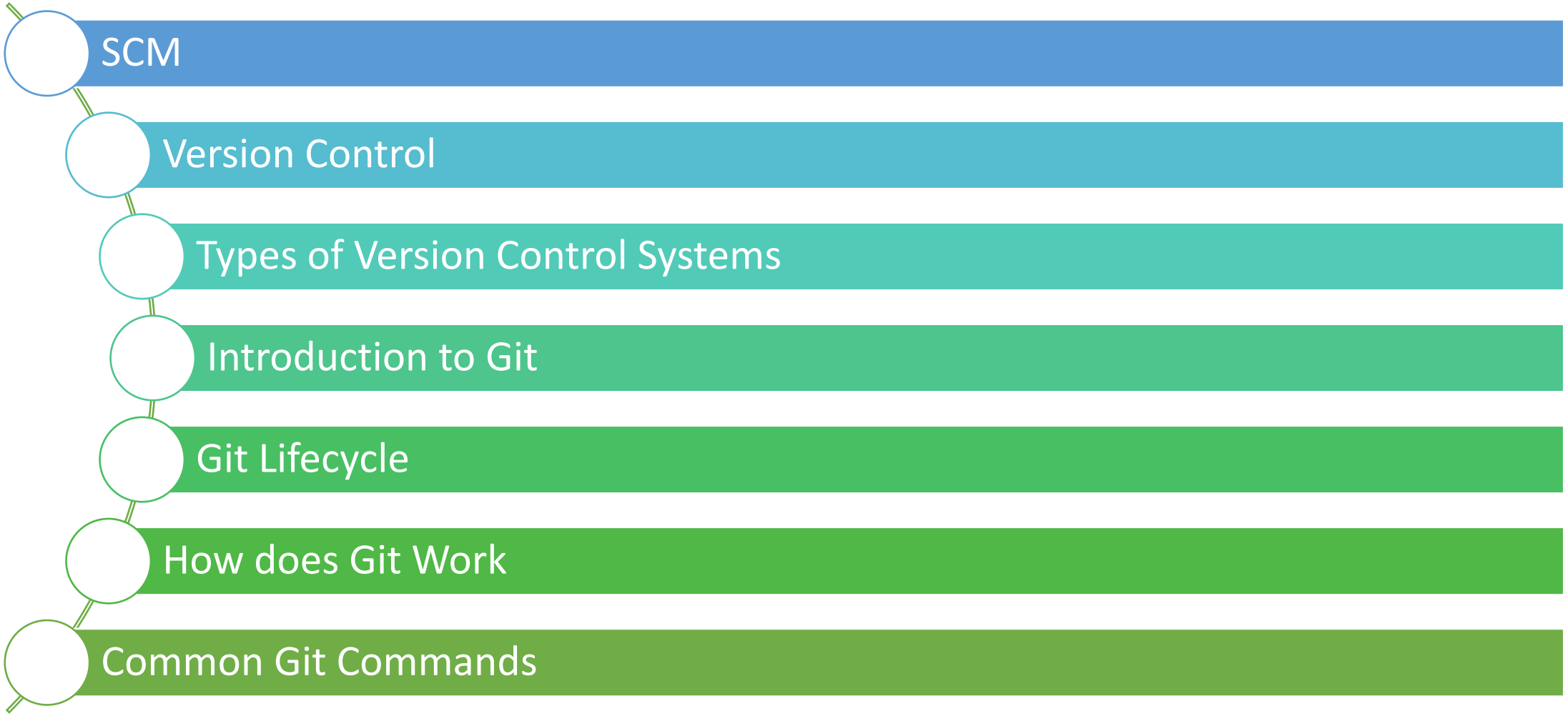
GitNGithub
Source Code Management

Version Control System (Git)

Presented by: [Santhosh Kiran J]

Agenda


SCM
Version Control
Types of Version Control Systems
Introduction to Git
Git Lifecycle
How does Git Work
Common Git Commands

Source Code Management (SCM)

- SCM is a broader term that encompasses all practices and tools used to manage changes to source code over time. It includes version control but also involves other aspects like tracking changes, managing code branches, and integrating code from multiple developers.
- **Features:** SCM tools provide a comprehensive suite of functionalities, including version control, code review, continuous integration, and deployment pipelines. They help in managing the entire lifecycle of software development.

