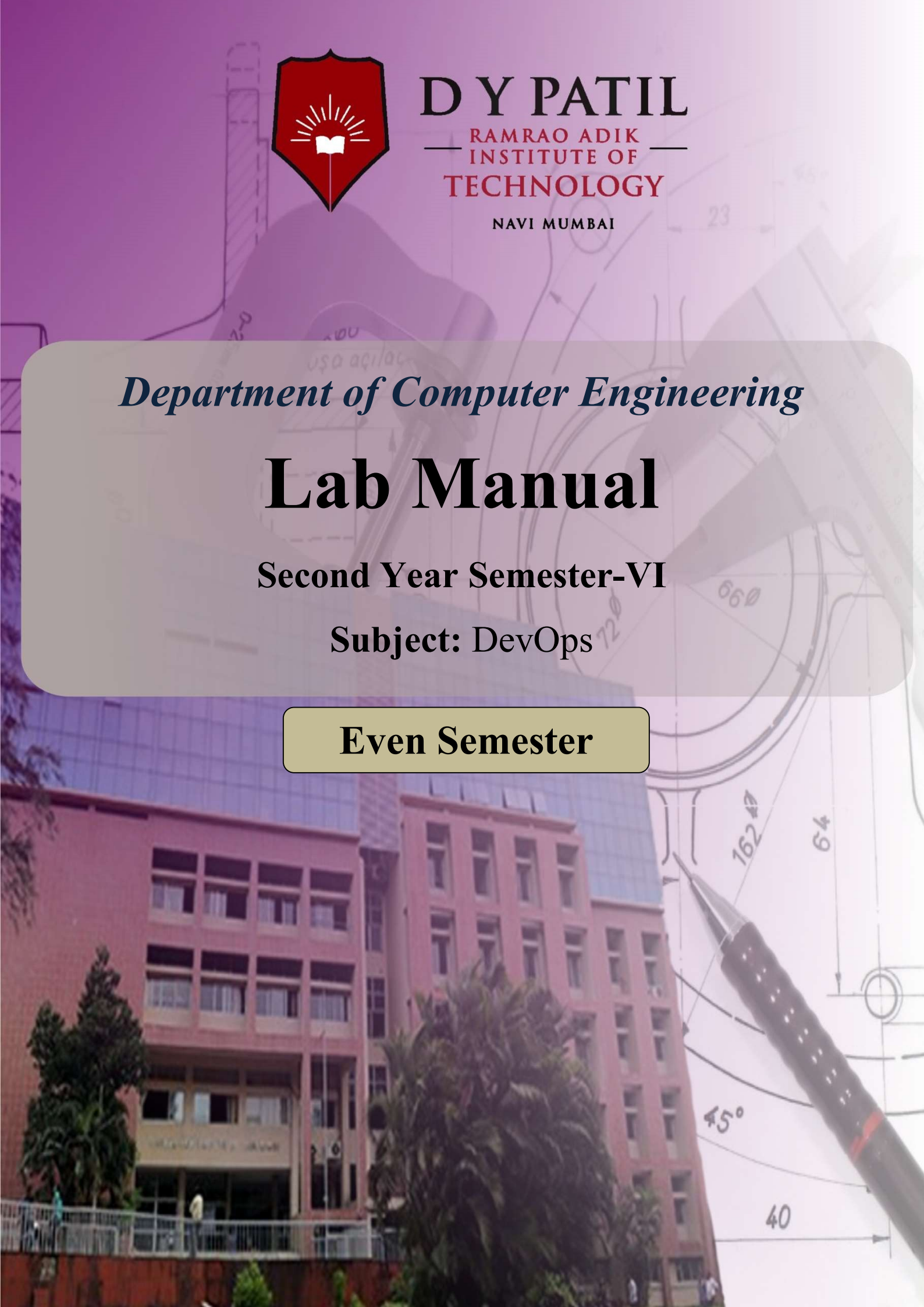


D Y PATIL
RAMRAO ADIK INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

Department of Computer Engineering

Lab Manual

Second Year Semester-VI

Subject: DevOps

Even Semester

Institutional Vision and Mission

Our Vision

To foster and permeate higher and quality education with value added engineering, technology programs, providing all facilities in terms of technology and platforms for all round development with societal awareness and nurture the youth with international competencies and exemplary level of employability even under highly competitive environment so that they are innovative adaptable and capable of handling problems faced by our country and world at large.

Our Mission

The Institution is committed to mobilize the resources and equip itself with men and materials of excellence thereby ensuring that the Institution becomes pivotal center of service to Industry, academia, and society with the latest technology. RAIT engages different platforms such as technology enhancing Student Technical Societies, Cultural platforms, Sports excellence centers, Entrepreneurial Development Center and Societal Interaction Cell. To develop the college to become an autonomous Institution & deemed university at the earliest with facilities for advanced research and development programs on par with international standards. To invite international and reputed national Institutions and Universities to collaborate with our institution on the issues of common interest of teaching and learning sophistication.

Our Quality Policy

ज्ञानधीनं जगत् सर्वम।

Knowledge is supreme.

OurQualityPolicy

Itisourearnestendeavourtoproducehighqualityengineeringprofessionalswhoare innovative and inspiring, thought and action leaders, competent to solve problems facedbysociety,nationandworldatlargebystrivingtowardsveryhighstandardsinlearning,teachingandtrainingmethodologies.

OurMotto: If itis not of quality, itis NOTRAIT!

Departmental VisionandMission

Vision

To impart higher and quality education in Computer Science with value added engineering and technology programs to prepare technically sound, ethically strong engineers with social awareness. To extend the facilities, to meet the fast changing requirements and nurture the youths with international competencies and exemplary level of employability and research under highly competitive environments.

Mission

To mobilize the resources and equip the department with men and materials of excellence to provide knowledge in the thrust areas of Computer Science and Engineering. To provide learning ambiance and exposure to the latest tools and technologies and the platforms to work on research and live projects. To provide the diverse platforms of sports, technical, co-curricular and extracurricular activities for the overall development of student with ethical attitude. To prepare the students to sustain the impact of computer education for social needs encompassing industry, educational institutions and public service. To collaborate with IITs, reputed universities and industries for the technical and overall upliftment of students for continuing learning and entrepreneurship.

Departmental Program Educational Objectives (PEOs)

1. **Learn and Integrate**
   To provide Computer Engineering students with a strong foundation in the mathematical, scientific and engineering fundamentals necessary to formulate, solve and analyze engineering problems and to prepare them for graduate studies.

2. **Think and Create**
   To develop an ability to analyze the requirements of the software and hardware, understand the technical specifications, create a model, design, implement and verify a computing system to meet specified requirements while considering real-world constraints to solve real world problems.

3. **Broad Base**
   To provide broad education necessary to understand the science of computer engineering and the impact of it in a global and social context.

4. **Techno-leader**
   To provide exposure to emerging cutting edge technologies, adequate training & opportunities to work as teams on multidisciplinary projects with effective communication skills and leadership qualities.

5. **Practice citizenship**
   To provide knowledge of professional and ethical responsibility and to contribute to society through active engagement with professional societies, schools, civic organizations or other community activities.

6. **Clarify Purpose and Perspective**
   To provide strong in-depth education through electives and to promote student awareness on the life-long learning to adapt to innovation and change, and to be successful in their professional work or graduate studies.

Departmental Program Outcomes (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 : Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 : Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 : Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 : Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes: PSO

PSO1: To build competencies towards problem solving with an ability to understand, identify, analyze and design the problem, implement and validate the solution including both hardware and software.

PSO2: To build appreciation and knowledge acquiring of current computer techniques with an ability to use skills and tools necessary for computing practice.

PSO3: To be able to match the industry requirements in the area of computer science and engineering. To equip skills to adopt and imbibe new technologies.

Index

List of Experiments

Sr. No.	Experiments Name
1	**Git installation and basic Git Commands** Perform installation of Git Bash and Explore usage of basic Git Commands. a. Git installation b. Git configuration c. Starting A Project d. Day-To-Day Work e. Git branching model f. Review your work
2	**Explore Git Commands to** a. Tagging known commits b. Reverting changes. c. Synchronizing repositories. d. Reverting Changes
3	**Installation of Jenkins** To install and configure Jenkins to setup a build Job.
4	**Continuous Integration** To build the pipeline of jobs in Jenkins, create a pipeline script to deploy an application over Server
5	**Installation of Jenkins over Tomcat Server** To install Tomcat Server on Windows and run Jenkins over Tomcat Server
6	**Test Software Applications** To Setup and Run Selenium Tests in Jenkins Using Maven
7	**Containerization with Docker** To understand Docker Architecture, install docker and execute docker commands to manage and interact with containers.
8	**Build an image with Docker** To learn Dockerfile instructions, build an image for sample web application on Docker Engine
9	To install and configure Pull based Software Configuration Management and provisioning tools using Ansible/Chef/Puppet
10	**Case Study** Comparative study of Software Development Life Cycle – Waterfall Cycle vs Agile vs DevOps.

Course Objective, Course Outcome&Experiment Plan

Course Objective:

1	To understand DevOps practices which aims to simplify Software Development Life
2	To be aware of different Version Control tools like GIT, CVS or Mercurial
3	To Integrate and deploy tools like Jenkins and Maven, which is used to build, test and deploy applications in DevOps environment
4	To be familiarized with selenium tool, which is used for continuous testing of applications deployed.
5	To use Docker to Build, ship and manage applications using containerization
6	To understand the concept of Infrastructure as a code and install and configure Ansible tool.

Course Outcomes:

CO1	To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.
CO2	To obtain complete knowledge of the “version control system” to effectively track changes augmented with Git and GitHub.
CO3	To understand the importance of Jenkins to Build and deploy Software Applications on server environment.
CO4	Understand the importance of Selenium and Jenkins to test Software Applications.
CO5	To understand concept of containerization and Analyze the Containerization of OS images and deployment of applications over Docker.
CO6	To Synthesize software configuration and provisioning using Ansible.

Experiment Plan:

Module No.	Week No.	Experiments Name	Course Outcome	Weightage
1	W1	**Git installation and basic Git Commands** Perform installation of Git Bash and Explore usage of basic Git Commands. a. Git installation b. Git configuration c. Starting A Project d. Day-To-Day Work e. Git branching model f. Review your work	CO2	05
2	W2	**Explore Git Commands to** a. Tagging known commits b. Reverting changes. c. Synchronizing repositories. d. Reverting Changes	CO2	05
3	W3	**Installation of Jenkins** To install and configure Jenkins to setup a build Job.	CO3	04
4	W4, W5	**Continuous Integration** To build the pipeline of jobs in Jenkins, create a pipeline script to deploy an application over Server	CO3	04
5	W6	**Installation of Jenkins over Tomcat Server** To install Tomcat Server on Windows and run Jenkins over Tomcat Server	CO3	02
6	W7, W8	**Test Software Applications** To Setup and Run Selenium Tests in Jenkins Using Maven	CO4	10
7	W9	**Containerization with Docker** To understand Docker Architecture, install docker and execute docker commands to manage and interact with containers.	CO5	05
8	W10	**Build an image with Docker** To learn Dockerfile instructions, build an image for sample web application on Docker Engine	CO5	05
9	W11	To install and configure Pull based Software Configuration Management and provisioning tools using Ansible/Chef/Puppet	CO6	10

10	W12	**Case Study** Comparative study of Software Development Life Cycle – Waterfall Cycle vs Agile vs DevOps.	CO1	10

Mapping of COs with POs:

Subject Weight	Course Outcomes (Weightage-100%)		Program Outcomes										
			PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
PRATICAL 100%	CO1	To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.	1	1	1	1	1				1		2	2
	CO2	To obtain complete knowledge of the "version control system" to effectively track changes augmented with Git and GitHub.	1	1	1		1	1			2	2		1
	CO3	To understand the importance of Jenkins to Build and deploy Software Applications on server environment.					2				2	2	2	2
	CO4	Understand the importance of Selenium and Jenkins to test Software Applications.					2				2	2	2	2
	CO5	To understand concept of containerization and Analyze the Containerization of OS images and deployment of applications over Docker.	1				2				2	1	2	2

	CO6	To Synthesize software configuration and provisioning using Ansible.					2				2	2	2	2

Mapping of Course outcomes with Program Specific Outcomes:

Course Outcomes		Contribution to Program Specific outcomes		
		PSO1	PSO2	PSO3
CO1	To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.	2	2	2
CO2	To obtain complete knowledge of the "version control system" to effectively track changes augmented with Git and GitHub.	2	2	2
CO3	To understand the importance of Jenkins to Build and deploy Software Applications on server environment.	2	2	2
CO4	Understand the importance of Selenium and Jenkins to test Software Applications.	2	2	2
CO5	To understand concept of containerization and Analyze the Containerization of OS images and deployment of applications over Docker.	2	2	2
CO6	To Synthesize software configuration and provisioning using Ansible.	2	2	2

Study and Evaluation Scheme

Course Code	Course Name	Teaching Scheme			Credits Assigned			
CESL601	**Skill Based Lab IV – DevOPs**	Theory	Practical	Tutorial	Theory	Practical	Tutorial	Total
			02+02*	-	-	02	-	02

Course Code	Course Name	Examination Scheme		
CESL601	**Skill Based Lab IV – DevOPs**	Term Work	Practical & Oral	Total
		25	25	50

Term Work:

1. Term work should consist of at least 10 experiments on above content.
2. The final certification and acceptance of term work ensures that satisfactory performance of laboratory work and minimum passing marks in term work.
3. Term Work: 25 Marks (Total) = 10 Marks (Experiments)+ 5 Marks (Mini Project)+ 5 Marks (Assignments)+ 5 Marks (Theory + Practical Attendance).

Practical/Experiments:

1. The final certification and acceptance of term work ensures that satisfactory perfor mance of laboratory work and minimum passing marks in term work.
2. Practical exam will be based on the above syllabus.

Skill Based Lab IV – DevOPs

Experiment No. : 1

Git installation and basic Git Commands

Experiment No. 1

1. **Aim:** Perform installation of Git Bash and explore usage of basic Git Commands.
    a. Git installation
    b. Git configuration
    c. Starting A Project
    d. Day-To-Day Work
    e. Git branching model
    f. Review your work

2. **Objectives:** From this experiment, the student will be able
    - To understand DevOps practices which aims to simplify Software Development Life.
    - To be aware of different Version Control tools like GIT, CVS or Mercurial

3. **Outcomes:** The learner will be able to
    - To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.
    - To obtain complete knowledge of the "version control system" to effectively track changes augmented with Git and GitHub.

4. **Hardware / Software Required :** Linux / Windows Operating System

5. **Theory:**

What is version control?

How version control helps high performing development and DevOps teams prosper

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software teams work faster and smarter. They are especially useful for DevOps teams since they help them to reduce development time and increase successful deployments.

Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

Good version control software supports a developer's preferred workflow without imposing one particular way of working. Ideally it also works on any platform, rather than dictate what operating system or tool chain developers must use. Great version control systems facilitate a smooth and continuous flow of changes to the code rather than the

frustrating and clumsy mechanism of file locking - giving the green light to one developer at the expense of blocking the progress of others.

Git is the best choice for most software teams today.

a. Git Install

You can download Git for free from the following website: https://www.git-scm.com/

Git for Windows stand-alone installer

Download the latest Git for Windows installer from https://gitforwindows.org/

When you've successfully started the installer, you should see the **Git Setup** wizard screen. Follow the **Next** and **Finish** prompts to complete the installation. The default options are pretty sensible for most users.

Using Git with Command Line

To start using Git, we are first going to open up our Command shell.

For Windows, you can use Git bash, which comes included in Git for Windows. For Mac and Linux you can use the built-in terminal.

The first thing we need to do, is to check if Git is properly installed:

Example

```
git --version

git version 2.30.2.windows.1
```

If Git is installed, it should show something like git version X.Y

b. Configure Git

Run the following commands to configure your Git username and email using the following commands, replacing Emma's name with your own. These details will be associated with any commits that you create:

```
$ git config --global user.name"Emma Paris" $ git config --
globaluser.email"eparis@atlassian.com"
```

Now let Git know who you are. This is important for version control systems, as each Git commit uses this information:

Example

```
git config --global user.name "pujagit"

git config --global user.email"padiya.puja@rait.ac.in"
```

Change the user name and e-mail address to your own. You will probably also want to use this when registering to GitHub later on.

Note: Use global to set the username and e-mail for **every repository** on your computer.

If you want to set the username/e-mail for just the current repo, you can remove global

c. Creating Git Folder (Starting a project)

Now, let's create a new folder for our project:

Example

mkdirmyproject

cdmyproject

mkdir **make**s a **new directory**.

cd **changes** the **current working directory**.

Now that we are in the correct directory. We can start by initializing Git!

Note: If you already have a folder/directory you would like to use for Git:

Navigate to it in command line, or open it in your file explorer, right-click and select "Git Bash here"

Initialize Git

Once you have navigated to the correct folder, you can initialize Git on that folder:

Example

gitinit

Initialized empty Git repository in /Users/user/myproject/.git/

You just created your first Git Repository!

Note: Git now knows that it should watch the folder you initiated it on.

Git creates a hidden folder to keep track of changes.

Note: Create a new local repository. If [project name] is provided, Git will create a new directory name [project name] and will initialize a repository inside it. If [project name] is not provided, then a new repository is initialized in the current directory.

$ git clone [project url]

Downloads a project with the entire history from the remote repository.

d. Day-To-Day Work

$ git status
Displays the status of your working directory. Options include new, staged, and modified files. It will retrieve branch name, current commit identifier, and changes pending commit.

$ git add [file]
Add a file to the staging area. Use in place of the full file path to add all changed files from the current directory down into the directory tree

$ git diff [file]
Show changes between working directory and staging area.

$ git diff --staged [file]
Shows any changes between the staging area and the repository.

$ git checkout -- [file]
Discard changes in working directory. This operation is unrecoverable.

$ git reset [file]
Revert your repository to a previous known working state.

$ git commit
Create a new commit from changes added to the staging area. The commit must have a message!

$ git rm [file]
Remove file from working directory and staging area.

e. <u>Git branching model</u>

$ git branch [-a]
List all local branches in repository. With -a: show all branches (with remote).

$ git branch [branch_name]
Create new branch, referencing the current HEAD.

$ git checkout [-b][branch_name]
Switch working directory to the specified branch. With -b: Git will create the specified branch if it does not exist.

$ git merge [from name]
Join specified [from name] branch into your current branch (the one you are on currently).

$ git branch -d [name]
Remove selected branch, if it is already merged into any other. -D instead of -d forces deletion.

f. Review your work

$ git log [-n count]
List commit history of current branch. -n count limits list to last n commits.

$ git log --oneline --graph --decorate
An overview with reference labels and history graph. One commit per line.

$ git log ref..
List commits that are present on the current branch and not merged into ref. A ref can be a branch name or a tag name.
$ git log ..ref
List commit that are present on ref and not merged into current branch.

$ git reflog
List operations (e.g. checkouts or commits) made on local repository

6. Conclusion and Discussion:
Students are supposed to write your own conclusion

7. Viva Questions:

- What is version control?
- What is staging area?
- What do mean by repository?
- What care is to be taken when merging two branches?

8. References:

- Jon Loeliger and Matthew McCullough, Version Control with Git, O'Reilly Media, Second Edition, 2012.
- Dennis Hutten, Git: Learn Version Control with Git A Step-By-Step Ultimate Beginners Guide.
- Scott Chacon and Ben Straub, Pro Git_ Everything You Need to Know About Git, apress, Second Edition

Skill Based Lab IV – DevOPs

Experiment No.: 2

Create and fork repositories in GitHub

Experiment No. 2

1. **Aim:** Create and fork repositories in GitHub
    a. Creating a GitHub account
    b. Synchronizing repositories.
    c. Tagging known commits
    d. Reverting changes

2. **Objectives:** From this experiment, the student will be able
    - To understand DevOps practices which aims to simplify Software Development Life.
    - To be aware of different Version Control tools like GIT, CVS or Mercurial

3. **Outcomes: The learner will be able to**
    - To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.
    - To obtain complete knowledge of the "version control system" to effectively track changes augmented with Git and GitHub.

