

Big Data Architectures For Machine Learning and Data Mining

Winter Semester 2019

Web Technology and Information Systems Group

Michael Voelske

`<firstname>.<lastname>@uni-weimar.de`

Big Data Architectures for Machine Learning and Data Mining

Seminar Organization

Session 1 (today):

- ❑ The What, Why & How of Big Data
- ❑ Virtual Cluster Environment
- ❑ Crash Course: Linux

Session 2 (April 29th):

- ❑ Crash Course (cont'd)
- ❑ A Complex Example: Hadoop Deployment and Usage

Session 3 (May 6th):

- ❑ First presenters can discuss their slides & any problems

Session 4+ (from May 13th):

- ❑ Student Talks

Big Data Architectures for Machine Learning and Data Mining

Seminar Deliverables

You can get 2 ETCS for this course. In order to obtain a grade, you'll have to:

1. **Register.** Fill out the paper form, or (if you miss the first session) send an email with your Name and matriculation number to michael.voelske@uni-weimar.de [Deadline: April 24th]
2. **Give a presentation.** You will have up to 30 minutes to present a particular big data technology. Your presentation should include installation and configuration instructions, concrete usage examples, and necessary theoretical background. Topics will be assigned at the next session.
3. **Submit your code.** Installation script and (simple) usage demo.
4. **Participate** in discussion after other students' talks.



Big Data.

What is “Big Data”

datascience@berkeley asked more than 40 experts in 2014...

“Big data” is data that can’t be processed using standard databases because it is **too big, too fast-moving, or too complex** for traditional data processing tools.

AnnaLee Saxenian (Dean, UC Berkeley School of Information)

Big data is when data grows to the point that the technology supporting the data has to change. It also encompasses a variety of topics relating to **how disparate data can be combined**, processed into insights, and/or reworked into smart products.

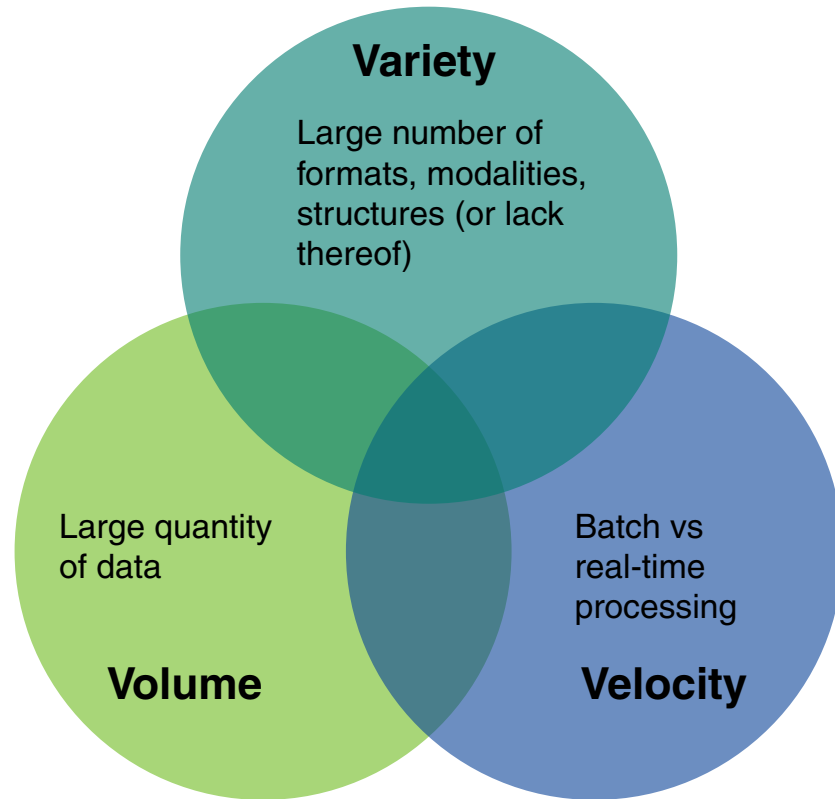
Anna Smith (Analytics Engineer, Rent the Runway)

In my view, big data is data that requires novel processing techniques to handle. Typically, **big data requires massive parallelism** in some fashion (storage and/or compute) to deal with volume and processing variety.

Brad Peters (Chief Product Officer, Birst)

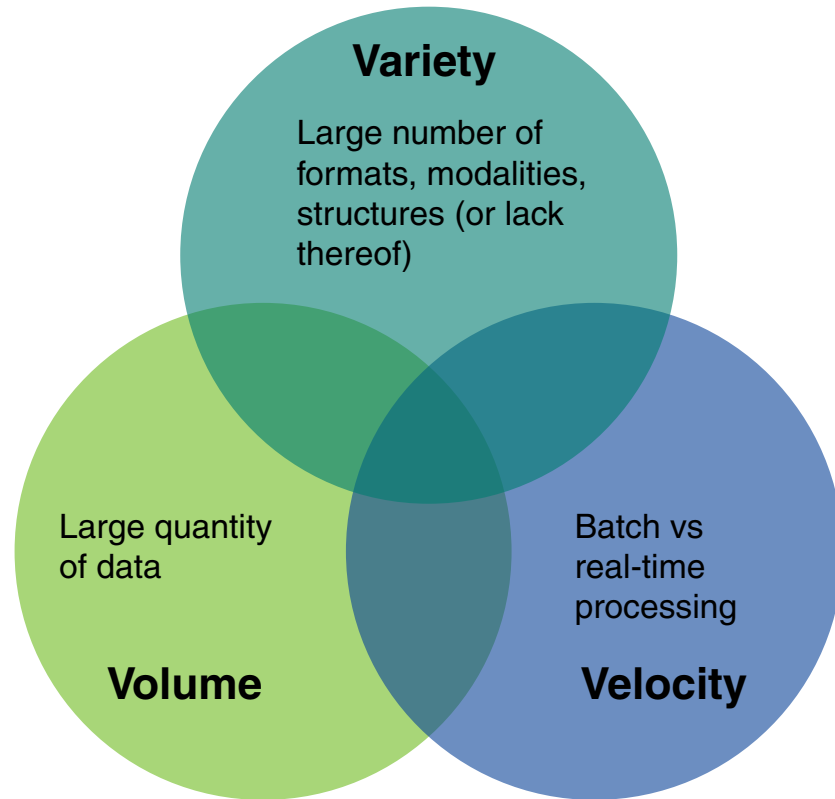
What is “Big Data”

“Three V’s” (and beyond) to characterize big data problems.

