Big Data Technologies

General Introduction

Lionel Fillatre

Polytech Nice Sophia

lionel.fillatre@univ-cotedazur.fr

Class website

- Class website

https://lms.univ-cotedazur.fr/2024/course/view.php?id=1849

- Password for the course registration: dhP57M&up

- **You must register yourself!**

Outlines

- Course Logistics
- Basic Definitions
- Big Data Landscape
- Big Data Framework
- Big Data with a Cluster
- Introduction to Docker
- Install Docker
- Conclusion

Course Logistics

Timeline of this course

1. 16/09/2024: General Introduction + Introduction to Docker and Scala
2. 23/09/2024: Hadoop Distributed File System
3. 30/09/2024: MapReduce
4. 07/10/2024: Spark + 1st scored lab
5. 14/10/2024: NoSQL Databases
6. 21/10/2024: SQL in Big Data
7. 04/11/2024: Advanced Analytics + 2nd scored lab
8. 18/11/2024: 1 hour written exam (8h-9h)

Grading

- Week 4: 1st scored lab (25% of the final grade)
- Week 7: 2nd scored lab (25% of the final grade)
- Week 8: final written exam (50% of the final grade)

- **Scored labs**
  - Work in pairs (or solo if the number of students is not even)
  - You need a personal laptop with Docker and the dedicated container installed
  - Results are important (of course!) but writing clear code comments is the key to get a good score!
  - Lab 3 is useful for scored lab 4
  - Lab 6 is useful for scored lab 7
  - Participation in labs might be considered for scoring

- **Final exam (no document)**
  - The final exam concerns all the course topics
  - Some technical questions on the course and some general questions (architecture, NoSQL, etc.)
  - Writing code is possible but very limited

Classrooms for labs

- E106 (in French)
  - Teacher: Antonin Bavoil
  - Students: MAM5

- B218 (in English)
  - Teacher: Antonio SOGGIA
  - Students: M2 IM + EIT Digital M1 + EIT Digital M2

- Classrooms may change: https://sco.polytech.unice.fr/

Basic Definitions

Big Data Definition

- No single standard definition!

- "**Big Data**" is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it.

The 10 Vs of Big Data

https://tdwi.org/articles/2017/02/08/10-vs-of-big-data.aspx

Who's Generating Big Data

Social media and networks
(all of us are generating data)

Scientific instruments
(collecting all sorts of data)

Mobile devices
(tracking all objects all the time)

Sensor technology and networks
(measuring all kinds of data)

- The progress and innovation is no longer hindered by the ability to collect data
- But, by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion

The Model Has Changed…

- **The Model of Generating/Consuming Data has Changed**

Old Model: Few companies are generating data, all others are consuming data

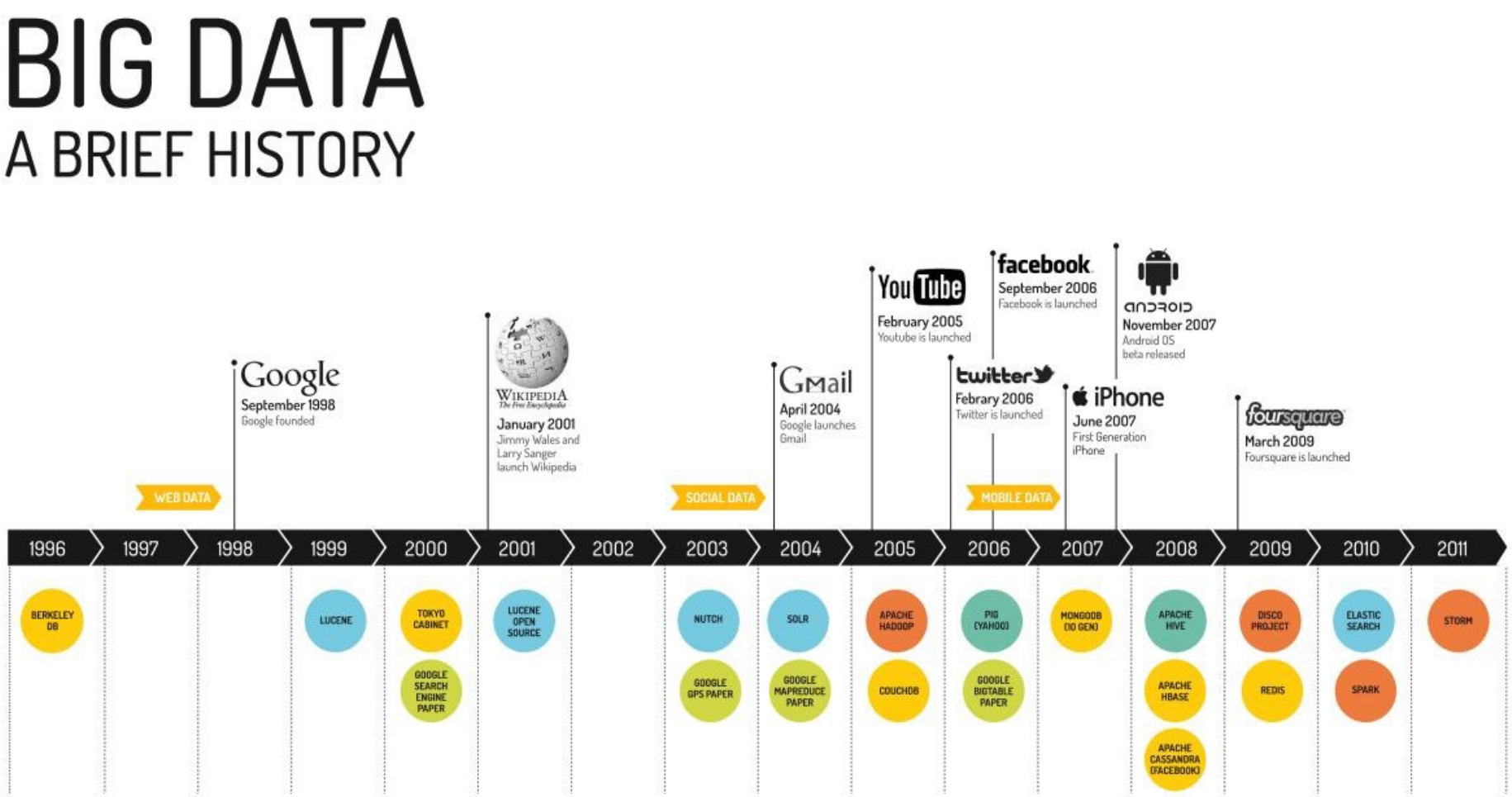
New Model: all of us are generating data, and all of us are consuming data

Challenges in Handling Big Data

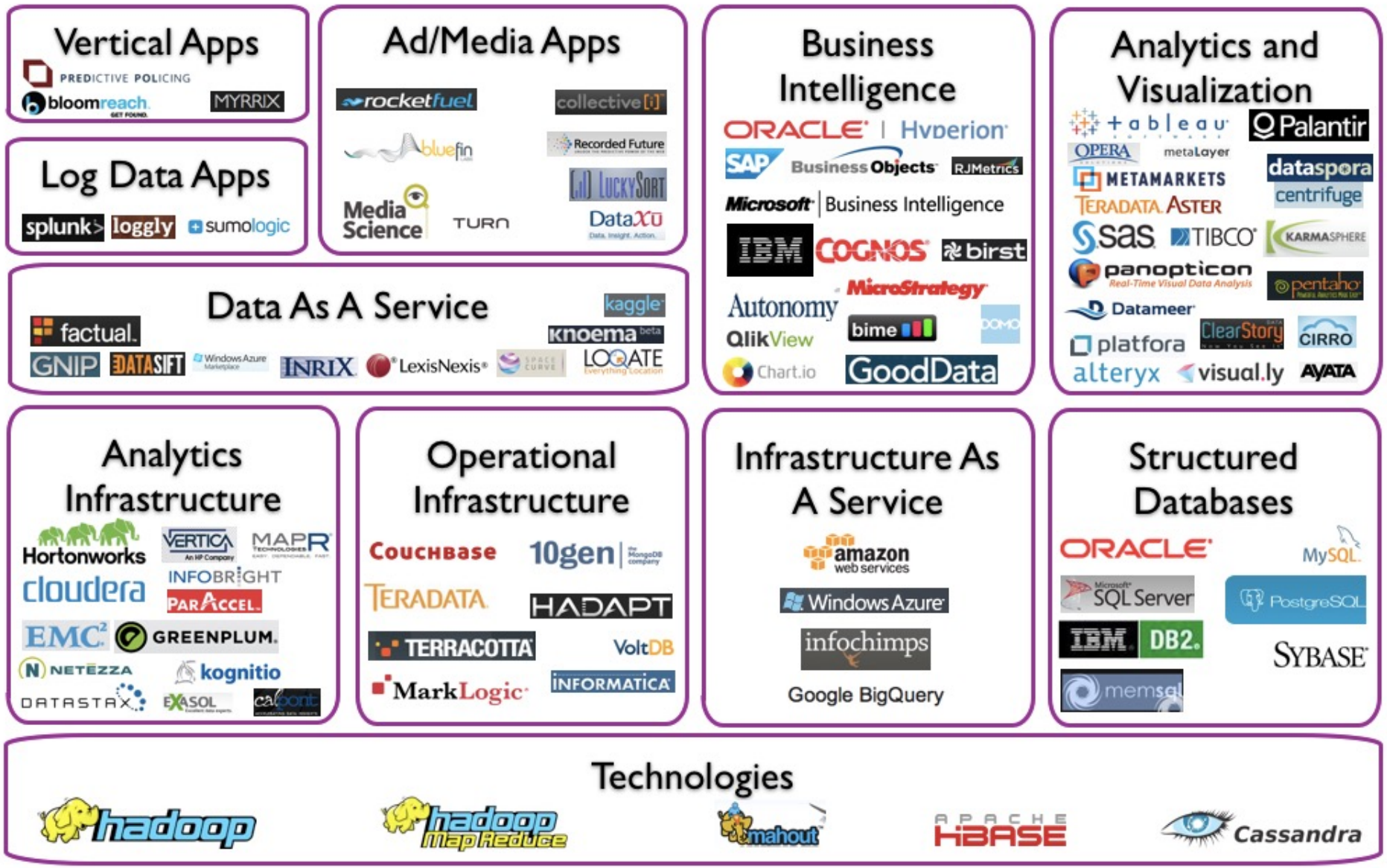
- **The Bottleneck is in technology**
  - New architecture, algorithms, techniques are needed
- **Also in technical skills**
  - Experts in using the new technology and dealing with big data
- **Recently, we observe a convergence to Artifical Intelligence**
  - Big data needs artificial intelligence to analyze the complex volume of data
  - This gives birth to advanced analytics

Big Data Landscape

What technology for Big Data?



