0UNIT-1 QB

1. What is DevOps? How is it different from Waterfall and Agile methodology?

DevOps is a culture different from traditional corporate cultures and requires a change in mindset, processes, and tools.

● It is often associated with continuous integration (CI) and continuous delivery (CD) practices, which are software engineering practices.

● It is also associated with Infrastructure as Code (IaC) which consists of codifying the structure and configuration of infrastructure.

● DevOps is a set of practices that combines software development and IT operations.

● It aims to shorten the systems development life cycle and provide continuous delivery with high software quality.

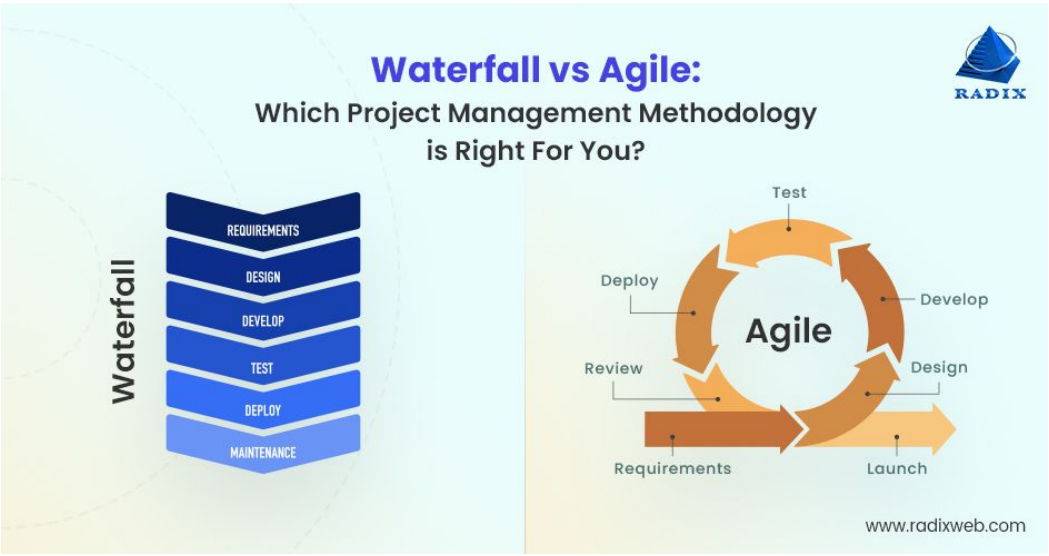
● The DevOps is the combination of two words Development+Operations

● Practice to promote the Development and Operation process Collectively.

AND

The main difference between DevOps and waterfall is the approach to software development. Agile focuses on continuous delivery of features, while waterfall focuses on delivering a complete product. DevOps uses an agile approach and emphasizes communication and collaboration between developers and operations teams.

DevOps also focuses on culture and people. It emphasizes collaboration between teams and a shared responsibility for the success of the project. Waterfall and agile tend to have siloed teams with responsibility for specific parts of the project.



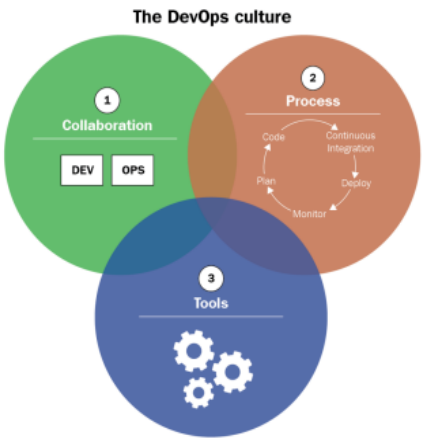
2. Explain the 3 Axis of DevOps Culture with a neat diagram.

● The DevOps movement is based on three axes:

The culture of collaboration: This is the very essence of DevOps—the fact that teams are no longer separated by specialization (developers, Ops, testers, and so on), but, on the contrary, these people are brought together by making multidisciplinary teams that have the same objective: to deliver added value to the product as quickly as possible

Processes: For rapid deployment, teams must follow development processes from agile methodologies with iterative phases that allow for better functionality quality and rapid feedback. These processes should not only be integrated into the development workflow with continuous integration but also into the deployment workflow with ***continuous delivery and deployment.***
The DevOps process is divided into several phases: ***The planning and prioritization of functionalities, Development, Continuous integration and delivery, Continuous deployment, Continuous monitoring.***

Tools: The choice of tools and products used by teams is very important in DevOps.
When teams were separated into Dev and Ops, each team used their specific tools—deployment tools for developers and infrastructure tools for Ops—which further widened communication gaps.
● Developers need to integrate with monitoring tools used by Ops teams to detect performance problems as early as possible and with security tools provided by Ops to protect access to various resources.
● Ops, on the other hand, must automate the creation and updating of the infrastructure and integrate the code into a code manager; this is called Infrastructure as Code, but this can only be done in collaboration with developers who know the infrastructure needed for applications.
● Ops must also be integrated into application release processes and tools.



3. List and explain benefits of establishing a DevOps culture.
The benefits of establishing a DevOps culture:

- Better collaboration and communication in teams.
- Shorter lead times to production, resulting in better performance and end user satisfaction.
- Reduced infrastructure costs with IaC.
- Significant time saved with iterative cycles that reduce application errors and automation tools that reduce manual tasks, so teams focus more on developing new functionalities with added business value.

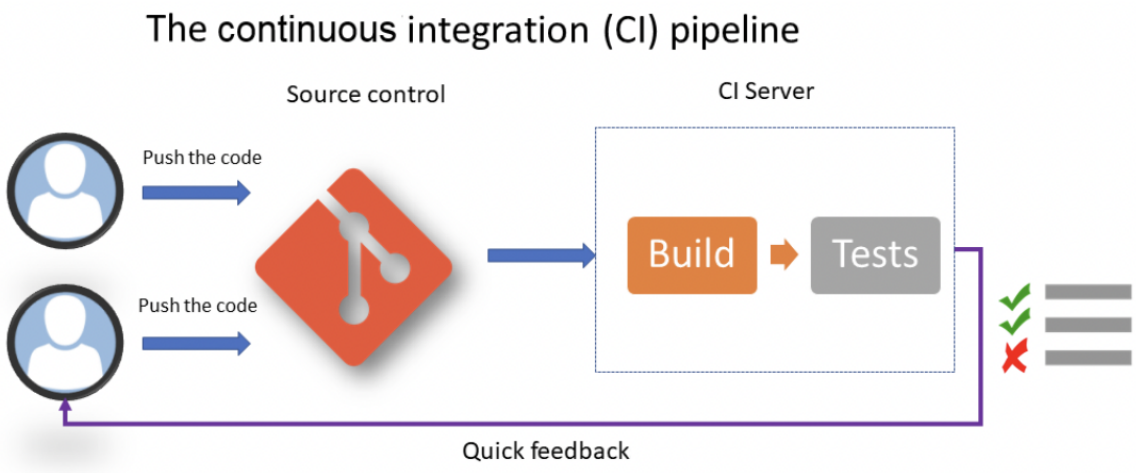
4. With a neat diagram explain the Continuous Integration (CI) in detail.

