Principles of Cloud Computing

Programming Assignment #1

**On-ground:** Handed out 9/8/2020; Due 9/24/2020 (11:59 CDT)

**2U:** Handed out Week3; Due Week 5 (submit by Friday 11:59 CDT)

# **Manual Deployment of Simple Data Analytics Pipeline in the Cloud using Virtual Machines**

The goal of this assignment is to deploy a data analytics pipeline in the cloud using a cluster of distributed virtual machines. The data analytics pipeline for now will comprise a Kafka messaging system and a database serving as a sink. No analytics as such will be performed in this assignment. You can try your solution in Chameleon. Once you are certain that everything is working, you can move your solution to AWS. For the CS4287/5287 on-ground section, you will demo the solution to the TA via Zoom. For the 2U section of this class, teams will grade the solutions and send me grading notes. Rubrics will be provided.

The following figure illustrates the expected deployment.

Laptop

VM1.n in VirtualBox

WS

VM2

VM3

AWS/Chameleon Cloud

Laptop

VM1.1 in VirtualBox

utilizations1

utilizationsN

**Configuration**

VM1 will be a VirtualBox VM that you create manually (Ubuntu 18.04 or 20.04). Since you are working in teams, each one of you will have a laptop that serves as a client. So, we could technically have multiple clients as shown in the figure. I call these VM1.1 thru VM1.n (typically n = 2 or 3 depending on team size). Each client VM1.x produces a separate topic, say from utilizations1 thru utilizationsN. The consumer subscribes to all these topics. VM2 and VM3 will be virtual machines created in the cloud. For Chameleon Cloud, these could be m1.small flavor

* VM1 will run the data generator (producer of the events. See producer.py scaffolding code).
* VM2 will run Kafka’s ZooKeeper server, and one messaging broker
* VM3 will run Kafka’s second messaging broker (here we are using replication of 2; typically 3 or more is needed but here we will just go with 2), a consumer code subscribing to all topics, and the database sink in which the consumer dumps all the incoming data.

**Expected Data Flow**

The producer program provided in the scaffolding code invokes the “top” command in Linux and sends the output of this command to a Kafka Broker. The IP Addresses in the scaffolding code are all hard-coded and so you will need to make appropriate changes. The producer runs this code in a loop for 100 iterations with a 1 second sleep per iteration. The producer code uses the Kafka Python streams API to encode and send the data to Kafka. Data is currently sent as a set of bytes and not in JSON or any such format. You will need to format it properly as JSON and send it to the Kafka broker. Also, include the timestamp when the data is produced and include it in the topic contents. This is easy to do in JSON as it will be one more key-value pair. Note also that the producer in its current form produces only one topic. But with multiple client machines, each client VM will produce its own topic and submit to the system.

A consumer program is provided in the scaffolding code, which uses the Kafka Python streams API to read the content of the incoming topic from the Kafka broker and simply prints the value field in ASCII. This code will need to be modified to dump the incoming contents into a Standalone CouchDB installation using JSON format. One could potentially use Kafka connectors to directly talk to CouchDB, however, we will do this via a consumer program. In later assignments we will include Spark as part of the pipeline. Also note that right now the consumer subscribes to only one topic but in your case it will be more than one topic: one per client VM.

**Technologies used:**

* Chameleon for testing; AWS for production quality code and demo.
* Ubuntu Linux version 18.04 (or 20.04) for the platform
* Python3 for the programming language
* Apache Kafka for messaging
* Apache CouchDB for data sink (standalone mode)
* Kafka-Python and CouchDB-Python libraries

**Why these technologies?**

* Python3 is most heavily used language and is the fastest way to put together code as it is interpreted.
* Ubuntu is usually used for cutting edge Linux capabilities and heavily used in education
* Kafka is very widely used in industry (see their web page for the many different companies using Kafka)
* CouchDB is not as popular as MongoDB or Cassandra or MySQL, however, it appears to be a simpler database to use AND it is designed using REST principles, which helps us understand REST through a real-world example, and adopts a number of Cloud-based principles like autoscaling and replication.

**Data collection**

As you are doing this assignment, although I have provided scaffolding code and slides to help you get started, there will still be enough learning curve, and trial/error. Please keep track of the manual effort expended in all the steps as part of this assignment. In a later assignment, we want to compare how much effort was saved through automation.

**Rubrics**

* Correctness (program works): 40%
* README file explaining how to run the code: 10%
* Teamwork (who did what and how much): 20%
* Report on effort expended (learning curve, etc): 20%
* Zoom-demo to TA or to the grading team: 10%

For this assignment, there may not be so much of a need to maintain a github repo other than the producer and consumer code, but maybe needed in subsequent assignments.