

Garmin Workshop

Resources

Web Server: <http://34.206.1.88>

Registration code: **garmin**

```
SSH username: centos
WEB SERVER VM:
=====
Web Server Name          Public DNS Name           Public IP      Private IP      Stoppable
jhorsch-garmin-web       cdp.34.206.1.88.nip.io 34.206.1.88  10.0.1.211  No
CLUSTER VMS:
=====
Cluster Name             Public DNS Name           Public IP      Private IP      Stoppable
jhorsch-garmin-cluster-0 cdp.44.204.245.46.nip.io 44.204.245.46 10.0.1.206  No
jhorsch-garmin-cluster-1 cdp.44.199.252.75.nip.io 44.199.252.75 10.0.1.16   No
jhorsch-garmin-cluster-2 cdp.18.209.214.123.nip.io 18.209.214.123 10.0.1.96   No
jhorsch-garmin-cluster-3 cdp.3.237.182.114.nip.io 3.237.182.114 10.0.1.30   No
jhorsch-garmin-cluster-4 cdp.18.209.210.179.nip.io 18.209.210.179 10.0.1.202  No
jhorsch-garmin-cluster-5 cdp.44.193.24.187.nip.io 44.193.24.187 10.0.1.177  No

Health checks:
instance               ip address        WEB  CM    EFM   NIFI  NREG  SREG  SMM   HUE   SSB   CDSW  Model Status  Viz Status
aws_instance.web        34.206.1.88     Ok
aws_instance.cluster[0] 44.204.245.46  Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok
aws_instance.cluster[1] 44.199.252.75  Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok
aws_instance.cluster[2] 18.209.214.123 Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok
aws_instance.cluster[3] 3.237.182.114 Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok
aws_instance.cluster[4] 18.209.210.179 Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok
aws_instance.cluster[5] 44.193.24.187  Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok   Ok
```

CML/CDSW Labs

Experiments and Models

Labs summary

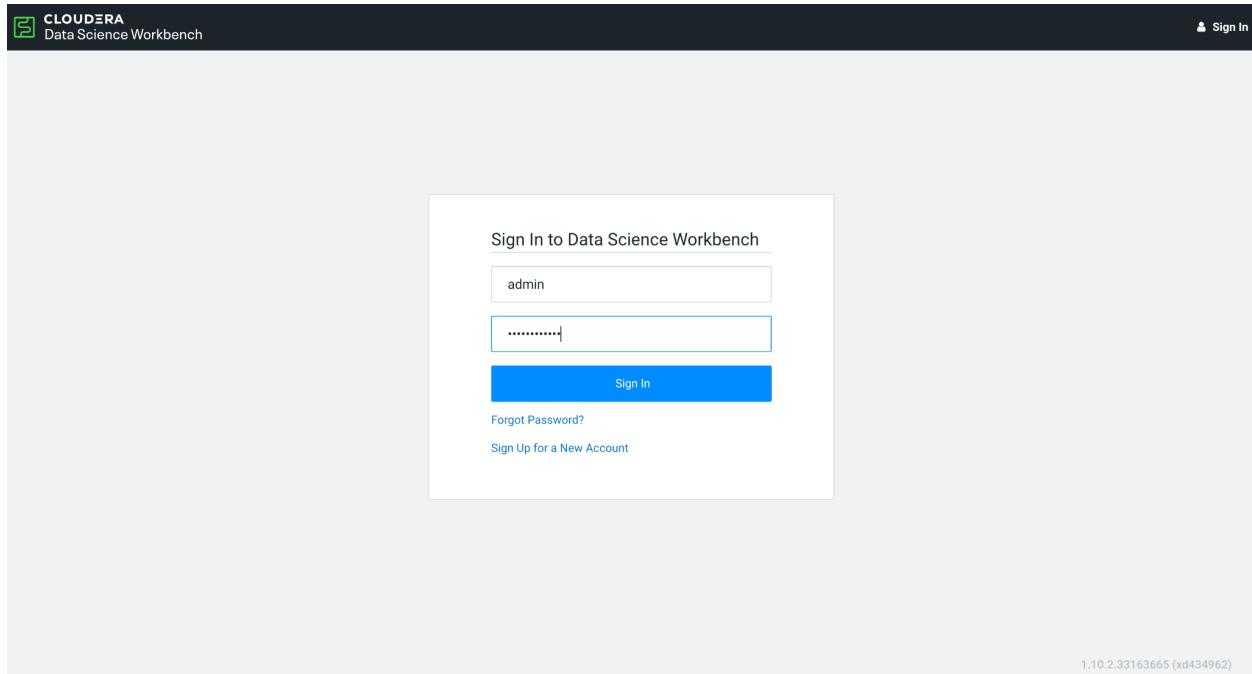
- Lab 1 - Train the model.
- Lab 2 - Deploy the Model.

Lab 1 - CDSW: Train the model

In this and the following lab, you will wear the hat of a Data Scientist. You will write the model code, train it several times and finally deploy the model to Production.

STEP 1: Configure CDSW

1. Open CDSW Web UI and log in as admin/Supersecret1, if you haven't yet done so.



2. Navigate to the CDSW Admin page to fine tune the environment:
 - i. In the Engines tab, add in Engines Profiles a new engine (docker image) with 2 vCPUs and 4 GB RAM, while deleting the default engine.
 - ii. Check if the following variable already exists under Environmental Variables. If not, add it:

Unset

iii.

HADOOP_CONF_DIR=/etc/hadoop/conf/

iv.

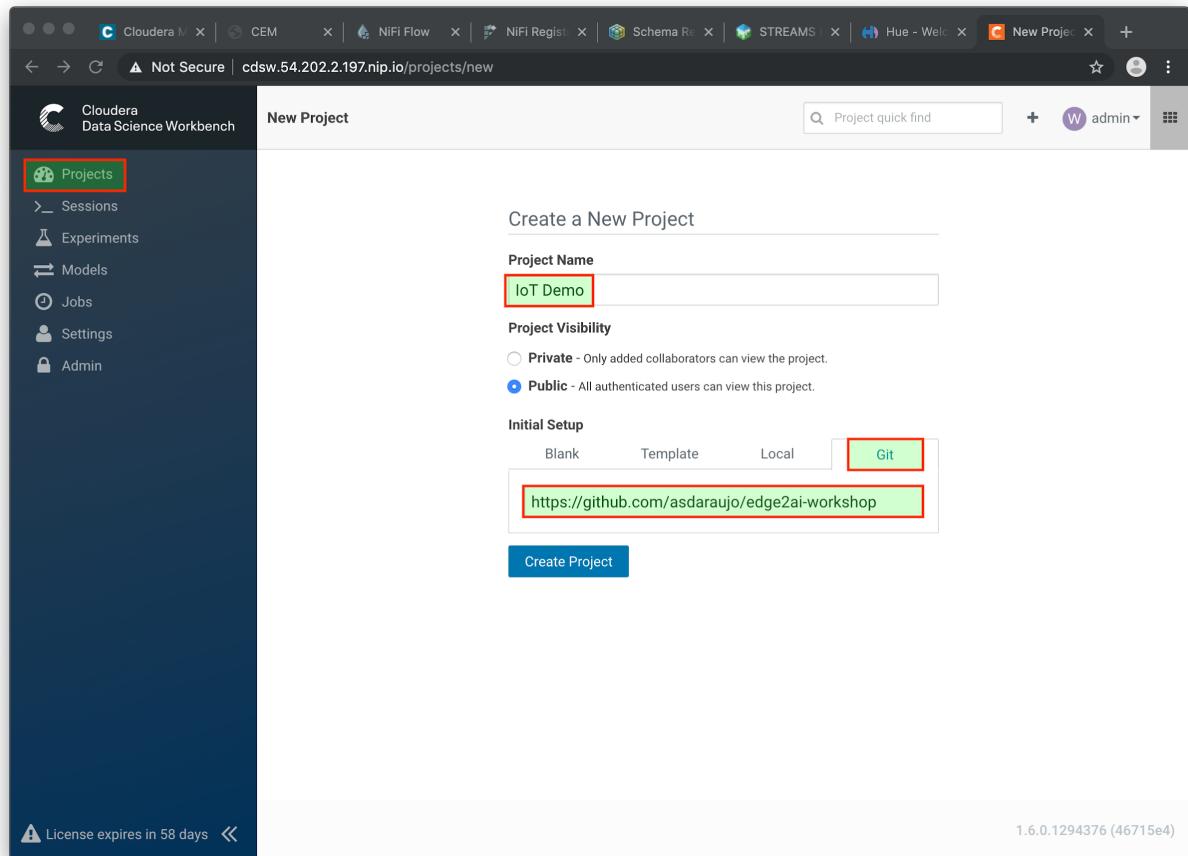
The screenshot shows the Cloudera Data Science Workbench interface with the URL `cdsw.54.202.2.197.nip.io/administration/engines`. The left sidebar has a dark theme with icons for Projects, Sessions, Experiments, Models, Jobs, Settings, and Admin (which is highlighted with a red box). The main content area is titled "Site Administration" and has tabs for Overview, Users, Activity, Models, Engines (which is selected and highlighted with a red box), Security, License, and Settings. The "Engines Profiles" section displays a table with one row:

Description	vCPU (burstable)	Memory (GiB)	Actions
2 vCPU / 4 GiB Memory	2	4	Edit Delete Add

