GENERAL ASSEMBLY

# MOBILE DEVELOPMENT

## INTRO TO INTERFACE BUILDER

William Martin
Head of Product, Floored

Angel X. Moreno
EIR, Developer

# LEARNING OBJECTIVES

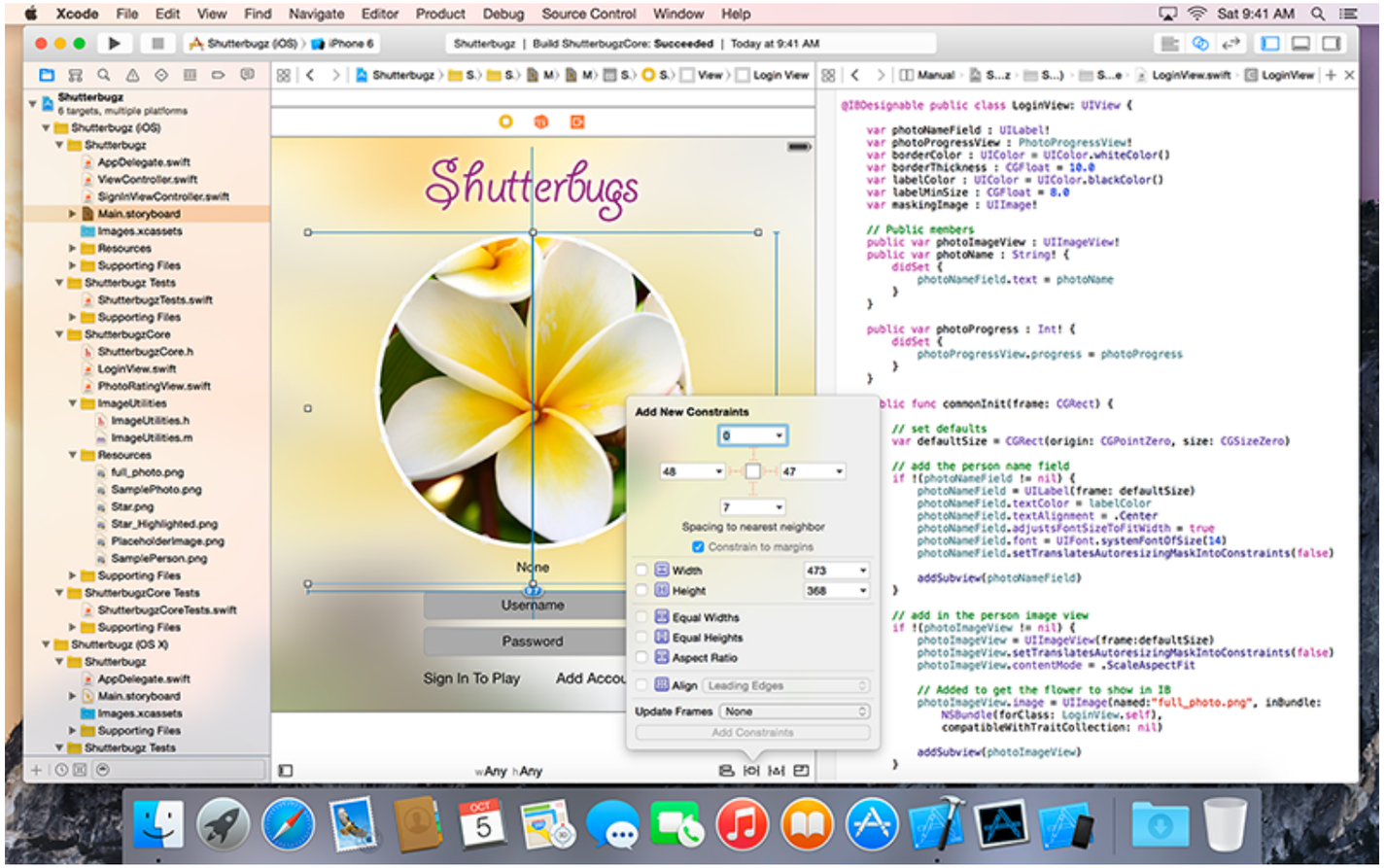
- Review and share your apps. Address any questions
- Review concepts from last class.
- Learn to draw and read system diagrams.
- Learn the logistics of our course's homework submission process with Github.

