18ITD44

Hall Ticket Number:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

**IV/IV B.Tech (Regular) DEGREE EXAMINATION**

**November, 2022** **Information Technology**

**Seventh Semester** **DevOps**

**Time:** Three Hours **Maximum:** 50 Marks

***Answer question 1 compulsory.*** **(10X1 = 10Marks)**

***Answer one question from each unit.*** **(4X10=40 Marks)**

| | | | CO | BL | M |
|---|---|---|---|---|---|
| 1 | a) | What is DevOps? | CO1 | L1 | 1M |
| | b) | Compare Continuous Integration and Integration stages in SDLC. | CO1 | L3 | 1M |
| | c) | List Continuous Deployment tools used in DevOps. | CO1 | L1 | 1M |
| | d) | Give syntax used for Git Push command. | CO2 | L2 | 1M |
| | e) | What is Continuous Integration? | CO2 | L1 | 1M |
| | f) | What is GitHub? | CO2 | L1 | 1M |
| | g) | List CI/CD pipeline tools? | CO3 | L1 | 1M |
| | h) | What is Selenium tool? | CO3 | L1 | 1M |
| | i) | Define Continuous Deployment. | CO4 | L1 | 1M |
| | j) | List the Continuous monitoring tools used? | CO4 | L1 | 1M |
| | | **Unit-I** | | | |
| 2 | a) | Discuss about Agile Methodology. | CO1 | L1 | 5M |
| | b) | Explain in detail about DevOps Life Cycle. | CO1 | L1 | 5M |
| | | **(OR)** | | | |
| 3 | a) | Discuss about DevOps stages. | CO1 | L1 | 5M |
| | b) | Distinguish Waterfall Vs Agile methodologies. | CO1 | L2 | 5M |
| | | **Unit-II** | | | |
| 4 | a) | Compare Centralized vs Distributed Version Control Systems | CO2 | L2 | 5M |
| | b) | Discuss about working with Remote Repositories | CO2 | L1 | 5M |
| | | **(OR)** | | | |
| 5 | a) | Discuss in detail about Git. | CO2 | L1 | 5M |
| | b) | Demonstrate the common Commands in Git. | CO2 | L2 | 5M |
| | | **Unit-III** | | | |
| 6 | a) | Discuss Continuous Integration | CO3 | L1 | 5M |
| | b) | Explain in detail about Continuous Testing using Selenium. | CO3 | L1 | 5M |
| | | **(OR)** | | | |
| 7 | a) | Explain in detail CI/CD Pipeline. | CO3 | L1 | 5M |
| | b) | Compare CI/CD Tools. | CO3 | L2 | 5M |
| | | **Unit-IV** | | | |
| 8 | a) | Discuss in detail Containerization with Docker. | CO4 | L1 | 5M |
| | b) | Compare Configuration Management Tools Puppet & Ansible. | CO4 | L2 | 5M |
| | | **(OR)** | | | |
| 9 | a) | Explain in detail about Continuous Deployment. | CO4 | L1 | 5M |
| | b) | Discuss Continuous Monitoring with Nagios. | CO4 | L1 | 5M |

# SCHEME OF EVALUATION

1. ***a) What is DevOps?*** ***CO1 L1 1M***

**Ans)** 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. This allows a single team to handle the entire application lifecycle, from development to testing, deployment, and operations.

1. ***b) Compare Continuous Integration and Integration stages in SDLC.*** ***CO1 L3 1M***

**Ans) Continuous integration** is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.
**In Integration stage** of SDLC, a software implementation is packaged and tested to assure quality.

1. ***c) List Continuous Deployment tools used in DevOps.*** ***CO1 L1 1M***

**Ans)** Puppet, Chef, Ansibel, AWS CodePipeine, BitBucket

1. ***d) Give syntax used for Git Push command.*** ***CO2 L2 1M***

**Ans)** $ git push <option> [<Remote URL><branch name><refspec>...]
Where:
Remote URL → is the url of the remote repository you want to push to.
branch name → is the name of the remote branch you want to push your changes to.

1. ***e) What is Continuous Integration?*** ***CO2 L1 1M***

**Ans)** Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day.

1. ***f) What is GitHub?*** ***CO2 L1 1M***

**Ans)** GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code.

1. ***g) List CI/CD pipeline tools?*** ***CO3 L1 1M***

**Ans)** Jenkins, BitBucket, CircleCI

1. ***h) What is Selenium tool?*** ***CO3 L1 1M***

**Ans)** Selenium consists of a number of tools that do different things: Selenium IDE, Selenium WebDriver and Selenium Grid. These tools support a number of browsers, operating systems, and programming languages.

1. ***i) Define Continuous Deployment.*** ***CO4 L1 1M***

**Ans)** Continuous deployment is a strategy in software development where code changes to an application are released automatically into the production environment.

1. ***j) List the Continuous monitoring tools used?*** ***CO4 L1 1M***

**Ans)** Nagios, Dyna Trace, Splunk

# Unit-I

***2 a) Discuss about Agile Methodology.*** ***CO1 L1 5M***

- In practical terms, agile software development methodologies are all about delivering small pieces of working software quickly to improve customer satisfaction.
- These methodologies use adaptive approaches and teamwork to focus on continuous improvement.
- Instead of developing software sequentially from one phase to the next, which is how the waterfall method ensures product quality, an agile method can promote development and testing as concurrent and continuous processes. Put another way, waterfall development holds that an entire phase should be completed before moving on to the next, whereas agile supports multiple sequences happening at the same time.

