Available online www.jsaer.com

Journal of Scientific and Engineering Research, 2021, 8(9):242-248

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Unlocking Data Processing Efficiency: A Comprehensive Guide to GCP Dataflow

Bhargav Bachina

Abstract GCP Dataflow offers a unified stream and batch data processing solution that is both serverless and cost-effective. As a fully managed service, it boasts a range of features, detailed on its official website. Apache Beam, on the other hand, provides an advanced unified programming model for implementing batch and streaming data processing jobs across various execution engines, with GCP Dataflow serving as one of its runners. This paper explores the initiation of GCP Dataflow, beginning with a straightforward Spring Boot application and demonstrating integration with Apache Beam. The discussion covers running the application on a local machine using the Direct Runner and includes practical examples. Topics include prerequisites, Apache Beam setup, batch versus stream processing distinctions, creation of batch and stream processing jobs, configuration of custom pipeline options, local machine pipeline execution, testing procedures, an illustrative example project, and concluding insights.

Keywords Google Cloud Platform, Programming, Web Development, Cloud Computing, Software Development

GCP Dataflow is a Unified stream and batch data processing that's serverless, fast, and cost-effective. It is a fully managed data processing service and has many other features which you can find on its website here (https://cloud.google.com/dataflow). Apache Beam is an advanced unified programming model that implements

batch and streaming data processing jobs that run on any execution engine. GCP dataflow is one of the runners that you can choose from when you run data processing pipelines.
At this time of writing, you can implement it in languages Java, Python, and Go. If you need to process large datasets or data stream processing Apache beam is the tool that can process with a unified, portable, and extensible programming model. You can get a lot of flexibility and advanced functionality that you need for data processing jobs. There are so many runners you can choose from, for example, If you want to run the whole thing on GCP you have Google Dataflow that you use as a runner.
In this post, we will see how we can get started with GCP Dataflow. We will start with a simple Spring Boot application and see how to integrate with Apache Beam and run it on your local machine with Direct Runner and see some example projects.

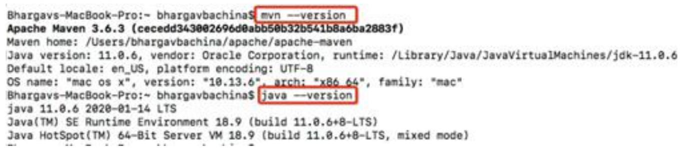
- Prerequisites
- How to get started with Apache Beam
- Batch vs Stream Processing Job
- Creating a Batch Processing Job
- Creating a Stream Processing Job
- Configure Custom Pipeline Options
- Run a Pipeline from Local Machine
- Testing Pipeline
- Example Project
- Conclusion

1. Prerequisites

There are some prerequisites for this project such as Apache Maven, Java SDK, and some IDE. You need to install all these on your machine if you want to run this example project on your machine.

- JDK Installation (https://www.oracle.com/java/technologies/javase-downloads.html)
- Apache Maven (https://maven.apache.org/download.cgi)
- IntelliJ IDEA Community Edition (https://www.jetbrains.com/idea/download/#section=mac)
- GCP Cloud SDK (https://cloud.google.com/sdk)
- Gsutil tool (https://cloud.google.com/storage/docs/gsutil)

Make sure you install Java and Maven on your machine by testing with these commands. You need to add these to your path so that you can run these commands.
java --versionmvn –version

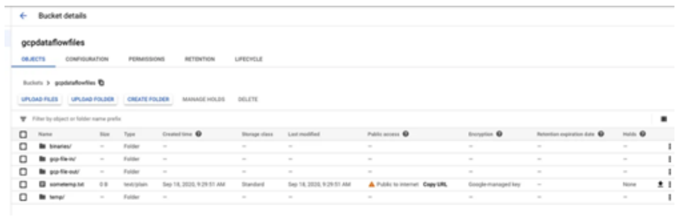


```
Bhargavs-MacBook-Pro:~ bhargavbachina$ mvn --version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: /Users/bhargavbachina/apache/apache-maven
Java version: 11.0.6, vendor: Oracle Corporation, runtime: /Library/Java/JavaVirtualMachines/jdk-11.0.6
Default locale: en_US, platform encoding: UTF-8
OS name: "mac os x", version: "10.13.6", arch: "x86_64", family: "mac"
Bhargavs-MacBook-Pro:~ bhargavbachina$ java --version
java 11.0.6 2020-01-14 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.6+8-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.6+8-LTS, mixed mode)
```

Figure 1: Checking Versions

A.GCP Prerequisites

- Create a new project.
- You need to create a Billing Account
- Link Billing Account With this project.
- Enable All the APIs that we need to run the dataflow on GCP.
- Download the Google SDK
- Create GCP Storage Buckets for source and sinks.