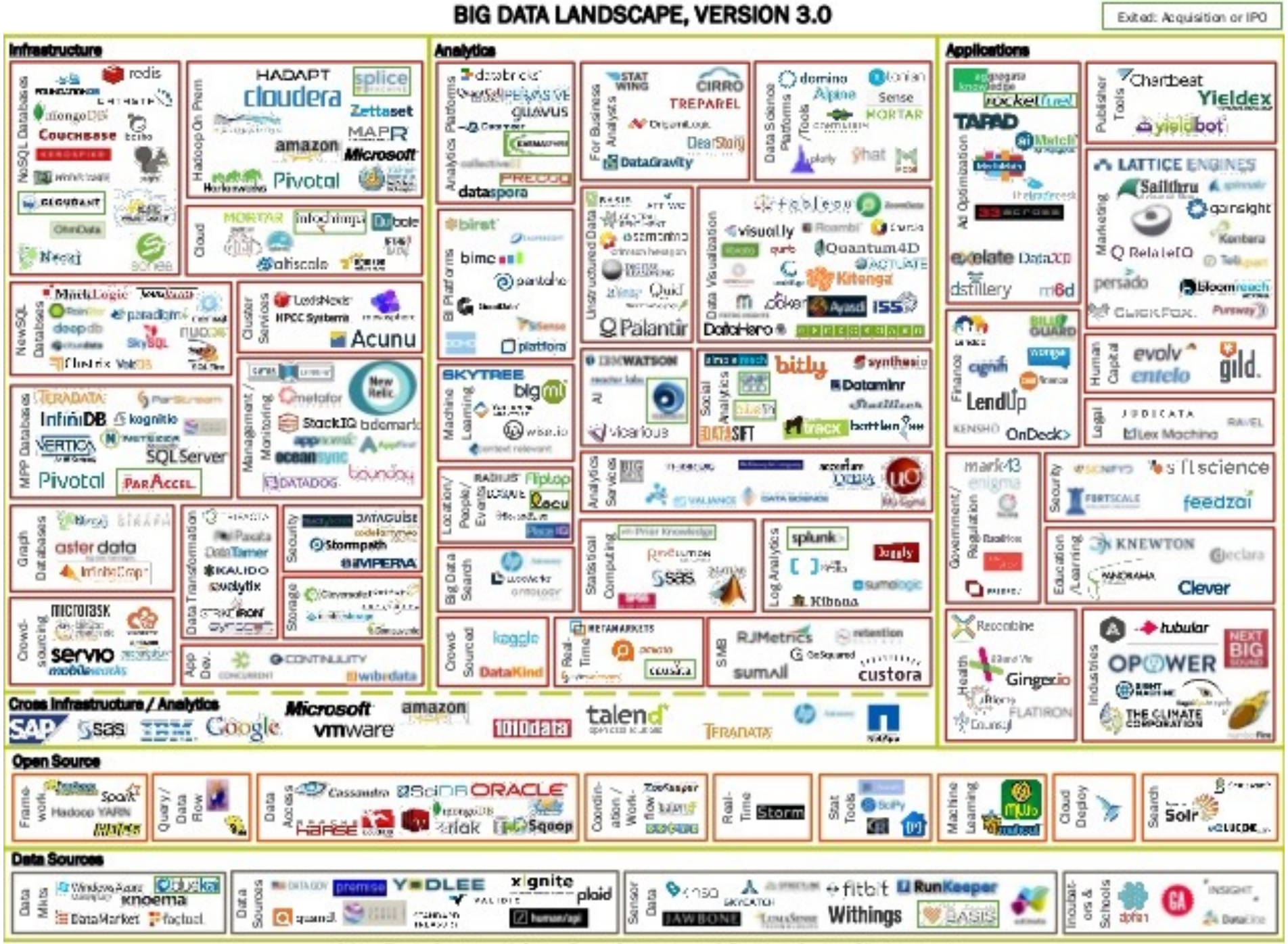
Big Data Landscape



dave@vcdave.com

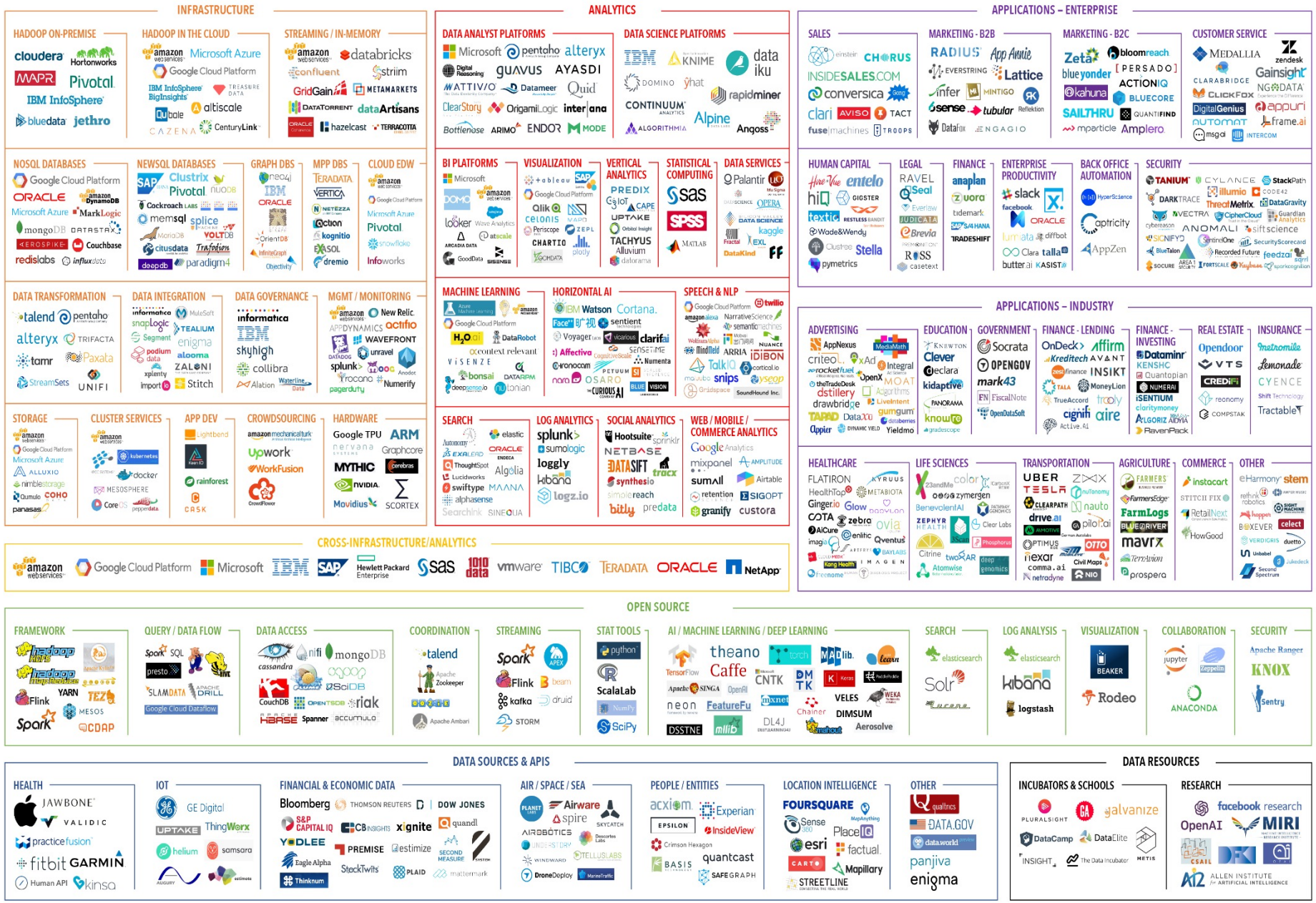
blogs.forbes.com/davefeinleib

Big Data Landscape (Version 2.0)



© Matt Turck (@mattturck), Sutian Dong (@sutiandong) & FirstMark Capital (@firstmarkcap)

