**OMOP Transformation on GCP**

Project Log

Technical processes, issues and learning

Last updated 06 Sep 2022

Contents

[1 Introduction 1](#_Toc152340092)

[2 GCP Install options: 1](#_Toc152340093)

[2.1 Odysseus image (Linux VM) 1](#_Toc152340094)

[2.2 Windows VM 1](#_Toc152340095)

[2.3 Linux VM 1](#_Toc152340096)

[3 Connection string 1](#_Toc152340097)

[3.1 Private key 1](#_Toc152340098)

[4 Walkthrough - using Odysseus image 3](#_Toc152340099)

[4.1 Introduction 3](#_Toc152340100)

[4.2 Deploy the image 4](#_Toc152340101)

[4.3 Copy the service account private key 4](#_Toc152340102)

[4.4 Update the webapi database to use the omop database 4](#_Toc152340103)

[4.5 Update to the latest version of R 5](#_Toc152340104)

[4.6 Install Achilles 5](#_Toc152340105)

[4.7 Install CDM Inspection 5](#_Toc152340106)

[4.8 Install Hades 5](#_Toc152340107)

[4.9 Install DataQualityDashboard 6](#_Toc152340108)

[4.10 Generate results schema 6](#_Toc152340109)

[4.11 Run Achilles 6](#_Toc152340110)

[4.12 Run DataQualityDashboard 6](#_Toc152340111)

[4.13 Run CDMInspectionReport 7](#_Toc152340112)

[5 Resources 8](#_Toc152340113)

# Introduction

The Connected Bradford source data is located on a Google Cloud Platform project. This isa guide to Imosphere’s process to install the necessary EHDEN prerequisites and build and deploy an ETL.

This space captures the learning, processes and issues encountered during the project related to working on GCP which is a less common approach to deploying a new target OMOP database, with less community knowledge and learning available.

# GCP Install options:

## Odysseus image (Linux VM)

**Advantages**: Prebuilt, most required packages included.  
**Disadvantages**: Sample data included; additional changes required to get it working properly.

## Windows VM

**Advantages**: Known installation and setup.  
**Disadvantages**: Additional costs to host, cannot use the standard installer due to BigQuery. No testing carried out.

## Linux VM

**Advantages**: Allows custom installation.  
**Disadvantages**: Additional time to setup. No testing carried out.

# Connection string

**ProjectId** - Google project id.

**DefaultDataset** - Default dataset to use in BigQuery. Required for Achilles, not required for WebAPI.

**OAuthServiceAcctEmail** - The email address assoicated with the service account to be used in connecting to BigQuery.

**LocationOfServiceAccountPrivateKeyFile** - A private key assoicated with the service account. Needs to be accessible to application using the connection string.

jdbc:bigquery://https://www.googleapis.com/bigquery/v2:443;ProjectId=<ProjectId>;DefaultDataset=<omop dataset>;OAuthType=0;OAuthServiceAcctEmail=<GoogleServiceAccountEmail>;OAuthPvtKeyPath=<LocationOfServiceAccountPrivateKeyFile>;Timeout=1000;AllowLargeResults=0;EnableHighThroughputAPI=1;UseQueryCache=1;LogLevel=0;

## Private key

A private key is required to connect to BigQuery with a service account that has the required permissions. If this isn’t provided, one will need to be created.

The private key file for the service account can be created on Google Cloud → APIs and services → Credentials

Graphical user interface, text, application, email

Description automatically generated

Select the service account



Select the keys tab.

Graphical user interface, text, application, email

Description automatically generated

Create new key

Graphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

# Walkthrough - using Odysseus image

Odysseus image - <https://console.cloud.google.com/marketplace/product/odysseusinc-public/ohdsi-atlas>

## Introduction

This walkthrough assumes that you have an OMOP database that has been populated in BigQuery. This walkthrough also doesn’t address public access to the data, or removing the sample data provided in the image. The default user name or passwords should be changed.

## Deploy the image

Make sure the correct google project is selected, and the open the image link and click Deploy.

Populate the details for the deployment. Review requirements, accept licences and click Deploy.

Graphical user interface, text, application, email

Description automatically generated

Make sure the correct google project is selected, and the open the the image link and click Deploy.

## Copy the service account private key

Connect to the deployed VM via SSH. The run the following commands.

cd /opt/workdir

sudo mkdir key

cd key

Click upload and browse to the key file and then upload. Run the following commands.

sudo cp /home/<user>/key.json /opt/workdir/key

## Update the webapi database to use the omop database

Update the web api source table. Make sure to specifiy the the project and account email, and datasets. When prompted, enter the password, by deafult this is @tlaS.

