Design and Data Structure

Data Structure Design

In this study, we want to develop a platform that collects data from multiple sources in the Department of Planning and Agricultural Economics, where it collects the data available in CSV format, MySQL database, and the current weather data from API Stream. The benefit is providing the data necessary to carry out an effective analysis process that helps make wise and more scientific decisions quickly and easily. The platform consists of:

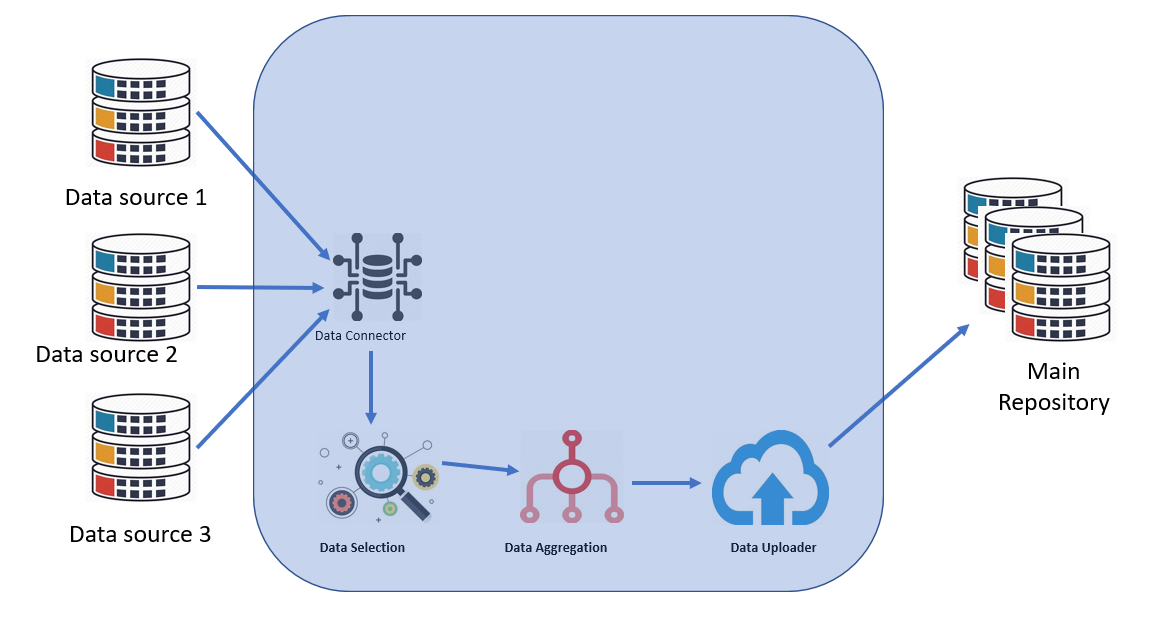
data ingestion: In this layer, we make sure to obtain data from the three different sources to be sent to the storage layer to ensure that it is stored in one warehouse. This process will be done automatically at specified intervals. It performs four functions:

**Data Connector:** This function enables the data integration subsystem to connect to the data sources of the external systems.

**Data Selector**: This function shall be responsible for selecting source data for data integration from different data sources.

**Data Aggregation**: This function shall be responsible for aggregating data selected by the Data Selection function.

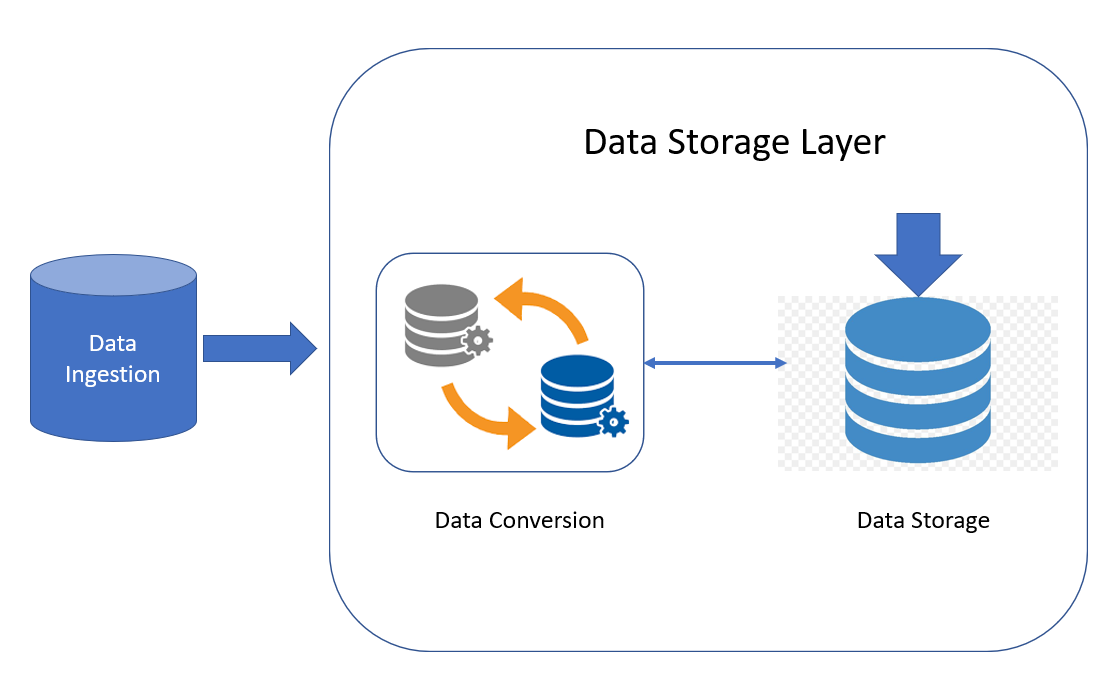
**Data Uploader:** Shall be responsible for uploading aggregated data to the main repository repository.

data ingestion layer

Data storage: The main responsibility for data storage layer is storing data and converting them into formats that are easy for analysis tools to handle. The data will be placed in HDFS with the use of hive tables, which we discussed in Chapter Two. It provides us with data access when needed while ensuring that it is stored. It also converts data into formats understood by analysis tools. So it offers two functions:

**Data conversion function:** This function converts the data in a format that analytical tools can understand.

**Data Storage Function:** Ensure that the data is stored in the main repository.



Analysis and visualization: We will use in this layer the analysis tools that provided by Hadoop, to analyze the data that we previously stored, and present it in a form, that is easy to understand for the end-user as it is presented in the form of chart and diagrams and this shows the benefit that this platform can contribute. This layer consists of three components

Data Selection Planner:Witch performs three functions.

**Data Connector:** Connects to the data storage layer.

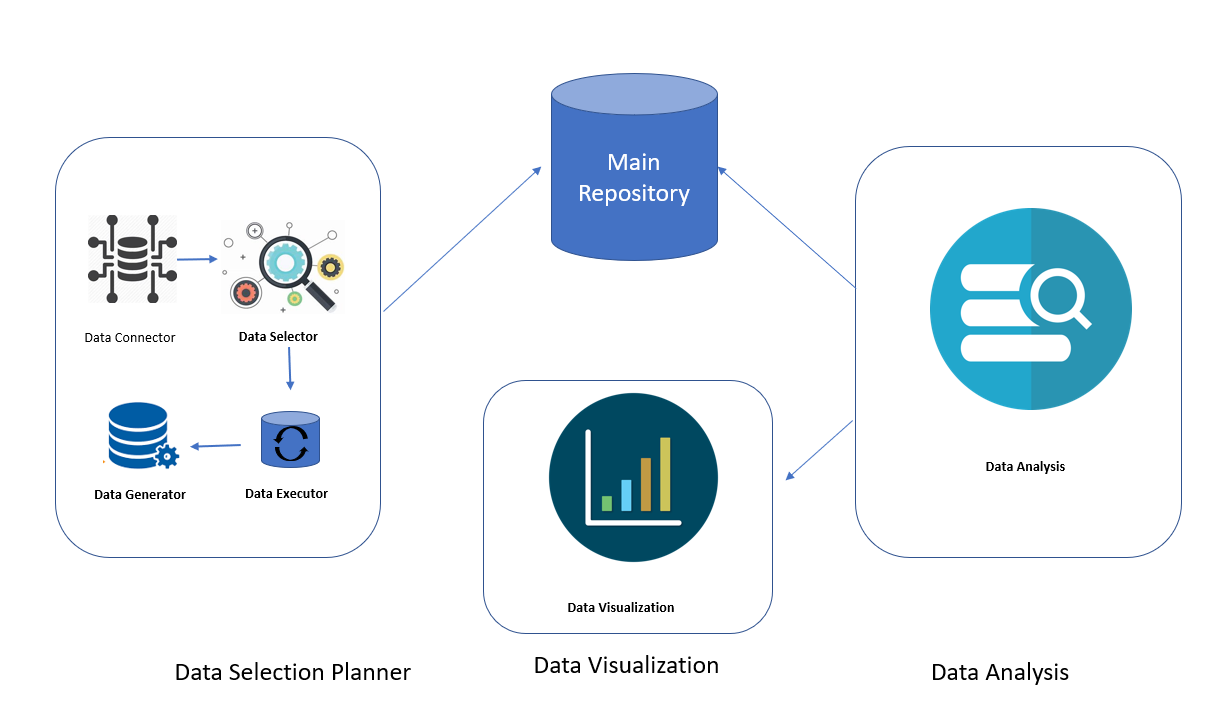
**Data selector:** this function enables one to select data stored in the main repository.

