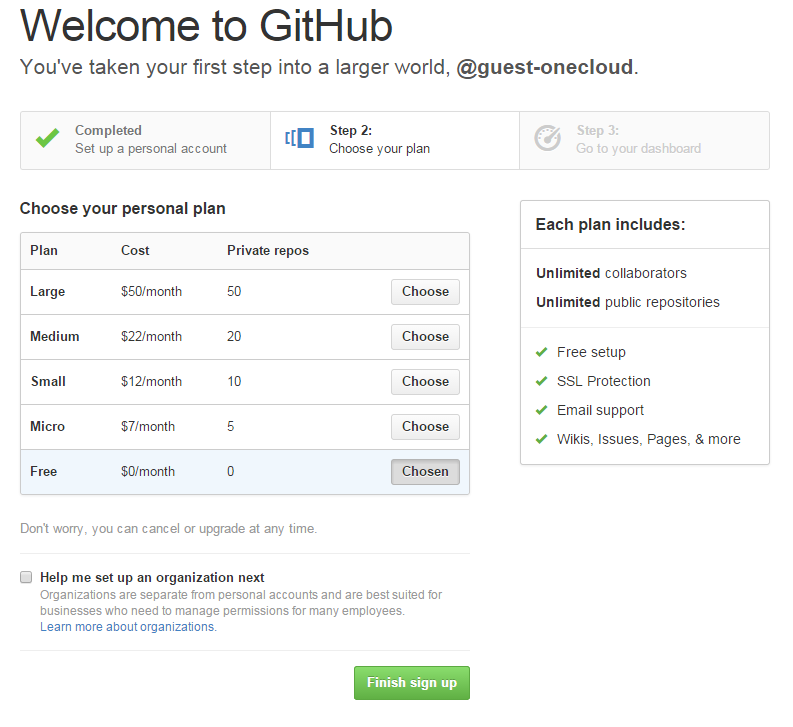
Lab 0 – GitHub Introduction

Introduction

GitHub is a popular web-based hosting service which offers distributed revision control and source code management (SCM). In this lab we will walk you through setting up your GitHub account, and cloning the lab repository into your own local repository you can work in.

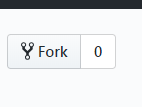
1. Setup GitHub Account

**1.1** We need to setup a GitHub account. Sign-up here for a new account: <http://www.github.com> and choose the Free plan (default).



**1.2** Next we need to fork a copy of the GitHub repository where all the lab code resides so we have our own copy of it.

Go to <https://github.com/ximtra/ansible_labs> and click on the Fork Icon



This will bring up a menu similar to below, you should see your username here, click it and it will create a fork of this repository under your own account.

Now we should be sitting at a GitHub page with a copy of the files located at something similar to [https://github.com/<username>/ansible\_labs](https://github.com/%3cusername%3e/ansible_labs) where username is whatever your GitHub username is.

This is your copy of the lab that you can update and modify and keep after the class is over, after each section we will commit our files and push it up to our own GitHub repository.

Please refer to the **Lab Connection Information.docx** for instructions on connecting to your Ansible host.

Now we need to tell git who we are and configure the CLI for it, replace the user.email and user.name with what you setup on GitHub.

git config --global user.email "[you@example.com](mailto:you@example.com)"

git config --global user.name "Your Name"

Now from our CentOS box we were given, let’s clone this repository so we can start working with it in our future labs.

git clone https://github.com/<username>/ansible\_labs

This will create a folder called ansible\_labs, we can now cd into this folder and see the directory layout

cd ansible\_labs

ls -lah

Notice how we have a .git file, this is where all the version control information is stored, we also have folders for each lab that we will be working on.

.git

lab0-git

lab1-cli-tools

lab2-ad-hoc

**1.3** Now let’s cd to the lab0-git folder and modify the test.txt file to have some data in it.

cd lab0-git

echo version data >> test.txt

Now we can tell git that we modified this file and then push it back up to our own repository that is separate from where we first cloned this repo.

git add test.txt

git commit -m ‘updated test.txt file’

Now here we are telling git to add the test.txt file to tell it we modified it, this would apply to new files as well, **git add** is the way to update or add files.

We can add multiple files at a time using **git add .** - This will only add files that we don’t already have ignored in our .gitignore file. Be very careful on using the -f flag as this will not adhere to your gitignore file and will add everything.

Then we are using the **git commit -m** which supplies a commit message so we can track what we did.

**1.4** Now Our last step now is to push this up to our repo so we can have our changes saved

git push origin master

Here we can see the output, it asks us for our GitHub username and password then updates our files once we authenticate successfully.

Username for ‘https://github.com’:

Password for ‘https://username@github.com’:

Counting objects: 7, done.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (4/4), 326 bytes | 0 bytes/s, done.

Total 4 (delta 1), reused 0 (delta 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

To https://github.com/username/ansible\_labs

dc40323..5dbfcfd master -> master

At the end of each lab you will want to run the **git add** on any new files, then **git commit** and **git push** to make sure your data is updated to the GitHub site for your profile for each lab.

If you check your GitHub page you should see the updated file now with the commit message applied.

Make sure to be careful when you are working on your own projects as you don’t want to add sensitive files to your repository. For future reading look into setting up a .gitignore to ignore adding these types of files. We have a .gitignore file already and as a test set it up to ignore files called vault, we will discuss this in future labs.

For more information on gitignore please see: <https://git-scm.com/docs/gitignore>