Forward School

Introduction to Git and GitHub

Instructor, Nero Chan Zhen Yu

Outline

Forward School

I. Introduction to source control

A. Version Control Systems

B. Centralized vs. distributed version control

II. Introduction to Git

A. *What is Git?* Basic Git concepts and architecture

B. Git workflows: Creating a new repo (adding, committing code)

C. HEAD

D. Git commands (checking out code)

E. Master vs branch concept

F. Creating a branch/switching between branches

G. Merging branches and resolving conflicts

III. Introduction to GitHub

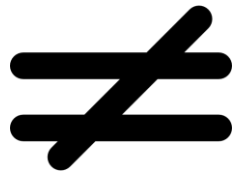
A. *What is GitHub?* Basic GitHub concepts

B. GitHub in practice: Distributed version control

C. Cloning a remote repo

D. Fetching/Pushing to a remote repo

E. Collaborating using Git and GitHub



What is Version Control System?

Forward School

- Version control, as known as Revision Control or Source Control is all about managing multiple versions of documents, programs, web sites.
    I. Almost all projects use version control system.
    II. Essential for team projects, but also very useful for individual projects.
- Every change made to the file is tracked, along with who made the change and a summary of the change, which can consist of why the change was made, what the changes are and/or any other information that are relevant to the change.
- Version Control Systems (VCS) are software implementations that simplify and sometimes automate the process of Version Control. They can be categorized into two categories; centralized and distributed.

What is Version Control System?

Forward School

- A way to manage files and directories
- Track changes over time
- Recall previous versions
- ‘Source Control’ is a subset of a VCS.

Why use a Version Control System?

Forward School

- For working by yourself:
  - Gives you a “time machine” for going back to earlier versions
  - Gives you great support for different versions (standalone, web app, etc.) of same basic project
- For working with others:
  - Greatly simplifies concurrent work, merging changes
- For getting an internship or job:
  - Any company with a clue uses some kind of version control
  - Companies without a clue are bad places to work

Why use a Version Control System?

Forward School

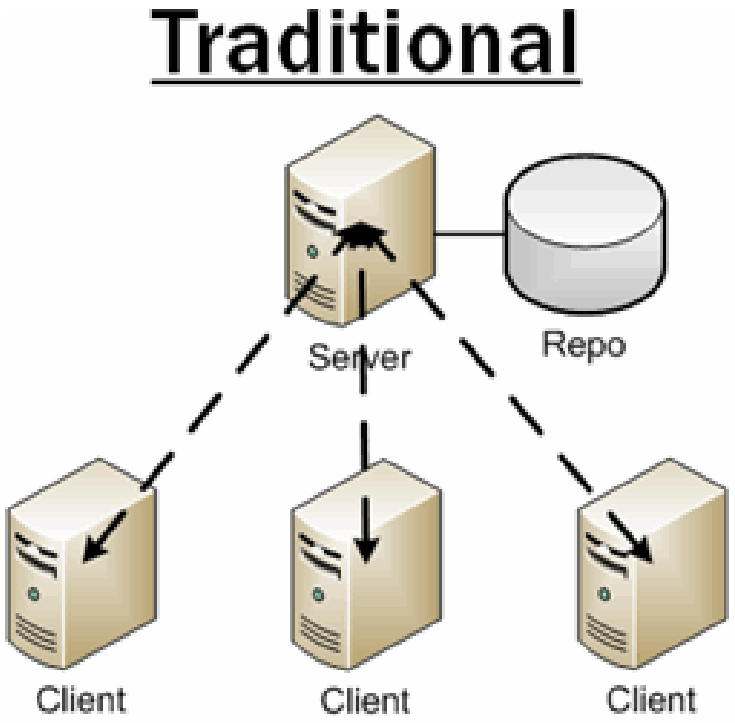
In summary, a VCS has the following features:

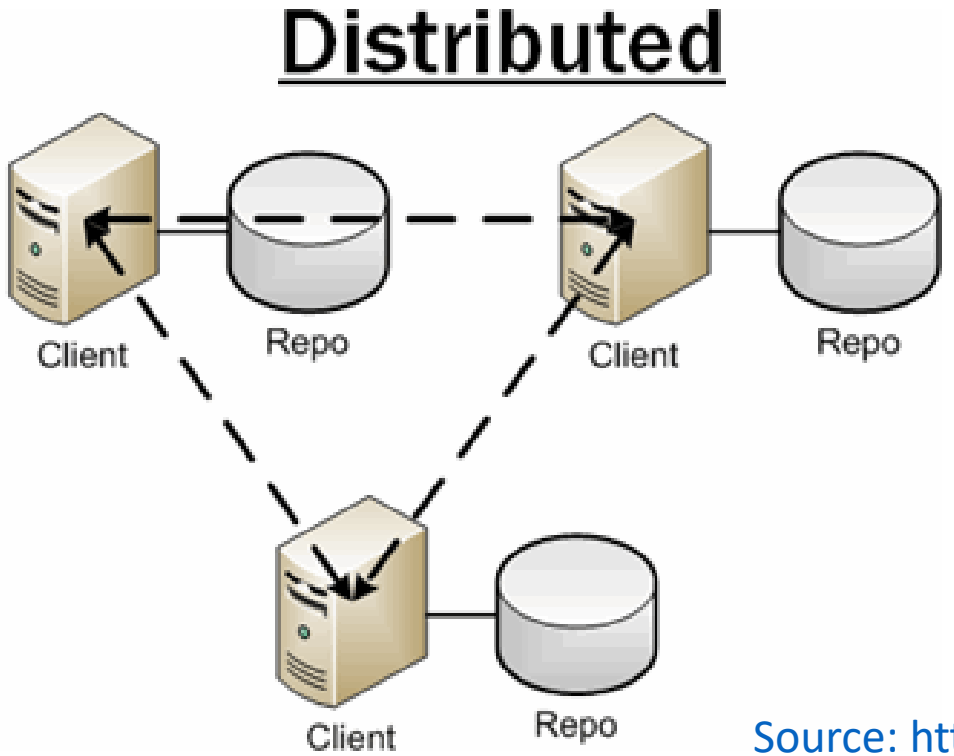
- Accountability
- Branching and Merging
- Change Tracking
- Concurrency (Multiple users can work on the same project)
- Reversibility

Centralized and Distributed Systems

Forward School

- The two main categories of version control systems are centralized (eg. subversion or svn) or decentralized, as know as distributed (eg. git).
- With a centralized system, the repository is located in one place. With as distributed system, each user has a copy of the repository.

