

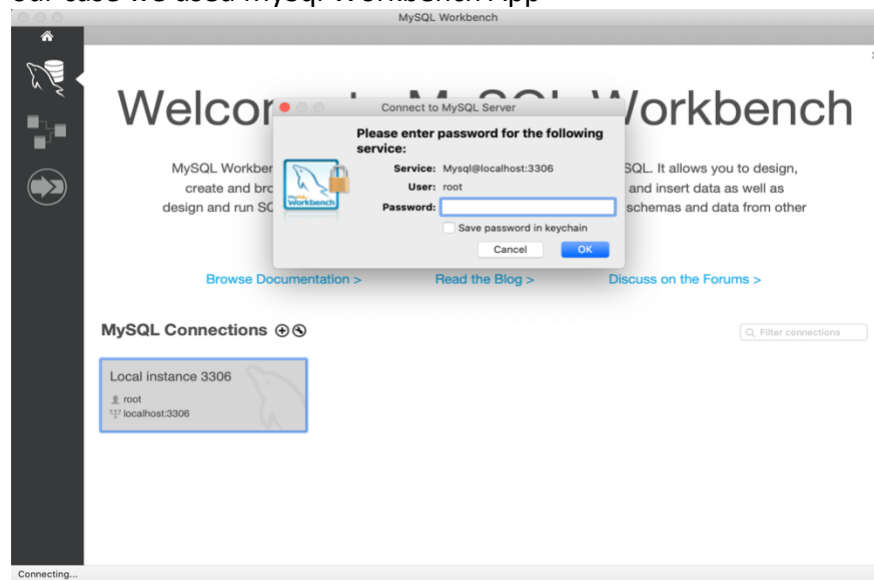
Miners Team

Loans Repayment Use Case

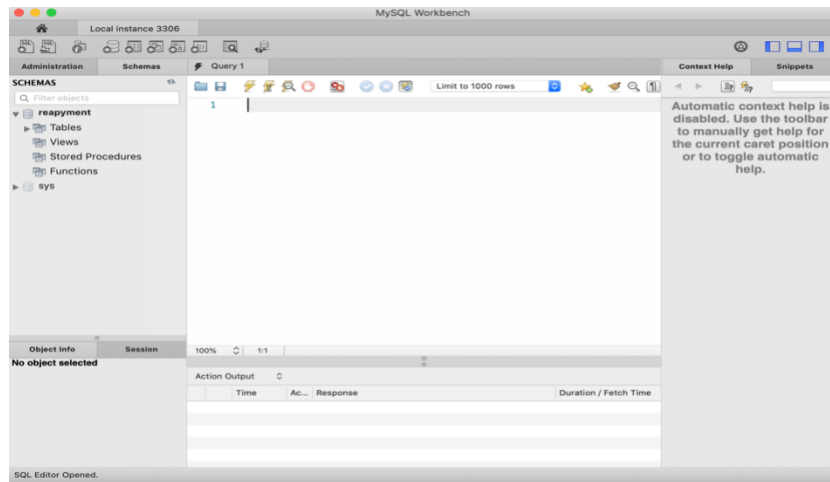
This is a “how to” guide to install, configure and run the use case correctly

