Cloudera

CDH

- Cloudera Distribution including Hadoop
- CDH is Cloudera’s open source platform distribution, including Apache Hadoop and built specifically to meet enterprise demands.
- CDH delivers everything that an enterprise needs for BigData workflows right out of the box.
- CDH integrated Hadoop with more than a dozen other critical open source projects.
- Current Version is 5.5.2

CDH Top Components

Apache Hadoop (Core) Reliable, scalable distributed storage and computing

Apache Accumulo A secure, distributed data store to serve performance-intensive Big Data applications

Apache Flume For collecting and aggregating log and event data and real-time streaming it into Hadoop

Apache HBase Scalable record and table storage with real-time read/write access

Apache Hive Familiar SQL framework with metadata repository for batch processing of Hadoop data

HUE The extensible web GUI that makes Hadoop users more productive

Impala The analytic database native to Hadoop for low-latency queries under multi-user workloads

Apache Kafka The backbone for distributed real-time processing of Hadoop data

Apache Pig High-level data flow language for processing data stored in Hadoop

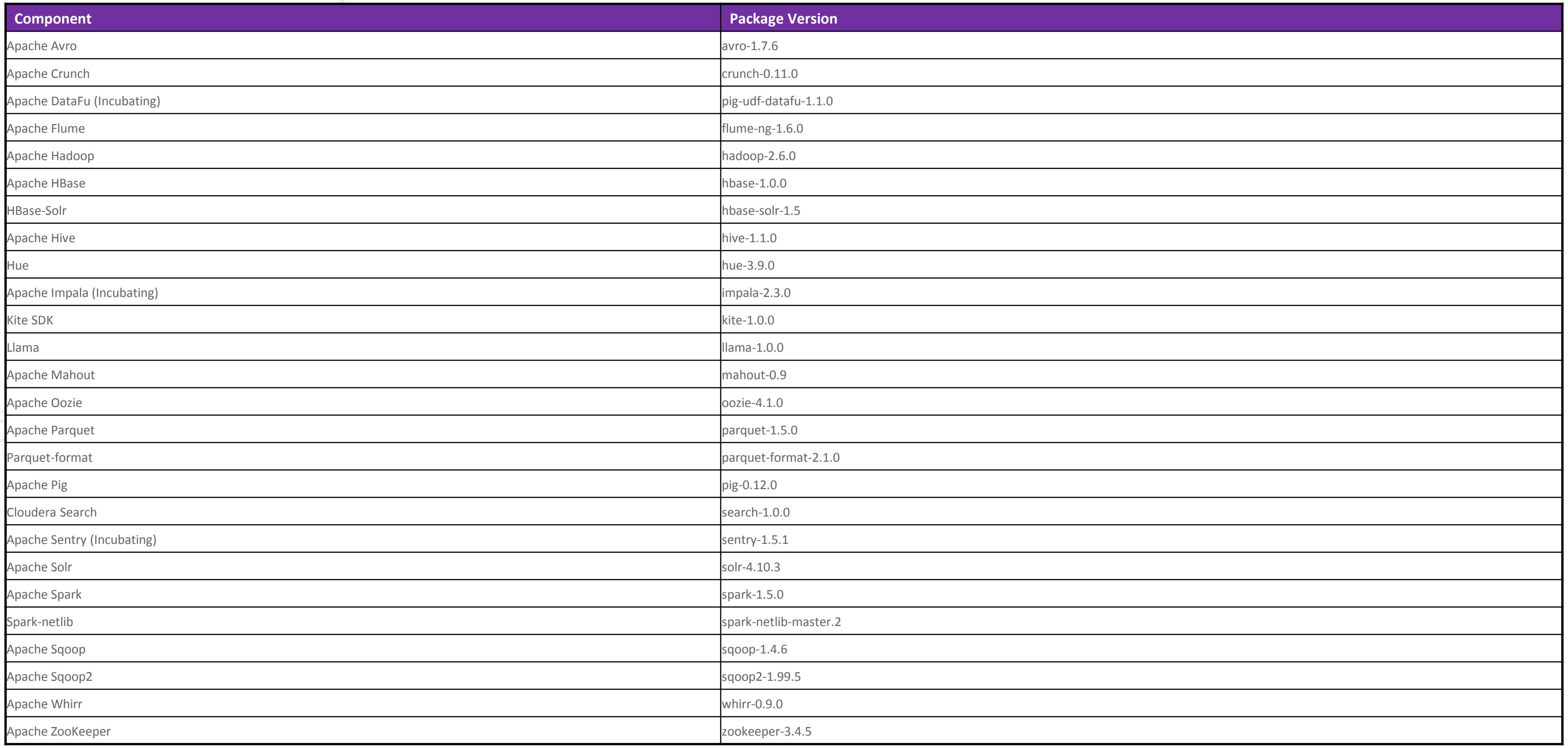
Apache Sentry Fine-grained, role-based authorization for Impala and Hive (incubating)

