Git & Xcode - Crash course

This will be a short introduction for initial iOS project setup in Xcode and storing it to Git

GitHub authentication setup

In order to be able to pull private repositories, we'll need to authenticate with GitHub. For that we'll be using SSH keys as shown in next steps. If you are not familoar with SSH keys authentication, you can check this post but for the sake of this tutorial no prior knowledge is expected. We'll be following this tutorial from GitHub so feel free to check it for more info on some steps if needed.

Generating a new SSH key

1. First, we'll generate a new SSH key. Open iTerm and paste the text below, substituting in your GitHub email address.

   ```
   ssh-keygen -t ed25519 -C "your_name@domain.com"
   ```

2. When you're prompted to `Enter a file in which to save the key,` press Enter. This accepts the default file location.
3. When you're prompted to `Enter passphrase (empty for no passphrase)` press Enter (**important for next steps**). This accepts empty passphrase.

Adding a new SSH key to the ssh-agent

1. Start the ssh-agent in the background.

   ```
   eval "$(ssh-agent -s)"
   ```

2. Copy this (CMD + C) chunk of text.

   ```
   Host *
     AddKeysToAgent yes
     IdentityFile ~/.ssh/id_ed25519
   ```

3. Then we run the command which will take the content of our clipboard and paste it to `config` file. Please don't copy the command below since you'll overwrite the clipboard. But if you need to copy it, paste it to the terminal and copy the chunk above before running the command.

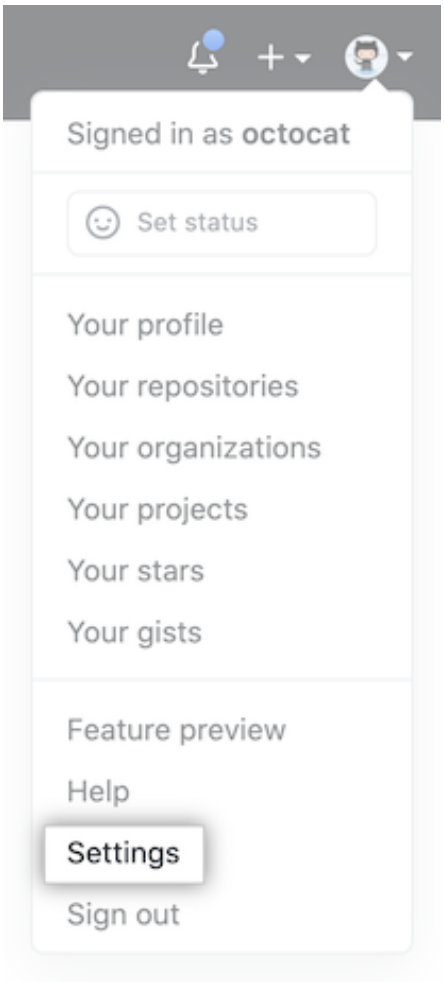
   ```
   pbpaste > ~/.ssh/config
   ```

4. Add your SSH private key to the ssh-agent. If you created your key with a different name replace *id_ed25519* in the command with the name of your private key file.

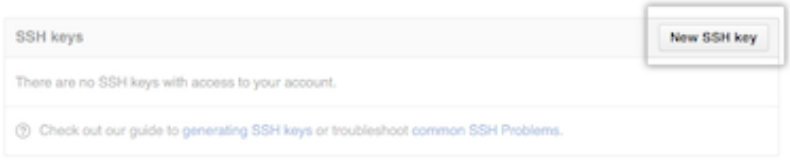
   ```
   ssh-add ~/.ssh/id_ed25519
   ```

Adding a new SSH key to your GitHub account

1. In the upper-right corner of any page, click your profile photo, then click Settings.

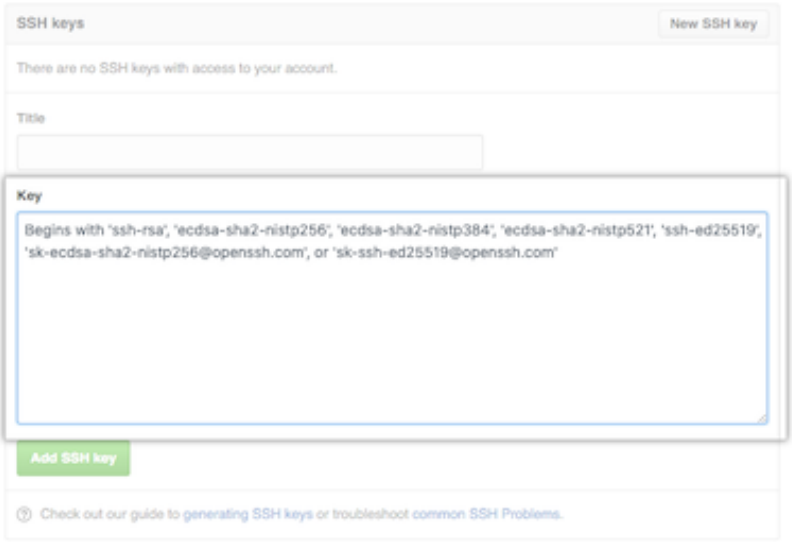


2. In the "Access" section of the sidebar, click SSH and GPG keys.
3. Click New SSH key or Add SSH key.



4. In the "Title" field, add a descriptive label for the new key. For example, if you're using a personal Mac, you might call this key `Personal MacBook Air`.
5. Open iTerm, copy the SSH public key to your clipboard via command and paste your key into the "Key" field.

```
pbcopy < ~/.ssh/id_ed25519.pub
```



6. Click Add SSH key and if prompted, confirm with your GitHub password.

Git setup

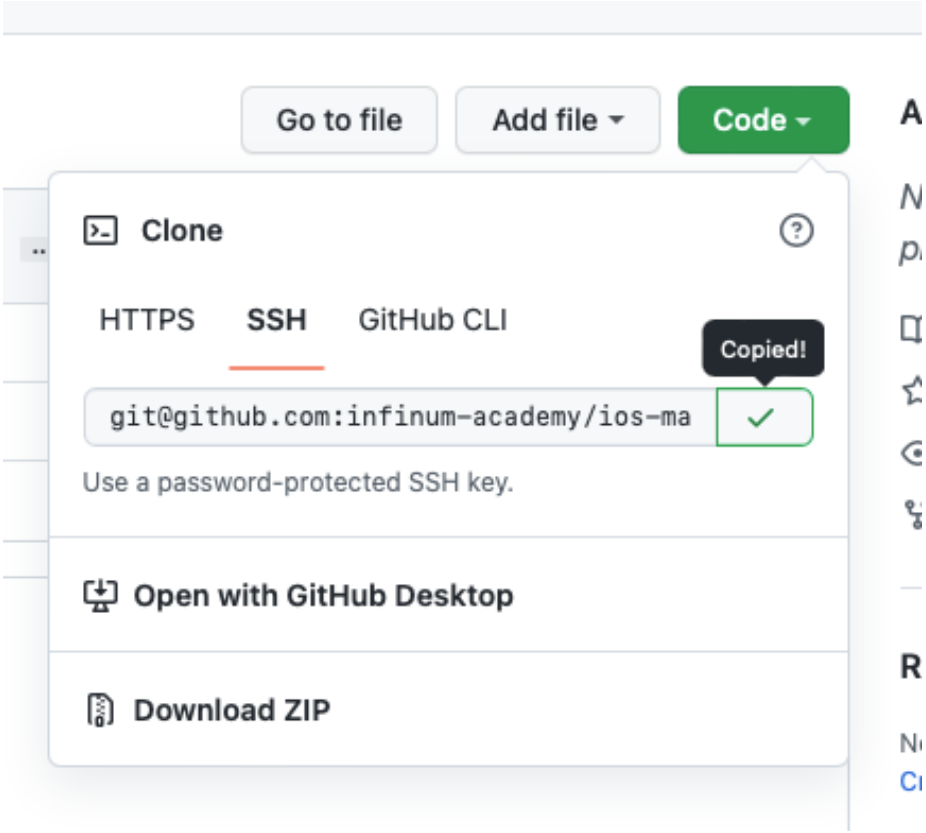
For start, we need to configure the Git on our macOS. To distinct our commits from others we need to set our name and email address. To do that open the terminal and enter these two commands (replace placeholders with your real values):

```
# Set your username
git config --global user.name "Your Name Here"

# Set your GitHub email address
git config --global user.email "your_name@domain.com
```

