

Linux Logger Documentation

Neustar-SSI-Coding-Assignmet_v1

Ignacio Álvarez Barrantes

The following document provides all the documentation related to the requirements, elaboration and use of the Linux Logger application.

Requirements

- Create 3 Linux VMs, you can create the VMs on the cloud (AWS, Google, Azure), or on your local machine using the virtualization tool of your choice.
- Create an application that lets you input “x” number of server IPs separated by comma (eg.10.10.10.1, 192.168.1.1, etc)
- Connect to each server with the provided Ips
- Perform The following logs:
 - Log the current running processes (one single interval).
 - Log the top 3 application that are consuming more CPU and top 3 that are consuming more Memory resources
 - Log how much capacity is remaining on each running VM
 - *Display that capacity in a meaningful human and machine-readable form*
- Log the output on a file
 - The output must contain the server IP and the information previously requested

Architecture/Design

For the first requirement (Creating 3 Linux VMs) the cloud service provided by AWS was used. Three simple EC2 instances with Ubuntu Server images running as their OS were created and hosted on the AWS Cloud services. As well a key was created for all 3 VMs in order to access them via a Secure Shell connection.

<input type="checkbox"/>	Name ▾	ID de la instancia	Estado de la i... ▾	Tipo de inst... ▾	Comprobación ...
<input type="checkbox"/>	Ubuntu VM1 🔗	i-0155348983f6bdaac	✓ En ejecución 🔍 🔗	t2.micro	✓ 2/2 comprobacion
<input type="checkbox"/>	Ubuntu VM2	i-09ba321fe0f5eb4a1	✓ En ejecución 🔍 🔗	t2.micro	✓ 2/2 comprobacion
<input type="checkbox"/>	Ubuntu VM3	i-082e66080cb56dded	✓ En ejecución 🔍 🔗	t2.micro	✓ 2/2 comprobacion

Each of them with the following IPs and Public DNS:

Ubuntu VM1:

Detalles	Seguridad	Redes	Almacenamiento	Comprobaciones de estado	Monitoreo	Etiquetas
▼ Resumen de instancia Información						
ID de la instancia 🔗 i-0155348983f6bdaac (Ubuntu VM1)		Dirección IPv4 pública 🔗 52.14.2.61 dirección abierta 🔗		Direcciones IPv4 privadas 🔗 172.31.36.68		
Dirección IPv6 -		Estado de la instancia ✓ En ejecución		DNS de IPv4 pública 🔗 ec2-52-14-2-61.us-east-2.compute.amazonaws.com dirección abierta 🔗		

Ubuntu VM2:

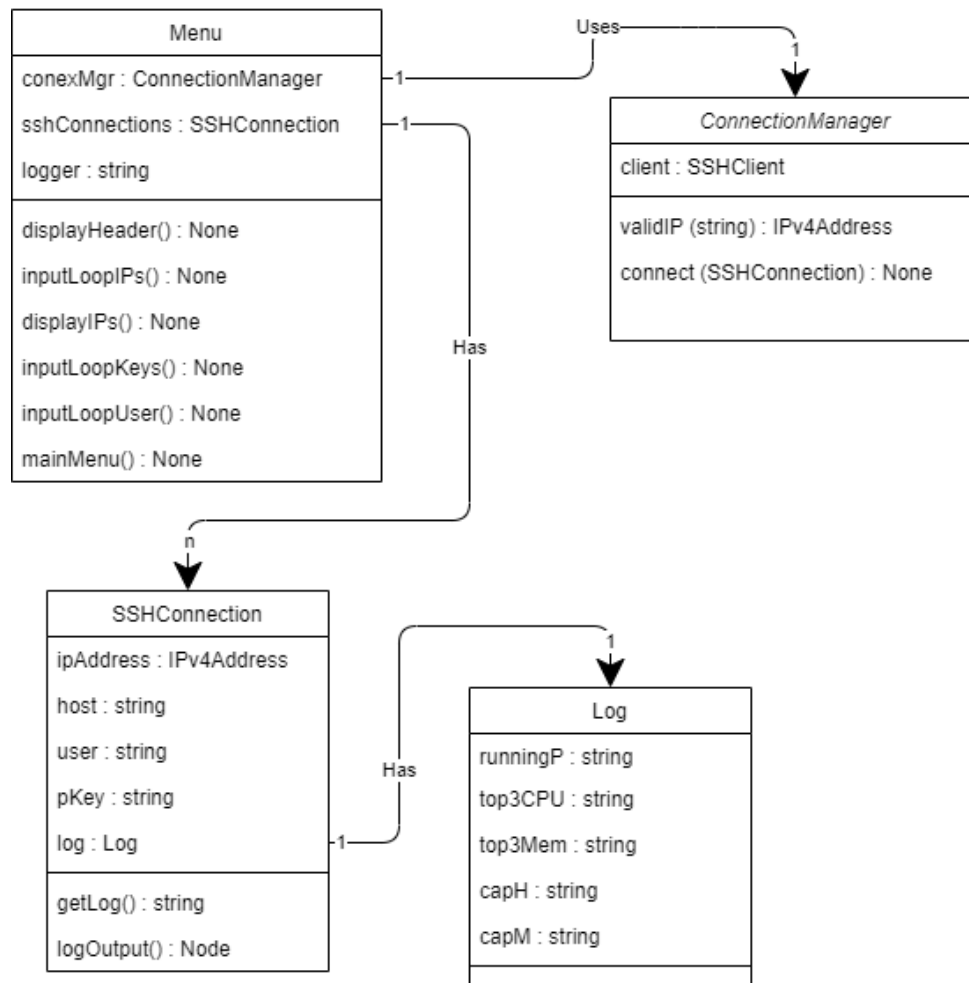
Detalles	Seguridad	Redes	Almacenamiento	Comprobaciones de estado	Monitoreo	Etiquetas
▼ Resumen de instancia Información						
ID de la instancia i-09ba321fe0f5eb4a1 (Ubuntu VM2)		Dirección IPv4 pública 18.118.133.97 dirección abierta		Direcciones IPv4 privadas 172.31.42.200		
Dirección IPv6 -		Estado de la instancia ✓ En ejecución		DNS de IPv4 pública ec2-18-118-133-97.us-east-2.compute.amazonaws.com dirección abierta		

Ubuntu VM3:

Detalles	Seguridad	Redes	Almacenamiento	Comprobaciones de estado	Monitoreo	Etiquetas
▼ Resumen de instancia Información						
ID de la instancia i-082e66080cb56dded (Ubuntu VM3)		Dirección IPv4 pública 18.222.57.51 dirección abierta		Direcciones IPv4 privadas 172.31.45.125		
Dirección IPv6 -		Estado de la instancia ✓ En ejecución		DNS de IPv4 pública ec2-18-222-57-51.us-east-2.compute.amazonaws.com dirección abierta		

For the elaboration of the software solution the selected language was Python due to its multiple existing libraries that facilitate IP address validations, SSH Connections and output to file. As well, an Object-Oriented Programming approach was chosen for the design of the application with the following UML Class Diagram representing the overall structure of the code.

UML



Code

The code may be found in this Github [repository](#). The repository contains all the files needed to compile the program via Python Interpreter and execute it, also, both executables (for Windows and Linux systems) are stored on the dist folder inside the Code Assignment directory. Each function and class contains heading comments for an easier lecture and comprehension of the code.

End-User Manual

This following section describes thoroughly how to compile the application for both systems (Windows and Linux), as well as how to use it to log successfully the VMs.

Windows Compilation

For compiling a Python program in windows first is necessary to have pip as an accessible command on the command line, as well as having the Python script folder included in Windows Path.

Step 1: Open the Windows Command Prompt (preferably as admin)

Step 2: Install Pyinstaller Package via pip with the following command

- `pip install pyinstaller`

Step 3: Move to the Linux Logger program Folder (Where all the Python scripts are found)

Step 4: Create the executable using Pyisntaller with the following command

- `pyinstaller --onefile Menu.py`

Step 5: Run the executable that will be found on the *dist* folder that was generated before the *pyinstaller* command

Linux Compilation

For compiling a Python program in Linux first is necessary to have pip as an accessible command on the terminal

Step 1: Open the Terminal

Step 2: Install Pyinstaller Package via pip with the following command

- `pip install pyinstaller`

Step 3: Move to the Linux Logger program Folder (Where all the Python scripts are found)

