Scenario

Scenario Introduction

The Product

We want to call the service “MachineSoup”. This will sit at the core of a few different products that require low-latency ingestion of data (e.g. for IoT), a **machine learning** processor to create the model, and then have the result queryable via this **innovative PostgreSQL module** that allows efficient vectorized spatial querying of the model. That might sound confusing, ***and it is***! We barely know what our Data Scientists are on to, but it sounds great and people really wants this. The exact processing that takes place will be a bit different between the different use-cases, so we want to design a secure way for Data Scientists to **upload code for these models**, to test out their models without stealing customer data, etc.

Scenario Introduction

The Environment

Assume that the environment upon which MachineSoup runs is basically a Linux-based OS that for the purposes of this exercise is “secure”.

Functional requirements

Producer sends raw data to the processing service.

- This service call is authenticated with a username and a service passphrase.
- The call also includes an orgId, modelId, and an optional model versioning label (modelId stays stable over time and the versioning label is described below)
- The orgId, modelId, and versioning label routes which actual processing service the raw data goes to.

The Processing Service will do some convolutional data-sciency stuff on the data then update the specified model in the special postgres database.

- The code run is based on the orgId, modelId and the versioning label
- Batches of raw data are stored in the postgres database as BLOB for later data-science efforts.
- BLOBs aren’t indexed so can’t be queried, just retrieved in relation to some timestamp, orgId, and modelId

Functional requirements

2/2

Consumers will use the query API to ask for the results.

- The Consumers are authenticated by a reusable component and the orgId and allowable modelIds + labels are known.
- If the consumer is now allowed to query a specific modelId or modelId+label, the query is denied.

Consumers with special privileges are allowed to “promote” a specific modelId + label to be the default when no label is specified for processing or querying.

- This is analogous to promoting some “staging” system into “production”.

The postgres database is queried by the query service on behalf of that end user.

- The results might be paginated/streamed via some in-memory caching system.

We need to be able to occasionally ship compiled C++ code for the PostgreSQL plugin that does the indexing.

Data Science requirements

Data scientists will want to tune the models on behalf of a customer, and so they need to be able to ship new code to the processing service

Data scientists can design new models by replaying the raw data (stored in BLOBs). This staged model is identified by modelId and mandatory label.

The Data scientist can use special credentials to access that staged model via the Consumer query API

Data scientists can stage these new models with a specific modelId for the Consumer to do their own testing

- The Data Scientists first upload a staged bit of code for the model then run code a job that replays the raw data from the BLOB

Legal requirements

Data Scientists are able to look at the raw data -- the ones stored in BLOBs for debugging purposes -- and test out models

Data Scientists cannot combine models or raw data in this type of system

Once entered, the raw data cannot leave the MachineSoup network environment, especially important to not have this data on laptops/workstations.

We need to send the customers a report of when exactly the data scientists access the data and there may be further regulation on how long sessions can be left idle.