"Continuous Integration is a software development practice where members of a team integrate their work frequently. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.“

- CI is an automatic process that allows you to check the completeness of an application's code every time a team member makes a change.
- This verification must be done as quickly as possible.
- DevOps culture is seen in CI very clearly, with the spirit of collaboration and communication, because the execution of CI impacts all members in terms of work methodology and therefore collaboration.
- Moreover, CI requires the implementation of processes (branch, commit,pull request, code review, and so on) with automation that is done with tools adapted to the whole team (Git, Jenkins, Azure DevOps, and so on).
- CI must run quickly to collect feedback on code integration as soon as possible and hence be able to deliver new features more quickly to users.
- To set up CI, it is necessary to have a Source Code Manager (SCM) that will allow the centralization of the code.
- This code manager can be of any type: Git, SVN (Subversion Repository), or Team Foundation Source Control (TFVC).
- Git is a distributed version control system, whereas SVN is a centralized version control system.
- Next is Automatic build manager (CI server) which supports continuous integration such as Jenkins, GitLab CI, TeamCity, Azure Pipelines, GitHub Actions, Travis CI, Circle CI, and so on.

- Each team member will work on the application code daily, iteratively and incrementally.
- Each task or feature must be partitioned from other developments with the use of branches.
- Team members archive or commit their code and with small commits (trunks) that can easily be fixed in the event of an error.
- This will be integrated into the rest of the code of the application with all of the other commits of the other members.
- This integration of all the commits is the starting point of the CI process.
- This process, executed by the CI server, must be automated and triggered at each commit.
- The server will retrieve the code and then do the following:
    - Build the application package—compilation, file transformation, and so on.
    - Perform unit tests (with code coverage).

● This CI process must be optimized as soon as possible so that it can run fast and developers can have quick feedback on the integration of their code.
● Bad practices can result in the failure of tests in the CI.
● Such practice can have serious consequences when the errors detected by the tests are revealed in production.
● The time saved during CI will be lost on fixing errors with hotfixes and redeploying them quickly with stress.
● Instead of developing new features, we spend time correcting errors.

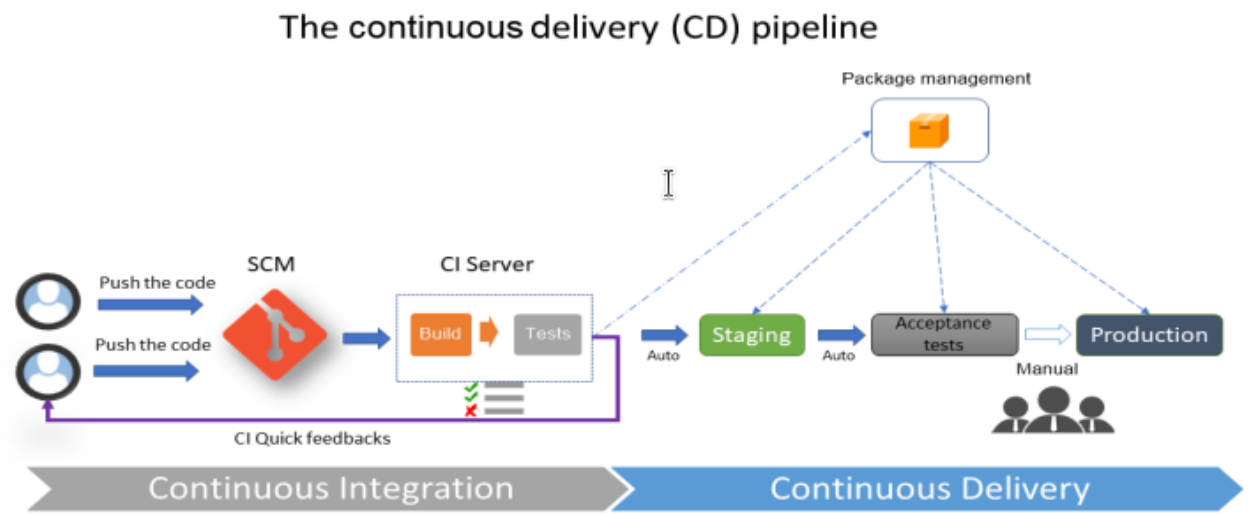


5. With a neat diagram explain the Continuous Delivery (CD) in detail.

● After CI, next step is to deploy the application automatically in one or more non-production environments, which is called staging.
● This process is called Continuous Delivery (CD).
● CD often starts with an application package prepared by CI, which will be installed according to a list of automated tasks.
● These tasks can be of any type: unzip, stop and restart service, copy files, replace configuration, and so on.
● The execution of functional and acceptance tests can also be performed during the CD process.
● CD aims to test the entire application with all of its dependencies.
● This is very visible in microservices applications composed of several services and APIs.
● CI will only test the microservice under development while, once deployed in a staging environment, it will be possible to test and validate the entire application as well as the APIs and microservices that it is composed of.
● In practice, it is common to link CI with CD in an integration environment; that is, CI deploys at the same time in an environment.
● Developers need to have the following at each commit
    - Execution of unit tests

    - Verification of the application as a whole (UI and functional), with the integration of the developments of the other team members.

● Important that the package generated during CI, deployed during CD is the same one that will be installed on all environments until production.
● There may be configuration file transformations that differ depending on the environment, but the application code (binaries, DLL, and JAR) must remain unchanged.
● If changes (improvements or bug fixes) are to be made to the code following verification in one of the environments, once done, the modification will have to go through the CI and CD cycle again.
● Tools set up for CI/CD:
    - A package manager: This constitutes the storage space of the packages generated by CI and recovered by CD. Must support feeds, versioning, and different types of packages. Tools in the market, such as Nexus, ProGet, Artifactory, and Azure Artifacts.
    - A configuration manager: This allows you to manage configuration changes during CD. Most CD tools include a configuration mechanism with a system of variables.
● In CD, the deployment of the application in each staging environment is triggered as follows:
● Triggered Automatically, following a successful execution on a previous environment.
● For example: Deployment in the pre-production environment is automatically triggered when the integration tests have been successfully performed in a dedicated environment.
● Triggered Manually, for sensitive environments such as the production environment, following a manual approval by a person responsible for validating the proper functioning of the application in an environment.
● What is important in a CD process is that the deployment to the production environment, that is, to the end user, is triggered manually by approved users.
● The CD process is a continuation of the CI process.
● The chain of CD steps are automatic for staging environments but manual for production deployments.
● Package is generated by CI and is stored in a package manager.
● The same package is deployed in different environments.



