GIT & GITHUB

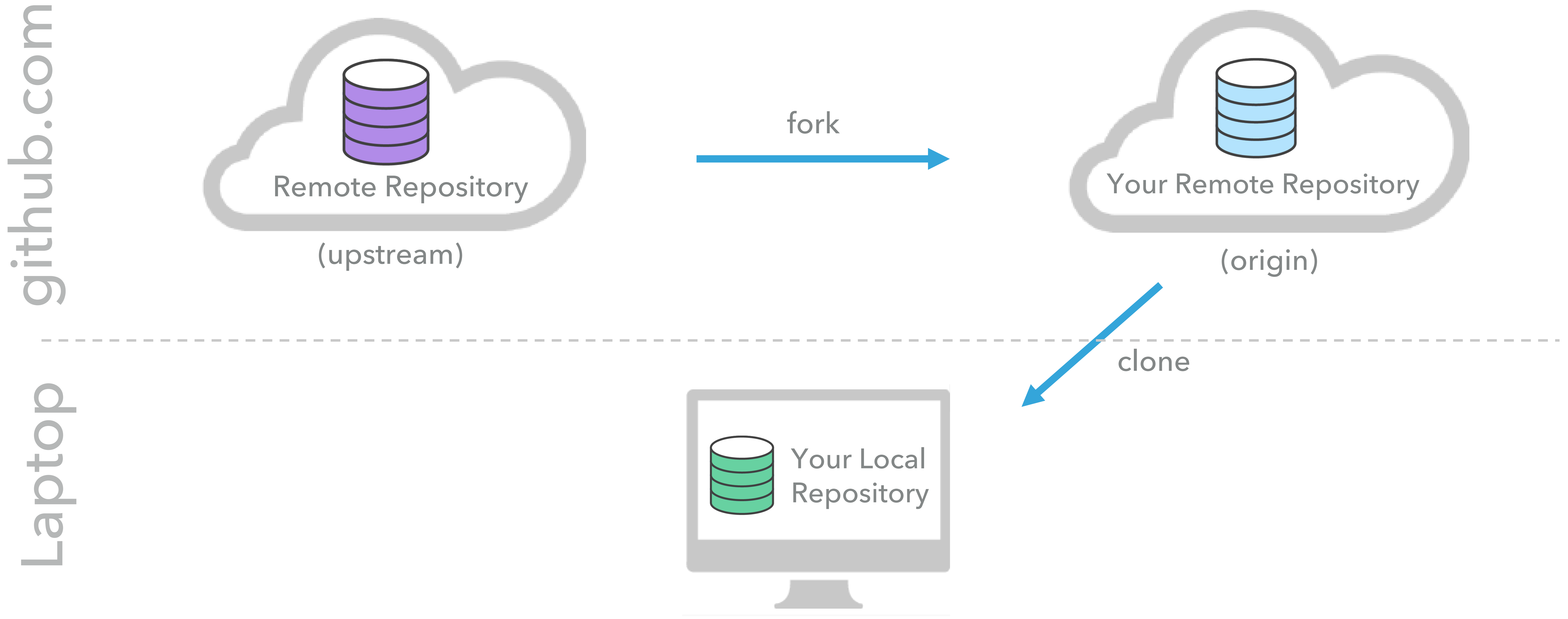
Build software better, together

- Git is a version control system (VCS) that is used for software development. As a distributed revision control system it is aimed at speed, data integrity and support for distributed, non-linear workflows

INTRODUCTION TO GIT

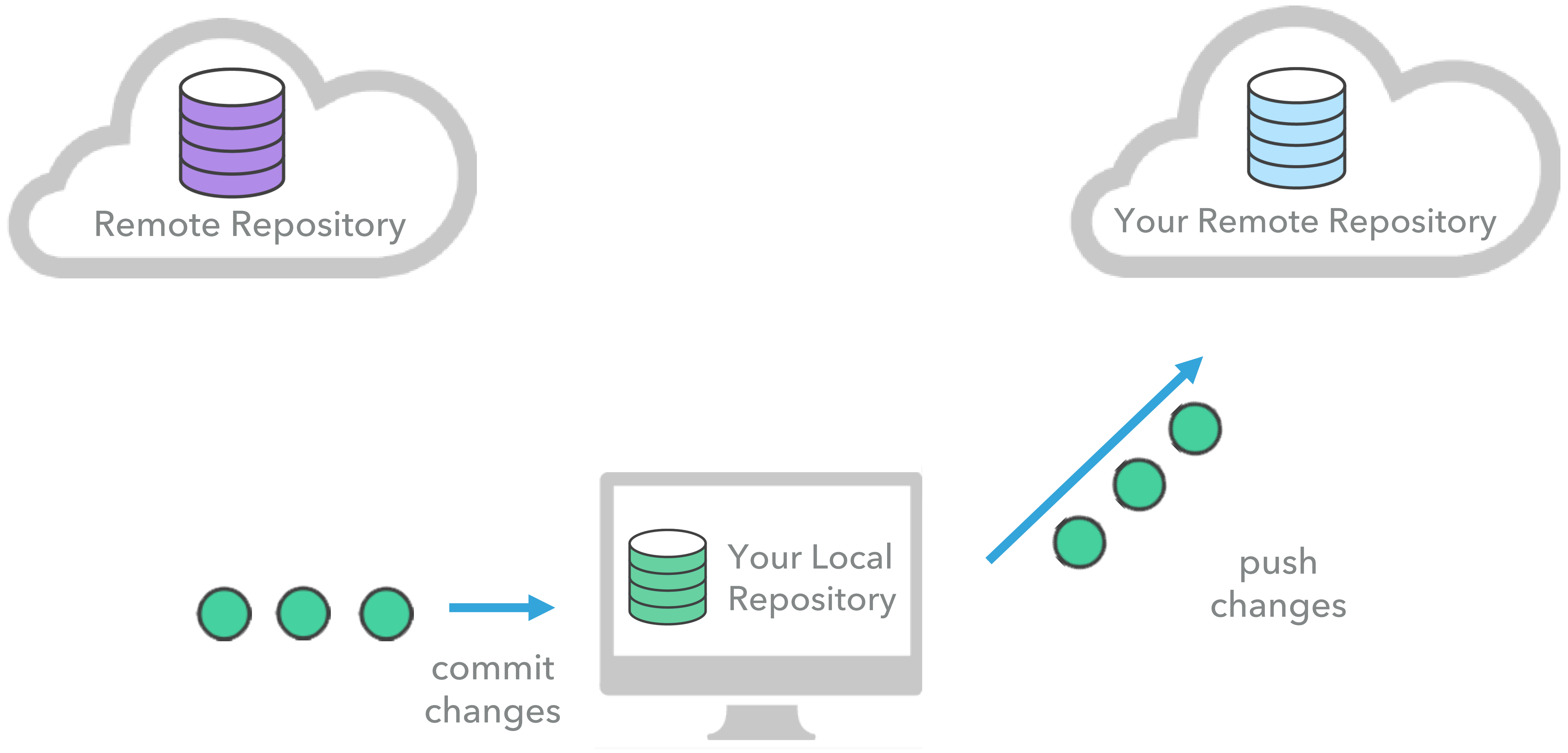
- Git is a **version control tool** that helps programmers to work together in groups and to keep track of their own changes
- It allows every user to have their own local copy of the project, work on it and later merge it to the project again
- If you currently work on a bigger task, it helps you to wrap and save packages that define milestones of your project

