Getting the Latest Version with Git and Compiling (Lab)

coursera.org/learn/git-distributed-development/supplement/1eKZN/getting-the-latest-version-with-git-and-compiling-lab

If you already have Git installed, a nifty way to get the latest version is to use Git itself. If your network connection is good enough, download from the public repository by doing:

```
$ git clone -v https://github.com/git/git.git
```

if you do not have git already installed, you can download a zipped archive from the same site, and then unpack it and the rest of the instructions for compiling and installing will be the same.

The full git repository will take up about 145 MB after being downloaded. Note that most of this space is taken up by the **.git** directory, which contains the whole revision history. The archive itself is much smaller.

If you have git installed, you can see what version you are running with:

```
$ git --version

git version 1.8.3.1
```

To see what the latest version of git is in the downloaded repository, go to the **git** directory created during the cloning operation:

```
$ cd git
```

There are a number of methods, including:

```
$ git tag | tail -1

v2.9.5
```

but that might not be accurate, as use of tags is voluntary. For example, doing:

```
$ git logcommit 90bbd502d54fe920356fa9278055dc9c9bfe9a56

Merge: 085f5f9 d32eb83

Author: Junio C Hamano <gitster@pobox.com>

Date:   Thu Mar 22 14:36:51 2018 -0700


    Sync with Git 2.16.3

....
```

seems to indicate a later version, which we can check once we prep, compile, and install using the steps we mentioned earlier:

```
$ cd git

$ ./configure

$ make

$ sudo make install
```

You may be able to skip the **./configure** step, as recent versions may not need it, and the script may not exist.

With this particular version, after compilation one finds:

```
$ ./git --version

git version 2.17.0.rc1.35.g90bbd50
```

By default, if you compile and then install everything will be placed under your **$HOME** directory, with the executables all in the **bin** directory therein. However, you can specify an alternative when you compile, as in:

```
$ make prefix=/usr/local
```

or

```
$ make prefix=/opt
```

If you then place the new directory in your path, you will be using the newer version of git, rather than the installed ones, as in:

```
$ export PATH=/opt/bin:$PATH
```

When you compile with the **make** command, it is possible you may have to install some additional software (usually for header files). For example, on Red Hat Enterprise Linux distributions and their close kin, you might need to do:

```
$ yum install curl-devel expat-devel openssl-devel
```

or, on deb-based systems:

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

As an alternative that loses some functionality, you can do:

```
$ make prefix=/opt NO_CURL=1 NO_EXPAT=1 NO_SSL=1
```

Reading the **Makefile** will show you other such options.

Once you have compiled, you can do:

```
$ make prefix=/opt install
```

and you have the latest version installed, which you can always use by putting it earlier in your path than the system version.

To learn further about all of this, read the **README**, **INSTALL**, and **Makefile** files in the main Git source directory.