

# ラーニングアナリティクス基盤システム

## Docker 環境構築手順書

第 1 版

作成日	2019年3月26日
最終更新日	2019年6月6日



## Table of Contents

ラーニングアナリティクス基盤システム DOCKER 環境構築手順書.....	1
1. LAAAS-DOCKER のインストール .....	3
2. MOODLE のインストール .....	3
2.1 MOODLE コンテナの起動.....	3
2.2 MOODLE の初期設定 .....	4
2.3 XAPI ステートメント作成用の一時テーブル作成 .....	4
3. ラーニングアナリティクス基盤の起動.....	5
3.1 LEARNING LOCKER の設定 .....	8
3.1.1 ログイン .....	8
3.1.2 メールアドレスの認証.....	9
3.1.3 LRS の作成.....	9
3.1.4 クライアント情報の確認 .....	10
3.2 XAPI ステートメント生成スクリプトの設定 .....	11
3.3 XAPI ステートメントの生成 .....	11
3.4 OPENLRW の設定 .....	13
3.5 CALIPER ステートメント生成スクリプトの設定 .....	14
3.6 CALIPER ステートメントの生成.....	14
3.7 SUPERSET の設定 .....	15
3.7.1 ログイン .....	15
3.7.2 Learning Locker データベースの登録 .....	15
3.7.3 ステートメントテーブルの作成 .....	18
3.8 METABASE の設定 .....	20
3.8.1 ログイン .....	20
3.8.2 初期設定およびLearning Locker の連携 .....	21
3.8.3 OpenLRW の連携 .....	22
3.8.4 ステートメントの分析 .....	23
3.9 JUPYTERHUB の設定 .....	25
3.9.1 ログイン .....	25
3.9.2 Learning Locker からステートメントを取得 .....	26
3.9.3 Open LRW からステートメントを取得 .....	27
3.9.4 作成したノートブックの取得 .....	27
3.9.5 ユーザの作成 .....	28
3.10 JUPYTERHUB と SUPERSET の連動 .....	28
3.10.1 LA Integration 機能の実行 .....	28
3.10.2 Superset への登録 .....	30
3.10.3 LA Integration に作成した Jupyter ノートブックを登録 .....	32

&lt;本手順書の前提&gt;

- Docker version 18.09.2
- Docker compose version 1.23.2

## 1. LAaaS-docker のインストール

```
$ git clone https://github.com/RCOSDP/LAaaS-docker.git
```

## 2. Moodle のインストール

### 2.1 Moodle コンテナの起動

Moodle3.4 をダウンロードする。

```
$ cd NII-RCOS
$ git clone https://github.com/moodlehq/moodle-docker
$ cd moodle-docker
$ git clone git://git.moodle.org/moodle.git
$ cd moodle
$ git checkout MOODLE_34_STABLE
$ cd ../
$ export MOODLE_DOCKER_WWWROOT=<path_to_NII-RCOS_directory>/moodle-docker/moodle
$ export MOODLE_DOCKER_DB=pgsql
$ cp config.docker-template.php $MOODLE_DOCKER_WWWROOT/config.php
```

Moodle データベースのユーザ名を `moodleuser` に変更する。

#### `moodle-docker/base.yml`

```
version: "2"
services:
  webserver:
    image: "moodlehq/moodle-php-apache:${MOODLE_DOCKER_PHP_VERSION}"
    depends_on:
      - db
    volumes:
      - "${MOODLE_DOCKER_WWWROOT}:/var/www/html"
      - "${ASSETDIR}/web/apache2_faidumps.conf:/etc/apache2/conf-enabled/apache2_faidumps.conf"
    environment:
      MOODLE_DOCKER_DBTYPE: postgresql
      MOODLE_DOCKER_DBNAME: moodle
      MOODLE_DOCKER_DBUSER: moodleuser
      MOODLE_DOCKER_DBPASS: [REDACTED]
      MOODLE_DOCKER_BROWSER: firefox
      MOODLE_DOCKER_WEB_HOST: "${MOODLE_DOCKER_WEB_HOST}"
  db:
    image: postgres:9.6.7
    environment:
      POSTGRES_USER: moodleuser
      POSTGRES_PASSWORD: [REDACTED]
      POSTGRES_DB: moodle
  exttests:
    image: moodlehq/moodle-exttests
  selenium:
    image: "selenium/standalone-firefox${MOODLE_DOCKER_SELENIUM_SUFFIX}:2.53.1"
    volumes:
      - "${MOODLE_DOCKER_WWWROOT}:/var/www/html:ro"
```

Moodle データベースのテーブル名に付与する接頭辞を `mdl_` とする。

`$MOODLE_DOCKER_WWWROOT/config.php`

```
$CFG->prefix = 'mdl_';
```

Moodle のコンテナを立ち上げる。

```
$ bin/moodle-docker-compose up -d
```

## 2.2 Moodle の初期設定

Moodle(<http://localhost:8000>)にアクセスし、画面の指示に従って初期設定を行う。

### Installation

#### Moodle - Modular Object-Oriented Dynamic Learning Environment

##### Copyright notice

Copyright (C) 1999 onwards Martin Dougiamas (<http://moodle.com>)

This program is free software: you can redistribute it and/or modify  
it under the terms of the GNU General Public License as published by  
the Free Software Foundation, either version 3 of the License, or  
(at your option) any later version.

This program is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

See the Moodle License information page for full details:  
<http://docs.moodle.org/dev/License>

### Confirm

Have you read these conditions and understood them?

Continue

Cancel

## 2.3 xAPI ステートメント作成用の一時テーブル作成

後述する xAPI ステートメント作成スクリプトでイベントの処理状況を管理するためのテーブルを作成する。

```
$ cd ..  
$ docker cp create_xapi_records_processed.sql moodle-docker_db_1:/  
$ docker exec -it moodle-docker_db_1 psql -f create_xapi_records_processed.sql -U moodleuser moodle
```

テーブル定義を確認する場合は以下のコマンドを実行する。

```
$ docker exec -it moodle-docker_db_1 psql -U moodleuser moodle -c " \d xapi_records_processed;"
```

### 3. ラーニングアナリティクス基盤の起動

次のコンテナを起動する。

項目	名称	コンテナ名
学習活動のデータストア	Learning Locker(Web/DB)	learning_locker
	OpenLRW(Web)	openlrw_web
	OpenLRW(DB)	openlrw_mongo
ステートメント生成	xAPI ステートメント生成	xapi_stmt_gen
	Caliper ステートメント生成	caliper
分析システム	Superset(Web)	superset
	Superset(DB)	superset_db
	Metabase(Web/DB)	metabase
	JupyterHub	jupyterhub

```
$ docker-compose up -d
```