***Advantages:***

- Highest priority is to satisfy the customer through early and continuous delivery of valuable software. Customer satisfaction and quality deliverables are the focus.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. Don't fight change, instead learn to take advantage of it.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. Continually provide results throughout a project, not just at its peeks.
- Business people and developers must work together daily throughout the project. Collaboration is key.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. Bring talented and hardworking members to the team and get out of their way.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. Eliminate as many opportunities for miscommunication as possible.
- Working software is the primary measure of progress. It doesn't need to be perfect, it needs to work.

**Dis Advantages:** **[3M]**

- **Less predictable.** The flexibility at the core of the Agile method also means a much lower degree of predictability**. It can be much more difficult to accurately estimate the time necessary or quantify the resources** and efforts required to complete a project. Many teams fear this uncertainty, and that fear can lead to frustration and **poor decision-making**.
- **More time and commitment.** Communication and collaboration is great, but that constant interaction takes more time and energy for everyone involved.
- **Greater demands on developers and clients.** Commitment from everyone involved is required for Agile Methodology to be effective. Anyone who isn't on board can negatively impact the quality of a project.
- **Lack of necessary documentation.** Because tasks are often completed just in time for development under the Agile Method, **documentation tends to be less thorough**, which can lead to misunderstanding and difficulties down the road.
- **Projects easily fall off track.** The less-structured nature of Agile Methodology means projects can easily go astray or run beyond the original scope of the project. ***[2M]***

***2. b) Explain in detail about DevOps Life Cycle.*** ***CO1 L1 5M***

The DevOps lifecycle includes seven phases:

7 stages in DevOps lifecycle:

1. Continuous Development
2. Continuous Integration
3. Continuous Testing
4. Continuous Monitoring
5. Continuous Feedback
6. Continuous Deployment
7. Continuous Operations

**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. [1M]

**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. **Jenkins** is a popular tool used in this phase.

**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.

**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. [2M]

**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. Tools used in this stage are: GetFeedback, Slack, Pendo.

**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**.

**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. [2M]

***(OR)***

***3. a) Discuss about DevOps stages.*** ***CO1 L1 5M***

*There are 5 stages in DevOps. They are discussed below.* ***5 stages → 5M***

1. Version Control
2. Continuous integration
3. Continuous delivery
4. Continuous deployment
5. Continuous monitoring

1. **Version Control:**

- Version control, also known as source control, is the practice of tracking and managing changes to software code. VCS are sometimes known as SCM (Source Code Management) tools or RCS (Revision Control System). Git is free and open source.
- Version control enables teams to deal with conflicts that result from having multiple people working on the same file or project at the same time, and provides a safe way to make changes and roll them back if necessary. Using version control early in your team's or product's lifecycle will facilitate adoption of good habits. Version control systems help software teams work faster and smarter

2. **Continuous integration:**

Continuous integration refers to the build and unit testing stages of the software release process. Every revision that is committed triggers an automated build and test. Continuous Integration in DevOps is the process of automating the build and deploy phase through certain tools and best practices.

Continuous integration has become a very integral part of any software development process. The continuous Integration process helps to answer the following questions for the software development team Addresses the following:

- Do all the software components work together as they should?
- Is the code too complex for integration purposes?
- Does the code adhere to the established coding standards?
- How much code is covered by automated tests?
- Were all the tests successful after the latest change?

3. **Continuous delivery:**

Continuous Delivery (CD) goes one step further from Continuous Integration (CI). It ensures that every code change is tested and ready for the production environment, after a successful build. CI ensures every code is committed to the main code repository whereas CD ensures the system is in an executable state at all times, after changes to code.

The automation of methods to safely deliver changes into production is known as Continuous Delivery.

Continuous Delivery can work only if Continuous Integration is in place. It involves running extensive regression, UI, and performance tests to ensure that the code is production-ready. It's applicable to both bug fixes and new feature releases.

4. **Continuous deployment:**

Continuous Delivery/Deployment (CD) is a process by which an application is delivered to various environments, such as testing or production, once someone (usually a product owner/manager) decides that it is ready to go.

Continuous deployment is a strategy or methodology for software releases where any new code update or change that makes it through the rigorous automated test process is deployed directly into the live production environment where it will be visible to customers.

5. **Continuous monitoring:**

Continuous monitoring in DevOps is the process of identifying threats to the security and compliance rules of a software development cycle and architecture. Also known as continuous control monitoring or CCM. This is one of the most crucial steps in a DevOps lifecycle and will help to achieve true efficiency and scalability.

This is an automated procedure that can be extended to detect similar inconsistencies in IT infrastructures. Continuous monitoring helps business and technical teams determine and interpret analytics to solve crucial issues, as mentioned above, instantaneously. Continuous monitoring or CM is a step towards the end of the DevOps process. The software is usually sent for production before continuous monitoring is conducted.

***3. b) Distinguish Waterfall Vs Agile methodologies.*** ***CO1 L2 5M***
***10 points → 5 marks***

| *Sl. No.* | *Agile* | *Waterfall* |
|---|---|---|
| 1 | It separates the project development lifecycle into sprints (small amount of time dev team has to complete a work). | Software development process is divided into distinct phases. |
| 2 | It follows an incremental approach | Waterfall methodology is a sequential design process. |
| 3 | Agile methodology is known for its flexibility. | Waterfall is a structured software development methodology so most times it can be quite rigid. |
| 4 | Agile is quite a flexible method which allows changes to be made in the project development requirements even if the initial planning has been completed. | There is no scope of changing the requirements once the project development starts. |
| 5 | Agile methodology, follow an iterative development approach because of this planning, development, prototyping and other software development phases may appear more than once. | All the project development phases like designing, development, testing, etc. are completed once in the Waterfall model. |
| 6 | Agile development is a process in which the requirements are expected to change and evolve. | The method is ideal for projects which have definite requirements and changes not at all expected. |
| 7 | In Agile methodology, testing is performed concurrently with software development. | In this methodology, the "Testing" phase comes after the "Build" phase |
| 8 | Agile introduces a product mindset where the software product satisfies needs of its end customers and changes itself as per the customer's demands. | This model shows a project mindset and places its focus completely on accomplishing the project. |
| 9 | Agile methodology works exceptionally well with Time & Materials or non-fixed funding. It may increase stress in fixed-price scenarios. | Reduces risk in the firm fixed price contracts by getting risk agreement at the beginning of the process. |
| 10 | Prefers small but dedicated teams with a high degree of coordination and synchronization. | Team coordination/synchronization is very limited. |