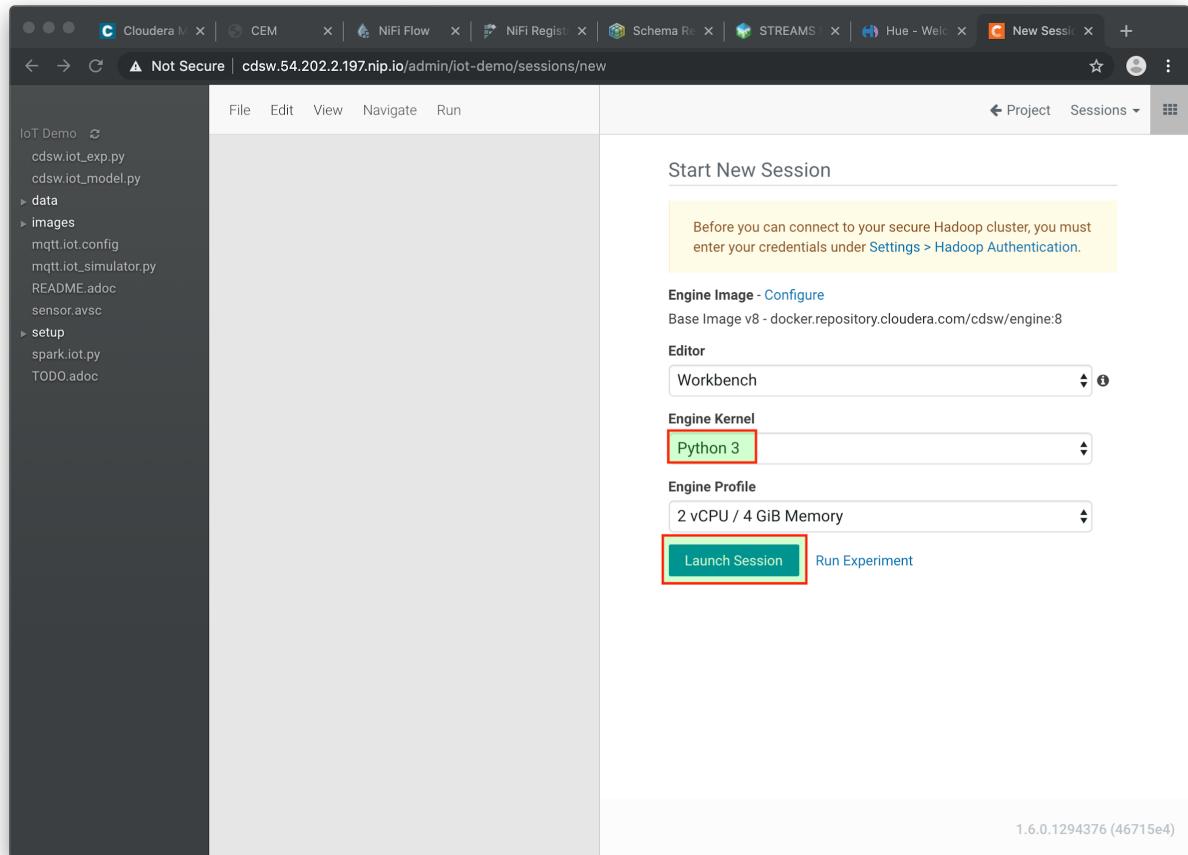
A note below the table states: "vCPU is expressed in fractional virtual cores and allows bursting. Memory is expressed in fractional GiB and is enforced by memory killer. GPU indicates the number of GPUs that need to be used by the engine. Configurations larger than the maximum allocatable CPU, memory and GPU per node will be unschedulable." The "Engine Images" section shows a single entry for "Base Image v8" with repository/tag "docker.repository.cloudera.com/cdsw/engine:8", editors "Jupyter Notebook", and actions "Edit" and "Deprecate". The "Environmental variables" section allows setting variables for all users' sessions and jobs. A table lists "HADOOP_CONF_DIR" with value "/etc/hadoop/conf/" and actions "Delete" and "Add". A footer note says "Whitelist Docker images for project owners to use in their jobs and sessions. These must be public images in registries that are accessible from the Cloudera Data Science Workbench hosts." A license warning at the bottom left says "⚠ License expires in 58 days" and a build ID at the bottom right says "1.6.0.1294376 (46715e4)".

STEP 2: Create the project

1. Return to the main page and click on New Project, using this GitHub project as the source: <https://github.com/cloudera-labs/edge2ai-workshop>



- Now that your project has been created, click on Open Workbench and start a Python3 session:



- Once the Engine is ready, run the following command to install some required libraries:

Unset

4.

```
!pip3 install --upgrade pip scikit-learn
```

5.

The project comes with a historical dataset. Copy this dataset into HDFS:

Unset

6.

```
!hdfs dfs -put -f data/historical_iot.txt /user/$HADOOP_USER_NAME
```

7.

The screenshot shows a terminal session in a web-based interface. The session is titled "Untitled Session" and is marked as "Running". It is a Python 3 session with 2 vCPU and 4 GiB Memory. The terminal window displays two commands:

```
> pip3 install --upgrade pip scikit-learn
Requirement already up-to-date: pip in ./local/lib/python3.6/site-packages (19.2.2)
Collecting scikit-learn
  Using cached https://files.pythonhosted.org/packages/a0/c5/d2238762d780dde84a26b8c761f563fe882b88c5a5fb83c056547c442a19/scikit_learn-0.21.3-cp36-cp36m-manylinux1_x86_64.whl
Requirement already satisfied, skipping upgrade: scipy>=0.17.0 in /usr/local/lib/python3.6/dist-packages (from scikit-learn) (1.3.0)
Requirement already satisfied, skipping upgrade: joblib>=0.11 in ./local/lib/python3.6/site-packages (from scikit-learn) (0.13.2)
Requirement already satisfied, skipping upgrade: numpy>=1.11.0 in /usr/local/lib/python3.6/dist-packages (from scikit-learn) (1.13.3)
Installing collected packages: scikit-learn
Successfully installed scikit-learn-0.21.3
> !hdfs dfs -put -f data/historical_iot.txt /user/$SHADOOP_USER_NAME
```

8. You're now ready to run the Experiment to train the model on your historical data.
9. You can stop the Engine at this point.

STEP 3: Examine cdsw.iot_exp.py

Open the file `cdsw.iot_exp.py`. This is a python program that builds a model to predict machine failure (the likelihood that this machine is going to fail). There is a dataset available on hdfs with customer data, including a failure indicator field.

The program is going to build a failure prediction model using the Random Forest algorithm. Random forests are ensembles of decision trees. Random forests are one of the most successful machine learning models for classification and regression. They combine many decision trees in order to reduce the risk of overfitting. Like decision trees, random forests handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions.

`spark.mllib` supports random forests for binary and multiclass classification and for regression, using both continuous and categorical features. `spark.mllib` implements random forests using the existing decision tree implementation. Please see the decision tree guide for more information on trees.

The Random Forest algorithm expects a couple of parameters:

- numTrees: Number of trees in the forest.
Increasing the number of trees will decrease the variance in predictions, improving the model's test-time accuracy. Training time increases roughly linearly in the number of trees.
- maxDepth: Maximum depth of each tree in the forest.
Increasing the depth makes the model more expressive and powerful. However, deep trees take longer to train and are also more prone to overfitting. In general, it is acceptable to train deeper trees when using random forests than when using a single decision tree. One tree is more likely to overfit than a random forest (because of the variance reduction from averaging multiple trees in the forest).

In the `cdsw.iot_exp.py` program, these parameters can be passed to the program at runtime, to these python variables:

```
param_numTrees = int(sys.argv[1])
param_maxDepth = int(sys.argv[2])
```

Also note the quality indicator for the Random Forest model, are written back to the Data Science Workbench repository:

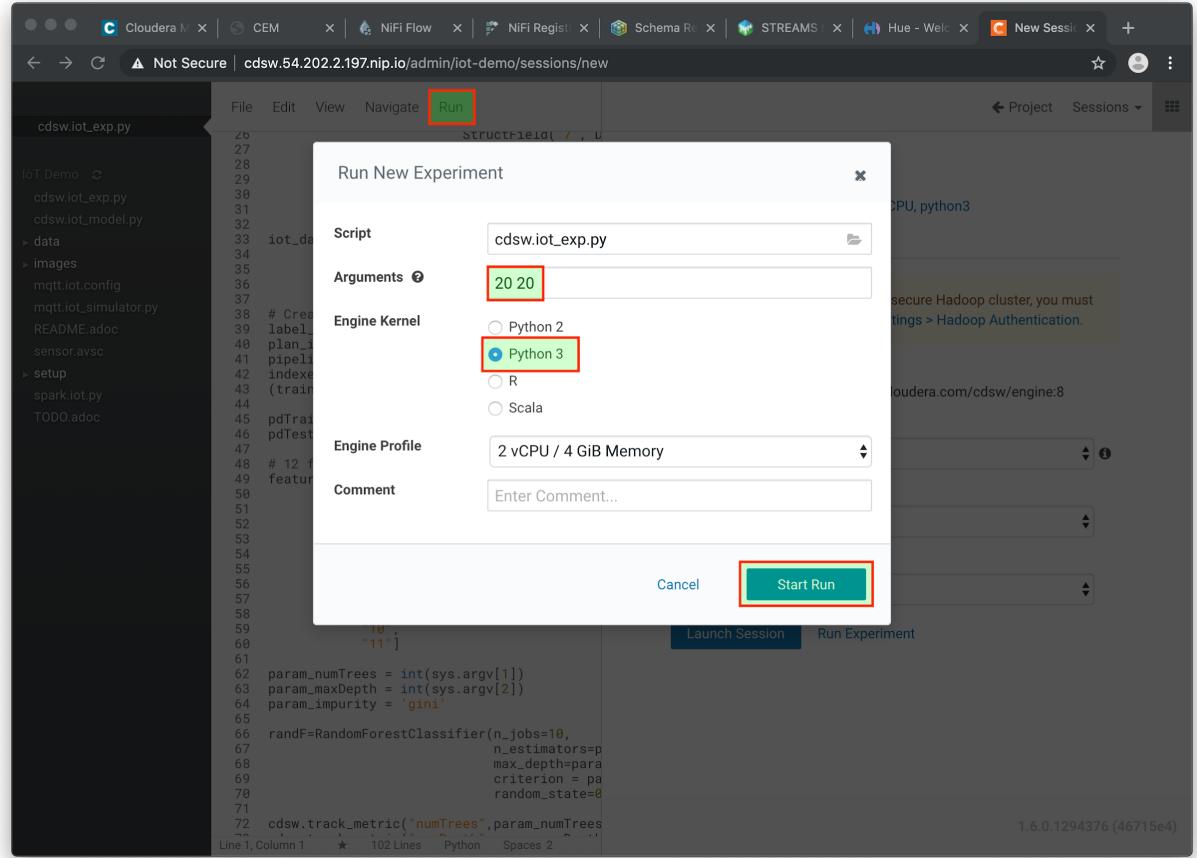
```
cdsw.track_metric("auroc", auroc)
cdsw.track_metric("ap", ap)
```

These indicators will show up later in the Experiments dashboard.

