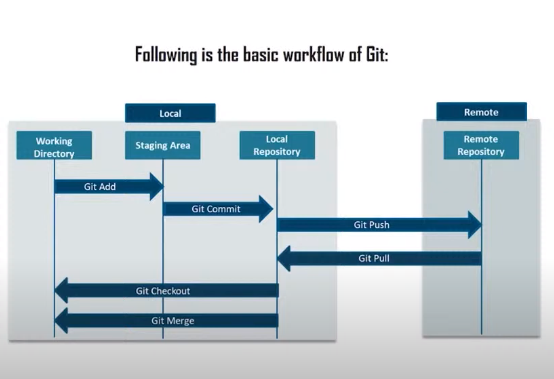
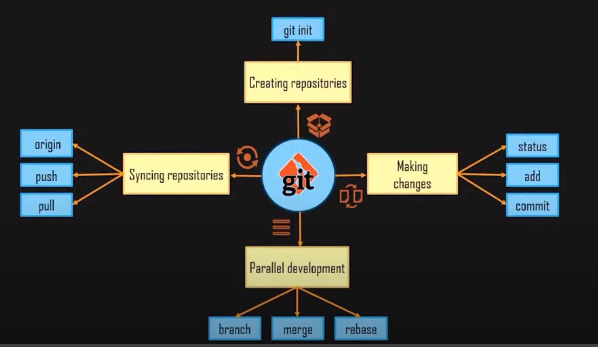
**Git**

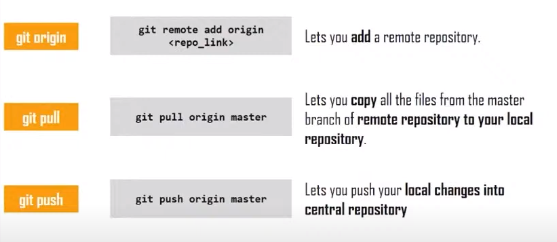


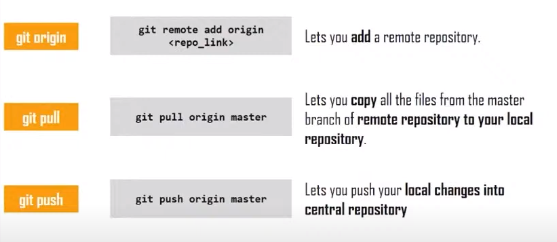


/

**How many ways we can create a repo?**







**Business Reasons to Integrate Application with GitHub**

* Preventing breaking changes from deploying
* Automatic review branches allow senior technicians to review the changes
* Integration with message boards boost communication with clients
* Leverage 3rd – party SAAS, saving money

**Git**

**What is Git?**

* Widely adopted source control system
* Distributed
* Free and open-source

**Advantages of Git**

* Fast
* Disconnected
* Powerful and easy to use
* Branching
* Pull requests

**Advantages of GitHub**

* Hosting service using git
* Much more than just a source code management
* Free and paid plans
* Based on no of users and load the cost will vary

**Basics of Git**

* Distributed Version Control (GitHub)
* We can work offline

**The Three States of Git**

* 1. Committed
* 2. Modified
* 3. Staged

**The Three Areas of Git**

* 1. Workspace
* 2. Staging Area/ Index Area
* 3. Local Repository / .git repo -> repository managed by Git only

**Adding a Remote**

* Remote repo GitHub

**Working with Git**

* Git CLI / Git bash
* Git GUI

**Using a UI for Git**

* GitHub desktop
* Source Tree
* Git Extensions

**Git installation**

* Git is widely supported all OS
* Download git from <https://git-scm.com/downloads>
* An editor (Notepad ++)
* GitHub account

**Git Config**

* $ git config --global user.name “subbarajpenmetsa”   --> // username will be added to config
* $ git config --global use.email “[subbarajpenmetsa@gmail.com](mailto:subbarajpenmetsa@gmail.com)”  --> // email will be added to the config file
* $ git config --edit --global --> // it will open the config file

**Adding note pad++ as an editor**

* $ config core.editor “notepad++ -multiInst -nosession”

**Git Fundamentals**

* $ git      --> // it all start with “git”
* $ git config  --> // configure the tooling
* $ git init      --> // initialize a git local repo
* $ git clone        --> // download a project from remote (GitHub)
* $ git add     --> // prepare a file (to staging)
* $ git commit  --> // commit the changes to local repo

