cprime

AWS Multi-Region Setup with Git and Jenkins Pipeline

NAME : T MANOJ

EMPID : LYAKE2KHS

Introduction to Multi-Region AWS GIT-JENKINS Setup

Overview

This documentation details a comprehensive continuous integration and continuous deployment (CI/CD) pipeline implemented across two AWS regions. The setup leverages multiple AWS services and open-source tools to create an efficient development workflow that automates code building, testing, and deployment processes.

Services and Tools Used

Amazon Web Services (AWS)

- **EC2 Instances**: Virtual servers in the cloud used to host both the code repository and CI/CD server.
- **Security Groups**: Virtual firewalls that control inbound and outbound traffic to our instances.
- **Multi-Region Architecture**: Utilizing instances in both Ohio and North Virginia regions for geographic distribution and resilience.

Operating Systems

- **Amazon Linux**: A Linux distribution provided by AWS, optimized for the EC2 environment. Used in the Ohio region for the Git repository.
- **Ubuntu**: A popular Debian-based Linux distribution used in the North Virginia region for running Jenkins.

Version Control

- **Git**: Distributed version control system used to track changes in source code during software development.
- **GitHub**: Cloud-based hosting service for Git repositories, facilitating collaboration and code sharing.

CI/CD Tools

- **Jenkins**: An open-source automation server that enables developers to build, test, and deploy their applications.
- **Webhooks**: Used to trigger Jenkins jobs automatically when code is pushed to the GitHub repository.

File Transfer Tools

- **WinSCP**: A free and open-source SFTP, FTP, WebDAV, and SCP client used to securely transfer files between local and remote systems.

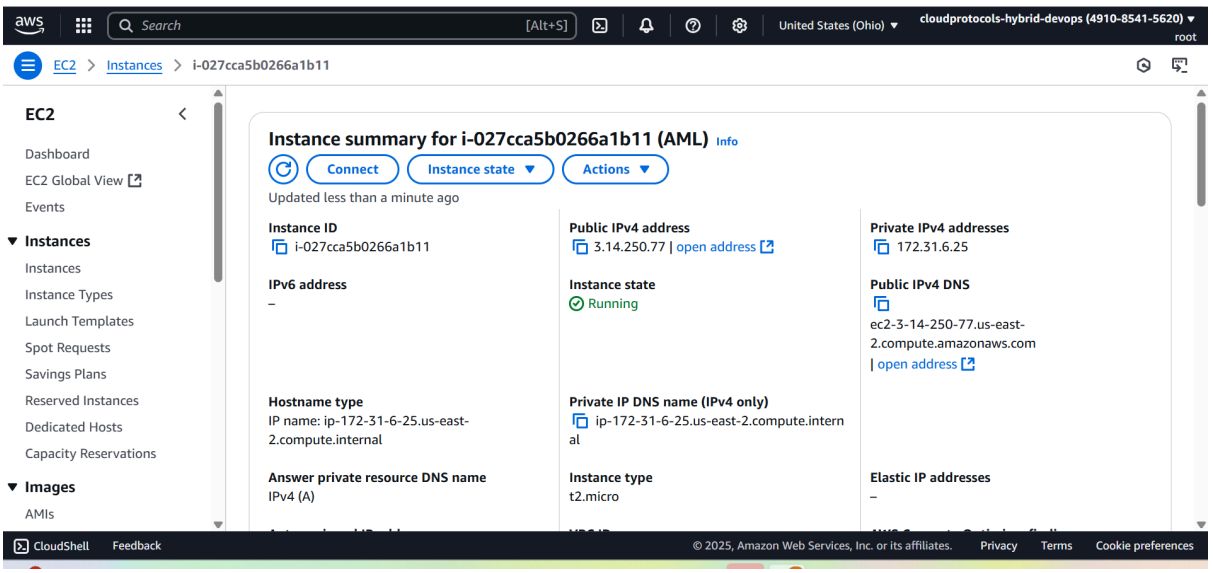
System Architecture

The architecture consists of two main components:

1. **Source Code Repository (Ohio Region)**
    - Amazon Linux EC2 instance
    - Git server hosting the application code
    - Acts as the origin for code pushes to GitHub
2. **CI/CD Server (North Virginia Region)**
    - Ubuntu EC2 instance
    - Jenkins server configured with two jobs:
        - GitBuilder: Triggered by GitHub webhooks to build and test code
        - GitDeploy: Executes after successful GitBuilder job to deploy the application

