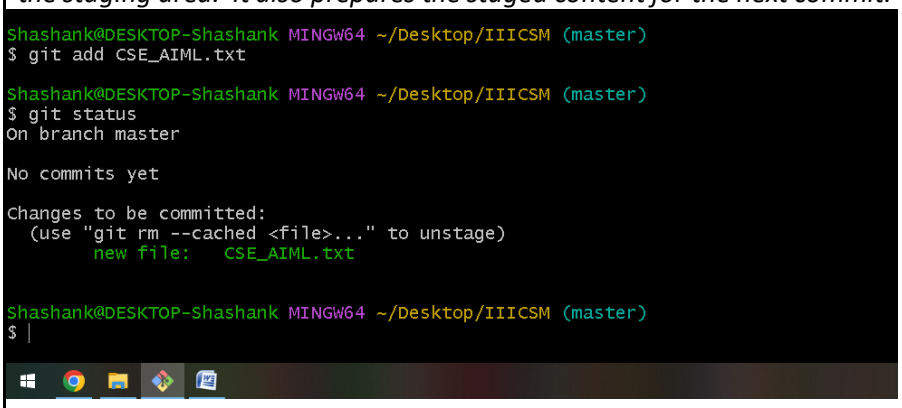
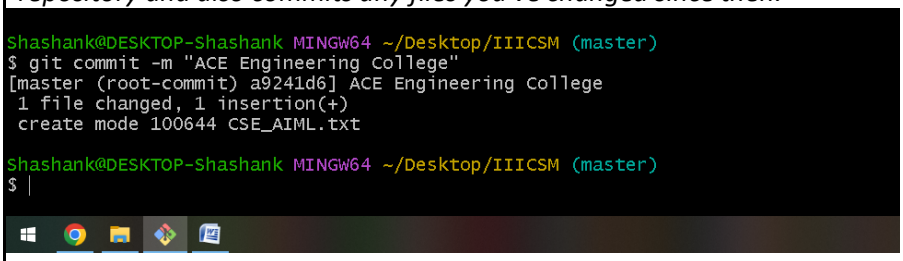
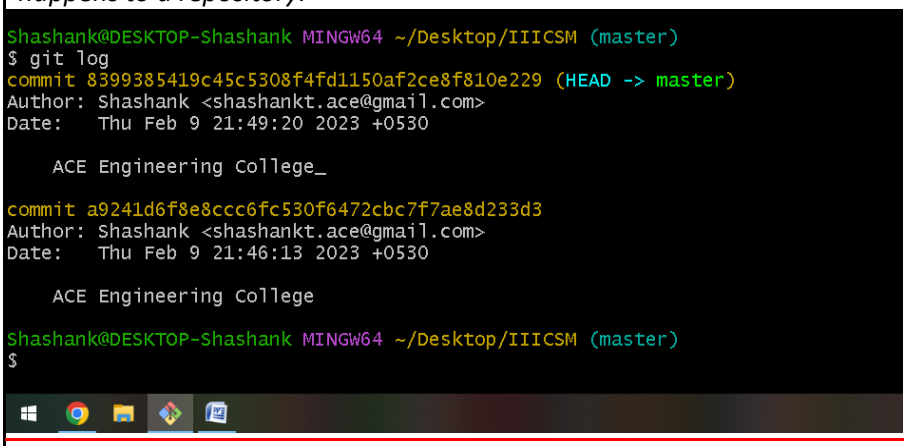
Lab 02 :

Explore Git and Github commands.

Git Environment Setup-
Git Config command
	Git supports a command called git config that lets you get and set configuration variables that control all facets of how Git looks and operates. It is used to set Git configuration values on a global or local project level. Setting user.name and user.email are the necessary configuration options as your name and email will show up in your commit messages.
	`$ git config --global user.name csmbcdevops`
	`aceec@rocky MINGW64 ~`
	`$ git config --global user.email stiwari.ace@gmail.com`
	`aceec@rocky MINGW64 ~`
	`$ git config --list`
	`diff.astextplain.textconv=astextplain`
	`filter.lfs.clean=git-lfs clean -- %f`
	`filter.lfs.smudge=git-lfs smudge -- %f`
	`filter.lfs.process=git-lfs filter-process`
	`filter.lfs.required=true`
	`http.sslbackend=openssl`
	`http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/cert`
	`core.autocrlf=true`
	`core.fscache=true`
	`core.symlinks=false`
	`pull.rebase=false`
	`credential.helper=manager`
	`credential.https://dev.azure.com.usehttppath=true`
	`init.defaultbranch=master`
	`user.name=csmbcdevops`
	`user.email=stiwari.ace@gmail.com`

Git init command	*This command is used to create a local repository.*
	The git init command is the first command that you will run on Git.
	The git init command is used to create a new blank repository.
	`Shashank@DESKTOP-Shashank MINGW64 ~`
	`$ pwd`
	`/c/Users/Shashank`
	`Shashank@DESKTOP-Shashank MINGW64 ~`
	`$ cd Desktop/IIICSM`
	`Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM`
	`$ git init`
	`Initialized empty Git repository in C:/Users/Shashank/Desktop/IIICSM/.git/`
	`Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)`
	`$`

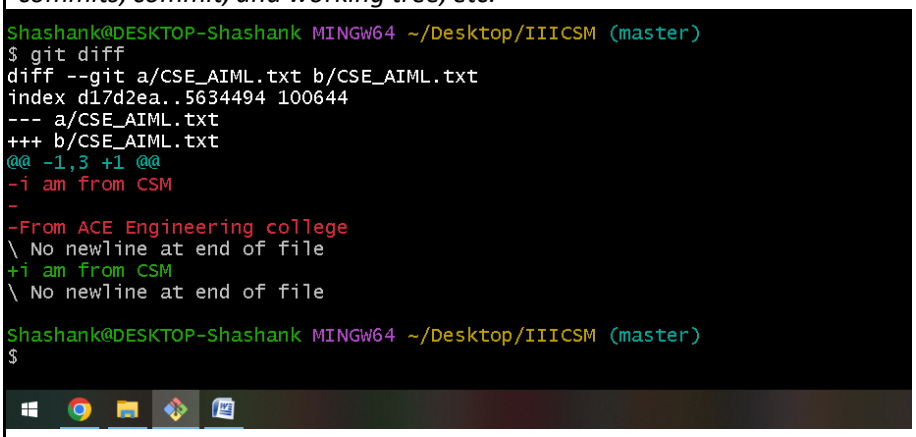
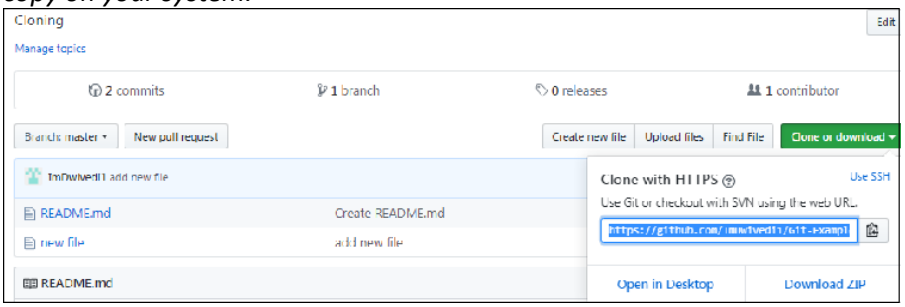
Git status command	*The status command is used to display the state of the working directory and the staging area.*
	`Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)`
	`$ git status`
	`On branch master`
	`No commits yet`
	`nothing to commit (create/copy files and use "git add" to track)`

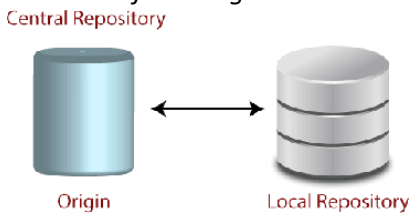
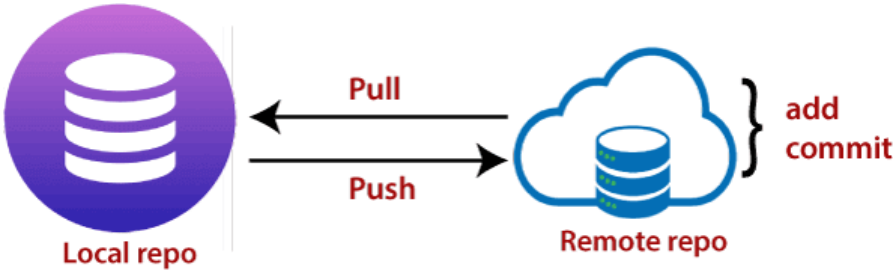
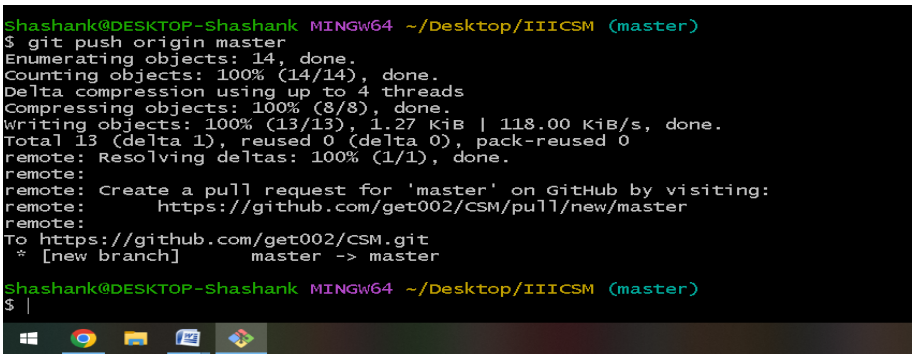
Git add command	*This command is used to add one or more files to staging (Index) area. The git add command is used to add file contents to the Index (Staging Area). This command updates the current content of the working tree to the staging area. It also prepares the staged content for the next commit.* 
Git commit command	*It is used to record the changes in the repository. It is the next command after the git add. This command commits any files added with git add in the repository and also commits any files you've changed since then.* 
Git log command	*This command is used to check the commit history. Git log is a utility tool to review and read a history of everything that happens to a repository.* 