GETTING STARTED WITH GIT AND GITHUB



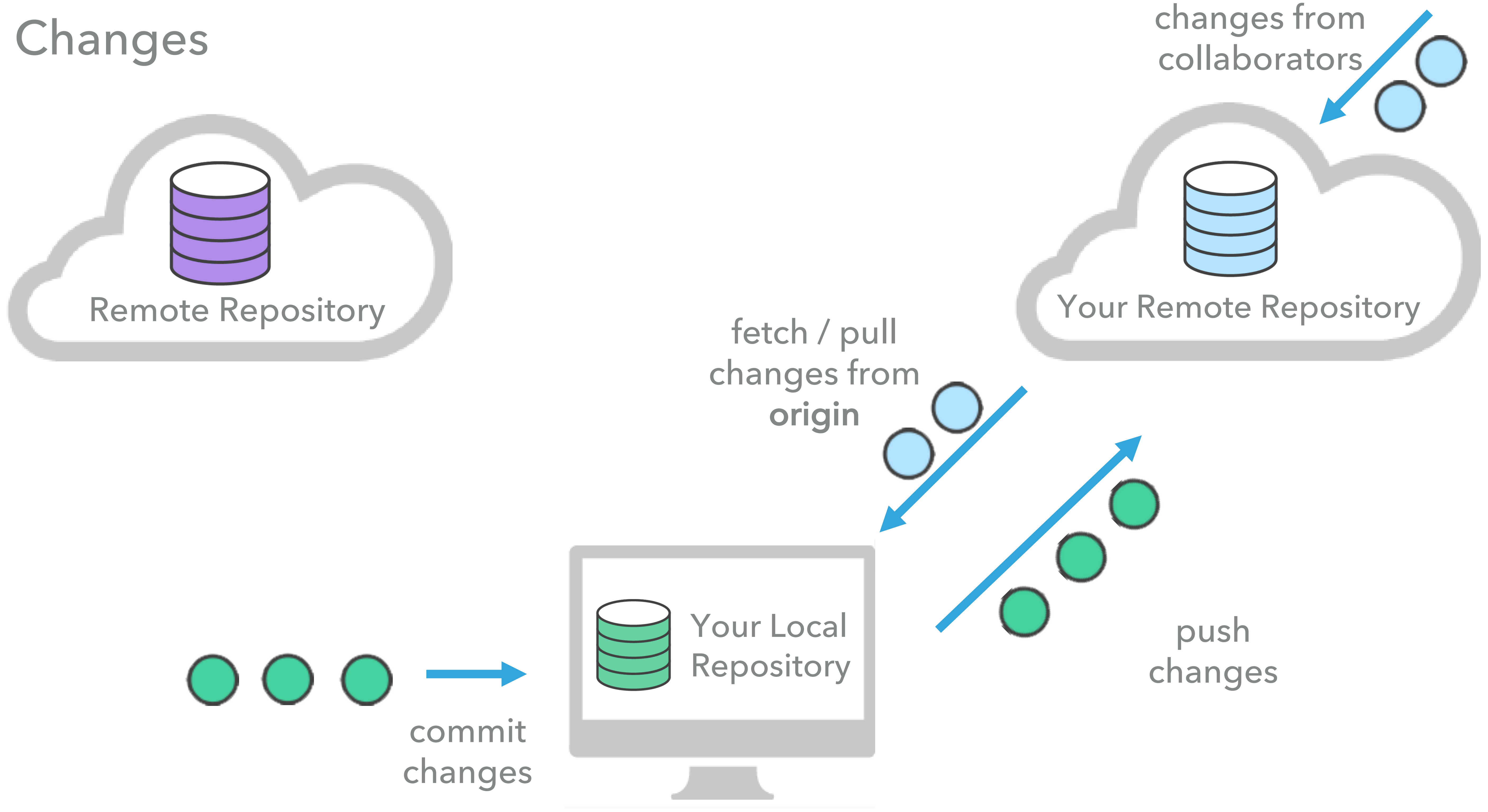
WORKING WITH GIT AND GITHUB

- Adding Changes



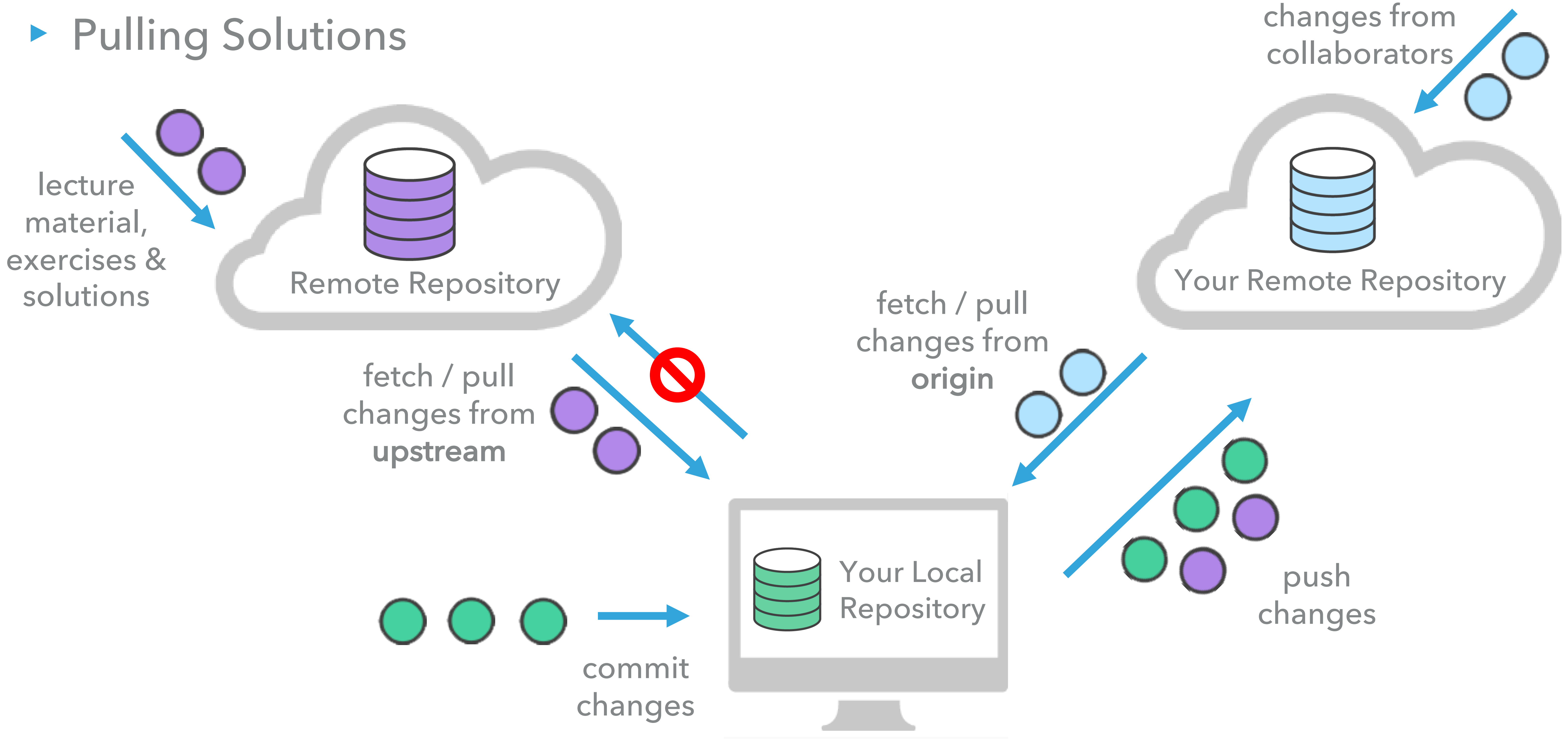
WORKING WITH GIT AND GITHUB

- Pulling Changes



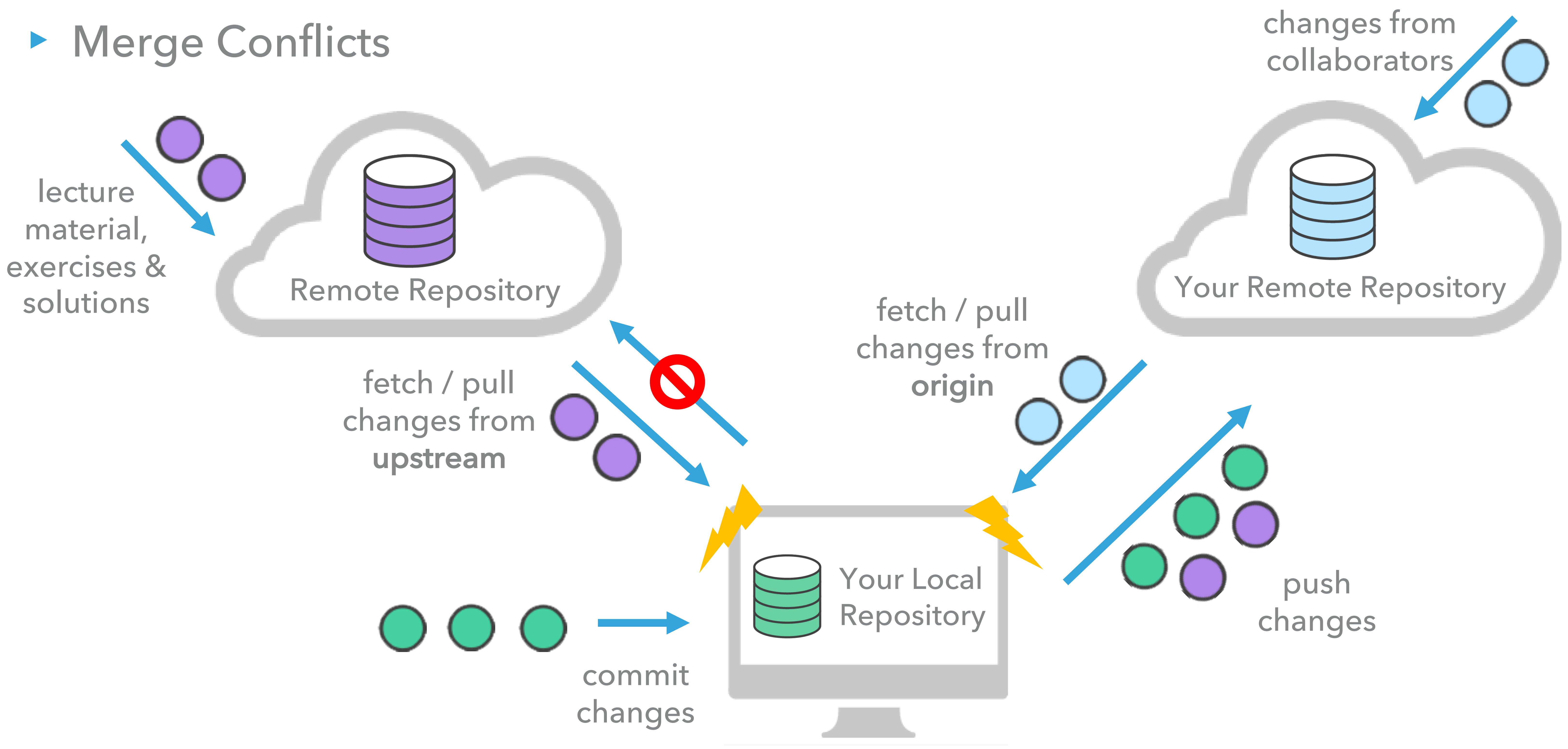
WORKING WITH GIT AND GITHUB

- Pulling Solutions


changes from collaborators
lecture material, exercises & solutions
Remote Repository
Your Remote Repository
fetch / pull changes from **origin**
fetch / pull changes from **upstream**
Your Local Repository
push changes
commit changes

