# GIT and GITHUB Training

Avinash Maskey

## Agenda

- ❑ Why Version Control? What is Version Control System (VCS)?
- ❑ VCS Benefits
- ❑ Introduction to GIT and GITHUB
- ❑ Installations and GUI’s
- ❑ Set Up and Configurations
- ❑ **GIT Commands:**
  - Staging and snapshots
  - Alias GIT Commands
  - Branching and merging
  - Merge Conflict – Solving it
  - Share and update
- ❑ **GITHUB Project Implementation (Web Hosting)**

## Why Version Control?

- **Scenario 1:**
  - Your program is working
  - You change “just one thing”
  - Your program breaks
  - You change it back
  - Your program is still broken--why?

- Has this ever happened to you?

# Why Version Control? (Contd.)

- **Scenario 2:**
  - Your program worked well enough yesterday
  - You made a lot of improvements last night...
    - ✓...but you haven't gotten them to work yet
  - You need to turn in your program now

- Has this ever happened to you?

## Why Version Control For Teams? (Contd.)

- **Scenario 3:**
  - You change one part of a program--it works
  - Your co-worker changes another part--it works
  - You put them together--it doesn’t work
  - Some change in one part must have broken something in the other part

- What were all the changes?

## Version Control Systems (VCS)

- A ***version control system*** (often called a source code control system) does these things:
  - Keeps multiple (older and newer) versions of everything (not just source code)
  - Requests comments regarding every change
  - Allows “check in” and “check out” of files so you know which files someone else is working on
  - Displays differences between versions

## Benefits of Version Control

- **For working by yourself:**
  - Gives you a “time machine” for going back to earlier versions
  - Gives you great support for different versions (standalone, web app, etc.) of the same basic project

- **For working with others:**
  - Greatly simplifies concurrent work, merging changes

## What is Git?

- Git is an open source and free **(Source Control Management)**.
- It was initially developed by **Linus Torvalds** in the year **2005** with other kernel developers for the development of **Linux Kernel**.
- Its name is short for **"get it together" or "global information tracker"**, because it's designed to maintain a collection of files in a system's version control.
- Git is so powerful that it's used by companies like ***Facebook***, ***Twitter*** and so on as well as by a wide range of open source projects.

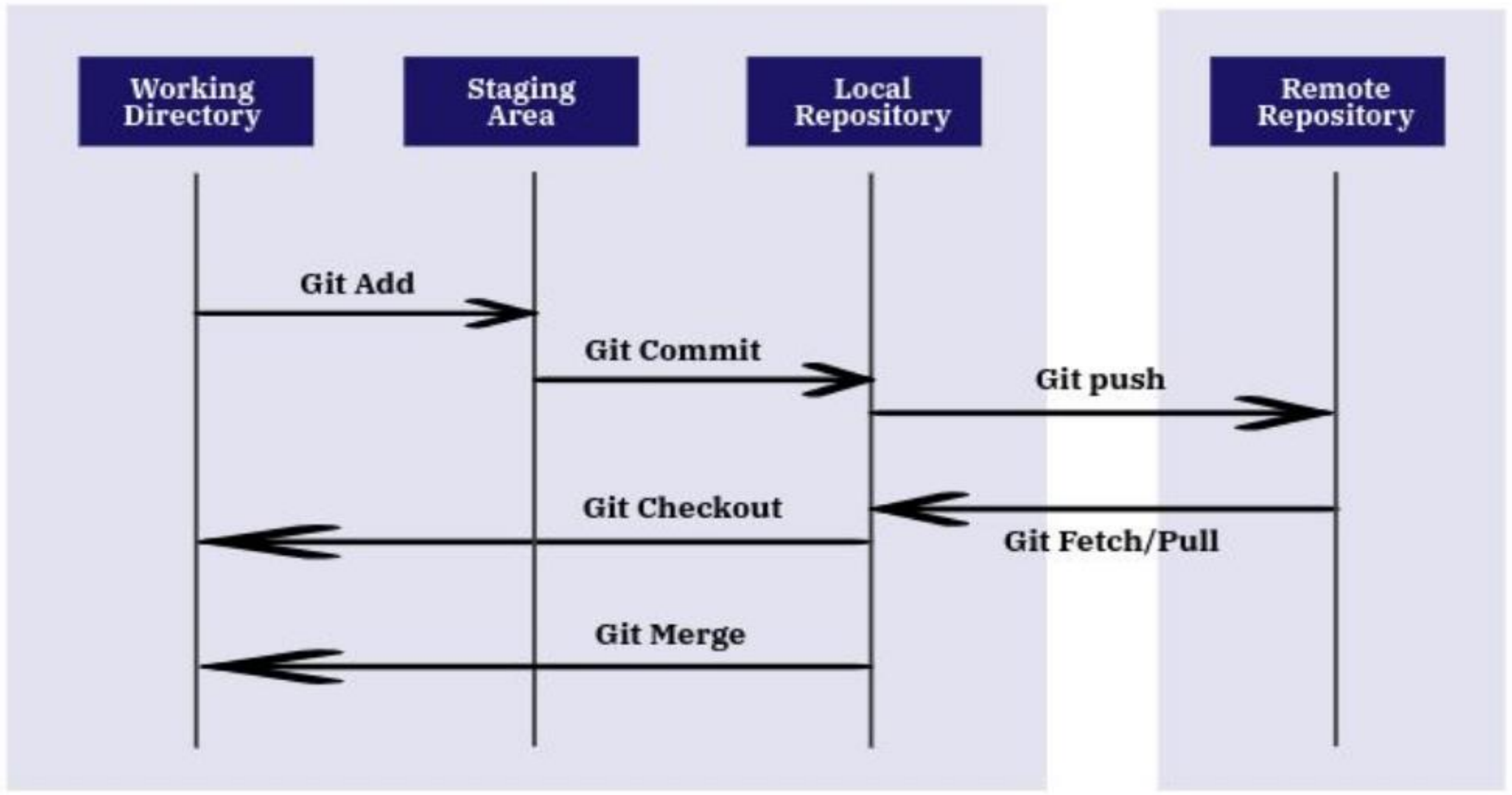
## What is Git? (Contd.)