Important tools

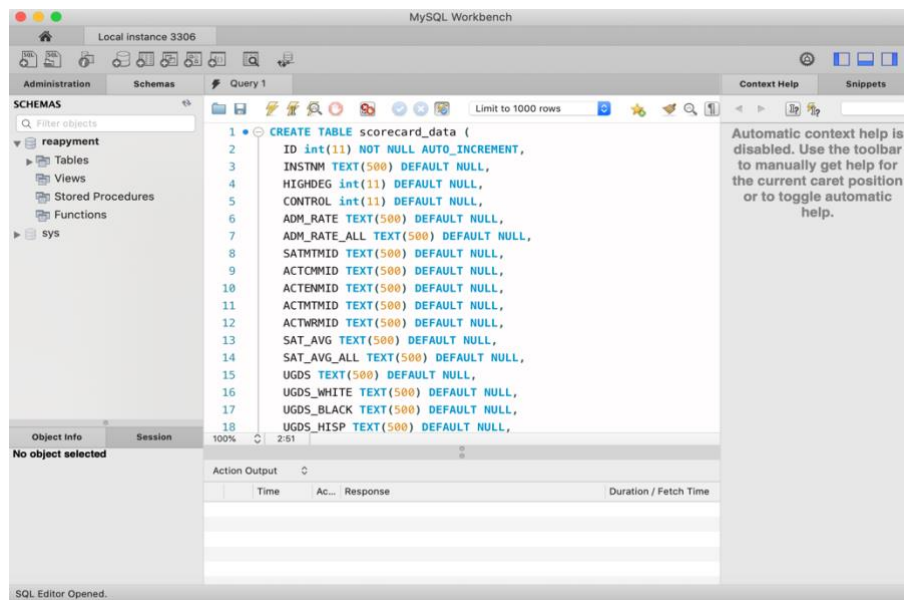
- 1- Install Python and make sure its environment variable is set correctly.
- 2- Install Java and be sure to set JAVA_PATH environment variable
- 3- install Hadoop on your machine and HDFS and configure them correctly and start hdfs
 - a. We followed the link below for installation and configuration of our machine
 - i. <https://gist.github.com/viecode09/ad56b09bea4da59b4240d45b666321cf>
 - b. Create hdfs directory with name “repayment” using this command “hdfs dfs -mkdir /repayment”
- 4- Install Spark
 - a. We followed the link below for installation and configuration of our machine
 - i. <https://www.tutorialkart.com/apache-spark/how-to-install-spark-on-macos/>
- 5- install MySQL (just download the binary and install it)
- 6- Configure database
 - First we used ETL tool to import excel files to SQL engine. To make our work reproducible, we generated the needed queries to get exact image of our database. Please follow the steps below to get exact image from our database:
 - 1- Open MySQL from any client application and connect to database server. In our case we used MySQL Workbench App



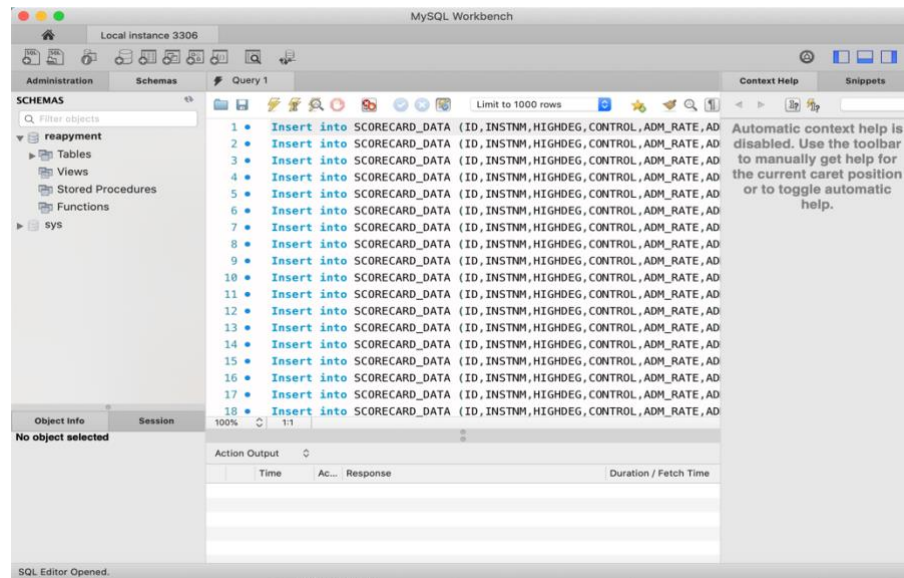
- 2- create a database with name “repayment”



- 3- Use the sql script that is in create_mysql_table_script.txt file to create table with the same name we use in our workflow

