Chapter 1. Getting Started

About Version Control

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Local Version Control Systems

- Many peoples version-control method of choice is to copy files into another directory (perhaps a time-stamped directory, if theyre clever).

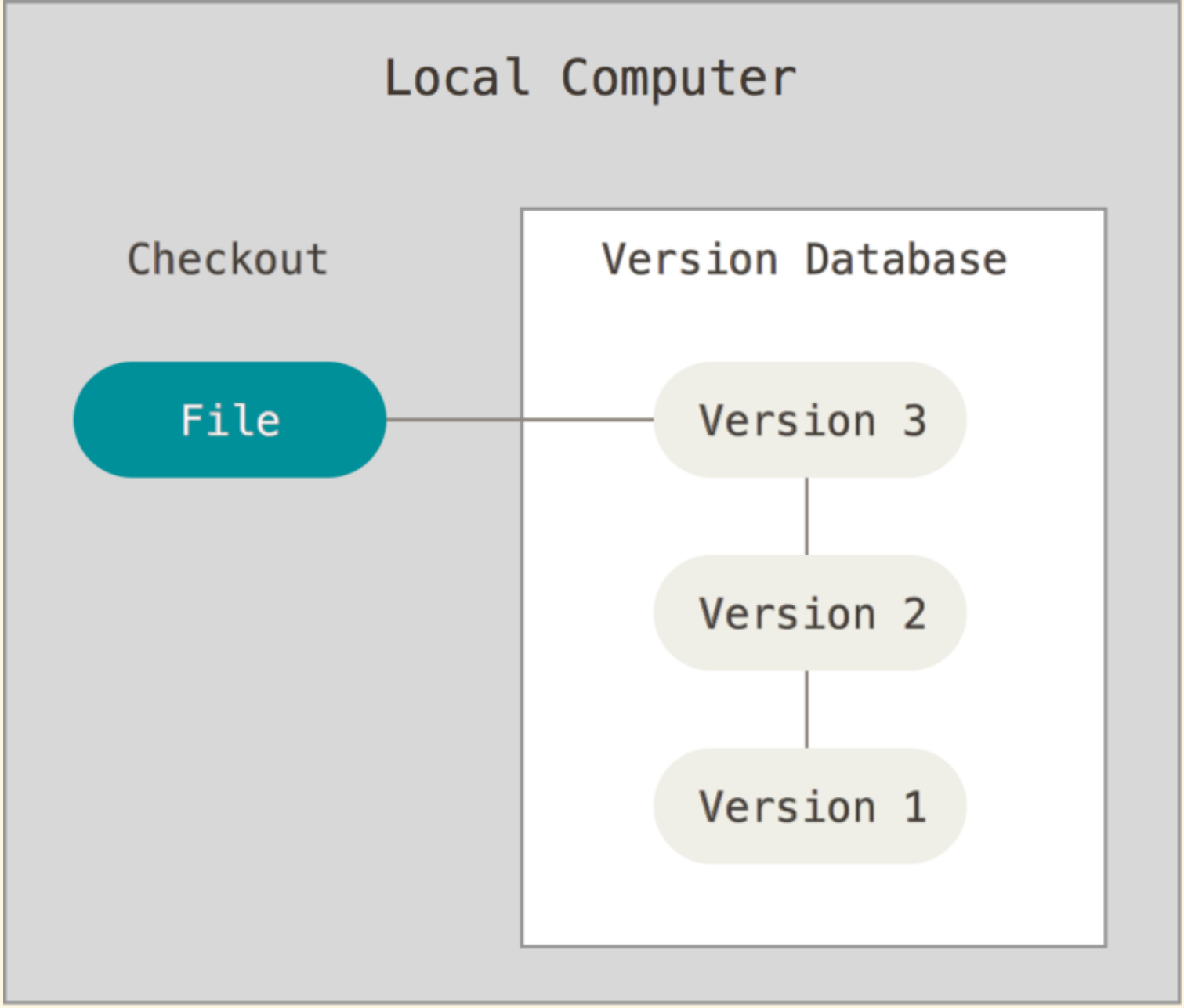


Figure 1-1. Local version control.

- One of the more popular VCS tools was a system called RCS.
- Mac OS X operating system includes the `rcs` command when you install the Developer Tools. RCS works by keeping patch sets (that is, the differences between files) in a special format on disk; it can then re-create what any file looked like at any point in time by adding up all the patches.

