Afternoon Challenge

```
Core:
* Complete the attachment 'Afternoon Challenge - Core - Rails Walkthrough'

* Follow the tutorial at:
    https://guides.rubyonrails.org/getting_started.html
```

Daily Lesson Breakdown

- Rails Intro
- Convention Over Configuration
- App Structure
- Gems Revision
- Deployment
- Environment Variables

Requirements

Resources

- https://guides.rubyonrails.org/

Lesson Material

Rails Intro

Rails is an **open source MVC framework** built on the ruby programming language. It is used by as the framework of choice for many different organisations such as:

- Airbnb
- Hulu
- Airtasker

The reason for its popularity is because it allows developers to rapidly develop dynamic web applications and web sites. It does this by following two main principles of convention over configuration which we will talk a bit more about in a second and DRY.

```
What does DRY stand for?
```

Do Not Repeat Yourself

```
And what do that mean?
```

We abstract our code in such a way so that we avoid writing the same lines of code over and over.

Rails also has built in commands that can generate lots and lots of boilerplate for us so we don't have to write it every time we need to create a new app or if we want to add features to an existing rails app.

```
What is boilerplate?
```

Boilerplate is a unit of code that can be included in many places with little to no alteration. Think about when we were creating our HTML websites. There was certain code we need to write for every page. Wouldn't it be better if that code was auto generated for us so we didn't have to write it?

Another great thing about Ruby on Rails is that there are many gems that extend the features of our application that doesn't require much effort from the developer. We can add in to our app sending emails, logging in with facebook, taking payments and much more.

Rails is considered an **opinionated framework** and you will hear all the time in the rails community the phrase, **"the rails way"**. It makes the assumption that there is a "best" way to do things, and it's designed to encourage that way - and in some cases to discourage alternatives. To better understand lets talk about what **convention over configuration** is and why we would want this

Convention Over Configuration

Convention over configuration is a simple concept that is primarily used in programming. It means that the environment in which you work (systems, libraries, language...) assumes many logical situations by default, so if you adapt to them rather than creating your own rules each time, programming becomes an easier and more productive task.

The goal is to decrease the number of decisions the programmer has to make and eliminate the complexity of having to configure each of the areas of application development. The immediate result is that you can create many more things in less time.

It does require a greater learning curve at the beginning because you need to become familiar where everything located and the convention being used. In any case, to use a system in which most things are pre-configured is always more productive than to use a completely open system in which you have to set all the rules and take all decisions.

Creating A Rails App

Before we begin it is important to know where we can find all the documentation associated with Ruby on Rails. We can find the documentation here: https://guides.rubyonrails.org/.

A really important key thing to know is that we are using **Rails V5.2**. Remember the initial number in versioning is considered a major change and code written for this version may be incompatible for different major versions. This is especially true between Rails V4 and V5, so do not use V4 as your code probably won't work in V5.

Another important resource of information for creating a new Rails app is actually within the help option of the rails command.

```
rails -h
```

The output shows you how to create a new Rails app along with some different options we could use during the creation process.

For the moment lets create a new Rails app using all of the default settings.

```
rails new first_app
```

Awesome! It's all installed. Lets cd into the directory.

```
cd first_app
```

And then lets run the help option again.

```
rails -h
```

What black magic is this! We get a whole bunch of other options we didn't have last time. That is because rails is smart enough to know if you are in a rails directory or not when you you are running the rails command.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
There is currently a bug in Rails version 5.2.2, you will need to modify line 9 of
Then do a bundle install
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Lets start up our Rails application.

```
rails server
```

This boots up **puma** which is the web server RoR comes pre-configured for.

```
What is a web server?
```

A web server allows a directory within our machine to be accessible by the network (internet in most cases). This is how we can host websites and web app that are accessible from anywhere in the world!

Within the output of the web server in the console you should see which address it is listening on. By default it is **localhost:3000**.

`What is the default IP address of localhost?`

127.0.0.1

If we go to the address in our web browser we can see our Rails application is running!

