

# Files, Shell, Bash, Scripting

On this first applied class we will start looking at files. The most basic unit to store information.

When dealing with very large amounts of data, visual tools like Excel or single host in-memory programming languages like python have a big performance degradation to the point of not working at all.

During the course of these lectures, we will learn how to work in environments capable of horizontally scale to datasets long above the 1M max rows of excel.

One of the original ways to use computers to process data was developed for the Unix Environment and followed its philosophy. Shortly summarized as:

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams because that is a universal interface.

## **Why starts this way?**

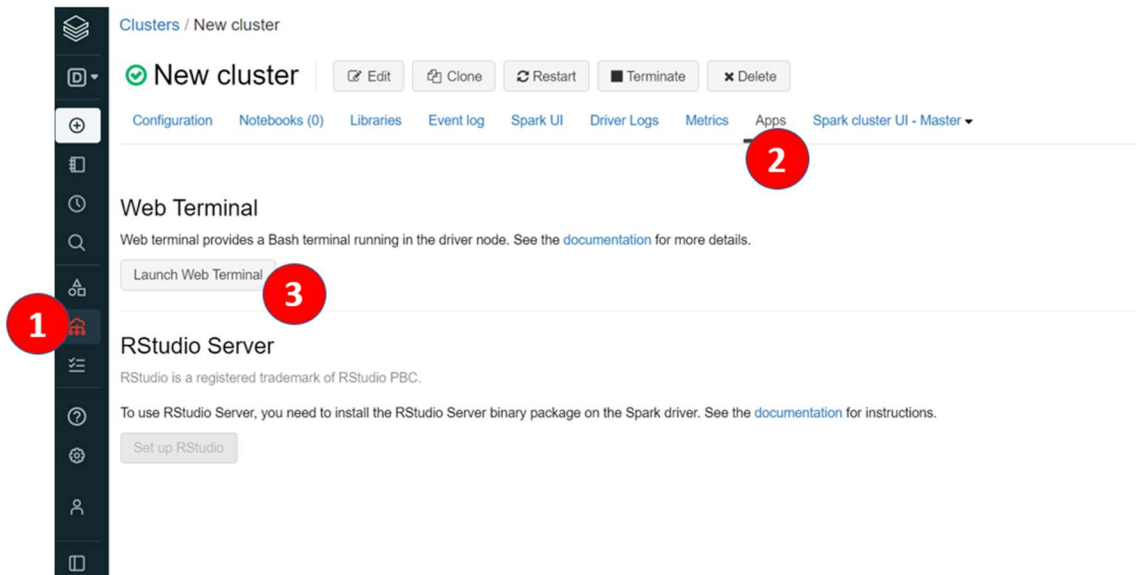
- Ease of execution of commands (no need to copy and paste every time)
- Powerful programming constructs

This mantra evolved extremely well, to the point that most of the commands developed within the Unix ecosystem, more than 30 years ago, are still relevant today. Not only that, but state of the art tools like Spark, which we will learn later in the course, follow a similar approach.

Today the course will be a brief introduction to Unix commands, often called Shell commands, and how they can be useful on modern day-to-day data processes.

