

Hands on with FHIR Data Pipes for enabling scalable FHIR Analytics

Bashir Sadjad



HL7 FHIR DevDays 2023 | Hybrid Edition, Amsterdam | June 6–9, 2023 | @HL7 | @FirelyTeam | #fhirdevdays | www.devdays.com

ORGANIZED BY

firely

HL7
International

Who am I?

- Bashir Sadjad
- Software Engineer, Google (13 yrs)
 - Healthcare data (mostly open-source)
 - Data-engineering and ML models for Display Ads
- Prior to Google: Rational Drug Design



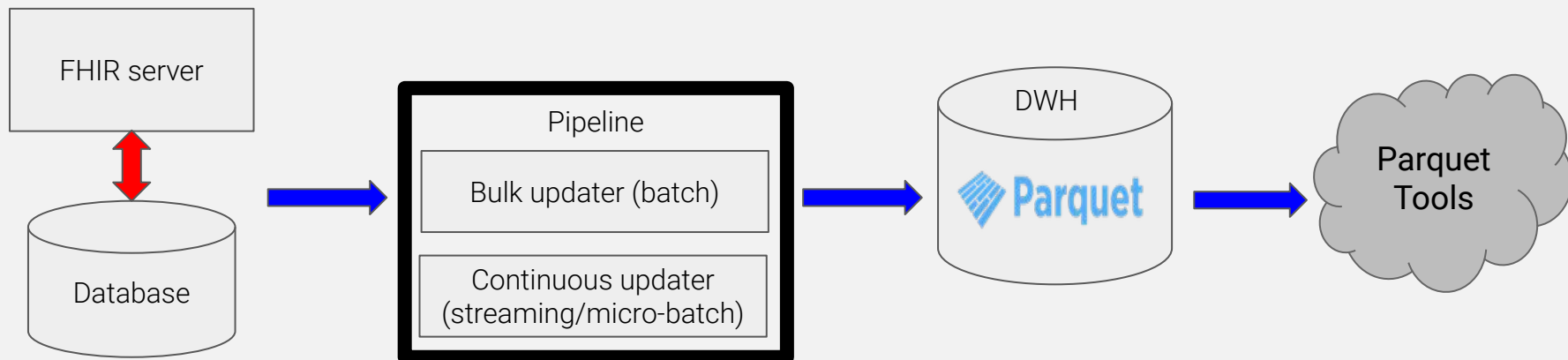
Learning Objectives

Same objectives as the morning tutorial talk with a hands-on focus:

- Doing “*Analytics*” on FHIR data
 - How to do it efficiently
 - How to make it *horizontally scalable*
- Transforming FHIR resources to Analytics friendly formats
 - The usual dilemma of analytical vs transaction processing
- Querying transformed data

The big picture

- Transform FHIR resources into Parquet files
 - Initial bulk transform
 - Continuous updates
- Query using Parquet-aware query engines



Prerequisites

- Attending the morning talk: “Transforming FHIR Data for horizontally scalable analytics”
- Access to a machine with [docker](#), [docker compose](#), git, and python
 - Depending on docker version, compose might be a sub-command not a separate tool; in that case replace `docker-compose` with `docker compose`
 - A variation of Unix/Linux is preferred but not required.
 - Java and [Maven](#) are needed if you want to build from source (not needed if you are okay using docker images).
 - [optional] Install [DBeaver](#) free community edition as the SQL console.
- [optional] Having a look at [this Wiki page](#)

The main doc and code

- We will mostly follow [this Wiki page](#) (summary below).
 - Set up everything on a single machine
 - Showing options how to run on a cluster
- Clone the repository: <https://github.com/google/fhir-data-pipes>
 - HTTPS (each shell command is indicated with a \$ prefix):

```
$ git clone https://github.com/google/fhir-data-pipes.git
```
 - SSH:

```
$ git clone git@github.com:google/fhir-data-pipes
```
- After cloning, make sure `docker/dwh` is world readable:
 - ```
$ chmod a+rx docker/dwh
```
  - This is where your generated Parquet files will be stored.