**Data Executor:** This function shall execute the selection plan.

**Data Generator:** This function shall generate a new dataset from the selected data.

#### Data Analysis Component (DAC): This component shall be responsible for performing data analysis on the data ingested to the central repository.

Data Visualization Component (DVC)**:** The data visualization componentshall be responsible for presenting the analyzed data in a form that would be easy to understand by end-users. It has only one function that enables the data to be presented in the form of bar charts, pie charts, histograms, line graphs, etc.



Data analysis and visualize layer

**Development Tools**

We used different tools from the Hadoop core and Hadoop echo system to build this platform. There are divided on platform layer as follow:

**Development tools for Data Ingestion Layer**

The researcher used Apache Nifi to build the data ingestion layer. Apache NiFi is a tool to migrate data automatically from one external system to another. In Apache Nifi, “data flow” is used to mean the automated and managed flow of information between systems [Apache Nifi Overview. [Online] Available:https://nifi.apache.org/docs/nifi-docs/html/overview.html.[Retrieved April 2019]].

Apache Nifi is a web-based tool that provides an experience between the design, control, feedback, and monitoring of a data ingestion platform. It is easy to configure and ensures guaranteed data ingestion, low latency, high throughput, and enables tasks to be dynamically prioritized. With Apache Nifi, flows can be modified at runtime without any disruption. It provides data provenance service, which allows dataflow to be tracked from the beginning to the end. Developers can rapidly build processors to ingest data from external sources and performs effective testing with this tool. It has very little security vulnerabilities as it supports SSL, SSH, HTTPS, encrypted content, and provides authorization for multiple tenants, internal authorization/policy management. [Apache Nifi Overview. [Online] Available:https://nifi.apache.org/docs/nifi-docs/html/overview.html.[Retrieved April 2019]].

Some of the core concepts of Apache Nifi are:

Flow File: Represents each object moving through the system. It represents a single piece of data. It is made up of two components, which are flow file attributes and flow file content. The flow file content is the data, and the flow file attribute provides information about the data.

Processors: The Nifi component responsible for listening for incoming data, pull data from external sources, publish data to external sources, and route, transform or extract information from flow files.

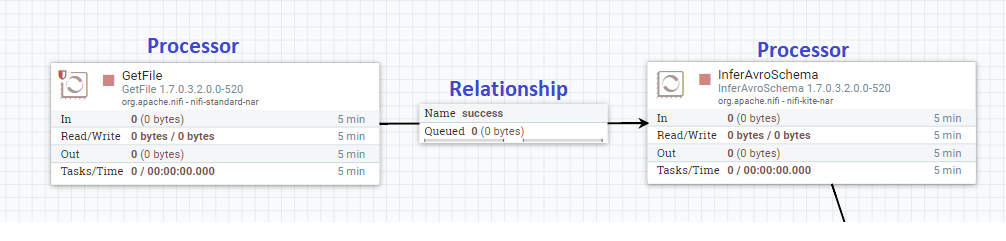
Relationships: An association between processors. A processor transfers the flow file to one of the relationships after it has finished processing it. The Data Flow manager then connects each of these relationships to other components to specify where the flow file should go next.

Controller Service: Controller service automatically starts up when NiFi starts up. It also provides information necessary for other components to use. They are being added and configured by the Data Flow Manager.

Process Group: Process Groups are used to group multiple components and mostly used in cases when a data flow becomes too complicated.

Port: Ports are used to connect two or more process groups. Input ports are used for incoming flow files into a process group, and output ports are used for outgoing flow files from a process group.

process group

processor relationship

**Development tools for the Data Storage layer**

to build the Data Storage layer the following tools are used

#### Apache Hadoop

Apache Hadoop is a software framework used for distributed storage and processing of enormous data sets on different clusters of computers built from hardware that are affordable and easy to obtain. It is, however, open source. Hadoop is built with the notion that hardware failures are common things that occur, and the framework should have the capability to handle such. Apache Hadoop enables data storage, data processing, data access, data governance, security, and operations.[29]

We speak about apache Hadoop in detail in chapter 2. So here we just want to mention which part of apache we will use in this layer.

HDFS: is used to store the central part of data to ensure that our data are stored in distributed manner.

MapReduce: makes it easy to write applications that are capable of processing a massive amount of data in parallel and on large clusters of affordable hardware in a reliable, fault-tolerant manner.

Apache Hive: as we said in chapter 2, it is a data warehouse software framework built on top of Hadoop

Development tool for the Data Analysis and Visualization layer

Apache Spark SQL

Apache Spark SQL and Apache Hive are used to perform data analysis on the stored data in the data storage layer. Apache Spark SQL is integrated, provides unified access, easily integrates with HIVE, provides standard connectivity and scalable

**Apache Zeppelin**: With Apache Zeppelin, Apache SparkSQL is configured to connect to the Hive warehouse, execute queries, and visualize the result of the analysis. HiveQL is also set on this editor to connect to the HIVE warehouse and execute HiveQL queries.

System implementation

To implement the platform according to the requirement. We use the following tools, as we mention in the data structures design part.

Apache Nifi is used in the development of the Data Integration Subsystem. Apache Hadoop and Apache Hive are used for the Data Storage Subsystem. Finally, Apache Hive and Apache Spark SQL are used for the Data Analysis and Visualization Subsystem. Apache Hadoop HDFS, Apache Hive, Apache Spark services are installed on the centos Operating system cluster of 5 nodes using Clodera Manager (CDH). We use a virtual machine to build a cluster consists of five machines: one master node and four slaves node.

**The base machine**

To download, install, and configure Centos Operating System, We download centos DVD iso 7.2 build no 1511. Install it on the virtual machine with the following parameter

Two GB of RAM

80gb HDD

Two processor

**Make the cluster**

To build a five machine cluster with the following configuration

|  |  |  |
| --- | --- | --- |
| Machine name | Machine IP | Type |
| Mgt01 | 10.0.2.100 | Master Node |
| Dn01 | 10.0.2.101 | Slave Node |
| Dn02 | 10.0.2.102 | Slave Node |
| Dn03 | 10.0.2.103 | Slave Node |
| Dn04 | 10.0.2.104 | Slave Node |

And we need to make a NAT network to let those machines seen each other.

So we make a full clone for the base machine. And to change network configuration we modify /etc/sysconfig/network-scripts/ifcfg-enp0s3 as following:

TYPE=Ethernet

BOOTPROTO=static

DEFROUTE=yes

PEERDNS=yes

PEERRPUTES=yes

IPV4\_FAILOURE\_FATAL=no

NAME=enp0s3

DEVICE=enp0s3

ONBOOT=yes

IPADDR=10.0.2.100

DNS1=8.8.8.8

DNS2=8.8.4.4

GATEWAY=10.0.2.1

NETMASK=255.255.255.0

Disable and stop firewall

systemctl disable firewalld

systemctl stop firewalld

Disable selinux

vi /etc/selinux/config

change selinux to selinux=disabled

Set hostname

Use nmtui command

Modify /etc/host

This will make the machine see each other

#vi /etc/host

10.0.2.100 mgt01.cloudera mgt01

10.0.2.101 dnt01.cloudera dn01

10.0.2.101 dnt02.cloudera dn02

10.0.2.101 dnt03.cloudera dn03

10.0.2.101 dnt04.cloudera dn04

#service network restart

We can clone mgt01 with reinitialize mac address and make the necessary change for hostname and IP address

Installing Cloudera Manager

First, we have to change the RAM capacity for mgt01 from 2 GB to 8gb with two core