- It’s available on many different operating systems, including ***Linux***, ***Mac OS X***, and ***Windows***.
- Git is a **V**ersion **C**ontrol **S**ystem **(VCS)** designed to make it easier to have multiple versions of a code base, sometimes across multiple developers or teams.
- It allows you to see changes you make to your code and easily revert them.
- **Note:** It is NOT GITHUB!

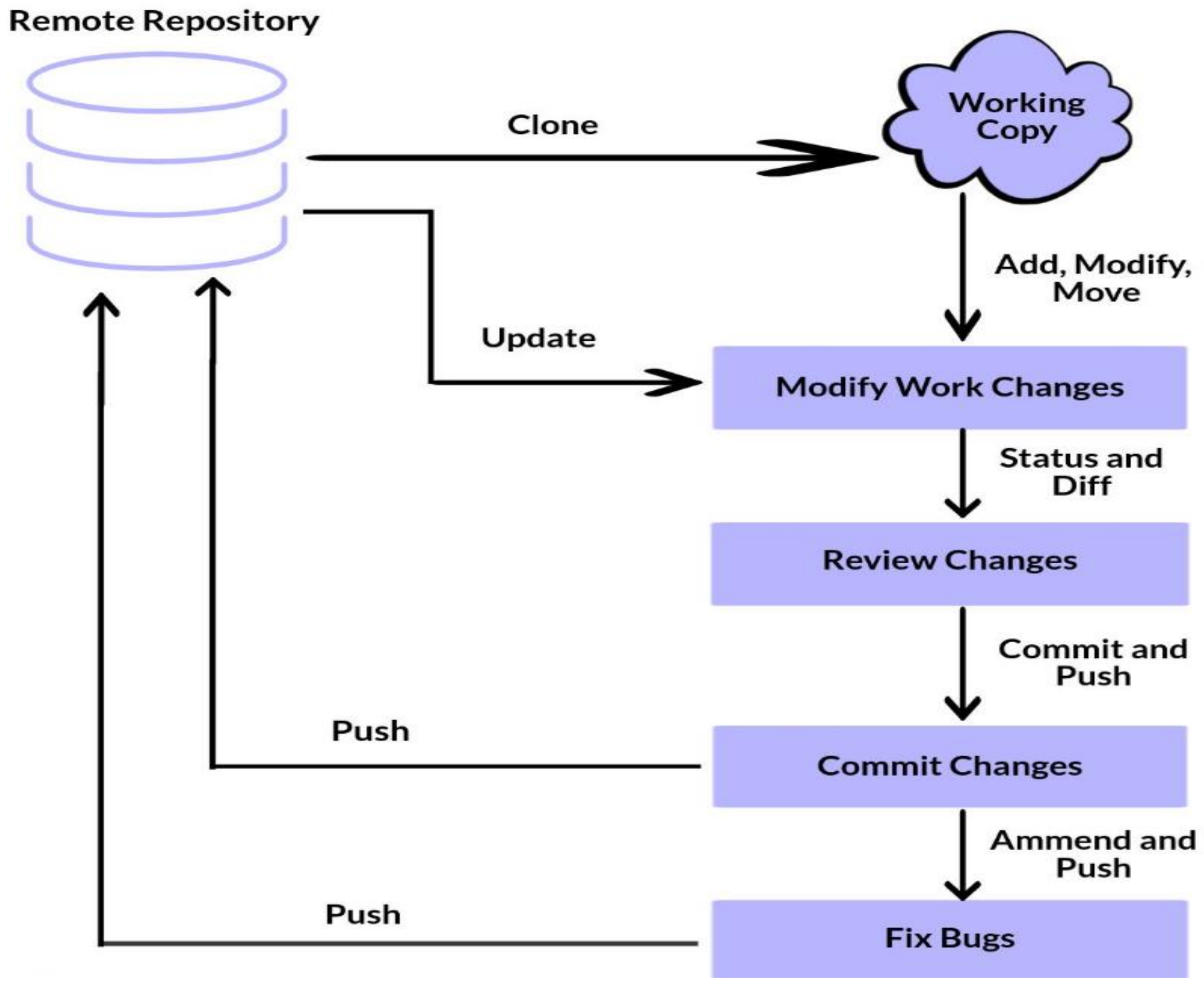
## Ok, then what is GitHub?

