DISTRIBUTED VERSION CONTROL WITH GIT

Presented By :
Arvind Ramachandran
15CO111

LINUS TORVALDS

- Creator and principal developer of Linux.
- Created git in a weekend.

What is git ?

- Version control system
- Tracks changes in computer files and coordinates work on those files among multiple people
- Source code management in software development
- Keeps track of changes in any set of files.
- Focus on speed and efficiency

Companies & Projects using Git

Google
facebook
Microsoft
twitter
LinkedIn
NETFLIX
PostgreSQL
android
RAILS
Qt
GNOME
eclipse

Why git ?

Manage Changes

- Stores snapshot of all your files
- Similar to checkpoint in games
- Easy modification of code

Collaborate with Others

- Git is fully distributed
- Separate local copies can be maintained
- Multiple can work simultaneously on a single piece of code
- Easily manage the changes made by others

GIT BASICS

All you need to know to start working

1. Install git

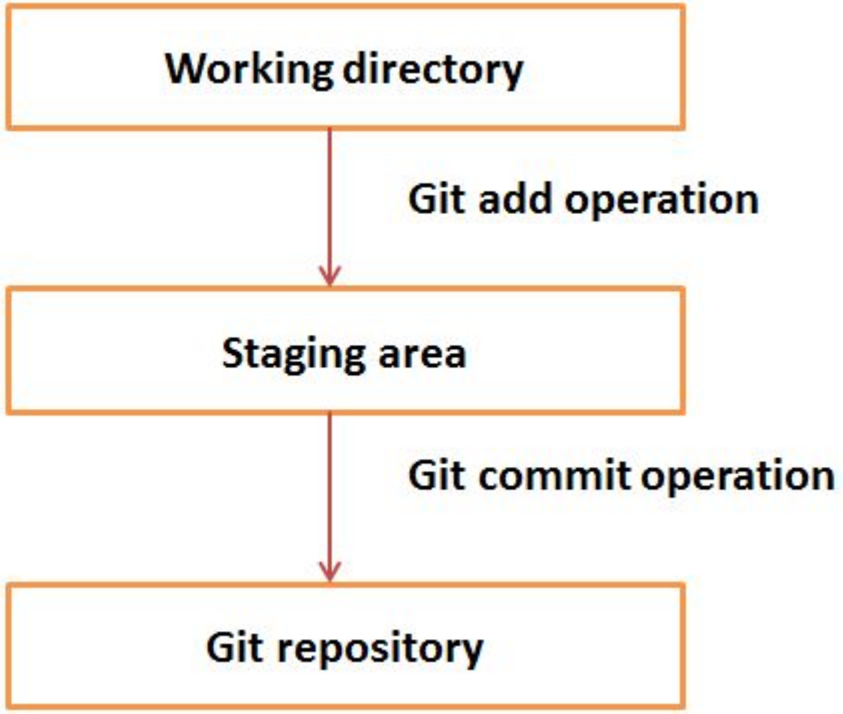
- Simple steps to download and install git
- Installers available for Linux, Windows and Mac

2. Set up git for a Project

- We need to inform git to tell it to keep track of our changes
    - Open the terminal
    - **cd** into the project folder
    - ***git init***

3. Check Status

- We need to check if everything worked
- ***git status***
- Displays the state of the working directory and the staging area
- This command will let us know what state our repo is in
- It is wise to keep running this command after each and every command we run or some change we make



4. Add your changes

- ***git add <file_name>***
- Add changes made to files to the staging area
- Updates the index using the current content found in the working tree
- Prepares the content staged for the next commit.

