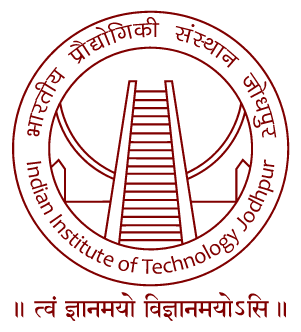
**INDIAN INSTITUTE OF TECHNOLOGY**

**JODHPUR**



**MINI PROJECT REPORT**

**ON**

**“STUDENT GRADE ANALYSIS & PREDICTION USING BIG DATA PIPELINE”**

**Prepared by**

Vivek Kumar Gupta

(PGD22DC026)

&

Arjun Vaibhav Srivastava

(PGD22DC005)

**STUDENT GRADE ANALYSIS & PREDICTION USING BIG DATA PIPELINE**

**1. Problem Statement**

The problem statement can be defined as follows ”Given a dataset containing attribute of set of students where using the features available from dataset and define classification algorithms to identify whether the student performs good in final grade exam, also to evaluate different machine learning models on the dataset.”

**2. Description of the Dataset**

This data approach student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school-related features) and it was collected by using school reports and questionnaires. Two datasets are provided regarding the performance in two distinct subjects: Mathematics (mat) and Portuguese language.

## 2.1 Attribute Information:

| Columns | Data type | Description |
| --- | --- | --- |
| school | Boolean | student's school (binary: 'GP' Gabriel Pereira or 'MS' Mousinho da Silveira) |
| sex | Boolean | student's sex (binary: 'F' female or 'M' male) |
| age | Int | student's age (numeric: from 15 to 22) |
| address | Boolean | student's home address type (binary: 'U' urban or 'R' rural) |
| famsize | Int | family size (binary: 'LE3' less or equal to 3 or 'GT3' greater than 3) |
| Pstatus | Boolean | parent's cohabitation status (binary: 'T' living together or 'A' apart) |
| Medu | Varchar (30) | mother's education (numeric: 0 none, 1 primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education) |
| Fedu | Varchar (30) | father's education (numeric: 0 none, 1 primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education) |
| Mjob | Varchar (30) | mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at\_home' or 'other') |
| Fjob | Varchar (30) | father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at\_home' or 'other') |
| reason | Varchar (30) | reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other') |
| guardian | Varchar (30) | student's guardian (nominal: 'mother', 'father' or 'other') |
| traveltime | Int | home to school travel time (numeric: 1 <15 min., 2 15 to 30 min., 3 30 min. to 1 hour, or 4 >1 hour) |
| studytime | Int | weekly study time (numeric: 1 <2 hours, 2 2 to 5 hours, 3 5 to 10 hours, or 4 >10 hours) |
| failures | Int | number of past class failures (numeric: n if 1<=n<3, else 4) |
| schoolsup | Boolean | extra educational support (binary: yes or no) |
| famsup | Boolean | family educational support (binary: yes or no) |
| paid | Boolean | extra paid classes within the course subject (Math or Portuguese) (binary: yes or no) |
| activities | Boolean | extracurricular activities (binary: yes or no) |
| nursery | Boolean | attended nursery school (binary: yes or no) |
| higher | Boolean | wants to take higher education (binary: yes or no) |
| internet | Boolean | Internet access at home (binary: yes or no) |
| romantic | Boolean | with a romantic relationship (binary: yes or no) |
| famrel | Int | quality of family relationships (numeric: from 1 very bad to 5 excellent) |
| freetime | Int | free time after school (numeric: from 1 very low to 5 very high) |
| goout | Int | going out with friends (numeric: from 1 very low to 5 very high) |
| Dalc | Int | workday alcohol consumption (numeric: from 1 very low to 5 very high) |
| Walc | Int | weekend alcohol consumption (numeric: from 1 very low to 5 very high) |
| health | Int | current health status (numeric: from 1 very bad to 5 very good) |
| absences | Int | number of school absences (numeric: from 0 to 93) |
| G1 | Int | score |
| G2 | Int | score |
| G3 | Int | socre |

**3. Methodology**

Since universities are prestigious places of higher education, students’ retention in these universities is a matter of high concern. It has been found that most of the students’ drop-out from the universities during their first year is due to lack of proper support in undergraduate courses. Due to this reason, the first year of the undergraduate student is referred as a “make or break” year. Without getting any support on the course domain and its complexity, it may demotivate a student and can be the cause to withdraw the course.

There is a great need to develop an appropriate solution to assist students retention at higher education institutions. Early grade prediction is one of the solutions that have a tendency to monitor students’ progress in the degree courses at the University and will lead to improving the students’ learning process based on predicted grades.

Using machine learning with Educational Data Mining can improve the learning process of students. Different models can be developed to predict students’ grades in the enrolled courses, which provide valuable information to facilitate students’ retention in those courses. This information can be used to early identify students at-risk based on which a system can 1 suggest the instructors to provide special attention to those students. This information can also help in predicting the students’ grades in different courses to monitor their performance in a better way that can enhance the students’ retention rate of the universities.

Using various packages such as cufflinks, seaborn & matplotlib to represent the data along with different attributes graphically or pictorially to analyze the dataset for predicting the Final Grade(G3).

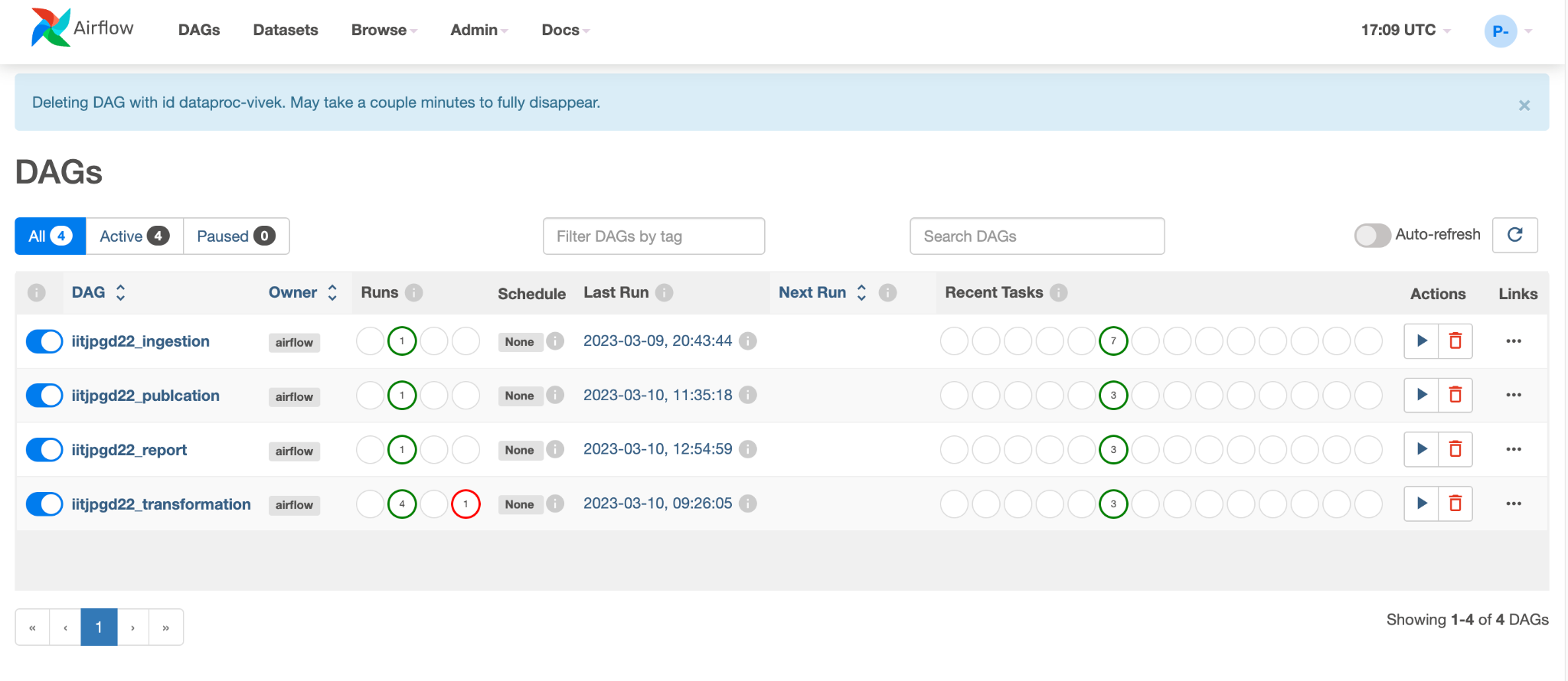
**4. Pipeline Design:**

Airflow DAGs:

Airflow DAGs (Directed Acyclic Graphs) are a collection of tasks that are organized in a way that reflects their relationships and dependencies. In Airflow, each DAG is defined as a Python script that instantiates a DAG object, and each task within the DAG is defined as an operator.Airflow DAGs are particularly useful for scheduling and orchestrating complex workflows, where different tasks need to be executed in a specific order, and some tasks may depend on the output of other tasks. By defining a DAG in Airflow, you can automate the execution of the different tasks, set their dependencies and schedule when they should be run.

**some key concepts related to Airflow DAGs:**

1. DAG: The overarching structure that encapsulates all the tasks and their relationships.
2. Operator: The individual task to be performed within a DAG. Each operator represents a unique action to be taken, such as running a script or sending an email.
3. Task: An instance of an operator within a DAG. Tasks are defined by the operator and have a specific set of parameters that dictate how the task will be executed.
4. Dependency: A relationship between two tasks in a DAG where one task must be completed before another can start. This is defined using the set\_upstream and set\_downstream methods on the operators.
5. Schedule: The timing of when a DAG or task should be run. This can be specified in the DAG definition as a cron expression or using Airflow's built-in scheduling options.
6. Airflow DAGs provide a powerful framework for automating complex workflows, allowing for easier management of dependencies and scheduling, and providing a clear overview of the status of each task in the DAG.

****

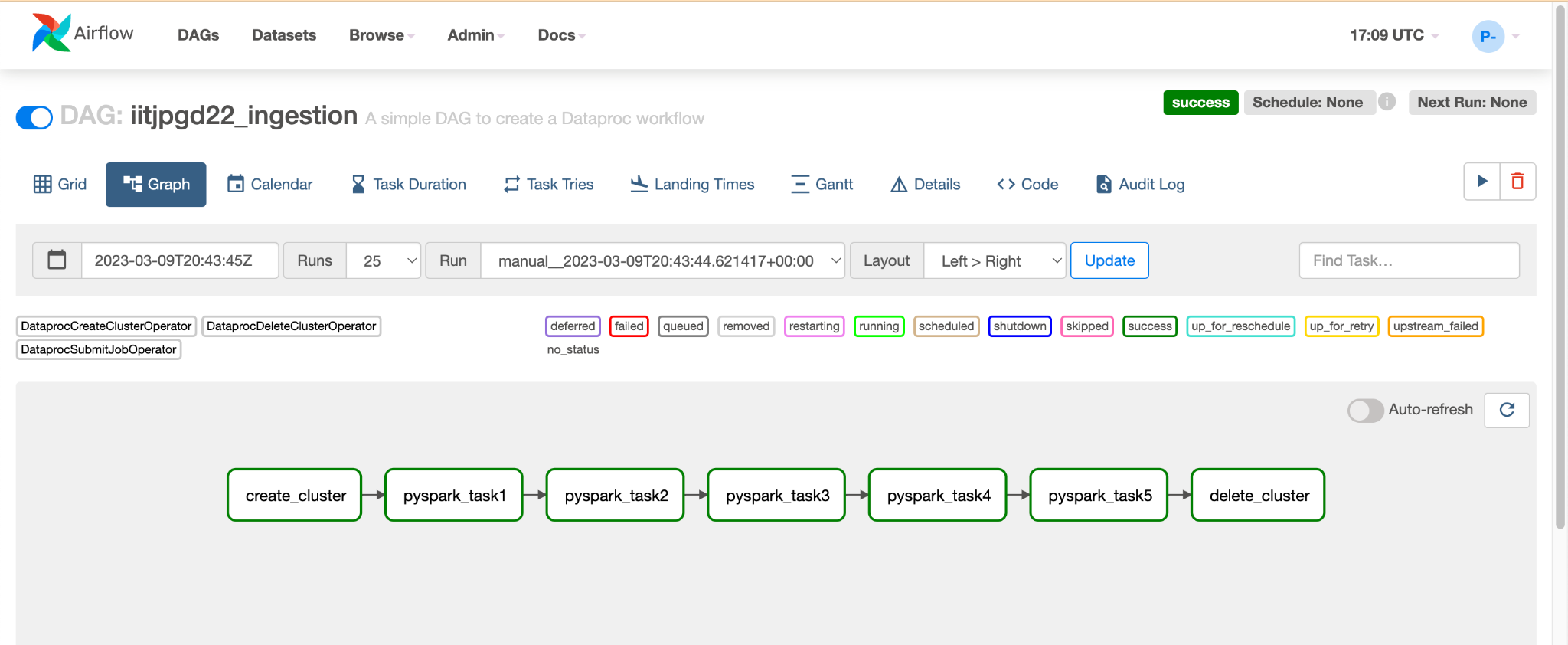
**4.1: Data Ingestion Layer:**

The data ingestion layer is the component of a data architecture that is responsible for collecting, extracting, and loading data from various sources into a centralized storage or data warehouse. It is the first step in the data pipeline and is critical to ensuring the accuracy and reliability of the downstream analysis.

The data ingestion layer typically consists of the following components:

1. Data Sources: These are the various sources from which data is collected. This can include databases, files, APIs, streaming data sources, and more.
2. Data Ingestion Tools: These are the tools and technologies used to extract and load data from the various sources into the data warehouse or centralized storage. Some common tools include Apache NiFi, Apache Kafka, Talend, Apache Flume, and AWS Glue.
3. Data Integration Framework: This is the framework that connects the data sources to the data ingestion tools and the centralized storage. It defines the data flow, transformation rules, and other business logic required to transform and integrate the data from various sources.
4. Data Quality and Governance: This is the set of practices and processes used to ensure the quality and integrity of the data being ingested. It includes data profiling, data validation, data cleansing, and data enrichment.

The data ingestion layer plays a critical role in ensuring the accuracy and reliability of the data used for analysis and reporting. By automating the collection and loading of data from various sources, it reduces the risk of errors and inconsistencies in the data. It also provides a centralized and consistent view of the data, making it easier to analyze and report on.

