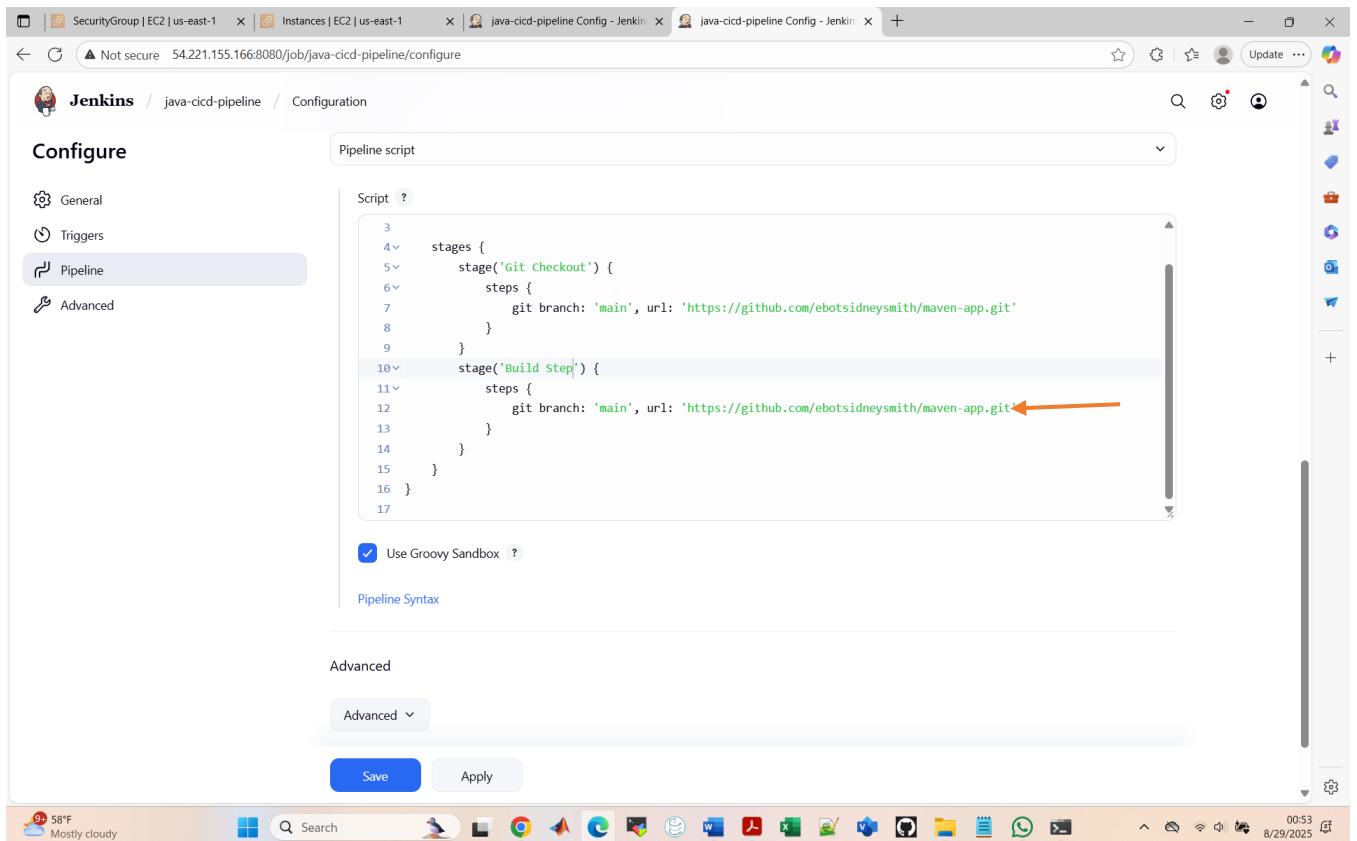


The screenshot shows the Jenkins Pipeline configuration page. The left sidebar has tabs for General, Triggers, Pipeline (which is selected), and Advanced. The main area is titled "Pipeline script" and contains the following Groovy code:

```
3
4  stages {
5    stage('Git Checkout') {
6      steps {
7        git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
8      }
9    }
10   stage('Git Checkout') {
11     steps {
12       git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
13     }
14   }
15 }
```

An orange arrow points to the second "stage('Git Checkout') { ... }" block at line 10.

Now, let us modify the second stage. We will replace “Git Checkout” with “Build Step”



The screenshot shows the Jenkins Pipeline configuration page after modification. The "Pipeline script" section now contains:

```
3
4  stages {
5    stage('Git Checkout') {
6      steps {
7        git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
8      }
9    }
10   stage('Build Step') {
11     steps {
12       git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git' ←
13     }
14   }
15 }
```

An orange arrow points to the "git" step in the second stage at line 12.

Then we modify the second line as follows:

```
sh 'mvn clean package'
```

The screenshot shows the Jenkins Pipeline configuration page. On the left, there's a sidebar with tabs: General, Triggers, Pipeline (which is selected), and Advanced. The main area is titled "Pipeline" and contains the following Groovy script:

```
6    }
7    stages {
8        stage('Git Checkout') {
9            steps {
10                git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
11            }
12        }
13        stage('Build Step') {
14            steps {
15                sh 'mvn clean package'
16            }
17        }
18    }
19 }
```

Below the script, there's a checkbox labeled "Use Groovy Sandbox". At the bottom, there are "Save" and "Apply" buttons.

Then we have to add the tool Maven at the top of the code with name "maven3"

The screenshot shows the Jenkins Pipeline configuration page again. The "Pipeline" section now includes a "tools" block at the top of the script:

```
1 pipeline {
2     agent any
3
4     tools{
5         maven 'maven3'
6     }
7
8     stages {
9         stage('Git Checkout') {
10            steps {
11                git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
12            }
13        }
14        stage('Build Step') {
15            steps {
```

At the bottom, there's a "Pipeline Syntax" link and a "Use Groovy Sandbox" checkbox. The "Save" and "Apply" buttons are also present.

Click on "Apply" followed by "Save"

The screenshot shows the Jenkins Java CICD Pipeline job page. The Stage View section displays a timeline with three stages: 'Git Checkout' (3s), 'Declarative: Tool Install' (148ms), and 'Build Step' (466ms). The 'Build Step' stage is highlighted in green. Below the Stage View is a 'Permalinks' section listing the last four builds. The build history table shows entries for '#2' (Aug 28 23:44) and '#1' (Aug 28 23:44). The Jenkins header indicates the version is 2.516.2.

Then click on “Build Now”

The screenshot shows the Jenkins Java CICD Pipeline job page after clicking 'Build Now'. The build history table now includes a new entry for '#6' (Aug 29 01:11). An orange arrow points to the dropdown menu next to the '#6' entry, which contains options like 'Delete', 'Log', 'Test Results', and 'View Log'.

Click on the drop down on “#6”

The screenshot shows the Jenkins Java CICD Pipeline job page. On the left, there's a sidebar with various Jenkins management links like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. Below that is a list of builds from today, starting with #8 at 4:22 PM. The main content area is titled "java-cicd-pipeline" and describes it as a "Java Based CICD Pipeline". It shows a summary table of build steps:

	Declarative: Tool Install	Git Checkout	Build Step
1	180ms	562ms	3s
2	153ms	510ms	3s
3	240ms	710ms	4s
4	148ms	466ms	3s

Below the table is a section titled "Permalinks" with a list of recent builds:

- Last build (#7), 10 hr ago
- Last stable build (#7), 10 hr ago
- Last successful build (#7), 10 hr ago
- Last completed build (#7), 10 hr ago

At the bottom of the main content area, there's a link to "54.221.155.166:8080/job/java-cicd-pipeline/8/console".

Select “Console Output”

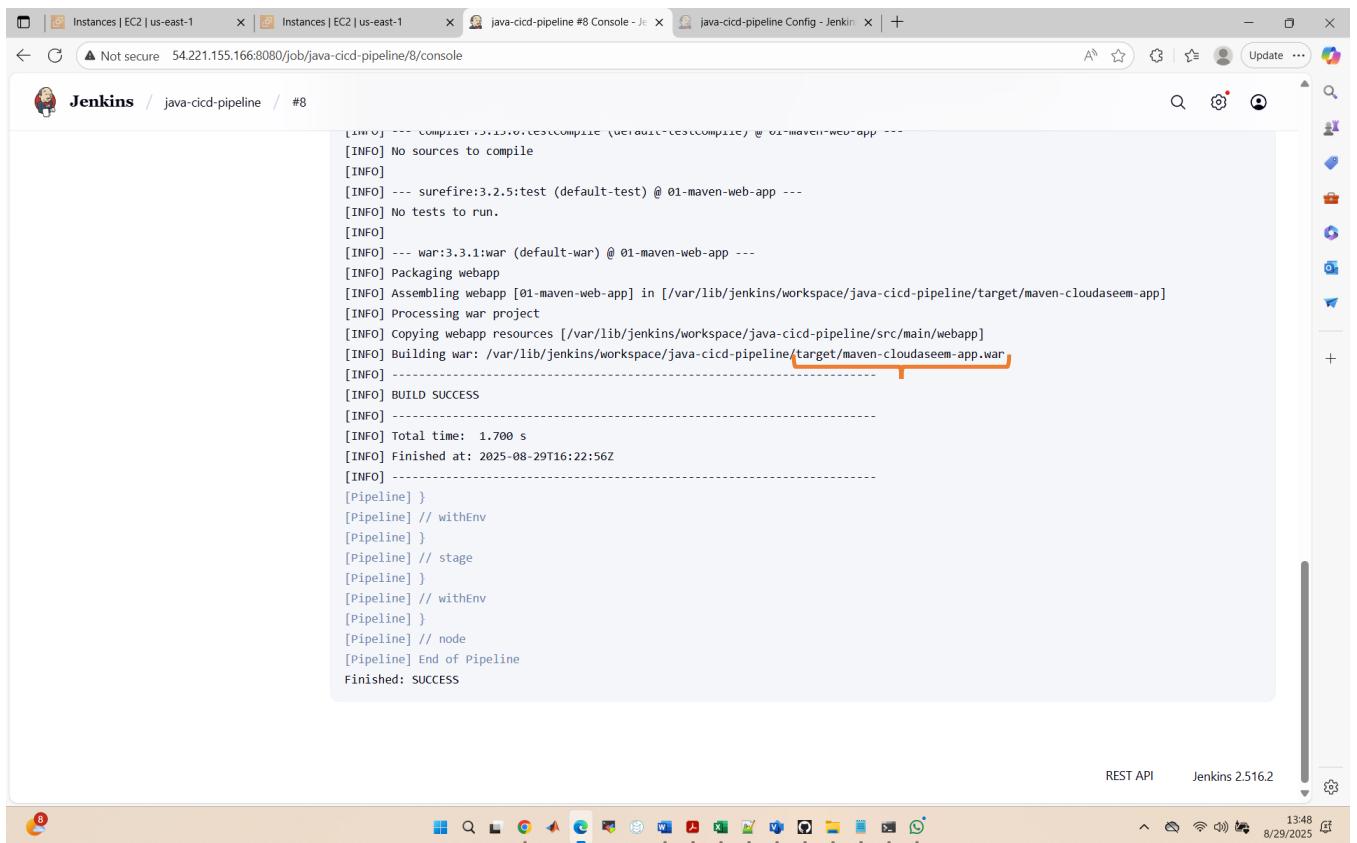
The screenshot shows the Jenkins Console Output page for build #8. The sidebar on the left is identical to the previous screenshot. The main content area is titled "Console Output" and shows the log output for build #8:

```

Started by user Sidney Ebot
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/java-cicd-pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Tool Install)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Git Checkout)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/java-cicd-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/ebotsidneysmith/maven-app.git # timeout=10
Fetching upstream changes from https://github.com/ebotsidneysmith/maven-app.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/ebotsidneysmith/maven-app.git +refs/heads/*:refs/remotes/origin/*
timeout=10

```

Scroll down to the end



```
[INFO] --- compiler:3.8.0:compile (default-compile) @ 01-maven-web-app ---
[INFO] No sources to compile
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ 01-maven-web-app ---
[INFO] No tests to run.
[INFO]
[INFO] --- war:3.3.1:war (default-war) @ 01-maven-web-app ---
[INFO] Packaging webapp
[INFO] Assembling webapp [01-maven-web-app] in [/var/lib/jenkins/workspace/java-cicd-pipeline/target/maven-cloudaseem-app]
[INFO] Processing war project
[INFO] Copying webapp resources [/var/lib/jenkins/workspace/java-cicd-pipeline/src/main/webapp]
[INFO] Building war: /var/lib/jenkins/workspace/java-cicd-pipeline/target/maven-cloudaseem-app.war
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.700 s
[INFO] Finished at: 2025-08-29T16:22:56Z
[INFO] -----
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Copy this part of the code, we will use it later
target/maven-cloudaseem-app.war

So, we have completed this step. We now go to the next step

Phase 4: Create Docker Image

In this step, we have to build the docker image. We will start by opening our Dockerfile and modify it.

The screenshot shows a GitHub repository page for 'ebotsidneysmith / maven-app'. The 'Dockerfile' tab is selected in the sidebar. The code editor displays the following Dockerfile:

```
1 # Use a specific version of Tomcat as base image
2 FROM tomcat:9.0
3
4 # Expose port 8080 to access the application
5 EXPOSE 8080
6
7 # Copy the WAR file from the target directory of your Maven project to the Tomcat webapps directory
8 COPY target/maven-cloudaseem-app.war /usr/local/tomcat/webapps/
```

A red box highlights the line 'COPY target/maven-cloudaseem-app.war /usr/local/tomcat/webapps/'. The GitHub interface includes a search bar, a blame history button, and a raw code link.

We have to modify **line 8** of our Dockerfile code. We will replace the part of the code show above with the part of the code we copied previously.

The screenshot shows the same GitHub repository page for 'ebotsidneysmith / maven-app'. The 'Dockerfile' tab is selected. A red arrow points to the 'Edit' icon in the top right corner of the code editor's toolbar. The code remains the same as in the previous screenshot.

To do this click on “Edit”

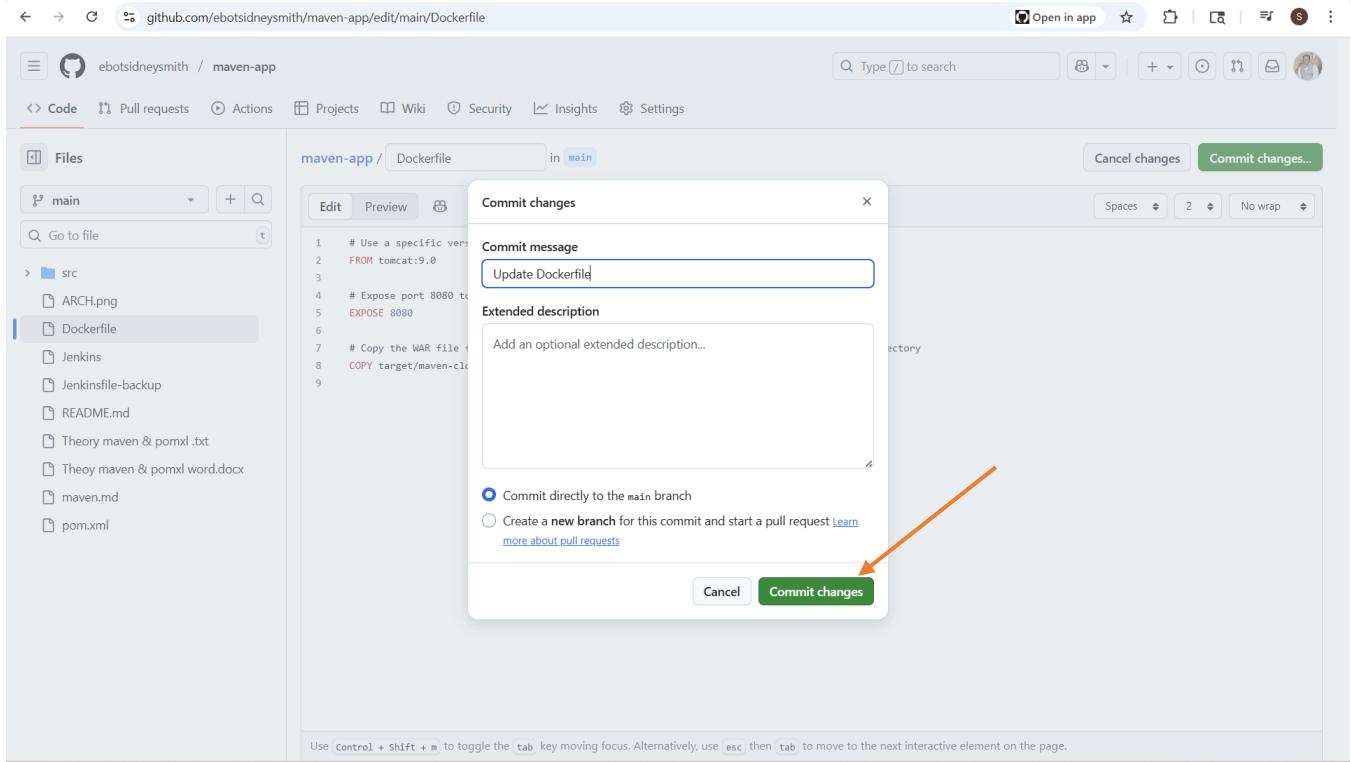
A screenshot of a GitHub code editor interface. The URL in the address bar is github.com/ebotsidneysmith/maven-app/edit/main/Dockerfile. The repository name is "maven-app". The file being edited is "Dockerfile" in the "main" branch. The code content is:

```
1 # Use a specific version of Tomcat as base image
2 FROM tomcat:9.0
3
4 # Expose port 8080 to access the application
5 EXPOSE 8080
6
7 # Copy the WAR file from the target directory of your Maven project to the Tomcat webapps directory
8 COPY target/maven-cloudaseem-app.war /usr/local/tomcat/webapps/
```

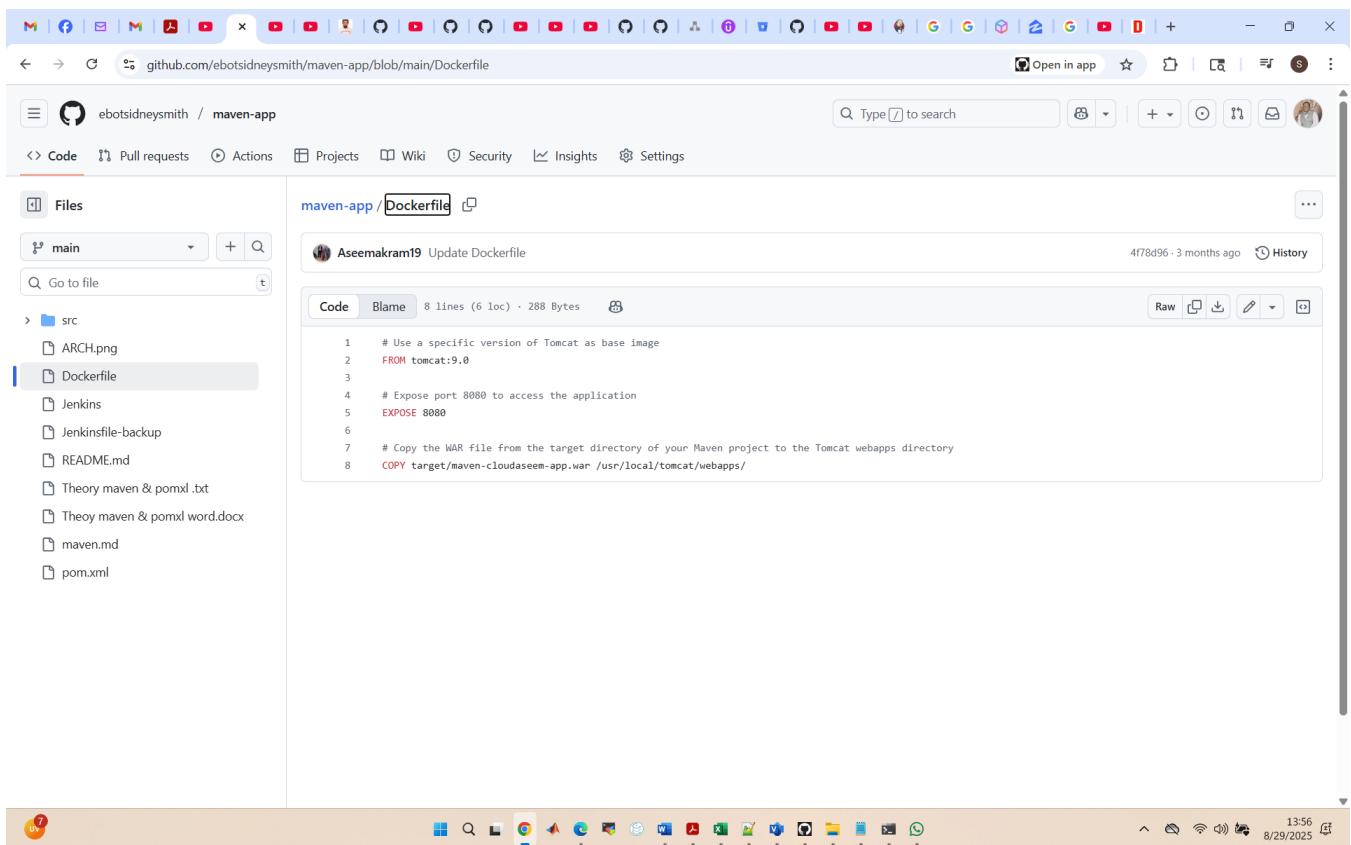
Paste the code here

A screenshot of the same GitHub code editor interface. The code has been pasted into the editor. A red arrow points to the green "Commit changes..." button in the top right corner of the editor window.

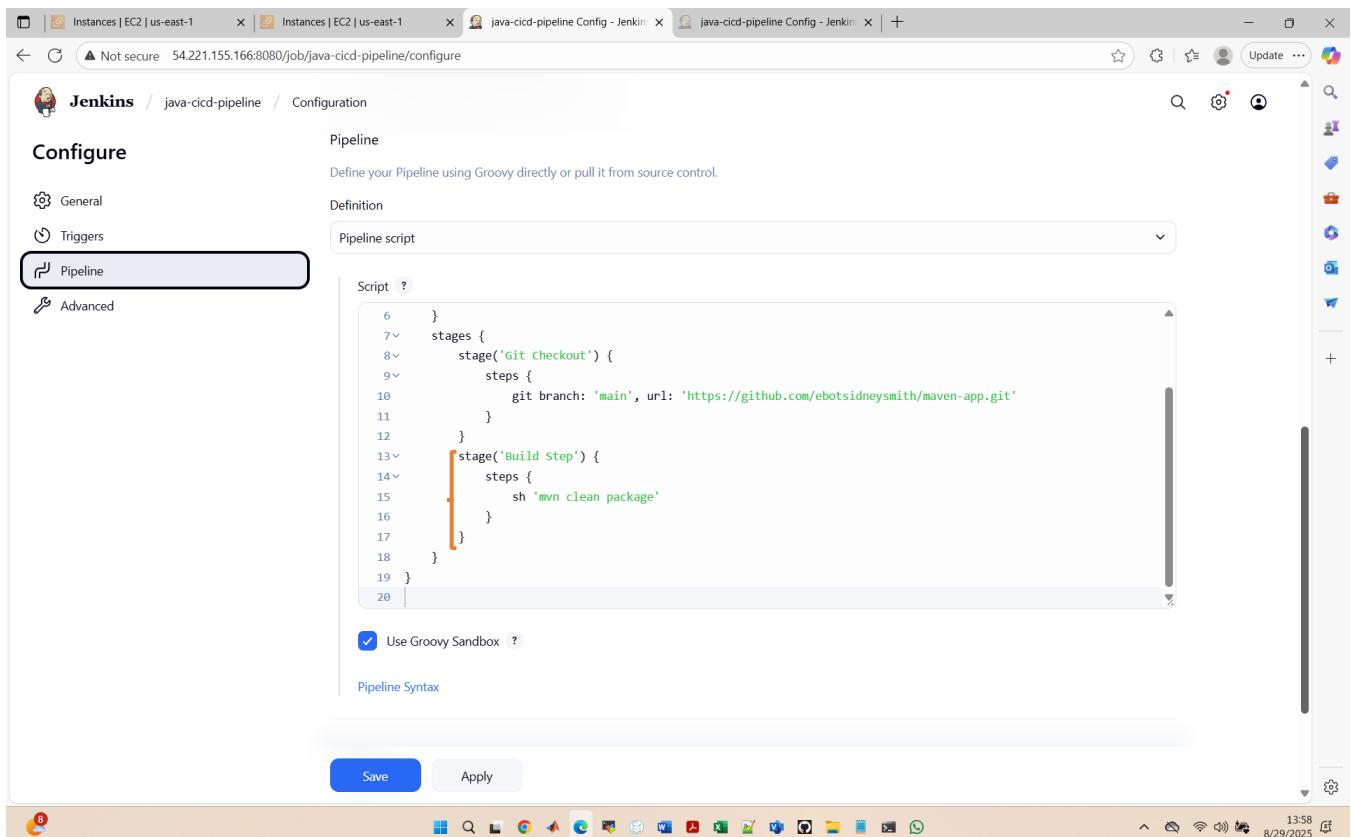
Then click on “Commit Changes”



Confirm by clicking on “Commit Changes” again



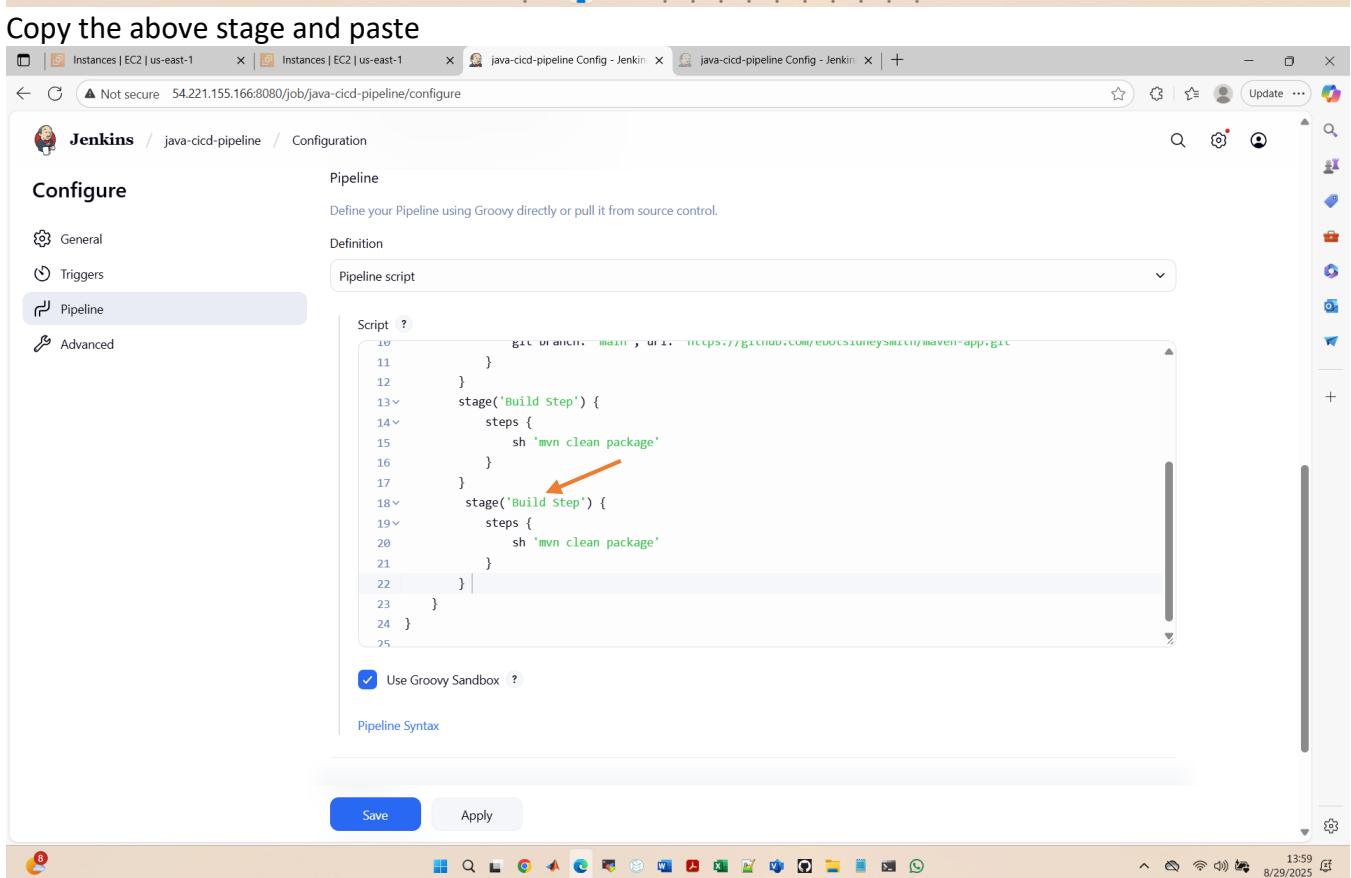
Now, let us go back to our pipeline and add one more stage



The screenshot shows the Jenkins Pipeline configuration page. The 'Pipeline' tab is selected. The pipeline script is as follows:

```
6    }
7    stages {
8        stage('Git Checkout') {
9            steps {
10                git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
11            }
12        }
13        stage('Build Step') {
14            steps {
15                sh 'mvn clean package'
16            }
17        }
18    }
19 }
```

A red arrow points to the second 'stage' block. Below the script, there is a checkbox for 'Use Groovy Sandbox'. At the bottom are 'Save' and 'Apply' buttons.

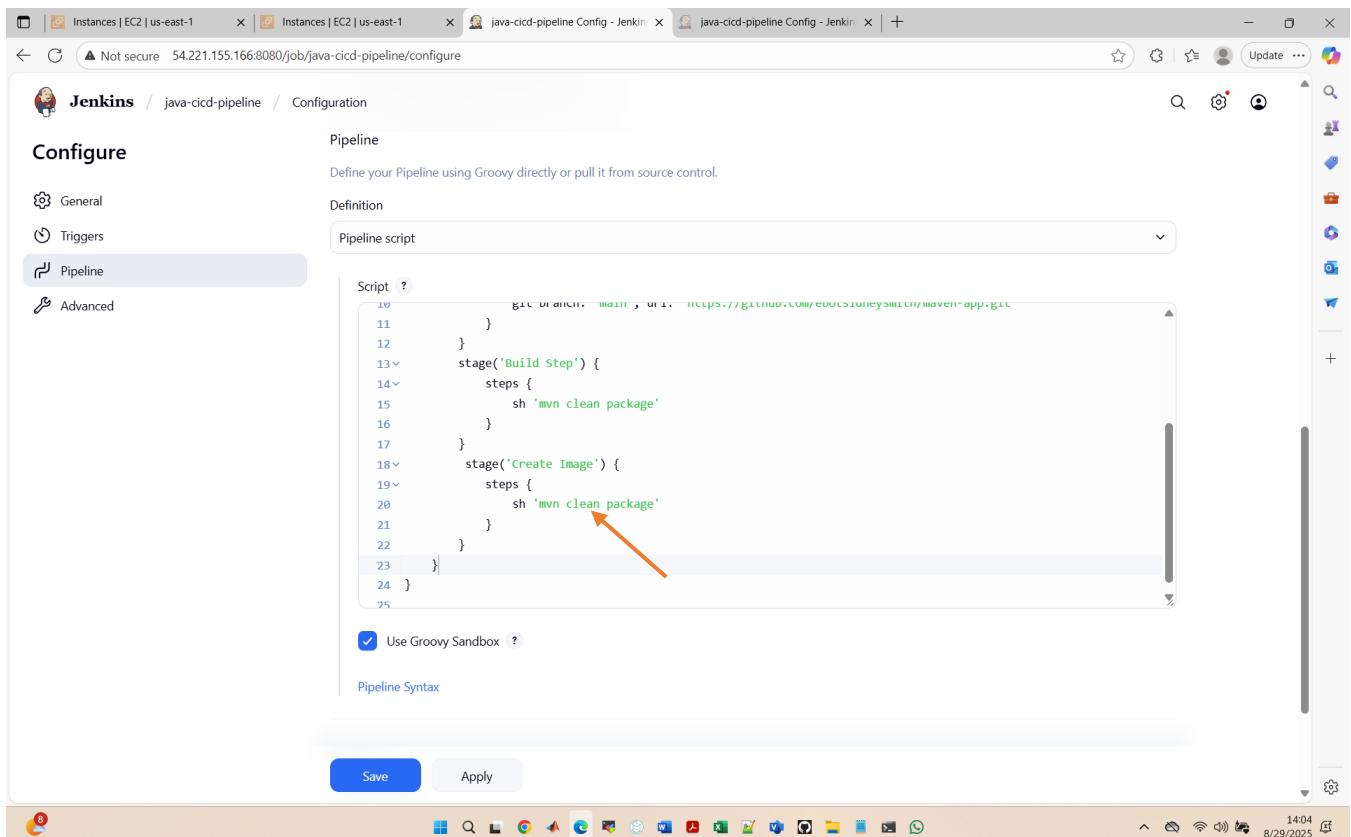


The screenshot shows the Jenkins Pipeline configuration page. The 'Pipeline' tab is selected. The pipeline script now includes an additional stage:

```
10    }
11    stages {
12        stage('Git Checkout') {
13            steps {
14                git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
15            }
16        }
17        stage('Build Step') {
18            steps {
19                sh 'mvn clean package'
20            }
21        }
22    }
23 }
24 }
```

A red arrow points to the second 'stage' block. Below the script, there is a checkbox for 'Use Groovy Sandbox'. At the bottom are 'Save' and 'Apply' buttons.

Let us now modify the added stage. We will change “**Build Step**” on this added stage with “**Create Image**”. Take note that you can decide to use any name for the stage.

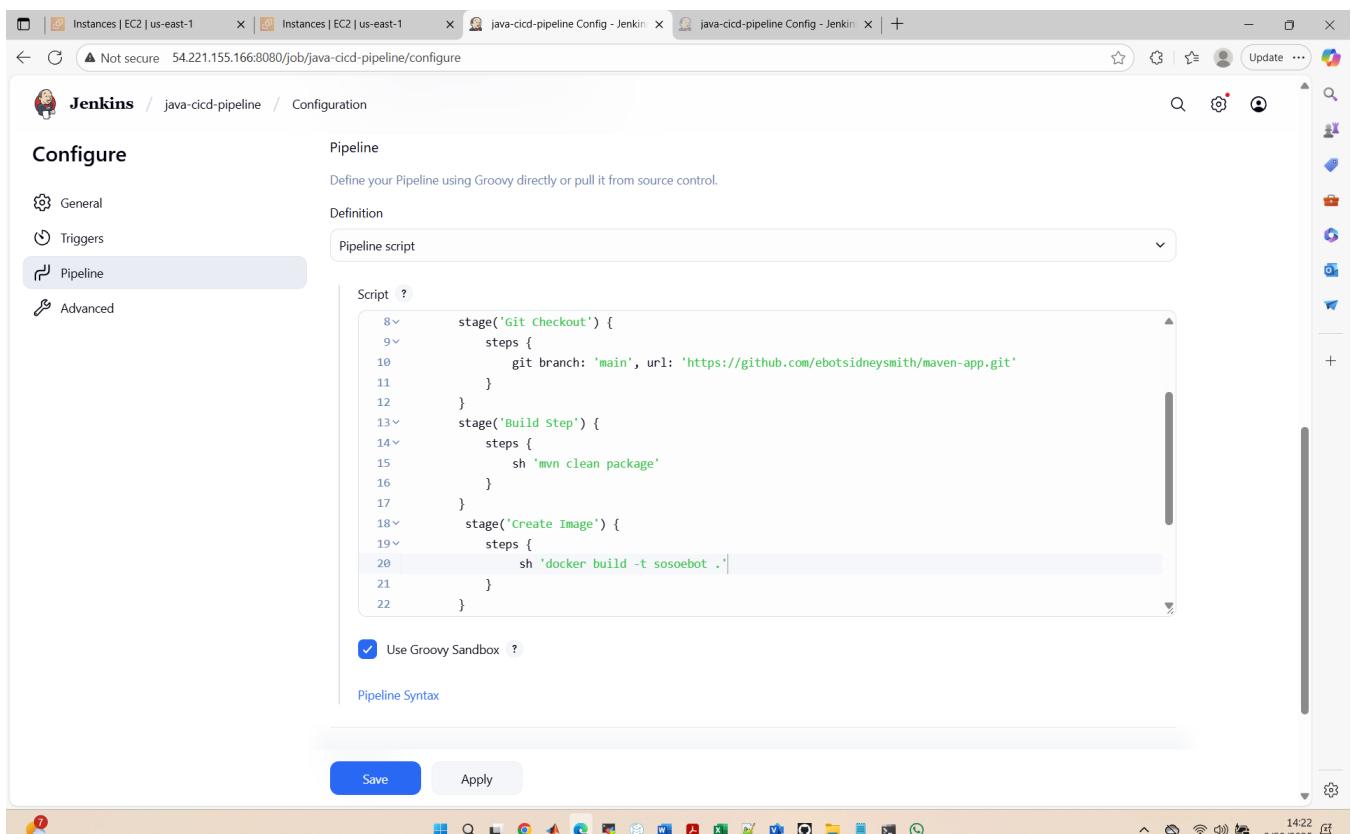


The screenshot shows the Jenkins Pipeline configuration page. The left sidebar has tabs for General, Triggers, Pipeline (which is selected), and Advanced. The main area is titled 'Pipeline' with the sub-instruction 'Define your Pipeline using Groovy directly or pull it from source control.' Below this is a 'Definition' section with a 'Pipeline script' dropdown set to 'Script'. The script editor contains the following Groovy code:

```
10      git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
11    }
12  }
13  stage('Build Step') {
14    steps {
15      sh 'mvn clean package'
16    }
17  }
18  stage('Create Image') {
19    steps {
20      sh 'mvn clean package'
21    }
22  }
23 }
24 }
```

An orange arrow points to the line 'sh 'mvn clean package'' in the 'Create Image' stage. At the bottom of the script editor are 'Save' and 'Apply' buttons.

Then, let us modify the step. We will replace “`sh 'mvn clean package'`” with “`sh 'docker build -t sosoebot .'`”. Where “`sosoebot`” is our image name.



The screenshot shows the Jenkins Pipeline configuration page with the Groovy script modified. The 'Pipeline' stage now contains:

```
8  stage('Git Checkout') {
9    steps {
10      git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
11    }
12  }
13  stage('Build Step') {
14    steps {
15      sh 'mvn clean package'
16    }
17  }
18  stage('Create Image') {
19    steps {
20      sh 'docker build -t sosoebot .'
21    }
22  }
```

The 'Pipeline Syntax' section at the bottom is visible. At the bottom of the page are 'Save' and 'Apply' buttons.

Now, let us add another stage in our pipeline code to containerize our application

Jenkins / java-cicd-pipeline / Configuration

Configure

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

```

16
17
18 } stage('Create Image') {
19 steps {
20     sh 'docker build -t sosoebot .'
21 }
22 }
23 stage('Create Image') {
24 steps {
25     sh 'docker build -t sosoebot .'
26 }
27 }
28 }
29 }
```

Use Groovy Sandbox ?

Pipeline Syntax

Save **Apply**

Click on “Apply” then “Save”

sosoebot.org - details Instances | EC2 | us-east-1 Prometheus Time Series Col... prometheus - Data sources java-cicd-pipeline - Jenkins Pipeline Syntax: Snippet Gen...

Jenkins / java-cicd-pipeline

Status **java-cicd-pipeline** Java Based CICD Pipeline

Changes Build Now Configure Delete Pipeline Full Stage View Stages Rename Pipeline Syntax

Stage View

Declarative: Tool Install	Git Checkout	Build Step
489ms	466ms	5s
#6 Sep 09 10:44 No Changes	489ms	466ms
		5s

Average stage times: (full run time: ~7s)

Builds

- #6 2:44PM
- #3 2:27PM
- #2 2:26PM
- #1 2:03PM

Permalinks

- Last build (#6), 4 min 29 sec ago
- Last stable build (#6), 4 min 29 sec ago
- Last successful build (#6), 4 min 29 sec ago
- Last completed build (#6), 4 min 29 sec ago

REST API Jenkins 2.516.2

Click on “Build Now”

Screenshot of the Jenkins Java CICD Pipeline job details page.

Job Information:

- Status: **java-cicd-pipeline** (green checkmark)
- Type: Java Based CICD Pipeline
- Last Build: #7 (Sep 09 10:49) - No Changes
- Build Time: 133ms
- Git Checkout: 465ms
- Build Step: 4s
- Create Image: 10s
- Average stage times: (full run time: ~11s)

Build History:

- #6 2:44PM
- #3 2:27PM
- #2 2:26PM
- #1 2:03PM

Permalinks:

- Last build (#6), 4 min 29 sec ago
- Last stable build (#6), 4 min 29 sec ago
- Last successful build (#6), 4 min 29 sec ago
- Last completed build (#6), 4 min 29 sec ago

REST API Jenkins 2.516.2

The build is successful. Let us add the next stage We will call the stage “Create Container”.

Screenshot of the Jenkins Java CICD Pipeline configuration page.

Configuration:

- General
- Triggers
- Pipeline** (selected)
- Advanced

Pipeline:

Define your Pipeline using Groovy directly or pull it from source control.

Definition: Pipeline script

```

17      }
18    }
19    stage('Create Image') {
20      steps {
21        sh 'docker build -t sosoebot .'
22      }
23    }
24    stage('Create Image') {
25      steps {
26        sh 'docker build -t sosoebot .'
27      }
28    }
29  }
30 }
31

```

An orange arrow points to the second 'Create Image' stage in the Groovy script.

Options:

- Use Groovy Sandbox ?

Buttons:

- Save
- Apply

REST API Jenkins 2.516.2

So, we will change “Create Image” to “Create Container”

The screenshot shows the Jenkins Pipeline configuration page. The pipeline script is defined as follows:

```
17 }
18 }
19 stage('Create Image') {
20   steps {
21     sh 'docker build -t sosoebot .'
22   }
23 }
24 stage('Create Container') {
25   steps {
26     sh 'docker run -d -p 9000:8080 --name sosoebot_container -t sosoebot'
27   }
28 }
29 }
30 }
```

An orange arrow points to the line `sh 'docker run -d -p 9000:8080 --name sosoebot_container -t sosoebot'`. Below the script, there are "Save" and "Apply" buttons.

Then, we will modify “`sh ‘docker build -t sosoebot’`” to “`sh ‘docker run -d -p 9000:8080 --name sosoebot_container -t sosoebot’`”. Where the image name is “`sosoebot`” and the container name is “`sosoebot_container`”.

The screenshot shows the Jenkins Pipeline configuration page with the modified script:

```
17 }
18 }
19 stage('Create Image') {
20   steps {
21     sh 'docker build -t sosoebot .'
22   }
23 }
24 stage('Create Container') {
25   steps {
26     sh 'docker run -d -p 9000:8080 --name sosoebot_container -t sosoebot'
27   }
28 }
29 }
30 }
```

The line `sh 'docker run -d -p 9000:8080 --name sosoebot_container -t sosoebot'` has been updated. Below the script, there are "Save" and "Apply" buttons.

Click on “**Apply**” followed by “**Save**”

The screenshot shows the Jenkins Java CICD Pipeline stage view. On the left, there's a sidebar with options like Status, Changes, Build Now (which has an orange arrow pointing to it), Configure, Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. Below that is a Builds section with a search bar. The main area shows a Stage View table with four columns: Declarative: Tool Install, Git Checkout, Build Step, and Create Image. The table has two rows of data. Row 1 (Build #7) shows times of 311ms, 465ms, 4s, and 10s respectively. Row 2 (Build #6) shows 133ms, 465ms, 3s, and 10s. Below the table is a Permalinks section with a bulleted list of recent builds. The bottom right corner shows Jenkins 2.516.2.

Then build the pipeline by clicking on “Build Now”

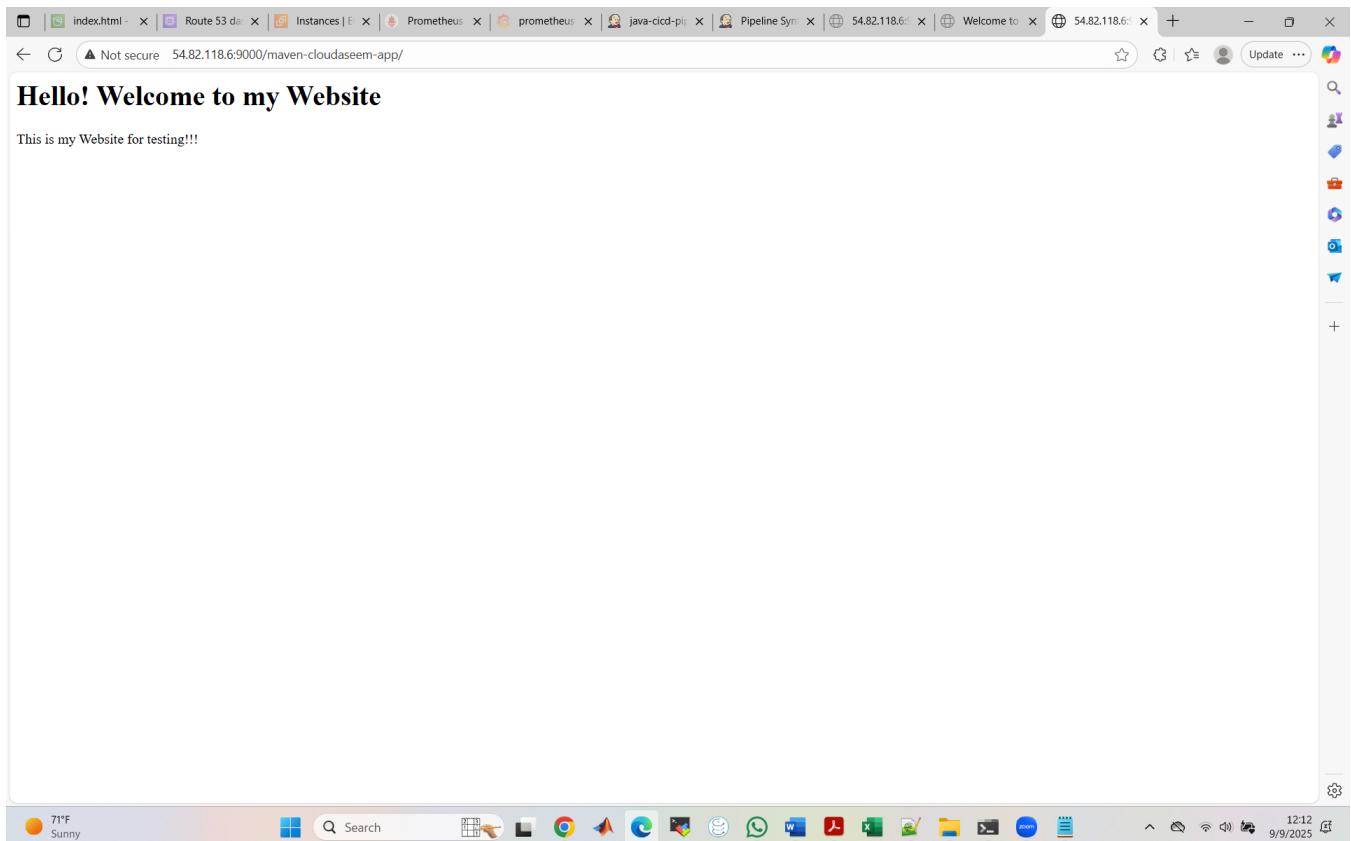
This screenshot is identical to the one above, but it shows the result of a successful build. The Build Now button is now grayed out. The Stage View table shows updated times: 255ms for Tool Install, 475ms for Git Checkout, 4s for Build Step, 5s for Create Image, and 753ms for Create Container. The Permalinks section remains the same. The bottom right corner shows Jenkins 2.516.2.

You can see that the build is successful.

Now, let us check if we can access our docker container by typing this one our browser

http://Public IPv4 Address:9000/maven-cloudaseem-app

That is http://54.82.118.6:9000/maven-cloudaseem-app



The application is working. So, we are able to access our Java application.

We can now verify if we have created an image using the command on the terminal:

```
docker images
```

A screenshot of a terminal window with a dark background. The prompt is 'root@ip-172-31-31-119:~#'. The user runs 'cat /var/lib/jenkins/secrets/initialAdminPassword' which outputs '930e9e2e4a514caeaff048f6b9b38703'. Then, the user runs 'docker images' which lists the following images:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
sosoebot	latest	ae2fdabd5cae	2 minutes ago	470MB
<none>	<none>	03cfc2ffb4a9	7 minutes ago	470MB
hello-world	latest	1b44b5a3e06a	4 weeks ago	10.1kB

You can see that we have created an image as expected. Let us also check if we have created a container using the command:

```
docker ps -a
```

```

root@ip-172-31-31-119:~# cat /var/lib/jenkins/secrets/initialAdminPassword
930e9e2e4a514caeaff048f6b9b38703
root@ip-172-31-31-119:~# docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
sosoebot        latest   ae2fdabd5cae  2 minutes ago  470MB
<none>          <none>   03cfc2fffb4a9  7 minutes ago  470MB
hello-world     latest   1b44bb5a3e06a  4 weeks ago   10.1kB
root@ip-172-31-31-119:~# docker ps -a
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
e773d3b7ed4f    sosoebot   "catalina.sh run"  3 minutes ago  Up 3 minutes  0.0.0.0:9000->8080/tcp, [::]:9000->8080/tcp   sosoebot_container
798a8b157df6    hello-world "/hello"      2 hours ago   Exited (0) 2 hours ago
b2fb6b49856    hello-world "/hello"      2 hours ago   Exited (0) 2 hours ago
r
root@ip-172-31-31-119:~#

```

You can see our container.

STEP 12: Install and configure Nginx and Certbot on EC2 server

Nginx is one of the most popular web servers in the world and is responsible for hosting some of the largest and highest-traffic sites on the internet. It is a lightweight choice that can be used as either a web server or reverse proxy.

Part 1: Update and upgrade package

In this step, we have to install Nginx. To do that we first Update Package Repository and Upgrade Packages using these commands:

```
sudo apt update
```

```

root@ip-172-31-31-119:~# + ~
e773d3b7ed4f    sosoebot   "catalina.sh run"  3 minutes ago  Up 3 minutes  0.0.0.0:9000->8080/t
cp, [::]:9000->8080/tcp  sosoebot_container
798a8b157df6    hello-world "/hello"      2 hours ago   Exited (0) 2 hours ago
serene_rosalind
b2fb6b49856    hello-world "/hello"      2 hours ago   Exited (0) 2 hours ago
pensive_stonebraker
root@ip-172-31-31-119:~# sudo apt 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]
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:6 https://pkg.jenkins.io/debian-stable binary/ Release
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.2 kB]
Hit:8 http://security.ubuntu.com/ubuntu noble-security InRelease
Fetched 141 kB in 1s (165 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
17 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-31-119:~#

```

```
sudo apt upgrade
```

```
root@ip-172-31-31-119: ~ + | x - □ ×
Reading state information... Done
17 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-31-119:~# sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following NEW packages will be installed:
  linux-aws-6.14-headers-6.14.0-1012 linux-aws-6.14-tools-6.14.0-1012 linux-headers-6.14.0-1012-aws
  linux-image-6.14.0-1012-aws linux-modules-6.14.0-1012-aws linux-tools-6.14.0-1012-aws
The following upgrades have been deferred due to phasing:
  coreutils
The following packages will be upgraded:
  fwupd libfwupd2 libpython3.12-minimal libpython3.12-stdlib libpython3.12t64 libudisks2-0 linux-aws
  linux-headers-aws linux-image-aws linux-tools-common openssh-client openssh-server openssh-sftp-server
  python3.12 python3.12-minimal udisks2
16 upgraded, 6 newly installed, 0 to remove and 1 not upgraded.
11 standard LTS security updates
Need to get 88.4 MB of archives.
After this operation, 178 MB of additional disk space will be used.
Do you want to continue? [Y/n] |
```

Type “Y” and press ENTER

```
root@ip-172-31-31-119: ~ + | x - □ ×
Running kernel version:
  6.14.0-1011-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.14.0-1012-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:
  systemctl restart networkd-dispatcher.service
  systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
  ubuntu @ session #2: sshd[1096]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-31-119:~# |
```

Now we want a domain name to take over the IP of the server so that instead of hitting the IP we can use the domain name that is TLS enabled. Let's see how to do so.

I will be using the sub-domain name **sonarqube.sosoebot.org**. This is mine. Please use yours.

Install Nginx

```
sudo apt install nginx -y
```

```
root@ip-172-31-31-119: ~ + - ×
Running kernel version:
  6.14.0-1011-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.14.0-1012-aws.