Bucket details

gcpdataflowfiles

OBJECTS CONFIGURATION PERMISSIONS RETENTION LIFECYCLE

Buckets > gcpdataflowfiles

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DELETE

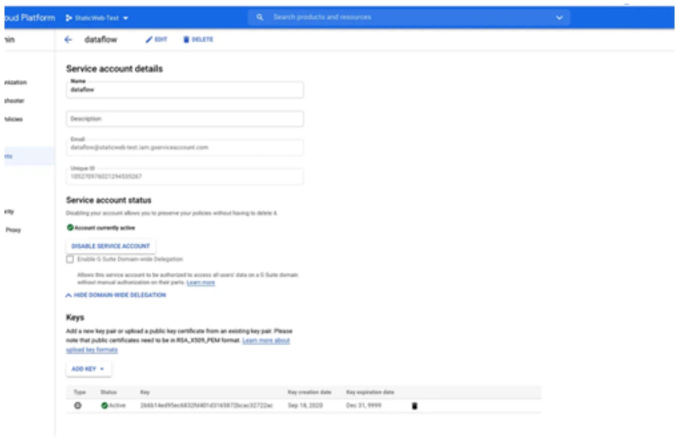
Filter by object or folder name prefix

Name	Size	Type	Created time	Storage class	Last modified	Public access	Encryption	Retention expiration date	Holds
binaries/	–	Folder	–	–	–	–	–	–	–
gcp-file-in/	–	Folder	–	–	–	–	–	–	–
gcp-file-out/	–	Folder	–	–	–	–	–	–	–
sometemp.txt	0 B	text/plain	Sep 18, 2020, 9:29:51 AM	Standard	Sep 18, 2020, 9:29:51 AM	Public to internet Copy URL	Google-managed key	–	None
temp/	–	Folder	–	–	–	–	–	–	–

Figure 2: GCP Storage Buckets

B. Service Account

Need to create a service account so that when you run the application from your local machine it can invoke the GCP dataflow pipeline with owner permissions.



dataflow EDIT DELETE

Service account details

Name: dataflow

Description

Email: dataflow@staticweb-test.iam.gserviceaccount.com

Unique ID: 105270976021294535267

Service account status

Disabling your account allows you to preserve your policies without having to delete it.

Account currently active

DISABLE SERVICE ACCOUNT

Enable G Suite Domain-wide Delegation

Allows this service account to be authorized to access all users' data on a G Suite domain without manual authorization on their parts. Learn more

HIDE DOMAIN-WIDE DELEGATION

Keys

Add a new key pair or upload a public key certificate from an existing key pair. Please note that public certificates need to be in RSA_X509_PEM format. Learn more about upload key formats

ADD KEY

Type	Status	Key	Key creation date	Key expiration date
	Active	266b14ed95ec6832fd401d31658 72bcac32722ac	Sep 18, 2020	Dec 31, 9999

Figure 3: Service Account

You have to generate the key and download and set the environment variable called **GOOGLE_APPLICATION_CREDENTIALS.**

export GOOGLE_APPLICATION_CREDENTIALS="/Users/bhargavbachina/gcp-credentials/gcp-dataflow-service-account.json"

Finally, you can run the following command to log in to your GCP account.

gcloud auth login

2. How to get started with Apache Beam

Apache Beam currently supports three SDKs Java, Python, and Go. All these SDKs provide a unified programming model that takes input from several sources. These sources can be finite data set from a batch data source or an infinite data set from a streaming data source.

