

Tracking Cryptocurrency Exchange Trades with Google Cloud Platform in Real-Time

GSP603



Google Cloud Self-Paced Labs

Overview

Today's financial world is complex, and the old technology used for constructing financial data pipelines isn't keeping up. With multiple financial exchanges operating around the world and global user demand, these data pipelines have to be fast, reliable and scalable.

Currently, using an [econometric](#) approach—applying models to financial data to forecast future trends—doesn't work for real-time financial predictions. And data that's old, inaccurate or from a single source doesn't translate into dependable data for financial institutions to use. But building pipelines with Google Cloud can solve some of these key challenges. In this post, we'll describe how to build a pipeline to predict financial trends in microseconds. We'll walk through how to set up and configure a pipeline for ingesting real-time, time-series data from various financial exchanges and how to design a suitable data model, which facilitates querying and graphing at scale.

You'll find a tutorial below on setting up and deploying the proposed architecture using Google Cloud, particularly these products:

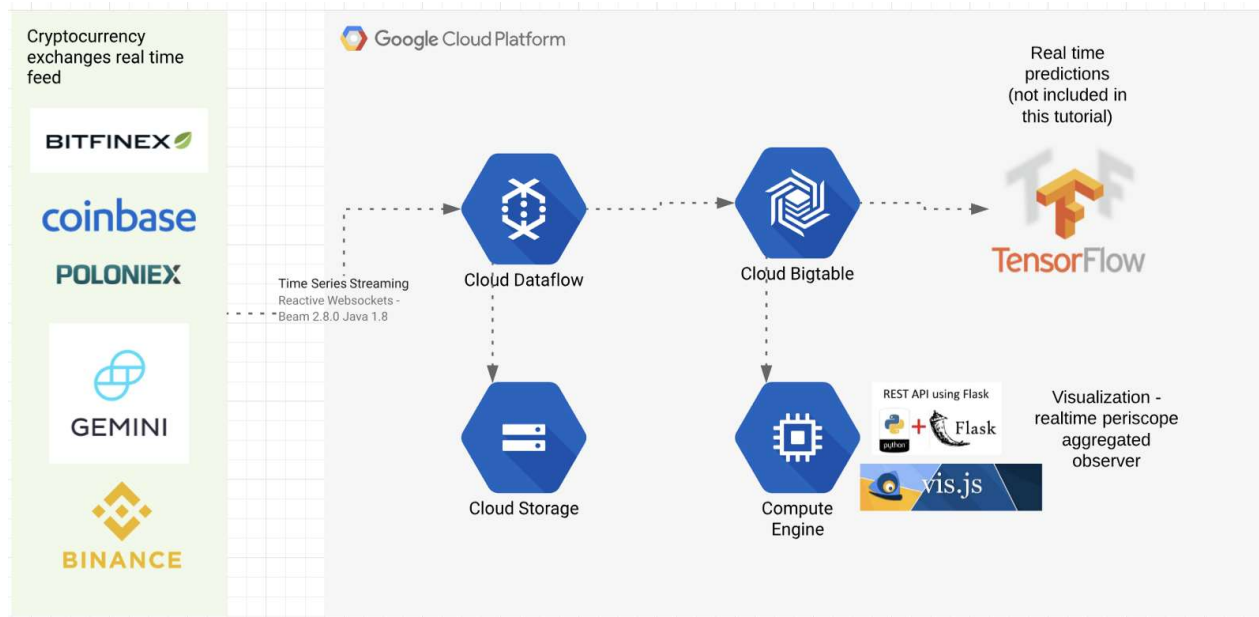
- [Cloud Dataflow](#) for scalable data ingestion system that can handle late data
- [Cloud Bigtable](#), our scalable, low-latency time series database that's reached [40 million transactions per second](#) on 3500 nodes. Bonus: Scalable ML pipeline using [Tensorflow eXtended](#), while not part of this tutorial, is a logical next step.

The tutorial will explain how to establish a connection to multiple exchanges, subscribe to their trade feed, and extract and transform these trades into a flexible format to be stored in Cloud Bigtable and be available to be graphed and analyzed.

This will also set the foundation for ML online learning predictions at scale. You'll see how to graph the trades, volume, and time delta from trade execution until it reaches our system (an indicator of how close to real time we can get the data). You can find more details on [GitHub](#) too.



Requirements / Solutions



Architectural overview

The typical requirement for trading systems is low latency data ingestion, and for this lab is extended with near real-time data storage and querying at scale. You will learn the following from this lab:

1. Ingest real-time trading data with low latency from globally scattered datasources / exchanges. Possibility to adopt data ingest worker pipeline location. Easily add additional trading pairs / exchanges. Solution: [Dataflow](#) + [Xchange Reactive Websockets Framework](#)
2. Demonstrate an unbounded streaming source code that is runnable with multiple runners. Solution: [Apache BEAM](#)
3. Strong consistency + linear scalability + super low latency for querying the trading data. Solution: [Bigtable](#)
4. Querying and visualization — Execute time series queries on Bigtable visualize it in on the webpage. Solution: [Python Flask](#) + [Vis.js](#) + [Google BigTable Python Client](#)

