



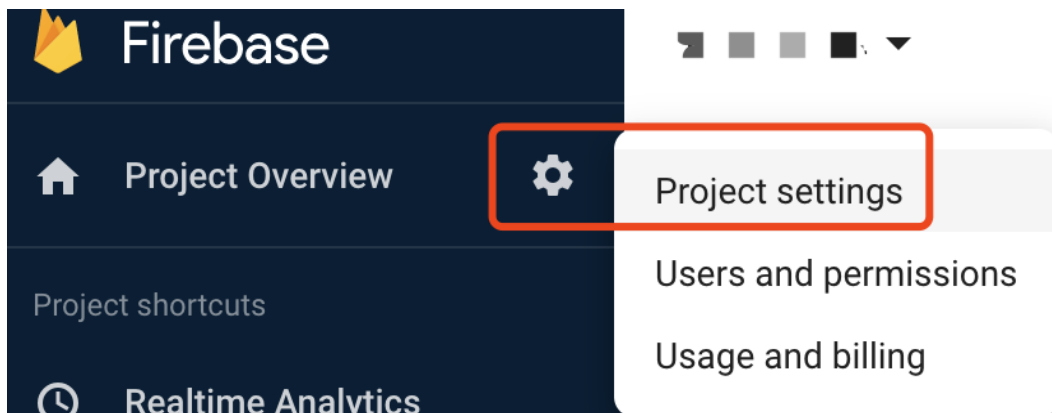
# Firestore & Bigquery

## Quick Start Manual

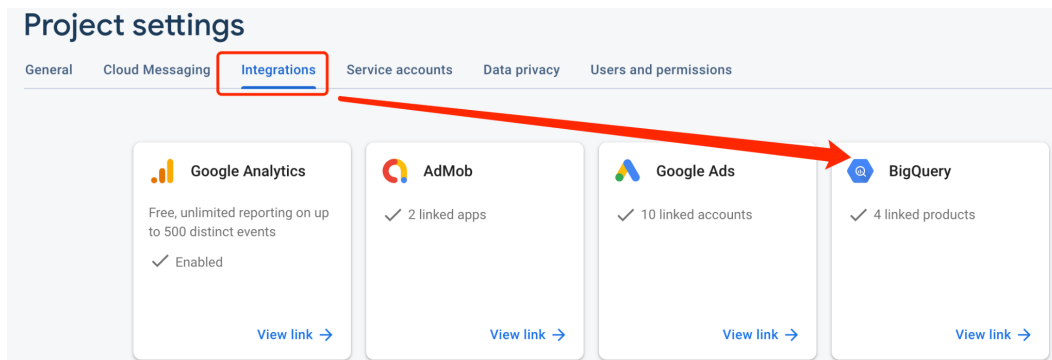
2023.04.25

### Turn on the data integration

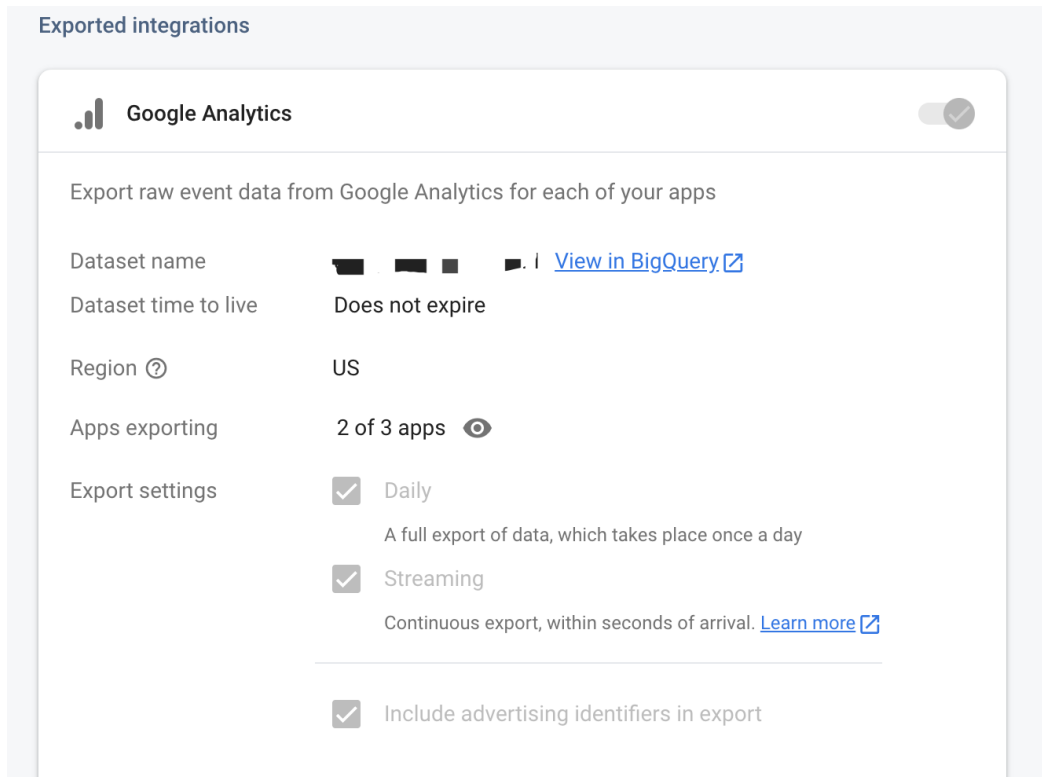
1. Go to the [Firestore console](#).
2. Click on the **Project settings** icon (⚙️).



