**1) How to check if Git is available on your System ?**

To check if Git is available on your system, you can follow these steps:

1) Open a terminal or command prompt.

2) Run the command git --version. If Git is installed on your system, you will see the version number of Git displayed in the terminal. For example, "git version 2.30.1".

3)If you see an error message indicating that the Git command is not found, then Git is not installed on your system.

**Note:** If you are using a Mac or Linux system, Git is usually pre-installed, but if you are using a Windows system, you may need to download and install Git from the official Git website.

**2) How to initialize a new Git repository ?**

Run the command git init to initialize the repository. This command creates a new .git directory in the current directory, which contains all the necessary Git configuration files and objects.

**3) How to tell Git about your name and G-mail ?**

To tell Git about your name and email, you need to follow these steps:

1) Open a terminal or command prompt.

2) Navigate to the directory that contains the Git repository you want to configure.

3) Run the command git config --global user.name "Your Name" to set your name in Git. Replace "Your Name" with your actual name.

4) Run the command git config --global user.email "your.email@example.com" to set your email in Git. Replace "your.email@example.com" with your actual email address.

5) You can confirm that the name and email have been set correctly by running the command git config --list. This command displays a list of all the Git configuration settings, including the user.name and user.email settings.

**Note:** The --global option in the Git config commands sets the configuration globally for all repositories on your system. If you want to set the configuration only for the current repository, you can omit the --global option.

**4) How to add the file to the staging area ?**

1) To add multiple files, you can run the command git add <file\_1> <file\_2> ... <file\_n>. Replace <file\_1>, <file\_2>, ..., <file\_n> with the names of the files you want to add.

2) Alternatively, you can run the command git add . to add all the modified and untracked files in the repository to the staging area.

**5) How to remove the file from staging area ?**

To remove a file from the Git staging area, you need to follow these steps:

1) Open a terminal or command prompt.

2) Navigate to the directory that contains the Git repository you want to remove files from.

3) Run the command git reset HEAD <file\_name> to remove a single file from the staging area. Replace <file\_name> with the actual name of the file you want to remove.

4) To remove multiple files, you can run the command git reset HEAD <file\_1> <file\_2> ... <file\_n>. Replace <file\_1>, <file\_2>, ..., <file\_n> with the names of the files you want to remove.

5) Alternatively, you can run the command git reset HEAD . to remove all the files from the staging area.

6) You can use the command git status to check the status of your repository and see which files are staged and which files have been removed from the staging area.

**Note: The HEAD in the Git reset command refers to the latest commit in the repository. By resetting the HEAD pointer, you are effectively removing the files from the staging area and returning them to the same state they were in before they were staged**

6) How to make a commit ?

To make a commit in Git, you need to follow these steps:

1) Open a terminal or command prompt.

2) Navigate to the directory that contains the Git repository you want to make a commit in.

3) Run the command git status to check the status of your repository and see which files are staged and ready to be committed.

4) If you want to include all the staged files in the commit, run the command git commit -m "commit message". Replace "commit message" with a descriptive message that explains what changes you made in this commit.

5) The -m option in the Git commit command is used to specify the commit message. This message is used to describe the changes you made in this commit and to help you and others understand what was changed.

6) You can use the command git log to view the history of the commits in your repository.

**Note:** Commits are an important part of the Git workflow, as they allow you to save and track the changes you make to your code over time. Each commit represents a snapshot of the code at a specific point in time, and you can use Git to revert to a previous commit if needed.

7) How to send your changes to a remote repository ?

To send your changes to a remote repository in Git, you need to follow these steps:

1) Open a terminal or command prompt.

2) Navigate to the directory that contains the Git repository you want to send changes to.

3) Make sure your repository is connected to a remote repository. You can use the command git remote -v to check the list of remote repositories associated with your local repository. If you haven't connected to a remote repository, use the command git remote add origin <remote\_repo\_url> to add a new remote repository, where <remote\_repo\_url> is the URL of the remote repository you want to connect to.

4) Run the command git push origin <branch\_name> to send your changes to the remote repository. Replace <branch\_name> with the name of the branch you want to push to. For example, if you want to push changes to the master branch, you would run git push origin master.

5) If you are pushing changes for the first time, you might be prompted to enter your username and password for the remote repository.

6) You can use the command git log to view the history of the commits in your repository, including the ones you just pushed to the remote repository.

**Note:** Pushing changes to a remote repository is a crucial step in the Git workflow, as it allows you to share your changes with others and collaborate on a project. It's also a good idea to regularly push your changes to a remote repository as a backup in case your local repository gets damaged or lost.

**8) What is the difference between clone and pull ?**

The main difference between clone and pull in Git is that clone is used to create a new local repository from a remote repository, while pull is used to update an existing local repository with changes from a remote repository.

**1) clone:** The clone command is used to create a new local repository that is a copy of a remote repository. This is typically done when you want to start working on a project that is stored on a remote repository. When you run the clone command, Git will download the entire history of the remote repository to your local machine, including all the branches and tags.

**2) pull:** The pull command is used to update an existing local repository with changes from a remote repository. This is typically done when you want to sync your local repository with the remote repository. When you run the pull command, Git will download the latest changes from the remote repository and merge them into your local repository.

In summary, clone is used to create a new local repository from a remote repository, while pull is used to update an existing local repository with changes from a remote repository.