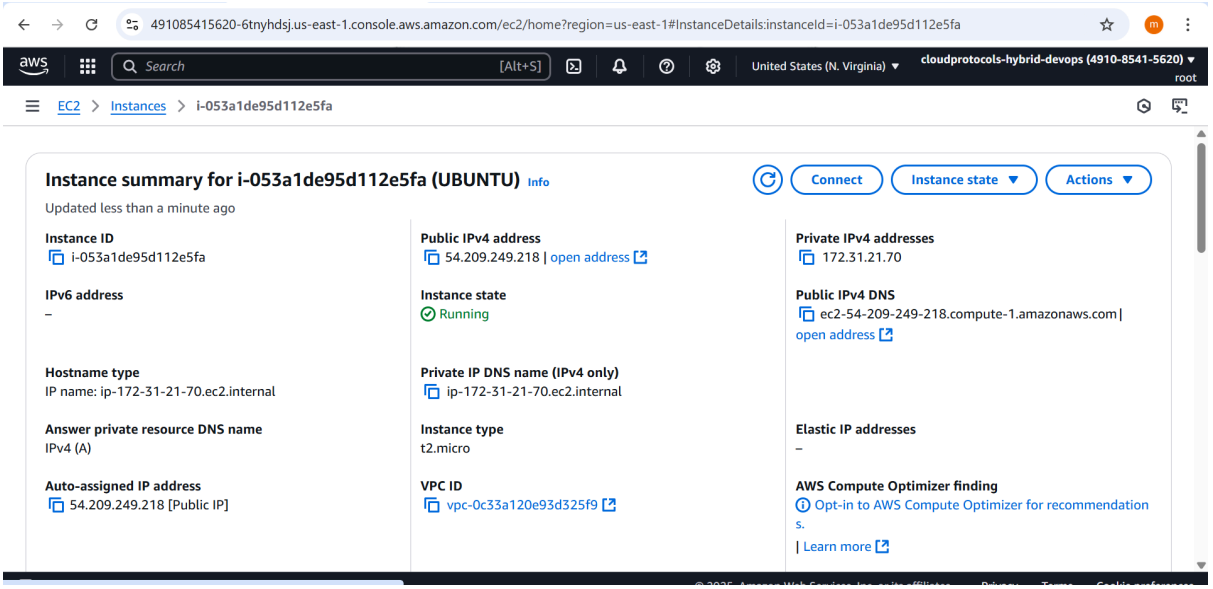
Infrastructure Overview

1. **Region 1: Ohio**
    - Operating System: Amazon Linux
    - Purpose: Source code repository (Git server)



2. **Region 2: North Virginia**

    - Operating System: Ubuntu
    - Purpose: CI/CD server (Jenkins)



Amazon Linux Setup (Ohio Region)

1. Git Installation and Repository Setup

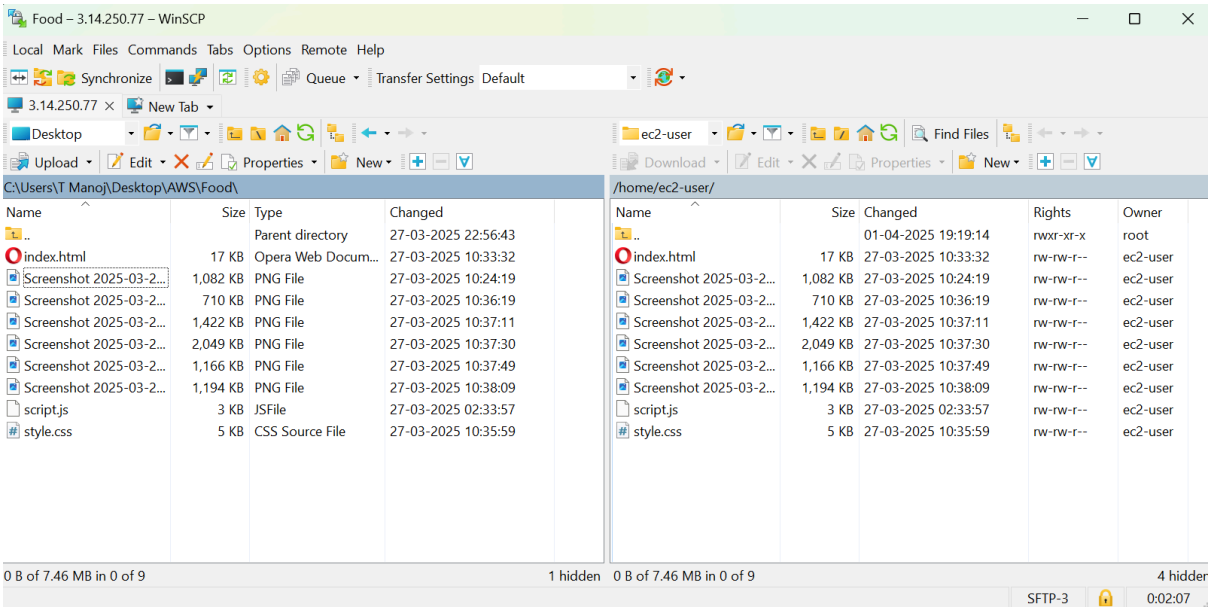
```
# Install Git
sudo yum update -y
sudo yum install git -y


# Verify Git installation
git --version
```

```
Installed:
  git-2.47.1-1.amzn2023.0.2.x86_64
  git-core-2.47.1-1.amzn2023.0.2.x86_64
  git-core-doc-2.47.1-1.amzn2023.0.2.noarch
  perl-Error-1:0.17029-5.amzn2023.0.2.noarch
  perl-File-Find-1.37-477.amzn2023.0.6.noarch
  perl-Git-2.47.1-1.amzn2023.0.2.noarch
  perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64
  perl-lib-0.65-477.amzn2023.0.6.x86_64

Complete!
[ec2-user@ip-172-31-6-25 ~]$
```