# REVIEW YOUR APPS

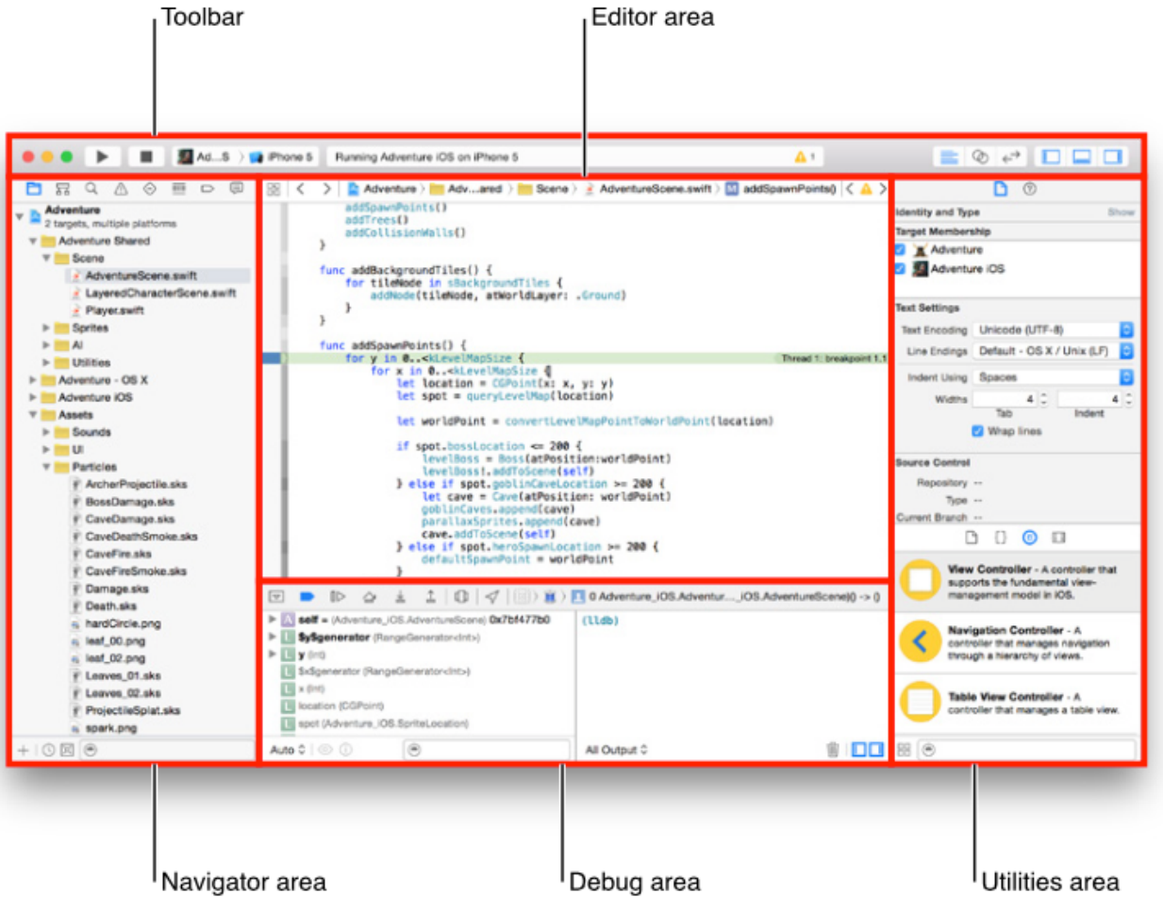
# DEV WORKFLOW AND STORYBOARDS

## GETTING STARTED

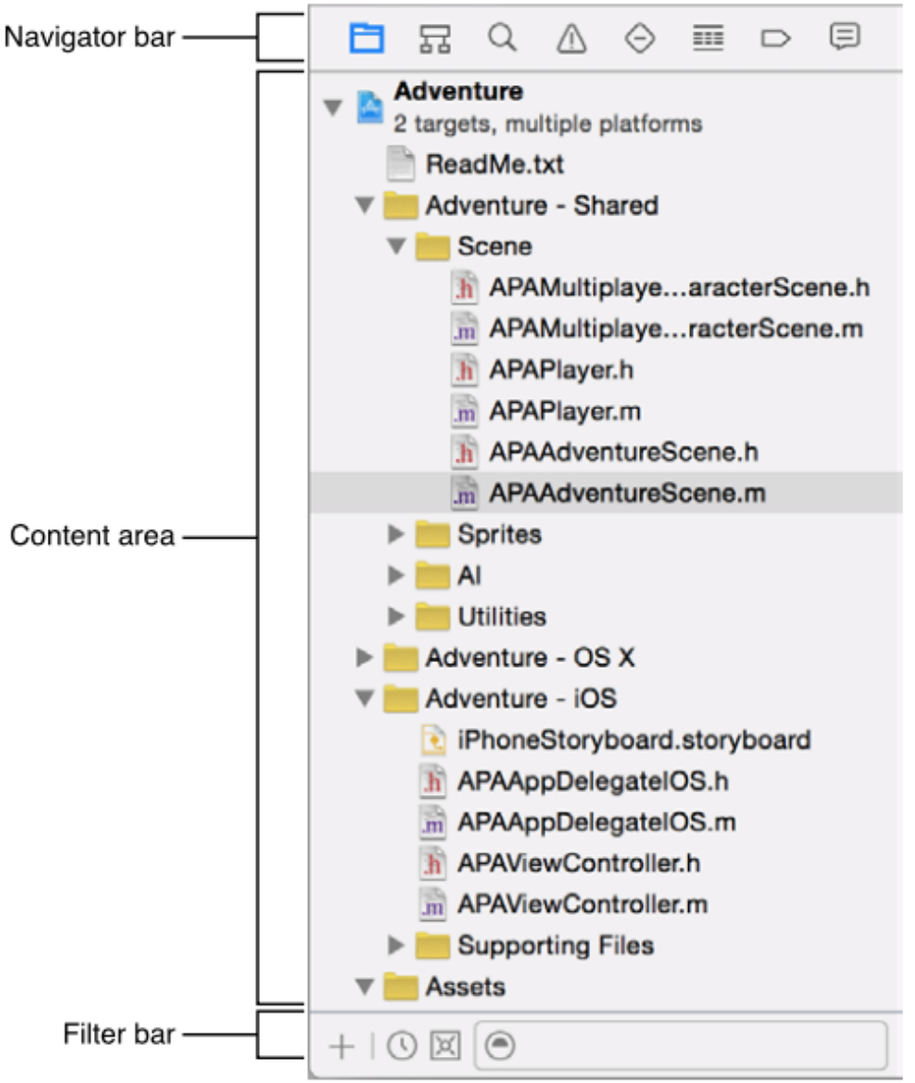
# XCODE REVIEW



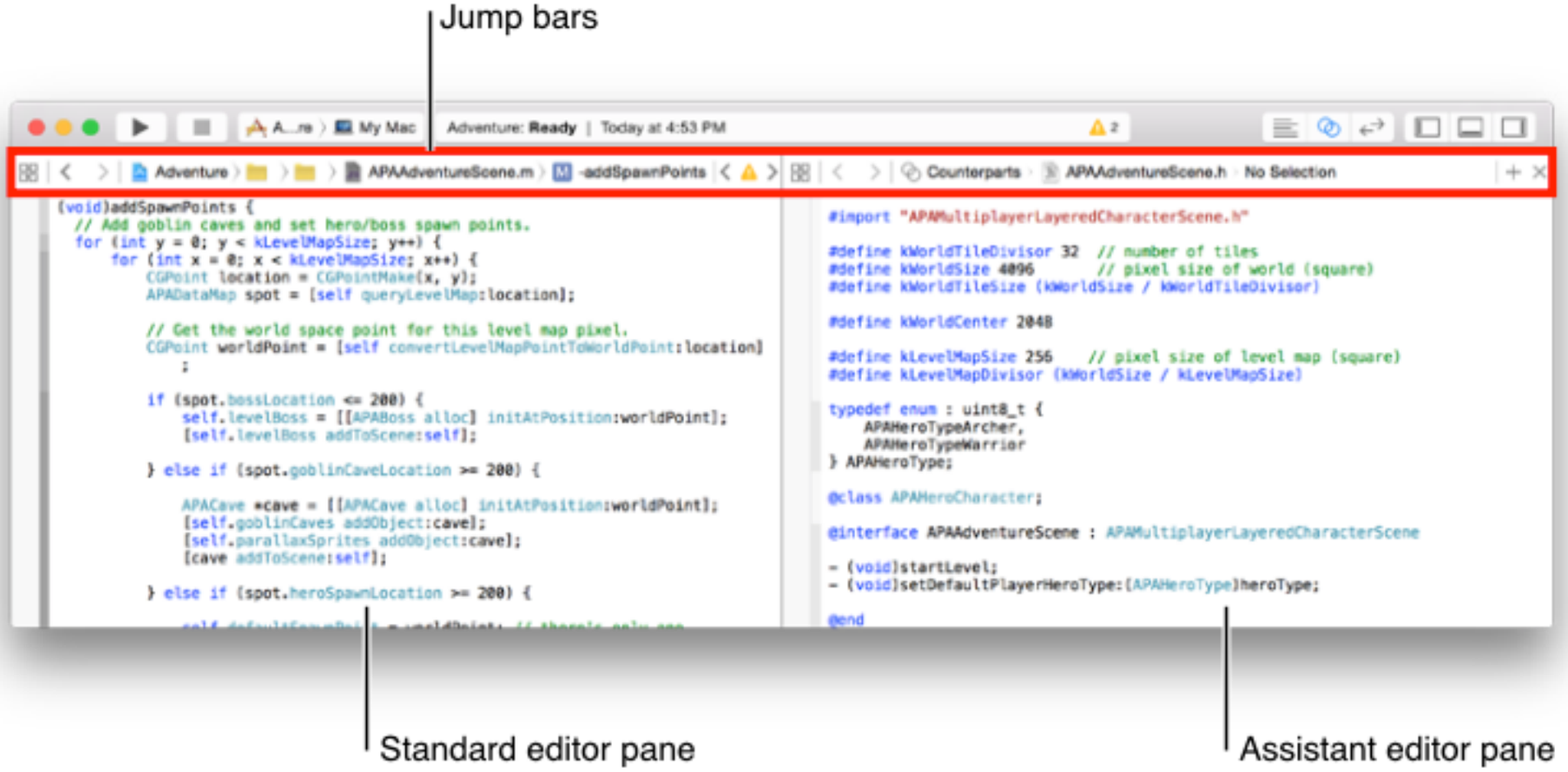
# NAVIGATING XCODE



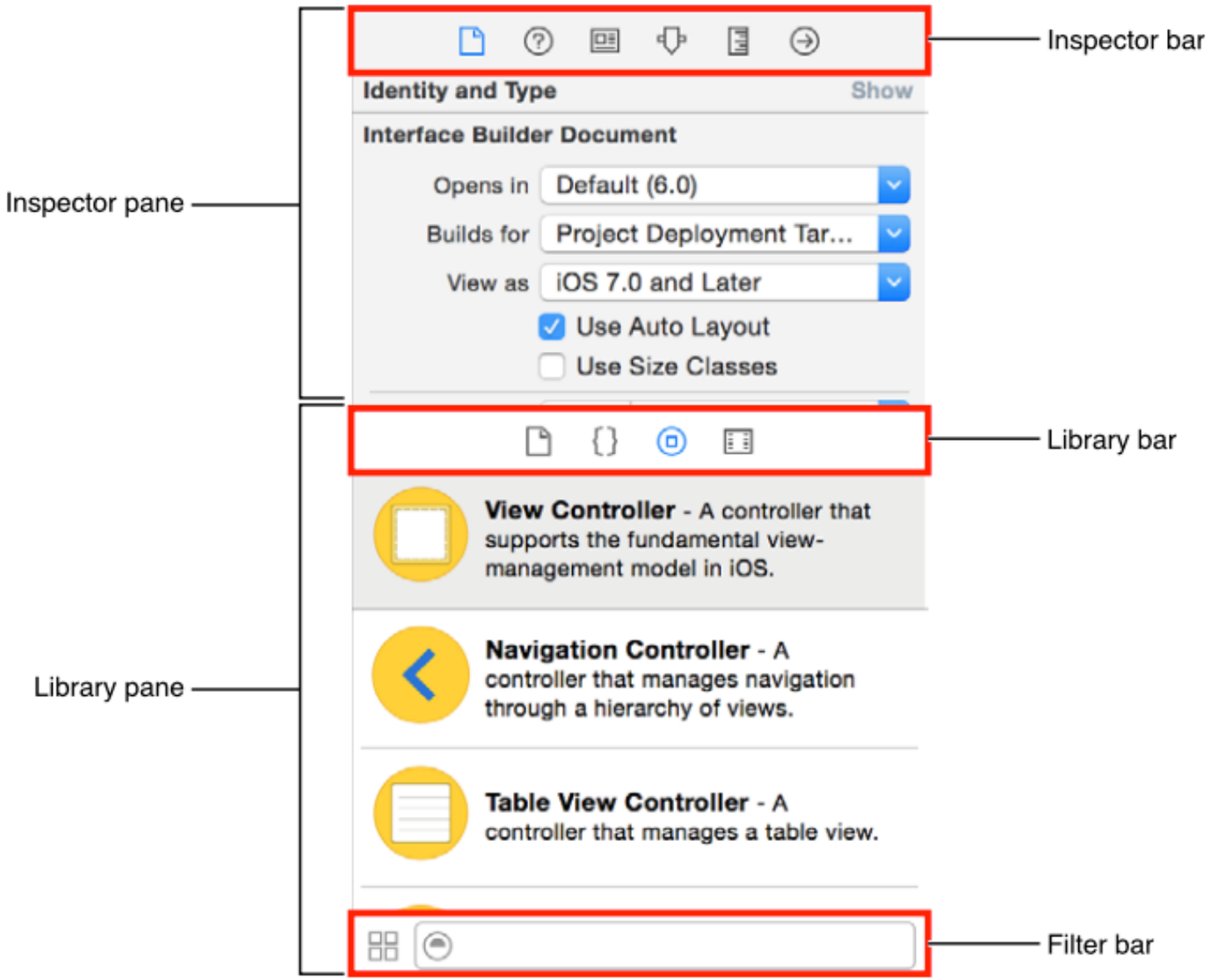
