

# Data Engineering with Google Cloud Platform Training





## The world's largest provider of classroom and online training courses

- ✓ World Class Training Solutions
- ✓ Subject Matter Experts
- ✓ Highest Quality Training Material
- ✓ Accelerated Learning Techniques
- ✓ Project, Programme, and Change Management, ITIL® Consultancy
- ✓ Bespoke Tailor Made Training Solutions
- ✓ PRINCE2®, MSP®, ITIL®, Soft Skills, and More

# Course Syllabus

**Module 1:** Google Cloud Dataproc Overview

**Module 2:** Running Dataproc Jobs

**Module 3:** Integrating Dataproc with Google Cloud Platform

**Module 4:** Unstructured Data with Google's Machine Learning APIs

**Module 5:** Serverless Data Analysis with BigQuery

**Module 6:** Serverless, Autoscaling Data Pipelines with Dataflow

**Module 7:** Getting Started with Machine Learning



# Course Syllabus

**Module 8:** Building ML Models with TensorFlow

**Module 9:** Scaling ML Models with CloudML

**Module 10:** Architecture of Streaming Analytics Pipelines

**Module 11:** Ingesting Variable Volumes

**Module 12:** Implementing Streaming Pipelines

**Module 13:** Streaming Analytics and Dashboards

**Module 14:** High Throughput and Low-Latency with Bigtable



# Course Syllabus

**Module 15:** Introduction to Data Engineering

**Module 16:** Big Data Tools

**Module 17:** Apache Spark and Kafka

**Module 18:** Introduction to Apache Airflow

**Module 19:** Data Lakes and Data Warehouses

**Module 20:** Architecting Data Pipelines in GCP

**Module 21:** Advanced Machine Learning with GCP



# Module 1: Google Cloud Dataproc Overview

- Dataproc Overview
- Creating and Managing Clusters
- Custom Machine Types and Preemptible Worker Nodes



# Dataproc Overview

- ✓ Google Cloud Dataproc is a fully managed cloud service that simplifies running Apache Spark and Hadoop clusters for big data processing.
- ✓ It offers fast, scalable, and cost-effective solutions for processing large datasets, leveraging Google Cloud's infrastructure.
- ✓ Dataproc allows seamless integration with other Google Cloud services, such as BigQuery and Cloud Storage, providing a robust environment for data engineers to build and manage workflows.
- ✓ With support for a variety of big data tools, it enables users to easily run both batch and real-time data processing tasks.



# Dataproc Overview

(Continued)

- ✓ *The following are the key aspects of google cloud dataproc:*

01 *Fully Managed Service*

04 *Quick Provisioning and Auto-Scaling*

02 *Cost Efficiency*

05 *Support for Hadoop Ecosystem*

03 *Integration with Google Cloud Services*

06 *Security*

# Dataproc Overview

- Fully Managed Service:** Dataproc automatically handles the creation, scaling, and management of Spark and Hadoop clusters, removing the complexity of infrastructure management.
- Cost Efficiency:** Dataproc uses per-second billing, meaning you only pay for the resources you use, making it an affordable choice for running large-scale data processing tasks.
- Integration with Google Cloud Services:** Dataproc integrates seamlessly with other Google Cloud products like BigQuery, Cloud Storage, and Pub/Sub, facilitating easier data transfer and collaboration.
- Quick Provisioning and Auto-Scaling:** Clusters are created in minutes, and the auto-scaling feature adjusts resources according to workload demands, optimising performance and cost.
- Support for Hadoop Ecosystem:** Dataproc supports a wide range of tools from the Hadoop ecosystem, including Apache Hive, HBase, and Pig, making it versatile for various big data processing tasks.
- Security:** Dataproc provides strong security features, including encryption at rest and in transit, identity and access management (IAM), and integration with Google Cloud's security tools.

# Creating and Managing Clusters

- ✓ Creating and managing clusters in Google Cloud Dataproc allows you to provision scalable resources for running big data workloads.
- ✓ It involves setting up a cluster with the required configurations and then managing its lifecycle, such as scaling, monitoring, and deleting.
- ✓ Dataproc's fully managed nature simplifies cluster management, enabling users to focus on data processing rather than infrastructure.
- ✓ Understanding how to efficiently create and manage clusters is essential for optimising performance and cost.

