

Git

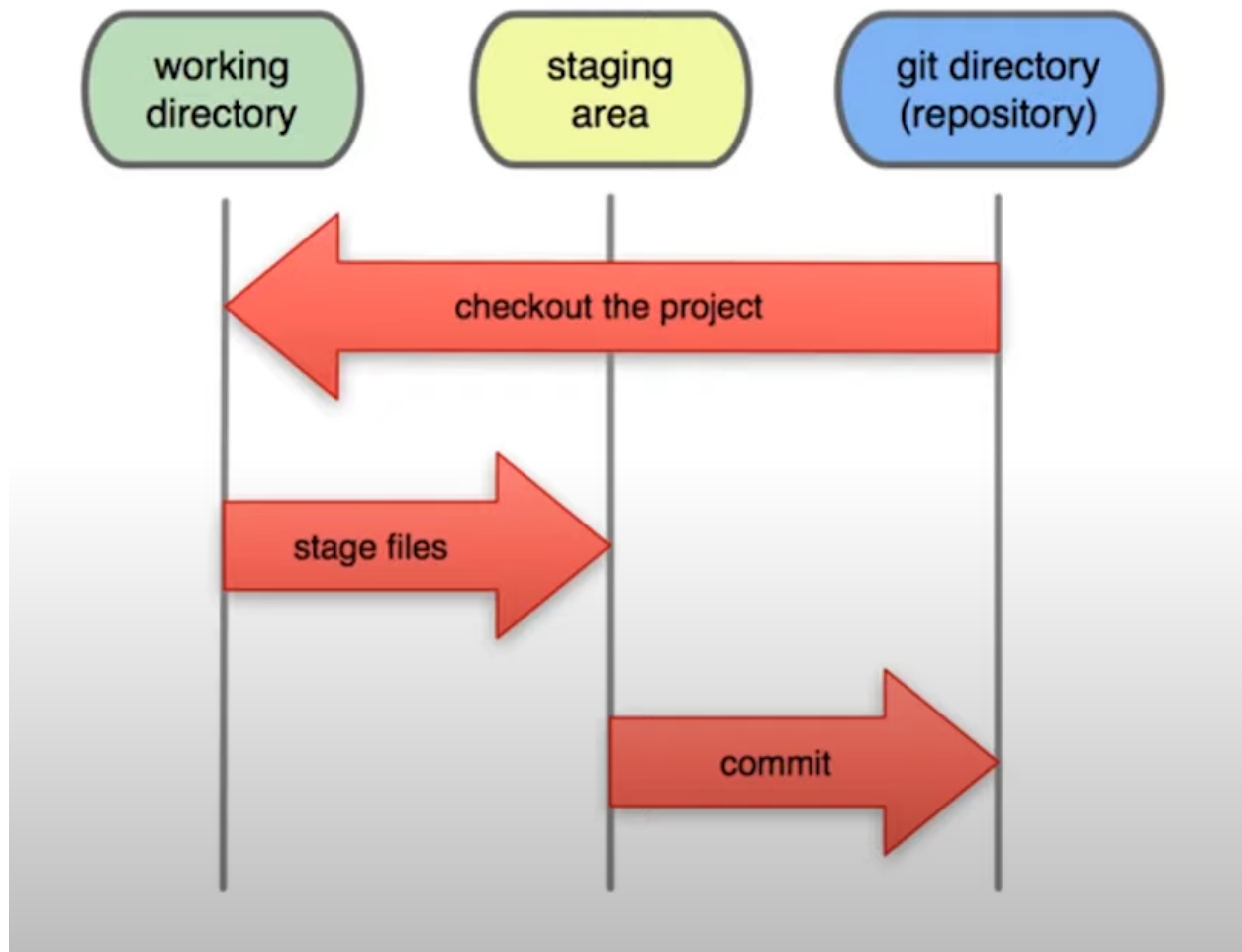
Create a new repository on the command line

```
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:Capt-SumitDas/Testweb.git
git push -u origin master
```

