**What is Git?**

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

**Features of Git**

1. Tracks history
2. Free and open source
3. Supports non-linear development
4. Creates backups
5. Scalable
6. Supports collaboration
7. Branching is easier
8. Distributed development

**Git Workflow**

The Git workflow is divided into three states:

* **Working directory** - Modify files in your working directory
* **Staging area (Index)** - Stage the files and add snapshots of them to your staging area
* **Git directory (Repository)** - Perform a commit that stores the snapshots permanently to your Git directory. Checkout any existing version, make changes, stage them and commit.

**Branch in Git**

Branch in Git is *used to keep your changes until they are ready.* You can do your work on a branch while the main branch (master) remains stable. After you are done with your work, you can merge it with the main office.

***Commands***

* **git init**

This command turns a directory into an empty Git repository. This is the first step in creating a repository.

* **git add**

Adds files in the to the staging area for Git. Before a file is available to commit to a repository, the file needs to be added to the Git index (staging area).

* **git commit**

Record the changes made to the files to a local repository. For easy reference, each commit has a unique ID.

* **git status**

This command returns the current state of the repository.

git status will return the current working branch. If a file is in the staging area, but not committed, it shows with git status.

* **git config**

With Git, there are many configurations and settings possible. git config is how to assign these settings. Two important settings are user user.name and user.email. These values set what email address and name commits will be from on a local computer.

* **git branch**

To determine what branch the local repository is on, add a new branch, or delete a branch.

* **git checkout**

To start working in a different branch, use git checkout to switch branches.

* **git merge**

Integrate branches together. git merge combines the changes from one branch to another branch.

* **git clone**

To create a local working copy of an existing remote repository, use git clone to copy and download the repository to a computer.

* **git pull**

To get the latest version of a repository run git pull. This pulls the changes from the remote repository to the local computer.

* **git push**

Sends local commits to the remote repository. git push requires two parameters: the remote repository and the branch

**What is DevOps?**

*DevOps is a culture that allows the development and operations teams to work together.* With this type of working environment, developers continuously develop and test codes, and there is continuous integration taking place throughout the lifecycle. The operations team continuously deploys the code to the production environment. *DevOps allows for better collaboration, increased trust, and faster software releases.*

**Tools in DevOps**

**Subversion (SVN), TFS,** and **Git** are source code management or software configuration management tools (SCM) used for both the planning and coding of the software.

**Subversion (SVN):**

1. SVN is a *centralized version control system*
2. It is distributed under the open-source license
3. It allows you to recover older versions of your data or examine how your data changes

**TFS:**

1. TFS is an application lifecycle management solution that *tracks issues and manages documents*
2. It has an open-source license
3. It enables you to recover older versions of your data or examine how your data changes

**Git:**

1. Git is a *distributed version control tool*
2. It is distributed under the GNU license
3. Git is used for maintaining historical and current versions of source code, web pages, etc

**Version Control System (VCS)**

1. VCS enables you to *track the history of a file collection.*
2. Each version captures a snapshot of the files at a certain point in time, and the VCS allows you to switch between these versions.

**Centralized Version Control System:**

1. Uses a *central server to store all the files*
2. Every operation is performed directly on the repository
3. All the versions of the file are stored on the Central VCS server
4. In case the central server crashes, the entire data of the project will be lost. Hence, distributed VCS was introduced.

**Distributed Version Control System:**

1. Every programmer has a copy of all the versions of the code on their local systems
2. Distributed VCS moves from the client-server approach of central VCS to a peer-to-peer approach
3. They can update their local repositories with new data from the central server and changes are reflected in the principal repository
4. Git is one such distributed VCS tool

**What is a CI/CD pipeline?**

*The primary goal of a CI/CD pipeline is to automate the software development lifecycle (SDLC).*

The pipeline will cover many aspects of a software development process, from writing the code and running tests to delivery and deployment. Simply stated, a CI/CD pipeline integrates automation and continuous monitoring into the development lifecycle. This kind of pipeline, which encompasses all the stages of the software development life cycle and connects each stage, is collectively called a CI/CD pipeline.