Centralized Version Control Systems

- Collaborate with developers on other systems.
- Centralized Version Control Systems (CVCSs) were developed.
- CVS, Subversion, and Perforce, have a single server that contains all the versioned files, and a number of clients that check out files from that central place.

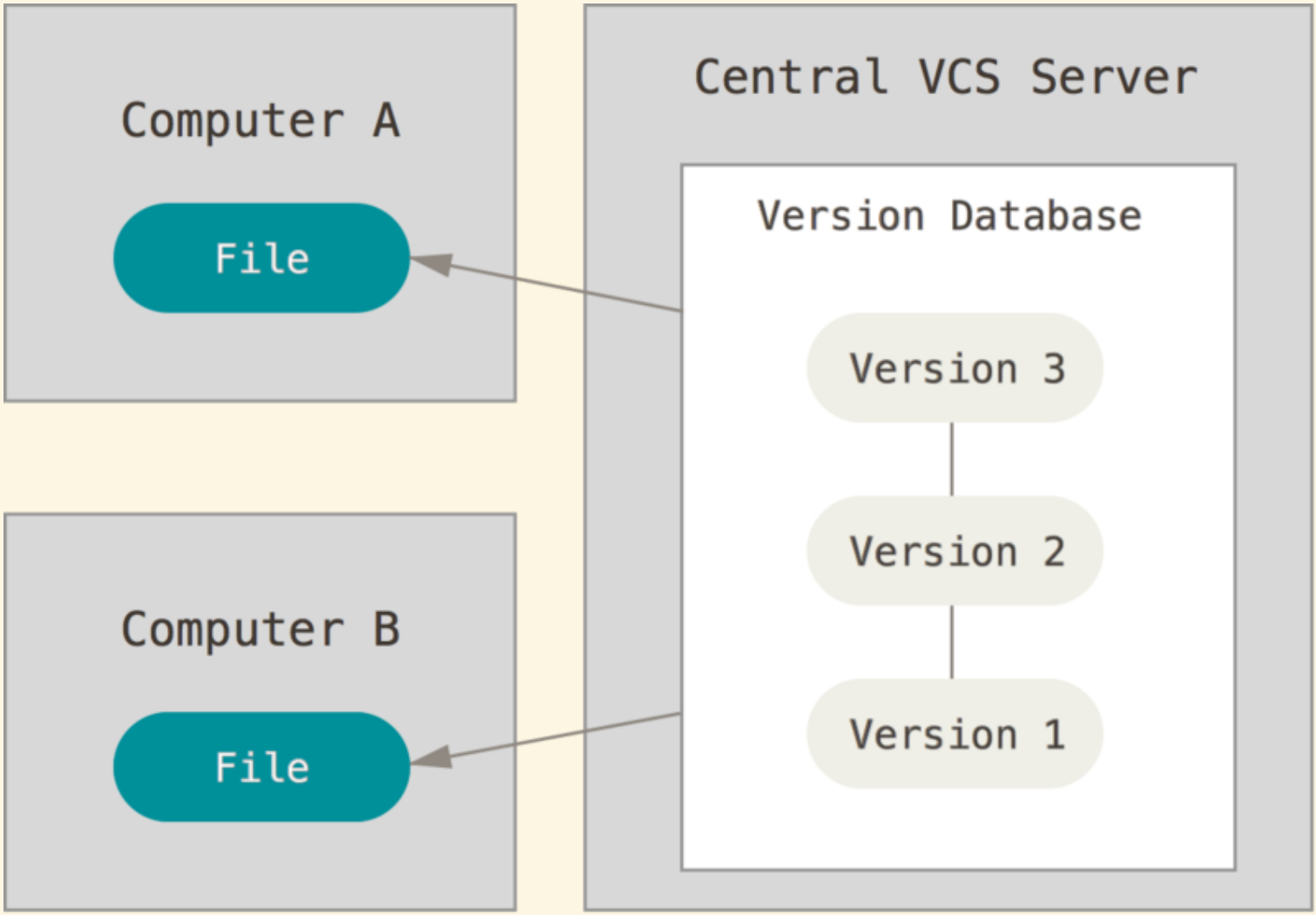
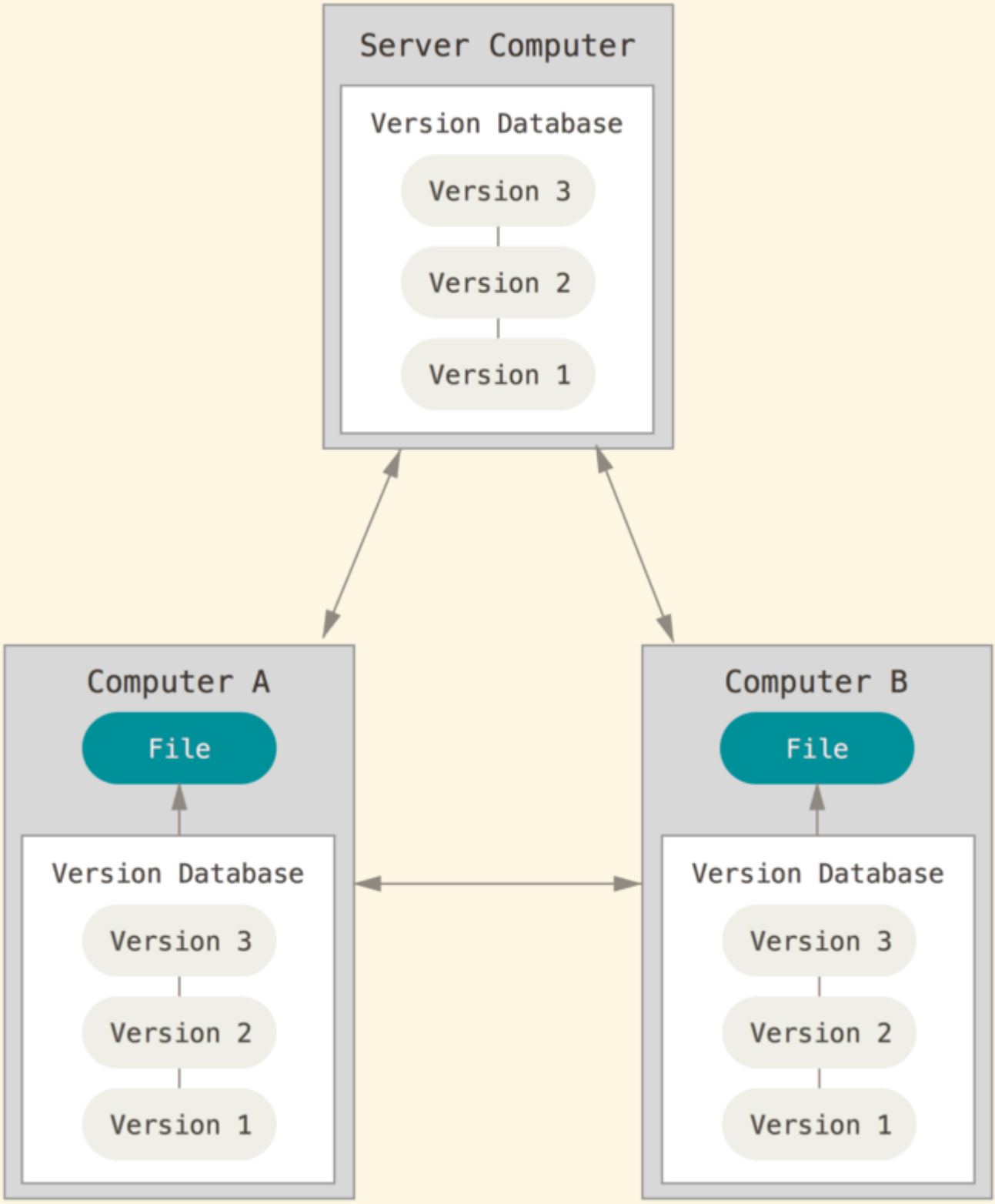


