Lesson: Git Advanced Features

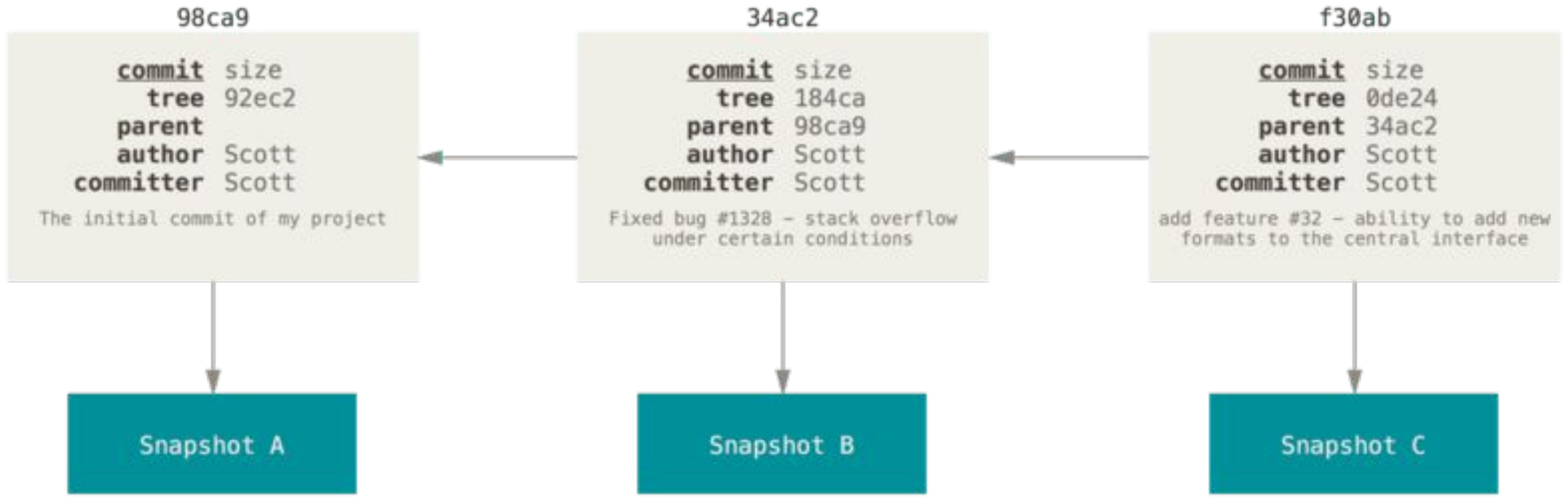
- MNNIT COMPUTER CLUB

March 8, 2018

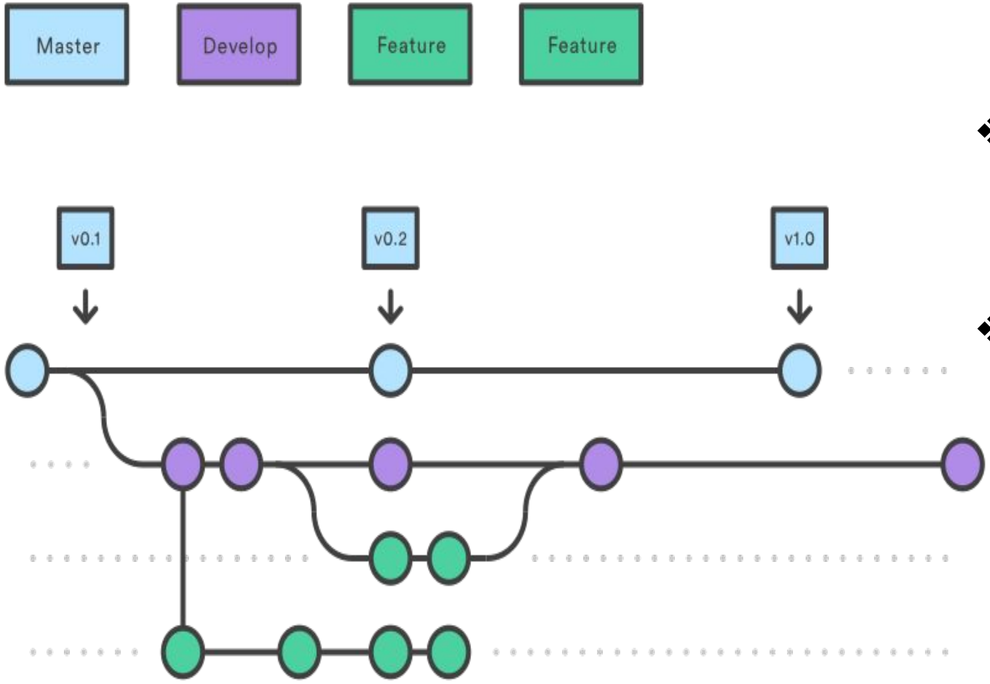
Git Branching

- When you make a commit, Git stores a commit object that contains a pointer to the snapshot of the content you staged. This object also contains the author’s name and email, the message that you typed, and **pointers to the commit or commits that directly came before this commit** (its parent or parents): zero parents for the initial commit, one parent for a normal commit, and multiple parents for a commit that results from a merge of two or more branches
- A **branch** in Git is simply a lightweight movable pointer to one of these commits. Every time you commit, it moves forward automatically.
- The default branch name in Git is master