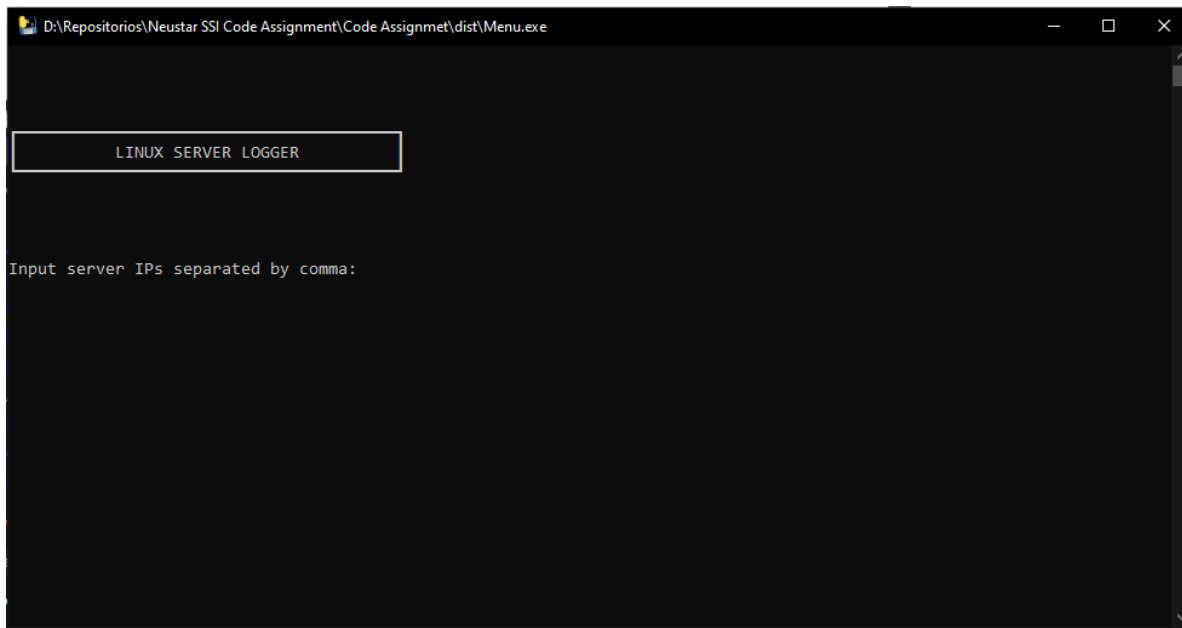
Step 4: Create the executable using Pyisntaller with the following command

- `python3 -m PyInstaller --onefile Menu.py`

Step 5: Run the executable that will be found on the *dist* folder that was generated before the *pyinstaller* command

User Manual

Once the executable is running it will display the menu containing a header and asking the user for the IPs of the VMs that he wants to connect to:



The program receives each IP separated by a comma, then once enter is pressed the program will validate each IP and verify if its valid, stores the ones that are valid and prompt the user with an error in case of invalid IP. When prompted error to the user, he can decide if he wants to retype the invalid IP or continue with the loaded ones.

```
D:\Repositorios\Neustar SSL Code Assignment\Code Assignmet\dist\Menu.exe

LINUX SERVER LOGGER

Input server IPs separated by comma:
52.14.2.61,18.118.133.97,18.222.57.511

IP address 52.14.2.61 is valid. Stored successfully
IP address 18.118.133.97 is valid. Stored successfully
IP address 18.222.57.511 is not valid
Retype the invalid ip addresses (or enter '.' to skip):
18.222.57.51_
```