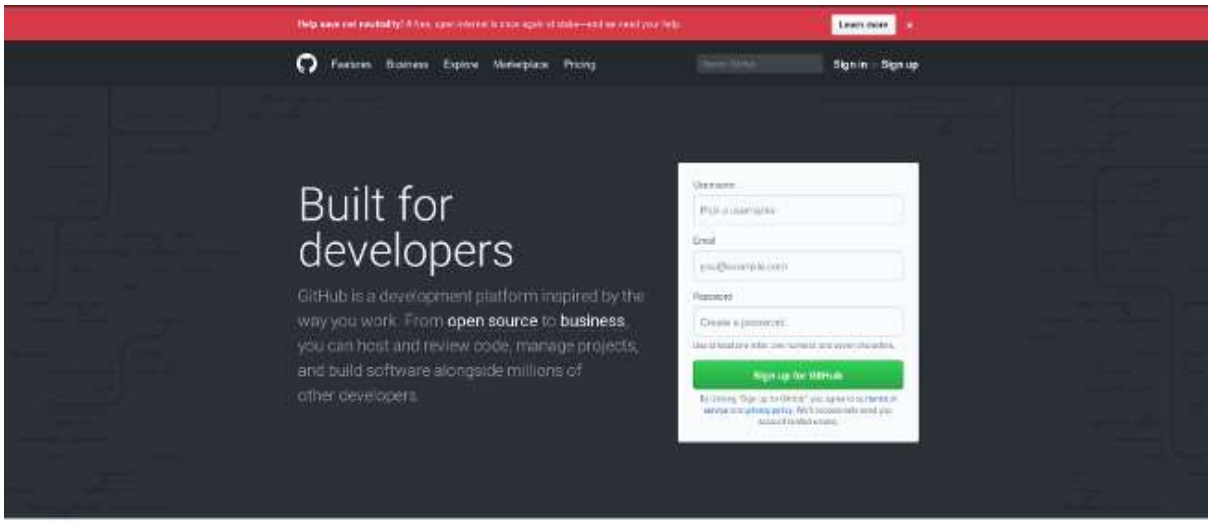
4. **Hardware / Software Required : Linux / Windows Operating System**

5. **Theory:**

a. <u>Creating a GitHub account</u>

1. Step 1: Create a GitHub account

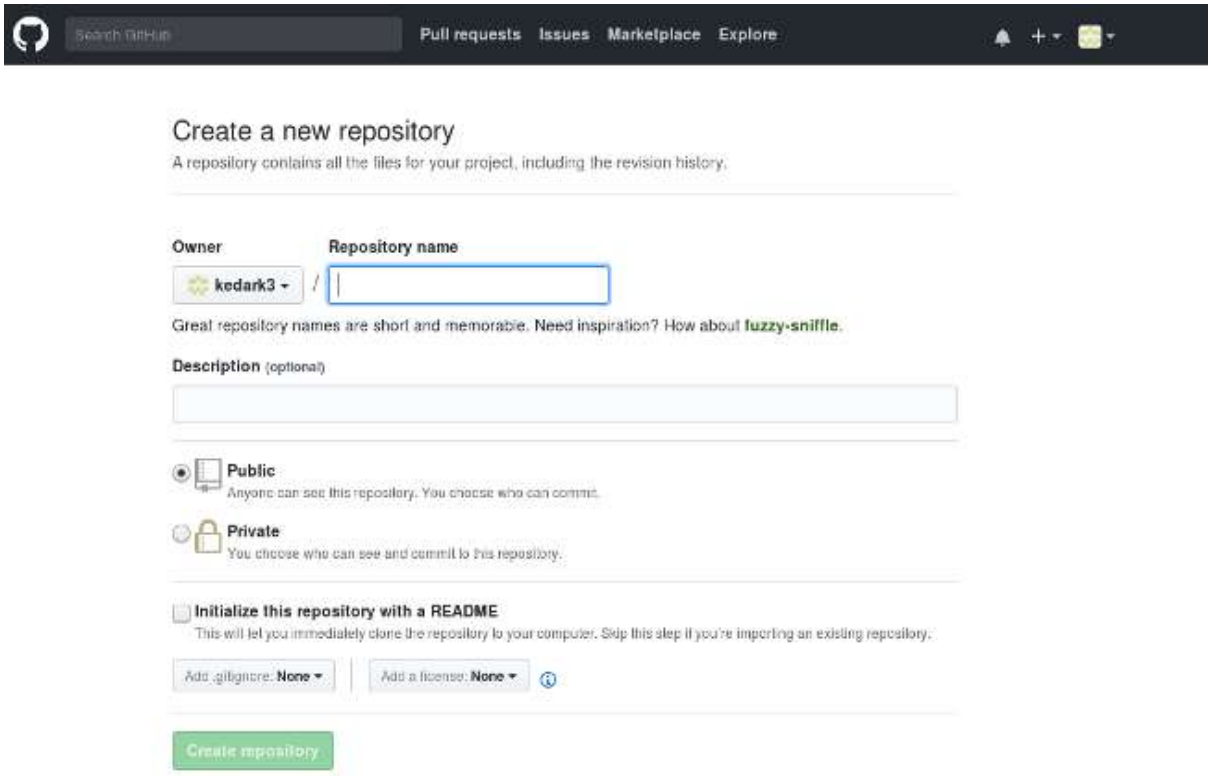
The easiest way to get started is to create an account on GitHub.com (it's free).



Pick a username (e.g., octocat123), enter your email address and a password, and click **Sign up for GitHub**. Once you are in, it will look something like this:

2. Step 2: Create a new repository

A repository is like a place or a container where something is stored; in this case we're creating a Git repository to store code. To create a new repository, select **New Repository** from the + sign dropdown menu (you can see I've selected it in the upper-right corner in the image above).

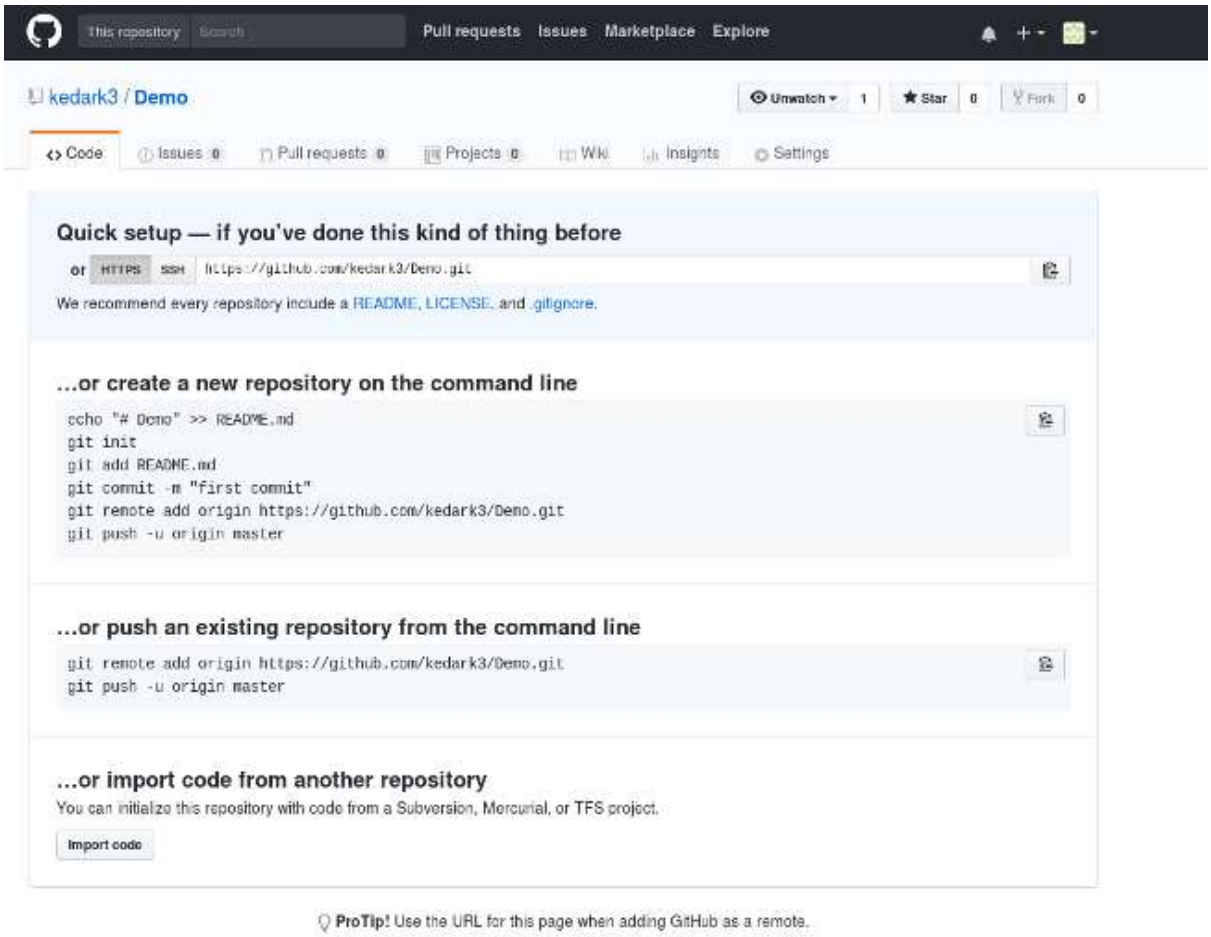


Enter a name for your repository (e.g, "Demo") and click **Create Repository**. Don't worry about changing any other options on this page.

Congratulations! You have set up your first repo on GitHub.com.

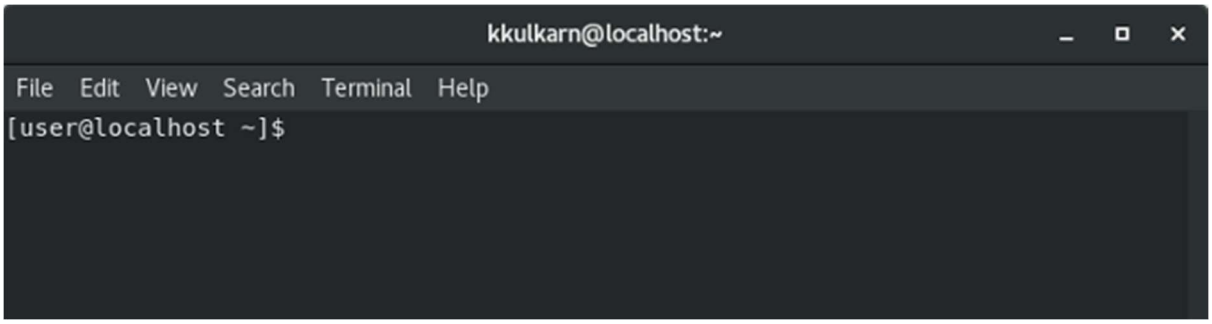
3. Step 3: Create a file

Once your repo is created, it will look like this:



Don't panic, it's simpler than it looks. Stay with me. Look at the section that starts "...or create a new repository on the command line," and ignore the rest for now.

Open the *Terminal* program on your computer.



Type git and hit **Enter**. If it says command bash: git: command not found, then install Git with the command for your Linux operating system or distribution. Check the installation by typing git and hitting **Enter**; if it's installed, you should see a bunch of information about how you can use the command.

In the terminal, type:

mkdir Demo

This command will create a directory (or folder) named *Demo*.

Change your terminal to the *Demo* directory with the command:

cd Demo

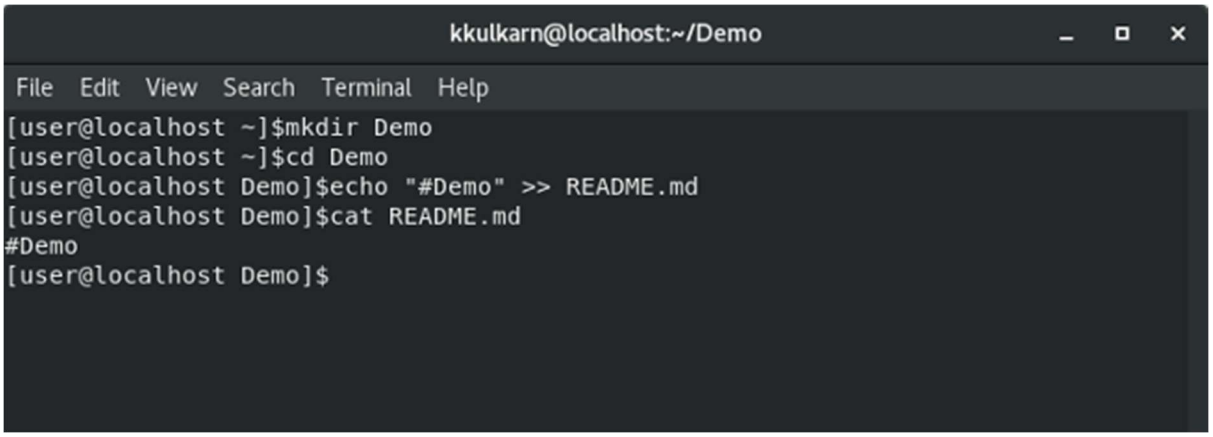
Then enter:

echo "#Demo" >> README.md

This creates a file named README.md and writes #Demo in it. To check that the file was created successfully, enter:

cat README.md

This will show you what is inside the README.md file, if the file was created correctly. Your terminal will look like this:



To tell your computer that *Demo* is a directory managed by the Git program, enter:

git init

Then, to tell the Git program you care about this file and want to track any changes from this point forward, enter:

git add README.md

4. Step 4: Make a commit

So far you've created a file and told Git about it, and now it's time to create a *commit*. Commit can be thought of as a milestone. Every time you accomplish some work, you can write a Git commit to store that version of your file, so you can go back later and see what it looked like at that point in time. Whenever you make a change to your file, you create a new version of that file, different from the previous one.

To make a commit, enter:

git commit -m "first commit"

That's it! You just created a Git commit and included a message that says *first commit*. You must always write a message in commit; it not only helps you identify a commit, but it also enables you to understand what you did with the file at that point. So tomorrow, if you add a new piece of code in your file, you can write a commit message that says, *Added new code*, and when you come back in a month to look at your commit history or Git log (the list of commits), you will know what you changed in the files.

b. Synchronizing repositories

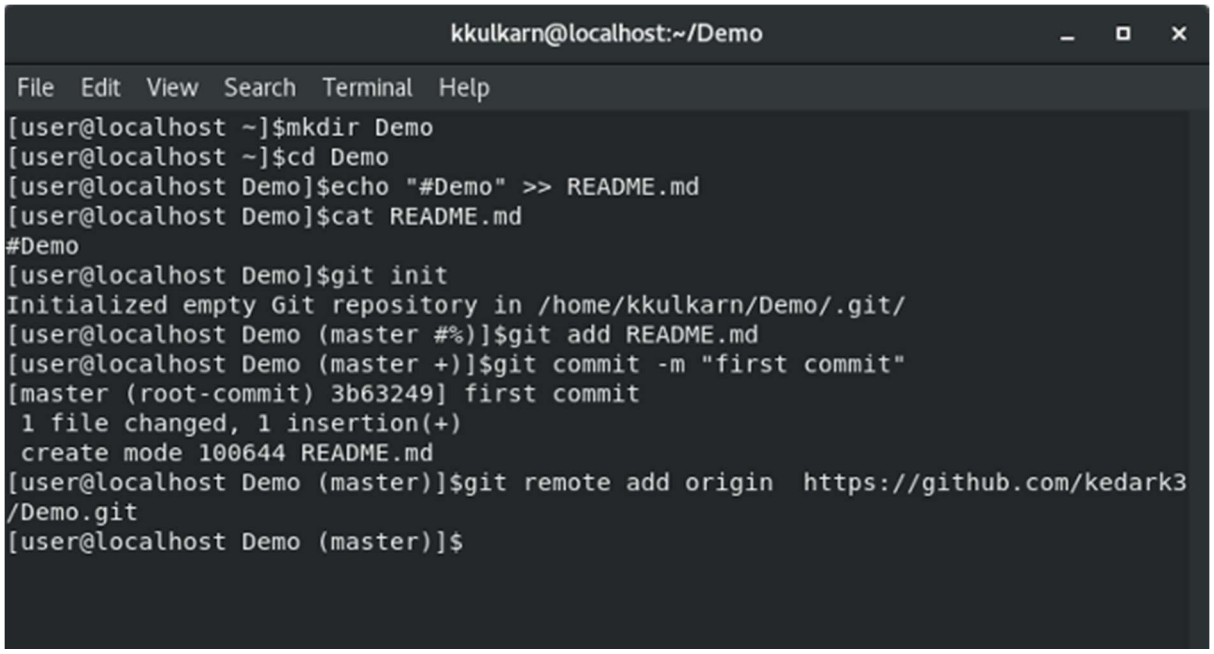
Step: Connect your GitHub repo with your computer

Now, it's time to connect your computer to GitHub with the command:

git remote add origin https://github.com/<your_username>/Demo.git

Let's look at this command step by step. We are telling Git to add a remote called origin with the address https://github.com/<your_username>/Demo.git (i.e., the URL of your Git repo on GitHub.com). This allows you to interact with your Git repository on GitHub.com by typing origin instead of the full URL and Git will know where to send your code.
Why origin? Well, you can name it anything else if you'd like.

Now we have connected our local copy of the *Demo* repository to its remote counterpart on GitHub.com. Your terminal looks like this:

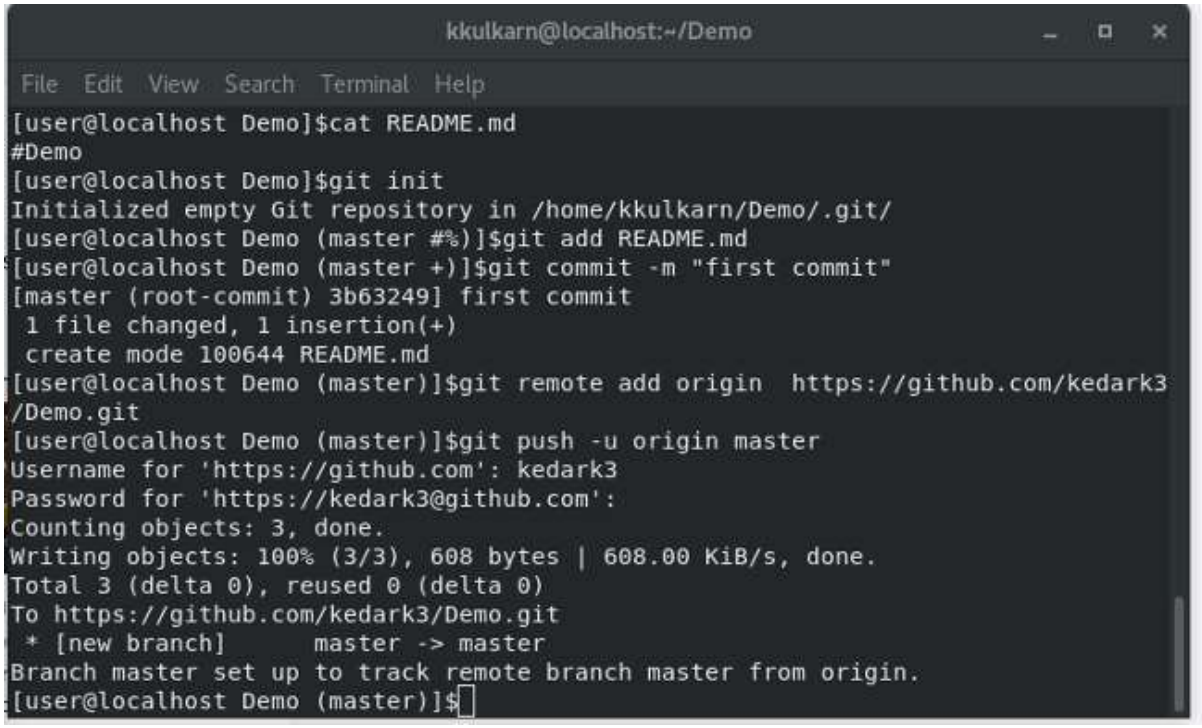


Now that we have added the remote, we can push our code (i.e., upload our README.md file) to GitHub.com.

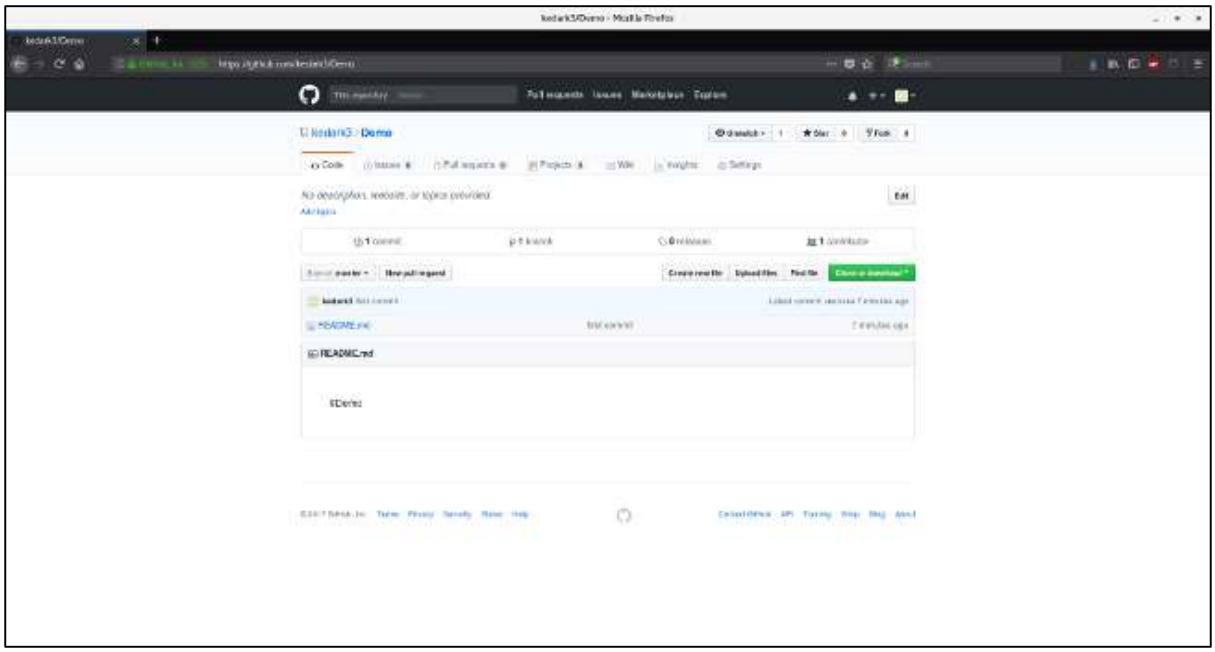
$ git push [--tags] [remote]
Push local changes to the remote. Use --tags to push tags.