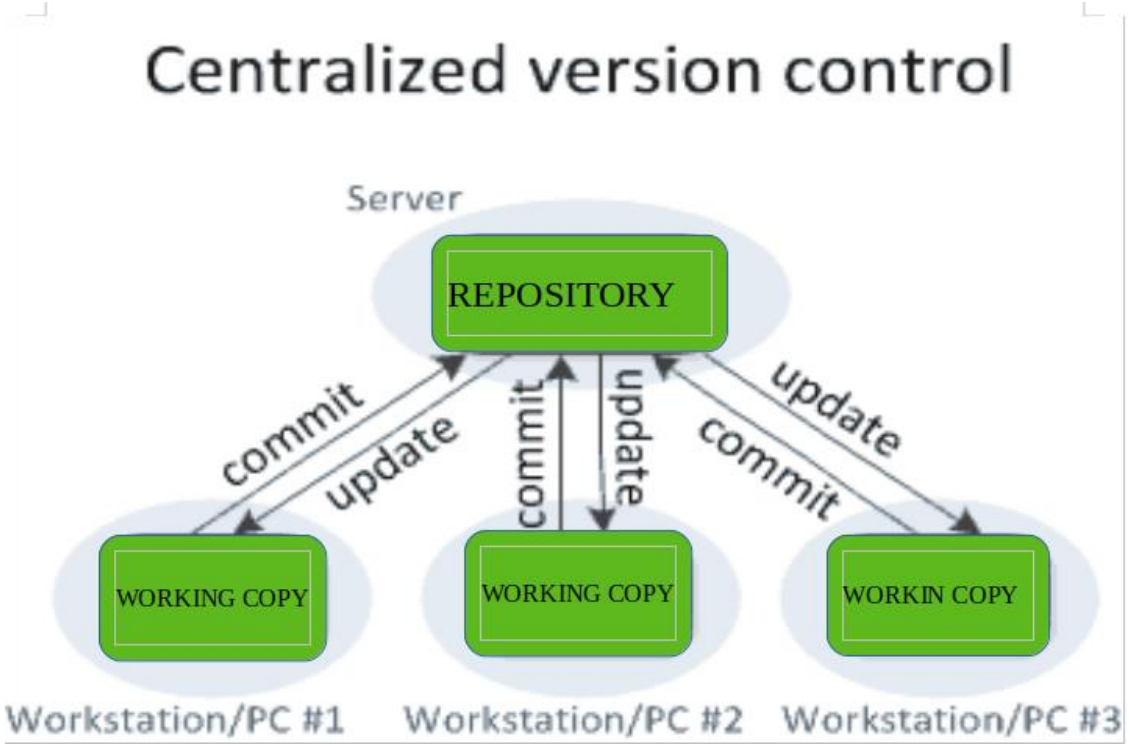


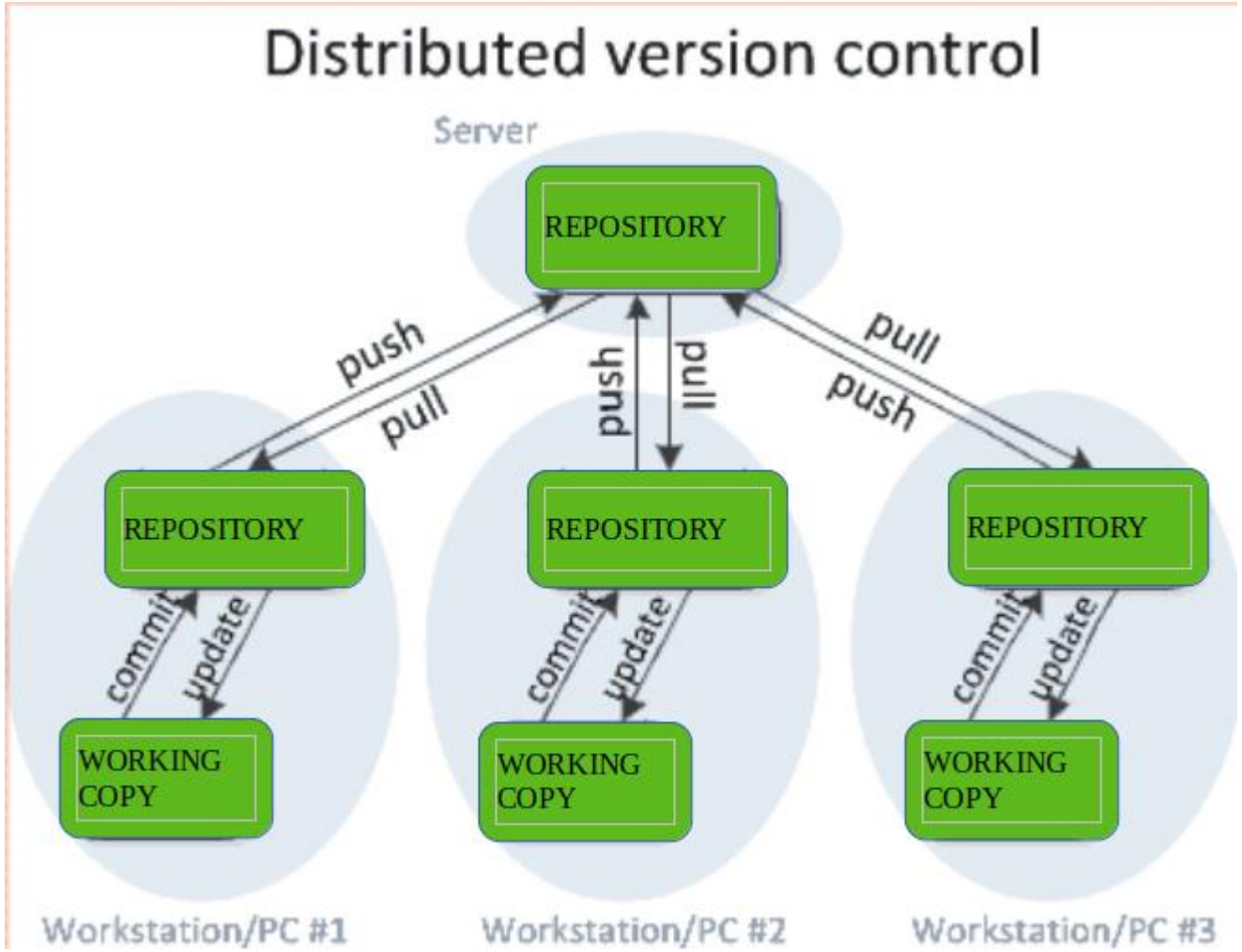


Source: http://bit.ly/1SH4E23

Centralized and Distributed Systems

Forward School





Centralized and Distributed Systems

Forward School

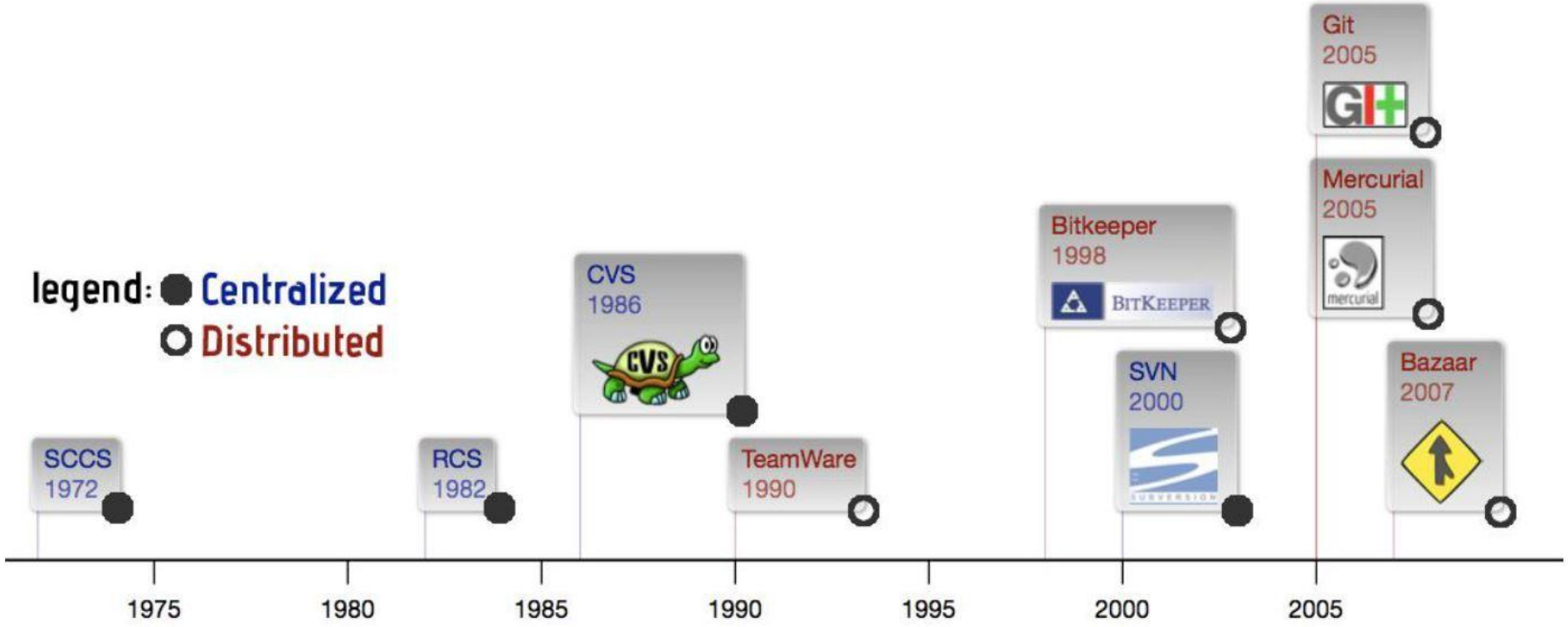


Figure 1: Timeline of the various VCS

What is Git?

- Created by Linus Torvalds, April 2005
- Replacement for BitKeeper to manage Linux kernel changes
- A command line version control program
- Uses checksums to ensure data integrity
- Distributed version control (like BitKeeper)
- Cross-platform (including Windows!)
- Open source, free

Popularity

Forward School

Distributed System (eg. Git) has many advantages over Centralized System (eg. CVS & Subversion)

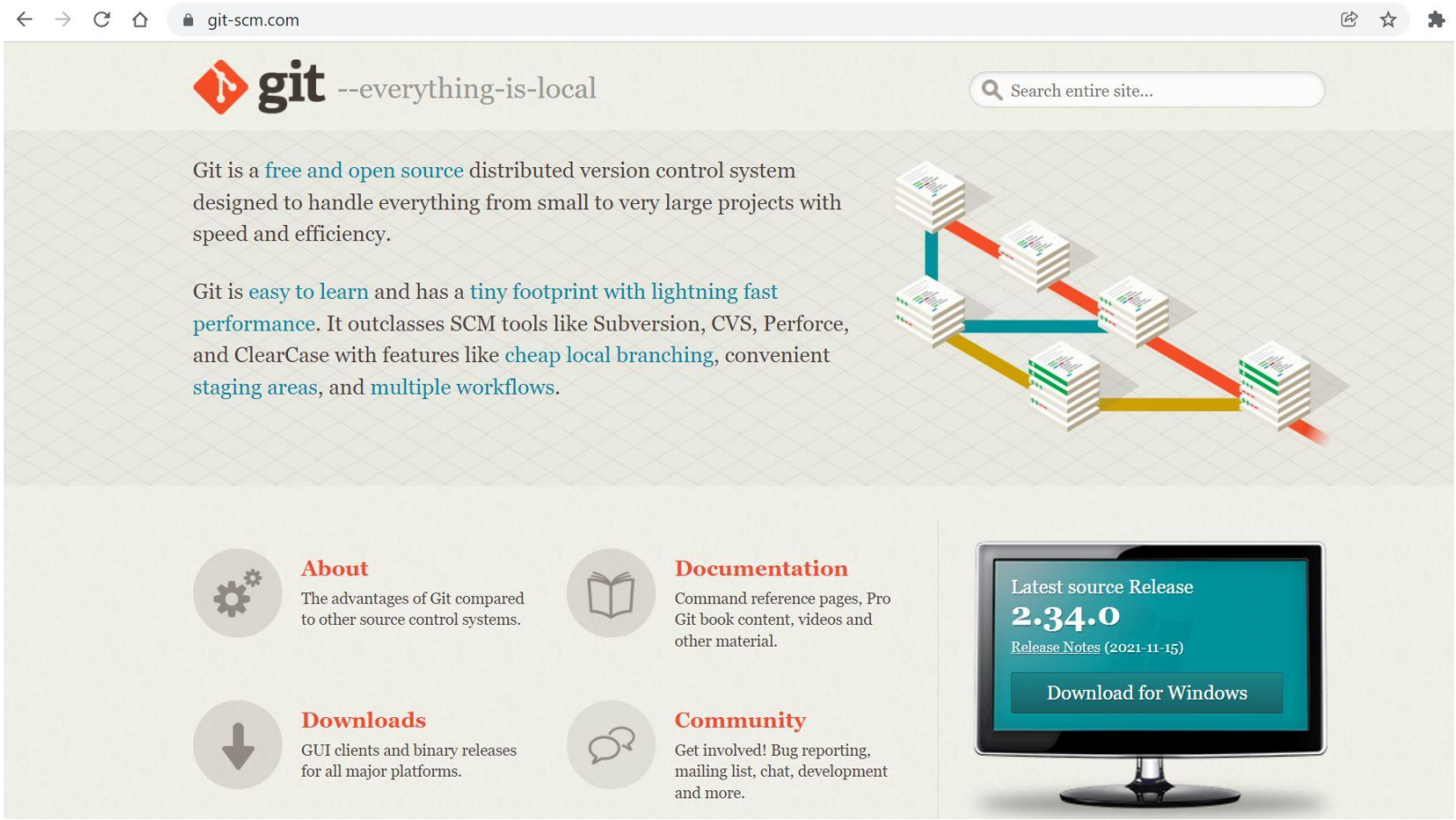
- More efficient and better workflow.
- See the literature for an extensive list of reasons
- Link 1: https://www.openhub.net/repositories/compare
- Link 2: https://qa.debian.org/popcon-graph.php

Download and Install Git

Forward School

Here is the standard download link.

- Download Link: http://git-scm.com/downloads



The Repository

Forward School

Your top-level **working directory** contains everything about your project.

- The working directory probably contains many subdirectories—source code, binaries, documentation, data files, etc.
- One of these subdirectories, named .git, is your repository

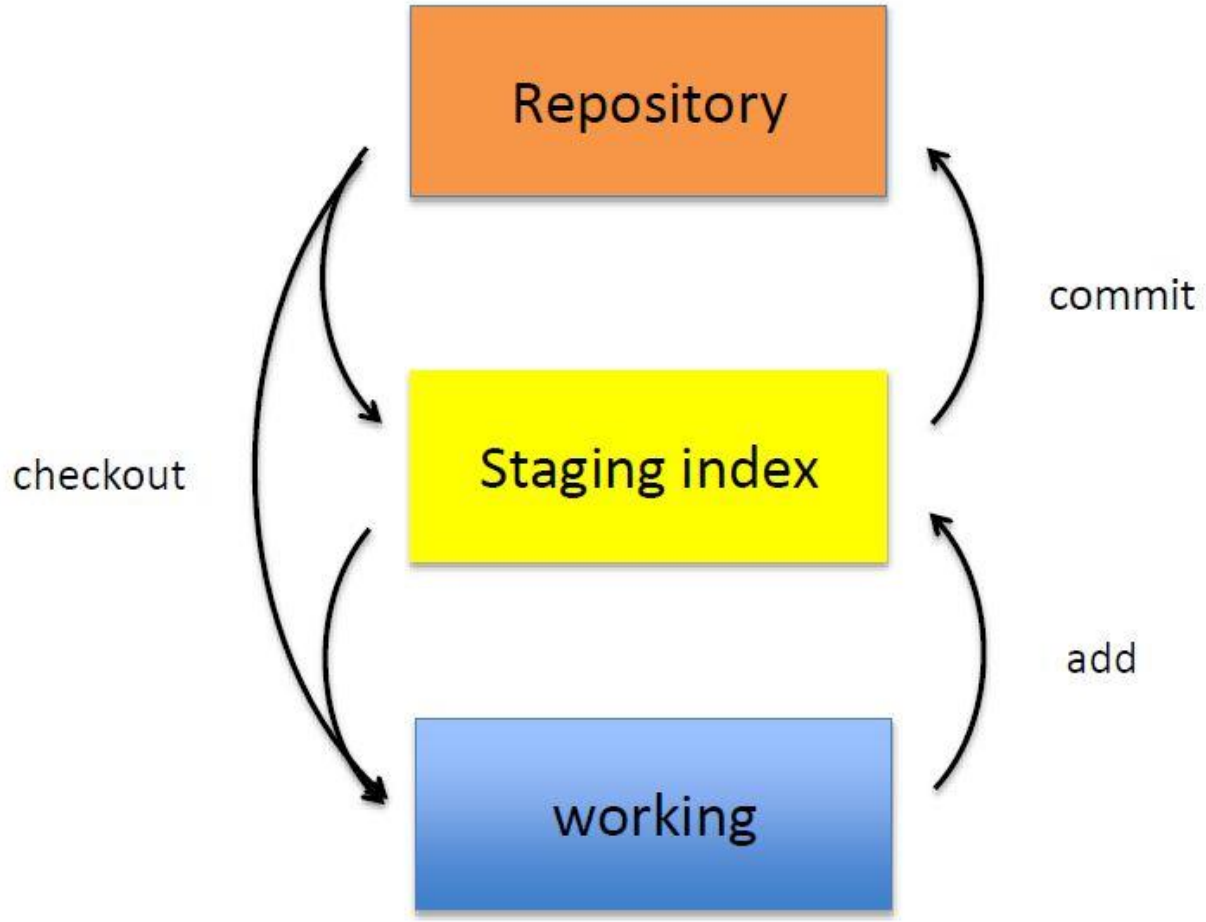
At any time, you can take a "snapshot" of everything (or selected things) in your project directory and put it in your repository.