# wget <https://archive.cloudera.com/cm5/installer/latest/cloudera-manager-installer.bin>

# chmod u+x cloudera-manager-installer.bin

# ./cloudera-manager-installer.bin

Connecting the cluster

To connect the cluster, we need to enable port forwarding from nat network

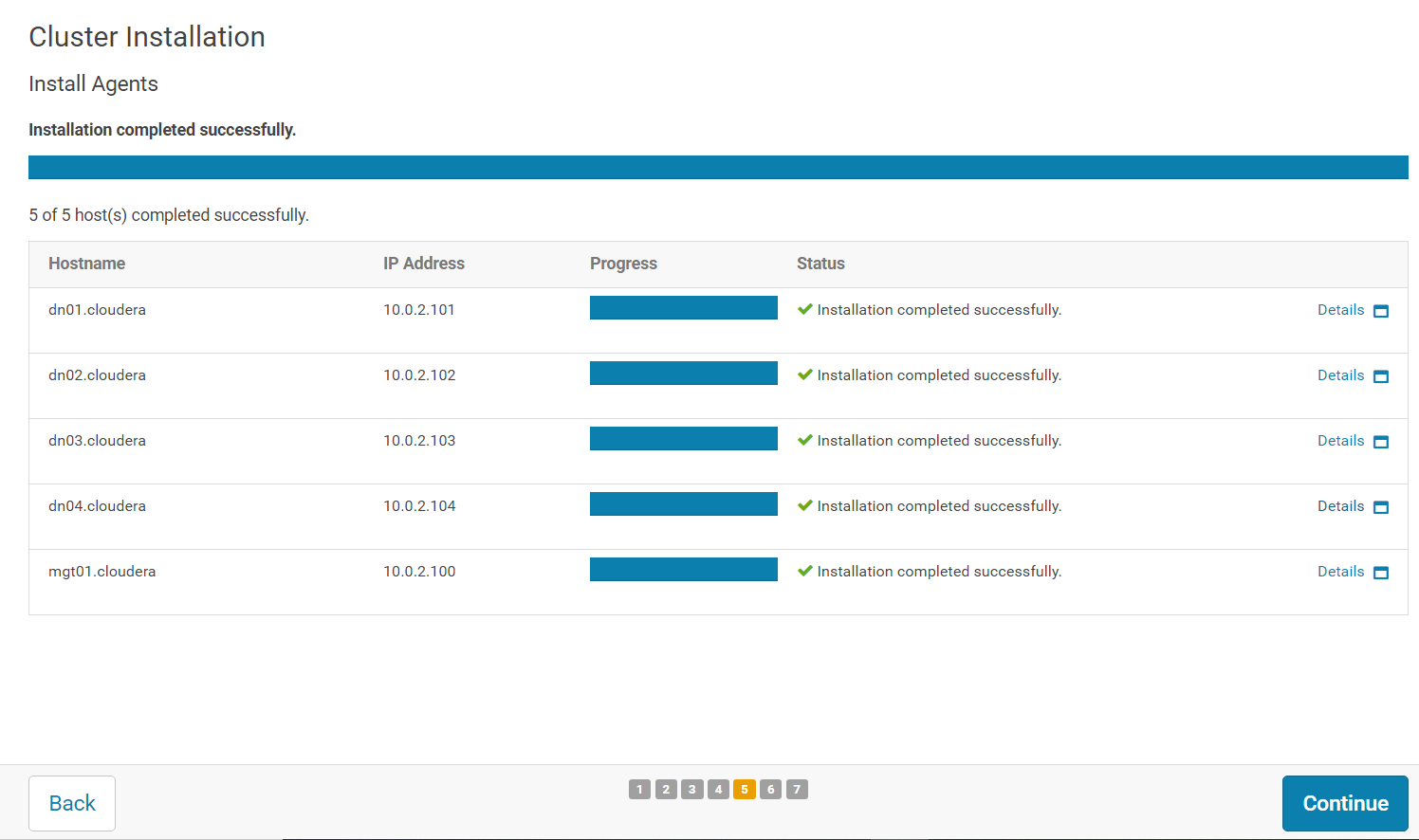
Deploying the cluster

To reach the master node in the purpose of deploying Hadoop to all nodes we use our host website to connect

Localhost: port

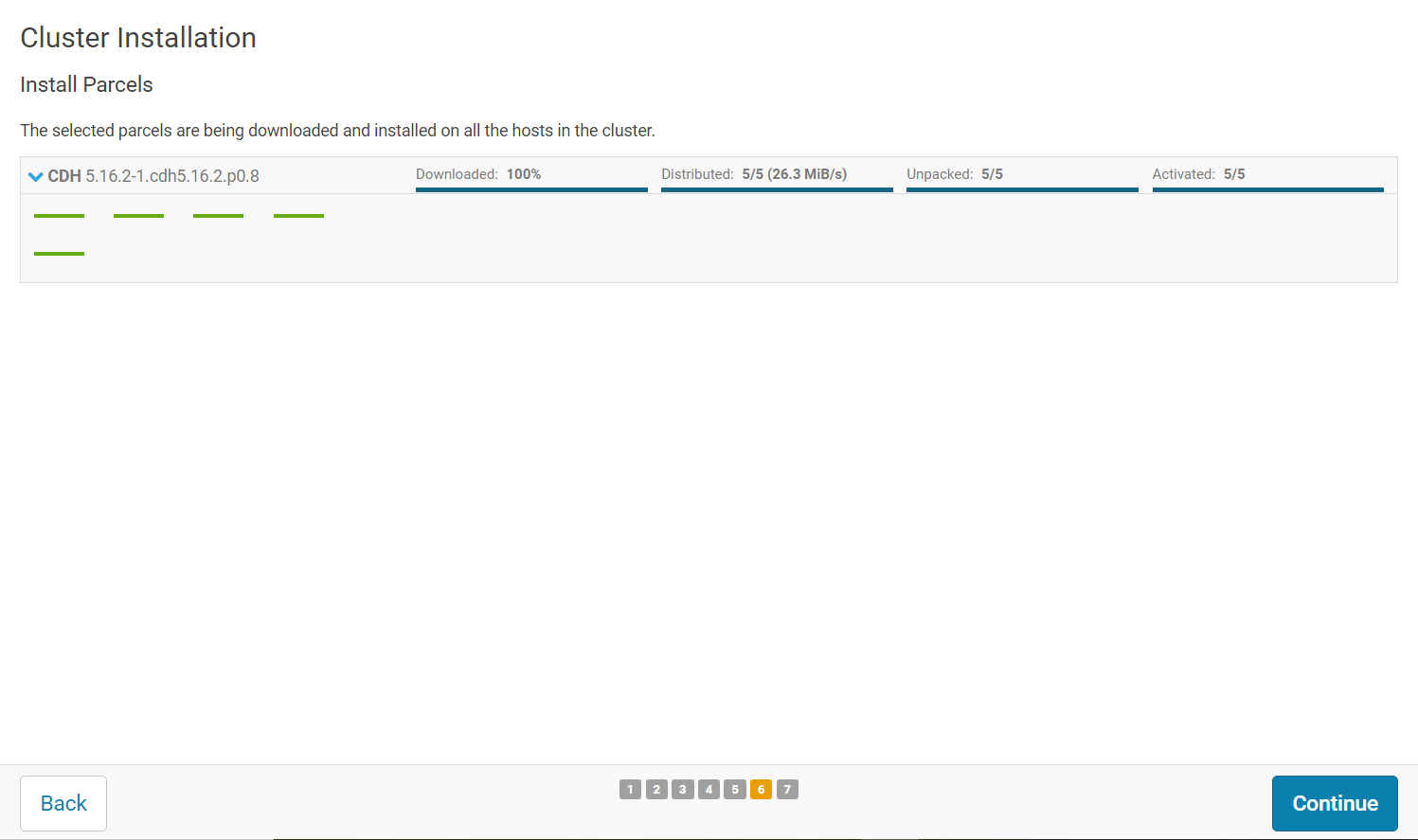
Will start a wizard with the following screen

1. End-User license
2. Cloudera edition. Here we will select express
3. Specify the host: here, we will use the pattern :
   1. dn01
   2. dn02.cloudera
   3. dn[03-04]
   4. 10.0.2.100
4. Select the repository
   1. Use Parcels as a method
   2. 5.9.0 for CDH
   3. Match release for this Cloudera manager server
5. Select JDK Installation Options
6. We won’t Enable single-user mode for security reasons. it makes one user control all services
7. Provide SSH login credentials; there are tow way eather providing ssh key or by using user name and password and that what we will going to chose
8. Cluster installation starts. It consists of many processes in order to deploy it on all five node



