Linux Commands and Shell Scripting - Final Project

Estimated time needed: **90** minutes Welcome to the hands-on lab for the final project! In this scenario, you are a lead Linux developer at the top-tech company ABC International Inc. As one of ABC Inc.'s most trusted Linux developers, you have been tasked with creating a script called `` `backup.sh` `` which runs every day and automatically backs up any encrypted password files that have been updated in the past 24 hours. Please complete the following tasks, and be sure to follow the directions as you go. Don't forget to save your work.

Getting started

Note:

Please save your project and capture screenshots wherever necessary.

1. As mentioned in the **Final Project Overview Criteria**, you can submit your project deliverables through either Option 1: AI-Graded Submission and Evaluation or Option 2: Peer-Graded Submission and Evaluation.
2. **For Option 1: AI-Graded Submission and Evaluation:**
    - Submission requires the **code snippet of backup.sh file for Task 1 and the terminal output** for Task 2 to 5.
3. **For Option 2: Peer-Graded Submission and Evaluation:**
    - Submission requires **screenshots for Tasks 1 to 17**, except **Task 14**, which requires uploading the **backup.sh** file.

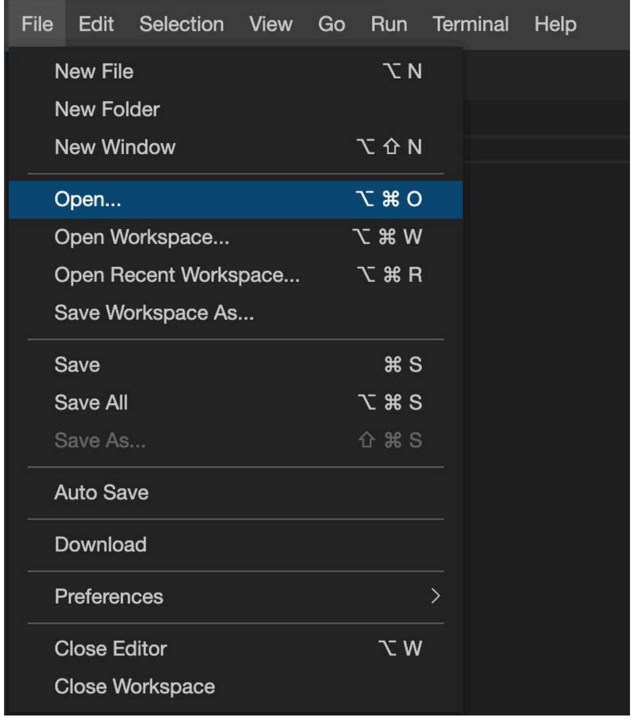
Task 0

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:

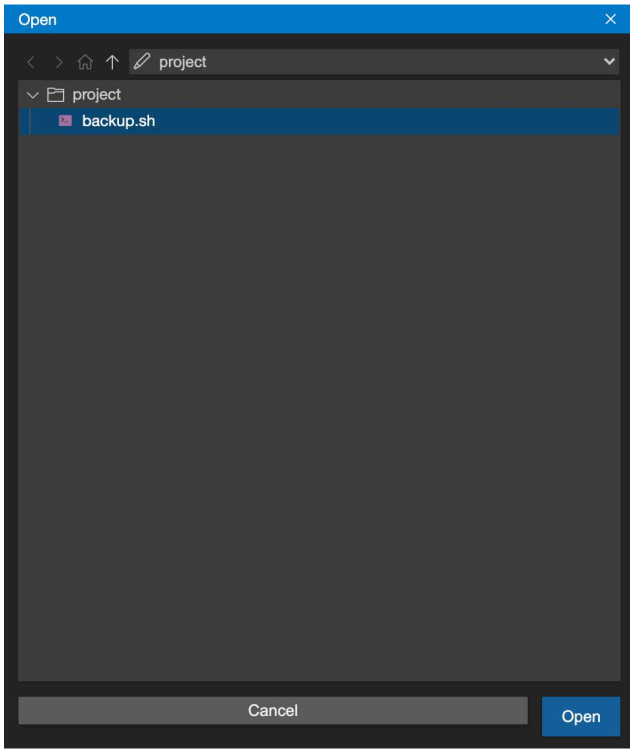
plaintext

```
wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/pWN3k
```