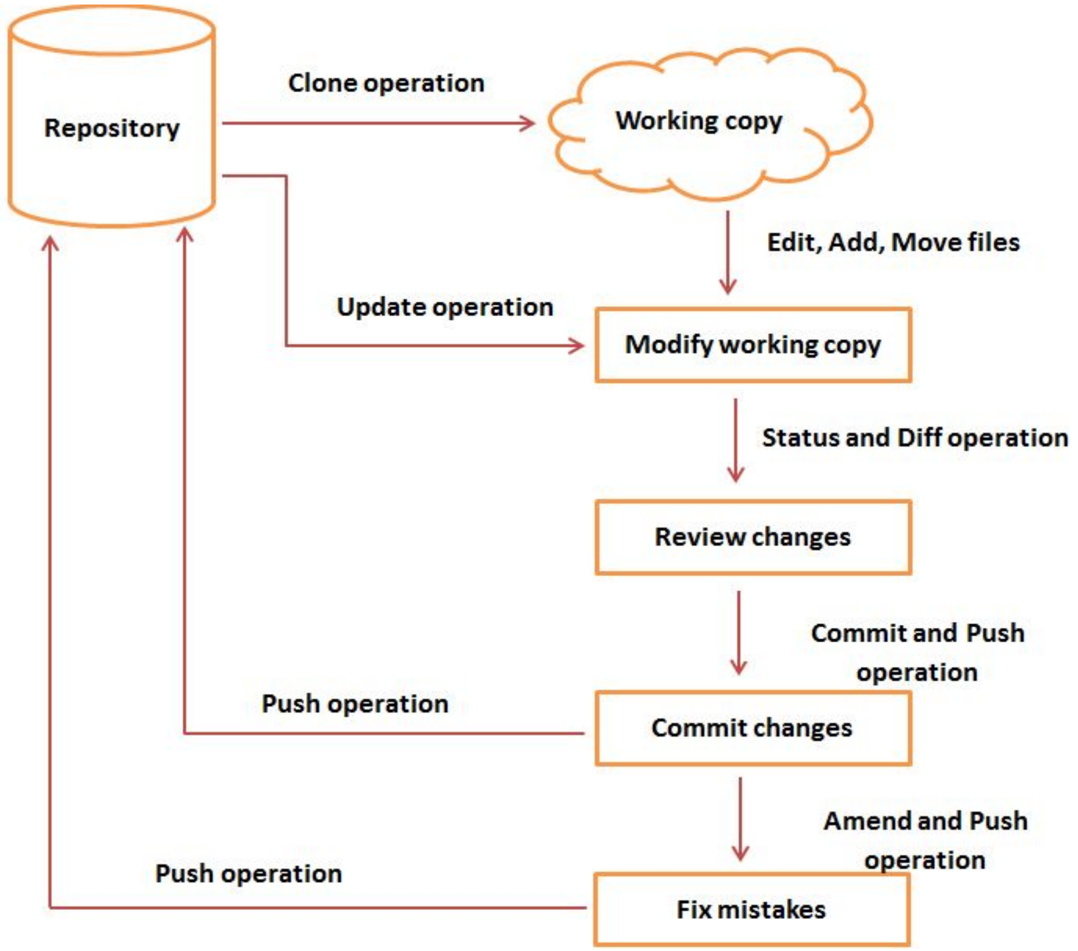
5. Create a checkpoint

- ***git commit -m "<description>"***
- A checkpoint in git is a **commit**
- When you have all changes you want in the staging area, then commit.

6. Seeing the commits

- We are able to keep track of the commits we make
- Take a peek through history
- ***git log***
- The hash for every commit is a unique reference to that commit

Git Life Cycle



Other Commands

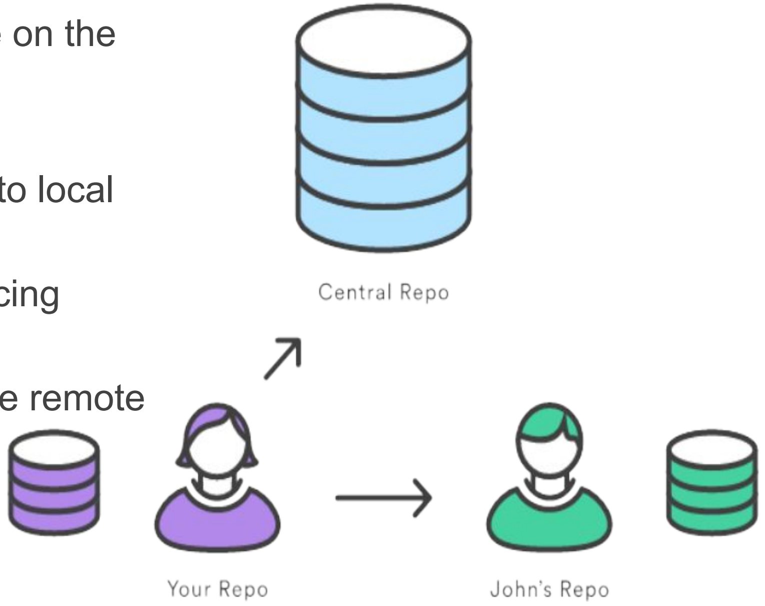
- ***git diff*** - Generates a diff between current working directory and last commit
- ***git reset*** - Cleans staging area
- ***git reset <hash>*** - Resets history to commit represented by hash and cleans staging area
- And many more ….

