Tutorial - GIT

Grupo de Linux da Universidade de Aveiro

2016

1 Introduction

Welcome to this tutorial. This is a brief introduction to version control software and GIT. This tutorial is separated in three parts:

- Introduction
- First steps
- Advanced commands

Introduction - we will talk about Version Control Systems and GIT.
First steps - we will talk about GIT's basic commands.
Advanced commands - we will talk about GIT's advanced commands.

1.1 What is a version control system?

A version control system (**VCS**) is a software that allows you to manage the changes in files, while retaining previous changes. It will also allow you to work with others more effectively, since it will track the changes for you.

1.2 Why use version control system?

Instead of using other kinds of systems, you can use **VCS** to track the changes in code and in file structure of your project. Just this aspect will allow you to focus in code production instead of always worrying if you have the right files or what others have changed in the files.

1.3 What is GIT?

GIT is one of the main **VCS** used for software development. **GIT** main goals are:

- Speed
- Data integrity
- Distributed
- Non-linear workflow

2 First steps

This is the beginning of a great journey for you. You will learn the basics of GIT, which will help until the end of your days as developer.

2.1 Create a local repository

First of all, lets create a new repository so you that you can start tracking all the changes in your project.

```
git init
touch README.md
git add README.md
git commit -m "Initial commit"
```

Lets dissecate each command:

- **git init** - this command will initialize a local git repository in the folder you're in.
- **touch README.md** - will create an empty file called README.md.
- **git add README.md** - will make your local GIT repository start tracking your file.
- **git commit -m "Initial commit"** - will set the actual state of the repository as the first snapshot taken.

2.2 Synchronize with a remote server

2.3 Adding files

Lets create a few other files so we can start tracking them with GIT.

```
touch file_1
touch file_2
mkdir folder_1
touch folder_1/file_1
touch folder_1/file_2
```

Lets check what is the state of the local GIT repository with **git status**.

```
On branch master
Untracked files:
   (use "git add <file>..." to include in what will be committed)

file_1
file_2
folder_1/

nothing added to commit but untracked files present (use "git add" to track)
```

There are two ways of start tracking these files. You can add all files with one command or you can add them one by one. If you know that you want to add them all, you can use this command:

```
git add .
```

If you don't want to track all files, you can add them individually like this:

```
git add <file>
```

In this case, we want to add all files, so we will use **git add .** and know lets check the state of the local repository with **git status**.

```
On branch master
Changes to be committed:
   (use "git reset HEAD <file>..." to unstage)

new file:  file_1
new file:  file_2
new file:  folder_1/file_1
new file:  folder_1/file_2
```

2.4 Making commits

Now that we have added some new files and started tracking them in our repository, lets save this state. To save this changes in the repository, we need to make a commit of the changes. To commit the changes, we will use the following command:

```
git commit -m "A message relevant to the changes made in this commit"
```

Know that we know which command to use, lets use it but don't forget to add a meaningful message, so that when others check the commits to the repository can understand what or why you did those changes.

2.5 Checking the log

Lets check the log to see the changes we have made to our local repository. To check these changes, we will check the local repository log and to do that we must run the following command:

```
git log
```

After running this command, we will get an output similar to this:

```
commit 3070d3621b60b5ebe46b2d58ad6be2537069d79d
Author: John Doe <john.doe@example.com>
Date:   Tue Nov 15 15:44:20 2016 +0000

   Added initial files

commit 5814279838033e7b0f14a620c73202f52f11cf99
Author: John Doe <john.doe@example.com>
Date:   Tue Nov 15 15:26:39 2016 +0000

   Initial commit
```

What can we check in the log? Well, we can see the commits that were made to the repository, when were they made and the message that the author of the commit wrote for that change.

Now lets say you want to check what a specific commit made. You can verify this with this command:

```
git show <commit hash>
```

Where the **<commit hash>** is the number that appears after the commit word when using the **git log**.

2.6 Updating the remote server

2.7 Creating a branch

2.8 Merge vs Rebase

2.8.1 Merge

2.8.2 Rebase

2.8.3 Difference between them

2.9 Reset

3 Advanced tricks

3.1 Cherry-pick

3.2 RefLog

3.3 Bisect

3.4