EXPERIMENT : 1

Date of Performance	
Date of Submission	

AIM

To understand DevOps: Principles, Practices, and DevOps Engineer Role and Responsibilities.

PROBLEM DEFINITION

Explore DevOps principles, practices, and the responsibilities of a DevOps Engineer.

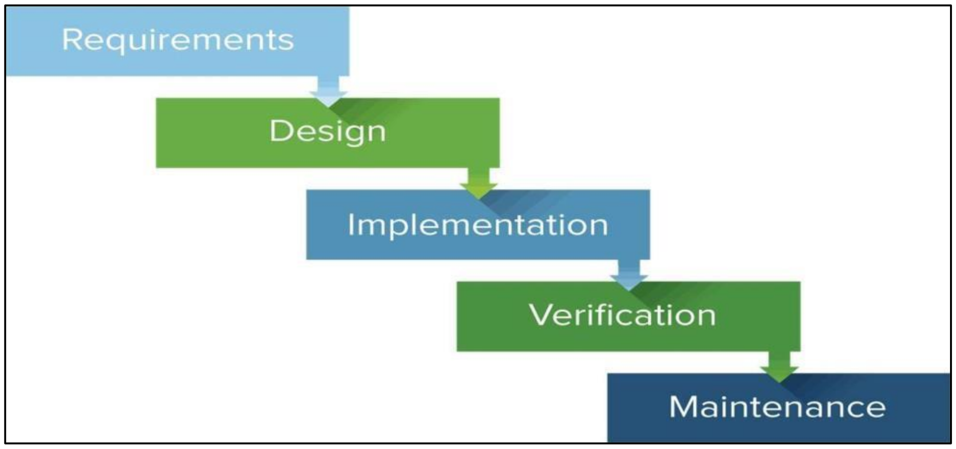
THEORY

EVOLUTION OF SOFTWARE DEVELOPMENT OVER THE YEARS :

The WATERFALL METHODOLOGY is a traditional, linear approach to software development.

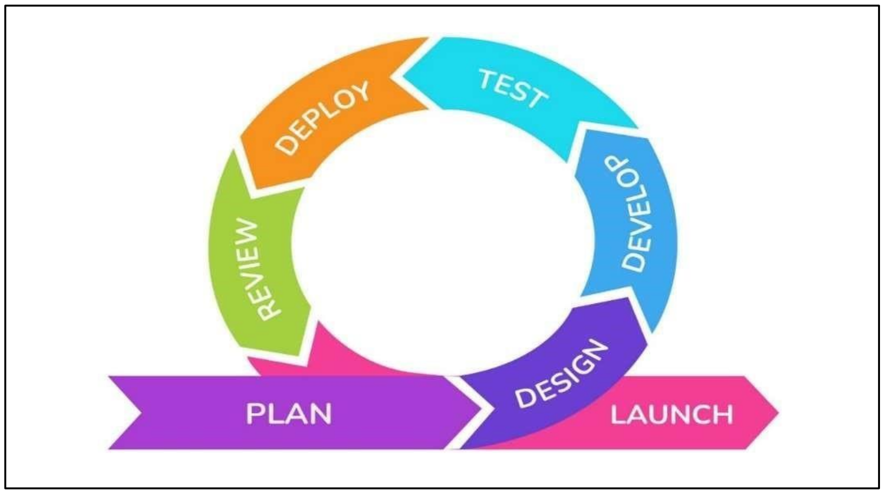
- Linear Process: Follows a sequential, step-by-step approach where each phase must be completed before the next begins.
- No Overlapping Stages: Each stage is strictly dependent on the previous one, and once a stage is completed, it cannot be revisited.
- Clear Documentation: Extensive documentation is produced at every stage, ensuring that the requirements and solutions are clearly defined.
- Limited Flexibility: Changing requirements during the development process is difficult, as each phase relies on the completion of the previous one.
- Easy to Manage: Its structured nature makes it easy for managers to track progress, deadlines, and costs.
- Best for Stable Projects: It is ideal for projects with well-understood requirements that are unlikely to change during development.

- Time-Consuming: The linear nature can result in longer project timelines due to the lack of iteration and flexibility.



The <u>AGILE METHODOLOGY</u> is a flexible, iterative approach to software development that emphasizes collaboration, adaptability, and continuous delivery.

- Iterative Process: Agile uses short, iterative cycles called sprints, producing functional software at the end of each sprint, allowing for ongoing feedback.
- Overlapping Stages: Development, testing, and design can occur simultaneously, enabling faster adjustments.
- Minimal Documentation: Focus is on working software rather than detailed documentation, prioritizing communication.
- High Flexibility: Agile adapts easily to changing requirements and feedback.
- Frequent Collaboration: Teams, stakeholders, and customers collaborate continuously to ensure alignment.
- Best for Uncertain Projects: Ideal for projects with evolving requirements, where flexibility is crucial.
- Faster Delivery: Enables frequent releases by delivering working software at the end of each sprint.
- Continuous Improvement: Regular retrospectives after each sprint help the team improve processes and performance.



WHY DEVOPS?

1. Bridging Development and Operations: DevOps was created to eliminate the gap between development and operations teams, which caused delays and errors in software releases.

2. Streamlining Delivery: By promoting collaboration, automation, and continuous feedback, DevOps accelerates software development, ensuring faster, high-quality product delivery while maintaining stability and security.

BENEFITS OF DEVOPS

- Faster Delivery: Continuous integration and delivery (CI/CD) pipelines enable rapid, frequent releases.
- Improved Collaboration: Breaks down silos between teams, leading to better communication and alignment.
- Increased Efficiency: Automation of repetitive tasks reduces manual errors and accelerates processes.
- Higher Quality: Early bug detection through continuous testing improves the reliability and stability of software.
- Scalability and Flexibility: DevOps practices like Infrastructure as Code (IaC) ensure scalable and consistent infrastructure management.
- Enhanced Security: Integrating security early in the development lifecycle (DevSecOps) ensures compliance and reduces vulnerabilities.

DEVOPS PRINCIPLES

DevOps emphasizes a culture of shared responsibility and collaboration between development and operations teams. Automation reduces manual tasks, increases speed, and minimizes errors. Continuous Integration and Delivery (CI/CD) ensures frequent testing, integration, and deployment for faster, reliable releases. DevOps is customerfocused, delivering updates based on feedback to enhance software quality.

DEVOPS PRACTICES

DevOps practices are designed to support the principles of the methodology:

- Infrastructure as Code (IaC) enables teams to manage and provision infrastructure using code, ensuring consistent and repeatable environments across development, testing, and production.
- Continuous Testing automates the testing process, allowing for early detection of bugs and issues before they reach production. This ensures higher quality code with every release.
- Monitoring and Logging help track the performance and health of systems in realtime, enabling teams to quickly respond to issues and maintain system reliability and uptime. This also feeds into continuous feedback loops for constant improvement.

DEVOPS ENGINEER ROLE AND RESPONSIBILITIES

A DevOps engineer is responsible for bridging the gap between development and operations. Their role includes:

- Automation: Automating workflows, including code deployments, system configurations, and environment setup, to streamline processes.
- Managing CI/CD Pipelines: Setting up and maintaining continuous integration and delivery pipelines to ensure quick, reliable, and frequent releases of code to production.
- Monitoring Infrastructure: Implementing monitoring tools and techniques to ensure that the systems are performing optimally and identifying potential bottlenecks or failures before they occur.
- Collaboration: Working closely with both development and IT operations teams to create seamless communication and efficient workflows.
- Security and Compliance: Ensuring that the automation processes follow security and compliance requirements throughout the software delivery lifecycle.

CONCLUSION

We conclude that DevOps unites development and operations through automation, continuous integration, and deployment, with engineers driving improved software quality and delivery speed.

EXPERIMENT : 2

Date of Performance	
Date of Submission	

AIM

To understand Version Control System / Source Code Management, install git and create a GitHub account.

PROBLEM DEFINITION

Learn about version control systems and source code management by setting up Git and creating a GitHub account.

THEORY

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning-fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Some basic operations in Git are:

1. Initialize
2. Add
3. Commit
4. Pull
5. Push

Some advanced Git operations are:

1. Branching
2. Merging
3. Rebasing

Installation of Git

1. In Windows, download Git from https://git-scm.com/ and perform the straightforward installation.
2. In Ubuntu, install Git using `$ sudo apt install git`. Confirm the version after installation with `$ git --version`.

Execution

To perform version control, create a directory named `dvcs` (Distributed Version Control System) and change the directory to `dvcs`:

```
$ mkdir git-dvcs
```

```
$ cd git-dvcs/
```

Check the user information using:

```
$ git config --global
```

Since there are no users defined, define them using the following commands:

```
$ git config --global user.name "abcd"
```

```
$ git config --global user.email “xyz….@gmail.com”
```

Check the list of users: $ git config

--global --list user.name=abcde

user.email=xyz….@gmail.com

Create a repository for version control named "git-demo-project":

$ mkdir git-demo-project

$ cd git-demo-project/

Initialize the repository:

$ git init

If you have an existing repository, delete the `.git` file and reinitialize it:

$ rm -rf .git/

$ git init

Add files to the repository:

$ git add .

To check the status of the repository:

$ git status

Commit the changes:

$ git commit -m "First Commit" Add

`index.html` to the directory $

git add .

Commit changes:

$ git commit -am "Express Commit"

Make changes to `index.html` and create a file `teststatus`:

```
$ nano index.html
$ touch teststatus
```

Discard changes:

```
$ git restore <file>
```

Add `index.html` and `teststatus`:

```
$ git add index.html
$ git add teststatus
$ git commit -am "Express commit"
```

View commit history:

```
$ git log
```

Create a repository on GitHub:

1. Open github.com and create an account.
2. After logging in, select "New repository" from the menu.
3. Specify a name for the repository and select the public option, then click "Create repository."

Fork the repository:

1. Log in with another account.
2. Copy and paste the URL of the repository.
3. Click "Fork" to clone it to another account.

Push changes to the web repository:

```
$ git remote add origin https://github.com/MSid01/TriviaQuiz.git
$ git remote show origin
```

$ git remote add origin https://github.com/bhushanjadhav1/Myrepository.git

$ git remote rm origin

$ git push -u origin master

Pull changes:

$ git pull

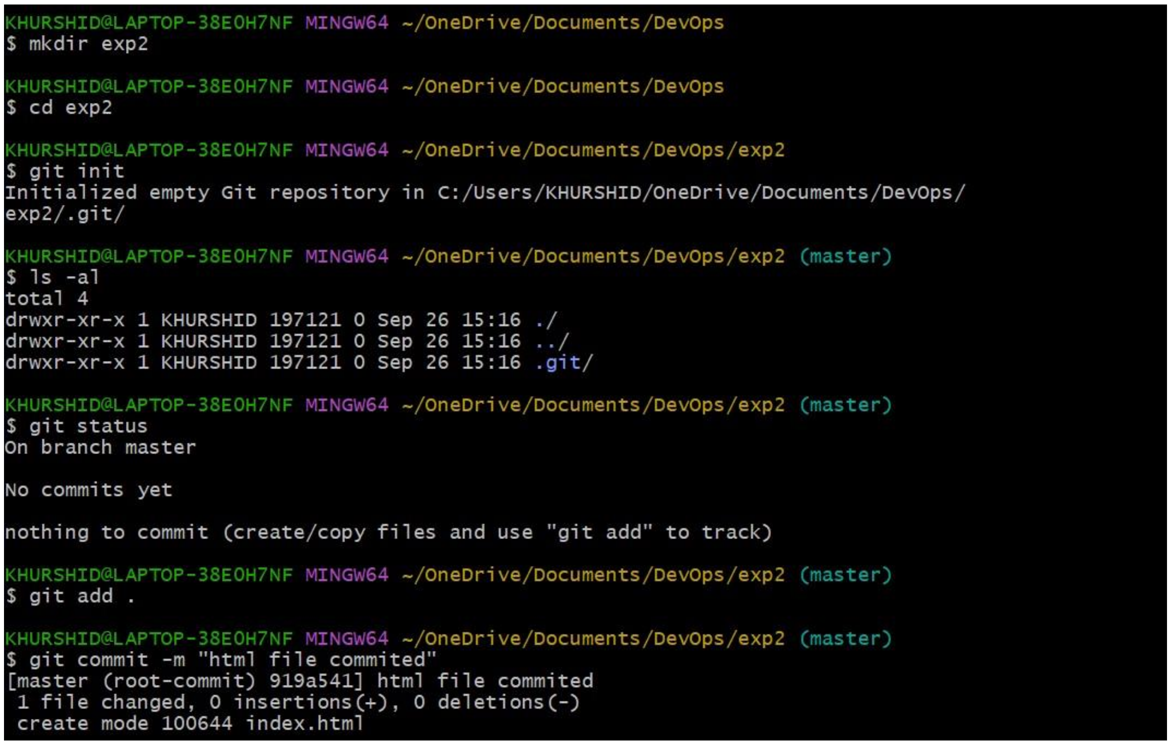
Fetch changes:

$ git fetch

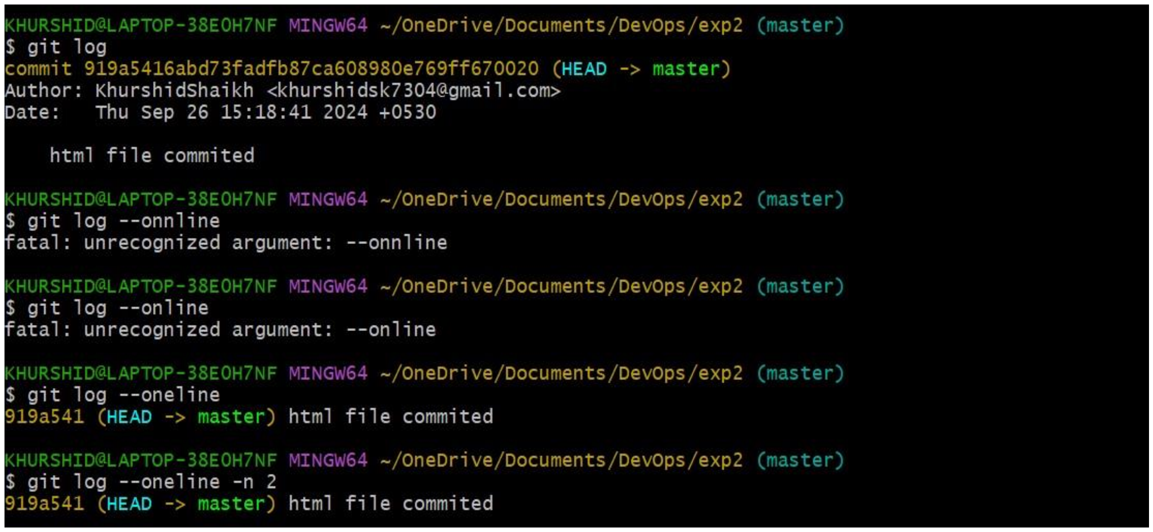
Merge fetched changes:

$ git merge origin/master

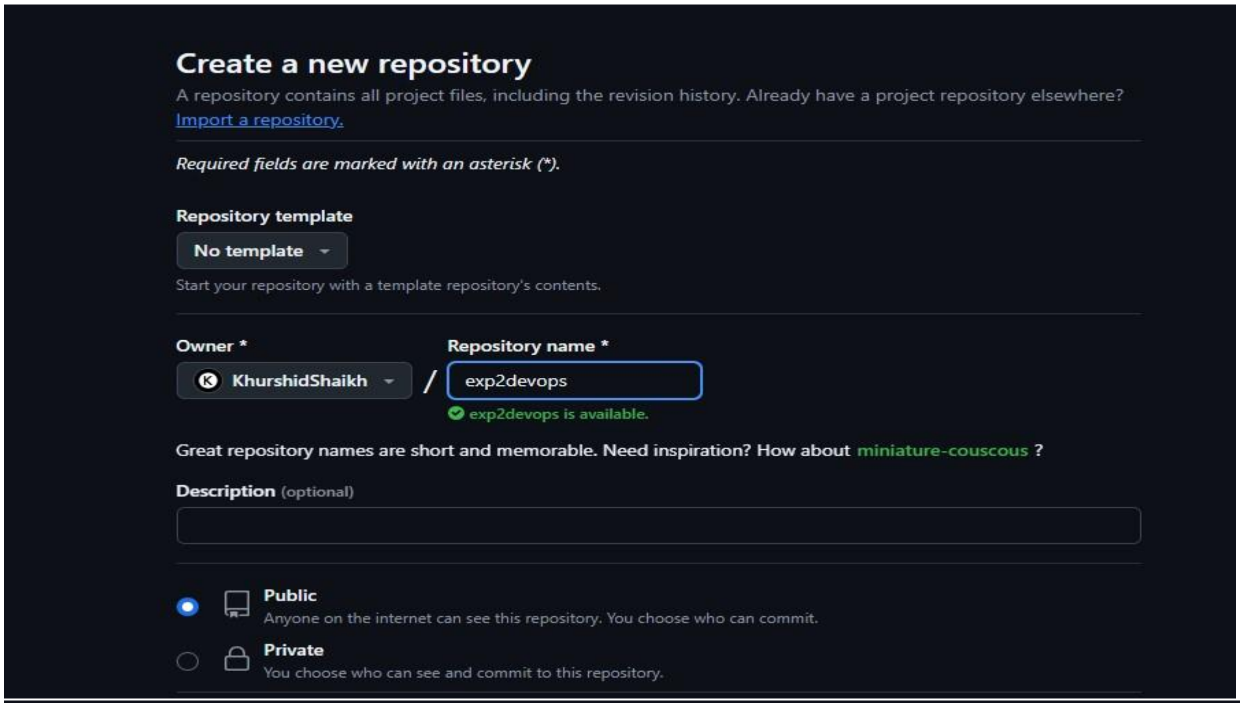
OUTPUT:



```
KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps
$ mkdir exp2

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps
$ cd exp2

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2
$ git init
Initialized empty Git repository in C:/Users/KHURSHID/OneDrive/Documents/DevOps/
exp2/.git/

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ ls -al
total 4
drwxr-xr-x 1 KHURSHID 197121 0 Sep 26 15:16 ./
drwxr-xr-x 1 KHURSHID 197121 0 Sep 26 15:16 ../
drwxr-xr-x 1 KHURSHID 197121 0 Sep 26 15:16 .git/

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ git add .

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ git commit -m "html file commited"
[master (root-commit) 919a541] html file commited
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
```



```
KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ git log
commit 919a5416abd73fadfb87ca608980e769ff670020 (HEAD -> master)
Author: KhurshidShaikh <khurshidsk7304@gmail.com>
Date:   Thu Sep 26 15:18:41 2024 +0530

    html file commited

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ git log --onnline
fatal: unrecognized argument: --onnline

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ git log --online
fatal: unrecognized argument: --online

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ git log --oneline
919a541 (HEAD -> master) html file commited

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ git log --oneline -n 2
919a541 (HEAD -> master) html file commited
```



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Repository template

No template

Start your repository with a template repository's contents.

Owner * KhurshidShaikh / **Repository name *** exp2devops

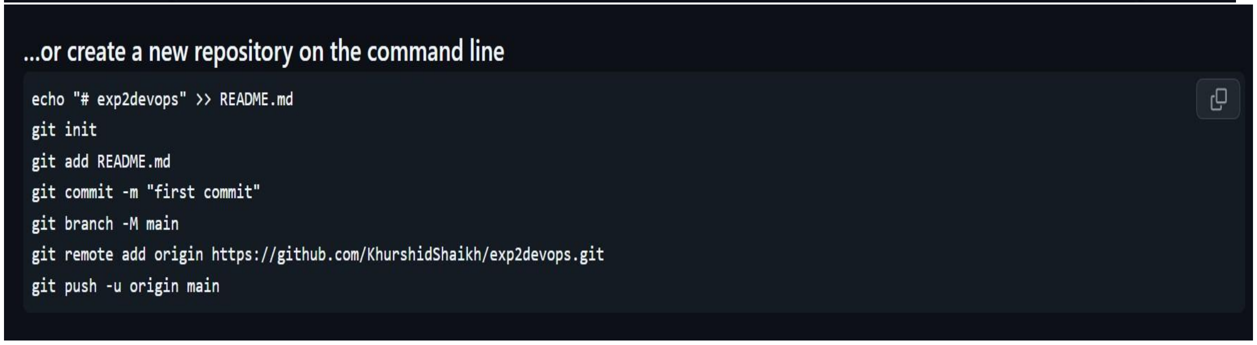
exp2devops is available.

Great repository names are short and memorable. Need inspiration? How about miniature-couscous ?

Description (optional)

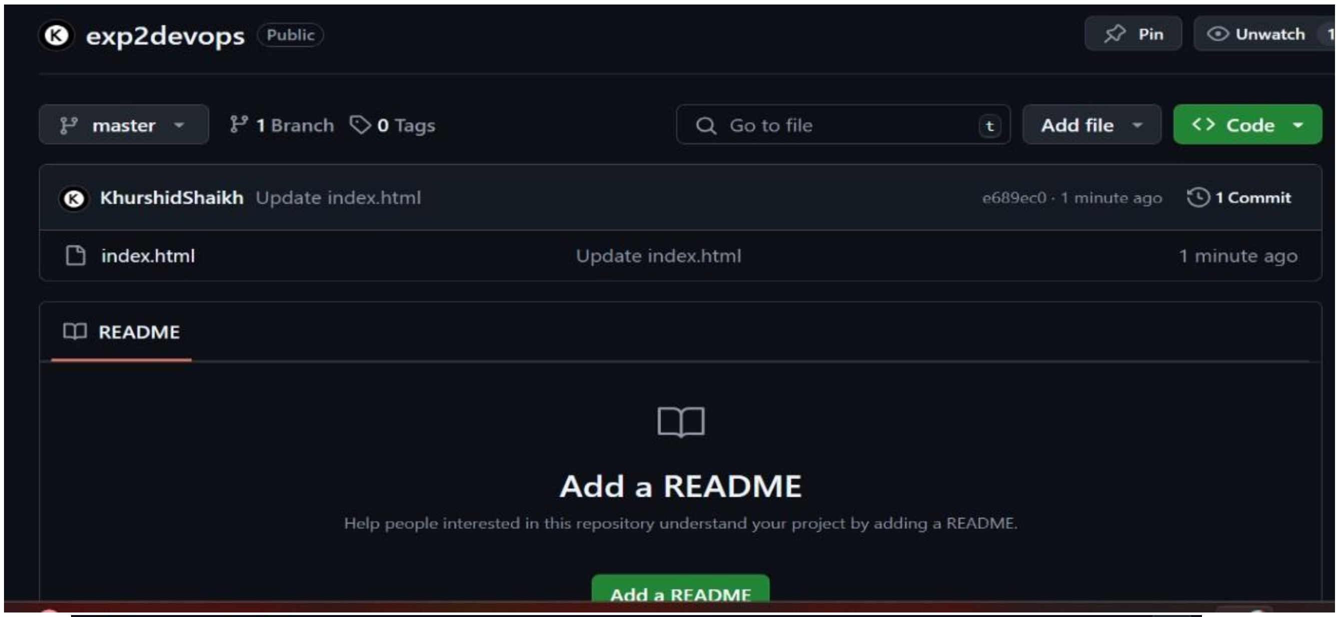
● **Public** Anyone on the internet can see this repository. You choose who can commit.

○ **Private** You choose who can see and commit to this repository.



...or create a new repository on the command line

```
echo "# exp2devops" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/KhurshidShaikh/exp2devops.git
git push -u origin main
```


exp2devops Public

master · 1 Branch · 0 Tags

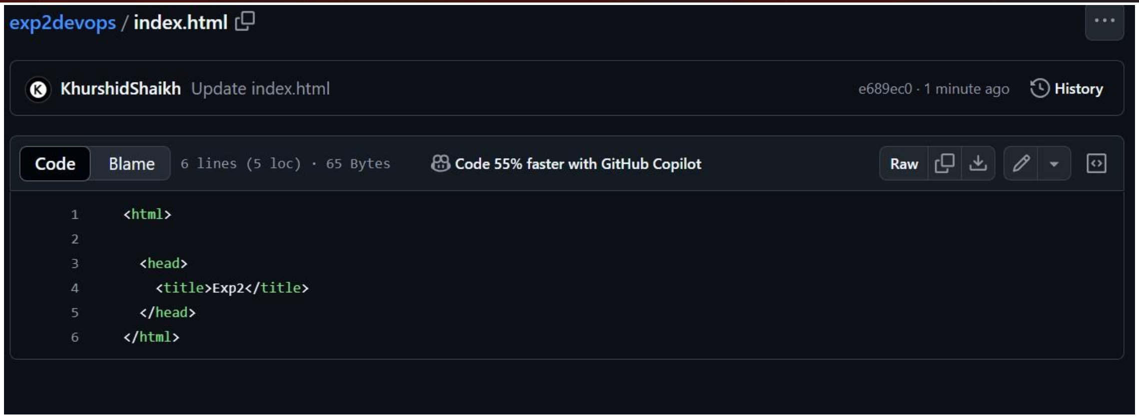
KhurshidShaikh Update index.html e689ec0 · 1 minute ago 1 Commit

index.html Update index.html 1 minute ago

README

Add a README

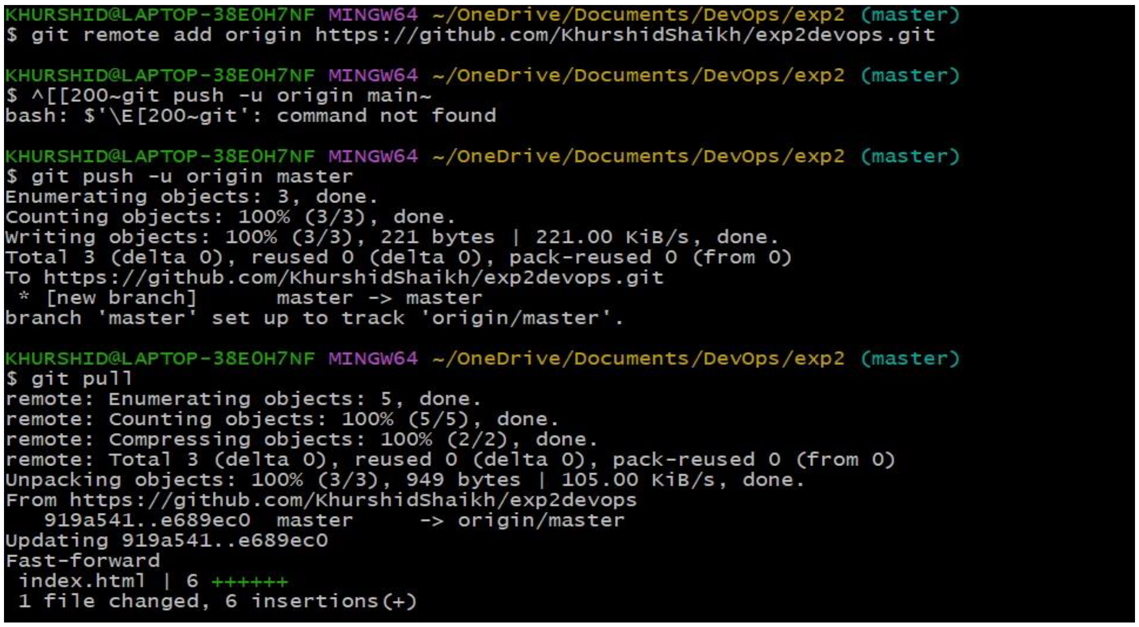
Help people interested in this repository understand your project by adding a README.


exp2devops / index.html

KhurshidShaikh Update index.html e689ec0 · 1 minute ago History

Code Blame 6 lines (5 loc) · 65 Bytes Code 55% faster with GitHub Copilot

```
<html>

  <head>
    <title>Exp2</title>
  </head>
</html>
```



```
KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ git remote add origin https://github.com/KhurshidShaikh/exp2devops.git

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ ^[[200~git push -u origin main~
bash: $'\E[200~git': command not found

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 221 bytes | 221.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/KhurshidShaikh/exp2devops.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp2 (master)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 949 bytes | 105.00 KiB/s, done.
From https://github.com/KhurshidShaikh/exp2devops
   919a541..e689ec0  master     -> origin/master
Updating 919a541..e689ec0
Fast-forward
 index.html | 6 ++++++
 1 file changed, 6 insertions(+)
```

CONCLUSION

Thus, we understood the Version Control System and Source Code Management by installing Git and creating a GitHub account, which facilitated efficient code management and collaboration.

EXPERIMENT : 3

AIM

To perform various GIT operations on local and remote repositories using GIT CheatSheet 4.

PROBLEM DEFINITION

Execute various Git operations on local and remote repositories with the help of a Git cheat sheet.

THEORY

What is GIT?

Git is the most widely used modern version control system in the world today. It is a mature, actively maintained open-source project originally developed in 2005 by Linus Torvalds, the famous creator of the Linux operating system kernel. A significant number of software projects, both commercial and open source, rely on Git for version control. Developers experienced with Git are well-represented in the pool of available software development talent, and it works well on a wide range of operating systems and Integrated Development Environments (IDEs).

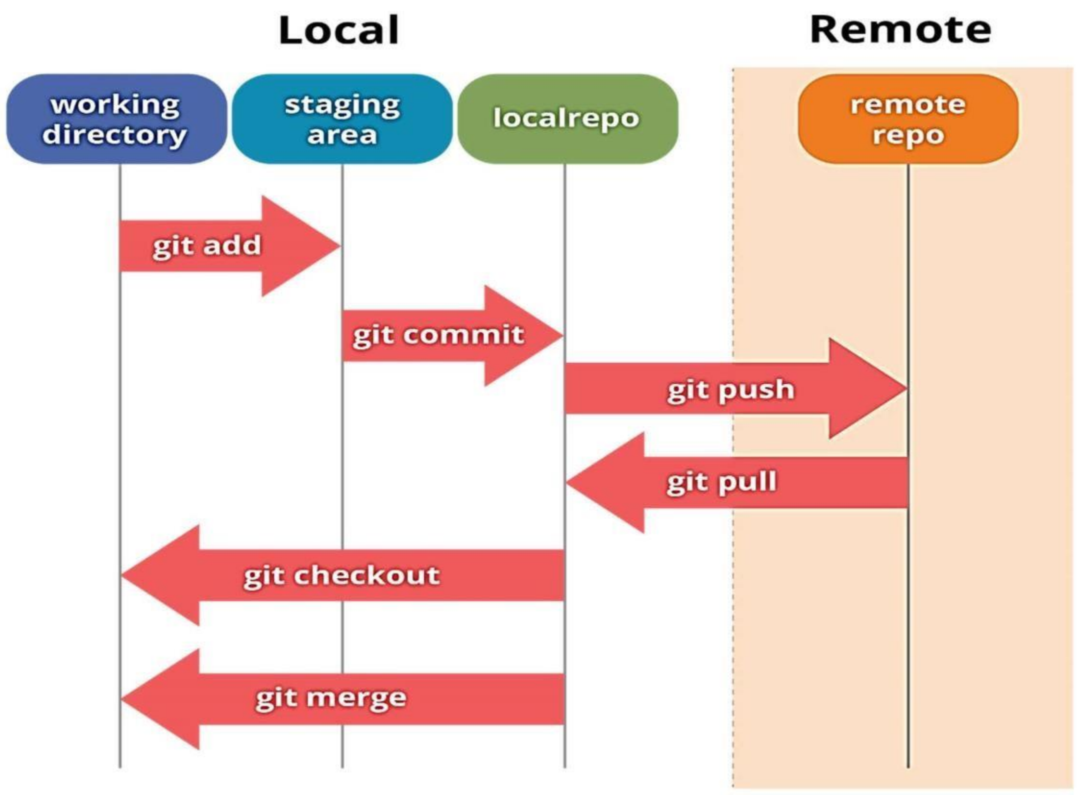
Git employs a distributed architecture, making it a Distributed Version Control System (DVCS). Unlike older version control systems such as CVS or Subversion (SVN), which have a single central repository for the full version history of the software, Git allows each developer's working copy of the code to be a repository that contains the complete history of all changes.