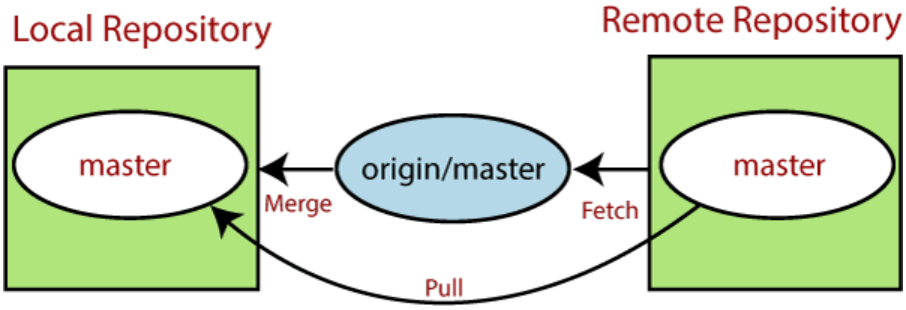
```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ git add CSE_AIML.txt

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   CSE_AIML.txt


Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ |
```

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ git commit -m "ACE Engineering College"
[master (root-commit) a9241d6] ACE Engineering College
 1 file changed, 1 insertion(+)
 create mode 100644 CSE_AIML.txt

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ |
```

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$ git log
commit 8399385419c45c5308f4fd1150af2ce8f810e229 (HEAD -> master)
Author: Shashank <shashankt.ace@gmail.com>
Date:   Thu Feb 9 21:49:20 2023 +0530

    ACE Engineering College_

commit a9241d6f8e8ccc6fc530f6472cbc7f7ae8d233d3
Author: Shashank <shashankt.ace@gmail.com>
Date:   Thu Feb 9 21:46:13 2023 +0530

    ACE Engineering College

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)
$
```

Git Diff command	*It compares the different versions of data sources.* *Git diff is a command-line utility. It's a multi use Git command. When it is executed, it runs a diff function on Git data sources. These data sources can be files, branches, commits, and more. It is used to show changes between commits, commit, and working tree, etc.* 
Git Clone	*In Git, cloning is the act of making a copy of any target repository.* *The target repository can be remote or local.* *You can clone your repository from the remote repository to create a local copy on your system.*  $ git clone https://github.com/ImDwivedi1/Git-Example.git

Git Origin Master	*In Git, The term origin is referred to the remote repository where you want to publish your commits.* *The default remote repository is called origin, although you can work with several remotes having a different name at the same time.* *The origin is a short name for the remote repository that a project was initially being cloned. It is used in place of the original repository URL. Thus, it makes referencing much easier.*  
Git Push Command	*The push term refers to upload local repository content to a remote repository. Pushing is an act of transfer commits from your local repository to a remote repository. Pushing is capable of overwriting changes; caution should be taken when pushing.*  

Git Pull Command	*The term pull is used to receive data from GitHub. It fetches and merges changes from the remote server to your working directory. The git pull command is used to pull a repository.*
	 
	
	`Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)`
	`$ git pull origin main`
	`From https://github.com/get002/CSM`
	`* branch main -> FETCH_HEAD`
	`fatal: refusing to merge unrelated histories`
	
	`Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)`
	`$ git pull origin main --allow-unrelated-histories`
	`From https://github.com/get002/CSM`
	`* branch main -> FETCH_HEAD`
	`Merge made by the 'ort' strategy.`
	`README.md | 1 +`
	`1 file changed, 1 insertion(+)`
	`create mode 100644 README.md`
	
	`Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/IIICSM (master)`
	`$`

Git Fetch command	*Git "fetch" Downloads commits, objects and refs from another repository. It fetches branches and tags from one or more repositories. It holds repositories along with the objects that are necessary to complete their histories to keep updated remote-tracking branches.*
	`HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/Git-Example (master)`
	`$ git fetch https://github.com/ImDwivedi1/Git-Example.git`
	`warning: no common commits`
	`remote: Enumerating objects: 6, done.`
	`remote: Counting objects: 100% (6/6), done.`
	`remote: Compressing objects: 100% (4/4), done.`
	`remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0`
	`Unpacking objects: 100% (6/6), done.`
	`From https://github.com/ImDwivedi1/Git-Example`
	`* branch HEAD -> FETCH_HEAD`

Operations on Branches - Create Branch	*You can create a new branch with the help of the git branch command. This command will be used as:*
	$ git branch <**branch** name>
	HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
	$ git branch B1

List Branch	*You can List all of the available branches in your repository by using the following command.*
	Either we can use git branch - list or git branch command to list the available branches in the repository.
	$ git branch --list
	HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
	$ git branch
	B1
	branch3
	* master
	HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
	$ git branch --list
	B1
	branch3
	* master

Operations on Branches - Delete Branch	*You can delete the specified branch. It is a safe operation. In this command, Git prevents you from deleting the branch if it has unmerged changes.*
	$ git branch -d<**branch** name>
	HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
	$ git branch -d B1
	Deleted branch B1 (was 554a122).

Operations on Branches - Delete a Remote Branch	*You can delete a remote branch from Git desktop application.*
	$ git push origin -delete <**branch** name>
	HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
	$ git push origin --delete branch2
	To https://github.com/ImDwivedi1/GitExample2
	- [deleted] branch2
	HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
	$

Operations on Branches - Switch Branch	*Git allows you to switch between the branches without making a commit. You can switch between two branches with the git checkout command.*
	$ git branch -m <old branch name> <new branch name>
	HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
	$ git branch -m branch4 renamedB1
	
	HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
	$ git branch
	* master
	renamedB1
	
	HiMaNshU@HiMaNshU-PC MINGW64 ~/Desktop/GitExample2 (master)
	$ |

Lab 03 :

Practice source code management on GitHub. Experiment with the source code written in exercise 1.

Check the status of local repository using git status command,

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

nothing added to commit but untracked files present (use "git add" to track)
```

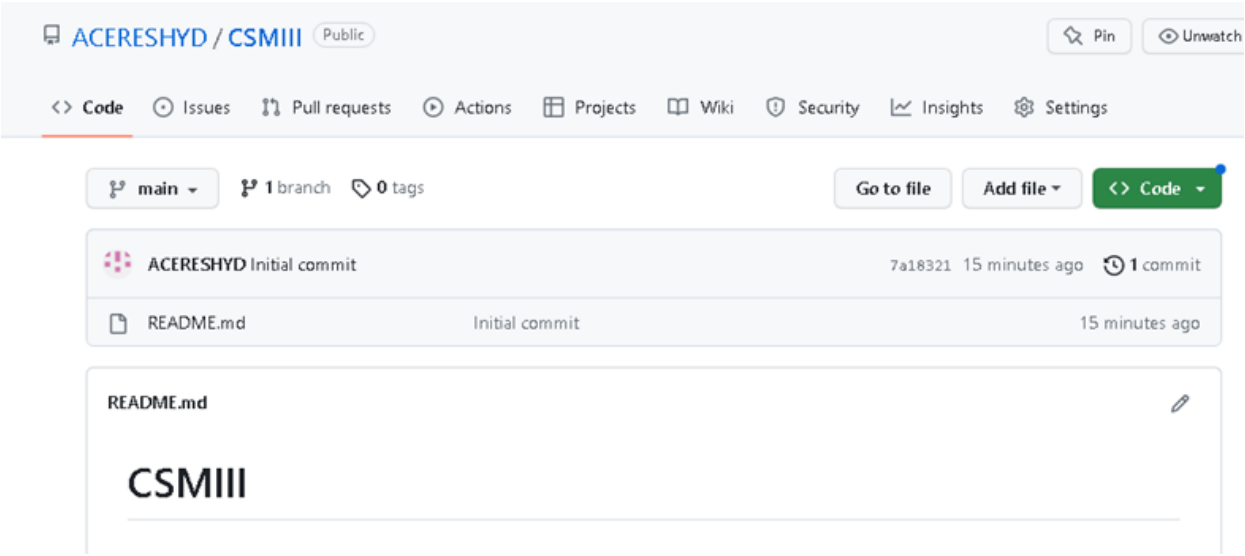
There is a file index.html at working area, Use git add command to add on staging area. Then commit the changes on local repository using git commit command.

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git add index.html

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git commit -m "index file"
[master (root-commit) dab7375] index file
 1 file changed, 92 insertions(+)
 create mode 100644 index.html
```

```
shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git log
commit dab7375858be19f5a9c3b7e35a230bdbe344a9b6 (HEAD -> master)
Author: Shashank <shashankt.ace@gmail.com>
Date:   Fri Mar 31 18:39:32 2023 +0530

    index file

shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git status
On branch master
nothing to commit, working tree clean
```

All changes updated on local repository.

