THE UNIVERSITY OF WESTERN ONTARIO

Amazon EC2 Reference for CS 2212a/b

September 13, 2012
Author: Jeff Shantz
# Contents

1. **Introduction**  
   1

2. **Starting an Instance**  
   3

3. **Rules of Engagement**  
   6

4. **Obtaining the Hostname of Your Instance**  
   7

5. **Opening Ports**  
   8

6. **Connecting to Your Instance**  
   10
   6.1 **Windows** .................................................. 10
   6.2 **Linux/Mac** .................................................. 15

7. **Installing a Web Server**  
   16
   7.1 **Stopping/Starting/Restarting Your Web Server** ...................... 17
   7.2 **Viewing the web server logs** ................................... 17

8. **Copying Files to/from your Instance**  
   19
   8.1 **Windows** .................................................. 19
   8.2 **Linux / Mac** .................................................. 22

9. **Creating a Simple PHP Script**  
   23
   9.1 Changing Ownership of `/var/www` .................................. 23
   9.2 **Installing a Text Editor** ......................................... 24
   9.3 **Adding a Form to the Coming Soon Page** .......................... 25
   9.4 **Creating the PHP Script** ........................................ 26
   9.5 **Installing PHP** ................................................ 26
   9.6 **Investigating PHP Errors** ....................................... 27
   9.7 **Understanding POST and GET Requests** ........................ 28
   9.8 **Processing POST Data in PHP** ................................ 28
   9.9 **Saving Email Addresses to a File** ................................ 31
   9.10 **Reading From a File** ......................................... 33

10. **Configuring SSL**  
    34
    10.1 **Prerequisite** ............................................... 34
    10.2 **Enabling HTTPS in Apache** ................................ 34

11. **Troubleshooting**  
    37
    11.1 I get a PHP error about cURL .................................. 37
    11.2 I get a `ClassNotFoundException` when I try to access my applet via HTTPS 37
    11.3 My friends get an `UnsupportedClassVersionError` when they try to load my applet .................. 37
    11.4 My applet works for me but not for my friends – nothing happens ........................ 38

12. **Conclusion**  
    39
1 Introduction

Amazon Elastic Compute Cloud – EC2 – is a low-cost, scalable means of bringing applications to the Internet. In cloud computing, a cloud provider such as Amazon provides an infrastructure of computers that run virtual machines (or, as Amazon calls them, instances) that are started on demand. These virtual machines are not actually physical computers – there may be many virtual machines running on any given physical Amazon computer (known as a host).

Thanks to the beauty of virtualization technology, though, for all intents and purposes, when you connect to your virtual machine and use it, it will seem as though it is a regular, physical server and that you are the only one using it! Each virtual machine has its own operating system installed, its own applications and configuration, and its own set of virtual hardware such as a virtual hard disk, a virtual network card, etc. In fact, on any given physical host, there may be virtual machines running a mix of operating systems such as Windows and Linux. Figure 1 shows four virtual machines running on a single host.

![Figure 1: Server Virtualization](image)

Never before has it been so easy to bring an application to the Internet. In the past, to build a production-quality web site, you would have had to pay thousands up front to purchase a web server, and then pay exorbitant monthly fees to host your server with an ISP. With EC2, you can get started right away, and you are charged only a small hourly rate – as little as $0.005 per hour for a server! As your site grows, you can simply add more servers as needed to scale to your incoming traffic.

To see why this is so revolutionary, consider Netflix – the popular video streaming service. Netflix has over 26 million subscribers worldwide. Consider that a high definition movie in 1080p resolution starts at about 8 GB. Now imagine the computing infrastructure required to build a service that potentially allows 26 million users to simultaneously watch an 8 GB movie – we’re talking about 198 petabytes streamed in the span of a 2 hour movie! The costs involved in building and maintaining such an infrastructure would be astronomical and highly prohibitive. However, with Amazon’s infrastructure, Netflix was able to build the successful service that it provides today by scaling incrementally with Amazon EC2 rather than purchasing their equipment up front.

\footnote{Of course, this is an exaggeration. It is unlikely that all 26 million users would be simultaneously watching a movie.}
As of April 2012, it is estimated that Amazon EC2 accounts for about one percent of all Internet traffic – a massive number when you consider that there are currently about 2.3 billion users on the Internet. In fact, it is estimated that about one third of all Internet users will visit a web site hosted on Amazon EC2 each and every day.

So, other than Netflix, who else is using EC2? Try Dropbox, Reddit, Instagram, Pinterest, NASA, NASDAQ, Zynga (makers of Farmville), and many, many more companies. And guess what? You’re using EC2, too. Or, at least, you will be after this lab.
2 Starting an Instance

In this section, you’ll log in to the Amazon EC2 console and start an EC2 instance. You will each have your own instances in this course (donated generously by Amazon), and, when it comes time to do your group projects, each group will have one instance to serve as the group instance.

1. Go to https://cs2212.signin.aws.amazon.com/console and log in with your UWO username and the password that was provided to you.

2. Select EC2 in the list of services.

3. Click the Launch Instance button.

4. Be sure that Classic Wizard is selected and click the Continue button.

5. On the Quick Start tab, select the 32-bit radio button beside Ubuntu Server 12.04 LTS, and then click the Select button.
6. Leave everything as is on the next page. Specifically, the following should be set:

- Number of Instances: 1
- Instance Type: Micro (t1.micro, 613 MiB)
- Launch Instances: checked

Click the Continue button.

