*18/09/2020*

**Clase 1**

1. Ir pensando ideas sobre temas para el TFM
   1. Es importante elegir un tema que nos guste y del que sepamos un poco para enfocarlo bien (e.g. SR-vs-CG, aerolíneas…)
   2. Buscar temas de los que se puedan obtener datos (preguntar en Altran? Airbus? Solamente si acceden a que sea público)
2. Diseño de productos de datos:
   1. Plantear la pregunta inicial
   2. Pensar si hay datos al respecto 🡪 si no, volver a preguntar
3. Hay 2 tutorías:
   1. 1ª: enfocada a plantear el tema
   2. 2ª: enfocada a desarrollar las ideas ya sabiendo ML
4. El TFM y el Github serán nuestro portfolio, así que mantenerlo actualizado continuamente, no todo al final

VM

1. Settings:
   1. Drag/Drop 🡪 Disabled
   2. Shared clipboard 🡪 Probar Enabled (pero no suele funcionar)
   3. Shared Folders 🡪 Cambiar la que pone “…dani” por uno propio, que va a ser un directorio compartido de la máquina invitada (VM) con la máquina anfitrión 🡪 Recomendación: “C:\Users\turge”
2. Si sale la pantalla en negro tras introducir la contraseña (“master”), pulsar “Ctrl (dcho) + F”
3. El 90% de los servidores son Unix, por lo que hay que aprender a manejar la terminal de Linux
4. En VirtualBox, es importante la tecla “Anfitrión” o “Host”, que por defecto es el Ctrl (dcho)
   1. La tecla Host se reserva para comunicar el VB con el anfitrión
   2. Para hacer “alt+tab” con el anfitrión, pulsar Y SOLTAR el botón Host
5. Terminal = Prompt
   1. Ctrl+shift+c = copiar
   2. Ctrl+shift+v = pegar
   3. “dsc” = usuario en Xubuntu
   4. “echo” repite lo que yo le pido: USER, TERM, etc. Cada una de ellas son variables
   5. Para escribir, usar las comillas dobles
   6. Usar siempre el teclado inglés
   7. Para escribir variables, utilizar la sintaxis: $(nombre\_variable)
   8. Si algo se queda colgado 🡪 Ctrl+c 🡪 Comando de interrupción
   9. Carácter de escape 🡪 \ (para que no te tenga en cuenta el siguiente carácter como carácter especial)
   10. pwd 🡪 “print working directory”
   11. cd .. = subir un nivel
   12. cd ~ = volver a base
   13. cd - = volver a la carpeta anterior 🡪 “atrás”
   14. drwxrwxr-x 26 dsc dsc 🡪 permisos (ojo con cambiarlos!! 🡪 chmod)
       1. r = read (4)
       2. w = write (2)
       3. x = execute (1)
       4. Se dividen en “yo”, “mi grupo” y “resto del mundo”
       5. 644 = yo leo y escribo / mi grupo lee / mundo lee
   15. NO UTILIZAR ESPACIOS EN UNIX!!! 🡪 Usar barra baja en su lugar (\_)
   16. NO UTILIZAR “rm –rf” DESDE LA RAIZ!!! 🡪 Utilizar en su lugar “rm –rf (rutas)”
6. En cualquier proyecto, usar Checkpoints con Git
   1. Git clone https//:… 🡪 Clona un repositorio de github con la URL
      1. “master” designa el repo original
      2. git reset HEAD –hard 🡪 vuelve al estado original de “master” 🡪 checkpoint por si la lías
      3. git add 🡪 añade archivos nuevos PREVIO al commit
      4. git commit 🡪