## *Unit-II*

***4 a) Cmpare Centralized vs Distributed Version Control Systems*** ***CO2 L2 5M***

| *Sl. No.* | *Centralized VCS* | *Distributed VCS* |
|---|---|---|
| 1 | Centralized VCS is the simplest form of version control in which the central repository of the server provides the latest code to the clients. | Distributed vcs is a form of vcs where the complete codebase (including the history) is mirrored on every developer's computer. |
| 2 | There are no local repositories | There are local repositories |
| 3 | Works comparatively slower | Works faster |
| 4 | Always require internet connectivity | Developers can work with a local repository without an internet connection |
| 5 | Considers the entire columns for compression. | Considers columns as well as partial columns. |
| 6 | Focuses on synchronizing, tracking and backing up files | Focuses on sharing changes |
| 7 | A failure in the central server terminates all the versions | A failure in the main server does not affect the development |

***4. b) Discuss about working with Remote Repositories*** ***CO2 L1 5M***
***5 operations → 5 M***

- ✓ To be able to collaborate on any Git project, you need to know how to manage your remote repositories. Remote repositories are versions of your project that are hosted on the Internet or network somewhere.
- ✓

Following are the commands used with Git:

- ✓ Git clone, Git Git config, Git init, Git clone, Git add, Git commit, Git status, Git push, Git pull, Git branch, Git merge, git remote, Git log commands.

**Showing Your Remotes:**

- ✓ To see which remote servers you have configured, you can run the git remote command. It lists the shortnames of each remote handle you've specified. If you've cloned your repository,

You can also specify -v, which shows you the URLs that Git has stored for the shortname to be used when reading and writing to that remote:

```
git remote -v
```

**Adding Remote Repositories:**

To add a new remote Git repository as a shortname you can reference easily, run

```
git remote add <shortname> <url>
```

**Fetching and Pulling from Your Remotes:**

As you just saw, to get data from your remote projects, you can run:

```
git fetch <remote>
```

**Pushing to Your Remotes:**

When you have your project at a point that you want to share, you have to push it upstream.

The command for this is simple: git push <remote> <branch>.

**Inspecting a Remote:**

If you want to see more information about a particular remote, you can use the command

```
git remote show <remote>
```

**Renaming and Removing Remotes:**

You can run git remote rename to change a remote's shortname. For instance, if you want to rename pb to paul, you can do so with git remote rename:

```
$ git remote rename pb paul
```

***(OR)***

***5 a) Discuss in detail about Git.*** ***CO2 L1 5M***

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

**It is used for:**

Tracking code changes

Tracking who made changes

Coding collaboration

**What does Git do?**

Manage projects with Repositories

Clone a project to work on a local copy

Control and track changes with Staging and Committing

Branch and Merge to allow for work on different parts and versions of a project

Pull the latest version of the project to a local copy

Push local updates to the main project

**To Install on Windows:**

- Installing Git on Windows is very easy. The msysGit project has one of the easier installation procedures. Simply download the installer exe file from the GitHub page, and run it:

  http://**msysgit.github.io**
- After it's installed, you have both a command-line version (including an **SSH client** that will come in handy later) and the **standard GUI.** [3M]

Following are the commands used with Git:

- ✓ Git clone, Git Git config, Git init, Git clone, Git add, Git commit, Git status, Git push, Git pull, Git branch, Git merge, git remote, Git log commands.

**Working with Git:**

- Initialize Git on a folder, making it a Repository
- Git now creates a hidden folder to keep track of changes in that folder
- When a file is changed, added or deleted, it is considered modified
- You select the modified files you want to Stage
- The Staged files are Committed, which prompts Git to store a permanent snapshot of the files
- Git allows you to see the full history of every commit.
- You can revert back to any previous commit.
- Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commi [[2M]

***5. b) Demonstrate the common Commands in Git.*** ***CO2 L2 5M***

Following are the commonly used Git commands: 12 commands → 5 Marks

Gitconfig command

- Git init command
- Git clone command
- Git add command
- Git commit command
- Git status command
- Git push Command

- Git pull command
- Git Branch Command
- Git Merge Command
- Git remote command
- Git log command

**Git config command**
git-config → Get and set repository or global options

**Git init command**
Usage: git init [repository name]

We have to navigate to our project directory and type the command git init to initialize a Git repository for our local project folder. Git will create a hidden .git directory and use it for keeping its files organized in other subdirectories.

**Git clone command**
**Usage: git clone [URL]**

Suppose, we want to work on a file that is on a remote Github repository as another developer. How can we do that? We can work on this file by clicking on **Clone or Download** and copying the link and pasting it on the terminal with the git clone command. This will import the files of a project from the remote repository to our local system.

