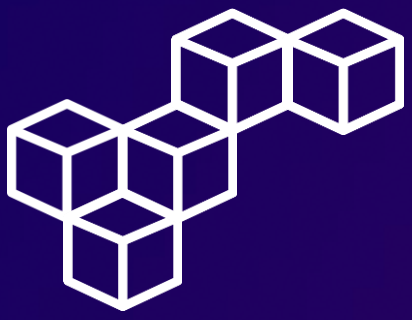
AWS re:Start

CHALLENGE LAB

Advanced Bash Shell Scripting

WEEK 3

aws re/start



Overview

Shell scripting empowers users on Unix-based systems to automate tasks. Scripts can manipulate files and directories, process text efficiently, and even manage system administration tasks. For anyone working with Unix systems, mastering shell scripting unlocks a world of automation, efficiency, and streamlined workflows.

Your Challenge

Write a Bash script based on the following requirements:

- Creates 25 empty (0 KB) files.
- The file names should be <yourName><number>, <yourName><number+1>, <yourName><number+2>, and so on.
- Design the script so that each time you run it, it creates the next batch of 25 files with increasing numbers starting with the last or maximum number that already exists.
- Do not hard code these numbers. You need to generate them by using automation.
- Test the script. Display a long list of the directory and its contents to validate that the script created the expected files.

Note: This lab was made using Windows Subsystem for Linux.

Task 1

Use SSH to connect to an Amazon Linux EC2 instance

Initial Preparations

In the AWS Management Console, select the EC2 instance and make note of the **Public IPv4 address**.

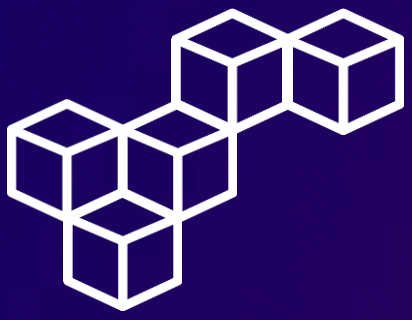
Download the **private key file** labsuser.pem. Change to the Downloads directory and modify the permissions on the key to be read-only (r--------).

Connect to the instance using SSH

Establish a connection to the EC2 instance using the ssh command, the key and the instance's public IPv4 address.

```
support@HP-Pavilion-Laptop:~/Downloads$ ssh -i labsuser.pem ec2-user@54.187.147.58
The authenticity of host '54.187.147.58 (54.187.147.58)' can't be established.
ED25519 key fingerprint is SHA256:tnVRrBWMDQeJclo5Y2AthPQcdk4cU6OttMrQ/ScgMcg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.187.147.58' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2025-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-10-0-10-188 ~]$
```

aws re/start



Task 2

Your Challenge

Adding the shebang line

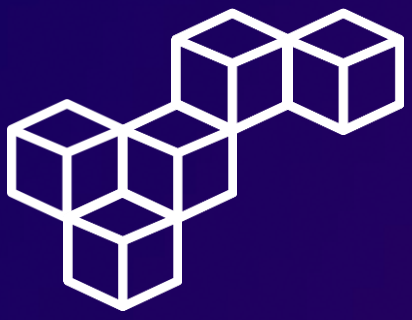
```
#!/bin/bash
```

This line at the beginning of the script is called the shebang line. It tells the system which interpreter to use to execute the script. In this case, #!/bin/bash specifies that the script should be run using the Bash shell.

Assigning the variable name

```
yourName="CristhianBecerra"
```

This line simply assigns the value "CristhianBecerra“, my name, to the variable yourName, which is used later in the script to construct the file names.



