SYSTEMS II
USING CLOUD STORAGE

Revision 1, 2014-Mar-08

For our projects, we have been using `remus` and `romulus`, the standard IT Linux servers. However, you now also have permission to login to `vega.cs.amherst.edu`, which is a CS-specific server that provides access to the same files as the standard servers, but offers a more updated system. If you want to use the cloud storage services described in this document, you should use `vega.cs` as your "home base" for the projects assigned for this class. This document outlines how (and, to some extent, why) to use cloud storage from this CS server and from other computers. It also introduces additional storage tools commonly used for programming and for collaborative projects that you may wish to explore in carrying out this course's assignments.

1 Why cloud storage

The terms *cloud* and *cloud storage* may be poorly-defined, marketing-driven terms that some of us[1] find annoying; however, their broad use indicate a meaningful movement toward more Internet-based services. The availability of software, storage, software platforms, and other services as part of "the cloud" provides each of us a meaningful increase in the computing capacity that is available through a wide variety of devices and interfaces. We will likely discuss some of these capabilities later in the semester.

As a practical matter, though, cloud storage is a readily available utility that we can put to good use in carrying out the course's projects. Moving files between the college's servers (e.g., `remus/romulus/vega.cs`, the `U:` drive, etc.), the CS department's systems (e.g., `castor.cs`), and your own personal computers tends to be a nuisance. Sharing such files with another user (e.g., during group projects), is messy even when properly configured by a system administrator, and then leaves the group tied to one system to do their work.

Cloud storage offers an opportunity to eliminate these problems. If each of you had a cloud storage account, then software installed on both the CS departmental systems and on your own computers can allow you to access the project files through either of those systems. These cloud services also allow you to share any directory or file with others of your choosing, with no need for special login groups, nor the usual problems with the management of access permissions. Finally, because these cloud storage services are automatically backed up[2], any attempt to work on a project on your own computer does not run the risk of data loss because of localized storage failure or management error. Your work, whether done on a college server or on your own computer, is automatically and continually synchronized with the cloud storage servers. These cloud services also keep copies of deleted files, allowing for easy recovery from mistaken deletions.[3]

As we will see in Section 2, every member of the Amherst College community has a Google Drive account. Not only do these accounts provide sufficient storage (5 GB) for our project work, but we will see, in Section 3, that synchronization tools make it easy to mount this storage on just about any system (including smartphones). Once mounted, changes made to any files are

[1] e.g., me.

[2] As are the College's and CS department's file systems.

[3] Some may reasonably be creeped out about this characteristic of cloud storage. See Section 4 if it troubles you.

automatically and immediately synchronized with the storage servers (unless you are not connected to the network, in which case synchronization occurs when you reconnect, allowing you to work while offline) and, consequently, with any other system that is also mounting the same storage.

The rest of this document outlines how to gain access to your Google Drive account; how to install and use the mouting/synchronizing software for this storage; how to use the storage securely if you want to put any private or confidentially data on it; and an introduction to more sophisticated tools commonly used for programming to manage and share your files.

2 Your Amherst College Google Drive account

There are many cloud storage services available—SkyDrive, Box, Dropbox, Amazon Cloud Drive, SugarSync, SpiderOak, and Ubuntu One, to name a few—but the one available through your College account is Google Drive. Specifcally, the facility that we Amherstians know as AC Apps is really just a College-branded use of the Google Apps facilities. AC Apps carries with it, though, 5 GB of cloud storage. Just as importantly, access to these services requires only your Amherst College username and password; you login using the usual College web login page, and that grants access to the Google Drive facilities.

Getting started with AC Apps is simple: Just visit the AC Apps page. You will be prompted to login. If you enter **your college email address** into the *username* field, then you will be brought to the College login page. There you can enter **your college username and password**. (I believe that if you are already logged into the College's web pages, then this second step won't be necessary—you will be automatically authenticated to Google.)