$ git push -u [remote] [branch]
Push local branch to remote repository. Set its copy as an upstream.

Once you are done, your terminal will look like this:



And if you go to https://github.com/<your_username>/Demo you will see something like this:



That's it! You have created your first GitHub repo, connected it to your computer, and pushed (or uploaded) a file from your computer to your repository called *Demo* on GitHub.com.

$ git fetch [remote]

Fetch changes from the remote, but not update tracking branches.

$ git fetch --prune [remote]
Delete remote Refs that were removed from the remote repository.

$ git pull [remote]
Fetch changes from the remote and merge current branch with its upstream.

c. Tagging known commits

$ git tag
List all tags.

$ git tag [name] [commit sha]
Create a tag reference named name for current commit. Add commit sha to tag a specific commit instead of current one.

$ git tag -a [name] [commit sha]
Create a tag object named name for current commit.

$ git tag -d [name]
Remove a tag from local repository

d. Reverting changes

$ git reset [--hard] [target reference]
Switches the current branch to the target reference, leaving a difference as an uncommitted change. When --hard is used, all changes are discarded.

$ git revert [commit sha]
Create a new commit, reverting changes from the specified commit. It generates an inversion of changes.

6. Conclusion and Discussion:

Students are supposed to write your own conclusion.

7. Viva Questions:

- What is difference between git and github?
- How do you push your project into remote repository?
- Is it possible to revert changes after commit? Is so, how?

8. References:

- Jon Loeliger and Matthew McCullough, Version Control with Git, O'Reilly Media, Second Edition, 2012.
- Dennis Hutten, Git: Learn Version Control with Git A Step-By-Step Ultimate Beginners Guide.
- Scott Chacon and Ben Straub, Pro Git_ Everything You Need to Know About Git, apress, Second Edition.

Skill Based Lab IV – DevOPs

Experiment No.: 3

Installation of Jenkins

To install and configure Jenkins to setup a build Job.

Experiment No. 3

1. Aim: To install and configure Jenkins to setup a build Job.

2. **Objectives:** From this experiment, the student will be able
   9. To install and build job in Jenkins for deploying application DevOps environment.

3. Outcomes: The learner will be able to
   10. To understand the importance of Jenkins to Build and deploy Software Applications on server environment.

4. Hardware / Software Required: Linux / Windows Operating System

5. Theory:

Jenkins is a Java-based open-source automation platform with built-in continuous integration plugins. Jenkins is used to continuously build and test your software projects, making it simpler for developers to incorporate changes to the project and for users to get a new build. By interacting with a wide range of testing and deployment tools, it also enables you to release your software continually.

Organizations can use Jenkins to automate and speed up the software development process. Jenkins combines all stages of the development lifecycle, including build, document, test, package, stage, deploy, static analysis, and many others.

Jenkins achieves Continuous Integration with the help of plugins. Plugins allow the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool. For example: Git, Maven 2 project, Amazon EC2, HTML publisher etc.

Advantages of Jenkins include:

- It is an open-source tool with great community support.
- It is easy to install.
- It has 1000+ plugins to ease your work. If a plugin does not exist, you can code it and share it with the community.
- It is free of cost.
- It is built with Java and hence, it is portable to all the major platforms.

Jenkins Features:

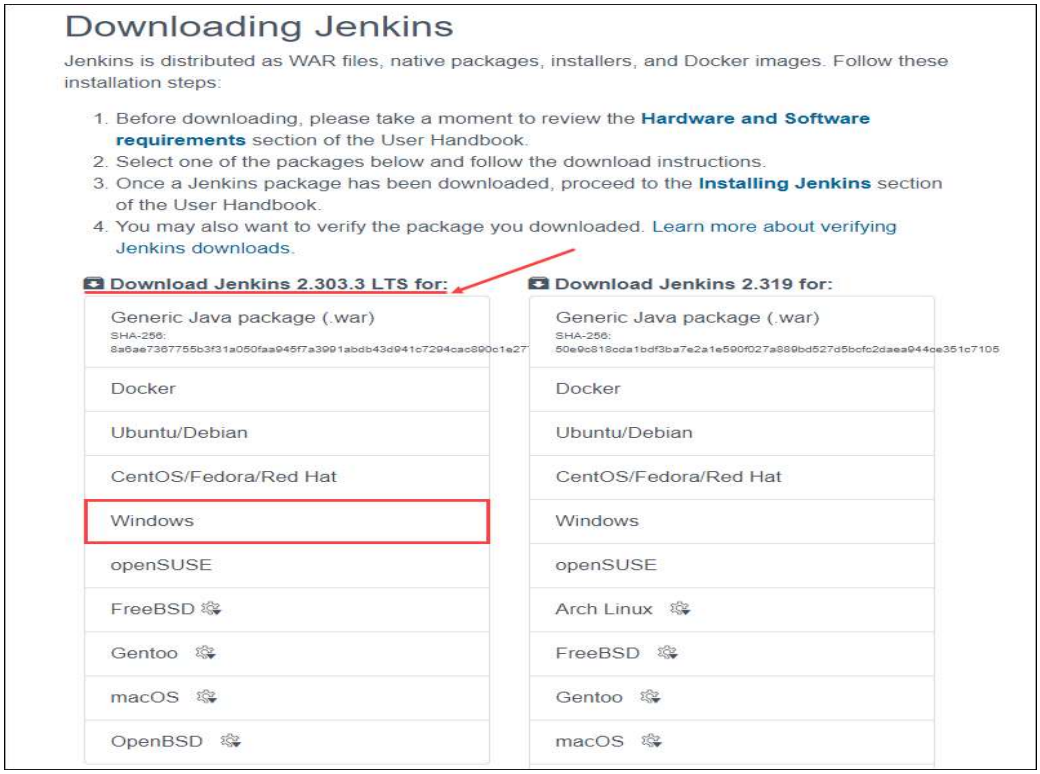
11. Continuous Integration and Continuous Delivery: As an extensible automation server, Jenkins can be used as a simple CI server or turned into the continuous delivery hub for any project.

12. **Easy configuration**
    Jenkins can be easily set up and configured via its web interface, which includes on-the-fly error checks and built-in help.
13. **Easy installation**
    Jenkins is a self-contained Java-based program, ready to run out-of-the-box, with packages for Windows, Linux, macOS and other Unix-like operating systems.
14. **Plugins**
    With hundreds of plugins in the Update Center, Jenkins integrates with practically every tool in the continuous integration and continuous delivery toolchain.
15. **Extensible**
    Jenkins can be extended via its plugin architecture, providing nearly infinite possibilities for what Jenkins can do.
16. **Distributed**
    Jenkins can easily distribute work across multiple machines, helping drive builds, tests and deployments across multiple platforms faster.

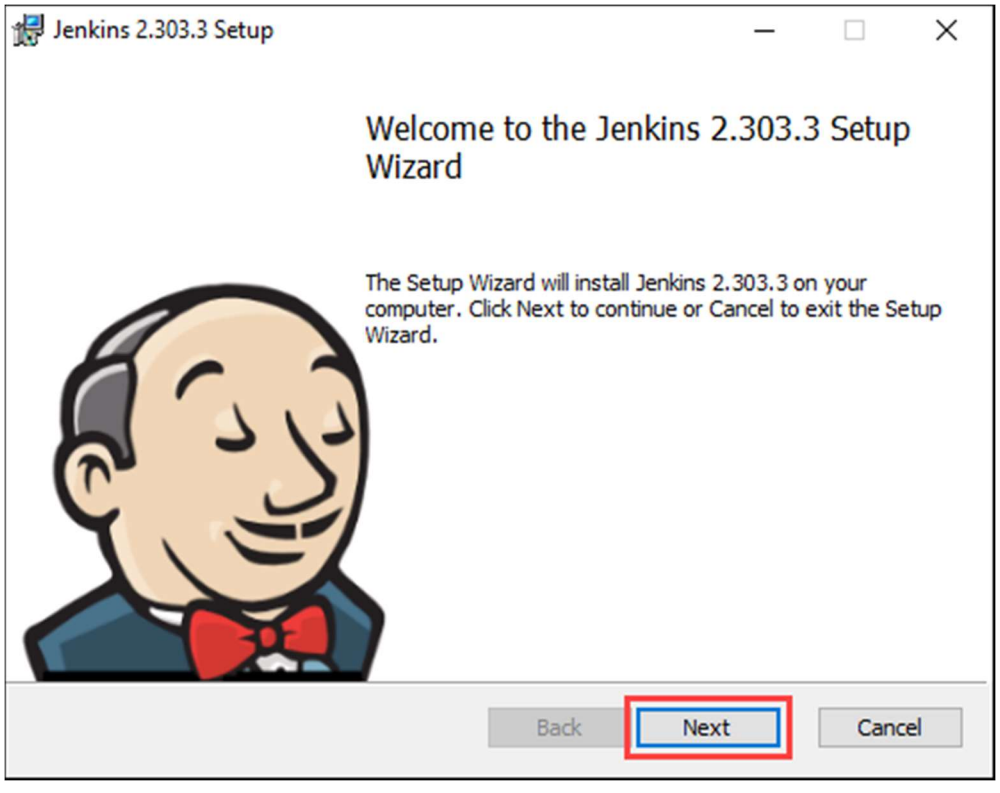
Steps for Installation of Jenkins:

1. **Installation of JAVA:** Download JDK 11/17 and choose windows 32-bit or 64-bit according to your system configuration. Click on "accept the license agreement." Set the Path for the Environmental Variable for JDK:
   - Go to System Properties. Under the "Advanced" tab, select "Environment Variables."
   - Under system variables, select "new." Then copy the path of the JDK folder and paste it in the corresponding value field. Similarly, do this for JRE.
   - Under system variables, set up a bin folder for JDK in PATH variables.
   - Go to command prompt and type the following to check if Java has been successfully installed:

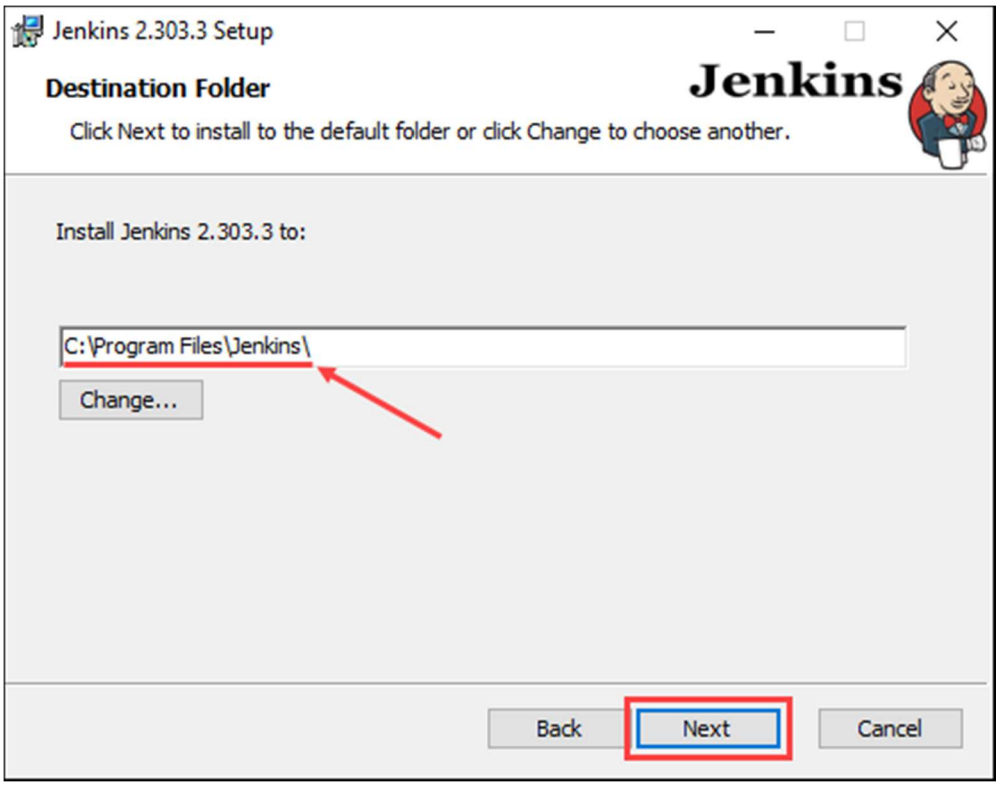
     C:\Users\Aditi>java -version
2. Browse to the official Jenkins download page. Under the Downloading Jenkins section is a list of installers for the long-term support (LTS) version of Jenkins. Click the Windows link to begin the download.
   Once the download is complete, run the **jenkins.msi** installation file.



3. The setup wizard starts. Click **Next** to proceed.



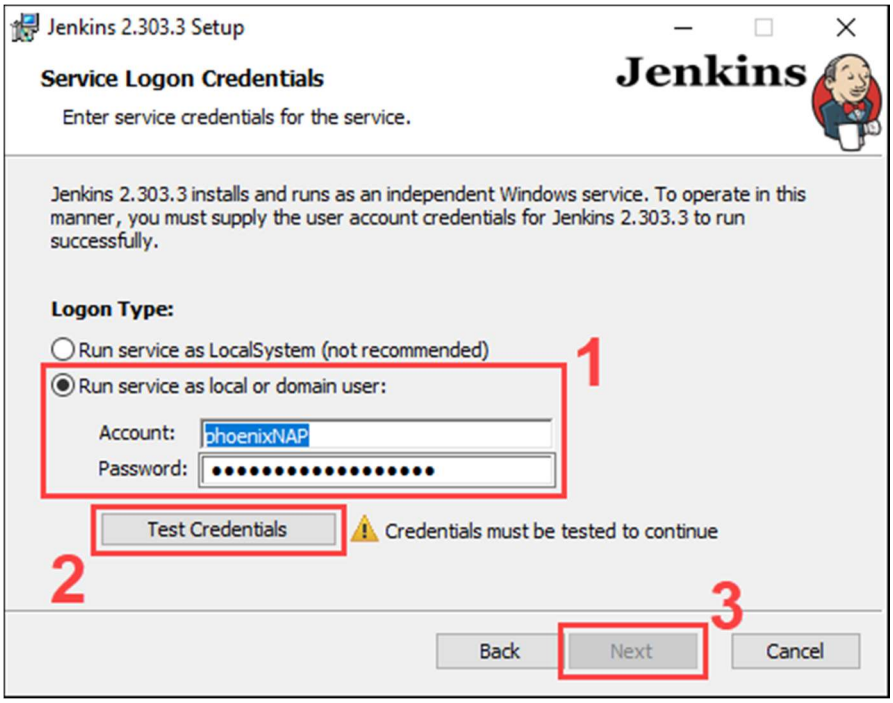
4. Select the install destination folder and click **Next** to continue.



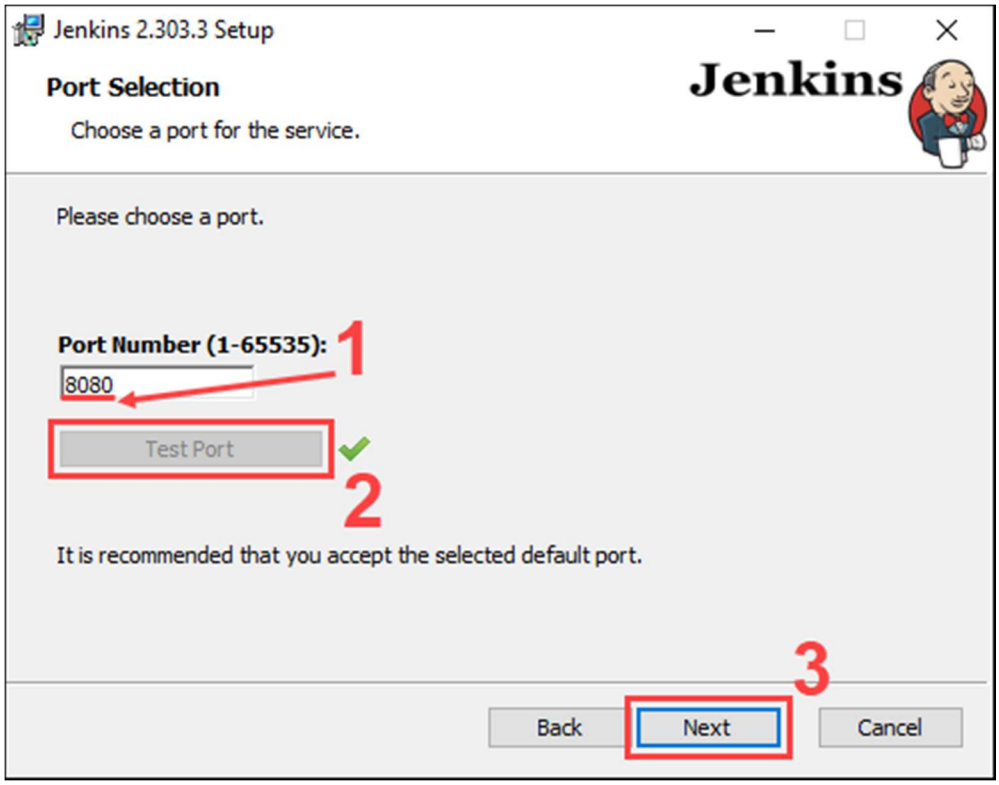
5. Under the **Run service as a local or domain user** option, enter the domain username and password for the user account you want to run Jenkins with. Click **Test Credentials** to verify the login data, then click **Next** to proceed.

Or

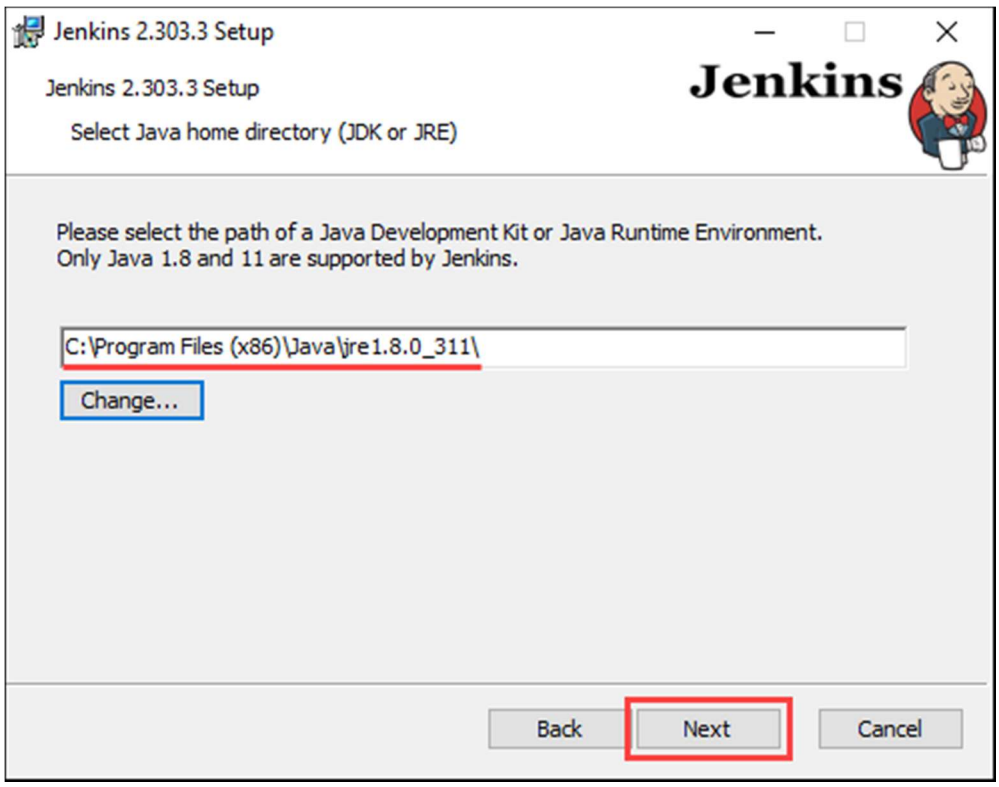
Using the **Run service as LocalSystem** option doesn't require you to enter user login data. Instead, it grants Jenkins full access to your system and services. Note that this is a less secure option and is thus not recommended.



