List of Practical

Sr. No.	List of Practical's
1.	Study practical on software development life cycle
2.	Study and implement Linux commands
3.	Study practical on installation of java, Tomcat Server
4.	Study practical on DevOps life cycle & stages
5.	Study practical on DevOps Tools (Docker, Jenkins, Git, Jira, copado)
6.	Learn about DevOps Pipeline (CI /CD) using any tool
7.	Study Practical on AWS for DevOps
8.	Study Practical on Microsoft Azur for DevOps
9.	Study Practical on Google Cloud for DevOps
10.	Study Practical on Salesforce with Copado for DevOps
11.	To setup and configure of Jenkins
12.	To create Job and manage it using Jenkins

Practical No. 1

Aim: Study practical on software development life cycle

Theory: Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.
SDLC defines the complete cycle of development i.e. all the tasks involved in planning, creating, testing, and deploying a Software Product.

Software Development Life Cycle Process
SDLC is a process that defines the various stages involved in the development of software for delivering a high-quality product. SDLC stages cover the complete life cycle of a software i.e. from inception to retirement of the product.
Adhering to the SDLC process leads to the development of the software in a systematic and disciplined manner.

Purpose: Purpose of SDLC is to deliver a high-quality product which is as per the customer's requirement.

SDLC has defined its phases as, Requirement gathering, Designing, Coding, Testing, and Maintenance. It is important to adhere to the phases to provide the Product in a systematic manner.

For Example, A software has to be developed and a team is divided to work on a feature of the product and is allowed to work as they want. One of the developers decides to design first whereas the other decides to code first and the other on the documentation part.

This will lead to project failure because of which it is necessary to have a good knowledge and understanding among the team members to deliver an expected product.

SDLC Phases:

Phase 1: Requirement collection and analysis
The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and recognization of the risks involved is also done at this stage.
This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project.
Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

Phase 2: Feasibility study
Once the requirement analysis phase is completed the next sdlc step is to define and document software needs. This process conducted with the help of 'Software Requirement Specification' document also known as 'SRS' document. It includes everything which should

be designed and developed during the project life cycle.

There are mainly five types of feasibilities checks:

Economic: Can we complete the project within the budget or not?
Legal: Can we handle this project as cyber law and other regulatory framework/compliances.
Operation feasibility: Can we create operations which is expected by the client?
Technical: Need to check whether the current computer system can support the software
Schedule: Decide that the project can be completed within the given schedule or not.

Phase 3: Design
In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

This design phase serves as input for the next phase of the model.

There are two kinds of design documents developed in this phase:

High-Level Design (HLD)

Brief description and name of each module
An outline about the functionality of every module
Interface relationship and dependencies between modules
Database tables identified along with their key elements
Complete architecture diagrams along with technology details
Low-Level Design (LLD)

Functional logic of the modules
Database tables, which include type and size
Complete detail of the interface
Addresses all types of dependency issues
Listing of error messages
Complete input and outputs for every module

Phase 4: Coding
Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

In this phase, Developer needs to follow certain predefined coding guidelines. They also need to use programming tools like compiler, interpreters, debugger to generate and implement the code.

Phase 5: Testing
Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

During this phase, QA and testing team may find some bugs/defects which they communicate

to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

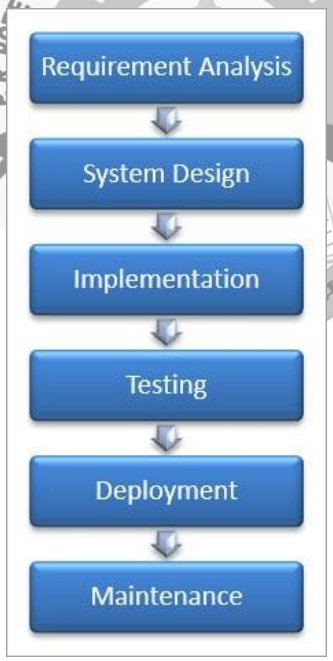
Phase 6: Installation/Deployment
Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

Phase 7: Maintenance
Once the system is deployed, and customers start using the developed system, following 3 activities occur

Bug fixing – bugs are reported because of some scenarios which are not tested at all
Upgrade – Upgrading the application to the newer versions of the Software
Enhancement – Adding some new features into the existing software
The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

Popular SDLC Models
Here, are some of the most important models of Software Development Life Cycle (SDLC):

Waterfall model in SDLC



The waterfall is a widely accepted SDLC model. In this approach, the whole process of the software development is divided into various phases of SDLC. In this SDLC model, the outcome of one phase acts as the input for the next phase.

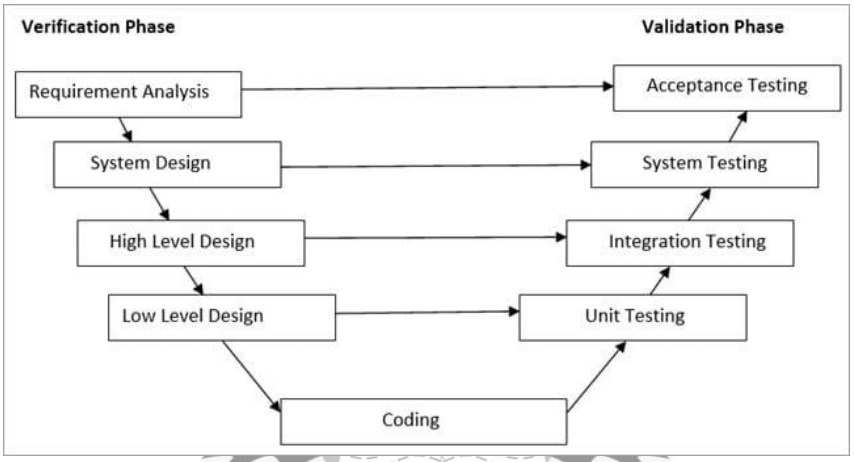
This SDLC model is documentation-intensive, with earlier phases documenting what need be performed in the subsequent phases.