You will be brought immediately to a web view of your Google Drive account. Here, you have the ability to manually upload or download files, create directories, or otherwise manipulate the collection of files and folders. While you may at times find it useful to access your files through the ubiquitous web interface, this is not the form of access to this storage that we seek to use for our projects. Below, Section 3 will allow you more direct access, mounting the cloud storage as a file system that you can access like any other set of files and directories.

Handling multiple Google accounts. You may already have a personal Google account. If you do, then you may already have your web browser logged into that account. You may even be using your own Google Drive space, including the desktop synchronization utility, which itself connects to your personal account.

In many ways, Google is well designed to handle multiple logins within a single browser. Having already logged into one account, you may click, in the upper-right corner of most Google services, on your account drop-down menu. You'll find a button to *Add Account*, allowing you to login to another Google account without logging out from the first, and then switching between them. You may therefore be logged into your personal and your college Google account simultaneously. We will see, in Section 3, that synchronizing multiple Google Drive storage accounts on your system **can** be done, although with a non-standard tool for some systems.

3 Using synchronization tools

What makes cloud storage more useful than simply having access to a server with which you may upload and download files is the synchronization software that you may use to connect to it. By installing one of these utilities on a system, you may then access the files stored on that server as though the files were directly on your system. Any changes to the set of files are quickly and automatically uploaded, synchronizing the server with your local copy. Any other system that is syncrhonizing with those same files then automatically downloads those changes, keeping all copies of those files and directories synchronized as well.

This section describes how to install and use synchronization utilities that work with Google Drive. Doing so will allow you to work with a single copy of your work, and having that single copy updated across all of the machines that you use, including the Google servers themselves.

3.1 Installing/configuring the synchronization clients

3.1.1 On your own computer

The easiest system with which to begin is your own. Assuming that you have either a Windows or a Mac laptop of your own, [4] and assuming that this is the only Google Drive account that you plan to use in this fashion, Google provides its own Google Drive app. You can download this utility from Google for your computer (it will figure out whether you're using a Windows or Mac computer) and follow the installation instructions. From there, you configure it to use your College account.

Once installed and configured, this utility will create a `Google Drive` folder within your home folder. Any files that may already be in your Google Drive account will automatically be downloaded. From this point onward, using the utility is a triviality: you simple create, remove, modify, or otherwise use files and folders within your `Google Drive` folder. The synchronization will happen quietly and in the background at any time you are connected to the Internet.

3.1.2 On the CS server

As footnoted above, the Google Drive synchronization utility does not exist for Linux,[5], and `vega.cs` uses Linux Mint, which is an Ubuntu derivative (which is a Debian derivative). Luckily, another utility, *Insync*, runs on Linux and provides the same synchronization capabilities as Google's own utility. Indeed, *Insync* runs not only on Linux, but also on Windows, Macs, and iOS/Android mobile devices.

Insync has been installed on `vega.cs` (as well as the CS server `castor.cs` and workstations in 007, but that's not so relevant here). To use it for the first time, I recommend that you login to `vega.cs` using some kind of graphical environment, such as *Xming* on Windows or `ssh -Y`

[4]If you have some other kind of laptop, fear not. If you have the rare *ChromeBook*, then Google Drive may well come with it. If you have a Linux computer, note that this Google Drive utility does not yet exist. Luckily, the *Insync* utility described in Section 3.1.2, which we use on `vega.cs`, will also work for your Linux laptop. It is shameful that Google does not provide their own utility for Linux, given that Google's own, massive infrastructure uses Linux, and that without this freely available operating system, Google would have had a great deal of difficulty as a fledgling company so many years ago, having to spend a great deal of money on systems software. Shame Google, shame…

[5]Shame, I say!

along with Xquartz on a Mac. (On linux, X11 is already running, so `ssh -Y` will do the trick.) Once you have done so, open a shell and use the following incantation to start the *Insync* client:[6] So, at the command line. . .

```
$ insync start
```

This command will start *Insync*. After a few moments for initialization, the *Insync* icon will appear in the system tray.[7] Additionally, a window should open, propting you to login and configure *Insync* by logging in. You should login using your college credentials (described in Section 2), and then otherwise choose the default settings. *Insync* will then "remember" the account information; each time you start the client, it will connect to your Google Drive storage automatically.

