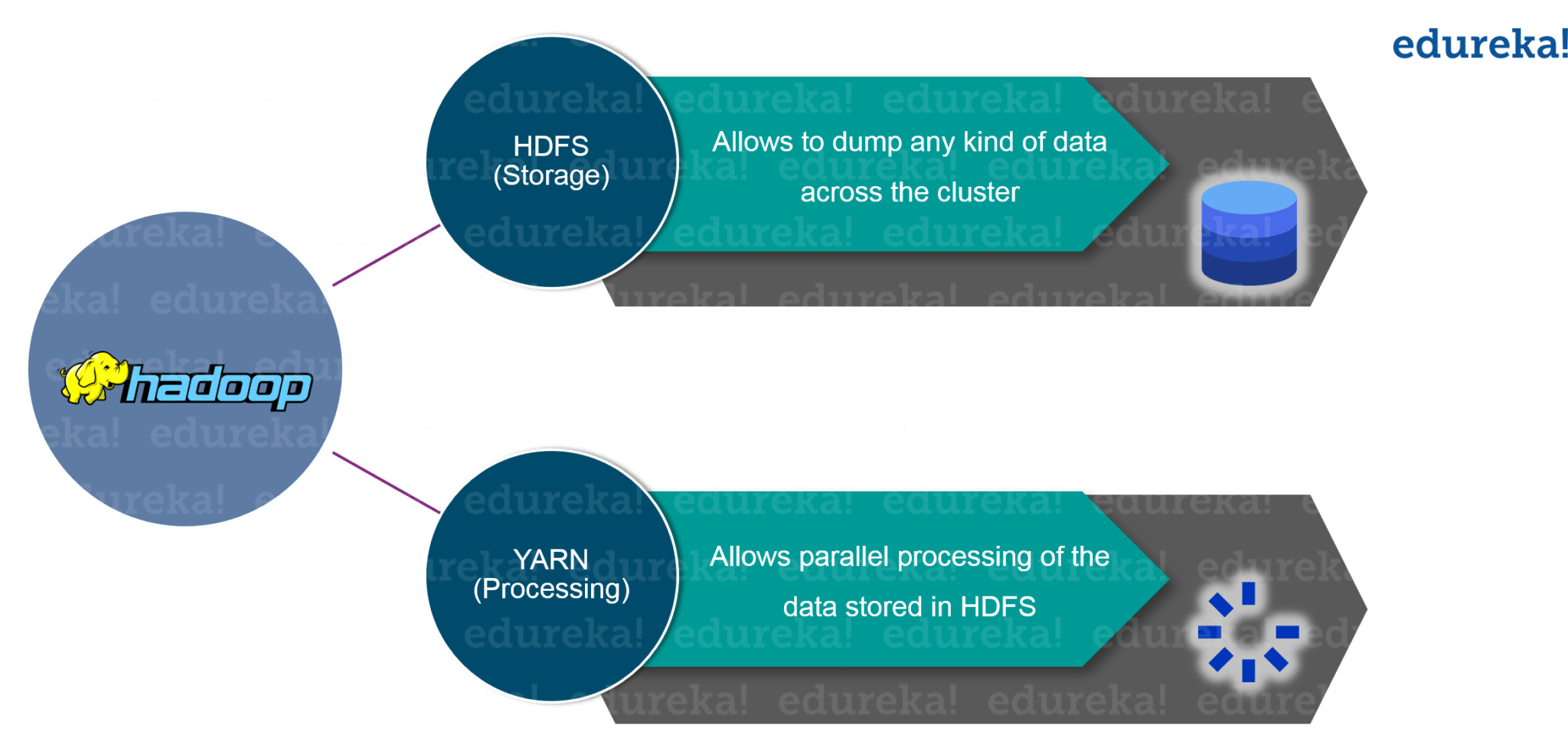
**What is Hadoop?**

Hadoop is a framework that allows you to first store Big Data in a distributed environment, so that, you can process it parallely. There are basically two components in Hadoop:



The first one is ***HDFS*** for storage (Hadoop distributed File System), that allows you to store data of various formats across a cluster. The second one is ***YARN***, for resource management in Hadoop. It allows parallel processing over the data, i.e. stored across HDFS.