Then the program will prompt the current loaded IPs for further verification. Immediately after this, the user will be asked to enter the path to a private key to stablish the SSH connection.

```
D:\Repositorios\Neustar SSL Code Assignment\Code Assignmet\dist\Menu.exe

LINUX SERVER LOGGER

Input server IPs separated by comma:
52.14.2.61,18.118.133.97,18.222.57.511

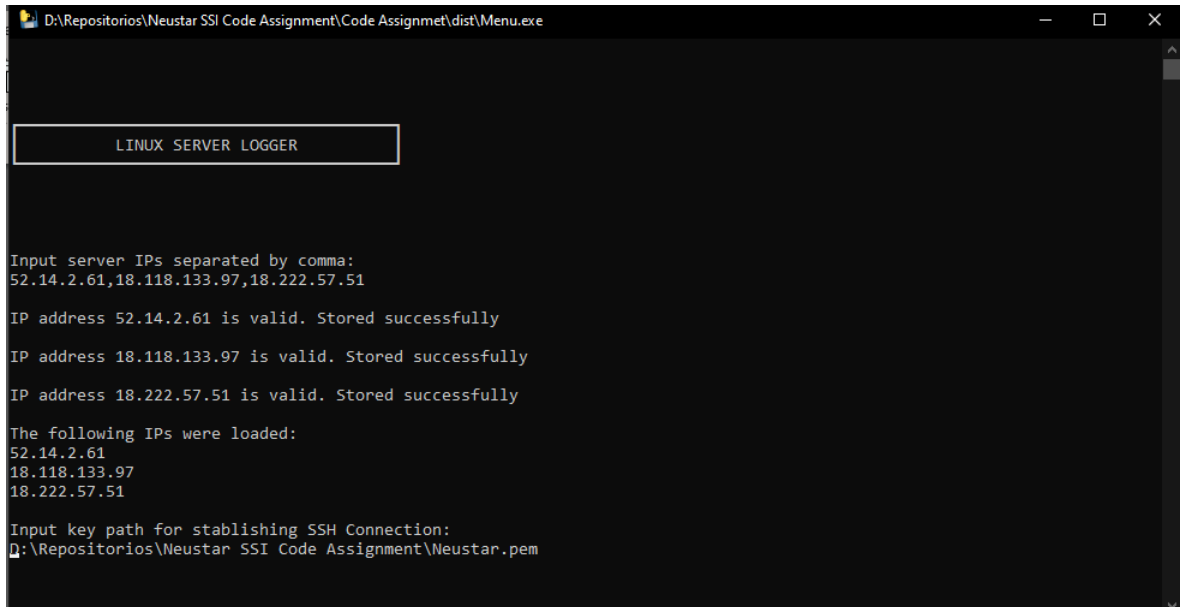
IP address 52.14.2.61 is valid. Stored successfully
IP address 18.118.133.97 is valid. Stored successfully
IP address 18.222.57.511 is not valid
Retype the invalid ip addresses (or enter '.' to skip):
18.222.57.51

IP address 18.222.57.51 is valid. Stored successfully

The following IPs were loaded:
52.14.2.61
18.118.133.97
18.222.57.51

Input key path for stablishing SSH Connection:
```

Important Note: The “.pem” keys provided by AWS when creating the EC2 instances can be used here. In case of other VMs it is required the usage of a private key with the OpenSSH format, otherwise the connection can’t be fulfilled successfully



```
D:\Repositorios\Neustar SSI Code Assignment\Code Assignmet\dist\Menu.exe

LINUX SERVER LOGGER

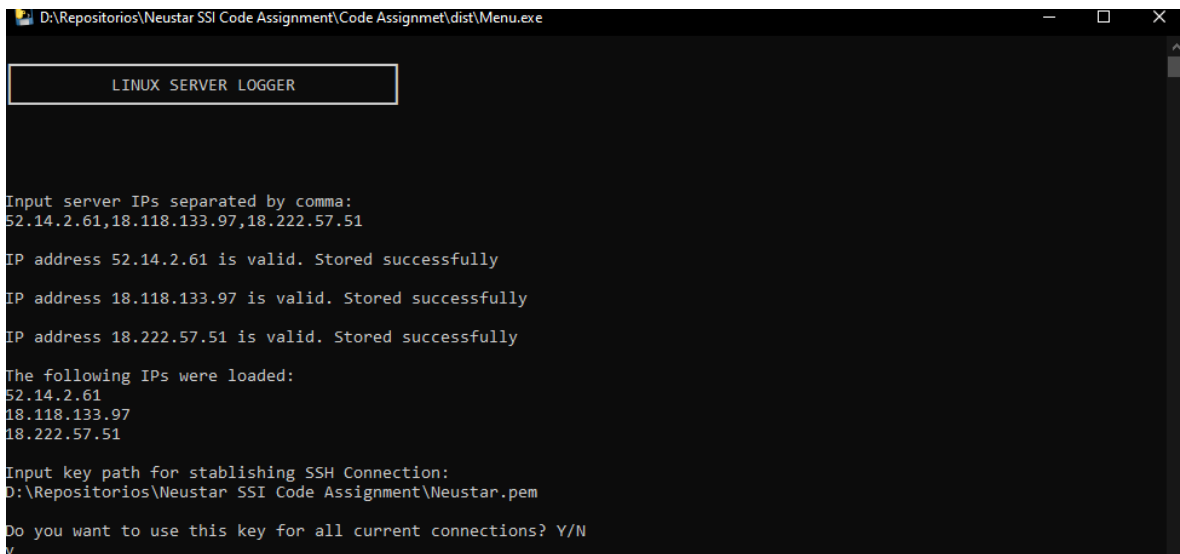
Input server IPs separated by comma:
52.14.2.61,18.118.133.97,18.222.57.51

IP address 52.14.2.61 is valid. Stored successfully
IP address 18.118.133.97 is valid. Stored successfully
IP address 18.222.57.51 is valid. Stored successfully

The following IPs were loaded:
52.14.2.61
18.118.133.97
18.222.57.51

Input key path for stablishing SSH Connection:
D:\Repositorios\Neustar SSI Code Assignment\Neustar.pem
```