各コンテナ間の関係を下図に示す。

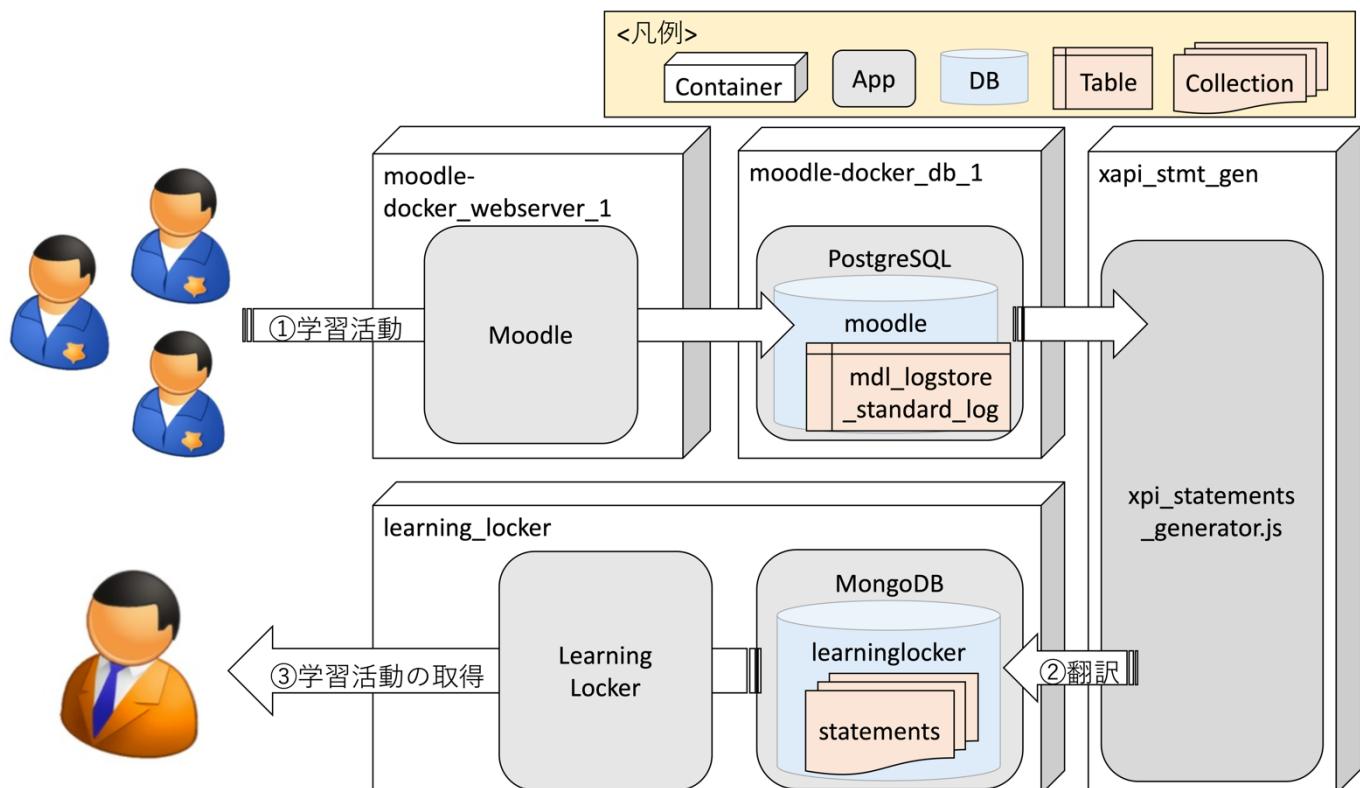


図 3-1 Moodle ログから生成した xAPI ステートメントを Learning Locker に登録

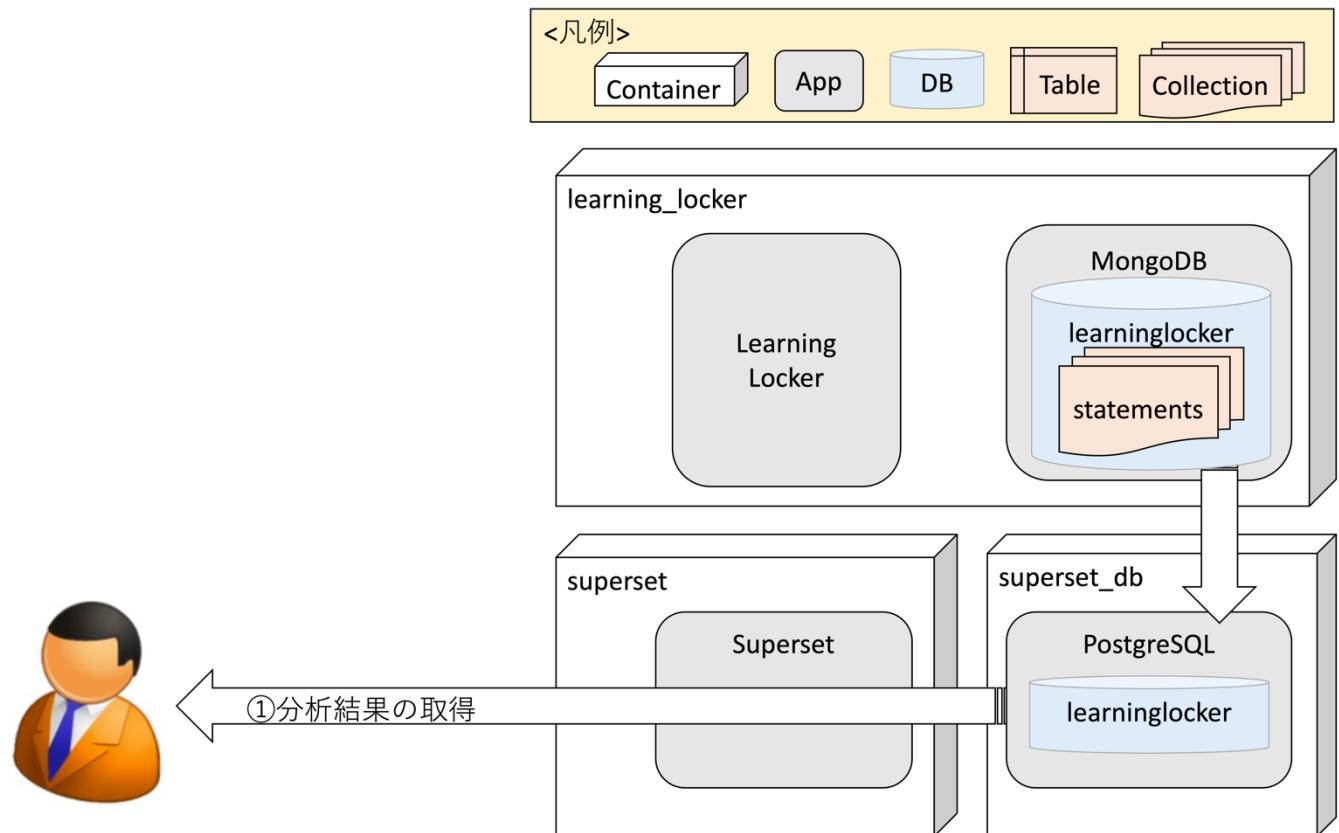


図 3-2 xAPI ステートメントの Superset を用いた分析

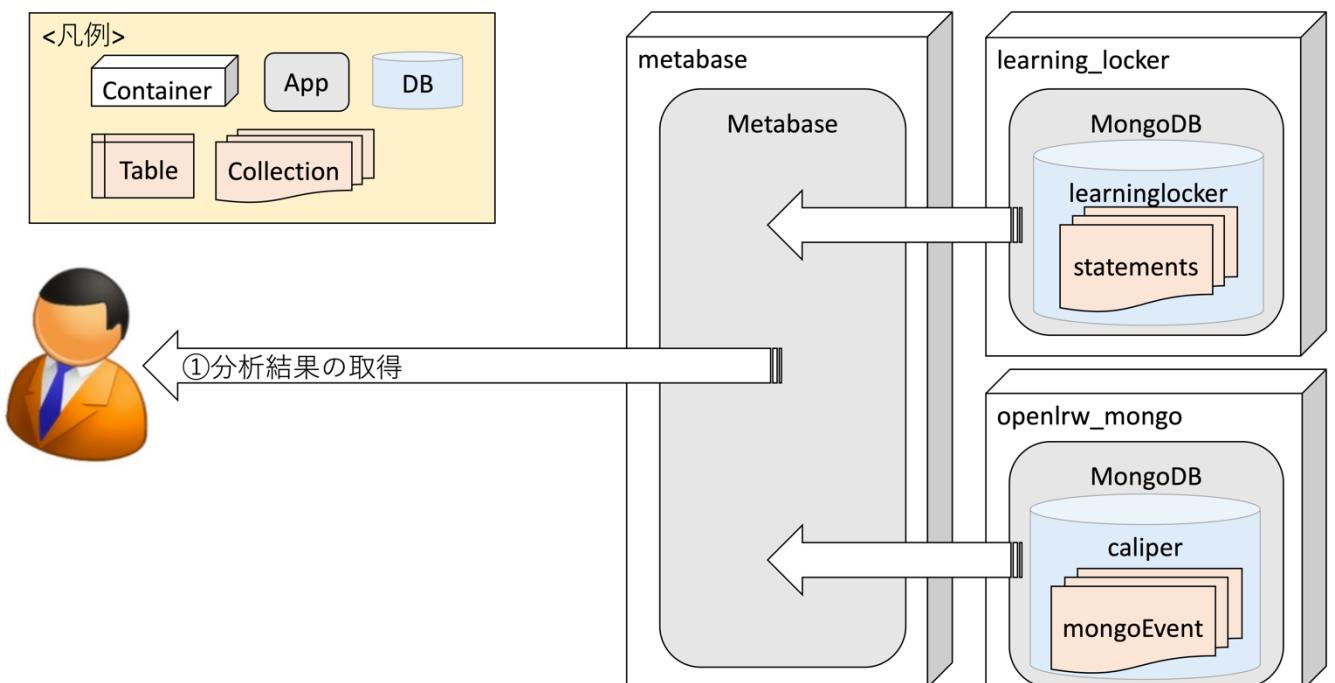


図 3-3 Metabase を用いたステートメントの分析

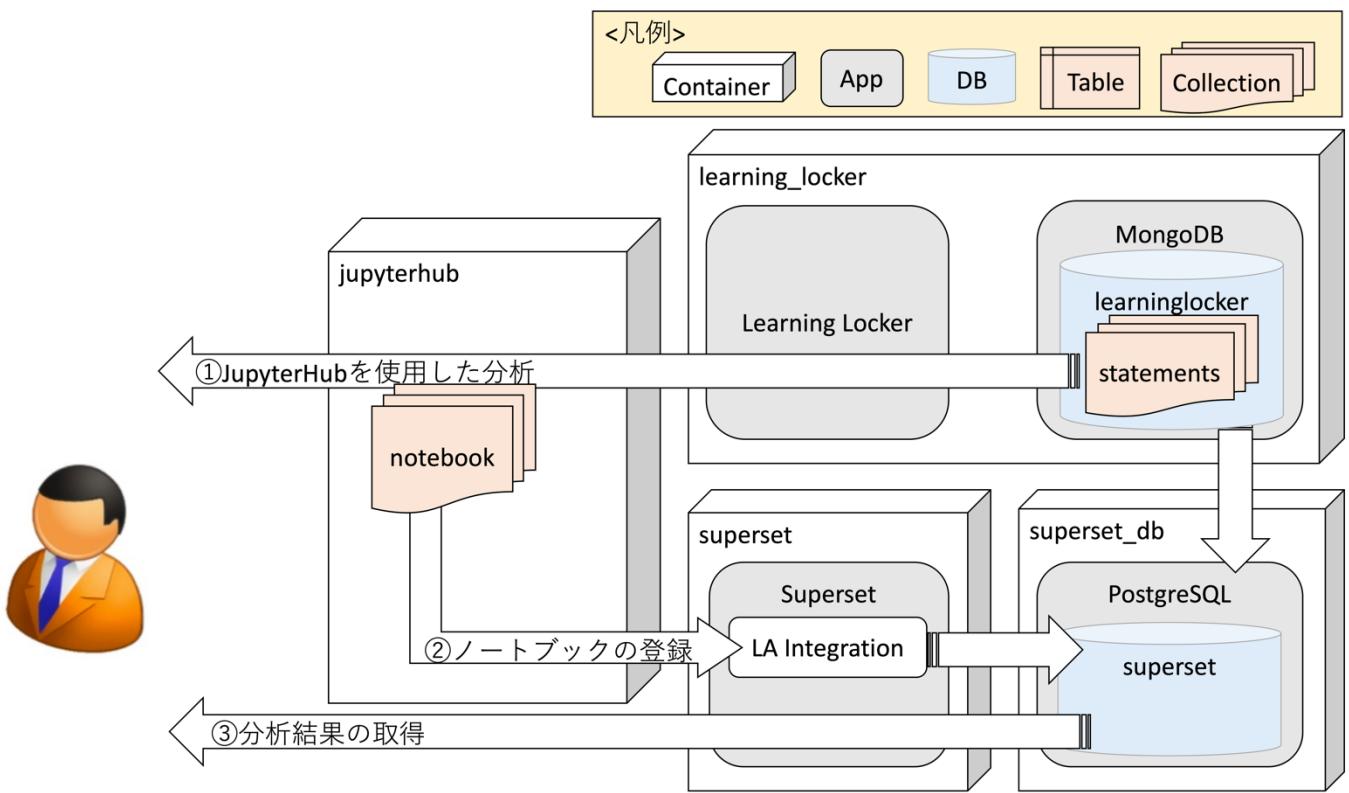


図 3-4 xAPI ステートメントの JupyterHub および Superset の拡張機能を用いた分析

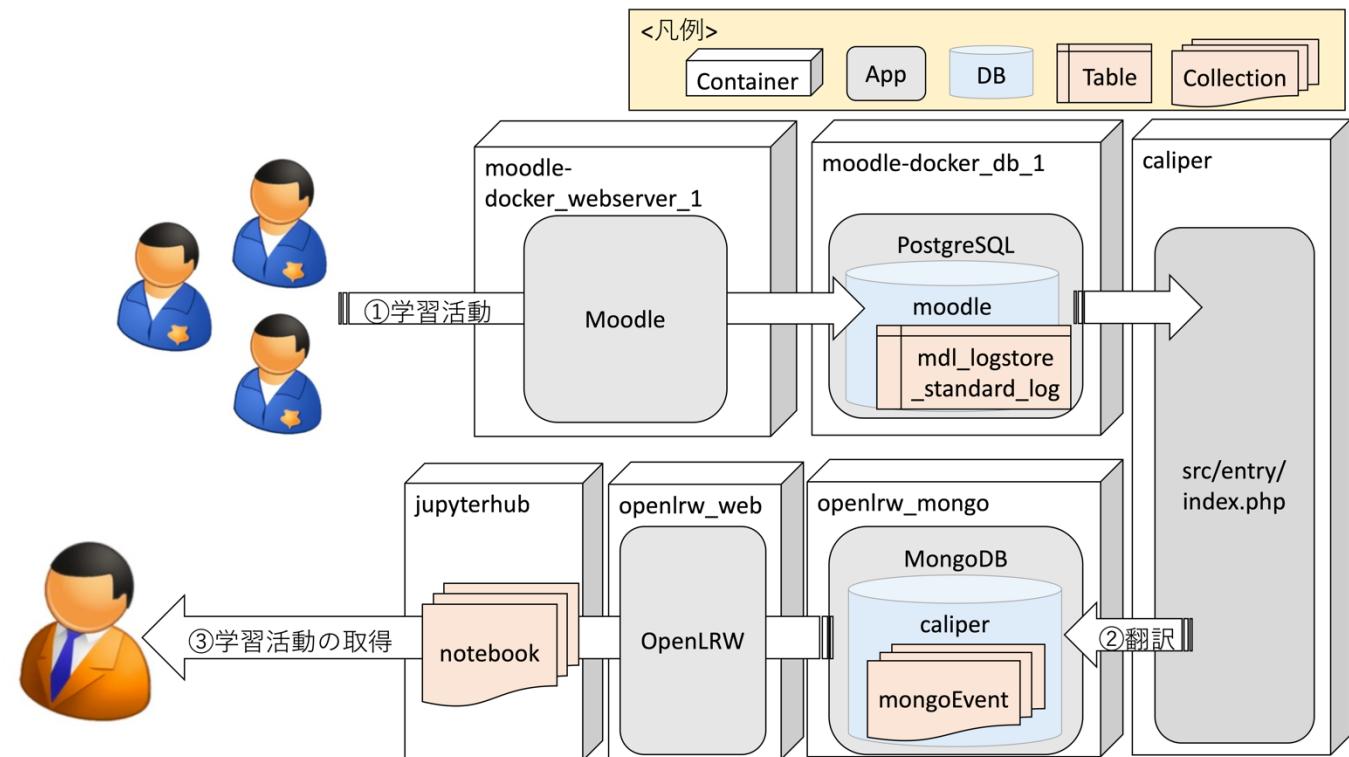
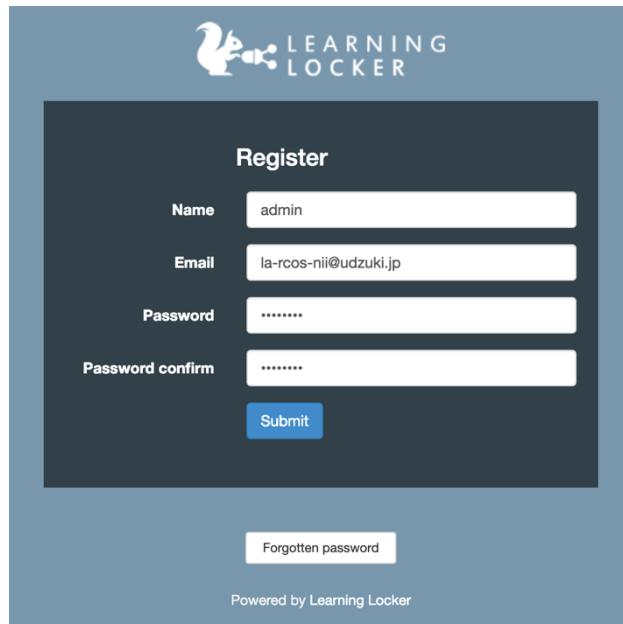


図 3-5 Caliper ステートメントの JupyterHub を用いた分析

### 3.1 Learning Locker の設定

#### 3.1.1 ログイン

Learning Locker(<http://localhost/>)にアクセスし、Admin ユーザのアカウントを作成する。



ダッシュボードが表示されることを確認する。

Admin Options

- Dashboard
- Settings
- LRSs
- Users

Powered by  
Learning Locker  
Version 1.17.0  
Github Releases

Admin Dashboard

Statements 0

Your daily average is 0 statements.

Since 2019/03/15 Until 2019/03/22 Update graph

No data.

0 Total LRSs 0 Total Statements 1 Total Users

Admin dashboard LRS List Settings Logout

### 3.1.2 メールアドレスの認証

サイドメニューの Users からユーザー一覧を表示する。

The screenshot shows the Admin Dashboard with the following details:

- Admin Options** sidebar with **Dashboard** selected.
- Admin Dashboard** title.
- Users** table:
 

Status	Name	Email	LRSs	Role	Joined	Reset Password	Delete
Verified	admin	la-rcos-nii@udzuki.jp		Super	2019-03-22 07:18:47		
- Powered by Learning Locker Version 1.17.0 Github Releases** footer.

Status 列の Verified ボタンを押下し、メールアドレスを認証する。

### Admin Dashboard

Status	Name	Email
Verified	admin	la-rcos-nii@udzuki.jp

### 3.1.3 LRS の作成

サイドメニューの LRSs から任意の名称で LRS を作成する。

The screenshot shows the Admin Dashboard with the following details:

- Admin Options** sidebar with **LRSs** selected.
- Admin Dashboard** title.
- Create an LRS** button.
- LRS** table header:
 

Title	Description	Statement #	User #	Created	Edit	Delete
-------	-------------	-------------	--------	---------	------	--------
- Create an LRS** form:
 

<b>Title</b>	<input type="text" value="moodle"/>
<b>Description</b>	<input type="text"/>
