Grand Valley State University
www.gvsu.edu

Software Engineering Configuration Management (2)

Erik Fredericks // frederer@gvsu.edu

Adapted from materials provided by Byron DeVries, Jagadeesh Nandigam

Outline

Classic view of configuration management (Part 1)

Modern configuration management (i.e., Git) (Part 2)

- Repositories
- Tagging
- Branching
- Merging
- Sharing
- Comparison
- Git Demo

Modern source control

Git:

- The most widely used source code management tool: 42.9% of professional software developers use it as their primary source control system.
- Developed by Linus Torvalds in 2005 for use developing the Linux kernel
- GitHub was developed in 2008, and provides free (and paid) Git repository hosting
    - Currently owned by none other than Microsoft!

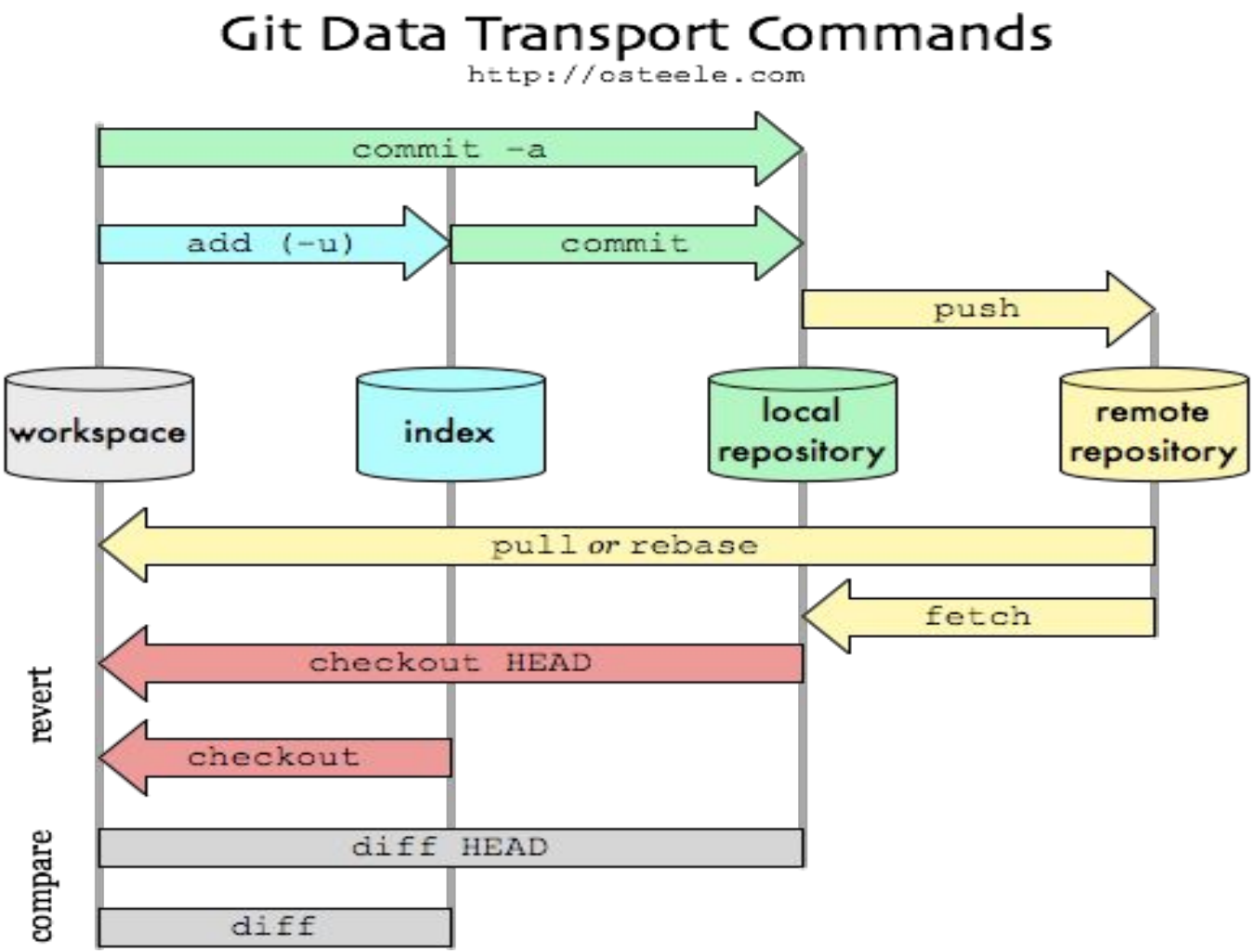


Image from http://blog.osteele.com/2008/05/my-git-workflow/

Git: Repositories

Git repositories contain:

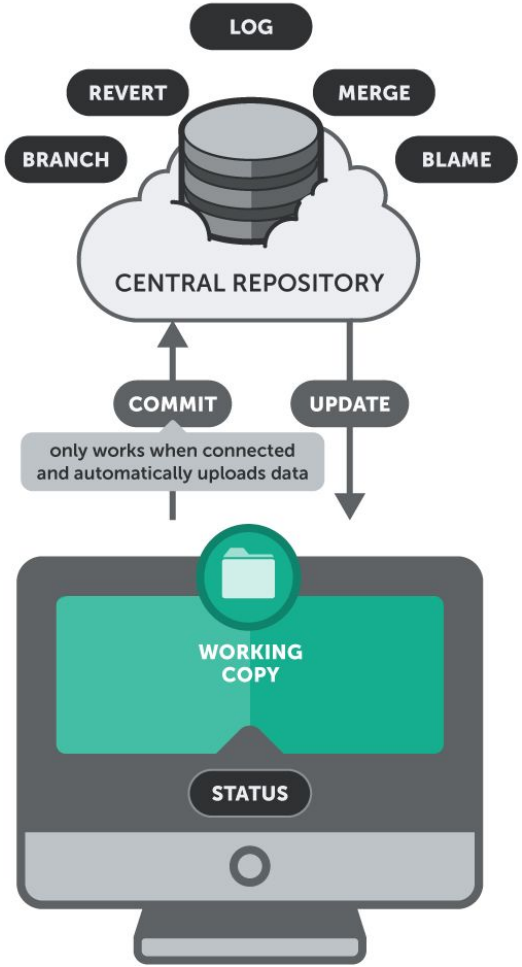
- A set of commit objects, and
- A set of references to commit objects (heads)

