Final Project Report

Data Analytics Cloud - Design and Implementation

MAY 5, 2017

TEAM 2

Rushikesh Ghatpande (rsghatpa) - Lead
Karan Chaudhri (krchaud) - Vice-lead
Nithya Kumar (nkumar8)
Payal Arun Chheda (pchheda)
Ran Tan (rtan2)

COMPUTER SCIENCE DEPARTMENT
NC STATE UNIVERSITY

Abstract

As the world becomes increasingly digitized, data volumes are growing exponentially. Extracting information from data has the potential to revolutionize every sector of business. The focus on data analytics is therefore increasing day-by-day. Thus, there arises a need for a flexible data-analytics platform. To address this problem, we harness the power of cloud and virtualization in our project. Our project is focussed on developing a data analytics cloud platform that runs an analytics engine to perform re-structuring and analysis of vague, unstructured data. This report details our design and implementation of this platform. Our platform can be used as an Integrated Development Environment (IDE) to develop and test new algorithms for advanced analytical studies. The main functions of the analytics engine are gathering, filtering, annotating, restructuring and visualizing of data to the end-user. High-performance Intel blade server deployed in the Oscar laboratory of North Carolina State University's Centennial campus has been used as the base machine for cloud stack development. The cloud platform is built on Red Hat Enterprise Linux 7.3 (RHEL 7.3) with OpenStack Platform 10. The SaaS stack is integrated with Apache open source projects and in-house Django applications, each performing its own set of motivated tasks. Our validation results show that our cloud platform successfully gathers (by web crawling), filters, annotates, restructures the data and finally, displays the structured data to the end-user via a user-friendly dashboard thus facilitating him to make informed decisions.

TABLE OF CONTENTS

List of Tables

Table **Page**

LIST OF FIGURES

Chapter 1

Introduction

As data volumes grow exponentially, the demand for flexible data analysis platforms is also growing increasingly. The state-of-the-art solutions are not flexible or portable enough to satisfy the requirements of data scientists. Data Analytics has use-cases spread across myriad of industry domains viz., Health Care, Insurance Services, Financial Services, Character Recognition etc. To deal with this ever-growing need, we designed and implemented a flexible and customizable analytics cloud platform that offers both Platform as a Service (PaaS) and Software as a Service (SaaS) to an end-user. In the rest of this section, we highlight the high level aspects of the project, viz., the problem statement (Section 1.1), motivation (Section 1.2), the issues faced with regard to designing our platform (Section 1.3), and the environment upon which our platform is built (Section 1.4). The remainder of the document explains the low-level details of the project.

1.1 Problem Statement

The Institute for Next Generation IT Systems (ITng) is focused on the use of augmented intelligence, web development, and workflow design techniques to propel business forward. One of its key focus areas is on the intersection of moving ‚Äúbig data‚Äù infrastructure into cloud, and the development of novel algorithms to solve specific problems. Functionally, this requires a cognitive platform that transforms information to useful knowledge while optimizing the computing infrastructure.[13] Thus, our project aims to set-up a flexible and customizable base cloud platform with data restructuring and analytics capabilities. Figure 1.1 portrays the basic analytics process within the cloud analytics engine.

The project focuses on developing a **Data Analytics Cloud infrastructure** that will offer **Analytics as a Service**. The ultimate objective of our project is to deliver an analytics cloud

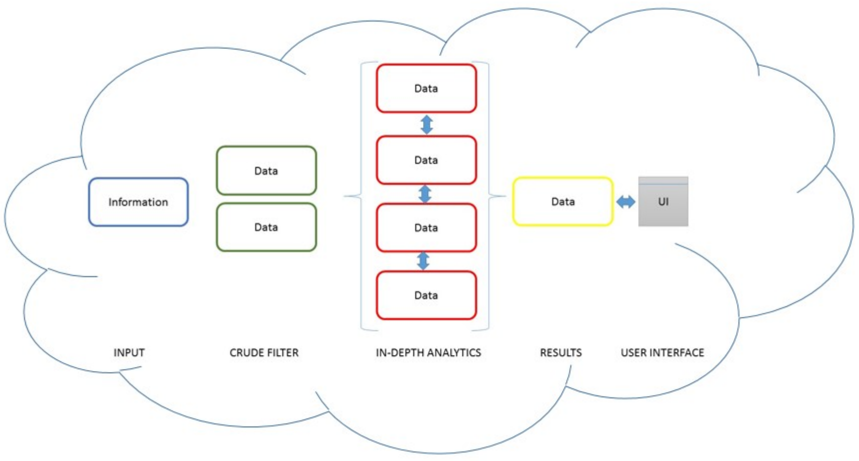


Figure 1.1: Big Data Analytics Cloud infrastructure.

platform built using OpenStack on Red Hat Enterprise Linux 7.3 (RHEL 7.3) and data analytics engine. The engine consists of two images - one is an analytics image and the other is a database image. The analytics image that does the entire data processing and analysis is built with Apache Nutch for web crawling, Solr (wrapped around Lucene) for indexing, UIMA for annotating the data, Mortar for filtering the data and Portrait for visualizing the data. The database image is built using Cassandra (NoSQL database) for storing unstructured data and PostgreSQL (relational database) for storing structured data.

1.2 Motivation

Our project is motivated to build a cloud analytics platform with the following properties:

1. **Flexible, customizable and easy to use**
   Present data-analysis platforms have some limitations. Some are built to cater to specific data domains while others might be too generic. Our analytics cloud platform is aimed at creating a master base platform that is equipped with the right set of minimalistic but yet comprehensive data analysis tools. Thus, this platform will be easy to use and also give the end user the power of customization. The user will be able to build upon our platform by installing additional tools as desired.

2. **Portable**
   Given our base analytics image, the end-user will be able to install additional tools as desired. The user will have the capability to download this image in a compatible format portable to many other platforms. The user will also have the capability to upload modified images.

3. **Capable of restructuring data**
   The cloud analytics platform will have the capability to turn large quantities of vague unstructured data to explicit structured data thereby enabling the end-user to make informed decisions.

1.3 Issues

Given our problem statement, we have identified the following set of issues that must be addressed by our cloud platform[15]:

1. **Security**
   Security is a prime concern for the cloud users due to the loss of control on data and applications. Moving data to the cloud can potentially compromise data security efforts and thus increase risk. Confidentiality, integrity, authentication and access control must be provided by the cloud service provider. In our project, we address some of these basic security concerns in Section 4.
2. **Ease of use**
   Usability of any system is directly proportional to its ease of use. Clean and visually appealing Graphical User Interfaces (GUI) are key to a system's ease of use. We address this concern by providing cloud management dashboard and data visualization interface in our project. Further details are explained in Section 4.
3. **Portability**
   To exploit the Pay-as-you-go cost model of cloud, any end-user would expect a cloud platform to support easy movement of images. This brings in the need for the cloud platforms to have compatible image formats. Section 4 details how this concern has been dealt with in our project.
4. **Structuring of unstructured data**
   In past decades, most data was in structured formats. Today, storage and data requirements are expanding due the boom in the quantities of unstructured data. Our project tackles this problem of restructuring the unstructured data into usable format as discussed in Section 4.

1.4 Environment

High-performance Intel blade server from Oscar laboratory owned by Institute for Next Generation IT Systems (ITng) at NCSU has been used as the base physical machine for the cloud platform. Machine can be accessed via ssh using appropriate credentials. Further environment details, including both software and hardware are further explained in Section 3.

Chapter 2

Requirements

This section elaborates the various functional and non-functional requirements that are identified as a part of our project[11]. In Section 5, each requirement is validated with a test case and we detail the one-to-one mapping between tasks done, test case validated and requirements satisfied.

2.1 Functional Requirements

The analytics cloud platform must provide the below mentioned functional requirements:

1. **Built on RHEL 7.3**
   The base operating system for the physical machine must be Red Hat Enterprise Linux 7.3.

2. **Cloud built using OpenStack Platform 10**
   OpenStack Platform 10 must be installed on top of the physical server using PackStack. The platform must use OpenStack to build and deploy VMs.

3. **Database VM**
   The cloud platform must have database image with capabilities to store data.

4. **Analytics VM**
   The cloud platform must have analytics image to provide data analytics capabilities.

5. **GPFS distributed store**
   All images in the analytics cloud platform must be stored to the provided IBM‚Äôs General Parallel File System (GPFS) distributed store

6. **Database VM for storage of both structured and unstructured data**
   The database image must have installed on it - Apache Cassandra 2.2.8 (NoSQL database) for storing unstructured data and PostgreSQL 9.2.1 (relational database) for storing structured data.

7. **Analytics VM for web crawling**
   The analytics image must have installed on it - Apache Nutch 2.2.1 for crawling the web based on input seed Uniform Resource Locators (URLs).

8. **Analytics VM for data indexing**
   The analytics image must have installed on it - Apache Solr 6.3.0 for indexing the unstructured data crawled by Nutch.

9. **Analytics VM for data filtering**
   The analytics image must have installed on it - Mortar (in-house Django application) for filtering the data indexed by Solr.

10. **Data import and export**
    The database VM must allow import of structured data in a Comma Separated Values (CSV) format from the analytics VM. Similarly, the analytics VM must allow export of structured data in a CSV format from Solr to PostgreSQL.

11. **Analytics VM for data visualization**
    The analytics image must have installed on it - Portrait (in-house Django application) for end-user visualization of data.

12. **Authentication and Authorization**
    The analytics cloud platform must prevent unauthorized access to VMs.

2.2 Non-Functional Requirements

The analytics cloud platform must provide the below mentioned non-functional requirements:

1. **Analytics cloud deployed on ITng physical server**
   The analytics cloud platform must be deployed on the Intel Blade Server (detailed specifications in Section 3.1) located in the Oscar Laboratory owned by NCSU's ITng.

2. **Stability and on-demand availability**
   The analytics cloud platform must be stable and available on-demand to an end-user.

3. **Extensibility**
   The analytics cloud platform must allow end-user the ability to customize the analytics image by adding new analytics package catering to his needs.

4. **Testability**
   The analytics cloud platform must be easy to test and validate.

5. **Installation of dependencies**
   All software package dependencies like Java 1.7, Apache Ant, Python libraries etc must be installed.

Chapter 3

System Environment

This section elaborates the specifications of our environment introduced in Section 1.4. Below is the list of hardware and software specifications for our project:

3.1 Hardware Specifications

High-performance Intel blade server from Oscar laboratory has been used as the base machine for cloud stack development. The 8 cores and 16 CPUs on it lead to a greater degree of parallelism.

The CPU architecture is Intel x86-64 and has caches of varying sizes. Each core has a 32 KB L1 instruction and data cache, and 256 KB L2 cache. L3 cache of size 12 MB is shared among all CPUs. The total physical memory of blade server is 24 GB.

3.2 Software Specifications

Below are the software specifications for our project:

- Redhat Enterprise Linux (RHEL), version 7.3 OS on the blade server
- Red Hat OpenStack Platform, version 10 installed on RHEL
- Kernel-based Virtual Machine (KVM) with Quick Emulator (QEMU) as the type-2 hypervisor IBM GPFS for image storage
- Ubuntu-16.04 LTS (Xenial Xerus) qcow2 image[14] as base image
- Two customized images - one for DBs and another for analytics

- The analytics image is mainly customized with: Nutch, Solr, Mortar, UIMA and Portrait. The database image will be configured with: Cassandra and PostgreSQL

Chapter 4

Design, Implementation and Results

In this section, we go into low-level details about our design decisions and their pros and cons. We further explain the implementation details of our design and the end results. Finally, we discuss the limitations of our approach.

4.1 Design

This section describes our design decisions and their tradeoffs. Figure 4.1 shows the high level architecture of our analytics cloud platform built using the hardware and software specifications as described in Section 3.

At the bottom of the cloud platform stack is our Intel Blade Server owned by Oscar laboratory of ITng group at NCSU as described in Section 3.1. This machines serves as our host machine and comes with RHEL 7.3 installed on it. On this machine, we have KVM with QEMU as the type-2 hypervisor. The host machine and guest VMs hosted on this machine can be accessed by ssh. The guest VMs are deployed using OpenStack Packstack. 144 TB of IBM‚Äôs General Parallel File System (GPFS) is available as high-performance distributed storage.

We now go into further details and list the the OpenStack modules that are a crucial part of our design:

1. **Nova**
   Nova provides compute capabilities and helps with creation, deployment, management, deletion etc of instances.

2. **Neutron**
   Neutron provides networking capabilities for communication among deployed instances (internal) and to the outside world (external).

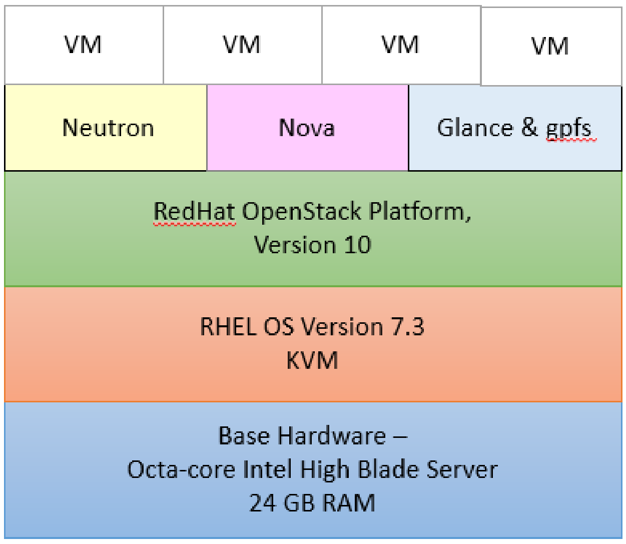

VM
VM
VM
VM
Neutron
Nova
Glance & gpfs
RedHat OpenStack Platform, Version 10
RHEL OS Version 7.3 KVM
Base Hardware – Octa-core Intel High Blade Server 24 GB RAM

Figure 4.1: High Level Architecture

3. **Glance**
   Glance provides imaging capabilities by allowing base image installed to be used as a template while deploying new virtual machine instances.