**When to use Hadoop ?**

Hadoop is used for:

* *Search* – Yahoo, Amazon, Zvents
* *Log processing* – Facebook, Yahoo
* *Data Warehouse* – Facebook, AOL
* *Video and Image Analysis* – New York Times, Eyealike

**When not to use Hadoop ?**

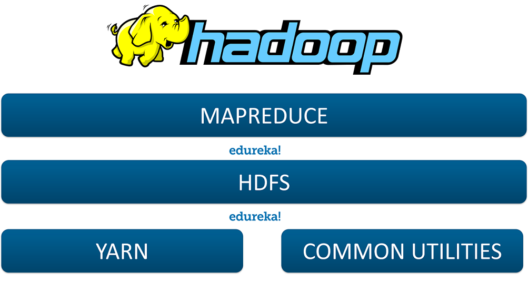
Following are some of those scenarios :

* *Low Latency data access* : Quick access to small parts of data
* *Multiple data modification* : Hadoop is a better fit only if we are primarily concerned about reading data and not modifying data.
* *Lots of small files* : Hadoop is suitable for scenarios, where we have few but large files.

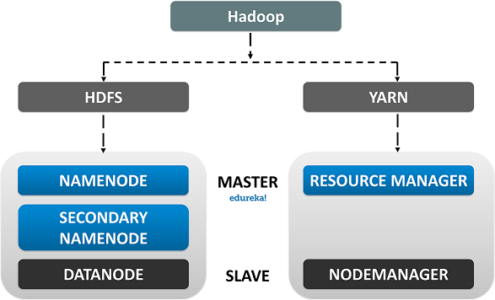
**Components of Hadoop?**

The Core Components of Hadoop are as follows:

* [**MapReduce**](https://www.edureka.co/blog/mapreduce-tutorial/)
* [**HDFS**](https://www.edureka.co/blog/hdfs-tutorial)
* [**YARN**](https://www.edureka.co/blog/hadoop-yarn-tutorial/)
* **Common Utilities**



**Hadoop Architecture**



**MapReduce:** It is a Software Data Processing model designed in Java Programming Language. MapReduce is a combination of two individual tasks, namely:

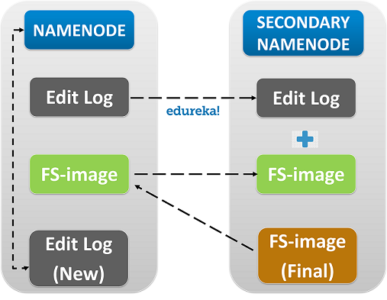
* **Map:** It takes data and set then divides it into chunks such that they are converted into a new format which would be in the form of a key-value pair.
* **Reduce:** It is the second part where the **Key/Value** pairs are reduced to tuples.

The **MapReduce** process enables us to perform various operations over the big data such as **Filtering** and **Sorting** and many such similar ones.

**HDFS** is the primary storage unit in the Hadoop Ecosystem. The HDFS is the reason behind the quick data accessing and generous Scalability of Hadoop.

The HDFS comprises the following components.

* **NameNode**
* **DataNode**
* **Secondary NameNode**

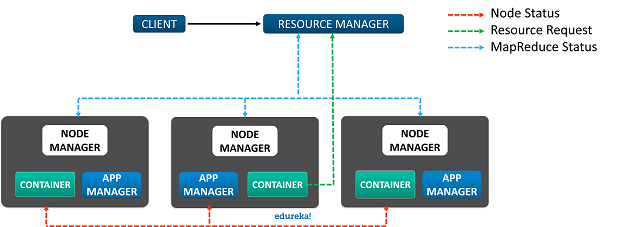


Let us Discuss each one of them in detail.

