IOT-BASED PLATFORM
AIR QUALITY (PM2.5) MONITORING SYSTEM

Big Data Tool with Apache Kylin

Project Members:

Younten Tshering

Suyogya Ratna Tamrakar

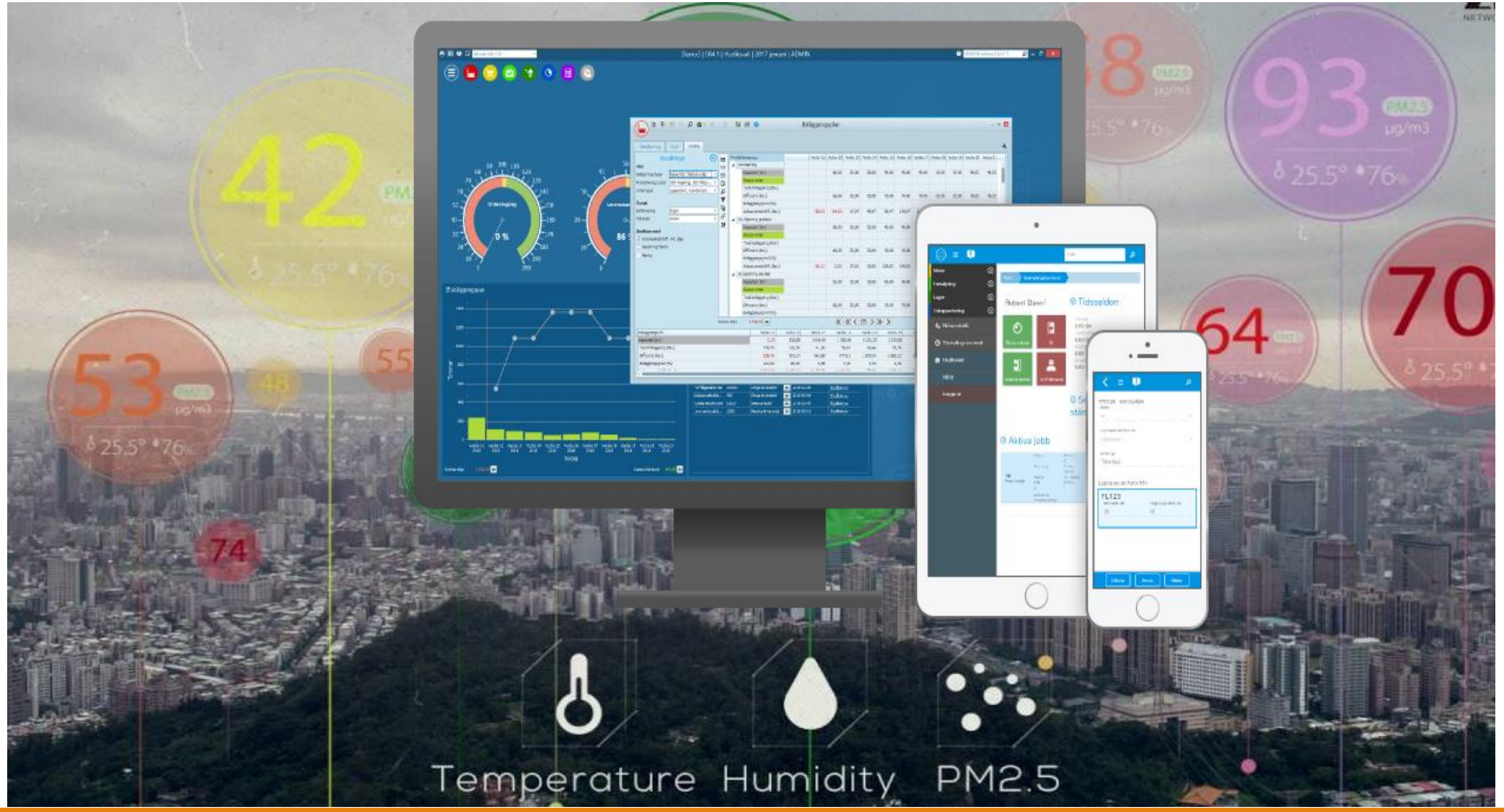
Smrity Baral

Shubhangini Gontia

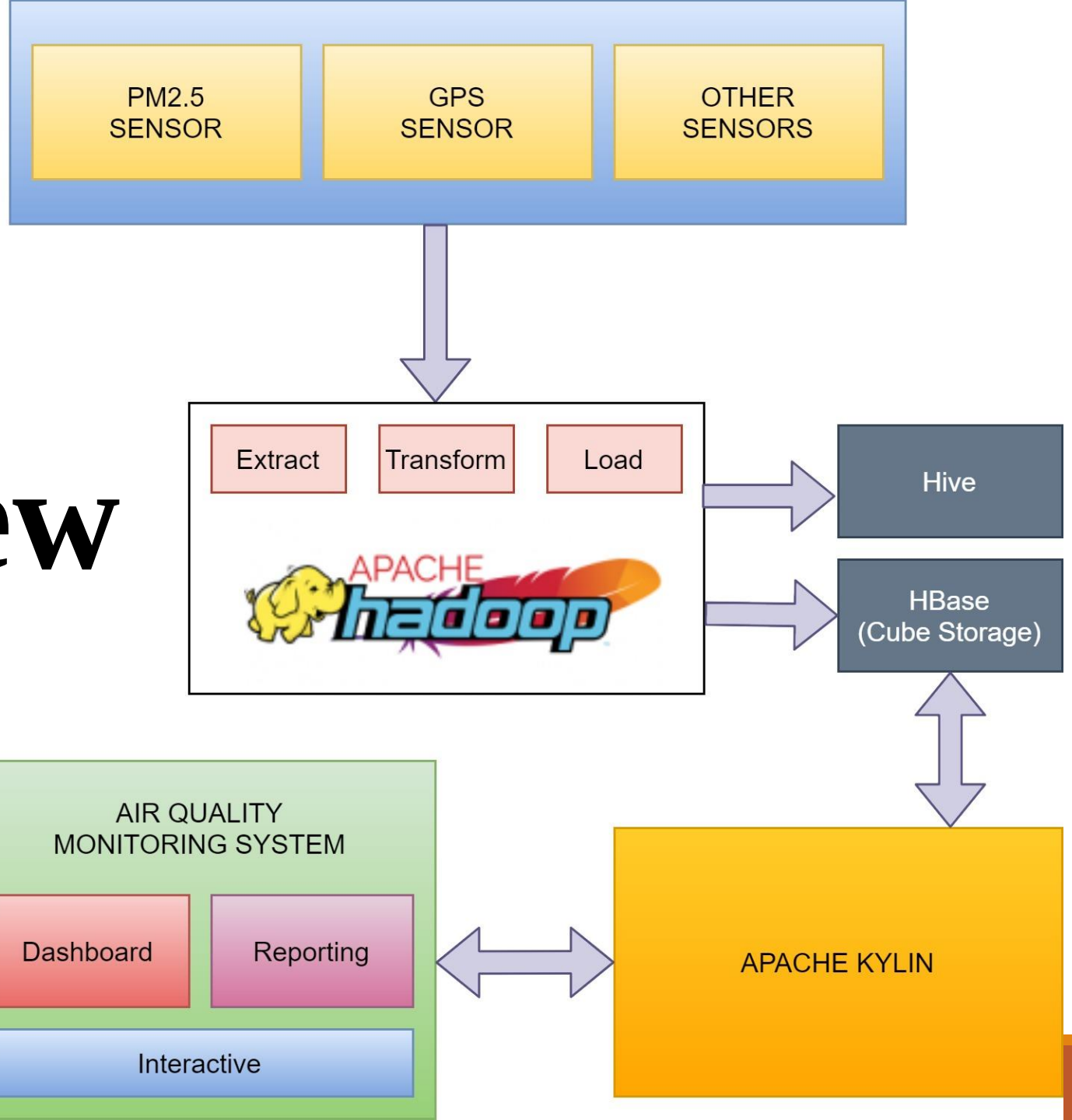
Outlines

1. Introduction
2. Software architecture
3. Design goals and trade-offs
4. Access Matrix
5. Framework/Library
6. User Acceptance Test Report
7. The status of the implementation
8. Real Demo

Introduction



Overview


PM2.5 SENSOR
GPS SENSOR
OTHER SENSORS
Extract
Transform
Load
APACHE hadoop
Hive
HBase (Cube Storage)
AIR QUALITY MONITORING SYSTEM
Dashboard
Reporting
Interactive
APACHE KYLIN

Functional requirements

Visualization Module

- The end users should be able to **view** an interactive dashboard of air quality monitoring with different forecasts and insights.
- The system should be able to **stream** real-time data from different nodes and stations.

System Admin Module

- The admin should be able to **login, logout** to the system and **modify** the system parameters and toggle dashboard controls.
- The admin should be able to register new sensors and **manage** the sensors in system.
- The admin should be able to **generate** reports of specific time periods and export those in varies formats.

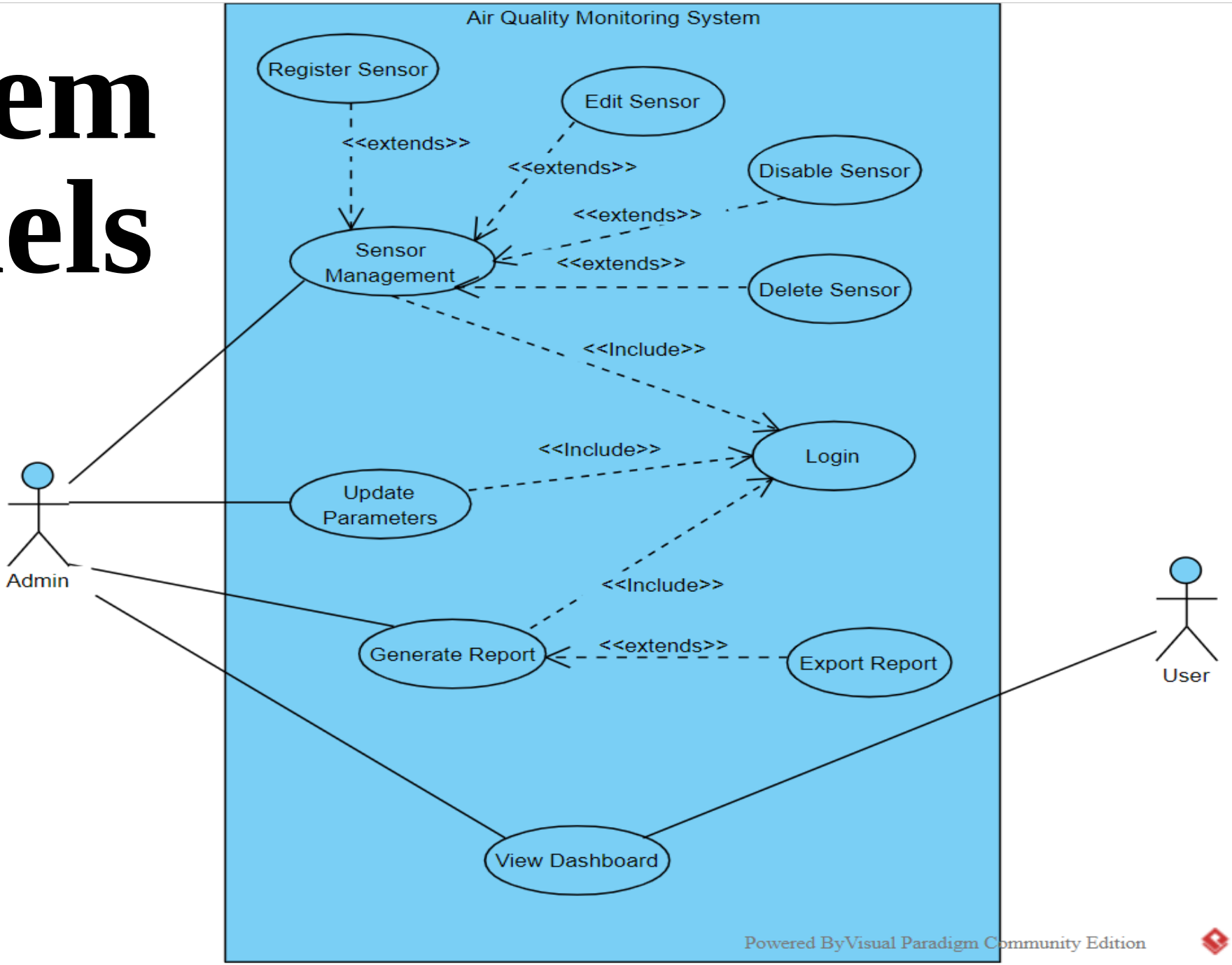
Data Collection Module

- Data will be extracted from sensors and stored in **Hadoop** which is acting as data warehouse using Hive.
- **Kylin** does aggregation functions on cube (HBase) and provide the required parameters to the system.

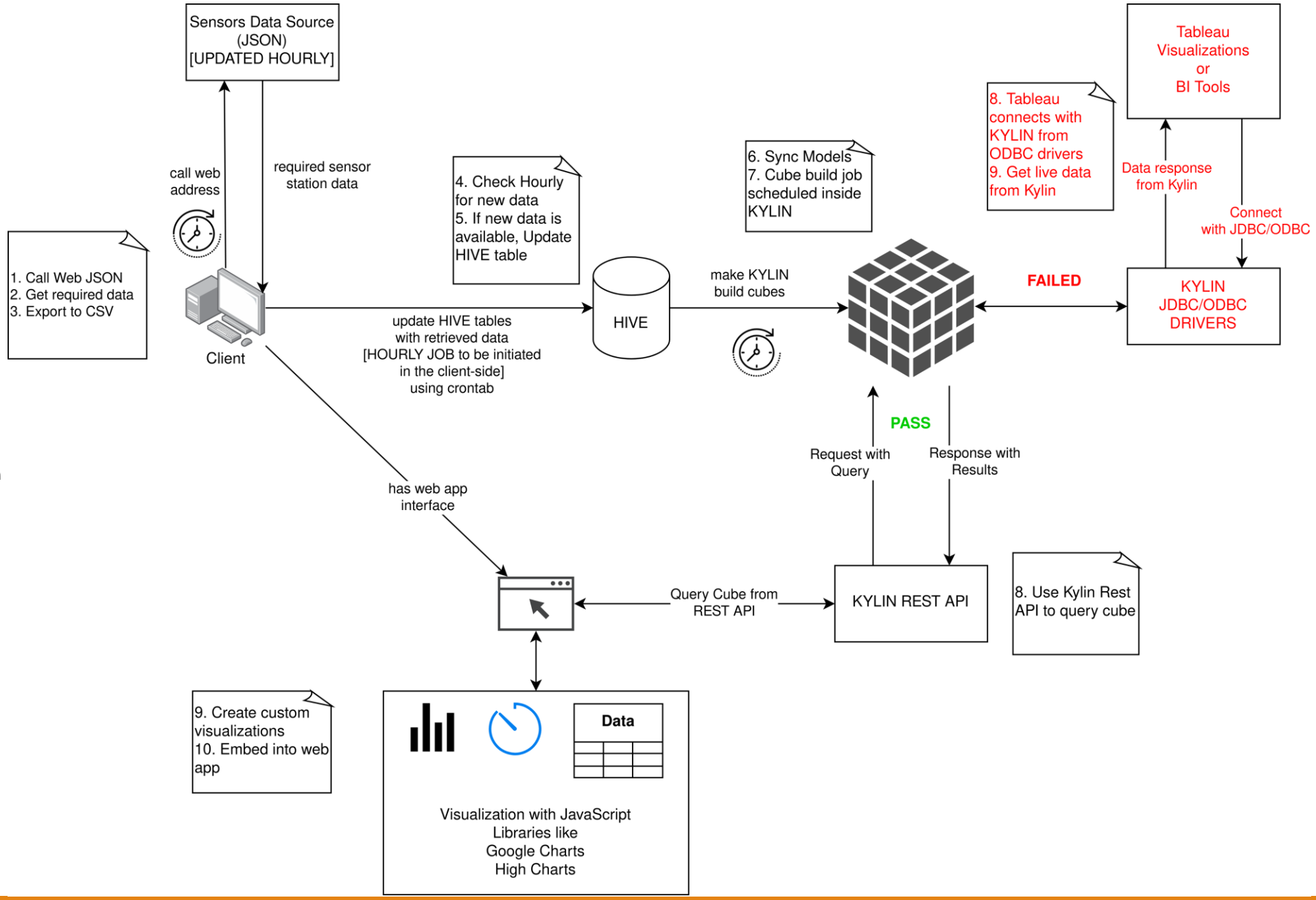
Nonfunctional requirements

- Performance
- Reliability
- Interoperability
- Portability
- Scalability
- Reusability

