# AWS EC2 shutdown script

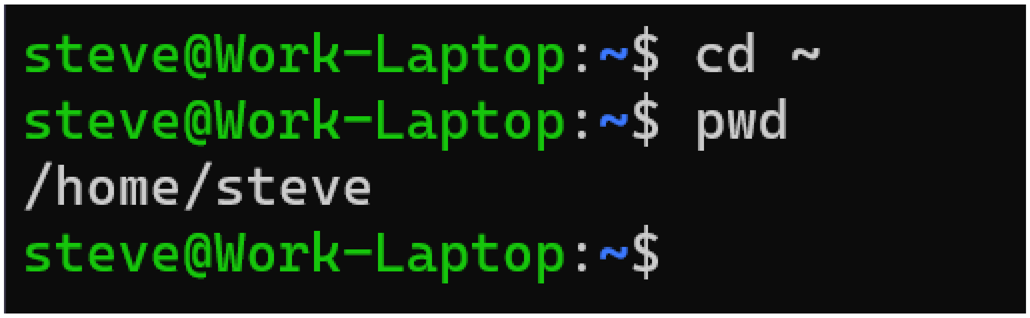
## ****REQUIREMENTS****

**You should have already completed the Lab titled “Configure your Workstation with AWS” before attempting these instructions!**
https://docs.google.com/document/d/1vRg1G-VLy6eEfd7h3C6hnPVoeJUxBn8MdB264a-R2Nw/view
**This script requires the AWS CLI to be installed and configured on your terminal to function properly!**

## Step 1. Navigate to your home directory

1. Open your terminal application (Ubuntu / WSL on Windows or Terminal on MacOS)
2. Navigate to your home directory. You can do this with the following commands:
   (the second command confirms which directory you are currently viewing on your terminal)

- ☐ **cd ~**
- ☐ **pwd**

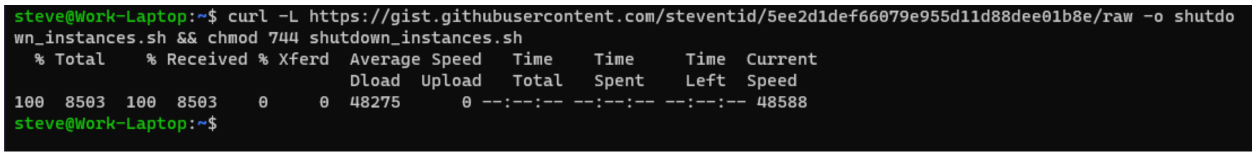


```
steve@Work-Laptop:~$ cd ~
steve@Work-Laptop:~$ pwd
/home/steve
steve@Work-Laptop:~$
```

## Step 2. Download the script to your machine

1. Run the following commands in your terminal to download the script and make it executable **(you should be able to copy/paste the commands!)**

- ☐ **curl -L tinyurl.com/3uwkxnnb -o shutdown_instances.sh**
- ☐ **chmod 744 shutdown_instances.sh**



```
steve@Work-Laptop:~$ curl -L https://gist.githubusercontent.com/steventid/5ee2d1def66079e955d11d88dee01b8e/raw -o shutdo
wn_instances.sh && chmod 744 shutdown_instances.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  8503  100  8503    0     0  48275      0 --:--:-- --:--:-- --:--:-- 48588
steve@Work-Laptop:~$
```

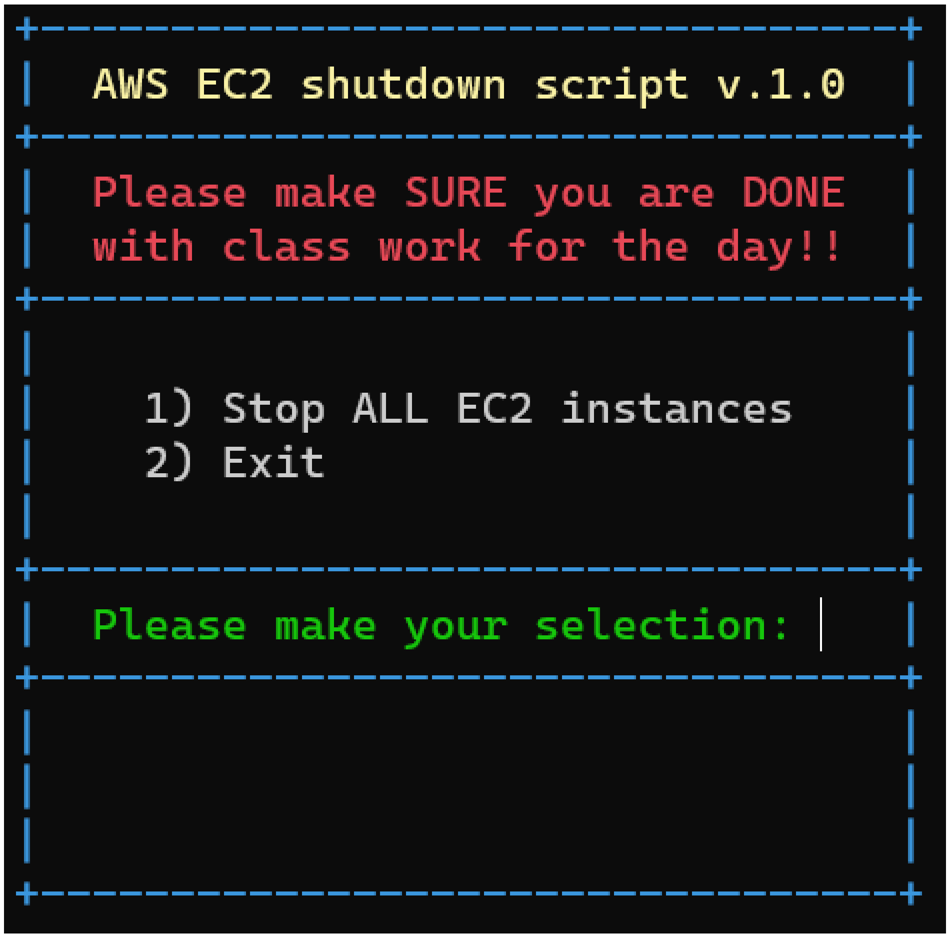
****NOTE** You only need to download the script once. After you do this the first time, you will just need to run the command from Step 3 to execute the script on your machine!**

