CSCI 6030: Frontend Web Application Development

Version Control, Git, and GitHub

Version Control and Git

- Version control, also known as source control, is the practice of tracking and managing changes to software code
- Version control systems are software tools that help software teams manage changes to source code over time
- The code for a project, app or software component is typically organized in a folder structure or "file tree"
- Git is a popular version control system. It is used for:
  - Tracking code changes
  - Tracking who made changes
  - Coding collaboration

Image source: https://git-scm.com/

Git and GitHub

- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.
- You can download Git for free from the following website: https://www.git-scm.com/
- You can create a free account on GitHub and also download the software package from: https://github.com/

Image sources: https://git-scm.com/; https://github.com/

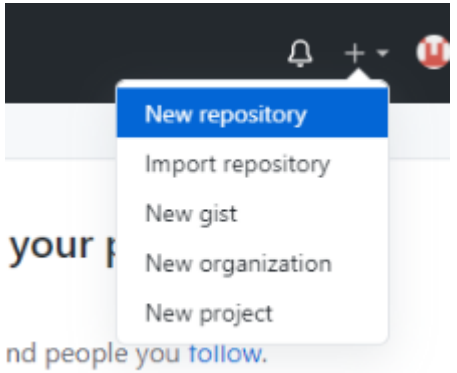
GitHub Setup and Basic Git Commands

- Easiest way to configure GitHub in your computer is to setup their Software Package and log in
- Check Git version:
  - git --version
- Configure User:
  - git config --global user.name "John"
  - git config --global user.email "John@google.com"
- Navigate to project folder
  - mkdir project_folder
  - cd project_folder
- Initialize Git
  - git init

GitHub Setup and Basic Git Commands

- Easiest way to configure GitHub in your computer is to setup their Software Package and log in
- Check Git version:
  - git --version
- Configure User:
  - git config --global user.name "John"
  - git config --global user.email "John@google.com"
- Navigate to project folder
  - mkdir project_folder
  - cd project_folder
- Initialize Git
  - git init

Basic Git Commands



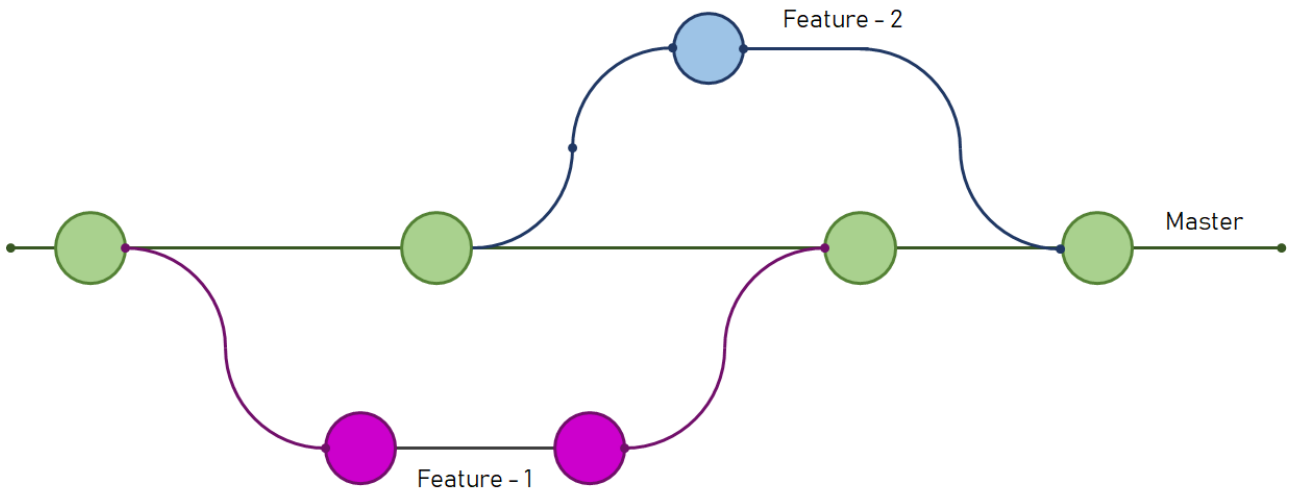
- You can create New Repository in your GitHub account
- Configure the repository to be public or private
- Add **readme** and **gitignore** file
- The **readme** file should explain the details of your project
- The **gitignore** file lists the files and directories that Git should ignore storing/keeping track of

Pushing local change to Remote Repo

- Add remote connection/origin
  - Syntax: **git remote add origin URL**
  - Example: git remote add origin https://github.com/john/hello-world.git
- Check local status
  - **git status**
- Stage all changes
  - **git add –all**
- Commit changes
  - Syntax: **git commit –m “commit message”**
  - Example: git commit –m “Added index file”
- Push changes
  - **git push origin**