Cloudera Search Powered by Solr to make Hadoop accessible to everyone via integrated full-text search

Apache Spark The open standard for in-memory batch and real-time processing for advanced analytics

Apache Sqoop Data transport engine for integrating Hadoop with relational databases

All CDH5.5 components and versions



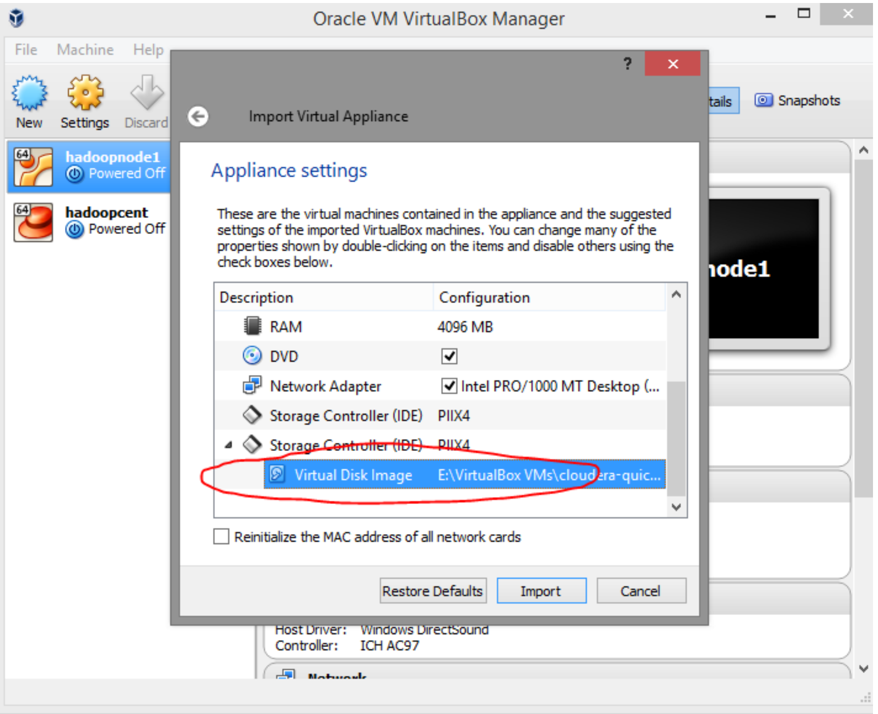
Component	Package Version
Apache Avro	avro-1.7.6
Apache Crunch	crunch-0.11.0
Apache DataFu (Incubating)	pig-udf-datafu-1.1.0
Apache Flume	flume-ng-1.6.0
Apache Hadoop	hadoop-2.6.0
Apache HBase	hbase-1.0.0
HBase-Solr	hbase-solr-1.5
Apache Hive	hive-1.1.0
Hue	hue-3.9.0
Apache Impala (Incubating)	impala-2.3.0
Kite SDK	kite-1.0.0
Llama	llama-1.0.0
Apache Mahout	mahout-0.9
Apache Oozie	oozie-4.1.0
Apache Parquet	parquet-1.5.0
Parquet-format	parquet-format-2.1.0
Apache Pig	pig-0.12.0
Cloudera Search	search-1.0.0
Apache Sentry (Incubating)	sentry-1.5.1
Apache Solr	solr-4.10.3
Apache Spark	spark-1.5.0
Spark-netlib	spark-netlib-master.2
Apache Sqoop	sqoop-1.4.6
Apache Sqoop2	sqoop2-1.99.5
Apache Whirr	whirr-0.9.0
Apache ZooKeeper	zookeeper-3.4.5

