Assignment 6: Cloud Native Visual Analytics
Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia

Introduction

This report documents the steps I followed to complete Assignment 6: Cloud Native Visual Analytics using ClickHouse and Grafana deployed on CSC Rahti OpenShift. The main objective of the assignment was to practice deploying cloud-native applications in a containerized environment and to build an end-to-end data analytics workflow. The work focused on deploying ClickHouse as a database service, importing and managing datasets using SQL, deploying Grafana for visualization, and creating interactive dashboards that demonstrate aggregation and analytical queries. Screenshots are included throughout the report to verify that each deployment and configuration step was completed successfully.

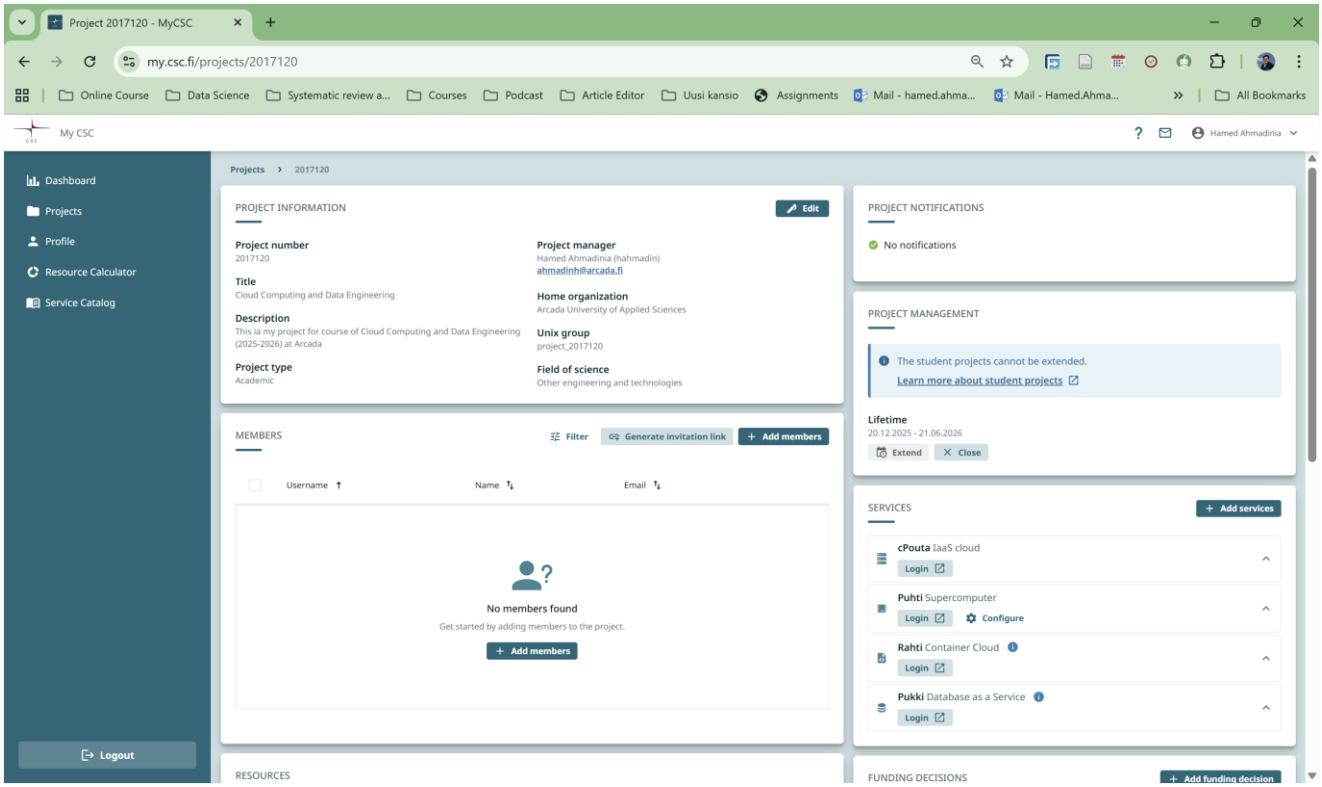
The assignment was divided into several phases. I began by preparing the Rahti OpenShift project and ensuring both the web console and oc CLI were correctly configured. Next, I deployed ClickHouse using the provided YAML configuration and verified that the pod, service, and route were working. After accessing the ClickHouse Web UI, I created tables and imported a movie dataset for analysis. I then deployed Grafana in the same OpenShift App to ensure internal networking, configured the ClickHouse data source, and created dashboards showing aggregations such as movies per year, average metascore by genre, and top directors. In the optional part, I added a second time-series dataset and implemented advanced analytics including monthly averages, yearly ranges, and percentile-based metrics.

During the assignment, I encountered several technical challenges. One key issue was ensuring that ClickHouse and Grafana were placed in the same OpenShift App so they could communicate over the internal service network. Another challenge involved correctly configuring database credentials and routes to avoid connection errors. Additionally, careful attention was required when writing SQL queries for aggregation and statistical metrics to ensure accurate results and efficient execution within Rahti's resource limits.

Phase A — Rahti project preparation (login and project selection)

In this phase, I prepared my Rahti OpenShift environment so I could deploy applications later without issues. First, I logged in to the CSC Rahti OpenShift web console using my university account. After logging in, I made sure I was in the Developer view instead of the Administrator view. Then I went to Home → Projects and selected my course project named my-assignment-6-cc. I confirmed that the project name appeared at the top of the page and that the Topology view opened correctly. This confirmed that I was working in the correct namespace. Next, I set up the OpenShift command line tool (oc CLI). In the web console, I opened Command Line Tools and downloaded the oc client for Windows (x86_64). After extracting the files, I opened Command Prompt in the folder where oc.exe

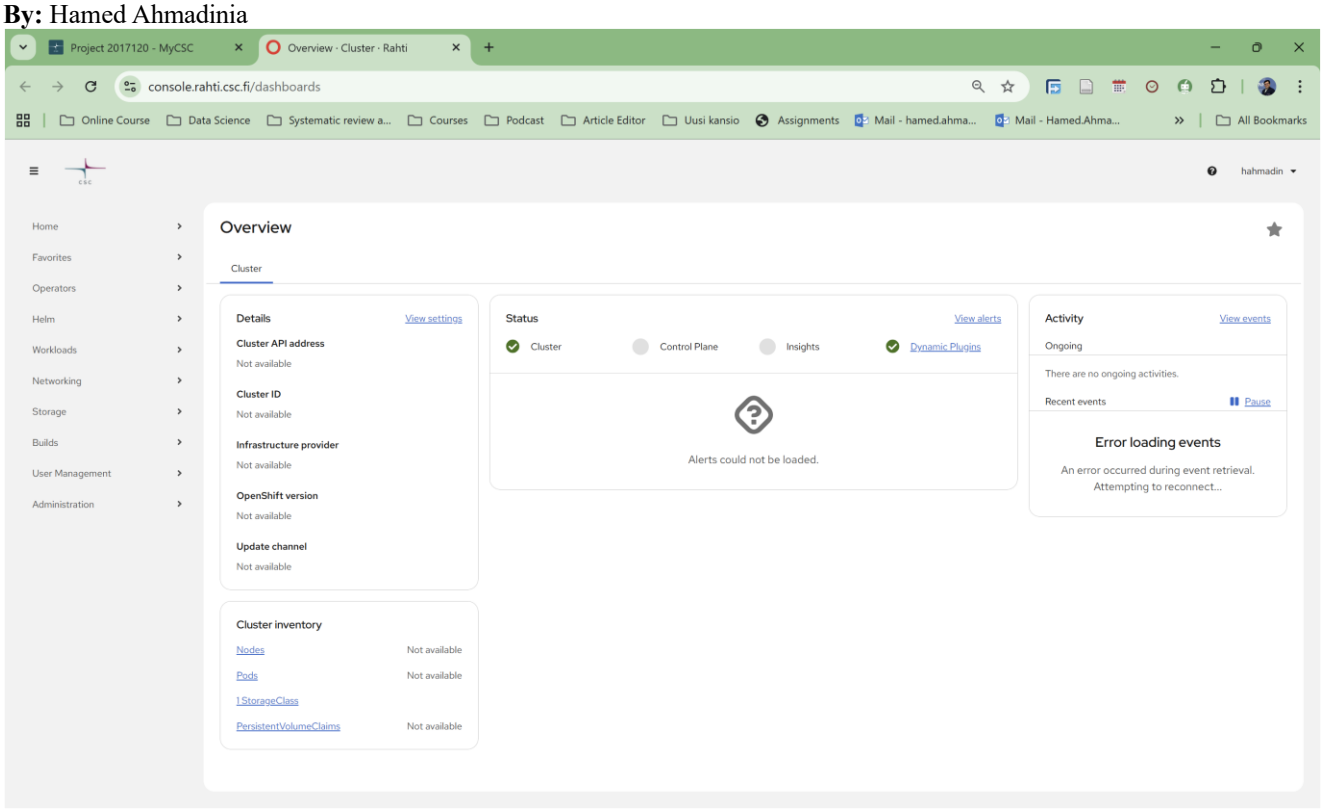
was located and ran oc version to test it. The command worked successfully. Then I copied the login command from the OpenShift console by clicking Copy login command and selecting Display Token. I copied the full oc login command and pasted it into the terminal. The login was successful. After that, I ran oc whoami to confirm my username and oc project to confirm that I was using the correct project, my-assignment-6-cc. At the end of this phase, both the web console and the oc CLI were working correctly. The correct project was selected, and the environment was ready for deployment in the next phase.