In addition to being distributed, Git is designed with performance, security, and flexibility in mind. Local repositories are physical, locally-managed repositories into which you can deploy artifacts. Using local repositories, Artifactory provides a central location to store your internal binaries. Repository replication enables sharing binaries with teams located in remote locations.A remote repository in Git, also known as a remote, is a Git repository hosted on the Internet or another network.

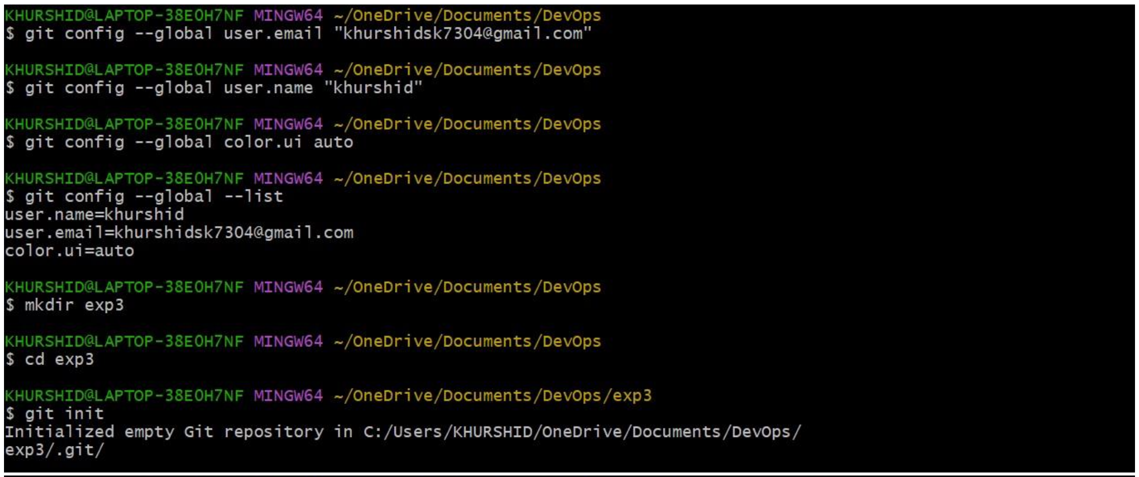
Here are some common Git operations:

1. `git init` - Initialize a new Git repository.
2. `git clone <repository>` - Clone an existing repository from a remote source.
3. `git add <file>` - Stage changes for the next commit.
4. `git commit -m "<message>"` - Commit staged changes with a descriptive message.
5. `git status` - Check the status of changes in the working directory and staging area.
6. `git log` - View the commit history.
7. `git branch` - List, create, or delete branches.
8. `git checkout <branch>` - Switch to a different branch.
9. `git merge <branch>` - Merge changes from one branch into the current branch.
10. `git pull` - Fetch and merge changes from a remote repository into the current branch.
11. `git push` - Upload local commits to a remote repository.
12. `git fetch` - Download changes from a remote repository without merging.
13. `git revert <commit>` - Create a new commit that undoes changes from a previous commit.
14. `git reset <file>` - Unstage a file from the staging area.
15. `git rm <file>` - Remove a file from the working directory and staging area.

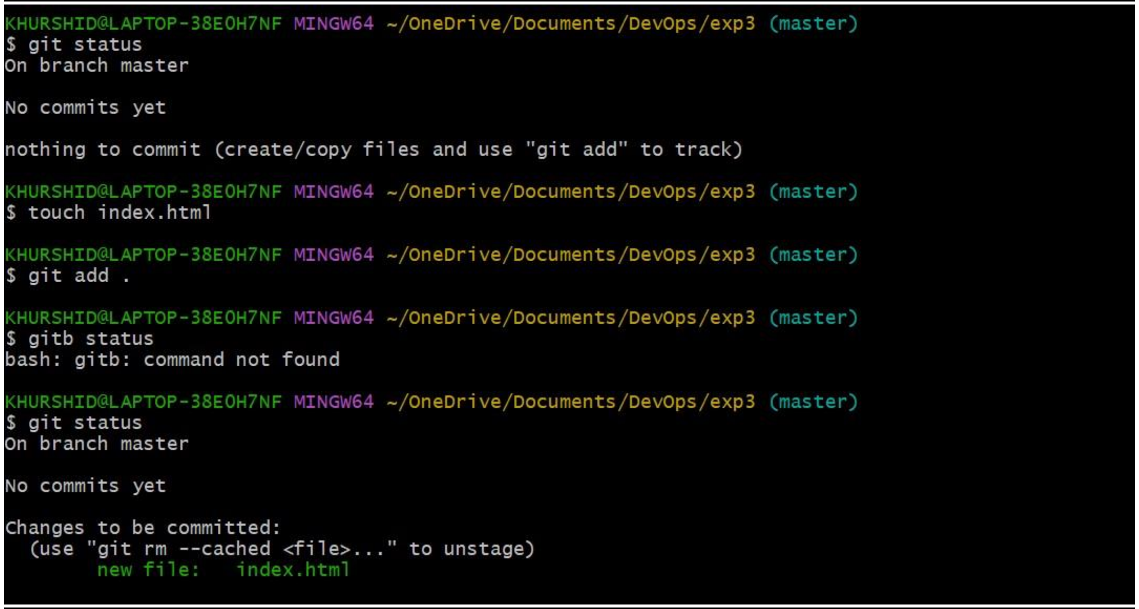
These operations cover most common tasks when working with Git repositories.


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout
git merge

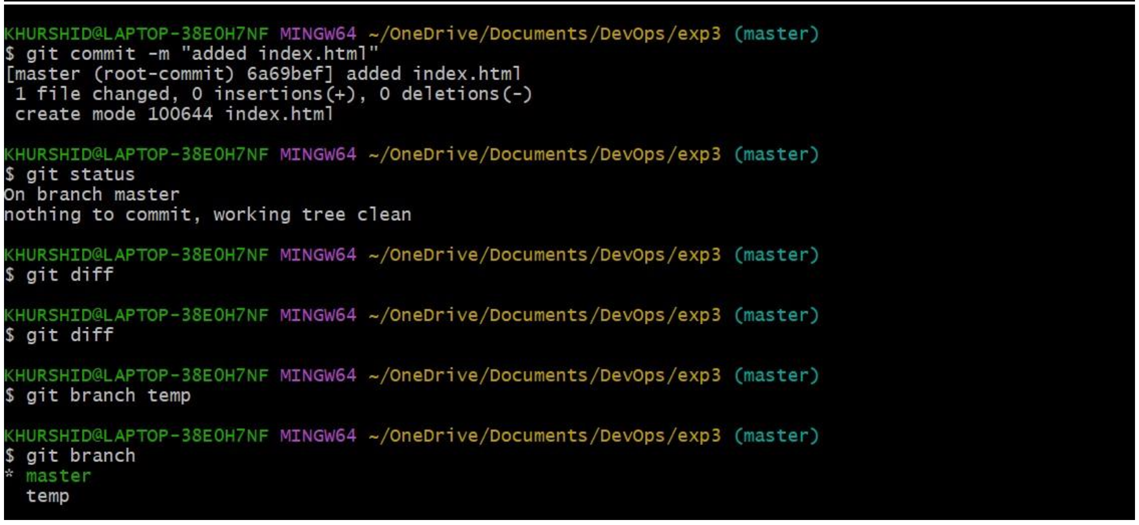
OUTPUT



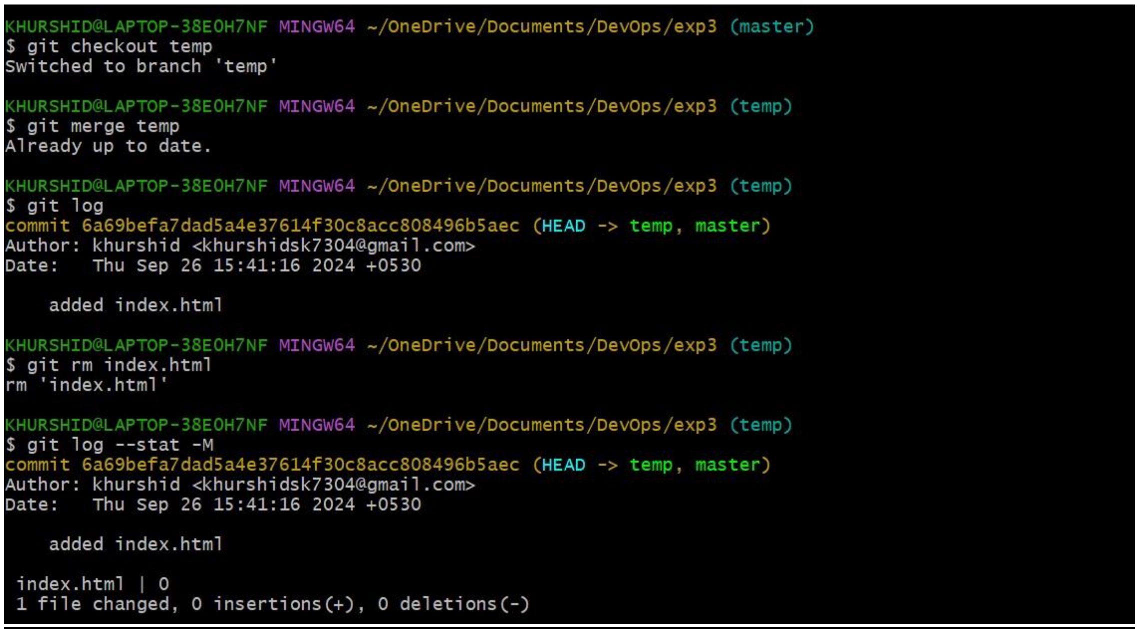
```
KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps
$ git config --global user.email "khurshidsk7304@gmail.com"

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps
$ git config --global user.name "khurshid"

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps
$ git config --global color.ui auto

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps
$ git config --global --list
user.name=khurshid
user.email=khurshidsk7304@gmail.com
color.ui=auto

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps
$ mkdir exp3

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps
$ cd exp3

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3
$ git init
Initialized empty Git repository in C:/Users/KHURSHID/OneDrive/Documents/DevOps/
exp3/.git/
```



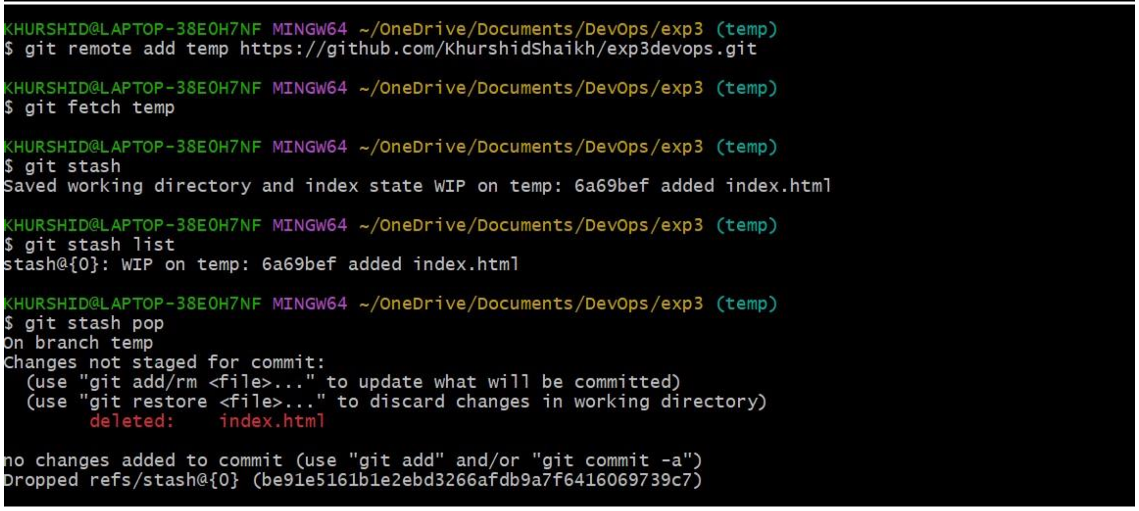
```
KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ touch index.html

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ git add .

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ gitb status
bash: gitb: command not found

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
```



```
KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ git commit -m "added index.html"
[master (root-commit) 6a69bef] added index.html
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ git status
On branch master
nothing to commit, working tree clean

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ git diff

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ git diff

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ git branch temp

KHURSHID@LAPTOP-38EOH7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ git branch
* master
  temp
```



```
KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (master)
$ git checkout temp
Switched to branch 'temp'

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (temp)
$ git merge temp
Already up to date.

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (temp)
$ git log
commit 6a69befa7dad5a4e37614f30c8acc808496b5aec (HEAD -> temp, master)
Author: khurshid <khurshidsk7304@gmail.com>
Date:   Thu Sep 26 15:41:16 2024 +0530

    added index.html

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (temp)
$ git rm index.html
rm 'index.html'

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (temp)
$ git log --stat -M
commit 6a69befa7dad5a4e37614f30c8acc808496b5aec (HEAD -> temp, master)
Author: khurshid <khurshidsk7304@gmail.com>
Date:   Thu Sep 26 15:41:16 2024 +0530

    added index.html

 index.html | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
```



```
KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (temp)
$ git remote add temp https://github.com/KhurshidShaikh/exp3devops.git

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (temp)
$ git fetch temp

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (temp)
$ git stash
Saved working directory and index state WIP on temp: 6a69bef added index.html

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (temp)
$ git stash list
stash@{0}: WIP on temp: 6a69bef added index.html

KHURSHID@LAPTOP-38E0H7NF MINGW64 ~/OneDrive/Documents/DevOps/exp3 (temp)
$ git stash pop
On branch temp
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    index.html

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (be91e5161b1e2ebd3266afdb9a7f6416069739c7)
```

CONCLUSION

Therefore, performing various Git operations on local and remote repositories using Git Cheat Sheets enabled effective code management and version control.