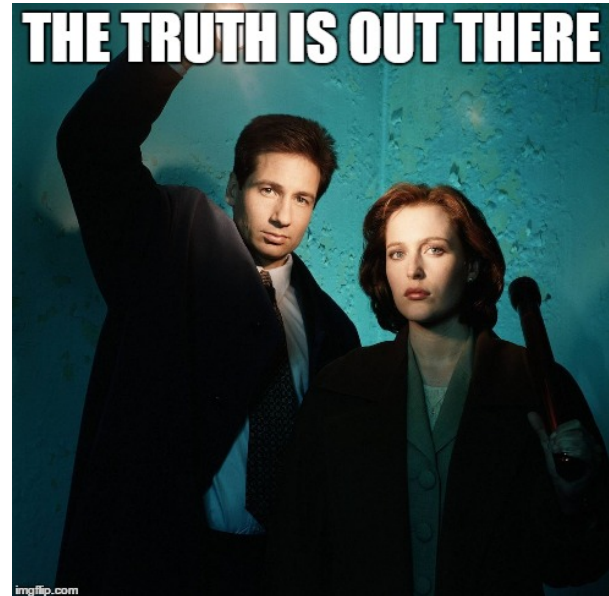


What is “Big Data”

“Three V’s” (and beyond) to characterize big data problems.



Increasingly important: **Veracity**, data quality, provenance.