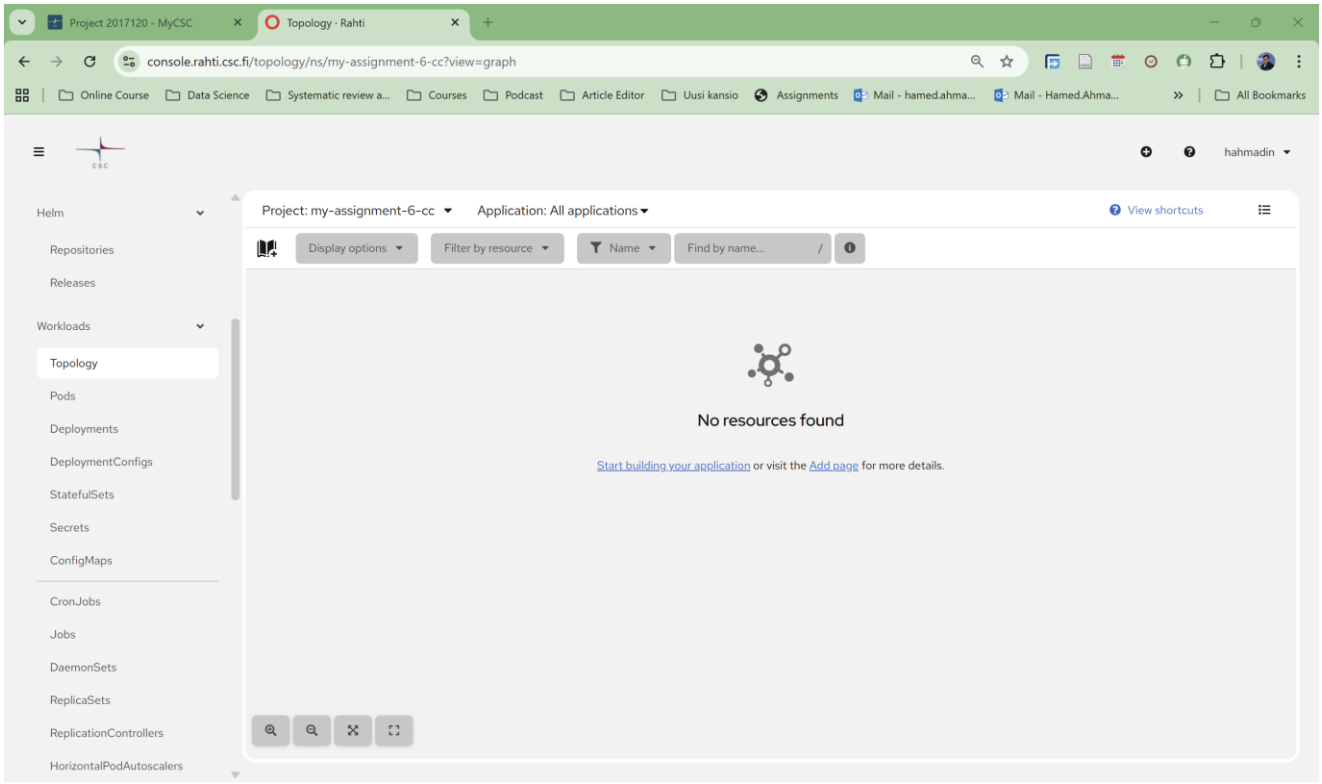


Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering

By: Hamed Ahmadinia

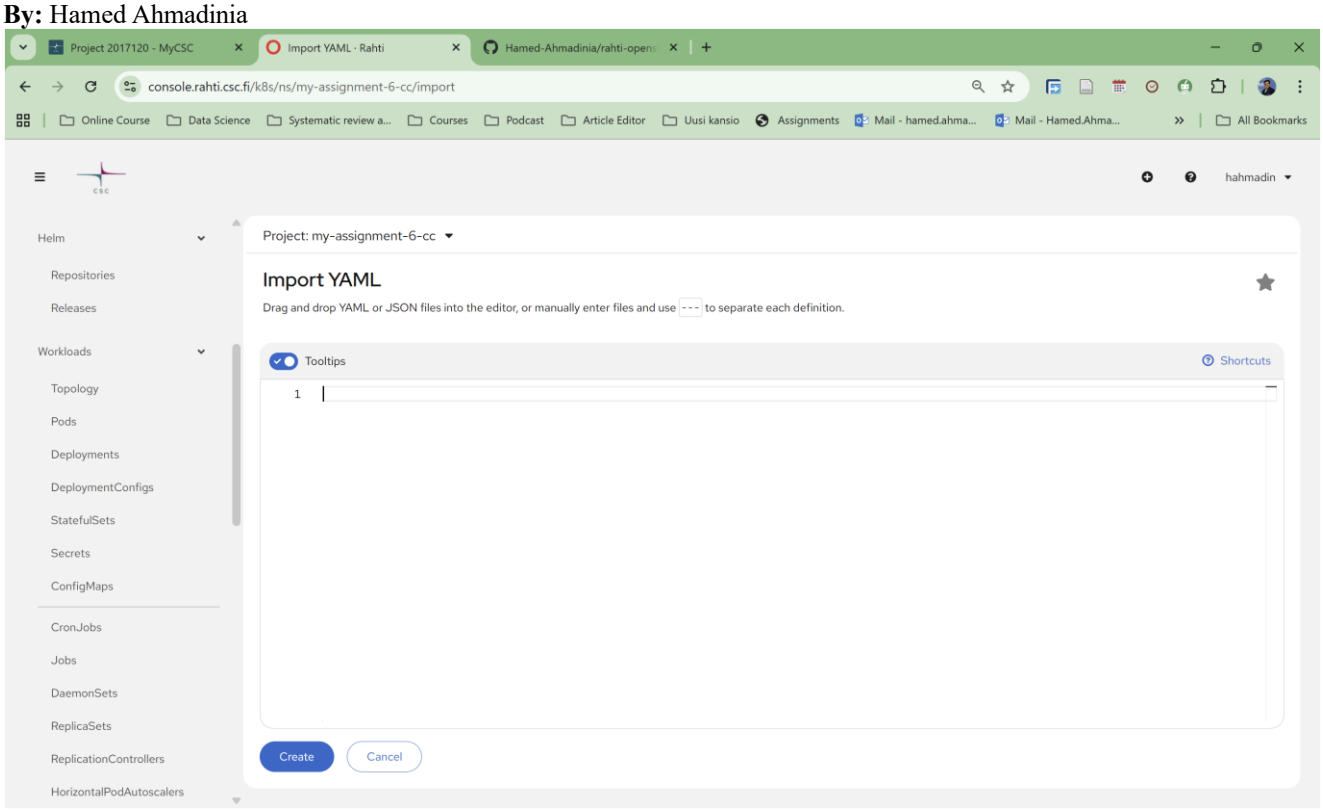


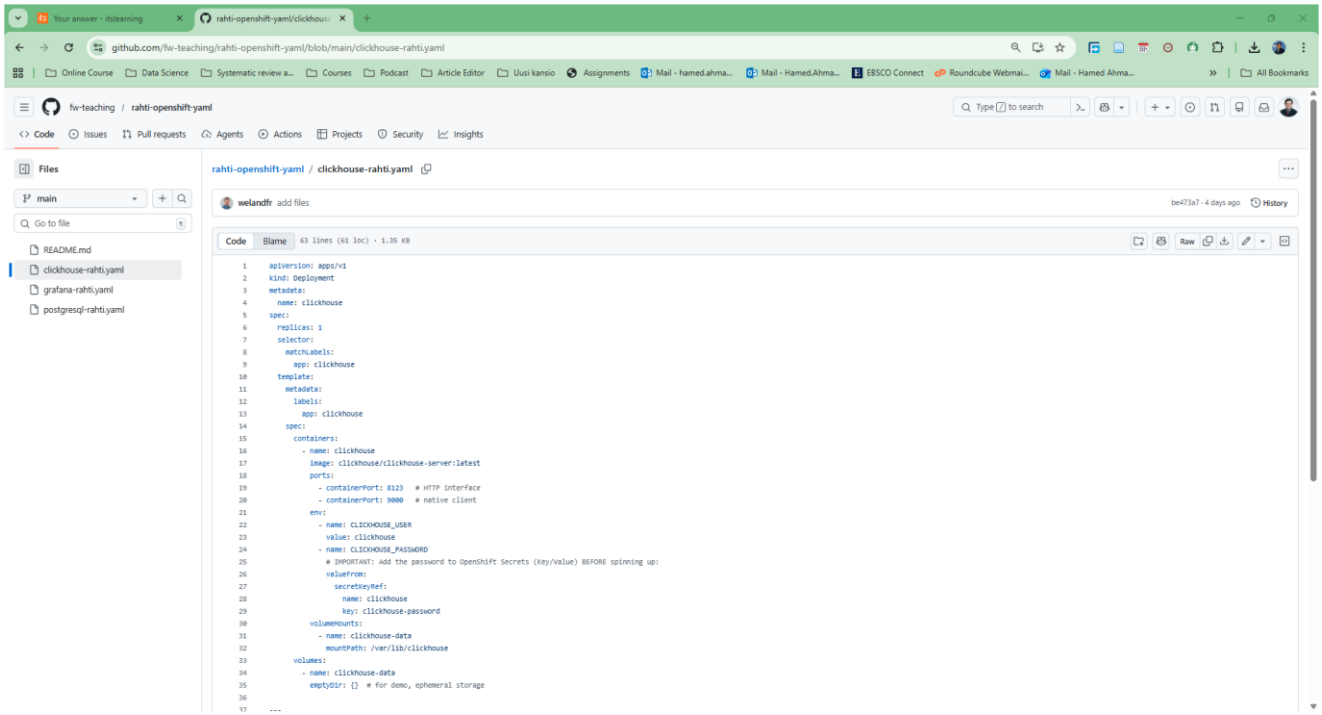


Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering

By: Hamed Ahmadinia

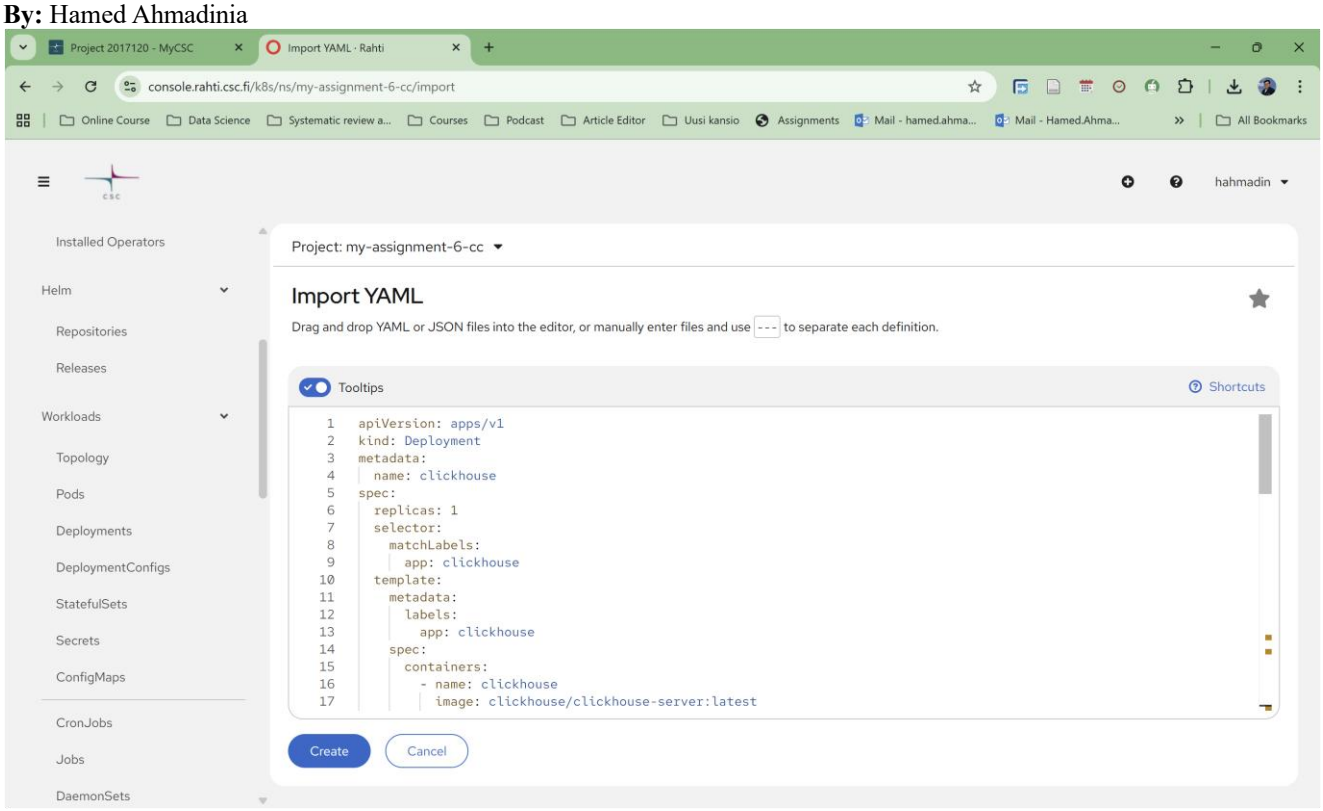


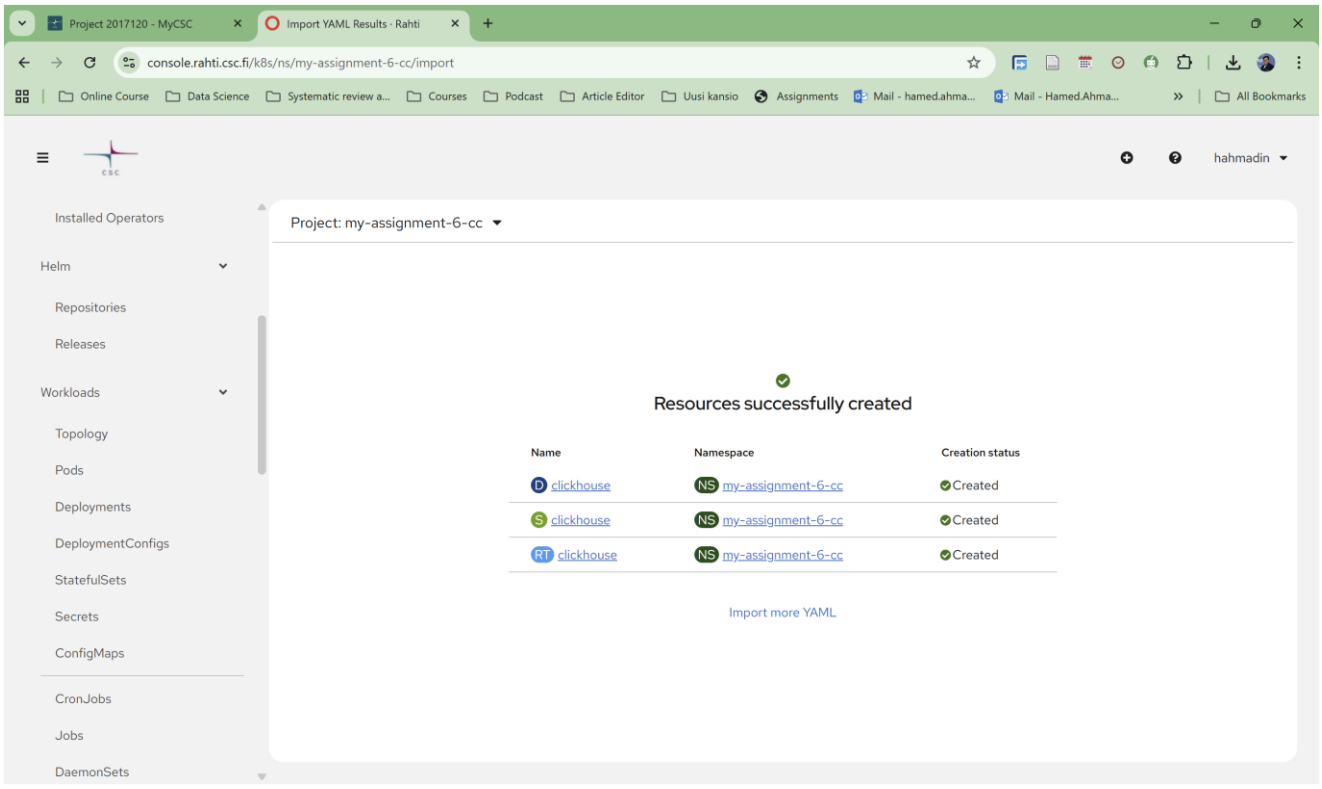


Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering

By: Hamed Ahmadinia

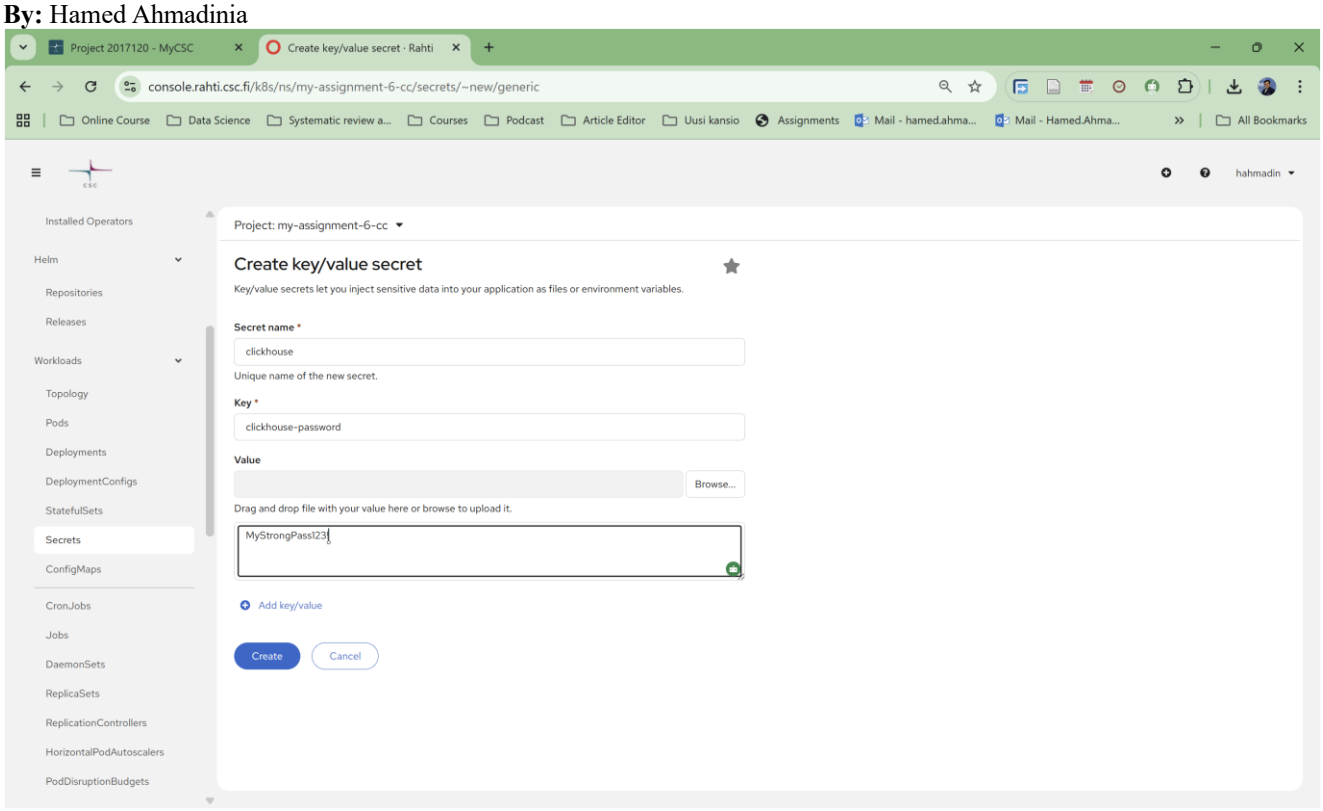


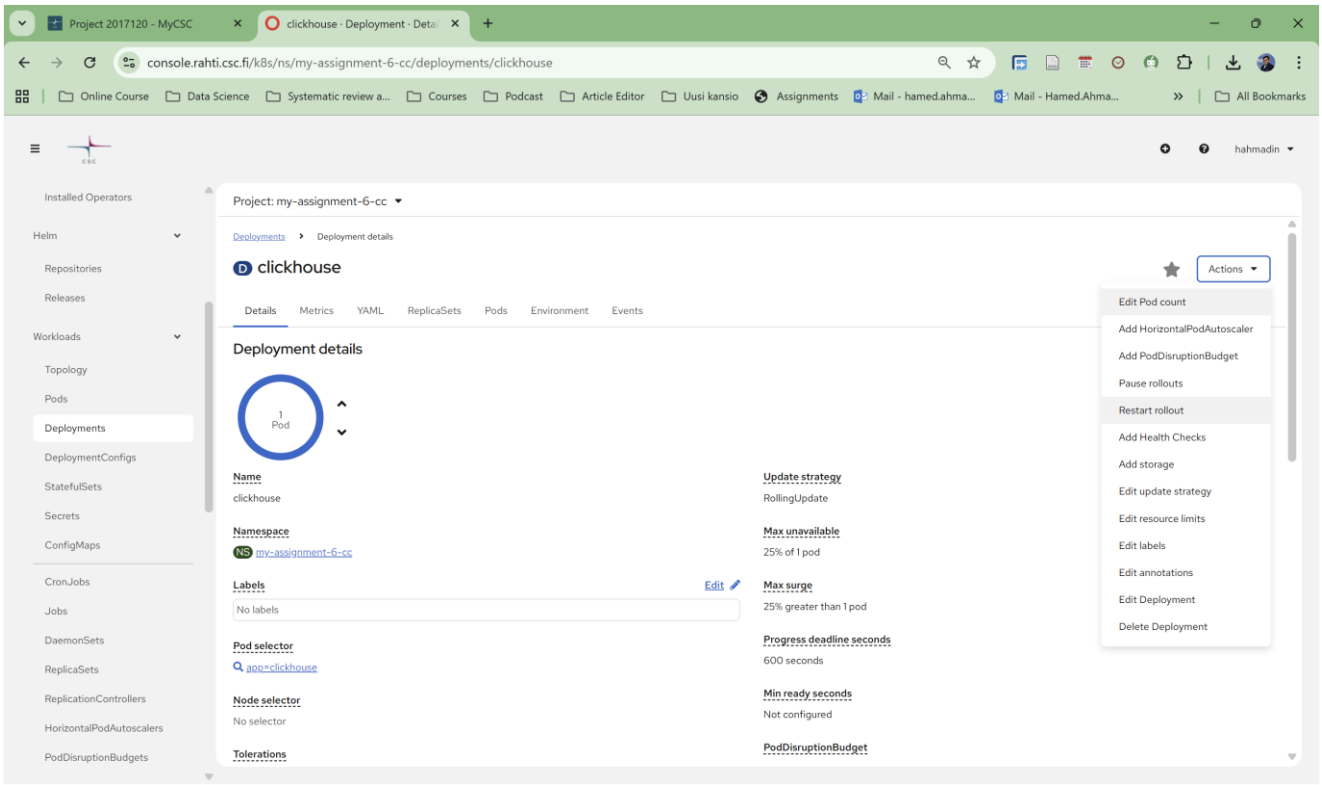


Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering

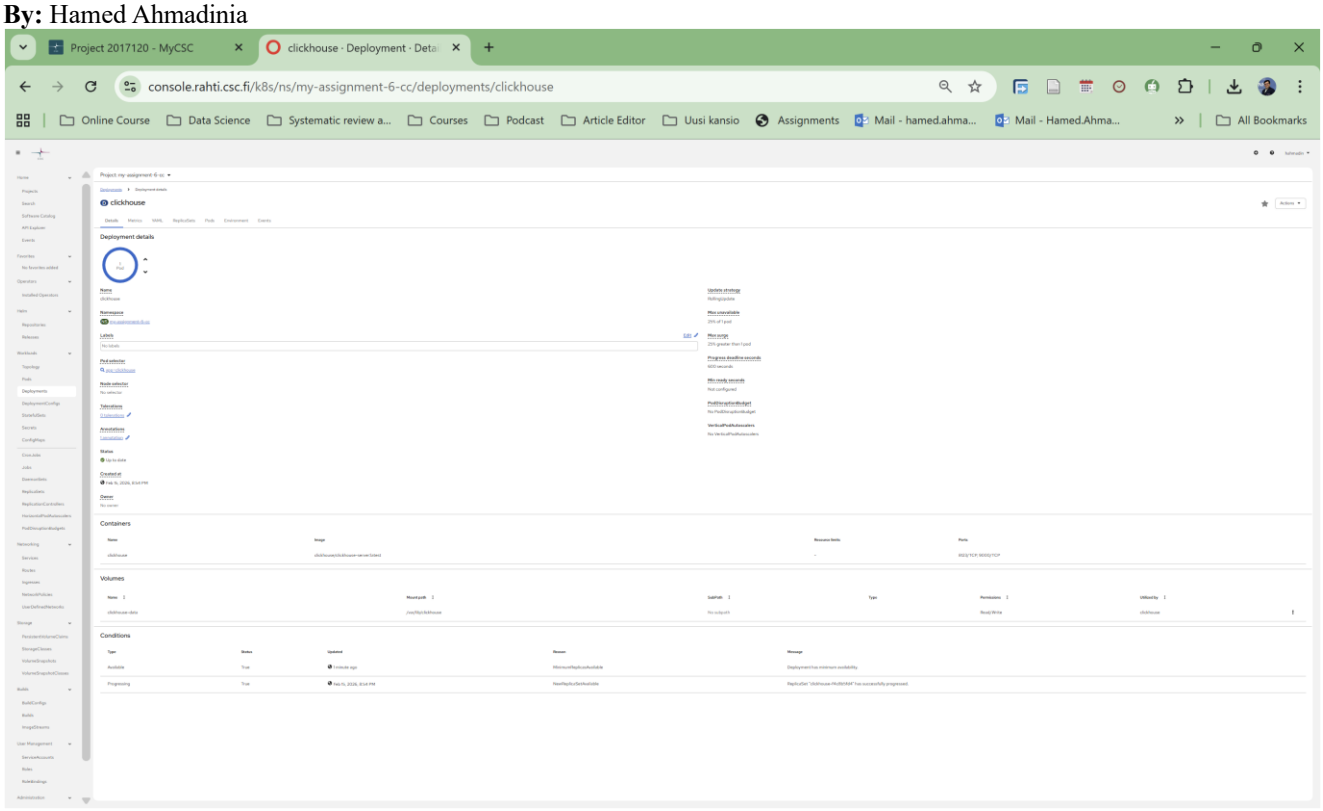
By: Hamed Ahmadinia

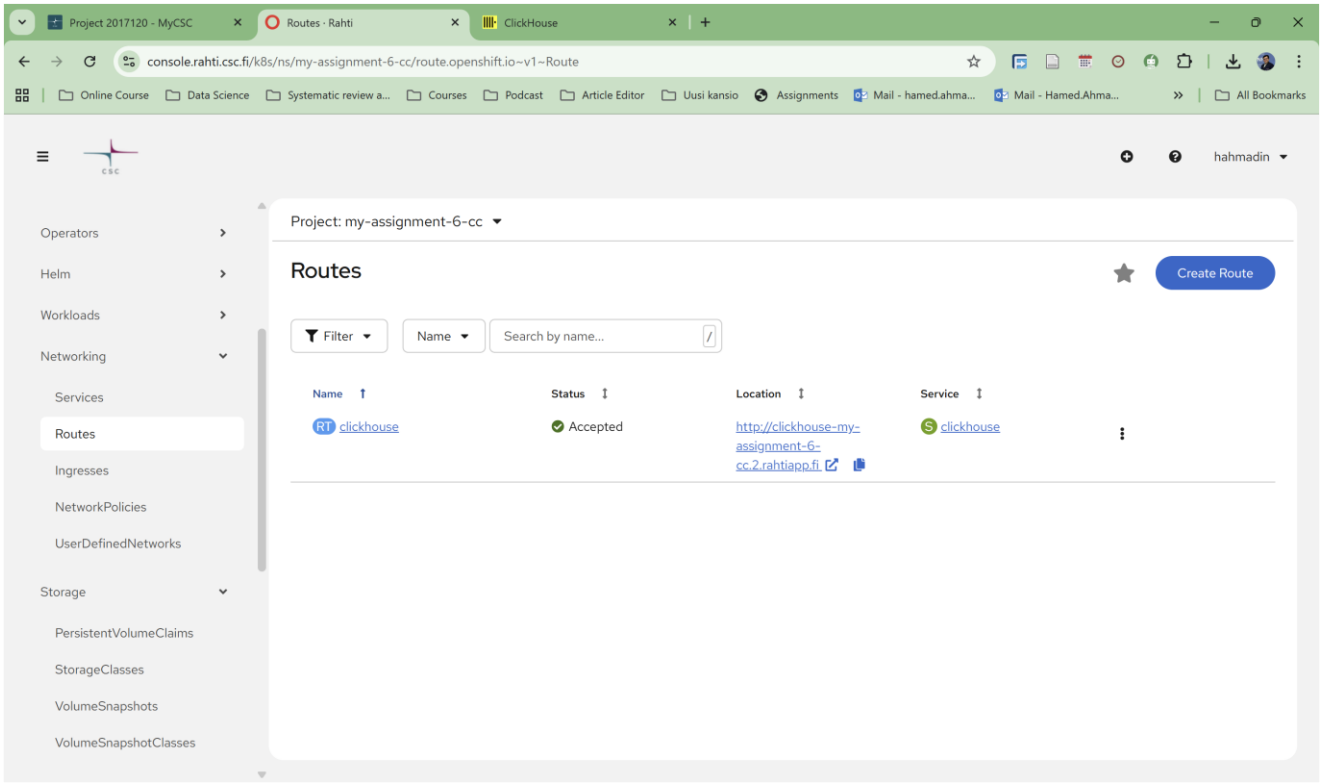




Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia

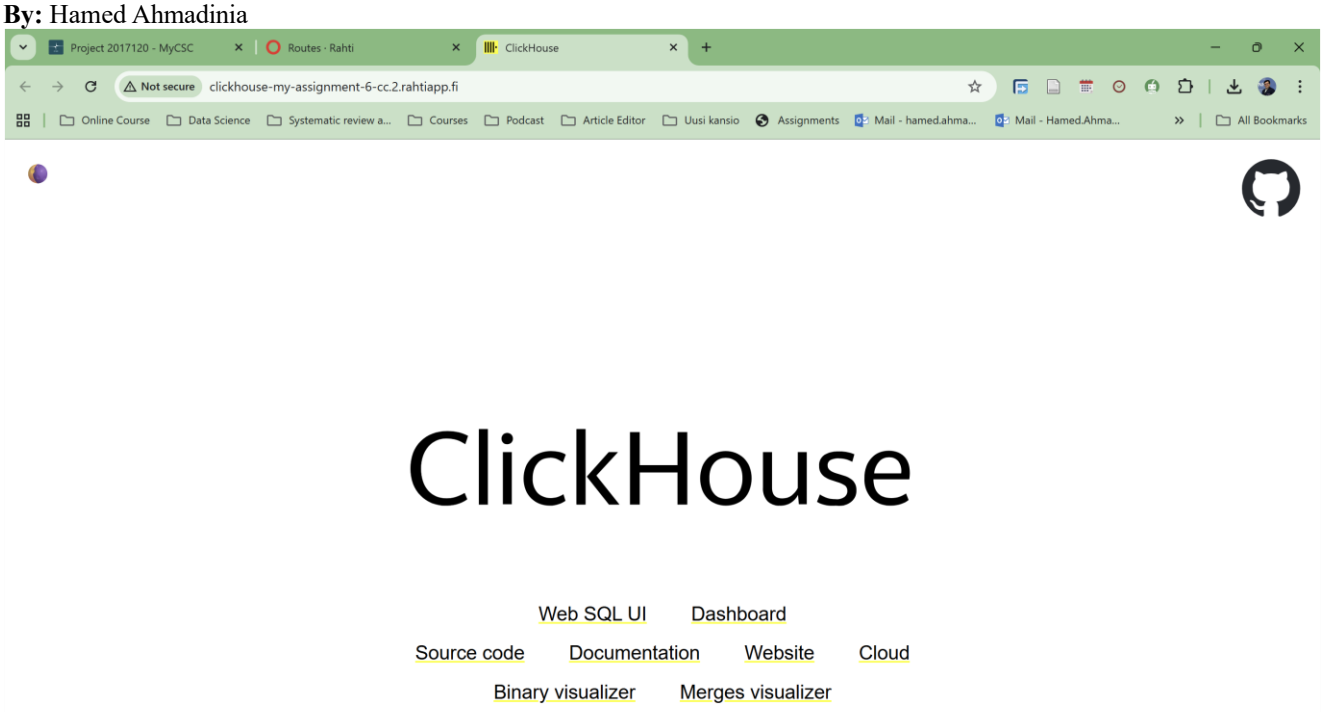


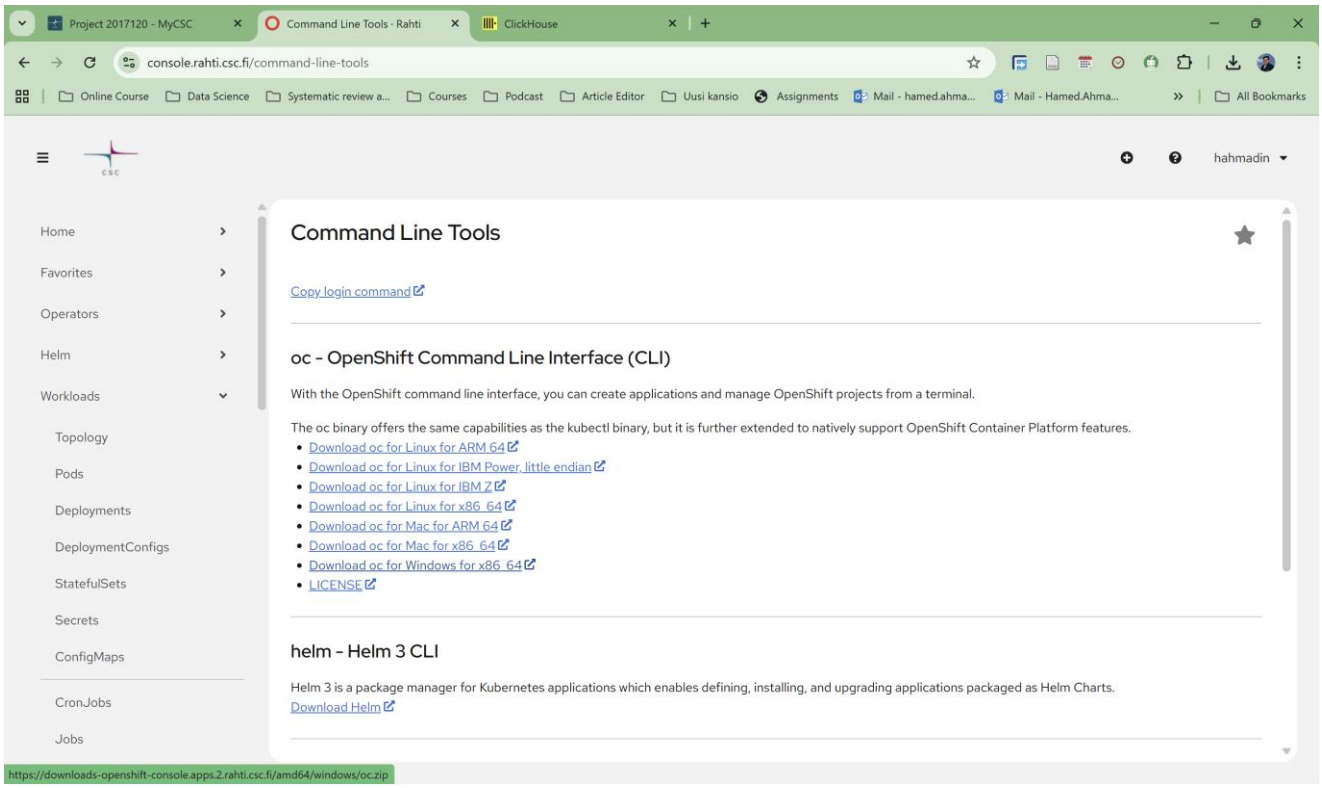


Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering

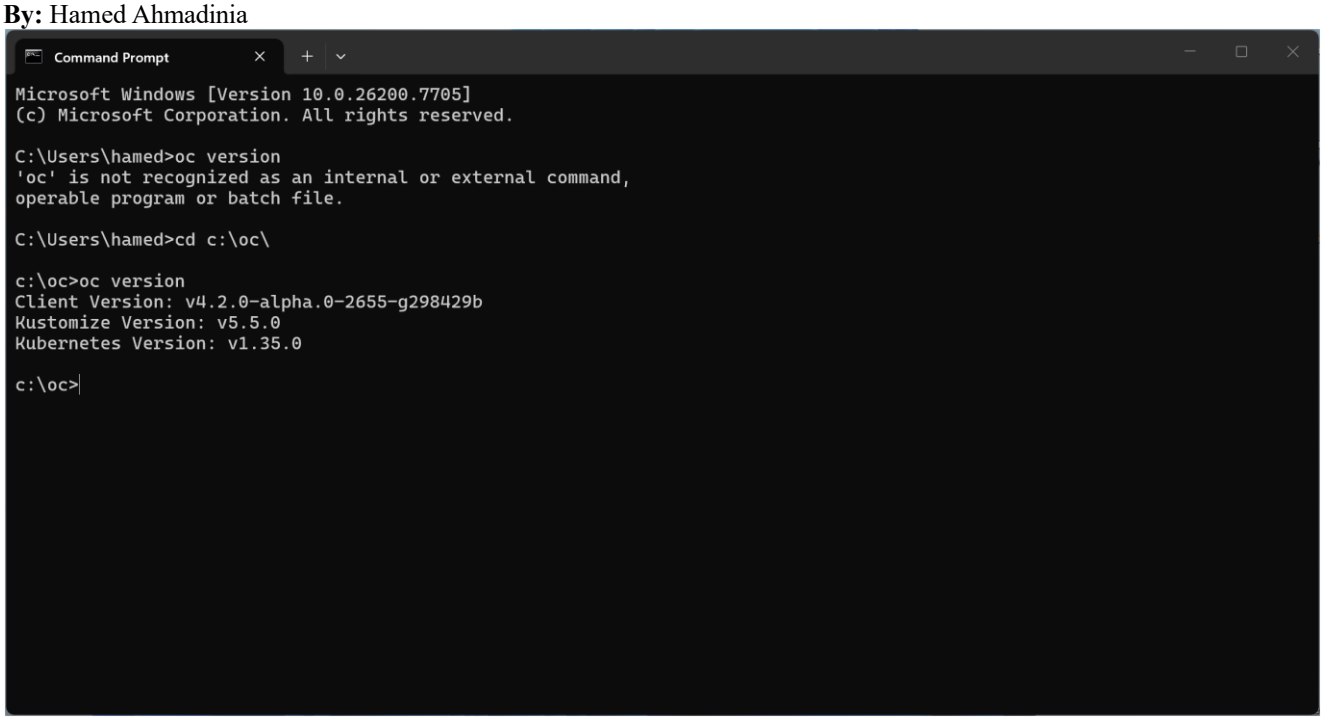
By: Hamed Ahmadinia

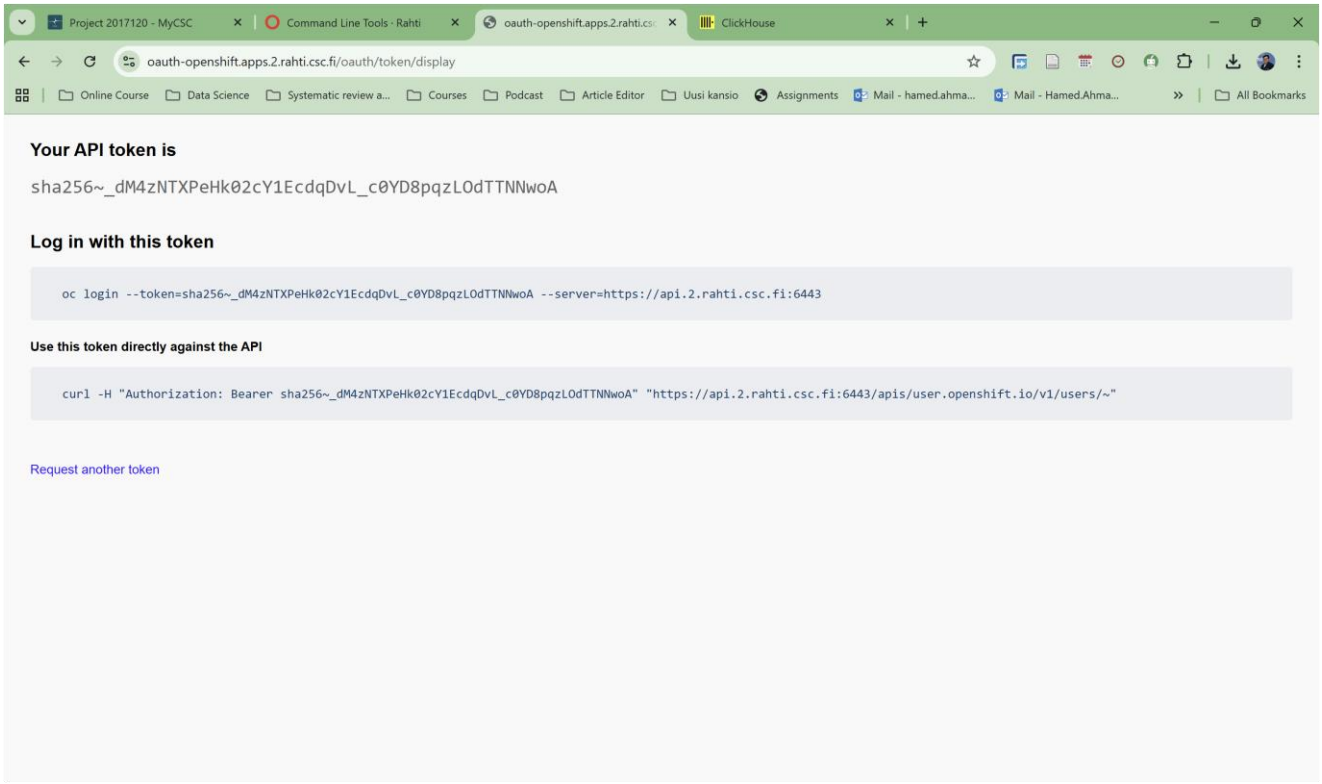




Assignment 6: Cloud Native Visual Analytics

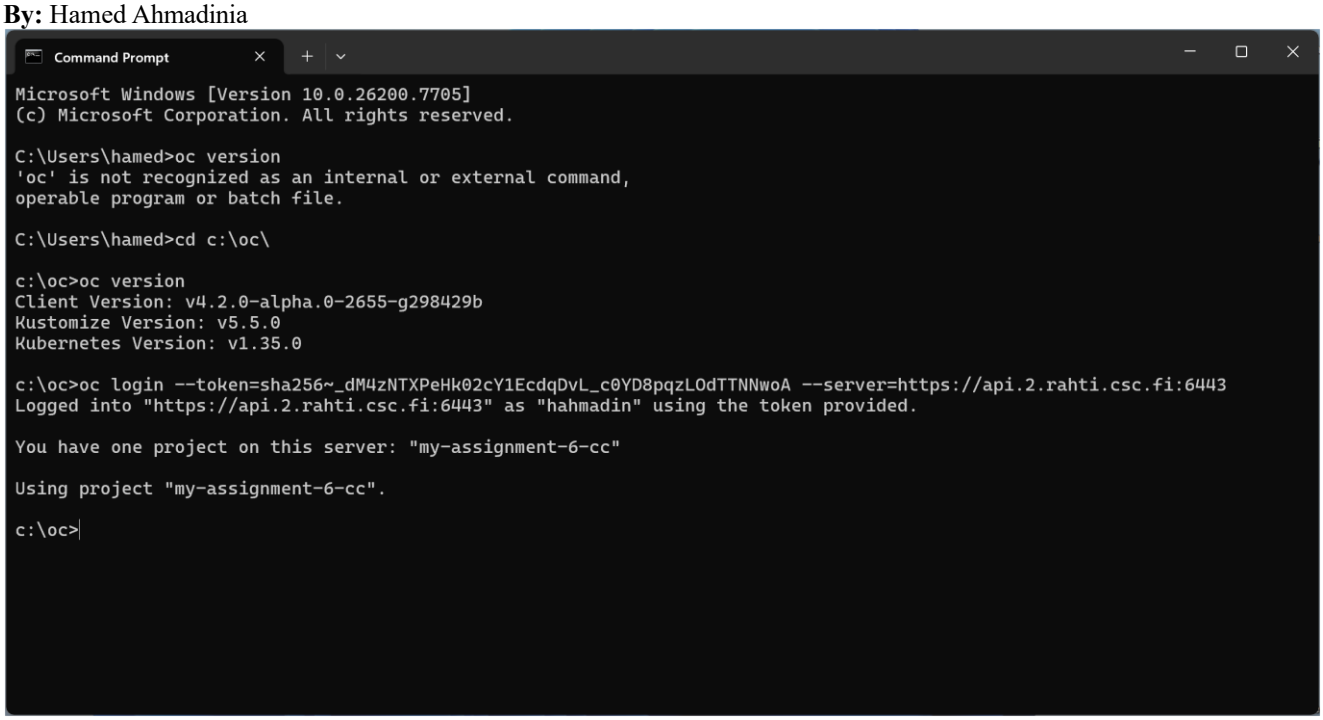
Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia

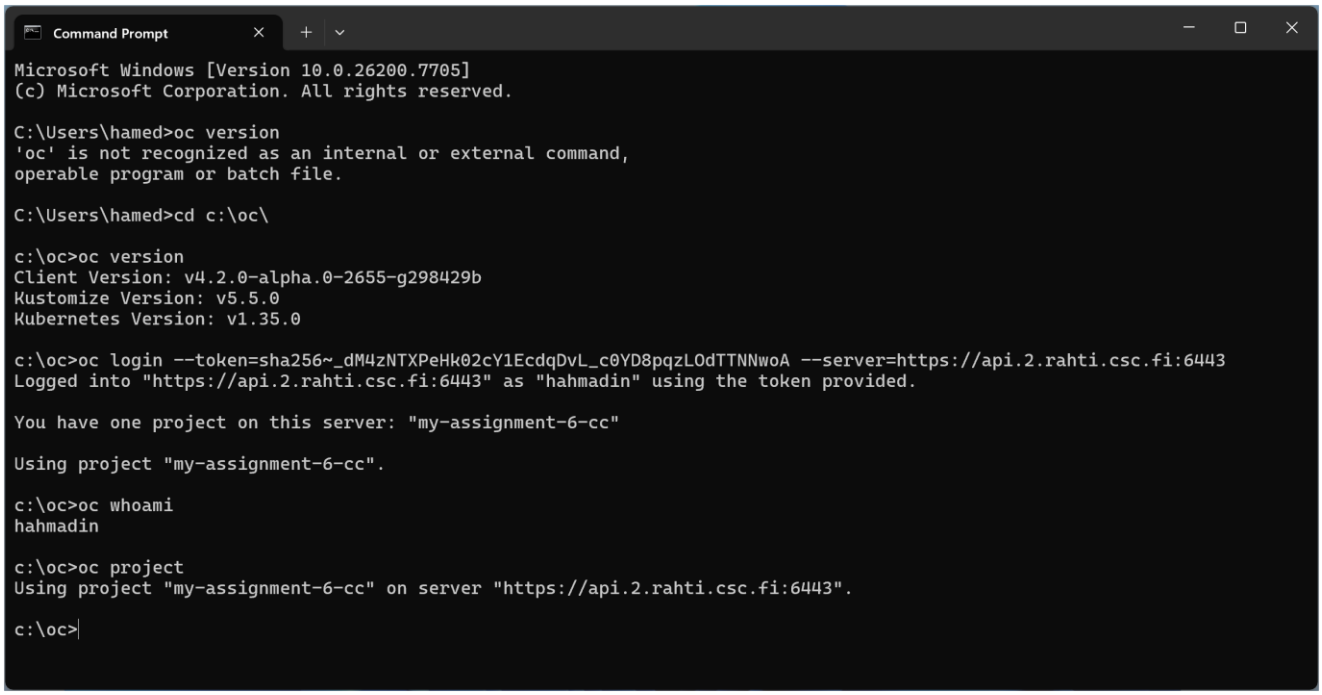




Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia





Assignment 6: Cloud Native Visual Analytics
Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia

Phase B — Deploy ClickHouse and Load Data