EXPERIMENT : 4

Date of Performance	
Date of Submission	

AIM

To understand Continuous Integration, install and configure Jenkins with Maven/Ant/Gradle to setup a build Job

PROBLEM DEFINITION

Implement continuous integration by installing and configuring Jenkins with Maven, Ant, or Gradle to establish a build job.

THEORY

Jenkins is a popular open-source tool for continuous integration and build automation. It allows executing a predefined list of steps, such as compiling Java source code and building a JAR from the resulting classes. The execution can be triggered by time or events, for instance, compiling your Java-based application every 20 minutes or after a new commit in the related Git repository. Possible steps executed by Jenkins include:

- Performing a software build using a build system like Apache Maven or Gradle

- Executing a shell script

- Archiving a build result

- Running software tests

Jenkins monitors the execution of these steps and can halt the process if any step fails. It can also send notifications in case of a build success or failure. Jenkins is extendable with additional plugins, such as those for building and testing Android applications.

Continuous integration is the process of integrating all development work as early as possible. The resulting artifacts are automatically created and tested, allowing errors to be identified early in the project. The Jenkins build server provides this functionality.How to Download Jenkins?

Steps to Install Jenkins:

1. Go to https://www.jenkins.io/download/ and select the platform (Windows in this case).

2. Go to the download location on your local computer and unzip the downloaded package. Double-click on the unzipped `jenkins.msi`. Note that using a WAR (Web Application Archive) is possible but not recommended.

3. In the Jenkins setup screen, click "Next."

4. Choose the installation location (default is `C:\Program Files (x86)\Jenkins`) and click "Next."

5. Once the installation is complete, click "Finish."

6. During installation, an info panel may appear indicating that a system reboot might be needed for a complete setup. Click "OK" when prompted.

How to Unblock Jenkins:

1. After completing the Jenkins installation, a browser tab will pop up asking for the initial Administrator password. Access Jenkins by navigating to

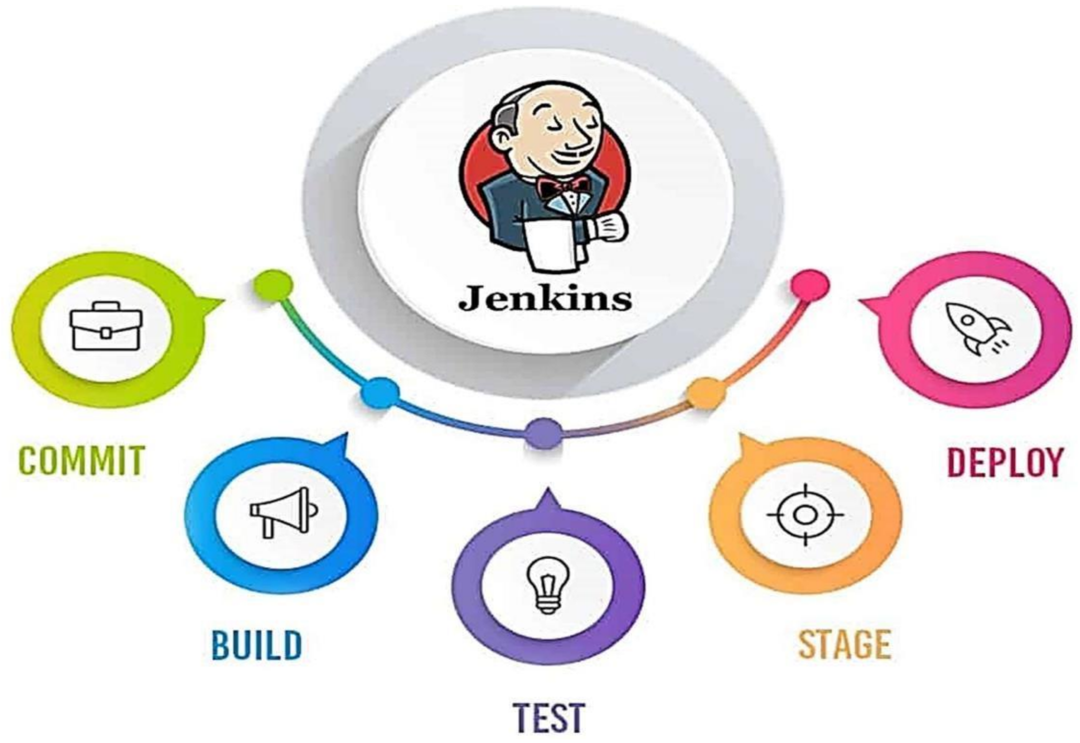
http://localhost:8080. If you can access this URL, Jenkins is successfully installed.

2. Find the initial Administrator password under the Jenkins installation path (set in Step 4). For the default location (`C:\Program Files (x86)\Jenkins`), the file `initialAdminPassword` is located at `C:\Program Files (x86)\Jenkins\secrets`. If a custom path was used, check that location for the `initialAdminPassword` file.

3. Open the `initialAdminPassword` file and copy its contents.

4. Paste the password into the browser's pop-up tab at

http://localhost:8080/login?form=%2F and click "Continue."



Customize Jenkins:

1. Click on the "Install suggested plugins" button to allow Jenkins to retrieve and install essential plugins. Jenkins will start downloading and installing all necessary plugins.

Note: You can choose the "Select Plugins to Install" option to manually select the plugins you want to install.

2. After installing the suggested plugins, the "Create First Admin User" panel will appear. Fill in the fields with desired account details and click "Save and Finish."

3. You will then be asked for URL information to configure the default instance path for Jenkins. Leave it as is to avoid confusion. If port 8080 is already in use, choose a different port for Jenkins, save the settings, and click "Save and Continue."

Congratulations! Jenkins is now successfully installed. Click the "Start using Jenkins" button to access your Jenkins instance, ready to create your first Jenkins jobs.

Adding Tools and Plugins:

1. Add Git.

2. Add Gradle.

3. Add Ant.

4. Add Maven.

Create a New Build Job in Jenkins:

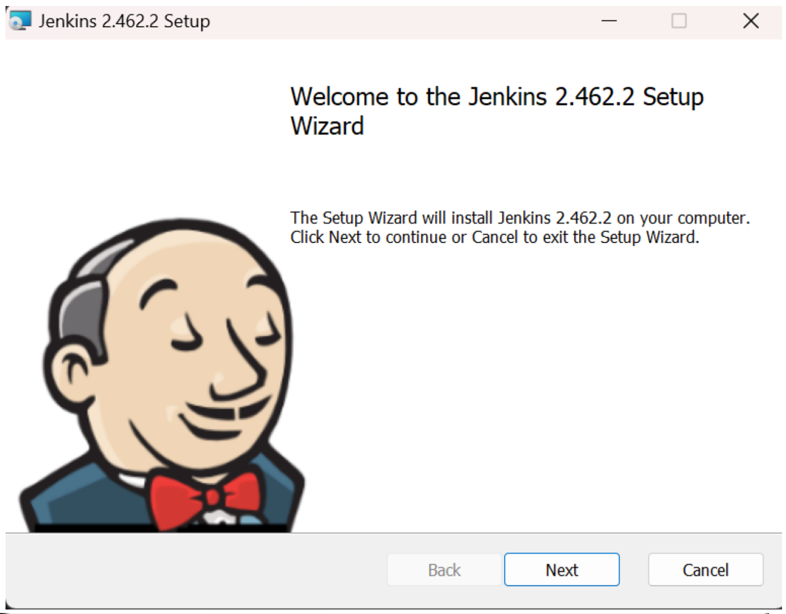
1. Connect to Jenkins for initial configuration by opening a browser and navigating to http://localhost:8080.

2. Copy the initial password from the file system of the server.

3. Select to install plugins, choosing "Install suggested Plugins" for a typical configuration.

4. Create an admin user and click "Save and Finish."

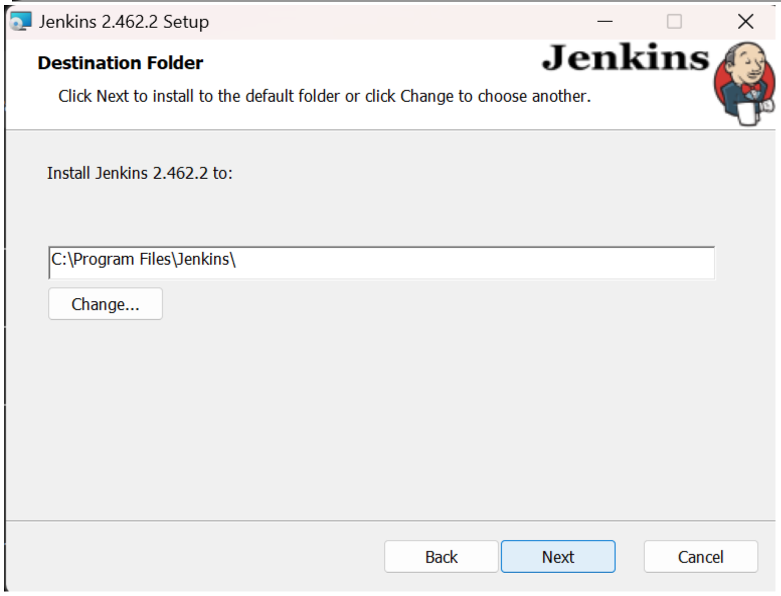
   Output:



Jenkins 2.462.2 Setup

Welcome to the Jenkins 2.462.2 Setup Wizard

The Setup Wizard will install Jenkins 2.462.2 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.



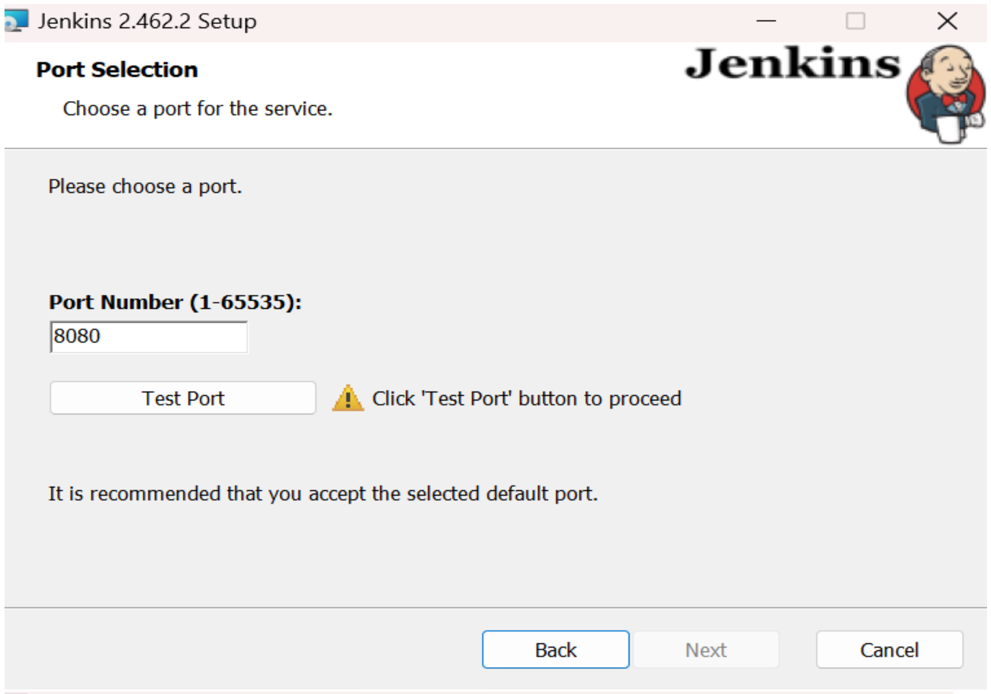
Jenkins 2.462.2 Setup

Destination Folder

Click Next to install to the default folder or click Change to choose another.

Install Jenkins 2.462.2 to:

C:\Program Files\Jenkins\


Jenkins 2.462.2 Setup

Port Selection

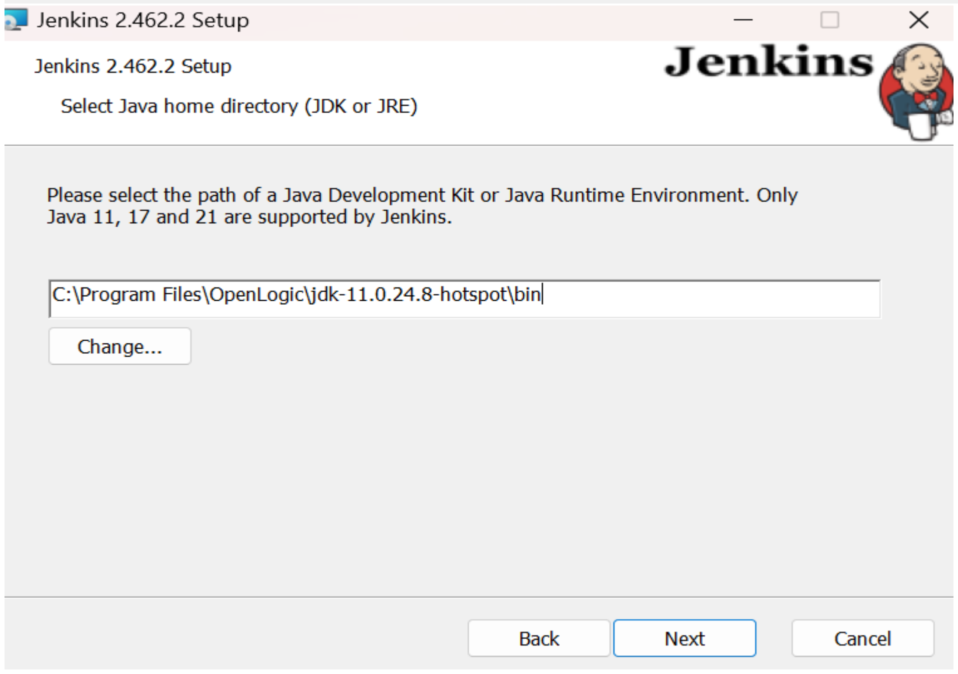
Choose a port for the service.

Please choose a port.

Port Number (1-65535):
8080

Test Port ⚠ Click 'Test Port' button to proceed

It is recommended that you accept the selected default port.

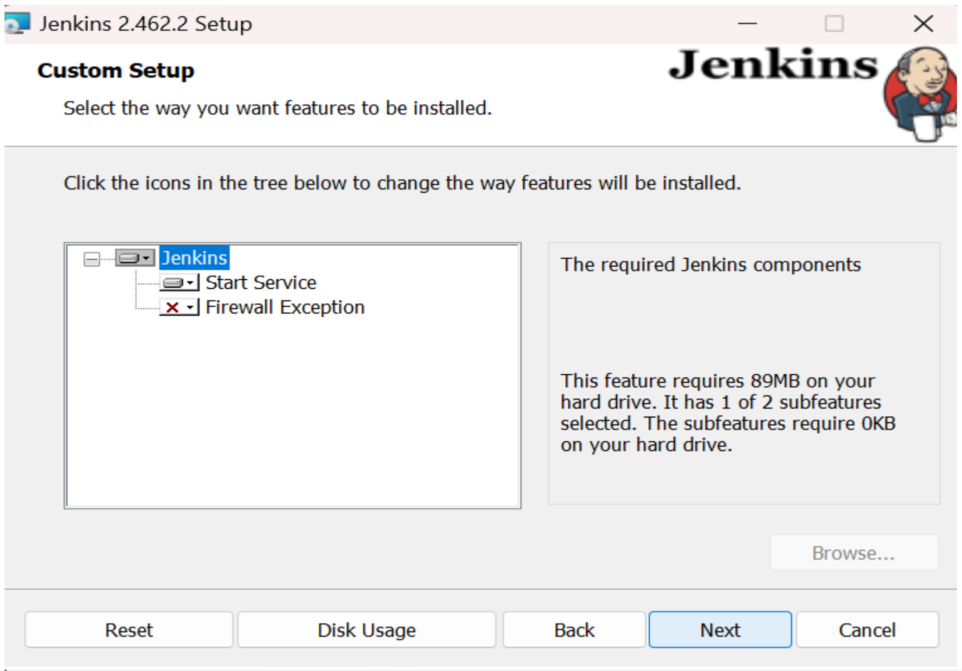

Jenkins 2.462.2 Setup

Jenkins 2.462.2 Setup

Select Java home directory (JDK or JRE)

Please select the path of a Java Development Kit or Java Runtime Environment. Only Java 11, 17 and 21 are supported by Jenkins.