Git Branching

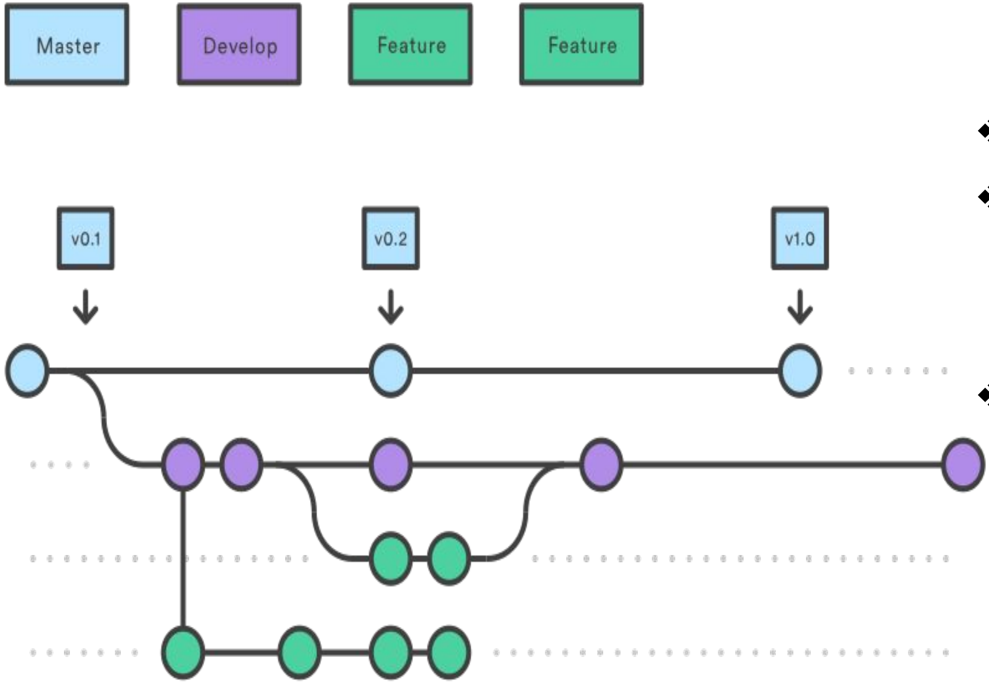


Creating a Branch



- git branch <branch_name>
- ❖ How does Git know what branch you're currently on? It keeps a special pointer called HEAD.
- ❖ Example: git branch testing

Switching a Branch



- git checkout <branch_name>
- ❖ Example: git checkout testing
- ❖ git log --oneline --decorate - to see where the branch pointers are pointing
- ❖ git checkout -b <branch_name> - to create and switch

Note that if your working directory or staging area has uncommitted changes that conflict with the branch you're checking out, Git won't let you switch branches (Stashing and Cleaning helps!!).

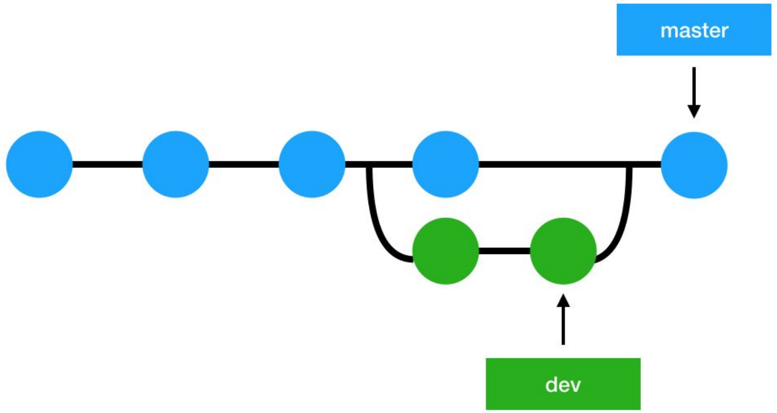
Git Merge

- git merge <new_branch_name> : merges the new_branch_name with the current working branch.