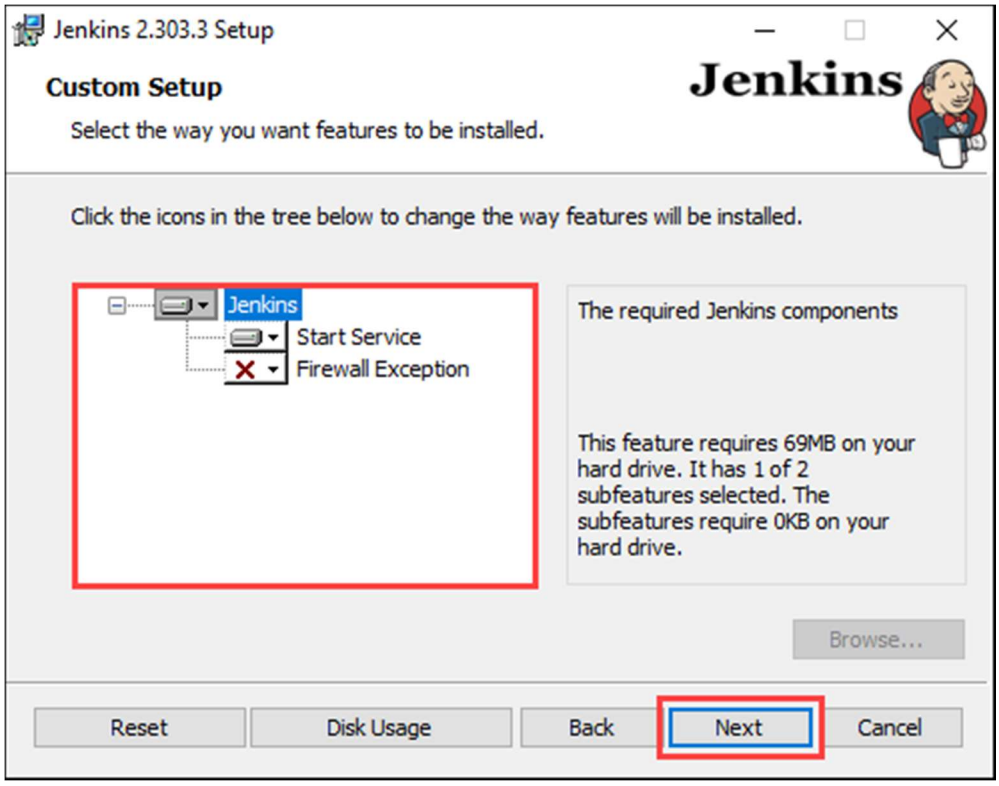
6.Enter the port number you want Jenkins to run on. Click **Test Port** to check if the selected port is available, then click **Next** to continue.



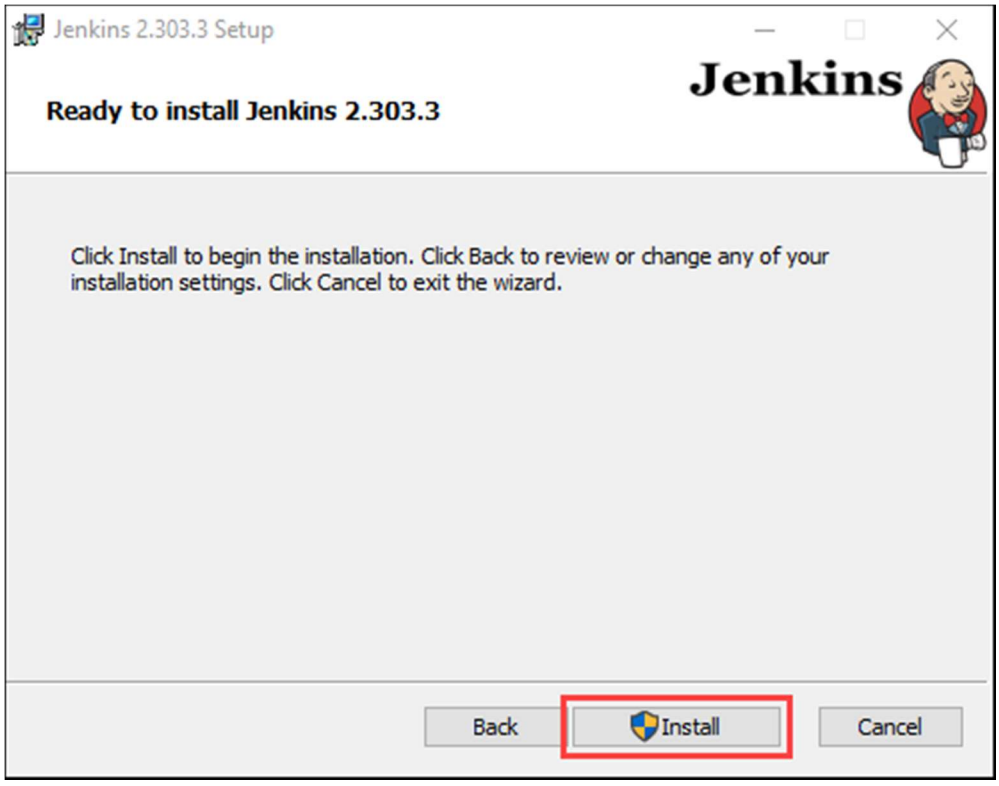
7. Select the directory where Java is installed on your system and click **Next** to proceed.



8. Select the features you want to install with Jenkins and click **Next** to continue.



9. Click **Install** to start the installation process.



10. Once the installation is complete, click **Finish** to exit the install wizard.

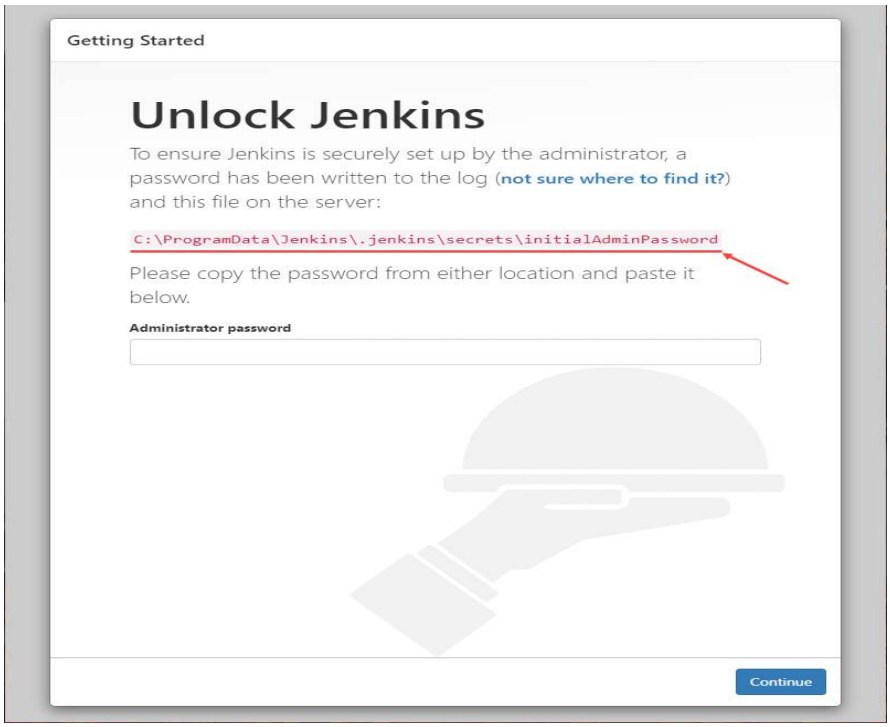
Unblock Jenkins

After completing the installation process, you have to unblock Jenkins before you can customize and start using it.

1. In your web browser, navigate to the port number you selected during the installation using the following address:

http://localhost:[port number]

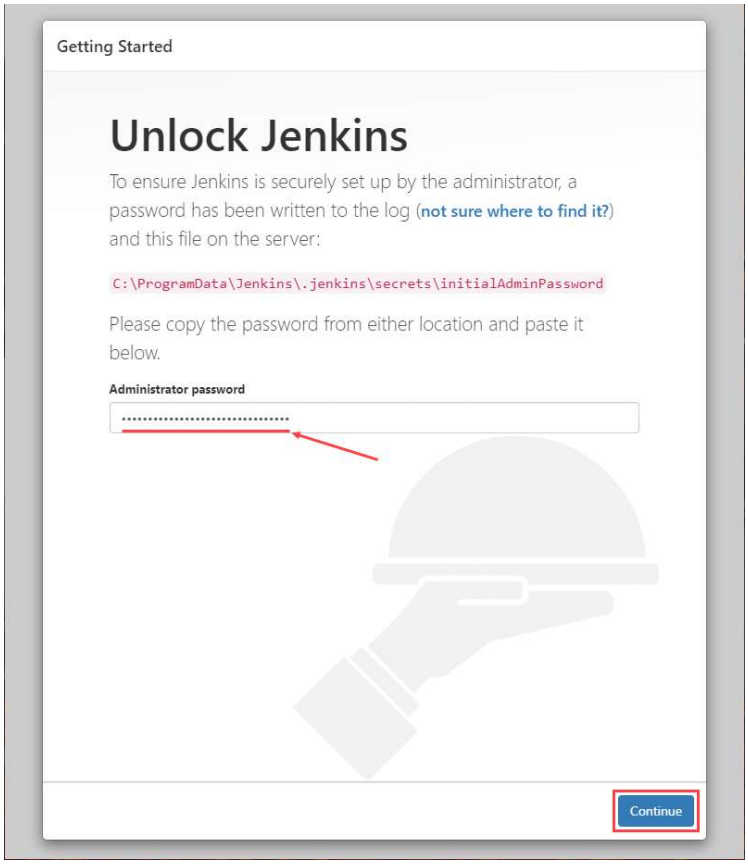
2. Navigate to the location on your system specified by the Unblock Jenkins page.



3. Open the **initialAdminPassword** file using a text editor such as Notepad.

4. Copy the password from the **initialAdminPassword** file.

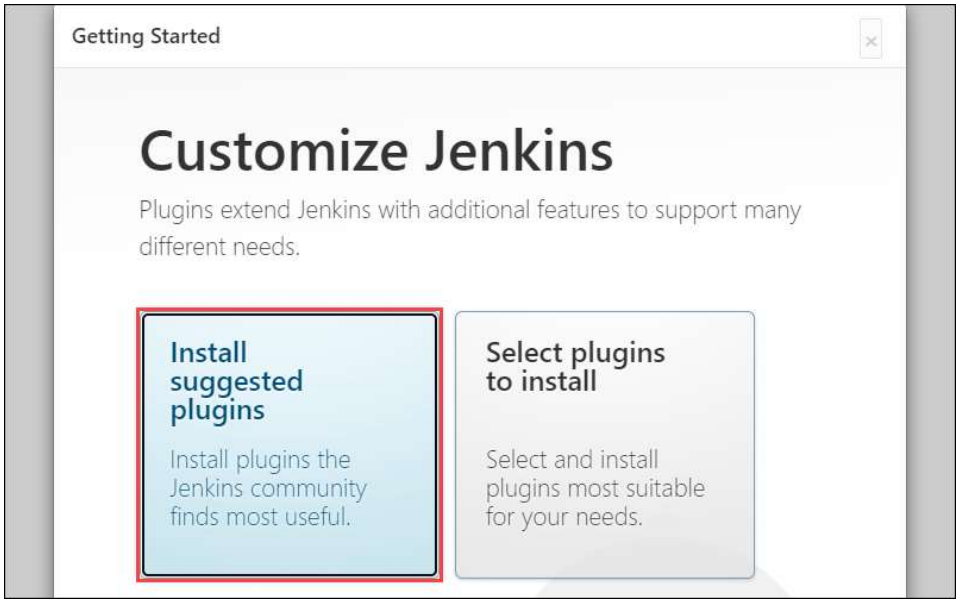
5. Paste the password in the **Administrator password** field on the Unblock Jenkins page and click **Continue** to proceed.



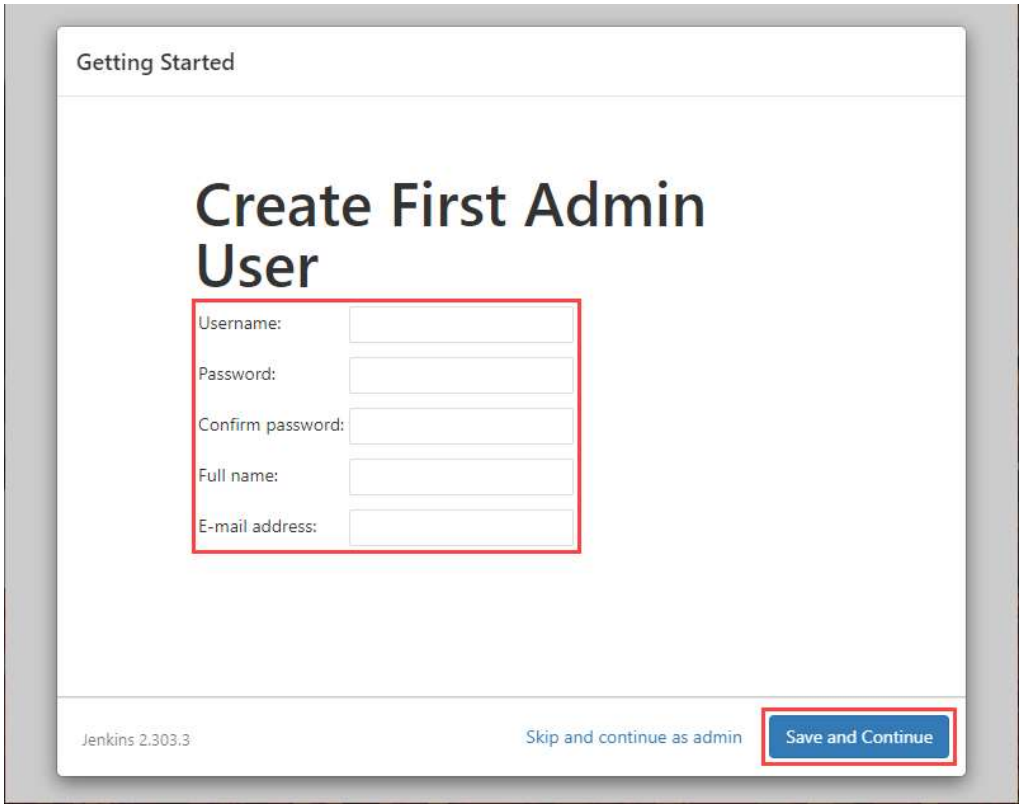
Customize Jenkins

Once you unlock Jenkins, customize and prepare the Jenkins environment.

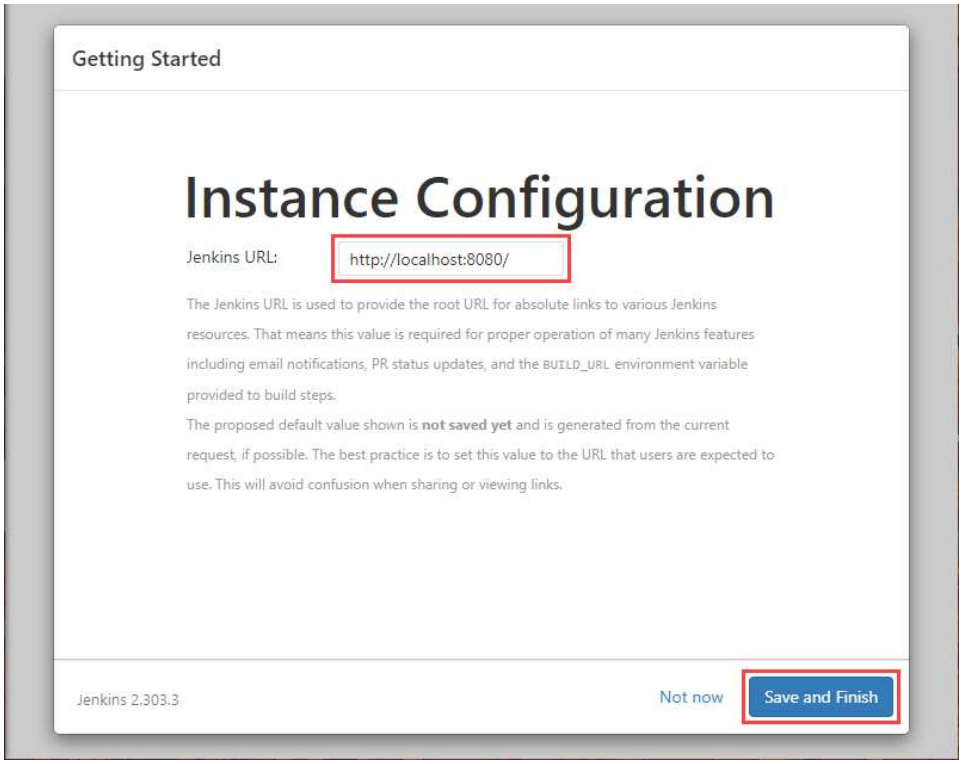
1. Click the **Install suggested plugins** button to have Jenkins automatically install the most frequently used plugins.



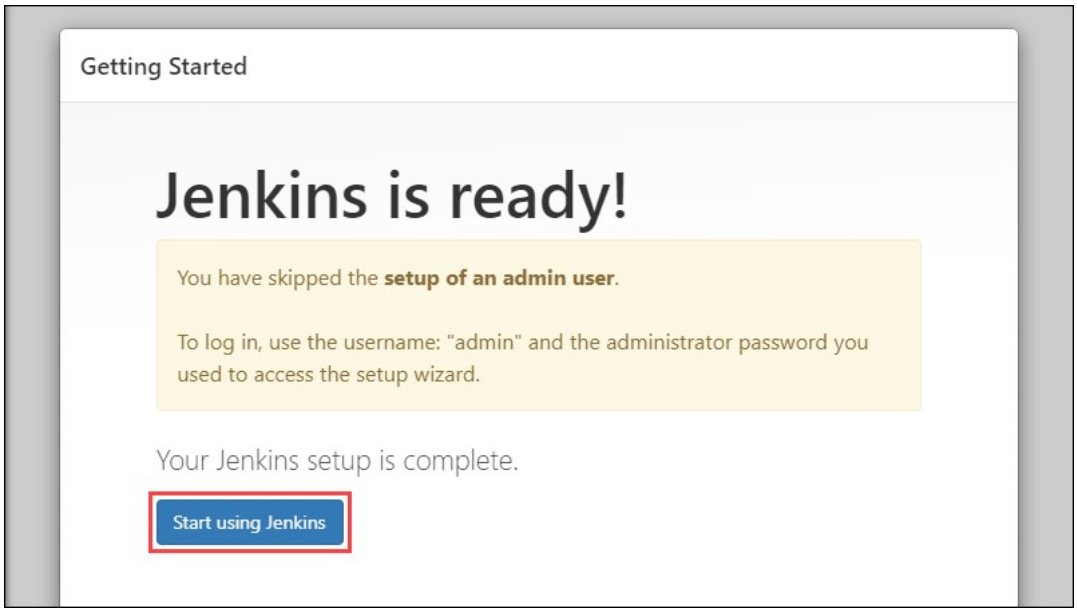
2. After Jenkins finishes installing the plugins, enter the required information on the **Create First Admin User** page. Click **Save and Continue** to proceed.



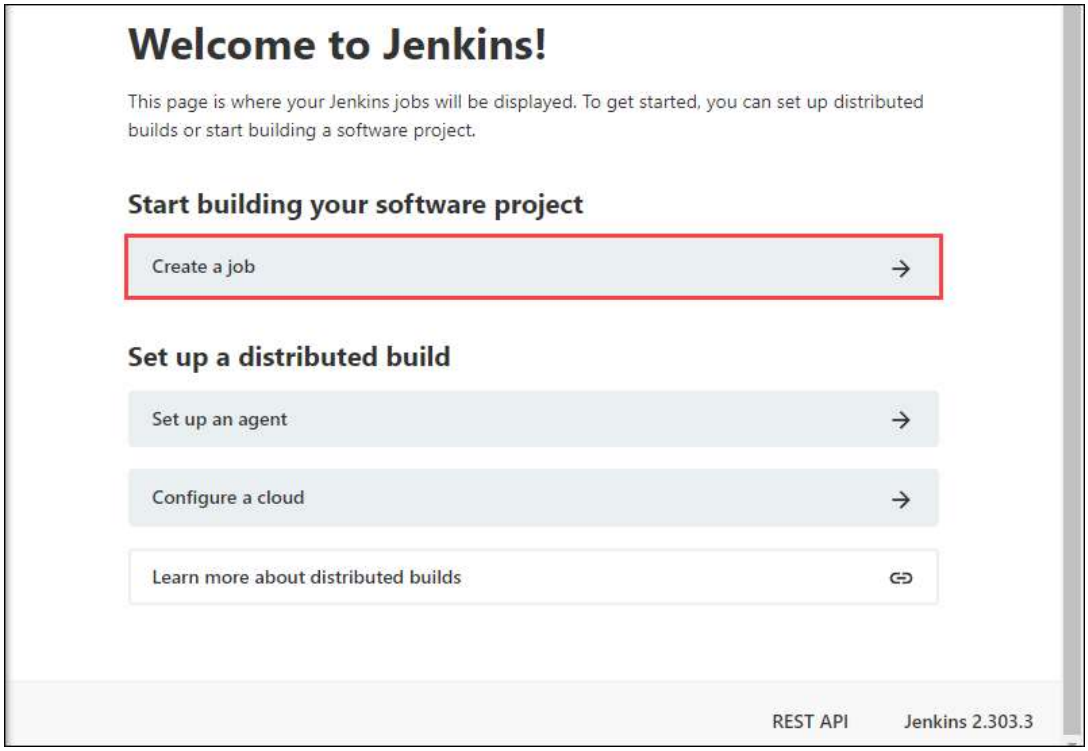
3. On the **Instance Configuration** page, confirm the port number you want Jenkins to use and click **Save and Finish** to finish the initial customization.



4. Click the **Start using Jenkins** button to move to the Jenkins dashboard.



5. Using the Jenkins dashboard, click **Create a job** to build your first Jenkins software project.



Students should build a job and execute in Jenkins environment.

