HDInsight Hadoop Cluster, HDFS, Spark, Hive

Lab 12

Deep.Azure@McKesson

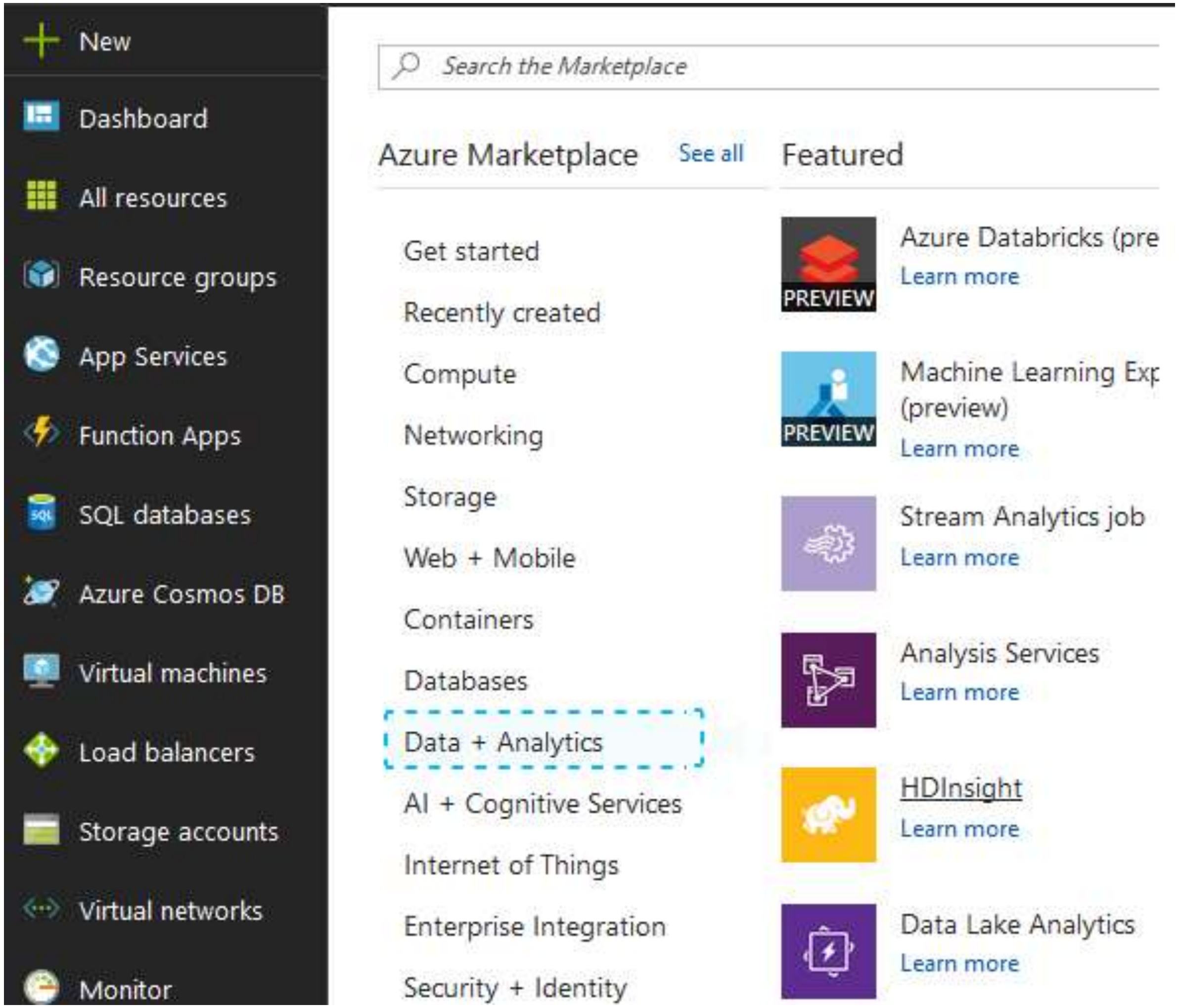
Zoran B. Djordjević

HDInsight Apps run on Clusters

- Hadoop and most applications and frameworks in Hadoop echo system require clusters of machines to run and scale successfully.
- Typically, you can run Hadoop, Spark, Storm, Kafka and most other applications on a cluster with a single node. You do such "pseudo-cluster" runs for development and testing purposes.
- Let us crate a HDInsight cluster that supports Hadoop.

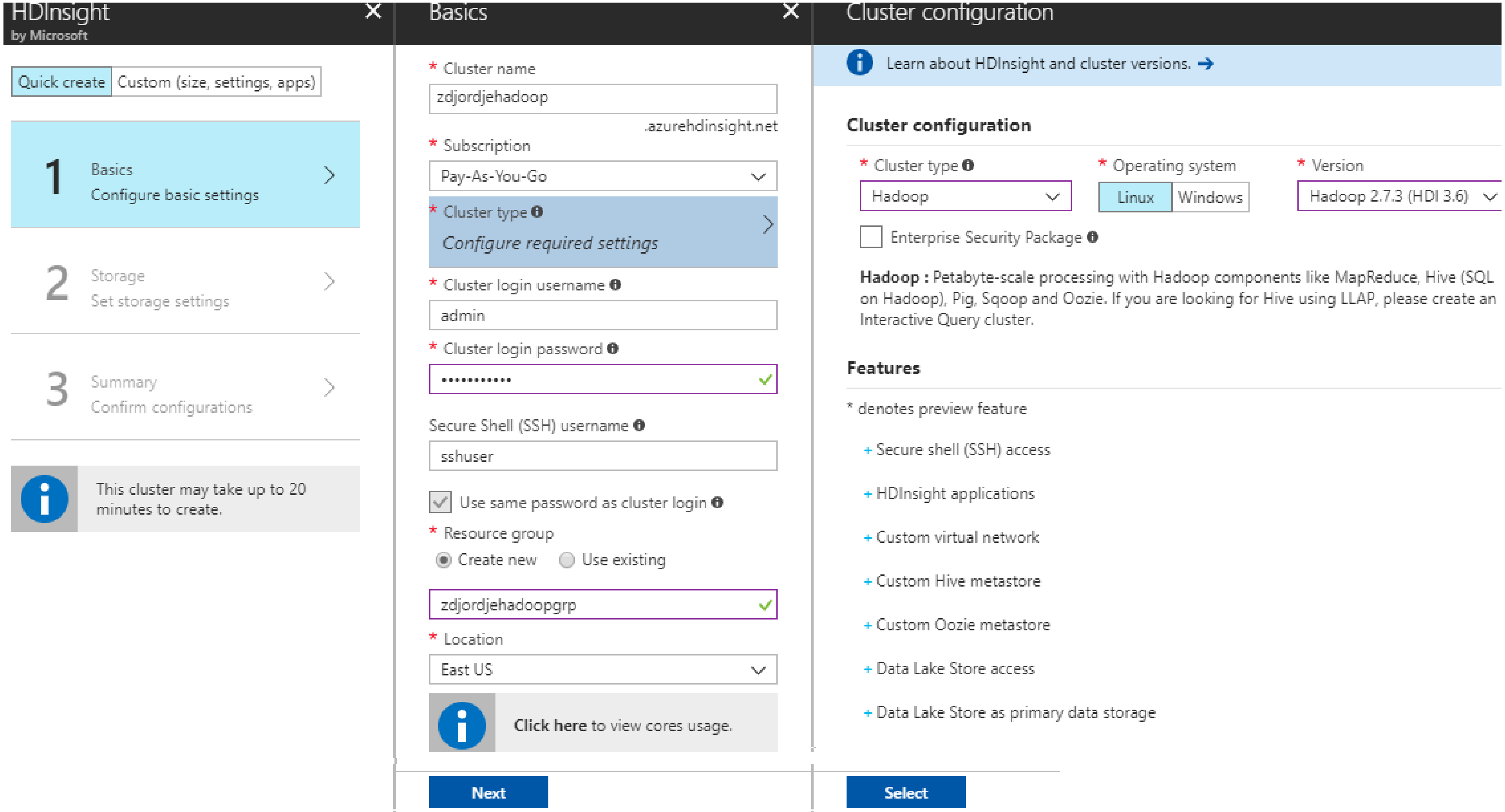
HDInsight Cluster

- In Azure Portal select +New, then Data + Analytics,
- Subsequently select HDInsight



Basic and Cluster Configuration

- Enter cluster name, cluster username and password, secure shell username, subscription, resource group and most importantly cluster type. Select Cluster type
- Cluster type can be: Hadoop, Hbase, Storm, Spark, R Server, Kafka & Interactive Query
- Select Hadoop. Hit Select.



HDInsight
by Microsoft

Quick create | Custom (size, settings, apps)

1 Basics
Configure basic settings

2 Storage
Set storage settings

3 Summary
Confirm configurations

This cluster may take up to 20 minutes to create.

Basics

* Cluster name
zdjordjehadoop
.azurehdinsight.net

* Subscription
Pay-As-You-Go

* Cluster type
Configure required settings

* Cluster login username
admin

* Cluster login password
•••••••••••

Secure Shell (SSH) username
sshuser

☑ Use same password as cluster login

* Resource group
● Create new ○ Use existing
zdjordjehadoopgrp

* Location
East US

Click here to view cores usage.

Cluster configuration

Learn about HDInsight and cluster versions. →

Cluster configuration

* Cluster type: Hadoop
* Operating system: Linux | Windows
* Version: Hadoop 2.7.3 (HDI 3.6)

□ Enterprise Security Package

Hadoop : Petabyte-scale processing with Hadoop components like MapReduce, Hive (SQL on Hadoop), Pig, Sqoop and Oozie. If you are looking for Hive using LLAP, please create an Interactive Query cluster.

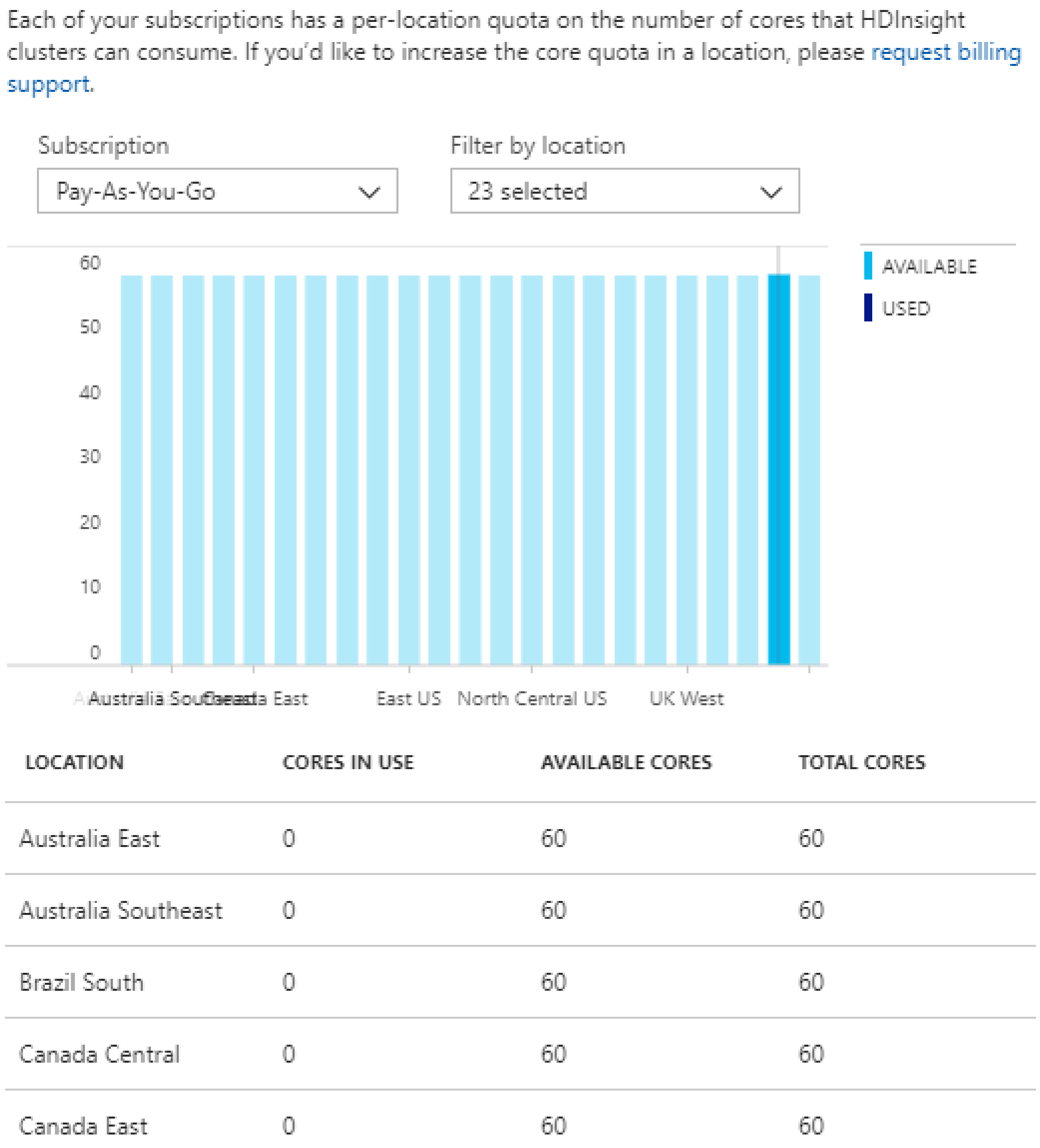
Features

* denotes preview feature

- + Secure shell (SSH) access
- + HDInsight applications
- + Custom virtual network
- + Custom Hive metastore
- + Custom Oozie metastore
- + Data Lake Store access
- + Data Lake Store as primary data storage

View cores usage

- Before hitting Next, Click to view core usage. It appears that Azure allows you to build clusters with up to 60 cores without asking for more. We will not ask.
- We will try to reduce the number of cores.
- Hit Next in Basic setup