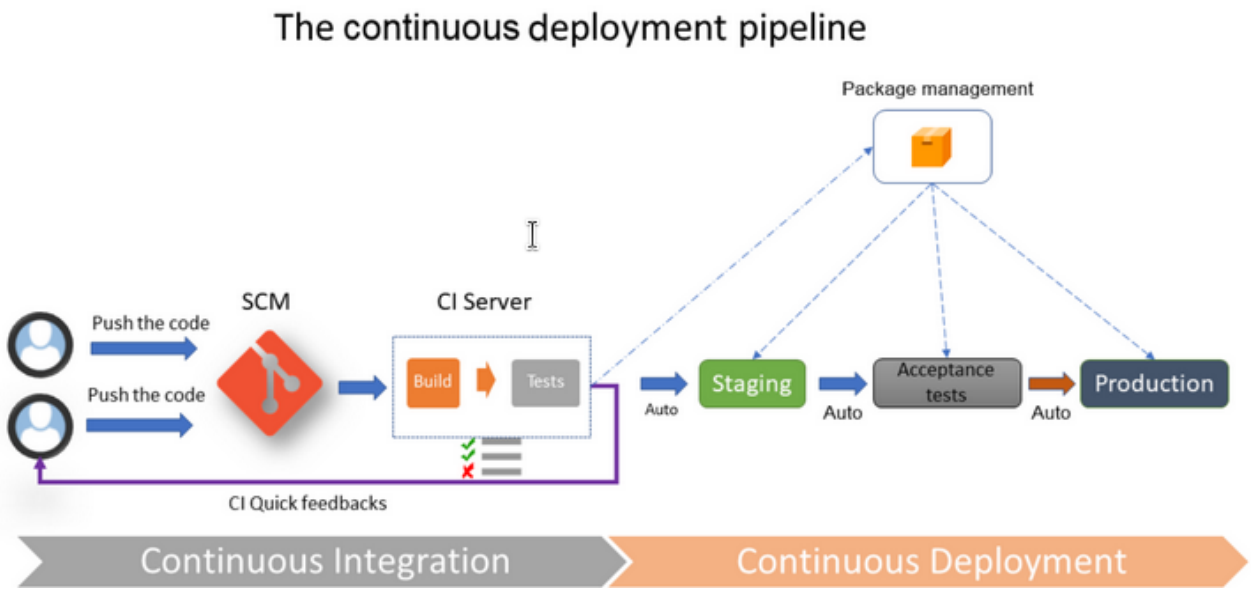
6. What is Continuous Deployment? How is it different from Continuous Delivery?

● Continuous deployment is an extension of CD, but with a process that automates the entire CI/CD pipeline from the moment the developer commits their code to deployment in production through all of the verification steps.

● This practice is rarely implemented because it requires a wide coverage of tests (unit, functional, integration, performance, and so on).

● Successful execution of these tests is sufficient to validate the proper functioning of the application with all of these dependencies.

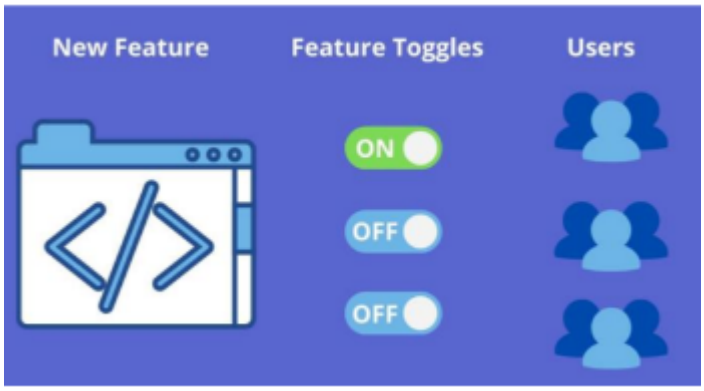
● The continuous deployment process must also take into account all of the steps to restore the application in the event of a production problem.



7. With a neat diagram explain the different techniques of implementing Continuous Deployment.

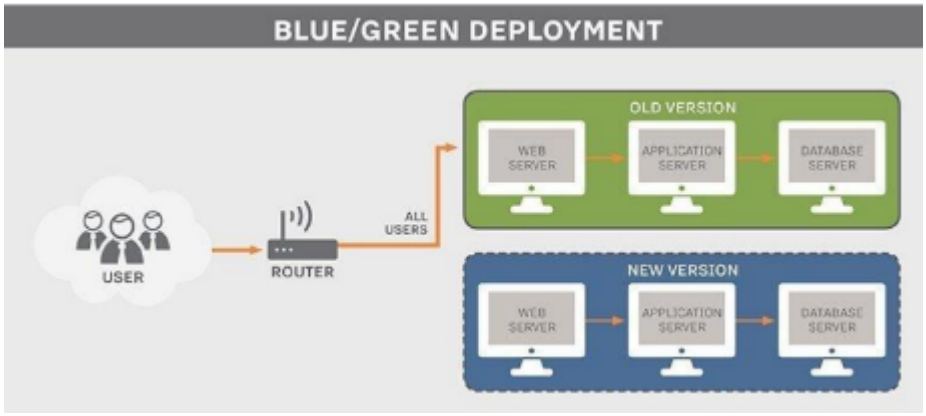
● Continuous deployment can be implemented with the use and implementation of Feature Toggle techniques (or feature flags)

- Encapsulating the application's functionalities in features.
- Activating its features on demand directly in production.
- Code of the application won't be redeployed.



● Another technique is to use a Blue-Green production infrastructure.

- Consists of two production environments, one blue and one green.
- First deploy to the blue environment, then to the green.
- To ensure that there is no downtime required.



8. What are IaC Practices? List the benefits of IaC Practices.

● IaC is the process of writing the code for provisioning and configuration of infrastructure components to automate its deployment in a repeatable and consistent manner.

● Practice began due to the rise of DevOps culture and with the modernization of cloud infrastructure.

● Ops teams that deploy infrastructures manually take time to deliver infrastructure changes due to inconsistent handling and the risk of errors.

● With the modernization of the cloud and its scalability, the way an infrastructure is built requires a review of provisioning and changing practices by adapting a more automated method.

● The benefits of IaC

- The standardization of infrastructure configuration reduces the risk of error.
- The code that describes the infrastructure is versioned and controlled in a source code manager.
- The code is integrated into CI/CD pipelines.
- Deployments that make infrastructure changes are faster and more efficient.
- There's better management, control, and a reduction in infrastructure costs.

9. Describe different IaC Languages used in DevOps. Explain the same with simple code snippet.

Scripting types: These are scripts such as Bash, PowerShell, or any other languages that use the different clients (SDKs) provided by the cloud Provider.