****

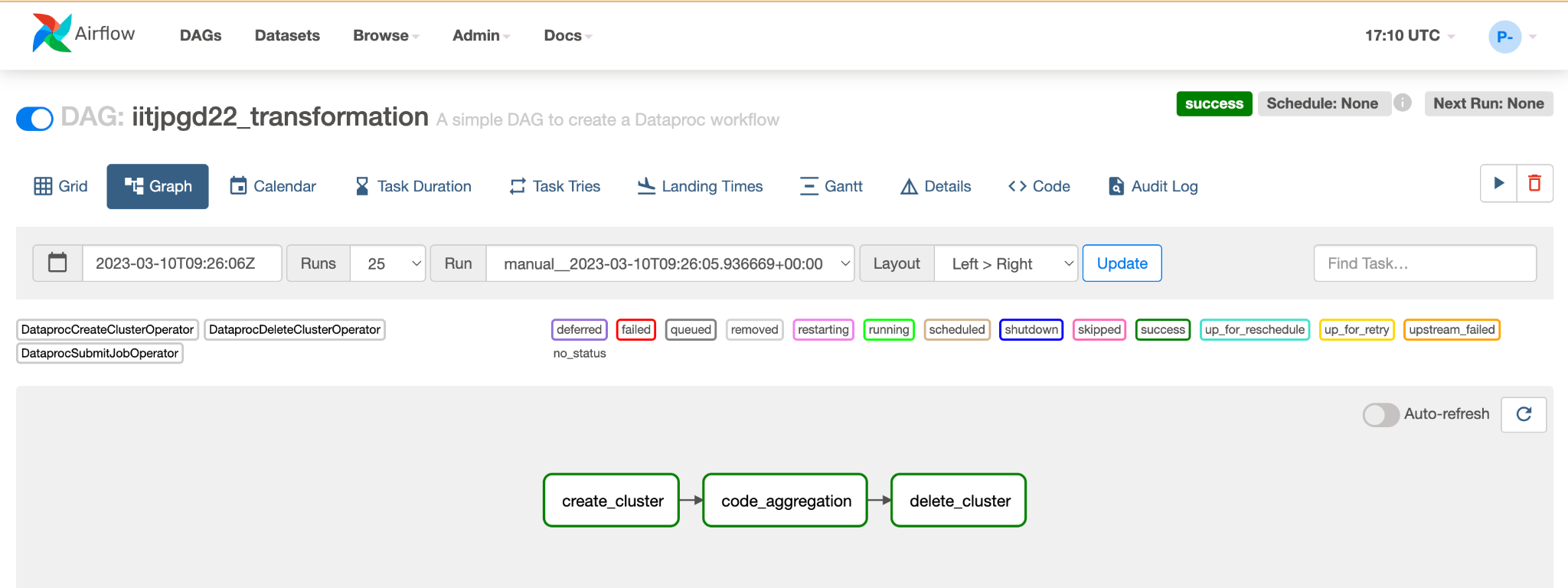
**4.2: Data Transformation Layer:**

The data transformation layer is the component of a data architecture that is responsible for transforming raw data into a format that is suitable for analysis and reporting. It is a critical step in the data pipeline, as it ensures that the data is accurate, complete, and consistent across different sources.

The data transformation layer typically consists of the following components:

1. Data Processing Tools: These are the tools and technologies used to process and transform the data. Some common tools include Apache Spark, Apache Flink, Apache Beam, and AWS Glue.
2. Data Integration Framework: This is the framework that connects the data processing tools to the data warehouse or centralized storage. It defines the data flow, transformation rules, and other business logic required to transform and integrate the data from various sources.
3. Data Quality and Governance: This is the set of practices and processes used to ensure the quality and integrity of the data being transformed. It includes data profiling, data validation, data cleansing, and data enrichment.
4. Data Analysis and Reporting: This is the final step in the data pipeline, where the transformed data is analyzed and reported on. This can include data visualization tools, BI tools, or custom applications.

The data transformation layer plays a critical role in ensuring that the data is accurate, complete, and consistent across different sources. By transforming the data into a format that is suitable for analysis and reporting, it makes it easier for data analysts and other stakeholders to make informed decisions based on the data. Additionally, it provides a consistent view of the data across different sources, reducing the risk of errors and inconsistencies in the data.

****

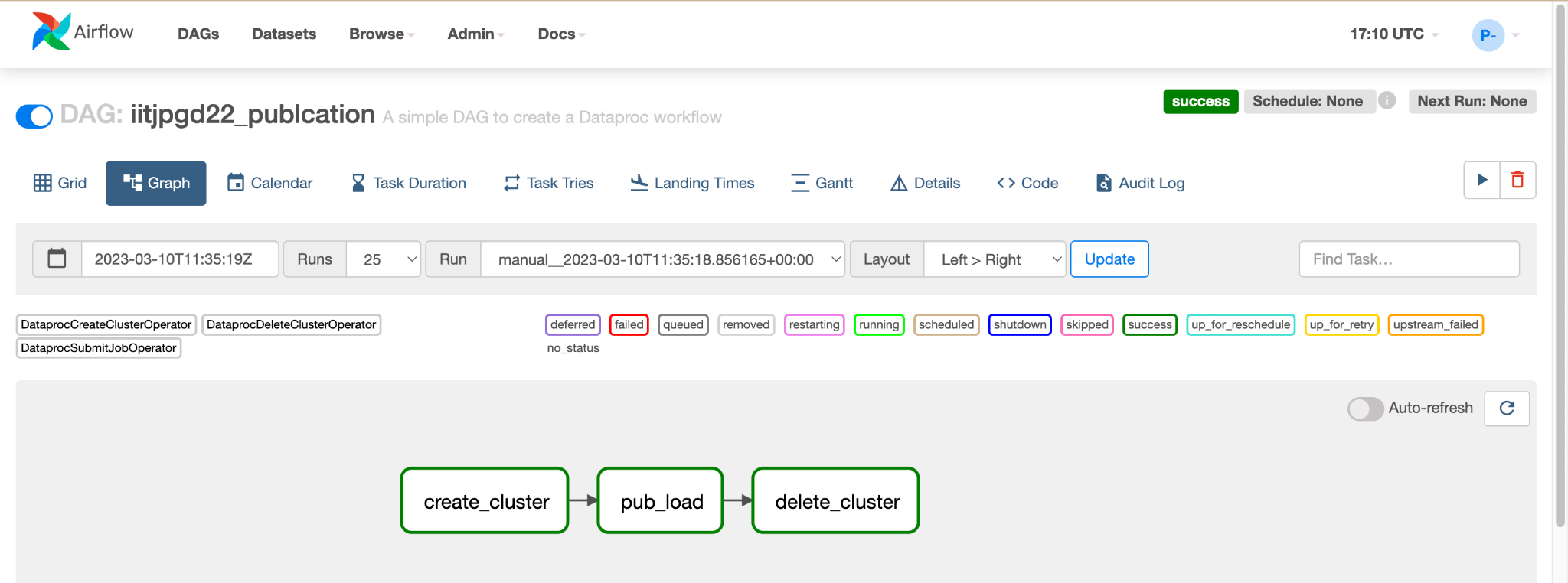
**4.3: Data Publication Layer:**

The data publication layer is the component of a data architecture that is responsible for making data available to end-users or other downstream systems. It is the final step in the data pipeline and is critical to enabling data-driven decision making.

The data publication layer typically consists of the following components:

1. Data Access APIs: These are the APIs that provide access to the data for end-users or downstream systems. They define the endpoints, data formats, and access rules for the data.
2. Data Visualization Tools: These are the tools and technologies used to visualize and present the data to end-users. Some common tools include Tableau, Power BI, and Looker.
3. Data Distribution Framework: This is the framework that connects the data access APIs to the data visualization tools or downstream systems. It defines the data flow, access rules, and other business logic required to distribute the data to the intended recipients.
4. Data Security and Compliance: This is the set of practices and processes used to ensure the security and compliance of the data being published. It includes data encryption, access controls, and compliance with regulatory requirements such as GDPR and HIPAA.

The data publication layer plays a critical role in enabling data-driven decision making by making data accessible and actionable for end-users and downstream systems. By providing access to the data through APIs and visualization tools, it makes it easier for users to interact with the data and gain insights from it. Additionally, by ensuring the security and compliance of the data, it reduces the risk of unauthorized access or data breaches.