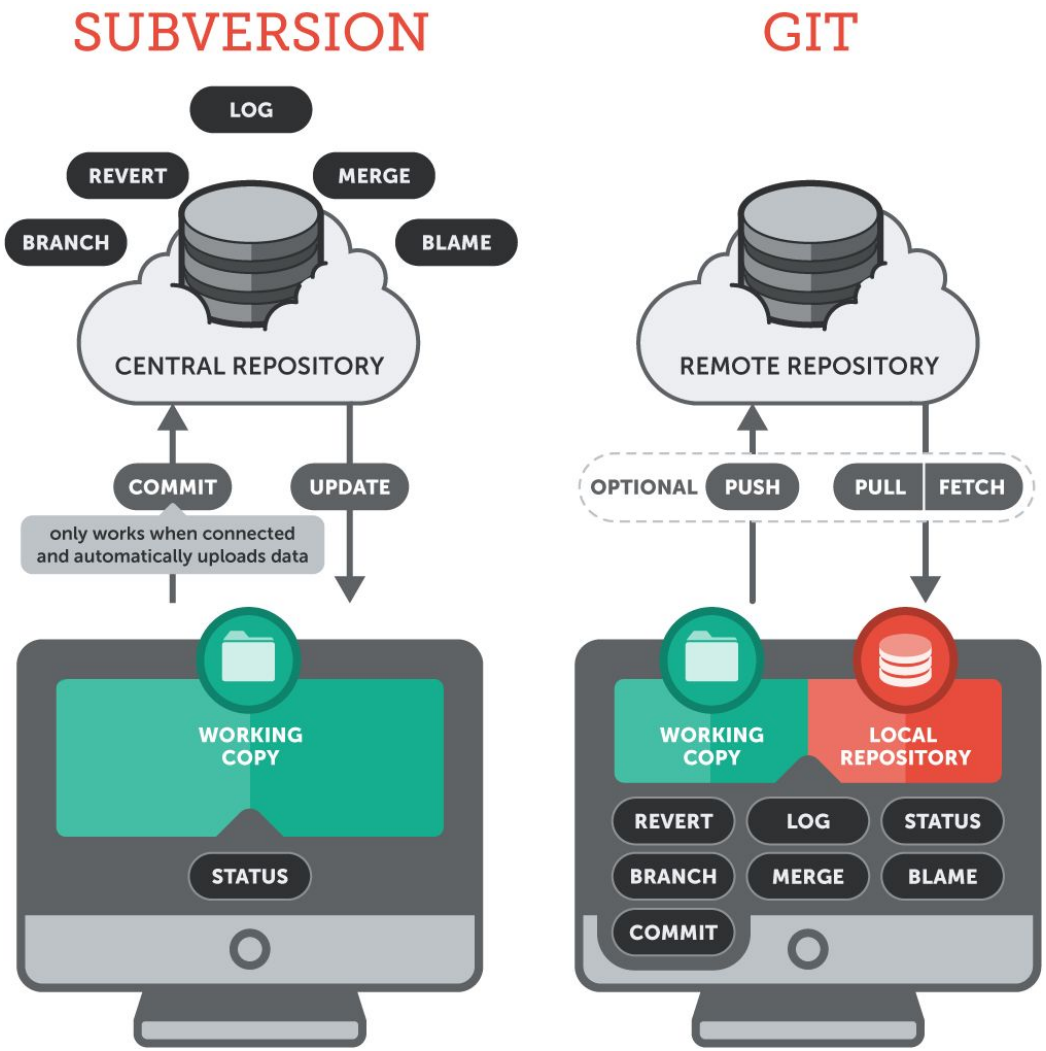
Git repositories are stored:

- In a .git sub-directory in the same directory as the project
- There is no central repository server

Git is **distributed**!

SUBVERSION


LOG
REVERT
MERGE
BRANCH
BLAME
CENTRAL REPOSITORY
COMMIT
UPDATE
only works when connected
and automatically uploads data
WORKING
COPY
STATUS


SUBVERSION
LOG
REVERT
MERGE
BRANCH
BLAME
CENTRAL REPOSITORY
COMMIT
UPDATE
only works when connected
and automatically uploads data
WORKING
COPY
STATUS
GIT
REMOTE REPOSITORY
OPTIONAL
PUSH
PULL
FETCH
WORKING
COPY
LOCAL
REPOSITORY
REVERT
LOG
STATUS
BRANCH
MERGE
BLAME
COMMIT