4. **Cinder**
   Cinder provides block storage management capabilities.

5. **Keystone**
   Keystone provides the authentication capabilities by providing client authentication API and service discovery.

6. **Horizon**
   Horizon provides provides a web based user interface serving as a dashboard to OpenStack services including Nova, Swift, Keystone etc. Although not essential, this component adds value to our project by facilitating our goal of ease-of-use.

The base image for providing Platform as a Service is Ubuntu-16.04 LTS (Xenial Xerus) qcow2 image as it takes full advantage of computers based on the AMD64 architecture. AMD64 is recommended for general computing use in Cloud. The base Ubuntu-16.04 LTS image is used to create two customized images - one for DBs (database image) and another for analytics (analytics image).

The further sub-sections 4.1.1, 4.1.2, 4.1.3 detail the network topology, software stack design and the sample workflow of our project respectively. Section 4.1.4 addresses the issues that we previously identified in Section 1.3.

4.1.1 Network Topology

Let us now consider the design of our network topology. Our design must ensure that the analytics and database VM are able to communicate with each other and with the outside world. Figure 4.2 illustrates the network topology as described below. We designed our topology to consist of two networks:

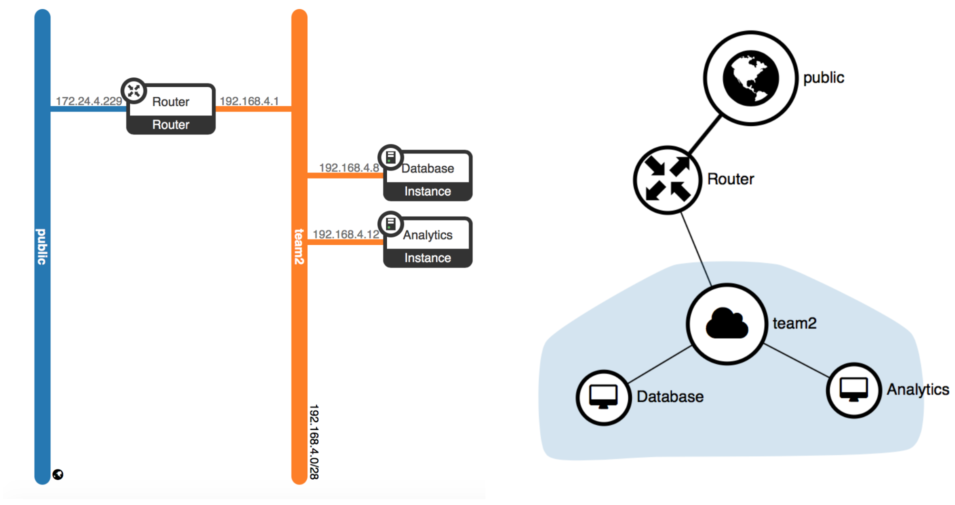


Figure 4.2: Network Topology of Analytics and Database Image.

1. **Private network**
   The two instances i.e. Analytics and Database image instances are connected to each other in this private network. They fall in the same subnet and hence shall be able to communicate directly to one another.

2. **External/Public network**
   In order for our instances to have connectivity to and from the external world, we have a router such that one of the interfaces is connected to the public network for external connectivity and the other interface is connected to private network.

We place our instances behind NAT to prevent any unauthorized access to our instances from the outside network. We assign floating ip only to our Analytics VM so that it can be accessed on public network and database VM is only accessible through private network via database client on the analytics VM.

4.1.2 Software Stack Design

The analytics image is mainly configured with: Nutch, Solr, Mortar, UIMA and Portrait [3–6]. The database image is mainly configured with: Cassandra [2] and PostgreSQL [10]. Thus, the software

Projects/Applications	Purpose/Description
Apache Nutch 2.2.1 [6]	Nutch is an open source web crawler. It uses Cassandra as a storage backend, as well as Solr as its indexer
Apache Ant 1.9.6 [1]	Ant is a Java build tool that allows easy configuration of large Java projects and their dependencies with a simple xml file. It is used to build both Nutch and Solr from source
Apache Cassandra 2.2.8 [2]	Cassandra is an open source database system that supports high availability and easy scalability. We chose Cassandra over other Nutch backends, such as Apache HBase or straight SQL, to avoid installing the entire Hadoop system
Apache UIMA 2.9.0 and Eclipse UIMA Workbench [4]	UIMA (Unstructured Information Management Applications) is an open source project for analyzing data and pulling out relevant entities as a basic Natural Language Processing toolkit. This is a user side installation
Apache Solr 6.3.0 [3]	Apache Solr is an open source search platform, wrapped around Lucene. Nutch is able to use Solr to index web page crawl results, and users can query Solr to see results without digging through Cassandra database
Mortar	Mortar is an in-house Django [5] web application built to create and manage PESTLE trees and mind maps. Mortar performs the first round of filtering on our dataset by allowing users to create custom queries based on their trees
Portrait	Portrait is an in-house Django [5] web application that leverages Django SQL Explorer to analyze and combine data into useful insights and visualizations
PostGreSQL 9.2.1 [10]	PostgreSQL is an open source object-relational database system

Table 4.1: Software packages.

stack is comprised of several Apache open source projects[12] and a few in house applications as shown in Table 4.1.

4.1.3 Workflow

The design workflow of the data analytics process followed by an end-user on our cloud platform can be explained with the following steps:

1. **Login & Input**
    a) User logs into the host machine via ssh
    b) User provides seed URLs as input for crawling the Analytics instance
2. **Data Ingestion**
    a) User logs into the analytics VM via ssh

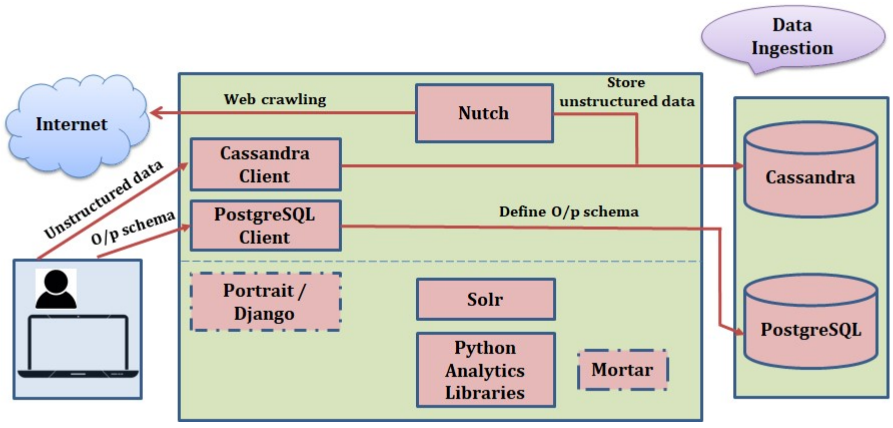


Figure 4.3: Platform Data-flow - Data Ingestion.

b) Using the Command Line Interface (CLI), user injects seed URLs into Cassandra

c) Using CLI, user prepares a batch file of the inputted seed URLS

d) Using CLI, user issues command to Nutch to crawl the web with batch file as input.

e) Using CLI, user sends the collected unstructured data to Cassandra for storage

Figure 4.3 above illustrates the flow of data during the data ingestion process.

3. **Data Culling & Restructuring**

a) Using CLI, user issues command to Solr to index the unstructured data collected by Nutch

b) User logs into Mortar web interface with appropriate credentials and applies filters to the indexed data in Solr

c) User develops annotations and applies these annotations to the data in Solr using UIMA

d) Using CLI, user exports the structured data in a CSV format file from Solr

e) Using CLI, user imports the above exported CSV format file to PostgreSQL

f) The user imports the structured data from Solr to the PostgreSQL

Figure 4.4 below illustrates the flow of data during the data culling and restructuring process.

4. **Visualization**

a) Using Portrait, user visualizes and queries the structured data stored in PostgreSQL

b) User makes informed decisions using the visualized data

Figure 4.5 below illustrates the data visualization process.

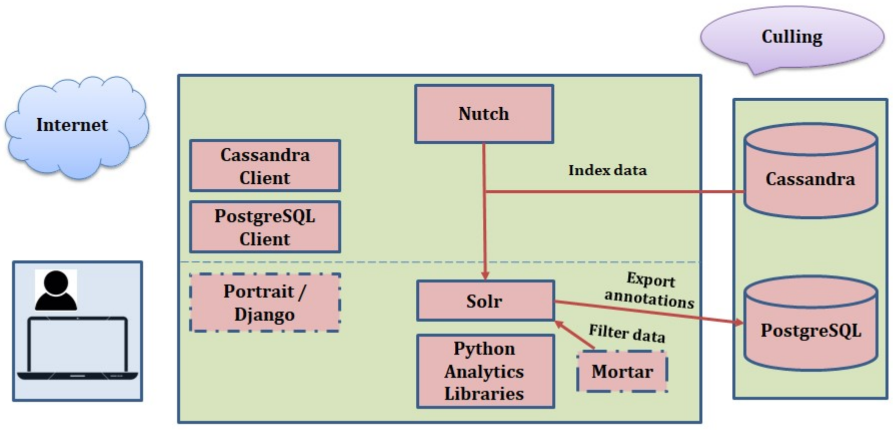


Figure 4.4: Platform Data-flow - Data Culling.

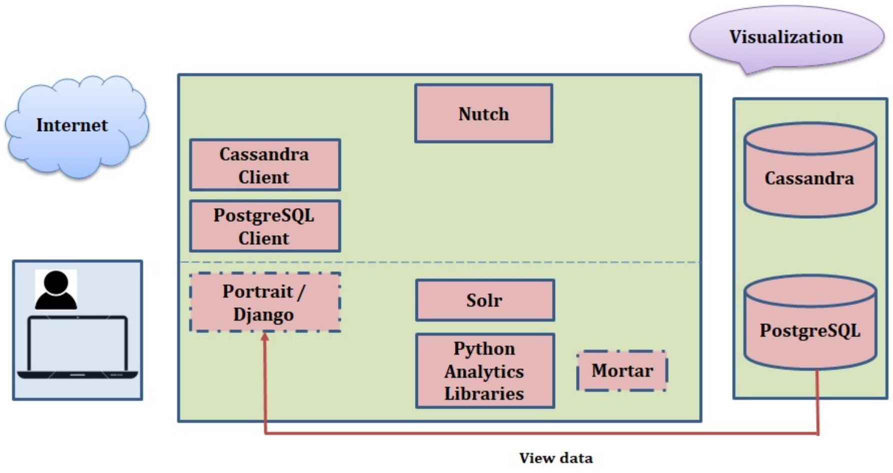


Figure 4.5: Platform Data-flow - Data Visualization.

4.1.4 Issues Addressed

In Section 1.3 we discussed the major issues concerning the design and implementation of the cloud analytics platform. In this section, we discuss measures taken in our design to address these issues.

1. **Security**
   Security is one of the major concerns in building a cloud platform. Although not extensive, we use the following measures to ensure basic security of our analytics cloud platform:

   a) **Keystone authentication**
      OpenStack Keystone provides identity service as the first level of user authentication by the mechanism of username and password authentication.

b) **Database access control**
In our design, we have two levels of access control for the database VM instance. One is, we assign floating IP only to the analytics VM instance and restrict direct user access to the database VM instance. The only way, an user can access the content of the database is through the database client installed on the analytics VM instance. A client can communicate with the database VM on only 2 ports : Cassandra Thrift Client port 9160 and the PostgreSQL service can be accessed on port 5432. Second is, database access is restricted using username password authentication mechanism.

c) **Firewalls**
We modify the firewall rules of our network to ensure there is no unauthorized access to database VM instance.

d) **SSH Configuration**
On the analytics VM, changes are made to ssh configuration files to ensure that clients can log into the machine. The associated private IP address and floating IP address is added as a ListenAddress and password based authentication has been set up through the /etc/ssh/sshd_config file. Client is authorized to setup Key-based authentication after he logs into his virtual machine.

2. **Ease of use**
Clean and visually appealing Graphical User Interfaces (GUI) are key to a system‚Äôs ease of use. Our design allows users to deploy and manage VMs using Horizon dashboard. Solr also has a web interface for indexing data. Mortar and Portrait provide GUIs to filter and visualize the data. CLI interaction is minimal as far as possible.

3. **Portability**
Since our images are in popular format QCOW2, there is a lot of support available in the market for this format of images. This addresses our issue of portability.

4. **Structuring of unstructured data**
This issue is addressed as explained in Section 4.1.3 subpoint 3. Data is indexed by Solr and structured using regex filters by Mortar and annotations by UIMA. Structure data is further stored in the PostgreSQL database in the database VM.

4.2 Implementation

This section describes the implementation steps for the design as explained in Section 4.1. We explain the exact configuration steps for certain processes in the Appendix.

1. **Installation of RedHat OpenStack Platform 10**
To implement the big-data analytics cloud, as our first step, we installed RedHat OpenStack

platform 10 using Packstack [9]. Packstack helps in installing all the mentioned components in a single node rather than distributed installation on several nodes as in production environment. Section A.1 of the Appendix details steps needed to install OpenStack using PackStack. It also outlines the errors faced in the process and the steps take to resolve these errors.

2. **Base image setup**
   Next, we set up our base image - Ubuntu 16.04.02 LTS (Xenial Xerus) qcow2. This image provides a robust platform for installation of our software packages. Further, we modified Glance to store images to /gpfs directory. For further details on the exact steps involved in installation, refer Appendix Section A.3.