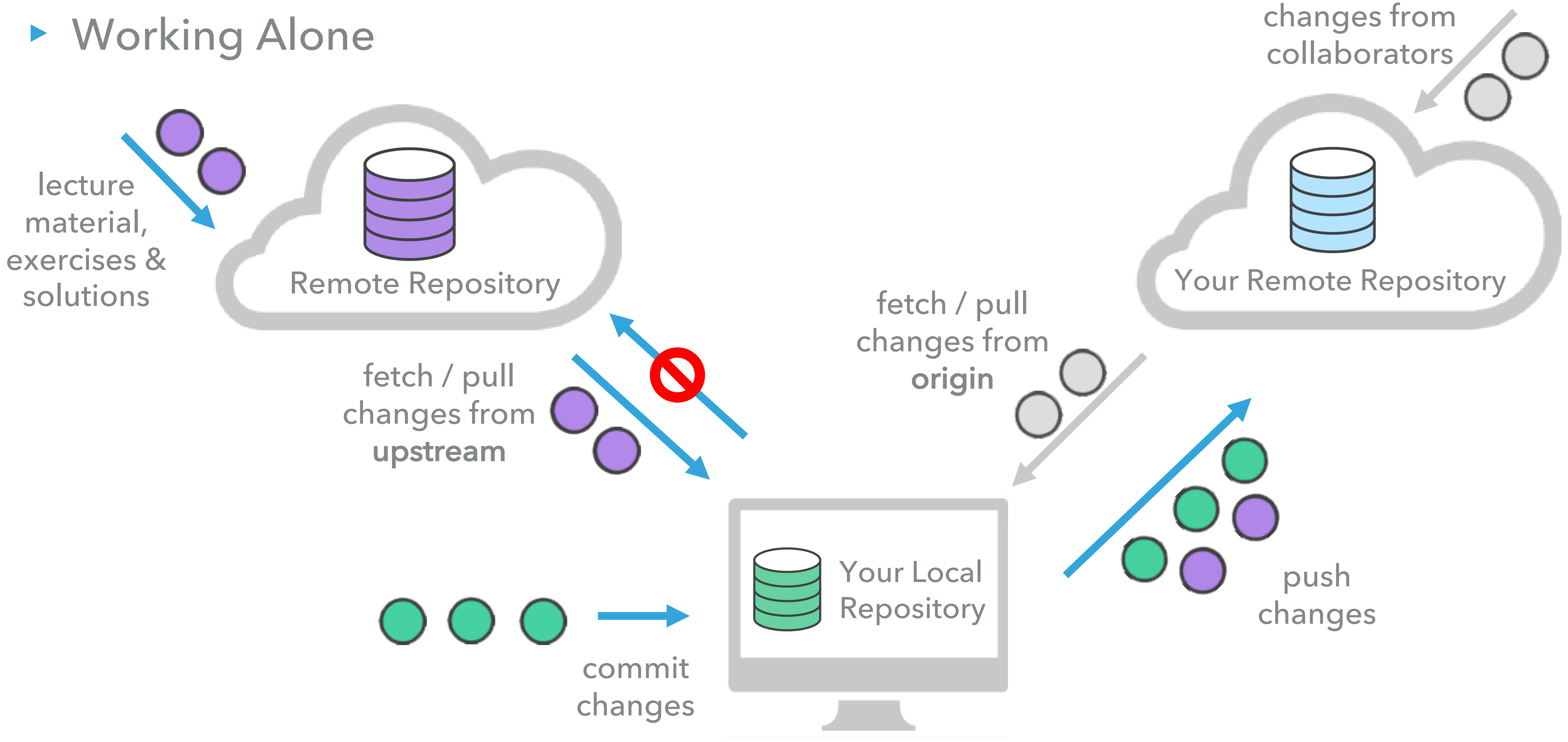
WORKING WITH GIT AND GITHUB

- Merge Conflicts

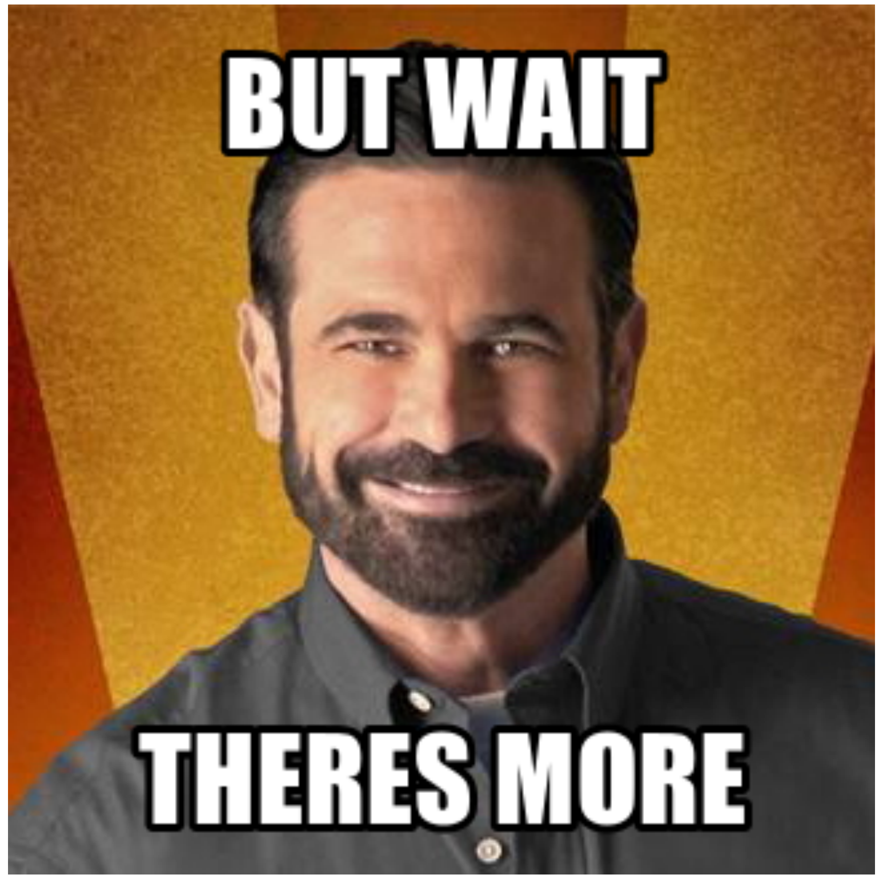

changes from collaborators
lecture material, exercises & solutions
Remote Repository
Your Remote Repository
fetch / pull changes from **origin**
fetch / pull changes from **upstream**
Your Local Repository
push changes
commit changes

WORKING WITH GIT AND GITHUB

- Working Alone


changes from collaborators
lecture material, exercises & solutions
Remote Repository
Your Remote Repository
fetch / pull changes from origin
fetch / pull changes from upstream
Your Local Repository
push changes
commit changes

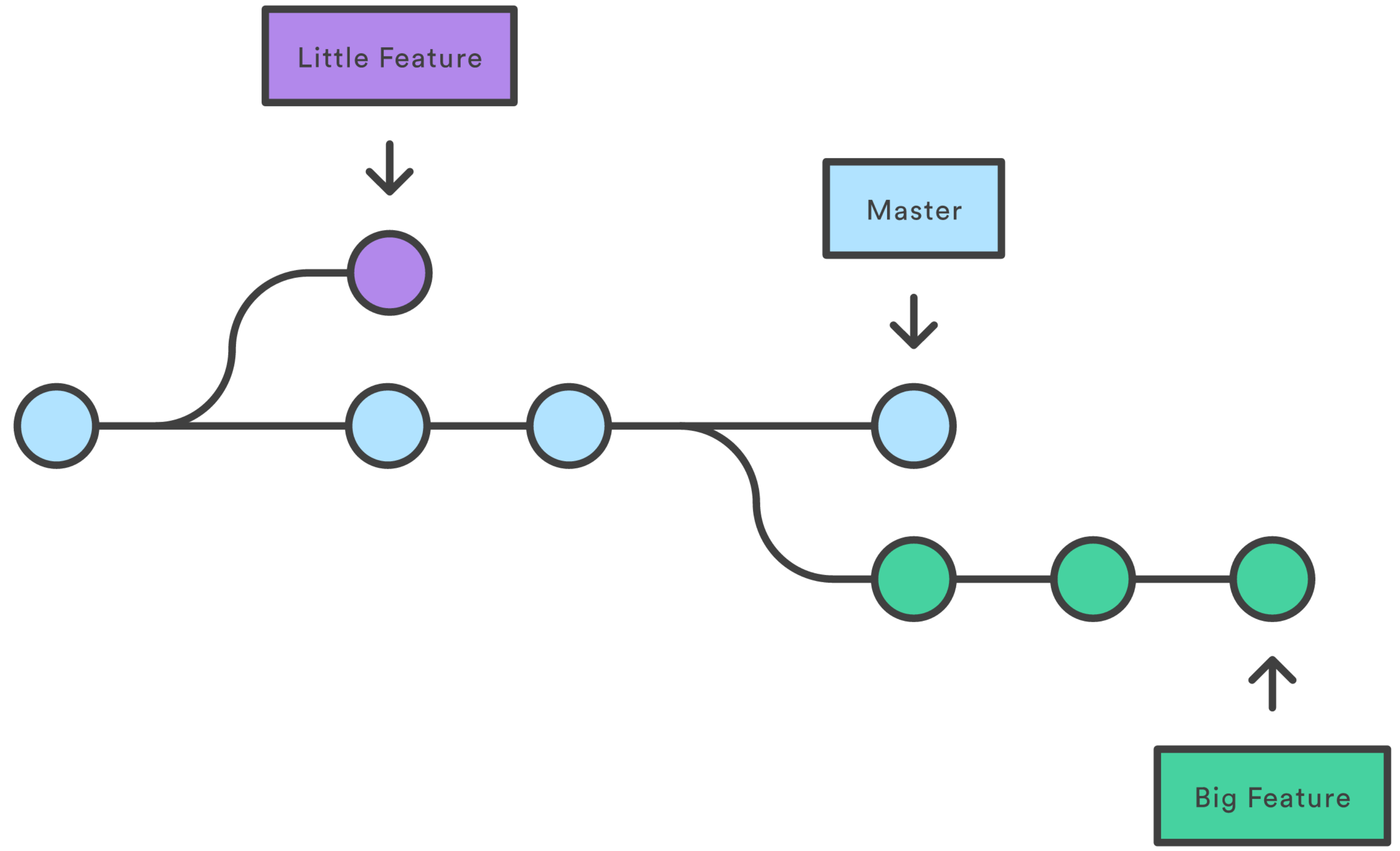
CONFUSED?



GIT BRANCHES

- A different line of development. Is used to develop single features without being affected by code changes of collaborating programmers.
- A single repository can contain multiple branches. The default branch in git is called *master*.

- Git Workflow
  - Once you start to work on a new feature you would create a new branch of the current master branch
  - After you implemented your feature you would open a **pull request**