7. Check the Prevention against accidental termination checkbox and ensure that the Shutdown Behaviour is set to Stop. Leave everything else as is, and click the Continue button.

8. On the next page, leave everything as is, and click the Continue button.

9. In the Value field beside the key Name, enter your UWO username (e.g. jshantz4). This is important. We will terminate any instances that do not have valid student usernames as their names since we will not know to whom they belong.
Click the **Continue** button.

10. Select the **Create a new Key Pair** radio button, and type your **UWO username** as the name of the key pair. Next, click the **Create & Download your Key Pair** link. Be sure to save the key downloaded in a safe place. If you lose this key, you irrevocably lose access to your instance.

![Create Key Pair](image1.png)

11. Select the **Create a new Security Group** radio button and enter the following:

   - **Group name**: *Your UWO username*
   - **Group Description**: *Your full name*

   Do not change anything else at this time. Click the **Continue** button.

![Create Security Group](image2.png)

12. Click the **Launch** button to start your instance, and then click the **Close** button.
3 Rules of Engagement

Before we continue, it’s important to set some ground rules for working with Amazon EC2 in this course. We wanted to give you direct access to the Amazon EC2 console to give you practical experience with EC2 that may just come in handy for you out in the workforce (remember: many companies are using EC2!). The console allows you to work with your own instance and to stop it, reboot, and terminate it (delete it) as needed. However, it will allow you to do the same with anyone else’s instance in the class. Hence, we remind you of the following excerpt from the department’s Rules of Ethical Conduct:

The following are considered to be **scholastic offences**:

- Unauthorized inspection, alteration, deletion, publication, copying or tampering with files.

- To engage in any action which denies or unreasonably restricts the use of computer facilities to their authorized users.

Hence, tampering with another student’s or group’s instance, security group, or key pair in any way will not be tolerated and will be treated as an academic offense. This includes, but is not limited to, stopping or rebooting an instance, or modifying or deleting an instance, key pair, or security group.

In short, you can work with your own instance, security group, and key pair as needed, but please do not touch those of other students or groups.

Additionally, you may have the need to delete your instance and start over (for example, if you mess up your instance and can’t get it work properly). This is quite fine, and actually a real benefit of using virtual machines. If you mess up your instance, you can simply terminate it, start a new one, and continue working. However, the rules we will follow in this course are as follows:

- You may only ever have 1 instance running at a given time. If you decide to terminate your instance and start a new one, you must wait for the old instance to reach the **terminated** state (more on this later) before you start a new one.

- You may only ever start an instance of type `t1.micro`. Amazon EC2 provides a variety of different instance types, each with their own specifications for RAM size and CPU speed, and each with their own hourly price. Amazon has generously allocated a specific budget to us for use in this course, and to avoid exceeding this allocation, we will be using the cheapest instances available. Nevertheless, even these micro instances are more powerful than we actually require for the purposes of CS 2212, so they will do just fine.

- Your instance is intended for your CS 2212 assignments and project only. You are not permitted to use it for other purposes.

Note that your actions in the EC2 console are logged and we will be monitoring the logs frequently to detect any unauthorized actions.

Alright, enough legalese. Let’s get back to EC2.
4 Obtaining the Hostname of Your Instance

To connect to your instance, you must first obtain its hostname (Internet address). In this section, you’ll see how to do just that.

1. In the left Navigation pane, click Instances

2. Filter the list of instances by entering your UWO username in the text box above the table of instances (see Figure 2 below).

3. You should see your instance in the list. If not, click the Refresh button a few times until it appears.

4. Wait until your instance is in the running state, and then select it in the list.

5. In the lower pane, the hostname of your instance will be listed. Your hostname will be of the form ec2-XXX-YYY-ZZZ-WWW.compute-1.amazonaws.com. Copy this and keep it somewhere safe, as you’ll be needing it quite a bit in this course.

Figure 2: Obtaining the hostname of your instance
5 Opening Ports

Any given Internet server might be hosting multiple services, such as a web server, FTP server, email server, etc. When you connect to such a server, how does the server know which service you are requesting? For example, when you type www.google.com in your browser, how does the server hosting www.google.com know that you want to request content from its web server and not any of the other services it might be running? Services on a machine are identified by port numbers. For example, a web server listens on port 80, while an FTP server listens on port 22. Thus, when you connect to www.google.com in your web browser, your web browser is actually connecting to www.google.com and requesting the service listening on port 80. Indeed, any Internet request consists of both an address and a port number, although you don’t normally see this because the software that you use – such as your browser – conceals this fact.

Recall that when you created your instance, you also created a security group. Your security group is the firewall for your instance – it specifies the ports on the instance that should be open and those that should be blocked. By default, an Amazon EC2 security group is configured to block all ports, meaning that no traffic will be accepted by your instance on any port. Even if you had a web server listening on port 80 on your instance, if you did not configure your security group to allow port 80, then no requests would ever get through to the server.

This is generally a good thing, because we want to keep our instances locked down as much as possible for security reasons. However, we need to be able to connect to our servers via SSH if we want to be able to do anything meaningful with them, so we will need to open the SSH port (port 22) in our security groups.

1. In the left Navigation pane, click Security Groups.

2. Filter the list of instances by entering your UWO username in the text box above the table of security groups.

3. Select your security group and click the Inbound tab.

4. In the Create a new rule drop down list, select SSH.
5. Click the **Add Rule** button and then the **Apply Rule Changes** button.

That’s it! Your instance will now accept connections on port 22 – the SSH port.
6 Connecting to Your Instance

6.1 Windows

In this section, you’ll see how to connect to your instance using Windows. If you’re using Linux or Mac, skip to the next section.