In this phase, I deployed ClickHouse in my Rahti OpenShift project and verified that it was working correctly. First, I used the provided YAML configuration from the course GitHub repository to deploy ClickHouse in my project namespace (my-assignment-6-cc). After applying the configuration, I opened the Topology view in the OpenShift web console and confirmed that the ClickHouse pod was created and running without errors. Next, I checked that the ClickHouse route was working. I opened the generated route URL in my browser, which loaded the ClickHouse Web SQL interface. This confirmed that the service was publicly accessible. I then logged in to the ClickHouse Web UI using the username clickhouse and the password that I created earlier as a secret in OpenShift. After logging in successfully, I tested the connection by running a simple SQL query:

SELECT 1;

The query returned a result without errors, confirming that ClickHouse was functioning correctly.

After confirming the connection, I created a new database called **movies** using:

CREATE DATABASE movies;

I verified that the database was created by running:

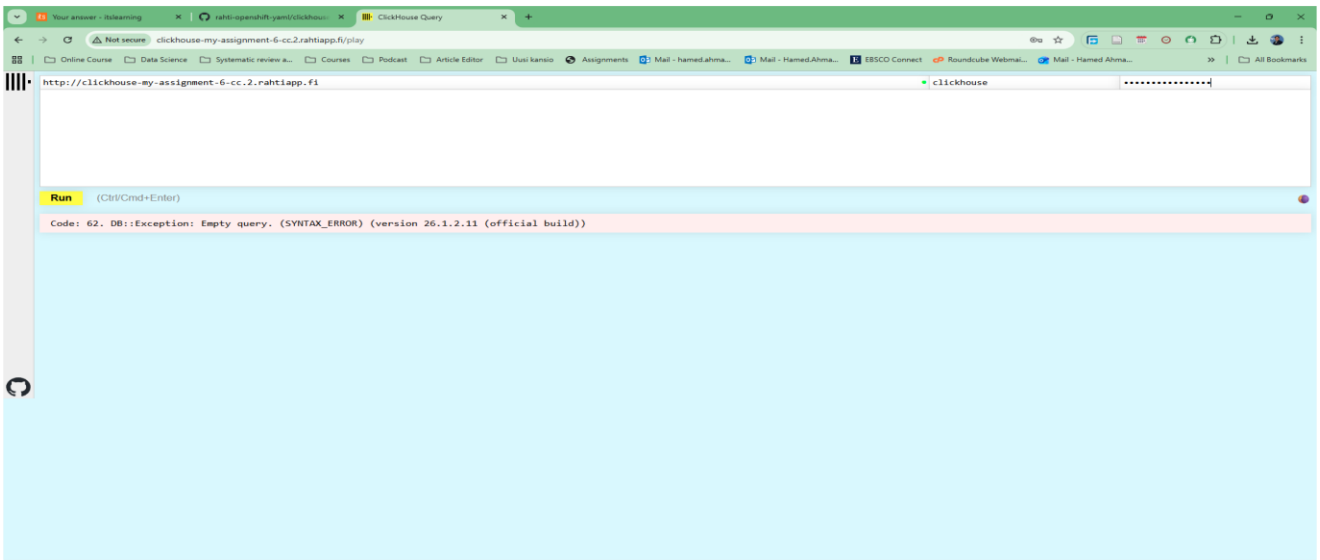
SHOW DATABASES;

The database “movies” appeared in the list.

Then, I created a table called movies_data inside the movies database. After creating the table, I verified its existence using:

SHOW TABLES FROM movies;

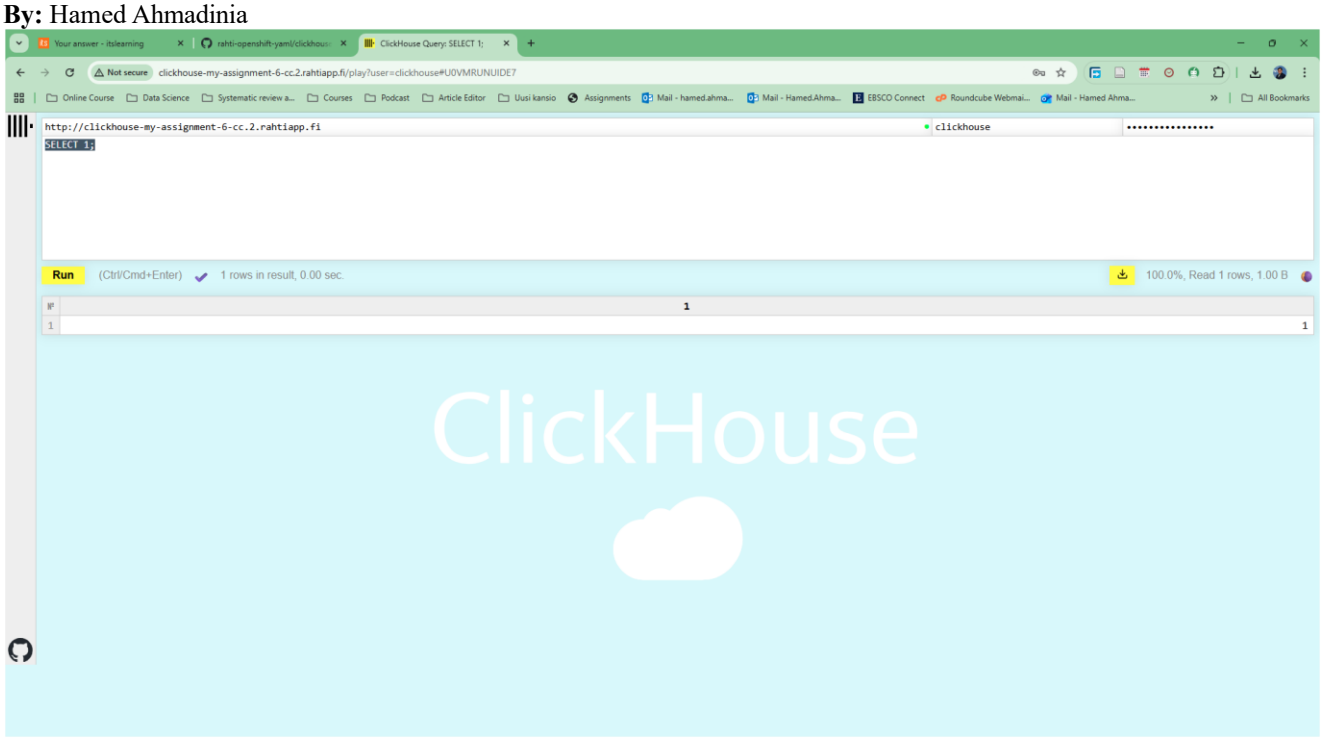
The table was listed successfully. At this stage, ClickHouse was deployed, accessible through the web interface, and ready to store data. The database and table structure were created successfully, and the system was prepared for importing the dataset in the next phase.