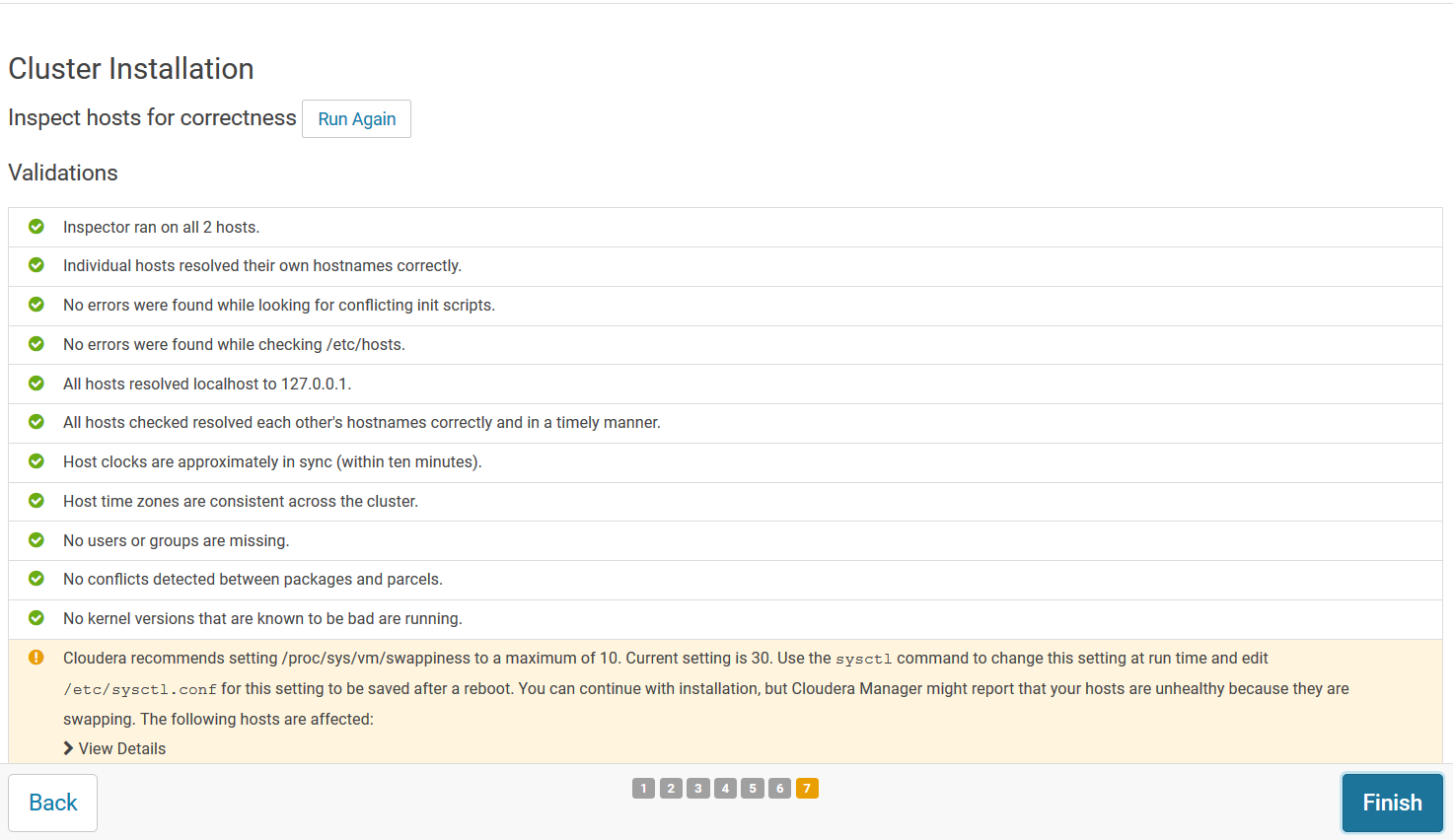
Finished Cluster installation

1. In the last step,, we only deploy the agent, and now we need to deploy the services. So Cloudera manager in this step will start downloading, distributing, unpackaging and activating the parcels we need for our cluster



