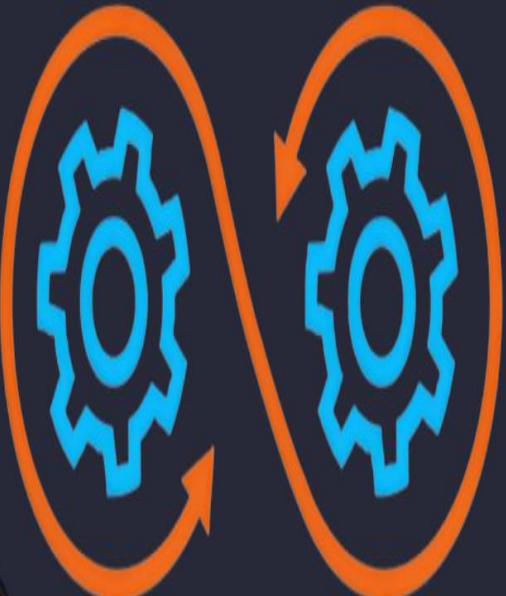


Project

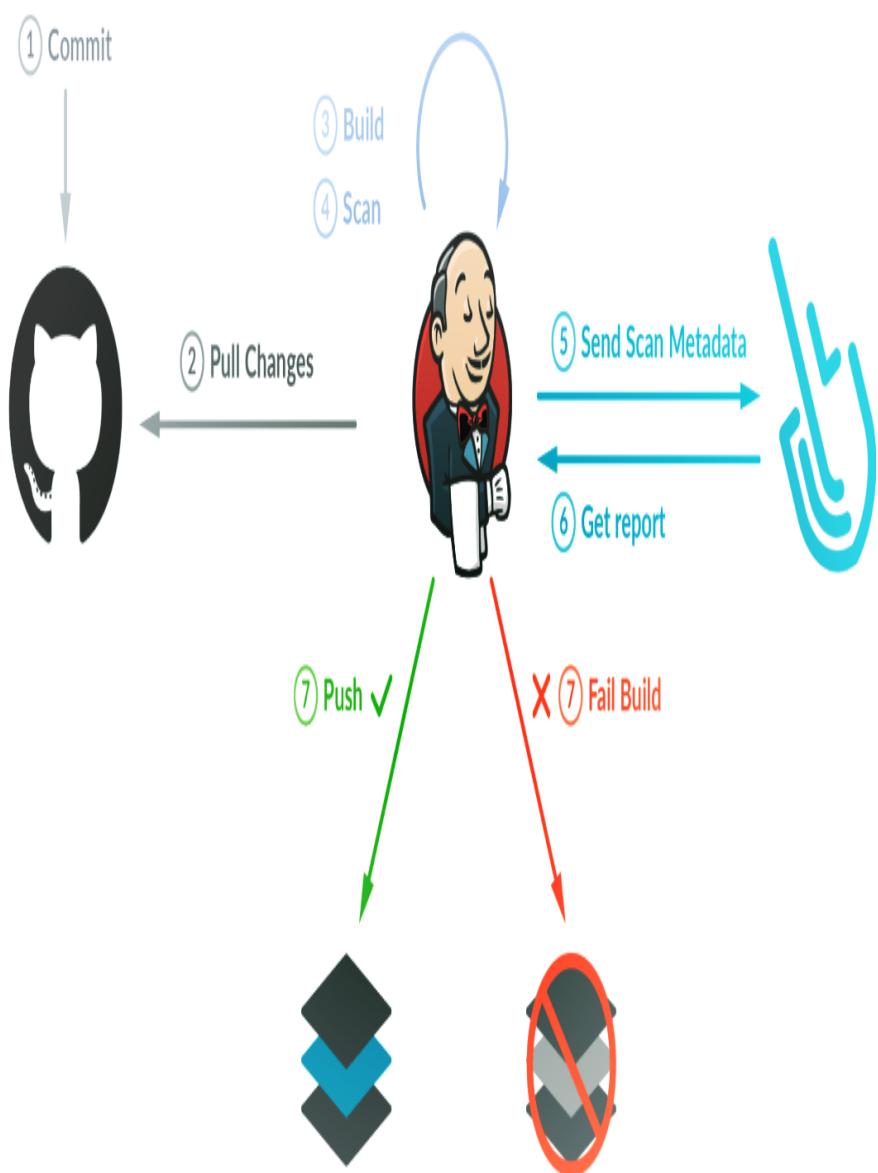
CI Pipelining using Jenkins

Created By : Aditya Kumar Verma

Guided By : Ashish Mittal Sir



Introduction



Jenkins is one of the most widely used open-source CI/CD DevOps tools. It enables developers to implement CI/CD pipelines within the development environment in a comprehensive manner. Jenkins is written in Java and supports various version control tools including Git and Maven. A pipeline in Jenkins is a set of plugins that support continuous delivery (CD) pipelines into Jenkins. A continuous delivery (CD) pipeline is an automated expression of your process for getting software from version control right through to your users and customers.

In Jenkins, a pipeline is defined as a series of steps that are executed in a specific order. These steps can be used to build, test, and deploy applications. You can use Jenkins to create a pipeline for your application by defining the steps that you want to execute. You can also use Jenkins to automate the entire process of building, testing, and deploying your application.