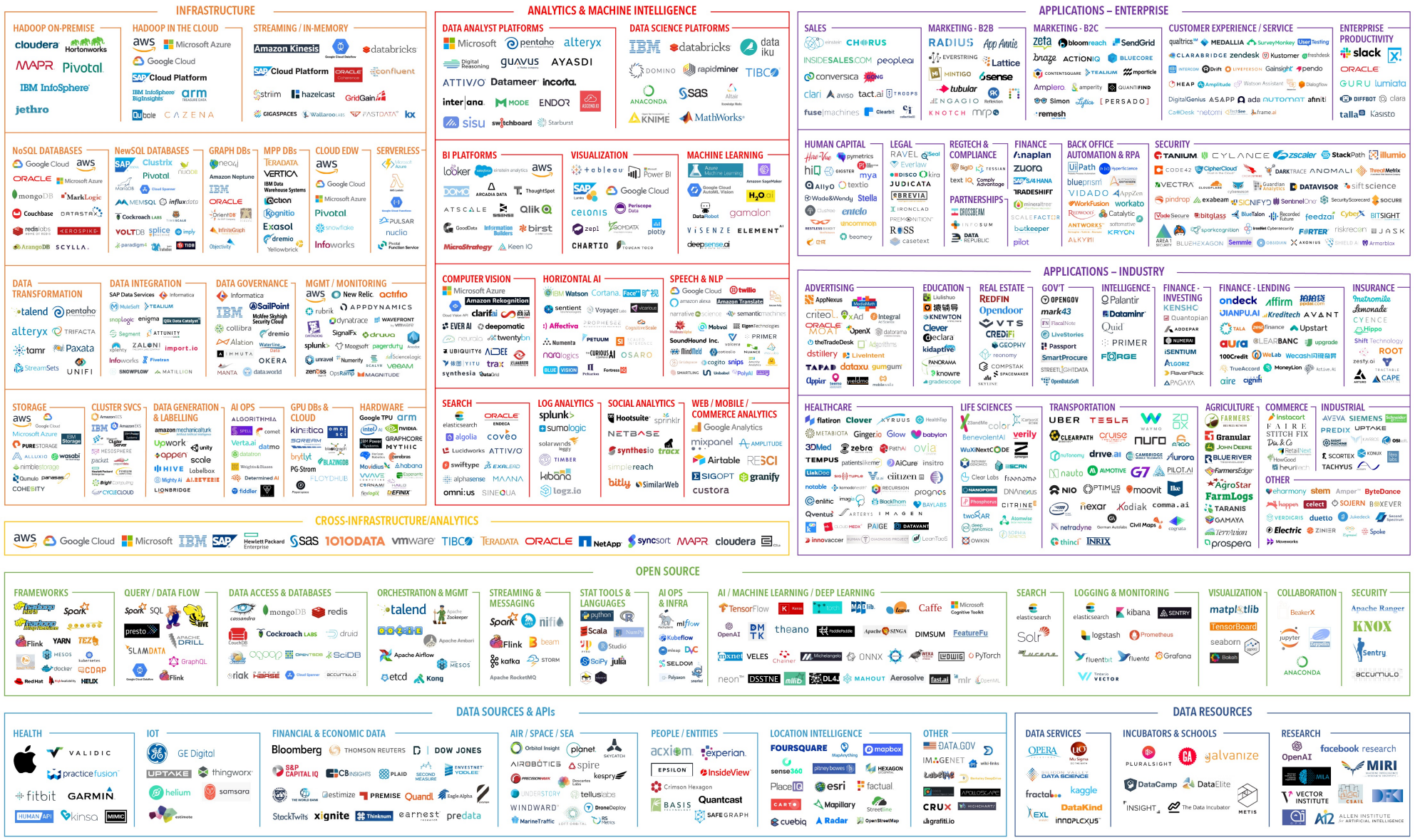
BIG DATA LANDSCAPE 2017



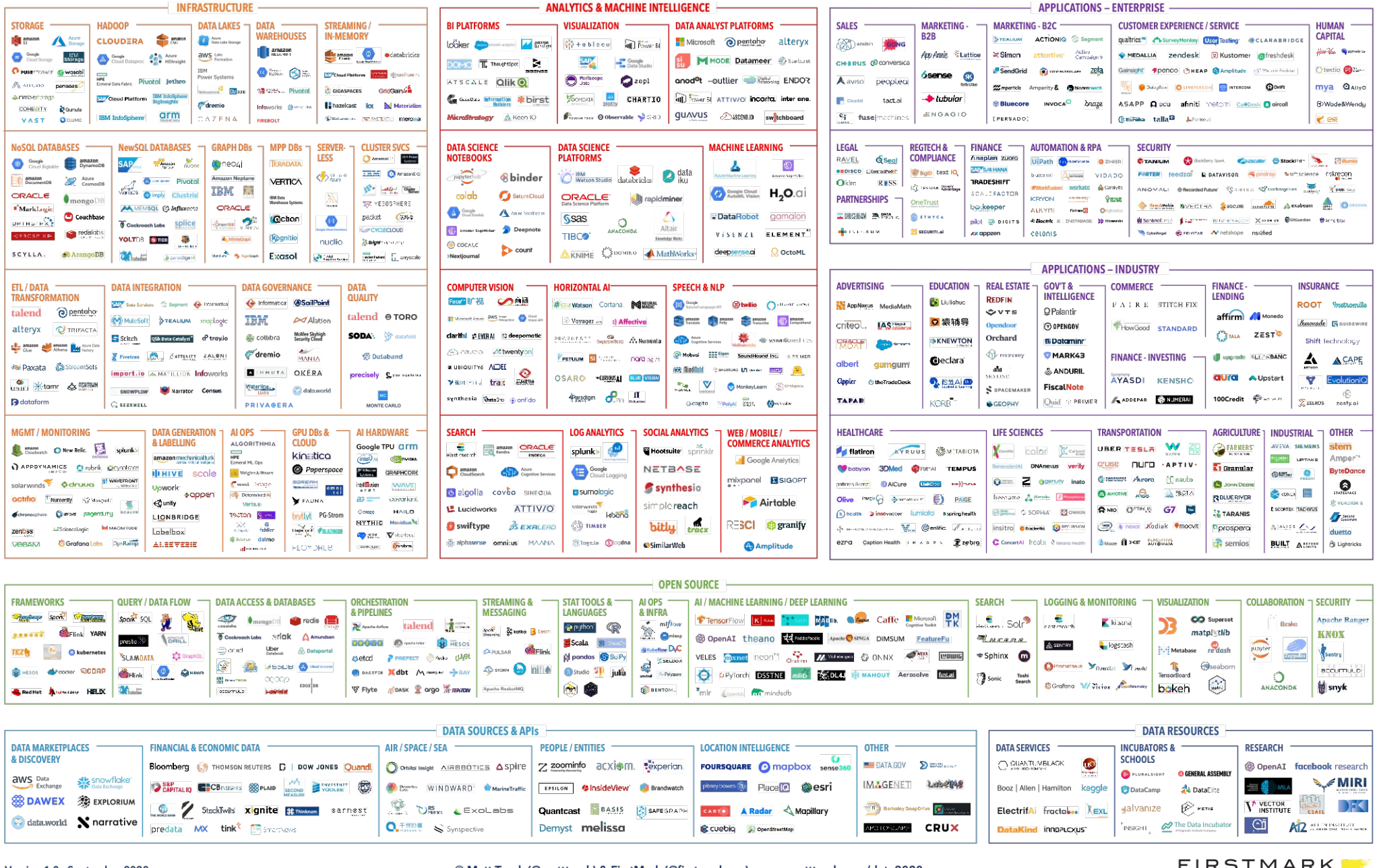
FIRSTMARK
EARLY STAGE VENTURE CAPITAL

BIG DATA & AI LANDSCAPE 2018

DATA & AI LANDSCAPE 2019

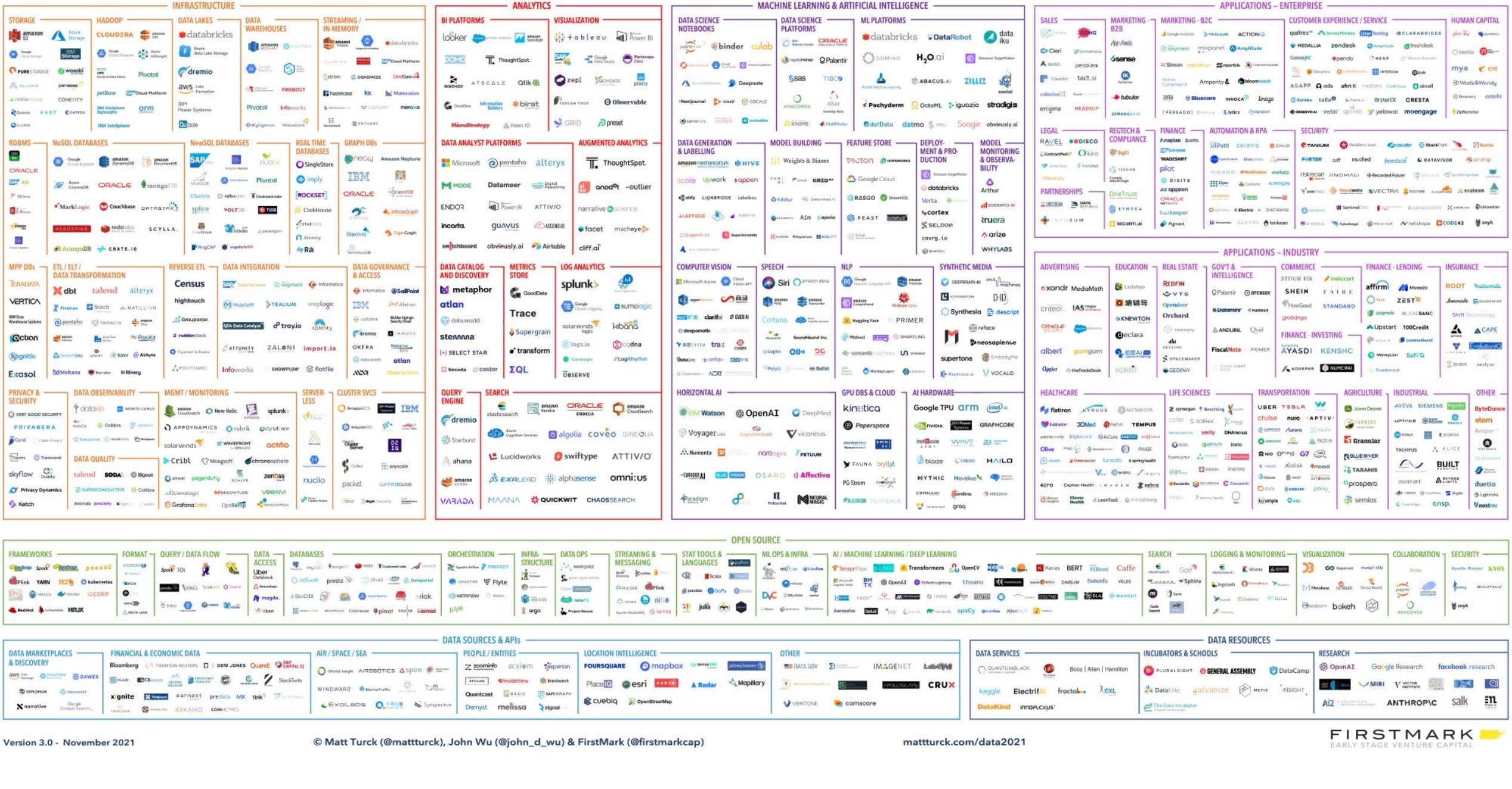


DATA & AI LANDSCAPE 2020



FIRSTMARK
EARLY STAGE VENTURE CAPITAL

MACHINE LEARNING, ARTIFICIAL INTELLIGENCE, AND DATA (MAD) LANDSCAPE 2021

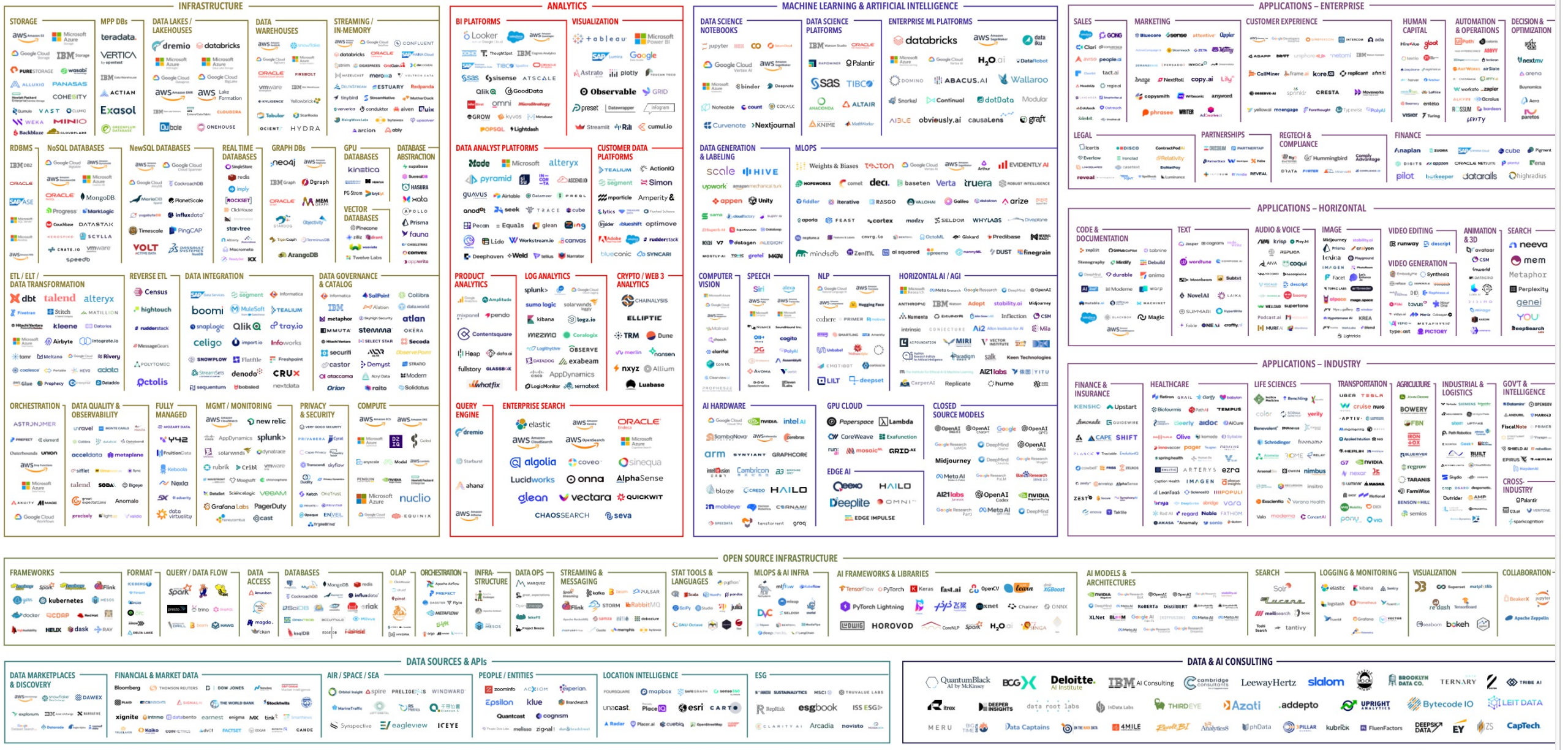


Version 3.0 - November 2021

mattturck.com/data2021

FIRSTMARK
EARLY STAGE VENTURE CAPITAL

THE 2023 MAD (MACHINE LEARNING, ARTIFICIAL INTELLIGENCE & DATA) LANDSCAPE



Version 1.0 - Feb 2023

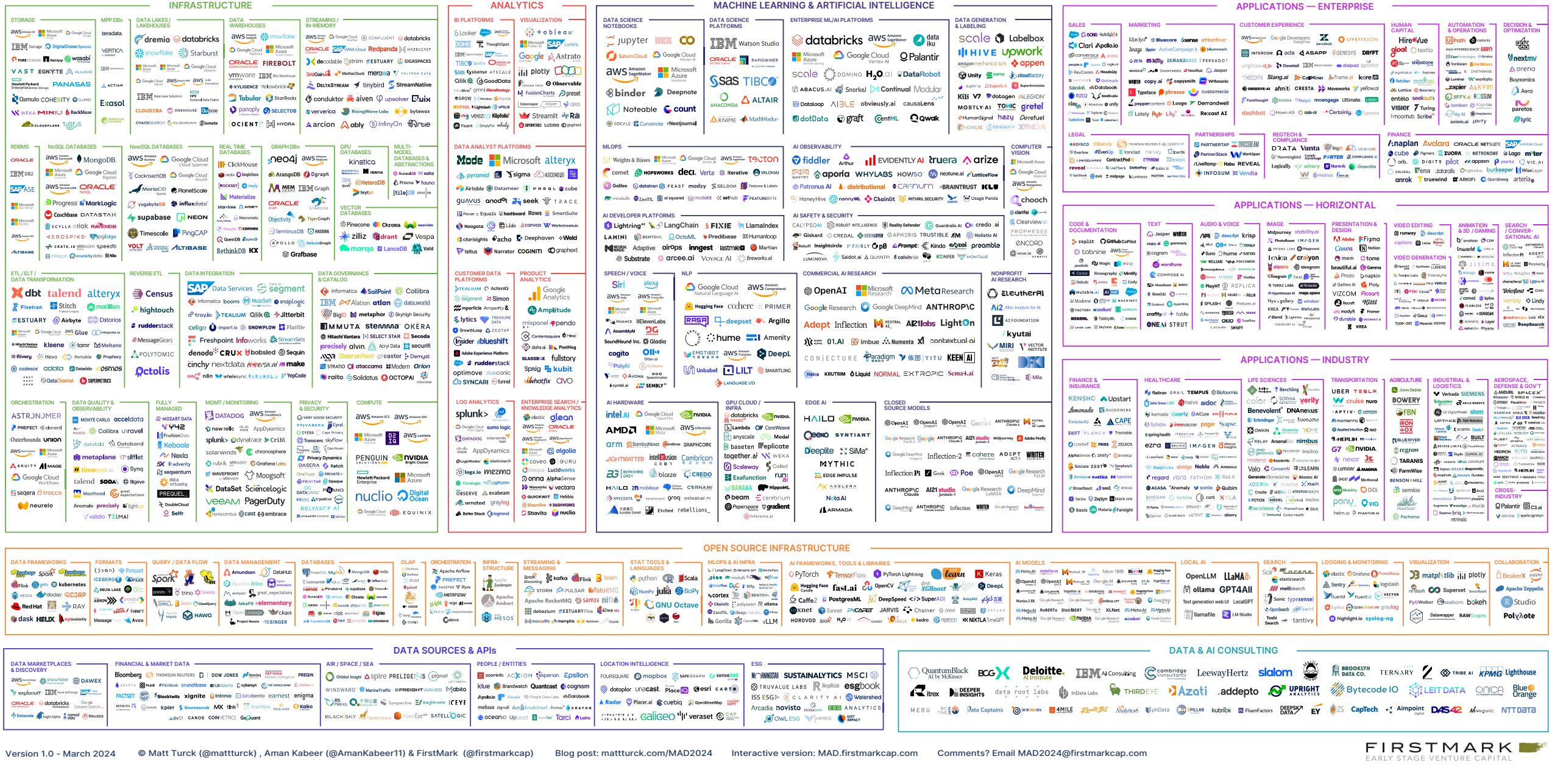
Blog post: mattturck.com/MAD2023

Interactive version: MAD.firstmarkcap.com

Comments? Email MAD2023@firstmarkcap.com

FIRSTMARK
EARLY STAGE VENTURE CAPITAL

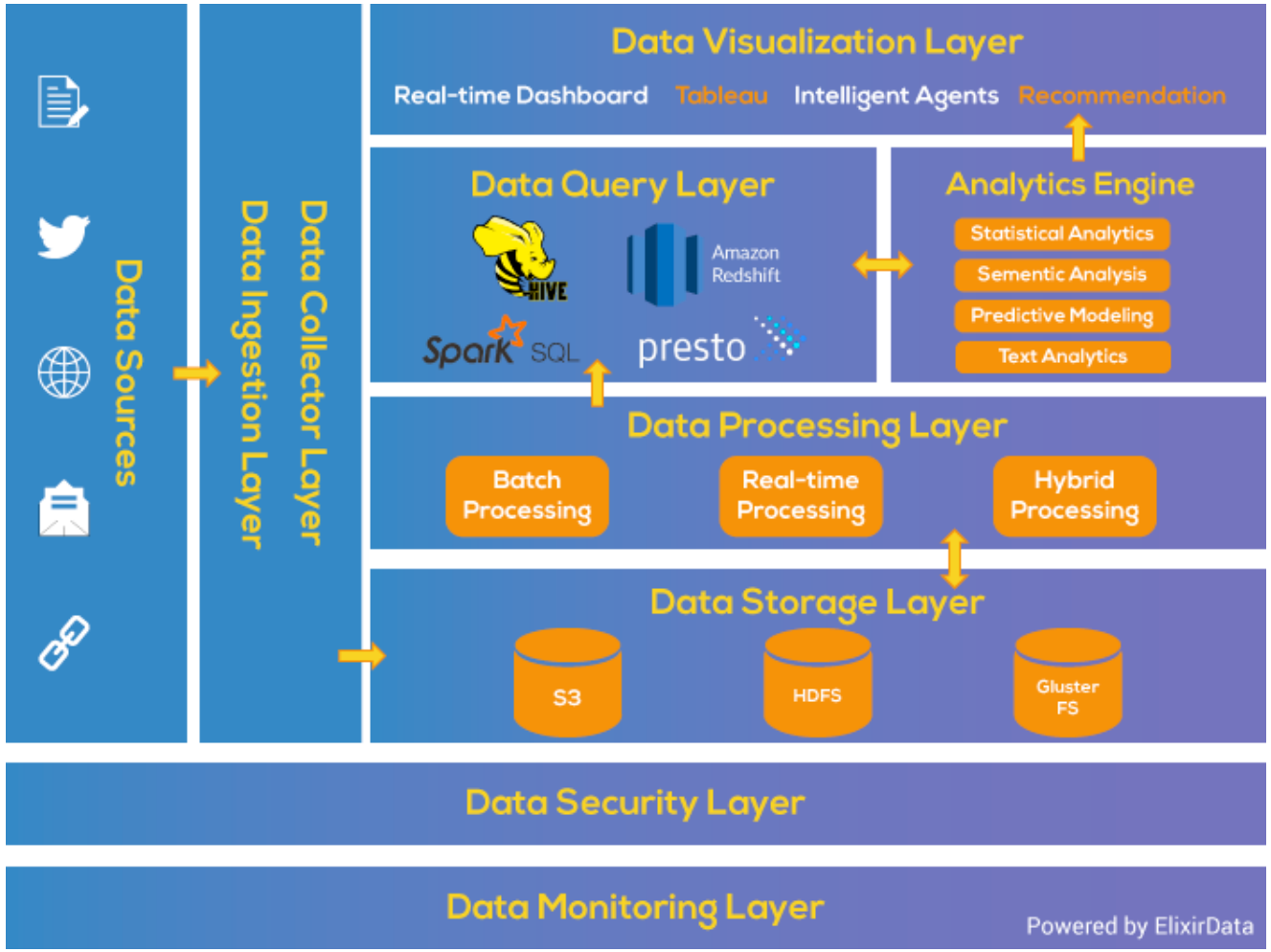
THE 2024 MAD (MACHINE LEARNING, ARTIFICIAL INTELLIGENCE & DATA) LANDSCAPE



FIRSTMARK
EARLY STAGE VENTURE CAPITAL

Big Data Framework

Big Data Framework



https://www.xenonstack.com/blog/ingestion-processing-data-for-big-data-iot-solutions

Data Ingestion Layer

- Ingest data flows from hundreds or thousands of various sources into data center (variable speed, different formats, etc.), extracting the data, and detecting the changed data.
- It's rightly said that "If starting goes well, then, half of the work is already done“
- Data can be streamed in real time or ingested in batches
- Prioritizing data sources, validating individual files and routing data items to the correct destination.
- Data serialization (Apache Thrift, Apache Avro, etc.) is the concept of converting data into a (compressed) format that allows it to be shared or stored such that its original structure can be recovered.
- Few data ingestion tools: Apache Flume, Apache Nifi, Elastic Logstash

Data Collector Layer

- The focus is on transportation data from ingestion layer to rest of data pipeline.
- Often, we use a messaging system that will act as a mediator between all the programs that can send and receive messages.
- Data Collector Tools: Apache Kafka

Data Processing Layer

- The focus is to specialize data pipeline processing system
- Processing can be done in 3 ways:
  - Batch Processing System : A pure batch processing system for off-line analytic. For example, Apache Sqoop efficiently transfers bulk data between Apache Hadoop and structured datastores such as relational databases.
  - Near Real Time Processing System: A pure online processing system for on-line analytic. For example, the Apache Storm cluster makes decisions about the criticality of the event and sends the alerts to the alert system (dashboard, e-mail, other monitoring systems).
  - Hybrid Processing system: This consist of batch and real-time processing system capabilities. For this type of processing, tool used can be Apache spark and apache Flink.

Data Storage Layer

- Keep data in the right place based on usage
- A new concept is useful: **polyglot persistence** is the idea of using multiple databases to power a single application.
- Polyglot persistence is the way to share or divide your data into multiple databases and leverage their power together.
- Tools used for Data Storage: HDFS, Gluster file systems (GFS), Amazon S3

Data Query Layer