Figure 1-2. Centralized version control.

Advantages
Administrators have fine-grained control over who can do what; and its far easier to administer a CVCS than it is to deal with local databases on every client.
The most obvious is the single point of failure that the centralized server represents.
If the hard disk the central database is on becomes corrupted, and proper backups havent been kept, you lose absolutely everything.

In a DVCS (such as Git, Mercurial, Bazaar or Darcs), clients dont just check out the latest snapshot of the files: they fully mirror the repository.
Every clone is really a full backup of all the data.



** Figure 1-3. Distributed version control.*

- This allows you to set up several types of workflows that arent possible in centralized systems, such as hierarchical models.

A Short History of Git

- Some of the goals of the new system were as follows:
    - Speed

- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like the Linux kernel efficiently (speed and data size)

Git Basics

- Git, try to clear your mind of the things you may know about other VCSs, such as Subversion and Perforce; doing so will help you avoid subtle confusion when using the tool.

Snapshots, Not Differences

- The major difference between Git and any other VCS, Git thinks about its data.
- Most other systems store information as a list of file-based changes.
- These systems (CVS, Subversion, Perforce, Bazaar, and so on) think of the information they keep as a set of files and the changes made to each file over time.

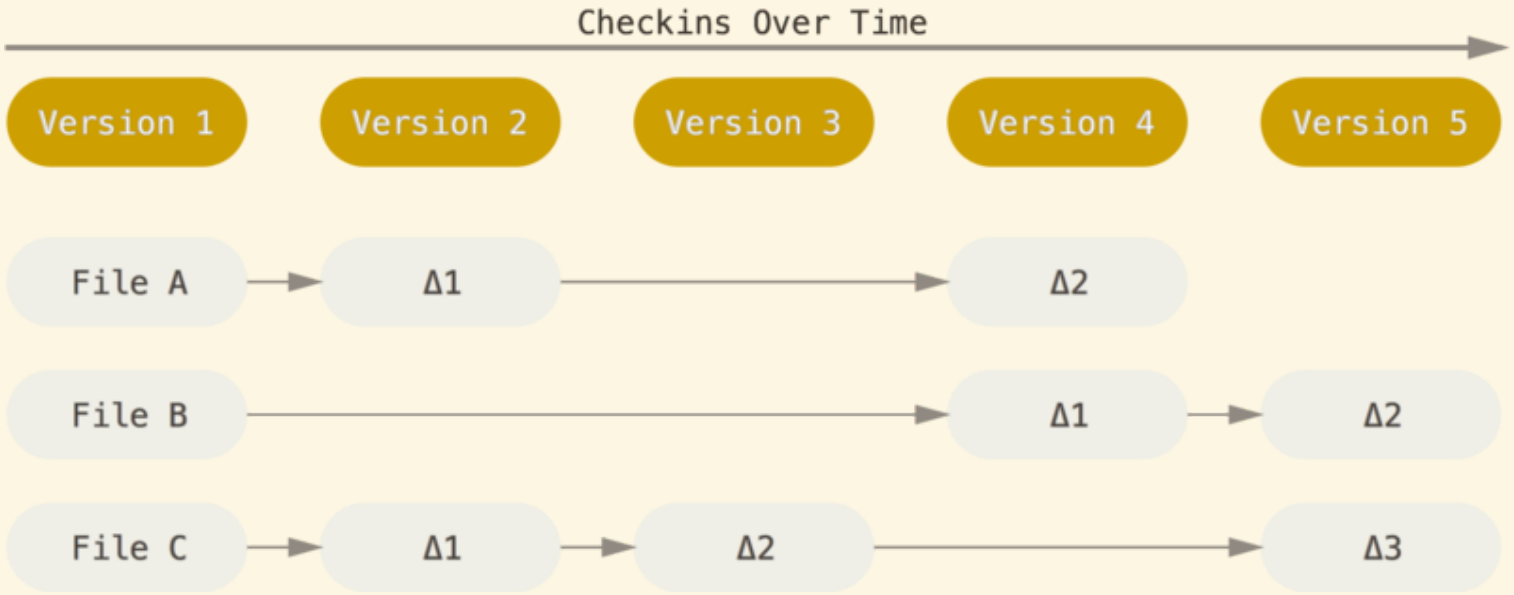


Figure 1-4. Storing data as changes to a base version of each file.

- Git doesnt think of or store its data this way.
- Instead, Git thinks of its data more like a set of snapshots of a miniature filesystem.
- Every time you commit, or save the state of your project in Git, it basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot.
- Git thinks about its data more like a **stream of snapshots**.

