**GitHub & Git Training Document**

# Objective

GitHub allows our team to store all our programming scripts on a **private** centralized repository. Prior to setting this up, all our programming scripts were stored on the shared drive, which led to version control issues. In addition, GitHub can facilitate collaboration and script sharing across departments and IPOs.

# Storage Structure

**Organization: cipo-research**

**Repository ...**

**Repository 1**

**(Private)**

**Repository 3**

**(Private)**

**Repository 2**

**(Private)**

|  |  |
| --- | --- |
| **Repository** | * A GitHub repository can be imagined to be a folder consisting of all scripts and other related files for a particular project. * A repository can be private or public. **It is recommended that all repositories for CIPO be made private.** * Public repositories do not require an account to view or download while private requires an account and access approval to view/edit. * Single files can be made available to non-GitHub users through a public repository (less secure) or to GitHub users through private repository upon access |
| **Organization** | * The GitHub organization serves as the homepage for CIPO and contains an overview of all our repositories. |

# Tools to be installed:

The following tools need to be installed on our computers in order to facilitate interaction between GitHub’s centralized repository and our computer’s local repository:

1. GitHub Desktop:

<https://desktop.github.com/>

1. GitBash (installation guide has been included as a separate PDF in this repository):

<https://git-scm.com/downloads>

**Note:**

* These tools can be downloaded on our work laptops without creating IT requests.
* Although either GitHub Desktop or GitBash can be used, it is generally recommended that analysts use the command-line interface that GitBash provides versus the GUI that GitHub Desktop provides.
* It was discovered that IT has blocked port 22 which GitBash uses to interact with GitHub’s centralized repository. As a result, only GitHub Desktop can be used on our work laptops.

# Documentation for GitHub

## Creating a new Private Central Repository

1. Go to [https://github.com/cipo-research](https://github.com/cipo-research%20) .
2. Click on the “New” button located on the right. It will leading you to the following page.

A screenshot of a cell phone

Description automatically generated

1. Enter an appropriate repository name under the “Repository Name” field.
2. Give a brief description for the repository in the Description field.
3. Select the “Private” option.
4. Check “Initialize this repository with a README”.
5. Press “Create Repository”.
6. You have successfully created a Private Central Github Repository. Please include a detailed explanation of the repository in the “README.md” file.

A screenshot of a social media post

Description automatically generated

## Adding Team Members to the Organization

1. Go to the Organization’s Homepage and select ‘People’.

A screenshot of a computer

Description automatically generated

1. A screenshot of a computer

   Description automatically generatedSelect ‘Invite Member’.
2. A screenshot of a cell phone

   Description automatically generatedSend invitation by typing the username, email, or name of the individual you wish to add to the organization.

## Inviting Outside Collaborators to a Repository

1. Go the home/main page of your repository on GitHub and click on “Settings”.A screenshot of a cell phone

   Description automatically generated
2. Click on “Manage Access”. If prompted to enter your password then, enter your password.

A screenshot of a social media post

Description automatically generated

1. A screenshot of a cell phone

   Description automatically generatedClick on “Invite a Collaborator”.
2. A screenshot of a cell phone

   Description automatically generatedSearch the user you wish to add by name, username, or e-mail and then select add. The person who you wish to add has to be a GitHub user.
3. A screenshot of a cell phone

   Description automatically generatedWait until the person who you wish to add accepts their invite (via the e-mail they signed in with).
4. A screenshot of a cell phone

   Description automatically generatedOnce the invitation has been accepted by the user, they would been labeled as a “Collaborator”. Please provide appropriate access rights to the user using the dropdown menu located on the right.

# GitBash commands

**NOTE:** When working with Git Bash do not use Ctrl + V and Ctrl + C to copy and paste as that will cause errors. Instead to Copy using Ctrl + Insert and to Paste use Shift + Insert.

