UNIVERSITY OF ALBERTA

Master of Science in Internetworking

Department of Electrical and Computer Engineering

Project Title:

Optimized Data pipeline between Google Cloud Platform (GCP) and on-premises

Supervisor:

Ali Tizghadam

(Principal Technology Architect -TELUS)

Provided By:

Sarwar Sulthana

Fall 2021 – Winter 2022

Contents

Table of Figures

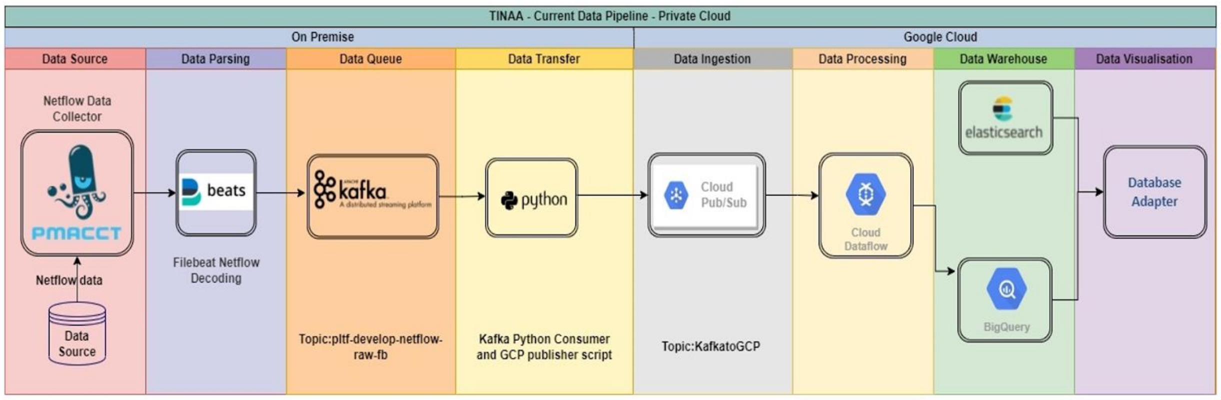
List of Tables

Abstract

Project Objective:
Create a data pipeline that would run parallel with an existing private cloud pipeline to extract data from an on-premises network. Then transform, store it using managed Google Cloud Platform (GCP) services and finally read the data from the TINAA datastore application.

TINAA program (TELUS Intelligent Network Analytics & Automation) has an existing data pipeline in their private cloud infrastructure. In this pipeline, the NetFlow records get collected at PMACCT (open source NetFlow collector), then processed using Spark, and then published to an internal Elasticsearch database.

Now, using the Google Cloud Platform (GCP) offered by Google to provide reliable and high-performance cloud services, we have developed an optimum and scalable data pipeline as shown below.



1. Firstly, we run a python based docker container to extract the data by subscribing to an existing Kafka topic. This data is then published to GCP PubSub Topic.
2. Secondly, we modernized the data pipeline using managed GCP services.
    - Google PubSub – Asynchronous messaging service to stream the NetFlow data.
    - Google DataFlow – Fully managed streaming analytics service to stream the NetFlow data from GCP PubSub to BigQuery table.
    - Google BigQuery – Serverless, cost-effective and multi-cloud data warehouse to store the NetFlow data.
3. Finally, we consumed the data stored in BigQuery using a custom database adapter interface from TINAA datastore service applications.

Acknowledgement

I would like to thank the following people, without whom I would not have completed this Capstone project.

I would like to express my sincere gratitude to my mentor, ***Prof. Ali Tizghadam,*** for his extensive support and guidance, with which I was able to complete this capstone project successfully. His reviews and suggestions were a real inspiration for me in completing this project.

I owe my profound gratitude to our Program director ***Dr. Mike MacGregor*** who gave me this wonderful opportunity to experience this project.

The outcome of this project required a lot of guidance and assistance from TELUS team members reporting Ali. I would like to thank them for their co-operation in providing all the necessary information regarding the project and helping me with their skills.

1. Project Architecture

1.1. Current Architecture:

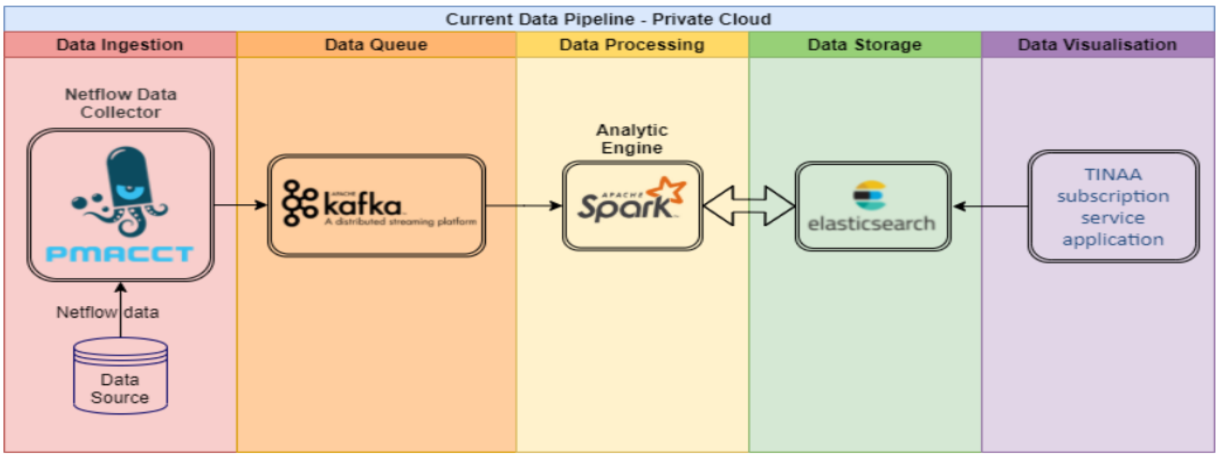


Figure 1. 1 Current Architecture (TINAA Pipeline)

1.2. Proposed Architecture:

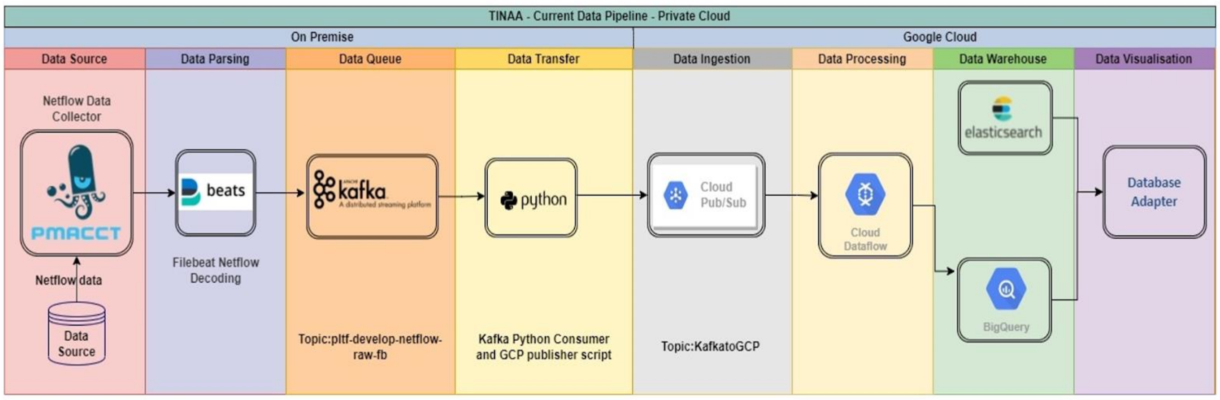


Figure 1. 2 Proposed Pipeline architecture

1.3. Protocol Details:

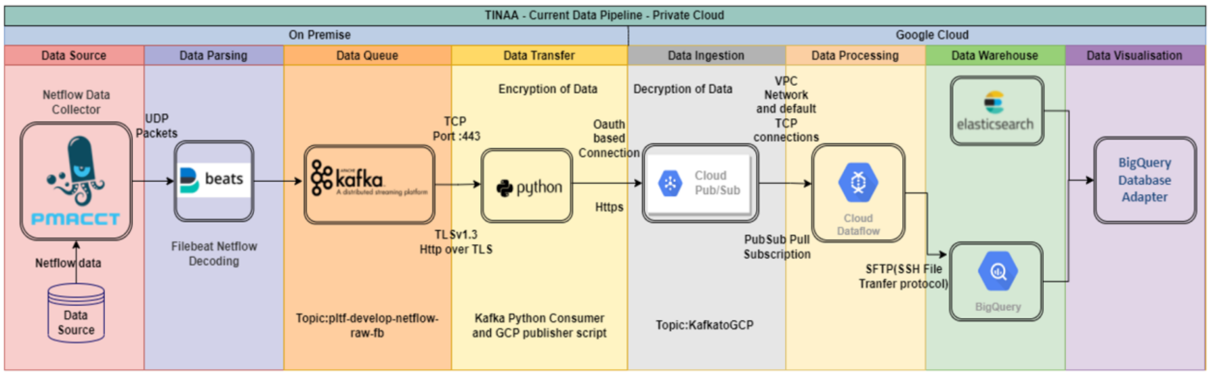


Figure 1. 3 Communication Protocols

1.4. Components of Data pipeline:

Data Source: Source that generates/ingest Netflow data into PMACCT.

PMACCT: A set of passive network monitoring tools to measure, account, classify, aggregate and export IPv4 and IPv6 traffic

Filebeat: It is a lightweight, efficient, relatively easy to use shipper for forwarding and centralizing log data. It is generally used for gathering, parsing, and saving logs.

Apache Kafka: Open-source distributed event streaming platform used for high-performance data pipelines, streaming analytics, data integrations

Elasticsearch: A distributed, free, open search and analytics engine for all types of data.

Docker: Docker is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called containers that have everything the software needs to run.

Cloud PubSub: Google Pub/Sub is a fully managed real-time messaging service that allows you to send and receive messages between independent applications

Cloud DataProc: Fully managed and highly scalable service for running Apache Spark.

Cloud DataFlow: Fully managed streaming analytics service that minimizes latency, processing time, and cost through autoscaling and batch processing

Cloud Big Query: BigQuery is Google’s fully managed enterprise data warehouse that helps you manage and analyze your data. It is a serverless cloud storage platform for large datasets.

BigTable Plugin: An interface to TINAA subscription service application to be able to read data from BigTable and test it.

1.5. Access and Pre-requisites

Request Access for TELUS Lab VPN

1. **Pulse Secure VPN:** To access TELUS Lab environment.
2. **Cisco AnyConnect:** to access TELUS Network.

Access is two types

1. **XID:** For Staff access
2. **VID:** For Vendor access

1.5.1. Application-level Access:

1. **GitLab:** To access Gitlab on Lab environment connect to Lab VPN→ Provide XID credentials→ Create a project (ensure you have Maintainer role)
2. **OpenShift Container Platform:** Project access (Developer Access)
3. **Kafka Web Server:** Toll Environment (Access to the Topic with netflow records)
4. **Google Cloud Platform:** Enable required Cloud resources
   a. Google PubSub: (PubSub Admin role)
   b. Cloud DataFlow:
   c. BigQuery: (BigQuery Admin role)
5. **TINAA google project access**

1.5.2. Software requirements:

Software	Version
Docker	**Version**: 1.13.1 Docker Desktop Installer
Python	**Version:** 3.9.1 **Installation**: *docker pull python* **Pip:** Python package manager **Pip Version:** pip 22.0.3
Visual Studio Code Editor	Visual Studio Code 1.64.2
Filebeat	**Filebeat:** 8.0.0 **Installation:** docker pull docker.elastic.co/beats/filebeat:8.0.0
PMACCT (for POC)	**Version**: latest ***Installation:*** *docker pull pmacct/pmacctd* **configuration file:** /etc/pmacct/pmacctd.conf: For Project use the existing TINAA PMACCT collector.
Gitbash/Terminal	git version 2.34.1.windows.1 **Terminal**: windows /command prompt
Microsoft Office	MS Word, One Note

Table 1. 1 Software requirements for Project

2. What is Data Pipeline:

A data pipeline is a set of tools and processes used to automate the movement and transformation of data between a source system and a target repository.

Source systems often have different methods of processing and storing data than target systems. It is a set of actions that ingest raw data from disparate sources and move the data to a destination for storage and analysis. More specially, a data pipeline is an end-to-end process to ingest, process, prepare, transform, and enrich structured, unstructured, and semi-structured data in a governed manner [1] .

Therefore, data pipeline software automates the process of extracting data from many disparate source systems, transforming, combining, and validating that data, and loading it into the target repository.

2.1. Why is building a Data Pipeline Important?

Deploying the data pipeline will help companies build and manage workloads in the cloud efficiently. Organizations can improve data quality, connect to diverse data sources, ingest structured and unstructured data into cloud data lake, and manage complex multi-cloud environments [2].

Data Pipelines are categorized based on customer use cases.

There are two types of data pipelines [3]:

2.1.1. Streaming Processing data pipeline:

Streaming data pipelines are used when the analytics, application or business process requires continually flowing and updating data. Instead of loading data in batches, streaming pipelines move data continuously in real-time from source to target.

- Data is sourced, manipulated, and loaded as soon as it’s created.

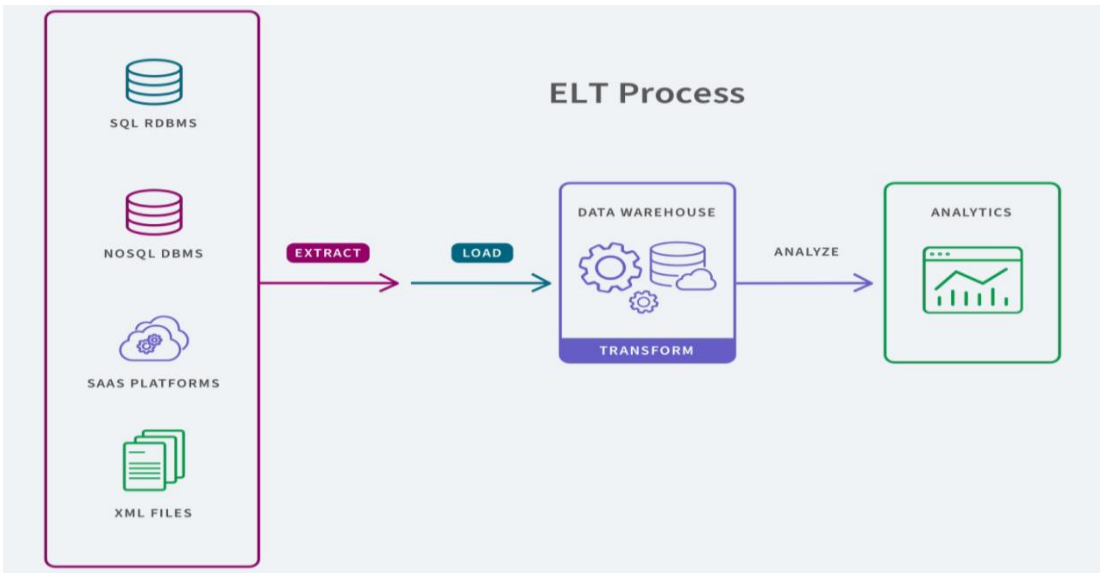


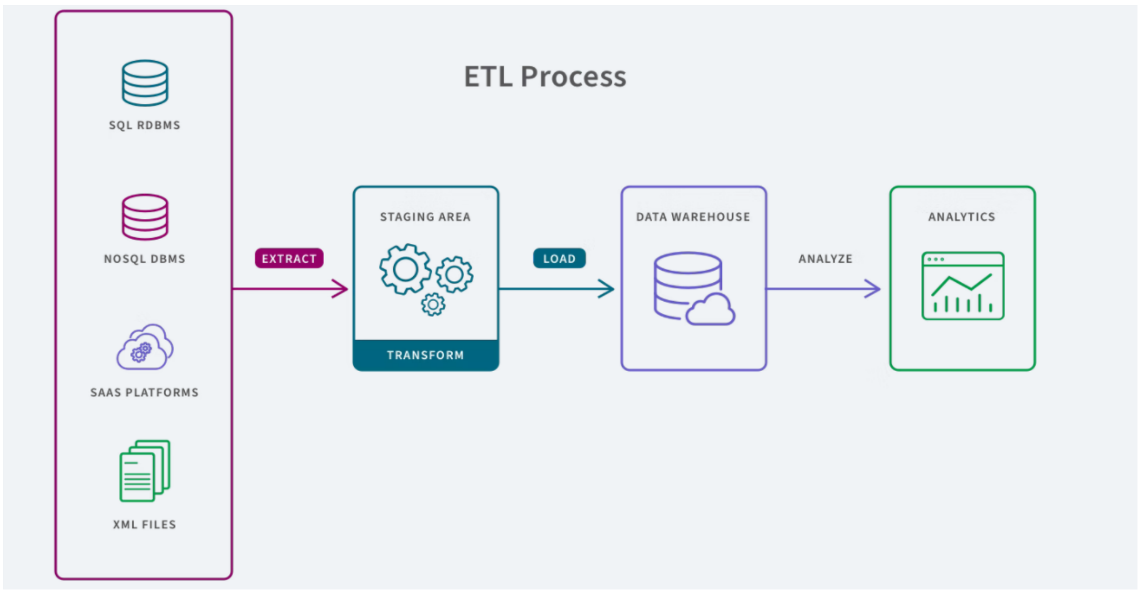
Figure 2. 1 Streaming Processing Pipeline (Continuous Flow)

2.1.2. Batch Processing data pipeline:

Historical data is typically used in BI and data analytics to explore, analyze, and gain insights on activities and information that has happened in the past. Therefore, traditional batch processing where data is periodically extracted, transformed, and loaded to a target system is sufficient. These batches can either be scheduled to occur automatically, can be triggered by a user query or by an application. Batch processing enables complex analysis of large datasets [3].

- Source data is collected periodically and sent to the destination system.

Figure 2. 2 Batch Processing Pipeline (Periodic Flow)



NOTE: In this project we are streaming continuous Netflow records from Kafka to google cloud for real-time processing and analytics. So, we are creating a streaming data pipeline

2.2. Stages in Data Pipeline

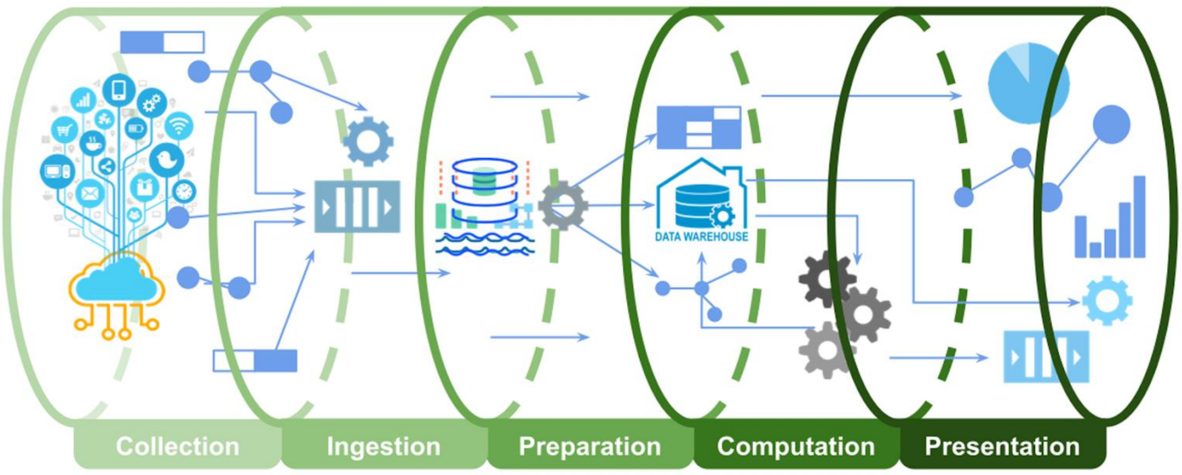


Figure 2. 3 Stages in Data Pipeline

2.2.1. Data Collection:

This is the first stage of data pipeline, where the data is collected from source, formatted, or parsed appropriately, and ingested into the pipeline.

1. Source generates the Netflow Data, and it is sent to PMACCT (open source netflow collector).
2. The PMACCT collects all the Netflow data records and the raw Netflow streams are stored in Kafka topic (**pltf-develop-netflow-raw)**

2.2.2. Data Parsing:

In this stage the raw netflow UDP records from PMACCT, which are unencrypted, are decoded and parsed to JSON using the filebeat netflow codec module and the results are stored in the Kafka topic **(pltf-develop-netflow-raw-fb)**

2.2.3. Data Queue:

In this stage, messages decoded are stored in the Kafka topic(**pltf-develop-netflow-raw-fb)**. When the PMACCT publishes NetFlow data to Kafka it will stream the events to its subscribers i.e., DataFlow and store the data in the GCP Destination (BigQuery).

2.2.4 Data Transfer:

Python Kafka Consumer client is created to subscribe to the Kafka Topic and consume the records from the topic.

2.2.5 Data Ingestion:

Stream of netflow records consumed by the Kafka consumer are now published to GCP PubSub topics for further processing. We have a python gcp-netflow-publisher, which publishes the netflow data from Kafka Topic to GCP PubSub Topic.

2.2.6 Data Processing:

At this stage, dataflow job is triggered to perform ETL operations. Data flow job moves the netflow records from PubSub Subscription to BigQuery Table.

2.2.7 Data Warehouse:

DataFlow job is triggered then netflow records are stored in GCP destination (BigQuery).

2.2.8 Data Visualization:

In this stage we have created a big query database adapter like the existing elastic database adapter. So, it can be consumed by any client application and end user will have a choice to consume the data from either Elastic or BigQuery.

3. Data Collection (Stage 1):

3.1. Data Source:

The netflow data is a protocol system that collects active IP network traffic as it flows in or out of an interface [4]

3.2. Pmacct netflow collector:

Pmacct is a small set of multi-purpose passive network monitoring tools. It can account, classify, aggregate, replicate and export forwarding-plane data, i.e., IPv4 and IPv6 traffic; collect and correlate control-plane data via BGP and BMP; collect and correlate RPKI data; collect infrastructure data via Streaming Telemetry [4].