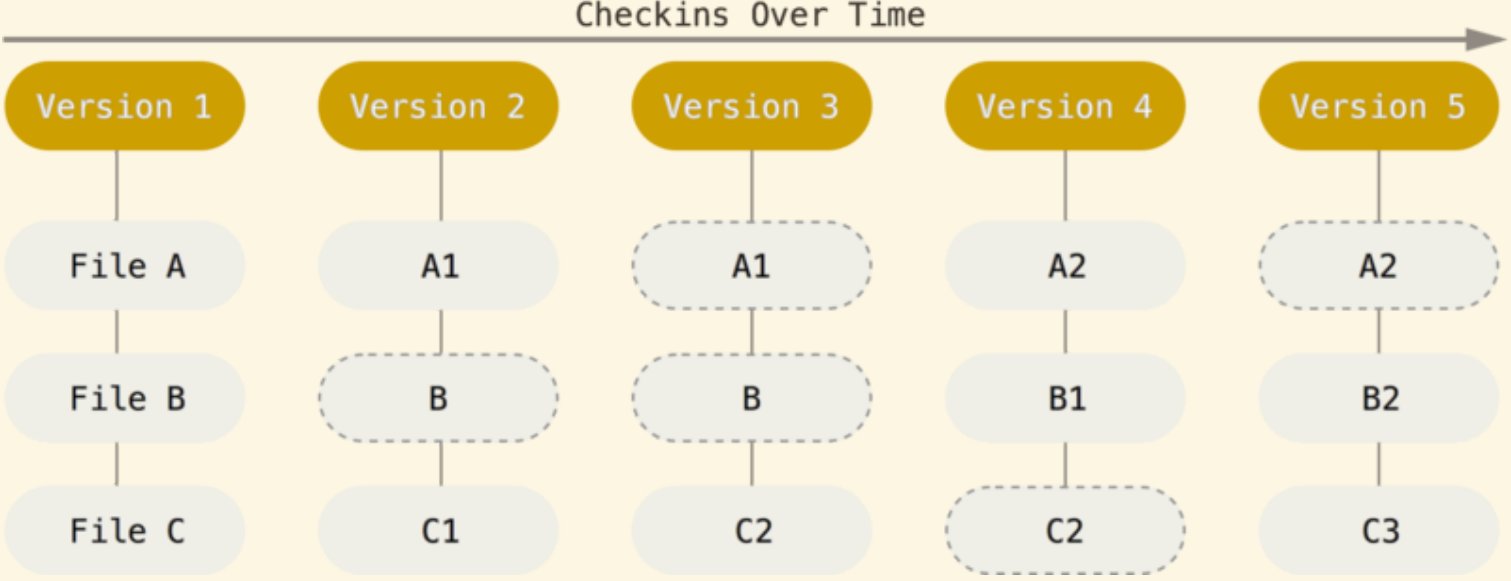


Figure 1-5. Storing data as snapshots of the project over time.

Nearly Every Operation Is Local

- Most operations in Git only need local files and resources to operate - generally no information is needed from another computer on your network.
- In Perforce, for example, you cant do much when you arent connected to the server; and in Subversion and CVS, you can edit files, but you cant commit changes to your database (because your database is offline).
- This may not seem like a huge deal, but you may be surprised what a big difference it can make.

Git Has Integrity

- Everything in Git is check-summed before it is stored and is then referred to by that checksum.
- This functionality is built into Git at the lowest levels and is integral to its philosophy.
- The mechanism that Git uses for this checksumming is called a SHA-1 hash.
- This is a 40-character string composed of hexadecimal characters (0-9 and a-f) and calculated based on the contents of a file or directory structure in Git.
- A SHA-1 hash looks something like this:

```
24b9da6552252987aa493b52f8696cd6d3b00373
```

- In fact, Git stores everything in its database not by file name but by the hash value of its contents.

Git Generally Only Adds Data

- When you do actions in Git, nearly all of them only add data to the Git database.
- It is hard to get the system to do anything that is not undoable or to make it erase data in any way.

- As in any VCS, you can lose or mess up changes you havent committed yet; but after you commit a snapshot into Git, it is very difficult to lose, especially if you regularly push your database to another repository.
- This makes using Git a joy because we know we can experiment without the danger of severely screwing things up.
- For a more in-depth look at how Git stores its data and how you can recover data that seems lost, see Undoing Things.

The Three States

- Git has three main states that your files can reside in: committed, modified, and staged.
- Committed means that the data is safely stored in your local database.
- Modified means that you have changed the file but have not committed it to your database yet.
- Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
- The three main sections of a Git project: the Git directory, the working directory, and the staging area.