# NAVIGATOR AREA



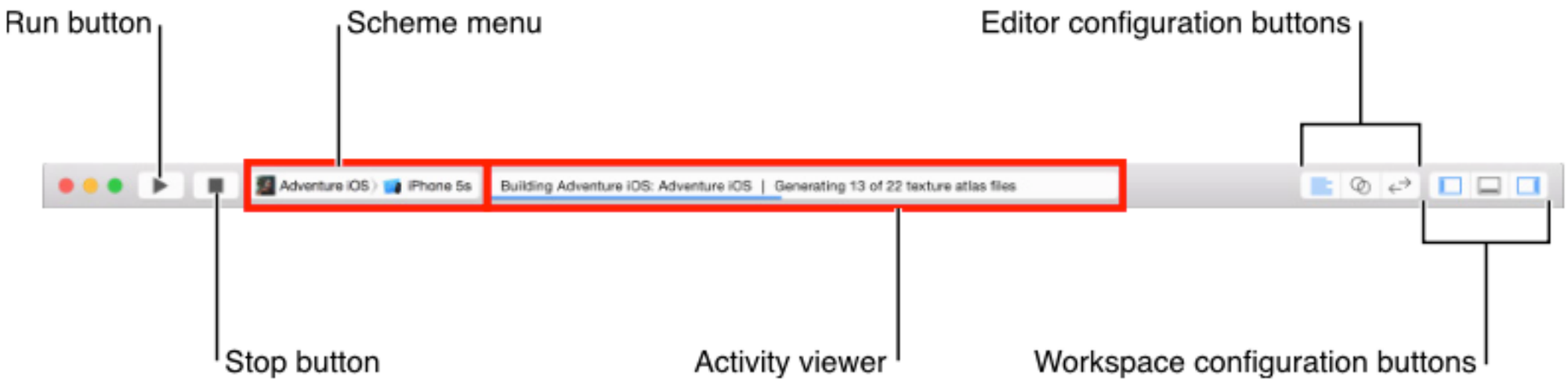
# JUMP BAR AND EDITOR PANES



# UTILITIES AREA



# WORKSPACE TOOLBAR



# DEV WORKFLOW

- Run Xcode
- Create new project
- Add user interface elements to project
- Change user interface element properties
- Discuss the different project templates