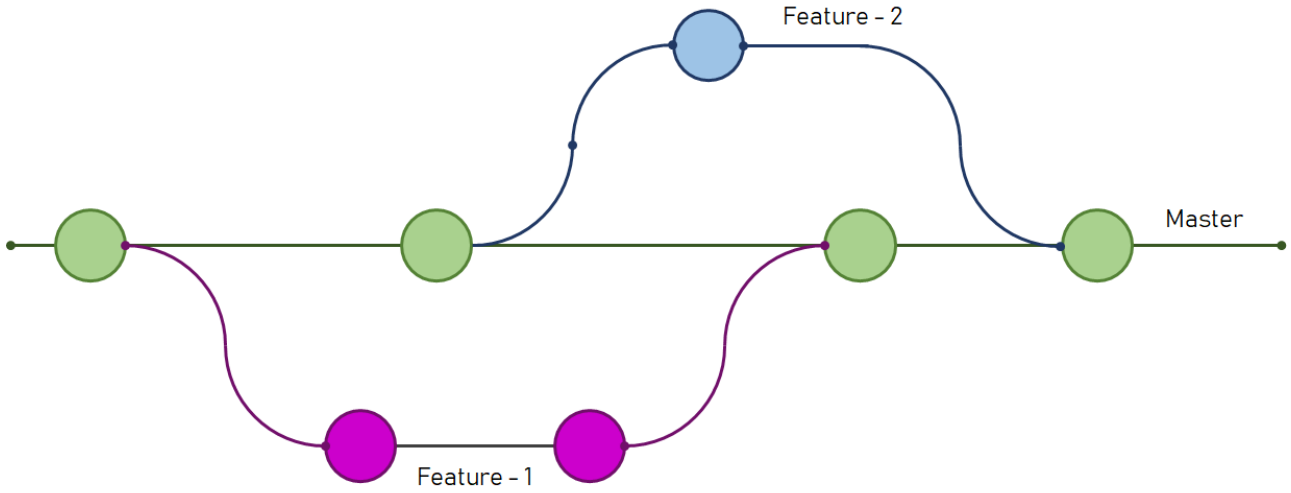
Git Branches

- Branches allow you to work on different parts of a project without impacting the main branch.
- When the work is complete, a branch can be merged with the main project.
- You can even switch between branches and work on different projects without them interfering with each other.

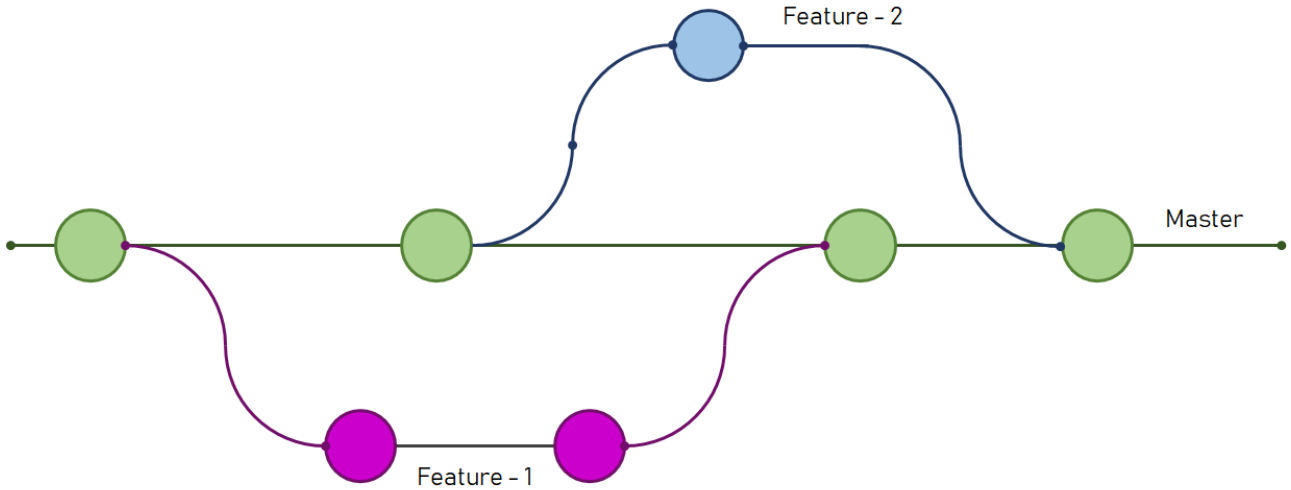


Git Branches

- List branches:
  - **git branch**
- Create new branch
  - **git branch <branch_name>**
  - Example: git branch feature/homepage
- Switch branch
  - **git checkout <branch_name>**
  - Example: git checkout feature/homepage
- Create and switch
  - **git checkout –b <branch_name>**
  - Example: git checkout –b feature/homepage