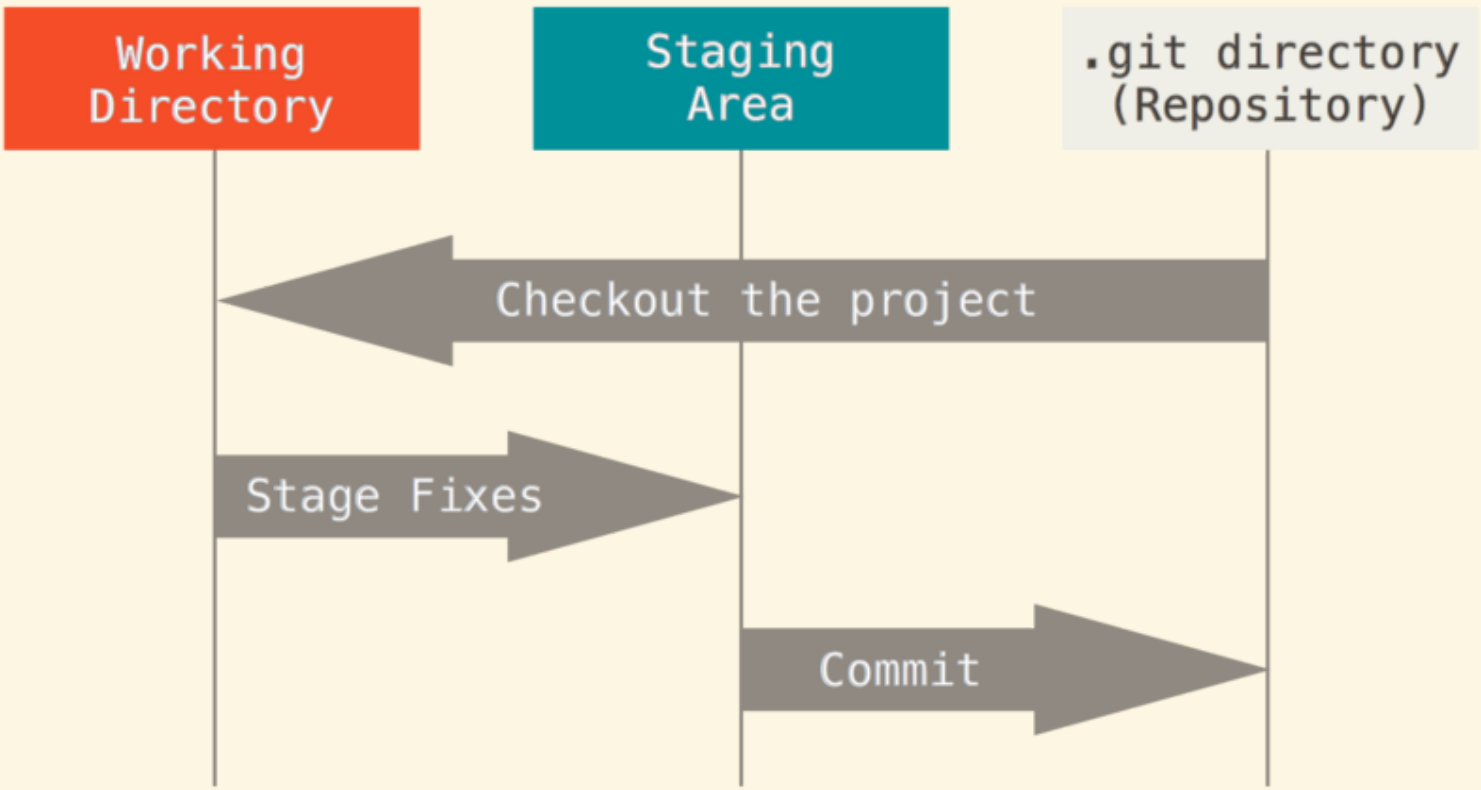


Figure 1-6. Working directory, staging area, and Git directory.

- The Git directory is where Git stores the metadata and object database for your project.
- This is the most important part of Git, and it is what is copied when you clone a repository from another computer.
- The working directory is a single checkout of one version of the project.
- These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.

- The staging area is a file, generally contained in your Git directory, that stores information about what will go into your next commit.
- Its sometimes referred to as the index, but its also common to refer to it as the staging area.
- The basic Git workflow goes something like this:
    1. You modify files in your working directory.
    2. You stage the files, adding snapshots of them to your staging area.
    3. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.

The Command Line

- The command line is the only place you can run **all** Git commands - most of the GUIs only implement some subset of Git functionality for simplicity.
- If you know how to run the command line version, you can probably also figure out how to run the GUI version, while the opposite is not necessarily true.