psql -U ohdsi -h localhost -d webapi -c "UPDATE webapi.source SET source\_name = 'atlasbig', source\_key = 'atlasbig', source\_connection = 'jdbc:bigquery://https://www.googleapis.com/bigquery/v2:443;ProjectId=<ProjectId>;OAuthType=0;OAuthServiceAcctEmail=<LocationOfServiceAccountPrivateKeyFile>;OAuthPvtKeyPath=/opt/workdir/key/key.json;Timeout=1000;AllowLargeResults=0;EnableHighThroughputAPI=1;UseQueryCache=1;LogLevel=0;',source\_dialect='bigquery',username='',password=''"

psql -U ohdsi -h localhost -d webapi -c "UPDATE webapi.source\_daimon SET table\_qualifier = '<omop\_dataset>' WHERE source\_daimon\_id = 1"

psql -U ohdsi -h localhost -d webapi -c "UPDATE webapi.source\_daimon SET table\_qualifier = '<omop\_dataset>' WHERE source\_daimon\_id = 2"

psql -U ohdsi -h localhost -d webapi -c "UPDATE webapi.source\_daimon SET table\_qualifier = '<omop\_results\_dataset>' WHERE source\_daimon\_id = 3"

psql -U ohdsi -h localhost -d webapi -c "UPDATE webapi.source\_daimon SET table\_qualifier = '<omop\_temp>' WHERE source\_daimon\_id = 4"

## Update to the latest version of R

When prompted, say Y

/home/<user>

sudo apt update -qq

sudo apt install --no-install-recommends software-properties-common dirmngr

wget -qO- https://cloud.r-project.org/bin/linux/ubuntu/marutter\_pubkey.asc | sudo tee -a /etc/apt/trusted.gpg.d/cran\_ubuntu\_key.asc

sudo add-apt-repository "deb https://cloud.r-project.org/bin/linux/ubuntu $(lsb\_release -cs)-cran40/"

sudo apt install --no-install-recommends r-base

sudo apt-get install r-base-dev

sudo add-apt-repository ppa:c2d4u.team/c2d4u4.0+

sudo apt-get install -y r-cran-rjava

sudo apt-get install xml2

sudo apt-get install libxml2-dev

sudo apt-get install libssl-dev

sudo apt-get -y install libcurl4-gnutls-dev

sudo apt-get install libfontconfig1-dev

sudo apt-get install libcairo2-dev

sudo apt install cmake

## Install Achilles

Open a R prompt using sudo R

install.packages("devtools")

install.packages("DatabaseConnector")

library(devtools)

install\_github("jdposada/BQJdbcConnectionStringR")

install\_github("OHDSI/Achilles")

## Install CDM Inspection

Open a R prompt using sudo R

remotes::install\_github("OHDSI/ROhdsiWebApi")

remotes::install\_github("EHDEN/CdmInspection")

## Install Hades

Open a R prompt using sudo R

install.packages("remotes")

library(remotes)

Sys.setenv(GITHUB\_PAT="ghp\_rMAUn15971MC45YY8Yx6id9UxsfOuE2ARSjW");

install\_github("ohdsi/Hades", upgrade = "always")

## Install DataQualityDashboard

Open a R prompt using sudo R

install.packages('DT')

devtools::install\_github("OHDSI/DataQualityDashboard")

## Generate results schema

Go to:

http://<vmipaddress>/WebAPI/ddl/results?dialect=bigquery&schema=<omop\_results\_dataset>&vocabSchema=<omop\_dataset>&tempSchema=<omop\_temp\_dataset>&initConceptHierarchy=true

Go to bigquery and run the results of the above url agasint the omop results dataset.

## Run Achilles

Open a R prompt using sudo R. Note some analysis ids don’t currently work.

bqDriverPath <- "/opt/workdir/bigquery"

connectionString <- BQJdbcConnectionStringR::createBQConnectionString(projectId = "<ProjectId>", defaultDataset = "<DefaultDataset>", authType = 0, accountEmail = "<OAuthServiceAcctEmail>", jsonCredentialsPath = "<LocationOfServiceAccountPrivateKeyFile>")

connectionDetails <- DatabaseConnector::createConnectionDetails(dbms="bigquery",connectionString=connectionString,user="",password='',pathToDriver = bqDriverPath)

databaseDescription <- "<omop>"

cdmDatabaseSchema <- "<omop\_dataset>"

resultsDatabaseSchema <- "<omop\_results\_dataset>"

vocabDatabaseSchema <- "<omop\_dataset>"

library(Achilles)

connection <- connect(connectionDetails)

achilles(connectionDetails, cdmDatabaseSchema = cdmDatabaseSchema, resultsDatabaseSchema=resultsDatabaseSchema, vocabDatabaseSchema = vocabDatabaseSchema, numThreads = 1, sourceName = databaseDescription, cdmVersion = "5.3.1", optimizeAtlasCache = TRUE, outputFolder = "OutputAchilles", analysisIds = c(0,1,2,3,4,5,7,8,9,10,11,12,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,200,201,202,203,204,206,207,209,210,211,212,213,220,221,225,226,230,300,301,303,325,400,401,402,403,404,405,409,410,411,412,413,414,415,416,420,425,430,500,501,502,504,505,506,509,510,511,512,513,514,515,525,530,600,601,602,603,604,605,606,609,610,612,613,620,625,630,691,700,701,702,703,704,705,706,709,710,711,712,713,715,716,717,720,725,730,791,800,801,802,803,804,805,806,807,809,810,812,813,814,815,820,822,823,825,826,827,830,891,900,901,902,903,904,906,907,908,910,911,920,930,1000,1001,1002,1004,1006,1007,1008,1010,1011,1020,1030,1100,1101,1102,1103,1200,1201,1202,1300,1301,1302,1303,1304,1306,1307,1309,1310,1311,1312,1313,1320,1321,1325,1326,1330,1406,1407,1408,1409,1410,1411,1412,1413,1414,1415,1425,1800,1801,1802,1803,1804,1805,1806,1807,1809,1810,1811,1812,1813,1814,1815,1817,1818,1819,1820,1821,1822,1823,1825,1826,1827,1830,1891,2000,2001,2002,2003,2100,2101,2102,2104,2105,2106,2110,2125,2130,2191,2200,2201))

