Experiment No 2

Vedant Jayesh Oza
TY-IT-T4_60

Aim :- To perform Continuous Integration & Continuous Delivery using Jenkins

Objective :-
1. To understand CI/CD
2. To know SDLC

Theory :-

CI/CD (Continuous Integration and Continuous Delivery/Deployment) is a software development practice that involves regularly integrating code changes into a central repository and automatically building, testing, and deploying those changes. The goal of CI/CD is to catch and fix errors as early as possible in the development process, and to deliver new features and updates to customers faster and more frequently.

Jenkins is an open-source automation server that is used to automate repetitive tasks in software development, such as building, testing, and deploying code changes. It is particularly useful for implementing Continuous Integration (CI) and Continuous Delivery/Deployment (CD) workflows.

There are many benefits of implementing a CI/CD pipeline, including:

1. Faster and more frequent releases: By automating the build, test, and deployment process, teams can deliver new features and updates to customers faster and more frequently.
2. Improved quality: By catching and fixing errors early in the development process, teams can deliver higher-quality software. Automated testing and continuous monitoring also help to catch issues before they make it to production.
3. Reduced risk: Automating the build, test, and deployment process helps to reduce the risk of human error, minimizing downtime and reducing the potential for bugs and other issues.

A Jenkins workflow is a set of steps or tasks that are automated using the Jenkins CI/CD tool. The exact steps and tools used in a Jenkins workflow will vary depending on the specific project and technology stack, but a typical Jenkins workflow might include the following steps:

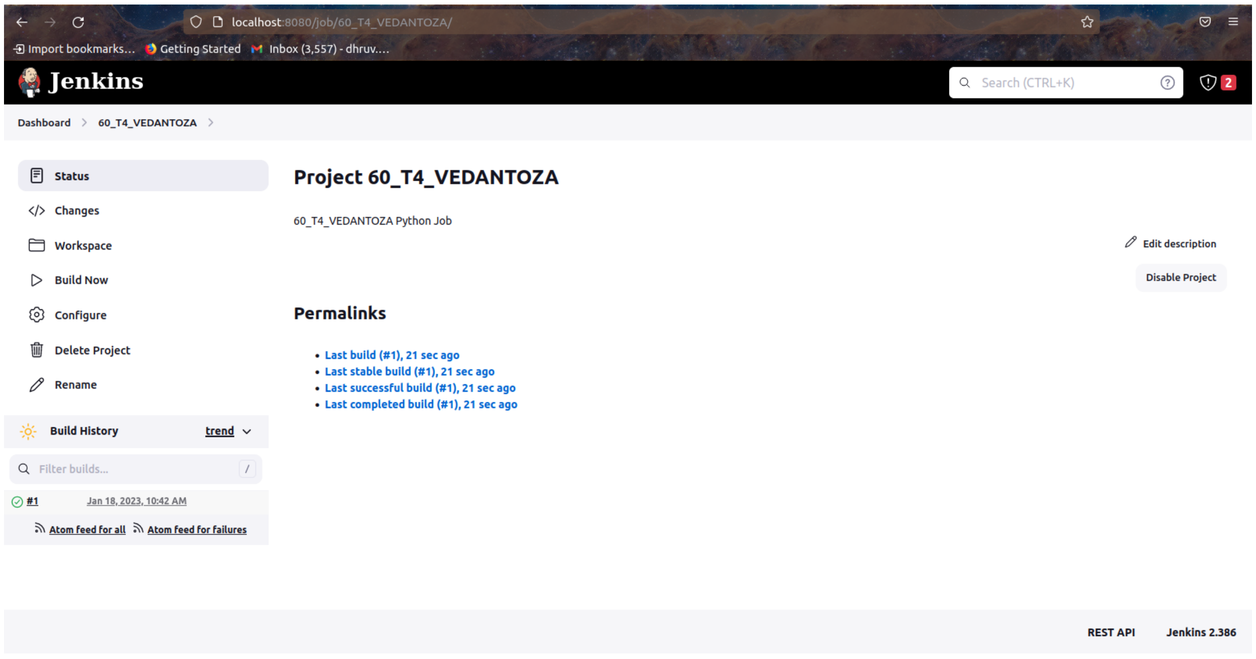
1. Code Integration: Code is integrated into a central repository, such as Git, multiple times a day.
2. Build: Code is automatically built using a build tool such as Maven or Gradle.
3. Test: Automated tests such as unit tests, integration tests, and acceptance tests are run to ensure that the code is working as expected.
4. Deployment: The code is deployed to a staging or production environment.
5. Continuous Monitoring: The application is continuously monitored for errors, bugs, and other issues.
6. Feedback: Feedback is provided to the development team with issues found.
7. Repeat: The process is repeated for every code change.

