



Credit Risk Assessment Backend in GCP

Ali Ghorbani

Humberto Ribeiro de Souza

Credit Score Assessment in GCP

Business Problem
Architecture
Project objective
Implementation
Results

Implements endpoints for provide credit score in production environment

For U of T class 3760



Business Problem

- **Credit Default Risk as a Tool for Standardized Decision-Making**

Credit risk tools are part of any financial institution to evaluate if money should be borrowed for all kinds of purposes. Some are purpose-built and more accurate for a given business case.

- **Credit Default Risk as an Inclusion Tool**

Diverse initiatives across the globe have been designed and executed, some of them being even being awarded the Nobel Prize. In this case, the “Home Credit” financial institution has posted in Kaggle a problem that would provide credit to the unbanked population using alternative data such as telco and transactional information, to evaluate default likelihood. More can be found [here](#).



Assumptions

- 
- 01 Assumed that the Credit Home Company, which posted the Kaggle Challenge, has accepted the model to be promoted to production. The company has an aggressive growth business plan and could go tenfold in two years
 - 02 Assumed that the team must select the best cloud provider for this need, considering that this is expected to be a low-profit and low-cost solution. Quality and simplicity are important, while response times are tolerable
 - 03 The scope of this project is to provide an endpoint to Credit Home Company to query about credit default risk of new applicants.
 - 04 Deprecated data for training models should be kept for audit purposes for 5 years. The model is expected to be updated fortnightly.



Cloud Technology Selection

- Project and Volumes

Credit Home Corp expects to query between 600 - 1200 applications per day.

The front-end will send a CSV containing a header and a row that belongs to the applicant.

- Security Risks

The system handles both financial and personal data therefore the connection must be secured. Sniffing, Man-in-the-middle and classical attacks should be avoided.

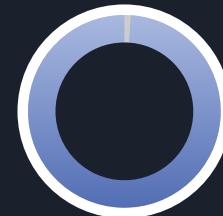
The company has sites in different states, has a team working from home and no VPN available at the moment.



On-Premises exclusion

Although the company has on-premise servers for their legacy Active Directory and related services, running models would be unsafe for the models are developed in Python 3.9 and Python libraries depend on operating systems library that needs to be patched.

Also, the solution would not scale according with the business plan without potential limitations to the business.



Cloud computing allows the application full-cycle to be managed without hardware and infrastructure concerns.



Cloud Provider Selection

The company has agreements with Google Cloud Platform and Amazon Web Services

The estimated monthly costs for the PROD environment are available in the table below

| Component | Expected Usage | AWS | GCP |
|--------------------------|---|-----------|------------------|
| Bucket per Gb | 5 Gb inbound; 4 Gb moving within cloud | 0.19 USD | 0.115 USD |
| ML Engine per query | 2 runs, notebook, 8 hours of nodes running | 9.63 USD | 4.72 USD |
| Database per Gb + access | 30 Mbytes monthly AWS DynamoDB; GCP FireStore | 0.30 USD | Free |
| Functions | 500 ms of running code per req, 2 Gb of Ram, 1200 reqs per day | 0.62 USD | Free |
| Total/Month | | 10.74 USD | 4.84 USD |





Simplicity x Transparency

Different approaches were considered for building the backend. The simplest would be based on all resources available in Vertex AI. The implementation time would be the smallest. An serverless model has been chosen for audit and transparency for a reasonable amount of implementation time.