- This "snapshot" is called a commit object
- The commit object contains (1) a set of files, (2) references to the "parents" of the commit object, and (3) a unique "SHA1" name
- Commit objects do *not* require huge amounts of storage.

You can work as much as you like in your working directory, but the repository isn't updated until you commit something.

Git 3-layers Tree Architecture

Forward School



Get Started with Git

Forward School

- Enter these lines (with appropriate changes):
  - `git config --global user.name "John Smith"`
  - `git config --global user.email jsmith@forwardschool.edu.my`
- You only need to do this once
- If you want to use a different name/email address for a particular project, you can change it for just that project
  - `cd` to the project directory
  - Use the above commands, but leave out the `--global`

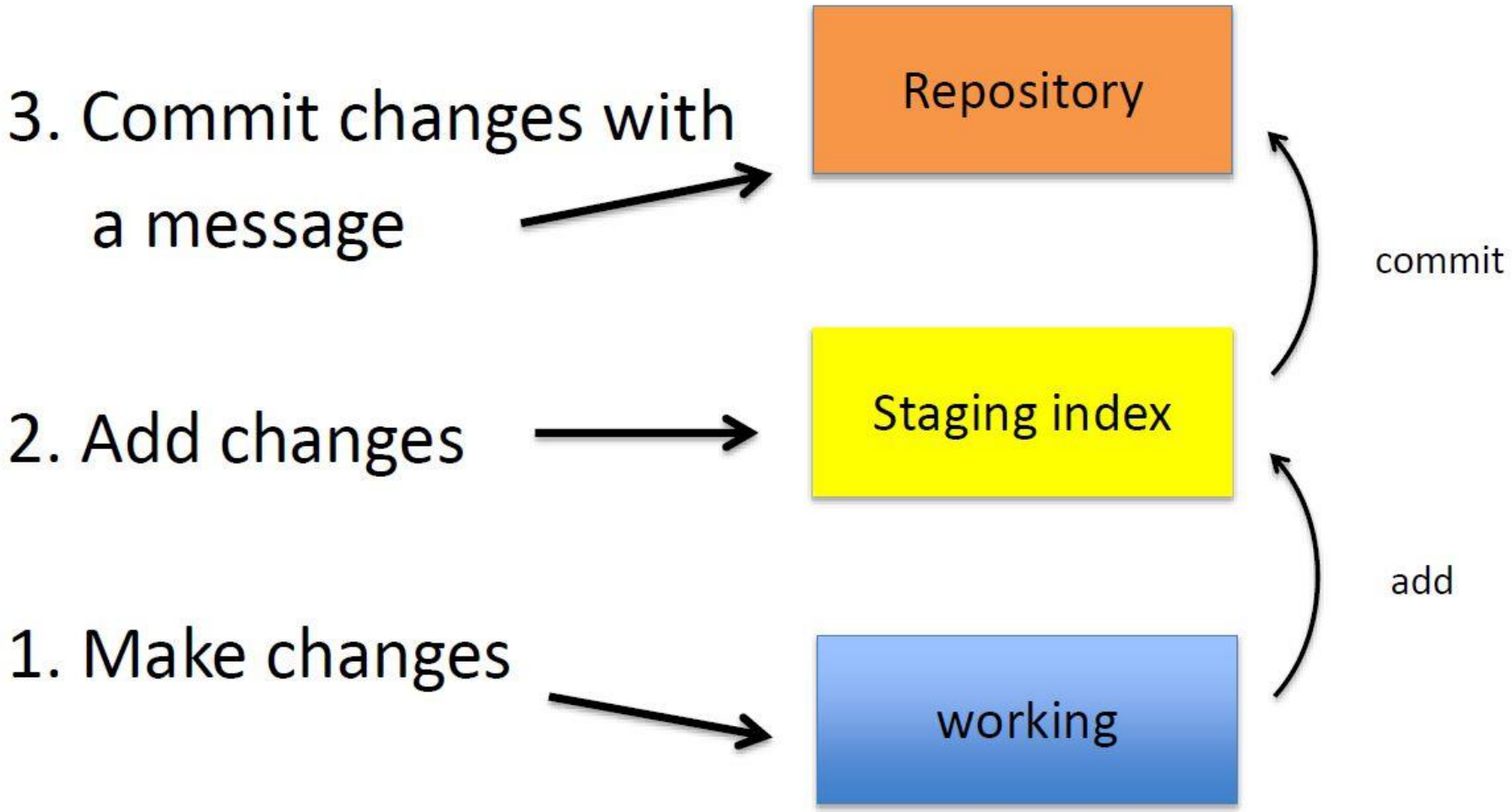
Git Workflow

Forward School

1. Change directory (cd) to the project directory you want to use.
2. Initialize a new project in a directory by using git init.
   - This creates a repository with name of .git
3. Add a file using a text editor to the directory.
4. Add the changes that has been made to the directory by using git add .
5. Commit the change to the repo by typing git commit –m “Initial Commit”
   - You can use different commit message.

Git Workflow

Forward School



Good and Bad Commit Message

Forward School

- Good: git commit –m “Add missing variable in CSS section”
- Bad: git commit –m “Typo fix”
- Bad: git commit –m “Update the table. We’ll discuss next Monday with Darrell.”

Good and Bad Commit Message

Forward School

- In git, “Commits are cheap.” Do them often.
- When you commit, you must provide a one-line message stating what you have done
  - Terrible message: “Fixed a bunch of things”
  - Better message: “Corrected the calculation of median scores”
- Commit messages can be very helpful, to yourself as well as to your team members
- You can’t say much in one line, so commit often

Choose Editor

Forward School

- When you “commit,” git will require you to type in a commit message
- For longer commit messages, you will use an editor
- The default editor is probably `vim`
- To change the default editor:
  - `git config --global core.editor /path/to/editor`
- You may also want to turn on colors:
  - `git config --global color.ui auto`

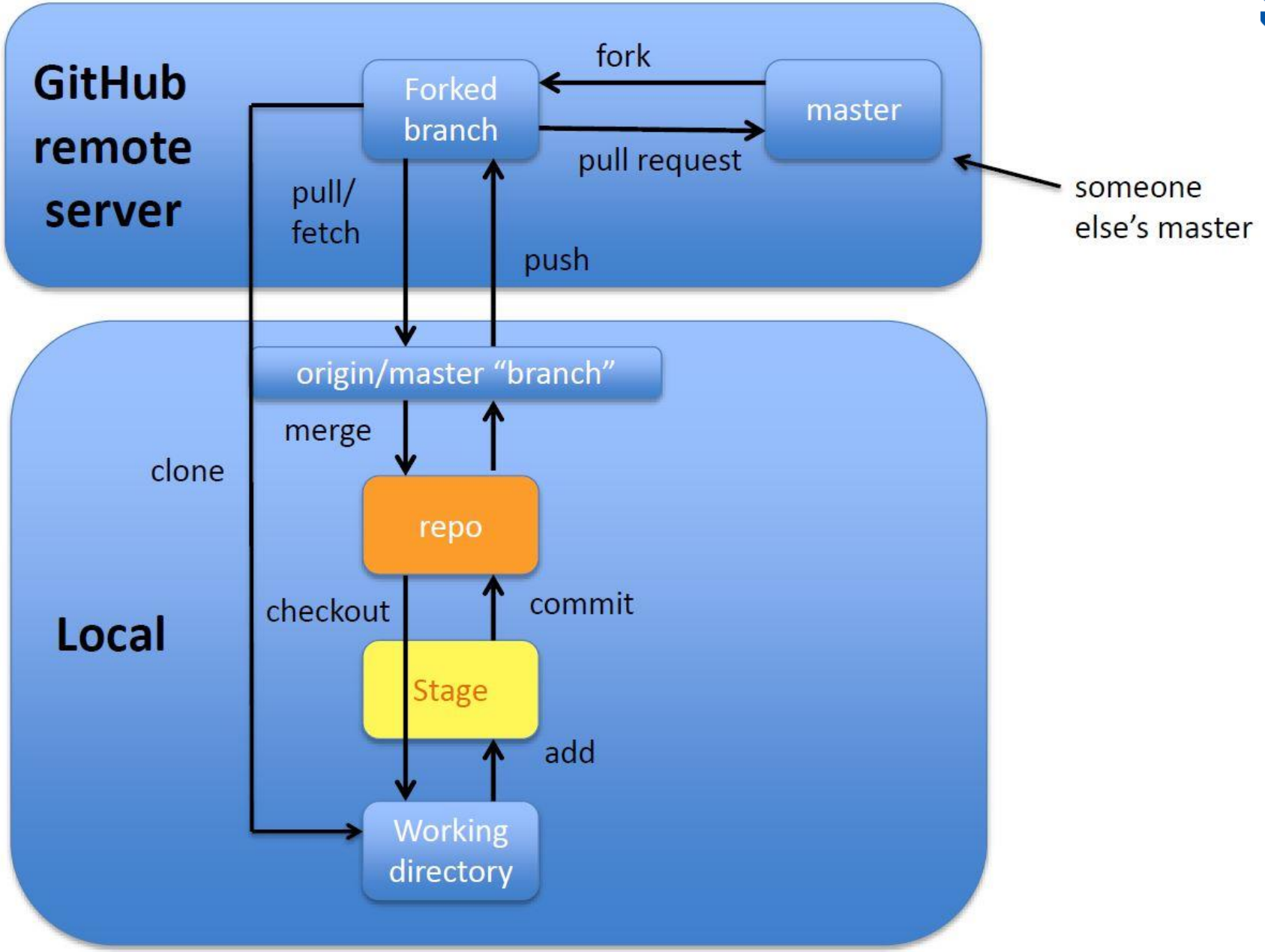
Clone Repository

Forward School

- `git clone URL`
- `git clone URL mypath`
  - These make an exact copy of the repository at the given URL
- `git clone git://github.com/rest_of_path/file.git`
  - Github is the most popular (free) public repository
- All repositories are equal
  - But you can treat some particular repository (such as one on Github) as the “master” directory
- Typically, each team member works in his/her own repository, and “merges” with other repositories as appropriate

GitHub - Cloning

Forward School



Init and the .git repository

Forward School

When you said `git init` in your project directory, or when you cloned an existing project, you created a repository

- The repository is a subdirectory named `.git` containing various files.
- The dot indicates a “hidden” directory.
- You do *not* work directly with the contents of that directory; various git commands do that for you.
- You *do* need a basic understanding of what is in the repository

Making commits

Forward School

- You do your work in your project directory, as usual
- If you create new files and/or folders, they are *not tracked* by Git unless you ask it to do so
  - git add *newFile1 newFolder1 newFolder2 newFile2*
- Committing makes a “snapshot” of everything being tracked into your repository
  - A message telling what you have done is required
  - git commit –m “Uncrevulated the conundrum bar”
  - git commit
    - This version opens an editor for you the enter the message
    - To finish, save and quit the editor
- Format of the commit message
  - One line containing the complete summary
  - If more than one line, the second line must be blank

Commits and Graphs

Forward School

- A commit is when you tell git that a change (or addition) you have made is ready to be included in the project
- When you commit your change to git, it creates a commit object
  - A commit object represents the complete state of the project, including all the files in the project
  - The *very first* commit object has no "parents"
  - Usually, you take some commit object, make some changes, and create a new commit object; the original commit object is the parent of the new commit object
    - Hence, most commit objects have a single parent
  - You can also merge two commit objects to form a new one
    - The new commit object has two parents
- Hence, commit objects form a **directed graph**
  - Git is all about using and manipulating this graph

