Linux Commands and Shell Scripting - Final Project

Estimated time needed: 90 minutes

Congratulations!! You have finished the modules. Now is the time to put your skills to test. Read through the scenario below.

This lab may use some bash concepts we haven't yet covered in this course. Whenever this happens, we will provide sufficient hints and/or the code for you.

Scenario

You are a lead linux developer at the top-tech company "ABC International INC." ABC currently suffers from a huge bottleneck - each day, interns must painstakingly access encrypted password files on core servers, and backup those that were updated within the last 24-hours. This introduces human error, lowers security, and takes an unreasonable amount of work.

As ABC INC's most trusted linux developer, you have been tasked with creating a script `backup.sh` which automatically backs up any of these files that have been updated within the past 24 hours.

Objectives

- The objective of this lab is to incorporate much of the shell scripting you've learned over this course into a single script.
- You will schedule your shell script to run every 24 hours using `crontab`.
- TIP: If you're unsure whether some of your code will work as wanted, you can try the command directly in the terminal - and even create your own test scripts!

Task 0

1. Open a new terminal by clicking on the menu bar and selecting **Terminal->New Terminal**:

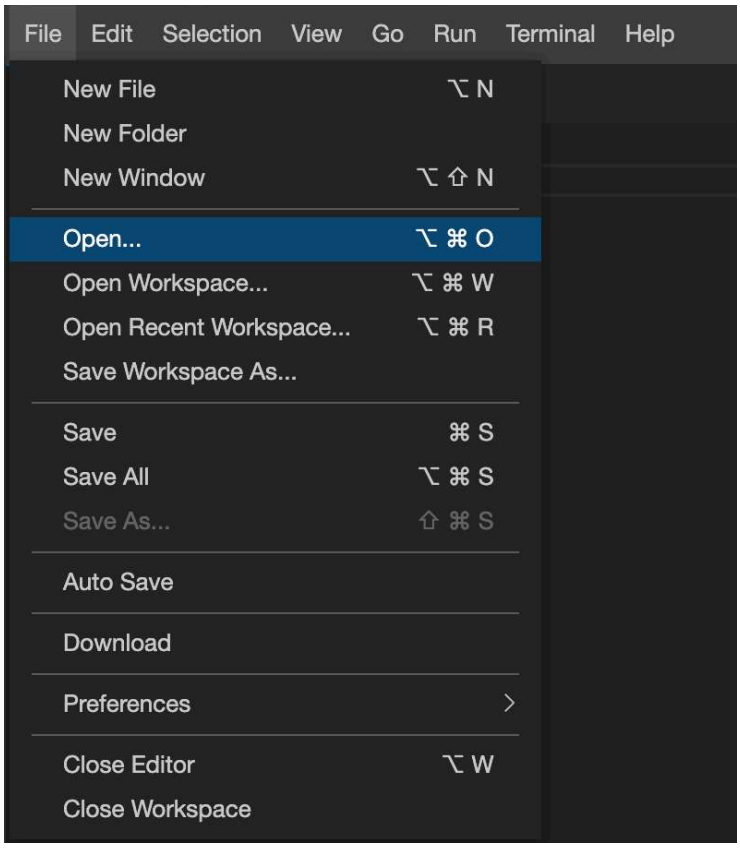
2. Download the template file `backup.sh` by running the command below:

```
1
```