Vertex AI and API
Endpoints,
Firestore



Trained model
available in Lambda,
FireStore



Cloud VMs,
Network and access
Management

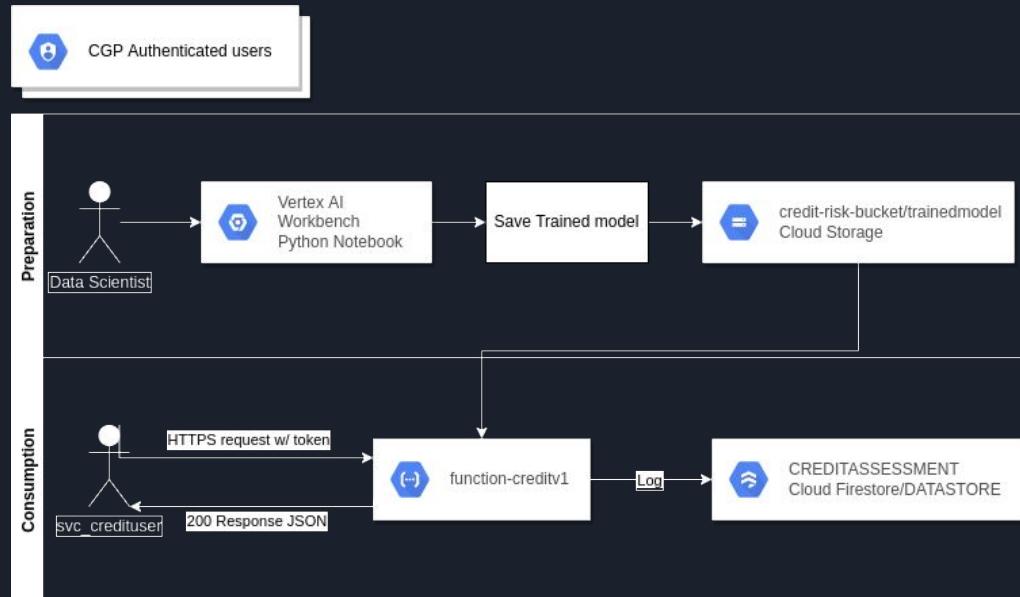


Acquire/setup
hardware, OS, VMs,
code



Architecture

The scope of this implementation is to expose the API endpoint for the Home Credit Company and allow assertion of the applicant credit score. The request must be recorded on a database for billing and auditing purposes. The model data is expected to be trained fortnightly and the previous data need to be stored so the model can be audited on a given point in time.



Project timeline - 2021/2022





Architecture

Justify selection of Cloud Functions over App engine or Cloud Run

- Event-driven solutions that extends to google are a good fit for cloud functions and we built function-based endpoint API therefore cloud function is the suitable choice.
- Less total cost of ownership and time-to-deploy
- App engine is suitable for many different services with a single application while cloud function preferred for individualized service.
- Cloud Functions-like services are found on other cloud providers and it is easily portable if a change of cloud provider is requested
- Cloud Run runs containers and is suitable choice to run in a stateless container. But, because of single query request of out end-pointAPI, we chose to use cloud function.



Architecture

Selection of Vertex AI Managed Notebooks over VM

- Maintain operating system (OS) and all software upgrade using VM is not core for the company nor its analysts
- Security risk of VM and continuously patching OS and ensure to keep it up to date
- Do not have native access to other GCP services and native integrations capabilities using VM. This can be arranged. However, the modules are frequently updated and it incurs in maintenance overhead.
- Not selected Auto ML to avoid platform dependency and maintain better visibility and easier explainability of the workflow



Architecture

Justify selection of FireStore/Datastore

- Endpoint API needs a highly scalable document NoSQL database for small to medium size operational workloads
- It is cheaper than all listed contenders*
- Firestore automatically handles sharding and replication, with a highly available and durable database that scales automatically
- FireStore is built for automatic scaling, high performance, and ease of application development
- Firestore has key values, attributes and indices
- Firestore supports Atomicity, Consistency, Isolation, and Durability (ACID) transactions, SQL-like queries, indexes.

