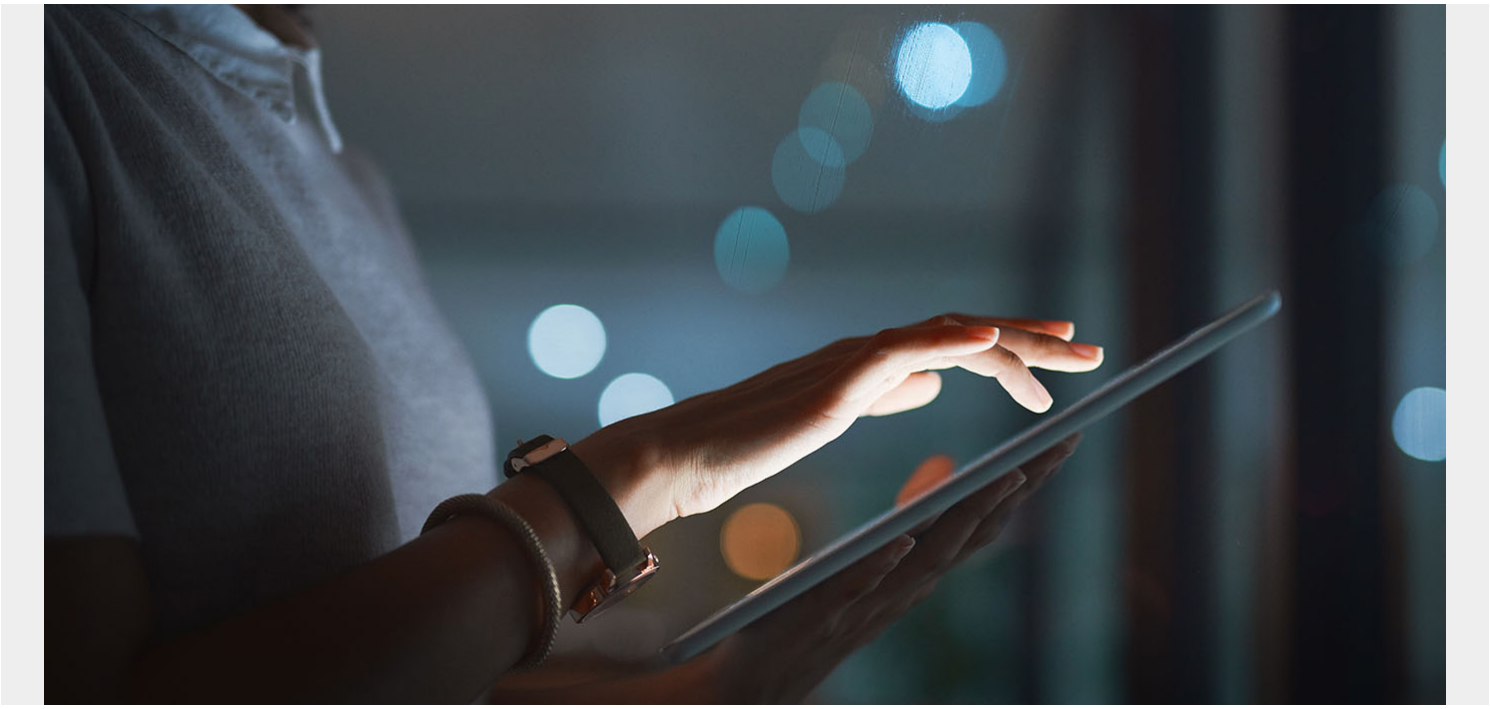
HOW TO ORCHESTRATE A DATA PIPELINE ON GOOGLE CLOUD WITH CONTROL-M FROM BMC



The Google Cloud Platform is designed specifically to accommodate organizations in a variety of positions along their cloud services journey, from large-scale, machine learning (ML) and data analysis to services tailored to SMB to hybrid-cloud solutions for customers that want to use services from more than one cloud provider. When BMC was migrating our Control-M application to this cloud ecosystem, we had to be very thoughtful about how we managed this change. The SADA engineering team worked alongside the BMC team to ensure that we had a seamless integration for our customers.

SADA supported this project by providing an inventory of the Google Cloud configuration options, decisions, and recommendations to enable the data platform foundation deployment, collaborated with BMC on the implementation planning, provided automation templates, and designed the Google Cloud architecture for the relevant managed services on the Google Cloud Platform.