3. Click on the **Integrations** tab.



4. Click on the **BigQuery** card.
5. Click on the **Link** button.

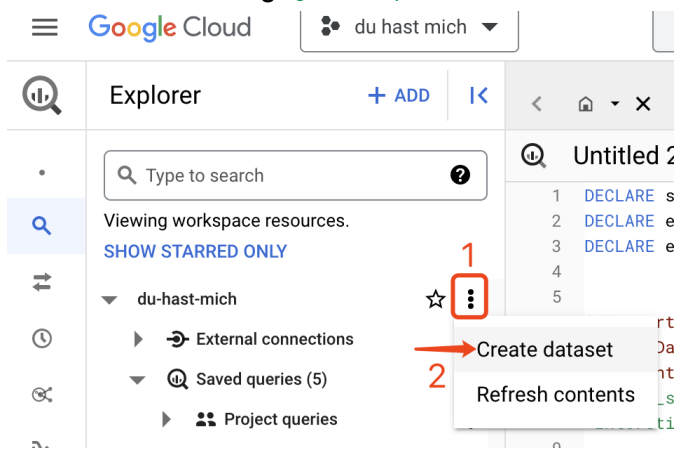


## Analyze the data in Bigquery

Before we start, the full SQL code repository can be found [here](#). Please note, scripts are querying a public dataset. You will need to change the targeting *table/dataset* to see your own data.

(Optional) Create a view pointing to sample/public dataset

1. Go to [Bigquery console](#)
2. Create a dataset, e.g. `ga4_sample_data`



## Create dataset

**Project ID**  
du-hast-mich [CHANGE](#)

**Dataset ID \***  
ga4\_sample\_data  
Letters, numbers, and underscores allowed

**Location type ?**

☐ Region  
Specifying a region provides dataset colocation with other GCP services

☒ Multi-region  
Letting BigQuery select a region within a group of regions provides higher quota limits

**Multi-region \***  
US (multiple regions in United States) ▼

**Default table expiration**

☐ Enable table expiration ?

Default maximum table age  Days

**Advanced options** ▼

[CREATE DATASET](#) [CANCEL](#)

3. Create a View pointing to the sample/public dataset  
Run the following SQL in BQ workbench

```
CREATE VIEW ga4_sample_data.events AS
select * from `bigquery-public-data.ga4_obfuscated_sample_ecommerce.events_*
```

Once done, you should be able to see the following view has been created

The screenshot shows the Google Cloud BigQuery interface. On the left, the 'Workspace' pane displays a tree view of resources for the 'du-hast-mich' project. The 'ga4\_sample\_data' dataset is expanded, and the 'events' view is highlighted with a red box. The main pane shows the 'events' view details, including a 'LINEAGE' tab. The lineage diagram shows a flow from 'events\_\*' to 'events'. The 'events' view is listed as 'ga4\_sample\_data.events'.

## Explore user retention

```
WITH analytics_data AS (  
  SELECT user_pseudo_id, event_timestamp, event_name,  
         UNIX_MICROS(TIMESTAMP("2020-11-01 00:00:00", "+8:00")) AS start_day,  
         3600*1000*1000*24*7 AS one_week_micros  
  FROM `bigquery-public-data.ga4_obfuscated_sample_ecommerce.events_*`  
)
```

```
SELECT week_0_cohort / week_0_cohort AS week_0_pct,  
       week_1_cohort / week_0_cohort AS week_1_pct,  
       week_2_cohort / week_0_cohort AS week_2_pct,  
       week_3_cohort / week_0_cohort AS week_3_pct  
FROM (  
  WITH week_3_users AS (  
    SELECT DISTINCT user_pseudo_id  
    FROM analytics_data  
    WHERE event_timestamp BETWEEN start_day+(3*one_week_micros) AND  
start_day+(4*one_week_micros)  
  ),  
  week_2_users AS (  
    SELECT DISTINCT user_pseudo_id  
    FROM analytics_data  
    WHERE event_timestamp BETWEEN start_day+(2*one_week_micros) AND  
start_day+(3*one_week_micros)  
  ),  
  week_1_users AS (  
    SELECT DISTINCT user_pseudo_id  
    FROM analytics_data  
    WHERE event_timestamp BETWEEN start_day+(1*one_week_micros) AND  
start_day+(2*one_week_micros)  
  ),  
  week_0_users AS (  
    SELECT DISTINCT user_pseudo_id  
    FROM analytics_data  
    WHERE event_name = 'first_visit'
```

```

        AND event_timestamp BETWEEN start_day AND start_day+(1*one_week_micros)
    )
SELECT
    (SELECT count(*)
     FROM week_0_users) AS week_0_cohort,
    (SELECT count(*)
     FROM week_1_users
     JOIN week_0_users USING (user_pseudo_id)) AS week_1_cohort,
    (SELECT count(*)
     FROM week_2_users
     JOIN week_0_users USING (user_pseudo_id)) AS week_2_cohort,
    (SELECT count(*)
     FROM week_3_users
     JOIN week_0_users USING (user_pseudo_id)) AS week_3_cohort
)