* <https://sourceforge.net/software/compare/Amazon-DynamoDB-vs-Google-Cloud-Firestore-vs-Openredis-vs-Oracle-Database/>



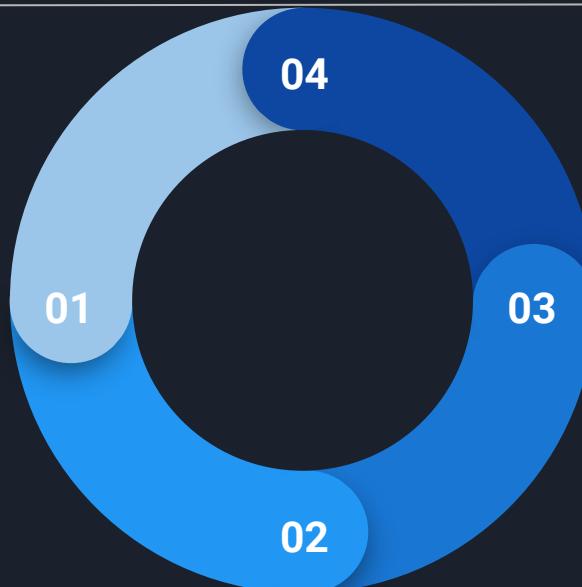
Cycle diagram

Update Model Fortnightly

A Data Analyst updates the model having the latest data at hand. Previous data is saved on bucket subfolders *

API is enabled

The front-end system consumes it



Audit

Review of results of a given applicant may be requested to run. The ensemble of information of usage and model data are necessary for the data analyst to run the audit*

Usage is recorded

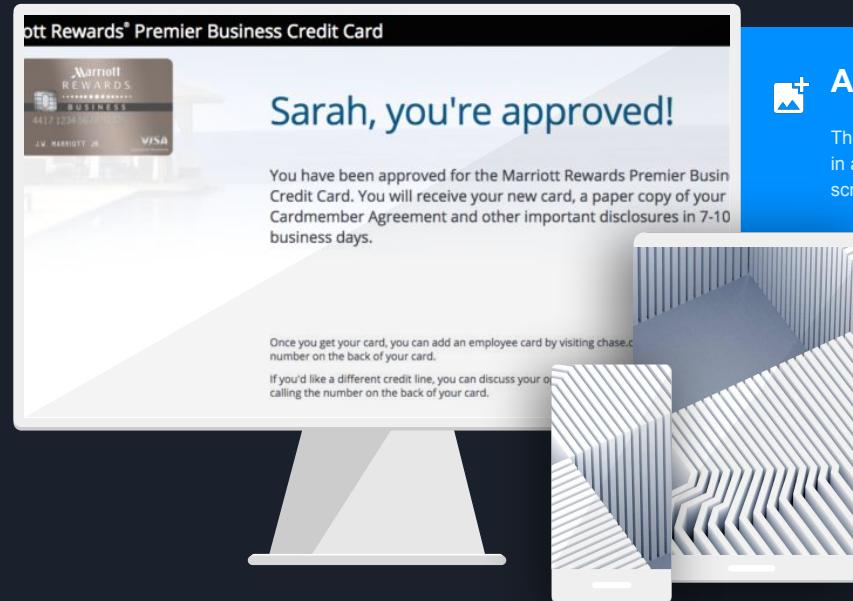
Usage is recorded for being queried by the front-end

* Automation of steps 1 and 4 are scope of another project

Introducing: Credit Risk Assessment System

The endpoint is called from an application that runs on desktop, tablet or mobile

Access to the endpoint is open by default. It requires the service account svc_credituser key to have access to it.



Approved/Reproved

The front-end system shows the status in a way the analyst can show the screen.