STEP 4: Run the experiment for the first time

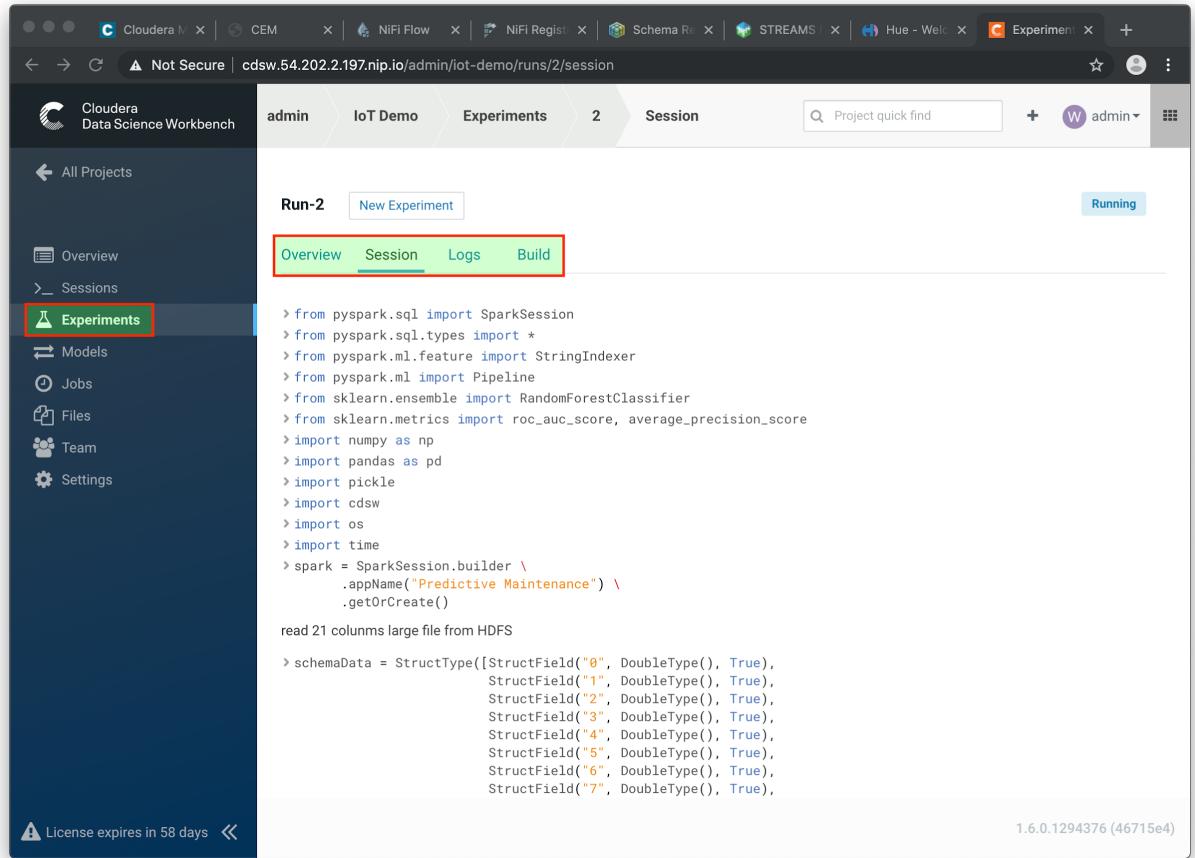
1. Now, run the experiment using the following parameters:
`numTrees = 20 numDepth = 20`
2. From the menu, select Run → Run Experiments.... Now, in the background, the Data Science Workbench environment will spin up a new docker container, where this

program will run.



3. Go back to the Projects page in CDSW, and hit the Experiments button.
4. If the Status indicates Running, you have to wait till the run is completed. In case the status is Build Failed or Failed , check the log information. This is accessible by clicking on the run number of your experiments. There you can find the session log, as well as

the build information.



The screenshot shows the CDSW interface with the following details:

- Header:** Not Secure | cdsw.54.202.2.197.nip.io/admin/iot-demo/runs/2/session
- User:** admin
- Project:** IoT Demo
- Experiment:** Experiments
- Run:** Run-2
- Status:** Running
- Tab Selection:** Overview (highlighted in green), Session, Logs, Build
- Code Preview:** A snippet of Python code for a machine learning pipeline, including imports for SparkSession, pandas, numpy, and various ML libraries, followed by schema definition and data reading logic.
- Footer:** License expires in 58 days, Version 1.6.0.1294376 (46715e4)

5. In case your status indicates Success, you should be able to see the auroc (Area Under the Curve) model quality indicator. It might be that this value is hidden by the CDSW user interface. In that case, click on the '3 metrics' links, and select the auroc field. It might be needed to de-select some other fields, since the interface can only show 3

metrics at the same time.

The screenshot shows the Cloudera Data Science Workbench interface. On the left, there's a sidebar with 'All Projects' and 'Experiments' selected. The main area is titled 'Experiments' and shows a table of runs. One row is highlighted with a red border, showing columns for Run ID (cdsw.iot_ex0020), Kernel (python3), Submitter (admin), Created At (8/21/19 10:00 PM), and three metrics: numTrees (20), maxDepth (20), and auroc (0.83...). A status bar at the bottom indicates 'Success' and '1 mins'. The bottom right corner shows the version 1.6.0.1294376 (46715e4).

Run	Script	Arguments	Kernel	Comment	Submitter	Created At	numTrees	maxDepth	auroc	Status	Duration	Actions
2	cdsw.iot_ex0020		python3		admin	8/21/19 10:00 PM	20	20	0.83...	Success	1 mins	

6. In this example, ~0.8383. Not bad, but maybe there are better hyper parameter values available.

STEP 5: Re-run the experiment several times

Go back to the Workbench and run the experiment 2 more times and try different values for NumTrees and NumDepth. Try the following values:

NumTrees NumDepth

15 25

1. 25 20
2. When all runs have completed successfully, check which parameters had the best quality (best predictive value). This is represented by the highest area under the curve:

auroc metric.

The screenshot shows the Cloudera Data Science Workbench interface. The left sidebar has a dark theme with icons for Overview, Sessions, Experiments (selected), Models, Jobs, Files, Team, and Settings. The main area is titled 'Experiments' and shows a table of runs. The table columns are: Run, Script, Arguments, Kernel, Comment, Submitter, Created At, numTrees, maxDepth, auroc, Status, Duration, and Actions. There are three rows:

Run	Script	Arguments	Kernel	Comment	Submitter	Created At	numTrees	maxDepth	auroc	Status	Duration	Actions
4	cdsw.iot_ex4520	python3	admin	8/21/19 10:14 PM	25	20	0.86...	Success	1 mins			
3	cdsw.iot_ex4525	python3	admin	8/21/19 10:13 PM	15	25	0.84...	Success	1 mins			
2	cdsw.iot_ex4020	python3	admin	8/21/19 10:00 PM	20	20	0.83...	Success	1 mins			

A red box highlights the '4' in the Run column of the first row. Another red box highlights the '0.86...' in the auroc column of the same row. A green box highlights the 'Success' status in the same row. At the bottom left, there is a warning message: '⚠ License expires in 58 days'. At the bottom right, it says '1.6.0.1294376 (46715e4)'.

STEP 6: Save the best model to your environment

1. Select the run number with the best predictive value (in the example above, experiment 4).
2. In the Overview screen of the experiment, you can see that the model, in Pickle format (.pkl), is captured in the file iot_model.pkl. Select this file and hit the Add to Project button. This will copy the model to your project directory.

Not Secure | cdsw.54.202.2.197.nip.io/admin/iot-demo/runs/4

Cloudera Data Science Workbench

admin IoT Demo Experiments 4 Overview Project quick find admin Success

All Projects Run-4 New Experiment

Overview Sessions Logs Build

Configuration

Script	cdsw.iot_exp.py
Arguments	25 20
Comment	
Build Snapshot	d7651bf2c94ca5696da58c50e8a24647be22497d
Created At	8/21/19 10:14 PM
Submitter	admin

Output

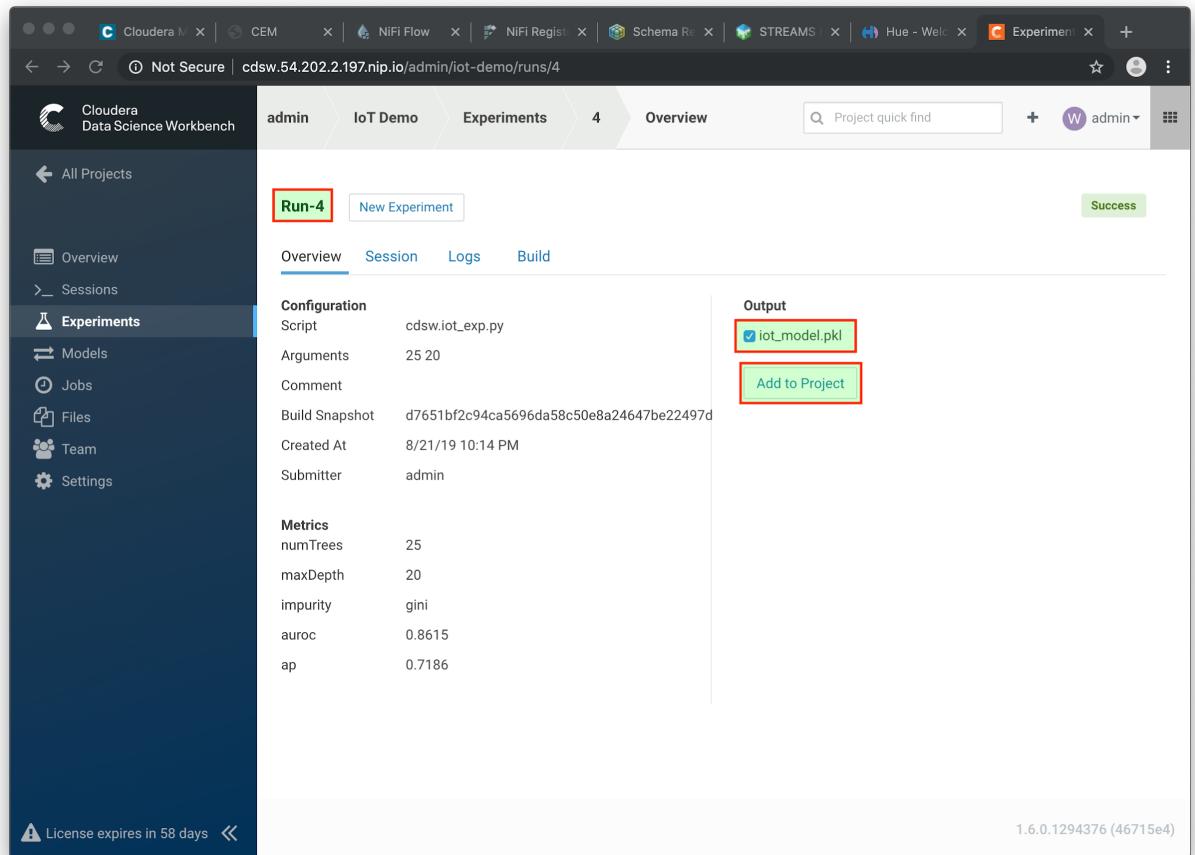
[iot_model.pkl](#)

