

# OMOP環境構築 手順書

2021年3月31日

第 1 . 0 版

# 目次

1. 本手順書について
2. システム構成
3. 事前準備
  3. 1. Dockerのインストール
  3. 2. docker-compose のインストール
  3. 3. R Studio のインストール
  3. 4. Libxml2 のインストール
  3. 5. GitHub からの資産ダウンロード
4. データベース構築
  4. 1. スキーマの作成
  4. 2. OMOP Common Data Model テーブルの作成
  4. 3. レコード投入方法のご紹介
5. Broadseaの構築
  5. 1. Broadseaイメージの取得
  5. 2. 設定ファイルの編集
  5. 3. Achilles / CohortMethod のインストール
  5. 4. 初回起動と起動後の追加設定
  5. 5. Achillesの実行
6. ATLAS画面表示

## 1. 本手順書について

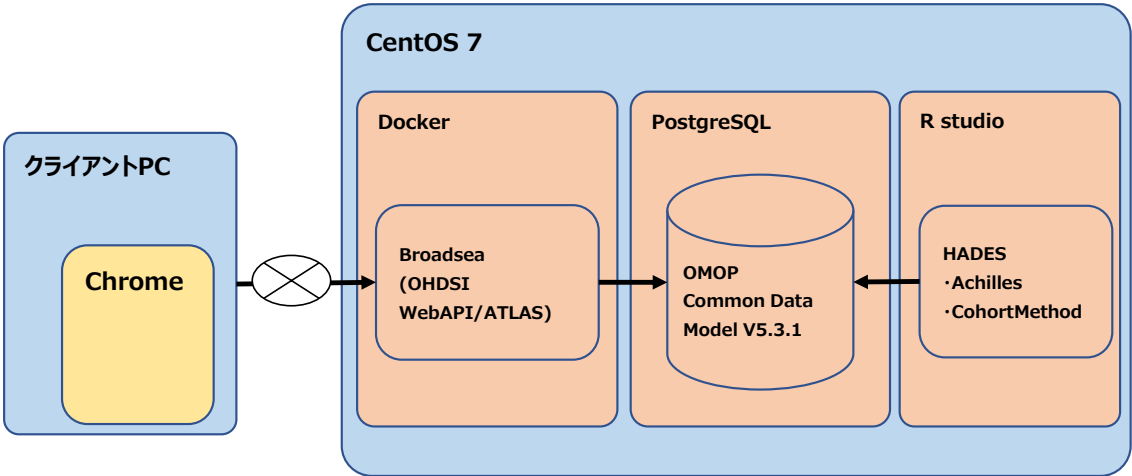
OMOP CDMの検証環境を構築するための手順をご説明いたします。

※画面イメージについてはダミーデータを用いているため、実際のデータとは

データ数・画面に表示される値が異なりますのでご注意ください。

※各パッケージのインストールには、インターネット接続が必要です。

## 2. システム構成



Docker	<a href="https://www.docker.com/">https://www.docker.com/</a> 本手順書では、バージョン20.10.3を使用します。
Broadsea	<a href="https://github.com/hapifhir/hapi-fhir-ipaserver-starter">https://github.com/hapifhir/hapi-fhir-ipaserver-starter</a>
R studio	<a href="https://rstudio.com/">https://rstudio.com/</a>
HADES	<a href="https://ohdsi.github.io/Hades/rSetup.html">https://ohdsi.github.io/Hades/rSetup.html</a>

### 3. 事前準備

Broadseaの動作環境構築のための事前準備をします。

#### 3. 1. Dockerのインストール

対象サーバに root 特権のあるユーザでログインします。

既存の yum パッケージを更新します。

```
# yum update
# yum upgrade
```

公式の安定版 Yum リポジトリを設定します。

```
# yum install -y yum-utils device-mapper-persistent-data lvm2
# yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

Docker をインストールします（Broadseaの操作に必要となります）。

```
# yum makecache fast
# yum install -y docker-ce
```

以下のコマンドを実行し、Dockerのバージョン情報が表示されればインストール完了です。

```
# docker version
```

Dockerを起動します。

```
# systemctl start docker
```

### 3. 2. docker-compose のインストール

対象サーバに root 特権のあるユーザでログインします。

docker-compose をインストールします（Broadsea 起動に必要となります）。