**add command**
Usage (i): git add [file(s) name]

This will add the specified file(s) into the Git repository, the staging area, where they are already being tracked by Git and now ready to be committed.
**Git commit command**
**Usage: git commit -m "message"**
This command records or snapshots files permanently in the version history. All the files, which are there in the directory right now, are being saved in the Git file system

**Git status command**
**Usage: git status**
- This command will show the modified status of an existing file and the file addition status of a new file, if any, that has to be committed.

**Git push Command**
- $ git push <option> [<Remote URL><branch name><refspec>...]
- We can use Github Desktop to push to github.

**Git pull command**
The git pull command is used to fetch and merge code changes from the remote repository to the local repository

**Git Branch Command**
A branch is a version of the repository that diverges from the main working project. It is a feature available in most modern version control systems.

**Git Merge Command**
Git allows you to merge the other branch with the currently active branch. You can merge two branches with the help of git merge command.
$ git merge **<branch** name>

**Ex: git merge B1** → merges B1 with the master branch

**Git remote command**
The git remote command lets you create, view, and delete connections to other repositories.
Git remote → List the remote connections you have to other repositories.
Git remote –v → same as 'git remote' but but include the URL of each connection.

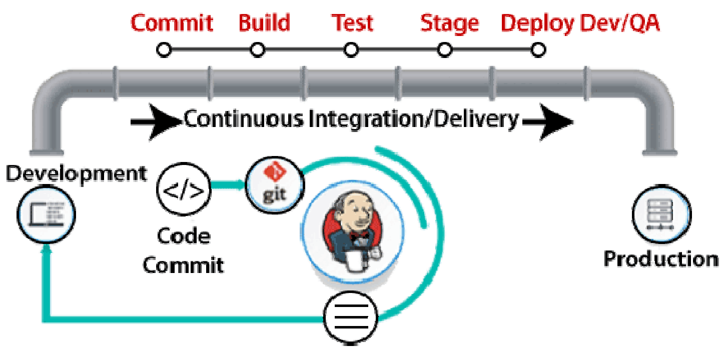
**Git log command**
The git log patch command displays the files that have been modified. It also shows the location of the added, removed, and updated lines.
$git log –patch

## *Unit-III*

***6 a) Discuss Continuous Integration*** ***CO3 L1 5M***

***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.



## Why Continuous Integration?

The continuous Integration process helps to answer the following questions for the software development team.

- First, a developer commits the code to the version control repository. Meanwhile, the Continuous Integration server on the integration build machine polls source code repository for changes (e.g., every few minutes).
- Soon after a commit occurs, the Continuous Integration server detects that changes have occurred in the version control repository, so the Continuous Integration server retrieves the latest copy of the code from the repository and then executes a build script, which integrates the software
- The Continuous Integration server generates feedback by e-mailing build results to the specified project members.
- Unit tests are then carried out if the build of that project passes. If the tests are successful, the code is ready to be deployed to either the staging or production server.

- The Continuous Integration server continues to poll for changes in the version control repository and the whole process repeats.

[3M]

**Reasons for CI is so important:**

- ✓ CI helps to avoid merge conflicts, difficult-to-fix bugs, duplicated code and discrepant coding strategies
- ✓ CI helps to decrease code review time and makes the project code more homogenous
- ✓ CI speeds up the development process and pushes releases closer
- ✓ CI ensures continuous feedback
- ✓ CI helps to reduce the project backlog

**Advantages of CI:**

- Improved developer productivity
- Deliver working software more often
- Find bugs earlier, fix them faster

**Disadvantages of CI:**

- ✓ Development of suitable test procedures necessary
- ✓ Waiting times may occur when multiple developers want to integrate their code around the same time

[2M]

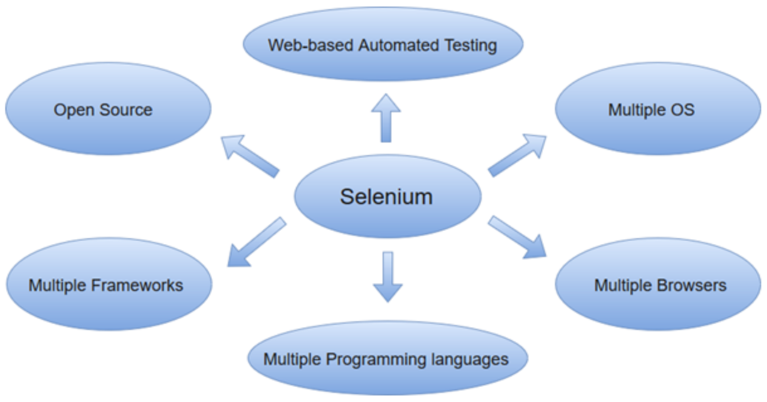
***6. b) Explain in detail about Continuous Testing using Selenium.*** ***CO3 L1 5M***

- Continuous Testing is a form of software testing in which the product is tested early, always, and during the whole process of Continuous Delivery (CD).
- Continuous Testing uses automated checks to ensure that teams receive immediate input during the lifecycle of software development to rapidly eliminate as many risks as possible.
- In addition, team members are able to learn about their product constantly and what can be done to improve consistency and reliability.

**Continuous Testing tools:**

- Based on certain parameters, including types of supported test, learning curve, used programming language, continuous testing support, CI/CD ecosystem support, breakthrough features:
  - Selenium
  - Jenkins
  - Katalon Studio
  - Appium
  - Eggplant

The following diagram shows the continuous testing using selenium diagram features.



