Introduction to Scientific Computing I

Lab 1

Amir Farbin

Notebooks

- Lets do a walk through

"Shell"

- Without a GUI, the typical interface to a machine is a command-line shell.

    - Presents a prompt… indicating where in the filesystem you are.

    - You type in a command → press enter to execute

- Typically shells

    - Use to mostly run other programs… by just typing in the name of the program.

        - They have to know where to look for programs to run…

    - In Unix, almost everything is a program: ls, rm, mkdir, rmdir, …

    - Have some native keywords: e.g. cd

    - Allow manipulation of environment variables:

        - map of string → string that programs can use to communicate basic things, e.g. where to find libraries.

        - Shell use the "PATH" environment variable to know where to look for programs.

Coding Workflow

- Most basic:

  - Use a text editor to edit code files (e.g. foo.py)

    - Generally write explanations into the code as comments

  - If not interpreted: Compile/link

  - Test in a shell

- Integrated Development Environments (IDEs)

  - Provide a unified place where you write your code, execute for testing, and debug

- Notebooks

 - Combine text (e.g. instructions or explanations), code, and output

git

- git is a Version Control System (VCS)

  - Useful for code development or writing a large document

  - Keeps track of the evolution of files

  - Facilitates collaboration between multiple coders or authors

  - Allows concurrent development of multiple versions

  - Can be completely local or on a server

  - Enables establishing releases

- GitLab is a implementation of a git server.

  - Many companies or projects host their own GitLab service enabling their employees or participants to collaborate.

  - Can automate the process of incorporating new code or changes to code, testing, and building releases.

  - Also provides a browser, markdown documentation, wiki, and other convenient features.

- GitHub is a service that runs a public instance of a git server

  - Has become the de-facto mechanism for sharing open-source code

Git Concepts

- **Repository**: a container for all of the source code/docs for a specific project. Typically consists of:

  - **Index:** keeps all of the information about the files, including previous versions, comments, ….

  - **Working directory:** a copy of the files that you can use or modify.

- **Add**: add a new file in the working directory to the index

- **Commit**: puts the current state of file(s) into the index

all collaborators has a repository each
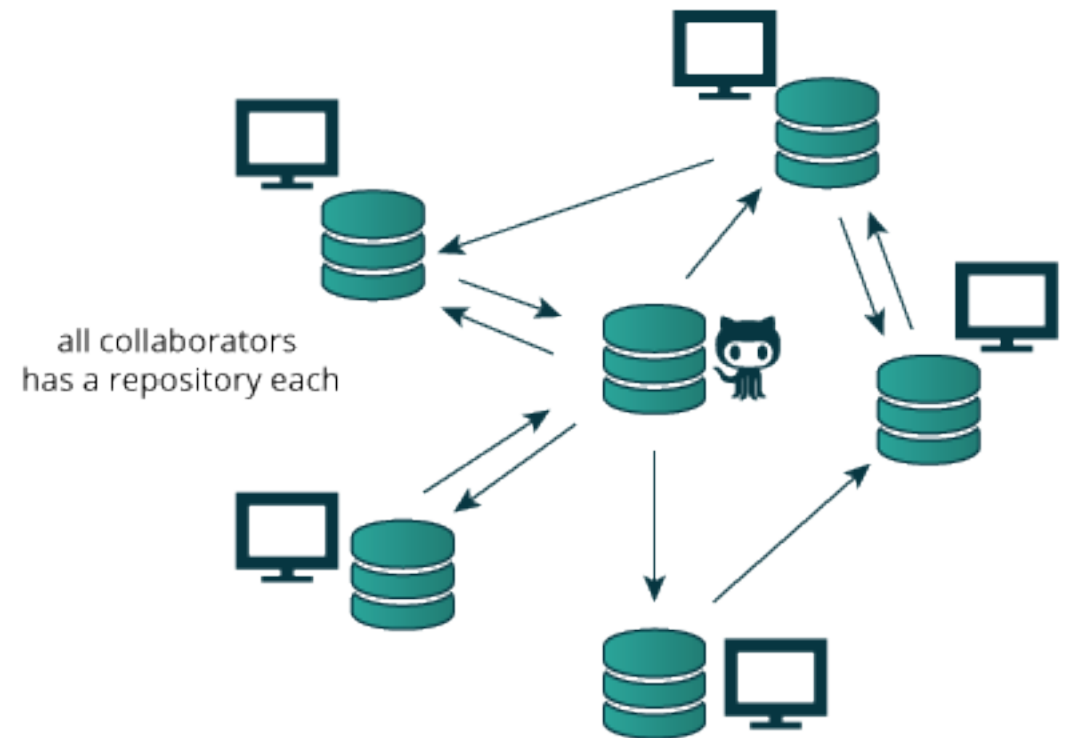
- **Clone**: a copy of a repository, usually local, and typically what the user interacts with.

- **Remote:** an instance of the repository in on a server.