GIT REMOTES

How to use git to share code ?

General Scenario

- One remote repository (somewhere on the internet)
- Contributors have local copies
- Commits are made by contributors to local git repo
- The remote repo is updated by syncing these commits (**push**)
- Local repos are also synced with the remote repos (**pull**)



Create / Get a remote repo

Create

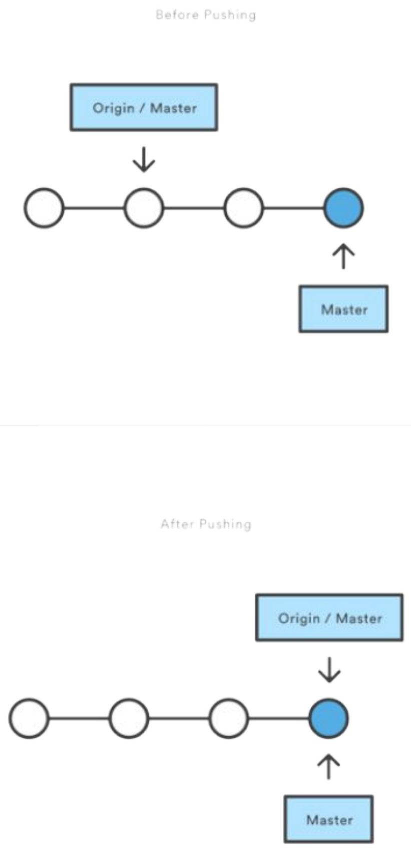
- Create a repo on the internet (GitHub is one place!)
- Add a reference to the remote repo to your local repo
- git remote add <remote_name> <remote_url>

Get

- Instead of creating a fresh repo, you might need to work with an existing one.
- git clone <remote_url>
- Cloning a repo gets you a local copy of the remote repo
- It’s remotes will already be configured

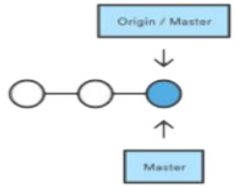
2. Send commits

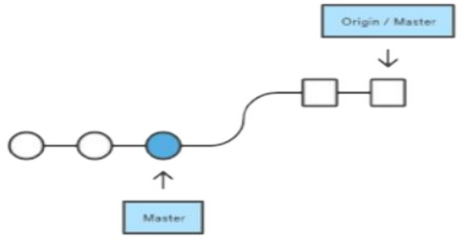
- Initially, our remote repo doesn't have our new local commits
- We need to send them across
- **push** is done to send local changes to remote repo
- ***git push <remote_name> <branch_name>***
- Repeat after more commits to send those also!

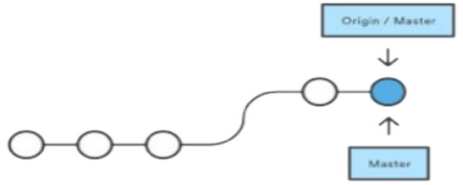


3. Get your remote commits

- git lets you have more than one local repo.
- You need to get your new remote commits onto a unsynced local repo!
- ***git pull <remote_name> <branch_name>***
- **pull** is done to get remote repo to local changes







