Herramientas utilidades para CLI Linux

En este documento se enumeran algunas de los paquetes y/o herramientas para utilizar desde el CLI (command-line interface) de sistemas operatvivos orientados al comando, del tipo basados en Unix (WSL Windows, VM Fedora Linux, o similares). Mostrándose algunos ejemplos. Para esta exposicion se despliegan VMs sobre dos laboratorios.

Básicamente en este documento se muestran los siguientes comandos:

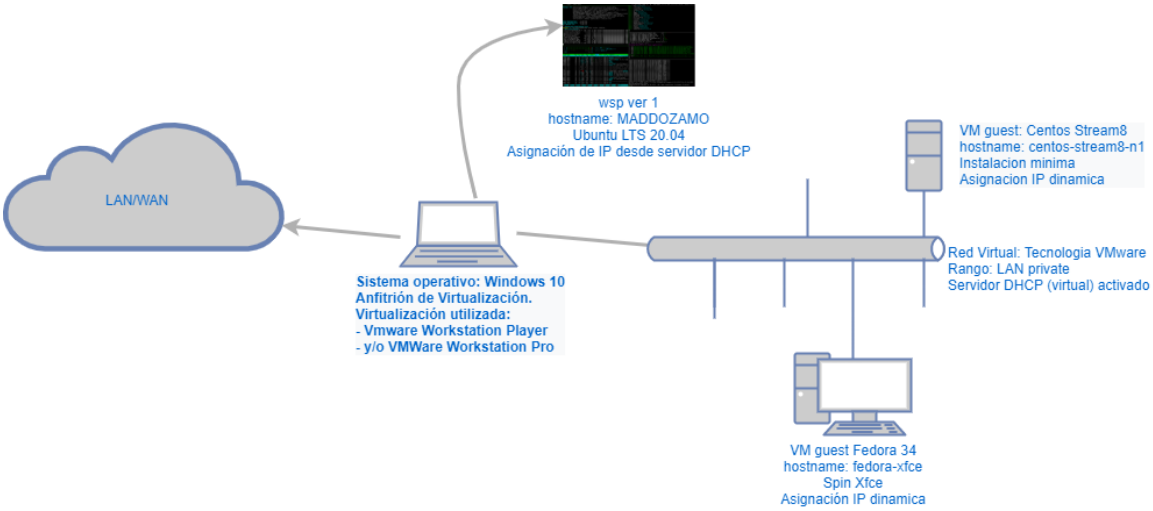
- `nmap` para descubrir los hosts encendidos actuales.
- `sshpass` --> para automatizar el ingreso de la password, en la autenticación sobre servidores ssh, (configuración de acceso vía contraseñas (***"advertencia"***)).
- Multiplexor de terminales --> uso de `tmux`.
- Ejecución de comandos en paralelo, sobre varios host/servidores remotos a la vez, a través de SSH. En particular uso de `pssh`