Google Cloud Platform Credit Risk Assessment Search products and resources

Vertex AI Notebooks NEW NOTEBOOK REFRESH SHOW INFO PANEL

MANAGED NOTEBOOKS PREVIEW USER-MANAGED NOTEBOOKS EXECUTIONS PREVIEW SCHEDULES PREVIEW

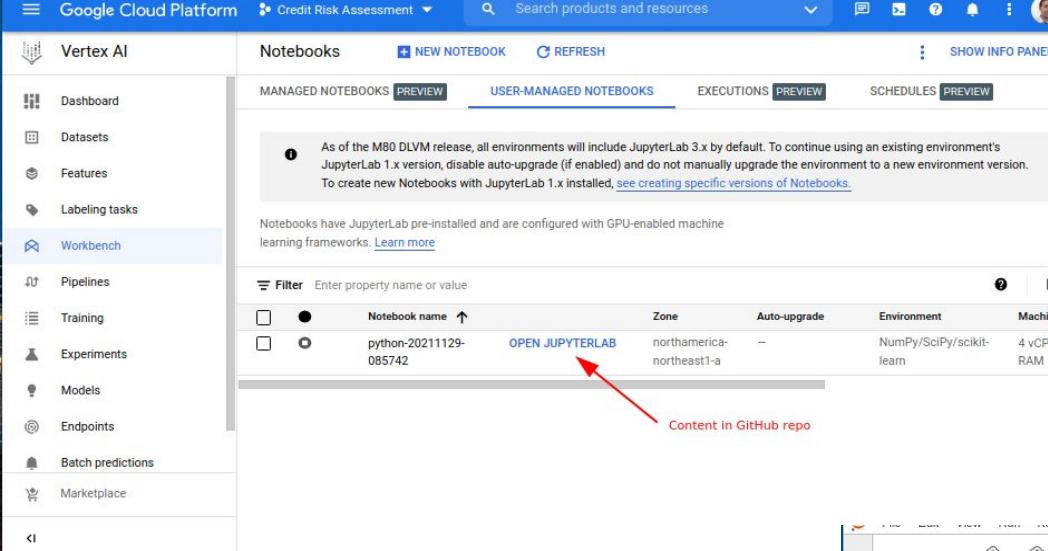
As of the M80 DLVM release, all environments will include JupyterLab 3.x by default. To continue using an existing environment's JupyterLab 1.x version, disable auto-upgrade (if enabled) and do not manually upgrade the environment to a new environment version. To create new Notebooks with JupyterLab 1.x installed, see creating specific versions of Notebooks.

Notebooks have JupyterLab pre-installed and are configured with GPU-enabled machine learning frameworks. Learn more

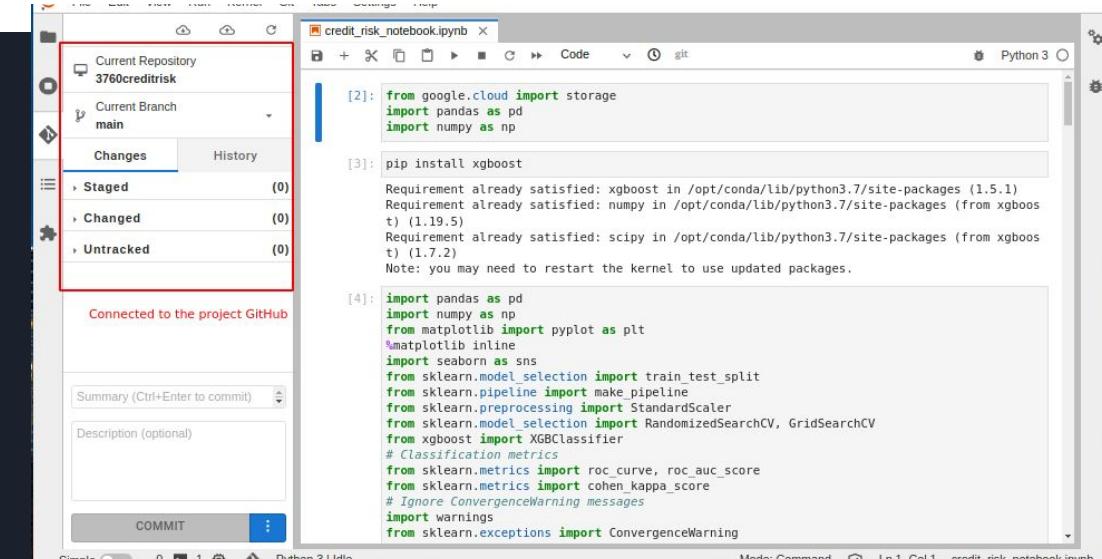
Filter Enter property name or value

| | Notebook name ↑ | Zone | Auto-upgrade | Environment | Machine type |
|--------------------------|------------------------|---------------------------|--------------|--------------------------|------------------|
| <input type="checkbox"/> | python-20211129-085742 | northamerica-northeast1-a | - | NumPy/SciPy/scikit-learn | 4 vCPUs, 128 RAM |

