Development of infrastructure

For the test task EastRockValley - steps of the development.

The task description

- Accepted assumptions:
- Realization:
- S3
- Airflow
- MySQL
- Metabase (Reporting)

Accepted assumptions:

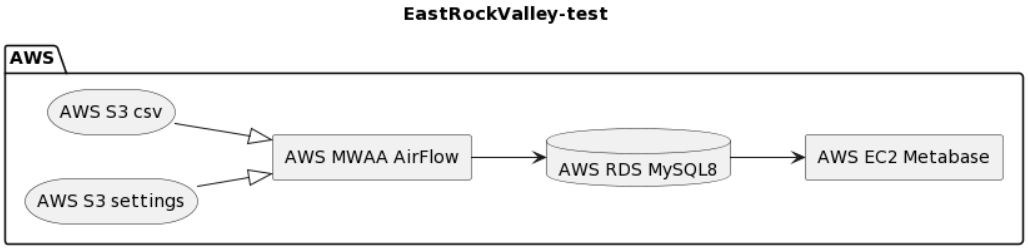
Based on the task, we operate with incoming data.

1. Apparently, this is not the original data on transactions and actions of player users. These are the amounts of time that players spend in a particular game, already aggregated for a date (day). We do not know anything about bets or gameplay results. In another file, we see player profiles. In the third file, we see financial transactions. Since Transaction ID is present, it is no longer possible to call this data aggregated to the date (day). So, we assume that information about the exact time and order of transactions is lost or not important. Let's assume the UserID in files 'games' and 'payments' contains the UserID which was imported before or exists in the current file 'users'.
2. Let us assume that these three files are all that we know about the data to be processed.
3. Let's assume that this data appears in our online storage in the form of CSV files.

Realization:

Chosen platform and software: AWS S3, MySQL 8 RDS, AirFlow (MWAA and Docker), Metabase (EC2)

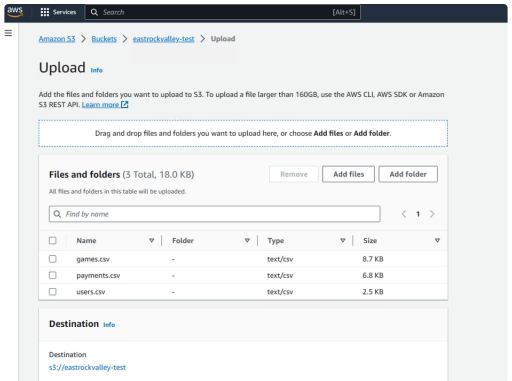
Software: IE (Chrome), Github, DataGrip, PyCharm, VSCode



common UML schema

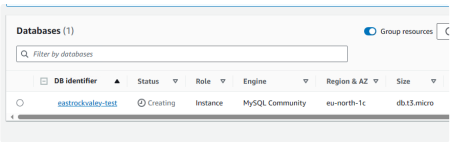
S3

Create the S3 storage for the incoming files:



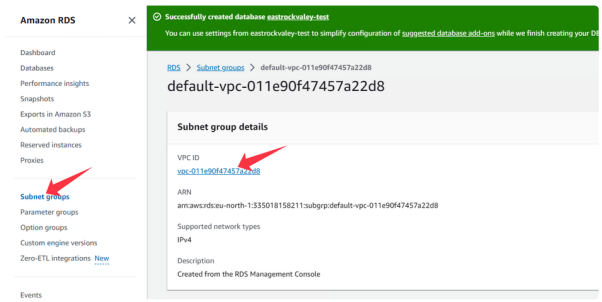
3 files uploaded

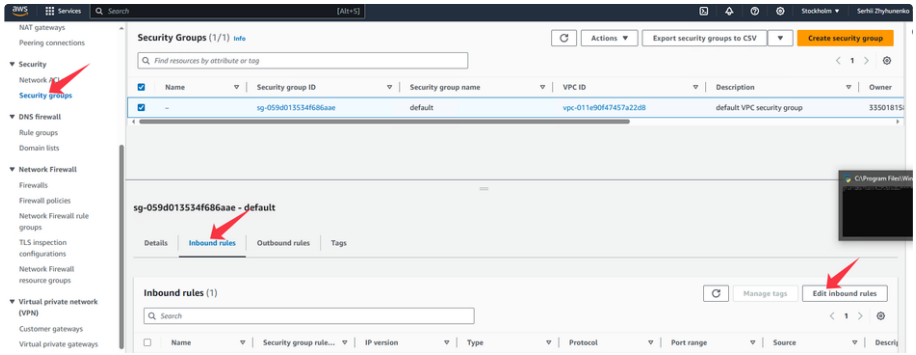
Create the RDS database in AWS. For this example, I created the MySQL database