**Stages in a CI/CD pipeline**

A CI/CD pipeline can be divided into ***four main stages***:

1. **Source**
2. **Build**
3. **Test**
4. **Deployment**

**Source stage:**

This is the first stage of any CI/CD pipeline. In this stage, the CI/CD pipeline will get **triggered by any change** in the program or a preconfigured flag in the code repository (repo). This stage **focuses on source control**, covering version control and tracking changes.

**Build stage:**

This second stage of the pipeline combines the source code with all its dependencies to aexecutable/runnable instance of the development.

**This stage covers:**

1. Software builds
2. Other kinds of buildable objects, such as Docker containers
3. This stage is the most important one. Failure in a build here could indicate a fundamental issue in the underlying code.

**Test stage:**

The test stage incorporates all the automated testing to validate the behavior of the software. The goal of this stage is to **prevent software bugs from reaching end-users**. Multiple types of testing from integration testing to functional testing can be incorporated into this stage. This stage will also expose any errors with the product.

**Deploy stage:**

This is the final stage of the pipeline. After passing all the previous stages, the **package is now ready to be deployed**. In this stage, the package is deployed to proper environments as first to a staging environment for further quality assurance (QA) and then to a production environment.

**Introduction DOCKER**

Docker tool is *basically used to create, deploy, and run the applications by using containers.* Through Docker DevOps, developers can pack all parts of an application like libraries and other dependencies easily and ship it out as a single package.

**What is Docker?**

It is an **open-source tool** and anyone can use it to meet his or her needs even if they need any additional feature or in case if they need to add a feature that does not exist already in the application.

**What is Docker used for in DevOps?**

*Docker has a “run” option through which a container can be created and run.* Container life is bounded by the process of life that means as soon as the process will be finished, containers will also get terminated. Following command can help you in knowing that what commands are available in Docker:

Following are the **commands** which are being covered:

1. **docker –version**

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

2. **docker pull**

Usage: docker pull <image name>

*This command is used to pull images from the docker repository*

3. **docker run**

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

*This command is used to create a container from an image*

4. **docker ps**

*This command is used to list the running containers*

5. **docker ps -a**

T*his command is used to show all the running and exited containers*

6. **docker exec**

Usage: docker exec -it <container id> bash

*This command is used to access the running container*

7. **docker stop**

Usage: docker stop <container id>

*This command stops a running container*

8. **docker kill**

Usage: docker kill <container id>

*This command kills the container by stopping its execution immediately*. The difference between ‘docker kill’ and ‘docker stop’ is that ‘docker stop’ gives the container time to shutdown gracefully, in situations when it is taking too much time for getting the container to stop, one can opt to kill it

9. **docker commit**

Usage: docker commit <conatainer id> <username/imagename>

*This command creates a new image of an edited container on the local system*

10. **docker login**

*This command is used to login to the docker hub repository*

11. **docker push**

Usage: docker push <username/image name>

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

12. **docker images**

*This command lists all the locally stored docker images*

13. **docker rm**

Usage: docker rm <container id>

*This command is used to delete a stopped container*

14. **docker rmi**

Usage: docker rmi <image-id>

*This command is used to delete an image from local storage*

15. **docker build**

Usage: docker build <path to docker file>

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

**Jenkins** is an **open source continuous integration/continuous delivery and deployment (CI/CD)** automation software DevOps tool written in the Java programming language. It is used to implement CI/CD workflows, called pipelines.

**Advantages of Jenkins include:**

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

**Difference between git clone and git push**

git clone is how you get a local copy of an existing repository to work on.

git pull (or git fetch + git merge) is how you update that local copy with new commits from the remote repository.

**What is Git**

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.

**What is GitHub**

GitHub is a *web-based interface* that uses Git, the open source version control software that lets multiple people make separate changes to web pages at the same time.

**Use of git push command**

The git push command is used to **upload local repository** content to a remote repository.

**Use of git pull command**

The git pull command is used to **fetch and download content** from a remote repository and immediately update the local repository to match that content*.*