Laboratorios utilizados para este trabajo

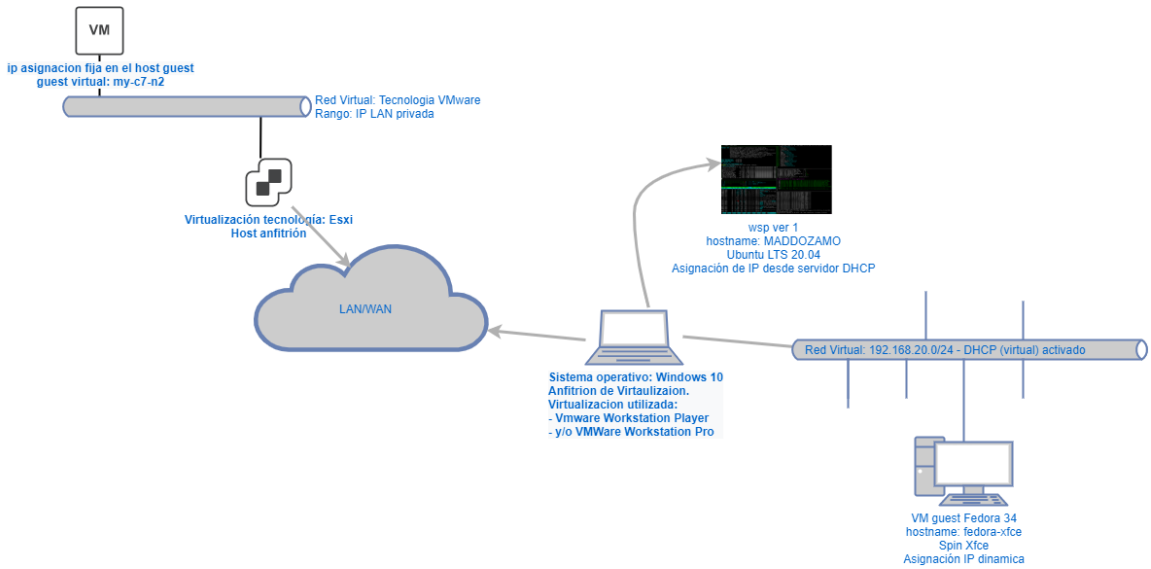
Para este trabajo se han desplegado [dos laboratorios]. Estos son:

- El laboratorio 1 de pruebas, sobre un anfitrión basado en tecnología VMWare, en un Windows 10. Se intenta aquí mostrar/debatir, una arquitectura que sirva de base a un despliegue común o consensuado. Mostrando algunas sesiones de trabajo, haciendo uso de un CLI Linux unificado o único/común.
- Basado en el anterior despliegue, se proveen otras VM, resultando el laboratorio 2, sobre el cual se ejecutan los comandos que se detallan en este documento.

> Laboratorio 1



> Laboratorio 2



Introducción a gestion de paquetes

Instalación de paquetes sobre Fedora 31 y/o superiores

`sudo dnf -y install rsync nmap lsof`

Instalación de paquetes sobre WSL Ubuntu LTS 20.04 y/o superiores

> Recordemos que WSL (principalmente la versión WSL 1, usada en este trabajo) esta basado en Hiper-V, y ***no es un kernel "puro" de Linux***, esto ocaciona que algunos de los paquetes no funcionaran adecuadamente, como si fuese un contenedor o VM real Linux. Paquetes como `netcat`, `tcpdump`, `systemctl` no se ejecutaran adecuadamente.
>
> Los paquetes que se instalan a continuación, han sido probados sobre un Ubuntu 20.04 (expuesta en este documento), funcional sobre WSL ver. 1.

```
sudo apt update && sudo apt -y upgrade
sudo apt install htop rsync dnsutils htop lsof
```

Instalación de paquetes sobre RHEL ver 8 y/o derivados (CentOS Stream, Alma Linux, Rocky Linux, Oracle Linux, etc)

```
`sudo dnf -y install htop rsync nmap`
```

Repositorio epel sobre CentOS Stream (y/o derivados)

A continuación se realiza:

- Agregar repositorio.
- Desactivarlo y establecer prioridad del repositorio
- Instalar `htop`

Nota: La siguiente secuencia de comandos, deberían funcionar en CentOS Stream 8 y/o derivados.

```
# Escalar de privilegio
sudo su
# Instalar repositorio EPEL
dnf -y install epel-release
# Desactivar el repositorio y establecer prioridad de sus paquetes
p=20; sed -i -e "s/enabled=[0-1]$/enabled=0\npriority=${p}/g"
/etc/yum.repos.d/epel.repo
# Instalar paquete desde repo desactivado
dnf -y --enablerepo=epel install htop
```

Laboratorio de pruebas

Las siguientes prácticas, han sido realizadas para la elaboración de este documento. En cada una de las mismas, se indicará desde que host (anfitrión desde el WSL - Ubuntu o el guest virtual correspondiente del laboratorio) es que se realizan las mismas.