2. File Transfer from Local Machine

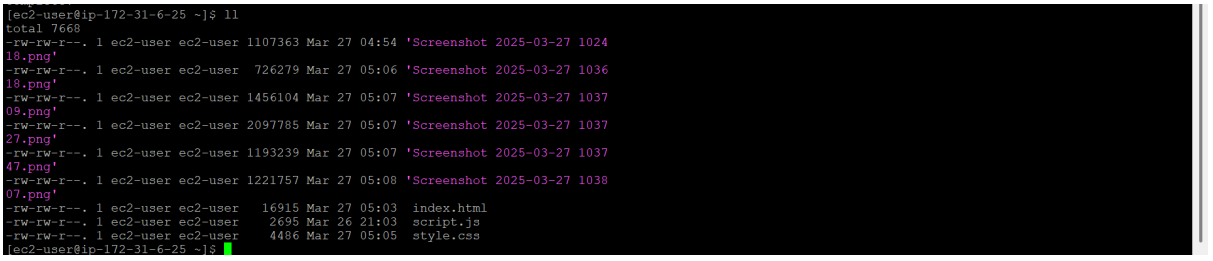
- Used WinSCP to copy files from local system to Amazon Linux instance

- Transferred project files to `/path/to/project` directory



3. Git Repository Creation

```
# Navigate to project directory
cd /path/to/project
```




```
# Initialize Git repository
git init
```

```
[ec2-user@ip-172-31-6-25 ~]$ ll
total 7668
-rw-rw-r--. 1 ec2-user ec2-user 1107363 Mar 27 04:54 'Screenshot 2025-03-27 1024
18.png'
-rw-rw-r--. 1 ec2-user ec2-user  726279 Mar 27 05:06 'Screenshot 2025-03-27 1036
18.png'
-rw-rw-r--. 1 ec2-user ec2-user 1456104 Mar 27 05:07 'Screenshot 2025-03-27 1037
09.png'
-rw-rw-r--. 1 ec2-user ec2-user 2097785 Mar 27 05:07 'Screenshot 2025-03-27 1037
27.png'
-rw-rw-r--. 1 ec2-user ec2-user 1193239 Mar 27 05:07 'Screenshot 2025-03-27 1037
47.png'
-rw-rw-r--. 1 ec2-user ec2-user 1221757 Mar 27 05:08 'Screenshot 2025-03-27 1038
07.png'
-rw-rw-r--. 1 ec2-user ec2-user   16915 Mar 27 05:03  index.html
-rw-rw-r--. 1 ec2-user ec2-user    2695 Mar 26 21:03  script.js
-rw-rw-r--. 1 ec2-user ec2-user    4486 Mar 27 05:05  style.css
[ec2-user@ip-172-31-6-25 ~]$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ec2-user/.git/
```

```
# Add all files to Git
git add .
```

```
# Commit changes
git commit -m "Initial commit"
```

```
[ec2-user@ip-172-31-6-25 ~]$ git add .
[ec2-user@ip-172-31-6-25 ~]$ git commit -m "initial commit"
[master (root-commit) 914a9b9] initial commit
 Committer: EC2 Default User <ec2-user@ip-172-31-6-25.us-east-2.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 13 files changed, 371 insertions(+)
 create mode 100644 .bash_logout
 create mode 100644 .bash_profile
 create mode 100644 .bashrc
 create mode 100644 .ssh/authorized_keys
 create mode 100644 Screenshot 2025-03-27 102418.png
 create mode 100644 Screenshot 2025-03-27 103618.png
 create mode 100644 Screenshot 2025-03-27 103709.png
 create mode 100644 Screenshot 2025-03-27 103727.png
 create mode 100644 Screenshot 2025-03-27 103747.png
 create mode 100644 Screenshot 2025-03-27 103807.png
 create mode 100644 index.html
 create mode 100644 script.js
 create mode 100644 style.css
[ec2-user@ip-172-31-6-25 ~]$
```

```
# Configure remote repository (GitHub)
git remote add origin https://github.com/username/repository.git
```

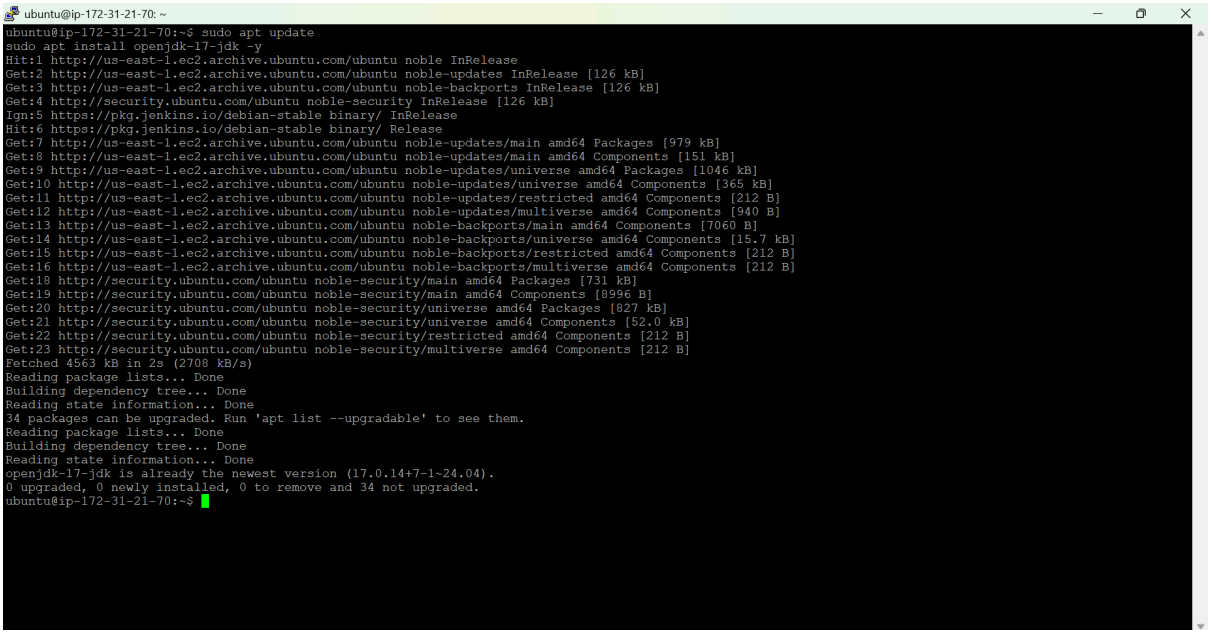
```
[ec2-user@ip-172-31-6-25 ~]$ git remote add origin https://github.com/tupakulamanoj/JENKINS.git
[ec2-user@ip-172-31-6-25 ~]$ git remote -v
origin  https://github.com/tupakulamanoj/JENKINS.git (fetch)
origin  https://github.com/tupakulamanoj/JENKINS.git (push)
```

```
# Push code to remote repository
git push -u origin main
```