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

Restarting services...

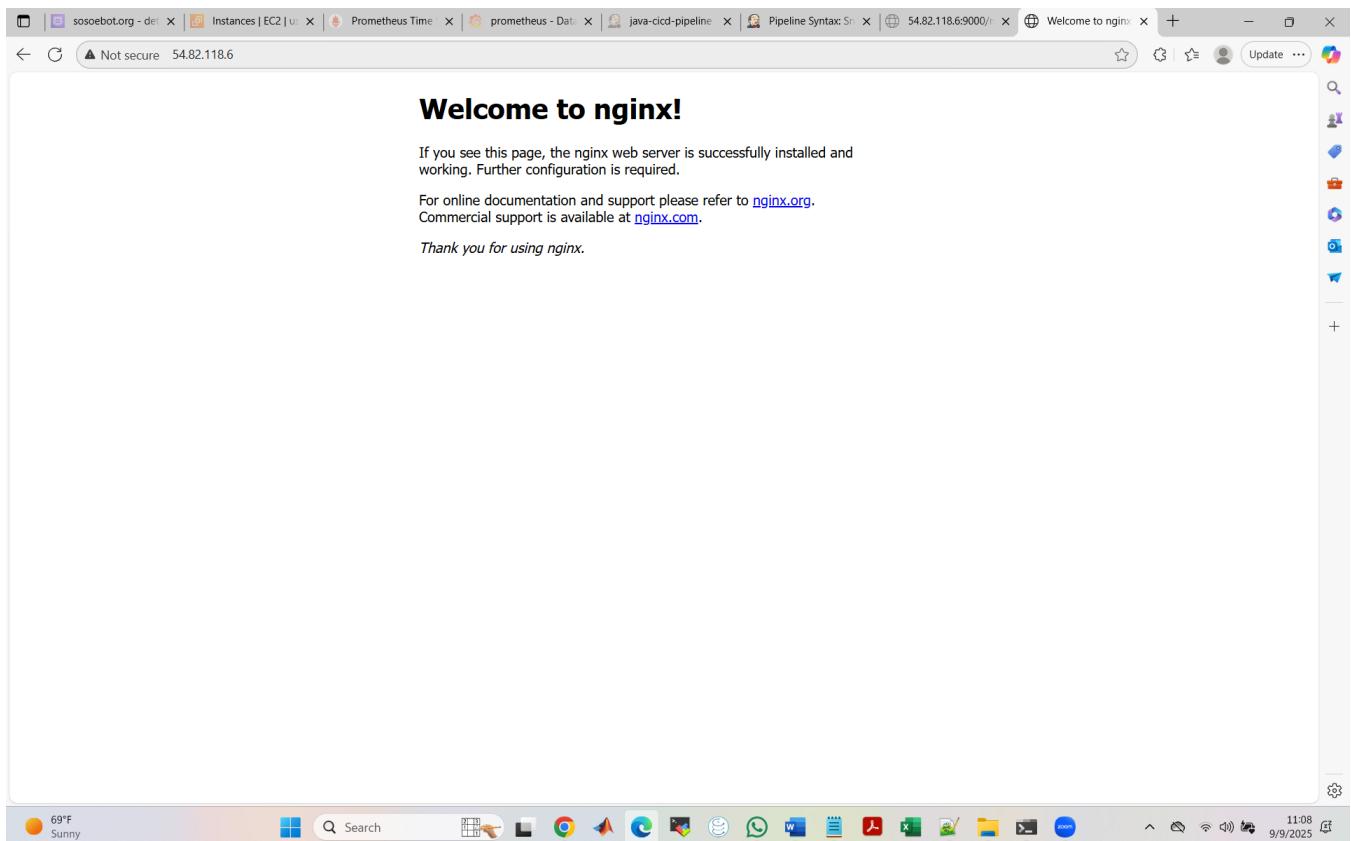
Service restarts being deferred:
  systemctl restart networkd-dispatcher.service
  systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
  ubuntu @ session #2: sshd[1096]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-31-119:~# |
```

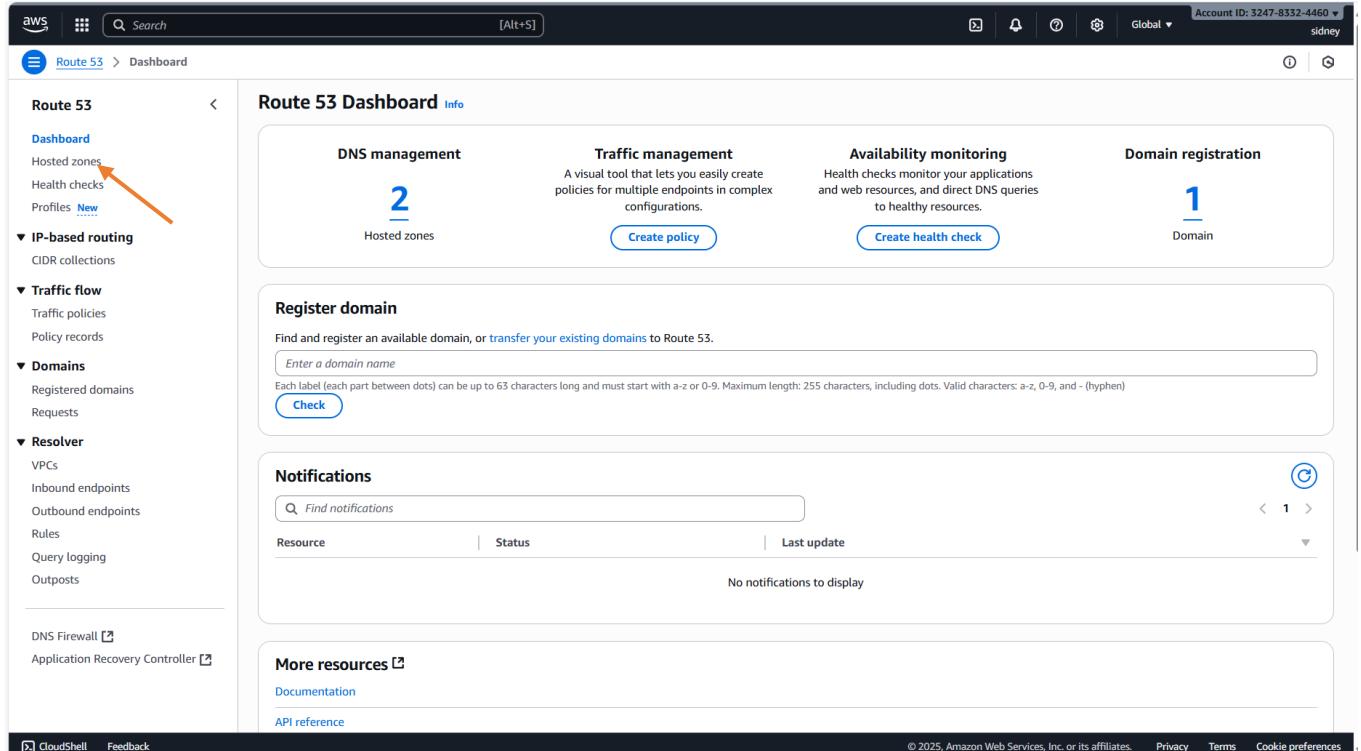
Verify installation by using: <http://Public IPv4 Address>, that is <http://54.82.118.6>



This shows that Nginx has been installed successfully

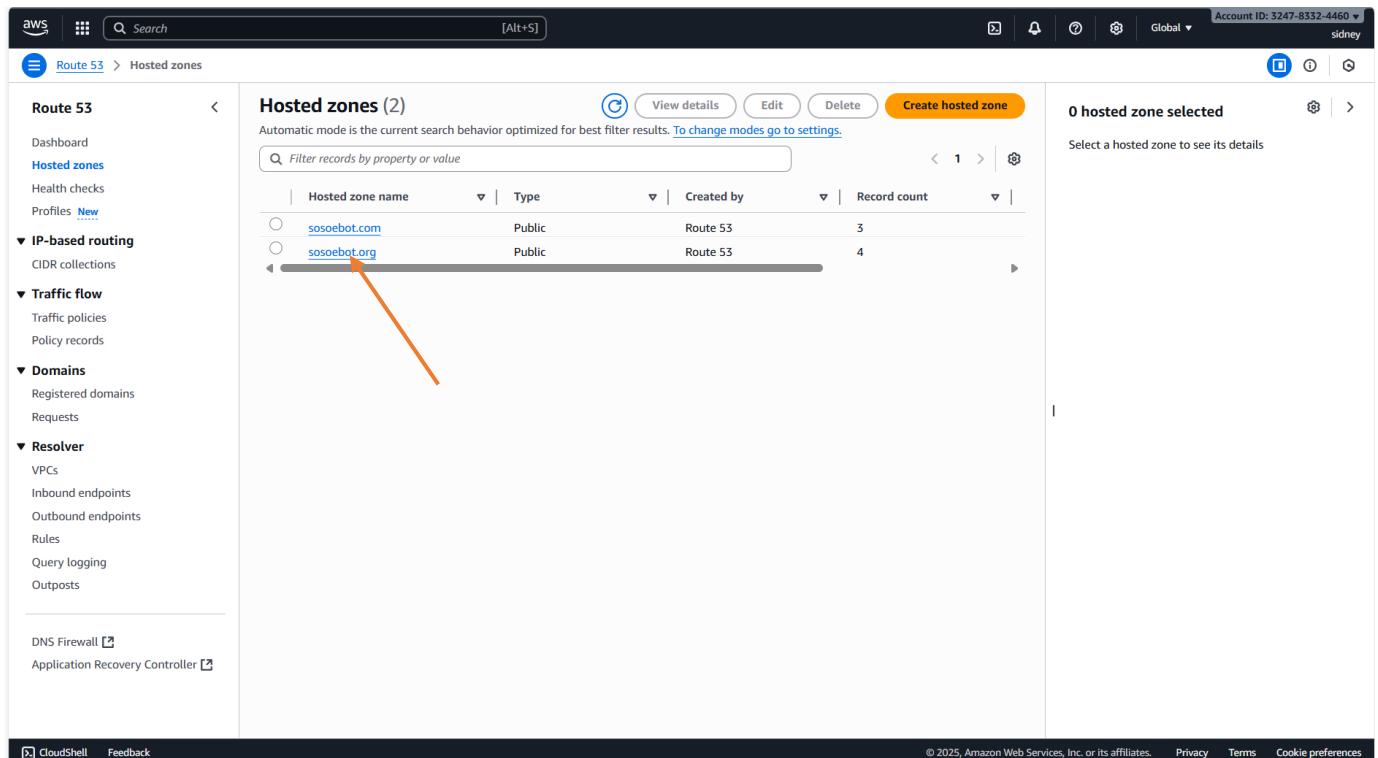
Add the DNS records

My domain is hosted on the AWS Route53, thus I will be adding the records in there. Go to AWS Console and search for “Route53”



The screenshot shows the AWS Route 53 Dashboard. On the left, a sidebar menu includes 'Hosted zones' (which is highlighted with an orange arrow), 'Health checks', 'Profiles', 'IP-based routing', 'Traffic flow', 'Domains', 'Resolver', 'DNS Firewall', and 'Application Recovery Controller'. The main content area is titled 'Route 53 Dashboard' and contains four sections: 'DNS management' (2 Hosted zones, 'Create policy'), 'Traffic management' (A visual tool for creating traffic policies), 'Availability monitoring' (Health checks monitor applications and web resources), and 'Domain registration' (1 Domain, 'Create health check'). Below these sections is a 'Register domain' form and a 'Notifications' section. At the bottom, there are 'More resources' links for 'Documentation' and 'API reference'.

Adding the IP of the server on the DNS zone. Click on “Hosted Zones”



The screenshot shows the 'Hosted zones' page under the 'Route 53' service. The sidebar is identical to the previous dashboard. The main table lists two hosted zones: 'sosoebot.com' (Public, created by Route 53, record count 3) and 'sosoebot.org' (Public, created by Route 53, record count 4). An orange arrow points to the 'sosoebot.org' row. To the right of the table, a sidebar says '0 hosted zone selected' and 'Select a hosted zone to see its details'. At the bottom, there are 'CloudShell', 'Feedback', and copyright information.

And now our domain sonarqube.sosoebot.org is ready to be used for the nginx configuration setup. Click on “sosoebot.org”

Click on “Create Record”

Enter “jenkins”

Screenshot of the AWS Route 53 'Create record' page for the domain 'sosoebot.org'. The 'Record name' field contains 'jenkins' and the 'Value' field contains '192.0.2.235'. An orange arrow points from the 'Value' input field to the 'Add another record' button at the bottom right.

► View existing records
The following table lists the existing records in sosoebot.org.

[Cancel](#) [Create records](#)

Screenshot of the AWS Route 53 'Create record' page for the domain 'sosoebot.org'. The 'Record name' field contains 'jenkins' and the 'Value' field contains '54.82.118.6'. An orange arrow points from the 'Create records' button at the bottom right to the 'Create records' button at the top right of the page.

Click on “Create records”

Create a new Nginx configuration file for the site