Cloning the repo

1. Open GitHub and navigate to your *Academy* repo. Select `Code` (green button on right). Select `SSH` and copy the link



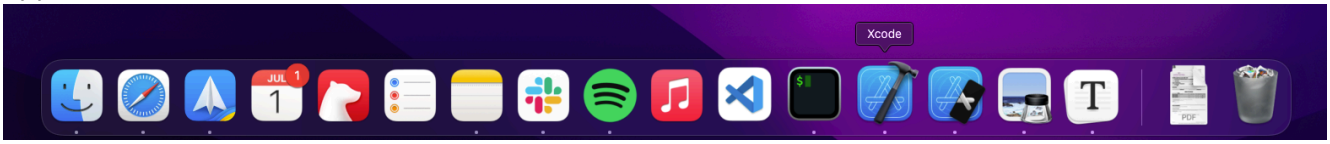
2. Open iTerm and navigate to a directory where you want to GitHub repositories with `cd` command. For example `cd ~/Documents`
3. Type the command which will clone your remote repo to Mac (replace link with one copied from previous step):

```
git clone git@github.com:infinum-academy/ios-materials-2022.git
```

4. You are ready to go!

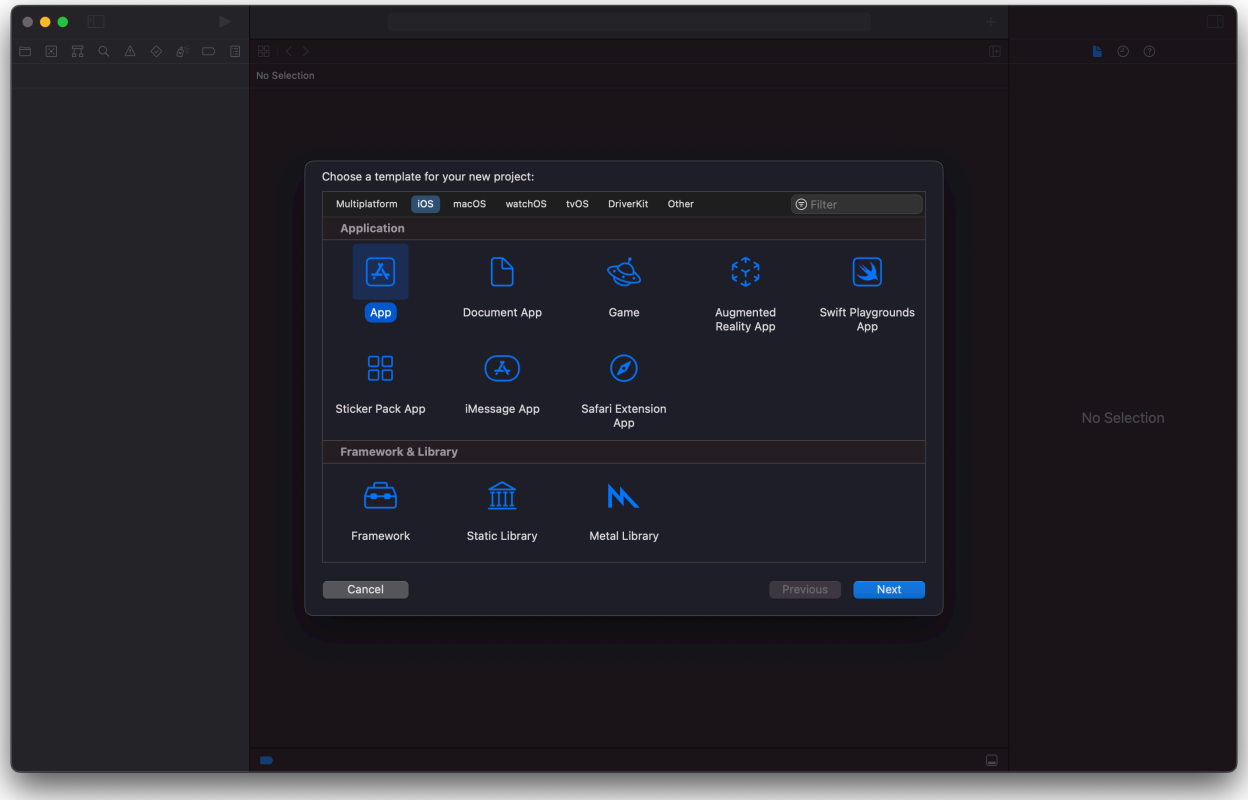
iOS project setup

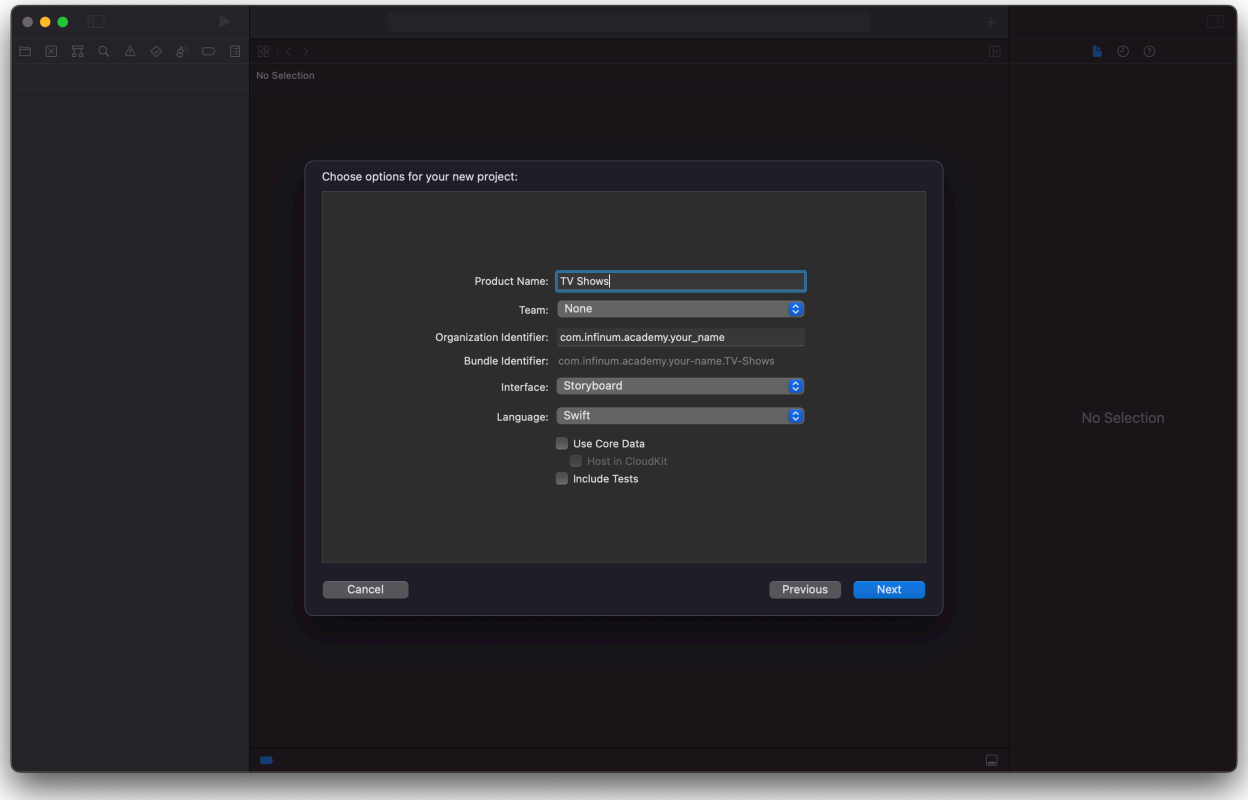
1. First, open your Xcode, you can find it on your *Dock* or in *Applications* folder (let us know if you can't find it). Also, if you like to use keyboard shortcuts, you can use *CMD* + *Space* shortcut and type the application name.