* **Name node:** The Name Node is the centralized piece of the HDFS. It known as the *Master* and it is designed to store the *Meta Data.* Name Node is responsible for monitoring the *Health* *Status* of the *Slave* *Nodes* and to assign *Tasks* to the Data Nodes.

* **Data Node:** Data Node is the actual unit which stores the data. It is known as the *Slave* and it responds to the Name Node about its Health Status and the task status in the form of a *Heartbeat.* If the Data Node fails to respond to the Name Node, then the Name Node considers the Slave Node to be Dead and reassigns the task to the Next available Data Node.

* **Secondary Name Node:** As the name speaks, the Secondary Name Node is not a backup of the name node. It acts as a *Buffer* to the Name Node. It stores the intermediate updates the *FS-image* of the Name Node in the *Edit-log* and updates the information to the*FinalFS-image* when the name node is inactive.



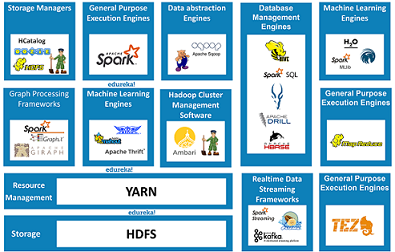
The **YARN** or **Yet Another Resource Negotiator** is the update to **Hadoop** since its second version. It is responsible for**Resource management** and**Job Scheduling**. Yarn comprises of the following components:

* **Resource Manager:** It is the core component of Yarn and is considered as the *Master****,*** responsible for providing generic and flexible frameworks to administer the computing resources in a *Hadoop Cluster.*
* **Node Manager:** It is the *Slave* and it serves the *ResourceManager.* Node Manager is assigned to all the Nodes in a Cluster. The main responsibility of the Node Manager is to monitor the Status of the Container and App Manager.
* **App Manager:** It manages data processing in the Container and requests the Container resources from the Resource Manager.
* **Container:** Container is where the actual data processing takes place.

**Hadoop Ecosystem**

The Components in the Hadoop Ecosystem are classified into:

* [**Storage**](https://www.edureka.co/blog/every-hadoop-component/#storage)
* [**General Purpose Execution Engines**](https://www.edureka.co/blog/every-hadoop-component/#general)
* [**Database Management Tools**](https://www.edureka.co/blog/every-hadoop-component/#database)
* [**Data Abstraction Engines**](https://www.edureka.co/blog/every-hadoop-component/#abstraction)
* [**Real-Time Data Streaming**](https://www.edureka.co/blog/every-hadoop-component/#realtime)
* [**Graph-Processing Engines**](https://www.edureka.co/blog/every-hadoop-component/#graph)
* [**Machine Learning**](https://www.edureka.co/blog/every-hadoop-component/#ml)
* [**Cluster Management**](https://www.edureka.co/blog/every-hadoop-component/#cluster)



**Data Storage**

**Hadoop Distributed File System,** it is responsible for **Data Storage.** It provides Distributed data processing capabilities to Hadoop. HDFS is **Fault Tolerant, Reliable** and most importantly it is generously **Scalable.**

* + - * + HDFS supports both **Vertical** and **Horizontal** Scalability.
        + HDFS has this unique **Replication Factor** which solves the issue of unexpected Data Loss.

**HCATALOG** is a **Table Management** tool for Hadoop. It provides tabular data store of **HIVE** to users such that the users can perform operations upon the data using the advanced data processing tools such as the **Pig, MapReduce** etc.

* + - * + Apart from **Interpreting** and **Translating,** HCatalog can Store information in **Binary Format.**
        + HCatalog offers **Kerberos** based **Authentication.**

[**Zookeeper**](https://www.edureka.co/community/1106/what-zookeeper-what-the-purpose-zookeeper-hadoop-ecosystem)is known as the centralized Open Source server responsible for managing the configuration information, naming conventions and synchronisations for Hadoop clusters.