6. **Conclusion and Discussion:**

Students are supposed to write your own conclusion

7. **Viva Questions:**
   17. What is Continuous Integration?
   18. What is CI/CD?
   19. Which tools can be plugged with Jenkins?

8. **References:**

1. Jenkins: The Definitive Guide: Continuous Integration for the Masses, by John Ferguson Smart Published by O'Reilly Media

2. Jenkins 2: Up and Running Authored by Brent Laster Published by O'Reilly Media

Skill Based Lab IV – DevOPs

Experiment No.: 4

To build the pipeline of jobs in Jenkins, create a pipeline script to deploy an application over Server

Experiment No. 4

1. Aim: To build the pipeline of jobs in Jenkins, create a pipeline script to deploy an application
overServer.

2. Objectives: From this experiment, the student will be able

20. To install and build job in Jenkins pipeline for deploying application DevOps environment.

3. Outcomes: The learner will be able to

21. To understand the importance of Jenkins Pipelines for continuous integration and continuous deployment CI/CD and deploy applications on server.

4. Hardware / Software Required: Linux / Windows Operating System, Jenkins

5. Theory:

A pipeline is a set of interconnected tasks that execute in a specific order. Additionally, Jenkins Pipeline is a suite of plugins that help users implement and integrate continuous delivery pipelines into Jenkins.Moreover, using Pipeline, you can create complex or straightforward delivery pipelines as code via the Pipeline domain-specific language(DSL) syntax. Subsequently, the below states represent a continuous delivery Pipeline**: -**

By using Jenkins pipeline continuous Integration, Testing, and Delivery is a seamless process, and one can achieve all this using the Pipeline concept, which enables the project to be production-ready always**.**

Jenkins Pipeline—1. Declarative Pipeline Syntax

The declarative syntax is a new feature that used code for the pipeline. It provides a limited pre-defined structure. Thereby, it offers an easy & simple continuous delivery pipeline. Moreover, it uses a pipeline block

2. Scripted Pipeline Syntax

Unlike declarative syntax, the *scripted pipeline syntax* is the old traditional way to write the *Jenkinsfile* on Jenkins web UI. Moreover, it strictly follows the groovy syntax and helps to develop a complex pipeline as code.

- ***Pipeline -*** A pipeline is a set of instructions that includes the processes of continuous delivery. For example, creating an application, testing it, and deploying the same. Moreover, it is a critical

  Pipeline { }

  element in declarative pipeline syntax, which is a collection of all stages in a Jenkinsfile. We declare different stages and steps in this block.

- ***Node -*** A node is a key element in scripted pipeline syntax. Moreover, it acts as a machine in

  Node {}

  Jenkins that executes the Pipeline.

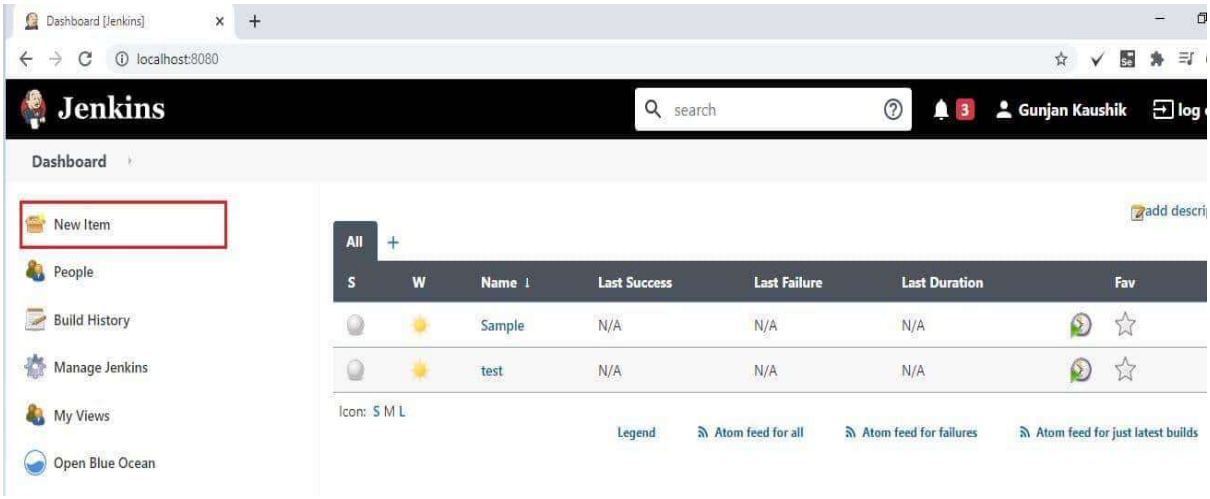
- ***Stage*** - *A* stage consists of a set of processes that the Pipeline executes. Additionally, the tasks are divided in each stage, implying that there can be multiple stages within a Pipeline.

```
pipeline{
    agent any
stages{
        stage('Build'){
          ........
         }
        stage('Test'){
          ........
         }
        stage('Deploy'){
          ........
         }
        stage('Monitor'){
          ........
         }
```

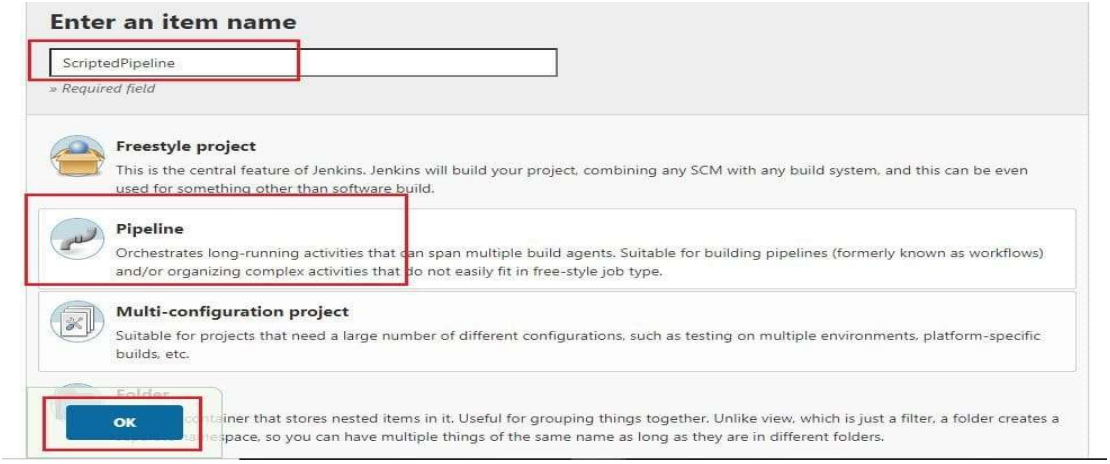
Steps of execution:

How to create Jenkins Scripted Pipeline?

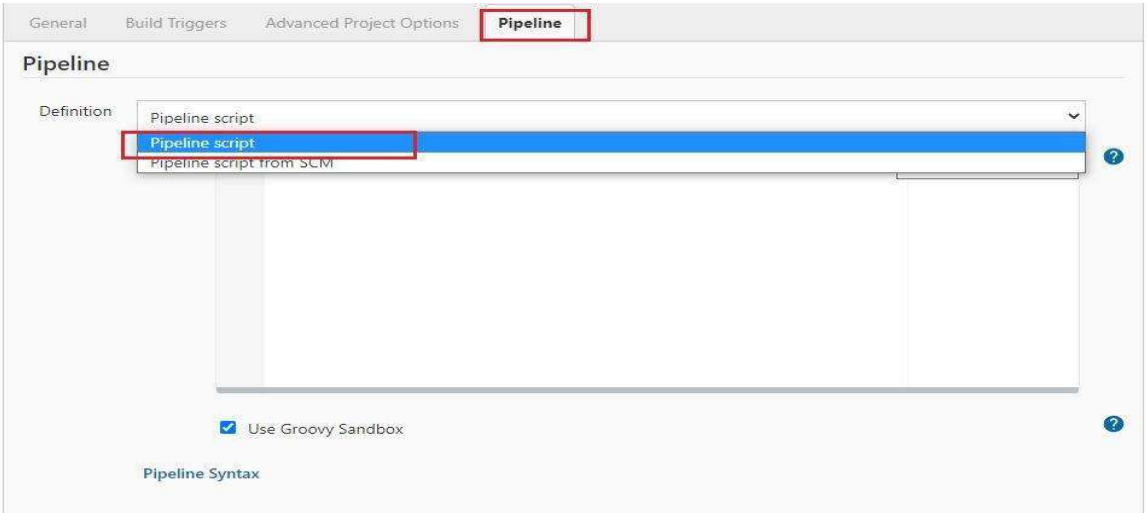
1. Firstly, from the *Jenkins dashboard*, click on *New Item* on the left panel.



2. Secondly, enter the *name* for your pipeline, select *Pipeline* from the list. After that, click *OK*.

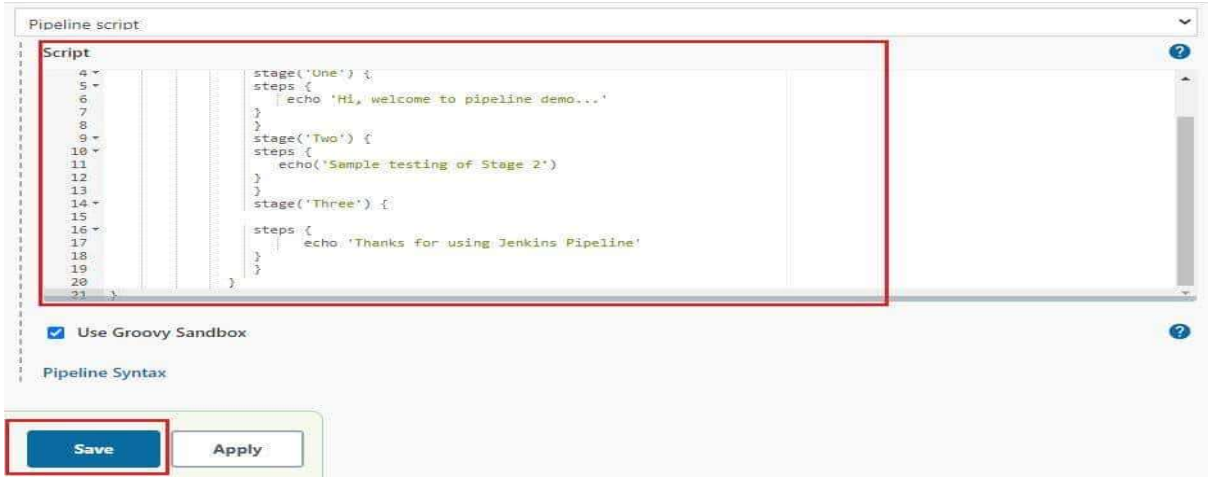


3. After that, go to the ***Pipeline*** tab, and from the ***Definition***, the dropdown selects the ***Pipeline*** script.



4. The next step is to write your *pipeline code in the web UI* provided by Jenkins. Let us see a sample pipeline example as available in Jenkins-

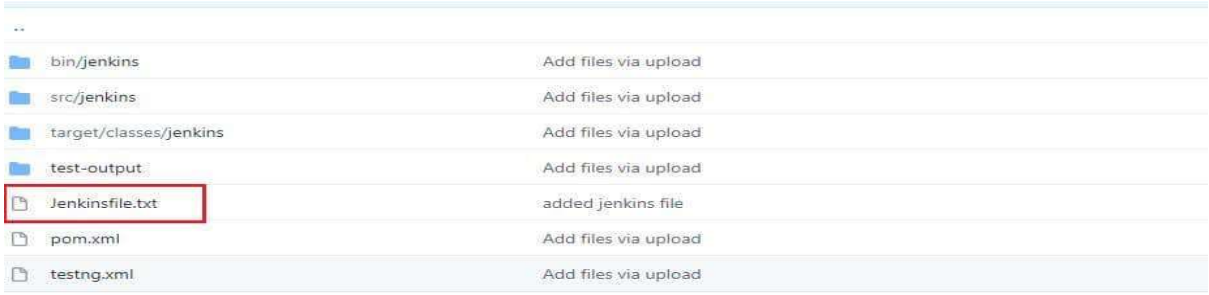
5. You need to copy and paste the same in UI.



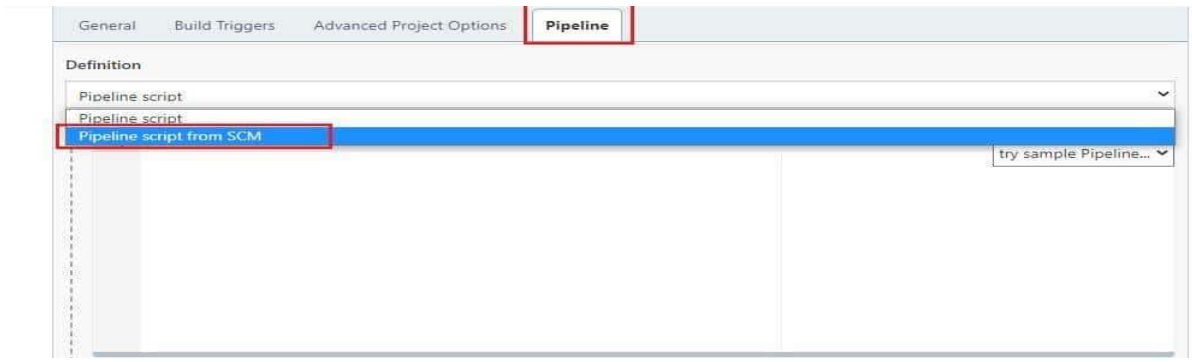
5. After that, click on ***Save***. Conclusively, this finishes the process.

How to create a Declarative Jenkins Pipeline?

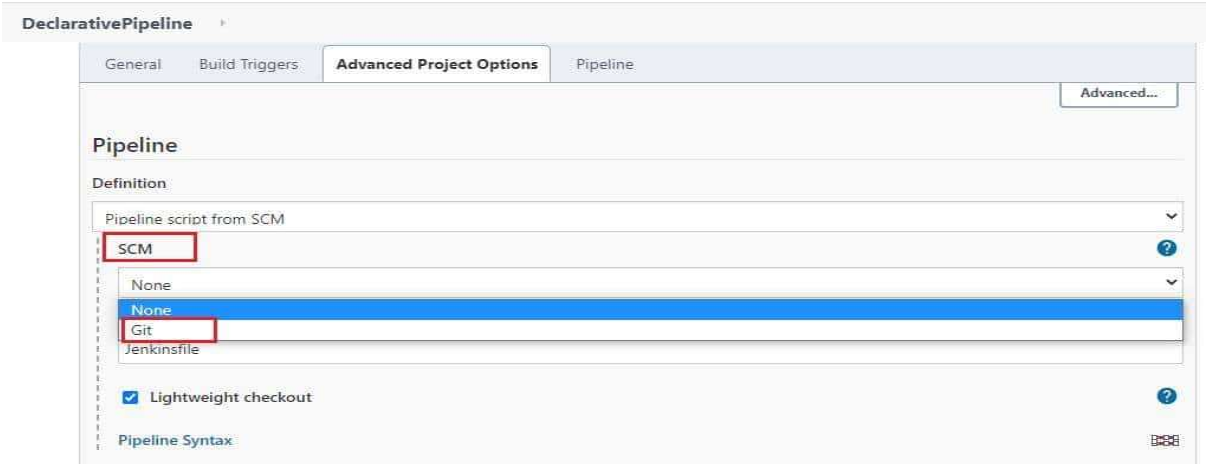
1. To create a declarative pipeline, you need to have a ***Jenkinsfile*** in place. Since I will be using the project from my Github account, I have already placed the ***Jenkinsfile*** in my project.



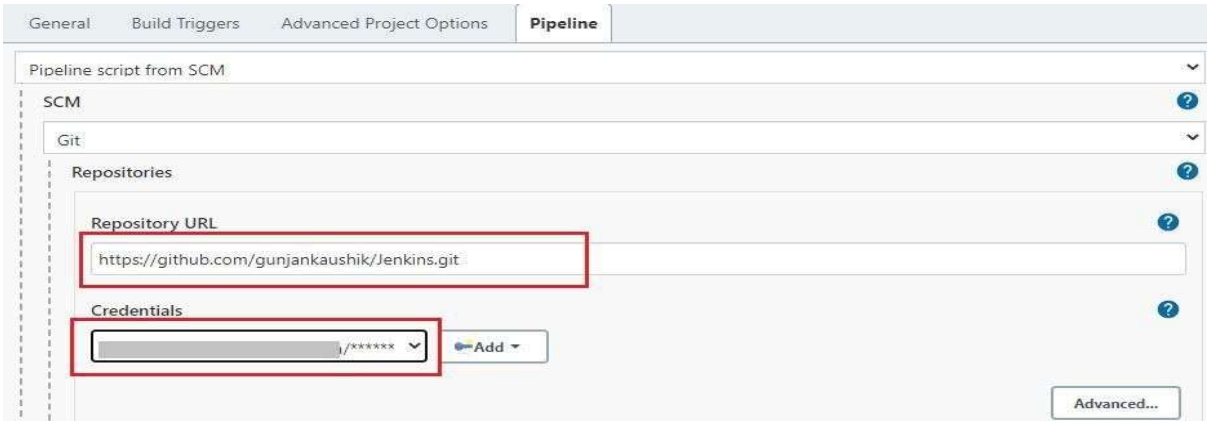
2. For creating a *Declarative Pipeline*, you may follow *step#1 and Step#2* from the scripted pipeline creation steps stated above and then follow the below steps-
   Go to the Pipeline tab, and from the ***Definition***, the dropdown selects the ***Pipeline script from SCM***



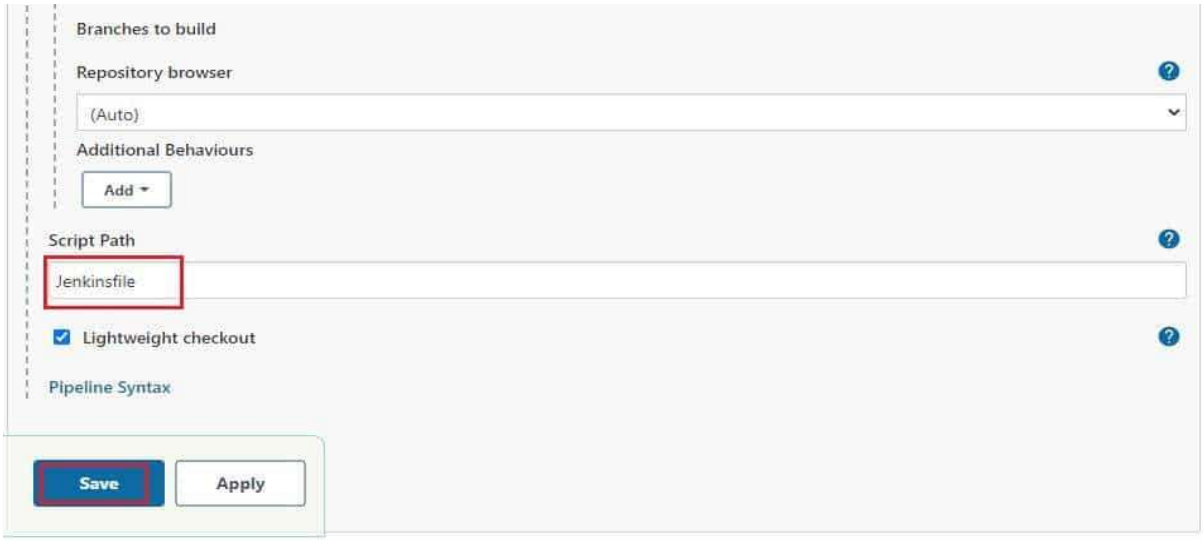
3. Go to the Pipeline tab, and from the ***Definition***, the dropdown selects the ***Pipeline script from SCM***



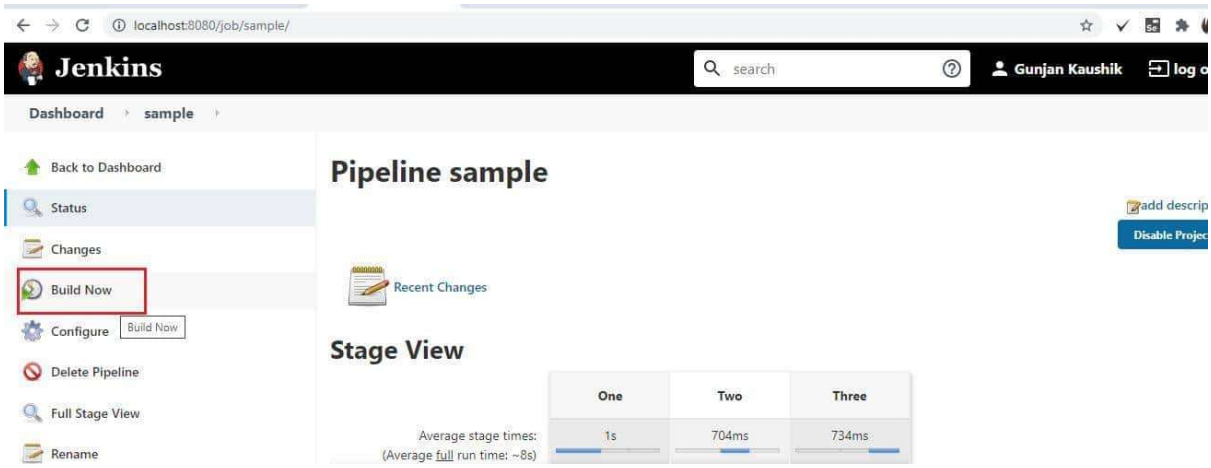
4. Now, you will get an option to input your *Repository URL* and *credentials*.