Setting up CDH

- One can use Cloudera Manager to download the required components in setting up a dist cluster
- For experimentation and learning a single node cluster with all components can be obtained via a QuickStart VM
  - Download QuickStart VM for VirtualBox from - https://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.5.0-0-virtualbox.zip
  - The VM is 4GB in size, hence pick it up from Lab ()
  - If you prefer, you could download the VMWare/Docker versions as well, but VirtualBox is most preferable way if you are on windows.

Cloudera QuickStart VM setup

1. Download and install VirtualBox on your machine: http://virtualbox.org/wiki/
2. Download the Cloudera Quickstart VM.
3. Uncompress the VM archive. Use a tool like Winrar to uncompress.
4. Start VirtualBox and click Import Appliance in the File dropdown menu. Click the folder icon beside the location field. Browse to the uncompressed archive folder, select
5. the .ovf file, and click the Open button. Click the Continue button. Click the Import button.
6. The VM is created in C: drive, if you have less space in C: drive choose an other drive in “Storage Controller” section of the Appliance Setting Dialog.
7. Your virtual machine should now appear in the left column. Select it and click on Start to launch it.
8. To verify that the VM is running and you can access it, open a browser to the URL: http://localhost:8088. You should see the resource manager UI.
9. The VM uses port forwarding for the common Hadoop ports, so when the VM is running, those ports on localhost will redirect to the VM.



Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Description	Configuration
RAM	4096 MB
DVD	☑
Network Adapter	☑ Intel PRO/1000 MT Desktop (...
Storage Controller (IDE)	PIIX4
Storage Controller (IDE)	PIIX4
Virtual Disk Image	E:\VirtualBox VMs\cloudera-quic...

☐ Reinitialize the MAC address of all network cards

When the QuickStart VM starts, a browser is automatically opened to Hue, a GUI interface for Hadoop and many other data tools in CDH. The credentials are:

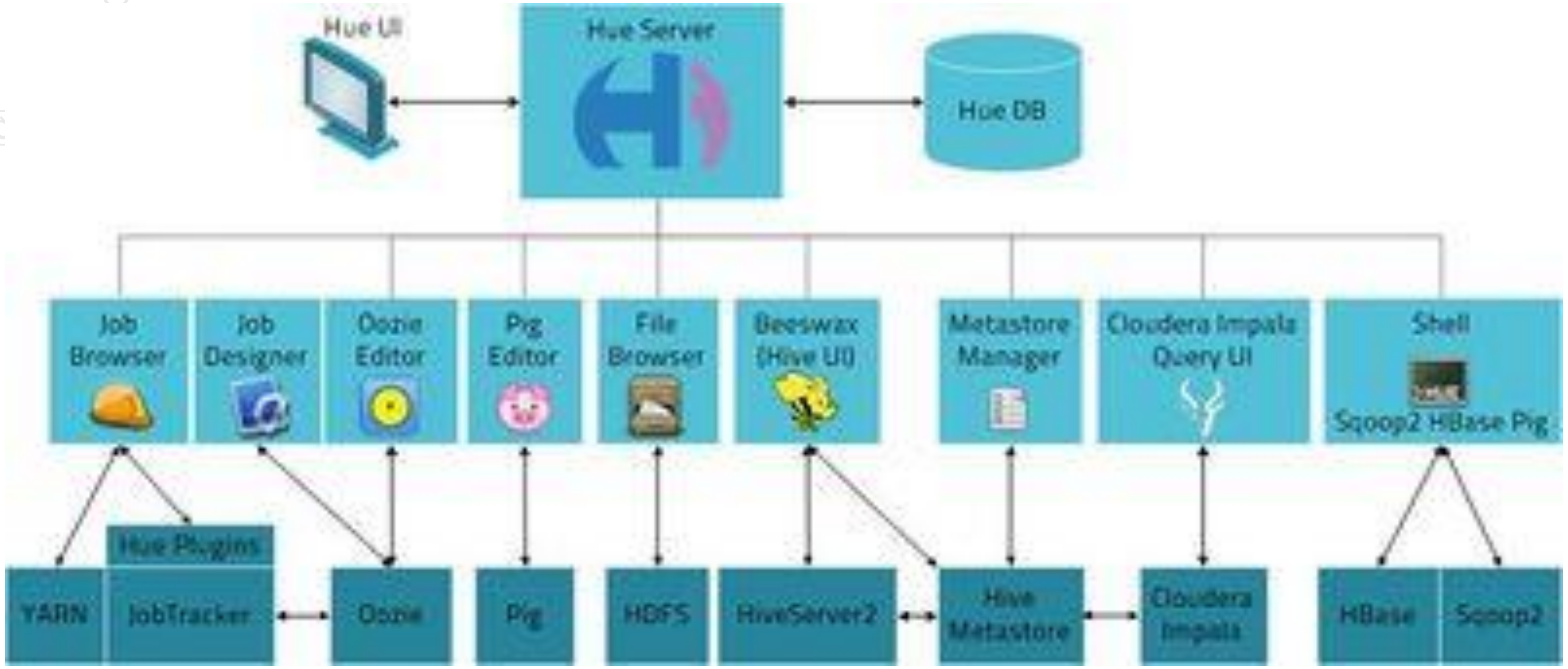
username: cloudera, password: cloudera

You could as well open Hue on your local browser by navigating to http://localhost:8888 (the port is already mapped).

Cloudera Manager

- Administration tool for Cloudera services (the ControlPanel and Monitoring app)
- Two Versions
  - Enterprise (for DataCenters and Clusters)
  - Express (for the QuickStart VM, Has monitoring, collectors and reporting apps in one host)
- You could turn on express Cloudera Manager with the below command, this would require 8GB **Free** RAM on your VM, once done, you could manage your cluster using the web page @ http://localhost:7180
  - sudo /home/cloudera/cloudera-manager --express --force

Hue


Hue UI
Hue Server
Hue DB
Job Browser
Job Designer
Oozie Editor
Pig Editor
File Browser
Beeswax (Hive UI)
Metastore Manager
Cloudera Impala Query UI
Shell
Sqoop2 HBase Pig
Hue Plugins
YARN
JobTracker
Oozie
Pig
HDFS
HiveServer2
Hive Metastore
Cloudera Impala
HBase
Sqoop2

