GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

# ECE 4813A Fall 2017

# Lab-8: Real-time Data Analytics with Storm & Kafka

References:

[1] A. Bahga, V. Madisetti, “Cloud Computing: A Hands-On Approach”, ISBN: 978-1494435141

[2] <http://ambari.apache.org/>

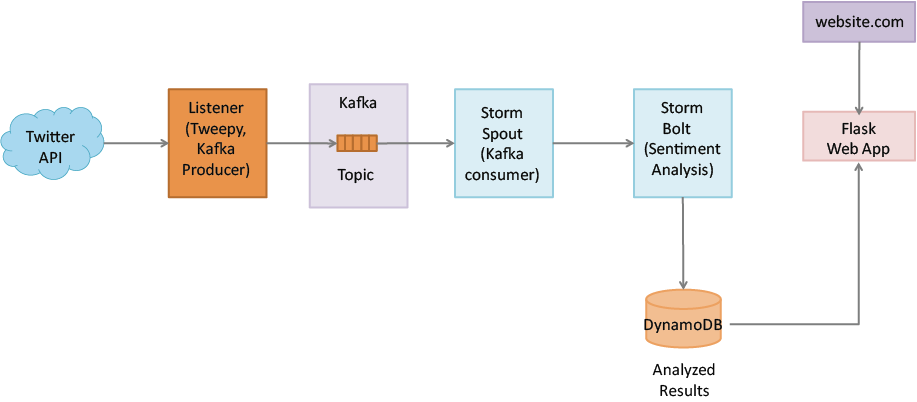
[3] <https://storm.apache.org/>

[4] <http://kafka.apache.org/>  
[5] <https://github.com/tweepy/tweepy>

Due Date: The lab report will be due on November 20, 2017.

# Introduction

In this lab you will be building a system for real-time sentiment analysis of twitter feeds using Apache Storm and Apache Kafka. Reference code for each of the components is provided. You have to build upon the reference code and integrate various components to have a working system.



Components to implement

* Listener: The listener component connects to Twitter with the streaming API and retrieves tweets in real-time. A Python library called *tweepy* is used for this. The listener will retrieve the tweets and publish them to a Kafka topic managed by a Kafka Broker.
* Storm Spout: Storm Spout contains a Kafka consumer which retrieves the tweets from the Kafka topic and emits tuples (containing tweets) to be processed by the Storm Bolt
* Storm Bolt: Storm Bolt analyzes the tweets and computes their sentiment using the AFINN sentiment lexicon.
* DynamoDB: The timestamped tweets and their sentiments are stored by the Storm Bolt to a DynamoDB table.
* Flask Web App: The Flask web application retrieves the tweets and their sentiments from DynamoDB and displays them.

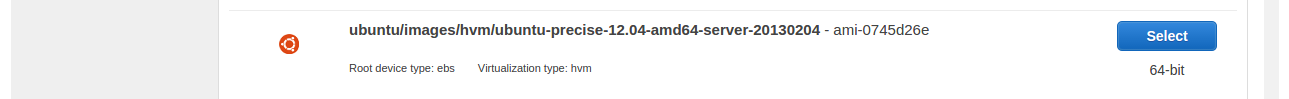
For each of the following components reference code has been provided

* Listener: listener\_tweepy\_example.py (contains code for fetching tweets)
* Kafka Producer: kafkaproducer\_example.py (contains code for publishing data to Kafka topic)
* Kafka Consumer: kafkaconsumer\_example.py (contains code for retrieving data from Kafka topic)
* Sentiment Analysis: compute\_sentiment\_example.py (contains code for computing sentiment. Uses a lexicon AFINN-111.txt and a sample tweet in rawTweet.txt)
* Storm Spout: storm\_spout\_example.py (reference code for Storm Spout)
* Storm Bolt: storm\_bolt\_example.py (reference code for Storm Bolt)
* Storm Project: contains files for creating a storm topology
* DynamoDB: dynamodb\_boto\_example.py (reference code for writing to and reading from DynamoDB)
* Flask: webapp\_example.py (reference code for a Flask webapp)

Setting up Storm and Kafka cluster

We will use Apache Ambari for provisioning, managing and monitoring a cluster that runs Storm and Kafka on top of YARN resource manager and uses HDFS as distributed filesystem and Zookeeper for coordination.

Launch an m3.xlarge instance with Ubuntu 12.04 (not 14.04) image below:

(The lab has been tested with the above image. Any other version of Ubuntu may or may not work.)  
  
Open all traffic from everywhere in the security groups.   
  
Connect to the instance by ssh and run the following commands:

|  |
| --- |
| sudo apt-get -q -y update  sudo ufw disable  sudo apt-get -q -y install ntp  sudo service ntp start  sudo wget http://public-repo-1.hortonworks.com/ambari/ubuntu12/1.x/updates/1.7.0/ambari.list  sudo cp ambari.list /etc/apt/sources.list.d/  sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com B9733A7A07513CAD  sudo apt-get -q -y update  sudo apt-get -q -y install ambari-server  sudo apt-get -q -y install ant gcc g++ libkrb5-dev libmysqlclient-dev libssl-dev libsasl2-dev libsasl2-modules-gssapi-mit libsqlite3-dev libtidy-0.99-0 libxml2-dev libxslt-dev python-dev python-simplejson python-setuptools maven libldap2-dev python2.7-dev make python-pip  sudo pip install kafka-python  sudo pip install tweepy |

On your local machine run the following command to copy the keypair to the EC2 instance (change IP to the public IP of your instance):

scp -i myKeyPair myKeyPair.pem ubuntu@52.1.101.179:/home/ubuntu/.ssh/

On your EC2 instance change the name of the keypair you just copied from your local machine to id\_rsa:

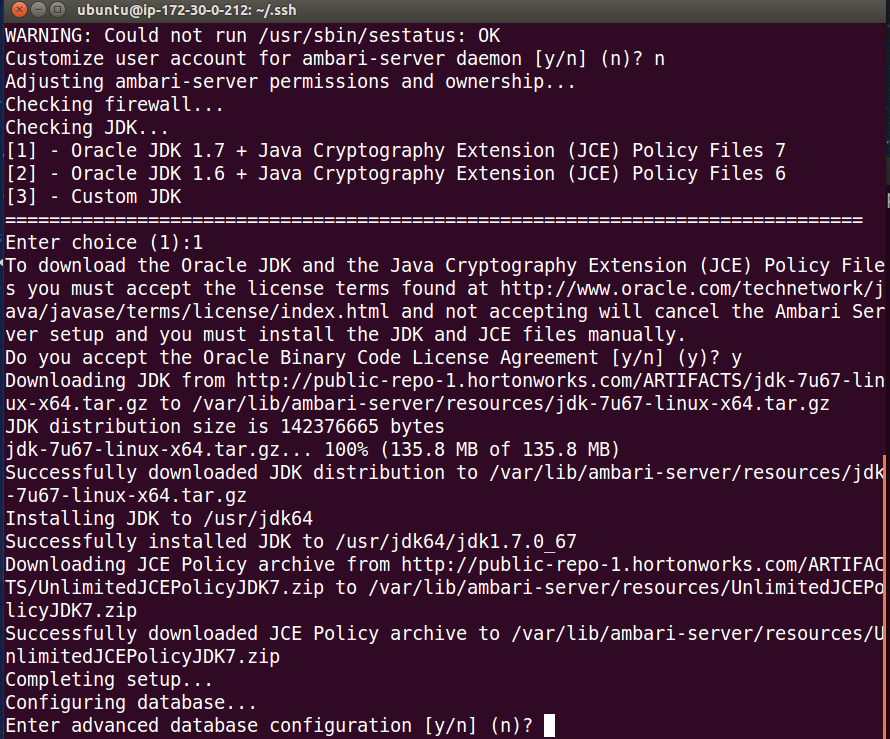
cd /home/ubuntu/.ssh/

mv myKeyPair.pem id\_rsa

Run the following command to setup Apache Ambari:

sudo ambari-server setup

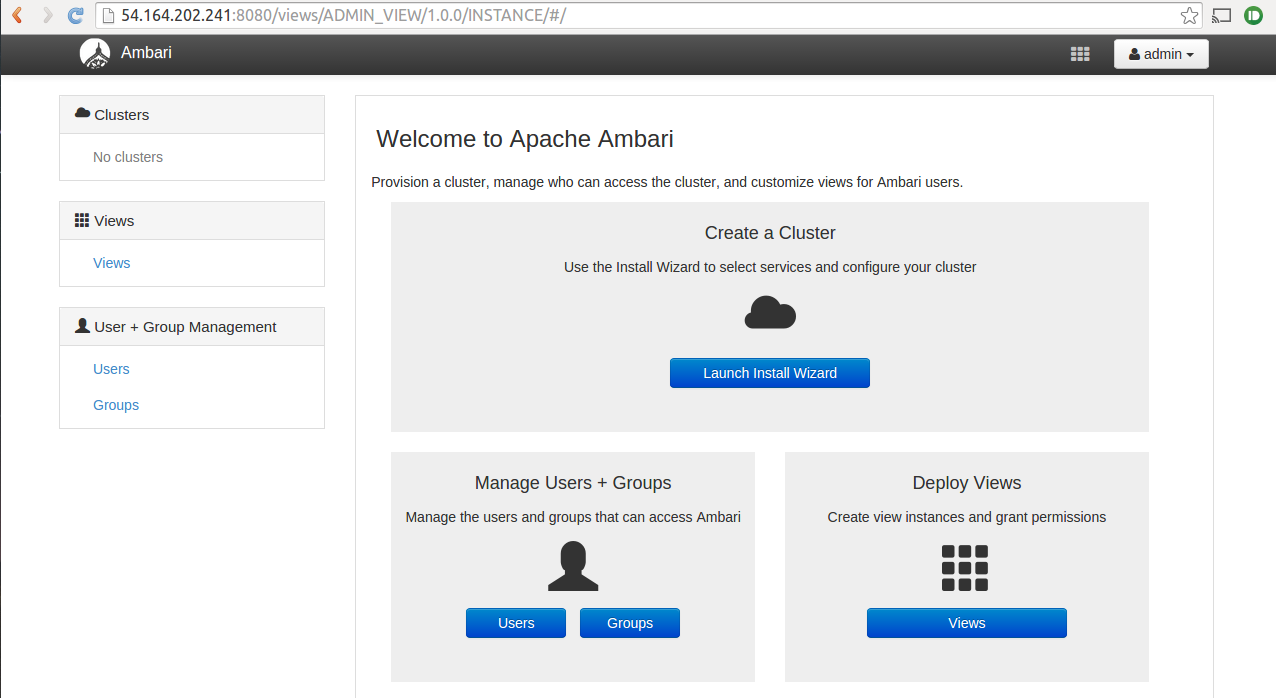
Choose default options in the setup as shown below:

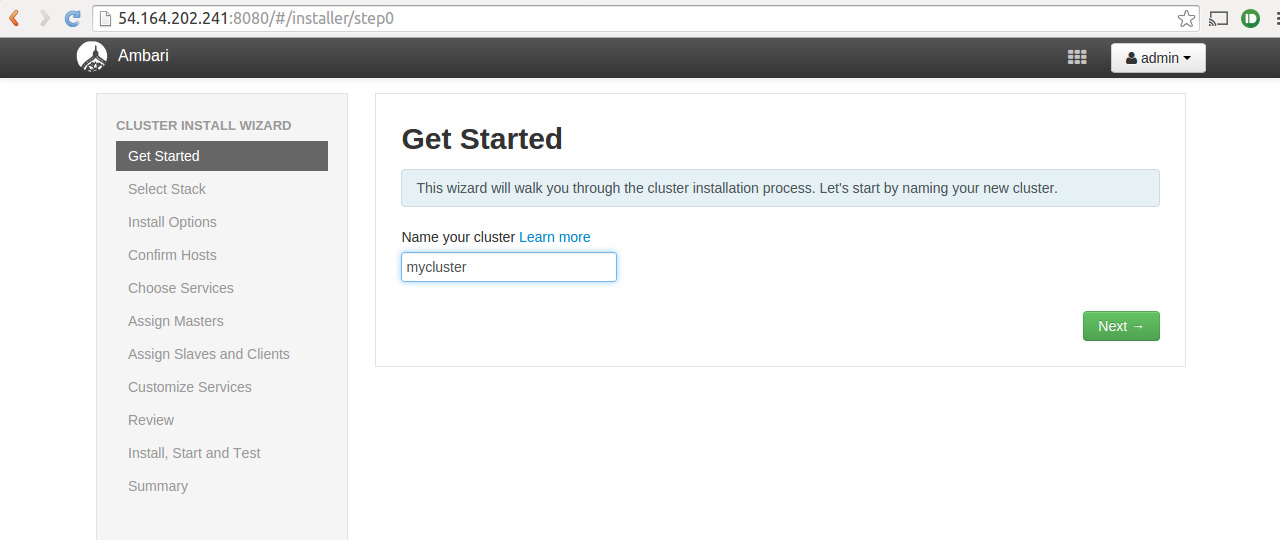
Start Apache Ambari:  
  