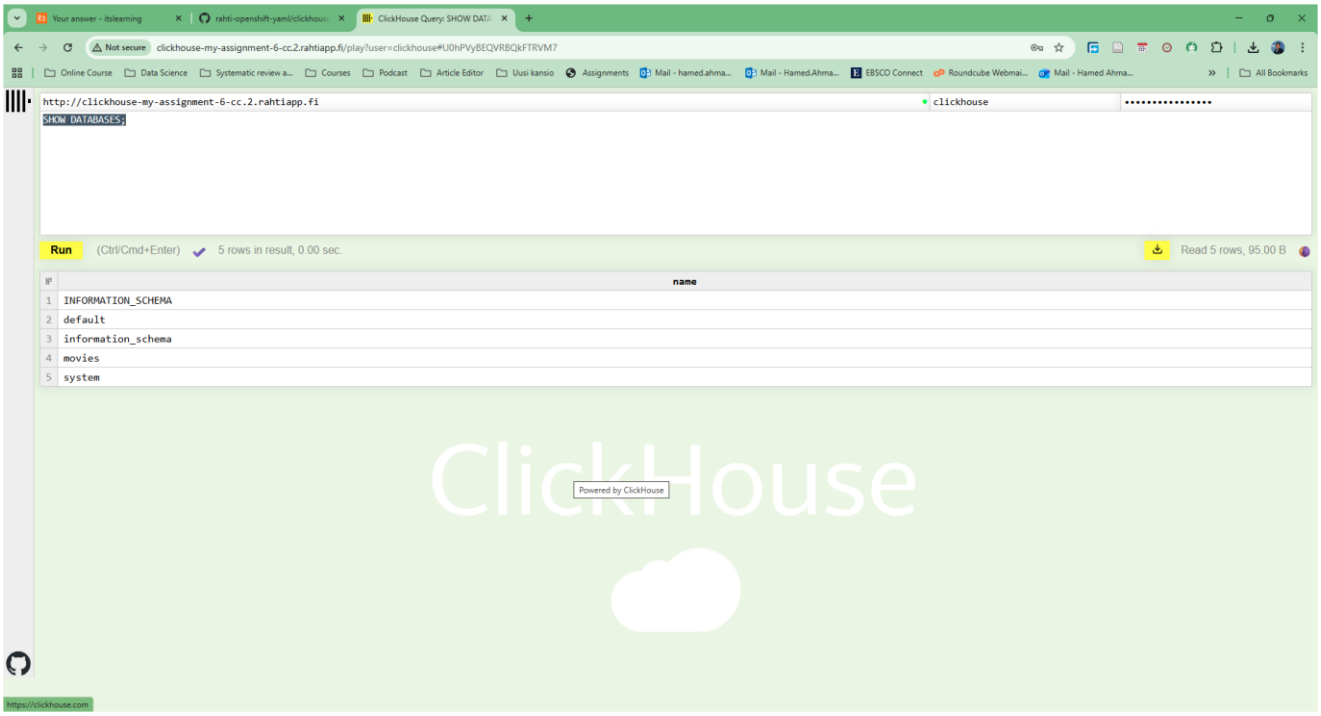


Assignment 6: Cloud Native Visual Analytics

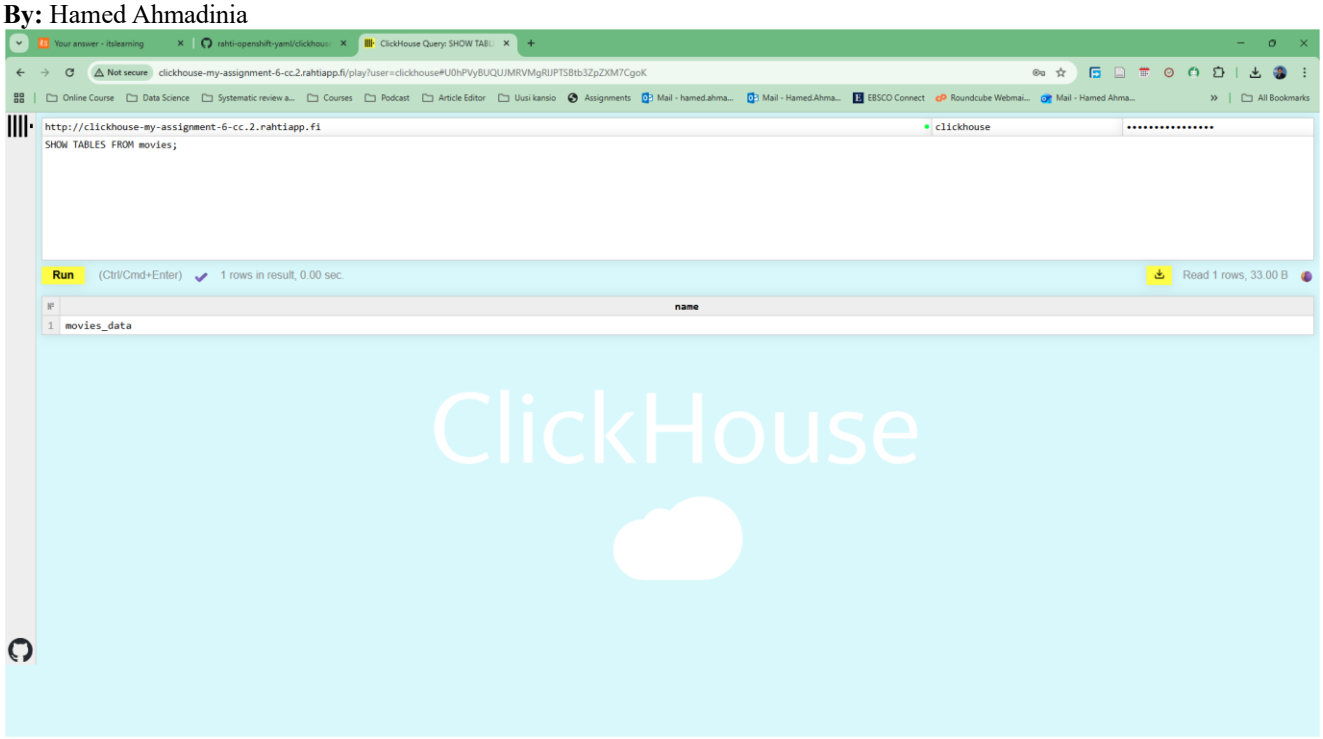
Course: Cloud Computing and Data Engineering

By: Hamed Ahmadinia





Assignment 6: Cloud Native Visual Analytics
Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia



Phase C — Import Movie Dataset into ClickHouse

In this phase, I imported the movie dataset into ClickHouse and verified that the data was stored correctly. First, I opened the ClickHouse Web SQL UI and logged in using my ClickHouse credentials. After logging in, I confirmed that the movies database and the movies_data table already existed from the previous phase. Next, I inserted the dataset into the table using a SQL command that loaded the CSV file directly from the provided URL. The file was imported using the CSVWithNames format so that ClickHouse could automatically recognize the column headers. The import process completed without errors. After inserting the data, I verified that the rows were successfully added. I ran:

SELECT count() AS rows FROM movies.movies_data;

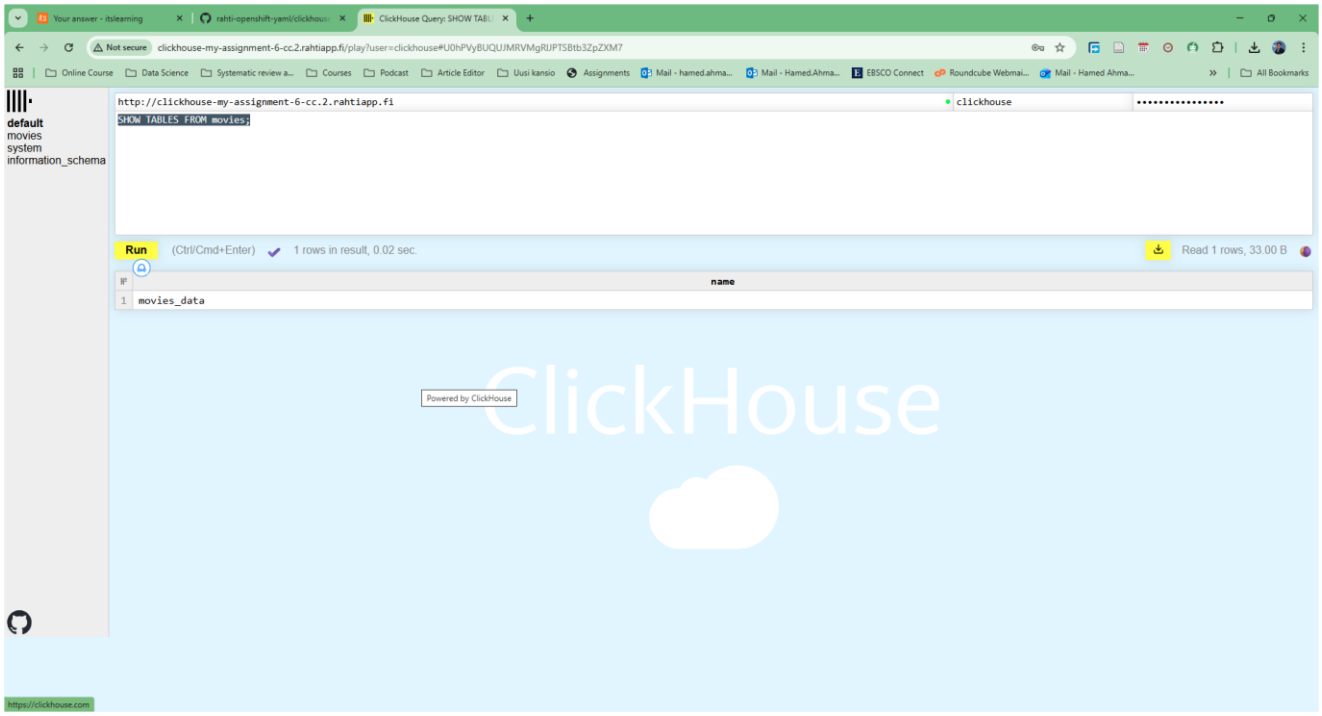
The result showed 16,939 rows, confirming that the dataset was fully imported. To make sure the data was correct and readable, I also ran:

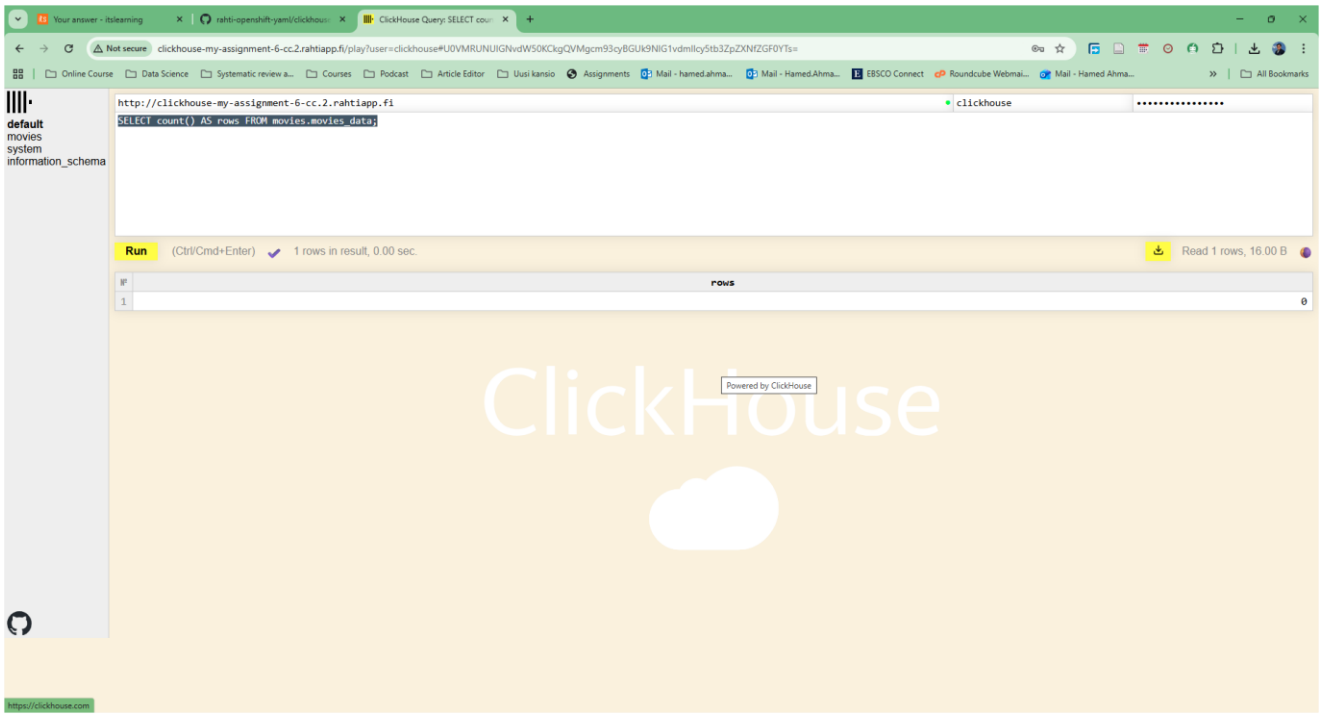
*SELECT * FROM movies.movies_data LIMIT 10;*

The query returned movie titles such as *Star Trek*, *Invictus*, and *Avatar*, which confirmed that the table structure and data were working properly. At this stage, the dataset was successfully imported into ClickHouse, and the database was ready to be connected to Grafana in the next phase.

Assignment 6: Cloud Native Visual Analytics

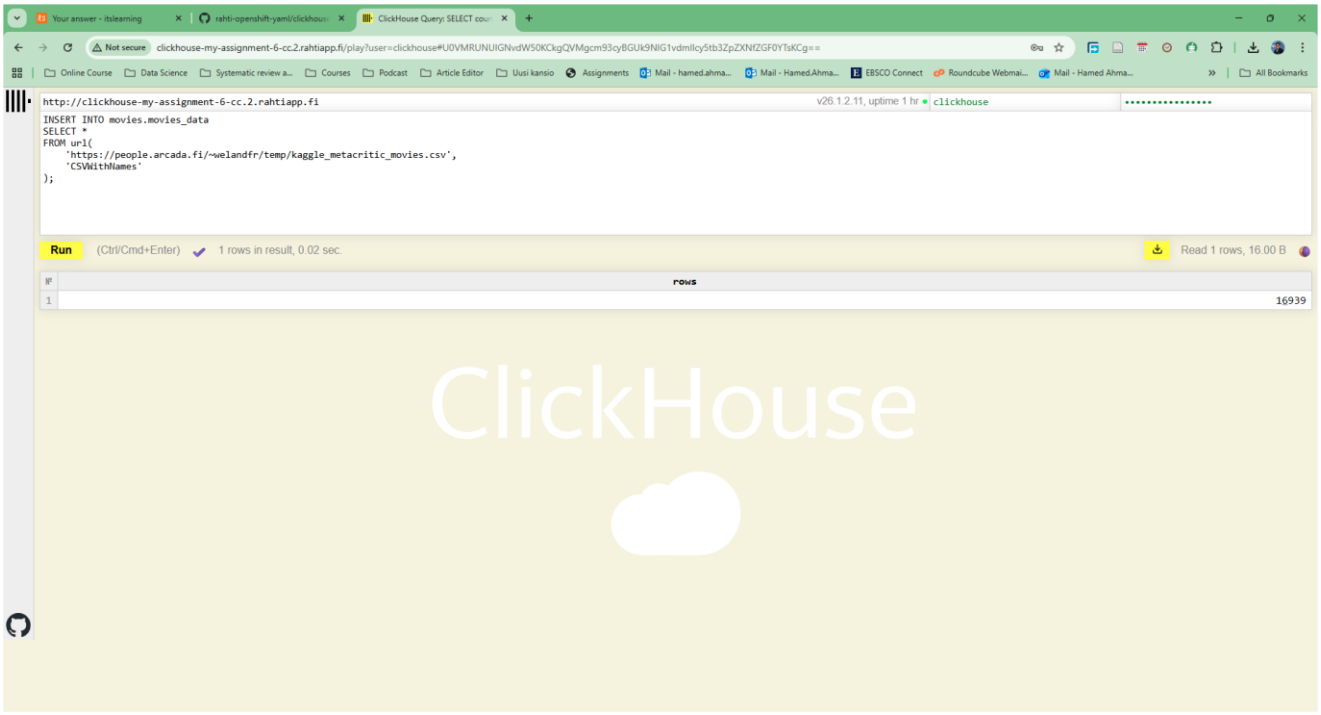
Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia

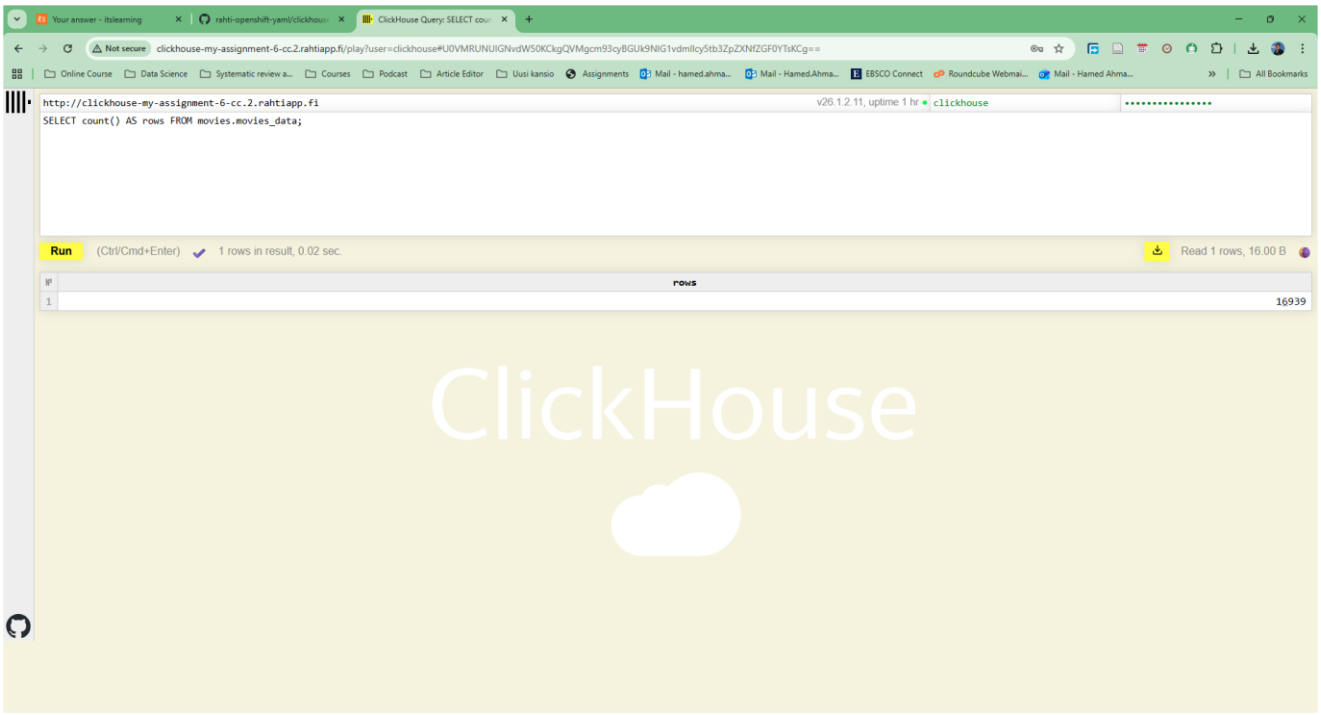




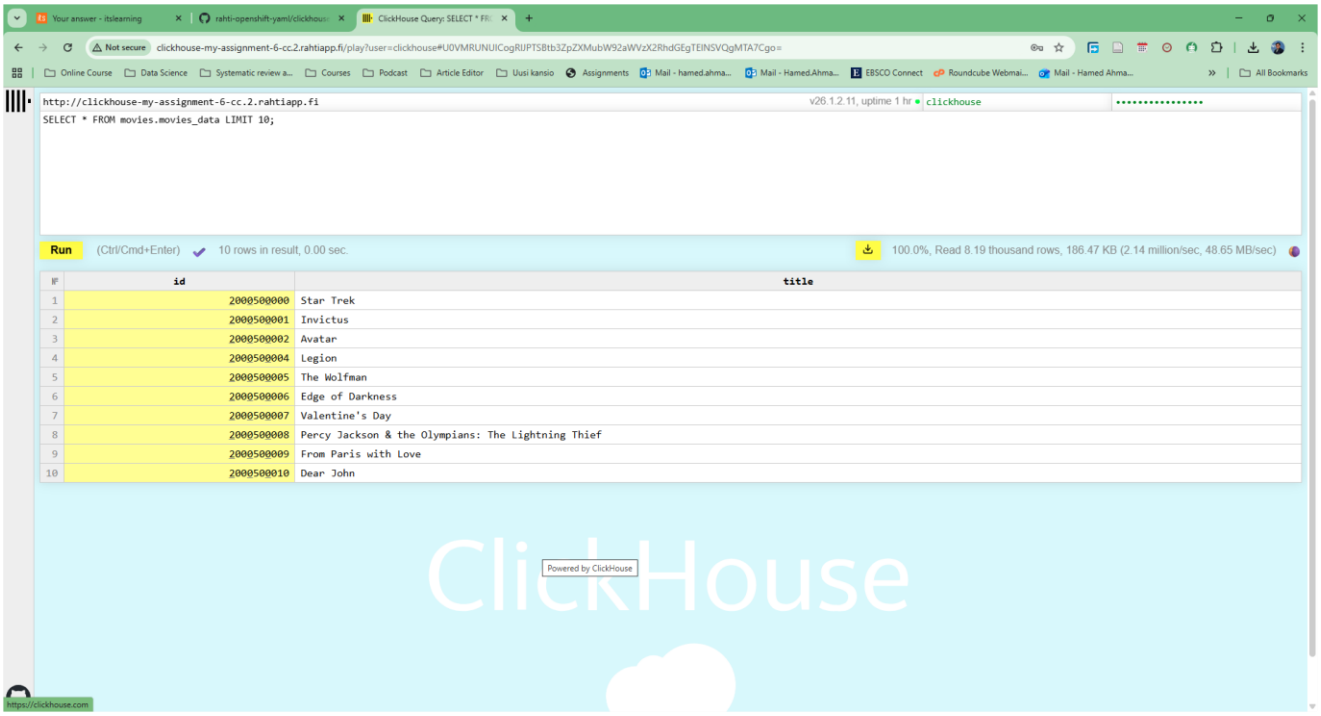
Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia





Assignment 6: Cloud Native Visual Analytics
Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia



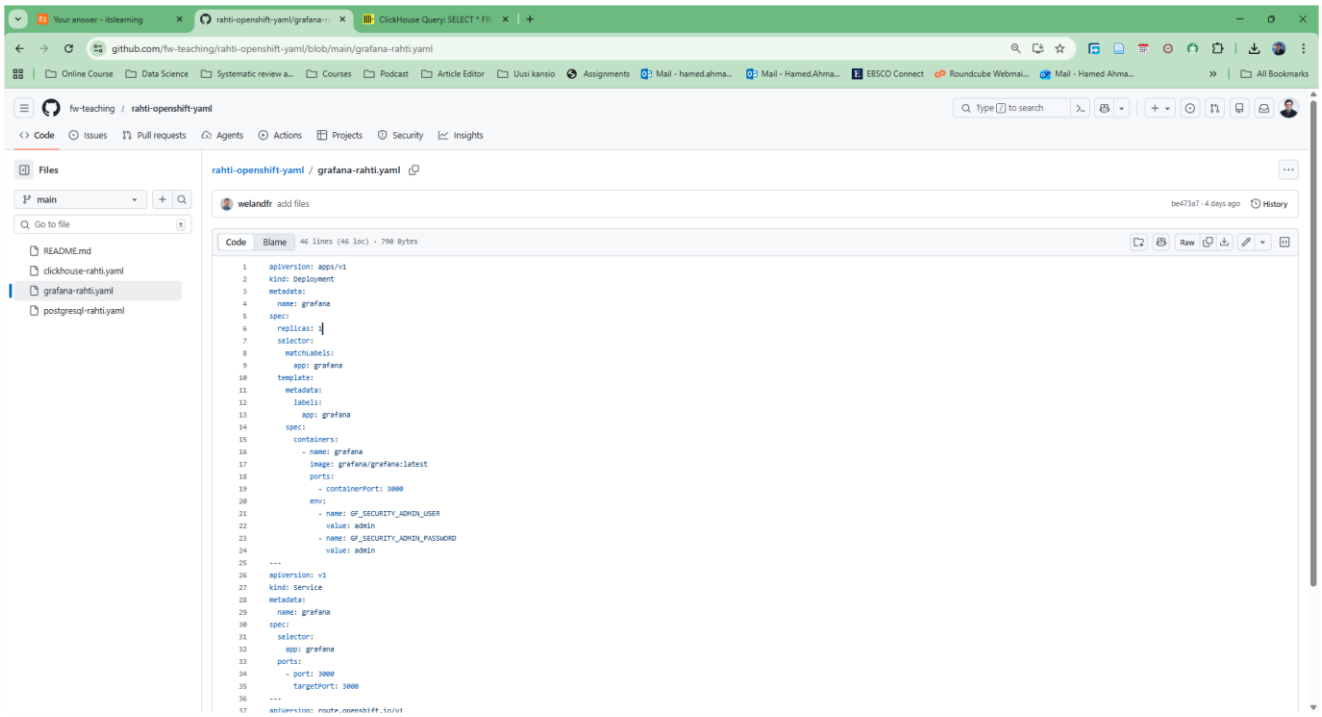
Phase D — Deploy Grafana + connect ClickHouse

In this phase, I deployed Grafana into my OpenShift project and connected it to the previously deployed ClickHouse database so that data visualization could be enabled. The goal was to make Grafana publicly accessible through a Route and verify that it could successfully communicate with ClickHouse inside the cluster. First, I deployed Grafana in my project my-assignment-6-cc using a Kubernetes YAML file. The configuration included a Deployment, a Service, and a Route. The Deployment used the image grafana/grafana:latest with one replica and exposed container port 3000. I also defined environment variables for the initial administrator account: GF_SECURITY_ADMIN_USER=admin and GF_SECURITY_ADMIN_PASSWORD=admin. After applying the YAML file using oc apply -f, OpenShift automatically created one Pod, one ClusterIP Service, and one public Route for Grafana. Next, I verified that all resources were running correctly. I ran oc get pods -l app=grafana and confirmed that the Pod status was Running (1/1 ready). Then I checked the service using oc get svc grafana and verified that it was of type ClusterIP and correctly mapped port 3000 to 3000 (TCP). After that, I ran oc get route grafana and confirmed that the Route had been created successfully. The system generated the public hostname http://grafana-my-assignment-6-cc.2.rahtiapp.fi. I also confirmed that the service had active endpoints pointing to the internal Pod IP on port 3000. These checks confirmed that the Deployment, Service, and Route were functioning correctly.

