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.

Conclusion:
Version control enables efficient collaboration, ensures code safety, and maintains project history. Git simplifies code management, branching, and merging, making it essential for modern software development and DevOps workflows.

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. Creating a GitHub Account

Step 1:
Go to GitHub.com and sign up for a free account.
Choose a username, enter your email and password, and complete registration.

Step 2:
Create a new repository.
Click the + icon → **New Repository**, name it (e.g., *Demo*), and click **Create Repository**.

Step 3:
Create a file and initialize Git.
Open Terminal and type:

```
mkdir Demo
cd Demo
echo "# Demo" >> README.md
cat README.md
git init
git add README.md
```

This creates a folder, adds a README file, and starts Git tracking.

Step 4:
Make your first commit:

```
git commit -m "first commit"
```

A commit records your current version.
Each commit message helps identify what was changed.

---

b. Synchronizing Repositories

Connect local repo to GitHub:

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

Push code to GitHub:

```
git push -u origin main
```

To sync updates:

```
git fetch          → fetch remote changes
git fetch --prune  → remove deleted refs
git pull           → fetch + merge updates
```

---

c. Tagging Known Commits

```
git tag                 → list all tags
git tag [name]          → tag current commit
git tag -a [name]       → create annotated tag
git tag -d [name]       → delete tag
```

---

d. Reverting Changes

```
git reset [--hard] [ref] → revert to a specific state
git revert [commit sha]  → undo a commit by creating a new one
```

Conclusion:

GitHub allows developers to host, manage, and share code efficiently.
It helps track every change, collaborate seamlessly, and maintain organized project versions — essential for modern software development.

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.

Conclusion:

Jenkins automates the entire software development pipeline, from integration to deployment. Its simple setup, extensible plugins, and powerful automation make it a vital tool for continuous integration and DevOps workflows.

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: -

Build -> Deploy -> Test -> Release

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 element in declarative pipeline syntax, which is a collection of all stages in a Jenkinsfile. We declare

  Pipeline { }

  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 Jenkins that executes the Pipeline.