3.2.1.Pmacct contains

- **Pmacctd:** a probe, i.e., it collects packages and exports Ipfix/Netflow via a tool called nfprobe [5].
- **Nfacctd:** The flow is sent nfacctd, a collector for IPFIX/netflow that is part of the Pmacct suite [6].

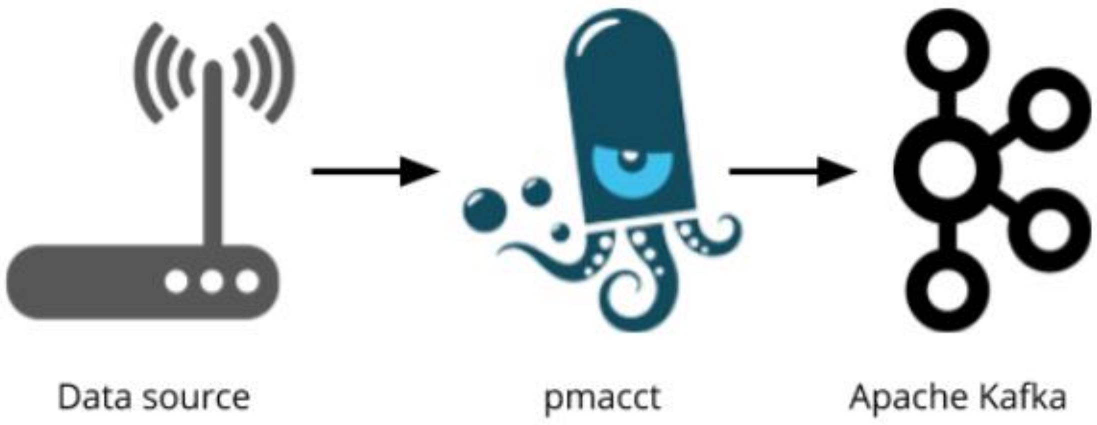
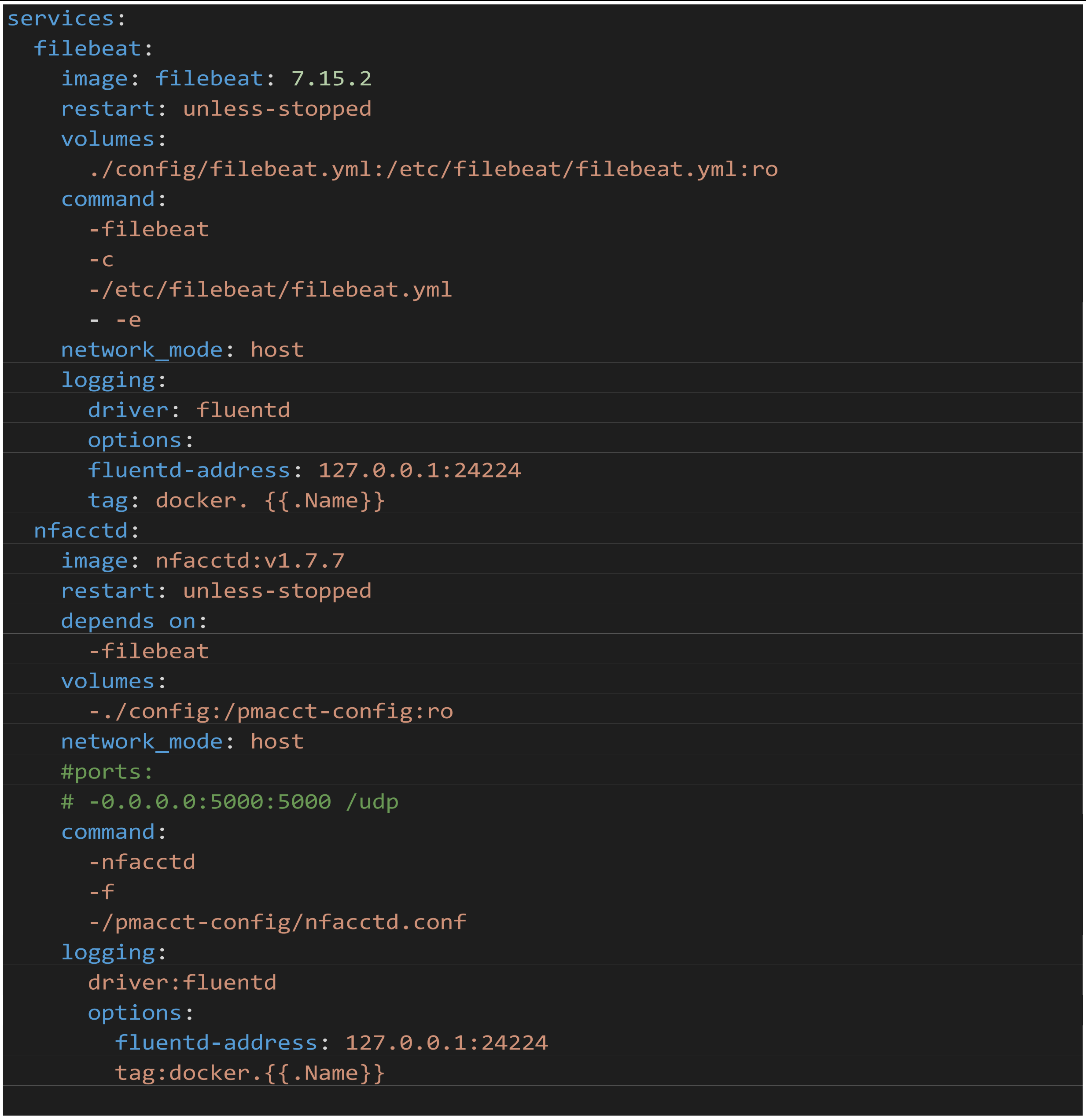


Figure 3. 1 Netflow ingestion from PMACCT to Kafka

In this project: We want to use the nfacct daemon as collector and exporter to Kafka topic (**pltf-develop-netflow**) [7]. So, we need to configure that. We create a file called nfacctd.conf with the following content:

3.3. Pmacct/nfacctd config:



```
services:
  filebeat:
    image: filebeat: 7.15.2
    restart: unless-stopped
    volumes:
      ./config/filebeat.yml:/etc/filebeat/filebeat.yml:ro
    command:
      -filebeat
      -c
      -/etc/filebeat/filebeat.yml
      - -e
    network_mode: host
    logging:
      driver: fluentd
      options:
      fluentd-address: 127.0.0.1:24224
      tag: docker. {{.Name}}
  nfacctd:
    image: nfacctd:v1.7.7
    restart: unless-stopped
    depends on:
      -filebeat
    volumes:
      -./config:/pmacct-config:ro
    network_mode: host
    #ports:
    # -0.0.0.0:5000:5000 /udp
    command:
      -nfacctd
      -f
      -/pmacct-config/nfacctd.conf
    logging:
      driver:fluentd
      options:
        fluentd-address: 127.0.0.1:24224
        tag:docker.{{.Name}}
```

Configuration 3.1 Pmacct/Nfacctd Configuration

NOTE:

We have set up the Pmacct/nfacctd config to collect the netflow records and export to Kafka topic (**pltf-develop-Netflow-raw**). This topic has the raw netflow records (UDP packets, which are not encrypted). The decoding of the raw data in the topic is done by Filebeat and the decoded data is published to another Kafka topic (**pltf-develop-Netflow-raw-fb**), which has JSON decoded netflow data.

4. Data Parsing (Stage 2)

Filebeat and netflow decoding

Filebeat is a lightweight shipper for forwarding and centralizing log data. Installed as an agent on your servers, Filebeat monitors the log files or locations that you specify, collects log events, and forwards them either to Elasticsearch or Logstash for indexing [8].

- Mainly used for analyzing the log files.

4.1. What is filebeat module:

Filebeat comes with interesting modules that simplify the collection, parsing, and visualization of common log formats down to a single command [9].

Filebeat Kafka module collects and parses the logs created by Kafka. In our project the raw netflow records in the topic are parsed by filebeat Kafka module.

Filebeat consists of two main components:

- Inputs
- Harvesters.

These components work together to tail files and send events data to the output that you specify.

4.2. Here's how Filebeat works:

When you start Filebeat, it starts one or more inputs that look in the locations you've specified for log data. For each log that Filebeat locates, Filebeat starts a harvester [10]. Each harvester reads a single log for new content and sends the new log data to Filebeat, which aggregates the events and sends the aggregated data to the output that you've configured for Filebeat.

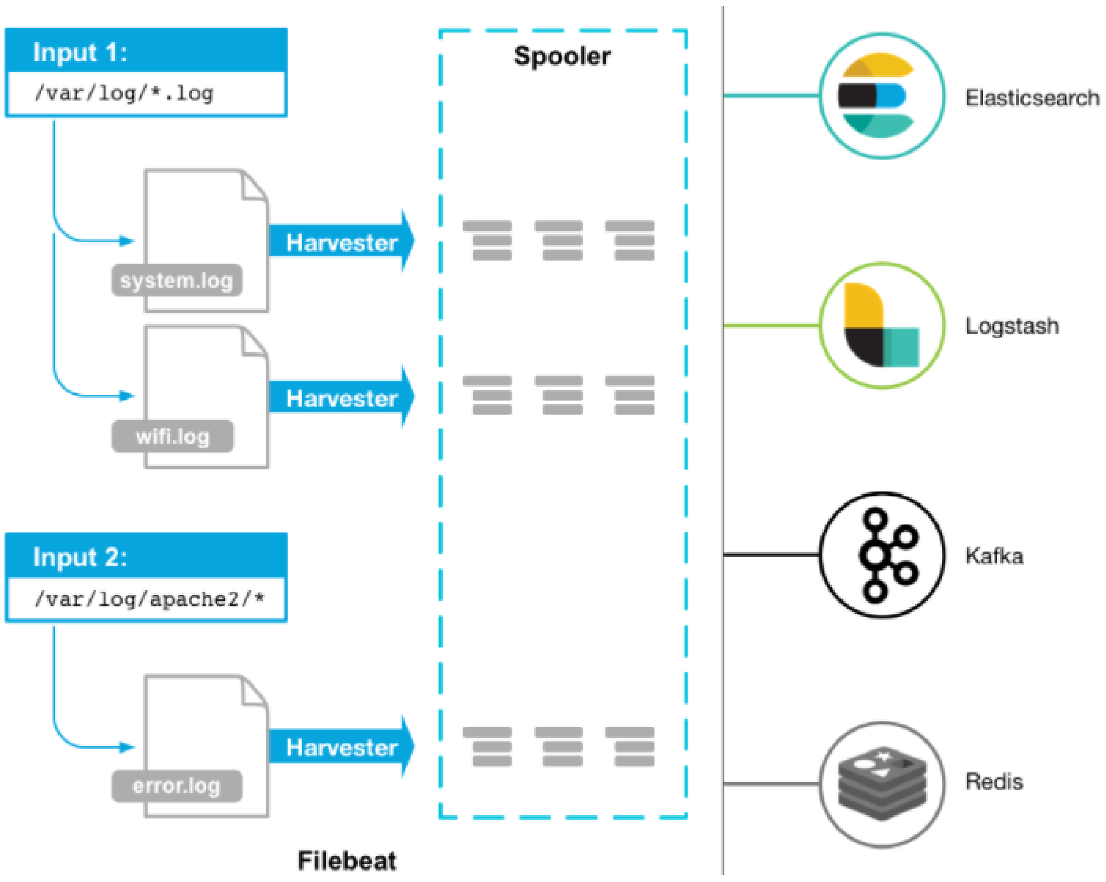


Figure 4. 1 Filebeat Functioning

4.2.1. What is a harvester?

A harvester is responsible for reading the content of a single file. The harvester reads each file, line by line, and sends the content to the output. One harvester is started for each file [10].

4.2.2. What is input?

An input is responsible for managing the harvesters and finding all sources to read from. If the input type is log, the input finds all files on the drive that match the defined glob paths and starts a harvester for each file [10].

4.3. How to Configure Filebeat:

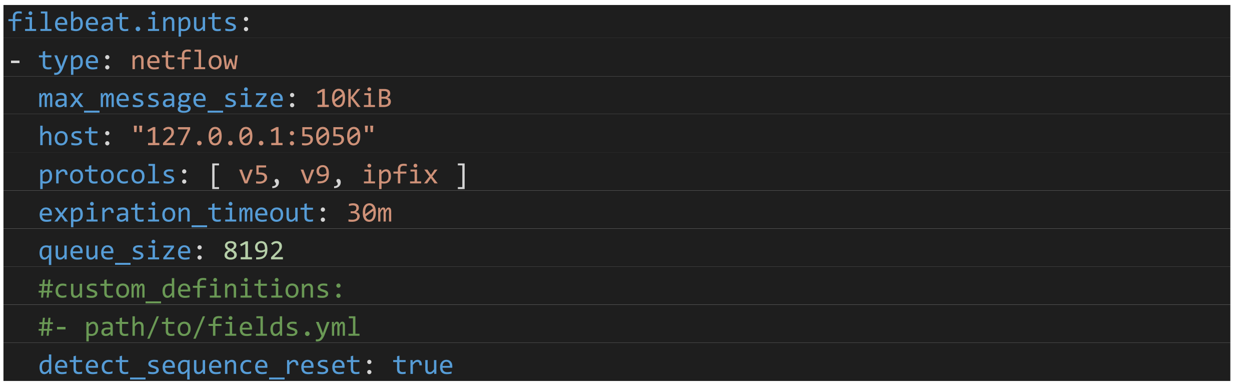
Filebeat is relatively easy to configure, and they all follow the same configuration step. Filebeat is configured using a YAML configuration file [11].

4.3.1.Filebeat inputs:

Filebeat inputs are responsible for locating specific files and applying basic processing to them. From this point, you can configure the path (or paths) to the file you want to track. Also, you can use additional configuration options such as the input type and the encoding used for reading the file, excluding, and including specific lines, adding custom fields and more [11].

Input to Filebeat: Netflow data is being sent as input on local host (127.0.0.1) and port (5050).

- **Filebeat Netflow Input for filbeat.conf.yml:**



```
filebeat.inputs:
- type: netflow
  max_message_size: 10KiB
  host: "127.0.0.1:5050"
  protocols: [ v5, v9, ipfix ]
  expiration_timeout: 30m
  queue_size: 8192
  #custom_definitions:
  #- path/to/fields.yml
  detect_sequence_reset: true
```

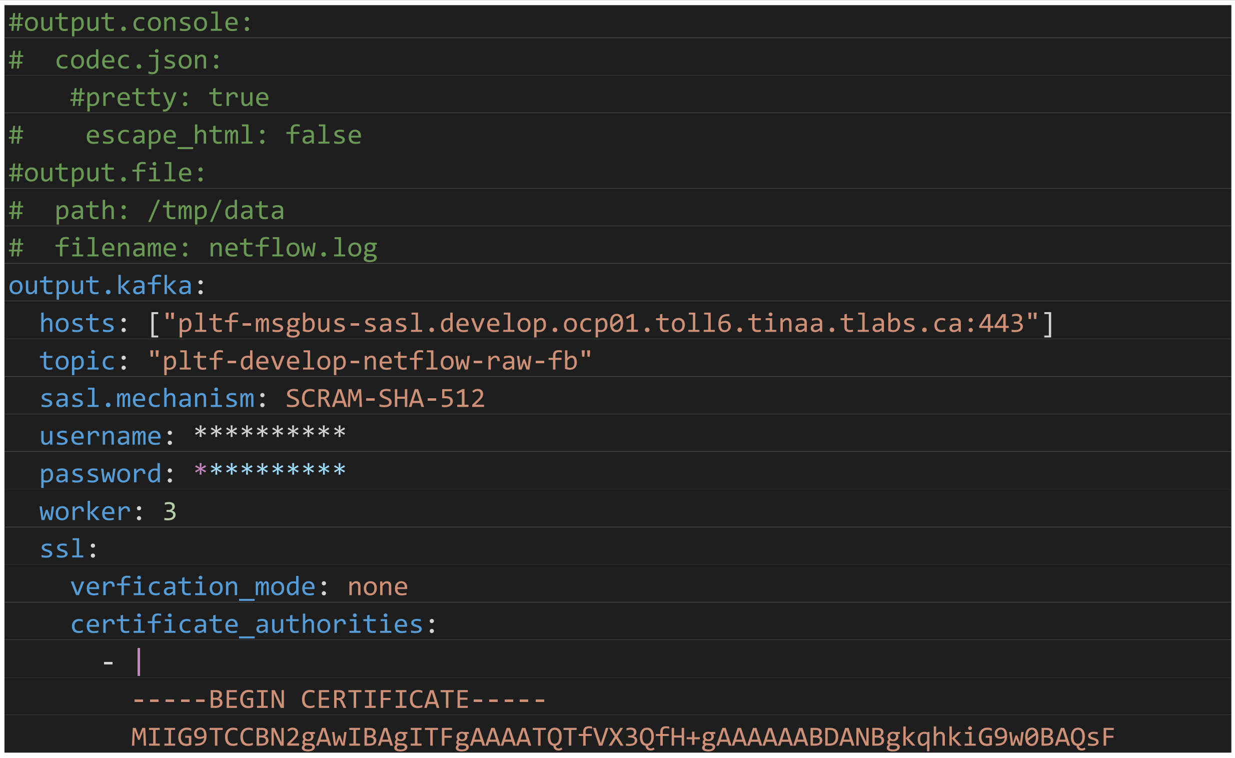
4.3.2.Filebeat outputs:

Here you can define where the data is going to be shipped. Most often you will use the Logstash or Elasticsearch output types. If you define a Logstash instance you can have advanced processing and data enhancement. You can define multiple outputs and use a load balancing option to balance the forwarding of data [11].

Filebeat Output: The output is decoded JSON netflow data exported to Kafka topic(*pltf-develop-Netflow-raw-fb)* using filebeat Codec module

Below figure is filebeat config that sends decoded data to Kafka topic **pltf-develop-netflow-raw-fb**

4.3.3. Filebeat Config (output) to decode the raw Netflow data:



```
#output.console:
#   codec.json:
    #pretty: true
#     escape_html: false
#output.file:
#   path: /tmp/data
#   filename: netflow.log
output.kafka:
  hosts: ["pltf-msgbus-sasl.develop.ocp01.toll6.tinaa.tlabs.ca:443"]
  topic: "pltf-develop-netflow-raw-fb"
  sasl.mechanism: SCRAM-SHA-512
  username: **********
  password: **********
  worker: 3
  ssl:
    verfication_mode: none
    certificate_authorities:
      - |
        -----BEGIN CERTIFICATE-----
        MIIG9TCCBN2gAwIBAgITFgAAAATQTfVX3QfH+gAAAAAABDANBgkqhkiG9w0BAQsF
```

NOTE: Decoded Netflow data (JSON format) is accumulated on Kafka topic using Filebeat.

5. Data Queue (Stage 3)

5.1 Data Stream:

A data stream is a set of events generated from different data sources at irregular intervals. They always travel from one system to another system, carrying the state changes that happened. An event stream represents related events in motion [12]. These events are immutable and never stay in one place

In Project: Data Stream = Netflow records (from PMACCT to Big Query)

5.2 What is Kafka:

Kafka: Apache Kafka is a highly scalable and distributed platform for creating and processing streams in real-time. It is a publishing-subscribe messaging system that maintains messages is partitioned and replicated topics [12].

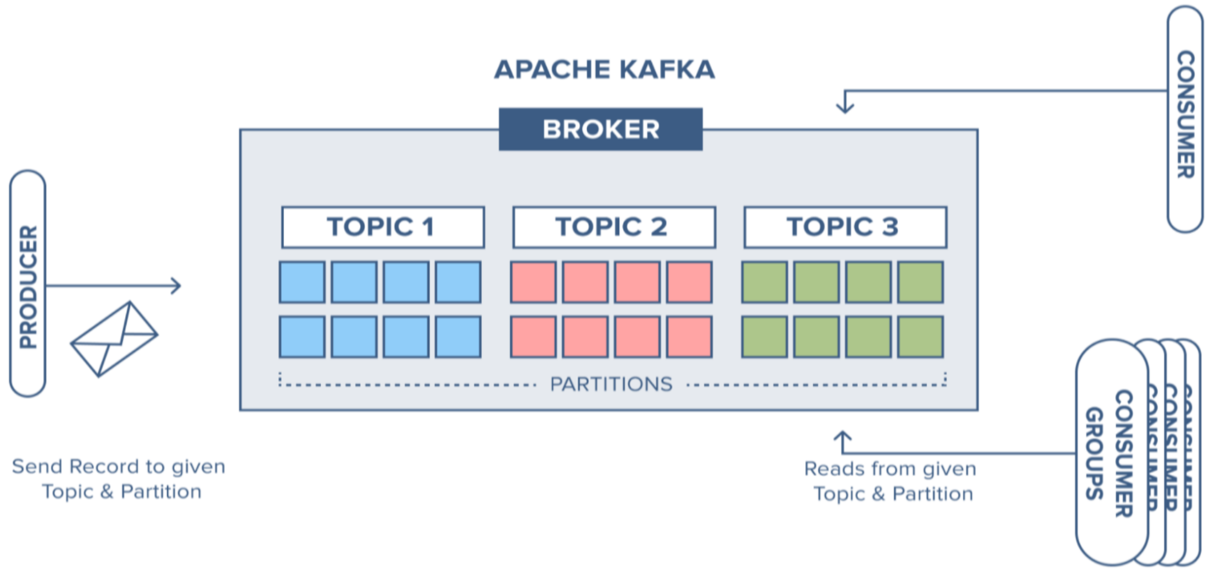


Figure 5. 1Apache Kafka Functioning

In Project: Kafka Broker:
https://pltf-msgbus-ui.develop.ocp01.toll6.tinaa.tlabs.ca/ui/develop-cluster/topic/pltf-develop-netflow-raw-fb/data?sort=NEWEST&partition=All

Topics: They are the logs that receive data from the producers and store them across their partitions. When an event stream enters Kafka, it is persisted as a topic. In other words, a topic is a stream at rest. Topics are the central concept in Kafka that decouples producers and consumers [12].