1. Go to [http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) and download the following files under the heading **For Windows on Intel x86**:

   - puttygen.exe
   - putty.exe

2. To connect to your instance using PuTTY, we must first convert the private key downloaded from the Amazon console into a format that PuTTY understands. To do this, open puttygen.exe and select **File > Load private key**.

3. Choose **All Files (*.*)** in the drop down filter and then select the .pem file that you downloaded in step ???. Click **Open**.
4. Click **OK** to dismiss the PuTTYgen Notice.

5. Click the **Save private key** button.

6. Click **Yes** to the PuTTYgen Warning to save the key without specifying a passphrase for it.

7. Save the file somewhere safe (do not lose it). Make sure that it has the extension `.ppk`. 
8. Open `putty.exe`. In the left pane, expand the **Connection** node and select **Data**. Configure the following:

- **Auto-login username**: `ubuntu`

9. Under the **Connection** node in the left pane, expand the **SSH** node and select the **Auth** node. Configure the following:

- **Private key file for authentication**: `<path to the .ppk file you saved in step ???>`
10. In the left pane, select the **Session** node and configure the following:

- **Host Name**: `<hostname you obtained in step ???>`
- **Port**: 22 (the SSH port)
- **Connection type**: SSH

11. In the **Saved Sessions** field, enter **ec2** and click **Save** to save the profile so that you can use it in the future without having to specify all these settings again.

12. To connect to your instance, double-click **ec2** in the **Saved Sessions** list.
13. Click **Yes** to the **PuTTY Security Alert**. You will receive this message only once for each server to which you connect.

14. If all goes according to plan, you should be connected and presented with a prompt. Type `exit` and press **Enter** to close the connection.
6.2 Linux/Mac

In this section, you’ll see how to connect to your instance using Linux or Mac. If you’re using Windows, skip to the next section.

1. Open a terminal window and enter the following commands to create your SSH directory and lock it down:

   ```bash
   mkdir -p ~/.ssh
   chmod 700 ~/.ssh
   ```

2. Copy the `.pem` file that you downloaded in step ?? into your `.ssh` directory and lock it down with 600 permissions:

   ```bash
   cp ~/Downloads/jshantz4.pem ~/.ssh
   chmod 600 ~/.ssh/jshantz4.pem
   ```

3. Connect to your instance via SSH, substituting the appropriate filename and hostname below (this should be the hostname you obtained in step ??).  

   ```bash
   ssh -i ~/.ssh/jshantz4.pem ubuntu@ec2-107-20-33-116.compute-1.amazonaws.com
   ```

   This command says:
   
   • SSH into a server
   • using the identity file (private key) `~/.ssh/jshantz4.pem`
   • the username should be `ubuntu`
   • and the hostname of the server is `ec2-107-20-33-116.compute-1.amazonaws.com`

4. Type `exit` to disconnect from the server.

This method works fine, but it’s a bit of a pain to type in such a long command to connect to the server. Fortunately, we can do better using the SSH configuration file.

5. Edit the file `/.ssh/config`. This file will likely not already exist on your system.

6. Enter the following in the file, substituting the correct identity filename and hostname:

   ```
   Host ec2
   Hostname ec2-107-20-33-116.compute-1.amazonaws.com
   User ubuntu
   IdentityFile `~/.ssh/jshantz4.pem`
   ```

7. Save the file and close your editor.

8. You can now type the following to connect to your instance:

   ```bash
   ssh ec2
   ```
7 Installing a Web Server

Because both the individual assignment and the project in this course consist of a web-based application, we will need a web server to serve our application to visitors. A popular and widely-used web server is Apache HTTP Server (also known as simply Apache) and we will make use of that server in this course. This section details the installation of Apache.

1. SSH in to your instance – see Section 6.1 (Windows) or Section 6.2 (Linux/Mac) for a reminder.

The package manager used on Ubuntu is called the Advanced Packaging Tool, or APT. This tool allows us to download and install new software on our server. Before we install anything, we first want to ensure that the list of packages that APT knows about is up-to-date so that we don’t end up downloading and installing old packages (software).

2. Enter the following command:

```
apt-get update
```

Observe that you receive an error. This is because the user under whom you are logged in (ubuntu) is not the root user; in other words, your current user account does not have permission to update the package list on the system. To perform a task as the root user, we use the command `sudo` (superuser do).

3. Enter the following command:

```
sudo apt-get update
```

The `apt-get` command will take a few moments to download updated package lists.

4. To install Apache, enter the following:

```
sudo apt-get install apache2
```

Type `Y` and press Enter when prompted to confirm the installation.

5. In a web browser, try visiting your instance to see if the web browser responds. For example, if the hostname of your instance is `ec2-107-20-33-116.compute-1.amazonaws.com`, then you will visit `http://ec2-107-20-33-116.compute-1.amazonaws.com`.

Observe that the server does not respond. This is because, as indicated in Section 5, all ports on the instance are blocked by default. Just as we opened the SSH port earlier, we will need to open the HTTP port (80).

