Git & Github Worksheet

Requirements:
To complete this section, you'll need an active internet connection. Feel free to reach out to a program manager if you need further assistance.

Troubleshooting:
We understand that sometimes things may not work as expected. We're here to help. Post your errors in the chat to get help.

Setup

Note: Safely skip the installation and configuration steps if you have already installed and configured git on your computer.

Installation

1. Use the following links to download and install git for your operating system
    - Download git for OSX → http://git-scm.com/download/mac
    - Download git for Windows → http://git-scm.com/download/win
2. Confirm that git is installed by running the following command:
    - `git --version`

Question 1: What was the output from running the above command?

Configuration

If this is your first time, setting up git on your computer, there are two initial commands that you need to run. Let git know who you are: (Replace the placeholder with your real information)

```
1. git config --global user.name "FIRST_NAME LAST_NAME"
2. git config --global user.email "MY_NAME@example.com"
```

Working with a Repository

Create a new repository

1. Create a new folder on your Desktop (or anywhere)
2. Run the following command to create a new git repository
    a. `git init`

Checking out a repository

1. Create a new folder on your Desktop (again)

1. Clone any repository from https://github.com/explore
2. A repository can be cloned by running the following command
    a. `git clone https://github.com/PATH-TO-REPOSITORY.git`
    b. ***TIP***: Github usually provides a link to clone the repository on the repository's home page, almost at the top right, click the green Code dropdown button.

Understanding the Git Workflow (Using GitHub)

Cloning someone else's repository from GitHub

1. Create a new repository on https://github.com

TIP *We recommend that you sign up for a GitHub account (don't worry, it's FREE)*

    a. Here are the steps to do this:
        i. Click here, to access the Github account creation wizard: https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home

Creating your own GitHub repository

TIP: *You need to be logged into your GitHub account before starting this section.*

2. Create a new repository
    a. Click the + button on the top right of the Github page (It's left of your GitHub account profile picture)
    b. Then click, New repository
    c. Give the repository a name and description of your choice
    d. Choose whether you want your repository to be public or private
    e. Check the box: Add a README file. Click the create repository button
    f. Clone the repository you created to your computer
    g. Open the repository in your code editor

Adding and committing files

TIP*: In this section, you will use the repository you created above on GitHub.*

1. Using your code editor, create a new file called: `hello.jac` (We'll start with this empty jac file for now)

2. The next step is to tell git that we want it to keep track of this file. This means when we make changes to the contents of `hello.jac` git will be able to track and keep a record (commit) of all the changes.
   Run the following command: `git add .`
   Note, dot adds all the files (We just have one, so don't worry), instead of adding the dot (.) you can also type the name of the file: `git add hello.jac` you want to add.
3. Now that we added the file to git, let's commit the current state of the file to git, by running: `git commit -m "added hello.jac"` (Commit messages are typed between quotes, these can be anything

TIP: *its recommended to be clear and concise in your commit messages) not only does being clear makes it easier to understand, but it gives other developers you're collaborating with a clearer sense of the reason behind your commits.*

Pushing changes

In the last section, we added a file to git and commit its current state to git. In this section, we'll walk through the steps of actually pushing the committed file to github.com.

1. Make sure that your terminal is opened in the repo and run the following command:
   a. `git push origin main`

Working with branches

TIP: *Branches are used to develop features isolated from each other. The master/main branch is the default branch when you create a new repository.*

The idea here is to use other branches for development and merge(more on this later) them back to the master branch upon completion.

1. Create a new branch named "feature_hello" and switch to it using:
   `git checkout -b feature_hello`
2. To switch back to the main branch: `run git checkout main`
3. You can delete a branch by running: `git branch -d feature_abc`

   Note*: creating a new branch locally as we did above won't be available on Github (and to others) unless we push the branch to the remote repository on GitHub. Let's do that next.*

4. Create a new branch called `feature_xyz` and switch to it (Remember the command from step 1 above?).

5. Push the branch to GitHub by running: `git push origin feature_xyz`

Update and merge

Let's imagine that another developer you are working with made some changes that are now in the main branch. To update your local main branch with the new changes run:
`git pull`

Merging branches

1. Switch to the `feature_xyz` branch
2. Open the `hello.jac` file and give it the following content:

   https://github.com/Jaseci-Labs/jaseci_bible/blob/main/assets/code/fam.jac

3. Commit and push changes
4. Now, merge the feature_xyz branch into the main branch:
   a. Switch to the main branch again
   b. Then, run `git merge feature_xyz`

***Note**: In some cases, merging changes from one branch to another is not always possible and results in conflicts.*
When conflicts happen, you are responsible for merging those conflicts manually by editing the files shown by git.

After changing, you need to mark them as merged with:
`git add <filename>`

Before merging changes, you can also preview them by using
`git diff <source_branch> <target_branch>`

Tagging (To keep in mind)

It's recommended to create tags for software releases. This is a known concept. You can create a new tag named 1.0.0 by running:
`git tag 1.0.0 ib2e1d63ff`
The highlighted string stands for the first 10 characters of the commit id you want to reference with your tag.

Let's move to the next section to see how we can get that commit id using git log

Log (To keep in mind)

In its simplest form, you can study repository history using… git log

You can add a lot of parameters to make them look like what you want.
Let's look at some examples, and you can try these on your own as well

1. To see only the commits of a certain author:
   `git log –author=bob`

2. To see a very compressed log where each commit is on one line:
   `git log –pretty=oneline`

3. Or maybe you want to see an ASCII art tree of all the branches decorated with the names of tags and branches:
   `git log - -graph - -oneline - -decorate - -all`

4. See only which files have changed:
   `git log –name-status`

5. These are just a few of the possible parameters you can use. For more see:
   `git log –help`