

Tools & Environment Setup

Mobile Web Services

Olivier Liechti & Yannick Iseli

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Initiate downloads....

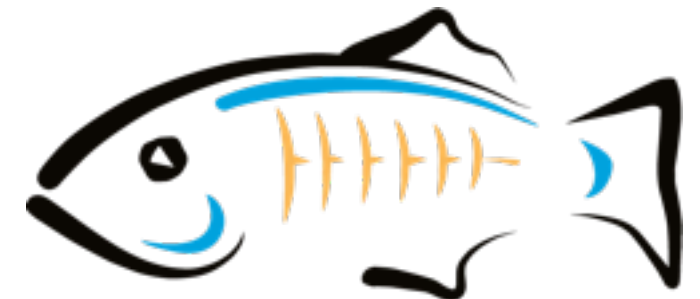
- **Step 1: download Netbeans 8**

- <https://netbeans.org/downloads/index.html>
- Select either the “Java EE” bundle or the “All” bundle



- **Step 2: download apache maven**

- <http://maven.apache.org/download.cgi>



- ***And now that bits are being moved around... let's get to work!***



Git & Github

- **Step 1: install Git**

- <http://git-scm.com/downloads>
- <http://git-scm.com/book/en/Getting-Started-Installing-Git>
- Check point: are you able to invoke the git command from the shell?

- **Step 2: configure Git**

- <https://help.github.com/articles/set-up-git>

- **Step 3: configure SSH**

- <http://guides.beanstalkapp.com/version-control/git-on-windows.html#installing-ssh-keys>



Using Git locally

```
$ mkdir my-project  
$ cd my-project  
$ git init  
$ ls -al
```

- **You do not *have to* use a server:** Git is already useful to manage versions of your files on your local machine.
- The **git init** command creates a **local repository**. If you look carefully, you will see a **hidden .git directory**, where Git keeps all of his data.
- **Important:** your **my-project** directory is your **working directory**. If you simply create files in it, they will not immediately be part of your repository!

Using Git locally

```
$ echo "text a" > a.txt
$ git status
$ git add a.txt
$ git commit -m "First version of a.txt"
$ echo "my mod on text a" > a.txt
$ git status
```

- A **commit** is a **snapshot** of your repository. Git maintains a **graph of commits** and you can always **recover the state** of a particular commit.
- When **you have modified files in your working directory**, you need to specify which ones should be **part of the next commit**.
- You use the **git add** command to add a file to the so-called **staging area**. It will be part of the next commit.
- You use the **git status** command to **check the content** of your working directory and of your staging area.

More info: <http://git-scm.com/book/en/Git-Basics-Recording-Changes-to-the-Repository>

Working Dir, Staging Area & Repository

This is a local directory, not
a remote server!

git add is used for both

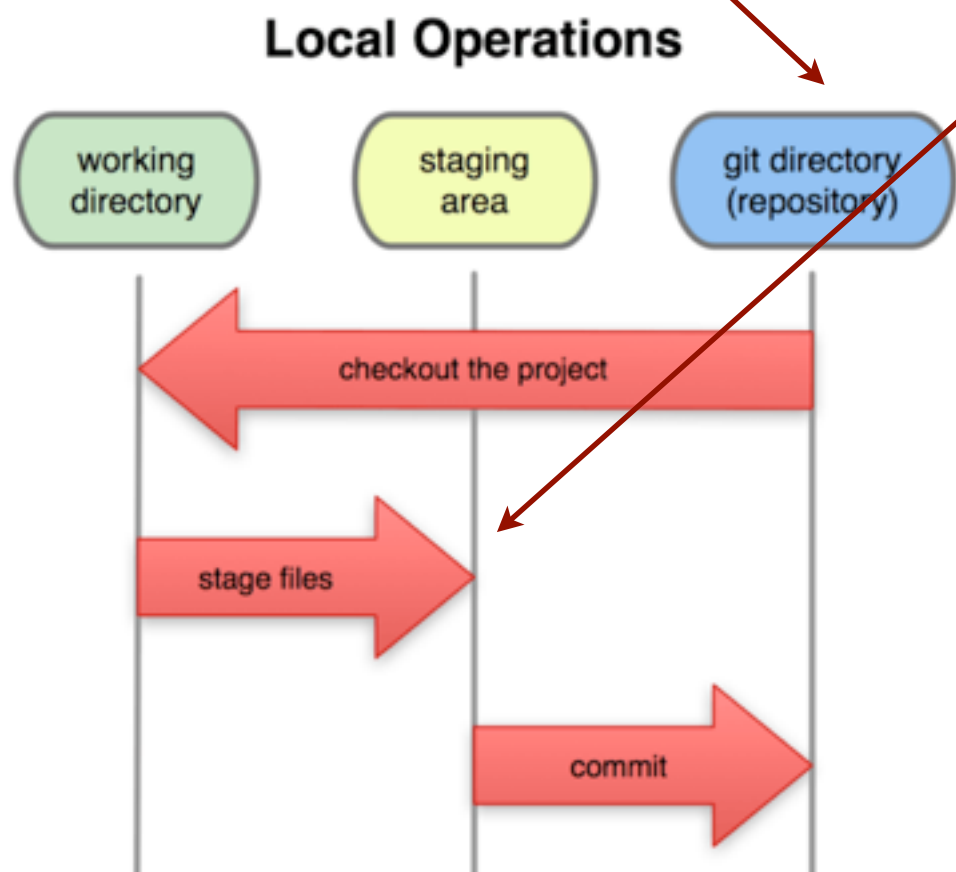


Figure 1-6. Working directory, staging area, and git directory.