- This is the layer where strong analytic processing takes place.
- This is a field where interactive queries are necessaries and it's a zone traditionally dominated by SQL expert developers.
- Companies from all industries use analytics queries to:
  - Increase revenue
  - Decrease costs
  - Increase productivity
- Tools Used for Analytics Query: Apache Hive, Spark SQL, Amazon Redshift

Analytics Engine

- Analytics engine is the use of advanced analytic techniques against very large, diverse data sets.
- Data analytics became an essential step while computing such a large amount of data.
- Analyzing big data allows analysts, researchers, and business users to make better and faster decisions using data that was previously inaccessible or unusable.
- Using advanced analytics techniques such as text analytics, machine learning, predictive analytics, data mining, statistics, and natural language processing.
- Analytics tools: Apache Spark, Machine learning libraries, R

Data Visualization Layer

- This layer focus on Big Data Visualization. We need something that will grab people's attention, pull them in, make your findings well-understood.
- That's why it provides full business infographics: your own findings from your own data need the annotation and the bold canvas.
- The data visualization layer often is the thermometer that measures the success of the project. This is the where the data value is perceived by the user.
- Few tools used for dashboards: Tableau, Kibana, AngularJS

Data Security Layer

- Security is the primary task for any kind of work.
- Security should be implemented at all layers of the lake starting from ingestion, through storage, analytics, discovery, all the way to consumption.

Data Monitoring Layer

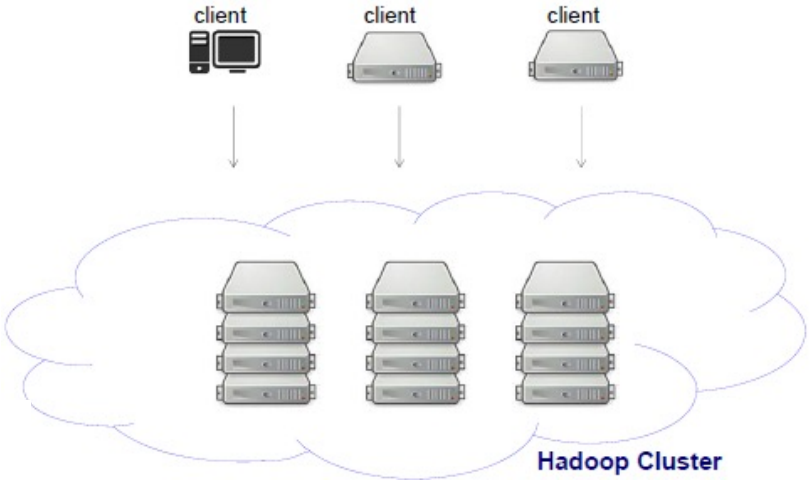
- Data in enterprise systems is like food: it has to be kept fresh, also it needs nourishment, otherwise it goes bad and doesn't help you in making strategic and operational decisions. Just as consuming spoiled food could make you sick, using "spoiled" data may be bad for your organization's health.

- Continuous monitoring of data is an important part of the governance mechanisms.

Big Data with a Cluster

Main Example: Hadoop Cluster

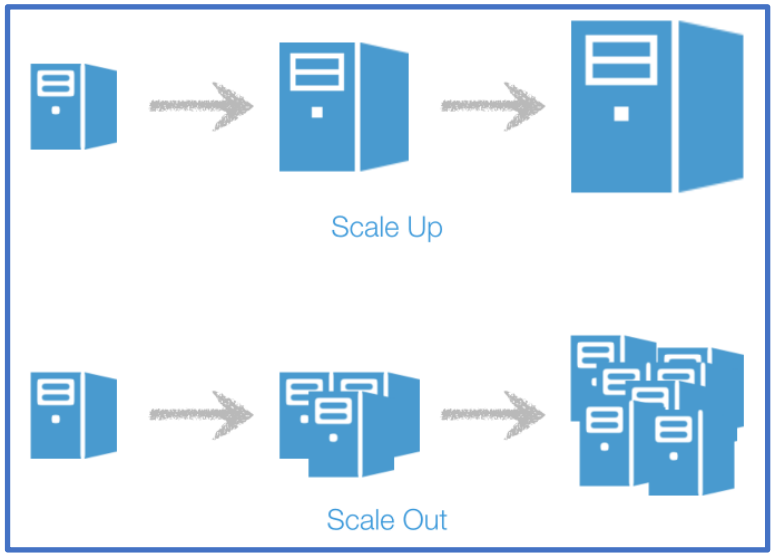
- **Many details on** http://hadoop.apache.org
- **A set of "cheap" commodity hardware**
  - No need for super-computers, use commodity unreliable hardware
  - Not desktops
- **Networked together**
- **May reside in the same location**
  - Set of servers in a set of racks in a data center



Big Data System Principles with a Cluster

- Scale-Out rather than Scale-Up
- Bring code to data rather than data to code
- Deal with failures – they are common
- Abstract complexity of distributed and concurrent applications

Scale-Out Instead of Scale-Up

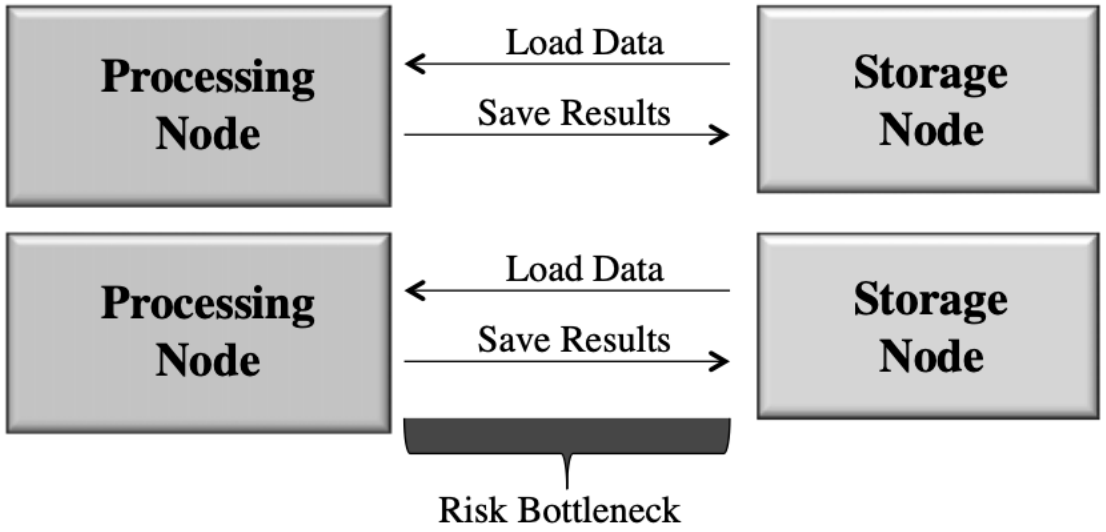


- **It is harder and more expensive to scale-up**
  - Add additional resources to an existing node (CPU, RAM)
  - New units must be purchased if required resources can not be added

- **Scale-Out**
  - Add more nodes/machines to an existing distributed application
  - Software Layer is designed for node additions or removal
  - A set of nodes are bonded together as a single distributed system
  - Very easy to scale down as well