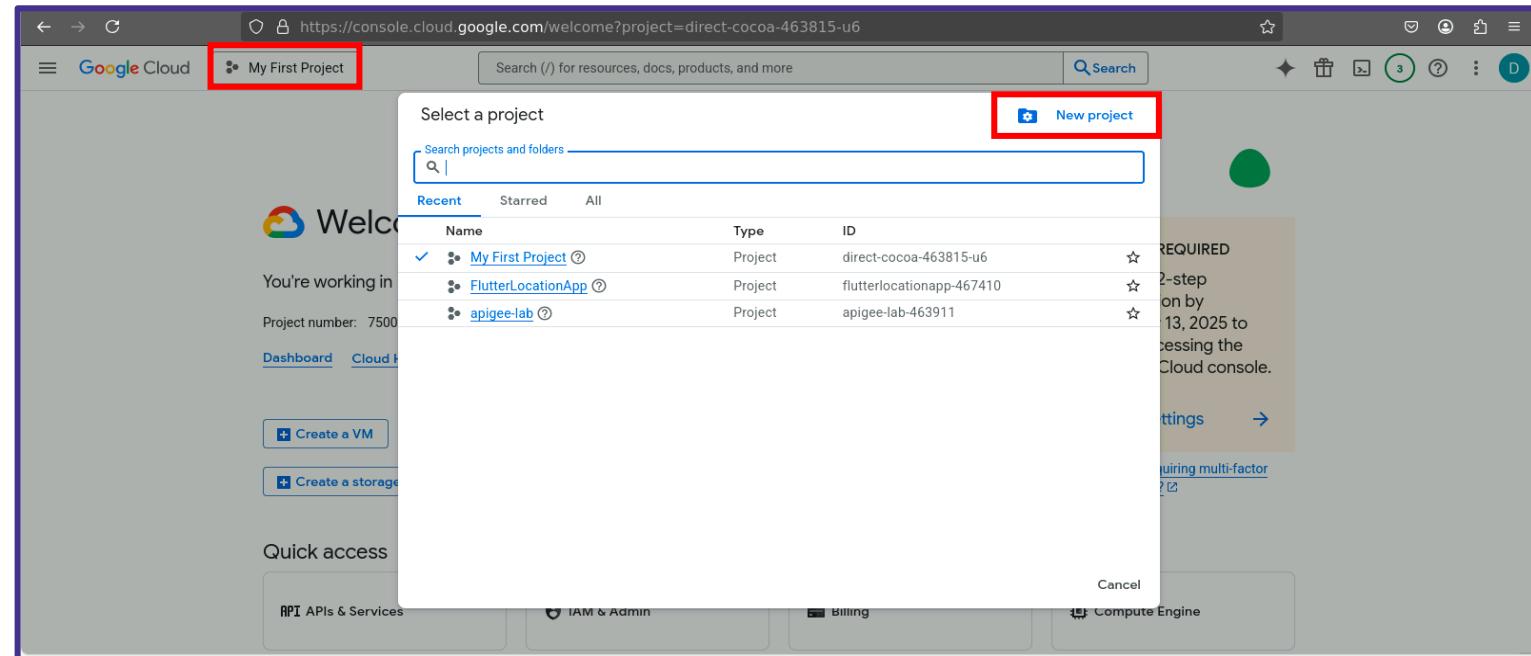


# Creating and Managing Clusters

(Continued)

- ✓ *The following are the practical steps for creating and managing clusters:*

**Step 1:** Navigate to Google Cloud Console. In the top-left corner, click on the **My First Project** dropdown and click on **New project**



# Creating and Managing Clusters

**Step 2:** Name your project and set billing and location details and then click on **Create**

The screenshot shows the 'New Project' page in the Google Cloud console. At the top left is the Google Cloud logo and a search bar. Below it, the title 'New Project' is displayed. A yellow warning box contains the message: 'You have 22 projects remaining in your quota. Request an increase or delete projects.' with a 'Learn more' link. Below this, the 'Project name \*' field is filled with 'My Project 0204'. To the right of the input field is a help icon (a question mark inside a circle). Below the project name, the 'Project ID' is listed as 'my-project-0204-469705. It cannot be changed later.' with an 'Edit' link. Under 'Location \*', the 'No organization' option is selected, with a 'Browse' button to its right. A note below says 'Parent organization or folder'. At the bottom left is a red-bordered 'Create' button, and to its right is a 'Cancel' button.