```
# sudo curl -L https://github.com/docker/compose/releases/download/1.16.1/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
# sudo chmod +x /usr/local/bin/docker-compose
```

### 3. 3. R Studio のインストール

対象サーバに root 特権のあるユーザでログインします。

R studio を動作させるためのユーザを追加します。

```
# useradd rstudio
# usermod -aG wheel rstudio
# echo '[新しいパスワード]' | passwd --stdin rstudio
```

EPELをインストールし、サードパーティ製のパッケージをインストール可能にします。

```
# yum install epel-release
```

R をインストールします。

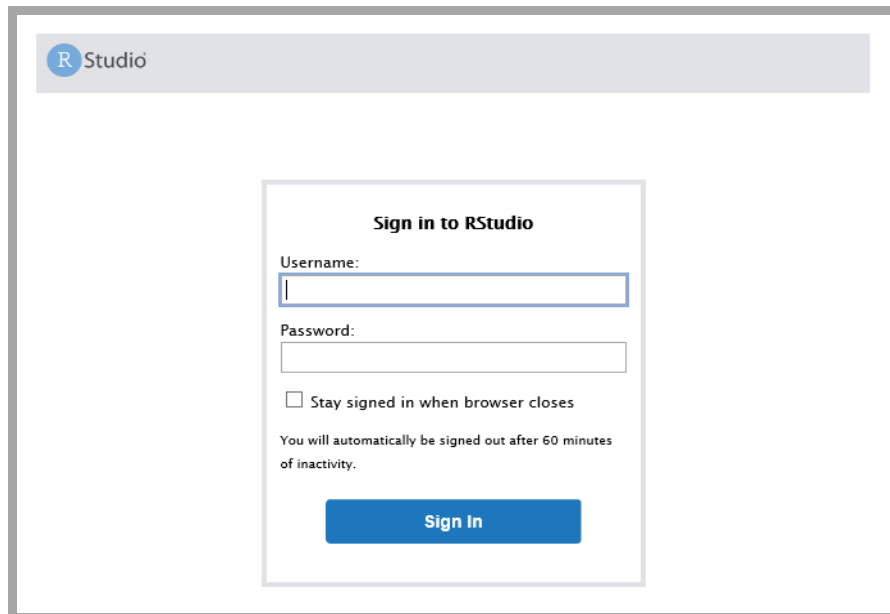
```
# yum install R
```

R studio をインストールします。

```
# sudo yum install -y --nogpgcheck https://download2.rstudio.org/rstudio-server-rhel-1.0.44-x86_64.rpm
```

ブラウザから [http://\[サーバIPアドレス\]:8787](http://[サーバIPアドレス]:8787) へアクセスし、R Studio を起動します。

ログイン画面が表示されたら R Studio のユーザ名・パスワードを入力し、ログインの確認を行います。

The image shows the R Studio login interface. At the top left is the R Studio logo. In the center is a box titled "Sign in to RStudio". Inside this box, there are two input fields: "Username:" and "Password:". Below the password field is a checkbox labeled "Stay signed in when browser closes". Underneath the checkbox is a note: "You will automatically be signed out after 60 minutes of inactivity." At the bottom of the box is a blue button labeled "Sign In".

### 3. 4. Libxml2 のインストール

Libxml2 をインストールします（Achilles をインストールする際に必要となります）。

```
# sudo yum install libxml2-devel
```

### 3. 5. GitHub からの資産ダウンロード

GitHubから以下の資産をダウンロードします。

root特権ユーザでログインし、Broadseaの設定ファイルをGitHubより取得します。

```
# git clone https://github.com/OHDSI/Broadsea.git
```

取得したファイルは、作業ディレクトリへ移動します。

※下記の例では、「/root/dockerwk」へ取得したファイルを移動しています。

```
# cp -r Broadsea /root/dockerwk
```

データベース管理ユーザでログインし、OMOP CDM（V5） のテーブル定義をGitHubより取得します。

```
# sudo curl -L https://github.com/OHDSI/CommonDataModel/archive/refs/tags/v5.3.1.tar.gz -o /home/postgres/
```

取得したファイルは、作業ディレクトリへ移動します。

※下記の例では、作業ディレクトリ「/home/postgres」へ取得したファイルを移動しています。

```
# cp v5.3.1.tar.gz /home/postgres
```

コピーしたファイルを展開します。

```
# cd /home/postgres
# tar -zxvf v5.3.1.tar.gz
```

PostgreSQL用のテーブル定義は取得した資産の下記ディレクトリにあります。

