

What's happening CEG2350?

Aiden Cox - CEG 2350 Lab Lead

Austin Kellough - CEG 2350 Lab Assistant

Quote of the week:

“A person who makes no mistakes is the person not doing anything”

How was Lab10?

The image displays three side-by-side screenshots from a computer interface.

- Screenshot 1 (Left):** A GitHub repository page for "CEG2350". The "main" branch is selected. The interface shows 10 branches and 0 tags. A search bar at the top says "Find or create a branch...". Below it, there are tabs for "Branches" and "Tags". A list of branches includes "main" (selected), "aidenc17-examReview-2", "api", "blue", "dotopots", "pattonsgirl-patch-1", "pwmgr", "reviewGuideUpdate", "rubrics-koppin", and "yellow". At the bottom, there's a link "View all branches".
- Screenshot 2 (Middle):** A terminal window titled "wildflow@wildflow-UB: ~". It shows the user running apt commands to update the package list and install build-essential dependencies. The terminal output includes:

```
wildflow@wildflow-UB: ~$ sudo apt update
[sudo] password for wildflow:
Hit:1 http://in.archive.ubuntu.com/ubuntu kinetic InRelease
Hit:2 http://in.archive.ubuntu.com/ubuntu kinetic-updates InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu kinetic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu kinetic-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
5 packages can be upgraded (use --show-upgraded to see them).
wildflow@wildflow-UB: ~$ sudo apt install build-essential
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9ubuntu1).
build-essential set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
wildflow@wildflow-UB: ~$ lsb_release -v
lsb_release: Version:Ubuntu 22.04 LTS
```
- Screenshot 3 (Right):** A code editor showing a Makefile. The file contains the following content:

```
Paul: main.cpp message.cpp message.h
1   output: main.o message.o
2     g++ main.o message.o -o output
3
4   main.o: main.cpp
5     g++ -c main.cpp
6
7   message.o: message.cpp message.h
8     g++ -c message.cpp
9
10  target: dependencies
11    action
12
```

The word "cpp" is highlighted in blue under the "target" line.

Git branching? Compilers?

Beginning Lab11

Lab Instructions: <https://pattonsgirl.github.io/CEG2350/Labs/Lab11/Instructions.html>

Lab Template:

<https://raw.githubusercontent.com/pattonsgirl/CEG2350/refs/heads/main/docs/Labs/Lab11/LabTemplate.md>

Feeling comfortable with the course pace?
Questions from lecture?

Branch Control - Part 0

Part 0 - branch control

Determine and **create** one or more GitHub Issues for this lab.

Determine and **create** one or more branches to resolve the Issues for this lab.

In the end, your lab submission (README .md) should be visible on the **main** branch.

At least one issue and one branch for the lab!

`Tar` it up - Part 1

1. For the `tar` command, fill in the table describing the most frequently used flags:

tar Option	Description
-c	
-v	
-f	
-z	
-x	
-t	

1. tar and compress folders of your choice using gzip compression. Write the command you used.

Do NOT add and commit your tar and compressed file to GitHub. It does not need to be in your repository folder and may in fact be too large to push to GitHub.

1. Determine *at least* one method to confirm your tar and compressed file is a tar and compressed file. Just stating it has a `.tar.gz` extension will not be sufficient for credit.

[TAR\(1\)](#)

GNU TAR Manual

[TAR\(1\)](#)

NAME

`tar` – an archiving utility

SYNOPSIS

Traditional usage

`tar {A|c|d|r|t|u|x} [GnSkUW0mpsMBiajJzZhPlRvwo] [ARG...]`

UNIX-style usage

`tar -A [OPTIONS] -f ARCHIVE ARCHIVE...`

`tar -c [-f ARCHIVE] [OPTIONS] [FILE...]`

`tar -d [-f ARCHIVE] [OPTIONS] [FILE...]`

`tar -r [-f ARCHIVE] [OPTIONS] [FILE...]`

For this part, we are looking to `tar` or `zip` up some folders.

Fill out the chart, and then zip up a folder!

STF- I mean SFTP – Part 2

sftp Command	Description
ls	
lls	
cd	
lcd	
pwd	
lpwd	
put	
get	
exit /bye	

Confirm you are on your local system using the terminal you normally use to ssh to your instance. *Do not ssh to your instance*

1. Connect to your AWS instance via sftp.
2. Retrieve the tar and compressed file (ending with .tar.gz) from your AWS instance and download it to your local system
3. Decompress & extract the file (.tar.gz) to a folder on your system
 - You can remove it after you get the command right, I'm not a space hog

For this part, we are looking to use SFTP to move our previously `tar`d file.

Fill out the chart, and then bring back your zipped folder to your local machine

STF- I mean SFTP – Part 2

How to use: Think of SSH!

```
~ > ssh -i ~/Keys/ceg2350.pem ubuntu@52.201.160.131
```

```
~ > sftp -i ~/Keys/ceg2350.pem ubuntu@52.201.160.131
Connected to 52.201.160.131.
sftp>
```

```
`sftp -i <path/to/keyfile> ubuntu@<yourIPaddress>`
```

Or, we can use our config file for `ssh`. More on that later

Extract and Profit - Part 3

Part 3 - Extract & Profit

For this part, you will be installing an open source web server to your AWS instance, extracting a compressed archive containing a static website, and testing that it works!

Do the following on your AWS instance.