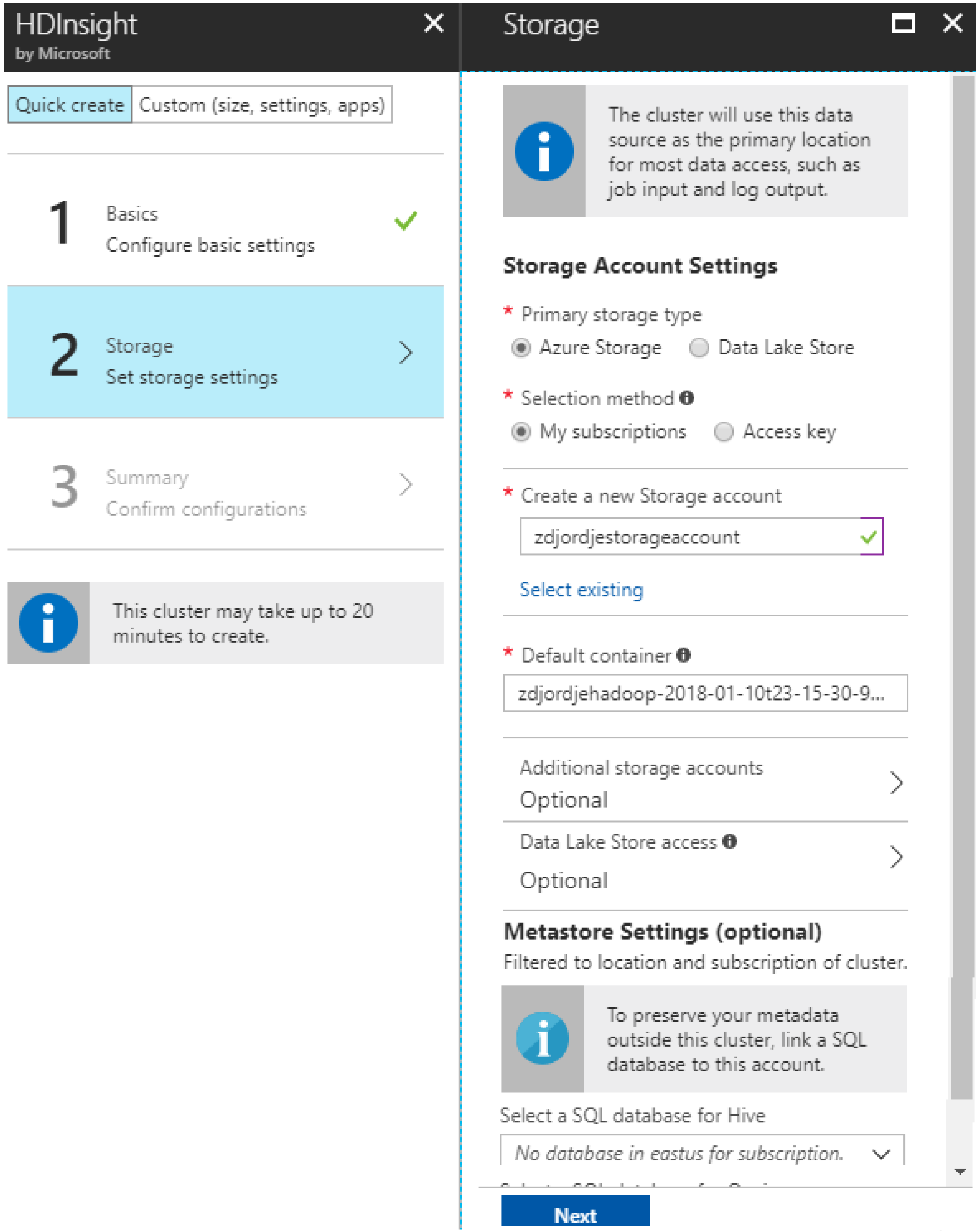
Each of your subscriptions has a per-location quota on the number of cores that HDInsight clusters can consume. If you'd like to increase the core quota in a location, please request billing support.

Subscription: Pay-As-You-Go

Filter by location: 23 selected

LOCATION	CORES IN USE	AVAILABLE CORES	TOTAL CORES
Australia East	0	60	60
Australia Southeast	0	60	60
Brazil South	0	60	60
Canada Central	0	60	60
Canada East	0	60	60

Storage



HDInsight
by Microsoft

Quick create | Custom (size, settings, apps)

1 Basics
Configure basic settings ✓

2 Storage
Set storage settings

3 Summary
Confirm configurations

This cluster may take up to 20 minutes to create.

Storage

The cluster will use this data source as the primary location for most data access, such as job input and log output.

Storage Account Settings

* Primary storage type
● Azure Storage ○ Data Lake Store

* Selection method
● My subscriptions ○ Access key

* Create a new Storage account
zdjordjestorageaccount ✓

Select existing

* Default container
zdjordjehadoop-2018-01-10t23-15-30-9...

Additional storage accounts
Optional

Data Lake Store access
Optional

Metastore Settings (optional)
Filtered to location and subscription of cluster.

To preserve your metadata outside this cluster, link a SQL database to this account.

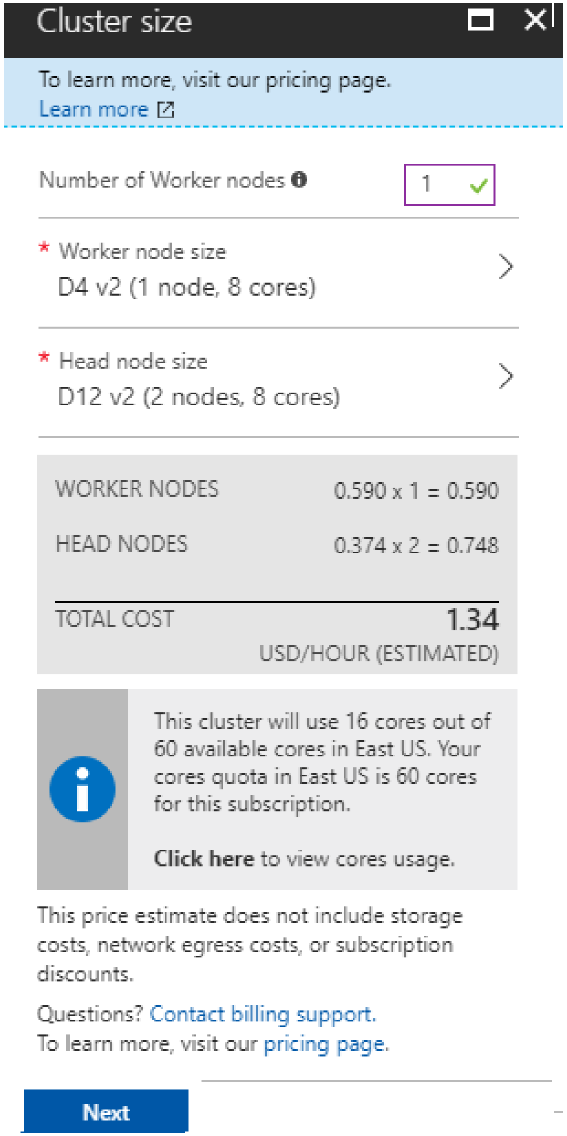
Select a SQL database for Hive
No database in eastus for subscription.

Next

- For storage, Azure gives you 2 options:
  - Azure storage or
  - Data Lake Store
- It turns out that Data Lake Store setup requires administrative privileges that most of us do not have and we will stay away from the data lake for now.
- If you do not have a storage account, create one.
- Hit Next

Cluster Summary, Applications, Cluster size

- On Cluster Summary page select Applications (Edit).
- This, just to see that you have options.
- Do not choose anything but rather select Next. Cluster size plate appears.
- Notice that Azure offers cluster with 2 head (master) and 4 worker nodes. Price per hour on my screen was $3.11. Change the number of workers to 1, price of cluster should come down to $1.34/hour. Number of cores used is reduced to 23.
- Select Next
- On Advanced settings page click Next



Cluster size

To learn more, visit our pricing page.
Learn more

Number of Worker nodes: 1 ✓

* Worker node size
D4 v2 (1 node, 8 cores)

* Head node size
D12 v2 (2 nodes, 8 cores)

WORKER NODES	0.590 x 1 = 0.590
HEAD NODES	0.374 x 2 = 0.748
TOTAL COST	1.34 USD/HOUR (ESTIMATED)

This cluster will use 16 cores out of 60 available cores in East US. Your cores quota in East US is 60 cores for this subscription.

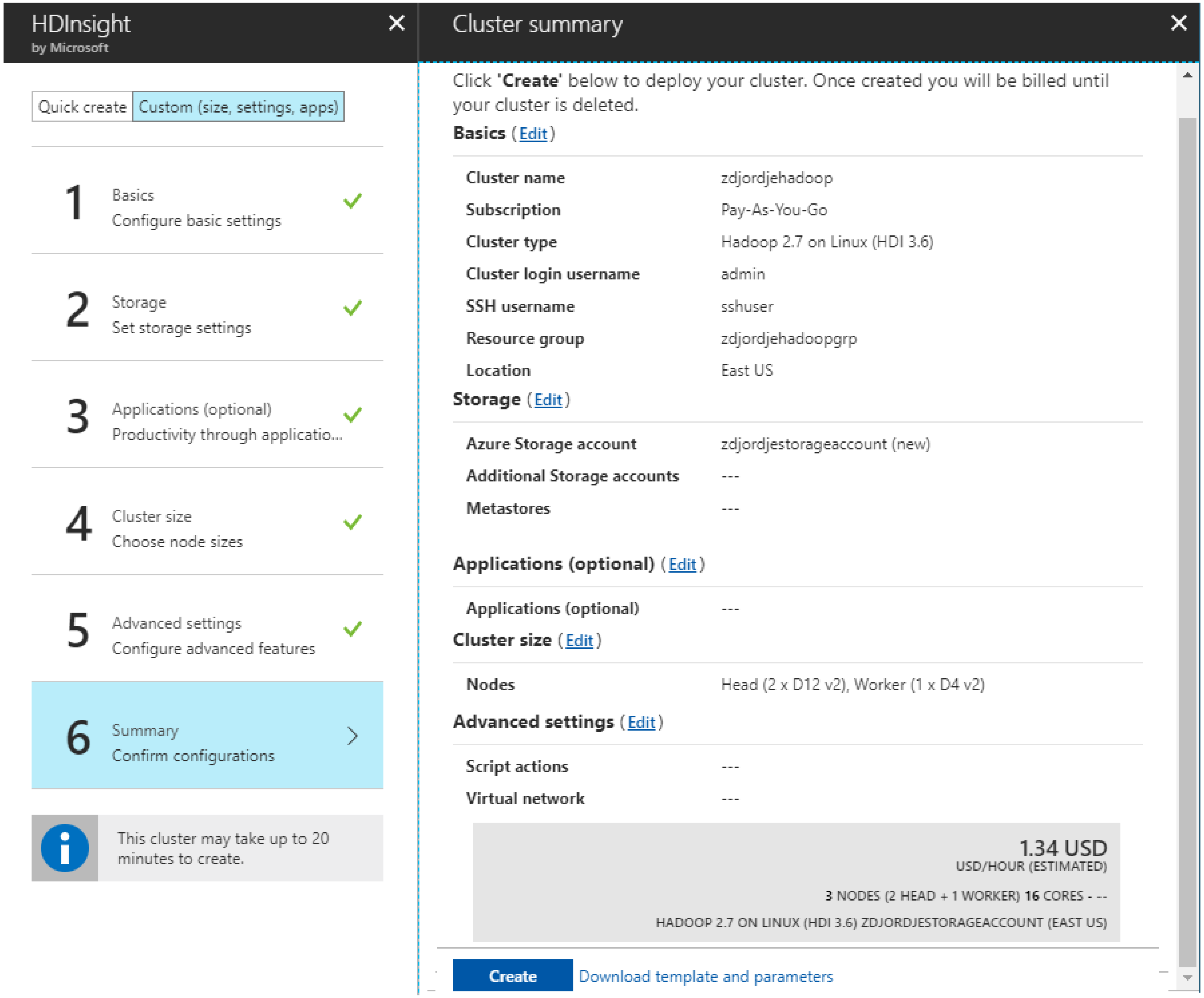
Click here to view cores usage.

This price estimate does not include storage costs, network egress costs, or subscription discounts.

Questions? Contact billing support.
To learn more, visit our pricing page.

Cluster summary

- On Cluster summary plate, you can decide to change parameters or hit Create



1 Basics
Configure basic settings ✓

2 Storage
Set storage settings ✓

3 Applications (optional)
Productivity through applicatio... ✓

4 Cluster size
Choose node sizes ✓

5 Advanced settings
Configure advanced features ✓

6 Summary
Confirm configurations

This cluster may take up to 20 minutes to create.

Cluster summary

Click **'Create'** below to deploy your cluster. Once created you will be billed until your cluster is deleted.

Basics (Edit)

Cluster name	zdjordjehadoop
Subscription	Pay-As-You-Go
Cluster type	Hadoop 2.7 on Linux (HDI 3.6)
Cluster login username	admin
SSH username	sshuser
Resource group	zdjordjehadoopgrp
Location	East US

Storage (Edit)

Azure Storage account	zdjordjestorageaccount (new)
Additional Storage accounts	---
Metastores	---

Applications (optional) (Edit)

Applications (optional)	---

Cluster size (Edit)

Nodes	Head (2 x D12 v2), Worker (1 x D4 v2)

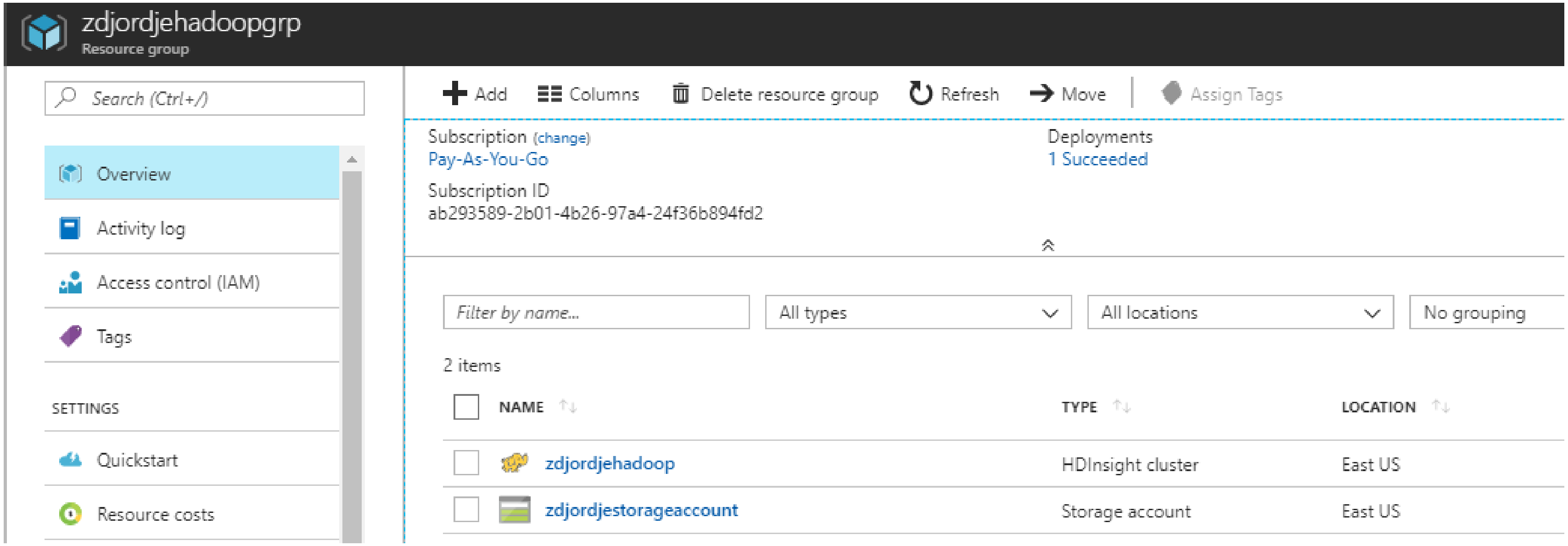
Advanced settings (Edit)

Script actions	---
Virtual network	---

1.34 USD
USD/HOUR (ESTIMATED)
3 NODES (2 HEAD + 1 WORKER) **16** CORES - --
HADOOP 2.7 ON LINUX (HDI 3.6) ZDJORDJESTORAGEACCOUNT (EAST US)