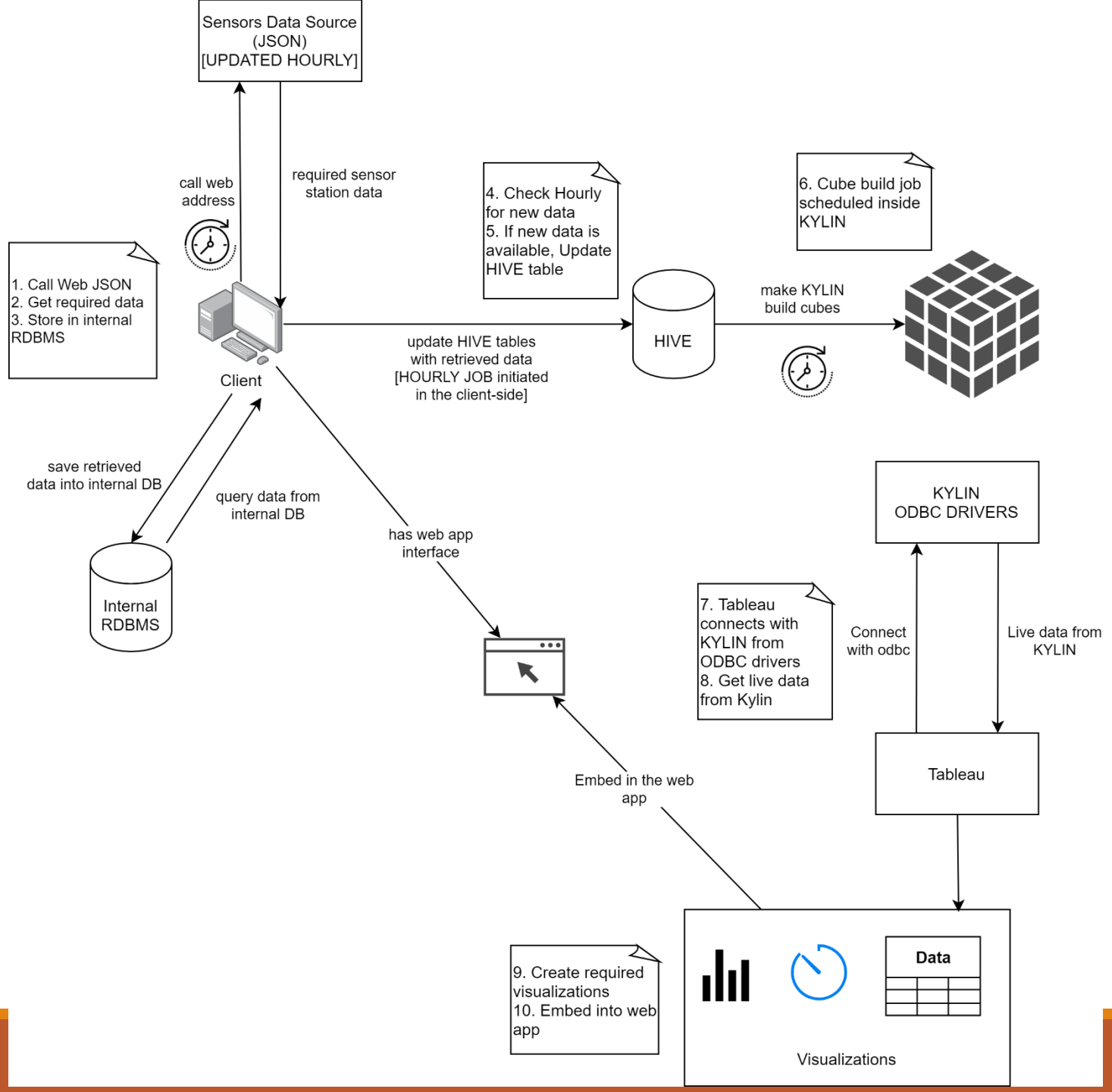
System models



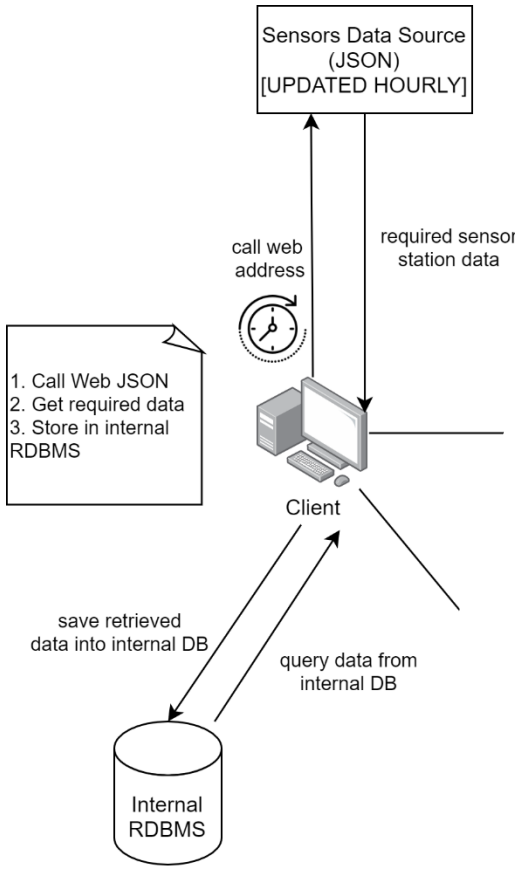
New Software Architecture



Software Architecture



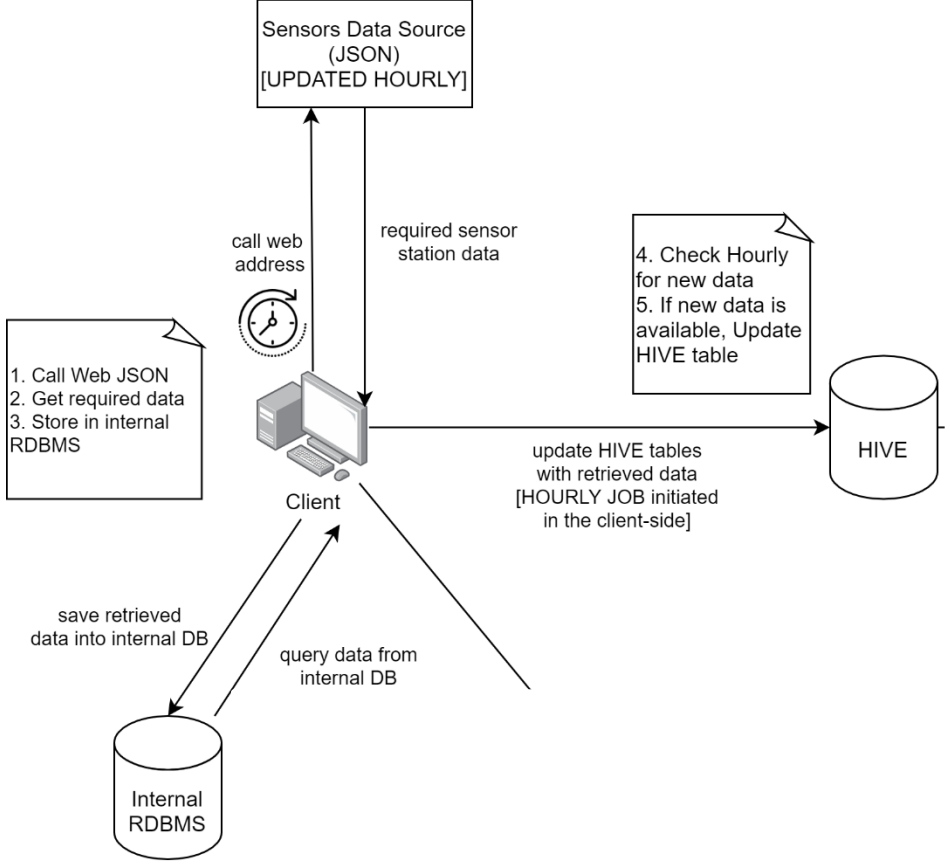
Data retrieval




```
home > srt > Documents > PythonTests > import.py > ...
 1  import csv, json, urllib, sys
 2  from datetime import datetime
 3  import os
 4
 5  # Fetch JSON from URL
 6  url = "http://www.air4thai.com/services/getNewAQI_JSON.php"
 7  response = urllib.urlopen(url)
 8  web_data = json.loads(response.read())
 9  web_data = web_data["stations"]
10
11
12  # Create our custom dictionary
13  data = []
14  for wd in web_data:
```

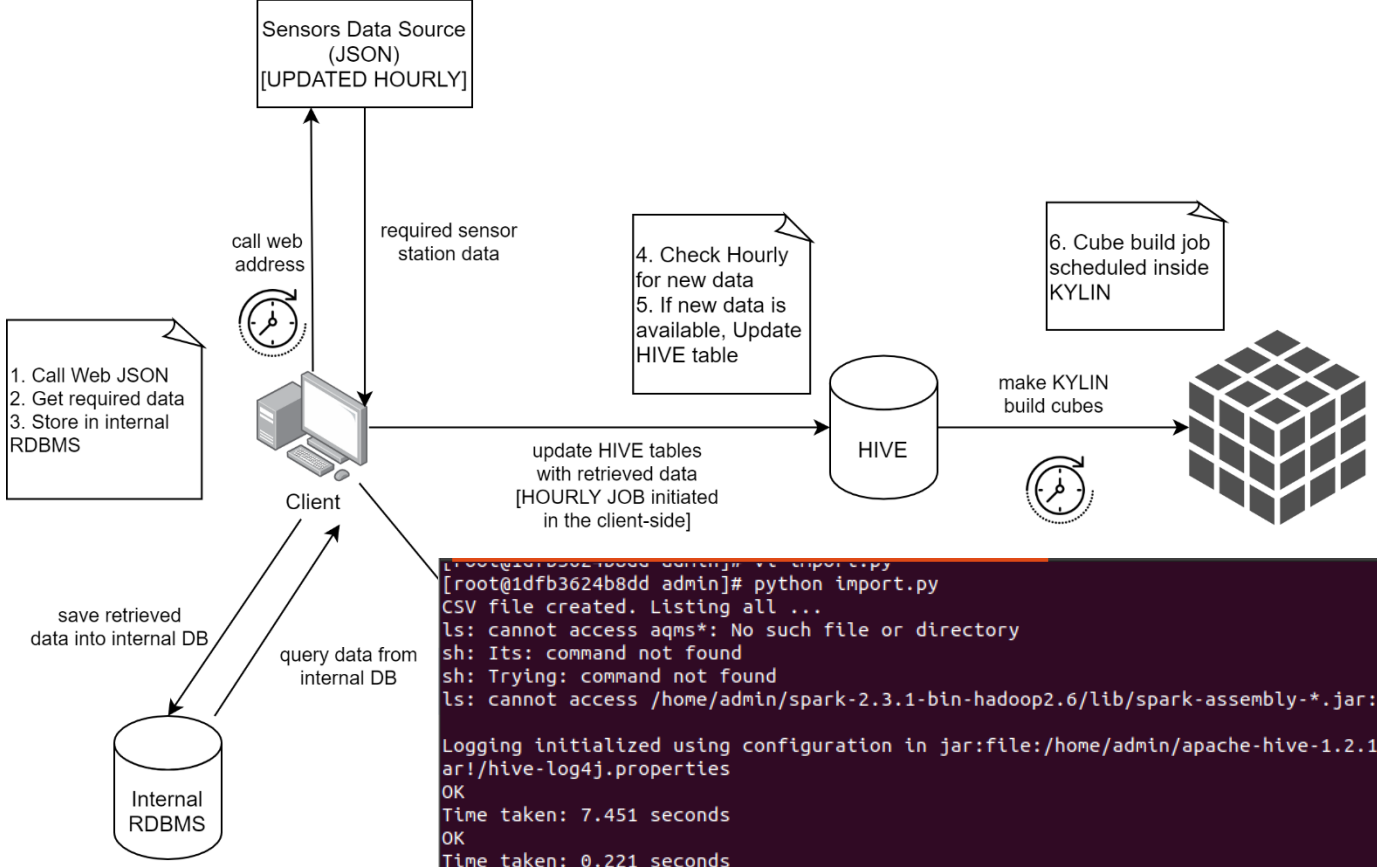
```
41
42  # output.writerow(data[0].keys()) # Header row
43
44  for row in data:
45      output.writerow(row.values()) # Values row
46
47
48  # Run shell script to run this file
49  os.system("echo CSV file created. Listing all ...")
50  os.system("ls -l aqms*")
51  os.system("Its time to connect to hive and copy this file into it")
52
53  # Load data into HIVE
54  os.system("Trying to load this generated file into HIVE TABLE")
55  os.system("hive -f load_aqms.sql")
56  os.system("hive")
```

Data Loading and Transformation



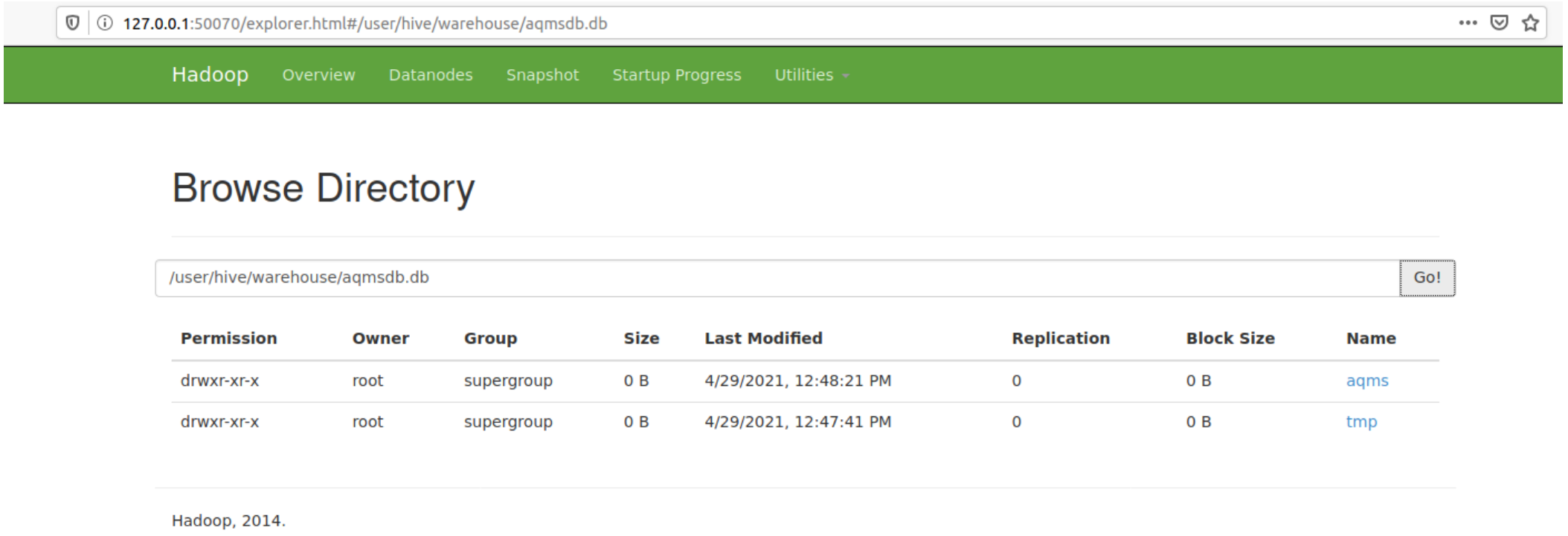
```
CREATE TABLE IF NOT EXISTS aqmsdb.tmp (
    last_updated STRING,
    station_type STRING,
    longitude FLOAT,
    station_ID STRING,
    latitude FLOAT,
    PM25_value INT,
    AQI INT,
    AQI_level INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;

describe aqmsdb.tmp;

load data local inpath '/home/admin/test_data.csv' overwrite into table aqmsdb.tmp;

CREATE TABLE IF NOT EXISTS aqmsdb.aqms (
    last_updated TIMESTAMP,
    station_type STRING,
    longitude FLOAT,
    station_ID STRING,
    latitude FLOAT,
    PM25_value INT,
    AQI INT,
    AQI_level INT
);

describe aqmsdb.aqms;

INSERT INTO TABLE aqmsdb.aqms
SELECT from_unixtime(unix_timestamp(last_updated, 'yyyy-MM-dd HH:mm')), station_type, longitude, s

select * from aqmsdb.aqms LIMIT 5;
```

Building Cube



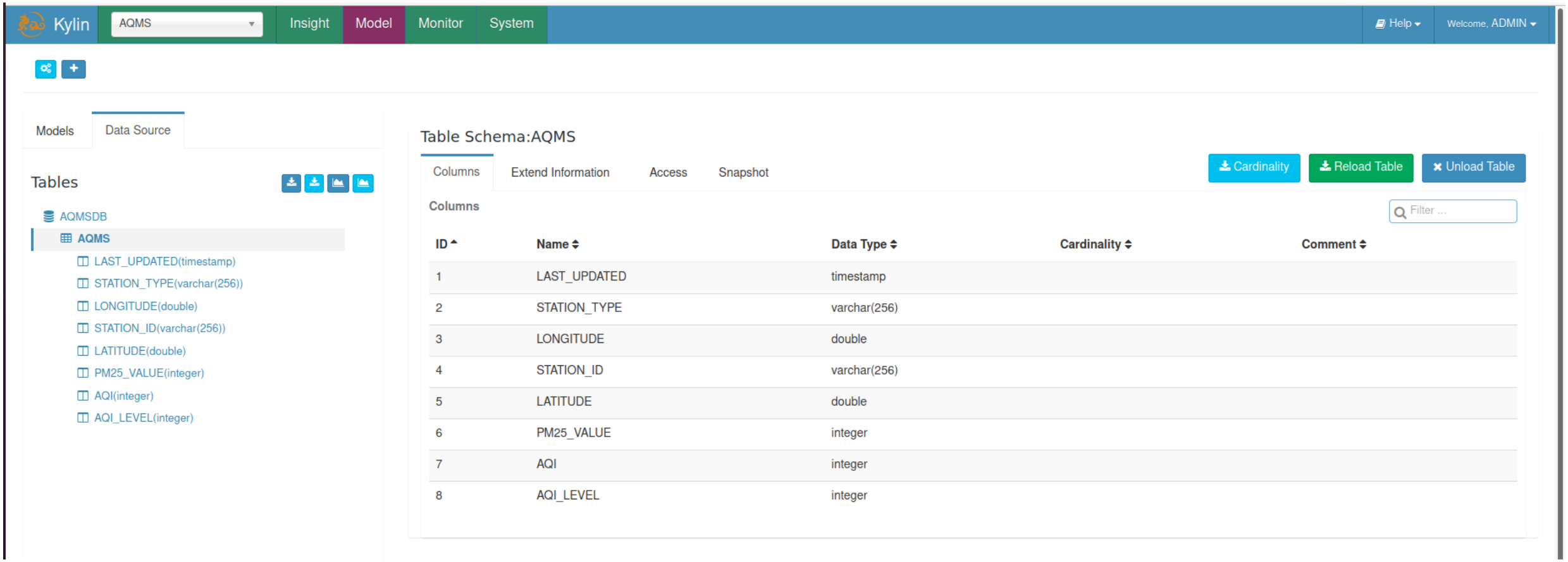

```
[root@1dfb3624b8dd admin]# python import.py
CSV file created. Listing all ...
ls: cannot access aqms*: No such file or directory
sh: Its: command not found
sh: Trying: command not found
ls: cannot access /home/admin/spark-2.3.1-bin-hadoop2.6/lib/spark-assembly-*.jar: No such file or directory

Logging initialized using configuration in jar:file:/home/admin/apache-hive-1.2.1-bin/lib/hive-common-1.2.1.j
ar!/hive-log4j.properties
OK
Time taken: 7.451 seconds
OK
Time taken: 0.221 seconds
OK
Time taken: 0.576 seconds
OK
Time taken: 0.519 seconds
OK
Time taken: 0.042 seconds
OK
Time taken: 0.7 seconds
OK
last_updated            string
station_type            string
longitude               float
station_id              string
latitude                float
pm25_value              int
aqi                     int
aqi_level               int
Time taken: 0.274 seconds, Fetched: 8 row(s)
Loading data to table aqmsdb.tmp
Table aqmsdb.tmp stats: [numFiles=1, numRows=0, totalSize=4096, rawDataSize=0]
OK
Time taken: 0.967 seconds
```

