Git Resources

Configuring author and email

```
# home directory
$ pwd
$ mkdir git-fast
$ cd git-fast
$ pwd
$ git config --global --list
$ git config --global user.name "Bibhash Roy"
$ git config --global --list
$ git config --global user.email "bibhash@whitepeaksoft.com"
$ git config --global --list

# vi is the editor, ~ is the home directory
$ vi ~/.gitconfig
$ cat .gitconfig
```

Initializing an empty repository

```
$ pwd
$ mkdir myRepoFromScratch
$ cd myRepoFromScratch
$ pwd

# notice the creation of .git directory
$ git init

# "ls -al" command lists all objects, including hidden
$ ls -al

# without using any editor
$ echo "this is my first file in empty repository" >> firstFileUsingEcho.txt
$ ls
$ cat firstFileUsingEcho.txt

# create the file and enter some contents
$ vi  secondFileUsingVI.txt
$ ls
$ cat secondFileUsingVI.txt
```

Existing unversioned project to a repository

```
$ pwd

# copy the downloaded file
$ cp /f/Bibhash/Downloads/initializr-verekia-4.0.zip .

# unzipping will create a folder called "initilizr"
$ unzip initializr-verekia-4.0.zip
$ ls -al

# remove copied file
$ rm initializr-verekia-4.0.zip

# rename directory
```

```
$ mv initializr/ myRepoFromExistingSource

# now we need to add source control to the unversioned repo "myRepoFromExistingSource"
$ cd myRepoFromExistingSource

# Initialized existing source
$ git init

# .git folder can be seen with ls -al
$ ls -al
```

Command Summary - configuration & repository creation

```
# Typically, you'll want to use the --global flag to set configuration options for the current
user.
# Define the author's name to be used for all commits by the current user.
$ git config --global user.name <name>

# Define the author's email to be used for all commits by the current user.
$ git config --global user.email <email>

# For listing all the global configuration at the user level
$ git config --global --list

# Transforms the current directory into a Git repository; the command is the same for creating
repo from ..contd..
# scratch or convert an existing unversioned codebase to a git repository
$ git init
```

Command Summary - Accessing Git Help system

```
# general help
$ git help

# lists sub-commands
$ git help -a

# lists concept guides
$ git help -g

# read about a specific sub-command
$ git help <command>

# read about a specific concept
$ git help <concept>
```

Copying a GitHub repository

```
$ cd <project_directory>
$ git clone https://github.com/<user>/<example_project>
$ cd example_project
$ ls -al
```

Adding your changes

```
# home directory
$ pwd
$ cd git-fast
$ ls
```

```
$ cd myRepoFromScratch
$ ls

# create a file and add some content
$ vi demoFile1
$ cat demoFile1
$ git status
$ git add demoFile1
$ git rm --cached demoFile1
$ git status
```

Committing changes

```
$ git commit -m "message goes here"
```

How to check your repo status

```
# cd to /c/Users/HP-PC/git-fast/myRepoFromScratch
$ pwd
$ ls

# start with a clean working directory
$ git status

# create file and add some contents such as "I want to shed 15 kg 5 weeks"
$ vi weightLossChart

# create file and add some contents such as "I want to add more green vegetables to my diet"
$ vi dietChart
$ git status

# same as "git status" since "long" option is the default one (compare output)
$ git status --long
$ git status -s                                  # status "??" for untracked file
$ git add weightLossChart
$ git status -s                                 # status "A"
$ git commit -m "1st commit for weightLossChart"
$ git status -s
$ vi weightLossChart                    # make some changes to the file
$ git status -s                                 # status "M"
$ git add weightLossChart
$ git status -s                                 # status "M"
$ git commit -m "2nd commit for weightLossChart"
$ git status -s
$ mv weightLossChart weightLossChart2
$ git status -s                              # status "D"
```

How to check commit history

```
# displays the entire commit history using the default formatting
$ git log

# oneline condensed view of each commit history
$ git log --oneline

# Only display commits that include the specified file
$ git log <file>

# Show only commits that occur between <since> and <until>. Both arguments can be either a commit
ID ...contd.
```

```
# a branch, name, HEAD, or any other kind of revision reference.
$ git log <since>..<until>

# Limit the number of commits by <limit>. For example, git log -n 3 will display only 3 commits.
$ git log -n <limit>
```