OPEN JUPYTERLAB Content in GitHub repo



Vertex AI Notebook - Model building and exporting



```
from google.cloud import storage
import pandas as pd
import numpy as np

pip install xgboost

Requirement already satisfied: xgboost in /opt/conda/lib/python3.7/site-packages (1.5.1)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from xgboost) (1.19.5)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from xgboost) (1.7.2)
Note: you may need to restart the kernel to use updated packages.

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
from xgboost import XGBClassifier
# Classification metrics
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import cohen_kappa_score
# Ignore ConvergenceWarning messages
import warnings
from sklearn.exceptions import ConvergenceWarning
```

- > Vertex AI - Notebook was used to adapt the model to work in the cloud.
- > The train and test dataset have been cleaned-up and feature-engineered.
- > The model uses XGBoost classifier using randomized search for hyperparameter optimization.
- > The validation method was AUC for the dataset is imbalanced.

Pickle files and function code are in this folder

| Name | Size | Type | Created | Storage class | Last modified |
|------------------------------------|----------|--------------------------|--------------|---------------|---------------|
| HomeCredit_columns_description.csv | 36.5 KB | application/vnd.ms-excel | Nov 22, 2023 | Standard | Nov 22, 2023 |
| POS_CASH_balance.csv | 36.3 MB | application/vnd.ms-excel | Nov 22, 2023 | Standard | Nov 22, 2023 |
| application_train.csv | 158.4 MB | application/vnd.ms-excel | Nov 22, 2023 | Standard | Nov 22, 2023 |
| bureau.csv | 89.7 MB | application/vnd.ms-excel | Nov 22, 2023 | Standard | Nov 22, 2023 |
| bureau_balance.csv | 358.2 MB | application/vnd.ms-excel | Nov 22, 2023 | Standard | Nov 22, 2023 |
| credit_card_balance.csv | 88.6 MB | application/vnd.ms-excel | Nov 22, 2023 | Standard | Nov 22, 2023 |
| function-code/ | - | Folder | - | - | - |
| function_code_new_record_eval1.c | 2.9 KB | text/plain | Dec 6, 2023 | Standard | Dec 6, 2023 |
| installments_payments.csv | 689.6 MB | application/vnd.ms-excel | Nov 22, 2023 | Standard | Nov 22, 2023 |
| previous_application.csv | 386.2 MB | application/vnd.ms-excel | Nov 22, 2023 | Standard | Nov 22, 2023 |

- Single bucket for the project: credit-risk-bucket
- Training model files for notebook are in the root
- Cloud functions-related code is sorted in functions-code folder
 - Trained model pickle
 - One-Hot encoded columns pickle
 - Zipped cloud functions code for automated deployment