● Ex: Command that creates a resource group in Azure:

- Using the Azure CLI:
- az group create -location westeurope -name MyAppResourcegroup
- Using Azure PowerShell:
- New-AzResourceGroup -Name MyAppResourcegroup -Location westeurope

Declarative types: These are languages in which it is sufficient to write the state of the desired system or infrastructure in the form of configuration and properties.

```
resource "azurerm_resource_group" "myrg" {
      name = "MyAppResourceGroup"
      location = "West Europe"

      tags = {
          environment = "Bookdemo"
      }
}
```

10. List and explain The IaC Topologies. (study from ppt)

- The deployment and provisioning of the infrastructure (Setting up servers from scratch)
- The server configuration and templating (modifying server resources and features)
- The containerization (deploying an app on the server as containers)
- The configuration and deployment in Kubernetes

11. Describe the IaC best practices followed in DevOps pipeline.

- **Everything** must be **automated** in the **code**
- The **code** must be **in** a **source control manager**
- The infrastructure code must be with the application code
- **Separation** of **roles** and **directories**
- Integration into a **CI/CD process**
- The code must be idempotent
- To be used as documentation
- The code must be modular
- Having a development environment

12. What is Ansible? List and explain the reasons for using it in DevOps.

Ansible is an open source automation and orchestration tool for software provisioning, **configuration management**, and software deployment. Can easily run and configure Unix-like systems as well as Windows systems to provide infrastructure as code.Contains its own declarative programming language for system configuration and management.

Why use Ansible?

- Free to use.
- No need for any special system administrator skills to install and use Ansible.
- Its modularity regarding plugins, modules, inventories, and playbooks make Ansible useful for orchestrating large environments.
- Lightweight, consistent, and no constraints regarding the OS or underlying hardware.
- Secure due to its agentless capabilities and use of OpenSSH security features.
- It has a template engine and a vault to encrypt/decrypt sensitive data.
- Comprehensive documentation and easy to learn structure and configuration.

13. What are Pull & Push Configuration Tools? With a neat diagram explain them.

Pull Based Configuration Management Tool
In this type of configuration management tool, the nodes pull the configuration information from the server (hence, the name).

A small software (called agent or client) is installed on every node. This agent/client will:

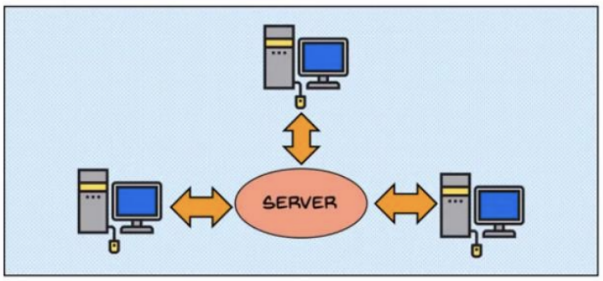
- at regular intervals, get the configuration from the server
- compare the configuration received from the server with the current configuration of the node
- if there is any mis-match, take the steps required to match the configuration of the node with the configuration received from the server.
- This means that, its always the agent/client that initiates communication, not the main server.

Chef & Puppet are good examples of such configuration management tools.

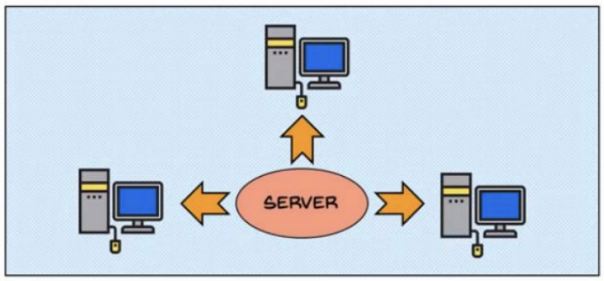
Push Based Configuration Management Tool
In this type of configuration management tool, the main server (where the configuration data is stored) pushes the configuration to the node (hence, the name). So, it is the main server that initiates communication, not the nodes. Which means that an agent/client may or may not be installed on each node.

Ansible is an example of a push based configuration management tool that doesn't need an agent to be installed on the nodes. SaltStack is an example of a push based configuration management tool that needs an agent (minion) to be installed on the nodes. In both cases, it's the main server that starts the communication and sends the configuration data to the nodes without the nodes asking for it.

Pull & Push Configuration Tool



Pull configuration: Nodes check with the server periodically and fetch the configurations from it

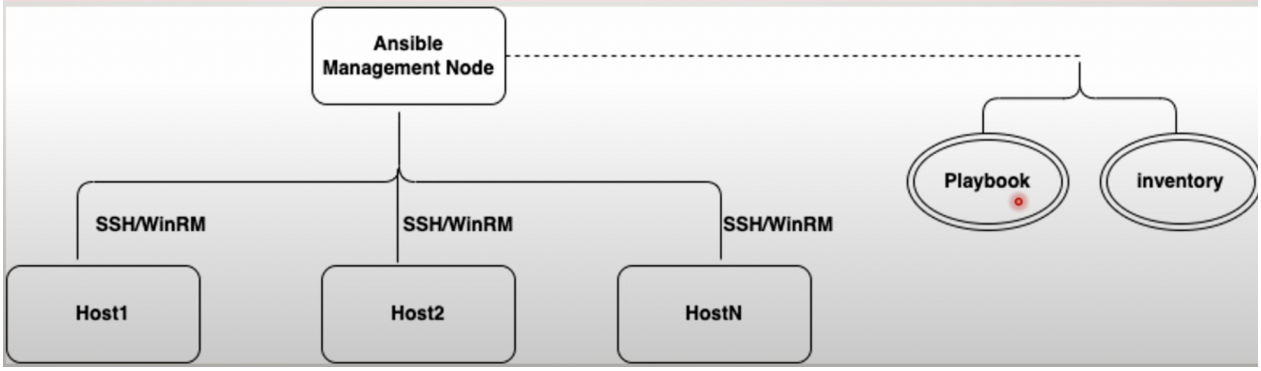


Push configuration: Server pushes configuration to the nodes

14. List and explain the important terms used in Ansible.

● **Ansible server**: The machine where Ansible is installed and from which all tasks and playbooks will be run.
● **Module**: It is a command or set of similar Ansible commands meant to be executed on the client-side.
● **Task**: A task is a section that consists of a single procedure to be completed.
● **Role**: A way of organizing tasks and related files to be later called in a Playbook.
● **Playbook**: List of instructions, written in YAML, to configure clients.
● **Inventory**: File containing data about the ansible client servers. Defined in examples as hosts file.
● **Play**: Execution of a playbook.
● **Handler**: Task which is called only if a notifier is present.
● **Notifier**: Section attributed to a task which calls a handler if the output is changed.
● **Tag:** Name set to a task which can be used later on to issue just that specific task or group of tasks.

15. Explain the working of Ansible Workflow with a neat diagram. (study



- The Management Node is the controlling node that controls the entire execution of the playbook.
- The inventory file provides the list of hosts where the Ansible modules need to be run.
- The Management Node makes an SSH connection and executes the small modules on the host's machine and install the software.
- Ansible removes the modules once those are installed.
- It connects to the host machine executes the instructions, and if it is successfully installed, then remove that code.