```

#### TODOs:

1. Try to build a 7-day retention query
2. Explore your own data against different app versions and/or specific devices

## Define your own closed funnels

```

SELECT count(distinct funnel_1) as funnel_1_total, count(distinct funnel_2) as
funnel_2_total from (
    SELECT
        IF (event_name = "session_start", user_pseudo_id, NULL) AS funnel_1,
        IF (event_name = "session_start" AND next_event = "purchase" AND next_timestamp -
event_timestamp < 20 * 60 * 1000 * 1000, user_pseudo_id, NULL) AS funnel_2
    FROM (
        SELECT event_name, user_pseudo_id , event_timestamp,
        LEAD(event_name, 1) OVER (PARTITION BY user_pseudo_id ORDER BY event_timestamp) AS
next_event,
        LEAD(event_timestamp, 1) OVER (PARTITION BY user_pseudo_id ORDER BY
event_timestamp) AS next_timestamp
        FROM `bigquery-public-data.ga4_obfuscated_sample_ecommerce.events_*`
        WHERE (event_name = "session_start" OR event_name = "purchase")
    )
)

```

```
ORDER BY 2,3
)
)
```

TODOs:

1. Build a more complex funnel, e.g. more steps
2. Compare the same funnel against different criterias, e.g. geo-location, devices etc.

## Visualized the data in Looker

### Transform data in BigQuery

The reason for the conversion is that we need to convert the data exported from firebase into the [schema](#) needed for the looker ml model, either as a table or as a view, here is the example:

1. Create a sql file called "create\_events\_view.sql" and write the following sql (please remember to change the project id and dataset)

```
CREATE VIEW {project_id}.{dataset}.v_gaming_events AS
SELECT
GENERATE_UUID() as unique_event_id,
TIMESTAMP_MICROS(event_timestamp) as event,
event_name,
event_bundle_sequence_id,
safe_cast(user_pseudo_id as STRING) as user_id,
TIMESTAMP_MICROS(user_first_touch_timestamp) as user_first_seen,
platform as device_platform,
device.mobile_brand_name as device_brand,
device.mobile_model_name as device_model,
device.operating_system_version as device_os_version,
device.language as device_language,
geo.continent as continent,
geo.region as region,
geo.country as country,
app_info.install_source as install_source,
(event_timestamp=user_first_touch_timestamp) as is_first_seen,
ecommerce.purchase_revenue as iap_revenue,
user_ltv,
( select
```

```

x
from UNNEST(ARRAY<STRUCT<x INT64, y STRING>>[(event_timestamp, event_name)])
where y='session_start') as session_start,
(CASE
  WHEN event_name!='session_start' THEN null
  ELSE TIMESTAMP_MICROS(LEAD(event_timestamp, 1) OVER (
    PARTITION BY (select z
      from UNNEST(ARRAY<STRUCT<x INT64, y STRING, z
STRING>>[(event_timestamp, event_name, user_pseudo_id)])
      where y='session_start') ORDER BY event_timestamp)
    )
  END
) as next_session_start,
( select value.float_value
from unnest(event_params)
where key='@ga_ad_revenue' limit 1) as ad_revenue,
( select value.float_value
from unnest(event_params)
where key='@ga_install_cost' limit 1) as install_cost,
( select value.int_value
from unnest(event_params)
where key='@ga_gems_earned' limit 1) as gems_earned,
( select value.string_value
from unnest(event_params)
where key='@ga_campaign_name' limit 1) as campaign_name,
( select value.string_value
from unnest(event_params)
where key='@ga_ad_network' limit 1) as ad_network,
( select value.string_value
from unnest(event_params)
where key='@ga_campaign_id' limit 1) as campaign_id,
( select value.string_value
from unnest(event_params)
where key='@ga_game_name' limit 1) as game_name,
( select value.string_value
from unnest(event_params)

```

```

    where key='@ga_game_version' limit 1) as game_version,
  ( select value.int_value
    from unnest(event_params)
    where key='@ga_player_level' limit 1) as player_level,
  ( select value.int_value
    from unnest(event_params)
    where key='@ga_session_number' limit 1) as player_session_sequence,
  ( select safe_cast(value.int_value as STRING)
    from unnest(event_params)
    where key='@ga_session_id' limit 1) as unique_session_id
FROM
`bigquery-public-data.ga4_obfuscated_sample_ecommerce.events_*` as e

```

## 2. Create a view point for Looker

Run the following command in console:

```

bq query --use_legacy_sql=false \
  --parameter=ga_ad_revenue::ad_revenue \
  --parameter=ga_install_cost::install_cost \
  --parameter=ga_gems_earned::gems_earned \
  --parameter=ga_campaign_name::campaign_name \
  --parameter=ga_campaign_id::campaign_id \
  --parameter=ga_ad_network::link_url \
  --parameter=ga_game_name::game_name \
  --parameter=ga_game_version::game_version \
  --parameter=ga_player_level::player_level \
  --parameter=ga_session_id::ga_session_id \
  --parameter=ga_session_number::ga_session_number < create_events_view.sql

```

In the SQL above, we'll need to adjust the following fields to match the actual event parameters being passed

3. (Optional) If your data is stored in multiple locations, you can change the way some fields are ingested, for example, querying the field "install cost" by multiple tables using a join table.