  - Once someone in charge of the master branch looked at the code in your feature branch he can approve your pull request and your branch can be merged back into the master branch.



Git explained: http://juristr.com/blog/2013/04/git-explained/
Atlassian Git Tutorial: https://www.atlassian.com/git/tutorials/what-is-git
A successful branching model: http://nvie.com/posts/a-successful-git-branching-model/

GIT – TERMINOLOGY

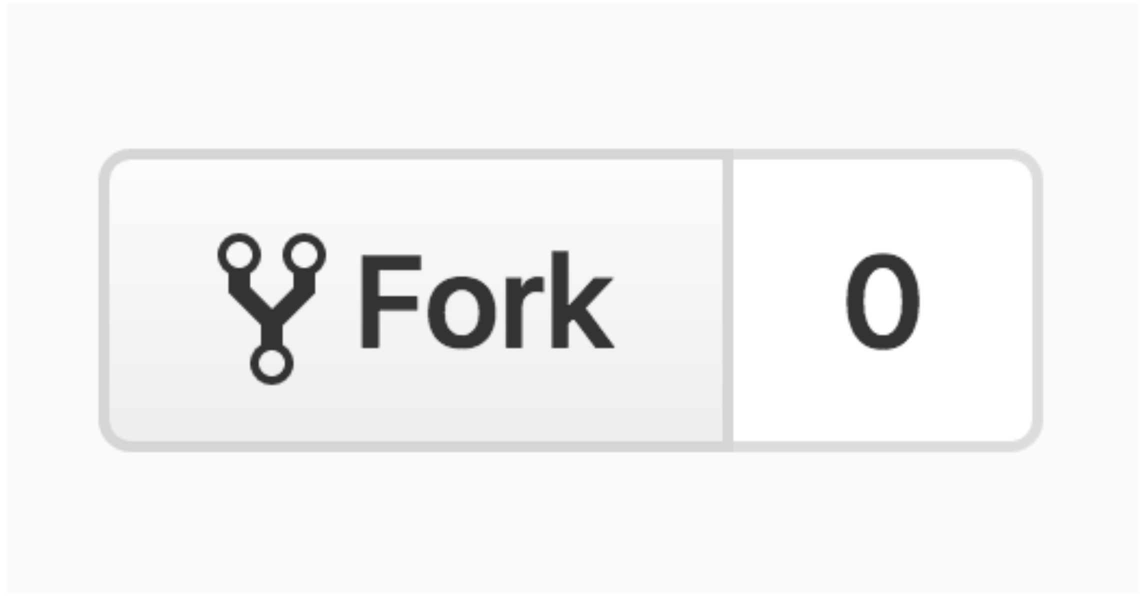
- **Repository**
  - This is your object database where your history and configuration is stored. May contain several branches.
  - Remote Repository: The repository in ‘the cloud’ where all collaborators have access
  - Local Repository: Your local working copy of the repository
- **Branch**
  - A different line of development. Is used to develop single features without being affected by code changes of collaborating programmers.
  - A single repository can contain multiple branches. The default branch in git is called *master*.
- **Commit**
  - A single point in the Git history; the entire history of a project is represented as a set of interrelated commits.