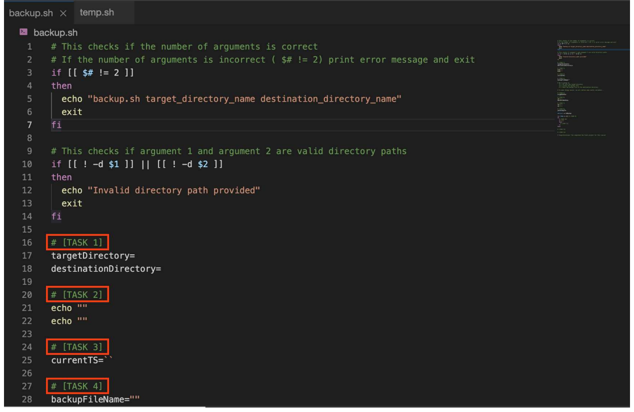
3. Open the file in the IDE by clicking **File**->**Open** as seen below:



then click on the file, which should have been downloaded to your `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:

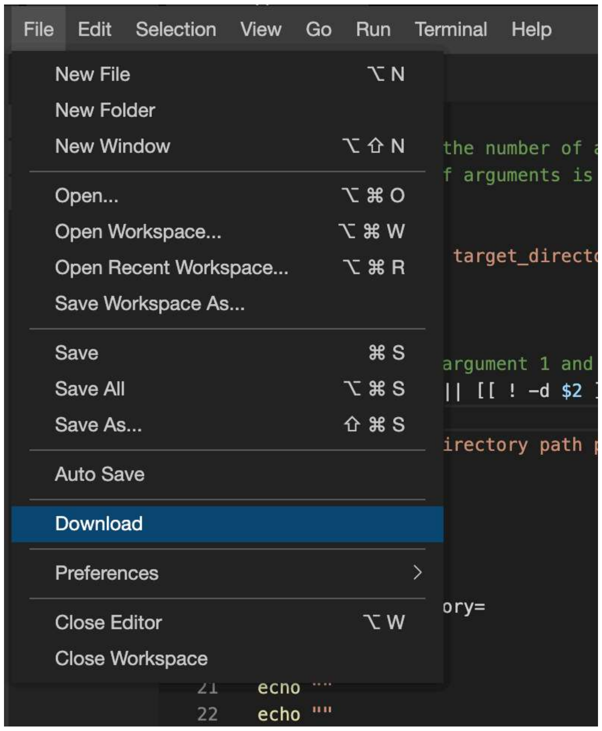


```
# This checks if the number of arguments is correct
# If the number of arguments is incorrect ( $# != 2) print error message and exit
if [[ $# != 2 ]]
then
  echo "backup.sh target_directory_name destination_directory_name"
  exit
fi

# This checks if argument 1 and argument 2 are valid directory paths
if [[ ! -d $1 ]] || [[ ! -d $2 ]]
then
  echo "Invalid directory path provided"
  exit
fi

# [TASK 1]
targetDirectory=
destinationDirectory=

# [TASK 2]
echo ""
echo ""

# [TASK 3]
currentTS=``

# [TASK 4]
backupFileName=""
```

2. Also, please do __not__ modify any existing code above `# [TASK 1]` in the script. ## Saving your progress 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:

```plaintext
<img src="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud
```

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 copy and paste your work 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

Implementiong the Final Project

Assessment: For Option 1: AI-Graded Submission and Evaluation: For Tasks 1–13, ensure that you save a local copy of your ``` `backup.sh` ``` file. The complete code, containing the implementation will be required for final submission and evaluation.

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

Assessment: >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code showing the first and second command line arguments assigned to variables and save it as \`01-Set_Variables.jpeg\` or \`01-Set_Variables.png\`. ## Task 2 1. Display the values of the two command line arguments in the terminal.

▶ Click here for Hint

Assessment: >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that displays the values of the two command line arguments and save it as \`02-Display_Values.jpeg\` or \`02-Display_Values.png\`. ## Task 3 1. Define a variable called \`currentTS\` as the current timestamp, expressed in seconds.

▶ Click here for Hint

Assessment: >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that defines the `currentTS` variable and save it as `03-CurrentTS.jpeg` or `03-CurrentTS.png`. ## 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`. **Assessment:** >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that defines the `backupFileName` variable and save it as `04-Set_Value.jpeg` or `04-Set_Value.png`. ## 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

Assessment: >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that defines the `origAbsPath` variable and save it as `05-Define_Variable.jpeg` or `05-Define_Variable.png`. ## Task 6 1. Define a variable called `destAbsPath` whose value equals the absolute path of the destination directory.