16. List and explain the benefits of using Ansible.

- Agentless
- Simple
- Efficient
- Flexible
- Idempotent
- Automated reporting

17. What is the need for using an inventory file in Ansible? Explain the different types of inventories used in Ansible.

The inventory contains the list of hosts on which Ansible will perform administration and configuration actions.

- Dynamic inventory:
  - The list of hosts is dynamically generated by an external script (Ex: With a Python script).
  - The dynamic inventory is used when the addresses of the hosts are not available.
- Static inventory:
  - Hosts are listed in a text file in INI (or YAML) format.
  - This is the basic mode of Ansible inventory.
  - The static inventory is used in cases where we know the host addresses

18. What is the purpose of using a Playbook in the Ansible? Explain with a sample code snippet.

● Playbook is one of the essential elements of Ansible.
● It contains the code of the actions or tasks that need to be performed to configure or administer a VM.
● Once the VM is provisioned, it must be configured, with the installation of all of the middleware needed to run the applications that will be hosted on this VM.
● Necessary to perform administrative tasks concerning the configuration of directories and their access.

```
  GNU nano 4.8                                    /etc/ansible/example_playbook.yaml
---
- hosts: all

  vars:
    names:
      - Alex
      - Bob
      - Charlie

  tasks:
  - name: using variables in ansible
    debug:
      msg: "{{ names[2] }}"
```

19. With an example explain how you improve the Playbooks with Roles in Ansible.

- Most configurations have common pieces of code.
- To avoid this code duplication create a directory called “roles” and copy-paste all the code into this directory
- This code can be access in the playbook by including the “roles:” attribute in the YAML file

```
---
- hosts: webserver
  roles:
    - php
    - apache
- hosts: database
  roles:
    - mysql
```

20. How to use variables in Ansible for better configuration? Explain.

● Using variables in Ansible for better configuration
● The different tasks of this role are as follows:

1. Updating packages
2. The installation of the MySQL server and Python MySQL packages
3. The creation of a MySQL user

```
- hosts: database
become: true
roles:
  - mysql
```

21. What is Ansible Vault? What is use of using the vault?(study from ppt)

Ansible Vault is a feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles.

- create : to create ansible vault file in the encrypted format
- view: to view data of encrypted file
- edit: to edit encrypted file
- encrypt : to encrypt an unencrypted file
- decrypt: to decrypt an encrypted file

22. What are the concerns with Automating configuration with Ansible? How does the Packer tool help to overcome this?

Packer is an open-source VM image creation tool from Hashicorp. It helps you automate the process of Virtual machine image creation on the cloud and on-premise virtualized environments. All the manual steps performed to create a Virtual machine image can be automated through a simple Packer config template.

concerns with Automating configuration with Ansible:

- Time consuming to create VM image
- Middleware versions might not match on all environments
- Security issues might appear

The benefits of using these images are as follows:

- The provisioning of a VM from an image is very fast.
- Each VM is uniform in configuration.
- It is safety compliant

23. Explain the following sections used in Packer templates in detail.

i. The builders section

The builders section is **mandatory** and contains all of the properties that define the image and its location, such as:

- Name

The variables section:

- Type of image
- Cloud provider on which the image will be generated
- Connection information to the cloud
- Base image to use
- And other properties that are specific to the image type.

ii. The provisioners section

This is an **optional** section and contains a list of **scripts** that will be executed by Packer on a temporary VM base image in order to build a custom VM image. If the Packer template does not contain a provisioners section, no configuration will be made on the base images.

24. What is the need to create a generalized VM image using Packer?

- Generalizing removes machine specific information so the image can be used to create multiple VMs.
- It is easier to configure these VM images with multiple VMs

OR -

Creating a generalized VM image can be useful for a number of reasons. Some benefits include:

1. Reduced setup time: By creating a pre-configured VM image, you can deploy new virtual machines more quickly and easily. This can save time and effort compared to manually installing and configuring the operating system and all necessary software on each individual VM.
2. Consistency: A generalized VM image ensures that all of your virtual machines are configured in the same way, which can be helpful for maintaining consistency across your infrastructure.
3. Reusable: A generalized VM image can be used as a starting point for multiple virtual machines, which can be helpful if you need to deploy a large number of VMs with similar configurations.
4. Version control: By using a tool like Packer to create your VM images, you can version control your images and track changes over time. This can be helpful for managing and maintaining your infrastructure.

Overall, creating a generalized VM image can help you to streamline the process of deploying and managing virtual machines, and can make it easier to maintain a consistent and reliable infrastructure.

25. What are the uses of Variables in Packer template? Demonstrate how to use them in different sections with code snippets.

In the Packer template, you need to use values that are not static in the code.
● This section is optional.

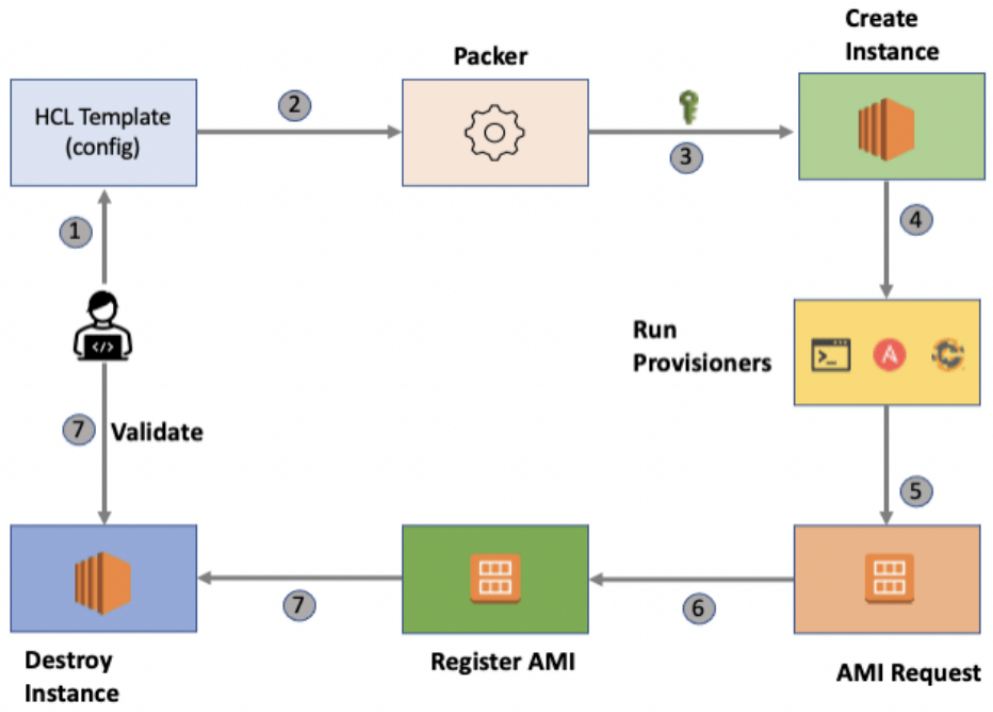
● Used to define variables that will be filled either
  - as command-line arguments
  - as environment variables.