# Creating and Managing Clusters

**Step 3:** Search for **Dataproc** in the search bar, click on **Cloud Dataproc API**

The screenshot shows a search results page with a red box highlighting the search bar containing 'Dataproc'. Below the search bar, there are two product suggestions: 'Product page' and 'BigQuery Studio'. The main content area is titled 'Documentation & tutorials' and contains three items: 'Dataproc', 'Dataproc overview | Dataproc Documentation', and 'Create a Dataproc Metastore service and Dataproc cluster'. Below this is a 'Marketplace' section with 'Cloud Dataproc' and 'Dataproc Resource Manager API'. The 'Cloud Dataproc API' item is highlighted with a red box. At the bottom, a note says 'Showing resource results for My First Project only'.

- Product page
- BigQuery Studio

Documentation & tutorials

- Dataproc
- Dataproc overview | Dataproc Documentation
- Create a Dataproc Metastore service and Dataproc cluster

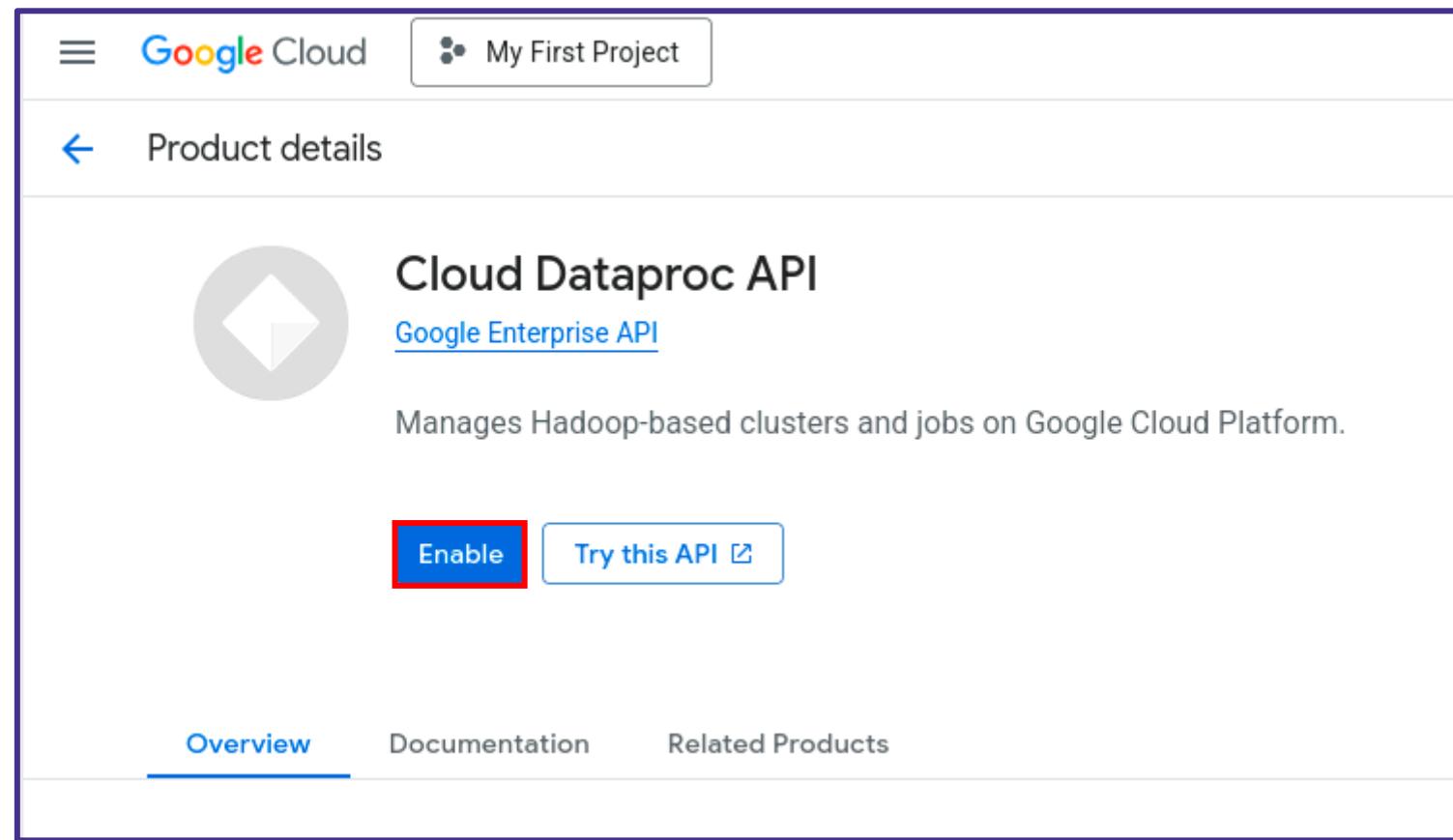
Marketplace

- Cloud Dataproc
- Dataproc Resource Manager API

Showing resource results for My First Project only

# Creating and Managing Clusters

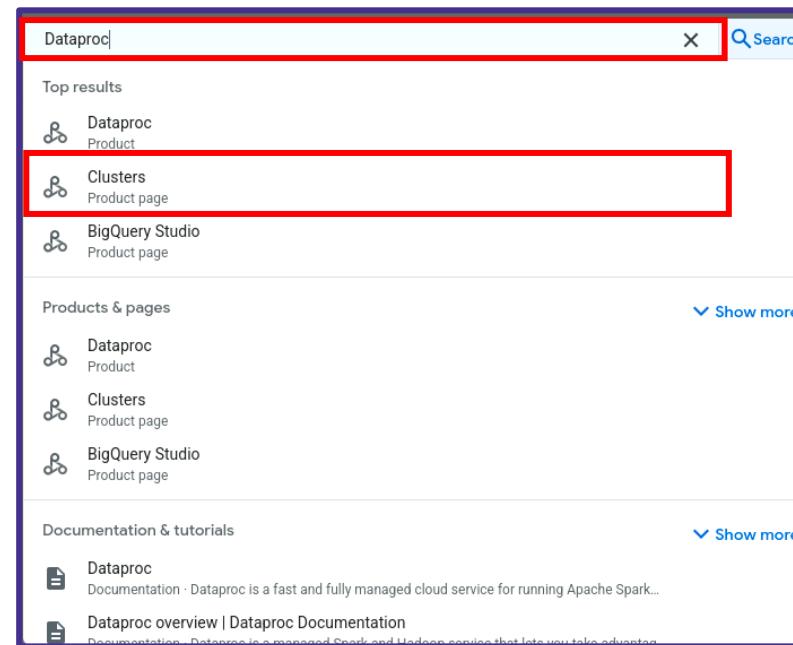
**Step 4:** After that, click **Enable**



# Creating and Managing Clusters

**Step 5:** On the left sidebar you will see a search bar at the top. Type **Dataproc** in the search bar

**Step 6:** Once you are in the Dataproc section, you will see the **Clusters** , click on it



# Creating and Managing Clusters

## Step 7: Click on Create Cluster

The screenshot shows the Google Cloud Platform interface for managing clusters. At the top, there's a navigation bar with 'Clusters' (selected), 'Create cluster' (with a plus icon), 'Refresh' (with a circular arrow icon), 'Start' (with a play icon), 'Stop' (with a square icon), 'Delete' (with a trash bin icon), 'Regions' (with a dropdown icon), and a gear icon. Below the navigation is a main content area with a title 'Cluster' and a specific section for 'Cloud Dataproc'. The 'Cloud Dataproc' section contains the following text:

Google Cloud Dataproc lets you provision Apache Hadoop clusters and connect to underlying analytic data stores.

There are no clusters in the currently selected Cloud Dataproc region(s). Create a cluster to get started.

A blue 'Create cluster' button is prominently displayed at the bottom of this section. This button is highlighted with a red rectangular box. To the right of the main content area, there's a sidebar with the following sections:

- Create a Dataproc on Compute Engine cluster**
  - Create a cluster using the Google Cloud console** (with a 'Help document' link)
  - Create a cluster using the gcloud CLI** (with a 'Help document' link)
  - Create a cluster using the Dataproc API** (with a 'Help document' link)
- All Dataproc documentation**

At the bottom right of the sidebar, there's a 'View more →' link.

# Creating and Managing Clusters

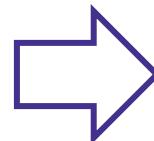
**Step 8:** Configure the cluster by fill the required details and then click on **Create**

Compute Engine

Name  
Cluster Name \*  [?](#)

Location  
Region \*  [?](#) Zone \*  [?](#)

Cluster type  
 Standard (1 master, N workers)  
 Single Node (1 master, 0 workers)  
Provides one node that acts as both master and worker. Good for proof-of-concept or small-scale processing  
 High Availability (3 masters, N workers)  
Hadoop High Availability mode provides uninterrupted YARN and HDFS operations despite single-node failures or reboots



Create a Dataproc cluster on Compute Engine

← Set up cluster  
Begin by providing basic information.

Configure nodes (optional)  
Change node compute and storage capabilities.

Customize cluster (optional)  
Add cluster properties, features, and actions.

