# 利用Dataflow SQL进行湖仓一体实时流式数据分析

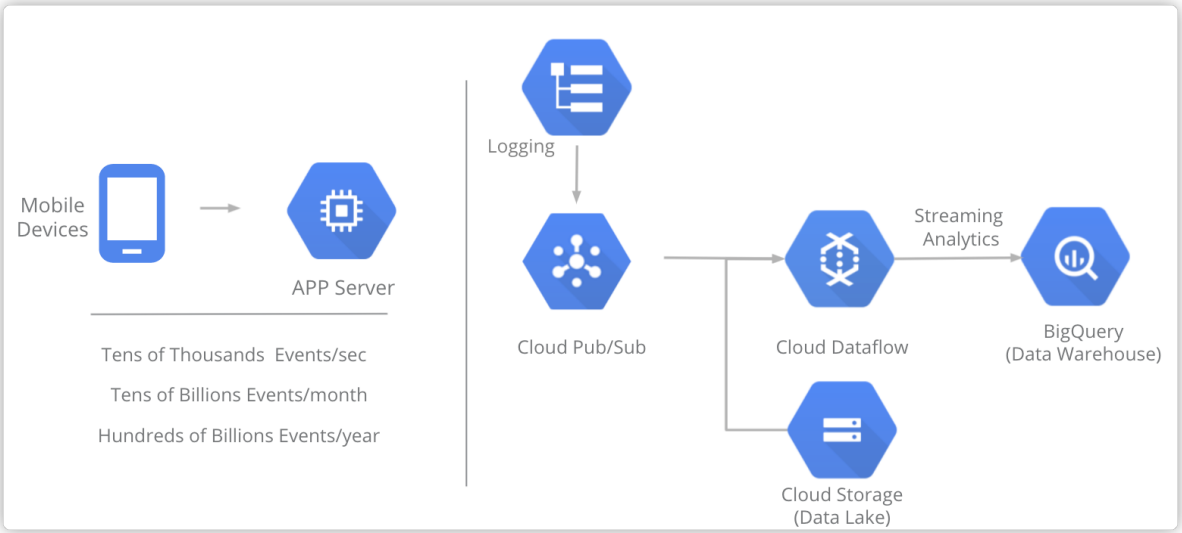
Author: jackylin@

方案介绍：

以下方案主要介绍如何利用GCP上各服务，包括Logging、Pub/Sub、Dataflow，Storage及Bigquery等服务，构建湖仓一体流式数据分析处理的过程。本文以Demo的形式，展示如何构建数据处理Pipeline。

方案架构:

APP Server产生大量的实时事件日志，通过Google Cloud Logging Agent能有效的把日志保存到Cloud Logging服务。Logging服务既能直接把全部内容流式插入到Bigquery，同时也能利用Cloud Pub/Sub服务把实时日志抽取出来，再通过Cloud Dataflow服务，结合存储在Cloud Storage数据湖上的数据，进行流式分析处理，并最终把结果存放到数据仓库Bigquery里面。




Demo 实现步骤：

以下步骤大部分在CLI环境下执行，请在操作界面先配置环境变量：

```
export GOOGLE_APPLICATION_CREDENTIALS="[your-service-account-key-path]"
export PROJECT_ID="[your-project-id]"
export REGION="[your-prefered-region]"
export BUCKET="[your-gcs-bucket]"
```

## 一、在App Server端安装Cloud Logging Agent并模拟日志产生

备注：App Server既可以是GCP的VM，也可以是其他云，或者本地数据中心的服务器。

1. 安装过程略，具体请参考以下文档：
   https://cloud.google.com/logging/docs/agent/logging/installation

2. 配置Logging Agent
自定义的事件日志保存到/tmp/test-structured-log.log 文件，通过以下命令创建自定义结构化（JSON）的日志配置文件

注意:配置完后，需要重启agent:

ssh -i ~/.ssh/google_compute_engine eugeneyu@35.236.86.47
dev-box

$ sudo service google-fluentd restart

```
sudo tee /etc/google-fluentd/config.d/test-structured-log.conf <<EOF
<source>
   @type tail
   <parse>
       # 'json' indicates the log is structured (JSON).
       @type json
   </parse>
   # The path of the log file.
   path /tmp/test-structured-log.log
   # The path of the position file that records where in the log file
   # we have processed already. This is useful when the agent
   # restarts.
   pos_file /var/lib/google-fluentd/pos/test-structured-log.pos
   read_from_head true
   # The log tag for this log input.
   tag structured-log
</source>
EOF
```