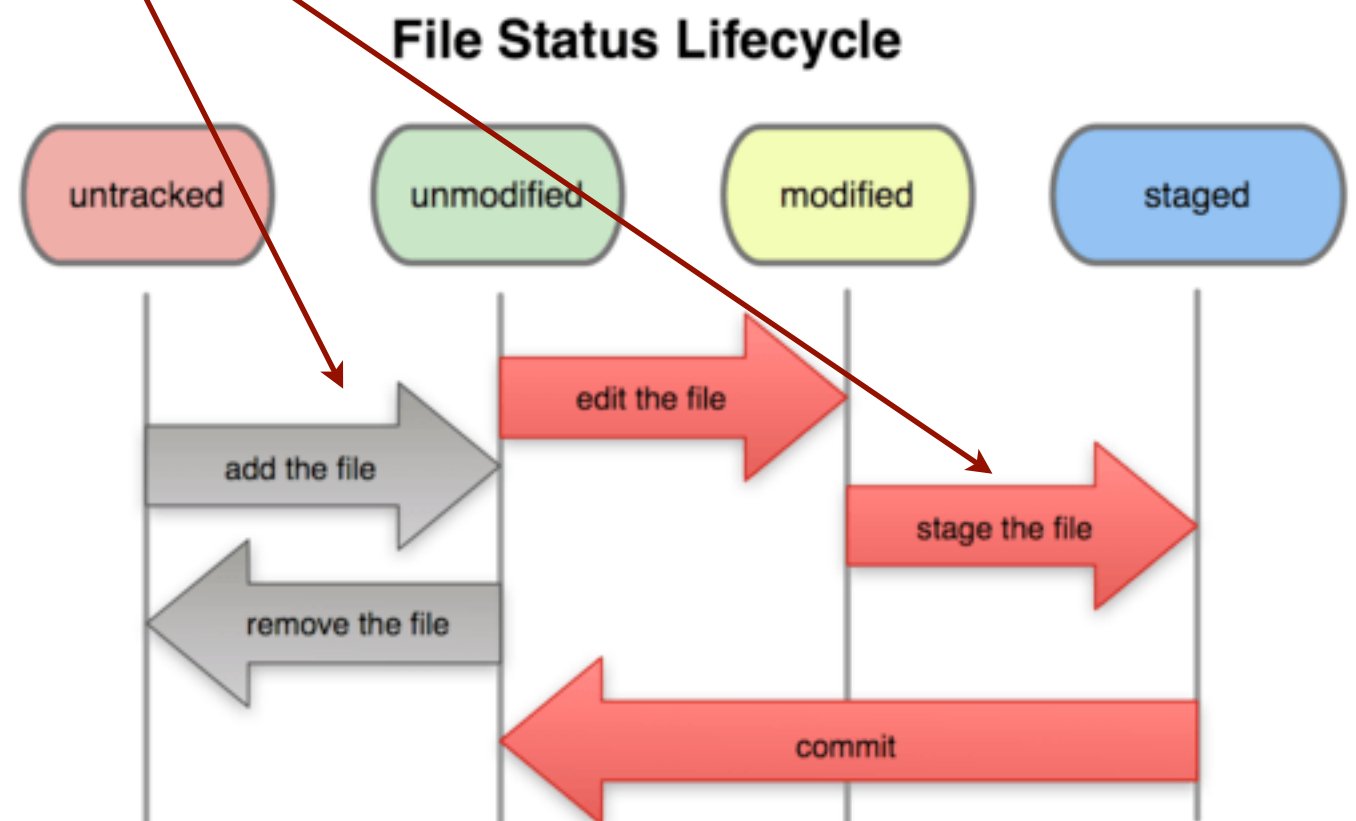


Figure 2-1. The lifecycle of the status of your files.

Git Commits & Snapshots

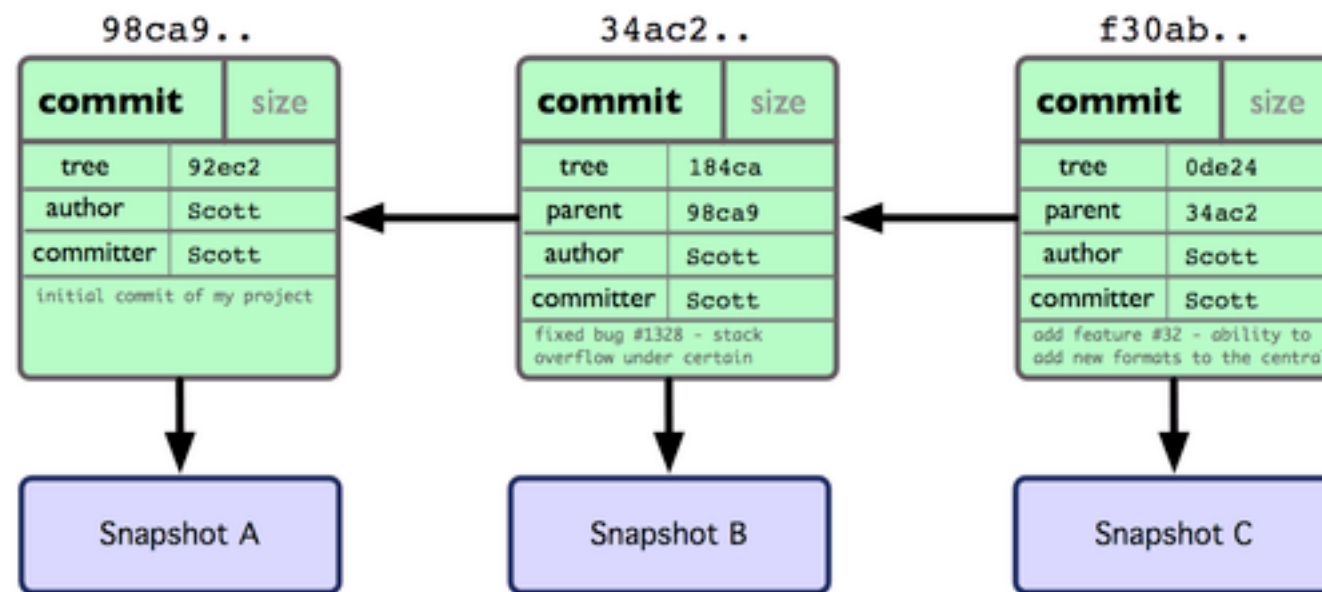


Figure 3-2. Git object data for multiple commits.