<u>Project Topics:</u>

Raw Netflow Topic: pltf-develop-netflow-raw

Decoded Netflow Topic: pltf-develop-netflow-raw-fb

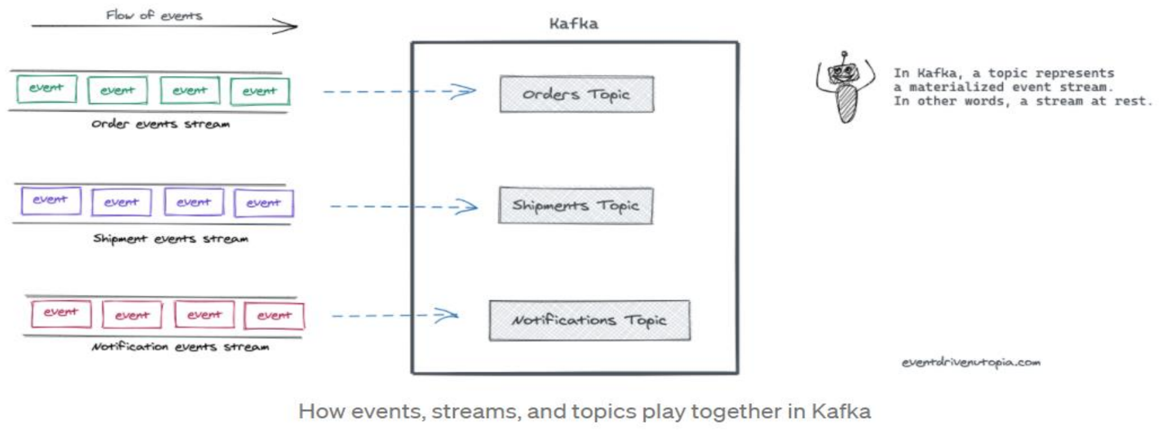

Flow of events
event
Order events stream
Shipment events stream
Notification events stream
Kafka
Orders Topic
Shipments Topic
Notifications Topic
In Kafka, a topic represents a materialized event stream. In other words, a stream at rest.
eventdrivenutopia.com
How events, streams, and topics play together in Kafka

Figure 5. 2 Messages/Events flow in Kafka Topics

Partitions: Kafka's topics are divided into several partitions. It is the smallest storage unit that holds a subset of records owned by a topic. Each partition is a single log file where records are written to it in an append-only fashion [12].

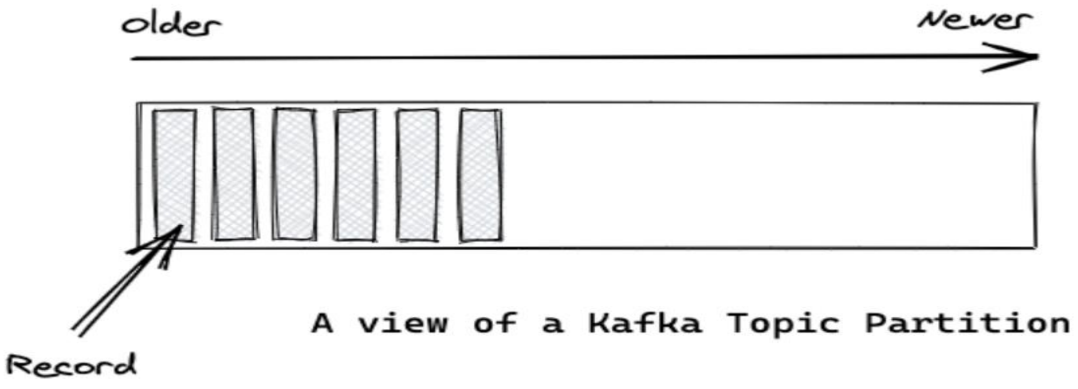


Figure 5. 3 Partitions in Kafka Topics

Consumer Groups: Kafka also has the concept of consumer groups where several consumers are grouped to consume a given topic [12].

In our scenario: The NetFlow data collected at PMACCT, decoded by Filebeat, and is stored in Kafka Topic.

6. Data Transfer (Stage 4)

6.1 Kafka Consumer

Now the netflow records are streaming in the Kafka Topic**: ‘pltf-develop-netflow-raw-fb’** In this stage we have a Kafka Consumer Python Client, which is subscribing to the topic and consuming all the netflow records. This consumer consumes netflow records from a Kafka Cluster [13].

Technical specifications for Python Consumer Client:

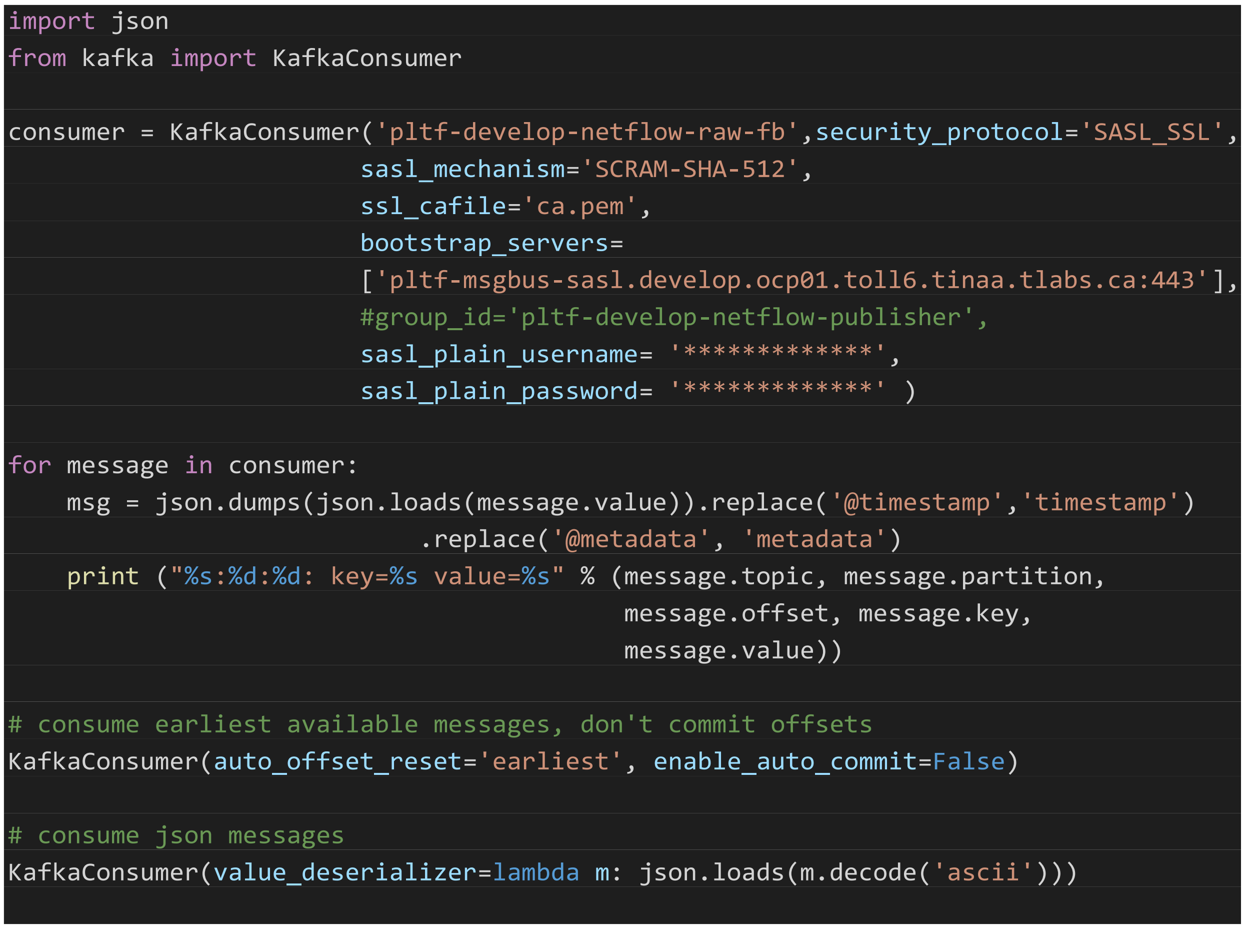
a) Programming language: Python
b) Version: 3.7.1
c) Installed Libraries: JSON, Kafka Consumer
d) Import Libraries:
   - pip install kafka-python
   - pip install json

Kafka Consumer: Consumers read the messages of a set of partitions of a topic of their choice at their own place. A single topic can be consumed by multiple consumers in parallel.

Reading records: Unlike the other pub/sub implementations, kafka doesn’t push messages to consumers. Instead, consumers have to pull messages off Kafka topic partitions. A consumer connects to a partition in a broker, reads the messages in order in which they were written [14].

6.2 Consuming the netflow data from topic (Python Client):

Kafka Python Consumer Client: Below is the code snippet for the Kafka Consumer [15], which consumes the data from the Kafka broker (toll environment), by subscribing to the topic **'pltf-develop-netflow-raw-fb**



```python
import json
from kafka import KafkaConsumer

consumer = KafkaConsumer('pltf-develop-netflow-raw-fb',security_protocol='SASL_SSL',
                        sasl_mechanism='SCRAM-SHA-512',
                        ssl_cafile='ca.pem',
                        bootstrap_servers=
                        ['pltf-msgbus-sasl.develop.ocp01.toll6.tinaa.tlabs.ca:443'],
                        #group_id='pltf-develop-netflow-publisher',
                        sasl_plain_username= '*************',
                        sasl_plain_password= '*************' )

for message in consumer:
    msg = json.dumps(json.loads(message.value)).replace('@timestamp','timestamp')
                            .replace('@metadata', 'metadata')
    print ("%s:%d:%d: key=%s value=%s" % (message.topic, message.partition,
                                          message.offset, message.key,
                                          message.value))

# consume earliest available messages, don't commit offsets
KafkaConsumer(auto_offset_reset='earliest', enable_auto_commit=False)

# consume json messages
KafkaConsumer(value_deserializer=lambda m: json.loads(m.decode('ascii')))
```

Below are the arguments that are passed [16]:

- **Topic:** Decoded netflow topic in our case **(pltf-develop-netflow-raw-fb)**
- **Security_protocol**: Protocol used to communicate with brokers. Valid values are PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL. Default: PLAINTEXT [16].
- **SASL_mechanism**: Authentication mechanism when security_ protocol is configured for SASL_PLAINTEXT or SASL_SSL [16].
- **bootstrap _servers:** ‘host[:port]’ string (or list of ‘host[:port]’ strings) that consumer should contact to bootstrap initial cluster metadata. It can be at least one broker that will respond to Metadata API request. Default port: 9092. Default value: localhost: 9092 [16].
- **group_id:** this consumer group to which the consumer belongs.
- **Ssl_cafile:** This is the file in pem format containing the client certificate, as well as any ca certificates needed to establish the certificates authenticity. Default: None

7. Data Ingestion (Stage 5)

7.1 Google Cloud Platform (GCP)

Google Cloud is a suite of Cloud Computing services offered by Google. The platform provided various services like compute, storage, networking, Big Data and many more than running on the same infrastructure that Google uses internally for its end users.

- **Cloud Console URL:** https://console.cloud.google.com/
- **Project:** nfv-pp-tinaa-02
- **Pre-Requisite:** Request access to GCP project you are supposed to work on.
- **Project Role:** Project Editor

7.2 What are Google Cloud Platform (GCP) Services?

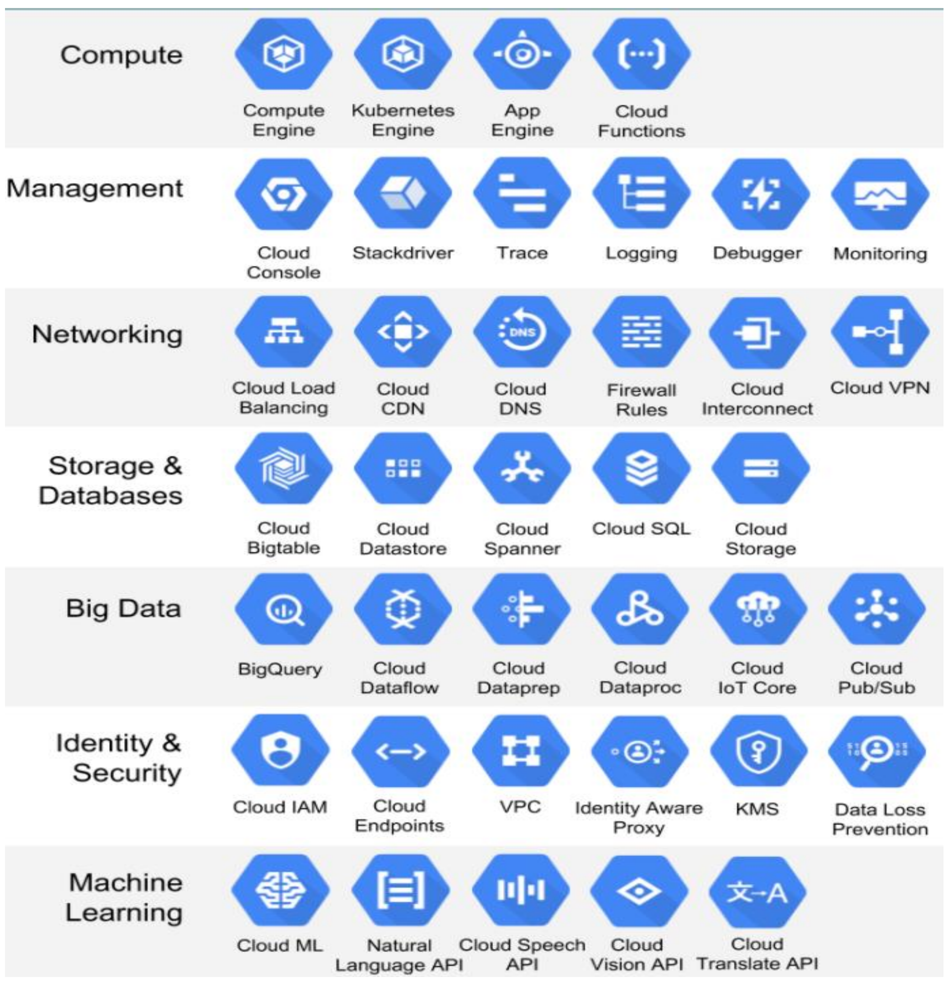


Figure 7. 1 GCP Services

7.3 What are Google Cloud Platform Services used in Data Pipeline Creation?

Google offers a wide range of services. The following are the Google cloud services used for creating the optimized data pipeline [17]:

1. Networking
2. Storage and Databases
3. Big Data
4. Identity & Security

1. **Networking:** The storage domain includes services related to networking; it includes the following services [17].

    - Google Virtual Private Cloud (VPC)

2. **Storage and Databases**: The storage domain includes services related to data storage; it includes the following services [17]

    - Google Cloud Storage

3. **Big Data:** The storage domain includes services related to big data; it includes the following services [17]

    - Google BigQuery
    - Google Cloud DataFlow
    - Google Cloud Pub/Sub

4. **Identity and Security:** The storage domain includes services related to security; it includes the following services [17].

    - Google Cloud IAM

7.4 Selection of resources for Optimized Data Pipeline

7.4.1 Apache Kafka/Cloud Kafka Vs Google PubSub [18]:

Point Of Difference	Apache Kafka	Google PubSub
Type	-Streaming log -Open Source or on Cloud	-Message Queue -Managed by Google
Recommended	If solution requires some Spark processing	If solution require process data in Streaming also need to support batch processing.
Authentication	Supports open authentication and encryption mechanisms	Authentication is based on GCPs IAM system
Push/Pull Mechanism	Kafka Consumer are Pull.	Pub/Sub consumers choose between a push or a pull mechanism.
Provisioning	Manually manage offsets of messages using external storage and Clusters.	Managed by GCP and scalable
Consumer Group and Subscriptions	Uses consumer group and Partition. When message is read from a specific partition, then any other consumer process which belong to same consumer group will not be able to read that message (because offset will eventually increase)	PubSub uses Subscriptions once message is read and ack the message for the subscription is gone.

Table 7. 1 Comparison between Apache Kafka and Google PubSub

7.4.2 Google DataFlow Vs Google Data Proc [19]:

Point Of Difference	Google DataFlow	Google DataProc
Unique For	ETL-Batch and Stream processing of Data Workloads	ETL-Batch processing workloads
Provisioning	Serverless, Automatic provisioning of clusters. Fully automated provisioning of clusters	Provisioning of clusters is done manually
System Integration	BigQuery and BigTable Apache Beam	Apache Spark and Hadoop
Ease Of Use	Relatively Easy to Use	Simple, easy to Use
Recommended for	Creating new pipelines on cloud	Migrating existing pipeline

Table 7. 2Comparison between Google DataFlow and DataProc

7.4.2 Google BigQuery Vs Google BigTable [20]:

Point Of Difference	Google BigQuery	Google BigTable
Type	Relational SQL for OLAP workloads	No SQL , suitable for OLTP workloads
Provisioning	Querying fully managed Data Warehousing Interactive querying, offline analytics	AdTech, Financial and IOT data Flat data, heavy read/write events, analytical data [20]
System Integration	Batch/Stream	Put row
Ease Of Use	SELECT rows	Scan rows
Recommended for	With BigQuery it is possible to run complex analytical SQL-based queries under large sets of data. BigQuery is a smart choice for queries that require a table scan or when you need to look across entire database.	NO SQL database service and does not support SQL or multirow transitions. So, this makes it unsuitable for a wide range of applications. Only suitable for multiple data sets. Overhead is too high
Latency	Query latency is slow, so best used for running queries with heavy workloads. It is immutable in nature means queries are executed efficiently in parallel [21].	Ideal for storing huge single-keyed data with low latency and can support high read and write throughput at low latency [21].

Table 7. 3Comparison between Google BigQuery and BigTable

NOTE: The above difference shows why PubSub is preferred over Kafka/Cloud Kafka, Google DataFlow over Google DataProc and Google Big Query over Google BigTable

7.5 Google PubSub:

Using GCP python publisher the netflow records are published to PubSub Topic in GCP

7.5.1 What is PubSub?

Google Cloud Pub/Sub is a fully managed messaging service for exchanging event data among applications and services [22].

- **Message**: The data that moves through the service.
- **Topic**: a named entity that represents a feed of messages.
- **Subscription**: a named entity that represents an interest in receiving messages on a particular topic.
- **Publisher**: creates messages and sends them to the messaging service on a specified topic.
- **Subscriber**: receives messages on a specified subscription.

7.5.1.2 PubSub Basics Overview:

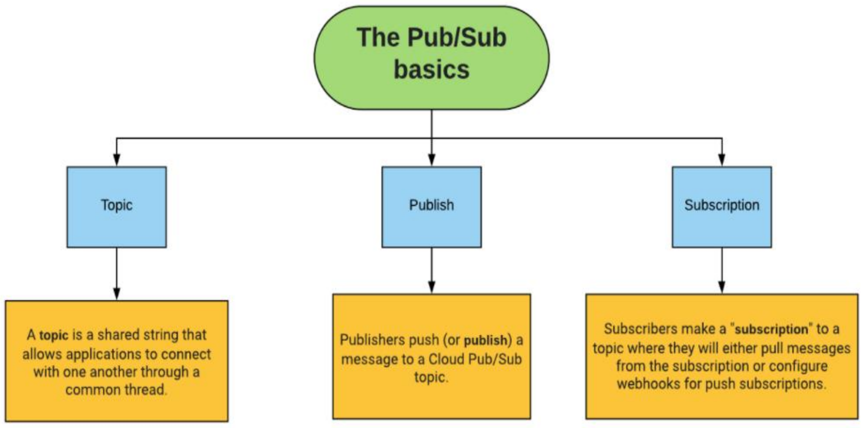

The Pub/Sub basics
Topic
Publish
Subscription
A topic is a shared string that allows applications to connect with one another through a common thread.
Publishers push (or publish) a message to a Cloud Pub/Sub topic.
Subscribers make a "subscription" to a topic where they will either pull messages from the subscription or configure webhooks for push subscriptions.

