# Backend Engineering - Take-home exercise

The intent for this project is to demonstrate your thought process in solution development as well as coding style and practices. We've taken a simplified / subset of a real problem we have in prod and made it standalone (i.e. referencing txt files vs database, etc).

The "problem" is we store our data in pieces where each piece is one minute of data for one device we get netflow for. Each device/minute is a directory and in that directory are a bunch of binary and index files related to that minute's data. This is carried across multiple servers. The directory name/structure indicates the timestamp for the data in the directory.

Today there is a database for our metadata system that has a row for each directory we store, and maps each minute/device to a server (not all minutes have data).

Deleting data is one of the problems. Today we can query the DB and find all directories older than x days (for a particular device or company) and delete them from the DB and servers. Tomorrow we are moving away from this DB and going to rely strictly on the file structure for the device/timestamp as well as some other hints as to which server the device is on.

File structure is `/data/company_id/device_id/year#/month#/day#/hour#/min#/`

What we want you to do is write some code that based on the company ID will delete all files older than some number of days. For example company #1 gets to keep 90 days of data, company #2 gets 60 days and any company that does not have an explicit retention period defaults to 30 days. Assume there is a configuration file (config.json) that indicates the default number of days (30) and the specified retention period for a given company ID. An example of the configuration file is provided below. The example configuration file indicates that for company id 1001 the retention period is 60 days, for company id 1017 the retention period is 120 days, and all other company ids not specified default to 30 days.

```
$ cat config.json
{
    "retention": {
       "default": 30,
       "1001": 60,
       "1017": 120
    }
}
```

Assume your system is just one server, but that shouldn't matter as you could run this on 1 or 10 servers and it should work.

Important: assume the server that this operates on is production, busy, latency sensitive (can't just become unresponsive or overloaded for 30 seconds at a time) , may be spinning or SSD drives. It should be as lightweight as possible and play nice with other processes running on the server.

Assume there are ~200 devices on a server (so you can get a gauge of how many dirs/files you will be deleting at once). Assume each directory has ~50 data files inside it.

We can run this daily or weekly, whatever you recommend. Assume that some days it may get skipped (server down, etc)... And assume that there maybe times we change the retention # of days for a customer. So don't assume it always is just deleting 1 day of data, exactly x days ago. It may have to catch up as well. It's probably not super-safe to assume the unix ctime of the dir is the actual creation time (think data restores, etc.).

Example of actual directory: `ns1:/data/1289/2466/2021/03/15/21/56`

```
    1289 is company ID
    2466 is the device number
    2021 year
    03/15 march 15th
    21:56 UTC
```

Please include in your submission a short README.md file (paragraph or a few) indicating what your thought process was, what you did and didn't do (things you might add/improve on if you had all the time in the world), things you see as potential issues with what you implemented... and clear instructions about how to run the program. Please also try to provide a rough estimate of how much time you spent on the project. Do not include personal information, as we want to review submissions anonymously.

Your solution should be packaged as a private github repository or a tarball, containing:

- `README.md`
- `Dockerfile`
- *the rest of your code, scripts etc... no built artifacts (executables, jars, ...)*

If using a private github repo, let us know and we'll provide a github handle to grant access to.

Bonus points if your solution builds and runs as a docker container. For reference, ideally we want to be able to use something like the following commands to run and test the code:

```
$ docker build -t cleaner .
$ docker run -v /path/to/our/test/data:/mnt/data cleaner path-to-config-file
path-to-data unix-timestamp
```

Note: `unix-timestamp` is what your program should use as "now". This is so we can run reproducible tests.

Return it any time in the next couple weeks.

That's it! Let us know if you have any questions.