    - **origin:** where your local commits are **pushed**

    - **upstream:** where changes/updates are **pulled** into your **local** repository

    - for your own packages origin and upstream will likely be the same.

- **Fork:** a clone of a repository that can evolve independently from other clones.

- **Branch:** a parallel version of the repository that doesn't conflict with others.

    - **Master:** name of the main branch.

    - Branches can be later merged.

    - Typically, someone will develop something new in their own branch, and then merge it with master when tested.

- **Tag:** Name associated with a specific version of all of the files
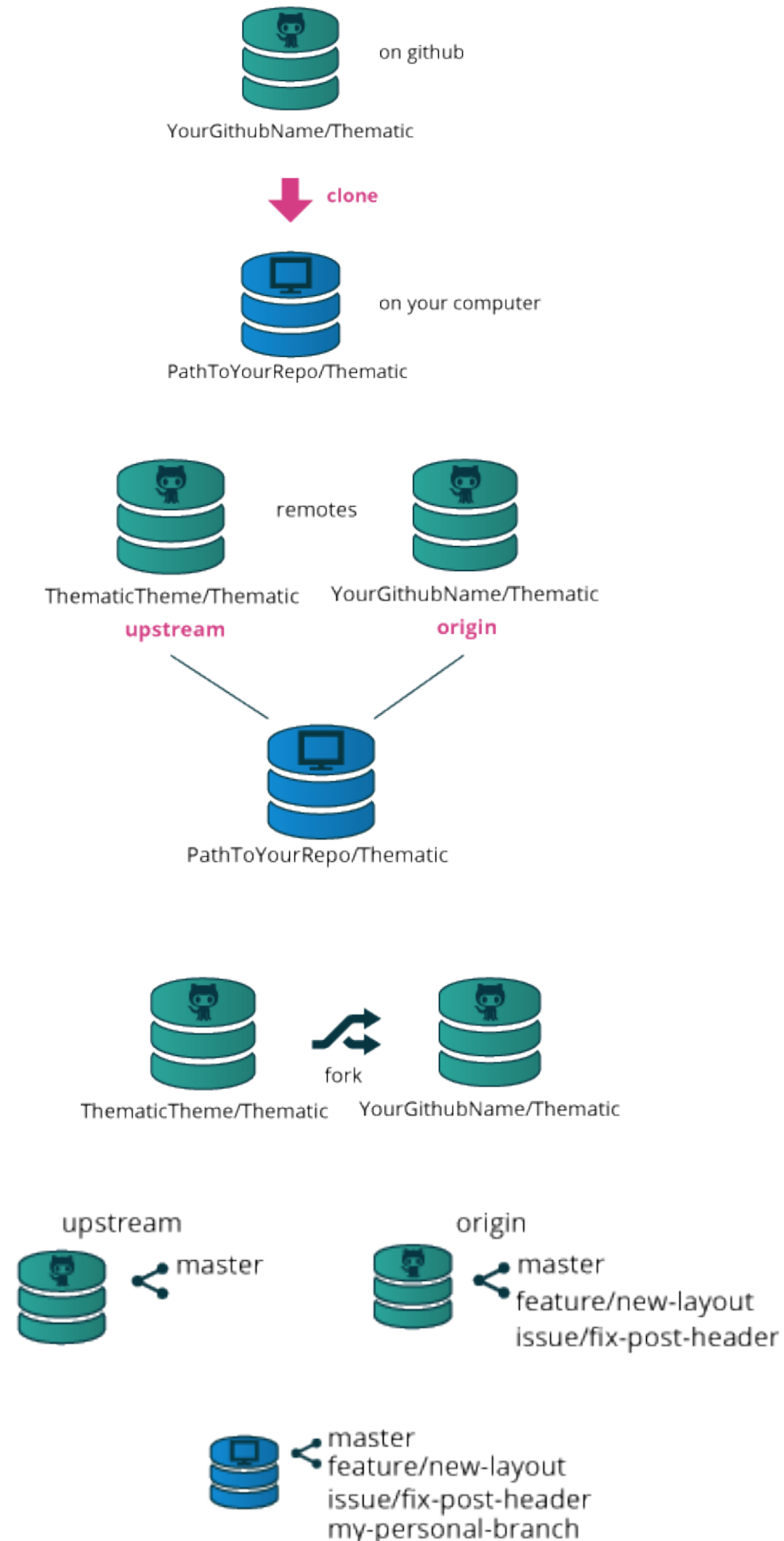
- **Fetch**: getting updates from a remote into your index

- **Merge:** incorporation changes in index into your working directory

  - **Merge conflict**: when incorporating changes isn't trivial and requires a manual **merge resolution**.

    - For example if two different people work on same file simultaneously

      - The first person to push to the remote would have no issues

      - The second person would have to pull from the repository and resolve any conflicts

- **Pull** = fetch (from upstream) + merge

  - **Pull request**: asking for pulling of your fork into another.

- **Push:** puts your index into your origin remote