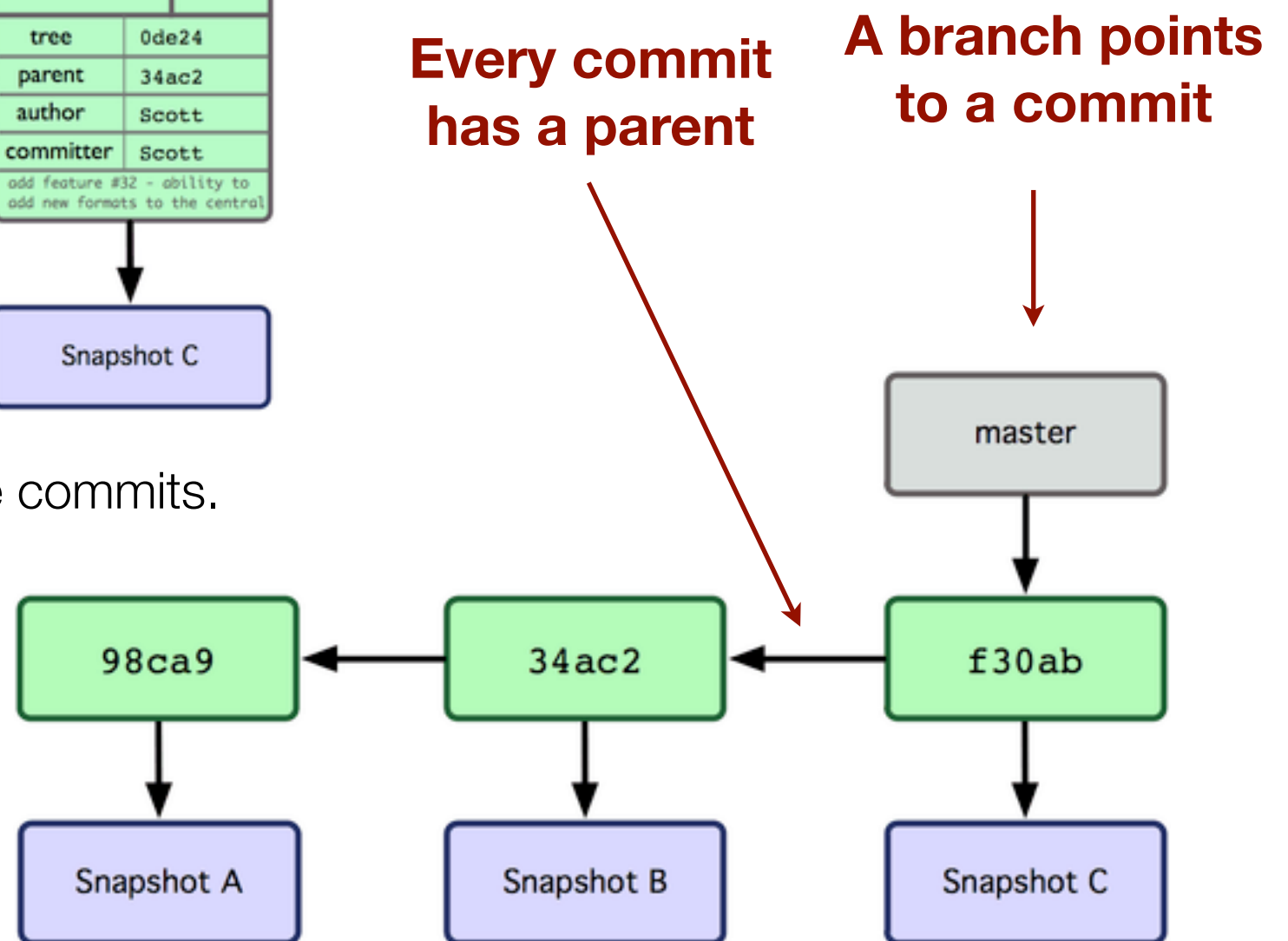


Figure 3-3. Branch pointing into the commit data's history.

Git Branches & Workflows

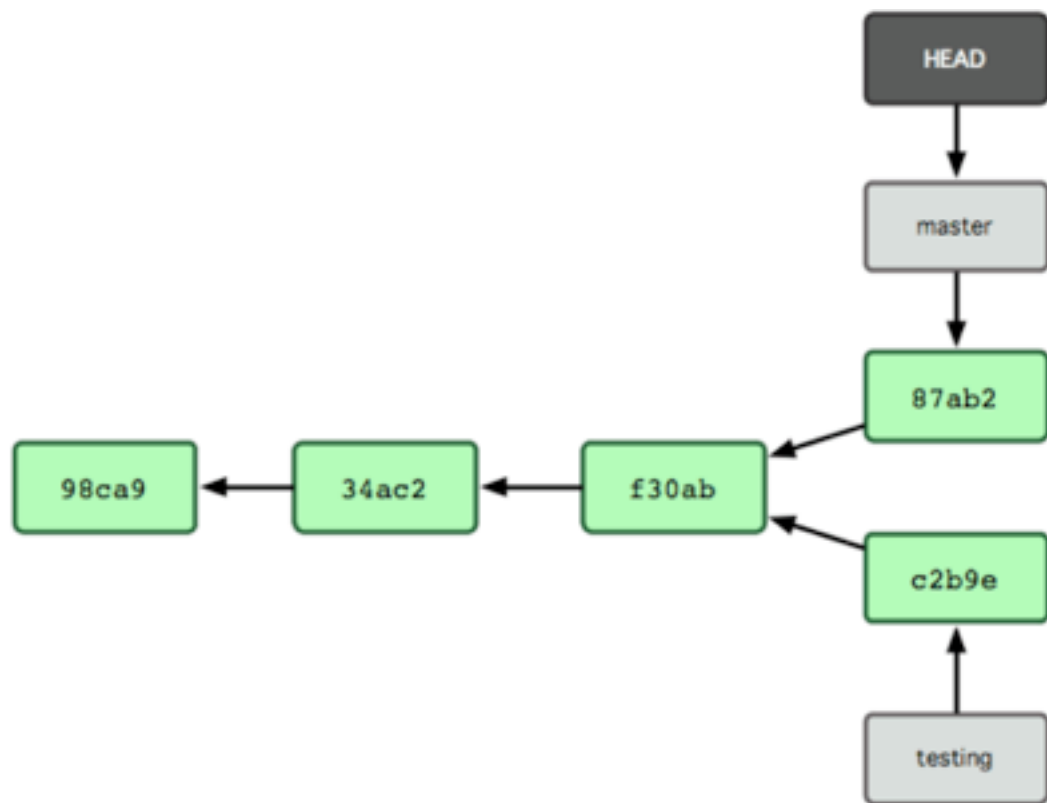


Figure 3-3. Branch pointing into the commit data's history.