3. 编写简单的Shell命令，生产事件日志，并保存到/tmp/test-structured-log.log文件
以下例子为每秒产生一条JSON形式的日志记录，内容为code、message、event_time、player组成。

```
#!/bin/sh

player_list=(Jacky Lucy Bob Jerry Michael John Eric Leo)

while true;
do
a=$[RANDOM%7+1];
player=${player_list[$a]};
echo '{"code": "structured-log-code", "message": "This is a log from the log
file, code: '$RANDOM'", "event_time": "'$(date "+%Y-%m-%d %H:%M:%S")'",
"player": "'$player'"}'  >> /tmp/test-structured-log.log ;
echo "insert completed, player: $player " ;
```

```
sleep 1;
done
```

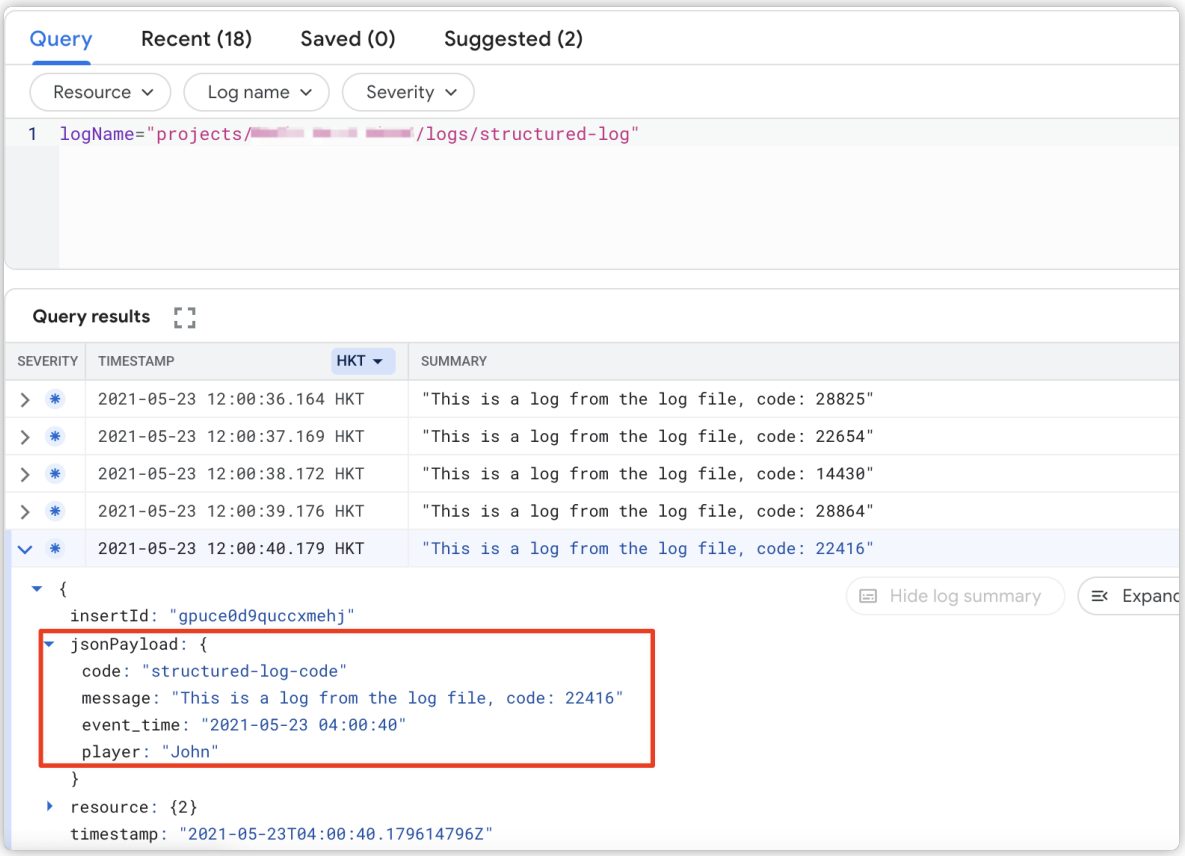
4. 执行脚本后能看到如下输出：

```
insert completed, player: Jerry
insert completed, player: Lucy
insert completed, player: Bob
insert completed, player: Jerry
insert completed, player: Leo
insert completed, player: John
...
```