Resource Group content

- My resource group, zdjordjehadoopgrp, no contains a HDInsight cluster named zdjordjehadoop



zdjordjehadoopgrp
Resource group

Subscription (change)
Pay-As-You-Go

Subscription ID
ab293589-2b01-4b26-97a4-24f36b894fd2

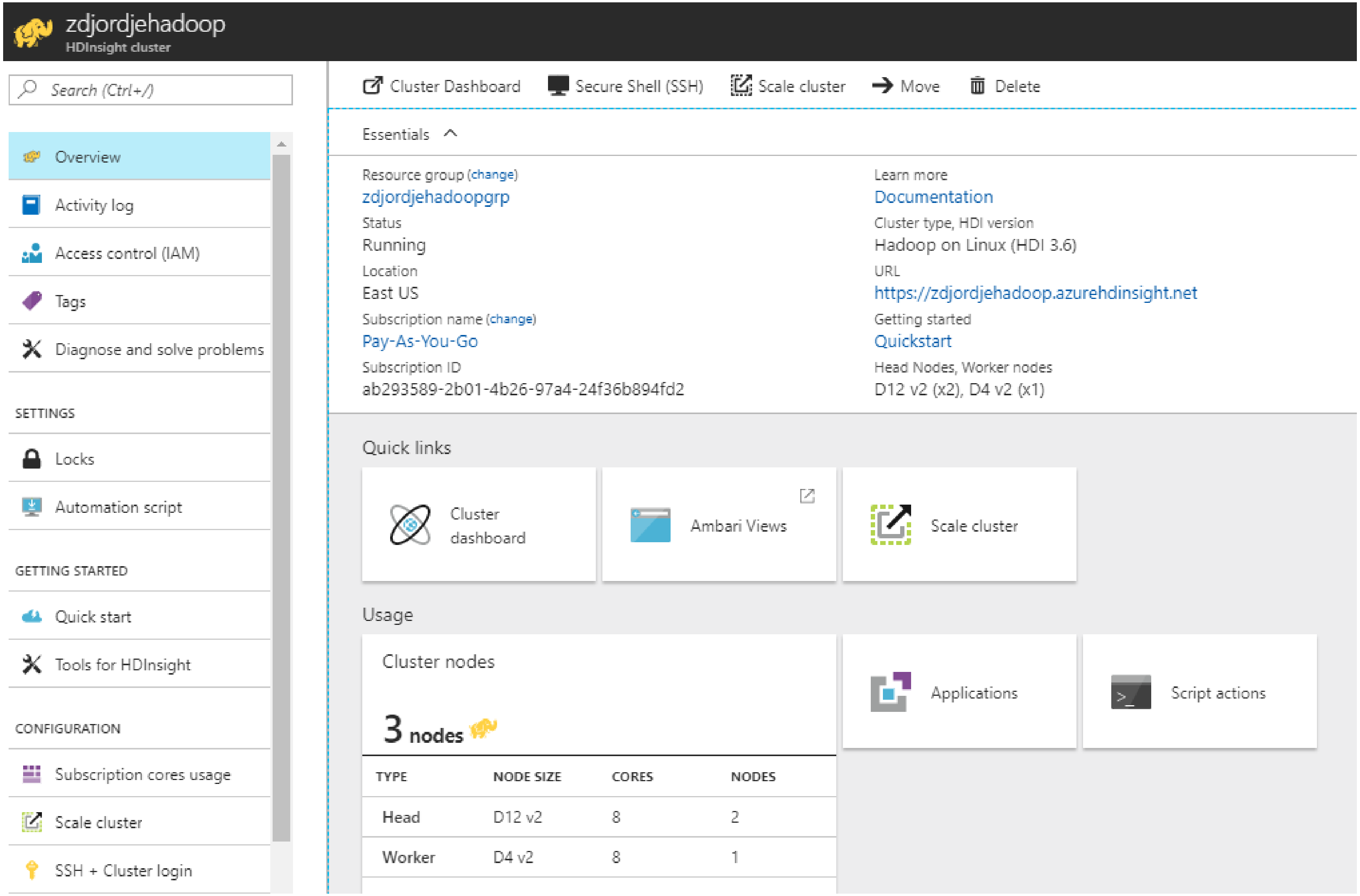
Deployments
1 Succeeded

2 items

NAME	TYPE	LOCATION
zdjordjehadoop	HDInsight cluster	East US
zdjordjestorageaccount	Storage account	East US

- Click on cluster name (zdjordjehadoop)

HDInsight Cluster View



Resource group (change)
zdjordjehadoopgrp

Status
Running

Location
East US

Subscription name (change)
Pay-As-You-Go

Subscription ID
ab293589-2b01-4b26-97a4-24f36b894fd2

Learn more
Documentation

Cluster type, HDI version
Hadoop on Linux (HDI 3.6)

URL
https://zdjordjehadoop.azurehdinsight.net

Getting started
Quickstart

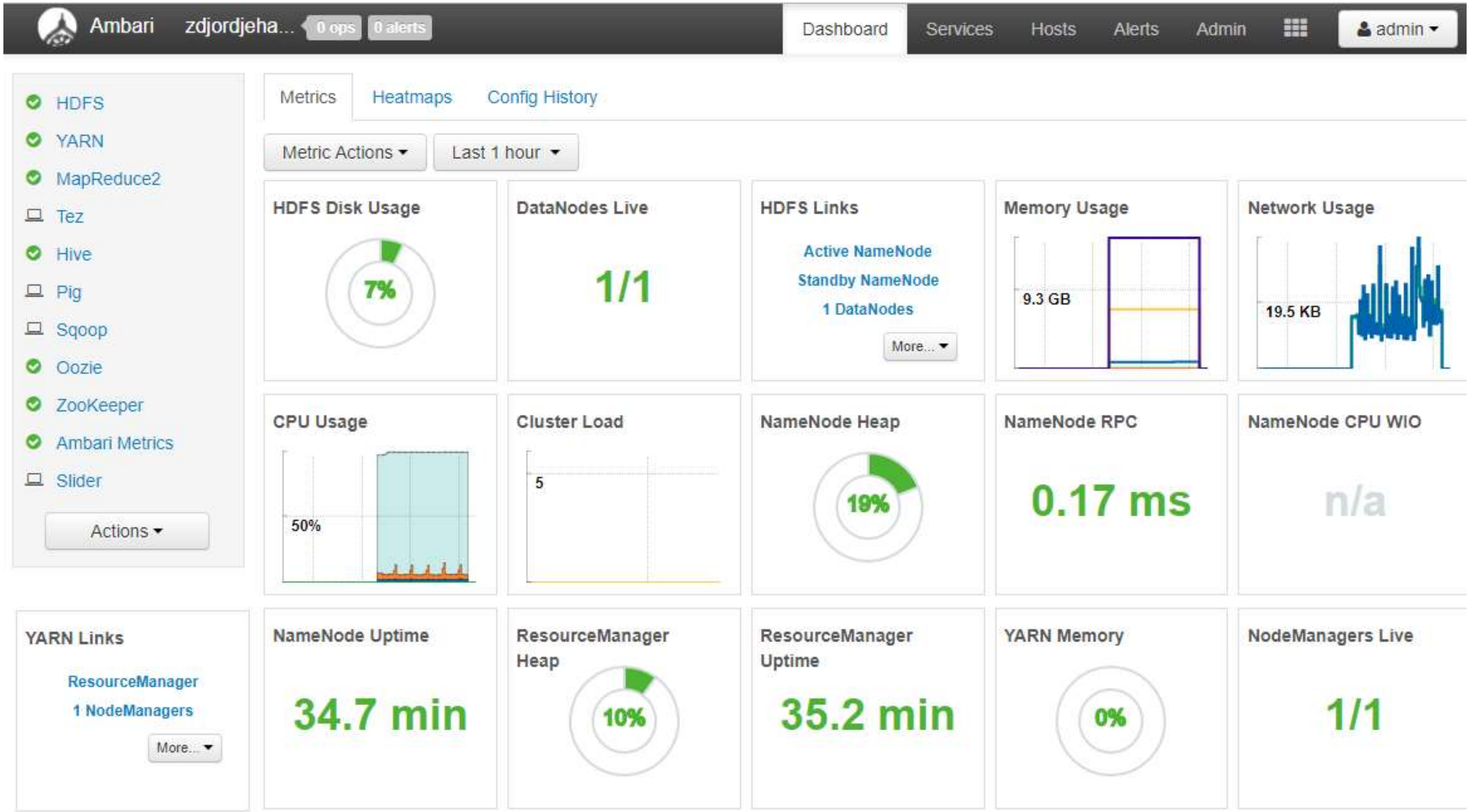
Head Nodes, Worker nodes
D12 v2 (x2), D4 v2 (x1)

Cluster nodes

3 nodes

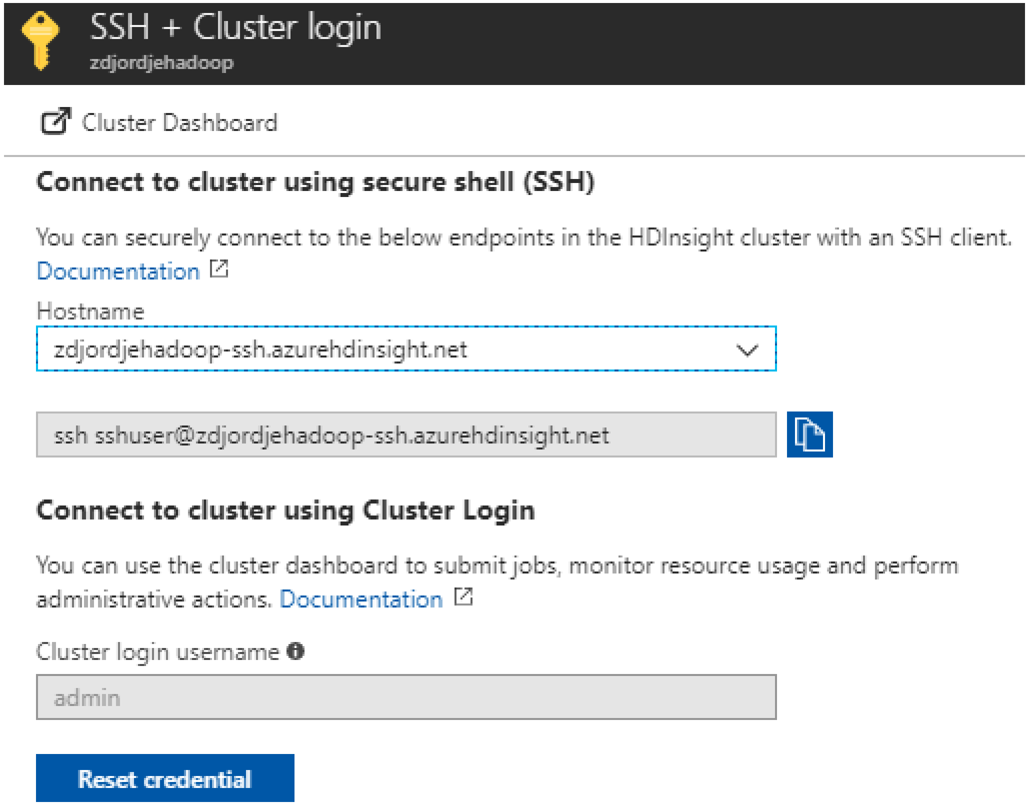
TYPE	NODE SIZE	CORES	NODES
Head	D12 v2	8	2
Worker	D4 v2	8	1

Cluster Dashboard



ssh Access

- HDInsight Dashboard gives you a convenient "Secure Shell (SSH)" link which tells you how to connect to the master node using ssh. We can do that using Cygwin.



SSH + Cluster login

zdjordjehadoop

Cluster Dashboard

Connect to cluster using secure shell (SSH)

You can securely connect to the below endpoints in the HDInsight cluster with an SSH client. Documentation

Hostname

zdjordjehadoop-ssh.azurehdinsight.net

```
ssh sshuser@zdjordjehadoop-ssh.azurehdinsight.net
```

Connect to cluster using Cluster Login

You can use the cluster dashboard to submit jobs, monitor resource usage and perform administrative actions. Documentation

Cluster login username

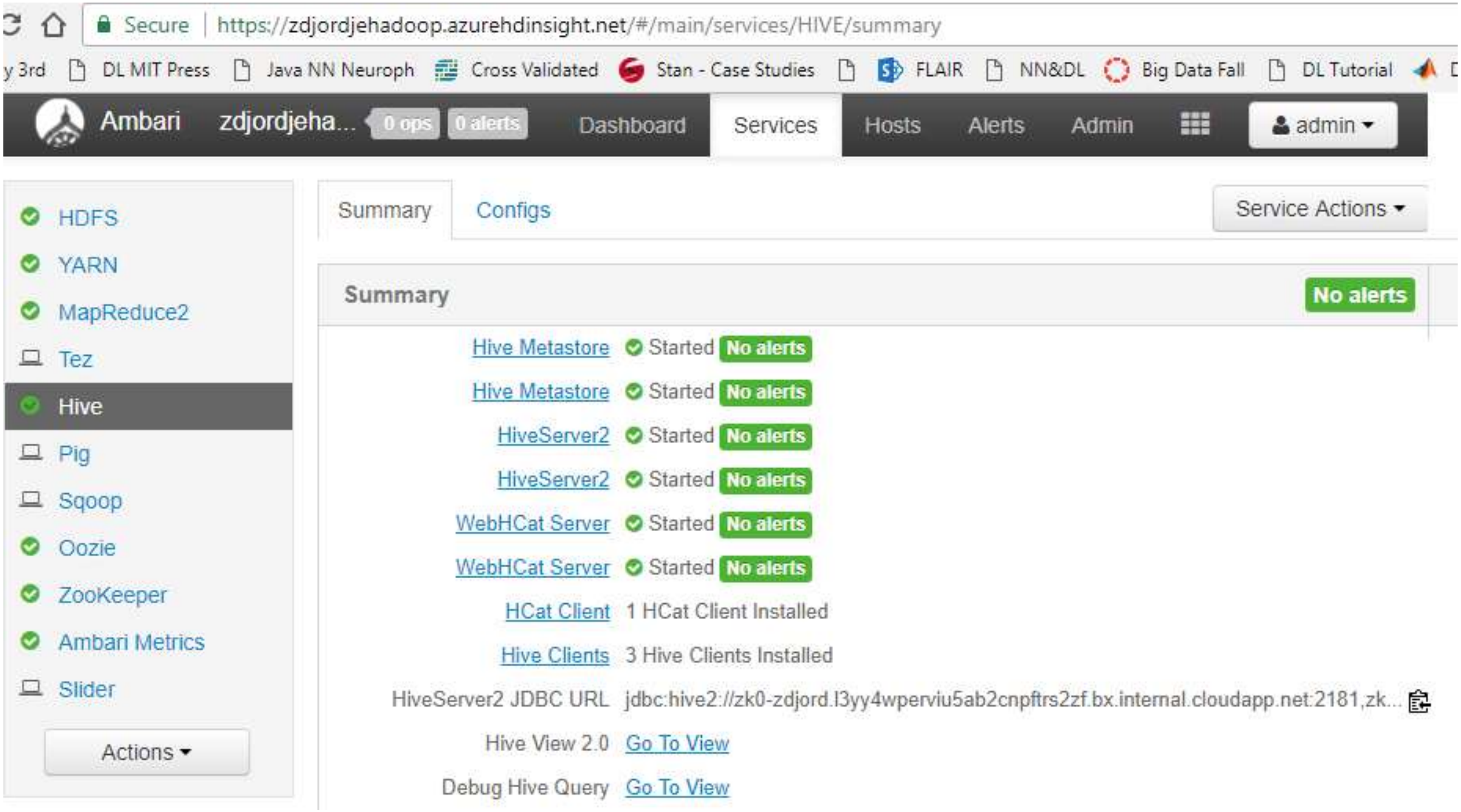
admin

Cygwin Login

```
$ ssh sshuser@zdjordjehadoop-ssh.azurehdinsight.net
The authenticity of host 'zdjordjehadoop-ssh.azurehdinsight.net (40.71.25.124)' can't
be established.
ECDSA key fingerprint is SHA256:PDav303yIBI+v6o/JVqDELV730NFfnhtixwM/VAlOz0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'zdjordjehadoop-ssh.azurehdinsight.net,40.71.25.124' (ECDSA)
to the list of known hosts.
Authorized uses only. All activity may be monitored and reported.
sshuser@zdjordjehadoop-ssh.azurehdinsight.net's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-101-generic x86_64)
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

82 packages can be updated.
45 updates are security updates.

Welcome to HDInsight. Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted
by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

sshuser@hn0-zdjord:~$
```