The Big Picture: Big Data Architecture Stack

Data
Consumption
Layer

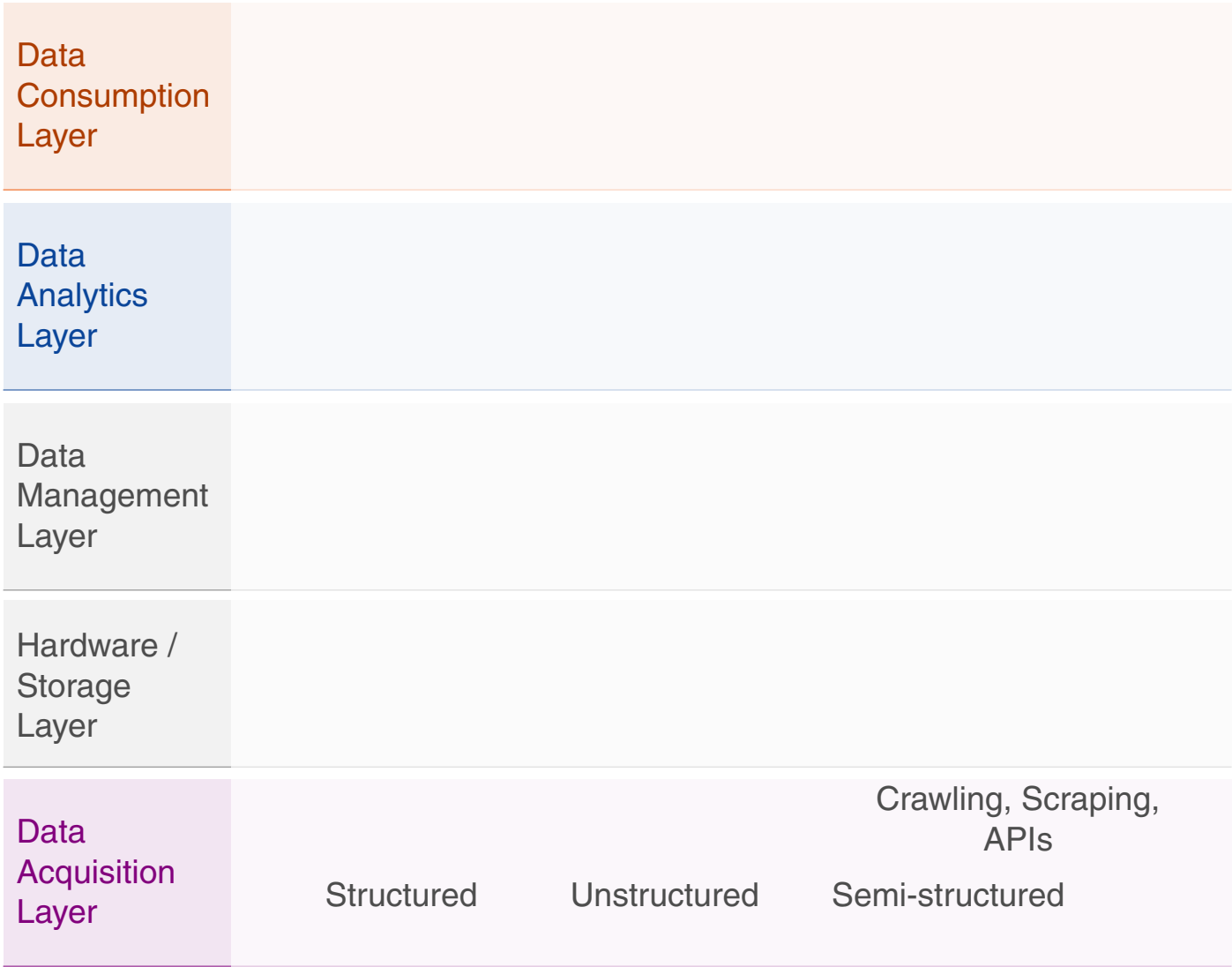
Data
Analytics
Layer

Data
Management
Layer

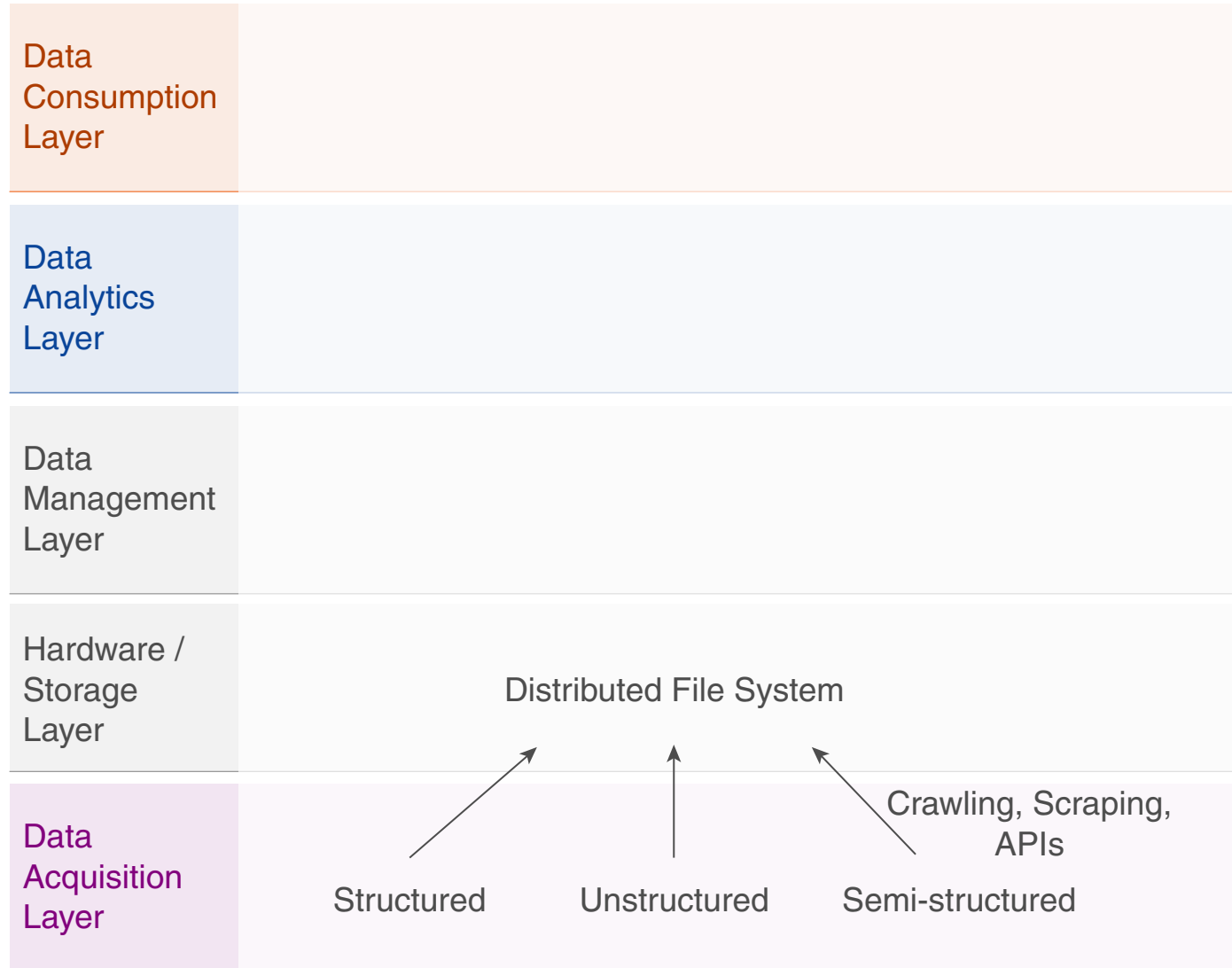
Hardware /
Storage
Layer

Data
Acquisition
Layer

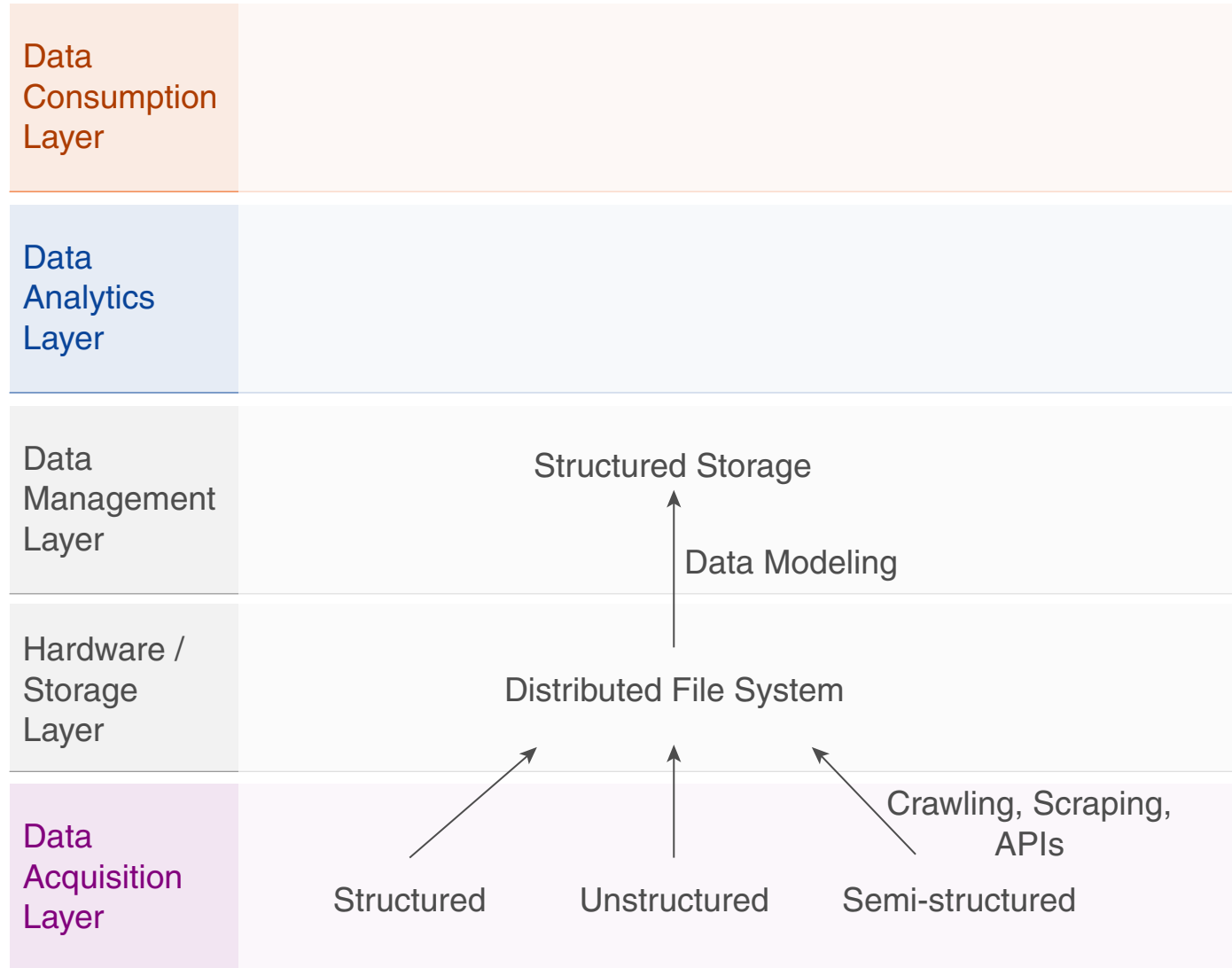
The Big Picture: Big Data Architecture Stack



