Git & GitHub

Introduction

- Version control is a process for tracking and managing changes to software code or other files. It's also known as source control or revision control. Version control systems (VCS) are software tools that help with this process

Example

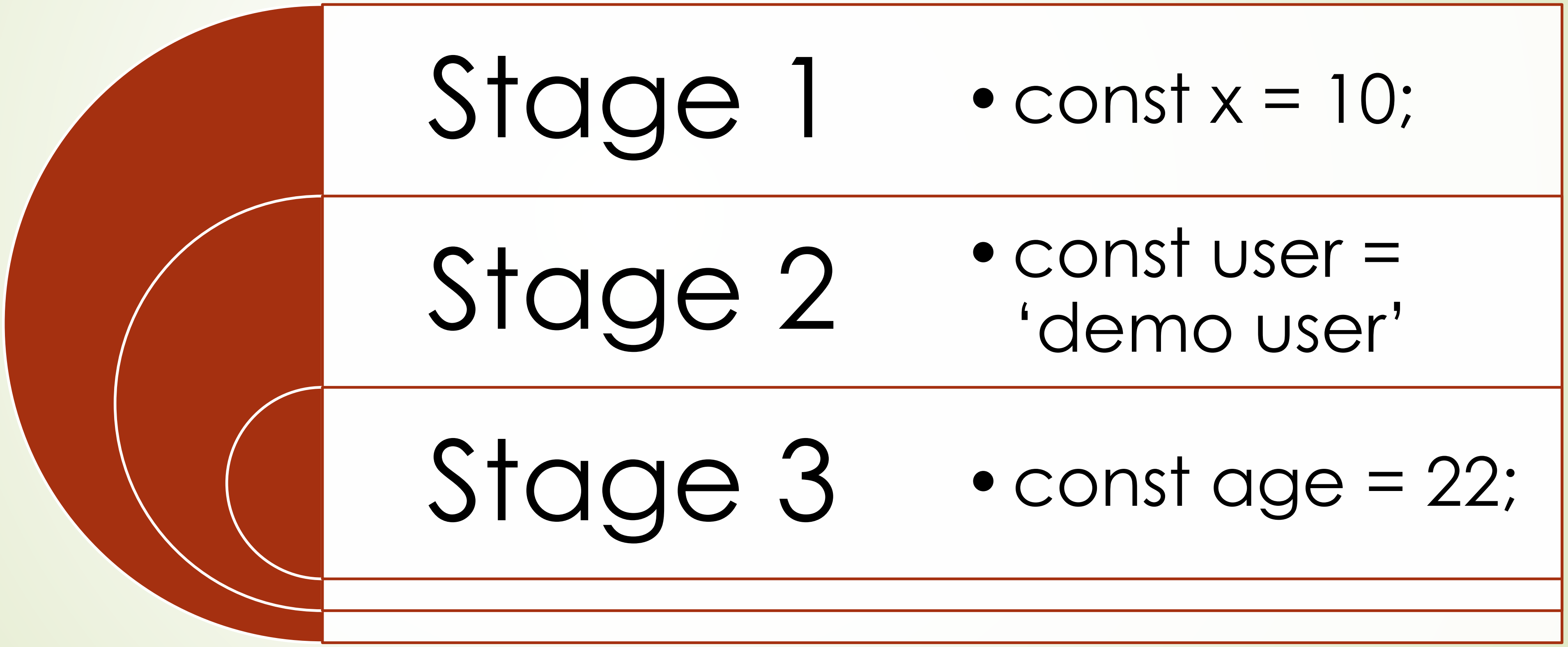
- File sample1.js

```
const x = 10;

const user = ‘demo user’

const age = 22;
```

Version Controlling / Tracking



Stage 1	• const x = 10;
Stage 2	• const user = 'demo user'
Stage 3	• const age = 22;

Version Control Systems

- Git
- Apache SubVersion
- Piper

What is GIT ?

- Git is a free, open-source version control system (VCS) that helps users store, track, and manage code.
- https://git-scm.com/

- Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

- Check version with git -v

GIT CLI

- Setup 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

git config --global color.ui auto

set automatic command line coloring for Git for easy reviewing

- Create new Folder and open folder in editor
- Create some files with editor

SETUP & INIT

Configuring user information, initializing and cloning repositories

git init

initialize an existing directory as a Git repository

- Now your newly created files are marked is **U** untracked.
- Now all the git config stored in .git hidden folder in your current folder.

Add file for tracking

git status

show modified files in working directory, staged for your next commit

git add [file]

add a file as it looks now to your next commit (stage)

git diff

diff of what is changed but not staged

Add All files to git add

- Use command git **add . (git dot)** To add all files of current file to git add.

git rm [file]

remove a file from git tracking.