Incremental Model in SDLC

The incremental model is not a separate model. It is essentially a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC life cycle process is repeated, with each release adding more functionality until all requirements are met. In this method, every cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.

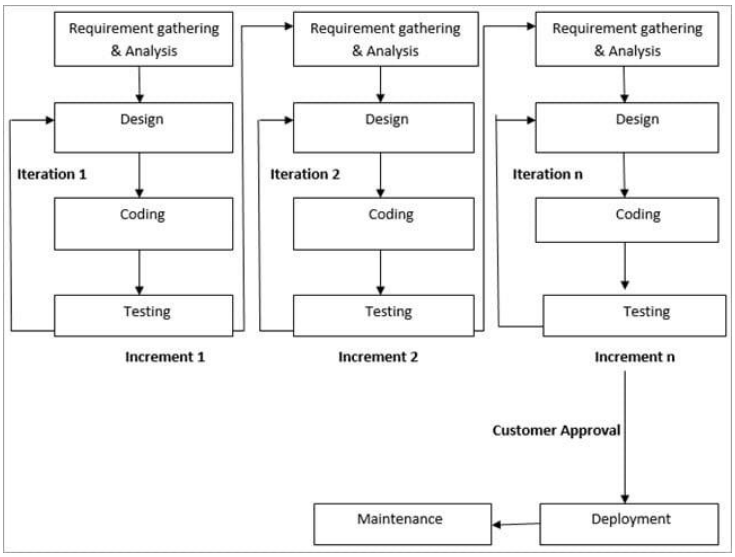
V-Model in SDLC

In this type of SDLC model testing and the development, the phase is planned in parallel. So, there are verification phases of SDLC on the side and the validation phase on the other side. V-Model joins by Coding phase.

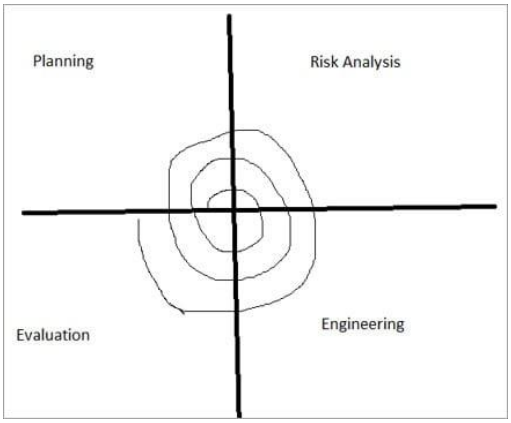


Agile Model in SDLC

Agile methodology is a practice which promotes continue interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.



Spiral Model



The spiral model is a risk-driven process model. This SDLC testing model helps the team to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc.

This model adopts the best features of the prototyping model and the waterfall model. The spiral methodology is a combination of rapid prototyping and concurrency in design and development activities.

Big bang model

Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come.

This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements is either unknown or final release date is not given.

- **Conclusion:**

Practical No. 2

Aim: Study and implement Linux commands

Theory:
Linux is a Unix-Like operating system. All the Linux/Unix commands are run in the terminal provided by the Linux system. This terminal is just like the command prompt of Windows OS. Linux/Unix commands are case-sensitive. The terminal can be used to accomplish all Administrative tasks. This includes package installation, file manipulation, and user management. Linux terminal is user-interactive. The terminal outputs the results of commands which are specified by the user itself. Execution of typed command is done only after you press the Enter key. The Linux command line is a text interface to your computer. Often referred to as the shell, terminal, console, prompt or various other names, it can give the appearance of being complex and confusing to use.

Linux provides a CLI (Command Line Interface) to communicate with the OS. Here are the most basic of the Linux Commands.

1. Basic Commands

- **Pwd :** This command Displays the current working directory of the terminal.
  syntax: $ pwd
- **echo:** This command writes its arguments to standard output.
  syntax:$ echo "<text>"
- **su:** This command is used to switch to root-user so that superuser permissions can be used to execute commands.
  syntax:$ su
- **su <username>**: This command is used to switch to a different user whose name is passed as the argument.
  syntax: $ su <username>
- **sudo:** This command executes only that command with root/ superuser privileges.

  syntax: $ sudo <command>

  - sudo useradd <username> Adding a new user
  - sudo passwd <username> Setting a password for the new user
  - sudo userdel <username> Deleting the user
  - sudo groupadd <groupname> Adding a new group
  - sudo groupdel <groupname> Deleting the group
  - sudo usermod -g <groupname> <username> Adding a user to a primary group

- **clear:** This command is used to clear the terminal screen. Contents will not actually be deleted in this case, only scrolled down. You can also clear the screen by pressing Ctrl+L on the keyboard.

syntax: $ clear

2. Working with Files:

- **cp:** This command copies files and directories. A copy of the file/directory copied, still remains in the working directory.

syntax:$ cp <flag> {filename} /pathname/

  - **cp -i** Enters interactive mode; CLI asks before overwriting files
  - **cp -n** Does not overwrite the file
  - **cp -u** Updates the destination file only when the source file is different from the destination file
  - **cp -R** Recursive copy for copying directories; Copies even hidden files
  - **cp -v** Verbose; Prints informative messages

- **mv:** This command moves files and directories from one directory to another. The file/directory once moved, is deleted from the working directory.

syntax:$ mv <flag> {filename} /pathname/

  - mv -i Enters interactive mode; CLI asks before overwriting files
  - mv -u Updates the destination file only when the source file is different from the destination file
  - mv -v Verbose; Prints source and destination files

- **rm:** This command removes files from a directory. By default, the rm command does not remove directories. Once removed, the contents of a file cannot be recovered.

syntax: $ rm <flag> {filename}

  - rm –r Removes even non-empty directories.
  - rm –rp Removes non-empty directories including parent and subdirectories.

