

Project Report

on

Deployment Of CI/CD Pipeline on Cloud



Submitted in partial fulfilment for the award of
**Post Graduate Diploma in High Performance Computing System
Administration from C-DAC ACTS (Pune)**

Under the Guidance of

Mr. Ashutosh Das

Submitted by:

Ruchi Bhutad	PRN: 230340127005
Kishor Yevale	PRN: 230340127014
Bharat Bansode	PRN: 230340127032
Shivshankar Gadeodhe	PRN: 230340127034
Akshay Gaikwad	PRN: 230340127035
Trupti Kokane	PRN: 230340127052



CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Ruchi Bhutad	PRN: 230340127005
Kishor Yevale	PRN: 230340127014
Bharat Bansode	PRN: 230340127032
Shivshankar Gadeodhe	PRN: 230340127034
Akshay Gaikwad	PRN: 230340127035
Trupti Kokane	PRN: 230340127052

have successfully completed their project on
Deployment of CI/CD pipeline on cloud.

Under the Guidance of Mr. Ashutosh Das

Project Guide

Project Supervisor

HOD ACTS

TABLE OF CONTENTS

1. Abstract.....
2. Introduction and Overview of Project.....
3. Deployment of CI/CD pipeline on cloud
4. System Requirement.....
4.1 Software Requirement.....
4.2 Hardware Requirement.....
4.3 User Interface.....
5. System Architecture.....
5.1 Data Flow Diagram.....
5.2 Activity Diagram.....
6. Introduction to Repository.....
7. Git Concepts.....
8. Jenkins server configuration.....
9. Docker Configuration.....
10. Pulling docker image.....
11. Jenkins Installation.....
12. Docker Installation.....
13. Scripts.....
13.1 Python-Flask Script.....
13.2 HTML Script.....
14. Jenkins files configuration.....
15. Test Plan.....
16. Application server.....
17. Results.....
18. Conclusion.....
19. References.....

1. ABSTRACT

Continuous Integration and Continuous Delivery (CI CD) pipeline approach has increased the efficiency of projects. In this project, the CI/CD pipeline is used to automate the deployment process, to ensure faster delivery, higher quality, and increased productivity. The pipeline typically involves four stages: source, build, test, and deployment. The deployment stage involves deploying the built application to the target environment, such as a cloud server or container, and include infrastructure provisioning, configuration, and containerization using technologies like Docker.

Jenkins is used to build and deploy applications in the CI/CD pipeline. A CI/CD pipeline supports continuous deployment, reducing the risk of integration issues and enabling faster bug detection. Docker xCAT image deployment can be used to deploy HPC applications in future enhancement.

2. OVERVIEW OF THE PROJECT

Project Scope and Objectives: The primary objective of the project is to implement a robust CI/CD pipeline that automates the software deployment process. This automation aims to expedite delivery, maintain high software quality, and boost efficiency. The pipeline comprises source code management, build automation, testing, and deployment stages.

CI/CD Pipeline Stages:

1. **Source Stage:** This is where the source code is stored in a version control system (e.g., Git). Changes made to the codebase trigger the pipeline to start.
2. **Build Stage:** The CI server (in your case, Jenkins) pulls the code from the repository and compiles it. This stage also includes any necessary pre-processing, such as compiling assets or bundling dependencies.
3. **Test Stage:** Automated tests are executed to ensure the code is functioning as expected. This stage includes unit tests, integration tests, and potentially other testing types like security or performance testing.
4. **Deployment Stage:** The built and tested application is deployed to the target environment, which could be a cloud server or a container. This step involves infrastructure provisioning, configuration, and containerization using Docker.

Benefits of CI/CD Pipeline:

- **Faster Delivery:** Automation streamlines the deployment process, reducing manual intervention and speeding up delivery times.
- **Higher Quality:** Automated tests catch bugs and issues early, enhancing software quality and stability.
- **Increased Productivity:** Developers spend less time on manual deployment tasks, allowing them to focus on coding.
- **Reduced Integration Issues:** Continuous integration minimizes integration issues by frequently merging changes into the main codebase.
- **Continuous Deployment Support:** The pipeline facilitates continuous deployment, allowing rapid, automated releases when code changes are ready.

Future Enhancements': Docker xCAT Image Deployment: In future iterations, enhancing the project to deploy HPC (High-Performance Computing) applications using Docker xCAT images could significantly benefit scientific and computational workload.

3. Deployment of CI/CD pipeline on cloud

A Jenkins CI/CD pipeline is a set of automated processes within the Jenkins continuous integration server that helps developers build, test, and deploy their code efficiently. Here is a simplified breakdown of how it works:

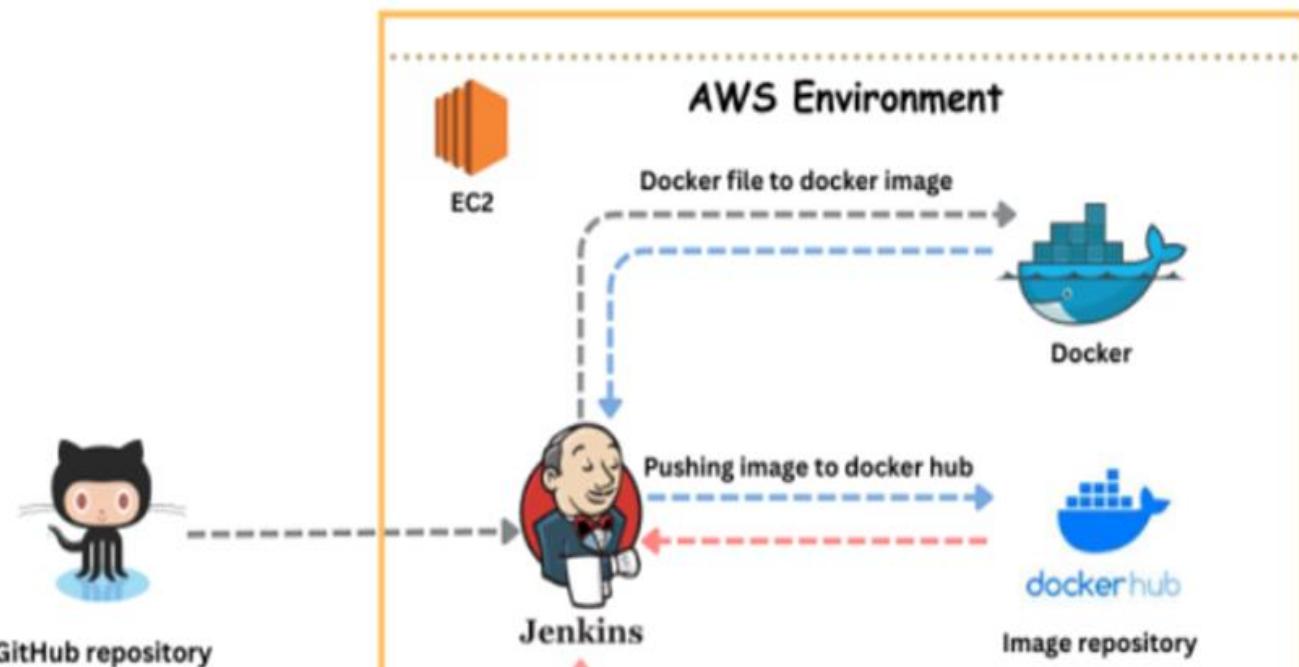
1. Code Push: When a developer pushes code to a version control repository (e.g., Git), Jenkins is triggered to start the pipeline.
2. Build: Jenkins pulls the latest code from the repository and compiles/builds the codebase. This step ensures that the code is syntactically correct and can be executed.
3. Automated Testing: After the build, the pipeline runs automated tests, such as unit tests, integration tests, and regression tests. These tests verify that the code functions as expected and has not introduced any regressions.
4. Code Analysis: Jenkins may perform static code analysis to identify coding issues, vulnerabilities, or coding style violations.
5. Deployment: If all tests pass and the code analysis are satisfactory, Jenkins deploys the code to a specified environment, such as a cloud server.
6. Additional Testing: More thorough testing may be performed in the staging environment to simulate real-world usage and identify any issues that weren't caught during earlier stages.
7. Final Deployment: Once the code has successfully passed through the various stages and received approval, Jenkins deploys it to the production environment.

The beauty of Jenkins pipelines lies in their configurability and automation. Developers can define the exact steps, tests, and environments their code goes through, reducing manual intervention and ensuring a consistent and reliable deployment process.

Jenkins declarative CI/CD pipeline using following tools:

1. GitHub - Code repository
2. Docker - Build tool

3. Docker HUB- Build/image repository
4. AWS EC2 instance
5. Jenkins - to automate the entire pipeline
- 6.Tomcat server



Containerized Web-based Application created using CI/CD declarative pipeline through Jenkins.

Implemented an automated CI/CD pipeline for web application by developing a Git Workflow that seamlessly triggers code built for every commit made to the master branch. Leveraged GitHub, EC2, Docker, Jenkins, Docker Hub to successfully build and automate the deployment process

4. System Requirement

Cloud Provider Account: We need an account with a chosen cloud provider (AWS, Azure, GCP, etc.) to access their services and resources.

Version Control System: Version control system (e.g., Git) and a repository to manage the codebase.

CI/CD Tool: The CI/CD tool meets the project's needs and is compatible with the chosen cloud provider. Checked the system requirements of the tool and ensured it's properly configured.

Source Code: The source code should be organized and stored in a way that's compatible with CI/CD tool's requirements.

Build Environment: Depending on application's technology stack, might need specific tools and dependencies to build applications. This could include compilers, libraries, and other software components.

Virtual Machines or Containers: This CI/CD pipeline requires virtual machines and containers to run build and test processes. Make sure you have the appropriate resources available for this.

Network Connectivity: A stable and reliable internet connection is essential for interacting with cloud services and for enabling CI/CD pipeline to access code repositories and external resources.

Tomcat server: Ensure Tomcat server configuration to display web pages.

Backup and Recovery Plan: Can have a plan in place for backing up CI/CD pipeline configurations and code repositories. This helps in case of accidental data loss or system failures.