**What is MR?**

<https://netjs.blogspot.com/2018/02/how-mapreduce-works-in-hadoop.html>

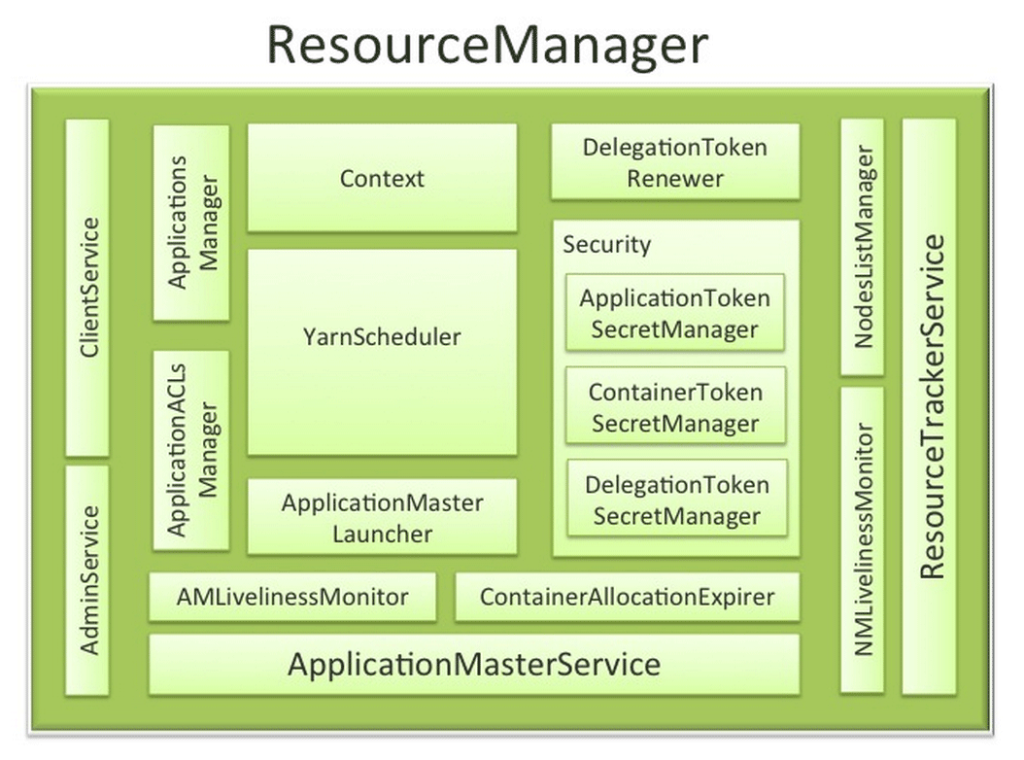
* By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing’s logic and helps to write applications which transform big data sets into a manageable one.
* MapReduce makes the use of two functions i.e. Map() and Reduce() whose task is:
  1. *Map()* performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair based result which is later on processed by the Reduce() method.
  2. *Reduce()*, as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples.

**No of I/p splits?**

<https://netjs.blogspot.com/2018/05/input-splits-in-hadoop.html>

**Resource manager?**

The Resource Manager is the core component of [**YARN**](http://data-flair.training/blogs/hadoop-yarn-tutorial/)**– Yet Another Resource Negotiator**. In analogy, it occupies the place of JobTracker of MRV1. Hadoop YARN is designed to provide a generic and flexible framework to administer the computing resources in the Hadoop cluster. In this direction, the YARN Resource Manager Service (RM) is the central controlling authority for resource management and makes allocation decisions ResourceManager has two main components: Scheduler and ApplicationsManager.

[](https://d2h0cx97tjks2p.cloudfront.net/blogs/wp-content/uploads/sites/2/2016/06/Resource-Manager.png)

*Hadoop YARN Resource Manager*

The **Scheduler** API is specifically designed to negotiate resources and not schedule tasks. The scheduler does not perform monitoring or tracking of status for the Applications. The Scheduler performs its scheduling function based the resource requirements of the applications; it does so base on the abstract notion of a resource Container which incorporates elements such as memory, CPU, disk, network etc.  
Hadoop Yarn Resource Manager does not guarantee about restarting failed tasks either due to application failure or hardware failures. Applications can request resources at different layers of the cluster topology such as nodes, racks etc. Hence, the scheduler determines how much and where to allocate based on resource availability and the configured sharing policy.