A pipeline is a set of instructions for automating the process of building, testing, and deploying software. In the context of CI/CD, a pipeline defines the steps that are taken to go from code changes to a deployed application.

There are several types of pipelines, including:

1. Build pipelines: These pipelines focus on automating the process of compiling and building code.
2. Test pipelines: These pipelines focus on automating the process of testing code, including unit tests, integration tests, and acceptance tests.
3. Deployment pipelines: These pipelines focus on automating the process of deploying code to different environments, such as development, staging, and production.
4. Hybrid pipelines: These pipelines combine elements of build, test and deployment pipelines into a single automated workflow.
5. Multi-branch pipelines: These pipelines are used to automate the build, test and deployment process for multiple branches in a version control system.

Output 1 :-



localhost:8080/job/60_T4_VEDANTOZA/

Dashboard > 60_T4_VEDANTOZA >

Project 60_T4_VEDANTOZA

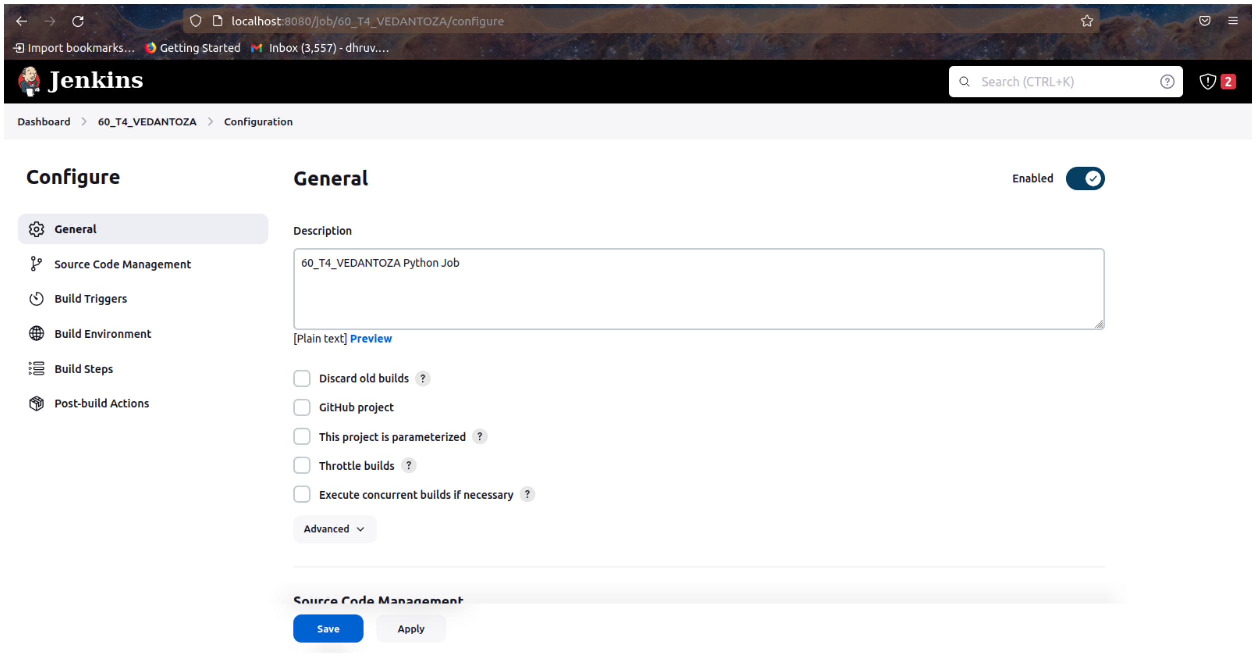
60_T4_VEDANTOZA Python Job

Permalinks

- Last build (#1), 21 sec ago
- Last stable build (#1), 21 sec ago
- Last successful build (#1), 21 sec ago
- Last completed build (#1), 21 sec ago