5. 在/tmp/test-structured-log.log保存如下记录，并每一秒插入一条新记录：

```
{"code": "structured-log-code", "message": "This is a log from the log file, code: 1906", "event_time": "2021-05-23 04:00:35", "player": "Jerry"}
{"code": "structured-log-code", "message": "This is a log from the log file, code: 28825", "event_time": "2021-05-23 04:00:36", "player": "Lucy"}
{"code": "structured-log-code", "message": "This is a log from the log file, code: 22654", "event_time": "2021-05-23 04:00:37", "player": "Bob"}
{"code": "structured-log-code", "message": "This is a log from the log file, code: 14430", "event_time": "2021-05-23 04:00:38", "player": "Jerry"}
{"code": "structured-log-code", "message": "This is a log from the log file, code: 28864", "event_time": "2021-05-23 04:00:39", "player": "Leo"}
{"code": "structured-log-code", "message": "This is a log from the log file, code: 22416", "event_time": "2021-05-23 04:00:40", "player": "John"}
```

6. 在Cloud Logging服务上能看到相关事件日志已经记录到Cloud Logging服务：




## 二、把Logging日志流式推送到Pub/Sub

1. 通过如下命令创建Pub/Sub主题sever-log-1

```
gcloud pubsub topics create server-log-1
```

2. 通过如下命令，创建日志流式推送到Pub/Sub sever-log-1主题

```
gcloud logging sinks create server-log-1 \
pubsub.googleapis.com/projects/${PROJECT_ID}/topics/server-log-1 \
--log-filter='logName="projects/${PROJECT_ID}/logs/structured-log"'
```

请注意这里**logName**里面的**your-project-id**需要修改成实际的项目**ID**

命令执行完成后会看到如下信息，请注意记下新创建serviceAccount，为下一步授权所用。

```
Created [https://logging.googleapis.com/v2/projects/[redacted]/sinks/server-log-1].
Please remember to grant `serviceAccount:p506[redacted]3@gcp-sa-logging.iam.gserviceaccount.com` the Pub/Sub Publisher role on the topic.
More information about sinks can be found at https://cloud.google.com/logging/docs/export/configure_export
```

3. 为新创建的服务账号授予该主题的publisher权限

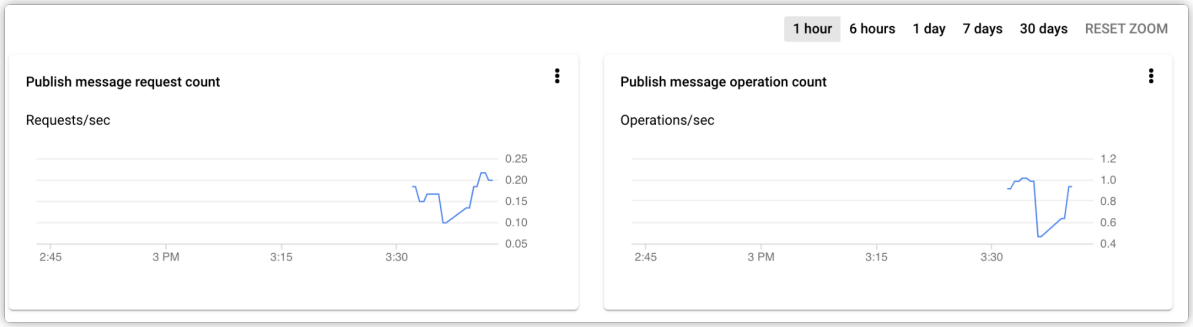
```
gcloud pubsub topics add-iam-policy-binding server-log-1 \
--member='serviceAccount:your-serviceaccount-id' \
--role=roles/pubsub.publisher
```

```
gcloud pubsub topics add-iam-policy-binding server-log-1 \
--member='serviceAccount:p247839977271-394911@gcp-sa-logging.iam.gserviceaccount.com' \
--role=roles/pubsub.publisher
```