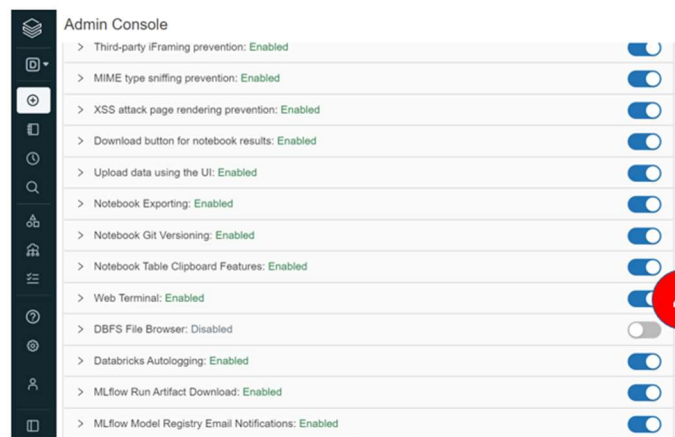
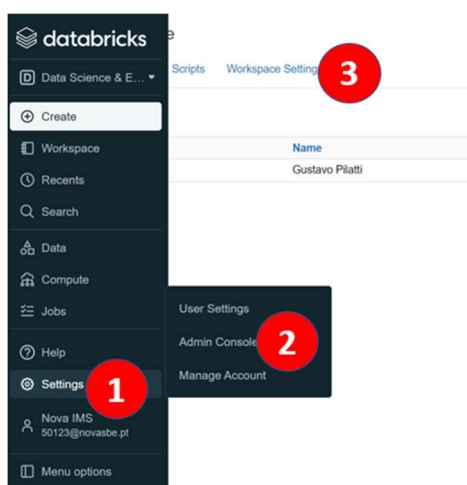
## Big Data Analytics - Lab 2

For the ones running MacOS or any Linux distribution you can run the class on your own computer. For windows users, Databricks runs on Linux, therefore all notebook exercises will work on Databricks.



If the Web Terminal is not enabled, follow these steps:

1. Go to the Settings
2. Click on Admin Console.
3. Click the Workspace Settings tab.
4. In the Advanced section, click the Web Terminal toggle.
5. Refresh the page.



## Big Data Analytics - Lab 2

Note: For the ones running windows it is also possible to run bash in it. Although the process is a bit more complicated but still possible. (<https://docs.microsoft.com/en-us/windows/wsl/install-win10>)

## Big Data Analytics - Lab 2

# Shell commands:

A Shell provides you with an interface to the operating system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

In this course we will use **Bash** ("Bourne Again Shell") shell as the main shell interpreter.

This shell is currently running on your local filesystem. Here is a list of shell commands to get comfortable with:

<b>ls</b>	ls [ <i>options</i> ]	Show directory contents, lists names of files.
<b>mkdir</b>	mkdir [ <i>options</i> ] directory	Creates a new directory of the specified name.
<b>cat</b>	cat [ <i>filename</i> ]	Display file's contents
<b>cd</b>	cd / <i>directorypath</i>	Change directory. Change to a certain directory name if provided.
<b>pwd</b>	pwd [-LP]  By default, 'pwd' behaves as if '-L' were specified.	Displays the name of the working directory.
<b>touch</b>	touch <i>filename</i>	Creates a blank file with a specified name.
<b>less</b>	less [ <i>options</i> ] [ <i>filename</i> ]	View contents of specified file, page by page.
<b>head/tail</b>	tail [ <i>options</i> ] [ <i>filename</i> ]	Displays the first/ last 10 lines of a file.
<b>rm</b>	rm [ <i>options</i> ] <i>directory</i>	Removes a specified file. There is no recycle bin
<b>history</b>	history [ <i>options</i> ]	Display a listing of the last commands you've run.

## Big Data Analytics - Lab 2

<b>cp</b>	<code>cp [options] source destination</code>	Copy specified file to a new named file. Use -r flag copy a directory.
<b>mv</b>	<code>mv [options] source destination</code>	Rename a specified file or directory.
<b>find</b>	<code>find [-H] [-L] [-P] [-Olevel] [-D debugopts] [path...] [expression]</code>	Search files and directories. Can use with wildcards (* ?[ ]).
<b>curl</b>	<code>curl [options] url</code>	Download a webpage
<b>help</b>	<code>help [-dms]</code>	Get help on a command eg. help ls
<b>echo</b>	<code>echo [options]</code>	Prints text to the terminal window
<b>grep</b>	<code>grep [options] pattern [filename]</code>	Used to search text for patterns specified by the user.
<b>uniq</b>	<code>uniq [options] [input [output]]</code>	report or filter out repeated lines in a file
<b>sort</b>	<code>Sort [options] [filename]</code>	sort or merge records (lines) of text and binary files
<b>wc</b>	<code>Wc [options] [filename]</code>	Word count
<b>sed</b>	<code>sed [options]</code>	Pattern-matched string replacement