Version Control System (VCS)

- **Definition:** A system that records changes to a file or set of files over time so that you can recall specific versions later.
- **Importance:** Helps in tracking changes, collaborating with others, and maintaining a history of project development.

Types of VCS

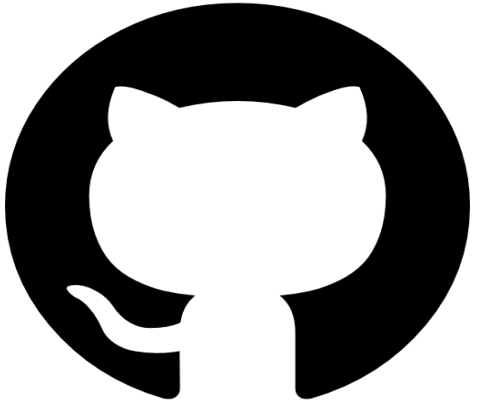
- **Local Version Control Systems:** Simple databases that keep all changes to files under revision control.
- **Centralized Version Control Systems (CVCS):** Single server containing all versioned files, with multiple clients checking out files.
- **Distributed Version Control Systems (DVCS):** Each user has a complete copy of the repository, including its full history.

Key differences between SCM and Version Control

- **Scope:** SCM covers a wider range of activities beyond just version control, including code integration, build management, and deployment. VCS is specifically about tracking changes to files.
- **Tools:** SCM tools often include VCS as part of their functionality. For example, Git is a VCS, while GitHub or GitLab are SCM platforms that provide additional features like issue tracking, CI/CD, and project management.

Examples of SCM Tools

GitHub



A popular tool for version control and source code management that allows developers to track and commit code changes. It's also a collaborative tool for developers to work together on projects.

Option A

GitLab

An open-source and free tool that's popular for source code management. It offers strong CI services and can be hosted on a user's own server.

Examples of SCM Tools

AWS CodeCommit

A fully managed source control solution from Amazon Web Services that allows teams to collaborate in a secure and scalable environment.

Microsoft Team Foundation Server (TFS)

An Application Life cycle Management (ALM) system that includes source code control, versioning, bug tracking, project management, and team collaboration features.

Examples of SCM Tools

Bitbucket

A top-rated DevOps tool for source code management that allows developers to code, test, and deploy from a single platform.

Examples of Version Control Tools

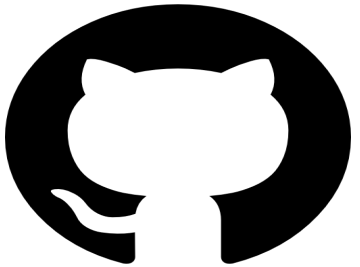
Git

A prominent version control system that's used to manage small and large projects. It helps track source code changes and allows different people to collaborate on different parts of a program.

Apache Subversion

An open-source version control system that's a reliable option for important data. It offers features like history tracking, security, and inventory management.

GitHub Features



Easy Project Management

A place where project managers and developers come together to coordinate, track, and update their work so that projects are transparent and stay on schedule.

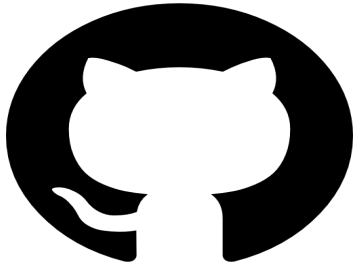
Increased Safety With Packages

Packages can be published privately, within the team, or publicly to the open-source community. The packages can be used or reused by downloading them from GitHub.

Effective Team Management

GitHub helps all the team members stay on the same page and organized. Moderation tools like Issue and Pull Request Locking help the team to focus on the code.