Build History: #1 Jan 18, 2023, 10:42 AM

Output 2:-



localhost:8080/job/60_T4_VEDANTOZA/configure

Dashboard > 60_T4_VEDANTOZA > Configuration

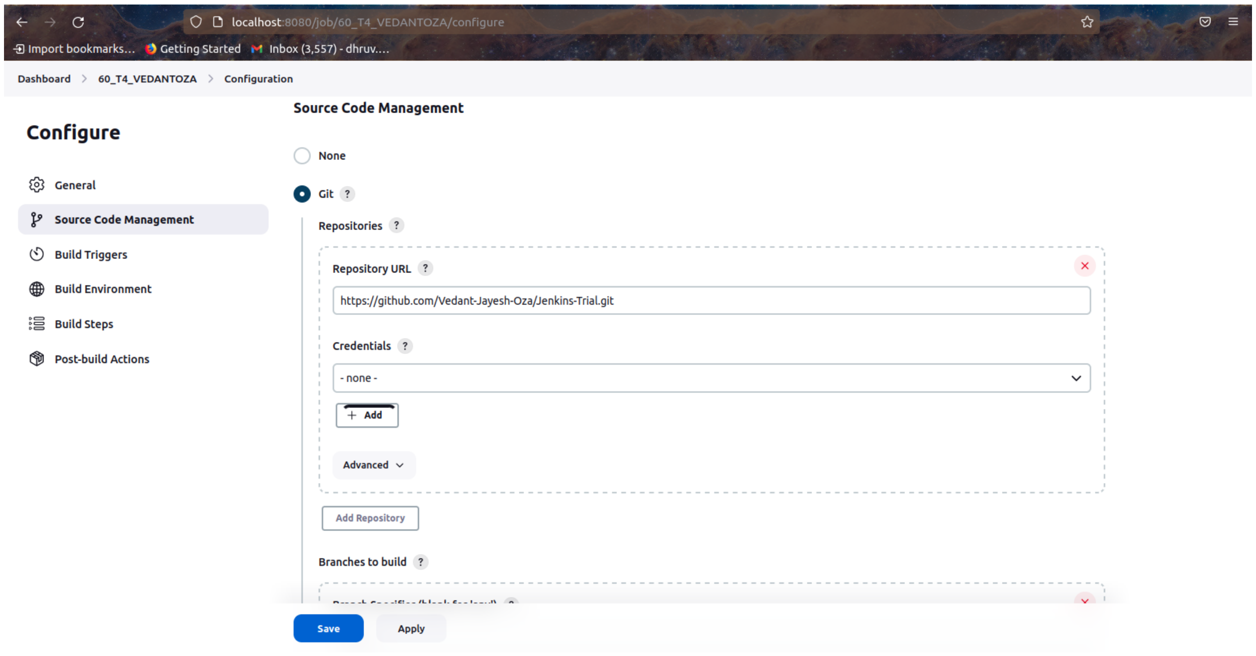
General Enabled

Description: 60_T4_VEDANTOZA Python Job

[Plain text] Preview

- [ ] Discard old builds
- [ ] GitHub project
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Execute concurrent builds if necessary

Output 3:-


localhost:8080/job/60_T4_VEDANTOZA/configure

Dashboard > 60_T4_VEDANTOZA > Configuration

Configure

General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions

Source Code Management

None [✓ Git]