<b>Submit</b>	
- Powered by Learning Locker Version 1.17.0 Github Releases** footer.

### Create an LRS

LRS List / Create an LRS

**Title**

**Description**

**Submit**

The LRS was created.

Title	Description	Statement #	User #	Created	Edit	Delete
moodle		0	1	2019-03-22 07:24:21		

### 3.1.4 クライアント情報の確認

作成した LRS を選択し、ダッシュボード画面に遷移する。

Statements 0

Your daily average is 0 statements.

Since 2019/03/15 Until 2019/03/22

No data.

Most active users

Popular Activities

Latest reports

サイドメニューの Manage clients を選択し、クライアント情報を確認する。

Clients

Endpoint: <http://localhost/data/xAPI/>

Name	Username	Password	Edit	Delete
New Client	82a894b535ea757ff1d18fd1efaab011898460ad	b5a93ba6d37f3eec313fc559d86f5d37e6f7ccc3		

### 3.2 xAPI ステートメント生成スクリプトの設定

xAPI ステートメント生成スクリプトの設定ファイルを編集し、上記で確認した LRS のクライアント情報を保存する。

```
$ docker exec -it xapi_stmt_gen bash
$ vi xapi_config.js # 以下参照
```

/usr/local/src/xapi\_stmt\_gen/xapi\_config.js

```
var config = {
...
  LRS: {
    url:"http://<learning-locker-container-ip-address>/data/xAPI",
    user:"82a894b535ea757ff1d18fd1efaab011898460ad",
    pass:"b5a93ba6d37f3eec313fc559d86f5d37e6f7ccc3"
  },
...
}
```

なお、<learning-locker-container-ip-address>は以下の方法で確認することができる。

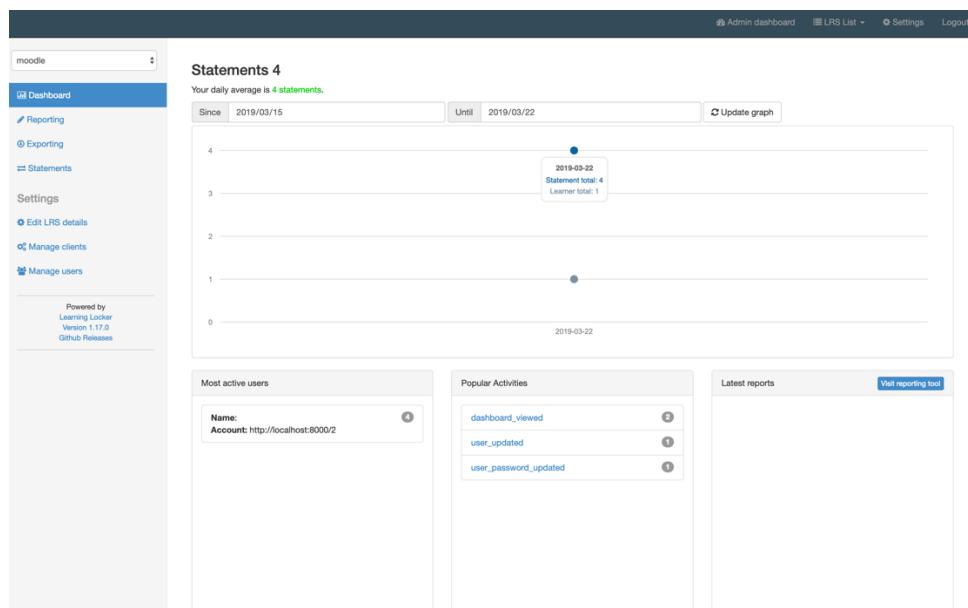
```
$ docker inspect learning_locker
...
  "NetworkSettings": {
...
    "Networks": {
      "moodle-docker_default": {
...
        "IPAddress": "xxx.xxx.xxx.xxx",
...
      }
    }
  }
...
```

### 3.3 xAPI ステートメントの生成

Moodle 上での学習活動を実施した後、以下のコマンドで Moodle ログを xAPI ステートメントに変換する。

```
$ docker exec -it xapi_stmt_gen node xapi_statements_generator.js
```

Learning Locker にアクセスし、ステートメントが正常に登録されていることを確認する。



なお、Moodle ログの処理済み件数を算出する場合は以下のコマンドを実行する。

```
$ docker exec -it moodle-docker_db_1 psql -U moodleuser moodle -c "SELECT count(*) FROM xapi_records_processed;"
```

また、スクリプトのバージョンアップ等に伴い処理済みの Moodle ログを再度処理したい場合は、以下のコマンドで変換処理の実行履歴を削除する。

```
$ docker exec -it moodle-docker_db_1 psql -U moodleuser moodle -c "DELETE FROM xapi_records_processed;"
```