sudo ambari-server start

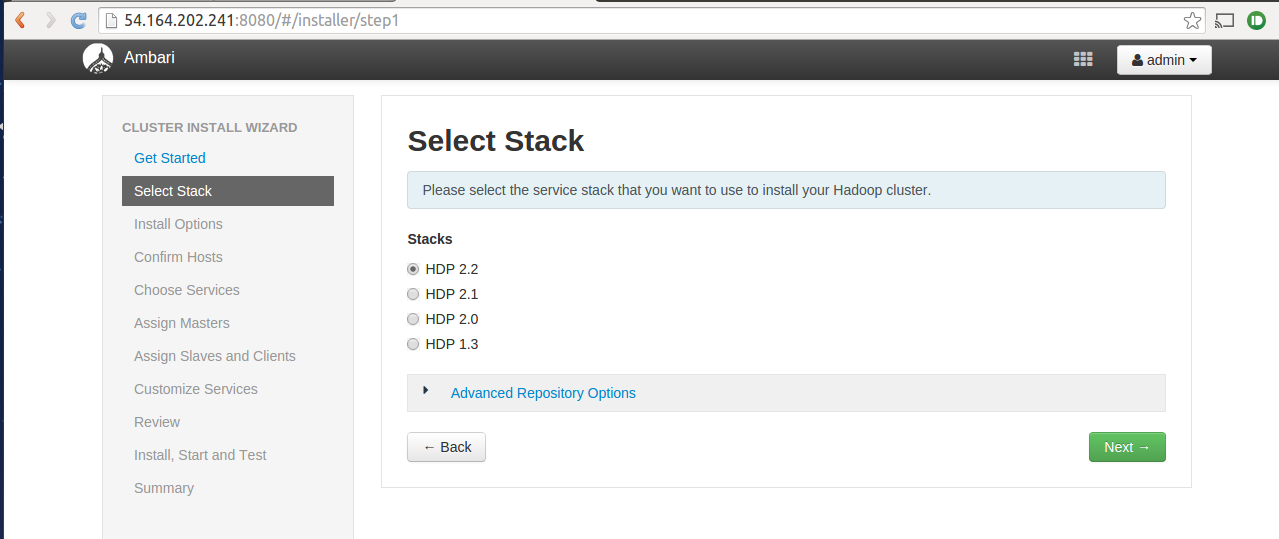
Open the link http://<public-ip>:8080

Login into Ambari Server with username admin and password also admin

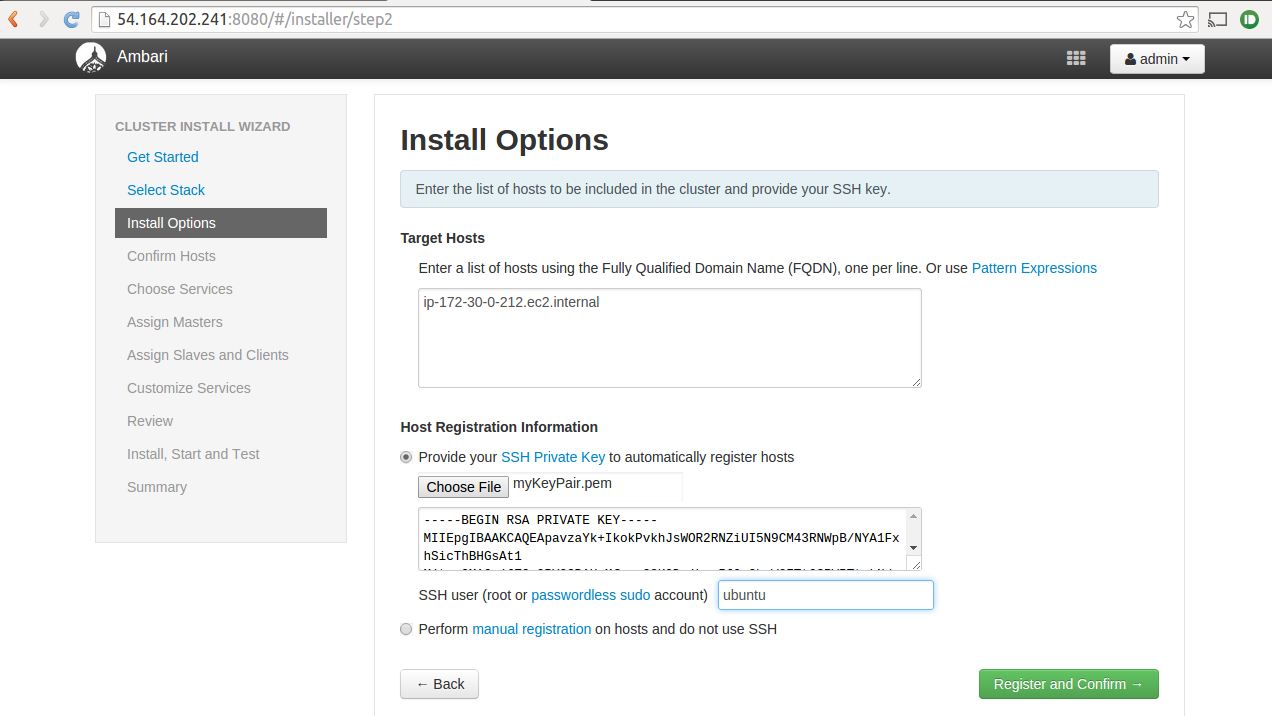
Follow the wizard to create a cluster as shown in the screenshots below:

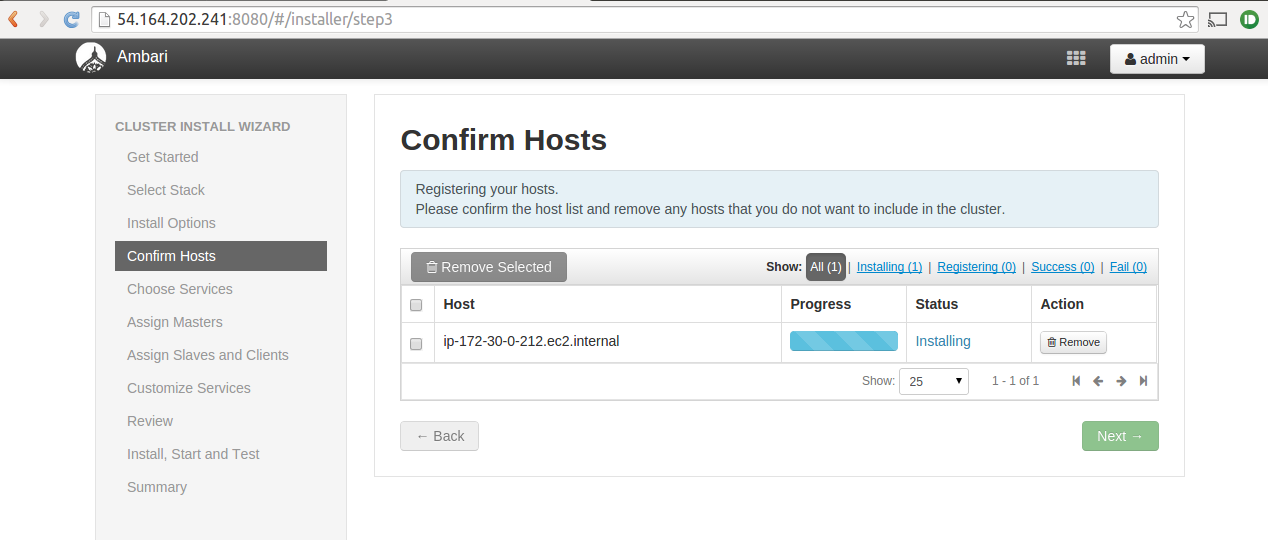


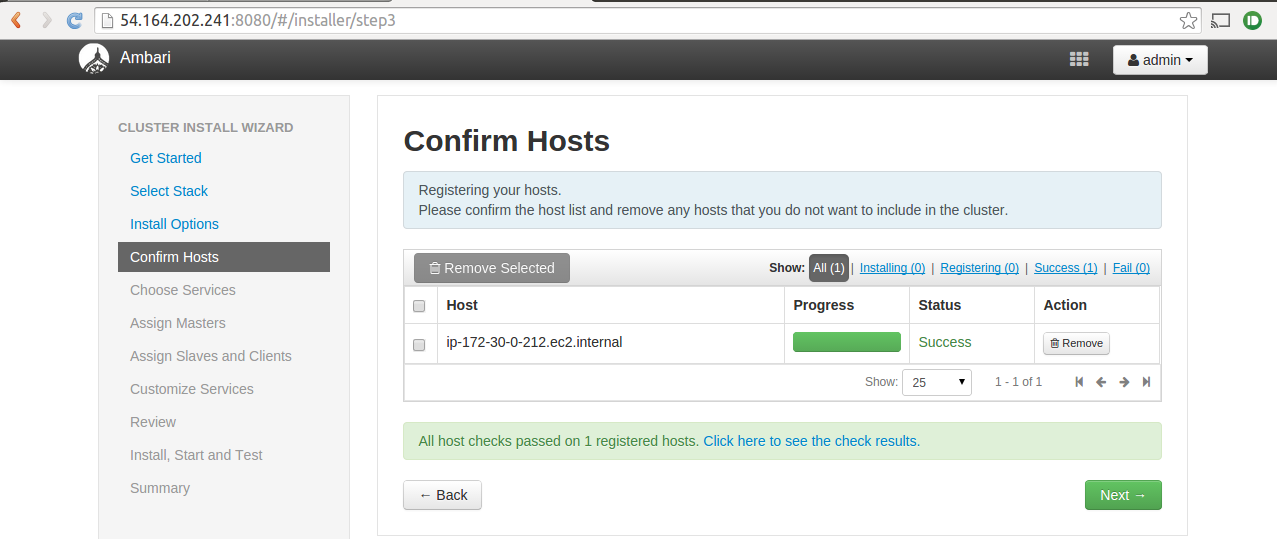




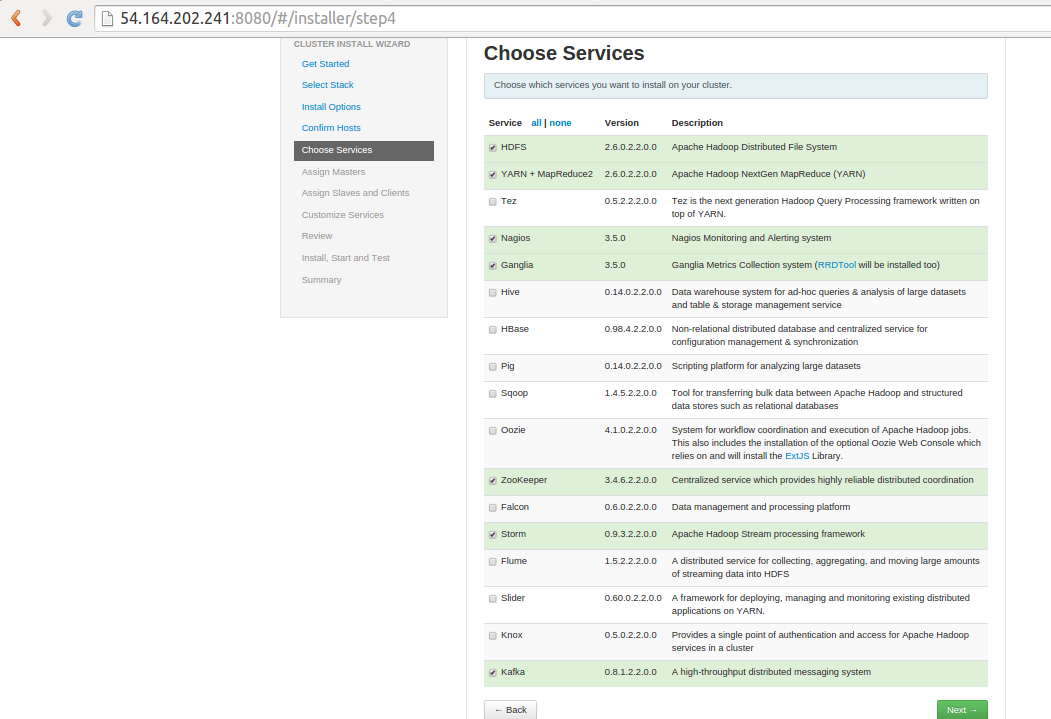
Enter the private DNS of the instance in the Target Hosts section. Select the EC2 key-pair file associated with the instance and change SSH user to ubuntu as shown below:

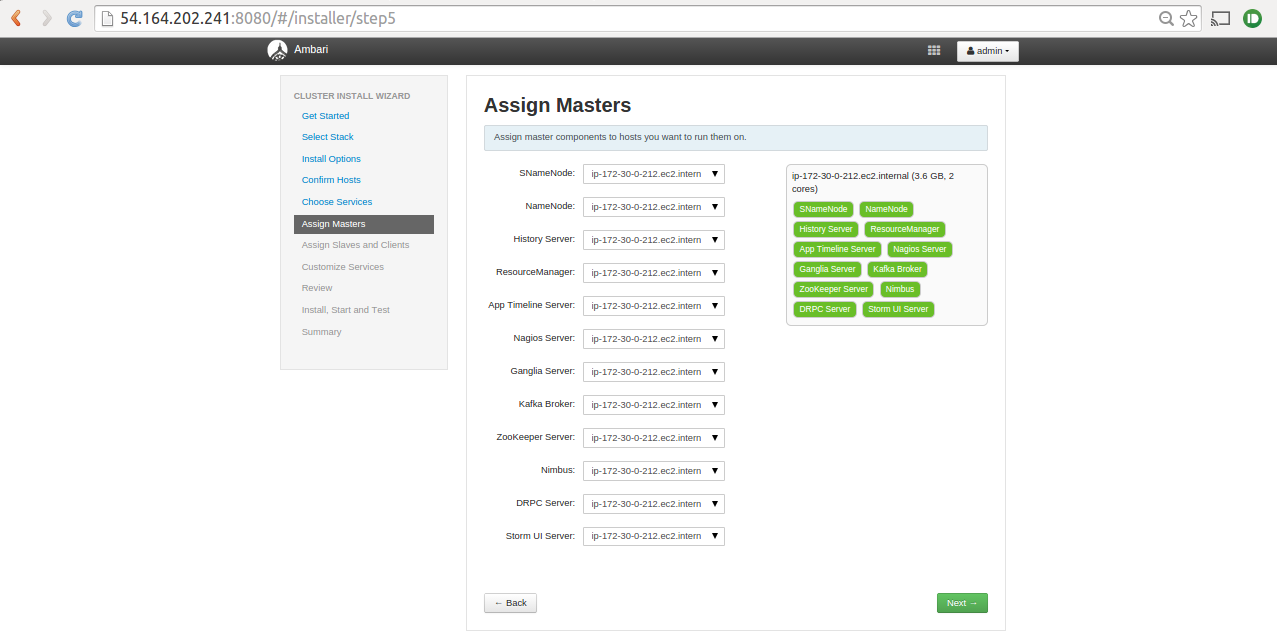


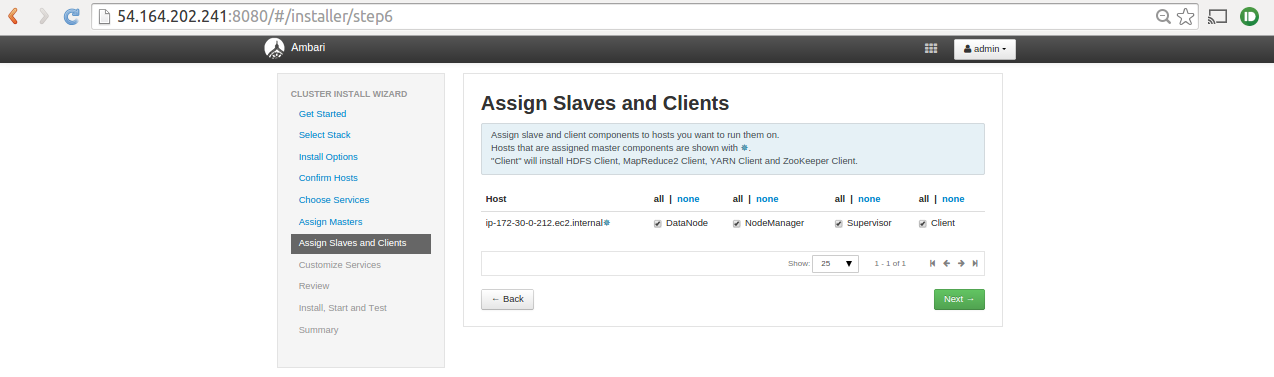


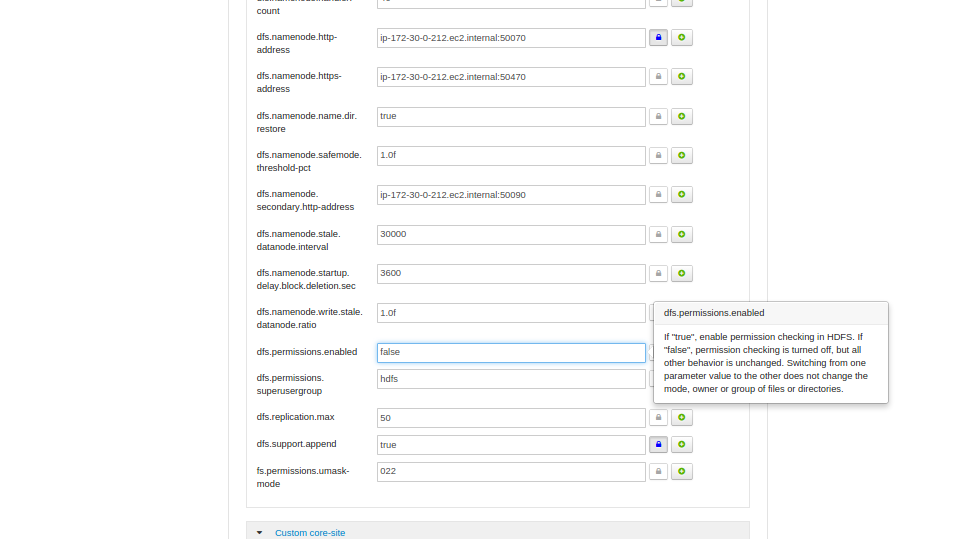


Select the following services:

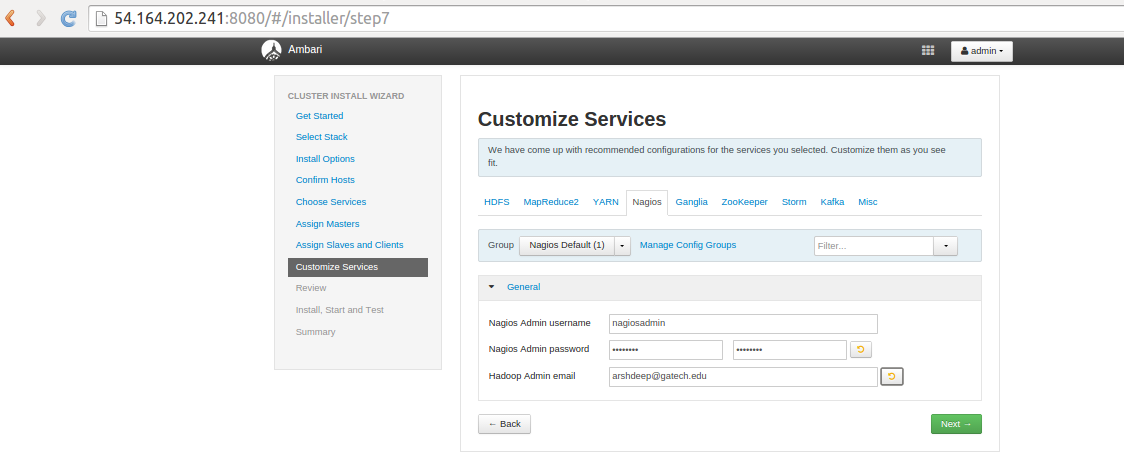


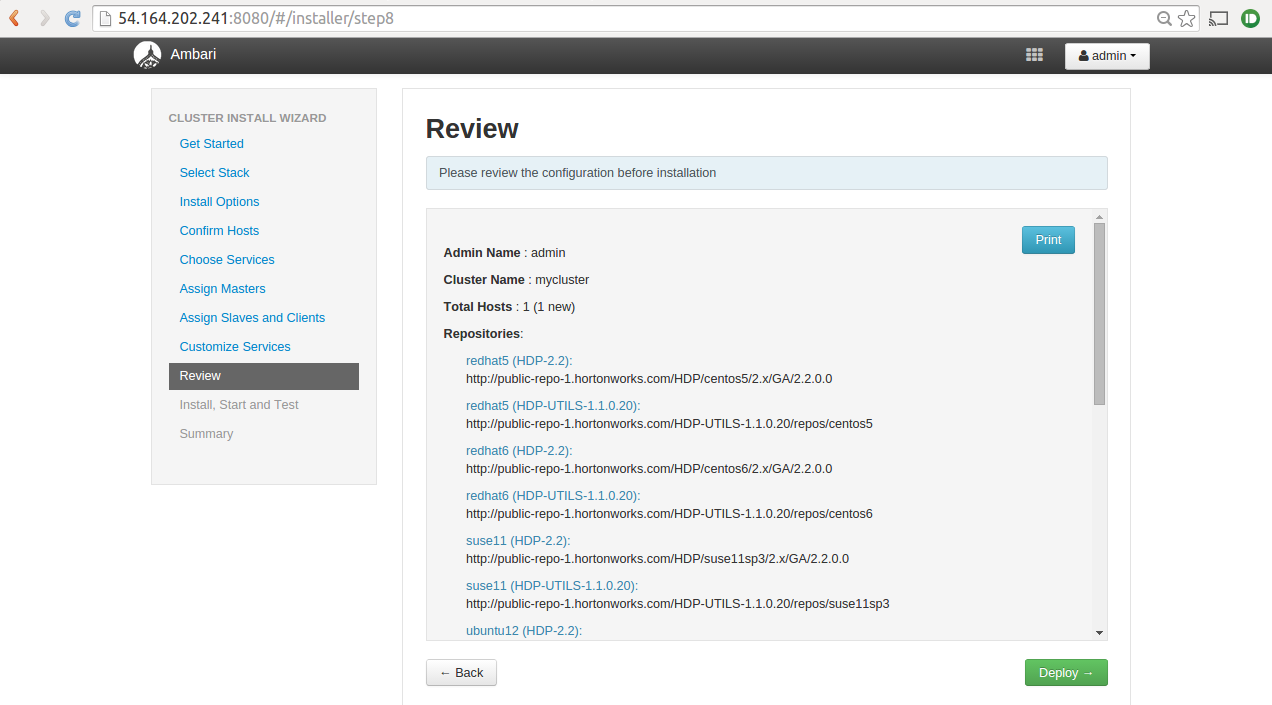


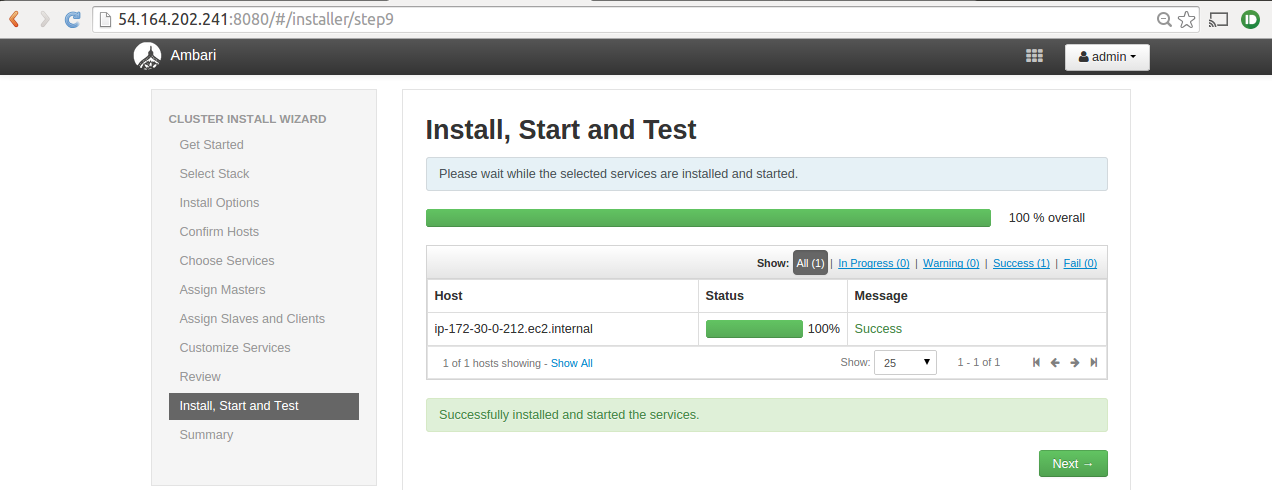
On the configurations page, change the HDFS property (dfs.permissions.enabled) to False.