3. **Customized image setup with software stack installation**
   The next step was to create two customized images - one for database VM and another for analytics VM through installation of specified software packages (work in progress). As mentioned in the software specifications, the analytics image is mainly customized with: Nutch, Solr, Mortar and Portrait. The database image is configured with: Cassandra and PostgreSQL. For further details on the exact steps involved in installation, refer Appendix Section A.4.

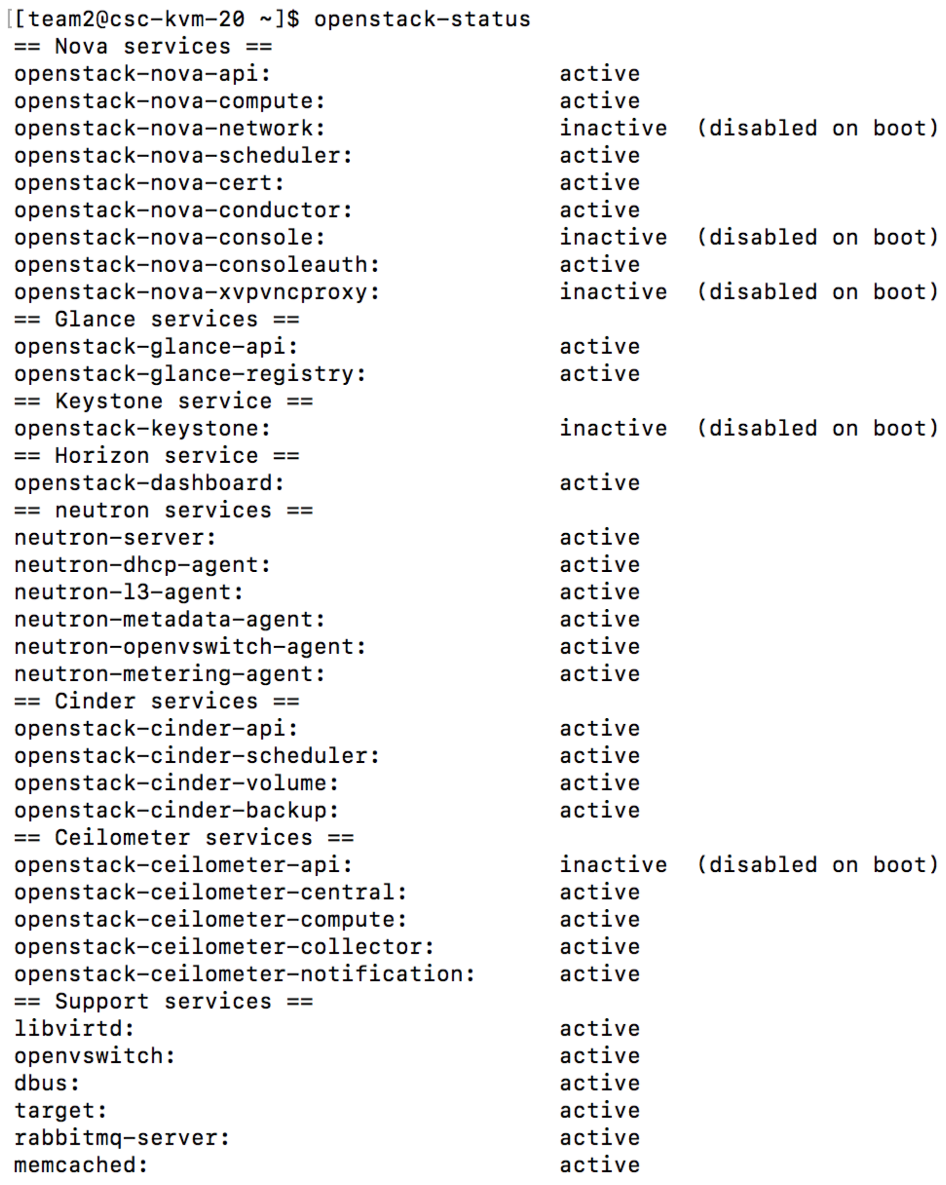
4. **Network , firewalls and ssh configuration**
   Once we set up our customized images, we implemented the network topology as designed in Section 4.1.1. We set up firewalls and ssh configuration as explained in Section 4.1.4 where we address the issue of security through the measures of firewalls and ssh.

5. **Validation of workflow**
   We validated the workflow and verified that all the test cases corresponding to our requirements are satisfied. Refer Section 5.1 for detailed understanding of our testing and validation. The results of our validation workflow are elaborated in Section 4.3

4.3 Results

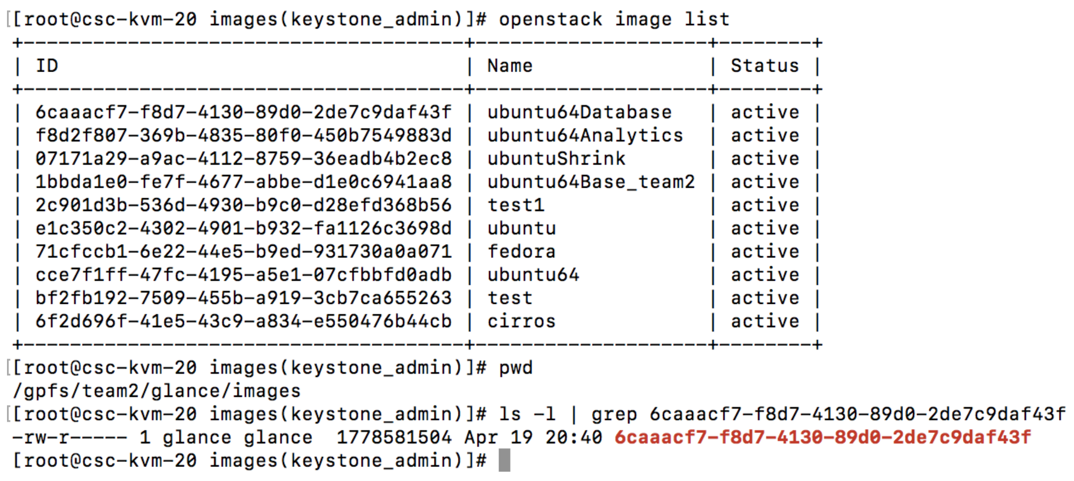
The design and implementation details of our analytics cloud platform were explained in Sections 4.1 and 4.2. The following are the actual results of each step involved in the process of providing the deliverables of our project.

1. **OpenStack installation on physical server using PackStack**
   Redhat OpenStack Platform 10 is successfully deployed on top of RHEL OS 7.3. KVM is used as the hypervisor for virtualization. Nova, Neutron, Glance, Cinder, Keystone and Horizon are running on Blade Server. Figure 4.6 displays the list of running services on OpenStack.



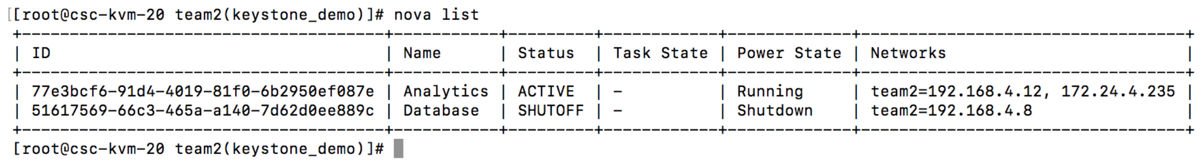
```
[[team2@csc-kvm-20 ~]$ openstack-status
 == Nova services ==
 openstack-nova-api:                        active
 openstack-nova-compute:                    active
 openstack-nova-network:                    inactive  (disabled on boot)
 openstack-nova-scheduler:                  active
 openstack-nova-cert:                       active
 openstack-nova-conductor:                  active
 openstack-nova-console:                    inactive  (disabled on boot)
 openstack-nova-consoleauth:                active
 openstack-nova-xvpvncproxy:                inactive  (disabled on boot)
 == Glance services ==
 openstack-glance-api:                      active
 openstack-glance-registry:                 active
 == Keystone service ==
 openstack-keystone:                        inactive  (disabled on boot)
 == Horizon service ==
 openstack-dashboard:                       active
 == neutron services ==
 neutron-server:                            active
 neutron-dhcp-agent:                        active
 neutron-l3-agent:                          active
 neutron-metadata-agent:                    active
 neutron-openvswitch-agent:                 active
 neutron-metering-agent:                    active
 == Cinder services ==
 openstack-cinder-api:                      active
 openstack-cinder-scheduler:                active
 openstack-cinder-volume:                   active
 openstack-cinder-backup:                   active
 == Ceilometer services ==
 openstack-ceilometer-api:                  inactive  (disabled on boot)
 openstack-ceilometer-central:              active
 openstack-ceilometer-compute:              active
 openstack-ceilometer-collector:            active
 openstack-ceilometer-notification:         active
 == Support services ==
 libvirtd:                                  active
 openvswitch:                               active
 dbus:                                      active
 target:                                    active
 rabbitmq-server:                           active
 memcached:                                 active
```

Figure 4.6: OpenStack services.



```
[root@csc-kvm-20 images(keystone_admin)]# openstack image list
+--------------------------------------+--------------------+--------+
| ID                                   | Name               | Status |
+--------------------------------------+--------------------+--------+
| 6caaacf7-f8d7-4130-89d0-2de7c9daf43f | ubuntu64Database   | active |
| f8d2f807-369b-4835-80f0-450b7549883d | ubuntu64Analytics  | active |
| 07171a29-a9ac-4112-8759-36eadb4b2ec8 | ubuntuShrink       | active |
| 1bbda1e0-fe7f-4677-abbe-d1e0c6941aa8 | ubuntu64Base_team2 | active |
| 2c901d3b-536d-4930-b9c0-d28efd368b56 | test1              | active |
| e1c350c2-4302-4901-b932-fa1126c3698d | ubuntu             | active |
| 71cfccb1-6e22-44e5-b9ed-931730a0a071 | fedora             | active |
| cce7f1ff-47fc-4195-a5e1-07cfbbfd0adb | ubuntu64           | active |
| bf2fb192-7509-455b-a919-3cb7ca655263 | test               | active |
| 6f2d696f-41e5-43c9-a834-e550476b44cb | cirros             | active |
+--------------------------------------+--------------------+--------+
[root@csc-kvm-20 images(keystone_admin)]# pwd
/gpfs/team2/glance/images
[root@csc-kvm-20 images(keystone_admin)]# ls -l | grep 6caaacf7-f8d7-4130-89d0-2de7c9daf43f
-rw-r----- 1 glance glance  1778581504 Apr 19 20:40 6caaacf7-f8d7-4130-89d0-2de7c9daf43f
[root@csc-kvm-20 images(keystone_admin)]#
```

Figure 4.7: Images are stored in /gpfs directory.



```
[root@csc-kvm-20 team2(keystone_demo)]# nova list
+--------------------------------------+-----------+---------+------------+-------------+----------------------------------+
| ID                                   | Name      | Status  | Task State | Power State | Networks                         |
+--------------------------------------+-----------+---------+------------+-------------+----------------------------------+
| 77e3bcf6-91d4-4019-81f0-6b2950ef087e | Analytics | ACTIVE  | -          | Running     | team2=192.168.4.12, 172.24.4.235 |
| 51617569-66c3-465a-a140-7d62d0ee889c | Database  | SHUTOFF | -          | Shutdown    | team2=192.168.4.8                |
+--------------------------------------+-----------+---------+------------+-------------+----------------------------------+
[root@csc-kvm-20 team2(keystone_demo)]#
```

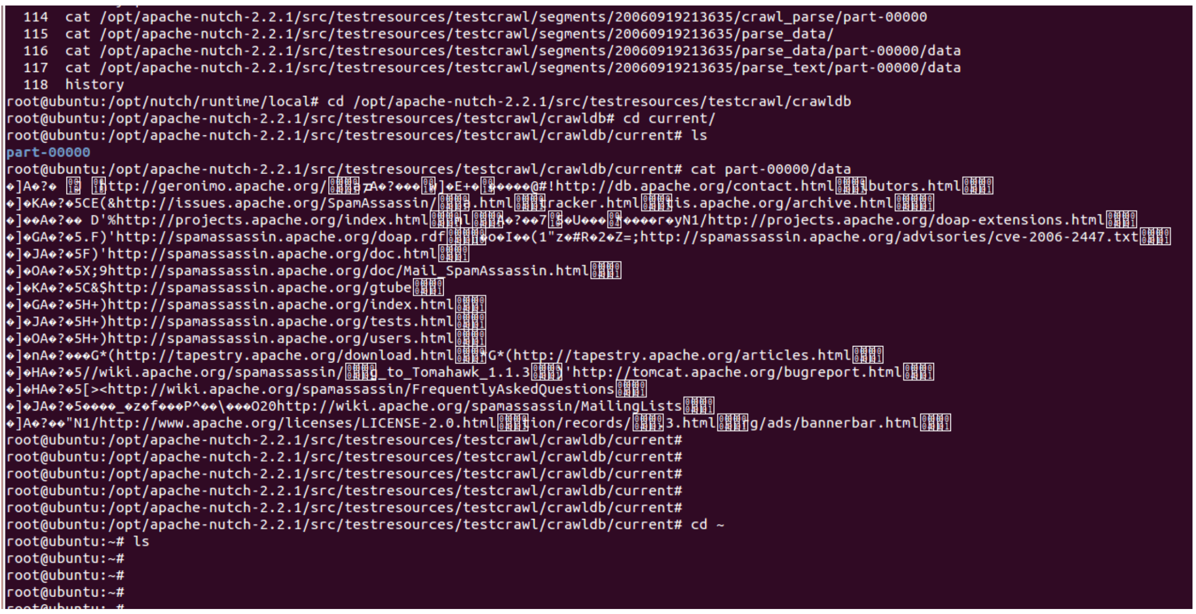
Figure 4.8: Database VM and Analytics VM were successfully created for client team2.