C:\Program Files\OpenLogic\jdk-11.0.24.8-hotspot\bin


Jenkins 2.462.2 Setup

Custom Setup

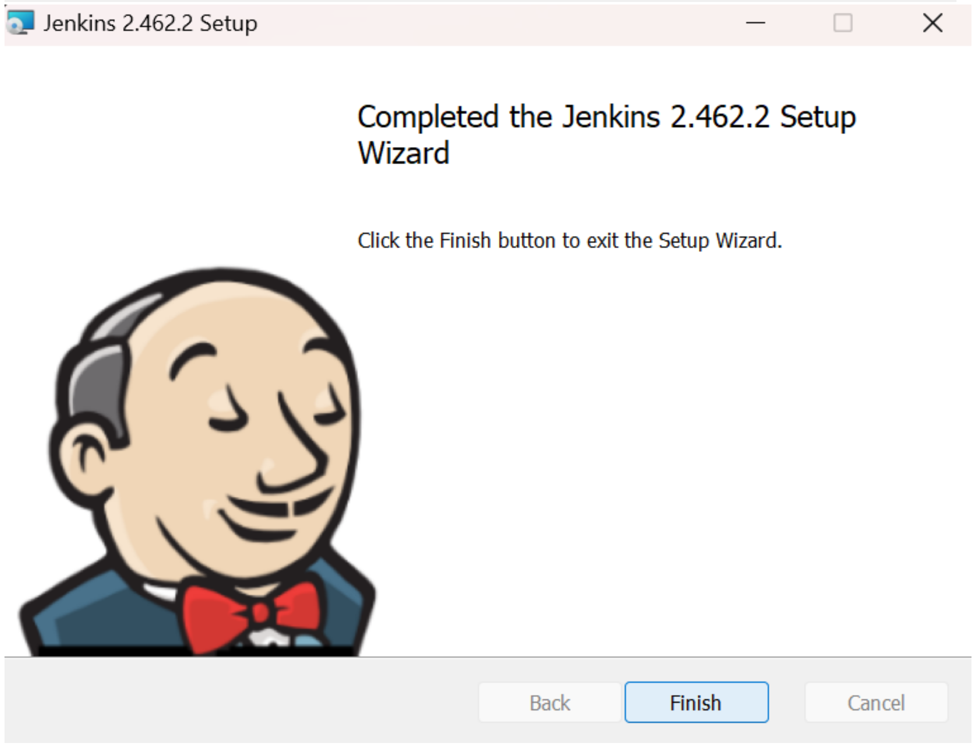
Select the way you want features to be installed.

Click the icons in the tree below to change the way features will be installed.

- Jenkins
  - Start Service
  - Firewall Exception

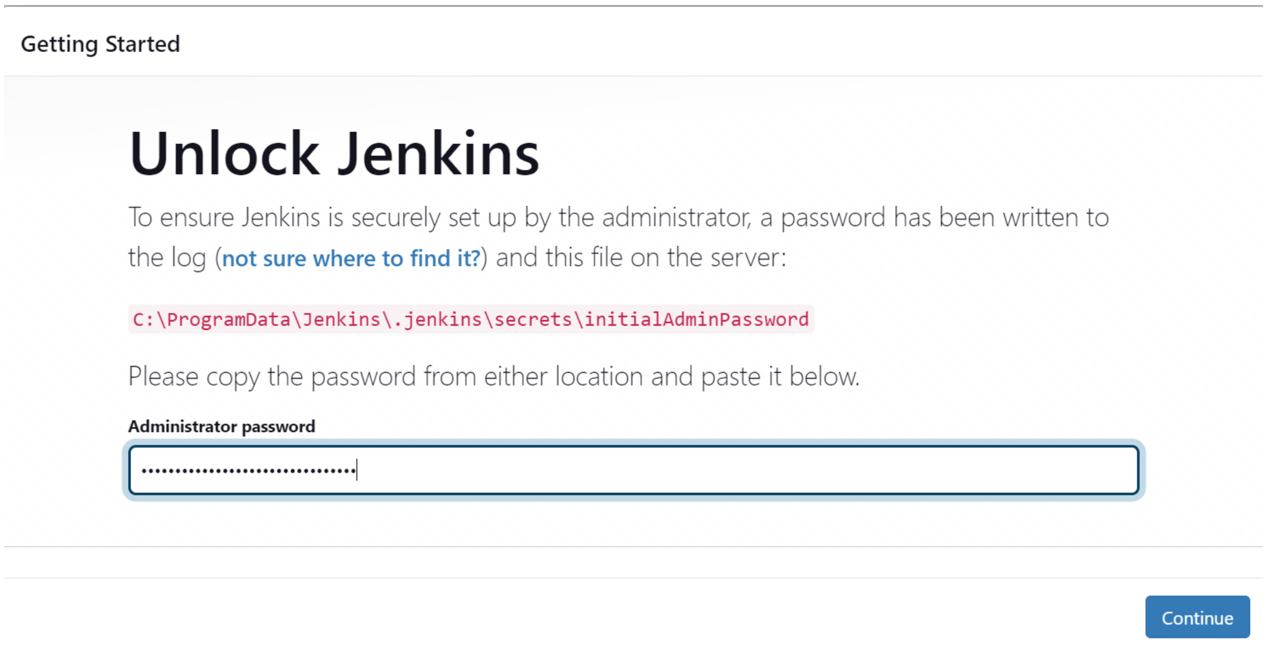
The required Jenkins components

This feature requires 89MB on your hard drive. It has 1 of 2 subfeatures selected. The subfeatures require 0KB on your hard drive.


Jenkins 2.462.2 Setup

Completed the Jenkins 2.462.2 Setup Wizard

Click the Finish button to exit the Setup Wizard.



Getting Started

Unlock Jenkins

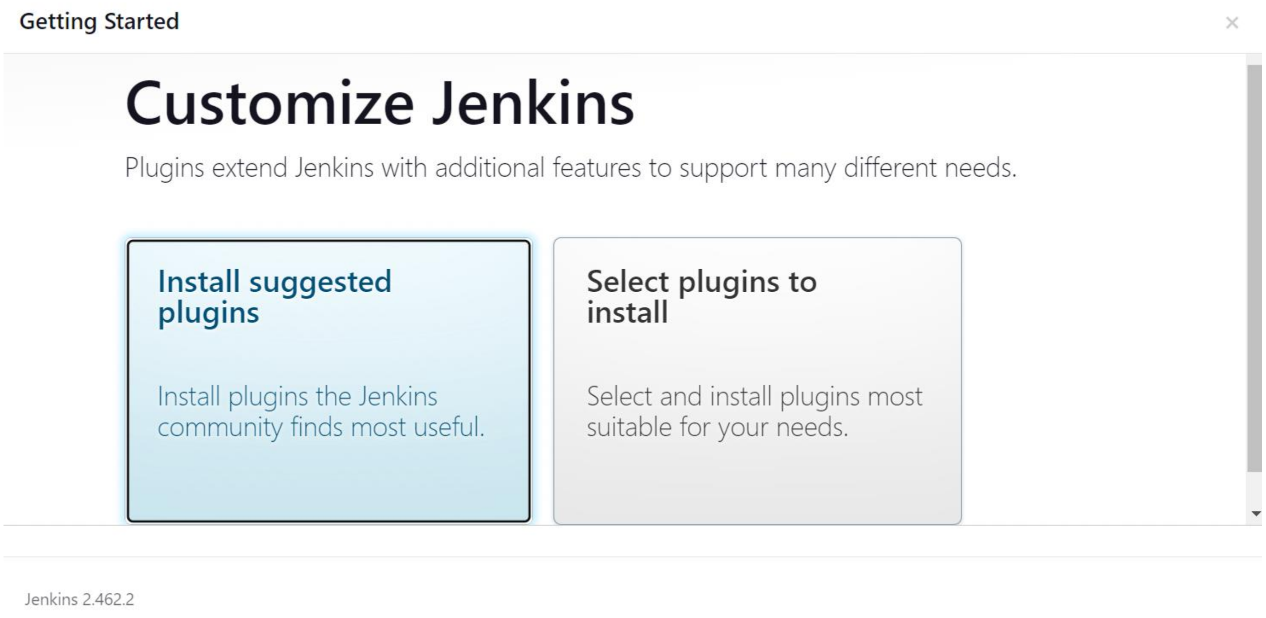
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
C:\ProgramData\Jenkins\.jenkins\secrets\initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

································



Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

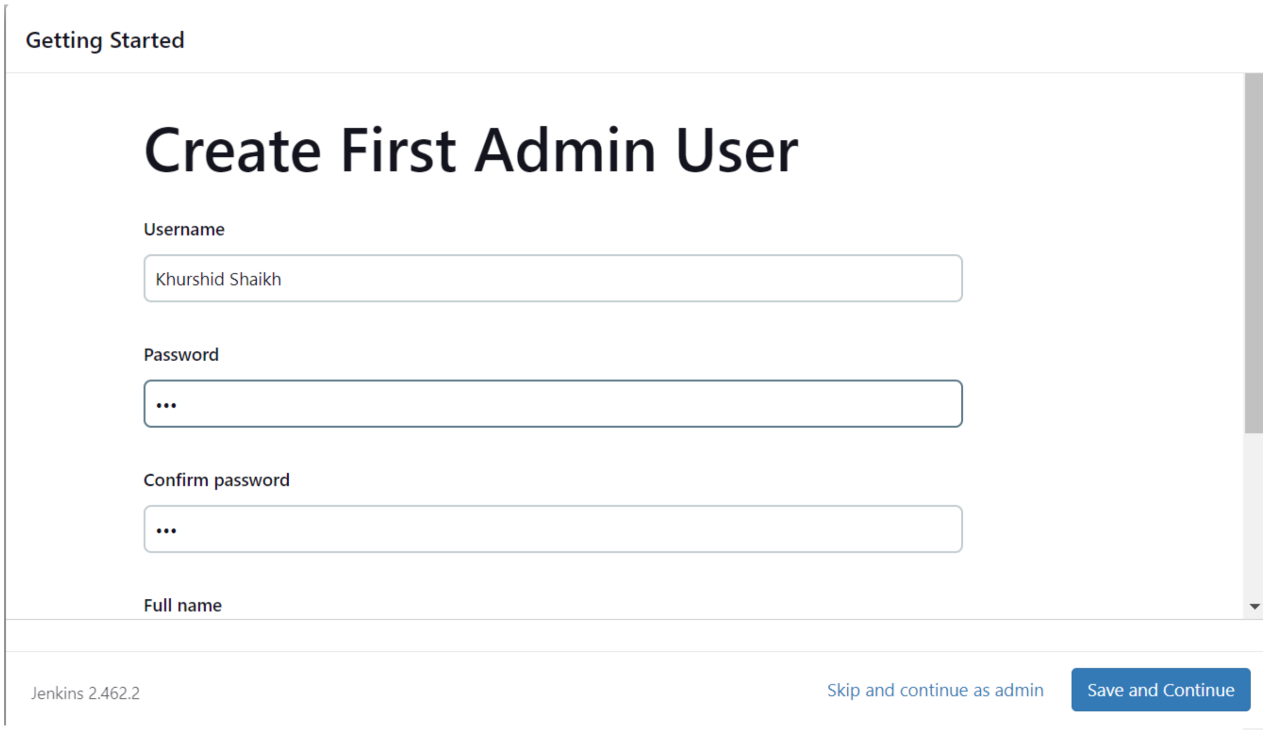
Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.462.2