Figure 7. 2 Google Pubsub basics

7.5.1.3 PubSub Functioning:

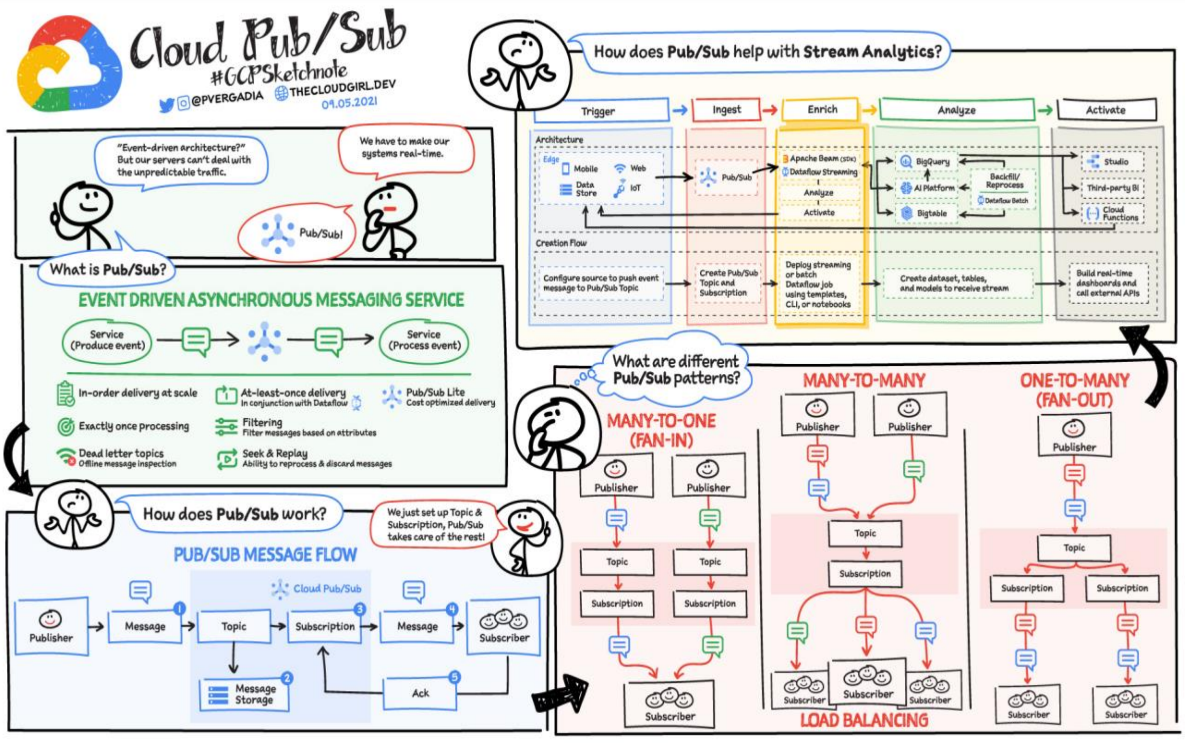

Cloud Pub/Sub
#GCPSketchnote
@PVERGADIA
THECLOUDGIRL.DEV
09.05.2021
"Event-driven architecture?" But our servers can't deal with the unpredictable traffic.
We have to make our systems real-time.
Pub/Sub!
What is Pub/Sub?
EVENT DRIVEN ASYNCHRONOUS MESSAGING SERVICE
Service (Produce event)
Service (Process event)
In-order delivery at scale
At-least-once delivery In conjunction with Dataflow
Pub/Sub Lite Cost optimized delivery
Exactly once processing
Filtering Filter messages based on attributes
Dead letter topics Offline message inspection
Seek & Replay Ability to reprocess & discard messages
How does Pub/Sub work?
We just set up Topic & Subscription, Pub/Sub takes care of the rest!
PUB/SUB MESSAGE FLOW
Cloud Pub/Sub
Publisher
Message
Topic
Subscription
Message
Subscriber
Message Storage
Ack
How does Pub/Sub help with Stream Analytics?
Trigger
Ingest
Enrich
Analyze
Activate
Architecture
Edge
Mobile
Web
Data Store
IoT
Pub/Sub
Apache Beam (SDK)
Dataflow Streaming
Analyze
Activate
BigQuery
AI Platform
Bigtable
Backfill/Reprocess
Dataflow Batch
Studio
Third-party BI
Cloud Functions
Creation Flow
Configure source to push event message to Pub/Sub Topic
Create Pub/Sub Topic and Subscription
Deploy streaming or batch Dataflow job using templates, CLI, or notebooks
Create dataset, tables, and models to receive stream
Build real-time dashboards and call external APIs
What are different Pub/Sub patterns?
MANY-TO-ONE (FAN-IN)
MANY-TO-MANY
ONE-TO-MANY (FAN-OUT)
Publisher
Topic
Subscription
Subscriber
LOAD BALANCING

Figure 7. 3 End to End functionality of Google PubSub

7.6 Identity and Access Management (IAM):

IAM lets administrators authorize who can take action on specific resources, giving you full control and visibility to manage Google Cloud resources centrally [23].

- IAM Enable the IAM API.

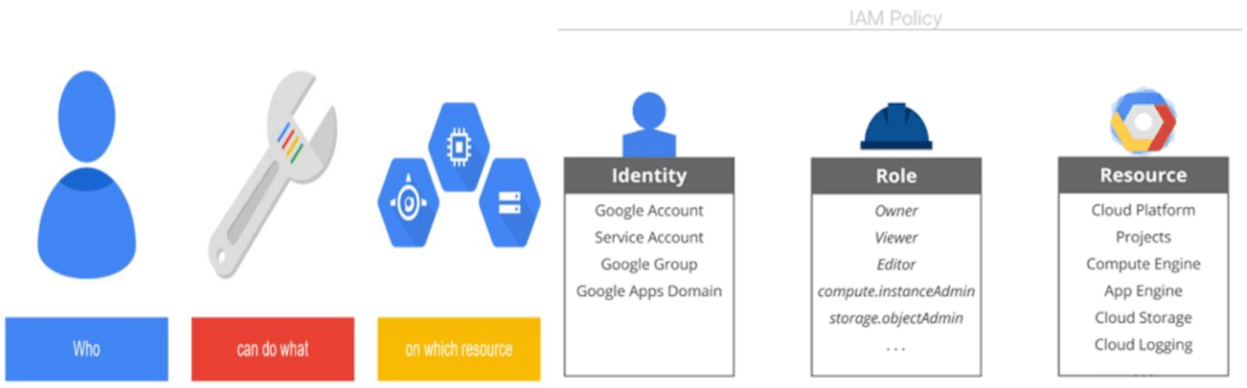


Figure 7. 4 Identity and Access Management (IAM) policy in GCP

7.6.1 Service Account:

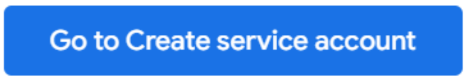
A service account is a special type of Google account intended to represent a non-human user that needs to authenticate and be authorized to access data in Google APIs. In our scenario service accounts are used in running workloads on on-premises workstations or data centers that call Google APIs [24].

Project Service Accounts:

- sarwar.sulthana@telus.com
- kafkatogcptest@nfv-pp-tinaa-02.iam.gserviceaccount.com

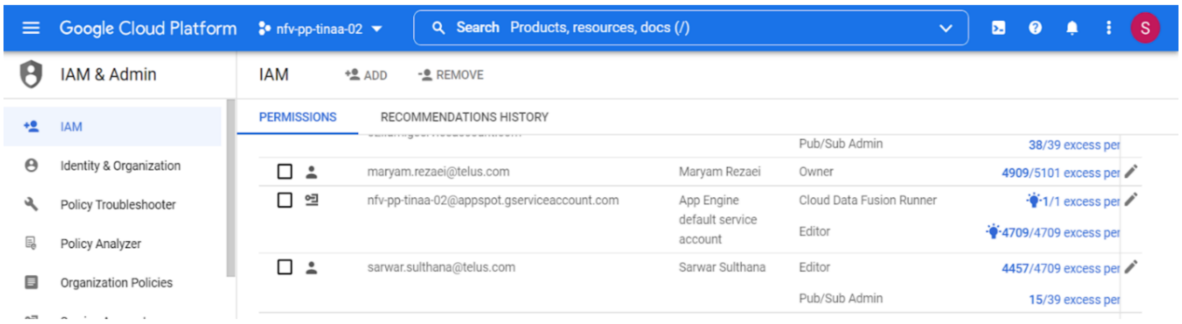
7.6.1.1 Steps for creating a Service Account *[25]*:

1. In the Cloud Console, go to the **Create Service account** page



2. Select project.
3. Enter a service account name to display in the cloud Console.
4. Optional: Enter a description of the service account.
5. If you do not want to set access controls now, click **Done** to finish creating the service account. To set access controls now, click **Create and continue** to the next step.

6. Optional: choose one or more IAM roles to grant to the service account on the project.
7. When you are done adding roles, click **Continue** [25].
8. Optional: In the **service account users role** field, add members that can impersonate the service account.
9. Optional: In the **Service account admin's role** field, add members that can manage the service account.
10. Click **Done** to finish creating the service account.



		Pub/Sub Admin	38/39 excess per
maryam.rezaei@telus.com	Maryam Rezaei	Owner	4909/5101 excess per
nfv-pp-tinaa-02@appspot.gserviceaccount.com	App Engine default service account	Cloud Data Fusion Runner	1/1 excess per
		Editor	4709/4709 excess per
sarwar.sulthana@telus.com	Sarwar Sulthana	Editor	4457/4709 excess per
		Pub/Sub Admin	15/39 excess per

Figure 7. 5 Service Account in GCP

7.6.2 Service Account key:

You can use service account key files to authenticate an application as a service account.

7.6.2.1 Creating and managing service account keys [26]:

To use a service account from outside of Google Cloud, such as another platform or on-premises, you must first establish the identity of the service account. Public/private key pairs provide a secure way of accomplishing this goal. When you create a service account key, the public portion is stored on Google Cloud, while private portion is available only to you.

Service account key creation steps from cloud Console [26]:

1. In the Cloud Console, go to the **Service accounts** page.

Go to Service accounts

2. Select a project [26].
3. Click the email address of the service account that you want to create a key for.
4. Click the **keys** tab.
5. Click the **Add key** drop-down menu, then select **Create new key.**
6. Select **JSON** as the **key type** and click **Create**.

Note: After you download the key file, you cannot download it again

7.7 Data Ingestion:

7.7.1 Pre-Requisites:

- Create a Service Account in the GCP console.
- Create a topic in PubSub
- Create a Subscription or use the default subscription created.
- Ensure Service account key has Pub/Sub Admin or Project Admin role

7.7.2 Service account access control:

You can control access via Google cloud Console (manage access control for your topics and projects).

To set access controls at the project level [27]:

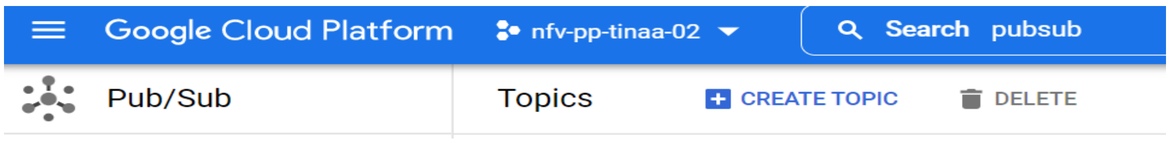
1. Open the IAM page in the cloud Console.
2. Select your project and click **Continue**.
3. Click, **Add principal**
4. Enter the email address of a new principal to whom you have not granted any IAM role previously.
5. Select a role from the drop-down menu.
6. Click Add.
7. Verify that the principal is listed under the role that you granted.

For Project Service account: Set role = PubSub Admin, BigQuery Admin

7.8 PubSub Topic and Subscription:

7.8.1 Steps to create Pub/Sub Topic [28]:

1. Go to Pub/Sub topics page in the Cloud Console
2. Click **Create Topic.**

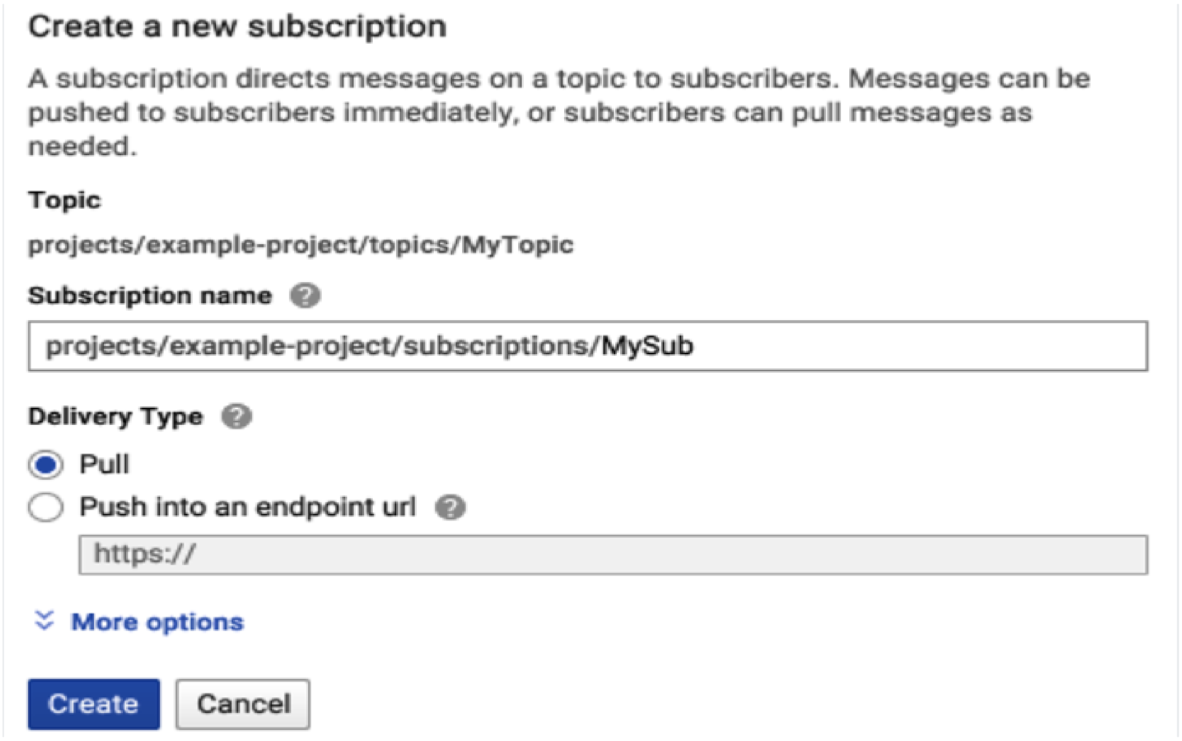


3. In the **Topic ID** field, provide a unique topic name, for example, MyTopic
4. Click **Save**

7.8.2 Add a Subscription [28]:

To add a subscription to the topic you just created.

5. Display the menu for the topic you just created, click **Create subscription.**
6. Type a name for the subscription such as **MySub.**



Create a new subscription

A subscription directs messages on a topic to subscribers. Messages can be pushed to subscribers immediately, or subscribers can pull messages as needed.

Topic

projects/example-project/topics/MyTopic

Subscription name

projects/example-project/subscriptions/MySub

Delivery Type

[✓ Pull]

Push into an endpoint url

https://

7. Leave the delivery type as **Pull**.
8. Click **Create**.

7.9 Publishing Messages to PubSub Topic (GCP Publisher Client):

How messages are published from Kafka Consumer to GCP PubSub topic:

Technical Specifications:

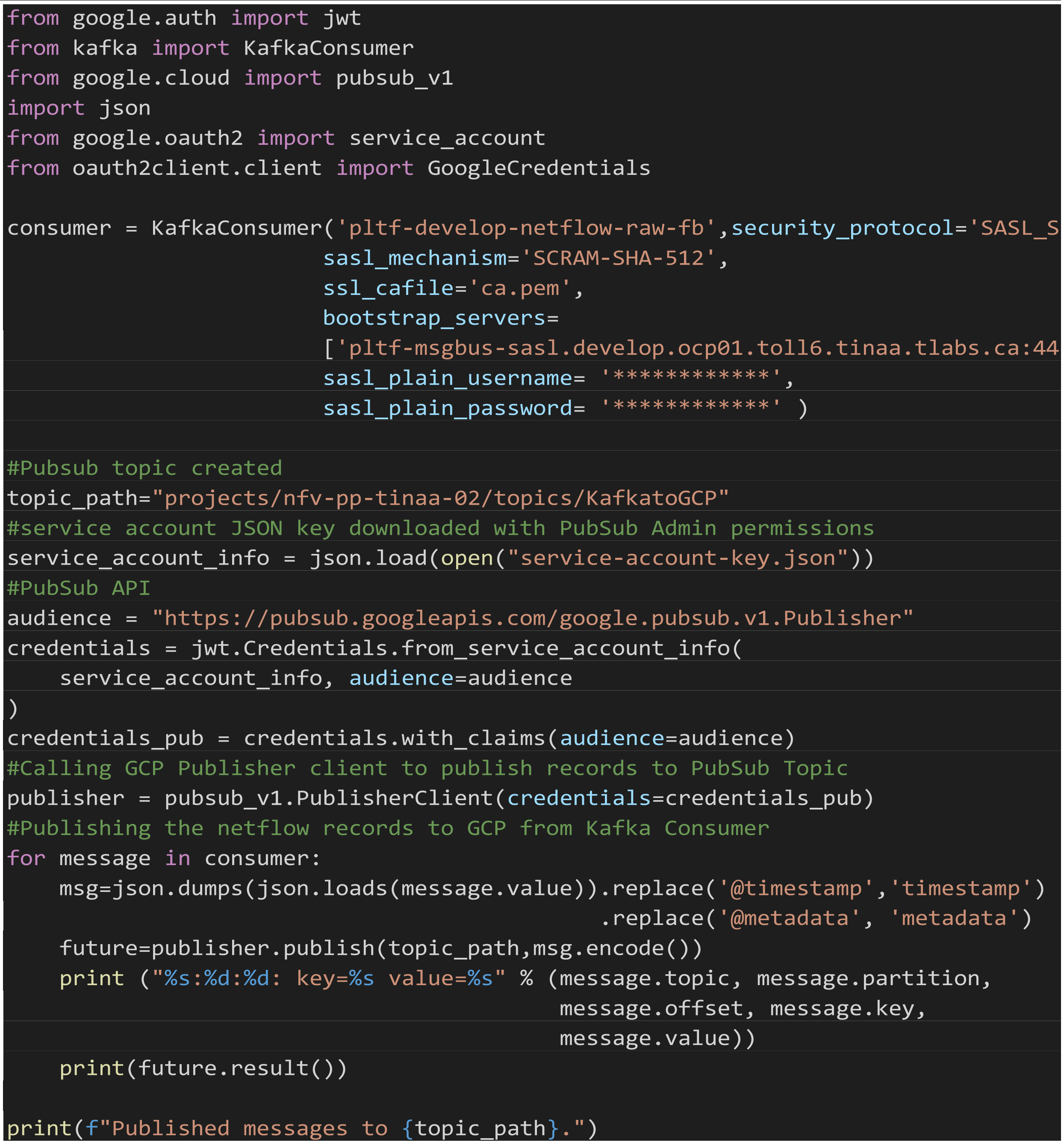
Python Libraries: Google credentials, service account, pubsub_v1

Package Installation:

- pip install google-cloud-language
- pip install --upgrade google-api-python-client oauth2client
- pip install google
- pip install google-cloud-pubsub
- pip install google-cloud

7.9.1 GCP Python Publisher Code Snippet:

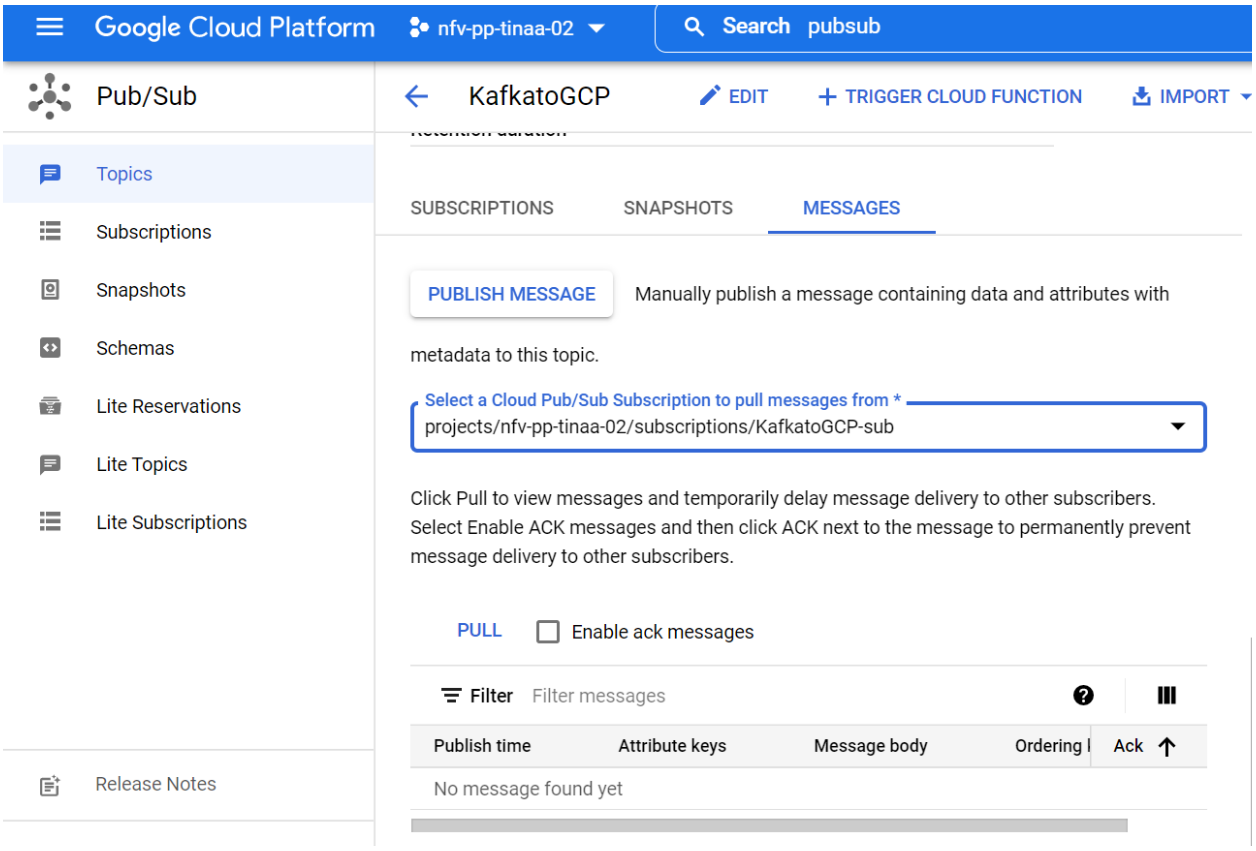
Publishing events to a PubSub topic (read Queue) from Kafka topic is as simple as the code snippet given below



```python
from google.auth import jwt
from kafka import KafkaConsumer
from google.cloud import pubsub_v1
import json
from google.oauth2 import service_account
from oauth2client.client import GoogleCredentials

consumer = KafkaConsumer('pltf-develop-netflow-raw-fb',security_protocol='SASL_S
                        sasl_mechanism='SCRAM-SHA-512',
                        ssl_cafile='ca.pem',
                        bootstrap_servers=
                        ['pltf-msgbus-sasl.develop.ocp01.toll6.tinaa.tlabs.ca:44
                        sasl_plain_username= '************',
                        sasl_plain_password= '************' )

#Pubsub topic created
topic_path="projects/nfv-pp-tinaa-02/topics/KafkatoGCP"
#service account JSON key downloaded with PubSub Admin permissions
service_account_info = json.load(open("service-account-key.json"))
#PubSub API
audience = "https://pubsub.googleapis.com/google.pubsub.v1.Publisher"
credentials = jwt.Credentials.from_service_account_info(
    service_account_info, audience=audience
)
credentials_pub = credentials.with_claims(audience=audience)
#Calling GCP Publisher client to publish records to PubSub Topic
publisher = pubsub_v1.PublisherClient(credentials=credentials_pub)
#Publishing the netflow records to GCP from Kafka Consumer
for message in consumer:
    msg=json.dumps(json.loads(message.value)).replace('@timestamp','timestamp')
                                          .replace('@metadata', 'metadata')
    future=publisher.publish(topic_path,msg.encode())
    print ("%s:%d:%d: key=%s value=%s" % (message.topic, message.partition,
                                          message.offset, message.key,
                                          message.value))
    print(future.result())

print(f"Published messages to {topic_path}.")
```

Now the Netflow records are published to GCP PubSub Topic.