```
CommonDataModel-5.3.1/PostgreSQL/
    OMOP CDM postgresql ddl.txt
    OMOP CDM postgresql pk indexes.txt
    OMOP CDM postgresql constraints.txt
```

上記の様にファイル名にスペースが含まれている状態のため、スペースを除去した形へリネームを行います。

```
# cd CommonDataModel-5.3.1/PostgreSQL/
# mv "OMOP CDM postgresql ddl.txt" OMOPCDMpostgresqlddl.txt
# mv "OMOP CDM postgresql pk indexes.txt" OMOPCDMpostgresqlpkindexes.txt
# mv "OMOP CDM postgresql constraints.txt" OMOPCDMpostgresqlconstraints.txt
```

## 4. データベース構築

OMOP Common Data Model のデータベース構築を行います。

### 4. 1. スキーマの作成

対象サーバに データベース管理ユーザでログインし、psqlを起動します。

```
# psql
```

OMOP CDMのテーブルを作成するスキーマ「cdmv5」と、WebAPI用スキーマ「ohdsi」を作成します。

```
postgres=# create schema cdmv5 authorization postgres;
postgres=# create schema ohdsi authorization postgres;
```

作成が完了したら「¥q」を入力し、psqlを終了します。

```
postgres=# ¥q
```

### 4. 2. OMOP Common Data Model テーブルの作成

対象サーバに データベース管理ユーザでログインし、psqlを起動します。

```
# psql
```

カレントスキーマを「cdmv5」へ変更します。

```
postgres=# set search_path to "cdmv5";
```

カレントスキーマが「cdmv5」に変更された事を確認します。

```
postgres=# select current_schema();
current_schema
-----
cdmv5
(1 行)
```



3. 4. で取得したテーブル定義を実行し、OMOP CDMのテーブルを作成します。

```
postgres=# \i /home/postgres/CommonDataModel-5.3.1/PostgreSQL/OMOPCDMpostgresddl.txt
postgres=# \i /home/postgres/CommonDataModel-5.3.1/PostgreSQL/OMOPCDMpostgresddl.txt
postgres=# \i /home/postgres/CommonDataModel-5.3.1/PostgreSQL/OMOPCDMResultspostgresddl.txt
```

テーブルが作成された事を確認します。

```
postgres=# \dt
                                     リレーションの一覧
 スキーマ |      名前      | 型   | 所有者
-----+-----+-----+-----
 cdmv5    | attribute_definition | テーブル | postgres
 cdmv5    | care_site          | テーブル | postgres
:
```

作成が完了したら「¥q」を入力し、psqlを終了します。

```
postgres=# ¥q
```

#### 4. 3. レコード投入方法のご紹介

CDMへのレコードの挿入は、タブ区切りファイル等をpsqlで取り込む事で行います。

OHDSI/ATHENA など配布されている Vocabularyファイルもタブ区切りの形式です。

FTP等によるサーバへのファイルの配置、またはGitHubからの取得などで区切りファイルを任意の場所に配置します。

ファイルを配置後、対象サーバに データベース管理ユーザでログインし、psqlを起動します。

```
# psql
```

カレントスキーマを「cdmv5」へ変更します。

```
postgres=# set search_path to "cdmv5";
```

カレントスキーマが「cdmv5」に変更された事を確認します。

```
postgres=# select current_schema();  
current_schema  
-----  
cdmv5  
(1 行)
```

ファイルを指定して、レコードの取込を行います。

```
postgres=# COPY [テーブル名] FROM  
'[パス情報含むファイル名]'  
WITH DELIMITER E'¥t' NULL AS '' ;
```

具体的な例は下記のようになります。

```
postgres=# COPY concept FROM  
'/home/postgres/CSVData/CONCEPT.csv'  
WITH DELIMITER E'¥t' NULL AS '' ;
```

CSVファイルの取り込みは頻繁に利用される方法ですので、操作を覚えておいてください。

## 5. Broadseaの構築

Broadseaコンテナの動作設定を行います。

### 5. 1. Broadseaイメージの取得

root特権ユーザでログインし、BroadseaのDockerイメージをDockerHubより取得します。

Dockerが起動している状態で実行します。