Why Use Jenkins?

Jenkins is a popular choice for CI (Continuous Integration) pipelining due to its versatility, extensive plugin ecosystem, and robust features. Some reasons include:

1. Flexibility and Extensibility: Jenkins provides a highly flexible and extensible framework that allows you to build custom CI/CD pipelines tailored to your specific needs. It supports a wide range of plugins for integrating with various tools, technologies, and platforms, allowing you to create complex workflows and automate diverse tasks within your pipeline.
2. Easy Integration: Jenkins seamlessly integrates with version control systems like Git, enabling it to monitor repositories for changes and trigger builds automatically. It can also integrate with other development and deployment tools, such as issue trackers, testing frameworks, and container orchestration platforms, providing a unified view of the entire development and delivery process.



3. Scalability: Jenkins is designed to handle projects of all sizes, from small teams to enterprise-scale organizations. It supports distributed builds, enabling you to distribute workloads across multiple agents or nodes, improving performance and scalability. You can also set up master-slave configurations to manage multiple Jenkins instances and delegate tasks effectively.

4. Pipeline as Code: Jenkins introduced the concept of "Pipeline as Code" with its Jenkins file, which allows you to define your entire CI/CD pipeline as a text file stored alongside your source code. This approach enables version control, code review, and collaboration on your pipeline code, promoting best practices and making it easier to manage and reproduce pipelines across environments.

5. Extensive Plugin Ecosystem: Jenkins offers a vast collection of plugins that extend its functionality and support integration with various tools and technologies. These plugins cover areas such as source code management, build automation, testing, static code analysis, artifact management, deployment, and more. The plugin ecosystem allows you to customize your pipeline to fit your specific requirements and leverage existing tools seamlessly.

Jenkins Pipelines

Jenkins pipeline is a powerful tool for automating the software delivery process. It provides a way to define and manage the entire build, test, and deployment workflow as a code-based pipeline. With Jenkins pipeline, developers can describe their entire software delivery process in a declarative or scripted manner, allowing for greater flexibility and control. The pipeline can be divided into stages, each representing a specific step in the process, such as building, testing, and deploying the application.

It integrates seamlessly with version control systems and allows for easy collaboration and continuous integration. Overall, Jenkins pipeline simplifies the software delivery process by providing a standardized and automated approach, ensuring faster, reliable, and consistent releases.

CI



Launching an Instance

- Create a EC2 Instance on AWS and connect it to web server.

Note : select t2.small or higher in the Instance Type.

The screenshot shows the AWS CloudShell interface with the 'Launch an instance' wizard open. The summary step is selected, showing the following configuration:

- Number of instances: 1
- Software Image (AMI): Canonical, Ubuntu, 22.04 LTS (ami-0f5ee92e2d63afc18)
- Virtual server type (instance type): t2.small
- Firewall (security group): New security group
- Storage (volumes): 1 volume(s) - 8 GiB

A tooltip for the 'Free tier' is displayed, stating: "Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet." At the bottom right of the summary step is a large orange 'Launch instance' button.

Below the summary step, the 'Application and OS Images (Amazon Machine Image)' step is visible, showing various AMI options like Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE. A specific Ubuntu Server 22.04 LTS (HVM), SSD Volume Type AMI is selected.

The 'Instance type' step is also partially visible at the bottom, showing the t2.small instance type selected.

Checking the 8080 port

- After launching an EC2 instance, now review the inbound rules under the security groups & add a new inbound rule for 8080 port. This is the most important step after launching the instance.

EC2 > Security Groups > sg-088c2ac61b5404440 - launch-wizard-20 > Edit inbound rules