Cloud Storage and service account IAM setup

```

shell:~/gcloud (credit-risk-assessment)$ history
install google-cloud-sdk --classic
st --console-only
service-accounts create svc-credituser
objects add-iam-policy-binding credit-risk-assessment --member="serviceAccount:svc-credituser@credit-risk-assessment."
"roles/cloudfunctions.invoker"
objects add-iam-policy-binding credit-risk-assessment --member="serviceAccount:svc-credituser@credit-risk-assessment."
"roles/storage.admin"
objects add-iam-policy-binding credit-risk-assessment --member="serviceAccount:svc-credituser@credit-risk-assessment."
"roles/storage.objectViewer"
objects add-iam-policy-binding credit-risk-assessment --member="serviceAccount:svc-credituser@credit-risk-assessment."
"roles/datastore.owner"
cloud

service-accounts keys create svc_tester_key.json --iam-account=svc-tester@credit-risk-assessment.iam.gserviceaccount
GOOGLE_APPLICATION_CREDENTIALS="/home/humberto/gcloud/svc_credituser_key.json"
https://northamerica-northeast1-credit-risk-assessment.cloudfunctions.net/function-7test -H "Authorization: bearer $(gcloud auth print-identity-token)" --data-binary @record_req1.csv "https://northamerica-northeast1-credit-risk-assessment.cloudfunctions.net/function-7test"

-H "Authorization: bearer $(gcloud auth print-identity-token)" --data-binary @record_req1.csv "https://northamerica-northeast1-credit-risk-assessment.cloudfunctions.net/function-7test"

shell:~/gcloud (credit-risk-assessment)$ ||

```

Google Cloud Platform Credit Risk Assessment Search products and resources

Cloud Functions Edit function

Configuration Code

Runtime Python 3.9

Source code Inline Editor

main.py requirements.txt

Entry point* process_request

```

21
22 def process_request(request):
23
24     storage_client = storage.Client()
25     bucket = storage_client.get_bucket('credit-risk-bucket')
26     # Get the uploaded file
27     fileContent = request.files['record']
28     blob = bucket.blob('function_code_new.' + fileContent.filename)
29     file_content = request.files['record'].read()
30     blob.upload_from_string(file_content.decode('utf-8'))
31
32     # # = bytes(mystring, 'utf-8')
33     #cleans and feature engineer the arrived dataset
34     clean_input = request_cleanup(file_content.decode('utf-8'))
35
36     # Load the trained model
37     trained_model = load_model_data(request.files['record'])
38
39     # Run the model against the requested record to assess
40     credit_score = run_model(clean_input, trained_model)
41
42     result = "Approved" if credit_score[0] == 0 else "Declined"
43
44     # Store the request log in Firestore in Datastore mode
45     store_log(request, result, str(credit_score[1]))
46
47     return ('score:' + result + ",default_probability:" + str(credit_score[1]) + ")
48
49     def store_log(request,result,probability):
50
51         record_data = "cr_request_origin:" + str(request.remote_addr) + ":" + \

```

PREVIOUS DEPLOY CANCEL

LEARN Home Recommended for you Create an HTTP function Create a simple function in the Cloud Console. Tutorials Troubleshooting Troubleshoot issues with Cloud Functions. Support Configuring Cloud Functions Use configuration options to control the behavior of your Cloud Functions. Tutorials You might also like Tutorials Walkthroughs and guides Concepts Deep dive explanations Resources Pricing, release notes, and tools Support Product support topics All product documentation

The function /get_credit_score_v1 calls

- The Cloud Storage for read and write operations
- Loads the trained model and reference record
- Assess the record and returns the status
- Saves the transaction log in FireStore running as Datastore

get_credit_score_v1 Version Version 1, deployed at Dec 6, 2021, 8:01:54 PM...

METRICS DETAILS SOURCE VARIABLES TRIGGER PERMISSIONS LOGS TESTING

Execution time It is compliant to the requirement Less than 1 second

Execution time Milliseconds/call

Memory utilization MB/call Can reduce system memory from 1 Gb to 512 Mbytes

Memory utilization MB/call

Active instances

Active instances

Functions Setup

Postman interface showing a POST request to https://northamerica-northeast1-credit-risk-assessment.cloudfunctions.net/api/assessment. The Body tab shows form-data with 'record' and 'logged_user' fields. The response status is 200 OK with a body of {"score": "Approved", "default_probability": "0.3067828"}.