**TIP:** You can press the up arrow to cycle through previous commands

**TIP:** When using windows, you can right-click to paste (instead of ctrl-v).

**TIP:**

- **[command] -h:** Display a file's help information.
- **[command] --help:** Display a file's help information.
- **whatis [command]:** Display a short blurb about the command.
- **Some commands do not have the --help builtin function**, as echo, so you can enable it by typing: `enable -n echo ; echo --help`

**IMPORTANT:** CTRL-C (cmd-C) will cancel any command running, this will be useful if you accidentally try to open a large file.

## Big Data Analytics - Lab 2

### Let us start:

We start by understanding where we are within the file system and the content of the current directory.

**hostname** – what machine you are on

**whoami** – who you are logged in as

**pwd** - Show the current directory. Folders are divided with slashes “/”.

**ls** - List the contents of the current directory.

Let us try to open a file with cat:

**cat /etc/lsb-release** - In this case we are opening a file called “lsb-release” in the “etc” folder that contains information about the operating system.

Now adding a bit of action, let us get a live stream of information.

**htop** - is an important command to check the status of the tasks running on the computer.

```

1  [] 1.40] Tasks: 37, 240 thr; 2 running
2  [] 2.40] Load average: 0.09 0.14 0.53
Mem[] 3.35G/10.1G] Uptime: 00:27:57
Swap[] 0K/10.00M]

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIDE+ Command
483 root 20 0 3485M 569M 34164 S 2.1 5.5 0:10.34 java -XX:+UseContainerSupport -XX:+PrintFlagsFinal -XX:+PrintGCDetails -XX:+PrintGCDateStamps -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -XX:-OmitStackTraceInFastThrow -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
390 root 20 0 3485M 569M 34164 S 1.4 5.5 0:11.60 java -XX:+UseContainerSupport -XX:+PrintFlagsFinal -XX:+PrintGCDetails -XX:+PrintGCDateStamps -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -XX:-OmitStackTraceInFastThrow -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
510 root 20 0 3485M 569M 34164 R 1.4 25.9 0:11.54 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
3537 root 20 0 4652 5376 1128 R 0.7 0.1 0:00.13 htop
560 root 20 0 3485M 569M 34164 S 0.7 25.9 0:01.55 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
483 root 20 0 3485M 569M 34164 S 0.7 5.5 0:00.50 java -XX:+UseContainerSupport -XX:+PrintFlagsFinal -XX:+PrintGCDetails -XX:+PrintGCDateStamps -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -XX:-OmitStackTraceInFastThrow
461 root 20 0 3485M 569M 34164 S 0.7 5.5 0:00.48 java -XX:+UseContainerSupport -XX:+PrintFlagsFinal -XX:+PrintGCDetails -XX:+PrintGCDateStamps -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -XX:-OmitStackTraceInFastThrow
72 /usr/bin 20 0 219M 4312 3884 S 0.0 0.0 0:00.02 /usr/sbin/rsyslogd -n -DDE
672 root 20 0 3485M 569M 34164 S 0.0 25.9 0:00.13 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
421 root 20 0 3485M 569M 34164 S 0.0 5.5 0:00.89 java -XX:+UseContainerSupport -XX:+PrintFlagsFinal -XX:+PrintGCDetails -XX:+PrintGCDateStamps -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -XX:-OmitStackTraceInFastThrow
490 root 20 0 3485M 569M 34164 S 0.0 25.9 0:02.46 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
644 root 20 0 3485M 569M 34164 S 0.0 25.9 0:00.90 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
336 samila 20 0 349M 11552 4880 S 0.0 0.1 0:15.10 /usr/sbin/gmetad --pid-file /var/run/gmetad.pid
496 root 20 0 3485M 569M 34164 S 0.0 25.9 0:33.77 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
614 root 20 0 3485M 569M 34164 S 0.0 25.9 0:02.55 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
323 samila 20 0 9088M 8232 3192 S 0.0 0.1 0:02.52 /usr/sbin/gmond --pid-file /var/run/gmond.pid
555 root 20 0 3485M 569M 34164 S 0.0 25.9 0:00.00 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
565 root 20 0 3485M 569M 34164 S 0.0 25.9 0:00.14 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
639 root 20 0 3485M 569M 34164 S 0.0 25.9 0:01.07 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
642 root 20 0 3485M 569M 34164 S 0.0 25.9 0:00.89 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
1 root 20 0 169M 30764 3304 S 0.0 0.1 0:11.18 /sbin/init
82 root 20 0 8824 2872 2344 S 0.0 0.0 0:00.15 /usr/bin/monit -c /etc/monit/monitrc
312 root 20 0 3485M 569M 34164 S 0.0 5.5 0:05.69 java -XX:+UseContainerSupport -XX:+PrintFlagsFinal -XX:+PrintGCDetails -XX:+PrintGCDateStamps -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -XX:-OmitStackTraceInFastThrow
462 root 20 0 3485M 569M 34164 S 0.0 5.5 0:00.45 java -XX:+UseContainerSupport -XX:+PrintFlagsFinal -XX:+PrintGCDetails -XX:+PrintGCDateStamps -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -XX:-OmitStackTraceInFastThrow
643 root 20 0 3485M 569M 34164 S 0.0 25.9 0:00.02 /usr/lib/jvm/zulu8-ca-amd64/jre/bin/java -Djava.io.tmpdir=/local_disk0/tmp -XX:MaxPermSize=512m -XX:OmitStackTraceInFastThrow
487 root 20 0 3485M 569M 34164 S 0.0 5.5 0:00.24 java -XX:+UseContainerSupport -XX:+PrintFlagsFinal -XX:+PrintGCDetails -XX:+PrintGCDateStamps -verbose:gc -XX:+HeapDumpOnOutOfMemoryError -XX:-OmitStackTraceInFastThrow