Once you have logged in, *Insync* will create a directory for itself, and within that, it will create a direectory for each account to which you connect it. So, to get at the Google Drive storage for your College account, change into that directory: [8]

```
$ cd ~/Insync/sfkaplan@amherst.edu
```

Any changes to files and folders in this directory are automatically synchronized. Any changes made to this storage through other systems are automatically updated on this system.

3.1.3 On your smartphone/tablet

Both *Google Drive* and *Insync* apps are available for *iOS* (through the Apple App Store) and for *Android* (through Google Play). You may download and install these apps for any such devices that you have, giving you access to your work from your mobile devices. How useful such access might be is up to you, but it's unlikely to hurt.

3.2 Sharing folders and files

Cloud storage is also particularly useful for collaborative work. You may share any folders or files in your Google Drive account, thus establishing a collaborative working group where each member may access and update any of those files. To share folders/files, follow these steps:

1. Access the Google Drive web interface.

2. Select the file(s)/folder(s) that you want to share by clicking the check-box next to them.

3. At the top of the window, there is a button with a little-person-pictogram and a plus-symbol. If you hover over this button, the text bubble identifies it as the *Share* button. Click it!

4. A new *Sharing settings* window will appear. Near the bottom, you may specify those who should have access to these shared files/folders. Since all of your classmates and faculty have Google Drive accounts based on their College usernames, you should easily be able to

[6] Insync runs only so long as you are logged in. You must start it again each time you login to `vega.cs`.

[7] The tray is usually at the top or bottom of your graphical environment, with an icon for each of a number of small activities, such as the network connection manager, a battery/power status, etc.

[8] Of course, use your username, not mine.

determine the email addresses of those with whom you want to share. Enter each username, one at a time. To the right of the space where you enter each username, you may also specify the degree of sharing, determining whether that user can merely read the files/folder, or also edit them.

5. Having added all of those with whom you want to share the files/folders, click *Share & save* at the bottom, and you're done!

Once you have specified the users with whom to share these files/folders, those people will receive an email, informing them of the sharing offer. They should follow the instructions in the email, accepting access to the files/folders. Once they do so, they will not only be able to see the files/folders in the web interface, but those files will also be synchronized onto their CS systems, desktops/laptops, and mobile devices.

4 Protecting private work

Much of the work you do may not be sensitive. Any files that you share with other users must be stored in a form that all of those users can access and use. However, you may also wish to store some files/folders that you not only do not need to share, but that you also do not want Google's automatic content-scanning algorithms to access. In that case, you may wish for some part of your cloud storage to be automatically encrypted. This concern may be valid for any cloud storage service, including SkyDrive, Dropbox, Ubuntu One, and others.

There are many utilities for encrypting files, but some are well suited to acting as an automatic layer between your access to your files on your systems and the files that are synchronized with the cloud servers. I will recommend one approach here because it can, just like the Google Drive service itself, work on nearly any type of system.

4.1 Overview

If you want to encrypt an entire folder automatically (whether that folder is storage on a cloud storage account or on some other device or service), then you should use *file system level encryption*. This use of *virtual file system* allows you to encrypt any portion of your home directory, having the encryption and decryption performed automatically. In particular, these virtual file systems are typically implemented at the *user level*—that is, regular users may use them without having special, administrator-level privileges. In fact, our cloud storage synchronization clients are themselves user-level virtual file systems.

To use this type of folder/file encryption, you must specify two folders:

- **A *source* or *root directory*:** This is a folder that will store all of the encrypted material. If you look inside this folder, you should see the gobbledy-gook[9] that is the result of encryption. For some utilities, the folders and file names look normal, but their contents are encrypted and thus unreadable in their raw format. Better utilities will also encrypt the folder and file names themselves.

[9]Not a technical term.

- **A *destination directory* or *mount point*:** This folder is the one that the encryption utility makes appear as a normal set of files/folders. It is really the result of decrypting the source directory and making its contents appear "normal" in this destination directory. Any changes to files made in this destination directory are automatically encrypted and stored in the source directory. The destination directory appears only on the system you are using; it is not actually stored on any storage device, since its encrypted counterpart is what is actually stored instead.