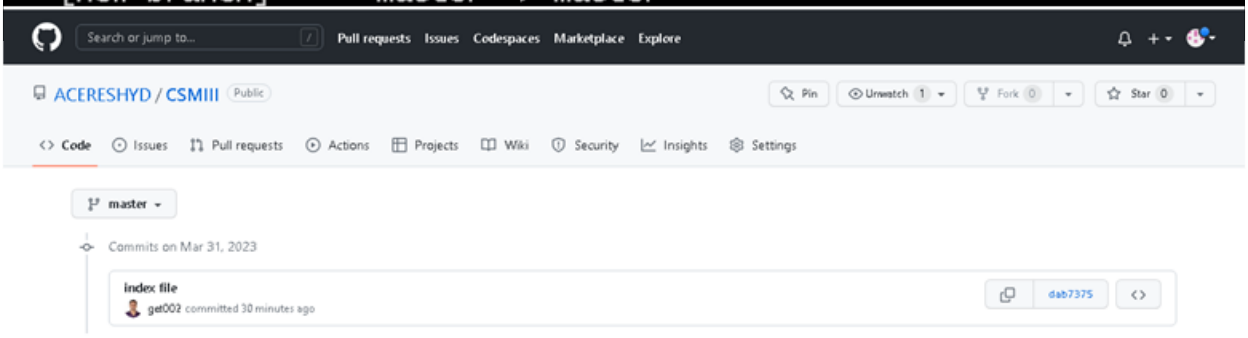
Now to push these changes on remote repository, first create a remote repository.




```
shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git remote add origin git@github.com:ACERESHYD/CSMIII.git
```

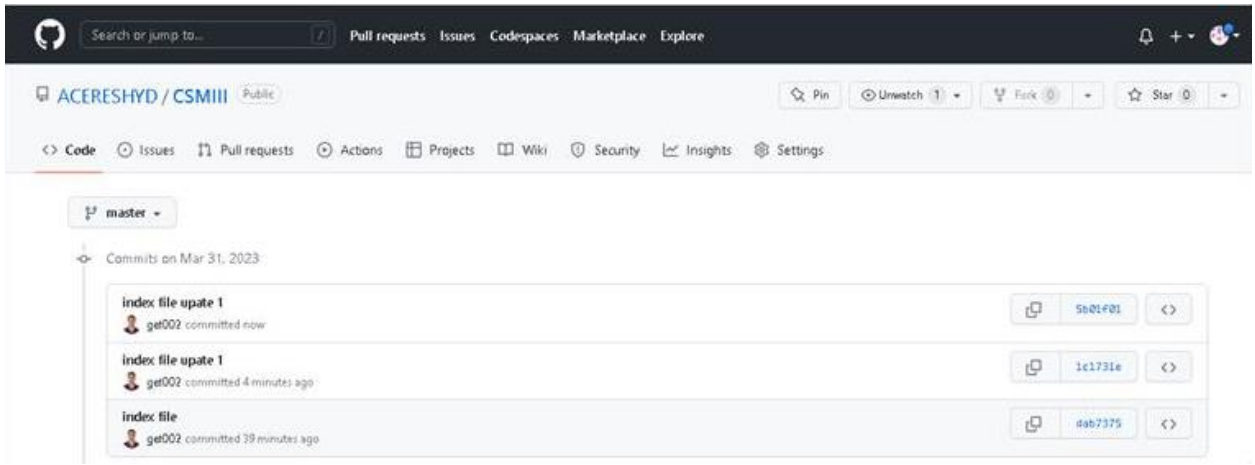
Now push the changes from local repository to remote repository using git push origin master

```
shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 1018 bytes | 169.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/ACERESHYD/CSMIII/pull/new/master
remote:
To github.com:ACERESHYD/CSMIII.git
 * [new branch]      master -> master
```



<u>Task : Makes atleast 6 commits on your remote repository from git bash and also from github to practise source code management.</u>

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git add index.html

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git commit -m "index file upate 1"
[master 1c1731e] index file upate 1
 1 file changed, 1 insertion(+), 1 deletion(-)

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 94.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:ACERESHYD/CSMIII.git
   dab7375..1c1731e  master -> master

Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git log
commit 1c1731eef8150d21459a893f660aa0e603d9db7b (HEAD -> master, origin/master)
Author: Shashank <shashankt.ace@gmail.com>
Date:   Fri Mar 31 19:14:24 2023 +0530

    index file upate 1

commit dab7375858be19f5a9c3b7e35a230bdbe344a9b6
Author: Shashank <shashankt.ace@gmail.com>
Date:   Fri Mar 31 18:39:32 2023 +0530

    index file
```



Here **HEAD** always represent current commit content. Now comparing latest commit with the previous commit using git diff command

```
Shashank@DESKTOP-Shashank MINGW64 ~/Desktop/WebProject (master)
$ git diff 5b01f017 1c1731ee
diff --git a/index.html b/index.html
index 377aeea..f76e7b0 100644
--- a/index.html
+++ b/index.html
@@ -6,7 +6,6 @@
 <h2 align="center"> An Autonomous institute</h2>
 <hr/>
 <title>Student Registration Form</title>
-<hr/>
 <style type="text/css">
 th {
```

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

index file upate 1

master

Browse files

get002 committed 12 minutes ago

1 parent 1c1731e commit 5b01f01

Showing 1 changed file with 1 addition and 0 deletions.

Split | Unified

1 index.html