Search hive tables and view permissions

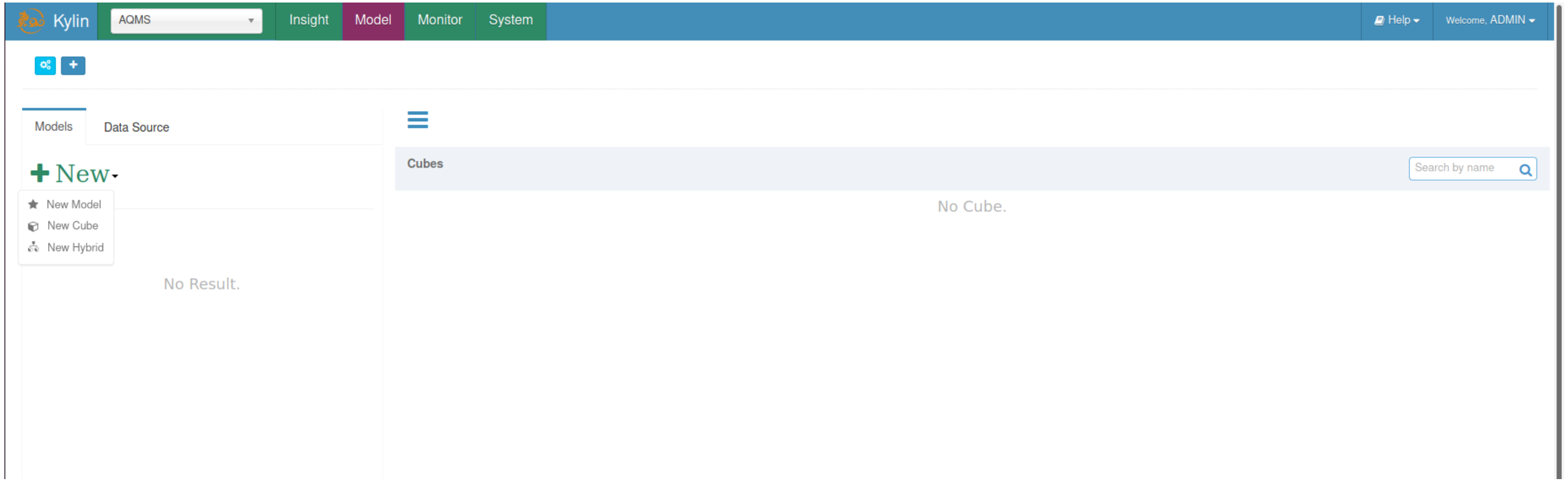


Load the table in Kylin from HIVE by navigating to data source

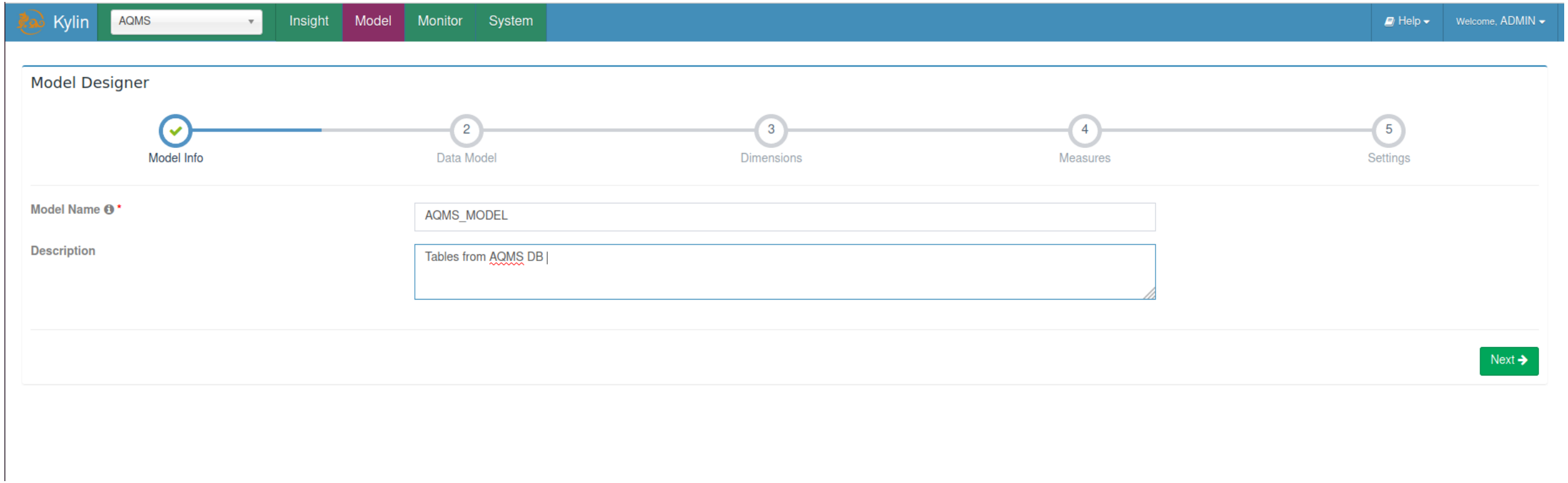
Table Schema displayed after the table is synced from HIVE



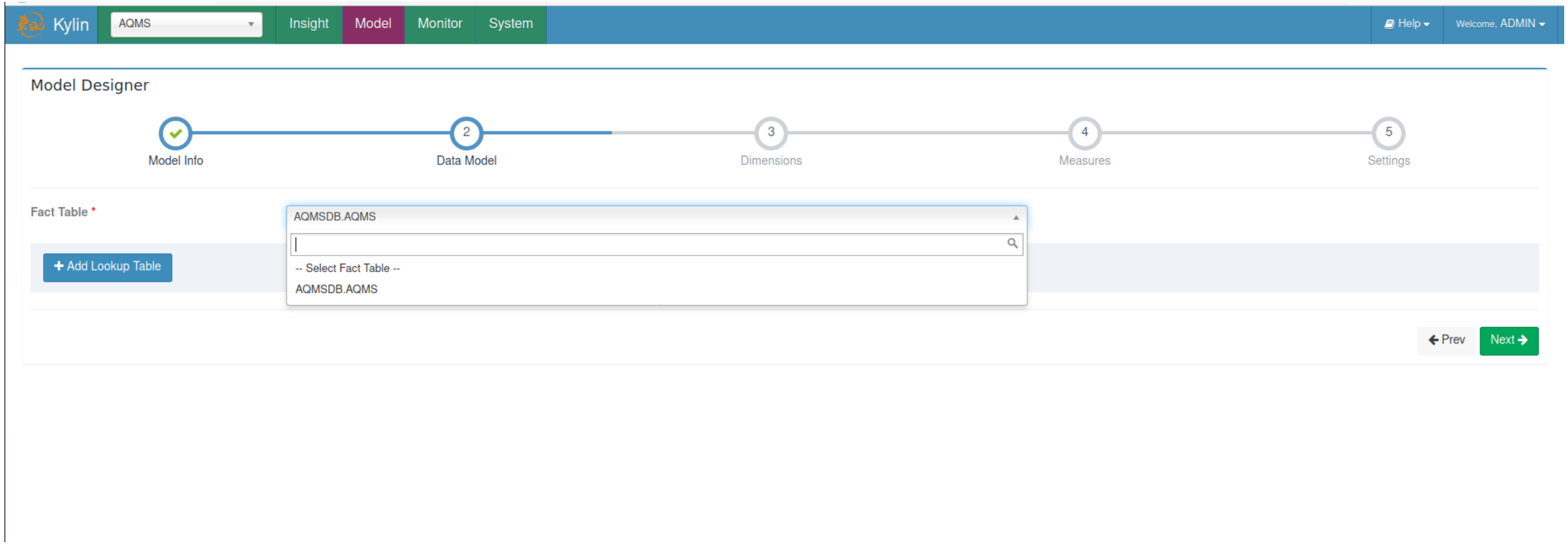
Create new Model



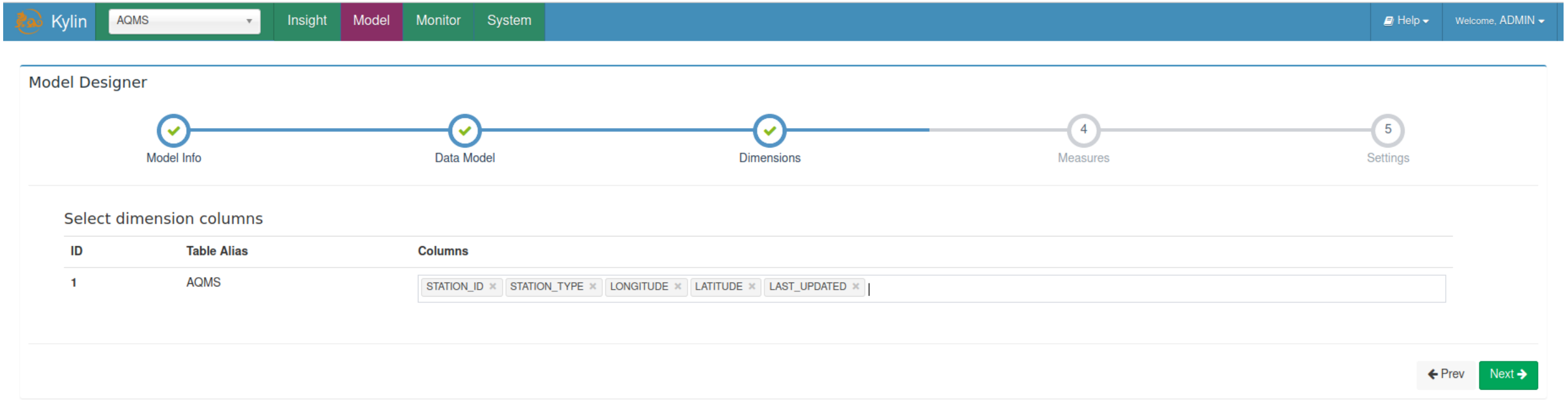
Enter model info



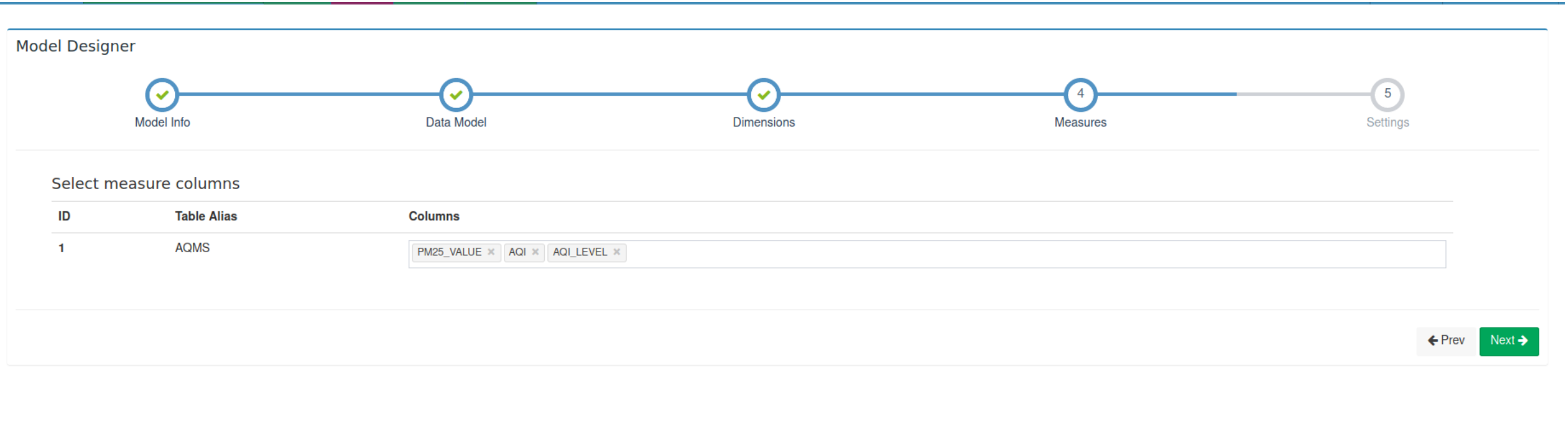
Add the fact table or base table from where all the dimension and measures are set