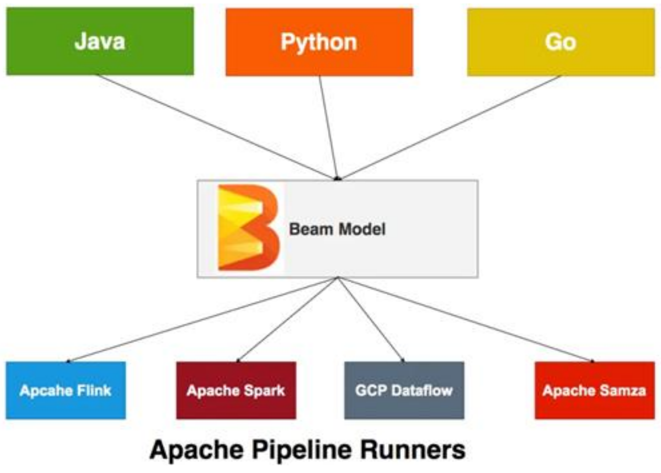


Figure 4: Apache Beam Programming Model

You define the pipeline for data processing, The Apache Beam pipeline Runners translate this pipeline with your Beam program into API compatible with the distributed processing back-end of your choice. It supports multiple Runners as shown in the above figure. You have to define an appropriate Runner when you run your Beam program so that the Runner executes your pipeline.
If you want to know how to get started with Apache Beam and be familiar with the Java example project, check out the below post.
How To Get Started with Apache Beam and Spring Boot (https://medium.com/bb-tutorials-and-thoughts/how-to-get-started-with-apache-beam-and-spring-boot-482654071a48)

3. Batch vs Stream Processing Job

There are two types of jobs in the GCP Dataflow one is a Streaming Job, and another is Batch Job. For example, you have one file, and you need to read it once a day and do all the processing. You can schedule a Batch Job that reads the entire file at once. But this is not the case all the time, what if you must read files as they come and do all the processing?
You need a streaming Job for that it constantly watches for the file in the GCS Bucket, and it reads the file as soon as it arrives in the bucket and does all the processing.

4. Creating a Batch Processing Job

If you have a finite amount of data and can be processed at once without interruption or interaction, the processing of this kind of data is termed Batch Processing. In the GCP Dataflow, you can read the entire file from the GCP Bucket and do processing or you can constantly listen to the buckets and keeps on reading files. Batch Processing applies to the first kind of work where you read the entire data set at once.
In the GCP Dataflow, you can use FileIO or TextIO to read the source.

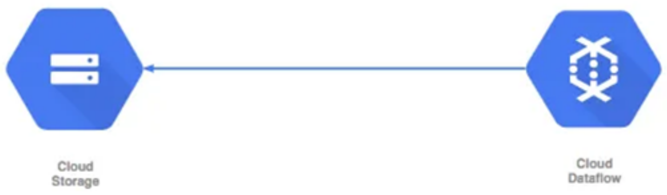


Figure 5: Dataflow Reads the file

If you look at the above diagram you can see that the pipelines in the Dataflow read the specific file patterns and read it completely. Below is the code snippet for both TextIO and FileIO I/O transforms.
https://gist.github.com/bbachi/37d562b75efbb1146b6301f8d144f3f4#file-pipeline-java
Here is a detailed example of how to create a Batch processing Job in GCP Dataflow.
How To Create a Batch Processing Job On GCP Dataflow (https://medium.com/bb-tutorials-and-thoughts/how-to-create-a-batch-processing-job-on-gcp-dataflow-bac3505422a7)

5. Creating a Stream Processing Job

In the GCP Dataflow, you can use FileIO or TextIO to continuously read the source for new files. You can use FileIO to continuously watch for specific file patterns and read those files.

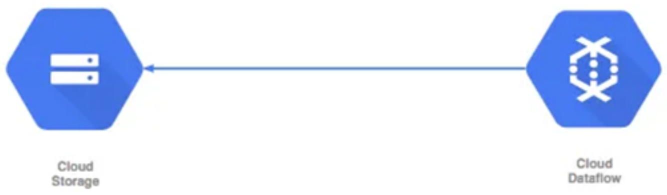


Figure 6: Dataflow Watches for file patterns

If you look at the above diagram you can see that the pipelines in the Dataflow constantly watch for the specific file patterns and read it. Below is the code snippet for both TextIO and FileIO I/O transforms.