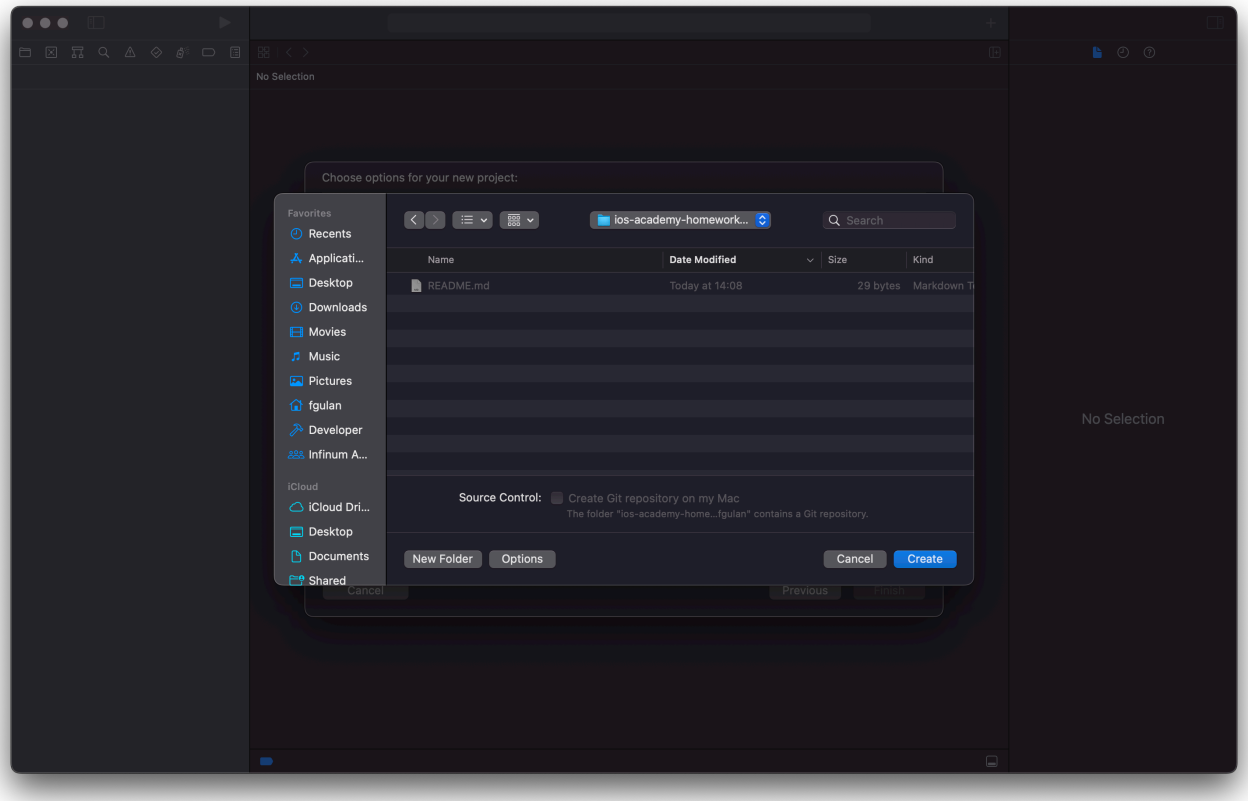


2. If everything goes fine you should see similar window. There you should select option `Create a new Xcode project`, which will create a new Xcode project for you application.

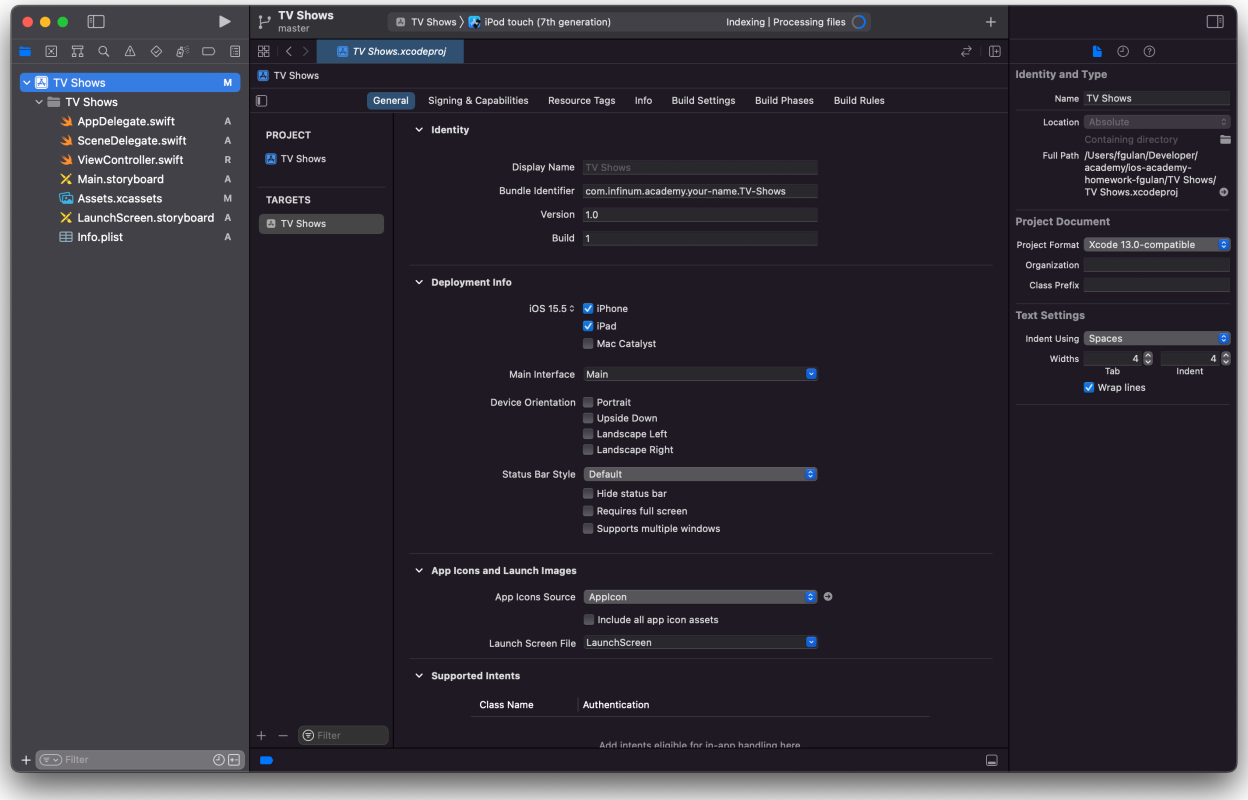
3. Now you'll need to go through few mandatory steps: choose project type (iOS - Single View App), set application name (TV Shows), organization identifier (similar to Java package naming - add some organization prefix/inverted URL + your username, because it needs to be unique), programming language (Swift), deselect all checkboxes and folder (**select your cloned repo folder**).