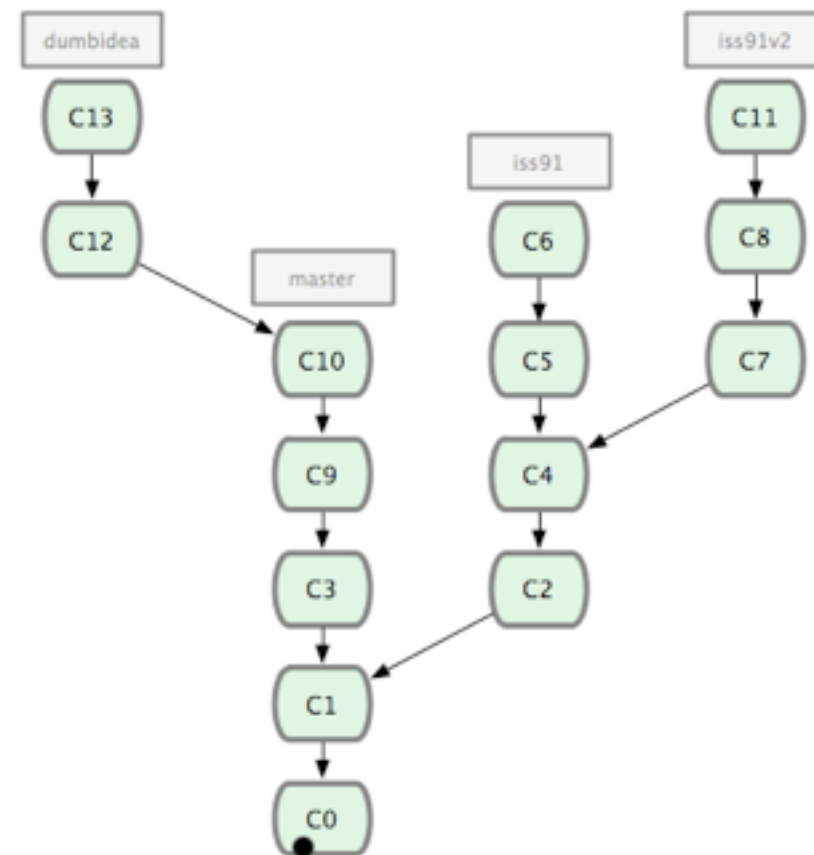
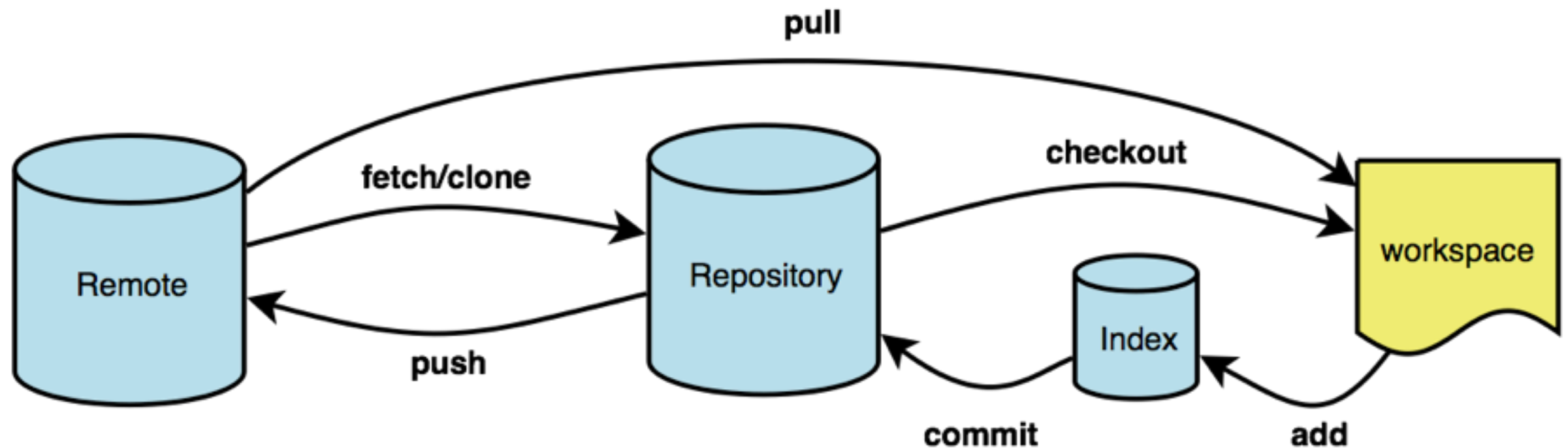


Figure 3-20. Your commit history with multiple topic branches.

Git & Remote Repositories



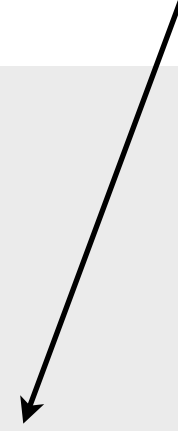
Source: <http://illustrated-git.readthedocs.org/en/latest/>



Source: <http://bramus.github.io/ws2-sws-course-materials/xx.git.html#/4/1>

Going Back To The Future

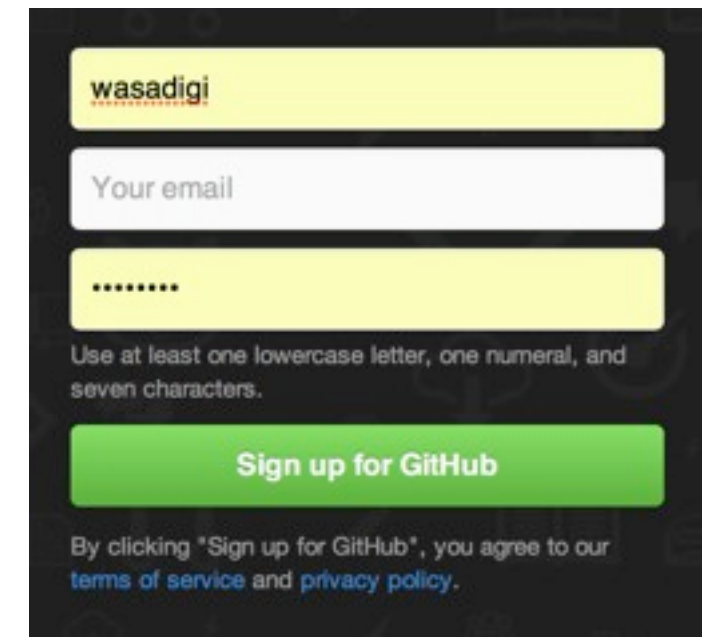
```
$ git add a.txt
$ git commit -m "Modif on a.txt"
$ git log
$ cat a.txt
$ git checkout 2c2363929018ffca88e53a801711f9f31fd2a7d7 .
$ cat a.txt
```



