# PILOT STUDENTS – Oct/Nov 2015 (this applies to you)

# (1) When you see an instruction to upload something, upload it to your desktop. If faculty needs it, they will ask for it. Contact info is in your FAQ.

# Intro to NoSQL - Cassandra – LAB

Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.

## Objective

Goal of this series is to learn the difference between relational systems and Cassandra and the types of use cases it can solve.

Picking the right data model is the hardest part of using Cassandra. We will walk you through the data modeling exercise. We will import StackOverflow posts & responses into Cassandra and inspect them with node app. Here are some guidelines:

**Basic Data Modeling Goals**

These are the two high-level goals for your Cassandra data model:

* Spread data evenly around the cluster
* Minimize the number of partitions read

Data Modeling Non-Goals (for those of us with relational modeling backgrounds)

* **Minimize the Number of Writes**: Cassandra is optimized for high write throughput. If you can perform extra writes to improve the efficiency of your read queries, it’s almost always a good tradeoff. Reads tend to be more expensive and are much more difficult to tune.
* **Minimize Data Duplication:** De-normalization and duplication of data is a fact of life with Cassandra. Don’t be afraid of it. Disk space is generally the cheapest resource (compared to CPU, memory, disk IOPs, or network), and Cassandra is architected around that fact. In order to get the most efficient reads, you often need to duplicate data.

## What Am I Supposed to Do?

1. Follow the instructions of the Lab.
2. Periodically, you’ll be instruction to **STOP** and do what is requested. Typically, we ask you to copy a screen shot. **YOU MUST DO THIS IF YOU WANT TO PASS THE COURSE!**

## BEGIN CASSANDRA LAB:

**Dependencies**

Node 4.2.1, Python 2.7

1. **Get source Code:**

Make sure you have your node.js Setup from Lab 1 up and running. You’ll need the following packages unpacked and **ready** (i.E. npm install executed within their directory)

* 1. nosql-lab-dataimport
  2. nosql-lab-cassandra

1. **Install & Start Cassandra:**
   1. Download the binary release [here](http://www.planetcassandra.org/cassandra/):

* Windows: Get Cassandra binary release >=v2.1.10. At this time v2.1.11 is the stable & recommended version. Pick the right 32-bit or 64-bit MSI installer for your Operating system. Follow the installation instructions.
  1. Starting a Cassandra Node

Start, stop, and manage the various services installed via the standard Windows services control panel interface. Services to look for:

* DataStax\_Cassandra\_Community\_Server
* DataStax\_OpsCenter\_Agent
* DataStax\_OpsCenter\_Community

* 1. Familiarize yourself with the Cassandra's Ops Center. This is a great tool to monitor the performance of the database and explore data. Ops Center may require registration with DataStax.

http://localhost:8888/opscenter/index.html

* 1. You will need python to work with Cassandra CLI. in Cassandra home directory use `python cqlsh`

1. **Create Keyspace & Column Families:**

Cassandra uses a SQL dialect called CQL that allows to the users familiar with SQL to start operating in a very short amount of time

* 1. Open the cqlsh and create a new key space

cqlsh> CREATE KEYSPACE nosqllab WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': 1};

cqlsh> USE nosqllab;

* 1. Primary key model is influenced by the query pattern. Create new tables to for the following data access use cases.
     1. Data Access Use case - Pull back list of posts in descending order of postdate by reading "roughly" one partition:
* Assumption: post ID is sequential created
* We will create partitions by create date (date only). There can be many comments posted in a single day we will clustered by hour.
* With a highly active site like StackOverflow.com we may be able to get sorted data from nearly one partition.

cqlsh> CREATE TABLE post(id int, title text, body text, created timestamp, score int, lastActivityDate timestamp, commentCount int, ownerUserId int, lastEditorUserId int, creationDate text, created\_hash text, PRIMARY KEY ((creationDate, created\_hash), id)) WITH CLUSTERING ORDER BY (id DESC);

* + 1. Data Access Use case - Pull back responses of posts in asc order (reading one partition):
* Assumption: response ID is sequential created
* We will create partitions by parentID and will be sorted by comment ID in ASC order.
* Data was partioned by by parentId to get sorted data from nearly one partition.

cqlsh> CREATE TABLE response(id int, body text, created timestamp, score int, parentId int, ownerUserId int, lastEditorUserId int, PRIMARY KEY(parentId, id));

# Stop and Document Your Work

In cqlsh console type in “DESCRIBE nosqllab”. Copy and paste a screenshot showing post & comment table here and then continue.

1. **Insert the data:**

Use nosql-lab-dataimport for this:

Open the command prompt and `cd` into the directory that has the code.

