# Octopus Energy technical challenge

This document describes a technical challenge that forms part of the Octopus Energy (OE) recruitment process for back-end engineers.

It should not take more than 3 hours. Don't worry if the project isn't complete by then - just email back what you have with some notes on where you got to and what's left to do.

## Background

OE needs to process so-called "flow" files in order to communicate with the energy industry. These are CSV files that are sent to us via sFTP. Our systems then import each file into our database.

There are lots of different types of flow files but, for this project, we are only concerned with the "NEM13" files which contain information about meter readings.

We need a new service that can import these files and allow their information to be browsed via the web by support staff. For this challenge, files will be imported via the command-line but, later on, a REST interface will be added to allow files to be uploaded via the web.

Some relevant industry terminology:

- **Meter point** - this is an abstract notion of a point of electricity consumption within a property. It is identified by a so-called NMI. It's not the same as a physical meter: for instance, a property could replace their analogue meter with a smart-meter but their meter point (and NMI) will not have changed.
- **Meter** - a physical device installed in someone's house for measuring electricity consumption. They normally have a serial number. A meter can record different types of electricity consumption using different "registers" (for example, meters in economy7 households have two registers to record day- and night consumption separately; meters in non-economy7 households normally only have one register).
- **Reading** - a decimal value that records the cumulative electricity consumption at a point in time.

# Requirements

## Deliverable

After the allotted time, a gzipped tarball should be emailed back to val@commoncode.io.

This tarball should contain:

- A Django web application (see below for application requirements)
- A .git folder with the commit history for the project
- A README that explains how to install and use the project

The project should work on macOS or Linux.

Feel free to document any assumptions made, or ideas for improving the project. Don’t upload the project to a public Github/Bitbucket repo (as this might offer an unfair advantage to future candidates if they stumble across it).

## Application requirements

The application should be a Django project that runs on Python 3.6 or 3.7.

It should have a management command that can be called with the path to a NEM13 file (or files). The specification for these files is included below.

The data for each reading in the file should be extracted and stored in a local database. The information available for each reading should include, as a minimum:

- NMI
- Meter serial number
- The reading value
- When the reading happened
- The filename of the flow file

It should provide a version of the Django admin site that allows a user to search for the reading values and dates associated with either:

- A NMI
- A meter serial number

There should be a test suite and instructions on how to run the tests.

### Application notes

Use a Postgres or SQLite database. If using SQLite, don’t commit the SQLite database file to source control.

The models and database schema are up to you.

Don’t worry about deployment to a remote environment - concentrate on ensuring the project works locally.

### Flow file specification

The specification of the NEM13 (and the other flow files) is publicly available on the AEMO website. Here’s the document for the NEM13 specification:
https://www.aemo.com.au/-/media/Files/Electricity/NEM/Retail_and_Metering/Metering-Procedures/2018/MDFF-Specification-NEM12--NEM13-v106.pdf

Warning: the structure of the documentation is a little hard to understand. The specification of the structure is on page 13 "Accumulation meter reading file specification and validation (NEM13)".

A sample NEM13 file can be found here:
https://gist.github.com/jamesbeith/3c71a46c5ede2683dd5118de985cc79b

## Success criteria

When reviewing your submission, we’ll be considering these aspects:

- Correctness - does it meet the requirements (i.e. can we follow your instructions to set up the project, import a file and see some data in the Django admin site)
- Maintainability - for example, how easy would it be for another developer to take over the project and start adding features
- Robustness - are errors handled gracefully