## Create new project in Looker

1. Create a new connection to connect your dataset in BigQuery.





# Connect your database to Looker

Fill out the connection details. The majority of these settings are common to most database dialects. [Learn more](#)

**Name \***  
looker-demo-gaming

**Dialect \***  
Google BigQuery Standard SQL ▼

**Billing Project ID \*** 

**Dataset \*** 

**Authentication \***  

Service Account OAuth

Upload service JSON or P12 file

Upload File

2. Create a new project, enter a name for it and select Blank Project.



Browse ▼

Explore

## New Project

**Project Name**  
gaming\_project

May contain lowercase letters, numbers, underscores, and dashes. Other characters will be lowercased or replaced with "\_".

**Starting Point**  

☐ Generate Model from Database Schema

☐ Generate from SQL

☐ Clone Public Git Repository

☒ Blank Project

Create Project

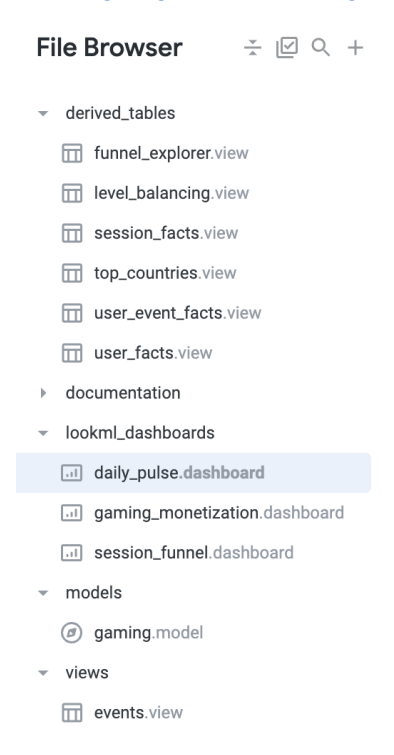
## Import model file

Import the following files in the folders as shown below:

- models:  
gaming.model ( need to change the “connection” )
- views:  
events.view ( need to change the “sql\_table\_name” )
- derived\_tables:  
funnel\_explorer.view  
level\_balancing.view  
session\_facts.view  
top\_countries.view  
user\_event\_facts.view  
user\_facts.view
- lookml\_dashboards:  
daily\_pulse.dashboard  
gaming\_monetization.dashboard  
session\_funnel.dashboard

The full code repository can be found here:

<https://gist.github.com/ping-coder/3ec7277d0c2714c8cf034b851755d7ea>



## Explore data



### ← Explore

gaming

#### ▼ Gaming

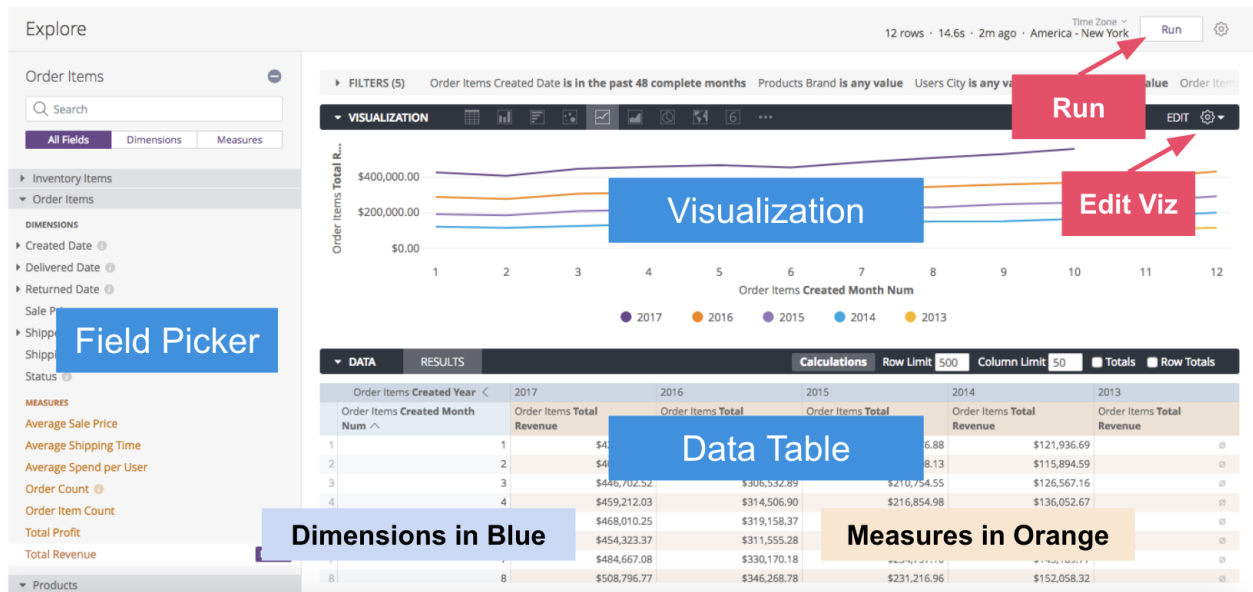
Events

Funnel Explorer

Level Balancing

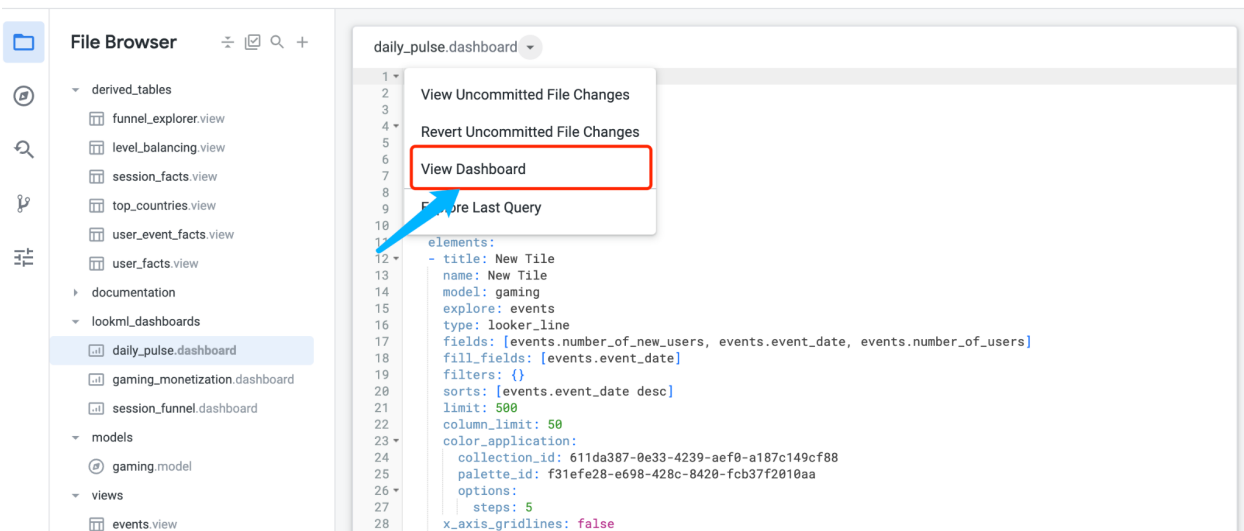
Sessions and Users

1. Go to Looker Console
2. Click on Explore menu
3. Go to Gaming Events Explore

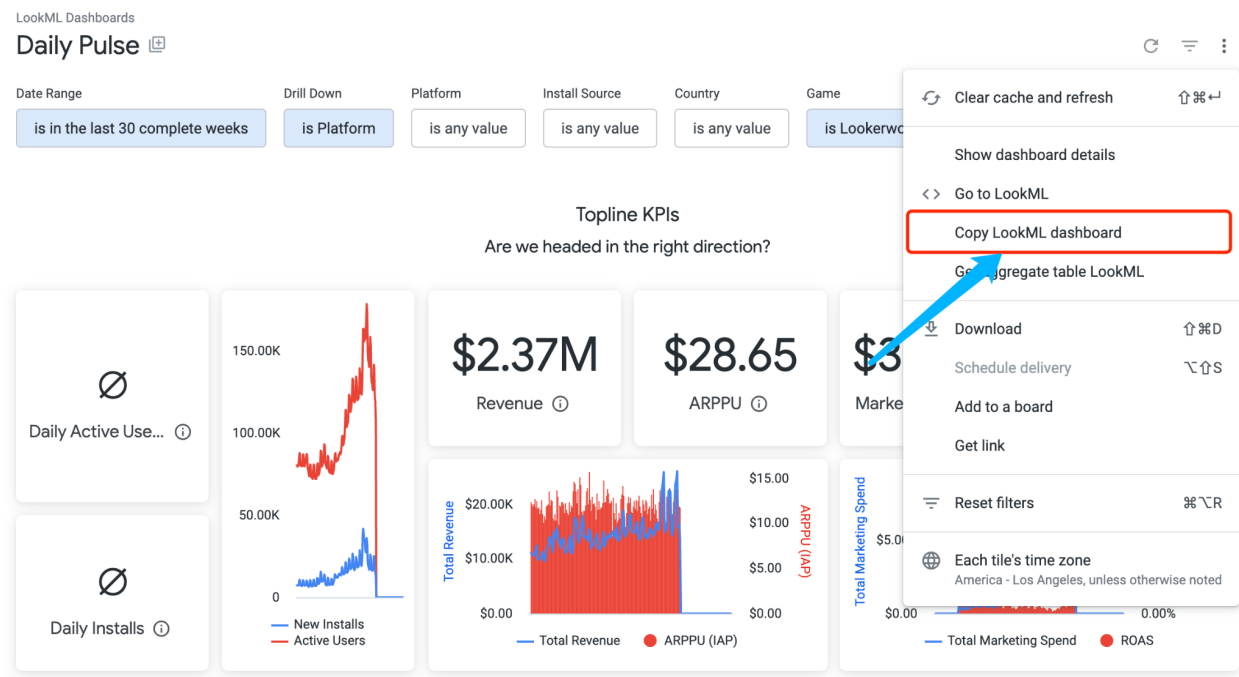


## Create and edit dashboard

1. Goto the console of LookML, select daily\_pulse.dashboard and click on "View Dashboard"



2. Goto the dashboard of Daily Pulse, and click on “Copy lookML dashboard”