## Step 3. Running the script

1. Run the following command in your terminal to run the script

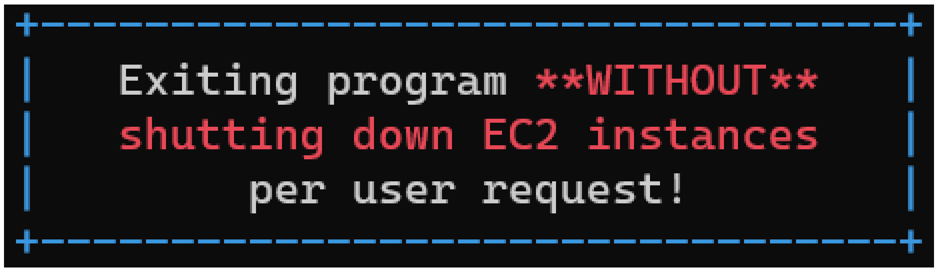
- [ ] **./shutdown_instances.sh**

Once you run the script, your terminal should clear and you will see the main menu.
(I included this so you don't accidentally shut down your instances before you're done for the day)



```
+----------------------------------+
|  AWS EC2 shutdown script v.1.0   |
+----------------------------------+
|  Please make SURE you are DONE   |
|  with class work for the day!!   |
+----------------------------------+
|                                  |
|    1) Stop ALL EC2 instances     |
|    2) Exit                       |
|                                  |
+----------------------------------+
|  Please make your selection:     |
+----------------------------------+
|                                  |
|                                  |
|                                  |
+----------------------------------+
```

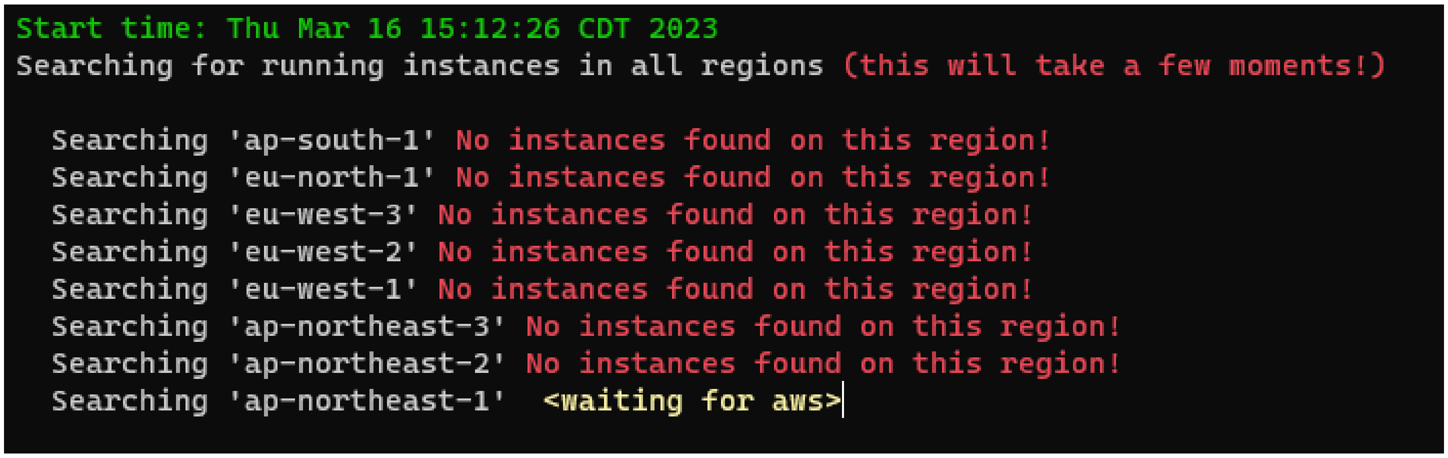
If you run the script by mistake, you can select the **Exit** option and you'll see confirmation in the bottom portion of the menu, then the script will exit.



```
+----------------------------------+
|    Exiting program **WITHOUT**   |
|    shutting down EC2 instances   |
|         per user request!        |
+----------------------------------+
```