```
@@ -6,6 +6,7 @@ <h1 align="center">ACE Engineering College </h1>
 <h2 align="center"> An Autonomous institute</h2>
 <hr/>
 <title>Student Registration Form</title>
+<hr/>
 <style type="text/css">
 th {
         color: #FFF;
```

0 comments on commit 5b01f01

Lock conversation

Lab 04:
Jenkins installation and setup, explore the environment.

- Jenkins is an open source automation tool written in Java programming language.
- Jenkins is free and allows continuous integration.
- What is continuous integration?
    - Continuous integration refers to the build and unit testing stages of the software release process. Every revision that is committed triggers an automated build and test. With continuous delivery, code changes are automatically built, tested, and prepared for a release to production.
- Jenkins is a server based application and requires a web server like Apache Tomcat.
- Jenkins builds and tests our software projects continuously.
    - This is the main reason for Jenkins to became so popular, continously monitoring of repeated tasks which arise during the development of a project.
    - Example, if your team is developing a project, Jenkins will continuously test your project builds and show you the errors in early stages of your development.
- Benefits of using Jenkins CI
    - **Reduced Development Cycle** – Since every commit is built and tested, it allows releasing new features to the user faster and with fewer errors.
    - **Shorter Time to Integrate Code** – Before the use of Jenkins CI, integration of code was done manually, thus taking a few days. In some cases, it might happen that the code is not running, and it is hard to debug as it might have gone through various commits in the repository. Integrating code after every commit ensures that the functionality is not broken after a commit.
    - **Faster Feedback Loops** – Developers get feedback and improve the code whenever a test breaks during a commit. Otherwise, debugging the issue can be very difficult, given teams would not be sure which commit resulted in the bug.
    - **Automated Workflow** – Teams should not worry about running a manual test for each commit. The Jenkins CI pipeline checks the latest code and builds the code along with the tests. The test can deploy the project in a specific environment if it is green. It can notify the developer by breaking the build.

Jenkins installation

Install Java Version 8

- Since Jenkins is a Java based application, therefore Java is a must.
- Download java8 from the below link:
  https://www.oracle.com/java/technologies/downloads/#java8
- Then install the Java as follows:

https://www.oracle.com/java/technologies/downloads/#jdk17-windows

Most Visited | Other Bookmarks

ORACLE | Products | Industries | Resources | Support | Events | Developer

JDK 17 will receive updates under these terms, until at least September 2024.

Java SE Development Kit 17 downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications and components using the Java programming language.

The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.

Documentation Download

Linux | macOS | Windows

Product/file description	File size	Download
x64 Compressed Archive	170.64 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256)
x64 Installer	151.99 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256)
x64 MSI Installer	150.88 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256)

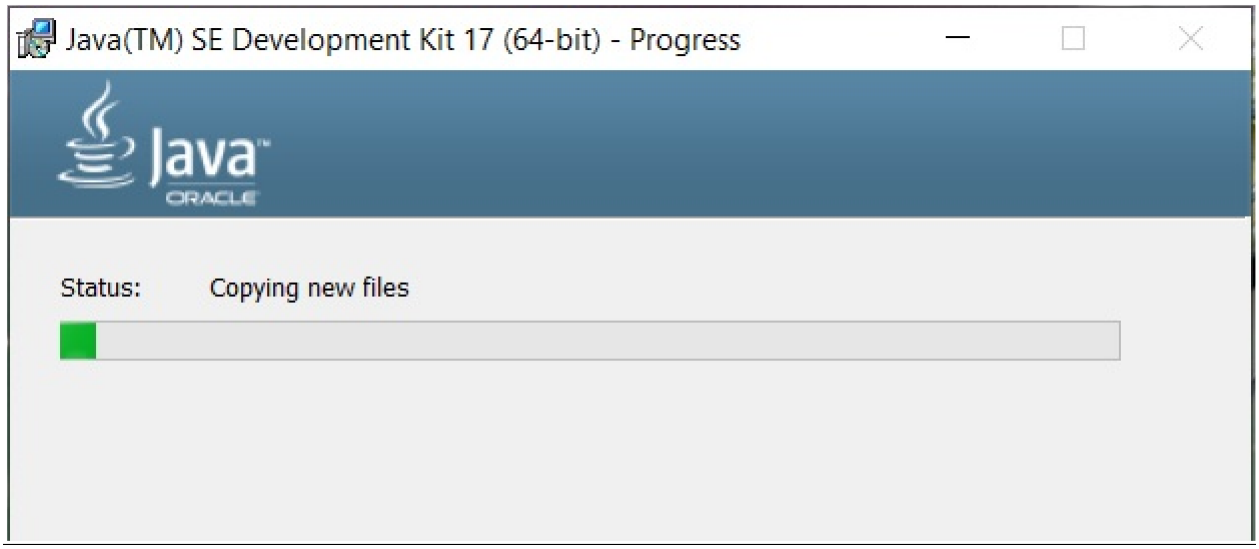
Java(TM) SE Development Kit 17 (64-bit) - Setup

Welcome to the Installation Wizard for Java SE Development Kit 17

This wizard will guide you through the installation process for the Java SE Development Kit 17.

Next > | Cancel


Java(TM) SE Development Kit 17 (64-bit) - Destination Folder
Java
ORACLE
Java(TM) SE Development Kit 17 (64-bit), including a private JRE and src.zip. This will require 420MB on your hard drive. Click the "Change" button to change the installation folder.