```
sudo vim /etc/nginx/sites-available/jenkins
```

```
sudo vim /etc/nginx/sites-enabled/ddsonarqube
```

Add this configuration so that Nginx will route incoming traffic to Jenkins.

```
server {
```

```

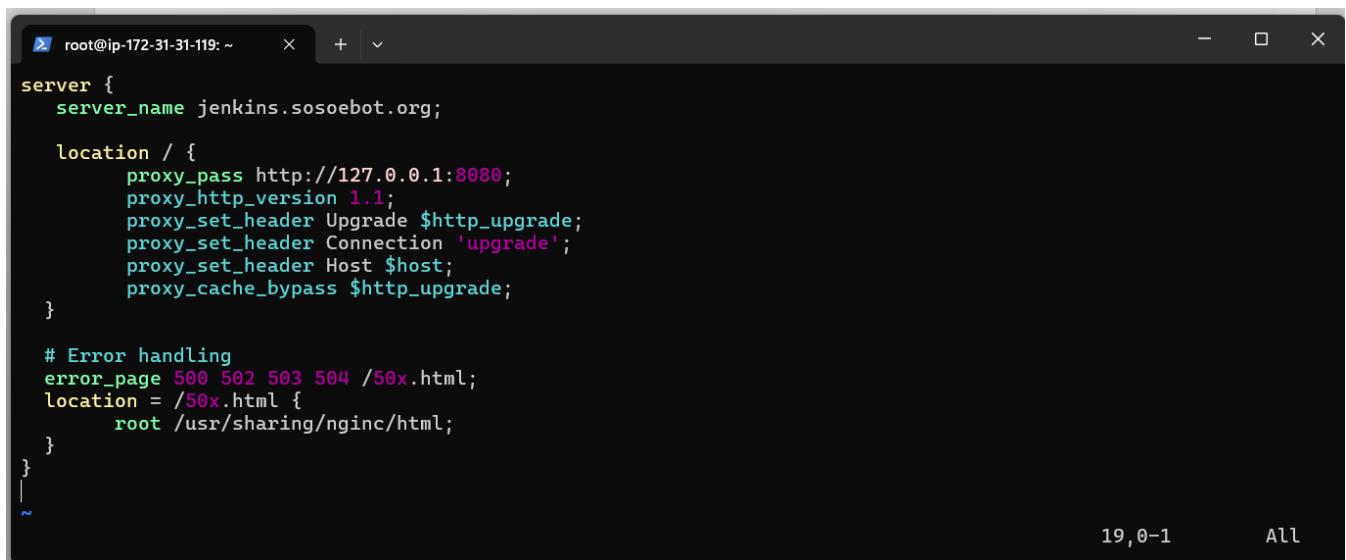
server_name jenkins.sosoebot.org;

location / {
    proxy_pass http://127.0.0.1:8080;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}

# Error handling
error_page 500 502 503 504 /50x.html;
location = /50x.html {
    root /usr/sharing/nginc/html;
}
}

```

Paste this code



A screenshot of a terminal window titled "root@ip-172-31-31-119: ~". The window contains the Nginx configuration code shown in the previous block. The code is syntax-highlighted, with "server" in blue, "location" in green, and various directives in different colors like purple and cyan. The Vim status bar at the bottom right shows "19,0-1" and "All".

```

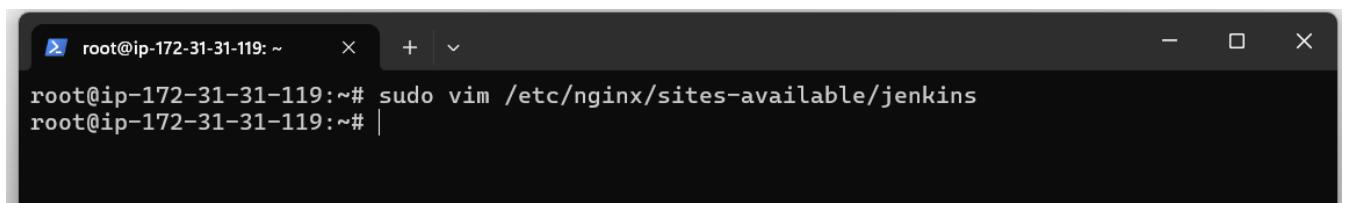
server {
    server_name jenkins.sosoebot.org;

    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    # Error handling
    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root /usr/sharing/nginc/html;
    }
}

```

Save it by Pressing ESC, followed by :wq and press ENTER



A screenshot of a terminal window titled "root@ip-172-31-31-119: ~". The user has run the command "sudo vim /etc/nginx/sites-available/jenkins". The Vim status bar at the bottom right shows "19,0-1" and "All".

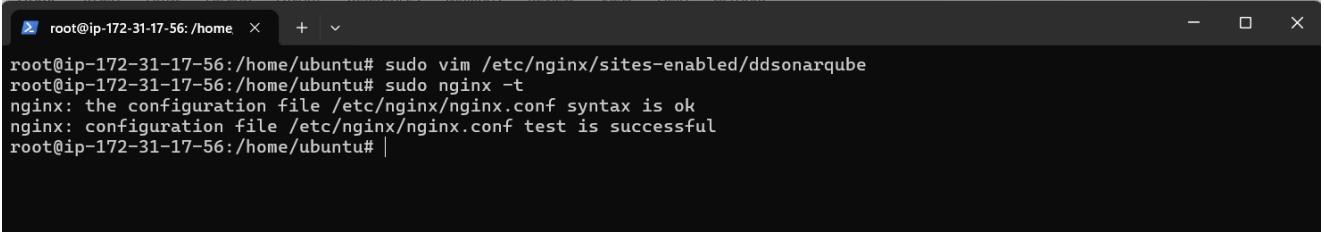
```

root@ip-172-31-31-119:~# sudo vim /etc/nginx/sites-available/jenkins
root@ip-172-31-31-119:~# |

```

Make sure your configuration file has no syntax errors.

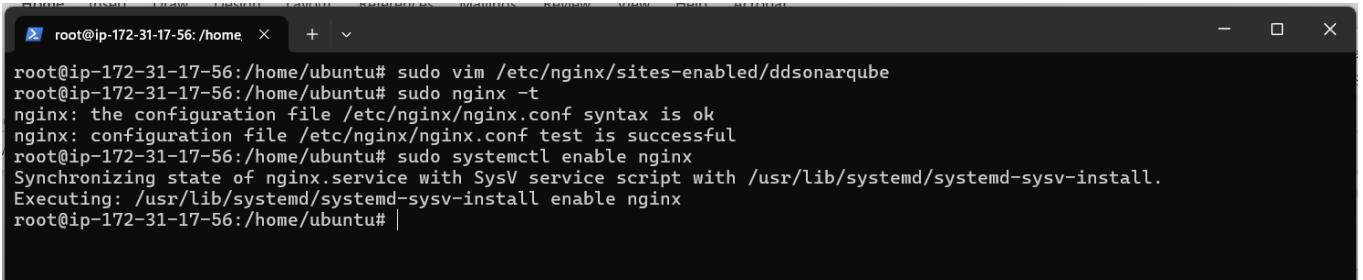
sudo nginx -t



```
root@ip-172-31-17-56:/home/ubuntu# sudo vim /etc/nginx/sites-enabled/ddsonarqube
root@ip-172-31-17-56:/home/ubuntu# sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@ip-172-31-17-56:/home/ubuntu#
```

Enable the Nginx site and restart Nginx

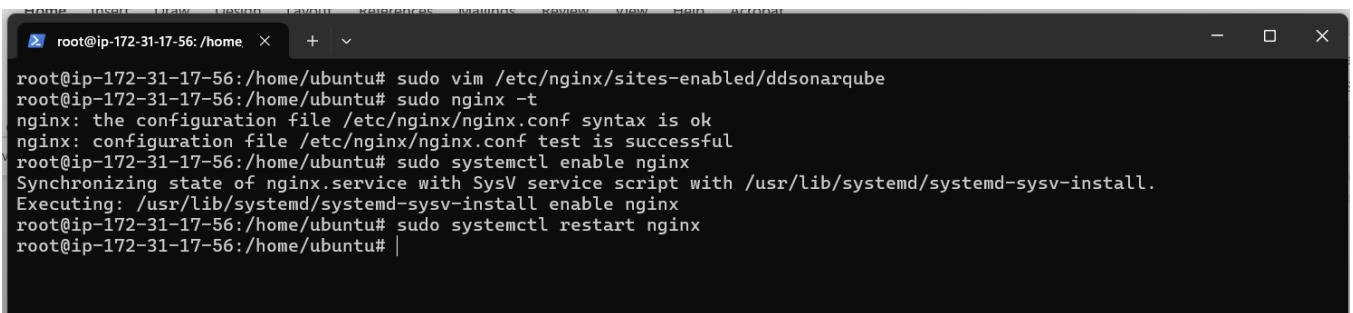
```
sudo systemctl enable nginx
```



```
root@ip-172-31-17-56:/home/ubuntu# sudo vim /etc/nginx/sites-enabled/ddsonarqube
root@ip-172-31-17-56:/home/ubuntu# sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@ip-172-31-17-56:/home/ubuntu# sudo systemctl enable nginx
Synchronizing state of nginx.service with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
root@ip-172-31-17-56:/home/ubuntu#
```

Restart the nginx server

```
sudo systemctl restart nginx
```



```
root@ip-172-31-17-56:/home/ubuntu# sudo vim /etc/nginx/sites-enabled/ddsonarqube
root@ip-172-31-17-56:/home/ubuntu# sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@ip-172-31-17-56:/home/ubuntu# sudo systemctl enable nginx
Synchronizing state of nginx.service with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
root@ip-172-31-17-56:/home/ubuntu# sudo systemctl restart nginx
root@ip-172-31-17-56:/home/ubuntu#
```

Check the status of the nginx server