Scaling Considerations: If the project grows, there are future enhancements considered while implementing this project.

Documentation: Documentation is done w.r.to the setup and configuration of your CI/CD pipeline, including any environment variables, dependencies, and configuration settings.

Budget and Cost Management: This project is developed considering the costs associated with cloud services.

4.1. Software Requirement

Version Control System:

Git: A distributed version control system used to manage your source code.

CI/CD Tools:

Jenkins: An open-source automation server that orchestrates the entire CI/CD process.

GitHub CI/CD: Built-in CI/CD functionality provided by GitHub, covering the entire software development lifecycle.

Build Tools:

Docker: A platform for developing, shipping, and running applications in containers.

EC2 instance: Instances for execution on AWS Cloud Formation.

Scripting Languages: Python is used for scripting automation tasks within your CI/CD pipeline.

AWS CLI: Command-line interface for managing AWS resources.

PyCharm: PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, along with automated code refactoring's and rich navigation capabilities.

4.2 Hardware Requirement

Compute Resources:

Virtual Machines: Choose the appropriate instance types based on the workload requirements of your CI/CD pipeline. These might include factors like CPU, memory, and storage capacity.

Networking:

Network Bandwidth: Make sure your cloud instances or containers have adequate network bandwidth for fetching code from repositories, pushing build artifacts, and communicating with external services.

Storage:

Disk Space: Allocate sufficient disk space to store code repositories, build artifacts, logs, and other data generated during the CI/CD process

4.3 User Interface

Jenkins:

Jenkins is a widely used open-source automation server. Its UI can be accessed through a web browser. Allows to configure and manage jobs (pipelines) through this interface. The UI allows to create new jobs, specify source code repositories, define build steps, configure triggers, set up integrations, and monitor job execution and logs.

GitHub CI/CD:

GitHub is used for version control to include its CI/CD functionality into the platform. The UI allows to configure CI/CD pipelines directly within GitHub repository. The UI shows pipeline status, logs, and artifacts.

AWS:

AWS provides a comprehensive suite of services for CI/CD. Its UI allows to set up pipelines by configuring build and release definitions. We can define triggers, configure source repositories, specify build steps, set up testing, and manage deployment stages. The UI also provides insights into builds, tests, and deployments.

The general process of configuring a CI/CD pipeline through these interfaces involves:

Defining a pipeline or job.

Connecting to version control repository.

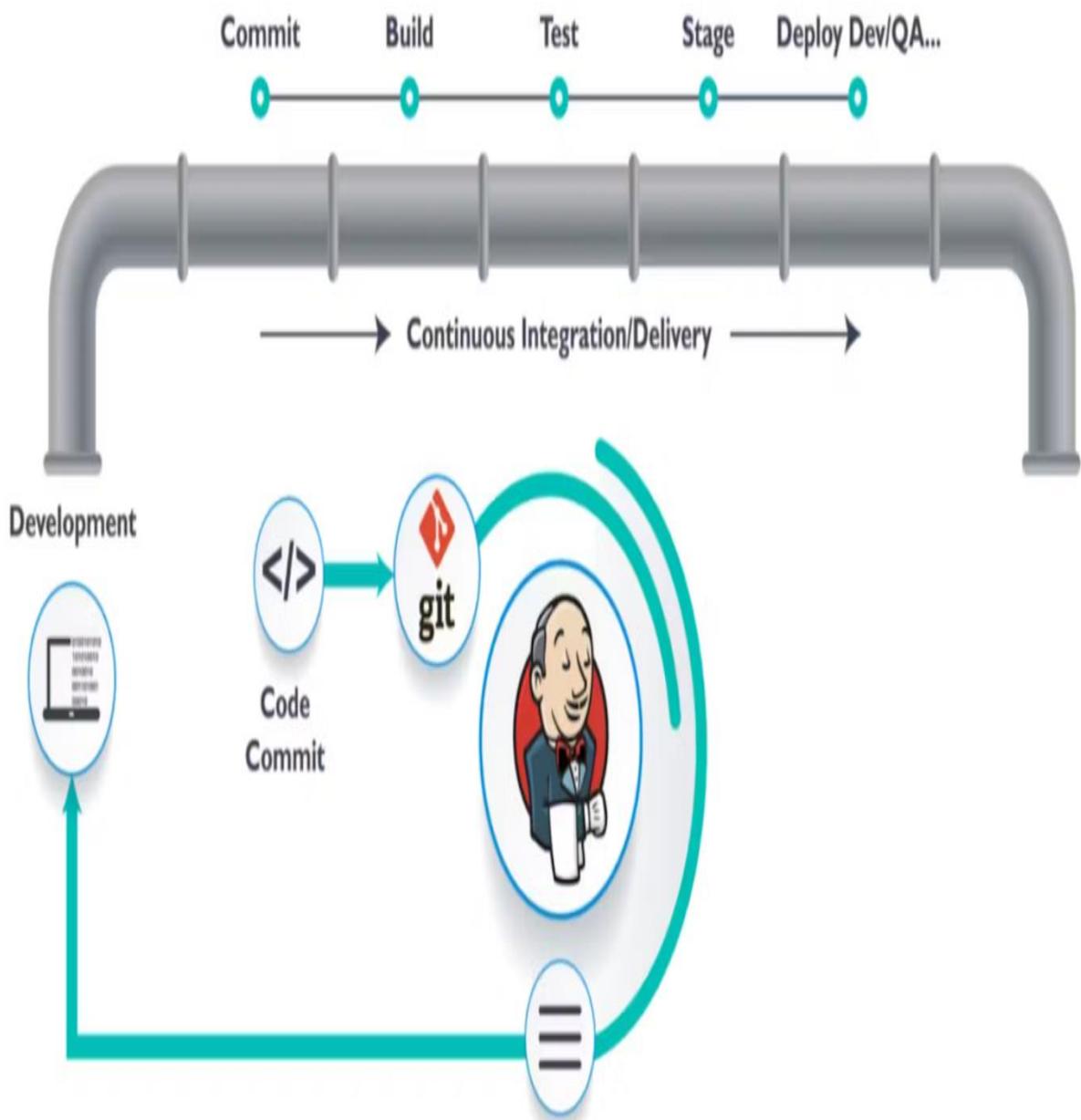
Specifying build and deployment steps.

Configuring triggers (manual, or scheduled).

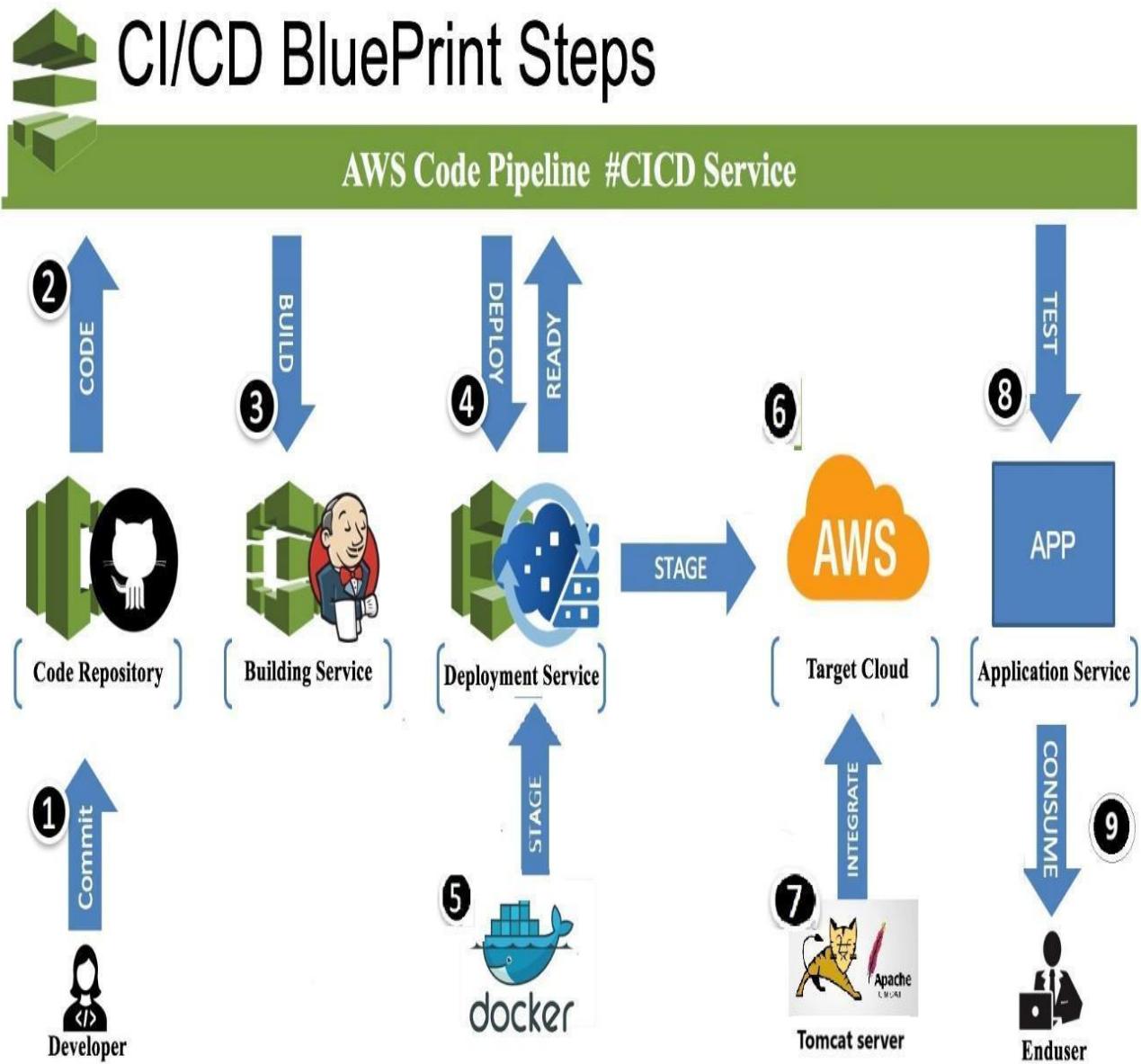
Setting environment variables and configurations.

