GINKGO ANALYTICS

Interview - Data Case

Data Dropzone Challenge

Hamburg, 2021

Content and formal explanations for applicant interview case

Instruction

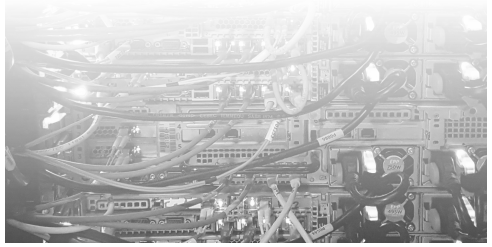
Your task is to create a service where clients will be able to send their data via POST requests.

You'll find a detailed description about the API we want you to build on slide 2. We have some more „nice-to-have" items collected on slide 3 which will help to turn the basic API from slide 2 into a more appropriate API. Think about these items and how you would approach implementing them. You don't need to implement them, but feel free to do so if time permits.

Data and Technology

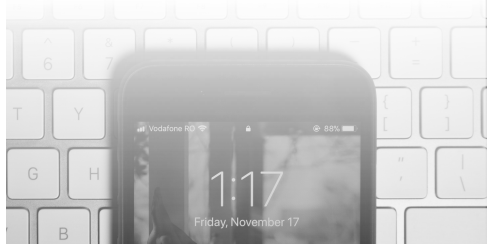
You will get access to a Google Cloud project where you will be assigned as an admin.

Feel free to use any coding language and technology that you feel is the most appropriate for the task and you feel comfortable with. The only requirement is that you version the code in Cloud Source Repository within the project so that we can review it and of course the final service should completely run within the Google Cloud project.



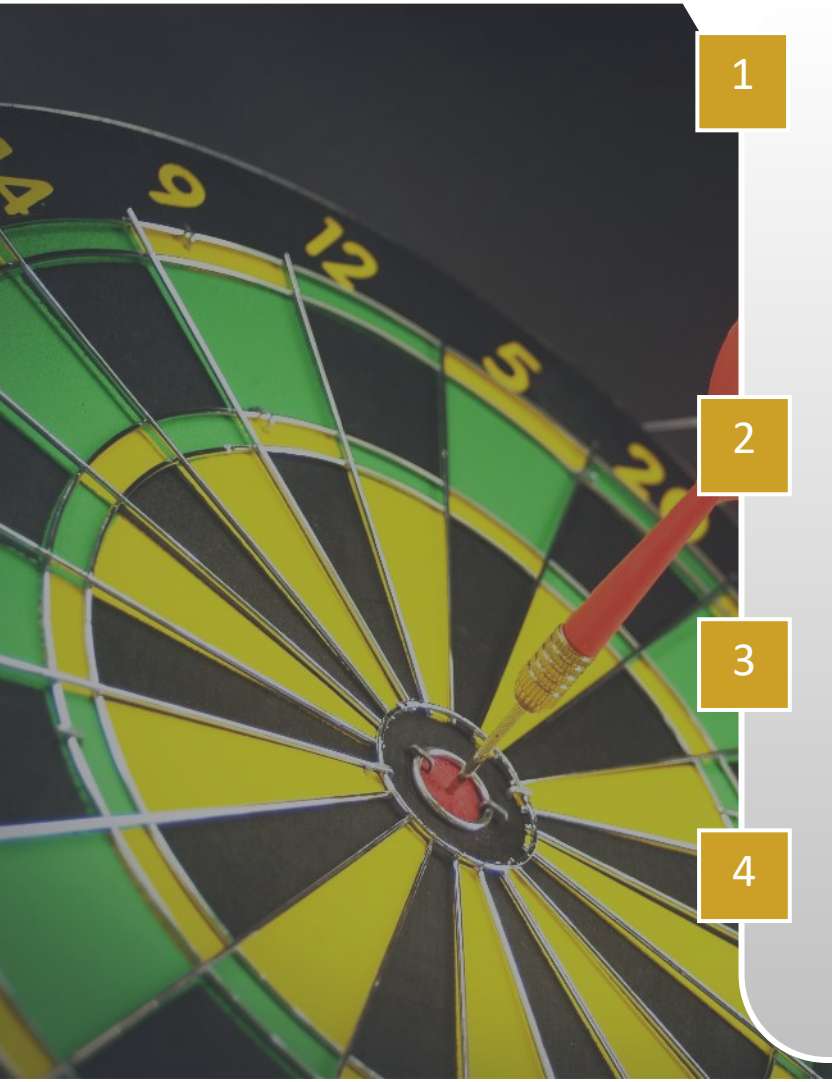
How to get access?