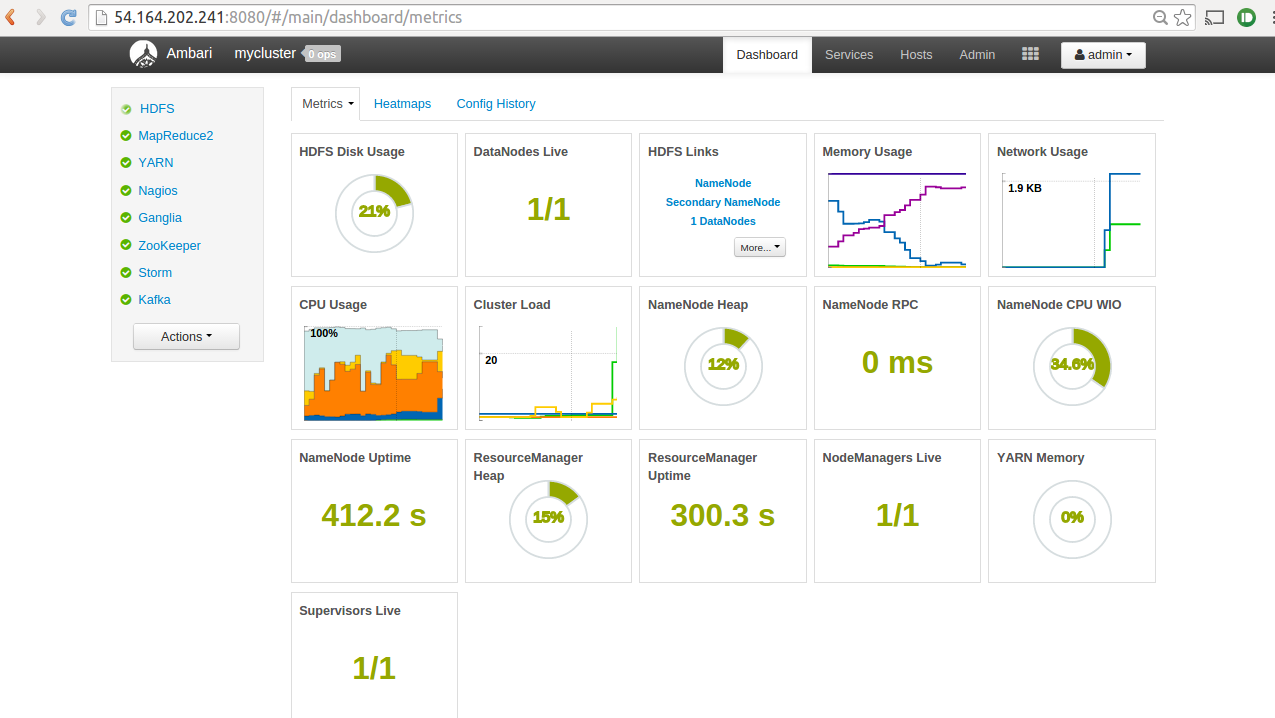
In the Nagios tab, enter some password and email address to use.

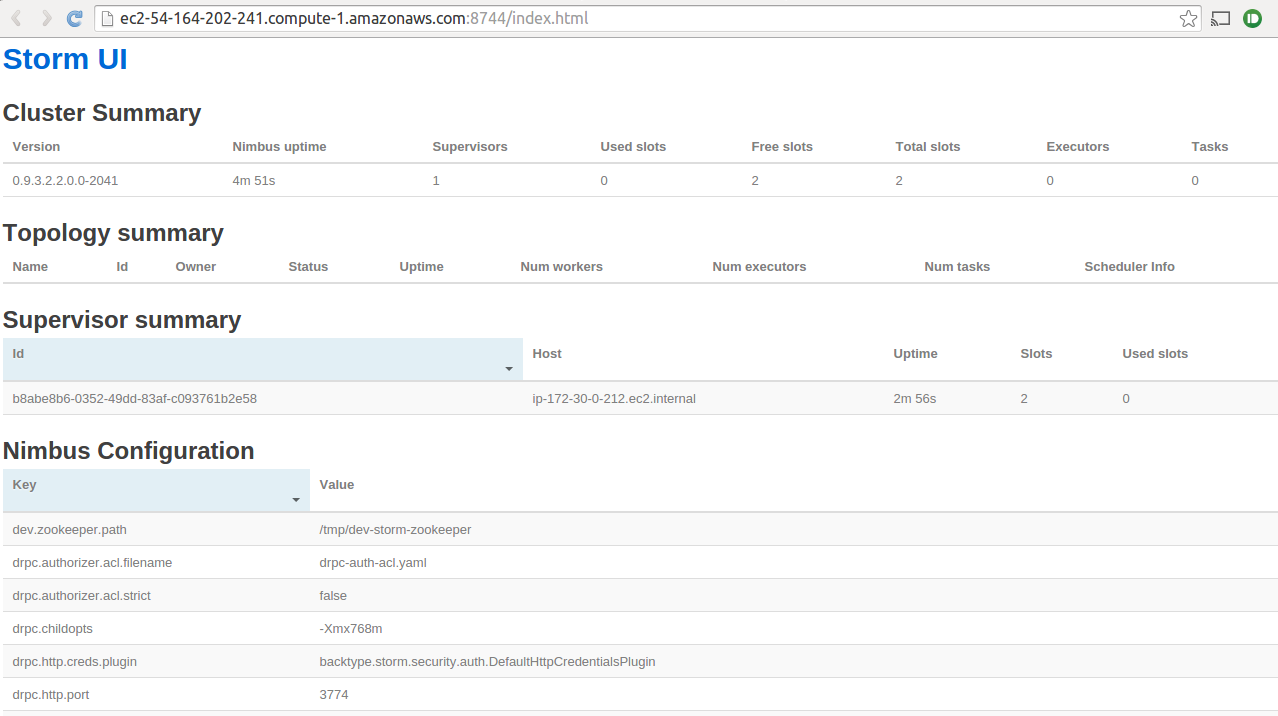


  
Complete the wizard to setup the cluster.