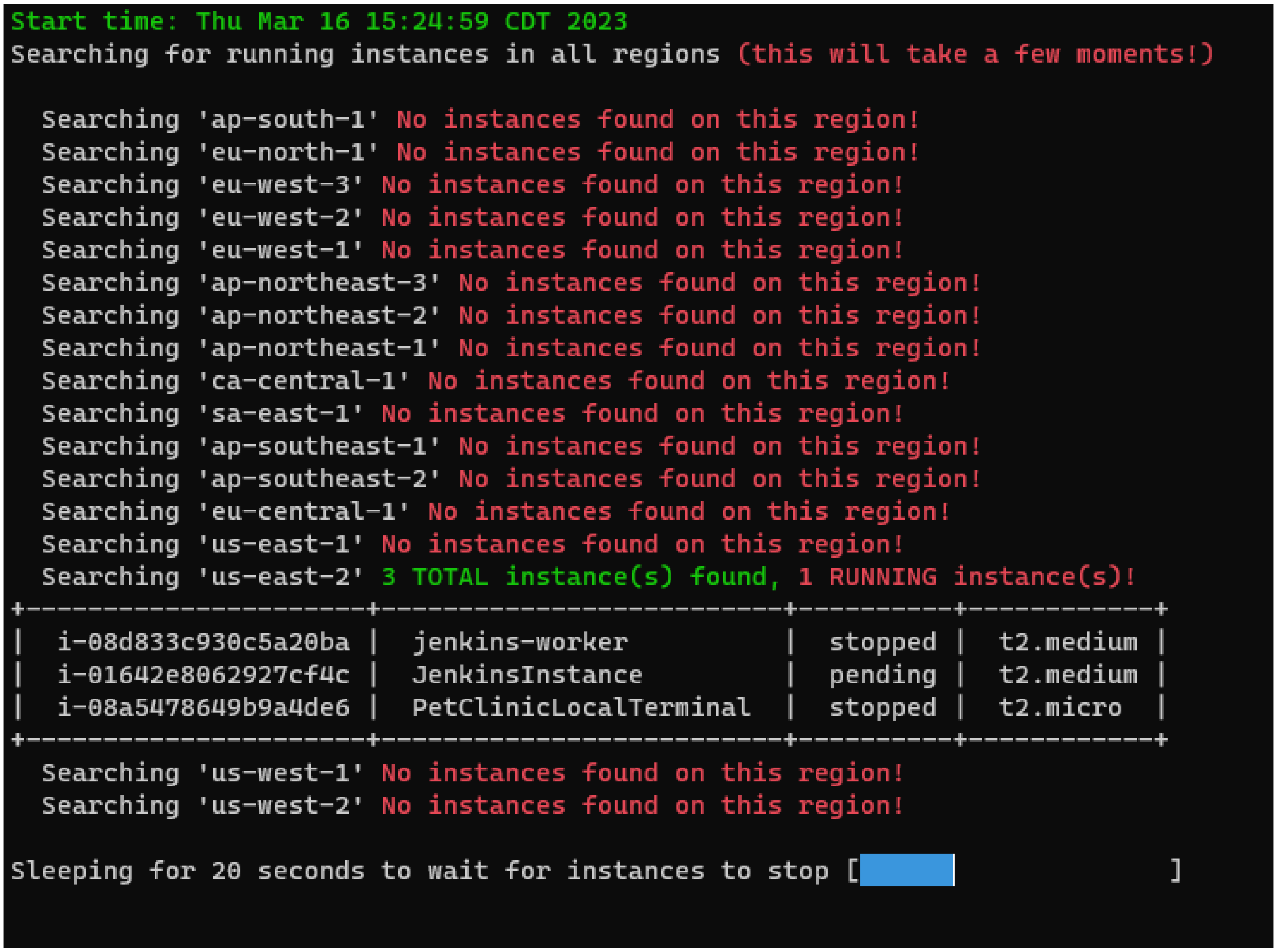
****When you exit this way, your EC2 instances are STILL RUNNING and have NOT been shut down!****

Once you run **Stop all EC2 instances**, it will take a few minutes to complete. This is because the script is requesting a list of regions from AWS and then checking them all for any running instances you may have on that region. Sometimes the AWS servers take a moment to respond, and the script must wait for a response from each region. Your output should look similar to the following (region names may differ):



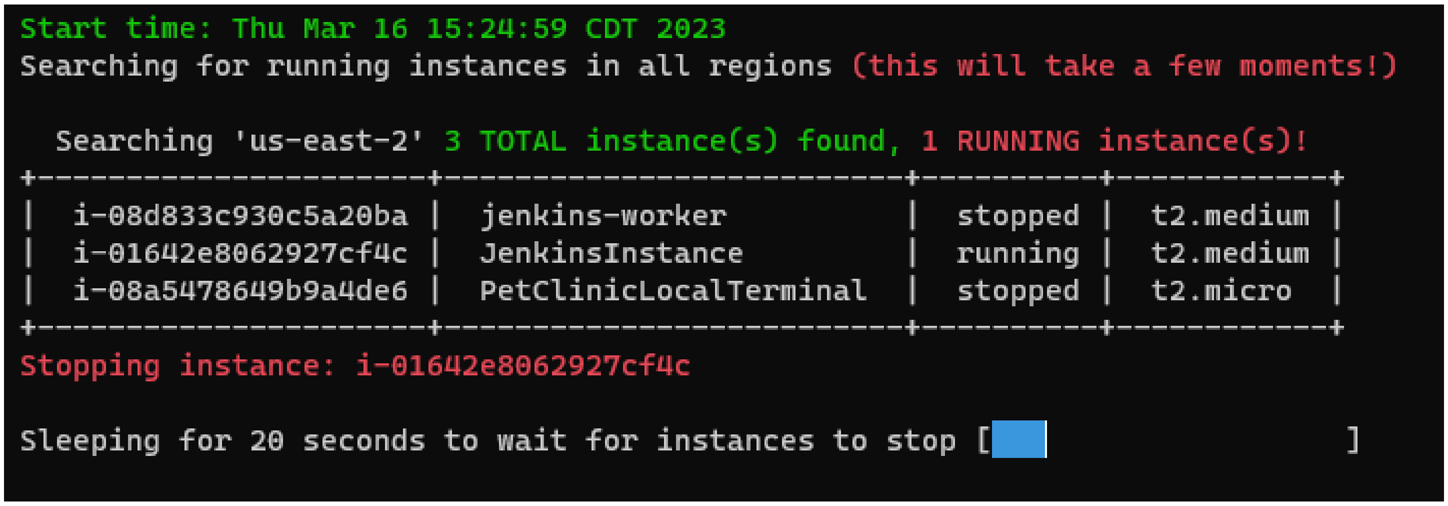
```
Start time: Thu Mar 16 15:12:26 CDT 2023
Searching for running instances in all regions (this will take a few moments!)

  Searching 'ap-south-1' No instances found on this region!
  Searching 'eu-north-1' No instances found on this region!
  Searching 'eu-west-3' No instances found on this region!
  Searching 'eu-west-2' No instances found on this region!
  Searching 'eu-west-1' No instances found on this region!
  Searching 'ap-northeast-3' No instances found on this region!
  Searching 'ap-northeast-2' No instances found on this region!
  Searching 'ap-northeast-1'  <waiting for aws>
```