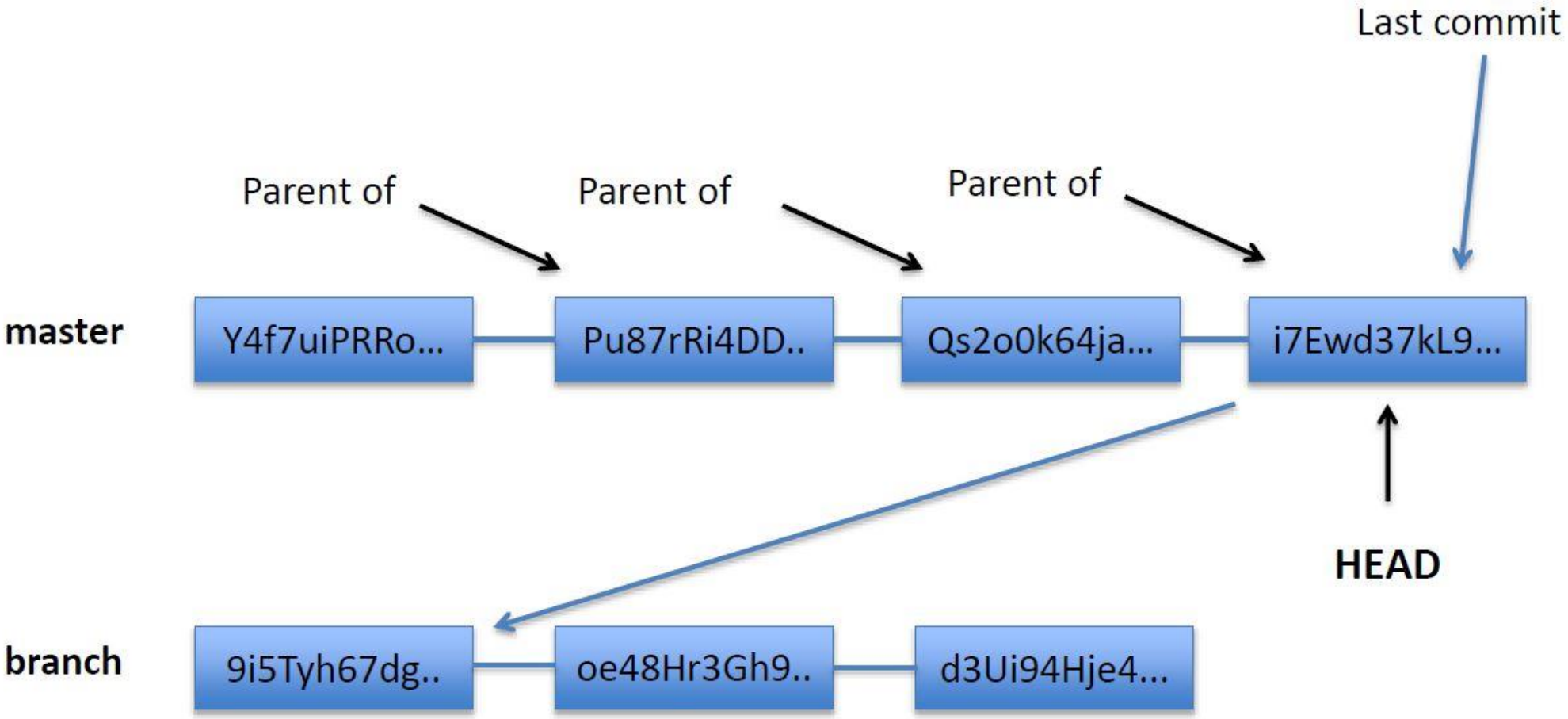
The Head Pointer

Forward School

- A head is a reference to a commit object
- The “current head” is called HEAD (all caps)
- Usually, you will take HEAD (the current commit object), make some changes to it, and commit the changes, creating a new current commit object
  - This results in a linear graph: A → B → C → …→ HEAD
- You can also take any previous commit object, make changes to it, and commit those changes
  - This creates a branch in the graph of commit objects
- You can merge any previous commit objects
  - This joins branches in the commit graph

The Head Pointer

Forward School



Working with others

Forward School

- All repositories are equal, but it is convenient to have one central repository in the cloud
- Here's what you normally do:
  - Download the current HEAD from the central repository
  - Make your changes
  - Commit your changes to your local repository
  - Check to make sure someone else on your team hasn't updated the central repository since you got it
  - Upload your changes to the central repository
- If the central repository *has* changed since you got it:
  - It is *your* responsibility to **merge your two versions**
    - This is a strong incentive to commit and upload often!
  - Git can often do this for you, if there aren't incompatible changes

Typical Workflow

Forward School

- `git pull` *`remote_repository`*
  - Get changes from a remote repository and merge them into your own repository
- `git status`
  - See what Git thinks is going on
  - Use this frequently!
- Work on your files (remember to `add` any new ones)
- `git commit -m` *“What I did”*
- `git push`

Git Commands

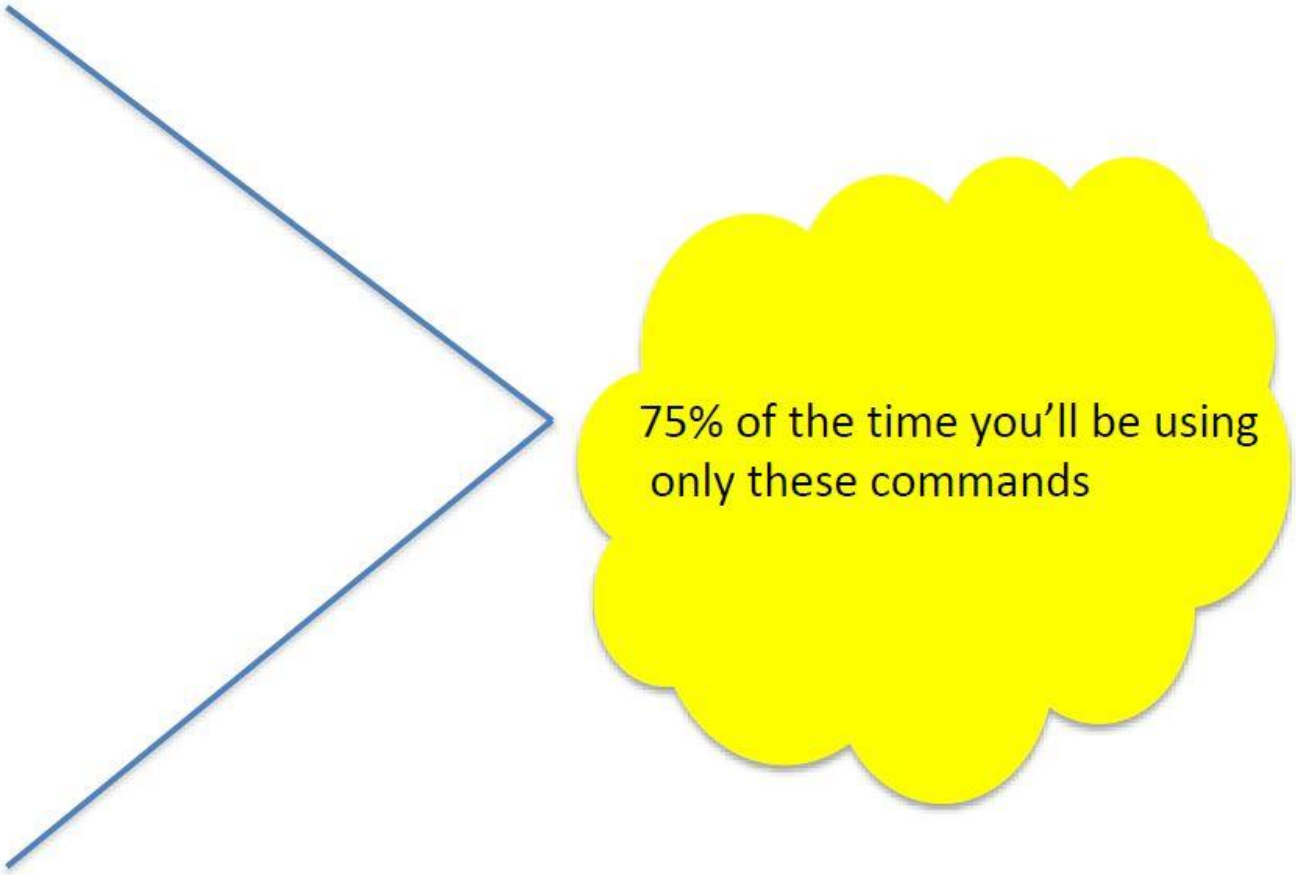
Forward School

- git status – allows one to see where files are in the 3-tree scheme.
- git diff – compares changes to files between repo and working directory
- git rm filename.txt - moves deleted file change to staging area and you need to commit the change.
- git mv filename1.txt filename2.txt – Moves or renames the files