Please, send an email to samo.turk@ginkgo-analytics.com and raffael.vogler@ginkgo-analytics.com so that they can create your access. They need an email address from you which is registered with a Google account. It doesn't need to be a gmail address.



- Send an email to our contacts to get access to the Google Cloud project.
- Within a work day you should get an email with the access link and be able to prepare the case
- Prepare **a 15 min presentation** to structure your work and be prepared to walk us through your code after the presentation
- State and explain honestly the effort you put into the presentation (30min – x hours)
- Remember, this case shall demonstrate both your data engineering and your consulting skills
- Feel free to reach out to us at Ginkgo Analytics in case you have questions regarding the data case: samo.turk@ginkgo-analytics.com or raffael.vogler@ginkgo-analytics.com

Tasks that the service should perform



1

Data ingestion

- The data will be submitted in JSON format as a POST request with a command equivalent to:

```bash
curl -X POST -d @file.json http://service.com/endpoint
# or as form data, up to you
curl -X POST -F "file_upload=@file.json" http://service.com/endpoint
```

- Example Json is in the appendix and also in the archive provided with this challenge.

2

Storing raw data

- The raw data should be stored immediately using an appropriate storage option.

3

Data transformation

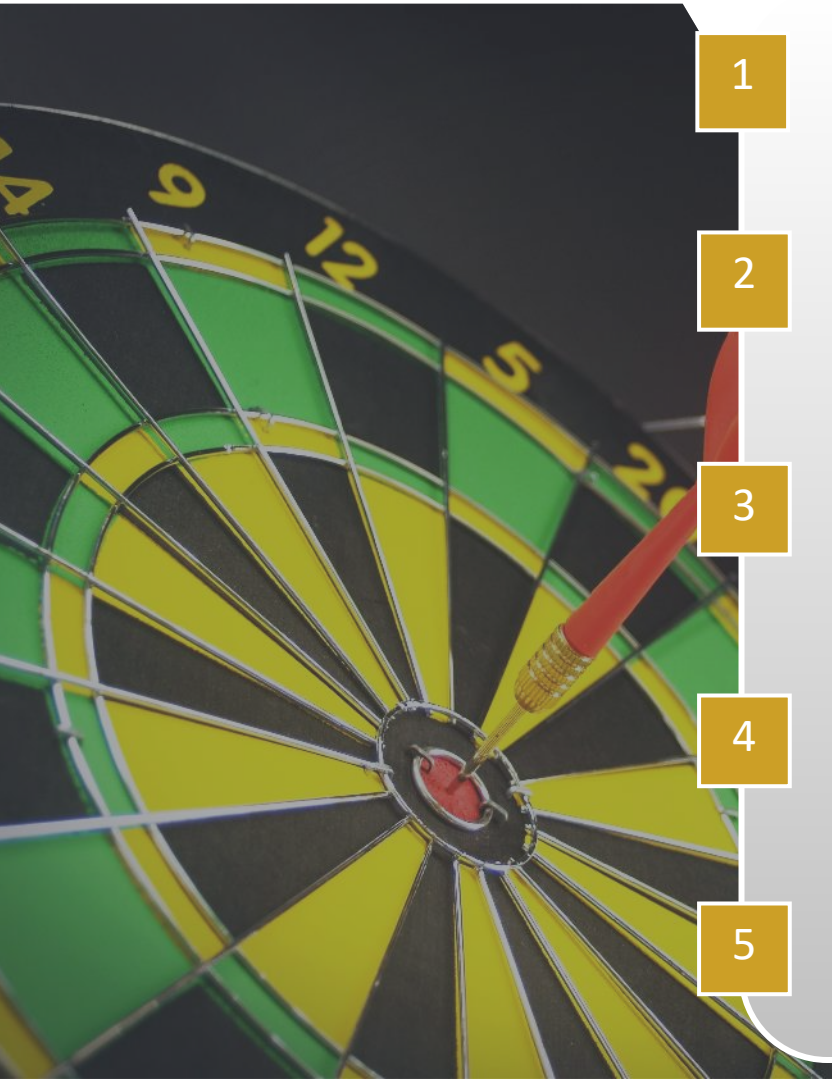
- Each json contains a `time_stamp` and `data` keys. Timestamps are in `US/Eastern` and should be converted to UTC. We are only interested in the mean and standard deviation of the data, so please calculate those.