# TO XCODE!

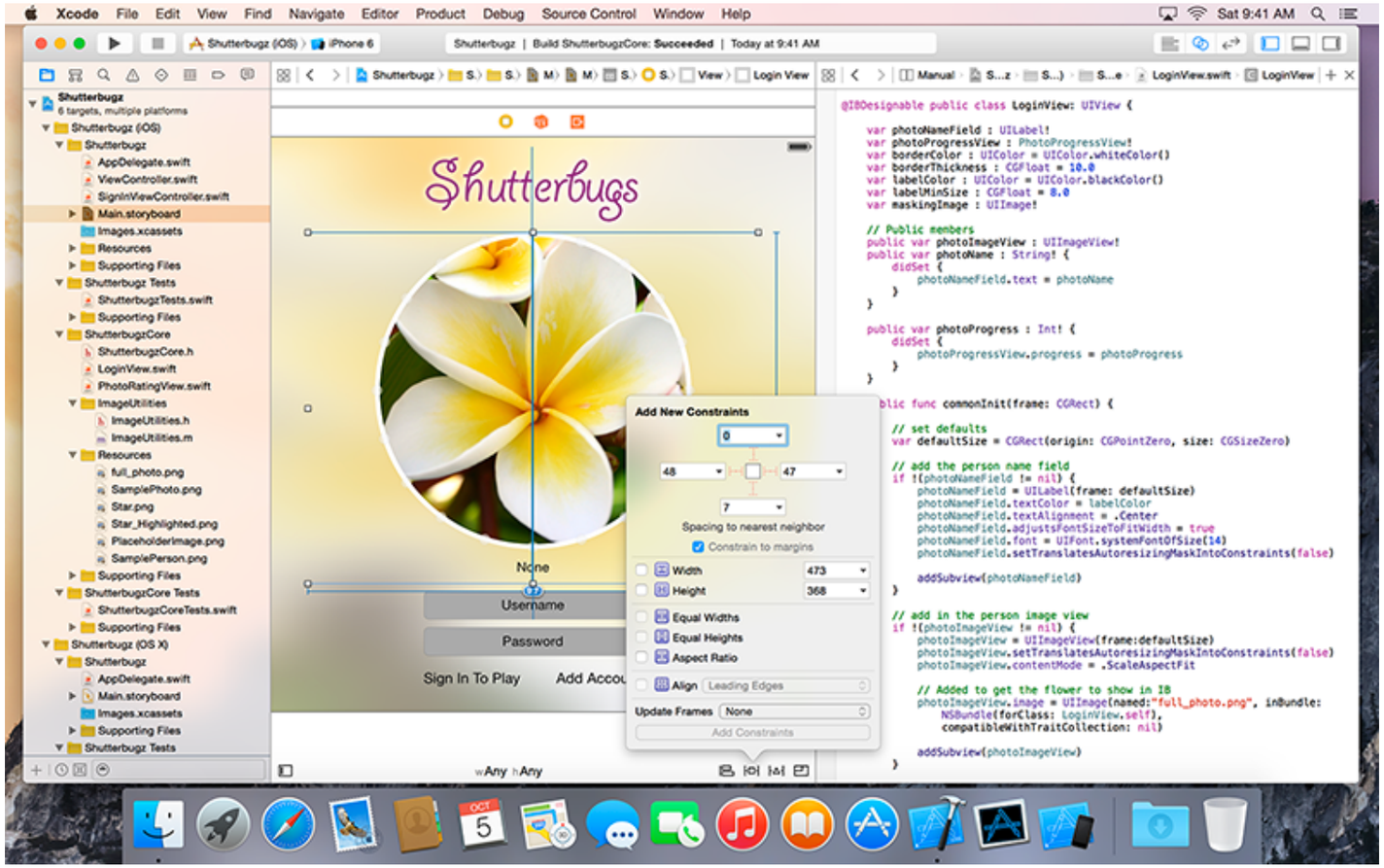
# REVIEW: DEV WORKFLOW

# DEV WORKFLOW OVERVIEW

- Run Xcode
- Create new project
- Briefly discuss the different project templates
- Add user interface elements to project
- Change user interface element properties
- Build / run the app
- Iterate
- Post to Github when done