In this article, we will discuss the end-result of this work, and look at an example using a credit-card fraud detection process to show how you can use Control-M to orchestrate a data pipeline seamlessly in Google Cloud.

FIVE ORCHESTRATION CHALLENGES

There are five primary challenges to consider when streamlining the orchestration of an ML data pipeline:

- **Understand the workflow.** Examine all dependencies and any decision trees. For example, if

data ingestion is successful, then proceed down this path; if it is not successful, proceed down that path.
- **Understand the teams.** If multiple teams are involved in the workflow, each needs to have a way to define their workflow using a standard interface, and to be able to merge their workflows to make up the pipeline.
- **Follow standards.** Teams should use repeatable standards and conventions when building workflows. This avoids having multiple jobs with identical names. Each step should also have a meaningful description to help clarify its purpose in the event of a failure.
- **Minimize the number of tools required.** Use a single tool for visualization and interaction with the pipeline (and dependencies). Visualization is important during the definition stage since it's hard to manage something that you can't see. This is even more important when the pipeline is running.
- **Include built-in error handling capabilities in the orchestration engine.** It's important to understand how errors can impact downstream jobs in the workflow or the business service level agreement (SLA). On the same note, failure of a job should not halt the pipeline altogether and involve human interaction. Criteria can be used to determine if a failed job can be restarted automatically or whether a human must be contacted to evaluate the failure, if, for instance, there are a certain number of failures involving the same error.

MEETING THE CHALLENGE

Meeting these orchestration challenges required a solid foundation and also presented opportunities for collaboration. BMC and SADA aligned using the SADA POWER line of services to establish the data platform foundation. Some notable elements in this technical alignment included work by SADA to:

- Apply industry expertise to expedite BMC's development efforts.
- Establish a best practices baseline around data pipelines and the tools to orchestrate them.
- Conduct collaborative sessions in order to understand BMC's technical needs and provide solutions that the BMC team could integrate and then expand upon.

SADA's Data Platform Foundation provided opportunities to leverage Google Cloud services to accomplish the complex analytics required of an application like Control-M. The BMC and SADA teams worked together to establish a strong foundation for a robust and resilient solution through:

- Selecting data and storage locations in Google Cloud Storage.
- Utilizing the advantages provided by Pub/Sub to streamline the analytics and data integration pipelines.
- Having thorough discussions around the extract, transform, and load (ETL) processes to truly understand the end state of the data.
- Using BigQuery and writing analytic queries.
- Understanding the importance of automation, replicability of processes, and monitoring performance in establishing a system that is scalable and flexible.
- Using Data Studio to create a visualization dashboard to provide the necessary business insights.

REAL-WORLD EXAMPLE

Digital transactions have been increasing steadily for many years, but that trend is now coupled with a permanent decline in the use of cash as people and businesses practice physical distancing. The adoption of digital payments for businesses and consumers has consequently grown at a much higher rate than previously anticipated, leading to increased fraud and operational risks.

With fraudsters improving their techniques, companies are relying on ML to build resilient and efficient fraud detection systems.

Since fraud constantly evolves, detection systems must be able to identify new types of fraud by detecting anomalies that are seen for the first time. Therefore, detecting fraud is a perpetual task that requires constant diligence and innovation.

Common types of financial fraud that customers work to prevent with this application include:

- **Stolen/fake credit card fraud:** Transactions made using fake cards, or cards belonging to someone else.
- **ATM fraud:** Cash withdrawals using someone else's card.

Fraud detection is composed of both real-time and batch processes. The real-time process is responsible for denying a transaction and possibly placing a hold on an account or credit card, thus preventing the fraud from occurring. It must respond quickly, sometimes at the cost of reduced accuracy.

To minimize false positives, which may upset or inconvenience customers, a batch phase is used to continuously fine-tune the detection model. After transactions are confirmed as valid or fraudulent, all recent events are input to the batch process on a regular cadence. This batch process then updates the training and scoring of the real-time model to keep real-time detection operating at peak accuracy. This batch process is the focus of this article.

USE OUR DEMO SYSTEM

SADA and BMC created a demonstration version of our solution so you can experiment with it on Google Cloud. You can find all of our code, plus sample data, in GitHub.

