Time	~ 1 hour
Learning Goals	• Setup a new Rails app from scratch • Prepare a simple Rails remote repository • Revise on how to create a repo and use Github

Now that we're ready to code with Rails, let's walk through a hands-on session about Rails initialization. Here, we'll be initializing a Rails app for the URL shortener app/Bit.ly clone we did in Week 4.

1) Installing Rails

By now, you should have your Rail set up inside your machine. Just to be sure, let's install the gem once again. We will be using Rails 5 for the coursework.

```
gem install rails -v 5.0.1
```

2) Creating A New Rails App

Now that you have the gem installed, it's time to build an app from scratch. Here, we'll be using a postgresql database instead of sqlite3.

To create a new Rails app, with additional options to tweak database, we use rails new command with the -d prefix to use Postgresql:

```
rails _5.0.1_ new url_shortener -d postgresql
```

You'll need to wait for Rails to bundle the necessary gems into the system. Once done, you'll see an app folder inside where you are. Enter the app folder by using cd:

```
cd url_shortener
```

Now check your Gemfile.lock and make sure that the Rails version of 5.0.1 is used.

(Check out this reference to understand more about managing multiple Rails versions.)

ADDITIONAL NOTE:

The actual command format is: rails new <app name> --d <type>. If you are interested, you can do some research on rails new and explore its options.

3) Initialize Git For Version Control

In Week 4 you would have already done a challenge on using git. Now that the app is created, we should implement version control to begin recording history of the codebase. Let's recap on how to initialize git.

3.1) Perform git initialization using git init:

This initializes your current folder as a git repository. All changes to the files and folders within your current folder can now be recorded.

```
git init
```

3.2) Add the files into staging for commit using git add:

Before we can commit new changes, we need to add the new files into staging. Let's add everything in the current directory by supplying '.'. You can also supply the commands '-A' for the same effect.

```
git add .
```

3.3) Commit the first commit using git commit once done with staging.

Here, to keep things simple, let's use -m option for adding short messages to our commit.

```
git commit -m 'This is the first commit'
```

IMPORTANT: Only use the -m option when you have a short commit message. Remember that your commit messages should be **helpful and descriptive enough** for you and your team members. If you need to write longer commit messages, use git commit without the -m option.

ADDITIONAL NOTES:

Remember that commits are small. When it is said small, it means one purpose. Example of commit messages:

1 Add Feature A
2 Fix bug #11212

Whenever you see yourself using 'and' in your messages, chances are you're combining large commits. Bad Example: add feature A and Fixed bug #11212.

Also, **be serious with your commit messages**. Git commit tracks your code copyrights. It reflects your credibility in your software journey.

4) Create Github Repository

Now that you have the local repository, let's create a remote version. In this bootcamp, we use Github as our service provider.

4.1) First create an empty repository in your GitHub account. You can locate it inside the navigation bar, shown in the picture below:

4.2) Fill in the necessary details.

Now you can fill in the necessary details into your new repository. For this app, we named it 'url_shortener'. You can fill in the description on your own.

By default, Github provides the public repository for free. If you're willing to buy, you can select private. For this tutorial, the public is good enough so I'll stick with it.

The next thing is about generating README file and license. In our case, since we have our rails app generated and ready, we don't have to create one.

Hence, the end result should look something like this:

4.3) Create the repository and get the next instruction.

Once you create the repository, there will be some on-screen instructions. It looks something like this:

Okay. By now, **let's learn how to read instruction**. As shown, our case is more suitable for existing repository. Let's follow its instruction to link our local repository to GitHub remote repo.

IMPORTANT NOTE:

You got to learn to read the documentation. We know, it's lengthy and sometimes is scary but that's where you get your next direction. **Most of the time**, the developer spends his/her time researching.

While you're reading always look forward to answering these questions:

1. What is the item I'm learning
2. Why am I learning this knowledge
3. How can I do next
4. When should I apply this knowledge

Anything you learn has to fit these questions. Otherwise, you won't learn much by skipping #1, #2 and #4.

4.4) Once everything is completed, you'll see your code hosted in your public repository. Hooray! Your first app development on the internet.

The end result is something like this:

Now that you are done, let's move on to build a Bitly clone with Rails as a warm-up before we dive into AirBnB!