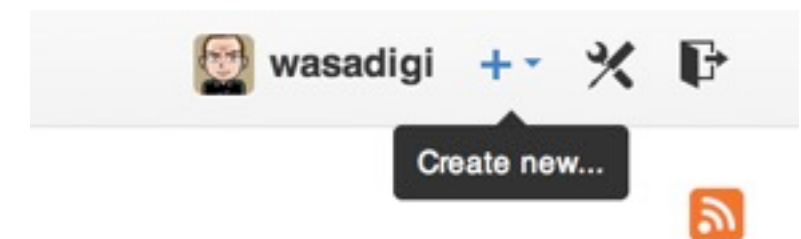
- If you have committed your working directory several times, your repository will contain **several snapshot** (remember: one commit = one snapshot).
- Each commit has a **unique identifier (hash)**
- With the **git checkout** command, you can retrieve a particular commit (based on its hash).
- After running the command, your **working directory** will now contain files in the state that they were at in that particular point in time.

Github Setup

- **Sign up** for GitHub and get your own account:
 - Go to <http://www.github.com>
- Add your **SSH key**:
 - Go to your accounts settings. You will find an option to manage your SSH keys.
 - If you don't have a SSH key yet, follow the instructions in the online help.
 - If you are using windows, you will need to use Git BASH.
- **Create** your first repo, hosted on Github.
- **Copy the SSH URL** of the repo.



The image shows the GitHub sign-up form. It has a yellow header with the username 'wasadigi'. Below it is a white input field for 'Your email', followed by a yellow input field for a password (represented by dots). A note below the password field says: 'Use at least one lowercase letter, one numeral, and seven characters.' At the bottom is a green button labeled 'Sign up for GitHub'. Below the button, it says: 'By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#).'



SSH clone URL

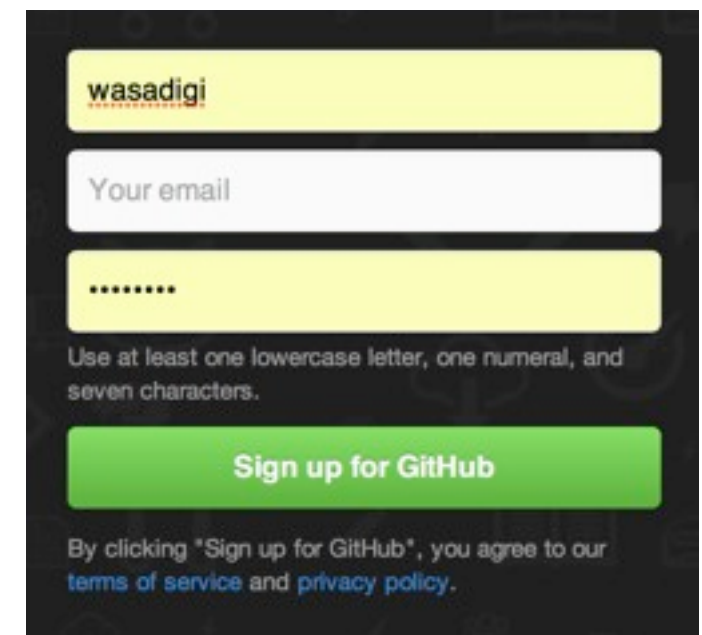
git@github.com:was

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#). ?

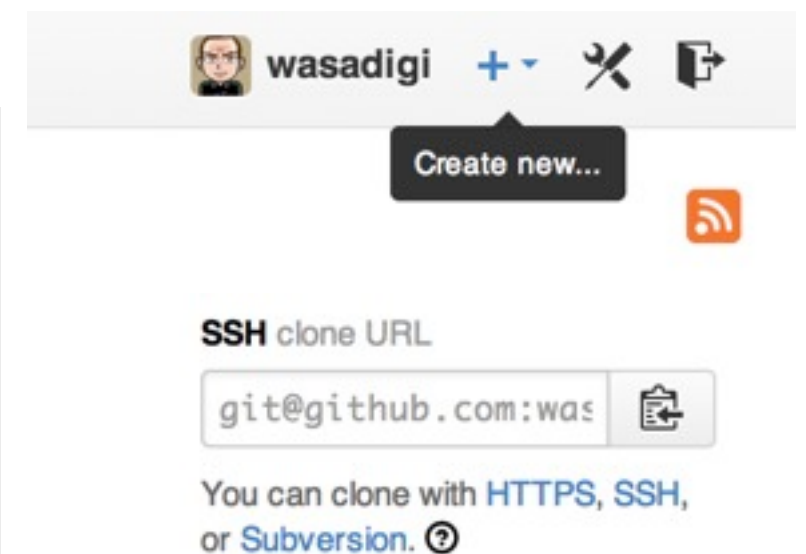
Validate your setup

- **Clone your repo to your laptop**
 - Open a **terminal window** (Terminal on Mac OS, Git BASH on Windows, etc.)
 - Create a **new directory** to host your clone of the repo and get into it.
 - **Clone** the repo, using the **SSH URL**.
 - **Create** a file, **add** it to the staging area, **commit** the changes and finally **push** the commit Github.

```
$ mkdir myspace
$ cd myspace
$ git clone git@github.com:UUUUU/RRRRR.git
$ git touch firstFile.txt
$ git add firstFile.txt
$ git commit -m "I have added my first file"
$ git push
```



The image shows a GitHub sign-up form for a user named 'wasadigi'. It features a yellow header with the username, a white input field for 'Your email', and a yellow input field for a password (represented by dots). Below the password field, there is a note: 'Use at least one lowercase letter, one numeral, and seven characters.' A green button labeled 'Sign up for GitHub' is positioned below the form. At the bottom, a small text line states: 'By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#).'