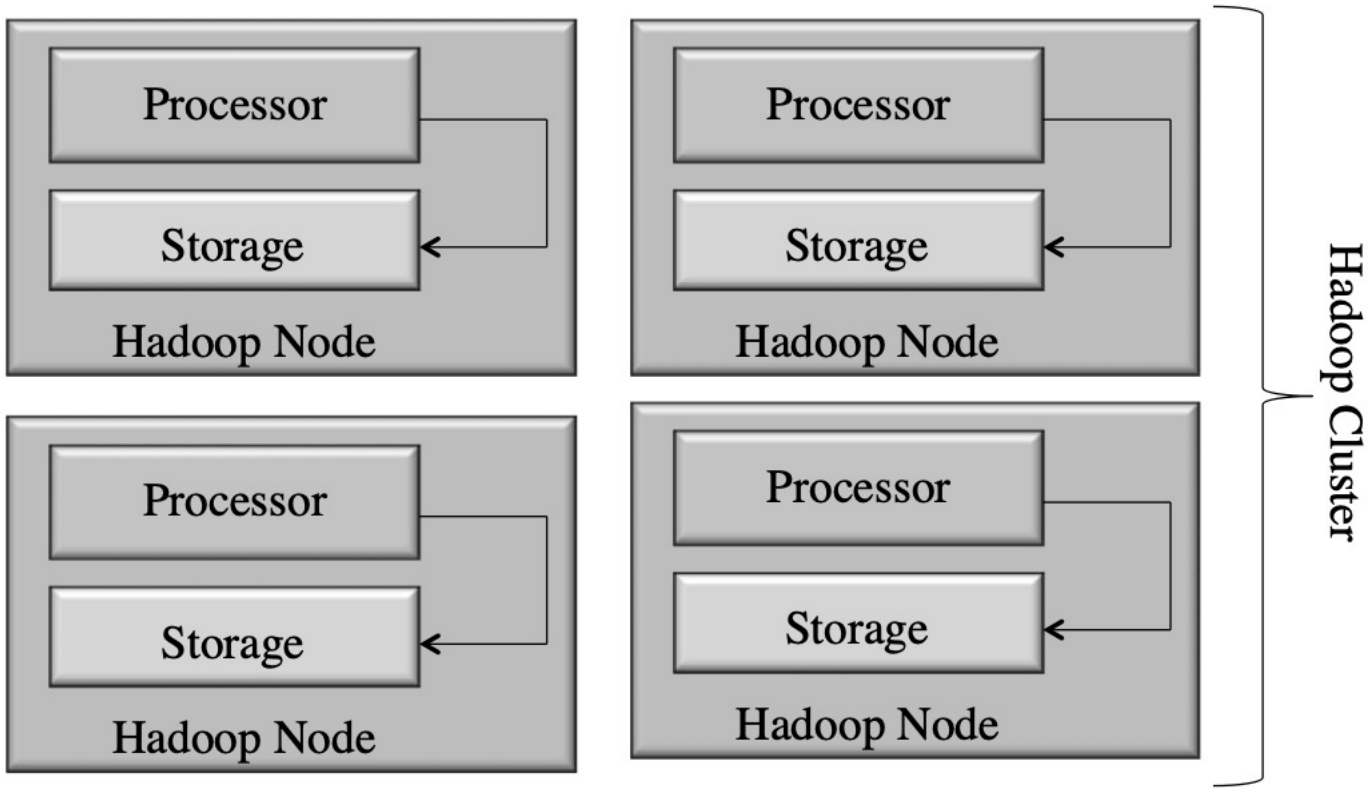
Code to Data

- **Traditional data processing architecture**
  - Nodes are broken up into separate processing and storage nodes connected by high-capacity link
  - Many data-intensive applications are not CPU demanding causing bottlenecks in network



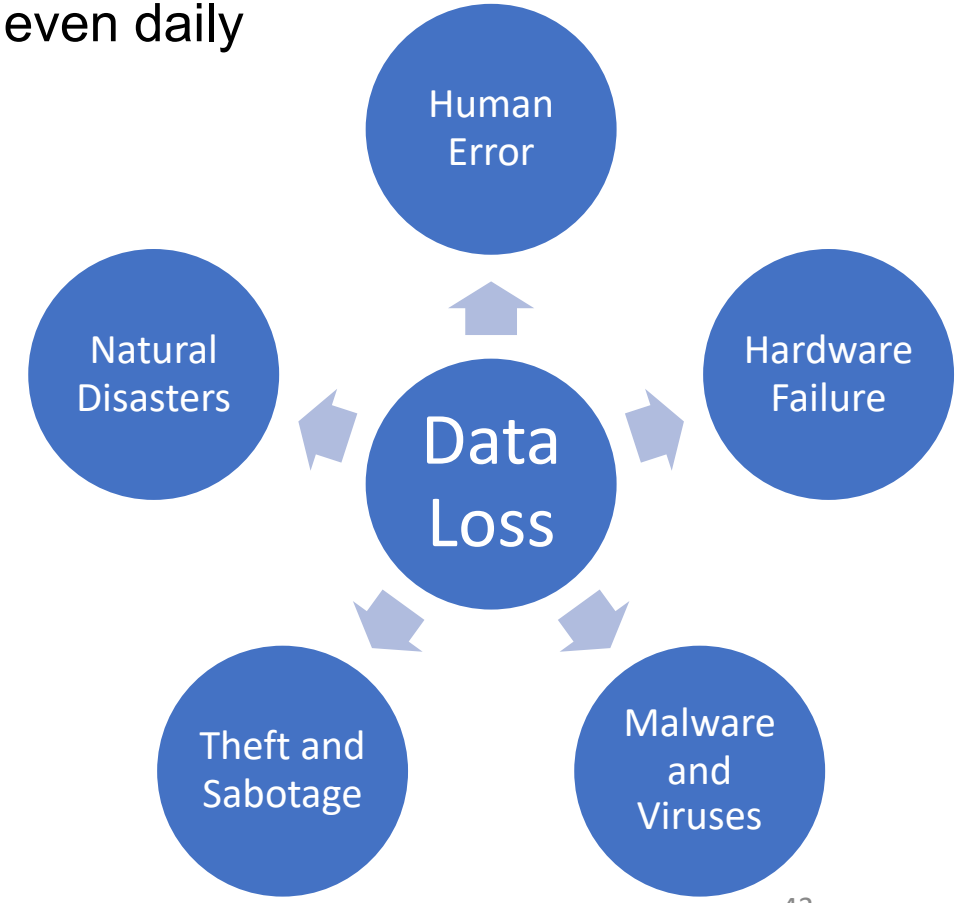
Code to Data

- **Co-locate processors and storage**
  - Code is moved to data (size is tiny, usually in KBs)
  - Processors execute code and access underlying local storage



Failures are Common

- **Given a large number machines, failures are common**
  - Large warehouses may see machine failures weekly or even daily
- **Hadoop is designed to cope with node failures**
  - Data is replicated
  - Tasks are retried

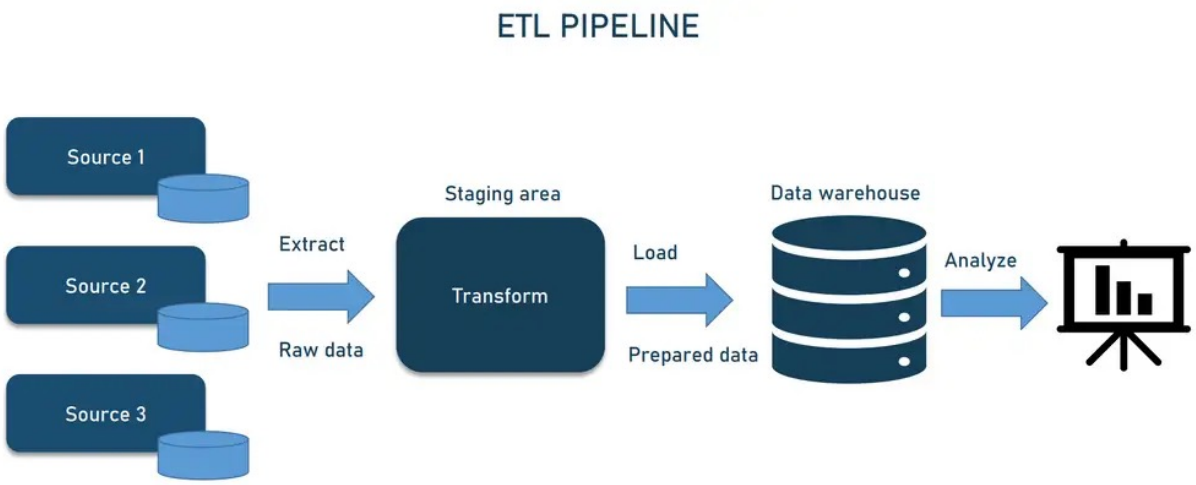


Abstract Complexity

- **Abstract many complexities in distributed and concurrent applications**
  - Defines small number of components
  - Provides simple and well-defined interfaces of interactions between these components

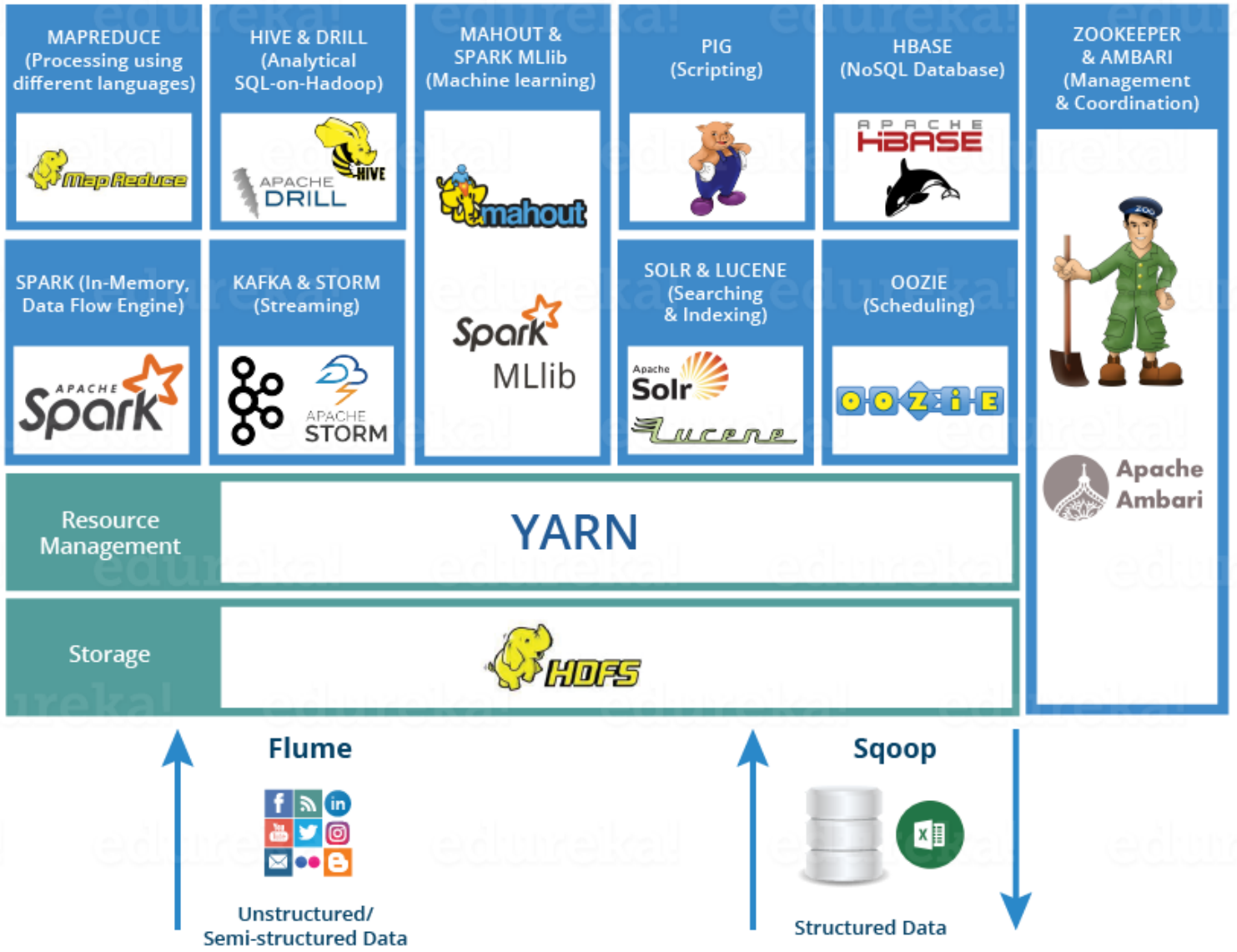
- **Frees developer from worrying about system level challenges**
  - Some issues: Race conditions, data starvation
  - Some solutions: Processing pipelines, data partitioning, code distribution

