Git tutorial

LeeMoo

May 6, 2015

Contents

0.1 Introduction

What does it take to be professional? Is it about how much you know? About knowing your topic inside out? Of course it is. But it's only one part of the equation. The other part is aobut using the right tools and cultivating the right habits. You won't find a five-star chef who works with a cheap knife - he knows that he will produce better results and work safer with the best toll for the job. Similarly, you won't find a professional tennis palyer who doesn't train his endurance - he knows that tennis isn't just about hitting the ball across the net. Just the same, you won't find a top programmer, web developer, or web designer who doesn't use version control. they know that things go wrong all the time in our industry and therefore prepare. They know that collaboration must be as safe easy as pobbile because teamwork is paramount in our industry. They know that, when the're working sloppy, they'll have to pay the bill in the end. Don't mind a little bit of sweat to learn version control. It's a big step on your way to becoming a better professional.

About This Book The goal of this book is to get you started with version control and Git as quickly and easily as possible. Unlike other books about this topic, this one doesn't requre a master's degree in computer science to read it. It's aimed at beginners of programming, at designers, at project managers... It tries not to require too much prior knowledge on the technicla side. It tries to

go slowly.

That being said, Git and version control in general remain highly technical topics. I can't spare you all of this, but I'll try to explain workflows backgrounds thoroughly and provide a lot of real-world examples.

Since everyone comes with his own, unique background, it's hard to determine a common starting point ofr everybody. For this reason, I have provided various basic topics in the appendix:
1. In case you're still unsure if you should use Git as your version control system, Appendix D: Why Git? might be worth a look.
2. If you're about to switch from SVN, you might find Appendix C: Switching from Subversion to Git helpful. It gives you an overview of the differences between the two systems.
3.Should you need an introdution to working on a command line interface, you should definitely take a look at Appendix B: Command Line 101.
Have fun learning Git.

Part I

Part1 - The Basics

0.2 What is Version Control?

You can think of a version control system (short:‘" Vcs ") as a kind of ‘"database". It lets you save a snapshot of your complete projet at any time you want. When you later take a look at an older snapshot (let's start calling it ‘"version "), your VCS shows you exactly how it differed from the previous one.
Version control is independent of the kind of project /technology /framework you're working with:
1. It works just as well for an HTML website as it does for a design project or an iPHone app.
2. It lets you work with any tool you like; it doesn't care what kind of text editor, graphics program, file manager or other tool you use.
Also, don't confuse a VCS with a backup or a deployment system. You don't have to chagne or replace any other part of your tool chain when you start using version control.
A version control system records the changes you make to your project's files. This is what version control is about. It's really as simple as it sounds.

0.3 Why Use a Version Control System?

There are many benefits of using a version control system for your projects. This chapter explains some of them in detail.
Collaboration
Without a VCS in place, you're probably working together in a shared folder on the same set of files. Shouting through the office that you are currently working on file ‘"xyz" and that, meanwhile, your teammates should keep their fingers off is not an acceptable workflow. It's extremely error-prone as you're essentially doing open-heart surgery all the time: sooner or later, someone will overwrite someone else's changes.
With a VCS, everybody on the team is able to work absolutely freely - on any file at any time. The VCS will later allow you to merge all the changes into a common version. There's no question where the latest version of a file or the whole project is. It's in a common, central place: your version control system.
Other benefits of using a VCS are even indenpendent of working in a team or on your own.
Storing Versions (Properly)
Saving a version of your project after making changes is an essential habit. But without A VCS, this becomes tedious and confusing very quickly.
1.How much do you save? Only the changed files or the complete project? In the first case, you'll have a hard time viewing the complete project at any point in time - in the latter case, you'll have huge amounts of unnecessary data lying on your hard drive.
2. How do you name these version? If you're a very organized person, you might be able to stick to an actually comprehensible naming scheme (if you're happy

with ‘” acme-ine-redesign-2015-11-12-v23”). However, as soon as it comes to variants (say, you need to prepare one version with the header area and one without it), chances are good you’ll eventually lose track.

3. The most important question, however, is probably this one: How do you know what exactly is dfferent in these versions? Very few people actually take the time to carefully document each important change and include this in a README file in the project folder.

A version control system acknowleges that there is only one project. Therefore, Three’s only the one version on your disk that you’re currently working on. Everything else - all the past versions and variants - are neatly packed up inside the VCS. When you need it, you can request any version at any time and you’ll have a snapshot of the complete project right at hand.

Part II