▶ Click here for Hint

>Note: Please Note that you can also use the cd "destinationDirectory" || exit which ensures that if the specified directory is incorrect or inaccessible, the script will terminate immediately at this step. This acts as an implicit validation check to confirm that the correct directory is provided before proceeding with further operations. Follow the same for Task 7 . **Assessment:** >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that defines the `destAbsPath` variable and save it as `06-Define_Variable.jpeg` or `06-Define_Variable.png`.

Checkpoint

Friendly reminder to save your work to your local computer! ::page{title=""} ## Task 7 1. Change directories from the current working directory to the target directory `targetDirectory`.

▶ Click here for Hint

Assessment: >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that changes the directory to the target directory and save it as `07-Change_Directory.jpeg` or `07-Change_Directory.png`. ## Task 8 You need to find files that have been updated within the past 24 hours. This means you need to find all files whose 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

Assessment: >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that defines the `yesterdayTS` variable and save it as `08-YesterdayTS.jepg` or `08-YesterdayTS.png`.

Note on arrays

In the script, you will notice the line: ` ` ` declare -a toBackup ` ` ` This line declares a variable called `toBackup`, which is an **array**. An array contains a list of values, and you can append items to arrays using the following syntax: ` ` ` myArray+=($myVariable) ` ` ` When you print or `echo` an array, you will see its string representation, which is simply all of its values separated by spaces: ` ` ` $ declare -a myArray $ myArray+=("Linux") $ myArray+=("is") $ myArray+=("cool!") $ echo ${myArray[@]} Linux is cool! ` ` ` This will be useful later in the script where you will pass the array `$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! ::page{title=""} ## Task 9 1. In the for loop, use the wildcard to iterate over all files and directories in the current folder.

▶ **Click here for Hint**

Assessment: >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that uses a wildcard in a for loop to list all files and directories and save it as `09-List_AllFilesandDirectoriess.jpeg` or `09-List_AllFilesandDirectoriess.png`. ## 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`. > >`if [[ $file_last_modified_date -gt $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 square brackets `[[]]`: ` ` ` `date -r $file +%s` -gt $yesterdayTS ` ` ` **Assessment:** >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that performs the modification time check inside the if condition and save it as `10-IF_Statement.jpeg` or `10-IF_Statement.png`. ## Task 11 1. In the `if-then` statement, add the `$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: ` ` ` toBackup+=($file) ` ` ` **Assessment:** >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that adds the file to the `toBackup` array and save it as `11-Add_File.jpeg` or `11-Add_File.png`.

Checkpoint

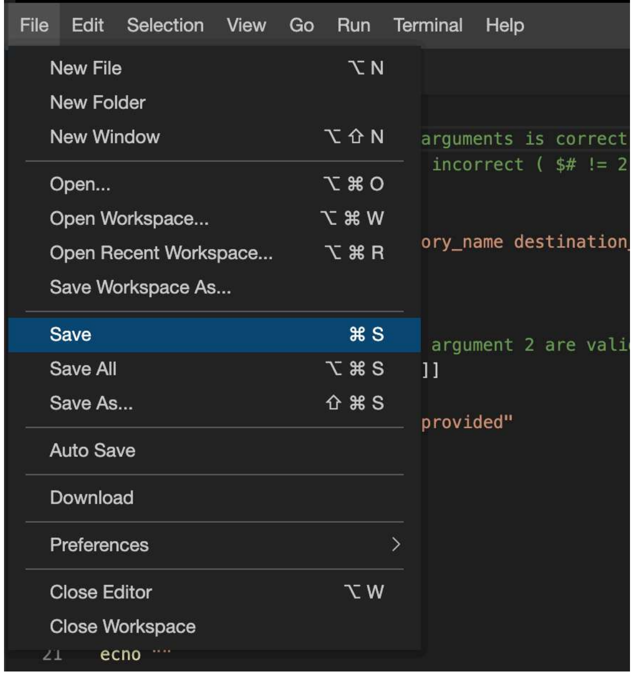
Friendly reminder to save your work to your local computer! ::page{title=""} ## Task 12 1. After the `for` loop, **compress** and **archive** the files, using the `$toBackup` array of filenames, to a file with the name `backupFileName`.