And if everything goes okay you should see the next screen, something like this:

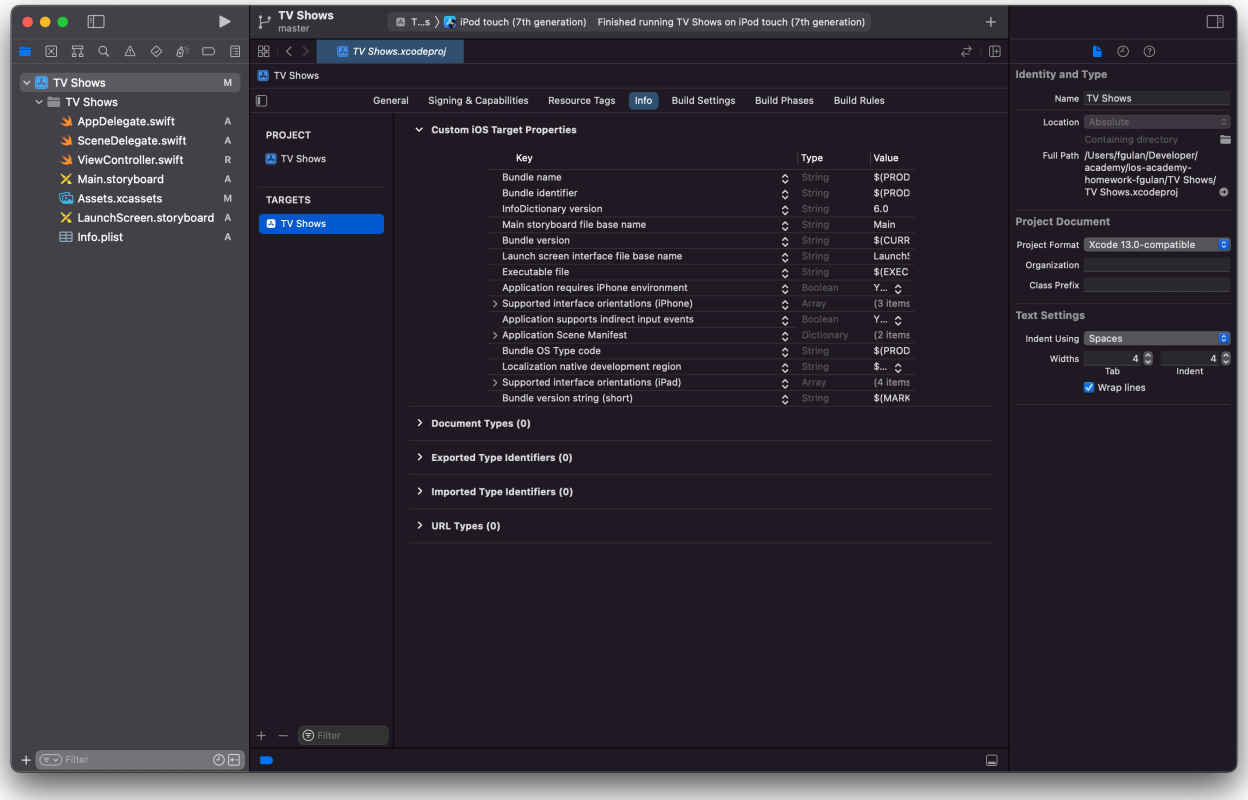


Now you have successfully created your iOS application Xcode project!

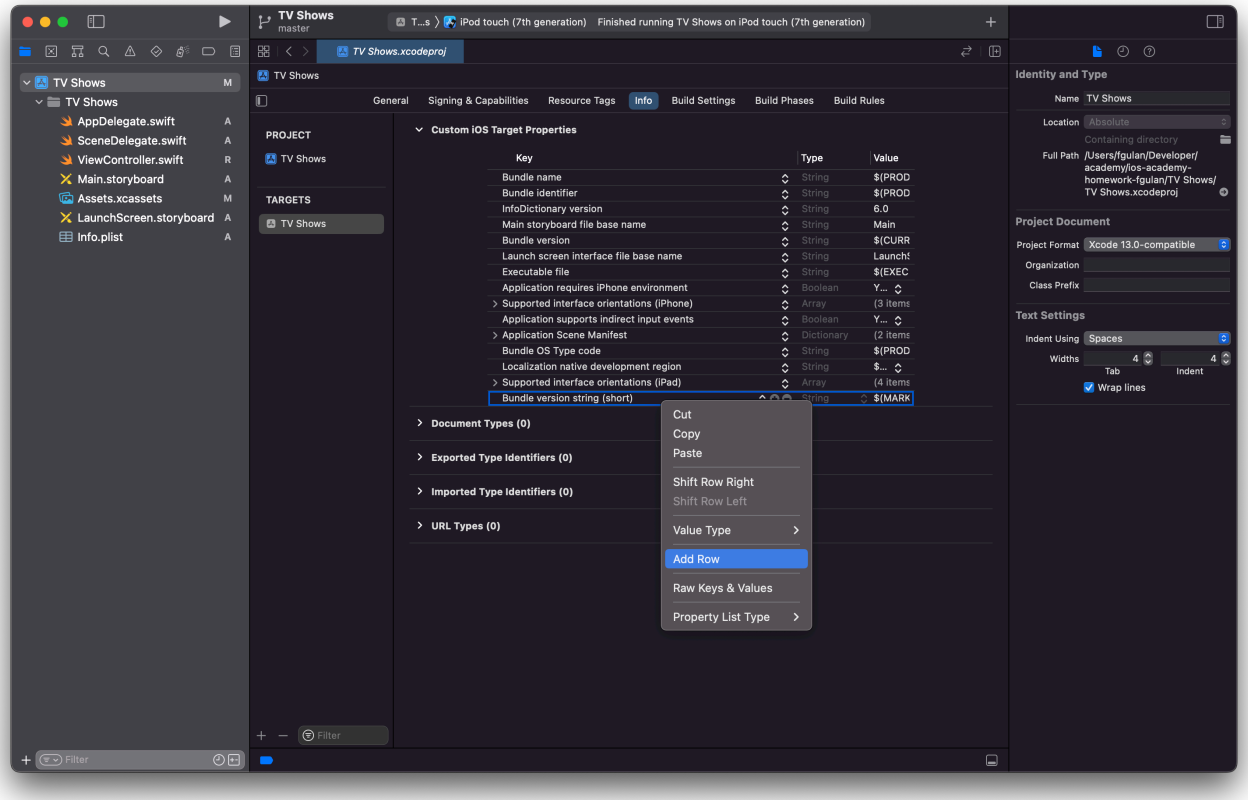
Basic iOS settings

With the Xcode we can control the minimum version of iOS on which our app can be installed. Practice is to support at least two latest major versions of iOS, in our case it will be iOS 13, 14 and 15 ATM. More info on why or why not you can find in this read. Let's change our project minimum iOS version to iOS 13 and set our appearance support only to Light variant!