Install Java(TM) SE Development Kit 17 (64-bit) to:
C:\java\oracle\jdk-17\
Change...
Back
Next
Cancel

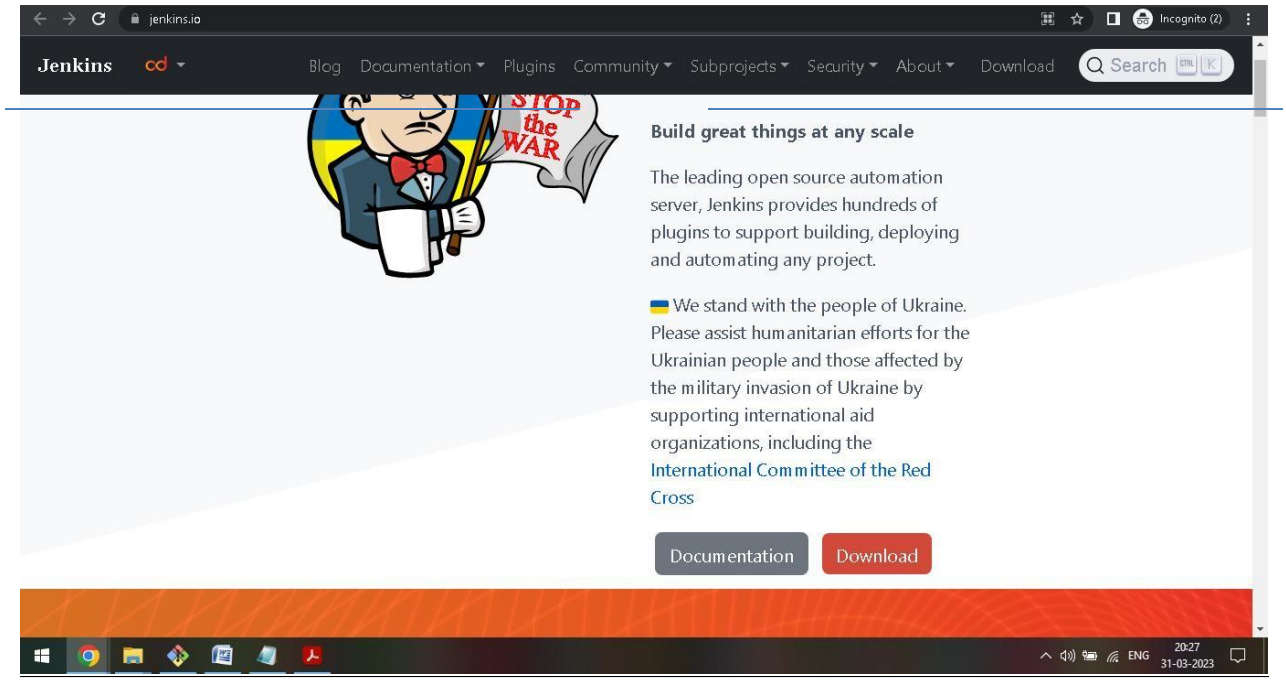

Java(TM) SE Development Kit 17 (64-bit) - Progress
Java
ORACLE
Status:
Copying new files

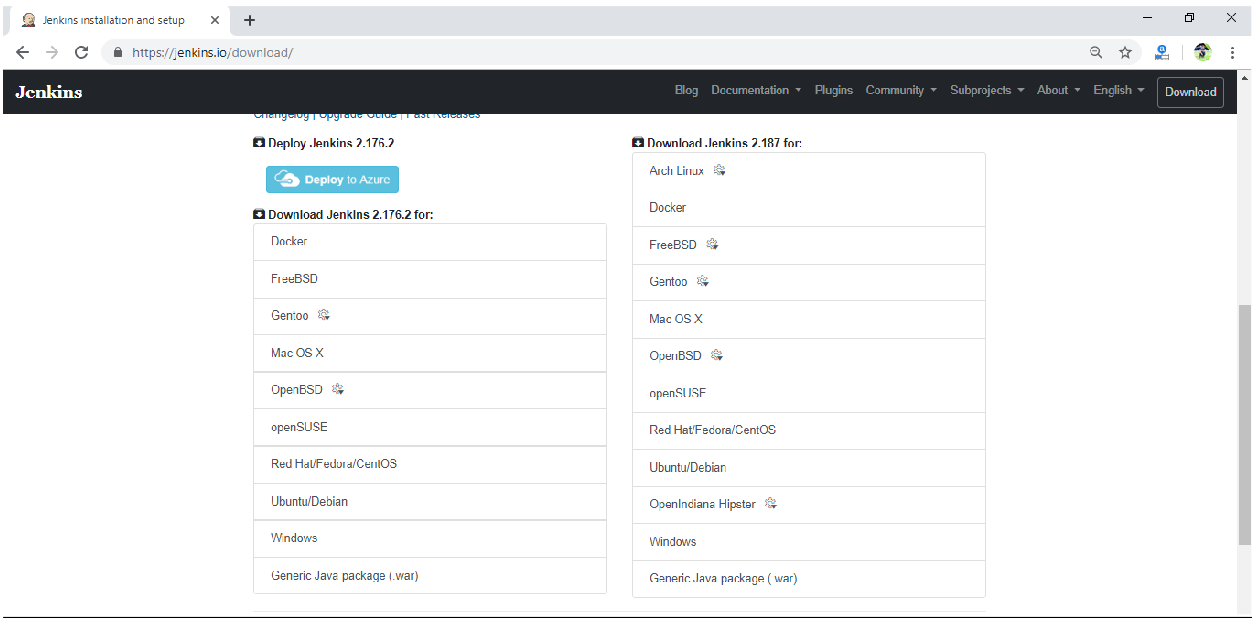

Java(TM) SE Development Kit 17 (64-bit) - Complete
Java
ORACLE
Java(TM) SE Development Kit 17 (64-bit) Successfully Installed
Click Next Steps to access tutorials, API documentation, developer guides, release notes and more to help you get started with the JDK.
Next Steps
Close


Command Prompt
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. All rights reserved.
C:\Users\bhagw>java -version
java version "17" 2021-09-14 LTS
Java(TM) SE Runtime Environment (build 17+35-LTS-2724)
Java HotSpot(TM) 64-Bit Server VM (build 17+35-LTS-2724, mixed mode, sharing)
C:\Users\bhagw>

Download Jenkins war File

Download from the following link
https://www.jenkins.io/



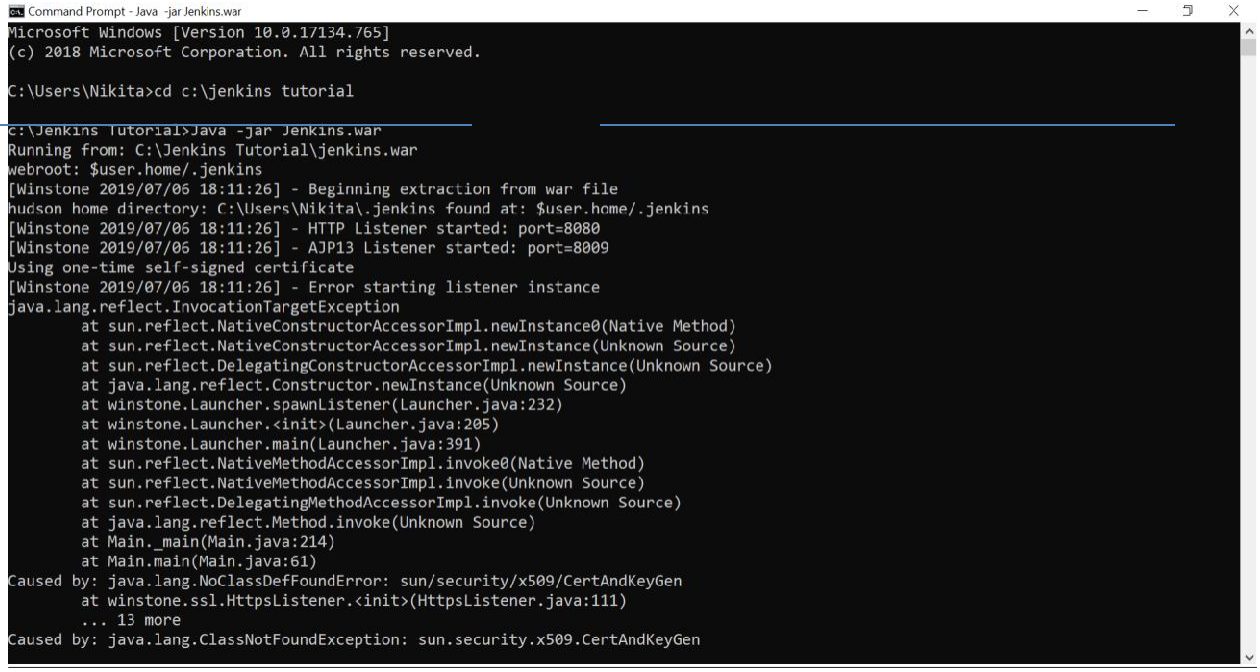


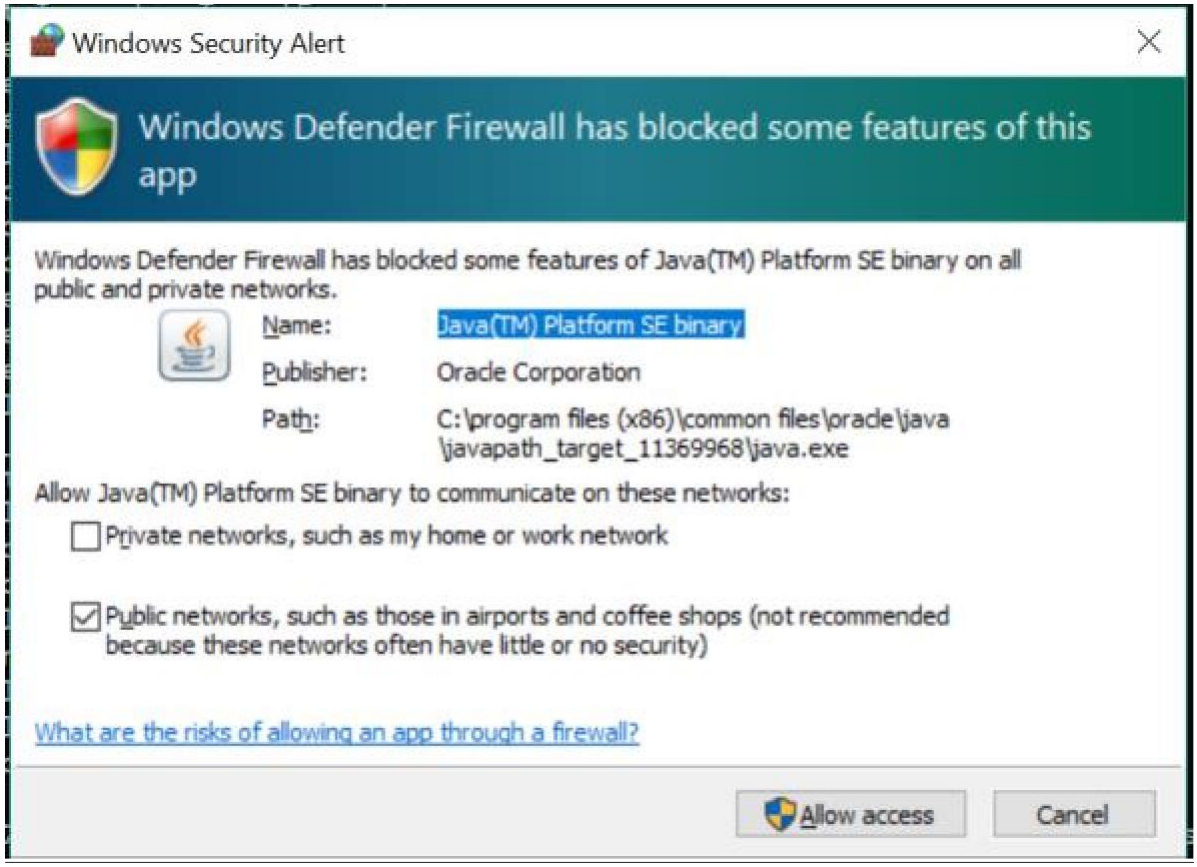
- Click on Generic Java Package (.war) to download the Jenkins war file.

- Open the command prompt and go to the directory where the Jenkins.war file is located. And then run the following command:

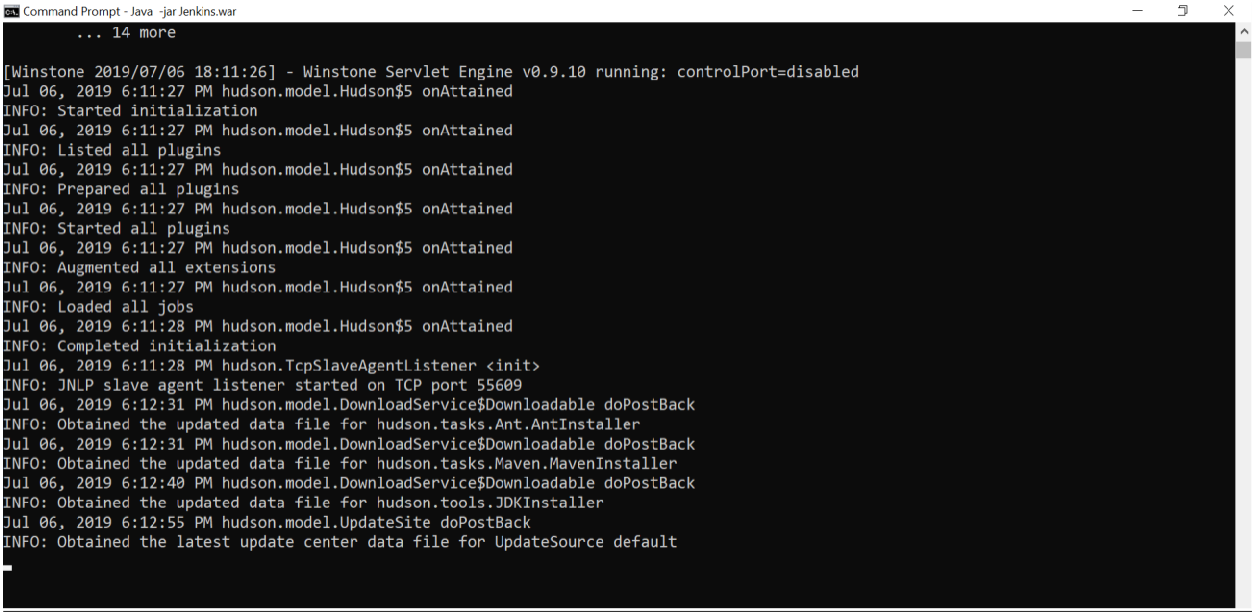
C:/Java -jar Jenkins.war