****

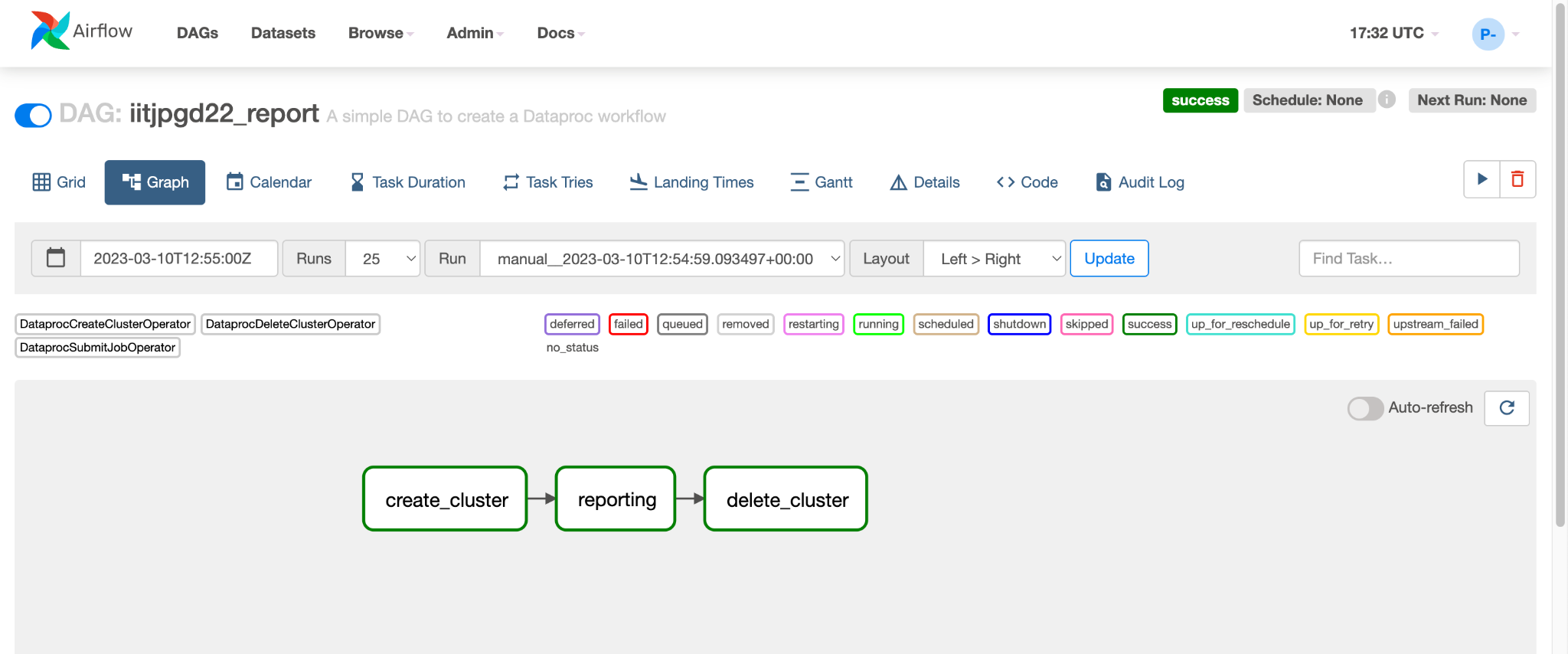
**4.4: Reporting Layer**

The reporting layer is the component of a data architecture that is responsible for creating and distributing reports based on the data stored in the data warehouse or centralized storage. It is a critical component of any data-driven organization, as it enables stakeholders to make informed decisions based on the insights gleaned from the data.

The reporting layer typically consists of the following components:

1. Reporting Tools: These are the tools and technologies used to create reports based on the data. Some common tools include Microsoft Excel, Power BI, Tableau, and SAP Crystal Reports.
2. Data Access APIs: These are the APIs that provide access to the data for the reporting tools. They define the endpoints, data formats, and access rules for the data.
3. Data Distribution Framework: This is the framework that connects the reporting tools to the data warehouse or centralized storage. It defines the data flow, access rules, and other business logic required to distribute the data to the reporting tools.
4. Report Generation and Distribution: This is the final step in the reporting process, where reports are generated and distributed to stakeholders. This can include automated report generation and distribution, as well as manual report generation and distribution.

The reporting layer plays a critical role in enabling stakeholders to make informed decisions based on the insights gleaned from the data. By providing access to the data through APIs and reporting tools, it makes it easier for users to create and distribute reports based on the data. Additionally, by automating the report generation and distribution process, it reduces the workload on IT teams and enables stakeholders to access the data they need in a timely manner.

****

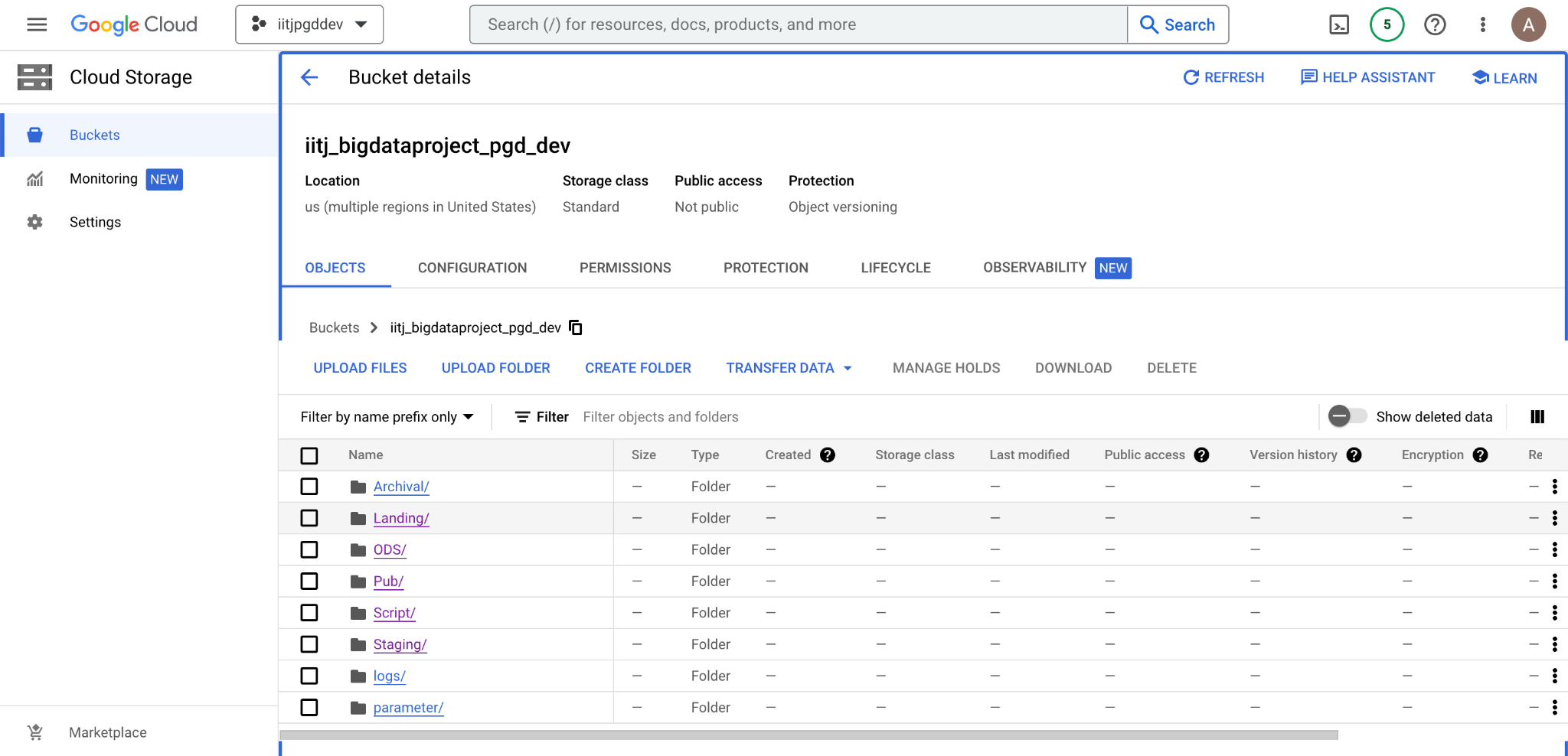
1. **GCP cloud storage:**

Google Cloud Storage (GCS), folders are not actual objects, but rather a logical concept that helps organize and structure data stored in buckets. A GCS bucket is a top-level container for storing data and can contain multiple folders, subfolders, and objects.