请注意这里**member**里面的**your-serviceaccount-id**需要修改成上一步生成的服务账号**id**

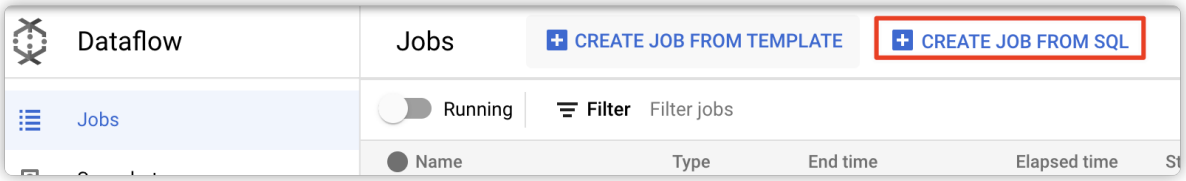
4. 给topic创建一个subscription，否则消息会全部丢弃

5. 若此时有执行之前的日志产生脚本，应该能在Pub/Sub控制台看到有消息记录

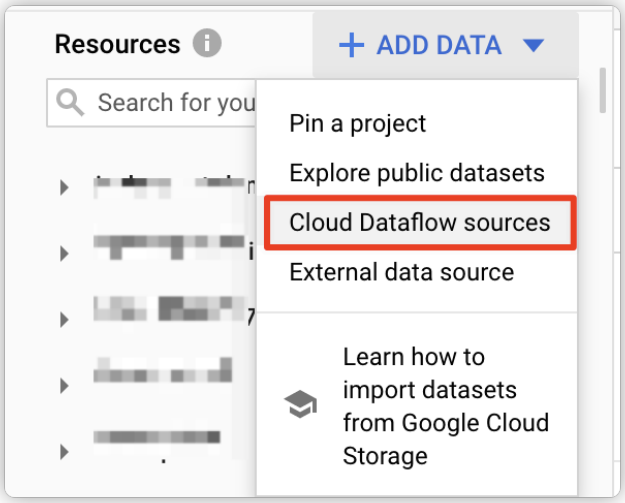



## 三、在Dataflow SQL中添加Pub/Sub数据源，并创建Schema

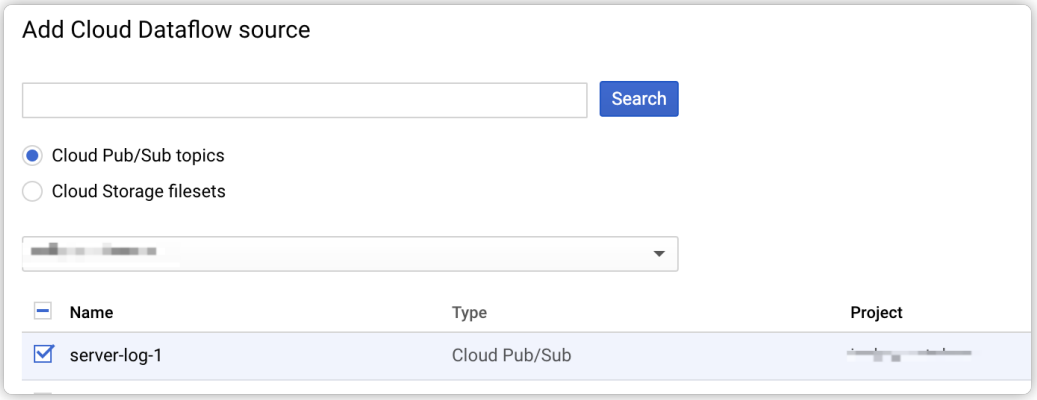
1. 在Dataflow控制台选择Create Job From SQL进入在Dataflow SQL控制界面