Repositories

Repository URL
https://github.com/Vedant-Jayesh-Oza/Jenkins-Trial.git

Credentials
- none -

Add

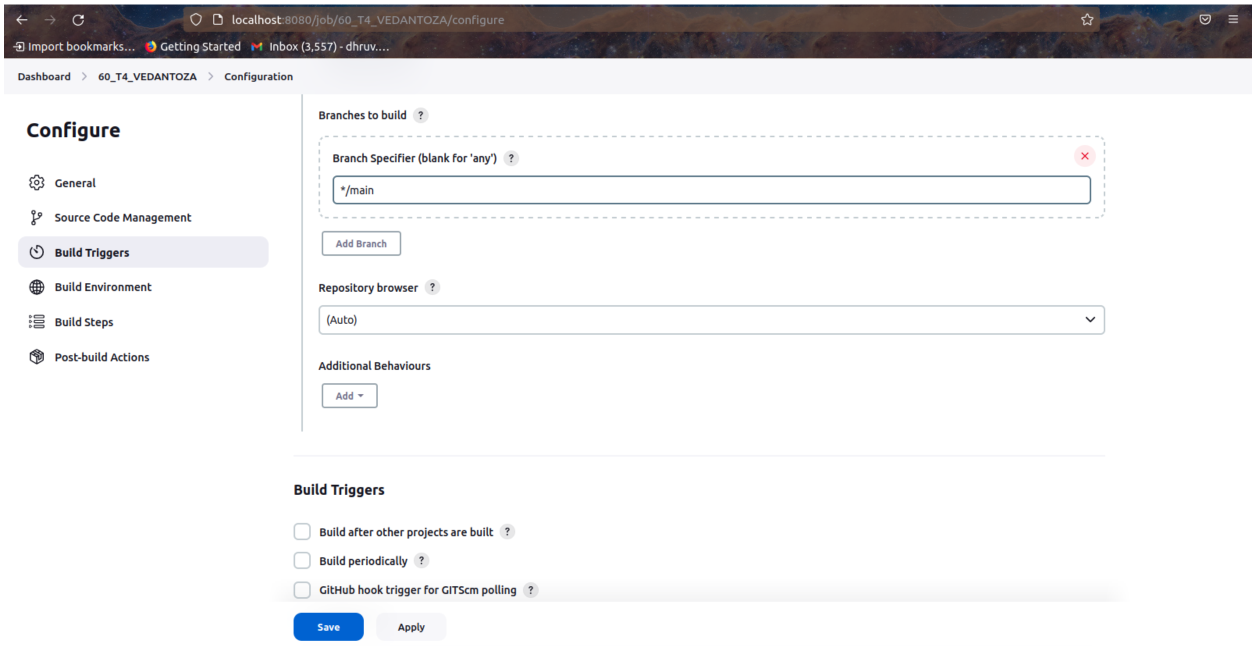
Advanced

Add Repository

Branches to build

Save Apply

Output 4:-


localhost:8080/job/60_T4_VEDANTOZA/configure

Dashboard > 60_T4_VEDANTOZA > Configuration

Configure

General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions

Branches to build

Branch Specifier (blank for 'any')
*/main

Add Branch

Repository browser
(Auto)