Getting Started

Create First Admin User

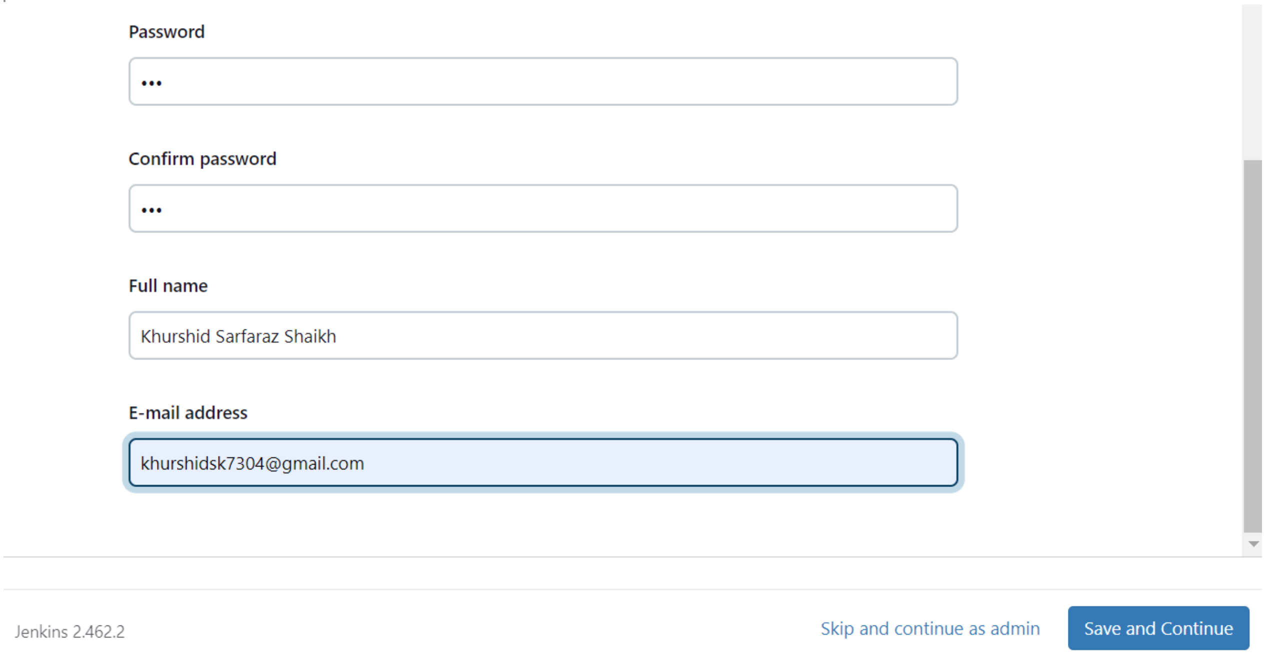
Username: Khurshid Shaikh

Password: •••

Confirm password: •••

Full name

Jenkins 2.462.2

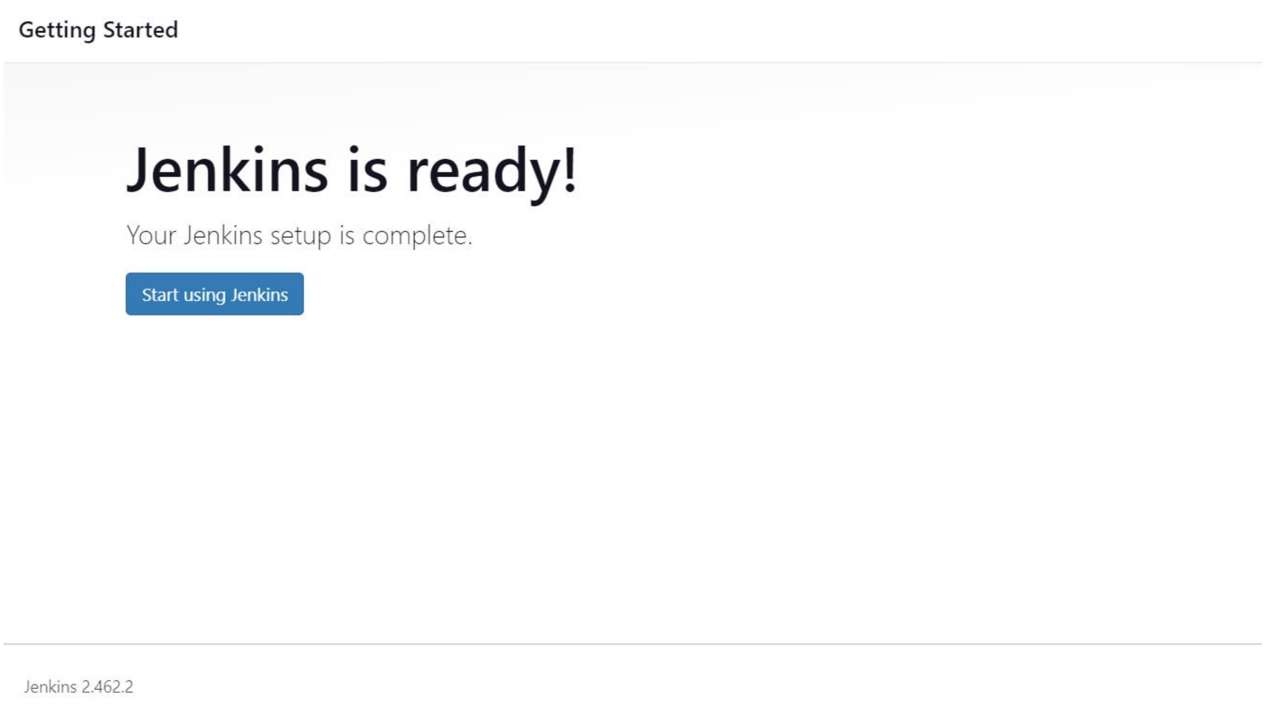

Password: •••

Confirm password: •••

Full name: Khurshid Sarfaraz Shaikh

E-mail address: khurshidsk7304@gmail.com

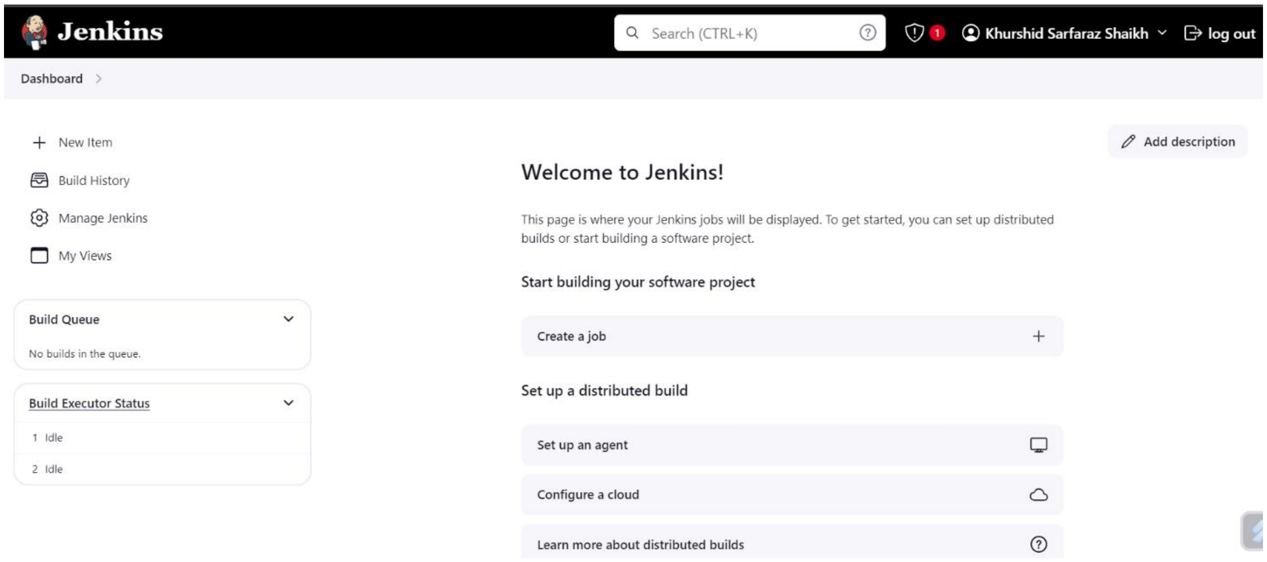
Jenkins 2.462.2


Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

Jenkins 2.462.2



Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Welcome to Jenkins!

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

Start building your software project

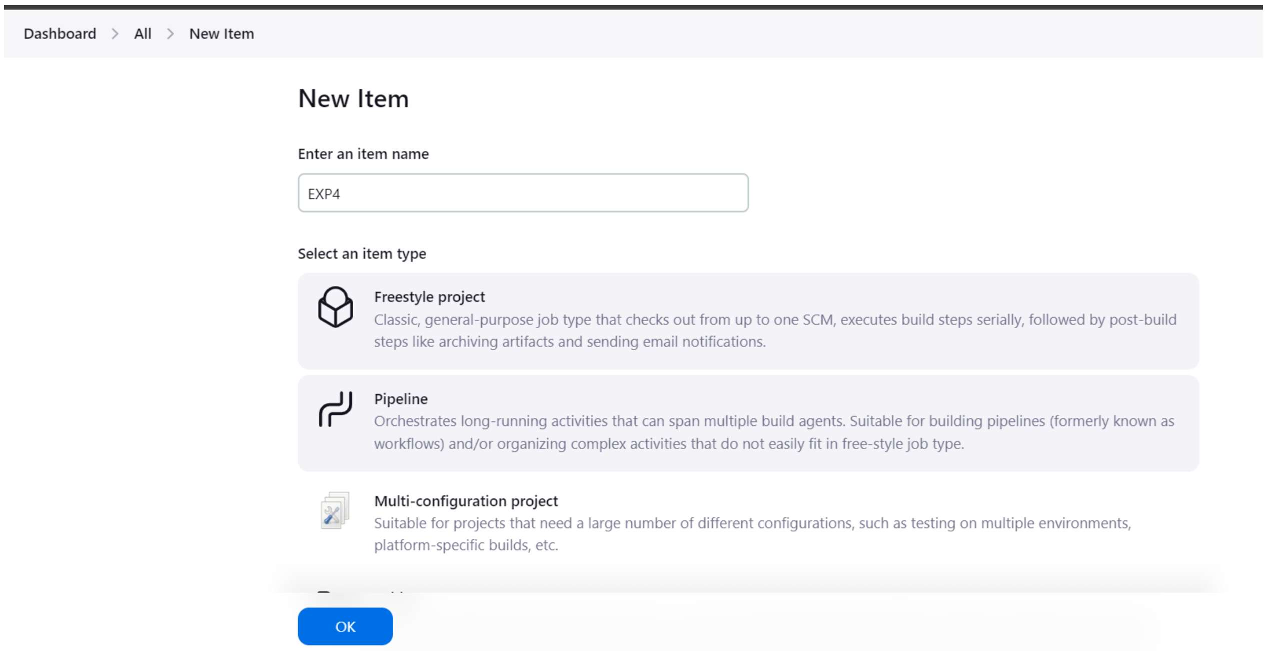
Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds



New Item

Enter an item name

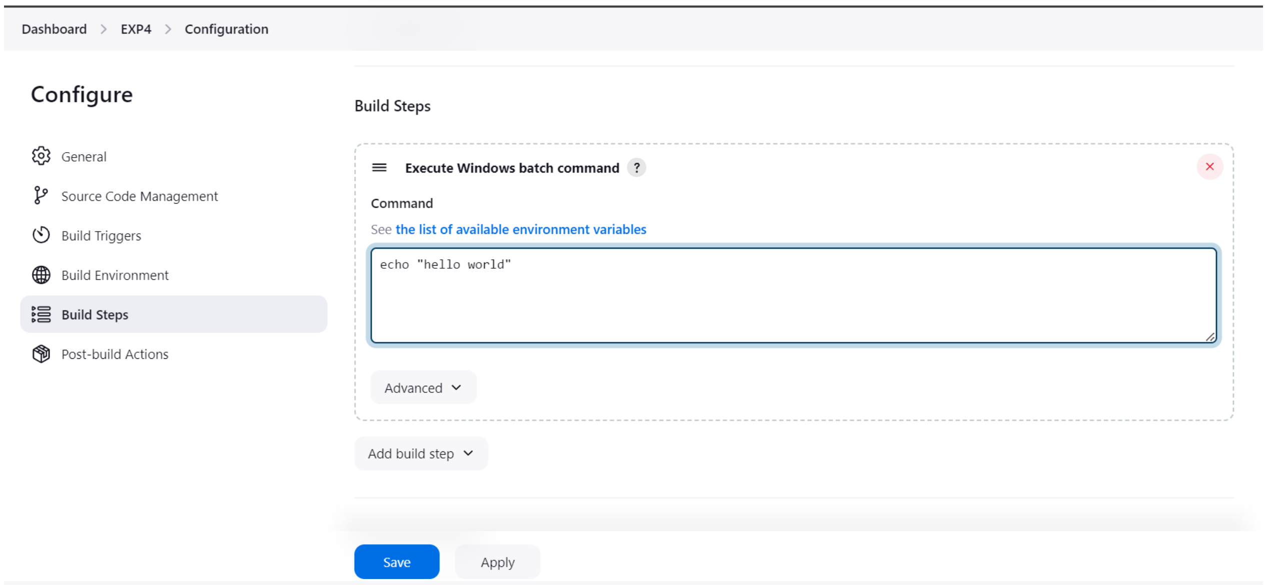
EXP4

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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.



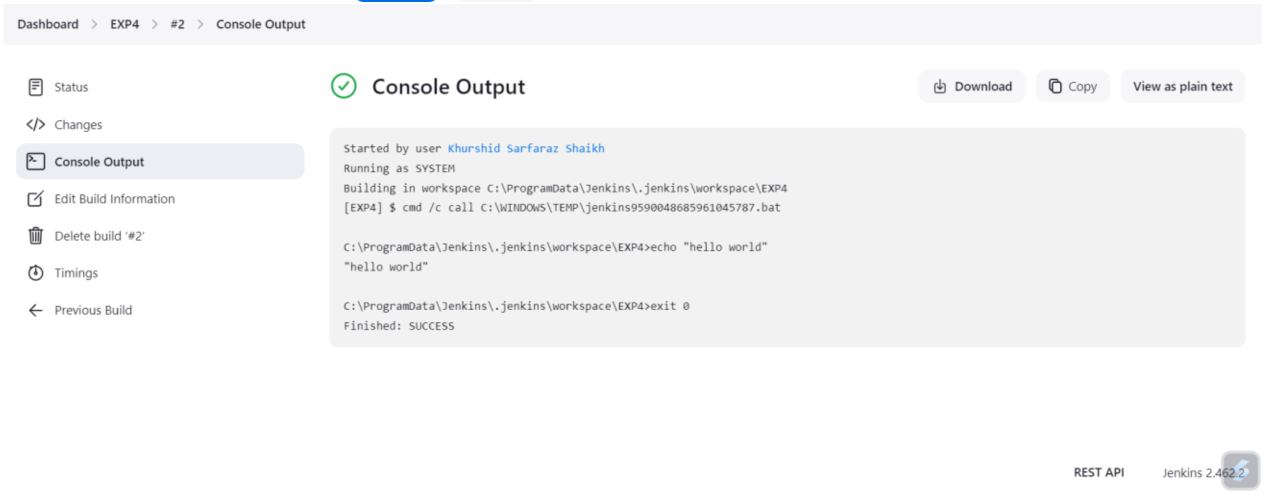
Build Steps

Execute Windows batch command ?

Command

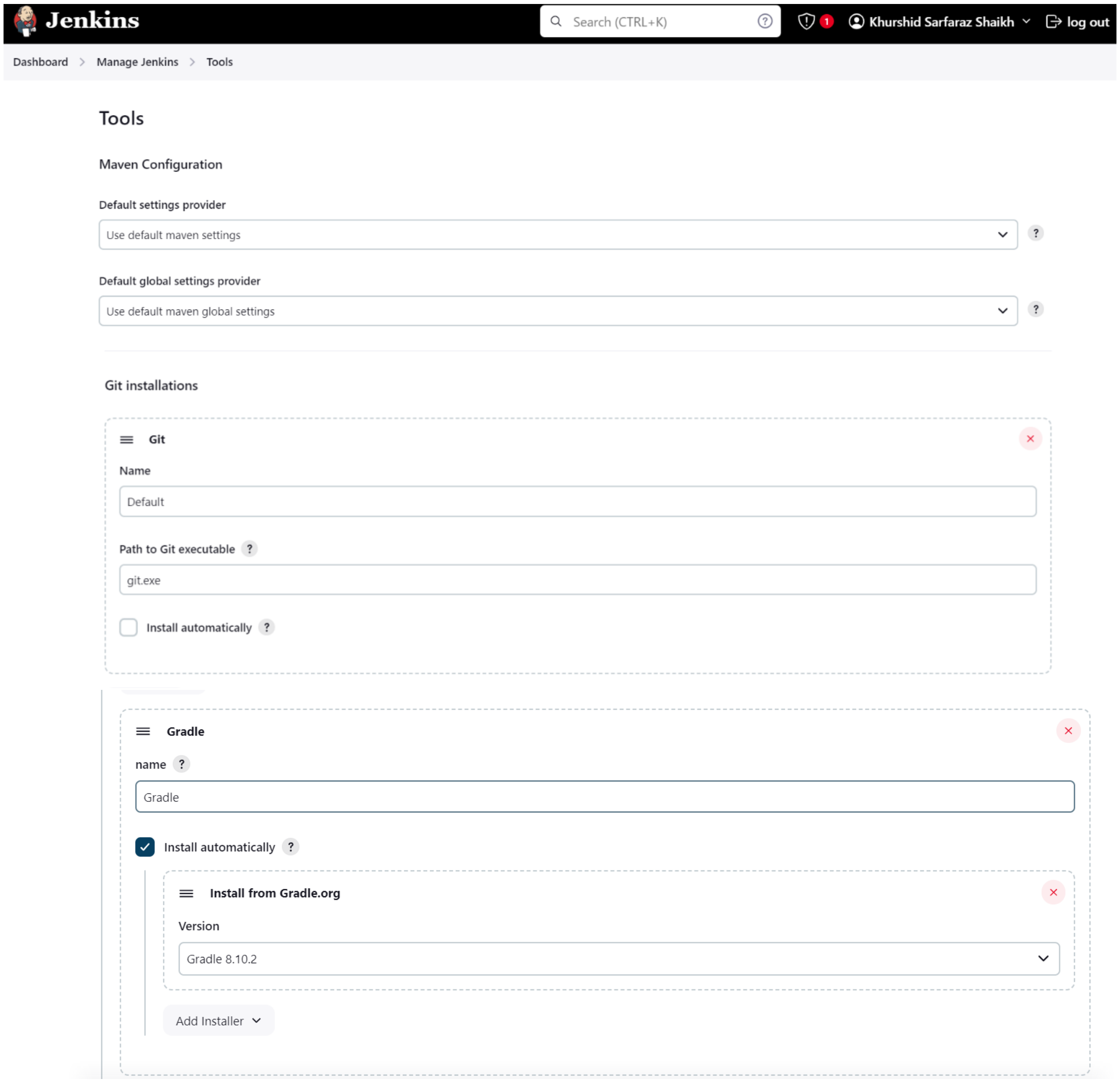
See the list of available environment variables

```
echo "hello world"
```