2. 在界面中点击ADD DATA，选择Cloud Dataflow sources添加数据源




3. 选择刚刚创建好的Pub/Sub主题server-log-1




4. 创建Pub/Sub主题的Schema


按实际记录需要创建Schema，以下为本Demo的Schema，主要框架是由Logging定义完成，我们只需要按实际情况定义jsonPayload这个STRUCT里面的架构即可。
注意：event_timestamp为必须值，该条目记录Pub/Sub获取日志的时间。

```
[
  {
    "name": "event_timestamp",
    "description": "Pub/Sub event timestamp",
```

```
        "mode": "REQUIRED",
        "type": "TIMESTAMP"
    },
    {
        "name": "logName",
        "description": "",
        "mode": "NULLABLE",
        "type": "STRING"
    },
    {
        "name": "resource",
        "description": "",
        "mode": "NULLABLE",
        "type": "STRUCT",
        "fields": [
            {
                "name": "type",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "labels",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRUCT",
                "fields": [
                    {
                        "name": "project_id",
                        "description": "",
                        "mode": "NULLABLE",
                        "type": "STRING"
                    },
                    {
                        "name": "instance_id",
                        "description": "",
                        "mode": "NULLABLE",
                        "type": "STRING"
                    },
                    {
                        "name": "zone",
                        "description": "",
                        "mode": "NULLABLE",
                        "type": "STRING"
                    }
```

```
                ]
            }
        ]
    },
    {
        "name": "textPayload",
        "description": "",
        "mode": "NULLABLE",
        "type": "STRING"
    },
    {
        "name": "jsonPayload",
        "description": "",
        "mode": "NULLABLE",
        "type": "STRUCT",
        "fields": [
            {
                "name": "code",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "message",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "event_time",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "player",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            }
        ]
    },
    {
        "name": "timestamp",
        "description": "",
```

```
        "mode": "NULLABLE",
        "type": "TIMESTAMP"
    },
    {
        "name": "receiveTimestamp",
        "description": "",
        "mode": "NULLABLE",
        "type": "TIMESTAMP"
    },
    {
        "name": "severity",
        "description": "",
        "mode": "NULLABLE",
        "type": "STRING"
    },
    {
        "name": "insertId",
        "description": "",
        "mode": "NULLABLE",
        "type": "STRING"
    },
    {
        "name": "httpRequest",
        "description": "",
        "mode": "NULLABLE",
        "type": "STRUCT",
        "fields": [
            {
                "name": "requestMethod",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "requestUrl",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "requestSize",
                "description": "",
                "mode": "NULLABLE",
                "type": "INT64"
            },
```

```
            {
                "name": "status",
                "description": "",
                "mode": "NULLABLE",
                "type": "INT64"
            },
            {
                "name": "responseSize",
                "description": "",
                "mode": "NULLABLE",
                "type": "INT64"
            },
            {
                "name": "userAgent",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "remoteIp",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "serverIp",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "referer",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "cacheLookup",
                "description": "",
                "mode": "NULLABLE",
                "type": "BOOL"
            },
            {
                "name": "cacheHit",
                "description": "",
```

```
                "mode": "NULLABLE",
                "type": "BOOL"
            },
            {
                "name": "cacheValidatedWithOriginServer",
                "description": "",
                "mode": "NULLABLE",
                "type": "BOOL"
            },
            {
                "name": "cacheFillBytes",
                "description": "",
                "mode": "NULLABLE",
                "type": "INT64"
            },
            {
                "name": "protocol",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            }
        ]
    },
    {
        "name": "labels",
        "description": "",
        "mode": "NULLABLE",
        "type": "STRUCT",
        "fields": [
            {
                "name": "compute_googleapis_com_resource_name",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            }
        ]
    },
    {
        "name": "operation",
        "description": "",
        "mode": "NULLABLE",
        "type": "STRUCT",
        "fields": [
            {
                "name": "id",
```

```
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "producer",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "first",
                "description": "",
                "mode": "NULLABLE",
                "type": "BOOL"
            },
            {
                "name": "last",
                "description": "",
                "mode": "NULLABLE",
                "type": "BOOL"
            }
        ]
    },
    {
        "name": "trace",
        "description": "",
        "mode": "NULLABLE",
        "type": "STRING"
    },
    {
        "name": "spanId",
        "description": "",
        "mode": "NULLABLE",
        "type": "STRING"
    },
    {
        "name": "traceSampled",
        "description": "",
        "mode": "NULLABLE",
        "type": "BOOL"
    },
    {
        "name": "sourceLocation",
        "description": "",
```

```
        "mode": "NULLABLE",
        "type": "STRUCT",
        "fields": [
            {
                "name": "file",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            },
            {
                "name": "line",
                "description": "",
                "mode": "NULLABLE",
                "type": "INT64"
            },
            {
                "name": "function",
                "description": "",
                "mode": "NULLABLE",
                "type": "STRING"
            }
        ]
    }
]
```