## Set up a local FHIR server

- Go to the docker directory and run the following docker compose:
  - `$ cd docker`
  - `$ docker network create cloudbuild`
  - `$ docker-compose -f hapi-compose.yml up`
  - This brings up a HAPI FHIR server on port 8091 backed by a PostgreSQL DB.
  - Add ``sudo`` if you have not [made your docker sudoless](#).
  - To test the FHIR server, from a separate terminal, try:

```
$ curl -H "Content-Type: application/json; charset=utf-8" \
 'http://localhost:8091/fhir/Patient' -v
```

- This should return with a HTTP status code 200 and a FHIR Bundle containing zero resources.

## Loading synthetic data

- There is a generator tool for synthetic HIV data at [synthea-hiv](#).
- We use the already generated data in [synthea-hiv/sample\\_data](#).
- To upload this data to our FHIR server we need Python:
  - [optional] Create a virtualenv to protect the system wide Python installation:
 

```
$ virtualenv -p python3.10 venv; source ./venv/bin/activate
```
  - Upload the sample data; from the repository root<sup>1</sup>:
 

```
$ pip install -r ./synthea-hiv/uploader/requirements.txt
$ python3 synthea-hiv/uploader/main.py HAPI \
 http://localhost:8091/fhir --input_dir \
 ./synthea-hiv/sample_data/ --cores 4
```
  - Check that there are now 79 Patient resources on the FHIR server:
    - ```
curl -H "Content-Type: application/json; charset=utf-8" \
  'http://localhost:8091/fhir/Patient?_summary=count&_total=accurate'
```

1) Depending on your Python installation, you may need to install "pip" as well, e.g.: `apt install python3-pip` on a Debian based system.

Running the pipelines

- We are now at the [Run the Single Machine configuration](#) stage of the Wiki page:
 - Follow the instructions to run the controller interface and Spark Thrift-server; we use a slightly different version to skip the “build” stage:

```
$ docker-compose -f compose-controller-spark-sql-released.yaml up
```
 - You can check controller’s status at: <http://localhost:8090/>
 - The above setup, depends on PIPELINE_CONFIG and DWH_ROOT config environment variables (defaults [here](#)).
 - Pipelines write into DWH_ROOT (default ./dwh) and Spark queries read from it.

FHIR Pipelines Control Panel

[Help](#)

Run pipelines

Incremental pipeline is scheduled to run at **2023-06-06T00:00**

Run Incremental Pipeline

This fetches the resources since last run and merges them into the data-warehouse.

[Run Incremental](#)

Run Full Pipeline

Fetch all of the resources and create a new data-warehouse snapshot

[Run Full](#)

List of DWH snapshots

Latest: /dwh/controller_DWH_TIMESTAMP_2023_06_03T00_00_26_958787083Z/

[Create Resource Tables](#)

Configuration Settings

[Main configuration parameters](#)

Running the pipelines

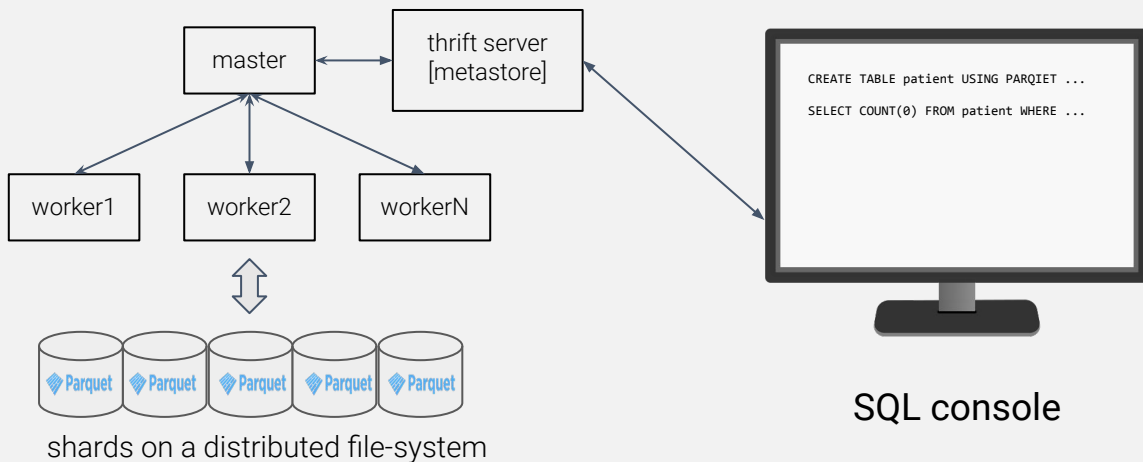
- We are now at the [Run the Single Machine configuration](#) stage of the Wiki page:
 - Follow the instructions to run the controller interface and Spark Thrift-server:
`$ docker-compose -f compose-controller-spark-sql-single.yaml up`
 - From [localhost:8090](#) do a “Run Full” (we skip the “incremental” part today).
 - Note what the controller interface does is similar to running the Beam pipeline directly, e.g., (with caveats re. `FLINK_CONF_DIR` and other defaults):