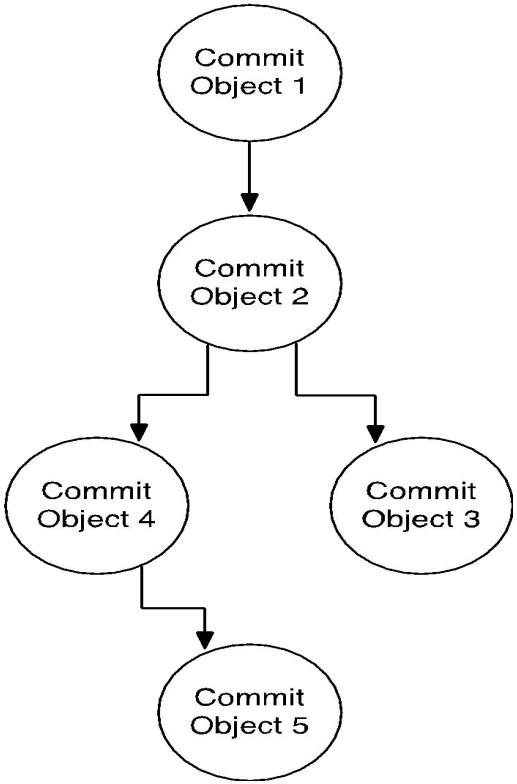
Git: Commit objects

Commit objects contain:

- A set of files (one version each),
- References to parent commit objects, and
- An SHA1 identifier that uniquely* identifies the commit

*If two commits are exactly the same, the identifier will be the same.

Git: View of commit objects in repository



Git

How do you access commit objects if you don’t know their identifier?

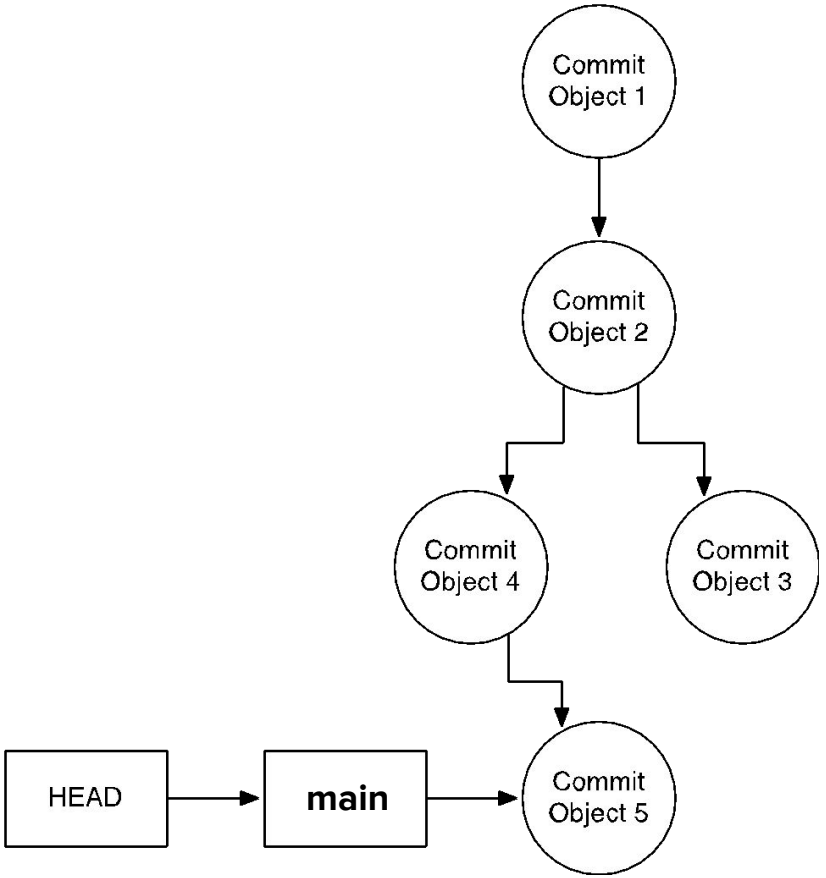
Git: Head(s)

Git heads:

Reference a commit object

Repositories always include:

- "main"
- “HEAD”



Git: Referring to a commit object

How do you reference a specific commit object?