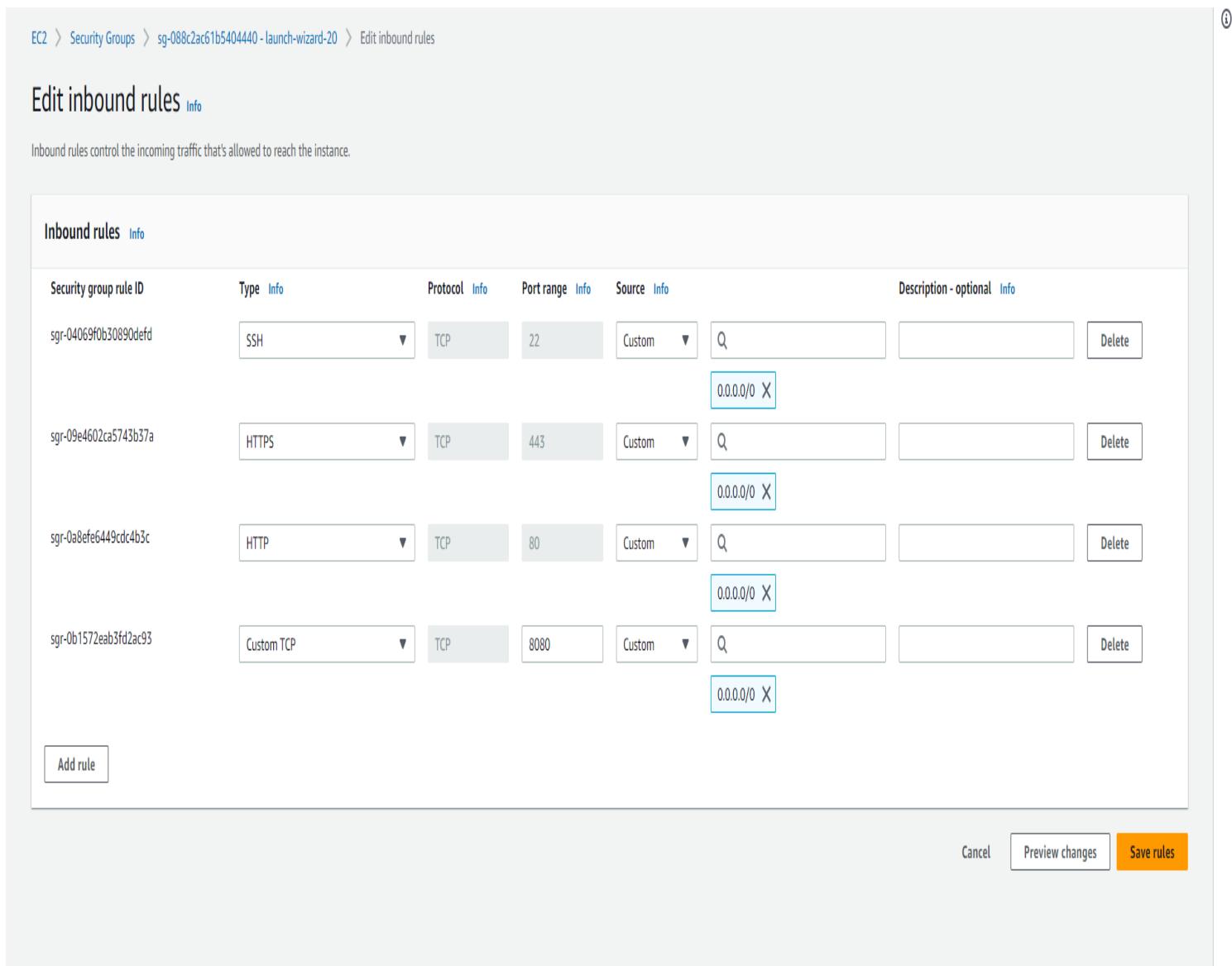
Edit inbound rules Info

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

Inbound rules <small>Info</small>	Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>	
sgr-04069f0b30890defd	SSH	TCP	22	Custom ▾	Q 0.0.0.0/0 X	Delete
sgr-09e4602ca5743b37a	HTTPS	TCP	443	Custom ▾	Q 0.0.0.0/0 X	Delete
sgr-0a8efe6449cdc4b3c	HTTP	TCP	80	Custom ▾	Q 0.0.0.0/0 X	Delete
sgr-0b1572eab3fd2ac93	Custom TCP	TCP	8080	Custom ▾	Q 0.0.0.0/0 X	Delete

Add rule

Cancel Preview changes Save rules



Installation of Java & Jenkins

- After this, connect the instance & using the below mentioned commands, install **Java & Jenkins** on your cmd.

```
15:~$ history
 1 exit
 2 ls
 3 sudo apt update
 4 sudo apt install openjdk-11-jdk -y
 5 curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee    /usr/share/keyrings/jenkins-keyring.asc > /dev/null
 6 echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]      https://pkg.jenkins.io/debian-stable binary/ | sudo tee    /etc/apt/sources.list.d/
jenkins.list > /dev/null
 7 sudo apt-get update
 8 sudo apt-get install fontconfig openjdk-11-jre
 9 sudo apt-get install jenkins
10 java --version
```