Here are some key points to consider when working with folders in GCS:

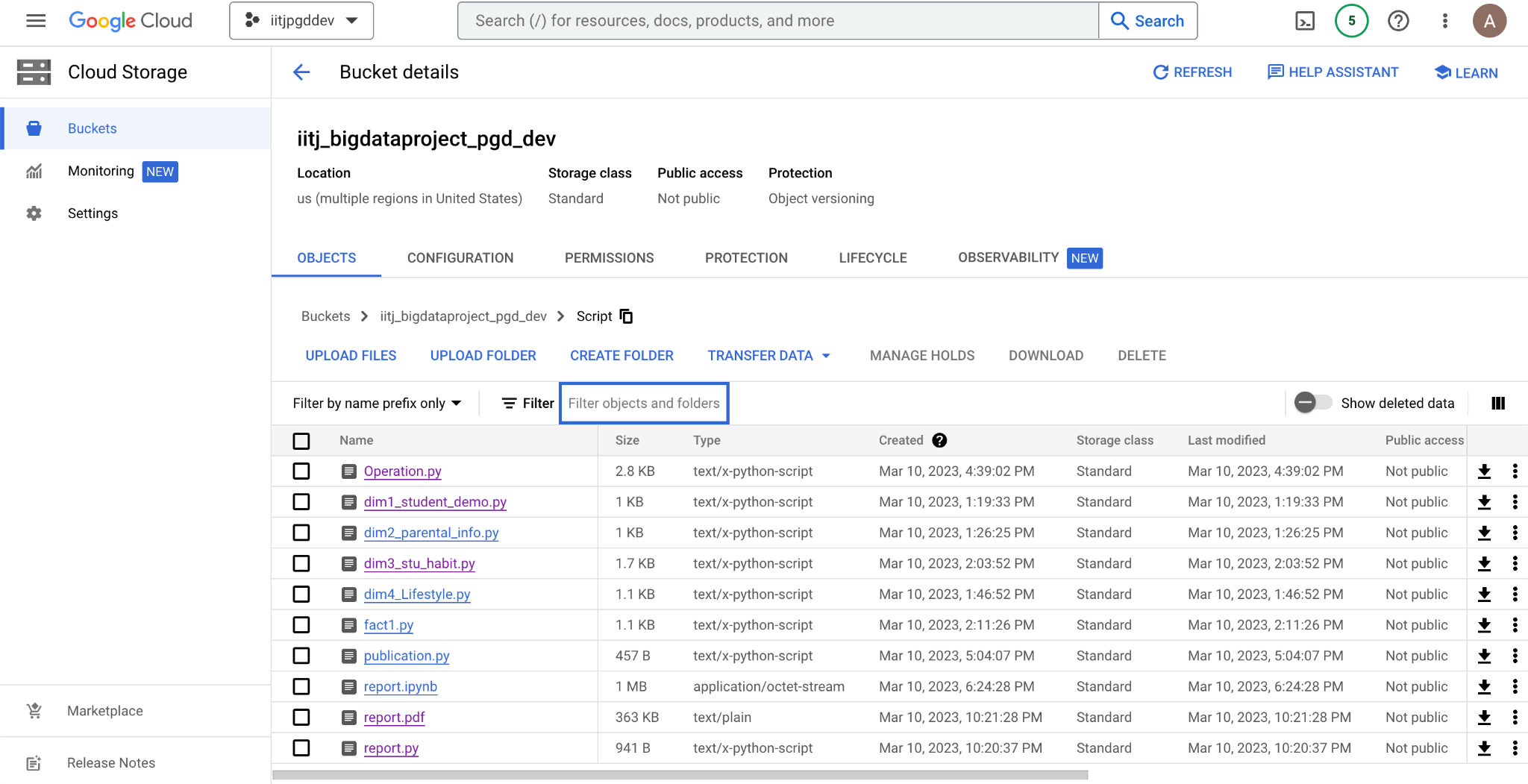
1. Folder names are part of the object names: When creating a folder in GCS, it is actually creating a zero-byte object with a specific prefix in its name. For example, if you create a folder named "my-folder", it will create an object named "my-folder/" (note the trailing slash).
2. Folders can have subfolders: You can create subfolders within a folder by specifying a prefix in the object name. For example, you can create a subfolder within "my-folder" called "sub-folder" by creating an object named "my-folder/sub-folder/".
3. Folders can be nested: You can create nested folders by creating subfolders within subfolders.
4. Folders do not have any special properties: From a storage perspective, folders are treated like any other object in GCS. They do not have any special properties or metadata associated with them.
5. Folders can be used to control access: Access control lists (ACLs) can be applied to folders to control who can access and manage the objects within them.

When working with folders in GCS, it is important to remember that they are just a logical concept for organizing and structuring data. While they can be useful for organizing large datasets, they do not provide any performance benefits over using a flat structure with objects in the bucket.

****

1. **Script path:**

A script path refers to the location or path of a script file on a computer or server. The path can be either an absolute path, which specifies the full path from the root directory of the file system, or a relative path, which specifies the path relative to the current working directory.