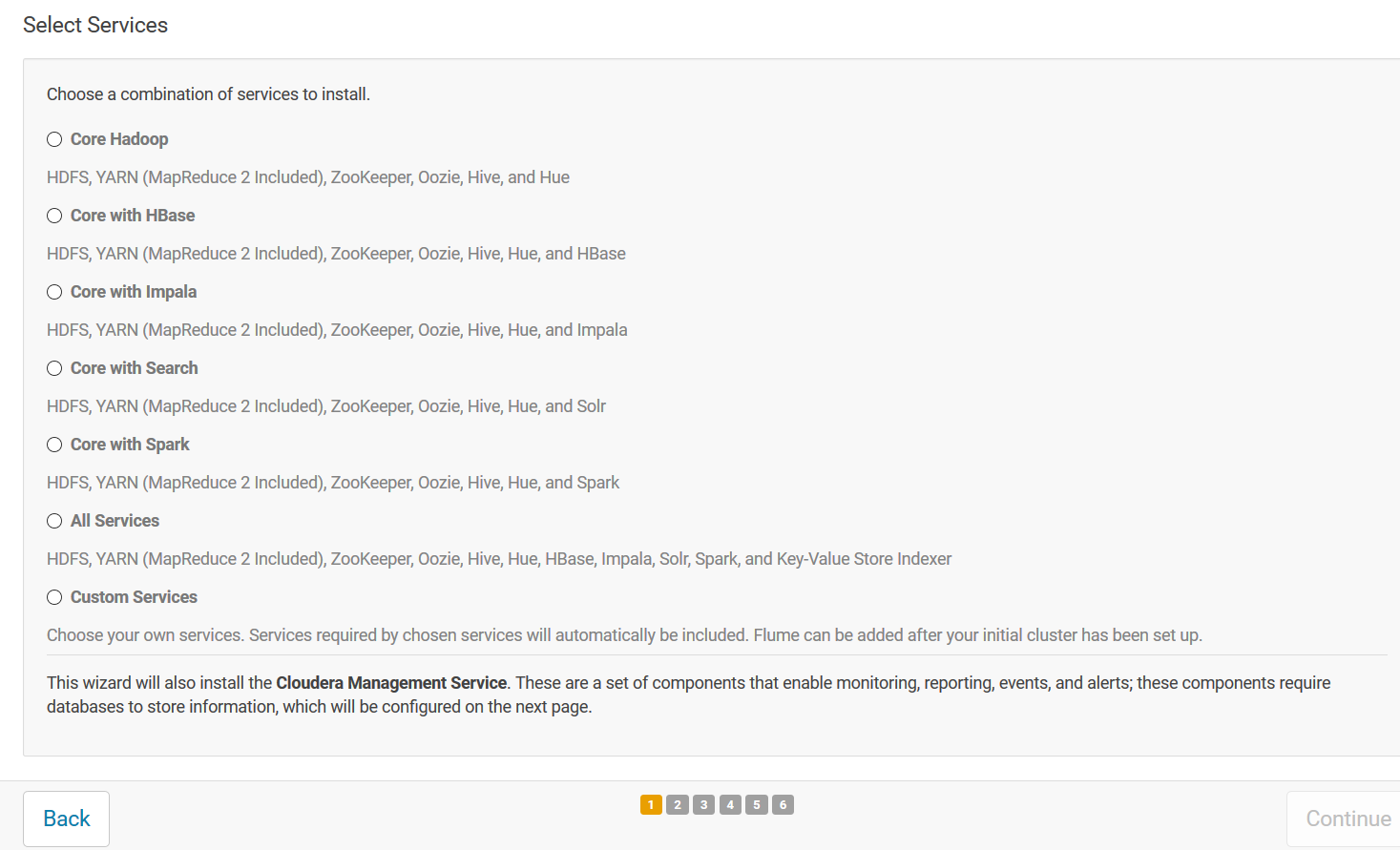
cluster installation step 6-finish

1. CM starts to inspect hosts for the correctness. it Will show us a summary with any suggestion needed



Finishing installation of the cluster and inspect host for the correctness

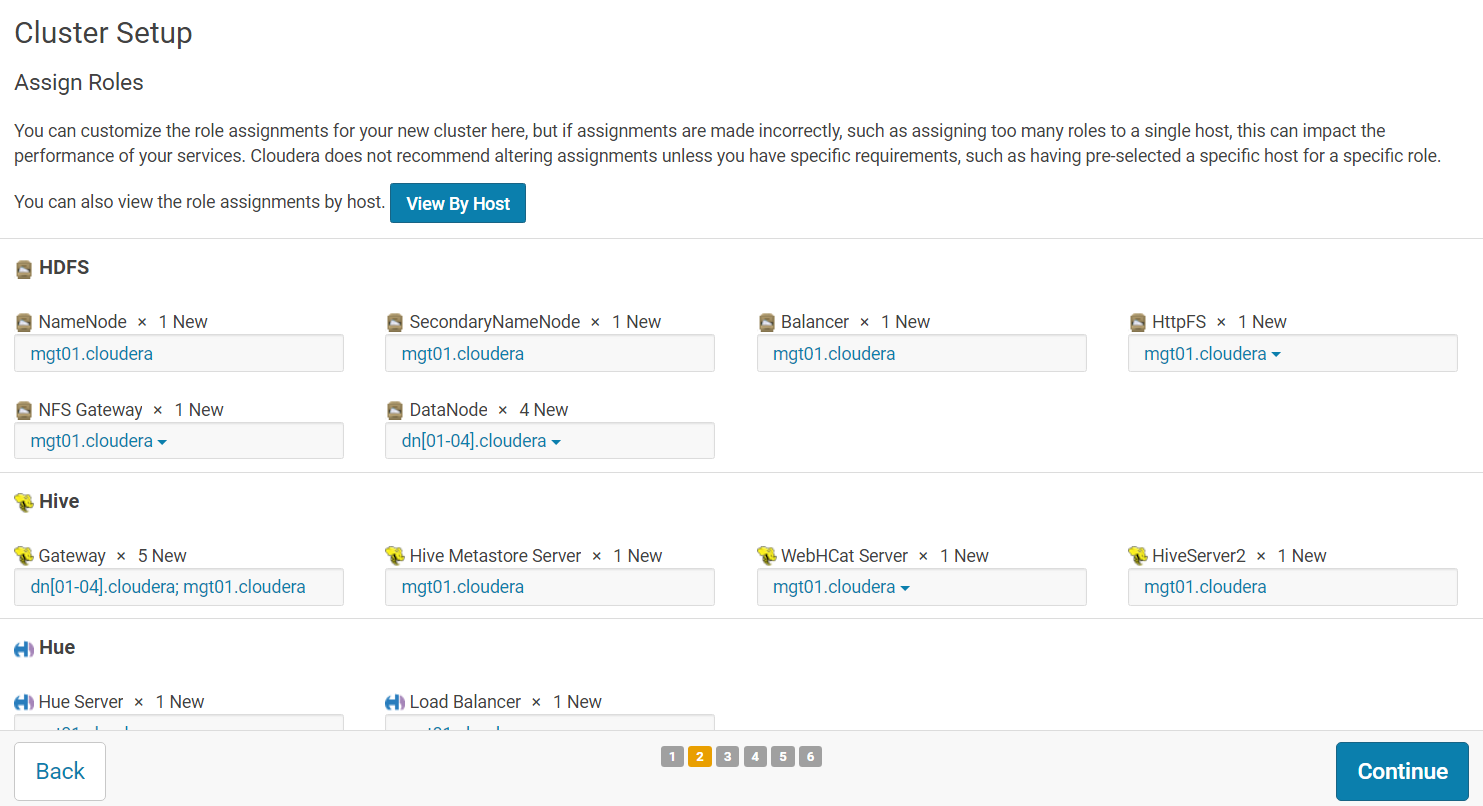
1. After finished the all previous step then we selected CDH services that we need, we started with just Core Hadoop which consist of HDFS, YARN included MapReduce 2, Zookeeper, Oozie, Hive, and Hue.



Select the type of services to install

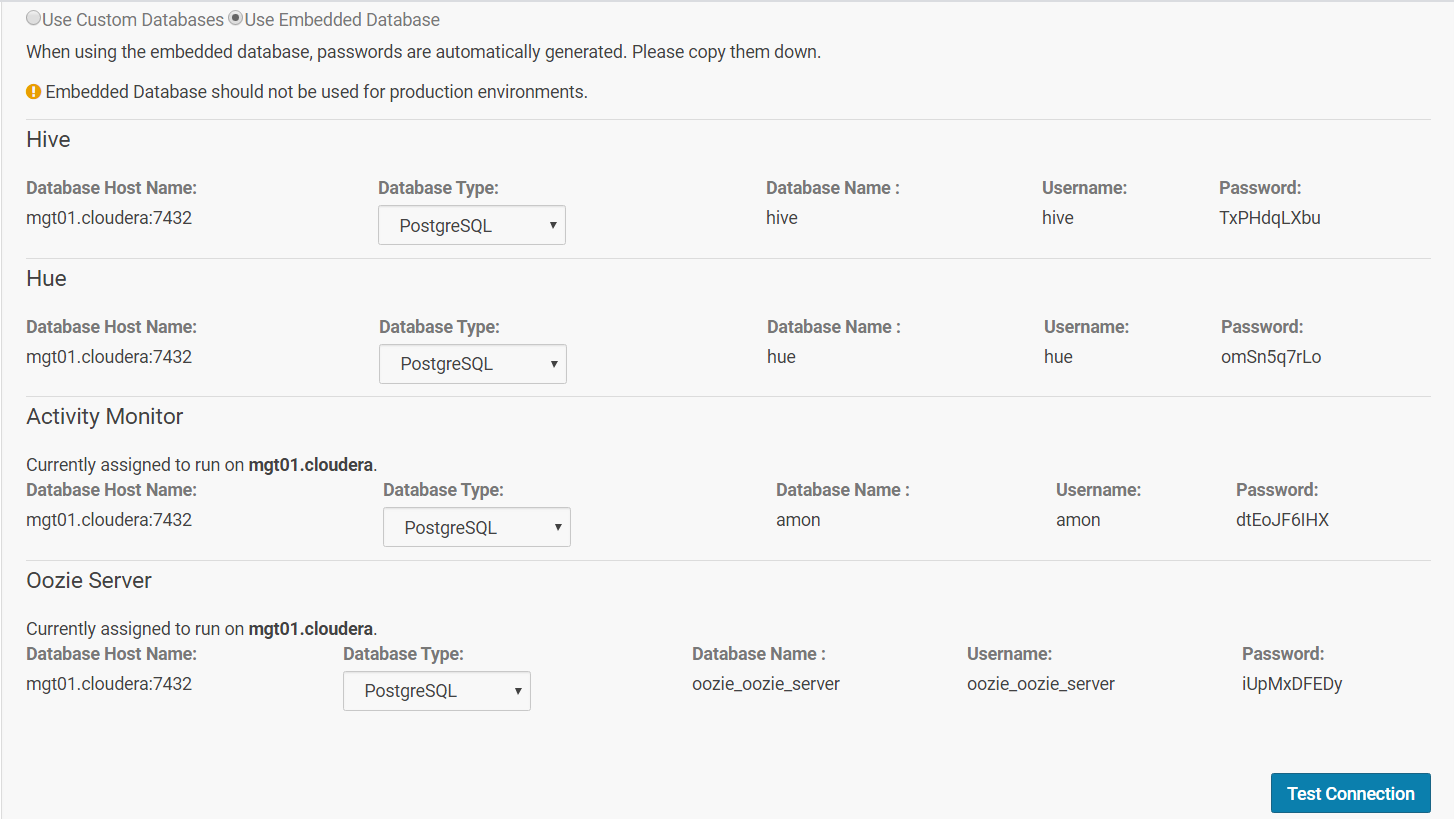
1. Customize Role Assignments: to specify the role assignment for the new clusters. We configure it by the following settings

|  |  |  |
| --- | --- | --- |
| Services | Node |  |
| HDFS | mgt01 |  |
| SecondaryNameNode | mgt01 |  |
| Balancer | mgt01 |  |
| HttpFS | mgt01 |  |
| DataNode | mgt01 |  |
| NFS Gateway | mgt01 |  |
| Hive Gateway | mgt01, dn[01-04] |  |
| Hive Metastore Server | mgt01 |  |
| Hive WebHCat | mgt01 |  |
| Hue | mgt01 |  |
| Activity Monitor | mgt01 |  |