Pull: Select the Cloud Pub/Sub subscription to pull messages from:



Manually publish a message containing data and attributes with metadata to this topic.

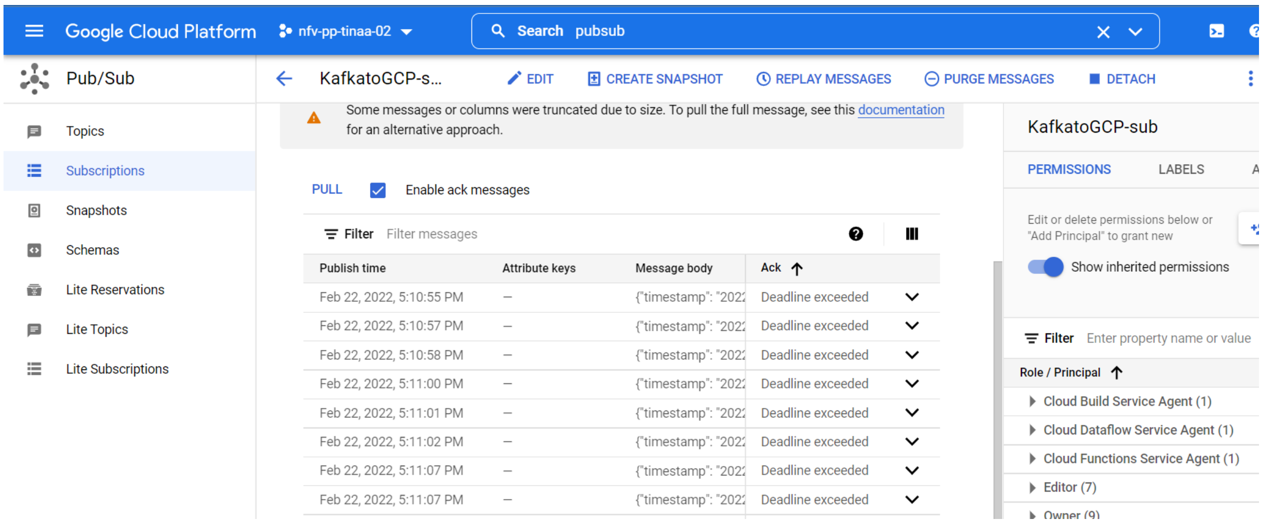
Select a Cloud Pub/Sub Subscription to pull messages from *
projects/nfv-pp-tinaa-02/subscriptions/KafkatoGCP-sub

Click Pull to view messages and temporarily delay message delivery to other subscribers. Select Enable ACK messages and then click ACK next to the message to permanently prevent message delivery to other subscribers.

Publish time	Attribute keys	Message body	Ordering	Ack
No message found yet				

Figure 7. 6 Message Publish and Acknowledgment

Click **Pull** to view all the message that are published



Some messages or columns were truncated due to size. To pull the full message, see this documentation for an alternative approach.

Publish time	Attribute keys	Message body	Ack
Feb 22, 2022, 5:10:55 PM	–	{"timestamp": "202	Deadline exceeded
Feb 22, 2022, 5:10:57 PM	–	{"timestamp": "202	Deadline exceeded
Feb 22, 2022, 5:10:58 PM	–	{"timestamp": "202	Deadline exceeded
Feb 22, 2022, 5:11:00 PM	–	{"timestamp": "202	Deadline exceeded
Feb 22, 2022, 5:11:01 PM	–	{"timestamp": "202	Deadline exceeded
Feb 22, 2022, 5:11:02 PM	–	{"timestamp": "202	Deadline exceeded
Feb 22, 2022, 5:11:07 PM	–	{"timestamp": "202	Deadline exceeded
Feb 22, 2022, 5:11:07 PM	–	{"timestamp": "202	Deadline exceeded

Edit or delete permissions below or "Add Principal" to grant new

Role / Principal
Cloud Build Service Agent (1)
Cloud Dataflow Service Agent (1)
Cloud Functions Service Agent (1)
Editor (7)
Owner (9)

Figure 7. 7 Netflow records published

8. Data Processing (Stage 5)

8.1 Google Cloud Dataflow:

Cloud Dataflow is a fully managed service for running Apache Beam pipelines on Google Cloud Platform. Cloud Dataflow executes data processing jobs. Dataflow is designed to run on a very large dataset, it distributes these processing tasks to several virtual machines in the cluster so they can process different chunks of data in parallel [29].

Below are the few reasons why Dataflow is chosen over DataProc for processing.

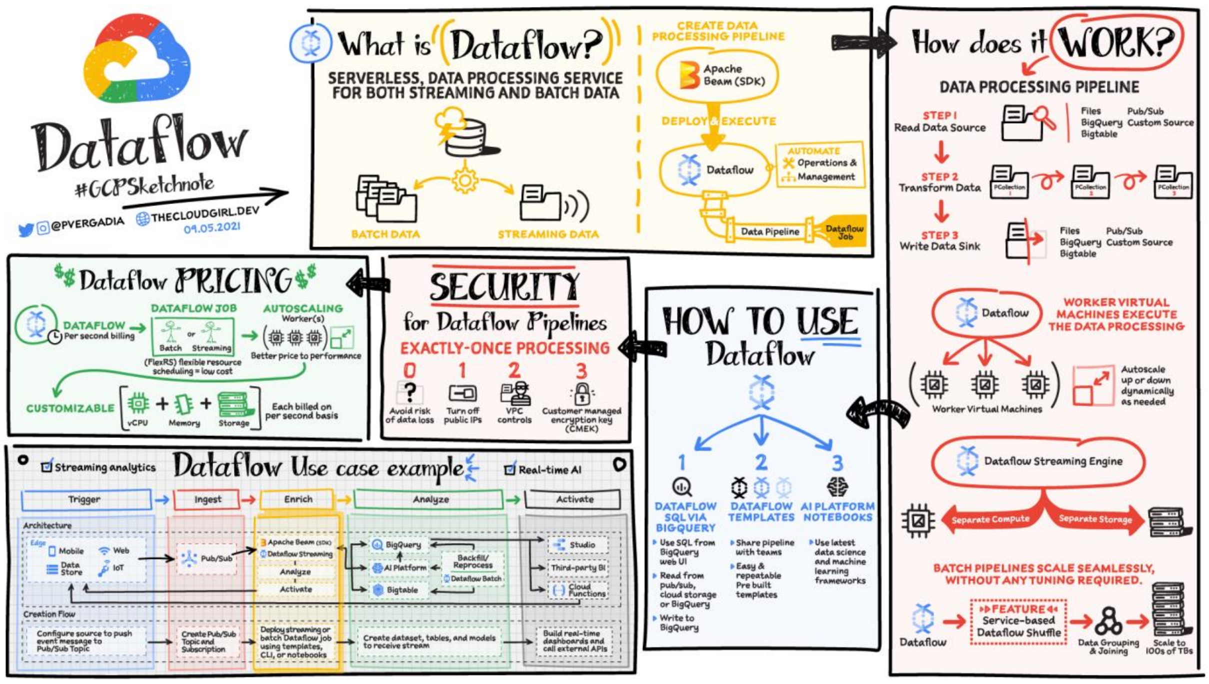
1. **Serverless:** We don't have to manage computing resources. It will automatically spins up and down a cluster of virtual machines while running the processing jobs. We can just focus on building the code instead of building the cluster. Apache Spark, on the other hand, requires more configuration even if it is running on Cloud Dataproc [29].
2. **Processing code is separate from the execution environment:** In 2016, Google donated open-source Dataflow SDK and a set of data connectors to access Google Cloud Platform which added additional features to the Apache Beam project. We can write beam programs and run them on the local system or Cloud Dataflow service. When we look at the Dataflow documentation, it suggests the Apache Beam website for the latest version of the Software Development Kit [29].
3. **Processing batch and stream mode with the same programming model:** Other Big data SDKs require different codes depending on whether data comes in batch or streaming form. On the other hand, Apache Beam addresses it with a unified programming model. Competitors like Spark are considering it but they are not quite there yet [29].

8.1.1 Project Pre-Requisites:

We are using a PubSub Subscription to BigQuery template to create a dataflow Job (to publish the records from PubSub to BigQuery Table).
DataFlow Job pre-requisites

- Create a VPC and Subnet
- Existing PubSub Topic and Subscription
- Table created in Big Query
- Cloud Storage

8.2 Cloud Data flow Functioning:



Dataflow
#GCPSketchnote
@PVERGADIA
THECLOUDGIRL.DEV
04.05.2021
What is Dataflow?
SERVERLESS, DATA PROCESSING SERVICE FOR BOTH STREAMING AND BATCH DATA
BATCH DATA
STREAMING DATA
CREATE DATA PROCESSING PIPELINE
Apache Beam (SDK)
DEPLOY & EXECUTE
Dataflow
AUTOMATE Operations & Management
Data Pipeline
Dataflow Job
How does it WORK?
DATA PROCESSING PIPELINE
STEP 1 Read Data Source
Files Pub/Sub BigQuery Custom Source Bigtable
STEP 2 Transform Data
PCollection
STEP 3 Write Data Sink
Files Pub/Sub BigQuery Custom Source Bigtable
WORKER VIRTUAL MACHINES EXECUTE THE DATA PROCESSING
Worker Virtual Machines
Autoscale up or down dynamically as needed
Dataflow Streaming Engine
Separate Compute
Separate Storage
BATCH PIPELINES SCALE SEAMLESSLY, WITHOUT ANY TUNING REQUIRED.
FEATURE Service-based Dataflow Shuffle
Data Grouping & Joining
Scale to 100s of TBs
Dataflow PRICING
DATAFLOW Per second billing
DATAFLOW JOB
Batch or Streaming
(FlexRS) flexible resource scheduling = low cost
AUTOSCALING Worker(s)
Better price to performance
CUSTOMIZABLE
vCPU + Memory + Storage
Each billed on per second basis
SECURITY for Dataflow Pipelines
EXACTLY-ONCE PROCESSING
0 Avoid risk of data loss
1 Turn off public IPs
2 VPC controls
3 Customer managed encryption key (CMEK)
HOW TO USE Dataflow
1 DATAFLOW SQL VIA BIGQUERY
Use SQL from BigQuery web UI
Read from pub/sub, cloud storage or BigQuery
Write to BigQuery
2 DATAFLOW TEMPLATES
Share pipeline with teams
Easy & repeatable Pre built templates
3 AI PLATFORM NOTEBOOKS
Use latest data science and machine learning frameworks
Streaming analytics
Dataflow Use case example
Real-time AI
Trigger
Ingest
Enrich
Analyze
Activate
Architecture
Edge
Mobile
Web
Data Store
IoT
Pub/Sub
Apache Beam (SDK)
Dataflow Streaming
Analyze
Activate
BigQuery
AI Platform
Bigtable
Backfill/Reprocess
Dataflow Batch
Studio
Third-party BI
Cloud Functions
Creation Flow
Configure source to push event message to Pub/Sub Topic
Create Pub/Sub Topic and Subscription
Deploy streaming or batch Dataflow job using templates, CLI, or notebooks
Create dataset, tables, and models to receive stream
Build real-time dashboards and call external APIs

Figure 8. 1 End to End Functionality of Google Dataflow

8.3 Cloud VPC

8.4.1 What is VPC:

A VPC is a secure, isolated private cloud hosted within a public cloud [30].

8.4.2 VPC Components:

1. **VPC Networks**

A Virtual Private Cloud (VPC) network is a virtual version of a physical network, implemented inside of Google’s production network, using Andromeda. VPC networks along with their associated routes and firewall rules, are global resources i.e., they are not associated with any distinct region or zone [31].

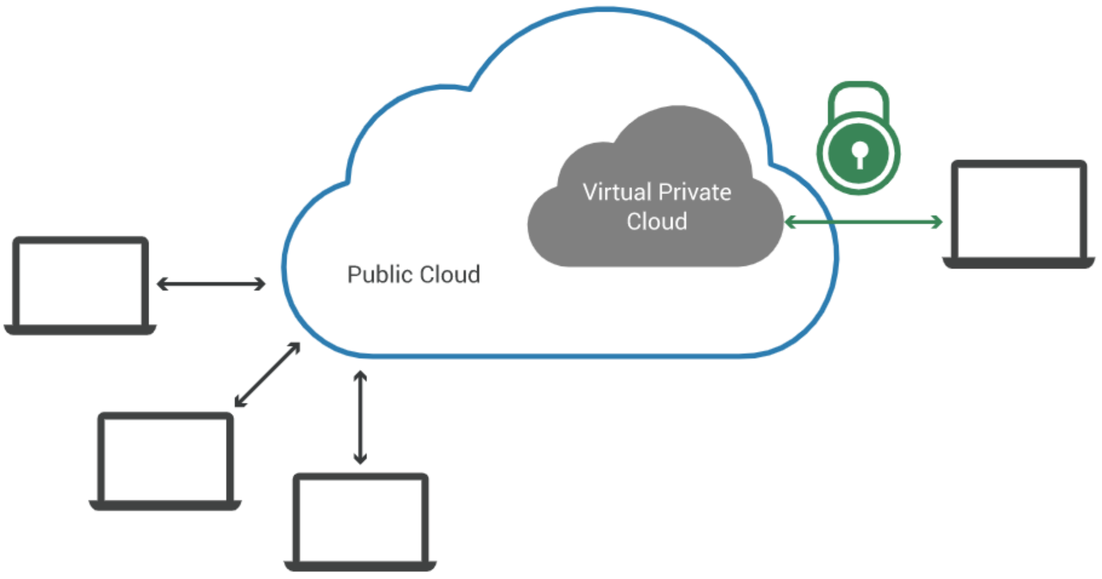


Figure 8. 2 Cloud VPC hosted within the public environment

A VPC network provides the following:

- Connects to on-premises networks using Cloud VPN tunnels and Cloud Interconnect attachments.
- Subnets.

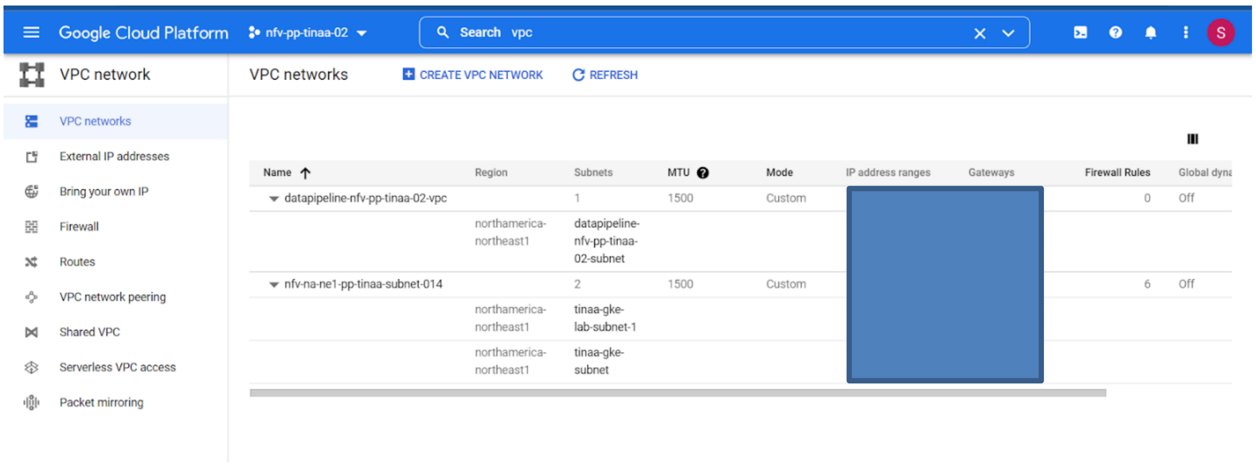
2. Subnets:

Each VPC network consists of one or more useful IP range partitions called subnets and each subnet is associated with a region. VPC networks do not have any IP address ranges associated with them, IP ranges are defined for the subnets. Subnets are regional resources. Each subnet defines a range of IP addresses.

Please Note: The terms subnet and subnetwork are synonymous. They are used interchangeably in Google Cloud Console, gcloud commands, and API documentation. However, a subnet is not the same thing as a VPC network. They both represent different types of objects in Google Cloud.

VPC created is global and we can use it in any of the regions. However, subnets are regional. Subnets you create can span across all these zones across multiple availability zones in a specific region.

In our Google cloud account, we will have a default VPC, which has a default subnet in each of the regions.



Name	Region	Subnets	MTU	Mode	IP address ranges	Gateways	Firewall Rules	Global dyna
datapipeline-nfv-pp-tinaa-02-vpc		1	1500	Custom	[redacted]	[redacted]	0	Off
	northamerica-northeast1	datapipeline-nfv-pp-tinaa-02-subnet			[redacted]	[redacted]		
nfv-na-ne1-pp-tinaa-subnet-014		2	1500	Custom	[redacted]	[redacted]	6	Off
	northamerica-northeast1	tinaa-gke-lab-subnet-1			[redacted]	[redacted]		
	northamerica-northeast1	tinaa-gke-subnet			[redacted]	[redacted]		

Figure 8. 3 VPC network details

1.) Why do we need VPC in GCP? [31]

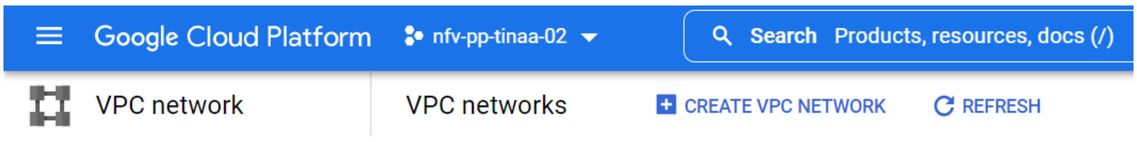
Answer: Google Cloud VPCs let users increase the IP space of any subnets without any workload shutdown or downtime which in return gives them flexibility and growth options to meet their needs.

2.) What is a subnet in the cloud? [31]

Answer: Subnets are a logical partition of an IP network into multiple, smaller network segments. The Internet Protocol (IP) is the method for sending data from one computer to another over the internet. Each computer, or host, on the internet, has at least one IP address as a unique identifier.

8.4.3 Steps for creating a VPC [32]:

1. Go to the VPC networks page in the Google Cloud Console.

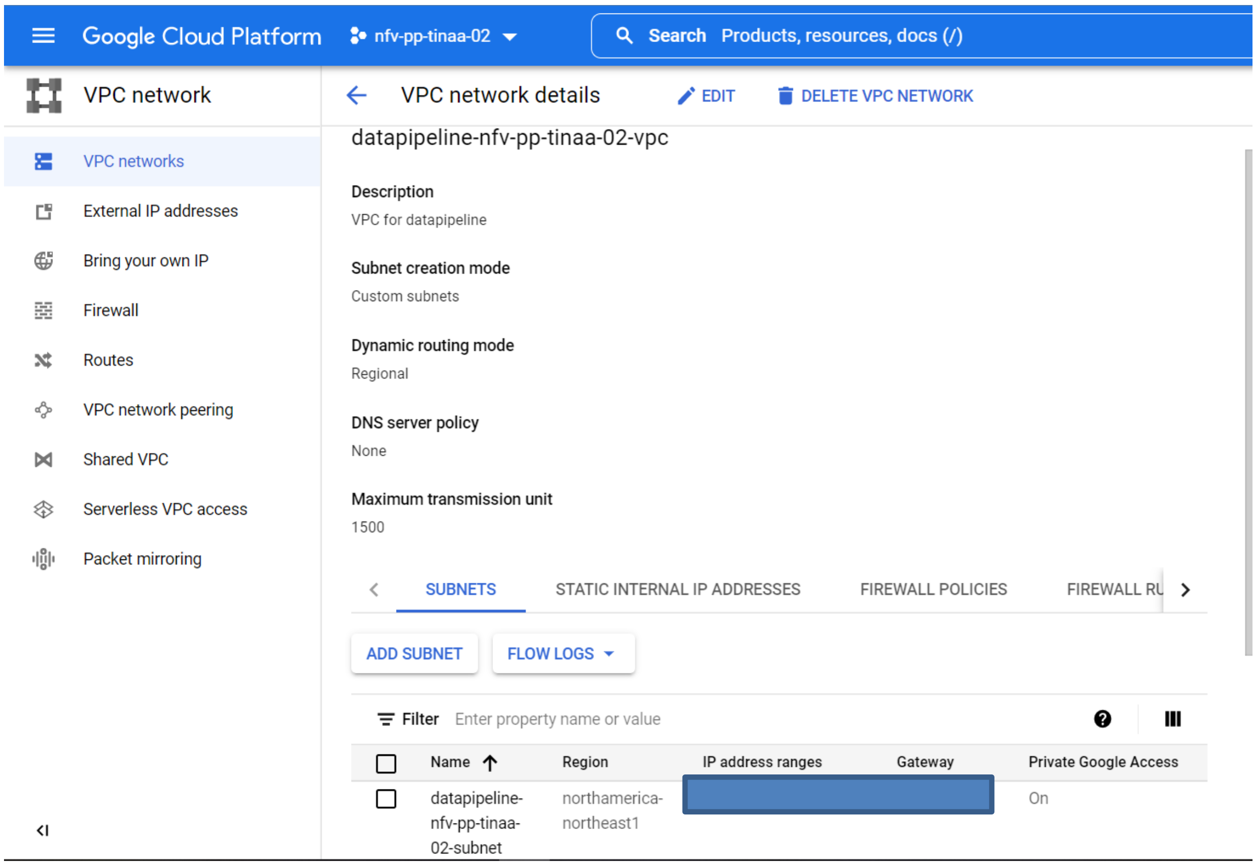


2. Click **Create VPC network.**
3. Enter a **Name** for the network.
4. Choose Automatic for the **Subnet** creation mode.
5. In the **Firewall rules** section, select zero or more predefined firewall rules. The rules address common use cases for connectivity to instances.