GIT – TERMINOLOGY

- to **fetch**
  - Fetching a branch means to get the branch from a remote repository to find out which objects are changed or missing.
- to **merge**
  - To bring the contents of another branch into the current branch.
  - Merging is performed by an automatic process that identifies changes made since the branches diverged, and then applies all those changes together.
  - In cases where changes conflict, manual intervention may be required to complete the merge.
- to **pull**
  - Pulling a branch means to fetch it and merge it.
- to **commit**
  - The action of storing a new snapshot of the project's state in the Git history. (“Saving your changes in the local repository”)
- To **push**
  - Taking your local changes and pushing to another repository (e.g. your remote repository)

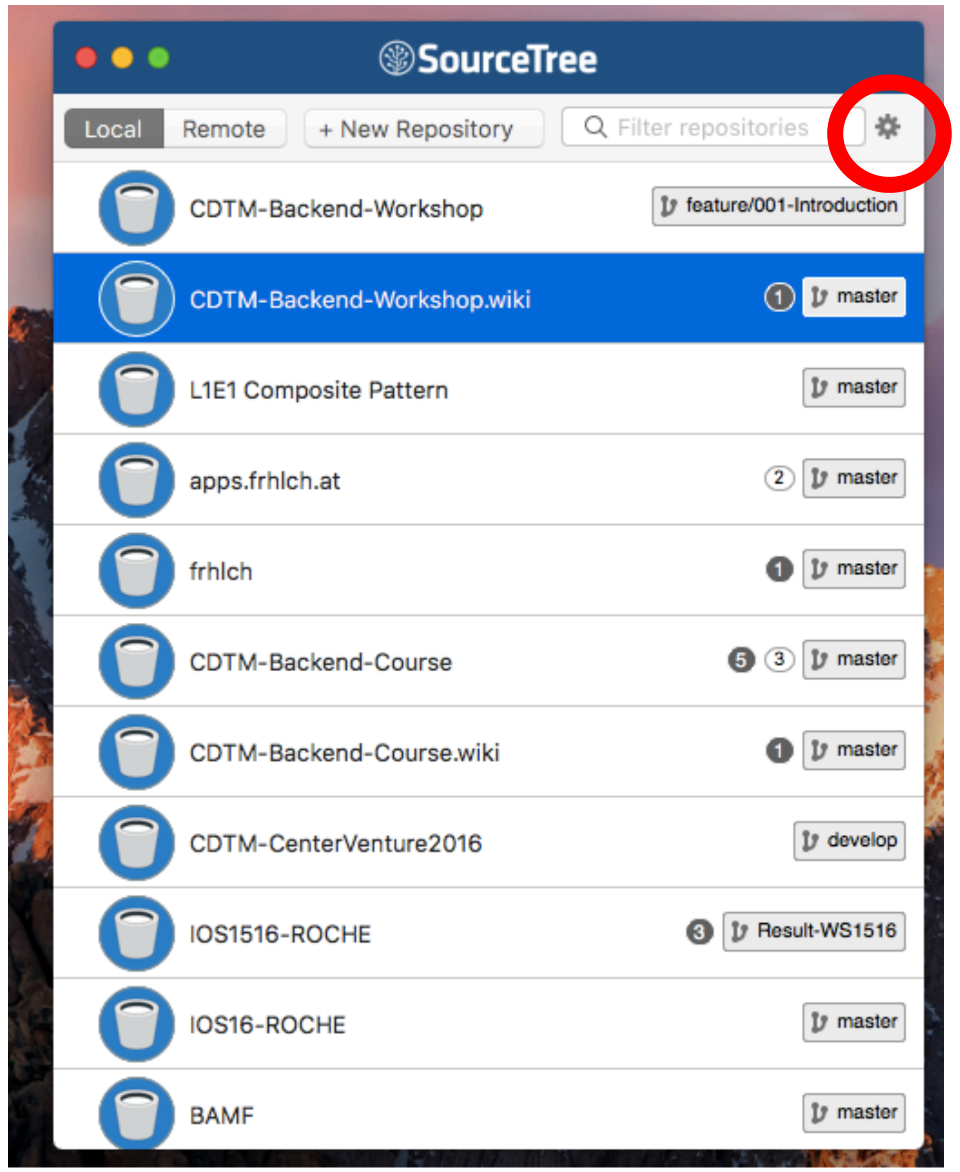
LET'S GET WORKING

- Fork the repository on Github
  - Creating a "fork" is producing a **personal copy** of someone else's project. Forks act as a sort of bridge between the original repository and your personal copy.
  - You can submit *Pull Requests* to help make other people's projects better by offering your changes up to the original project. Forking is at the core of social coding at GitHub.
  - During this workshop you will only submit code to your "forked" repository. We will make the lecture material and solutions available in the original repository and you can pull the changes into your repository.