### 3.4 OpenLRW の設定

openlrw\_mongo の起動状況により、openlrw が openlrw\_mongo に接続できず終了している場合があるので確認する。

```
$ docker-compose ps
```

openlrw のステータスが Exit 0 になっていた場合、起動に失敗しているので再起動する。

```
$ docker-compose up -d openlrw
```

ログを確認し、最終行に Started OpenLRW と表示されていることを確認する。

```
$ docker-compose logs -f openlrw
```

(表示例)

```
...
openlrw_web | 2019-03-24 16:43:36.147  INFO 8 --- [           main]
org.apereo.openlrw.OpenLRW                 : Started OpenLRW in 20.922 seconds (JVM running for
23.221)
```

### 3.5 caliper ステートメント生成スクリプトの設定

openlrw\_mongo に接続し、mongodb へログインする。ログインに必要な情報は open\_lrw/.env に記載されている。

```
$ docker-compose exec openlrw_mongo bash
# mongo <OPENLRW_DATABASE> -u <OPENLRW_USERNAME> -p <OPENLRW_PASSWORD>
```

次のコマンドを実行し、apiKey を取得する。

```
> db.mongoOrg.find0.pretty()
(表示例)
{
    "_id" : ObjectId("5c9921dea1de1800064caff1"),
    "apiKey" : "77bb4005-1a77-4984-97bd-1c033fc101e9",
    "apiSecret" : "d2f39c9d-cf5a-43e6-aa2b-2fa993b2cd2f",
    "tenantId" : "5c9921dda1de1800064caff0",
    "org" : {
        "sourcedId" : "7410cd7b-b761-4150-8cf9-a7fdb89a704e",
        "status" : "active",
        "metadata" : {
            "https://matthews/tenant" : "5c9921dda1de1800064caff0"
        },
        "dateLastModified" : ISODate("2019-03-25T18:45:49.988Z"),
        "name" : "DEFAULT_ORG",
        "type" : "other"
    },
    "_class" : "org.apereo.openlrw.oneroster.service.repository.MongoOrg"
}
```

取得した apiKey で caliper/src/entry/index.php 内の setApiKey を設定する。変更を保存したのち、caliper コンテナを再起動する。

```
$ docker-compose build caliper
$ docker-compose up -d caliper
```

### 3.6 caliper ステートメントの生成

Moodle 上での学習活動を実施した後、以下のコマンドで Moodle ログを caliper ステートメントに変換する。

```
$ docker-compose exec caliper php src/entry/index.php
```

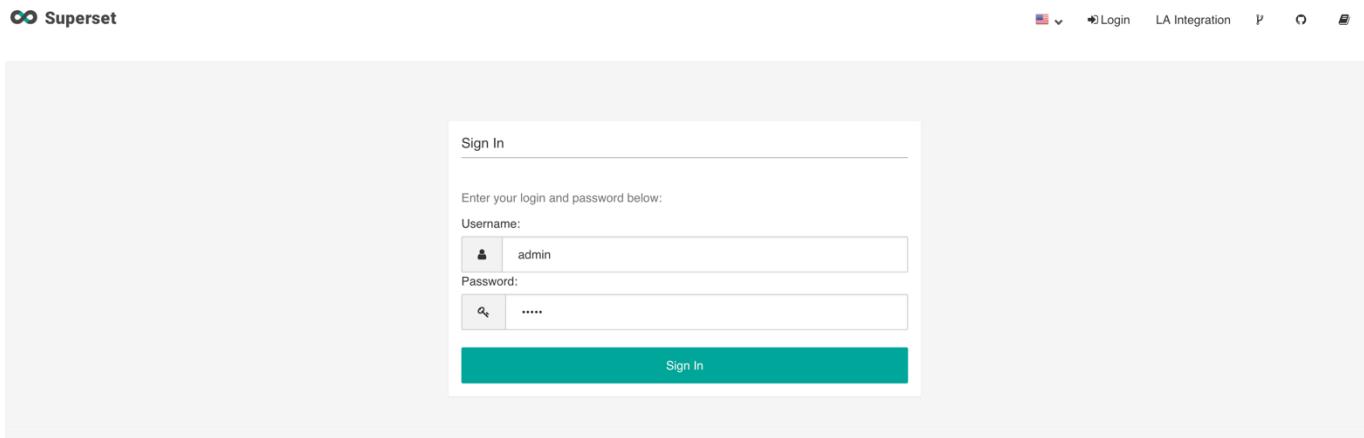
openlrw\_mongo に接続し、データベースに送出したステートメントが登録されていることを確認する。

```
$ docker-compose exec openlrw_mongo bash
# mongo <OPENLRW_DATABASE> -u <OPENLRW_USERNAME> -p <OPENLRW_PASSWORD>
> db.mongoEvent.find0.pretty()
```

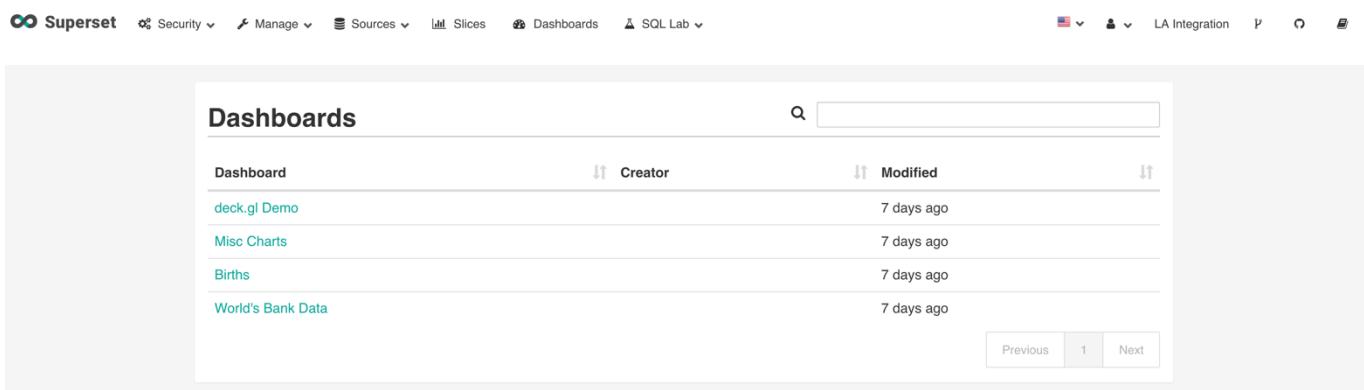
## 3.7 Superset の設定

### 3.7.1 ログイン

Superset(<http://localhost:8088>)にアクセスし、Admin ユーザでログインする(Username:admin、Password:admin)。

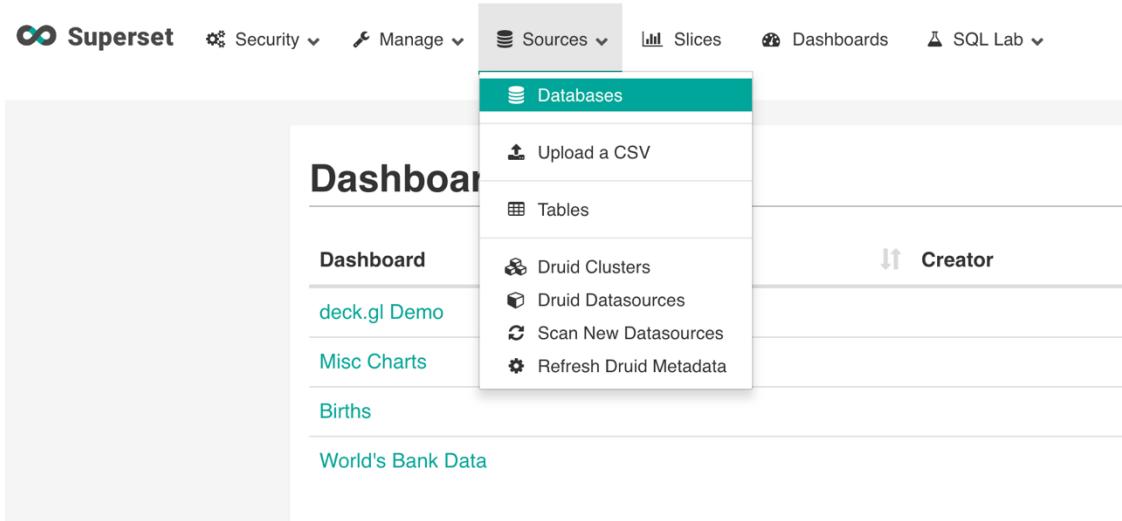


ダッシュボードの一覧が表示されることを確認する。



### 3.7.2 Learning Locker データベースの登録

Sources メニューから Databases を選択する。



データベース一覧の右上にある+ボタンを押下する。

List Databases												
		Database	Backend	Allow Run Sync	Allow Run Async	Allow DML	Creator	Modified				
<input type="checkbox"/>		main	sqlite	True	False	False		7 days ago				
<a href="#">Actions</a>		Record Count: 1										

以下の通りにデータベースの設定を行う。

- Database : learninglocker
- SQLAlchemy URI : postgresql://postgres:postgres@superset\_db:5432/learninglocker
- Expose in SQL Lab : チェック

Add Database	
Database	<input type="text" value="learninglocker"/>
SQLAlchemy URI	<input type="text" value="postgresql://postgres:postgres@superset_db:5432/learninglocker"/> Refer to the <a href="#">SqlAlchemy docs</a> for more information on how to structure your URI. <a href="#">Test Connection</a>
Cache Timeout	<input type="text" value="Cache Timeout"/>
Extra	<pre>{     "metadata_params": {},     "engine_params": {} }</pre> <p>JSON string containing extra configuration elements. The <code>engine_params</code> object gets unpacked into the <code>sqlalchemy.create_engine</code> call, while the <code>metadata_params</code> gets unpacked into the <code>sqlalchemy.MetaData</code> call.</p>
Expose in SQL Lab	<input checked="" type="checkbox"/> Expose this DB in SQL Lab
Allow Run Sync	<input checked="" type="checkbox"/> Allow users to run synchronous queries, this is the default and should work well for queries that can be executed within a web request scope (<~1 minute)
Allow Run Async	<input type="checkbox"/> Allow users to run queries, against an async backend. This assumes that you have a Celery worker setup as well as a results backend.
Allow CREATE TABLE AS	<input type="checkbox"/> Allow CREATE TABLE AS option in SQL Lab
Allow DML	<input type="checkbox"/> Allow users to run non-SELECT statements (UPDATE, DELETE, CREATE, ...) in SQL Lab
CTAS Schema	<input type="text" value="CTAS Schema"/> When allowing CREATE TABLE AS option in SQL Lab, this option forces the table to be created in this schema
Impersonate the logged on user	<input type="checkbox"/> If Presto, all the queries in SQL Lab are going to be executed as the currently logged on user who must have permission to run them. If Hive and hive.server2.enable.doAs is enabled, will run the queries as service account, but impersonate the currently logged on user via hive.server2.proxy.user property.
<a href="#">Save</a>	

Test Connection ボタンを押下し、Seems OK!と表示されることを確認する。

The screenshot shows the Superset interface. On the left, there's a sidebar with 'Sources' and 'Slices' tabs. A modal window is open, displaying the message 'localhost:8088 says Seems OK!' with an 'OK' button. Below the sidebar, there's a list of databases. The first database listed is 'learninglocker'. Underneath it, the database URI 'postgresql://postgres:postgres@superset\_db:5432/learninglocker' is shown. A note below the URI says 'Refer to the [SqlAlchemy docs](#) for more information on how to structure your URI.' A blue box highlights the 'Test Connection' button, which is located next to the database URI. At the bottom of the screen, there's a 'Cache Timeout' section.