Hive View

- On HDInsight plate, hit Cluster dashboard link, then, again hit HDInsight cluster dashboard. On the right navigation list select Hive. If asked provide password for user admin. Under Summary, select Hive View 2.0



Summary No alerts

Hive Metastore	✓ Started No alerts
Hive Metastore	✓ Started No alerts
HiveServer2	✓ Started No alerts
HiveServer2	✓ Started No alerts
WebHCat Server	✓ Started No alerts
WebHCat Server	✓ Started No alerts
HCat Client	1 HCat Client Installed
Hive Clients	3 Hive Clients Installed
HiveServer2 JDBC URL	jdbc:hive2://zk0-zdjord.l3yy4wperviu5ab2cnpftrs2zf.bx.internal.cloudapp.net:2181,zk...
Hive View 2.0	Go To View
Debug Hive Query	Go To View

Hive Query View

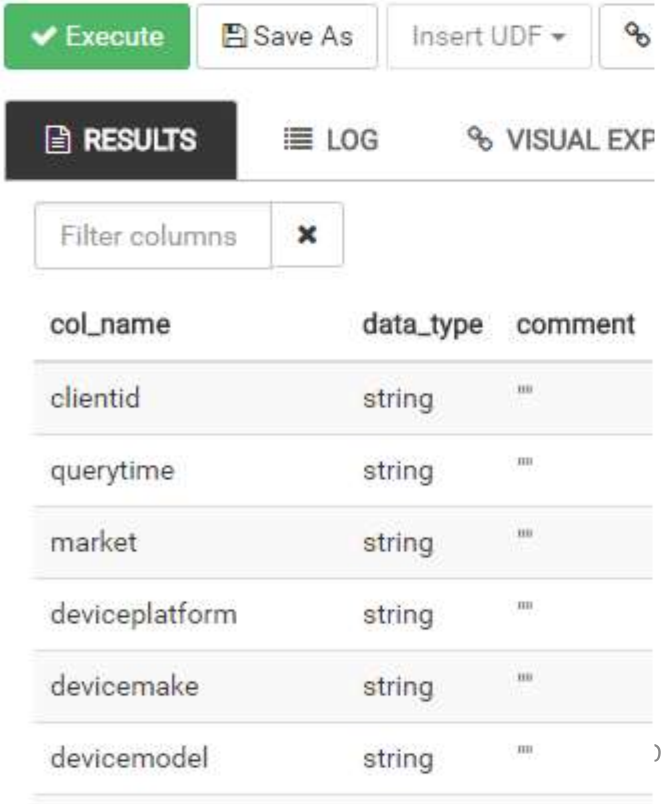
- In Hive Query view that appears enter:

`1. show tables;` # note Hive queries are terminated with ";" (not Python)

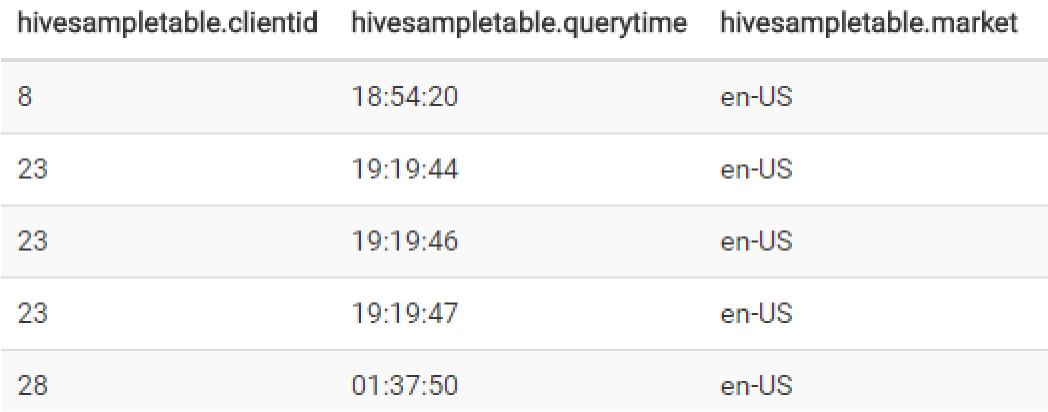
- Hit Execute on the bottom. You will see a single table name: hivesampletable.
- Write another query:

```
2. describe hivesampletable;
3. select * from hivesampletable;
```

You will get table description and content:



col_name	data_type	comment
clientid	string	""
querytime	string	""
market	string	""
deviceplatform	string	""
devicemake	string	""
devicemodel	string	""



hivesampletable.clientid	hivesampletable.querytime	hivesampletable.market
8	18:54:20	en-US
23	19:19:44	en-US
23	19:19:46	en-US
23	19:19:47	en-US
28	01:37:50	en-US

- To stop consuming resources, wipe out your resource group and included cluster.

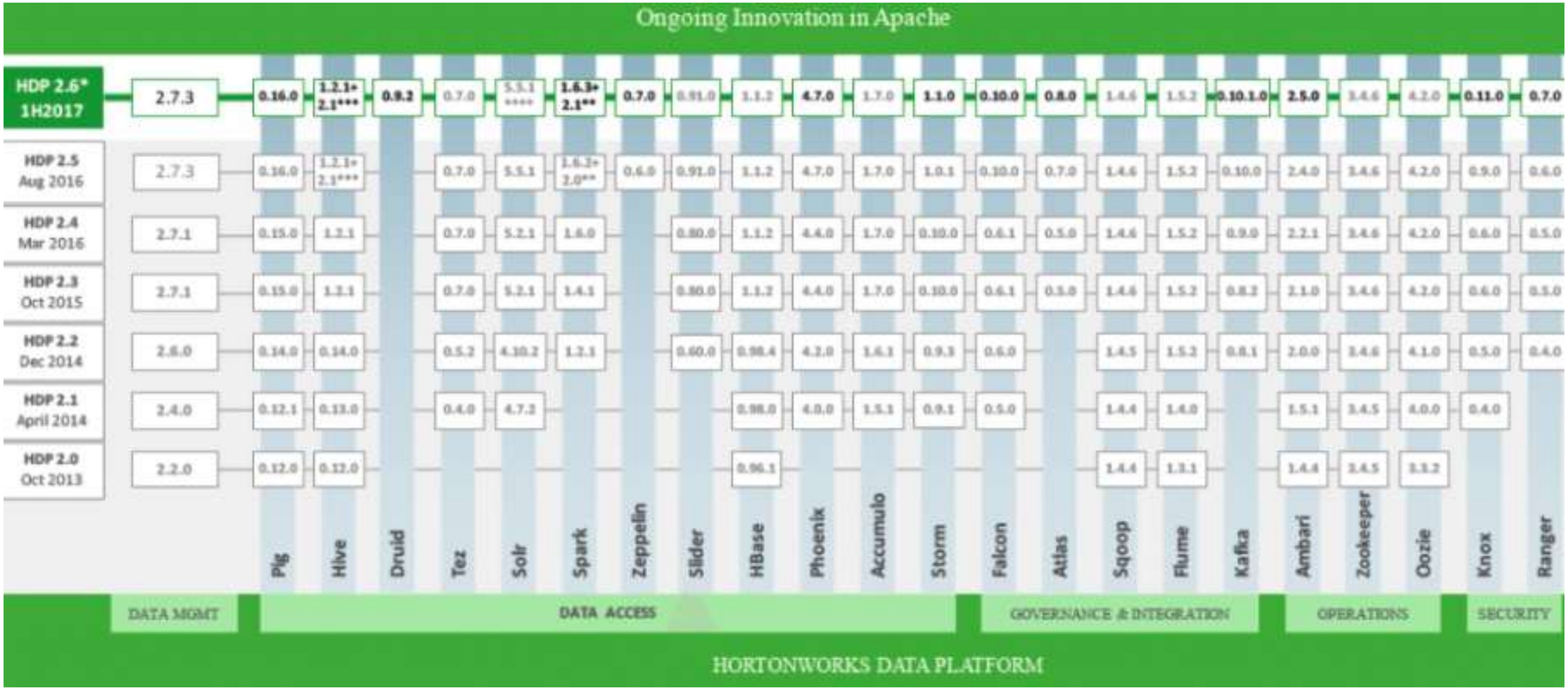
Hortonworks VMs

- Microsoft’s Azure VMs are based on Hortonworks HDP framework. When developing software, you are better of working on Hortonworks VMs on your laptop, PC or Mac, where they do not cost you a penny.
- One problem with those VMs is that they require at least 8GB of RAM to run.
- Hortonworks has to family of products HDP (Hortonworks Data Platform) and HDF (Hortonworks Data Flow) and corresponding VMs. We will use HDP VMs and in particular so called sandbox VMs.
- The HDP Sandbox, a single node VM, lets you get started with Apache Hadoop, Falcon, Atlas, Tez, Sqoop, Flume, Kafka, Pig, Hive, HBase, Accumulo, Storm, Solr, Spark, Ranger, Knox, Ambari, ZooKeeper, Oozie, Phoenix, NiFi, HAWQ, Zeppelin, Atlas, Slider, Mahout, MapReduce, HDFS, YARN, Metron Apache projects
- The HDF Sandbox lets you get started with Apache NiFi, Apache Kafka, Apache Storm, Druid and Streaming Analytics Manager.

- Go to https://hortonworks.com/products/sandox and download HDP Sandbox

Content of HDP

- On URL https://hortonworks.com/product/data-platforms/hdp you can see detailed content of HDP framework in terms of various Apache and other open source packages.



Ongoing Innovation in Apache

		Pig	Hive	Druid	Tez	Solr	Spark	Zeppelin	Slider	HBase	Phoenix	Accumulo	Storm	Falcon	Atlas	Sqoop	Flume	Kafka	Ambari	Zookeeper	Oozie	Knox	Ranger
HDP 2.6* 1H2017	2.7.3	0.16.0	1.2.1+ 2.1***	0.9.2	0.7.0	5.5.1 ****	1.6.3+ 2.1**	0.7.0	0.91.0	1.1.2	4.7.0	1.7.0	1.1.0	0.10.0	0.8.0	1.4.6	1.5.2	0.10.1.0	2.5.0	3.4.6	4.2.0	0.11.0	0.7.0
HDP 2.5 Aug 2016	2.7.3	0.16.0	1.2.1+ 2.1***		0.7.0	5.5.1	1.6.2+ 2.0**	0.6.0	0.91.0	1.1.2	4.7.0	1.7.0	1.0.1	0.10.0	0.7.0	1.4.6	1.5.2	0.10.0	2.4.0	3.4.6	4.2.0	0.9.0	0.6.0
HDP 2.4 Mar 2016	2.7.1	0.15.0	1.2.1		0.7.0	5.2.1	1.6.0		0.80.0	1.1.2	4.4.0	1.7.0	0.10.0	0.6.1	0.5.0	1.4.6	1.5.2	0.9.0	2.2.1	3.4.6	4.2.0	0.6.0	0.5.0
HDP 2.3 Oct 2015	2.7.1	0.15.0	1.2.1		0.7.0	5.2.1	1.4.1		0.80.0	1.1.2	4.4.0	1.7.0	0.10.0	0.6.1	0.5.0	1.4.6	1.5.2	0.8.2	2.1.0	3.4.6	4.2.0	0.6.0	0.5.0
HDP 2.2 Dec 2014	2.6.0	0.14.0	0.14.0		0.5.2	4.10.2	1.2.1		0.60.0	0.98.4	4.2.0	1.6.1	0.9.3	0.6.0		1.4.5	1.5.2	0.8.1	2.0.0	3.4.6	4.1.0	0.5.0	0.4.0
HDP 2.1 April 2014	2.4.0	0.12.1	0.13.0		0.4.0	4.7.2				0.98.0	4.0.0	1.5.1	0.9.1	0.5.0		1.4.4	1.4.0		1.5.1	3.4.5	4.0.0	0.4.0	
HDP 2.0 Oct 2013	2.2.0	0.12.0	0.12.0							0.96.1						1.4.4	1.3.1		1.4.4	3.4.5	3.3.2		