Forks & Clones

If you do this, you will have **YOUR**
clone of **MY** repo hosted on Github

heig-vd

Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

The screenshot shows a web browser window displaying the GitHub repository page for 'wasadigi / Teaching-COMEM-MWS'. The browser's address bar shows the URL 'https://github.com/wasadigi/Teaching-COMEM-MWS'. The repository page includes a header with the repository name, a search bar, and navigation links. Below the header, there are statistics for the repository: 6 commits, 2 branches, 0 releases, and 1 contributor. A commit history table shows a recent commit by Olivier Liechti. The main content area displays the README file, which has a title 'Welcome to the Mobile Web Services (MWS) Git Repository' and an 'Introduction' section. The introduction text states: 'This repository is used for the Mobile Web Services course, organized at the University of Applied Sciences of Western Switzerland in the COMEM Department.' On the right side of the repository page, there is a sidebar with links to 'Code', 'Issues', 'Pull Requests', 'Wiki', 'Pulse', 'Graphs', 'Network', and 'Settings'. At the bottom of the sidebar, there is a section for cloning the repository, showing the SSH clone URL 'git@github.com:was' and buttons for 'Clone in Desktop' and 'Download ZIP'. A red arrow points from the text 'If you do this, you will have YOUR clone of MY repo hosted on Github' to the 'Fork' button in the repository header.

wasadigi / Teaching-COMEM-MWS

Unwatch 2 Star 0 Fork 0

Course on Mobile Web Services — Edit

6 commits 2 branches 0 releases 1 contributor

branch: master Teaching-COMEM-MWS

Adding new README.md file

Olivier Liechti authored 6 hours ago latest commit 7cc13a349f

README.md Adding new README.md file 6 hours ago

Welcome to the Mobile Web Services (MWS) Git Repository

Introduction

This repository is used for the Mobile Web Services course, organized at the University of Applied Sciences of Western Switzerland in the COMEM Department.

Code

Issues 0

Pull Requests 0

Wiki

Pulse

Graphs

Network

Settings

SSH clone URL

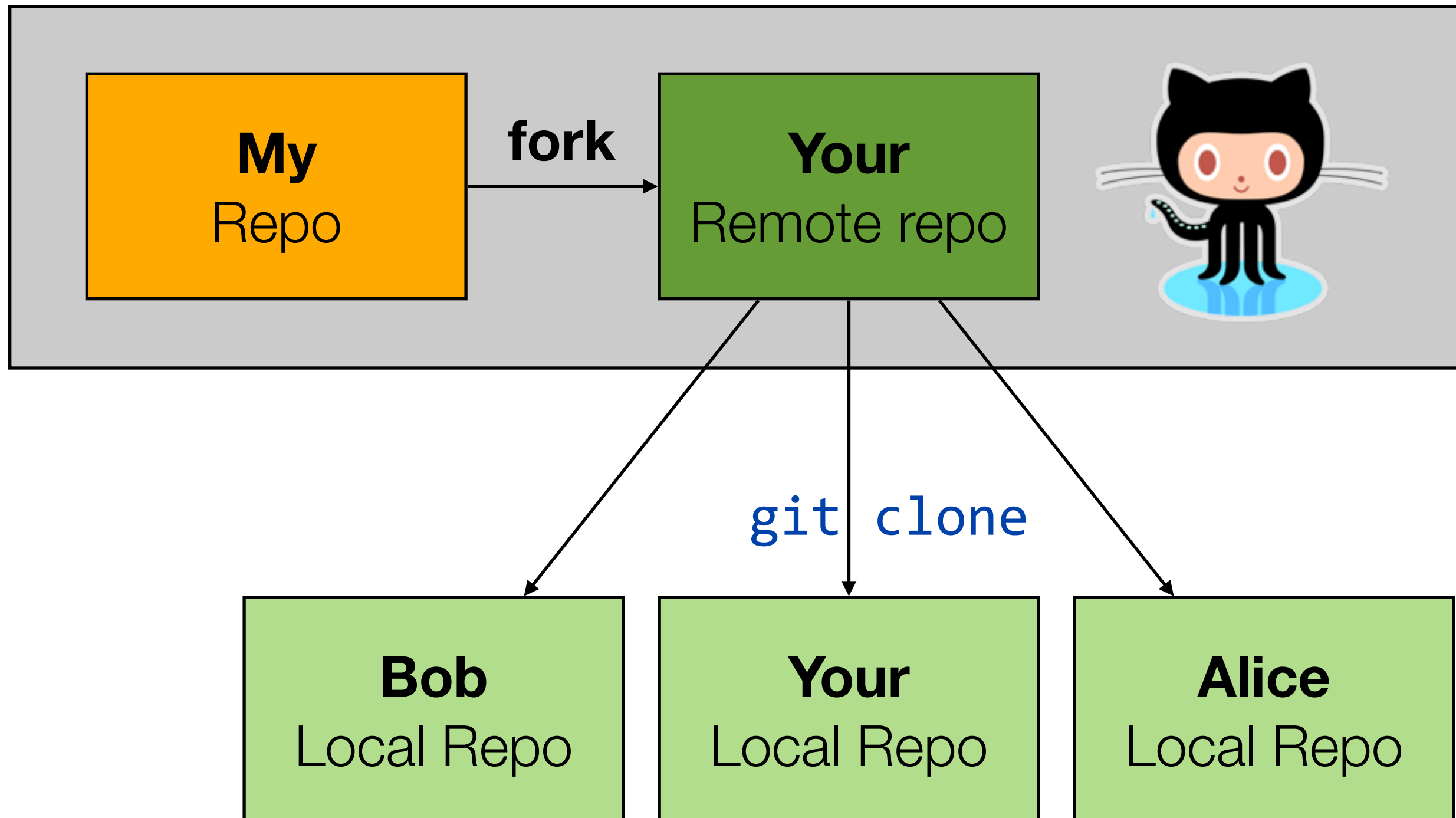
git@github.com:was

You can clone with HTTPS, SSH, or Subversion.

