**Introducing Git**

The Git version control system (VCS) is fast becoming the de facto standard, not only in Android application development, but for software programming in general. Unlike earlier version control systems that require the use of a central server, Git is *distributed*, which means that each copy of the repository contains the entire history of the project, and no contributor is privileged. Git was developed by Linus Torvalds of Linux fame in order to manage the development of the Linux operating system. Like the open source movement itself, Git is systemically nonhierarchical and encourages collaboration.

While Git offers a wealth of features from the command line, this chapter focuses primarily on using Git from within Android Studio. The IntelliJ platform underpinning Android Studio has offered outstanding support for several VCS systems over the years, including Git. The consistency with the different supported systems is presented in a way that makes it easy for both newcomers and professionals to be proficient. However, it is important to understand the differences between using Git from within Android Studio and using Git from the command line. This chapter explains everything you need to get started with Git in great detail. You’ll reuse the Reminders app that you began in earlier chapters to learn the fundamentals of committing, branching, pushing, and fetching, among other important commands. You’ll work with both local and remote Git repositories and see how to use Git and Android Studio in a collaborative environment.

Open the HelloWorld project you created in [Chapter 1](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch01.xhtml). If you skipped that chapter, create a new project from scratch named **HelloWorld**. Use all of the default settings as you progress through the wizard. You will use this project briefly to understand the basics of Git setup.

**Installing Git**

Before you can begin using Git, you need to install it. Point your browser to <http://git-scm.com/downloads>. Click the Download button for your operating system, as shown in [Figure 7-1](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch07.xhtml#Fig1).

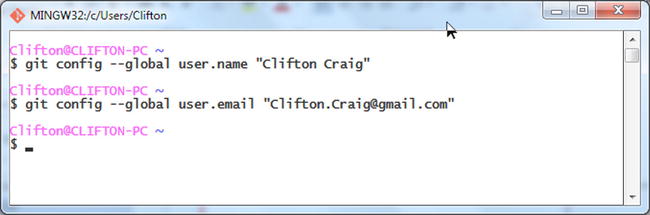


[***Figure 7-1***](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch07.xhtml#_Fig1)***.*** *Git download page*

We recommend installing Git in the C:\java\ directory on Windows or in the ~/java directory on Mac or Linux. Wherever you decide to install it, be sure that the entire path is free from spaces. For example, do not install Git in the C:\Program Files directory, because there is a space between Program and Files. Command line oriented tools like Git can potentially have trouble with directories that have a space in their name. Once your installation is complete, you must be sure that the C:\java\git\bin\ directory is part of your PATH environmental variable. See [Chapter 1](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch01.xhtml) for detailed instructions on how to add a path to the PATH environmental variable.

Launch the Git Bash terminal by clicking the Git Bash icon. If you’re running a Mac or Linux, just open a terminal. You need to configure Git with your name and e-mail so that your commits will have an author. From Git Bash, issue the following commands and replace John Doe’s name and e-mail address with your own. [Figure 7-2](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch07.xhtml#Fig2) shows an example.

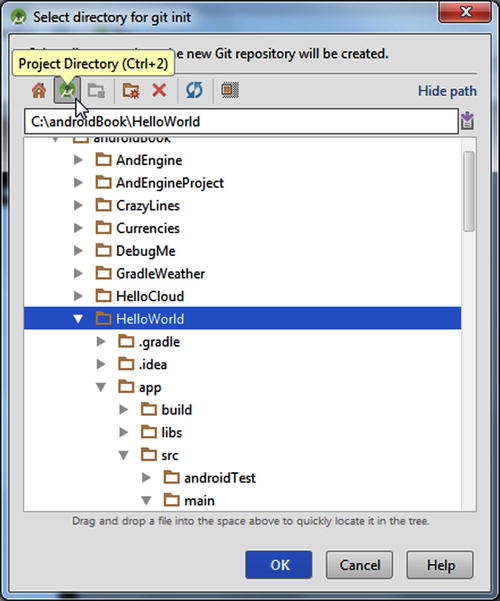
$ git config --global user.name "John Doe"  
$ git config --global user.email [johndoe@example.com](mailto:johndoe@example.com)



[***Figure 7-2***](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch07.xhtml#_Fig2)***.*** *Adding your name and e-mail to Git*

Return to Android Studio to continue setting up Git integration with Android Studio. Navigate to File imageSettings, and then find Git under the Version Control section in the left pane. Click the ellipsis button and navigate to the Git binary you just installed. Click the Test button to ensure that your Git environment is operational. You should see a pop-up indicating that Git executed successfully, as well as the version of Git you installed.

Navigate to VCS imageImport into Version Control imageCreate Git Repository. When the dialog box prompts you to select the directory where the new Git repository will be created, make sure you choose the project root directory HelloWorld. You can optionally click the little Android Studio icon in the directory chooser dialog box. This icon will navigate to the project’s root directory, as illustrated in [Figure 7-3](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch07.xhtml#Fig3). Click the OK button, and your local Git repository will be created.



