In this lecture, we will discuss…

- ✧ What is Git and how does it compare to other Version Control Systems
- ✧ Some good Git resources and references

Version Control Systems

- ✧ **Version Control System (VCS)**
  - System that **keeps track of changes** made to files
- ✧ Also known as SCM (Source Code Management)

Centralized VCS

- ✧ CVS, Subversion
  - Repo resides on some **central server**
  - Client only has **one version** of trunk or branch

Distributed VCS

- Git, Mercurial
  - The full repo resides **locally**
  - Contains **full history**
  - Server is (almost) not involved
    - Commit often and offline
    - Work on the beach / train
  - Can **push and pull** between repos
  - Back ups - **trivial and readily available**

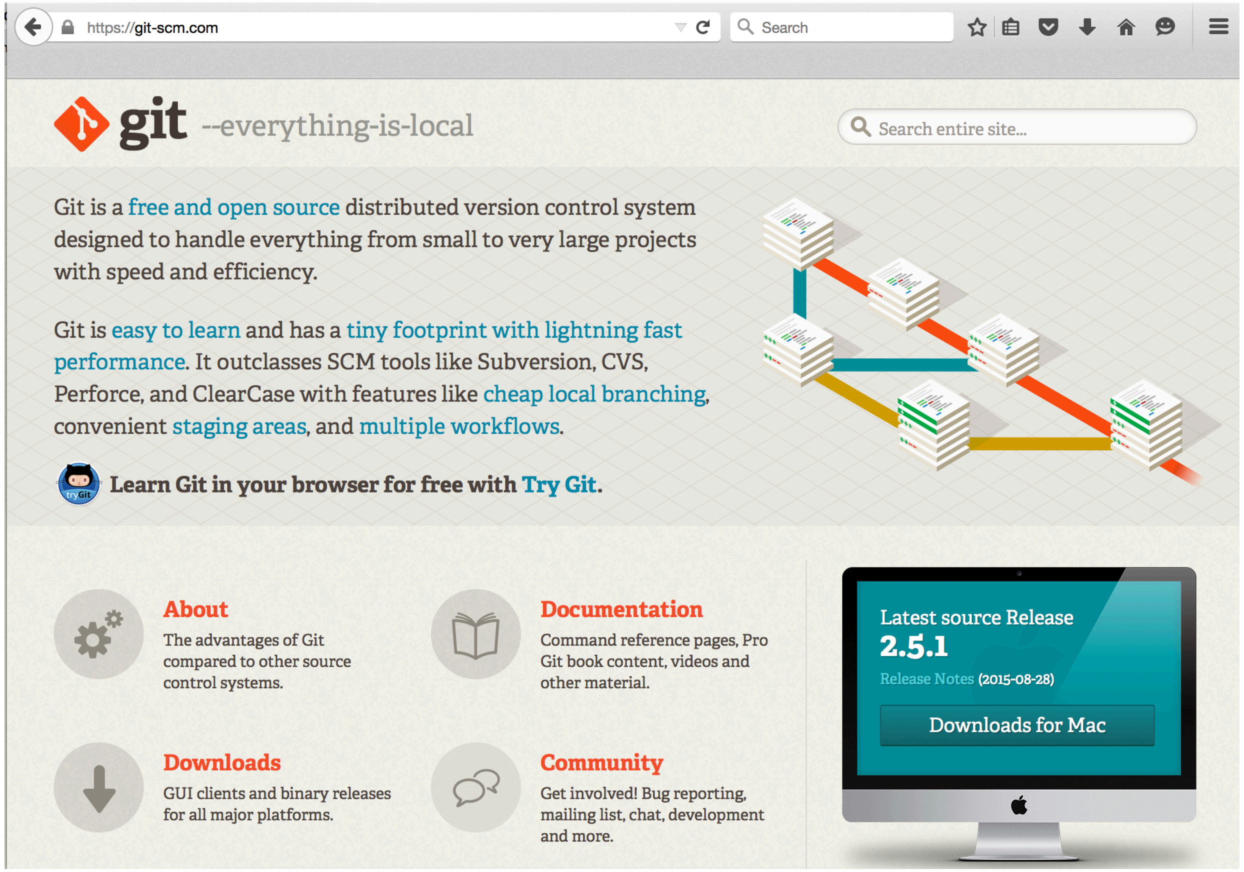
Git Basics

- Only **one** `.git` directory at the **top level** (not sprinkled throughout directory structure like SVN)

General Workflow

1. (empty) **Create** or (existing) **clone** repo
2. **Add changes** to staging area
3. Commit **changes** (from staging area to local repo)
4. Push **changes** from local to remote repo

Git's Official Site (git-scm.com)



Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Learn Git in your browser for free with Try Git.

About
The advantages of Git compared to other source control systems.