*19/09/2020*

**Clase 2**

1. git config –global user.name “malonsol”
2. git config –global user.email “m.alonso.lopez123@gmail.com”
3. Estando desde el Home 🡪 cat .gitconfig 🡪 Proporciona el usuario activo de Git
4. Para importar el repo de GitHub 🡪 git clone <url\_repo>
   1. Si se usa 🡪 git clone <url\_repo> Otro\_nombre 🡪 Lo clona con otro nombre, sino utiliza el del repo de GH por defecto
   2. O incluso en otro directorio 🡪 git clone <url\_repo> ../Pictures/Otro\_nombre\_2 🡪 En este ejemplo, partiendo de home/dsc/master\_kschool, subimos un nivel con “../”, y luego vamos a Pictures
5. Si haciendo 🡪 ls –a 🡪 aparece un .git 🡪 significa que estamos dentro de un repositorio
6. El repo local está en .git
7. Para hacer el commit 🡪 git commit -m "first commit"
8. Tras el git add 🡪 se hace l git add –u 🡪 para actualizar antes del commit
9. Tener distintos repos:
   1. Master
   2. TFM
   3. Python
   4. R
   5. …
10. Desde Home (cd) 🡪 cat .history 🡪 da un historial de todos los commandos introducidos
11. Copiar el archivo history del Home a un sitio concreto 🡪 cp ~/.history ./
    1. cp = copy
    2. ~/ = carpeta donde se encuentra el archivo a copiar
    3. .history = archivo a copiar
    4. ./ = carpeta donde lo quiero pegar (en este caso, en la ruta en la que te encuentras)
12. Se pueden crear varios directorios simultáneamente: mkdir fld1 fld2 fld3 …
13. Para copiar un archivo como oculto 🡪 ./.text\_file.txt🡪 ./. ¡!!
14. A mi derecha Javi, trabaja para Sopra.
15. Dos a mi derecha Jaime, trabaja en EY
16. Tres tipos:
    1. Scripts
    2. Binarios
    3. Alias
       1. Los alias son para sustituir comandos, por ejemplo 🡪 ll='ls -lhX'
       2. Se pueden definir los que queramos, por ejmplo cambiar “git status” por “gs” 🡪 alias gc='git commit -m'
       3. Cada vez que se sale de Terminal (exit), se pierden los nuevos alias
       4. [dsc@vm:~] [base] 18s % cp .zshrc .zshrc.backup 🡪 Hacemos un backup de los alias originales
17. which git 🡪 nos indica qué es 🡪 Por ejemplo which ll
    1. which python 🡪 dónde está ubicado python
18. head / tail
    1. Para coger la fila 8, por ejemplo 🡪 head -8 optd\_aircraft.csv | tail -1
    2. cat -n optd\_aircraft.csv | head -8 | tail -1
       1. Eso podemos querer volcarlo en otro archivo 🡪 > 🡪 ejemplo 🡪 echo Test >test\_file
       2. Si queremos **añadir** 🡪 >> 🡪 echo Test2 >>test\_file
       3. Al contrario, para acceder al archivo, se puede usar 🡪 < 🡪 <test\_file wc
19. word count 🡪 wc optd\_aircraft.csv 🡪 print line, word, and byte counts
    1. Es conceptualmente diferente a 🡪 cat optd\_aircraft.csv | wc
       1. “ | “ se usa para emplear la salida de la primera operación (cat optd\_aircraft.csv) con el segundo operador (wc) 🡪 Esto es posible porque wc espera un texto, y la salida de la primera operación es texto
       2. Guardar los últimos 20 comandos del history en un archivo 🡪 tail -20 ~/.history >last\_20\_commands
20. Búsqueda de archivos o datos dentro de archivos 🡪 find 🡪 es muy potente combinándolo con sus distintos parámetros
21. –exec 🡪 ejecuta una acción sobre lo anterior 🡪 find . -type f -iname "\*.csv" -exec echo "FOUND ONE" \; 🡪 printea FOUND ONE cada vez que encuentra uno de esos archivos
22. el “;” se usa para ejecutar dos líneas de comando escribiéndolos en la misma línea
23. Jobs 🡪 muestra los procesos suspendidos
    1. kill %N 🡪 “mata” el proceso N
    2. Los procesos se suspenden con ctrl+Z

*25/09/2020*

**Clase 3**

1. Antonio Almagro 🡪 ingeniero aeronáutico, dos años en aerolínea y después doctorado en Mecánica de Fluidos, donde empezó a utilizar enormemente los datos. Estancia en San Diego
2. Cluster = muchos ordenadores conectados
3. Por lo general, es más cómodo usar rutas relativas, pero son más seguras las rutas absolutas
4. JAMÁS HACER 🡪 rm / 🡪 Borra TODO el ordenador
5. La extensión “.md” viene de lenguaje Markdown 🡪 Permite escribir texto con formato muy rápido
6. El “add” es “existo” (para que el repo lo reconozca) y el “commit” es para que lo introduzca
7. cat ~/.gitconfig 🡪 Muestra las credenciales del Git activo
8. git ls-files 🡪 Muestra los archivos trackeados en el repo
9. git checkout README.md 🡪 Recupera un archivo del repo en remoto tras haberlo borrado del repo físico
10. Cuando se hace un commit, es importante poner un texto representativo para identificarlo después
11. git checkout master 🡪 vuelve al último commit
12. Las variables de entorno van como “$HOME”, mientras que los comandos van como “$(date)”
13. seq 1 2 10 🡪 crea una secuencia del 1 al 10 con paso 2
14. diff 🡪 sirve para comparar dos archivos
15. uniq 🡪 se queda con los valores únicos, pero compara uno a uno!!!
    1. Por eso se combina muchas veces con “sort”
    2. –c 🡪 cuenta repeticiones
    3. –d 🡪 muestra solo las líneas duplicadas
16. El pipeline ( | ) se usa para utilizar el output de un comando como input del siguiente comando
17. grep –E 🡪 Activa las expresiones regulares