## Creating a Local Repository for Users to sync with the Centralized Repository

1. Create a new folder that you wish to store your work in.
2. Right click on the folder and select “Git Bash Here”. You should be greeted with a command prompt.
3. To initialize this folder as a Git Repository, in the Git Bash window type the command *“git init”.* Your should be returned with a message that says “Initialized empty Git repository in *FILE\_PATH*”.
4. (Optional) To ensure git is working on your system, a .git folder must have been created, to see this folder, go to “View” on the top status bar, then “Options” then “Change folder and search options” then “View”, and check “Show hidden files, folders, and drives”.
5. Find the Central Repository you wish to connect it to on GitHub, and go to the main page of that repository.
6. Click the “Clone or Download” button and copy the provided URL.
7. On command prompt use the command “*git remote add origin URL*” to connect your local repository to the central repository.
8. Now you have a local repository (the folder) that is capable of syncing with the Central Repository.

## Creating a Local Repository of a previously created Centralized Repository (Cloning)

1. Go to [https://github.com/cipo-research](https://github.com/cipo-research%20) and identify the repository that you wish to store on your system.
2. Once you are on the home page of the repository click the “Clone or Download” button and copy the URL of the GitHub repository.
3. Go to the folder you wish to store the repository and right-click in a blank area.
4. Select “Git Bash Here” and type in the command “*git clone URL*” replacing the URL with the one you just copied.
5. Hit enter and the repository should load in the folder.

## How to update the Central Repository with your work and locally save it

1. Find the local repository/folder on your computer that is connected to a GitHub repository
2. Right-click on the folder and select “Git Bash Here” to open up the command prompt.
3. Check which files need to be updated to the Central Repository (i.e. have changed) by using the git status.
4. Any files that are labelled in red need to be saved locally through a command called git commit, but before we are able to commit, we have to add these files to a queue. Committing allows the user to keep a log of the changes they are making to the files and it is a good practice to do it A close up of a logo

   Description automatically generatedregularly.
5. As you can see the terminal returns a list titled ‘untracked files’ of files that have not been updated or accounted for.
6. To track these files, we must use the git add command and specify the name of the file(s) and their file type after that. For example, in this case, if you wanted to add the file ‘Applicant country v2.the’ you have to enter the command git add ‘Applicant country v2.the’.
7. After adding the file to the queue, you can now verify this by using the git status command again. This time you will notice that the file you added is now labelled in green (if not repeat the previous step). In this case, ’Applicant country v2.the’should be in green text.

A screenshot of a cell phone

Description automatically generated

1. Suggesting that the that file is being tracked and ready to be committed, and we can repeat this process for .DS\_Store as well.
2. An alternate way of quickly adding multiple files would be to do ‘git add *.*’ (including the period) this will add all untracked files into the queue.
3. If you wish to remove a file from the queue you can use the command git rm --cached FILE\_NAME .
4. After all the files that you wish to save are in green, you can commit/save your changes by performing the git commit -m ‘your\_message’ command. NOTE: it is important you include -m and a brief message describing the change as it is not only a good practice for documentation, but a message is required for the command to work.
5. Now your work has been locally saved through a commit, you can either choose to continue working and updating your or proceed to publish your work to the Central Repository.

## To publish/push your work

Note: before you are able to push you must have a GitHub account and have permission from the repository admin.

1. Once you have committed your work, you can push you work to the Central Repository by using the command “*git push -u origin master*” or if you are using a clone repository simply use “*git push*”.
2. You may be prompted to enter your password to GitHub to authenticate your identity.
3. You can if your work has been updated by visiting the repository on the GitHub Website and viewing your Central Repository.

## How to update your local repository to be the latest version of the Central Repository

1. Find the local repository/folder on your computer that is connected to a GitHub repository
2. Right-click on the folder and select “Git Bash Here” to open up the command prompt.
3. Enter the command git pull to download changes that were made to the repository. If any changes exist, then it will incorporate those changes in local repository.

## How to Revert to an Older Version

1. Go to the local repository folder, right-click, and select “Git Bash Here”
2. In the command prompt enter “*git log FILE\_PATH*” to view the version log of a specific file.
3. Once you have identified the version which you would like to revert to, copy the hash value of the commit.
4. Then to revert the file use the command “*git checkout HASH\_VALUE FILE\_PATH* ” and then “*git commit* ” .

## Summary

1. *git status:* can be used to check which files need to be synced with the central repository.
2. *git pull:* can be used to sync your local repository with most recent version of the Central Repository.
3. *git push:* can be used to send out changes that you have made to local repository and update the Central Repository.
4. *git commit:* can be used to save the changes you made the local repository.