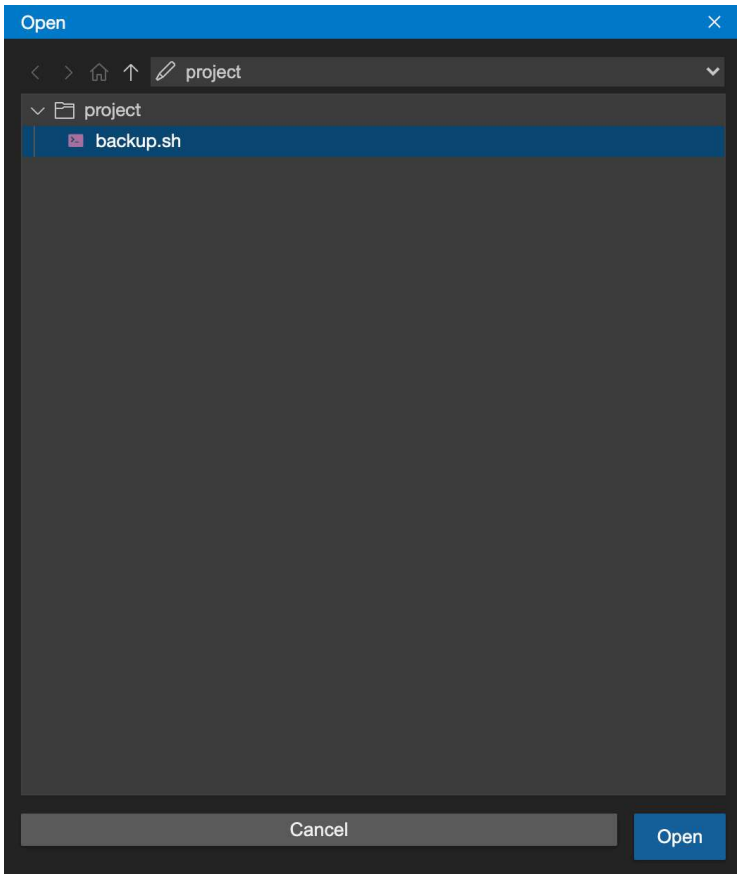
```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-LX0117EN-SkillsNetwork/labs/Final%20Project/backup.sh
```

Copied!

3. Open the file in the IDE by clicking **File->Open** as seen below:



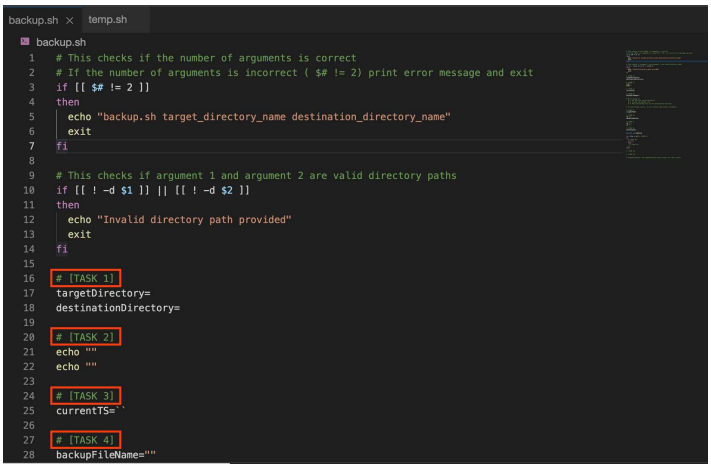
and then click on the file, which should have been downloaded in the project directory:



About the template script, `backup.sh`

1. You will notice the template script contains comments (lines starting with the `#` symbol). Do **not** delete these.

- The ones that look like "`# [TASK {number}]`" will be used by your grader:

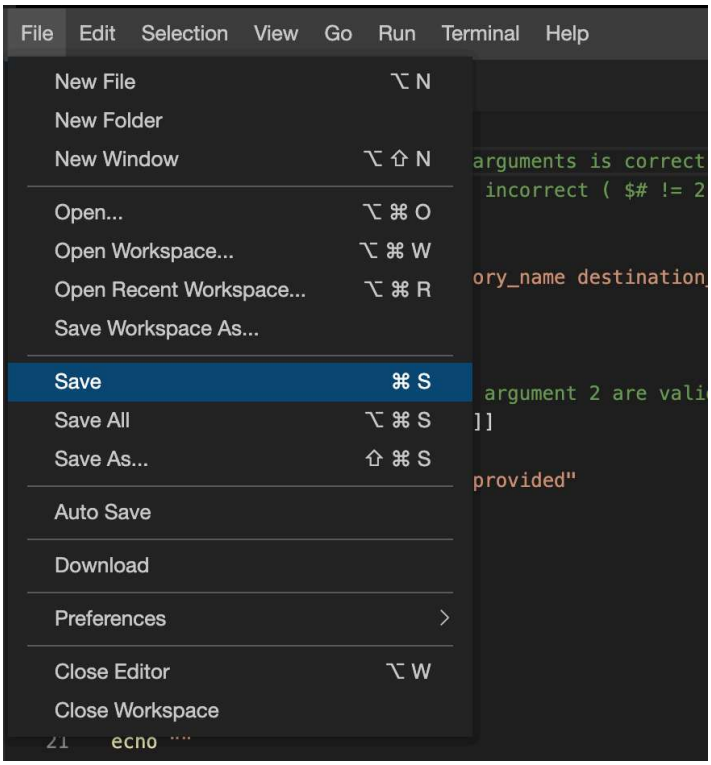


2. Also, please do **not** modify any existing code above `# [TASK 1]` in the script.

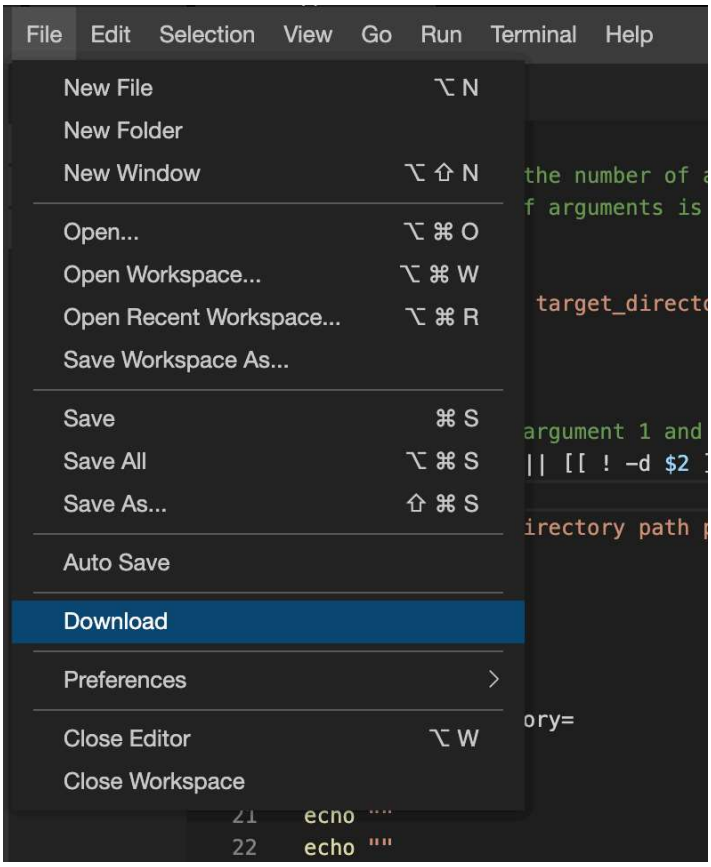
Your work **will not** be saved if you exit your session.

In order to save your progress:

1. Save the current working file (`backup.sh`) with `CTRL-S` [Windows/Linux], `CMD-S` [MAC], or navigate to **File->Save** as seen below:



2. Download the file to your local computer by navigating to **File->Download** as seen below:



3. Unfortunately, our editor does **not** currently support file uploading, so you will need to use 'copy and paste' as follows:
    - To "upload" your in-progress `backup.sh` file and continue working on it:
        1. Open a terminal and type `touch backup.sh`
        2. Open the empty `backup.sh` file in the editor
        3. Copy paste the contents of your locally-saved `backup.sh` file into the empty `backup.sh` file in the editor

Task 1

Navigate to `# [TASK 1]` in the code.

Set two variables equal to the values of the first and second command line arguments, as follows:

1. Set `targetDirectory` to the first command line argument
2. Set `destinationDirectory` to the second command line argument

(This task is meant to help with code readability)

▼ Click here for Hint

The command line arguments interpreted by the script can be accessed via `$1` (first argument) and `$2` (second argument)

Task 2

1. Display the values of the two command line arguments in the terminal.

▼ Click here for Hint

Rememeber, you can use the command `echo` as a print command.

- Ex: `echo "The year is $year"`

Task 3

1. Define a variable called `currentTS` as the current timestamp, expressed in seconds.

▼ Click here for Hint

Remember you can customize the output format of the `date` command.

- To set a variable equal to the output of a command you can use command substitution: `$()` or `` ` ` ``
  - For example: `currentYear=$(date +%Y)`

Task 4

1. Define a variable called `backupFileName` to store the name of the archived and compressed backup file that the script will create.

- The variable `backupFileName` should have the value `"backup-[$currentTS].tar.gz"`
  - For example, if `currentTS` has the value `1634571345`, then `backupFileName` should have the value `backup-1634571345.tar.gz`.