Architecture/How it works

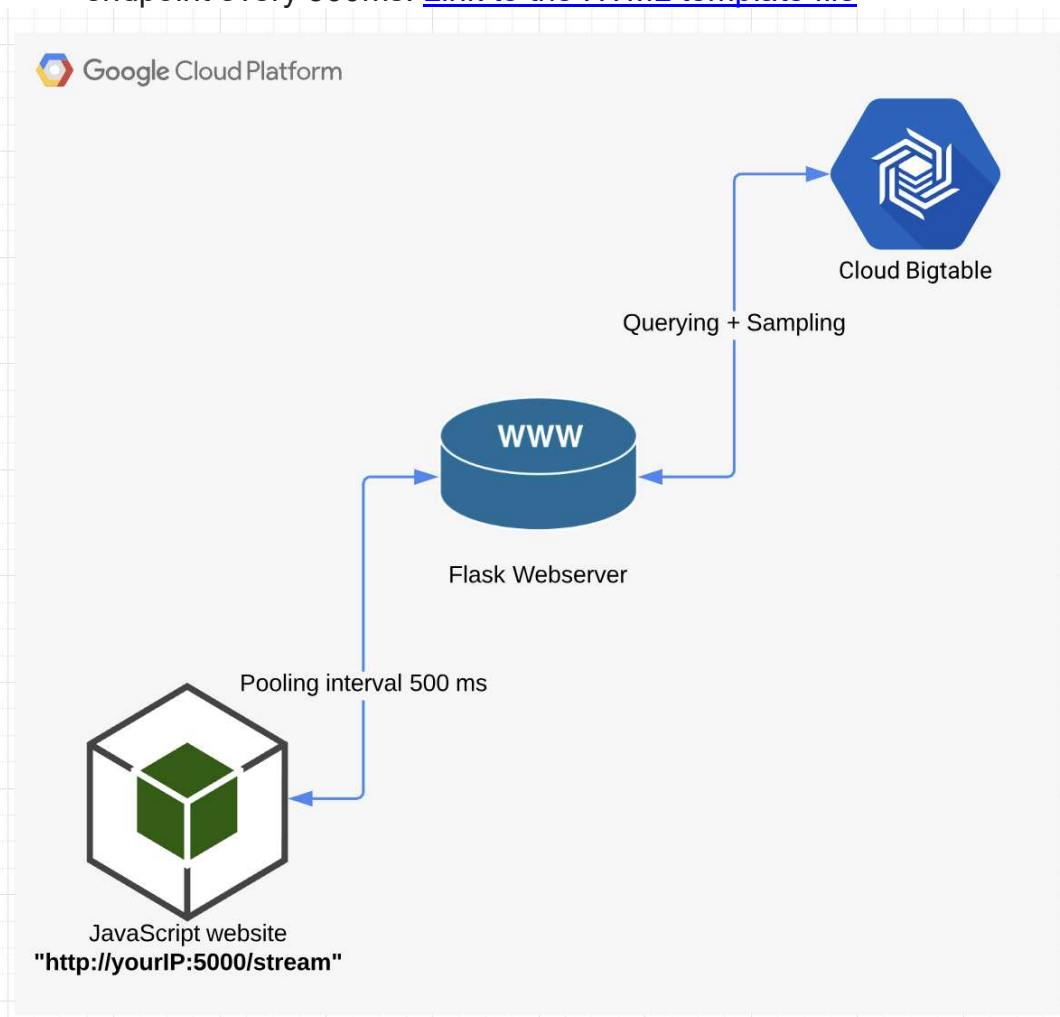
The source code is written in Java 8, Python 3.7, JavaScript; and Maven, PIP for dependency/build management.

The code can be divided into five main framework units:

1. Data ingestion — The XChange Stream framework ([Github link](#)) Java library provides a simple and consistent streaming API for interacting with Bitcoin and other cryptocurrency exchanges via WebSocket protocol. XChange library is providing

new interfaces for streaming API. Users can subscribe for live updates via reactive streams of RxJava library. We use this JAVA 8 framework to connect and configure some exchanges (BitFinex, Poloniex, BitStamp, OkCoin, Gemini, HitBTC, Binance...). [Link to the exchange / trading pair configuration code](#)

2. Parallel processing — Apache Beam ([Github link](#)) Apache Beam is an open source unified programming model to define and execute data processing pipelines, including ETL, batch, and stream (continuous) processing. Supported runners: Apache Apex, Apache Flink, Apache Gearpump, Apache Samza, Apache Spark, and Google Cloud Dataflow. You will learn how to create an unbounded streaming source/reader and manage basic watermarking, checkpointing, and record ID for data ingestion. [Link to the bridge between BEAM and XChange Stream framework](#)
3. BigTable sink — Cloud Bigtable with Beam using the HBase API. ([Github link](#)) Connector and writer to Bigtable. You will see how to create a row key and create a Bigtable mutation function prior to writing to Bigtable. [Link to the BigTable key creation / mutation function](#)
4. Realtime API endpoint — Flask web server at port:5000 + BigTable client ([Github link](#)) will be used to query the Bigtable and serve as API endpoint. [Link to the BigTable query builder](#) + [results retrieval and sampling](#)
5. JavaScript Visualization — [Vis.JS](#) Flask template that will query the real-time API endpoint every 500ms. [Link to the HTML template file](#)