5. System Architecture

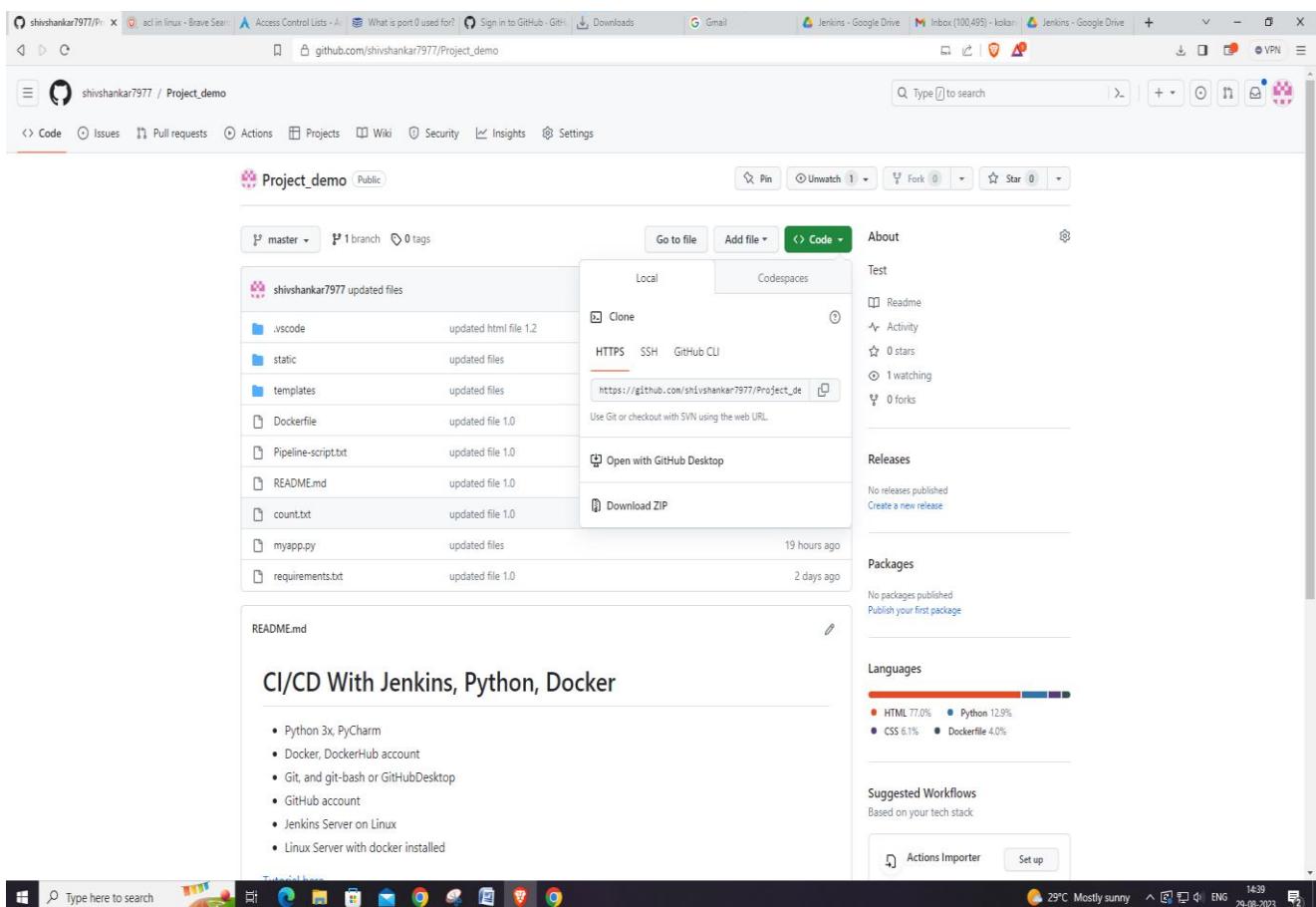
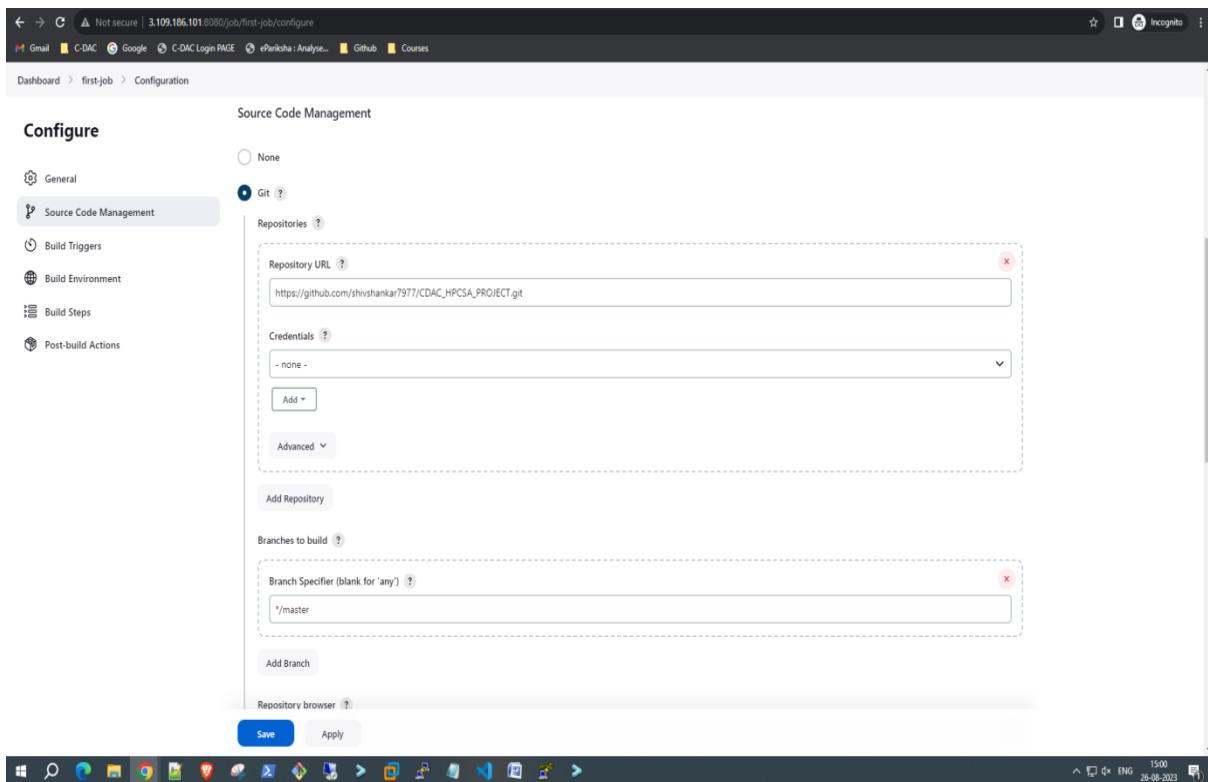
5.1 Data Flow Diagram



5.2 Activity Diagram



6. Introduction to Repository



The screenshot shows a GitHub repository page for 'Project_demo'. The repository has 7 commits from shivshankar7977, 1 branch, and 0 tags. The README.md file contains the following content:

CI/CD With Jenkins, Python, Docker

- Python 3x, PyCharm
- Docker, DockerHub account
- Git, and git-bash or GitHubDesktop
- GitHub account
- Jenkins Server on Linux
- Linux Server with docker installed

The Languages section shows a breakdown: HTML 77.0%, Python 12.9%, CSS 6.1%, and Dockerfile 4.0%. Suggested Workflows are based on the tech stack.

7. Git Concepts

Git is a version control system that is used to manage and track changes made to source code and other files. It was created by Linus Torvalds in 2005 to manage the development of the Linux operating system. With Git, developers can track changes to their code, collaborate with others on a project, and maintain multiple versions of their codebase. Git uses a distributed model, which means that every developer has a complete copy of the codebase on their local machine. This allows developers to work offline and independently, and then synchronize their changes with others when they're ready.

Git also provides features like branching and merging, which allow developers to create parallel versions of their codebase and merge changes made by multiple developers back into the main codebase. These features make it easier to manage complex development workflows and collaborate on large projects. Overall, Git is a powerful and widely used tool for managing software development projects.

Git clone git is a command used to create a copy of a Git repository on your local machine. To use git clone, you need to specify the URL of the remote repository you want to clone, as well as the location on your local machine where you want to create the copy.

Git push is a command used to upload local repository content to a remote repository. When you make changes to your local repository, you can use git push to send those changes to a remote repository that you have previously cloned or created. The basic syntax of the command is: git push File pushed

8. Jenkins server configuration

JENKINS

Jenkins is a popular open-source automation server that can be used to implement Continuous Integration/Continuous Deployment (CI/CD) pipelines in software development projects. It provides a wide range of plugins and integrations with various tools and technologies, making it a flexible and powerful tool for implementing a CI/CD pipeline.

In a typical CI/CD pipeline, Jenkins can be used to automate the build, test, and deployment processes. It can integrate with source code repositories, such as Git, to automatically trigger a build when new code is pushed to the repository. Jenkins can then compile the code, run unit tests, and generate reports on the test results.

One of the key advantages of using Jenkins in a CI/CD pipeline is its ability to provide visibility and control over the entire pipeline. Jenkins can provide real-time feedback on the status of each stage in the pipeline, making it easier to identify and address any issues that arise. It can also be configured to send notifications and alerts to team members when a build fails or when an issue is detected. Overall, Jenkins is a valuable tool for implementing a robust and efficient CI/CD pipeline in software development projects. Its flexibility and extensive plugin library make it an ideal choice for teams looking to automate their build, test, and deployment processes while maintaining control and visibility over the pipeline.



Fig 3.1 Jenkins Logo

9. Docker Configuration

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code, you can significantly reduce the delay between writing code and running it in production.