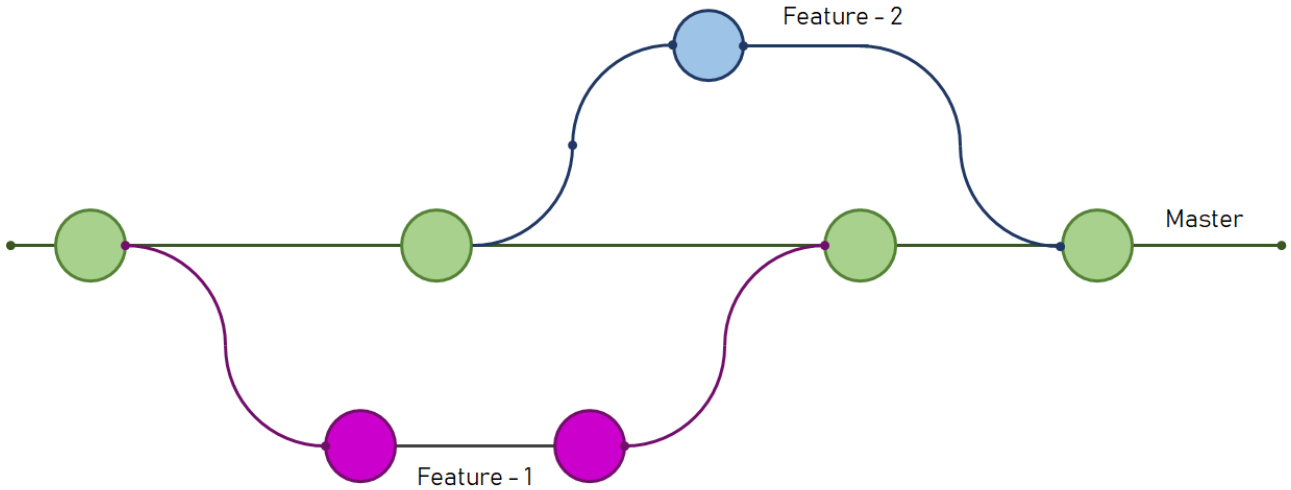


Git Branches



- Delete branch:
  - **git branch -d <branch_name>**
  - ‘-d’ means deleting the branch only if the branch is pushed and merged with the remote branch
- Force delete:
  - **git branch -D <branch_name>**
  - ‘-D’ means deleting forcefully without checking whether the branch is pushed or not
- Remove remote branch
  - **git push origin --delete <branch_name>**

Git Branches



- Delete branch:
  - **git branch -d <branch_name>**
  - ‘-d’ means deleting the branch only if the branch is pushed and merged with the remote branch
- Force delete:
  - **git branch -D <branch_name>**
  - ‘-D’ means deleting forcefully without checking whether the branch is pushed or not
- Remove remote branch
  - **git push origin --delete <branch_name>**

Common Git Commands

- **git init** initializes your local directory as a new git repository. You must run this before you can commit any of your work.
- **git status** shows the current status of your repo. It will show you if you have any work that is unstaged, what branch you are on, how many commits you are ahead of the master remote on github, and other useful things.
- **git diff** shows you the changes in your unstaged code.
- **git remote -v** shows you all the remotes for your repo. The v stands for verbose, which shows you the URL of the repository on github, if any, that your local repository is pointing to rather than just the name of the remote repo.
- **git add .** takes all unstaged work and stages it, making it ready to be committed. You can also specify a particular file to stage with git add file-path/name-of-file
- **git commit -m** "write commit message here" commits all staged work. It's important to write a brief, clear commit message so you know what each commit is for. "Final commit" is not the commit message you're looking for exactly 100% of the time.

Sources: https://frontend.turing.edu/lessons/module-1/git-commands.html

Common Git Commands

- **git commit -m "commit message"** commits all staged work. It's important to write a brief, clear commit message so you know what each commit is for. "Final commit" is not the commit message you're looking for exactly 100% of the time.
- **git pull** once you've committed all your local work and running git status shows that you have nothing to commit, you pull down any changes from your remote. By default, this will pull from the origin remote's master branch. To be specific about which remote and branch to pull from, you can use: git pull name-of-remote name-of-branch
- **git push** pushes your local changes up to your remote. By default, this will push to the origin remote's master branch. Like pull, you can push to a specific remote and branch with: git push name-of-remote name-of-branch. This is useful if you are using branches and pull requests. If you get an error message, it's probably because you haven't pushed your local branch up to github yet. Try git push -u name-of-remote name-of-branch.
- **git branch** shows you all your local branches and indicates which branch you are currently on.
- **git checkout -b** name-of-new-branch makes a new branch and switches to that branch.

Sources: https://frontend.turing.edu/lessons/module-1/git-commands.html

Common Git Commands

- **git merge name-of-branch** will merge the specified branch into the branch you are currently on.
- **git branch -d name-of-branch-to-delete** deletes the specified branch
- **git log** will show you the full list of commits and authors for your repo
- **history** will show you your past git commands
- **git stash** stashes any unstaged changes in your repository. They will not be present in your codebase, but they are not deleted.
- **git stash pop** gives you back the last staged changes you stashed
- **git blame file-path/name-of-file** shows you line-by-line who wrote the code in the specified file. Useful when you have a question about how something works and want to figure out who to ask, and also great source of shame when you realize you wrote the chunk of code you've been swearing at for the last hour.

Sources: https://frontend.turing.edu/lessons/module-1/git-commands.html

Commands & Tutorials

- Detailed tutorials and list of commands:
  - https://git-scm.com/docs
  - https://www.w3schools.com/git/default.asp
  - https://www.atlassian.com/git/glossary#commands
  - https://frontend.turing.edu/lessons/module-1/git-commands.html

Image sources: https://git-scm.com/; https://github.com/