git init:: creates a repo (git storage area)

git add adds the file to index, git commit then sends it to repo

git diff commit1 commit2

git checkout commit1

git branch Name

git switch BranchName

git merge BranchName

Jira: tracks work, allocates tasks

Jenkins: builds and packages the system, deploy to testing/prod, monitor, automated tests

Why use Git? fast, safe

History:

Subversion (client-side version control) aka Centralized Version Control System

Distributed aka Concurrent Version System (Git)

-she clones the repo

-maintains a link to the shared repo (remote)

-git pull: gets the latest commits from the remote repo to local repo. Push is the other way

-git environment is a peer to peer system

Peer to Peer (P2P)

-means each computer can act as a server/client for others

-decentralized communications model where each party has the same capabilities

Github is a GUI, What are github/gitlab/bitbucket

Instead of getting local clone of repo, we want a remote clone inside Github (Fork) git fork is NOT a command. Fork is a feature/concept that Github provides

Difference between git clone

-forking creates your own repo in a remote location (on Github). Own copy means you can make changes to your copy without affecting the original. While cloning makes a local copy of repo, not your own copy

So don’t clone the other person’s because push your changes to their repo without permission. Clone your forked one then do a PR

So clone doesn’t make your own repo, fork does

Getting Started with Git 2

Git

-VCS: software designed to record changes over time

Windows: command prompt or powershell

What is git bash?

ls -> dir

touch -> con

git config

--global user.name “Seong Moon”

--global user.email [sjm@gmail.com](mailto:sjm@gmail.com)

--list

Tip: don’t use global if you want different settings for one project

man git

porcelain (high-level, most used)

plumbing (low-level, meant to used in scripts, advanced)

git help config (takes you to config)

git help (shorter list of porcelain)

.git contains metadata

-To use SSH urls, you needs SSH key pair and then add public key to your Github account

git add -> adds files to staging area

git remote add origin (HTTP-URL)

git push -u origin master

Basic Commands of Everyday Git?

A branch in Git, is a lightweight movable pointer to your project at a specific point in time

A screenshot of a short status

Description automatically generated

Stages: tracked and untracked

Tracked: 3 states: committed -> modified -> staged

open -a atom <filename>

git status -s OR –short

git diff –staged (compares to committed and staged)

Ned to go back on it

A diagram of a diagram

Description automatically generated with medium confidence

--mixed is the default option

How Git Works

Porcelain: basic commands

Plumbing: low level

Values: sequences of bytes

A diagram of a diagram

Description automatically generatedKeys: git calculates hashes using SHA1

Ex: echo “Apple Pie” | git hash-object –stdin

Output: a long hash

Branches Demystified

-A branch is just a reference to a commit

Fast Forward

-git wants to save space/commits. It happened on git merge main (on idea branch)

Detached HEAD

-when the HEAD is not point to a branch instead a commit, so must git checkout COMMIT

A diagram of a diagram

Description automatically generatedEx

1. Checkout a commit

2. Made multiple commits then checkout to main

3. The commits are isolated, can only reach it by directly their hashes

4. GC will delete them soon

To save them, create a branch on Detached HEAD

-remember if you want to create a reference point with a detached HEAD, use git branch NAME otherwise garbage collector might delete them. Also, if you don’t have a branch reference then only SHA id is how

Rebase

-the branch you are on gets detached and gets put on top of the branch you are git rebase branch (WAIT its too simple)

-changes the base

-A rebase can be fast-forwarded just like a merge

Rebase

-makes new commits with new hashes, because commits are database objects (immutable)

-if you change anything in a commit, you will get a different hash which means different commit

-git will GC unreachable commits(objects) afterwards

Merge Tradeoffs

-preserve history as exactly as it happened but can look complicated

-if conflicts, merge commit would includes fixes to the conflict

Rebase Tradeoffs

-refactor history (looks cleaner but a lie)

Tags

-can be likes branches but just a reference. So can’t contain data, just a name

-reference points to a database object of type tag

Annotated Tags: branch like objects that points to a tag object that points to a commit

Lightweight Tags: just points to a commit (doesn’t have metadata)

Lightweight Tags: just points to a commit (doesn’t have metadata)

A diagram of a remote work flow

Description automatically generated-We will lose the commit by GC so use force pushing carefully

-Also the history will be very conflicting

-It doesn’t solve conflicts, just puts the burden on other users to solve conflicts

-Instead do a git fetch then merge then push

A diagram of a diagram

Description automatically generatedPull is fetch and merge in one

A blue background with white text

Description automatically generatedA diagram of a diagram

Description automatically generated

A diagram of a cloud with text

Description automatically generated

Working with Git Branches

Added most commands in Notes

Fast Forward

-when the reference for main simply moves forward to the last commit on the branch you are merging (think about watching a movie on multiple devices, it will FF you where you left off)

A close-up of a code

Description automatically generated

-Target: refers to the branch you want to modify by merging changes in

-Source: refers to the branch that has changes you want

git diff branch1 branch2

A screenshot of a computer

Description automatically generated

git diff // only looks at unstaged changes

git diff –cached //see only staged changes

git diff HEAD //staged and unstaged changes

git diff <commit> //shows its one file?

git show <commit>

git diff branch1…branch2

git diff branch1 branch2 fileName

A screenshot of a computer program

Description automatically generated

git merge --abort

A screenshot of a computer

Description automatically generated

A diagram of a remote work flow

Description automatically generated

-fetch: will download from remote

Pull: (fetch & merge): will download and initiate a merge on your local branch

A screenshot of a computer

Description automatically generated

-u set the origin as upstream flag

A screenshot of a workflow

Description automatically generated

A screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated

A logo of a book

Description automatically generated

A screenshot of a screen

Description automatically generated

Squash

-Git copies all of the changes from each commit into that new commit

A screen shot of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

A screenshot of a black screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

git rebase main

Rebase branch from main

1. Any changes in main, will be pulled over
2. Your commits from other branch will be put on top

-this help create a linear history when you merge back to main

-rebase can’t just move your old commits, it will create new ones

A diagram of a network

Description automatically generated

Git cherry-pick

-appends any commit to the working head of the branch you are in

-does not move the commit, but it creates a copy of the commit but with a new parent in the branch you’re cherry picking into

Rewriting Git 2.X History

A black text on a white background

Description automatically generated

A diagram of a diagram

Description automatically generated

A diagram of a diagram

Description automatically generated

A close up of a logo

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Github Fundamentals

SSH

-secure shell

-protocol that allows for secure communication

-won’t be prompted to enter credentials



-a: list hidden files/directory

-l: list files/directory in a table format with extra information

A black and orange rectangular object with white text

Description automatically generated

-Generate the key then add it to Github by copying the id\_rsa.pub

A computer screen shot of white text

Description automatically generated