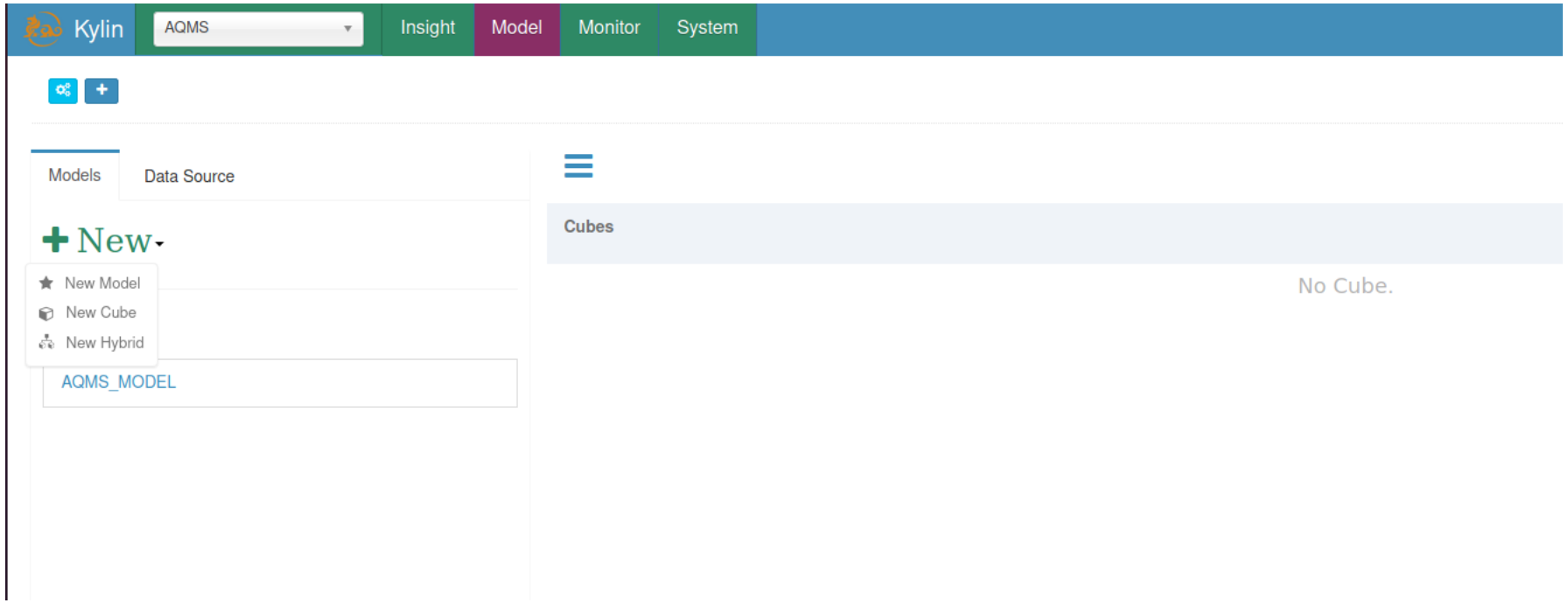
Set dimensions for the model



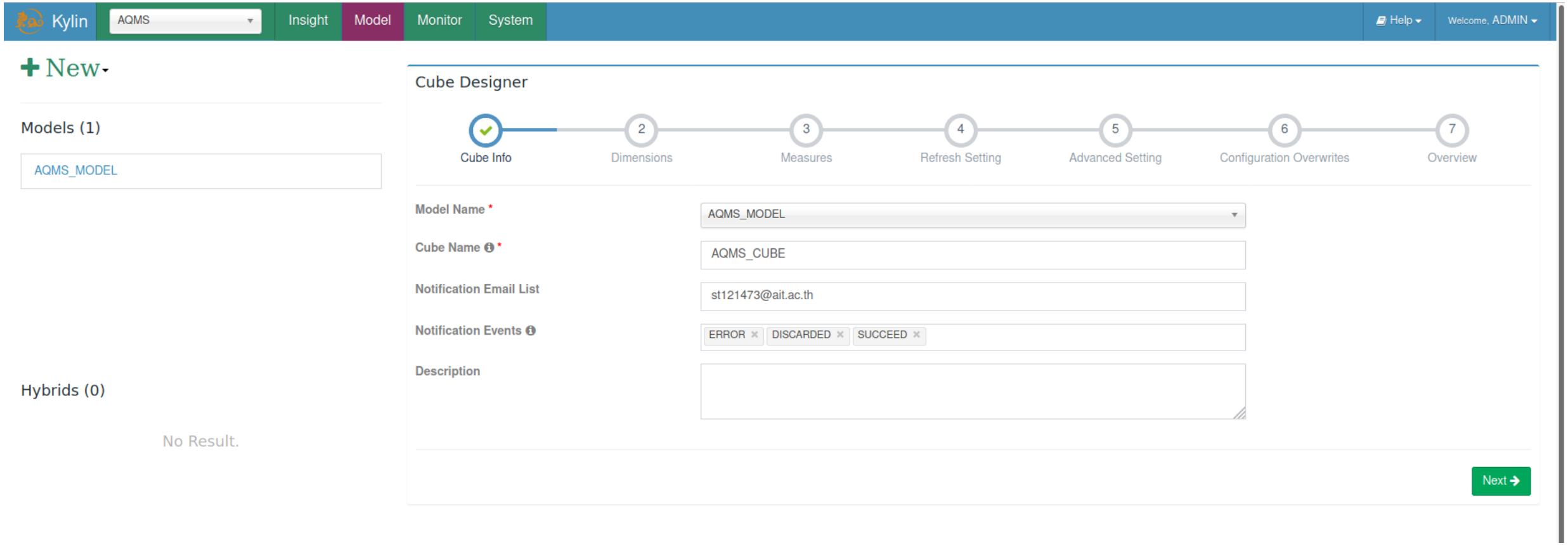
Set measure columns for the model



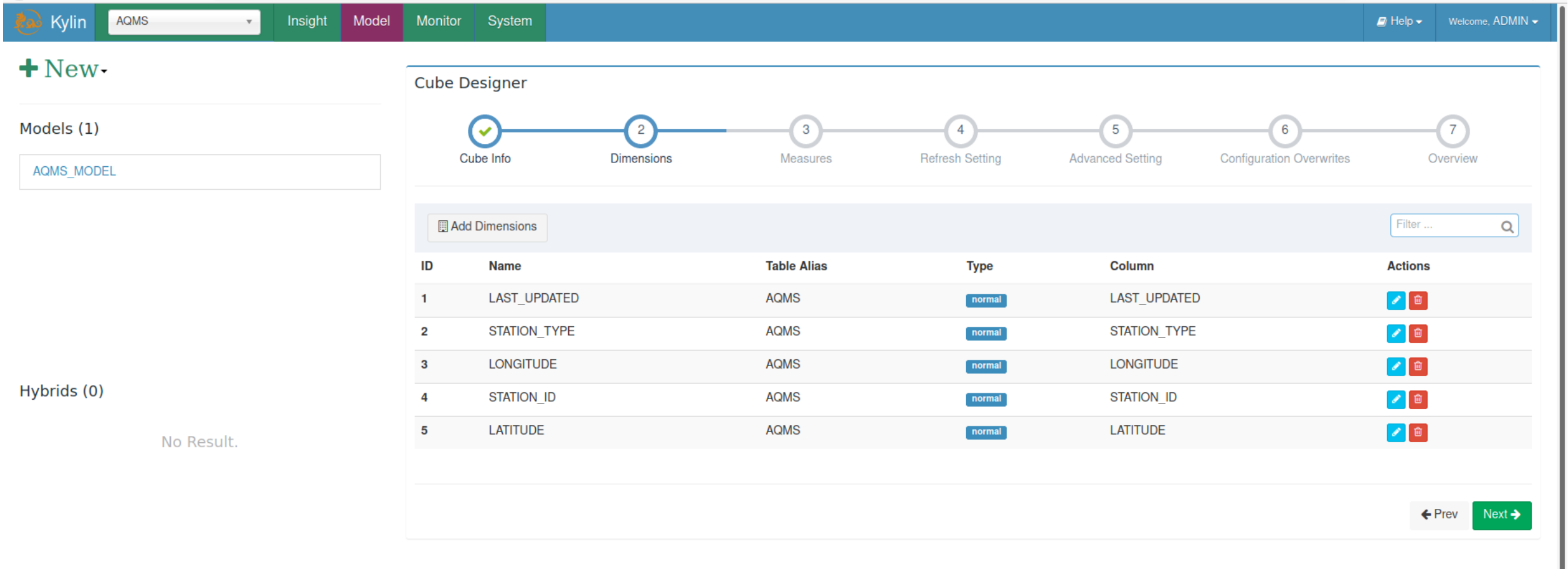
Create a new cube once the model is created



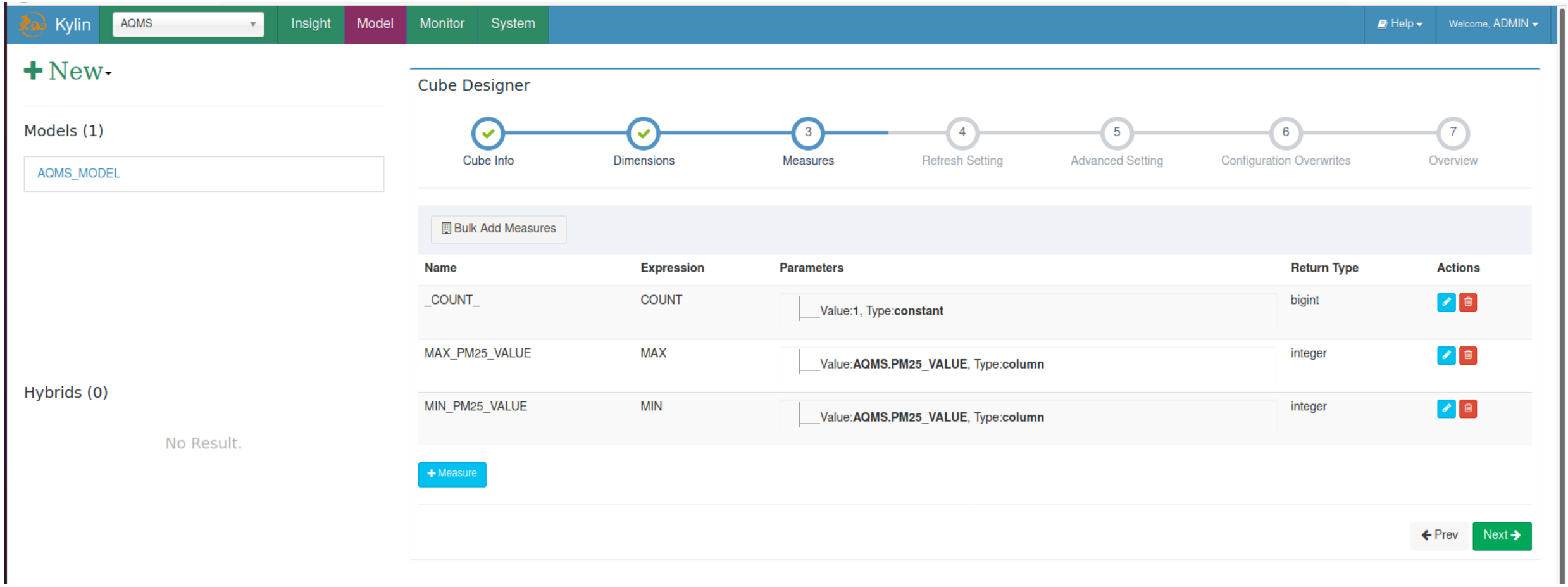
Set cube info



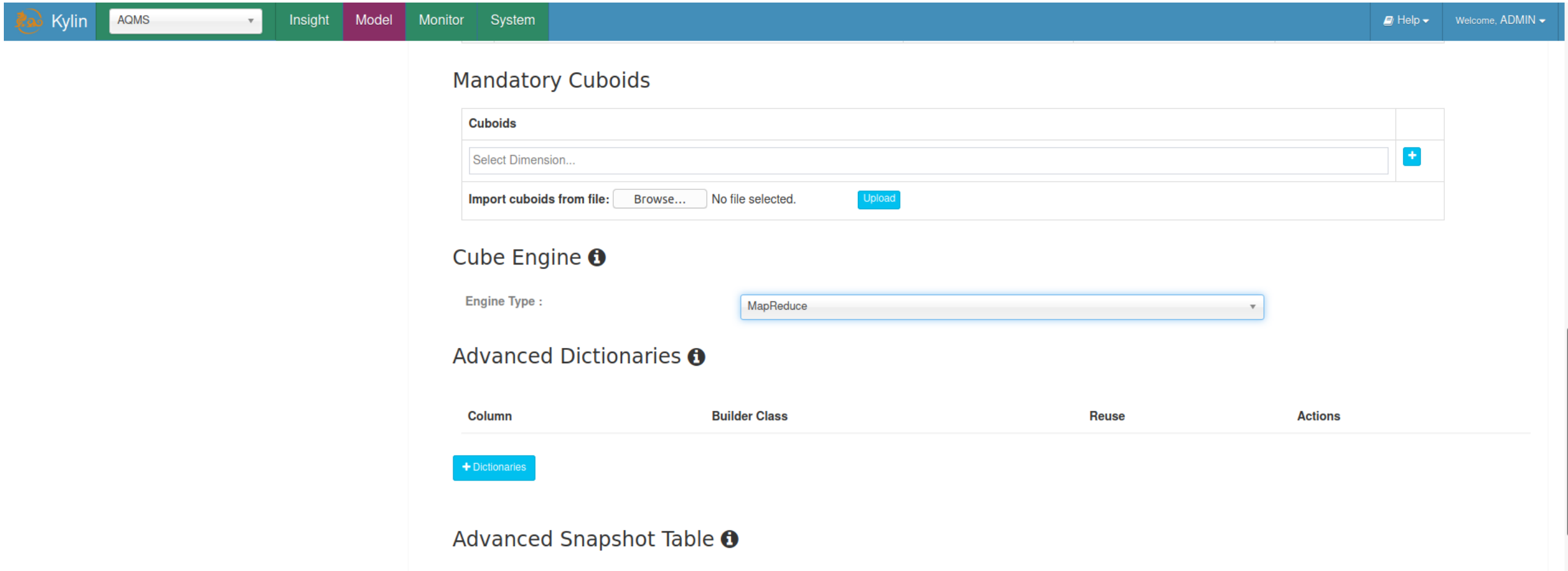
Select the dimension from the set dimension of the model