```
sudo systemctl status nginx
```

```

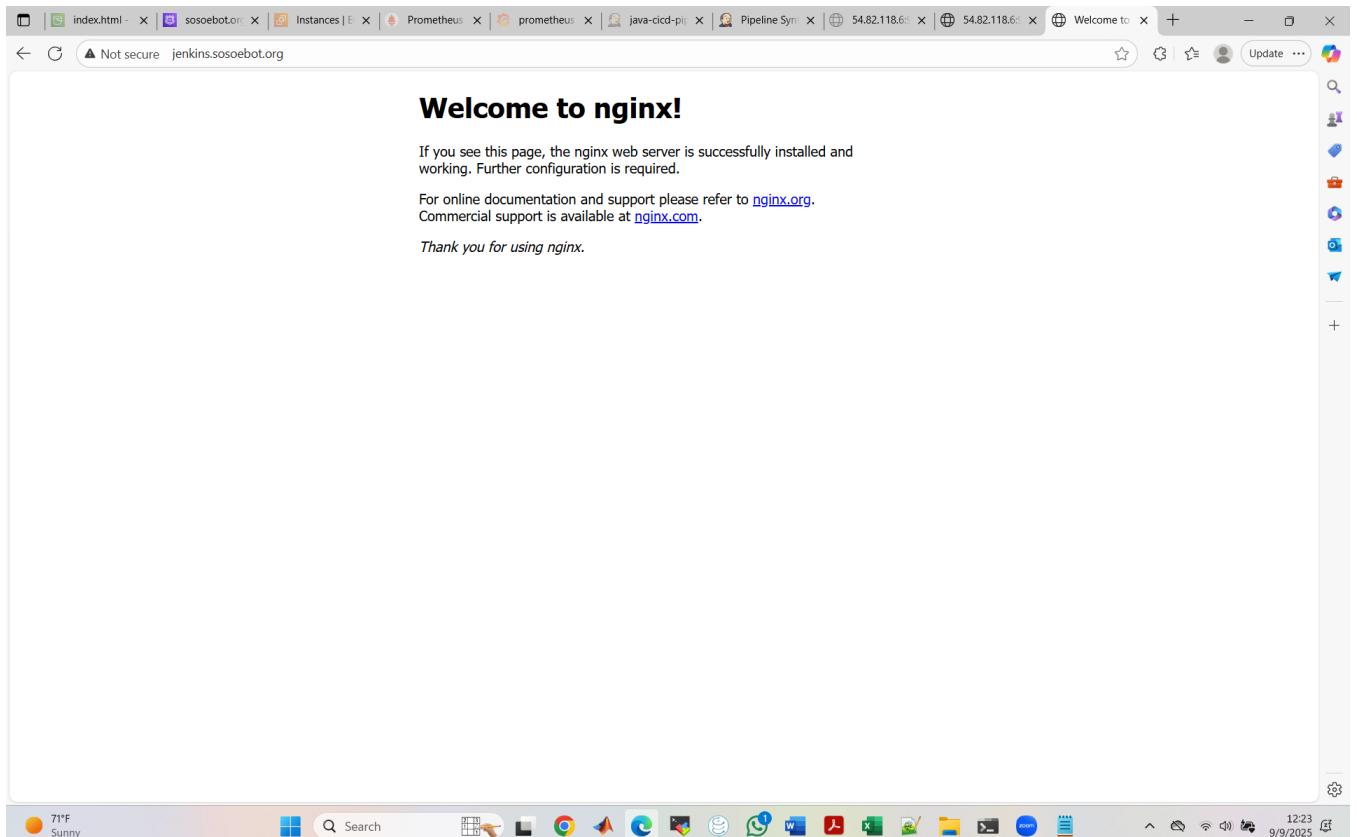
root@ip-172-31-17-56:/home/ubuntu# sudo vim /etc/nginx/sites-enabled/ddsonarqube
root@ip-172-31-17-56:/home/ubuntu# sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@ip-172-31-17-56:/home/ubuntu# sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
root@ip-172-31-17-56:/home/ubuntu# sudo systemctl restart nginx
root@ip-172-31-17-56:/home/ubuntu# sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
    Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
    Active: active (running) since Sun 2025-09-07 02:09:27 UTC; 1min 23s ago
      Docs: man:nginx(8)
   Process: 2808 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 2810 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 2811 (nginx)
    Tasks: 3 (limit: 4667)
   Memory: 2.4M (peak: 2.7M)
      CPU: 12ms
     CGroup: /system.slice/nginx.service
             ├─2811 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─2812 "nginx: worker process"
             └─2813 "nginx: worker process"

Sep 07 02:09:27 ip-172-31-17-56 systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy >
Sep 07 02:09:27 ip-172-31-17-56 systemd[1]: Started nginx.service - A high performance web server and a reverse proxy s>
lines 1-17/17 (END)

```

Use the configured domain

You can now visit <http://jenkins.sosoebot.org> in your web browser. You will be greeted with the Jenkins web interface.

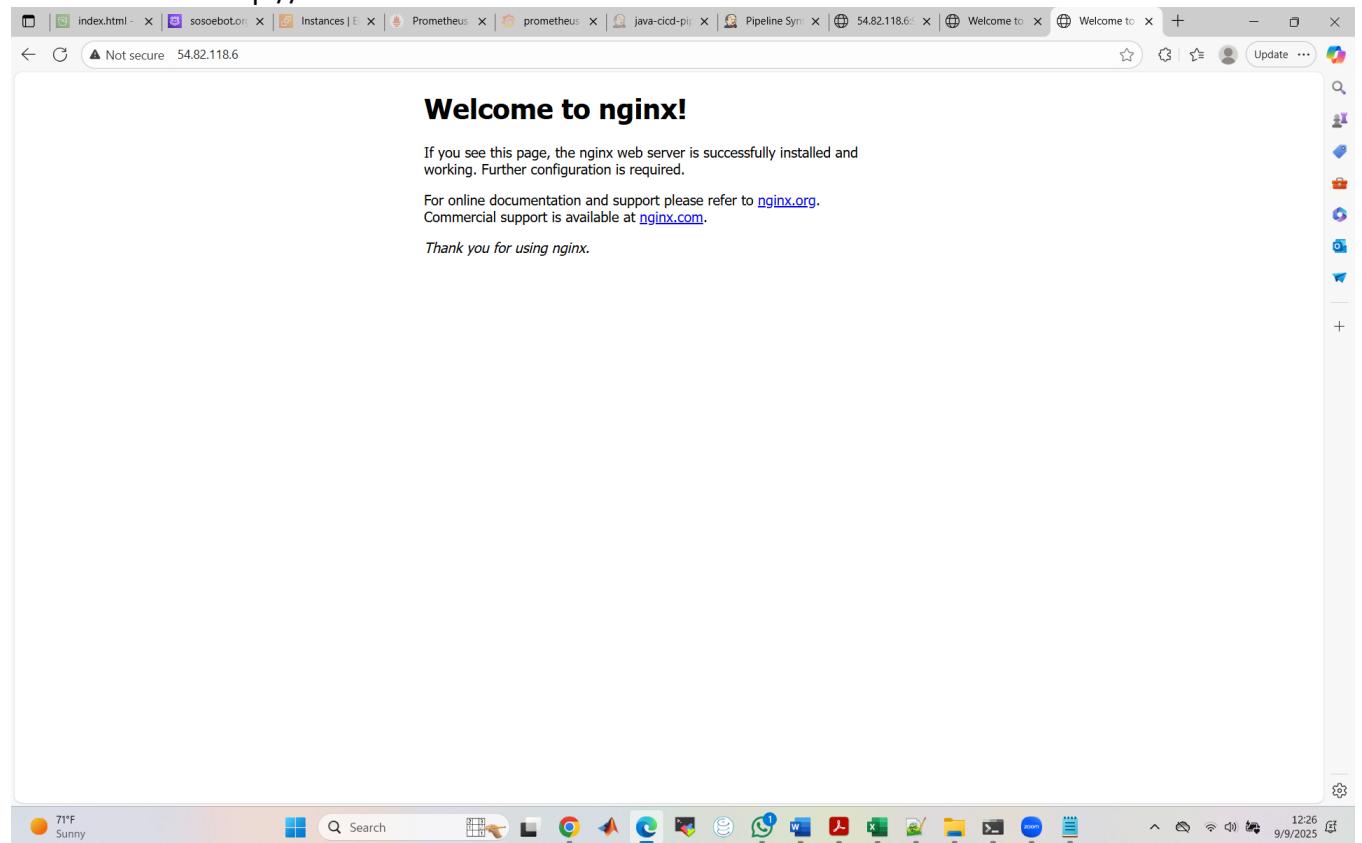


The configured domain sonarqube.sosoebot.org is working perfectly. Wait, seems there is a problem ahead, let's solve it. There seems to be a problem ahead

Problem:

But the domain is working on HTTP protocol and we need to make it secure by making it work with the HTTPS protocol with SSL certificate.

Now let us test <http://54.82.118.6>



Part 2: Secure with SSL using Certbot

We will Install SSL certificate for maximum security. We will use the free SSL certificate provided by Certbot for our sub-domain sonarqube.sosoebot.org. Let's see how we will do this.

Installing certbot

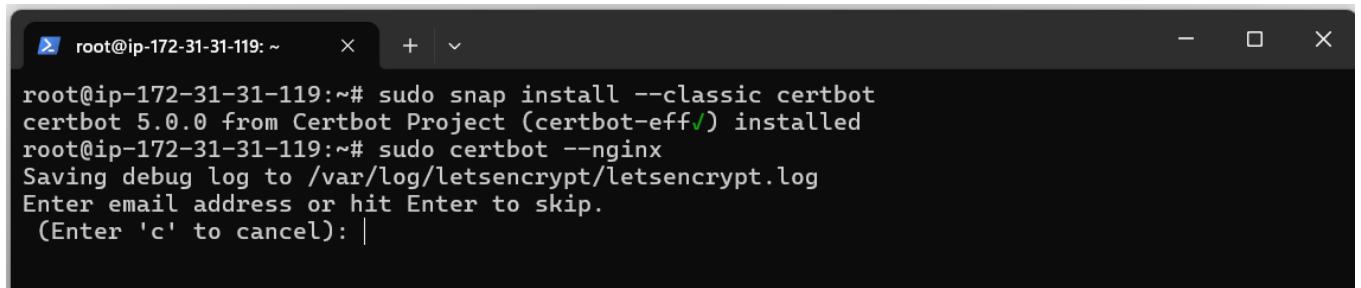
```
sudo snap install --classic certbot
```

```
root@ip-172-31-31-119:~# sudo snap install --classic certbot
certbot 5.0.0 from Certbot Project (certbot-eff) installed
root@ip-172-31-31-119:~# |
```

Obtain and install an SSL certificate on the subdomain

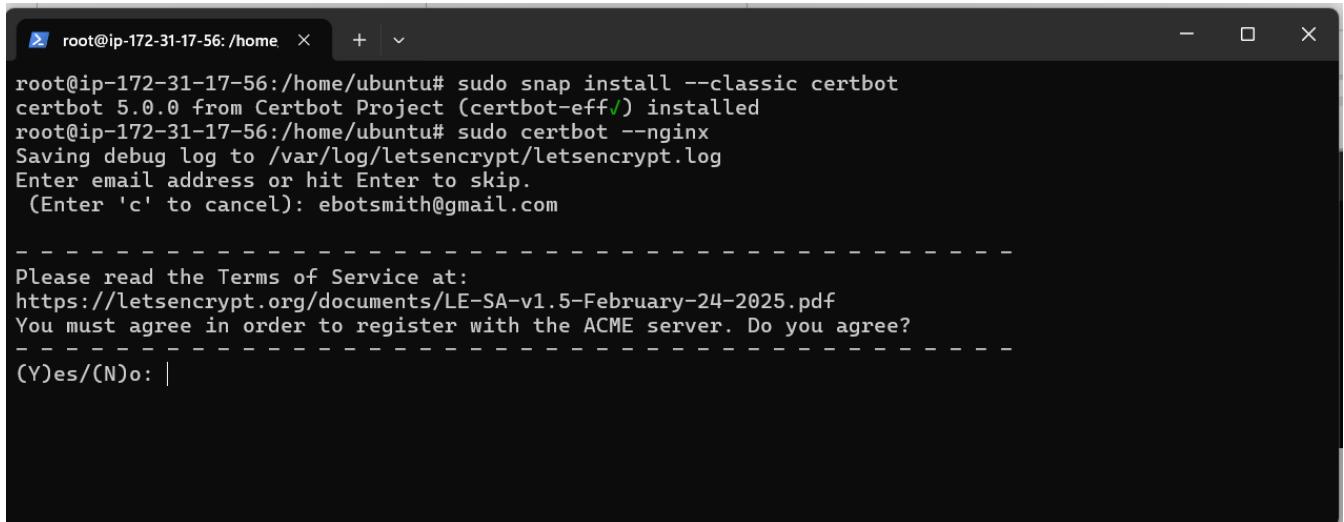
Generating SSL certificate for the required domain. Please read the asked question carefully and answer them accordingly.