```
# docker pull ohdsi/broadsea-webtools:latest  
  
# docker pull ohdsi/broadsea-methodslibrary:latest
```

### 5. 2. 設定ファイルの編集

3. 3. で取得した設定ファイルを編集します。

Broadsea/postgresqlディレクトリにあるファイル、docker-compose.ymlをBroadseaディレクトリにコピーします。

※ 3. 3 の例と同様に、取得したファイルを「/root/dockerwk」へ移動した前提として記載しています。

```
# cd /root/dockerwk/Broadsea/postgresql  
  
# cp docker-compose.yml ../
```

コピーした docker-compose.yml を開き、内容を編集します。

```
# cd /root/dockerwk/Broadsea  
  
# vi docker-compose.yml
```

編集内容は下記の通りです。

```
version: '2'

services:

  broadsea-methods-library:
    image: ohdsi/broadsea-methodslibrary
    ports:
      - "8787:8787"
      - "6311:6311"
    environment:
      - PASSWORD=mypass

  broadsea-webtools:
    image: ohdsi/broadsea-webtools
    ports:
      - "8080:8080"
    volumes:
      - .:/tmp/drivers/:ro
      - ./config-local.js:/usr/local/tomcat/webapps/atlas/js/config-local.js:ro
    environment:
      - WEBAPI_URL=http://[ホストのIPアドレス]:8080 →①
      - env=webapi-postgresql
      - datasource_driverClassName=org.postgresql.Driver
      - datasource_url=jdbc:postgresql://[ホストのIPアドレス]:5432/[データベース名] →②
      - datasource_cdm.schema=[CDM用のスキーマ名] →③
      - datasource_ohdsi.schema=[WebAPI用のスキーマ名] →④
      - datasource_username=[データベース接続ユーザ名] →⑤
      - datasource_password=[データベース接続パスワード] →⑥
      - spring.jpa.properties.hibernate.default_schema=[WebAPI用のスキーマ名]
      - spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
      - spring.batch.repository.tableprefix=ohdsi.BATCH_
      - flyway_datasource_driverClassName=org.postgresql.Driver
      - flyway_datasource_url=jdbc:postgresql://[ホストのIPアドレス]:5432/[データベース名] →②
      - flyway_schemas=[WebAPI用のスキーマ名] →④
      - flyway_placeholders.ohdsiSchema=[WebAPI用のスキーマ名] →④
      - flyway_datasource_username=[データベース接続ユーザ名] →⑤
      - flyway_datasource_password=[データベース接続パスワード] →⑥
      - flyway_locations=classpath:db/migration/postgresql
```

- ①[ホストのIPアドレス] : Dockerが動作しているホストのIPアドレスを指定します。
- ②[データベース名] : CDMを作成したデータベース名を指定します。
- ③[CDM用のスキーマ名] : 4. 1. で作成したCDM用のスキーマ名を指定します。
- ④[WebAPI用のスキーマ名] : 4. 1. で作成したWebAPI用のスキーマ名を指定します。
- ⑤[データベース接続ユーザ名] : データベースへ接続するためのユーザ名を指定します。
- ⑥[データベース接続パスワード] : データベースへ接続するユーザのパスワードを指定します。

続いて、Broadseaディレクトリにあるファイル、config-local.jsを編集します。

```
# cd /root/dockerwk/Broadsea
# vi config-local.js
```

```
define([], function () {
    var configLocal = {};

    // clearing local storage otherwise source cache will obscure the override settings
    localStorage.clear();

    var getUrl = window.location;
    var baseUrl = getUrl.protocol + "://" + getUrl.host;

    // WebAPI
    configLocal.api = {
        name: 'OHDSI',
        // url: baseUrl + '/WebAPI/'
        url: 'http://[ホストのIPアドレス]:8080/WebAPI/' →①
    };

    configLocal.cohortComparisonResultsEnabled = false;
    configLocal.userAuthenticationEnabled = false;
    configLocal.plpResultsEnabled = false;