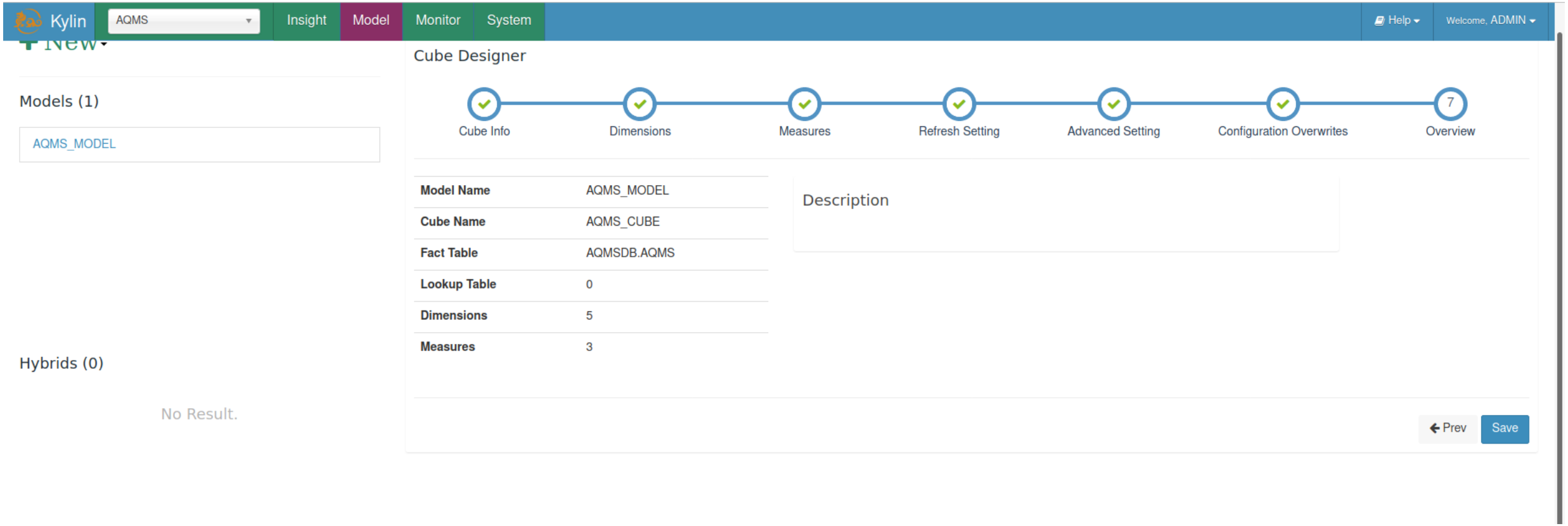
Select bulk measure for the measures



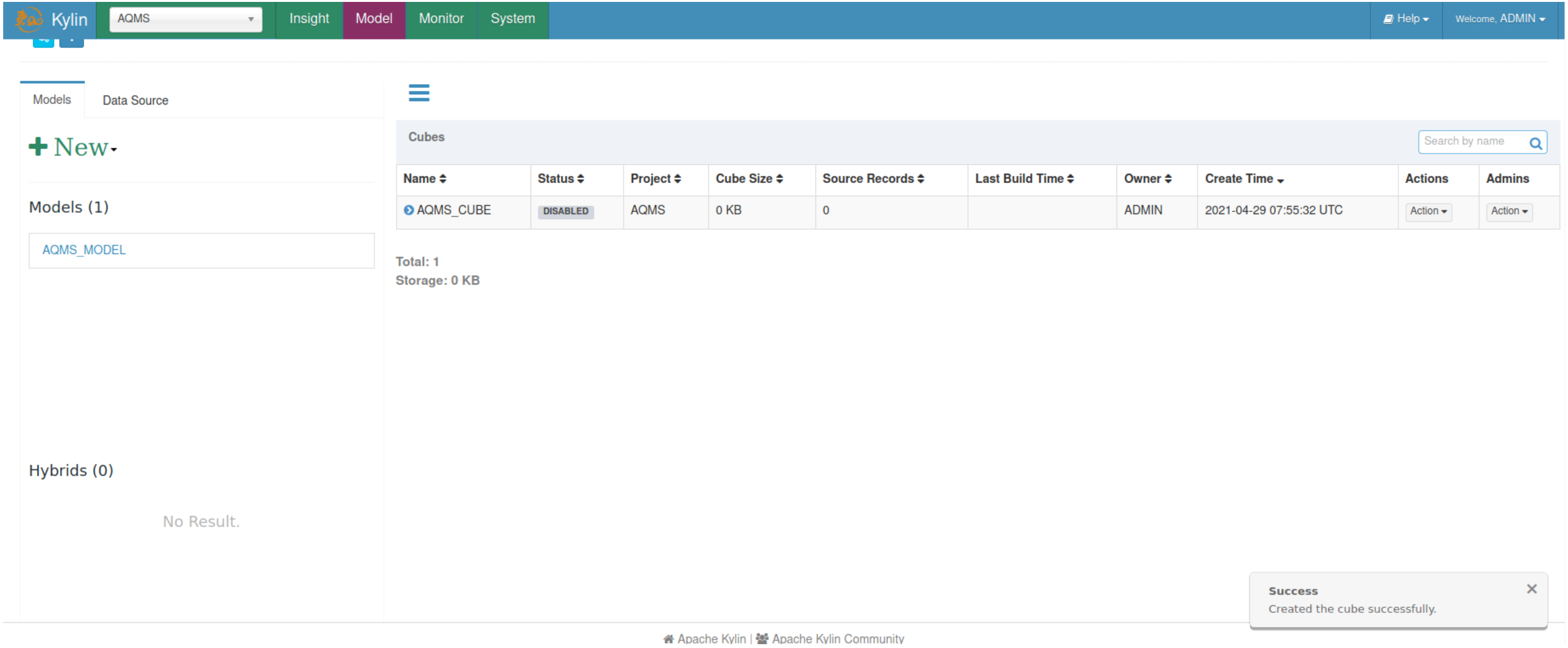
Select cube engine for the cube data



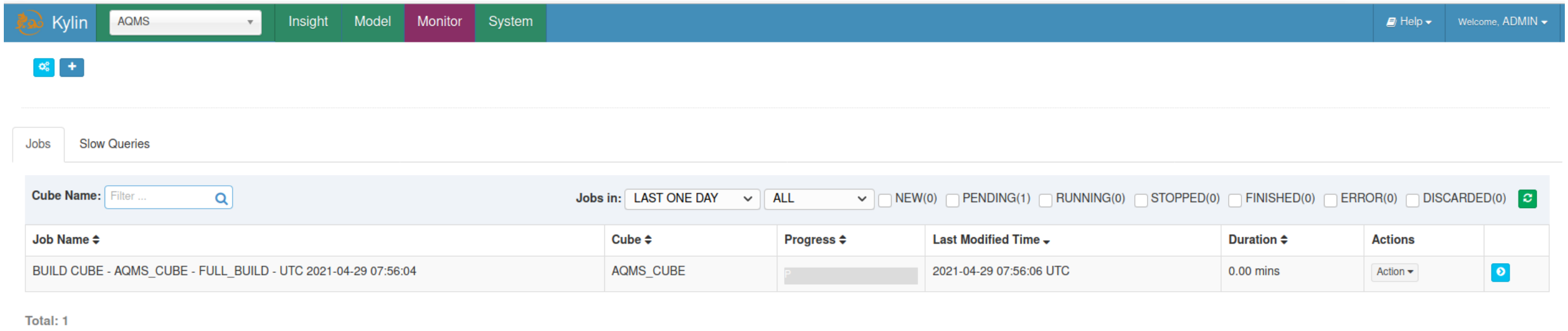
Cube info preview



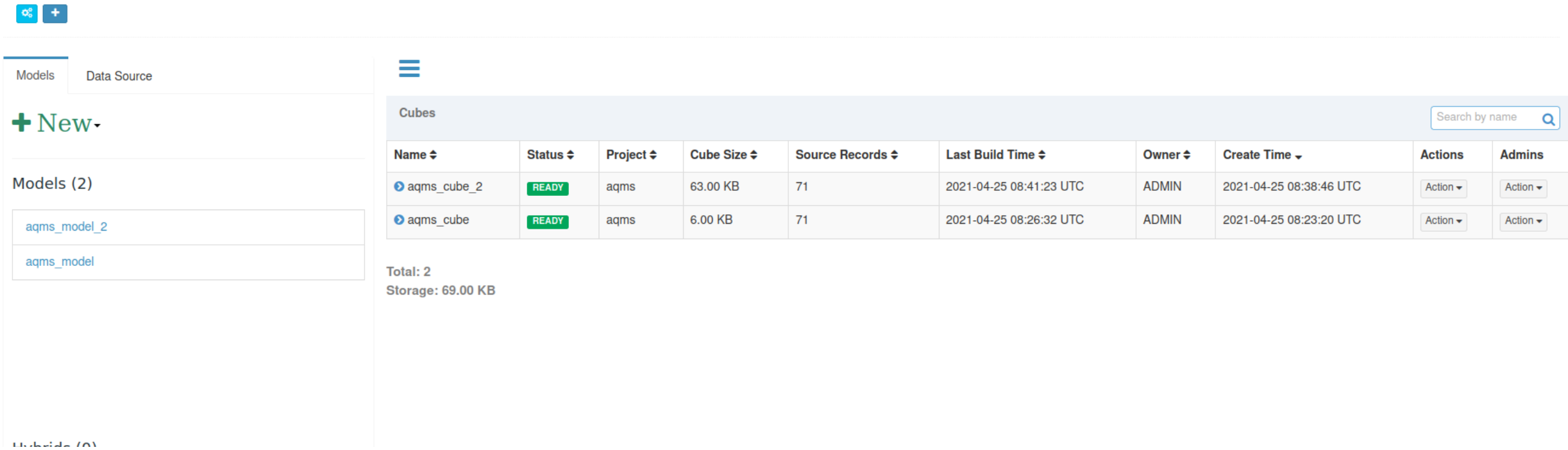
Cube is created and is in disabled status



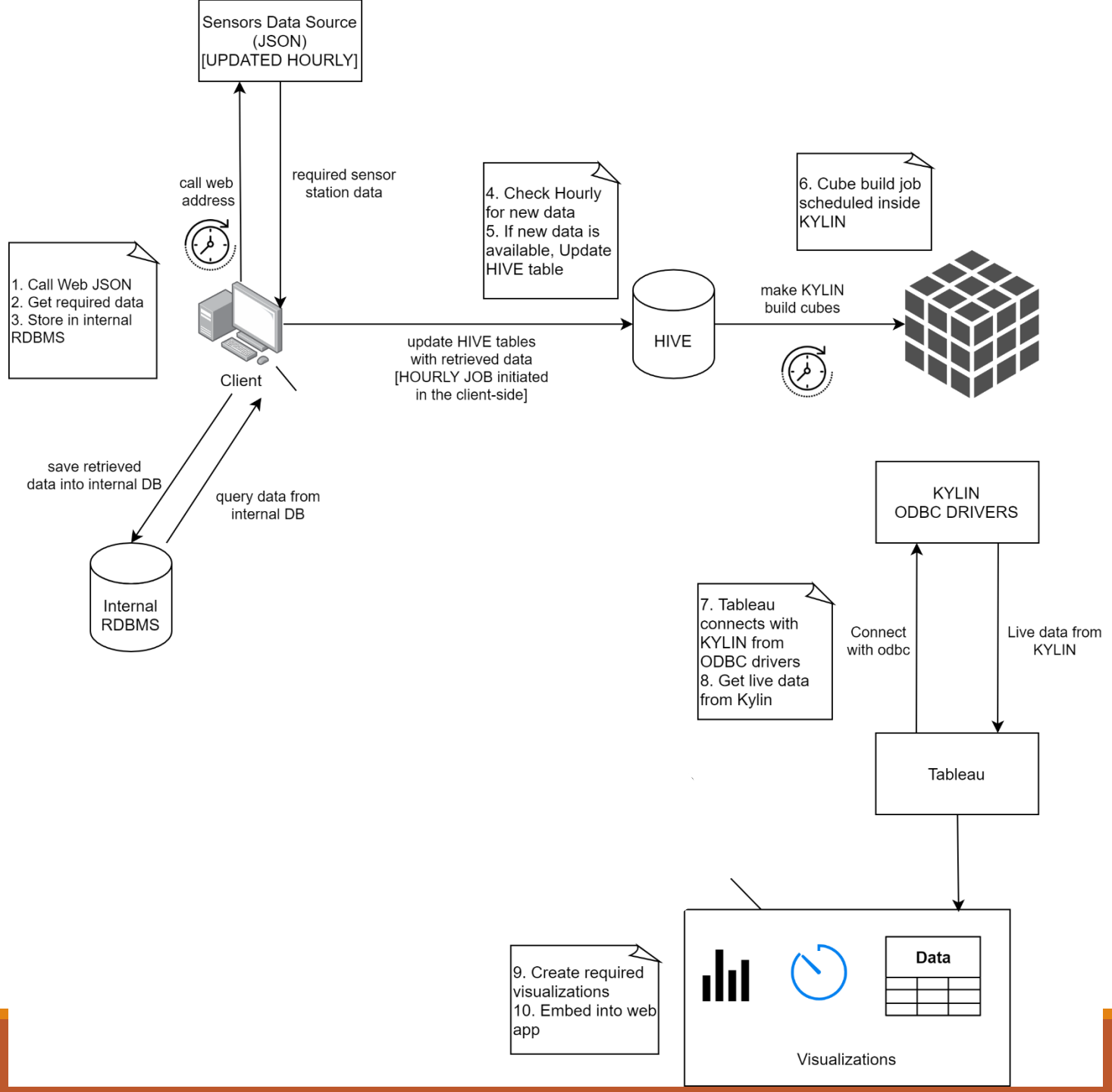
Start build process for cube and track in monitor tab

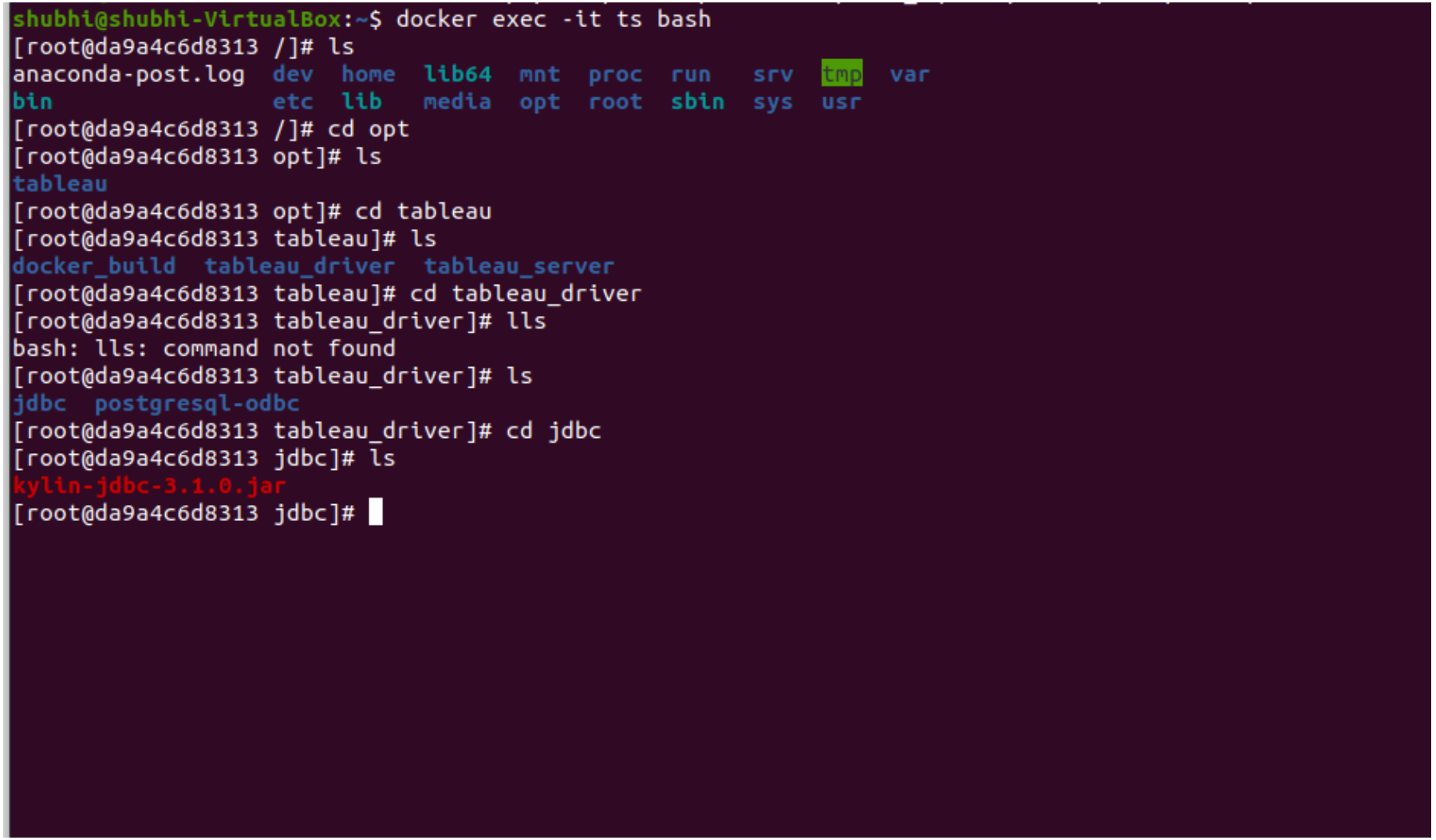


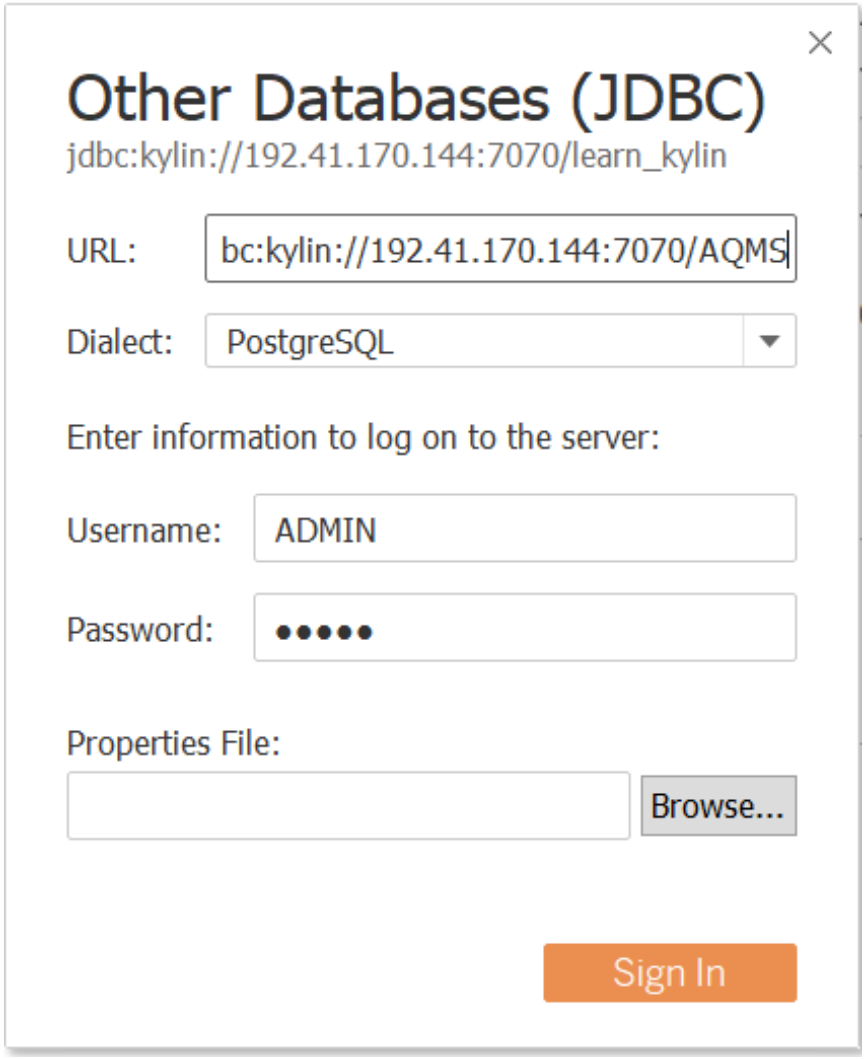
Once the cube is building is completed it’s in ready status



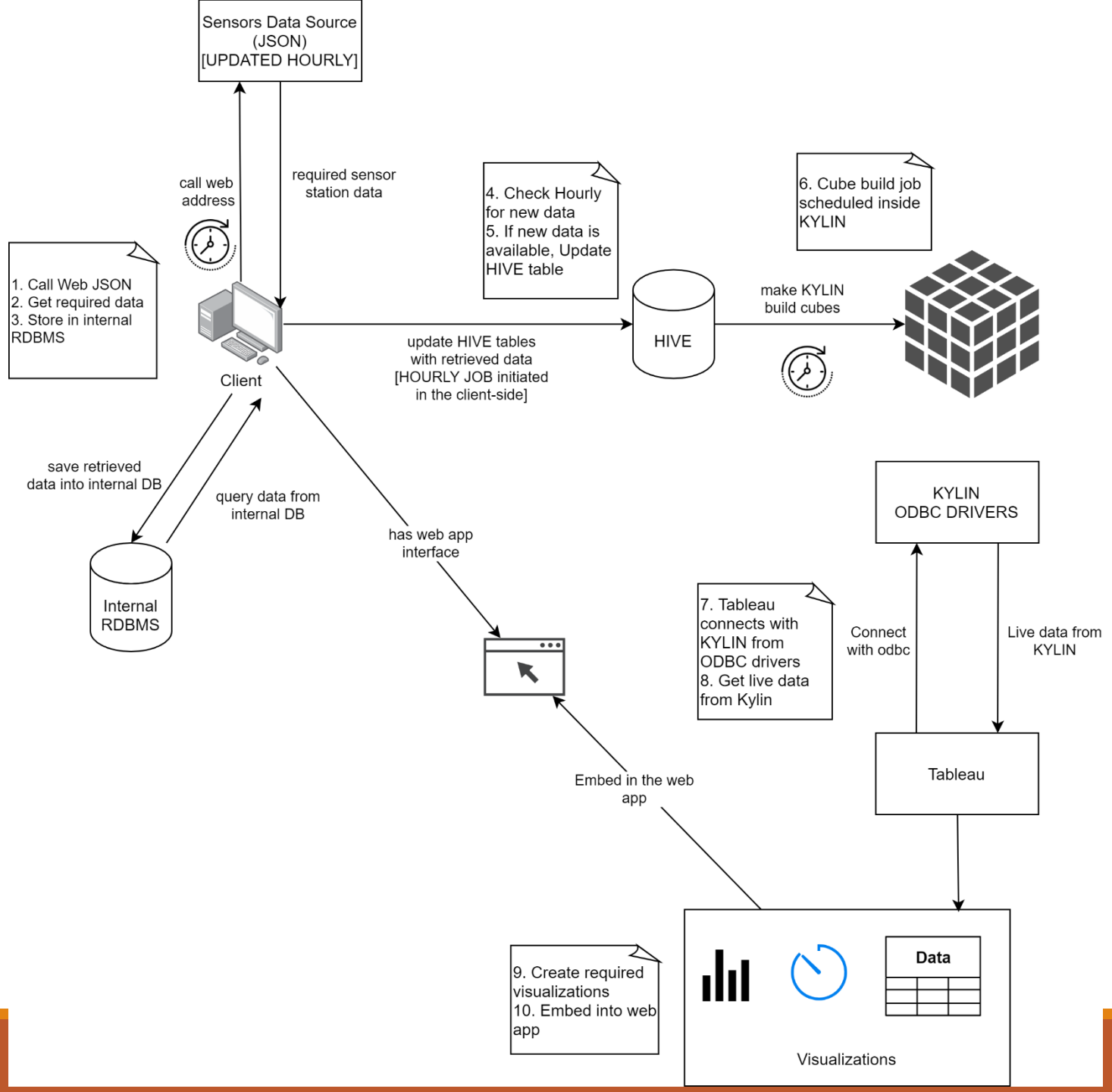
Connecting Kylin and Tableau




shubhi@shubhi-VirtualBox:~$ docker exec -it ts bash
[root@da9a4c6d8313 /]# ls
anaconda-post.log dev home lib64 mnt proc run   srv tmp var
bin   etc lib   media opt root sbin sys usr
[root@da9a4c6d8313 /]# cd opt
[root@da9a4c6d8313 opt]# ls
tableau
[root@da9a4c6d8313 opt]# cd tableau
[root@da9a4c6d8313 tableau]# ls
docker_build tableau_driver tableau_server
[root@da9a4c6d8313 tableau]# cd tableau_driver
[root@da9a4c6d8313 tableau_driver]# lls
bash: lls: command not found
[root@da9a4c6d8313 tableau_driver]# ls
jdbc postgresql-odbc
[root@da9a4c6d8313 tableau_driver]# cd jdbc
[root@da9a4c6d8313 jdbc]# ls
kylin-jdbc-3.1.0.jar
[root@da9a4c6d8313 jdbc]#