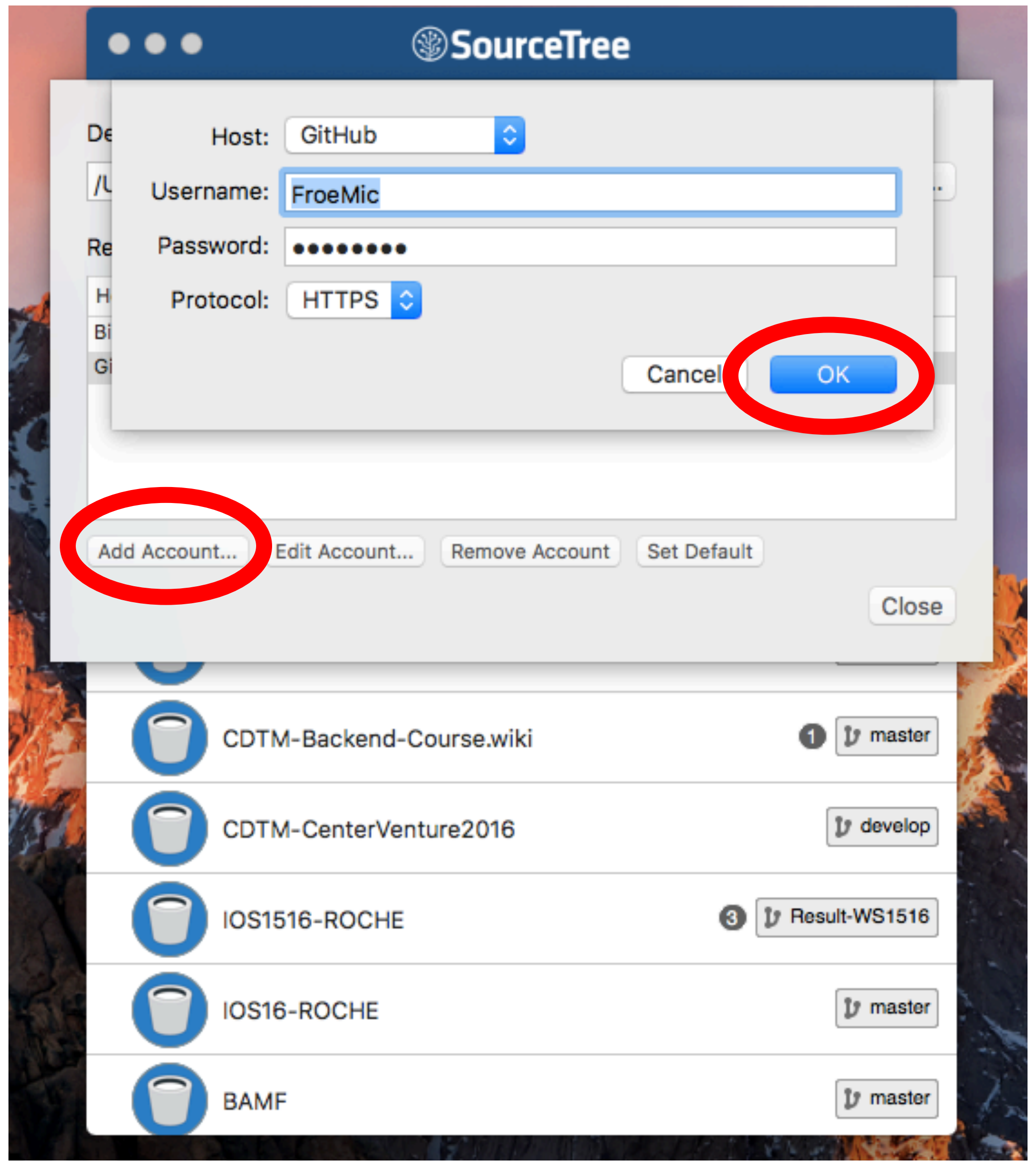


SET UP SOURCETREE

- Add Github Account to SourceTree



CDTM-Backend-Workshop	feature/001-Introduction
CDTM-Backend-Workshop.wiki	1 master
L1E1 Composite Pattern	master
apps.frhlch.at	2 master
frhlch	1 master
CDTM-Backend-Course	5 3 master
CDTM-Backend-Course.wiki	1 master
CDTM-CenterVenture2016	develop
IOS1516-ROCHE	3 Result-WS1516
IOS16-ROCHE	master
BAMF	master

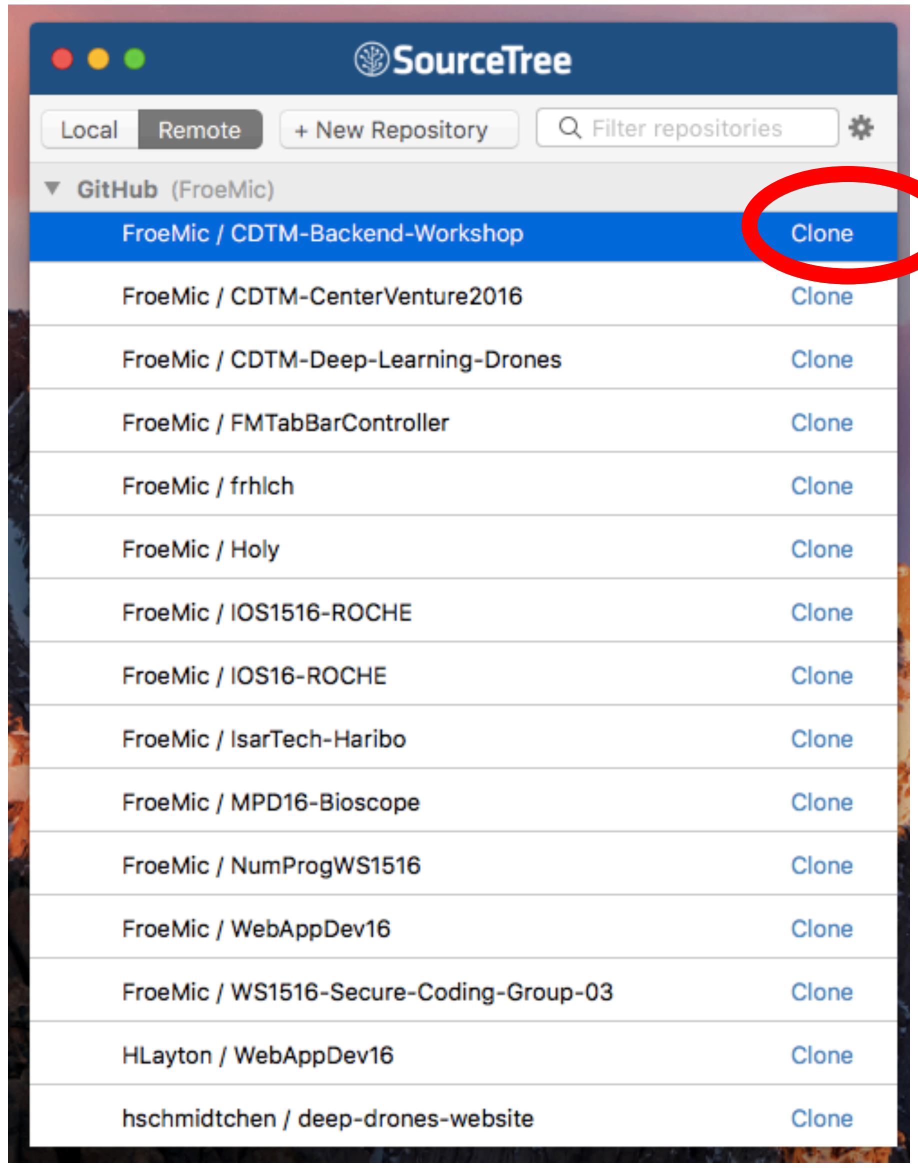


Host: GitHub
Username: FroeMic
Password: ••••••••
Protocol: HTTPS

CDTM-Backend-Course.wiki	1 master
CDTM-CenterVenture2016	develop
IOS1516-ROCHE	3 Result-WS1516
IOS16-ROCHE	master
BAMF	master

SET UP SOURCETREE

- Clone Repository (create local version)

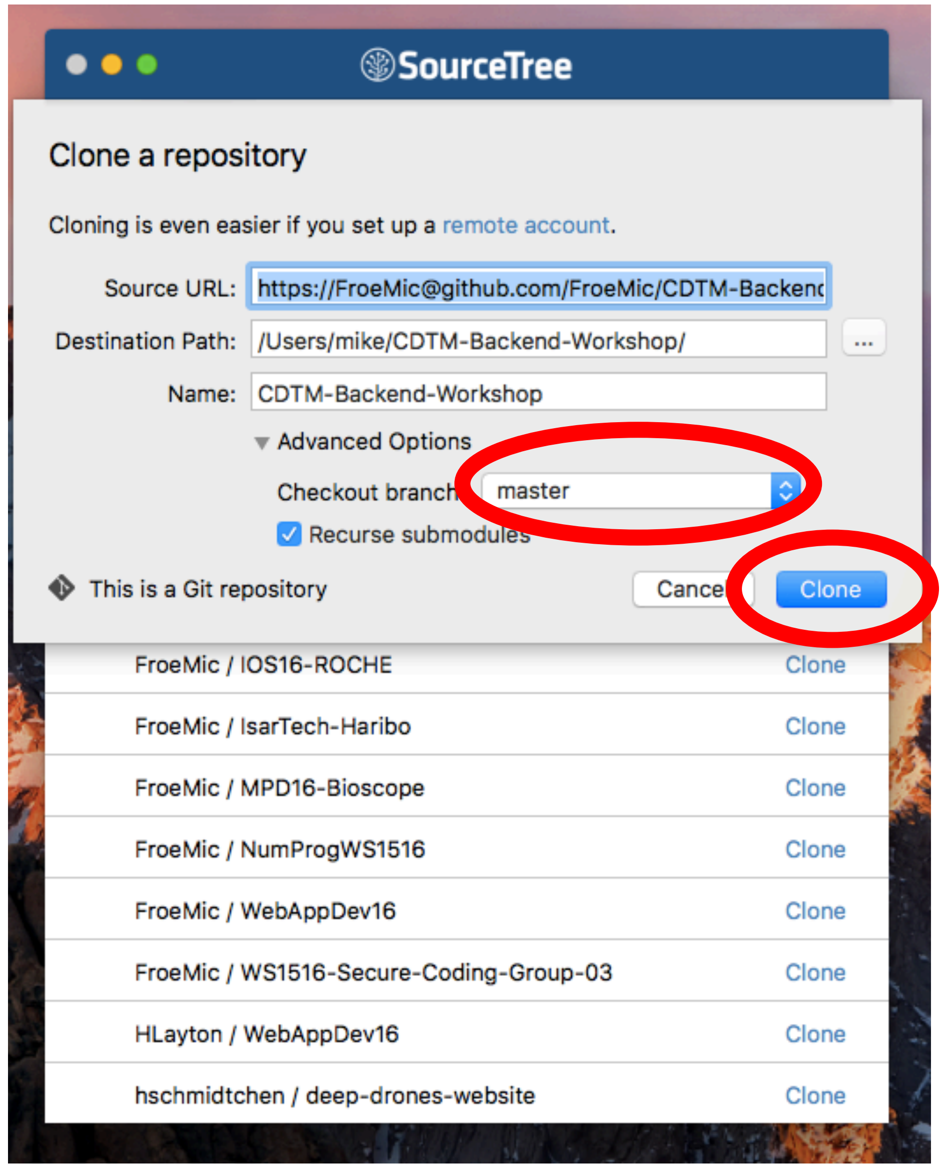


SourceTree

Local | Remote | + New Repository | Filter repositories

GitHub (FroeMic)