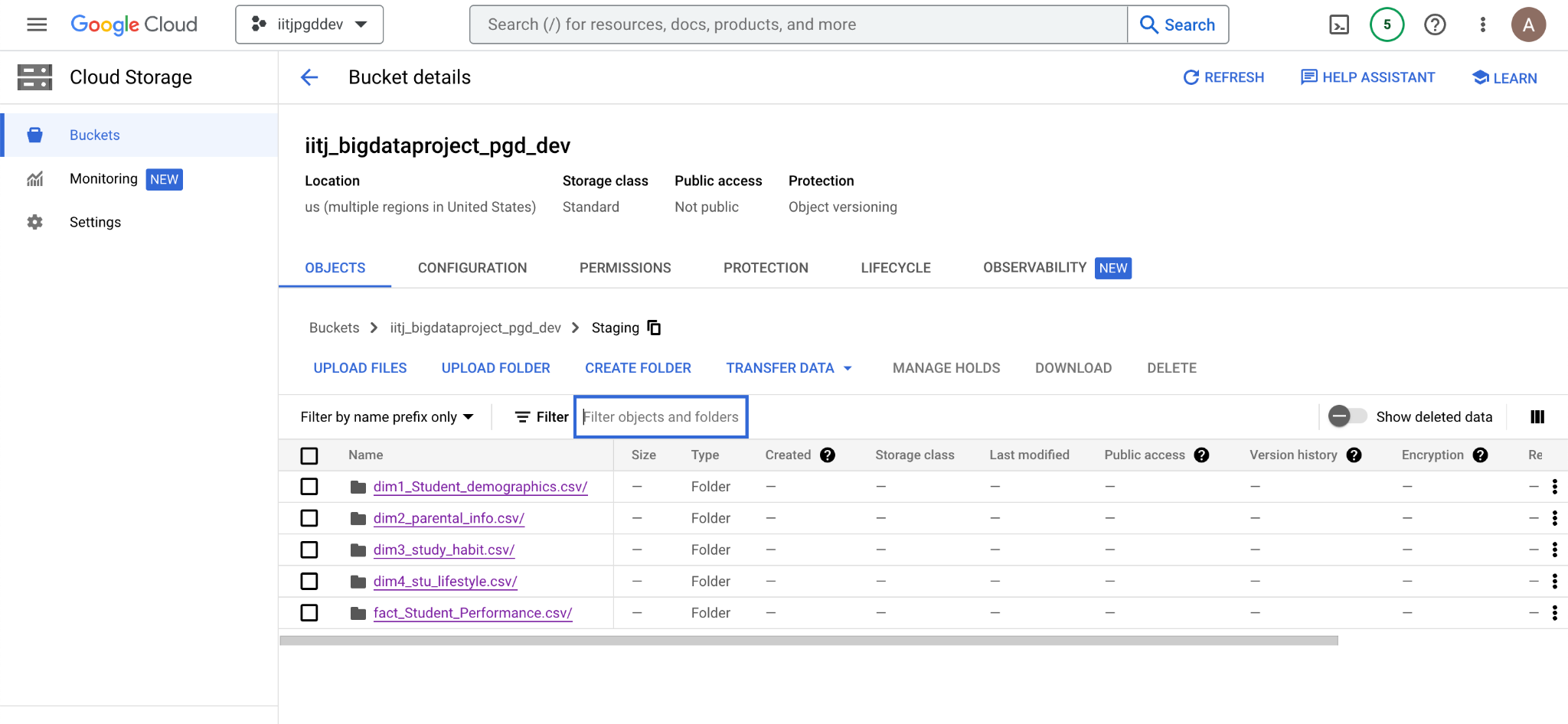
****

1. **Staging path:**

staging path is a location or directory where raw or unprocessed data is temporarily stored before being transformed and loaded into a target system or data warehouse.

Typically, the staging path is used to store data that has been extracted from the source system or data source, but has not yet been processed or transformed in any way. This allows for a separation of concerns between the data extraction, transformation, and loading processes, and provides a buffer or checkpoint for data to be validated and cleaned before it is loaded into the target system.

The staging path can be a folder or directory on a local file system or network file share, or it can be a location in a cloud-based data storage service such as Amazon S3, Google Cloud Storage, or Microsoft Azure Blob Storage. The exact location and structure of the staging path will depend on the specific requirements of the ETL process and the technology stack being used.

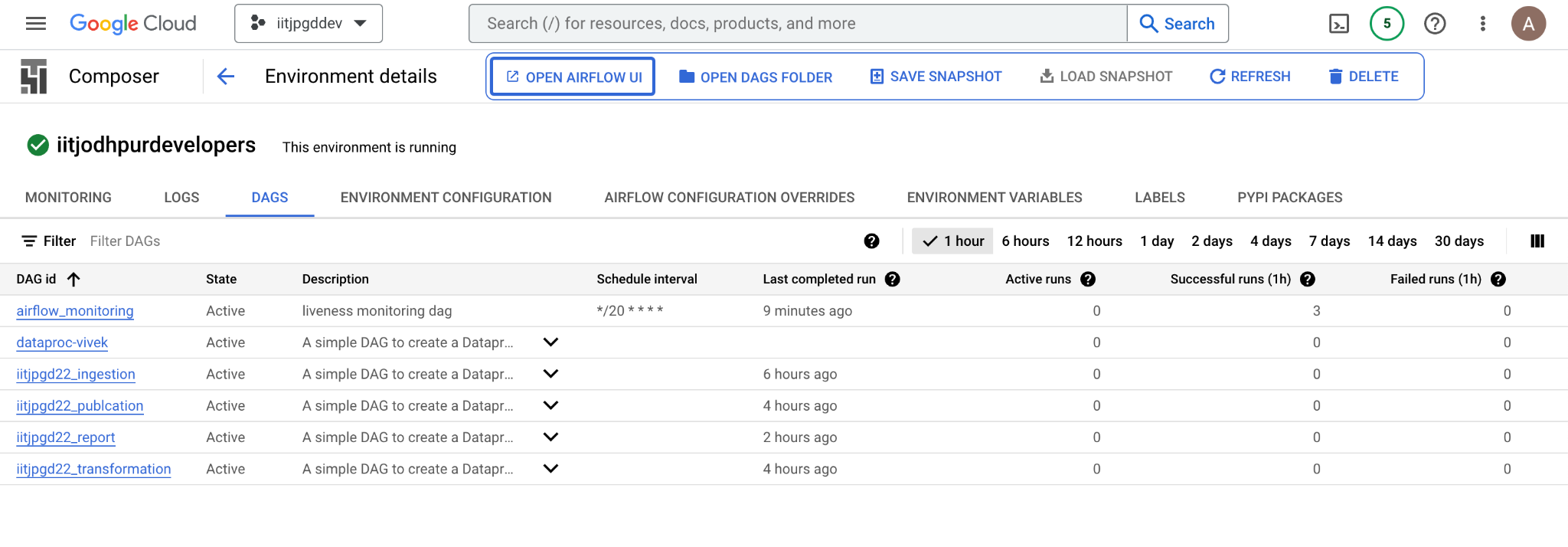
****

1. **Dag folder:**

In Apache Airflow, a DAG (Directed Acyclic Graph) folder is a directory that contains one or more DAG definition files. Each DAG definition file contains the code that defines the structure and workflow of a single DAG, which is a collection of tasks and their dependencies.

The DAG folder is typically located within the Airflow home directory, which is specified by the AIRFLOW\_HOME environment variable. By default, the DAG folder is located in the dags subdirectory of the Airflow home directory.

When Airflow starts up, it scans the DAG folder for DAG definition files and loads them into the Airflow metadata database. The DAGs are then scheduled and executed according to their specified schedule and dependencies.

