Setting credentials

Contents

1 Introduction

In this vignette, we show you how to set up credentials to use within a `Ulysses` directory and load them automatically into a `config.yml` file; a file used to quickly and securely access your credentials throughout the development of a study. The `config.yml` file is automatically ignored within the project directory, ensuring that it is never loaded into the git repository. Important credentials include:

- Connection Information
  - dbms: the database management system. Currently accept snowflake, postgres, redshift, sql server and oracle.
  - user: the user name for your connection
  - password: the password for your connection
  - connectionString: a jdbc connection to interface with the database hosting the OMOP CDM
- Database Schemas
  - cdmDatabaseSchema: the location of the OMOP CDM
  - workDatabasSchema: a read/write schema to save the cohort table and other analysis tables
  - tempEmulationSchema: a temp schema used in snowflake and oracle to process temp tables

Each site has their own OMOP database and credentials; they should be stored securely. Likely you have the same credentials for every study, so only you want to load them once across all your studies; updating them only when required by your IT setup at your institution. In this vignette we first explain how to set-up a credentials store that Ulysses can automatically access. Second we explain how Ulysses will build the `config.yml` file throughout the study.

This vignette shows you how to set up the `config.yml` file automatically.

```r
library(Ulysses)
```

2 Setup

There are two ways you can set up your credentials. Option 1 is to build a csv file of your credential information per database. When building the `config.yml` file it will ingest all preset credentials, orient them for the study. Option 2 is to use the keyring api to store all your credentials. A templated `config.yml` will wrap each credential with a call to keyring to grab the credentials.

2.1 Option 1: Using credentials table (`shhh.csv`)

Our first option in setting up credentials is to setup a credentials table as a csv file. This reference file is accessed when creating a `config.yml` for a project. To begin building the credentials table we use the codeblock below (shown as an example):

```
initCredentialsTable(
  db_id = "omop_dat",
  db_full_name = "My OMOP Database",
  dbms = "redshift",
  user = "mike_jones",
  password = "who?",
  connection_string = "jdbc:redshift://281.330.8004?remaining_string",
  cdm_database_schema = "cdm",
  work_database_schema = "scratch"
)
```

This table of credentials will be saved as a file called `shhh.csv` to your home directory, by default. To check what your home directory is run: `fs::path_home()`. To check your stored credentials you can run the command `openCredentials()`. Future releases will allow you to edit the credential table in the R studio viewer.

2.2 Option 2: Using Keyring

The second option is to store credentials for each database connection in the keyring API. Ulysses provides a helper function to assist with this setup.

```
initCredentialsKeyring(
  db_id = "omop_dat"
)
```

When running this block of code, a dialog box will appear asking you to input required credentials. Type them into the dialog box one by one. At the end the function will return blurred output of your credentials to review.

3 Initializing the `config.yml`

Once you have setup your database credentials outside the Ulysses project, you can now utilize them at the start of a new study.

3.1 On build

This is a future option. When using the `newOhdsiStudy` you can state that you want to initialize the `config.yml` during the project build. This however is not yet available.

3.2 After build

After you have run `newOhdsiStudy` and navigated into the new project, you should initialize the `config.yml` file. This is done using the code-block below.

```
initConfigUsingTable() # initialize the config using the shhh.csv file

initConfigUsingKeyring() # initialize the config using keyring
```

A new `config.yml` file will open after running these functions. If initializing from the table, the credentials are read in "as-is". If initializing from keyring, the keyring R package is wrapped on every line of the config prefixed by `!expr`. This notifies R to evaluate the item in the file. To check that the keyring works, you can run the `keyring::key_get` command in the config line in the console. If it evaluates then it will work on the config call, if not an error will occur when evaluating.

4 `config.yml` in action

The config file allows you to quickly and securely shuffle between connection to the OMOP CDM for a study. Ulysses uses the config package to pull the correct configuration from this file to connect to the database.

Here is an example:

```
# set connection Block
# <<<
configBlock <- "[Add config block]"
# >>>

# provide connection details
connectionDetails <- DatabaseConnector::createConnectionDetails(
  dbms = config::get("dbms", config = configBlock),
  user = config::get("user", config = configBlock),
  password = config::get("password", config = configBlock),
  connectionString = config::get("connectionString", config = configBlock)
)
```

By setting the configBlock to a block header in the `config.yml` file, we can pull the connectionDetails of the correct database. Say we had a configBlock in the yml called `omop_data`. If I set `omop_data` in the script, it will evaluate the connection to correct configuration using the `config` package. While not required to script in this fashion, Ulysses leverages this pattern often in its templated files.

5 Comments

5.1 Editing the `config.yml` file

It is possible to edit the `config.yml` file by either changing an existing item or adding a new block. Be sure to save and restart the R session after making any changes.

5.2 Common Issues

Keyring can be difficult to debug. A known issue when working in a Linux environment is the need to install libsodium and libsecret. Be sure to run these commands in the terminal prior to insalling keyring.

```
sudo yum install libsodium libsodium-devel
sudo yum install libsecret libsecret-devel
sudo chmod 777 /home/idies/.config
```

If the `config` package begins to return weird errors, it is likely that it is not able to evaluate the commands wrapped in keyring. To debug, either check the `keyring::key_get` call in the console or hard code the credential into the `config.yml` file.