3. Type the title, and click on “Copy”

## Copy Daily Pulse

**i** The copied dashboard will not be a LookML dashboard.

Title \*

Daily Pulse

Folder \*

▼ Select a folder

My folder selected

► Shared

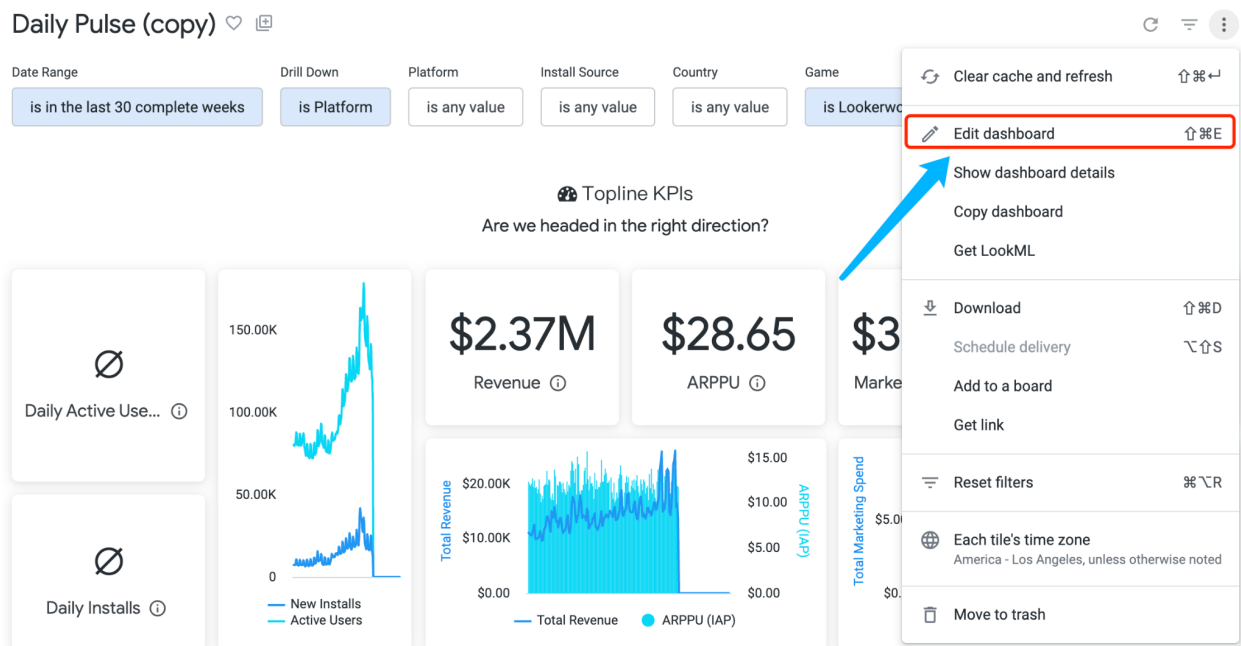
► Users

☐ Preserve locale keys

Cancel

Copy

4. Goto My folder, select the dashboard named “Daily Pulse”, and click on “Edit dashboard”



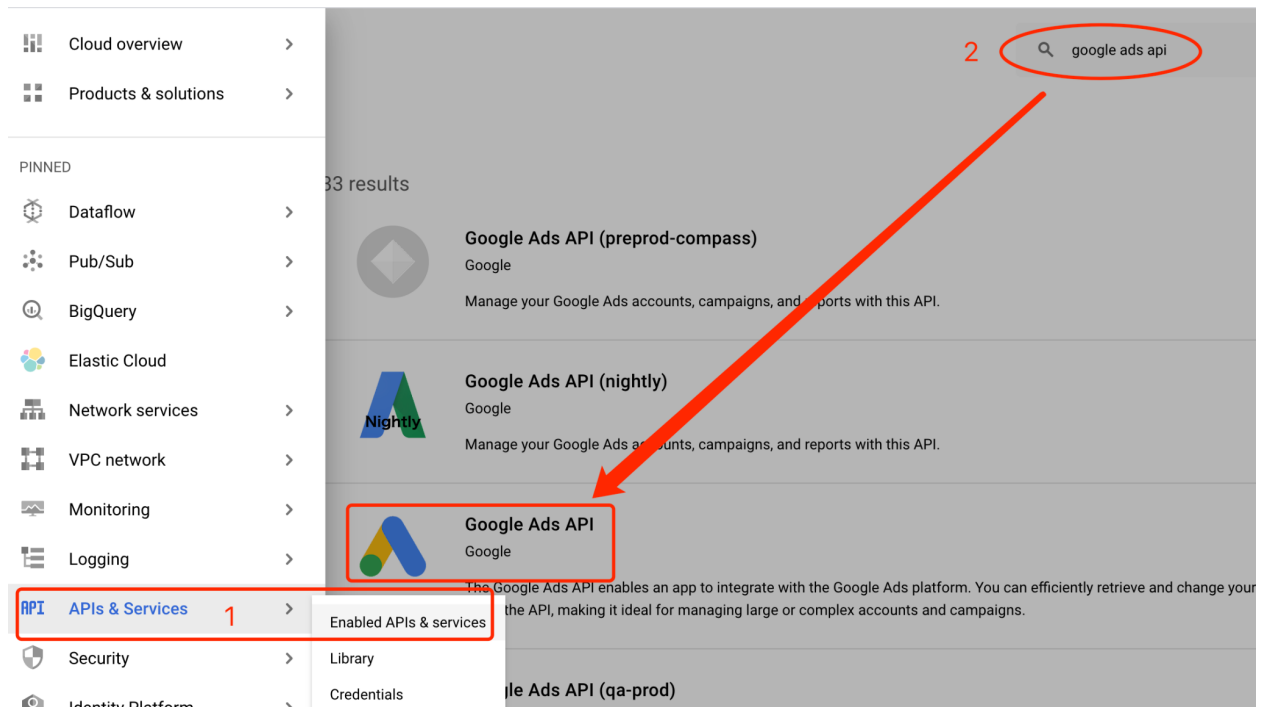
Finally, the dashboard tests are complete, and we need to finish configuring the git in the lookml page, and commit and publish to production.