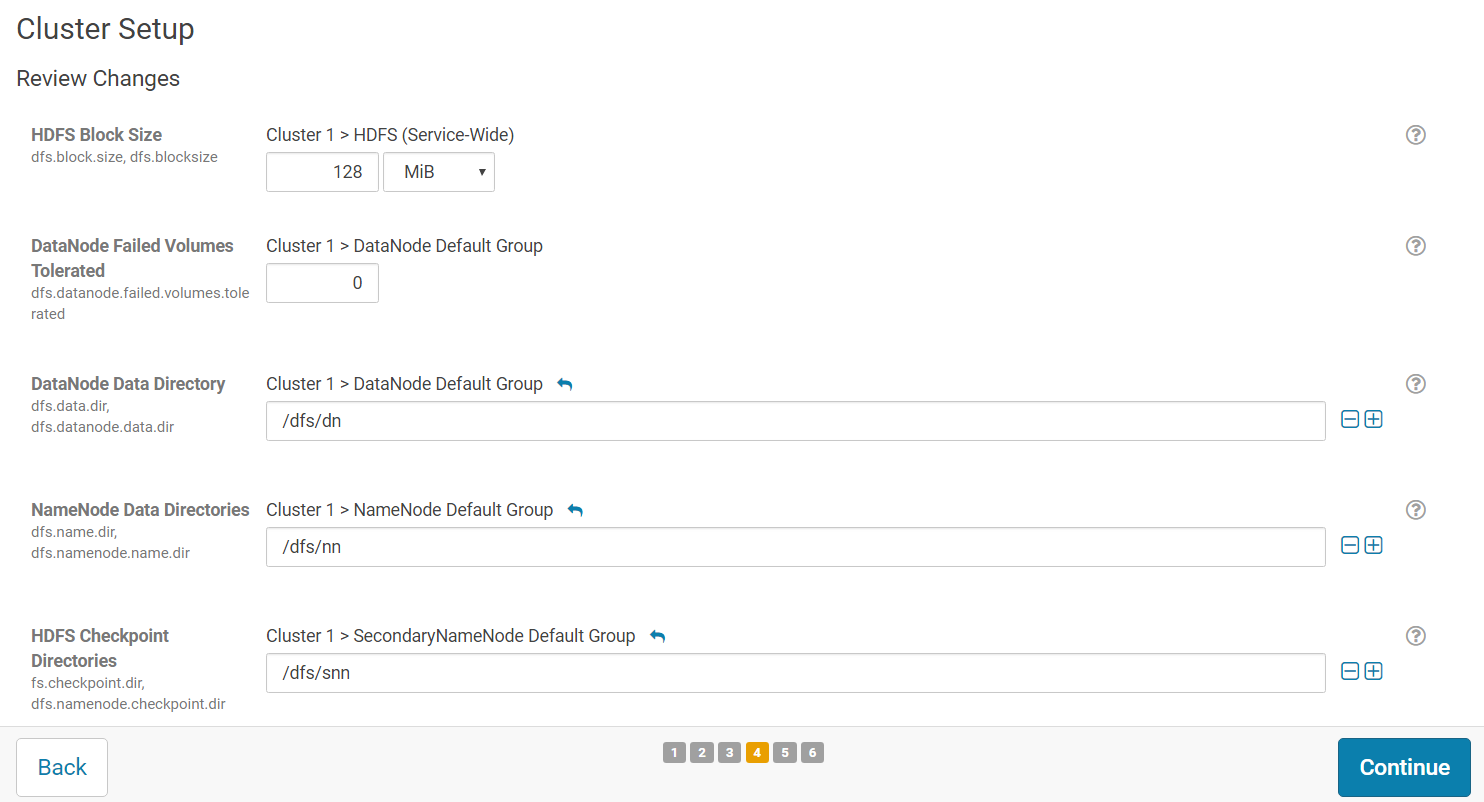
Assigning rule to node

1. Database Setup: to configure and test the database connection for each service. Here we use an embedded PostgreSQL database. And the important point here to take a note about the credentials the setting by default as we may need it for later use.



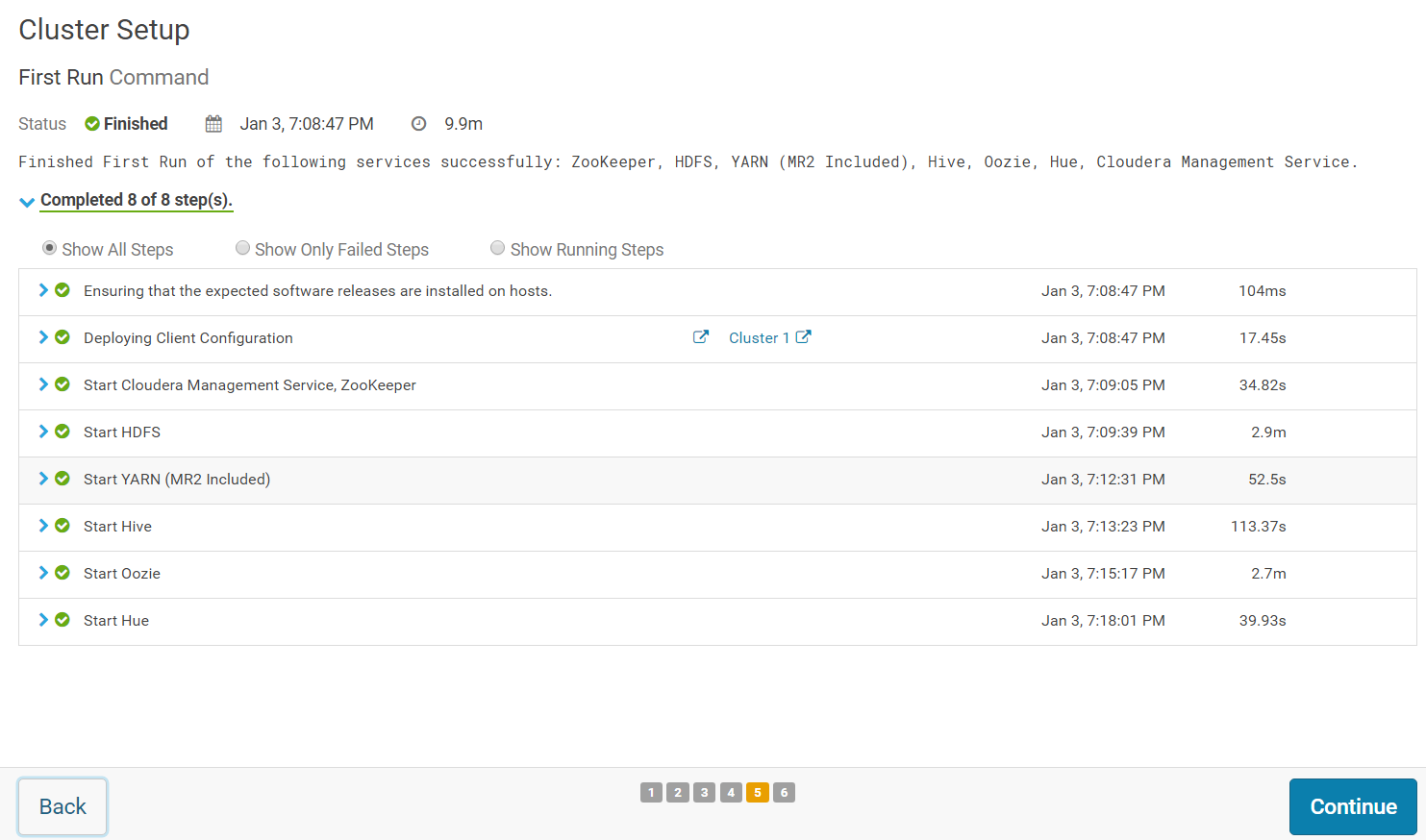
The default setting for embedded database