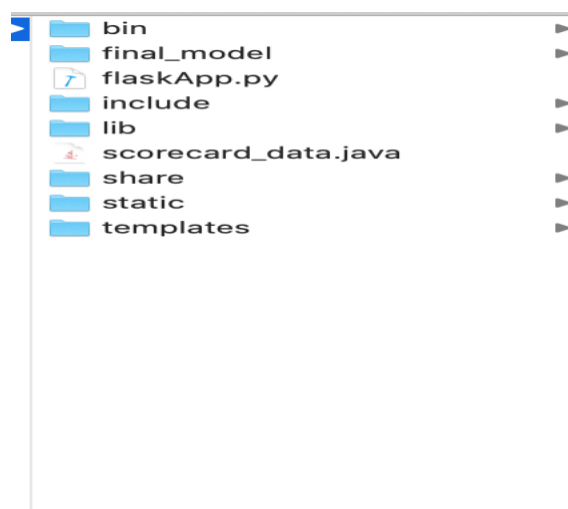


- 4- Use the sql script that is in scorecard_data_insertion.txt to import data the data to this table



7- Install Flask Server

- We followed this link
<https://gist.github.com/dineshviswanath/af72af0ae2031cd9949f>
- Then copy all files from end-user-app/ repayment-api to your installation directory.
 This is how our installation directory looks like after copying



- Some files point to this directory as part of their work. Flask listens on final_model subdirectory to detect changes automatically and redeploy the new model. Also the training script save the trained model to this directory (final_model), so you have to change them to your new directory.
- Open data-engineering-part/engineering-app.py file, and change the directory to your new directory

```

24 for index in range(len(dataFrame.columns)):
25     dataFrame = dataFrame.withColumnRenamed(dataFrame.columns[index], dataFrame.columns[index].replace("C
26     return dataFrame
27
28 f __name__ == "__main__":
29
30     sconf = SparkConf().setAppName("Repayment-Project")
31     # sc = SparkContext(conf=sconf)
32     spark = SparkSession.builder \
33         .config('spark.master', 'local[2]') \
34         .appName('Final-Project') \
35         .getOrCreate()
36     sc = spark.sparkContext
37     web_app_model_path = "/Users/apple/Desktop/flask_projects/repayment-api/final_model"
38     data_path = "hdfs://localhost:9000/repayment/scorecard.txt"
39
40     db_cols = ["ID", "INSTNM", "HIGHDEG", "CONTROL", "ADM_RATE", "ADM_RATE_ALL", "SATMTMID", "ACTCOMID", "ACT
41     scorecard_data_rdd = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load
42     scorecard_data = scorecard_data_rdd.toDF(*db_cols)
43
44
45     features = ["CONTROL", "ADM_RATE", "ADM_RATE_ALL", "SAT_AVG_ALL", "SATMTMID", "UGDS", "HIGHDEG", "TUITFTE",
46               "COSTT4_A", "PCTFLOAN", "COMP_ORIG_YR2_RT", "UGDS_WHITE", "UGDS_BLACK", "UGDS_HISP", "UGDS_ASIAN", "UGDS_AI
47     categ_features = ["CONTROL", "HIGHDEG"]
48     target = "COMPL_RPY_1YR_RT"
49
50     # this preprocessing step is important only in training, so it will not be included in the pipeline.
51     scorecard_data_cleaned = cleanPrivacySuppressed(scorecard_data)
52     # select not null target rows
53     scorecard_data_cleaned = scorecard_data_cleaned.where(F.col(target).isNotNull()).where(F.col(target) != np.
54
55     # hot encoder preprocessing
56     hot_encoder = OneHotEncoderEstimator(inputCols=categ_features, outputCols=["(0)_ENCODED".format(colName)
57
58     #vector assembler
  
```

- Also don't forget to do the same step above for Open data-engineering-part/engineering-notebook.py if you want to check the engineering steps using jupyter notebook file.