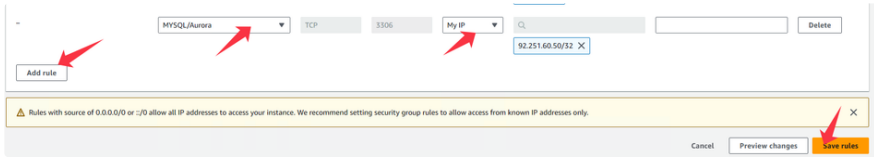


MySQL 8, inside AWS. No special settings

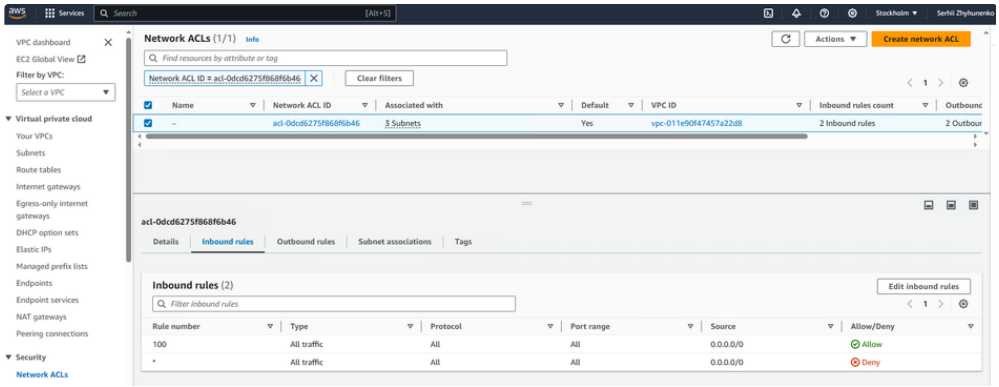
For the new database, need to open inbound rules to operate it remotely:

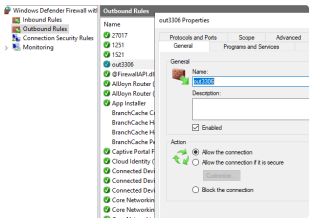






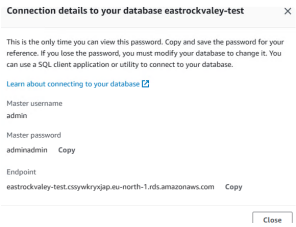
Should be set for whitelisted IP of the company's office for sure





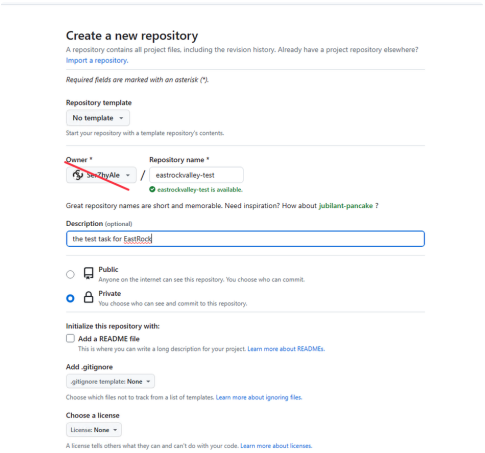
well I have to open the MySQL port in my firewall as well

The database was created and I can note its creds:

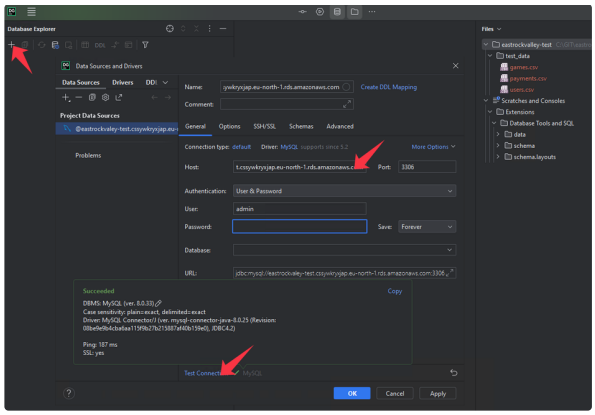


simple password was taken for example

Before doing any code, let's create the project in GitHub for this:

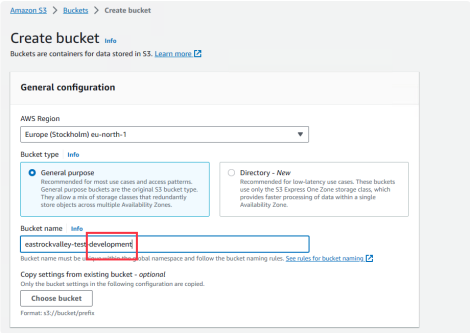


Now everything is ready to create the project and connect my DataGrip to the new database and Github:

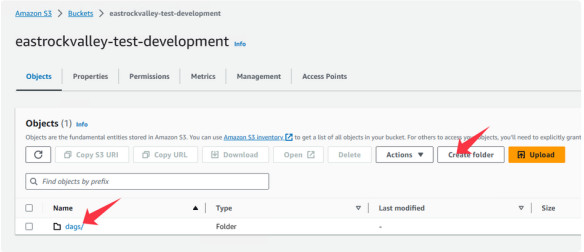


Before MySQL code, will put here all the other files of the project

Before creating AirFlow instance in AWS we need to create another bucket for the code of DAGs:



I did it again in S3



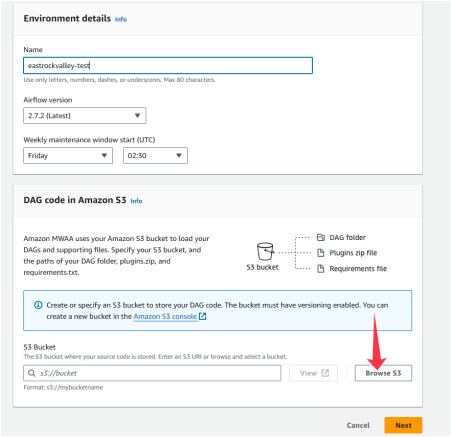
and folders in it (dags, plugins)

Airflow

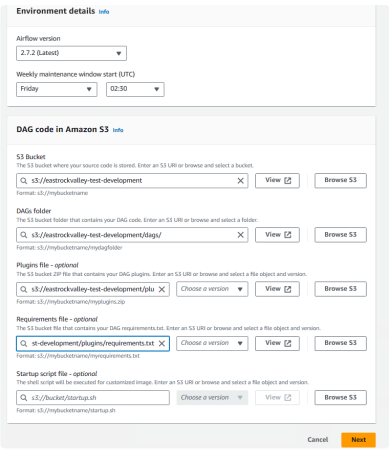
Create file `requirements.txt with` two additional Python packages:

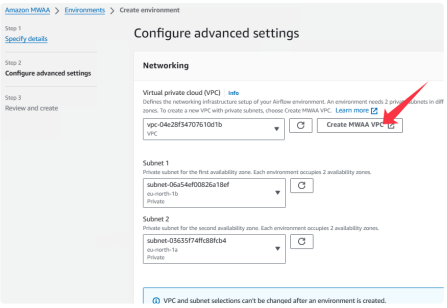
download from the site pandas last wheel, zip it into pligins.zip, upload it in the development bucket, and select for airflow (not on images)

Later I decided to use airflow.providers.amazon.aws.transfers.s3_to_sql instead



Browse the created before DAG bucket and requirments.txt





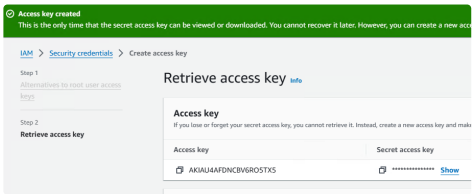
The creation of a cheap airflow instance can take a lot of time on AWS side. Have to wait before it's available.

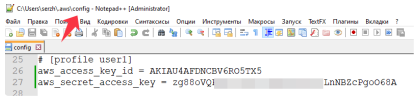
While it is going, we can prepare the DAG.