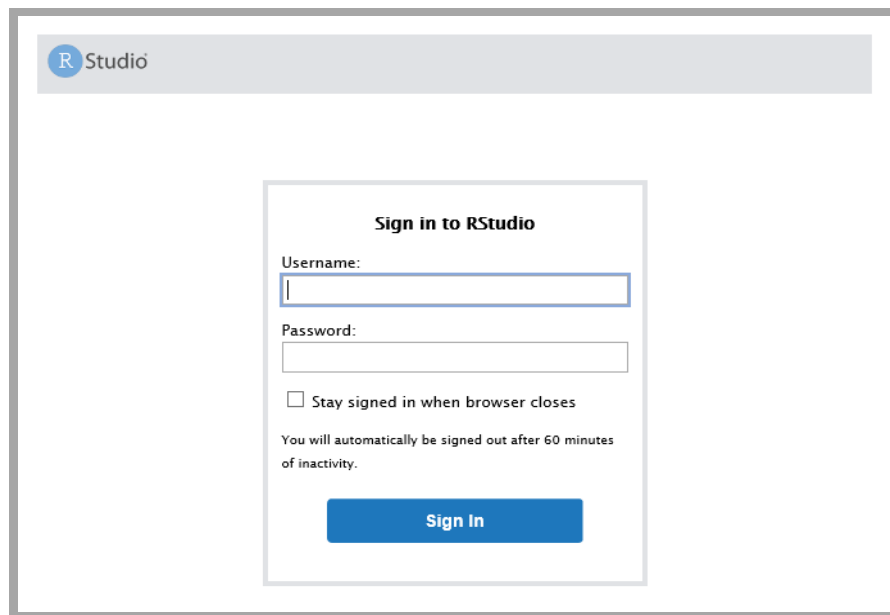
    return configLocal;
});
```

- ①[ホストのIPアドレス] : Dockerが動作しているホストのIPアドレスを指定します。

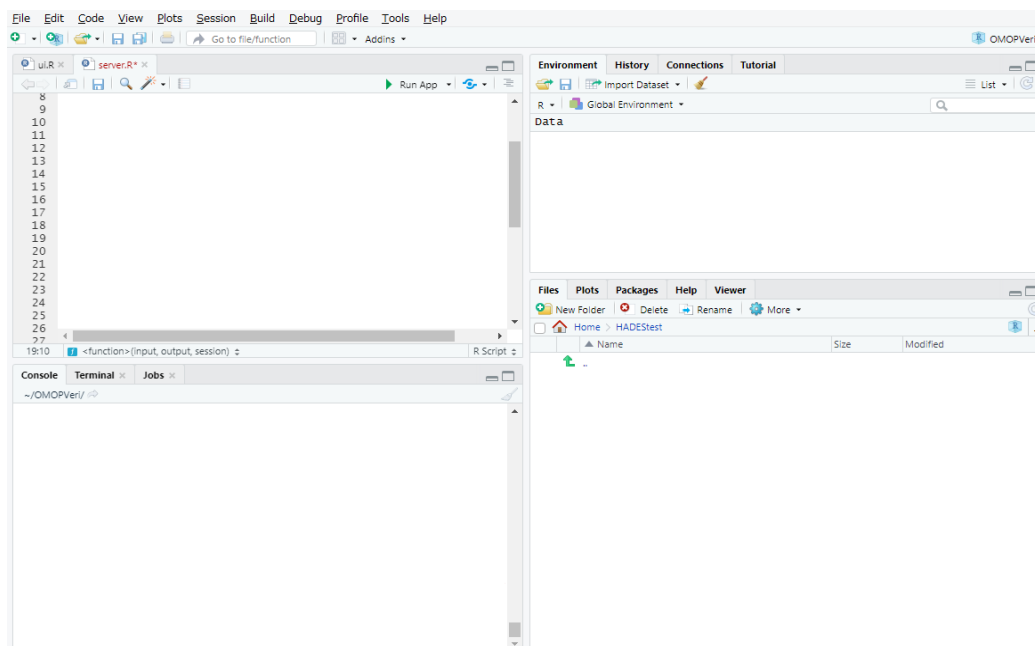
### 5. 3. Achilles / CohortMethod のインストール

ブラウザから [http://\[サーバIPアドレス\]:8787](http://[サーバIPアドレス]:8787) にアクセスし、R Studio を起動します。

ログイン画面が表示されたら R Studio のユーザ名・パスワードを入力し、ログインします。



ログイン後、画面左下のコンソールより下記コマンドを入力し、Achillesをインストールします。



```
> install.packages("remotes")
> remotes::install_github("ohdsi/Achilles")
```

同様に、R Studio から下記コマンドを入力し、CohortMethod パッケージをインストールします。

```
> remotes::install_github("OHDSI/CohortMethod")
```

※尚、CohortMethodインストール時に以下PKGも同時にインストールされます。

```
DatabaseConnector (>= 4.0.0)

Cyclops (>= 3.0.0)