## Run DataQualityDashboard

Open a R prompt using sudo R

bqDriverPath <- "/opt/workdir/bigquery"

connectionString <- BQJdbcConnectionStringR::createBQConnectionString(projectId = "<ProjectId>", defaultDataset = "<DefaultDataset>", authType = 0, accountEmail = "<OAuthServiceAcctEmail>", jsonCredentialsPath = "<LocationOfServiceAccountPrivateKeyFile>")

connectionDetails <- DatabaseConnector::createConnectionDetails(dbms="bigquery",connectionString=connectionString,user="",password='',pathToDriver = bqDriverPath)

outputFile <- "Data Quality Dashboard Results.json"

outputFolder <- "OutputDQD"

checkNames <- c()

cdmSourceName = <- "<omop>"

cdmDatabaseSchema <- "<omop\_dataset>"

resultsDatabaseSchema <- "<omop\_results\_dataset>"

vocabDatabaseSchema <- "<omop\_dataset>"

DataQualityDashboard::executeDqChecks(connectionDetails = connectionDetails, cdmDatabaseSchema = cdmDatabaseSchema, resultsDatabaseSchema = resultsDatabaseSchema, cdmSourceName = cdmSourceName, numThreads = 1, sqlOnly = FALSE, outputFile = outputFile, outputFolder = outputFolder, verboseMode = TRUE, writeToTable = TRUE, checkLevels = c("TABLE", "FIELD", "CONCEPT"), tablesToExclude = c("COST", "SPECIMEN", "DOSE\_ERA"), checkNames = checkNames)

## Run CDMInspectionReport

Open a R prompt using sudo R. Note: This isn’t currently working. There seems to issues with the generated sql.

library(CdmInspection)

fftempdir <- "<fftemp>"

options(fftempdir = fftempdir

bqDriverPath <- "/opt/workdir/bigquery"

connectionString <- BQJdbcConnectionStringR::createBQConnectionString(projectId = "<ProjectId>", defaultDataset = "<DefaultDataset>", authType = 0, accountEmail = "<OAuthServiceAcctEmail>", jsonCredentialsPath = "<LocationOfServiceAccountPrivateKeyFile>")

connectionDetails <- DatabaseConnector::createConnectionDetails(dbms="bigquery",connectionString=connectionString,user="",password='',pathToDriver = bqDriverPath)

authors <- "<Authors>"

databaseId <- "<omop>"

databaseName <- "<omop>"

databaseDescription <- "<omop>"

oracleTempSchema <- NULL

outputFolder <- file.path(getwd(), "OutputInspection", databaseId)

cdmSourceName = <- "<omop>"

cdmDatabaseSchema <- "<omop\_dataset>"

resultsDatabaseSchema <- "<omop\_results\_dataset>"

vocabDatabaseSchema <- "<omop\_dataset>"

baseUrl <- "http://<vmipaddress>/WebAPI"

smallCellCount <- 5

verboseMode <- TRUE

results<-cdmInspection(connectionDetails,

cdmDatabaseSchema = cdmDatabaseSchema,

resultsDatabaseSchema = resultsDatabaseSchema,

vocabDatabaseSchema = vocabDatabaseSchema,

oracleTempSchema = oracleTempSchema,

databaseName = databaseName,

runVocabularyChecks = TRUE,

runDataTablesChecks = TRUE,

runPerformanceChecks = TRUE,

runWebAPIChecks = TRUE,

smallCellCount = smallCellCount,

baseUrl = baseUrl,

sqlOnly = FALSE,

outputFolder = outputFolder,

verboseMode = verboseMode)

generateResultsDocument(results,outputFolder, authors=authors, databaseDescription = databaseDescription, databaseName = databaseName, databaseId = databaseId, smallCellCount = smallCellCount)

# Resources

Odysseus image - <https://console.cloud.google.com/marketplace/product/odysseusinc-public/ohdsi-atlas>

Odysseus image documentation - <https://docs.odysseusinc.com/gcp/OHDSI_ATLAS_v2.7.8.pdf>

SynPuf BigQuery - <https://console.cloud.google.com/marketplace/product/hhs/synpuf?filter=solution-type:dataset&q=omop&id=c77e3443-4391-4aef-9e29-070e4301cba3&project=great-firewall-290804>