BIG DATA PROGRAMMING AND MANAGEMENT (CSC 343)

Dr. Garrett Dancik

What is Big Data?

- Datasets that are too “large” or “complex” to be stored and analyzed in traditional ways
  - Typically includes distributed data spread across multiple computers (nodes)
- Generated by scientific studies, technology, and commerce
- Examples from technology:
  - https://www.domo.com/blog/data-never-sleeps-6/
  - http://www.internetlivestats.com/
- Understanding digital memory: https://www.makeuseof.com/tag/memory-sizes-gigabytes-terabytes-petabytes/

Big Data Examples

- What does Target know about you?
  - http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html
- https://www.mrc-productivity.com/blog/2015/06/7-real-life-use-cases-of-hadoop/
- https://content.pivotal.io/blog/20-examples-of-roi-and-results-with-big-data

Cloudera and Hadoop

- **Apache Hadoop** (https://hadoop.apache.org/) is "a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models."

  "The name my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless, and not used elsewhere. Those are my naming criteria. Kids are good at generating such."
  http://www.balasubramanyamlanka.com/origin-of-the-name-hadoop/

- **Cloudera CDH**, or *Cloudera's Distribution Including Apache Hadoop*, is 100% open source, heavily tested and widely used. (https://www.cloudera.com/)

The Hadoop Ecosystem

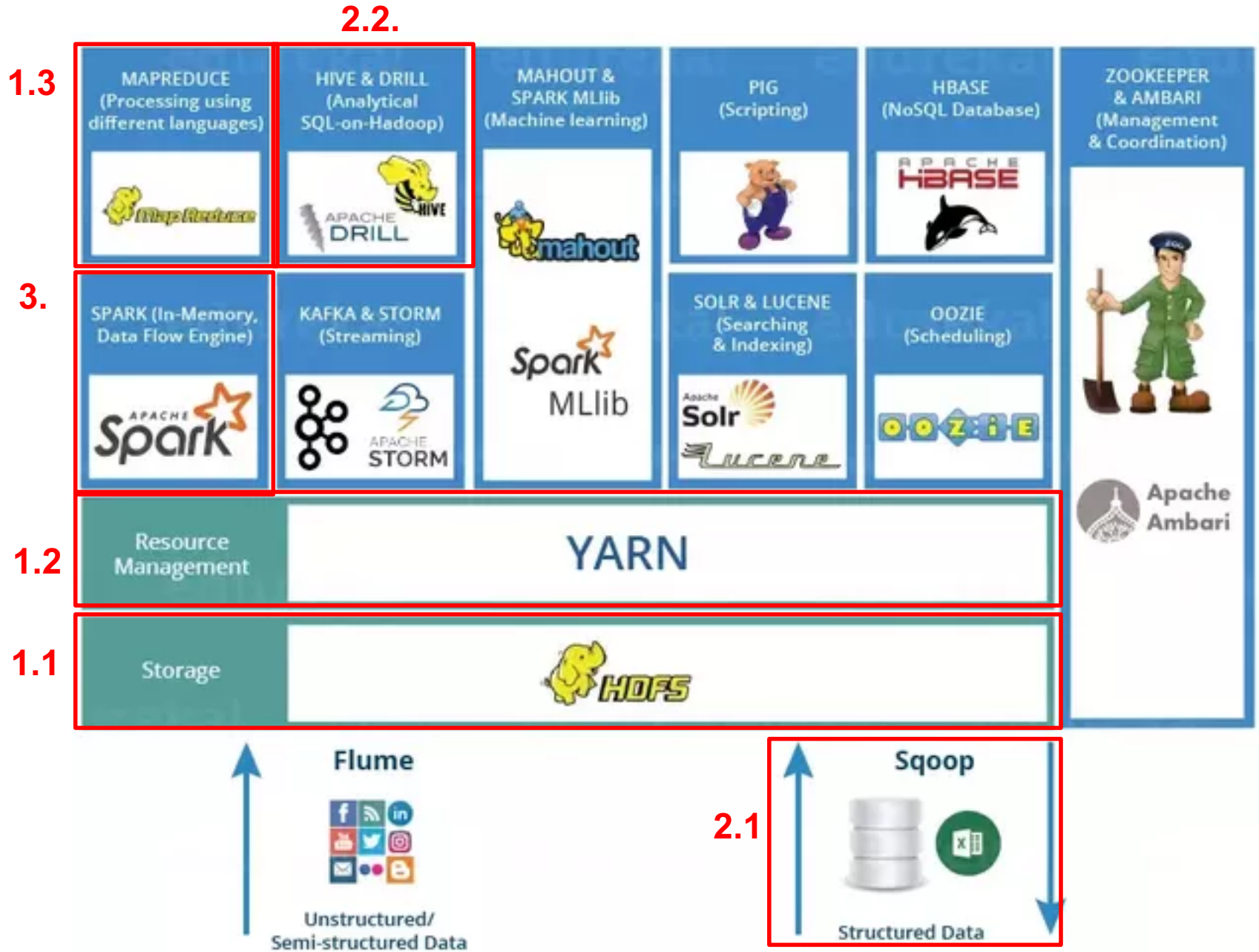

Image posted by Shubham Sinha at quora.com

Docker

- Docker (https://www.docker.com/) provides container images which are a “lightweight, standalone, executable package of software that includes everything needed to run an application”.
- A docker *image* defines a *container* that is produced from an image at run time.
- In this class we will (a slightly modified version of) Cloudera’s quickstart docker container (https://hub.docker.com/r/cloudera/quickstart/)

Docker examples

- Make sure that docker is running before running the commands below from the command line
- Pull the *centos* image, which contains the centos OS
  - `docker pull centos`
- List the images that are available on our machine
  - `docker images`
- Create a new container and output “hello world”, by running the command below. Here the image name is *centos* and the command after it (*echo “hello world”*) is the command we wish to run inside the container.
  - `docker run centos echo "hello world"`

Docker examples (con't)

- To see a list of running containers, type
  - `docker ps`
- To see a list of all containers, type
  - `docker ps –a`
- To remove a stopped container type the following, where name is the container *NAME* or the CONTAINER ID
  - `docker rm name`
- You can remove all stopped containers by typing
  - `docker system prune`
- Now let's run a bash shell in a new container, using the *–it* flags to indicate we wish to run the command in an *interactive terminal*
  - `docker run –it centos bash`

Docker examples (con't)

- Create a file in the container by following the in-class instructions, then exit the container by typing *exit*
- Find the name of this container (how?)
- Create another container from the centos image, and run a bash shell, by using the command from the previous slide, and note that changes made to the container are not saved
- However, the container that includes the file still exists, though it is stopped.
- Start the container using the following command (where *name* is the name or id of the container)
  - `docker start name`
- Now execute a bash shell in the running container, using
  - `docker exec –it name bash`

Docker examples (con't)

- To save changes, you need to *commit* changes from a container to an image. In the command below the arguments must be lowercase and are
  - *name* - the container name or id
  - *username* – your Docker Hub username, required for pushing your image to the cloud (but otherwise optional)
  - *image* - the name of the *image*,
  - *tag* - an optional tag for the image

  - `docker commit name username/image:tag`
- Now run a bash shell in a new container from your saved image to confirm that the changes have been saved.
- To push an image to your *dockerhub*, use
  - `docker login`
  - `docker push username/image:tag`