Task 5

1. Define a variable called `origAbsPath` with the absolute path of the current directory as the variable's value.

▼ Click here for Hint

You can get the absolute path of the current directory using the `pwd` command.

Task 6

1. Define a variable called `destAbsPath` with value equal to the absolute path of the destination directory.

▼ Click here for Hint

First use `cd` to go to `destinationDirectory`, and then use the same method you used in **Task 5**

Checkpoint

Friendly reminder to save your work to your local computer!

Task 7

1. Change directories from the current working directory to the target directory `targetDirectory`.

▼ Click here for Hint

`cd` into the original directory `origAbsPath` and then `cd` into `targetDirectory`.

Task 8

You need to find files that have been updated within the past 24 hours.
This means you need to find all files who's last modified date was 24 hours ago or less.

To do make this easier:

1. Define a numerical variable called `yesterdayTS` as the timestamp (in seconds) 24 hours prior to the current timestamp, `currentTS`

▼ Click here for Hint

Math can be done using `$(())`; for example: `zero=$((3 * 5 - 6 - 9))`

- Thus, to get the timestamp in seconds of 24 hours *in the future*, you would use:
  - `tomorrowTS=$(($currentTS + 24 * 60 * 60))`

Note on arrays

You will notice the line:

```
1
```

```
declare -a toBackup
```

Copied!

in the script.

This line declares a variable called `toBackup`, which is an array.
An array contains a list of values, and items can be appended to arrays using the syntax:

```
1
```

```
myArray+=($myVariable)
```

Copied!

When you print (or `echo`) an array you will see its string representation, which is simply all of its values separated by spaces:

```
1
2
3
4
5
6
```

```
$ declare -a myArray
$ myArray+=("Linux")
$ myArray+=("is")
$ myArray+=("cool!")
$ echo ${myArray[@]}
Linux is cool!
```

Copied!

This will be useful later in the script where you will pass the array `$toBackup`, consisting of the names of all files that need to be backed up, to the `tar` command.
This will archive all files at once!

Task 9

1. Within the `$()` expression inside the `for` loop, write a command that will return all files and directories in the current folder.

▼ Click here for Hint

There is a very clean way of doing this using `ls`

Task 10

1. Inside the for loop, you want to check whether the `$file` was modified within the last 24 hours.

- To get the last-modified date of a file in seconds, use `date -r $file +%s`
- Then compare the value to `yesterdayTS`
  - Idea: `if [[ $file_last_modified_date > $yesterdayTS ]]` then the file was updated within the last 24 hours!

2. Since much of this wasn't covered in the course, for this task you may copy the code below and paste it into the double round brackets `(())`:

```
1
```

```
`date -r $file +%s` > $yesterdayTS
```

Copied!

Task 11

1. In the `if-then` statement, add the `$file` that was updated in the past 24-hours to the `toBackup` array.
2. Since much of this wasn't covered in the course, you may copy the code below and place after the `then` statement for this task:

```
1. 1

1. toBackup+=($file)
```

Copied!

Checkpoint

Friendly reminder to save your work to your local computer!

Task 12

1. After the `for` loop, **compress** and **archive** the files, using the `$toBackup` array of filenames, to a file with the name `backupFileName`.

▼ Click here for Hint

Use `tar -czvf $backupFileName ${toBackup[@]}`

Task 13

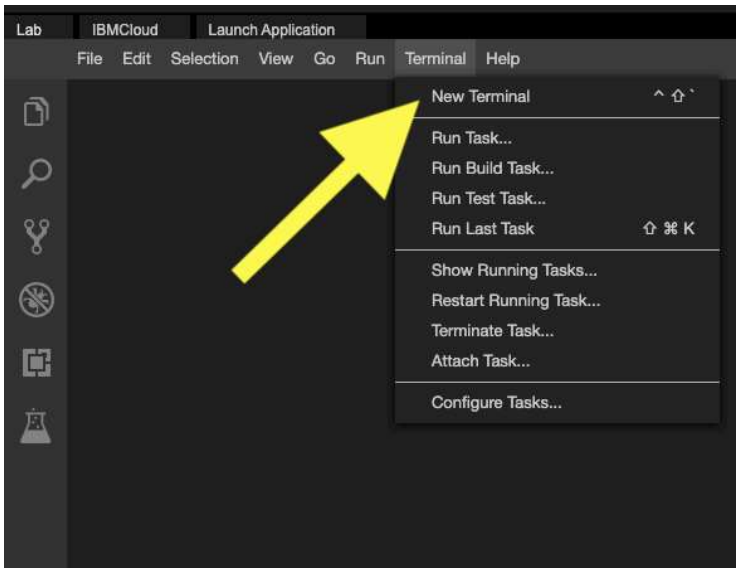
Now the file `$backupFileName` is created in the current working directory.