- **Allows developers to focus on application development and business logic**



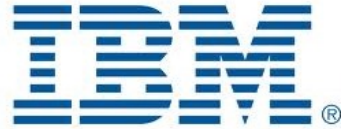
ETL: Extract, Transform and Load

Big Data Ecosystem (few elements)



Many Distribution Vendors

- Cloudera Distribution for Hadoop
- MapR Distribution
- IBM Open Platform
- VMware Tanzu
- Google Cloud Platform
- Microsoft Azure



VMware Tanzu

Google Cloud Platform

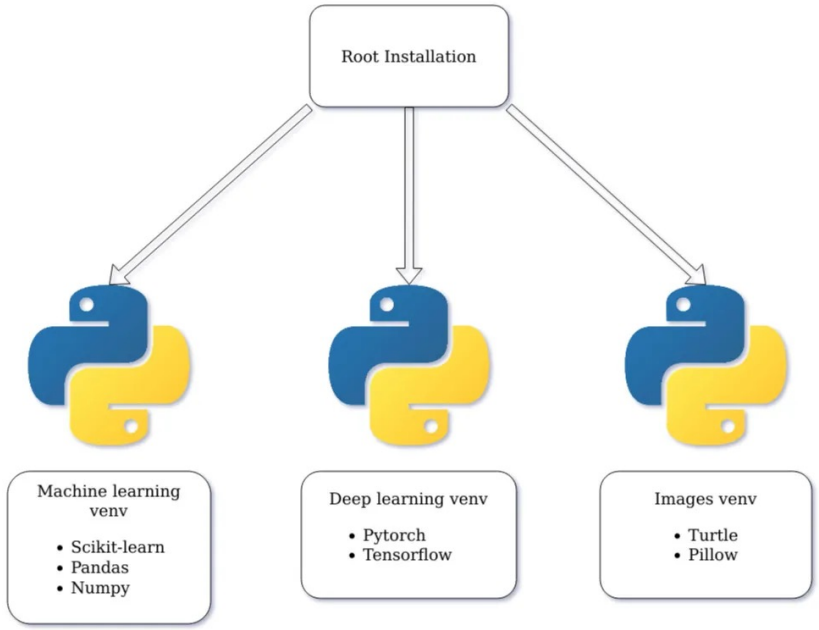
Introduction to Docker

Virtual Environment

Why should we use virtual environment?

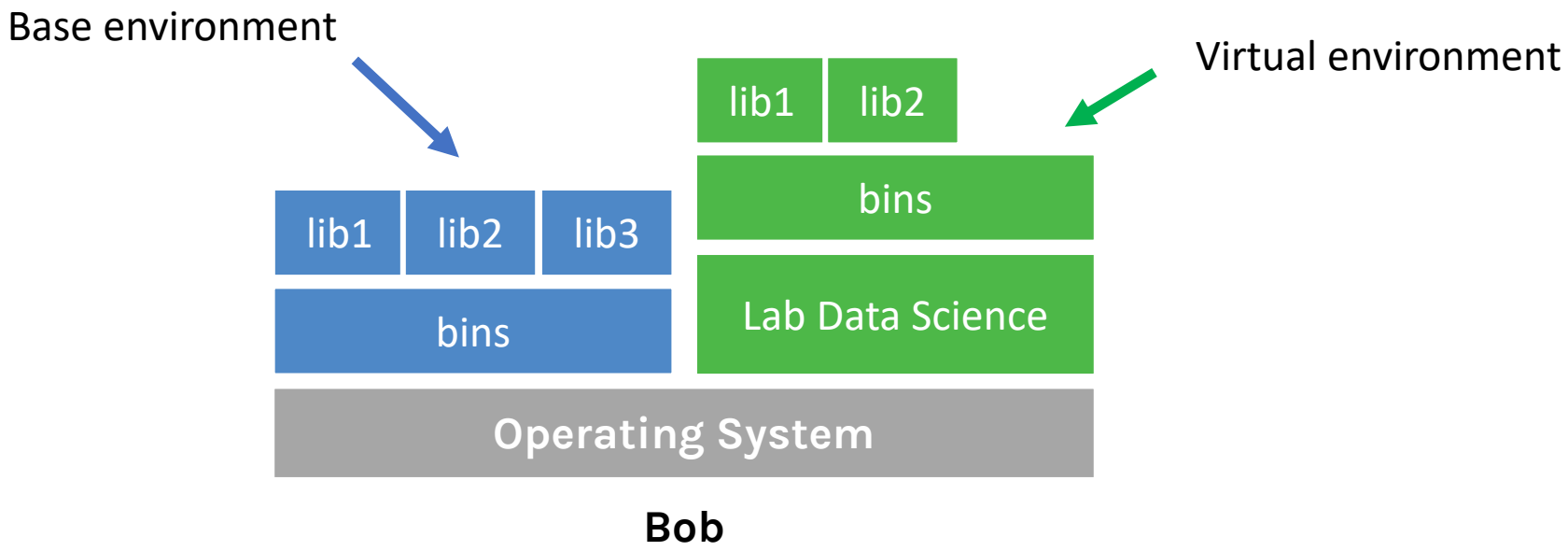
- Virtual environments help to make development and use of code more streamlined.
- Virtual environments keep dependencies in separate “sandboxes” so you can switch between both applications easily and get them running.
- Given an operating system and hardware, we can get the exact code environment set up using different technologies.

Example in Python



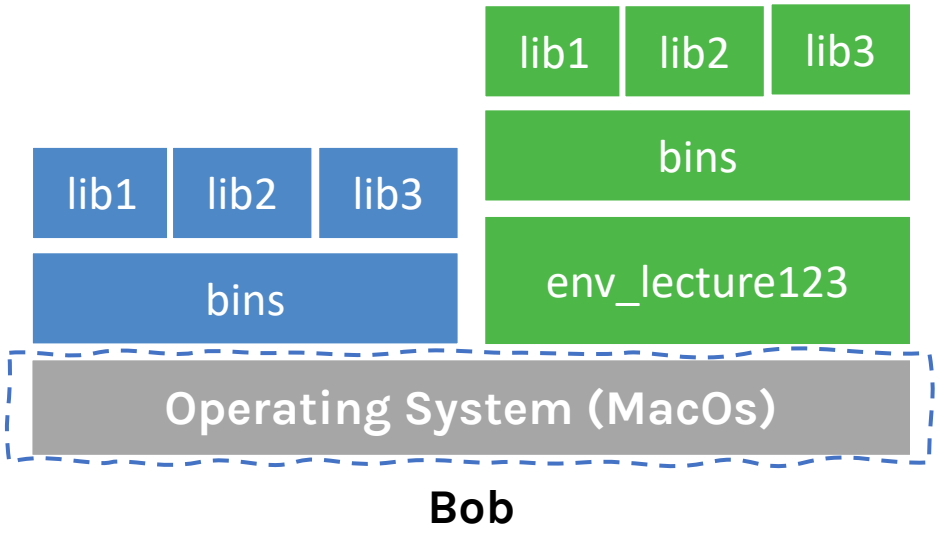
Example with Python

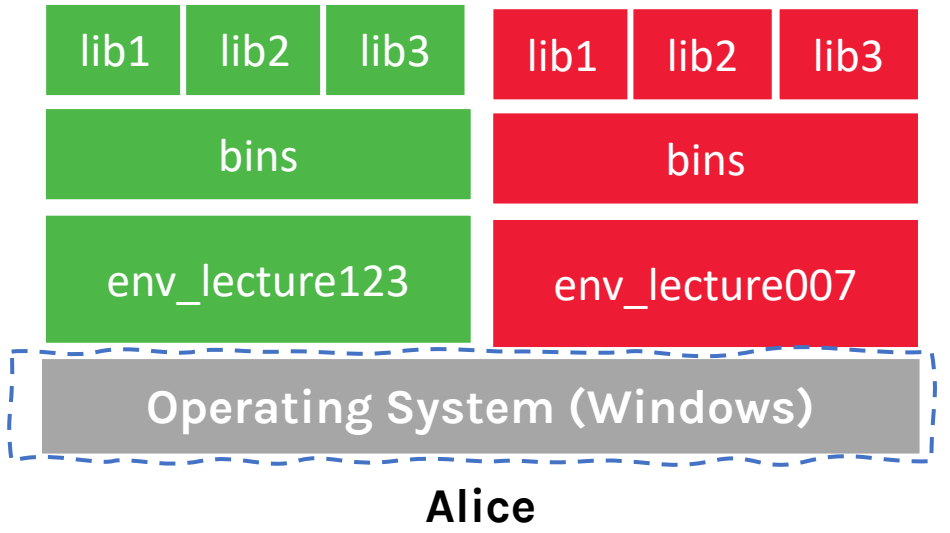
- For its lab in data science, Bob thinks that it would be good to isolate the new environment from the base environment avoiding any conflict with the installed packages.
- He adds a layer of abstraction called **virtual environment** that helps him keep the modules organized and avoid misbehaviors while developing new projects.



Main drawback

- What could go wrong? Unfortunately, Bob and Alice reproduce different results and they think the issue relates to their operating systems. Indeed while Bob has a MacOS, Alice uses a Windows OS.

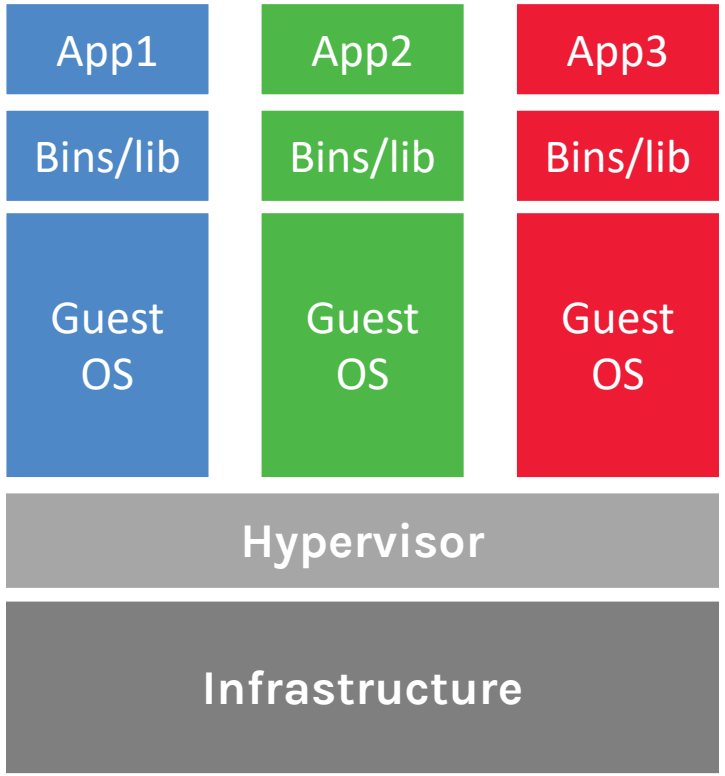




Virtual Machine

What are Virtual Machines (VM)?

- Virtual machines have their own virtual hardware: CPUs, memory, hard drives, etc.
- VMs need a hypervisor that manages different virtual machines on server
- **Hypervisor** can run as many virtual machines as you wish
- Operating system is called the « **host** » while those running in a virtual machine are called « **guest** »
- You can install a completely different operating system on this virtual machine



Machine Virtualization

Why should we use virtual machines?

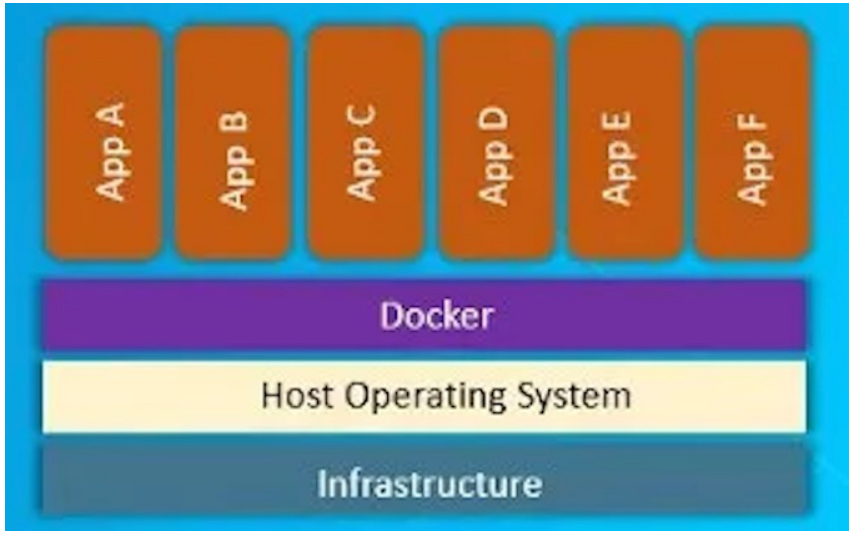
- Even though by using virtual environments we isolate our computations, we might need to use the same operating system that must run regardless of the physical machine

- **Isolation!**
  - Each virtual machine is kept isolated from its host physical system and other virtualized machines.
  - If one virtual model crashes, the other virtual machines and the host system don’t affect each other.
  - In addition, data isn't shared between one virtual model and another.