The Scheduler has a pluggable policy plug-in, which is responsible for partitioning the cluster resources among the various queues, applications etc. The current [**Map-Reduce**](http://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/) schedulers such as the CapacityScheduler and the FairScheduler would be some examples of the plug-in ApplicationsManager is responsible for maintaining a collection of submitted applications. It accepts a job from the client and negotiates for a container to execute the application specific ApplicationMaster and it provide the service for restarting the ApplicationMaster in the case of failure.  
It also keeps a cache of completed applications so as to serve users’ requests via web UI or command line long after the applications in question finished. Though the above two are the core component, for its complete functionality the Resource Manager depend on various other components

**Where it resides ?**

Similar to data residing in a local file system of a personal computer system, in Hadoop, data resides in a distributed file system which is called as a **Hadoop Distributed File system**. The processing model is based on **'Data Locality'** concept wherein computational logic is sent to cluster nodes(server) containing data. This computational logic is nothing, but a compiled version of a program written in a high-level language such as Java. Such a program, processes data stored in Hadoop HDFS

#### **NameNode and DataNodes**

<https://sites.google.com/site/amitsciscozone/home/hadoop/hadoop-components>

Hadoop comes with a filesystem and since it manages the storage of files across several machines, it is called *Hadoop Distributed FileSystem (HDFS)*. In HDFS, large files are broken down into smaller blocks (64MB, by default) which are stored as independent units. The size of the block is configurable but most implementations use default value (or sometimes changed to 128MB) to allow relatively low-latency disk reads and writes.

To ensure against corrupted block/disk or machine failure, each block is replicated to physically separate machines. The default replication factor is 3 but this is also configurable. The blocks on the failed machine is replicated back from alternative location to other machines.

HDFS cluster has two types of nodes operating in master-slave relationship: *NameNode (master) and DataNodes (slaves)*. The NameNode maintains the filesystem metadata of HDFS; it keeps track of all files that are broken down into blocks and which DataNodes store these blocks. All transaction logs are stored in a file called **EditLog** which is stored on the local disk. The entire filesystem namespace, including the mapping of blocks to files and filesystem properties, are stored in a file called **FsImage** which is also stored on the local disk.

The DataNodes are the real workhorses of HDFS; they store and retrieve blocks when they receive instructions from *Clients* or NameNode. A DataNode may communicate with other DataNodes to replicate blocks for redundancy. Upon initialization, each DataNode reports back to the NameNode the list of blocks they are storing. After this, the DataNodes continually poll NameNode to provide information regarding local changes as well as receive instructions to create, move or delete blocks from local disk.

#### **Secondary NameNode**

There is only one NameNode in the cluster and it is the most critical component of the Hadoop architecture as it maintains HDFS for the cluster, monitors the health of DataNodes and co-ordinates access to data. Without the NameNode, the filesystem cannot be used and all the files would be lost as there is no way to reconstruct the files from the blocks on the DataNodes. Hence, the NameNode should be made resilient to failure with good server configuration, redundant power supplies, fans, NICs, etc. Hadoop has two mechanisms-

The first option is to backup the files that make up the persistent state of filesystem metadata (these files are namespace and edit log). A common configuration is to write to local disk and remote disk mount.

The second option is to run a *Secondary NameNode*. The Secondary NameNode isn't actually part of HDFS, but it communicates with the NameNode on a periodic interval (configurable value) to take the snapshot of the HDFS metadata. This information is used in case of NameNode failure. However, the information on Secondary NameNode is normally not up-to-date and some data loss is almost certain.