4.2 EncFS

The encrypting, virtual file system tool EncFS does exactly what is described above. It provides effective encryption by using the AES encryption algorithm, applying it to all of the folders and files at a given mount point, storing the encrypted results in a given source directory. Below are instructions for installing and/or configuring *EncFS*, with a partiular focus on placing the source directory into your automatically synchronized cloud storage.

4.2.1 Linux and the college systems

EncFS is installed on the college and departmental systems, including `vega.cs/remus/romulus`. If you run your own Linux system, it is a software package that is likely available in your system's default repositories, and therefore easily installed using those tools (e.g., `apt-get`, `yum`, `zypper`).

In order to configure *EncFS* so that you have an encrypted directory in your cloud storage space, follow these steps:

1. **Create a source directory:** Within your cloud storage folder, create a new directory into which all of the encrypted content will be stored. This will be the folder whose contents, when examined directly, will be gibberish. You may create it as you would create any other folder:

   ```
   $ mkdir ~/Insync/sfkaplan@amherst.edu/encfs-source
   ```

2. **Create a mount point:** Create a directory that will serve as your destination for decrypted files/folders from your source directory. It is in this directory that you will normally store, manipulate, and otherwise use your files and folders in a normal manner. Again, it's creation is quite normal:

   ```
   $ mkdir ~/encfs-mount
   ```

3. **Initialize the encrypted source directory:** The very first invocation of *EncFS* will prompt you to create the encrypted source directory. These prompts **should appear only once** in the lifetime of an encrypted source directory. The initialization sequence should look something like this:

```
$ encfs ~/Insync/sfkaplan@amherst.edu/encfs-source ~/encfs-mount
Creating new encrypted volume.
Please choose from one of the following options:
 enter "x" for expert configuration mode,
 enter "p" for pre-configured paranoia mode,
 anything else, or an empty line will select standard mode.
?> p

Paranoia configuration selected.

Configuration finished.  The filesystem to be created has
the following properties:
Filesystem cipher: "ssl/aes", version 3:0:2
Filename encoding: "nameio/block", version 3:0:1
Key Size: 256 bits
Block Size: 1024 bytes, including 8 byte MAC header
Each file contains 8 byte header with unique IV data.
Filenames encoded using IV chaining mode.
File data IV is chained to filename IV.
File holes passed through to ciphertext.

------------------------- WARNING --------------------------
The external initialization-vector chaining option has been
enabled.  This option disables the use of hard links on the
filesystem. Without hard links, some programs may not work.
The programs 'mutt' and 'procmail' are known to fail.  For
more information, please see the encfs mailing list.
If you would like to choose another configuration setting,
please press CTRL-C now to abort and start over.

Now you will need to enter a password for your filesystem.
You will need to remember this password, as there is absolutely
no recovery mechanism.  However, the password can be changed
later using encfsctl.

New Encfs Password:
Verify Encfs Password:
```

First, notice that I suggest selecting *paranoia mode*. You may read about the options and choose standard mode, but the paranoia mode encrypts pathnames as well as file contents, and uses a 256-bit key for the AES algorithm, improving the security provided by the encryption.

Of course, choose a good password for this encrypted directory. The password is unrelated to your login password for this system or any other. I recommend using diceware as a sound approach for choosing a *high-entropy* password. The argument in favor of this form of password selection is easy to follow.

4. **Mount the encrypted directory:** The previous step—particularly, the use of the `encfs` command—not only initializes the encrypted directory, but also *mounts* it. That is, the use of the `encfs` command makes the decrypted contents of the source directory visible at the mount point. At any subsequent time that you wish to mount the encrypted directory, use that command, specifying both encrypted and unencrypted directories.

4.2.2 Macs and Windows

Usual desktop/laptop computers running Mac OS X or Windows operating systems can also use *EncFS*, although as part of another software package. Specifically, *Boxcryptor* is an implementation of *EncFS* for those systems. Visit the *Boxcryptor* site for instructions on how to download, install, and use this utility.