1. Move the file `backupFileName` to the destination directory located at `destAbsPath`.

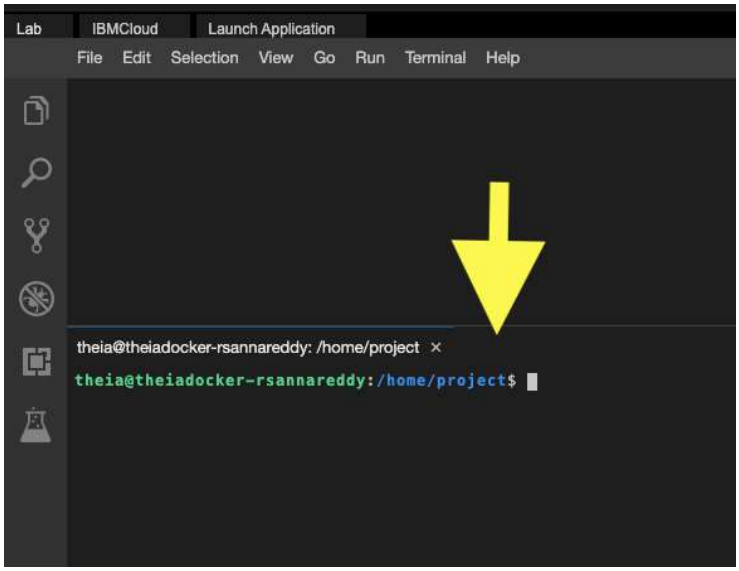
You are now done the coding portion of the lab!

Task 14

1. Open a new terminal by clicking on the menu bar and selecting **Terminal->New Terminal**, as in the image below:



- This will open a new terminal at the bottom of the screen as seen below:



Task 15

1. Save the file you're working on (`backup.sh`) and make it executable.

▼ Click here for Hint

Use the `chmod` command with the correct options

2. Verify the file is executable using the `ls` command with the `-l` option:

```
1
ls -l backup.sh
```

Copied!

3. Take a screenshot of the output of the command above and save as `15-executable.jpg` (or `.png`)

Task 16

1. Download the following `zip` file with the `wget` command:

```
1
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-LX0117EN-SkillsNetwork/labs/Final%20Project/important-documents.zip
```

Copied!

2. Unzip the archive file:

```
1
```

```
unzip -DDo important-documents.zip
```

Copied!

(`-DDo` to overwrite and not restore original modified date)

Update the file's last modified date to now:

```
1
```

```
touch important-documents/*
```

Copied!

3. Test your script using the following command:

```
1
```

```
./backup.sh important-documents .
```

Copied!

4. This should have created a file called `backup-[CURRENT_TIMESTAMP].tar.gz` in your current directory
5. Take a screenshot of the output of `ls -l` and save as `16-backup-complete.jpg` (or `.png`)

Task 17

1. **Copy** (don't `mv`) the `backup.sh` script into the `/usr/local/bin/` directory.
    - Note: You may need to use `sudo cp` in order to create a file in `/usr/local/bin/`
2. Test the cronjob to see if the backup script is getting triggered by scheduling it for every 1 minute.

▼ Click here for Hint

```
1
```

```
*/1 * * * * /usr/local/bin/backup.sh /home/project/important-documents /home/project
```

Copied!

3. Please note that since the Theia Lab is a virtual environment, we need to explicitly start the cron service using the below command.

```
1
```

```
sudo service cron start
```

Copied!

