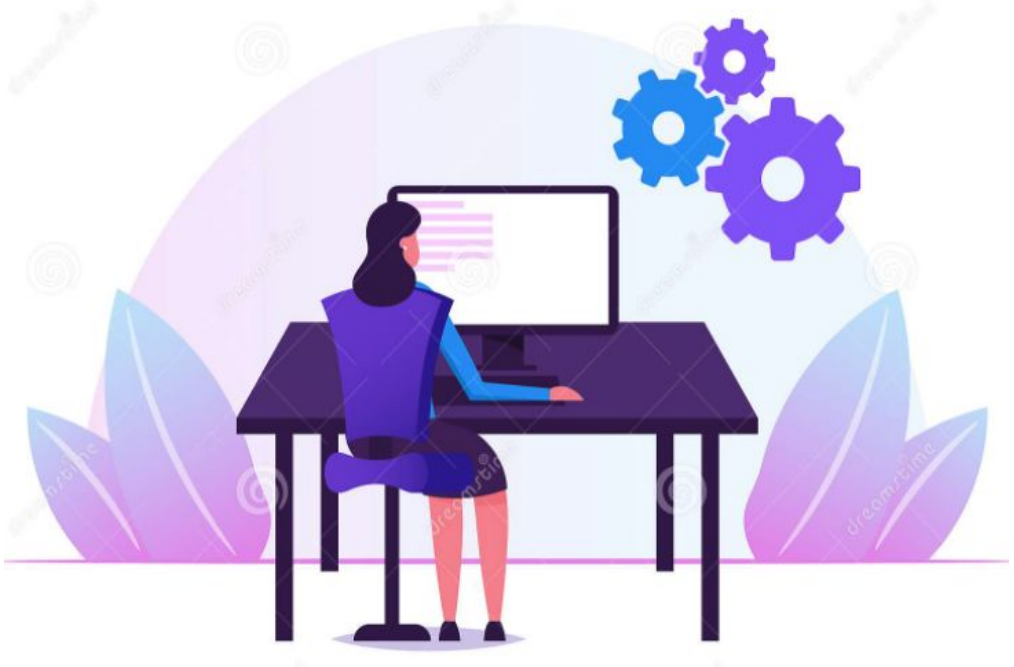
Welcome!



WWCode San Francisco - Backend Study Group

December 8, 2022

- We'll start in a moment. :)
- We are **RECORDING** tonight's event.
- We may plan to take screenshots for social media.
- If you are comfortable, turn the video ON. If you want to be anonymous, then turn the video off.
- We'll introduce the hosts & make some time for Q&A at the end of the presentation.
- Feel free to take notes.
- Online event best practices:
    - Don't multitask. Distractions reduce your ability to remember concepts.
    - Mute yourself when you aren't talking.
    - We want the session to be interactive.
    - Use the 'Raise Hand' feature to ask questions.
- **By attending our events, you agree to comply with our Code of Conduct.**

WOMEN WHO CODE /san-francisco

Introduction & Agenda

- Welcome from WWCode!
- Our mission: Empower diverse women to excel in technology careers.
- Our vision: A tech industry where diverse women and historically excluded people thrive at any level.

Prachi Shah
Instructor,
Senior Software Engineer.
Director, WWCode SF

Anjali Bajaj
Host,
Volunteer, WWCode SF

- Git and Version Control System:
- What is Version Control?
- What is Git?
- Introduction to Git commands.
- Introduction to Git flow workflow.
- Q & A.

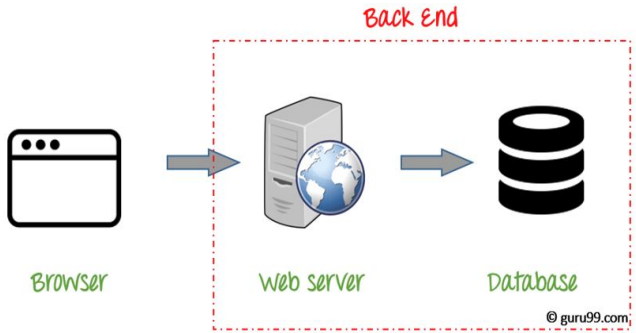
Disclaimer:
- Sessions can be heavy!
- Lots of acronyms.
- Instructor doesn't know everything.

WOMEN WHO CODE /san-francisco

Backend Engineering

- Design, build and maintain server-side web applications.

- Common terms: Client-server architecture, networking, APIs, web frameworks, platform, micro-service, databases, web fundamentals, operating systems, etc.