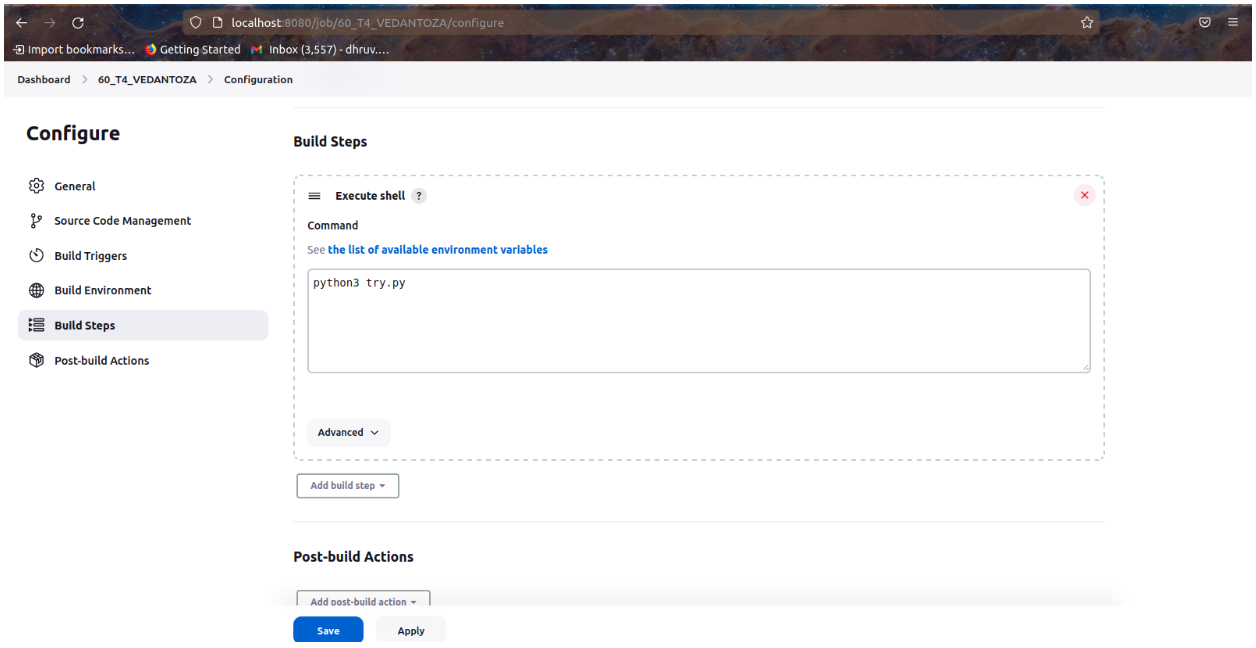
Additional Behaviours
Add

Build Triggers

- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling

Save Apply

Output 5 :-


localhost:8080/job/60_T4_VEDANTOZA/configure

Dashboard > 60_T4_VEDANTOZA > Configuration

Configure

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

```
python3 try.py
```

Advanced

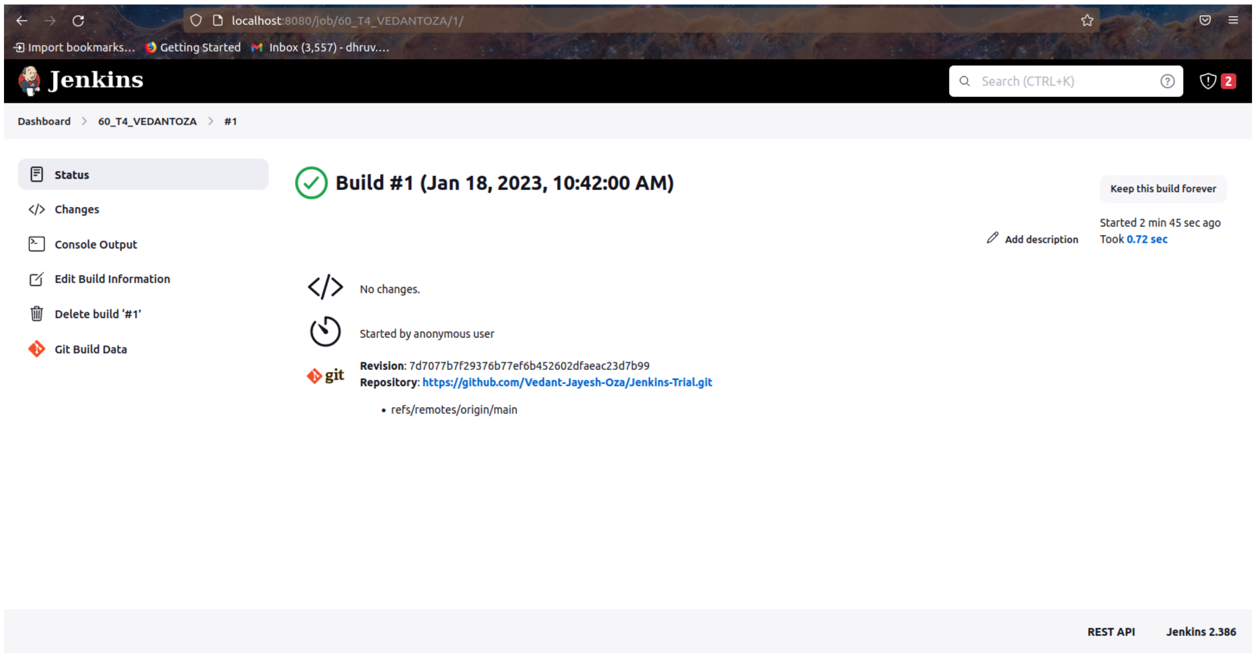
Add build step

Post-build Actions

Add post-build action

Save Apply

Output 6 :-


localhost:8080/job/60_T4_VEDANTOZA/1/

Jenkins

Dashboard > 60_T4_VEDANTOZA > #1

Status
Changes
Console Output
Edit Build Information
Delete build '#1'
Git Build Data

Build #1 (Jan 18, 2023, 10:42:00 AM)

Keep this build forever

Add description

Started 2 min 45 sec ago
Took 0.72 sec

No changes.