- Selenium supports Android, iOS, Windows, Linux, Mac, Solaris. Languages supported by Selenium include C#, Java, Python, PHP, Ruby, Perl, and JavaScript. Browsers supported by Selenium include Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc
- Selenium can be used to automate functional tests and can be integrated with automation test tools such as Maven, Jenkins, & Docker to achieve continuous testing. It can also be integrated with tools such as TestNG, & JUnit for managing test cases and generating reports. [2M]

**Selenium Tool Suite:**

- Selenium is not just a single tool but a suite of software, each with a different approach to support automation testing. It comprises of four major components which include:

1. Selenium Integrated Development Environment (IDE)
2. Selenium Remote Control (Now Deprecated)
3. WebDriver
4. Selenium Grid

**Selenium Integrated Development Environment (IDE)**

- Selenium IDE is implemented as Firefox extension which provides record and playback functionality on test scripts. It allows testers to export recorded scripts in many languages like HTML, Java, Ruby, RSpec, Python, C#, JUnit and TestNG. You can use these exported script in Selenium RC or Webdriver.

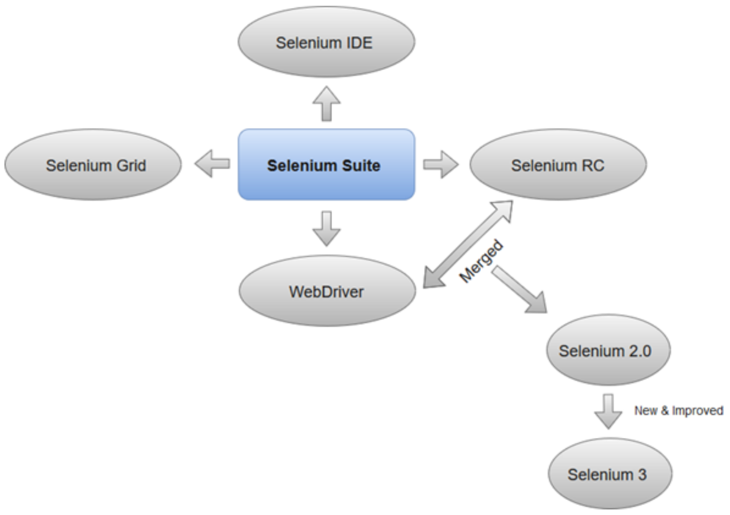
**Selenium Remote Control:**

- Selenium RC (officially deprecated by selenium)allows testers to write automated web application UI test in any of the supported programming languages. It also involves an HTTP proxy server which enables the browser to believe that the web application being tested comes from the domain provided by proxy server.

- Selenium RC Server (acts as a HTTP proxy for web requests).
- Selenium RC Client (library containing your programming language code).

**3. Selenium WebDriver**

- Selenium WebDriver (Selenium 2) is the successor to Selenium RC and is by far the most important component of Selenium Suite.
- Selenium WebDriver provides a programming interface to create and execute test cases. Test scripts are written in order to identify web elements on web pages and then desired actions are performed on those elements.
- Selenium WebDriver performs much faster as compared to Selenium RC because it makes direct calls to the web browsers. RC on the other hand needs an RC server to interact with the web browser.



[3M]

***(OR)***

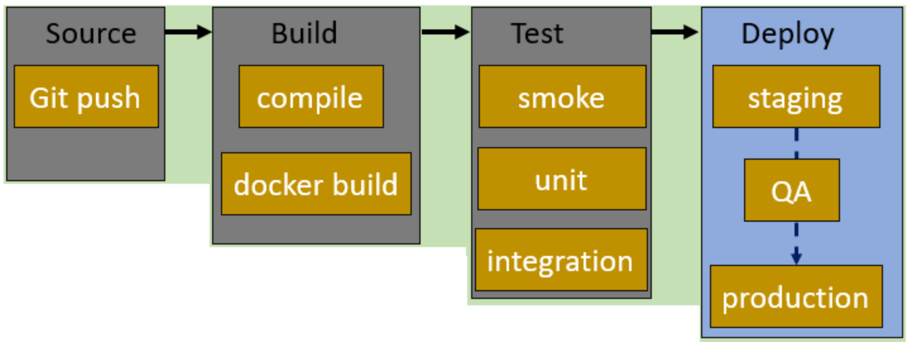
*7 a) **Explain in detail CI/CD Pipeline.*** ***CO3 L1 5M***

- A continuous integration/continuous delivery (CI/CD) pipeline is a framework that emphasizes iterative, reliable code delivery processes for agile DevOps teams. It involves a workflow encompassing continuous integration, testing, delivery, and continuous delivery/deployment practices. The pipeline arranges these methods into a unified process for developing high-quality software.
- A CI/CD pipeline automates the process of software delivery. It builds code, runs tests, and helps you to safely deploy a new version of the software.
- CI/CD pipeline reduces manual errors, provides feedback to developers, and allows fast product iterations.
- CI/CD pipeline introduces automation and continuous monitoring throughout the lifecycle of a software product.
- It involves from the integration and testing phase to delivery and deployment. These connected practices are referred as CI/CD pipeline.
- Continuous integration is a software development method where members of the team can integrate their work at least once a day. In this method, every integration is checked by an automated build to search the error.
- Continuous delivery is a software engineering method in which a team develops software products in a short cycle. It ensures that software can be easily released at any time. (manual)
- Continuous deployment (automated) is a software engineering process in which product functionalities are delivered using automatic deployment. It helps testers to validate whether the codebase changes are correct, and it is stable or not.
- 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.

[2M]

**CI/CD Stages:**

- 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. The following diagram shows various stages in CI/CD.



