Welcome!

- We'll start in a moment :)
- We are NOT recording tonight's event. We may plan to take screenshots for social media.
    - ***If you want to remain anonymous***, change your name & keep video off.
- We'll introduce the hosts and break in-between for Q/A.
- We will make some time for Q&A at the end of the presentation as well.
- You can come prepared with questions. And, 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.
    - Feel free to unmute and ask questions in the middle of the presentation.
    - Turn on your video if you feel comfortable.
    - Disclaimer: Speaker doesn't knows everything!

Check out:

- Technical Tracks and Digital Events
- Get updates – join the Digital mailing list
- Give us your feedback – take the Survey

WOMEN WHO CODE

WWCode Digital + **Backend**
Backend Study Group

August 26, 2021

WOMEN WHO CODE

Introduction & Agenda

- Welcome from WWCode!
- Our mission: Inspiring women to excel in technology careers.
- Our vision: A world where women are representative as technical executives, founders, VCs, board members and software engineers.
- What is Backend Engineering?
- Mini-series on Git process (Part 1 of 2):
  - What is Version Control?
  - What is Git?
  - Introduction to Git commands.
  - Demo.
  - Git flow workflow.
  - Q/A.

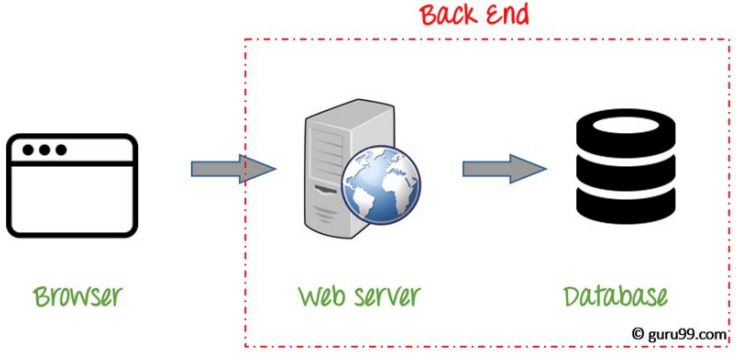
Prachi Shah
Senior Software Engineer @ Metromile

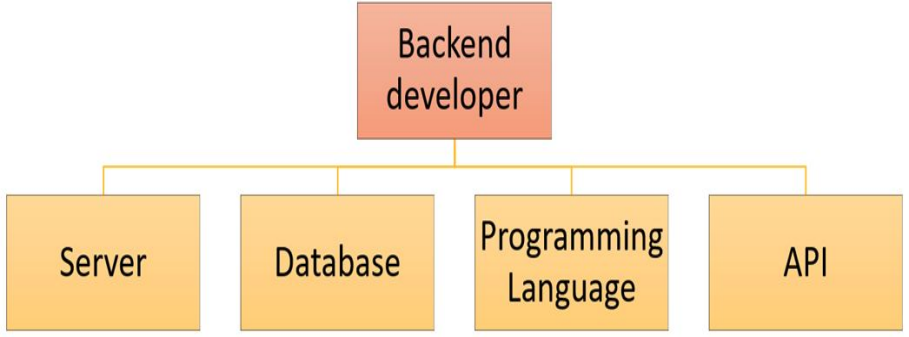
Mahalakshmi (Maha) Rajkumar
Lead @ WWCode SF

WOMEN WHO CODE

Backend Engineering

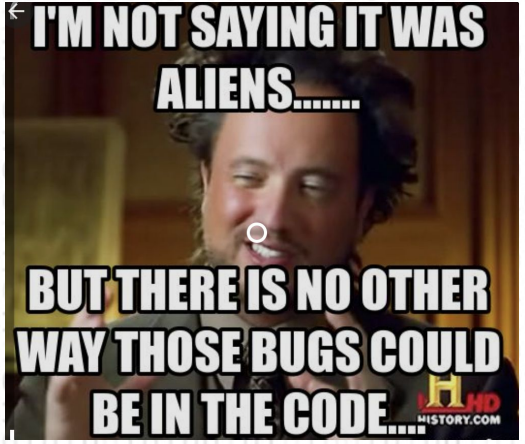
- What is Backend Engineering?
- Design, build and maintain server-side web applications.
- Concepts: Client-server architecture, API, micro-service, database engineering, distributed systems, storage, performance, deployment, availability, monitoring, scripting, software design, business rules, etc.