FeatureExtraction (>= 3.0.0)

Andromeda (>= 0.3.0)
```

続いて、DB接続ドライバ（JDBC）をインストールします。

インストール先を環境変数「DATABASECONNECTOR\_JAR\_FOLDER」に設定して実行します。

※ディレクトリは任意となります。この例では「/home/rstudio/jdbcDrivers」としています。

```
> Sys.setenv("DATABASECONNECTOR_JAR_FOLDER" = "/home/rstudio/jdbcDrivers")
> downloadJdbcDrivers("postgresql")
```

※DBがPostgreSQL以外の場合は対応するドライバをインストールしてください。

下記でCohortMethodパッケージが正しくインストールされているか確認します。

```
> connectionDetails <- createConnectionDetails(dbms="postgresql",

server="localhost/postgres",          →①

user = "postgres",                    →②

password = " [PostgreSQLインストール時に設定したパスワード] ")

> checkCmInstallation(connectionDetails)
```

①serverには" {ホスト名（またはホストのIPアドレス）} / {DB名} "を記載します。

※host名はHADES環境とDBを同一端末で構築した場合は"localhost"と記載します。

※DB名にはpostgresインストール時にデフォルトで作成される"postgres"を記載します。

②Userには上記デフォルトで作成されるDBのユーザ名"postgres"を記載します。

上記を実行し、下記が表示されていればインストール成功です。

```
> connectionDetails <- createConnectionDetails(dbms="postgresql",
+ server="localhost/postgres",
+ user = "postgres",
+ password = " ")
> checkCmInstallation(connectionDetails)
Checking database connectivity
Connecting using PostgreSQL driver
- ok

Checking large scale regression engine
- ok

Checking support for large data objects
- ok

CohortMethod is correctly installed
Response code: 387848
> |
```

※createConnectionDetails関数を呼び出せない場合は、

下記コマンドでdatabaseConncection関数を呼び出してから再度、実行してください。

```
> library(databaseConncection)
```

#### 5. 4. 初回起動と起動後の追加設定

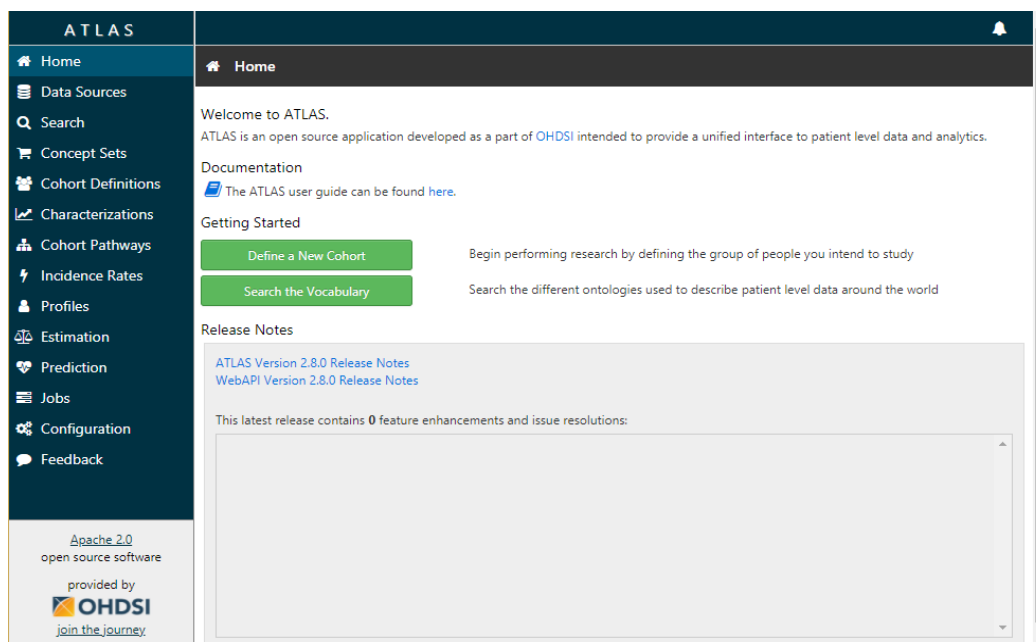
設定ファイルの編集、及びインストールが完了したら、下記コマンドを実行し、コンテナを起動します。

コマンドは、編集したdocker-compose.yml、config-local.jsがあるディレクトリで実行します。