Save ボタンを押下し、learninglocker データベースが正常に追加されることを確認する。

The screenshot shows the 'List Databases' page in Superset. A green header bar indicates 'Added Row'. The main table lists two databases: 'learninglocker' and 'main'. The 'learninglocker' row has a timestamp 'now' under 'Modified'. At the bottom left, there's an 'Actions' dropdown menu, and at the bottom right, it says 'Record Count: 2'.

	Database	Backend	Allow Run Sync	Allow Run Async	Allow DML	Creator	Modified
learninglocker	postgres	True	False	False	admin user	now	
main	sqlite	True	False	False		7 days ago	

### 3.7.3 ステートメントテーブルの作成

Sources メニューから Tables を選択する。

The screenshot shows the Superset interface. At the top, there is a navigation bar with 'y ~' (Logout), 'Manage', 'Sources' (selected), 'Slices', 'Dashboards', and 'SQL Lab'. Below the navigation bar, there is a sidebar titled 'List Databases' which contains a table with three rows. The first row has empty columns and icons for search, edit, and delete. The second row has a checkbox, a search icon, and the database name 'main'. The third row also has a checkbox, a search icon, and the database name 'main'. To the right of the sidebar, a modal window is open under the 'Tables' section of the 'Sources' dropdown. The modal lists several options: 'Databases', 'Upload a CSV', 'Tables' (which is highlighted with a teal background), 'Druid Clusters', 'Druid Datasources', 'Scan New Datasources', and 'Refresh Druid Metadata'. Below the modal, there is a table with two columns: 'Allow Run Sync' and 'Allow Impersonate'. The first row has 'True' and 'False'. The second row has 'True' and 'False'. The third row has 'True' and 'False'.

テーブル一覧の右上にある+ボタンを押下する。

The screenshot shows the 'List Tables' page in Superset. At the top, there is a header 'List Tables' and a 'Add Filter' button. Below the header, there is a table with the following data:

	Table	Database	Changed By	Modified
<input type="checkbox"/>	random_time_series	main		7 days ago
<input type="checkbox"/>	multiformat_time_series	main		7 days ago
<input type="checkbox"/>	birth_france_by_region	main		7 days ago
<input type="checkbox"/>	long_lat	main		7 days ago
<input type="checkbox"/>	birth_names	main		7 days ago
<input type="checkbox"/>	wb_health_population	main		7 days ago
<input type="checkbox"/>	energy_usage	main		7 days ago

At the bottom left, there is a 'Actions' dropdown menu. At the bottom right, it says 'Record Count: 7'.

以下の通りにテーブルの設定を行う。

- Database : learninglocker
- Table Name : statements

Add Table

Database *	learninglocker
Schema	Schema Schema, as used only in some databases like Postgres, Redshift and DB2
Table Name	statements Name of the table that exists in the source database

**Save** [←](#)

Save ボタンを押し、 statements テーブルが正常に追加されることを確認する。

The table was created. As part of this two phase configuration process, you should now click the edit button by the new table to configure it.

Added Row

List Tables

	Table	Database	Changed By	Modified
<input type="checkbox"/>	statements	learninglocker	admin user	now
<input type="checkbox"/>	random_time_series	main		7 days ago
<input type="checkbox"/>	multiformat_time_series	main		7 days ago
<input type="checkbox"/>	birth_france_by_region	main		7 days ago

statements テーブルを選択し、Learning Locker に登録済みのステートメント件数が表示されることを確認する(以下の例では 4 件)。

Run Query Save

Datasource & Chart Type

DataSource: statements

Visualization Type: Table View

Time

Time Column: stored, Time Grain: Select 9, Since: 7 days ago, Until: now

GROUP BY

COUNT(\*)

undefined - untitled

00:00:01.63

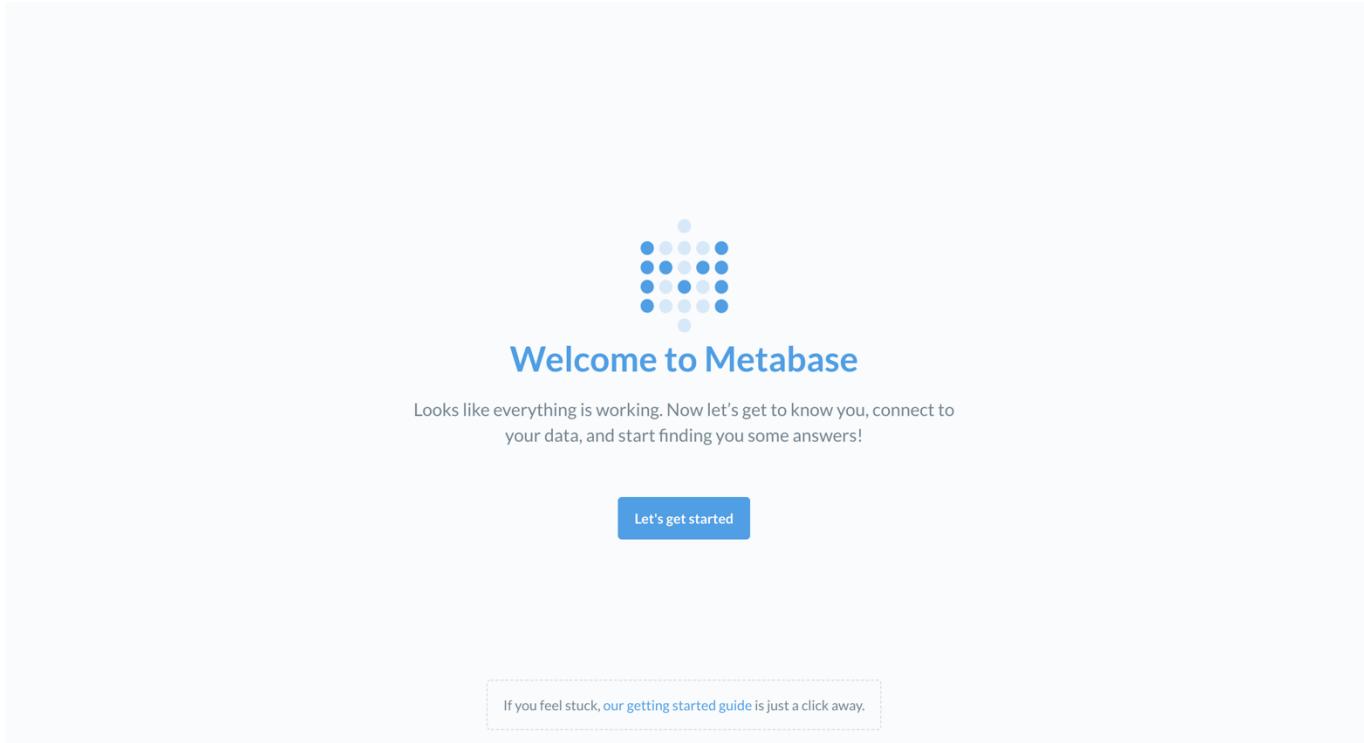
4.00

View Query

### 3.8 Metabase の設定

#### 3.8.1 ログイン

Metabase(<http://localhost:3000/>)にアクセスする。



### 3.8.2 初期設定および Learning Locker の連携

画面の指示に従い、アカウントの作成および分析対象のデータベースを設定する。  
データベースの設定では以下の通り、Learning Locker のデータベースを指定する(パスワードは **udzuki**)。

2 Add your data

You'll need some info about your database, like the username and password. If you don't have that right now, Metabase also comes with a sample dataset you can get started with.