**High availability?**

<https://data-flair.training/blogs/hadoop-high-availability-tutorial/>

**Services run on NN?**

ps -ef | grep hadoop | grep -P 'namenode|datanode|tasktracker|jobtracker'

./hadoop dfsadmin -report, Sudo jps

**What is sqoop?**

Sqoop − “**SQ**L to Had**oop** and Hadoop to SQL”  
While it comes to transfer data between Hadoop and relational database servers, Sqoop is the best tool. To be more specific, we use it to import data from relational databases such as MySQL, Oracle to Hadoop [**HDFS**](https://data-flair.training/blogs/hadoop-hdfs-tutorial/), and export from the Hadoop file system to relational databases. Basically, it is provided by the Apache Software Foundation.  
Moreover, Sqoop uses two main tools. Like:

1. [**Sqoop import**](https://data-flair.training/blogs/sqoop-import/) (Copy data from RDBMS to HDFS)
2. [**Sqoop export**](https://data-flair.training/blogs/sqoop-export/) (Copy data from HDFS to RDBMS)

**Parameter used for table in sqoop?**

<https://data-flair.training/blogs/sqoop-import-all-tables/>

**How to import data in avro file using sqoop? Which parameter we use?**

<https://docs.cloudera.com/documentation/enterprise/5-4-x/topics/cdh_ig_sqoop.html>

**Can we direct import data from RDBMS to Hive?**

<https://acadgild.com/blog/sqoop-import-data-from-mysql-to-hive>

<https://docs.cloudera.com/HDPDocuments/HDP3/HDP-3.1.5/migrating-data/content/hive_import_rdbms_data_into_hive.html>

**What is hive?**

Hive is a tool in Hadoop ecosystem which provides an interface to organize and query data in a databse like fashion and write SQL like queries. It is suitable for accessing and analyzing data in Hadoop using SQL syntax.

**Different file system in hive?**

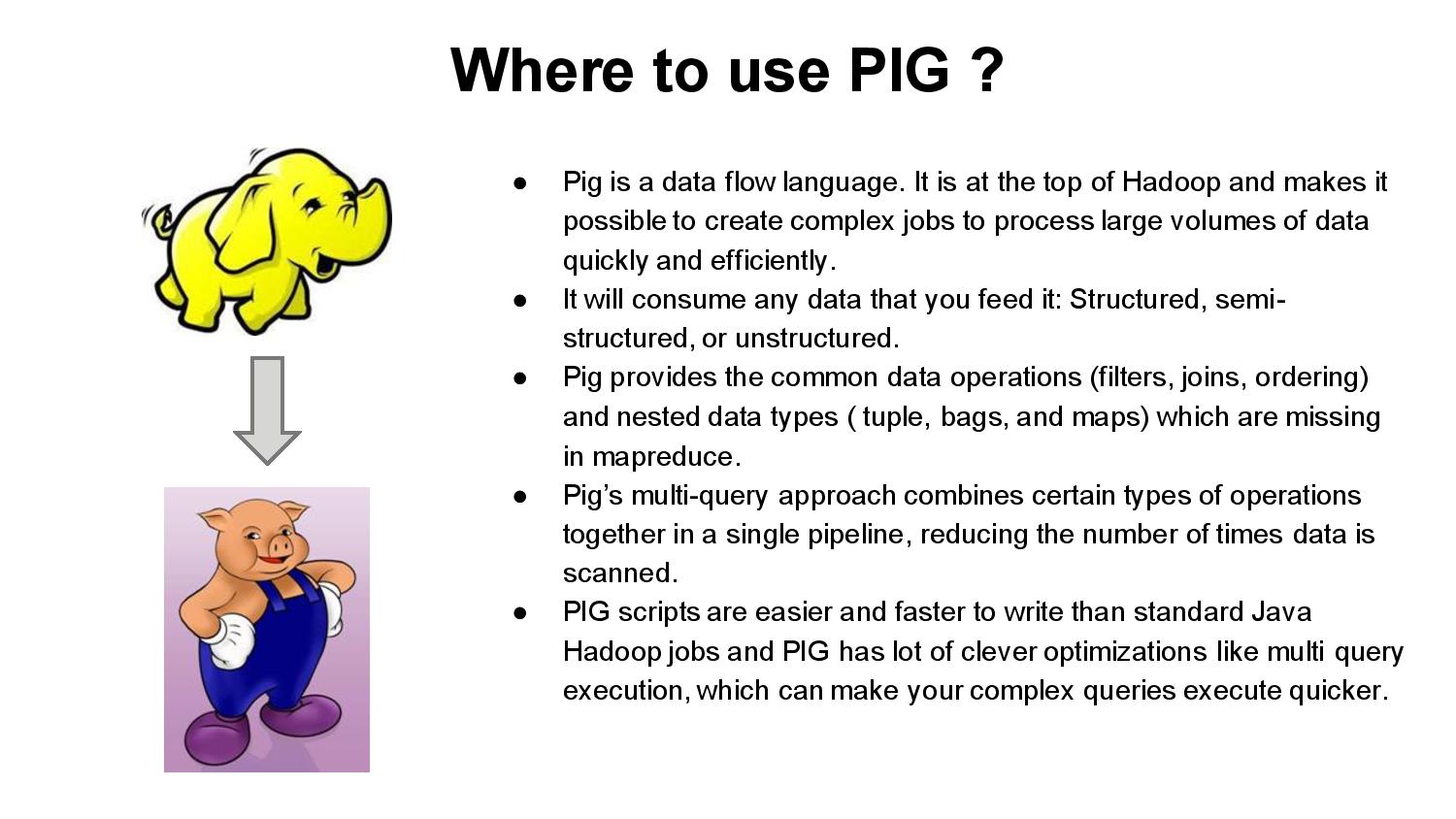
<https://dwgeek.com/hive-different-file-formats-text-sequence-rc-avro-orc-parquet-file.html/>