- When you run this command, various tasks will run, one of which is the extraction of the war file which is done by an embedded web server called winstone.





Click on **Allow access** button to allow the access.

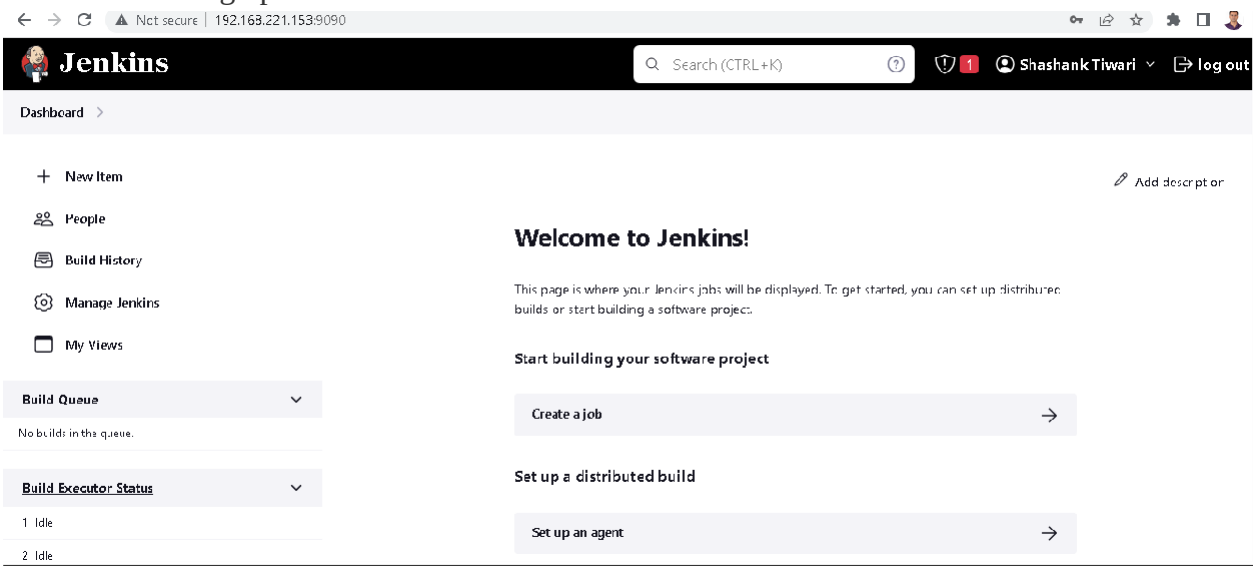


Accessing Jenkins

Now you can access the Jenkins. Open your browser and type the following url on your browser:

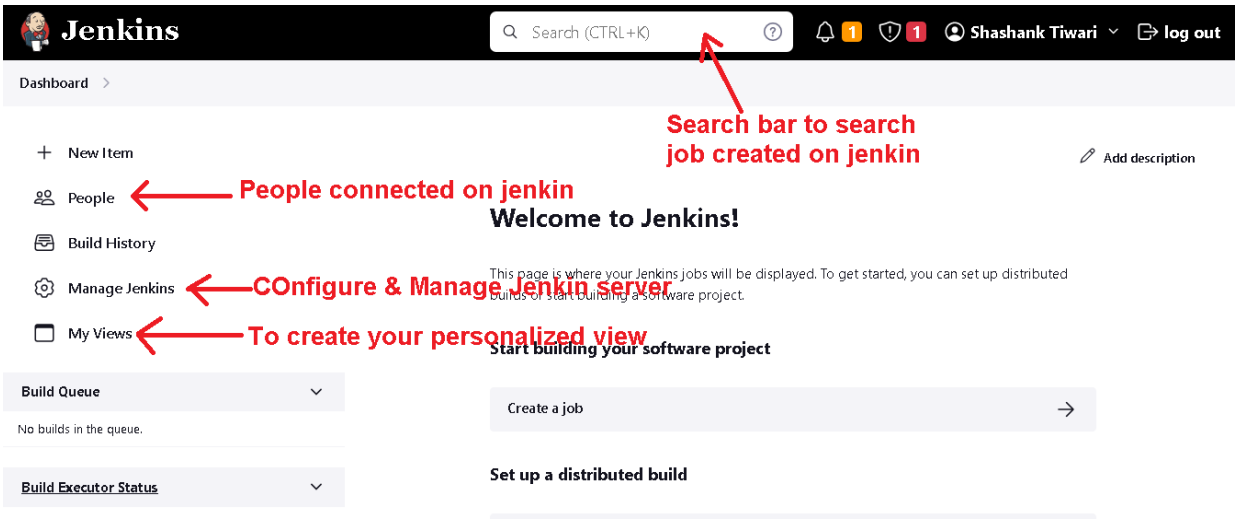
http://localhost:8080

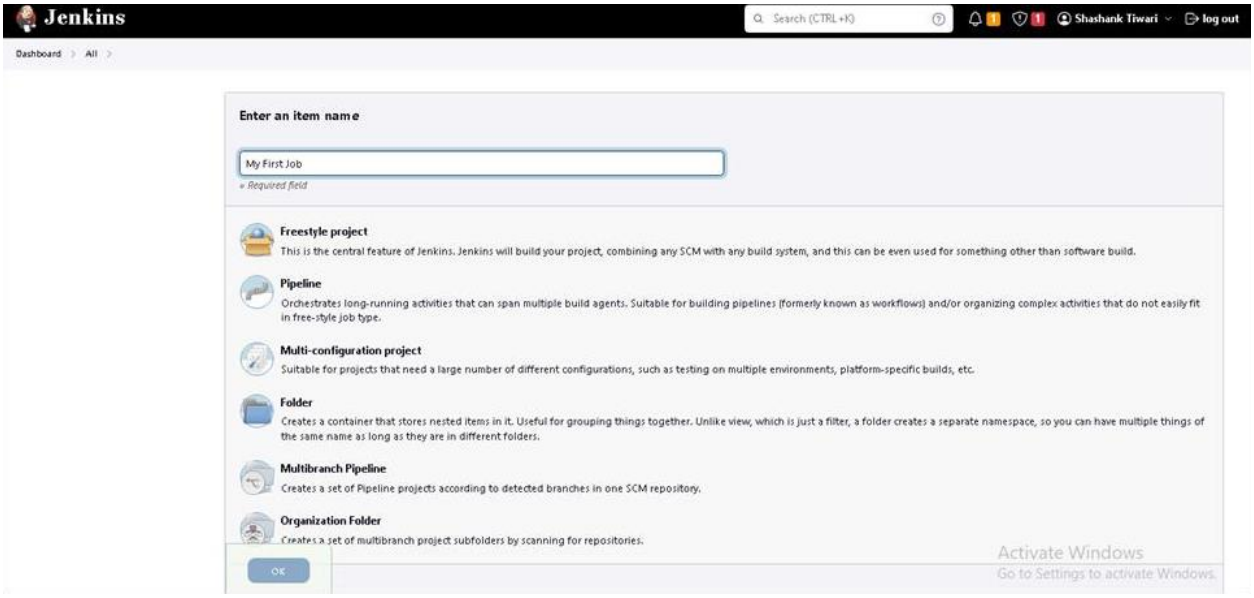
This url will bring up the Jenkins dashboard.



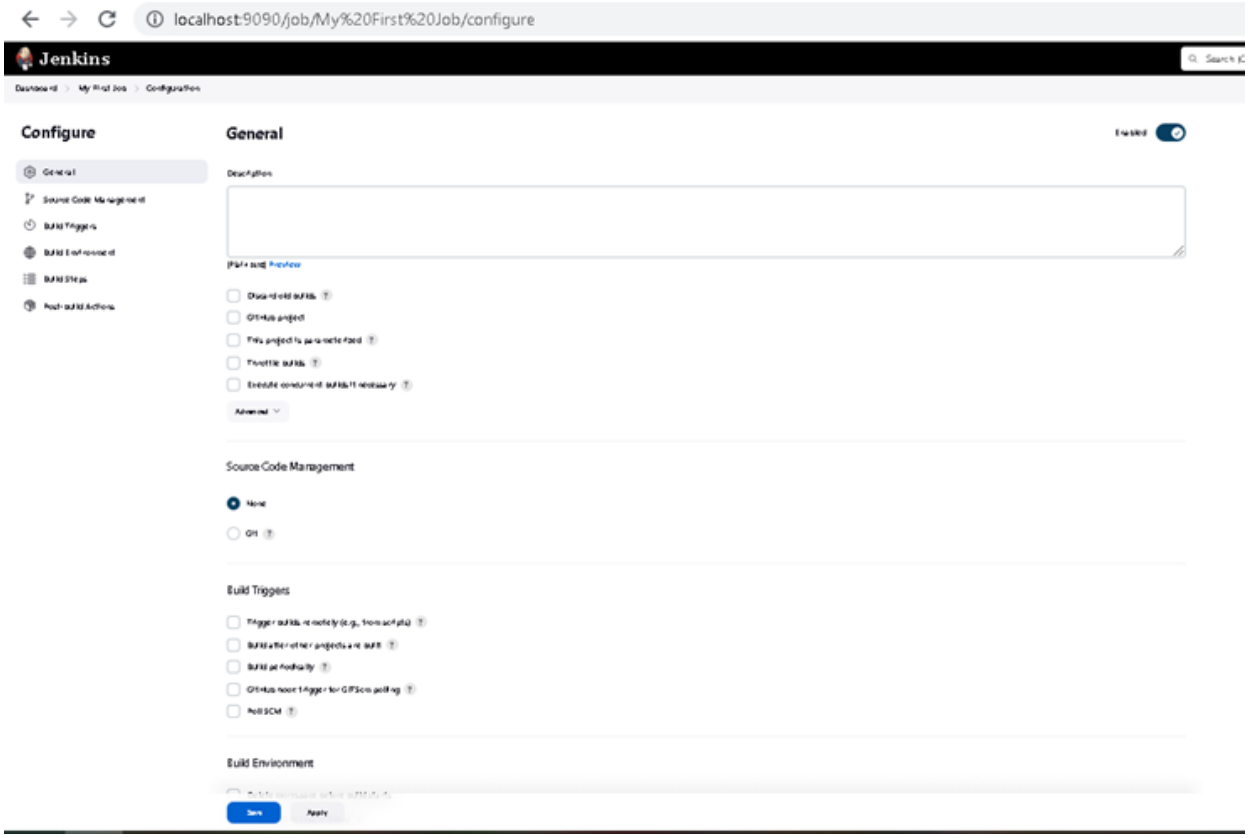
Exploring the Jenkin environment

- Login to your jenkin server

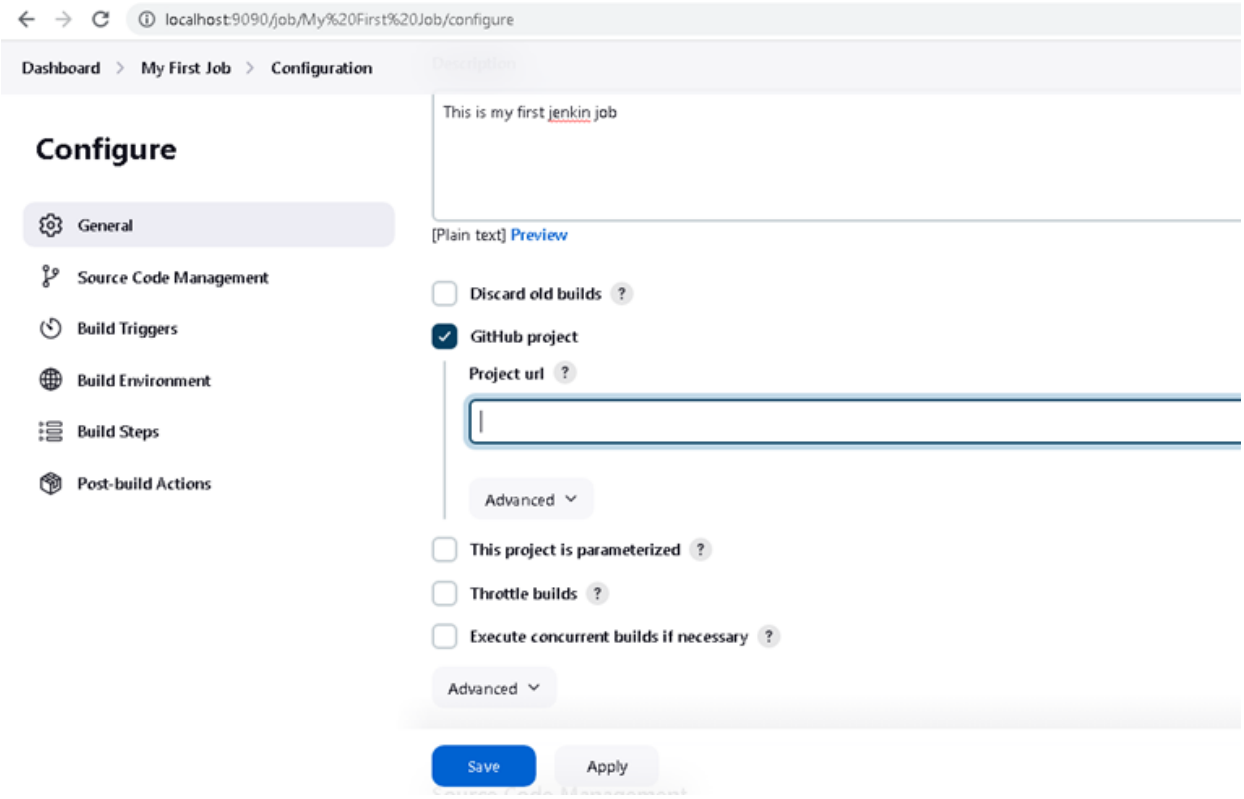




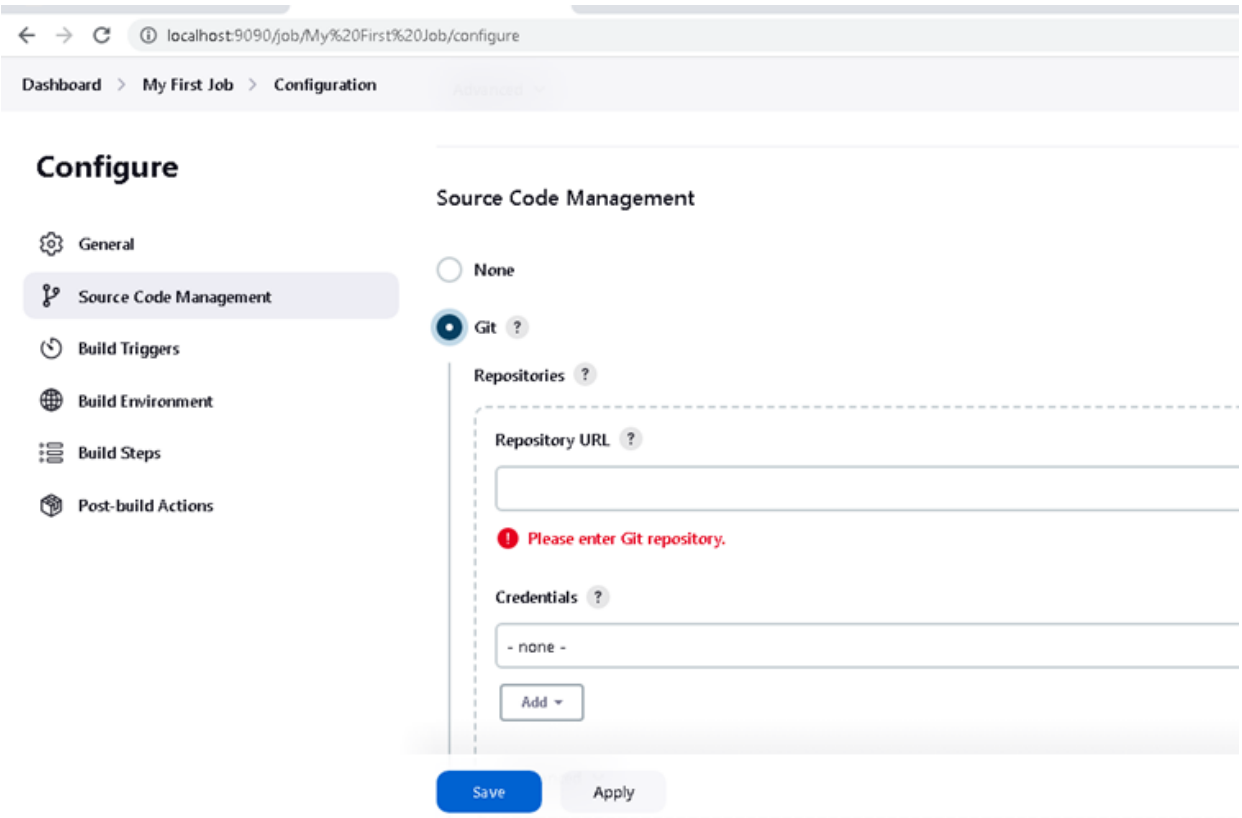
Next, define the parameter with the job



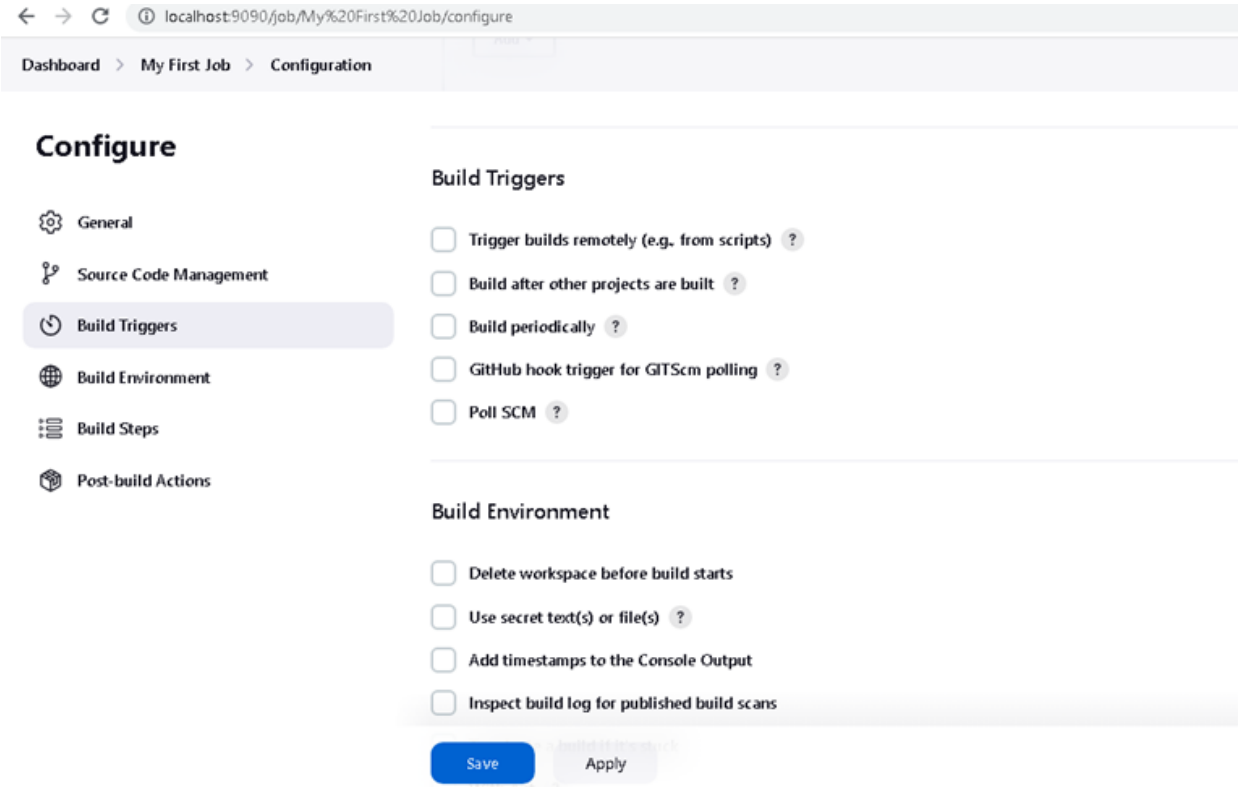
You can define your github project



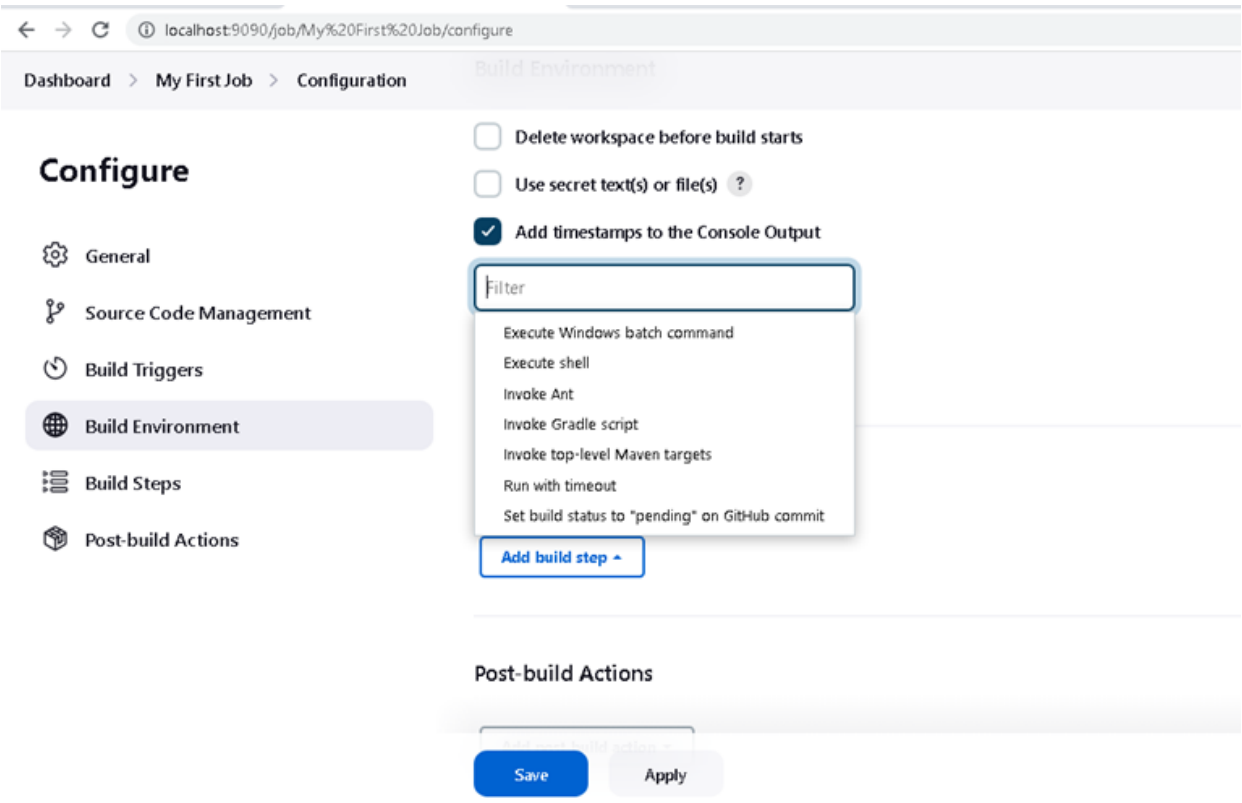
Source code management section

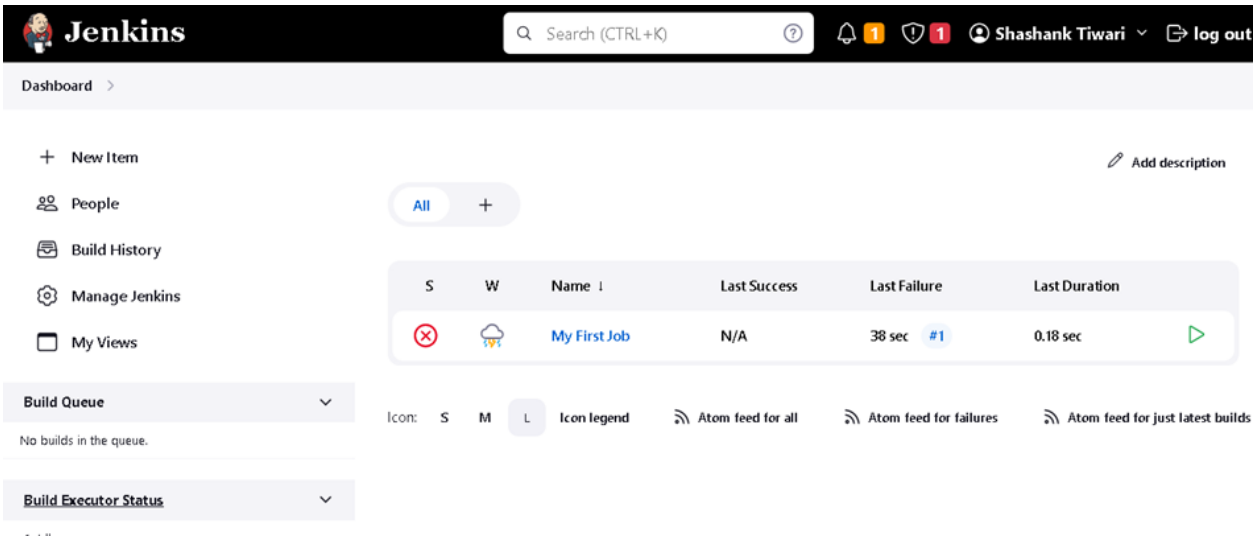


Next, Set the trigger of build in build trigger section



We can define the task in the build section




Jenkins
Search (CTRL+K)
1
1
Shashank Tiwari
log out
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
Add description
All
S
W
Name ↓
Last Success
Last Failure
Last Duration
My First Job
N/A
38 sec
#1
0.18 sec
Icon:
S
M
L
Icon legend
Atom feed for all
Atom feed for failures
Atom feed for just latest builds