```
sudo certbot --nginx
```



```
root@ip-172-31-31-119:~# sudo snap install --classic certbot
certbot 5.0.0 from Certbot Project (certbot-eff✓) installed
root@ip-172-31-31-119:~# sudo certbot --nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address or hit Enter to skip.
(Enter 'c' to cancel): |
```

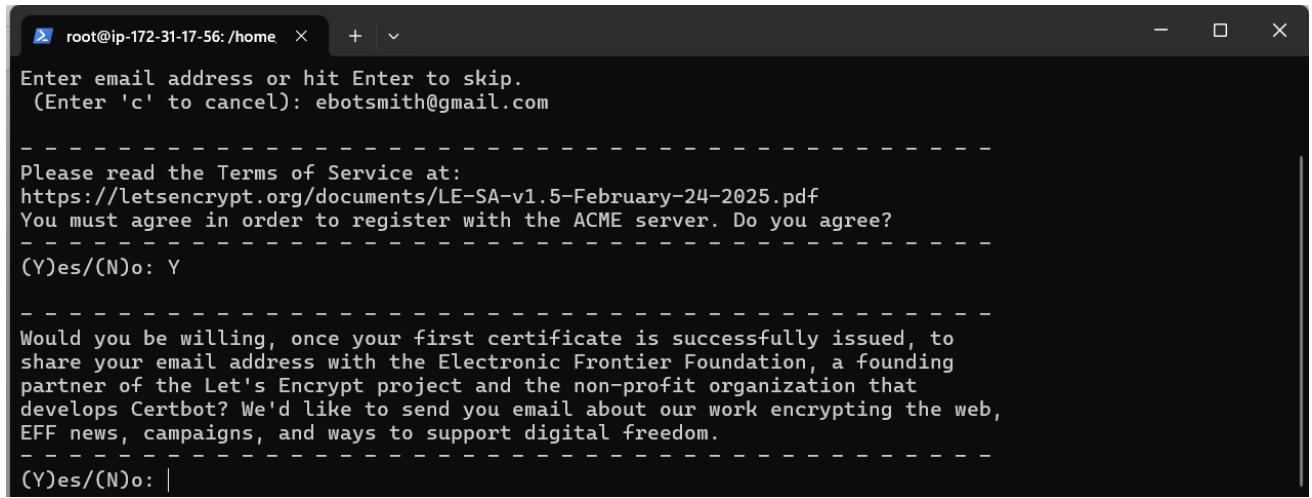
Enter your email address. I will enter “ebotsmith@gmail.com”



```
root@ip-172-31-17-56:/home/ubuntu# sudo snap install --classic certbot
certbot 5.0.0 from Certbot Project (certbot-eff✓) installed
root@ip-172-31-17-56:/home/ubuntu# sudo certbot --nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address or hit Enter to skip.
(Enter 'c' to cancel): ebotsmith@gmail.com

-----
Please read the Terms of Service at:
https://letsencrypt.org/documents/LE-SA-v1.5-February-24-2025.pdf
You must agree in order to register with the ACME server. Do you agree?
-----
```

Enter “Y” and press ENTER



```
root@ip-172-31-17-56:/home#
Enter email address or hit Enter to skip.
(Enter 'c' to cancel): ebotsmith@gmail.com

-----
Please read the Terms of Service at:
https://letsencrypt.org/documents/LE-SA-v1.5-February-24-2025.pdf
You must agree in order to register with the ACME server. Do you agree?
-----
```

(Y)es/(N)o: Y

```
-----
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
-----
```

(Y)es/(N)o: |

Enter “N” and press ENTER

```
root@ip-172-31-31-119: ~      + | - | X
Please read the Terms of Service at:
https://letsencrypt.org/documents/LE-SA-v1.5-February-24-2025.pdf
You must agree in order to register with the ACME server. Do you agree?
----- (Y)es/(N)o: Y
----- Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
----- (Y)es/(N)o: N
Account registered.
Please enter the domain name(s) you would like on your certificate (comma and/or
space separated) (Enter 'c' to cancel): |
```

Enter "jenkins.sosoebot.org"

```
root@ip-172-31-31-119: ~      + | - | X
Please read the Terms of Service at:
https://letsencrypt.org/documents/LE-SA-v1.5-February-24-2025.pdf
You must agree in order to register with the ACME server. Do you agree?
----- (Y)es/(N)o: Y
----- Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
----- (Y)es/(N)o: N
Account registered.
Please enter the domain name(s) you would like on your certificate (comma and/or
space separated) (Enter 'c' to cancel): jenkins.sosoebot.org|
```

Press Enter

```
root@ip-172-31-31-119: ~      + | - | X
Key is saved at: /etc/letsencrypt/live/jenkins.sosoebot.org/privkey.pem
This certificate expires on 2025-12-08.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

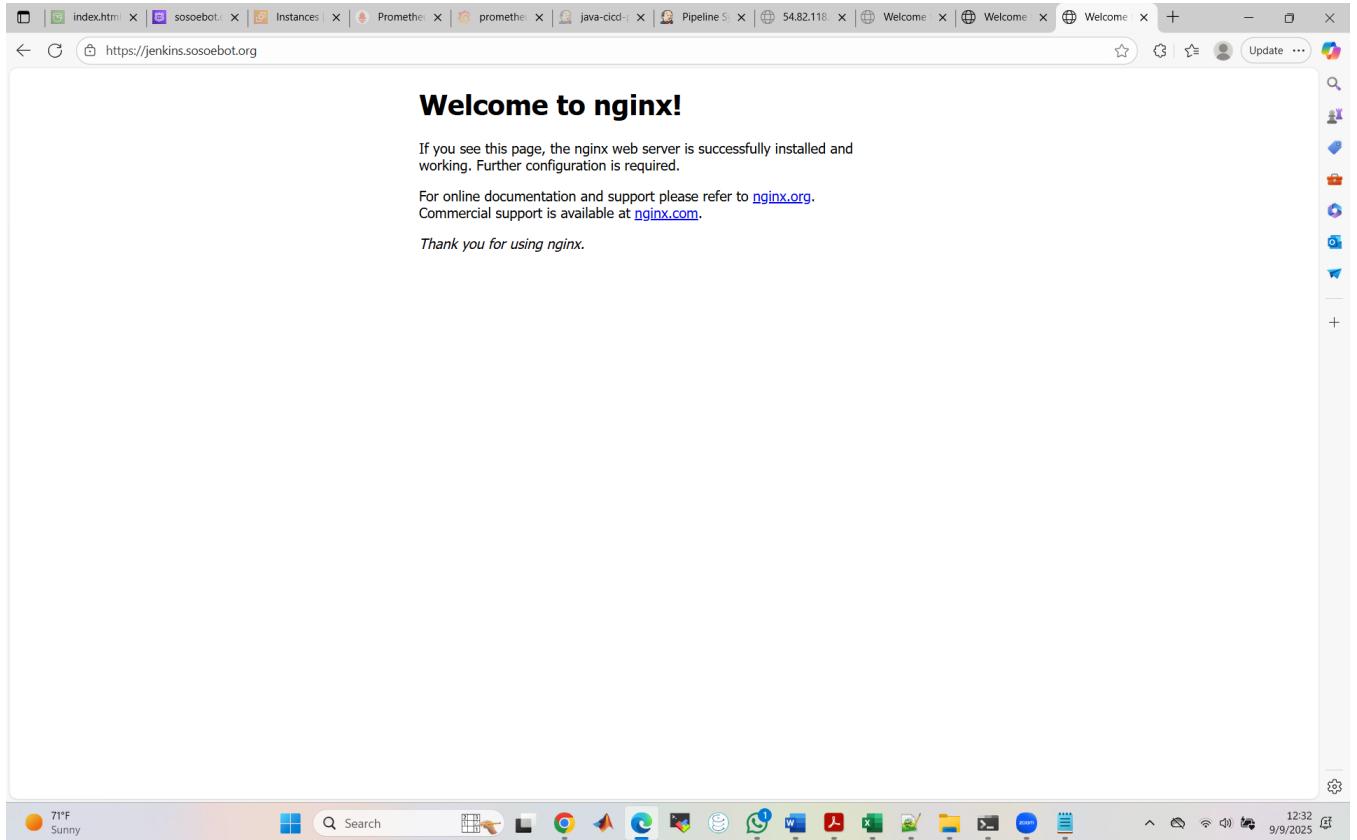
Deploying certificate
Successfully deployed certificate for jenkins.sosoebot.org to /etc/nginx/sites-enabled/default
Congratulations! You have successfully enabled HTTPS on https://jenkins.sosoebot.org

----- If you like Certbot, please consider supporting our work by:
* Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
* Donating to EFF: https://eff.org/donate-le
----- root@ip-172-31-31-119:~# |
```

Success message that the SSL certificate has been deployed on the domain sonarqube.sosoebot.org

Verify if Certificate deployed successfully

Now if we hit the domain **jenkins.sosoebot.org**, it automatically redirects to HTTPS.



Now you can access SonarQube through your domain name or EC2 public IP address via Nginx (on port 80 or 443 since SSL has been configured).

It is directing us to the Nginx web page, but we want to access our webpage. We will fix this we have to give the port access. To do this, we have to add another stage in our pipeline.

The screenshot shows the Jenkins Pipeline configuration page. The left sidebar has tabs for General, Triggers, Pipeline (which is selected), and Advanced. The main area is titled 'Pipeline' with the sub-instruction 'Define your Pipeline using Groovy directly or pull it from source control.' Below this is a 'Definition' section with a 'Pipeline script' dropdown. A large code editor window displays the following Groovy script:

```

21
22
23
24
25
26
27
28
29
30
31
32
33
34
35

```

An orange arrow points to the second 'stage('Create Container')' line. At the bottom of the code editor, there is a checked checkbox labeled 'Use Groovy Sandbox'.

Change the stage name from “Create Container” to “Check Port”

The screenshot shows the Jenkins Pipeline configuration page after the stage name change. The left sidebar and main area are identical to the previous screenshot, but the code editor now shows the modified script:

```

21
22
23
24
25
26
27
28
29
30
31
32
33
34
35

```

An orange arrow points to the second 'stage('Check Port')' line. The rest of the interface is identical to the first screenshot.

Then, modify the step as follows:

steps {

```

script {
    def portStatus = sh(script: 'netstat -tuln | grep ":9000"', returnStatus: true)
    if (portStatus == 0) {
        echo "Port 9000 is in use, stopping and removing the existing container"
        sh 'docker stop sosoebot_container'
        sh 'docker rm sosoebot_container'
    } else {
        echo "Port 9000 is available"
    }
}
}

```

The screenshot shows the Jenkins Pipeline configuration page. The left sidebar has tabs for General, Triggers, Pipeline (which is selected and highlighted with a red box), and Advanced. The main area is titled 'Pipeline' with the sub-instruction 'Define your Pipeline using Groovy directly or pull it from source control.' Below this is a 'Definition' section with a dropdown set to 'Pipeline script'. A large text area contains the Groovy script provided above. A checkbox labeled 'Use Groovy Sandbox' is checked. At the bottom are 'Save' and 'Apply' buttons.

Then click on “**Apply**” followed by “**Save**”

Jenkins / java-cicd-pipeline

java-cicd-pipeline

Java Based CICD Pipeline

Stage View

Average stage times:
(full run time: ~8s)

Declarative: Tool Install	Git Checkout	Build Step	Create Image	Create Container
172ms	540ms	3s	10s	1s
#10 Aug 29 14:28 1 commit	148ms	476ms	3s	10s
#11 Aug 29 12:22 No Changes	153ms	510ms	3s	
#7 Aug 29 01:30 No Changes	240ms	710ms	4s	
#6 Aug 29 01:11 No Changes	148ms	466ms	3s	

Permalinks

- Last build (#10), 7 hr 26 min ago
- Last stable build (#10), 7 hr 26 min ago
- Last successful build (#10), 7 hr 26 min ago

Builds

Filter

August 29, 2025

- #10 6:28 PM
- #8 4:22 PM
- #7 5:30 AM
- #6 5:11 AM
- #5 5:08 AM
- #2 3:44 AM
- #1 2:38 AM

21:54 8/29/2025

Then click on "Build Now"

Jenkins / java-cicd-pipeline

java-cicd-pipeline

Java Based CICD Pipeline

Stage View

Average stage times:
(full run time: ~8s)

Declarative: Tool Install	Git Checkout	Build Step	Create Image	Create Container	Check Port
236ms	491ms	4s	3s	675ms	1s
#12 Sep 09 12:36 No Changes	197ms	455ms	3s	766ms	502ms
#10 Sep 09 12:12 No Changes	218ms	579ms	3s	748ms	771ms
#8 Sep 09 10:54 No Changes	145ms	494ms	3s	1s	753ms
#7 Sep 09 10:49 No Changes	133ms	465ms	3s	10s	
#6 Sep 09 10:44 No Changes	489ms	466ms	5s		

Permalinks

Builds

Filter

Today

- #12 4:36 PM
- #10 4:12 PM
- #8 2:54 PM
- #7 2:49 PM
- #6 2:44 PM
- #3 2:27 PM

72°F Sunny 12:36 9/9/2025

The build is successful

Jenkins Pipeline Code

```
pipeline {
    agent any

    tools{
        maven 'maven3'
    }

    stages {
        stage('Git Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/ebotsidneysmith/maven-app.git'
            }
        }
        stage('Build Step') {
            steps {
                sh 'mvn clean package'
            }
        }
        stage('Create Image') {
            steps {
                sh 'docker build -t sosoebot .'
            }
        }
        stage('Create Container') {
            steps {
                sh 'docker run -d -p 9000:8080 --name sosoebot_container -t sosoebot'
            }
        }
        stage('Check Port') {
            steps {
                script {
                    def portStatus = sh(script: 'netstat -tuln | grep ":9000"', returnStatus: true)
                    if (portStatus == 0) {
                        echo "Port 9000 is in use, stopping and removing the existing container"
                        sh 'docker stop sosoebot_container'
                        sh 'docker rm sosoebot_container'
                    } else {
                        echo "Port 9000 is available"
                    }
                }
            }
        }
    }
}
```