8- Start Flask Server

- From cmd, goto installation path of your flask project (the path where you put end-user-app/repayment-api files in)
- Type "source bin/activate". Then "python flaskApp.py"
- Now, open the browser and type <http://127.0.0.1:5000/>
- You should get a page like this

Enter Model Hyperparameters	
AVERAGE AGE OF ENTRY AGE_ENTRY	AVERAGE SAT SAT_AVG_ALL
MATH SAT SCORES SATMTMID	NUMBER OF UNDERGRADUATES UGDS
NUMBER OF NON-WORKING STUDENTS COUNT_NWNE_P10	NUMBER OF WORKING STUDENTS COUNT_WNE_P10
MEAN EARNINGS OF WORKING STUDENTS MN_EARN_WNE_P10	MEDIAN EARNINGS OF WORKING STUDENTS MD_EARN_WNE_P10
NET TUITION REVENUE PER FTE STUDENT TUITFTE	INSTRUCTIONAL EXPENDITURES PER FTE STUDENT INEXPFTE
IN-STATE TUITION AND FEES TUITIONFEE_IN	OUT-OF-STATE TUITION AND FEES TUITIONFEE_OUT
PROGRAM-YEAR TUITION AND FEES TUITIONFEE_PROG	AVG COST OF ACADEMIC-YEAR ATTENDANCE COSTT4_A

e. Fill the form and press “Build Model”

The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000". The page contains a form with two columns of input fields. The left column includes fields for "UGDS_ASIAN", "NUMBER OF NATIVE HAWAIIAN UNDERGRADUATES" (UGDS_NHPI), "NUMBER OF NON-RESIDENT UNDERGRADUATES" (UGDS_NRA), "% UNDERGRADUATES RECEIVING FEDERAL LOANS" (PCTFLOAN), "% PART-TIME UNDERGRADUATES" (PPTUG_EF), "COMPLETION RATE WITHIN 3-YEARS" (COMP_ORIG_YR3_RT), "COMPLETION RATE WITHIN 3-YEARS WITH FEDERAL LOANS" (LOAN_COMP_ORIG_YR3_RT), "CONTROL", and "DEGREE TYPE". The right column includes fields for "UGDS_AIAN", "NUMBER OF UNDERGRADUATES WITH 2+ MORE RACES" (UGDS_2MOR), "NUMBER OF UNDERGRADUATES WITH UNKNOWN RACE" (UGDS_UNKN), "% UNDERGRADUATES RECEIVING PELL GRANT" (PCTPELL), "COMPLETION RATE WITHIN 2-YEARS" (COMP_ORIG_YR2_RT), "COMPLETION RATE WITHIN 4-YEARS" (COMP_ORIG_YR4_RT), "PERCENT DIED WITHIN 4-YEARS" (DEATH_YR4_RT), and "DEGREE TYPE". A yellow "Build Model" button is located at the bottom center of the form.

f. The, you get a prediction for the institute with the provided information

The screenshot shows the same web browser window, but the address bar now displays "127.0.0.1:5000/predict". The page title is "Miners Model Client Interface". A red circle highlights the text "prediction is: 0.5210390628492307". Below this, there is a section titled "Enter Model Hyperparameters" with two columns of input fields. The left column includes fields for "AVERAGE AGE OF ENTRY" (AGE_ENTRY), "MATH SAT SCORES" (SATMTMID), "NUMBER OF NON-WORKING STUDENTS" (COUNT_WNE_P10), "MEAN EARNINGS OF WORKING STUDENTS" (MN_EARN_WNE_P10), "NET-TUITION REVENUE PER FTE STUDENT" (TUITFTE), "IN-STATE TUITION AND FEES" (TUITIONFEE_IN), and "PROGRAM YEAR TUITION AND FEES". The right column includes fields for "AVERAGE SAT" (SAT_AVG_ALL), "NUMBER OF UNDERGRADUATES" (UGDS), "NUMBER OF WORKING STUDENTS" (COUNT_WNE_P10), "MEDIAN EARNINGS OF WORKING STUDENTS" (MD_EARN_WNE_P10), "INSTRUCTIONAL EXPENDITURES PER FTE STUDENT" (INEXPFTE), "OUT-OF-STATE TUITION AND FEES" (TUITIONFEE_OUT), and "AVG COST OF ACADEMIC YEAR ATTENDANCE".