2. **Ubuntu 16.04.02 LTS (Xenial Xerus) qcow2 Base image setup**
   Base .iso image is created using qemu and virt-manager. Steps for base image setup are mentioned in Appendix A.2.

3. **Networking configuration setup**
   As described in Section 4.1.1, our network topology consists of two networks, named 'public' and 'team2' for providing external as well as private network connectivity respectively. It consists of a Router whose one of the interface is connected to the public network for external connectivity and the other interface is connected to private network named 'team2'. Figure 4.2 shows the networking configuration of our platform.

4. **Modified Glance to store images to /gpfs directory**
   On adding the customized images to Glance, they are stored in the /gpfs/glance directory as indicated. Images are immediately available for new users to use to boot up their VMs. Figure 4.7 and 4.8 describe this step.

5. **Data ingestion using Nutch**
   The user provides seed URL which are placed in seed.txt. The pages crawled are indexed and kept in crawldb, a nutch module. Data is successfully ingested into Cassandra server from a client machine. Several formats, including csv files can be used to ingest data into



```
  114  cat /opt/apache-nutch-2.2.1/src/testresources/testcrawl/segments/20060919213635/crawl_parse/part-00000
  115  cat /opt/apache-nutch-2.2.1/src/testresources/testcrawl/segments/20060919213635/parse_data/
  116  cat /opt/apache-nutch-2.2.1/src/testresources/testcrawl/segments/20060919213635/parse_data/part-00000/data
  117  cat /opt/apache-nutch-2.2.1/src/testresources/testcrawl/segments/20060919213635/parse_text/part-00000/data
  118  history
root@ubuntu:/opt/nutch/runtime/local# cd /opt/apache-nutch-2.2.1/src/testresources/testcrawl/crawldb
root@ubuntu:/opt/apache-nutch-2.2.1/src/testresources/testcrawl/crawldb# cd current/
root@ubuntu:/opt/apache-nutch-2.2.1/src/testresources/testcrawl/crawldb/current# ls
part-00000
root@ubuntu:/opt/apache-nutch-2.2.1/src/testresources/testcrawl/crawldb/current# cat part-00000/data
�]A�?� ▯ ▯http://geronimo.apache.org/▯▯▯zA�?���▯]�E+�▯����@#!http://db.apache.org/contact.html▯▯▯butors.html▯▯▯
�]�KA�?�5CE(&http://issues.apache.org/SpamAssassin/▯▯▯.html▯▯▯racker.html▯▯▯is.apache.org/archive.html▯▯▯▯
�]��A�?�� D'%http://projects.apache.org/index.html▯▯▯l▯▯▯A�?��7▯�U���▯����r�yN1/http://projects.apache.org/doap-extensions.html▯▯▯
�]�GA�?�5.F)'http://spamassassin.apache.org/doap.rdf▯▯▯▯o�I��(1"z�#R�2�Z=;http://spamassassin.apache.org/advisories/cve-2006-2447.txt▯▯▯
�]�JA�?�5F)'http://spamassassin.apache.org/doc.html▯▯▯
�]�OA�?�5X;9http://spamassassin.apache.org/doc/Mail_SpamAssassin.html▯▯▯
�]�KA�?�5C&$http://spamassassin.apache.org/gtube▯▯▯
�]�GA�?�5H+)http://spamassassin.apache.org/index.html▯▯▯
�]�JA�?�5H+)http://spamassassin.apache.org/tests.html▯▯▯
�]�OA�?�5H+)http://spamassassin.apache.org/users.html▯▯▯
�]�nA�?���G*(http://tapestry.apache.org/download.html▯▯▯G*(http://tapestry.apache.org/articles.html▯▯▯
�]�HA�?�5//wiki.apache.org/spamassassin/▯▯▯g_to_Tomahawk_1.1.3▯▯▯'http://tomcat.apache.org/bugreport.html▯▯▯
�]�HA�?�5[><http://wiki.apache.org/spamassassin/FrequentlyAskedQuestions▯▯▯
�]�JA�?�5����_�z�f���P^��\���O20http://wiki.apache.org/spamassassin/MailingLists▯▯▯
�]A�?��"N1/http://www.apache.org/licenses/LICENSE-2.0.html▯▯▯ion/records/▯▯▯3.html▯▯▯g/ads/bannerbar.html▯▯▯
root@ubuntu:/opt/apache-nutch-2.2.1/src/testresources/testcrawl/crawldb/current#
root@ubuntu:/opt/apache-nutch-2.2.1/src/testresources/testcrawl/crawldb/current#
root@ubuntu:/opt/apache-nutch-2.2.1/src/testresources/testcrawl/crawldb/current#
root@ubuntu:/opt/apache-nutch-2.2.1/src/testresources/testcrawl/crawldb/current#
root@ubuntu:/opt/apache-nutch-2.2.1/src/testresources/testcrawl/crawldb/current#
root@ubuntu:/opt/apache-nutch-2.2.1/src/testresources/testcrawl/crawldb/current# cd ~
root@ubuntu:~# ls
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~#
```

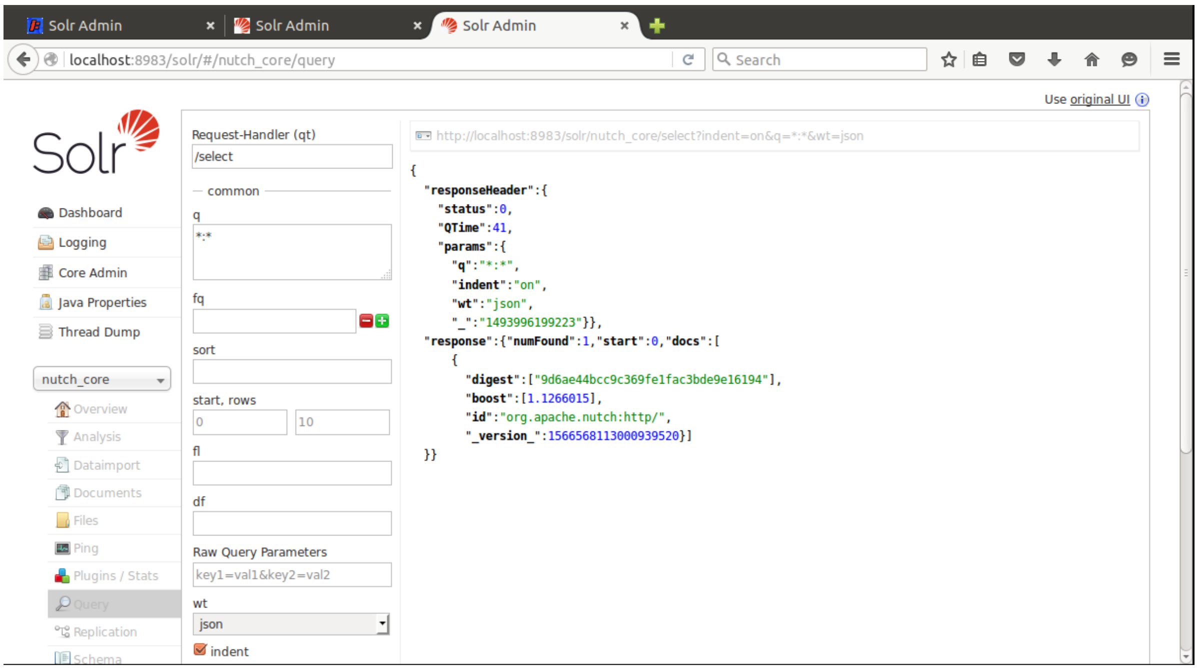
Figure 4.9: Web pages are successfully crawled using nutch.

Cassandra. On querying, the imported data was successfully retrieved. Data generated is further sent to Solr for indexing. Figure 4.9 identifies this web crawling step.

6. **Data indexing using Solr**
   Solr dashboard displays the indexed data uploaded from Nutch. Figure 4.10 and 4.12 displays the results of this step.

7. **Data filtering using Mortar**
   Mortar interface is used to query data using PESTLE tree. Filtered data can be viewed using the Mortar interface.

8. **Data visualization using Portrait**
   The structured data from PostGreSQL can be visualized using Portrait interface. This data can be used by the user to make informed decisions. Figure 4.11 shows the Portrait visualization of our project.

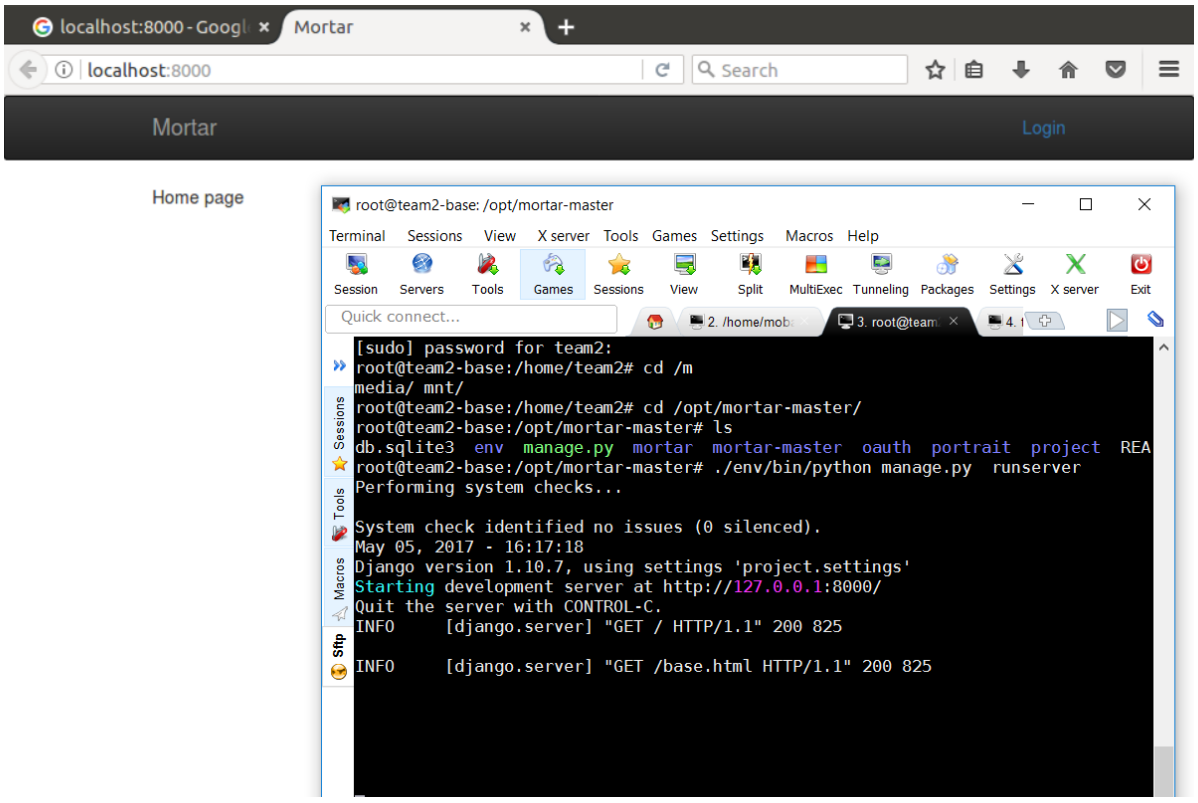


Request-Handler (qt): /select
q: *:*
start, rows: 0 10
Raw Query Parameters: key1=val1&key2=val2
wt: json
☑ indent

http://localhost:8983/solr/nutch_core/select?indent=on&q=*:*&wt=json

```
{
  "responseHeader":{
    "status":0,
    "QTime":41,
    "params":{
      "q":"*:*",
      "indent":"on",
      "wt":"json",
      "_":"1493996199223"}},
  "response":{"numFound":1,"start":0,"docs":[
      {
        "digest":["9d6ae44bcc9c369fe1fac3bde9e16194"],
        "boost":[1.1266015],
        "id":"org.apache.nutch:http/",
        "_version_":1566568113000939520}]
  }}
```

Figure 4.10: Solr Query returns indexed data.