❖ When you try to merge one commit with a commit that can be reached by following the first commit's history, Git simplifies things by moving the pointer forward because there is no divergent work to merge together — this is called a "**fast-forward**."

  ➢ git branch -d <branch_name> : deletes branch <branch_name>

  ➢ git branch : lists the branches

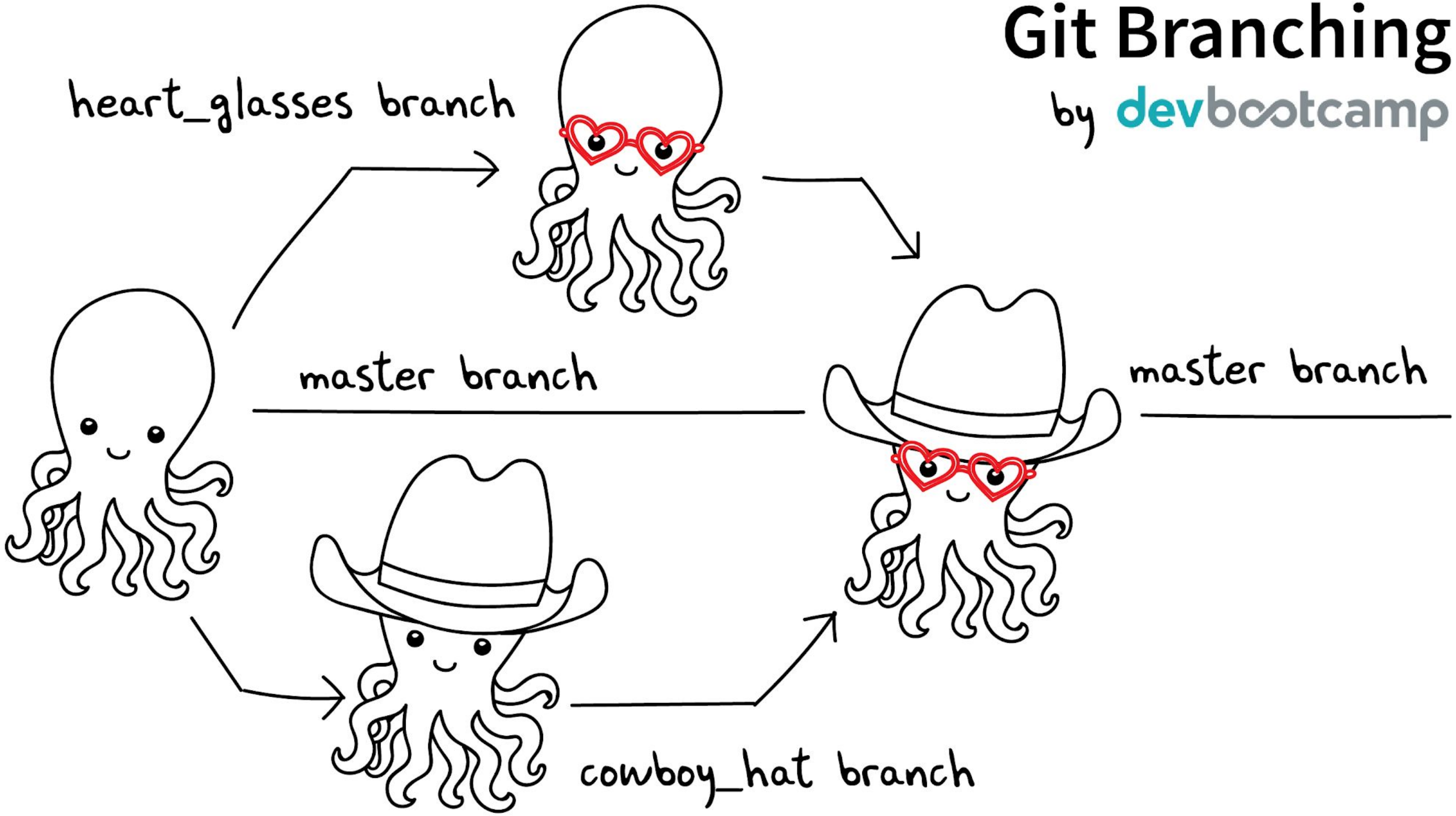


Basic Merge Conflicts

- If you change the same part of the same file differently in the two branches you’re merging together, Git won’t be able to merge them cleanly.
- Anything that has merge conflicts and hasn’t been resolved is listed as unmerged.
- After you’ve resolved each of these sections in each conflicted file, run git add on each file to mark it as resolved. Staging the file marks it as resolved in Git.
    - Read git mergetool yourself
    - git branch --merged: shows branches merged into the current branch
    - git branch --no-merged: shows branches not merged into the current branch(They can only be deleted forcefully(-D))

Git Branching

by devbootcamp



Remote Repositories

- Remote repositories are versions of your project that are hosted on the Internet or network somewhere.
- Remote repositories can be on local machine.
- In layman terms, remote is a kind of remote controller for any repository hosted anywhere else.
- git remote command is used for listing the remote servers. For cloned repositories, its origin.