9- Install Azkaban Scheduler

a. Clone the github repository of this link

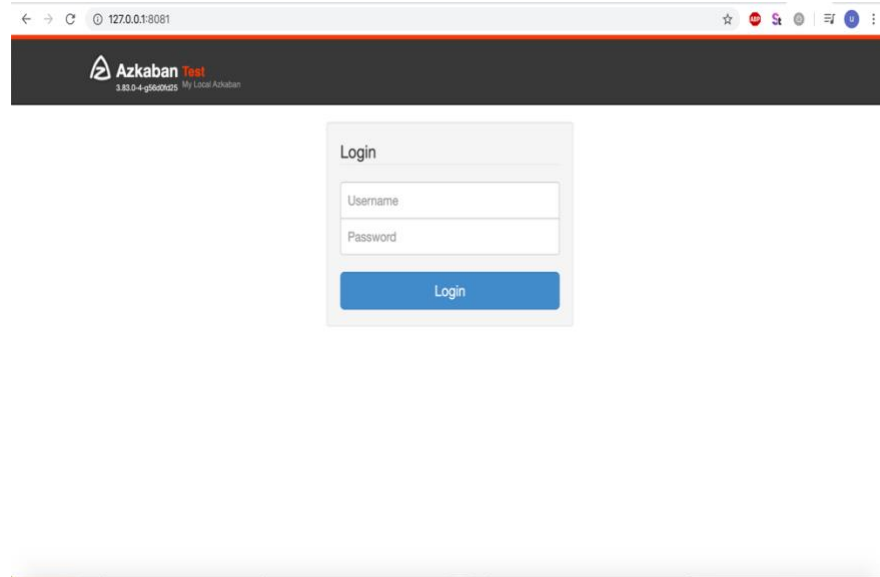
i. <https://github.com/azkaban/azkaban>

ii. From terminal, go to the directory of the cloned github project above and type `./gradlew build`

iii. Start Azkaban server by going to `azkaban-solo-server/build/install/azkaban-solo-server` from cmd and type `bin/start-solo.sh`

- iv. Now you can start the GUI by typing in the web browser

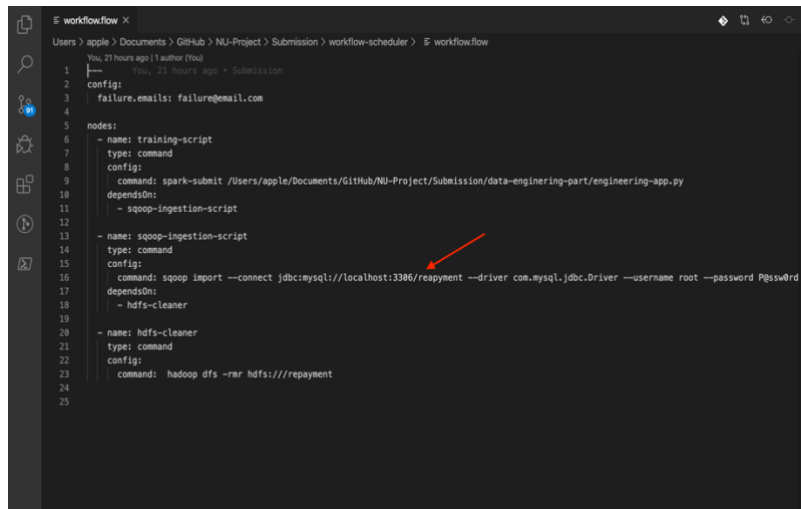
<http://127.0.0.1:8081>



- v. The default username and password are “azkaban”
- vi. Open workflow-scheduler/workflow.flow and edit the following
1. Edit the path of data-engineering-part/engineering-app.py to your path on your machine

A screenshot of a code editor showing the 'workflow.flow' file. The file is located at 'Users > apple > Documents > GitHub > NU-Project > Submission > workflow-scheduler > workflow.flow'. The code defines a workflow with three nodes: 'training-script', 'sqoop-ingestion-script', and 'hdfs-cleaner'. The 'training-script' node's command is highlighted with a red oval: 'command: spark-submit /Users/apple/Documents/GitHub/NU-Project/Submission/data-engineering-part/engineering-app.py'. The 'sqoop-ingestion-script' node's command is 'command: sqoop import --connect jdbc:mysql://localhost:3306/repayment --driver com.mysql.jdbc.Driver --username root --password P@ssw0rd'. The 'hdfs-cleaner' node's command is 'command: hadoop dfs -rmr hdfs:///repayment'.