- ***github.com*** is a website that hosts git repositories on a remote server.
- Hosting repositories on **GitHub** facilitates the sharing of codebases among teams by providing a **GUI** to ***easily fork or clone repos to a local machine.***
- By pushing your repositories to **GitHub**, you will pretty much automatically create your own ***developer portfolio*** as well!

# GIT Workflow



# GIT Lifecycle



# INSTALLATION & GUI'S

- **Git for All Platforms:** https://git-scm.com/
- **Desktop GUI's:**
  - For Windows – https://windows.github.com
  - For Mac – https://mac.github.com
  - For Linux and Solaris platforms, the latest release is available on the official Git web site.
  - Or https://desktop.github.com/
- **Web GUI's: (We'll use this)**
  - https://github.com/ (Create an account)
- **Version Check and Update Commands:**
  - git --version
  - git update-git-for-windows

## Free Domain Registration Link

- https://register.com.np/

## SET UP - Configuring GIT

- Configuring user information used across all local repositories. Your commits will have your name and email attached to them.

- To confirm that this information is correct, run the following commands:
  - **git config --global user.name “firstname lastname”**
    - ✓set a name that is identifiable for credit when review version history
  - **git config --global user.email “valid-email”**
    - ✓set an email address that will be associated with each history marker

- Creating default branch to “main” if required:
  - **git branch --move master main**

## Setting Up and Initialization

- Configuring user information, initializing and cloning repositories.
- GIT Commands:
  - **git init**
    - ✓initialize an existing directory as a Git repository
  - **git clone “url”**
    - ✓retrieve an entire repository from a hosted location via URL

# Staging (before commit) & Snapshots (or commit)

- Working with snapshots and the Git staging area.
- GIT Commands:
  - **git status**
    - ✓show modified files in working directory, staged for your next commit
  - **git add <filename> or git add .**
    - ✓add a file as it looks now to your next commit (stage)
  - **git reset <filename> or git rm --cached <filename>**
    - ✓unstage a file while retaining the changes in working directory
  - **git diff**
    - ✓diff of what is changed but not staged
  - **git commit -m “descriptive message”**
    - ✓commit your staged content as a new commit snapshot

## Alias GIT Commands

- GIT Commands:
  - **git log**
    - ✓show all commits in the current branch’s history
  - **git rm <filename>**
    - ✓it removes or deletes a required file from the working directory
  - **git restore --staged “filename”**
    - ✓to get back the file removed from a working directory
  - **git restore “filename”**
    - ✓to retrieve back the deleted file first fire the above **git restore --staged “filename”** command and then fire the **git restore “filename”** command.
    - ✓or you can simply use **git checkout HEAD~1 “filename”** command
  - **git commit –m ‘new message’ --amend**
    - ✓this amends or changes the recent commit message

# Branching and Merging

- Isolating work in branches, changing context, and integrating changes.
- GIT Commands:
  - **git branch**
    - ✓list your branches. A * will appear next to the currently active branch
  - **git branch <branch-name>**
    - ✓create a new branch at the current commit
  - **git switch <branch-name>**
    - ✓switch to another branch and check it out into your working directory
  - **git merge <branch-name> or git merge –m ‘message’ <branch-name>**
    - ✓merge the specified branch’s history into the current one
  - **git branch –d <branch-name>**
    - ✓deletes a recently created branch in your working directory

## GIT Merge Conflict

- A ***merge conflict*** is an event or scenario that takes place when Git is unable to automatically resolve differences in code between two commits. Git can merge the changes automatically only if the commits are on different lines or branches.
- How to resolve GIT conflict?
  - we can use the command **git merge --abort**; but this is not the right to handle the issues.
  - Hence, we should perform the following steps:
    - ✓The easiest way to resolve a conflicted file is to open it and make any necessary changes.
    - ✓After editing the file, we can use the **git add** a command to stage the new merged content.
    - ✓The final step is to create a new commit with the help of the **git commit** command.
    - ✓Git will create a new merge commit to finalize the merge.

# SHARE & UPDATE

- Retrieving updates from another repository and updating local repos.
- GIT Commands:
  - **git remote add origin “url”**
    - ✓ it automatically creates a remote connection called ***origin*** pointing back to the cloned repository
  - **git branch –M main**
    - ✓ it automatically links to the main branch
  - **git push –u origin main**
    - ✓ it uploads or push all updates from local repository to the Git remote repository to main branch
  - **git fetch –all or git fetch**
    - ✓ fetch down all the branches from that Git remote repository
  - **git push -all**
    - ✓ push up all the branches from local repository to Git remote repository
  - **git pull**
    - ✓ fetch and merge any commits from the tracking remote branch

## THANK YOU!