```
$ java -jar ./pipelines/batch/target/batch-bundled-0.1.0-SNAPSHOT.jar  
--outputParquetPath=./PATH --runner=FlinkRunner  
--fhirDatabaseConfigPath=utils/hapi-postgres-config_local.json  
--jdbcModeHapi --fhirServerUrl=http://localhost:8091/fhir
```

- To run on a Dataflow cluster, we just need to change the `--runner` value and add a few other Dataflow specific command line options.

How to query: the engine

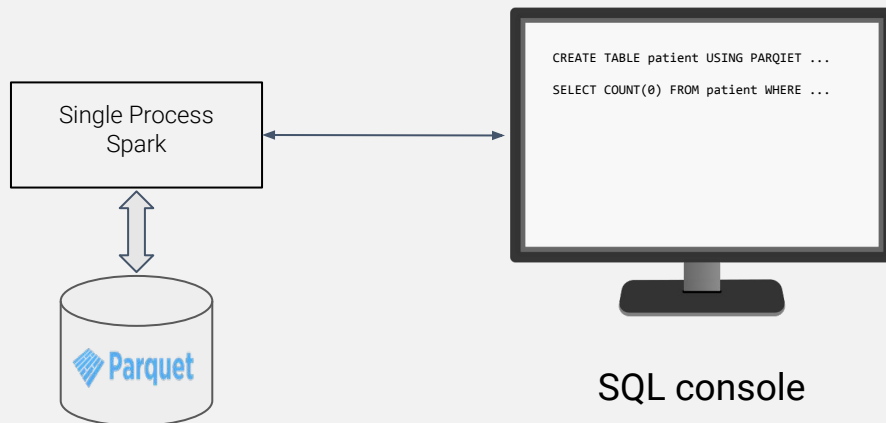
- Any query engine that understands Parquet (there are many).
 - A subset of these can scale horizontally.
 - Picked Spark as an example ([sample setup](#))



How to query: the engine

- For this codelab:
 - A very simple local spark with just one process using [this setup](#).
 - This is what we have already set up by running:

```
$ docker-compose -f compose-controller-spark-sql-single.yaml up
```



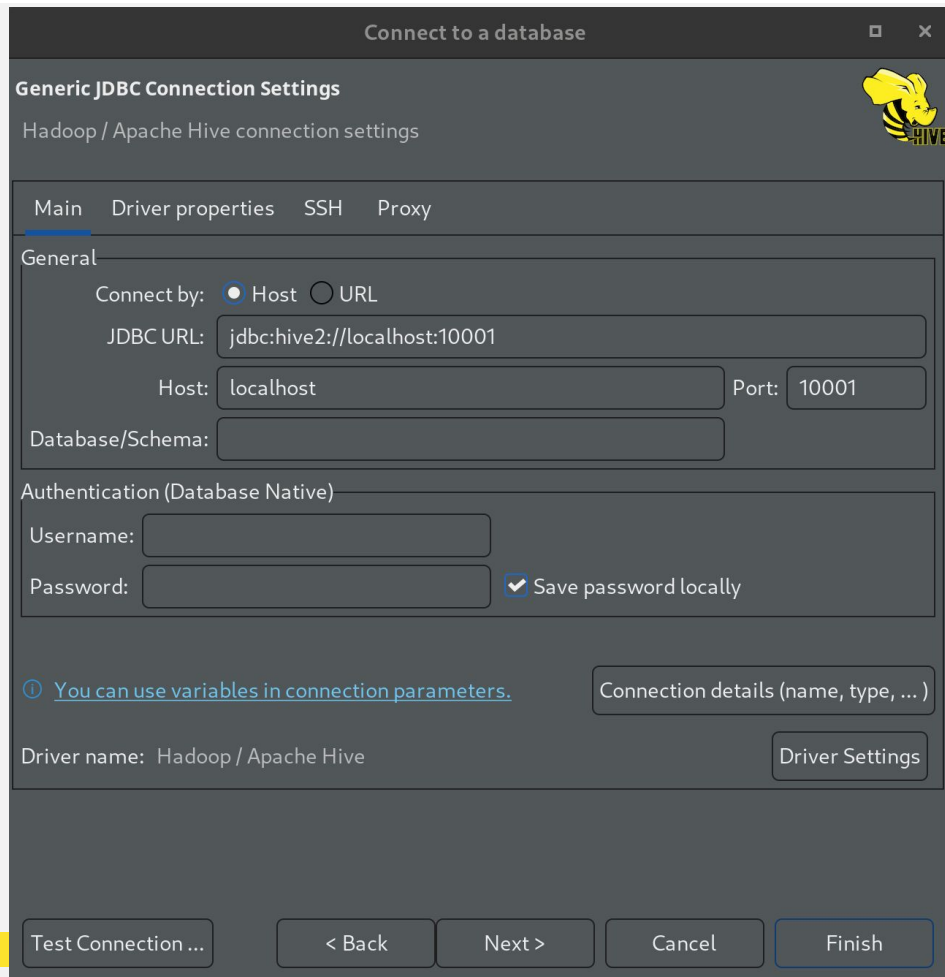
SQL queries

- You can use any tool that can connect to a Spark Thrift-server (or Hive Metastore).
 - [DBeaver](#) is one example with a free community edition.
 - Alternatively you can use the built-in beeline tool:

```
$ docker exec -it spark-thriftserver bin/beeline -u \  
jdbc:hive2://localhost:10000
```
- Follow the example queries in [this section of the Wiki page](#); but first:
 - `SELECT COUNT(0) FROM patient;`
 - `SELECT COUNT(0) FROM observation;`
- Exact same SQL queries can be submitted to a Spark cluster.
 - You can set up one quickly and easily on [Dataproc](#).

DBeaver connection

- Use the “Hive” option
- Port: 10001
- Download the required drivers at prompt



The screenshot shows the 'Connect to a database' dialog box in DBeaver. The title bar reads 'Connect to a database'. Below the title bar, it says 'Generic JDBC Connection Settings' and 'Hadoop / Apache Hive connection settings'. There is a small yellow bee icon with the word 'HIVE' below it in the top right corner. The dialog has four tabs: 'Main', 'Driver properties', 'SSH', and 'Proxy'. The 'Main' tab is selected. Under the 'General' section, there are two radio buttons for 'Connect by: Host' (selected) and 'URL'. Below that is a 'JDBC URL' field containing 'jdbc:hive2://localhost:10001'. There are also separate fields for 'Host' (localhost) and 'Port' (10001). A 'Database/Schema' field is empty. Under the 'Authentication (Database Native)' section, there are fields for 'Username' and 'Password', and a checked checkbox for 'Save password locally'. At the bottom, there is a link that says 'You can use variables in connection parameters.' and a button for 'Connection details (name, type, ...)'. The 'Driver name' is set to 'Hadoop / Apache Hive' with a 'Driver Settings' button next to it. At the very bottom of the dialog, there are four buttons: 'Test Connection ...', '< Back', 'Next >', and 'Finish'.

More to do

- If you have more time, try FHIR-views on Spark through [this notebook](#).
 - We have already done the Prerequisites section of that notebook.
- You can follow instructions on [this Wiki page](#) for integrating a dashboard tool.