

Laboratorio: Trabajo en remoto con VSCode



Aprendizaje
Profundo

Grado en Ingeniería y Ciencia de datos (Universidad de Oviedo)

Pablo González, Pablo Pérez
{gonzalezgpablo, pabloperez}@uniovi.es
Centro de Inteligencia Artificial, Gijón

Introducción

Cuando trabajamos en aprendizaje profundo, es muy normal que las máquinas donde lancemos los trabajos no estén físicamente donde estamos nosotros. Normalmente además se tratará de servidores **Linux** con una o más **tarjetas gráficas instaladas**.

Hay una serie de herramientas que nos facilitan el trabajo en remoto. Por citar algunas:

- ① VSCode (permite a través de plugins el trabajo en remoto)
- ② Screen (útil para lanzar entrenamientos largos)

Herramientas para el trabajo en remoto

La herramienta principal utilizada para el trabajo en remoto es **SSH**. Es una herramienta muy potente que nos permitirá acceder a una consola de la máquina así como a redireccionar puertos a través de túneles para el acceso a servicios de la misma.

¿Qué es SSH?

¿Qué es SSH?

- SSH (Secure Shell) es un protocolo de red que permite a los usuarios acceder y gestionar de forma segura dispositivos y sistemas remotos.
- Proporciona un canal seguro de comunicación sobre redes inseguras, como Internet, utilizando técnicas de cifrado para proteger los datos transmitidos, reemplazando a herramientas como Telnet, que envían información sin cifrar, lo que representa un riesgo para la seguridad.



¿Como conectarse?

Necesitamos un **cliente de SSH**. En linux ya lo tenemos instalado por defecto en nuestra consola. En Windows, si no tenemos powershell, podemos utilizar **Putty**.

```
ssh user@machineip
```

Autenticación

Podemos realizar la autenticación usando una **password** tradicional o usando un mecanismo de **clave pública y privada**.

VSCode

Más adelante veremos que VSCode tiene su propio cliente de SSH integrado.

Comandos útiles [http]

El primer comando útil en un servidor para conocer su estado general es **htop**. Con él podemos ver el estado de los procesos, manejarlos, la memoria, entre otros muchos datos.

```
1 [|||||100.0%] 6 [|||||100.0%] 11 [|||||100.0%] 16 [|| 1.3%]
2 [|||||100.0%] 7 [|||||0.0%] 12 [|||||100.0%] 17 [|||||100.0%]
3 [|||||100.0%] 8 [|||||100.0%] 13 [|||||100.0%] 18 [|||||100.0%]
4 [|||||100.0%] 9 [|||||100.0%] 14 [|||||100.0%] 19 [|||||100.0%]
5 [|||||100.0%] 10 [|||||100.0%] 15 [|||||100.0%] 20 [|||||100.0%]
Mem[|||||10.96/62.66] Tasks: 132, 573 thr; 19 running
Swp[|||||7.496/31.06] Load average: 18.00 18.00 17.18
Uptime: 309 days(!), 23:23:24
```

DISK READ	PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
no perm	14359	quevedo	20	0	726M	408M	54076	R	100.	0.6	0:52.60	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14153	quevedo	20	0	1070M	752M	54356	R	100.	1.2	5:13.25	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14171	quevedo	20	0	1062M	744M	54288	R	100.	1.2	4:45.22	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14298	quevedo	20	0	889M	570M	54104	R	100.	0.9	1:46.76	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14271	quevedo	20	0	1077M	759M	54292	R	100.	1.2	2:03.37	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14350	quevedo	20	0	727M	408M	53640	R	100.	0.6	0:57.30	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14378	quevedo	20	0	671M	352M	54232	R	100.	0.6	0:42.58	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14144	quevedo	20	0	1057M	740M	54556	R	99.8	1.2	5:14.62	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14135	quevedo	20	0	1073M	755M	54080	R	99.8	1.2	5:24.02	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14307	quevedo	20	0	871M	552M	53852	R	99.8	0.9	1:46.37	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14289	quevedo	20	0	894M	576M	54336	R	99.8	0.9	1:54.42	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14280	quevedo	20	0	932M	613M	53672	R	99.8	1.0	1:54.87	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14369	quevedo	20	0	886M	368M	54408	R	99.8	0.6	0:43.10	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	13977	quevedo	20	0	1091M	773M	54044	R	99.8	1.2	9:54.77	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14387	quevedo	20	0	646M	327M	54000	R	99.8	0.5	0:33.34	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14396	quevedo	20	0	641M	323M	54444	R	99.8	0.5	0:27.46	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	13955	quevedo	20	0	997M	679M	54076	R	99.8	1.1	11:28.11	python 1B_HacerExperimentosEvaluacion.py //Folds
no perm	14162	quevedo	20	0	1075M	757M	54024	R	99.1	1.2	5:03.09	python 1B_HacerExperimentosEvaluacion.py //Folds
0.00 B/s	14368	pgonzalez	20	0	29068	4416	3196	R	1.3	0.0	0:00.68	htop
0.00 B/s	28439	pgonzalez	20	0	6349M	179M	25020	S	1.3	0.3	22:04.89	/home/pgonzalez/.dropbox-dist/dropbox-lnx.x86_64-
F1Help	F2Setup	F3Search	F4Filter	F5Tree	F6SortBy	F7Nice	F8Nice	F9Kill	F10Quit			

