GIT :::

- Git is Open-Source Distributed Version Control System
- Used to Version Control the Source Code Changes
- Used to Track the Source Code Changes
- Used to perform Parallel Development(Using Branching Techiques)

git config :::

Local Configuration :
- It is applicable with in a specific repository

```
git config user.name "Loksai"
git config user.email "Loksai@adsfasd.com"
```

Global Configuration :
- It is applicable to all the repositories

```
git config --global user.name "Loksai"
git config --global user.email "Loksai@adsfasd.com"
```

Working Dir ::: ---git add----> Staging_Area ---->git commit---> Local_Repo ---Pushed---> Remote Repo.

git log :::

```
git log

git log -2

git log --oneline

git log --oneline -2

git log --stat -1
```

git show <commit_id> # Show the Commit Details

Remove the Changes from Staging Area :::

```
git rm --cached <file_name>     # Just Unstage
                                # The Change will be moved back to working directory

git rm -f <file_name>           # Just Unstage
                                # The Change will be permanentlt
```

removed from staging area as well as from working directory

git ls-files   # Is a git Command to get the list of files/Folders being tracke
dby git.

Reset Options ::

                        git reset --soft <previous_Commit_ID> :::
                                - git reset will reset the HEAD pointer to the
previous commit point.
                                - It will take the changes back to staging area
                                - The Changes will be available in staging area and
working directory

                        git reset --mixed <previous_Commit_ID> :::
                                - git reset will reset the HEAD pointer to the
previous commit point.
                                - It will take the changes back to working
directory
                                - The Changes will be available only in working
directory

                        git reset --hard <previous_Commit_ID> :::
                                - git reset will reset the HEAD pointer to the
previous commit point.
                                - It will permanently delete the files from the
system

git revert :::

                - Git Revert is same as git reset --hard option
                - git revert is used to undo a specific commit
                - git revert will create a new commit point for tracking purpose.

                - git revert will maintain the commit history without deleting the
commit of a specific commit
                - git revert is recommended in shared repositories

                Syntax ::
                        git revert <specific_Commit_ID>

GIT Branching Techniques ::::

                In Distributed VCS, Branches are used to perform parallel
Development

Default Branch - master - is considered as a production version of source code.

Create Branch

```
git branch
# Used get the list of branches

git branch <new_branch_name>            # Create New Branch

git switch -c <new_branch_name>         # Create New Branch

git switch <branch_name>                        # Switch to any existing branch

git merge
# Used to Merge the Changes to Target Branch

# Should be executed from the target branch
git merge feature1
```

How Merge Conflict Occurs ::

- When more than one user/feature try to update a same line in the same file on the same target branch, merge Conflict Occurs.

How to fix the Merge Conflict ::

1. Identify the file(s) causing the Merge Conflict
2. Open and review the file content
3. Upon review, decide which record should be retained or deleted from that file
4. Open the File in edit mode, and delete the header and footer lines, and update the file with required records and save it.
5. perform git add and commit to the target branch

Prevent Merge Conflict ::

As per DevOps Process, review the Changes before merge

- git rebase :::

- Rebase is used to maintain linear commit history
- Rebase is used to keep the current branch in-sync with target branch
- Rebase can prevent merge-conflicts in the target branch
- As a best practise, it is always recommended to use rebase before merge

Difference Between Merging and Rebasing

Git Merge	Git Rebase
Git Merge merges two branches to create a “feature” branch.	Git Rebase rebases the feature branch to add the feature branch to the main branch.
Git Merge is comparatively easy.	Git Rebase is comparatively harder.
Git Merge safeguards history.	Git Rabse doesn’t safeguard history.
Git Merge is more suitable for projects with the less active main branch.	Git Rebase is suitable for projects with frequently active main branches.
Git Merge forms a chain-like structure.	Git Rebase forms a linear structure.
Git Merge is preferable for large no. of people working on a project.	Git Rebase is preferable for small groups of people.
Single line command is:	Single line command is:
git merge feature main	git rebase main