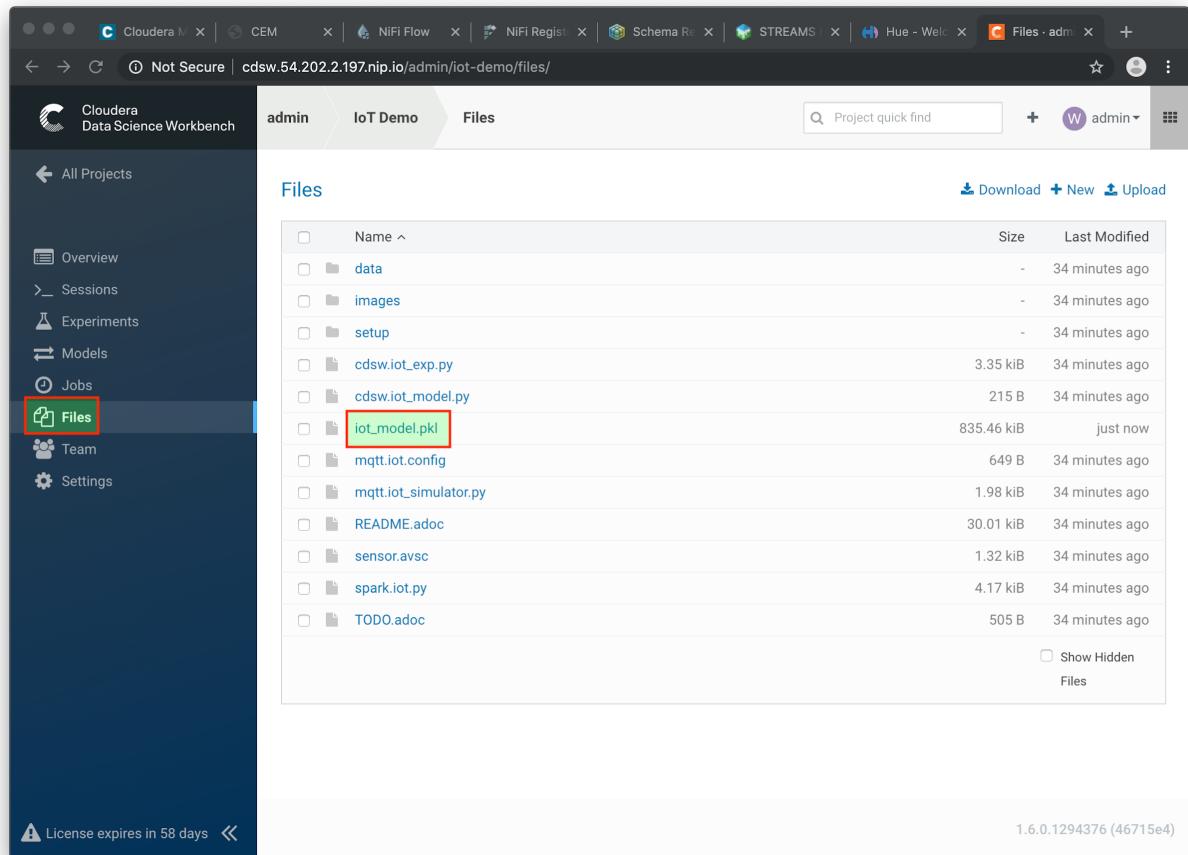
[Add to Project](#)

Metrics

numTrees	25
maxDepth	20
impurity	gini
auroc	0.8615
ap	0.7186

⚠ License expires in 58 days << 1.6.0.1294376 (46715e4)



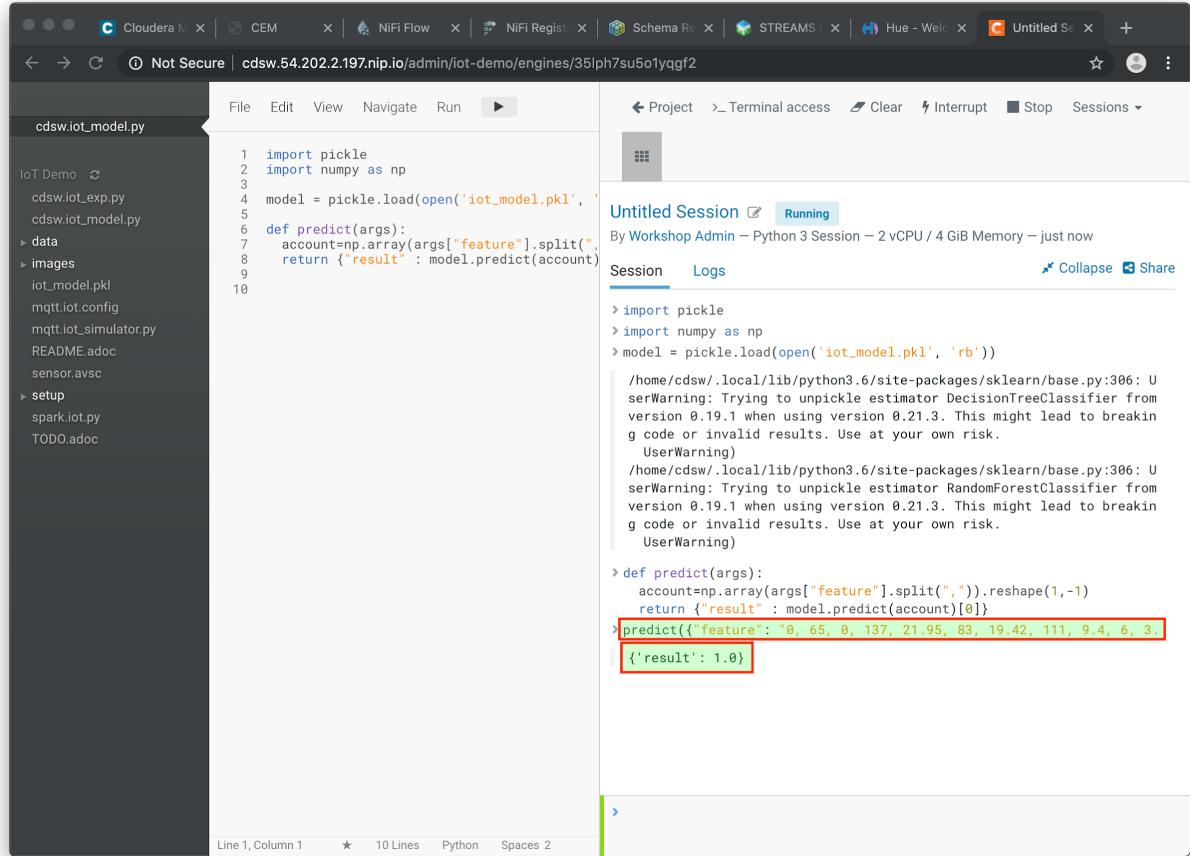


Lab 2 - CDSW: Deploy the model

STEP 1: Examine the program `cdsw.iot_model.py`

1. Open the project you created in the previous lab and examine the file in the Workbench. This PySpark program uses the pickle.load mechanism to deploy models. The model is loaded from the `iot_model.pkl` file, which was saved in the previous lab from the experiment with the best predictive model.
The program also contains the `predict` definition, which is the function that calls the model, passing the features as parameters, and will return a result variable.
2. Before deploying the model, try it out in the Workbench: launch a Python3 engine and run the code in file `cdsw.iot_model.py`. Then call the `predict()` method from the prompt:

```
predict({"feature": "0, 65, 0, 137, 21.95, 83, 19.42, 111, 9.4, 6, 3.43, 4"})
```



```
cdsw.iot_model.py
```

```
File Edit View Navigate Run Project >_ Terminal access Clear Interrupt Stop Sessions v
```

```
Untitled Session Running
```

```
By Workshop Admin - Python 3 Session - 2 vCPU / 4 GiB Memory - just now
```

```
Session Logs ✎ Collapse Share
```

```
import pickle
import numpy as np
model = pickle.load(open('iot_model.pkl', 'rb'))
def predict(args):
    account=np.array(args["feature"].split(","))
    return {"result": model.predict(account)[0]}
```

```
> import pickle
> import numpy as np
> model = pickle.load(open('iot_model.pkl', 'rb'))
/home/cdsw/.local/lib/python3.6/site-packages/scikit-learn/base.py:306: UserWarning: Trying to unpickle estimator DecisionTreeClassifier from version 0.19.1 when using version 0.21.3. This might lead to breaking code or invalid results. Use at your own risk.
UserWarning
/home/cdsw/.local/lib/python3.6/site-packages/scikit-learn/base.py:306: UserWarning: Trying to unpickle estimator RandomForestClassifier from version 0.19.1 when using version 0.21.3. This might lead to breaking code or invalid results. Use at your own risk.
UserWarning
> def predict(args):
    account=np.array(args["feature"].split(",")).reshape(1,-1)
    return {"result": model.predict(account)[0]}
predict({'feature': '0, 65, 0, 137, 21.95, 83, 19.42, 111, 9.4, 6, 3.43, 4'})
{'result': 1.0}
```

3. The functions returns successfully, so we know we can now deploy the model. You can now stop the engine.

STEP 2: Deploy the model

From the main page of your project, select the Models button. Select New Model and specify the following configuration:

Name: IoT Prediction Model

Description: IoT Prediction Model

File: cdsw.iot_model.py

Function: predict

Example Input: {"feature": "0, 65, 0, 137, 21.95, 83, 19.42, 111, 9.4, 6, 3.43, 4"}

Kernel: Python 3

Engine: 2 vCPU / 4 GB Memory

1. Replicas: 1

The screenshot shows the 'Create a Model' interface in the Cloudera Data Science Workbench. The left sidebar has a 'Models' tab selected, which is highlighted with a red box. The main area is titled 'Create a Model' and contains two sections: 'General' and 'Build'. In the 'General' section, there is a 'Name' field with the value 'IoT Prediction Model' and a 'Description' field with the value 'IoT Prediction Model'. In the 'Build' section, there is a 'File' field with the value 'cdsw.iot_model.py' and a 'Function' field with the value 'predict'. Both the 'File' and 'Function' fields are highlighted with red boxes. Below the 'Build' section is an 'Example Input' field containing a JSON object: { "feature": "0, 65, 0, 137, 21.95, 83, 19.42, 111, 9.4, 6, 3.43, 4" }.

2. After all parameters are set, click on the Deploy Model button. Wait till the model is deployed. This can take several minutes.

STEP 3: Test the deployed model

1. When your model status change to Deployed, click on the model name link to go to the Model's Overview page. From the that page, click on the Test button to check if the model is working.
2. The green circle with the success status indicates that our REST call to the model is working. The 1 in the response {"result": 1}, means that the machine from where these

temperature readings were collected is unlikely to experience a failure.

The screenshot shows the Cloudera Data Science Workbench interface. On the left sidebar, the 'Models' option is selected and highlighted with a red box. The main content area displays the 'IoT Prediction Model' page. At the top right, there are buttons for 'Deployed' (highlighted with a red box), 'Stop', 'Restart', and 'Deploy New Build'. Below this, tabs for 'Overview', 'Deployments', 'Builds', 'Monitoring', and 'Settings' are shown, with 'Overview' selected. The 'Description' field contains 'IoT Prediction Model'. Under 'Sample Code', there are buttons for 'Shell', 'Python' (highlighted with a red box), and 'R'. A code snippet is displayed:

```
curl -H "Content-Type: application/json" -X POST http://cdsw.54.202.2.197.nip.io/api/altus-ds-1/models/call-model -d '{"accesskey": "mlhrpsq9j168aropqh10i6ju10pn7izk", "request": {"feature": "0, 65, 0, 137, 21.95, 83, 19.42, 111, 9, 4, 6, 3.43, 4"}'}
```

. The 'Test Model' section contains an 'Input' field with the following JSON:

```
{ "feature": "0, 65, 0, 137, 21.95, 83, 19.42, 111, 9, 4, 6, 3.43, 4" }
```

. Below it are 'Test' and 'Reset' buttons, with 'Test' highlighted with a red box. The 'Result' section shows a 'Status' field with a green circle and the word 'success' and a 'Response' field containing the JSON:

```
{ "result": 1 }
```

. To the right, the 'Model Details' and 'Model Resources' sections provide deployment and resource information. The bottom right corner shows the version '1.6.0.1294376 (46715e4)'.