Git Commands

Forward School

git init
git status
git log
git add
git commit
git diff
git rm
git mv

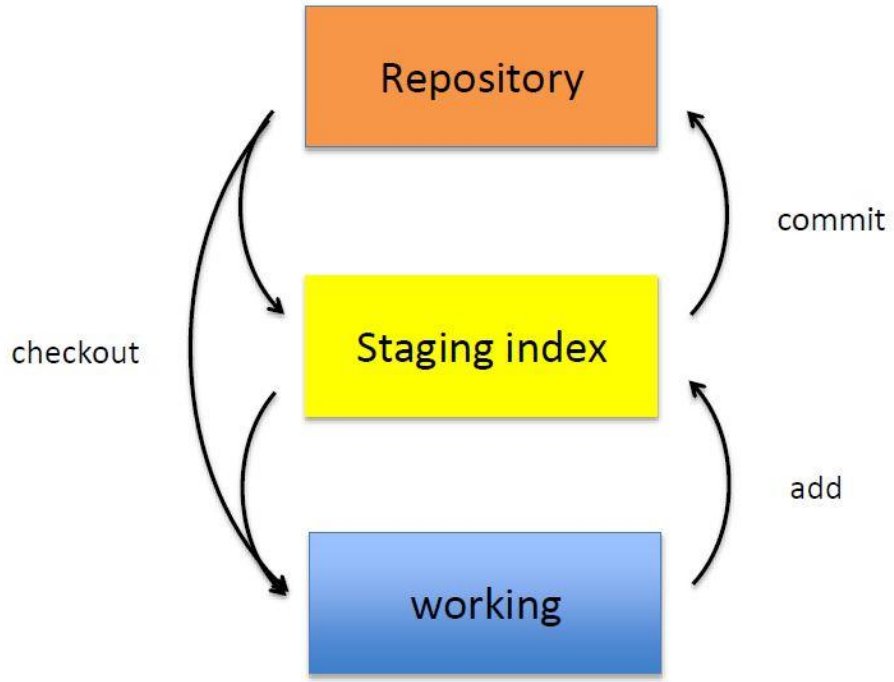


Git Commands

Forward School

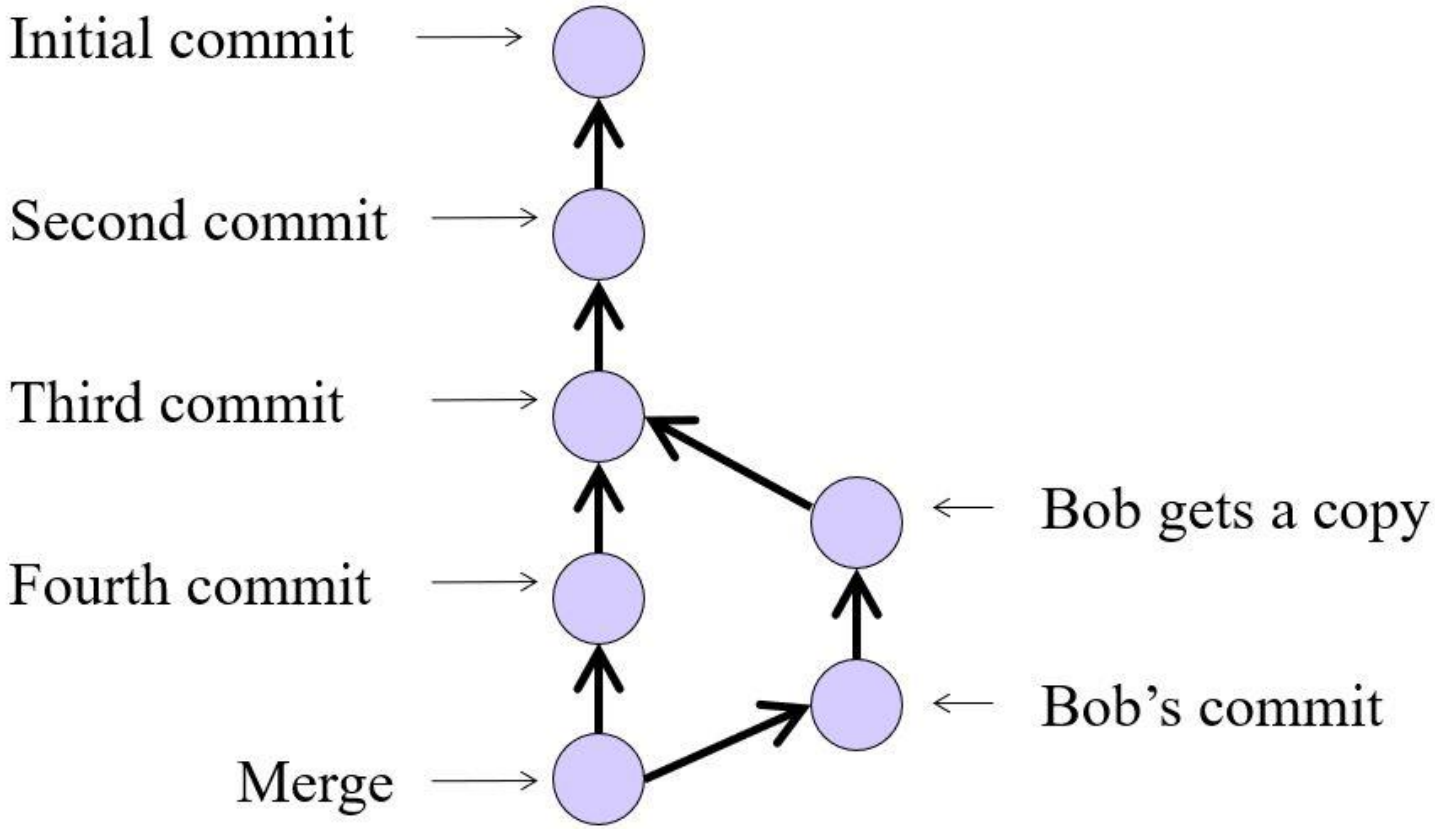
- git checkout *something* – Undo changes made to working directory. “something” is a file or an entire branch.
- git reset HEAD filename.txt – Undo changes added to staging area.
- git commit --amend -m “message” – Undo changes committed to the repo.
- git ls-tree tree-ish – Check which files are in a repo.

  tree-ish – a way to reference a repo full SHA, part SHA, HEAD, others



Forward School

Multiple Versions

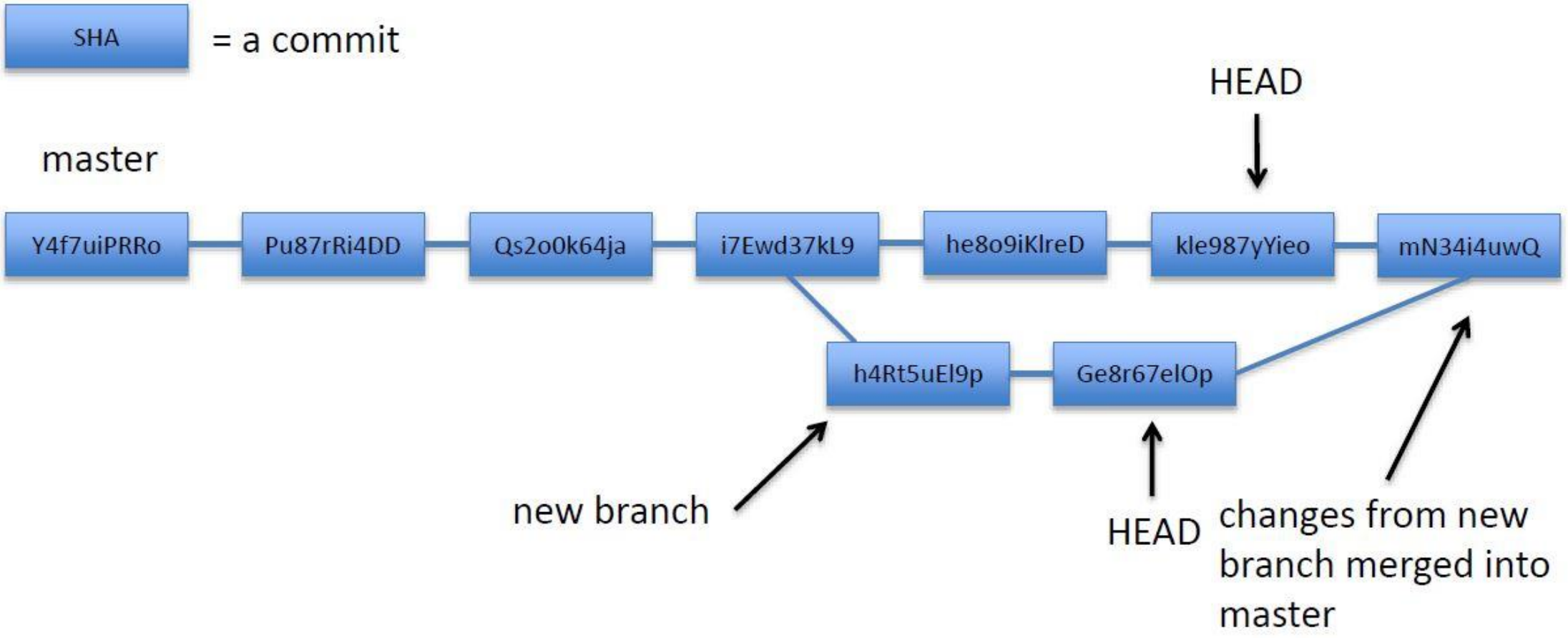


Branching

- Allows one to try new ideas
- If an idea doesn't work, throw away the branch. Don't have to undo many changes to master branch.
- If it *does* work, merge ideas into master branch.
- There is only one working directory.

Forward School

Branching and Merging Example



Source: http://hades.github.io/2010/01/git-your-friend-not-foe-vol-2-branches/

Git Commands

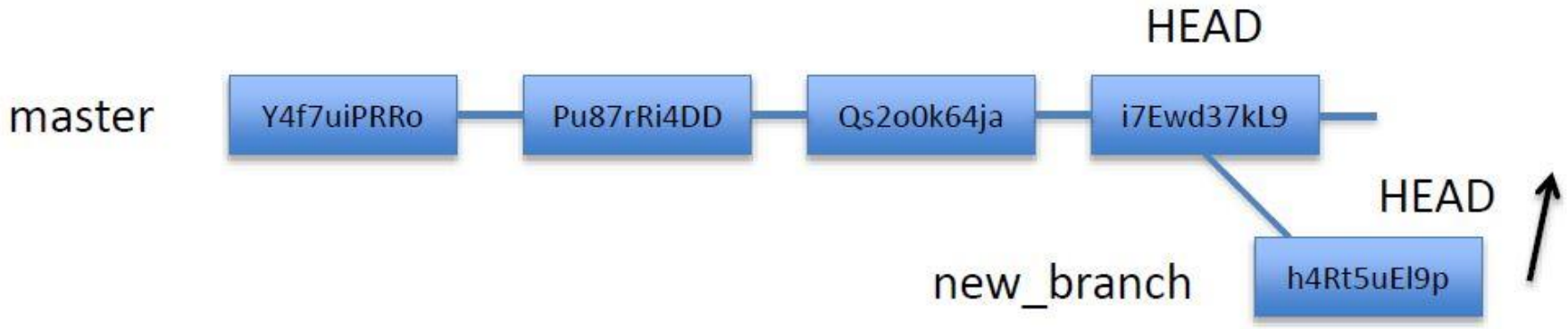
Forward School

- git branch – Check the current branch
- git branch new_branch_name – create a new branch.
- git checkout new_branch_name - switch between branches, making commits, etc. in either branch, while the two stay separate from one another.
- git diff first_branch..second_branch – Compare the different between 1st branch and 2nd branch.
- git merge branch_to_merge – Merge the current branch and target branch.
- git log --graph --oneline --all –decorate – graphing merge history.

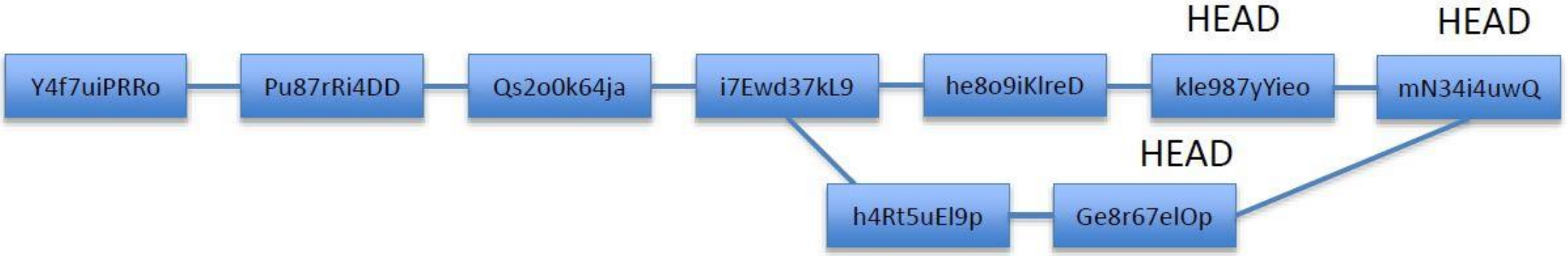
Merging

Forward School

"**fast-forward**" merge occurs when HEAD of master branch is seen when looking back.



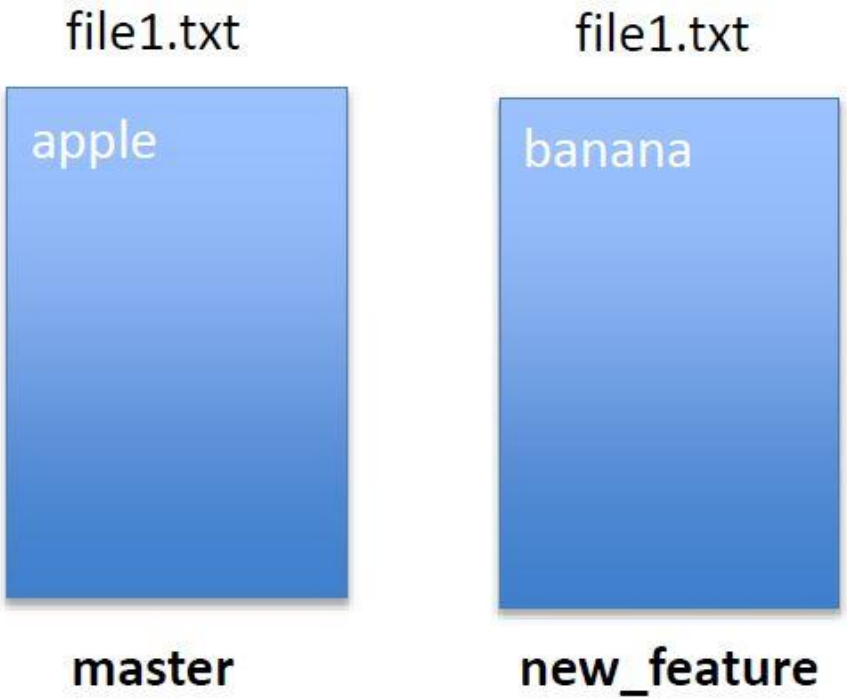
"**recursive**" merge occurs by looking back and combining ancestors to resolve merge.



Merging Conflicts

Forward School

- What if there are two changes to same line in two different commits?