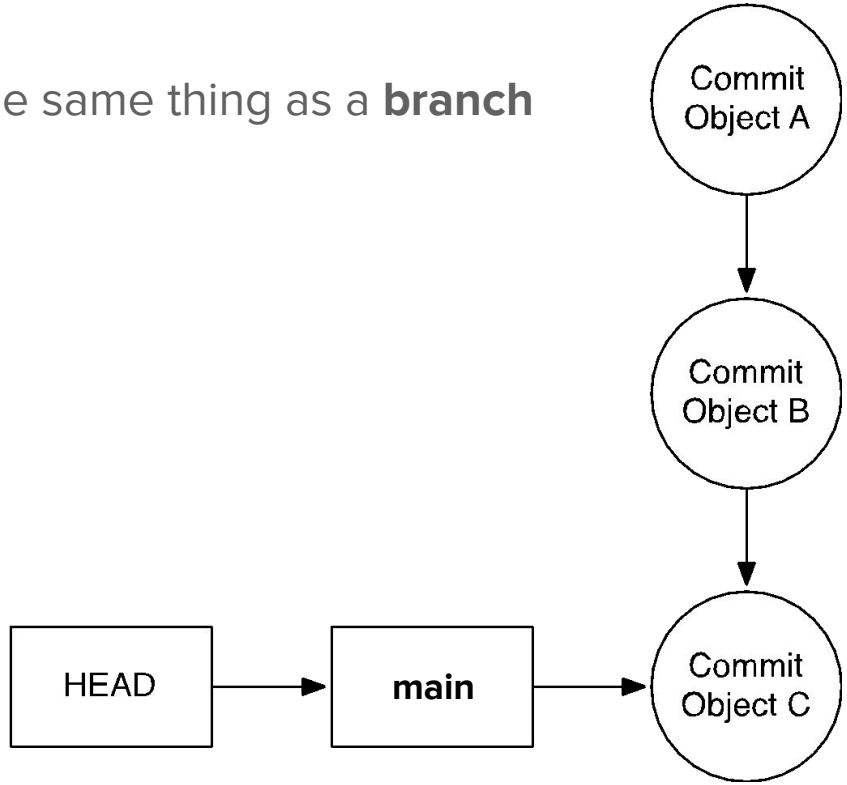
- Its identifier (i.e., the SHA1 calculated for the commit object)
- The first few characters of the identifier
- A head that references it (e.g., HEAD or main)

Git: Branching

Why would you branch?