- **grep:** This command is used to search for a particular string/ word in a text file. This is similar to “Ctrl+F”, but executed via a CLI.

syntax:$ grep <flag or element_to_search> {filename}

  - grep -i Returns the results for case insensitive strings
  - grep -nReturns the matching strings along with their line number
  - grep -vReturns the result of lines not matching the search string
  - grep -cReturns the number of lines in which the results matched the search string

- **cat :**This command can read, modify or concatenate text files. It also displays file contents.

syntax: $ cat <flag> {filename}

  - cat -b This is used to add line numbers to non-blank lines
  - cat -n This is used to add line numbers to all lines

- cat -s This is used to squeeze blank lines into one line
- cat –E Show $ at the end of line

3. Working with Directories:

- **ls:**This command lists all the contents in the current working directory.

syntax: $ ls <flag>

  - ls <path name> By specifying the path after ls, the content in that path will be displayed
  - ls –l Using 'l' flag, lists all the contents along with its owner settings, permissions & time stamp (long format)
  - ls –a Using 'a' flag, lists all the hidden contents in the specified directory
  - ls –author Using '–author' flag, lists the contents in the specified directory along with its owner
  - ls –S Using 'a' flag, sorts and lists all the contents in the specified directory by size
  - ls *.html Using '*' flag, lists only the contents in the directory of a particular format
  - ls –lS > file.txt Using '>' flag, copies the result of ls command into a text file

- **cd:** This command is used to change the current working directory of the user.

syntax: $ cd /pathname/

  - cd ~ This command also changes the directory to home directory
  - cd / Changes the directory to root directory
  - cd .. Changes the directory to its parent directory
  - cd 'xx yy' We specify the folder name in inverted commas because there is a space in the folder name

- **sort:** This command sorts the results of a search either alphabetically or numerically. Files, file contents and directories can be sorted using this command.

syntax: $ sort <flag> {filename}

  - sort -r the flag returns the results in reverse order;
  - sort -f the flag does case insensitive sorting
  - sort -n the flag returns the results as per numerical order

- **mkdir:** This command is used to create a new directory.

syntax: $ mkdir <flag> {directoryname} /pathname/

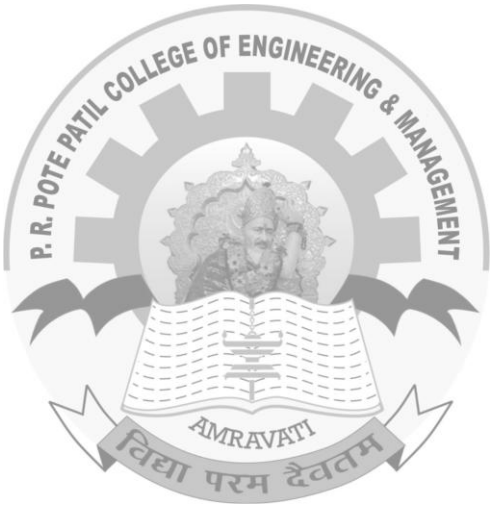
  - mkdir -p Creates both a new parent directory and a sub-directory
  - mkdir –p <filename1>/{f1,f2,f3} This is used to create multiple subdirectories inside the new parent directory

- **rmdir:** This command is used to remove a specified directory. Although by default, it can only remove an empty directory, there are flags which can be deployed to delete the non-empty directories as well.

syntax:$ rmdir <flag> {directoryname}

- rmdir –p Removes both the parent and child directory
- rmdir –pv Removes all the parent and subdirectories along with the verbose.

- **Conclusion:**



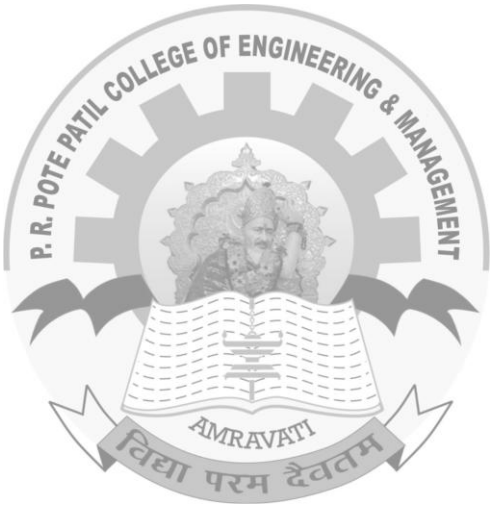
Practical No. 3

Aim: Study practical on installation of java, Tomcat Server .

Theory:

Write Down Basic Theory and Installation steps for JAVA and Apache Tomcat Server

- **Conclusion:**



Experiment No: 4

Aim: Study practical on DevOps life cycle & stages.

Theory:

The DevOps is the combination of two words, one is Development and other is Operations. It is a culture to promote the development and operation process collectively.