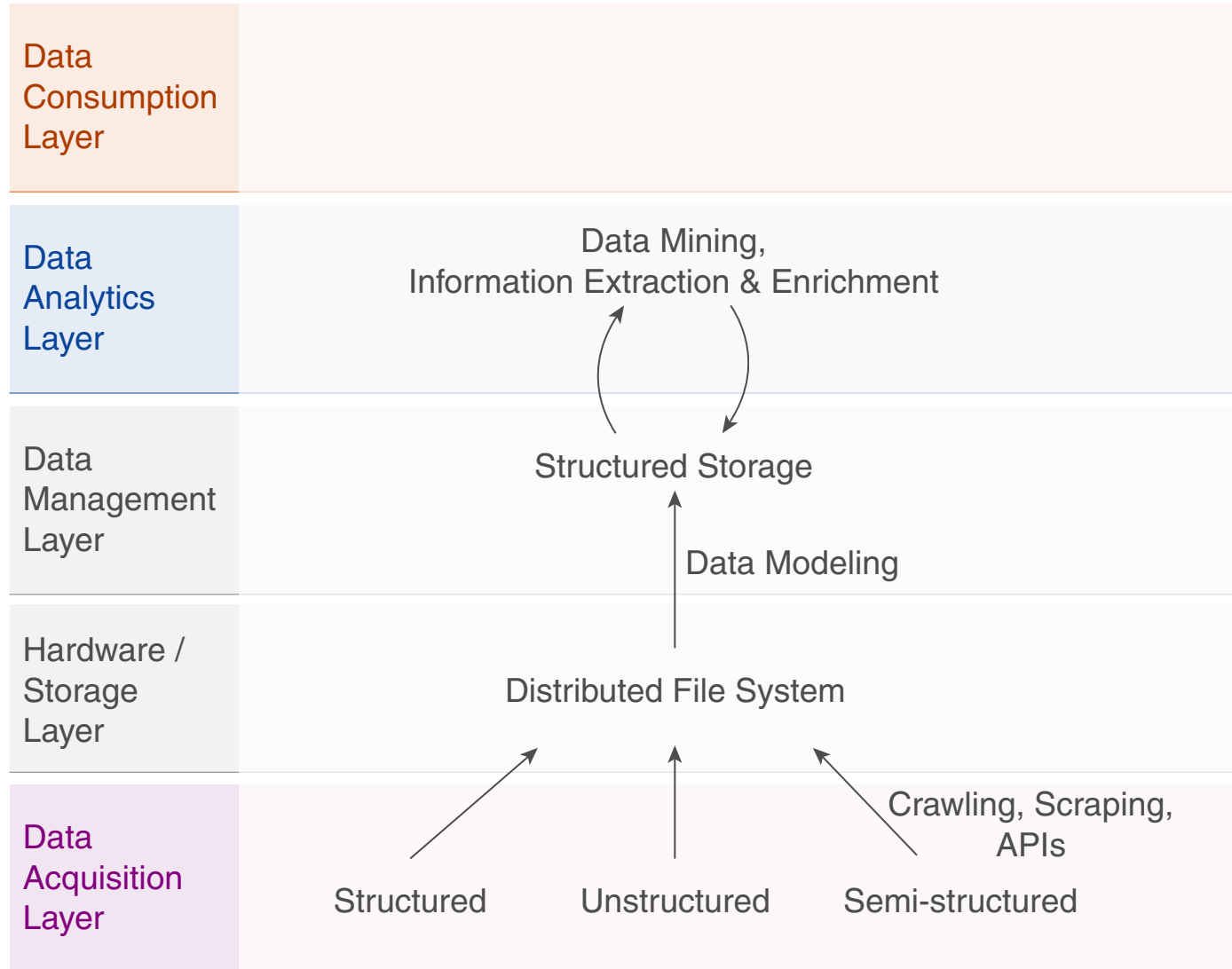
The Big Picture: Big Data Architecture Stack



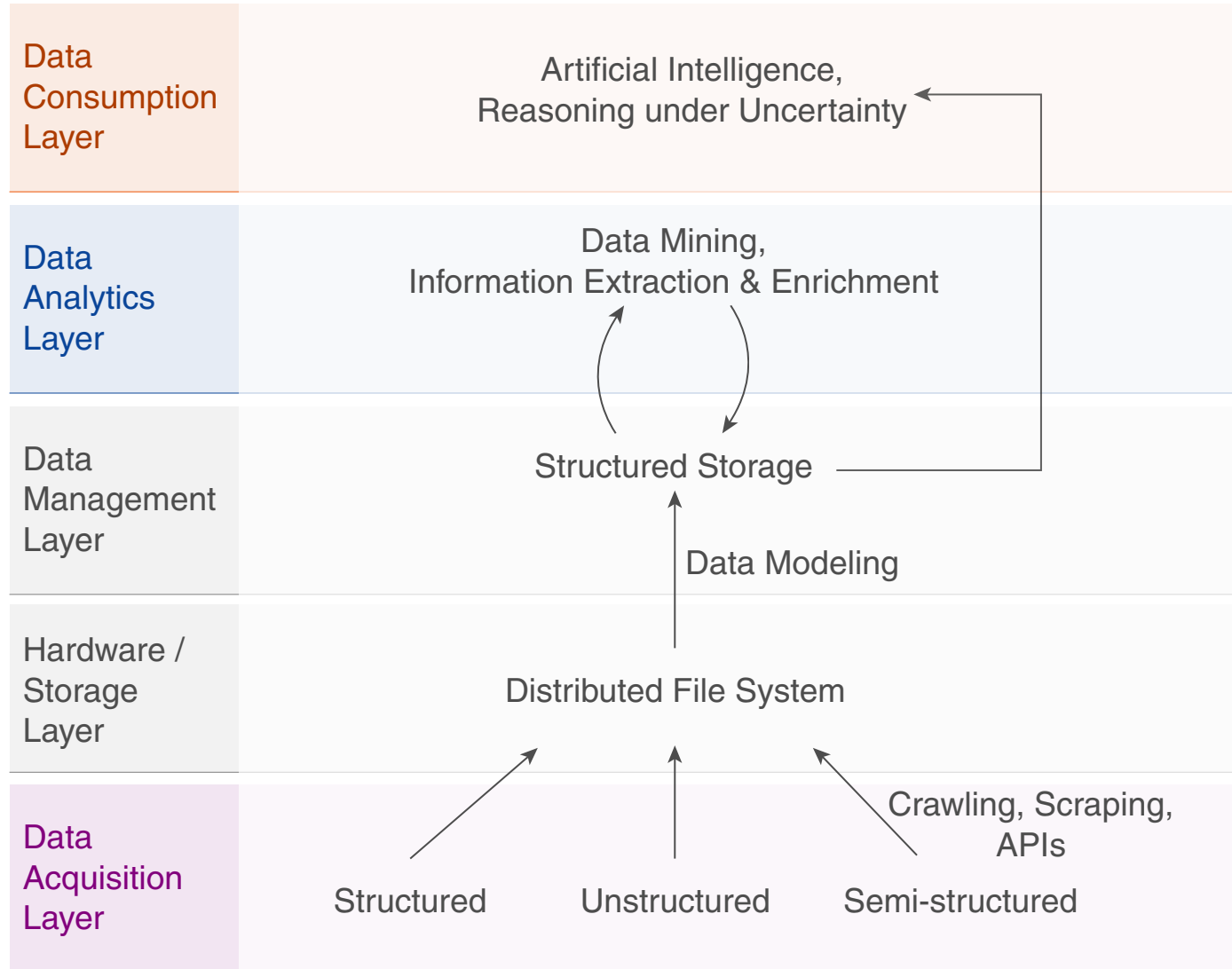
The Big Picture: Big Data Architecture Stack



