**Comandos BASH para trabajar con grandes ficheros de datos.**

Vamos a empezar a hacer la maleta básica de comandos para cualquier analista o ingeniero de datos que quiera empezar a trabajar con BigData.

**Recursos del sistema**

Normalmente trabajaremos en máquinas Linux y muchas veces alojadas en la nueva por la facilidad de contratar nodos. En este sentido, al entrar en un proyecto lo primero que necesitaremos será entender los recursos (Básicamente disco, RAM y CPUs) que tenemos disponibles(No empecéis a mover datos sin saber los recursos que tenemos disponibles):

# Saber cuanto disco tenemos disponible (-h para ver las unidades fácil en Gb, Mb, etc.)  
df -h  
  
# Saber cuanta RAM tenemos disponible  
free -h  
  
# Saber el número de CPUs  
nproc  
  
# Información de las CPUs  
lscpu  
cat /proc/cpuinfo

¡Importante! No empieces a exportar/descomprimir/mover grandes ficheros de datos sin saber que tendremos suficientes recursos para llevar a cabo la tarea. Quedarte sin disco puede colgar la máquina o no tener suficiente RAM y CPUs puedes no poder terminar la tarea.

**Leer ficheros de textos**

Los formatos de datos más habituales para trabajar son CSV y JSON donde normalmente en cada línea tienes un registro. Es importante tener herramientas para obtener información rápido de lo que hay dentro:

# Número de líneas de un fichero(con -w encontrarás el número de palabras)  
wc -l <fichero>  
  
# Leer las 5 primeras filas de un fichero  
head -5 <fichero>  
  
# Leer las 5 últimas filas de un fichero  
tail -5 <fichero>  
  
# Leer la fila 28 de un fichero  
head -28 <fichero> | tail -1  
  
# Obtener valores de un documento JSON  
echo '{ "foo": 123, "bar": 456 }' | jq '.foo'

**Unir y dividir ficheros**

Otra funcionalidad que debemos dominar es la de dividir y después unir ficheros, ya que es habitual tener que dividir la información para cargarla en clusters o simplemente enviarla más fácilmente. Para esto tenemos los comandos ‘split' y ‘cat':

# Dividir un fichero binario en bloques de 1000000 bytes(Nos generará ficheros

# con el nombre ‘divididoaa,dibididoab, etc.')

split -b 1000000 <fichero\_original> <fichero\_dividido>  
  
# Dividir un fichero de texto en bloques de 2000000 líneas.  
split -l 200000 <fichero\_original> <fichero\_dividido>  
  
# Unir ficheros  
cat dividido\* > <fichero\_original>

**Comprimir y descomprimir ficheros**

El formato más usado para comprimir y descomprimir ficheros BigData es el Gzip(.gz). Para comprimir y descomprimir (¡Atención! Comprime y descomprime sin dejar el fichero original):

# Comprimir  
gzip <fichero\_original>  
  
# Descomprimir  
gzip -d <fichero\_original>

**Procesar un fichero de texto JSON línea a línea y convertirlo a CSV**

El siguiente script Bash procesa un fichero de texto donde en cada línea hay un documento JSON con la información del registro y genera otro documento CSV con los datos procesados (Tanto el fichero de entrada(JSON) como el de salida(CSV) se pasan como parámetros):

#!/bin/bash  
  
fIn="$1"  
fOut="$2"  
  
echo "keyword,search\_volume,cpc,com  
  
while IFS= read -r line  
do  
 jq -r '[ .keyword, .keyword\_info.search\_volume, .keyword\_info.cpc,   
.keyword\_info.competition ] | @csv' <<< $line >> $fOut  
done < "$fIn"

**Tarea 1. Procesar ficheros de datos JSON y CSV en Bash, Python, Java y Go.**

En tu trabajo han llegado archivos JSON de enorme tamaño. Pero en tu empresa se trabaja con archivos CSV. Te solicitan que realices un estudio para tomar una decisión de cual sería el mejor lenguaje script para procesar esos archivos JSON y convertirlos en CSV. Te dan cuatro posibles opciones:

* Bash
* Python
* Java
* Go

Diseña un plan de pruebas que demuestren a tus jefes cual serían el lenguaje más adecuado para realizar esa tarea.

Nota:

Te dejo una carpeta para que realices la práctica. Lee el archivo readme.md, para saber que datos tienes.

El trabajo consiste en trabajar con un archivo JSON de gran tamaño y convertirlos a CSV. Tendrás que utilizar scripts en Bash, Python, Java y Go para realizar ese trabajo y valorar cuanto tiempo tarda cada script en ejecutarse.