Manage security (optional)  
Change access, encryption, and security settings.

**Create** [Cancel](#)

Versioning  
Use a custom image to load pre-installed packages. [Learn more](#)

Image Type and Version

Release Date  
First released on 06/09/20; [Change](#)

Spark performance enhancements

Enable advanced optimizations [?](#)  
 Enable advanced execution layer [?](#)  
 Enable Google Cloud Storage caching [?](#)

Autoscaling  
Automates cluster resource management based on an autoscaling policy.

Policy

# Creating and Managing Clusters

**Step 9:** Click on the cluster you want to delete, click **Delete** to remove the cluster

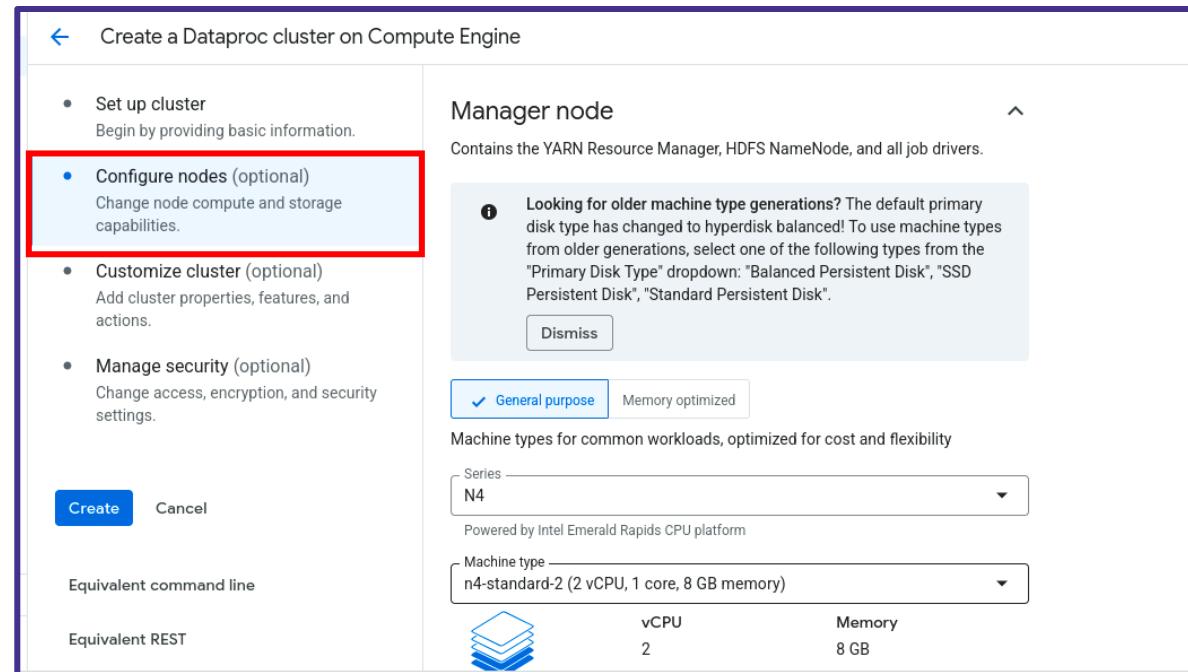
The screenshot shows the Google Cloud Kubernetes Engine Clusters interface. On the left, a sidebar under 'Resource Management' has 'Clusters' selected. The main area displays a single cluster named 'autopilot-cluster-1' in the 'us-central1' location with a 'Standard' tier. A red box highlights the 'Delete' button at the top right of the cluster card. Another red box highlights the entire cluster card for 'autopilot-cluster-1'. The cluster card includes sections for Overview, Observability, and Cost Optimization, along with a table of cluster details.

Status	Name	Location	Tier	Number of nodes	Total vCPUs	Total memory	Notificat
<input checked="" type="checkbox"/>	<a href="#">autopilot-cluster-1</a>	us-central1	Standard	0	0 GB	<a href="#">Verify webhook endpoint</a>	