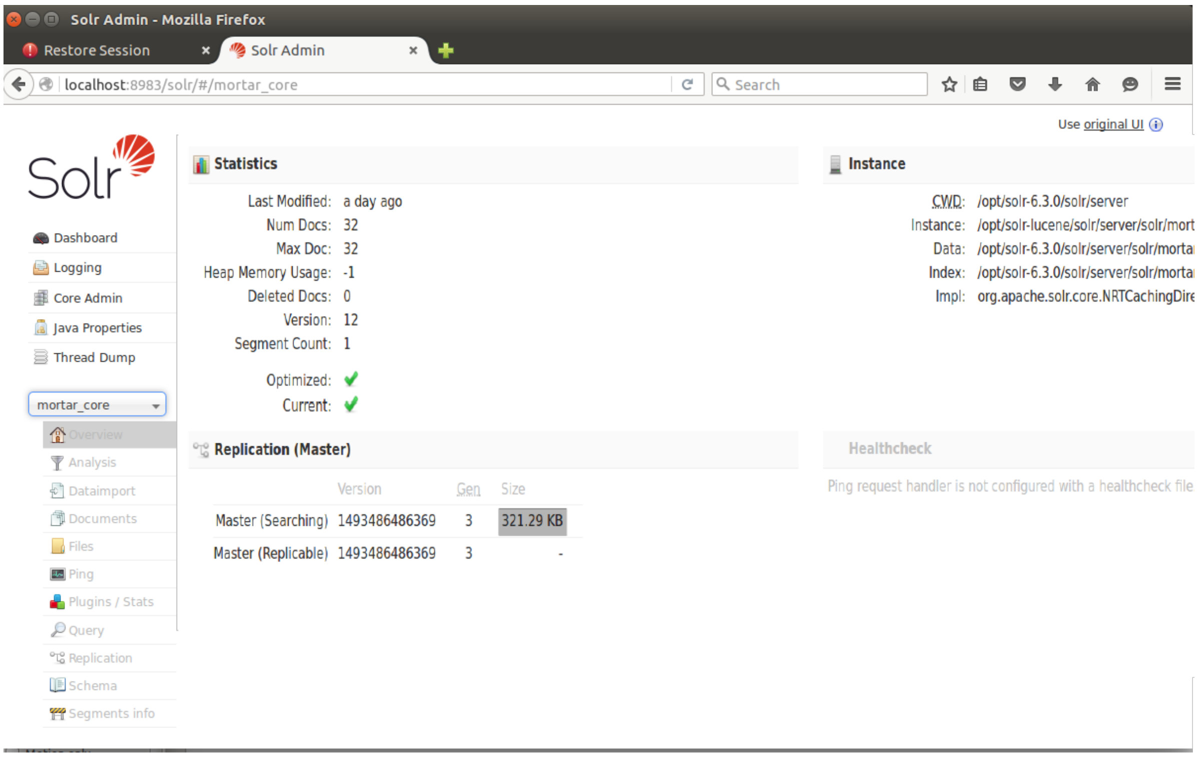


```
[sudo] password for team2:
root@team2-base:/home/team2# cd /m
media/ mnt/
root@team2-base:/home/team2# cd /opt/mortar-master/
root@team2-base:/opt/mortar-master# ls
db.sqlite3  env  manage.py  mortar  mortar-master  oauth  portrait  project  REA
root@team2-base:/opt/mortar-master# ./env/bin/python manage.py  runserver
Performing system checks...

System check identified no issues (0 silenced).
May 05, 2017 - 16:17:18
Django version 1.10.7, using settings 'project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
INFO      [django.server] "GET / HTTP/1.1" 200 825

INFO      [django.server] "GET /base.html HTTP/1.1" 200 825
```

Figure 4.11: Mortar service running.



Statistics

Last Modified:	a day ago
Num Docs:	32
Max Doc:	32
Heap Memory Usage:	-1
Deleted Docs:	0
Version:	12
Segment Count:	1
Optimized:	✓
Current:	✓

Instance

CWD:	/opt/solr-6.3.0/solr/server
Instance:	/opt/solr-lucene/solr/server/solr/mort
Data:	/opt/solr-6.3.0/solr/server/solr/morta
Index:	/opt/solr-6.3.0/solr/server/solr/morta
Impl:	org.apache.solr.core.NRTCachingDire

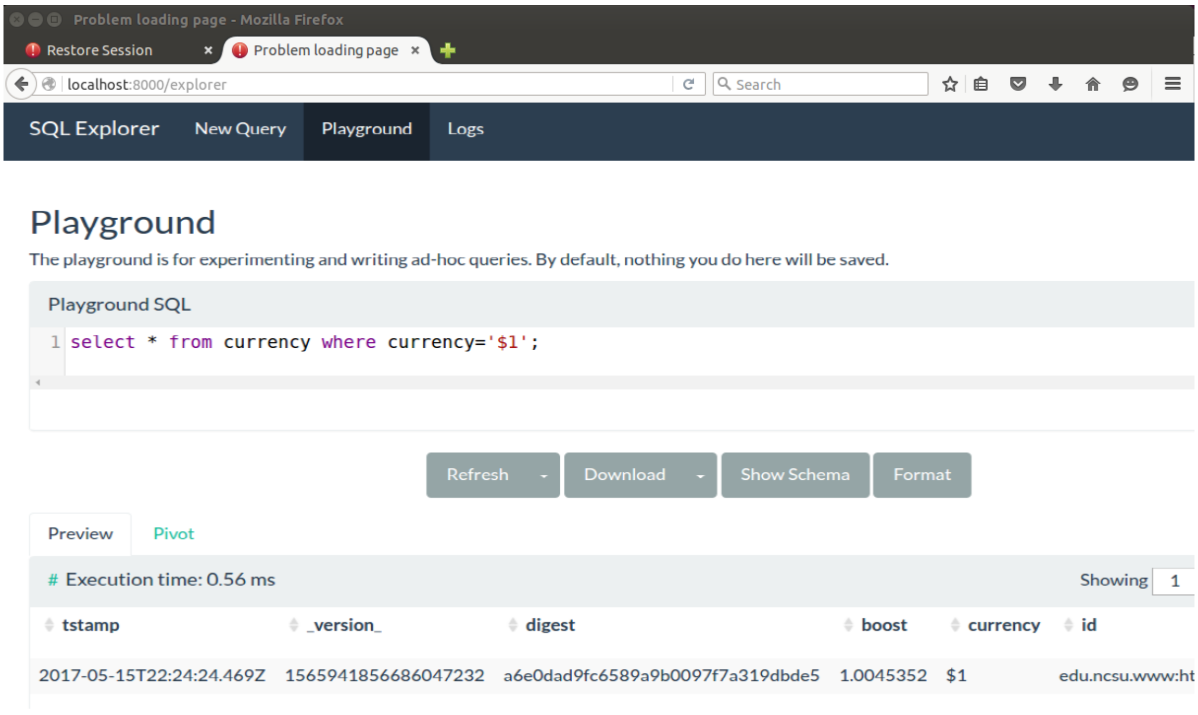
Replication (Master)

	Version	Gen	Size
Master (Searching)	1493486486369	3	321.29 KB
Master (Replicable)	1493486486369	3	-

Healthcheck

Ping request handler is not configured with a healthcheck file

Figure 4.12: Data filtered by Mortar.



Playground

The playground is for experimenting and writing ad-hoc queries. By default, nothing you do here will be saved.

Playground SQL

```sql
select * from currency where currency='$1';
```

Execution time: 0.56 ms

Showing 1

tstamp	_version_	digest	boost	currency	id
2017-05-15T22:24:24.469Z	1565941856686047232	a6e0dad9fc6589a9b0097f7a319dbde5	1.0045352	$1	edu.ncsu.www:ht

Figure 4.13: Data Visualization Using Portrait.

Chapter 5

Verification and Validation

Various scenarios have been tested to ensure that our implementation of analytics cloud platform meets all the functional and nonfunctional requirements.

5.1 Test Cases

In this section, we discuss the test cases we formulated to validate the functional and nonfunctional requirements as listed in Section 2.

5.1.1 Functional Requirement 1 – Built on RHEL 7.3

Test case: Verify that host physical machine has RHEL 7 as base operating system
Test steps:

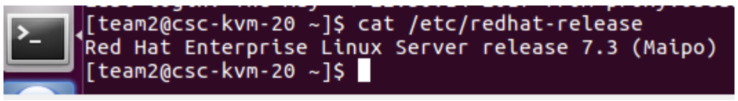
1. The base host machine (Intel blade server) already came with RHEL 7.3 installed on it.
2. We verified this using command – *cat /etc/redhat-release*

Expected Behavior	**Observed Behavior**
RHEL 7.3 is installed as the base operating system	As expected, refer figure 5.1

Table 5.1: Test case – Built on RHEL 7.3.

5.1.2 Functional Requirement 2 – Cloud built using OpenStack Platform 10

Test case: Verify that OpenStack Platform 10 is installed using PackStack.
Test steps:



```
[team2@csc-kvm-20 ~]$ cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.3 (Maipo)
[team2@csc-kvm-20 ~]$ 
```

Figure 5.1: RHEL 7.3 base OS.

1. We verify that OpenStack is installed and all the services are up and running

2. We verified this using command – *openstack-status*

Expected Behavior	**Observed Behavior**
OpenStack is successfully deployed on top of RHEL 7.3 and all the essential components of OpenStack like Nova, Neutron and Cinder are up and running	As expected, refer figure 4.6

Table 5.2: Test case – Cloud built using OpenStack Platform 10.

5.1.3 Functional Requirement 3 – Database VM

Test case: Verify that the cloud platform must have customized database image.
Test steps:

1. We verified this using command – *nova list*

Expected Behavior	**Observed Behavior**
Database VM instance has been created and configured	As expected, refer figure 4.8

Table 5.3: Test case – Database VM.

5.1.4 Functional Requirement 4 – Analytics VM

Test case: Verify that the cloud platform must have customized analytics image.
Test steps:

1. We verified this using command – *nova list*

Expected Behavior	**Observed Behavior**
Analytics VM instance has been created and configured	As expected, refer figure 4.8

Table 5.4: Test case – Analytics VM.

5.1.5 Functional Requirement 5 – GPFS Distributed store

Test case: Verify that the glance is configured to use the GPFS Distributed store for storage of image.
Test steps:

1. Run the command – *cd /gpfs/team2/glance/images/*
2. Run the command – *ls -l*
3. Verify that images are present in this directory

Expected Behavior	Observed Behavior
On adding the customized images to – Glance, they are stored in the /gpfs/glance directory as indicated	As expected, refer figure 4.7

Table 5.5: Test case – GPFS distributed store.

5.1.6 Functional Requirement 6 – Database VM for storage of both structured and unstructured data

Test case: Verify that the database VM has capabilities to store both structured data in PostgreSQL 9.2.1 [10] and unstructured data in Cassandra 2.2.8 [2].
Test steps:

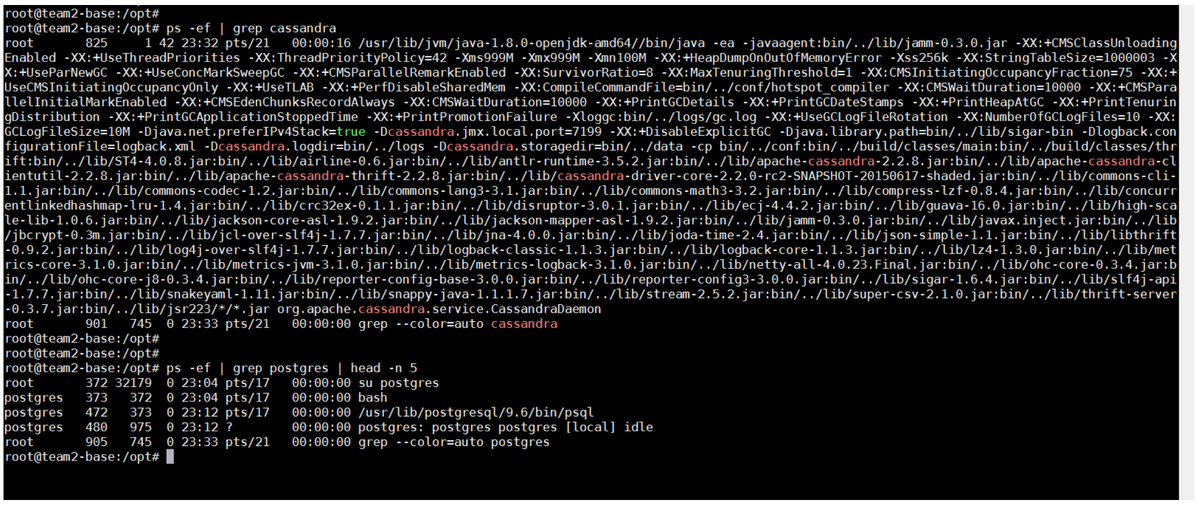
1. ssh into Database VM
2. Run command – *ps -ef | grep cassandra* to show running Cassandra process
3. Run command – *ps -ef | grep postgres* to show running PostgreSQL process

Expected Behavior	Observed Behavior
Database VM is correctly configured with Cassandra [2] and PostgreSQL [10]	As expected, refer figure 5.2

Table 5.6: Test case – Database VM for storage of both structured and unstructured data.