**Creating Local Repo and file**

* $ cd “d:\code\subbu” --> // go to your directory
* $ git init DemoApp     --> // initialize local repo
* $ cd DemoApp          --> // go to directory and it will be (master)
* $ ls –la                     --> // it will display the list of files in the current directory = master
* $ git status               --> // it will display the status of the files
* $ notepad++ README.md --> // a note pad++ will open the filename
* $ git add README.md      --> // adding the file to staging area
* $ git commit –m “initial commit” --> // committing file to local repo

**Module-3: Getting started with GitHub**

**Overview of GitHub**

* A web based hosting service for Git
* Distributed version control and source code management
* Project Management
* Focus on social aspect of coding
* Microsoft acquired by Microsoft

**GitHub Main Features**

* Code. Code. Code
* Pull requests   --> // code review
* Issued and Project Management Tool
* Documentation
* Notifications
* Teams          --> // Role based system to give permissions to the members in a team
* Gists             --> // A simple way to share a single file or a part of a file (snippets)
* Integrations  --> //

**GitHub Plans**

* Free
* Paid
* Developer
* Team
* Business Cloud
* Enterprise

**Common Alternatives for GitHub**

* GitLab
* BitBucket

**GitHub Account**

* Create an account
* Your GitHub profile
* Creating your first repository

**GitHub Repository**

* It’s a folder for your project
* Its gives you the history of your each and every file
* Owned by you or an organization
* Public or Private

**Creating a Repository in GitHub**

* Go to your GitHub account
* Create a new repository: firstrepo
* Initialize this repository with a README file
* Add.gitignore file  --> //
* Click on create
* Go to Git CLI
* $ git status
* $ git remote –v   --> // it will give connected repo
* $ git remote add origin <https://github.com/gill-orange/firstrepo1.git> --> // config remote repository
* $ git remote –v
* $ git push –u origin master --> // it will create a master branch in GitHub in the configured repo

**Connecting with GitHub**

* HTTPS
* SSH  --> // using SSH you can use SSO and it is secure

**Connecting using SSH**

* **Creating SSH key**
* $ ssh –keygen –t rsa –b 4096 –c “[subbarajpenmetsa@gmail.com](mailto:subbarajpenmetsa@gmail.com)”
* Enter file in which to save the key (/c/users/subbu/.ssh/id\_rsa):
* Enter passphrase (empty for no paraphrase):
* Go to the directory to check the key
* Open the file using notepad++ and copy everything
* Go to GitHub account > Profile> Settings > SSH and GPG keys
* Enter and new ssh key
* Specify Title : Deslktop Key
* Key: past the date which you copied from the key file
* Click on Add SSH key
* $ ssh –T [git@github.com](mailto:git@github.com) --> // to check the SSH
* You will get successfully authenticated message

**Finding Stuff on GitHub**

* Global search
* Scope of your search
* Repositories, code, issues
* Advanced search page
* Search syntax allows further

**Module-4: Working with Repositories in GitHub**

**Repositories in GitHub**

* Base building blocks of GitHub
* Folder for your project
* Owned by you or your organization
* Public or Private

**Special Files in GitHub Repository**

* **README**file is a special file known by GitHub
* **–Root, .github or docs**folder
* Rendered automatically on landing page
* Typically written markdown (.md)
* **LICENSE**file which contains the required licenses of your project
* **CONTRIBUTING and CONTRIBUTORS**file which contains all the users
* **CHANGELOG**file which contains all the changes between different version of the file
* **SUPPORT**file which inform people about possible ways to get help with the project
* **CODE\_OF\_CONDUCT**file is guidelines for the people under the rules they have to obey interacting with the project

**Creating files on GitHub**

* **README**file

**Working with GitHub locally**

* $ git add  --> // adding files to staging area
* $ git commit  --> // committing files to local repository (.git repo)
* $ git push    --> // pushing changes to remote server = GitHub repository
* $ git fetch  --> // bring down the changes to local repository from GitHub repository
* $ git merge  --> //
* $ git pull  --> // a fetch and merge it will fetch the changes from GitHub to local repo

EX:

* Go to GitHub repository and click on clone or download, copy the link
* $ git clone url copied
* Go to the git directory
* $ ls –la  --> // to show the list of files
* $ git status  --> // show the status of the files
* $ git add .  --> // add files to staging area
* $ git commit –m “added website files”   --> // commit changes to local repo
* $ git push origin master --> // it will push the files from local master to GitHub master repo

**Adding and Editing Files on GitHub**

* $ git fetch  --> // it will fetch only the changes. No merge operation
* $ git pull --> // it will fetch and merge with your local changes
* $ git status