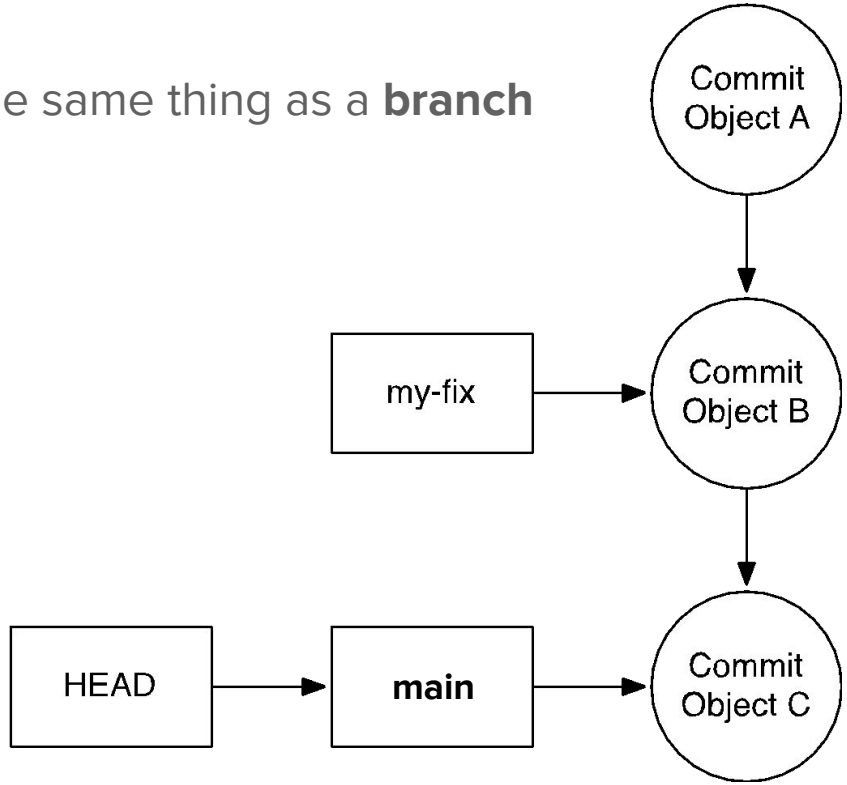
Git: Branching

In Git, a **head** is almost the same thing as a **branch**



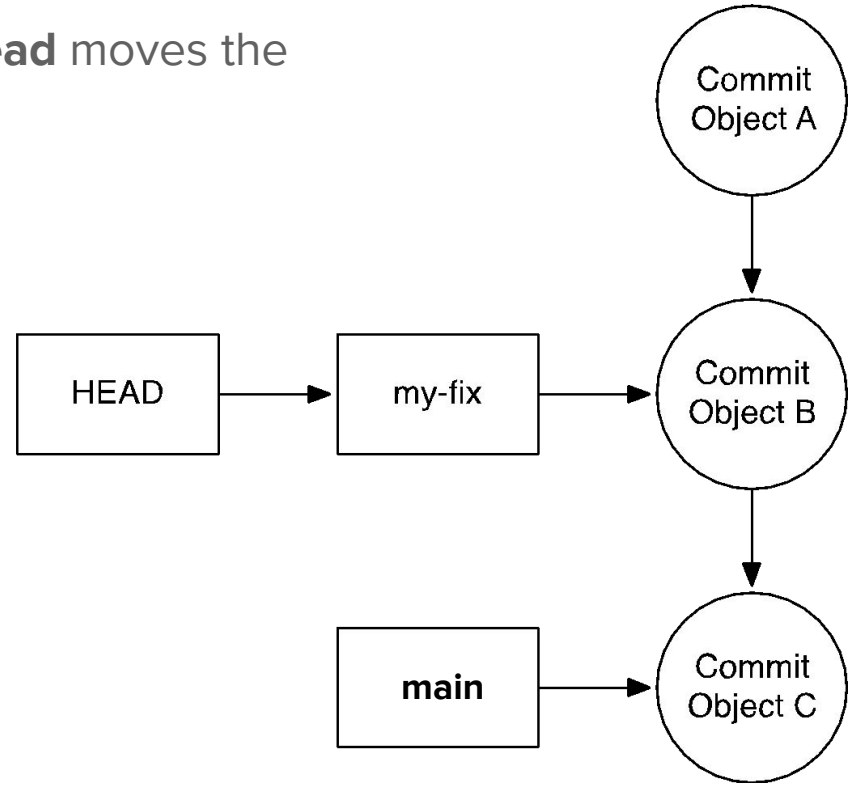
Git: Branching

In Git, a **head** is almost the same thing as a **branch**



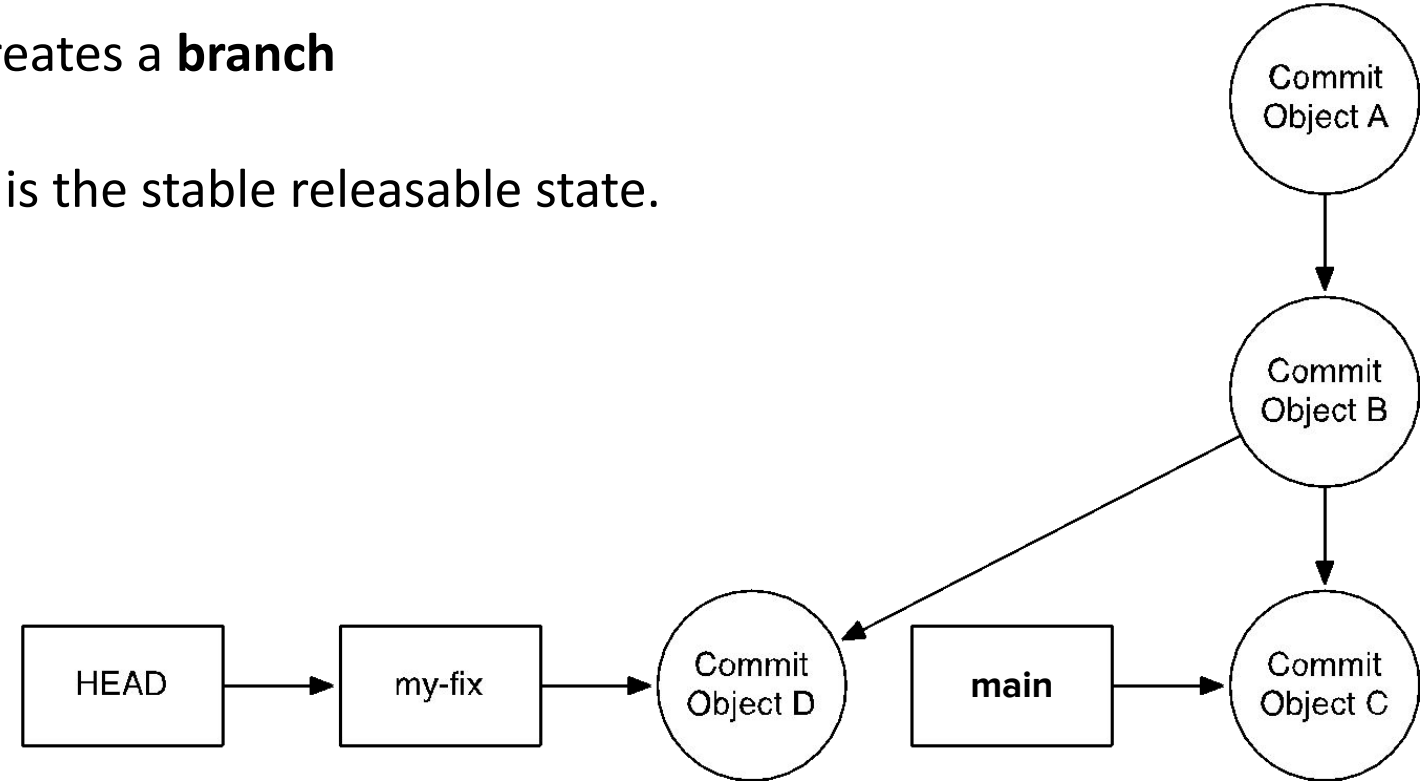
Git: Branching