Enabling Jenkins

- After successful installation of Java & Jenkins in the system, it is the time to enable Jenkins for use & checking its running status.

```
ubuntu@ip-172-31-6-215:~ % + %
Synchronizing state of jenkins.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable jenkins
ubuntu@ip-172-31-6-215:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor pre
   Active: active (running) since Mon 2023-07-17 12:30:39 UTC; 1min 52s ago
     Main PID: 5114 (java)
        Tasks: 36 (limit: 2349)
       Memory: 368.5M
          CPU: 50.087s
         CGroup: /system.slice/jenkins.service
             └─5114 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/jenkins/jenkins.war

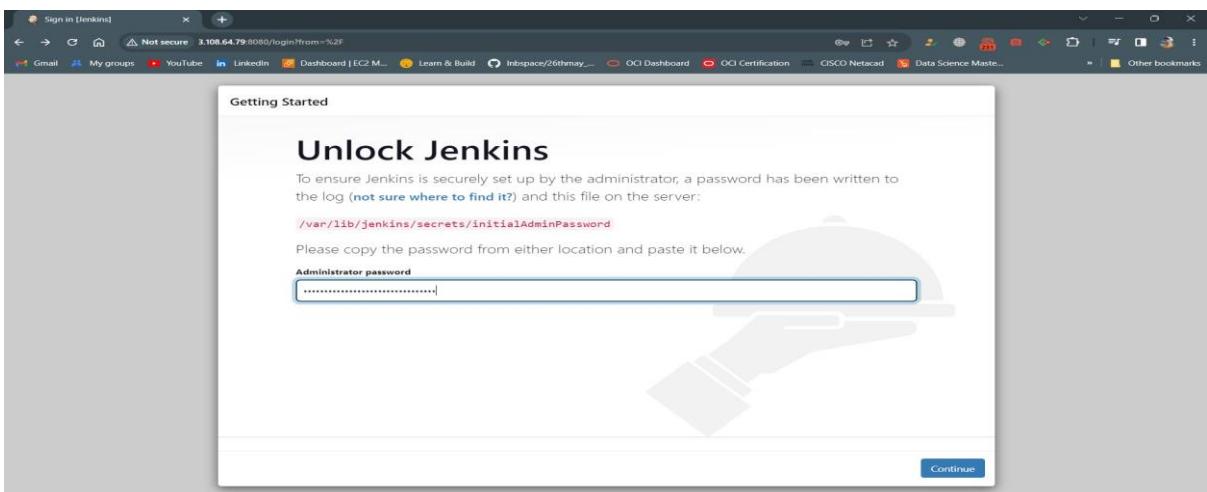
Jul 17 12:29:57 ip-172-31-6-215 jenkins[5114]: 282769c175a34cc583976170acdbJul 17 12:29:57 ip-172-31-6-215 jenkins[5114]: This may also be found at: /Jul 17 12:29:57 ip-172-31-6-215 jenkins[5114]: *****
Jul 17 12:29:57 ip-172-31-6-215 jenkins[5114]: *****
Jul 17 12:30:39 ip-172-31-6-215 jenkins[5114]: 2023-07-17 12:30:39.388+0000Jul 17 12:30:39 ip-172-31-6-215 jenkins[5114]: 2023-07-17 12:30:39.414+0000Jul 17 12:30:39 ip-172-31-6-215 systemd[1]: Started Jenkins Continuous InteJul 17 12:30:40 ip-172-31-6-215 jenkins[5114]: 2023-07-17 12:30:40.440+0000Jul 17 12:30:40 ip-172-31-6-215 jenkins[5114]: 2023-07-17 12:30:40.443+0000Ubuntu@ip-172-31-6-215:~$ history
1 exit
2 ls
3 sudo apt update
4 sudo apt install openjdk-11-jdk -y
5 curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee    /usr/share/keyrings/jenkins-keyring.asc > /dev/null
6 echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]      https://pkg.jenkins.io/debian-stable binary/ | sudo tee    /etc/apt/sources.list.d/
jenkins.list > /dev/null
7 sudo apt-get update
8 sudo apt-get install fontconfig openjdk-11-jre
9 sudo apt-get install jenkins
10 java --version
11 sudo systemctl start jenkins
12 sudo systemctl enable jenkins
13 sudo systemctl status jenkins
14 history
ubuntu@ip-172-31-6-215:~$ |
```

Setting Up Jenkins

- After enabling Jenkins, redirect to a new tab in a browser & search :- <public ip:8080> to open jenkins.

Note : Copy the public IPV4 address from your built AWS EC2 Instance & use in the place of public ip.

- Run “`sudocat var/lib/jenkins/secrets/initialAdminPassword`” in your instance and copy paste the password in the ‘Administrator password’ field and hit Continue. Select “Install Suggested plugIns ” box and wait for successful installation.



```
ubuntu@ip-172-31-6-215:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
282769c175a34cc583976170acdbdd03
ubuntu@ip-172-31-6-215:~$ |
```

Getting Started with Jenkins

- Now fill up the necessary credentials to create first admin user on Jenkins.

Note : Save the “Jenkins URL” showing on Instance Configuration page for future use.

- Complete the Jenkins Setup.

The screenshot shows the Jenkins Setup Wizard on a Windows desktop. The title bar says "Setup Wizard [Jenkins]". The address bar shows "Not secure 3.108.64.79:8080". The main window is titled "Create First Admin User". It contains five input fields: "Username" (aadii_39), "Password" (redacted), "Confirm password" (redacted), "Full name" (Aditya Kumar Verma), and "E-mail address" (redacted). At the bottom left is the note "Jenkins 2.401.2", and at the bottom right are two buttons: "Skip and continue as admin" and "Save and Continue".

The screenshot shows the Jenkins Setup Wizard on a Windows desktop. The title bar says "Setup Wizard [Jenkins]". The address bar shows "Not secure 3.108.64.79:8080". The main window is titled "Instance Configuration". It has a single input field "Jenkins URL" containing "http://3.108.64.79:8080/". Below it is a detailed description of what the Jenkins URL is used for. At the bottom right is a "Save and Continue" button.

The screenshot shows the Jenkins Setup Wizard on a Windows desktop. The title bar says "Setup Wizard [Jenkins]". The address bar shows "Not secure 3.108.64.79:8080". The main window is titled "Jenkins is ready!". It displays the message "Your Jenkins setup is complete." and a "Start using Jenkins" button. At the bottom right is a "Finish" button.