Try

- Some HDFS operations from Hue
- A PigScript from Hue
- A Hive query from Hue

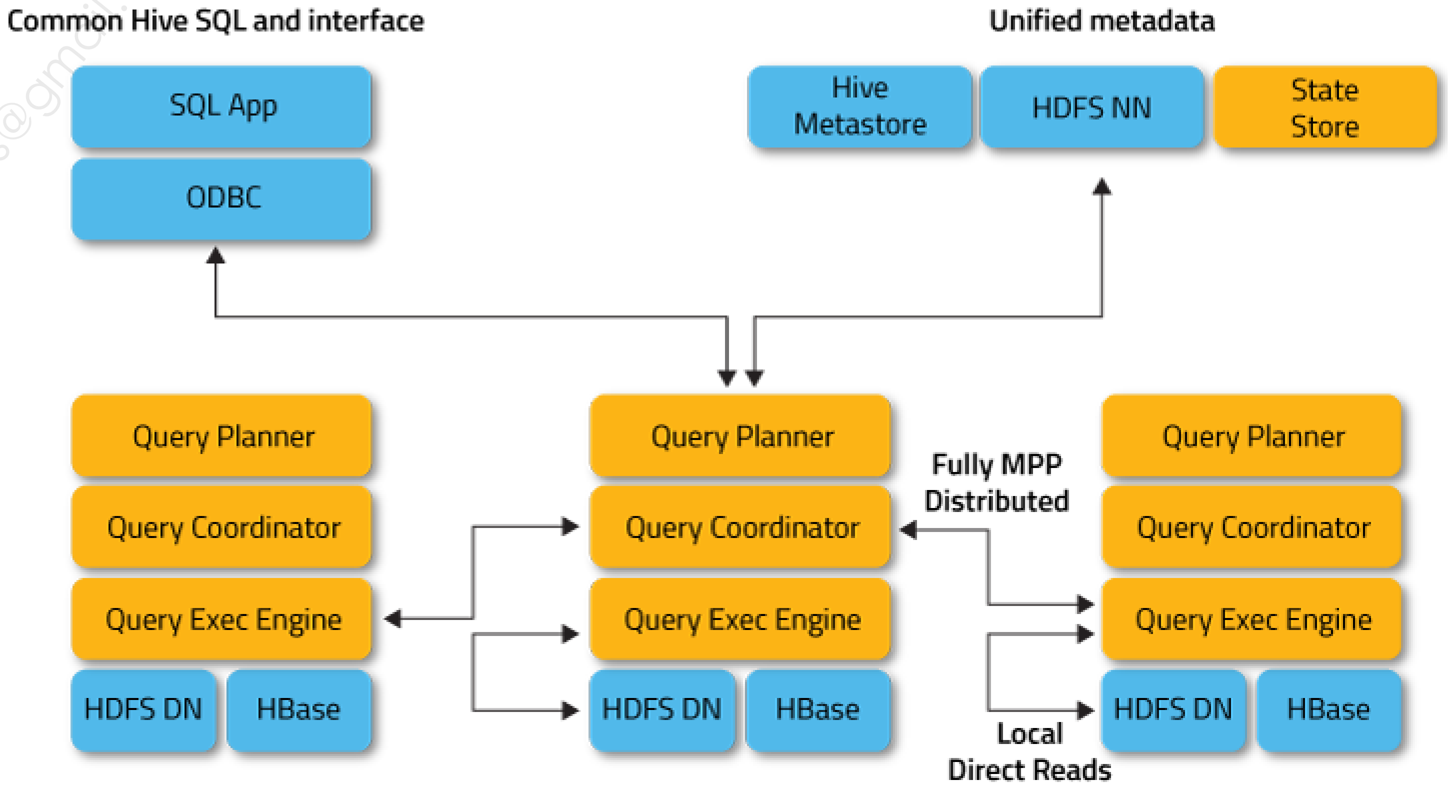
Impala

- Impala is a open source, native analytic MPP database for Apache Hadoop.
- Impala was specifically targeted for integration with standard business intelligence environments, and to that end supports most relevant industry standards: clients can connect via ODBC or JDBC.
- Impala is written with a goal to improve SQL query performance on Apache Hadoop while retaining a familiar user experience.
- Impala uses the same metadata as of Hive (via Metastore), It has the same syntax of Hive SQL, and one could use the same JDBC/ODBC drivers of Hive, thus providing a familiar and unified platform for batch-oriented or near real-time queries.