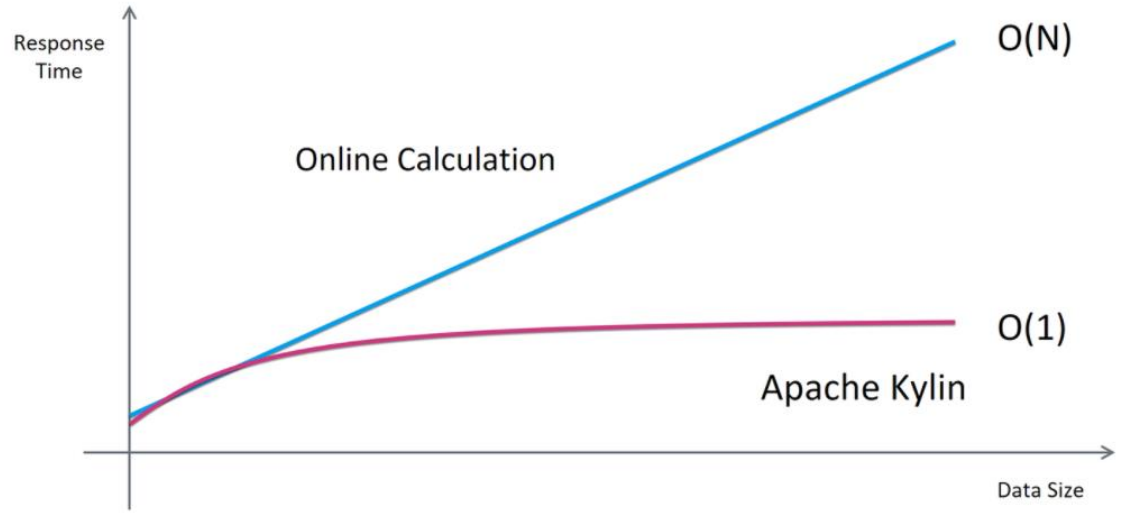

Other Databases (JDBC)
jdbc:kylin://192.41.170.144:7070/learn_kylin
URL:
bc:kylin://192.41.170.144:7070/AQMS
Dialect:
PostgreSQL
Enter information to log on to the server:
Username:
ADMIN
Password:
Properties File:
Browse...
Sign In

Software Architecture

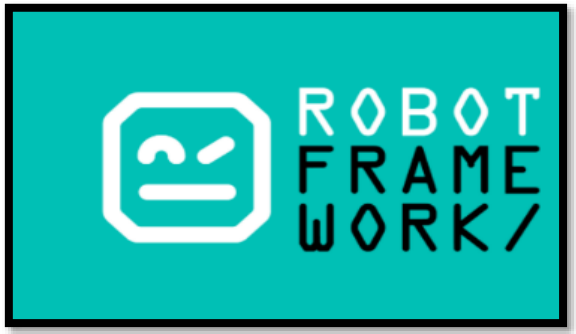


Apache Kylin

- Run Kylin with Docker
- Connecting Tableau Desktop and Tableau Server with Apache Kylin



Framework/Library



Design goals and trade-offs

Quality attributes	Priority Rating (L, M, H)	Priority Justification
Performance	H	The system shall be designed with high performance level to handle concurrent or multiple information from sensors which are placed at different locations in real time. Large numbers of data collected from the sensor should be displayed on the responsive web application and we need to focus on concurrency, response time and block time. ✓ The application should respond to a user within 2 seconds. ✓ The application should be able to handle **100 transactions per second in the peak load time**.
Usability	H	Once the system is deployed, the maintenance of the system including tasks such as monitoring the system, repairing problems that arise, updating or upgrading software components, should be easy to be sufficiently performed by any person with a basic understanding of the dashboard system. ✓ The web app should be **easy to operate by users with a certain navigation menu or option**. No need for a user manual.

Design goals and trade-offs

Portability	M	The system shall be designed in a way that allows it to be run on multiple computers with different browsers. As it is a web application, mobile phone web browsers can also access the application. ✓ The web app must support latest Web browsers for any OS. ✓ The web app should be responsive.
Reliability	M	Reliability is one of the key attributes of the system. Back-ups will be made regularly so that restoration with minimal data loss is possible in the event of unforeseen events. The system will also be thoroughly tested to ensure reliability. ✓ The system should be able to restore backward data of 24 hours (maximum 3 Days) within 2 hours as a recovery function.
Scalability	L	With data, the storage size will increase but can be managed with time. This app can be made horizontally scalable when there are issues of memory storage.

Access Matrix

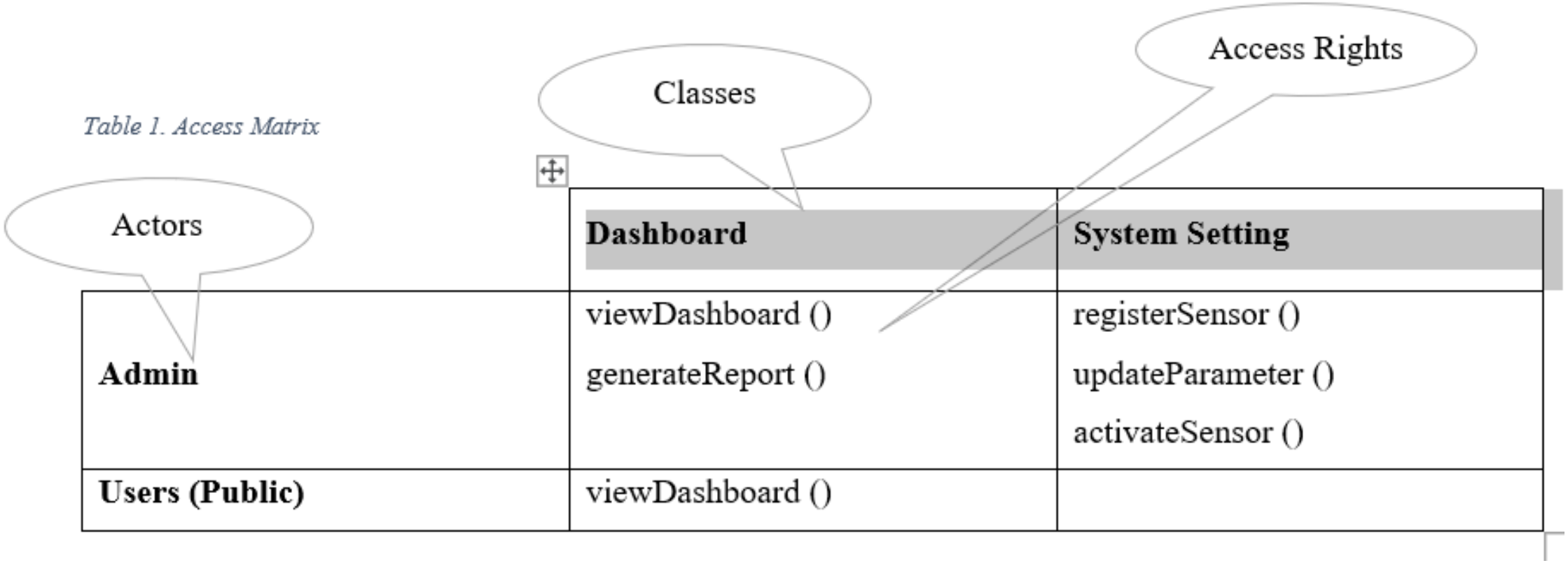
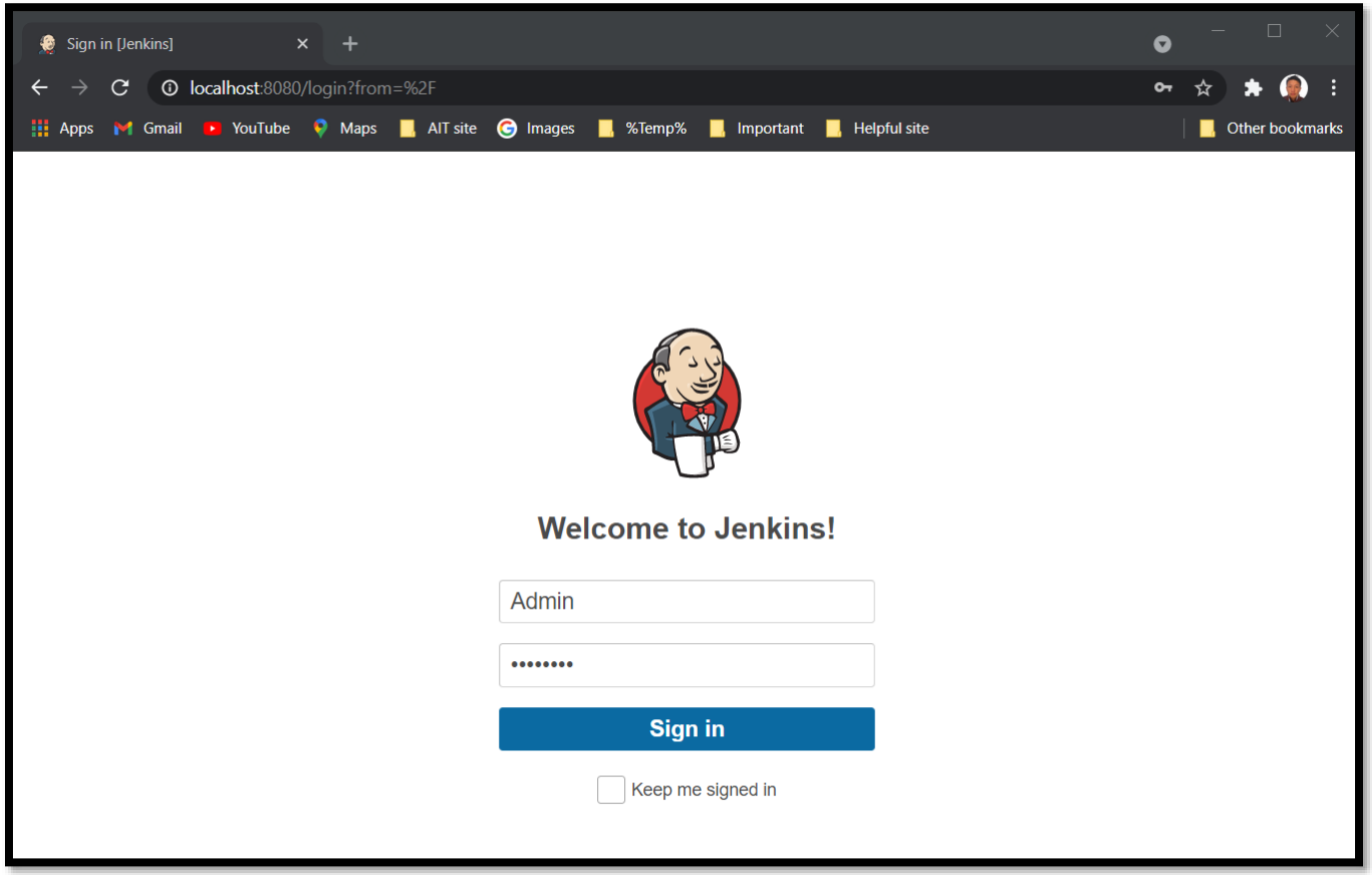


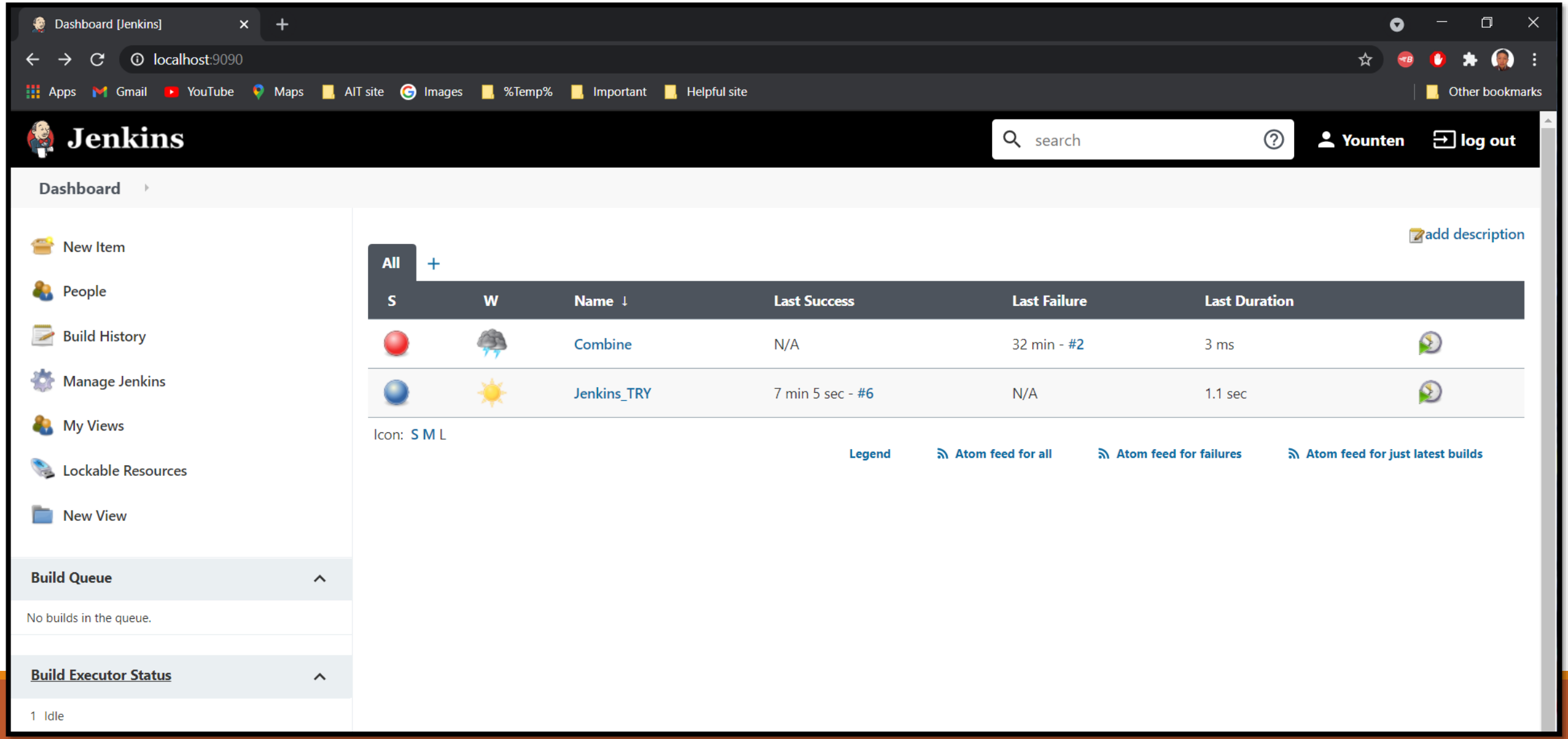
Table 1. Access Matrix

Actors	Dashboard	System Setting
Admin	viewDashboard () generateReport ()	registerSensor () updateParameter () activateSensor ()
Users (Public)	viewDashboard ()	