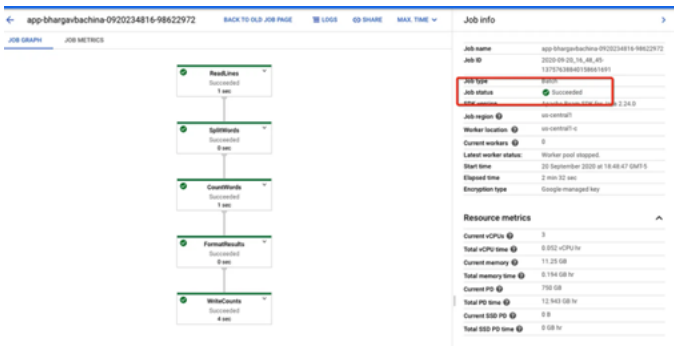
https://gist.github.com/bbachi/857b8ea27754b41131dd089974421620#file-pipeline-java
Here is a detailed example of how to create a Stream processing Job in GCP Dataflow.
How To Create a Stream Processing Job on GCP Dataflow (https://medium.com/bb-tutorials-and-thoughts/how-to-create-a-streaming-job-on-gcp-dataflow-a71b9a28e432)

6. Configure Custom Pipeline Options

We can configure default pipeline options and how we can create custom pipeline options so that we can pass them as command-line arguments when invoking the pipeline.
The options are limited here, for example, you can't set the input file name or output file name in the standard PipelineOptions
If you want to set these options as pipeline options, you can configure it. Let's see how we can do that. First, we need to define the interface that extends PipelineOptions like below.
https://gist.github.com/bbachi/fec69022a8c629c6e68712da65a25d53#file-runtimeoptions-java
Once you define the interface like above all you need to register this interface like below and you can set the options or use the default options.
// register the classPipelineOptionsFactory.*register*(RuntimeOptions.class);
RuntimeOptions options = PipelineOptionsFactory.*as*(RuntimeOptions.class);
It's recommended that you register your interface with PipelineOptionsFactory and then pass the interface when creating the PipelineOptions object. When you register your interface with PipelineOptionsFactory, the --help can find your custom options interface and add it to the output of the --help command. PipelineOptionsFactory will also validate that your custom options are compatible with all other registered options.
Here is a detailed example of how to configure custom pipeline options.
How To Configure Custom Pipeline Options In Apache Beam (https://medium.com/bb-tutorials-and-thoughts/how-to-configure-custom-pipeline-options-in-apache-beam-37a32f84d1aa)

7. Run a Pipeline from Local Machine

If you want to run the whole thing on your local machine the only thing you need to change is the input and output files and the type of runner that you want to run this pipeline on, in this case, it is Direct Runner. You can change these values in this example project, or you can check this below post as well.
How To Get Started With Apache Beam and Spring Boot (https://medium.com/bb-tutorials-and-thoughts/how-to-get-started-with-apache-beam-and-spring-boot-482654071a48)
Once you set up all the options and authorize the shell with GCP Authorization all you need to tun the fat jar that we produced with the command mvn package.
Make sure you set this environment variable like below and then run the remaining commands.
export GOOGLE_APPLICATION_CREDENTIALS="/Users/bhargavbachina/gcp-credentials/gcp-dataflow-service-account.json"
// mvn cleanmvn clean
// packagemvn package
// Run the applicationjava -jar gcp-pipeline-1.1-SNAPSHOT.jar
https://miro.medium.com/v2/resize:fit:720/0*Pk9AdmkUAHlI89pX.gif
Once you run the command java -jar gcp-pipeline-1.1-SNAPSHOT.jar, It invokes the pipeline on GCP.
https://miro.medium.com/v2/resize:fit:720/0*5yKuxD0Amv1IyqsD.gif
Once the pipeline is run, you can see the status message as succeeded


app-bhargavbachina-0920234816-98622972
BACK TO OLD JOB PAGE
LOGS
SHARE
MAX. TIME
JOB GRAPH
JOB METRICS
ReadLines Succeeded 1 sec
SplitWords Succeeded 0 sec
CountWords Succeeded 1 sec
FormatResults Succeeded 0 sec
WriteCounts Succeeded 4 sec
Job info
Resource metrics

Job name app-bhargavbachina-0920234816-98622972
Job ID 2020-09-20_16_48_45-13757638840158661691
Job type Batch
Job status Succeeded
SDK version Apache Beam SDK for Java 2.24.0
Job region us-central1
Worker location us-central1-c
Current workers 0
Latest worker status: Worker pool stopped.
Start time 20 September 2020 at 18:48:47 GMT-5
Elapsed time 2 min 32 sec
Encryption type Google-managed key

Current vCPUs 3
Total vCPU time 0.052 vCPU hr
Current memory 11.25 GB
Total memory time 0.194 GB hr
Current PD 750 GB
Total PD time 12.943 GB hr
Current SSD PD 0 B
Total SSD PD time 0 GB hr