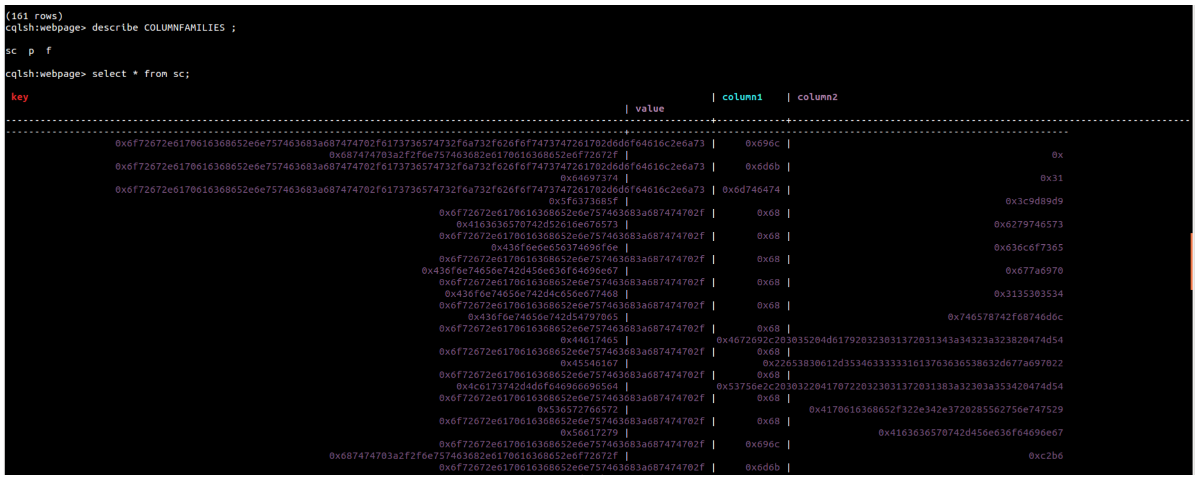
5.1.7 Functional Requirement 7 – Analytics VM for web crawling

Test case: Verify that the Analytics VM has capabilities to crawl the web given input seed URLs and gather unstructured data.
Test steps:



```
root@team2-base:/opt#
root@team2-base:/opt# ps -ef | grep cassandra
root       825     1 42 23:32 pts/21   00:00:16 /usr/lib/jvm/java-1.8.0-openjdk-amd64//bin/java -ea -javaagent:bin/../lib/jamm-0.3.0.jar -XX:+CMSClassUnloading
Enabled -XX:+UseThreadPriorities -XX:ThreadPriorityPolicy=42 -Xms999M -Xmx999M -Xmn100M -XX:+HeapDumpOnOutOfMemoryError -Xss256k -XX:StringTableSize=1000003 -X
X:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled -XX:SurvivorRatio=8 -XX:MaxTenuringThreshold=1 -XX:CMSInitiatingOccupancyFraction=75 -XX:+
UseCMSInitiatingOccupancyOnly -XX:+UseTLAB -XX:+PerfDisableSharedMem -XX:CompileCommandFile=bin/../conf/hotspot_compiler -XX:CMSWaitDuration=10000 -XX:+CMSPara
llelInitialMarkEnabled -XX:+CMSEdenChunksRecordAlways -XX:CMSWaitDuration=10000 -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintHeapAtGC -XX:+PrintTenurin
gDistribution -XX:+PrintGCApplicationStoppedTime -XX:+PrintPromotionFailure -Xloggc:bin/../logs/gc.log -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:
GCLogFileSize=10M -Djava.net.preferIPv4Stack=true -Dcassandra.jmx.local.port=7199 -XX:+DisableExplicitGC -Djava.library.path=bin/../lib/sigar-bin -Dlogback.con
figurationFile=logback.xml -Dcassandra.logdir=bin/../logs -Dcassandra.storagedir=bin/../data -cp bin/../conf:bin/../build/classes/main:bin/../build/classes/thr
ift:bin/../lib/ST4-4.0.8.jar:bin/../lib/airline-0.6.jar:bin/../lib/antlr-runtime-3.5.2.jar:bin/../lib/apache-cassandra-2.2.8.jar:bin/../lib/apache-cassandra-cl
ientutil-2.2.8.jar:bin/../lib/apache-cassandra-thrift-2.2.8.jar:bin/../lib/cassandra-driver-core-2.2.0-rc2-SNAPSHOT-20150617-shaded.jar:bin/../lib/commons-cli-
1.1.jar:bin/../lib/commons-codec-1.2.jar:bin/../lib/commons-lang3-3.1.jar:bin/../lib/commons-math3-3.2.jar:bin/../lib/compress-lzf-0.8.4.jar:bin/../lib/concurr
entlinkedhashmap-lru-1.4.jar:bin/../lib/crc32ex-0.1.1.jar:bin/../lib/disruptor-3.0.1.jar:bin/../lib/ecj-4.4.2.jar:bin/../lib/guava-16.0.jar:bin/../lib/high-sca
le-lib-1.0.6.jar:bin/../lib/jackson-core-asl-1.9.2.jar:bin/../lib/jackson-mapper-asl-1.9.2.jar:bin/../lib/jamm-0.3.0.jar:bin/../lib/javax.inject.jar:bin/../lib
/jbcrypt-0.3m.jar:bin/../lib/jcl-over-slf4j-1.7.7.jar:bin/../lib/jna-4.0.0.jar:bin/../lib/joda-time-2.4.jar:bin/../lib/json-simple-1.1.jar:bin/../lib/libthrift
-0.9.2.jar:bin/../lib/log4j-over-slf4j-1.7.7.jar:bin/../lib/logback-classic-1.1.3.jar:bin/../lib/logback-core-1.1.3.jar:bin/../lib/lz4-1.3.0.jar:bin/../lib/met
rics-core-3.1.0.jar:bin/../lib/metrics-jvm-3.1.0.jar:bin/../lib/metrics-logback-3.1.0.jar:bin/../lib/netty-all-4.0.23.Final.jar:bin/../lib/ohc-core-0.3.4.jar:b
in/../lib/ohc-core-j8-0.3.4.jar:bin/../lib/reporter-config-base-3.0.0.jar:bin/../lib/reporter-config3-3.0.0.jar:bin/../lib/sigar-1.6.4.jar:bin/../lib/slf4j-api
-1.7.7.jar:bin/../lib/snakeyaml-1.11.jar:bin/../lib/snappy-java-1.1.1.7.jar:bin/../lib/stream-2.5.2.jar:bin/../lib/super-csv-2.1.0.jar:bin/../lib/thrift-server
-0.3.7.jar:bin/../lib/jsr223/*/*.jar org.apache.cassandra.service.CassandraDaemon
root       901   745  0 23:33 pts/21   00:00:00 grep --color=auto cassandra
root@team2-base:/opt#
root@team2-base:/opt#
root@team2-base:/opt# ps -ef | grep postgres | head -n 5
root       372 32179  0 23:04 pts/17   00:00:00 su postgres
postgres   373   372  0 23:04 pts/17   00:00:00 bash
postgres   472   373  0 23:12 pts/17   00:00:00 /usr/lib/postgresql/9.6/bin/psql
postgres   480   975  0 23:12 ?        00:00:00 postgres: postgres postgres [local] idle
root       905   745  0 23:33 pts/21   00:00:00 grep --color=auto postgres
root@team2-base:/opt# 
```

Figure 5.2: Cassandra and Postgresql services are running.



```
(161 rows)
cqlsh:webpage> describe COLUMNFAMILIES ;

sc  p  f

cqlsh:webpage> select * from sc;

 key                                                                                                            | column1    | column2
                                                                                              | value
------------------------------------------------------------------------------------------------------------------+------------+------------------------------------------------------
----------------------------------------------------------------------------------------------+---------------------------------------------------------------------------
         0x6f72672e6170616368652e6e757463683a687474702f6173736574732f6a732f626f6f7473747261702d6d6f64616c2e6a73 |     0x696c |
                                                      0x687474703a2f2f6e757463682e6170616368652e6f72672f |                                                                    0x
         0x6f72672e6170616368652e6e757463683a687474702f6173736574732f6a732f626f6f7473747261702d6d6f64616c2e6a73 |     0x6d6b |
                                                                                          0x64697374 |                                                                  0x31
         0x6f72672e6170616368652e6e757463683a687474702f6173736574732f6a732f626f6f7473747261702d6d6f64616c2e6a73 | 0x6d746474 |
                                                                                        0x5f6373685f |                                                            0x3c9d89d9
                                                         0x6f72672e6170616368652e6e757463683a687474702f |       0x68 |
                                                                 0x4163636570742d52616e676573 |                                                          0x6279746573
                                                         0x6f72672e6170616368652e6e757463683a687474702f |       0x68 |
                                                                       0x436f6e6e656374696f6e |                                                          0x636c6f7365
                                                         0x6f72672e6170616368652e6e757463683a687474702f |       0x68 |
                                                           0x436f6e74656e742d456e636f64696e67 |                                                            0x677a6970
                                                         0x6f72672e6170616368652e6e757463683a687474702f |       0x68 |
                                                                0x436f6e74656e742d4c656e677468 |                                                          0x3135303534
                                                         0x6f72672e6170616368652e6e757463683a687474702f |       0x68 |
                                                                   0x436f6e74656e742d54797065 |                                                0x746578742f68746d6c
                                                         0x6f72672e6170616368652e6e757463683a687474702f |       0x68 |
                                                                                  0x44617465 | 0x4672692c203035204d617920323031372031343a34323a323820474d54
                                                         0x6f72672e6170616368652e6e757463683a687474702f |       0x68 |
                                                                                  0x45546167 |       0x22653830612d3534633333316137636365386322d677a697022
                                                         0x6f72672e6170616368652e6e757463683a687474702f |       0x68 |
                                                                 0x4c6173742d4d6f646966696564 | 0x53756e2c203032204170722032303137203138a32303a353420474d54
                                                         0x6f72672e6170616368652e6e757463683a687474702f |       0x68 |
                                                                              0x536572766572 |                  0x4170616368652f322e342e3720285562756e747529
                                                         0x6f72672e6170616368652e6e757463683a687474702f |       0x68 |
                                                                                  0x56617279 |                            0x4163636570742d456e636f64696e67
                                                         0x6f72672e6170616368652e6e757463683a687474702f |     0x696c |
                                    0x687474703a2f2f6e757463682e6170616368652e6f72672f |                                                                0xc2b6
                                                         0x6f72672e6170616368652e6e757463683a687474702f |     0x6d6b |
```

Figure 5.3: Unstructured data collected by Nutch stored in Cassandra.

1. ssh into Analytics VM.
2. Place the seed URL in $NUTCH_HOME/urls/seed.txt text file. Run command – *nutch inject urls*
3. Generate a batch of top 100 URLs using injected urls by running command – *nutch generate -topN 100*
4. Run fetch job for batch-id obtained in step 3 to generate fetch-list using command – *nutch fetch <batch-id> resume*
5. Parse the fetched URLs by running command – *nutch parse <batch-id> -force*
6. Check the crawled content within the folder – */opt/apache-nutch-2.2.1/src/testresources/testcrawl/segments*

Expected Behavior	Observed Behavior
Analytics VM is correctly configured with Nutch and able to crawl the web. The seed url were stored in crawldb – a module of Nutch. A fetch-list was generated, which was crawled successfully. The crawled data was placed in segments directory of Nutch.	As expected, refer figures 4.9 and 5.3

Table 5.7: Test case – Analytics VM for web crawling.

5.1.8 Functional Requirement 8 – Analytics VM for data indexing

Test case: Verify that the Analytics VM has capabilities to index the data crawled by Nutch.
Test steps:

1. ssh into Analytics VM
2. Run steps 1-6 from test case for Functional Requirement 7
3. Next, run the command – *nutch solrindex*
4. Run select query on Solr dashboard to view indexed data

Expected Behavior	Observed Behavior
Solr dashboard displays the indexed data uploaded from Nutch	As expected, refer figures 5.4, 5.5, 4.10 and 4.12

Table 5.8: Test case – Analytics VM for data indexing.

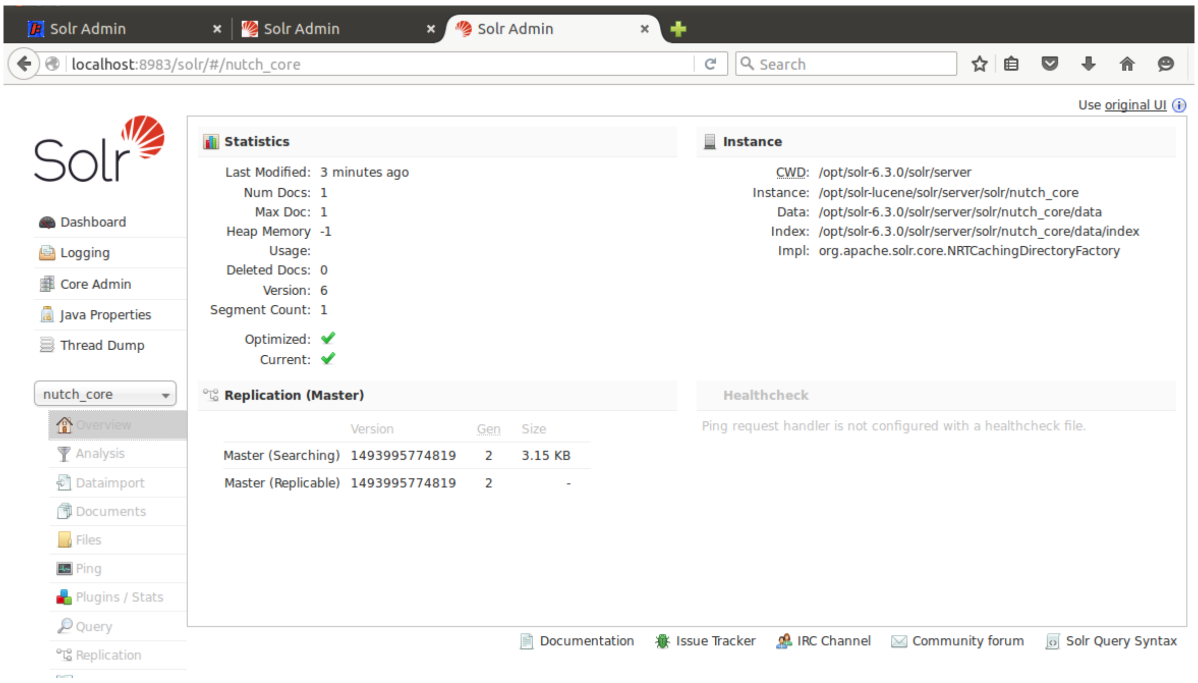
5.1.9 Functional Requirement 9 – Analytics VM for data filtering

Test case: Verify that the Analytics VM has capabilities to filter the data indexed by Solr using installed package Mortar.
Test steps:

1. Log into the Mortar interface.
2. Create a filter tree and Query Solr with that filter tree.

Expected Behavior	Observed Behavior
Mortar interface is used to query data using PESTLE tree. Filtered data can be viewed using the Mortar interface	As expected, refer figure 4.12

Table 5.9: Test case – Analytics VM for data filtering.