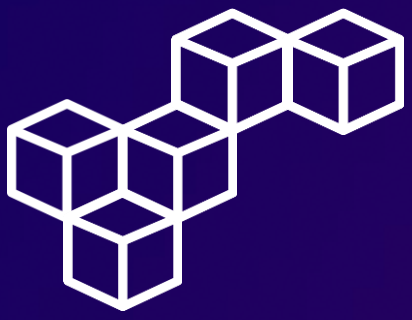
Task 2

Your Challenge

Finding the maximum number in existing files

```
maxNum=$(ls -1 | grep -oP “${yourName}\K\d+" | sort -n | tail -n1)
```

- **ls -1**: Lists the files in the directory one per line.
- **grep -oP "${yourName}\K\d+"**: Uses grep to search for the specified name followed by digits (\d+) using a regular expression (-oP). The -o option displays only the part that matches the regular expression. The P option enables grep to use Perl-compatible regular expressions for advanced pattern matching.
- **\K**: In the regular expression, \K means "forget what has been matched before." In this case, it forgets the name and only shows the following digits (\d+).
- **sort -n**: Sorts the found numbers numerically (-n) instead of alphabetically.
- **tail –n1**: Takes the last line, which will be the highest number found in the existing file names.



Task 2

Your Challenge

Setting the initial number for new files

```
if [ -z “$maxNum" ]; then
  startNum=1
else
  startNum= $((maxNum + 1))
fi
```

The -z condition checks if the string $maxNum is empty. In this context, $maxNum would be empty if no files were found with the name "CristhianBecerra," which means startNum is set to 1. Otherwise, if existing files were found, startNum is set to the highest number found plus one.

Creating 25 empty files with incremental names

```
for i in {0..24}; do
  touch "${yourName}$((startNum + i))"
done
```

This for loop iterates 25 times. In each iteration, touch is used to create a file with the constructed name by concatenating name and the current number (startNum + i).

Task 2

Your Challenge

Typing the script

Type the script using the Vim text editor. Be sure to include comments using # to enhance the readability of your code. After saving your changes and closing the editor, change the file permissions to make it executable.

```
[ec2-user@ip-10-0-10-188 ~]$ vim cristhian_becerra.sh
[ec2-user@ip-10-0-10-188 ~]$ chmod 700 cristhian_becerra.sh
[ec2-user@ip-10-0-10-188 ~]$
```

```
#!/bin/bash

# Set the name
yourName="CristhianBecerra"

# Find the maximum number in the current files
maxNum=$(ls -1 | grep -oP "${yourName}\K\d+" | sort -n | tail -n1)

# Set the starting number for the new files
if [ -z "$maxNum" ]; then
  startNum=1
else
  startNum=$((maxNum + 1))
fi

# Create 25 empty files with increasing numbers
for i in {0..24}; do
  touch "${yourName}$((startNum + i))"
done
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                    19,5          All
```