Resources included are:

- Kaggle datasets of transaction data, fraud status, and demographics
- Queries
- Schema
- User-defined functions (UDFs)

HOW IT WORKS

For each region in which the organization operates, transaction data is collected daily. Details collected include (but are not limited to):

- **Transaction details.** Describes each transaction, including the amount, item code, location, method of payment, and so on.
- **Personal details.** Describes the name, address, age, and other details about the purchaser.

This information is pulled from corporate data based on credit card information and real-time fraud detection that identifies which transactions were flagged as fraudulent.

New data arrives either as batch feeds or is dropped into Cloud Storage by Pub/Sub. This new data is then loaded into BigQuery by Dataflow jobs. Normalization and some data enrichment is performed by UDFs during the load process.

Once all the data preparation is complete, analytics are run against the combined new and historical data to test and rank fraud detection performance. The results are displayed in Data Studio dashboards.

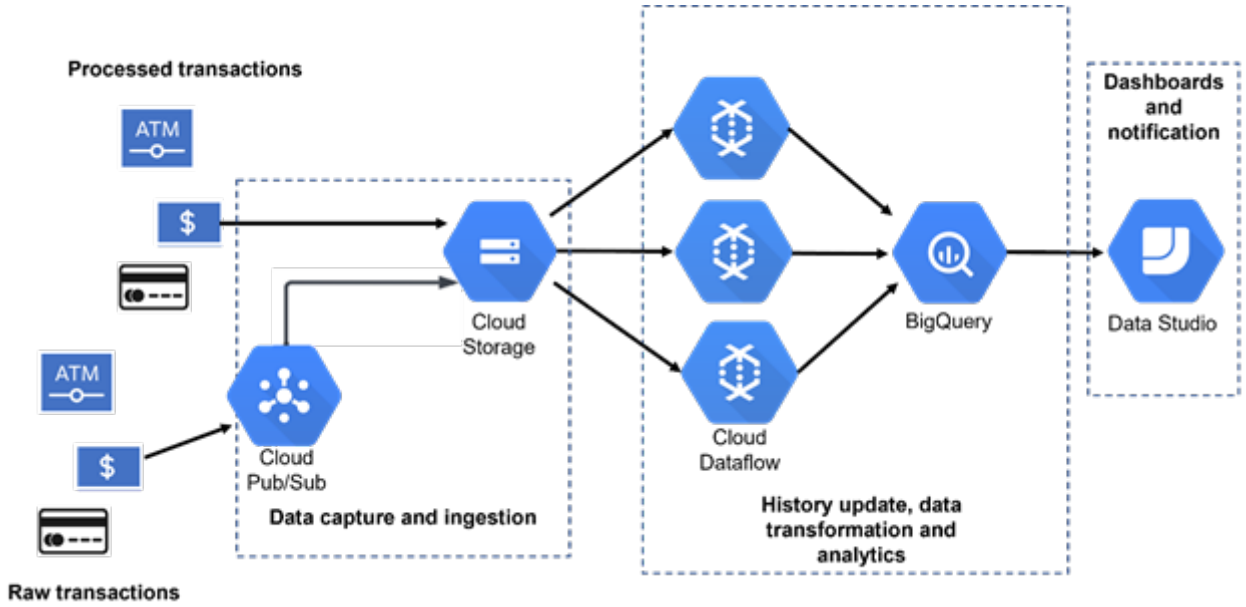


Figure 1: Control-M orchestration

GOOGLE CLOUD SERVICES IN THE PIPELINE

Cloud storage provides a common landing zone for all incoming data and a consistent input for downstream processing. Dataflow is Google Cloud's primary data integration tool.

SADA and BMC selected Big Query for data processing. Earlier versions of this application used Hadoop, but while working with the team at SADA, we converted to BigQuery as this is the recommended strategy from Google for sophisticated Data Warehouse or Data Lake applications. This choice also simplified setup by providing out-of-the-box integration with Cloud Dataflow. UDFs provide a simple mechanism for manipulating data during the load process.

TWO WAYS TO DEFINE PIPELINE WORKFLOWS

You can use Control-M to define your workflow in two ways:

- **Using a graphical editor.** This provides the option of dragging and dropping the workflow steps into a workspace and connecting them.
- **Use RESTful APIs.** Define the workflows using a jobs-as-code method, then use JSON to integrate into a continuous integration/continuous delivery (CI/CD) toolchain. This method improves workflow management by flowing jobs through a pipeline of automated building, testing, and release. Google Cloud provides a number of developer tools for CI/CD, including

Cloud Build and Cloud Deploy.