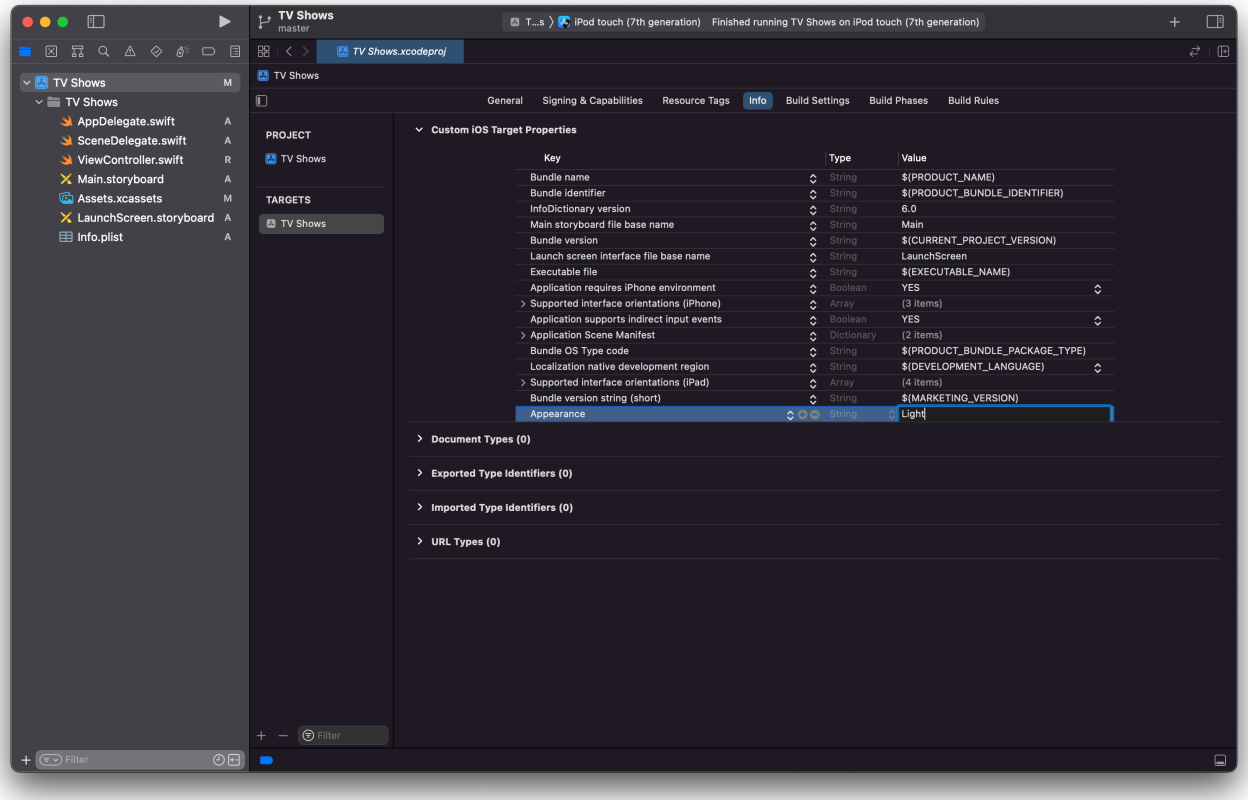
1. On project settings screen pick `Info` from toolbar



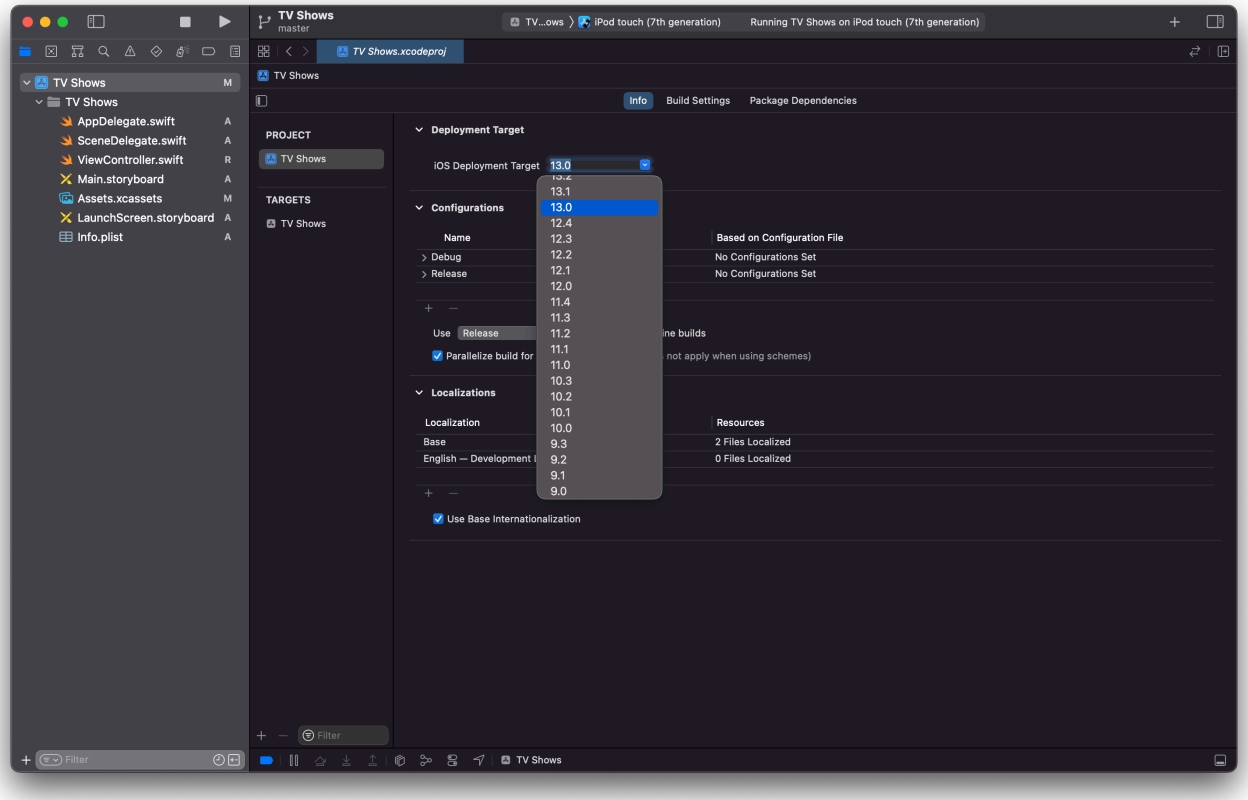
2. Right click on any row in Info and select `Add row` option