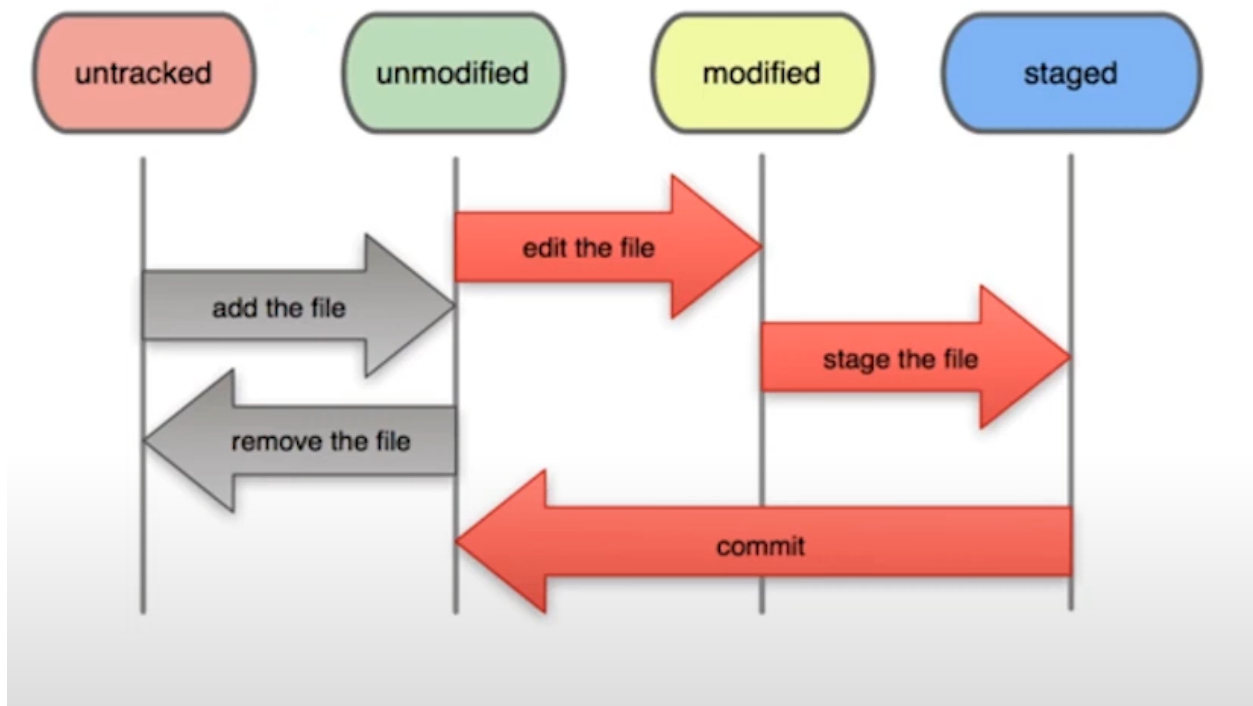
Push an existing repository from the command line

```
git remote add origin git@github.com:Capt-SumitDas/test.git
git push -u origin master
```

Local Operations



File Status Lifecycle



git status # to check git status

git init # to create git repo

git add --a # all file go to staging area or tracking

git add . # all file go to staging area or tracking

git add <file_name> # chosing file go to staging area

git restore --staged <file_name> #Unstage the file

git checkout -- <file_name> #Unmodify the change

git checkout -f #Unchange and go the last commit

git commit -m "first commit" # take a file screenshot

git commit -a -m "update message" #direct commit which already in tracking
#also called skipping staging area

git commit --amend #Change old commit

git mv file.txt fileRenamed.txt #Rename the file and already staged

git rm --cached file_name #untrack the file

rm -rf .git # to delete the .git file (git repo)

git rm file.txt # remove the file

git clone <URL> <foldername_which_you_want> # to clone repo

ignore file in **Git**

- Create a **.gitignore** file
- write into file which which you want to hide
- if you want to ignore specific file like log file then write in **.gitignore** file the * and the file extension name ex: ***.log**
- if you want to ignore specific folder then write the name of folder(ex: **foldername/**) in the **.gitignore** file.