MongoDB

Name:  
**Learning Locker**

Host:  
**learning\_locker**

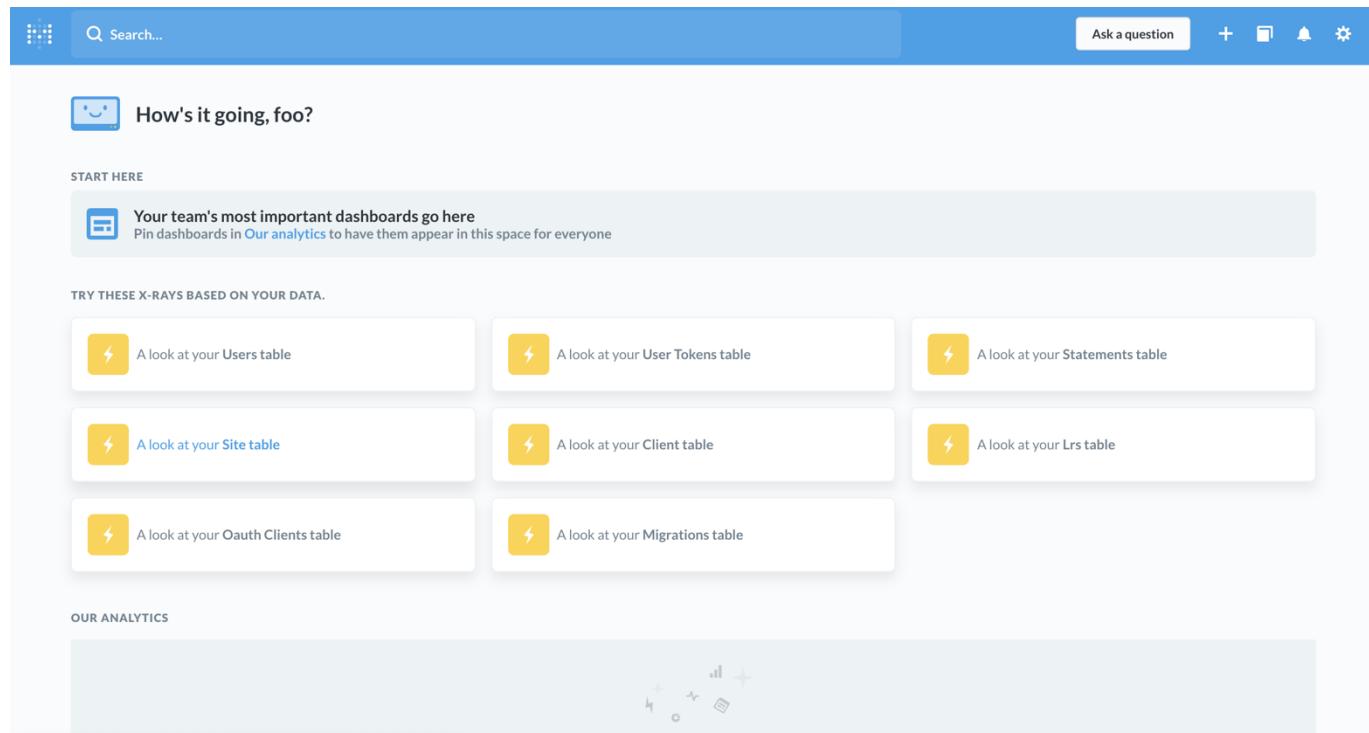
Database name:  
**learninglocker**

Port:  
**27017**

Database username:  
**udzuki**

Database password:  
••••••

初期設定の完了後、以下の通りトップページが表示されることを確認する。

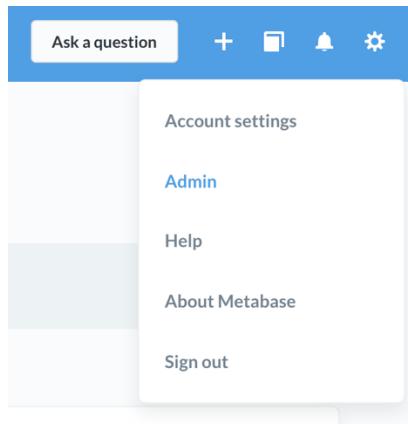


The screenshot shows the Metabase dashboard with the following elements:

- Top navigation bar with icons for user, search, and settings.
- Search bar: Search...
- Right side header: Ask a question, +, 📈, 🚨, 🛡️.
- Main content area:
  - A message: "How's it going, foo?" with a smiley face icon.
  - A "START HERE" section with a blue button: "Your team's most important dashboards go here". It says: "Pin dashboards in [Our analytics](#) to have them appear in this space for everyone".
  - A "TRY THESE X-RAYS BASED ON YOUR DATA." section with six cards:
    - "A look at your Users table"
    - "A look at your User Tokens table"
    - "A look at your Statements table"
    - "A look at your Site table"
    - "A look at your Client table"
    - "A look at your Lrs table"
    - "A look at your Oauth Clients table"
    - "A look at your Migrations table"
  - A "OUR ANALYTICS" section with a small chart icon.

### 3.8.3 OpenLRW の連携

画面右上のメニューから Admin を選択する。



Setup>GET CONNECTED>Add a database を選択する。

A screenshot of the Metabase 'Settings' page. On the left, a sidebar lists various setup options: General, Updates, Email, Slack, Authentication, Maps, Formatting, and Public Sharing. Below this is a link to 'Embedding in other Applications'. The 'Setup' option is currently selected and highlighted in blue. The main content area has two sections: 'Getting set up' and 'RECOMMENDED NEXT STEP'. The 'Getting set up' section contains a 'Set up email' card with a description. The 'RECOMMENDED NEXT STEP' section contains two cards: 'GET CONNECTED' (with a 'Add a database' button) and 'Set Slack credentials'. Both cards have circular icons with checkmarks or question marks next to them.

以下の通り、OpenLRW のデータベースを指定する(パスワードは caliper)。

Database type:  
MongoDB

Name:  
**OpenLRW**

Host:  
**openlrw\_mongo**

Database name:  
**caliper**

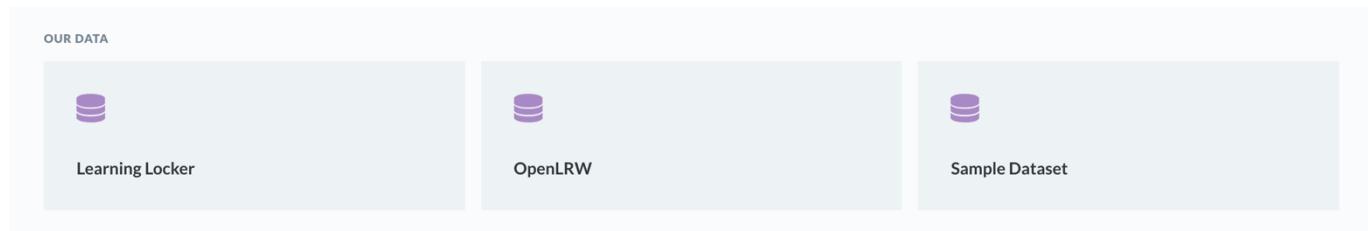
Port:  
**27017**

Database username:  
**caliper**

Database password:  
**\*\*\*\*\***

### 3.8.4 ステートメントの分析

トップページの OUR DATA から Learning Locker または OpenLRW を選択する。



#### 3.8.4.1 xAPI ステートメントの分析

OUR DATA で Learning Locker を指定した後、Statements を選択する。

The screenshot shows the 'LEARNING LOCKER' section of a web application. It displays several components:

- Client**, **Lrs**, **Migrations**
- Oauth Clients**, **Sessions**, **Site**
- Statements** (highlighted with a red box)
- User Tokens**, **Users**

xAPIステートメントの一覧が表示されることを確認する。

New question		SAVE			
DATA		VIEW			
Statements		FILTERED BY			
Add filters to narrow your answer		Raw data			
GROUPED BY		Add a grouping			
VISUALIZATION		Showing 223 rows			
<span>Table</span> <span>grid</span>		<span>Refresh</span>			
ID	Active	Client ID	Lrs	Lrs ID	Statement
5c9c781b5596678c018b4567	true	5c99f147af1ec72a008b4568	{"_id": "5c99f147af1ec72a008b4567"}	5c99f147af1ec72a008b4567	{"stored": "2019-03-28T07:30:32.178700+00:00", "verb": {"id": "urn:x:moodl
5c9c781b5596679a018b4567	true	5c99f147af1ec72a008b4568	{"_id": "5c99f147af1ec72a008b4567"}	5c99f147af1ec72a008b4567	{"stored": "2019-03-28T07:30:32.246200+00:00", "verb": {"id": "urn:x:moodl
5c9c781b5596679b018b4567	true	5c99f147af1ec72a008b4568	{"_id": "5c99f147af1ec72a008b4567"}	5c99f147af1ec72a008b4567	{"stored": "2019-03-28T07:30:32.404400+00:00", "verb": {"id": "urn:x:moodl
5c9c781a55966778018b4567	true	5c99f147af1ec72a008b4568	{"_id": "5c99f147af1ec72a008b4567"}	5c99f147af1ec72a008b4567	{"stored": "2019-03-28T07:30:31.022400+00:00", "verb": {"id": "urn:x:moodl
5c9c781a55966774018b4567	true	5c99f147af1ec72a008b4568	{"_id": "5c99f147af1ec72a008b4567"}	5c99f147af1ec72a008b4567	{"stored": "2019-03-28T07:30:31.242400+00:00", "verb": {"id": "urn:x:moodl
5c9c781a55966780018b4567	true	5c99f147af1ec72a008b4568	{"_id": "5c99f147af1ec72a008b4567"}	5c99f147af1ec72a008b4567	{"stored": "2019-03-28T07:30:31.122900+00:00", "verb": {"id": "urn:x:moodl
5c9c781a55966773018b4567	true	5c99f147af1ec72a008b4568	{"_id": "5c99f147af1ec72a008b4567"}	5c99f147af1ec72a008b4567	{"stored": "2019-03-28T07:30:31.197800+00:00", "verb": {"id": "urn:x:moodl
5c9c781a5596677e018b4567	true	5c99f147af1ec72a008b4568	{"_id": "5c99f147af1ec72a008b4567"}	5c99f147af1ec72a008b4567	{"stored": "2019-03-28T07:30:31.231300+00:00", "verb": {"id": "urn:x:moodl
E-0-704-EE00-L7D0-70-000LAEZ7	true	E-004447-44-70-000LAEZ7	{"_id": "E-0-704-EE00-L7D0-70-000LAEZ7"}	E-004447-44-70-000LAEZ7	{"stored": "2019-03-28T07:30:24.767900+00:00", "verb": {"id": "urn:x:moodl

#### 3.8.4.2 Caliper ステートメントの分析

OUR DATA で OpenLRW を指定した後、Mongo Event を選択する。



The screenshot shows the OpenLRW dashboard interface. At the top, there is a search bar with the placeholder "Search..." and a "Ask a question" button. On the right side of the header, there are icons for adding a new item, opening a new tab, notifications, and settings. Below the header, the navigation path "OUR DATA > OPENLRW" is displayed. The main content area contains five items arranged in two rows: "Admin User", "Mongo Event" (which is highlighted with a red rectangular border), "Mongo Org", "Mongo Result", and "Tenant". Each item has a small icon to its left.