## Turn on Ads related APIs from Cloud Console

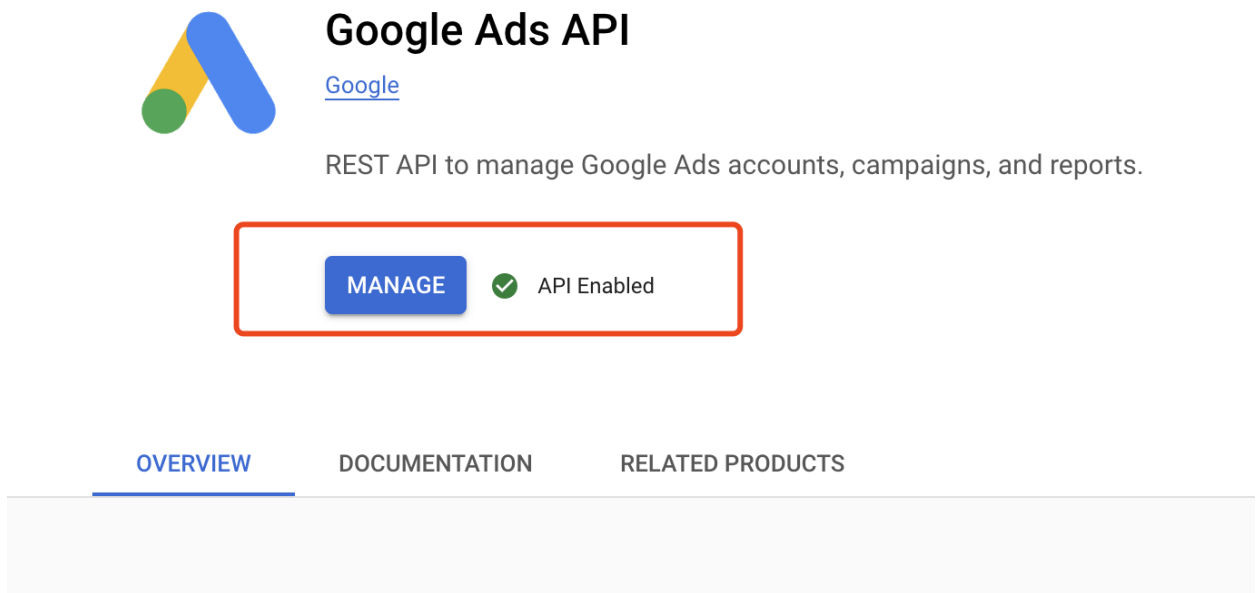
### Google Ads API

1. Go to [cloud console](#)
2. From the left navigation menu select API & Services

### 3. Search “Google Ads API” from the mid-top text box

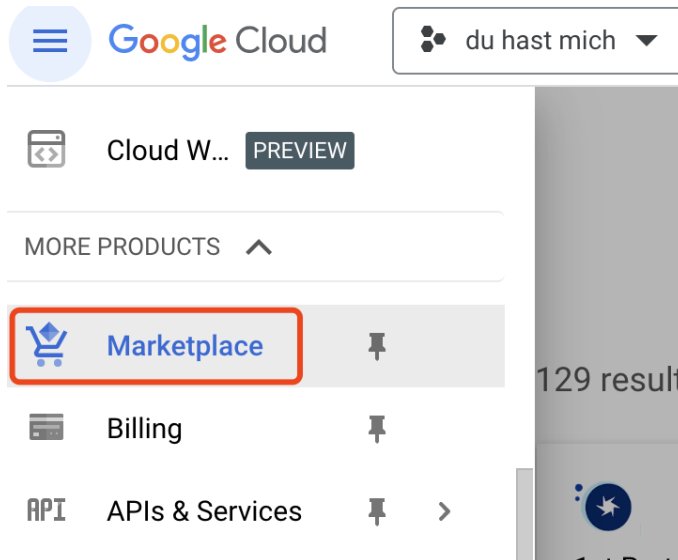


### 4. Click the “Enable” button to turn on the API

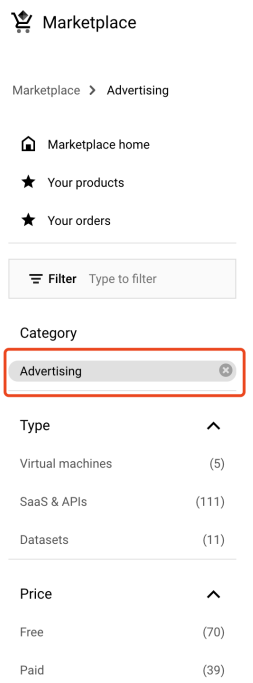


## Full list of APIs

### 1. From the navigation menu select “Marketplace”



## 2. Set the filter to “Advertising”



## 3. A full list should be showing up

Marketplace

Marketplace home

Filter

Type to filter

Category

Advertising

Type

Virtual machines

(8)

SaaS & APIs

(11)

Datasets

(11)

Price

Free

(70)

Paid

(39)

129 results

1st Party Data Platform

Flywheel Software

1st Party Data Platform on BigQuery

Type: SaaS & APIs

Ad Exchange Buyer API II

Google

Build applications that interact directly with the DoubleClick Ad Exchange platform.

Type: SaaS & APIs

Ad Exchange Seller API

Google

Programmatically access Ad Exchange publisher inventory.

Type: SaaS & APIs

AdMob API

Google

The Google AdMob API lets you programmatically get reports on earnings.

Type: SaaS & APIs

AdSense Host API

Google

The AdSense Host API gives AdSense Hosts access to report generation, ad code generation...

Type: SaaS & APIs

AdSense Management API

Google

The AdSense Management API allows publishers to access their inventory and run earnings and...

Type: SaaS & APIs

Airship Mobile Engagement Platform

Airship

Compose and deliver personalized cross-channel messages

Type: SaaS & APIs

Apache Kafka Server on CentOS 8 Server

Cloud Infrastructure Services

Apache Kafka is an open-source distributed event streaming platf

Type: Virtual machines

Argentina Real Estate Listings

Properati

Monthly property listing data for Argentina since 2016

Type: Datasets

Bing Ads by Fivetran

Fivetran

Effortlessly replicate all your Bing Ads data into BigQuery.

Type: SaaS & APIs

Bluecore

Bluecore

Create and activate personalized retail marketing campaigns

Type: SaaS & APIs

Brazil Real Estate Listings

Properati

Monthly property listing data for Brazil since 2016

Type: Datasets

Campaign Manager 360 API

Google

Manage your Campaign Manager 360 ad campaigns and reports