3. Enter `Appearance` as key, and `Light` as value and press Enter



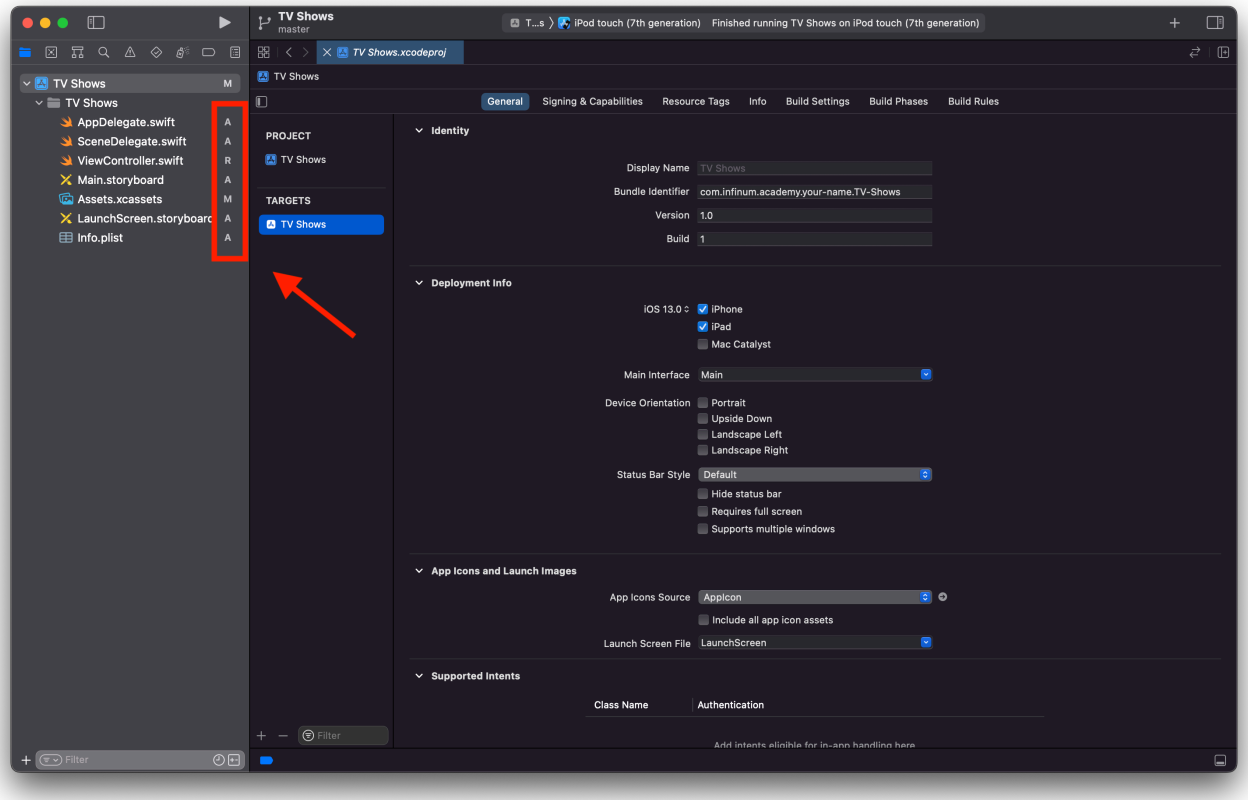
4. In project navigator select your project (blue icon on top), on the left side menu pick again blue icon in Project section (not target) and set iOS deployment target to 13.0!



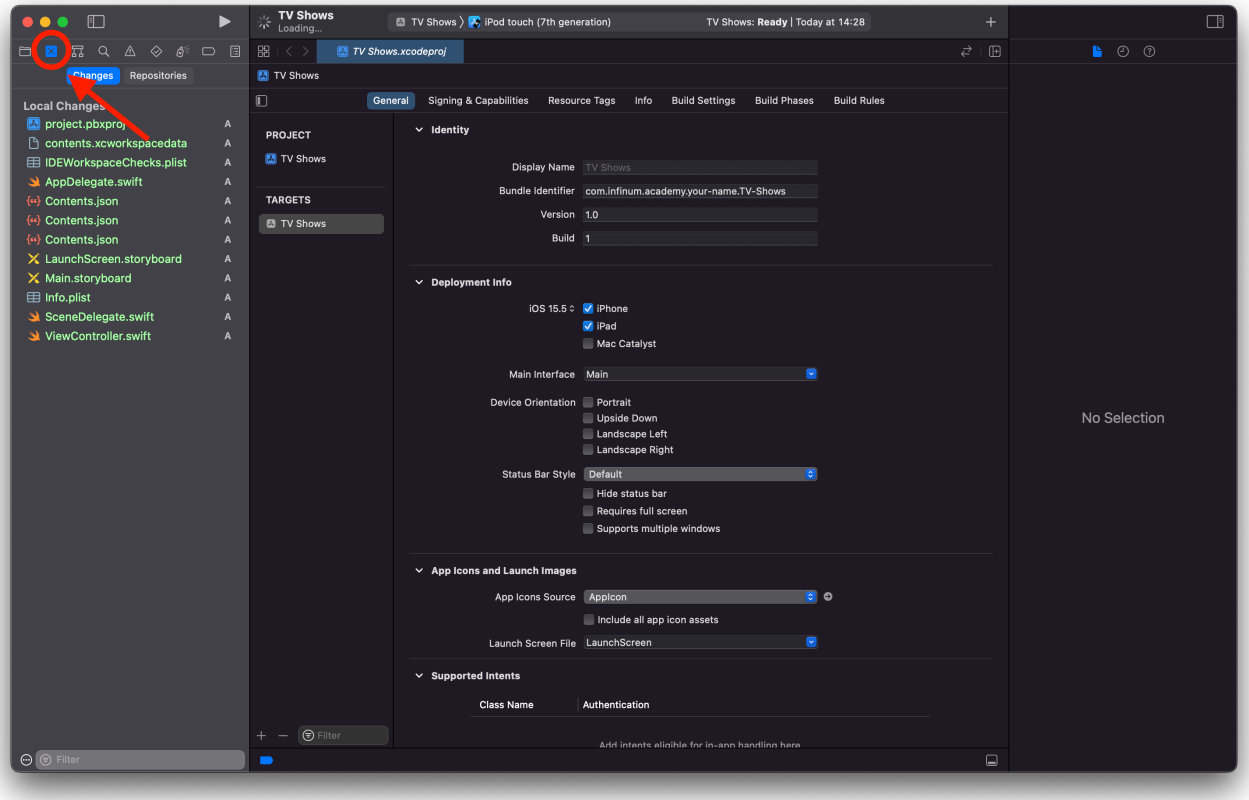
Connecting Git and your Xcode project

In this section we will explain it how to use Git inside your Xcode project.