- **Full autonomy:** it works like a separate computer system, it is like run a computer within a computer.
- **Very secure:** the software inside the virtual machines can't affect the actual computer.
- **Lower costs:** buy one machine and run multiple operating systems.

Container

What is a container?



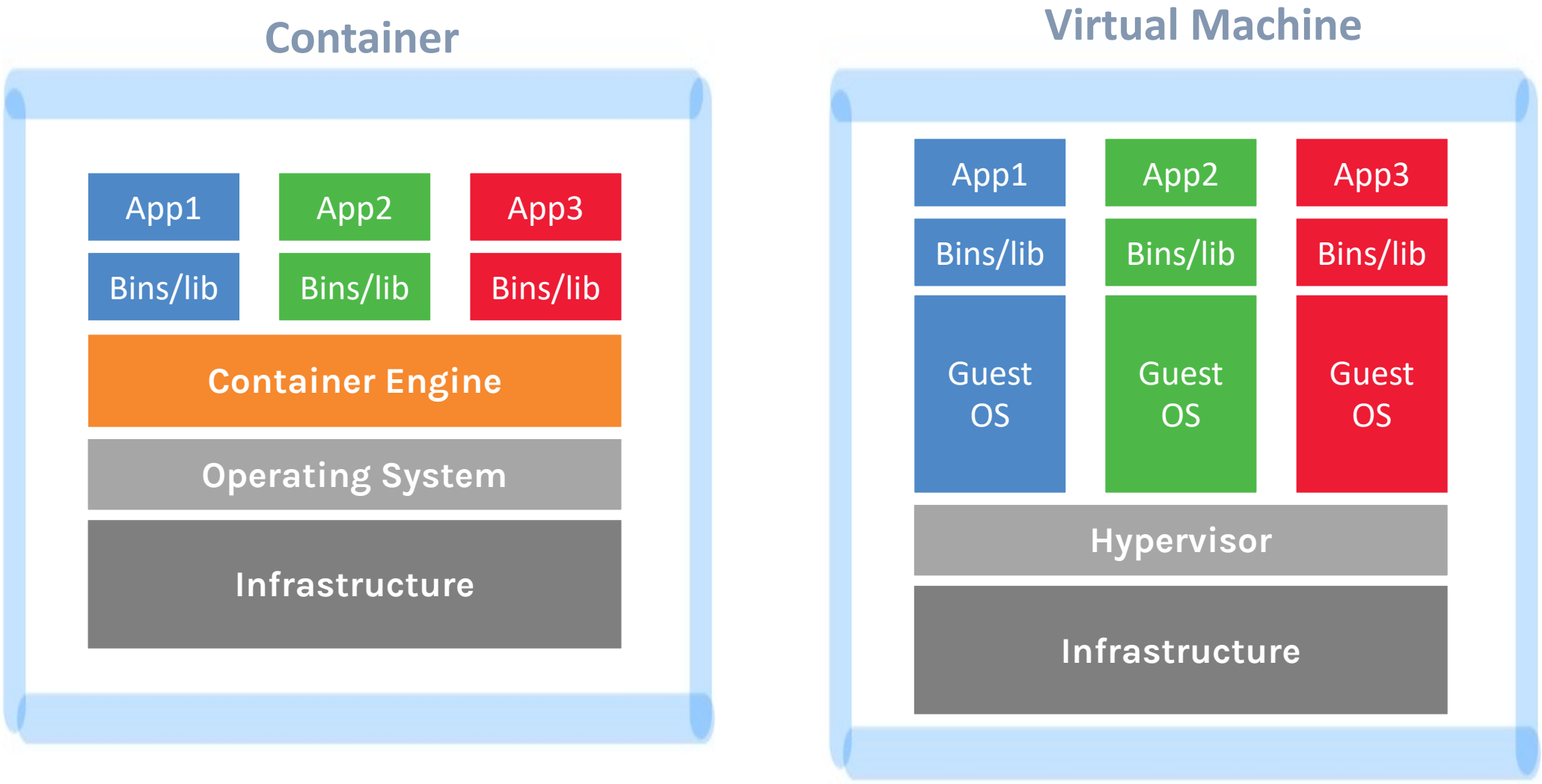
- Standardized packaging for software dependencies
- Isolate apps from each other
- Works for all major Linux distributions, MacOS, Windows

Docker Containers are not Virtual Machines

Virtual Machines

Containers

Docker Container vs Virtual Machines



Docker Container vs Virtual Machines (VM)

VMs

- Each VM runs its own OS
- Boot up time is in minutes
- Not version controlled
- Cannot run more than couple of VMs on an average laptop
- Only one VM can be started from one set of VMX and VMDK files

Docker

- Container is just a user space of OS
- Containers instantiate in seconds
- Images are built incrementally on top of another like layers.
- Images can be diffed and can be version controlled.
- Can run many Dockers in a laptop
- Multiple docker containers can be started from one Docker image

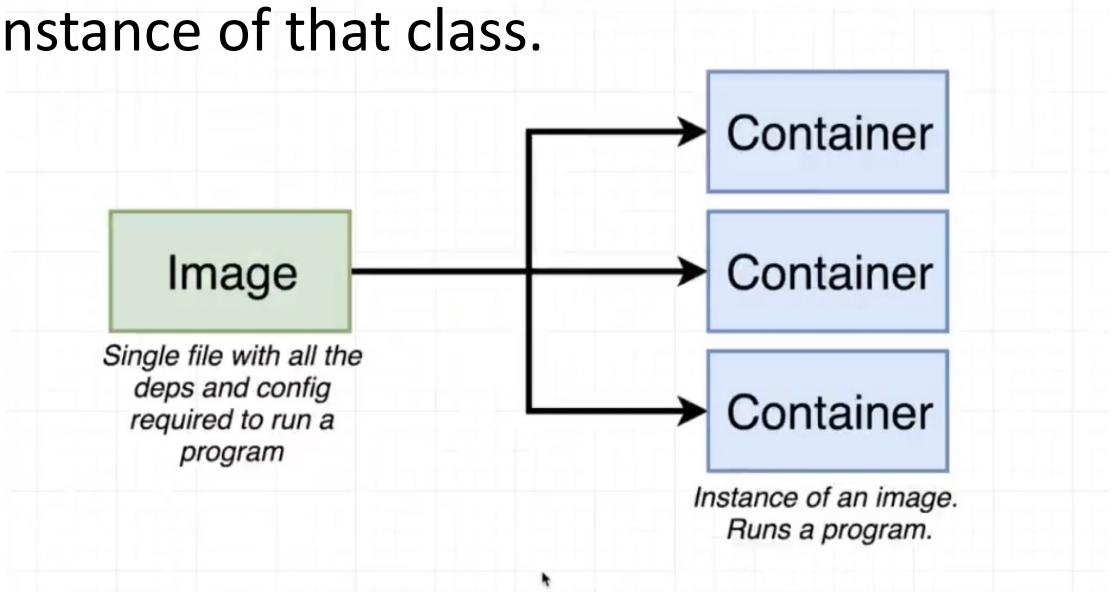
History of Containers (just a few dates)

- 1979: Unix V7 (beginning of process isolation)
- 2000: FreeBSD Jails
- 2001: Linux VServer
- 2004: Solaris Containers
- 2008: LXC (LinuX Containers)
- 2013: **Docker**
- 2018: The Gold Standard (containerization becomes the foundation for modern software infrastructure)
- 2020: Kubernetes Grows Up and **Docker Desktop** for Windows Home is released
- 2022: Record Adoption of Container Technologies

https://www.aquasec.com/blog/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016/

Images and Containers

- Docker Image is a template aka blueprint to create a running Docker container.
- Docker uses the information available in the Image to create (run) a container.
- Possible interpretations
  - Image is like a recipe, container is like a dish
  - You can think of an image as a class (object-oriented programming) and a container is an instance of that class.

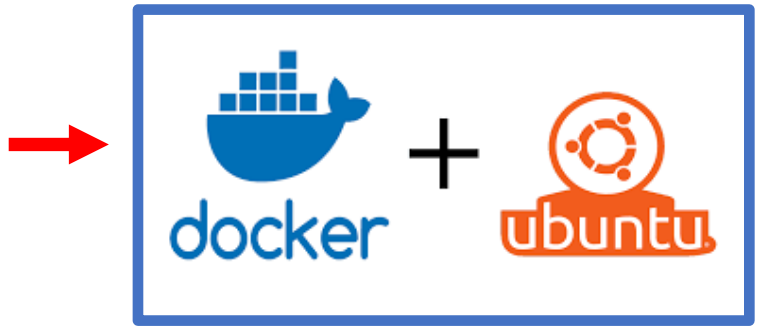


How to build an image

- We use the Dockerfile, a simple text file, to build the Docker Image, which are iso files and other files.
- We run the Docker Image to get Docker Container.

```
FROM ubuntu:22.04

# install packages
RUN apt-get update

# install python and Jupyterlab
RUN apt-get install -y python3-pip

# expose a specific port
EXPOSE 8888

# Command to run when the container starts
CMD ["/bin/bash"]
```

Dockerfile



Docker image
(Ubuntu for example)

Docker container

Inside the Dockerfile

```
FROM alpine:latest
RUN apk update
RUN apk add nmap
ENTRYPOINT [``nmap’’]
CMD [``localhost’’]
```

Dockerfile

FROM: This instruction in the Dockerfile tells the daemon which base image to use while creating our new Docker image.

- In the example here, we are using a very minimal OS image called alpine (just 5 MB of size).

RUN: This command instructs the Docker daemon to run the given commands as it is while creating the image.

- A Dockerfile can have multiple RUN commands

ENTRYPOINT: The ENTRYPOINT instruction is used when you would like your container to run the same executable every time. Usually, ENTRYPOINT is used to specify the binary and CMD to provide parameters.

CMD: The CMD sets default command and/or parameters when a docker container runs.

Nmap ("Network Mapper") is a free and open source utility for network discovery and security auditing.

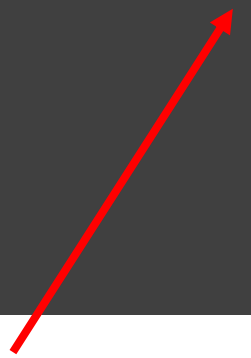
Multiple containers from same image

- You could think of an image as instating a class.
- You could instate it with different parameters using the CMD and therefore different containers will be different.

```
FROM ubuntu:latest
RUN apt-get update
ENTRYPOINT ["/bin/echo", "Hello"]
CMD ["world"]
```

Dockerfile

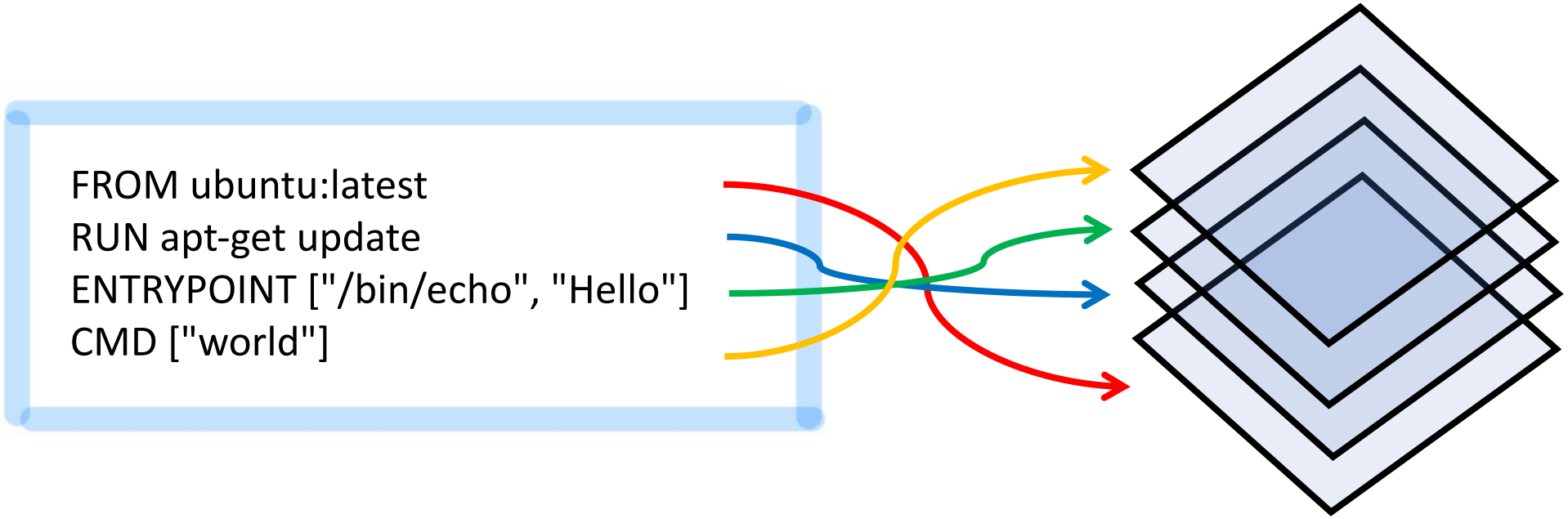
```
> docker build -t hello:first -f Dockerfile .

> docker run -it hello:first
> Hello world
> docker run -it hello:first Bob
> Hello Bob
```