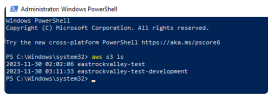
Prepare the Python on a local computer:
(full code in GIT env.txt)

Install or update the latest version of the AWS CLI - AWS Command Line Interface (amazon.com)

Create and retrieve and set AWS access keys on the local computer:

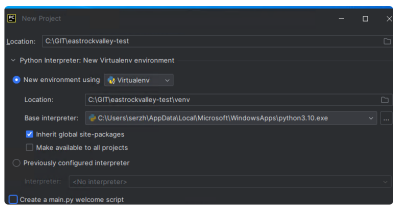




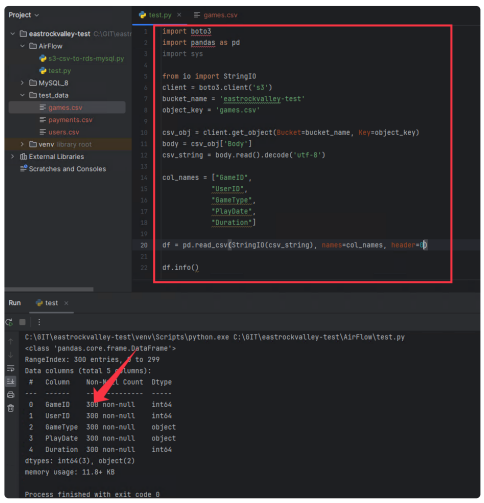


check the s3 is available
from aws cli

Open PyCharm and create a new project in the same folder

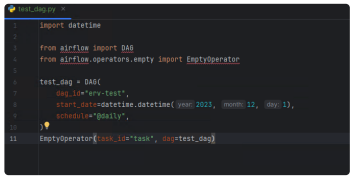


We can create a small test app to check the data availability:



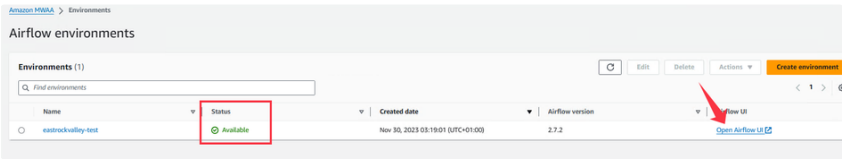
300 is a number of rows of data are available

Also, we can create some test DAG .py file



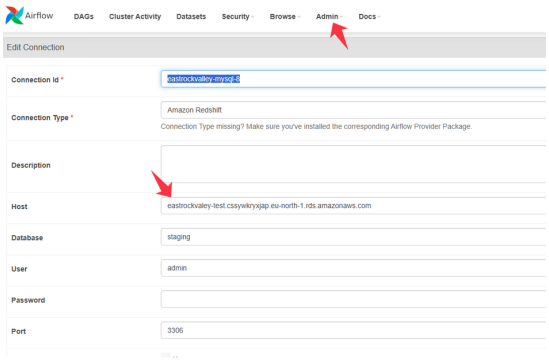
and put it into Dags folder (bucket for development)

Once Airflow is ready, we can open its user interface

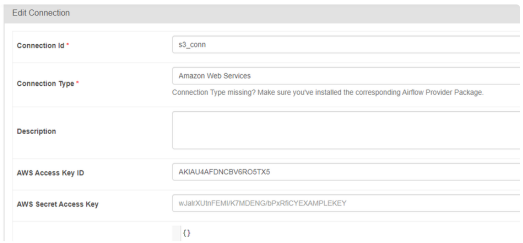


and we can see the test DAG can be triggered

In AirFlow UI Go to Admin - connections add next two:



connection to MySQL



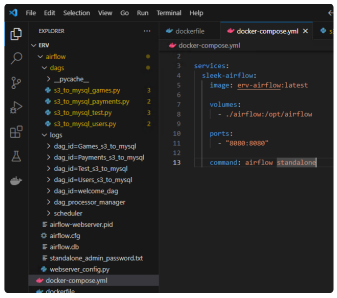
S3 connection

Time to write our S3-to-MySQL DAG code:

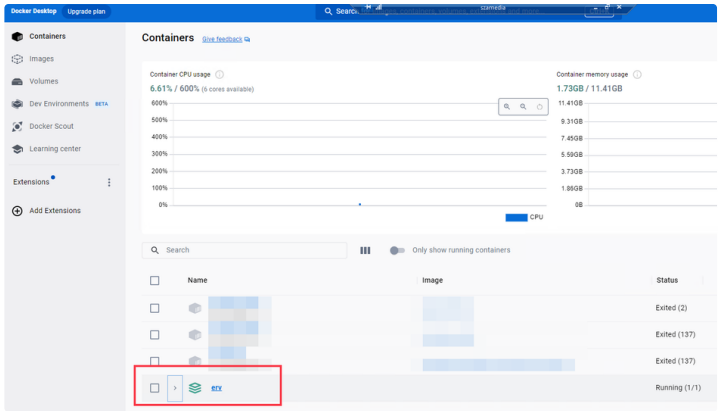
(presented in GiHub and attached).

I found I did not want to use pandas and took `S3ToSqlOperator` from Amazon provider.

Also, I found out that Amazon started billing me real money for Airflow usage online, So I decided to delete this online instance and rebuild it locally on Docker:

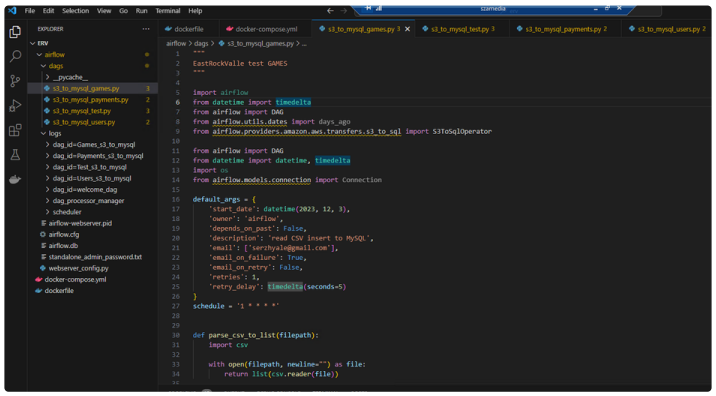


New project in VSCode with docker compose (windows)

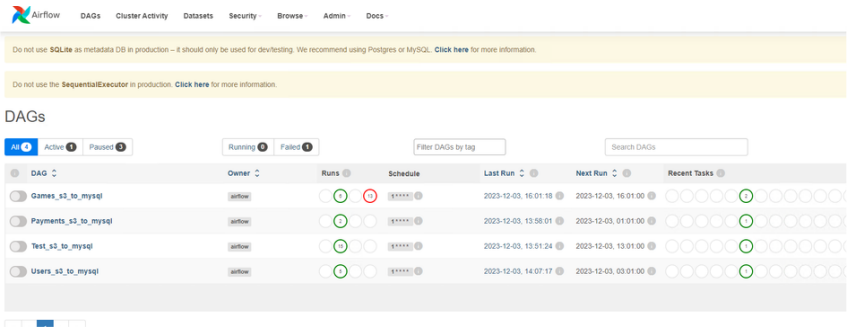


docker Airflow container

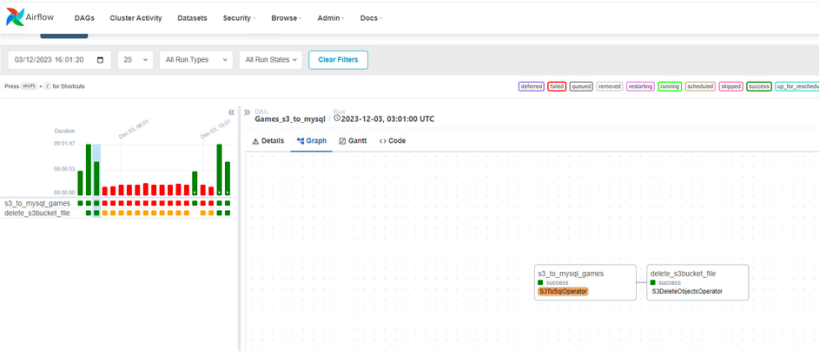
I repeat the connections in this local Airflow instances (as shown above for online)