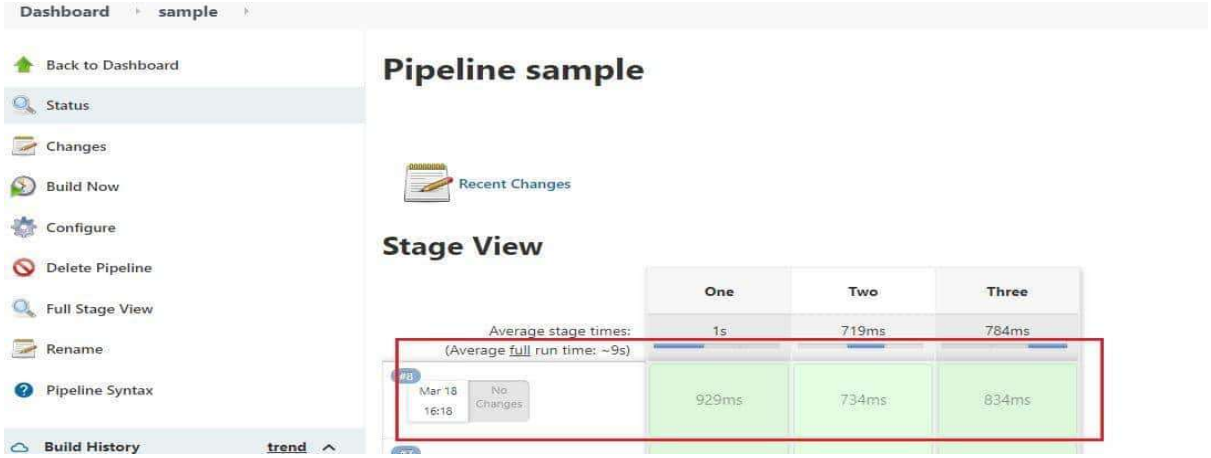
5. 4. Next, you may set the ***branch*** or let it be blank for any branch. In the script path, you need to write the ***Jenkinsfile*** name that exists in your repository. Click on ***Save***, and there you go, your declarative pipeline is ready for use.



6. Now that you are all set with your pipelines, you can execute the same from your *Jenkins UI*. All you need to do is select your pipeline and click on ***Build Now*** link on the left panel.



7. Once you run the pipeline, you will see the results displayed on the stage view as shown below-



Note: Students should build a pipeline job and demonstrate CI/CD in Jenkins environment.

6. Conclusion and Discussion:

Students will be writing their own conclusion based on experiment performed.

7. Viva Questions:

- What is pipeline?
- What is declarative pipeline?
- What is scripted pipeline?

8. References:

- Jon Loeliger and Matthew McCullough, Version Control with Git, O'Reilly Media, Second Edition, 2012.
- Dennis Hutten, Git: Learn Version Control with Git A Step-By-Step Ultimate Beginners Guide.
- Scott Chacon and Ben Straub, Pro Git_ Everything You Need to Know About Git, apress, Second Edition.

Skill Based Lab IV – DevOPs

Experiment No.: 5

To install Tomcat server on windows and run Jenkins over Tomcat server.

Experiment No.5

Aim: To install Tomcat server on windows and run Jenkins over Tomcat server.

2. Objectives: From this experiment, the student will be able

- To understand DevOps practices which aims to simplify Software Development Life.
- To understand the working of web server

9. Outcomes: The learner will be able to

- To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.
- To obtain complete knowledge of the "continuous integration" to effectively track and manage changes.

10. Hardware / Software Required : Linux / Windows Operating System

11. Theory:

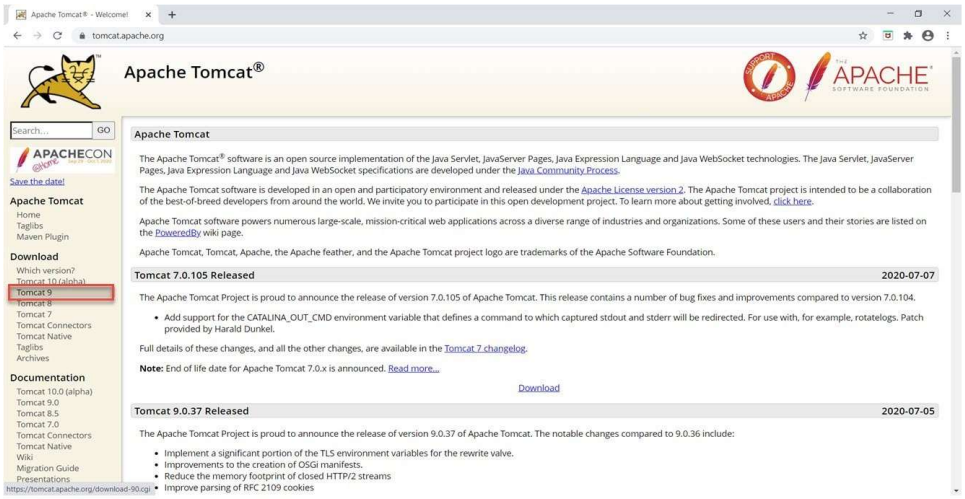
Tomcat server

It is an open-source Java servlet container that implements many Java Enterprise Specs such as the Websites API, Java-Server Pages and last but not least, the Java Servlet. The complete name of Tomcat is "Apache Tomcat" it was developed in an open, participatory environment and released in 1998 for the very first time. It began as the reference implementation for the very first Java-Server Pages and the Java Servlet API. However, it no longer works as the reference implementation for both of these technologies, but it is considered as the first choice among the users even after that. It is still one of the most widely used java-sever due to several capabilities such as good extensibility, proven core engine, and well-test and durable. Here we used the term "servlet" many times, so what is java servlet; it is a kind of software that enables the webserver to handle the dynamic(java-based) content using the Http protocols.

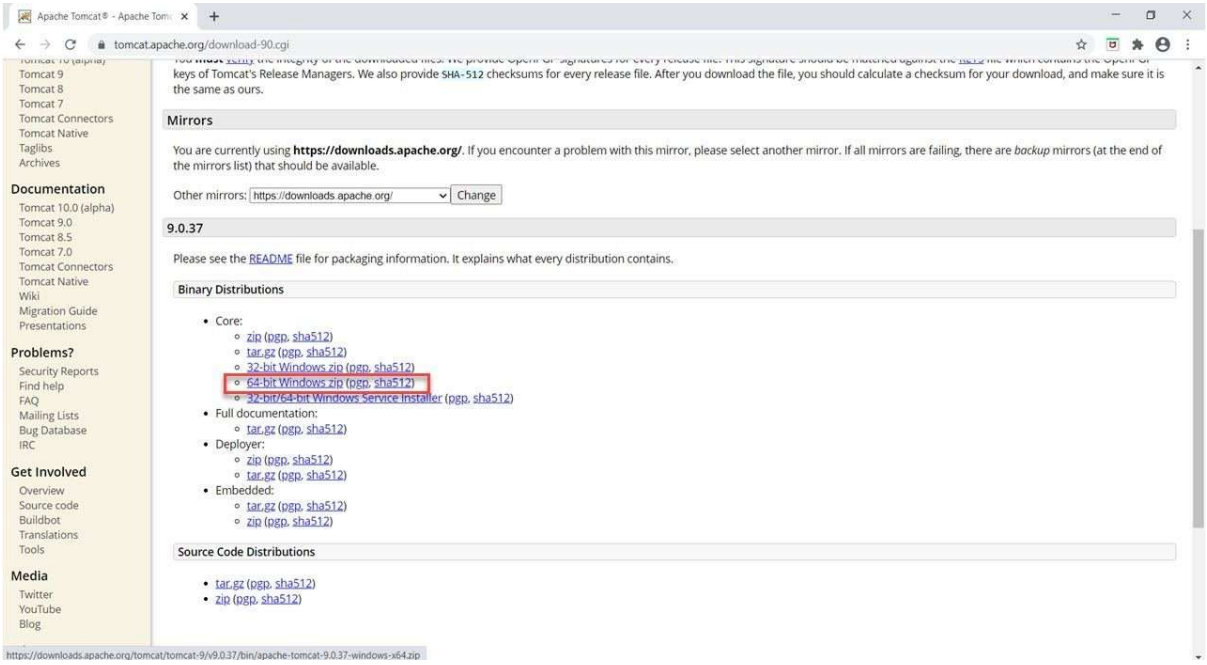
Step 1: **Environment Configuration for tomcat7 setup**

- Example- tomcat7 setup
- Install Tomcat7 in CENTOS server at /apps/tomcat/tomcat7 directory and this is e.g CATALINA_HOME
- Now you need to find your CATALINA_HOME before continuing with the next steps.

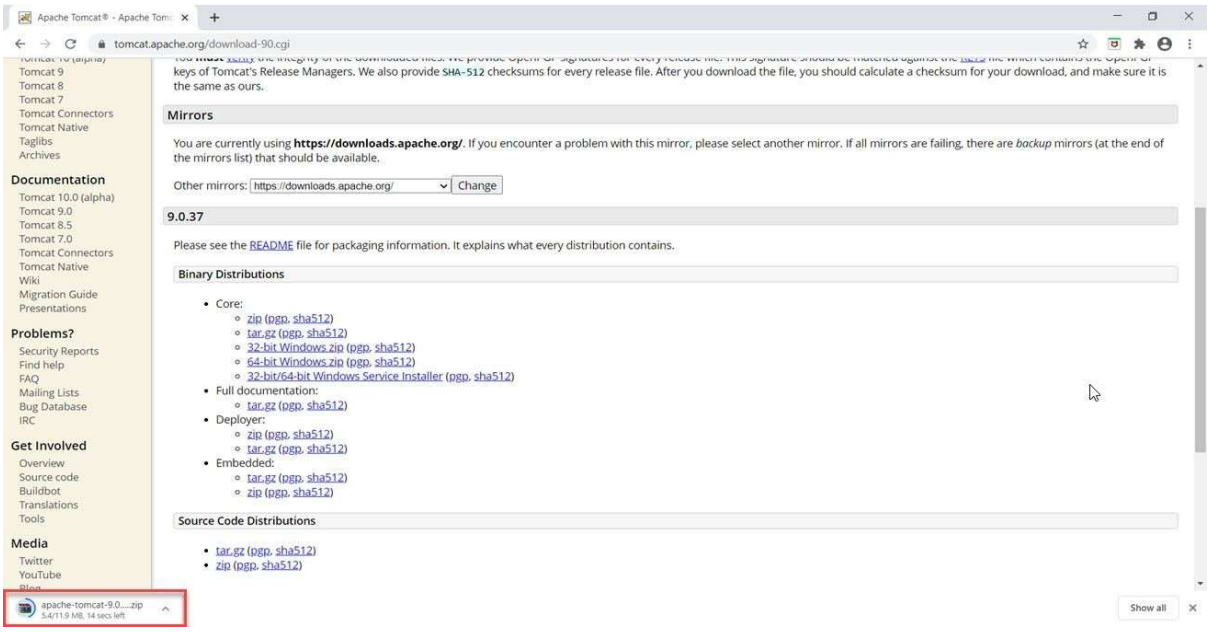
Step 2 : Go to the official ***Apache Tomcat website*** and choose the latest stable version, as we chose ***Tomcat 9***.



Step 3 : Choose the appropriate binary distribution according to your machine configuration, as I am going to choose *64 bit Windows zip*.

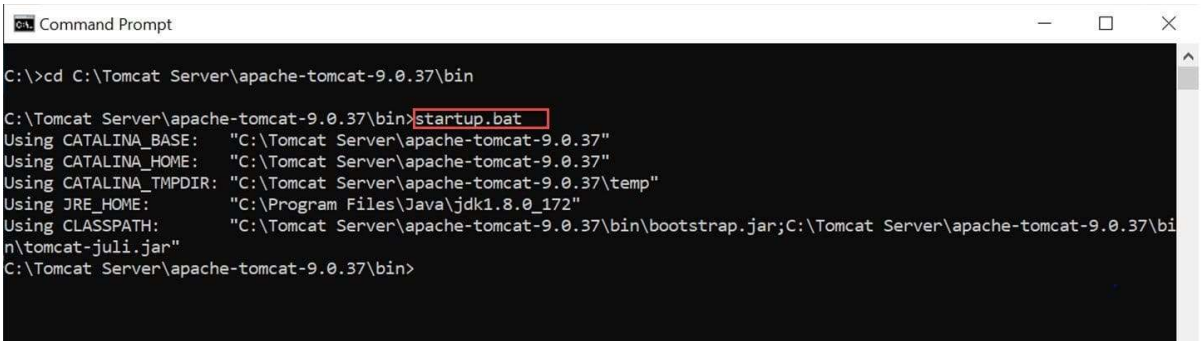


Step 4 : Download the zip file and store it in any of your working directories. Also, unzip this file after storing it in the working directory.

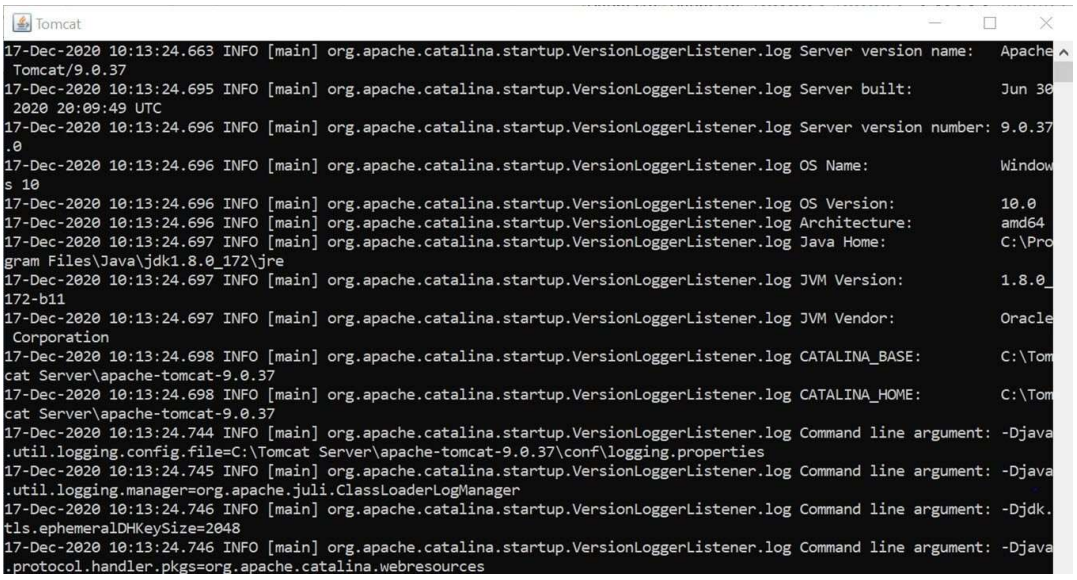


Step 5 : Now, open the command prompt, go to bin folder path, and type the below command:

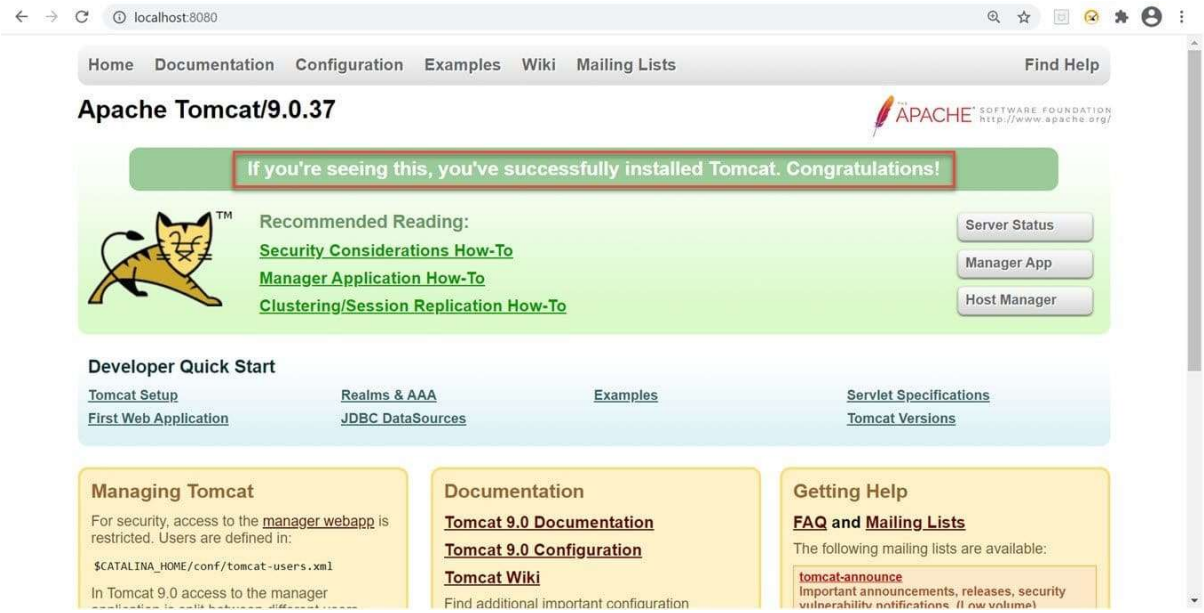
```
startup.bat
```



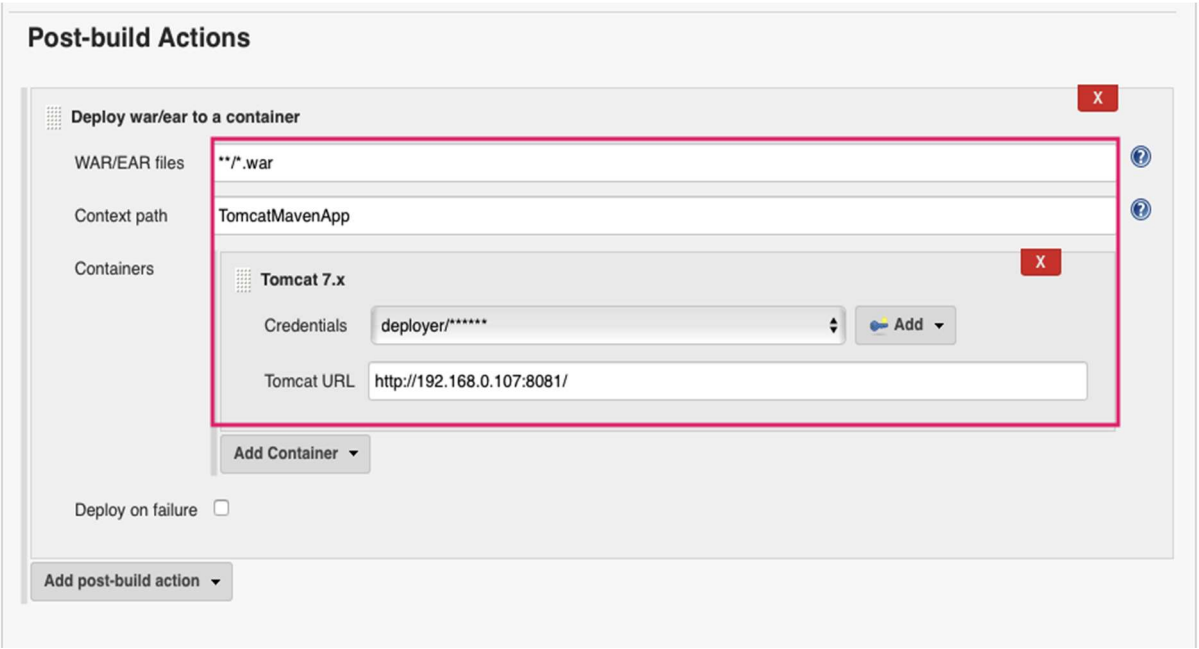
Step 6 : As soon as the command will execute, a new window "***Tomcat***" will open in which some processing will happen.



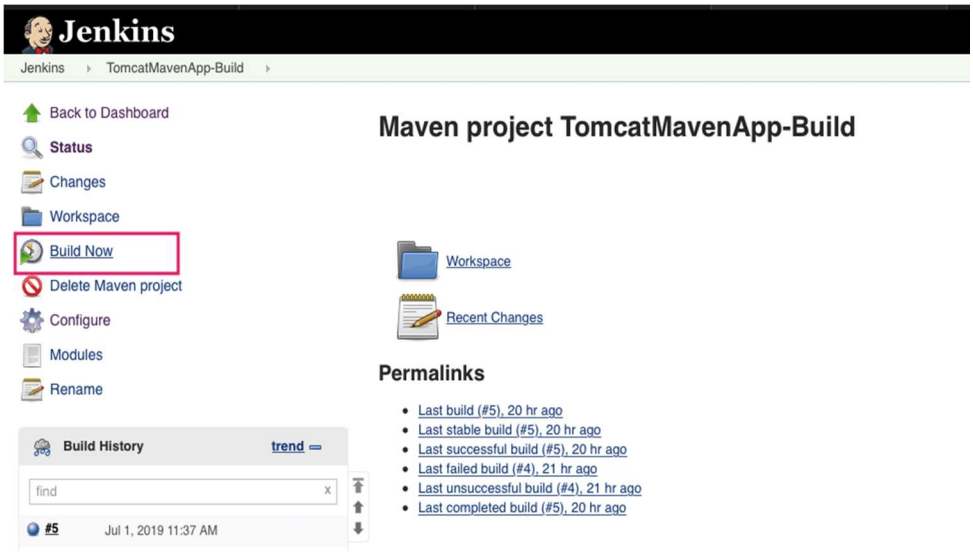
Step 7 : Once processing will be done, just for validation whether the tomcat server is installed on our system, open the browser, and hit the URL ***http://localhost:8080/***. After hitting the *URL*, we will see below the window as shown below, if there is a successful installation of tomcat on our machine.