Practicando el comando `nmap`

Escanear/descubrir mi red de prueba

Escanear la red interna virtual del rango `192.168.20.0/24` (red privada virtual del laboratorio desplegado), para saber que host estan encendidos.

Resolución:

Desde el guest `centos-stream8-n1` se ejecuta el comando `nmap` siguiente, para descrubrir que host están encendidos en nuestra LAN virtual. Con lo recabado es que serán realizadas las pruebas. Se ejecuta:

```
# nmap -sP 192.168.20.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2021-10-07 11:16 CEST
Nmap scan report for 192.168.20.1
Host is up (0.00097s latency).
MAC Address: 00:50:56:C0:00:08 (VMware)
Nmap scan report for 192.168.20.2
Host is up (0.00011s latency).
MAC Address: 00:50:56:E0:38:16 (VMware)
Nmap scan report for 192.168.20.133
Host is up (0.00017s latency).
MAC Address: 00:0C:29:73:03:BD (VMware)
Nmap scan report for 192.168.20.254
Host is up (-0.10s latency).
MAC Address: 00:50:56:F8:FA:CE (VMware)
Nmap scan report for 192.168.20.138
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 4.38 seconds
```

Consultar mas datos sobre los host

Escanear la red interna virtual del rango `192.168.20.0/24` (red privada virtual del laboratorio desplegado), consultando que puertos abiertos, e intentar descubrir que sistema operativo poseen.

Resolución:

Desde el mismo guest `centos-stream8-n1` se puede ejecutar el comando `nmap` siguiente (se resume la salida para acortar este documento):

```
# for i in $(nmap -sP 192.168.20.0/24|grep 'Nmap scan report for'|cut -d' ' -f5);
do echo "--> Host ${i}"; nmap -O ${i}; echo -e "<---\n"; done
...
...

--> Host 192.168.20.138
Starting Nmap 7.70 ( https://nmap.org ) at 2021-10-07 12:09 CEST
Nmap scan report for 192.168.20.138
Host is up (0.000030s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.7 - 3.10
Network Distance: 0 hops

OS detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.87 seconds
...
...
```

Practicando sobre servidor de ssh

Comando sshpass

> Valido principalmente para conectarse a servidores ssh con validación por contraseña)
>
> La utilización de este comando tiene sus precauciones. La contraseña debe ser ingresada en texto plano en la propia linea de comando al invocar el comando.
>
> Para enmascarar esto se mostraran dos estrategíaas.

Con ***SSHPass*** se puede automatizar el ingreso de la contraseña en las conexiones hacia servidores **ssh**, donde su autenticación este basada en "contraseña".

1. A continuación se instala el paquete `sshpass` en dos de los host de nuestro laboratorio.
2. Se define también la variable de entorno SSHPASS (esta evita tener que pasar la propia contraseña sobre la línea de comando, al invocar el mismo).
3. Se propone también un ejemplo de definir contraseñas como variables al entorno del usuario, para que sean pasadas de modo oculto, en las propias conexiones de comandos emitidos usando `ssh`

Instalar sshpass

Instalación sshpass en WSL - Ubuntu 20.04 LTS

```
sudo su
apt update && apt -y upgrade && apt -y install sshpass
```

Instalación sshpass en Fedora 34 y/o derivados

```
sudo dnf -y update
sudo dns -y install sshpass
```

Uso básico de sshpass

A continuación se muestra un uso básico inicial de `sshpass`.

> `sshpass` Invocado con la opción `-p` pasamos el `password` de forma visual en la pantalla, en el propio CLI de la consola.