App Structure

Lets take a deep look into our Rails application and go through each file and understand exactly what it is used for.

`Go through all of the files in the Rails application, below is a generic table of each section and what it is for. This has been pulled directly from the Rails documentation.`

`When you get to the config directory take a special look at credentials.yml.enc and master.key because these two things go together. The master.key is how you encrypt the credentials file and should never be included in your GIT repository. That is because you never want anyone to get ahold of the information within the credentials file.`

`Show them how to decrypt the file and save new information using this command:`

```
EDIT="code --wait" rails config:edit
```

Video

File/Folder	Purpose
app/	Contains the controllers, models, views, helpers, mailers, channels, jobs and assets for your application. You'll focus on this folder for the remainder of this guide.
bin/	Contains the rails script that starts your app and can contain other scripts you use to setup, update, deploy or run your application.
config/	Configure your application's routes, database, and more. This is covered in more detail in Configuring Rails Applications.
config.ru	Rack configuration for Rack based servers used to start the application. For more information about Rack, see the Rack website.
db/	Contains your current database schema, as well as the database migrations.
Gemfile, **Gemfile.lock**	These files allow you to specify what gem dependencies are needed for your Rails application. These files are used by the Bundler gem. For more information about Bundler, see the Bundler website.
lib/	Extended modules for your application.
log/	Application log files.

File/Folder	Purpose
package.json	This file allows you to specify what npm dependencies are needed for your Rails application. This file is used by Yarn. For more information about Yarn, see the Yarn website.
public/	The only folder seen by the world as-is. Contains static files and compiled assets.
Rakefile	This file locates and loads tasks that can be run from the command line. The task definitions are defined throughout the components of Rails. Rather than changing Rakefile, you should add your own tasks by adding files to the lib/tasks directory of your application.
README.md	This is a brief instruction manual for your application. You should edit this file to tell others what your application does, how to set it up, and so on.
test/	Unit tests, fixtures, and other test apparatus. These are covered in Testing Rails Applications.
tmp/	Temporary files (like cache and pid files).
vendor/	A place for all third-party code. In a typical Rails application this includes vendored gems.
.gitignore	This file tells git which files (or patterns) it should ignore. See GitHub - Ignoring files for more info about ignoring files.
.ruby-version	This file contains the default Ruby version.

Gems Revision

Remember gems are just packages of code pre build for us by other developers that we can download and include in our applications. In Ruby there are generic gems that can work in any Ruby application and then there are gems that are specifically for Ruby on Rails. Both of these gems are downloaded the same way though.

In Ruby on rails we use the bundler package management gem to install and keep track of the gems we are using within our application. These gems are added to our Gemfile.

Lets add another gem to our application through bundler. Remember we need to be in the same directory and our Ruby on Rails application before we run the below command.

```
bundle add faker
```

Deployment

Remember we need a web server to make our application accessible on the internet. Ruby on rails came with a built in web server named puma that allowed us to view our application on our local machines. But we can also deploy our code and run our web server on other machines.

`What is the benefit of doing this?`

- **Able to scale** - can increase the amount of power the machine has easily
- **Networking is already setup** - we don't have to worry about how our application is going to be able to hook into the internet
- **Some security already in place** - this is more networking security than application security such as protection from DDOS
- **Don't have to maintain any hardware** - all of the bare metal uptime is guaranteed by the hosting company
- **Cost** - normally cheaper to run in the cloud

`Ok so if those are the benefits then what are the downsides?`

- **Less control** - you don't normally have full control over what you can install, networking, etc
- **No access to the machine** - normally you don't have any access to the machine to test if your issues are hardware not software
- **Shared space** - when you use cloud hosting you are sharing a machine with many other customers which could cause you issues
- **Cost** - hidden fees or usage can add up quickly

There are many different cloud hosting providers out there such as AWS, Digital Ocean, Azure, Heroku, etc. Lets deploy our app to Heroku! The first thing we need to do is signup for a Heroku account.