- **What is DevOps?**

The DevOps is a combination of two words, one is software Development, and second is Operations. This allows a single team to handle the entire application lifecycle, from development to testing, deployment, and operations. DevOps helps you to reduce the disconnection between software developers, quality assurance (QA) engineers, and system administrators. DevOps promotes collaboration between Development and Operations team to deploy code to production faster in an automated & repeatable way. DevOps helps to increase organization speed to deliver applications and services. It also allows organizations to serve their customers better and compete more strongly in the market.

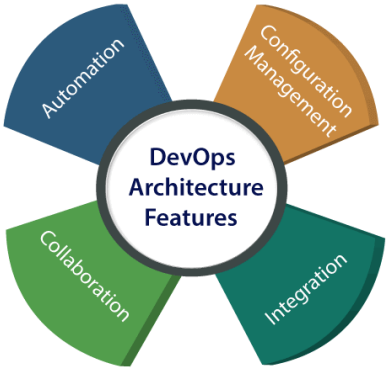
DevOps can also be defined as a sequence of development and IT operations with better communication and collaboration.

DevOps has become one of the most valuable business disciplines for enterprises or organizations. With the help of DevOps, quality, and speed of the application delivery has improved to a great extent.

DevOps is nothing but a practice or methodology of making "Developers" and "Operations" folks work together. DevOps represents a change in the IT culture with a complete focus on rapid IT service delivery through the adoption of agile practices in the context of a system-oriented approach.

- **Why DevOps?**
  - The operation and development team worked in complete isolation.
  - After the design-build, the testing and deployment are performed respectively. That's why they consumed more time than actual build cycles.
  - Without the use of DevOps, the team members are spending a large amount of time on designing, testing, and deploying instead of building the project.
  - Manual code deployment leads to human errors in production.
  - Coding and operation teams have their separate timelines and are not in synch, causing further delays.

- **DevOps Architecture Features:**



1) Automation
Automation can reduce time consumption, especially during the testing and deployment phase. The productivity increases, and releases are made quicker by automation. This will lead in catching bugs quickly so that it can be fixed easily. For contiguous delivery, each code is defined through automated tests, cloud-based services, and builds. This promotes production using automated deploys.

2) Collaboration
The Development and Operations team collaborates as a DevOps team, which improves the cultural model as the teams become more productive with their productivity, which strengthens accountability and ownership. The teams share their responsibilities and work closely in sync, which in turn makes the deployment to production faster.

3) Integration
Applications need to be integrated with other components in the environment. The integration phase is where the existing code is combined with new functionality and then tested. Continuous integration and testing enable continuous development. The frequency in the releases and micro-services leads to significant operational challenges. To overcome such problems, continuous integration and delivery are implemented to deliver in a **quicker, safer**, and **reliable manner**.

4) Configuration management
It ensures the application to interact with only those resources that are concerned with the environment in which it runs. The configuration files are not created where the external configuration to the application is separated from the source code. The configuration file can be written during deployment, or they can be loaded at the run time, depending on the environment in which it is running.

- **DevOps Advantages and Disadvantages**

Advantages:

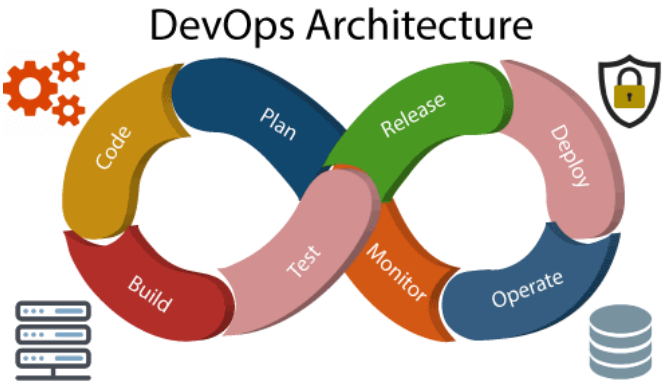
- DevOps is an excellent approach for quick development and deployment of applications.
- It responds faster to the market changes to improve business growth.
- DevOps escalate business profit by decreasing software delivery time and transportation costs.
- DevOps clears the descriptive process, which gives clarity on product development and delivery.
- It improves customer experience and satisfaction.
- DevOps simplifies collaboration and places all tools in the cloud for customers to access.
- DevOps means collective responsibility, which leads to better team engagement and productivity.

Disadvantages:

- DevOps professional or expert's developers are less available.
- Developing with DevOps is so expensive.
- Adopting new DevOps technology into the industries is hard to manage in short time.
- Lack of DevOps knowledge can be a problem in the continuous integration of automation projects.

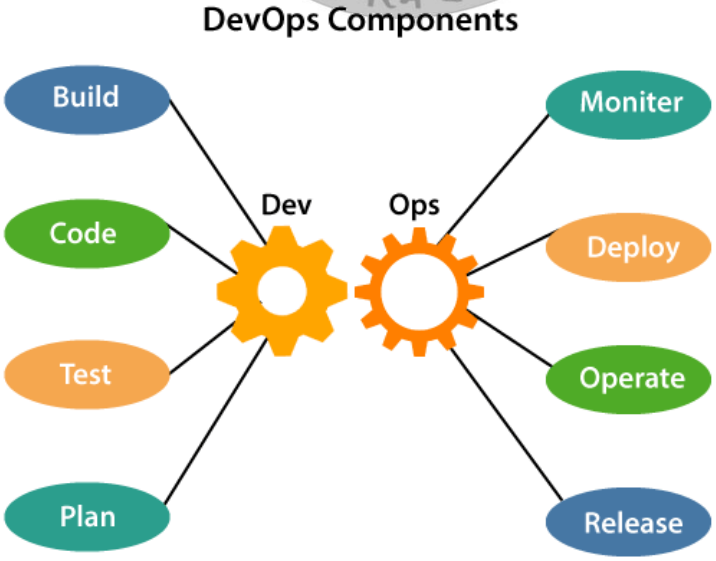
- **Prerequisite:**To learn DevOps, you should have basic knowledge of Linux, and at least one Scripting language.