WOMEN WHO CODE

Backend Engineering



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.

WOMEN WHO CODE

Backend Engineering

What is source code?

What is version control?

Why do we need source code management?

Can you give examples of version control system (VCS)?

You can unmute and talk or use the chat.

WOMEN WHO CODE

Backend Engineering

What is source code?
Collection of programming statements in high-level language.

What is version control?
Process of managing and tracking code changes across projects.

Why do we need source code management?
Log a history of all code changes, option to revert to previous stable version, track changes and bugs, collaborate and get feedback from development teams.

Can you give examples of version control system (VCS)?

You can unmute and talk or use the chat.

WOMEN WHO CODE

Backend Engineering

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.

WOMEN WHO CODE

Backend Engineering

What is Git?

What is GitHub?

What is the difference between Git vs GitHub?

You can unmute and talk or use the chat.

WOMEN WHO CODE

Backend Engineering

What is Git?
VCS for fast and efficient development, code tracking, testing and shipping of code to production.

What is GitHub?
CI/CD platform for hosting and managing projects that leverage Git.

What is the difference between Git vs GitHub?
Git is a tool and a practice for CI/CD, and GitHub is a platform to support Git projects' development.

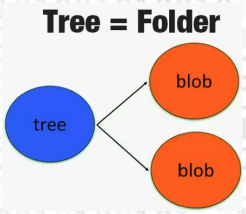
You can unmute and talk or use the chat.

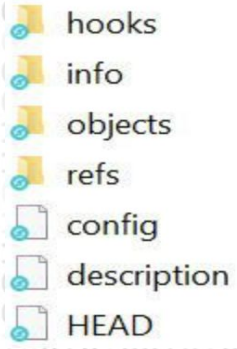
WOMEN WHO CODE

Backend Engineering

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
- 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.





WOMEN WHO CODE

Backend Engineering

Git 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.

WOMEN WHO CODE

Backend Engineering

Git 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.

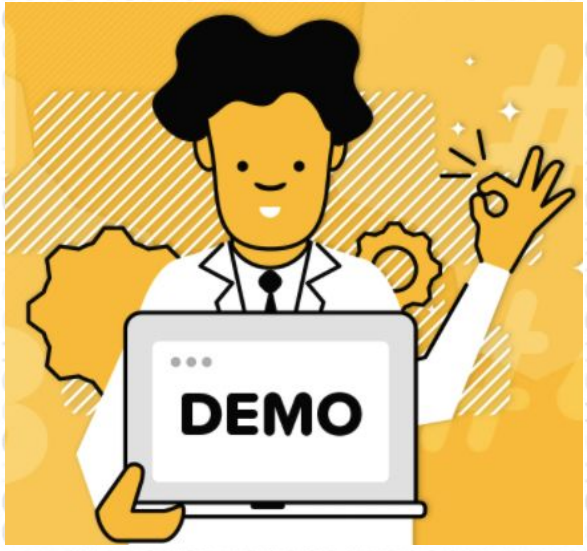
- Stage changes:
  - Make file changes as needed (add, update, rename, delete files).
  - `$ git add .` adds all changed files. Changed files are staged for a commit.
- Commit changes: `$ git commit "my first commit"` commits staged changes to the branch.
- Push: `$ git push <remote> <branch>` uploads commits to the remote branch, and sync 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.

WOMEN WHO CODE

Backend Engineering



(will be recorded)

- Git command-line in a code editor to create a pull request (PR):
  - Fetch, Pull, Checkout, Add, Commit, Push, Status.
- GitHub Desktop tool (GUI) to create a pull request (PR):
  - Fetch, Pull, Commit, Push, Create a PR.

WOMEN WHO CODE

Backend Engineering

What are mention some of the common Git Terminologies?

Can you mention the two Git repository initialization processes?

What are mention some of the common Git commands?

Do you prefer command-line or GUI approach, and why?

You can unmute and talk or use the chat.