Ejemplo: Consultar que versión de Linux basado en Red Hat tiene instalado el IP `192.168.20.138`. Como todos los comandos invocados con `ssh` (o basados en él), la siguiente ejecución será realizada en el host destino.

```
sshpass -p <PASSWORD_TEXTO_PLANO> ssh root@192.168.20.138 cat /etc/redhat-
release
```

Establecer SSHPASS como variable de entorno

Al invocar `sshpass` con la La opción `-e`, el comando hace que el password sea establecido desde la variable de entorno de sesión `SSHPASS`. En el siguiente ejemplo se establece esta variable en la sesión del usuario. De este modo logramos ocultar o enmascar la contraseña, al invocar la ejecución del `ssh`

Ejemplo: Verificar cuanto tiempo hace (comando `uptime`) que el host remoto `192.168.20.138` esta encendido, invocando `ssh` mediante `sshpass` con la opción `-e`.

Resolución:

[1] Definir la varible `SSHPASS` al entorno de sesión del usuario.

> En el siguiente ejemplo mostrado, la variable se establece en el fichero ~/.bashrc del usuario autenticado para que el cambio sea permanente.

```
# Verificar que no este asignada, ni establecida
echo $SSHPASS
# Agregar variable a fichero de entorno inicial
echo -e "\n# Add my vars\nexport SSHPASS='<MY_PASSWORD_HOST_REMOTE>'" >>
~/.bashrc
```

[2] Verificación

Una vez creada asignada la variable de entorno `SSHPASS` se puede invocar `ssh` (o comandos basados en el protocolo/aplicación como `rsync`, `scp`, `etc`) y pasar el password mediante `sshpass`. Es lo que se realiza a continuación.

```
# Cargar el environment del usuario (sin tener que desloguearse)
source ~/.bashrc
# Verificación
## Emitir el comando uptime en el host remoto
sshpass -e ssh root@192.168.20.138 uptime
# Si el comando anterior da error o no muestra retorno (como que no se ha
ejecutado en el remoto), es quizas la primera vez que se intenta conectar al
servidor 192.168.20.138 mediante ssh. Por lo que emitiendo la siguiente opción se
acepta/fuerza que las key sean agregadas al ~/.ssh/known_hosts
sshpass -e ssh -o "StrictHostKeyChecking no" root@192.168.20.138 uptime
```

Nota: `'<MY_PASSWORD_HOST_REMOTE>'` es el password del usuario del servidor ssh al que se quiere conectar.

Multiplexor de terminales

Multiplexar la terminal tty del CLI de Linux suele ser practico para gestionar los servidores desde un ordenador remoto. Con este tipo de programas nos permite tener múltiples terminales dentro una sola ventana.
Existen varias alternativas para esta funcionalidad. Algunas de las posibles son:

- `Terminator` ---> requiere de librerias mas propias del subsistema ambiente gráfico de Linux.
- `Screen` ---> es quizas la herramienta de este tipo, con mayor trayectoria.
- `Tmux`
- `Byobu` ---> soporta `tty` virtuales tanto de `tmux` como de `screen`

En este documento se presenta `tmux`.

Tmux

Instalar tmux en Fedora 21 y/o superiores/derivados

```
dnf -y update
dnf -y install tmux
```

Instalar tmux sobre WSL - Ubuntu 20.04 LTS (y/o derivados/similares)

```
apt -y update && apt -y upgrade
apt -y install tmux
```

Uso básico de tmux

Combinación de teclas

Sesiones - sessions:
`:new<CR>` new session
`s` list sessions
`$` name session

Ventanas - windows (tabs):
`c` create window
`w` list windows
`n` next window
`p` previous window
`f` find window
`,` name window
`&` kill window