**Archiving Repositories**

* Go to repo page in GitHub > Settings > Danger Zone > Archive this repository

**Repository Features**

* Topics  --> // it is a tag classify your project. It is used to categorize your repository
* Projects  --> // allowing you to manage projects
* Issues    --> // it is like an issue tracker
* Pull Requests  --> // requesting me to pull those changes to new branch
* Insights --> //
* Settings  --> //

**Topics**

* Go to GitHub repo and click on Add topics
* Enter topic “This is a website”

**Issue**

* Click on “Create a New issue”
* Specify the issue details “Title is not well aligned”

**Bring in More People – Collaborators**

* **Collaborators** --> // fixed group of people of core development team working on a project. They have more permissions on the project. They can change the code in the main repository
* **Adding**>Repo > settings > Collaborators >
* **Contributors --**> // everyone outside of the core team. Someone who uses your project and do the development activates. They have less permissions and they can’t commit the code to the main branch

**Repository Insights**

* Contributors
* Traffic
* Commits
* Code Frequency

**Module-5: Branching, Merging and Pull Requests in GitHub**

**An overview of Branch**

* Diverging from the main development line
* Much more lightweight
* Branching is very fast
* Branch is a pointer to a commit
* Default branch is master
* Points to latest commit
* Moves forward with every commit
* Use branches for
* Features
* Bugs
* Experiments

**Creating Branch on GitHub**

* Go to GitHub.com
* Search or Create a new branch

**Commands for Branching**

* $ git branch [branch-name]  --> // creating branch
* $ git checkout [branch-name] --> // switching to branch
* $ git push – u [origin] [branch-name] --> // pushing the branch to GitHub repo

EX:

* $ git status
* $ git branch subbu  --> // new branch subbu will be created
* $ git checkout subbu  --> // switching to subbu
* $ git checkout –b “[branch-name]” -->// creation and switching branch using a single command
* $ touch file1.txt  --> // creating file1.txt using touch command
* $ git add .  --> // adding file1.txt to staging area
* $ git commit –m “comments”  --> // commit file1.txt to local repository
* $ git push –u origin add-installation --> // local repo will sync with remote repo = GitHub

**Pull Requests and GitHub flow**

* Tell what you have pushed to a branch of a repo
* Discuss, review and add more commits
* Merge into base branch = master

**Creating Pull request for a branch**

* Go to GitHub.com
* Create pull request

**Module 6: Creating Tags and Releases**

**Tags**

* A branch points to a snapshot in the history of your project
* A tag points to an important point in the history
* Tags can be added later on an existing commit

**Tags Types**

* 1. Lightweight >
* 2. Annotated
* Message, checksum

**Adding Tags**

* $ git tag   --> // adding tag
* $ git log  --> // log command will you the history of the current branch
* $ git tag --> // it will show all the current tags
* $ git tag stable master --> // it will create “stable” tag on my master branch
* $ git tag  -->// it will display the current tags
* $ git log --oneline --graph --decorate --all  --> // it will display the log
* $ git tag -a v0.1 -m “0.1 release” a6b446e  --> // it will create annotated tag

**Push Tags to GitHub repo**

* $ git push --tags --> // it will push all the tags to GitHub repo

**Deleting Tags**

* $ git tag  --> // display tags
* $ git tag -d “stable”  --> // delete stable tag from local repo
* $ git push --> // pushing changes to GitHub

**Working with Releases**

* Release notes
* Providing links to binary files

**Creating Releases**

* Go to GitHub.com and create a release

**Module-7: The Social Aspect of GitHub**

**Working with Forks**

* It’s a copy of repository
* Doesn’t impact the original repository
* Changes can be merged again via pull requests in fork
* We can merge the changes to original from fork
* Available to everyone
* It’s is your own repository
* Only related to GitHub not Git
* It is used for testing purpose

**Working with Forks**

* Go to GitHub.com
* Go to the required repository
* Click on “fork”
* Clone this forked repo to local
* $ git clone address of cloned repository

**Pull Requests**

* Update others about the changes you have made in a branch or fork
* Discuss and review
* Optionally merged into main branch

**Using Gists**

* Gist is a simple way to share snippets and notes with others
* Allow sharing single file, snippet or even a full project
* Repository-based
* Public or secret
* Downloadable in the form of zip
* Go to [https://gist.github.com](https://gist.github.com/)

**Creating GitHub Pages**

* Static site hosting from without server-side code
* Server less
* Site is effectively a repo
* Can be created online or offline
* Go to GitHub.com
* Settings > GitHub Pages > Specify the repository which you want to use as GitHub pages

**Enabling Notifications for projects**

* Types of notifications
* Participating
* Watching notifications