6. Open the HTTP port in the EC2 console (see Section 5).

7. In a web browser, try visiting your instance again. You should see the following if everything is working correctly:

```
It works!
This is the default web page for this server.
The web server software is running but no content has been added, yet.
```
7.1 Stopping/Starting/Restarting Your Web Server

On occasion, you may have need to restart your web server. When we change any of the web server’s configuration, or install new plugins (called Apache modules) for the server, we have to restart it. Table 1 displays commands that you may need to use on occasion:

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start the web server</td>
<td>sudo service apache2 start</td>
</tr>
<tr>
<td>Stop the web server</td>
<td>sudo service apache2 stop</td>
</tr>
<tr>
<td>Restart the web server</td>
<td>sudo service apache2 restart</td>
</tr>
<tr>
<td>Print the status of the web server</td>
<td>sudo service apache2 status</td>
</tr>
</tbody>
</table>

Table 1: Commonly used Apache commands

7.2 Viewing the web server logs

Each time someone visits your web site, the web server adds an entry with some information to its log file. It maintains two log files:

- **access.log** - Logs normal accesses to the site
- **error.log** - Logs any erroneous requests to the site; useful for debugging server-side scripts (e.g. PHP scripts)

By default, these logs are stored in the directory `/var/log/apache2`.

1. Print the contents of **access.log**:

```
cat /var/log/apache2/access.log
25.214.159.201 - - [13/Sep/2012:06:27:29 +0000] "GET / HTTP/1.1" 200 483 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/21.0.1180.89 Safari /537.1"
```

Briefly, the first column is the visitor’s IP address, followed by the date/time and the path that was requested (in this case, a GET request was issued for the root path `/`). The request was successful (HTTP code 200) and the user visited the site in Google Chrome on a Mac.

2. Print the contents of **error.log**:

```
cat /var/log/apache2/error.log
```

3. We will try simulating a few errors. First, enter the following command:

```
tail -f /var/log/apache2/error.log
```

The **tail** command displays the end of a file, while the **-f** flag tells **tail** to wait for more data to be added to the file, and to display it as it is added. This is useful in debugging as we can issue the **tail -f** command on a log file, and then visit our site in our browser and watch what gets added to the log.
4. In your browser, go to http://your-instance/fakepage.html where your-instance is the hostname of your instance. You will get an error since this page does not exist.

5. Switch back to the terminal and observe that a line has been added to error.log reporting the error:

   [Thu Sep 13 06:47:26 2012] [error] [client 25.214.159.201] File does not exist: /var/www/fakepage.html

6. Press Ctrl+C to quit tail.
8 Copying Files to/from your Instance

You will frequently need to transfer files back and forth between your computer and your instance. This can be accomplished using the Secure Copy (SCP) protocol. In this section, we’ll see how to SCP files to/from our instances.

8.1 Windows

In this section, you’ll learn how to SCP files to/from your instance using WinSCP on Windows. If you are using Linux or Mac, please see Section 8.2.

1. In your browser, download the file [http://bit.ly/RL0FDa](http://bit.ly/RL0FDa). This is a tarball that you will upload to your instance using WinSCP.

2. Next, go to [http://winscp.net/eng/download.php](http://winscp.net/eng/download.php) and download the latest stable Installation Package. Alternatively, if you are using a lab computer or do not otherwise want to install WinSCP, you can download the Portable executables, which do not require installation.

3. Install the downloaded WinSCP package. During installation, when asked to select a user interface style, select **Commander interface**.

4. Run WinSCP and enter the following information:

   - **Host name**: Your EC2 hostname
   - **Port number**: 22
   - **Username**: ubuntu

   In the **Private Key** field, enter the path to the `.ppk` file you saved in Section 6.1.
5. Click the **Save** button to save the session settings for future connections, and save it under a descriptive name such as **CS 2212 Instance**.

6. In the **Stored Sessions** list, double-click the session that you saved.

7. Click **Yes** when you receive a warning.
8. You are now connected to your instance. The files displayed on the left are on your local computer, while the files on the right are on your instance. In the left pane, find the \texttt{cs2212.tar.gz} file that you downloaded in step 1, and drag it to an empty space on the right.

9. Click \textbf{Copy} in the dialog that appears.

10. After the file is successfully copied, you may close WinSCP.

You have now successfully copied a file to your instance. Copying files from your instance back to your local computer is as simple as dragging from right-to-left instead of left-to-right.
8.2 Linux / Mac

In this section, you’ll learn how to SCP files to/from your instance using Linux or Mac. If you are using Windows, please see Section 8.1.

1. In your browser, download the file http://bit.ly/RL0FDa and save it in a path you’ll remember (such as your home directory). This is a `tarball` that you will upload to your instance using the `scp` program.

2. Open a terminal window.

3. Type the following command:

   ```bash
   scp ~/Downloads/cs2212.tar.gz ec2:
   ```

   This assumes that you saved the file in `~/Downloads`. Also, note the colon (:) at the end of the command. This tells `scp` to upload the file `cs2212.tar.gz` to your home directory on your instance.

4. If you wanted to copy a file to a different directory other than your home directory, you could specify this after the colon, e.g.

   ```bash
   scp file.txt ec2:/var/www
   ```

5. If you needed to copy an entire directory, you would use the `-r` flag, e.g.

   ```bash
   scp -r dir1/ ec2:
   ```

6. Finally, if you wanted to copy the file `file.txt` from the home directory on your instance back to your local computer, you could issue the following command:

   ```bash
   scp ec2:file.txt .
   ```

   Note the `.` at the end. This indicates that you want `scp` to copy the file into the current directory on your local computer.
9 Creating a Simple PHP Script

In this section, we’ll configure Apache to serve PHP scripts and build a simple PHP script to ensure everything is working. In doing so, we may just learn a little bit about HTML and PHP along the way. Before doing so, however, a little bit of background knowledge is useful.