After installing and doing the obligatory `npm install` to satisfy all dependencies do a `node [db.js]` to run. Please note that the Cassandra \*\*must be running\*\*.

*$ node cassandra.js*

In case of error start over with step 2 after executing DROP KEYSPACE nosqllab;

* 1. Stop and Test Your Work
     1. **STOP:** List ids of 5 posts. Paste the query you used here.
     2. **STOP:** Pick id retrieved from previous query and retrieve the tile of the post. List the error you receive here.
     3. **STOP:** Add index on id field within the post table to fix the above issue and re-run the query. Paste the query you used here.
     4. **STOP:** Retrieve id & title of responses for the post. Paste the query you used here. Explain why you do not need a secondary index in this case.

1. **Look at the data in action:**

Make sure node & npm commands can be executed from command line. If not verify steps from Lab 1.

* 1. Open the command promp and `cd` into the directory that has the code.
  2. run `npm install` to get project dependencies
  3. run `node app.js` to launch the web application

Default port for the webserver in the app is 3000, so head to

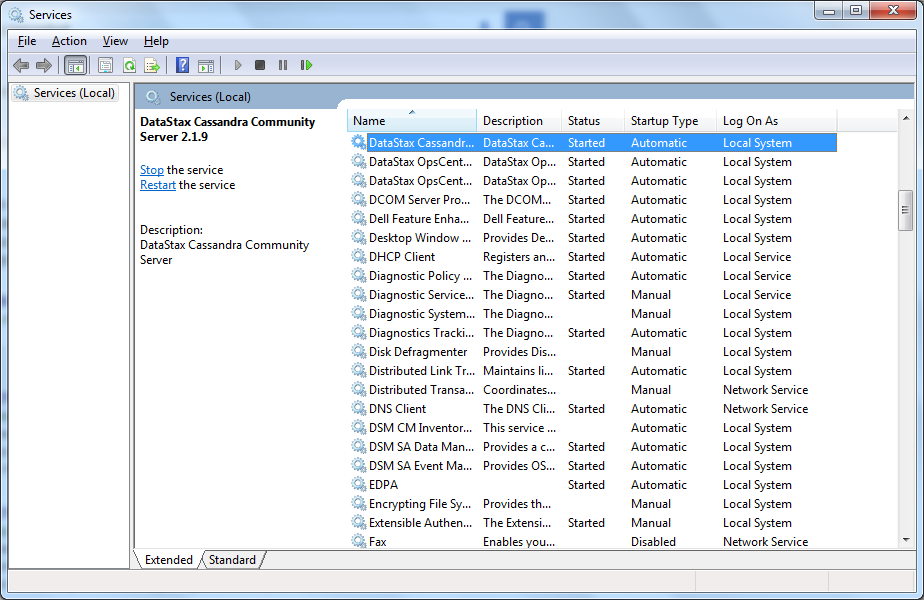
http://127.0.0.1:3000/ **\*\*now\*\*** and enjoy a warm welcome.

* 1. Stop and Document Your Work
     1. Copy and paste a screenshot for <http://127.0.0.1:3000/> here and then continue.
     2. Click into a post and copy and paste a screenshot for [http://127.0.0.1:3000/[POST\_ID]](http://127.0.0.1:3000/14) here.

1. How would you extend the application so it displayed the name of the user instead of the ownerUserId (in red)
   1. Run node cassandra-users.js in the data import directory to create user table and import user data.
   2. Cassandra does not allow you to join tables. We learnt at the beginning of this exercise that reads are expensive and writes are cheap. De-normalizing data is acceptable as disk is cheap. Keeping this in mind how would you change the column families to display the username?
   3. Update post & response column families based on the answer in 6.2
   4. Inspect & run the cassandra-post-user.js script to update post. Query the count of the number of posts & responses before and after the script finishes and past it here. Notice how the data operations are idempotent. No matter how many times you insert the data, because of how the primary key is modeled, data is overwritten seamlessly.
   5. Update the node application to replace [USER\_NAME\_COLUMN] with the name of the column that you added in the post & response column families. (You will need to update app.js, index-user.js & details-user.js.)
      1. Copy and paste a screenshot for <http://127.0.0.1:3000/user/> here and then continue.
      2. Click into a post and copy and paste a screenshot for [http://127.0.0.1:3000/[POST\_ID]](http://127.0.0.1:3000/14) here.

## Installation Notes

Cassandra will be installed as a Windows Service and always running in the background. To stop it from starting automatically (but not uninstalling) you can disable Automatic Startup in Services Management. Launch it by starting services.msc and look for DataStax Services:



Just change the Startup Type to Manual ☺. Services to look for:

* + DataStax\_Cassandra\_Community\_Server
  + DataStax\_OpsCenter\_Agent
  + DataStax\_OpsCenter\_Community