2. Configure sqoop script in the same file with your system configuration



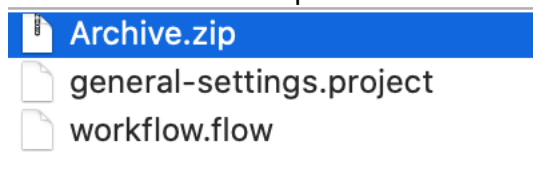
```
1 1
2 config:
3 failure.emails: failure@gmail.com
4
5 nodes:
6 - name: training-script
7   type: command
8   config:
9     command: spark-submit /Users/apple/Documents/GitHub/NU-Project/Submission/data-engineering-part/engineering-app.py
10  dependsOn:
11    - sqoop-ingestion-script
12  - name: sqoop-ingestion-script
13    type: command
14    config:
15      command: sqoop import --connect jdbc:mysql://localhost:3306/repayment --driver com.mysql.jdbc.Driver --username root --password P@ssw0rd
16    dependsOn:
17      - hdfs-cleaner
18  - name: hdfs-cleaner
19    type: command
20    config:
21      command: hadoop dfs -rmr hdfs:///repayment
22
23
24
25
```

a.

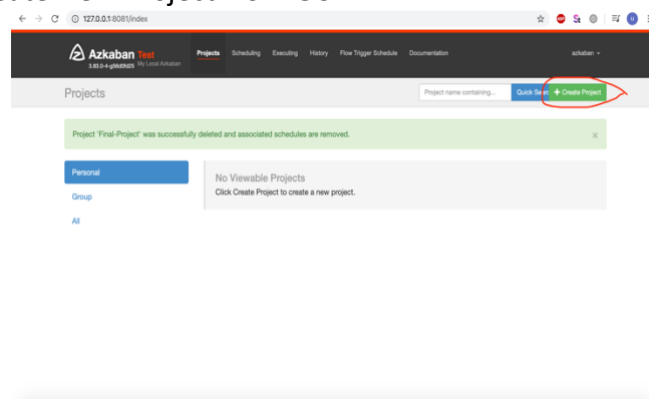
b. The changes you must do to the script are

- Change “import --connect jdbc:mysql://localhost:3306/repayment” to your Mysql port instead of 3306
- Change username and password of database to yours “--username root --password P@ssw0rd”
- Download SQL connector jar file from internet and put it in sqoop lib folder and mention this folder in “--bindir your-sqoop-lib-path-folder” option instead of current configuration “--bindir /usr/local/Cellar/sqoop/1.4.6_1/libexec/lib”

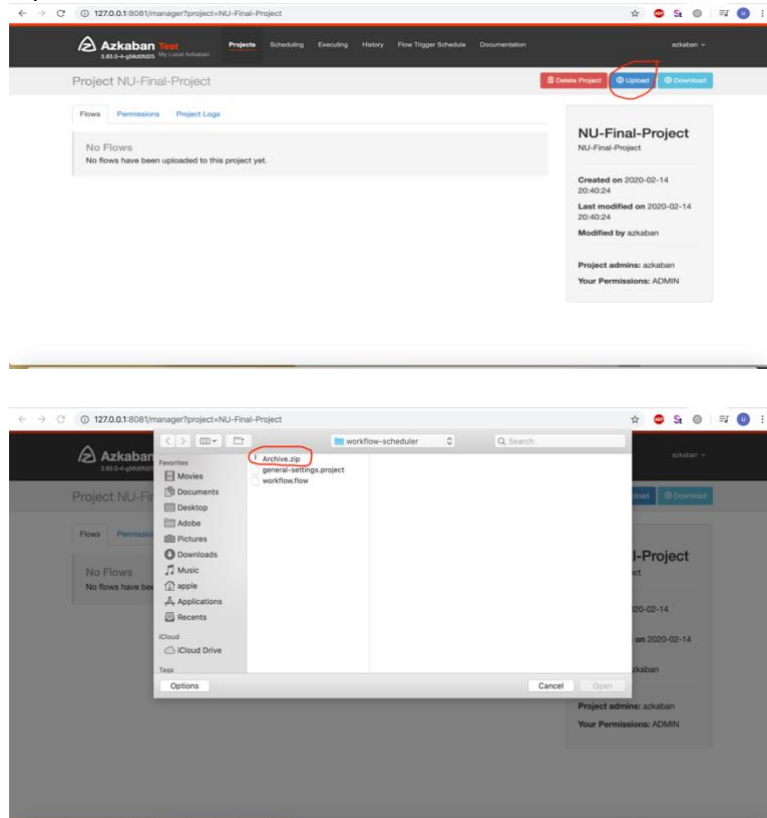
3. Compress/Archive workflow-scheduler/general-settings.project file and workflow-scheduler/workflow.flow file after modifications above to one zip file as shown below