**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.

**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.

**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.

**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.

[3M]

***7. b) Compare CI/CD Tools.*** ***CO3 L2 5M***

***5 tools → 5 marks***

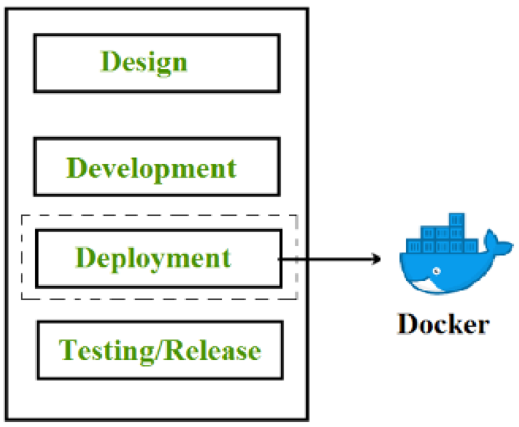
| | ***Execution Environments*** | ***Integrations & Plugins*** | ***Pros and Cons*** | ***Pricing*** |
|---|---|---|---|---|
| ***Jenkins*** | Windows<br>Mac<br>Linux<br>Docker<br>Kuberneters | Limitless due to 1800+ plugins | Open Source<br>Free of Charge<br>Platform agonsistic<br>Ease to setup<br>Poor IO | Free |
| ***TeamCity*** | Windows<br>Mac<br>Linux<br>Docker<br>Kuberneters | .NET tools,<br>Jira, Bugzilla.<br>Maven<br>Visual Studio<br>Main VSCs<br>400+ | Feature-rich<br>Ease to set-up<br>Great documentation<br>Sleep learning curve<br>Manual upgradng | Free trial ( 14 days) |
| ***Bamboo*** | Windows<br>Mac<br>Linux<br>Docker<br>Kuberneters | Jira<br>BitBucket<br>AWS CodeDepoly<br>185 plugins | Native integrations<br>Ease of deployment<br>Great Docs,<br>No Cloud version<br>High Price | Free Trial (30 days) |

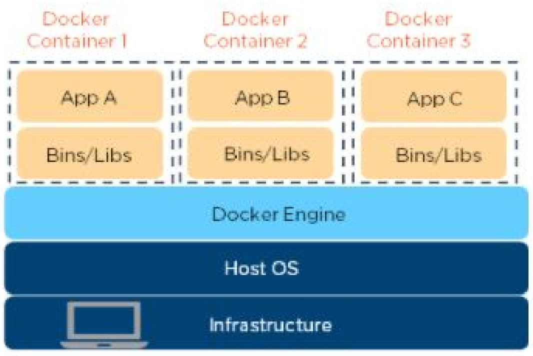
| | | | | |
|---|---|---|---|---|
| ***Tavis CI*** | Windows<br>Mac<br>Linux<br>Docker (Except for MacOsSD | GitHub<br>GitLab<br>BitBucket | 30+ languages support,<br>More deployment options<br>Limited plug ins<br>Lack of scalability | Free Trial (30 days) |
| ***Circle CI*** | Windows<br>Mac<br>Linux<br>Docker<br>MacOsSD | GitHub<br>BitBucket<br>AWS, Google, Azure<br>150 integrations' total | Pre-built configuration packages.<br>More tutorials, Custom services, Community support | Free Tier ( 6000 build minutes/month) |

## ***<u>Unit-IV</u>***

***8 a) Discuss in detail Containerization with Docker.*** ***CO4 L1 5M***

- Docker is an OS-level virtualization software platform that helps users in building and managing applications in the Docker environment with all its library dependencies.
- Docker is the world's leading software container platform. It was launched in 2013 by a company called Dotcloud, Inc which was later renamed Docker, Inc.
- It is written in the Go language.
- Docker is designed to benefit both developers and system administrators making it a part of many DevOps toolchains.
- Developers can write code without worrying about the testing and production environment.
- Sysadmins need not worry about infrastructure as Docker can easily scale up and scale down the number of systems. Docker comes into play at the deployment stage of the software development cycle.





***[2M]***

## Adv:

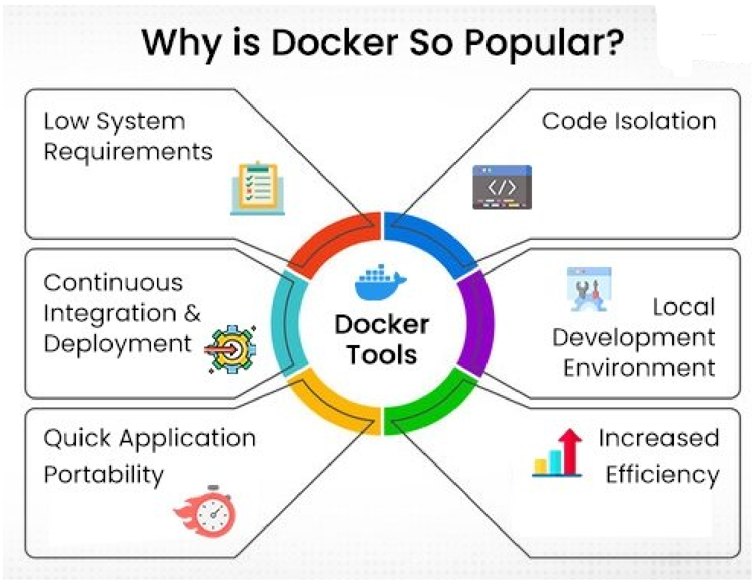
Enables integrated user interface to view and monitor Docker containers:

Quickly starts Docker within ten seconds

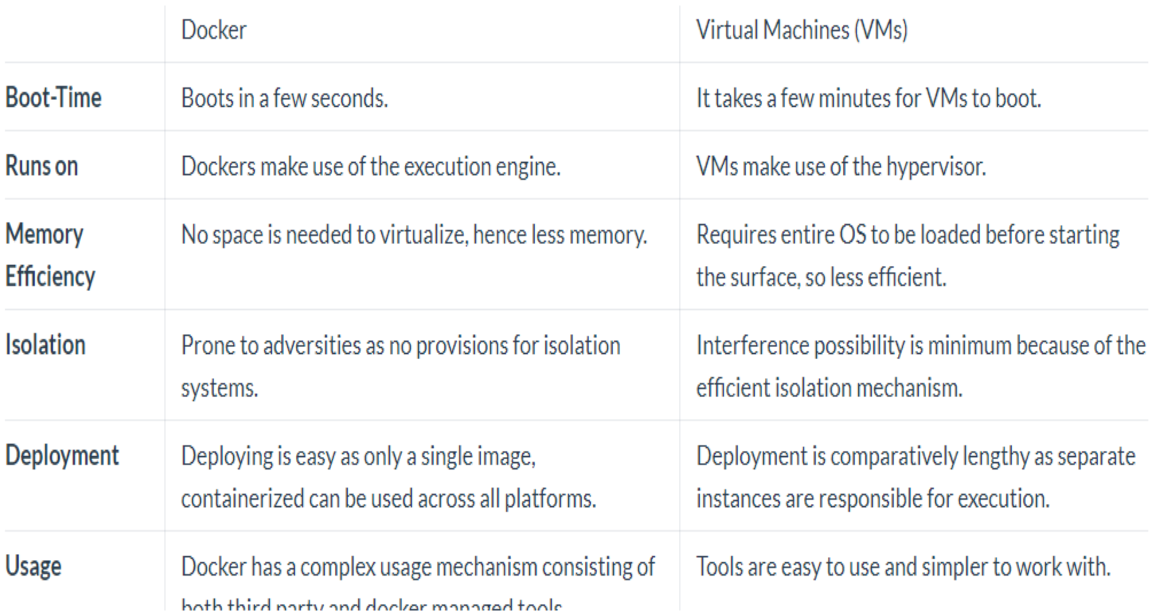
- Easy to use Linux Workspaces
- Allots required resource and memory space
- Consists of CA synchronization
- Supports HTTP proxy settings

**Elements required to create Docker Container:**

- Docker Engine: It runs on the host machine to build and run containers
- Docker Daemon: It manages Docker containers
- Docker Client: It runs commands. The command is translated using REST API and delivered to the Docker Daemon
- Docker Compose: It runs two containers in a single service


Why is Docker So Popular?
Low System Requirements
Code Isolation
Continuous Integration & Deployment
Docker Tools
Local Development Environment
Quick Application Portability
Increased Efficiency

**ocker vs VM:**



| | Docker | Virtual Machines (VMs) |
|---|---|---|
| Boot-Time | Boots in a few seconds. | It takes a few minutes for VMs to boot. |
| Runs on | Dockers make use of the execution engine. | VMs make use of the hypervisor. |
| Memory Efficiency | No space is needed to virtualize, hence less memory. | Requires entire OS to be loaded before starting the surface, so less efficient. |
| Isolation | Prone to adversities as no provisions for isolation systems. | Interference possibility is minimum because of the efficient isolation mechanism. |
| Deployment | Deploying is easy as only a single image, containerized can be used across all platforms. | Deployment is comparatively lengthy as separate instances are responsible for execution. |
| Usage | Docker has a complex usage mechanism consisting of both third party and docker managed tools | Tools are easy to use and simpler to work with. |

[3M]

***8. b) Compare Configuration Management Tools Puppet & Ansible.*** ***CO4 L2 5M***
***5 points → 5 marks***

| **Category** | **Puppet** | **Ansible** |
|---|---|---|
| Configuration Language | RUBY, Puppet DSL | Python, YAML |
| Architecture | Master-Agent | Only Master (Agentless) |
| Installation Process | Time Intensive due to master-agent certificate sign in. | Easy |
| Configuration Management | Pull | Push and Pull |
| Scalability | High | Very High |
| Interoperability | Puppet master works only on Linux | Server on Linux, Client on Windows, |
| Pricing | Medium | High |

***(OR)***

***9 a) Explain in detail about Continuous Deployment.*** ***CO4 L1 5M***

Continuous deployment is a software development strategy where a new code or a change is deployed directly to the production environment after going through a set of rigorous, automated tests. The changes in the software (after deployment) are visible to the end-users of the application.

Continuous Deployment (CD) is the continuation of Continuous Integration. Once the tests have been validated in the dev environment, they must be deployed to production. Continuous deployment, therefore, consists of automating deployment actions that were previously performed manually. This is why we often talk about CI/CD together.

here is, however, a major difference between the two. With continuous delivery, someone will need to approve the release of the feature to production but with continuous deployment, this process happens automatically upon passing automated testing.

Continuous deployment is part of a continuous process. The first step starts with continuous integration when developers merge their changes into the trunk or mainline on a frequent basis. This helps teams evade what is known as 'merge hell', which happens when developers attempt to merge several separate branches to the shared trunk on a less regular basis. Then comes continuous delivery when code is deployed to a testing or production environment. Here, developers' changes are uploaded to a repository, where they are then deployed to a production environment. With continuous delivery, you can decide to release daily, weekly or monthly but it is usually recommended to release as often as possible in small batches to be able to easily and quickly fix any issue that arises. [3M]

**Continuous deployment goes one step further and combines continuous integration and continuous delivery to make software automatically available to users without any human intervention.** Hence, continuous deployment requires the complete automation of all deployments and so refers to the process of automatically releasing developers' changes from the repository to production, bypassing the need for developer approval for each release. Consequently, this process relies heavily on well-designed test automation.