Docker file:

```
#Dockerfile, image, container

FROM python:3.8-slim-buster
ADD . /python-flask
WORKDIR /python-flask
RUN pip install -r requirements.txt
CMD [ "python", "./myapp.py" ]
```

requirements.txt

```
click==8.0.3
colorama==0.4.4
Flask==2.0.2
itsdangerous==2.0.1
Jinja2==3.0.3
MarkupSafe==2.0.1
Werkzeug==2.0.2
```

10. Pulling docker image

A screenshot of a web browser showing the Docker Hub search results for "python:3.8-slim-buster". The search bar at the top contains the query. On the left, there are filters for Products (Images, Extensions, Plugins), Trusted Content (Docker Official Image, Verified Publisher, Sponsored OSS), Operating Systems (Linux, Windows), Architectures (ARM, ARM 64, IBM POWER, IBM Z, PowerPC 64 LE, x86, x86-64), and a section for "Best Match". The main area displays three search results:

- ekgf/debian-awscli**: 10K+ stars, updated 2 years ago. Description: Based on python:3.8-slim-buster, adds AWS CLI and some Python libs for RDF processing. Tags: Linux, x86-64.
- xzzcx/mtool**: 182 stars, updated 3 years ago. Description: 用树莓派定时播放音乐. Music tool on python:3.8-slim-buster. Tags: Linux, arm.
- okelly3/python-3.8-slim-buster-ibm_db**: 5 stars, updated 4 years ago. Description: Docker image with support for ibm-db based on python:3.8-slim-buster. Tags: Linux, arm.

```
depolyer@tomcat-virtual-machine:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
bbc0522ce128 shivshankar7977/python-jenkins "python ./myapp.py" 10 hours ago Up 10 hours 0.0.0.0:5001->5000/tcp, :::5001->5000/tcp Python-CI-CD-Pipeline
depolyer@tomcat-virtual-machine:~$
```

A screenshot of a web browser showing the Docker Hub repository page for "shivshankar7977/python-jenkins". The URL in the address bar is "hub.docker.com/r/shivshankar7977/python-jenkins". The page includes a purple header with the text "DockerCon 2023: Our annual developer event is back – online & in person. Learn more.". The repository card shows the name "shivshankar7977/python-jenkins" with 1 star, updated 5 hours ago, and 18 pulls. It features a blue cube icon and a "Image" button. Below the card, there are tabs for "Overview" (which is selected) and "Tags". The "Overview" section contains a message: "No overview available. This repository doesn't have an overview." To the right, there is a "Docker Pull Command" box with the command "docker pull shivshankar7977/python-jenkins" and a copy icon. At the bottom of the page, there is a footer with a search bar and various system icons.

11.Launching EC2 instance

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with various services like EC2 Dashboard, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, and AMI Catalog. The main content area displays a table of instances. One instance is listed: Jenkins, with Instance ID l-01d6f92e689404151, Instance state Running, Instance type t2.micro, Status check 2/2 checks passed, Alarm status No alarms, Availability Zone ap-south-1a, Public IPv4 DNS ec2-3-109-186-101.ap-south-1.amazonaws.com, Public IPv4 IP 3.109.186.101, and Elastic IP -.

The screenshot shows the AWS Security Groups page. The URL indicates we're viewing the security group sg-0862ea59b20597889. The main content area shows the security group details: Name sg-0862ea59b20597889, Description launch-wizard-35 created 2023-08-26T08:13:04.670Z, Owner 589371121731, Inbound rules count 2 Permission entries, and Outbound rules count 1 Permission entry. Below this, the Inbound rules section lists two rules:

Name	Security group...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0fa6a8374782ba4f5	IPv4	Custom TCP	TCP	8080	0.0.0.0/0	-
-	sgr-03d3f4cca516d30ba	IPv4	SSH	TCP	22	0.0.0.0/0	-

The screenshot shows the AWS EC2 Instance Connect interface. At the top, there's a navigation bar with links like Gmail, C-DAC, Google, C-DAC Login PAGE, ePariksha : Analyse..., GitHub, Courses, AWS Services, and a search bar. The main title is "Connect to instance". Below it, a sub-header says "Connect to your instance i-01d6f92e689404151 (Jenkins) using any of these options". There are three tabs: "EC2 Instance Connect", "Session Manager", and "SSH client" (which is selected). Under "SSH client", there's a section for "Instance ID" with the value "i-01d6f92e689404151 (Jenkins)". It lists four steps: 1. Open an SSH client, 2. Locate your private key file, 3. Run this command, if necessary, to ensure your key is not publicly viewable (with examples for chmod 400 admin.pem), and 4. Connect to your instance using its Public DNS (with examples for ssh -i "admin.pem" ubuntu@ec2-3-109-186-101.ap-south-1.compute.amazonaws.com and ec2-3-109-186-101.ap-south-1.compute.amazonaws.com). A note at the bottom says "Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name." A "Cancel" button is at the bottom right.

```
# sudo apt update -y
```

```
ubuntu@ip-172-31-1-37:~$ sudo apt update -y
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
112 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

```
# sudo apt upgrade -y
```

```
ubuntu@ip-172-31-1-37:~$ sudo apt upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
#
# You can verify the status of security fixes using the `pro fix` command.
# E.g., a recent Ruby vulnerability can be checked with: `pro fix USN-6219-1`
# For more detail see: https://ubuntu.com/security/notices/USN-6219-1
#
The following NEW packages will be installed:
  linux-aws-6.2-headers-6.2.0-1009 linux-headers-6.2.0-1009-aws linux-image-6.2.0-1009-aws linux-modules-6.2.0-1009-aws
The following packages will be upgraded:
```

```
# java --version
```

```
ubuntu@ip-172-31-1-37:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install openjdk-11-jre-headless # version 11.0.20+8-1ubuntu1~22.04, or
sudo apt install default-jre          # version 2:1.11-72build2
sudo apt install openjdk-17-jre-headless # version 17.0.8+7-1~22.04
sudo apt install openjdk-18-jre-headless # version 18.0.2+9-2~22.04
sudo apt install openjdk-19-jre-headless # version 19.0.2+7-0ubuntu3~22.04
sudo apt install openjdk-8-jre-headless # version 8u382-ga-1~22.04.1
ubuntu@ip-172-31-1-37:~$
```

```
# sudo apt install openjdk-11-jre -y
```

```
ubuntu@ip-172-31-1-37:~$ sudo apt install openjdk-11-jre -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
alsa-topology-conf alsu-ucm-conf at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-service fontconfig-config fonts-dejavu-core
fonts-dejavu-extra gsettings-desktop-schemas java-common libasound2 libasound2-data libatk-bridge2.0-0 libatk-wrapper-java
libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3 libcups2 libdconf1
libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libfontenc1 libglf7 libgl1 libgl1-amber-dri libgl1-mesa-dri
libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgrahpite2-3 libharfbuzz0b libice6 libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm15 libpciaccess0
libpcssclite1 libsensors-config libsensors5 libsm6 libx11-xcb1 libxaw7 libxcb-dr12-0 libxcb-dr13-0 libxcb-glx0 libxcb-present0 libxcb-randr0
libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libcomposite1 libxfixes3 libxf86 libxinerama1 libxkbfile1 libxmu6 libxpm4
libxrandr2 libxrender1 libxshmfence1 libxt6 libxtst6 libxv1 libxf86dga1 libxf86vm1 openjdk-11-jre-headless session-migration x11-common
x11-utils
Suggested packages:
```

```
# sudo apt install openjdk-17-jre -y
```

```
ubuntu@ip-172-31-1-37:~$ sudo apt install openjdk-17-jre
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
adwaita-icon-theme fontconfig gtk-update-icon-cache hicolor-icon-theme humanity-icon-theme libcairo-gobject2 libcairo2 libdatrie1 libdeflate0
libgail-common libgail18 libgdk-pixbuf-2.0-0 libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common libgtk2.0-0 libgtk2.0-bin libgtk2.0-common libjbig0
libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpixman-1-0 librsvg2-2 librsvg2-common libthai-data libthai0 libtiff5 libwebp7
libxcb-render0 libxcursor1 libxdamage1 openjdk-17-jre-headless ubuntu-mono
Suggested packages:
gvfs librsvg2-bin libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
adwaita-icon-theme fontconfig gtk-update-icon-cache hicolor-icon-theme humanity-icon-theme libcairo-gobject2 libcairo2 libdatrie1 libdeflate0
libgail-common libgail18 libgdk-pixbuf-2.0-0 libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common libgtk2.0-0 libgtk2.0-bin libgtk2.0-common libjbig0
libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpixman-1-0 librsvg2-2 librsvg2-common libthai-data libthai0 libtiff5 libwebp7
libxcb-render0 libxcursor1 libxdamage1 openjdk-17-jre openjdk-17-jre-headless ubuntu-mono
0 upgraded, 34 newly installed, 0 to remove and 0 not upgraded.
Need to get 61.1 MB of archives.
After this operation, 251 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

12.Jenkins Installation

```
# curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null  
  
# echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]  
https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
ubuntu@ip-172-31-1-37:~$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null  
ubuntu@ip-172-31-1-37:~$ 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  
ubuntu@ip-172-31-1-37:~$ █
```

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

```
ubuntu@ip-172-31-1-37:~$ sudo apt-get install jenkins -y  
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 0 not upgraded.  
Need to get 89.1 MB of archives.  
After this operation, 90.4 MB of additional disk space will be used.  
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 net-tools amd64 1.60+git20181103.0eebece-1ubuntu5 [204 kB]  
Get:2 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.414.1 [88.9 MB]  
Fetched 89.1 MB in 6s (15.5 MB/s)  
Selecting previously unselected package net-tools.  
(Reading database ... 110026 files and directories currently installed.)  
Preparing to unpack .../net-tools_1.60+git20181103.0eebece-1ubuntu5_amd64.deb ...  
Unpacking net-tools (1.60+git20181103.0eebece-1ubuntu5) ...  
Selecting previously unselected package jenkins.
```