Resolving Merge Conflicts

Forward School

Git will notate the conflict in the files.

```
<<<<<<< HEAD
apple
=======
banana
>>>>>>> new_feature
```

Solutions:

1. Abort the merge using git merge –abort
2. Manually fix the conflict
3. Use a merge tool (there are many out there)

Tips to reduce merging pain

Forward School

- merge often
- keep commits small/focused
- bring changes occurring to master into your branch frequently (“tracking”)

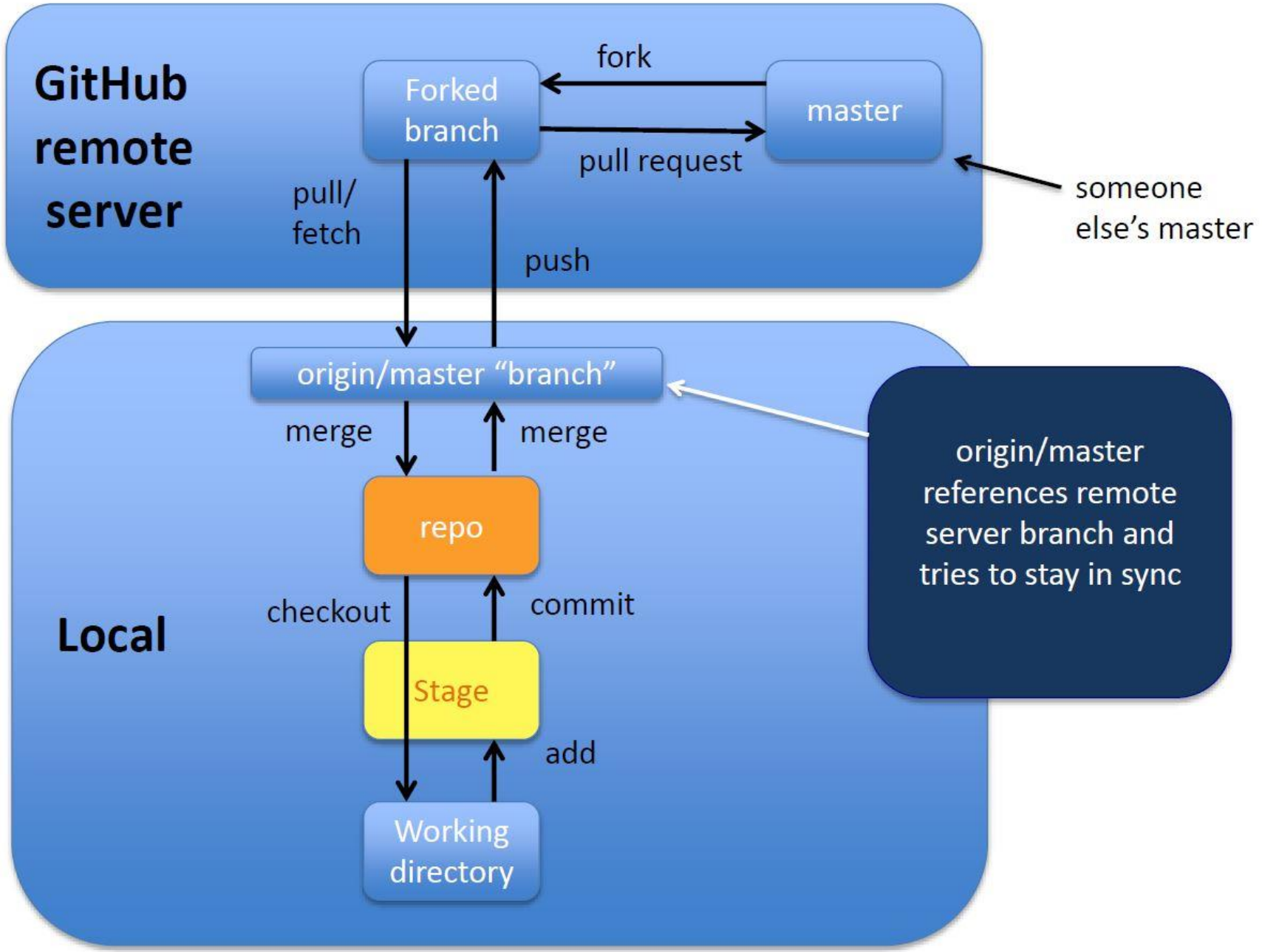
Forward School

GitHub

- **a platform to host git code repositories**
- http://github.com
- most popular Git host
- allows users to collaborate on projects from anywhere
- GitHub makes git social!
- Free to start