aws re/start

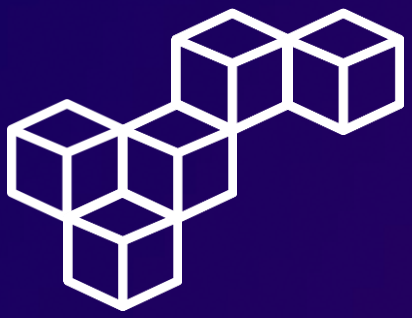
Task 2

Your Challenge

Testing the script

Test the script. Display a long list of the directory and its contents to validate that the script created the expected files.

```
[ec2-user@ip-10-0-10-188 ~]$ ls
companyA  cristhian_becerra.sh
[ec2-user@ip-10-0-10-188 ~]$ ./cristhian_becerra.sh
[ec2-user@ip-10-0-10-188 ~]$ ls
CristhianBecerra1   CristhianBecerra14  CristhianBecerra19  CristhianBecerra23  CristhianBecerra5  companyA
CristhianBecerra10  CristhianBecerra15  CristhianBecerra2   CristhianBecerra24  CristhianBecerra6  cristhian_becerra.sh
CristhianBecerra11  CristhianBecerra16  CristhianBecerra20  CristhianBecerra25  CristhianBecerra7
CristhianBecerra12  CristhianBecerra17  CristhianBecerra21  CristhianBecerra3   CristhianBecerra8
CristhianBecerra13  CristhianBecerra18  CristhianBecerra22  CristhianBecerra4   CristhianBecerra9
[ec2-user@ip-10-0-10-188 ~]$ ./cristhian_becerra.sh
[ec2-user@ip-10-0-10-188 ~]$ ls
CristhianBecerra1   CristhianBecerra18  CristhianBecerra26  CristhianBecerra34  CristhianBecerra42  CristhianBecerra50
CristhianBecerra10  CristhianBecerra19  CristhianBecerra27  CristhianBecerra35  CristhianBecerra43  CristhianBecerra6
CristhianBecerra11  CristhianBecerra2   CristhianBecerra28  CristhianBecerra36  CristhianBecerra44  CristhianBecerra7
CristhianBecerra12  CristhianBecerra20  CristhianBecerra29  CristhianBecerra37  CristhianBecerra45  CristhianBecerra8
CristhianBecerra13  CristhianBecerra21  CristhianBecerra3   CristhianBecerra38  CristhianBecerra46  CristhianBecerra9
CristhianBecerra14  CristhianBecerra22  CristhianBecerra30  CristhianBecerra39  CristhianBecerra47  companyA
CristhianBecerra15  CristhianBecerra23  CristhianBecerra31  CristhianBecerra4   CristhianBecerra48  cristhian_becerra.sh
CristhianBecerra16  CristhianBecerra24  CristhianBecerra32  CristhianBecerra40  CristhianBecerra49
CristhianBecerra17  CristhianBecerra25  CristhianBecerra33  CristhianBecerra41  CristhianBecerra5
[ec2-user@ip-10-0-10-188 ~]$ ./cristhian_becerra.sh
[ec2-user@ip-10-0-10-188 ~]$ ls
CristhianBecerra1   CristhianBecerra21  CristhianBecerra33  CristhianBecerra45  CristhianBecerra57  CristhianBecerra69
CristhianBecerra10  CristhianBecerra22  CristhianBecerra34  CristhianBecerra46  CristhianBecerra58  CristhianBecerra7
CristhianBecerra11  CristhianBecerra23  CristhianBecerra35  CristhianBecerra47  CristhianBecerra59  CristhianBecerra70
CristhianBecerra12  CristhianBecerra24  CristhianBecerra36  CristhianBecerra48  CristhianBecerra6   CristhianBecerra71
CristhianBecerra13  CristhianBecerra25  CristhianBecerra37  CristhianBecerra49  CristhianBecerra60  CristhianBecerra72
CristhianBecerra14  CristhianBecerra26  CristhianBecerra38  CristhianBecerra5   CristhianBecerra61  CristhianBecerra73
CristhianBecerra15  CristhianBecerra27  CristhianBecerra39  CristhianBecerra50  CristhianBecerra62  CristhianBecerra74
CristhianBecerra16  CristhianBecerra28  CristhianBecerra4   CristhianBecerra51  CristhianBecerra63  CristhianBecerra75
CristhianBecerra17  CristhianBecerra29  CristhianBecerra40  CristhianBecerra52  CristhianBecerra64  CristhianBecerra8
CristhianBecerra18  CristhianBecerra3   CristhianBecerra41  CristhianBecerra53  CristhianBecerra65  CristhianBecerra9
CristhianBecerra19  CristhianBecerra30  CristhianBecerra42  CristhianBecerra54  CristhianBecerra66  companyA
CristhianBecerra2   CristhianBecerra31  CristhianBecerra43  CristhianBecerra55  CristhianBecerra67  cristhian_becerra.sh
CristhianBecerra20  CristhianBecerra32  CristhianBecerra44  CristhianBecerra56  CristhianBecerra68
[ec2-user@ip-10-0-10-188 ~]$
```



Task 3

Add something extra!

Validating user input

```
echo –n "Please enter your name. Use only letters and underscores: "
read yourName

if [[ ! $yourName =~ ^[a-zA-Z_]+$ ]]; then
    echo "Usage: The name should only contain letters or underscores."
    exit 1
fi
```

- **echo –n "Please enter your name. Use only letters and underscores:"**: Prompts the user to input a name.
- **read yourName**: Reads user input and stores it in the variable yourName.
- **if [[! $yourName =~ ^[a-zA-Z_]+$]]; then**: Checks if the input contains only letters (uppercase or lowercase) or underscores.
- **echo "Usage: The name should only contain letters or underscores."**: Displays an error message if the input contains invalid characters.
- **exit 1**: Exits the script with an error code of 1 if an invalid input is detected.