Commit

- The git commit command **captures a snapshot of the project's currently staged changes**. Committed snapshots can be thought of as "safe" versions of a project—Git will never change them unless you explicitly ask it to.

git commit -m "[descriptive message]"

commit your staged content as a new commit snapshot

git log

show all commits in the current branch's history

- Add some new code in your files. Your git will track all changes.

git diff

diff of what is changed but not staged

git add [file]

add a file as it looks now to your next commit (stage)

git commit -m “[descriptive message]”

commit your staged content as a new commit snapshot

Log

- **git log**
- show the commit history for the currently active branch
- **git log --oneline**
- show the commit history for the currently active branch

Show

- **git show [SHA]**
- show any object in Git in human-readable format

blame

- **git blame filename**
- show any object in Git in human-readable format

Staging Areas

- The staging area is **a file, generally contained in your Git directory, that stores information about what will go into your next commit**. Its technical name in Git parlance is the "index", but the phrase "staging area" works just as well.

git status

show modified files in working directory, staged for your next commit

git add [file]

add a file as it looks now to your next commit (stage)

git commit -m “[descriptive message]”

commit your staged content as a new commit snapshot

Revert Back

- Read HEAD to desired commit to revert back to previous code.
- HEAD always point latest commit.

HEAD -> main

- Git log –oneline
- Git reset –hard Commit ID
- Git log –oneline

- How it works. The git revert command is **used for undoing changes to a repository's commit history**. Other 'undo' commands like, git checkout and git reset, move the HEAD and branch ref pointers to a specified commit. Git revert also takes a specified commit, however, git revert does not move ref pointers to this commit.

- Git revert Commit ID
- Try GIT Graph Extension for visualize you git repo.

GIT and GITHUB

Local PC with GIT Setup

Local PC with GIT Setup

Local PC with GIT Setup

Index.js

let x = 20

Local PC with GIT Setup

Index.js

let x = 50

Single Source of Truth

Local PC with GIT Setup

Index.js

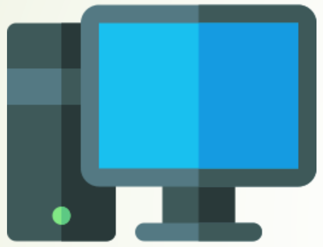
let x = 20

Local PC with GIT Setup

Index.js

let x = 50

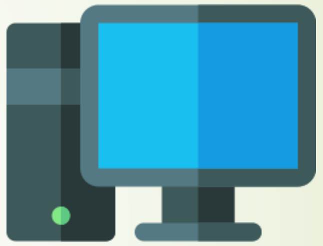
Single Source of Truth
Remote Server



Local PC with GIT Setup

Index.js

let x = 20

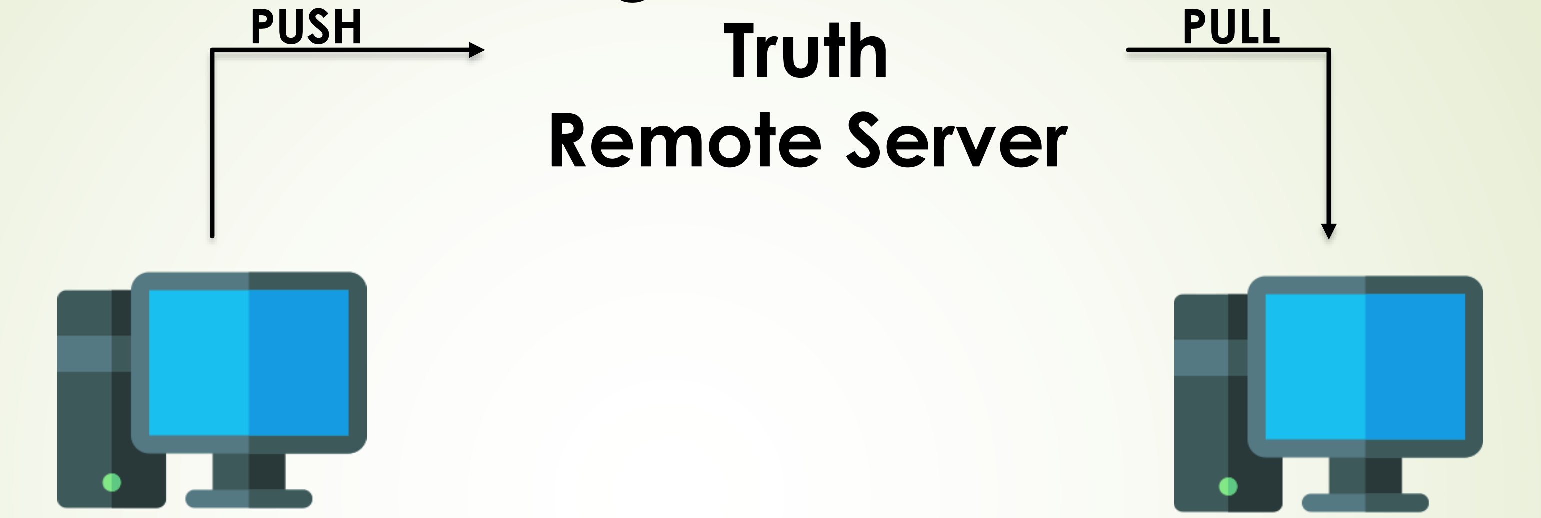


Local PC with GIT Setup

Index.js

let x = 50

Single Source of Truth
Remote Server



Local PC with GIT Setup

Index.js

let x = 20

Local PC with GIT Setup

Index.js

let x = 50

- GitHub is Public Server for provide remote server to store files.
- also use gitlab, bitbucket

