1. Write a Docker Script to perform the following actions.

* Download the Docker image for SQL Server Developer Edition
* Have a folder called DBScripts in the Docker root container where all the SQL Server Database scripts are present
* The .sql file in the DBScripts folder should run in Ascending order in the Database when the Docker image is built
* The SQL Server Developer edition should be exposed to the Docker host machine so that the Reporting tools / SSMS can connect to it from the host machine

please test with the following SQL scripts in the DBScripts folder.

* 01-create-database.sql

CREATE DATABASE testDB;

* 02-create-table.sql

USE DATABASE testDB;

CREATE TABLE Test

(

Id int,

Data varchar(50)

);

* 03-insert-data.sql

INSERT INTO Test (

Id, Data

)

VALUES (

1, 'A'

);

The expected outcome should be

* Docker container with the SQL Server Developer tools installed
* SQL Port exposed outside to the host image
* The Database and Table created with the Sample

For this question I created a docker-compose.yml file that

1. Installs mssql into the container with the image mcr.microsoft.com/mssql/server:2019-latest .

2.set the environment variable for sql like password.

3. set docker container to use root user.

4. copied sql scripts from local to docker container.

5. In bash mode started sql server and wait 30 seconds for it to start.

6. executed scripts in ascending order.

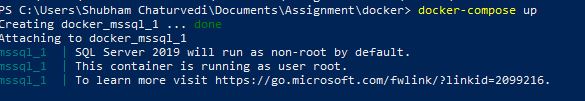
**How to execute the docker script:**

To execute the pipeline please navigate to the data\_pipeline directory to the Assignments/docker directory and execute the docker script as:

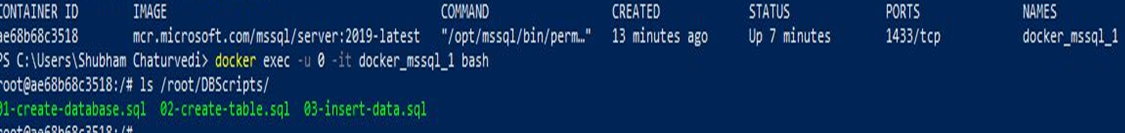
**docker-compose up**

**Outputs:**

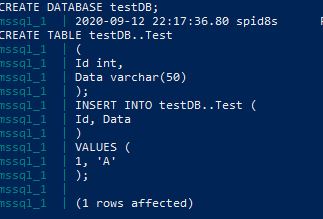
1. Sql tools installed.



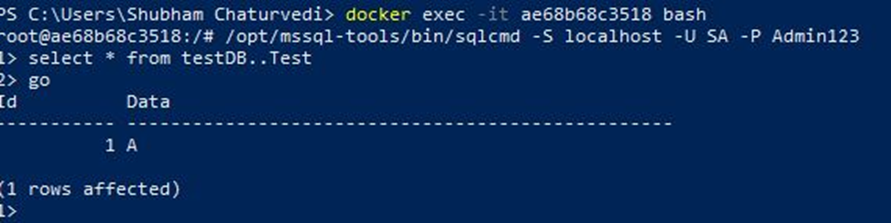
2. Scripts copied to DBScripts in the root of container.



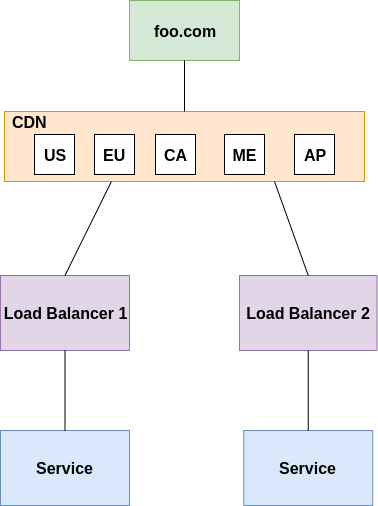
3. Database and table created and rows inserted.



4. Access sql server in bash mode from host and query database.



1. We have a high availability application running with the following Architecture.



* Explain the Architecture model of this and the Benefit
* The application is uses a content delivery network (CDN) + Load balancer architecture.
* Each of the CDN server holds a cached copy of the media and files that are required to display the website and rest of the application is hosted on a server cluster managed by a Software/Hardware load balancer.
* CDN-load balancer architecture is a great option to use for a production application to maintain continuaous uptime and optimize the experience of end-users in disparate geographic regions.