● These variables will then be used in the builders or provisioners sections.

The access_key variable with the ACCESS_KEY environment variable The image_folder variable with the /image value The value of the VM image size, which is the vm_size variable

```
{
 "variables": {
     "access_key": "{{env `ACCESS_KEY`}}",
     "image_folder": "/image",
     "vm_size": "Standard_DS2_v2"
 },
....
}
```

26. Explain how to build a VM image using Packer with a neat diagram.



1. Declare all the required VM configurations in an HCL (Hashicorp configuration language) or a JSON file. Let's call it the Packer template.
2. To build the VM image, execute Packer with the Packer template.
3. Packer authenticates the remote cloud provider and launches a server. If you execute Packer from a cloud environment, it leverages the cloud service account for authentication.
4. Packer takes a remote connection to the server (SSH or Winrm).

5. Then it configures the server based on the provisioner you specified in the Packer template (Shell script, Ansible, Chef, etc).
6. Registers the AMI
7. Deletes the running instance.

UNIT-2 QB

1. What is a Version Control System? Explain the benefits of VCS.

Version control systems are software tools that help software teams manage changes to source code over time. The system records all the changes made to a file so a specific version may be rolled if required in the future.

Benefits:

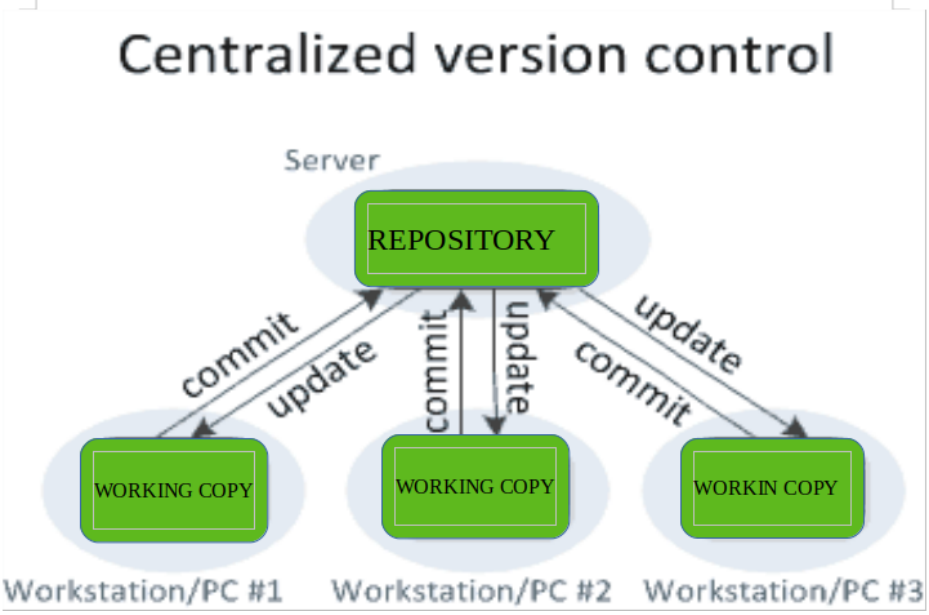
- Enhances the project development speed by providing efficient collaboration,
- Leverages the productivity, expedites product delivery, and skills of the employees through better communication and assistance,
- Reduce possibilities of errors and conflicts meanwhile project development through traceability to every small change,
- Employees or contributors of the project can contribute from anywhere irrespective of the different geographical locations through this VCS,
- For each different contributor to the project, a different working copy is maintained and not merged to the main file unless the working copy is validated. The most popular example is Git, Helix core, Microsoft TFS,
- Helps in recovery in case of any disaster or contingent situation,
- Informs us about Who, What, When, Why changes have been made.

2. What are the different types of Version Control System? Explain with a neat diagram.

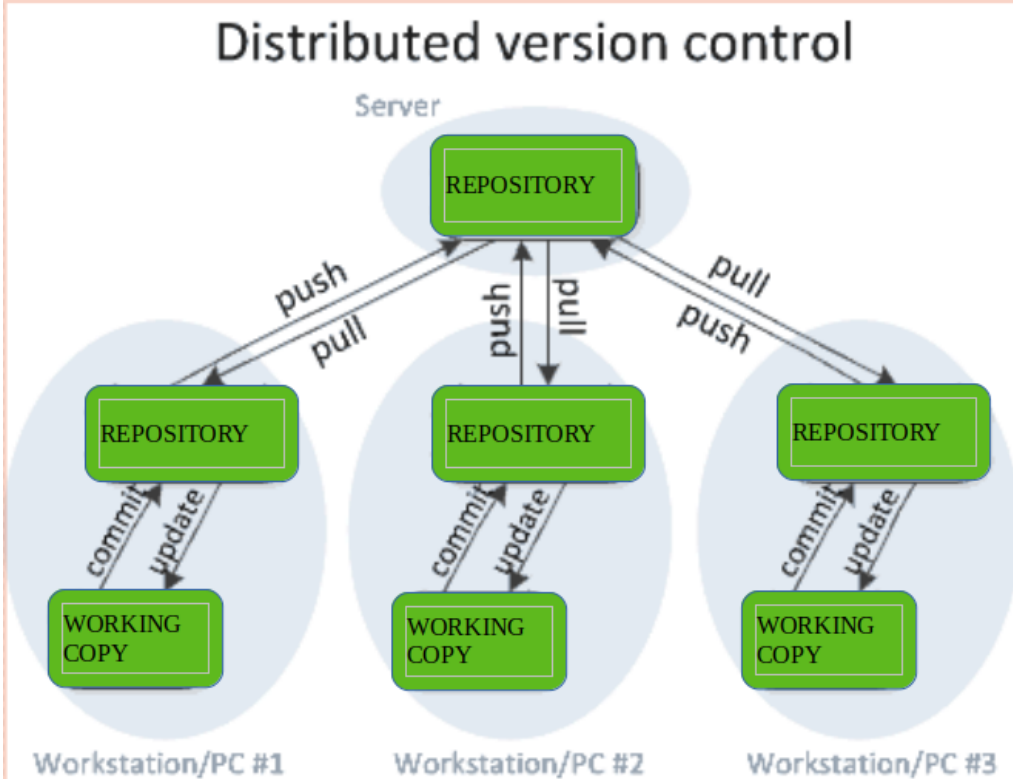
There are two types of VCS:

- Centralized Version Control System (CVCS): With centralized version control systems, you have a single “central” copy of your project on a server and commit your changes to this central copy.

svn



- Distributed Version Control System (DVCS): With distributed version control systems (DVCS), you don't rely on a central server to store all the versions of a project's files. Instead, you clone a copy of a repository locally so that you have the full history of the project. Two common distributed version control systems are Git and Mercurial.