1. We can change the block size and directory for services. But we left every thing as default.



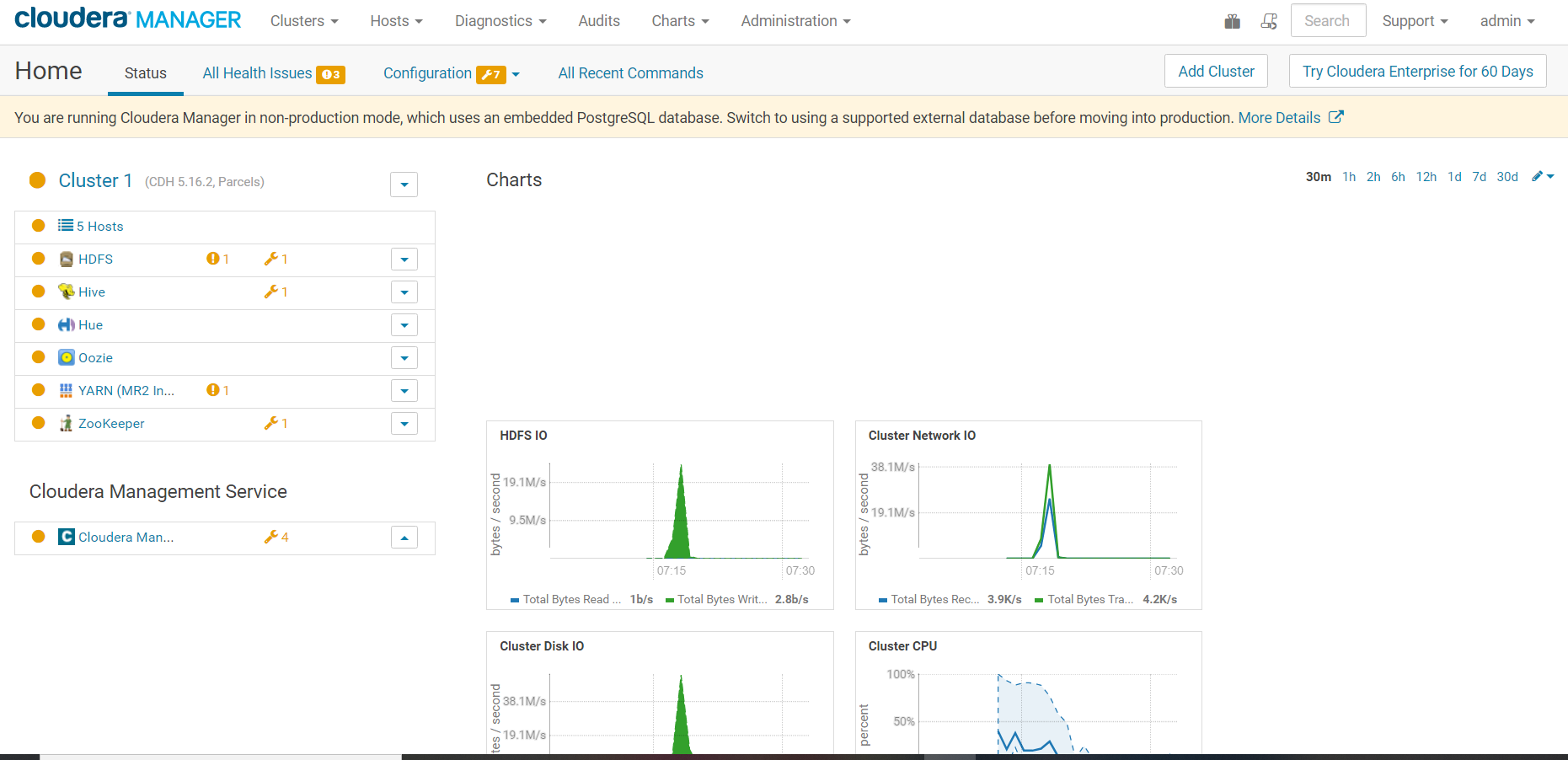
Setting the block size

1. CM will run all services for first time



Run all services to check

1. Then it will take us to the main administrative screen of Cloudera manager wich give us a full overview of our cluster. Where we can start, stop, restart, install, and delete our services.



Cloudera manager main screen

Installing NiFi in CDH

1. Get the CFM CSD files

Firest we have to connect to Cloudera Manager node using the root account, we had adding the following:

cd /opt/cloudera/csd

wget <http://archive.cloudera.com/CFM/csd/1.0.0.0/NIFI-1.9.0.1.0.0.0-90.jar>

wget <http://archive.cloudera.com/CFM/csd/1.0.0.0/NIFICA-1.9.0.1.0.0.0-90.jar>

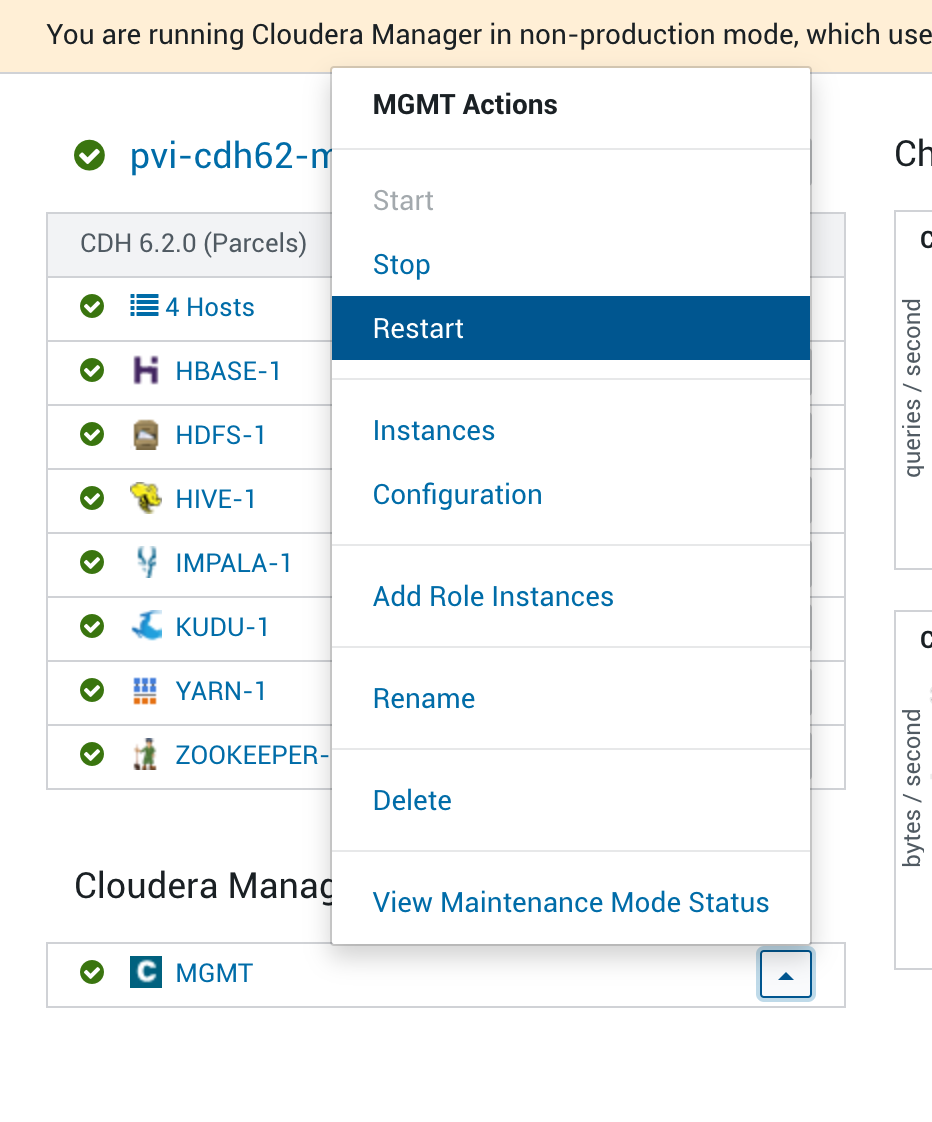
wget <http://archive.cloudera.com/CFM/csd/1.0.0.0/NIFIREGISTRY-0.3.0.1.0.0.0-90.jar>

chown cloudera-scm:cloudera-scm NIFI\*.jar

chmod 644 NIFI\*.jar

service cloudera-scm-server restart

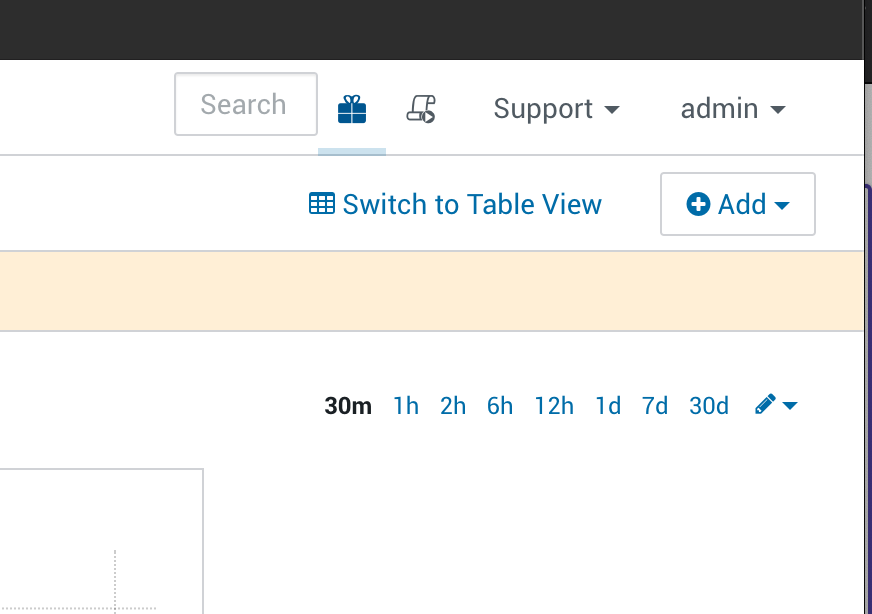
when the restart operation done, we loged to CM and restart the Cloudera Management Service:



Restarting the Cloudera Management Service

1. Get the CFM Parcel

Getting CFM parcel is important to install NiFi on ClouderaCDH, we achieved that from parcels configuration (parcel icon top right):