After verifying the resources, I accessed Grafana through the public Route in a web browser. The login page loaded successfully, which confirmed that the Route and Service routing were working and

that the Pod was serving HTTP traffic correctly. I logged in using the default credentials (admin / admin). On the first login, Grafana required a mandatory password change, and I updated the password successfully. This confirmed that authentication and session handling were working properly. Next, I installed and configured the ClickHouse data source plugin inside Grafana. From the Grafana interface, I navigated to Connections → Add new connection and selected ClickHouse. I installed the ClickHouse data source plugin and proceeded to configure the connection settings. For the server address, I used the internal OpenShift service DNS name: clickhouse.my-assignment-6-cc.svc.cluster.local. I set the port to 8123 (the ClickHouse HTTP interface), selected HTTP as the protocol, and left secure connection disabled. For authentication, I entered the credentials Username: clickhouse and Password: MyStrongPass123!, and set the default database to movies.

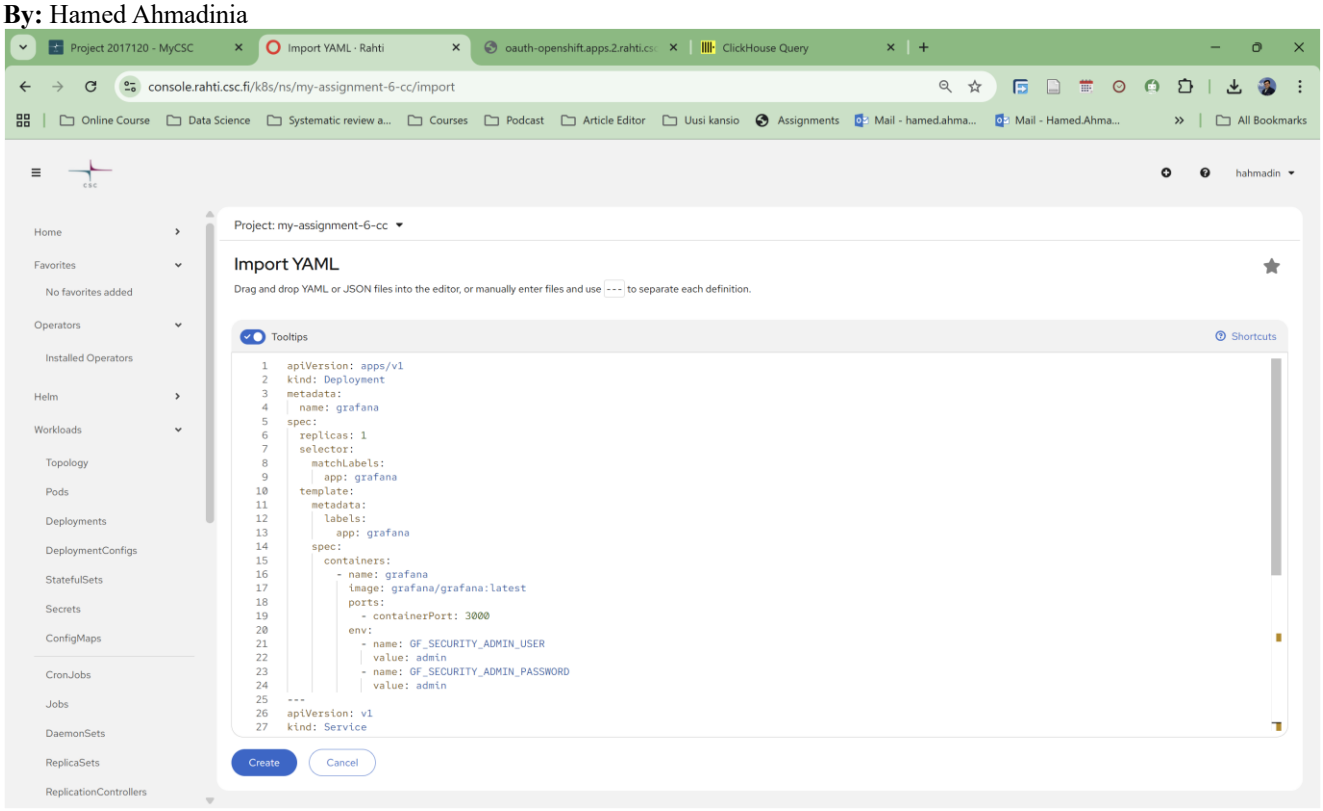
After completing the configuration, I clicked Save & Test. Grafana returned the message “Data source is working,” which confirmed that the internal cluster networking was functioning correctly, the ClickHouse HTTP interface was reachable, the credentials were valid, and the database was accessible. At the end of this phase, all verification steps were successful: the Grafana Pod was running, the Service had active endpoints, the public Route was accessible, login worked correctly, and the ClickHouse data source connection test passed without errors. Phase D was therefore completed successfully. Grafana is fully deployed, publicly accessible, properly authenticated, and correctly connected to the ClickHouse database. The system is now ready for the next phase, which is dashboard creation and data visualization.