-

3. List and explain the different Git vocabulary used.

- **Initialize** - The git init creates an empty Git repository or re-initializes an existing one.
- **Add** - The git add command adds new or changed files in your working directory
- **Status** - The git status command lists all the modified files which are ready to be added to the local repository.
- **Commit** - The "commit" command is used to save your changes to the local repository.
- **Pull** - The git pull origin main command syncs all the files from remote repo to local repo.
- **Push -** upload local repos 0710itory content to a remote repository
- **Branching -** creating branches to make independent changes
- **Merging -** procedure to connect the forked history. It joins two or more development history together
- **Rebasing**

Git Definitions and Terminology Cheat Sheet | A Cloud Guru

4. Explain the following operations with commands.

i. Retrieving a remote repository

clone and fetch download remote code from a repository's remote URL to your local computer

```
$ git clone https://github.com/USERNAME/REPOSITORY.git
```

```
$ git fetch REMOTE-NAME
```

ii. Initializing a local repository

The git init creates an empty Git repository or re-initializes an existing one. It basically creates a .git directory with sub directories and template files.

iii. Configuring a local repository

The git config command is a convenience function that is used to set Git configuration values on a global or local project level. These configuration levels correspond to .gitconfig text files.

$ git config --global user.email "your_email@example.com"

iv. Adding a file for the next commit

This command updates the index using the current content found in the working tree and then prepares the content in the staging area for the next commit.

$ git add .

v. Creating a commit

git commit: This will commit the staged snapshot.
$ git commit –a –m “Message”

vi. Updating the remote repository

Push the files: git push origin main

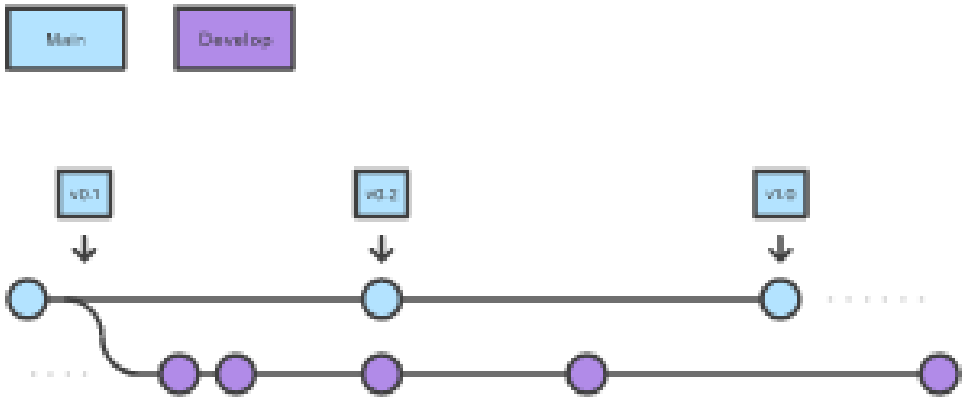
5. Explain the advanced GIT commands with examples.`

When you have two branches in a project (e.g. a development branch and a master branch), both of which have changes that need to be combined, the `git merge` command is the natural and straightforward way to unify them.

- **Merge** : A `merge` adds the development history of one branch as a merge commit to the other. While this preserves both histories in complete detail, it can make the overall history of the project difficult to follow. In some cases, you might want a simpler and cleaner result.
- **Rebase** : The `git rebase` command also merges two branches, but does it a little differently. A `git rebase` rewrites the commit history of one branch so that the other branch is incor``porated into it from the point where it was created.

- **Branch** : In Git, a branch is a new/separate version of the main repository.
- **Git bisect - This command uses a binary search algorithm to find which commit in your project's history introduced a bug**. You use it by first telling it a "bad" commit that is known to contain the bug, and a "good" commit that is known to be before the bug was introduced. Then `git bisect` picks a commit between those two endpoints and asks you whether the selected commit is "good" or "bad". It continues narrowing down the range until it finds the exact commit that introduced the change.

6. With a neat diagram explain the GitFlow pattern in detail.



- Gitflow is an alternative Git branching model that involves the use of feature branches and multiple primary branches.
- Gitflow has numerous, longer-lived branches and larger commits.
- Developers create a feature branch and delay merging it to the main trunk branch until the feature is complete
- These long-lived feature branches require more collaboration to merge and have a higher risk of deviating from the trunk branch.
- They can also introduce conflicting updates.

7. What is the work of a CI Server? Explain the different types of CI Servers. Explain with examples.

CI is achieved by an automatic task suite that is executed on a server, following similar patterns executed on a developer's laptop that has the necessary tools for continuous integration; this server is called the CI server.

- CI servers (also known as build servers) automatically compile, build, and test every new version of code committed to the central team repository.
- CI server ensures that the entire team is alerted any time the central code repository contains broken code.

The type of CI servers are :

1. **On-premise type -** installed in the company data center such as Jenkins or TeamCity
2. **Cloud type** - such as Azure Pipelines or GitLab CI.

8. What is Continuous Delivery (CD)? Explain the different tools used for setting up a CI/CD pipeline.

Once the application has been packaged and stored in a package manager during CI, the Continuous Delivery process is ready to retrieve the package and deploy it in different environments except production env

The different tools for setting up a CI/CD pipeline are as follows:

1. A source control version
2. A package manager
3. A CI server
4. A configuration manager - to modify the configuration of the application in the generated package in order for it to be adapted to the target environment.

9. What are Azure Artifacts? List the advantages.

Azure Artifacts enables developers to share their code efficiently and manage all their packages from one place. With Azure Artifacts, developers can publish packages to their feeds and share it within the same team, across organizations, and even publicly. Developers can also consume packages from different feeds and public registries such as NuGet.org or npmjs.com. Azure Artifacts supports multiple package types such as NuGet, npm, Python, Maven, and Universal Packages.

Advantages:

- It is fully integrated with other Azure DevOps services such as Azure Pipelines, which allows managing CI/CD pipelines.
- In Azure Artifacts, there is also a type of package called universal packages that allows storing all types of files (called a package) in a feed that can be consumed by other services or users.
- Azure Artifacts is in SaaS offering mode, so there is no installation or infrastructure to manage.

10. What is Jenkins? Why is it very popular in DevOps?

Jenkins is one of the oldest continuous integration tools, initially released in 2011. It is an open-source automation tool written in Java with plugins built for continuous integration.