FroeMic / CDTM-Backend-Workshop	Clone
FroeMic / CDTM-CenterVenture2016	Clone
FroeMic / CDTM-Deep-Learning-Drones	Clone
FroeMic / FMTabBarController	Clone
FroeMic / frhlch	Clone
FroeMic / Holy	Clone
FroeMic / IOS1516-ROCHE	Clone
FroeMic / IOS16-ROCHE	Clone
FroeMic / IsarTech-Haribo	Clone
FroeMic / MPD16-Bioscope	Clone
FroeMic / NumProgWS1516	Clone
FroeMic / WebAppDev16	Clone
FroeMic / WS1516-Secure-Coding-Group-03	Clone
HLayton / WebAppDev16	Clone
hschmidtchen / deep-drones-website	Clone



SourceTree

Clone a repository

Cloning is even easier if you set up a remote account.

Source URL: https://FroeMic@github.com/FroeMic/CDTM-Backenc
Destination Path: /Users/mike/CDTM-Backend-Workshop/
Name: CDTM-Backend-Workshop
Advanced Options
Checkout branch: master
☑ Recurse submodules

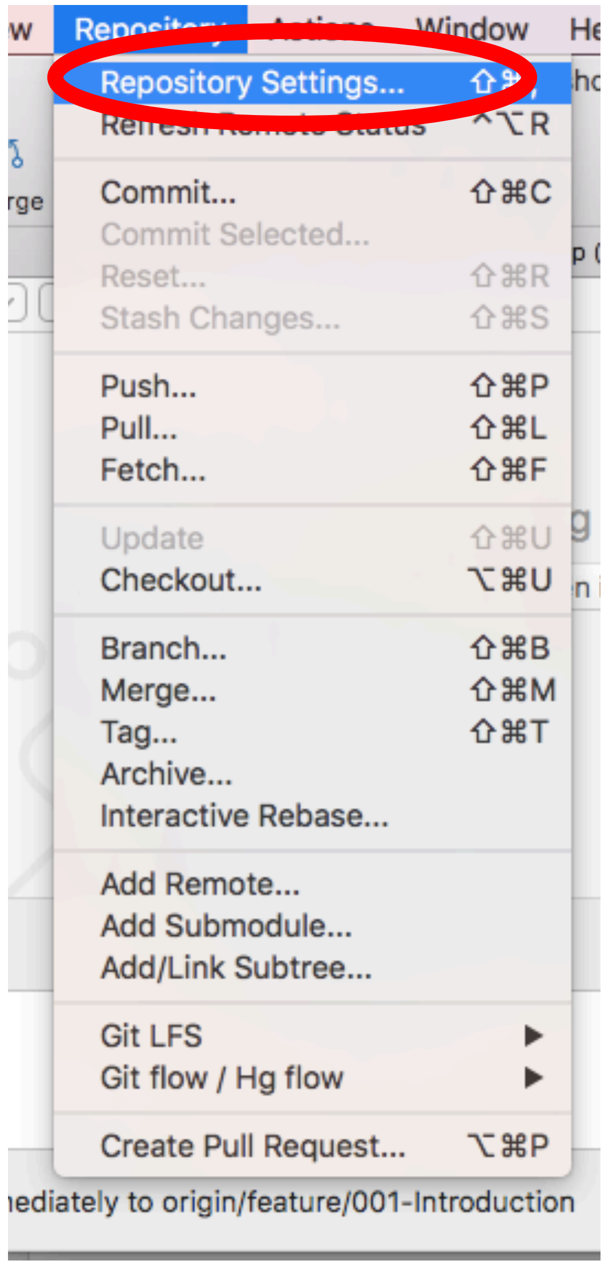
This is a Git repository

Cancel | Clone

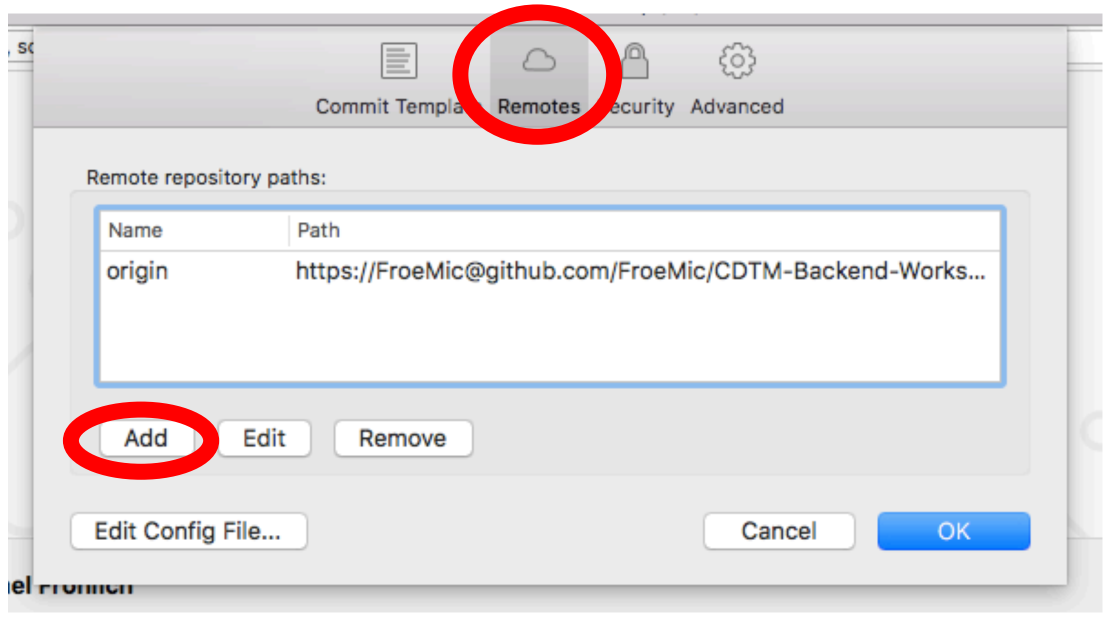
FroeMic / IOS16-ROCHE	Clone
FroeMic / IsarTech-Haribo	Clone
FroeMic / MPD16-Bioscope	Clone
FroeMic / NumProgWS1516	Clone
FroeMic / WebAppDev16	Clone
FroeMic / WS1516-Secure-Coding-Group-03	Clone
HLayton / WebAppDev16	Clone
hschmidtchen / deep-drones-website	Clone