- if you have 2 folders with same name but 1 in into another folder and you want to ignore outer file then you write ex: **/foldername/**
- if you want to ignore under the folder directory then first write folder name and then directory name ex: **FolderName/DirectoryName**

git log # to see commits

- **git log -p** #show what is add and what is remove
- **git log -p -2** #show to commits only
- **git log --stat** #show summary of add or delete
- **git log --pretty=oneline** #show commits in one line
- **git log --pretty=short** #show commits in shorts
- **git log --pretty=full** #show commits in more details(author and commit)
- **git log --since=2.days** #last 2 days commits
- **git log --since=2.weeks** #last 2 weeks commits
- **git log --since=2.months** #last 2 months commits
- **git log --since=2.years** #last 2 years commits

How to deploy a SSH key to connect with Github account

1. Open Git Bash.
2. Paste the text below, substituting in your GitHub email address.

```
$ ssh-keygen -t ed25519 -C "your_email@example.com"
```

Note: If you are using a legacy system that doesn't support the Ed25519 algorithm, use:

```
$ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

3. Firstly type yes and then if you want to fill the option or skip this to Press the ENTER.
4. Adding your SSH key to the ssh-agent
 - a. Ensure the ssh-agent is running. You can use the "Auto-launching the ssh-agent" instructions in "[Working with SSH key passphrases](#)", or start it manually:

```
# start the ssh-agent in the background
$ eval "$(ssh-agent -s)"
> Agent pid 59566
```

- b. Add your SSH private key to the ssh-agent. If you created your key with a different name, or if you are adding an existing key that has a different name, replace *id_ed25519* in the command with the name of your private key file.

```
$ ssh-add ~/.ssh/id_ed25519
```

- c. 1. Add the SSH key to your account on GitHub. For more information, see "[Adding a new SSH key to your GitHub account](#)."

5. Copy the SSH key to your clipboard