If the script detects any instances which have a status other than **stopped**, it will wait for 20 seconds and then repeat the scan of regions.On any passes after the first one, the script will **only** scan regions in which you have instances, which results in faster performance on any passes after the first one.



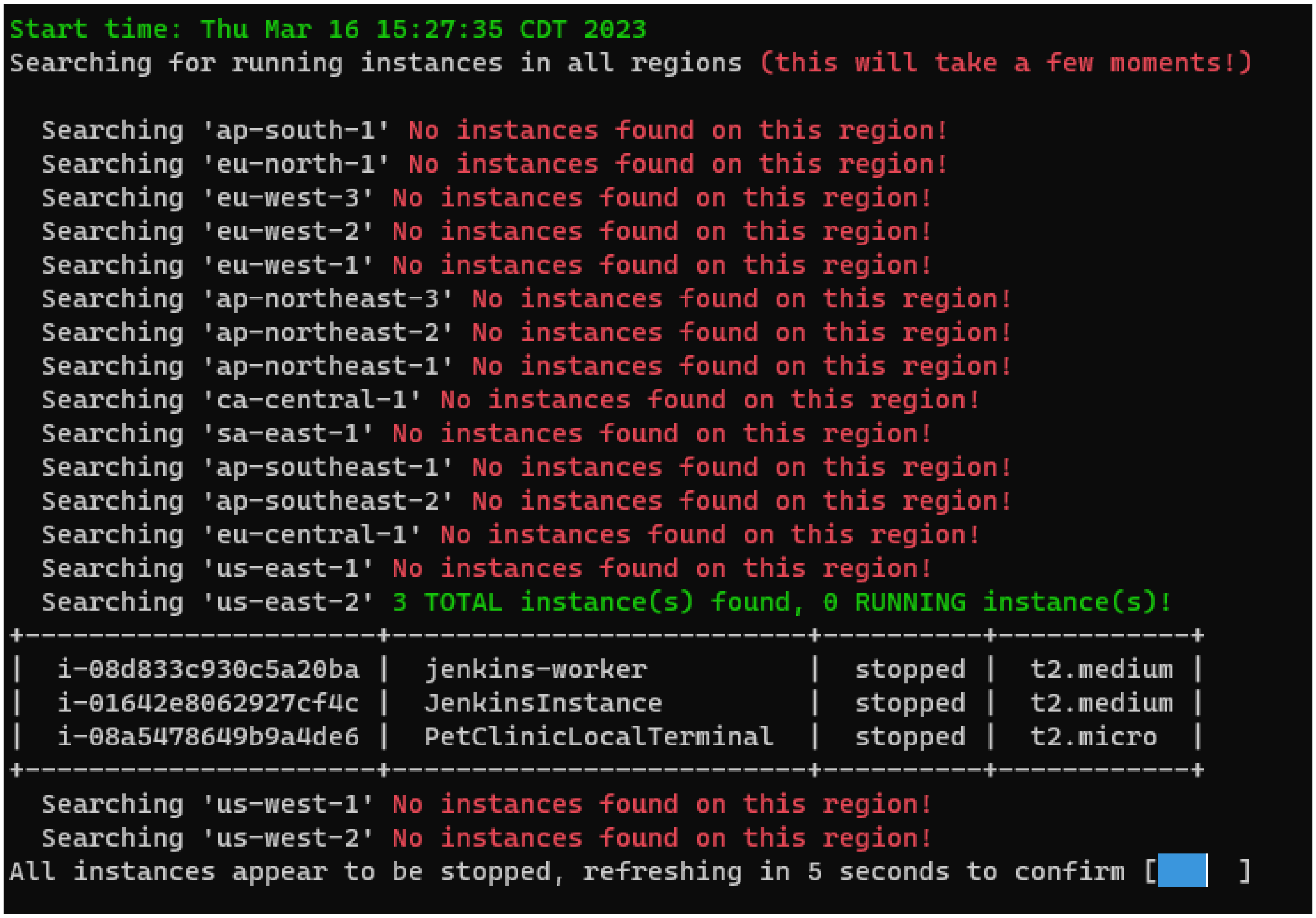
```
Start time: Thu Mar 16 15:24:59 CDT 2023
Searching for running instances in all regions (this will take a few moments!)

  Searching 'ap-south-1' No instances found on this region!
  Searching 'eu-north-1' No instances found on this region!
  Searching 'eu-west-3' No instances found on this region!
  Searching 'eu-west-2' No instances found on this region!
  Searching 'eu-west-1' No instances found on this region!
  Searching 'ap-northeast-3' No instances found on this region!
  Searching 'ap-northeast-2' No instances found on this region!
  Searching 'ap-northeast-1' No instances found on this region!
  Searching 'ca-central-1' No instances found on this region!
  Searching 'sa-east-1' No instances found on this region!
  Searching 'ap-southeast-1' No instances found on this region!
  Searching 'ap-southeast-2' No instances found on this region!
  Searching 'eu-central-1' No instances found on this region!
  Searching 'us-east-1' No instances found on this region!
  Searching 'us-east-2' 3 TOTAL instance(s) found, 1 RUNNING instance(s)!
+----------------------+--------------------------+----------+------------+
|  i-08d833c930c5a20ba |  jenkins-worker          |  stopped |  t2.medium |
|  i-01642e8062927cf4c |  JenkinsInstance         |  pending |  t2.medium |
|  i-08a5478649b9a4de6 |  PetClinicLocalTerminal  |  stopped |  t2.micro  |
+----------------------+--------------------------+----------+------------+
  Searching 'us-west-1' No instances found on this region!
  Searching 'us-west-2' No instances found on this region!

Sleeping for 20 seconds to wait for instances to stop [                  ]
```

