**Code Repository**

Optum

Student Exercise Manual

Contents

[Exercise 1: Introduction 3](#_Toc519254245)

[Exercise 2: Initializing Git Repository 5](#_Toc519254246)

[Exercise 3: Clone the Git Repository 6](#_Toc519254247)

[Exercise 4: Basic Git Operations 7](#_Toc519254248)

[Exercise 5: Basic Git Workflow 9](#_Toc519254249)

[Exercise 6: Merge Conflict Resolution 10](#_Toc519254250)

[Exercise 7: Git Pull Request 11](#_Toc519254251)

## Exercise 1: Introduction

**Overview**

**Time: 20-30 Minutes**

In this exercise, you will:

* Ensure that Git is installed on your system and be able to get familiar with the user interface.

**Lab Goals:**

1. Install Hadoop in Windows, Mac or Linux.

2. Start the basic git command line.

**STEP 1) Install Git.**

Open the following link in your browser:

<http://git-scm.com/book/en/v2/Getting-Started-Installing-Git>Follow the instructions for your operating system.

**STEP 2) Specifically for Windows**

* Since most Linux or Mac developers are usually used to working on the command line, this step concentrates on Git for Windows.
* Open this link: <http://git-scm.com/download/win>
* And install the executable that will be downloaded on your computer.
* ACCEPT ALL THE DEFAULTS. This will assure that every student has the same installed configuration.
* After you are done with the labs, you will be able to experiment with various installed options.

**STEP 3) Open git command line in a directory**

* If you are in Windows, prepare a new 'git-labs' directory in the file browser, then right-click on that directory and choose "**Git-Bash**" menu. This will open the git command line.
* If you are on Mac or Linux, just open a terminal in the newly-created **'git-labs**' directory the Git command will be available, since you have installed it in step 1.

**STEP 4) Experiment with git Command**

Type

**git**

Then hit the **ENTER** key.

Observe git help.

Type

**git status**

You should get 'Not a git repository'.

Type

**pwd**

Note that you will see the working directory.

Deduce from here that you are in a fully implemented Linux environment. Indeed, this is a Cygwin implementation. Experiment with Linux commands, such as 'ls'.

## Exercise 2: Initializing Git Repository

**Overview**

**Time: 15 Minutes**

In this exercise, you will:

* Learn how to initialize a git repository, as well as how to deal with objects.

**Lab Goals:**

1. Initialize git repository.

2. Get familiar with the three git areas: working, staging, commit.

3. Get a feel for git objects.

**STEP 1) Initialize the git repository.**

The following steps are done in Git Bash if in Windows, and on Terminal, if in Mac/Linux:

* mkdir test
* cd test

Note that at this time 'git status' gives you an error message.

Now, initialize the git repository:

$ git init

Initialized empty Git repository in /tmp/test/.git/

Investigate the .git area.

Explain the directories and object found in it.

**STEP 2) Experiment with getting objects from the git store.**

$ find .git/objects

.git/objects

.git/objects/info

.git/objects/pack

Explain the above command.

* find .git/objects –type f

Explain the results of this command.

## Exercise 3: Clone the Git Repository

**Overview**

**Time: 15 Minutes**

In this exercise, you will:

* Learn how to clone a git repository, as well as how to use the Git UI.

**Lab Goals:**

1. Initialize the git repository.

2. Install Git UI.

3. Clone on the command line and in the UI.

The instructor will prepare a common repository on GitHub, and it will be used for training. (If the student group has an internal repository already, the labs will be done in it).

**STEP 1) Clone the git repository.**

The following steps are done in Git Bash if in Windows, and on Terminal, if in Mac/Linux

$ git clone <url>

(The instructor will provide the URL, but we will call it git‑labs for now)

$ cd git-labs

Investigate the contents of the git-labs directory.

**STEP 2) Install your favorite git UI.**

This may be an Eclipse plugin, IDEA or NetBeans plugin., the UI that comes with Git install, SmartGit or any other client of your choice.