Comandos útiles [nvidia-smi]

Otro comando muy útil es **nvidia-smi**. Con él podemos ver el estado de la tarjeta gráfica. La memoria ocupada, su utilización, temperatura, etc.

```
Cada 0,1s: nvidia-smi                                     Wed Jul 26 11:10:07 2023

Wed Jul 26 11:10:07 2023
+-----+
| NVIDIA-SMI 450.51.05      Driver Version: 450.51.05      CUDA Version: 11.0      |
+-----+
| GPU   Name               Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M.         |
+-----+-----+
|  0  TITAN Xp              Off        | 00000000:17:00.0 Off |           N/A       |
| 23%   43C   P2              60W / 250W | 811M1B / 12196M1B |           1%      Default |
|                                           N/A               |
+-----+-----+
|  1  TITAN Xp              Off        | 00000000:65:00.0 Off |           N/A       |
| 23%   34C   P8              9W / 250W | 2M1B / 12194M1B |            0%      Default |
|                                           N/A               |
+-----+-----+

+-----+
| Processes: |
| GPU   GI    CI        PID   Type   Process name                  GPU Memory |
| ID   ID     ID              |                 | Usage      |
+-----+-----+
|  0   N/A   N/A         15029    C      python                       809M1B   |
+-----+
```

Comando watch

El comando **watch** se usa en combinación con **nvidia-smi**. Permite ver la salida del comando cada cierto tiempo: `watch -n 0,1 nvidia-smi`.

Usando VSCode

Usando la extensión **Remote - SSH** (`ms-vscode-remote.remote-ssh`) podemos abrir una conexión remota por SSH sobre cualquier máquina a la que tengamos acceso. Esto nos permite:

- **Abrir y almacenar archivos** en el almacenamiento del servidor.
- **Ejecutar y depurar** programas usando los recursos del servidor.
- Crear **redirecciones de puertos** automáticamente para servicios lanzados en el servidor.
- Utilizar **terminales** remotas sin necesidad de ningún programa extra.

VSCode-server

También existe la opción de correr de manera íntegra VSCode en el servidor y acceder a él a través de un navegador web. Para ello tendremos que instalar <https://code.visualstudio.com/docs/remote/vscode-server>.

Usando VSCode



Lanzando procesos [nohup y screen]

Generalmente los entrenamientos en el aprendizaje profundo son procesos que están pueden llegar a ejecutarse un largo tiempo. Es importante conocer como lanzarlos para que Linux no los termine cuando cerremos la terminal o el equipo desde el que estamos trabajando. Las dos mejores opciones para ello son:

- **nohup**. nohup comando [opciones] &. Se utiliza para ejecutar un programa o comando en segundo plano y evitar que se detenga cuando se cierra la terminal o sesión de inicio de la que se lanzó.
- **screen**. Se utiliza para crear y administrar sesiones de terminal virtuales. Estas sesiones son como ventanas separadas en las que puedes ejecutar comandos o programas de forma independiente y luego volver a conectarte a ellas en cualquier momento, incluso después de cerrar la sesión original. Una guía rápida de screen se puede encontrar [aquí](#).