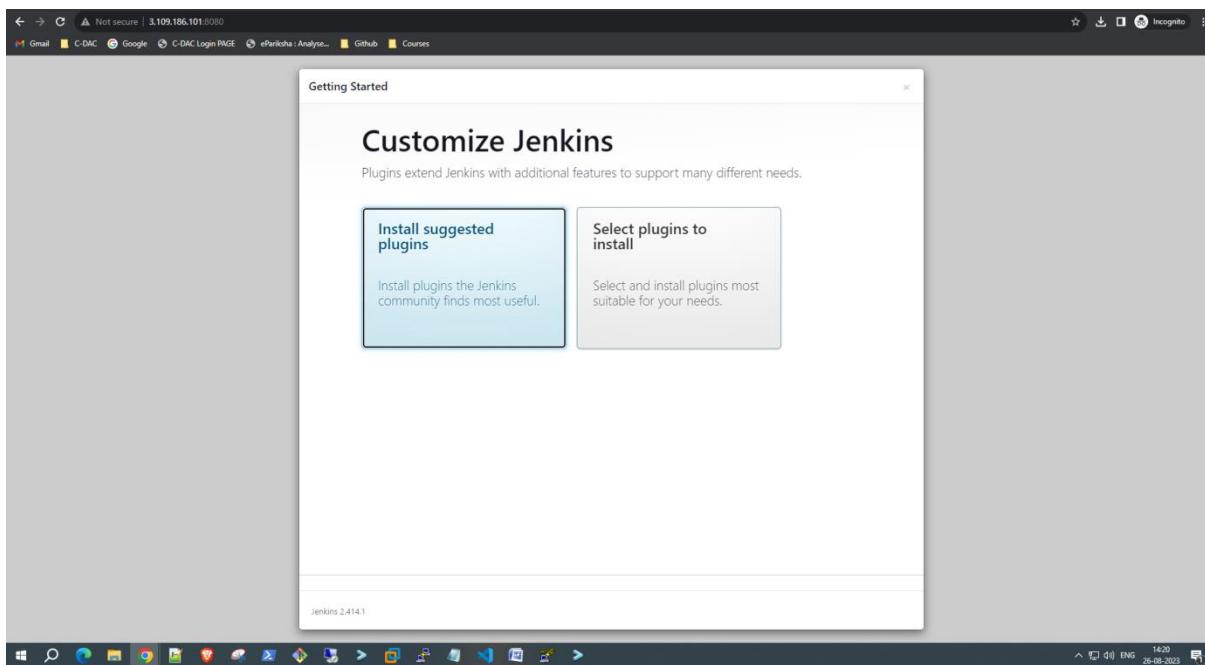
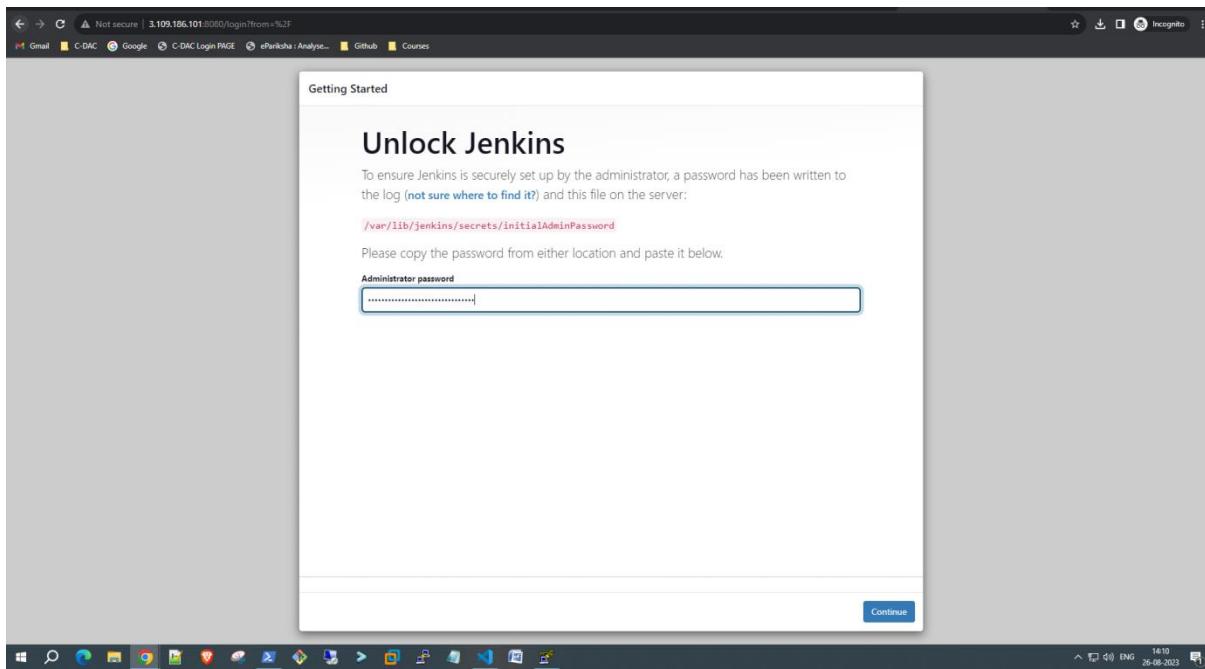
```
# sudo systemctl status jenkins
```

```
ubuntu@ip-172-31-1-37:~$ sudo systemctl status jenkins  
● jenkins.service - Jenkins Continuous Integration Server  
  Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)  
  Active: active (running) since Sat 2023-08-26 08:47:48 UTC; 14s ago  
    Main PID: 26812 (java)  
      Tasks: 45 (limit: 1141)  
        Memory: 305.1M  
          CPU: 37.389s  
        CGroup: /system.slice/jenkins.service  
              └─26812 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
```

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

```
sudo vim /etc/sudoers
```

```
ubuntu@ip-172-31-1-37:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
3fd41bd974d84cddba444ba1f12fb53  
ubuntu@ip-172-31-1-37:~$ █
```



Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.414.1 Skip and continue as admin Save and Continue

Getting Started

Instance Configuration

Jenkins URL:

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 2.414.1 Not now Save and Finish

The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with links to Gmail, C-DAC, Google, C-DAC Login PAGE, ePanika : Analyse..., GitHub, and Courses. The main header says "Jenkins" with a user icon. Below the header, there's a search bar and a "Dashboard" link. On the left, there are several dropdown menus: "Build Queue" (No builds in the queue), "Build Executor Status" (1 Idle, 2 Idle), "New Item" (+ New Item), "People" (People), "Build History", "Manage Jenkins", and "My Views". The central area has a "Welcome to Jenkins!" message, a note about displaying Jenkins jobs, and a "Start building your software project" section with links to "Create a job", "Set up a distributed build", "Set up an agent", "Configure a cloud", and "Learn more about distributed builds".

This screenshot shows the "Enter an item name" dialog box. The input field contains "first-job" and has a note "(Required field)". Below the input field, there are six project types listed 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 dialog is an "OK" button.

Not secure | 3.109.186.101:8080/job/first-job/configure

Gmail C-DAC Google C-DAC Login PAGE ePariksha : Analyse... GitHub Courses

Jenkins

Dashboard > first-job > Configuration

Configure General

Description: website deployment

Plain text: Preview

Discard old builds

GitHub project

Project url: https://github.com/shivshankar7977/CDAC_HPCSA_PROJECT

This project is parameterized

Throttle builds

Execute concurrent builds if necessary

Advanced

Save Apply

This screenshot shows the 'General' configuration page for a Jenkins job named 'first-job'. The 'GitHub project' section is active, with the URL 'https://github.com/shivshankar7977/CDAC_HPCSA_PROJECT' entered. Other options like 'Discard old builds' and 'Execute concurrent builds if necessary' are also visible.

Not secure | 3.109.186.101:8080/job/first-job/configure

Gmail C-DAC Google C-DAC Login PAGE ePariksha : Analyse... GitHub Courses

Jenkins

Dashboard > first-job > Configuration

Configure Source Code Management

Source Code Management: Git

Repository URL: https://github.com/shivshankar7977/CDAC_HPCSA_PROJECT.git

Credentials: none

Add Repository

Branches to build: */master

Add Branch

Repository browser

Save Apply

This screenshot shows the 'Source Code Management' configuration page for the same Jenkins job. It specifies 'Git' as the provider and sets the repository URL to 'https://github.com/shivshankar7977/CDAC_HPCSA_PROJECT.git'. It also defines the branch to build as '*/master'.

Not secure | 3.109.186.101:8080/job/first-job/configure

Gmail C-DAC Google C-DAC Login PAGE ePariksha : Analyse... GitHub Courses

Dashboard > first-job > Configuration

Configure

Build Triggers

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Poll SCM

Schedule ?
H/15 * * * *

Would last have run at Saturday, August 26, 2023 at 9:40:42 AM Coordinated Universal Time: would next run at Saturday, August 26, 2023 at 9:55:42 AM Coordinated Universal Time.

Ignore post-commit hooks ?

Build Environment

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

With Ant ?

Save **Apply**

This screenshot shows the Jenkins configuration interface for a job named 'first-job'. The 'Build Triggers' section is active, showing a 'Poll SCM' trigger set to run every 15 minutes. The 'Build Environment' section contains several optional checkboxes for workspace management and timestamps. At the bottom, there are 'Save' and 'Apply' buttons.

Not secure | 3.109.186.101:8080/job/first-job/configure

Gmail C-DAC Google C-DAC Login PAGE ePariksha : Analyse... GitHub Courses

Dashboard > first-job > Configuration

Configure

Build Environment

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Build Steps

Execute shell ?

Command
See the list of available environment variables

```
sudo apt-get install apache2 -y
sudo systemctl start apache2
sudo systemctl enable apache2
sudo git clone https://github.com/shivshankar7977/CDAC_HPCSA_PROJECT.git /var/www/html
```

Advanced ▾

Add build step ▾

Save **Apply**

This screenshot shows the Jenkins configuration interface for the same job. The 'Build Environment' section is active, and the 'Build Steps' section is expanded. It contains a single 'Execute shell' step with a command to install Apache and clone a GitHub repository. There is also an 'Advanced' dropdown and an 'Add build step' button.