```
# cd /root/dockerwk/Broadsea
# docker-compose up -d
Starting broadseamaster_broadsea-methods-library_1 ...
Starting broadseamaster_broadsea-methods-library_1
Starting broadseamaster_broadsea-webtools_1 ...
Starting broadseamaster_broadsea-webtools_1 ... Done
```



コンテナ起動後、ブラウザから [http://\[ホストのIPアドレス\]:8080/atlas/](http://[ホストのIPアドレス]:8080/atlas/) へアクセスします。



ATLASの画面が表示されたら、一度画面を閉じます。

画面を閉じたらroot特権ユーザでログインし、コンテナを停止します。

「docker ps -a」でコンテナの状態を表示、コンテナIDを確認し、対象のコンテナを「docker stop コンテナID」

で停止します。

```
# docker ps -a
CONTAINER_ID  IMAGE                                COMMAND                                . . .
84ed9b76a32a  ohdsi/broadsea-webtools             "/usr/bin/supervisord"              . . .
# docker stop 84ed9b76a32a
```

データベース管理ユーザでログインし、psqlを起動します。

```
# psql
```

カレントスキーマを WebAPI用スキーマ（この例では ohdsi）へ変更します。

```
postgres=# set search_path to "ohdsi";
```

変更後のカレントスキーマ確認します。

```
postgres=# select current_schema();
current_schema
-----
ohdsi
(1 行)
```

カレントスキーマ確認後にテーブル一覧を表示すると、初回起動時にいくつかのテーブルが作成されている事が分かります。

作成されたテーブルの中に sourceとsource\_daimon がある事を確認します。

```
postgres=# \dt
リレーションの一覧
スキーマ |          名前          | 型   | 所有者
-----+-----+-----+-----
ohdsi    | analysis_generation_info | テーブル | postgres
ohdsi    | batch_job_execution      | テーブル | postgres
ohdsi    | batch_job_execution_context | テーブル | postgres
          :
ohdsi    | source                   | テーブル | postgres
ohdsi    | source_daimon            | テーブル | postgres
ohdsi    | user_import_job          | テーブル | postgres
ohdsi    | user_import_job_weekdays | テーブル | postgres
```

次のSQLを実行し、WebAPIの設定情報を登録します。

```
postgres=# INSERT INTO ohdsi.source
(source_id, source_name, source_key, source_connection, source_dialect)
VALUES (1, 'OHDSI CDM V5 Database', 'OHDSI-CDMV5',
'jdbc:postgresql://[ホストのIPアドレス]:5432/[データベース名]?user=[ユーザー名]&password=[パスワード]',
'postgresql');
```

※CDM ドメインの設定

```
postgres=# INSERT INTO ohdsi.source_daimon
(source_daimon_id, source_id, daimon_type, table_qualifier, priority)
VALUES (1, 1, 0, '[CDMスキーマ名]', 2);
```

#### ※VOCABULARY ドメインの設定

```
postgres=# INSERT INTO ohdsi.source_daimon
(source_daimon_id, source_id, daimon_type, table_qualifier, priority)
VALUES (2, 1, 1, '[CDMスキーマ名]', 2);
```

#### ※RESULTS ドメインの設定

