Assignment1

Submit Assignment

Due Tuesday by 12:01pm **Points** 100 **Submitting** a file upload
Available Sep 3 at 12am - Sep 12 at 12pm 10 days

Assignment 1: Environment Setup [1 week]

Due: Sep 10, Tuesday, noon EST

Submission requirements - common to all assignments

- Solution Document - a Word or PDF file, named <LastName>_<FirstName>_HW1.docx[pdf] (for example: Popova_Marina_HW1.docx) based on the Submission template: **Assignment01_SubmissionTemplate.docx**
- Full source code (no compiled classes, or generated by IDEs artifacts like .project etc.) - submitted as either separate files (say, hw1_problem1.py or Problem1.java) or archived into a zip/tar archive; archive should be named: <LastName>_<FirstName>_HW<number>.zip[tar]
- Result files (if any) - can be placed into the same archive with the source code

Objectives

The goal of this assignment is to practice setting up virtual machines and container to run various software. We will be setting up various machines and containers with various applications throughout the semester. We want to use a disposable machine that we can afford to experiment with and make mistakes on. We would not want to use a personal computer or laptop which could potentially be corrupted with the various installations and configuration settings we will be making.

For virtual machines, it is recommended that you use an AWS account for the following reasons:

- Amazon provides us with student credits which can be used to do some of our work for free
- The teaching staff will be able to respond to questions about using AWS
- It is a powerful platform which is often used in real-world projects

You may choose to use a different cloud service (Google, Microsoft, Oracle, etc), but the teaching staff will not be able to support you if you encounter issues.

For containers - we will be using Docker

Problem 1: [25 points] (slightly modified Problem 1a from the Assignment 0)

- Write a program (Java, Python or Scala) that takes the following input parameters:

- number of files to generate - "numFiles"
- number of lines per file to generate - "numLines"

- and does the following:
    - generate a file with the specified number of lines:
    - each line should have 3 random numbers in the range [0-10]; your lines would look like:
        - *"1 7 3"*
        - *" 2 1 3"*
    - each file has to be generated by a separate thread in your program - the easiest way is to just create/start as many threads as "numFiles"
    - File names has to be in the format: <yourFirstName>_<yourLastName>_threadNumber.txt
    - For example, if 'numFiles' = 3, the following files should be generated:
        - marina_popova_0.txt
        - marina_popova_1.txt
        - marina_popova_2.txt

Problem 2: [25 points] Set up a machine and demonstrate that it works

- Set up an AWS account
- For this assignment we will provide you the preconfigured CentOS AMI with Java 8 and Python installed. You should be able to locate it under public AMIs under the name “Harvard-e88-HW1” in the region '*US EAST Ohio'* . If you wish to hand build your own machine you may, and it is a very good exercise.
- Launch the machine, and log in to it
- Transfer your Problem1 program to this machine and demonstrate that it can execute successfully

Refer [**https://docs.docker.com/get-started/** **(https://docs.docker.com/get-started/)**] for information on Docker

Problem 3: [25 points] Run Redis server and clients as Docker containers and demonstrate that they work

- Using provided Docker image [“ https://hub.docker.com/_/redis/ **(https://hub.docker.com/_/redis/)** “] - Create and start a Redis server
- Start two more Docker containers with Redis clients
- In the Redis client1: connect to the server and create a variable named "x" with value 10
- In the Redis client2: get the value of 'x', confirm it is 10, set it to 20
- In the Redis client1: get the value of 'x' - verify it is 20 now

Refer https://redis.io/topics/introduction **(https://redis.io/topics/introduction)** for information on Redis

Problem 4: [25 points] Run Postgres DB as a Docker container and demonstrate that it works

- Using provided Docker image [“ https://hub.docker.com/_/postgres/ (https://hub.docker.com/_/postgres/) ”] - create and start a PostgresDB server
- Using psql command shell or DBeaver GUI client, connect to the DB and:
- In a 'public' schema, create a table named "<your_last_name>_data" that contains the following:
  - an ID of type integer that cannot be null, and must increment automatically
  - a name that is a character string and cannot be null
  - a creation date that is of type date
  - a primary key (what should your primary key be?)
- Insert three sets of dummy records into your table
- Query your database for all records
- Delete all records

Refer https://www.postgresql.org/about/ (https://www.postgresql.org/about/) for information on Postgres

Problem 5 [Bonus, 15 points]: Start multiple Docker containers via Compose

- Create a Docker compose configuration to start all above servers together - Redis server, 2 Redis clients, Postgres server
- Verify they are all functional (connect to each and verify connectivity to the DBs)
- Refer [https://docs.docker.com/compose/ (https://docs.docker.com/compose/)] for information on Docker Compose