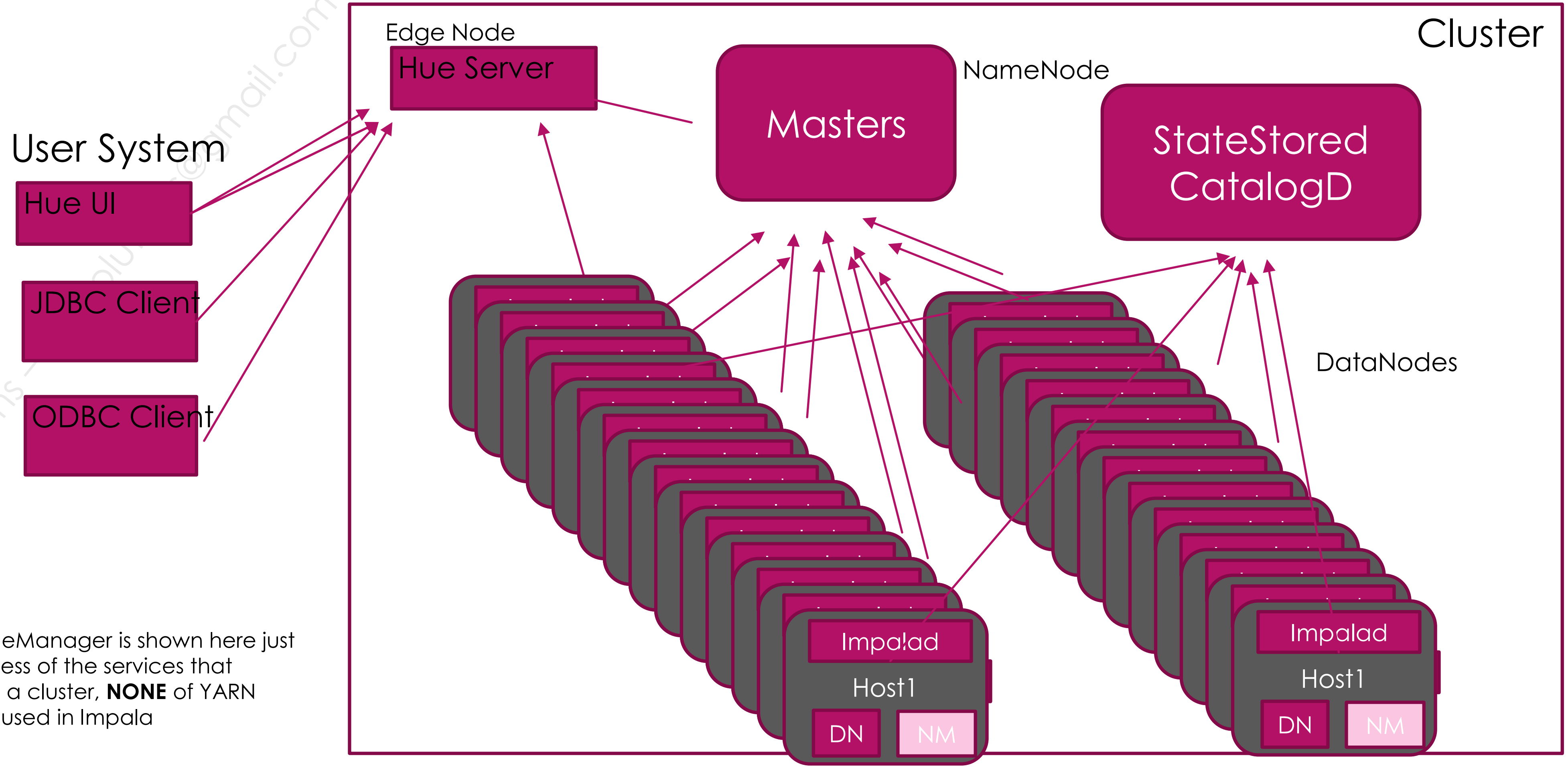
Architecture

- Impala circumvents MapReduce to directly access the data through a specialized distributed query engine that is very similar to those found in commercial parallel RDBMSs. The result is order-of-magnitude faster performance than Hive, depending on the type of query and configuration.
- Impala processes the query locally on the data nodes, thus network bottlenecks are avoided.

Architecture



Impala Physical Architecture



Note that NodeManager is shown here just for completeness of the services that typically run in a cluster, **NONE** of YARN processes are used in Impala

Impala Services

Impala has 3 Services.

- impalad - The executor's, these are run almost on all data nodes. i.e. there could be many impalad processes in a give cluster.
- statestored - One instance of this process is run in the cluster.
- catalogd- One instance of this process is run in the cluster.

Impalad

- impalad - the Impala daemon service is dually responsible for accepting queries from client processes and orchestrating their execution across the cluster, and for executing individual query fragments on behalf of other Impala daemons.