On Ubuntu, files served by Apache are located in /var/www. When you visit http://your-instance in your browser, Apache looks for a file called index.html in this /var/www. If it finds it, it will send it to your browser where it will be displayed. Similarly, if you had a directory /var/www/test and you visited http://your-instance/test in your browser, then Apache would look for a file /var/www/test/index.html and send that to your browser.

By default, /var/www is owned by root, meaning that we will not be able to add, edit, or delete files within this directory under the ubuntu user. To solve this and make our lives a little easier, we will change ownership of this directory to the ubuntu user. Otherwise, we would have to use sudo before each and every change that we made.

9.1 Changing Ownership of /var/www

1. SSH in to your instance.

2. Change to the directory /var:

   cd /var

3. List the contents of the directory and notice that the www directory is owned by root:

   ls -l

   .
   .
   drwxr-xr-x 2 root root 4096 Sep 13 06:23 www

4. Change ownership of the www directory and all of its files and subdirectories to the ubuntu user:

   sudo chown -R ubuntu www

5. List the contents of the directory again and notice that www is now owned by ubuntu:

   ls -l

   .
   .
   drwxr-xr-x 2 ubuntu root 4096 Sep 13 06:23 www

   This means that we can now write to this directory.

2 Alternatively, we could switch to the root user, but this is considered a security risk.
6. Change to `/var/www` and list the contents of the file `index.html`. Notice that this file contains the **It Works!** text that you saw earlier when you visited your instance in your web browser.

```bash
cd www
cat index.html
```

7. Delete the file `index.html` and untar the contents of the `cs2212.tar.gz` file that we uploaded to the instance into the current directory:

```bash
rm index.html
tar zxvf ~/cs2212.tar.gz
ls -l
drwxr-xr-x 4 ubuntu ubuntu 4096 Sep 13 08:30 cs2212
```

8. In your browser, visit the URL `http://your-instance/cs2212`. You should see a **Coming Soon** page. Note: if you are using a browser Internet Explorer, this page may not display as nicely as it might in other browsers.

---

### 9.2 Installing a Text Editor

Before we can make changes to our Coming Soon page, we will first need to install a text editor. The choice of text editor in Linux/UNIX can greatly affect your productivity, as well as your self-worth. Table 2 gives the commands needed to install and use your favourite editor, along with the social status you can expect to attain depending on your choice of editor.

<table>
<thead>
<tr>
<th>Editor</th>
<th>Installation/Use</th>
<th>Audience</th>
<th>Probable Social Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nano</code></td>
<td><code>apt-get install nano</code>&lt;br&gt;<code>nano file.txt</code></td>
<td>Newbs</td>
<td>People will laugh at you behind your back.</td>
</tr>
<tr>
<td><code>emacs</code></td>
<td><code>apt-get install emacs</code>&lt;br&gt;<code>emacs file.txt</code></td>
<td>Weenies</td>
<td>People will laugh at you behind your back.</td>
</tr>
<tr>
<td><code>vim</code></td>
<td><code>apt-get install vim</code>&lt;br&gt;<code>vim file.txt</code></td>
<td>Hardcore Ninjas</td>
<td>People will revere you; one day, they will write songs about you.</td>
</tr>
</tbody>
</table>

---

*As a self-respecting second-year Computer Science student, and one who will one day go out and represent Western in the world, you are implored to use a good, standards-compliant browser like Google Chrome or Mozilla Firefox. Ignoring its dismal security record, Internet Explorer has historically held back advances in web design due to Microsoft’s unwillingness to adhere to web standards. This costs companies a great deal of money as they have to spend time implementing special hacks to get their pages to look right in Internet Explorer. So, use a real browser. You are then invited to close your eyes, picture an Internet Explorer user, and shake your head condescendingly. Bonus points if you snort in derision.*
1. Taking into account Table 2 install your text editor of choice on your instance.

9.3 Adding a Form to the Coming Soon Page

We wish to add a form to our Coming Soon page to allow a user to enter their email address and sign up to receive updates when the site is launched. In this section, we’ll see how to do just that.

1. Change to the `cs2212` directory in `/var/www` and open the file `index.html` in your editor of choice:

   ```
   cd cs2212
   vim index.html
   ```

2. If you are not familiar with HTML, take a few moments to compare the source code of the page to the page you see in your browser. Observe the following tags:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>html</td>
<td>Starts and ends every HTML document</td>
</tr>
<tr>
<td>head</td>
<td>Provides metadata for the page, such as its title.</td>
</tr>
<tr>
<td>title</td>
<td>Specifies the page title</td>
</tr>
<tr>
<td>link</td>
<td>Includes CSS (Cascading Style Sheets) styles into the page to spruce it up</td>
</tr>
<tr>
<td>body</td>
<td>Starts and ends the page content</td>
</tr>
<tr>
<td>h1</td>
<td>Starts and ends a large block of header text</td>
</tr>
<tr>
<td>p</td>
<td>Starts and ends a paragraph.</td>
</tr>
</tbody>
</table>

3. After the ending `p` tag, add the following form:

   ```html
   <form method="post" action="subscribe.php">
     <input type="text" name="email" placeholder="Email address" />
     <input type="submit" value="Sign Up" />
   </form>
   ```

