# INFSCI 2750: Cloud Computing

# Mini Project 2 - Group 4

# Spring 2024

**Part 1: Setting up Cassandra (50 points)**

**Configuring a Cassandra Cluster on a Three-Node Cluster**

This documentation guides through the process of configuring a Cassandra distribution on a cluster of VMs. Cassandra's "master-less" architecture allows all nodes to be configured in the same way, making it suitable for a multi-node cluster setup. We set up a two/three-node cluster using Ubuntu and Debian packages for Cassandra installation.

* VMs running Ubuntu with sufficient resources for Cassandra.
* Java 8 installed on all VMs.

1. **Installation:**

On Ubuntu, Cassandra was installed easily from the Debian packages:

1. **Configuration:**

The Cassandra configuration file `cassandra.yaml` located at `/etc/cassandra/` on all nodes was edited. The nodes were configured to work together by updating the following.

* `seeds: "IP\_of\_Node1, IP\_of\_Node2, IP\_of\_Node3"` - Specified the IP addresses of the nodes.
* `listen\_address: IP\_of\_Current\_Node` - Set the listen address of each node.
* `rpc\_address: IP\_of\_Current\_Node` - Set the RPC address of each node.

1. **Starting Cassandra Nodes**

* Any existing Hadoop and Spark services were stopped to free up memory.
* The Cassandra service was started on all nodes.

1. **Monitoring and Verification:**

*nodetool* was used to check the status of the Cassandra cluster:

It was verified that all nodes were up and part of the cluster. (attach screenshot)  
  
A screenshot of a computer

Description automatically generated

A black background with a black square

Description automatically generated with medium confidence

A black background with a black square

Description automatically generated with medium confidence

A close-up of a number

Description automatically generated

**Test the setup!**

1. **Setting Up Keyspace and Table:**

* We used the Cassandra Query Language (CQL) to import the access logs into Cassandra. First, we created a keyspace and a table to store all the logs.
* created a keyspace named `patient` with replication settings.
* created a table named `exam` to store log details, including `patient\_id`, `id`, `date`, and `details`:

1. **Importing Data**

We used CQL commands to insert sample data into the `exam` table:

Inserted data for patient 1

Inserted data for patient 2 and patient 3

1. **Verifying Data**

To verify the data insertion, we used the following command to select all records for `patient\_id=1`:

A screenshot of a computer code

Description automatically generated

**Part 2: Import Data into Cassandra (25 points)**

To import the access logs into Cassandra using CQL or the Java driver, follow these steps:

**1. Download the log data set:**

- Download the log data set from the provided link in Canvas. (We have used the new small log dataset)

**2. Create a Key space:**

- Open the CQL shell (`cqlsh`) and create a keyspace to store the log data.   
 CREATE KEYSPACE IF NOT EXISTS log\_keyspace  
 WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': 1};

**3. Create a Table:**

- Create a table to store the log data. Define the table schema based on your log data structure.

CQL:  
CREATE TABLE IF NOT EXISTS log\_data\_small\_log (  
    id UUID PRIMARY KEY,  
    ip\_address TEXT,  
    datetime TIMESTAMP,  
    method TEXT,  
    url TEXT,  
    protocol TEXT,  
    status\_code INT,  
    response\_size INT,  
    user\_agent TEXT  
);

**4. Import Data using DataStax Bulk Loader (DSBulk):**

**-** If you have a CSV file of the log data, you can use DataStax Bulk Loader (DSBulk) to efficiently import the data into the table.

**-** Place the CSV file in a location accessible to the machine where DSBulk is installed, and then use the following command:

/home/ubuntu/Log\_data/dsbulk-1.11.0/bin/dsbulk load -k log\_keyspace -t log\_data\_small\_log -url /home/ubuntu/Log\_data/output\_log\_file\_New.csv -header true -f dsbulk.conf -h 10.254.3.108 --port 9042

- Refer to the README file included for more details on setup and usage.

**5. Verify Data:**

- After importing the data, you can verify that it has been successfully imported by querying the table:

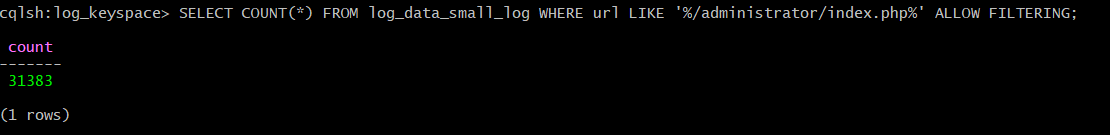
Cql: SELECT \* from log\_data\_small\_log limit 20;

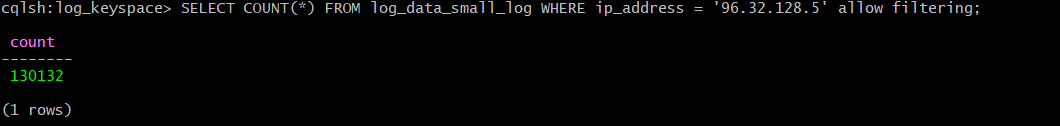
A screen shot of a computer screen

Description automatically generated

**Part 3: Operate Data in Cassandra (25 points)**

In this part of the project, we used the Cassandra Query Language (CQL) and the Java driver for Cassandra to analyze access logs. We have used the small log dataset and aimed to answer the below specific questions

1. **Problem: How many hits were made to the website item "/administrator/index.php"?**
2. **Problem: How many hits were made from the IP: 96.32.128.5?**



1. **Problem: Which path in the website has been hit most? How many hits were made to the path?**

A black background with white text

Description automatically generated

1. **Problem: Which IP accesses the website most? How many accesses were made by it?**A black background with white text

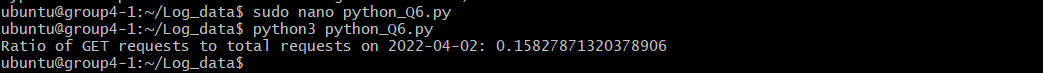
   Description automatically generated
2. **Problem: How many accesses were made by Firefox (Mozilla)?**

**For Firefox:  
A black background with white text

Description automatically generated**

**For Mozilla:  
A black background with white text

Description automatically generated**

1. **Problem: For all requests on 02/Apr/2022, what is the ratio of GET request?**
2. **Problem: How many requests are lower than or equal to 404 bytes?**  
   A black screen with white text

   Description automatically generated
3. **Problem: List the IPs that have more than ten 404 requests. If no ip fulfills, print the ip  
   that has most 404 requests and the number of requests.** A computer screen with many small white dots

   Description automatically generated with medium confidence