  Node {}

- ***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'){
          .....
         }
```

Conclusion:

Jenkins Pipelines automate the software development workflow by combining integration, testing, and deployment in a continuous manner.
They provide both **declarative** and **scripted** approaches, making CI/CD efficient, reliable, and easy to manage within Jenkins.

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

Conclusion:

Apache Tomcat is a powerful and reliable open-source Java servlet container that supports running Java-based web applications efficiently. It simplifies deploying and managing dynamic web content through servlets and JSPs. By setting up and configuring Tomcat correctly, developers can host, test, and integrate applications seamlessly with Jenkins or other CI/CD tools, ensuring smooth and automated web application deployment.

Experiment No.6

Aim: Test Software Applications: To Setup and Run Selenium Tests in Jenkins Using Maven

Objectives: From this experiment, the student will be able

- To setup and run Selenium tests in Jenkins using Maven for a web application.
- To setup and run Selenium tests with Python for a web application.

Outcomes: The learner will be able to

- To have a basic understanding of how to use Selenium with Python to test a web application and will be able to write and run basic Selenium tests.

Hardware / Software Required: Linux / Windows Operating System, Python, Java JDK

Prerequisites with Python:

1. Python installed and set up on your machine.
2. Selenium and ChromeDriver installed on your machine.
3. A web application to test.

Theory:

Selenium WebDriver: Selenium WebDriver is a tool that allows interaction with web pages using various programming languages, including Python. It enables the automation of web browsers, testing web applications, and performing repetitive tasks.

Python Virtual Environment: A Python virtual environment is a self-contained directory that contains all the necessary Python packages and modules for a project. It allows isolation of the project's dependencies and avoids conflicts with other Python packages installed on the system.

Pytest: Pytest is a testing framework for Python that allows the creation of simple and scalable tests. It provides various features like fixtures, parameterization, and plugins for writing complex tests.

Jenkins: Jenkins is a popular open-source automation server used for continuous integration and continuous delivery (CI/CD) of software applications. It enables the automation of building, testing, and deploying software applications.

General Steps-

Step 1: Install Selenium WebDriver for Python:

- Download and install Python on your machine.
- Install Selenium WebDriver for Python using pip.

Step 2: Create a Python virtual environment for Selenium tests:

- Create a new directory for your project.
- Change into the newly created directory.
- Create a new Python virtual environment for your project.

Step 3: Write basic Selenium tests using Python:

- Create a new Python file for your Selenium tests.
- Open the Python file and import the required packages.
- Create a new WebDriver instance and navigate to the web page you want to test.
- Find an element on the web page and interact with it.
- Use Pytest to write and run tests.
- Save the file and run the test locally to make sure it works.

Step 4: Integrate Selenium tests with Jenkins:

- Create a new Jenkins job.
- Add a new "Execute shell" step and enter the command to install Selenium and run Pytest.
- Save the job configuration and run the job to make sure it builds and runs the Selenium tests successfully.

Conclusion:

By the end of this experiment, we will gain a clear understanding of how to use Selenium with Python to test web applications. We will be able to set up a Python virtual environment for Selenium testing, write and execute basic Selenium scripts, and integrate them with Jenkins for continuous testing. We will also be capable of extending these tests further, connecting them with other tools, and configuring Jenkins to run them automatically on a schedule or whenever new code is pushed to the repository.

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.

Conclusion:

By the end of this experiment, we understood the core concepts of Docker, including containers, images, and registries. We learn how to install Docker Desktop, create a Docker Hub account, and execute basic Docker commands such as building, running, pushing, and pulling images. Through this, we gained practical knowledge of containerization, how applications and their dependencies can be packaged and deployed consistently across different environments, making software delivery faster, reliable, and portable.

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.

Conclusion:

By the end of this experiment, we learned how to create and use a **Dockerfile** to build custom Docker images. We understood the role of each instruction within a Dockerfile and how the docker build and docker run commands help generate and execute containers. By building a sample PHP web application image, we gained hands-on experience in containerizing applications, exposing ports, and running them locally, showing how Docker simplifies deployment and ensures consistent environments across different systems.

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. 90

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. 91 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. 92

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.

Conclusion:

In this experiment, we learned how to install and configure Ansible on Windows, integrate it with Jenkins, and execute playbooks through Jenkins pipelines. We also practiced running Ansible ad hoc commands to perform tasks like rebooting servers, transferring files, managing directories, and installing packages. This gave us hands-on experience in automating and managing IT infrastructure efficiently using configuration management principles.

Experiment No.10

Case Study

Aim: A Comparative study of Software Development Life Cycle – Waterfall Cycle vs Agile vs DevOps

Objectives: To compares SDLC methodologies used in the industry and provides insights on which approach to adopt for software development projects.

Outcome: To understand and evaluate strengths and weaknesses of compared methodologies.

Introduction: The software development life cycle (SDLC) is a structured approach to software development that involves planning, designing, implementing, testing, and deploying software. Over the years, several SDLC models have been developed to manage the software development process, and each has its own advantages and disadvantages. This case study compares three of the most popular SDLC models: Waterfall Cycle, Agile, and DevOps, to highlight the differences between them and help organizations choose the best one based on their needs.

Waterfall Cycle: The Waterfall Cycle is a linear, sequential approach to software development that involves completing each phase of the development process before moving onto the next one. This model is popular among companies that have a well-defined scope, clear requirements, and a predictable development environment. The Waterfall Cycle is beneficial for projects that require a high level of documentation, regulatory compliance, and risk management. However, it is less suitable for projects that require flexibility, rapid iteration, and continuous feedback.

Agile: Agile is an iterative, incremental approach to software development that emphasizes flexibility, collaboration, and continuous improvement. The Agile model is based on the principles of the Agile Manifesto, which values individuals and interactions, working software, customer collaboration, and responding to change. Agile is suitable for projects that require frequent feedback, customer involvement, and changing requirements. The Agile model allows for faster delivery, more frequent releases, and improved quality through continuous testing and integration.

DevOps: DevOps is a software development model that emphasizes collaboration between development and operations teams to streamline the development process and improve software delivery. DevOps aims to break down the barriers between development and operations teams, allowing them to work together seamlessly. The DevOps model uses automated tools to improve the speed, quality, and reliability of software development. DevOps is suitable for organizations that require fast, reliable, and scalable software development and deployment.

Comparative Study: The table below summarizes the differences between the Waterfall Cycle, Agile, and DevOps models:

Model	Advantages	Disadvantages
Waterfall Cycle	Predictable, well-defined process	Inflexible, limited customer involvement
Agile	Flexible, iterative, customer-focused	Requires more active involvement from the customer
DevOps	Fast, reliable, scalable, automation-driven	Requires significant investment in automation

Note: In this case study students will summarize strengths and weaknesses of compared methodologies in terms introduction, features advantages disadvantages etc. in a table in their own words.

Additional Learning: The choice of SDLC methodology depends on the specific requirements of the project. The Waterfall SDLC methodology is suitable for projects with clear and well-defined requirements, whereas the Agile SDLC methodology is suitable for projects where changes are expected. The DevOps SDLC methodology is suitable for projects that require continuous deployment and release of new features. Ultimately, the company should choose the approach that best fits its specific needs, taking into account factors such as budget, project scope, and timelines.

Conclusion: Choosing the right SDLC model depends on the specific needs and goals of each organization. The Waterfall Cycle is suitable for projects with a well-defined scope and clear requirements, while Agile is ideal for projects that require flexibility, rapid iteration, and frequent customer involvement. DevOps is a more advanced model that emphasizes collaboration between development and operations teams and uses automation to streamline the development process. By understanding the differences between these three models, organizations can choose the one that best fits their needs and achieve successful software development.