   Whether or not you select pre-defined rules, you can create your own firewall rules after you create the network.
6. Choose the Dynamic routing mode for the VPC network.

   For more information, see dynamic routing mode. You can change the dynamic routing mode later.

7. **Maximum transmission unit (MTU):** Choose whether the network has an MTU of 1460 (default) or 1500. Review the MTU information in the concepts guide before setting the MTU to 1500.
8. Click **Create.**



datapipeline-nfv-pp-tinaa-02-vpc

Description
VPC for datapipeline

Subnet creation mode
Custom subnets

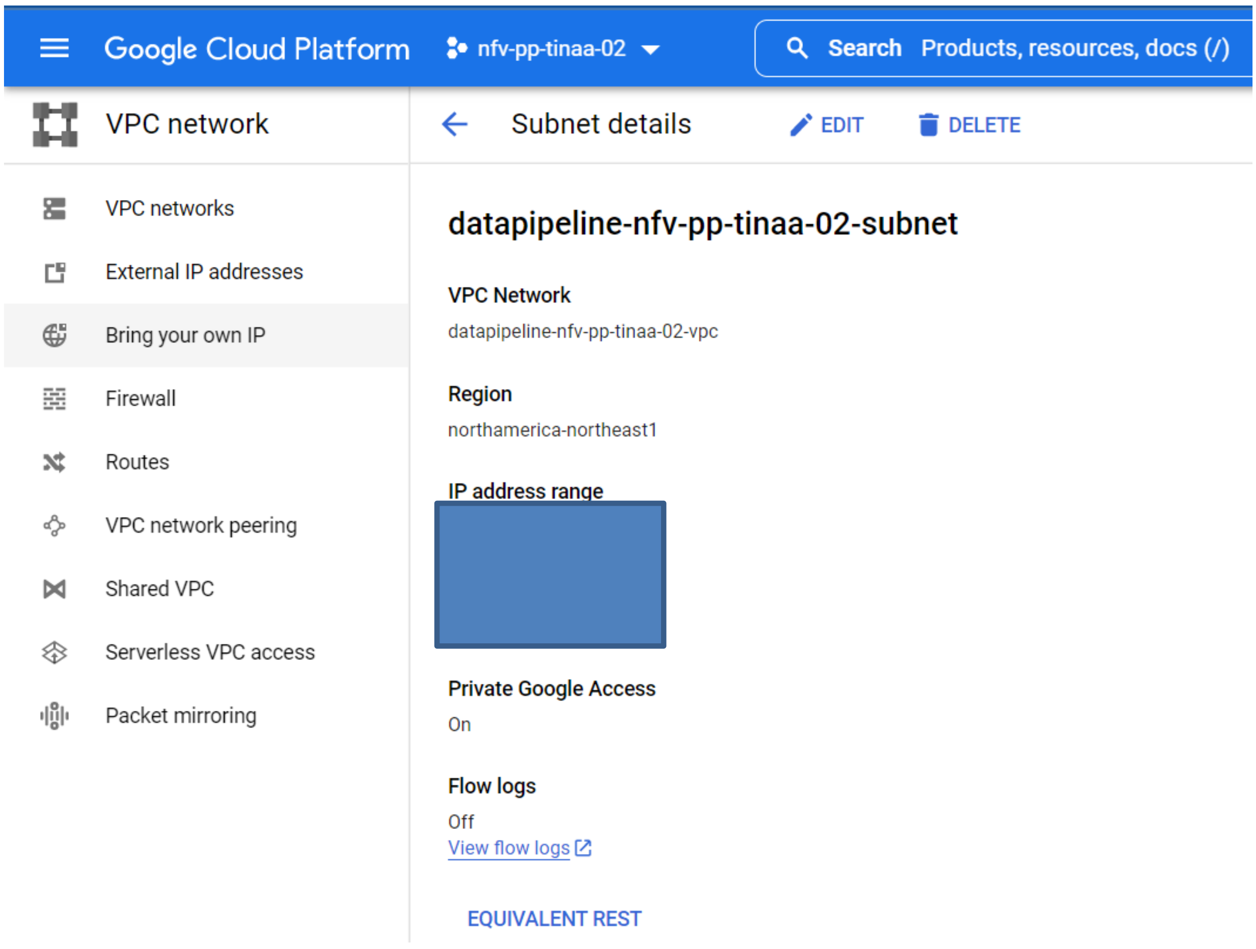
Dynamic routing mode
Regional

DNS server policy
None

Maximum transmission unit
1500

Name	Region	IP address ranges	Gateway	Private Google Access
datapipeline-nfv-pp-tinaa-02-subnet	northamerica-northeast1	[redacted]	[redacted]	On

Figure 8. 4 VPC network summary



Google Cloud Platform nfv-pp-tinaa-02 Search Products, resources, docs (/)

VPC network | Subnet details | EDIT | DELETE

- VPC networks
- External IP addresses
- Bring your own IP
- Firewall
- Routes
- VPC network peering
- Shared VPC
- Serverless VPC access
- Packet mirroring

datapipeline-nfv-pp-tinaa-02-subnet

VPC Network
datapipeline-nfv-pp-tinaa-02-vpc

Region
northamerica-northeast1

IP address range
[redacted]

Private Google Access
On

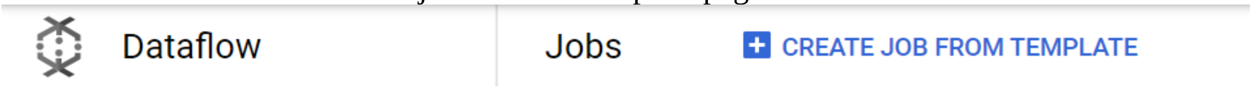
Flow logs
Off
View flow logs

EQUIVALENT REST

Figure 8. 5 VPC Subnet summary

8.4 Creating a DataFlow Job template [33]:

1. Go to the Dataflow → Create job from the template page.



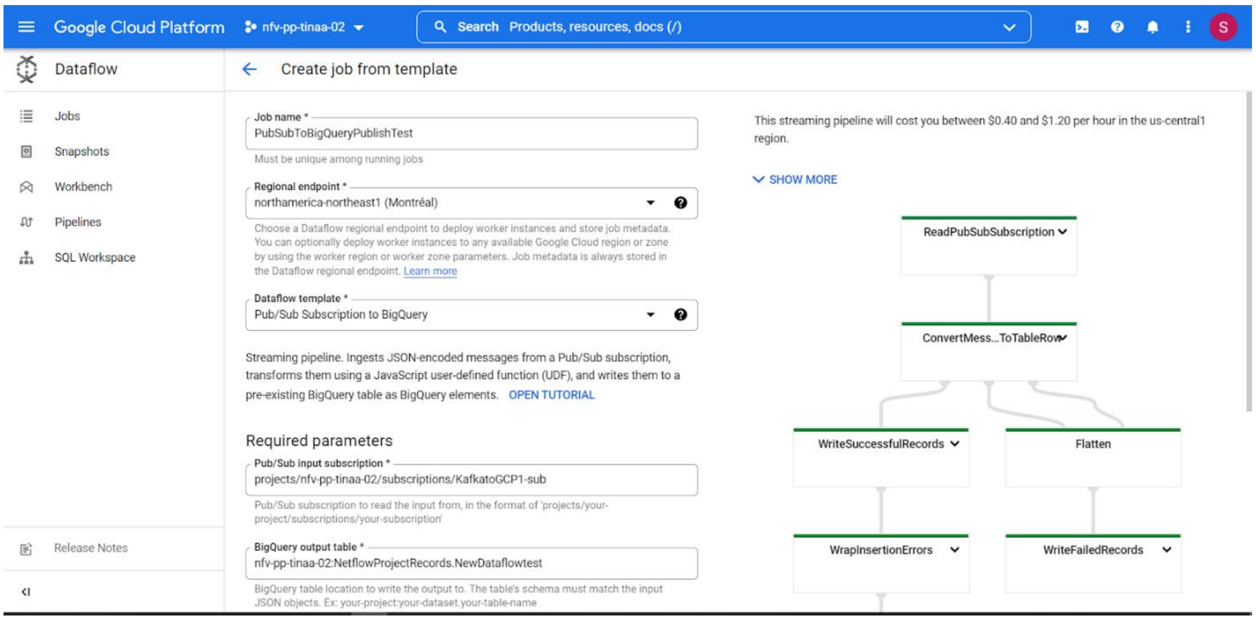
Dataflow | Jobs | CREATE JOB FROM TEMPLATE

2. Go to Create job from template In the Job name field, enter a unique job name.
3. **Optional:** For Regional endpoint, select a value from the drop-down menu. The default regional endpoint is us-central1.

   For a list of regions where you can run a Dataflow job, see Dataflow locations.
4. From the Dataflow template drop-down menu, select the required template.
5. In the parameter fields provided, enter your parameter values.
6. **Click Run job.**

8.3.1 Pub/Sub Subscription to BigQuery:

The Pub/Sub subscription to BigQuery template is a streaming pipeline that reads the JSON-formatted messages from Pub/Sub subscription and writes them to a BigQuery table. You can use the template as a solution to move Pub/Sub data to BigQuery. The template reads JSON formatted messages from Pub/Sub and converts them to BigQuery elements [34].



Job name *
PubSubToBigQueryPublishTest
Must be unique among running jobs

Regional endpoint *
northamerica-northeast1 (Montréal)
Choose a Dataflow regional endpoint to deploy worker instances and store job metadata. You can optionally deploy worker instances to any available Google Cloud region or zone by using the worker region or worker zone parameters. Job metadata is always stored in the Dataflow regional endpoint. Learn more

Dataflow template *
Pub/Sub Subscription to BigQuery

Streaming pipeline. Ingests JSON-encoded messages from a Pub/Sub subscription, transforms them using a JavaScript user-defined function (UDF), and writes them to a pre-existing BigQuery table as BigQuery elements. OPEN TUTORIAL

Required parameters

Pub/Sub input subscription *
projects/nfv-pp-tinaa-02/subscriptions/KafkatoGCP1-sub
Pub/Sub subscription to read the input from, in the format of 'projects/your-project/subscriptions/your-subscription'

BigQuery output table *
nfv-pp-tinaa-02:NetflowProjectRecords.NewDataflowtest
BigQuery table location to write the output to. The table's schema must match the input JSON objects. Ex: your-project:your-dataset.your-table-name

This streaming pipeline will cost you between $0.40 and $1.20 per hour in the us-central1 region.

Figure 8. 6 Dataflow Job Template

When you are all set, click Run Job and wait for Dataflow to execute the template, which takes a few minutes.

Template Parameters:

Parameter	**Description**
input Subscription	The Pub/Sub input subscription to read from. For example, projects/<project>/subscriptions/<subscription> [35].
outputTableSpec	The BigQuery output table location, in the format of <my-project>:<my-dataset>. <my-table>
dataset	(Optional) The type of logs sent via Pub/Sub, for which we have an out-of-the box dashboard. Known log types of values are audit, vpcflow, and firewall. Default: pubsub.

9. Data Warehouse (Stage 6)

9.1 Google BigQuery:

Google BigQuery is an enterprise data warehouse built using BigTable and Google Cloud Platform. It is a serverless, highly scalable data warehouse that comes with a built-in query engine. The query engine can run SQL queries on terabytes of data in a matter of seconds, and petabytes in a minute. You get this performance without having to manage any infrastructure and without having to create or rebuild indexes.

9.1.1 BigQuery Functioning:

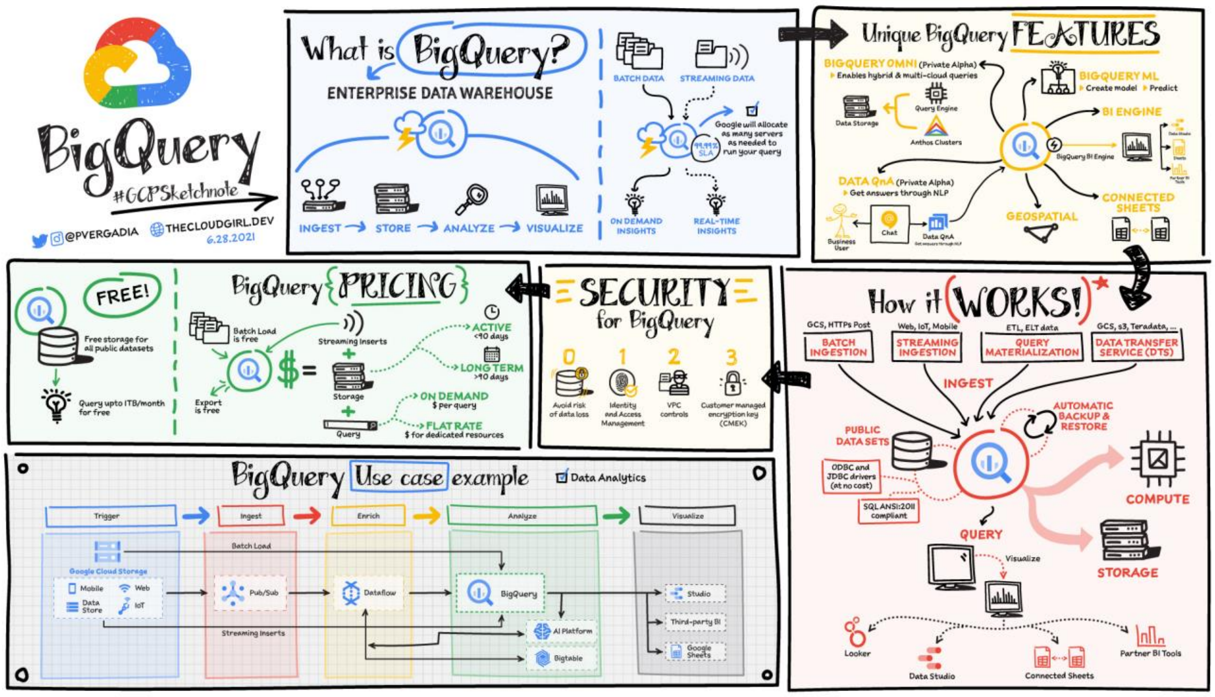


Figure 9. 1 End to End Functionality of Google BigQuery

9.1.2 Project pre-Requisites:

- Create BigQuery Dataset
- Create BigQuery Table
- Import the required Schema for Table
- Ensure Service account has BigQuery admin privileges.

9.2 How to use Google BigQuery:

Consumers can easily access this service from their familiar cloud interface console. Before creating a table in BigQuery, first:

- Create a BigQuery dataset

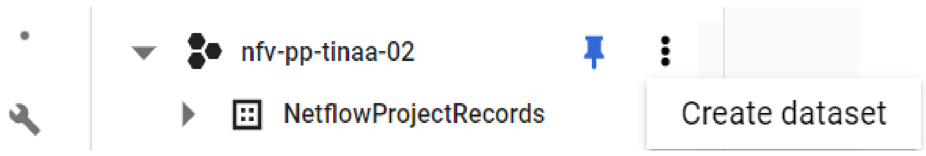
9.2.1 Creating Datasets:

You can create datasets in the following ways [36]:

1. Using the Cloud Console
2. Using SQL query.
3. Using Client libraries.
4. Copying an existing dataset.

9.2.1.1 Steps for creating a dataset using Cloud Console:

1. Open the BigQuery page in the Cloud Console.



2. In the Explorer panel, select the project where you want to create the dataset.
3. Expand the **: Actions** option and click dataset.
4. On the Create **dataset** page:
   - For **Dataset ID**, enter a unique dataset name.
   - For **Data Location**, choose a geographical location for the dataset. After a dataset is created, the location can't be changed.
5. Click **Create dataset**.

9.2.2 BigQuery Table Creation:

Now that the BigQuery dataset is created, we can create a table in BigQuery in the following ways [37]:

1. Manually using the cloud Console
2. Programmatically by calling the tables.insert API method.
3. By using the client libraries.
4. From query results.
5. By using a CREATE TABLE data definition language (DDL) statement.

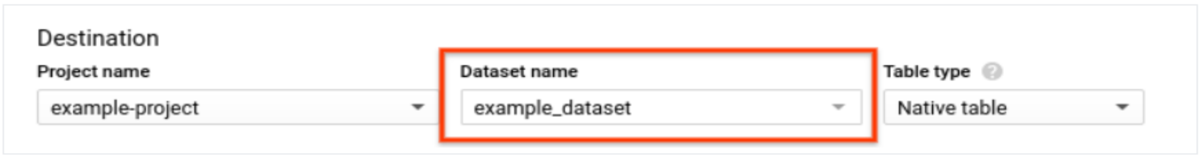
9.2.2.1 Required Permissions:

To create a table, you need the following IAM permissions [38]:

- bigquery.tables.create
- bigquery.tables.updateData
- bigquery.jobs.create

9.2.2.2 Steps for Creating an empty table with a schema definition:

1. In the Cloud Console, open the BigQuery page.
2. In the **Explorer** panel, expand your project and select a dataset.
3. Expand the **: Actions** and click **Open**.
4. In the details panel, click **Create table**.
5. On the **Create table** page, in the **Source** selection, select **Empty table**
6. On the **Create table** page, in the **Destination** selection:
   - For **Dataset name**, choose the appropriate dataset.



Destination

Project name: example-project

Dataset name: example_dataset

Table type: Native table

   - In the Table name field, enter the name of the table you're creating in BigQuery.
   - Verify the Table type is set to Native table.
7. In the Schema section, enter the schema definition.
   - Enter the schema information manually
8. For **Partition and cluster settings** leave the default value – No partitioning.
9. In the Advanced options section, for **Encryption** leave the default value: Google-Managed key. By default, Big Query encrypts customer content stored at rest.
10. Click **Create Table.**

9.2.3 Schema Creation:

Key components of BigQuery Schema [39]:

1. **Column Names:** In the column Name, you are required to specify the parameter for which each column is responsible such as Date, User_Id, Products, etc.

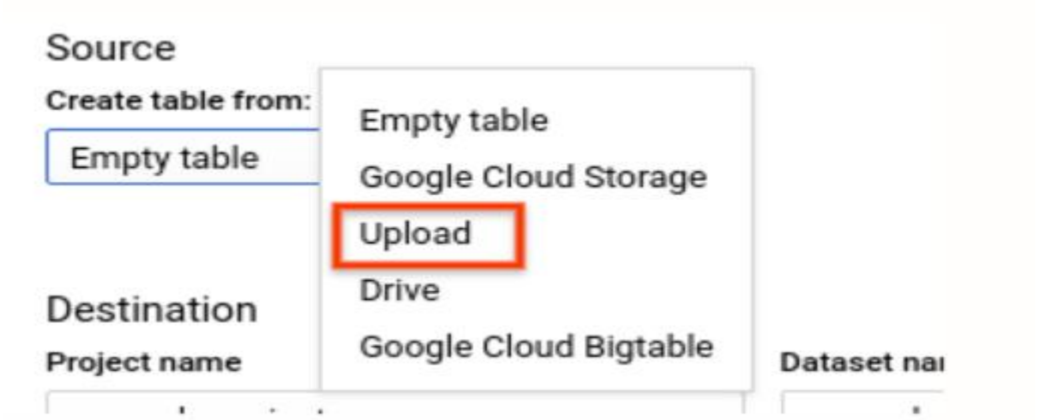
   NOTE: Column names cannot start by Prefixes: _Table_, @File. Duplicate names are not allowed either.

2. **Data Type:** BigQuery also lets you specify the Data Type. Some of the major ones that you can use to specify in your schema are **INT64**, **FLOAT64**, **NUMERIC**, **BOOLEAN**, **STRING**.
3. **Modes**: BigQuery supports some Modes. These are particularly: Nullable, Required, Repeated. In the
    - **Nullable Mode:** Null values are allowed.
    - **Required Mode:** Does not allow any null values.
    - **Repeated Mode:** Contains an array of values of the specified type in column.

9.2.3.1 Specify Schema in BigQuery Table *[39]*:

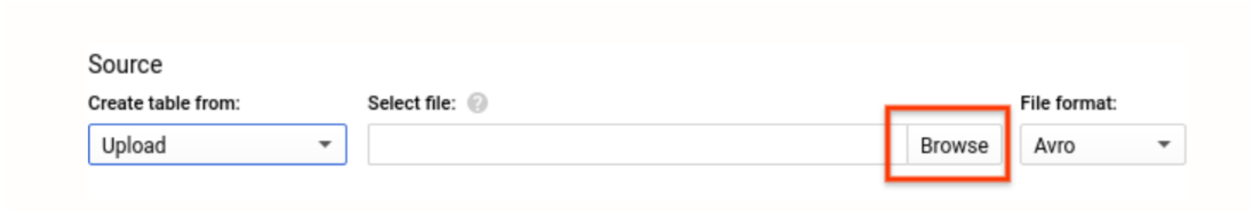
In our project: JSON file is uploaded to BigQuery Table using the Web User Interface: You can directly load a JSON/CSV file from your Local/ GCS Bucket directly into a BigQuery table using WebUI. Better yet, the process will automatically create the table as well:

1. Open the **BigQuery web UI in the Cloud Console.**
2. Choose the right project in the Resources dropdown present at the top header, then expand your Google Cloud project and click to select a dataset.
3. On the right side of the window, in the details panel, click Create table (or Plus + symbol) The process for loading data is the same as the process for creating an empty table [39].
4. You will see a Popuppage. On the create table page.
5. From **Create table from**, select **Upload**.



(**Note**: Choose Google Cloud Storage if your source file resides in GCS.)