Now, let's change the input parameters and call the predict function again. Put the following values in the Input field:

```
{  
  "feature": "0, 95, 0, 88, 26.62, 75, 21.05, 115, 8.65, 5, 3.32, 3"  
  3.  
  4. With these input parameters, the model returns 0, which means that the machine is likely  
  to break.
```

Creating Dashboards with Cloudera Data Viz

In this workshop you will create a simple interactive real-time dashboard to visualize sensor data that is being stored in Kudu.

The data you will use is the sensor data collected and processed in previous workshops (see Preparation below).

Preparation

This workshop builds upon the content developed in the [Edge](#) and [Nifi](#) workshops.

To clean your environment and make it ready for the beginning of this lab, please SSH to your cluster host and run the following command:

Note	The command below will undo everything done in the cluster in previous workshops.
------	---

```
/tmp/resources/reset-to-lab.sh dataviz 1
```

SSH to your host as centos/supersecret1 and run the above command

Labs summary

- Lab 1 - Navigate to Cloudera Data Visualization
- Lab 2 - Creating a new connection
- Lab 3 - Exploring the data
- Lab 4 - Creating a dashboard
- Lab 5 - Adding a chart

Lab 1 - Navigate to Cloudera Data Visualization

This lab shows you how to navigate to Cloudera Data Visualization (DataViz) page.

If you are in a guided workshop you may already been given the link to the DataViz page. If that's the case, feel free to skip to the next lab.

1. Open CDP Data Visualization and log in

CDP Data Visualization can be accessed through the Cloudera Data Science Workbench (CDSW). Follow the navigation steps below if you don't know how to get there:

- i. In Cloudera Manager, click on Clusters > Cloudera Data Science Workbench.
- ii. On the CDSW page, click on the CDSW Web UI link.
- iii. Log on to CDSW.

- iv. On the CDSW page, click on Applications and then on the "Viz Server Application", which has been previously set up for the workshop.

The screenshot shows the Cloudera Data Science Workbench interface. The left sidebar has a dark blue background with white icons and text. The 'Applications' icon is highlighted with a red box. The main area is titled 'Applications' and contains a table with one row. The row is highlighted with a green box. A tooltip appears over the 'Actions' button for this row, reading 'open the application (http://viz.cdsweb.52.26.198.174.nip.io)'. The table columns are: Name, Subdomain, Created By, Started By, Last Updated, Status, and Actions. The single row in the table has the following values: Name = 'Viz Server Application', Subdomain = 'viz', Created By = 'admin', Started By = 'admin', Last Updated = '10/19/2021 3:25 PM', Status = 'Running', and Actions = 'Actions ▾'. At the bottom left of the main area, there is a warning message: '⚠ License expires in 56 days <<'. At the bottom right, the version is listed as '1.9.2.14556745 (f260849)'.

Name	Subdomain	Created By	Started By	Last Updated	Status	Actions
Viz Server Application	viz	admin	admin	10/19/2021 3:25 PM	Running	Actions ▾

- v. Log in to the Cloudera Data Visualization application. After logging in you should see the application home page:

The screenshot shows the Cloudera Data Visualization application interface. At the top, there's a navigation bar with links to Status, YARN, Kudu, CEM, NIFI F, NIFI R, Schema, STRE, Hue, Applic, Clickstream, Stream, and TIMES. Below the navigation bar, the URL is visible as `viz.cdsweb.52.26.198.174.nip.io/arc/apps/home`. The main header says "CLOUDERA Data Visualization". On the right, it shows "Workshop Admin". The main content area has a "Get Started" section with four cards: "16 DASHBOARDS", "1 APPS", "11 DATASETS", and "0 TOTAL VIEWS". Below this, there's a section titled "Sample Dashboards" with four preview cards: "Taxi rides application", "Iris species w/ images", "Consumer View", and "Inspector View". To the right, there's a sidebar with buttons for "NEW DASHBOARD" and "NEW APP", and a message about activity over the last 7 days: "16 Dashboards were created", "10 Apps were created", and "11 Datasets were created". A "Get Started" button is also present in the sidebar. At the bottom right, a note says "Your last login was on: Oct. 22, 2021 at 03:35 PM".

Lab 2 - Creating a new connection

Kudu is purely a storage engine and does not provide a SQL interface for querying. SQL access to Kudu is done through an Impala engine, which is what you will use in this workshop. You will set up a new connection to the Impala engine to use for your dashboard queries.

1. Select the Data tab and click on NEW CONNECTION.

The screenshot shows the Cloudera Data Visualization web application. At the top, there's a navigation bar with links like Status, YARN, Kudu, CEM, NIFI F, NIFI R, Schema, STRE, Hue, Applic, Click, Stream, and TIMES. Below the navigation bar, the title bar says 'Not Secure | viz.cdsrv.52.26.198.174.nip.io/arc/apps/data'. The main header 'CLOUDERA Data Visualization' is followed by 'HOME', 'VISUALS', and 'DATA' (which is highlighted with a red box). There's also a search bar and a user dropdown. On the left, there's a sidebar with 'All Connections' (selected), 'Impala' (selected), and 'samples'. The main content area has tabs for 'Datasets' (selected) and 'Connection Explorer'. Below these tabs is a table with columns: Title/Table, Created, Last Updated, Modified, By, and # Visuals. The table currently displays 'No data'.

At the top of the form, set the following properties:

Connection type: Impala

2. Connection name: Local Impala

In the Basic tab set the following:

Hostname: <CLUSTER_HOSTNAME> (something like: cdp.x.x.x.x.nip.io)

Port #: 21050

Username: [leave blank]

3. Password: [leave blank]

Create New Data Connection

Connection type	Impala	
Connection name	Local Impala	
Basic Advanced Parameters Cache		
Hostname or IP address	cdp.52.26.198.174.nip.io <small>(example: prod_db.yourcompany.com or 10.0.1.20)</small>	
Port #	21050	
Credentials		
Username		
Password		
TEST	CANCEL	CONNECT

In the Advanced tab set the following:

Connection mode: Binary

Socket type: Normal

4. Authentication mode: NoSasl

5. Click on TEST to test the connection.
You should see "Connection Verified", as shown below.

Create New Data Connection

Connection type ▾

Connection name

[Basic](#) [Advanced](#) [Parameters](#) [Cache](#)

Connection mode Binary HTTP

Socket type Normal SSL SSL with certificate

Authentication mode NoSasl Plain LDAP Kerberos

Query Timeout

Session Timeout

Socket Timeout

Queue Depth

Impersonation ⓘ Enabled

Trusted ⓘ Enabled
Impersonation

Connection Verified



[TEST](#)

[CANCEL](#)

[CONNECT](#)

6. Click on CONNECT.

Lab 3 - Exploring the data

Cloudera Data Visualization provides a Data Explorer tool that enables you to explore, transform and create views of the data to suit your needs. In this lab you will look at the data available in Kudu and prepare it for your dashboard.

1. Select the newly created Local Impala connection, which you can see on the left-hand pane.
2. Select the Connection Explorer tab, then the default database and finally the sensors table. A preview with sample data will be loaded.

The screenshot shows the Cloudera Data Visualization interface. On the left, under 'All Connections', the 'Local Impala' connection is selected and highlighted with a red box. In the center, the 'Connection Explorer' tab is active, showing the 'default' database with one table named 'sensors', also highlighted with a red box. At the bottom, a preview of the 'sensors' table data is displayed, with the 'Table Statistics' tab selected. The table has columns: sensor_id, sensor_ts, sensor_0, sensor_1, sensor_2, sensor_3, sensor_4, sensor_5, sensor_6, and sensor_7. The data preview shows five rows of sample data.

sensor_id	sensor_ts	sensor_0	sensor_1	sensor_2	sensor_3	sensor_4	sensor_5	sensor_6	sensor_7
15	1635118021053513	4	13	4	28	38	57	98	103
17	1635117982704108	5	17	0	42	60	65	69	111
17	1635118009762765	1	19	1	23	37	73	34	123
48	1635117931702566	3	14	3	37	18	81	29	95
68	1635117936195550	3	14	0	24	57	74	62	126

You can see in the data sample that the `sensor_ts` column contains the timestamp in microseconds. For your dashboard you need to convert these values into seconds instead. In the next steps you will create a new dataset and make the necessary data adjustments.

3. Click on the New dataset option besides the sensor table. Name the dataset "sensor data"

Datasets

Connection Explorer

Table Name # Datasets

- [_impala_builtins] 0
- [default] 0

New dataset Refresh Compute statistics

A new dataset will be created and displayed under the Datasets tab:

Title/Table	Created	Last Updated	Modified By	# Visuals
sensor data default.sensors	Oct 25, 2021	a few seconds ago	admin	0

- Click on the dataset to open it and select the Fields tab. You will notice that DataViz didn't automatically detect any dimension for the dataset. Since the sensor_ts column is of a numeric type, and not a date/time, which is indicated by the # icon besides the field name, it was classified as a measure rather than a dimension. You will fix in the next steps.

CLOUDERA Data Visualization

Dataset: sensor data

Fields EDIT FIELDS Show Comments NEW DASHBOARD

Dimensions

Measures

sensor

- # sensor_id
- # sensor_ts**
- sensor_0
- sensor_1
- sensor_2
- sensor_3
- sensor_4
- sensor_5
- sensor_6
- sensor_7
- sensor_8
- sensor_9
- sensor_10
- sensor_11
- # is_healthy

5. You need to convert the numeric fields from microseconds to seconds and convert it to a TIMESTAMP data type. In order to do this, click on the EDIT FIELDS button.

The screenshot shows the Cloudera Data Visualization interface. At the top, there is a navigation bar with links for HOME, VISUALS, and DATA. Below the navigation bar, the page title is "Dataset Detail" and the dataset name is "sensor data". On the left side, there is a sidebar with sections for Dataset Detail, Related Dashboards, Fields, and Data Model. The "Fields" section is currently selected. At the top of the main content area, there is a "Fields" header and an "EDIT FIELDS" button, which is highlighted with a red box. To the right of the "Fields" header, there is a link "Show Comments".

6. In the Measures list, find the sensor_ts measure, open its drop-down menu and click on Clone. A new measure Copy of sensor_ts will appear.

The screenshot shows the "Measures" list. The "sensors" category is expanded, showing eight measures: sensor_id, sensor_ts, sensor_0, sensor_1, sensor_2, sensor_3, sensor_4, and sensor_5. The "sensor_ts" measure is highlighted with a red box. To the right of the list, a modal window titled "Edit Field" is open, containing a "Clone" button, which is also highlighted with a red box. The number "15" is displayed in the top right corner of the main measures list area.