Caliperステートメントの一覧が表示されることを確認する。

The screenshot shows the Jupyter Notebook interface with a blue header bar. The top left has a search bar and the top right has an 'Ask a question' button, a plus sign, a file icon, a bell icon, and a gear icon. Below the header, the title 'New question' is displayed. On the left, there are tabs for 'DATA' (selected), 'FILTERED BY' (with a dropdown for 'Mongo Event' and a '+ Add filters to narrow your answer' button), 'VIEW' (with 'Raw data' selected and a '+ Add a grouping' button), and 'GROUPED BY'. On the right, there are buttons for 'SAVE' and various document icons. Below these, the word 'VISUALIZATION' is followed by a dropdown menu ('Table' selected) and a gear icon. In the center, there is a 'Refresh' button. To the right, it says 'Showing 6 rows' with a download arrow icon. The main area displays a table with six rows of data:

ID	Class	Event
5ca19b6eadf81c0006974537	org.apereo.openlrw.caliper.service.repository.MongoEvent	{"_id": "urn:uuid:a117dd11-2e68-4028-aa80-b9e2907f1920", "context": "http://purl.imsglobal.org/ctx/caliper/v1p1", "type": "SessionEvent"}
5ca19b71adf81c0006974538	org.apereo.openlrw.caliper.service.repository.MongoEvent	{"_id": "urn:uuid:7e4a907a-8186-4426-94fd-06c37d39419d", "context": "http://purl.imsglobal.org/ctx/caliper/v1p1", "type": "SessionEvent"}
5ca19b71adf81c0006974539	org.apereo.openlrw.caliper.service.repository.MongoEvent	{"_id": "urn:uuid:947b2bab-a3ac-4080-8814-7bd04455ba20", "type": "NavigationEvent", "agent": {"_id": "https://example.edu/user/2", "type": "UserAgent", "label": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4369.90 Safari/537.36", "platform": "Windows 10", "version": "89.0.4369.90", "browser": "Chrome", "os": "Windows 10", "device": "Desktop", "language": "en-US", "viewport": "width=device-width, initial-scale=1"}, "page": {"_id": "urn:uuid:996ea9299d1", "url": "https://example.edu/page/1", "title": "Page 1", "content": "This is the content of Page 1.", "type": "Page"}, "event": {"_id": "urn:uuid:996ea9299d1", "name": "Page View", "start_time": "2021-09-21T14:30:00Z", "end_time": "2021-09-21T14:30:15Z", "duration": 15000, "type": "Event"}, "timestamp": "2021-09-21T14:30:00Z"}}, "type": "NavigationEvent"}
5ca1a052adef81c000697453a	org.apereo.openlrw.caliper.service.repository.MongoEvent	{"_id": "urn:uuid:8e5fadda-9100-4446-898c-a996ea9299d1", "context": "http://purl.imsglobal.org/ctx/caliper/v1p1", "type": "SessionEvent"}
5ca1a052adef81c000697453b	org.apereo.openlrw.caliper.service.repository.MongoEvent	{"_id": "urn:uuid:e3667615-c0ff-405c-bea-3c7b87ace77c", "context": "http://purl.imsglobal.org/ctx/caliper/v1p1", "type": "SessionEvent"}

### 3.9 JupyterHub の設定

#### 3.9.1 ログイン

JupyterHub(<http://localhost:8001>)にアクセスし、jupyter ユーザ(Username:jupyter、Password:jupyter)でログインする。

Sign in

Warning: JupyterHub seems to be served over an unsecured HTTP connection. We strongly recommend enabling HTTPS for JupyterHub.

Username: jupyter

Password:  ······

ノートブック一覧が表示されることを確認する。

jupyter

Logout Control Panel

Files Running Clusters

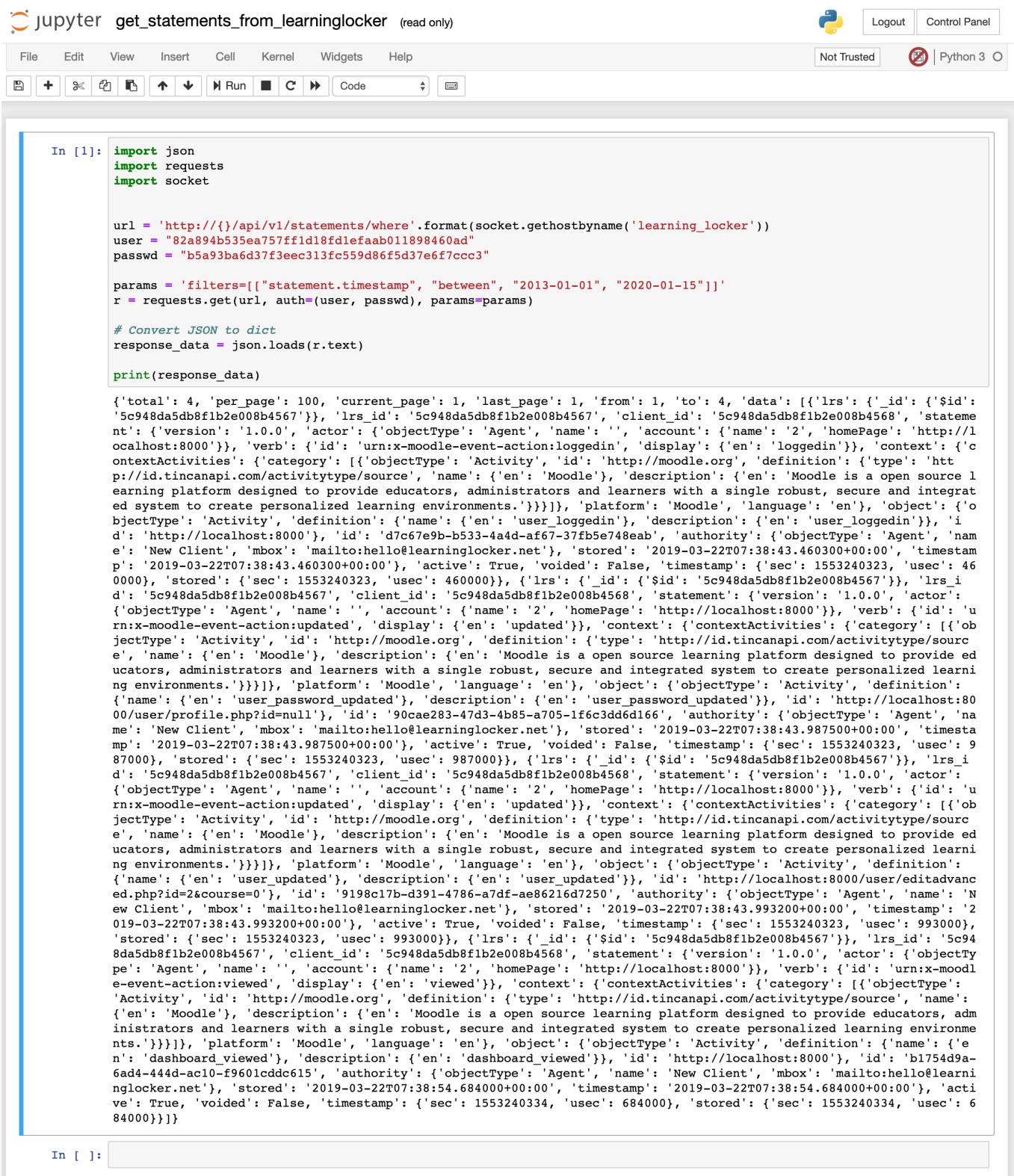
Select items to perform actions on them.

	Name	Last Modified	File size
<input type="checkbox"/> 0	/		
<input type="checkbox"/>	get_statements_from_learninglocker.ipynb	a day ago	18.1 kB
<input type="checkbox"/>	get_statements_from_openlrw.ipynb	17 hours ago	1.62 kB

### 3.9.2 Learning Locker からステートメントを取得

サンプルとして提供するノートブック「get\_statements\_from\_learninglocker」を活用し、Learning Locker の API を使用したステートメント取得が可能であることを確認する。

user、passwd に LRS クライアントの Username、Password をそれぞれ設定し、params でフィルタ条件を変更することで任意のステートメントを取得する。



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Jupyter get\_statements\_from\_learninglocker (read only)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Not Trusted, Logout, Control Panel, Python 3
- Code Cell (In [1]):**

```

In [1]: import json
import requests
import socket

url = 'http://{}:api/v1/statements/where'.format(socket.gethostname('learning_locker'))
user = "82a894b535ea757ff1d18fd1efaab011898460ad"
passwd = "b5a93ba6d37f3eec313fc559d86f5d37e6f7ccc3"

params = 'filters=[["statement.timestamp", "between", "2013-01-01", "2020-01-15"]]' 
r = requests.get(url, auth=(user, passwd), params=params)

# Convert JSON to dict
response_data = json.loads(r.text)

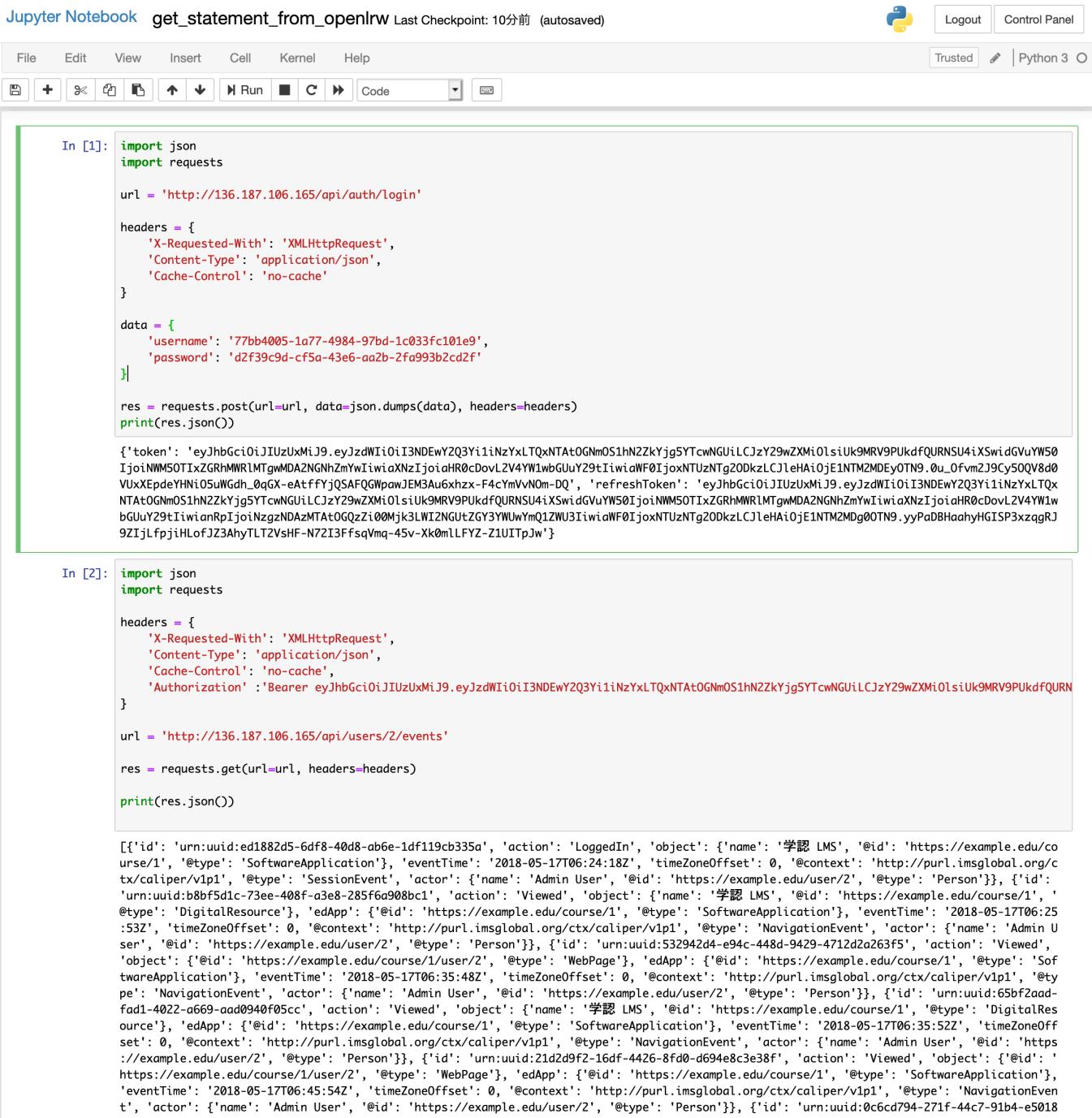
print(response_data)

```
- Output:** The output cell shows a large JSON object representing the fetched statements, which is too long to display fully here.