GIT BRANCHING & MERGING

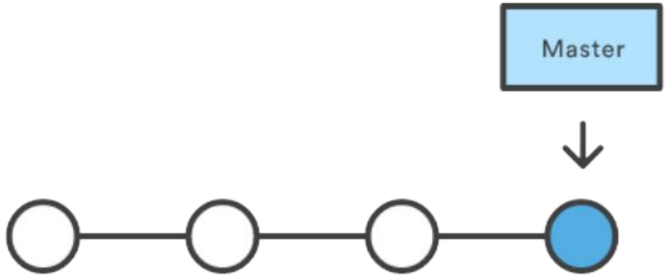
Using git to develop several features at once

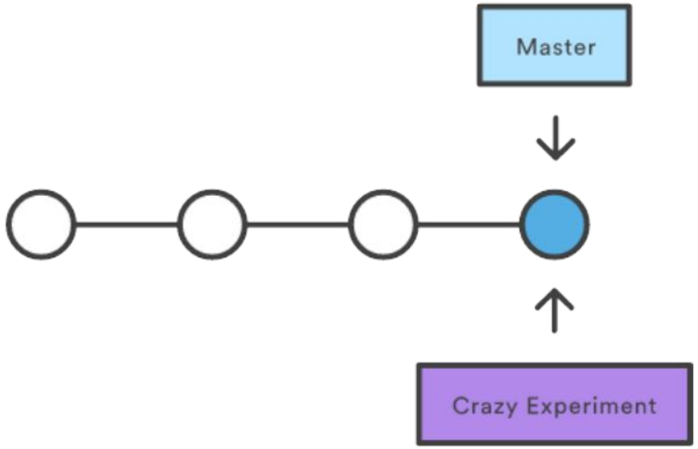
Branches

- Every git repo has a default branch called master
- Branches are a way to have separate commit trees
- Once branched, only commits made on that branch effect it.
- The developer can work on the feature in isolation
- The commits can be merged back to master when completed

1. Create a Branch

- ***git branch <branch_name>***

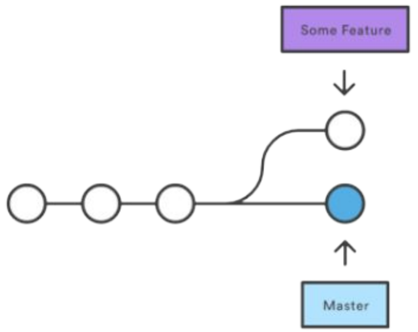




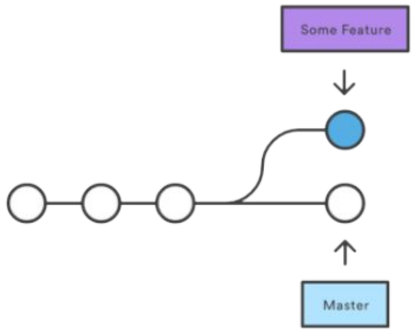
2. Change between branches

- ***git checkout <branch_name>***
- The above command is used to move between branches
- Essentially, you will be changing commit histories here.
- A git log will show you the differences!

Checking Out Master



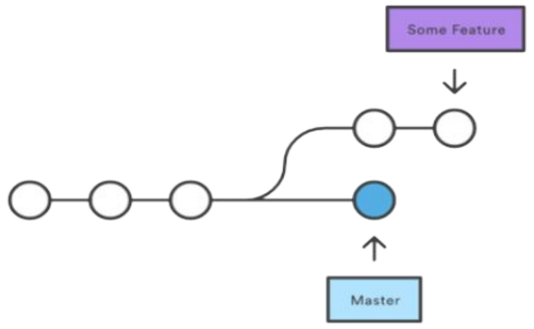
Checking Out Some Feature



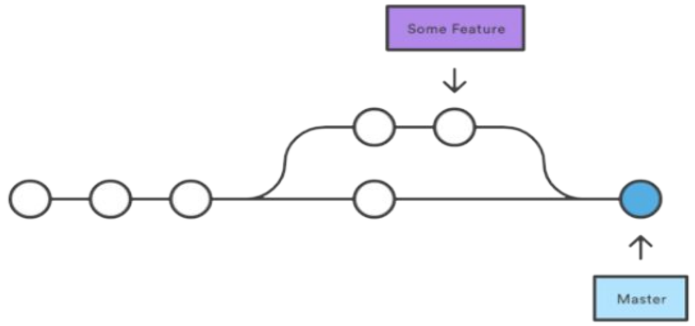
3. Merge

- Use this to merge the 2 trees.
- Get your feature into master!
- Conflicts can arise here.