Ubuntu Setup (North Virginia Region)

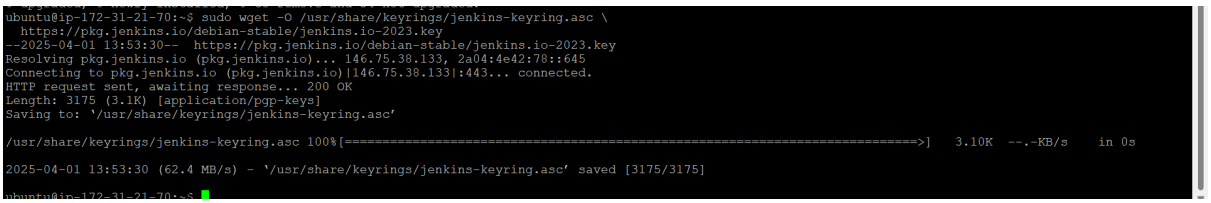
1. Java Installation (Prerequisite for Jenkins)

```
# Update package index
sudo apt update

# Install OpenJDK 17
sudo apt install openjdk-17-jdk -y

# Verify Java installation
java -version
```

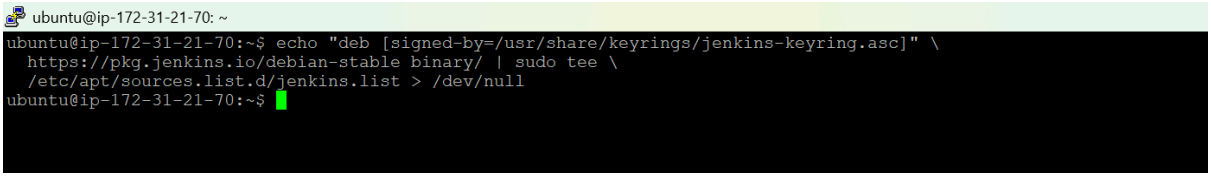


2. Jenkins Installation

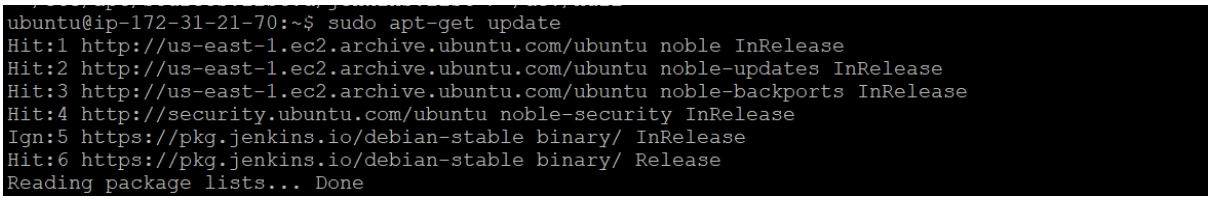
```
# Import Jenkins repository key
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
```



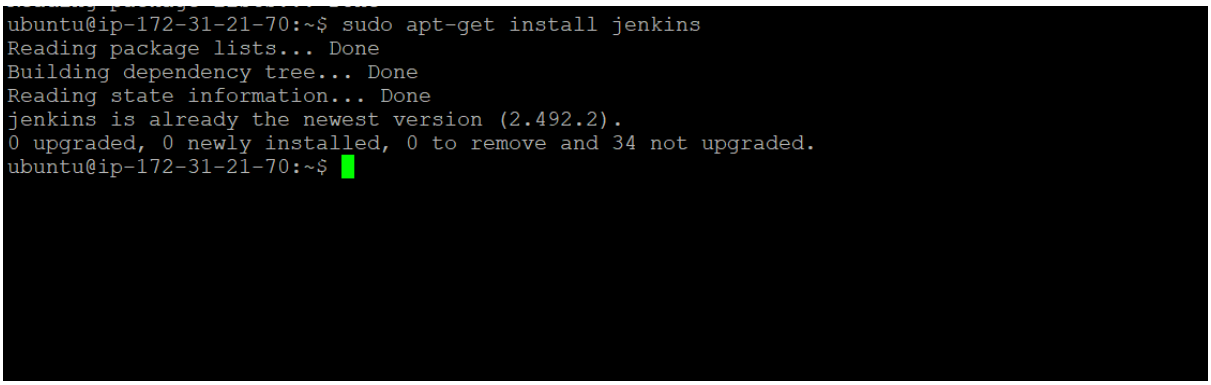

```
# Add Jenkins repository to sources list
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
```