```

```

1 %bash
2 htop

0: PID USER PRI NI VIRT RES SHR S CPU% MEM% TIDE+ Command 1674 root 20 0 10.5G 2975M 60088 S 50.0 28.8 0:00.22 /usr/lib/jvm/zulu1656 root20 0 703M 84160 22628 R 50.0 0.8 0:01.74 /local_disk0/pyth1696 root20 0 9352 4968 3140 R 50.0 0.0 0:00.04 htop1 root20 0 100M 10764 8304 S 0.0 0.1 0:12.11 /sbin/init40 root19 -1 35080 11656 10760 S 0.0 0.1 0:00.28 /lib/systemd/syst49 systemd -r 20 0 23964 12740 8624 S 0.0 0.1 0:00.05 /lib/systemd/syst52 messagebu 20 0 7412 4220 3776 S 0.0 0.0 0:00.04 /usr/bin/obus-dae5 5 root20 0 29224 17752 10124 S 0.0 0.2 0:00.11 /usr/bin/python368 syslog 20 0 219M 4312 3884 S 0.0 0.0 0:00.01 /usr/sbin/rsyslog69 syslog 20 0 219M 4312 3884 S 0.0 0.0 0:00.02 /usr/sb1n/rsyslog57 syslog 20 0 219M 4312 3884 S 0.0 0.0 0:00.07 /usr/sbin/rsyslog62 root20 0 16480 6416 5696 S 0.0 0.1 0:00.06 /lib/sytemd/syst74 root20 0 2632 1756 1644 S 0.0 0.0 0:00.00 /sbin/agetty -o -93 root20 0 74396 3276 2780 S 0.0 0.0 0:00.00 /usr/sbin/n tpd -p76 root20 0 74396 3276 2780 S 0.0 0.0 0:00.14 /usr/sbin/ntpd -pF1
Help F2 Setup F3 Search F4 Filter F5 Tree F6 Sortby F7 Nice F8 Nice F9 Kill F10 Quit
Command took 1.35 seconds -- by 50123@novasbe.pt at 06/02/2022 10:14:38 on New cluster