Measure	Type	Format	Value
sensor_id	Mes	#	
sensor_ts	Mes	#	
sensor_0	Mes	1.2	
sensor_1	Mes	1.2	
sensor_2	Mes	1.2	
sensor_3	Mes	1.2	
sensor_4	Mes	1.2	
sensor_5	Mes	1.2	

7. Open the drop-down menu for this new measure, and select Edit field.

The screenshot shows a 'Measures' interface with a list of sensors. The list includes:

- sensor_id
- sensor_ts
- Copy of sensor_ts** (highlighted with a red box)
- sensor_0
- sensor_1
- sensor_2
- sensor_3
- sensor_4

A context menu is open over the 'Copy of sensor_ts' entry, with the following options:

- Edit Field (highlighted with a green box)
- Clone
- Remove

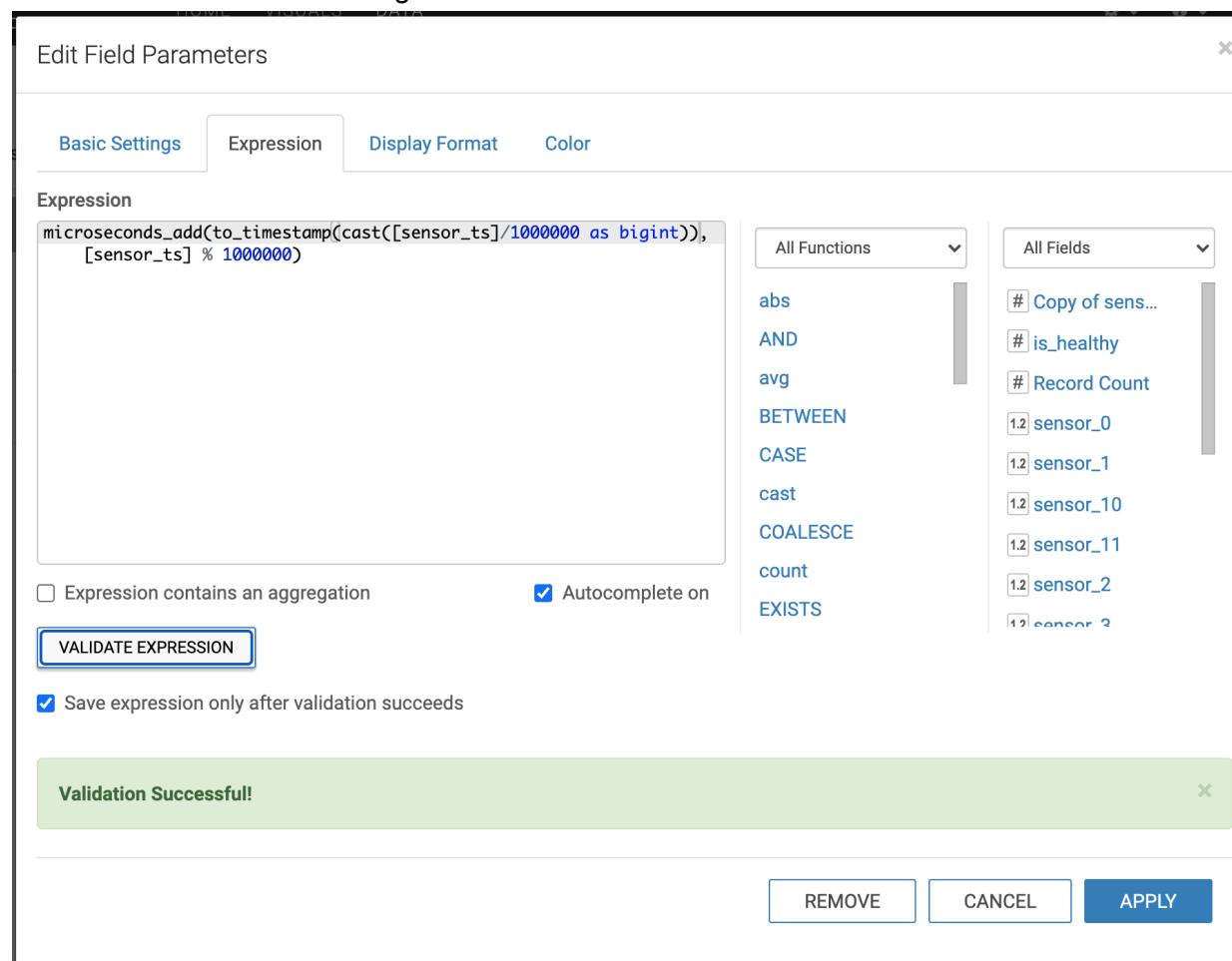
8. In the Edit Field Parameters window, change the following:

In the Basic Settings tab:

Display Name: sensor_timestamp

- Category: Dimension
- In the Expression tab, enter the following expression:
microseconds_add(to_timestamp(cast([sensor_ts]/1000000 as bigint)),
[sensor_ts] % 1000000)
- Validate the expression by clicking on VALIDATE EXPRESSION.

- iv. Click APPLY to save the changes



9. You will notice that the category (Dim), data type (calendar icon) and field name were updated. The field still shows up in the Measure category, though.

The screenshot shows the 'Measures' section of a data modeling interface. The 'sensors' category is expanded, displaying the following measures:

- sensor_id (Mes, #, sensor_id)
- sensor_ts (Mes, #, sensor_ts)
- sensor_timestamp** (Dim, Calendar, sensor_timestamp) - This field is highlighted with a red border.
- sensor_0 (Mes, 1.2, sensor_0)
- sensor_1 (Mes, 1.2, sensor_1)

This is just refresh issue. Click on the REFRESH button at the top and you should see the sensor_timestamp field "jump" to the Dimensions category.

Dataset: sensor data

Fields REFRESH SAVE Show Comments NEW DASHBOARD

To add a new calculated field, use the down arrow to the right of a field to clone it, and then edit the expression of the cloned field.

Dimensions
▼ sensors
= Dim # sensor_timestamp

Measures
▼ sensors
Mes # sensor_id
Mes # sensor_ts
Mes 1.2 sensor_0
Mes 1.2 sensor_1
Mes 1.2 sensor_2

10. The sensor_id field is also a dimension and needs to be moved to the correct category. To do this, find the sensor_id field under the Measures category and click on the

Mes Dim . Click on the REFRESH button again and you should see the following structure for your dataset:

Dimensions
▼ sensors
Dim # sensor_id
= Dim # sensor_timestamp

Measures
▼ sensors
Mes # sensor_ts
Mes 1.2 sensor_0
Mes 1.2 sensor_1
Mes 1.2 sensor_2
Mes 1.2 sensor_3
Mes 1.2 sensor_4
Mes 1.2 sensor_5
Mes 1.2 sensor_6
Mes 1.2 sensor_7
Mes 1.2 sensor_8
Mes 1.2 sensor_9
Mes 1.2 sensor_10
Mes 1.2 sensor_11
Mes # is_healthy

11. Save your changes by clicking the green Save button.

You have just created a dataset to feed your dashboard and performed the necessary adjustments for your data source. In the next lab you will create the dashboard from it.

Lab 4 - Creating a dashboard

You have everything ready now to start building your dashboard. Let's jump straight into it:

1. On your dataset page, click on the NEW DASHBOARD button.

The screenshot shows the Cloudera Data Visualization interface. In the top navigation bar, 'HOME', 'VISUALS', and 'DATA' are visible. The 'DATA' tab is selected. On the left, there's a sidebar with 'Dataset Detail', 'Related Dashboards', 'Fields' (which is currently selected), 'Data Model', and 'Time Modeling'. The main area shows 'Dataset: sensor data' and two sections: 'Dimensions' and 'Measures'. The 'Dimensions' section under 'sensors' contains '# sensor_id' and '=sensor_timestamp'. The 'Measures' section also under 'sensors' contains '# sensor_ts', '1.2 sensor_0', and '1.2 sensor_1'. A green box highlights the 'NEW DASHBOARD' button in the top right corner.

2. Since we initiated the dashboard creation from the dataset page, will you notice that the dashboard is already created by default with a "table visual" displaying all fields of the dataset.

The screenshot shows the Dashboard Designer interface. The top navigation bar includes 'LAYOUT', 'SAVE', and 'PRIVATE'. The main area has a title 'enter title...' and subtitle 'enter subtitle...'. Below is a table visual showing data for sensor_id, sensor_ts, sensor_timestamp, sensor_0, sensor_1, sensor_2, and sensor_3. The table has 6 rows of data. To the right is the 'Dashboard Designer' panel with tabs for 'VISUALS', 'DATA', 'DASH.', 'Visuals', 'Filters', 'Settings', 'Style', 'BUILD', and 'Settings', 'Style'. Under 'VISUALS', a 'Table' icon is selected. The 'DATA' tab shows the 'sensor data' dataset with fields like '# Record Count', '# sensor_ts', '1.2 sensor_0', etc. The 'BUILD' tab is currently selected, showing sections for 'Dimensions', 'Measures', 'Tooltips', and 'Filters', each with a 'drag or click fields to add here' placeholder. A 'REFRESH VISUAL' button is at the bottom of the build panel.

3. Click on the table visual to ensure it is selected (you see a blue border around the visual when it is selected). With the table visual selected, click on the Build tab on the right.

4. Click on the Measures input box to select it. Then click on the fields sensor_0 and sensor_1 from the Measures list. These fields will be added to the Measures input box.

The screenshot shows the Data Studio interface with the following components:

- VISUALS Panel:** Contains a grid of visualization icons including Table, Bar Charts, Line Charts, Word Cloud, and various data exploration tools.
- Dimensions Panel:** A section labeled "Dimensions" with a placeholder "drag fields to add here".
- Measures Panel:** A section labeled "Measures" containing two items: "# sum(sensor_0)" and "# sum(sensor_1)". This section is highlighted with a red border.
- Tooltips Panel:** A section labeled "Toolips" with a placeholder "drag fields to add here".
- Filters Panel:** A section labeled "Filters" with a placeholder "drag fields to add here".
- Limit Input:** An input field set to "100".
- REFRESH VISUAL Button:** A blue button with a circular arrow icon and the text "REFRESH VISUAL".
- DATA Panel:** Shows a connection named "sensor data" with edit and refresh icons. It includes a "Sample Mode: OFF" switch, a search bar, and a "Dimensions" section listing "sensors" and their fields: "# sensor_id" and "# sensor_timestamp".
- Measures Panel:** Shows a list of 15 measures under the "sensors" category, each represented by a small icon and a label: Record Count, sensor_ts, sensor_0, sensor_1, sensor_2, sensor_3, sensor_4, sensor_5, sensor_6, sensor_7, sensor_8, sensor_9, and sensor_10.
- Build Panel:** A vertical sidebar on the right with tabs for DASH., Visuals, Filters, Settings, Style, and another set of Visuals, Settings, Style tabs. The "Build" tab is currently selected.