6. Click on Browse and choose the file (with netflow data - Key Value pairs).


Source

Create table from: Upload

Select file:

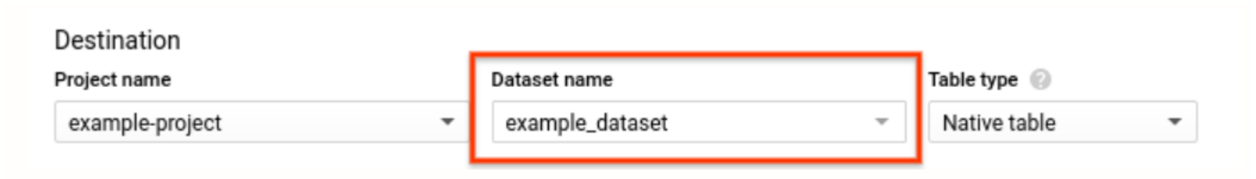
File format: Avro

Browse to the file and click Open. Note that wildcards and comma-separated lists are not supported for local files.

For File format, select JSON (newline delimiters), AVRO, CSV, Parquet or ORC.

7. Scroll down, to the Destination section:

   For Dataset name, choose the appropriate dataset.


Destination

Project name: example-project

Dataset name: example_dataset

Table type: Native table

   In the Table name field, enter the name of the table you're creating in BigQuery.

8. In the Schema section, enter the schema definition.

   For CSV and JSON files, you can check the Auto-detect option to enable schema to auto-detect. Schema information is self-described in the source data for other supported file types.

   You can also enter the schema information manually by: Clicking Edit as text and entering the table schema as a JSON array (or) Using add field to manually input the schema [39].

9. Finally, Select the applicable items in the Advanced options section and then click **Create Table.**

10. Software Pipeline:

10.1 Deployment Architecture

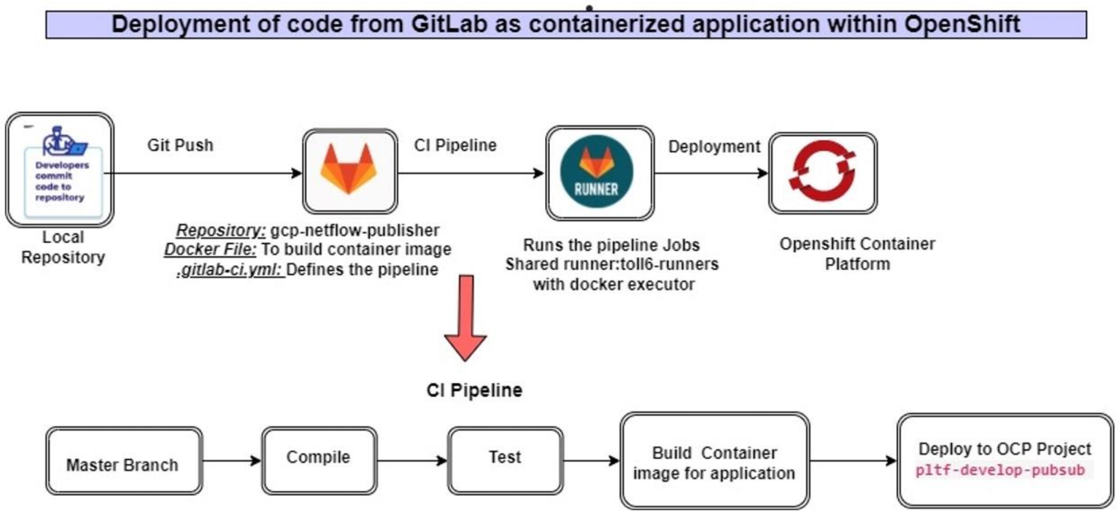

Deployment of code from GitLab as containerized application within OpenShift
Developers commit code to repository
Local Repository
Git Push
Repository: gcp-netflow-publisher
Docker File: To build container image
.gitlab-ci.yml: Defines the pipeline
CI Pipeline
RUNNER
Runs the pipeline Jobs
Shared runner:toll6-runners
with docker executor
Deployment
Openshift Container Platform
CI Pipeline
Master Branch
Compile
Test
Build Container image for application
Deploy to OCP Project
pltf-develop-pubsub

Figure 10. 1 Application Deployment (CI Pipeline)

10.2 Deploying to OpenShift Container Platform:

10.2.1 SSH Keys for GitLab

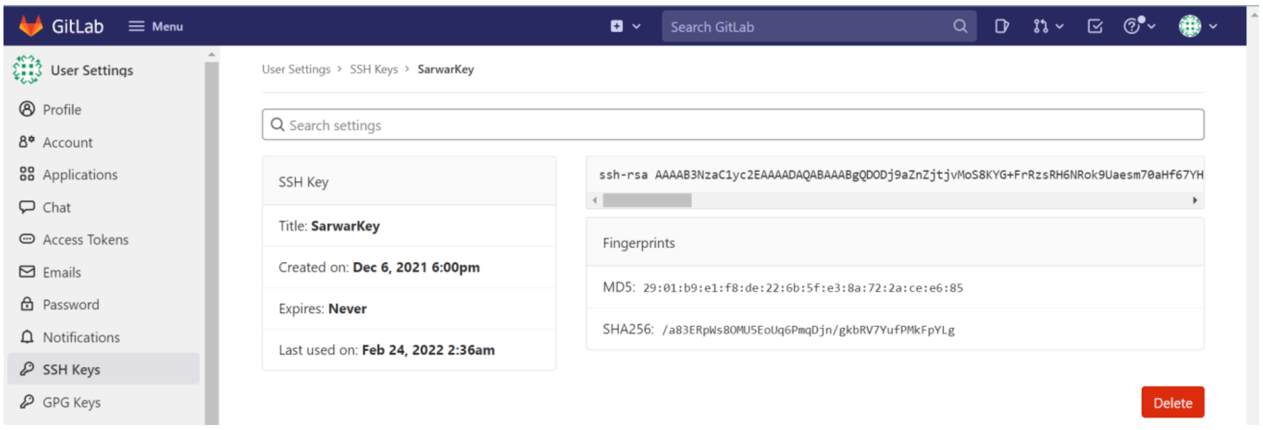
10.2.1.1 What is SSH key

- *SSH – Secured Shell* [40]
    1. Used for Authentication
    2. By setting up the ssh key you can connect to GitLab server without using username and password each time
- *Steps to create SSH keys:* [40]
    1. Run command **ssh-keygen**
    2. *On Mac*: run command on terminal; *On Windows*: Use Putty or git bash
    3. Enter the file location in which you want to save the SSH key.
    4. Navigate to root folder (c:/users/administrator) → Go to .ssh folder → Open the **pub key file** for SSH key generated.

10.2.1.2 Set up SSH Key in GitLab [40]:

- *Steps to set up SSH keys in Gitlab:*
    1. Login to TINAA GitLab.
    2. Go to account→ Settings/Preferences→ SSH keys→ Add an SSH key → Paste the Public SSH key from **step 4 (or from the pub key file generated in .ssh folder)**

3. Give the title for individual SSH key → Click Add key button.



SSH Key

Title: **SarwarKey**

Created on: **Dec 6, 2021 6:00pm**

Expires: **Never**

Last used on: **Feb 24, 2022 2:36am**

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDODj9aZnZjtjvMoS8KYG+FrRzsRH6NRok9Uaesm70aHf67YH
```

Fingerprints

MD5: `29:01:b9:e1:f8:de:22:6b:5f:e3:8a:72:2a:ce:e6:85`

SHA256: `/a83ERpWs8OMU5EoUq6PmqDjn/gkbRV7YufPMkFpYLg`

Figure 10. 2 SSH Keys for Gitlab

10.2.2 Git Push Steps:

- ***Push the code/commit the code changes from local to gitlab repository (TINAA).***
  1. Create a project in GitLab (**gcp-netflow-publisher**) [41].
  2. **Git clone** "*https://gitlab.tinaa.tlabs.ca/tinaa-platform/pub-sub/publishers/gcp-netflow-publisher*"
  3. Cd change your directory to that repository (in local).
  4. **git branch -a**
  5. git switch -c new-branch (develop) [41]
  6. Check if you are switched to a new branch
  7. Push all the files in your local repo to GitLab = **git add .**
  8. **git commit -m "message/comment"** [41]
  9. **git push origin** (origin being the remote server that we have cloned from- GitLab TINAA project)
  10. **git push –set-upstream origin new-branch.**
  11. **Git commit -m "message"**
  12. **Git push –set -upstream origin new-branch.** [41]

10.2.3 Docker Image and Containerization?

10.2.3.1 Docker:

It is an open-source software platform for building, deploying, and managing testing applications quickly. It enables developers to package applications into containers. that have everything the software needs to run including libraries, system tools, code, and runtime. Using Docker, you can quickly deploy and scale applications into any environment and know your code will run [42].

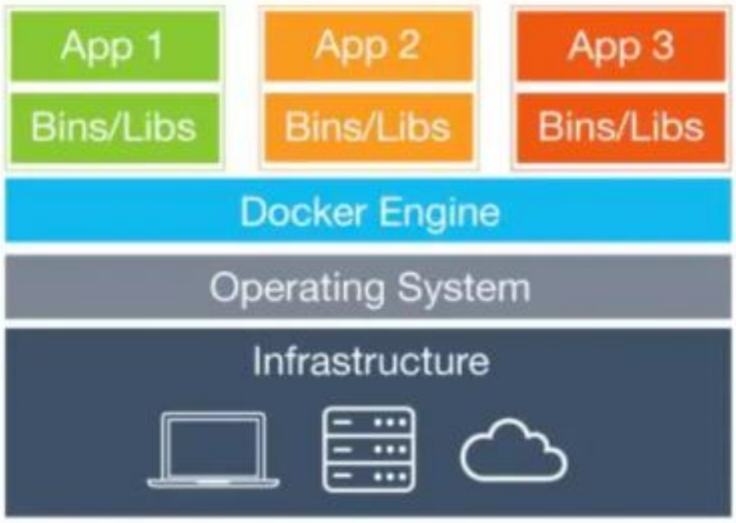


Figure 10. 3 Docker Overview

10.2.3.2 Docker Tools and Terms *[42]*:

Docker File: Every Docker container starts with a simple text file containing instructions for how to build the Docker container image. This text file provides a set of instructions to build a Docker image, including the operating system, languages, environmental variables, file locations, network ports, and any other components it needs to run. Docker file automates the process of Docker image creation. It's essentially a list of command-line interface (CLI) instructions that Docker Engine will run to assemble the image [43].

Docker Images: These contain executable application source code as well as the tools, libraries, and dependencies that the application code needs to run as a container. When you run the Docker image, it becomes an instance (or multiple instances) of the container [43].

Docker Containers: Docker containers are live, running instances of Docker images. While Docker images are read-only files, containers are live, ephemeral, executable content. Users can interact with them, and administrators can adjust their settings and conditions using docker commands.

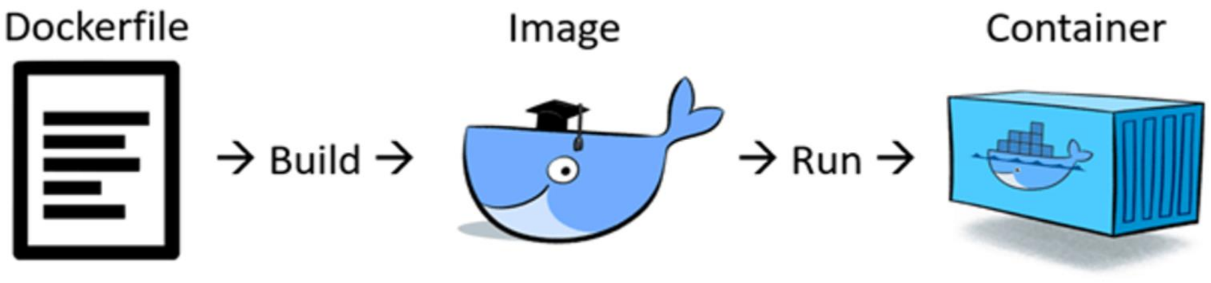


Figure 10. 4 Docker Image Build and Containerization

10.2.3.3 Docker File for Kafka Consumer and GCP Publisher:

```
#FROM python:3.9-alpine
FROM python:3.10

# Allow statements and log messages to immediately appear in the Cloud Run logs
ENV PYTHONUNBUFFERED True
# Update the timezone
#RUN apk update && apk add git openssh
#RUN pip install google-cloud-pubsub==2.3.0
RUN pip install wheel
#RUN pip3 install --upgrade pip setuptools wheel
RUN git config --global http.sslVerify "false"
# create work directory and assign permissions to user for work directory
RUN mkdir -p /opt/kafka-GCP-consumer/

# copy code and config files to work directory
COPY . /opt/kafka-GCP-consumer/
# COPY conf/development/ /opt/plan-file-consumer/conf/

# Specify the user to execute all commands below
USER root
#ENV PYTHONPATH "${PYTHONPATH}:/opt/kafka-GCP-consumer"
WORKDIR /opt/kafka-GCP-consumer/
# COPY requirements.txt requirements.txt
RUN pip install -e . --trusted-host gitlab.tinaa.tlabs.ca --use-
deprecated=legacy-resolver
RUN pip install grpcio-tools
RUN pip install --upgrade google-api-python-client oauth2client
#RUN pip install google-cloud
RUN pip install -e . google-cloud-pubsub==2.3.0

WORKDIR /opt/kafka-GCP-consumer/
CMD python3 MainSript.py
```

10.2.3.4 Commands to build docker image and run docker container (windows)

- Command Build image: **docker build -t sample-test8 .**
- Command Run the image to create a container: **docker run --privileged sample-test8**

10.2.3 Deploy applications on OpenShift with GitLab CI:

10.2.3.1 GitLab CI /CD Pipeline

Now the Code changes are pushed to Gitlab instance. Deploy the changes using the GitLab CI.

GitLab CI/CD is a tool for software development using the continuous methodologies:

10.2.3.1.1 GitLab CI/CD Concepts [44]:

GitLab CI/CD uses several concepts to describe and run your build and deploy.

Concept	Description
Pipelines	Structure your CI/CD process through pipelines.
CI/CD variables	Reuse values based on variable/value key pair
Environments	Deploy your application to different environments (for example develop, staging, test, pre-production , production)
Job Artifacts	Output, use and reuse job artifacts
GitLab runner	Configure your own runners to execute the scripts
Test cases	Create testing scenarios

Table 10. 1Gitlab CI Pipeline keywords

10.2.3.1.2 GitLab CI/CD Configuration:

GitLab CI/CD supports numerous configuration options [44].

Configuration	Description
Pipeline triggers	Triggers pipelines through the API.
. gitlab-ci.yml	This yaml file automatically runs whenever you push a commit to the server.
Merge request pipelines	Design a pipeline structure for running pipeline in merge requests.
Integrate with OpenShift cluster	Connect your project to OpenShift cluster
SSH keys for CI/CD	Using SSH keys in your CI pipelines.

Table 10. 2 Gitlab CI pipeline configurations

10.2.3.2 CI/CD settings:

10.2.3.2.1.gitlab-ci.yml:

This file automatically runs whenever you push a commit on the server. This triggers a notification to the runner you specified in the yaml file and then it processes the series of tasks you specified.

NOTE: Since we are using Docker, the tasks always start with a clean state of the image. This means that all files and modifications that you put or do inside.gitlab-ci.yml, will be reverted each time you push a commit to the server.

- **Stages**: you can define what stage is going to run by first specifying a stage name parent key.
- **Pipelines**: When you push git repo to GitLab with the .gitlab-ci.yml file on it, it will automatically trigger the pipelines. The pipelines are the stages that you defined in your.gitlab-ci.yml.

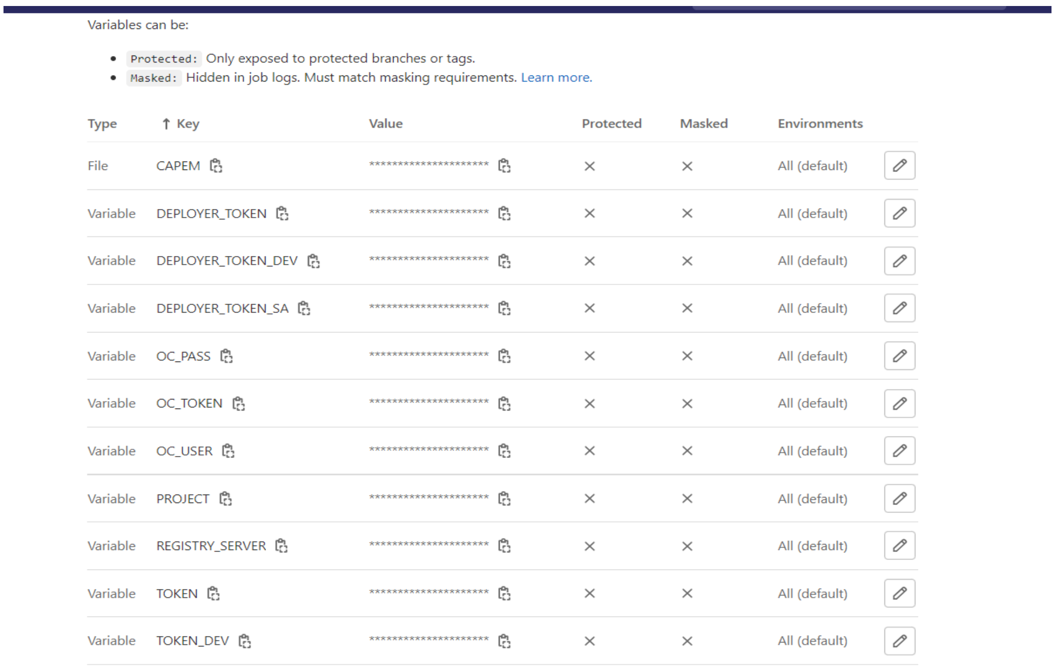
10.2.3.2.2 Environment Variables

Before writing the .gitlab-ci.yml file we need to add some environment variables to the project settings (the OpenShift CLI needs these variables to connect and deploy on the server) [45].

- OPENSHIFT_IP
- OPENSHIFT_PORT
- OPENSHIFT_TOKEN

Variables: Environment variables are applied to environments via the runner. They can be protected by only exposing them to protected branches or tags. You can use environment variables for passwords, secret keys, or whatever you want.

To set the below variables go to GitLab repository → Settings→CI/CD → Expand variables and set the Key values for variables.



Variables can be:

- `Protected:` Only exposed to protected branches or tags.
- `Masked:` Hidden in job logs. Must match masking requirements. Learn more.

Type	↑ Key	Value	Protected	Masked	Environments
File	CAPEM	********************	×	×	All (default)
Variable	DEPLOYER_TOKEN	********************	×	×	All (default)
Variable	DEPLOYER_TOKEN_DEV	********************	×	×	All (default)
Variable	DEPLOYER_TOKEN_SA	********************	×	×	All (default)
Variable	OC_PASS	********************	×	×	All (default)
Variable	OC_TOKEN	********************	×	×	All (default)
Variable	OC_USER	********************	×	×	All (default)
Variable	PROJECT	********************	×	×	All (default)
Variable	REGISTRY_SERVER	********************	×	×	All (default)
Variable	TOKEN	********************	×	×	All (default)
Variable	TOKEN_DEV	********************	×	×	All (default)

Figure 10. 5 Gilab CI Environment variables

For Example: To obtain the value of OPENSHIFT_TOKEN, you need:

- To connect to the web console of your OpenShift cluster as an administrator (the default credentials are system/admin)
- To click on the dropdown list of your profile (on the top right corner)
- And choose this menu item: **Copy Login Command**

You should get something like that in your clipboard:

```
oc login https://192.168.1.100:8443 --token=8W-OX8XB-K7r7P6vmSz_aE7xM2ZK1fnHz-a3aaW89Vs
```

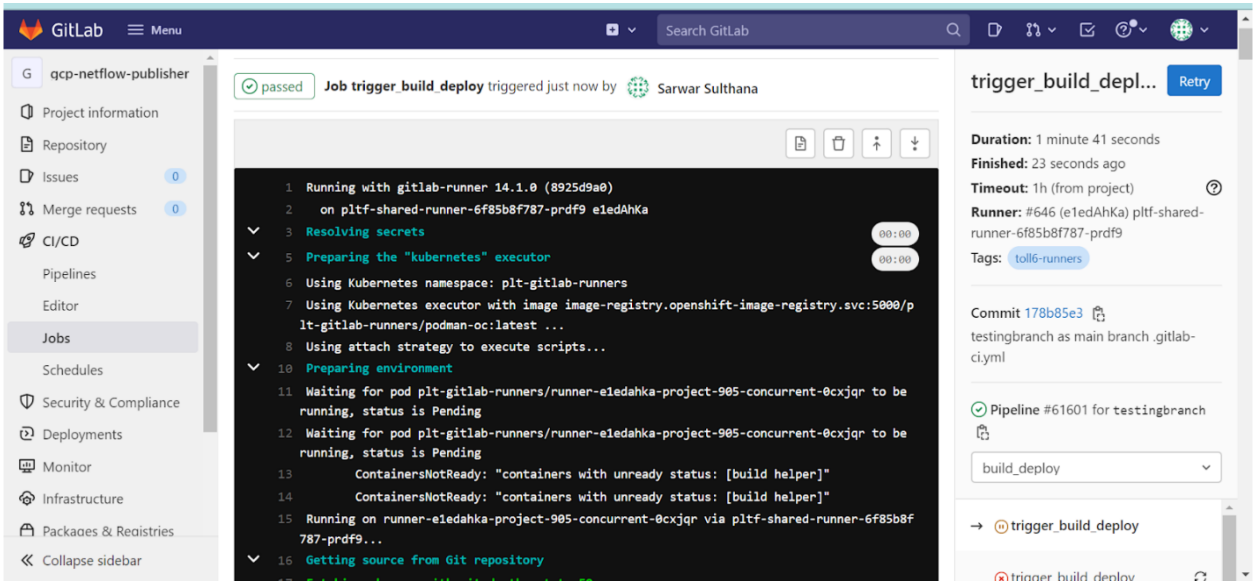
With this command, you can connect to a terminal to your OpenShift cluster. So, now you can set OPENSHIFT_TOKEN with 8W-OX8XB-K7r7P6vmSz_aE7xM2ZK1fnHz-a3aaW89Vs

10.2.3.3 CI/CD Script:

Add a .gitlab-ci.yml file to your project with the below content