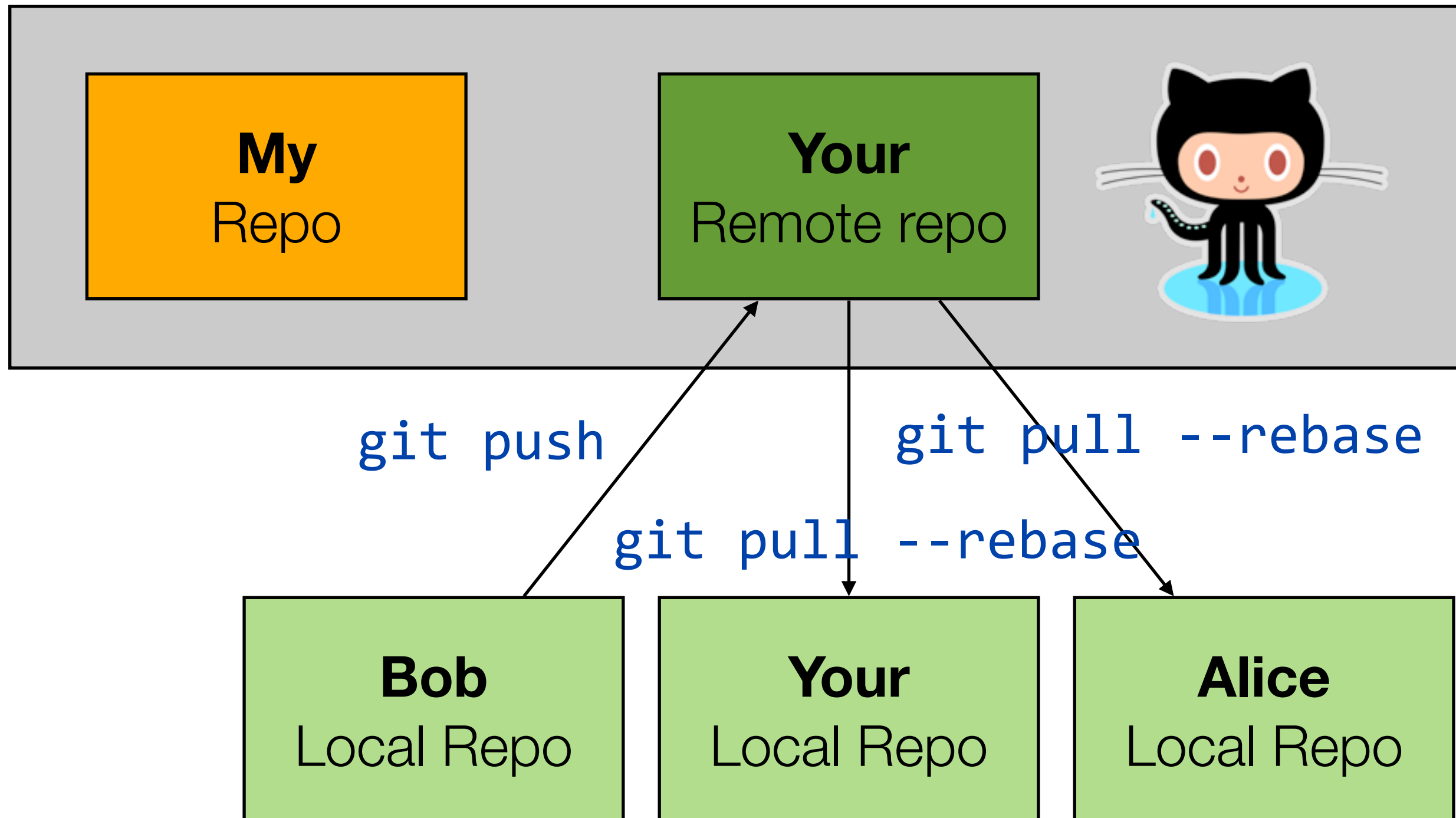
Clone in Desktop

Download ZIP

Forking My Repo on Github

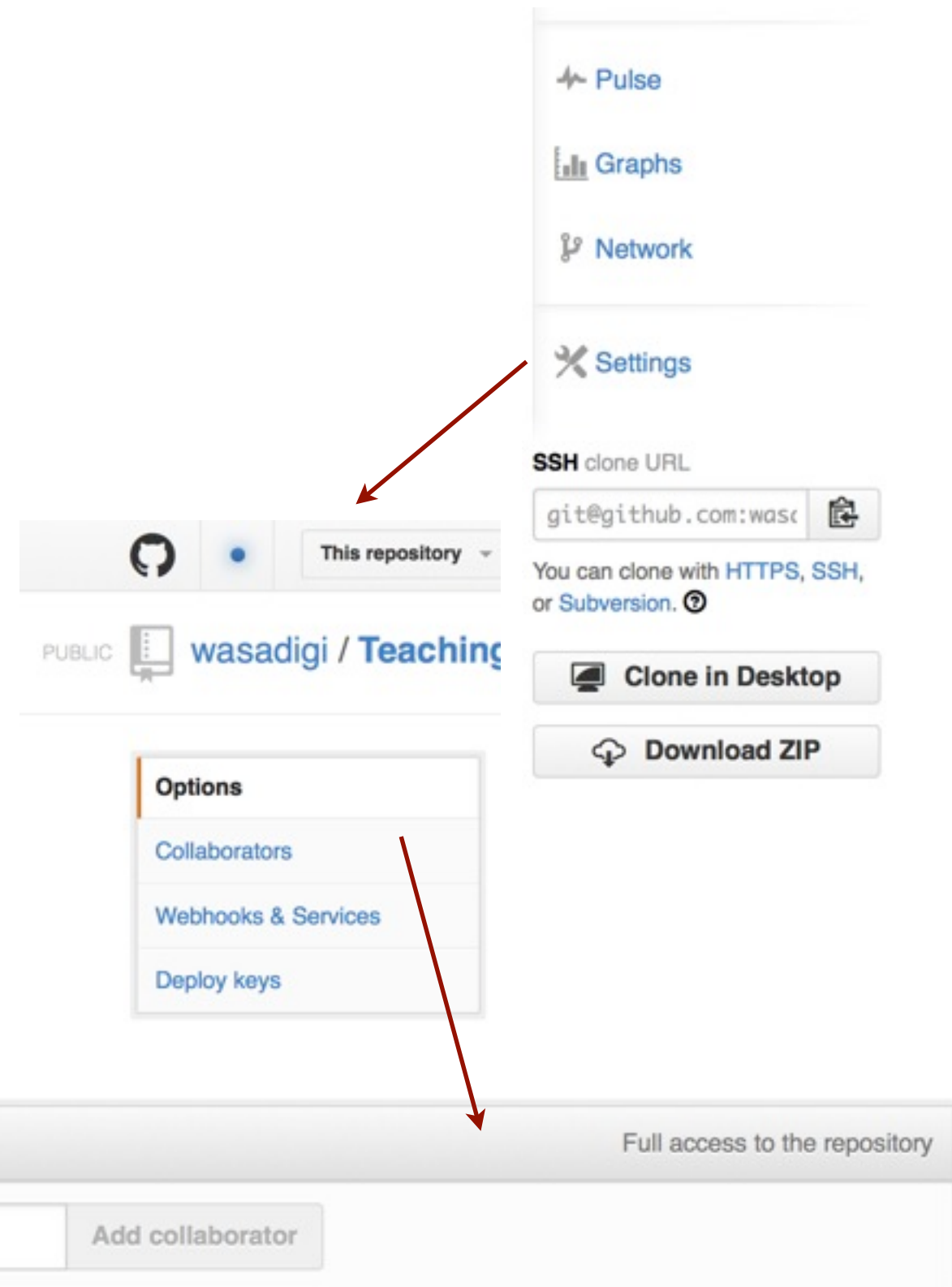


Forking My Repo on Github



Setup Your Forks

- **Setup the Project Team**
 - In each group, **one person** (and only one) needs to fork my repo.
 - Nominate the person and let him/her **fork my repo**.
 - Go to the settings of your fork and **add the Github users** of your team members.
 - This will allow everyone in the team to **push commits** to the fork.
- Everyone on the Team should then **clone** your fork on their laptop.

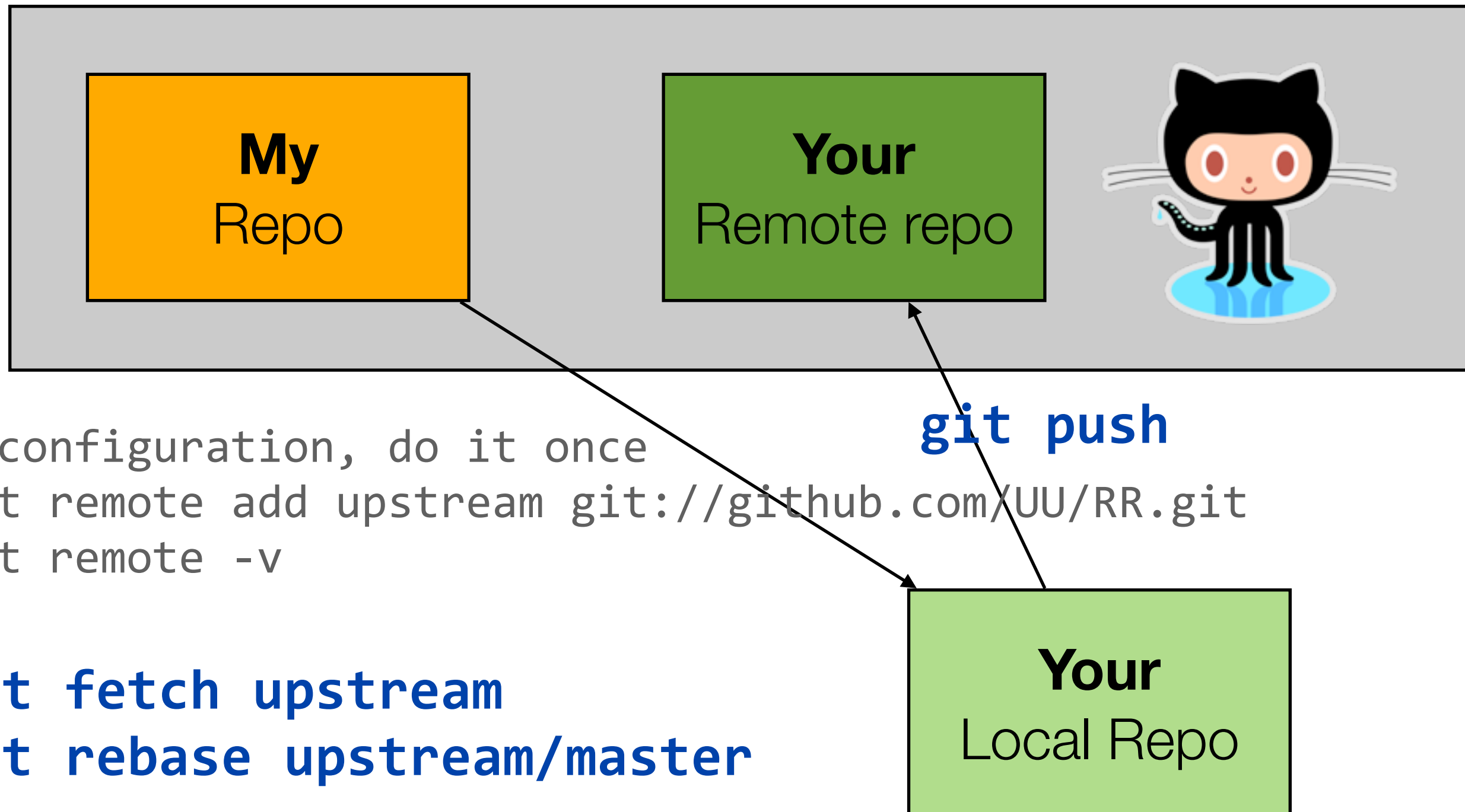


Validate the Setup

- **In each Team, every member should:**
 - make sure that the fork has **cloned** your fork on their machine.
 - **create a file** named firstName-lastName.txt (containing “hello”)
 - **add** this file to the staging area and **commit** the change
- **Now, the fun part: share your work. Everybody should:**
 - start by **fetching** the changes done on the remote (your fork). This will not change your local repository
 - apply the changes to your local repo by doing a **git pull --rebase**; this should not create any conflict
 - send your own commit to your fork, by doing a **git push**
- **Check the commit history on the GitHub web interface AND git with a git log**

```
$ git fetch
$ git pull --rebase
$ git push
```

How Do Will You Get **My** Updates?



Validate the Setup

- **Let me do a change in my original GitHub repo**
- **Connect your local repo to my GitHub repo**

```
$ git remote add upstream git@github.com:wasadigi/Teaching-COMEM-MWS.git  
$ git remote -v  
$ git push
```

- **Fetch and apply my changes**

```
$ git fetch upstream  
$ git rebase upstream/master  
$ git pull --rebase  
$ git commit -m "Applying Olivier's changes"  
$ git push
```