DEFINING JOBS IN THE PIPELINE

The basic Control-M execution unit is referred to as a *job*. There are a number of attributes for every job, defined in JSON:

- **Job type.** Options include script, command, file transfer, Dataflow, or BigQuery.
- **Run location.** For instance, which host is running the job?
- **Identity.** For example, is the job being "run as…" or run using a connection profile?
- **Schedule.** Determines when to run the job and identifies relevant scheduling criteria.
- **Dependencies.** This could be things like whether the job must finish by a certain time or output must arrive by a certain time or date.

Jobs are stored in folders and the attributes discussed above, along with any other instructions, are applied to all jobs in that folder.

We can see in the code sample below an example of the JSON code that describes the workflow used in the fraud detection model ranking application. You can find the full JSON code in the Control-M Automation API Community Solutions GitHub repo. While there, you can also find some solutions, the Control-M Automation API guide, and other code samples in the same repository.

```
{
"Defaults" : {
},
"jog-mc-gcp-fraud-detection": {"Type": "Folder",
"Comment" : "Update fraud history, run, train and score models",
"jog-gcs-download" : {"Type" : "Job:FileTransfer",…},
"jog-dflow-gcs-to-bq-fraud": {"Type": "Job:Google DataFlow",…},
"jog-dflow-gcs-to-bq-transactions": {"Type": "Job:Google DataFlow",…},
"jog-dflow-gcs-to-bq-personal": {"Type": "Job:Google DataFlow",…},
"jog-mc-bq-query": {"Type": "Job:Database:EmbeddedQuery", …},
"jog-mc-fm-service": {"Type": "Job:SLAManagement",…},
},
"flow00": {"Type":"Flow", "Sequence":},
"flow01": {"Type":"Flow", "Sequence":},
"flow02": {"Type":"Flow", "Sequence":}

}
}
```

The jobs shown in this workflow correspond directly with the steps illustrated previously in Figure 1.

The workflow contains three fundamental sections:

- **Defaults.** These are the functions that apply to the workflow. This could include details such as who to contact for job failures or standards for job naming or structure.

```
{  "Defaults" : {"RunAs" : "ctmagent", "OrderMethod": "Manual", "Application"
:
```

```
       "multicloud", "SubApplication" : "jog-mc-fraud-modeling",
      "Job" : {"SemQR": { "Type": "Resource:Semaphore", Quantity": "1"},
      "actionIfError" : {"Type": "If", "CompletionStatus":"NOTOK",
"mailTeam":
          {"Type": "Mail", "Message": "Job %%JOBNAME failed", "Subject":
                "Error occurred", "To": deng_support@bmc.com}}}
    },
```

- **Job definitions.** This is where individual jobs are specified and listed. See below for descriptions of each job in the flow.
- **Flow statements.** These define the relationships of the job, both upstream and downstream.

```
"flow00": {"Type":"Flow", "Sequence":},
"flow01": {"Type":"Flow", "Sequence":},
"flow02": {"Type":"Flow", "Sequence":}
```

SCHEDULING PIPELINE WORKFLOWS

Control-M uses a server-and-agent model. The server is the central engine that manages workflow scheduling and submission to agents, which are lightweight workers. In the demo described in this article, the Control-M server and agent are both running on Google Compute Engine VM instances.

Workflows are most-commonly launched in response to various events such as data arrival but may also be executed automatically based on a predefined schedule. Schedules are very flexible and can refer to business calendars; specify different days of the week, month, or quarter; define cyclic execution, which runs workflows intermittently or every “n” hours or minutes; and so on.

PROCESSING THE DATA

File Transfer job type

Looking at the first job (Figure 2), called jog-gcs-download, we can see that this job, of the type Job:FileTransfer, transfers files from a conventional file system described by ConnectionProfileSrc to Google Cloud Storage described by ConnectionProfileDest.

The File Transfer job type can watch for data-related events (file watching) as a prerequisite for data transfer, as well as perform pre/post actions such as deletion of the source after a successful transfer, renaming, source and destination comparison, and restart from the point of failure in the event of an interruption. In the example, this job moves several files from a Linux® host and drops them into Google Cloud Storage buckets.

```
"jog-gcs-download" : {"Type" : "Job:FileTransfer",
        "Host" : "ftpagents",
        "ConnectionProfileSrc" : "smprodMFT",
        "ConnectionProfileDest" : "joggcp",
        "S3BucketName" : "prj1968-bmc-data-platform-foundation",
        "Description" : "First data ingest that triggers downstream
```

```
applications",
        "FileTransfers" :
      },
```

Dataflow

Dataflow jobs are executed to push the newly arrived data into BigQuery. The jobs appear complex, but Google Cloud provides an easy-to-use process to make the definitions simple.