Flask web server will be run in the Google Cloud VM instance

Pipeline definition

For every exchange + trading pair, a [different pipeline instance](#) is created. The pipeline consists of 3 steps:

1. [UnboundedStreamingSource](#) that contains 'Unbounded Streaming Source Reader' (bitStamp2)
2. [BigTable pre-writing mutation and key definition](#) (ETH-USD Mut2)
3. [BigTable write step](#) (ETH-USD2)



Bigtable row key design decisions

The DTO for this lab looks like this:

Class name	Package	Description
TradeLoad	data	DTO - Data transport object consisting of <ol style="list-style-type: none"> 1) <i>trading volume</i> (may be negative for type ASK on some exchanges) 2) <i>price</i> - current price 3) <i>orderType</i> - BID or ASK 4) <i>market</i> - exchange e.g. bitfinex, bitstamp 5) <i>delta</i> - milliseconds difference between system time and time when the trade was received in dataflow 6) <i>exchangeTime</i> - timestamp of trade from exchange

The row key structure is formulated in the following way:

`TradingCurrency#Exchange#SystemTimestampEpoch#SystemNanosTime`

E.g: a row key might look like **BTC/USD#Bitfinex#1546547940918#63187358085**

BTC/USD — Trading Pair

Bitfinex — Exchange

1546547940918 — Epoch timestamp ([more info](#))

63187358085 — System Nano time ([more info](#))

Why is nanotime added at the key end?

Nanotime is used to avoid multiple versions per row for different trades. Two DoFn mutations might execute in the same Epoch ms time if there is a streaming sequence of TradeLoad DTOs. `NanoTime` at the end will split Millisecond to an additional one million.

In your own environment, if this is not enough, you can hash the volume / price ratio and attach the hash at the end of the row key.

Row cells will contain an exact schema replica of the exchange TradeLoad DTO (see earlier in the table above). This choice will help you go from a specific (trading pair) — (exchange) to less specific (timestamp — nanotime) and avoid hotspots when you query the data.

Set up

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

To complete this lab, you need:

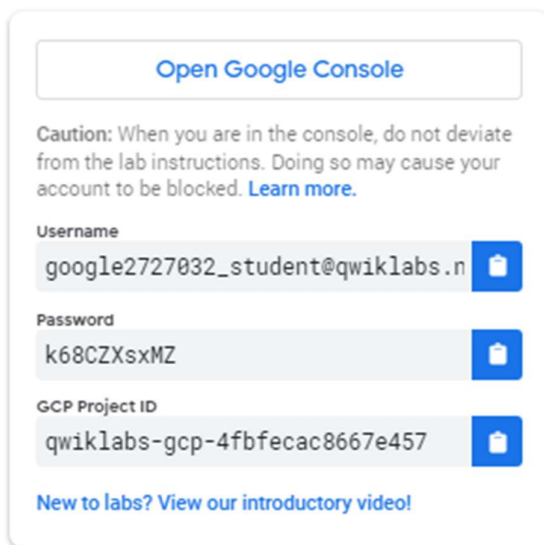
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

Note: If you are using a Pixelbook, open an Incognito window to run this lab.

How to start your lab and sign in to the Google Cloud Console

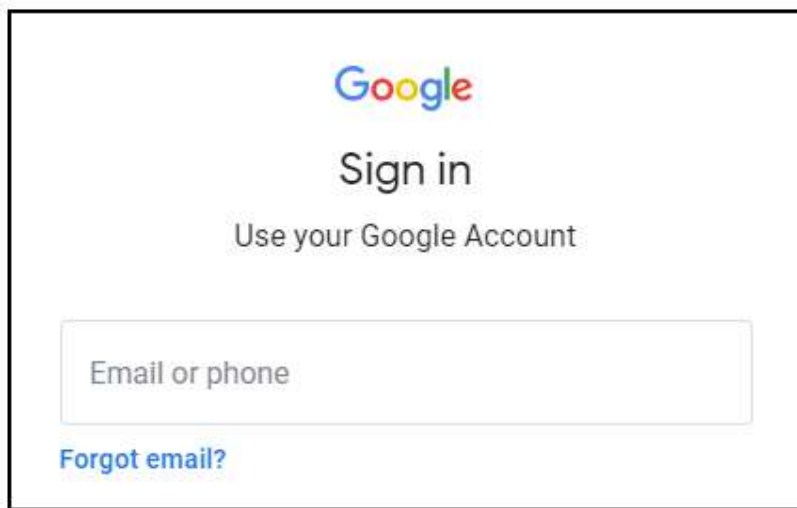
1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



The screenshot shows a sign-in panel with the following elements:

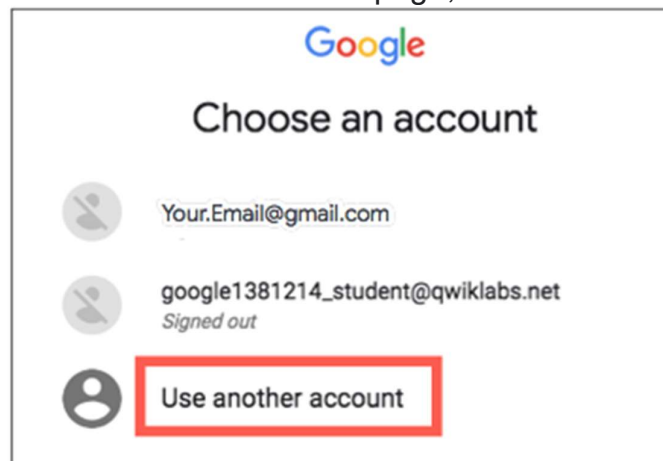
- A button at the top labeled "Open Google Console".
- A caution message: "Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)"
- Three input fields, each with a copy icon to its right:
 - Username:** google2727032_student@qwiklabs.n
 - Password:** k68CZXsxMZ
 - GCP Project ID:** qwiklabs-gcp-4fbfecac8667e457
- A link at the bottom: "New to labs? View our introductory video!"

2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Tip: Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another**



Account.

3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

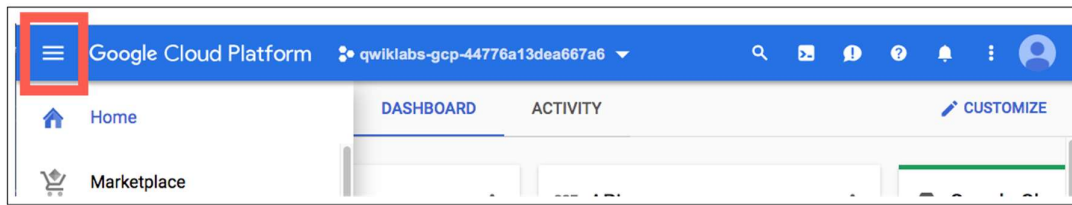
4. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at

the top-left.

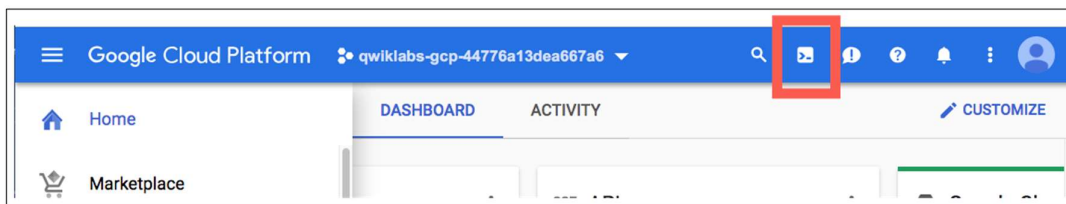


Start Cloud Shell

Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



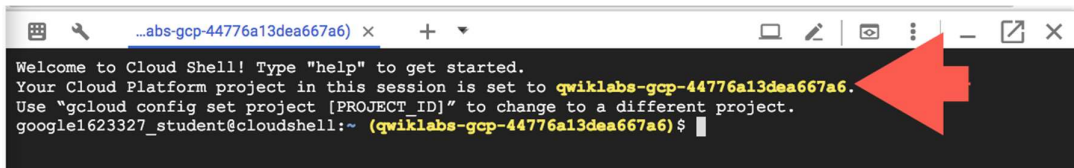
Click **Continue**.

Cloud Shell

Google Cloud Shell provides you with command-line access to your cloud resources directly from your browser. You can easily manage your projects and resources without having to install the Google Cloud SDK or other tools on your system. [Learn more.](#)

Continue

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



```
...abs-gcp-44776a13dea667a6) x + -
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6) $
```

`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

(Output)

```
Credentialed accounts:
- <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:
- google1623327_student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

```
[core]
project = <project_ID>
```


(Example output)

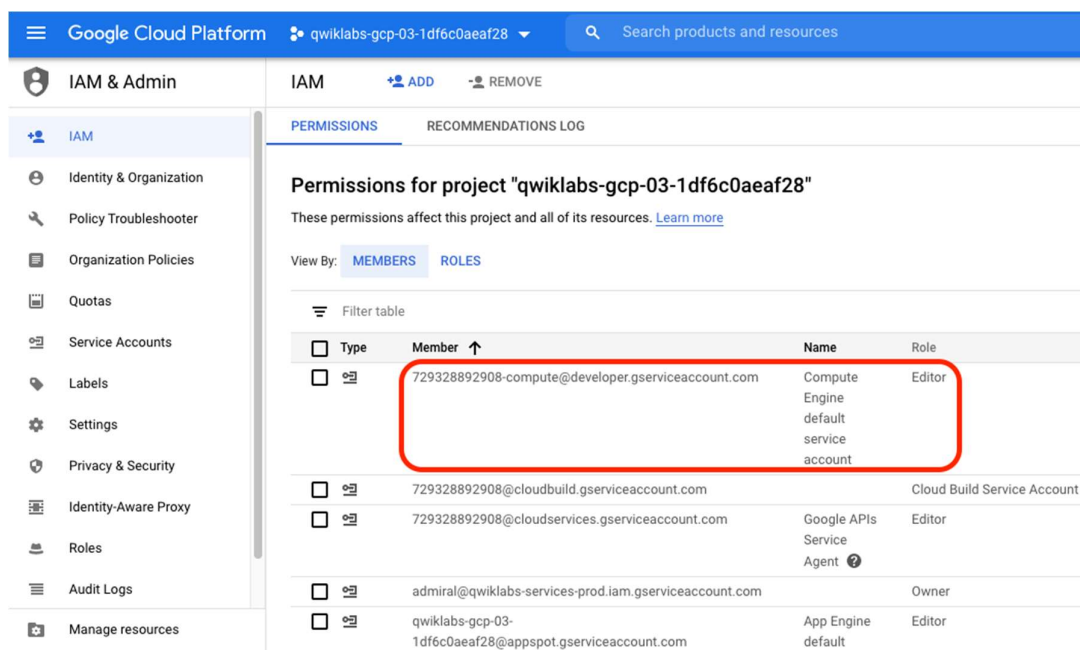
```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the [gcloud command-line tool overview](#).