If any of the instances are still in the **running** state on the current pass, the script will issue the **stop** command to that instance and display a message to confirm which instance(s) were stopped on this pass:



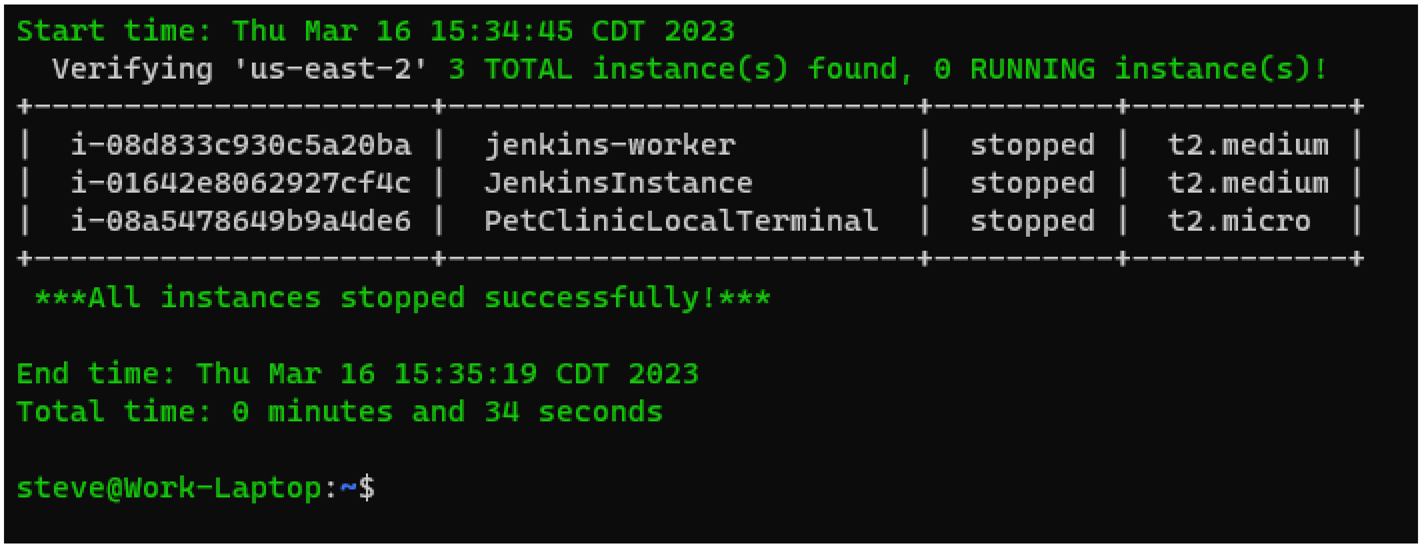
```
Start time: Thu Mar 16 15:24:59 CDT 2023
Searching for running instances in all regions (this will take a few moments!)

  Searching 'us-east-2' 3 TOTAL instance(s) found, 1 RUNNING instance(s)!
+----------------------+--------------------------+----------+------------+
|  i-08d833c930c5a20ba |  jenkins-worker          |  stopped |  t2.medium |
|  i-01642e8062927cf4c |  JenkinsInstance         |  running |  t2.medium |
|  i-08a5478649b9a4de6 |  PetClinicLocalTerminal  |  stopped |  t2.micro  |
+----------------------+--------------------------+----------+------------+
Stopping instance: i-01642e8062927cf4c

Sleeping for 20 seconds to wait for instances to stop [███                ]
```