Go to the Dataflow Jobs page (Figure 2). If you have an existing job, choose to **Clone it** or **Create Job from Template**. Once you've provided the desired parameters, click on **Equivalent REST** at the bottom to get this information (Figure 3), which you can cut and paste directly into the job's Parameters section.

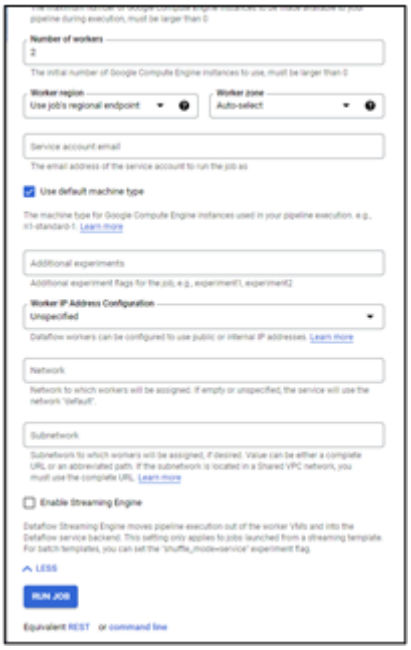


Figure 2: Dataflow Jobs page

Figure 3: Cut and paste into job Parameters

```
"jog-dflow-gcs-to-bq-fraud": {"Type": "Job:ApplicationIntegrator:AI Google
DataFlow",
        "AI-Location": "us-central1",
        "AI-Parameters (JSON Format)": "{\"jobName\": \"jog-dflow-gcs-to-bq-
fraud\",
        \"environment\": {        \"bypassTempDirValidation\": false,
        \"tempLocation\": \"gs://prj1968-bmc-data-platform-
foundation/bmc_fraud_details/temp\",
        \"ipConfiguration\": \"WORKER_IP_UNSPECIFIED\",
        \"additionalExperiments\": []    },
        \"parameters\": {
        \"javascriptTextTransformGcsPath\": \"gs://prj1968-bmc-data-platform-
foundation/bmc_fraud_details/bmc_fraud_details_transform.js\",
        \"JSONPath\": \"gs://prj1968-bmc-data-platform-
foundation/bmc_fraud_details/bmc_fraud_details_schema.json\",
        \"javascriptTextTransformFunctionName\": \"transform\",
        \"outputTable\": \"sso-gcp-dba-ctm4-pub-
cc10274:bmc_dataplatform_foundation.bmc_fraud_details_V2\",
        \"inputFilePattern\": \"gs://prj1968-bmc-data-platform-
foundation/bmc_fraud_details/bmc_fraud_details.csv\",
        \"bigQueryLoadingTemporaryDirectory\": \"gs://prj1968-bmc-data-
platform-foundation/bmc_fraud_details/tmpbq\"    }}",
        "AI-Log Level": "INFO",
        "AI-Template Location (gs://)": "gs://dataflow-templates-us-
central1/latest/GCS_Text_to_BigQuery",
        "AI-Project ID": "sso-gcp-dba-ctm4-pub-cc10274",
        "AI-Template Type": "Classic Template",
        "ConnectionProfile": "JOG-DFLOW-MIDENTITY",
        "Host": "gcpagents"
      },
```

SLA management

This job defines the SLA completion criteria and instructs Control-M to monitor the entire workflow as a single business entity.

```
"jog-mc-fm-service": {"Type": "Job:SLAManagement",
         "ServiceName": "Model testing and scoring for fraud detection",
         "ServicePriority": "3",
         "JobRunsDeviationsTolerance": "3",
         "CompleteIn": {
            "Time": "20:00"
          }
        },
```

The ServiceName specifies a business-relevant name that will appear in notifications or service incidents, as well as in displays for non-technical users, to make it clear which business service may be impacted. Important to note, Control-M uses statistics collected from previous executions to automatically compute the expected completion so that any deviation can be detected and reported at the earliest possible moment. This gives monitoring teams the maximum opportunity to course-correct before any impact to business services is detected.