If you don't plan to use a UI, you can skip this step.

**STEP 3) Clone the git-labs project in the git UI.**

Follow the instructions of the particular UI you have chosen to work with.

## Exercise 4: Basic Git Operations

**Overview**

**Time: 15 Minutes**

In this exercise, you will:

* Learn the basics of editing the files and committing the changes to git.

**Lab Goals:**

1. Change the file, stage it, and commit it to git.

2. Work with branches.

3. Push and pull the changes.

**STEP 1) In the git-labs, create your own branch.**

git branch <your-name>

*In windows* ***git branch <your-name>*** *is not working*

*Error*

*$ git branch test*

*fatal: Not a valid object name: 'master'*

Working command to create a branch

$ git checkout -b test

Switched to a new branch 'test'

git checkout <your-name>>

**STEP 2) Edit the file and commit the changes.**

* Edit a file (any file) in the directory git status.
* Explain what the git status is telling your git add <file>
* Add the file to staging git commit.
* Commit the staged file.
* Explain every step in terms of git and in terms of the working directory / .git

**STEP 3) Push the change to the remote.**

git push

**STEP 4) Pull the changes from the remote.**

git pull

**STEP 5) Bonus**

Creatively alternate edits, commits, pushes and pulls in various branches. Observe the interaction between yours and other students’ changes.

**STEP 6) Merge (the instructor will tell one student to merge his/her changes into dev)**

merge dev

git status

Explain the results.

## Exercise 5: Basic Git Workflow

**Overview**

**Time: 15 Minutes**

In this exercise, you will:

* Learn the basics of normal git workflow, by imitating a developer's day.

**Lab Goals:**

1. Imitate the developer’s morning work.

2. Imitate an emergency fix.

3. Go back to the normal work.

By now, you should have sufficient knowledge of basic git commands, in order to re-enact the scenario below. Please refer back to the slides for more details.

**STEP 1) Morning**

* Create your 'new-work' branch.
* Work in this branch for a while.

**STEP 2) Lunch**

* Emergency fix in the 'master' is required.
* Checkout the master branch.
* Create a 'hotfix' branch.
* Perform and test the fix.
* Merge the fix into master.
* Test, push the master to remote.
* Delete the 'hotfix' branch.

**STEP 3) Afternoon**

* Go back to the 'new-work' branch.
* Checkout the 'new-work' and continue working.

## Exercise 6: Merge Conflict Resolution

**Overview**

**Time: 15 Minutes**

In this exercise, you will:

* Learn to resolve merge conflicts.

**Lab Goals:**

1. Create a conflict.

2. Resolve the conflict.

3. Commit the changes.

**STEP 1) Create a conflict.**

The easiest way to create a conflict is by cloning your project to another directory, then editing the same file in the same place in both directories, committing in one directory and pulling the changes in the other.

Bonus: Create a conflict in a different way (hint: merge branches that have conflicting edits).

**STEP 2) Resolve the conflict.**

Manually open the file and edit it until the conflict is satisfied.

Tell git that you are done with the changes.

git add <file>

**STEP 3) Push the changes.**

git commit -a

Note the message that will appear.

Explain that message.

git push

**STEP 4) Configure your preferred merge resolution tool in git.**

Use the merge resolution tool to resolve the conflict.

## Exercise 7: Git Pull Request

I **Overview**

**Time: 15 Minutes**

In this exercise, you will:

* Learn how to use the git pull request.

**Lab Goals:**

1. Create a developer with read-only access.

2. Fork the project, edit it, perform the pull request.

3. Merge the changes.

**STEP 1) Create a developer role with read-only access.**

Do this on GitHub.

**STEP 2) Fork the project.**

Fork the project on GitHub.

Edit the forked project.

Commit the changes and issue the pull request.

**STEP 3) Accept and merge the pull request changes.**

You will receive an email from GitHub.

Follow the instructions in this email.

Follow these instructions: [https://help.github.com/articles/using‑pull‑requests/](https://help.github.com/articles/using-pull-requests/)