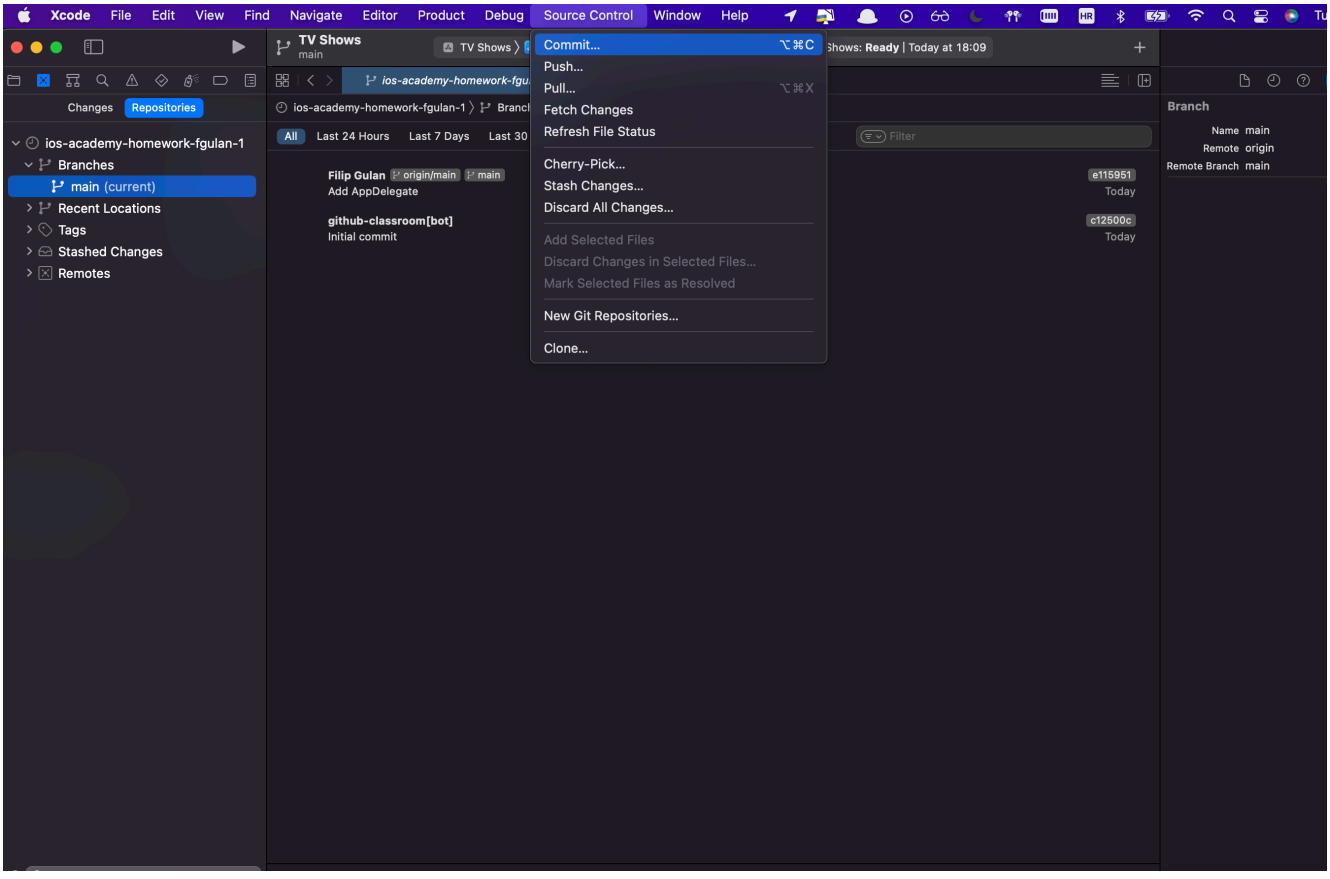
1. In *Source Navigator* you can see all project files and folders. On right on each of them there are some letters like **A** (Added), **M** (Modified)... which are showing the *git* status of that file since last commit. Since we have created project, and we didn't commit it yet, now is the right time :)



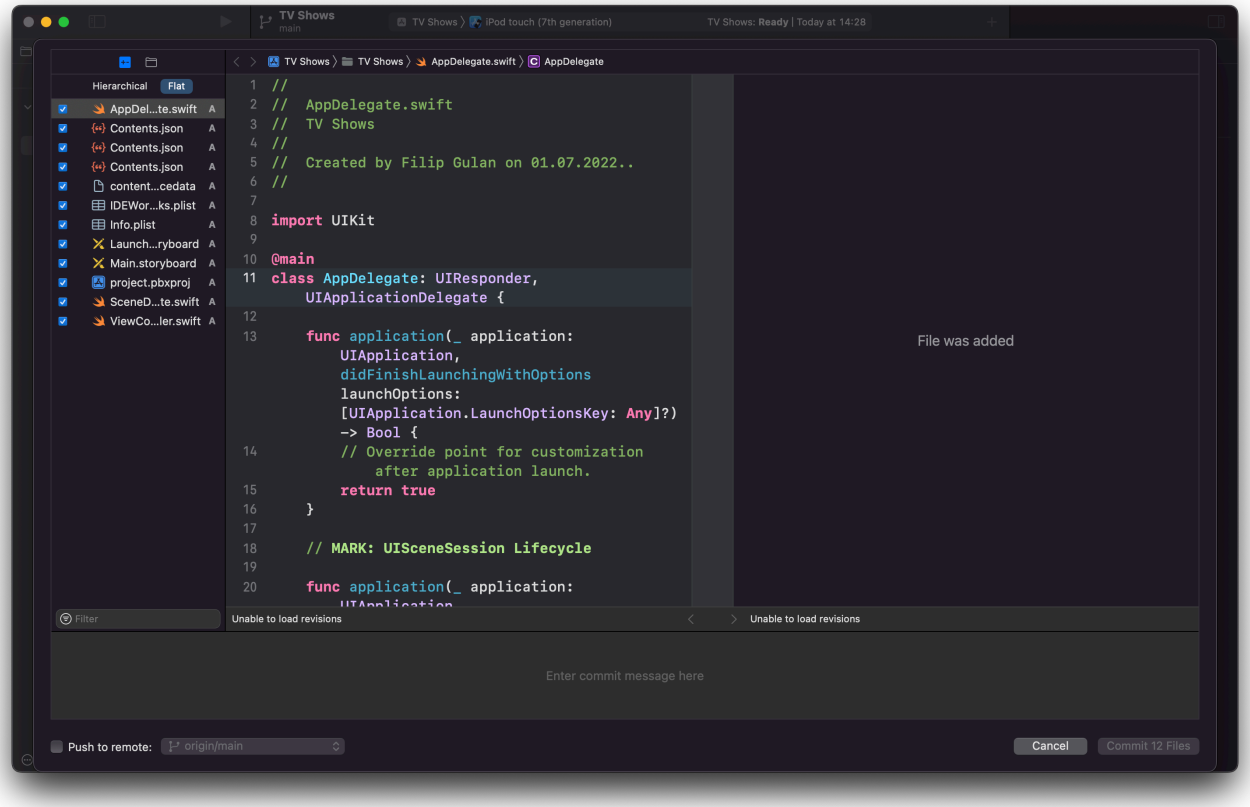
2. Select second icon inside *Project Navigator* menu. This will open a source control menu where it will list all your local and remote branches.