### 3.9.3 Open LRW からステートメントを取得

サンプルとして提供するノートブック「get\_statements\_from\_openlrw」を活用し、OpenLRW の API を使用したステートメント取得が可能であることを確認する。

まず username、password に OpenLRW クライアントの Username、Password をそれぞれ設定し、JWT トークンを取得する。username、password は、それぞれ 3.5 で取得した apiKey と apiSecret である。その後、Authorization ヘッダーに取得した JWT トークンを設定し、url のユーザー ID を変更することで任意のユーザーのイベントを取得する。



```

In [1]: import json
import requests

url = "http://136.187.106.165/api/auth/login"

headers = {
    'X-Requested-With': 'XMLHttpRequest',
    'Content-Type': 'application/json',
    'Cache-Control': 'no-cache'
}

data = {
    'username': '77bb4005-1a77-4984-97bd-1c033fc101e9',
    'password': 'd2f39c9d-cf5a-43e6-aa2b-2fa993b2cd2f'
}

res = requests.post(url=url, data=json.dumps(data), headers=headers)
print(res.json())

{'token': 'eyJhbGciOiJIUzUxMiJ9.eyJzdWIoiI3NDEwY2Q3Yi1iNzYxLTQxNTAtOGNmOS1hN2ZkYjg5YTcwNGUiLCJzY29wZXMiOlsiuK9MRV9PUkdfQURNSU4iXSwidGVuYW50IjoiNW50TIXZGRhMWRlMTgwMDA2NGNhZmYwIiwiAxNzIjoiaHR0cDovL2V4YW1wbGUyZ9tIiwiWF0IjoxNTuZNg20DkzLCJleHAiOjE1NTM2MDEyOTN9.0u_0Fvm2J9Cy50QV8d0VUxExpdpeYHnI0SuWgdh_0qGX-eatffYjQSAFQGWpawJEM3au6xhxz-F4cYmVvN0m-DQ', 'refreshToken': 'eyJhbGciOiJIUzUxMiJ9.eyJzdWIoiI3NDEwY2Q3Yi1iNzYxLTQxNTAtOGNmOS1hN2ZkYjg5YTcwNGUiLCJzY29wZXMiOlsiuK9MRV9PUkdfQURNSU4iXSwidGVuYW50IjoiNW50TIXZGRhMWRlMTgwMDA2NGNhZmYwIiwiAxNzIjoiaHR0cDovL2V4YW1wbGUyZ9tIiwiRpijoiNzgNDAzMTAtOGQzZi0Mjk3LWI2NGuTzGY3YWUwYmQ1ZWu3IiwiWF0IjoxNTuZNg20DkzLCJleHAiOjE1NTM2MDg00TN9.yyPaDBHaahyHGISP3xzqgRJ9ZijLfpjiHLofJZ3AhylTLT2VsHF-N7213FsqVm-45v-Xk0mlLFYZ-Z1U1TpJw'}
```

```

In [2]: import json
import requests

headers = {
    'X-Requested-With': 'XMLHttpRequest',
    'Content-Type': 'application/json',
    'Cache-Control': 'no-cache',
    'Authorization': 'Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIoiI3NDEwY2Q3Yi1iNzYxLTQxNTAtOGNmOS1hN2ZkYjg5YTcwNGUiLCJzY29wZXMiOlsiuK9MRV9PUkdfQURNSU4iXSwidGVuYW50IjoiNW50TIXZGRhMWRlMTgwMDA2NGNhZmYwIiwiAxNzIjoiaHR0cDovL2V4YW1wbGUyZ9tIiwiWF0IjoxNTuZNg20DkzLCJleHAiOjE1NTM2MDg00TN9.yyPaDBHaahyHGISP3xzqgRJ9ZijLfpjiHLofJZ3AhylTLT2VsHF-N7213FsqVm-45v-Xk0mlLFYZ-Z1U1TpJw'}
```

### 3.9.4 作成したノートブックの取得

作成したノートブックは jupyterhub コンテナの/home/<username>に保存される。

```
$ docker exec -it jupyterhub ls /home/jupyter
get_statements_from_learninglocker.ipynb  get_statements_from_openlrw.ipynb
```

### 3.9.5 ユーザの作成

以下のコマンドで jupyterhub コンテナ上にユーザを作成する。

```
$ docker exec -it jupyterhub useradd -m -p $(echo "<password>" | openssl passwd -1 -stdin) -s /bin/bash <username>
```

### 3.10 JupyterHub と Superset の連動

#### 3.10.1 LA Integration 機能の実行

Superset にログインし、画面右上の LA Integration メニューを選択する。

Dashboard	Creator	Modified
deck.gl Demo		7 days ago
Misc Charts		7 days ago
Births		7 days ago
World's Bank Data		7 days ago

LA Integration では、superset コンテナの/root/Superset\_Dir に格納した Jupyter ノートブックを Analytical Model として実行できる。

ノートブックを実行する対象の Moodle のコースを選択し、提供するサンプルノートブック

「example\_book\_for\_customized\_superset.ipynb」が選択されていることを確認し、Execute Model ボタンを押下する。

Select a Model	Analytical Models
<input checked="" type="radio"/>	example_book_for_customized_superset.ipynb

実行完了画面が表示されることを確認する。

以下のコマンドで `superset_db` コンテナ上にノートブックの実行によって加工されたステートメントが保存されていることを確認する。

```
$ docker exec -it superset_db psql superset -c "SELECT * FROM test1csvtable;"  
timestamp | lrs_id | client_id  
-----+-----+-----  
2019-03-22 09:01:16.458000 | 5c948da5db8f1b2e008b4567 | 5c948da5db8f1b2e008b4568  
2019-03-22 09:01:16.461000 | 5c948da5db8f1b2e008b4567 | 5c948da5db8f1b2e008b4568  
(2 rows)
```

### 3.10.2 Supersetへの登録

Superset 上で上記テーブルが格納された superset データベースを以下の通り登録する。

- Database : superset
- SQLAlchemy URI : postgresql://postgres:postgres@superset\_db:5432/superset
- Expose in SQL Lab : チェック

Add Database

Database	superset
SQLAlchemy URI	postgresql://postgres:postgres@superset_db:5432/superset Refer to the <a href="#">SqlAlchemy docs</a> for more information on how to structure your URI.
Test Connection	
Cache Timeout	Cache Timeout
Extra	<pre>{   "metadata_params": {},   "engine_params": {} }</pre> <p>JSON string containing extra configuration elements. The <code>engine_params</code> object gets unpacked into the <code>sqlalchemy.create_engine</code> call, while the <code>metadata_params</code> gets unpacked into the <code>sqlalchemy.MetaData</code> call.</p>
Expose in SQL Lab	<input checked="" type="checkbox"/> Expose this DB in SQL Lab
Allow Run Sync	<input checked="" type="checkbox"/> Allow users to run synchronous queries, this is the default and should work well for queries that can be executed within a web request scope (<~1 minute)
Allow Run Async	<input type="checkbox"/> Allow users to run queries, against an async backend. This assumes that you have a Celery worker setup as well as a results backend.
Allow CREATE TABLE AS	<input type="checkbox"/> Allow CREATE TABLE AS option in SQL Lab
Allow DML	<input type="checkbox"/> Allow users to run non-SELECT statements (UPDATE, DELETE, CREATE, ...) in SQL Lab
CTAS Schema	CTAS Schema When allowing CREATE TABLE AS option in SQL Lab, this option forces the table to be created in this schema
Impersonate the logged on user	<input type="checkbox"/> If Presto, all the queries in SQL Lab are going to be executed as the currently logged on user who must have permission to run them. If Hive and hive.server2.enable.doAs is enabled, will run the queries as service account, but impersonate the currently logged on user via hive.server2.proxy.user property.

Save Back

Tables:  
test1csvtable

Superset データベースの test1csvtable を以下の通りに登録する。

- Database : superset
- Table Name : test1csvtable

Add Table

Database *	superset
Schema	Schema Schema, as used only in some databases like Postgres, Redshift and DB2
Table Name	test1csvtable Name of the table that exists in the source database

Save Back

The table was created. As part of this two phase configuration process, you should now click the edit button by the new table to configure it.

Added Row

List Tables

	Table	Database	Changed By	Modified	I
<input type="checkbox"/>	test1csvtable	superset	admin user	now	
<input type="checkbox"/>	random_time_series	main		7 days ago	
<input type="checkbox"/>	multiformat_time_series	main		7 days ago	
<input type="checkbox"/>	birth_france_by_region	main		7 days ago	
<input type="checkbox"/>	long_lat	main		7 days ago	
<input type="checkbox"/>	birth_names	main		7 days ago	
<input type="checkbox"/>	wb_health_population	main		7 days ago	
<input type="checkbox"/>	energy_usage	main		7 days ago	

Actions ▾ Record Count: 8

ノートブックが出力した加工済みのステートメント件数が表示されることを確認する。

Run Query Save

Datasource & Chart Type

Datasource: test1csvtable

Visualization Type: Table View

Time

Time Column: Select 0 Time Grain: Select 9

Since: 7 days ago Until: now

GROUP BY

NOT GROUPED BY

Options

SQL

Filters

+ Add Filter

undefined - untitled

COUNT(\*)

00:00:00.54 2.00

### 3.10.3 LA Integration に作成した Jupyter ノートブックを登録

JupyterHub(<http://localhost:8001>)で作成したノートブック(Python2 で実装すること)を jupyterhub コンテナの /home/<username> からダウンロードし、superset コンテナの /root/SuperDir にアップロードする。

```
$ docker cp jupyterhub:/home/<username>/<notebook> .
$ docker cp <notebook> superset:/root/Superset_Dir/
```

作成するノートブックは以下の機能を実装すること(example\_book\_for\_customized\_superset.ipynb を参考)。

- ・LRS からのステートメント取得・加工
- ・/root/Superset\_Dir/csv-folder に CSV 形式でステートメントを出力