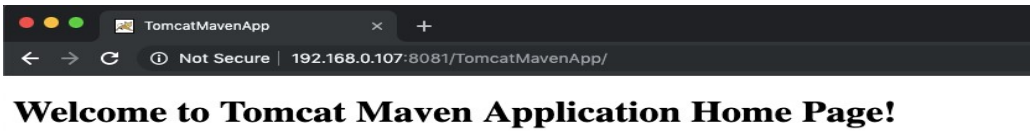
Step 8 :Configure the Post-build Action and Specify the Tomcat Server Details



Step 9 :Build Jenkins Job



Step 10 :Testing the Application



12. Conclusion and Discussion:

Students are supposed to write your own conclusion

13. Viva Questions:

- What is Tomcat server?
- What is a web server?
- What are the other web servers available for deployment?

14. References:

- Tomcat: The Definitive Guide 2e Paperback – 6 November 2007by Jason Brittain (Author), Ian F Darwin (Author)O'Reilly publication.
- https://www.jenkins.io/doc/book/

Skill Based Lab IV – DevOPs

Experiment No.: 7

To understand Docker Architecture, install docker desktop and execute docker commands to manage and interact with containers.

Experiment No. 7

1. Aim: To understand Docker Architecture, install docker and execute docker commands
to manage and interact with containers.

2. Objectives: From this experiment, the student will be able

- To understand Docker architecture and installation of Docker and execute Docker commands

3. Outcomes: The learner will be able to

- To understand the importance of Docker and use of Docker architecture in Devops environment.

4. Hardware / Software Required: Linux / Windows Operating System, Docker

5. Theory:

1. **Docker &need of Docker**– Docker is a containerization platform that packages your application and all its dependencies together in the form of a docker container to ensure that your application works seamlessly in any environment.
2. **Container?** – Docker Container is a standardized unit which can be created on the fly to deploy a particular application or environment. It could be an Ubuntu container, CentOs container, etc. to full-fill the requirement from an operating system point of view. Also, it could be an application-oriented container like CakePHP container or a Tomcat-Ubuntu container etc.
3. **Docker Image**

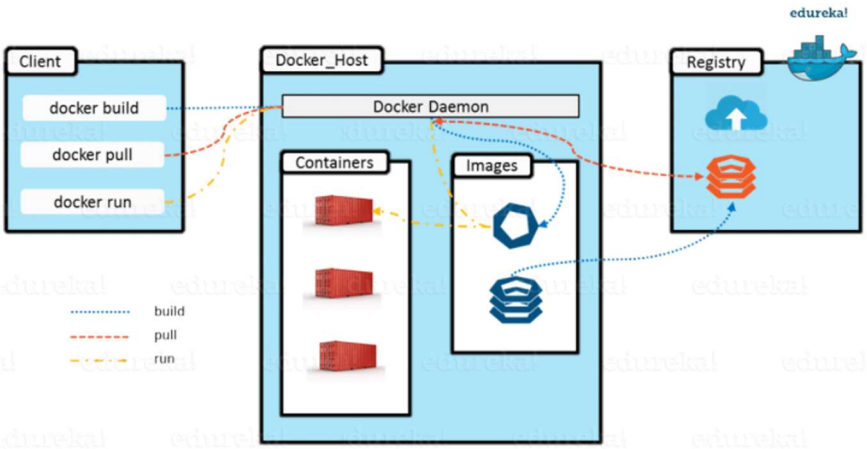
   Docker Image can be compared to a template which is used to create Docker Containers. They are the building blocks of a Docker Container. These Docker Images are created using the build command.
4. **Docker Container**

   Docker Containers are the ready applications created from Docker Images. Or you can say they are running instances of the Images and they hold the entire package needed to run the application. This happens to be the ultimate utility of the technology.
5. **Docker Registry**

   Finally, Docker Registry is where the Docker Images are stored. The Registry can be either a user's local repository or a public repository like a Docker Hub allowing multiple users to collaborate in building an application. Even with multiple teams within the same organization can exchange or share containers by uploading them to the Docker Hub, which is a cloud repository similar to GitHub.

6. **Docker Architecture?**

   Docker Architecture includes a Docker client – used to trigger Docker commands, a Docker Host – running the Docker Daemon and a Docker Registry – storing Docker Images. The Docker Daemon running within Docker Host is responsible for the images and containers.



7. **Install Docker Desktop on Windows**

1. Double-click **Docker Desktop Installer.exe** to run the installer.

   If you haven't already downloaded the installer (Docker Desktop Installer.exe), you can get it from **Docker Hub**. It typically downloads to your Downloads folder, or you can run it from the recent downloads bar at the bottom of your web browser.
2. When prompted, ensure the **Use WSL 2 instead of Hyper-V** option on the Configuration page is selected or not depending on your choice of backend.

   If your system only supports one of the two options, you will not be able to select which backend to use.

3. Follow the instructions on the installation wizard to authorize the installer and proceed with the install.

4. When the installation is successful, click **Close** to complete the installation process.

5. If your admin account is different to your user account, you must add the user to the **docker-users** group. Run **Computer Management** as an **administrator** and navigate to **Local Users and Groups** > **Groups** > **docker-users**. Right-click to add the user to the group. Log out and log back in for the changes to take effect.

8. Docker hub account creation

Your Docker ID becomes your user namespace for hosted Docker services, and becomes your username on the Docker forums. To create a new Docker ID:

1. Go to the Docker Hub signup page.

2. Enter a username that will become your Docker ID.

   Your Docker ID must be between 4 and 30 characters long, and can only contain numbers and lowercase letters. **Once you create your Docker ID you can't reuse it in the future if you deactivate this account**.

3. Enter a unique, valid email address.

4. Enter a password that's at least 9 characters.

5. Complete the Captcha verification and then select **Sign up**.

Docker sends a verification email to the address you provided.

6. Verify your email address to complete the registration process.

9. Docker command- following are the Docker commands to be executed

1. **docker –version**

   This command is used to get the currently installed version of docker.

2. **docker login**

   This command is used to login to the docker hub repository

3. **docker run**

   Usage: docker run -it -d <image name>

   This command is used to create a container from an image

4. **docker build**

   Usage: docker build <path to docker file>

   This command is used to build an image from a specified docker file

5. **docker images**

   This command lists all the locally stored docker images

6. **docker push**

   Usage: docker push <username/image name>

   This command is used to push an image to the docker hub repository

7. **docker pull**

   Usage: docker pull <image name>

   This command is used to pull images from the **docker repository**(hub.docker.com)

8. **docker ps**

   This command is used to list the running containers

9. **docker stop**

   Usage: docker stop <container id>

   This command stops a running container

Note: Students should install docker desktop, download any docker official image and push updated official docker image on docker hub.

10. Conclusion and Discussion:

Students will be writing their own conclusion based on experiment performed.

11. Viva Questions:

- What is a Container?
- Why Learn Docker?
  What are docker images?

12. References:

- Jon Loeliger and Matthew McCullough, Version Control with Git, O'Reilly Media, Second Edition, 2012.
- Dennis Hutten, Git: Learn Version Control with Git A Step-By-Step Ultimate Beginners Guide.
- https://docs.docker.com/desktop/install/windows-install/
- Scott Chacon and Ben Straub, Pro Git_ Everything You Need to Know About Git, apress, Second Edition.

Skill Based Lab IV – DevOPs

Experiment No.: 8

To learn Dockerfile instructions, build an image for sample web application on Docker Engine.

Experiment No. 8

1. Aim: To learn Dockerfile instructions, build an image for sample web application on Docker Engine.

2. Objectives: From this experiment, the student will be able

- To understand Docker file and build an image for web applications Docker Engine

3. Outcomes: The learner will be able to

- To understand the importance of Dockerfile building an image.

4. Hardware / Software Required: Linux / Windows Operating System, Docker, VsCode

5. Theory: Dockerfile

A Dockerfile is a script/text configuration file that contains collections of commands and instructions that will be automatically executed in sequence in the docker environment for building a new docker image.This file is written in a popular, human-readable Markup Language called YAML.A Dockerfile is a text configuration file written using a special syntaxIt describes step-by-step instructions of all the commands you need to run to assemble a Docker Image.

- **Create a Dockerfile**

Creating a Dockerfile is as easy as creating a new file named "Dockerfile" with your text editor of choice and defining some instructions. The name of the file does not really matter. Dockerfile is the default name but you can use any filename that you want (and even have multiple dockerfiles in the same folder)

- **docker build**

The docker build command processes this file generating a Docker Image in your Local Image Cache, which you can then start-up using the docker run command, or push to a permanent Image Repository**.**The docker build command processes this file generating a Docker Image in your Local Image Cache, which you can then start-up using the docker run command, or push to a permanent Image Repository.

❖ **Steps for the building an image for sample web application on Docker Engine. Create a container image in Visual Studio Code**

1. First, create a folder named PHP Hello Docker and open it in Visual Studio Code.
2. Open this folder in Visual Studio Code and create a file named hello.php or can take any simple html file with hello.html .

```
<html>
<?phpecho"Hello world from php container ">
</html>
```

3. Next, create a file called Dockerfile (without any extension) and write the below code into the file. Now with Docker file let construct an image.

```
FROM php
```

```
COPY ./ ./

EXPOSE 3000

CMD ["php","-S","0.0.0.0:3000"]
```

This is a very basic Dockerfile. Dockerfiles describe how to build your Docker image.

4. Use the docker build command on terminal

   docker build . -t dockerhubaccountname/foldername //building image

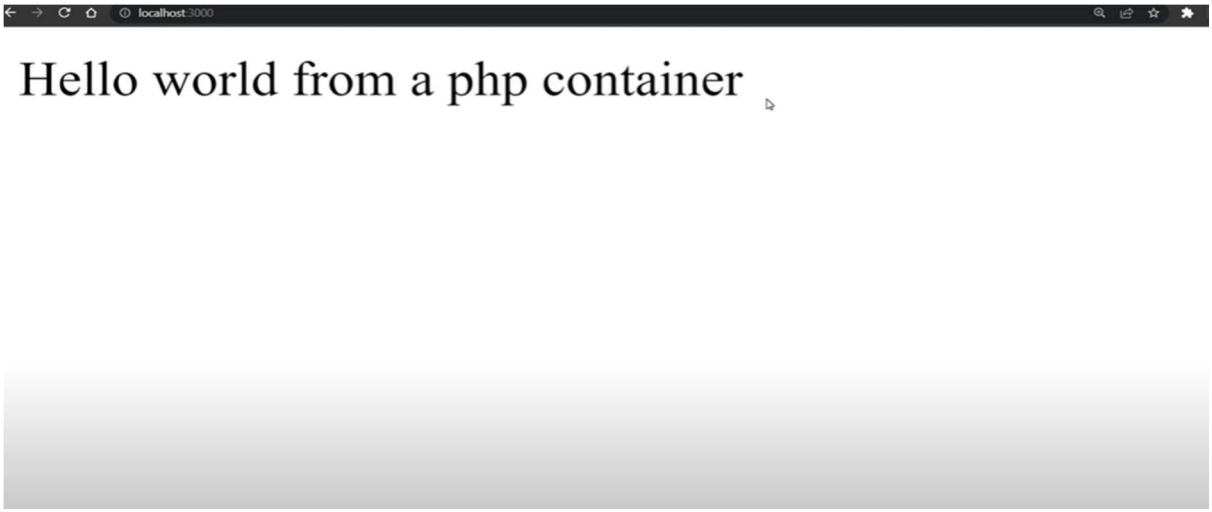
   e.g. docker build . -t bharnesmita45/myfirstPHP_image
   where myfirstPHP_image is the name of folder created in local machine

5. Now run the container using following command.

   docker run --name=php -p=3000:3000 dockerhubaccountname/foldername

   e. g. docker run --name=php -p=3000:3000 bharnesmita45/myfirstphp_image

6. After that new container is running on to browser window.
   Type localhost**:3000** on browser and we will get following output window.



Note: Students should build web application, crate image through Dockerfile and push image on Docker hub.

6. Conclusion and Discussion:

Students will be writing their own conclusion based on experiment performed.

7. Viva Questions:

- What is a dockerfile?
- What is Docker hub?

- How do you create a docker container from an image?

8. References:

- Jon Loeliger and Matthew McCullough, Version Control with Git, O'Reilly Media, Second Edition, 2012.
- Dennis Hutten, Git: Learn Version Control with Git A Step-By-Step Ultimate Beginners Guide.
- Scott Chacon and Ben Straub, Pro Git_ Everything You Need to Know About Git, apress, Second Edition.

Skill Based Lab IV – DevOPs

Experiment No.: 9

To install and configure software configuration management and provisioning tool using ansible.

Experiment No.9

Aim: To install and configure software configuration management and provisioning tool using ansible.

2. Objectives: From this experiment, the student will be able

- To understand DevOps practices for configuration management.
- To understand the working of Ansible configuration management tool.

15. Outcomes: The learner will be able to

- To understand the fundamentalsof DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.
- To understand the fundamentals and importance of configuration management.

16. Hardware / Software Required : Linux / Windows Operating System

17. Theory:

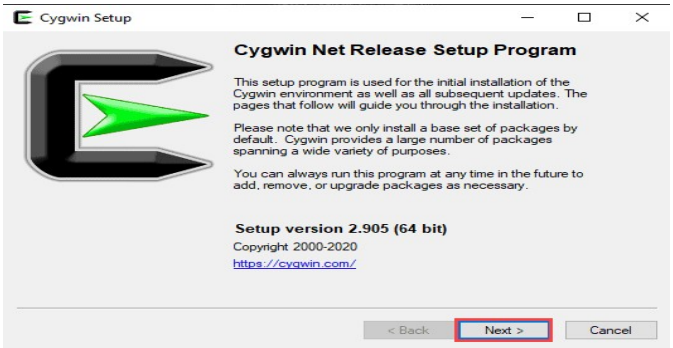
Configuration Management

The process of standardizing and administering resource configurations and entire IT infrastructure in an automated way is Configuration Management. It is the concept where you put your server infrastructure as code.It helps to systematically manage, organize, and control the changes in the documents, codes and other entities during the SDLC.It aims to control costs and work effort involved in making changes to the software system

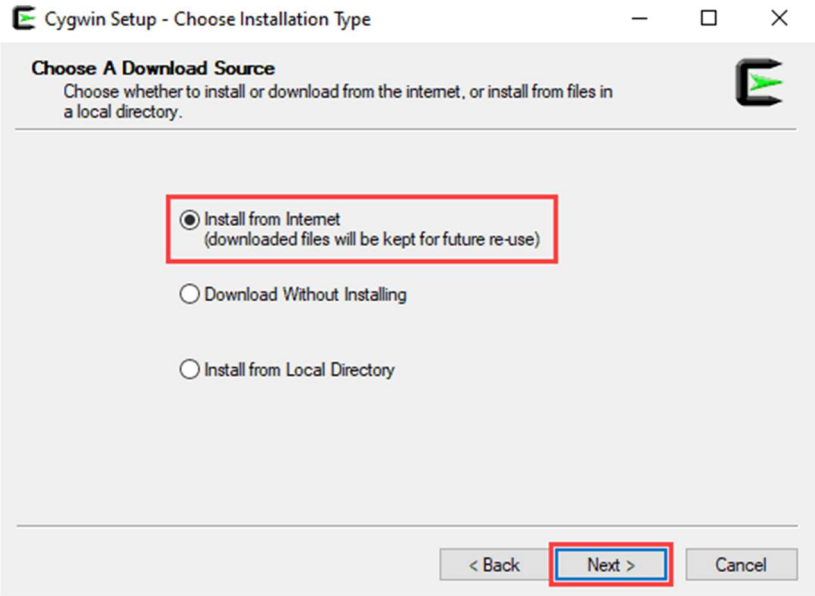
To install Ansible on Windows using Cygwin, follow these steps:

1. Download the Cygwin installation file. This file is compatible with both the 32-bit and 64-bit versions of Windows 10. It automatically installs the right version for your system.

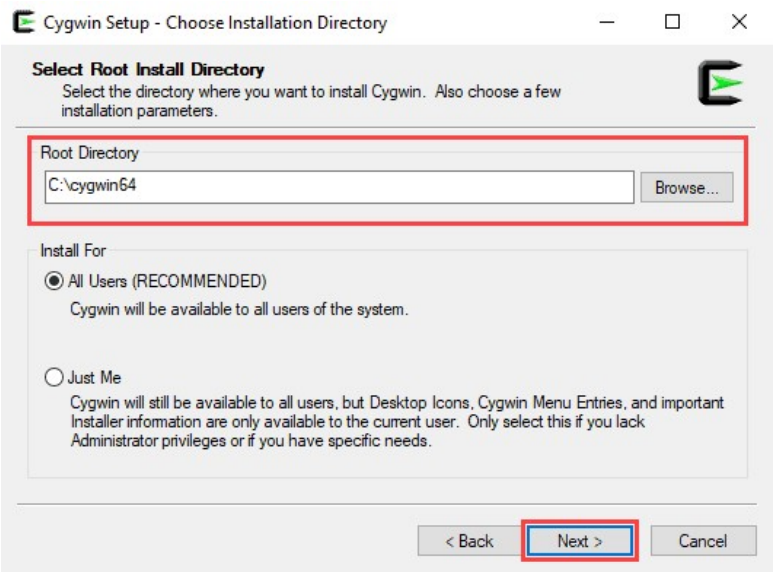
2. Run the Cygwin installation file. On the starting screen of the installation wizard, click **Next** to continue.



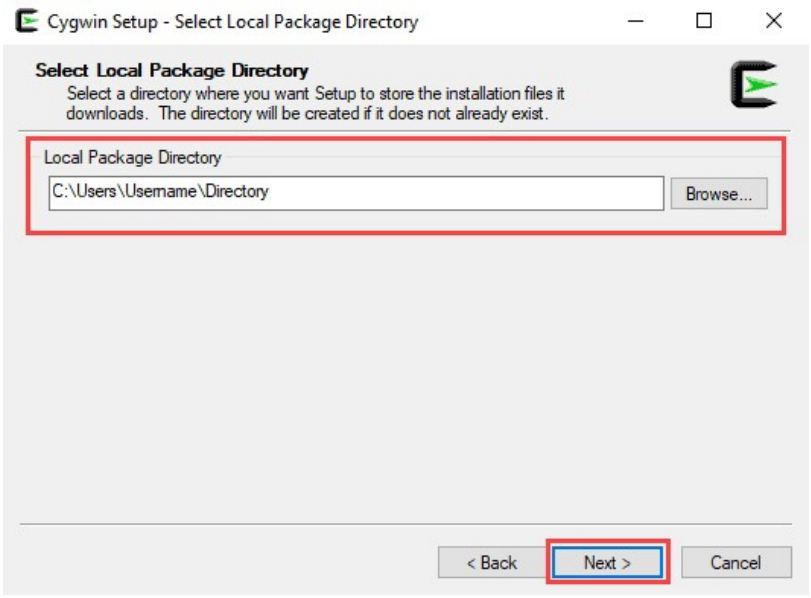
3. Select **Install from Internet** as the download source and click **Next**.



