Monday, July 7, 2014

CDIPS Data Science Workshop: Pre-Workshop Week

Day 1: Version Control with Git and GitHub

- **Presentation: Welcome, Pre-Workshop Overview, and Introduction to Git**
- **Individual Activity: Getting Started with Git**
  - Install Git on Personal Computer:
    - http://git-scm.com/book/en/Getting-Started-Installing-Git
  - Set up your Git environment:
    - http://git-scm.com/book/en/Getting-Started-First-Time-Git-Setup
- **Walkthrough: Basic Git Commands**
- **Individual Activity: Git Practice**
  - Clone the **Public.PreWS** repository in your local environment:

```
>> git clone https://github.com/mbcole/Public.PreWS.git
```

  - Navigate into **Public.PreWS** and checkout a new branch called **private,** tracking the **origin/master** branch:

```
>> git checkout -b private origin/master
```

    - Whenever you pull on this **private** branch, your changes will be merged with the master branch in the remote repository. The purpose of this command is to set up this tracking behavior as the default. The **-b** flag signals the creation of a new branch before you check it out.
  - Within the **Public.PreWS** directory, make a directory called **Privat**e
  - Check the status of your repo:

```
>> git status
```

  - Create a **README.txt** file within **Public.PreWS/Privat**e. Check the status of your repo now.
  - Add **README.txt** to your staging area:

```
>> git add Private/README.txt
```

  - Check the status again. Now, commit the changes:

```
>> git commit
```

    - You'll be navigated to a command-line text editor to enter your commit message. Adding the **-m** flag after the commit command expedites this whole process:

```
>> git commit -m “Hello Repo”
```

  - **You've set up your first local repository!** Every once in a while, you should merge your private branch with the origin/master branch on the GitHub server:

```
>> git pull
```

    - Note: Although this exercise gives you a little experience entering git commands, it is a little strange to set up a tracking branch that you will never share with the public; you will not be pushing any of your files to our GitHub repository. If you're uncomfortable with this flow, feel free to pull to the master branch and store your pre-workshop work in a separate directory … perhaps within a team repository … stay tuned!

- **Individual Activity: Getting Started with GitHub**
    - Set up your free GitHub account
        - https://github.com/join

- **Group Activities: Shared GitHub Repositories and Version Control Exercise**
    - Form PreWS Teams of 4-5 (mixed experience levels are preferred)
    - Break the Ice!
        - Take some time to share your favorite meme with the group. Absolutely critical…
        - Be sure to come up with a PreWS team name!

https://help.github.com

http://www.maheshsubramaniya.com/

- Choose one member to host a Team Repository on GitHub
    - Host only: Create it and allow public editing (Default) : https://help.github.com/articles/create-a-repo
    - All: "git push" a personal folder to repo. Make sure it contains an equally personal **README.md** file.
    - Host only: Add a file named **team.meme.txt** to the main repo directory. It should contain ironic descriptions of each team member's favorite meme. Make sure descriptions are on separate lines, double-spaced, and sorta funny (like you didn't try too hard):

```
Team Name: <Team Name>

<Meme 1>

<Meme 2>

<Meme 3>

<Meme 4>

<Meme 5>
```

- Team Competition: Commit Challenge
  - Curious? We are too.
- Restore your team's repository to its formal glory… do whatever you want with **team.meme.txt**
- In GitHub Settings, restrict editing to **collaborators only**.
- Add All Team Members as Collaborators

Congratulations on Completing Day 1!
See you tomorrow for Day 2: Python Installation and Introduction

http://hortonworks.com/