Before Merging



After a 3-way Merge



Branching : Things to take care

- Do not commit to master.
- Make branches. Commit. Merge back.
- Makes development clean
- Great way of managing many people working on one project

GitHub

How to use github ? How is it different from git ?

Github

- GitHub is a company that lets you host your code
- GitHub is currently the most popular code hosting website.
- Lot’s of major open source projects are on GitHub now.
- GitHub has good tutorials on most of its features.
- GitHub repositories are git repos
- Just ***git clone*** the url!

Forks

- You don’t have write permissions to origin ? How do we contribute then ?
- Forking means you create a copy of the original repository in your profile.
- You have write access to the fork!
- So add your fork as a remote, and push and pull to that remote!

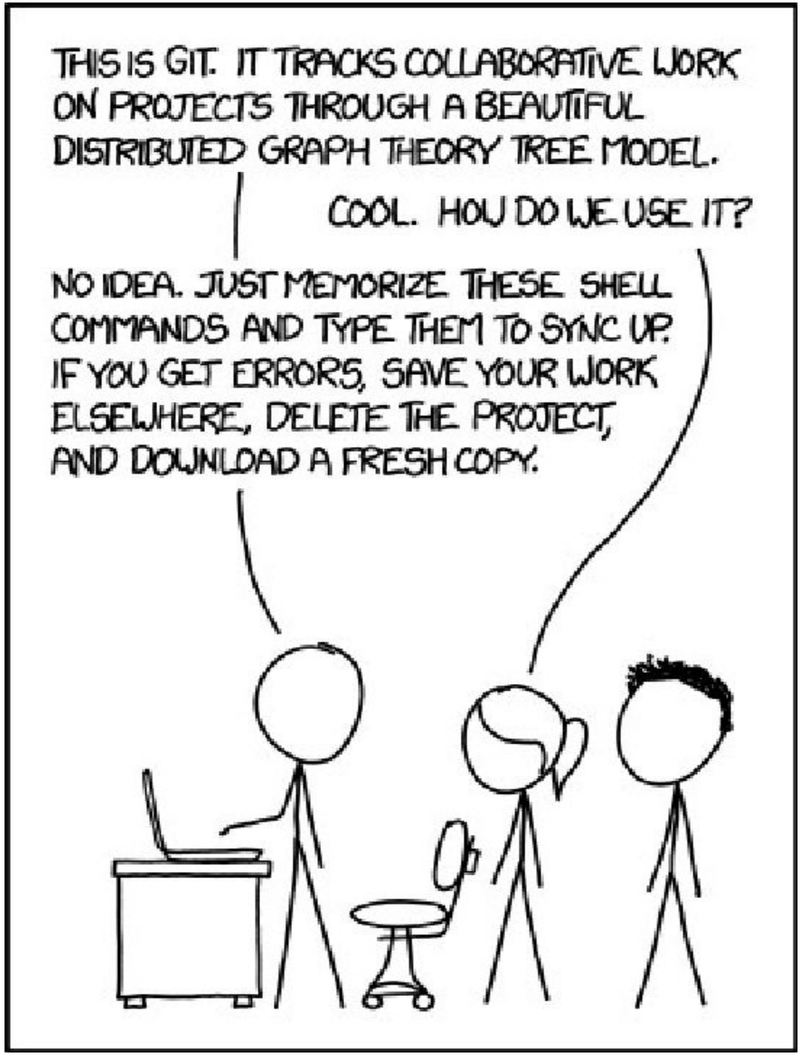
Pull Requests

- How do you get the original repo to see my contributions ?
- Create a pull request in the original repo with a short description of the changes you have made.
- Maintainers will comment on it, make you refine it till they are happy with it and then merge it!

Issues

- GitHub has an issues facility for their repositories.
- As a user you can file your bug reports/worries/ ideas about the repository in the issues.
- As a developer, you can look through the issues and try and fix some of them!
- Look for labels to figure out beginner level ones or ones in your area of interest.

Thank You!


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.