The screenshot shows the Jenkins interface for the 'first-job' project. The top navigation bar includes links for Gmail, C-DAC, Google, C-DAC Login PAGE, ePanika : Analyse..., Github, and Courses. The main title is 'Project first-job' under 'website deployment'. On the left, a sidebar lists options like Status, Changes, Workspace, Build Now, Configure, Delete Project, Git Polling Log, and Rename. The 'Build History' section shows two builds: one successful build at 9:48 AM on Aug 26, 2023, and one failed build at 9:46 AM on Aug 26, 2023. A 'Permalinks' section provides links to the latest build and the last four builds. At the bottom, there are Atom feed links.

REST API Jenkins 2.414.1

The screenshot shows the Jenkins interface for the second build of the 'first-job' project. The title is 'Build #2 (Aug 26, 2023, 9:48:46 AM)'. The build status is green with a checkmark. The sidebar includes Status, Changes, Console Output, Edit Build Information, Delete build '#2', Git Build Data, and Previous Build. The build details show 'No changes.', 'Started by user admin', 'Revision: 2c6bc0c9ea91029da37fb4b5ec30988b3bc3ebd', 'Repository: https://github.com/shivshankar7977/CDAC_HPCSA_PROJECT.git', and 'refs/remotes/origin/master'. Buttons for 'Keep this build forever', 'Add description', and 'Started 40 sec ago Took 3.3 sec' are visible. The bottom status bar shows the date as 26-08-2023.

REST API Jenkins 2.414.1

Not secure | 3.109.186.101:8080/job/first-job/2/console

Jenkins

Dashboard > first-job > #2 > Console Output

Console Output

```

Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/first-job
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/first-job/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/shivshankar7977/CDAC_HPCSA_PROJECT.git # timeout=10
Fetching upstream changes from https://github.com/shivshankar7977/CDAC_HPCSA_PROJECT.git
> git --version # timeout=10
> git --version # "git version 2.34.1"
> git fetch --tags --progress -- https://github.com/shivshankar7977/CDAC_HPCSA_PROJECT.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 2c0bc0c9ea91029da377f4b5ec30988b3c3ebd2 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 2c0bc0c9ea91029da377f4b5ec30988b3c3ebd2 # timeout=10
Commit message: "Update Jenkins_Installation.md"
> git rev-list --no-walk 2c0bc0c9ea91029da377f4b5ec30988b3c3ebd2 # timeout=10
[first-job] $ /bin/sh -e /tmp/Jenkins4266478549651906029.sh
+ sudo apt-get install apache2 -y
Reading package lists...
Building dependency tree...
Building state information...
apache2 is already the newest version (2.4.52-1ubuntu4.6).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
+ sudo systemctl start apache2
+ sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/system-sysv-install.
Executing: /lib/systemd/system-sysv-install enable apache2
+ sudo git clone https://github.com/shivshankar7977/CDAC_HPCSA_PROJECT.git /var/www/html
Cloning into '/var/www/html'...
Finished: SUCCESS

```

Search (CTRL+K) | admin | log out

Not secure | 3.109.186.101:8080

Jenkins

Dashboard >

New Item **Add description**

S	W	Name	Last Success	Last Failure	Last Duration
Green	Yellow	first-job	1 min 26 sec #16	19 min #9	3.7 sec

Build Queue **Icon:** S M L **Icon legend:** Atom feed for all Atom feed for failures Atom feed for just latest builds

No builds in the queue.

Build Executor Status **1 Idle**

REST API Jenkins 2.414.1

13. Docker Installation

On Jenkins server:

```
master@master:~$ sudo apt-get install docker* -y
[sudo] password for master:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'docker-compose' for glob 'docker*'
Note, selecting 'docker.io-doc' for glob 'docker*'
Note, selecting 'docker2aci' for glob 'docker*'
Note, selecting 'docker-doc' for glob 'docker*'
Note, selecting 'docker-ce' for glob 'docker*'
Note, selecting 'docker.io' for glob 'docker*'
Note, selecting 'docker' for glob 'docker*'
Note, selecting 'docker-clean' for glob 'docker*'
Note, selecting 'docker-registry' for glob 'docker*'
Note, selecting 'docker-doc' instead of 'docker.io-doc'
docker is already the newest version (1.5-2).
docker-clean is already the newest version (2.0.4-4).
docker-compose is already the newest version (1.29.2-1).
docker-registry is already the newest version (2.8.0+ds1-4).
docker2aci is already the newest version (0.17.2+dfsg-2.1).
Suggested packages:
  aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap rinse zfs-fuse | zfsutils
The following packages will be upgraded:
  docker-doc docker.io
```

On Tomcat server:

```
tomcat@tomcat-virtual-machine:~$ sudo apt-get install docker* -y
[sudo] password for tomcat:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Note, selecting 'docker-compose' for glob 'docker*'
Note, selecting 'docker.io-doc' for glob 'docker*'
Note, selecting 'docker2aci' for glob 'docker*'
Note, selecting 'docker-doc' for glob 'docker*'
Note, selecting 'docker-ce' for glob 'docker*'
Note, selecting 'docker.io' for glob 'docker*'
Note, selecting 'docker' for glob 'docker*'
Note, selecting 'docker-clean' for glob 'docker*'
Note, selecting 'docker-registry' for glob 'docker*'
Note, selecting 'docker-doc' instead of 'docker.io-doc'
docker is already the newest version (1.5-2).
docker-clean is already the newest version (2.0.4-4).
docker-compose is already the newest version (1.29.2-1).
docker-registry is already the newest version (2.8.0+ds1-4).
docker2aci is already the newest version (0.17.2+dfsg-2.1).
docker-doc is already the newest version (20.10.25-0ubuntu1~22.04.2).
docker.io is already the newest version (20.10.25-0ubuntu1~22.04.2).
0 upgraded, 0 newly installed, 0 to remove and 15 not upgraded.
tomcat@tomcat-virtual-machine:~$
```

14. Scripts

14.1 Python-Flask Script on PyCharm.

Flask script

myapp.py

```
from flask import Flask, render_template
app = Flask(__name__)

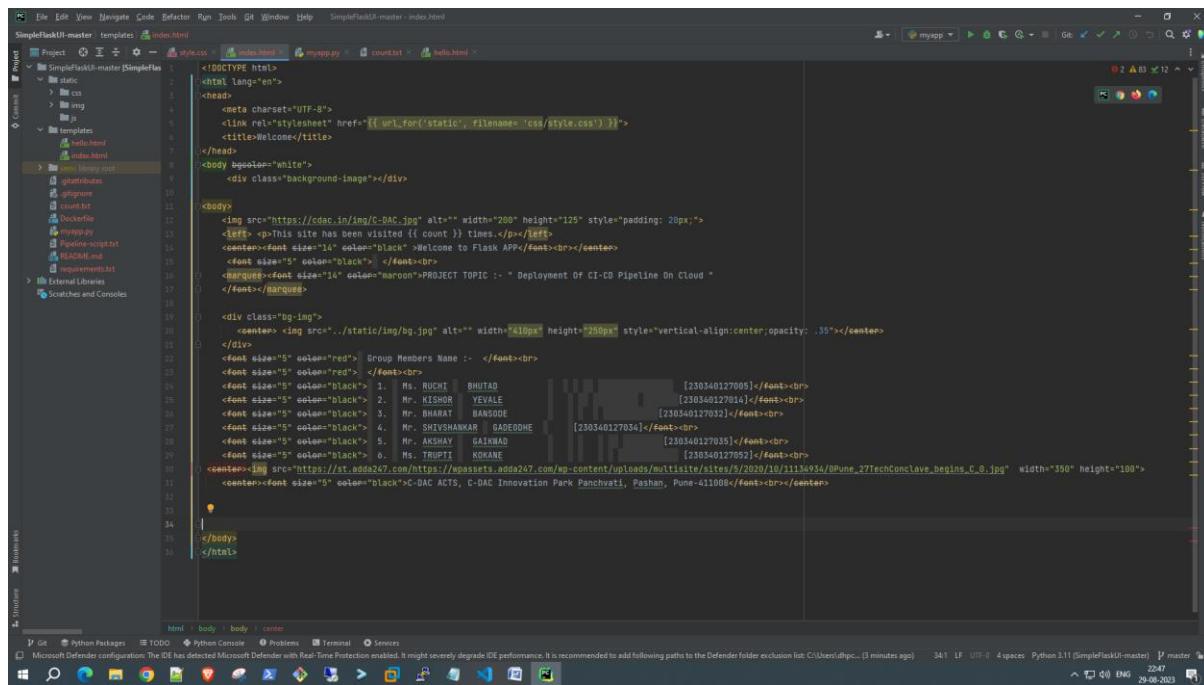
@app.route('/')
def hello():
    return render_template('index.html')

@app.route('/hello/<name>')
def hello_name(name):
    # Load current count
    f = open("count.txt", "r")
    count = int(f.read())
    f.close()
    # Increment the count
    count += 1
    # Overwrite the count
    f = open("count.txt", "w")
    f.write(str(count))
    f.close()

    return render_template('index.html', name=name, count=count)

if __name__ == '__main__':
    app.run(host ='0.0.0.0', debug = True)
```