**CDN**

* CDN here is a network of servers spread across a wide geographic area. Each server within this CDN could host certain parts of the service and working together they can provide fast delivery of Internet content.
* CDN does not host content and can’t replace the need for proper web hosting.but it does hold cache content at the network edge, which improves website performance and reduces hosting bandwidth.
* The globally distributed nature of a CDN will reduce distance between users and website resources. Instead of having to connect to a website’s origin server which may live in a dirrerent geographical location, a CDN lets users connect to a geographically closer data center. Less travel time means faster service.

**Load Balancer:**

* The load balancer will distribute the application across the cluster of servers(all of which could be located in the same data center or multiple data centers), which is important for optimizing uptime and performance. The CDN will help optimize the experience of end-users in varied geographic regions.
* The load balancer ensures that requests for an application or data are spread evenly across the network of servers that hosts the application or data. Here for instance, we have 2 servers that each host an instance of an application, and to ensure high availability just one server can’t be handling all the requests as it can fail to respond when overwhelmed with traffic while the other sits idle. So a load balancer is used to distribute the work across both servers in order to avoid overburdening any of them.
* Load balancer will also ensure uptime by redirecting traffic to another server in the event that one server goes down.
* The Load balancers can be configured to operate on different network OSI “layers” depending on how traffic needs to be filtered. For example, the load balancer could operate on layer 4 and filtertraffic based based on IP addresses, or on layer 7 and filter based on HTTP header data. There are also different types of algorithms and strategies that load balancers can employ to decide which data is sent where.
* Load balancing Provides the flexibility to add or subtract servers as demand dictates.

**Benefits:**

**CDN:**

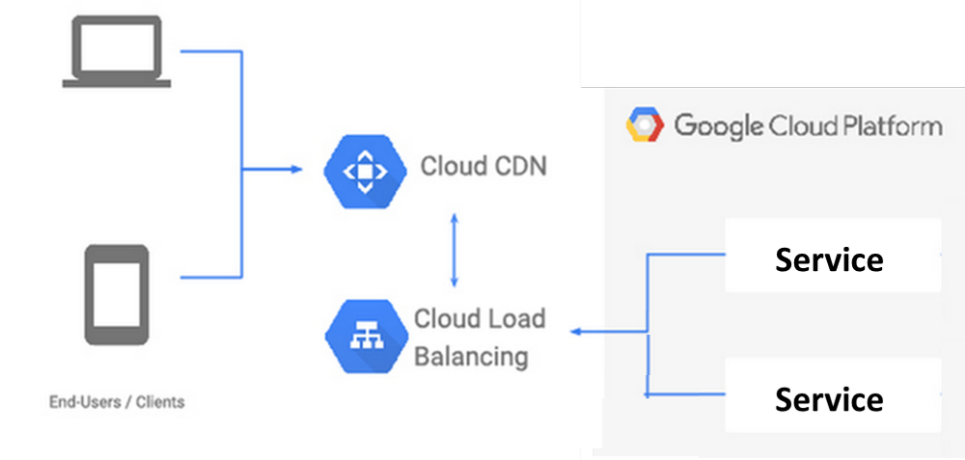
* **Faster response time:**A CDN allows for the quick transfer of assets needed for loading Internet content including HTML pages, javascript files, stylesheets, images, and videos
* **Uptime:** If one of the servers in your CDN goes down, other servers on the CDN will still be up and able to serve the content. A CDN isn’t the only way to maximize data availability, but it can help.
* **Access restrictions**:In certain cases, CDNs can help overcome access restrictions to content. For example, if a firewall blocks a client machine from accessing a server in one country, a CDN network can provide copies of the content in another country.
* **Security:** If properly configured CDN can help protect websites against some common malicious attacks.
* User monitoring to measure engagement

**Load balancer:**

* **Reduced Downtime:** If one server fails others can take over to maintain availability
* **Scalability**: On demand addition and subtraction of servers
* **Redundancy**
* **Flexibility** : Choose which L4 or L7 to filter requests on and which load balancing algorithm to use.
* **Efficiency:** Since load is evenly distributed over server cluster no single server is over burdened and hence application performs faster(in this case webpages load faster).
* **Global Server Load Balancing**: Extends L4 and L7 capabilities to servers in different geographic locations
* How do you roll out an update to the service without any downtime impact to the app foo.com
* To deploy an update to the service without any downtime impact to the app foo.com we can use Rolling deployment
* It is also also known as an incremental deployment, where new software is delivered, usually to one or more deployment targets at a time, until all of the targets have the updated version of the software rolled out.