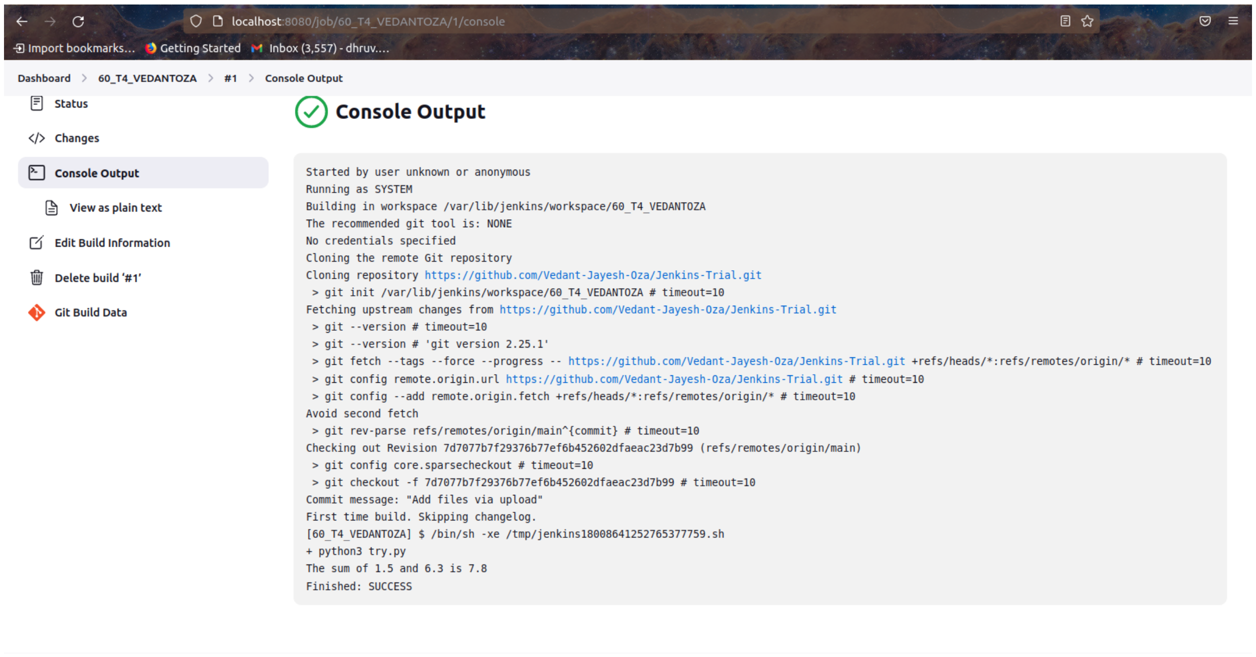
Started by anonymous user

Revision: 7d7077b7f29376b77ef6b452602dfaeac23d7b99
Repository: https://github.com/Vedant-Jayesh-Oza/Jenkins-Trial.git

- refs/remotes/origin/main

REST API Jenkins 2.386

Output 7:-



```
Started by user unknown or anonymous
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/60_T4_VEDANTOZA
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/Vedant-Jayesh-Oza/Jenkins-Trial.git
 > git init /var/lib/jenkins/workspace/60_T4_VEDANTOZA # timeout=10
Fetching upstream changes from https://github.com/Vedant-Jayesh-Oza/Jenkins-Trial.git
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
 > git fetch --tags --force --progress -- https://github.com/Vedant-Jayesh-Oza/Jenkins-Trial.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/Vedant-Jayesh-Oza/Jenkins-Trial.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 7d7077b7f29376b77ef6b452602dfaeac23d7b99 (refs/remotes/origin/main)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 7d7077b7f29376b77ef6b452602dfaeac23d7b99 # timeout=10
Commit message: "Add files via upload"
First time build. Skipping changelog.
[60_T4_VEDANTOZA] $ /bin/sh -xe /tmp/jenkins18008641252765377759.sh
+ python3 try.py
The sum of 1.5 and 6.3 is 7.8
Finished: SUCCESS
```

Outcome:-
1.Install Jenkins
2.Use Jenkins Server for Continuous Intergration.

Conclusion :- In conclusion, the experiment on Continuous Integration and Continuous Delivery using Jenkins has demonstrated the effectiveness of using a CI/CD pipeline in software development. By using Jenkins as the CI/CD tool, we were able to automate the build, test, and deployment of our application, reducing the time and effort required to manually perform these tasks. The use of Jenkins also allowed us to easily integrate with other tools such as Git for version control and JUnit for testing, further streamlining the development process. Overall, the experiment has shown that implementing a CI/CD pipeline with Jenkins can greatly improve the efficiency and reliability of software development, making it an essential tool for any modern development team.