SET UP SOURCETREE

- Add UPSTREAM Repository

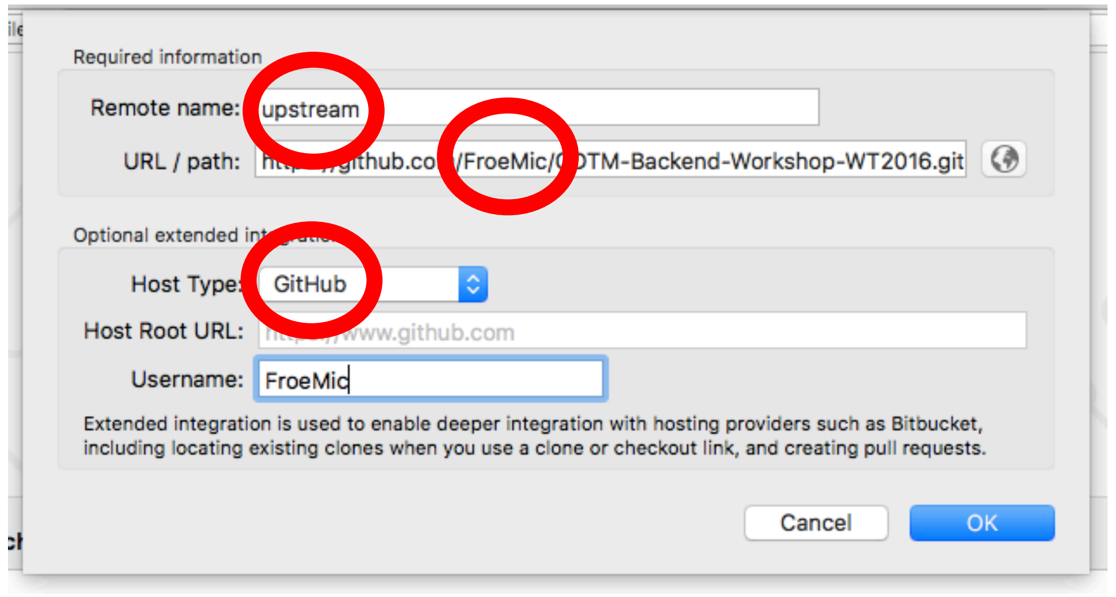


ediately to origin/feature/001-Introduction



Remote repository paths:

Name	Path
origin	https://FroeMic@github.com/FroeMic/CDTM-Backend-Works...



Required information

Remote name: upstream

URL / path: https://github.com/FroeMic/CDTM-Backend-Workshop-WT2016.git

Optional extended integration

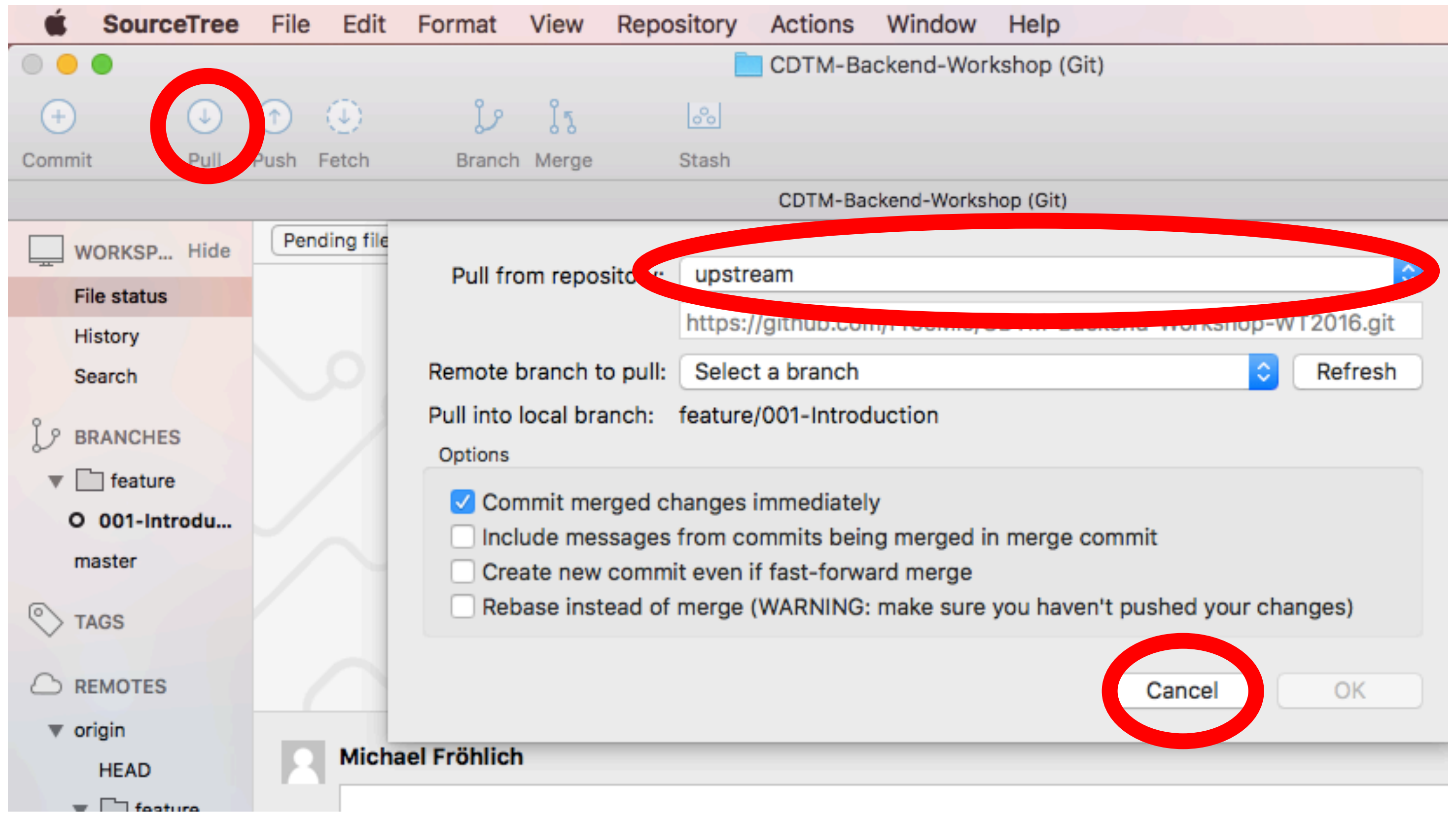
Host Type: GitHub

Host Root URL: https://www.github.com

Username: FroeMic

Extended integration is used to enable deeper integration with hosting providers such as Bitbucket, including locating existing clones when you use a clone or checkout link, and creating pull requests.

- 1. Go to 'Repository Settings'
- 2. Select 'Remotes' and press 'Add'
- 3. Name it 'upstream' and add the repository link of the repository you forked from. Select 'Github' and enter your username. Press OK.
- 4. Press 'Pull' and check whether you can select 'upstream'. If so, press cancel.



Pull from repository: upstream

https://github.com/FroeMic/CDTM-Backend-Workshop-WT2016.git

Remote branch to pull: Select a branch

Pull into local branch: feature/001-Introduction

Options

- [✓] Commit merged changes immediately
- [ ] Include messages from commits being merged in merge commit
- [ ] Create new commit even if fast-forward merge
- [ ] Rebase instead of merge (WARNING: make sure you haven't pushed your changes)

SET UP SOURCETREE

- Committing Changes to your **local repository**
  - Every time you want to 'save' your changes, press *Commit* and add all the files you want to *save*.
  - Add a short but descriptive *Commit-Message* so you know what changed later on.
  - 'Press Commit' -> Your changes are now saved in your local repository.
- Pushing Commit to Github
  - To transmit your commits to your **remote repository** on github.com you have to *push* them.
  - This will, however, fail if the remote repository changed since you last *pulled* from it.
  - To resolve this, you have to *pull* and *merge* the changes (and resolve arising merge-conflicts) before you can *push* again.
- Pulling Changes from Upstream
  - Pulling from 'Upstream' means that you *pull* and *merge* changes from the original repository into the local version of your forked repository.
  - To transmit the changes to your remote repository, you have to *push* them

Repository Structure

The repository structure will look as follows. Please note that there are certain guideline to follow, in order to avoid problems throughout the workshop.

1. Only modify files which are inside the `/src` folder
2. Do not modify anything within the `/src/server/static` folder

```
.
+-- README.md
+-- slides/ ⚡                       (lecture slides)
+-- src/                            (YOUR CODE)
|   +-- exercises/                  (single exercises will go here)
    |   +-- exerciseXYZ/
    |   +-- .../
|   +-- server/                     (your server implementation)
    |   +-- server.py
    |   +-- static/ ⚡               (frontend -> don't modify)
    |   +-- ...
+-- solutions/ ⚡                    (solutions will be publish here)
|   +-- server/                     (solutions for each step of the server)
    |   +-- server-01-description
    |   +-- ...
```