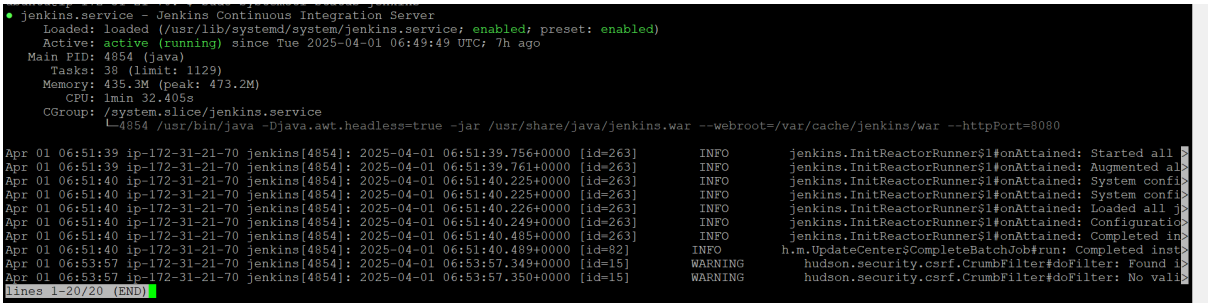

```
# Update package index
sudo apt-get update
```




```
# Install Jenkins
sudo apt-get install jenkins -y
```




```
# Check Jenkins status
sudo systemctl status jenkins
```



3. Jenkins Initial Configuration

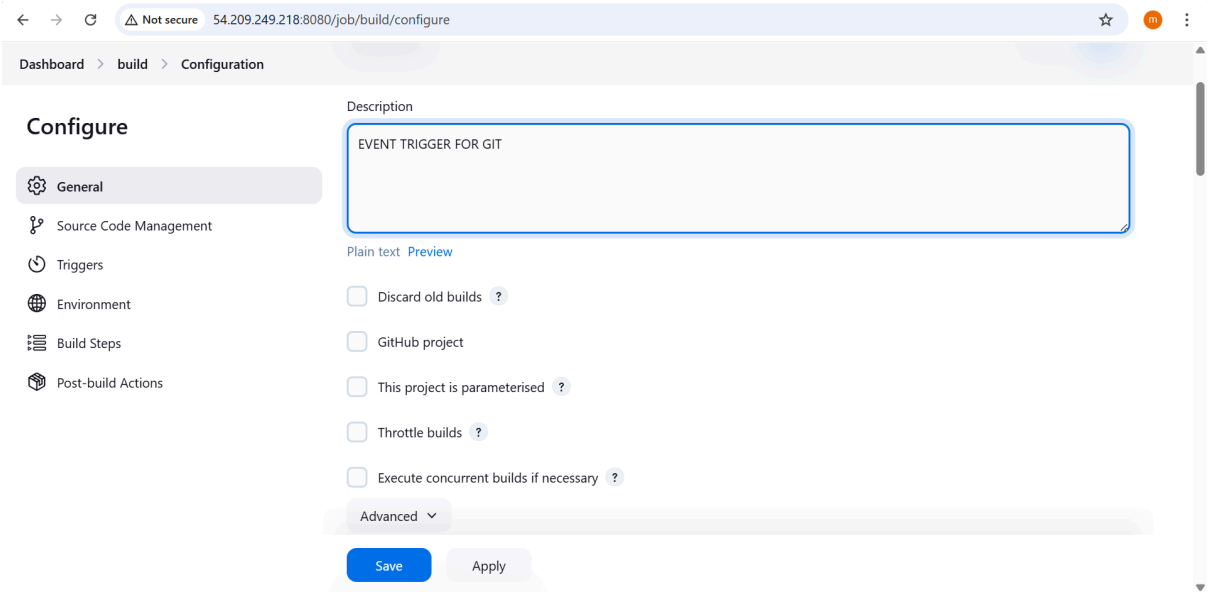
# Get Jenkins initial admin password
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

- Open Jenkins in web browser: `http://<Ubuntu-Instance-IP>:8080`
- Enter the initial admin password
- Install suggested plugins
- Create admin user and complete initial setup