Process will be something like this:

* With two servers running say version 1.0(v1.0 ) of the of the application, drain or take one server out of the load-balancer pool, and leave the other server online to serve traffic:
* Stop the v1.0 application from running on the drained node, then deploy the new v1.1 version. Verify the deployment was successful by running tests on your updated application server. All this while,we must maintain the other server to run v1.0 of your application. All traffic will be handled by this server during this time.
* After ensuring deployment is successful add the server with v1.1 back to the load balancer pool.Proceed with draining the second server still running v1.0 of your application.
* Stop the v1.0 application on the second serve, deploy the new v1.1 version. Again, verify the deployment was successful and add it back to load balancer pool.
* rolling deployment is complete.
* If the servers use separate databases data will need to syced between the two after each deployment is complete.
* If you have to deploy this App to the Cloud what services would you use and explain the Architecture. You can mention for Cloud of your choice (AWS / Amazon / Google Cloud (GCP) / Oracle / Alibaba Cloud)
* Google cloud platform offers a the option to CDN and HTTP(S) Load Balancing services.We can pull content or reach web services that are google or in another cloud, using Google’s global high-performance network. The figure below represents the architecture that can be used.



* To build an application on the architecture above we can use:
* **CDN offered by google:** As part of Google Cloud, Cloud CDN caches content in 96 locations around the world and hands it off to 134 network edge locations, placing content close to users, usually within one network hop through their ISP. We can leverage this to creade a CDN for foo.com
* **Load balancer:** Cloud Load Balancing feature from GCP can be used to distribute compute resources/servers in single or multiple regions as needed to meet high availability requirements. Cloud Load Balancing can put your resources behind a single anycast IP and scale your resources up or down with intelligent autoscaling. Cloud Load Balancing with be integrated with Cloud CDN for optimal application and content delivery. In contrast to DNS-based global load balancing solutions, Cloud Load Balancing reacts instantaneously to changes in users, traffic, network, backend health, and other related conditions.Cloud Load Balancing can be applied to all of your traffic: HTTP(S), TCP/SSL, and UDP.
* **Application/service**: The application or the service can also be be hosted on the google cloud platform itself. Google’s cloud sql and datastore can be used as the SQL and NoSQL databases respectively.Google also has an option of Cloud Bigtable which is a high performance noSQL database.
* If we want to use a service that resides outside of Google Cloud, such as a web server or load balancer running on-prem, or object storage at a third-party cloud provider Google’s internet network endpoint groups allow us to configure a publicly addressable endpoint.

1. Write a Python program to build an automated Data Pipeline. The program should be a single file python code and should be able to do the following.

For this assignment I have created an example data pipeline in python.

* This pipeline is executed by the mains.py script that futher esecites 3 more scripts.
* The **first script** downloads data from different data sources. In realtime applications data of different formats and from different sources may need to be accessed and used. In this assignment I am downloading 4 .csv files from an online resource one for each year(2015-2018)
* These files contain total compensation of employees scoss various departments and job titles from a city.
* The **second script** does some data cleanup and transformations on each of these datasets. The column headers are made standard for each column and the unwanted columns are removed . Since in this assignment I am using data from only the Information and technology services department, I cleaned up data pertaining to just that department. This involved renaming the department in all datasets to a standard name. Finally the datasets are combined to create a final dataset comprising of data from all 4 years. In a real time pipeline we could save this data now to the datawarehouse or datacenter but for purposes on assignment I save it to local machine .

Path : (Assignments/data\_pipeline/data/dat\_final.csv).

* The **third script** reads this final data into a dataframe and performs some analysis on it.

1. A report of the average salary each year for each Job title in the Information and technology services department is generated. In a real time pipeline it could be emailed to stakeholders or sent to location accessible to multiple users but I am just printing and saving the report to local machine(Assignments/data\_pipeline/data/dat\_avg.csv).

2. Next we find out the 5 top paying jobs of the latest year and print that.

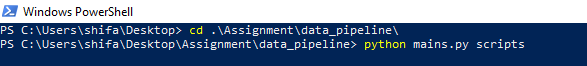
3. Finally a plot is generated to show the Salary trends for the top 5 job titles

**How to execute the script:**

* To execute the pipeline please navigate to the data\_pipeline directory to the Assignments/data\_pipeline directory and in there execute the mains.py lise so:

**python mains.py scripts**

“ scripts” should be the first argument



* 1. Navigate to a particular folder and execute other Python scripts which contains some kind of Data Ingestion / Transformation.

The script navigates to a folder called “scripts”(this is provided as the first argument to the script).

It then navigates to this folder to execute the 3 other scripts.