Git Fetch

- The command goes out to that remote project and pulls down all the data from the remote project that you don’t have yet.
- After running this command, you have references to all the branches of that remote repo.

❖ git fetch <remote>

Git Pull

- The command goes out to that remote project and pulls down all the data from the remote project that you don’t have yet and merges the remote branch into your current branch in case there are no merge conflicts.

❖ git pull <remote> <branch>

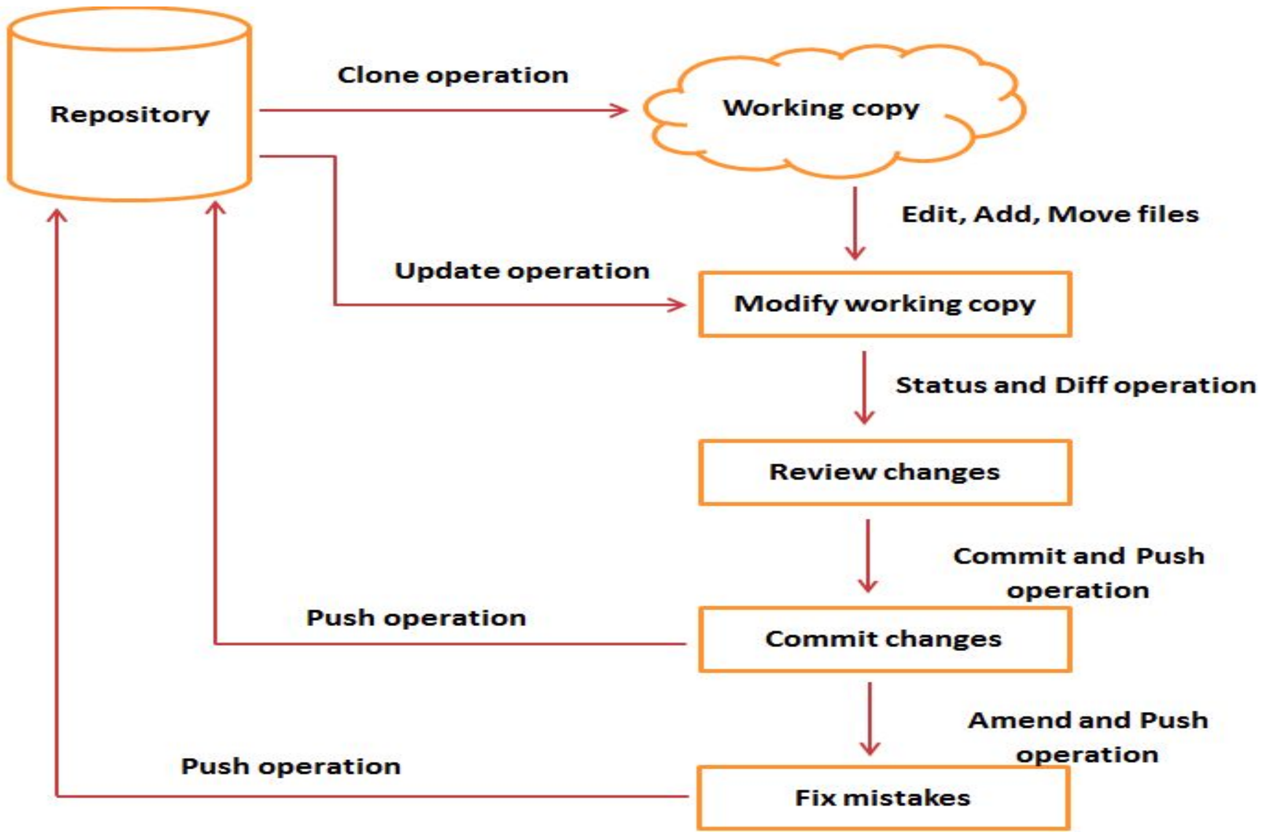
Git Push

- This command is used for adding your commits/changes to the remote repository.
- For this command to work, the working tree should remain updated with the most recent changes of the remote repo.

❖ git push <remote> <branch>

❖ **Read yourself:** git stash

Summary


Repository
Clone operation
Working copy
Edit, Add, Move files
Update operation
Modify working copy
Status and Diff operation
Review changes
Commit and Push operation
Push operation
Commit changes
Amend and Push operation
Push operation
Fix mistakes