Console Output

```
Started by user Khurshid Sarfaraz Shaikh
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\EXP4
[EXP4] $ cmd /c call C:\WINDOWS\TEMP\jenkins9590048685961045787.bat

C:\ProgramData\Jenkins\.jenkins\workspace\EXP4>echo "hello world"
"hello world"

C:\ProgramData\Jenkins\.jenkins\workspace\EXP4>exit 0
Finished: SUCCESS
```


Jenkins | Search (CTRL+K) | 1 | Khurshid Sarfaraz Shaikh | log out

Dashboard > Manage Jenkins > Tools

Tools

Maven Configuration

Default settings provider: Use default maven settings

Default global settings provider: Use default maven global settings

Git installations

Git

Name: Default

Path to Git executable: git.exe

☐ Install automatically

Gradle

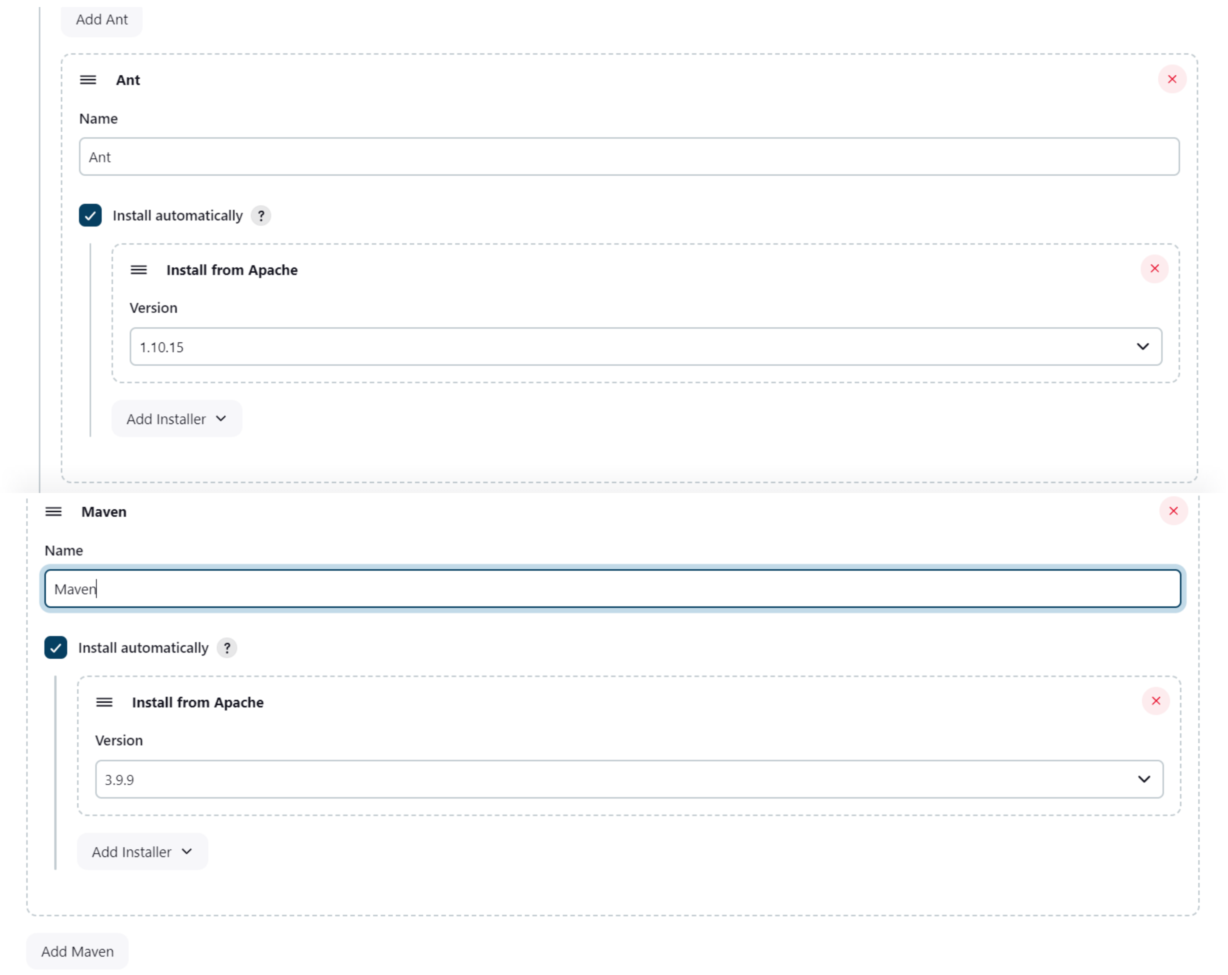
name: Gradle

☑ Install automatically

Install from Gradle.org

Version: Gradle 8.10.2

Add Installer


Add Ant

Ant

Name

Ant

✓ Install automatically ?

Install from Apache

Version

1.10.15

Add Installer

Maven

Name

Maven

✓ Install automatically ?

Install from Apache

Version

3.9.9

Add Installer

Add Maven

CONCLUSION

As a result, we understood Continuous Integration by installing and configuring Jenkins with Maven/Ant/Gradle, setting up build jobs to streamline the integration process.

EXPERIMENT : 5

Date of Performance	
Date of Submission	

AIM

To Build the pipeline of jobs using Maven / Gradle / Ant in Jenkins, create a pipeline script to Test and deploy an application over the tomcat server

PROBLEM DEFINITION

Develop a pipeline of jobs using Maven, Gradle, or Ant in Jenkins and create a script to test and deploy an application on a Tomcat server.

THEORY

Maven

Maven is a powerful project management tool based on the POM (Project Object Model). It is used for building projects, managing dependencies, and documentation.
Maven simplifies the build process similar to ANT but is more advanced. In short, Maven is a tool used for building and managing Java-based projects, making the daytoday work of Java developers easier and improving comprehension of Java-based projects.

What Maven Does:

1. Build projects easily using Maven.
2. Add JARs and other dependencies to the project with ease.
3. Provide project information (log documents, dependency list, unit test reports, etc.).
4. Help update the central repository of JARs and other dependencies.
5. Build various projects into output types like JAR, WAR, etc., without scripting.
6. Integrate projects with source control systems (such as Subversion or Git).

Pipeline

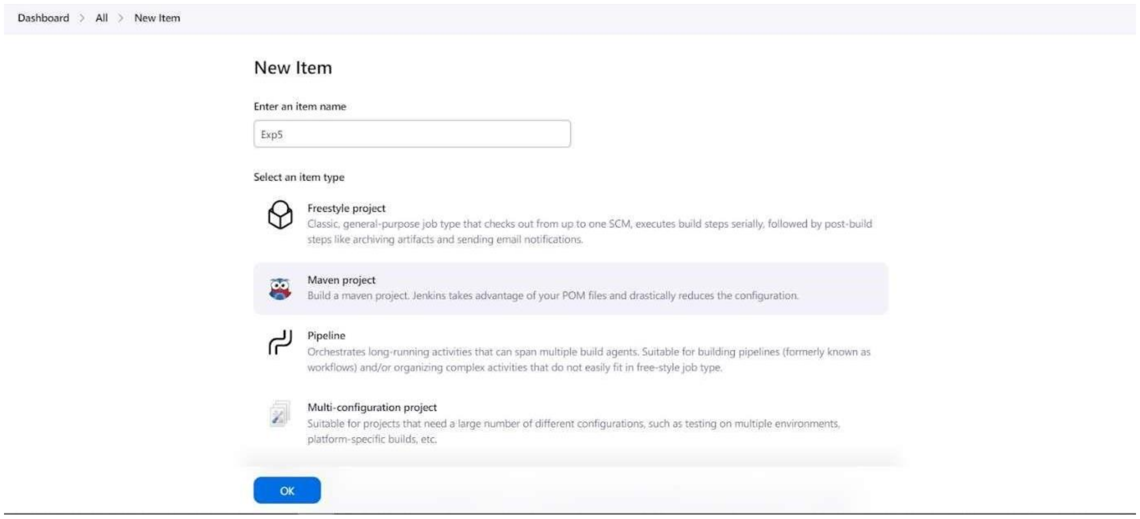
Jenkins Pipeline, or simply 'Pipeline,' is a suite of plugins that supports implementing and integrating continuous delivery pipelines into Jenkins. A continuous delivery pipeline is an automated process for delivering software from version control to users and customers. Jenkins Pipeline provides an extensible set of tools for modeling simpleto-complex delivery pipelines as code. The definition of a Jenkins pipeline is typically written into a text file called a Jenkinsfile, which is checked into a project's source control repository.

Both Declarative and Scripted Pipelines are DSLs used to describe parts of your software delivery pipeline. While standard Jenkins 'Freestyle' jobs support simple Continuous Integration by defining sequential tasks in an application lifecycle, they do not create a persistent record of execution, enable one script to address all steps in a complex workflow, or offer the advantages of a pipeline.

Pipeline Functionality:

1. Durable: Pipelines can survive both planned and unplanned restarts of Jenkins.
2. Pausable: Pipelines can optionally pause and wait for human input or approval before completing jobs.
3. Efficient: Pipelines support and can restart from various saved checkpoints.

OUTPUT


Dashboard > All > New Item

New Item

Enter an item name

Exp5

Select an item type

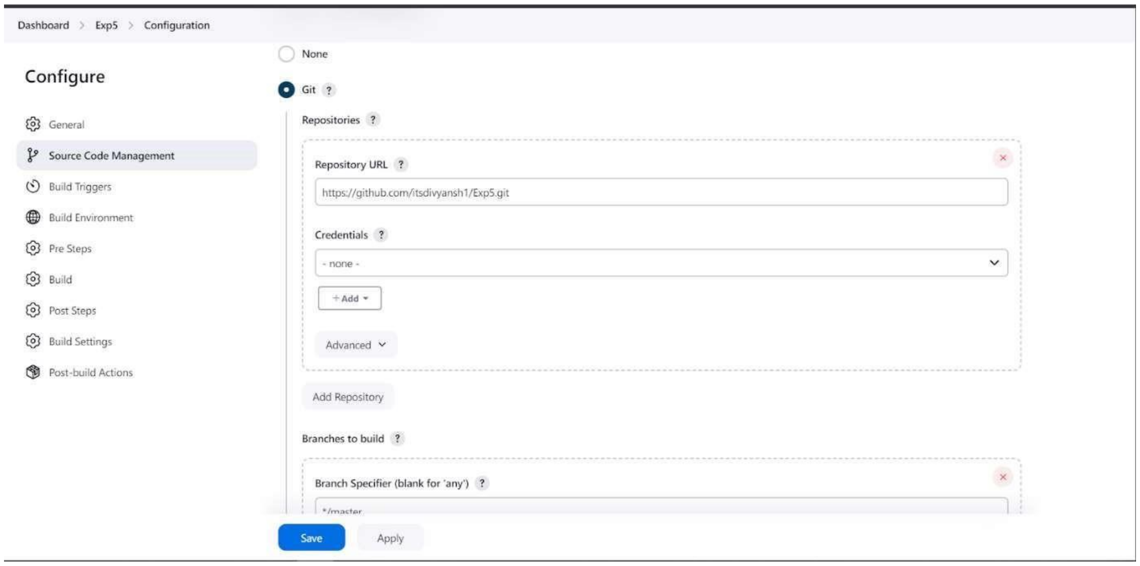
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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.

OK


Dashboard > Exp5 > Configuration

Configure

None

[✓ Git]

Repositories

Repository URL

https://github.com/itsdivyansh1/Exp5.git

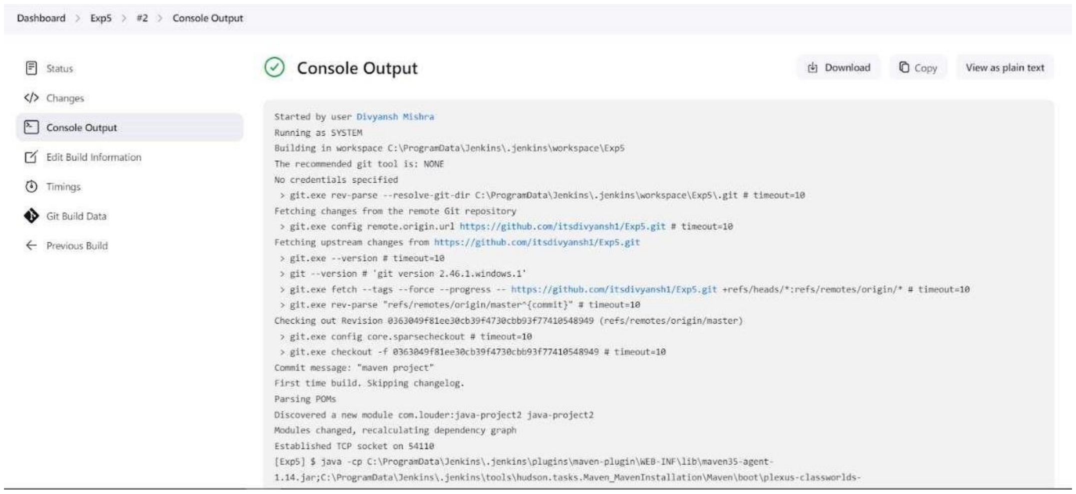
Credentials

- none -

Branches to build

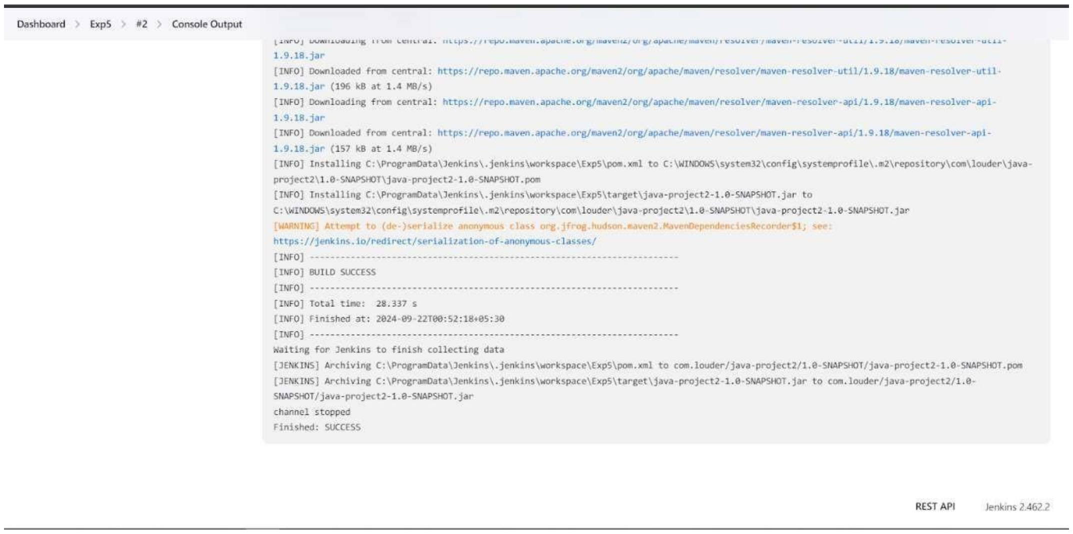
Branch Specifier (blank for 'any')