```

## Big Data Analytics - Lab 2

As a way to chain operations bash uses the concept operators. They are used to combine several operations together; this is where the power of bash comes from. By chaining the simple commands shown in the previous section it allows for more advanced and useful operations.

### BASH operators:

#### List terminators

- “;”: Will run one command after another has finished, irrespective of the outcome of the first. E.g.: `command1 ; command2` First `command1` is run, in the foreground, and once it has finished, `command2` will be run.

#### Pipe operator

- “|”: The pipe operator, it passes the output of one command as input to another. A command built from the pipe operator is called a pipeline. E.g.: `command1 | command2` Any output printed by `command1` is passed as input to `command2`.

#### Redirection operators

These allow you to control the input and output of your commands. They can appear anywhere within a simple command or may follow a command. Redirections are processed in the order they appear, from left to right.

- “<”: Gives input to a command. `command < file.txt` will execute `command` on the contents of “file.txt”.
- “>”: Directs the output of a command into a file. `command > out.txt` will save the output of `command` as “out.txt”. If the file exists, its contents will be overwritten and if it does not exist it will be created.
- “>>”: Does the same as “>”, except that if the target file exists, the new data are appended. `command >> out.txt` If “out.txt” exists, the output of `command` will be appended to it, after whatever is already in it. If it does not exist it will be created.

#### Multi-line execution

Long commands can be separated into multiple lines by using a backslash after each line

```
January February March April \
May June July August September October November \
December
```

## Big Data Analytics - Lab 2

### Quotes

Text enclosed in quotes is considered literal value. Example:

```
root@0206-124507-fis7e8yb-10-172-224-5:/databricks/driver# mkdir 'Documents and Settings'
root@0206-124507-fis7e8yb-10-172-224-5:/databricks/driver# ls
'Documents and Settings'  conf  eventlogs  ganglia  logs  metastore_db  preload_class.lst
root@0206-124507-fis7e8yb-10-172-224-5:/databricks/driver# mkdir Documents and Settings
root@0206-124507-fis7e8yb-10-172-224-5:/databricks/driver# ls
Documents  'Documents and Settings'  Settings  and  conf  eventlogs  ganglia  logs  metastore_db  preload_class.lst
root@0206-124507-fis7e8yb-10-172-224-5:/databricks/driver#
```

**Tip:** attention to ``rm -r -f *``. For more information, follow [this link](#). **\*\*Never run it\*\*\***

## Example

Let's create a file and deal with it!

1. Create a file with the name "myfile.txt" with the words "banana apple carrot" as the file content

```
echo "banana apple carrot" > myfile.txt
```

2. Verify if your file was created

```
ls
```

3. All entries are in the same line. Let's delete this file

```
rm myfile.txt
```

4. We need to create a file with one entry per line. Call this file 'data.txt'

```
echo "banana" > data.txt
```

5. Append more lines with new entries

```
echo "apple" >> data.txt | echo "carrot" >> data.txt | echo "water-melon" >> data.txt
```

6. Search for words with 'a'

```
grep 'a' data.txt
```

7. Search for words with 'p' and 'c'

```
grep '[pc]' data.txt
```

**Tip:** square parentheses are a matching set.



## Big Data Analytics - Lab 2

# Exercises

Find today's exercise in Moodle directory

High quality guide

<http://www.compciv.org/bash-guide/>