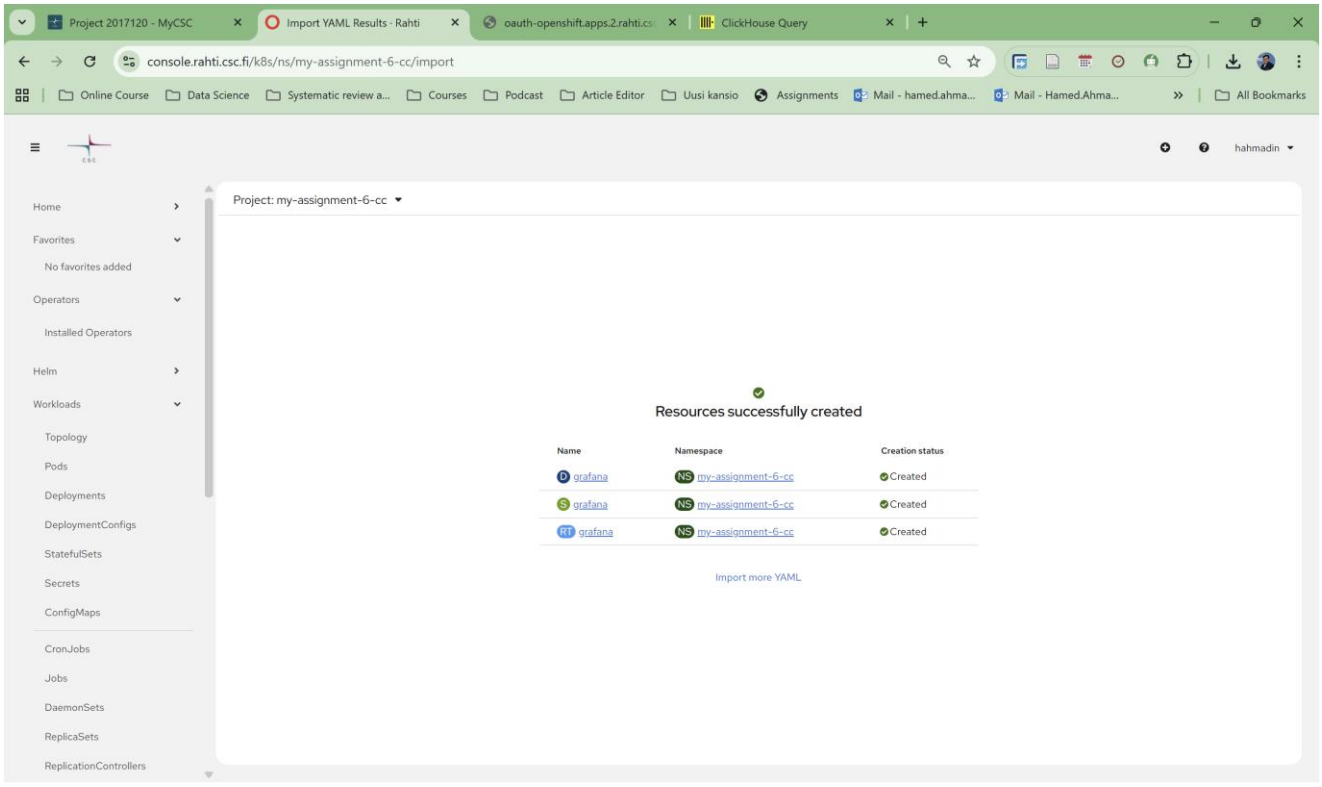


Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering

By: Hamed Ahmadinia

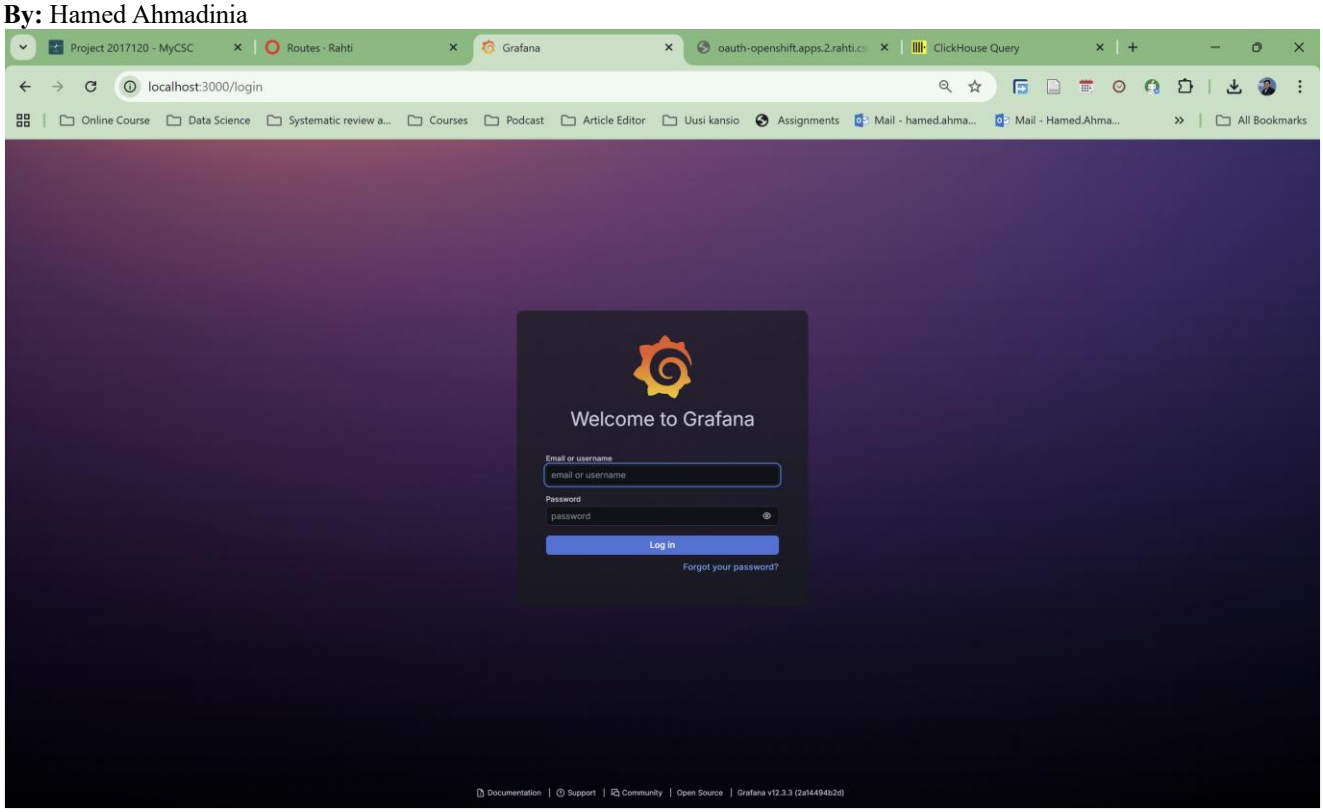


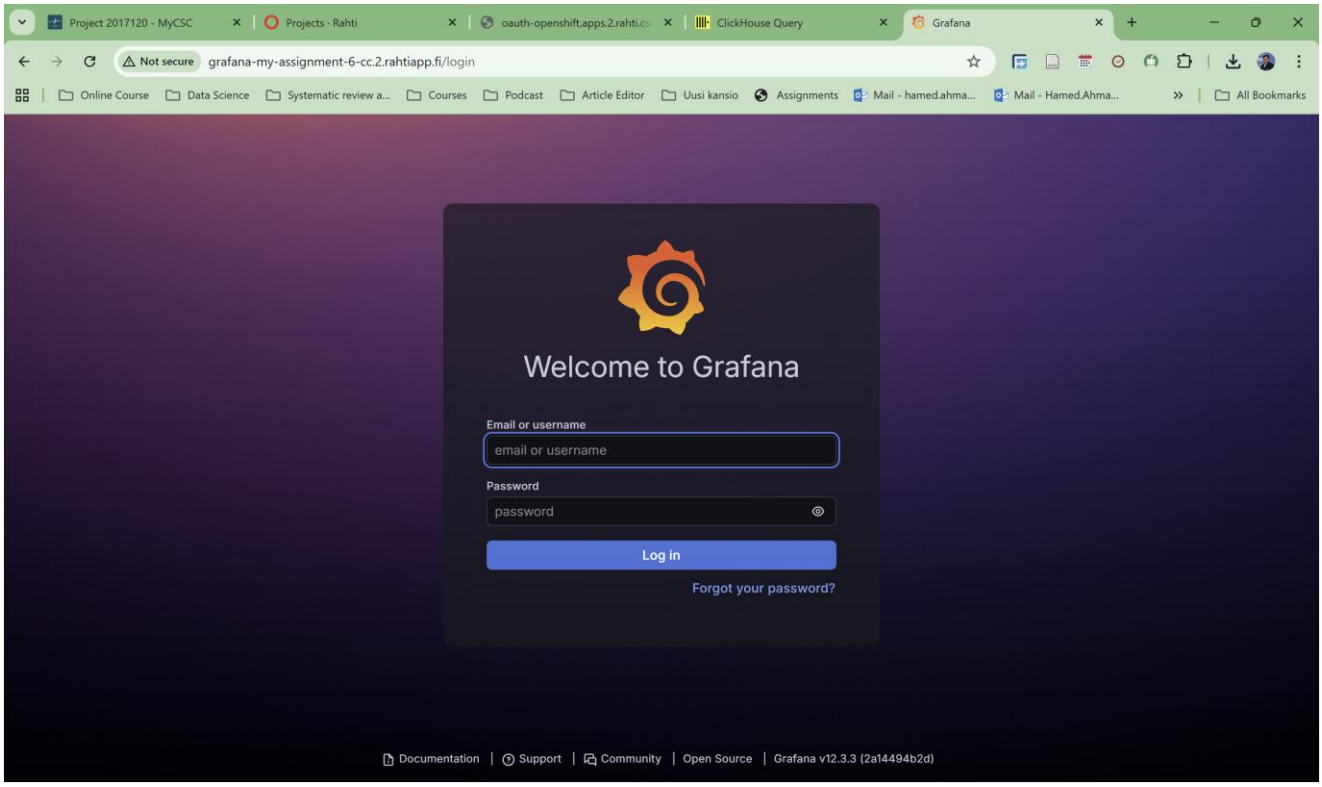


Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering

By: Hamed Ahmadinia

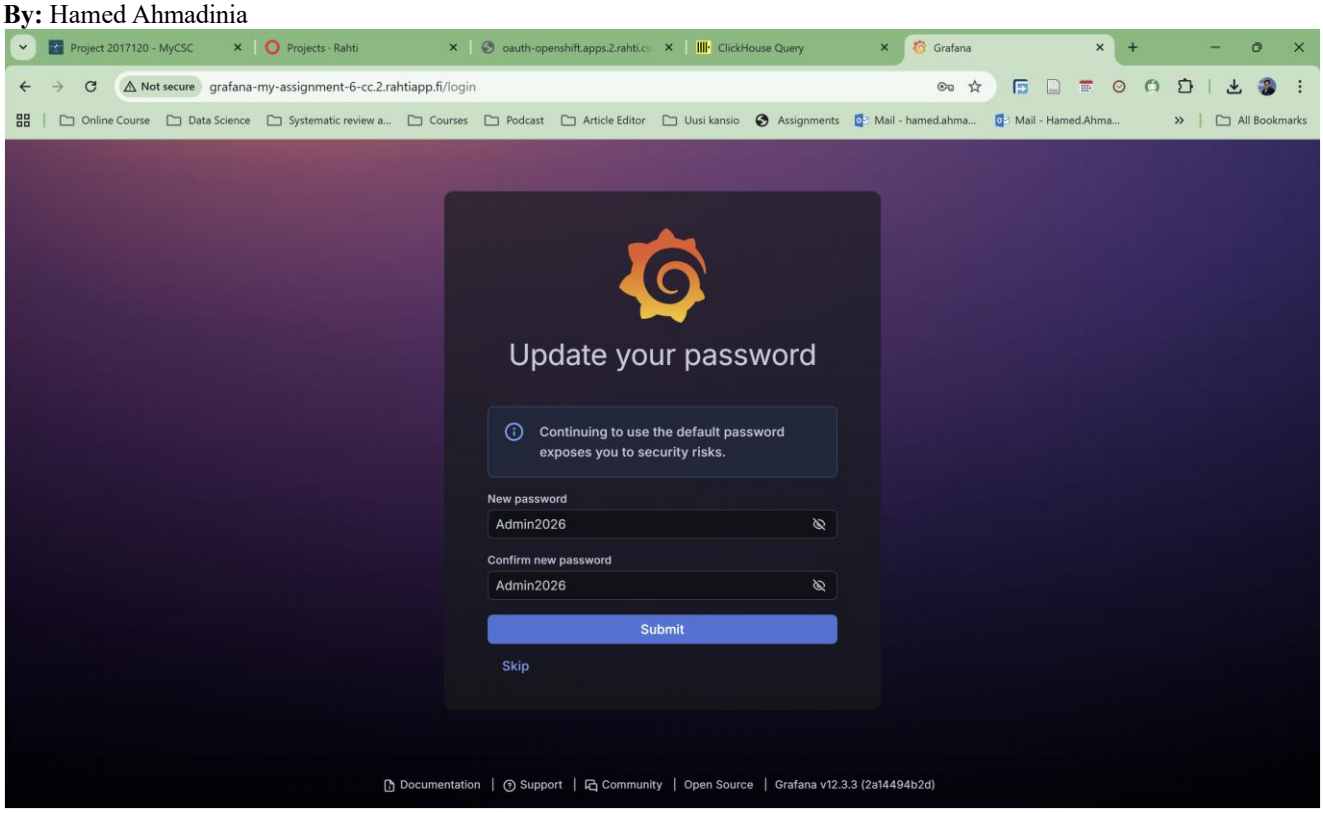


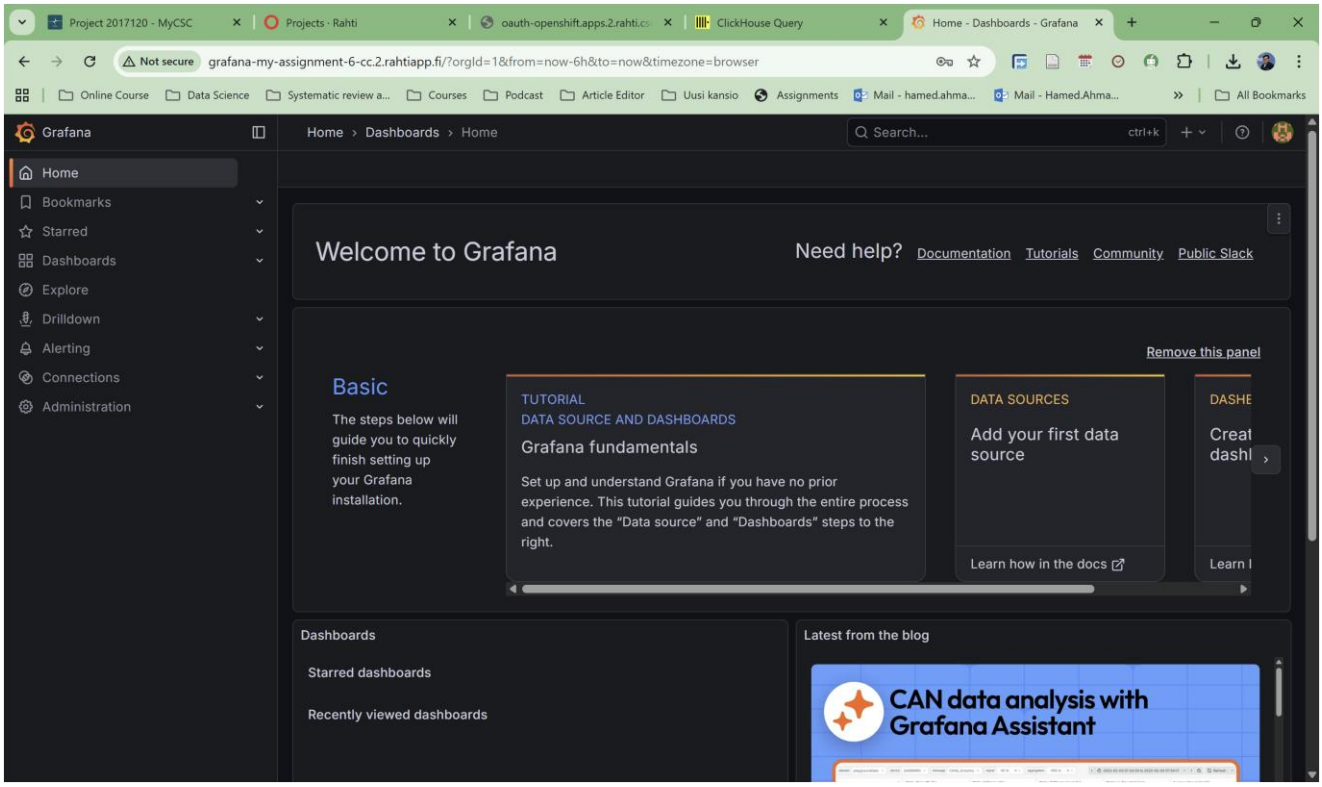


Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering

By: Hamed Ahmadinia

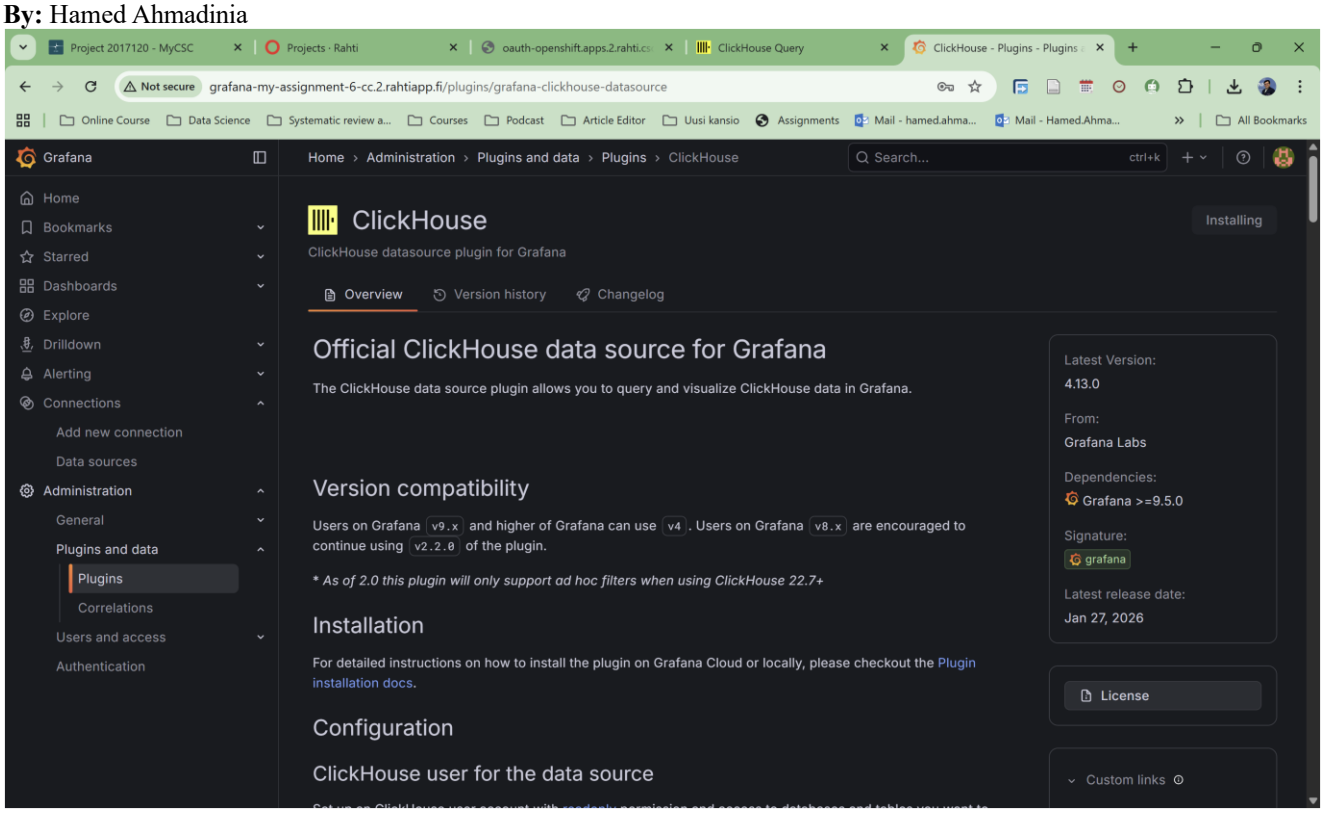


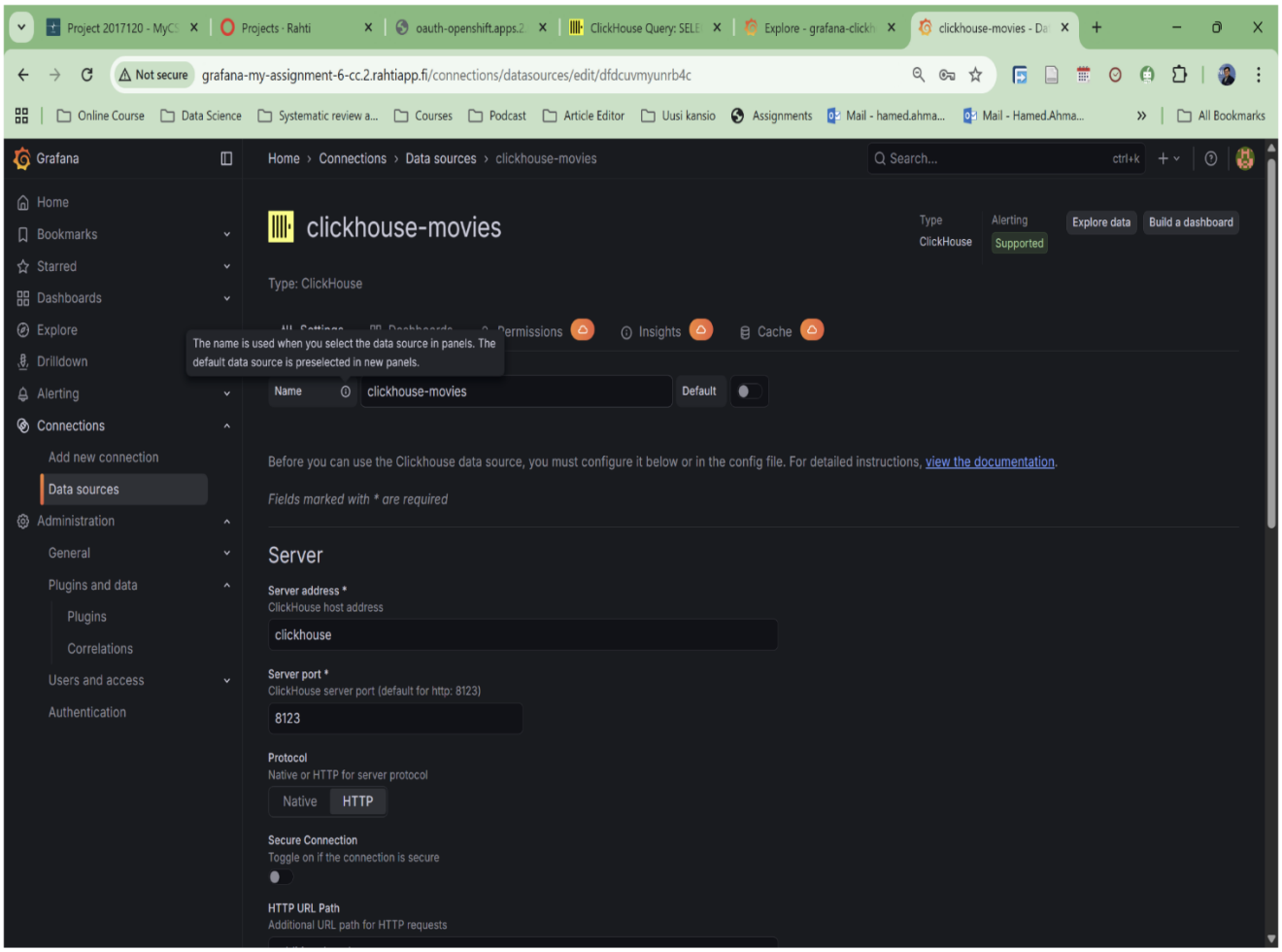


Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering

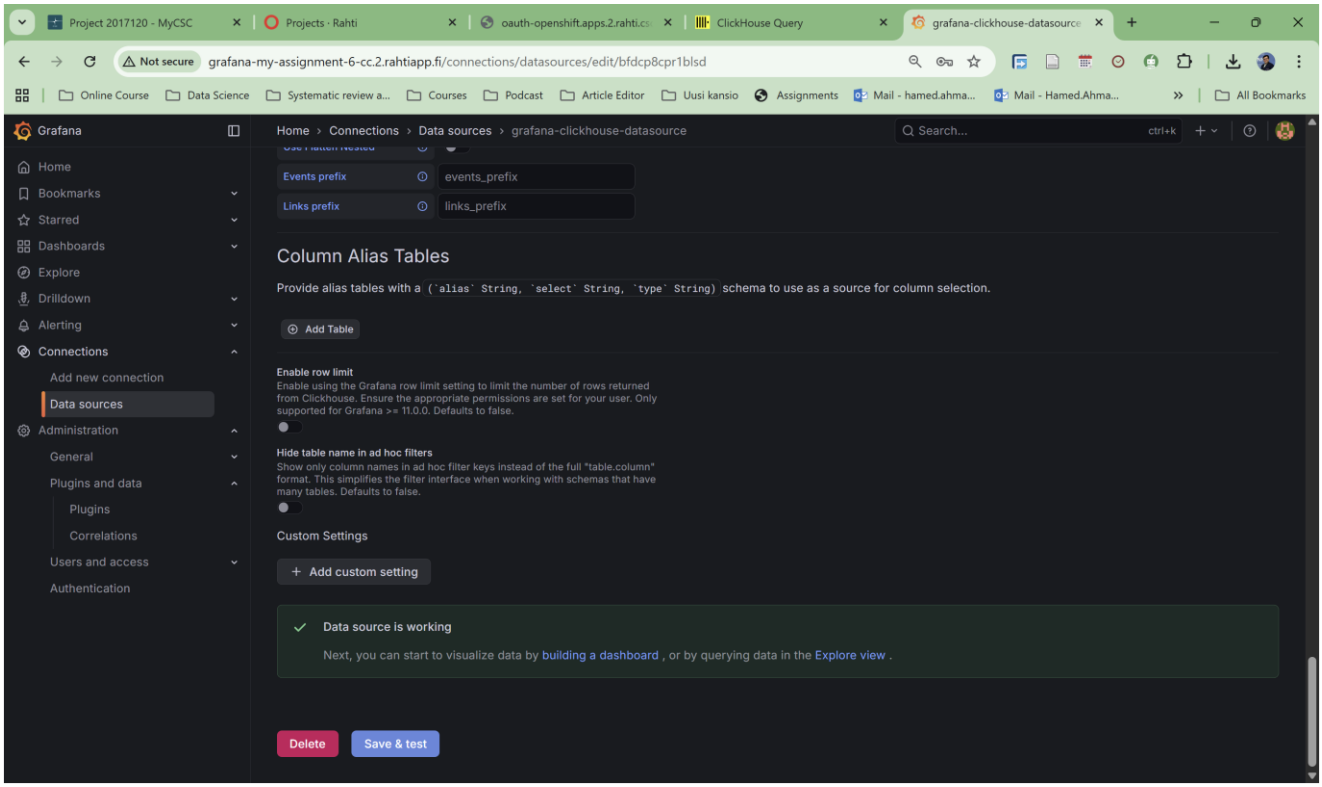
By: Hamed Ahmadinia

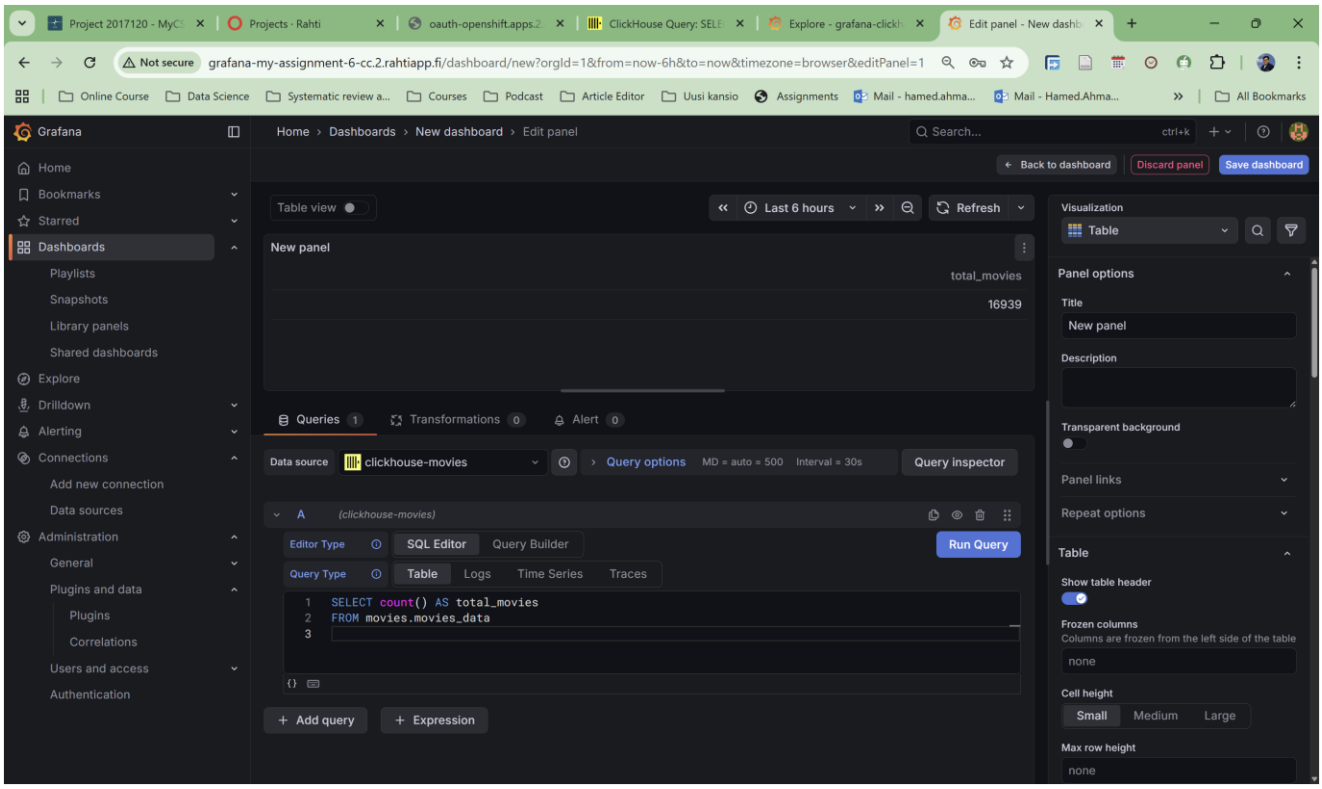




Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia





Phase F — Build Grafana Dashboard (Panel-by-Panel with Exact Queries)

In this phase, I created analytical dashboards in Grafana using the ClickHouse database as the data source. The objective was to construct meaningful visualizations from the imported movie dataset and verify that aggregation queries executed correctly within the OpenShift environment.

After confirming that the ClickHouse data source (clickhouse-movies) was operational, I created a dashboard titled **"Assignment 6 – Movies Analytics".** Multiple panels were added to visualize aggregated metrics from the movies.movies_data table.

The first panel, **Total Movies**, was implemented using a Stat visualization. The following query was used:

```
SELECT count() AS total_movies
FROM movies.movies_data
```

The query returned 16,939 records, matching the total number of imported rows. This verified successful data ingestion and correct database connectivity.

The second panel, **Movies per Year**, was implemented as a Bar chart. Since release_date was stored as a string, ClickHouse's date parsing function was used to extract the year:

```
SELECT
  toYear(parseDateTimeBestEffortOrNull(release_date)) AS year,
  count() AS movies_count
FROM movies.movies_data
WHERE parseDateTimeBestEffortOrNull(release_date) IS NOT NULL
GROUP BY year
ORDER BY year
```

This query grouped movies by release year and produced a chronological distribution of movie counts. The X-axis was configured with the year field and the Y-axis with movies_count, successfully visualizing temporal trends.

Additional panels were created using similar aggregation logic, including:

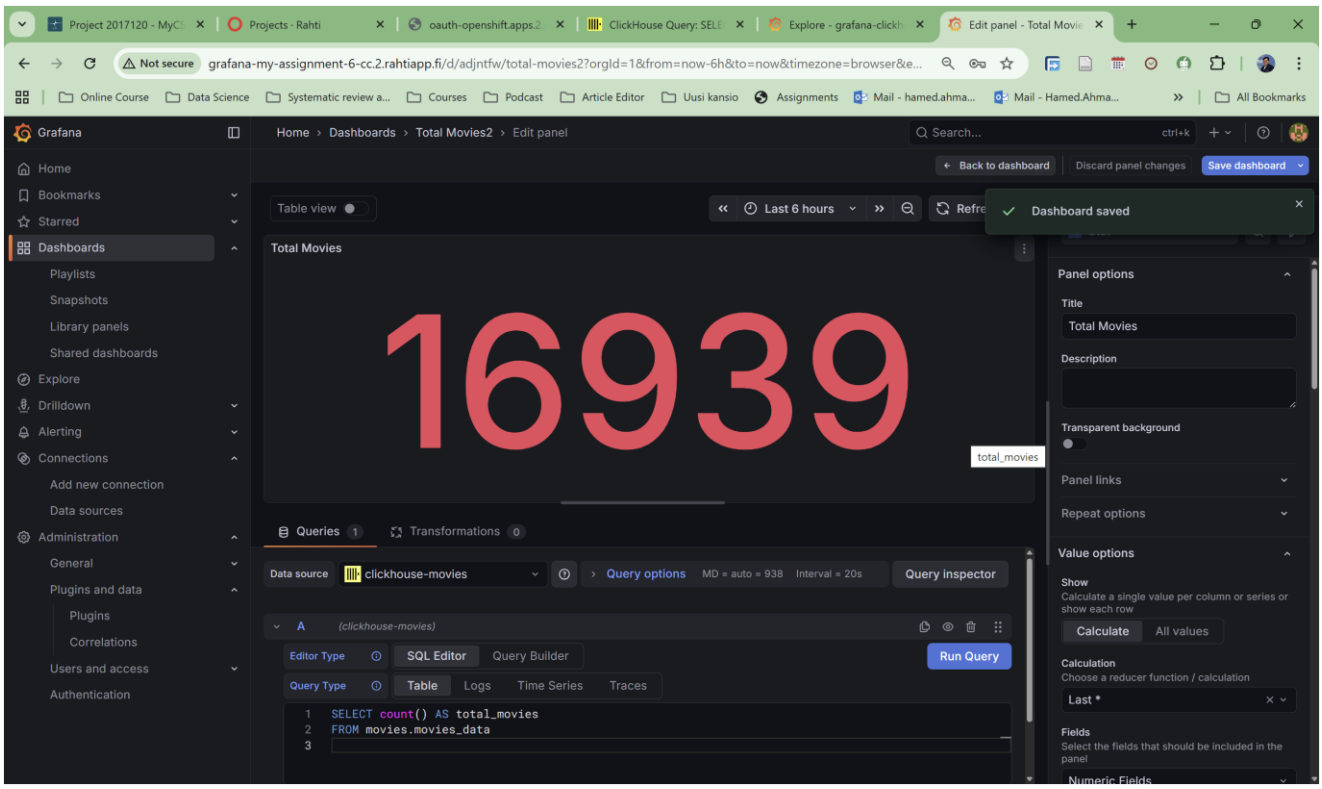
- **Average Metascore by Genre** using avg(metascore) grouped by genre.
- **Top Directors by Movie Count** using count() grouped by director with ranking.

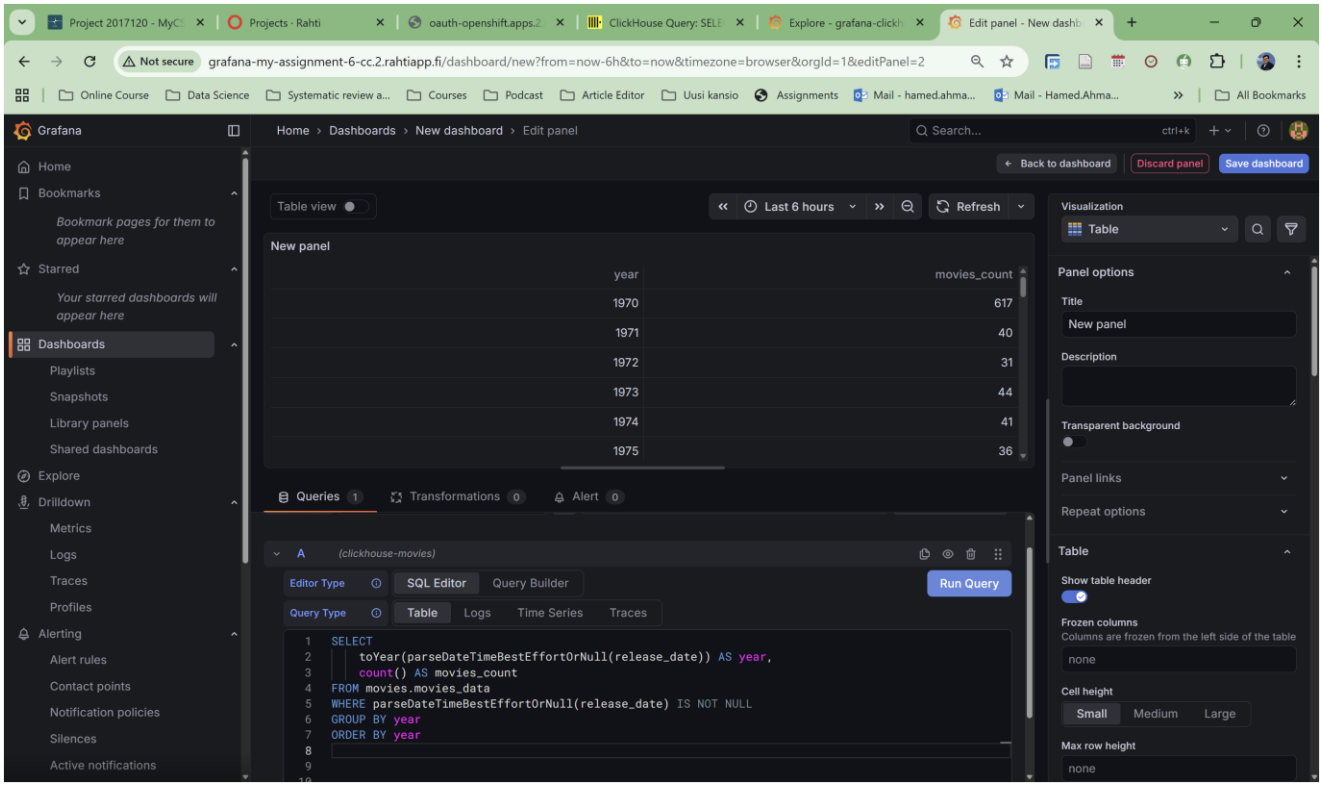
All panels executed without errors, rendered correctly, and supported dashboard interactions such as refresh and time-range selection.