**How we load data to into hive table?**

<http://www.hadooplessons.info/2014/12/loading-data-into-hive-table.html>

<https://community.cloudera.com/t5/Support-Questions/How-to-load-data-into-hive-table-with-existing-table-without/td-p/181869>

**What is pig?**



**Loading data using pig?**

<https://www.tutorialspoint.com/apache_pig/apache_pig_reading_data.htm>

<https://www.tutorialspoint.com/apache_pig/apache_pig_storing_data.htm>

<https://mapr.com/docs/60/Hive/RetrieveDataFromPig.html>

**What is Hbase?**

<https://www.edureka.co/blog/interview-questions/hbase-interview-questions/>

**What is Flume?**

<https://data-flair.training/blogs/flume-interview-questions/>

<https://mindmajix.com/apache-flume-interview-questions>

**Zookeeper?**

<https://data-flair.training/blogs/zookeeper-interview-questions/>

<https://www.corejavaguru.com/bigdata/zookeeper/interview-questions-part-1>

<https://www.wisdomjobs.com/e-university/apache-zookeeper-interview-questions.html>

**Do you know MySQL?**

<https://career.guru99.com/top-50-mysql-interview-questions-answers/>

<https://www.javatpoint.com/mysql-interview-questions>

<https://www.softwaretestinghelp.com/mysql-interview-questions/>

**MySQL version?**

The version of the installed MySQL server can be found out easily by running the following command from the MySQL prompt.

**mysql> SHOW VARIABLES LIKE “%version%”;**

**Which are those files when we configure Hadoop?**

<https://data-flair.training/forums/topic/what-are-the-configuration-files-in-hadoop/>

<https://www.edureka.co/blog/hadoop-cluster-configuration-files/>

**What is Solr?**

<https://www.tutorialspoint.com/apache_solr/apache_solr_on_hadoop.htm>

<https://developer.ibm.com/hadoop/2016/01/01/indexing-hadoop-docs-solr/>

**Which platform you are working on?**

**What configuration you did?**

**Have you created job?**

**Have you done that twitter job?**

**What lines about zookeeper are displayed when we run sqoop command?**

**Which engine u have used?**

**What is triggers ?**

**Which table you use to check all data?**

**Syntax for trigger?**

**What business logic you have implemented in Ur project?**

**Apache binary configuration**