You should be aware of two drawbacks to using *Boxcryptor*. The first is that it is commercial software. A free copy allows you to use it with limited capabilities: a smaller AES key; no encryption of file/folder names; only one mounted directory at a time. You may remove these limitations if you wish to pay for a full license to the software. The second limitation is that *Boxcryptor*, while an implementation of *EncFS*, does not (particularly in its free version) mount any encrypted directory created with standard *EncFS*. To elide this problem, you may wish to create your encrypted volume with *Boxcryptor*. Once you have done so, you will be able to mount and to use that encrypted directory with *Boxcryptor* or with *EncFS* on other systems (e.g., the college and department servers).

4.2.3 Mobile devices

The *Boxcryptor* people keep busy! This package is also available for iOS (through the Apple Apps Store) and Android (through Google Play). Feel free to download and use it to access your encrypted cloud directories, if that sounds like fun.

5 Versioning and revision control

Cloud storage is useful (as are encrypted virtual file systems), but developers also have additional tools that are useful to creating and maintaining software, either individually or in groups. In particular, revision control software allows programmers to keep arbitrarily long history of a set of files, being able to restore the state of each file from any point in time. Indeed, revision control programs also have additional capabilities, including the splitting (*forking*) and re-joining (*merging*) of multiple paths for a set of code files. If you wish to delve more deeply into their capabilities, you should find references and tutorials on the capabilities of specific tools. For our purposes, it will be enough to track a history of changes and, in so doing, allow multiple people to draw from and contribute to a given project (that is, set of files).

5.1 Packages you should not use

There are a number of revision control packages that were once widely used but no longer are. In particular, *RCS* was used through the 1980's, while emphCVS succeeded it in the 1990's. While there are many other current packages that have, at times, enjoyed significant use (e.g., the commercial product *BitKeeper*, the most commonly used, best documented, freely available tools are described below. I strongly recommend sticking to those for now.

5.2 Subversion

In order to address the significant shortcomings of *CVS*, not to mention its awkward interface, *Subversion* was created to take its place. It provides a much kinder interface, mimicking common file system commands, and allowing for what many believe is a more common sense collection of operations. To its detriment, it tends to be slow when operating on large files or directories, and it requires a central repository for any project, copying *CVS*'s client-server architecture, thus allowing a single point of contention and failure.

If you wish to learn more about its use, you may search for the abundant documentation. Among these is a thorough, freely available textbook, *Version Control with Subversion*. A copy of this package is already installed on the college and departmental servers, as well as just about any current Linux/UNIX system. It can also easily be obtained for Mac OS X, Windows, and mobile systems.

5.3 git and github

In order to address the shortcomings of *Subversion*, Linus Torvalds (the overload of the *Linux* kernel) created *git*. This package is fully distributed—that is, there is no single, central repository for a project's files, and each copy (*clone*) of the project is equivalent to every other. Even in the presence of large files or directories, *git* is remarkably fast. To its detriment, the conceptual model behind *git* is somewhat more complex than for *Subversion*; worse, the interface—particularly the set of available commands—is a step backwards towards the unintuitive interface of *CVS*.

Nevertheless, *git* has become the most commonly used tool for revision control. A large collection of tools have been developed to make its use easier (including a number of graphical interfaces). Just as importantly, the web site *GitHub* has become a common place at which to host programming projects, where *git* is the central tool for obtaining and updating code from the projects there. By providing a single location for a *git clone* of each project, finding a given project is easier, and groups can more easily share access to a project. *GitHub* provides additional tools that make it easier to work collaboratively, and for all members of a group to see the updates of others.

Once again, if *git* or *GitHub* look interesting to you, then you should try them. Signing up for *GitHub* is an easy matter, and educational accounts are given some non-trivial resources to use. Like *Subversion*, there's a great deal of supporting documentation, including a freely available textbook, *Pro Git*. You should be aware that *GitHub* provides a great deal of support information of its own about using the site and the revision control tool.