4. In the **Root Directory** field, specify where you want the application installed, then click **Next**.

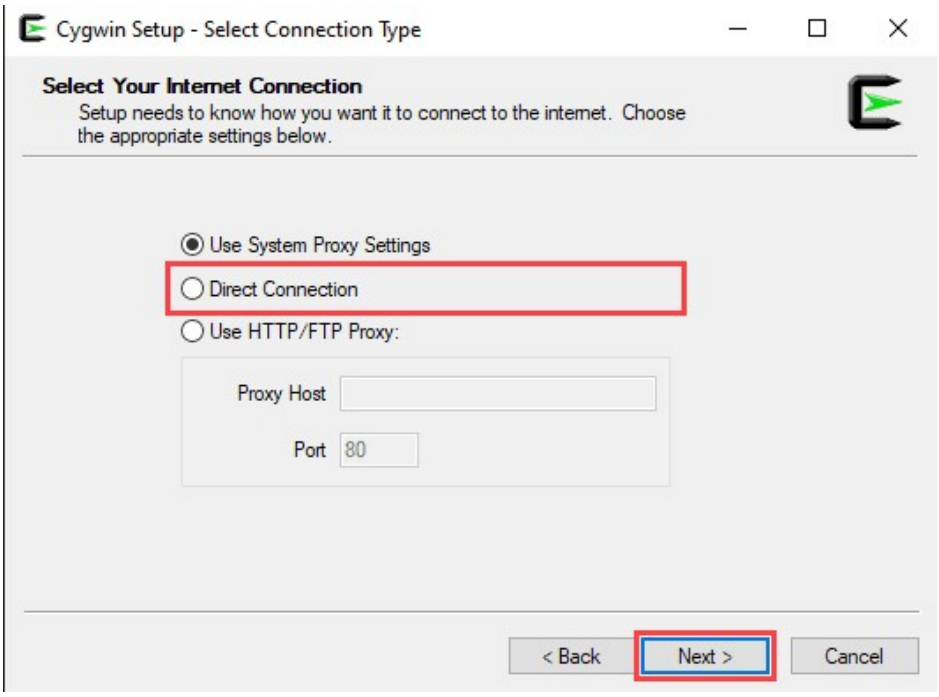


5. In the **Local Package Directory** field, select where you want to install your Cygwin packages, then click **Next**.

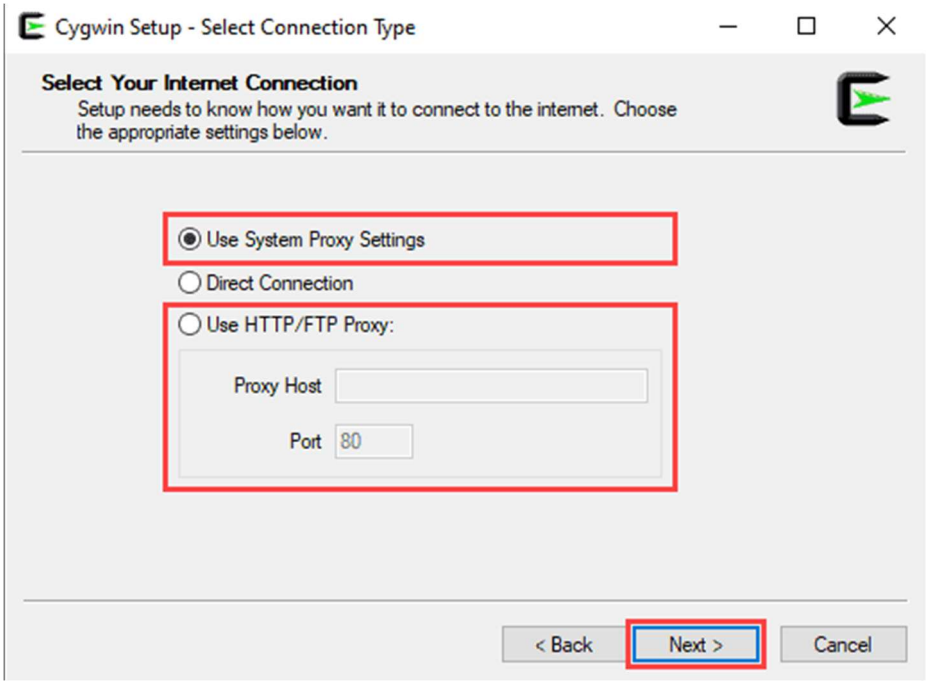


6. Choose the appropriate Internet connection option.

If you aren't using a proxy, select **Direct Connection**.

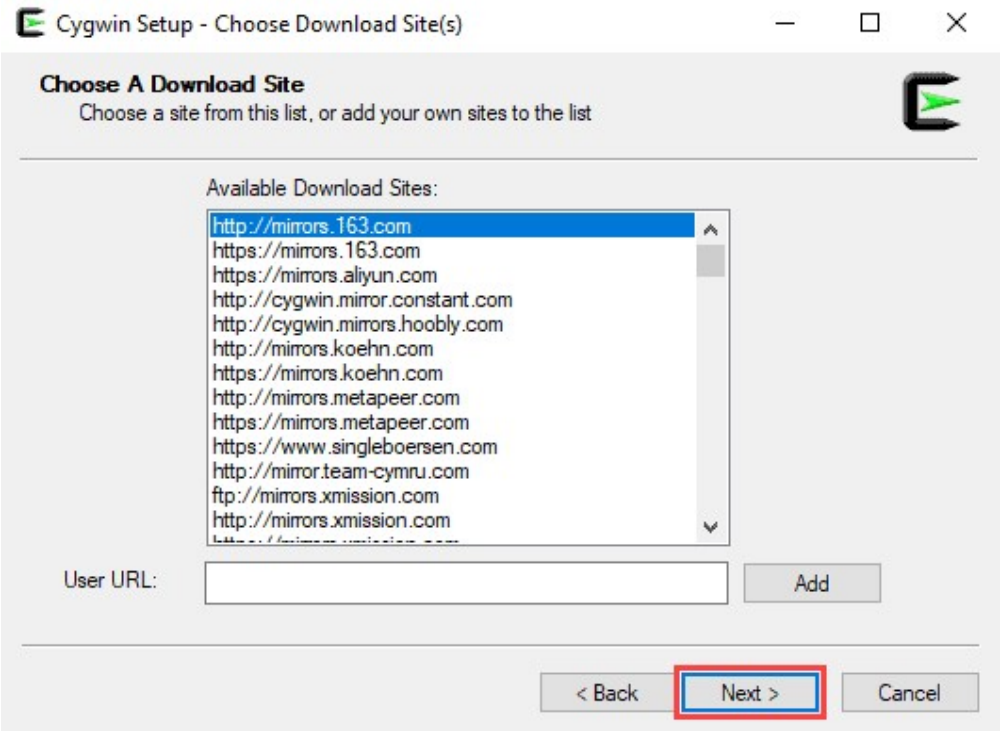


If you are using a proxy, select **Use System Proxy Settings** or enter the proxy settings manually with the **Use HTTP/FTP Proxy.**



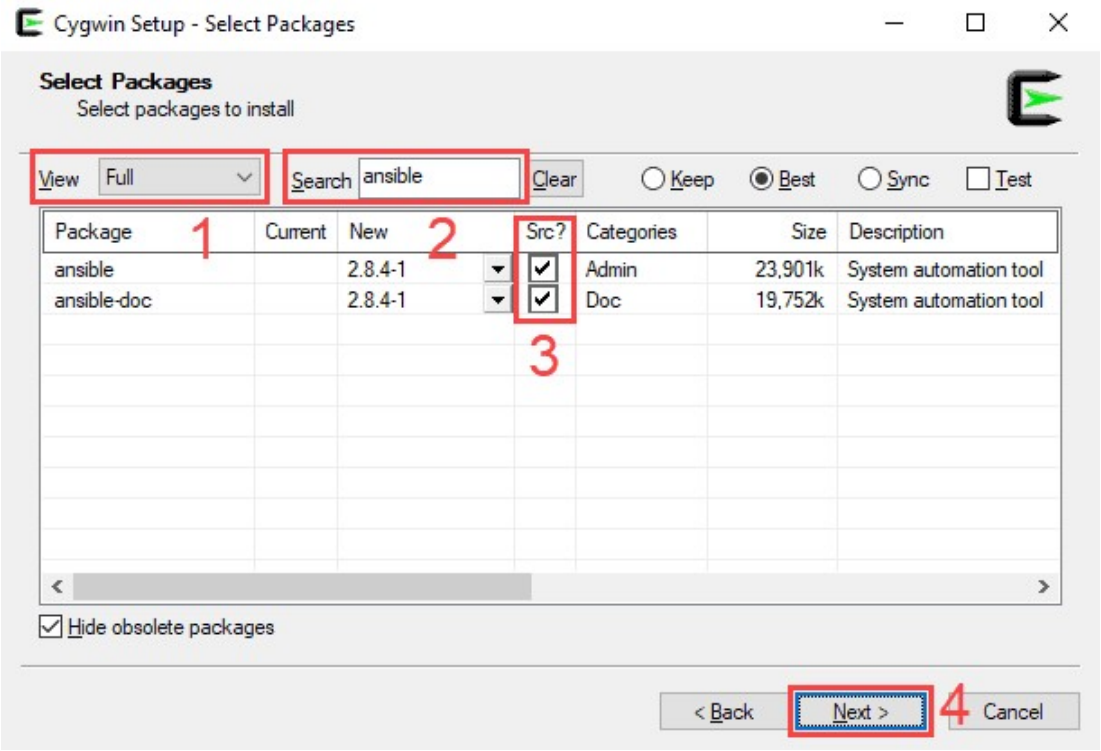
Click **Next** to continue.

7. Choose one of the available mirrors to download the installation files, then click **Next**.

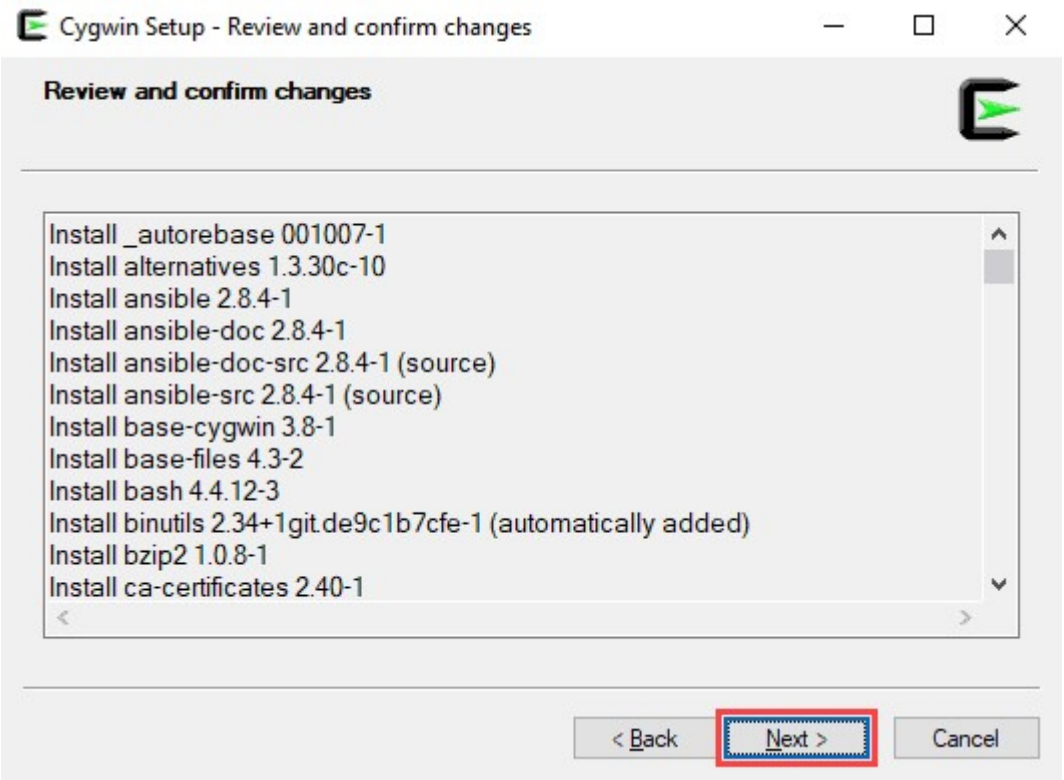


8. On the **Select Packages** screen, change the **View** option to **Full** and type ‘ansible’ in the search bar.

Select both **Ansible** and **Ansible Doc** by checking the boxes under **Src?** and click **Next** to continue.

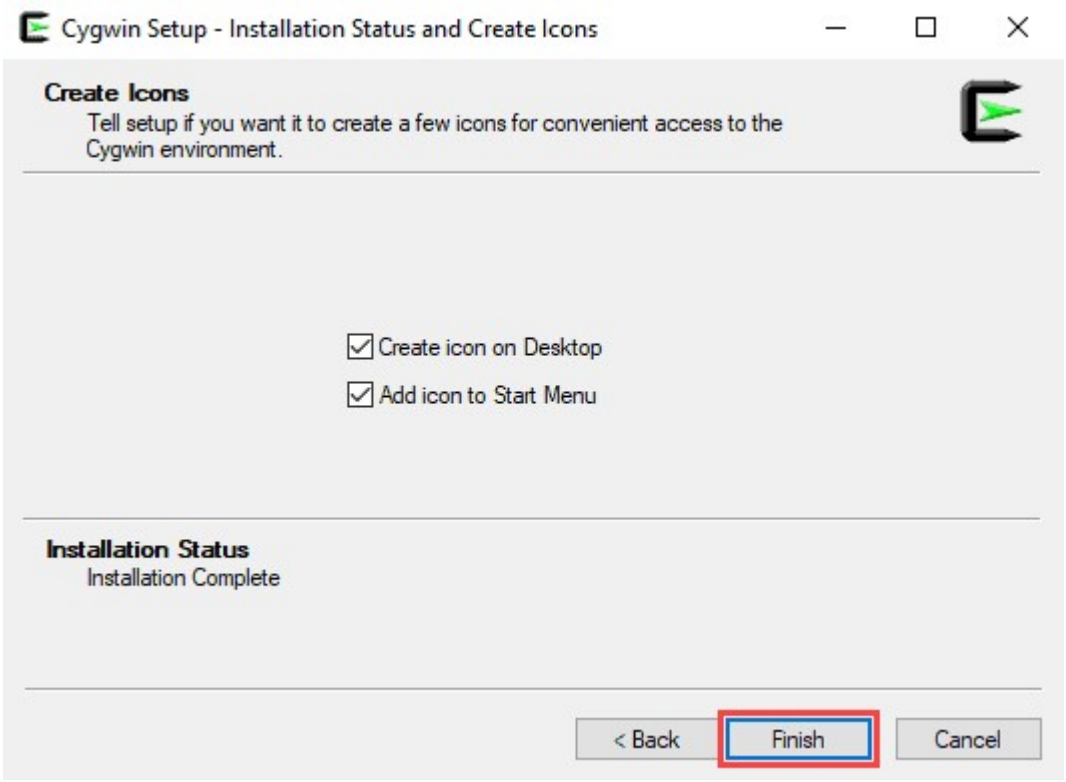


9. This screen lets you review the installation settings. To confirm and begin the install process, click on **Next**.



10. The install wizard will download and install all the selected packages, including Ansible.

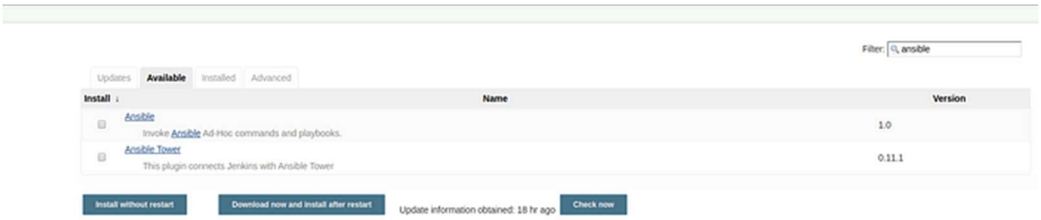
11. Once the installation is complete, select whether you want to add a Cygwin desktop and Start Menu icon, then click on **Finish** to close the wizard.



Ansible Playbook with Jenkins Pipeline

For Jenkins Pipeline we need to install the **Ansible** plugin.

Go to **Manage Jenkins > Manage Plugins >Available >** search **Ansible.**



If you are already installed **Ansible Plugin** on your Jenkins It will display in the **Installed** section.

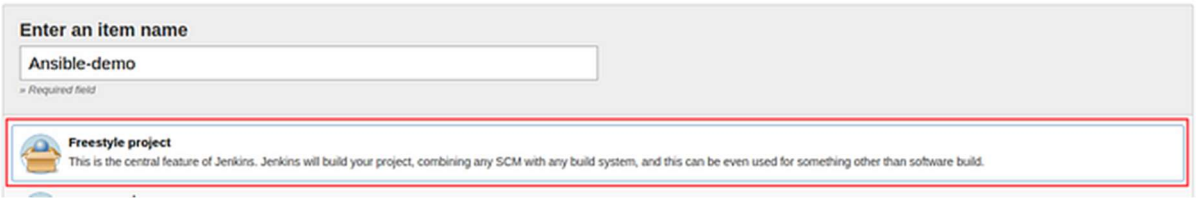
Now we can see the **Invoke Ansible Playbook** option in the **Build Environment** section but we need to configure Ansible path for Jenkins.

Now let's configure Ansible on our Jenkins.

Go to **Manage Jenkins > Global Tool Configuration >** It will display **Ansible on** the list.

Now let's Create New project to execute Ansible playbook.

Step 2) Goto Jenkins Home > New Item > Create New Freestyle Project.

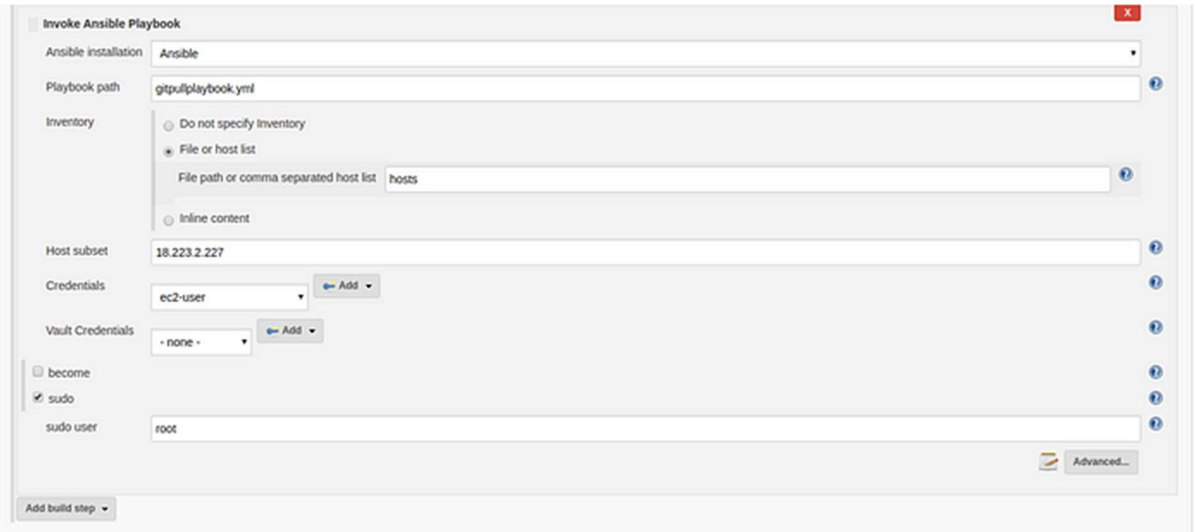


We create a new freestyle project now lets configure our project.

Goto source code management section and add your code repository here you are free to use any of the source code management platforms like Github, Gitlab and also Bit bucket.

Now let's configure ansible plugin.

Goto **Build** section and select **Invoke Ansible Playbook.** Once you select that option it will display ansible-playbook configuration option like below.



Let see all option provided by Ansible Plugin.

- **Ansible Installation**: It will display all Ansible installation list that we are configured in **Global Tool Configuration > Ansible.**
- **Playbook path**: We have to provide the **Absolute** or **Relative** path of our Ansible-playbook file.
- **Inventory**: It gives 3 options to configure our host file.

Do not specify Inventory*: We have to give our host list in the subdomain option.*

File or host list*: add a path of our HOST file.*

Inline content*: It means we are set our host list on the starting of our ansible-playbook file.*

- **Host subset**: If we want to filter a specific set of the host from the provided Host file.
- **Credentials**: Configure added host's username and password.
- **Vault Credentials**: IF you are using ansible vault then configure that credentials.
- **become**: If you want to set a specific user for ansible-playbook execution select and add the name of that user.

- **sudo**: If you need to use a sudo user(root) then select this and add the name of that user.

Ansible Ad hoc commands

Ad hoc commands are commands which can be run independently, to perform immediate functions.
These commands are of one time usage.
Syntax of a typical ad hoc command:
Ansible [-m module_name] [-a args] [options]
Some of the ad hoc commands and their usage is listed below:

Commands	Usage
$ Ansible def -a "/sbin/reboot" -f 10	Run reboot for all your company servers in a group, 'def', in 10 parallel forks
$ Ansible abc -a "/sbin/reboot" -f 12 -u username	To change from the default current user account, you will have to pass the username in Ad-hoc commands
$ Ansible def -m copy -a "src = /etc/yum.confdest = /tmp/yum.conf"	Transferring a file to many servers/machines using SCP (Secure Copy Protocol)
$ Ansible def -m file -a "dest = /path/user1/new mode = 777 owner = user1 group = user1 state = directory"	Creating new directory
$ Ansible def -m file -a "dest = /path/user1/new state = absent"	Deleting whole directory and files
$ Ansible def -m yum -a "name = demo-tomcat-1 state = present"	Checks if yum package is installed, but does not update it
$ Ansible def -m yum -a "name = demo-tomcat-1 state = absent"	Check the package is not installed
$ Ansible def -m yum -a "name = demo-tomcat-1 state = latest"	Checks the latest version of package is installed
$ Ansible all -m setup	Finds information of all your facts

18. Conclusion and Discussion:
Students are supposed to write your own conclusion

19. Viva Questions:
- What is configuration management?
- Why we need configuration management?
- What are the other tools for configuration management?

20. References:

- "Ansible for DevOps" **Server and Configuration Management for Humans by** Jeff Geerling Midwestern Mac publication, LLC, 05-Aug-2020

- “Ansible: Up and Running”, 3rd Editionby Bas Meijer, Lorin Hochstein, René Moser. Released July 2022. Publisher(s): O'Reilly Media, Inc.