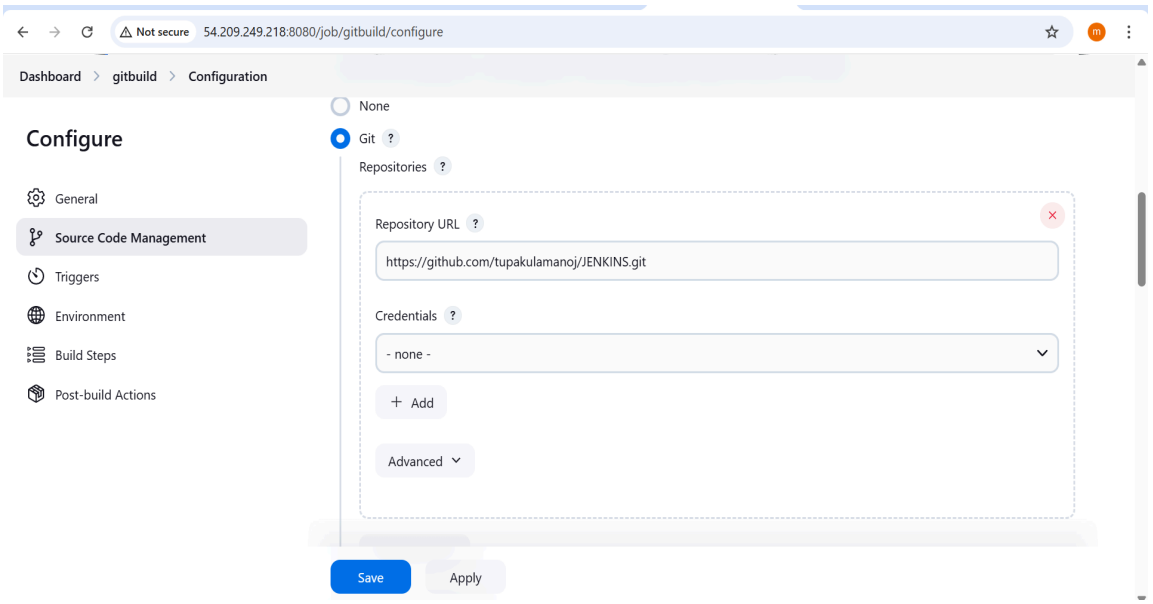
Jenkins Jobs Configuration

1. GitBuilder Job Setup

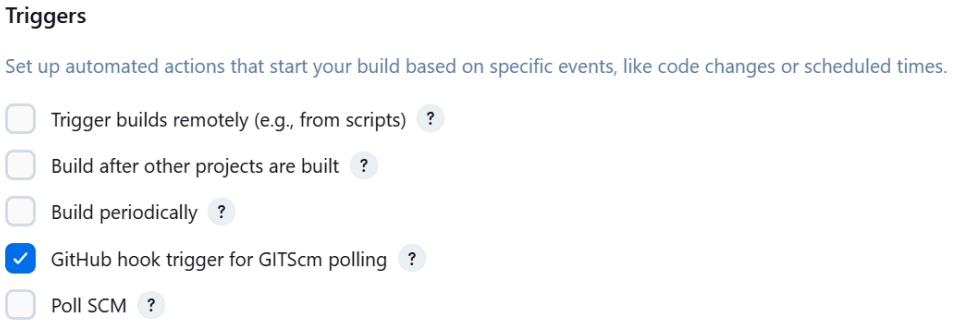
- From Jenkins dashboard, click "New Item"
- Enter "GitBuilder" as job name and select "Freestyle project"



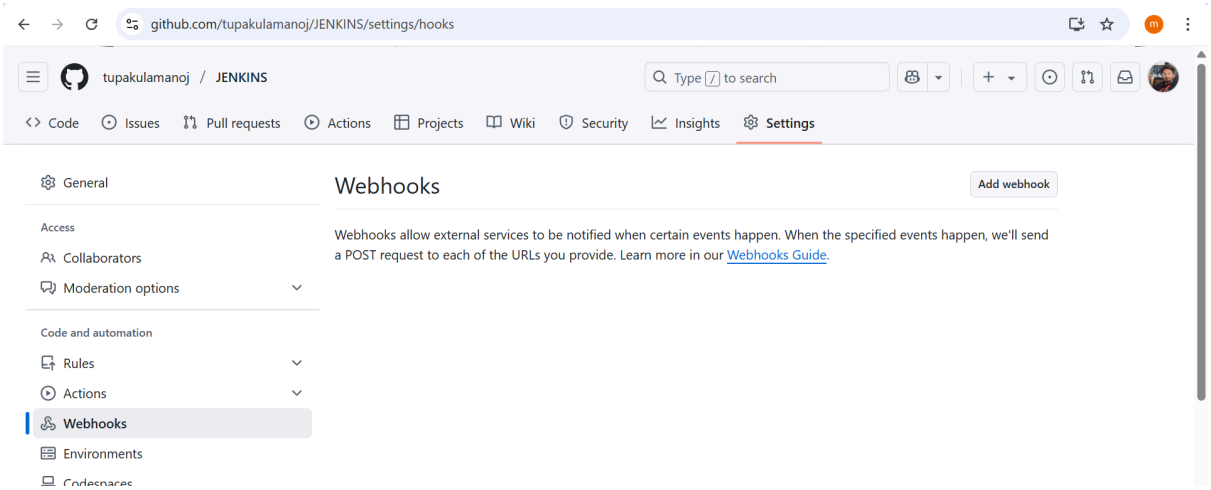
- Configure source code management:
  - Select Git
  - Enter repository URL: https://github.com/username/repository.git