Check project permissions

Before you begin your work on Google Cloud, you need to ensure that your project has the correct permissions within Identity and Access Management (IAM).

1. In the Google Cloud console, on the **Navigation menu** () , click **IAM & Admin > IAM**.
2. Confirm that the default compute Service Account `{project-number}-compute@developer.gserviceaccount.com` is present and has the `editor` role assigned. The account prefix is the project number, which you can find on **Navigation menu > Home**.



Google Cloud Platform qwiklabs-gcp-03-1df6c0aeaf28 Search products and resources

IAM & Admin

IAM +ADD -REMOVE

PERMISSIONS RECOMMENDATIONS LOG

Permissions for project "qwiklabs-gcp-03-1df6c0aeaf28"

These permissions affect this project and all of its resources. [Learn more](#)

View By: MEMBERS ROLES

Filter table

Type	Member ↑	Name	Role
<input type="checkbox"/>	729328892908-compute@developer.gserviceaccount.com	Compute Engine default service account	Editor
<input type="checkbox"/>	729328892908@cloudbuild.gserviceaccount.com		Cloud Build Service Account
<input type="checkbox"/>	729328892908@cloudservices.gserviceaccount.com	Google APIs Service Agent	Editor
<input type="checkbox"/>	admiral@qwiklabs-services-prod.iam.gserviceaccount.com		Owner
<input type="checkbox"/>	qwiklabs-gcp-03-1df6c0aeaf28@appspot.gserviceaccount.com	App Engine default	Editor

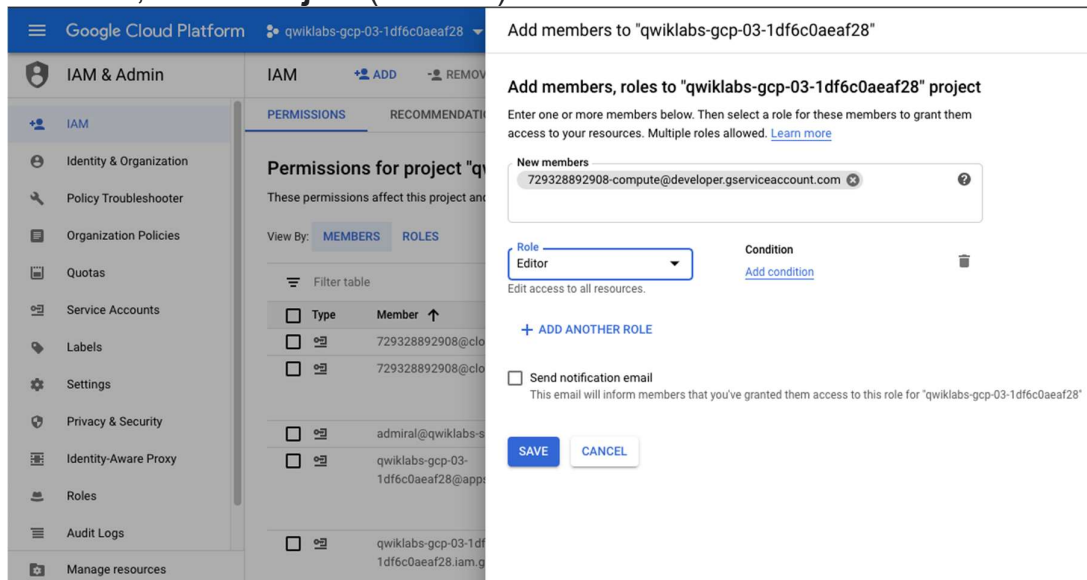
If the account is not present in IAM or does not have the `editor` role, follow the steps below to assign the required role.

- In the Google Cloud console, on the **Navigation menu**, click **Home**.
- Copy the project number (e.g. 729328892908).
- On the **Navigation menu**, click **IAM & Admin > IAM**.
- At the top of the **IAM** page, click **Add**.
- For **New members**, type:

```
{project-number}-compute@developer.gserviceaccount.com
```

Replace `{project-number}` with your project number.

- For **Role**, select **Project** (or Basic) > **Editor**. Click **Save**.



Create your lab resources

You need a virtual machine to perform the creation of the pipeline and use as your website.