The Big Picture: Big Data Architecture Stack



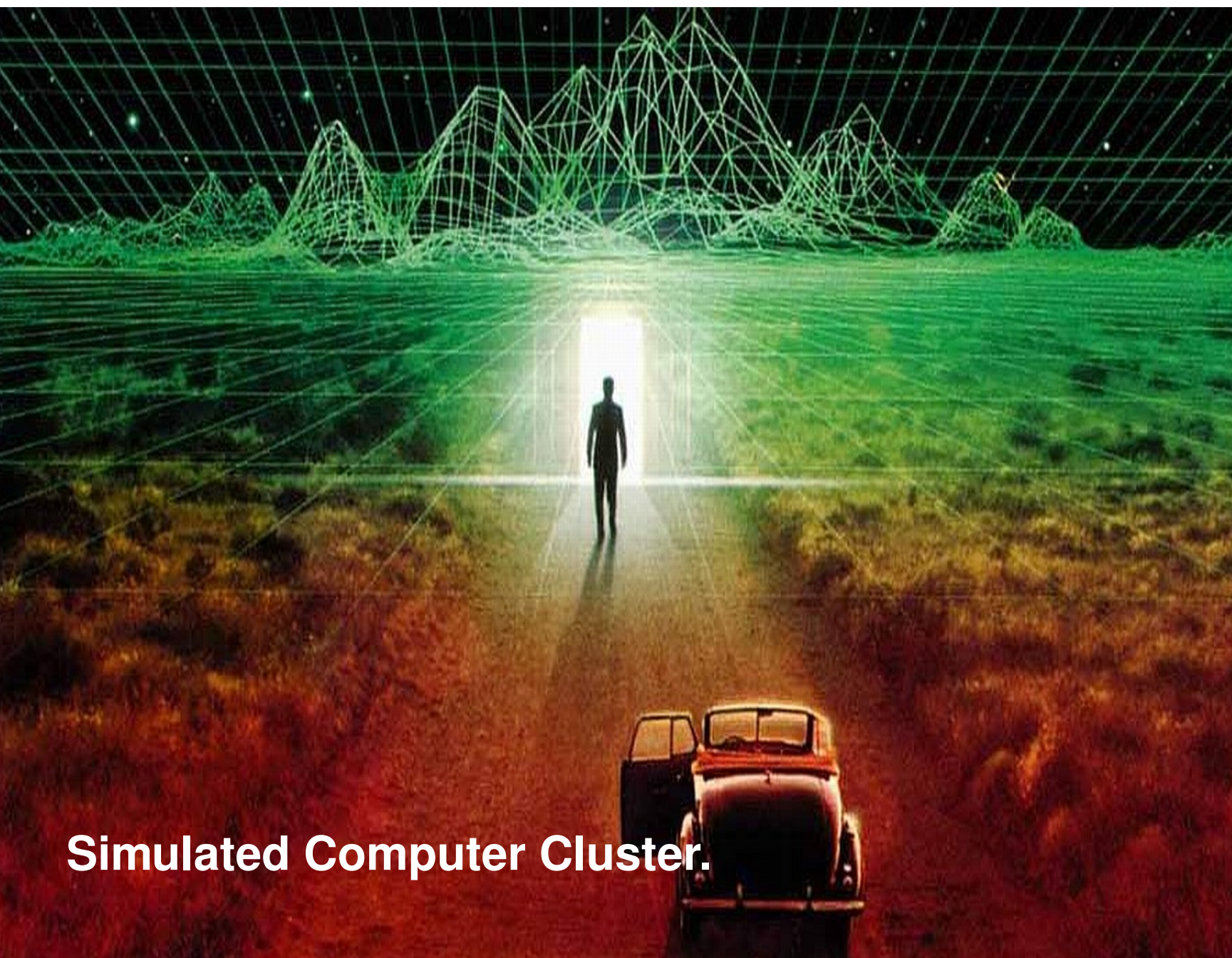
The Big Picture: Big Data Architecture Stack



14

© Matt Turck (@mattturck), Demi Obayomi (@demi_obayomi), & FirstMark (@firstmarkcap)





Simulated Computer Cluster.

A Virtual Cluster Environment


Components


Big data architectures are typically deployed on large computer clusters. A cluster comprises an arbitrary number of *nodes* connected through a local area network.





Ten of 135 nodes from the webis betaweb cluster.

We will learn the basics of cluster computing using a virtual cluster environment built with the help of the following software:

 **VirtualBox.** Simulates a Virtual Machine (VM) with its own operating system (in our case, Linux).

 **Vagrant.** Sets up (“provisions”) multiple VMs and organizes them into a virtual cluster.

 **Git.** Provides version control (& shell for Windows users; Mac/Linux users already have one).

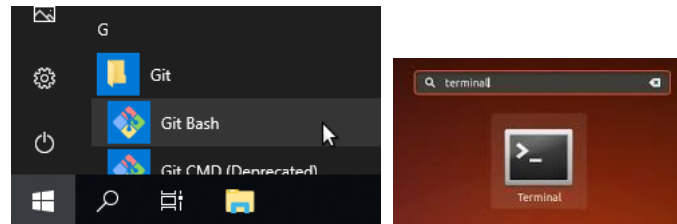
 **FoxyProxy.** Connects your web browser to the virtual cluster network.

A Virtual Cluster Environment

Setting Up

Open a terminal

(On Windows, use “Git Bash”).