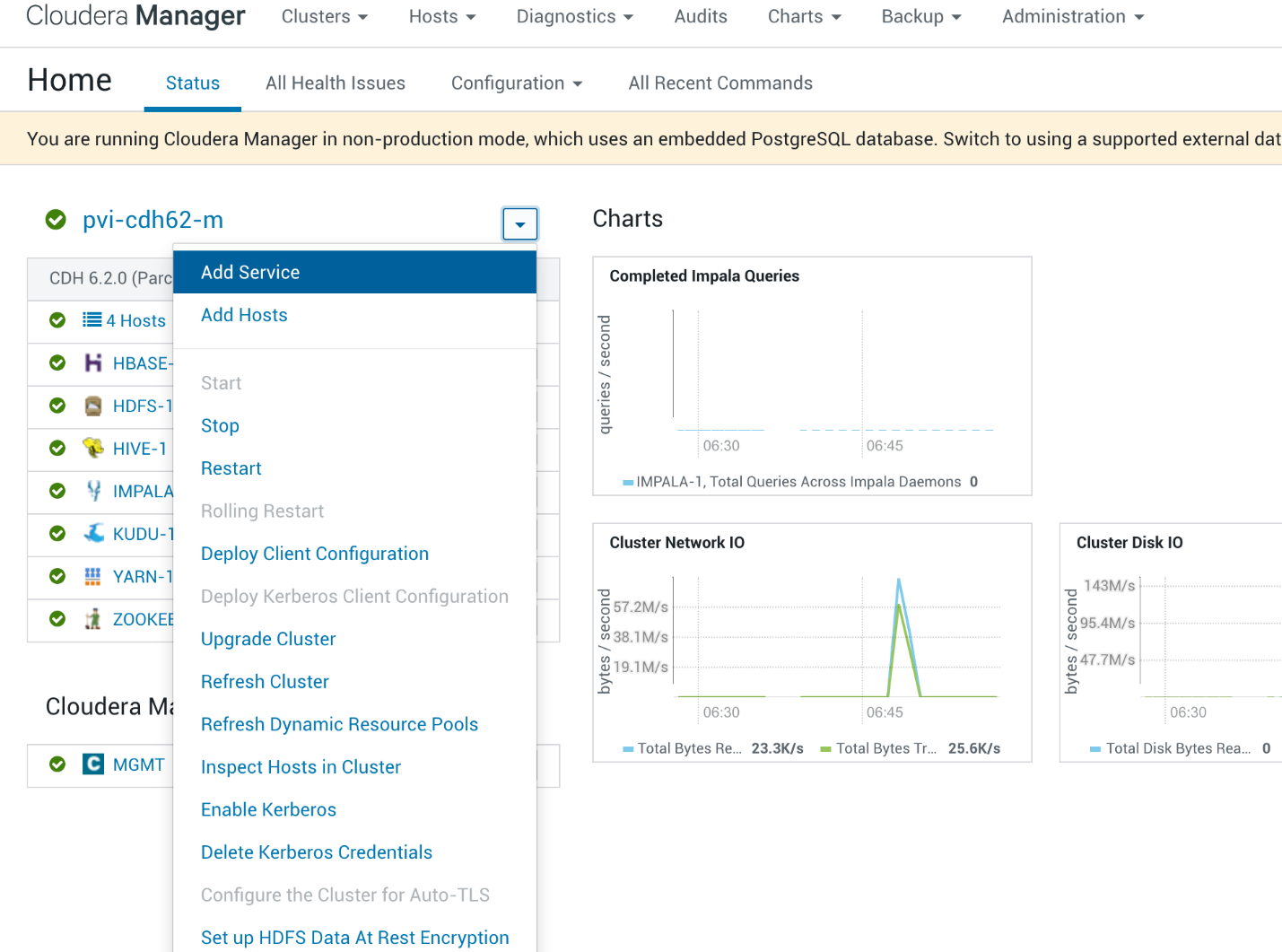
NiFi parcel configuration obtained <http://archive.cloudera.com/CFM/parcels/1.0.0.0/> and added to the parcel repository list. click on the “+” icon:



By clicking “Check for New Parcels” button, we should see a CFM parcel, then we manage to click “Download”. After the download completes, it will distribute and activate the parcel.

1. Install Services

We need to add NiFi service, so from CM main page, and “Add Service”:

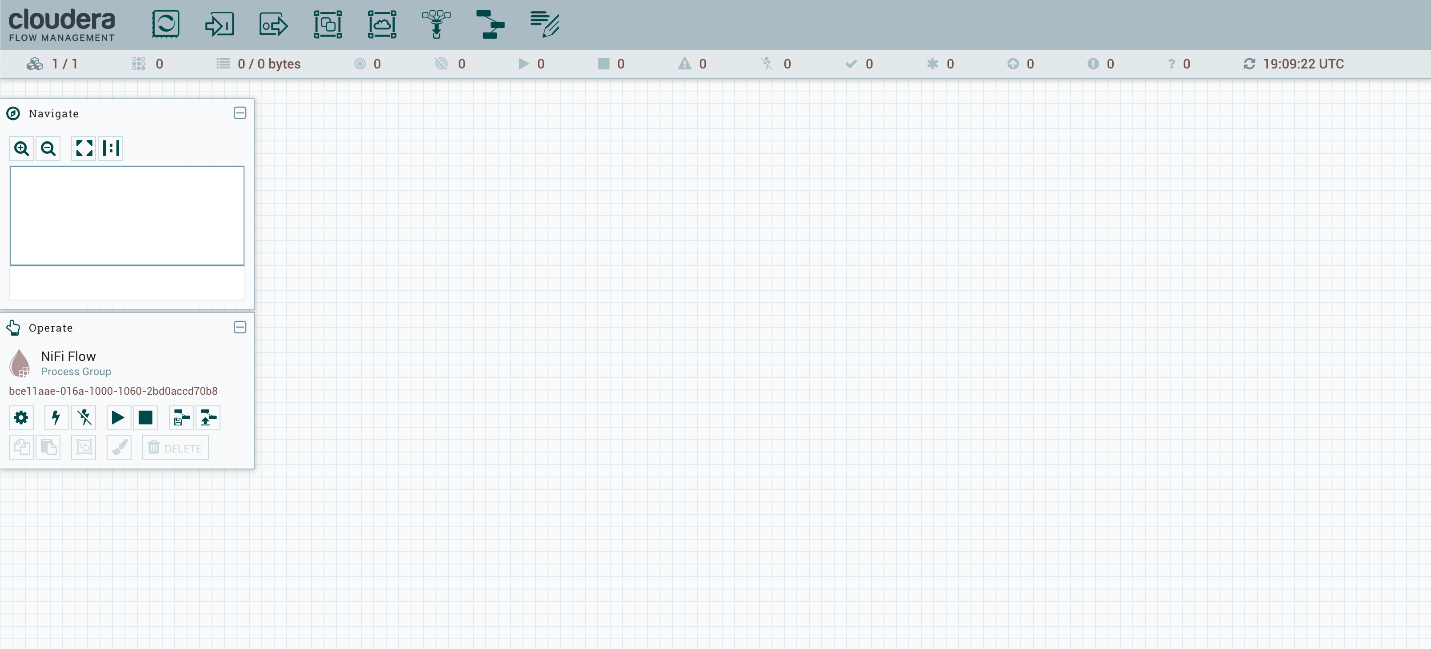


Adding NiFi service

Then we select the Nifi service, and specify the host onto which we want to install Nifi:

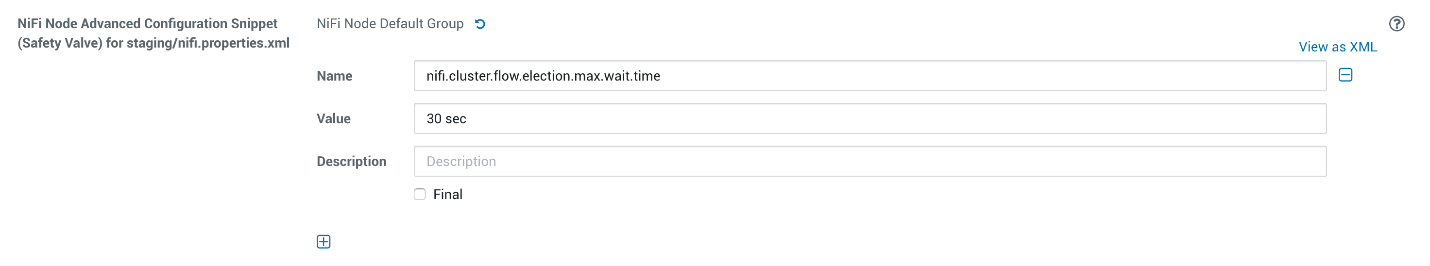
1. Profit

If we want to see main screen we had to go to Nifi URL:



NiFi main screen

The property nifi.cluster.flow.election.max.wait.time had been setting to 5 minutes by default, which can be confusing by giving rise to a log message like the cluster is still in the process of voting. to modify this property we can configure the nifi.properties.xml in CM, as follows:



Restart your Nifi instance via CM.