WOMEN WHO CODE

Backend Engineering

What are mention some of the common Git Terminologies?
Repository, branch, commit, pull request, etc.

Can you mention the two Git repository initialization processes?

- Initialize a project into a Git repo.
- Clone an existing Git repo.

What are mention some of the common Git commands?
git clone, git pull, git commit, etc.

Do you prefer command-line or GUI approach, and why?

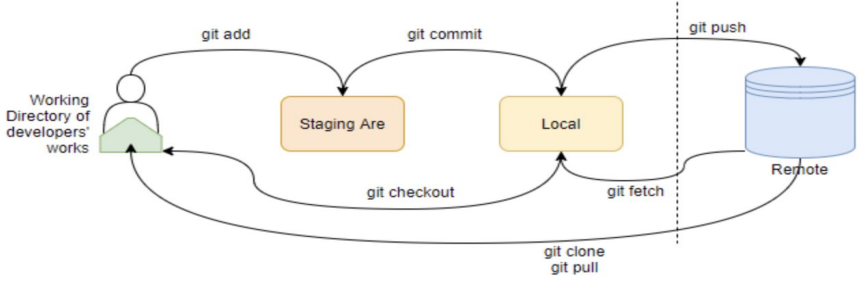
You can unmute and talk or use the chat.

WOMEN WHO CODE

Backend Engineering

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.

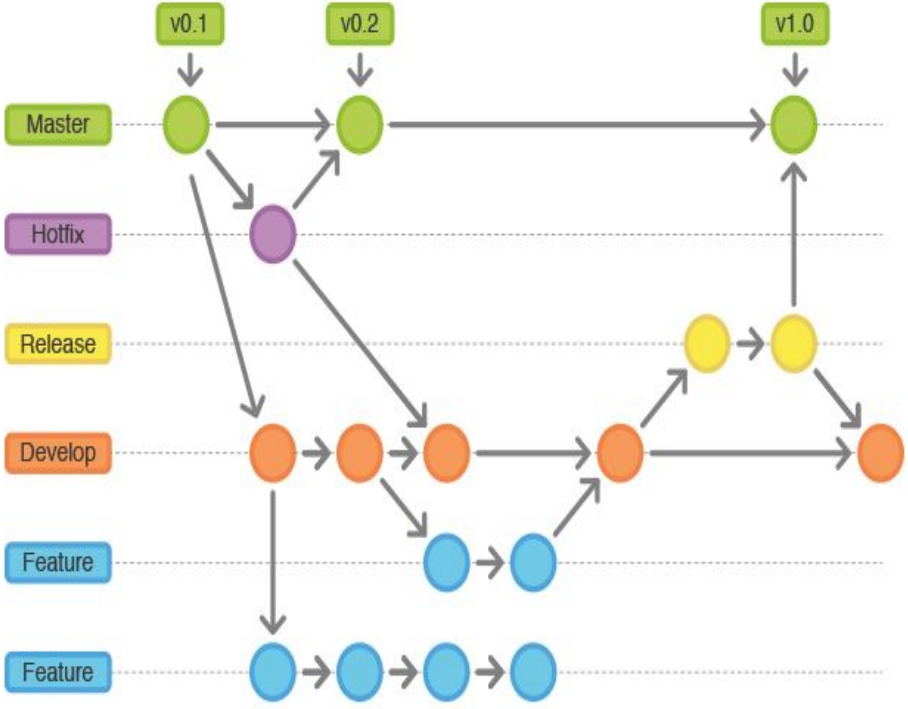


WOMEN WHO CODE

Backend Engineering

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 *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 created off of *main* branch.
- When hotfix is ready, it is merged into *main* branch, and into the *develop* branch.
- Diagram Reference: [Git Flow vs Github Flow]

