<<Git   >>

<<Git   >>

13   ISBN   9787564122607

10   ISBN   7564122609

        2010-6

        310

                PDF

        http://www.tushu007.com

<<Git   >>

AudienceWhile some familiarity with revision control systems will be good background material,a reader who is not familiar with any other system will still be able to learn enoughabout basic Git operations to be productive in a short while. More advanced readersshould be able to gain insight into some of Git's internal design and thus master someof its more powerful techniques.The main intended audience for this book should be familiar and comfortable with theUnix shell, basic shell commands, and general programming concepts.Assumed FrameworkAlmost all examples and discussions in this book assume the reader has a Unix-likesystem with a command-line interface. The author developed these examples on Debian and Ubuntu Linux environments. The examples should work under other environments, such as Mac OS X or Solaris, but the reader can expect slight variations.A few examples require root access on machines where system operations are needed.Naturally, in such situations you should have a clear understanding of the responsibilities of root access. Book Layout and OmissionsThis book is organized as a progressive series of topics, each designed to build uponconcepts introduced earlier. The first 10 chapters focus on concepts and operationsthat pertain to one repository. They form the foundation for more complex operationson multiple repositories covered in the final six chapters.If you already have Git installed or have even used it briefly, you may not need theintroductory and installation information in the first two chapters, nor even the quicktour presented in the third chapter.

<<Git   >>

Git

LinilusTorvalds   Linux

Git

Git   Git

<<Git   >>

Jon Loeliger   Linux   U-Boot
Git
Linux World   Git   Linux Magazine   Git

<<Git   >>

It's important to see Git as something more than a version control system: Git is a content tracking system. This distinction, however subtle, guides much of the design of Git and is perhaps the key reason Git can perform internal data manipulations with relative ease. Yet this is also perhaps one of the most difficult concepts for new users of Git to grasp, so some exposition is worthwhile. Git's content tracking is manifested in two critical ways that differ fundamentally from almost all other* revision control systems. First, Git's object store is based on the hashed computation of the contents of its objects, not on the file or directory names from the user's original file layout. Thus, when Git places a file into the object store, it does so based on the hash of the data and not on the name of the file. In fact, Git does not track file or directory names, which are asso-ciated with files in secondary ways. Again, Git tracks content instead of files. If two separate files located in two different directories have exactly the same content, Git stores a sole copy of that content as a blob within the object store. Git computes the hash code of each file according solely to its content, determines that the files have the same SHA1 values and thus the same content, and places the blob object in the object store indexed by that SHA1 value. Both files in the project, regardless of where they are located in the user's directory structure, use that same object for content. If one of those files changes, Git computes a new SHA1 for it, determines that it is now a different blob object, and adds the new blob to the object store. The original blob remains in the object store for the unchanged file to use. Second, Git's internal database efficiently stores every version of every file——not their differences——as files go from one revision to the next. Because Git uses the hash of a file's complete content as the name for that file, it must operate on each complete copy of the file. It cannot base its work or its object store entries on only part of the file's content, nor on the differences between two revisions of that file.

——Don Marti

<<Git   >>

Git   (   )   Git   Git
                                                      Git   Git
                    (rebasing)   (hook)   (   )
     Git   subversion

PDF

:http://www.tushu007.com