All these practices come together to facilitate and accelerate releases making deployment less risky.

***Continuous deployment tools are: Jenkins, Gitlab CI, Circle CI, AWS CodeDeploy***

***With the Continuous Deployment, the following are the advantages:***

- ✓ Fast Time to market
- ✓ Reduced market Risk
- ✓ Faster testing and bug fixing
- ✓ Create and efficient infrastructure

[2M]

***9. b) Discuss Continuous Monitoring with Nagios.*** ***CO4 L1 5M***

Continuous monitoring is a process to detect, report, respond all the attacks which occur in its infrastructure. Continuous monitoring starts when the deployment is done on the production servers. From then on, this stage is responsible to monitor everything happening. This stage is very crucial for the business productivity.

**Benefits of CM (Continuous Monitoring):**

- It detects all the server and network problems.
- It finds the root cause of the failure.
- It helps in reducing the maintenance cost.
- It helps in troubleshooting the performance issues.

- It helps in updating infrastructure before it gets outdated.
- It can fix problems automatically when detected.
- It makes sure the servers, services, applications, network is always up and running.
- It monitors complete infrastructure every second.

***Nagios:***

- Nagios is an open source continuous monitoring tool which monitors network, applications and servers. It can find and repair problems detected in the infrastructure, and stop future issues before they affect the end users. It gives the complete status of your IT infrastructure and its performance.
- Nagios is used for continuous monitoring of systems, applications, service and business process in a DevOps culture. **[2M]**

***Features :***

- It can monitor Database servers such as SQL Server, Oracle, Mysql, Postgres
- It gives application level information (Apache, Postfix, LDAP, Citrix etc.).
- Provides active development.
- Has excellent support from huge active community.
- Nagios runs on any operating system.
- It can ping to see if host is reachable.

***Benefits of Nagios:***

- It helps in getting rid of periodic testing.
- It detects split-second failures when the wrist strap is still in the "intermittent" stage.
- It reduces maintenance cost without sacrificing performance.
- It provides timely notification to the management of control and breakdown.
-

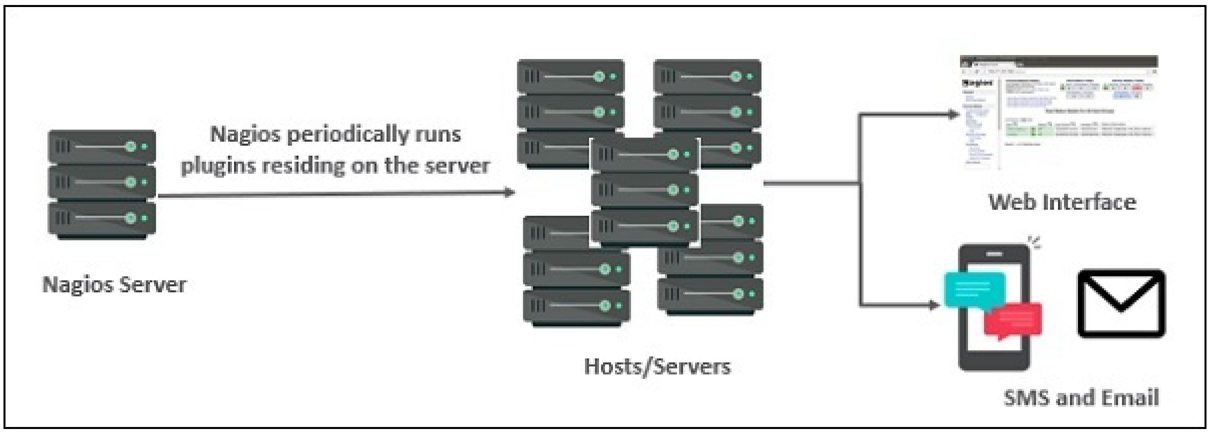
***Disadvantages of Using Nagios***

- Important feature like wizards or interactive dashboard are only available on Nagios XI, which is quite an expensive tool
- Nagios core has a confusing interface
- There're many configuration files which are very hard to configure for users
- Nagios can't monitor network throughput
- The tool not allows you to manage the network but only allows to monitor the network
- *Nagios makes no difference between various devices like servers, routers, or switches as it treats every device as a host*

***Negios Architecture:***

The following points are worth notable about Nagios architecture –

- Nagios has server-agent architecture.
- Nagios server is installed on the host and plugins are installed on the remote hosts/servers which are to be monitored.
- Nagios sends a signal through a process scheduler to run the plugins on the local/remote hosts/servers.
- Plugins collect the data (CPU usage, memory usage etc.) and sends it back to the scheduler.
- Then the process schedules send the notifications to the admin/s and updates Nagios GUI.



**Nagios Applications:**

- Nagios can be applicable to a wide range of applications. They are given here –

1. Monitor host resources such as disk space, system logs etc.
2. Monitor network resources – http, ftp, smtp, ssh etc.
3. Monitor log files continuously to identify infra-issue.
4. Monitor windows/linux/unix/web applications and its state.
5. Nagios Remote Plugin Executer (NRPE) can monitor services remotely.
6. Run service checks in parallel.
7. SSH or SSL tunnels can also be used for remote monitoring.
8. Send alerts/notifications
9. via email, sms, pager of any issue on infrastructure
10. Recommending when to upgrade the IT infrastructure. **[3M]**

**Signature of the Internal Examiner**

**Signature of the HOD**

**Signature of the External Examiner**