Statistics

Last Modified:	3 minutes ago
Num Docs:	1
Max Doc:	1
Heap Memory Usage:	-1
Deleted Docs:	0
Version:	6
Segment Count:	1
Optimized:	✓
Current:	✓

Instance

CWD:	/opt/solr-6.3.0/solr/server
Instance:	/opt/solr-lucene/solr/server/solr/nutch_core
Data:	/opt/solr-6.3.0/solr/server/solr/nutch_core/data
Index:	/opt/solr-6.3.0/solr/server/solr/nutch_core/data/index
Impl:	org.apache.solr.core.NRTCachingDirectoryFactory

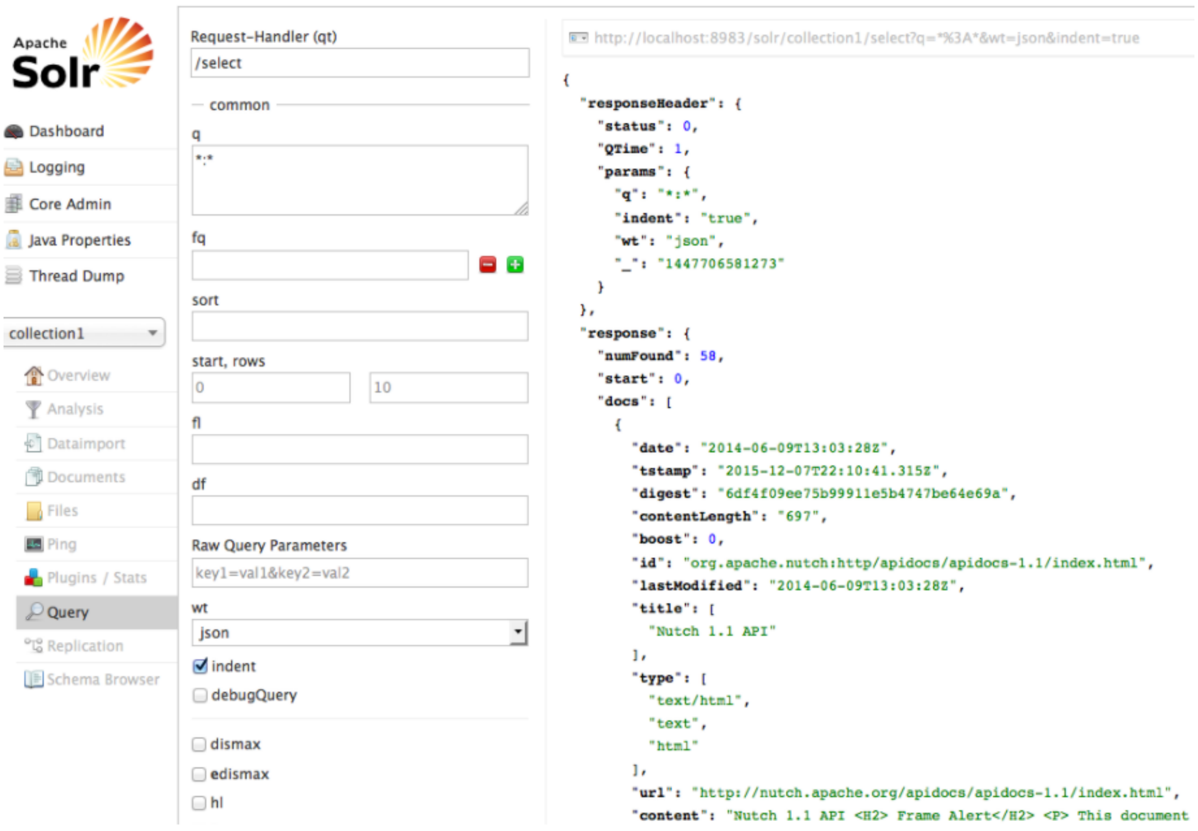
Replication (Master)

	Version	Gen	Size
Master (Searching)	1493995774819	2	3.15 KB
Master (Replicable)	1493995774819	2	-

Healthcheck

Ping request handler is not configured with a healthcheck file.

Figure 5.4: Nutchcore entry in Solr.



Request-Handler (qt)	/select
common	
q	*:*
fq	
sort	
start, rows	0 10
fl	
df	
Raw Query Parameters	key1=val1&key2=val2
wt	json
indent	✓
debugQuery	☐
dismax	☐
edismax	☐
hl	☐

http://localhost:8983/solr/collection1/select?q=*%3A*&wt=json&indent=true

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "q": "*:*",
      "indent": "true",
      "wt": "json",
      "_": "1447706581273"
    }
  },
  "response": {
    "numFound": 58,
    "start": 0,
    "docs": [
      {
        "date": "2014-06-09T13:03:28Z",
        "tstamp": "2015-12-07T22:10:41.315Z",
        "digest": "6df4f09ee75b99911e5b4747be64e69a",
        "contentLength": "697",
        "boost": 0,
        "id": "org.apache.nutch:http/apidocs/apidocs-1.1/index.html",
        "lastModified": "2014-06-09T13:03:28Z",
        "title": [
          "Nutch 1.1 API"
        ],
        "type": [
          "text/html",
          "text",
          "html"
        ],
        "url": "http://nutch.apache.org/apidocs/apidocs-1.1/index.html",
        "content": "Nutch 1.1 API <H2> Frame Alert</H2> <P> This document
```

Figure 5.5: Querying Solr.

5.1.10 Functional Requirement 10 – Data import and export

Test case: Verify that the Analytics VM has capabilities to export filtered data in Solr in CSV format. Verify that the Database VM has capabilities to import the CSV file into PostgreSQL [7]
Test steps:

1. Export data from Solr into CSV format.

2. Import CSV file into PostGreSQL

Expected Behavior	**Observed Behavior**
The structured data from the CSV file is present in PostGreSQL	As expected

Table 5.10: Test case – Data import and export.

5.1.11 Functional Requirement 11 – Analytics VM for data visualization

Test case: Verify that the Analytics VM has capabilities to visualize the structured data stored in PostgreSQL server on Database VM through Portrait interface.
Test steps:

1. Log into Portrait using appropriate credentials

2. Query structured data and verify if it is correct

Expected Behavior	**Observed Behavior**
The structured data from PostGreSQL can be visualized using Portrait interface	As expected, refer figure 4.13

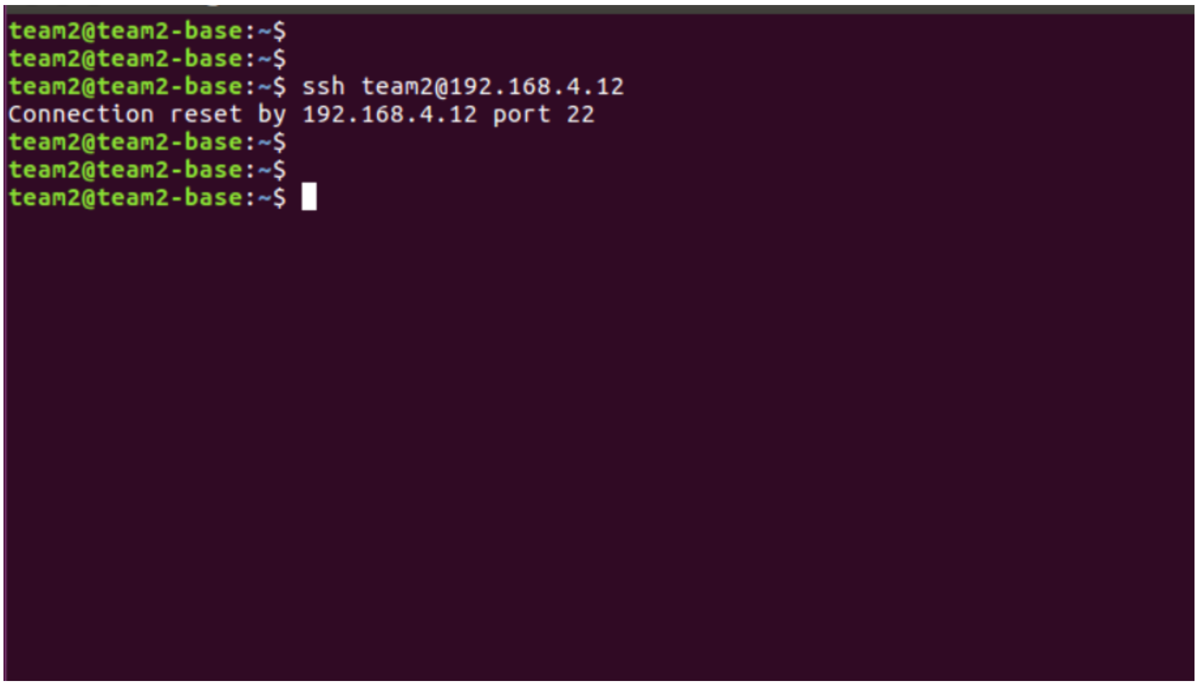
Table 5.11: Test case – Analytics VM for data visualization.

5.1.12 Functional Requirement 12 – Authentication and Authorization

Test case: Verify that the analytics cloud platform prevents unauthorized access to VMs.
Test steps:

1. ssh into client DEV VM.

2. Try pinging, logging into Database VM.