▶ Click here for Hint

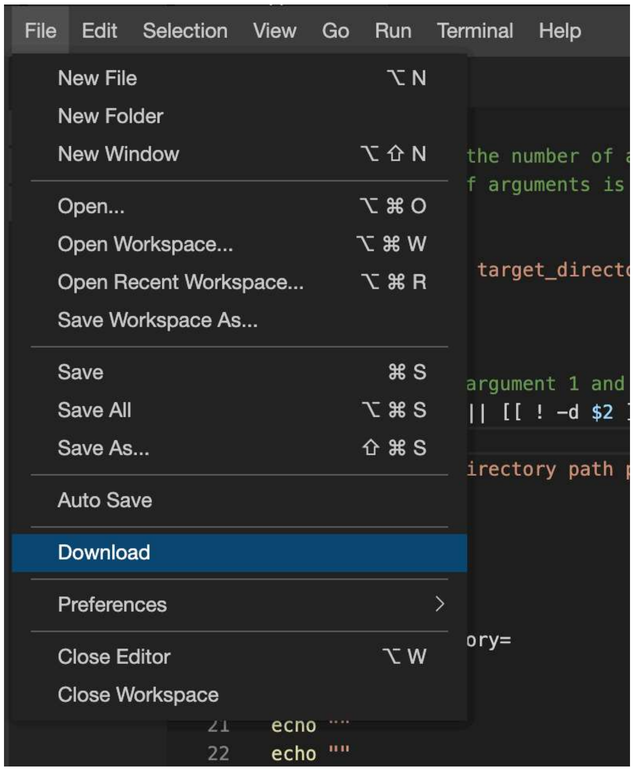
Assessment: >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that creates the compressed backup file using the `toBackup` array and save it as `12-Create_Backup.jpeg` or `12-Create_Backup.png`. ## Task 13 Now the file `$backupFileName` is created in the current working directory.

▶ Click here for Hint

Assessment: >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the code that moves the backup file to the destination directory and save it as `13-Move_Backup.jpeg` or `13-Move_Backup.png`. ------------ Congratulations! You have now done the coding portion of the lab! ## Task 14 - Save the current working file `backup.sh` with `CTRL`+`s` [Windows/Linux], `CMD`+`s` [MAC] or by



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

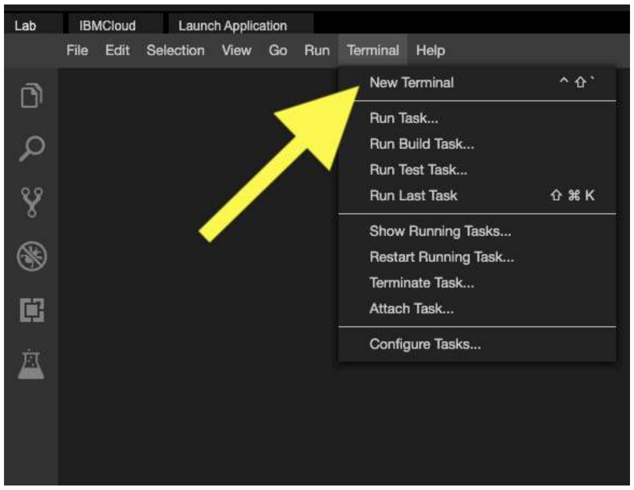


Assessment: >**For Option 2: Peer-Graded Submission and Evaluation**: - Save your completed `backup.sh` file.

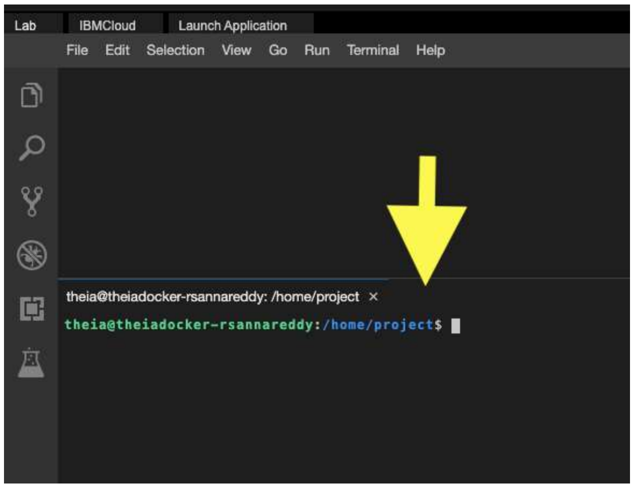
- You will later submit this file for peer grading. ::page{title=""}

Task 15

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:



```
theia@theiadocker-rsannareddy:/home/project$
```

2. Save the `backup.sh` file you\'re working on and make it executable.

▶ Click here for Hint

3. Verify the file is executable using the `ls` command with the `-l` option: ```sh ls -l backup.sh ``` **Assessment:** >**For Option 1: AI-Graded Submission and Evaluation**: - Copy and save the terminal output of the `ls -l backup.sh` command saved in the file named `backup-permissions`, which shows that the `backup.sh` file