DATA MGMT

DATA ACCESS

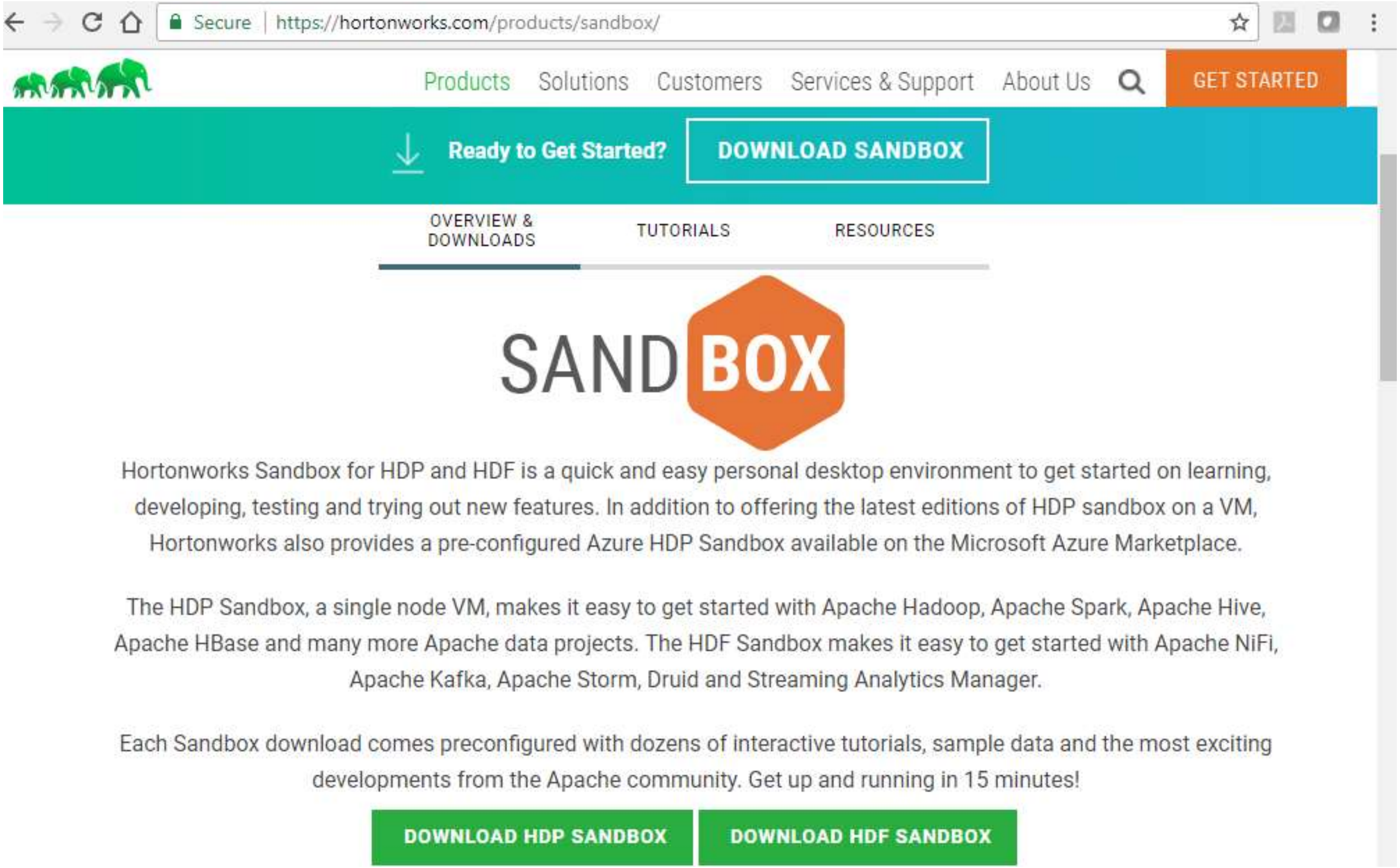
GOVERNANCE & INTEGRATION

OPERATIONS

SECURITY

HORTONWORKS DATA PLATFORM

Download Hortonworks HDP Sandbox



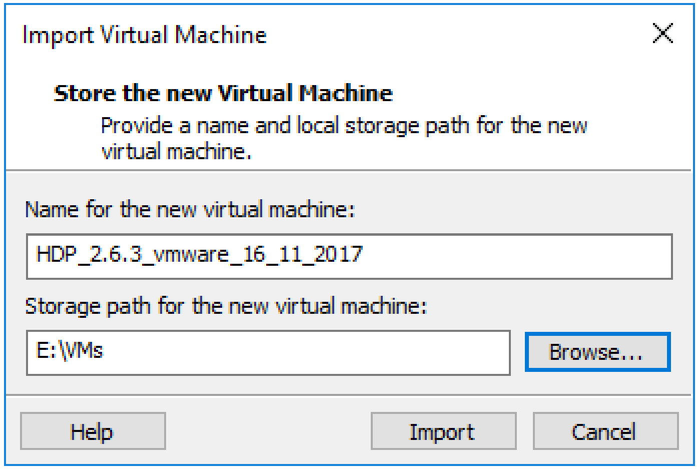
Hortonworks Sandbox for HDP and HDF is a quick and easy personal desktop environment to get started on learning, developing, testing and trying out new features. In addition to offering the latest editions of HDP sandbox on a VM, Hortonworks also provides a pre-configured Azure HDP Sandbox available on the Microsoft Azure Marketplace.

The HDP Sandbox, a single node VM, makes it easy to get started with Apache Hadoop, Apache Spark, Apache Hive, Apache HBase and many more Apache data projects. The HDF Sandbox makes it easy to get started with Apache NiFi, Apache Kafka, Apache Storm, Druid and Streaming Analytics Manager.

Each Sandbox download comes preconfigured with dozens of interactive tutorials, sample data and the most exciting developments from the Apache community. Get up and running in 15 minutes!

Handle Sandbox VM

- You can select a VMware VM, Virtual Box VM or a Docker container.
- I downloaded `HDP_2.6.3_vmware_16_11_2017.ova` file with a volume of 10GB.
- An OVA file is a virtual appliance used by virtualization applications such as VMware Workstation and Oracle VM Virtualbox. It is a package that contains files used to describe a virtual machine, which includes an .OVF descriptor file, optional manifest (.MF) and certificate files, and other related files.
- `HDP_263_vmware…ova` file can be opened by VMWare Workstation, which will transform it into a regular VMWare VM.
- When you open VMWare Workstation, select Open a Virtual Machine and then select downloaded *.ova file. Select the storage path for new virtual machine.
- Select Import



Import Virtual Machine

Store the new Virtual Machine

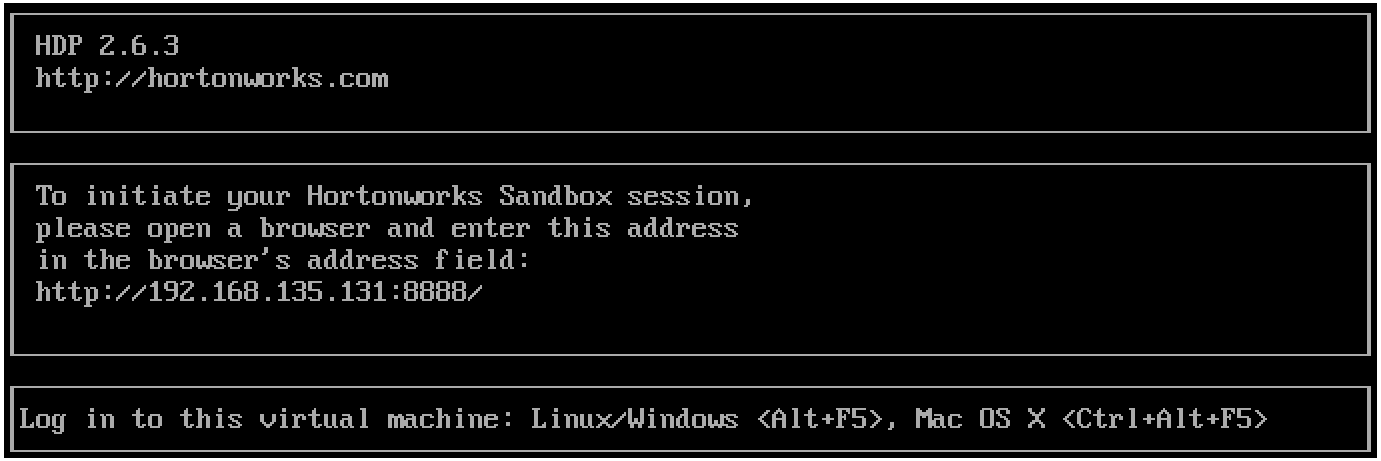
Provide a name and local storage path for the new virtual machine.

Name for the new virtual machine: HDP_2.6.3_vmware_16_11_2017

Storage path for the new virtual machine: E:\VMs

The first screen and IP Address

- Once the import is finished, run the VM. It will display a confirmation screen.



```
HDP 2.6.3
http://hortonworks.com

To initiate your Hortonworks Sandbox session,
please open a browser and enter this address
in the browser's address field:
http://192.168.135.131:8888/

Log in to this virtual machine: Linux/Windows <Alt+F5>, Mac OS X <Ctrl+Alt+F5>
```

- Note the IP address of your machine. My IP address is 192.168.135.131
- On your Windows machine as an administrator open Notepad and then edit file: `C:\Windows\system32\Drivers\etc\hosts`
- And add a line associating the above IP address with your favorite machine names.
- For example:

```
192.168.135.131 localhost sandbox.hortonworks.com sandbox-hdp.hortonworks.
```

- On Mac you would add that line to /private/etc/host and on Linux to /etc/hosts file.
- If you follow messages displayed during the boot-up, you would notice that this VM runs a CentOS 7 operating system.

Run ssh client

- To login into your VM, on Linux/Windos do `Alt+F5`. On Mac OS, do `Ctrl+Alt+F5`
- The default password for user `root` is `hadoop` (all lowercase)
- You can connect to your machine using Cygwin or any other ssh client. Note that this VM listens to ssh traffic on port `2222`. `Type:`

```
$ ssh root@192.168.135.131 -p 2222
The authenticity of host '[192.168.135.131]:2222
([192.168.135.131]:2222)' can't be established.
RSA key fingerprint is
SHA256:0igBYD9FHqqbVcTryp+T3RxYa2l0JvTTbPNihSLUXyg.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[192.168.135.131]:2222' (RSA) to the
list of known hosts.
root@192.168.135.131's password:hadoop
You are required to change your password immediately (root
enforced)
Last login: Thu Jan 11 21:08:19 2018 from 192.168.135.1
Changing password for root.
(current) UNIX password:hadoop
New password:***********
Retype new password:**********
[root@sandbox-hdp ~]#
```

Copy files to and from VM using `scp`

- Using Cygwin or a terminal of your choice, you can transfer files to/from sandbox and local machine.
- Transfer file from local machine to sandbox directory /tmp by issuing the following command:

```
scp -P 2222 local_dir_or_file root@sandbox-hdp.hortonworks.com:/tmp
```

- Transfer file from sandbox to local machine:

```
scp -P 2222 root@sandbox-hdp.hortonworks.com:/tmp/sandbox_dir_or_file
local_dir_or_file
```

- Notice the difference between the two commands?
- To send data from local machine to sandbox, the local machine directory path comes before sandbox directory. To transfer data from sandbox to local machine, the command arguments are reversed.

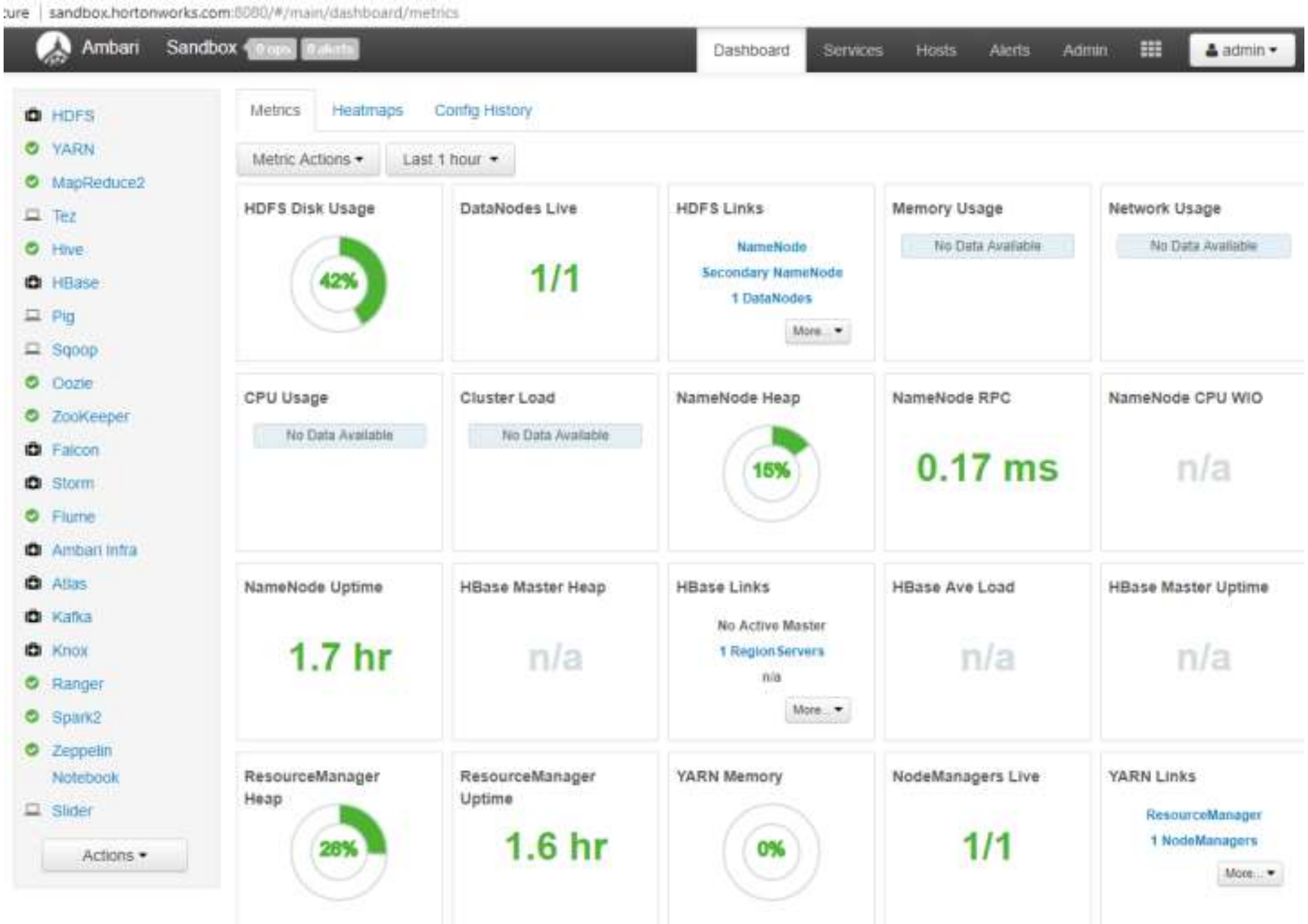
Default Users

- VM comes with a set of default users: admin, maria_dev,raj_ops, holger_gov, amy_ds. Those users have password the same as the username.
- User admin requires change of password. To accomplish the change, run the following command on the Linux prompt:

```
[root@sandbox-hdp ~]# ambari-admin-password-reset
Please set the password for admin:
Please set the password for admin:
Please retype the password for admin:

The admin password has been set.
Restarting ambari-server to make the password change effective...

Using python  /usr/bin/python
Restarting ambari-server
Waiting for server stop...
Ambari Server stopped
Ambari Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Ambari database consistency check started...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start.................
```

Dashboard

- Go to http://sandbox.hortonworks.com:8888
- Then select LAUNCH DASHBOARD. Enter username admin and the password you have just assigned to that user. Singin! A dashboard appears:



Please remember port `2222`

- Hortonworks Sandbox runs on docker container.
- So some profile configs are loaded only when you do "`ssh on port 2222`"
- This is the way port forwarding works is that it exposes port `22` of the sandbox machine to the outside network of your PC via port `2222`.
- If you try ssh-ing into your VM through port `22`, you might get in but would see a different environment

- Once you create user `centos`, for example, you will `ssh` into the sandbox VM by typing:

```
$ ssh centos@192.168.1356.132 -p 2222
```

Introduction to
Hadoop Distribute File System (HDFS)

Examine Directories in your HDFS

- User `hdfs` has complete insight into all directories in your HDFS.
- User `hdfs` can create user (home) directories and perform maintenance operation on HDFS.
- To see home directories in HDFS, type:

```
$ sudo –u hdfs hdfs dfs –ls /user/
drwxr-xr-x   - root root       0 2017-07-19 06:28 /user/root
drwxr-xr-x   - mapred   hadoop          0 2017-07-19 06:29 /user/history
drwxrwxrwx   - hive     supergroup      0 2017-07-19 06:31 /user/hive
drwxrwxrwx   - hue      supergroup      0 2017-07-19 06:30 /user/hue
drwxrwxrwx   - jenkins  supergroup      0 2017-07-19 06:29 /user/jenkins
drwxrwxrwx   - oozie    supergroup      0 2017-07-19 06:30 /user/oozie
drwxrwxrwx   - root     supergroup      0 2017-07-19 06:29 /user/root
drwxr-xr-x   - hdfs     supergroup      0 2017-07-19 06:31 /user/spark
```

- You can see inside all of those directories, just type

```
$ sudo –u hdfs hdfs dfs –ls  -R /user/hive
drwxrwxrwx   - hive supergroup      0 2017-07-19 06:31 /user/hive/warehouse
```

Create Linux user `centos`

- If you need to create a new Linux user, e.g. `centos`, log in as `root`, or type:

```
$ su –
```

  and enter `root`’s password.