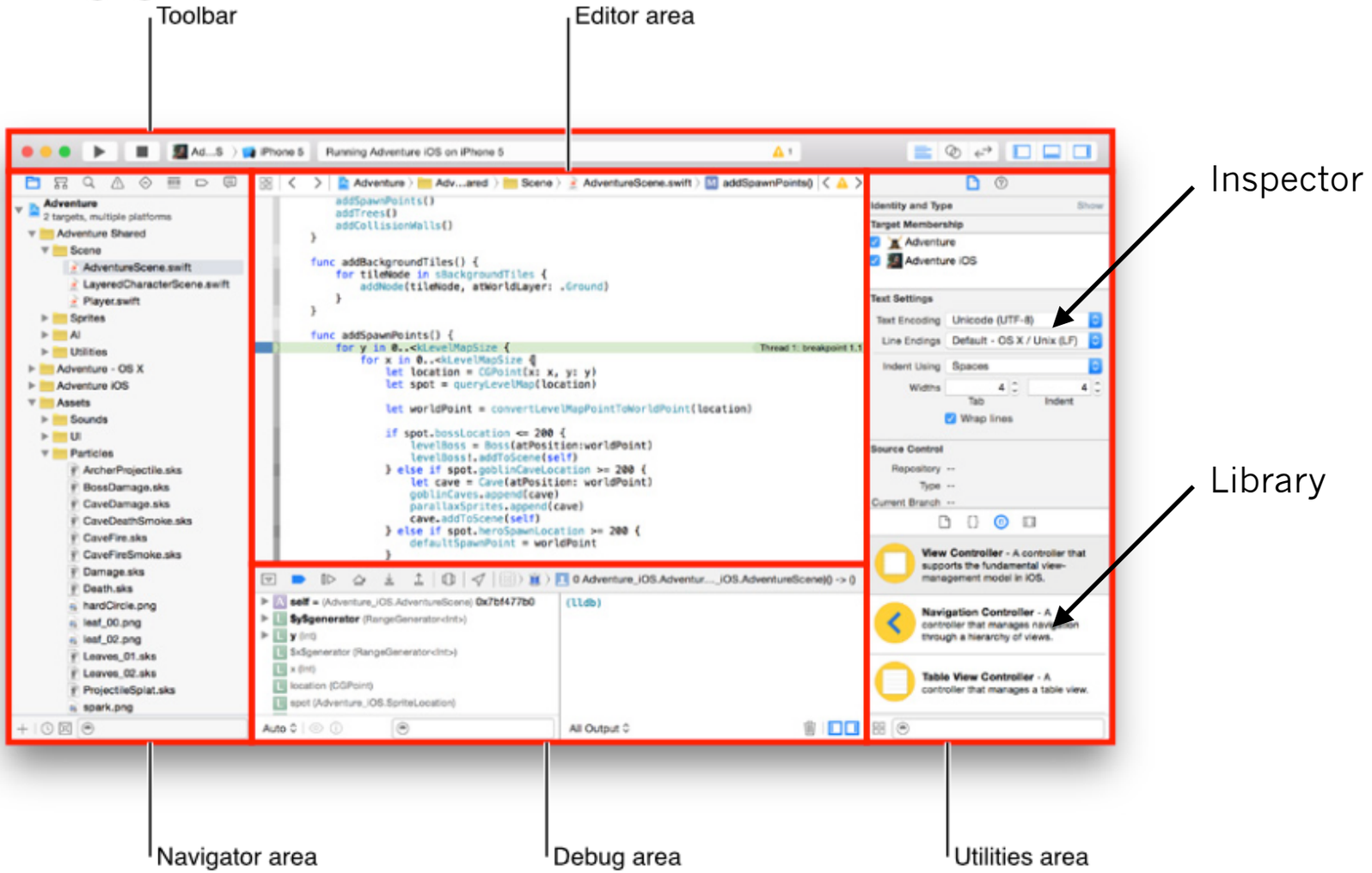
## REVIEW: DEV WORKFLOW

# WHAT IS XCODE?



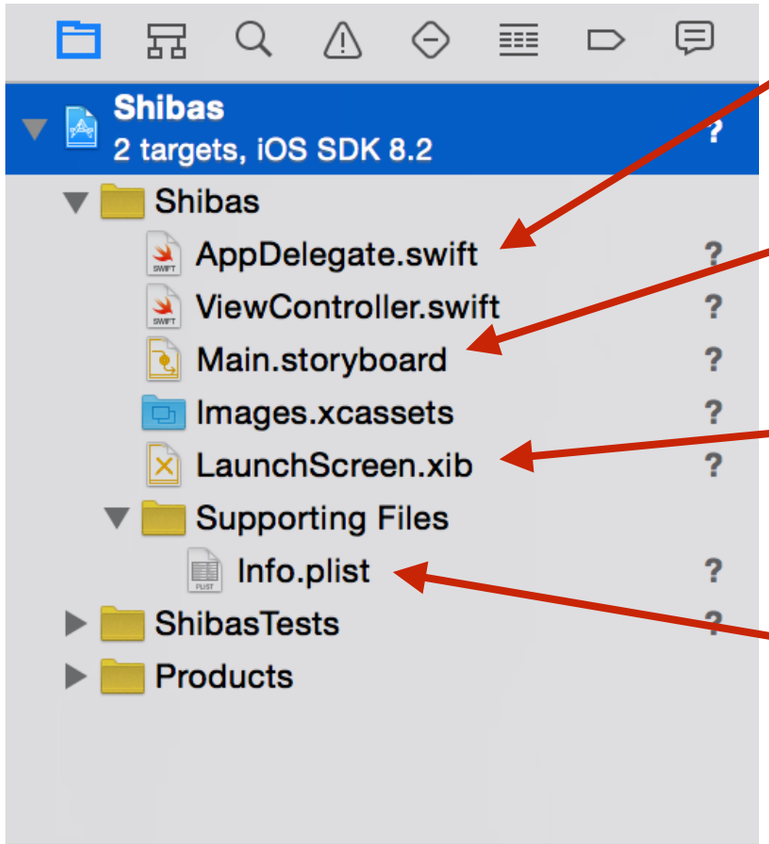
# REVIEW: DEV WORKFLOW

# NAVIGATING XCODE



## REVIEW: DEV WORKFLOW

# FILE TYPES

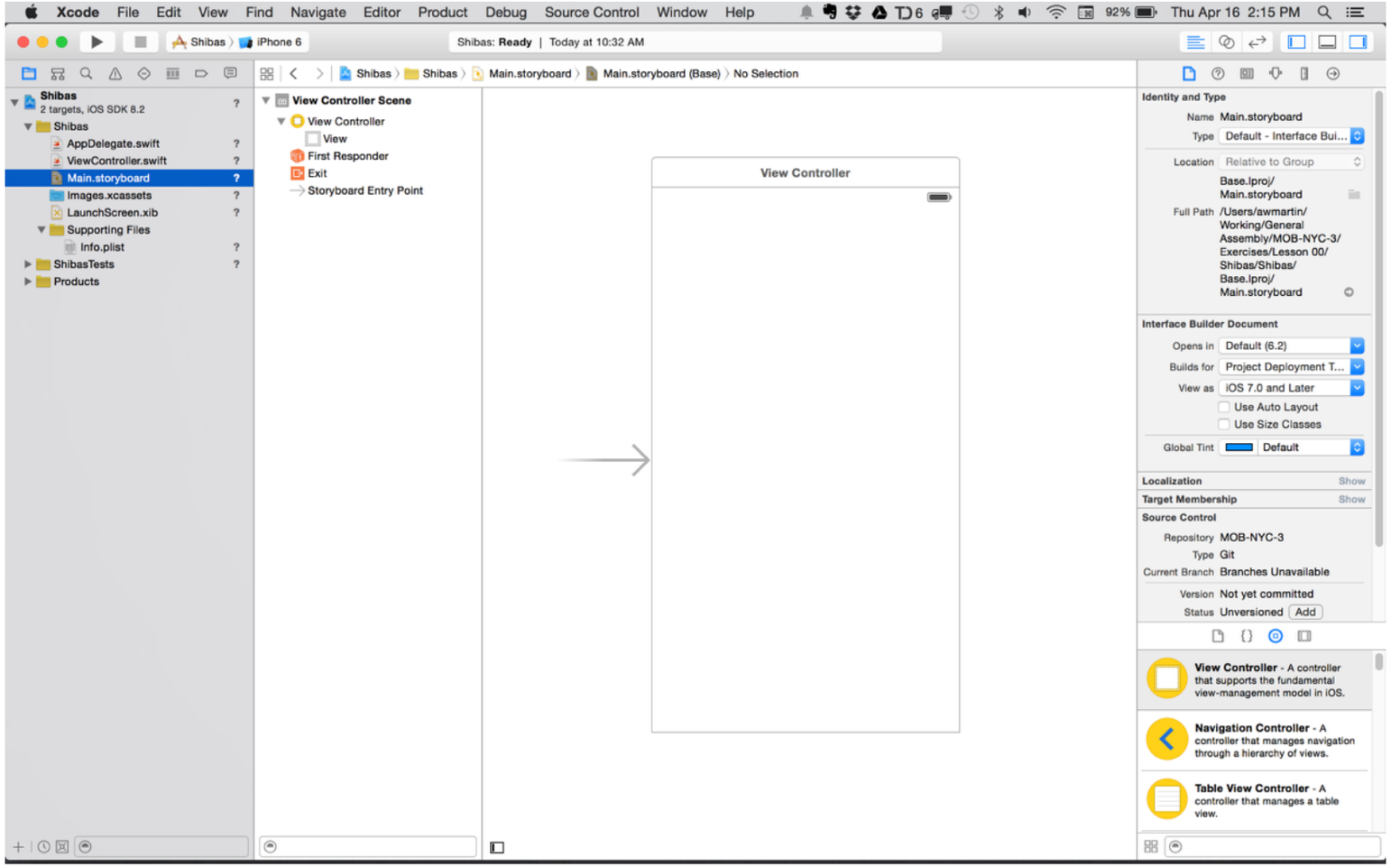



- .swift
  Swift source code file
- .storyboard
  an Interface Builder “storyboard” file
- .xib
  an Interface Builder “NIB” file
- .plist
  a “property list”

# REVIEW: IB + VIEWS

# REVIEW: IB + VIEWS

# INTERFACE BUILDER



# GETTING VIEWS ON SCREEN

- To start understanding iOS apps, we'll first tackle 'views' on the screen
- Almost everything we see on screen is a **view.**
- There are lots of kinds of views:
  - Buttons, labels, tables, images, etc

# VIEWS ON THE WHITEBOARD

# GIT/GITHUB, HOMEWORK, SUBMISSION

## REVIEW: IB + VIEWS

# COLLABORATION SOFTWARE

# GITHUB

- A social network for sharing and collaborating on code
- What we'll use to get slides, submit homework, post resources, collaborate on the final project
- Free, as long as what you post is public

# GIT

- The ‘pipes’ that power github.
- Many developers use the command line, though we’ll use an app.
- A general-purpose ‘version control’ tool that lets us:
  - Back up
  - Revert
  - Collaborate
  - …our code

