Chapter 2

Lab Environment Setup

Objective: Provision a local Ubuntu virtual machine to run labs on

1. Install VirtualBox
   VirtualBox is a hypervisor that can emulate entire operating systems. We will use it to run a local Ubuntu 16.04 instance. Follow the installation instructions to get VirtualBox on your machine.

2. Install Vagrant
   Vagrant is a virtual machine management tool. It operates through a command line interface that allows you to create and manage environment instances from a variety of providers. Follow the installation instructions to get Vagrant on your machine.

3. Provision a Ubuntu 16.04 box
   Now that VirtualBox and Vagrant are installed, we are ready to create a Ubuntu **`vagrant`** box. Create a directory for your box to live in and initialize the box:

   ```
   $ vagrant init --minimal ubuntu/xenial64
   ```

   This created a **`Vagrantfile`** in your lab directory, which declares various configurations for the box (ex: startup-scripts, forwarded ports, hardware allocation, etc.). The labs will require some extra configuration to run, so edit your **`Vagrantfile`** with the following:

   ```
   $ Vagrant.configure("2") do |config|
   config.vm.box = "ubuntu/xenial64"
   config.vm.network "forwarded_port", guest: 80, host: 1080
   config.vm.network "forwarded_port", guest: 1337, host: 1337
   config.vm.network "forwarded_port", guest: 9292, host: 9292
   config.vm.network "forwarded_port", guest: 8080, host: 18080
   config.vm.network "forwarded_port", guest: 8081, host: 18081
   config.vm.provider "virtualbox" do |vb|
     vb.memory = "2048"
   end
   end
   ```

   Finally, run the **`vagrant up`** command to boot the VM, and **`vagrant ssh`** to shell into it. Check out the vagrant documentation for more information on the capabilities of Vagrant.

Lab 2.1

Objective: Create a Git repository; make sure changes can be made and pushed to GitHub.

1. Sign up for a GitHub account. If one is not already created, follow the directions posted here: https://github.com/join. Once registered, ensure the **Git** utility is installed on the Ubuntu system.

2. To see if it is installed and what version it is, type the following command in the command line:

   ```
   $ git --version
   ```

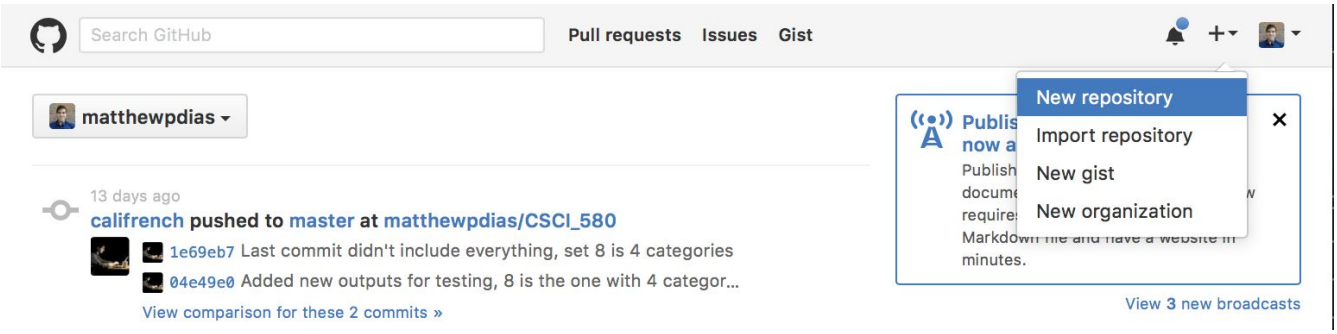
   If it is installed, move on to *Step 3*; otherwise, install **Git** with the following commands:

   ```
   $ sudo apt-get update
   $ sudo apt-get install git
   ```

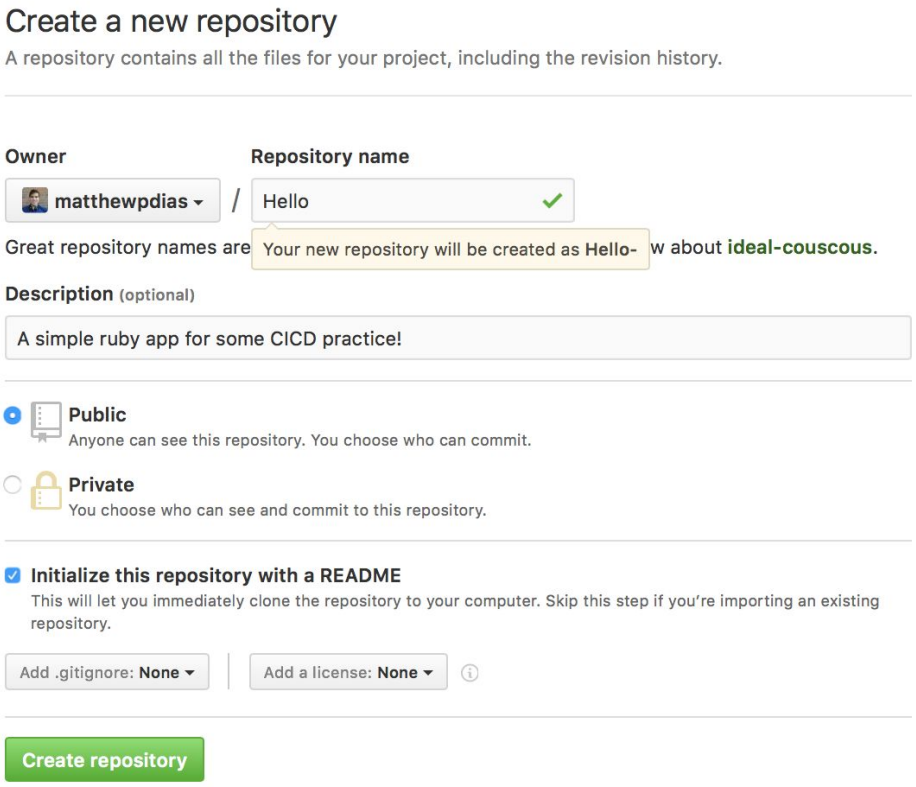
   Next, configure the **Git** install to reflect the GitHub account details:

   ```
   $ git config --global user.email "email@example.com"
   $ git config --global user.name "username_example"
   ```

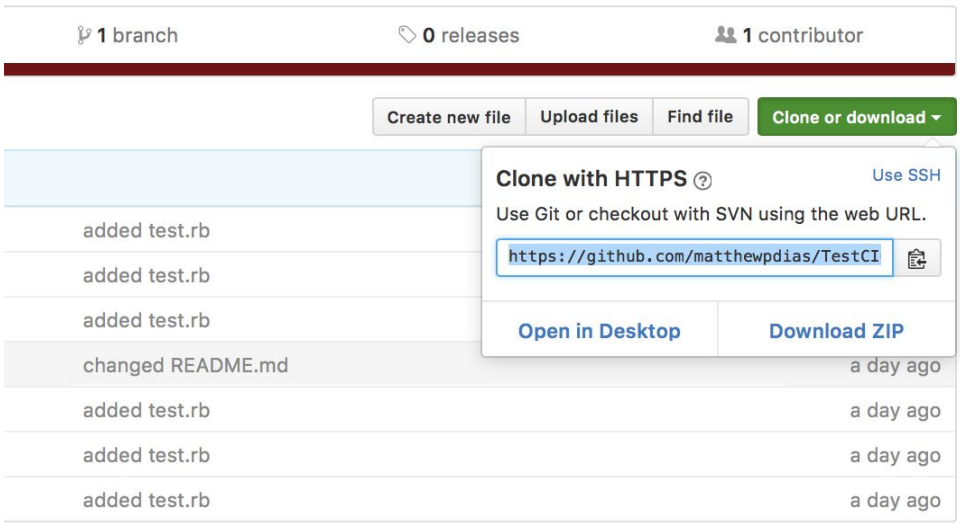
3. Log on to the GitHub account, and navigate to the user page. On the top-right, click the "+" button, and select "New repository".



   Name this project "Hello" and give it a description. Select "Initialize this repository with a README", and then, click on the "Create repository" button.



4. Now, the repository needs to be cloned to the local system. Copy the HTTPS URL to the clipboard, by clicking on the green "Clone or download" button in the GitHub repository. Select "Use HTTPS", and then, copy the URL supplied to the clipboard. This URL will be passed as an argument to the **`git clone`** command, as seen below:



In the Ubuntu workspace, clone the repository:

```
$ git clone https://github.com/username_example/hello.git
```

This repository will be created immediately beneath the present working directory. Navigate into the newly cloned directory:

```
$ cd hello
```

5. Open **README.md** in a preferred text editor, and add some text. The file needs to be changed so that it can be committed into **Git**.

```
$ git commit -am "Changed the README.md"
```

This commits all changed files (**-a**) with the message **"Changed the README.md"**.

Now, push those changes up to the copy of the repository on GitHub:

```
$ git push origin master
```

Git will ask for the GitHub username and password. This is an HTTPS connection, so it is fairly secure. To secure things even further, look into the SSH keys and enable second-factor authentication (2FA) through GitHub.

Now, look at the files on the repository in the GitHub's web interface. The changes should be visible.

Note: All of the labs work on a MacBook Pro with 8GB memory. However, it is important to control memory usage and it might be necessary to restart your computer.