****

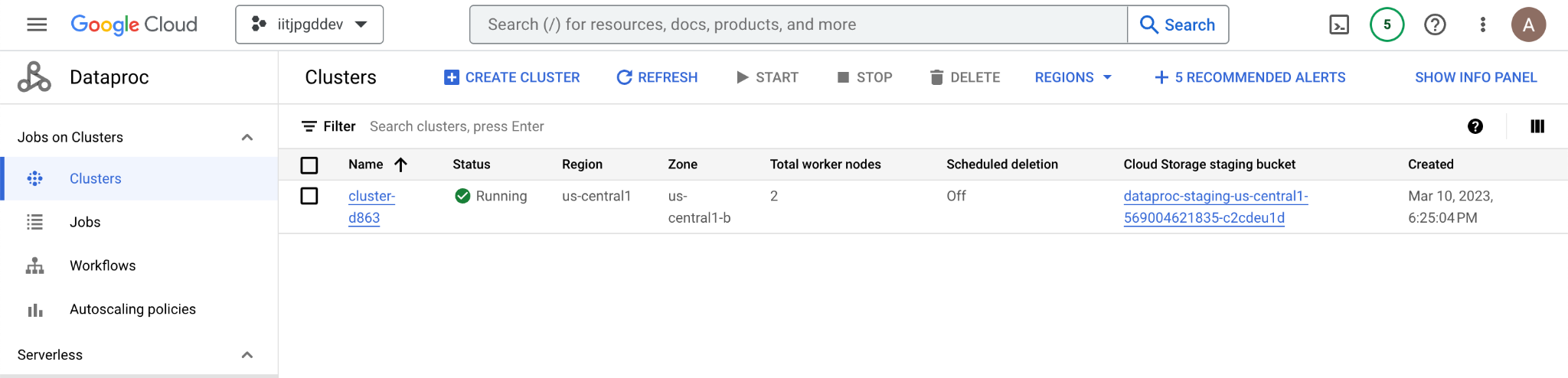
1. **DATAPROC Cluster Details:**

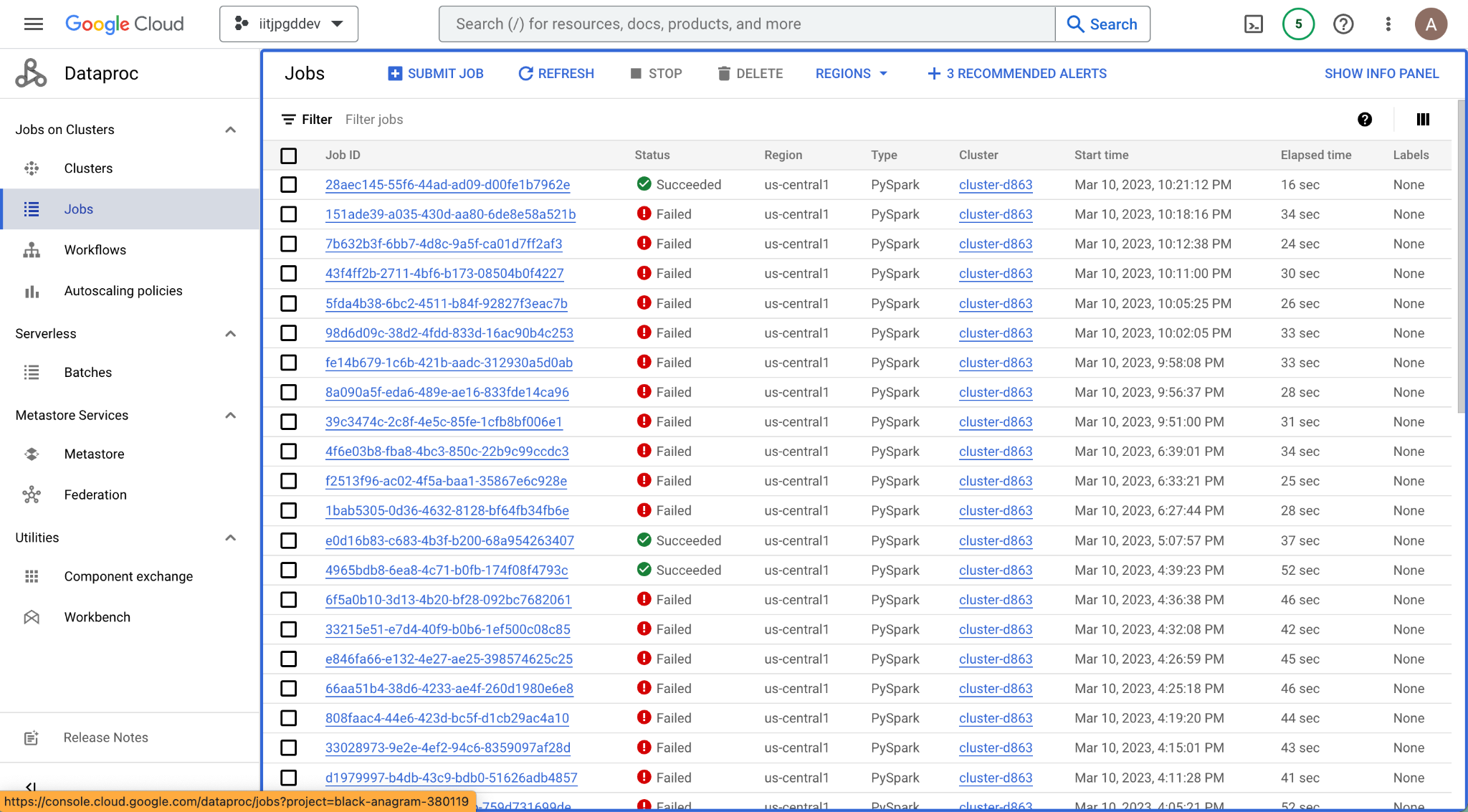
Google Cloud Dataproc is a fully-managed cloud service that provides a fast, easy, and cost-effective way to run Apache Spark, Apache Hadoop, Apache Hive, Apache Pig, and other big data processing frameworks on Google Cloud.

To create a Dataproc cluster in GCP, you can specify a number of configuration options to customize the size and performance of the cluster. Some of the key cluster details that can be specified include:

1. Cluster Name: A user-defined name for the cluster.
2. Number of Nodes: The number of nodes in the cluster. This can range from a single node to thousands of nodes.
3. Machine Type: The type of virtual machine (VM) to use for the cluster nodes. This can range from small to large, and can include custom machine types.
4. Region: The GCP region where the cluster will be located.
5. Zone: The GCP zone where the cluster will be located. This can be specified or automatically selected by GCP.
6. Network and Subnetwork: The network and subnetwork that the cluster will use for communication.
7. Initialization Actions: Optional scripts or commands that are run on each node of the cluster during initialization.
8. Cluster Software Configuration: The version of the Hadoop and Spark software that will be installed on the cluster nodes.
9. Autoscaling: Whether or not the cluster should automatically add or remove nodes based on workload.
10. Security: The security settings for the cluster, including firewall rules, service accounts, and encryption options.

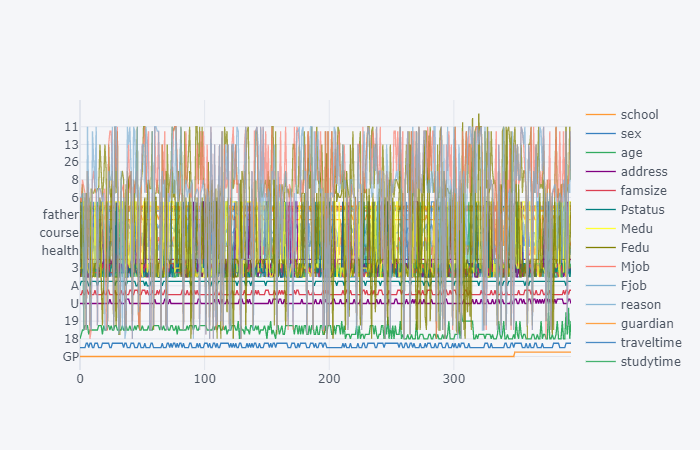
Once the cluster is created, you can monitor its performance and usage through the Dataproc web console or by using the Dataproc API or command-line tools. You can also customize the cluster settings, add or remove nodes, and manage the cluster lifecycle as needed.