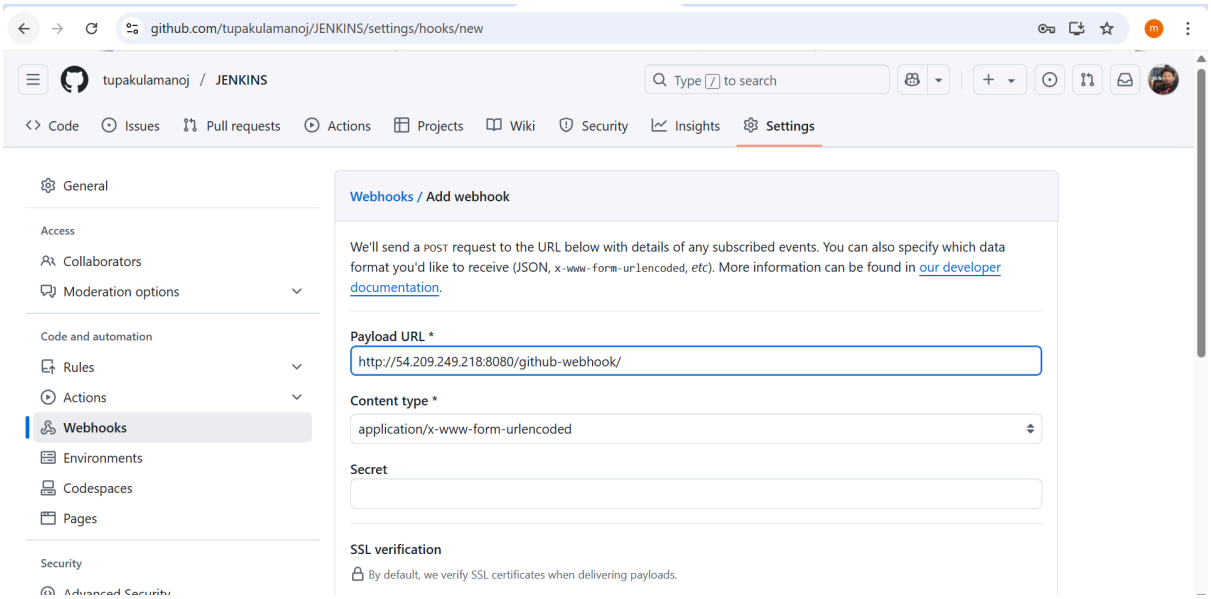


- Configure build triggers:
    - Check "GitHub hook trigger for GITScm polling"



- Configure webhook in GitHub:
    - Go to GitHub repository → Settings → Webhooks → Add webhook
    - Payload URL: `http://<Ubuntu-Instance-IP>:8080/github-webhook/`
    - Content type: `application/json`
    - Events: Select "Just the push event"
    - Save webhook