Type: SaaS & APIs

Chile Real Estate Listings

Properati

Monthly property listing data for Chile since 2016

Type: Datasets

Chrome Signage

Chrome Signage

Free Digital Signage Trial

Type: Datasets

Columbia Real Estate Listings

Properati

Monthly property listing data for Columbia since 2016

Type: Datasets

Content API for Shopping

Google

Manage your product listings and accounts for Google Shopping

Type: SaaS & APIs

DoubleClick Bid Manager API

Google

Manage your DoubleClick Bid Manager ad campaigns and reports.

Type: SaaS & APIs

DoubleClick Campaign Manager (DCM) Transfers

Google

Move DoubleClick Campaign Manager reporting data to BigQuery

Type: SaaS & APIs

Facebook Ad Insights Connector by Fivetran

Fivetran

Effortlessly replicate all your Facebook Ads data into BigQuery

Type: SaaS & APIs

FCC Political Ads

Federal Communications Commission

Public inspection files of political ad sales

Type: Datasets

Glimr Geo Intelligence

Glimr

Geo intelligence with scale

Type: Datasets

Google Ad Manager Transfers

Google

Move Google Ad Manager reporting data to BigQuery

Type: SaaS & APIs

Google Ads API

Google

REST API to manage Google Ads accounts, campaigns, and reports.

Type: SaaS & APIs

Google Ads Transfers

Google

Load Google Ads reporting data into BigQuery

Type: SaaS & APIs

Google Analytics Reporting API

Google

Access report data in Google Analytics.

Type: SaaS & APIs

Google Analytics Sample

Obfuscated Google Analytics 360 data

Twelve months of obfuscated Google Analytics 360 data

Type: Datasets

Google Merchant Center Transfers

Google

Load Google Merchant Center reporting data into BigQuery

Type: SaaS & APIs

Google Partners API

Google

Lets advertisers search certified companies and create contact leads with them, and also audits...

Type: SaaS & APIs

LinkedIn Ad Analytics Connector by Fivetran

Fivetran

Effortlessly replicate all your LinkedIn Ads data into BigQuery.

Type: SaaS & APIs

Local Services API

Google

REST API to manage Local Services Ads accounts and reports.

Type: SaaS & APIs

Mexico Real Estate Listings

Properati

Monthly property listing data for Mexico since 2016

Type: Datasets

## Appendix

[Link Google Ads to Firebase](#)  
[Service Account for Accessing Cloud Services/APIs](#)