****

****

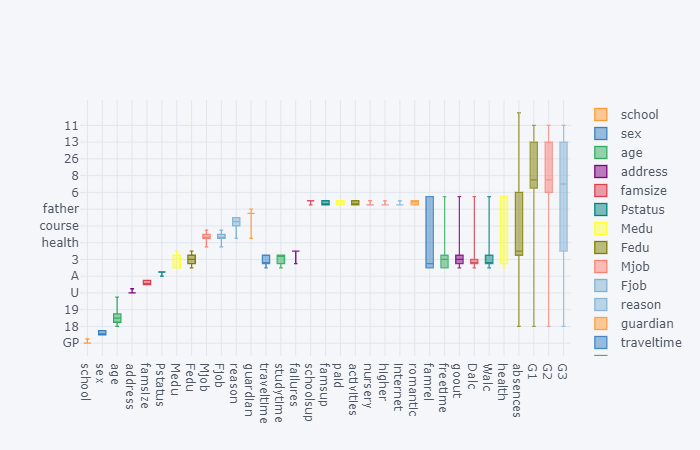
**10. Reporting Results**

10.1 - KDE Plot to view all attributes using cufflinks

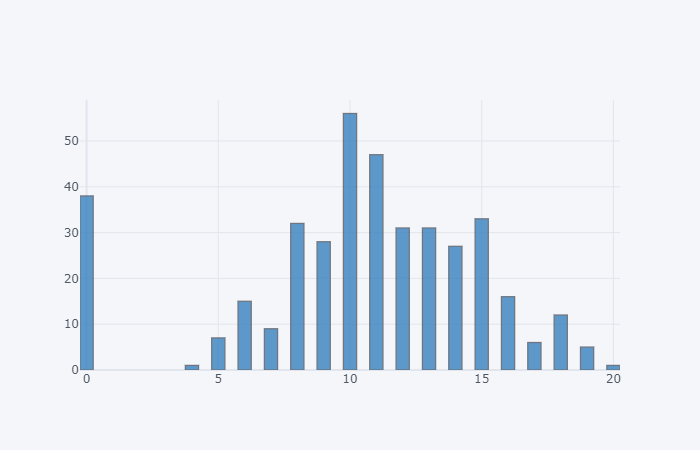


Observation: cufflink connects plotly with pandas to create graphs and charts of dataframes directly.

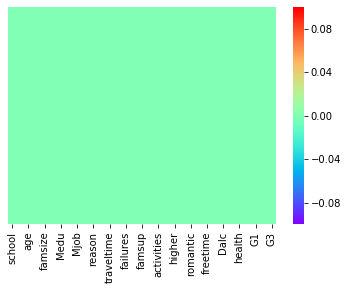
10.2 - Box Plot to view all attributes using cufflinks



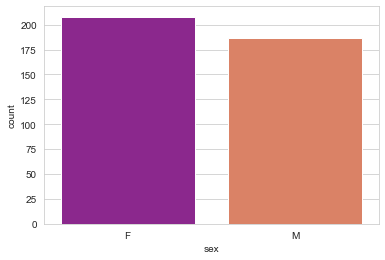
10.3 - Histogram Plot for G3 (Final Grade) using cufflinks



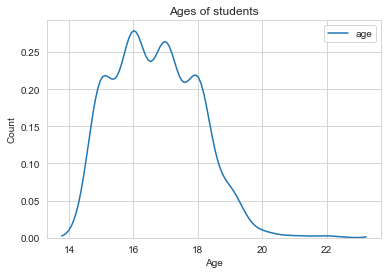
10.4 - Pictorial representation of any null data present in the dataset.



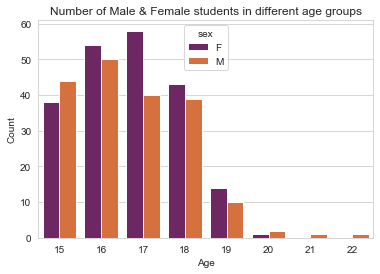
10.5 - Count Plot for Student Sex Attribute



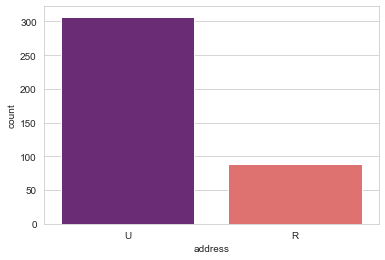
10.6 - Kernel Density Estimation for Age of Students.



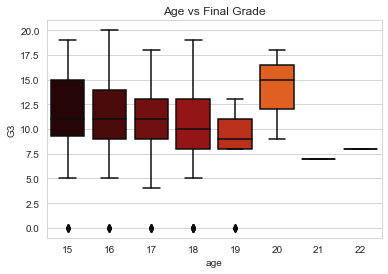
10.7 - Count PLot for Male & Female students in different age groups.



10.8 - Count Plot for students from Urban & Rural Region.



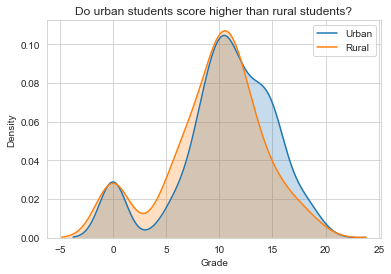
### 10.9 - Does age affect final grade?



Observation:

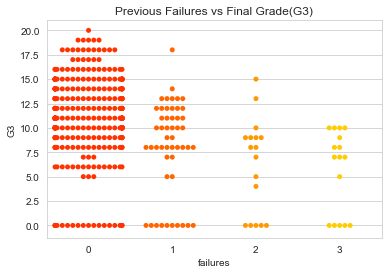
* Plotting the distribution rather than statistics would help us better understand the data.
* The above plot shows that the median grades of the three age groups(15,16,17) are similar. Note the skewness of age group 19. (may be due to sample size). Age group 20 seems to score highest grades among all.

## 10.10 - Do urban students perform better than rural students?



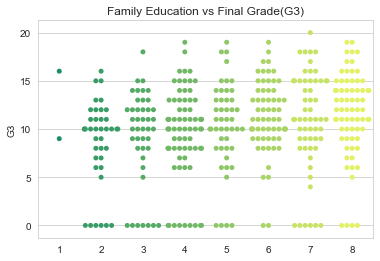
Observation: The above graph clearly shows there is not much difference between the grades based on location.

### 10.11 - Previous Failures vs Final Grade(G3)



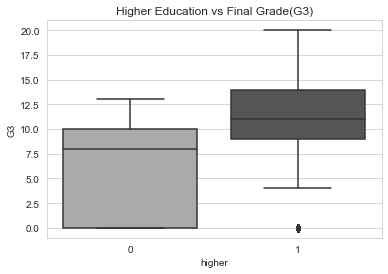
Observation: Student with less previous failures usually score higher.

10.12 - Family Education vs Final Grade(G3)



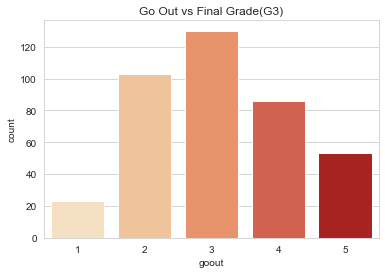
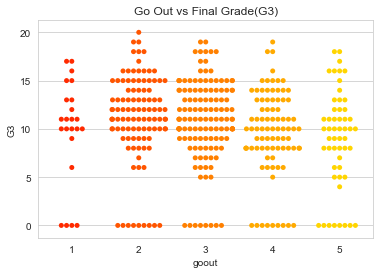
Observation: Educated families result in higher grades

10.13 - Higher Education vs Final Grade(G3)



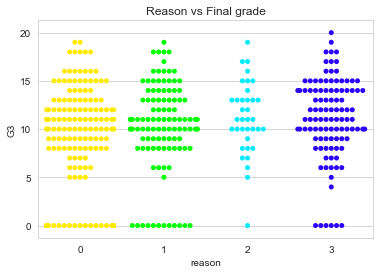
Observation: Students who wish to go for higher studies score more.

10.14 - Go Out vs Final Grade(G3)

Observation: The students have an average score when it comes to going out with friends & Students who go out a lot score less.

10.15 - Reason vs Students Count



Observation : The students have an equally distributed average score when it comes to reason attribute.

**11. Conclusion**

As we see both MAE & Model RMSE that the Linear Regression is performing the best in both cases.