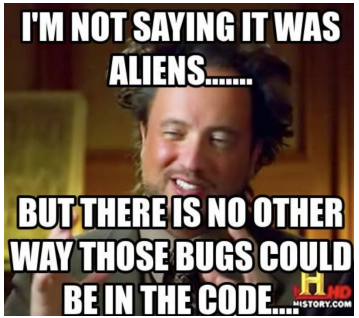
- Tech Stack: Java, PHP, .NET, C#, Ruby, Python, REST, AWS, Node, SQL, NoSQL, etc.

- Other domains: Front end engineering, full stack engineering, design & user experience, mobile development, devOps engineering, machine learning, etc.

Version Control

Source Code:

- Collection of code/programming statements written by a software engineer.
- Code can be in any high-level programming language. Example: Java, Scala, etc.



Source Code Management:

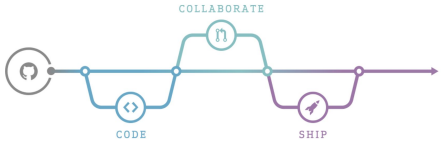
- Process of tracking and managing code changes in projects.
- Log a history of continuous code development.
- Contributions can be made by many developers simultaneously.
- Identifies code dependencies and resolves code merge conflicts.
- Easy to revert to a previous version of the code.
- Tracking makes it fast and efficient to track breaking changes. Example: Code bugs.
- Development teams can collaborate, get feedback, and assert quality.
- Crucial to the development step in the SDLC (software development lifecycle).
- Examples: GitHub, GitLab, Mercurial, MS TFS, Subversion, Bitbucket, etc.

Git

Git:

- Free and open source distributed version control system (VCS).
- Global Information Tracker • Track changes in a collection of different files (types).
- Distributed system where developers can clone the whole repository and view history of changes.
- Branching model for creating multiple local branches for independent code development.
- Fast development, testing and shipping code to production environment.
- Supported by various software development VCS systems like GitLab, GitHub, Bitbucket, etc.
- Git repository: Initializing a project with Git to enable version control and tracking.

GitHub:

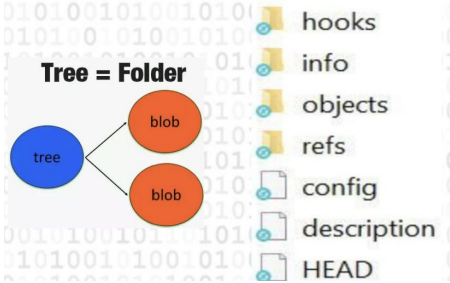


- Git server as a service. Online platform with features.
- Cloud-based hosting service to manage Git repositories.
- Access control, continuous integration, code feedback, continuous deployment for testing, etc.
- Branching strategies & best practices for continuous integration/ continuous development (CI/CD).
- Supports frameworks and languages like Java, Python, .NET, JavaScript, Ruby, iOS, Android, etc.
- In connection, Git is a tool and GitHub is a service for projects that leverage Git.

Terminology

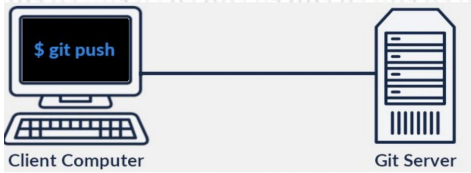
Repository:

- Initialize a project and creates .git/ folder inside a project.
- This folder tracks all changes and build version history.



- Changes: Add new files, delete files, rename files, and update files.
- Blob: Binary large object is the data structure that represents contents of a file.
- Tree: It is the directory structure and points to blobs (files).
- Snapshot: Snapshot of code changes at a given point in time for tracking.
- Commit: Commit is the creation of a snapshot. Every new snapshot change creates a new commit.
- main branch: Default branch name. A snapshot of the code that is deployed in production.
- Branch: A clone of the main branch for custom development. Feature branch for large code changes.
- Head: A named reference to the latest commit in a branch.
- HEAD revision: Most current version of code in the remote repository.
- Tag: Mark a commit in a branch for logging important commits.
- Rollback: Rollback or revert current changes to the previous commit in the branch.
- Merge and merge conflict: Integrate all development changes and resolve conflicts in your branch.
- Pull Request (PR): A collection of commits which are proposed new changes to be merged.

Development



Initialize a project to be a Git Repository:

- macOS command-line install: $ brew install git
- macOS GUI install: $ brew install git-gui
- Account: Create an account on GitHub.com to manage projects, role access and utilize features.
- In the project, run command $ git init which creates the .git folder and initializes the project.
- Make code changes and commit them. Push these changes to view the repository on GitHub.

Clone an existing Git repository:

- Create a Git repository-initialized project with a Git hosting platform (like GitHub).
- Clone the repository to the local workstation. This create a copy of the main branch.
  Note: Internally git clone git-initializes a project if not already done.
- Create a development branch to add new code changes.
- Make code changes to this branch and commit them.
- Push code changes to the this branch and create a pull request (PR). These are proposed new changes for the main branch.
- After development team feedback is applied, merge this PR to the main branch.