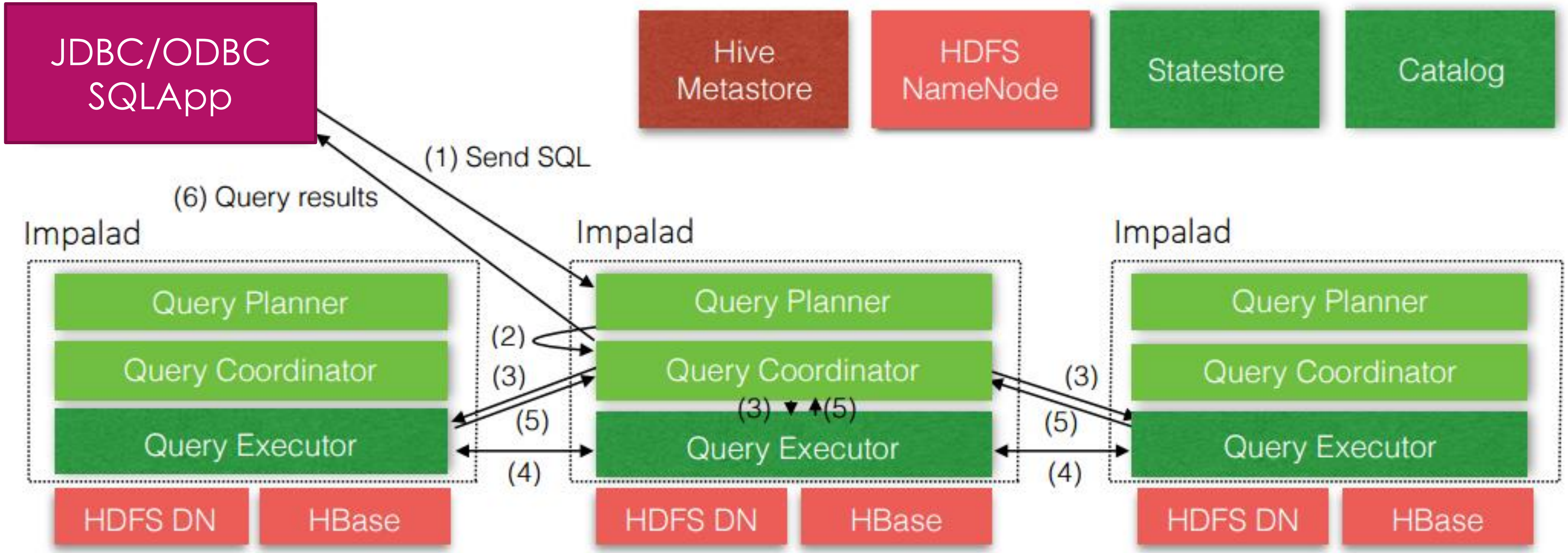
- When an Impala daemon operates in the first role by managing query execution, it is said to be the coordinator for that query. However, all Impala daemons are symmetric; they may all operate in all roles. This property helps with fault-tolerance, and with load-balancing. One Impala daemon is deployed on every machine in the cluster that is also running a datanode process - the block server for the underlying HDFS deployment - and therefore there is typically one Impala daemon on every machine. This allows Impala to take advantage of data locality, and to read blocks from the filesystem without having to use the network.

StateStored and Catalogd processes

- statestored - The Statestore daemon is Impala's metadata publish-subscribe service, which disseminates clusterwide metadata to all Impala processes.

- Catalogd - The Catalog daemon (catalogd), serves as Impala's catalog repository and metadata access gateway. Through the catalogd, Impala daemons may execute DDL commands that are reflected in external catalog stores such as the Hive Metastore. Changes to the system catalog are broadcast via the statestore.

Runtime Arch


JDBC/ODBC SQLApp
Hive Metastore
HDFS NameNode
Statestore
Catalog
(1) Send SQL
(6) Query results
Impalad
Query Planner
Query Coordinator
Query Executor
HDFS DN
HBase
Impalad
Query Planner
Query Coordinator
(3) (5)
Query Executor
HDFS DN
HBase
Impalad
Query Planner
Query Coordinator
Query Executor
HDFS DN
HBase
(2)
(3)
(5)
(4)
(3)
(5)
(4)