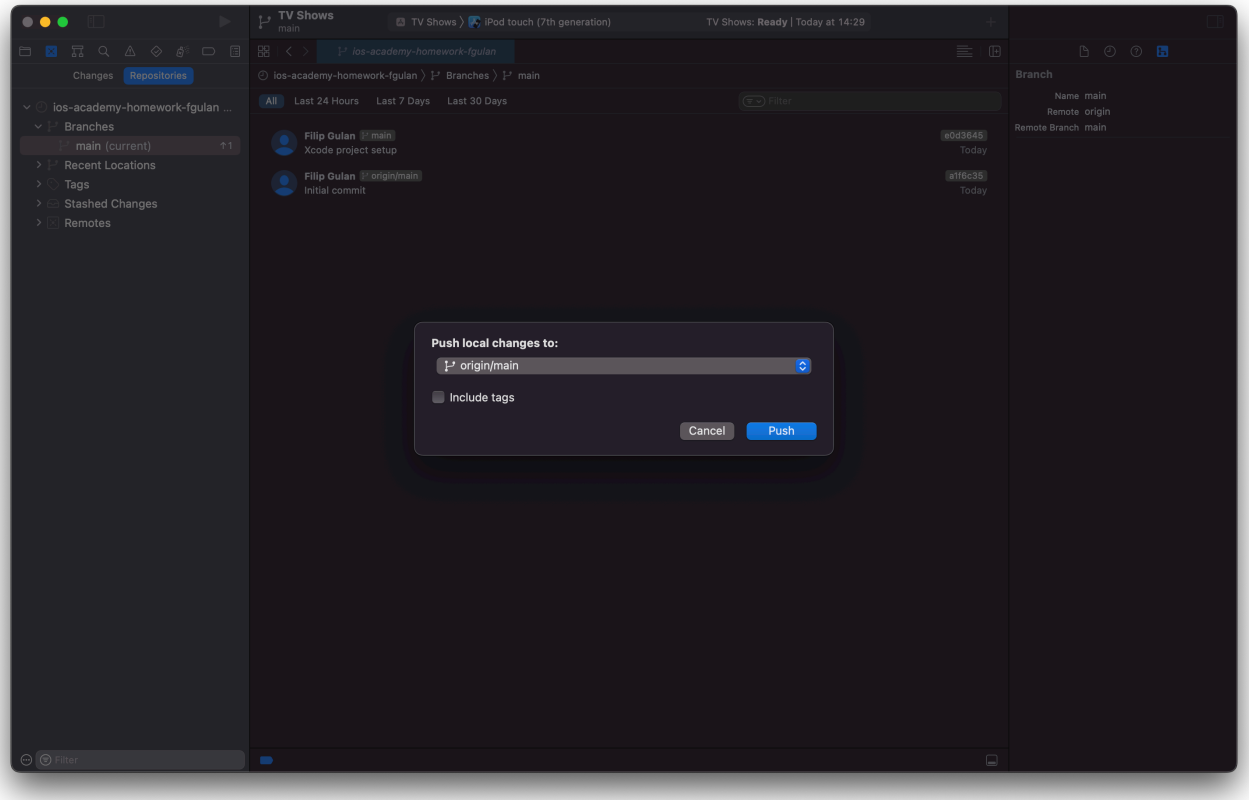
3. To commit made changes navigate to *Source Control* menu and select *Commit*.



4. Now you should see a window with all your changes and files to commit. Select ones you want to commit, add some sane commit message and press *Commit n Files*.

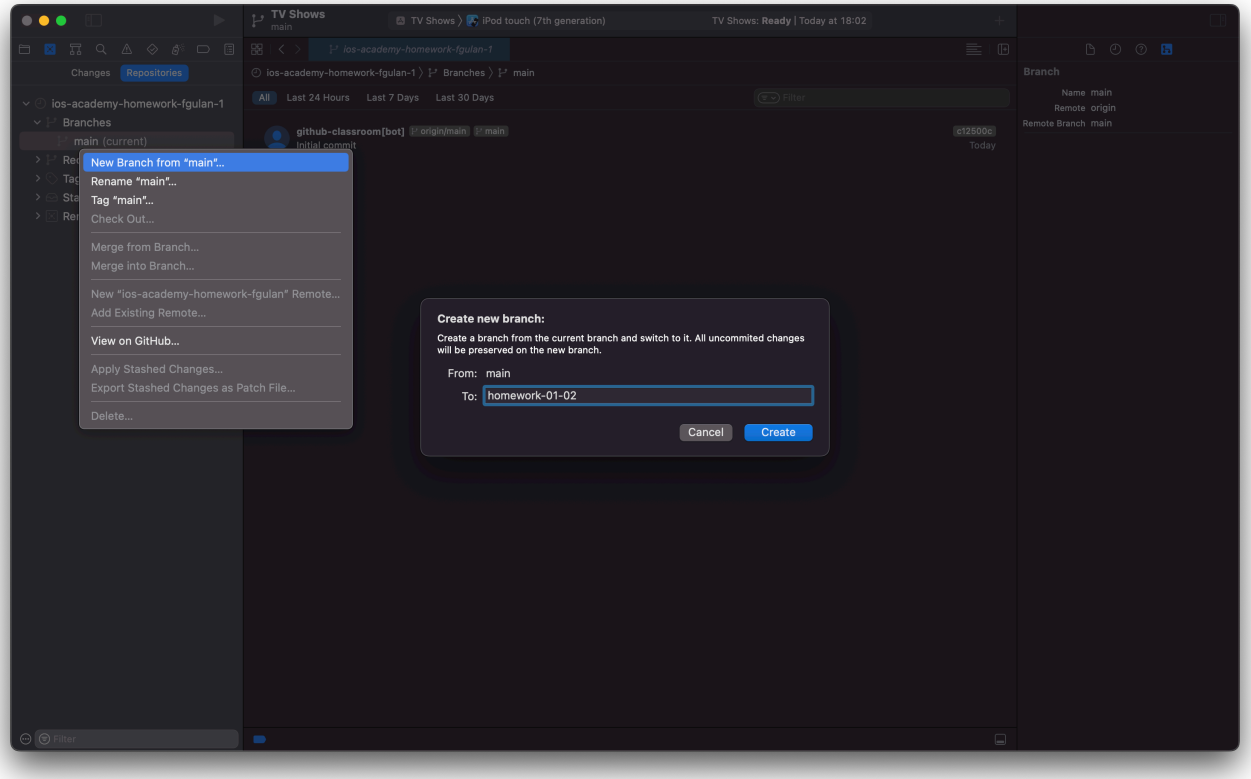


5. Navigate to *Source Control* menu, but now select *Push*. Now you should see small window wich asks you which branch do you want to push, in this case *main*. Press *Push*.



Creating a new branch

1. Now you'll create a new branch from `main` and name it `home-assignment-01`. Expand `Branches` section, right-click on `main` and select `New Branch from "main"`. It will open a popup where you'll give your new branch name, in this case `home-assignment-01`, and create it. When new branch is created it will automatically switch you to new branch. If you want to manually switch to another branch you'll need to select `Checkout` on desired branch.



2. After new branch is created lets play with commiting stuff. Navigate back to *Source Navigator* menu (first icon on *Project Navigator* menu). Select `ViewController.swift` file and add `print("Hello world!")` statement inside `viewDidLoad()` method. Press `CMD + S` to save your changes. Now you should see `M` sign next to the opened file in *Source Navigator*. That indicates that you have some changes that are not commited to Git.
3. **Commit and push changes!**