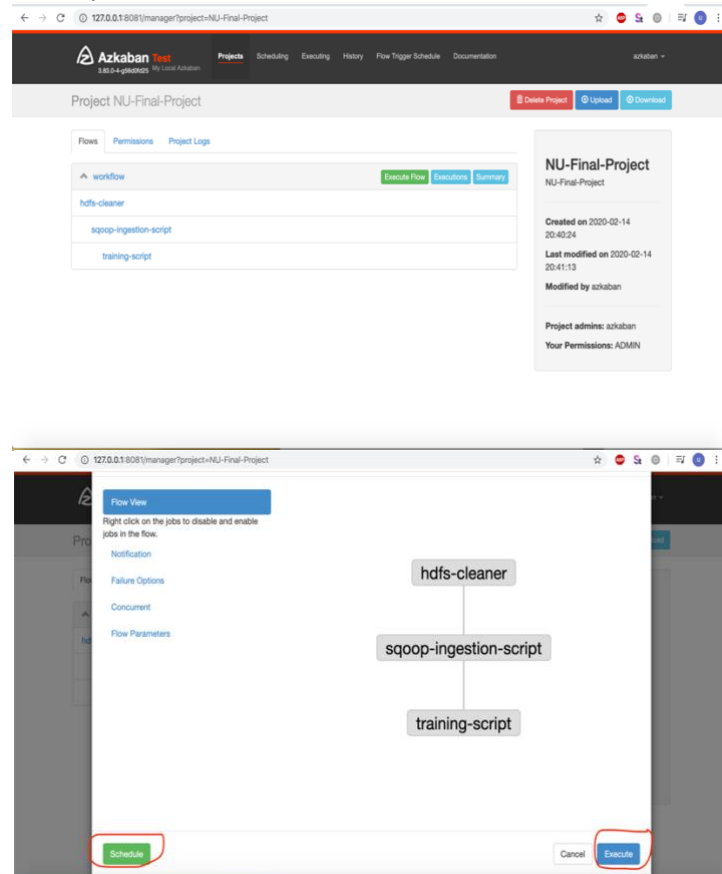
4. Create New Project from GUI



5. Upload archived file to Azkaban server




6. Execute/Schedule workflow from “Execute Flow” button



7. Monitor Working Jobs

← → 127.0.0.1:8081/executor?execid=14#jobslist ☆ 🔍 🌐 ⚙️

 **Azkaban** Test
3.83.0-4-g6a09d5c My Local Azkaban

[Projects](#) [Scheduling](#) [Executing](#) [History](#) [Flow Trigger Schedule](#) [Documentation](#) [azkaban](#)

Flow Execution 14 **RUNNING**

Submit User
azkaban

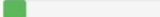

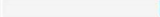
Duration 33 sec

Start Time 2020-02-15 07:55
26s
End Time -

[Project NU-Final-Project](#) / [Flow workflow](#) / Execution 14

[Graph](#) [Flow Trigger List](#) [Job List](#) [Flow Log](#) [Stats](#)

[Kill](#) [Pause](#)

Name	Type	Timeline	Start Time	End Time	Elapsed	Status	Details
hdfs-cleaner	command		2020-02-15 07:55 26s	2020-02-15 07:55 30s	4 sec	Success	Log
sqoop-ingestion-script	command		2020-02-15 07:55 30s	2020-02-15 07:55 56s	25 sec	Success	Log
training-script	command		2020-02-15 07:55 56s	-	3 sec	Running	Log