## LOGISTICS: HOMEWORK

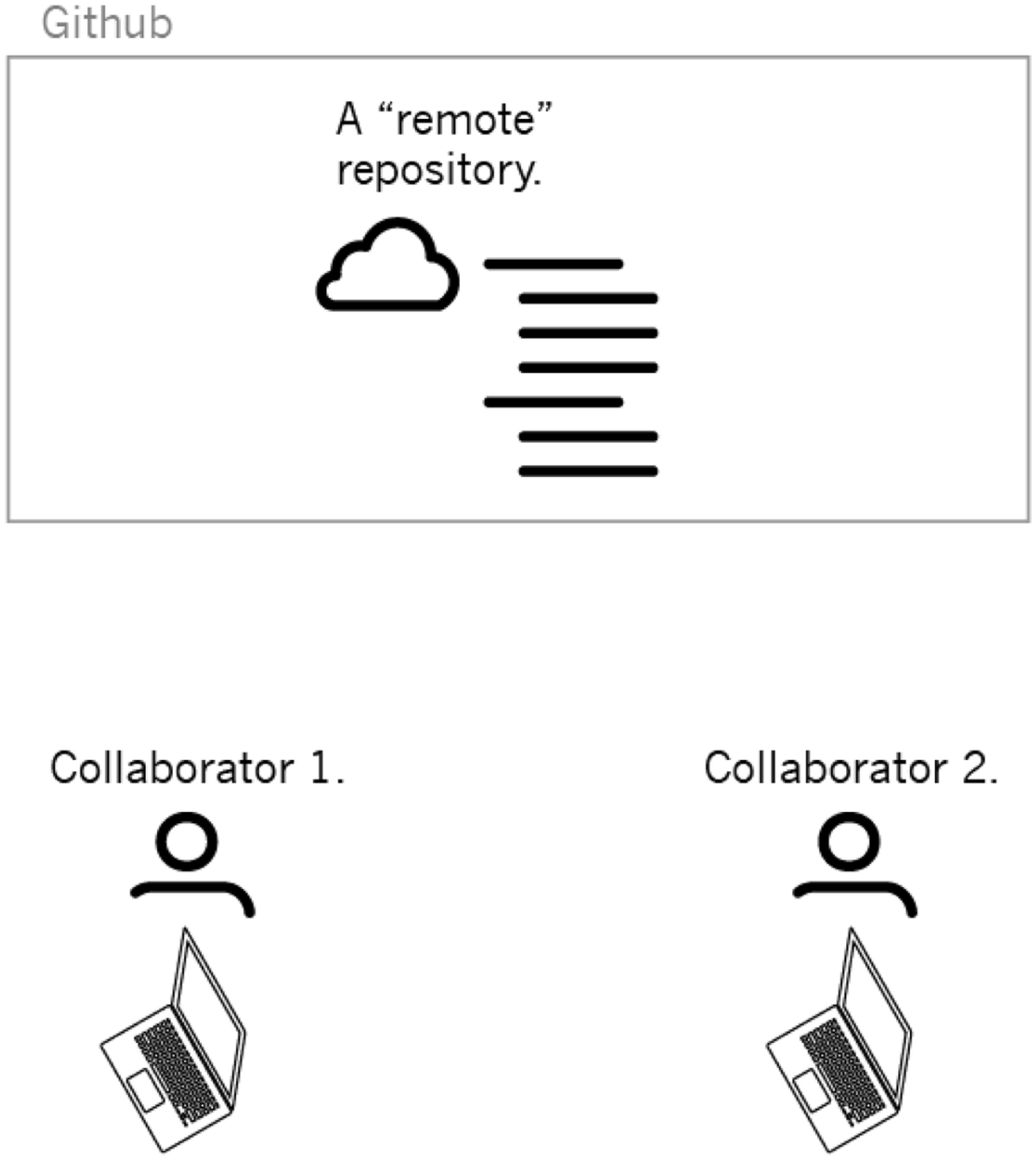
# GITHUB WALKTHROUGH

HTTP://GITHUB.COM/GA-STUDENTS/MOB-NYC-3

## LOGISTICS: HOMEWORK

# GIT WORKFLOW

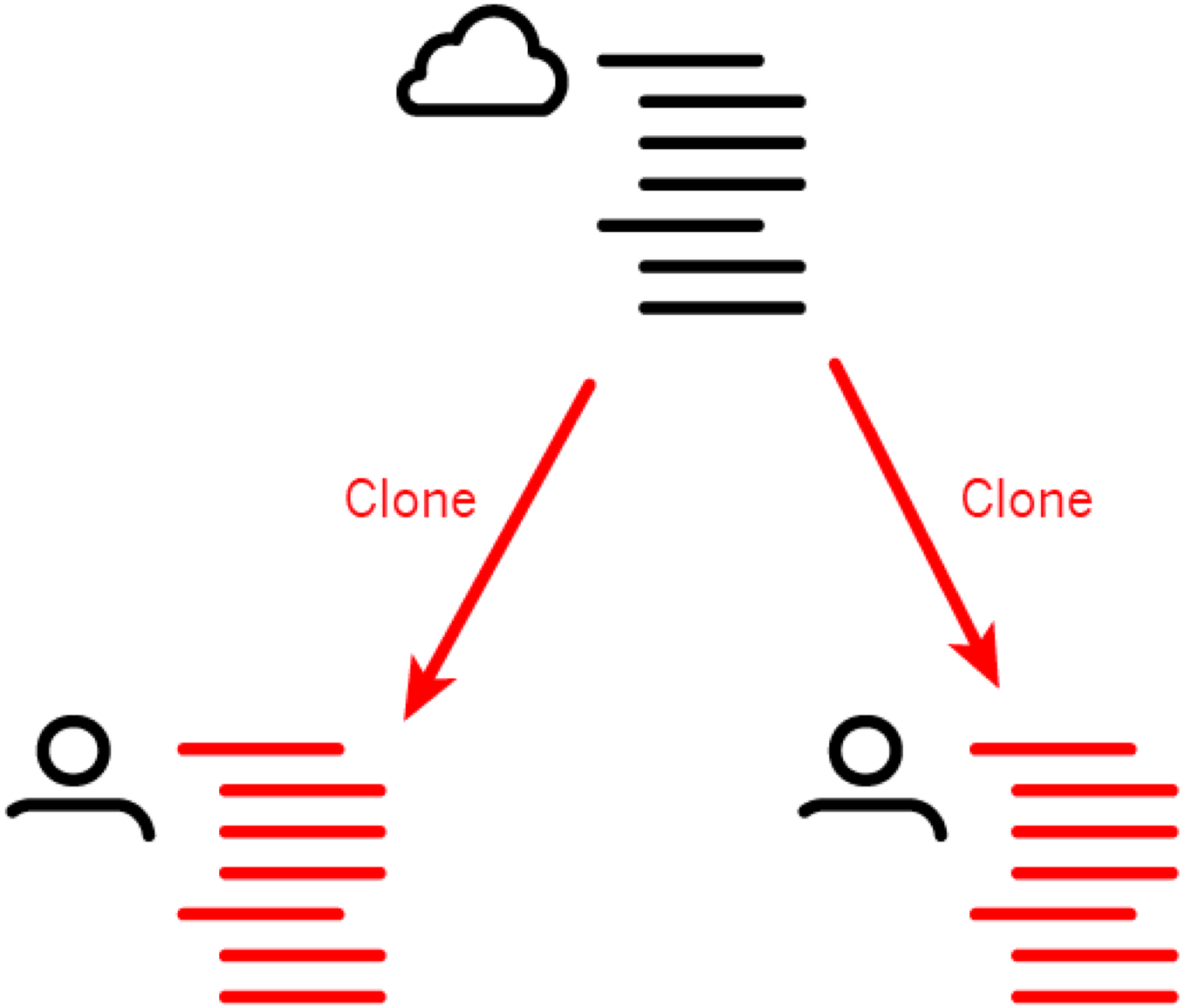
- Git manages changes in a project between many different collaborators.
- It helps everyone manage and contribute to code that could be constantly changing.