## 四、在Google Cloud Storage创建数据，并建立Dataflow可以访问的文件集

备注：(1). Cloud Storage 文件集必须具有架构，并且只能包含没有标题行的 CSV 文件。
(2). Cloud Storage数据可以直接创建csv文件并保存至存储桶中。但实际应用场景下，大部分的数据都是由大数据平台产生，并存放到Cloud Storage上，所以本Demo展示通过Hive创建的csv文件集。

1. 利用Dataproc执行Hive，在 Cloud Storage中创建外部表user_table，数据存放到存储桶的/user_table_csv/csv/下

首先在us-central1创建一个名为presto-cluster的Dataproc集群。

REGION=us-central1
BUCKET=youzhi-lab

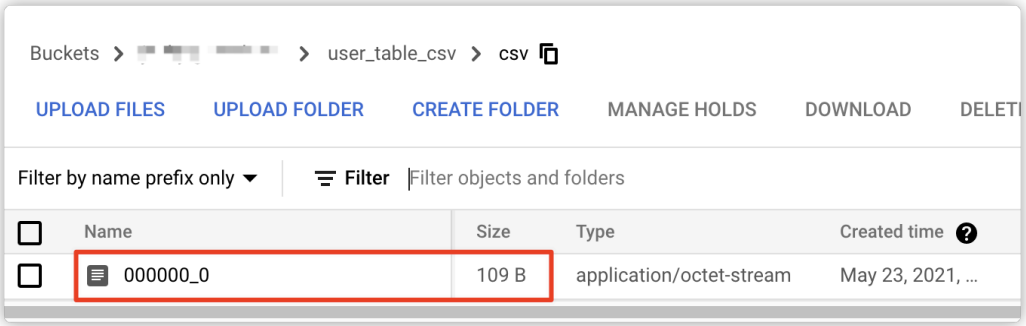
```
gcloud dataproc jobs submit hive \
   --cluster presto-cluster \
   --region=${REGION} \
   --execute "
```

```
      CREATE EXTERNAL TABLE user_table(
        name STRING ,
        gender STRING ,
        location STRING )
      ROW FORMAT DELIMITED
      FIELDS TERMINATED BY ','
      STORED AS TEXTFILE
      location 'gs://${BUCKET}/user_table_csv/csv/';"
```

2. 插入测试样例数据

```
gcloud dataproc jobs submit hive \
   --cluster presto-cluster \
   --region=${REGION} \
   --execute "
      insert into user_table (name , gender, location)
      values ('Jacky' , 'male','GZ'),
      ('Lucy' , 'female','SZ'),
      ('Bob' , 'male','SH'),
      ('Jerry' , 'male','BJ'),
      ('Michael' , 'male','CD'),
      ('John' , 'male','WH'),
      ('Eric' , 'male','TJ'),
      ('Leo' , 'male','HZ');"
```

3. 该数据文件集能在Cloud Storage里面找到




数据表内容为：

| user_table_csv.name | user_table_csv.gender | user_table_csv.location |
|---|---|---|
| Jacky | male | GZ |
| Lucy | female | SZ |
| Bob | male | SH |
| Jerry | male | BJ |
| Michael | male | CD |
| John | male | WH |
| Eric | male | TJ |
| Leo | male | HZ |

4. 利用DataCatalog创建Storage文件集

4.1 创建名字为server_log的条目组

```
gcloud data-catalog entry-groups create server_log \
   --location=us-central1
```