Commands

Create a Git repository:

- Inside the main project folder, $ git init
- Creates a HEAD and default main branch.
- Adds a .git/ subdirectory.
- master branch will be referred as main branch.

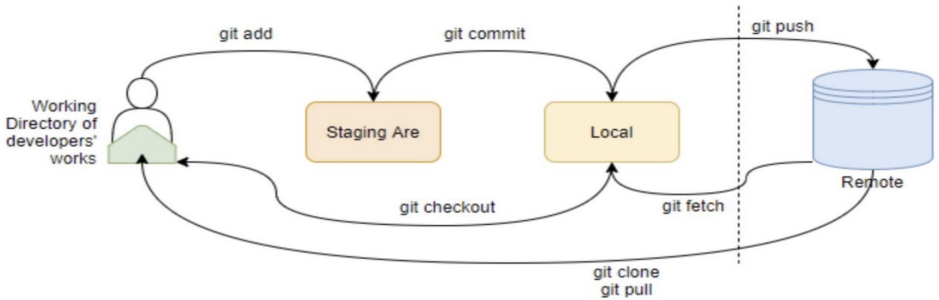
- Clone a remote repository locally: $ git clone https://repo_path.git
- Create a branch: $ git checkout -b my_first_branch_name creates a new local development branch.
- Stage changes:
  - Make file changes as needed (add, update, rename, delete files).
  - $ git add . adds all changed files. Changed files are staged for a commit.
- Commit changes: $ git commit “my first commit” commits staged changes to the branch.
- Push: $ git push <remote> <branch> uploads commits to the remote branch, and syncs changes.

After this step, a PR can be opened.

- Fetch: $ git fetch imports commits to the local branch.
- Pull: $ git pull downloads and applies commits to the local branch.
- Status: $ git status gives a status of all files changed in the local branch.
- Diff: $ git diff gives the diff changes of all files changed in the local branch.

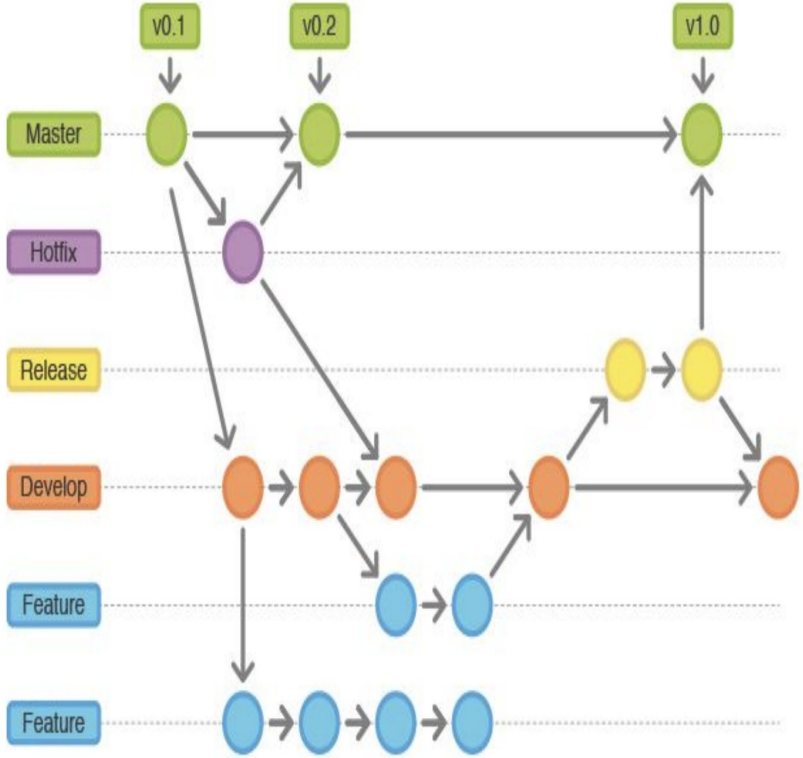
Git flow workflow

- Remote repository: Project hosted on a server.
- Local repository: Local copy of a remote repository, and used for local development.
- Methodology and best practices to integrate different branches (types).
- Best if used for frequent development, frequent maintenance, frequent release projects.
- Development Flow diagram.
- Types of branches:
    - Feature: New features, non-emergency bug fixes.
    - Release: Code ready for release.
    - Main: Code in production.
    - Develop: Contains latest release code.
    - Hotfix: Emergency bug fixes.



