DAY-9

TOPIC: Docker

1. Scenario: You are building a microservices-based application using Docker. Design a Docker Compose file that sets up three containers: a web server container, a database container, and a cache container. Ensure that the containers can communicate with each other properly.
2. Scenario: You want to scale your Docker containers dynamically based on the incoming traffic. Write a Python script that utilizes Docker SDK to monitor the CPU usage of a container and automatically scales the number of replicas based on a threshold.
3. Scenario: You have a Docker image stored on a private registry. Develop a script in Bash that authenticates with the registry, pulls the latest version of the image, and runs a container based on that image.

TOPIC: Airflow

1. Scenario: You have a data pipeline that requires executing a shell command as part of a task. Create an Airflow DAG that includes a BashOperator to execute a specific shell command.
2. Scenario: You want to create dynamic tasks in Airflow based on a list of inputs. Design an Airflow DAG that generates tasks dynamically using PythonOperator, where each task processes an element from the input list.
3. Scenario: You need to set up a complex task dependency in Airflow, where Task B should start only if Task A has successfully completed. Implement this dependency using the "TriggerDagRunOperator" in Airflow.

TOPIC: Sqoop

1. Scenario: You want to import data from an Oracle database into Hadoop using Sqoop, but you only need to import specific columns from a specific table. Write a Sqoop command that performs the import, including the necessary arguments for column selection and table mapping.
2. Scenario: You have a requirement to perform an incremental import of data from a MySQL database into Hadoop using Sqoop. Design a Sqoop command that imports only the new or updated records since the last import.

3. Scenario: You need to export data from Hadoop to a Microsoft SQL Server database using Sqoop. Develop a Sqoop command that exports the data, considering factors like database connection details, table mapping, and appropriate data types.

Submission Guidelines:

1. Answer all the questions in a single Jupyter Notebook file (.ipynb).
2. Include necessary code, comments, and explanations to support your answers and implementation.
3. Ensure the notebook runs without errors and is well-organized.
4. Create a GitHub repository to host your assignment files.
5. Rename the Jupyter Notebook file using the format "date_month_topic.ipynb" (e.g., "12_July_Dockers.ipynb").
6. Place the Jupyter Notebook file in the repository.
7. Commit and push any additional files or resources required to run your code (if applicable) to the repository.
8. Ensure the repository is publicly accessible.
9. Submit the link to your GitHub repository as the assignment submission.

Grading Criteria:

1. Understanding and completeness of answers: 40%
2. Clarity and depth of explanations: 25%
3. Correct implementation and evaluation of matrix operations: 15%
4. Proper code implementation and organization: 10%
5. Overall presentation and adherence to guidelines: 10%

Note:- Create your assignment in Jupyter notebook and upload it to GitHub & share that uploaded assignment file link through your dashboard. Make sure the repository is public.