14.2 HTML Script

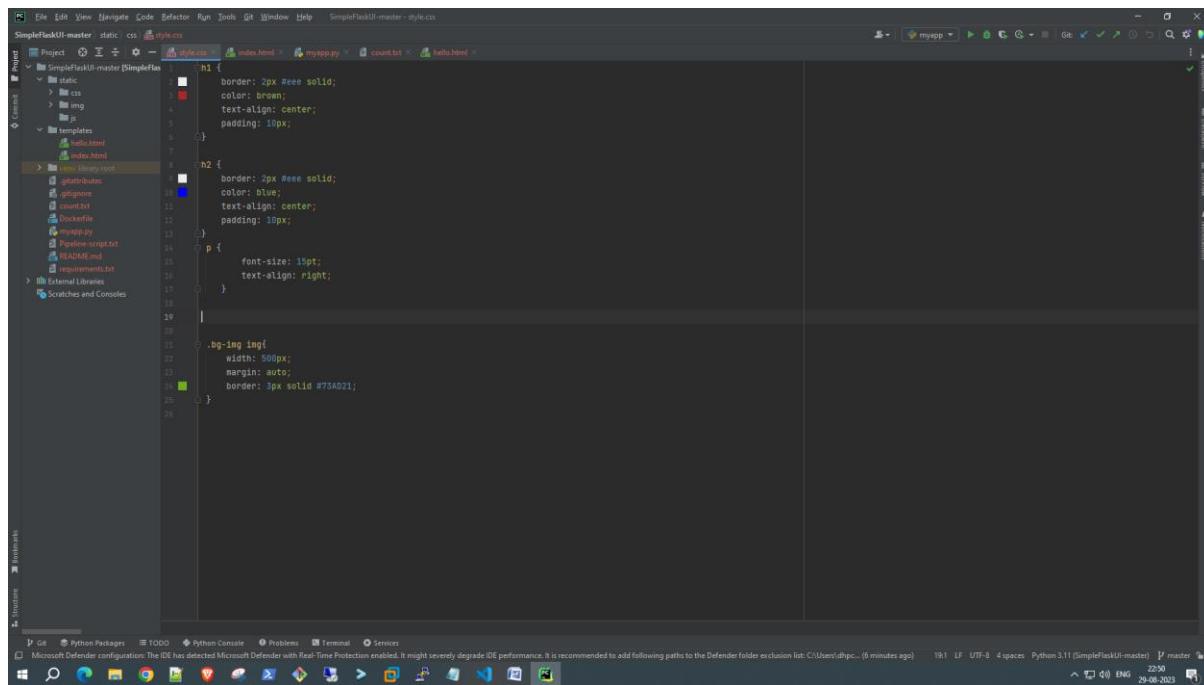


The screenshot shows the PyCharm IDE interface with the file `index.html` open. The code is an HTML document with a welcome message and a list of names:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}>
    <title>Welcome</title>
</head>
<body style="background-color: white">
    <div class="background-image"></div>

    <body>
        
        <left> <p>This site has been visited {{ count }} times.</p>
        <center><font size="16" color="black"> Welcome to Flask APP </font><br></center>
        <font size="5" color="black"> </font><br>
        <marquee><font size="14" color="maroon">PROJECT TOPIC :- Deployment Of CI-CO Pipeline On Cloud </font></marquee>
        <br>
        <div class="bg-img">
            <center> </center>
        </div>
        <font size="5" color="red"> Group Members Name : </font><br>
        <font size="5" color="red"> </font><br>
        <font size="5" color="black"> 1. Ms. RUCHI BHUTAD [230340127005]</font><br>
        <font size="5" color="black"> 2. Mr. KISHOR YEVALE [230340127014]</font><br>
        <font size="5" color="black"> 3. Mr. BHARAT BANSODE [230340127033]</font><br>
        <font size="5" color="black"> 4. Mr. SHIVSHANKAR GADEODE [230340127034]</font><br>
        <font size="5" color="black"> 5. Mr. AKSHAY GAIRWADE [230340127035]</font><br>
        <font size="5" color="black"> 6. Ms. TRUPTI KORANE [230340127052]</font><br>
        <center><img alt="https://st.adda247.com/https://pagesets.adda207.com/wp-content/uploads/multisite/sites/5/02/10/11164934/0Pune_27TechConclave_begins_C_B.jpg" width="350" height="100">
        <center><font size="5" color="black"> C-DAC ACTS, C-DAC Innovation Park Panchvati, Panvel, Pune, 411008</font></center>
    </body>
</html>
```

14.3 CSS Script



The screenshot shows the PyCharm IDE interface with the file `style.css` open. The CSS styles define the appearance of various elements:

```
h1 {
    border: 2px #eee solid;
    color: brown;
    text-align: center;
    padding: 10px;
}

h2 {
    border: 2px #eee solid;
    color: blue;
    text-align: center;
    padding: 10px;
}

p {
    font-size: 15pt;
    text-align: right;
}

.bg-img img {
    width: 500px;
    margin: auto;
    border: 3px solid #75A021;
}
```

14.4 myapp.py (Flask App Code)

The screenshot shows the PyCharm IDE interface with the 'myapp.py' file open. The code implements a simple Flask application that reads a counter from a file, increments it, and displays it along with a greeting name. The IDE's status bar at the bottom indicates Python 3.11, master branch, and a date of 29-08-2023.

```
from flask import Flask, render_template
app = Flask(__name__)

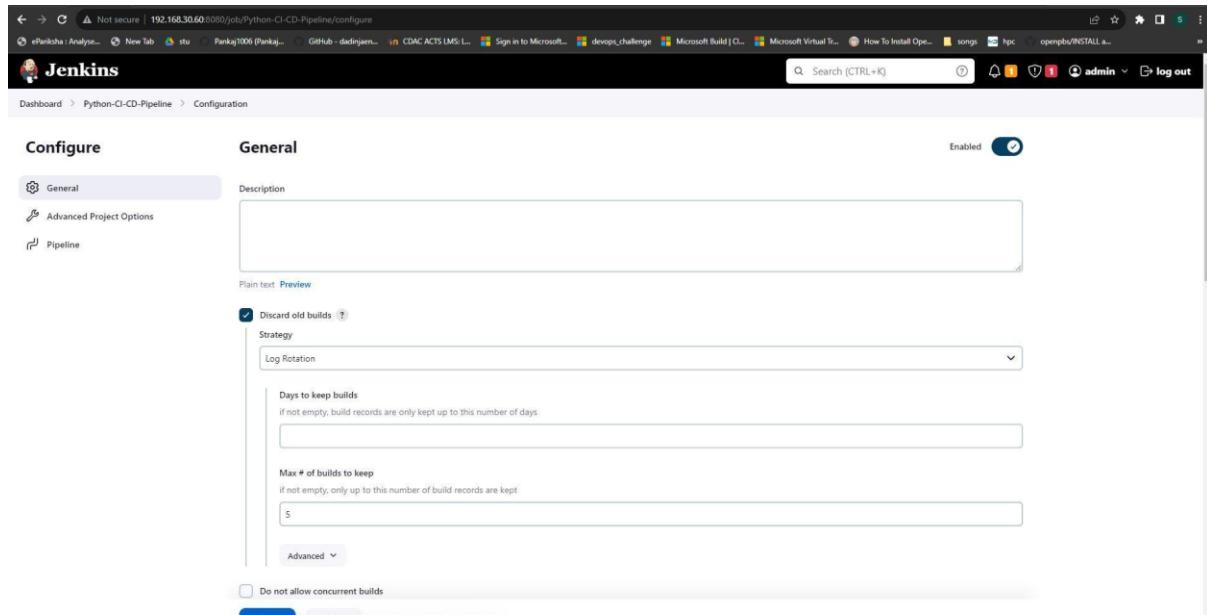
@app.route('/')
def hello():
    return render_template('index.html')

@app.route('/hello/')
def hello_name(name):
    # Load current count
    f = open("count.txt", "r")
    count = int(f.read())
    f.close()
    # Increment the count
    count += 1
    # Overwrite the count
    f = open("count.txt", "w")
    f.write(str(count))
    f.close()

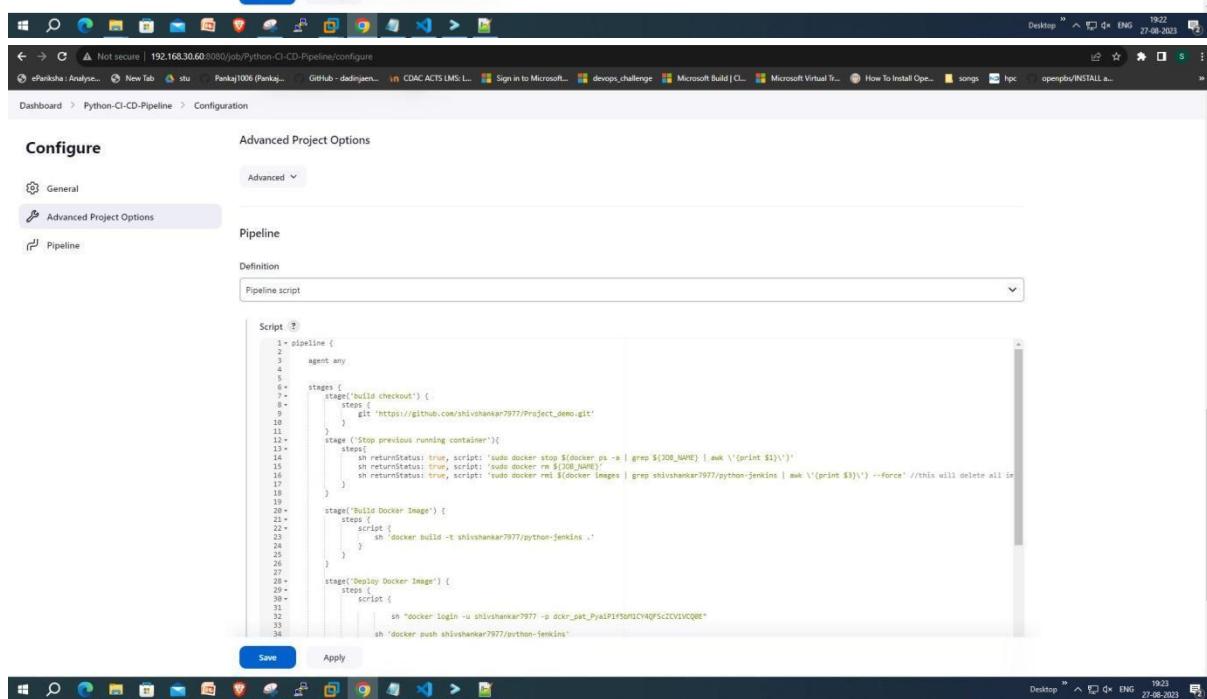
    return render_template('index.html', name=name, count=count)

if __name__ == '__main__':
    app.run(host='0.0.0.1', debug=True)
```

15. Jenkins files configuration



The screenshot shows the Jenkins General Configuration page. Under the 'General' tab, there is a 'Description' field with placeholder text 'Plain text' and 'Preview' buttons. Below it, a checkbox 'Discard old builds' is checked. A dropdown menu 'Strategy' is set to 'Log Rotation'. Under 'Log Rotation', there is a 'Days to keep builds' field with a placeholder 'if not empty, build records are only kept up to this number of days' and a 'Max # of builds to keep' field with a value of '5'. An 'Advanced' button is visible. At the bottom are 'Save' and 'Apply' buttons.



The screenshot shows the Jenkins Advanced Project Options page. Under the 'Advanced Project Options' tab, there is a 'Pipeline' section with a 'Definition' dropdown set to 'Pipeline script'. Below it is a large code editor containing a Jenkins Pipeline script:

```
1# pipeline {
2#     agent any
3#
4#     stages {
5#         stage('build checkout') {
6#             steps {
7#                 git 'https://github.com/shivshankar7977/Project_demo.git'
8#             }
9#         }
10#        stage('Stop previous running container'){
11#            steps{
12#                sh returnStatus: true, script: 'sudo docker stop $(docker ps -a | grep ${JOB_NAME} | awk \'{print $1}\')'
13#                sh returnStatus: true, script: 'sudo docker rm ${JOB_NAME}'
14#                sh returnStatus: true, script: 'sudo docker rmi $(docker images | grep shivshankar7977/python-jenkins | awk \'{print $3}\') --force' //this will delete all images
15#            }
16#        }
17#        stage('Build Docker Image') {
18#            steps {
19#                script {
20#                    sh 'Docker build -t shivshankar7977/python-jenkins .'
21#                }
22#            }
23#        }
24#        stage('Deploy Docker Image') {
25#            steps {
26#                script {
27#                    sh "docker login -u shivshankar7977 -p dckr_pat_PyaiP1f50f1C1QF5c2C1V1CQ0E"
28#                    sh 'docker push shivshankar7977/python-jenkins'
29#                }
30#            }
31#        }
32#    }
33#}
```

At the bottom are 'Save' and 'Apply' buttons.

Dashboard > Python-Cl-CD-Pipeline > Configuration

Configure

General

Advanced Project Options

Pipeline

```

Script: Pipeline script

26
27
28+
29+
30+
31+
32+
33+
34+
35+
36+
37+
38+
39+
40+
41+
42+
43+
44+
45+
46+
47+
48+
49+
50+
51+
52+
53+
54+
55+
56+
57+
58+
59+
60+
61+
62+
63+
64+

```

Use Groovy Sandbox

Pipeline Syntax

Save **Apply**

Desktop 19:23 27-08-2023

Not secure | 34.204.97.159:8080/jenkins/blue/organizations/jenkins/python_deployment/configure

Dashboard > Pipeline_for_python_deployment > Configuration

Configure

General

Advanced Project Options

Pipeline

```

Script: Pipeline script

1+
2+
3+
4+
5+
6+
7+
8+
9+
10+
11+
12+
13+
14+
15+
16+
17+
18+
19+
20+
21+
22+
23+
24+
25+
26+
27+
28+
29+
30+
31+
32+
33+
34+
35+
36+
37+
38+
39+
40+
41+
42+
43+
44+
45+
46+
47+
48+
49+
50+
51+
52+
53+
54+
55+
56+
57+
58+
59+
60+
61+
62+
63+
64+

```

Save **Apply**

Desktop 19:23 28-08-2023

Not secure | 34.204.97.159:8080/manage/credentials/

Jenkins

Dashboard > Manage Jenkins > Credentials

Credentials

T	P	Store	Domain	ID	Name
		System	(global)	tomcat-ssh	deployer
		System	(global)	docker_hub	shivshankar7977/*****

Stores scoped to Jenkins

P	Store	Domains
	System	(global)

Icon: S M L



16. Test Plan

Project execution on physical machine.

The screenshot shows the Jenkins Pipeline Python-Cl-CD-Pipeline stage view. The top navigation bar includes links like 'Not secure | 192.168.30.60:8080/job/Python-Cl-CD-Pipeline/' and various Microsoft-related links. The main interface features a 'Status' button, 'Changes' link, 'Build Now' button, 'Configure' link, 'Delete Pipeline' link, 'Add description' link, and a 'Disable Project' link. A search bar at the top right says 'Search (CTRL+K)'. Below these are links for 'Full Stage View', 'Rename', and 'Pipeline Syntax'. The central part of the screen is the 'Stage View' table, which displays build times for six recent builds (#75 to #70). The columns represent stages: build checkout, Stop previous running container, Build Docker Image, Deploy Docker Image, Test - Run Docker Container on Jenkins node, and Deploy to Test Server. Build #70 is highlighted in red, indicating failure, with '394ms' and '59ms' shown in the last two columns. The bottom of the screen shows the Windows taskbar with various icons and system status.

	build checkout	Stop previous running container	Build Docker Image	Deploy Docker Image	Test - Run Docker Container on Jenkins node	Deploy to Test Server
#75 Aug 25, 2023, 12:21 AM	805ms	956ms	3s	17s	506ms	10s
#74 Aug 25, 2023, 12:05 AM	712ms	1s	3s	18s	532ms	12s
#73 Aug 25, 2023, 12:02 AM	719ms	805ms	3s	17s	536ms	12s
#72 Aug 24, 2023, 11:54 PM	766ms	807ms	3s	17s	394ms	59ms
#71 Aug 24, 2023, 11:50 PM	989ms	1s	3s	17s	536ms	12s

17. Application server

Tomcat server:

The screenshot shows the Apache Tomcat 9.0.80 web interface. At the top, there's a banner that says "If you're seeing this, you've successfully installed Tomcat. Congratulations!" with a cartoon cat icon. Below the banner, there are several sections: "Developer Quick Start" with links to "Tomcat Setup", "First Web Application", "Realms & AAA", and "JDBC Data Sources"; "Documentation" with links to "Tomcat 9.0 Documentation", "Tomcat 9.0 Configuration", and "Tomcat Wiki"; "Getting Help" with links to "FAQ and Mailing Lists" and "Tomcat Announcements". At the bottom, there are links for "Other Downloads" (Tomcat Connectors, Tomcat Native, Tomcat JDBCPool, Tomcat Overlay), "Other Documentation" (Tomcat Connectors, most Javadoc, Tomcat Native, Tomcat JDBCPool, Tomcat Overlay), "Get Involved" (Overview, Source Repositories, Mailings Lists, etc.), "Miscellaneous" (Contact, Legal, Subscriptions, Thanks), and "Apache Software Foundation" (Home Page, Interface, Apache Home, Resources). The footer includes a copyright notice: "Copyright ©1999-2023 Apache Software Foundation. All Rights Reserved".

18. Results

Screenshot of the Jenkins Pipeline for Python deployment interface. The pipeline consists of six stages: build checkout, Stop previous running container, Build Docker Image, Deploy Docker Image, Test - Run Docker Container on Jenkins node, and Deploy to Test Server. The average stage times are 4s, 1s, 15s, 4s, 2s, and 13s respectively. The pipeline history shows one build from Aug 28, 2023, at 11:11 PM.

Screenshot of a web browser showing the Jenkins pipeline results. The URL is 192.168.30.63:5001/hello/DeVops. The page displays the Jenkins logo and a message indicating the site has been visited 33 times.

Screenshot of a web browser displaying the output of a Flask application. The title "Welcome to Flask APP" is shown in yellow. Below it, the project topic "PROJECT TOPIC :- " Deployment Of CI-CD Pipeline On Cloud " is displayed in red. An illustration of two people working on a laptop with various icons (CSS, JS, DB, etc.) is shown. A group of members is listed with their names and IDs:

1. Ms. RUCHI BHUTAD	[230340127005]
2. Mr. KISHOR YEVALE	[230340127014]
3. Mr. BHARAT BANSODE	[230340127032]
4. Mr. SHIVSHANKAR GADEODHE	[230340127034]
5. Mr. AKSHAY GAIKWAD	[230340127035]
6. Ms. TRUPTI KOKANE	[230340127052]



C-DAC ACTS, C-DAC Innovation Park Panchvati, Pashan, Pune-411008

19. Conclusion

In conclusion, the deployment of a CI/CD pipeline on the cloud represents a transformative approach to software development and deployment. Through this project, we have successfully designed, implemented, and harnessed the power of cloud infrastructure to create a seamless and automated pipeline that has revolutionized our development processes.

By leveraging cloud resources, we have established a dynamic and scalable environment that caters to the entire software development lifecycle. The CI/CD pipeline has enabled us to continuously integrate code changes, rigorously test applications, and deploy them with utmost efficiency. This has resulted in accelerated delivery times, heightened software quality, and enhanced productivity across our development teams.

The multi-stage pipeline, encompassing source code management, build automation, testing, and deployment, has provided a comprehensive framework for orchestrating complex workflows.

In summary, the deployment of a CI/CD pipeline on the cloud has ushered in a new era of agility, reliability, and security in our software development endeavors. This project serves as a testament to the possibilities that emerge when cutting-edge cloud technology converges with modern software delivery practices. As we move forward, we are well-equipped to rapidly deploy new features, adapt to evolving requirements, and continuously innovate, all while upholding the highest standards of security and quality.

20. References

1. Reference for Docker
<https://docs.docker.com/build/guide/>
2. Jenkins Documentation for pipeline <https://www.jenkins.io/doc/tutorials/#pipeline>
3. References from GitHub
<https://docs.github.com/>
4. AWS Documentation
<https://docs.aws.amazon.com/>
5. PyCharm Documentation
<https://www.jetbrains.com/help/pycharm/getting-started.html>
6. python flask documentation
<https://flask.palletsprojects.com/>