And type the following

(replace `your_workspace_directory` with a path that makes sense to you):

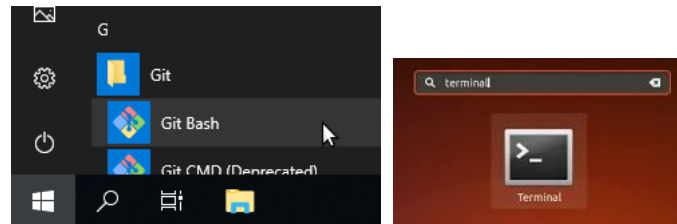
```
> cd your_workspace_directory
git clone https://github.com/mvoelske/bigdata-seminar.git
cd bigdata-seminar
```

(Feel free to fork the repo first, and use your fork to keep track of your seminar work)

A Virtual Cluster Environment

Setting Up

Open a terminal
(On Windows, use “Git Bash”).



And type the following
(replace `your_workspace_directory` with a path that makes sense to you):

```
> cd your_workspace_directory
git clone https://github.com/mvoelske/bigdata-seminar.git
cd bigdata-seminar
```

(Feel free to fork the repo first, and use your fork to keep track of your seminar work)

Now, we can create the VM environment:

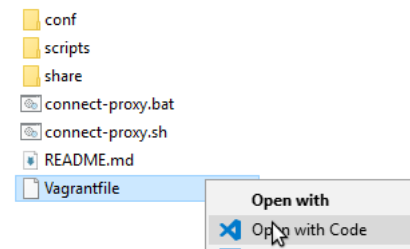
```
> vagrant up
```



This can take some time. So, while we wait, let's take a closer look at what we “`git clone`”d.

A Virtual Cluster Environment

Exploring the Repository



The `Vagrantfile` describes the VMs in our virtual cluster, and how they are networked together. The syntax is Ruby, but you don't really need to know Ruby to work with the provided setup—follow the comments at the start of the file.

Useful to know:

- ❑ Your virtual cluster starts with one node, but you can increase this to as many as your computer can handle.
- ❑ Host names: `node0`, `node1`, ...
- ❑ IP addresses `10.40.23.100`, `10.40.23.101`, ...
- ❑ The folder `share` and its contents are available on all nodes (any files you create there will be immediately visible in the VMs)

A Virtual Cluster Environment

We've already run a vagrant command to bring up the virtual cluster. Some further commands are given in the comments in the `Vagrantfile`, and the rest are documented online: [\[www.vagrantup.com/docs/cli\]](http://www.vagrantup.com/docs/cli)

For example, if you later break something inside the VMs (you should, that's how you learn) and want to re-start from scratch, run:

```
 vagrant destroy --parallel -f && vagrant up
```

A Virtual Cluster Environment

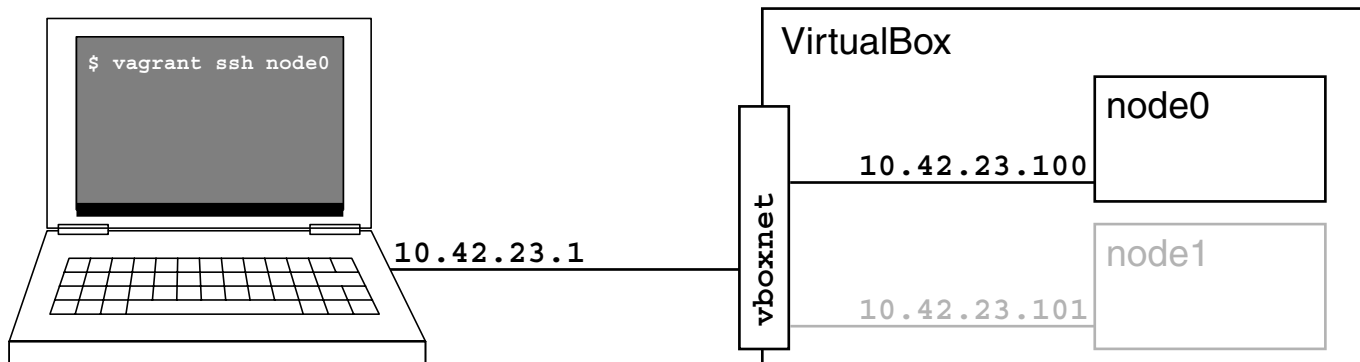
We've already run a vagrant command to bring up the virtual cluster. Some further commands are given in the comments in the `Vagrantfile`, and the rest are documented online: [\[www.vagrantup.com/docs/cli\]](http://www.vagrantup.com/docs/cli)

For example, if you later break something inside the VMs (you should, that's how you learn) and want to re-start from scratch, run:

```
> vagrant destroy --parallel -f && vagrant up
```

But first, we have to get in:

```
> vagrant ssh node0
```



A Virtual Cluster Environment

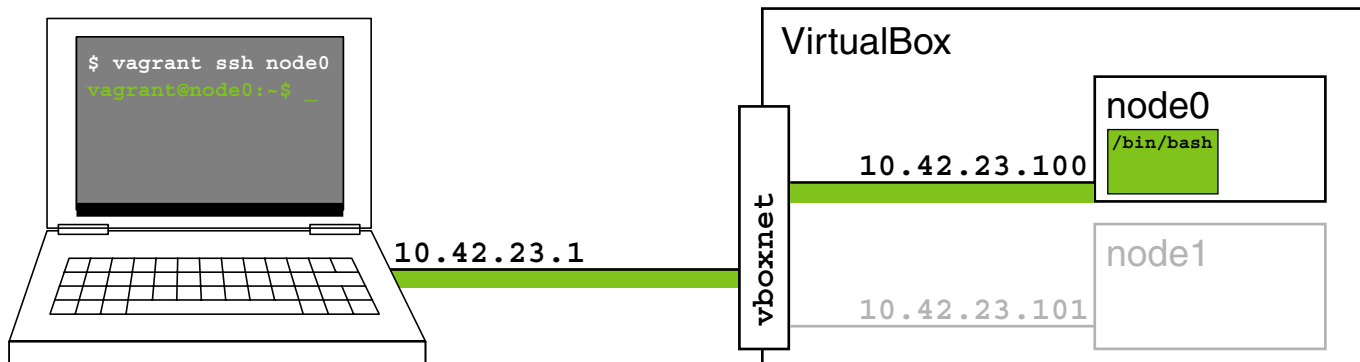
We've already run a `vagrant` command to bring up the virtual cluster. Some further commands are given in the comments in the `Vagrantfile`, and the rest are documented online: [\[www.vagrantup.com/docs/cli\]](http://www.vagrantup.com/docs/cli)