Git remote

- Open github.com
- Create account
- And create new repository for practice
- Now we sync our local git code to remote git server (github)

git remote add [alias] [url]

add a git URL as an alias

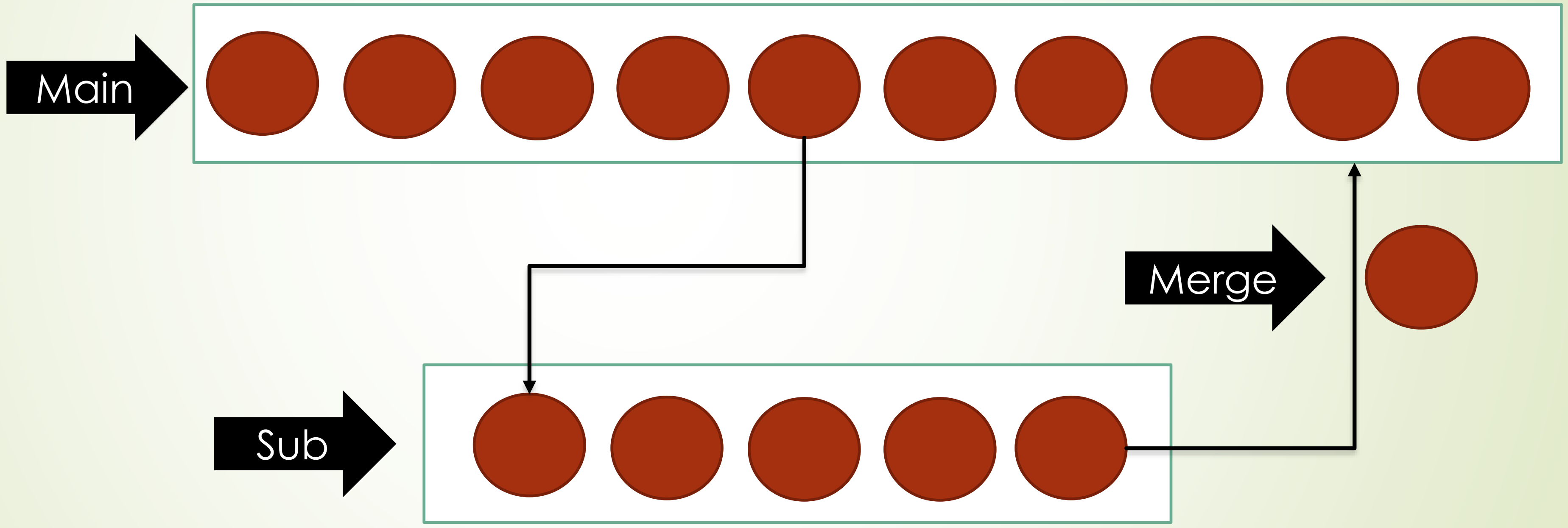
git push –u server branch

You can also used multiple remote repo

- SSH Key Generate for secure file transfer.
- Open your github account and go to settings and select SSH and GPG keys.
- Create new SSH key

Branching

- Git branching is a feature that allows developers to work on different parts of a project simultaneously without affecting the main branch. Branches are separate instances of the project, and developers can switch between them without interfering with each other's work. When the work is complete, the branch can be merged with the main project.


Main
Merge
Sub

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 checkout

switch to another branch and check it out into your working directory

git log

show all commits in the current branch’s history

- Make some changes in project and try to commit and push it will give an error.
- git push –set upstream server branch-name

Merge Branch

- Get checkout main
- Git branch
- Git merge server branch
- Git status
- Git push
- Git log

New branch another way

- Git checkout –b ‘branch-name’
- Git add .
- Git commit –m ‘new branch added’
- Git push
- Git status

- Open github.com
- Go to pull request
- Select branch
- And select main
- Add title and pull it.
- Merge pull request

- Open git
- Git checkout main
- Git pull

Branch names

- **Naming conventions for Git Branches**

- **Group tokens**
- Use "grouping" tokens in front of your branch names.

- **Short well-defined tokens**
- Choose short tokens so they do not add too much noise to every one of your branch names. I use these:

Wip

- Works in progress; stuff I know won't be finished soon

Feat

- Feature I'm adding or expanding

Bug

- Bug fix or experiment

Junk

- Throwaway branch created to experiment

Git branch “feat/feat-A”

Git checkout “feat/feat-A”

Or

Git checkout –b “feat/feat-A”

Testing

- https://git-school.github.io/visualizing-git/
- https://learngitbranching.js.org/

REWRITE HISTORY

Rewriting branches, updating commits and clearing history

git rebase [branch]

apply any commits of current branch ahead of specified one

git reset --hard [commit]

clear staging area, rewrite working tree from specified commit

Stash

TEMPORARY COMMITS

Temporarily store modified, tracked files in order to change branches

git stash

Save modified and staged changes

- When you modify data on remote directly and try to pull repo you will get an error about fetch repo about un staged changes.
- Git stash used to get changes in tmp folder.
- Get stash apply