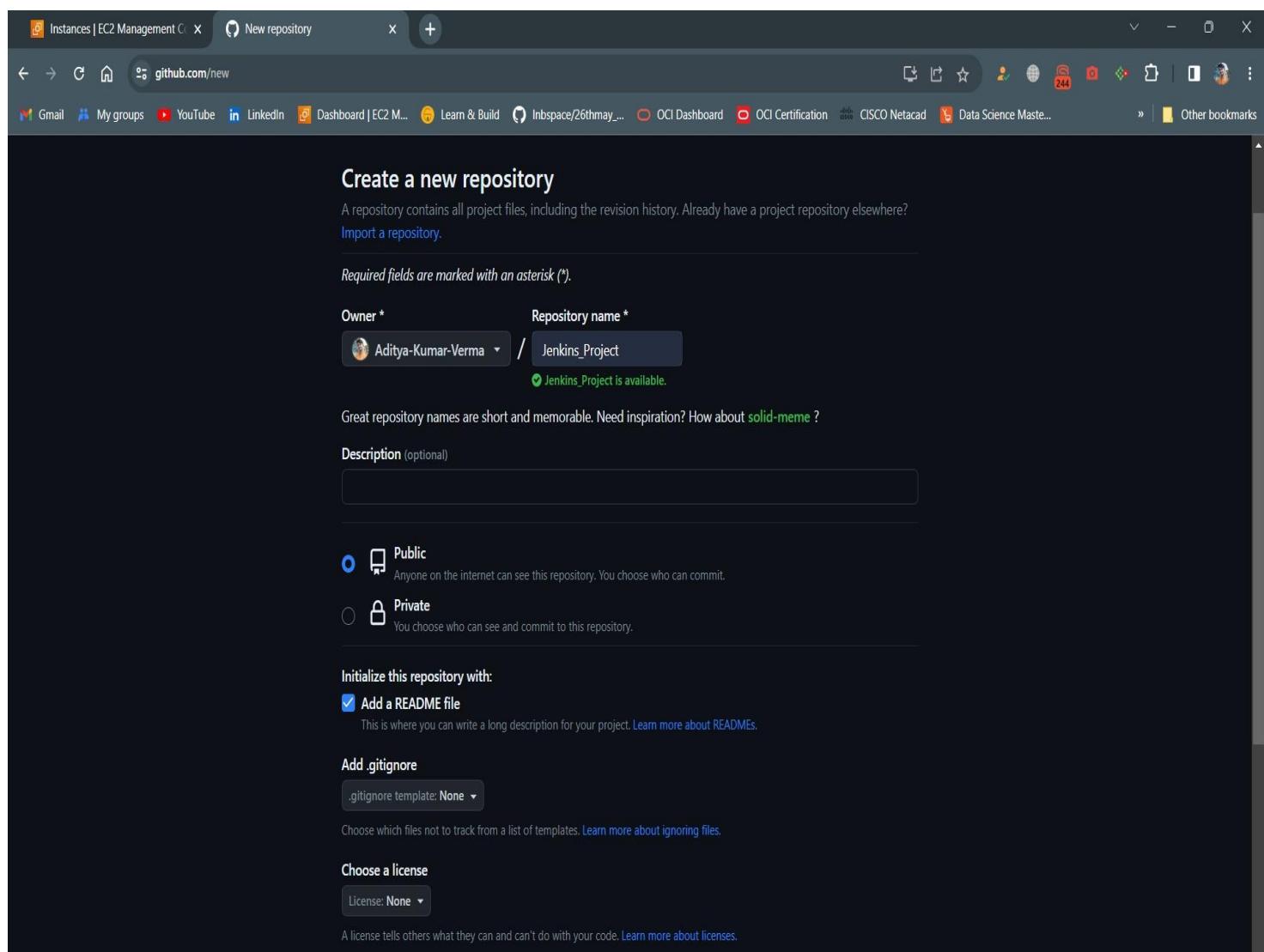
Jenkins Dashboard

The screenshot shows a browser window with the Jenkins dashboard at 3.108.64.79:8080. The dashboard includes:

- Left sidebar:** Links for 'New Item', 'Build Queue' (empty), 'Build Executor Status' (1 Idle, 2 Idle), 'People', 'Build History', 'Manage Jenkins', and 'My Views'.
- Welcome section:** 'Welcome to Jenkins!' message, 'Create a job' button, and 'Start building your software project' link.
- Central area:** 'Set up a distributed build' section with 'Set up an agent' and 'Configure a cloud' buttons, and a 'Learn more about distributed builds' link.
- Bottom right:** 'REST API' and 'Jenkins 2.401.2' links.

Creating a New Git Repository

- Simultaneously create a new Git repository for the project. And save the repository URL for further use.



Creating a New project for CI Pipelining

- **Step 1:** Select New item and make a new Build job (Jenkins_Project).

The screenshot shows a web browser window titled 'New Item [Jenkins]' at the URL '3.108.64.79:8080/view/all/new.job'. The address bar includes a 'Not secure' warning. The page displays a form for creating a new Jenkins item. The first field, 'Enter an item name', contains the value 'Jenkins_Project' and is marked as a 'Required field'. Below this, a list of project types is shown with their icons and descriptions:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.