Once no other instances are in the **running** state, the script will wait for 5 seconds and then refresh one final time to confirm that all instances are now **stopped** on all regions.



```
Start time: Thu Mar 16 15:27:35 CDT 2023
Searching for running instances in all regions (this will take a few moments!)

  Searching 'ap-south-1' No instances found on this region!
  Searching 'eu-north-1' No instances found on this region!
  Searching 'eu-west-3' No instances found on this region!
  Searching 'eu-west-2' No instances found on this region!
  Searching 'eu-west-1' No instances found on this region!
  Searching 'ap-northeast-3' No instances found on this region!
  Searching 'ap-northeast-2' No instances found on this region!
  Searching 'ap-northeast-1' No instances found on this region!
  Searching 'ca-central-1' No instances found on this region!
  Searching 'sa-east-1' No instances found on this region!
  Searching 'ap-southeast-1' No instances found on this region!
  Searching 'ap-southeast-2' No instances found on this region!
  Searching 'eu-central-1' No instances found on this region!
  Searching 'us-east-1' No instances found on this region!
  Searching 'us-east-2' 3 TOTAL instance(s) found, 0 RUNNING instance(s)!
+----------------------+--------------------------+----------+------------+
|  i-08d833c930c5a20ba |  jenkins-worker          |  stopped |  t2.medium |
|  i-01642e8062927cf4c |  JenkinsInstance         |  stopped |  t2.medium |
|  i-08a5478649b9a4de6 |  PetClinicLocalTerminal  |  stopped |  t2.micro  |
+----------------------+--------------------------+----------+------------+
  Searching 'us-west-1' No instances found on this region!
  Searching 'us-west-2' No instances found on this region!
All instances appear to be stopped, refreshing in 5 seconds to confirm [███  ]
```

When **all** of your instances have been stopped, you should see an output similar to the following. This shows a list of all of your instances (now in **stopped** status) as well as the message:
*****All instances stopped successfully!*****



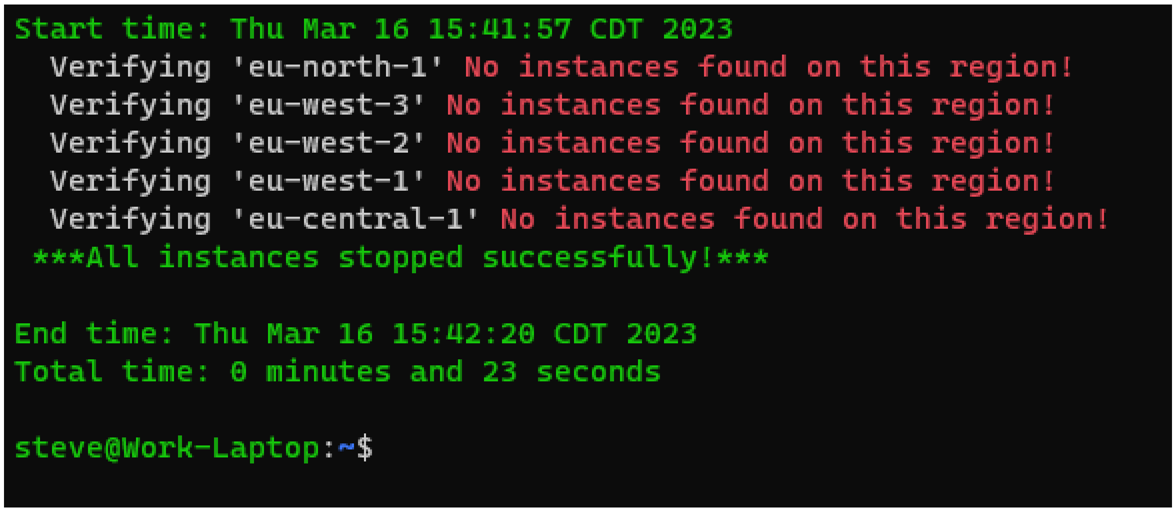
```
Start time: Thu Mar 16 15:34:45 CDT 2023
  Verifying 'us-east-2' 3 TOTAL instance(s) found, 0 RUNNING instance(s)!
+----------------------+--------------------------+----------+------------+
|  i-08d833c930c5a20ba |  jenkins-worker          |  stopped |  t2.medium |
|  i-01642e8062927cf4c |  JenkinsInstance         |  stopped |  t2.medium |
|  i-08a5478649b9a4de6 |  PetClinicLocalTerminal  |  stopped |  t2.micro  |
+----------------------+--------------------------+----------+------------+
 ***All instances stopped successfully!***

End time: Thu Mar 16 15:35:19 CDT 2023
Total time: 0 minutes and 34 seconds

steve@Work-Laptop:~$
```

**When you see an output similar to the above, you should now take a screenshot of the terminal and send it to your TA to sign off on your EC2 instances being shut down for the day.**

## Troubleshooting

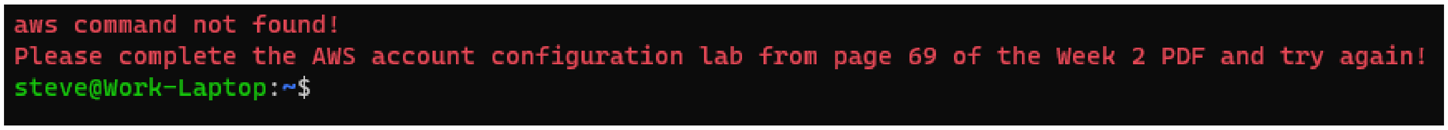
If your AWS CLI is unable to locate **any** instances on the regions available to you, then your output may look similar to the one below. (This should only happen in rare cases since the AWS CLI is accessing **your** specific AWS account and **should** be able to query which regions are available to you, but it may still occur!)

