Git Basics

Preface

Version control is the practice of tracking and managing changes to software code. Version control software keeps track of every modification to the code. Developers can turn back the clock and compare earlier versions of the code to review previous changes, fix mistakes, or implement changes. By far, the most widely used version control software in the world today is Git. Other version control tools such as SVN exist but are far inferior to the ease of use and functionality of Git.

Install Git

Note: When installing Git for Windows, keep all the settings as default. The password setting can be set as none, as this will not make any security difference.

For **Windows**-based distributions (OS), use the following link and download the most recent version: Git

For **Linux** users, use your default package manager. For example:

- `yum` for Linux, UNIX, RedHat distributions
- `apt` or `apt-get` for Debian/Ubuntu distributions

Install Git on Linux

To install Git using Yum package manager:

```
$ sudo yum install git
```

Configure Git

Do you have a GitHub account? If the answer is no, please click the Link and create a GitHub account. You may want to use your personal email address to create your GitHub and then link your QUT example@connect.qut.edu.au email address to the account to gain access to free education features such as GitHub co-pilot, which is an incredibly useful tool for working in your IntelliJ workspaces.

Once you have your GitHub account complete the following in your Git Bash terminal, or for Linux users in your CLI.

To set your username globally in Git, use the command:

```
$ git config --global user.name "<username>"
```

Similarly, to set your email address globally, use:

```
$ git config --global user.email "<email>"
```

Key Generation

This process will allow you to make changes to the GitHub, through the terminal or through IntelliJ or any other IDE (Integrated Development Environment i.e. VSCode). Arguments for ssh-keygen function: -t -b -f -C <allows you to input a comment in the “” field>. Unless you absolutely know what you are doing, just go with the first option and do not change anything. One reason is key security and there are funny things, like for example GitHub does not recognize dsa as a valid key type. So, changing these elements is unnecessary. It will ask you when you generate your key whether you would like a password simply just click enter twice to save no password, and when it asks you about the default directory just click enter as well. This will set you ssh-key to its default settings and directory.

Generating a key if you haven't done it before (do this in Git Bash):

```
$ ssh-keygen -t rsa -b 4096 -C "your comment i.e. key name"
```

If you want to specify a specific name for your key, follow these steps:

```
$ ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_<keyname> -C "your comment i.e. key name"
```

You can then execute the following commands:

- Use the following command to navigate to the ssh root directory:

```
$ cd /.ssh
```

- To list all items in the ssh root directory, use the command:

```
$ ls -l
```

You should be able to view id_rsa and id_rsa.pub in the directory listing.

- Display your public key in the Command Line Interface by executing the command:

```
$ cat id_rsa.pub
```

Add Key to GitHub

For the point of this project, we will have a central GitHub owned by one user, thus the public key will be provided to them. However, for your personal GitHub you should add your key. This will mean you can upload anything, anytime to any of the repositories that you are a part of or own. This will also give you a chance to practice.

1. At the GitHub home page, click on your **profile** in the right-hand corner.
2. Locate **settings** towards the pullout tab on the right-hand side of the screen.
3. Under **access** on the left hand side of the screen click the section that says **SSH and GPG keys**.
4. On the right hand side of the screen click the button **New SSH key**.
5. Input a title of your choice such as “Parkers-XPS15”.
6. Select **Authentication Key** as the **Key type**.
7. Input the entire contents of your id_rsa.pub file (or whatever name you chose) into the Key section on GitHub.
8. Finally click the **Add SSH key** button in the middle of the screen.

Post-face: That is initialisation and setup completely done, you are now ready to go. Below will be some common and crucial commands and also explanations of how they work / might be used in a production context. For Windows users I strongly recommend you learn these commands for Git Bash as they give you better control over your work. But you are also welcome to use your IntelliJ / VSCode GitHub extension. Either way, this is a good refresher on the functionality of Git.

Using Git

Note: All the work below will be done in a terminal instance, for Linux distribution users this will be your native CLI, for Windows distributions this will be Git Bash.

Create a local workspace:

- Navigate to the root directory.

- Create a directory named "Projects" in the "Documents" folder.
- Change to the "Projects" directory.
- Clone your files from GitHub using the command:

General Git Workflow

Git workflow follows a fairly standard process of **Initialising, Cloning, Pulling, Adding, Committing, and Pushing.** There are other aspects of Git workflow which are useful to understand, but for simplicity's sake sticking to the basics at the start is the best way to learn and pick up the skill.

Note: A repository can refer to either the local repository or remote repository, depending on the stage and context of workflow being discussed. Local repository refers to the tracked files in your local directory, i.e. /data/project/test_project. Remote repository refers to the tracked files that reside on a server, i.e. GitHub.

Initialisation is the creation of a new local repository. This step can also be performed remotely on GitHub. This action is performed once per repository:

```
$ git init
```

Cloning is the action of creating a local copy of a repository from a remote server. This action is typically performed once. But can be performed many times if multiple versions of that repository need to reside in different locations on your machine:

```
$ git clone git@github.com:username/Reponame.git
```

Pulling is the action of updating changes between the local repository and the remote repository. This action is performed regularly to ensure that you are working from an up-to-date repository.

```
$ git pull
```

Adding is the action of adding a file to your local repository. This action is performed regularly to ensure that you are preparing files that have been updated to be committed.

```
$ git add <filename>
```

```
$ git add * // Adds all files within the directory.
```

Committing is the action of confirming the changes made to local files in the local repository. This action is performed regularly to ensure you are updating the local repository.

```
$ git commit -m “<commit message>”
```

Pushing is the action of sending the commit changes to your remote server repository. This action is performed regularly to ensure that you are updating the remote server repository.

```
$ git push -u origin
```

Git Workflow Example

In this example we can use a repository named **Sandbox** to practice Git workflow.

The general workflow we will follow is:

1. Creating a new file and adding it to the local Git repository.
2. Committing the file to the local Git repository.
3. Pushing the file to the remote Git repository.

Understand that the workflow below is Git in its very basic form, there are more parts of Git such as CI/CD integrations, fetch, merge, issues, deployments, forks ect. If you would like to learn more about Git I would definitely recommend it. It is the most common form of version control and is used in nearly every company across the world. You will find it used in places like Google, Microsoft, Apple, Government and many, many more.

I will guide you through the process of setting up a directory and initializing a repository for your project. Please follow the steps below:

1. Create a directory named "Sandbox" within the "Project" folder in your Documents:

```
$ mkdir ~/Documents/Project/Sandbox
$ cd ~/Documents/Project/Sandbox
```

2. Initialize the repository:

```
$ git init
```

3. Create a test file named "test.txt" within the Sandbox directory:

```
$ cat > test.txt
>>> This is a test file.
```

After typing your text, press **CTRL+D** to close the file.

4. Add the test file to your local repository:

```
$ git add *
```

5. Make a commit to save the changes to the test file:

```
$ git commit -m "Committing changes to test file version 1.0"
```

6. Push the changes to your remote repository:

```
$ git push
```