## LOGISTICS: HOMEWORK

# GIT WORKFLOW

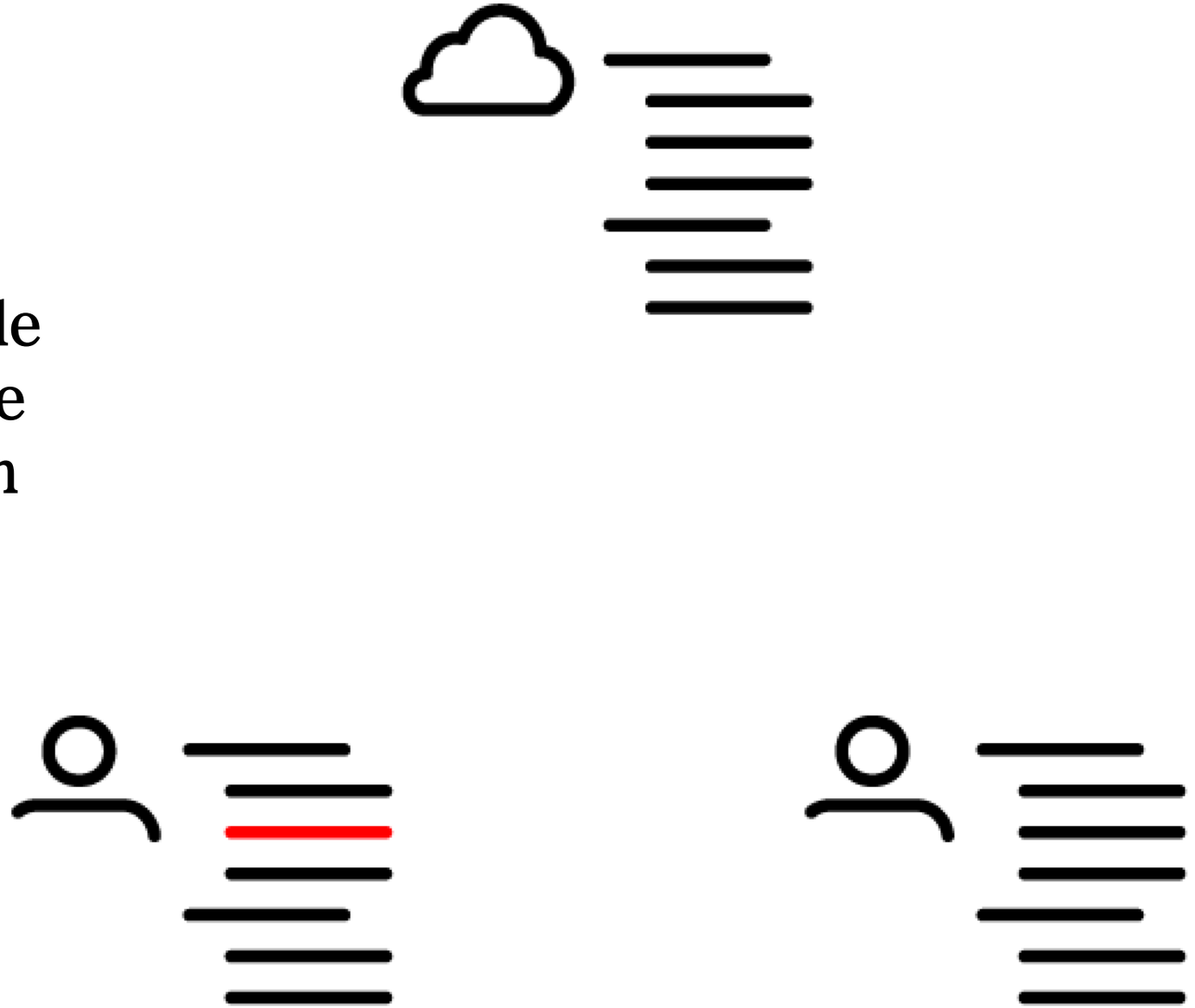
- So the way git works is that everyone has a copy of the repository on their own computer, and a “remote” copy on the server (in the “cloud”).
- That initial step to get the “remote” copy is called “clone.”



## LOGISTICS: HOMEWORK

# GIT WORKFLOW

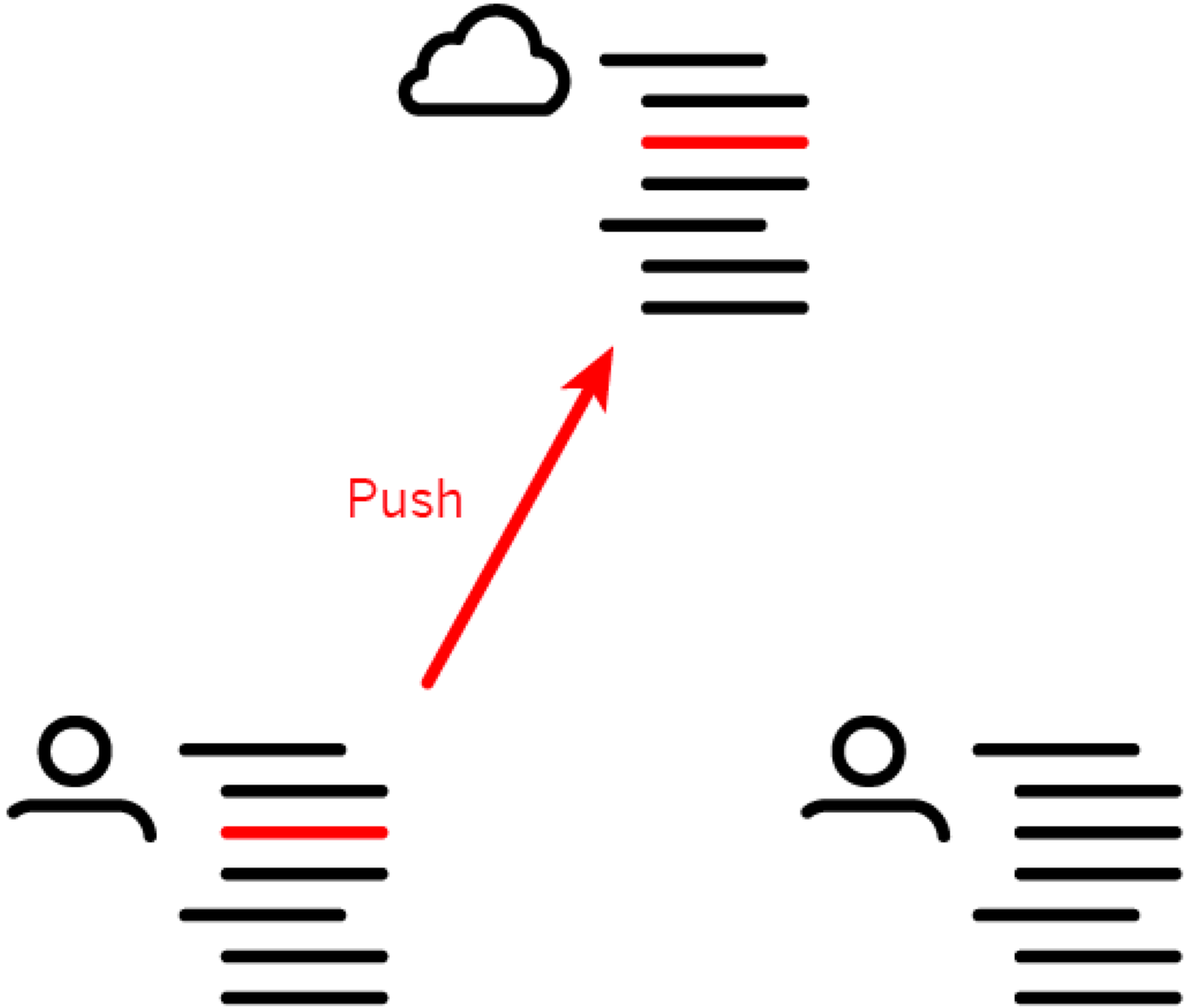
- When someone changes a file (the red line represents a file that's changed), the problem is how to update everyone else on the team with that change.