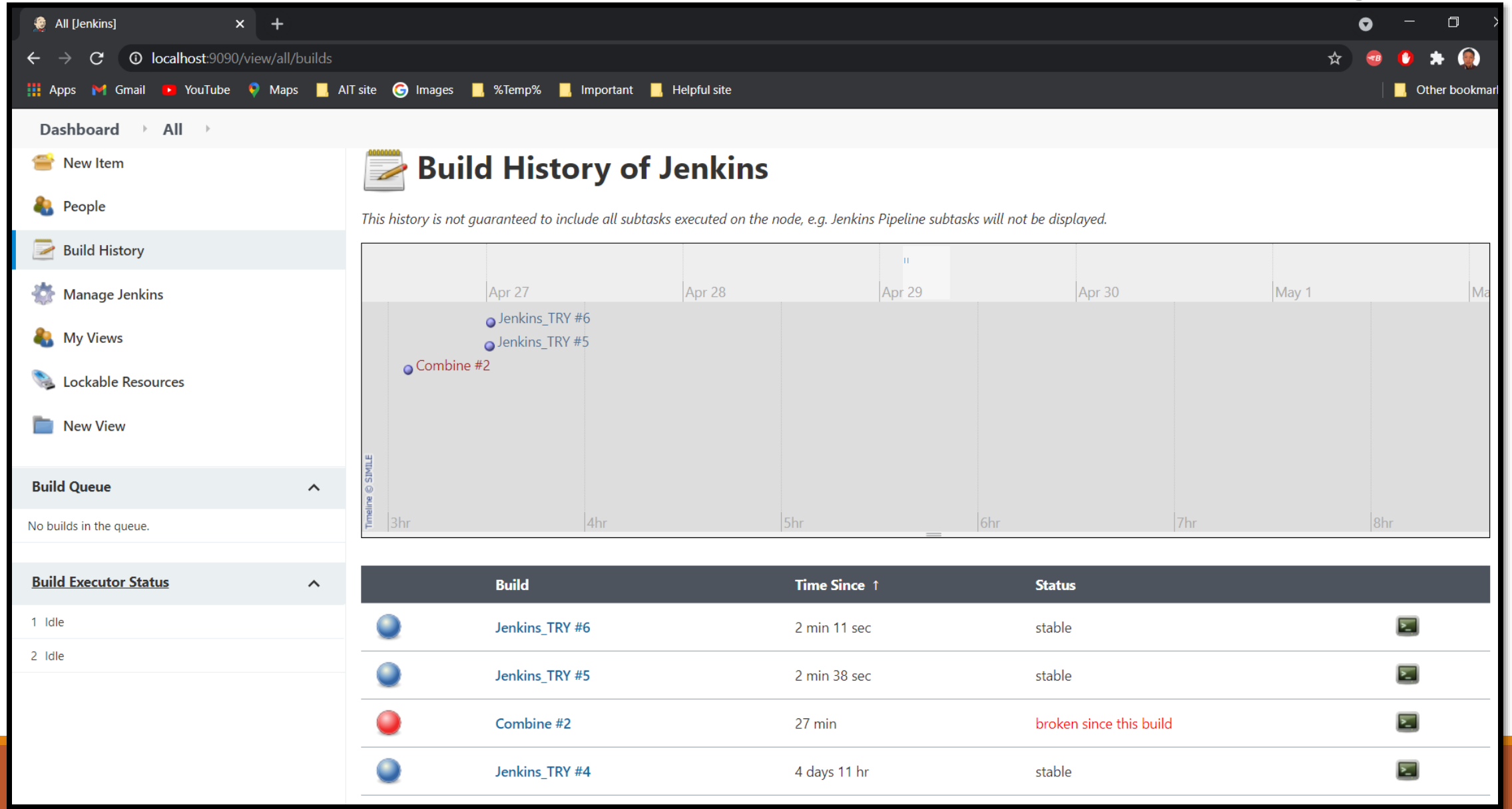
Continuous integration



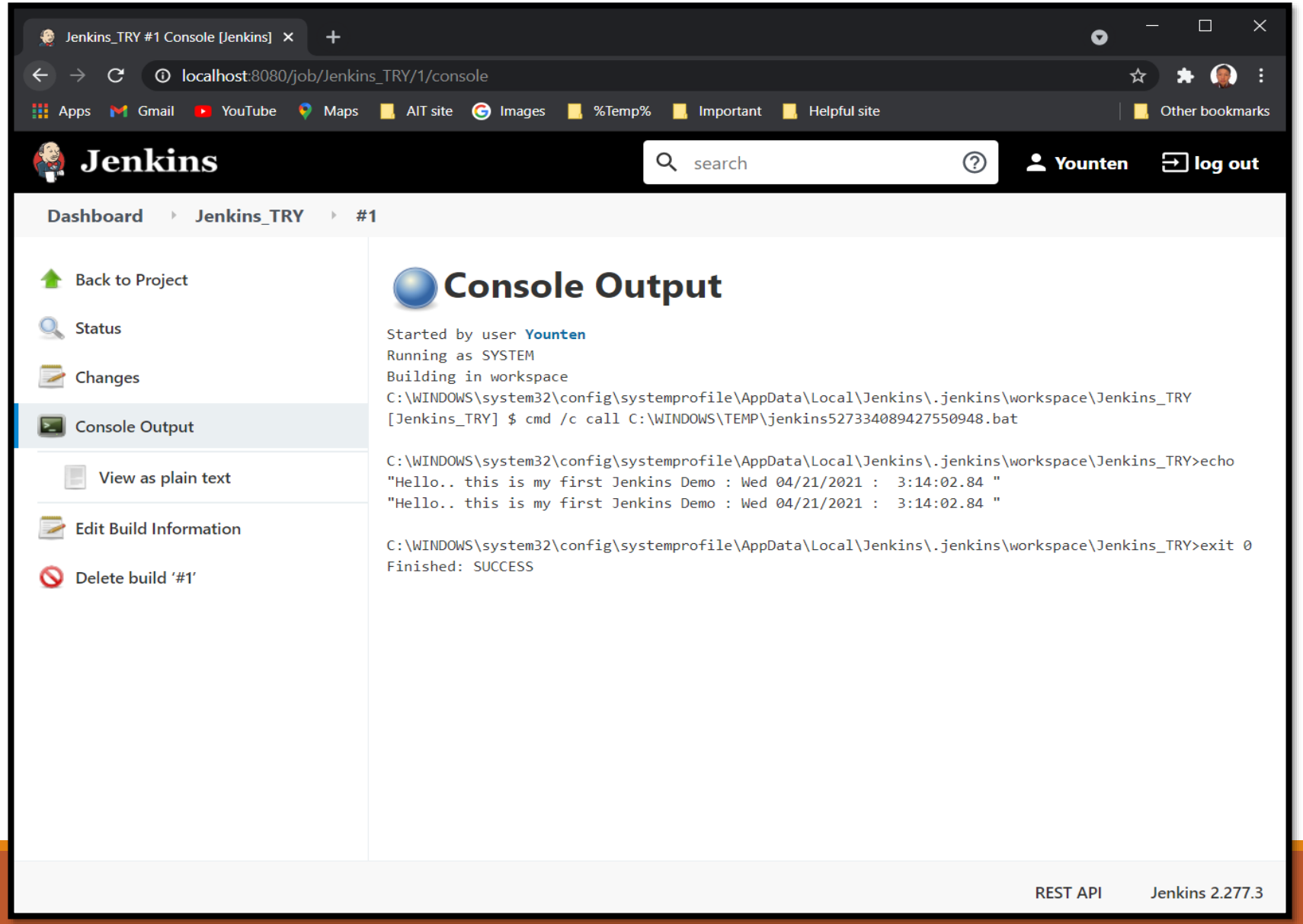
Creating Pipeline



Build and Build History



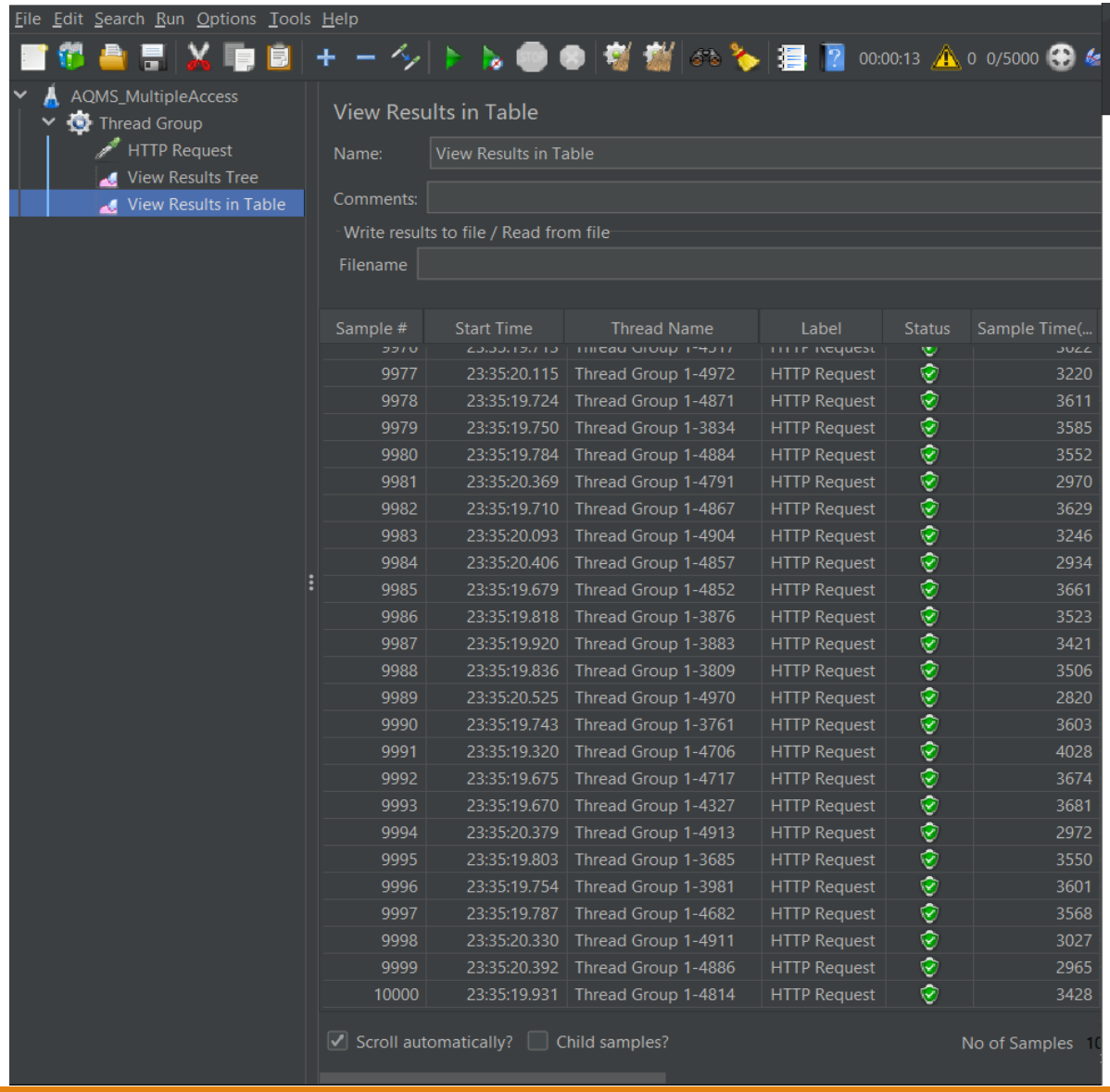
Result

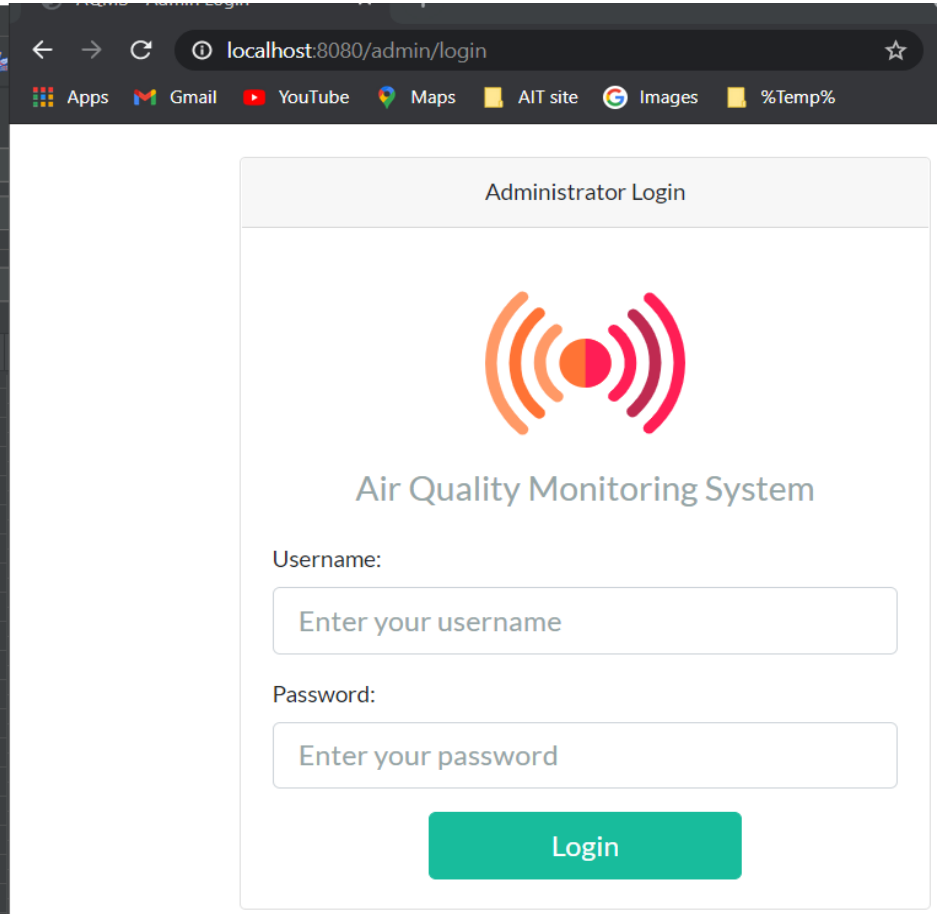

Jenkins_TRY #1 Console [Jenkins]
localhost:8080/job/Jenkins_TRY/1/console
Apps
Gmail
YouTube
Maps
AIT site
Images
%Temp%
Important
Helpful site
Other bookmarks
Jenkins
search
Younten
log out
Dashboard
Jenkins_TRY
#1
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#1'
Console Output
Started by user Younten
Running as SYSTEM
Building in workspace
C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\Jenkins_TRY
[Jenkins_TRY] $ cmd /c call C:\WINDOWS\TEMP\jenkins527334089427550948.bat
C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\Jenkins_TRY>echo
"Hello.. this is my first Jenkins Demo : Wed 04/21/2021 : 3:14:02.84 "
"Hello.. this is my first Jenkins Demo : Wed 04/21/2021 : 3:14:02.84 "
C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\Jenkins_TRY>exit 0
Finished: SUCCESS
REST API
Jenkins 2.277.3

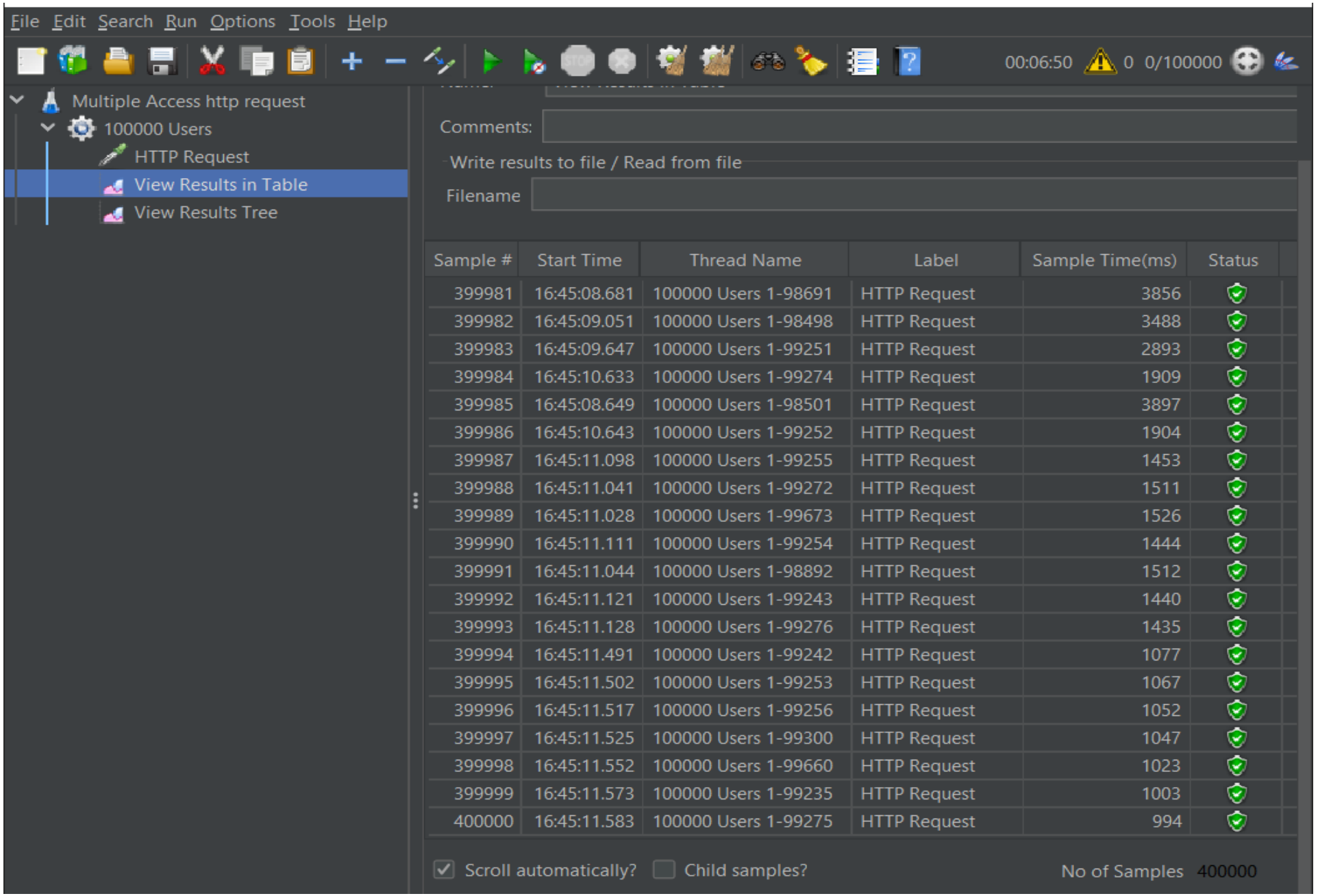
User Acceptance Test Report

- With **JMeter**, we can increase the number of concurrent users slowly or quickly throughout our test to record how performance is affected under continuous load.
- The idea behind concurrent user testing is to **identify the response of a web app for a specified number of concurrent users** making requests to our website. A concurrent user test is often used to identify bottlenecks in the performance of a website – *basically to find out how many concurrent users can make requests of a website until the performance of the site is significantly degraded.*
- **Robot Framework** is a **test automation framework** that is Python-based. Test Robot with simple webapp access by writing the script in VS Code with robot file extension and running will **show test case pass or fail**.