Switching between **heads**, or **checkout** a **head** moves the current HEAD



A new **commit** creates a **branch**

Ideally, the **main** is the stable releasable state.



Git: Commands

`git branch` – with no arguments lists the existing heads

`git branch [new head] [location]` – creates a new head a the location

`git diff [head 1]..[head 2]` – shows the difference between the commit objects referenced by head 1 and head 2

`git log` – shows the change log

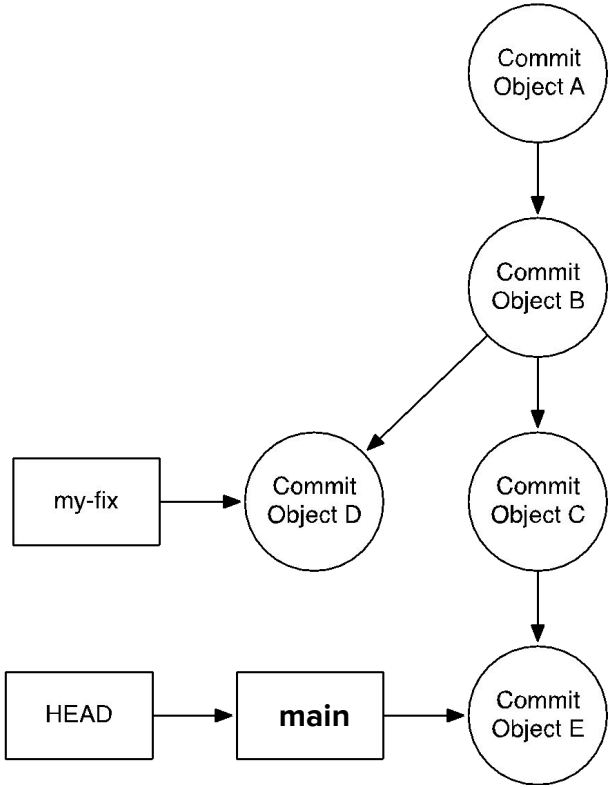
`git log[head 1]..[head 2]` – shows the change log between head 2 and the common ancestor of head 1 and head 2