*/master

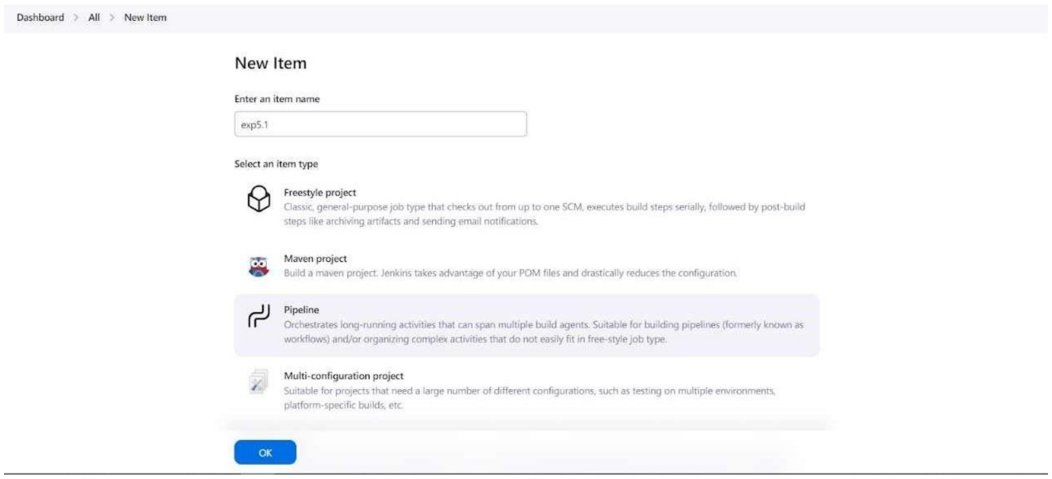


Console Output

```
Started by user Divyansh Mishra
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\Exp5
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\Exp5\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/itsdivyansh1/Exp5.git # timeout=10
Fetching upstream changes from https://github.com/itsdivyansh1/Exp5.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.46.1.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/itsdivyansh1/Exp5.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 0363049f81ee30cb39f4730cbb93f77410548949 (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 0363049f81ee30cb39f4730cbb93f77410548949 # timeout=10
Commit message: "maven project"
First time build. Skipping changelog.
Parsing POMs
Discovered a new module com.louder:java-project2 java-project2
Modules changed, recalculating dependency graph
Established TCP socket on 54110
[Exp5] $ java -cp C:\ProgramData\Jenkins\.jenkins\plugins\maven-plugin\WEB-INF\lib\maven35-agent-
1.14.jar;C:\ProgramData\Jenkins\.jenkins\tools\hudson.tasks.Maven_MavenInstallation\Maven\boot\plexus-classworlds-
```



```
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/resolver/maven-resolver-util/1.9.18/maven-resolver-util-
1.9.18.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/resolver/maven-resolver-util/1.9.18/maven-resolver-util-
1.9.18.jar (196 kB at 1.4 MB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/resolver/maven-resolver-api/1.9.18/maven-resolver-api-
1.9.18.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/resolver/maven-resolver-api/1.9.18/maven-resolver-api-
1.9.18.jar (157 kB at 1.4 MB/s)
[INFO] Installing C:\ProgramData\Jenkins\.jenkins\workspace\Exp5\pom.xml to C:\WINDOWS\system32\config\systemprofile\.m2\repository\com\louder\java-
project2\1.0-SNAPSHOT\java-project2-1.0-SNAPSHOT.pom
[INFO] Installing C:\ProgramData\Jenkins\.jenkins\workspace\Exp5\target\java-project2-1.0-SNAPSHOT.jar to
C:\WINDOWS\system32\config\systemprofile\.m2\repository\com\louder\java-project2\1.0-SNAPSHOT\java-project2-1.0-SNAPSHOT.jar
[WARNING] Attempt to (de-)serialize anonymous class org.jfrog.hudson.maven2.MavenDependenciesRecorder$1; see:
https://jenkins.io/redirect/serialization-of-anonymous-classes/
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  28.337 s
[INFO] Finished at: 2024-09-22T00:52:18+05:30
[INFO] ------------------------------------------------------------------------
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving C:\ProgramData\Jenkins\.jenkins\workspace\Exp5\pom.xml to com.louder/java-project2/1.0-SNAPSHOT/java-project2-1.0-SNAPSHOT.pom
[JENKINS] Archiving C:\ProgramData\Jenkins\.jenkins\workspace\Exp5\target\java-project2-1.0-SNAPSHOT.jar to com.louder/java-project2/1.0-
SNAPSHOT/java-project2-1.0-SNAPSHOT.jar
channel stopped
Finished: SUCCESS
```



New Item

Enter an item name

exp5.1

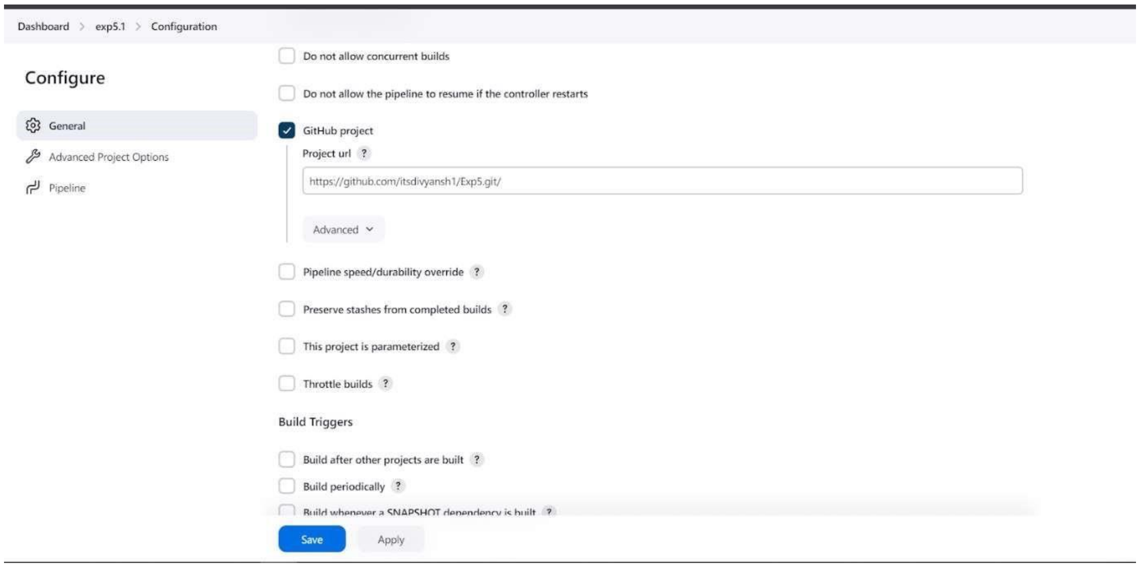
Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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.



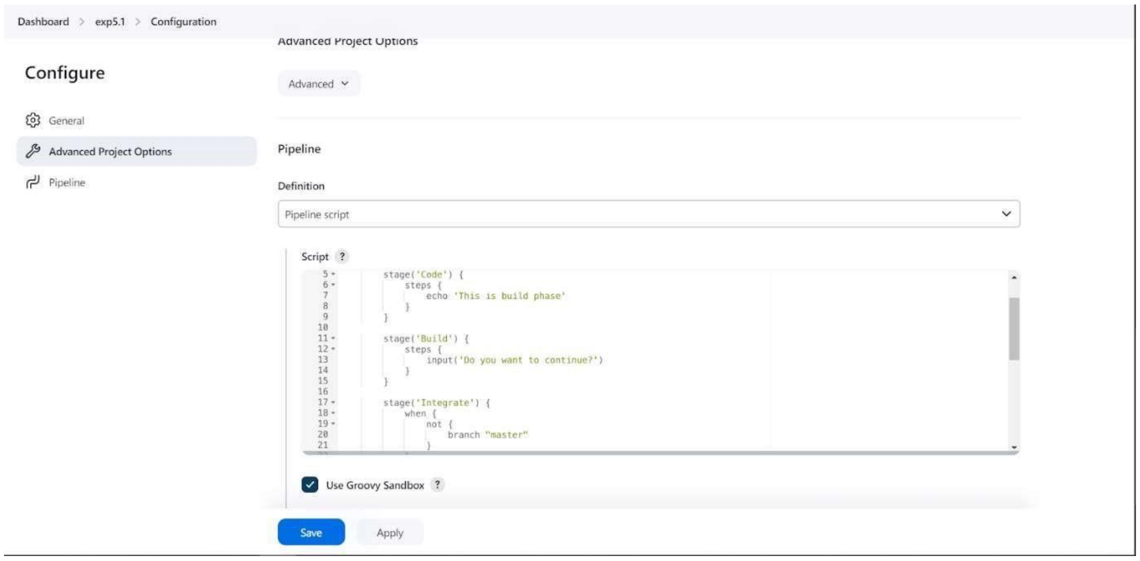
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [x] GitHub project

Project url: https://github.com/itsdivyansh1/Exp5.git/

- [ ] Pipeline speed/durability override
- [ ] Preserve stashes from completed builds
- [ ] This project is parameterized
- [ ] Throttle builds

Build Triggers

- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] Build whenever a SNAPSHOT dependency is built



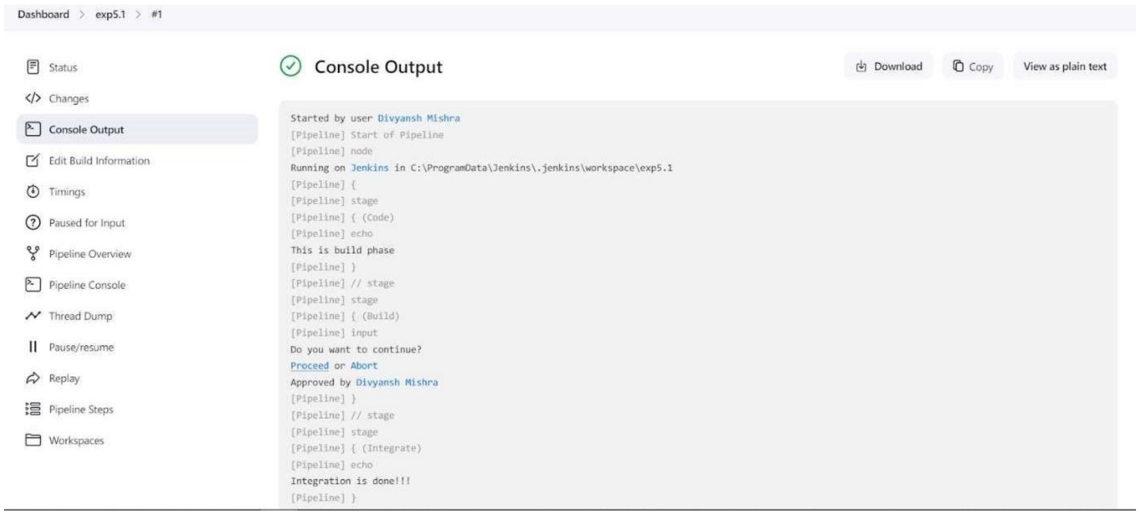
Pipeline

Definition: Pipeline script

Script

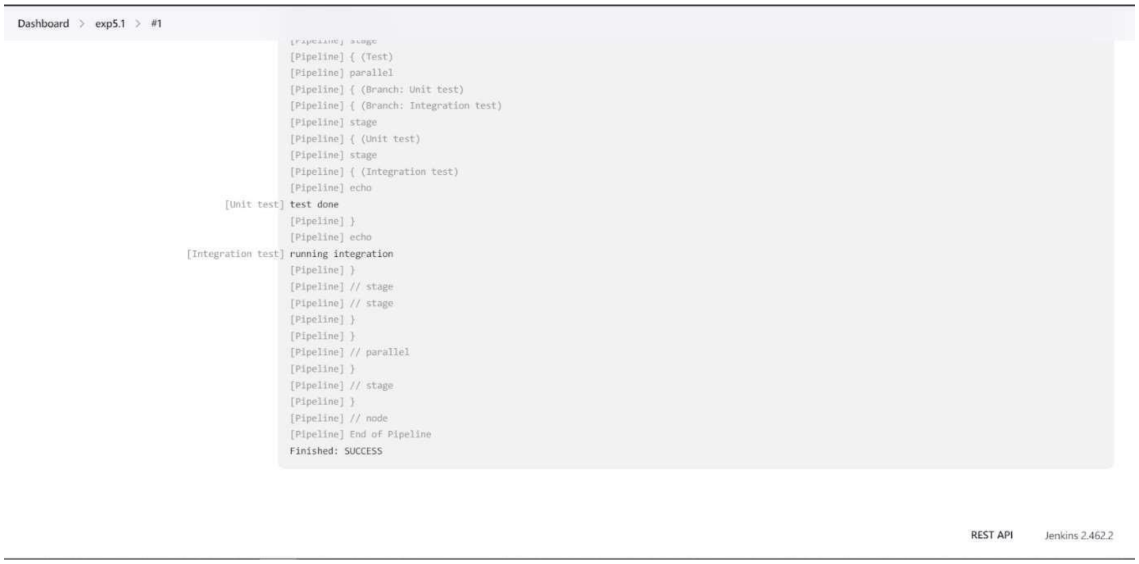
```
        stage('Code') {
            steps {
                echo 'This is build phase'
            }
        }

        stage('Build') {
            steps {
                input('Do you want to continue?')
            }
        }

        stage('Integrate') {
            when {
                not {
                    branch "master"
                }
```

- [x] Use Groovy Sandbox



Console Output

```
Started by user Divyansh Mishra
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\exp5.1
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Code)
[Pipeline] echo
This is build phase
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] input
Do you want to continue?
Proceed or Abort
Approved by Divyansh Mishra
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Integrate)
[Pipeline] echo
Integration is done!!!
[Pipeline] }
```



```
                    [Pipeline] stage
                    [Pipeline] { (Test)
                    [Pipeline] parallel
                    [Pipeline] { (Branch: Unit test)
                    [Pipeline] { (Branch: Integration test)
                    [Pipeline] stage
                    [Pipeline] { (Unit test)
                    [Pipeline] stage
                    [Pipeline] { (Integration test)
                    [Pipeline] echo
         [Unit test] test done
                    [Pipeline] }
                    [Pipeline] echo
  [Integration test] running integration
                    [Pipeline] }
                    [Pipeline] // stage
                    [Pipeline] // stage
                    [Pipeline] }
                    [Pipeline] }
                    [Pipeline] // parallel
                    [Pipeline] }
                    [Pipeline] // stage
                    [Pipeline] }
                    [Pipeline] // node
                    [Pipeline] End of Pipeline
                    Finished: SUCCESS
```

CONCLUSION

Consequently, we built pipelines of jobs using Maven/Gradle/Ant in Jenkins and created pipeline scripts for testing and deploying applications over Tomcat, enhancing automated deployment processes.