- Useful commands: `apt`, `systemctl`, `wget`, `chown`, `chmod`, `tar`, `vim`, `curl`
1. Install `apache2` or `nginx` with `apt`
 2. Confirm that your chosen web serving service is running
 3. Download `simple-site.tar.gz` to your AWS instance
 - URL to download `.tar.gz` file
 4. Change the permissions for the folder `/var/www/html` so that your user is the owner and primary group, your user (and members of your group) can read and write, and others can only read.
 5. Extract the compressed archive to `/var/www/html`
 6. Edit the `index.html` file in `/var/www/html` with:
 - your name where `YOUR LAST NAME HERE` is. Make this `div` visible
 - replace `Insert something fun here` with a fact about you

Website Website Website!

We are making a static website on this part!

`README.md` will include a screenshot of working website with URL included.

```
<section>
  <h2>CEG 2350 Lab 11</h2>
  <div style="display:none">YOUR NAME HERE</div>
  <h3>Facts about me</h3>
  <p>Insert something fun here</p>
</section>
```

Extract and Profit - Part 3

```
ubuntu@ceg2350-sandbox:~$ wget https://github.com/pattonsgirl/CEG2350/blob/main/Labs/Lab11/simple-site.tar.gz
```

Grab file from github on the CEG2350 Course public repo.

```
ubuntu@ceg2350-sandbox:~$ sudo apt install <apache or nginx>
```

Install one of the open source web servers

```
<section>
    <h2>CEG 2350 Lab 11</h2>
    <div style="display:none">YOUR NAME HERE</div>
    <h3>Facts about me</h3>
    <p>Insert something fun here</p>
</section>
```

Follow rest of instructions to make your website!

Extract and Profit - Part 3

Should look something like this!

The screenshot shows a web browser window with the following details:

- Address Bar:** Not secure | 44.215.23.55
- Toolbar:** Import favorites, wings, Pilot, Cengage, ZyBooks, GitHub, McGraw, 1, 2, AWS Learner Lab, Other favorites
- Content Area:**
 - Welcome to My CEG 2350 Website**
 - Navigation menu:
 - [Home](#)
 - [About](#)
 - [Contact](#)
 - CEG 2350 Lab 11**
 - Austin Kellough
 - Facts about me**
 - I have 4 dogs Willow, Magic, Zoey, Whiskey
 - © 2024 CEG 2350 Website

`ssh` Keys - Part 4

- Fill in the following table of options commonly used in a config file for ssh:

ssh config Option	Description
Host	
HostName	
User	
Port	
IdentityFile	

- On your local system, make a new key pair - with a non-default name. Change at minimum the default name
- On your AWS instance, create a user & user home directory or use the your `firstinitiallastname` account
- Put the public key of the key pair you created in the AWS instnace user's `~/.ssh/authorized_keys` file on the AWS instance
- From your local system, ssh in to the AWS instance using the user's username and the private key of the keypair you created.
 - Format reminder: `ssh -i path/to/privatekey username@hostname_or_ip`
- Write an entry in your local system's `~/.ssh/config` file with the new connection information.
- Write the `ssh` command that will use your `config` file information if correctly entered.

- Useful commands: `adduser`, `getent passwd <username>`, `ssh-keygen`, `vim`, `ssh`

- Fill in the following table of commonly configured ssh files:

ssh File Path / Name	Purpose
<code>~/.ssh/config</code>	
<code>~/.ssh/id_rsa</code>	
<code>~/.ssh/id_rsa.pub</code>	
<code>~/.ssh/id_ed25519</code>	
<code>~/.ssh/id_ed25519.pub</code>	
<code>~/.ssh/authorized_keys</code>	
<code>~/.ssh/known_hosts</code>	

More fill out tabely things

`ssh` Keys - Part 4

1. On your local system, make a new key pair - with a non-default name. Change at minimum the default name
2. On your AWS instance, create a user & user home directory or use the your `firstinitiallastname` account
3. Put the public key of the key pair you created in the AWS instance user's `~/.ssh/authorized_keys` file on the AWS instance
4. From your local system, ssh in to the AWS instance using the user's username and the private key of the keypair you created.
 - Format reminder: `ssh -i path/to/privatekey username@hostname_or_ip`
5. Write an entry in your local system's `~/.ssh/config` file with the new connection information.
6. Write the `ssh` command that will use your `config` file information if correctly entered.

Rest of lab listed here. More of a walkthrough, next slide is pieces of a config file and how to make!

`ssh` Keys - Part 4

```
~ > ssh -i ~/Keys/ceg2350.pem ubuntu@52.201.160.131
```



```
1 Host aws  
2   HostName 54.204.2.113  
3   User ubuntu  
4   Port 22  
5   IdentityFile ~/Keys/ceg2350.pem  
6
```



```
~ > ssh aws
```

`ssh` Keys - Part 4

Your SSH command to this point looks like:

```
`ssh -i <path/to/keyfile> <username>@<hostname>`
```

We can move this fields into a `config` file in our `~/.ssh` directory

```
1 Host aws
2   HostName 54.204.2.113
3   User ubuntu
4   Port 22
5   IdentityFile ~/Keys/ceg2350.pem
6
```

Move your fields into their corresponding places! The indentations matter!

The top `Host` can be named whatever you want. Whatever you name it will be what you use to `ssh` in if using the `config` file.

Have a Good Weekend!

Don't hesitate to reach out and ask questions!

Quote of the week:

“A person who makes no mistakes is the person not doing anything”