`git checkout [head]` – points HEAD to the commit object referenced by head & writes all files in directory to match new reference

Git: Merge

`git merge [head]`:

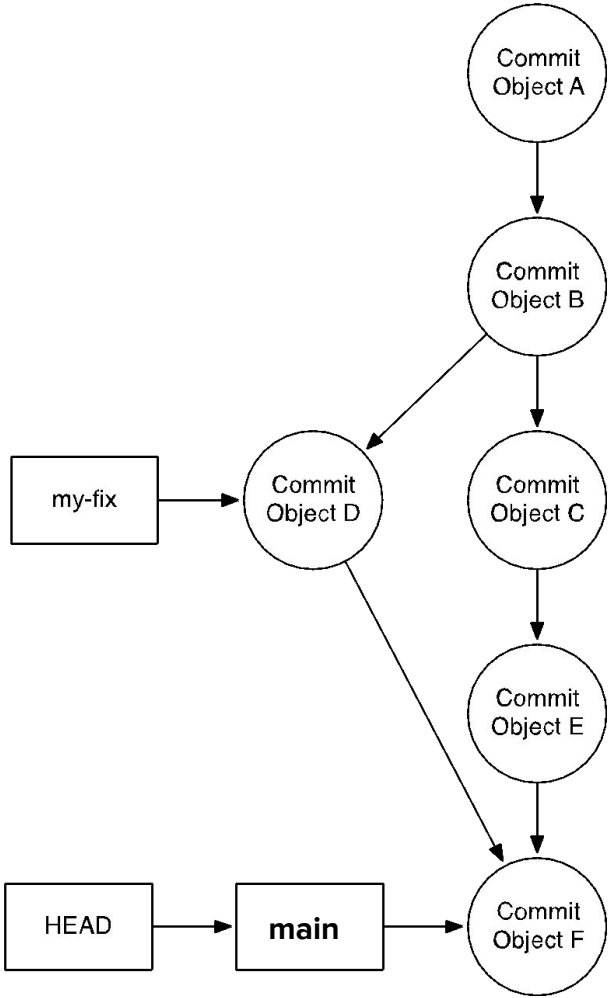
- Identify the common ancestor of HEAD and head
- If the same, do nothing.
- If ancestor is HEAD, head=HEAD
- Otherwise...



Git: Merge

`git merge [head]`:

- Identify changes between head and ancestor
- If no conflicts, create a new commit object with two parent objects (HEAD and head)
- If a conflict, inform user and don't commit



Git

But all this happens on my own computer!

- How do I share?

Git: Sharing

Make a copy of a repository:

- `git clone [remote] (e.g., ~/friend/termproject/)`

Branch from the remote repository:

- `git branch --track [new-local-branch] [remote-branch]`

Receive changes from the remote repository:

- `git fetch [remote-repository-reference]`
- `git pull [remote-repository-reference] [remote-head-name]`

Send changes to the remote repository

- `git push [remote-repository-reference] [remote-head-name]`

A refresher:

What is the difference between Git and conventional source code repositories?