4

Data storage

- For each processed json the service should store the UTC timestamp, mean and standard deviation in a database.

Nice-to-have items



1

API design

- API that comes with documentation and is standardized is developer friendly. Consider using Swagger/OpenAPI.

2

Scalability

- Try to select technology that scales well and design individual components also in a such way. For example it might make sense to separate the REST service and the part that is performing the transformations. You can connect them for example with a messaging service.

3

Security

- APIs exposed to the internet should be secured and should offer some way of authentication. What are the common way of performing authentication and how do you normally secure communication.

4

Testing

- Well tested code is usually more stable, secure, and easier to maintain. Do you know of any testing frameworks and how would you normally approach this?

5

CI/CD

- Automated deployment makes day to day operations much easier and more reproducible.

Appendix

Data example


```json
{"time_stamp": "2019-05-01T06:00:00-04:00", "data": [0.37948282157787916, 1.5890471705541012, 2.1883813840381885, 1.1890237286379994,
1.637653455859299, 2.877524839556612, 2.7595291419608494, 0.7476783472492925, 2.4951611280056394, 3.0294316660809453, 3.068577817188679, 1.1831557905625592,
1.5342857773722 73, 1.2785699123945657, 1.8281862430351499, 2.37098343002181, 0.5842991615849848, 0.7098352937365224, 1.092283952816859, 0.7480644642759615,
1.0640991188681785, 1.5510551937467487, -0.07653143433976473, 0.3156607006753255, 1.9009668439818326, 1.501715498813034, 1.431489168317764, 1.3623989582565668,
0.6824460690703411, 0.20728376966473516, -0.407879956629583, 0.21752811010881368, 1.5784411107634106, 2.389902962389714, -0.5365616480218045, -0.8296723795786696,
2.1287470425544344, 1.296716012226446, 0.20360163658128605, 0.10515623591517487, 0.7291980591945955, -0.9020872444853947, 1.8807109491949274, 1.111856784855985,
-0.5068115041337455, 1.3024598193063068, 2.4516981153284885, 2.253604583787444, 0.710617057111469, 0.42821986135102896, 1.2612573042481043, 0.16530677944550654,
-1.4877113196238811, 0.9649682869023166, 1.7069344582209716, 0.6994544311316577, 1.1015574622759359, -0.8978611941172703, 0.8466065812103564, -0.03582818653220454,
-0.5983353440444654, 0.29087666709478177, 0.71732006843272, -0.645888185757802, 0.9191159284564496, 1.7297312101247915, 0.8427795283341938, 2.7088810132044676,
0.7804664448279086, 1.9132547567469969, 0.4443548966571 8764, -0.7564611194085393, 2.123804007250331, 1.5451418182375432, 0.04384584830123639, 1.4201820763436874,
1.6040895477011274, 2.507884687679396, 2.7963764011678895, 1.4007502145419017, 1.0045678021645934, 0.1952036691188912, 1.51093143452691, 1.8299695553012922,
-1.4641845697404046, -0.029725350507989967, 0.6345081080159558, 1.8931406476303387, 0.4905885506279598, 0.25869324520466797, 1.0933920715888004, 1.307169026011377,
0.7120099139672635, -0.48948097889697006, 0.8808422003796418, 2.209799345501847, 1.867903190303514, 0.5308461059635922, 0.21304286987397825, 0.2535682481865035]}
```