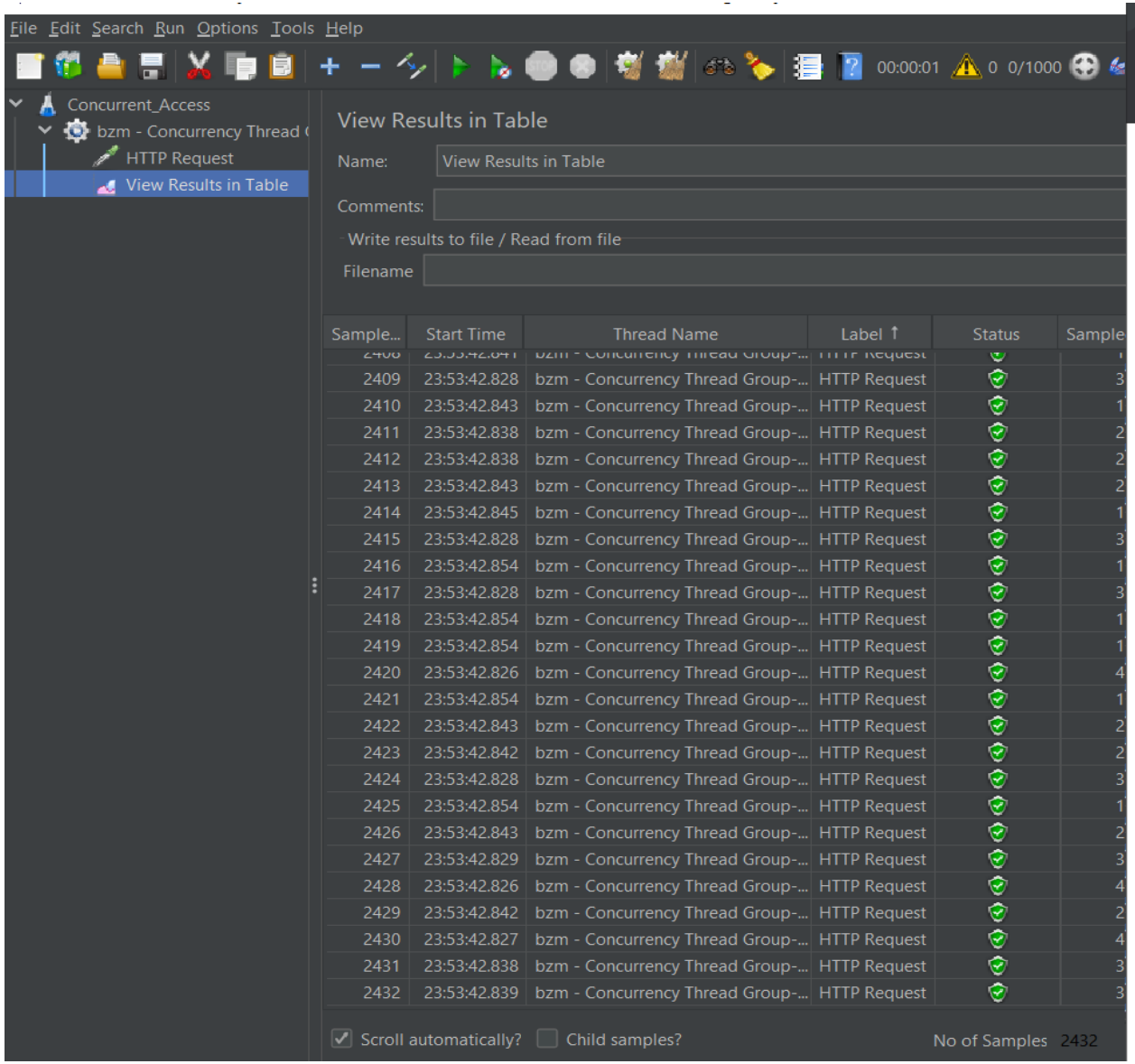
Test multiple access

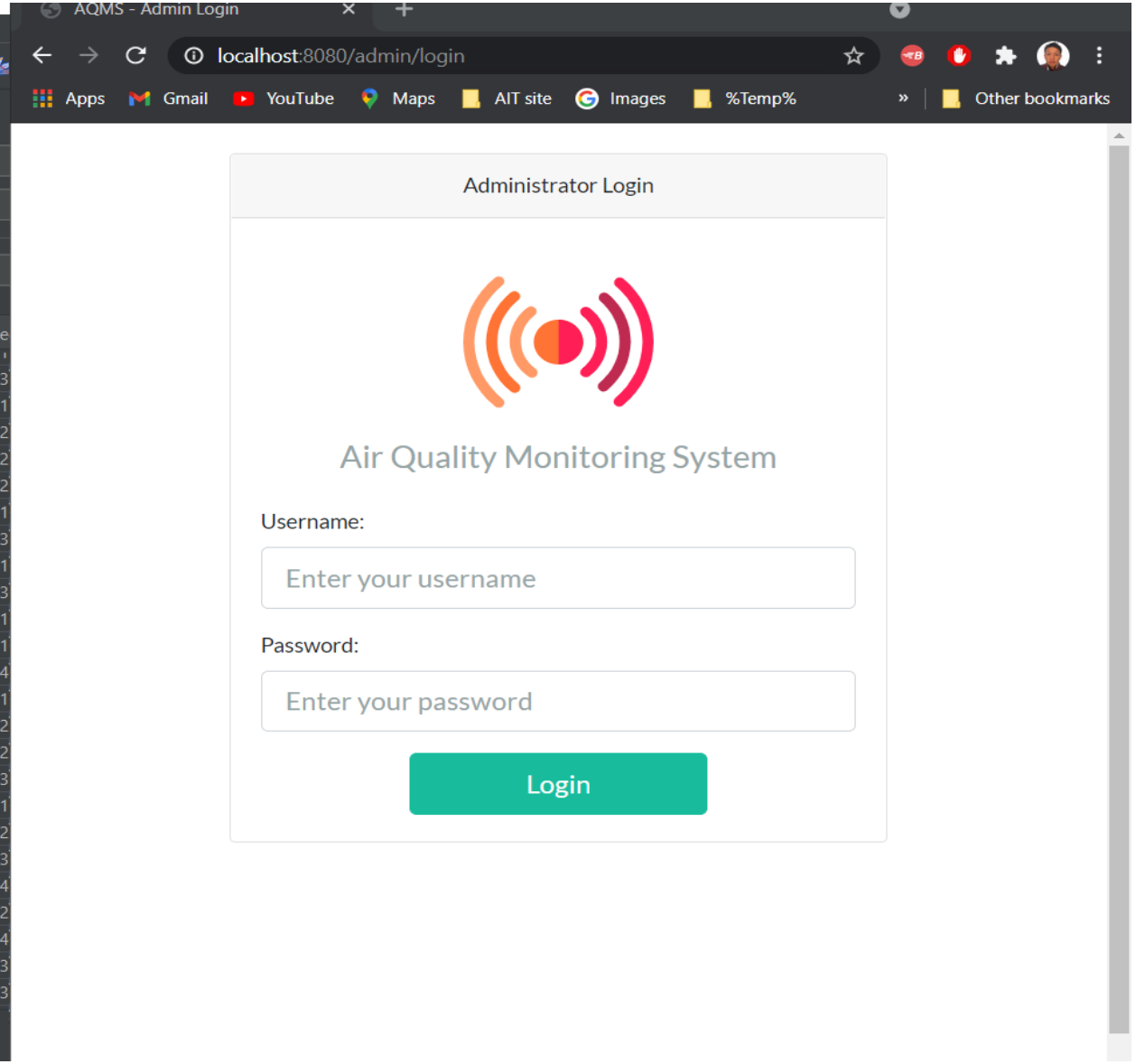




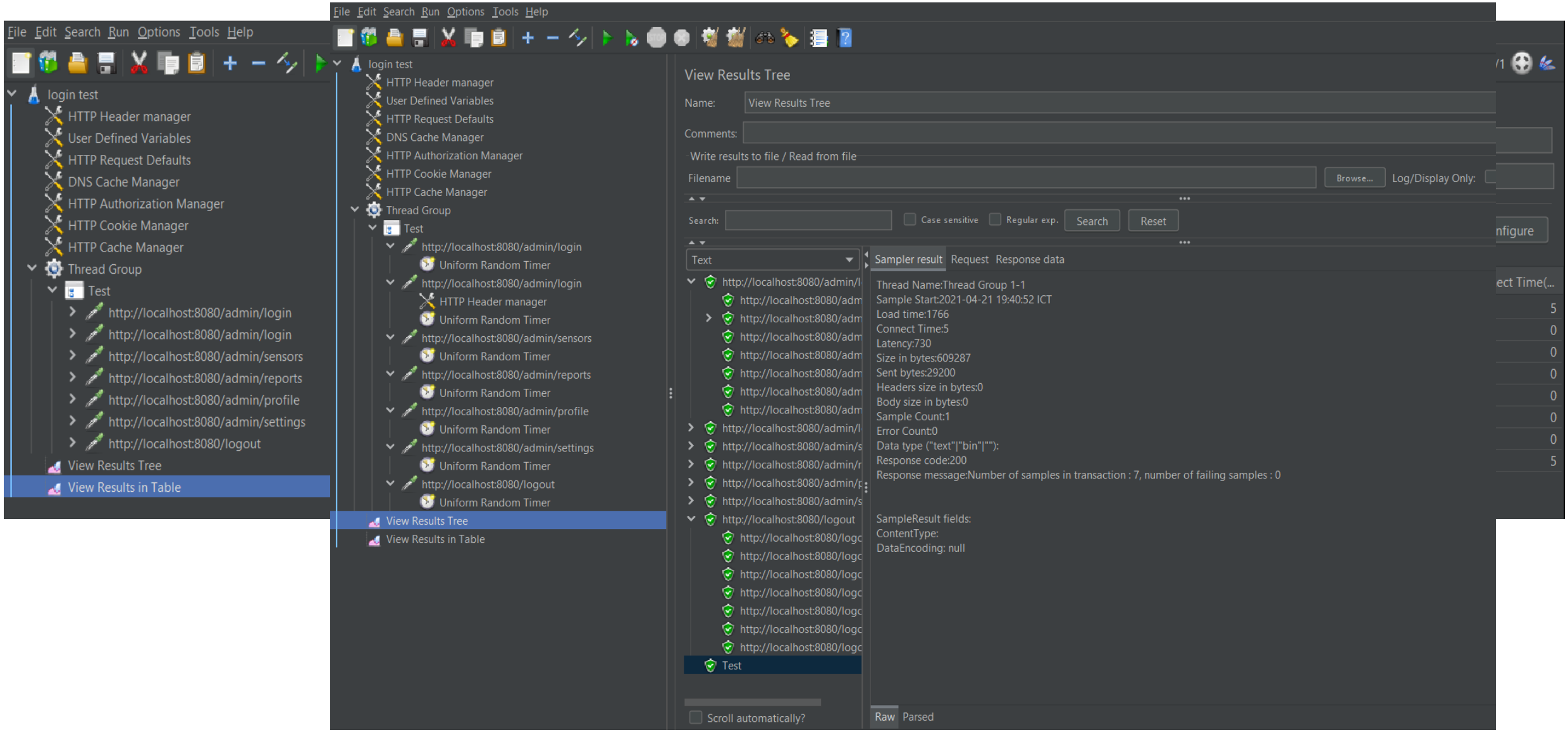

File Edit Search Run Options Tools Help
00:06:50 0 0/100000
Multiple Access http request
100000 Users
HTTP Request
View Results in Table
View Results Tree
Comments:
Write results to file / Read from file
Filename
Sample # Start Time Thread Name Label Sample Time(ms) Status
399981 16:45:08.681 100000 Users 1-98691 HTTP Request 3856
399982 16:45:09.051 100000 Users 1-98498 HTTP Request 3488
399983 16:45:09.647 100000 Users 1-99251 HTTP Request 2893
399984 16:45:10.633 100000 Users 1-99274 HTTP Request 1909
399985 16:45:08.649 100000 Users 1-98501 HTTP Request 3897
399986 16:45:10.643 100000 Users 1-99252 HTTP Request 1904
399987 16:45:11.098 100000 Users 1-99255 HTTP Request 1453
399988 16:45:11.041 100000 Users 1-99272 HTTP Request 1511
399989 16:45:11.028 100000 Users 1-99673 HTTP Request 1526
399990 16:45:11.111 100000 Users 1-99254 HTTP Request 1444
399991 16:45:11.044 100000 Users 1-98892 HTTP Request 1512
399992 16:45:11.121 100000 Users 1-99243 HTTP Request 1440
399993 16:45:11.128 100000 Users 1-99276 HTTP Request 1435
399994 16:45:11.491 100000 Users 1-99242 HTTP Request 1077
399995 16:45:11.502 100000 Users 1-99253 HTTP Request 1067
399996 16:45:11.517 100000 Users 1-99256 HTTP Request 1052
399997 16:45:11.525 100000 Users 1-99300 HTTP Request 1047
399998 16:45:11.552 100000 Users 1-99660 HTTP Request 1023
399999 16:45:11.573 100000 Users 1-99235 HTTP Request 1003
400000 16:45:11.583 100000 Users 1-99275 HTTP Request 994
Scroll automatically? Child samples? No of Samples 400000

Test Concurrent access





Testing login and other options



Test using Robot Framework

AQMS.robot M

AQMS > AQMS.robot > ...

```
*** Settings ***
Library  Selenium2Library

*** Variables ***
${expect}  LocationMind
${url}  http://localhost:8080
${Browser}  chrome
${delay}  1

*** Test Cases ***
1. Open Website
    Open Browser  ${url}  ${Browser}  options=add_experimental_option("excludeSwitches", ["enable-logging"])
    Maximize Browser Window
    Set Selenium Speed   0.5

2. Click on login link
   Click Link  //a[contains(text(),'Login')]

3. Input username and password
   Input Text  name=username  admin
   Input Text  name=password  admin

4. Login
   Click Button  name=submit

5. Check page info
   Click Link  //a[contains(text(),'Sensors')]

6. Start and Stop service
   Click Button  name=Stop
   Click Button  name=Start
   Click Link  //a[contains(text(),'Map View')]

7. Logout
   Click Link  //a[contains(text(),'Logout')]
8. close Browser
   Close Browser
```

Test using Robot Framework

```
PROBLEMS 9   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS> robot .\AQMS.robot
==============================================================================
AQMS
==============================================================================
1. Open Website                                                       | PASS |
------------------------------------------------------------------------------
2. Click on login link                                                | PASS |
------------------------------------------------------------------------------
3. Input username and password                                        | PASS |
------------------------------------------------------------------------------
4. Login                                                              | PASS |
------------------------------------------------------------------------------
5. Check page info                                                    | PASS |
------------------------------------------------------------------------------
6. Start and Stop service                                             | PASS |
------------------------------------------------------------------------------
7. Logout                                                             | PASS |
------------------------------------------------------------------------------
8. close Browser                                                      | PASS |
------------------------------------------------------------------------------
AQMS                                                                  | PASS |
8 tests, 8 passed, 0 failed
==============================================================================
Output:  C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS\output.xml
Log:     C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS\log.html
Report:  C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS\report.html
```

The status of the implementation

- Link to our GitHub with readme file about the project and status of the implementation.
https://github.com/shubhanginigon/SDQI2021_G1

- Link to our JIRA project with tasks and roadmap:
https://sqdi2021g1.atlassian.net/jira/software/c/projects/AQMS/boards/4/roadmap

- Link to folder having installation and usage instruction as System Documentation and User Documentation, respectively.
- Updated Software Requirement Specification and Design Documents(Final Document)
https://github.com/shubhanginigon/SDQI2021_G1/tree/main/Related%20Documents/Final%20Report%20Document

CONCLUSION

- The system ‘Air Quality (PM2.5) Monitoring System’ will be especially designed to be used by any user who are cautious about their health.

Future Scope:

1. Add more sensor data sources and test
2. Deploy in real servers

Real Demo

References

Apache Kylin. (2015). Bring OLAP back to big data! Retrieved from Apache Kylin | Analytical Data Warehouse for Big Data

Fann,N.,& Risley,D. (2011,January 5). The public health context for PM2.5 and ozone air quality trends. Air Qual Atmos Health 6, 1–11 (2013). https://doi.org/10.1007/s11869-010-0125-0

Geetha,S.M.N. (2021, March 19). Hadoop for Analyst-Apache Druid, Apache Kylin and Interactive query tools. Retrieved from https://www.saigeetha.in/post/hadoop-for-analysts-apache-druid-apache-kylin-and-interactive-query-tools?fbclid=IwAR0RRXXxKmv8onswnS-g5mV5Hh_L5R9zOSWly6YO8d4kb6oYYW4rrjF5wlo

Gupta,A.k., & Johari,R. (2019). IOT based electrical device surveillance and control system. International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), https://doi.org/10.1109/IoT-SIU.2019.8777342

Nethu, M.V. (2018, September 25). OLAP on Hadoop-Apache Kylin. Retrieved from https://medium.com/@mvneethu90/olap-in-hadoop-apache-kylin-bf0377d8b44f

Sinha, S. (2016, October 28). Hadoop ecosystem- Get to know the Hadoop tools for crunching Big Data. edureka. Retrieved from https://medium.com/edureka/hadoop-ecosystem-2a5fb6740177

Sinha,S. (2014, October 9). Hadoop tutorial- A comprehensive guide to Hadoop. edureka. Retrieved from https://medium.com/edureka/hadoop-tutorial-24c48fbf62f6

Questions and Feedback