```
team2@team2-base:~$
team2@team2-base:~$
team2@team2-base:~$ ssh team2@192.168.4.12
Connection reset by 192.168.4.12 port 22
team2@team2-base:~$
team2@team2-base:~$
team2@team2-base:~$
```

Figure 5.6: Client cannot SSH into Database VM

Expected Behavior	**Observed Behavior**
Client should be unable to SSH into database VM due to the firewall rules that are in place to ensure there is no unauthorized access to database VM. One can connect to database machine only on the 2 thrift client ports assigned for Cassandra and PostgreSQL	As expected, refer figure 5.6

Table 5.12: Test case – Authentication and Authorization.

Chapter 6

Schedule and Personnel

6.1 Schedule

Date	Agenda
02/02/2017	Meeting 1 - Meeting with Prof Streck and Prof Hall to get the system requirements and understanding of base architecture.
02/02/2017	Meeting 2 - Meeting among team-members to discuss problem statement, clear use-case ambiguities and plan initial tasks.
02/07/2017	Meeting 3 - Got ssh access to physical server. Team meeting to ensure every member has access to the physical server setup on their personal machines.
02/09/2017	Milestone 1 - OpenStack installation on physical server. Deployment carried out using PackStack. Met Prof Hall post lecture to resolve issues occurring in accessing Apache server running on the physical machine.
03/02/2017	Meeting 4 - Meeting with Dr. Kowolenko, Prof Streck and Prof Hall to gather requirements on Big Data Analytics and what the application should offer an end-user.
03/06/2017	Meeting 5 - Meeting among team-members to decide on the open source frameworks for databases, data ingestion, analytics and UI that need to be installed on the image. Each member was assigned a set of frameworks to understand, install and create a script for their installation.
03/14/2017	Meeting 6 - Meeting among team-members to assess each other‚Äôs progress on the assigned tasks.
03/15/2017	Milestone 2 - Ubuntu 16.04.02 LTS base image creation

03/16/2017	Meeting 7 - Meeting among team-members to complete the intermediate project report. Milestone 3 - Image creation with the database, data ingestion, analytics and UI tools/packages.
03/24/2017	Meeting 8 - Meeting among team-members to decide on presentation topics.
03/26/2017	Milestone 4 - Proof of Concept for Nutch application for SaaS case.
03/27/2017	Milestone 5 - Moving images from machine to /gpfs directory
03/28/2017	Milestone 6 - Developing intermediate project presentation slides
03/30/2017	First Project Presentation
04/18/2017	Meeting 9 - End-to-end integration plan and documentation discussion
04/19/2017	Milestone 7 - Proof of Concept for Cassandra data ingestion for PaaS case
04/28/2017	Milestone 8 - Nutch-Cassandra-Solr Integration
05/02/2017	Second Project Presentation
05/03/2017	Milestone 9 - Mortar, Potrait & UIMA installation and testing
05/05/2017	Milestone 10 - End to end integration and validation of entire model
05/05/2017	Milestone 11 - Submit final report

Table 6.1: Schedules.

6.2 Task Distribution

Task	Rushikesh	Karan	Payal	Nithya	Ran
Meeting 1 02/02/2017	✓	✓	✓	✓	✓
Milestone 1 OpenStack installation 02/09/2017	✓	✓	✓	✓	✓
Meeting 4 03/02/2017	✓	✓	✓	✓	✓
Milestone 2 Ubuntu base image creation 03/15/2017		✓			✓
Milestone 3 Image creation	✓	✓	✓	✓	✓
Packages installation	✓	✓	✓	✓	✓
Milestone 4 Nutch POC	✓		✓		
Milestone 5 Move images to /gpfs		✓	✓		
Milestone 6 Intermediate project presentation meeting	✓	✓		✓	✓
First project presentation 03/30/2017	✓	✓	✓	✓	✓

Documentation update 04/12/2017			✓	✓	
Milestone 7 04/19/2017	✓			✓	
Milestone 8 Nutch-Cassandra-Solr Integration 04/28/2017	✓	✓	✓	✓	
Milestone 9 Mortar, Portrait & UIMA installation and testing 05/03/2017	✓		✓		✓
Milestone 10 End to end integration and validation of entire model 05/05/2017	✓	✓	✓	✓	✓
Milestone 11 Submit final report 05/05/2017	✓	✓	✓	✓	✓

Table 6.2: Task Distribution.

Bibliography

[1] *Apache Ant*.
`http://ant.apache.org/`.
Apache Ant - Pure Java build tool, simpler and easier to use than GNU Make.

[2] *Apache Cassandra*.
`http://cassandra.apache.org`.
Apache Cassandra - A free and open-source distributed NoSQL database management system.

[3] *Apache Solr*.
`http://lucene.apache.org/solr/`.
Apache Solr - An open-source search server built on Apache Lucene.

[4] *Apache UIMA*.
`https://uima.apache.org/`.
Apache UIMA - Unstructured Information Management application which analyzes large volumes of unstructured information in order to discover knowledge that is relevant to an end user.

[5] *Django: The web framework for perfectionists with deadlines*.
`https://www.djangoproject.com/`.
Django – A complete high-level web framework that encourages rapid development and clean, pragmatic design.

[6] *Highly extensible, highly scalable Web crawler*.
`http://nutch.apache.org/`.
Apache Nutch - A highly extensible, highly scalable Web crawler.

[7] *Import CSV data into postgresql, the comfortable way ;-)*.
`http://blog.aplikate.eu/2015/01/03/import-csv-data-into-postgresql-the-comfortable-way/`.
Reference for importing CSV data into PostgreSQL.

[8] *Neutron with existing external network*.
`https://www.rdoproject.org/networking/neutron-with-existing-external-network/`.

Neutron with existing external network - This site was followed to setup OpenStack PackStack to integrate with the current network infrastructure on the host. As discussed in the appendix, the commands on this page had to be modi ed to work with the current OpenStack version and tools.

[9] *PackStack*.
`https://www.rdoproject.org/install/quickstart/`.
Packstack – Packstack is a utility that uses Puppet modules to deploy various parts of OpenStack.

[10] *PostgreSQL*.
`https://www.postgresql.org/`.
PostgreSQL - PostgreSQL is a powerful, open source object-relational database system.

[11] J. STRECK, *Project abstracts for spring 2017 cloud computing (csc 547) projects 1 through 4 configure a cloud for big data analytics*.
`https://github.ncsu.edu/nkumar8/CSC547_S17_Project/tree/master/Documentation/project_abstracts.pdf`, 2017.
This is the first document we received to get an initial scope of the project requirements.

[12] M. JOHNSON, *Alchemy cookbook open source text analytics system setup and usage*.
`https://github.ncsu.edu/nkumar8/CSC547_S17_Project/blob/master/AlchemyCookbook.pdf`, 2017.
This guide was used to design the data analysis engines, install the required packages, and create a workflow.

[13] M. KOWOLENKO AND M. A. VOUK, *Developing an open source 'bigdata' cognitive computing platform*.
This document is a reference paper for Big Data Analytics. It was used to gain better understanding of the data analytics platform and aiding us to build the needed data analytics engine for our project.

[14] UBUNTU, *Ubuntu 16.04.2 LTS (Xenial Xerus)*.
`http://releases.ubuntu.com/16.04/`.
Ubuntu – An open source operating system. This particular version of Ubuntu was installed as the base image in our project.

[15] W. ZHU AND M. GUPTA AND V. KUMAR AND S. PEREPA AND A. SATHI AND C. STATCHUK, *Building Big Data and Analytics Solutions in the Cloud*.
`http://www.redbooks.ibm.com/redpapers/pdfs/redp5085.pdf`.
This is the reference paper for cloud issues and environment in a general big data analytics platform. We used this to narrow down the issues we need to address in our project.

APPENDIX A

1. Installation of OpenStack using PackStack

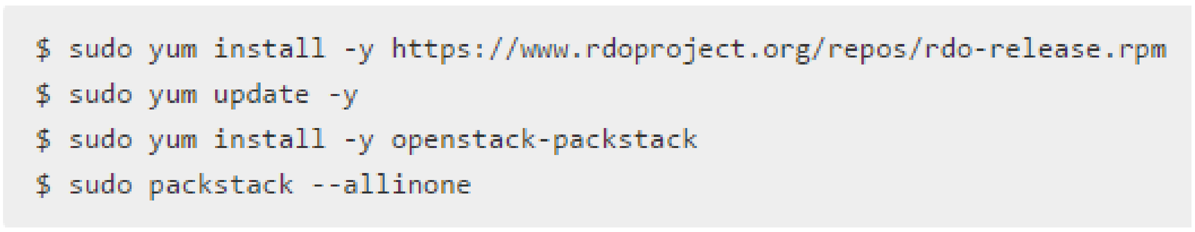
This section covers the commands used to install OpenStack with PackStack. Packstack is a command line utility suitable for deploying both single node proof of concept installations and more complex multi-node installations.

As suggested by Mr. Hall, to install OpenStack using PackStack, we use instructions from PackStack documentation [8].

The figure above shows the commands to install OpenStack using PackStack utility.During the installation process, we faced an error as follows while running the last command – *sudo packstack –allinone*

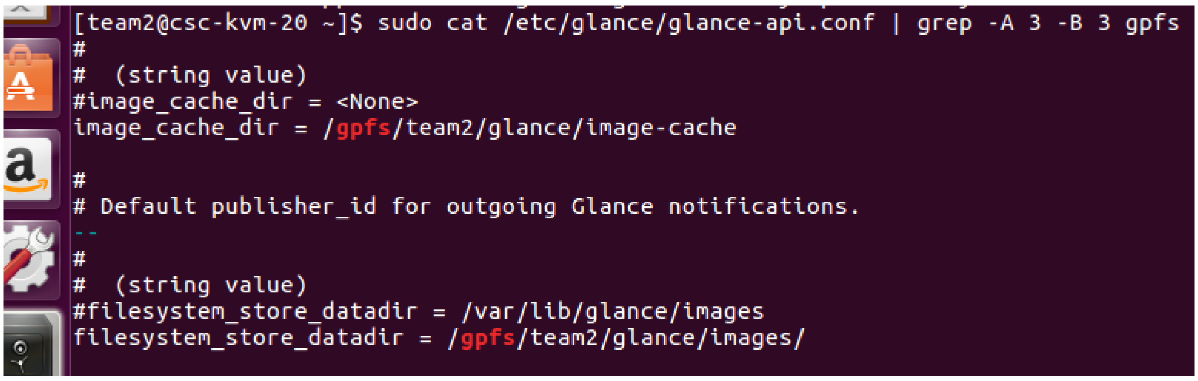
```
ERROR : Error appeared during Puppet run: 10.26.20.10_controller.pp
Error: Execution of '/usr/bin/yum -d 0 -e 0 -y install openstack-gnocchi-metricd'
returned 1: Transaction check error:
You will find full trace in log
/var/tmp/packstack/20170209-135026-rJ3Pz0/manifests/10.26.20.10_controller.pp.log
```

In order to resolve this error we executed command: *sudo yum remove python-cadf* and re-ran the last command – *sudo packstack –allinone*. On completion, OpenStack had been successfully installed on our machine. Test case for functional requirement 2.1.2. Cloud built using OpenStack Platform 10 correctly validates that all components are up and running.



```
$ sudo yum install -y https://www.rdoproject.org/repos/rdo-release.rpm
$ sudo yum update -y
$ sudo yum install -y openstack-packstack
$ sudo packstack --allinone
```

Figure 1: OpenStack install using PackStack.



```
[team2@csc-kvm-20 ~]$ sudo cat /etc/glance/glance-api.conf | grep -A 3 -B 3 gpfs
#
#  (string value)
#image_cache_dir = <None>
image_cache_dir = /gpfs/team2/glance/image-cache

#
# Default publisher_id for outgoing Glance notifications.
--
#
#  (string value)
#filesystem_store_datadir = /var/lib/glance/images
filesystem_store_datadir = /gpfs/team2/glance/images/
```

Figure 2: Configuring Glance to use GPFS.

2. Configuring Glance Service to use GPFS for storing the images

Functional Requirement 5. GPFS Distributed store stated that all images in the analytics cloud platform must be stored to the provided IBM's General Parallel File System (GPFS) distributed store. This section identifies the steps taken to complete this requirement.

The configuration of glance service to use GPFS as storage backend by changing two parameter values in file /etc/glance/glance-api.conf. We modify parameters as follows:

1. Set parameter filesystem_store_datadir value to /gpfs/team2/glance/images/
2. Set parameter image_cache_dir value to /gpfs/team2/glance/image-cache

3. Base image creation

This section of the Appendix details the steps followed to create our Ubuntu 16.04 Base image. Following are the steps followed:

1. Download Ubuntu 16.04 ISO from ubuntu website.
2. Install qemu-kvm and virt-manager
   Command – *sudo apt-get install qemu-kvm; sudo apt-get install virt-manager*
3. Create Qemu Image
   Command – *qemu-img create -f qcow2 /tmp/ubuntu_team2.qcow2 10G*
4. Use virt-install to start the image installation
   Command – *virt-install –virt-type kvm –name ubuntu –ram 1024 cdrom=/path/to/iso/ubuntu64.iso –disk /tmp/ubuntu_team2.qcow2,format=qcow2 –network network=default -graphics vnc,listen=0.0.0.0 –noautoconsole –os-type=linux*

5. Use Virt-Manager GUI to complete ubuntu installation.

6. After Completing the installation, Start the VM, Login into it and install various required packages.
   Commands – *sudo apt-get install openssh-server; sudo apt-get install cloud-init cloud-utils; dpkg-reconfigure cloud-init*

7. Shutdown the instance
   Command – *virsh stop ubuntu*

8. Cleanup Mac Address
   Command – *virt-sysprep -d ubuntu*

9. Undefine the libvirt domain
   Command – *virsh undefine ubuntu*

10. Compress created image
    Command – *virt-sparsify –compress /tmp/ubuntu_team2.qcow2 ubuntu_team2.qcow2*

11. Upload the newly created image

4. Installation of Software Stack on customized images

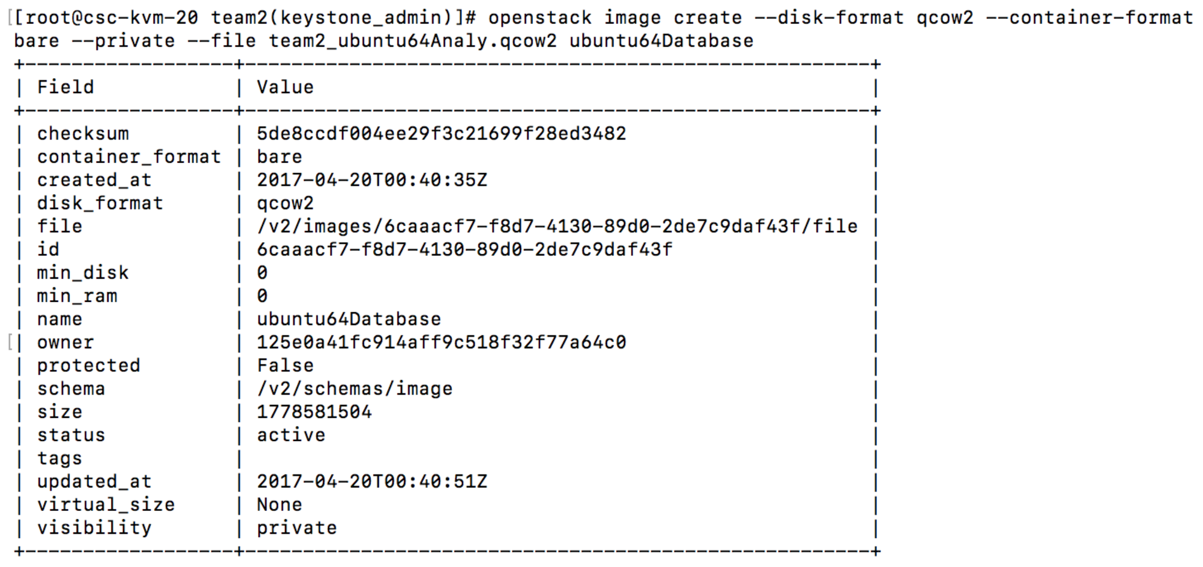
This section details the commands used for creation of Database and analytics VM and the steps followed to install the software stack as described in Section 4.1.2.

- Command to create a customized Analytics image
  openstack image create –disk-format qcow2 –container-format bare –private –file team2_ubuntu64Analy.qcow2 ubuntu64Analytics

- Command to create a customized Database image
  openstack image create –disk-format qcow2 –container-format bare –private –file team2_ubuntu64Analy.qcow2 ubuntu64Database

The figure below illustrates the command run to create the customized Database image.

Once the customized images are set up, Database and Analytics instances are brought up. We then install the software stack on each of these instances using Meagan,Äôs steps. Since Meagan,Äôs steps [12] were for a Centos Base image, and our base image is Ubuntu, we adopted her script steps and installed the softwares. Re-iterating the steps in the Alchemy cookbook shall be redundant, hence we do not repeat the steps here. The adapted scripts for installations can be accessed at
https://github.ncsu.edu/nkumar8/CSC547_S17_Project/tree/master/MeganScripts



```
[[root@csc-kvm-20 team2(keystone_admin)]# openstack image create --disk-format qcow2 --container-format
 bare --private --file team2_ubuntu64Analy.qcow2 ubuntu64Database
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | 5de8ccdf004ee29f3c21699f28ed3482                     |
| container_format | bare                                                 |
| created_at       | 2017-04-20T00:40:35Z                                 |
| disk_format      | qcow2                                                |
| file             | /v2/images/6caaacf7-f8d7-4130-89d0-2de7c9daf43f/file |
| id               | 6caaacf7-f8d7-4130-89d0-2de7c9daf43f                 |
| min_disk         | 0                                                    |
| min_ram          | 0                                                    |
| name             | ubuntu64Database                                     |
[| owner            | 125e0a41fc914aff9c518f32f77a64c0                     |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 1778581504                                           |
| status           | active                                               |
| tags             |                                                      |
| updated_at       | 2017-04-20T00:40:51Z                                 |
| virtual_size     | None                                                 |
| visibility       | private                                              |
+------------------+------------------------------------------------------+
```

Figure 3: Adding the customized image.

In Order to aid the data scientist, to customize the image further to his likings, we prepared scripts for installation of various data analysis packages like pip, anaconda, nltk, scikit etc. These additional software package installation scripts can be accessed at https://github.ncsu.edu/nkumar8/CSC547_S17_Project/tree/master/Scripts