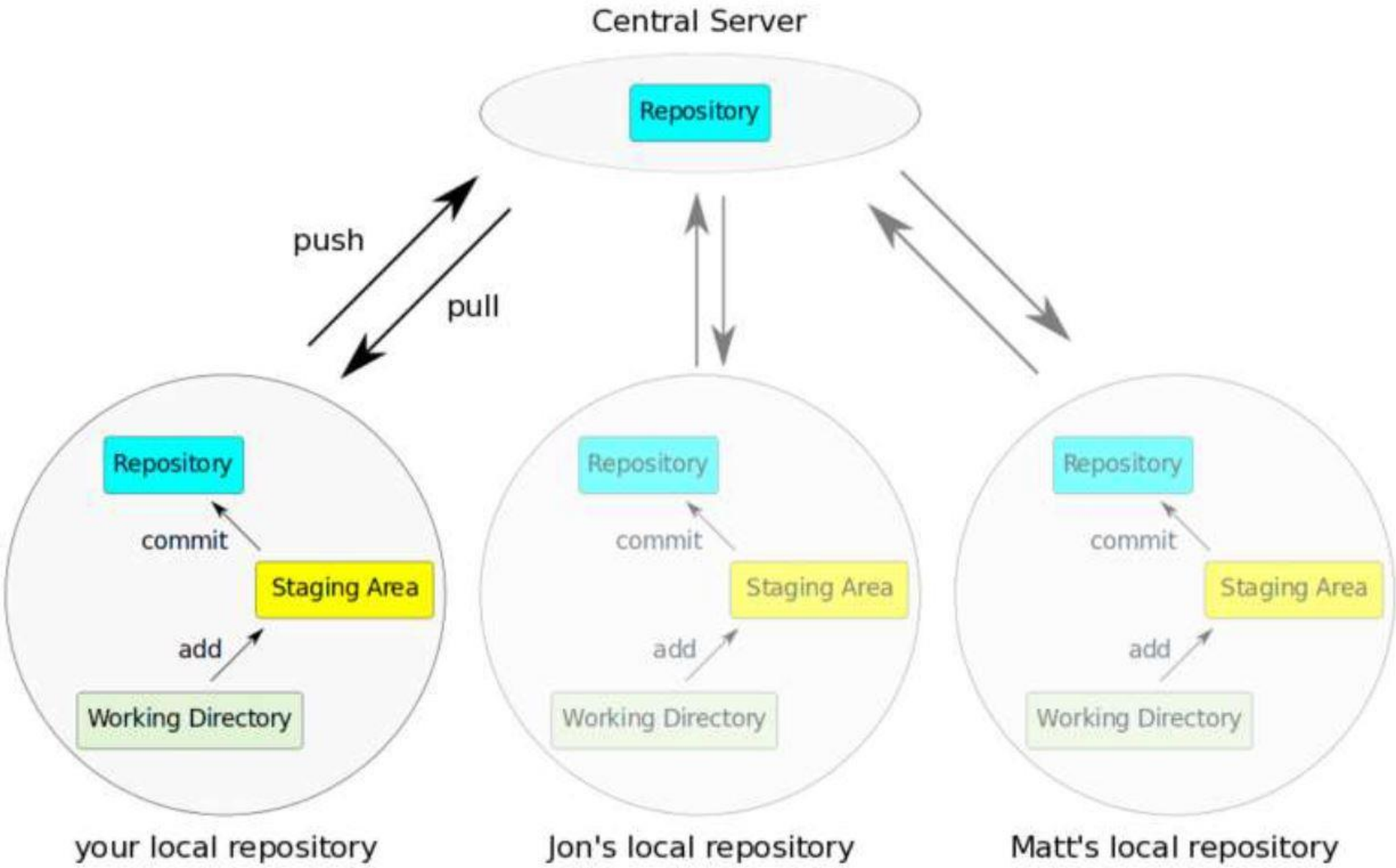
GitHub

Forward School



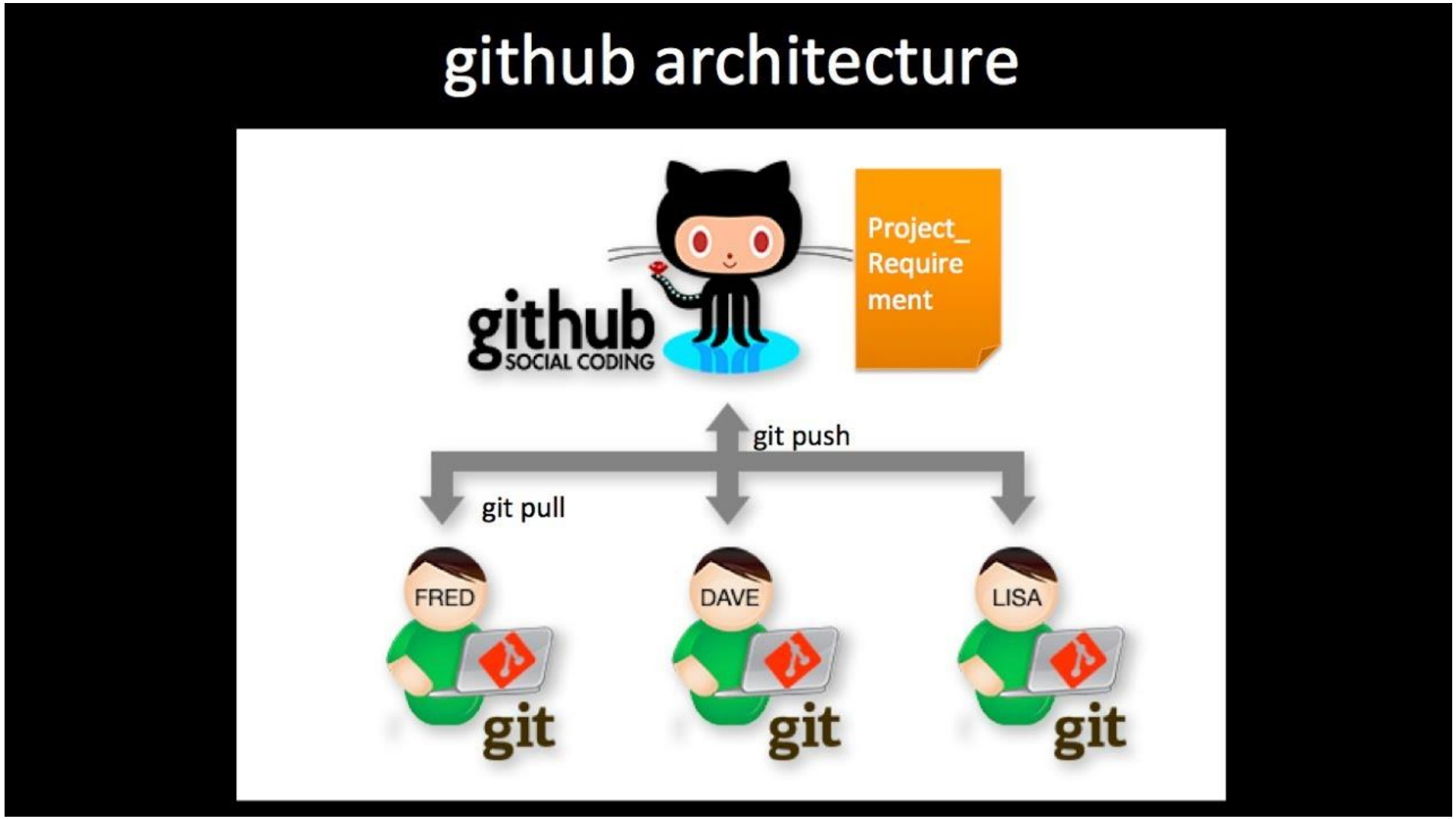
GitHub

Forward School



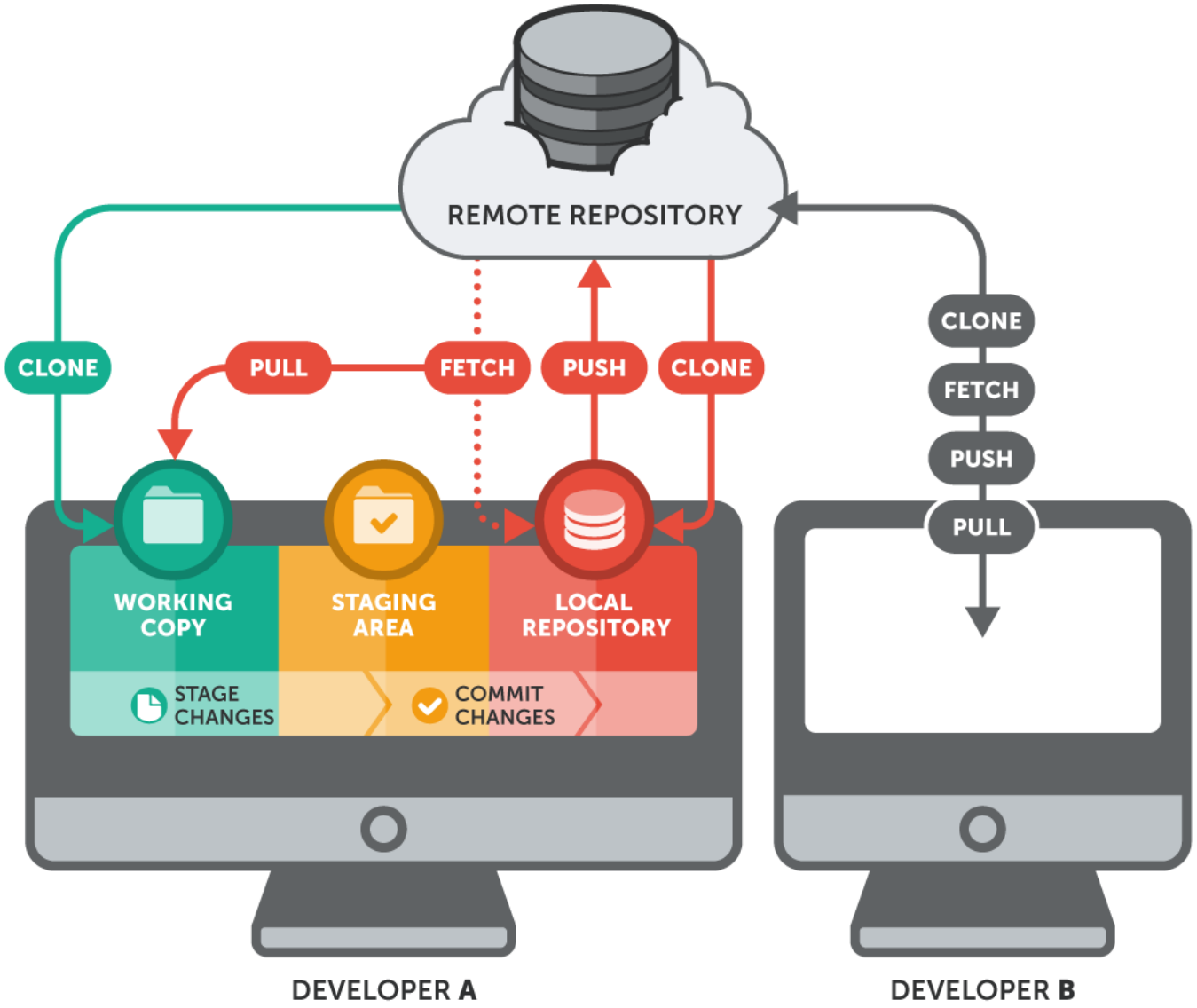
Git and GitHub Architecture

Forward School



Git and GitHub Architecture

Forward School



Remote Repository

Forward School

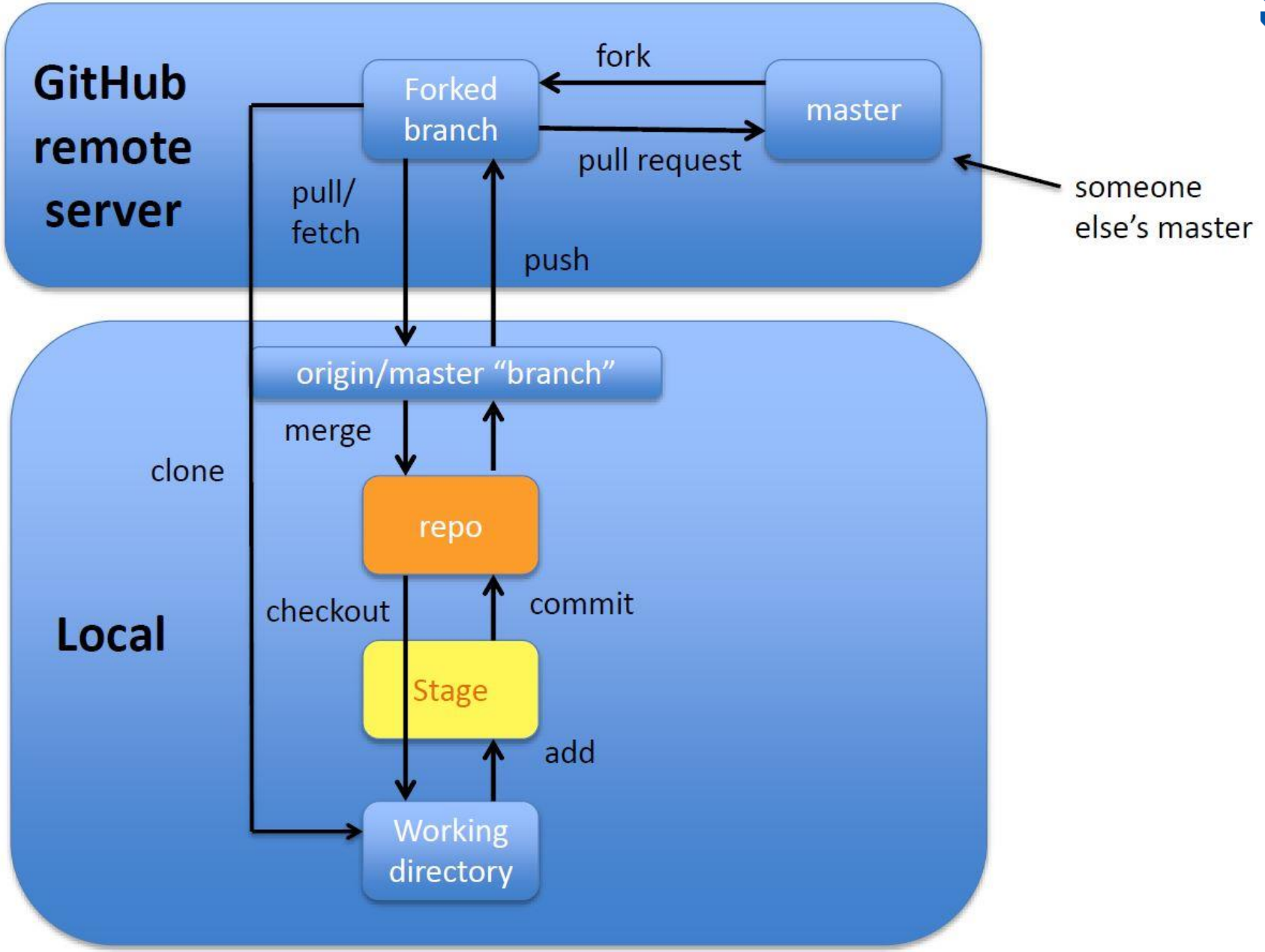
Git Commands

Forward School

- git clone URL <new_dir_name> – Clone files from remote repo to local machine.
- git remote add <alias> <URL> – Link my local repo to a remote repo.
- git remote – Check the remote that git linked to.
- git push local_branch_alias branch_name – Pushing to a remote repo.
- git fetch remote_repo_name – Fetching from a remote repo.