4. We will discuss the specifics of this form shortly, but, for now, save the file and visit the site in your browser. You should now see the form displayed on the page. Compare the source code of the form that you added to what you see on the page. Observe that the `input type="text"` produces the field where an email address can be entered. Similarly, the `input type="submit"` produces the Sign Up button.
5. Click the **Sign Up** button and observe that you get an error indicating that subscribe.php does not exist. This is because we have not implemented it yet. However, it illustrates the purpose of the `action="subscribe.php"` attribute in the form that we added to the page:

```html
<form method="post" action="subscribe.php">
  
  In essence, this says, “*when this form is submitted, send everything entered into the form to the file subscribe.php.*”

9.4 Creating the PHP Script

We now have a form on the page, but we have nothing that will take action on the server when the form is submitted. Thus, we must implement subscribe.php.

1. In your favourite editor, create a file subscribe.php.

2. Enter the following:

   ```php
   <?php
   echo phpinfo();
   ?>
   ```

3. Save the file and exit your editor.

4. Try submitting the form again in your browser. Notice that Apache serves up the PHP script as a download, rather than executing it. This is because we have not yet installed PHP.

9.5 Installing PHP

To run PHP scripts, we must first install PHP, along with a PHP module for Apache that allows Apache to communicate with PHP.

1. Install PHP and the PHP module for Apache with the following command:

   ```bash
   sudo apt-get install php5 libapache2-mod-php5
   ```

2. Restart Apache to ensure that the module is enabled:

   ```bash
   sudo service apache2 restart
   ```

3. Visit your form in your browser again and click the **Sign Up** button again. Observe that, this time, we do not receive the script as a download, but we do receive an error. We’ll investigate this in the next section.
9.6 Investigating PHP Errors

At times, our PHP scripts will produce errors and it will be difficult to identify the source of those errors from within the browser. As we saw earlier, each time Apache encounters an erroneous request, it logs the request to the file `/var/log/apache2/error.log`. This log is your friend.

- Display the contents of `/var/log/apache2/error.log`:

```bash
cat /var/log/apache2/error.log
```

```bash
[Thu Sep 13 10:54:58 2012] [error] [client 25.214.159.201] PHP
```

Observe the following information:

- The error occurred in `/var/www/cs2212/subscribe.php` on line 3
- The referring page was `http://ec2-107-20-33-116.compute-1.amazonaws.com/cs2212/`
- The cause of the error was due to the `php_info()` function that we called in `subscribe.php`. This function does not exist.

- Oops! We made a typo in `subscribe.php`. Before we fix it, though, let's change the PHP configuration so that errors display in the browser.

- Edit the file `/etc/php5/apache2/php.ini` as `root`:

```bash
sudoedit /etc/php5/apache2/php.ini
```

- At the end of the file, add the following lines:

```plaintext
error_reporting = E_ALL & ~E_DEPRECATED & ~E_NOTICE
display_errors = On
display_startup_errors = On
```

- Save and close the file and then restart the web server:

```bash
sudo service apache2 restart
```

- Resubmit the form in your web browser. You should now see the error right in the browser.

- Fix the error by editing the file `subscribe.php` and changing `php_info()` to `phpinfo()`.

- Resubmit the form in your browser. You should now see the default PHP info screen. This means that PHP is properly installed and that Apache is able to properly communicate with it.
9.7 Understanding POST and GET Requests

Recall the attribute `method="post"` that we added to our form:

```html
<form method="post" action="subscribe.php">
</form>
```

1. In your browser, return to the form and enter something in the email address field. Click the `Sign Up` button.

2. Observe that the URL to which you are redirected is of the form:
   
   ```
   http://your-instance/cs2212/subscribe.php
   ```

3. Open `index.html` and change `method="post"` to `method="get"`.

4. Submit the form again. Observe that the URL to which you are redirected is now of the form:
   
   ```
   http://your-url/cs2212/subscribe.php?email=jeff@example.com
   ```

A GET request places the form data in the *query string* of the URL, while a POST request does not. Instead, with a POST request, the form data is sent to the server behind the scenes. It is often undesirable to have the form data placed in the query string since it makes it easier for a user to tamper with the data that gets sent to your script. However, even with POST, it is quite easy to do so.

More often, the choice between a GET and a POST request depends on semantics. Usually the rule of thumb is approximately as follows:

- If the action to be performed will not affect the server state in any way, then a GET request is used. For example, if we were accessing a calendar and specifying the month to display using the URL `http://calendar.com/display.php?month=11`

- If the action to be performed will affect the server state (such as creating, editing, or deleting something on the server), then a POST request is used. For example, a user sign up form on a web site would use a POST request, since the result of the form would be a user being created on the server.

Because our mailing list form would presumably save the user’s email address in a database in a real-life scenario, a POST request is appropriate here.

9.8 Processing POST Data in PHP

Recall that the email field in our form was specified as follows:

```html
<input type="text" name="email" placeholder="Email address" />
```

Here, the `name` attribute is important. When the form data gets posted to the server, it is placed into a dictionary called `$_POST`. The key used to identify the value of a particular form element is the value of its `name` attribute.

Hence, in the case of our form, we should expect to find the user’s email address in `$_POST[‘email’]`.

1. Open the file `index.html` and change `method="get"` back to `method="post"`. Save and close the file.
2. Open the file `subscribe.php` in your favourite text editor and remove the `echo phpinfo();` line.

3. Modify the file so that it appears as follows:

```php
<?php
    $email = $_POST['email'];
    if ($email) {
        echo "Thank you. $email has been added to my mailing list.";
    } else {
        echo "Please enter your email address.";
    }
?>
```

4. Save the file and try submitting the form with and without an email address.

This looks OK, but it would be nice if the resulting pages had the same style as we have on our form page. Fortunately, this is simple to accomplish. We can intermingle PHP code and HTML quite easily, as we'll see below.

5. Open `index.html` and copy the first 7 lines (up to and including the `body` tag).

6. Open `subscribe.php` and modify it as follows, pasting the copied 7 lines at the top:

```html
<html>
<head>
<title>Coming Soon!</title>
<link href='http://fonts.googleapis.com/css?family=Lato:400,700,900,400italic' rel='stylesheet' type='text/css' />
<link rel="stylesheet" type="text/css" href="styles/screen.css" />
</head>
<body>
<?php
    $email = $_POST['email'];
    if ($email) {
        <h1>Thank you</h1>
        <p>&lt;?= $email ?&gt; has been added to my mailing list.&lt;/p&gt;
    }
?>
```
Observe that we begin chunks of PHP code with <?php and end them with ?>. If we need to include something from PHP-land in our HTML, we can do so using <?= ... ?>. We did this above to include the user’s email address in the HTML output, e.g. <?= $email ?>.

7. Try submitting the form again with and without an email address. Things should look a little nicer now.

Things are looking better, but the mix of PHP and HTML in our script is making things a bit messy. Fortunately, we can include files in our PHP scripts, so we can put the header and footer of the page in separate files and clean things up a bit.

8. Open subscribe.php and cut the first 7 lines from the file (up to and including the body tag)

9. Paste these lines into a new file called header.php.inc. We could choose any name for this file, but a standard convention is to use an extension of .inc to indicate that the file is going to be included in another PHP script.

10. Open subscribe.php and cut the last 2 lines from the file (the closing body and html tags)

11. Paste these lines into a new file called footer.php.inc

12. Open subscribe.php and modify it to look as follows:

```php
include 'header.php.inc';
$email = $_POST[‘email’];
```
if ($email) {
    <h1>Thank you</h1>
    <p>&lt;?= $email ?&gt; has been added to my mailing list.</p>
} else {
    <h1>Error</h1>
    <p>Please enter your email address.</p>
} include 'footer.php.inc';

13. Resubmit the form both with and without an email address and ensure that everything is working properly.

9.9 Saving Email Addresses to a File

We’ll finish off our subscribe.php by having it write each email address submitted to a file. PHP provides the functions fopen, fwrite, and fclose that will come in handy for this task.

1. Open subscribe.php and modify it as follows:

```php
include 'header.php.inc';
$database = "addresses.txt";
$email = $_POST['email'];

if ($email) {
    # Open a file handle
    $fh = fopen($database, 'w') or die("<h1>Can’t open file</h1>");
```
2. Try submitting the form with an email address. You should get the message `Can’t open file`.

3. At the terminal, enter the command:

```
ps aux | grep apache
```

The `ps` command shows all processes running on the server. Observe that Apache is running as the `www-data` user. This user does not have access to write to the directory `/var/www/cs2212`. This is a good thing, since, for security reasons, we don’t generally want to allow the web server process to be able to write to the server. However, we do want it to be able to write to `addresses.txt` only, so we’ll create an empty file and change its owner to the user `www-data` so that Apache can write to it.

4. Create an empty file `addresses.txt`:

```
touch addresses.txt
```

5. Change its owner to the user `www-data`:

```
sudo chown www-data addresses.txt
```

6. Try resubmitting the form again with an email address. This time, it should work.

7. Inspect the contents of `addresses.txt` to make sure the email address was written properly:

```
cat addresses.txt
```

8. Resubmit the form again and inspect the contents of `addresses.txt` again. Notice that any previous addresses that were written to the file have been overwritten. This is because we used the file mode `'w'` when we issued the `fopen` command. Instead, we probably want to use `'a'` (append).

9. Edit `subscribe.php` and change the file mode in the `fopen` call to `'a'`.

10. Resubmit the form several times and ensure that each email address is saved in the file on a new line.
Reading from a file is just as easy in PHP using the `fread` command. Let's whip up a quick script to display the contents of addresses.txt.

1. Create a file `list.php` and add the following to it:

```php
<?php

include 'header.php.inc';

# Open a file handle for reading
$fh = fopen('addresses.txt', 'r');

<?php

<h1>Mailing List</h1>

<ul>
<?php
cline = null;

while ($line = fgets($fh)) {

    <li><?= $line ?></li>

<?php
}

</ul>

<?php
fclose($fh);
include 'footer.php.inc';
?>
```
10 Configuring SSL

In preparation for developing an application that communicates with Facebook, we need to enable HTTPS (secure HTTP) support on our web servers. This is because Facebook requires an encrypted connection between it and your server. This section will detail the process of enabling HTTPS on your server.

10.1 Prerequisite

Before completing this section, a subdomain of cs2212.ca will need to pointed at your instance. Your instructor will notify you once this has been done. Your subdomain will be:

\[
\text{uwo_username.cs2212.ca}
\]

Hence, you will be able to access your instance via the URL \text{uwo_username.cs2212.ca} instead of the long hostname that you have been using so far.

Do not complete this section until you have been advised that a subdomain has been pointed to your instance. This is because the certificate file used to provide HTTPS access to your instance is only valid for *.cs2212.ca – it will not provide a secure connection for any other hostname.

10.2 Enabling HTTPS in Apache

In this section, you will configure Apache for HTTPS access.

1. SSH in to your instance.

2. Download the certificates:
   \[
   \]

3. Extract the certificates:
   \[
   \text{tar zxvf certs.tar.gz}
   \]

4. Move files \text{star.cs2212.ca.crt} and \text{cs2212-ca-bundle.crt} to /etc/ssl/certs:
   \[
   \text{sudo mv star.cs2212.ca.crt cs2212-ca-bundle.crt /etc/ssl/certs}
   \]

5. Move the file \text{star.cs2212.ca.key} to /etc/ssl/private:
   \[
   \text{sudo mv star.cs2212.ca.key /etc/ssl/private}
   \]

6. Download the Apache configuration files:
   \[
   \text{wget http://bit.ly/PFnCE8 -O apache.tar.gz}
   \]

7. Extract the Apache configuration files:
   \[
   \text{tar zxvf apache.tar.gz}
   \]

8. Delete the existing Apache site configuration file:
9. Move the file 000-cs2212 to /etc/apache2/sites-enabled:

```
sudo mv 000-cs2212 /etc/apache2/sites-enabled
```

10. Move the file cs2212.conf to /etc/apache2/conf.d:

```
sudo mv cs2212.conf /etc/apache2/conf.d
```

11. As root, edit the file /etc/apache2/sites-enabled/000-cs2212 to insert the appropriate ServerName lines as shown below, according to the subdomain that has been assigned to your instance:

```
sudo vim /etc/apache2/sites-enabled/000-cs2212

<VirtualHost *:80>
    ServerName jshantz4.cs2212.ca:80
    Include conf.d/cs2212.conf
</VirtualHost>

<VirtualHost *:443>
    ServerName jshantz4.cs2212.ca:443
    Include conf.d/cs2212.conf
    SSLCertificateFile /etc/ssl/certs/star.cs2212.ca.crt
    SSLCertificateKeyFile /etc/ssl/private/star.cs2212.ca.key
    SSLCACertificateFile /etc/ssl/certs/cs2212-ca-bundle.crt
</VirtualHost>
```

12. Enable the SSL Apache module:

```
sudo a2enmod ssl
```

13. Restart the web server:

```
sudo service apache2 restart
```

14. In the EC2 console, open the HTTPS port (port 443) in your security group.

15. In your browser, try accessing your subdomain using HTTPS. For instance, if your UWO username was jshantz4, you would access the URL https://jshantz4.cs2212.ca. The request should be successful and, depending on your browser, you should see a lock icon somewhere in or near the browser bar.

16. You may wish to now take this opportunity to update your SSH configuration (either in ~/.ssh/config for Linux/Mac) or in PuTTY (for Windows). Change your hostname to your new subdomain. This way, if you ever terminate your instance and start a new one,
you won’t have to change the hostname at that time. Instead, your SSH client will simply be configured to connect to your `username.cs2212.ca` subdomain. Of course, if you ever terminate your instance and start a new one, please inform your instructor so that your subdomain can be pointed to your new instance. Note that it may take several hours for a DNS change (e.g. pointing your subdomain to a new server) to propagate to all DNS servers worldwide.
11 Troubleshooting

11.1 I get a PHP error about cURL
Install the `php5-curl` package and restart the web server.

```
sudo apt-get install php5-curl
sudo service apache2 restart
```

11.2 I get a ClassNotFoundException when I try to access my applet via HTTPS
As root, edit the file `/etc/apache2/sites-enabled/000-cs2212` and make sure that you have the appropriate `ServerName` lines as shown below, according to the subdomain that has been assigned to your instance:

```
sudo vim /etc/apache2/sites-enabled/000-cs2212
```

```
<VirtualHost *:80>
  ServerName jshantz4.cs2212.ca:80
  Include conf.d/cs2212.conf
</VirtualHost>

<VirtualHost *:443>
  ServerName jshantz4.cs2212.ca:443
  Include conf.d/cs2212.conf
  SSLCertificateFile /etc/ssl/certs/star.cs2212.ca.crt
  SSLCertificateKeyFile /etc/ssl/private/star.cs2212.ca.key
  SSLCACertificateFile /etc/ssl/certs/cs2212-ca-bundle.crt
</VirtualHost>
```

Once these lines are present, restart the web server and try again:

```
sudo service apache2 restart
```

11.3 My friends get an UnsupportedClassVersionError when they try to load my applet
The problem is almost certainly that you are compiling `.class` files using a specific version of the JDK (e.g. 1.7.4), but your friends have a lower version of Java running on their systems. So, when they try to run the applet, their Java runtime does not know how to load the `.class` file developed for a higher version of Java.

The solution is to tell the Java compiler to target a specific Java version. It is probably safest to target Java 1.5, as everyone these days should be running at least Java 1.5. If you were compiling `Test.java`, you would enter:

```
javac -source 1.5 -target 1.5 Test.java
```

This will ensure that the `.class` file built is targeted to Java 1.5. Have your friends try again after you have recompiled. Note that Java often caches `.class` files, so your friends may either...
need to kill the \texttt{java.exe} process using Task Manager, or, alternatively, reboot their systems before reloading your page.

Of course, it is tedious to have to type the above command each time you want to recompile your applet. Hence, it is recommended that you create a short script named \texttt{build} in the directory containing your code on the server. The script should contain the following:

```
#!/bin/sh
javac -source 1.5 -target 1.5 $@
```

Next, make this script executable using \texttt{chmod +x build}. Finally, when you want to build your files, you can simply use this script:

```
./build File1.java File2.java ...
```

11.4 My applet works for me but not for my friends – nothing happens

Try the solutions to the previous two questions.
12 Conclusion

Congratulations! You’re now a seasoned developer ready to produce hardcore web apps for the cloud. We will add to this document as the semester progresses and questions/issues come up, so please check it out on the course web site every now and then.