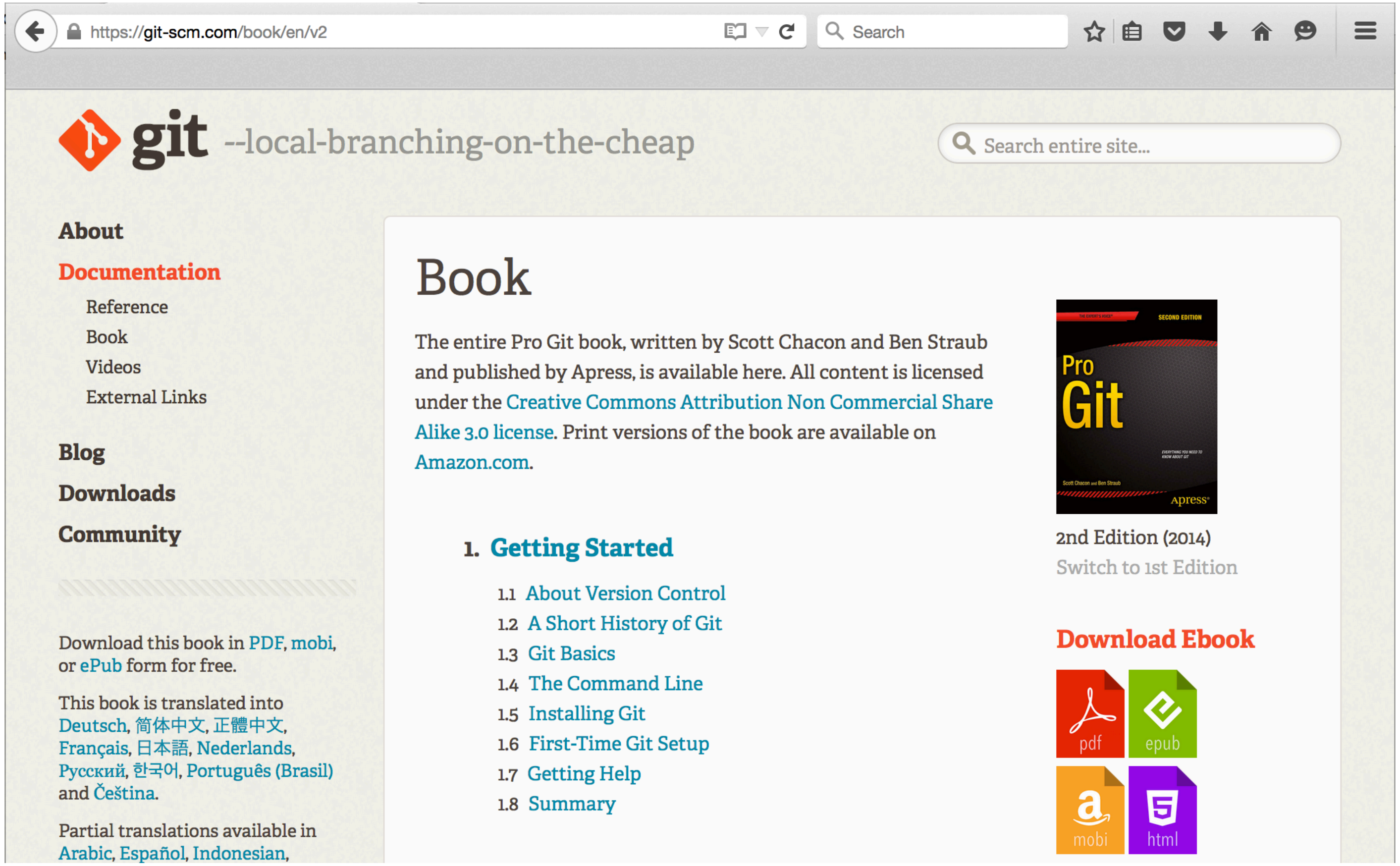
Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.5.1
Release Notes (2015-08-28)
Downloads for Mac

Pro Git - Free Git book (git-scm.com/book)



https://git-scm.com/book/en/v2

git --local-branching-on-the-cheap

Search entire site...

About

Documentation

- Reference
- Book
- Videos
- External Links

Blog

Downloads

Community

Download this book in PDF, mobi, or ePub form for free.

This book is translated into Deutsch, 简体中文, 正體中文, Français, 日本語, Nederlands, Русский, 한국어, Português (Brasil) and Čeština.