```
clip < ~/.ssh/id_rsa.pub
```

6. Go to setting > SSH and GPG keys > New SSH key and paste

Note:

you also follow this page to configure the remote access.

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

git remote #show which is repo is add

git remote -v #show URL

git remote add origin git@github.com:Capt-SumitDas/test.git #add a repo and give a name origin

git push -u origin master #to push your work into Github repo

git diff # compare the staging to working area

git diff --staged #compare last commit to current staging area

git config --global alias.st status #use st as status, means we attach st as status.
alias means use custom command a original command or want to short command.

one more example

git config —global alias.unstage 'restore --staged --'

git checkout -b develop #create a branch and change

git checkout master # Change branch

git branch #to see branches

git branch -v # to see last commit msg with branches

git merge <want to copy the branch> #for merge the braches

git branch --merged #already merged branches

git branch --no-merged #not merged branches

git branch -d <branch name> #it not delete the not merged branch

git branch -D <branch name> #no error and branch gets deleted

git remote -v #to check remote repo

git remote add origin https://github.com/Capt_Sumitdas/demo.git # add repo

git push -u origin master #push on master branch

git push origin <branch name>

git push origin <Branch name>:<New Branch name>

git push -d origin <Branch name> #To delete the Branch

GIT BASICS

git init <directory>	Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.
git clone <repo>	Clone repo located at <repo> onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH.
git config user.name <name>	Define author name to be used for all commits in current repo. Devs commonly use --global flag to set config options for current user.
git add <directory>	Stage all changes in <directory> for the next commit. Replace <directory> with a <file> to change a specific file.
git commit -m "message"	Commit the staged snapshot, but instead of launching a text editor, use <message> as the commit message.
git status	List which files are staged, unstaged, and untracked.
git log	Display the entire commit history using the default format. For customization see additional options.
git diff	Show unstaged changes between your index and working directory.

UNDOING CHANGES

git revert <commit>	Create new commit that undoes all of the changes made in <commit>, then apply it to the current branch.
git reset <file>	Remove <file> from the staging area, but leave the working directory unchanged. This unstages a file without overwriting any changes.
git clean -n	Shows which files would be removed from working directory. Use the -f flag in place of the -n flag to execute the clean.

GIT CONFIG

git config --global user.name <name>	Define the author name to be used for all commits by the current user.
git config --global user.email <email>	Define the author email to be used for all commits by the current user.
git config --global alias.<alias-name> <git-command>	Create shortcut for a Git command. E.g. alias.glog "log --graph --oneline" will set "git glog" equivalent to "git log --graph --oneline".
git config --system core.editor <editor>	Set text editor used by commands for all users on the machine. <editor> arg should be the command that launches the desired editor (e.g., vi).
git config --global --edit	Open the global configuration file in a text editor for manual editing.

GIT LOG

git log --<limit>	Limit number of commits by <limit>. E.g. "git log -5" will limit to 5 commits.
git log --oneline	Condense each commit to a single line.
git log -p	Display the full diff of each commit.
git log --stat	Include which files were altered and the relative number of lines that were added or deleted from each of them.
git log --author= "pattern"	Search for commits by a particular author.
git log --grep="pattern"	Search for commits with a commit message that matches <pattern>.
git log <since>..<until>	Show commits that occur between <since> and <until>. Args can be a commit ID, branch name, HEAD, or any other kind of revision reference.
git log -- <file>	Only display commits that have the specified file.
git log --graph --decorate	--graph flag draws a text based graph of commits on left side of commit msgs. --decorate adds names of branches or tags of commits shown.

REWRITING GIT HISTORY

git commit --amend	Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit's message.
git rebase <base>	Rebase the current branch onto <base>. <base> can be a commit ID, branch name, a tag, or a relative reference to HEAD.
git reflog	Show a log of changes to the local repository's HEAD. Add --relative-date flag to show date info or --all to show all refs.

GIT BRANCHES

git branch	List all of the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>.
git checkout -b <branch>	Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.
git merge <branch>	Merge <branch> into the current branch.

REMOTE REPOSITORIES

git remote add <name> <url>	Create a new connection to a remote repo. After adding a remote, you can use <name> as a shortcut for <url> in other commands.
git fetch <remote> <branch>	Fetches a specific <branch>, from the repo. Leave off <branch> to fetch all remote refs.
git pull <remote>	Fetch the specified remote's copy of current branch and immediately merge it into the local copy.
git push <remote> <branch>	Push the branch to <remote>, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

GIT DIFF

git diff HEAD	Show difference between working directory and last commit.
git diff --cached	Show difference between staged changes and last commit

GIT RESET

git reset	Reset staging area to match most recent commit, but leave the working directory unchanged.
git reset --hard	Reset staging area and working directory to match most recent commit and overwrites all changes in the working directory.
git reset <commit>	Move the current branch tip backward to <commit>, reset the staging area to match, but leave the working directory alone.
git reset --hard <commit>	Same as previous, but resets both the staging area & working directory to match. Deletes uncommitted changes, and all commits after <commit>.

GIT REBASE

git rebase -i <base>	Interactively rebase current branch onto <base>. Launches editor to enter commands for how each commit will be transferred to the new base.
-------------------------	---

GIT PULL

git pull --rebase <remote>	Fetch the remote's copy of current branch and rebases it into the local copy. Uses git rebase instead of merge to integrate the branches.
-------------------------------	---

GIT PUSH

git push <remote> --force	Forces the git push even if it results in a non-fast-forward merge. Do not use the --force flag unless you're absolutely sure you know what you're doing.
git push <remote> --all	Push all of your local branches to the specified remote.
git push <remote> --tags	Tags aren't automatically pushed when you push a branch or use the --all flag. The --tags flag sends all of your local tags to the remote repo.