At the bottom of the form, there is a note: 'If you want to create a new item from other existing, you can use this option:' followed by a 'Copy from' button and a dropdown menu labeled 'Type to autocomplete' with an 'OK' button.

- **Step 2:** Select Git in Source Code Management section. Here, paste the Repository URL of the GitHub (mentioned earlier to save).

The screenshot shows the Jenkins Project Configuration page for a project named "Jenkins_Project". The "Source Code Management" section is selected. Under the "Git" tab, the "Repository URL" field contains the value "https://github.com/Aditya-Kumar-Verma/Jenkins_Project". The "Branches to build" section has a "Branch Specifier" field containing the value "*/*main". At the bottom, there are "Save" and "Apply" buttons.

Jenkins_Project Config [Jenkins] + ▲ Not secure 3.108.64.79:8080/job/Jenkins_Project/configure

Gmail My groups YouTube LinkedIn Dashboard | EC2 M... Learn & Build Inbspace/26thmay_... OCI Dashboard OCI Certification CISCO Netacad Data Science Maste... Other bookmarks

Dashboard > Jenkins_Project > Configuration

Configure Source Code Management

General None

Source Code Management Git

Build Triggers

Build Environment

Build Steps

Post-build Actions

Repositories ?

Repository URL ?

https://github.com/Aditya-Kumar-Verma/Jenkins_Project

Credentials ?

- none -

Add Advanced

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/*main

Add Branch

Save Apply

- **Step 3:** Adjust the Build Triggers section. And also mention the Upstream & Down-stream projects.

Screenshot of the Jenkins Project Configuration page showing the Build Triggers section.

Configure Jenkins_Project

Build Triggers

General settings:

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?

Projects to watch:

Jenkins_Upstream.

Trigger options:

- Trigger only if build is stable
- Trigger even if the build is unstable
- Trigger even if the build fails
- Always trigger, even if the build is aborted

GitHub hook trigger for GITScm polling ?

Periodic triggers:

- Build periodically ?
- Poll SCM ?

Build Environment

Environment settings:

- Delete workspace before build starts
- Use secret text(s) or file(s) ?
- Add timestamps to the Console Output
- Inspect build log for published build scans
- Terminate a build if it's stuck

Buttons at the bottom:

Save Apply

- **Step 4:** Click Save.



Screenshot of a web browser showing the Jenkins Project page. The URL is 3.108.64.79:8080/job/Jenkins_Project/. The page title is "Jenkins Project [Jenkins]". The main content area shows the "Project Jenkins_Project" configuration page with sections for Upstream Projects (Jenkins_Upstream) and Downstream Projects (Jenkins_Downstream). The Build History section lists two builds: #2 (17-Jul-2023, 1:50 PM) and #1 (17-Jul-2023, 1:42 PM). The Permalinks section lists four links: Last build (#2), Last stable build (#2), Last successful build (#2), and Last completed build (#2). The bottom right corner of the page footer shows "REST API" and "Jenkins 2.401.2".

Jenkins Project [Jenkins] Not secure 3.108.64.79:8080/job/Jenkins_Project/

Gmail My groups YouTube LinkedIn Dashboard | EC2 M... Learn & Build Inbspace/26thmay... OCI Dashboard OCI Certification CISCO Netacad Data Science Maste... Other bookmarks

Jenkins Search (CTRL+K) Aditya Kumar Verma log out

Dashboard > Jenkins_Project >

Status Project Jenkins_Project

</> Changes Add description

Workspace Disable Project

Build Now

Configure

Delete Project

Rename

Upstream Projects

Jenkins_Upstream

Downstream Projects

Jenkins_Downstream

Build History trend

Filter builds... /

#2 17-Jul-2023, 1:50 PM

#1 17-Jul-2023, 1:42 PM

Last build (#2), 4.3 sec ago
Last stable build (#2), 4.3 sec ago
Last successful build (#2), 4.3 sec ago
Last completed build (#2), 4.3 sec ago

Atom feed for all Atom feed for failures

REST API Jenkins 2.401.2

- **Step 5:** Create a new WebHook in GitHub.
Paste the Jenkins URL (saved earlier in project) in the Payload URL field and add //github-webhook at the end of URL.

The screenshot shows a browser window with the GitHub interface. The left sidebar has a dark theme with various project settings like General, Access, Collaborators, and Webhooks. The 'Webhooks' section is currently selected. The main content area is titled 'Webhooks / Add webhook'. It explains that a POST request will be sent to the specified URL with event details. The 'Payload URL' field contains 'http://3.108.64.79:8080/github-webhook'. The 'Content type' dropdown is set to 'application/json'. There is a large empty text input field for 'Secret'. Below, under 'Which events would you like to trigger this webhook?', there are three radio button options: 'Just the push event.' (selected), 'Send me everything.', and 'Let me select individual events.'. At the bottom, a checked checkbox labeled 'Active' with the note 'We will deliver event details when this hook is triggered.'

Check the working of Webhook

- To check the proper working of webhook, commit a file in github and check it in jenkins workspace.