1. In Cloud Shell, run the following command:

```
gcloud beta compute instances create crypto-driver \
--zone=us-central1-a \
--machine-type=n1-standard-1 \
--subnet=default \
--network-tier=PREMIUM \
--maintenance-policy=MIGRATE \
--service-account=$(gcloud iam service-accounts list --format='value(email)' --
filter="compute") \
--scopes=https://www.googleapis.com/auth/cloud-platform \
--image=debian-9-stretch-v20200618 \
--image-project=debian-cloud \
--boot-disk-size=20GB \
--boot-disk-type=pd-standard \
--boot-disk-device-name=crypto-driver
```

The compute engine service account is used with the cloud API scope. This provides the necessary permissions to create the necessary resources for your environment.

Wait for the instance to start.

Click **Check my progress** to verify the objective.

Create a virtual machine to perform the creation of the pipeline and use as your website.

Check my progress

Connect to the instance via SSH

2. In the **Cloud Platform Console**, on the **Navigation** menu, click **Compute Engine > VM Instances**.
3. For the instance called **crypto-driver**, click **SSH**.

A window will open and you will be automatically logged into the instance. You will run all commands for the remainder of this lab in the SSH window.

4. Run the following commands to install all the necessary tools (such as java, git, maven, pip, python and cloud bigtable command line tool cbt):

```
sudo -s
apt-get update -y
sudo apt install python3-pip -y
sudo pip3 install virtualenv
virtualenv -p python3 venv
source venv/bin/activate
sudo apt -y --allow-downgrades install openjdk-8-jdk git maven google-cloud-sdk=271.0.0-0 google-cloud-sdk-cbt=271.0.0-0
```

This will also install virtualenv for the Python environment. All the Python related code will be executed in the Virtualenv.

5. Now create the Bigtable resource. The first gcloud command will enable the required BigTable and dataflow API in the project. The next command will create a BigTable Cluster called “cryptorealtime-c1” with one instance called “cryptorealtime”. The instance type is Development, and therefore it will be a one node instance. And finally, using the cbt command you are creating a table called “cryptorealtime” with one column family called “market” in the BigTable instance.

```
export PROJECT=$(gcloud info --format='value(config.project)')
export ZONE=$(curl "http://metadata.google.internal/computeMetadata/v1/instance/zone" -H "Metadata-Flavor: Google"|cut -d/ -f4)
gcloud services enable bigtable.googleapis.com \
bigtableadmin.googleapis.com \
dataflow.googleapis.com \
--project=${PROJECT}

gcloud bigtable instances create cryptorealtime \
  --cluster=cryptorealtime-c1 \
  --cluster-zone=${ZONE} \
  --display-name=cryptorealtime \
  --cluster-storage-type=HDD \
  --instance-type=DEVELOPMENT
cbt -instance=cryptorealtime createtable cryptorealtime families=market
```

Click **Check my progress** to verify the objective.

Create the Bigtable instance

Check my progress

For this lab, one column family called `market` is used to simplify the schema design. For more on that you can read this [link](#).

6. Run the following command to create a bucket:

```
gsutil mb -p ${PROJECT} gs://realtimecrypto-${PROJECT}
```

This bucket will be used by the Dataflow job as a staging area for the Jar files.

Click **Check my progress** to verify the objective.

Create a Cloud Storage bucket

Check my progress

7. Now clone the application source code repository:

```
git clone https://github.com/GoogleCloudPlatform/professional-services
```

8. Run the following to build the software:

```
cd professional-services/examples/cryptorealtime
mvn clean install
```

(Output)

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 5.862 s
[INFO] Finished at: 2018-12-13T00:29:30+00:00
[INFO] Final Memory: 49M/253M
[INFO] -----
```

The code for this lab is written in Java, which needs to be compiled and packaged as a Jar file using Maven build tool. The build will take a couple of minutes, please wait until you see the BUILD SUCCESS message.

9. Once the build is finished, start the pipeline:

```
./run.sh ${PROJECT} \
cryptorealtime gs://realtimecrypto-${PROJECT}/temp \
cryptorealtime market
```

If you check the logs, don't worry about the errors you see. It is safe to ignore illegal thread pool exceptions.

Wait for the command to complete.

`run.sh` script is a wrapper to submit the crypto tracking dataflow job on Google Cloud. It takes the following arguments:

- Project_Name,
- BigTable Instance name,
- Cloud Storage bucket to use as staging area,
- BigTable Table name and

- Column family name to write the output of the pipeline.
When the pipeline starts, it will create two worker nodes. At this point, you will see two additional VMs in your project.

- Wait a couple of minutes, then run the following to observe the incoming trades by peeking into Bigtable. You can do this by using Cloud BigTable CLI tool (called cbt). If the pipeline is successfully executing, you should see new data appearing in the cryptorealtime table.

```
cbt -instance=cryptorealtime read cryptorealtime
```

You will see something like the following if the data is flowing. If you see nothing, please wait for a minute and try again.