4. Once the cron service is started, please check in the directory (`/home/project`) if the tar files are getting created.
5. If yes, then please stop the cron service using the below command, else it will continue to create tar files every minute.

```
1
```

```
sudo service cron stop
```

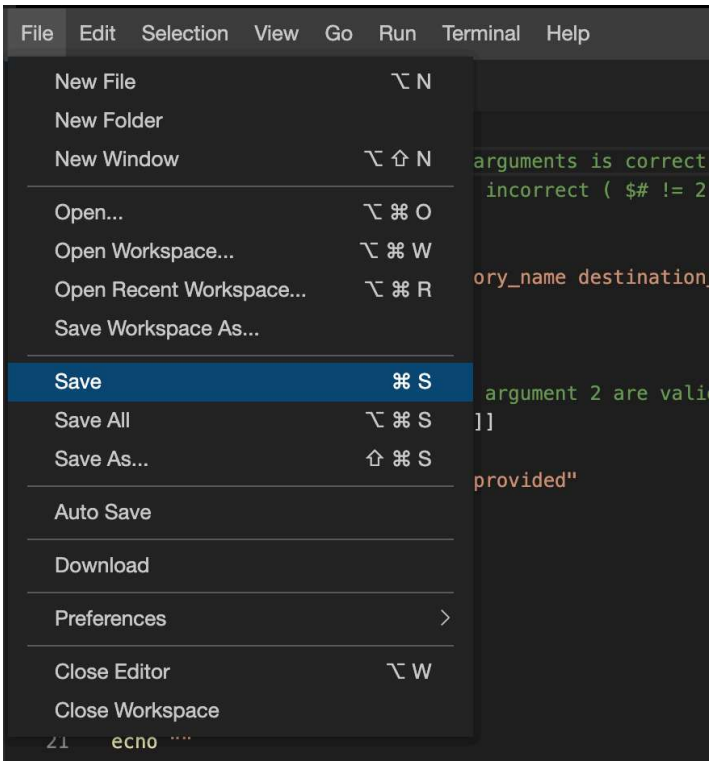
Copied!

6. Using `crontab`, schedule your `/usr/local/bin/backup.sh` script to backup the `important-documents` folder every 24 hours to the directory (`/home/project`).
7. Take a screenshot of the output of `crontab -l` and save as `17-crontab.jpg` (or `.png`)

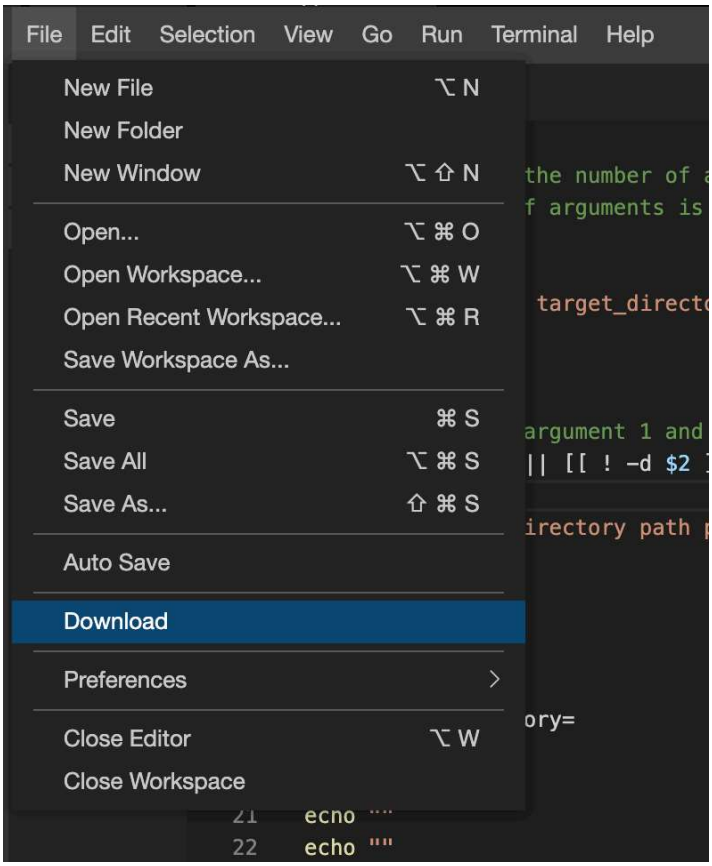
   Ensure the cron service is running or start the cron service if need be, when you are setting up cron jobs in a real-life scenario.

Task 18

1. Save the current working file (`backup.sh`) with CTRL-S [Windows/Linux], ⌘-S [MAC] or navigating to **File->Save** as seen below:



2. Download the file to your local computer by navigating to **File->Download** as seen below:



(You may save the file as `backup.sh`)

3. You will later submit this file will for peer-grading.

Congratulations!

You have completed the final lab for this course!