➢ **DevOps Architecture:**



Development and operations both play essential roles in order to deliver applications. The deployment comprises analyzing the requirements, designing, developing, and testing of the software components or frameworks.
The operation consists of the administrative processes, services, and support for the software. When both the development and operations are combined with collaborating, then the DevOps architecture is the solution to fix the gap between deployment and operation terms; therefore, delivery can be faster.
DevOps architecture is used for the applications hosted on the cloud platform and large distributed applications. Agile Development is used in the DevOps architecture so that integration and delivery can be contiguous. When the development and operations team works separately from each other, then it is time-consuming to design, test, and deploy. And if the terms are not in sync with each other, then it may cause a delay in the delivery. So DevOps enables the teams to change their shortcomings and increases productivity.

Below are the various components that are used in the DevOps architecture:



1. **Build:** Without DevOps, the cost of the consumption of the resources was evaluated based on the pre-defined individual usage with fixed hardware allocation. And with DevOps, the usage of cloud, sharing of resources comes into the picture, and the build is dependent upon the user's need, which is a mechanism to control the usage of resources or capacity.

2. **Code:** Many good practices such as Git enables the code to be used, which ensures writing the code for business, helps to track changes, getting notified about the reason behind the difference in the actual and the expected output, and if necessary reverting to the original code developed. The code can be appropriately arranged in files, folders, etc. And they can be reused.

3. **Test:** The application will be ready for production after testing. In the case of manual testing, it consumes more time in testing and moving the code to the output. The testing can be automated, which decreases the time for testing so that the time to deploy the code to production can be reduced as automating the running of the scripts will remove many manual steps.

4. **Plan:** DevOps use Agile methodology to plan the development. With the operations and development team in sync, it helps in organizing the work to plan accordingly to increase productivity.

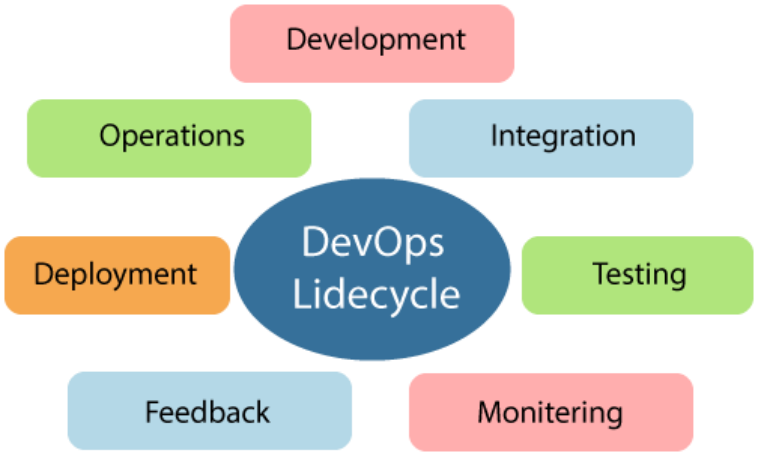
5. **Monitor:** Continuous monitoring is used to identify any risk of failure. Also, it helps in tracking the system accurately so that the health of the application can be checked. The monitoring becomes more comfortable with services where the log data may get monitored through many third-party tools such as Splunk.

6. **Deploy:** Many systems can support the scheduler for automated deployment. The cloud management platform enables users to capture accurate insights and view the optimization scenario, analytics on trends by the deployment of dashboards.

7. **Operate:** DevOps changes the way traditional approach of developing and testing separately. The teams operate in a collaborative way where both the teams actively participate throughout the service lifecycle. The operation team interacts with developers, and they come up with a monitoring plan which serves the IT and business requirements.

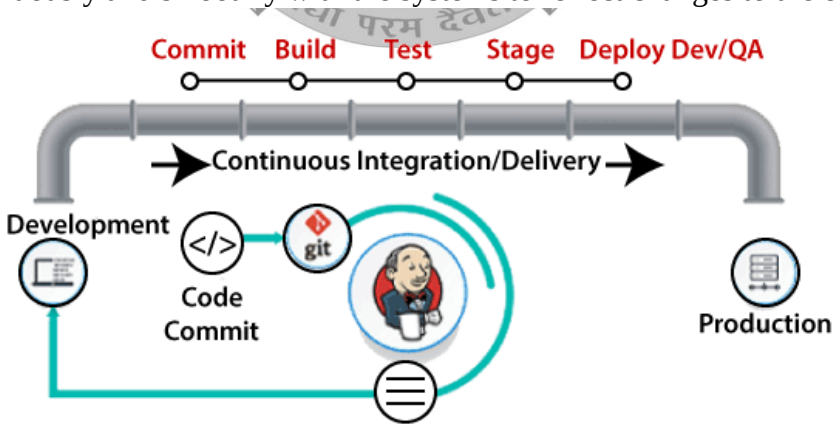
8. **Release:** Deployment to an environment can be done by automation. But when the deployment is made to the production environment, it is done by manual triggering. Many processes involved in release management commonly used to do the deployment in the production environment manually to lessen the impact on the customers.

- **DevOps Lifecycle:** DevOps defines an agile relationship between operations and Development. It is a process that is practiced by the development team and operational engineers together from beginning to the final stage of the product.



1) Continuous Development: This phase involves the planning and coding of the software. The vision of the project is decided during the planning phase. And the developers begin developing the code for the application. There are no DevOps tools that are required for planning, but there are several tools for maintaining the code.