Task 3

Add something extra!

Implementing proposed script updates

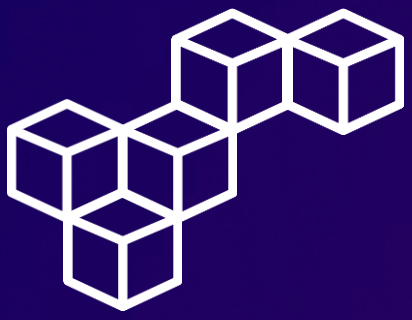
In the Vim text editor, update the script by replacing the existing line yourName="CristhianBecerra" with the proposed modifications to prompt the user for their name and validate the input.

```
#!/bin/bash

# Prompt the user for a name and validate user input
echo -n "Please enter your name. Use only letters and underscores: "
read yourName

if [[ ! $yourName =~ ^[a-zA-Z_]+$ ]]; then
    echo "Usage: The name should only contain letters or underscores."
    exit 1
fi

# Find the maximum number in the current files
-- INSERT --                                                    1,1           Top
```

Testing the input validation feature

Delete any previously generated empty files, then proceed to test the new input validation feature of the updated script.

```
[ec2-user@ip-10-0-10-188 ~]$ ./cristhian_becerra.sh
Please enter your name. Use only letters and underscores: Cristhian Becerra
Usage: The name should only contain letters or underscores.
[ec2-user@ip-10-0-10-188 ~]$ ./cristhian_becerra.sh
Please enter your name. Use only letters and underscores: CristhianBecerra99
Usage: The name should only contain letters or underscores.
[ec2-user@ip-10-0-10-188 ~]$ ./cristhian_becerra.sh
Please enter your name. Use only letters and underscores: Cristhian_Becerra%
Usage: The name should only contain letters or underscores.
[ec2-user@ip-10-0-10-188 ~]$ ./cristhian_becerra.sh
Please enter your name. Use only letters and underscores: Cristhian_Becerra
[ec2-user@ip-10-0-10-188 ~]$ ls
Cristhian_Becerra1   Cristhian_Becerra14  Cristhian_Becerra19  Cristhian_Becerra23  Cristhian_Becerra5  companyA
Cristhian_Becerra10  Cristhian_Becerra15  Cristhian_Becerra2   Cristhian_Becerra24  Cristhian_Becerra6  cristhian_becerra.sh
Cristhian_Becerra11  Cristhian_Becerra16  Cristhian_Becerra20  Cristhian_Becerra25  Cristhian_Becerra7
Cristhian_Becerra12  Cristhian_Becerra17  Cristhian_Becerra21  Cristhian_Becerra3   Cristhian_Becerra8
Cristhian_Becerra13  Cristhian_Becerra18  Cristhian_Becerra22  Cristhian_Becerra4   Cristhian_Becerra9
[ec2-user@ip-10-0-10-188 ~]$
```



Conclusions

Bash scripting
Bash scripting is a versatile tool that automates tasks in Unix systems, enhancing efficiency in system management and development workflows.

Regular expressions
Regular expressions are powerful patterns used for text manipulation, including tasks like pattern matching, data validation, and text extraction.

The if statement
The if-else-fi construct in Bash facilitates conditional execution, enabling scripts to make decisions based on specific conditions and enhancing their decision-making capabilities.

The for loop
The for loop in Bash simplifies repetitive tasks by iterating over values or items, streamlining processes like file handling, command execution, and data processing within scripts.

Testing scripts
Testing scripts is essential to ensure their functionality, reliability, and compatibility across different scenarios and environments, including input validation, error handling, and output verification.

aws re/start

Cristhian Becerra

cristhian-becerra-espinoza

+51 951 634 354

cristhianbecerra99@gmail.com

Lima, Peru