**Git**

### Git and GitHub Fundamentals: Overview

This guide covers the essentials of **Git** and **GitHub**, explaining basic concepts, installation, and usage, including how to work with repositories, branches, tags, and other common tasks.

### 1. **Introduction to SCM (Source Control Management) and Git**

* **Source Control Management (SCM)** is the practice of tracking and managing changes to code over time.
* **Git** is a distributed version control system (VCS) that tracks changes to files and allows multiple people to collaborate on a project.
* **GitHub** is a cloud-based platform built on Git, offering additional features like collaboration, bug tracking, and project management.

### 2. **Installation of Git**

To install Git on various operating systems:

* **Linux (RHEL/CentOS)**:
* sudo yum install git
* **Windows**: Download the Git installer from the official Git website.
* **macOS**:
* brew install git

### 3. **Basics of Git**

* **Git File System**: Git uses a **repository** to store files and track changes. It has three key states:
  + **Working Directory**: The files in your project.
  + **Staging Area**: Where files are staged before committing.
  + **Git Repository**: Where the entire history of changes is stored.
* **Creating a Local Repository**: Initialize a new Git repository in your project directory:
* git init
* **Basic Git Configuration**: Set your user name and email for commits:
* git config --global user.name "Your Name"
* git config --global user.email "your.email@example.com"

### 4. **Adding Files to a Project**

* To add files to the staging area, use:
* git add <file\_name>
* To add all changes:
* git add .

### 5. **Status of the Project**

* To see the current state of the repository (which files are staged, modified, or untracked):
* git status

### 6. **Committing Changes**

* After staging files, commit them to the repository:
* git commit -m "Commit message"

### 7. **Ignoring Specific File Types**

* Create a .gitignore file to specify file patterns that Git should ignore, e.g.:
* \*.log
* \*.tmp
* node\_modules/

### 8. **Tags, Branching, Merging, and Reverting**

#### Tags:

* Tags are used to mark specific points in history (usually release versions).
  + **Create a tag**:
  + git tag <tag\_name>

#### Branching:

* **Branching** allows you to work on multiple versions of a project simultaneously.
  + **Create a branch**:
  + git branch <branch\_name>
  + **Switch to a branch**:
  + git checkout <branch\_name>

#### Merging:

* Merge changes from one branch into another:
* git merge <branch\_name>

#### Rebasing:

* Rebasing allows you to apply changes from one branch onto another in a linear history:
* git rebase <branch\_name>

#### Reverting a Commit:

* To undo the effect of a commit but keep it in history:
* git revert <commit\_id>

#### Diff:

* The diff command shows the differences between files or commits:
* git diff

### 9. **Git Garbage Collection**

* Git performs garbage collection automatically, but you can manually trigger it to clean up unused files and optimize performance:
* git gc

### 10. **Git Logging and Auditing**

* To view the history of commits:
* git log
* **Log Options**:
  + View a simple log:
  + git log --oneline
  + Show log with differences for each commit:
  + git log -p

### 11. **Cloning Repositories**

#### Clone a Local Repository:

* Copy an existing local repository to another directory:
* git clone /path/to/repo

#### Clone a Remote Repository over HTTPS:

* To clone from a remote repository on GitHub:
* git clone https://github.com/username/repository.git

### 12. **Forking**

* **Forking** creates a personal copy of someone else’s repository. You can make changes and propose those changes back to the original project.

### 13. **Securing GitHub Account with SSH Keys**

* SSH keys provide a secure way to connect to GitHub:
  + **Generate SSH key**:
  + ssh-keygen -t rsa -b 4096 -C "your\_email@example.com"
  + **Add SSH key to GitHub**:
    - Copy the public key (typically found in ~/.ssh/id\_rsa.pub) and add it to your GitHub account under "Settings" > "SSH and GPG keys".

### 14. **WebHooks**

* **WebHooks** allow GitHub to notify external services when specific events (like pushes, pull requests) occur in a repository.
  + A webhook sends a POST request to a URL you define whenever the event is triggered.
  + Useful for continuous integration (CI) systems.

### Conclusion

This guide covers the foundational concepts and tasks in Git and GitHub. Mastering these will help you effectively manage code versions, collaborate with others, and automate workflows in software development.