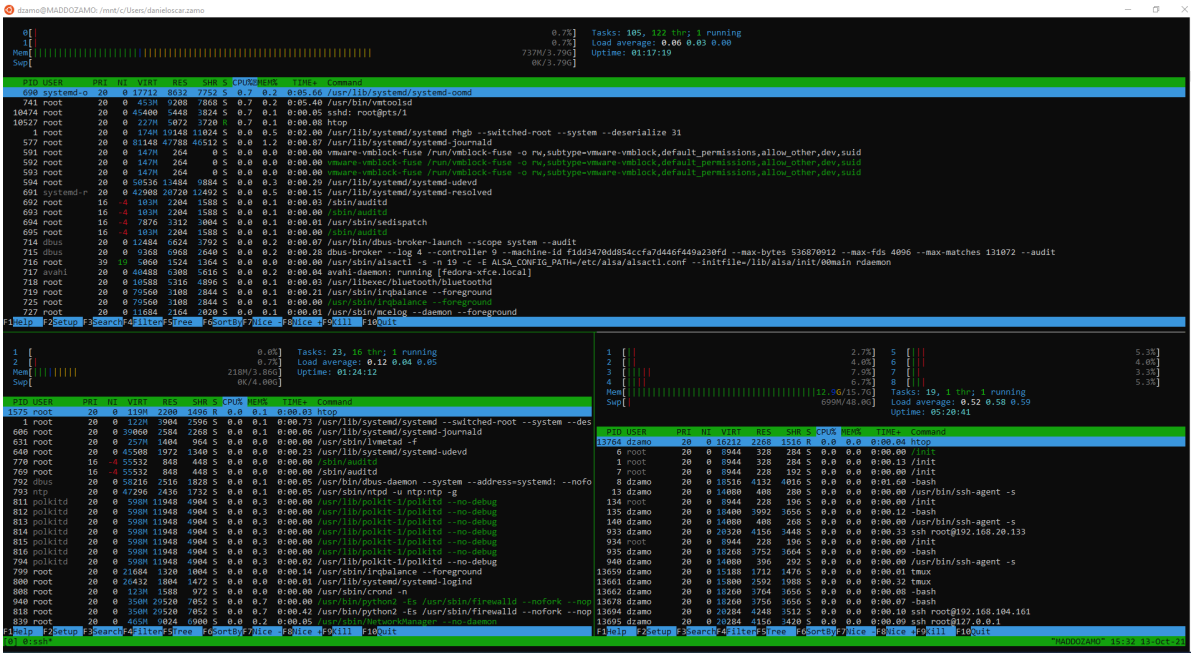
Paneles - Panes (splits)
`%` vertical split
`"` horizontal split

`o` swap panes
`q` show pane numbers
`x` kill pane
`+` break pane into window (e.g. to select text by mouse to copy)
`-` restore pane from window
`␣` space - toggle between layouts

Ejemplo: Conectarse a los 3 servidores y ejecutar el comando `htop`

- Combinaciones de teclas usadas:
  - `CTRL+b+"` --> dividir pantalla en dos paneles horizontales
  - `CTRL+b+%` --> dividir pantalla en dos paneles verticales
  - `CTRL+b+:setw synchronize-panes [on/off]` --> sincronizar ventanas, para ejecutar un mismo comando en los diferentes `tty`

En la captura se muestra la salida del comando `htop` lanzado en forma sincronizada sobre las VM: `WLS - Ubuntu 20.04`, `fedora-xfce` y `my-c7-n2`



Ejecutando comandos en paralelo, sobre varios servidores

Aunque en este tipo de paquetes tambien son varias las opciones, en este apartado se muestra la instalacion y uso básico/inicial de `pssh`. Se utiliza esta, pues es una herramienta que se puede instalar y usar funciona tanto en WSL - Ubuntu como en un Linux puro/completo.

Instalando pssh sobre debian y/o derivados (ejemplo WSL - Ubuntu 20.04 LTS)

```
sudo su
apt update && apt -y upgrade
apt -y install pssh
```

Instalando pssh sobre CentOS 7

```
yum -y update
yum -y --enablerepo epel install pssh
```

Instalando pssh sobre CentOS8 Stream (y derivados) y/o Fedora 21 o superior

```
dnf -y update
dnf -y --enablerepo epel install pssh
```

Referencias

- 4 Useful Tools to Run Commands on Multiple Linux Servers
- How to use parallel ssh (PSSH) for executing commands in parallel on a number of Linux/Unix/BSD servers