Then program will ask the user if he wants to use this key for all the other loaded connections. If not, the user must specify the key's path for every single connection



```
D:\Repositorios\Neustar SSI Code Assignment\Code Assignmet\dist\Menu.exe

LINUX SERVER LOGGER

Input server IPs separated by comma:
52.14.2.61,18.118.133.97,18.222.57.51

IP address 52.14.2.61 is valid. Stored successfully
IP address 18.118.133.97 is valid. Stored successfully
IP address 18.222.57.51 is valid. Stored successfully

The following IPs were loaded:
52.14.2.61
18.118.133.97
18.222.57.51

Input key path for stablishing SSH Connection:
D:\Repositorios\Neustar SSI Code Assignment\Neustar.pem

Do you want to use this key for all current connections? Y/N
y
```

In a similar manner as before, the user will be prompted to introduce the username that will be used for connection via SSH

```
D:\Repositorios\Neustar SSI Code Assignment\Code Assignmet\dist\Menu.exe

Input server IPs separated by comma:
52.14.2.61,18.118.133.97,18.222.57.51

IP address 52.14.2.61 is valid. Stored successfully
IP address 18.118.133.97 is valid. Stored successfully
IP address 18.222.57.51 is valid. Stored successfully

The following IPs were loaded:
52.14.2.61
18.118.133.97
18.222.57.51

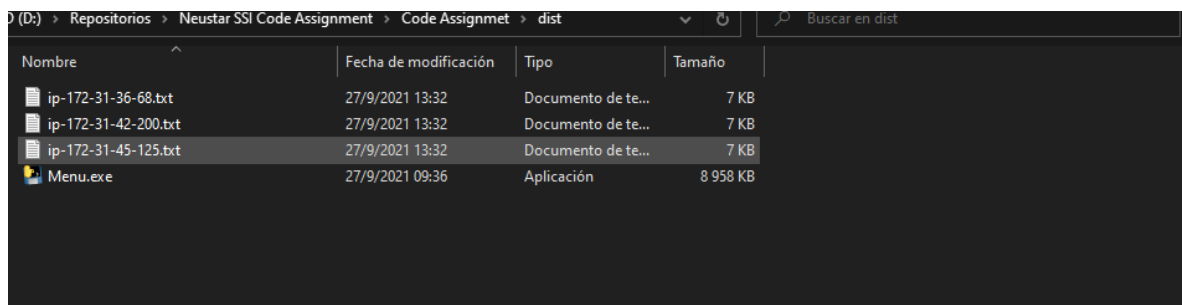
Input key path for stablishing SSH Connection:
D:\Repositorios\Neustar SSI Code Assignment\Neustar.pem

Do you want to use this key for all current connections? Y/N
y

Input user for stablishing SSH Connection:
ubuntu

Do you want to use this user for all current connections? Y/N
y
```

Once all the users are set for each of the connections, the program will connect to each VM and proceed to log the information previous mentioned. Then these logs will be store in the same directory as the executable with the hostname of the VM as the filename.



Nombre	Fecha de modificación	Tipo	Tamaño
ip-172-31-36-68.txt	27/9/2021 13:32	Documento de te...	7 KB
ip-172-31-42-200.txt	27/9/2021 13:32	Documento de te...	7 KB
ip-172-31-45-125.txt	27/9/2021 13:32	Documento de te...	7 KB
Menu.exe	27/9/2021 09:36	Aplicación	8 958 KB