- Now, as user `root`, type:

```
$ useradd –g mapred centos
```

- The above creates new user `centos`, as a member of group `mapred`.
- Please note that a user running Hadoop MapReduce programs must be a member of `mapred` group. To create password for new user, type:

```
$ passwd centos
```

- At the `New password:` prompt , enter a password for user `centos`, press [Enter].
- At the `Retype new password:` prompt, enter the same password to confirm.
- If user `root` wants to become user `centos`, `root` types:

```
[root@sandbox-hdp ~]$ su -- centos
Password: xxxxxxxxxx
[centos@sandbox-hdp root]$ exit # turns you back into root
```

- Should you want to remove Linux user `centos`, type:

```
$ sudo userdel centos
```

Make `centos` a `sudo` user

- Allow users in group wheel to run `sudo` command without password.
- As user root, type

```
$ chmod +rw /etc/sudoers
$ visduo -f /etc/sudoers
```

- Uncomment the last (second) line with `%wheel :`

```
## Allows people in group wheel to run all commands
# %wheel        ALL=(ALL)       ALL

## Same thing without a password
%wheel  ALL=(ALL)       NOPASSWD: ALL
```

- Add user `centos` to group `wheel`:

```
[root]$ usermod -aG wheel centos
```

Create HDFS Directories for user `centos`

- Next, create HDFS home directory for a new MapReduce user `centos`.
- By the way, on a truly distributed cluster you will do all of this on the Name Node.
- Type:

```
$ sudo –u hdfs hdfs dfs –mkdir /user/centos
```

- `hdfs dfs –mkdir` command automatically creates parent directories if they don't already exist. This is similar to the Unix `mkdir` command with the `–p` option.
- Once we have the directory we want to grant the ownership to user `centos`.

```
$ sudo –u hdfs hdfs dfs -chown centos:mapred /user/centos
```

- Finally we want to give full read-write-execute right on that directory.

```
$ sudo –u hdfs hdfs dfs –chmod 1777 /user/centos
```

- Alternatively, if the Linux user already exist, you can login as that user and create the home directory as follows:

```
$ sudo –u hdfs hdfs dfs -mkdir /user/$USER
$ sudo –u hdfs hdfs dfs -chown $USER /user/$USER
```

- If your system does not have HDFS directories for user `root`, you could use the above procedure to create them

hadoop script

- You used user hdfs and command hdfs only for high level system maintenance of HDFS.
- To manipulate files in your (HDFS) home directory, transfer files from local file system to HDFS and back, ordinary users use command hadoop.
- Let us find out what hadoop is. If you type:

```
$ which hadoop        # you will get:
/usr/bin/hadoop
```

- If you open /usr/bin/hadoop file, you will see that it invokes another file

```
/usr/hdp/2.6.3.0-235//hadoop/bin/hadoop.distro
```

- That other hadoop file is also a script which runs various Java programs depending on the options you pass to it. In the original Hadoop script we see that Hortonworks VM uses the following environmental variables:

```
HADOOP_HOME=/usr/hdp/2.6.3.0-235/hadoop
HADOOP_MAPRED_HOME=/usr/hdp/2.6.3.0-235/hadoop-mapreduce
HADOOP_YARN_HOME=/usr/hdp/2.6.3.0-235/hadoop-yarn}
HADOOP_LIBEXEC_DIR=${HADOOP_HOME}/libexec
```

- To see the options of hadoop command, invoke hadoop by itself:

```
$ hadoop
```

Options of hadoop script

```
[root]# hadoop
Usage: hadoop [--config confdir] COMMAND
       where COMMAND is one of:
  fs                     run a generic filesystem user client
  version                print the version
  jar <jar>              run a jar file
  checknative [-a|-h]    check native hadoop and compression
                        libraries availability
  distcp <srcurl> <desturl> copy file or directories recursively
  archive -archiveName NAME -p <parent path> <src>* <dest> create
                               a hadoop archive
  classpath              prints the class path needed to get the
  credential             interact with credential providers
                         Hadoop jar and the required libraries
  daemonlog              get/set the log level for each daemon
  trace                  view and modify Hadoop tracing settings or
  CLASSNAME              run the class named CLASSNAME

Most commands print help when invoked w/o parameters.
```

Hadoop Distributed File System Shell, `fs` command

- Hadoop has access to both local, Linux, file system, and its own distributed file system (HDFS, Hadoop Distributed File System)
- We access HDFS through Hadoop distributes file system shell, `fs`;
- By invoking command itself:

```
$ hadoop  fs
```

- we get a long list of options. Some of those resemble Unix (Linux) commands. Some are different.
- We use those commands to create directories in the HDFS, copy files between HDFS and the local file system, Internet and AWS S3 buckets.
- When you use `fs`, you always prefix it with `hadoop`.

Options of file system shell, `hadoop fs`

```
[root]$ hadoop fs
Usage: hadoop fs [generic options]
         [-appendToFile <localsrc> ... <dst>]
         [-cat [-ignoreCrc] <src> ...]
         [-checksum <src> ...]
         [-chgrp [-R] GROUP PATH...]
         [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
         [-chown [-R] [OWNER][:[GROUP]] PATH...]
         [-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]
         [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
         [-count [-q] [-h] [-v] [-x] <path> ...]
         [-cp [-f] [-p | -p[topax]] <src> ... <dst>]
         [-createSnapshot <snapshotDir> [<snapshotName>]]
         [-deleteSnapshot <snapshotDir> <snapshotName>]
         [-df [-h] [<path> ...]]
         [-du [-s] [-h] [-x] <path> ...]
         [-expunge]
         [-find <path> ... <expression> ...]
         [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
         [-getfacl [-R] <path>]
         [-getfattr [-R] {-n name | -d} [-e en] <path>]
         [-getmerge [-nl] <src> <localdst>]
```

File system shell `fs`

```
           [-help [cmd ...]]
            [-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [<path> ...]]
            [-mkdir [-p] <path> ...]
            [-moveFromLocal <localsrc> ... <dst>]
           [-moveToLocal [-crc] <src> <localdst>]
           [-mkdir <path>]
           [-setrep [-R] [-w] <rep> <path/file>]
           [-touchz <path>]
           [-test -[ezd] <path>]
           [-stat [format] <path>]
           [-tail [-f] <file>]
           [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
           [-chown [-R] [OWNER][:[GROUP]] PATH...]
           [-chgrp [-R] GROUP PATH...]
           [-help [cmd]]
Generic options supported are
-conf <configuration file>     specify an application configuration file
-D <property=value>            use value for given property
-fs <local|namenode:port>      specify a namenode
-jt <local|jobtracker:port>    specify a job tracker
-files <comma separated list of files>    specify comma separated files to be
copied to the map reduce cluster
-libjars <comma separated list of jars>    specify comma separated jar files to
include in the classpath.
```

Working with HDFS and Files

- If you run `hadoop fs -ls` command as an ordinary user, e.g. `centos or root,` and do not specify the root of HDFS directory tree, `/user/,` you only see the directories and files that belong to you and reside above your home directory.
- Let's check

```
[root]$ hadoop fs -ls
```

- There might be nothing there if user `root` has no subdirectories.
- If you want to see all the subdirectories, in a way similar to Unix's `ls` with the `-r` option, you can use `hadoop fs -ls -R` command .

`[root]$ hadoop fs -ls -R` You'll see all the files and directories recursively.

`[root]$ hadoop fs -ls /    # /` is the root of the HDFS directory tree

- Command finds 12 items, some of which are:

```
drwxr-xr-x   - mapred hdfs     0 2017-11-10 14:38 /mapred
drwxrwxrwx   - mapred hadoop   0 2017-11-10 14:38 /mr-history
drwxr-xr-x   - hdfs   hdfs     0 2017-11-10 14:37 /ranger
drwxrwxrwx   - spark  hadoop   0 2018-01-12 23:36 /spark2-history
drwxrwxrwx   - hdfs   hdfs     0 2017-11-10 15:00 /tmp
drwxr-xr-x   - hdfs   hdfs     0 2018-01-12 23:18 /user
drwxr-xr-x   - hdfs   hdfs     0 2017-11-10 14:38 /webhdfs
```

Copying a file to new HDFS directory

- We are ready to add files to HDFS.
- We could fetch the .txt version of James Joyce's Ulysses by issuing the following command on the command prompt:

```
$ wget http://www.gutenberg.org/files/4300/4300-0.txt
```

- We can place file `4300-0.txt` into `ulysses` directory of user `root`

```
$ hadoop fs –mkdir ulysses
$ hadoop fs -put 4300-0.txt ulysses
$ hadoop fs –ls ulysses
[root]$ hadoop fs -ls ulysses
Found 1 items
-rw-r--r-- 1 root root 1580890 2018-01-12 19:54 ulysses/4300-0.txt
```

- The number 1 in the above listing tells us how many times is a particular file replicated. Since we have a single machine, 1 is appropriate.
- The replication factor is 3 by default, but could be set to any number.

Fetching and examining Files from HDFS

- The Hadoop command `get` does the reverse of `put`. It copies files from HDFS to the local file system.
- Let us first move file `/home/centos/4300-0.txt` to `/tmp`

```
$ mv 4300-0.txt /tmp
$ ls
```

- Nothing there.
- To retrieve file `4300-0.txt` from HDFS and copy it into the current local working directory, we run the command

```
hadoop fs -get ulysses/4300-0.txt .
```

- File `4300-0.txt` reappeared in /home/centos directory.
- A way to examine files in HDFS is to display data. For small HDFS files, Hadoop `cat` command is convenient.

```
hadoop fs -cat ulysses/4300-0.txt | wc -l
32710
```

- We can follow any Hadoop file command by Linux pipe to forward its output for further processing by a Linux commands. For example, if the file is huge (as typical Hadoop files are) and you're interested in a quick check of its content, you can pipe the output of Hadoop's `cat` into a Unix `head`.

```
hadoop fs -cat ulysses/4300-0.txt  | head
```

- Hadoop natively supports `tail` command for looking at the last kilobyte of a file.

```
hadoop fs -tail ulysses/4300-0.txt
```

Deleting Files and Directories

- Hadoop command for removing files is `rm`.

```
hadoop fs -rm ulysses/4300-0.txt
```

- To delete files and directories recursively use

```
hadoop fs -rm -R directory/*
```

- To delete empty directories use

```
hadoop fs -rmdir  directory
```

Running a MapReduce Example on YARN

- As user `centos`, we made a directory in HDFS called `input` and copied file `4300-0.txt` into that directory by running the following commands:

```
hadoop fs -mkdir input
hadoop fs -put 4300-0.txt input
hadoop fs -ls input
Found 1 items
-rw-r--r--   1 centos mapred    1580890 2018-01-12 23:57 input/4300-0.txt
```

- To run MapReduce examples we need to set `HADOOP_MAPRED_HOME` for user `centos.` Best, enter that environmental variable into `.bash_profile` file and then source that file. You can do it for a single use only, as well.

```
$ export HADOOP_MAPRED_HOME=/usr/hdp/2.6.3.0-235/hadoop-mapreduce
```

- Hadoop’s MapReduce jobs always read and write from and to HDFS files and/or directories. The above is a standard procedure with regular Hadoop’s MaprReduce jobs.

Running a MapReduce `grep` Job

- On our VM example MapReduce scrips (jobs) are contained in `$HADOOP_MAPRED_HOME/hadoop-mapreduce-examples.jar` file.
- One of the scripts is referred to as `grep` job which counts how many times every word appears in the analyzed corpus.
- In our case, Hadoop `grep` would scan the file (with James Joyce novel) placed in the specified (HDFS) directory `"input"` and create a tab delimited report named `ulysses_freq`.
- Hadoop `grep` uses simple regular expression `'\w+'` to select all multi-character words. In this regular expression `‘\w’`, with a lowercase `w`, represents word characters.
- This `grep` is different from Unix (Linux) `grep`. Unix `grep` returns lines where a pattern appears. Hadoop example `grep` counts word frequencies.
- The command to run Hadoop grep reads:

```
$ hadoop jar /usr/hdp/2.6.3.0-235/hadoop-mapreduce/hadoop-
   mapreduce-examples.jar grep input/4300-0.txt ulysses_freq '\w+'
```

- Job takes a few minutes. You could monitor progress of all map jobs and
- reduce jobs. The output is placed in HDFS directory `ulysses_freq`

WordCount in Hadoop MapReduce API

```
package org.apache.hadoop.examples; import java.io.IOException;
import java.util.StringTokenizer; import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class WordCount {
  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());  context.write(word, one);
      }
    }
  }
```

WordCount in Hadoop MapReduce API

```
public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {   sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}
public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();
  String[] otherArgs = new
      GenericOptionsParser(conf,args).getRemainingArgs();
  if (otherArgs.length < 2) {
    System.err.println("Usage: wordcount <in> [<in>...] <out>");
    System.exit(2);
  }
```

WordCount in Hadoop MapReduce API

```
    Job job = Job.getInstance(conf, "wordcount");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    for (int i = 0; i < otherArgs.length - 1; ++i) {
      FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
    }
    FileOutputFormat.setOutputPath(job,
      new Path(otherArgs[otherArgs.length - 1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Running a MapReduce example

- After a lot of console output, the job completes.
- We can find the output in the HDFS directory named `ulysses_freq`, because we specified that particular output directory to Hadoop.
- Type:

```
$ hadoop fs -ls
```

- You will see directories `input` and `ulysses_freq.` We list the content of `ulysses_freq`

```
$ hadoop fs -ls ulysses_freq
```

```
Found 2 items
-rw-r--r--   1 centos mapred          0 2018-01-13 00:17 ulysses_freq/_SUCCESS
-rw-r--r--   1 centos mapred     351794 2018-01-13 00:17 ulysses_freq/part-r-00000
```

- The content of the output file `part-r-hadoop00000` can be seen using `fs -cat` command:

```
[centos]$ hadoop fs -cat ulysses_freq/part-r-00000 | head
13686    the
8169     of
6697     and
5892     a
4872     to
4743     in
3063     his
2982     I
2970     he
2701     s
```

Introduction to Spark

Select Version of Spark

- Hortonworks VM comes with two versions of Spark, Spark 1.6.3, referred to as `Spark1` and Spark 2.2.0.2 referred to as `Spark2`.
- To choose one or the other version you set the environmental variable SPARK_MAJOR_VERSION to value 1 or value 2, like:

```
$ export SPARK_MAJOR_VERSION=1 or

$ export SPARK_MAJOR_VERSION=2
```

- The following slide presents the output of `pyspark`, the command shell used for Python Spark interactive programming, when the version is set to `Spark1`.

pyspark

- If you type `pyspark` on the Linux command prompt, in `centos` home directory, you will see the following:

```
[centos@sandbox-hdp ~]$ pyspark
SPARK_MAJOR_VERSION is set to 1, using Spark
Python 2.6.6 (r266:84292, Aug 18 2016, 15:13:37)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-17)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
18/01/13 00:31:18 INFO spark.SparkContext: Running Spark version 1.6.3
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
18/01/13 00:42:04 INFO netty.NettyBlockTransferService: Server created on 44301
18/01/13 00:42:04 INFO storage.BlockManagerMaster: Trying to register BlockManager
18/01/13 00:42:04 INFO storage.BlockManagerMasterEndpoint: Registering block manager
localhost:44301 with 511.1 MB RAM, BlockManagerId(driver, localhost, 44301)
18/01/13 00:42:04 INFO storage.BlockManagerMaster: Registered BlockManager
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.3
      /_/

Using Python version 2.6.6 (r266:84292, Aug 18 2016 15:13:37)
SparkContext available as sc, HiveContext available as sqlContext. >>>
>>> quit()
$
```

Load Data (RDD), Spark 1.6

- In Spark, we express our computation through operations on distributed collections that are automatically parallelized across the cluster.
- These collections are called *resilient distributed datasets*, or RDDs.
- When we load some data, i.e. a file into a shell variable, we are creating an RDD, like

```
>>> lines = sc.textFile("file:///home/centos/4300-0.txt")
>>> lines.count()
32742
```

- `4300-0.txt` is a text file residing in the home directory of user `centos`.
- What we've just done is create and populate an RDD named `lines` using a mysterious object "`sc`" and its method `textFile().` We populated RDD lines with data in file `4300-0.txt.`
- `"sc"` stands for an implicit `SparkContext`.
- `SparkContext` communicates with the execution environment.
- It appears that RDD-s are also (Object Oriented) objects and have methods, such as `count()` which gave use the exact number of lines in file `4300-0.txt`.
- We could ran the above load process using command:

```
lines = sc.textFile("file:///home/centos/4300-0.txt")
```

- Emphasizing that the file resides in a regular file system

Spark could read HDFS files

- We could have ran previous commands like:

```
lines2 = sc.textFile("/user/centos/ulysses/4300-0.txt")
lines2.count()
32710
```

- Spark, as this command shows, by default, reads from HDFS files. Owe could have emphasized use of HDFS by formulating this command as:

```
lines3 = sc.textFile("hdfs:///user/centos/ulysses/4300-0.txt")
lines3.count()
32710
```

Shell verbosity and `log4j.properties` file

- The output we have seen is long and annoying. We could shorten it by creating `log4j.properties` files in the directories:

```
/usr/hdp/2.6.3.0-235/etc/spark2/conf/log4j.properties.template
/usr/hdp/2.6.3.0-235/etc/spark/conf/log4j.properties.template
```

- You create those files by copying provided file `log4j.properties.template` and by changing line

```
log4j.rootCategory=INFO, console
```

- to read:

```
log4j.rootCategory=ERROR, console
```

- That lowers (raises?) the logging level so that we see only the ERROR messages, and above.
- You might want to replace all INFO levels with `WARN`, which is more verbose that ERROR but less than INFO.
- You might also want to replace all WARN levels with ERROR. Adjust those levels as you find convenient.
- If, after changing those log4j.properties files, you open pysparkpyspark you will see that the console output is much less verbose.

Spark 2

- If we run:

```
$ export SPARK_MAJOR_VERSION=2
$ pyspark
SPARK_MAJOR_VERSION is set to 2, using Spark2
Python 2.6.6 (r266:84292, Aug 18 2016, 15:13:37)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-17)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
18/01/13 01:23:44 WARN Utils: Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
/usr/hdp/current/spark2-client/python/pyspark/context.py:205: UserWarning:
Support for Python 2.6 is deprecated as of Spark 2.0.0
  warnings.warn("Support for Python 2.6 is deprecated as of Spark 2.0.0")
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.2.0.2.6.3.0-235
      /_/

Using Python version 2.6.6 (r266:84292, Aug 18 2016 15:13:37)
SparkSession available as 'spark'.
>>>
```

Load Data into DataFrame, Spark 2.2

- In Spark 2.2 RDDs are still around but you are advised to use a different type of data objects called DataFrame-s. Also, in Spark 2.2 SparkContext (object "sc") is implicit and you rather use spark session object, denoted by "spark".
- In Spark 2, previous commands would read:

```
>>> lines = spark.read.text("file:////home/centos/4300-0.txt")
>>> lines.count()
32742
>>> lines
DataFrame[value: string]
```

- In `spark-shell` which we use to work with Scala, you would use a very similar command:

```
scala> val textFile = spark.read.textFile("
file:////home/centos/4300-0.txt ")
textFile: org.apache.spark.sql.Dataset[String] = [value: string]
```

- Spark 2.x DataFrame is strongly-typed like an RDD, but with richer optimizations under the hood.
- DataFrame objects have better performance than RDDs.

Load Data (DataFrame) from HDFS

- We could load the same data from the Hadoops Distributed File System (HDFS)
- We will learn how to use HDFS in the next lecture.
- If we happen to have the `4300-0.txt` file in `centos` home directory in HDFS, we could do the following:

```
>>> blines = spark.read.text ("hdfs:///user/centos/4300-0.txt")
>>> blines.count()
32710
>>> blines.first()
Row(value=u'')
```

- What we did above was create an RDD named `blines` and populate that RDD with data from HDFS resident file `4300-0.txt`.
- We also see in action another method of DataFrames-s, `first(),` which tells us that the first line in `blines` is some uninterested collection of characters **(u'').**
- Please note that we are not terminating our commands with a semi-colon (";") or anything else aside from the carriage return. Savings on typing all those semi-colons is one of the greatest contributions of Python to the computer science.
- By the way, our commands are in Python

Examine a DataFrame

- DataFrame method show() list first 20 rows

```
>>> lines.show()
+--------------------+
|               value|
+--------------------+
|The Project Guten...|
                    |
|This eBook is for...|
|no restrictions w...|
|it under the term...|
|eBook or online a...|
|      Title: Ulysses|
|                    |
| Author: James Joyce|
|                    |
|Release Date: Aug...|
|Last Updated: Aug...|
|                    |
|   Language: English|
|                    |
|Character set enc...|
+--------------------+
only showing top 20 rows

>>>
```

`RDD filter()` Method

- RDD method `filter()` takes a function returning `True` or `False` and applies it to a sequence (list) and returns only those members of the sequence for which the function returned `True`.
- So, in the Python code :

```
heavens =lines.filter(lambda line: "Heaven" in line)
```

- Method `filter()` acts on the collection `lines`, and passes every element of that collection as the variable `line` as the argument to the anonymous function created using lambda construct. That anonymous function uses a simple regular expression to test whether string `"Heaven"` exists in variable `line`.
- If the regular expression returns `True` for a particular `line`, an element of collection `lines`, the anonymous function will return `True` and for that particular `line`, `filter()` will return/add variable `line` building up a new collection called `heavens`.
- You can accomplish the same in Java 1.7 and older with several lines of code. In Java 1.8 you have similarly efficient lambda constructs.

RDD `filter()` method

```
>>> lines = sc.textFile("/home/centos/4300-0.txt")
heaven = lines.filter(lambda line: "heaven" in line)
>>> heaven.count()
50
```

- We see in action a method of RDD-s, `filter()`, which apparently let us inquire how many times is `heaven` mentioned in the Ulysses.
- Heaven is mentioned 52 time, i.e. some 0.15% of the time (> Bible)

DataFrame filtering

- If we load data into a DataFrame, like

```
>>> dset = spark.read.text("file:///home/centos/4300-0.txt")
```

- and want to extract all lines with word `heaven` we would use slightly different syntax:

```
>>> dset.count()
32710
>>> heavens = dset.filter(dset.value.contains('heaven'))
>>> heavens.count()
47
```

Python `lambda` Syntax

- Python supports the creation of anonymous inline functions (i.e. functions that are not bound to a name) at runtime, using a construct called "lambda".
- The following code shows the difference between a normal function definition ("f") and a lambda function ("g"):

```
>>> def f(x): return x**2
>>> print f(8)
64
>>> g = lambda x: x**2
>>> print g(8)
64
```

- As you can see, `f()` and `g()` do exactly the same thing. The lambda definition does not include a "return" statement. The last expression is returned.
- You can put a lambda definition anywhere a function is expected, and you don't have to assign it to a variable at all.

Standalone Application in Python 1.6

- To create an application (Python script) we need to import some Python classes and create `SparkContext` object.
- The rest of the application is coded as if you are writing code in PySpark shell

```
from pyspark import SparkConf, SparkContext

conf = SparkConf().setMaster("local").setAppName("MyApp")
sc = SparkContext(conf = conf)
lines = sc.textFile("/user/centos/Ulysses/4300-0.txt")
lifeLines = lines.filter(lambda line: "life" in line)
print lifeLines.first()
```

- If we invoke the above with `spark-submit` script:

```
$ spark-submit my_script.py
```

- We get:

```
Arius, warring his life long upon the consubstantiality of
the Son with
```

Standalone App in Python, Spark 2.2

- Applications can be submitted to a cluster of any type using the `spark-submit` script.

```
from pyspark.sql import SparkSession

textfile = "/user/centos/Ulysses/4300-0.txt"  # some file on your HDFS
spark = SparkSession.builder.appName(appName).getOrCreate()
textfile = spark.read.text(textfile).cache()

numAs = textfile.filter(textfile.value.contains('a')).count()
numBs = textfile.filter(textfile.value.contains('b')).count()

print("Lines with a: %i, lines with b: %i" % (numAs, numBs))

spark.stop()
```

- This program just counts the number of lines containing ‘a’ and the number containing ‘b’ in a text file.

```
$ spark-submit your_script.py
```

Introduction to Hive

Access Hive, using `hive` command line client

- On the master node of your HDInsight cluster or on the command line of our Hortonworks HDP VM, type `hive` and Hive shell opens:

```
[centos@sandbox-hdp ~]$ hive
log4j:WARN No such property [maxFileSize] in
org.apache.log4j.DailyRollingFileAppender.

Logging initialized using configuration in file:/etc/hive/2.6.3.0-235/0/hive-
log4j.properties
hive> show tables;
OK
sample_07
sample_08
Time taken: 3.782 seconds, Fetched: 2 row(s)
hive> select * from sample_07 limit 10;
OK
00-0000 All Occupations 134354250       40690
11-0000 Management occupations  6003930 96150
11-1011 Chief executives        299160  151370
11-1021 General and operations managers 1655410 103780
11-1031 Legislators     61110   33880
11-2011 Advertising and promotions managers     36300   91100
11-2021 Marketing managers      165240  113400
11-2022 Sales managers  322170  106790
11-2031 Public relations managers       47210   97170
11-3011 Administrative services managers        239360  76370
Time taken: 1.775 seconds, Fetched: 10 row(s)
hive>
```

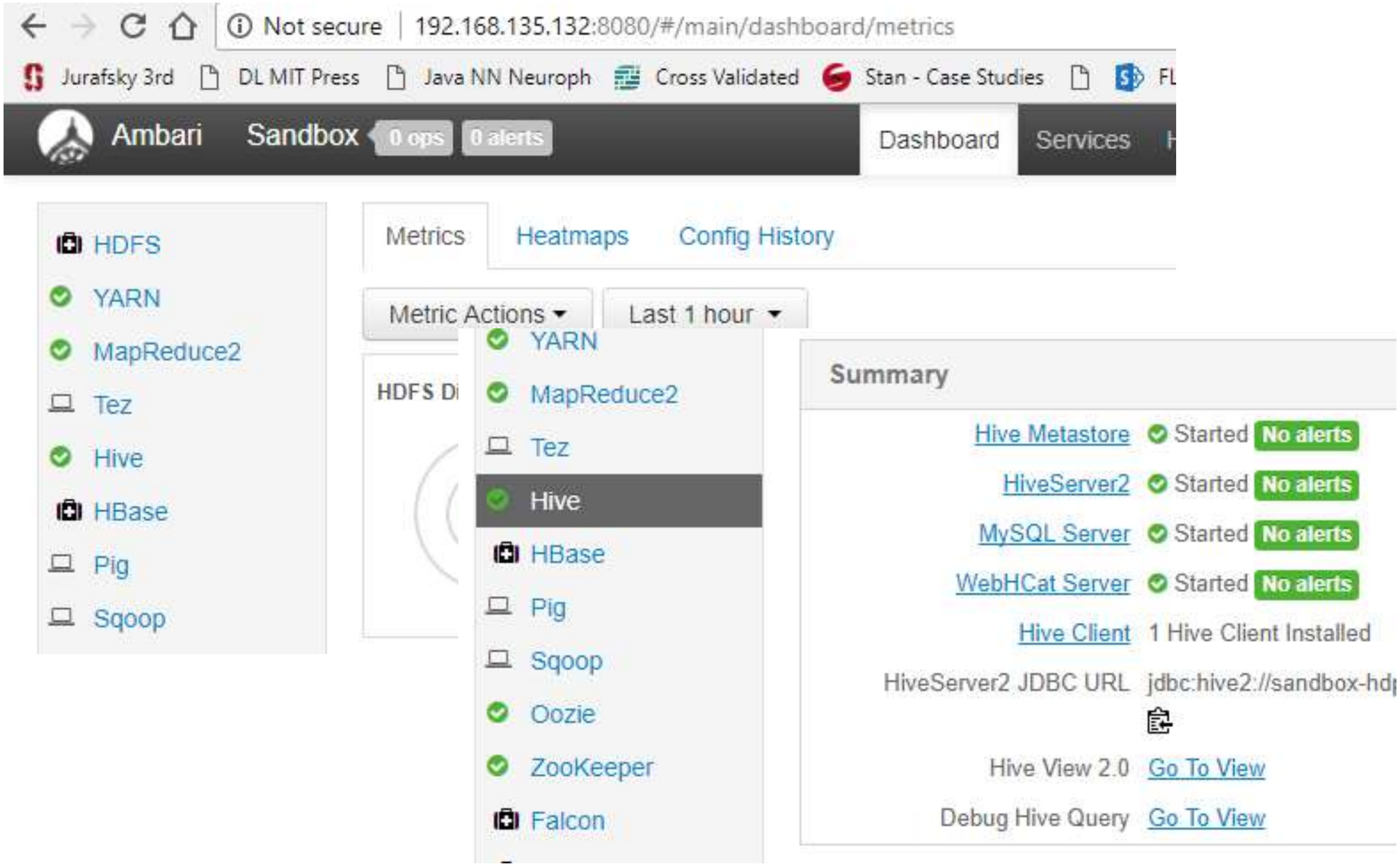
Examine Table Definitions

- To connect to hive, a new client for Hive you do this:

```
hive> desc sample_08;
OK
code                    string
description             string
total_emp               int
salary                  int
Time taken: 0.69 seconds, Fetched: 4 row(s)
hive> desc sample_07;
OK
code                    string
description             string
total_emp               int
salary                  int
Time taken: 0.638 seconds, Fetched: 4 row(s)
hive>
```

Hive Visual Client

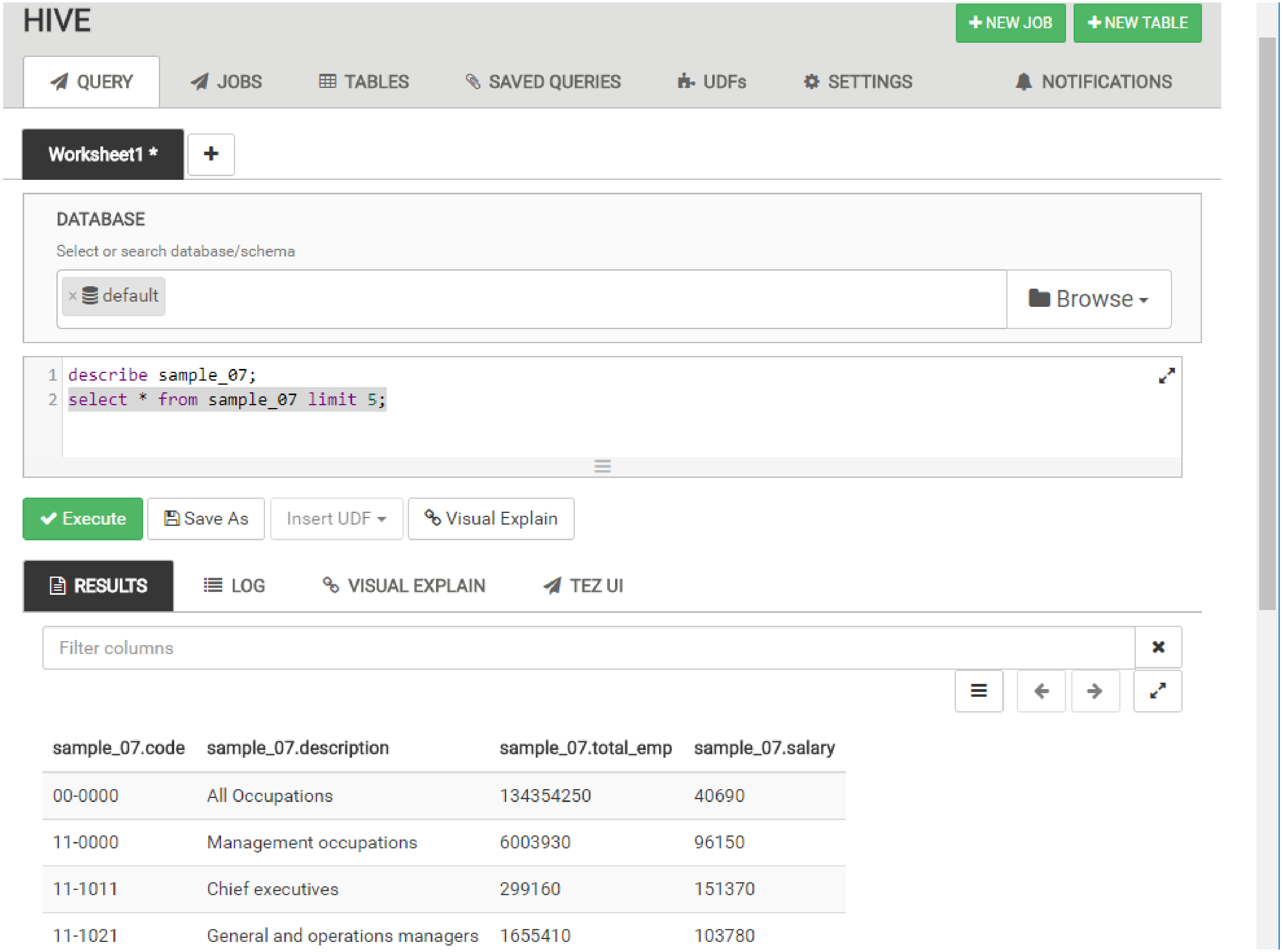
- Go to Amabari dashboard and select Hive in the left column.
- On the next screen select Hive View 2.0 Go to View



Summary

Hive Metastore ✓ Started No alerts
HiveServer2 ✓ Started No alerts
MySQL Server ✓ Started No alerts
WebHCat Server ✓ Started No alerts
Hive Client 1 Hive Client Installed
HiveServer2 JDBC URL jdbc:hive2://sandbox-hd
Hive View 2.0 Go To View
Debug Hive Query Go To View

Hive Query Editor



```
describe sample_07;
select * from sample_07 limit 5;
```

sample_07.code	sample_07.description	sample_07.total_emp	sample_07.salary
00-0000	All Occupations	134354250	40690
11-0000	Management occupations	6003930	96150
11-1011	Chief executives	299160	151370
11-1021	General and operations managers	1655410	103780

Sample Data, Shakespeare and King James

- On my machine there are two files:

  `all-shakespeare` and `all-bible`
- One contains complete works of Shakespeare and the other King James Bible. Both works use somewhat archaic form of English.
- You could examine the files by perhaps doing:

```
$ cat all-shakespeare | wc  or
$ cat all-shakespeare | tail -n 100

$ cat all-bible | wc  or
$ cat all-bible | tail -n 100
```

- We could transfer both files to our VM using `scp` command:

```
$ scp -P 2222 all-bible all-shakespeare centos@192.168.135.132:/home/centos
centos@192.168.135.132's password:
all-bible           100% 5135KB  27.4MB/s   00:00
all-shakespeare     100% 5160KB  47.4MB/s   00:00
```

Copy `all-shakespeare, all-bible` into HDFS

- We will copy these file into HDFS directories: `input_bible` and `input_shake` respectively:

```
$ hadoop fs -mkdir input_bible
$ hadoop fs -mkdir input_shake
$ hadoop fs -put all-bible input_bible
$ hadoop fs -put all-shakespear input_shake
```

- We could convince ourselves that the data inside HDFS is still the same old Shakespeare by typing something like:

```
 $ hadoop fs -cat input_shake/all-shakespeare | head -n 20

        ALL'S WELL THAT ENDS WELL

        DRAMATIS PERSONAE

KING OF FRANCE  (KING:)

DUKE OF FLORENCE        (DUKE:)

BERTRAM Count of Rousillon.

LAFEU   an old lord.

PAROLLES        a follower of Bertram.
```

Running a MapReduce `grep` Job

- Next we will use MapReduce grep job to count the number of word occurences in both texts.
- We will place respective outputs in HDFS directories `bible_freq` and `shake_freq:`

```
$ hadoop jar /usr/hdp/2.6.3.0-235/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input_bible bible_freq '\w+‛
```

```
$ hadoop jar /usr/hdp/2.6.3.0-235/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input_shake shake_freq '\w+'
```

Examine Result of `grep`

- Examine the HDFS directories by typing:

```
drwx------   - centos mapred          0 2018-01-13 14:40 .Trash
drwxr-xr-x   - centos mapred          0 2018-01-13 01:38 .hiveJars
drwx------   - centos mapred          0 2018-01-13 15:39 .staging
drwxr-xr-x   - centos mapred          0 2018-01-13 15:37 bible_freq
drwxr-xr-x   - centos mapred          0 2018-01-12 23:57 input
drwxr-xr-x   - centos mapred          0 2018-01-13 15:26 input_bible
drwxr-xr-x   - centos mapred          0 2018-01-13 15:27 input_shake
drwxr-xr-x   - centos mapred          0 2018-01-13 15:39 shake_freq
drwxr-xr-x   - centos mapred          0 2018-01-12 23:42 ulysses
```

Examine Content of the output files

- We could also see partial content of the output files:

```
[centos@sandbox-hdp ~]$ hadoop fs -cat bible_freq/part-r-00000 | head -n 10
62394   the
38985   and
34654   of
13526   to
12846   And
12603   that
12445   in
9764    shall
9672    he
8940    unto
cat: Unable to write to output stream.
[centos@sandbox-hdp ~]$ hadoop fs -cat shake_freq/part-r-00000 | head -n 10
25578   the
23027   I
19654   and
17462   to
16444   of
13524   a
12697   you
11296   my
10699   in
8857    is
cat: Unable to write to output stream.
[centos@sandbox-hdp ~]$
```

- These are frequency - word pairs, as expected.

Create tables to accept `grep` data

- Before opening `hive` client, and in particular before importing any data from HDFS, add your current Linux user, e.g. `centos`, to group Hadoop:

```
$ su –
Passwd:
$ usermod –aG hadoop centos
```

- We want to import of Shakespeare frequency data. We first create tables `shakespeare` and `bible` by issuing the following commands:

```
hive> create table shakespeare (freq INT, word STRING) ROW FORMAT
   DELIMITED FIELDS TERMINATED BY '\t' stored as textfile;
```

- This created table `shakespeare` with out any data

```
hive> show tables;
shakespeare
Time taken: 8.268 seconds
hive> describe shakespeare;
OK
freq    int
word    string
Time taken: 1.253 seconds
```

- Similarly, for bible:

```
hive> create table bible (freq INT, word STRING) ROW FORMAT
   DELIMITED FIELDS TERMINATED BY '\t' stored as textfile;
```

Load grep Data into `shakespeare` Table

- To load data we go back to the `hive` and type:

```
hive> LOAD DATA INPATH "/user/centos/shake_freq" OVERWRITE INTO
   TABLE shakespeare;     # From HDFS file system  or
hive> LOAD DATA LOCAL INPATH "/home/centos/part-r-00000" INTO TABLE
   shakespeare;     # From the local file system
Loading data to table shakespeare
OK
Time taken: 0.213 seconds
```

- OVERWRITE clause is optional. You need it to avoid duplicates or simply overwrite an existing table content.
- On the load command, Hive moved HDFS content of `shake_freq` into its own HDFS directory. That directory on Hortonworks HDP VM is located in HDFS directory:

```
/apps/hive/warehouse/shakespeare
```

- Please examine that directory using `hadoop fs` commands
- Note again, the directory `/apps/hive/warehouse` is in HDFS, not on Linux OS.

Verify that shakespeare has grep Data

```
hive> select * from shakespeare limit 10;
OK
25848   the
23031   I
19671   and
18038   to
16700   of
14170   a
12702   you
11297   my
10797   in
8882    is
Time taken: 0.095 seconds
hive>
```

- This statement read from the table (actually as part of optimization, it read directly from the HDFS file) and presented us with the first 10 lines.
- This is the same data we saw previously.

More Advanced Query

- Slightly more advanced query would perhaps be this one:

```
hive> SELECT * FROM shakespeare
WHERE freq > 100 SORT BY freq ASC
LIMIT 10;
```

- Notice that for a large data set this is not an entirely trivial job.
- Data has to be sorted before we could see 10 rows of words that have frequency just above 100.
- Notice how hive reports on map-reduce job it is starting.
- If the job takes too long you are given the job id and the command that you could execute to tell Hadoop to kill the job:

```
Starting Job = job_201404021324_0005, Tracking URL =
   http://quickstart:50030/jobdetails.jsp?jobid=job_201404021324_0005
Kill Command = /usr/lib/hadoop/bin/hadoop job  -
   Dmapred.job.tracker=quickstart:8021 -kill job_201404021324_0005
```

Even More Complex Query

- The "users", linguists perhaps, would like to know the number of words which appear with the most common frequencies.

```
hive> SELECT freq, COUNT(1) AS f2
  FROM shakespeare GROUP BY freq SORT BY f2 DESC LIMIT 10;
```

```
OK
1    13426
2    4274
3    2342
4    1502
5    1111
6    873
7    656
8    598
9    474
10   381
```

- This tells us that there are 13426 words that appears only once.
- 4274 words appear twice. 2342 words appear three times, etc.
- SQL command with minor deviation: ORDER BY is replaced by SORT BY.

Joining Tables

- One of the most powerful feature of Hive is the ability to create queries that joins tables together using regular SQL syntax.
- We have `(freq, word)` data for Shakespeare works and for King James Bible.
- We want to examine which words show up in both volumes of text.

Import data into bible

```
hive> LOAD DATA INPATH "/user/cloudera/bible_freq" INTO TABLE
   Bible;
OK
Time taken: 0.781 seconds
hive> select * from bible limit 20;
OK
62394    the
38985    and
34654    of
13526    to
12846    And
12603    that
12445    in
6913     be
6884     is
6649     him
6647     LORD
. . .
Time taken: 0.111 seconds
hive>
```

Create an Intermediate Table

- We need a table that will list most common words in both volumes with corresponding frequencies

```
hive> CREATE TABLE merged
         (word STRING, shake_f INT, kjb_f INT);
```

- For this table we do not need to specify how will date be stored.
- Hive will determine that by itself.

- Next, we will run a query that will select data from tables: `shakespeare` and `bible`, create a join and insert, i.e. overwrite the content of new table.
- In our case the table happens to be empty. If it were not empty and we insist on overwriting, table data would be lost. If we only perform an insert, new data would be appended to the old.

Populate `merged` table using `join`

```
hive> INSERT OVERWRITE TABLE merged
SELECT s.word, s.freq, k.freq FROM
shakespeare s JOIN bible k ON
(s.word = k.word)
WHERE s.freq >= 1 AND k.freq >= 1;
```

```
Query ID = centos_20180113172124_cc3bc7a0-928f-4c28-a36d-f6200dfa2d0d
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1515797541938_0017)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Map 2 ..........   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 9.64 s
--------------------------------------------------------------------------------
Loading data to table default.merged
Table default.merged stats: [numFiles=1, numRows=12432, totalSize=150351,
    rawDataSize=137919]
OK
Time taken: 19.161 seconds
```

Most common common words

- What the words that appear most frequently in both corpuses?

```
hive> SELECT word, shake_f, kjb_f,(shake_f + kjb_f) AS ss
FROM merged SORT BY ss DESC LIMIT 20;
the      25848    62394    88242
and      19671    38985    58656
of       16700    34654    51354
I        23031    8854     31885
to       18038    13526    31564
in       10797    12445    23242
a        14170    8057     22227
that     8869     12603    21472
And      7800     12846    20646
is       8882     6884     15766
my       11297    4135     15432
you      12702    2720     15422
he       5720     9672     15392
his      6817     8385     15202
not      8409     6591     15000
be       6773     6913     13686
for      6309     7270     13579
with     7284     6057     13341
it       7178     5917     13095
shall    3293     9764     13057
```

To examine common non-Stop Word, go deeper

```
SELECT word, shake_f, kjb_f, (shake_f + kjb_f) AS ss
FROM merged SORT BY ss DESC LIMIT 200;
. . . . .
heaven  626    578    1204
When    847    349    1196
Of      1006   63     1191
most    1017   135    1152
where   813    335    1148
tell    960    188    1148
blood   699    447    1146
doth    961    63     1146
set     451    694    1145
It      890    241    1131
ever    634    475    1109
Which   977    130    1107
whom    375    732    1107
Time taken: 46.988 seconds
```