Jenkins is used to **build and test your software projects continuously** making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.
• Jenkins has become famous due to the large community working on it and its plugins.
• More than 1,500 Jenkins plugins.
• If one of your tasks does not have a plugin, then you can create it yourse**lf.**

11. Illustrate the working of Jenkins with a neat diagram.
Illustrate the working of Jenkins with a neat diagram.

- Developers commit changes to the source code, found in the repository.
- The Jenkins CI server checks the repository at regular intervals and pulls any newly available code.
- The Build Server builds the code into an executable file. In case the build fails, feedback is sent to the developers.
- Jenkins deploys the build application to the test server. If the test fails, the developers are alerted.
- If the code is error-free, the tested application is deployed on the production server.

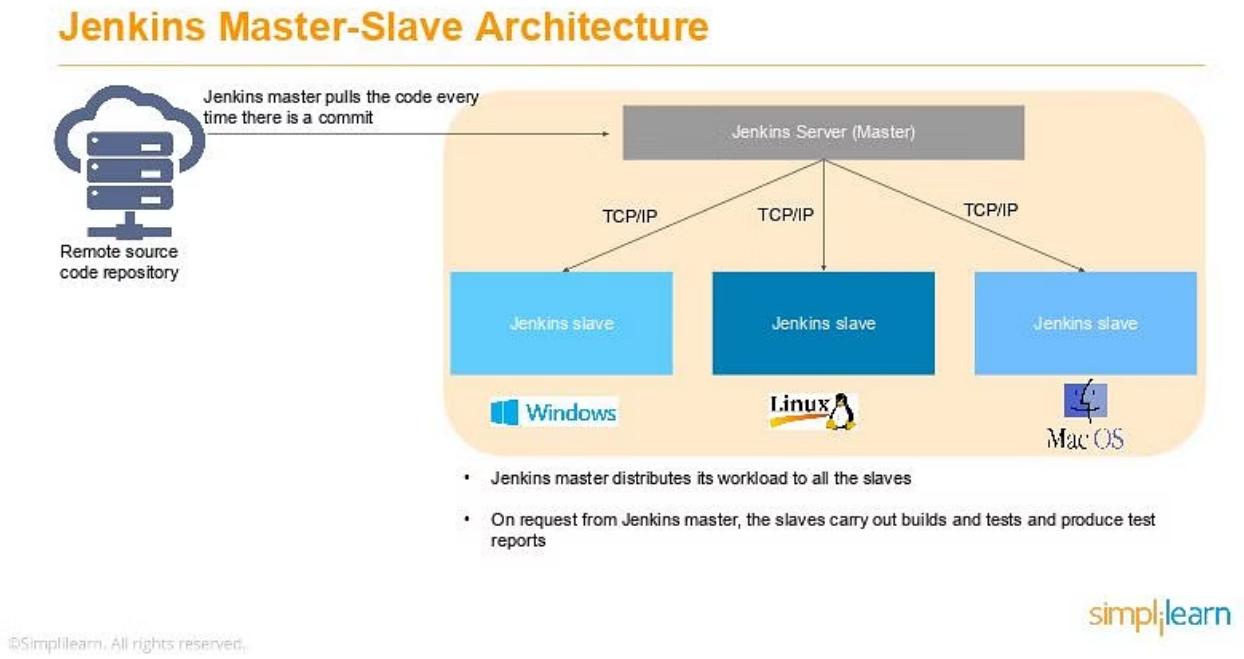
The files can contain different code and be very large, requiring multiple builds. However, a single Jenkins server cannot handle multiple files and builds simultaneously; for that, a distributed Jenkins architecture is necessary.

The Jenkins server accesses the master environment on the left side and the master environment can push down to multiple other Jenkins Slave environments to distribute the workload.

That lets you run multiple builds, tests, and product environments across the entire architecture. Jenkins Slaves can be running different build versions of the code for different operating systems and the server Master controls how each of the builds operates.

Supported on a master-slave architecture, Jenkins comprises many slaves working for a master. This architecture - the Jenkins Distributed Build - can run identical test cases in

different environments. Results are collected and combined on the master node for monitoring.



12. What are the Jenkins Features? Explain.

- Easy Installation
  Jenkins is a platform-agnostic, self-contained Java-based program, ready to run with packages for Windows, Mac OS, and Unix-like operating systems.
- Easy Configuration
  Jenkins is easily set up and configured using its web interface, featuring error checks and a built-in help function.
- Available Plugins
  There are hundreds of plugins available in the Update Center, integrating with every tool in the CI and CD toolchain.
- Extensible
  Jenkins can be extended by means of its plugin architecture, providing nearly endless possibilities for what it can do.

- Easy Distribution
  Jenkins can easily distribute work across multiple machines for faster builds, tests, and deployments across multiple platforms.
- Free Open Source
  Jenkins is an open-source resource backed by heavy community support

13. List and explain the advantages of Jenkins.

- 1. Free to Use
  - Jenkins is fully open-source and free to use. Since its development in 2011, it is the most preferred CI/CD tool used by developers in both early-stage startups and big organizations.
- 2. Rich Plugin Ecosystem
  - There are over a thousand different plugins that can be used to enhance the functionality of a Jenkins environment and suit the specific needs of an organization.
- 3. Easy Integration
  - Jenkins can be easily integrated with a number of popular cloud platforms such as Google Cloud, Digital Ocean, Amazon EC2 and more.

14. Explain the following
i. Azure Repos
ii. Azure Boards
iii. Azure Pipelines
iv. Azure Artifacts
v. Azure Test Plans

Service Name	Description
Azure Repos	It is a Source Code Versioning System.
Azure Boards	It is a service for project management in agile mode with sprints, backlogs, and boards.
Azure Pipelines	It is a service that allows the management of CI/CD pipelines.
Azure Artifacts	It is a private package manager.
Azure Test Plans	It allows you to make and manage a manual test plan.

Azure Repos is a set of version control tools that you can use to manage your code. (write about version control system)

Azure Repos provides two types of version control:

- Git: distributed version control
- Team Foundation Version Control (TFVC): centralized version control

Azure Boards is designed to support software development processes through the default process models selected for a project.
Azure Boards provides software development teams with the interactive and customizable tools they need to manage their software projects. It provides a rich set of capabilities including native support for Agile, Scrum, and Kanban processes, calendar views, configurable dashboards, and integrated reporting. These tools scale as your business grows.

Azure Pipelines automatically builds and tests code projects.
Azure Artifacts enables developers to share their code efficiently and manage all their packages from one place.(explained earlier)
Azure Test Plans include creating test plans, test cases, run manual tests, and explanatory tests for every person involved in the software development process. Features include -

- Test on any platform
- Rich diagnostic data collection
- End to End traceability
- Integrated analytics
- Extensible platform

15. Describe the five tasks used in the Azure CI pipeline.

Step/task	**Description**
Restore	Restores the packages referenced in the project.
Build	Builds the project and generates binaries.
Test	Runs unit tests.
Publish	Creates a ZIP package that contains the binary files of the project.
Publish Build Artifacts	Defines an artifact that is our ZIP of the application, which we will publish in Azure DevOps, and which will be used in the deployment release, as seen in the previous, Use package manager, section.