## LOGISTICS: HOMEWORK

# GIT WORKFLOW

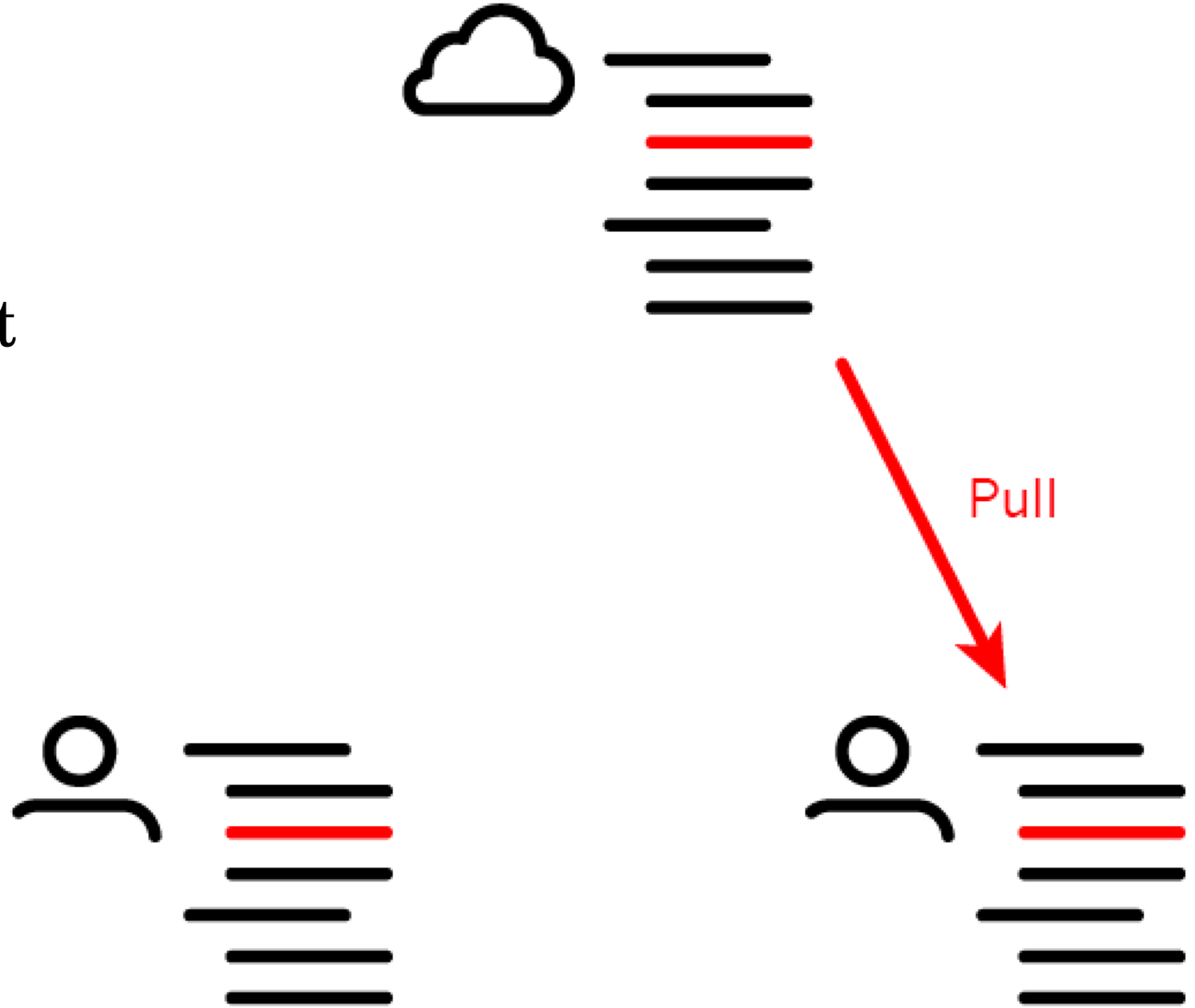
- All the changes are centralized through the remote copy of the repo.
- To send that change to the repository, so it “knows” about it, we use a command called “push.”
- We often say “push your changes” to refer to this process.



## LOGISTICS: HOMEWORK

# GIT WORKFLOW

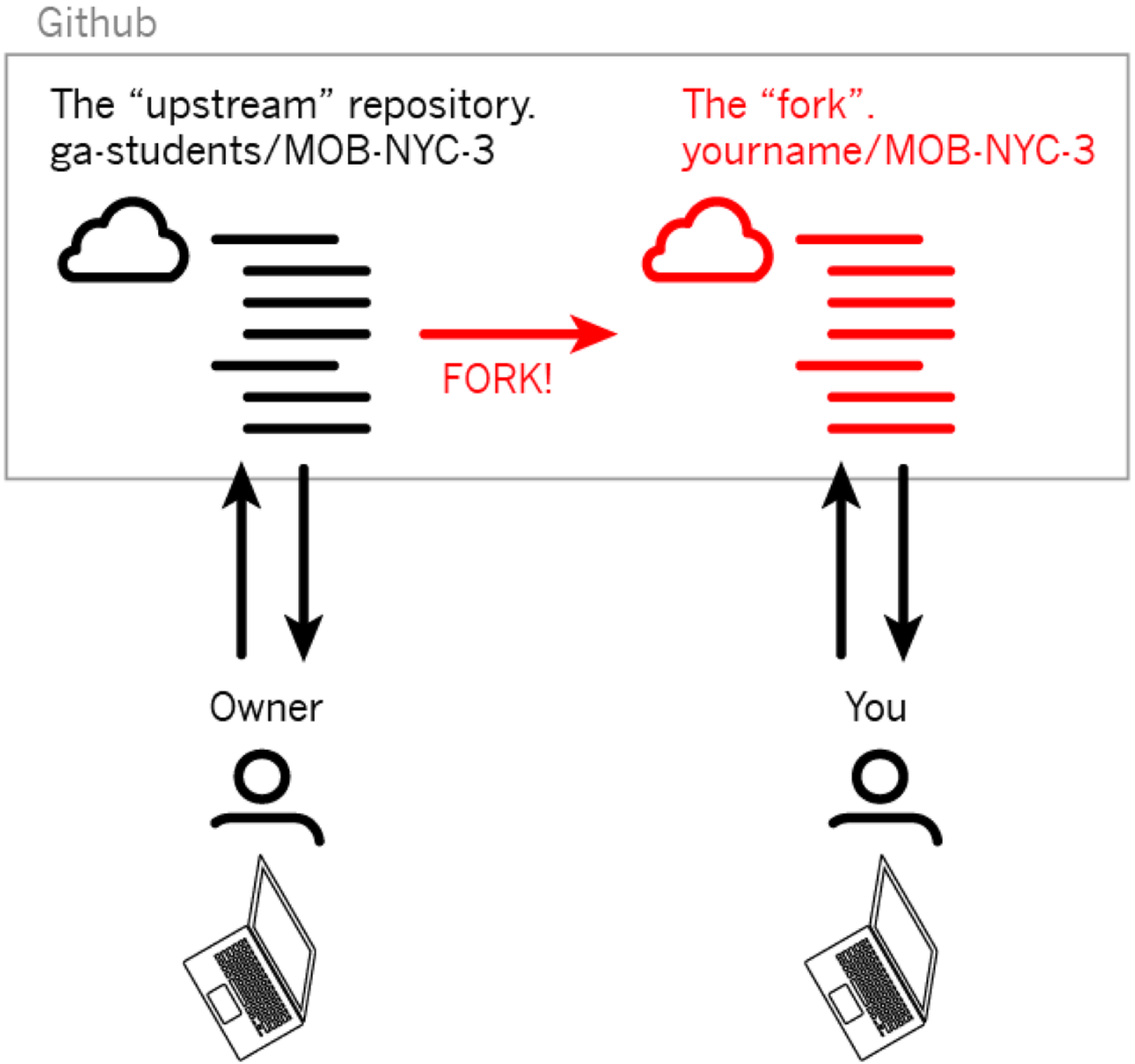
- To retrieve any changes that others have made to the remote repository, we “pull” those changes down.
- This actually represents a complex process of “fetching” those changes, and “merging” them into your copy of the code.



## LOGISTICS: HOMEWORK

# GITHUB WORKFLOW

- To work in Github, we'll be "forking" the main repository to create our own copies of it.
- This will enable you to submit homework without interfering with other students' work.



# Q&A