* 1. Take the following as command line input (bash style preferred)
     1. A folder name – Mandatory - The name of the folder to work upon

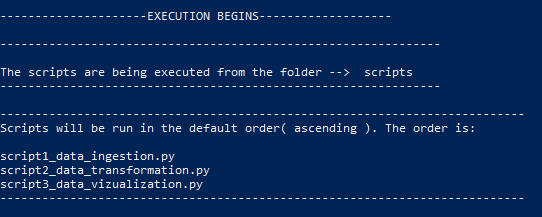
As seen in the screenshot above the script mains.py accepts the foldername(“scripts”) as an argument.

* + 1. Script Order – Optional - Ascending / Descending – The list of files order to work upon. Default Behavior – Ascending

There are 2 different ways of successfully executing the script.

1. script is executed like so: **python mains.py scripts**

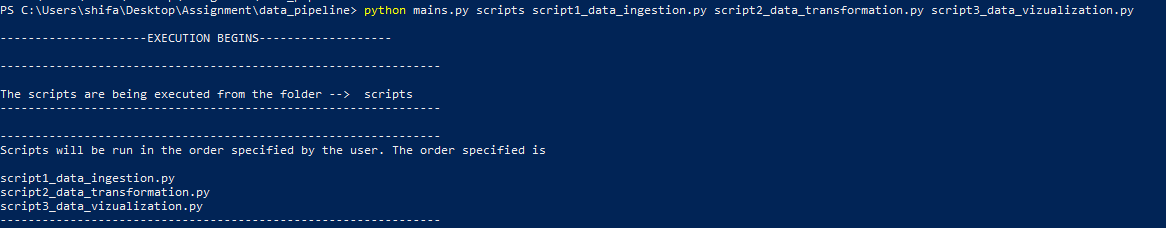
Scripts will be executed in the default ascending order.



2. script is executed like so: **python mains.py scripts script1\_data\_ingestion.py script2\_data\_transformation.py script3\_data\_vizualization.py**

Here we provide a s arguments an order in which scripts are run.

First argument must be location of scripts(“scripts”) followed by the scripts to execute in the order of execution.

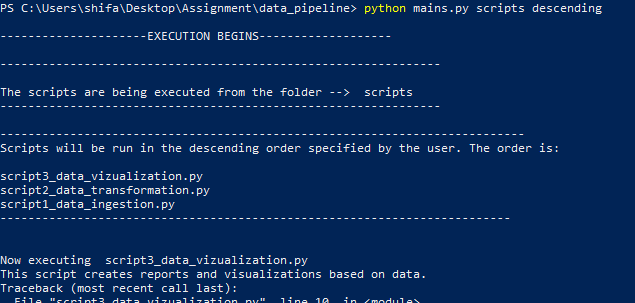


There is a third order that the scripts can be run in and the mains,py is equipped to run it but if we run these scripts in that order they will error out.

**python mains.py scripts descending**

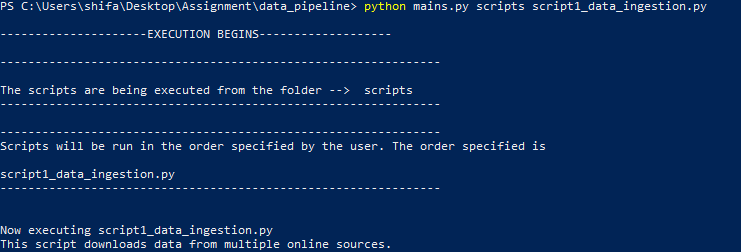
First argument must be location of scripts(“scripts”) and second the word “descending”.

**Note: Please don’t use this way of execution.**



* + 1. File name – Optional – The exact file name to run. Default Behavior – All the files

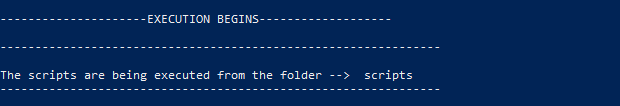
If script is executed like so: **python mains.py scripts script1\_data\_ingestion.py** only the specied script will be executed



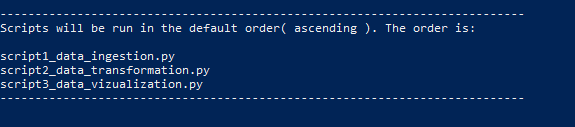
* + 1. Outfile – Optional – Log the output to the mentioned file name. Default behavior – Console

Output is displayed on console.

* + 1. Help – Optional - Display the help and usage information
  1. The following should be displayed as Output (beautification / Table preferred)
     1. The Folder where it is executed



* + 1. The file/files it is executed (in the order of execution)



* + 1. The output of the files executed. Just the console output returned should be okay