GitHub Features



Improved Code Writing

Pull requests help the organizations to review, develop, and propose new code. Team members can discuss any implementations and proposals through these before changing the source code.

Increased Code Safety

Packages can be published privately, within the team, or publicly to the open-source community. The packages can be used or reused by downloading them from GitHub.

Easy Code Hosting

All the code and documentation are in one place. There are millions of repositories on GitHub, and each repository has its own tools to help you host and release code.

Git Terminology

- **Branch**: A version of the codebase that diverges from the main branch to isolate changes for specific features, fixes, or experiments.
- **Commit**: A snapshot of your changes, saved to your local repository. Each commit is uniquely identified by a checksum.
- **Stage**: The area where Git tracks changes that are ready to be included in the next commit. Files in the staging area are prepared (staged) for the next commit.
- **Merge**: The process of integrating changes from one branch into another, typically the main branch.
- **Pull Request**: A proposal to merge changes from one branch into another, often used in collaborative environments to review and discuss changes before they are merged.

Git Terminology

- **Fork**: A personal copy of someone else's project that lives on your GitHub account.
- **Clone**: The act of downloading a repository from a remote source to your local machine.
- **Remote**: A common repository that all team members use to exchange their changes.
- **Origin**: The default name Git gives to the server from which you cloned.
- **Upstream**: The original repository that was cloned.
- **Master:** The default branch name given to a repository when it is created. In modern practice, it is often replaced with main.
- **Repository**: A storage location where your project lives, containing all the files and revision history.
- **Working Directory**: The directory on your computer where you are making changes to your project.

Git Terminology

- **Staging Area**: Also known as the "Index," it's an area where Git tracks changes that are ready to be committed.
- **Index**: Another name for the staging area, where Git tracks changes that are ready to be committed.
- **HEAD**: A reference to the last commit in the currently checked-out branch.
- **Checkout**: The action of switching from one branch to another or to a specific commit.
- **Push**: The action of sending your commits to a remote repository.
- **Pull**: The action of fetching changes from a remote repository and merging them into your current branch.
- **Fetch**: The action of retrieving updates from a remote repository without merging them into your current branch.

Git Commands: Working With Local Repositories

❖git init

Command: git init

- creates an empty Git repository.
- After the git init command is used, a .git folder is created in the directory with some subdirectories. Once the repository is initialized, the process of creating other files begins.

❖git config

Command: git config --global user.name “any user name”
git config --global user.email <email id>

- The git config command is used initially to configure the user.name and user.email. This specifies what email id and username will be used from a local repository.
- When git config is used with --global flag, it writes the settings to all repositories on the computer.

Git Commands: Working With Local Repositories

❖git add

Command: git add . Or git add <filename>

- Add command is used after checking the status of the files, to add those files to the staging area.
- Before running the commit command, "git add" is used to add any new or modified files.

❖git status

Command: git commit –m “commit message"

- The git status command tells the current state of the repository.
- The command provides the current working branch. If the files are in the staging area, but not committed, it will be shown by the git status. Also, if there are no changes, it will show the message no changes to commit, working directory clean.

Git Commands: Working With Local Repositories

❖**git commit**

- ➢The commit command makes sure that the changes are saved to the local repository.
- ➢The command "allows you to describe everyone and help them understand what has happened.

Command: git commit –m “commit message"

❖**git branch**

- ➢The git branch command is used to determine what branch the local repository is on.
- ➢The command enables adding and deleting a branch.

Commands:

git branch <branch_name> # Create a new branch
git branch –a # List all remote or local branches
git branch -d <branch_name> # Delete a branch

Git Commands: Working With Local Repositories

❖git checkout

- ➢The git checkout command is used to switch branches, whenever the work is to be started on a different branch.
- ➢The command works on three separate entities: files, commits, and branches.

Commands:

git checkout <branch_name> # Checkout an existing branch

git checkout -b <new_branch> # Checkout and create a new branch with that name

❖git merge

- ➢The git merge command is used to integrate the branches together. The command combines the changes from one branch to another branch.
- ➢It is used to merge the changes in the staging branch to the stable branch.