**IF YOUR OUTPUT DOES NOT SHOW ANY INSTANCES, YOU WILL STILL NEED TO SHOW YOUR TA THE AWS EC2 DASHBOARD TO CONFIRM THAT YOUR INSTANCES WERE STOPPED!**



```
Start time: Thu Mar 16 15:41:57 CDT 2023
  Verifying 'eu-north-1' No instances found on this region!
  Verifying 'eu-west-3' No instances found on this region!
  Verifying 'eu-west-2' No instances found on this region!
  Verifying 'eu-west-1' No instances found on this region!
  Verifying 'eu-central-1' No instances found on this region!
 ***All instances stopped successfully!***

End time: Thu Mar 16 15:42:20 CDT 2023
Total time: 0 minutes and 23 seconds

steve@Work-Laptop:~$
```

If you receive the following error message, you should first **complete the lab** to configure AWS CLI on your local terminal (**the lab is linked at the top of this document for reference**)



```
aws command not found!
Please complete the AWS account configuration lab from page 69 of the Week 2 PDF and try again!
steve@Work-Laptop:~$
```

## If you encounter ANY other errors or issues, please reach out to your TA or Steve (since he wrote the script) 😅

****Feel free to skip the following section unless you're curious as to how the script works under the hood!****

## Technical Explanation of the script

For anyone interested in how the script works, there are quite a few comments scattered throughout the code as well. Below is a high-level overview of what's going on, as well as a breakdown of some of the bash commands and what they do.

First, the script will run a brief test to see if the user has installed the AWS CLI as instructed in the prior lab by attempting to run the command:

**aws --version | grep -iFo 'aws-cli'**

This will run the AWS version check command and then pass the output through a pipe into grep with flags -iFo. **F** means to treat the search parameter as a string instead of a regex expression, **i** is to make the search case-insensitive, and the **o** flag will output only the match found. If the user has successfully installed the AWS CLI program, the output should contain the text 'aws-cli' followed by a version number. If, however, the command was not found, the variable AWSCHECK, where the result of the command was stored, will be empty, which is tested by the conditional:

**if [ -z $AWSCHECK ];**

This indicates that the command was not able to run, and thus that the user does not have AWS CLI available on the machine. If this is the case, the script will output an error message and exit, because AWS CLI is required to run the remaining steps

The script will then display the frame for the main menu, then use the tput command to position the cursor, set text colors, and write menu text in the correct place.

Next, the script enters a while loop to get user input via the read command:

**read -n1 -t1 REPLY**

This command uses the -n1 flag to only read the first character from the input and the -t1 flag to timeout after 1 second, storing the input into the REPLY variable. The ERRMSG variable is also cleared here after the 1 second timeout so that it is only displayed briefly after an incorrect command is entered.

Then, the script enters a case statement to compare the contents of the variable REPLY to the 2 menu options, and by default (denoted by the * in the case statement) if the user enters anything other than 1 or 2, the script will display an error message and then redraw the menu and wait for additional input.

After initializing some additional variables, the script will iterate over a list of known AWS regions until one replies with a list of all regions available to the current AWS account, then break out of the loop. Initially, the script will search **all** the regions returned by running the command below and saving the output to the REGION_LIST variable. This list will initially contain any region in which the user may have ec2 instances.

```
aws ec2 –describe-regions –output text –region $REGION
```

This command calls the AWS CLI program to get a list of regions available to the current account in text format, specifically requesting them from the region currently in the $REGION variable, which is filled in by iterating over the $KNOWN_REGIONS array (this parameter is required because if the user did **not** set a default region in the AWS CLI configuration step, then AWS CLI will throw an error and exit instead of letting the script perform additional commands)

When going through the loop, if the script is able to find any instances on this region, they are added to the REGIONS array (if not already present) with this command

```
if [[ ! "${REGIONS[*]}" =~ "$REGION" ]]; then
  REGIONS+=($REGION)
fi
```

The syntax may look a little strange, but this is how to test in bash if something is already present in an array, and if not, then add it to the array. Here is a detailed breakdown:

1. **[[** and **]]** are bash's conditional expressions that allow for more complex conditions than the single **[** and **]** brackets.
2. **!** is a logical operator that negates the expression that follows it.
3. **"${REGIONS[*]}"** expands the array REGIONS into a single string with all elements separated by the first character of the IFS variable (usually a space). (IFS is a bash shell built-in variable)
4. **=~** is a regular expression match operator that checks if the left operand matches the pattern on the right operand.
5. **"$REGION"** is the variable that holds the value being searched for in the REGIONS array.

So, if the value of $REGION is not found in the REGIONS array, the commands inside the if statement will be executed. In this case, the script will add the region currently being searched to the array of regions, but only if the script found any ec2 instances on this region **and** the region is not already in the list of known regions.

On later passes, when the REGIONS array contains values, these are known to be the only regions on which the user has any ec2 instances. At this point, the script copies this data over to the REGIONS_LIST variablel so that the script will run faster after the initial pass.