Graded Submission and Evaluation**: - Take a screenshot of the output of the `ls -l backup.sh` command showing that the file is executable and save it as `15-executable.jpeg` or `15-executable.png`. ## Task 16 1. Download the following `.zip` file with the `wget` command: ```sh wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-LX0117EN-SkillsNetwork/labs/Final%20Project/important-documents.zip ``` 2. Unzip the archive file: ```sh unzip -DDo important-documents.zip ``` > **Note**: `-DDo` overwrites without restoring original modified date. 3. Update the file's last-modified date to **now**: ``` touch important-documents/* ``` 4. Test your script using the following command: ```sh ./backup.sh important-documents . ``` > This should have created a file called `backup-[CURRENT_TIMESTAMP].tar.gz` in your current directory. **Assessment:** >**For Option 1: AI-Graded Submission and Evaluation**: - Copy and save the terminal output of the `ls -l` command saved in the file named `backup-file-check`, which shows that the backup script ran successfully and that the backup file with the correct name is present in the current directory. >**For Option 2: Peer-Graded Submission and Evaluation**: - Take a screenshot of the output of the `ls -l` command showing the created backup file and save it as `16-backup-complete.jpeg` or `16-backup-complete.png`. ## Task 17 1. **Copy** the `backup.sh` script into the `/usr/local/bin/` directory. (Do ***not*** use `mv`.) > **Note**: You may need to use `sudo cp` in order to create a file in `/usr/local/bin/`. **Assessment:** >**For Option 1: AI-Graded Submission and Evaluation**: - Run the command `sudo cp backup.sh /usr/local/bin/` to copy the script, then run `ls -l /usr/local/bin/backup.sh` to verify it. Copy and paste the terminal output saved in the file named `backup-script-copy`, which confirms that the `backup.sh` script exists in the `/usr/local/bin/` directory. 2. Test the cronjob to see if the backup script is getting triggered by scheduling it for every 1 minute.

▶ Click here for Hint

3. Please note that since the Theia Lab is a virtual environment, we need to explicitly start the cron service using the below command: ```sh sudo service cron start ``` 4. Once the cron service is started, check in the directory `/home/project` to see if the `.tar` files are being created.

5. If they are, then stop the cron service using the below command, otherwise it will continue to create `` `.tar` `` files every minute:

sh

```sh
sudo service cron stop
```

Run

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` ``.
   Assessment:

   > ***For Option 1: AI-Graded Submission and Evaluation:***

- Copy and paste the terminal output of the `` `crontab -l` `` command saved in the file named `` `crontab-schedule` `` that shows the crontab routine is scheduled to run every 24 hours.

   > ***For Option 2: Peer-Graded Submission and Evaluation:***

- Take a screenshot of the output of `` `crontab -l` `` showing the scheduled backup job and save it as `` `17-crontab.jpeg` `` or `` `17-crontab.png` ``.

   > *Tip: When you are setting up cron jobs in a real-life scenario, ensure the cron service is running, or start the cron service if needed.*

Checklist for submission

Follow the checklist below to verify that your project meets all requirements before submission. **Submit your work through either Option 1: AI-Graded Submission and Evaluation or Option 2: Peer-Graded Submission and Evaluation, depending on the submission path you choose for project evaluation. Checklist for Option 1: AI-Graded Submission and Evaluation** After completing the final project, you should have saved:

- The complete code of the backup.sh script, with the correct implementation for each of the 13 tasks.
- The terminal output of the file named **backup-permissions**, which shows that the **backup.sh** file has executable permissions.
- The terminal output of the file named **backup-file-check**, which shows the **backup.sh** file in the current directory with the correct name.
- The terminal output of the file named **backup-script-copy**, which shows that the **backup.sh** script has been copied to the **/usr/local/bin/** directory.
- The terminal output of the file named **crontab-schedule**, which shows that the crontab routine is scheduled to run every 24 hours. **Checklist for Option 2: Peer Review Evaluation**

After completing the final project, you should have:

- The screenshots of sections from the **backup.sh** script displaying the correct code for each 13 tasks.
- The completed **backup.sh** file.
- The screenshot of the output of the **ls -l backup.sh** command showing that the file is executable and named it as **15-executable.jpeg or png**.
- The screenshot of the output of the **ls -l** command showing the created backup file

- The screenshot of the output of **crontab -l** showing the scheduled backup job and named it as **17-crontab.jpeg or png**.

Congratulations!

You have completed the final lab for this course! Well done!

Authors

Md Haroon Hussain Rajashree Patil

Other Contributors

Rav Ahuja

Getting started

← Linux Commands and Sh…

Note: →