Validating with POSTMAN and FireStore

Credit Risk Assessment

| | Name/ID | active | billed | last_updated | logged_user | probability | remote_ip | result |
|--------------------------|---------------------|--------|--------|-------------------------------|--------------------------|-------------|---------------|----------|
| <input type="checkbox"/> | id=5083538508480512 | true | false | 2021-12-06 (20:25:59.605) EST | JohnDoe@outlook.com | 0.3067828 | 169.254.8.129 | Approved |
| <input type="checkbox"/> | id=6199595021369344 | true | false | 2021-12-06 (20:21:13.021) EST | JohnDoe@outlook.com | 0.3067828 | 169.254.8.129 | Approved |
| <input type="checkbox"/> | id=4831169585610752 | true | false | 2021-12-06 (20:14:01.250) EST | MarcioGarcia@outlook.com | 0.3067828 | 169.254.8.129 | Approved |
| <input type="checkbox"/> | id=4865928487501824 | true | false | 2021-12-06 (20:13:58.508) EST | MarcioGarcia@outlook.com | 0.3067828 | 169.254.8.129 | Approved |
| <input type="checkbox"/> | id=6263382969679872 | true | false | 2021-12-06 (20:13:54.093) EST | MarcioGarcia@outlook.com | 0.3067828 | 169.254.8.129 | Approved |
| <input type="checkbox"/> | id=5073695114526720 | true | false | 2021-12-06 (20:08:48.365) EST | MarcioGarcia@outlook.com | 0.3067828 | 169.254.8.129 | Approved |
| <input type="checkbox"/> | id=5712837116690432 | true | false | 2021-12-06 (20:00:50.384) EST | MarcioGarcia@outlook.com | 0.3067828 | 127.0.0.1 | Approved |
| <input type="checkbox"/> | id=5704568633556992 | true | false | 2021-12-06 (18:14:12.462) EST | MarcioGarcia@outlook.com | 0.3067828 | 127.0.0.1 | Approved |
| <input type="checkbox"/> | id=6273303371055104 | true | false | 2021-12-06 (18:12:39.065) EST | – | 0.3067828 | 127.0.0.1 | Approved |
| <input type="checkbox"/> | id=623854469164032 | true | false | 2021-12-06 (18:09:53.514) EST | – | 0.3067828 | 127.0.0.1 | Approved |

An authorized request can be made using Postman or curl.

All parameters are requested, along with the authorization key.

The result is a JSON record having

- Result: ['Approved', 'Declined']
- Default_probability in %

Top Challenges

1. Working with Cloud Functions specificities
2. Setting up IAM/Authorization roles without breaking everything
3. Choosing tools for a forecast of exponential growth

The image shows a dual-pane interface. On the left is a screenshot of a Visual Studio Code editor window titled "test_end_point.py - 3760creditrisk - Visual Studio Code". It displays Python code for a Cloud Function, specifically a function named "request_cleanup". The code reads a CSV file, performs data cleaning, and replaces certain occupation types. Below the code is a terminal window showing the function running on port 5000. On the right is a screenshot of the Google Cloud Firestore IAM roles page. The "Predefined roles" section lists four roles: "roles/datastore.owner", "roles/datastore.user", and two others partially visible. Each role has a list of permissions and a brief description.

Google Cloud

Why Google Solutions Products Pricing Docs Support English Console Contact Us

store Overview Guides Reference Samples Support Resources

Predefined roles

With IAM, every API method in Firestore requires that the account making the API request has the appropriate permissions to use the resource. Permissions are granted by setting policies that grant roles to a user, group, or service account. In addition to the primitive roles, `owner`, `editor`, and `viewer`, you can grant Firestore roles to the users of your project.

The following table lists the Firestore IAM roles. You can grant multiple roles to a user, group, or service account.

| Role | Permissions | Description |
|------------------------------------|--|---|
| <code>roles/datastore.owner</code> | <code>appengine.applications.get</code> <code>datastore.*</code> <code>resourcemanager.projects.get</code> <code>resourcemanager.projects.list</code> | Full access to Firestore. |
| <code>roles/datastore.user</code> | <code>appengine.applications.get</code> | Read/write access to data in a Firestore database. Intended for application |

Thank you!

Ali Ghorbani

Humberto Ribeiro de Sousa

2021-12-07