EXAMINING THE STATE OF THE PIPELINE

Now that you have an idea of how jobs are defined, let's take a look at what the pipeline looks like when it's running.

Control-M provides a user interface for monitoring workflows (Figure 4). In the screenshot below, the first job completed successfully and is green, the next three jobs are executing and depicted in yellow. Jobs that are waiting to run are shown in gray.

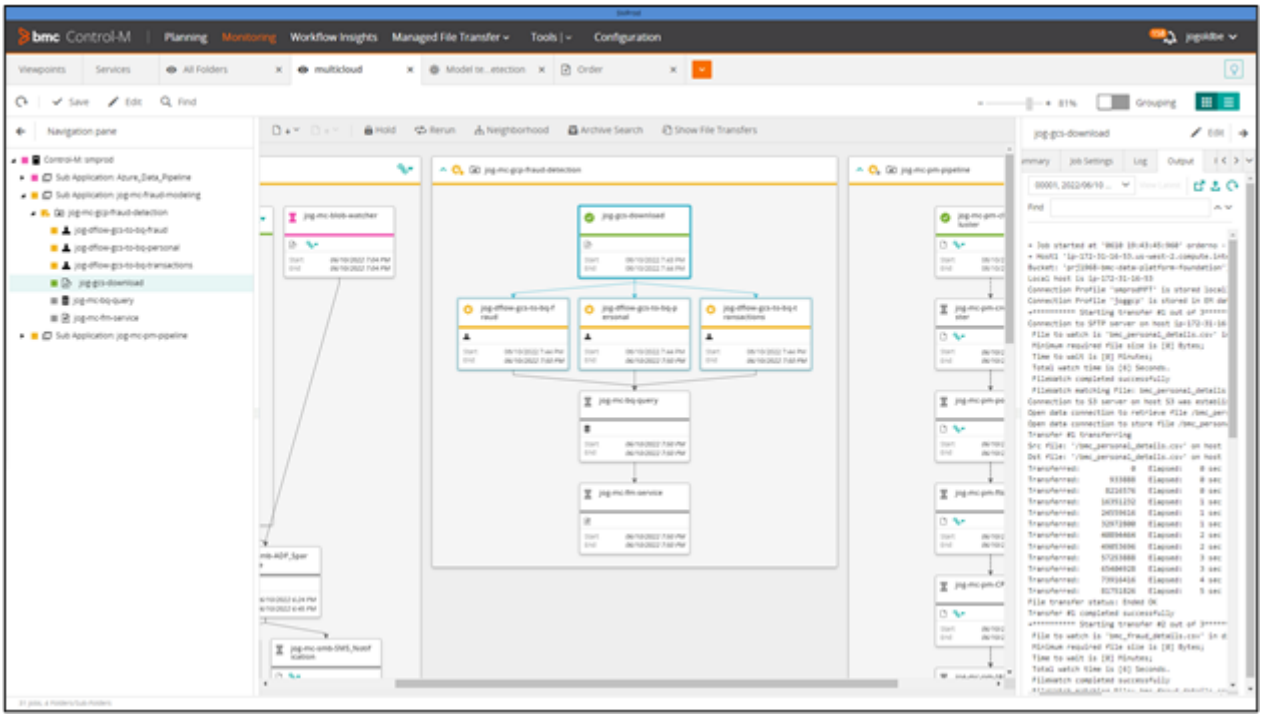


Figure 4: Control-M Monitoring Domain

You can access the output and logs of every job from the pane on the right-hand side. This capability is vital during daily operations. To monitor those operations more easily, Control-M provides a single pane to view the output of jobs running on disparate systems without having to connect to each application's console.

Control-M also allows you to perform several actions on the jobs in the pipeline, such as hold, rerun, and kill. You sometimes need to perform these actions when troubleshooting a failure or skipping a job, for example.

All of the functions discussed here are also available from a REST-based API or a CLI.

CONCLUSION

In spite of the rich set of ML tools that Google Cloud provides, coordinating and monitoring workflows across an ML pipeline remains a complex task.

Anytime you need to orchestrate a business process that combines file transfers, applications, data sources, or infrastructure, Control-M can simplify your workflow orchestration. It integrates, automates, and orchestrates application workflows whether on-premises, on the Google Cloud, or in a hybrid environment.