put all dags in the Airflow docker-linked folder

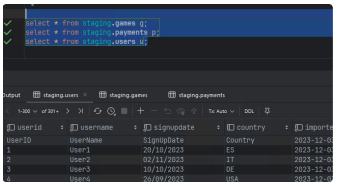


dags ready to be triggered



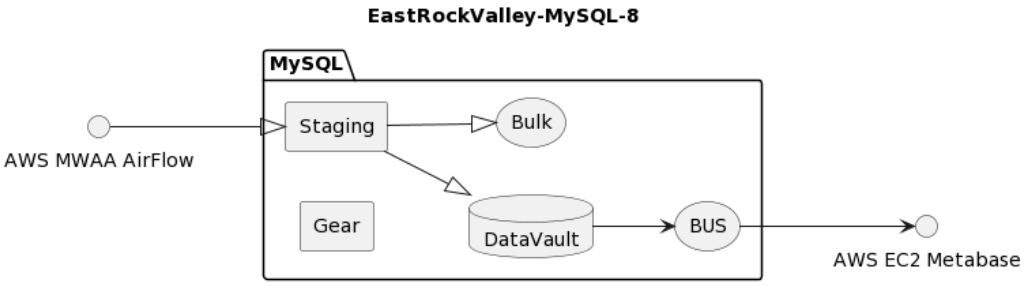
triggering DAGs to move data to MySQL database and delete the source file

In my scenario, new files will appear in the same place with the same name.



In MySQL we can see data is coming

MySQL



MySQL

MySQL code. Return to DataGrip

(Code is presented in Github or attached)

schema Staging - a place for landing all incoming data 1:1

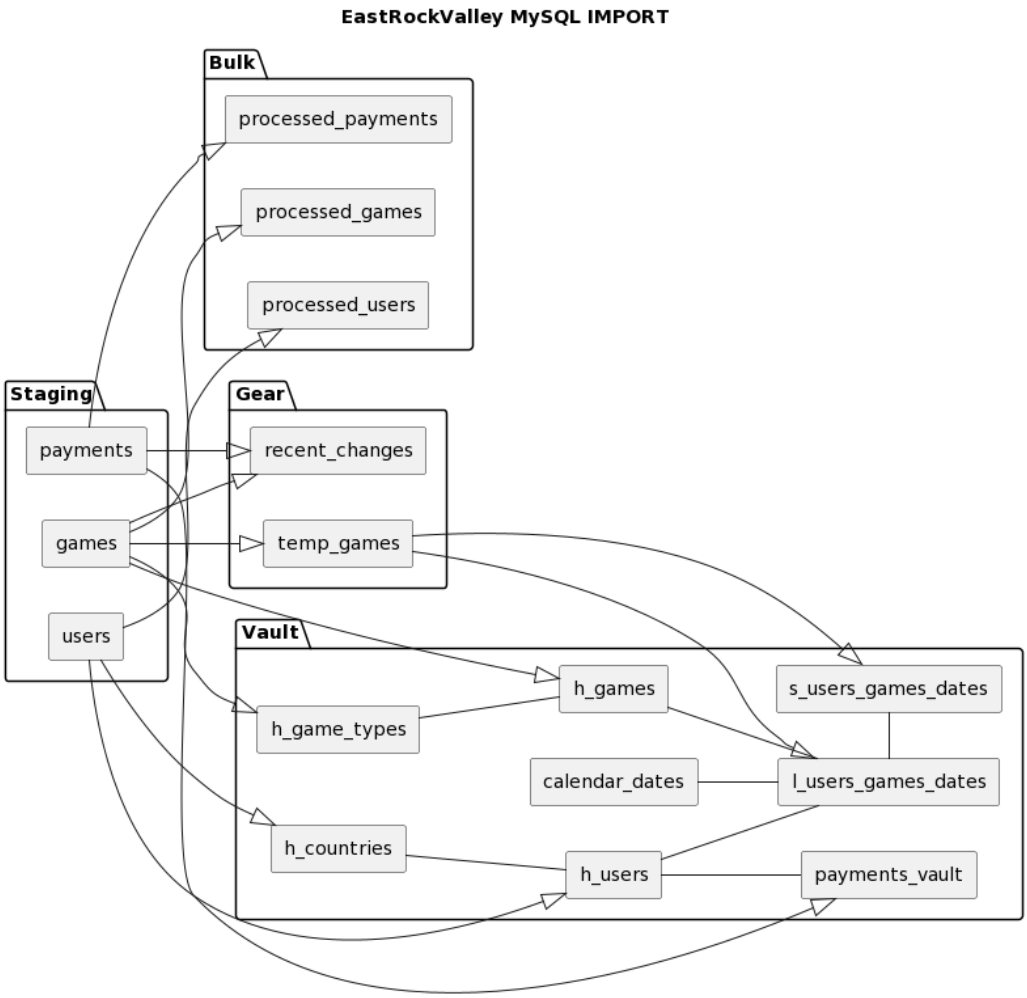
schema Bulk – to archive already processed data