Git flow workflow

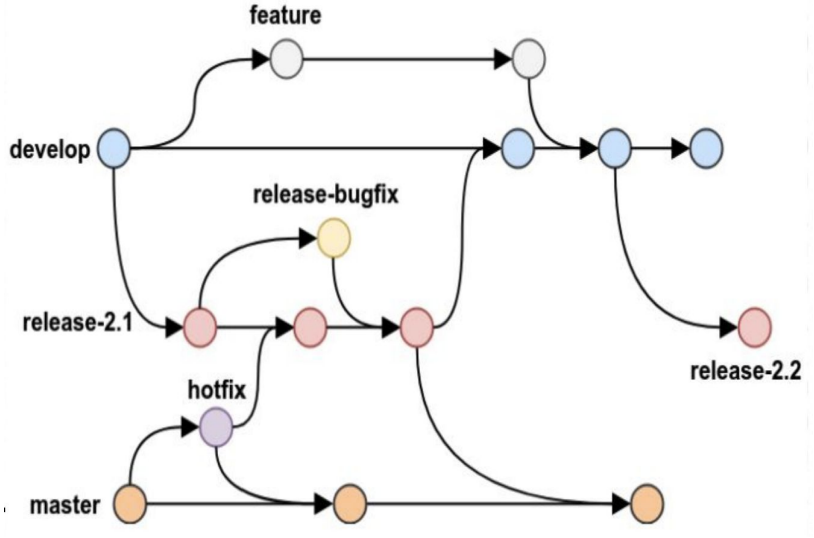
- feature branch is created for new development and branched off of the develop branch.
- Completed features and bug-fixes are merged into the develop branch when release-ready.
- A release branch is created off of the develop branch for retaining and releasing the code.
- Any bug fixes are done on the release branch.
- When release is ready, it is merged into main branch, and into the develop branch.
- main branch contains latest release code.
- develop branch contains latest “released” code changes for new development.
- hotfix branch contains emergency bug-fixes and is created off of the main branch.
- When hotfix is ready, it is merged into main branch, and into the develop branch.



Git flow workflow

Advantages:

- Parallel and isolated development with feature branches.
- Efficient for multiple-developers collaboration on specific feature (branch).
- develop branch is the staging are for all features therefore, a release branch off of the develop contains all recent changes/features.
- For frequent development, maintenance & release.

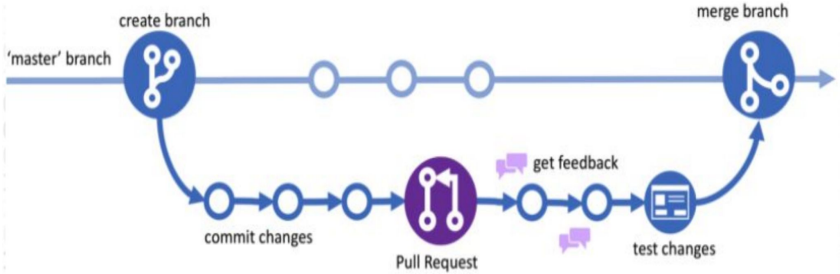


Disadvantages:

- No need for different branches if no major difference between release and main branches.
- Sometimes workflow for feature and hotfix branches can be the same.
- Additional complexity when release are numbered and release happens frequently.
- Multiple versions of an application are published, this can cause confusion when creating versioned release branches.

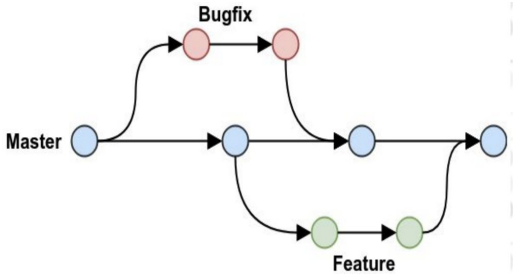
Git flow workflow

- Combines develop and release branches into main.
- feature and hotfix branch are the same.
- Advantages:
    - Quick changes can be made for CI/CD.
    - Development time is shorter.
    - Low branching strategy complexity.



Summary:

- Version Control: Tracking and managing source code.
- Git: Distributed VCS for simultaneous code development.
- GitHub: Platform to host and manage Git repositories.
- Git Flow workflows: Branching techniques for Git repositories.



Backend Engineering

References:

- Backend development
- Source Code Management
- About Git
- How Git works

Backend Study Group:

- Presentations on GitHub and session recordings are found on WWCode YouTube channel.
- System Design Series:
  - January 19th, 2023: Part 1 - System Design.
  - February 23rd, 2023: Part 2 - Design Considerations.
  - March 16th, 2023: Part 3 - Interview Questions.

Women Who Code:

- Technical Tracks and Digital Events for more events.
- Join the Digital mailing list for updates about WWCode.
- Contacts us at: contact@womenwhocode.com
- Join our Slack workspace and join *#backend-study-group*!

You can unmute and talk or use the chat.

WOMEN WHO CODE /san-francisco