4.2 创建名字为server_log_1的文件集

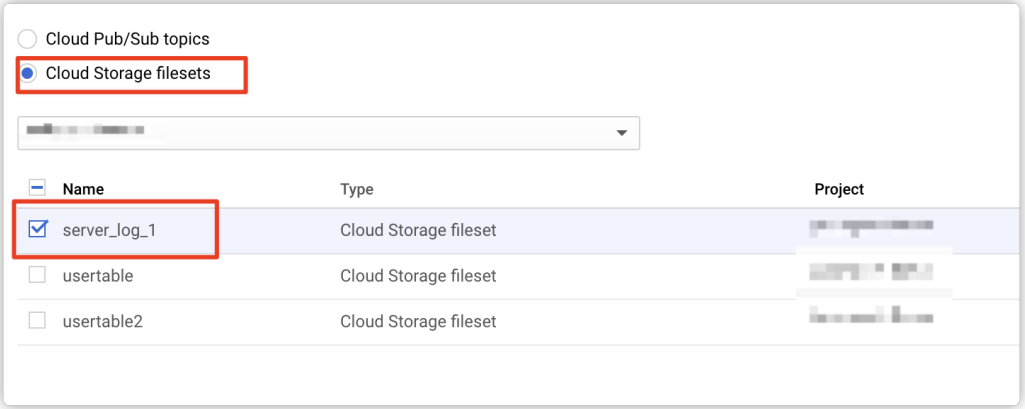
```
gcloud data-catalog entries create server_log_1 \
   --location=${REGION} \
   --entry-group=server_log \
   --type=FILESET \
   --gcs-file-patterns=gs://${BUCKET}/user_table_csv/csv/* \
   --schema-from-file=/tmp/fileset_schema.json
```

其中--schema-from-file所需要定义的文件集Schema为：

```
[
 {
   "column": "name",
   "description": "Player Name",
   "mode": "NULLABLE",
   "type": "STRING"
 },
 {
   "column": "gender",
   "description": "Player Gender",
   "mode": "NULLABLE",
   "type": "STRING"
 },
 {
   "column": "location",
   "description": "Player Location",
   "mode": "NULLABLE",
   "type": "STRING"
 }
```

```
]
```