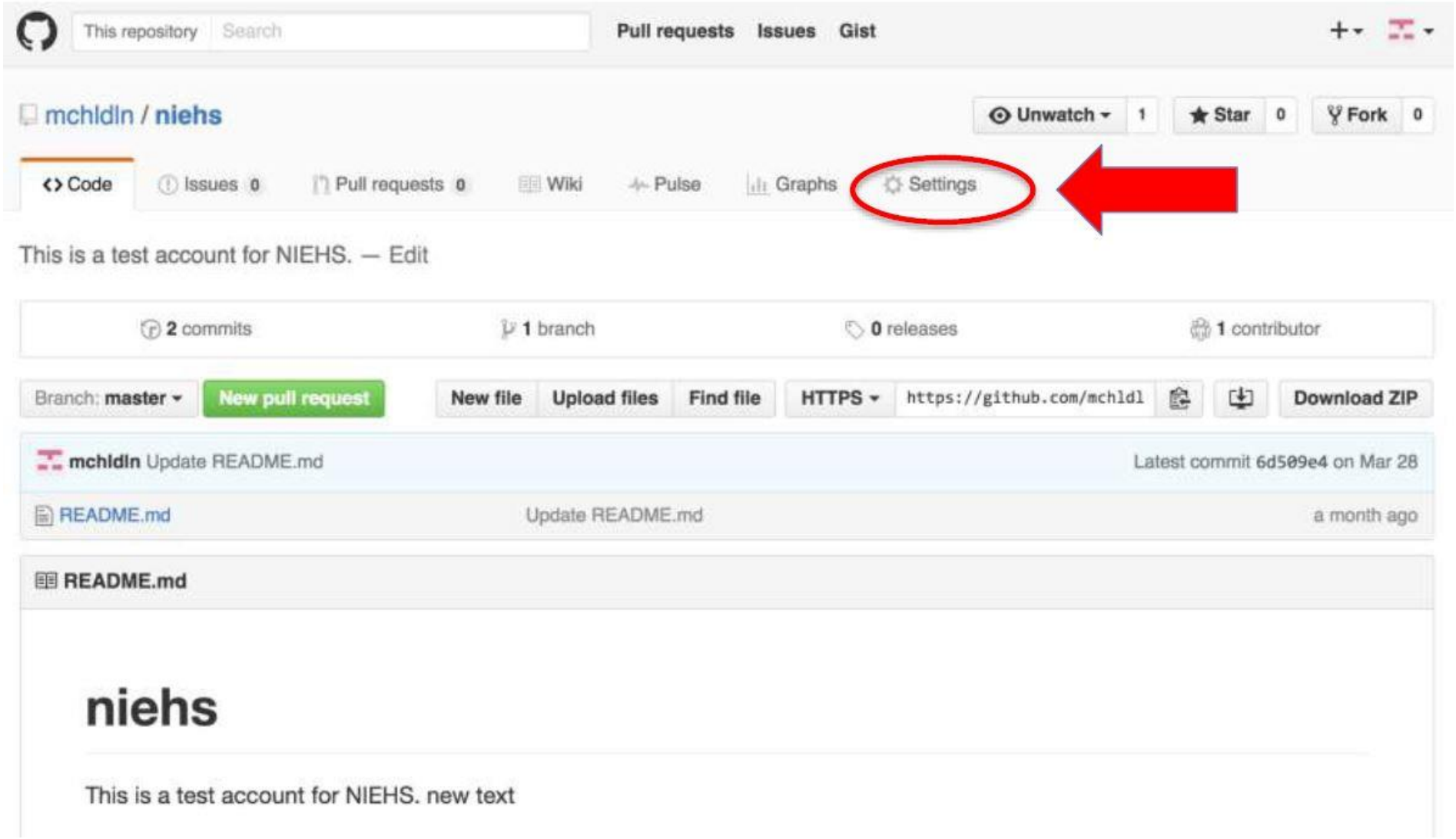
GitHub - Cloning

Forward School



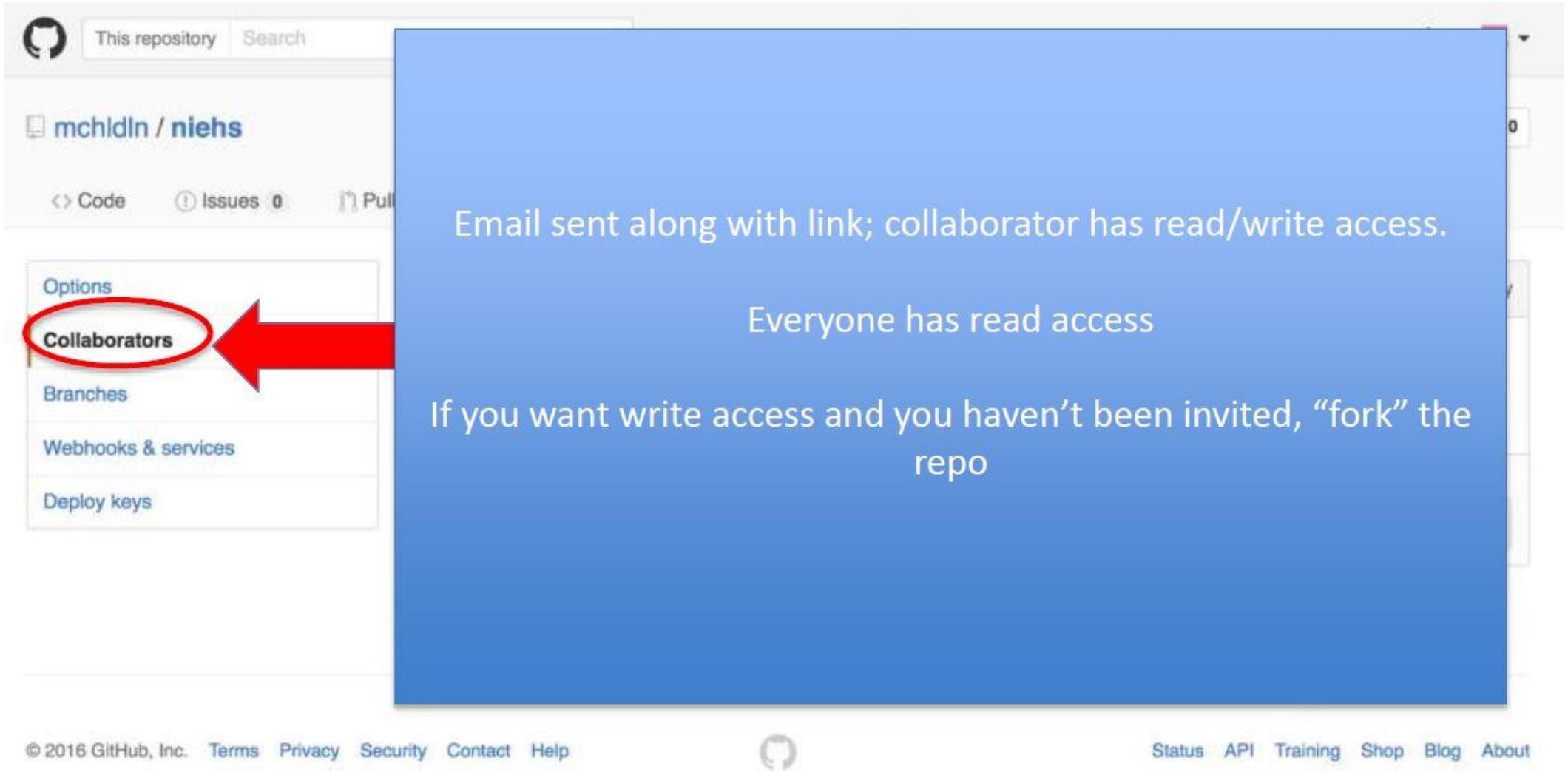
Collaborating in GitHub

Forward School



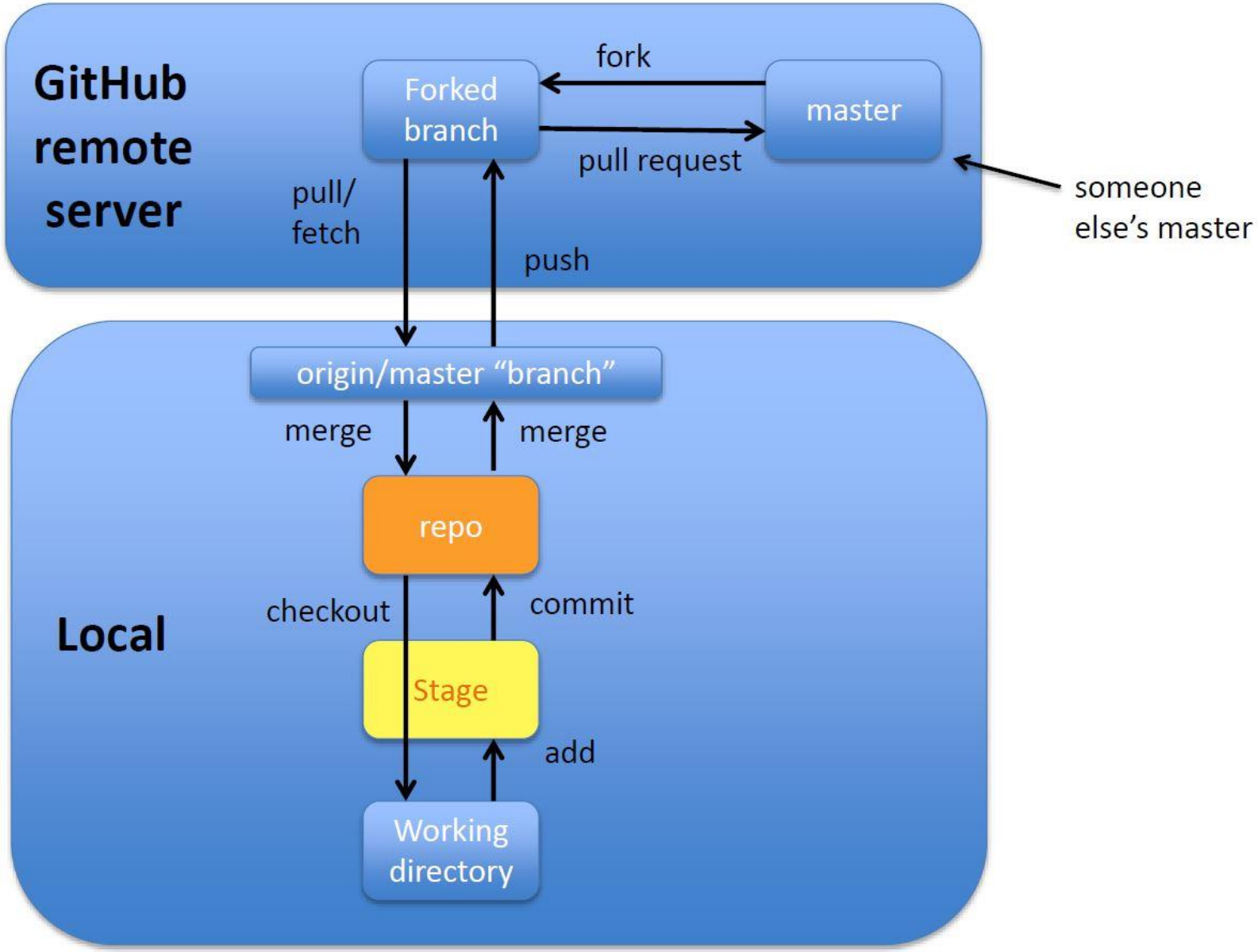
Collaborating in GitHub

Forward School



Collaborating in GitHub

Forward School



Summary of Git Commands

Forward School

- git commit –a – Allows one to add to staging index and commit *at the same time.*
- git log –online – Gets first line and checksum of all commits in current branch.
- git branch –m/--move old_name new_name – Rename the branches
- git branch –d branch_name – Delete the branches
- git tag – Light weight tagging. a pointer to a specific comment basically a SHA stored in a file
- git tag –a tag_name –m “message” – Annotated. A full object stored in the Git database SHA, tagger name, email, date and message.
- git show tag_name – Display the tags.

Useful Resources

Forward School

- Git from Git: https://git-scm.com/book/en/v2
- A guided tour that walks through the fundamentals of Git: https://githowto.com
- Linus Torvalds on Git: https://www.youtube.com/watch?v=idLyobOhtO4
- Git tutorial from Atlassian: https://www.atlassian.com/git/tutorials/
- A number of guides by the GitHub folks: https://guides.github.com

Keep it Simple

Forward School

- Make sure you are current with the central repository
- Make some improvements to your code
- Update the central repository before anyone else does
- Then you don’t have to worry about resolving conflicts or working with multiple branches
  - All the complexity in git comes from dealing with these
- Therefore:
  - Make sure you are up-to-date before starting to work
  - Commit and update the central repository frequently

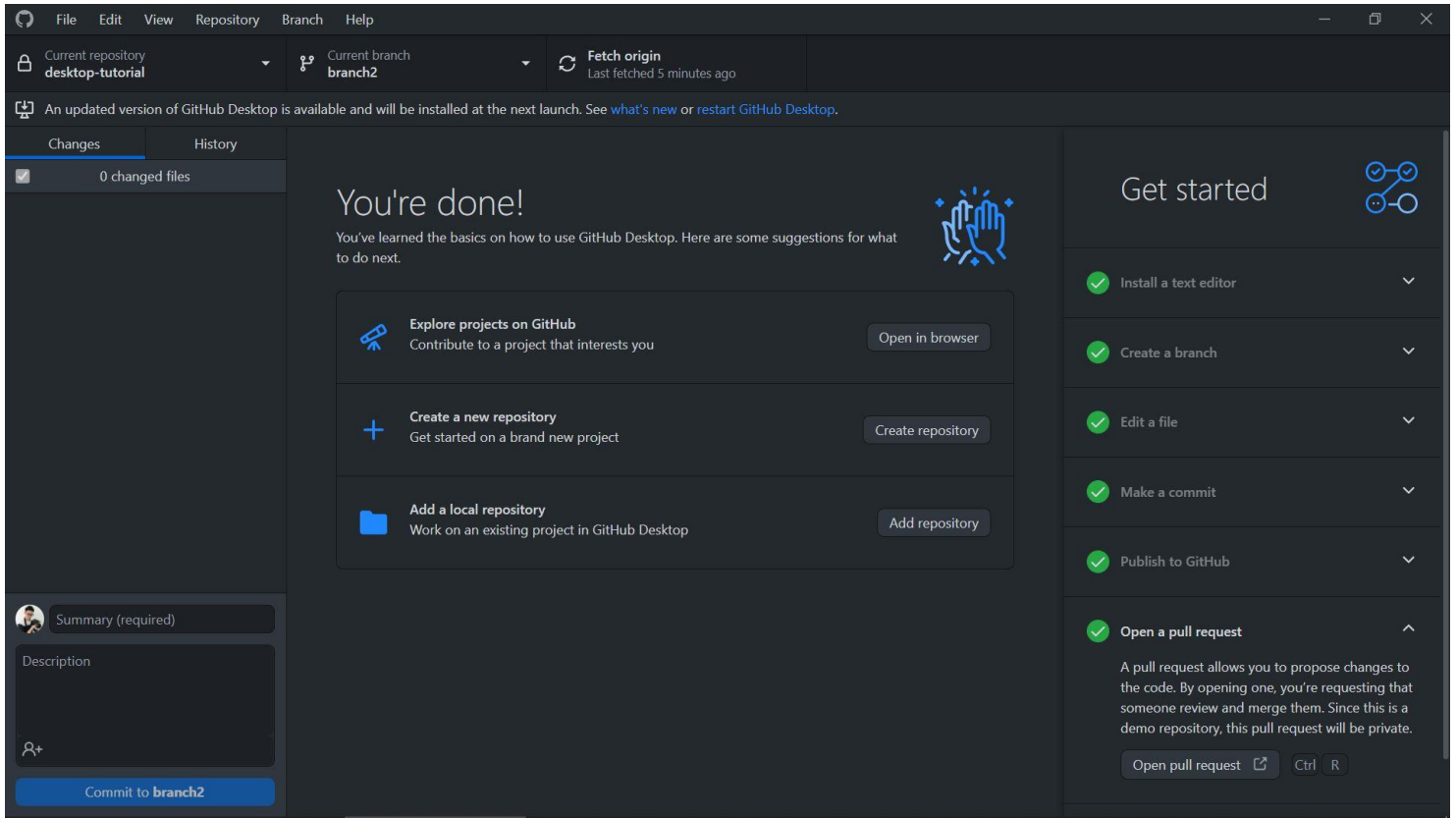
Get Started with Git and Git Hub

Forward School

Activity 1 – GitHub Desktop (30mins)

- Download GitHub Desktop from link https://desktop.github.com/
- Completed the 6 steps tutorials.

GitHub



Get Started with Git and Git Hub

Forward School

Activity 2 – Git (15mins)

- Download Link: http://git-scm.com/downloads
- Create the 6 steps tutorial by using Git Bash and Git GUI