Partial translations available in Arabic, Español, Indonesian,

Book

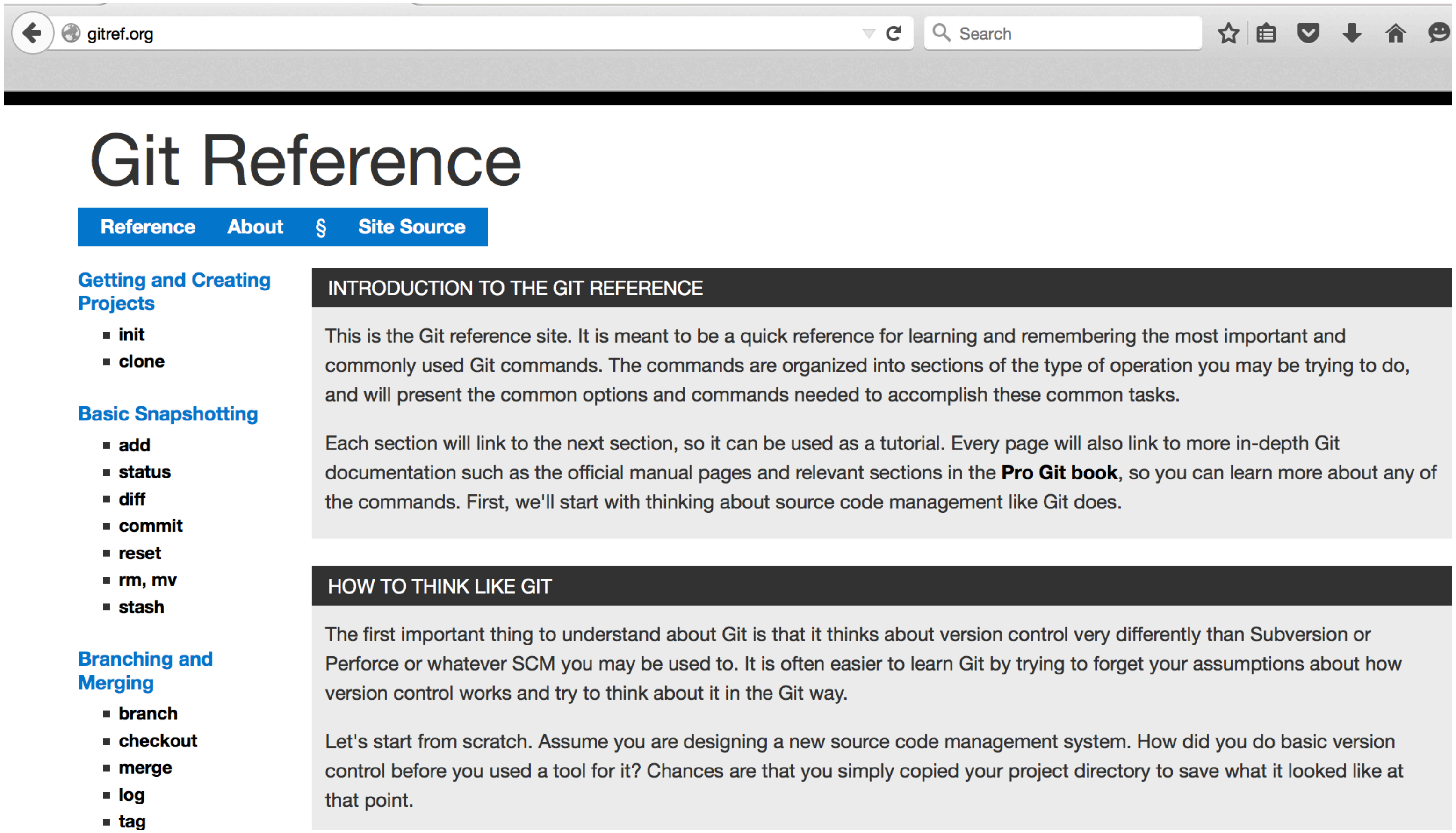
The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the Creative Commons Attribution Non Commercial Share Alike 3.0 license. Print versions of the book are available on Amazon.com.

2nd Edition (2014)

Switch to 1st Edition

Download Ebook

Good Git Reference (gitref.org)



Git Reference

Reference About § Site Source

Getting and Creating Projects

- init
- clone

Basic Snapshotting

- add
- status
- diff
- commit
- reset
- rm, mv
- stash

Branching and Merging

- branch
- checkout
- merge
- log
- tag

INTRODUCTION TO THE GIT REFERENCE

This is the Git reference site. It is meant to be a quick reference for learning and remembering the most important and commonly used Git commands. The commands are organized into sections of the type of operation you may be trying to do, and will present the common options and commands needed to accomplish these common tasks.

Each section will link to the next section, so it can be used as a tutorial. Every page will also link to more in-depth Git documentation such as the official manual pages and relevant sections in the **Pro Git book**, so you can learn more about any of the commands. First, we'll start with thinking about source code management like Git does.

HOW TO THINK LIKE GIT

The first important thing to understand about Git is that it thinks about version control very differently than Subversion or Perforce or whatever SCM you may be used to. It is often easier to learn Git by trying to forget your assumptions about how version control works and try to think about it in the Git way.

Let's start from scratch. Assume you are designing a new source code management system. How did you do basic version control before you used a tool for it? Chances are that you simply copied your project directory to save what it looked like at that point.

Summary

✧ Git lets you snapshot changes to your code
✧ Promotes committing changes often

What's next?

✧ Working with Git's local repository