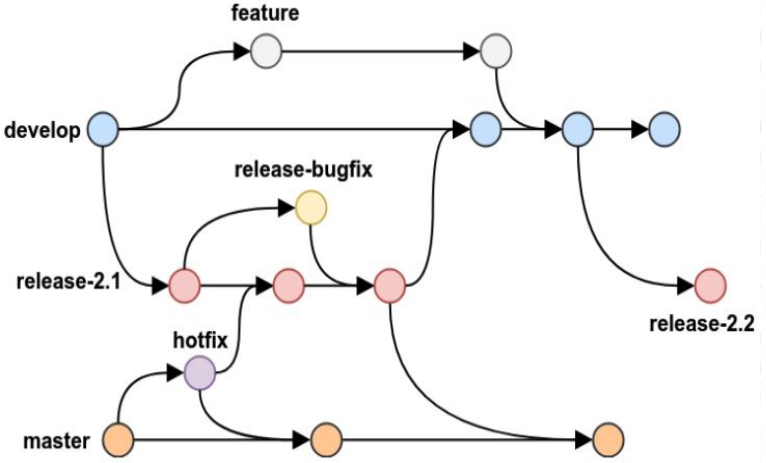


WOMEN WHO CODE

Backend Engineering

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 *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.

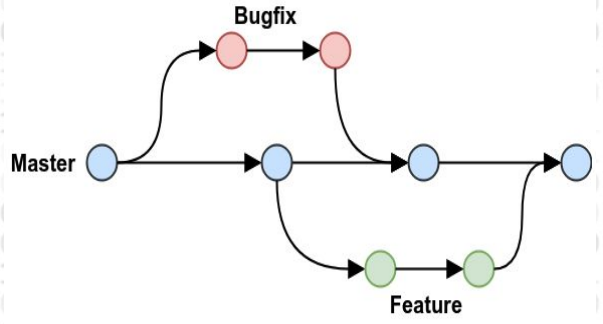


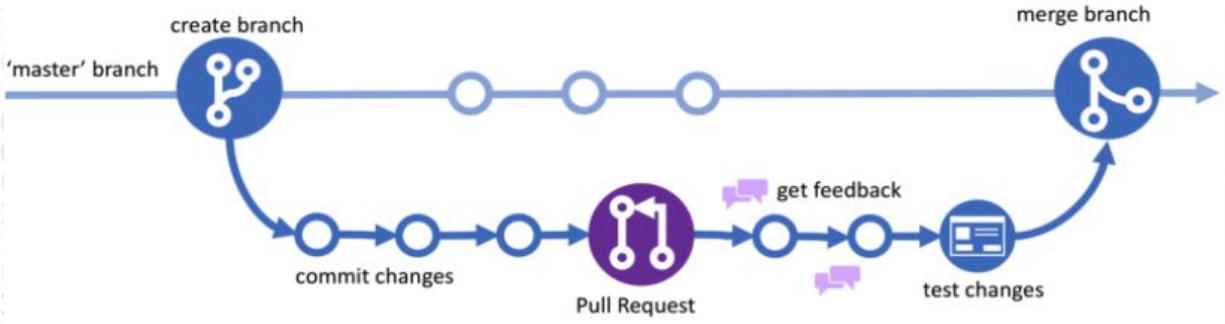
WOMEN WHO CODE

Backend Engineering

GitHub 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.





WOMEN WHO CODE

Backend Engineering

What is the Git Flow workflow?

Why does Git Flow workflow seem complicated?

What is the GitHub Flow workflow?

Why does GitHub Flow workflow seem simpler?

You can unmute and talk or use the chat.

WOMEN WHO CODE

Backend Engineering

What is the Git Flow workflow?
Best practices for integrating different branches and suitable for high development, maintenance & release projects.

Why does Git Flow workflow seem complicated?
Branching strategy is complex. Keeping a track of all branches and how they are related is tedious.

What is the GitHub Flow workflow?
Best practices for integrating different branches and suitable for faster CI/CD.

Why does GitHub Flow workflow seem simpler?
Branching strategy is simple. Keeping a track of all branches and how they are related is minimal.

You can unmute and talk or use the chat.

WOMEN WHO CODE

Backend Engineering

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

Backend Study Group:
- WWCode Presentation and session recordings found here: WWCode YouTube channel
- **NEXT SESSION** on 9-9-2021: Mini-series on Git process (Part 2 of 2):
    - Different source code management tools and comparison.
    - Code review best practices (language specific as well).
    - Live code-review session and discussion.

Resources & References:
- Backend development
- Source Code Management
- About Git
- How Git works

You can unmute and talk or use the chat.

WOMEN WHO CODE