```
postgres=# INSERT INTO ohdsi.source_daimon
(source_daimon_id, source_id, daimon_type, table_qualifier, priority)
VALUES (3, 1, 2, '[WebAPIスキーマ名]', 2);
```

#### ※EVIDENCE ドメインの設定

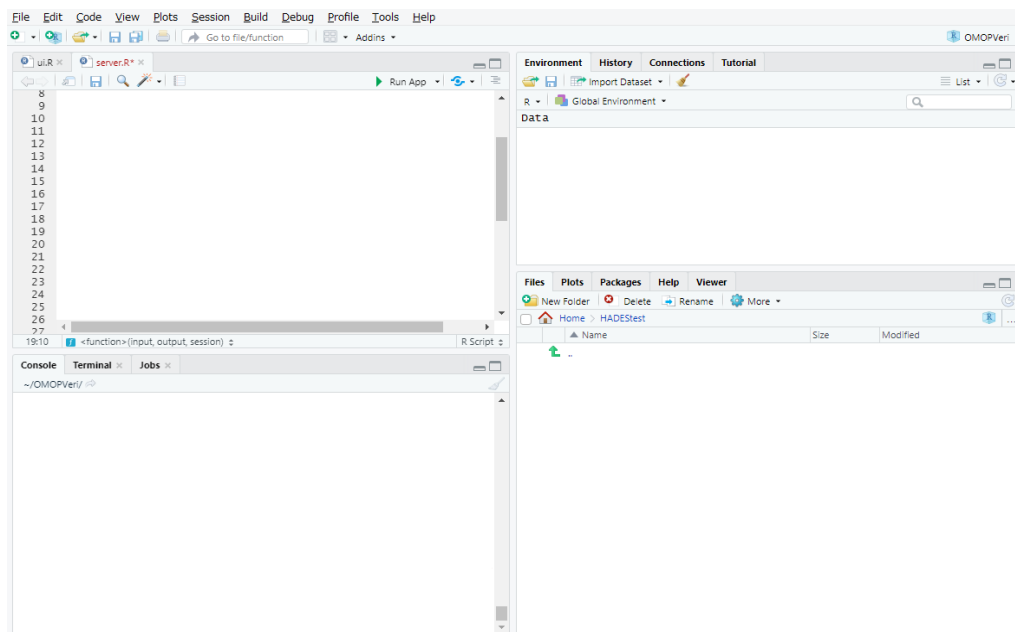
```
postgres=# INSERT INTO ohdsi.source_daimon
( source_daimon_id, source_id, daimon_type, table_qualifier, priority)
VALUES (4, 1, 3, '[WebAPIスキーマ名]', 2);
```

SQL実行後、root特権ユーザにて、コンテナの再起動を行います。

```
# cd /root/dockerwk/Broadsea
# docker-compose up -d
Starting broadseamaster_broadsea-methods-library_1 ...
Starting broadseamaster_broadsea-methods-library_1
Starting broadseamaster_broadsea-webtools_1 ...
Starting broadseamaster_broadsea-webtools_1 ... Done
```

## 5. 5. Achillesの実行

ブラウザから [http://\[サーバIPアドレス\]:8787](http://[サーバIPアドレス]:8787) へアクセスし、R Studio を起動し、ログインします。



ログイン後、画面左下のConsole部分に下記のコマンドを入力し、Achilles を起動します。

```
> library(Achilles)
> Sys.setenv("DATABASECONNECTOR_JAR_FOLDER"="/home/rstudio/jdbcDrivers")
> connectionDetails <- createConnectionDetails(
  dbms="postgresql",
  server="[サーバIPアドレス]/[データベース名]",      →①
  user="[データベース接続ユーザ名]",                →②
  password="[データベース接続パスワード]",          →③
  port="5432")
> achilles(connectionDetails,
  cdmDatabaseSchema = "[CDMのスキーマ]",              →④
  resultsDatabaseSchema="[WebAPIのスキーマ]",          →⑤
  vocabDatabaseSchema = "[CDMのスキーマ]",             →⑥
  numThreads = 1,
  sourceName = "[テーブル設定内容]",                  →⑦
  cdmVersion = "5.3.1",
  runHeel = FALSE,
  runCostAnalysis = FALSE)
```

①[ホストのIPアドレス]：Dockerが動作しているホストのIPアドレスを指定します。

[データベース名]：CDMを作成したデータベース名を指定します。

②[データベース接続ユーザ名]：データベースへ接続するためのユーザ名を指定します。

③[データベース接続パスワード]：データベースへ接続するユーザのパスワードを指定します。

④[CDM用のスキーマ名]：5. 4. でCDMドメインに指定した内容を設定します。

⑤[WebAPI用のスキーマ名]：5. 4. でRESULTドメインに指定した内容を設定します。

⑥[CDM用のスキーマ名]：5. 4. でVOCABULARYドメインに指定した内容を設定します。

⑦[テーブル設定内容]：5. 4. でsourceテーブルに登録した「source\_name」の内容を設定します。

以上で環境構築の手順は完了となります。

## 6. ATLAS画面表示

「docker ps -a」コマンドを実行し、コンテナが起動されている事を確認します。

```
# docker ps -a
```

CONTAINER_ID	IMAGE	COMMAND	...	STATUS
84ed9b76a32a	ohdsi/broadsea-webtools	"/usr/bin/supervisord"	...	UP 9hours

コンテナが起動されている事を確認し、ブラウザから [http://\[ホストのIPアドレス\]:8080/atlas/](http://[ホストのIPアドレス]:8080/atlas/) へアクセスして下さい。