Up next, the script will enter a for loop to iterate over all the regions with the command below, saving the output to the FOUND variable

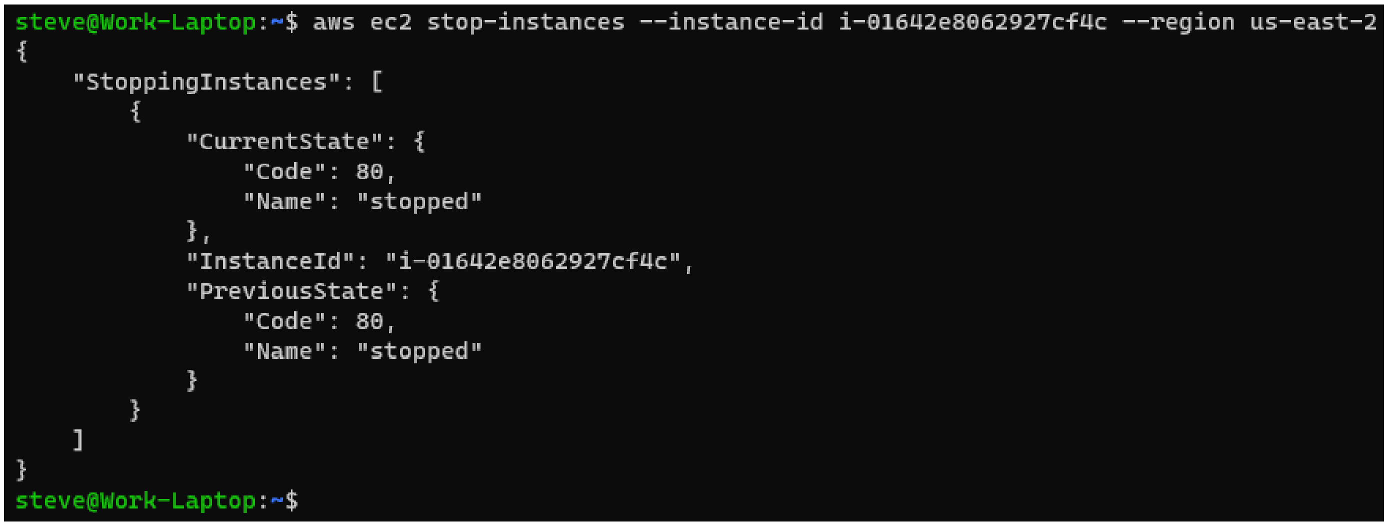
```
aws ec2 describe-instances --query
"Reservations[*].Instances[*].{ID:InstanceId,Type:InstanceType,State:State.Name,Name:Tags[0].Value}"
--output=text --region $REGION | grep -iFv 'terminated'
```

This command calls the AWS CLI program to get a list of instances and specify some query parameters so it will return the **instance ID** (the long number beginning with i-), **instance type** (t2.micro, t2.medium, etc), **state** (running, stopped, pending, terminated, etc), and the **name** (ie: JenkinsInstance). The parameter –output=text sets the output to plain text (other options include table and JSON to name a few). The –region parameter passes the REGION variable from the for loop to query for any instances which are running on this specific region. The output is then piped through grep with the flags -iFv 'terminated'. The **F** means to treat the parameter as a string instead of a regex expression, **i** makes the search case-insensitive, and **v** will invert the match. When it's all put together, this will **exclude** any instances which were terminated from the list, so that the script doesn't list them. The instances may still be listed if they were terminated recently on your AWS dashboard, but if they're listed as terminated, the script should not try to stop them or wait for them to stop since they technically have been deleted.

If the total number of running instances is greater than 0, the script will loop over all instances on the current region to see if any of them are currently in the **running** state. This is because ec2 instances may be **pending** if they were recently started, **stopping** if they're in the process of shutting down, or several other intermediate states besides just **stopped** or **running**. If an instance is currently in the **running** state, the script will then send the stop command to the instance:

**aws ec2 stop-instances --instance-id $INSTANCE --region $REGION > /dev/null 2>&1**

This command calls the AWS CLI program and sends the stop-instances command with the –instance-id flag being passed with the INSTANCE variable and the –region flag with theREGION variable. This will tell the instance to stop on the specific region on which it is located. The command also makes use of the output redirection operator **>** to send the output from both stdout and stderr to /dev/null. This is so that while the script is running, the output on the terminal is not suppressed, instead of displaying something similar to the following:



```
steve@Work-Laptop:~$ aws ec2 stop-instances --instance-id i-01642e8062927cf4c --region us-east-2
{
    "StoppingInstances": [
        {
            "CurrentState": {
                "Code": 80,
                "Name": "stopped"
            },
            "InstanceId": "i-01642e8062927cf4c",
            "PreviousState": {
                "Code": 80,
                "Name": "stopped"
            }
        }
    ]
}
steve@Work-Laptop:~$
```

Once the script has searched for instances in all of the regions, it will test to see if there were any instances which were not in the **stopped** state on the current pass. If this is the case, the script will then sleep for 20 seconds (mostly to allow the servers on AWS to process the shutdown commands) and then start checking the instances again to confirm that they were stopped.

When there are no instances found to be **running**, the script will then set the ACTION variable from 'Searching' to 'Verifying' to instruct the script that this will be the final pass to verify that all known instances are currently in the **stopped** state. If the final pass shows all servers in the **stopped** state, then the script will display the verification screen, which includes the script start time, end time, and the total time elapsed for documentation purposes when the screenshots are sent to your TA.

## Further Reading

**If you would like more information on bash scripting, the manual is located here:**
**[https://www.gnu.org/software/bash/manual/bash.html](https://www.gnu.org/software/bash/manual/bash.html)**

**For further reading into the aws cli commands and parameters, the documentation is here:**
**[https://docs.aws.amazon.com/cli/index.html](https://docs.aws.amazon.com/cli/index.html)**