Installing Git

Installing on Linux

- If you want to install Git on Linux via a binary installer, you can generally do so through the basic package-management tool that comes with your distribution.
- If youre on Fedora for example, you can use yum:

```
$ sudo yum install git
```

- If youre on a Debian-based distribution like Ubuntu, try apt-get:

```
$ sudo apt-get install git
```

Installing on Mac

- The easiest is probably to install the Xcode Command Line Tools.
- On Mavericks (10.9) or above you can do this simply by trying to run *git* from the Terminal the very first time.
- If you dont have it installed already, it will prompt you to install it.

Figure 1-7. Git OS X Installer.

- You can also install it as part of the GitHub for Mac install.
- Their GUI Git tool has an option to install command line tools as well.

Installing on Windows

- Few ways to install Git on Windows. The most official build is available for download on the Git website.
- Just go to *http://git-scm.com/download/win* and the download will start automatically.
- Another easy way to get Git installed is by installing GitHub for Windows.

Installing from Source

- If you do want to install Git from source, you need to have the following libraries that Git depends on: curl, zlib, openssl, expat, and libiconv.
- For example, if youre on a system that has yum (such as Fedora) or apt-get (such as a Debian based system), you can use one of these commands to install the minimal dependencies for compiling and installing the Git binaries:

```git
$ sudo yum install curl-devel expat-devel gettext-devel \
```

```
openssl-devel zlib-devel
```

```
$ sudo apt-get install libcurl4-gnutls-dev libexpat1-dev gettext \
  libz-dev libssl-dev
```

```
* In order to be able to add the documentation in various formats (doc, html, info), these

 ```git
    $ sudo yum install asciidoc xmlto docbook2x


    $ sudo apt-get install asciidoc xmlto docbook2x
```

- Then, compile and install:

```
$ tar -zxf git-2.0.0.tar.gz
$ cd git-2.0.0
$ make configure
$ ./configure --prefix=/usr
$ make all doc info
$ sudo make install install-doc install-html install-info
```

- After this is done, you can also get Git via Git itself for updates:

```
$ git clone git://git.kernel.org/pub/scm/git/git.git
```

First-Time Git Setup

- These variables can be stored in three different places:

    1. `/etc/gitconfig` file: Contains values for every user on the system and all their repositories. If you pass the option `\--system` to `git config`, it reads and writes from this file specifically.

    2. `~/.gitconfig` or `~/.config/git/config` file: Specific to your user. You can make Git read and write to this file specifically by passing the `\--global` option.

    3. `config` file in the Git directory (that is, `.git/config`) of whatever repository youre currently using: Specific to that single repository.

- Each level overrides values in the previous level, so values in `.git/config` trump those in `/etc/gitconfig`.

- On Windows systems, Git looks for the `.gitconfig` file in the `$HOME` directory (`C:\Users\$USER` for most people).

- It also still looks for `/etc/gitconfig`, although its relative to the MSys root, which is wherever you decide to install Git on your Windows system when you run the installer.

Your Identity
```

- The first thing you should do when you install Git is to set your user name and e-mail address.
- Every Git commit uses this information, and its immutably baked into the commits you start creating:

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

- If you want to override this with a different name or e-mail address for specific projects, you can run the command without the `\--global` option when youre in that project.

Your Editor

- Can configure the default text editor that will be used when Git needs you to type in a message.
- If not configured, Git uses your systems default editor, which is generally Vim.
- If you want to use a different text editor, such as Emacs, you can do the following:

```
$ git config --global core.editor emacs
```

Checking Your Settings

- Can use the `git config --list` command to list all the settings Git can find at that point:

```
$ git config --list
user.name=John Doe
user.email=johndoe@example.com
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

- You may see keys more than once, because Git reads the same key from different files (`/etc/gitconfig` and `~/.gitconfig`, for example).
- In this case, Git uses the last value for each unique key it sees.

```
$ git config user.name
John Doe
```

Getting Help

- There are three ways to get the manual page (manpage) help for any of the Git commands:

```
$ git help <verb>
$ git <verb> --help
$ man git-<verb>
```

- For example, you can get the manpage help for the config command by running

```
$ git help config
```

- These commands are nice because you can access them anywhere, even offline.

Summary

- Should have a basic understanding of what Git is and how its different from the centralized version control system you may have previously been using.
- Should also now have a working version of Git on your system thats set up with your personal identity.