`Go to heroku.com and create an account.`

Next we need to install the Heroku CLI.

`Visit` https://devcenter.heroku.com/articles/heroku-cli#download-and-install `and follow the steps to install the CLI.`

Once that is done you need to run the below command and follow the prompts.

```
heroku login
```

Awesome! Now we just need to hook up Heroku to our Rail application. Go to the terminal and cd into your Rails app. Once in the correct directory run:

```
heroku create
```

The really cool thing about Heroku is that it allows us to deploy by using GIT. What just happened when we ran that command is in Heroku it spun up a new virtual machine for us and then added a new remote repository to our Rails app named Heroku. To deploy we just have to push all of our code up to the this new remote.

```
git push heroku master
```

And now it's deployed! How easy was that!

Environment Variables

Ok so you thought we were done but we are still missing a very crucial piece! Remember that we have a master.key file that is used to decrypt all of our secret information associated to our application. Well if you look inside our .gitignore file you will see that master.key is listed, so it is never tracked by GIT! This is definitely what we want because if we pushed our master.key to a GIT repository what was the point of encrypting it in the first place?

But now we have a dilemma that we don't want our master.key in GIT but we need it for our application to properly work on Heroku. Hmm.....how can we solve this issue?

Environment variables saves the day! An environment variables is a variable we can access within our application that is set on the machine itself. Think about it this way, we have variables that we can declare inside of our ruby code correct? Well we can also declare variables in our CLI that our code runs on so that means our code could access the variables without them being coded into the program themselves. We do it this way for security reasons. Unless someone can directly access your machine they have no way to get the values of those variables.

```
What would be other good information to store as environment variables?
```

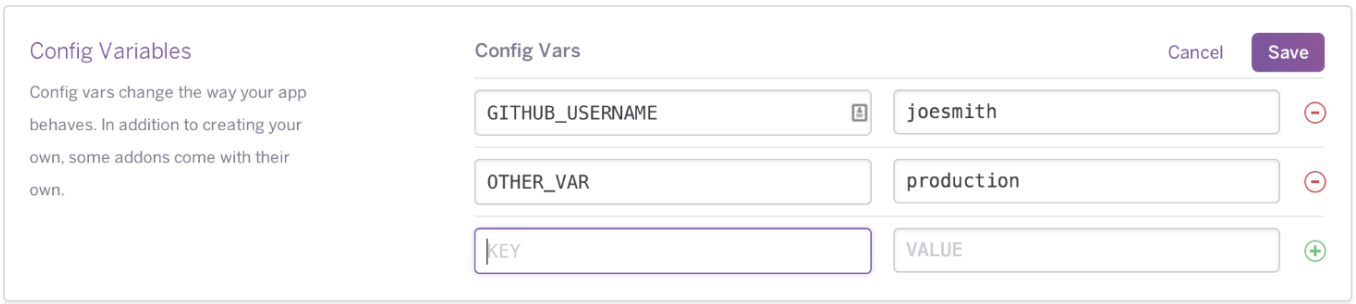
- Application keys
- Usernames and passwords
- Information that changes depending on which environment its deployed

Now Rails is awesome because we don't normally need to use environment variables because we have that awesome encrypted credentials file. But in other frameworks we would need to rely heavily on them.

Lets set our environment variable in Heroku. We can either do it through the command line or within the online Heroku application.

```
heroku config:set RAILS_MASTER_KEY=<your-master-key>
```

or you could set it here:


Config Variables
Config vars change the way your app behaves. In addition to creating your own, some addons come with their own.
Config Vars
Cancel
Save
GITHUB_USERNAME
joesmith
OTHER_VAR
production
KEY
VALUE

Video

Rails File Structure - 1

Rails File Structure - 2

Key	Value
Author	garretb; Converted to `md` - Hamish
Keywords	FastTrack; ftb; Lesson plans; Rails Intro; Convention Over Configuration; App Structure; Gems Revision; Deployment; Environment Variables