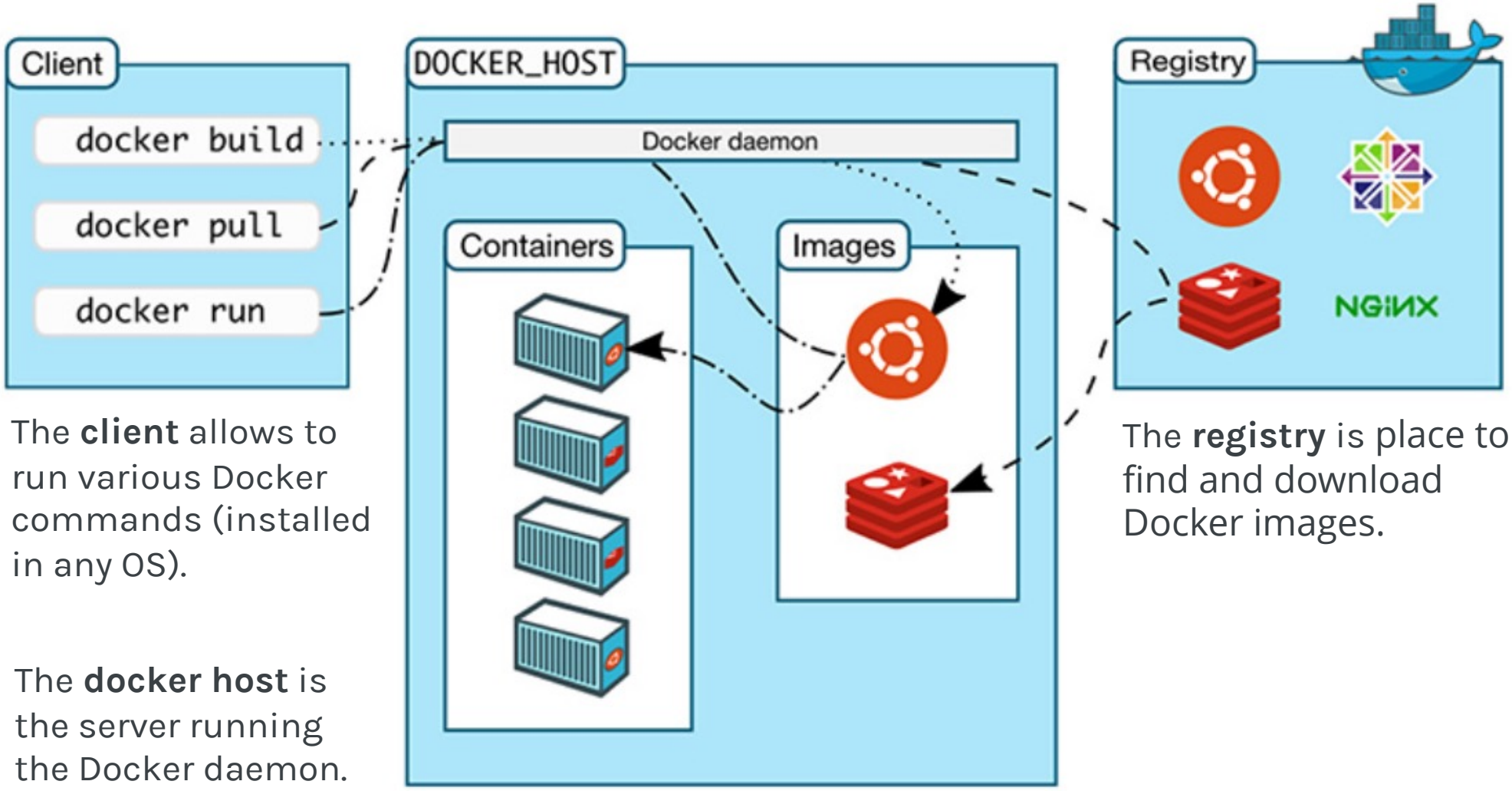
The period means the current working directory.

Docker Image as Layers

- When we execute the build command, the daemon reads the Dockerfile and creates a layer for every command.
- The builder attempts to reuse layers from earlier builds.



The Docker Engine Architecture



The **client** allows to run various Docker commands (installed in any OS).

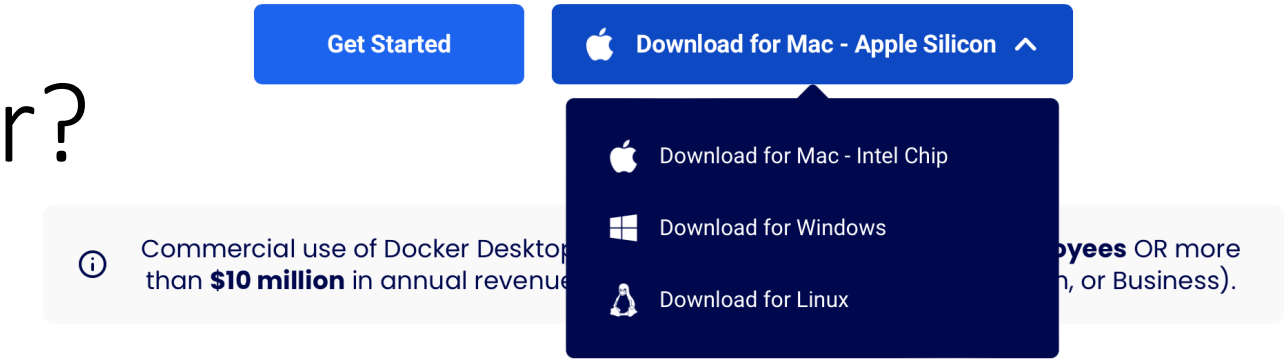
The **docker host** is the server running the Docker daemon.

The **registry** is place to find and download Docker images.

https://nickjanetakis.com/blog/understanding-how-the-docker-daemon-and-docker-cli-work-together

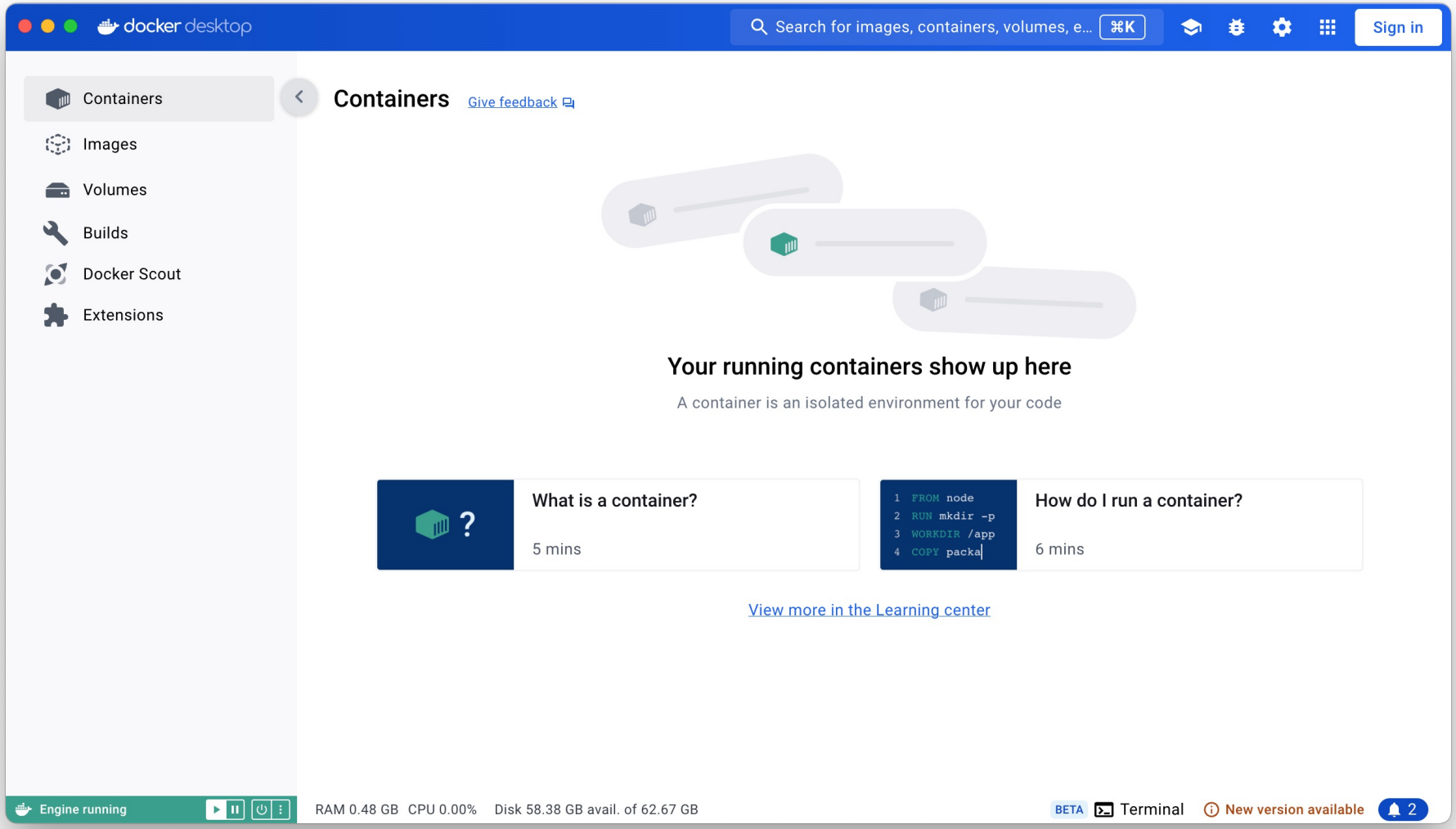
Install Docker

How to install Docker?



- Go to the webpage
  https://www.docker.com/products/docker-desktop/
- Download Docker Desktop ; choose the version appropriate for your OS (Mac, Windows, Linux)
- Install Docker Desktop
- Open Docker Desktop
  - Accept “Docker Subscription Service Agreement”
  - Use recommended settings (requires password)
- It is not required to sign in the application
- You can skip the survey
- You may have to restart your computer
- Start Docker Desktop

Docker Desktop



Create your first container

- Open a terminal
- Create a folder « BigData » where you wish

```
mkdir BigData
```

- Create a subfolder « Lab0 » and go inside it

```
cd BigData
mkdir Lab0
cd Lab0
```

- Edit and save the textfile « Dockerfile » in « Lab0 » with the content

```
FROM ubuntu:22.04
ENTRYPOINT ["/bin/sh", "-c", "echo 'It works!'"]
```

- Build your first Docker image by running the command

```
docker build -t linux-docker .
```

- Run your first container based on this image

```
docker run linux-docker
```

- In the terminal, you should read « It works! »

Create your second container (for Lab 1)

- Open a terminal
- Create a subfolder « Lab1 » inside the folder « BigData » and go inside it

```
mkdir Lab1
cd Lab1
```

- Copy the textfile « Dockerfile » from the footnote url to « Lab1 »
- Build your first image

```
docker build -t lab1:v1 .
```

- Run your first contained based on this image

```
docker run -it -p 8888:8888 lab1:v1 /usr/bin/bash
```

- Open a brower and go to

```
http://127.0.0.1:8888/lab?token=datatoken
```

https://raw.githubusercontent.com/lionel-fillatre/BigData/main/Lab1/Dockerfile

Some useful Docker commands

- List all your images

```
docker images
```

- To view the list of running containers

```
docker ps
```

- Remove all unused containers, networks, images (both dangling and unused), and optionally, volumes

```
docker system prune
```

https://docs.docker.com/reference/cli/docker/

Recommandation

- **First installation**
  - We can help you (if possible) during the lab to finish/test your installation
  - **BUT you must have downloaded Docker desktop** (because the download time might be very long in the lab room)

- **Before each lab**
  - Before each lab, **build the image the day before** at home with high speed internet connexion and time!
  - Some builds need more than 10 minutes, even 20 minutes!

Conclusion

Conclusion

- Big Data frameworks are composed of many tools.
- Labs on Big Data Technology will studied some of these tools.
- Tools will run inside containers.
- Containers can be deployed from a specific image on many platforms, such as workstations, virtual machines, public cloud, etc.
- Containers are extremely popular, and their popularity is growing.
- Multiple containers can be managed by a service called Kubernetes (not studied in this course).