- git stash
    - It used to save the uncommitted changes to the temporary area.

```
git stash save "msg"            # Create an entry in the stash list

git stash list                          # Get the list of item from the stash list

git stash apply                         # Apply/Copy the Latest entry from stash list back to staging area
git stash apply stash@{2}       # Apply/Copy a specific entry from stash list back to staging area

git stash pop                           # move the Latest entry
```

```
from stash list back to staging area
                git stash pop stash@{2}         # move a specific entry from stash
list back to staging area

                git stash drop                          # Delete the Latest entry
from stash list
                git stash drop stash@{2}        # Delete a specific  entry from
stash list

                git stash clear                         # Clean-up / Delete all the
entries from stash list


Remote Repository Handling :

                github - Remote Repository Server !

                Azure Repos

                AWS CodeCommit

                gitlab

                bitbucket


GIT Cli Commands :::

                - git clone                     # To Copy/Clone the remote
repository to local machine

                - git add                               # To add the changes from
Working Directory to Staging Area

                - git commit                    # To Commit the changes from
Staging to Local Repository

                - git push                              # To Push the Changes from
Local Repository to Remote Repository

                - git fetch/git pull ::

                        --> Both Git Fetch and Git Pull Commands are used to handle
the incremental changes from Remote Repository


git remote -v                                   # Get the list of remote
repositories linked to the local repository
```

```
git push -u origin <branch_name>



                git remote add origin
https://github.com/SA-AWS-DevOps-July24/testrepo2.git                    # add
remote repo to local repo

                git remote remove origin
# Remove the remote repo from local repo



git clone:
        Purpose: Creates a local copy of an existing remote repository.
        Usage: git clone <repository-url>
        Functionality:
        Copies all files, branches, and commit history from the remote repository
to your local machine.
        Automatically sets up a remote named origin pointing to the cloned
repository.
        Example: git clone https://github.com/user/repo.git


git remote add
        Purpose: Adds a new remote repository to an existing local repository.
        Usage: git remote add <name> <repository-url>
        Functionality:
        Associates a remote repository with a name (e.g., origin, upstream).
        Allows you to push and pull changes to and from the remote repository.
        Example: git remote add origin https://github.com/user/repo.git
Key Differences
        Initialization: git clone initializes a new local repository by copying an
existing one, while git remote add is used to link an existing local
repository to a remote one.
        Setup: git clone sets up the remote automatically, whereas git remote add
requires you to manually specify the remote repository and its name.
        Use Case: Use git clone when you want to start working on a project by
copying it to your local machine. Use git remote add when you want to add a
  new remote to an already existing local repository


Git Branching Features :

                - git rebase
                - git merge --squash
                - git stash

git diff                                  Shows the changes between the working
directory and the staging area (index).
```

```
git diff <commit1> <commit2>            Displays the differences between two
commits.
git diff –staged or git diff –cached    Displays the changes between the staging
area (index) and the last commit.


git branch -a   Lists all local and remote branches.
git branch -r   Lists all remote branches.

git merge <branch>      Merges the specified branch into the current branch.
git log                 Displays the commit history of the current branch.
git log <branch-d       Displays the commit history of the specified branch.
git log –all            Displays the commit history of all branches.

git fetch                       Retrieves change from a remote repository,
including new branches and commit.
git fetch <remote>              Retrieves change from the specified remote
repository.
git fetch –prune                Removes any remote-tracking branches that no longer
exist on the remote repository.
git pull                        Fetches changes from the remote repository and
merges them into the current branch.
git pull <remote>               Fetches changes from the specified remote
repository and merges them into the current branch.
git pull –rebase                The git pull --rebase command is used to fetch
changes from a remote repository and reapply your local changes on top of the
                fetched changes, rather than merging them. This helps maintain a
cleaner, linear project history by avoiding unnecessary
merge commits.
git push                        Pushes local commits to the remote repository.
git push <remote>               Pushes local commits to the specified remote
repository.
git push <remote> <branch>      Pushes local commits to the specified branch of the
remote repository.
git push –all                   Pushes all branches to the remote repository.
git remote                      Lists all remote repositories.
git remote add <name> <url>     Adds a new remote repository with the specified
name and URL.
git remote remove <name>

git show                        Shows the details of a specific commit, including
its changes.
git show <commit>               Shows the details of the specified commit,
including its changes.


git revert <commit>             Creates a new commit that undoes the changes
introduced by the specified commit.
git revert –no-commit <commit>  Undoes the changes introduced by the specified
commit, but does not create a new commit.
```

```
 git checkout -b news
 echo jaya>ok ok
 git add .
 git commit -m "ok"
 git checkout main
 git push origin news  --> it push the news branch into remote repository


 git checkout main
 git diff news main             -->it display the difference between these two
branches
 git merge news                 -->current local repository(main) merges with news
repository
 git diff news                  -->it is also display the difference between news
and main branches
 git push origin --delete news  -->it delete the branch in remote repository
 git branch -d news             -->it delete the branch in local repository
 touch jayajaya
 git add jayajaya
 git commit -m "jaya"
 git push origin main
```

In Git, “origin” is a conventional name used to refer to the default remote repository from which a project was originally cloned. When you clone a repository, Git automatically names the remote repository as “origin” to make it easier to reference. This setup allows you to interact with the remote repository without needing to remember or repeatedly type its full URL12.

Here are some common commands involving “origin”:

Fetching updates: git fetch origin
Pushing changes: git push origin <branch_name>
Pulling updates: git pull origin <branch_name>