For example, if you later break something inside the VMs (you should, that's how you learn) and want to re-start from scratch, run:

```
> vagrant destroy --parallel -f && vagrant up
```

But first, we have to get in:

```
> vagrant ssh node0
```



A Virtual Cluster Environment

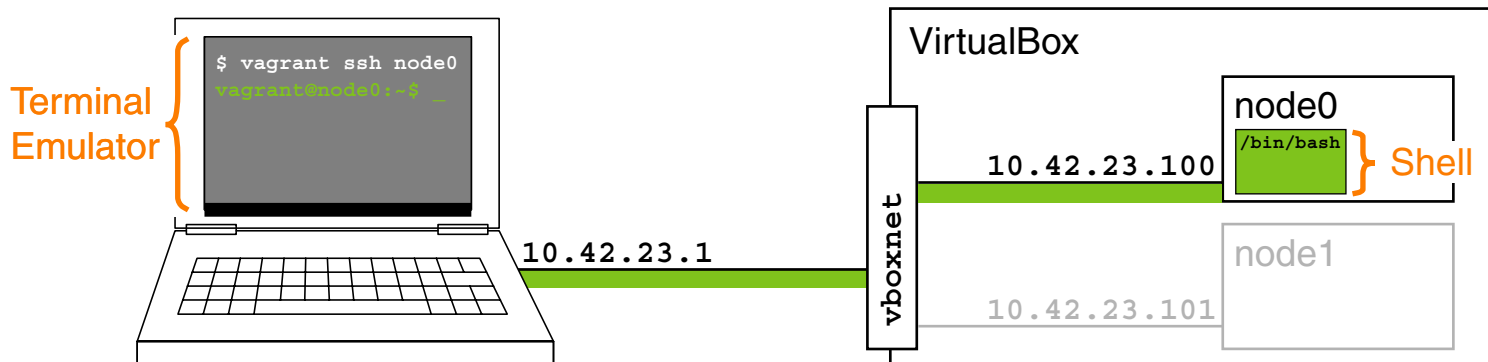
We've already run a vagrant command to bring up the virtual cluster. Some further commands are given in the comments in the `Vagrantfile`, and the rest are documented online: [\[www.vagrantup.com/docs/cli\]](http://www.vagrantup.com/docs/cli)

For example, if you later break something inside the VMs (you should, that's how you learn) and want to re-start from scratch, run:

```
> vagrant destroy --parallel -f && vagrant up
```

But first, we have to get in:

```
> vagrant ssh node0
```



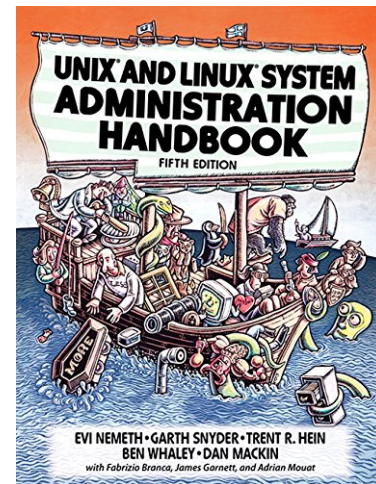
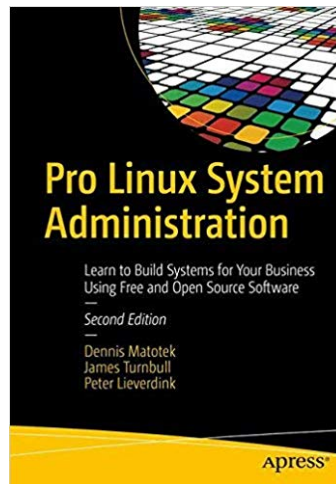
A Brief Tour of the Linux Commandline.



Linux Commandline Crash Course

A Word of Caution

The following is a very brief tour of working with the Linux command line. It leaves out relevant details. If you remain confused, please consult additional material.



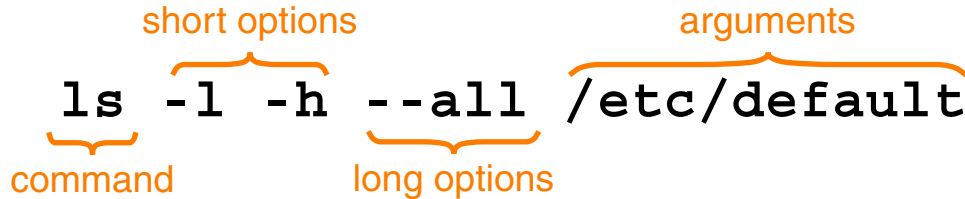
[\[linuxcommand.org\]](http://linuxcommand.org) [\[ryanstutorials.net/linuxtutorial/\]](http://ryanstutorials.net/linuxtutorial/) [\[edx.org/course/introduction-to-linux\]](http://edx.org/course/introduction-to-linux)

[\[linux.die.net/Bash-Beginners-Guide\]](http://linux.die.net/Bash-Beginners-Guide) [\[linux.die.net/abs-guide\]](http://linux.die.net/abs-guide)

Linux Commandline Crash Course

The Most Important Things to Know

What a shell command looks like:

 [\[explainshell.com\]](https://explainshell.com)

Arguments are whitespace separated. Quote arguments to include spaces or special characters.

How to get help:



```
ls --help
```



```
Usage: ls [OPTION]... [FILE]...  
List information about the FILES (the current directory by default)  
(lengthy explanation of options follows ...)
```

Or:



```
man ls
```

To show the **manual page** for some command. [\[linuxcommand.org/lc3_lts0060.php\]](https://linuxcommand.org/lc3_lts0060.php)

Linux Commandline Crash Course

Some Simple Basics

Where am I? (“Print Working Directory”)



```
pwd
```



```
/home/vagrant
```

What’s here? (“List”) [linuxcommand.org/lc3_lts0030.php]



```
ls
```

/ is the top-level directory. Below that, a set of standard directories exist:



```
ls /
```



```
bin          etc  lib  mnt  root  srv  vagrant
boot         home lib64 run  sys  var  initrd.img
lost+found  opt  sbin tmp  vmlinuz
...
```

The “Filesystem Hierarchy Standard” specifies these.

[refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html] [linuxcommand.org/lc3_lts0040.php]

Linux Commandline Crash Course

Moving Around



```
cd /usr/bin  
pwd
```



```
/usr/bin
```

Linux Commandline Crash Course

Moving Around



```
cd /usr/bin  
pwd
```



```
/usr/bin
```



```
cd  
pwd
```



```
/home/vagrant
```

[http://linuxcommand.org/lc3_lts0020.php]

Linux Commandline Crash Course

Moving Around



```
cd /usr/bin  
pwd
```



```
/usr/bin
```



```
cd  
pwd
```



```
/home/vagrant
```

[http://linuxcommand.org/lc3_lts0020.php]

Users and Permissions



```
ls -l
```



```
drwxr-xr-x 1 vagrant vagrant 4096 Apr 12 11:31 share
```

Permissions, left to right: is directory. Owner **vagrant** can read, write, execute. Group **vagrant** can read and execute. All other users can read and execute. [linuxcommand.org/lc3_lts0090.php]

Linux Commandline Crash Course

Manipulating Files

Create an empty file:



```
touch my-test-file
```

Create a directory:



```
mkdir my-test-directory
```

Move the file:



```
mv my-test-file my-test-directory
```

Delete everything:



```
rm -r my-test-directory
```

[\[linuxcommand.org/lc3_lts0050.php\]](http://linuxcommand.org/lc3_lts0050.php)

Linux Commandline Crash Course

Basic Text Processing

In the following, we'll work with the `/etc/passwd` file, which contains some basic information about all users.

See the first n lines:



```
head -n 2 /etc/passwd
```



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
```

Extract columns:



```
cut -d : -f 1,3 /etc/passwd
```

Extract lines matching a regular expression: www.debuggex.com/cheatsheet/regex/pcr



```
grep 'sys.*-net' /etc/passwd
```

Get file contents in sorted order:




```
sort /etc/passwd
```


Linux Commandline Crash Course

Redirection and Pipes

Write the output of a command to a file:

```
 head -n 2 /etc/passwd > my-file.txt
```

Redirect different streams to different files

```
 grep root /etc/passwd /does/not/exist  
grep root /etc/passwd /does/not/exist > found.txt 2> not-found.txt
```

Use the output of one command as input for another command:

```
 head -n 3 /etc/passwd | wc -l
```


[linuxcommand.org/lc3_lts0070.php]

Linux Commandline Crash Course

Basic Package Management

We use the “Ubuntu” Linux distribution, which has the `apt` suite of tools for searching and installing new software.

To search by name or description:




```
apt-cache search 'ASCII banner'
```



```
figlet - Make large character ASCII banners out of ordinary text
```

To install a package (note the `sudo`):



```
sudo apt-get install -y figlet
```


[\[help.ubuntu.com/lts/serverguide/package-management.html.en\]](http://help.ubuntu.com/lts/serverguide/package-management.html.en)

Linux Commandline Crash Course

Basic Package Management

We use the “Ubuntu” Linux distribution, which has the `apt` suite of tools for searching and installing new software.

To search by name or description:




```
apt-cache search 'ASCII banner'
```



```
figlet - Make large character ASCII banners out of ordinary text
```


To install a package (note the `sudo`):



```
sudo apt-get install -y figlet
```

[\[help.ubuntu.com/lts/serverguide/package-management.html.en\]](https://help.ubuntu.com/lts/serverguide/package-management.html.en)

Let's also install these:



```
sudo apt-get install -y pv jq csvtool
```

Linux Commandline Crash Course

Slightly More Advanced Text Processing

Let's put what we already know together. Here's a big text file:



```
cd /share/example/data
wc shakespeare.txt
less shakespeare.txt
```

Less is a *pager*, which helps you read text files.

How many lines contain the word “thou?”



```
pv shakespeare.txt | grep thou | wc -l
```

pv (Pipe Viewer) works just like `cat`, but gives you a progress bar.

Linux Commandline Crash Course

Slightly More Advanced Text Processing

Let's put what we already know together. Here's a big text file:



```
cd /share/example/data
wc shakespeare.txt
less shakespeare.txt
```

Less is a *pager*, which helps you read text files.

How many lines contain the word “thou?”



```
pv shakespeare.txt | grep thou | wc -l
```

pv (Pipe Viewer) works just like `cat`, but gives you a progress bar.

More complex example: what are the five most frequent words, and how often do they occur?



```
pv shakespeare.txt | sed 's/\s\+/\n/g' | sort | uniq -c | \
    sort -k1nr | head -n 5
```

(Look at stages of the pipeline in isolation, and refer to `man sed`, `man uniq` and `man sort` to understand what happens here)

Linux Commandline Crash Course

Slightly More Advanced Text Processing (continued)

The `jq` tool is very useful for processing JSON data. Without any arguments, it works as a pretty-printer:



```
cd /share/example/data  
less worldcup.json  
jq ' ' worldcup.json | less
```

Various JSON processing operations are possible with a *filter* argument. The output of one filter can be passed to another, also with the `|` symbol.



```
jq 'keys' worldcup.json  
jq '.rounds | length' worldcup.json
```

Linux Commandline Crash Course

Slightly More Advanced Text Processing (continued)

The `jq` tool is very useful for processing JSON data. Without any arguments, it works as a pretty-printer:



```
cd /share/example/data
less worldcup.json
jq ' ' worldcup.json | less
```

Various JSON processing operations are possible with a *filter* argument. The output of one filter can be passed to another, also with the `|` symbol.



```
jq 'keys' worldcup.json
jq '.rounds | length' worldcup.json
```

More complex example—find the number of goals per player across all games:



```
jq -r '.rounds | .[].matches | .[] | .goals1,.goals2 | .[].name' \
worldcup.json | sort | uniq -c
```


Refer to the `jq` man page or [\[stedolan.github.io/jq/\]](https://stedolan.github.io/jq/) to understand how this works, and to find out what else this tool can do.

Linux Commandline Crash Course

Slightly More Advanced Text Processing (continued)


We've also installed `csvtool`, which makes working with Comma-Separated Values easier.

For example, we can extract columns by name (if the file has a header row):

```
 csvtool namedcol Sector sp500-companies.csv
```

Question: Why can't we simply use `cut` to get columns from CSV files?

And use this to find out how often different industrial sectors occur among S&P 500 companies:

```
 csvtool namedcol Sector sp500-companies.csv | sort | uniq -c
```

Run `csvtool -help` to find out what else it can do.

Linux Commandline Crash Course

Next time...

Homework: Go through the preceding slides. Read the man pages of the mentioned commands. Break things. Then, do this:

Next week, we'll look at networking multiple virtual machines together. In preparation, edit the `Vagrantfile` to bring up three nodes instead of just one:

```
# The number of nodes in the cluster, by default 3.
# creating the VMs, the number of nodes is used.
# to manually run the destination command, use the number.
# reducing the number.
$num_nodes = 3
```

After this change, bring up the new nodes:



```
vagrant up
```

You now have a three node cluster.

That's all for today.

HACKERMAN