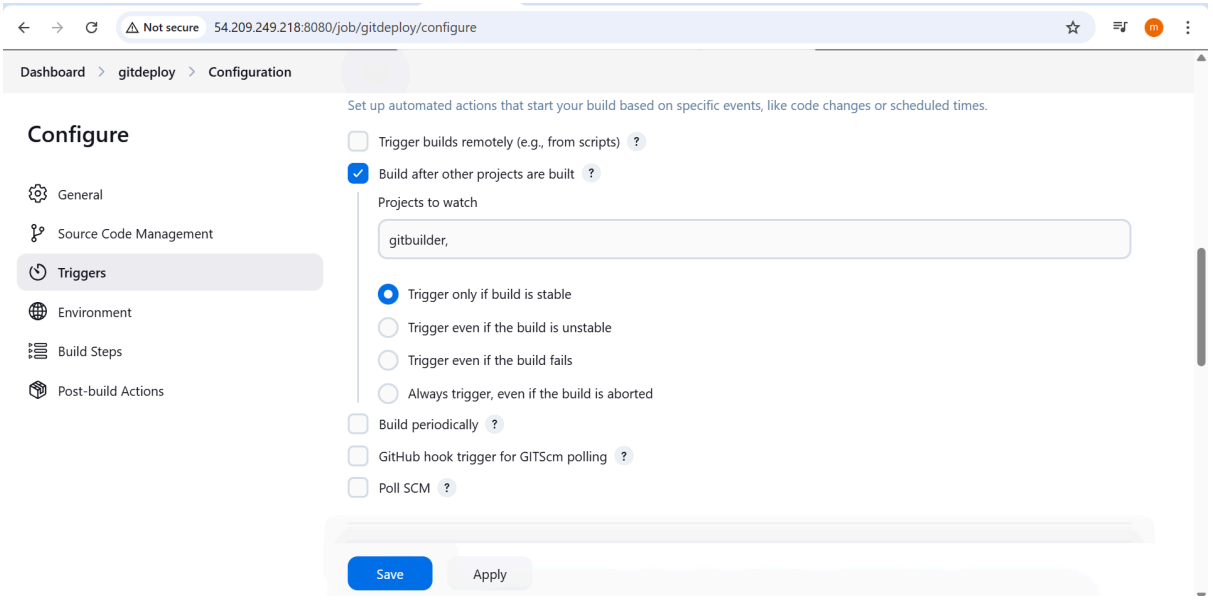




2. GitDeploy Job Setup

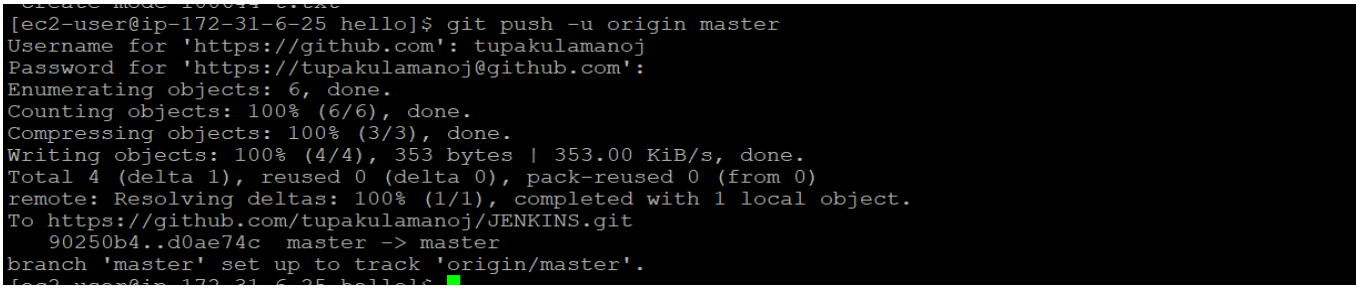
- From Jenkins dashboard, click "New Item"
- Enter "GitDeploy" as job name and select "Freestyle project"
- Configure build triggers:
    - Check "Build after other projects are built"
    - Enter "GitBuilder" in the project name field

Add deployment build steps (e.g., shell commands)

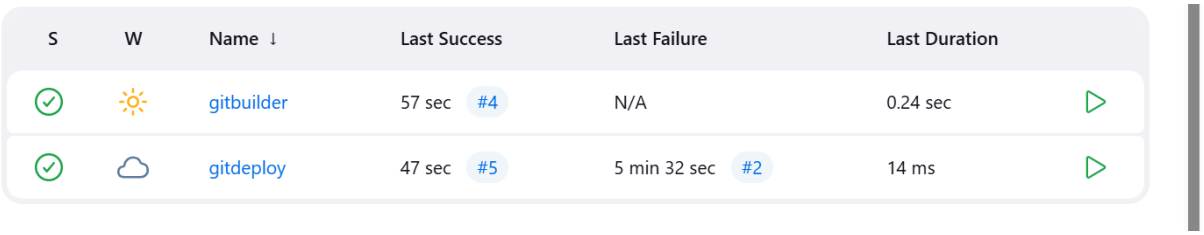


CI/CD Workflow

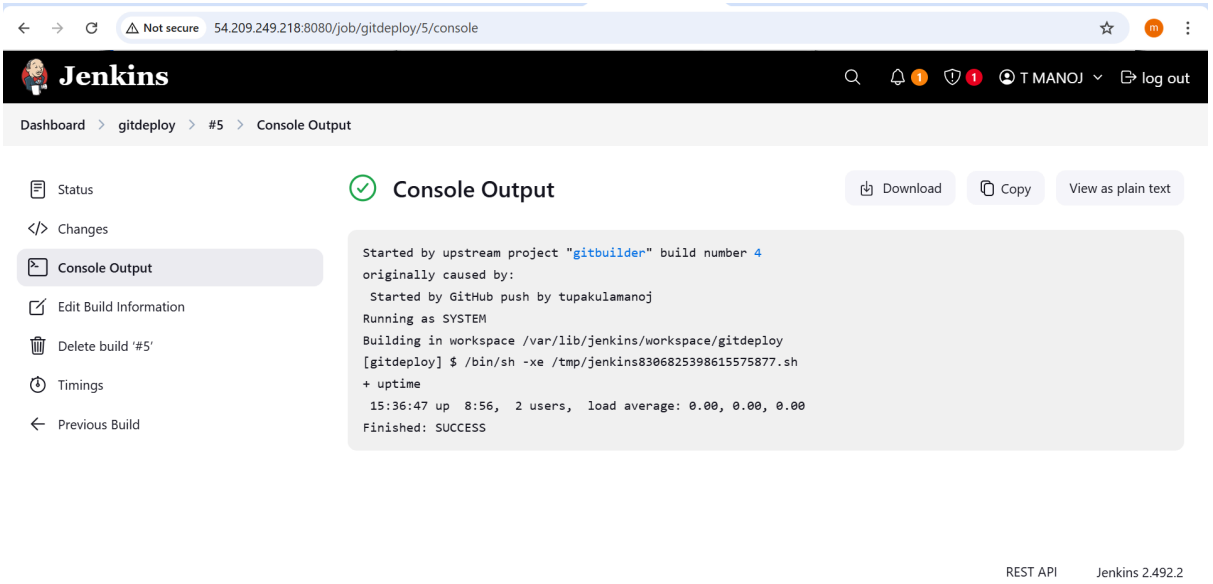
1. Developer pushes code to GitHub repository (from Amazon Linux in Ohio)



2. GitHub webhook triggers the GitBuilder job in Jenkins
3. GitBuilder job:
    - Pulls latest code from GitHub
    - Builds and tests the application



4. After GitBuilder completes successfully, GitDeploy job is automatically triggered
5. GitDeploy job:
    - Deploys the application to the target environment
    - Restarts services if needed



Security Considerations

- Ensure both AWS instances have appropriate security groups configured
- Configure Jenkins security (authentication, authorization, etc.)
- Use SSH keys or secure credentials for deployment