5. The measures are added, by default, with the sum() aggregation. Change it to avg() by selecting each one of the newly added measures and selecting Aggregates > Average.

Ensure this is done for both measures.

VISUALS

Table

1234 LABEL WORD CLOUD ACTION

Dimensions

drag fields to add here

Measures

avg(sensor_0)

sum(sensor_1)

Tooltips

drag fields to add here

Filters

drag fields to add here

Limit: 100

REFRESH VISUAL

FIELD PROPERTIES

Aggregates

Sum

Count

Approx Distinct Count

Exact Distinct Count

Minimum

Maximum

Average

Approx Median

Standard Deviation

Population Standard Deviation

Variance

Population Variance

String Concat

Date/Time Functions

Text Functions

Analytic Functions

Change Type

Order and Top K

[] Enter/Edit Expression

Background Color

Aggregate Display

DASH.

Visuals

Filters

Settings

Style

VISUAL

Build

Settings

Style

6. Click on the Dimensions input box to select it. Then click on the fields sensor_timestamp and sensor_id from the Dimensions list. These fields will be added to the Dimensions input box.
7. Highlight sensor_timestamp field in the Measures input box and select Order and Top K > Descending. This will show the values in the table visual in descending order with the

newest sensor readings on top.

VISUALS

Table

1234 LABEL WORD CLOUD ACTION

sensor_timestamp sensor_id

avg(sensor_0) avg(sensor_1)

drag fields to add here

drag fields to add here

Limit: 100

REFRESH VISUAL

FIELD PROPERTIES

- Date/Time Functions
- Text Functions
- Analytic Functions
- Change Type
- Order and Top K
- Descending
- Ascending

Top K: eg. 100

Bottom K: eg. 100

Top K/Bottom K applies to granular dimensions

[] Enter/Edit Expression

Enable Expansion

Background Color

Display Format

Alias

Description

Duplicate

Save Expression

Remove

DASH.

Visuals

Filters

Settings

Style

VISUAL

Build

Settings

Style

- Click on Refresh visual to update the visual with the latest changes.

9. Finally, select the Settings tab on the right of the screen and change the value for Auto-refresh period (sec) to 5.

The screenshot shows the 'Dashboard Designer' interface. At the top, there's a teal header bar with the title 'Dashboard Designer'. Below it, the main area has a light gray background. On the left, there's a sidebar with several tabs: 'DASH.' (selected), 'Visuals', 'Filters', 'Settings' (selected), 'Style', and 'VISUAL'. The main content area is titled 'SETTINGS' and contains a section for 'General' settings. It includes the following options:

- Display context menus within visuals
- Enable data detail in visuals
- A setting for 'Detailed Data Col...' with a numeric input field set to '1000' with a minus sign on the left and a plus sign on the right.
- A checkbox for 'Refresh visuals manually' with an information icon next to it.
- A setting for 'Auto-refresh perio...' with a numeric input field set to '5' with a minus sign on the left and a plus sign on the right.
- Enable animations

Below these general settings, there are three more sections: 'Filter Settings' (indicated by a right-pointing arrow), 'Downloads' (indicated by a right-pointing arrow), and 'Sensor' (which is currently selected).

10. Click on the Save button at the top of the dashboard to save the changes and click View to enter view/publish mode. This is what your dashboard consumers will see: the sensor

reading coming in through the streaming pipeline, displayed in a real-time dashboard, updating automatically.

The screenshot shows a web browser window for 'viz.cds.w.18.157.157.193.nip.io/arc/apps/app/100'. The title bar says 'Not Secure'. The header includes the 'CLOUDERA Data Visualization' logo, 'HOME', 'VISUALS' (which is underlined), and 'DATA'. Below the header are buttons for 'EDIT', '...', and 'PRIVATE'. The main content area displays a table with the following data:

sensor_timestamp	sensor_id	avg(sensor_0)	avg(sensor_1)
2021-03-10 00:58:42.049217	37	3	18
2021-03-10 00:58:40.909602	39	5	16
2021-03-10 00:58:39.782189	25	3	10
2021-03-10 00:58:38.635853	71	2	18
2021-03-10 00:58:37.492364	33	2	20
2021-03-10 00:58:36.369134	15	2	11
2021-03-10 00:58:35.246077	46	5	9
2021-03-10 00:58:34.125410	11	3	15

At the bottom right of the table, there are navigation arrows labeled '< 1 2 3 4 5 >'.

Lab 5 - Adding a chart

Dashboards are usually synonym with graphs and charts. Cloudera Data Visualization comes with a myriad of charts types to help visualize your data. In this lab you'll add a simple bar chart to your dashboard to make it more interesting.

1. On the view mode dashboard above, click on the EDIT button to go back into editing mode.

2. Click on the Visuals tab on the right. Ensure the Local Impala connection and the sensor data dataset are selected and click on the NEW VISUAL button.

The screenshot shows the 'ADD VISUALS' interface. On the right side, there is a vertical navigation bar with tabs: 'DASH.', 'Visuals' (which is highlighted with a red border), 'Filters', 'Settings', and 'Style'. The main area has a header 'ADD VISUALS' with a back arrow and close button. Below it is a placeholder text 'Click on a visual to add a clone to the dashboard'. A section titled 'Recently Created Visuals' shows a card for 'Untitled'. At the bottom, a red box highlights the 'sensor data' section, which contains dropdowns for 'Local Impala' (selected) and 'sensor data' (selected), and a large green 'NEW VISUAL' button.

ADD VISUALS

Click on a visual to add a clone to the dashboard

Recently Created Visuals

Untitled

sensor data

Local Impala

sensor data

NEW VISUAL

DASH.

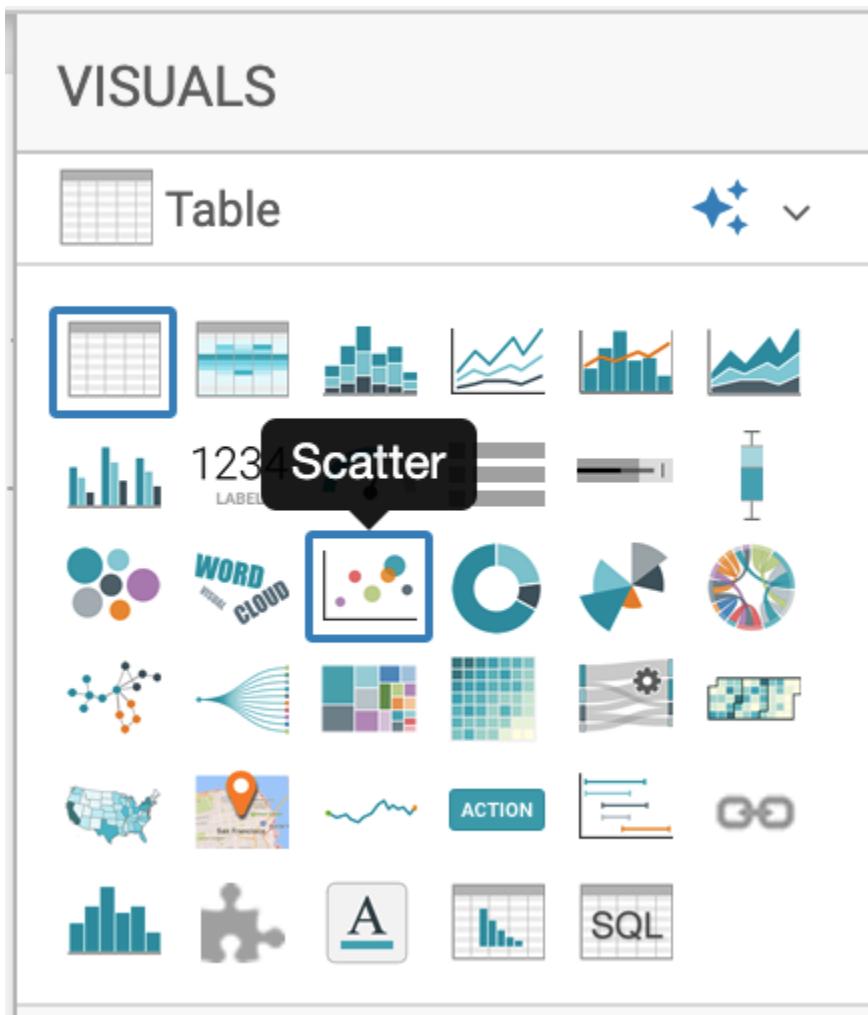
Visuals

Filters

Settings

Style

3. On the Visuals tab, select the Scatter visual type:



Based on what you learned in the previous lab, enter the following properties:

X Axis: sensor_id

Y Axis: avg(sensor_0)

Colors: sensor_id

Size: avg(sensor_0)

4. Filters: sensor_timestamp

5. Click on the sensor_timestamp filter to select it and then click on  Enter/Edit Expression.

VISUALS

Scatter

sensor_id
* Y Axis
avg(sensor_0)

Colors
sensor_id

Size
avg(sensor_0)

Transition
drag fields to add here

Tooltips
drag fields to add here

Filters
sensor_timestamp

REFRESH VISUAL

FILTER PROPERTIES

Select values
[] Enter/Edit Expression

Clear values
Remove

6. Enter the following expression in the Enter/Edit Expression window to limit the data shown in the chart to the last minute of data received. This will create a chart over a rolling window of 1 minute.

[sensor_timestamp] > seconds_sub(now(), 60)

7. Validate the expression and click Save.

8. Click on VISUAL > Style on the right-hand tab, and select a colorful palette in the Colors section.

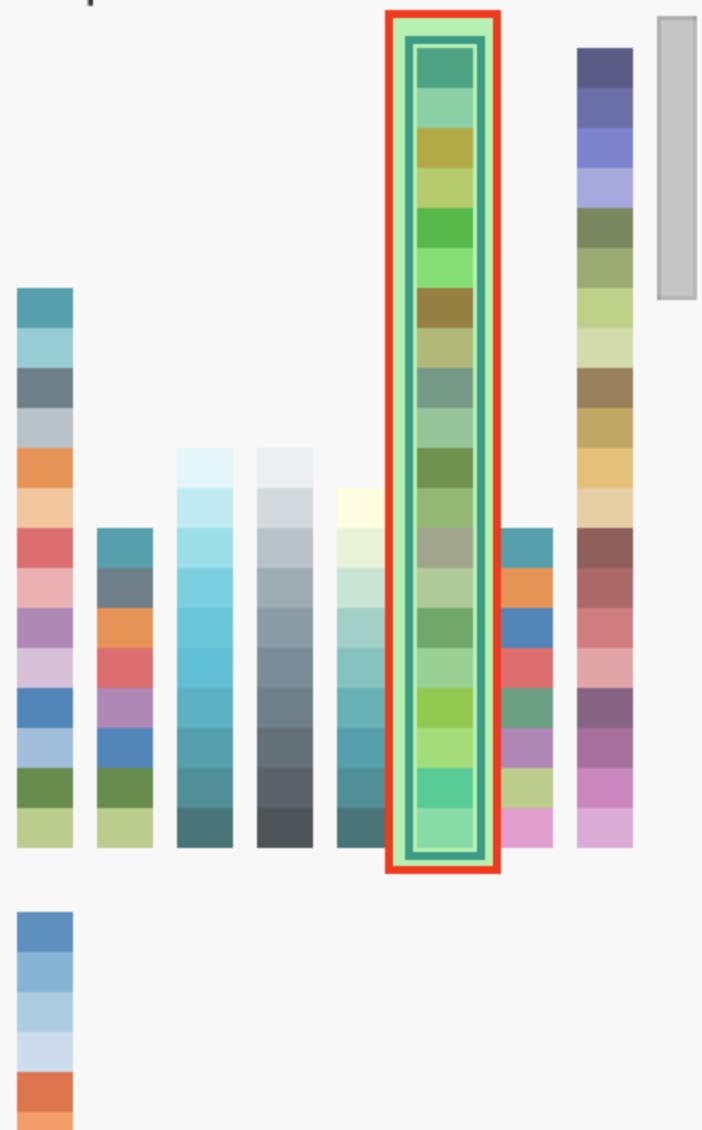
> VISUAL STYLE



Apply future changes to
all visuals

▼ Colors

Reverse order of colors in
palettes



DASH.