```yaml
include:
  - project: 'tinaa-platform/deployment/cicd'
    ref: develop
    file: 'build-deploy/build.gitlab-ci.yml'

variables:
  REGISTRY_USER: builder
  REGISTRY_SERVER: default-route-openshift-image- registry.apps.tinaaocp01.nfvde
  PIPELINE_ID: $CI_JOB_ID
  BUILD_ARG: CI_JOB_TOKEN
  PROJECT_DEV: pltf-develop-pubsub
  SERVICE_IMAGE_NAME: kafkatogcptest2

stages:
  - build_deploy

trigger_build_deploy:
  stage: build_deploy
  only:
    - /(develop|qa|preprod|master)/
```

As soon as you commit the file, your first deployment will start. Then the CI Job will be triggered and based on the docker image the container will be created in the OpenShift registry.



Job trigger_build_deploy triggered just now by Sarwar Sulthana

```
Running with gitlab-runner 14.1.0 (8925d9a0)
  on pltf-shared-runner-6f85b8f787-prdf9 e1edAhKa
Resolving secrets                                                        00:00
Preparing the "kubernetes" executor                                      00:00
Using Kubernetes namespace: plt-gitlab-runners
Using Kubernetes executor with image image-registry.openshift-image-registry.svc:5000/p
lt-gitlab-runners/podman-oc:latest ...
Using attach strategy to execute scripts...
Preparing environment
Waiting for pod plt-gitlab-runners/runner-e1edahka-project-905-concurrent-0cxjqr to be
running, status is Pending
Waiting for pod plt-gitlab-runners/runner-e1edahka-project-905-concurrent-0cxjqr to be
running, status is Pending
       ContainersNotReady: "containers with unready status: [build helper]"
       ContainersNotReady: "containers with unready status: [build helper]"
Running on runner-e1edahka-project-905-concurrent-0cxjqr via pltf-shared-runner-6f85b8f
787-prdf9...
Getting source from Git repository
```

trigger_build_depl...

Duration: 1 minute 41 seconds
Finished: 23 seconds ago
Timeout: 1h (from project)
Runner: #646 (e1edAhKa) pltf-shared-runner-6f85b8f787-prdf9
Tags: toll6-runners

Commit 178b85e3
testingbranch as main branch .gitlab-ci.yml

Pipeline #61601 for testingbranch

→ trigger_build_deploy
trigger_build_deploy

Figure 10. 6 CI CD Job run successful, and container is running in OpenShift platform

11. Testing the Pipeline

11.1 Testing the functioning of Data Pipeline

Below are the steps to test if the Data Pipeline is functioning as expected. The netflow records are being streamed from on-premises Kafka Topic → Google PubSub→ Google Dataflow→ Google Big Query.

- **Step1:** Run the GitLab CI Job then once the job is successfully triggered and pipeline status is successful. The container will always be running on OpenShift platform.

- **Step 2:** Check the data on the topic in toll environment. The netflow streams should be available continuously in the Kafka topic (**pltf-develop-netflow-raw-fb**).

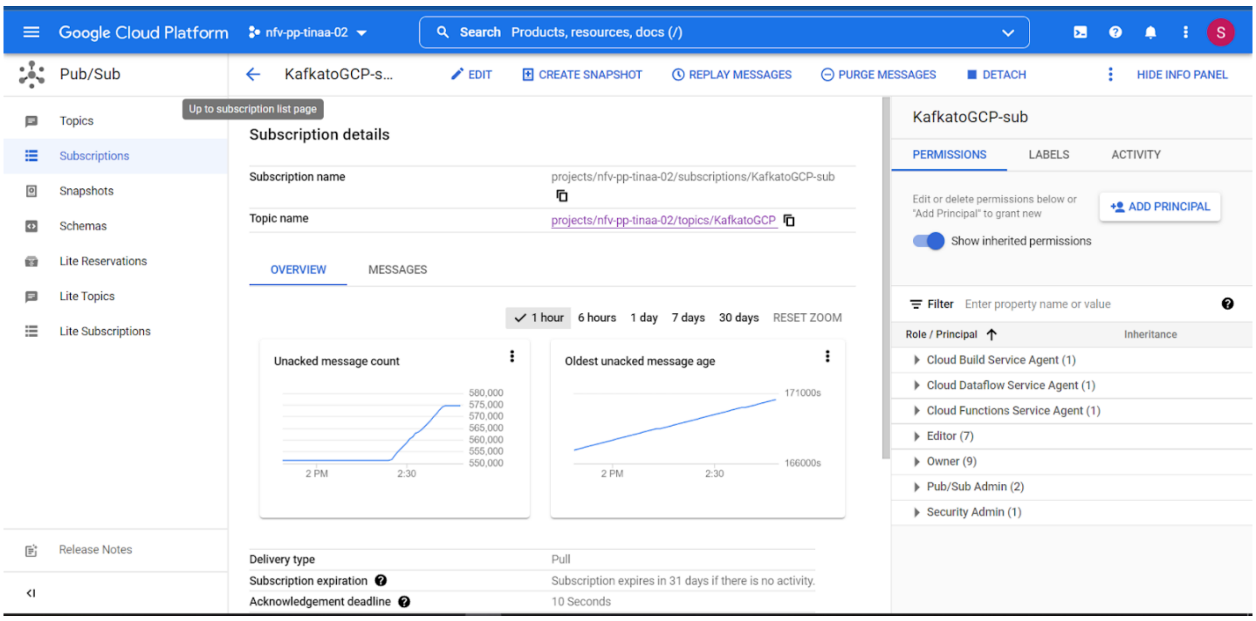
- **Step 3:** As the gcp-netflow-publisher will consume the Kafka records and publish them to GCP pubsub topic **(KafkatoGCP (projects/nfv-pp-tinaa-02/topics/KafkatoGCP)).**

  - **Kafka Topic:** pltf-develop-netflow-raw-fb
  - **PubSub Topic:**

    KafkatoGCP (projects/nfv-pp-tinaa-02/topics/KafkatoGCP)
  - **PubSub Subscription:**

    KafkatoGCP-sub (projects/nfv-pp-tinaa-02/subscriptions/KafkatoGCP-sub)

- **Step 4 Check Data ingested to PubSub Topic:** Navigate to the PubSub Topic to check if the netflow records are being published. Go to the overview to view the summary of Unacked message count and oldest message age.



Subscription details

Subscription name	projects/nfv-pp-tinaa-02/subscriptions/KafkatoGCP-sub
Topic name	projects/nfv-pp-tinaa-02/topics/KafkatoGCP
Delivery type	Pull
Subscription expiration	Subscription expires in 31 days if there is no activity.
Acknowledgement deadline	10 Seconds

KafkatoGCP-sub

Edit or delete permissions below or "Add Principal" to grant new

Show inherited permissions

- Cloud Build Service Agent (1)
- Cloud Dataflow Service Agent (1)
- Cloud Functions Service Agent (1)
- Editor (7)
- Owner (9)
- Pub/Sub Admin (2)
- Security Admin (1)

Figure 11. 1 Overview of messages published to PubSub

Click on the subscriptions tab →go to messages → Check if the messages are published →Click Pull to view messages and check the enable ACK messages to acknowledge

- **Step 5 Trigger Dataflow Job:** Go to cloud Dataflow→ Navigate to dataflow job (Pubsub subscription to BigQuery) → Check if the job is running successfully. If not trigger the job with below parameters.

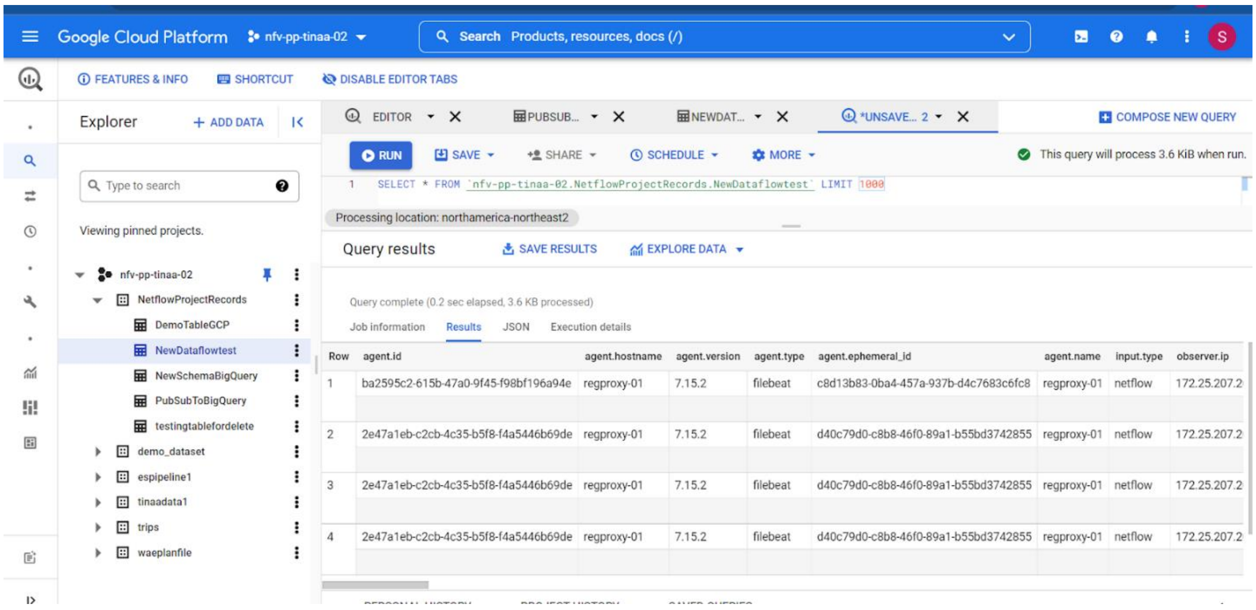
Data Flow Job Parameters:

DataFlow Job Parameters	Value
Job Name	Dataflow-PubSub-BigQuery
Regional Endpoint	northamerica-northeast1 (Montreal)
Dataflow template:	Pub/Sub Subscription to BigQuery
Pub/Sub Input Subscription	projects/nfv-pp-tinaa-02/subscriptions/KafkatoGCP1-sub
BigQuery Output table:	nfv-pp-tinaa-02:NetflowProjectRecords.NewDataflowtest
Temporary Location	gs://sbucket436/sarwarstorage
Worker IP Address Configuration	Private

Subnetwork	https://www.googleapis.com/compute/v1/projects/nfv-pp-tinaa-02/regions/northamerica-northeast1/subnetworks/datapipeline-nfv-pp-tinaa-02-subnet s

Table 11. 1 Dataflow Job Template values

- **Step 6 Datawarehouse:** Go to Cloud BigQuery→ Open the table → Query the results and check if the netflow records are populated in the table.



This query will process 3.6 KiB when run.

```
SELECT * FROM `nfv-pp-tinaa-02.NetflowProjectRecords.NewDataflowtest` LIMIT 1000
```

Processing location: northamerica-northeast2

Query complete (0.2 sec elapsed, 3.6 KB processed)

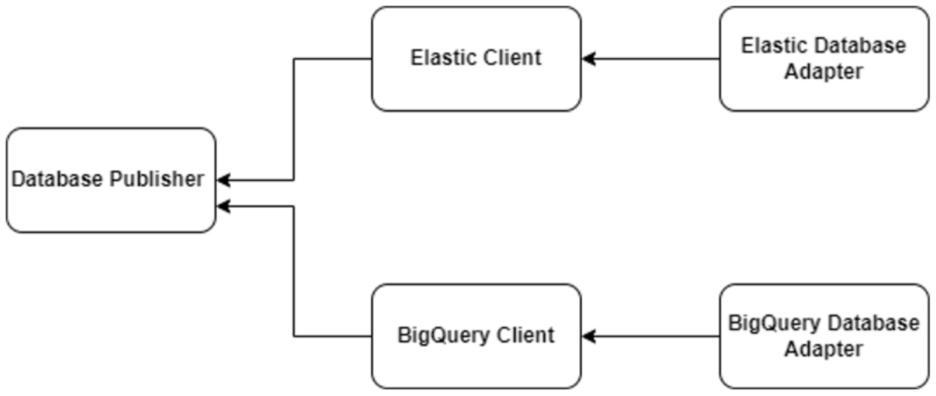
Row	agent.id	agent.hostname	agent.version	agent.type	agent.ephemeral_id	agent.name	input.type	observer.ip
1	ba2595c2-615b-47a0-9f45-f98bf196a94e	regproxy-01	7.15.2	filebeat	c8d13b83-0ba4-457a-937b-d4c7683c6fc8	regproxy-01	netflow	172.25.207.2
2	2e47a1eb-c2cb-4c35-b5f8-f4a5446b69de	regproxy-01	7.15.2	filebeat	d40c79d0-c8b8-46f0-89a1-b55bd3742855	regproxy-01	netflow	172.25.207.2
3	2e47a1eb-c2cb-4c35-b5f8-f4a5446b69de	regproxy-01	7.15.2	filebeat	d40c79d0-c8b8-46f0-89a1-b55bd3742855	regproxy-01	netflow	172.25.207.2
4	2e47a1eb-c2cb-4c35-b5f8-f4a5446b69de	regproxy-01	7.15.2	filebeat	d40c79d0-c8b8-46f0-89a1-b55bd3742855	regproxy-01	netflow	172.25.207.2

Check to see if the netflow records are published as expected. Search the results using the SQL query.

12. TINAA Datastore Adapter

12.1 TINAA Datastore

TINAA Datastore package abstracts APIs that are needed to connect to the various databases and follows the adaptor design pattern. Thus, various databases clients (e.g. Elasticsearch Python Client, Google BigQuery Client) can be used in the future by creating an Adaptor class without modifying the Client class. The Adaptor class facilitates the interaction between the Client class and the Adaptee. In the current version, only Elasticsearch database is supported. However, the code is extended to support BigQuery database also.



12.1.1 BigQuery Database Adapter

Therefore, to use TINAA Datastore package you need to instantiate the TINAAESAdaptor or TINAABigQueryAdapter class and pass it to the Client class.
So we have created a BigQuery adapter class with similar functions of Elastic. So that any function can be consumed in the database publisher (client). The end user will have a choice to select the database either BigQuery or elastic search.
Ex: Common functions to BigQuery and Elastic like Database Create Document or Search etc. can be called from any client (internally calls BigQuery or Elastic adapter) and based on user selection respective BigQuery or Elastic data will be called.

12.1.2 Python BigQuery Client:

Pre-requisites:

e) Service Account with BigQuery API enabled
f) Service Account JSON Key (with BigQuery admin role)
g) Installed Libraries: JSON, BigQuery
h) Import Libraries:

- pip install --upgrade google-cloud-BigQuery
- pip install json

Below is the code snippet for python BigQuery Client: This script will extract the data from BigQuery.

```python
import json
from ast import literal_eval
from google.cloud import BigQuery
import os

#service account key
key_path = "nfv-pp-tinaa-02-a09814f6d26e-BigQuery.json"
os.environ["GOOGLE_APPLICATION_CREDENTIALS"]=key_path
credentials = service_account.Credentials.from_service_account_file(
    key_path, scopes=["https://www.googleapis.com/auth/cloud-platform"],
)
#initialize client to authenticate and connect to BigQuery API
client = bigquery.Client(credentials=credentials,
project=credentials.project_id,)
BigQuery_client = BigQuery.Client()

#Perform Query from the BigQuery Table
name_group_query = """
    SELECT * FROM `nfv-pp-tinaa-02.NetflowProjectRecords.NewDataflowtest` LIMIT
1000

"""
query_results = BigQuery_client.query(name_group_query)

#display the results of the query
for result in query_results:
    print(str(result[0]))
```

References

[1] "DataPipeline," [Online]. Available: https://www.qlik.com/us/data-integration/data-pipeline.

[2] "Why is Data Pipeline Important/," [Online]. Available: https://towardsdatascience.com/scalable-efficient-big-data-analytics-machine-learning-pipeline-architecture-on-cloud-4d59efc092b5 .

[3] "Types of Data Pipelines," [Online]. Available: https://www.informatica.com/ca/resources/articles/data-pipeline.html#:~:text=Deploying%20the%20data%20pipeline%20will,manage%20complex%20multi%2Dcloud%20environments..

[4] "PMACCT Collector," [Online]. Available: http://www.pmacct.net/.

[5] "PmacctConfiguration," [Online]. Available: https://www.ntop.org/products/netflow/nprobe/.

[6] "NfacctdConfiguration," [Online]. Available: https://imply.io/blog/an-end-to-end-streaming-analytics-stack-for-network-telemetry-data .

[7] "PmacctNetflow," [Online]. Available: https://markelic.de/how-to-collect-flows/ .

[8] "Filebeat(Elastic)," [Online]. Available: https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html#:~:text=Filebeat%20is%20a%20lightweight%20shipper,Elasticsearch%20or%20Logstash%20for%20indexing .

[9] "Filebeat Module," [Online]. Available: https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-modules.html.

[10] "How Filebeat Works," [Online]. Available: https://www.elastic.co/guide/en/beats/filebeat/current/how-filebeat-works.html .

[11] "Configure Filebeat," [Online]. Available: https://logz.io/blog/filebeat-tutorial/ .

[12] "Data Streams -Netflow," [Online]. Available: https://medium.com/event-driven-utopia/understanding-kafka-topic-partitions- ae40f80552e8.

[13] "What is Kafka Consumer," [Online]. Available: https://towardsdatascience.com/kafka-python-explained-in-10-lines-of-code-800e3e07dad1.

[14] "Kafka Consumer," [Online]. Available: https://kafka-python.readthedocs.io/en/master/apidoc/KafkaConsumer.html.

[15] Kafka Consumer Python, [Online]. Available: https://kafka-python.readthedocs.io/en/master/.

[16] Python Kafka Consumer Parameters, [Online]. Available: https://kafka-python.readthedocs.io/en/master/_modules/kafka/consumer/group.html#KafkaConsumer .

[17] "Google Cloud Platform Services," [Online]. Available: https://www.edureka.co/blog/what-is-google-cloud-platform/ .

[18] "KafkaVsPubSub," [Online]. Available: https://stackoverflow.com/questions/38572071/i-am-evaluating-google-pub-sub-vs-kafka-what-are-the-differences#:~:text=In%20general%2C%20both%20are%20very,both%20Cloud%20and%20On%2Dprem. .

[19] "CloudDataFlow Vs Cloud DataProc," [Online]. Available: https://stackoverflow.com/questions/46436794/what-is-the-difference-between-google-cloud-dataflow-and-google-cloud-dataproc .

[20] "BigQuery Vs BigTable," [Online]. Available: https://stackoverflow.com/questions/39919815/whats-the-difference-between-bigquery-and-bigtable.

[21] "BigQuery Vs BigTable," [Online]. Available: https://hevodata.com/learn/bigtable-vs-bigquery/.

[22] PubSub, [Online]. Available: https://cloud.google.com/pubsub/docs/overview.

[23] "Identity and Access Management," [Online]. Available: https://cloud.google.com/iam/docs/creating-managing-service-account-keys.

[24] "Service Account," [Online]. Available: https://cloud.google.com/iam/docs/understanding-service-accounts.

[25] "Steps to Create Service Accont in GCP," [Online]. Available: https://cloud.google.com/iam/docs/creating-managing-service-accounts.

[26] "Service Account Key in GCP," [Online]. Available: https://cloud.google.com/iam/docs/creating-managing-service-account-keys.

[27] "Access Control with IAM," [Online]. Available: https://cloud.google.com/pubsub/docs/access-control.

[28] "Topics and Subscriptions Creation," [Online]. Available: https://cloud.google.com/pubsub/docs/create-topic-console.

[29] "Cloud DataFlow," [Online]. Available: https://medium.com/swlh/apache-beam-google-cloud-dataflow-and-creating-custom-templates-using-python-c666c151b4bc.

[30] "Cloud Dataflow," [Online]. Available: https://cloud.google.com/vpc/docs/vpc.

[31] "VPC Components," [Online]. Available: https://k21academy.com/google-cloud/google-cloud-vpc/#VPC_Components.

[32] "Steps to create VPC network," [Online]. Available: https://cloud.google.com/vpc/docs/vpc.

[33] "Dataflow template," [Online]. Available: https://cloud.google.com/dataflow/docs/guides/templates/provided-templates.

[34] "PubSub To BigQuery," [Online]. Available: https://cloud.google.com/dataflow/docs/guides/templates/provided-streaming#cloudpubsubsubscriptiontobigquery.

[35] "PubSub To BigQuery," [Online]. Available: https://cloud.google.com/dataflow/docs/guides/templates/provided-streaming#cloudpubsubsubscriptiontobigquery.

[36] "BigQuery dataset creation," [Online]. Available: https://cloud.google.com/bigquery/docs/datasets.

[37] "BigQuery Table Creation," [Online]. Available: https://cloud.google.com/bigquery/docs/tables#creating_an_empty_table_without_a_schema_definition.

[38] "Table in BigQuery Steps," [Online]. Available: https://cloud.google.com/bigquery/docs/tables.

[39] "BigQuery Schema," [Online]. Available: https://hevodata.com/learn/specifying-bigquery-schema/.

[40] "SSH Keys," [Online]. Available: https://www.youtube.com/watch?v=mNtQ55quG9M&t=390s.

[41] "GitPush Steps," [Online]. Available: https://www.youtube.com/watch?v=RBk1I-G2YA4.

[42] "What is Docker," [Online]. Available: https://www.infoworld.com/article/3204171/what-is-docker-the-spark-for-the-container-revolution.html.

[43] "Docker Images," [Online]. Available: https://docs.docker.com/get-started/overview/.

[44] "GitLabCIPipeline," [Online]. Available: https://docs.gitlab.com/ee/ci/.

[45] "EnvironmentVariables," [Online]. Available: https://k33g.gitlab.io/articles/2019-07-26-OPENSHIFT.html.