2) Continuous Integration: This stage is the heart of the entire DevOps lifecycle. It is a software development practice in which the developers require to commit changes to the source code more frequently. This may be on a daily or weekly basis. Then every commit is built, and this allows early detection of problems if they are present. Building code is not only involved compilation, but it also includes unit testing, integration testing, code review, and packaging.
The code supporting new functionality is continuously integrated with the existing code. Therefore, there is continuous development of software. The updated code needs to be integrated continuously and smoothly with the systems to reflect changes to the end-users.



Jenkins is a popular tool used in this phase. Whenever there is a change in the Git repository, then Jenkins fetches the updated code and prepares a build of that code, which is an executable file in the form of war or jar. Then this build is forwarded to the test server or the production server.

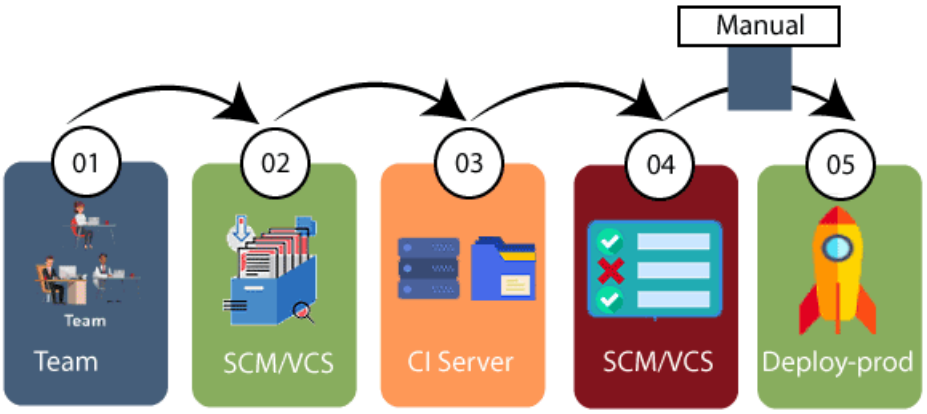
3) Continuous Testing: This phase, where the developed software is continuously testing for bugs. For constant testing, automation testing tools such as **TestNG, JUnit, Selenium**, etc are used. These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there is no flaw in the functionality. In this phase, **Docker** Containers can be used for simulating the test environment.

4) Selenium does the automation testing, and TestNG generates the reports. This entire testing phase can automate with the help of a Continuous Integration tool called **Jenkins**.
Automation testing saves a lot of time and effort for executing the tests instead of doing this manually. Apart from that, report generation is a big plus. The task of evaluating the test cases that failed in a test suite gets simpler. Also, we can schedule the execution of the test cases at predefined times. After testing, the code is continuously integrated with the existing code.

4) Continuous Monitoring: Monitoring is a phase that involves all the operational factors of the entire DevOps process, where important information about the use of the software is recorded and carefully processed to find out trends and identify problem areas. Usually, the monitoring is integrated within the operational capabilities of the software application.
It may occur in the form of documentation files or maybe produce large-scale data about the application parameters when it is in a continuous use position. The system errors such as server not reachable, low memory, etc are resolved in this phase. It maintains the security and availability of the service.

5) Continuous Feedback: The application development is consistently improved by analyzing the results from the operations of the software. This is carried out by placing the critical phase of constant feedback between the operations and the development of the next version of the current software application.
The continuity is the essential factor in the DevOps as it removes the unnecessary steps which are required to take a software application from development, using it to find out its issues and then producing a better version. It kills the efficiency that may be possible with the app and reduce the number of interested customers.

6) Continuous Deployment: In this phase, the code is deployed to the production servers. Also, it is essential to ensure that the code is correctly used on all the servers.



The new code is deployed continuously, and configuration management tools play an essential role in executing tasks frequently and quickly. Here are some popular tools which are used in this phase, such as Chef, Puppet, Ansible, and SaltStack.

Containerization tools are also playing an essential role in the deployment phase. Vagrant and Docker are popular tools that are used for this purpose. These tools help to produce consistency across development, staging, testing, and production environment. They also help in scaling up and scaling down instances softly.

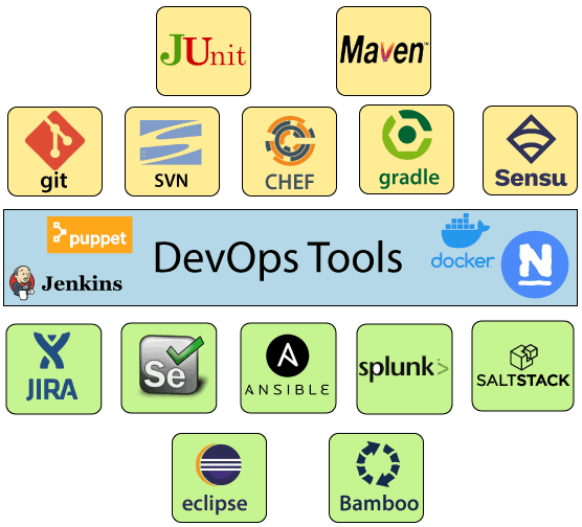
Containerization tools help to maintain consistency across the environments where the application is tested, developed, and deployed. There is no chance of errors or failure in the production environment as they package and replicate the same dependencies and packages used in the testing, development, and staging environment. It makes the application easy to run on different computers.

7) Continuous Operations: All DevOps operations are based on the continuity with complete automation of the release process and allow the organization to accelerate the overall time to market continuingly.
It is clear from the discussion that continuity is the critical factor in the DevOps in removing steps that often distract the development, take it longer to detect issues and produce a better version of the product after several months. With DevOps, we can make any software product more efficient and increase the overall count of interested customers in your product.