+
Visuals

+
Filters

Settings

Style

VISUAL

Build

Settings

Style

9. Click on VISUAL > Settings on the right-hand tab, and set the Y Axis Scale to log10 in the Axes section.

> VISUAL SETTINGS  

Apply future changes to all visuals 

 Parameters

 Data

 Trellis

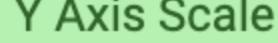
 Marks

 Axes

Auto-detect time columns to plot a continuous axis

X Axis Scale

Linear \log_{10}

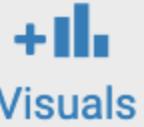
 Y Axis Scale

Linear \log_{10}

Make X axis continuous 

Make Y axis continuous 

DASH.



Visuals



Filters



Settings



Style

VISUAL



Build



Settings



Style

10. Expand the Marks section and set the Legend style to None.

> VISUAL SETTINGS  

Apply future changes to all visuals 

 > Data

 > Trellis

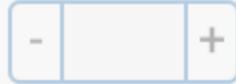
 > Marks

- Show detail data button in context menu
- Display tooltips
- Label marks by color

Mark size range  10-50
1 100

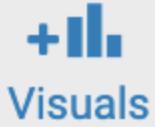
Legend Style:

 None  Right  Bottom  Above

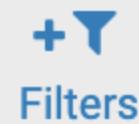
Legend item width...  - +

Reverse legend order

DASH.



Visuals



Filters



Settings



Style

VISUAL



Build



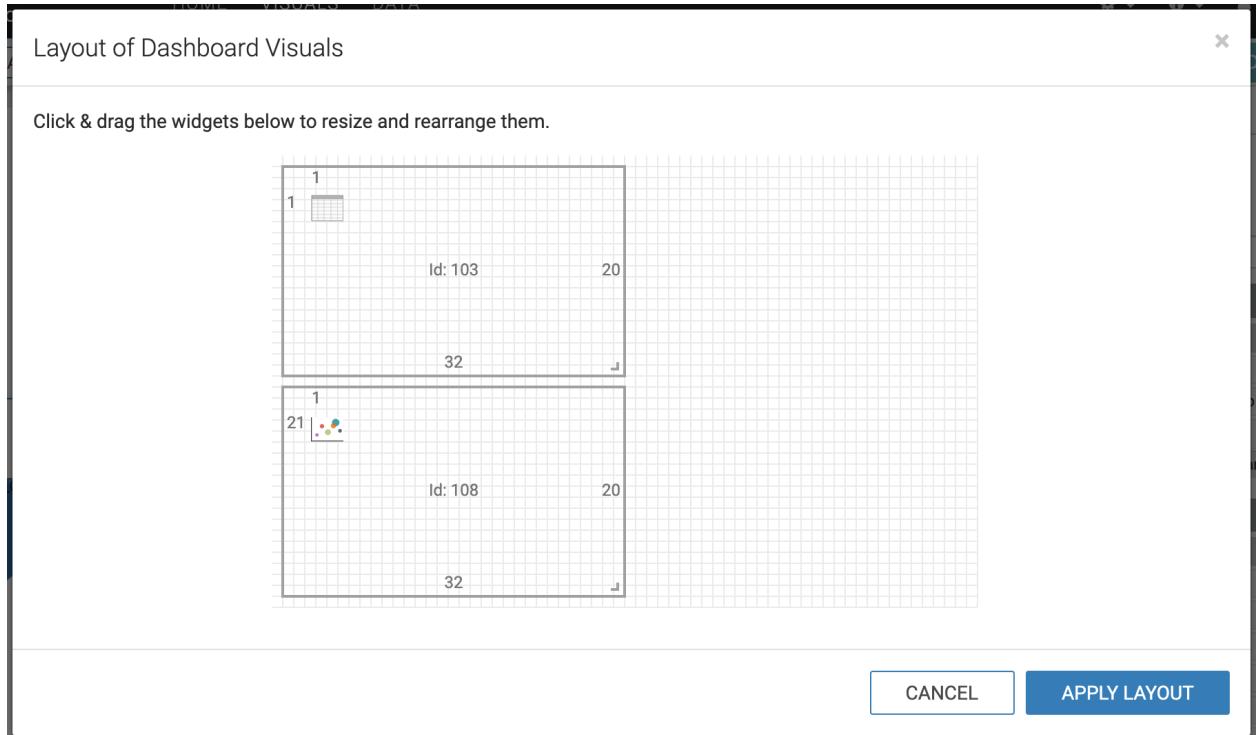
Settings



Style

LAYOUT

11. Click on the **LAYOUT** button, at the top of the Dashboard Designer to arrange the visuals in your dashboard. Drag the two visuals in the diagram to position them as you would like. Once you are done, click on **APPLY LAYOUT**.



12. Click on the Save buttons to save the changes to your dashboard and then click on View to switch to the view mode and check your real-time dashboard in action:

CLOUDERA Data Visualization HOME VISUALS DATA

EDIT
...
PRIVATE ▾

sensor_id	sensor_ts	sensor_timestamp	sensor_0	sensor_1	sensor_2	sensor_3
1	1.63P	2021-10-22 04:16:18.075267	4	12	1	1
1	1.63P	2021-10-22 04:47:43.052927	3	9	5	1
1	1.63P	2021-10-22 05:23:13.810994	3	1	1	1
1	1.63P	2021-10-22 05:48:27.356695	2	10	1	1
1	1.63P	2021-10-22 07:53:11.367616	5	1	5	1
1	1.63P	2021-10-22 07:57:59.608734	5	1	0	1
1	1.63P	2021-10-22 08:17:49.557199	1	10	0	1
1	1.63P	2021-10-22 08:21:12.620700	4	11	5	1

< 1 2 3 4 5 >

A bubble chart visualizing sensor data. The x-axis is labeled "sensor_id" and ranges from 0 to 100 on a logarithmic scale. The y-axis represents "avg(sensor_0)" on a logarithmic scale with major ticks at 1, 10, 100, and 1,000. The size of each bubble corresponds to the value of avg(sensor_0). Data points are labeled with their respective sensor_ids. A large blue bubble at sensor_id 6 has a value of 6. A large orange bubble at sensor_id 19 has a value of 19. A large red bubble at sensor_id 33 has a value of 33. A large pink bubble at sensor_id 68 has a value of 68.

sensor_id	avg(sensor_0)
1	6
9	19
16	33
22	68
27	1
32	10
39	100
43	1,000
45	10
50	1
51	10
54	100
60	1
65	10
66	100
71	1
75	10
78	100
79	1,000
80	10
87	100
88	1,000
93	10
94	1
97	100
99	1,000

Deep Learning Image Analysis AMP

Go to **AMPs**, choose **Deep learning for Image Analysis**, select Configure Project and click Launch Project.

Configure Project: Deep Learning for Image Analysis - admin

AMP Name: Image Analysis (v2)
Prototype to demonstrate semantic image search

Environment Variables
This prototype does not define any environment variables.

Default Engine:

Runtime

Editor	Kernel	Edition	Version
Workbench	Python 3.9	Standard	2022.11

Enable Spark

Runtime Image
- docker-private.infra.cloudera.com/cloudera/cdsw/ml-runtime-workbench-python3.9-standard:2022.11.1-b2

Setup Steps
 Execute AMP setup steps

Once all the AMP Setup steps are completed, click open link shown in step 4.

AMP Setup Steps

AMP Name: Image Analysis (v2)
Prototype to demonstrate semantic image search

Completed all steps

✓ Step 3 Running FAISS index creation job. [View details](#) completed 1/25/2023 11:15 AM

Running FAISS index creation job.

```
index (lib.FaissIndex): index to be updated
image_dir (str): directory of images to be added to index
index_save_dir (str): directory to save index on disc.
"""
features, ids = extractor.extract_from_dir(image_dir, efficientnet_model)
print(features.shape, len(ids))
index.add(features, id_strings=ids)
index.save(index_save_dir)
index = create_index(fashion_images_dir, os.getcwd() + "/faiss")
update_index(index, iconic_images_dir, os.getcwd() + "/faiss")
```

✓ Step 4 Create an application to serve the image analysis UI [View details](#) [Open](#) completed 1/25/2023 11:15 AM

```
return jsonify(result)
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Application parameters')
    parser.add_argument('-p', '--port', dest='port', type=int,
                        help='port to run model',
                        default=os.environ.get("CDSW_READONLY_PORT"))

    args, unknown = parser.parse_known_args()
    port = args.port
    app.run(port=port)
```

imageanalysis-6fcq4x.ml-744cb583-5e4.apps.lab03.pvc-ds-bc.athens.cloudera.com/#/

YouTube Maps Cloudera

ConvNet Playground Semantic Search Embeddings Live Search

Explore Precomputed Results of Semantic Search Queries

Dataset : iconic200
MODEL : XCEPTION
LAYER : BLOCK34_SEPC ..
METRIC : COSINE

When you select an image (by clicking it), a neural network model [XCEPTION] looks at the content of all images in our dataset and identifies the **top 10** most similar images to the selected image. Click advanced options to select a different dataset, model (or intermediate model) and distance metric used for semantic search.

Advanced Options

Top 10 results based on your search configuration| MODEL: XCEPTION | LAYER: block14_sepconv2_act | DISTANCE METRIC: COSINE |

100.0%
What is this? ⓘ

SELECTED IMAGE FERRARYELLOW

All Data By Category

Dataset : iconic200 A dataset of 200 images across 10 categories (20 images per category) crawled from the Flickr API.

Made with ❤ at [Cloudera Fast Forward Labs](#).