Figure 7: Pipeline Successful

Here is a detailed article on how to run a pipeline from a local machine.
How To Run a GCP Dataflow Pipeline from Local Machine (https://medium.com/bb-tutorials-and-thoughts/how-to-run-a-gcp-dataflow-pipeline-from-local-machine-de4a6a4ff611)

8. Testing Pipeline

Testing the pipeline is as important as developing the pipeline. When you test your pipeline, you can completely test it on your local machine with the help of DirectRunner. It's faster in that way you don't have to test everything on remote runners such as GCP Dataflow, etc.
When you are testing a pipeline End-to-End you need to use classes such as TestPipeline and PAssert. Here are the suggestions from the official docs of Apache Programming Model

- For every source of input data to your pipeline, create some known static test input data.
- Create some static test output data that matches what you expect in your pipeline's final output PCollection(s).
- Create a TestPipeline in place of the standard Pipeline.create.
- In place of your pipeline's Read transform(s), use the Create transform to create one or more PCollections from your static input data.
- Apply your pipeline's transforms.
- In place of your pipeline's Write transform(s), use PAssert to verify that the contents of the final PCollections your pipeline produces match the expected values in your static output data.

Here is a detailed article on how to test the pipeline.
How To Test GCP Dataflow Pipeline (https://medium.com/bb-tutorials-and-thoughts/how-to-test-gcp-dataflow-pipeline-2e1b56862c41)

9. Example Project

- **How To Get Started with Apache Beam and Spring Boot** (https://medium.com/bb-tutorials-and-thoughts/how-to-get-started-with-apache-beam-and-spring-boot-482654071a48)
- **How To Create a Batch Processing Job On GCP Dataflow** (https://medium.com/bb-tutorials-and-thoughts/how-to-create-a-batch-processing-job-on-gcp-dataflow-bac3505422a7)
- **How To Create a Stream Processing Job On GCP Dataflow** (https://medium.com/bb-tutorials-and-thoughts/how-to-create-a-streaming-job-on-gcp-dataflow-a71b9a28e432)
- **How To Configure Custom Pipeline Options In Apache Beam** (https://medium.com/bb-tutorials-and-thoughts/how-to-configure-custom-pipeline-options-in-apache-beam-37a32f84d1aa)
- **How To Run a GCP Dataflow Pipeline from Local Machine** (https://medium.com/bb-tutorials-and-thoughts/how-to-run-a-gcp-dataflow-pipeline-from-local-machine-de4a6a4ff611)
- **How To Test GCP Dataflow Pipeline** (https://medium.com/bb-tutorials-and-thoughts/how-to-test-gcp-dataflow-pipeline-2e1b56862c41)
- **How To Read a PubSub Messages on GCP Dataflow** (https://medium.com/bb-tutorials-and-thoughts/how-to-read-a-pubsub-messages-on-gcp-dataflow-2658b23efa97)

10. Summary

This paper introduces GCP Dataflow, a serverless, efficient, and cost-effective data processing service, along with Apache Beam, a versatile programming model for batch and streaming data processing. It highlights the flexibility of Apache Beam and the various runners available, including Google Dataflow. The paper details the setup process, Apache Beam integration, and local execution using the Direct Runner. Key topics covered include Apache Beam setup, batch versus stream processing, job creation, custom pipeline configurations, local pipeline execution, testing procedures, and an example project, providing a comprehensive overview of GCP Dataflow initiation.

11.Conclusion

In conclusion, this paper has provided an overview of GCP Dataflow and Apache Beam, emphasizing their roles in streamlining data processing tasks with efficiency and cost-effectiveness. By leveraging GCP Dataflow's serverless architecture and Apache Beam's unified programming model, developers gain access to a powerful toolset for handling both batch and streaming data processing jobs. The availability of multiple language implementations and flexible execution options underscores the versatility of these technologies in addressing diverse data processing needs. Through practical examples and demonstrations, readers have been equipped with the knowledge necessary to begin utilizing GCP Dataflow and Apache Beam, laying the groundwork for effective data processing workflows. With a focus on prerequisites, job creation, configuration options, and testing procedures, this paper serves as a valuable resource for those embarking on data processing initiatives using these innovative tools.

References

[1]. Google Cloud Documentation https://cloud.google.com/docs
[2]. GCP Dataflow Documentation https://cloud.google.com/dataflow
[3]. Java Documentation https://docs.oracle.com/en/java/