- **Conclusion:**

Experiment No: 5

Aim: Study practical on DevOps Tools (Docker, Jenkins, Git, Jira, copado)



1) Puppet: Puppet is the most widely used DevOps tool. It allows the delivery and release of the technology changes quickly and frequently. It has features of versioning, automated testing, and continuous delivery. It enables to manage entire infrastructure as code without expanding the size of the team.

Features

- Real-time context-aware reporting.
- Model and manage the entire environment.
- Defined and continually enforce infrastructure.
- Desired state conflict detection and remediation.
- It inspects and reports on packages running across the infrastructure.
- It eliminates manual work for the software delivery process.
- It helps the developer to deliver great software quickly.

2) Ansible: Ansible is a leading DevOps tool. Ansible is an open-source IT engine that automates application deployment, cloud provisioning, intra service orchestration, and other IT tools. It makes it easier for DevOps teams to scale automation and speed up productivity.
Ansible is easy to deploy because it does not use any agents or custom security infrastructure on the client-side, and by pushing modules to the clients. These modules are executed locally on the client-side, and the output is pushed back to the Ansible server.

Features

- It is easy to use to open source deploy applications.
- It helps in avoiding complexity in the software development process.
- It eliminates repetitive tasks.
- It manages complex deployments and speeds up the development process.

3) Docker: Docker is a high-end DevOps tool that allows building, ship, and run distributed applications on multiple systems. It also helps to assemble the apps quickly from the components, and it is typically suitable for container management.

Features

- It configures the system more comfortable and faster.
- It increases productivity.
- It provides containers that are used to run the application in an isolated environment.
- It routes the incoming request for published ports on available nodes to an active container. This feature enables the connection even if there is no task running on the node.
- It allows saving secrets into the swarm itself.

4) Nagios: Nagios is one of the more useful tools for DevOps. It can determine the errors and rectify them with the help of network, infrastructure, server, and log monitoring systems.

Features

- It provides complete monitoring of desktop and server operating systems.
- The network analyzer helps to identify bottlenecks and optimize bandwidth utilization.
- It helps to monitor components such as services, application, OS, and network protocol.
- It also provides to complete monitoring of Java Management Extensions.

5) CHEF: A chef is a useful tool for achieving scale, speed, and consistency. The chef is a cloud-based system and open source technology. This technology uses Ruby encoding to develop essential building blocks such as recipes and cookbooks. The chef is used in infrastructure automation and helps in reducing manual and repetitive tasks for infrastructure management.

Chef has got its convention for different building blocks, which are required to manage and automate infrastructure.

Features

- It maintains high availability.
- It can manage multiple cloud environments.
- It uses popular Ruby language to create a domain-specific language.
- The chef does not make any assumptions about the current status of the node. It uses its mechanism to get the current state of the machine.

6) Jenkins: Jenkins is a DevOps tool for monitoring the execution of repeated tasks. Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. It helps to integrate project changes more efficiently by finding the issues quickly.

Features

- Jenkins increases the scale of automation.
- It can easily set up and configure via a web interface.
- It can distribute the tasks across multiple machines, thereby increasing concurrency.
- It supports continuous integration and continuous delivery.
- It offers 400 plugins to support the building and testing any project virtually.
- It requires little maintenance and has a built-in GUI tool for easy updates.

7) Git: Git is an open-source distributed version control system that is freely available for everyone. It is designed to handle minor to major projects with speed and efficiency. It is developed to co-ordinate the work among programmers. The version control allows you to track and work together with your team members at the same workspace. It is used as a critical distributed version-control for the DevOps tool.

Features

- It is a free open source tool.
- It allows distributed development.
- It supports the pull request.
- It enables a faster release cycle.
- Git is very scalable.
- It is very secure and completes the tasks very fast.

8) SALTSTACK: Stackify is a lightweight DevOps tool. It shows real-time error queries, logs, and more directly into the workstation. SALTSTACK is an ideal solution for intelligent orchestration for the software-defined data center.

Features

- It eliminates messy configuration or data changes.
- It can trace detail of all the types of the web request.
- It allows us to find and fix the bugs before production.
- It provides secure access and configures image caches.
- It secures multi-tenancy with granular role-based access control.
- Flexible image management with a private registry to store and manage images.

9) Splunk: Splunk is a tool to make machine data usable, accessible, and valuable to everyone. It delivers operational intelligence to DevOps teams. It helps companies to be more secure, productive, and competitive.

Features

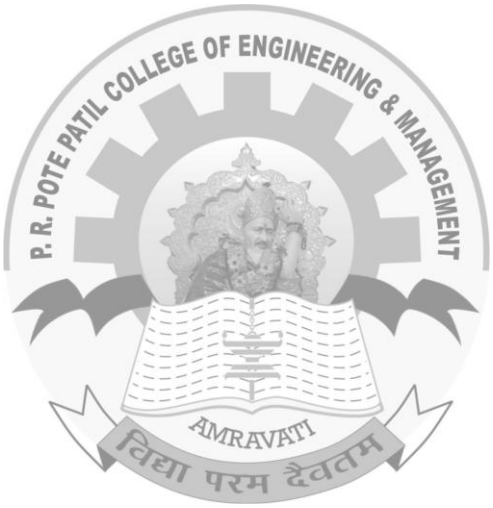
- It has the next-generation monitoring and analytics solution.
- It delivers a single, unified view of different IT services.
- Extend the Splunk platform with purpose-built solutions for security.
- Data drive analytics with actionable insight.