```

=====
BTC/EUR#bitStamp#1544239702209#63201787255
market:delta @ 2018/12/08-03:28:22.209000
"209"
market:exchangeTime @ 2018/12/08-03:28:22.209000
"1544239702000"
market:market @ 2018/12/08-03:28:22.209000
"bitStamp"
market:orderType @ 2018/12/08-03:28:22.209000
"ASK"
market:price @ 2018/12/08-03:28:22.209000
"3014.8499999999999"
market:volume @ 2018/12/08-03:28:22.209000
"0.072261450000000005"
=====

```

Click **Check my progress** to verify the objective.

Run the Dataflow pipeline

Check my progress

Examine the Dataflow pipeline

11. In the **Cloud Platform Console**, on the **Navigation menu**, click **Dataflow**.
12. Click the name of the existing pipeline.

You should see the status as Running for the listed jobs.

The screenshot displays the Google Cloud Dataflow console interface. The top navigation bar includes a back arrow, 'Job details', a link to 'BACK TO OLD JOB PAGE', a red status indicator with the number '5', a 'STOP' button, and a 'MAX TIME' dropdown menu. Below the navigation bar, there are two tabs: 'JOB GRAPH' (selected) and 'JOB METRICS'. The 'JOB GRAPH' tab shows a pipeline with four parallel stages, each represented by a box with a green status icon and a dropdown arrow. The stages are: 'bitfinex3' (Running, 3 min 41 sec), 'XRP-USD Mut' (0 elements/s, 0 sec), 'bitfinex2' (Running, 3 min 3 sec), 'ETH-USD Mut' (Running, 0 sec), 'bitfinex' (Running, 3 min 40 sec), 'BTC-USD Mut' (Running, 1 sec), 'bitfinex4' (Running, 3 min 39 sec), and 'BCH-USD Mut' (Running, 0 sec). Below these, there are four more boxes for the output stages: 'XRP-USD' (0 elements/s, 0 sec), 'ETH-USD' (Running, 0 sec), 'BTC-USD' (Running, 2 sec), and 'BCH-USD' (Running, 0 sec). On the right side, the 'Job info' panel is visible, showing details for the job 'runthepipeline-root-0709101743-61b48a96'. The job ID is '2020-07-09_03_17_54-3369472305600000305'. The job type is 'Streaming', the job status is 'Running', and the SDK version is 'Apache Beam SDK for Java 2.8.0'. A warning message indicates that this version of the SDK is deprecated and will eventually be no longer supported, with a 'Learn more' link. The job region is 'us-central1', the worker location is 'us-central1-a', the start time is 'July 9, 2020 at 3:47:55 PM GMT+5', the elapsed time is '6 min 27 sec', and the encryption type is 'Google-managed key'.

Job name	runthepipeline-root-0709101743-61b48a96
Job ID	2020-07-09_03_17_54-3369472305600000305
Job type	Streaming
Job status	Running
SDK version	Apache Beam SDK for Java 2.8.0 <small>⚠ This version of the SDK is deprecated and will eventually be no longer supported. Learn more</small>
Job region	us-central1
Worker location	us-central1-a
Start time	July 9, 2020 at 3:47:55 PM GMT+5
Elapsed time	6 min 27 sec
Encryption type	Google-managed key

Ignore the javalang error type and move ahead in the lab.

Visualizing the data

You will configure the environment and run the python Flask frontend server visualization in these next steps.

13. Go back to the SSH session and run the following command to open firewall port 5000 for visualization:

```
gcloud compute --project=${PROJECT} firewall-rules create crypto-dashboard \
--direction=INGRESS \
--priority=1000 \
--network=default \
--action=ALLOW \
--rules=tcp:5000 \
--source-ranges=0.0.0.0/0 \
--target-tags=crypto-console \
--description="Open port 5000 for crypto visualization tutorial"
```

14. Now link the VM with the firewall rule:

```
gcloud compute instances add-tags crypto-driver --tags="crypto-console" --zone=${ZONE}
```

Click **Check my progress** to verify the objective.

Open firewall port 5000 for visualization

Check my progress

15. Next, navigate to the `frontend` directory:

```
cd frontend/
pip install -r requirements.txt

python app.py ${PROJECT} cryptorealtime cryptorealtime market
```

`app.py` is a Python application to visualize the Crypto currency data stored in the BigTable table “cryptorealtime”.

16. Open another SSH session and run the following command to find your external IP address for the `crypto-driver` instance:

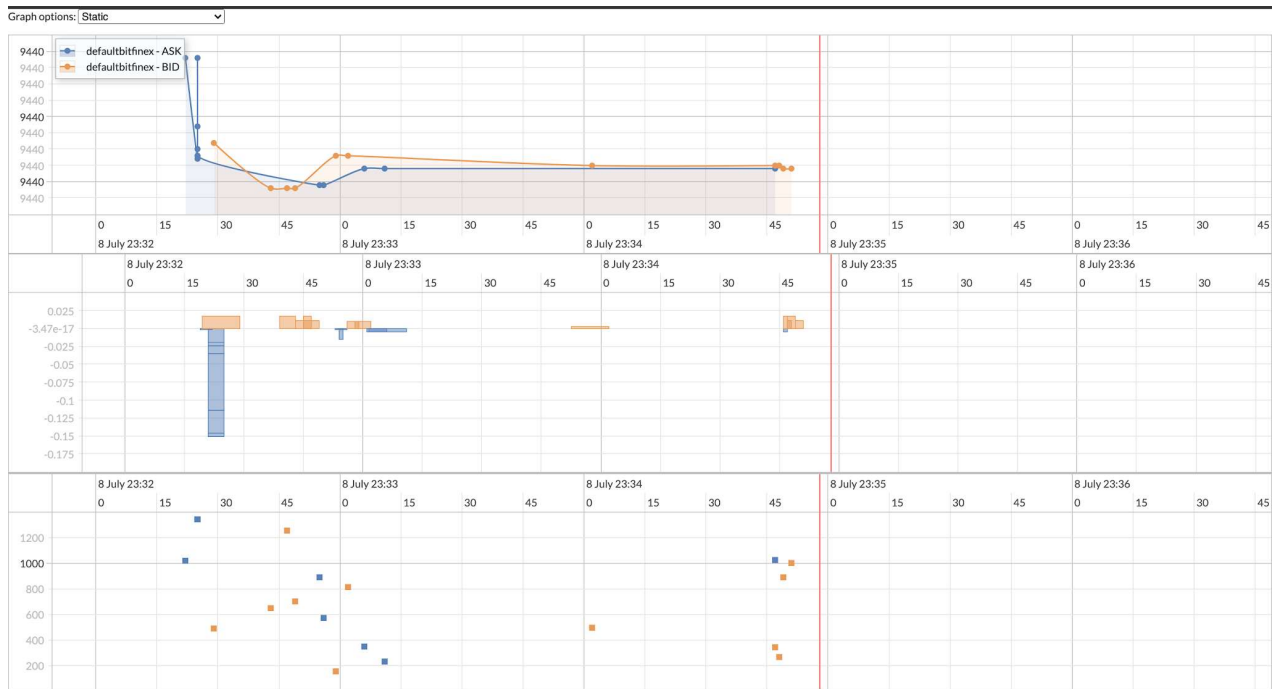
```
gcloud compute instances list --format='value(EXTERNAL_IP)' --filter="name:crypto-driver"
```

Copy the EXTERNAL IP address displayed. You will use for the next command.

17. Open a new tab in your web browser and use the following URL to see the visualization, replacing `<external-ip>` with the IP address returned from the previous command:

```
http://<external-ip>:5000/stream
```

You should see the visualization of aggregated ASK/BID pair on several exchanges (without predictor part).



You have created a real-time "periscope" multi-exchange observer.

Clean up

When you end this lab, Qwiklabs will delete the resources you used. In your own environment, it is useful to know how to save costs by cleaning up your unused, or no longer required, resources.

You can stop the pipeline in either the Console or in the SSH session.

- Console: Return to Dataflow page, click on the name of your job and click **Stop**. Select **Cancel**, then **Stop Job**.

Stop job

☒ Cancel

Dataflow will immediately stop your job and abort all data ingestion and processing. Any buffered data may be lost.

☐ Drain

Dataflow will cease all data ingestion, but will attempt to finish processing any remaining buffered data. Pipeline resources will be maintained until buffered data has finished processing and any pending output has finished writing.

[Read more about stopping Dataflow jobs](#)

DO NOTHING

STOP JOB

- Inside the first SSH session run:

```
gcloud dataflow jobs cancel \  
$(gcloud dataflow jobs list \  
--format='value(id)' \  
--filter="name:runthepipeline*" \  
--region="us-central1")
```

This will takes few minutes with either method.

Return to the command line in the SSH session by pressing **Ctrl + c**.

Inside the SSH session run the following commands to empty and delete the bucket:

```
gsutil -m rm -r gs://realtimecrypto-${PROJECT}/*  
gsutil rb gs://realtimecrypto-${PROJECT}
```

Inside the SSH session run the following command to delete the Bigtable instance:

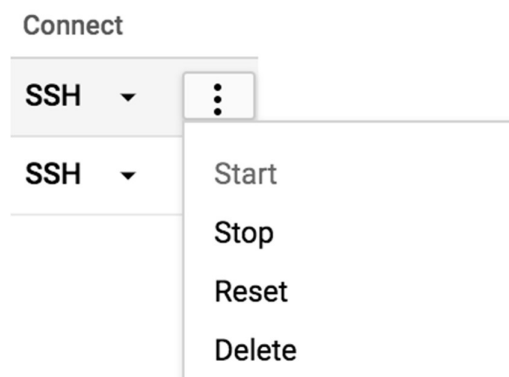
```
gcloud bigtable instances delete cryptorealtime
```

If prompted for, Do you want to continue (Y/n), press **Y**.

Close the SSH console.

In the **Cloud Platform Console**, on the **Navigation menu**, click **Compute Engine > VM Instances**.

Check the box next to the **crypto-driver** instance then click **Delete**, then **Delete** again to confirm your action.



Congratulations!

You have learned how to do the following:

- Set up and configure a pipeline for ingesting real-time time-series data from various crypto exchanges.
- Design a suitable data model, which facilitates querying and graphing at scale.
- Set up and deploy the proposed architecture using Google Cloud.

You established a connection to multiple exchanges, subscribed to their trade feed, then extracted and transformed these trades into a flexible format to be stored in Bigtable to be graphed and analyzed.

Next steps/ Read more

This lab is based on [this Medium article](#) by Ivo Galic.

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Tested October 13, 2020

Manual Last Updated October 13, 2020

Copyright 2021 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.