Phase F was therefore completed successfully. The system now provides a fully functional cloud-native analytics dashboard built on OpenShift, ClickHouse, and Grafana.

Assignment 6: Cloud Native Visual Analytics

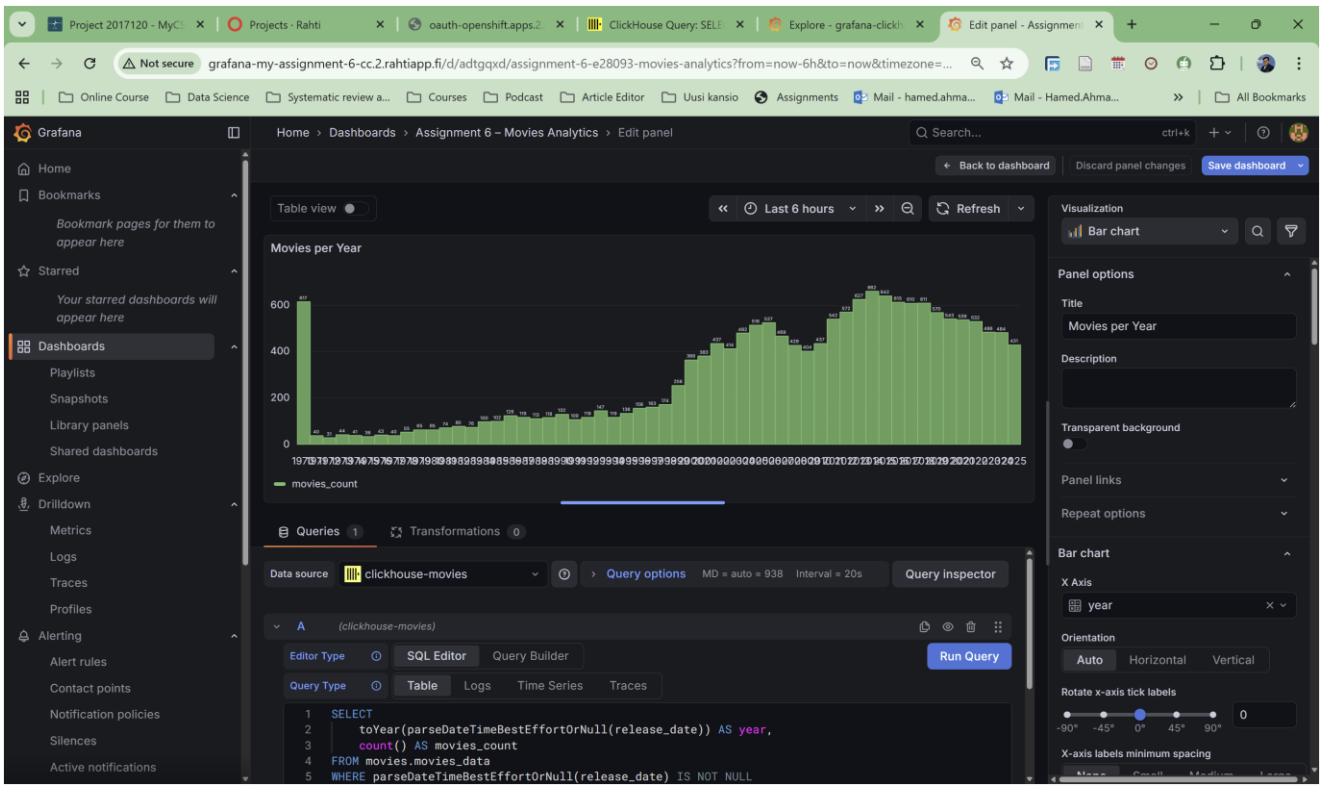
Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia

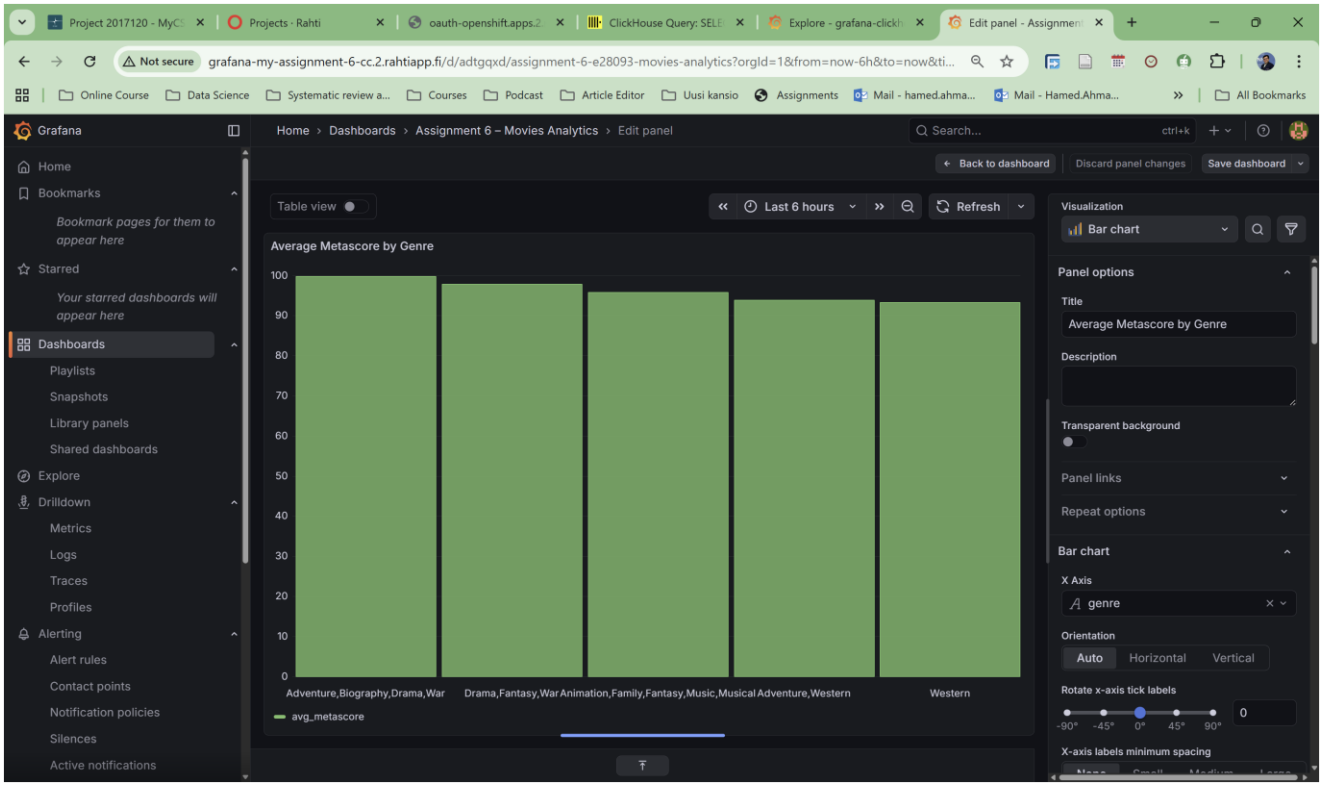




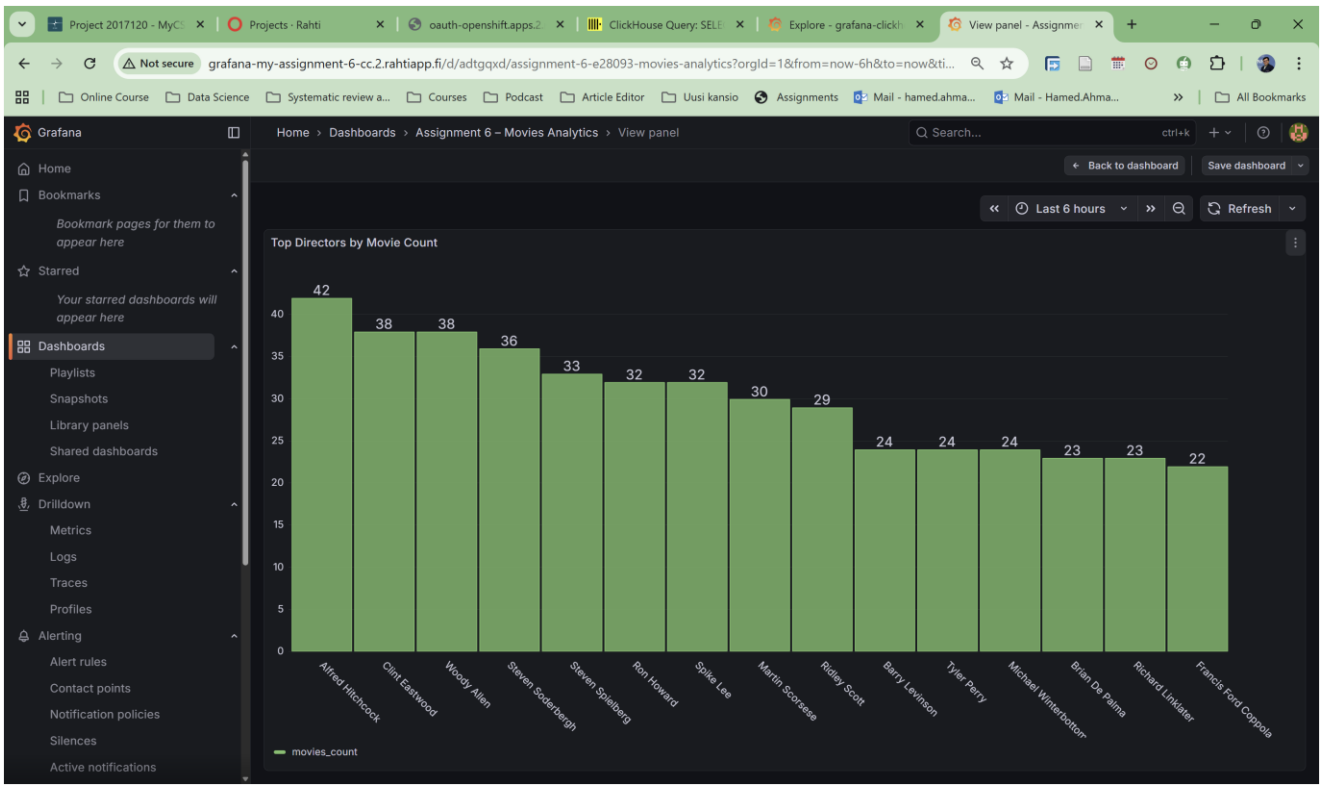
Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia





Assignment 6: Cloud Native Visual Analytics
Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia

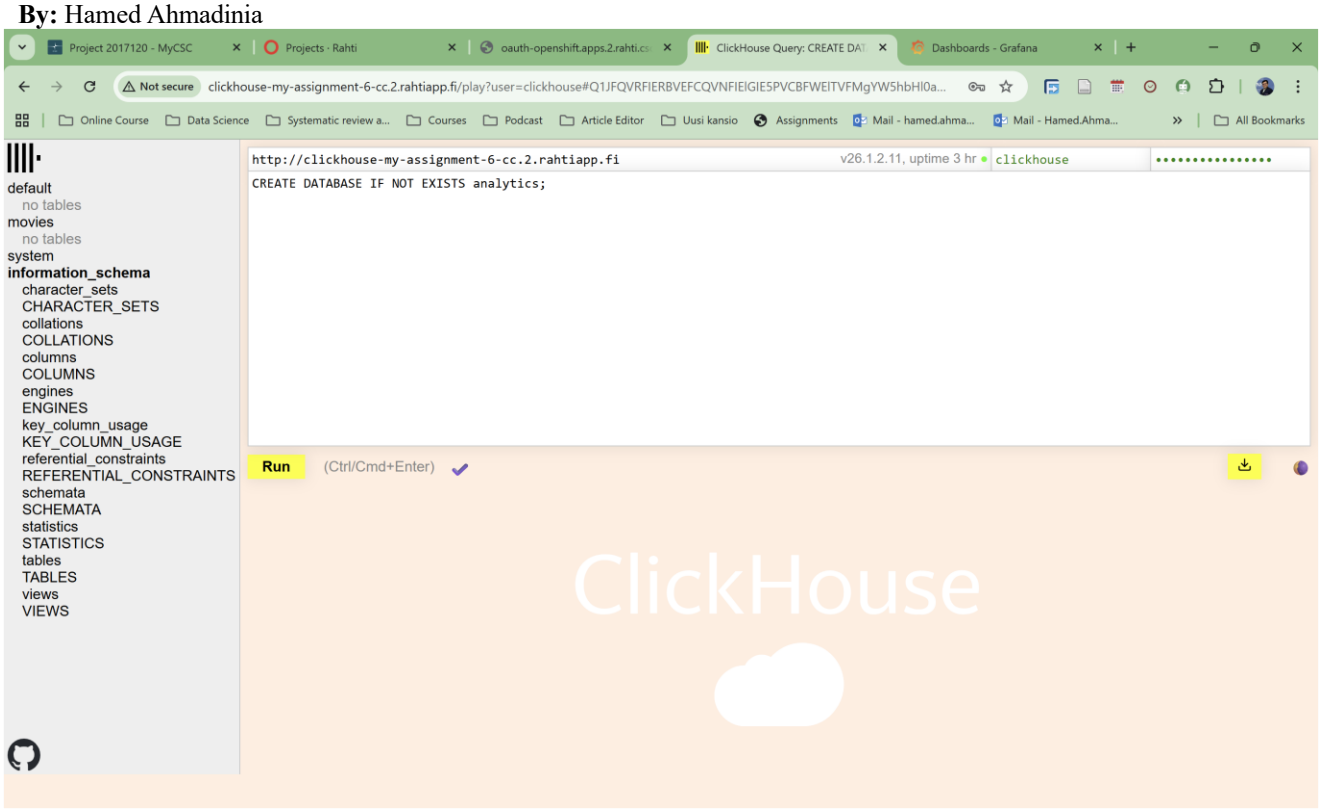


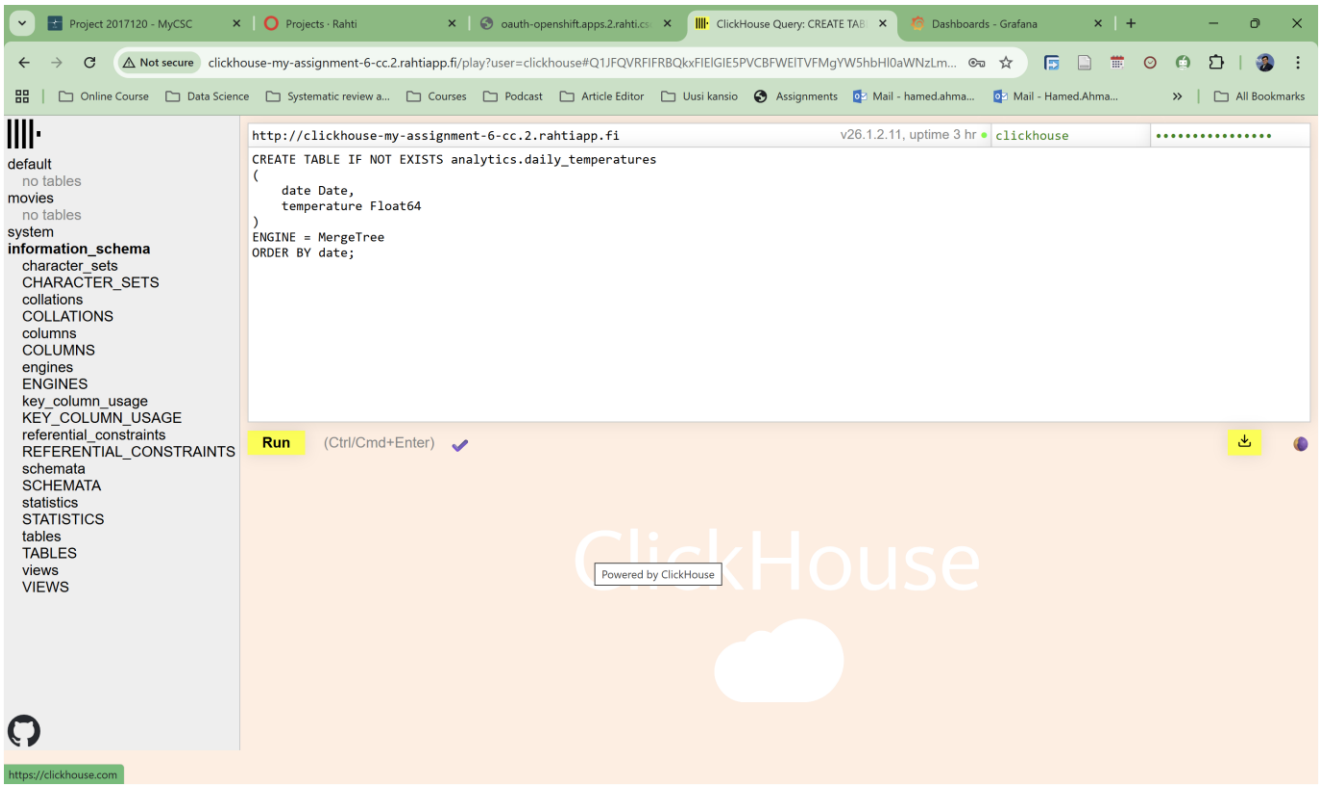
Phase G — Optional: Second Dataset and Advanced Analytics

In this optional phase, I added a second dataset to demonstrate advanced analytics using ClickHouse and Grafana. I selected a public daily minimum temperature dataset (1981–1990), containing approximately 3,650 rows, which fits the resource limits of CSC Rahti while still enabling meaningful time-series analysis. I created a new database (analytics) and a MergeTree table ordered by date, then imported the CSV directly from a public URL using ClickHouse's url() function. After verifying the row count and sample records, I created a second Grafana dashboard titled *Temperature Analytics Dashboard*. This dashboard includes a daily time-series panel, a monthly average temperature panel (time-based aggregation), a yearly temperature range panel (max–min derived metric), and a 90th percentile temperature panel using ClickHouse's quantile(0.9) function (advanced statistical metric). These panels demonstrate time aggregation, group-by operations, and derived metrics, fully satisfying the optional assignment requirements while maintaining efficient resource usage within the same OpenShift deployment.

Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia

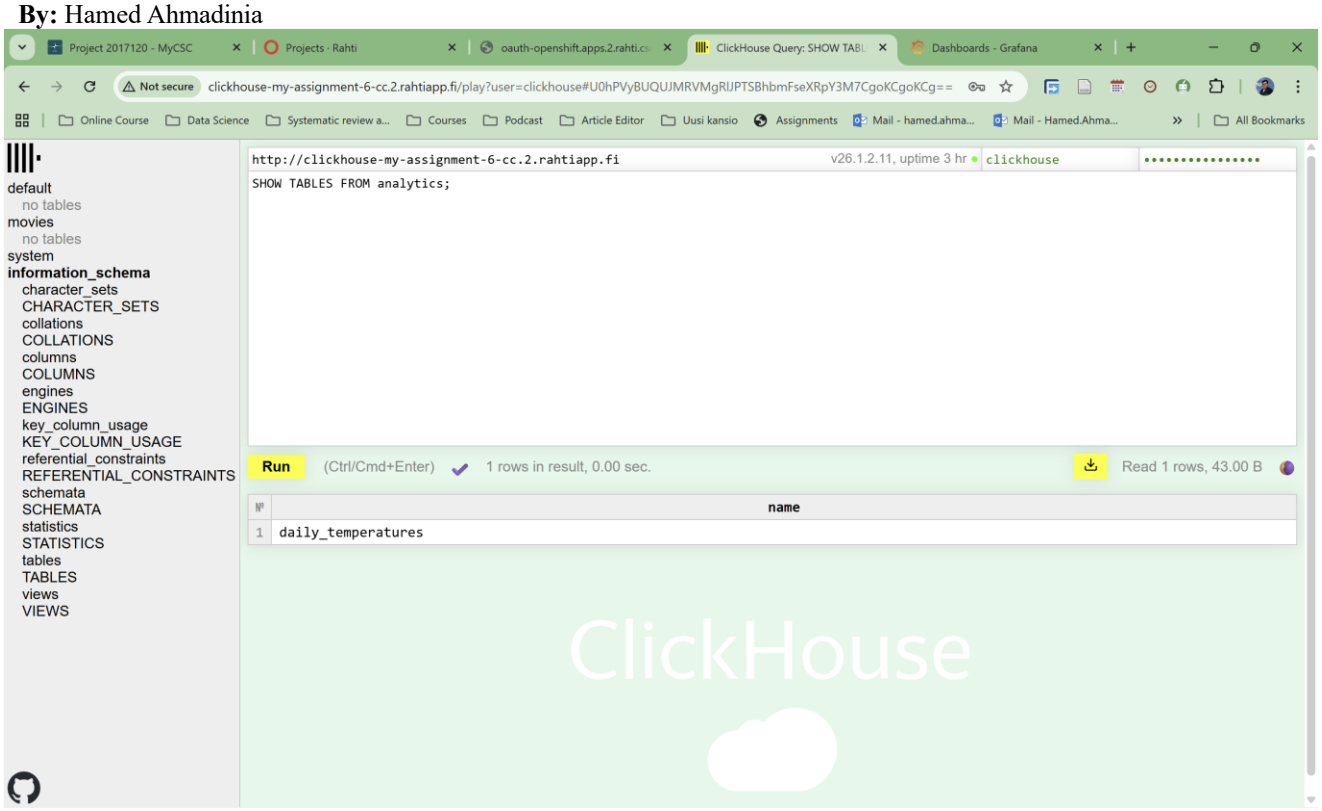


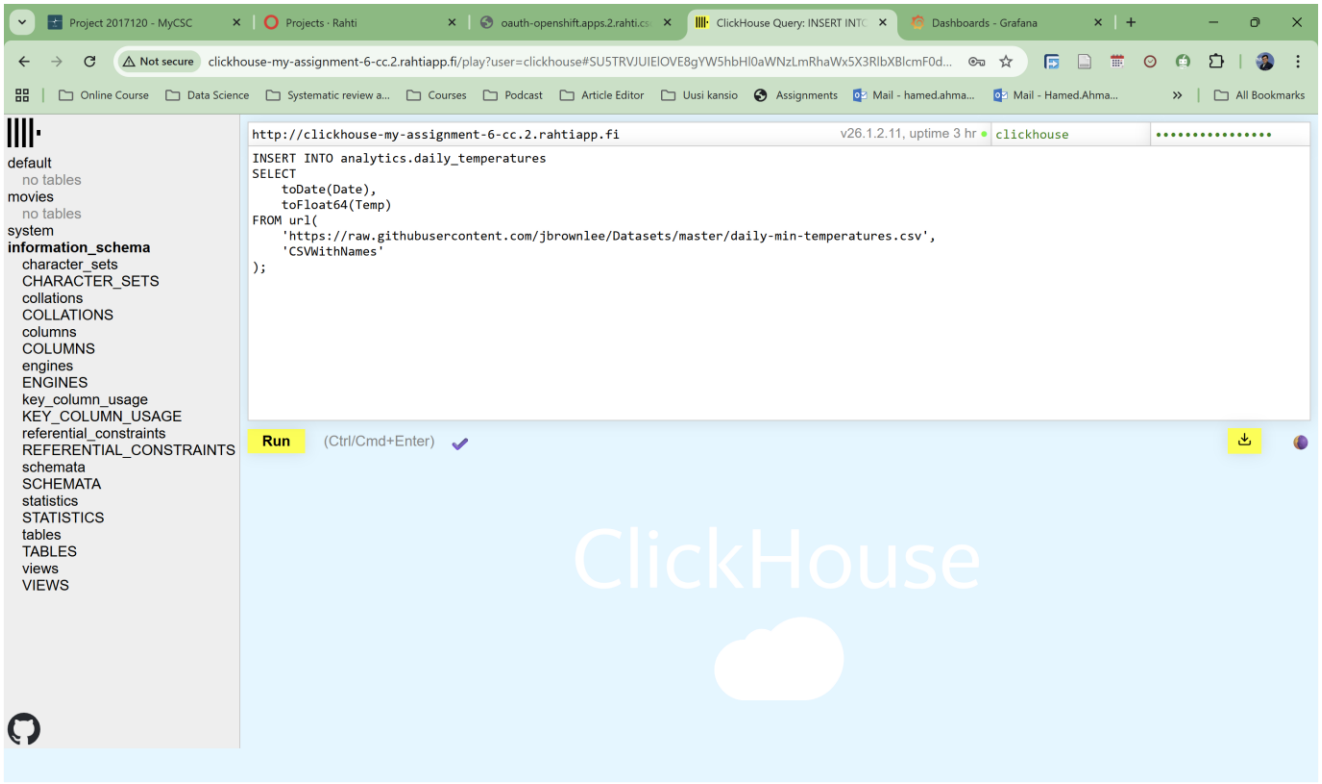


Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering

By: Hamed Ahmadinia

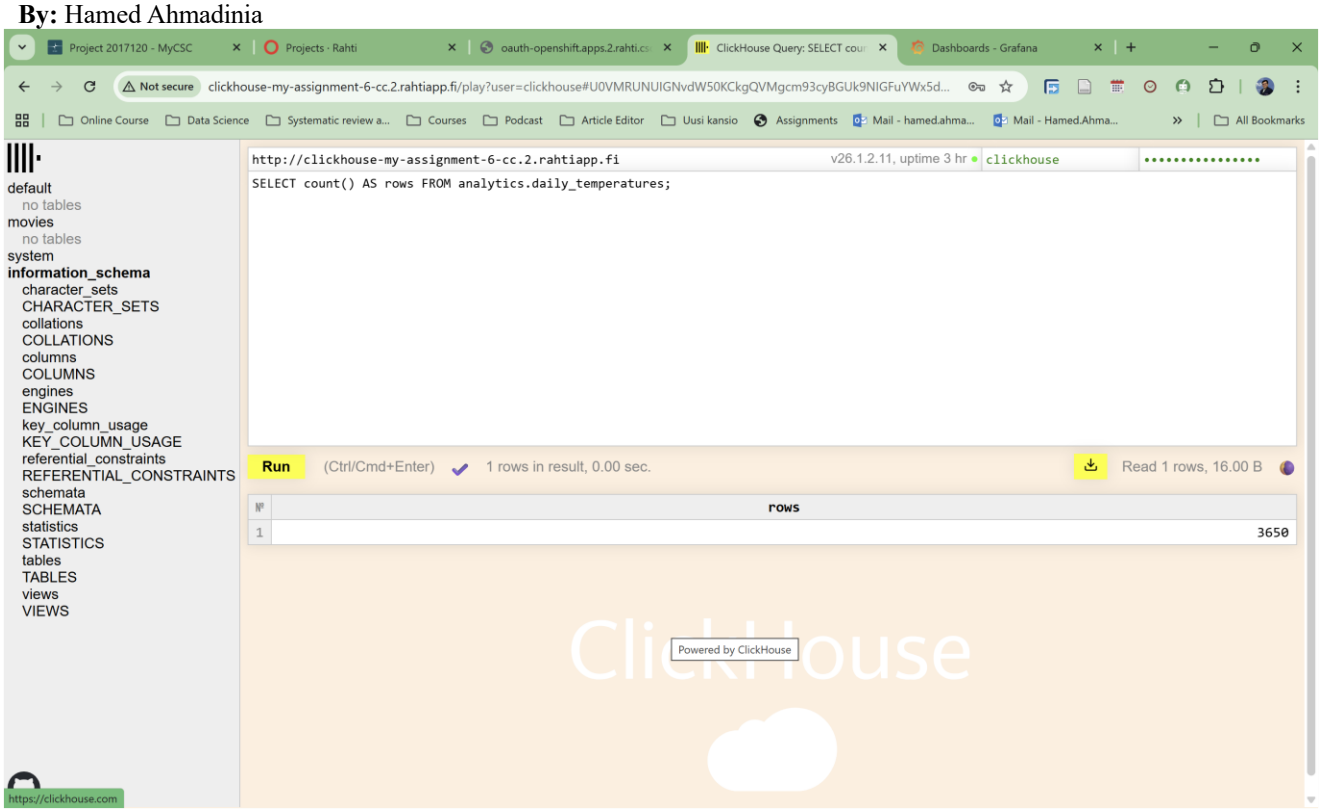


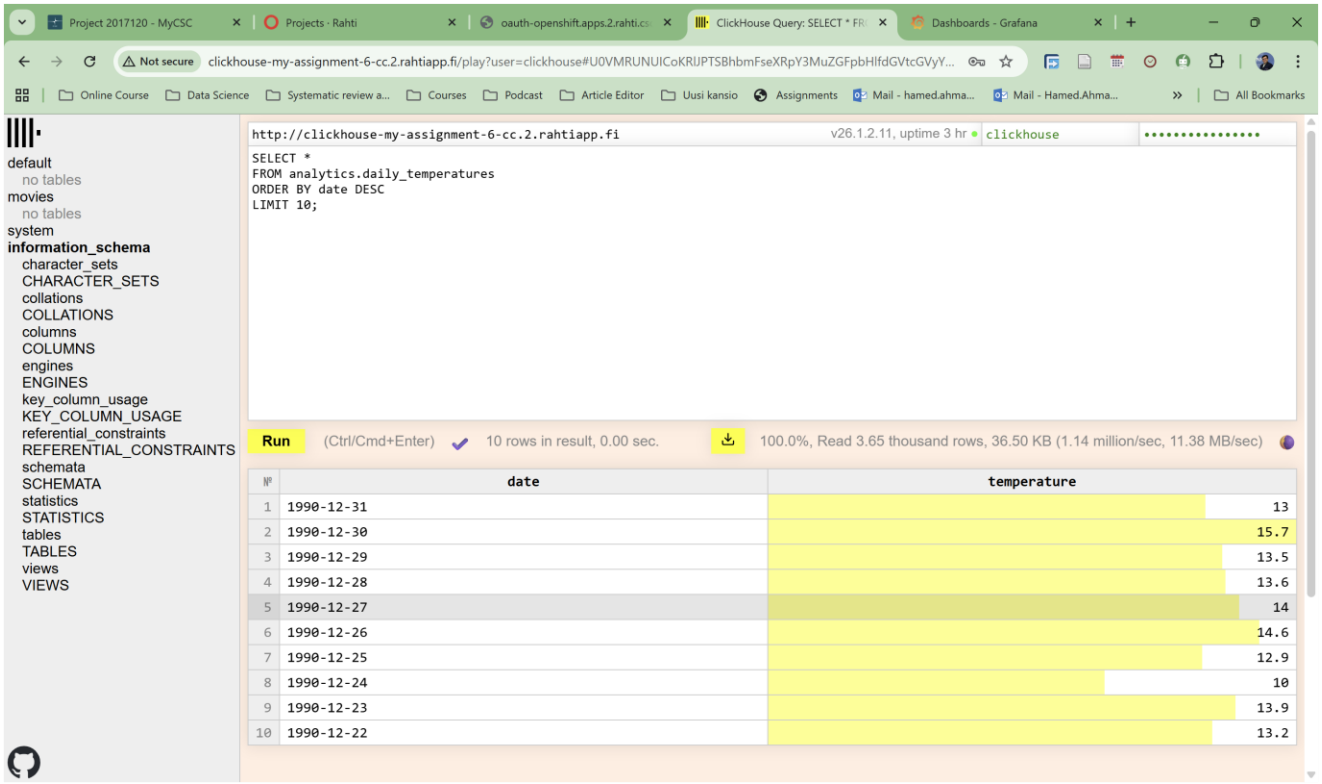


Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering

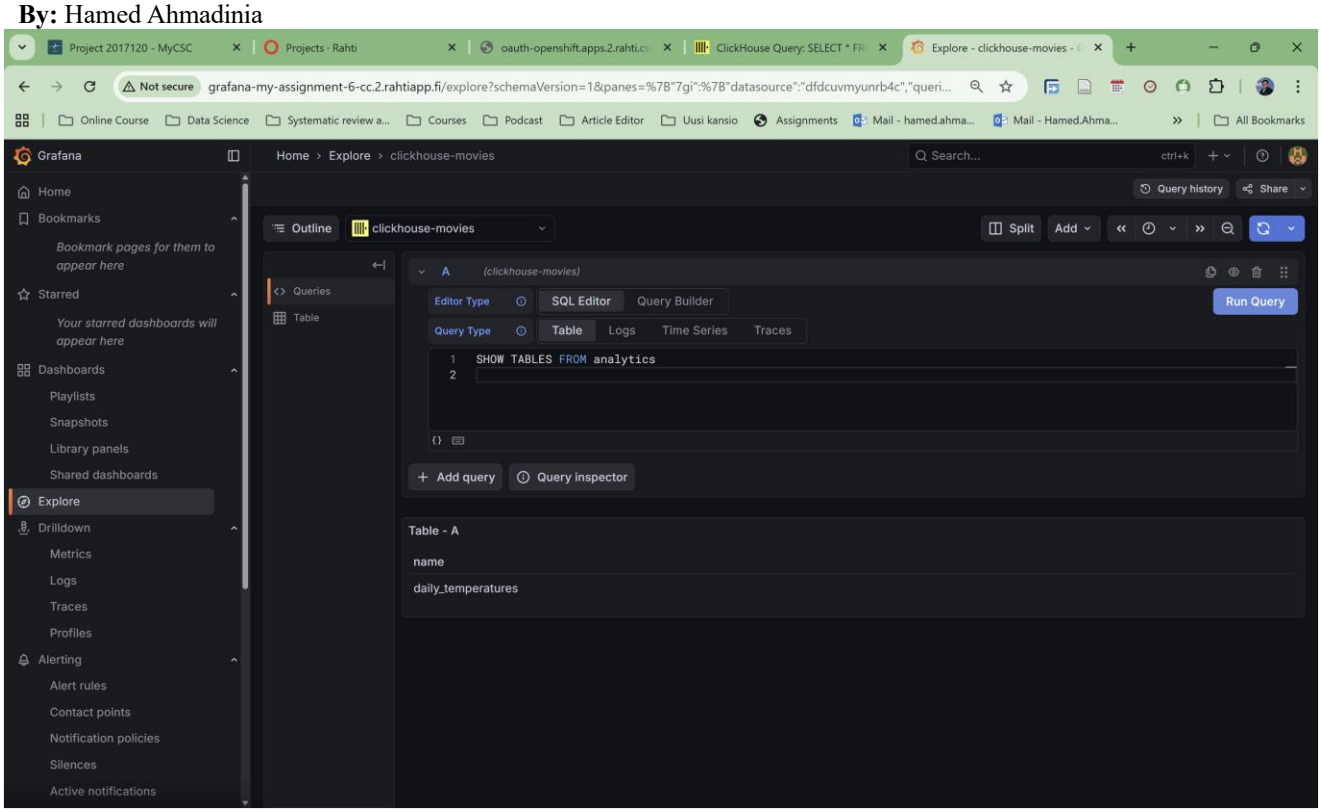
By: Hamed Ahmadinia

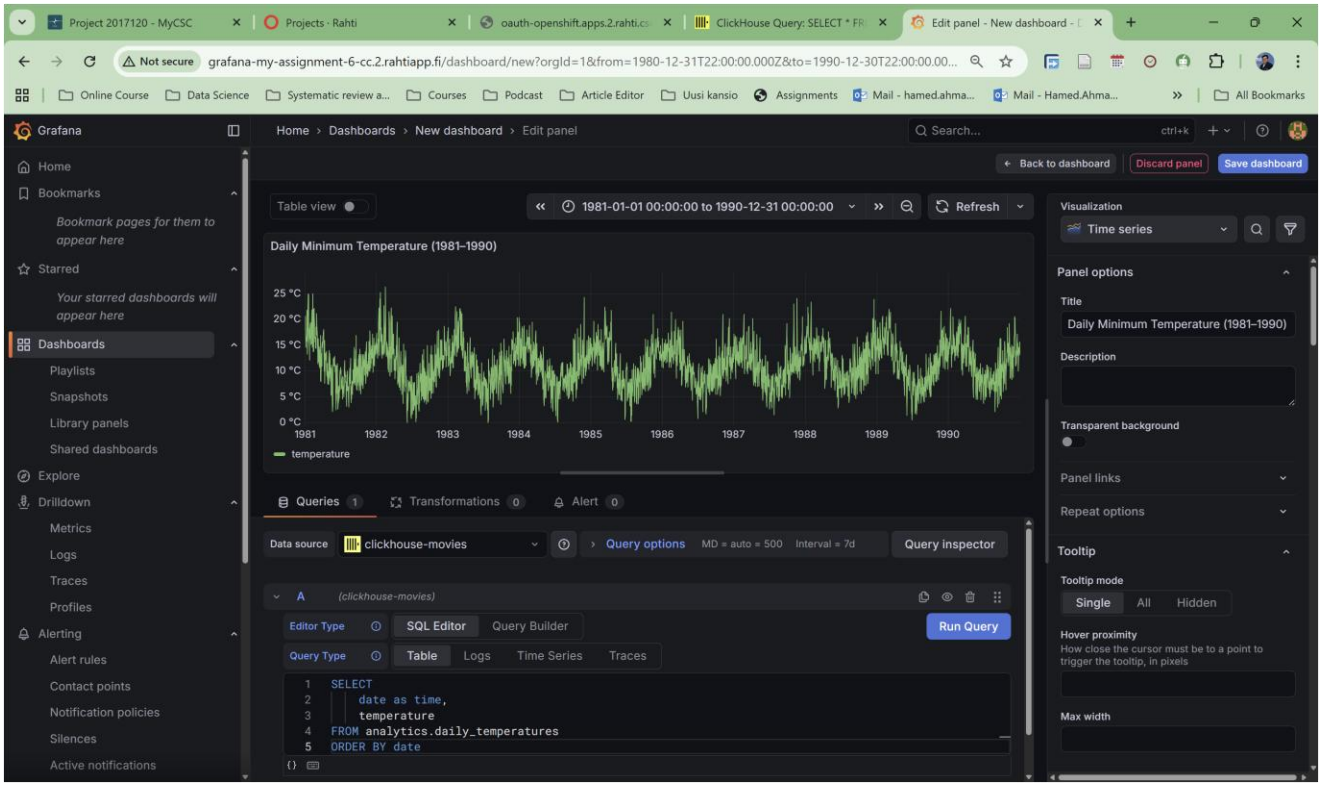




Assignment 6: Cloud Native Visual Analytics

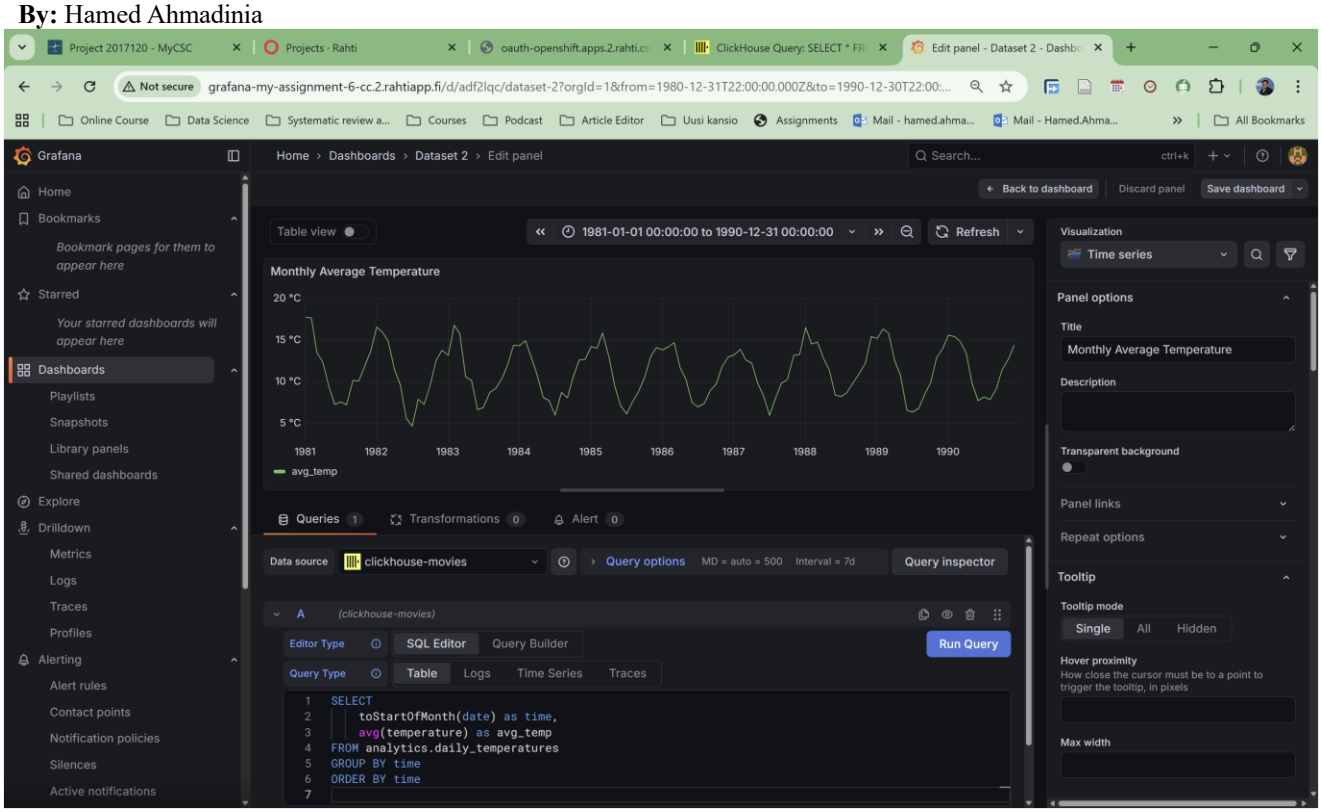
Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia

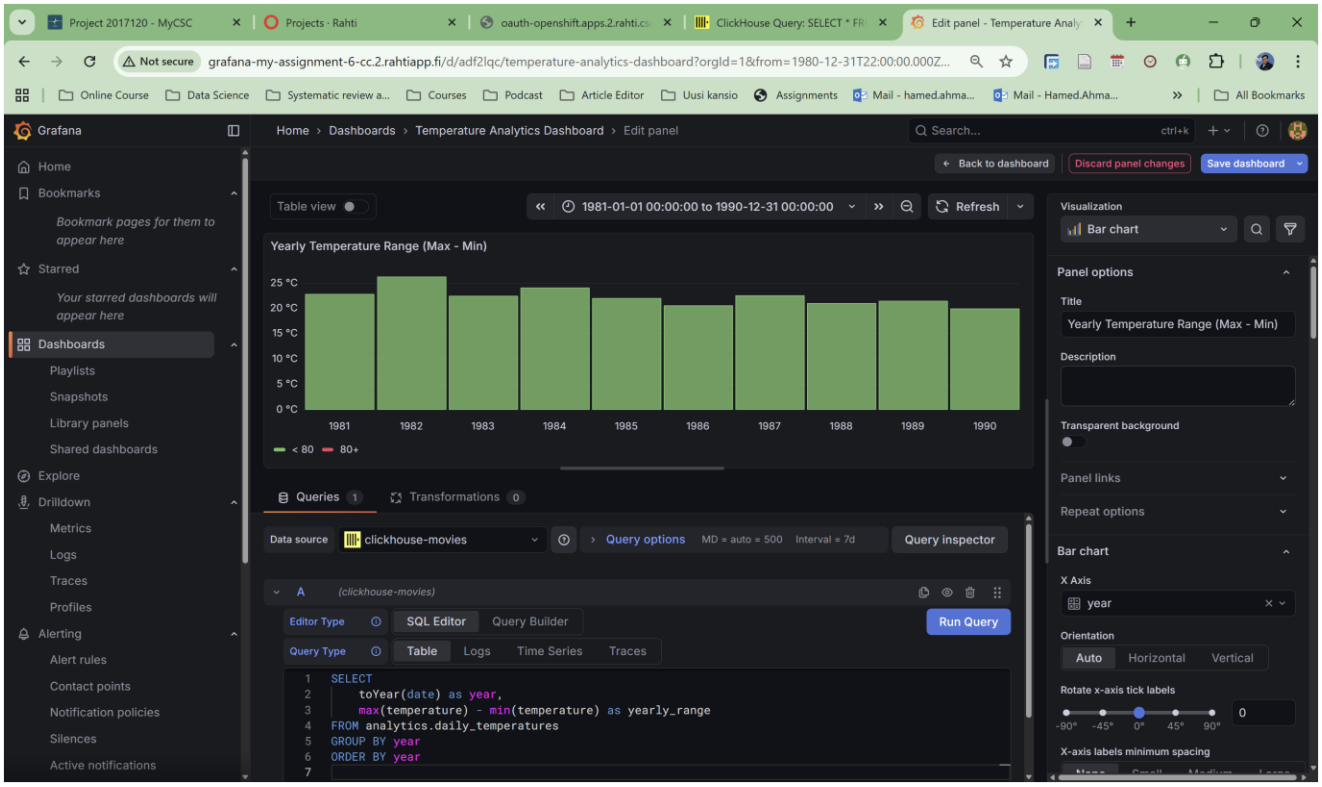




Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia





Assignment 6: Cloud Native Visual Analytics

Course: Cloud Computing and Data Engineering
By: Hamed Ahmadinia