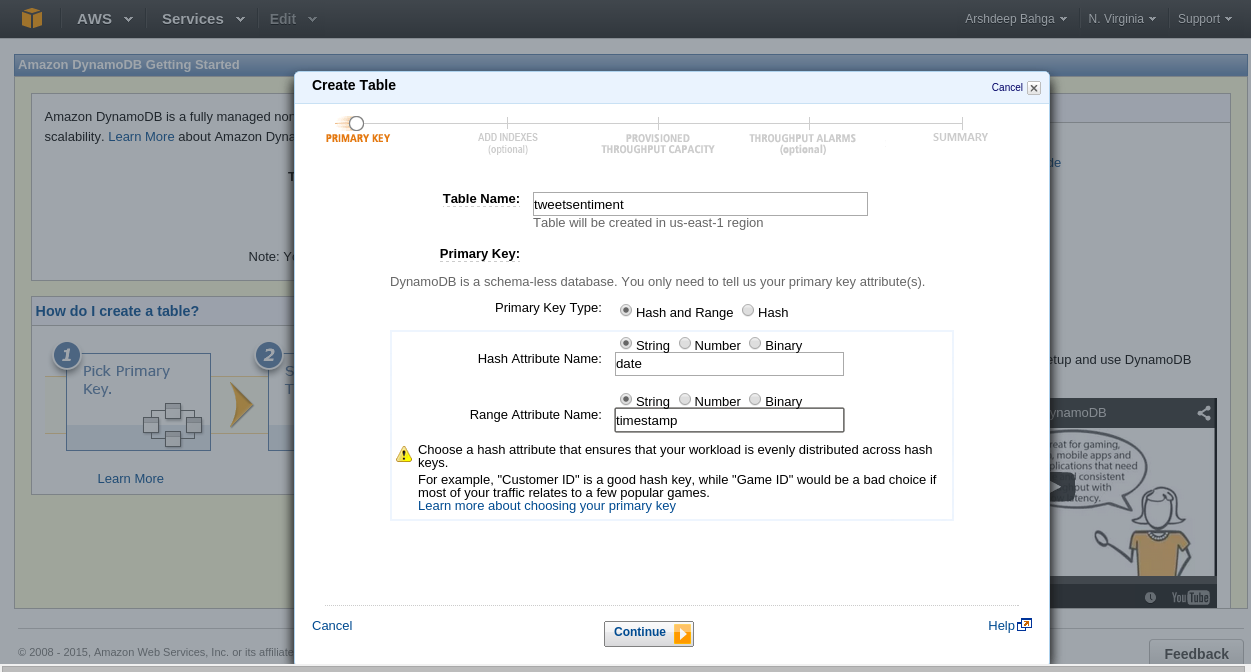


Finally, you will have HDFS, MapReduce, YARN, Nagios, Ganglia, Zookeeper, Storm and Kafka setup. You can monitor the services from the dashboard. Make sure all services are running. You can also restart services from the dashboard.

Storm UI can be accessed at http://<public-ip>:8744.



Create DynamoDB table



Create Kafka topic

Run the following commands on your EC2 instance:

cd /usr/hdp/2.2.0.0-2041/kafka

bin/kafka-topics.sh --create --zookeeper ip-10-179-181-24.ec2.internal:2181 --replication-factor 1 --partitions 1 --topic forestfire

bin/kafka-topics.sh --list --zookeeper ip-10-179-181-24.ec2.internal:2181

(Change the private DNS to in the aboce commands to the private DNS of your EC2 instance)

Build Storm Project

After implementing the Storm Spout and Bolt, use the storm-project folder attached with the lab to build a Storm project. Copy your Storm Spout and Bolt files to the

storm-project/multilang/resources folder.

The topology (called mytopology) is defined in

storm-project/src/main/java/com/mycompany/app/App.java file.

The topology includes one Spout and Bolt with field grouping between them. You don't need to edit this file. Just make sure the Spout and Bolt names in App.java match the Python filenames.

Run the following commands on your EC2 instance:

cd /home/ubuntu/storm-project

mvn clean package

Now you will have the Storm project JAR file (storm-project-0.9.3-jar-with-dependencies.jar) within storm-project/target folder.

Submit Storm Topology

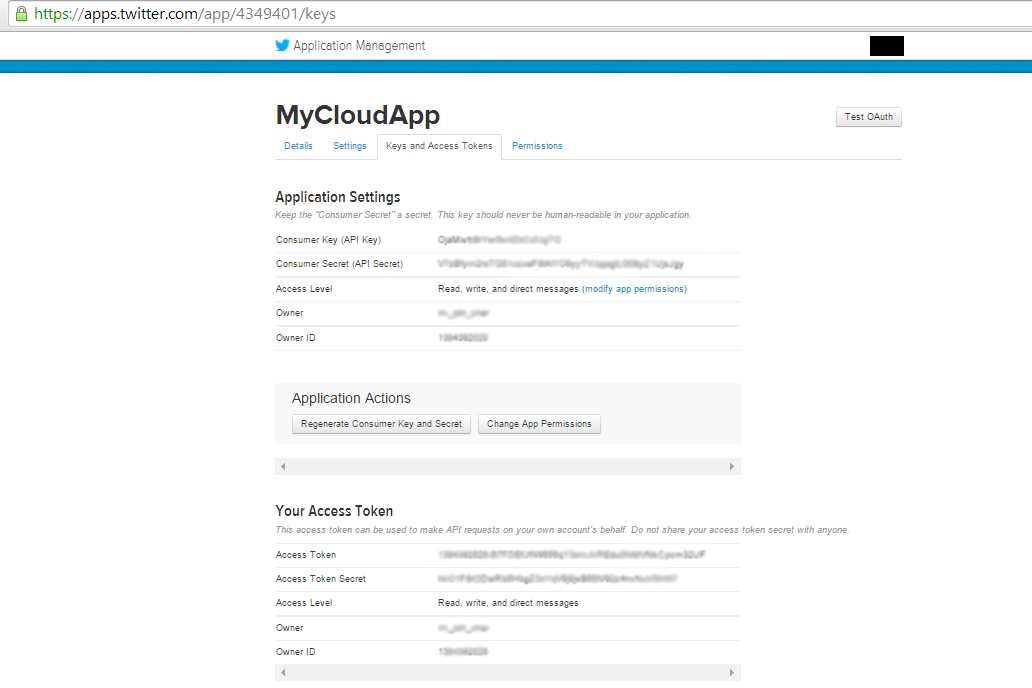
Run the following command on your EC2 instance:

storm jar /home/ubuntu/storm-project/target/storm-project-0.9.3-jar-with-dependencies.jar com.mycompany.app.App mytopology

Getting Twitter App Keys

The listener requires Twitter application keys. You will have to register for a Twitter developer account and create a new application at: <https://dev.twitter.com/apps>

Note down your access\_token, access\_token\_secret, consumer\_key and consumer\_secret from the application details page as shown below and add them to your listener Python code.



Checklist for Integrating Components

* Setup Storm and Kafka Cluster (including other dependencies) using Ambari
* Create Kafka topic
* Create DynamoDB table from AWS dashboard
* Implement Storm Spout and Bolt
* Update Kafka topic endpoint in Storm Spout
* Update DynamoDB table name in Storm Bolt
* Build Storm project
* Implement the listener
* Update the Kafka topic endpoint in listener
* Run listener
* Submit Storm topology
* Check DynamoDB dashboard for results
* Check Storm UI for count of tuples processed by Storm topology
* Run Flask WebApp

Grading Rubric

* Implementation/source code of all components - 40
* Screenshot of Storm topology pages (topology home, spout and bolt pages showing count of emitted and consumed tuples) - 20
* Screenshot of DynamoDB table (in AWS dashboard) showing tweets and sentiments - 20
* Screenshot of webapp showing analyzed tweets - 20