10) Selenium: Selenium is a portable software testing framework for web applications. It provides an easy interface for developing automated tests.

Features

- It is a free open source tool.
- It supports multiplatform for testing, such as Android and ios.
- It is easy to build a keyword-driven framework for a WebDriver.
- It creates robust browser-based regression automation suites and tests.

- **Conclusion:**


P. R. POTE PATIL COLLEGE OF ENGINEERING & MANAGEMENT
AMRAVATI
विद्या परम दैवतम

Experiment No: 6

Aim: Learn about DevOps Pipeline (CI /CD) using any tool

A pipeline in software engineering team is a set of automated processes which allows DevOps professionals and developer to reliably and efficiently compile, build, and deploy their code to their production compute platforms. The most common components of a pipeline in DevOps are build automation or continuous integration, test automation, and deployment automation. A pipeline consists of a set of tools which are classified into the following categories such as:

- Source control
- Build tools
- Containerization
- Configuration management
- Monitoring

- **Continuous Integration Pipeline:** Continuous integration (CI) is a practice in which developers can check their code into a version-controlled repository several times per day. Automated build pipelines are triggered by these checks which allows fast and easy to locate error detection.

  Some significant benefits of CI are:
  - Small changes are easy to integrate into large codebases.
  - More comfortable for other team members to see what you have been working.
  - Fewer integration issues allowing rapid code delivery.
  - Bugs are identified early, making them easier to fix, resulting in less debugging work.

- **Continuous Delivery Pipeline:** Continuous delivery (CD) is the process that allows operation engineers and developers to deliver bug fixes, features, and configuration change into production reliably, quickly, and sustainably. Continuous delivery offers the benefits of code delivery pipelines, which are carried out that can be performed on demand.

  Some significant benefits of the CD are:
  - Faster bug fixes and features delivery.
  - CD allows the team to work on features and bug fixes in small batches, which means user feedback received much quicker. It reduces the overall time and cost of the project.

➢ **DevOps Methodology**

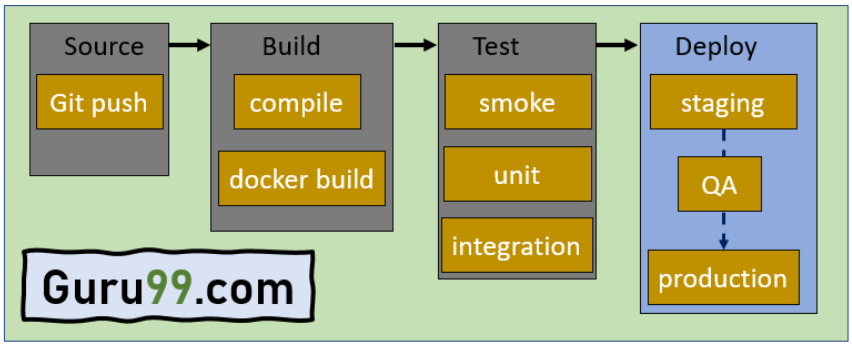
We have a demonstrated methodology that takes an approach to cloud adoption. It accounts for all the factors required for successful approval such as people, process, and technology, resulting in a focus on the following critical consideration:

- **The Teams:** Mission or project and cloud management.
- **Connectivity:** Public, on-premise, and hybrid cloud network access.
- **Automation:** Infrastructure as code, scripting the orchestration and deployment of resources.
- **On-boarding Process:** How the project gets started in the cloud.
- **Project Environment:** TEST, DEV, PROD (identical deployment, testing, and production).
- **Shared Services:** Common capabilities provided by the enterprise.
- **Naming Conventions:** Vital aspect to track resource utilization and billing.
- **Defining Standards Role across the Teams:** Permissions to access resources by job function.

➢ **Stages of a CI/CD pipeline**

A CI/CD pipeline is a runnable specification of the steps that any developer should perform to deliver a new version of any software. Failure in each and every stage triggers a notification via email, Slack, or other communication platforms. It enables responsible developers to know about the important issues.

Here are the important Stages of CI/CD pipeline:



1. Source Stage

In the source stage, CI/CD pipeline is triggered by a code repository. Any change in the program triggers a notification to the CI/CD tool that runs an equivalent pipeline. Other common triggers include user-initiated workflows, automated schedules, and the results of other pipelines.

2. Build Stage

This is the second stage of the CI/CD Pipeline in which you merge the source code and its dependencies. It is done mainly to build a runnable instance of software that you can potentially ship to the end-user.

Programs that are written in languages like C++, Java, C, or Go language should be compiled. On the other hand, JavaScript, Python, and Ruby programs can work without the build stage.

Failure to pass the build stage means there is a fundamental project misconfiguration, so it is better that you address such issue immediately.

3. Test Stage

Test Stage includes the execution of automated tests to validate the correctness of code and the behaviour of the software. This stage prevents easily reproducible bugs from reaching the clients. It is the responsibility of developers to write automated tests.

4. Deploy Stage

This is the last stage where your product goes live. Once the build has successfully passed through all the required test scenarios, it is ready to deploy to live server.

- **Example of CI/CD Pipeline**
  - **Source Code Control:** Host code on GitHub as a private repository. This will help you to integrate your application with major services and software.
  - **Continuous integration:** Use continuous integration and delivery platform CircleCI and commit every code. When the changes notify, this tool will pull the code available in GitHub and process to build and run the test.
  - **Deploy code to UAT:** Configure CircleCI to deploy your code to AWS UAT server.
  - **Deploy to production:** You have to reuse continuous integration steps for deploying code to UAT.

- **Conclusion:**