[***Figure 7-3***](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch07.xhtml#_Fig3)***.*** *Selecting the directory for your Git repository*

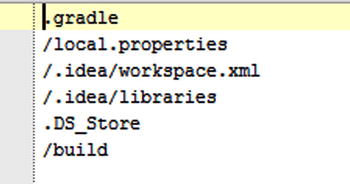
You will notice that most of the file names in your Project tool window have turned brown. This means that these files are recognized by Git locally but are not being tracked by Git and not scheduled to be added. Git manages commits in a two-stage approach (which is different from the approach used by other VCS tools such as Subversion and Perforce). The staging area is where Git organizes changes prior to a commit. The differences between the changes in progress, the staging area changes, and the committed changes are significant and can overwhelm new users. As a result, Android Studio does not expose these differences. Instead you get one simple changes interface that allows you to manage modified files and commit them with ease.

**Ignoring Files**

When you create the local repository, Android Studio generates special .gitignore files that prevent certain paths from being tracked. Unless you specify otherwise, Git will continue to track all the files in this directory and its subdirectories. However, .gitignore files can tell Git to ignore certain files or entire directories.

Typically, you will have one .gitingore file for the root directory, and one .gitignore file for each project. In *HelloWorld*, one .gitignore is located in the root of HelloWorld, and one .gitignore is located in the root of the *app* folder. Open the .gitignore file located in the root of HelloWorld and inspect its contents. [Figure 7-4](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch07.xhtml#Fig4) illustrates the generated .gitignore file in the project’s root directory. By default, Android Studio sets certain files to be excluded from your Git repository. The list includes files that are either generated by the project build or control settings specific to your local machine. For instance, the /.idea/workspace.xml file controls settings for your local configuration of Android Studio. Though it is possible to track this in Git, it is not necessarily a part of the project you are building and may in fact pose a problem because this file is unique to every workspace e.g. computer. Notice that one of the entries in .gitignore is /local.properties. Like workspace.xml, local.properties is unique to every computer.

Pay attention to the /build entry in the list. Gradle, the Android Studio build system covered in depth in [Chapter 13](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch13.xhtml), places all of its output here as you compile and run your project. Because this folder will contain everything from .class files to .dex files to the final installable Android package, and because its contents are constantly changing, it makes little sense to track it with Git. Find the local.properties file in the Project tool window. You will notice that it’s black, whereas the other files are brown.



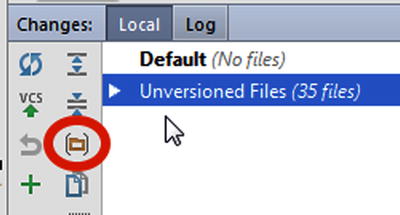
[***Figure 7-4***](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch07.xhtml#_Fig4)***.*** *The root .gitignore file contents*

Android Studio uses a color scheme that allows you to easily identify what your version control system will see as you work. As we’ve already stated, brown indicates that a a file is recognized by Git locally but is not being tracked by Git, and is not scheduled to be added. Blue indicates a file that is being tracked by Git and has been changed. Green is used for brand-new files that are being tracked by Git. Black indicates files that either have not been changed or are not being tracked. Android Studio constantly keeps track of files that are added to your project and prompts you as necessary to keep these files in sync with Git.

**Adding Files**

Open the Changes view at the bottom of the screen. It includes two sections: Default and Unversioned Files. The Default section, initially empty, represents the active changelist. As you modify and create files, they will fall under this section, because it holds files that are ready to be committed to your VCS. The Unversioned Files section contains everything that is not being tracked by VCS.

Because all of the project files are not yet tracked, they fall under the Unversioned Files section. You will want to add these to your repository. On the left side of the Changes view are two columns of icons. In the right column, click the third icon from the top (a folder icon); see the circled icon in [Figure 7-5](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch07.xhtml#Fig5). This is a toggle that enables you to group files by folder to better understand their relative location within your project. Right-click the Unversioned Files section header and click Add to VCS from the context menu to add these files to the Git index. Alternatively, you can click and drag the entire section to the bold Default section.



[***Figure 7-5***](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch07.xhtml#_Fig5)***.*** *Group files by folders*

After adding all the files, click the VCS icon with the green arrow pointing upward. This opens the familiar Commit dialog box you began using in [Chapter 5](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch05.xhtml). Click Commit to record your changes, and the Default section will eventually empty out. You can also press Ctrl+K | Cmd+K to perform the same action. From this point on, each file you touch while in Android Studio will be tracked under Git.

**Cloning the Reference App: Reminders**

This section extends the Reminders app that you created in [Chapters 5](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch05.xhtml) and [6](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch06.xhtml). We invite you to clone this project using Git in order to follow along, though you will be recreating this project with a new Git repository based forked from the repostory used in [Chapters 5](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch05.xhtml) and [6](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch06.xhtml). If you do not have Git installed on your computer, see [Chapter 7](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch07.xhtml). Open a Git-bash session in Windows (or a terminal in Mac or Linux) and navigate to C:\androidBook\reference\ (If you do not have a reference directory, create one. On Mac navigate to /your-labs-parent-dir/reference/) and issue the following git command: git clone <https://bitbucket.org/csgerber/reminders-git.git> RemindersGit. You will use Git features to modify the project as if you were working on a team. Through the process, you will learn how to fork and clone a project, and set-up and maintain branches as you develop features. Before beginning this exercise, rename the Reminders project you completed in [chapter 6](file:///C:\Users\rjones01\AppData\Local\Temp\adbwelua.3my\OEBPS\9781430266013_Ch06.xhtml) to RemindersChapter6 because you will be recreating this folder shortly. In windows you can right click the folder in Explorer and choose rename. On Linux or Mac run the following command: mv ~/androidBook/Reminders ~/androidBook/RemindersChapter6.