Command:

git merge <branch_name>

Git Commands: Working With Remote Repositories

❖git remote

- ➢The git remote command is used to create, view, and delete connections to other repositories.
- ➢The connections here are not like direct links into other repositories, but as bookmarks that serve as convenient names to be used as a reference.

Commands:

git remote add origin <address>

❖git clone

- ➢The git Clone command is used to point to an existing repo and make a clone or copy of that repo at in a new directory, at another location.
- ➢It is used to merge the changes in the staging branch to the stable branch.

Command:

git clone <remote_URL>

Git Commands: Working With Remote Repositories

❖git pull

- ➢The git pull command is used to fetch and merge changes from the remote repository to the local repository.
- ➢The command "git pull origin master" copies all the files from the master branch of the remote repository to the local repository.

Commands:

git pull <branch_name> <remote URL>

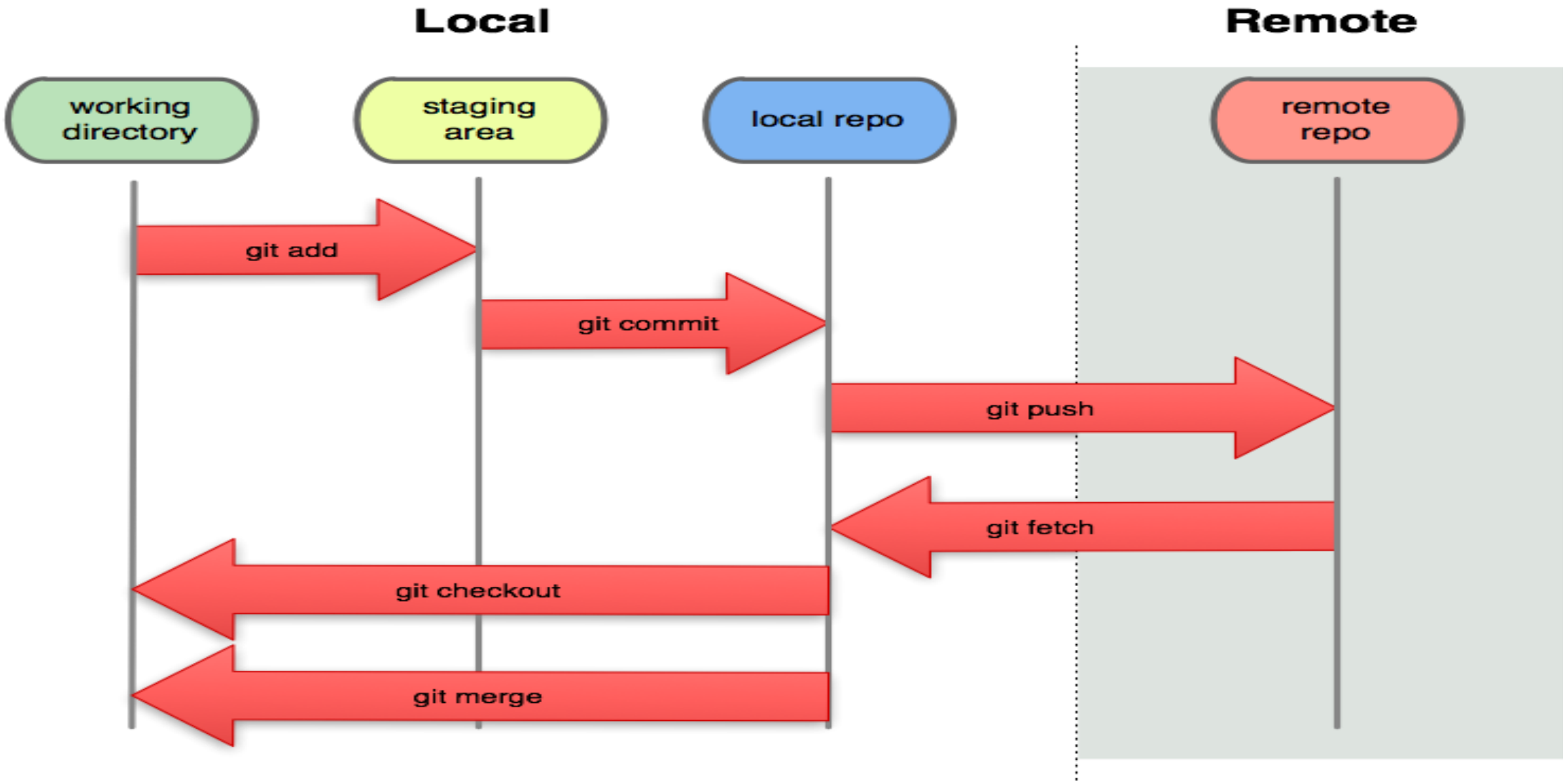
❖git push

- ➢The command git push is used to transfer the commits or pushing the content from the local repository to the remote repository.
- ➢The command is used after a local repository has been modified, and the modifications are to be shared with the remote team members.

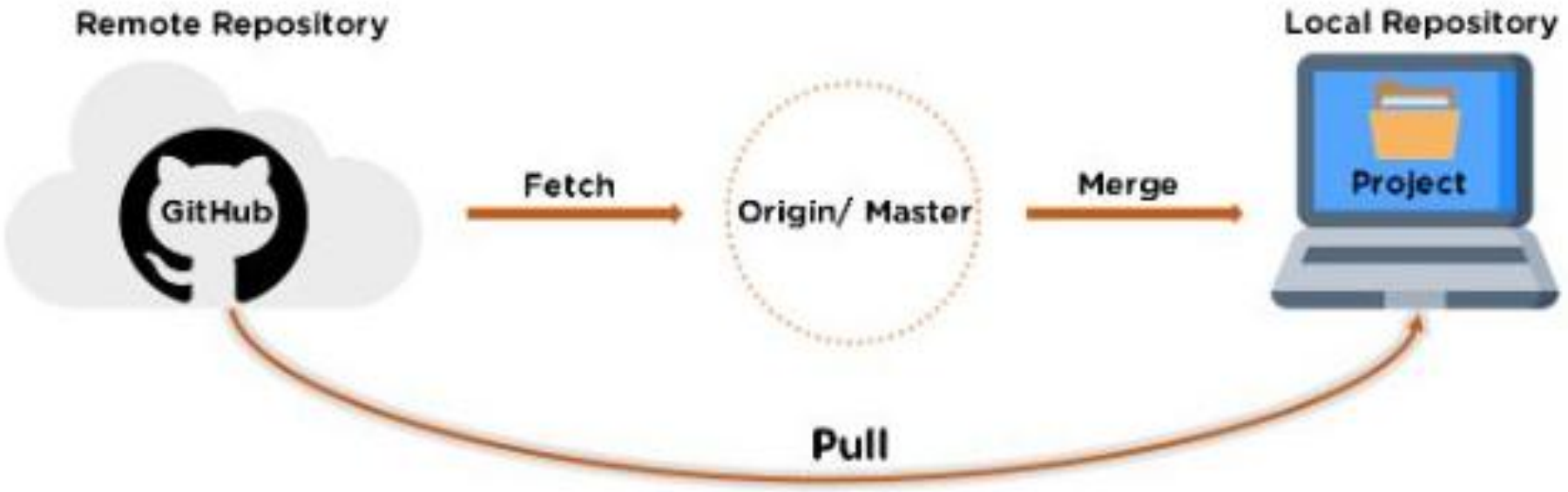
Command:

git push -u origin master (--set-upstream*)*

Git Commands



Git Flow


Remote Repository
GitHub
Fetch
Origin/ Master
Merge
Local Repository
Project
Pull

References

- https://github.com/joshnh/Git-Commands
- https://education.github.com/git-cheat-sheet-education.pdf
- https://docs.github.com/en/get-started/using-git
- https://docs.github.com/en/get-started/learning-about-github