schema Gear - all processing stored procedures (routines) and temporary tables

schema Vault - data storage with datavault policy

schema BUS – for report layer aggregated tables and views. Source for Reporting System

calendar_dates - a table of dates generated additionally (taken from my other project)



uml schema presented in txt file as well

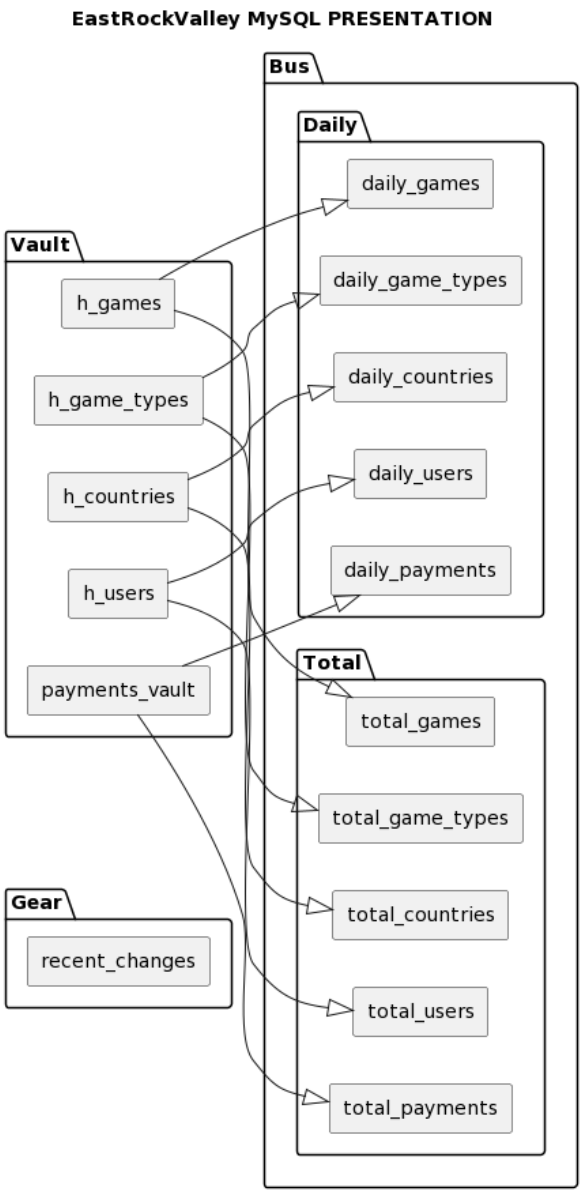
The landing is in an “every 10 minutes” event:

```
-- ------------------------------------------------------------------------ --
DROP PROCEDURE IF EXISTS gear.exec_process_staging;
CREATE PROCEDURE gear.exec_process_staging()
BEGIN
    -- import
    CALL gear.exec_process_users();
    CALL gear.exec_process_games();
    CALL gear.exec_link_users_and_games();
    CALL gear.exec_process_payments();

END;

DROP EVENT IF EXISTS gear.ten_minutes_routine;
CREATE EVENT gear.ten_minutes_routine
    ON SCHEDULE EVERY 10 MINUTE
        STARTS '2023-12-01 00:00:01' ON COMPLETION PRESERVE ENABLE DO
    CALL gear.exec_process_staging();
```

The next step is aggregation. This step is independent of the landing one.

By the code, it is waiting for the changes in table “recent_changes” and recalculates only aggregations for the updated (added) period and for updated (added) group elements. Everything is realized as a series of stored procedures.

```
DROP EVENT IF EXISTS gear.ten_minutes_routine_aggregations;
CREATE EVENT gear.ten_minutes_routine_aggregations
    ON SCHEDULE EVERY 10 MINUTE
        STARTS '2023-12-01 00:00:01' ON COMPLETION PRESERVE ENABLE DO
    CALL gear.calc_aggregations();

DROP EVENT IF EXISTS gear.midnight_routine_aggregations;
CREATE EVENT gear.midnight_routine_aggregations
    ON SCHEDULE EVERY 24 HOUR
        STARTS '2023-12-01 00:00:01' ON COMPLETION PRESERVE ENABLE DO
    CALL gear.aggregate_Excel_payments();
```