(BEFORE COMMITTING)

Instances | EC2 Management C × Workspace of Jenkins_Project × Aditya-Kumar-Verma/Jenkins_P × Jenkins [Jenkins] × +

Not secure 3.108.64.79:8080/job/Jenkins_Project/ws/

Gmail My groups YouTube LinkedIn Dashboard | EC2 M... Learn & Build Inbspace/26thmay... OCI Dashboard OCI Certification CISCO Netacad Data Science Maste... Other bookmarks

Jenkins

Search (CTRL+K) Aditya Kumar Verma log out

Dashboard > Jenkins_Project > Workspace

Status

Changes

Workspace

Wipe Out Current Workspace

Build Now

Configure

Delete Project

Rename

Build History trend

Filter builds... /

#4 17-Jul-2023, 2:53 PM

#3 17-Jul-2023, 2:47 PM

#2 17-Jul-2023, 1:50 PM

#1 17-Jul-2023, 1:42 PM

Atom feed for all Atom feed for failures

- Click build now in your project and come to workspace and refresh the tab.

(AFTER COMMITTING)

The screenshot shows a web browser window with the following details:

- Address Bar:** Jenkins_Project/ at main / Jenkins [Jenkins]
- Toolbar:** Includes standard browser controls like back, forward, search, and refresh, along with a tab for "Not secure" and the URL 3.108.64.79:8080/job/Jenkins_Project/ws/.
- Bookmark Bar:** Shows various bookmarks including Gmail, YouTube, LinkedIn, Dashboard | EC2 M..., Learn & Build, Inbspace/26thmay..., OCI Dashboard, OCI Certification, CISCO Netacad, Data Science Maste..., and Other bookmarks.
- User Profile:** Aditya Kumar Verma and log out link.
- Page Content:**
 - Left Sidebar:** Buttons for Status, Changes, Workspace (which is selected), Wipe Out Current Workspace, Build Now, Configure, Delete Project, and Rename.
 - Workspace Area:** Title "Workspace of Jenkins_Project on Built-In Node". A "git" folder contains files: After_Webhook (17-Jul-2023, 2:55:20 PM, 42 B), File1 (17-Jul-2023, 2:47:46 PM, 76 B), and README.md (17-Jul-2023, 1:42:37 PM, 17 B). A link "(all files in zip)" is also present.
 - Build History:** A table showing five builds (#1 to #5) with their dates: 17-Jul-2023, 1:42 PM; 17-Jul-2023, 1:50 PM; 17-Jul-2023, 2:47 PM; 17-Jul-2023, 2:53 PM; and 17-Jul-2023, 2:55 PM. Each build row has a green circular icon with a checkmark.
 - Bottom Links:** Atom feed for all and Atom feed for failures.

Configuring the Jenkins job

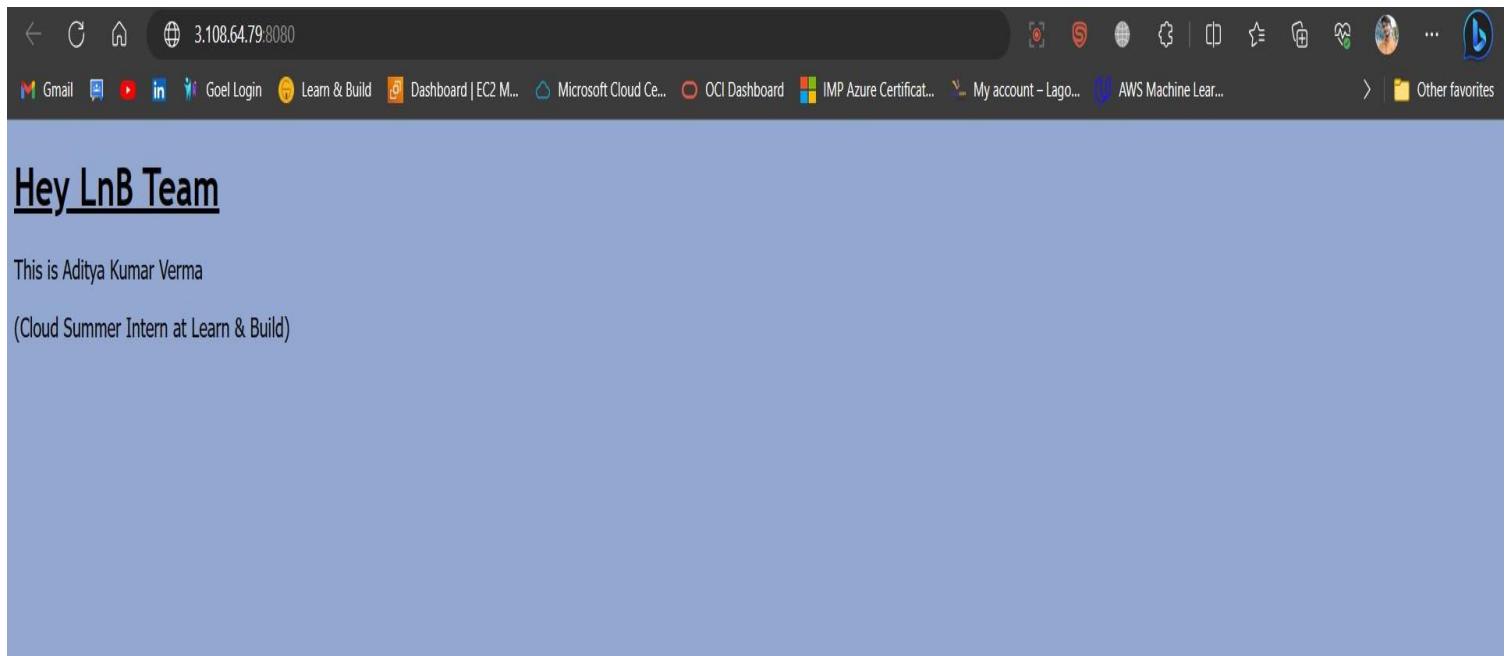
- Head back to the Configure section of the job; in the Execute Shell present in the Build Steps section, enter the following commands.