5. 在Dataflow SQL控制界面中添加该文件集为数据源




## 五、通过Dataflow SQL，把实时的服务器事件日志与数据湖中的数据进行流式分析处理

1. 分析结果将保存到Bigquery，创建Bigquery数据集server_log_dataset

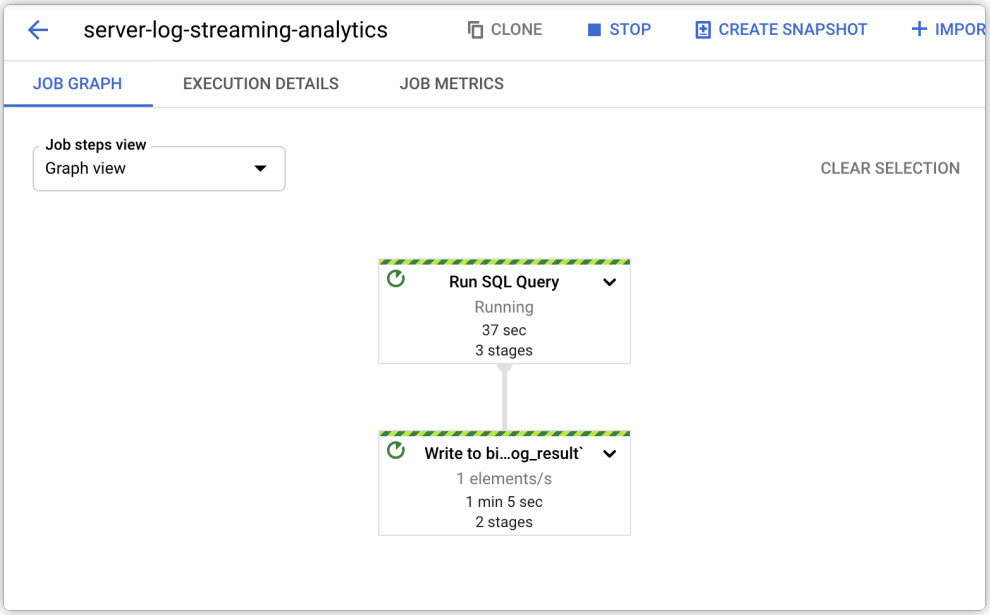
```
bq mk server_log_dataset
```

2. 执行Dataflow SQL命令，进行流式数据分析

```
gcloud dataflow sql query \
 --job-name=server-log-streaming-analytics-02\
 --region=${REGION}\
 --bigquery-dataset=server_log_dataset \
 --bigquery-table=server_log_result \
 'SELECT
   tr.timestamp as event_timestamp,
   tr.resource.labels.instance_id  as instance_id,
   tr.jsonPayload.message as message,
   tr.jsonPayload.player as player,
   tr.jsonPayload.event_time as game_event_time,
   sr.gender as gender,
   sr.location as location
 FROM pubsub.topic.`your-project-id`.`server-log-1` as tr
   inner join
datacatalog.entry.`your-project-id`.`your-region`.server_log.server_log_1 as
sr
   on tr.jsonPayload.player = sr.name'
```

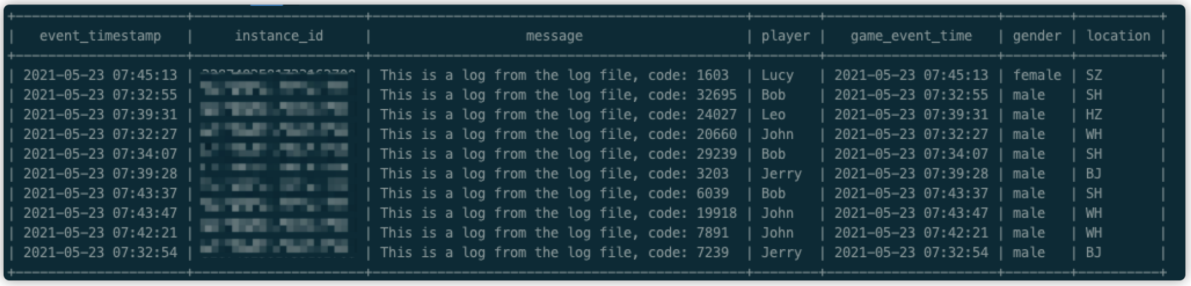
请注意这里的**your-project-id**为项目**ID**，**your-region**为文件集所在的**region**，请注意修改。

3. 通过Dataflow控制台可以看到流式处理任务正在处理




4. 查询Bigquery数据表，检查结果

```
bq query "select * from server_log_dataset.server_log_result limit 10"
```



| event_timestamp | instance_id | message | player | game_event_time | gender | location |
|---|---|---|---|---|---|---|
| 2021-05-23 07:45:13 | [redacted] | This is a log from the log file, code: 1603 | Lucy | 2021-05-23 07:45:13 | female | SZ |
| 2021-05-23 07:32:55 | [redacted] | This is a log from the log file, code: 32695 | Bob | 2021-05-23 07:32:55 | male | SH |
| 2021-05-23 07:39:31 | [redacted] | This is a log from the log file, code: 24027 | Leo | 2021-05-23 07:39:31 | male | HZ |
| 2021-05-23 07:32:27 | [redacted] | This is a log from the log file, code: 20660 | John | 2021-05-23 07:32:27 | male | WH |
| 2021-05-23 07:34:07 | [redacted] | This is a log from the log file, code: 29239 | Bob | 2021-05-23 07:34:07 | male | SH |
| 2021-05-23 07:39:28 | [redacted] | This is a log from the log file, code: 3203 | Jerry | 2021-05-23 07:39:28 | male | BJ |
| 2021-05-23 07:43:37 | [redacted] | This is a log from the log file, code: 6039 | Bob | 2021-05-23 07:43:37 | male | SH |
| 2021-05-23 07:43:47 | [redacted] | This is a log from the log file, code: 19918 | John | 2021-05-23 07:43:47 | male | WH |
| 2021-05-23 07:42:21 | [redacted] | This is a log from the log file, code: 7891 | John | 2021-05-23 07:42:21 | male | WH |
| 2021-05-23 07:32:54 | [redacted] | This is a log from the log file, code: 7239 | Jerry | 2021-05-23 07:32:54 | male | BJ |

5. 从结果来看，符合之前在Dataflow SQL里面建立的SQL语句进行联合查询的预期。