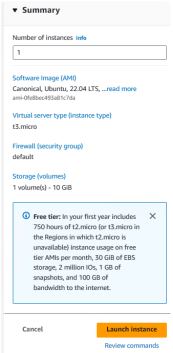


BUS layer is going to be shown in Metabase

Metabase (Reporting)

Take JAR file of the free version of Metabase from https://www.metabase.com/start/oss/jar

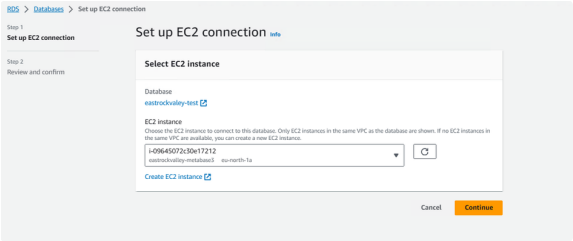
Create EC2 instance with Ubuntu in AWS



create pem, copy jar, yum install java-devel, java -jar metabase.jar

add inbound rule for port 3000 (metabase)

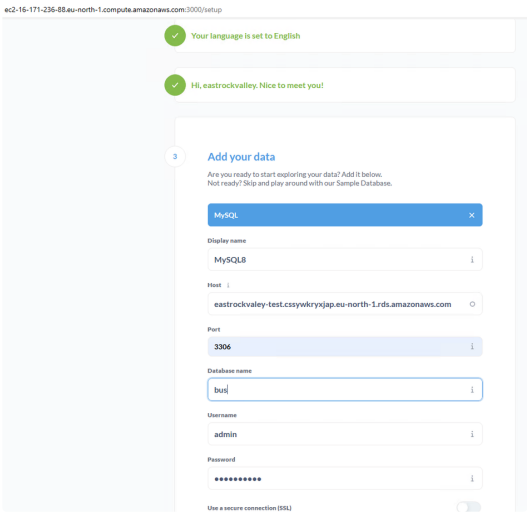
for mysql instance open connection to metabase:



Metabase (ec2-16-171-236-88.eu-north-1.compute.amazonaws.com)
Metabase

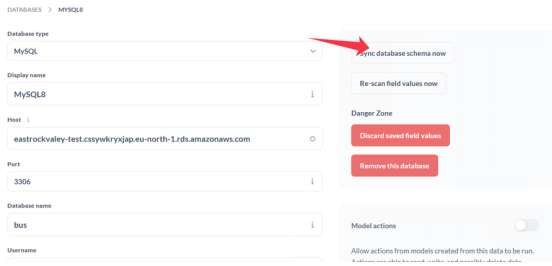
login serzhyale@gmail.com

password: eastrockvalley1

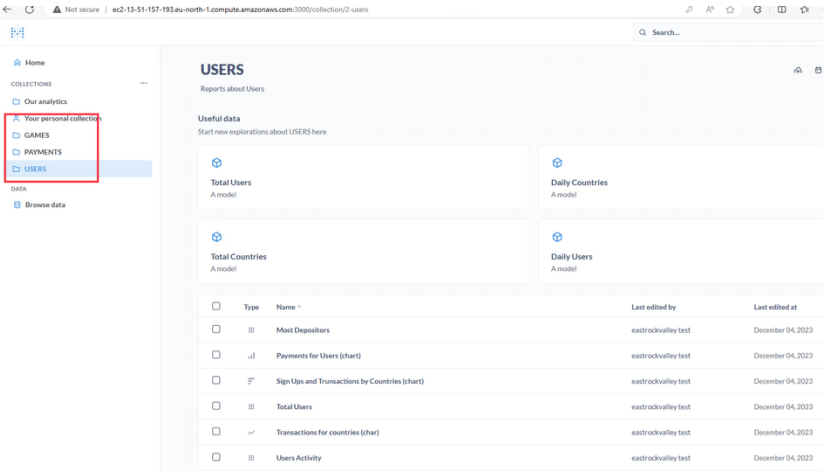


connect metabase on bus schema

Metabase (ec2-16-171-236-88.eu-north-1.compute.amazonaws.com)



refresh schema



In Metabase all Reports placed in blocks