The screenshot shows a web browser window with three tabs open: 'Instances | EC2 Management', 'Jenkins_Project Config [Jenkins]', and 'Aditya-Kumar-Verma/Jenkins_...'. The main content area is the Jenkins configuration page for the 'Jenkins_Project' job. The 'Configuration' tab is selected. On the left, there's a sidebar with 'General', 'Source Code Management', 'Build Triggers', 'Build Environment' (which is highlighted), and 'Post-build Actions'. The 'Build Steps' section is currently active. It contains one step: 'Execute shell'. The 'Command' field contains the following script:
sudo -s yum install httpd -y
sudo systemctl enable --now httpd
sudo cp Intro.html /var/www/html/Intro.html

CI Pipelining Demonstration

- **Step 1:** Now open new tab and search for: <public ip:8080> to open your github project file.

Note : Copy the public IPV4 address from your built AWS EC2 Instance & use in the place of public ip.



- **Step 2:** To check the successful functioning of CI pipelines, edit the github file (here Intro.html) and build the jenkins project again, by clicking on Build Now.



A screenshot of a GitHub code editor showing the file `Intro.html`. The code contains HTML and CSS, including a background color and several `` tags. A blue rectangular selection highlights the first few lines of the code. The GitHub interface includes tabs for 'Edit' and 'Preview', and a toolbar with 'Cancel changes' and 'Commit changes...' buttons. The browser's address bar shows the URL `github.com/Aditya-Kumar-Verma/Jenkins_Project/edit/main/Intro.html`.

```

1  <!DOCTYPE html>
2  <html>
3  <body style="background-color:#92A8D1;">
4
5  <h1 style="font-family:Trebuchet MS;"><u>Hey LnB Team</u></h1>
6  <p style="font-family:Tahoma;">This is Aditya Kumar Verma</p>
7  <p style="font-family:Tahoma;">(Cloud Summer Intern at Learn & Build)</p>
8
9  <br>
10 <h1 style="font-family:Trebuchet MS;"><u>To Connect Visit</u> => </h1>
11 <a href="https://www.linkedin.com/in/aditya-kumar-verma3/">Linkedin Id</a>
12 <br> https://www.linkedin.com/in/aditya-kumar-verma3/<br>
13 <br>
14 <a href="https://www.instagram.com/aadii_39/">Instagram Id</a>
15 <br> https://www.instagram.com/aadii_39/ <br>
16 <br>
17 <br>
18 <br>
19 <br>
20 <br>
21 <br>
22 <br>
23 <br>
24 <br>
25 <br>
26 <br>
27 <br>
28 </body>
29 </html>

```

Use `Control + Shift + m` to toggle the `tab` key moving focus. Alternatively, use `esc` then `tab` to move to the next interactive element on the page.

- **Step 3:** Check the Build history which shows the auto integration of the repository.



Screenshot of a web browser showing the Jenkins Project page. The URL in the address bar is 3.108.64.79:8080/job/Jenkins_Project/. The page displays the Jenkins Project dashboard with sections for Upstream Projects (Jenkins_Upstream) and Downstream Projects (Jenkins_Downstream). The Build History section shows the following build details:

Build #	Timestamp
#14	(pending—In the quiet period. Expires in 0.31 sec)
#6	17-Jul-2023, 3:07 PM
#5	17-Jul-2023, 2:55 PM
#4	17-Jul-2023, 2:53 PM
#3	17-Jul-2023, 2:47 PM
#2	17-Jul-2023, 1:50 PM
#1	17-Jul-2023, 1:42 PM

At the bottom of the page, there are links for Atom feed for all and Atom feed for failures.

Step 3: Finally refresh the tab in which the project file has been opened.

IT UPDATES !!!

A screenshot of a web browser window. The address bar shows the URL 3.108.64.79:8080. The page content is a blue-tinted message:

Hey LnB Team

This is Aditya Kumar Verma
(Cloud Summer Intern at Learn & Build)

To Connect Visit =>

[Linkedin Id](#)
<https://www.linkedin.com/in/aditya-kumar-verma3/>

[Instagram Id](#)
https://www.instagram.com/aadii_39/

Conclusion

This shows the demonstrative approach
to use Jenkins for CI Pipelining using
Github