# Custom Machine Types and Preemptible Worker Nodes

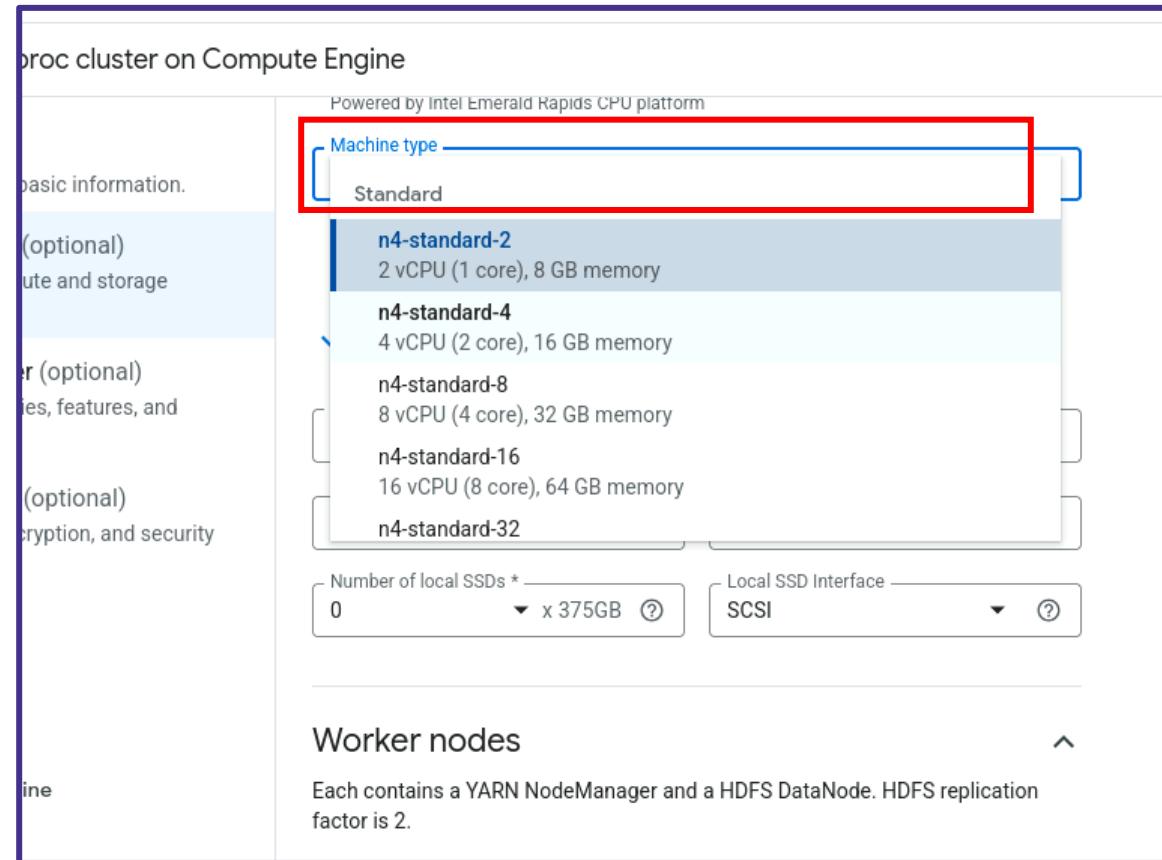
- ✓ Custom machine types in Google Cloud Dataproc allow you to configure your cluster to meet specific requirements, such as more CPU or memory, without the need to select predefined machine types.
- ✓ ***The following are the steps to custom machine types:***

**Step 1:** Navigate to Create Cluster. In the Cluster Configuration screen, go to the **Configure Nodes** section



# Custom Machine Types and Preemptible Worker Nodes

Step 2: Under **Machine type**, choose Custom instead of the predefined options



# Custom Machine Types and Preemptible Worker Nodes

**Step 3:** Set the number of vCPUs and **Memory (RAM)** based on your requirements for both master and worker nodes

**Step 4:** Review your settings and click **Create** to deploy your cluster with custom machine types

The screenshot shows the 'Create a Dataproc cluster on Compute Engine' interface. Under the 'General purpose' tab, the 'Machine type' dropdown is set to 'e2-standard-2 (2 vCPU, 1 core, 8 GB memory)', which is highlighted with a red border. Below the dropdown, the configuration shows 2 vCPUs and 8 GB of Memory. The 'CPU platform' is set to 'Automatic'. Other tabs like 'Compute optimized' and 'Memory optimized' are visible but not selected.

The screenshot shows the 'Create a Dataproc cluster on Compute Engine' interface. The 'Configure nodes (optional)' section is currently active. On the right, the cluster configuration is displayed: 'n4-standard-2 (2 vCPU, 1 core, 8 GB memory)' with 2 vCPUs. The 'CPU platform and GPU' section shows 2 worker nodes, 200 GB primary disk size, and 0 local SSDs. The 'Create' button is highlighted in blue. A progress bar at the bottom indicates 'Creating cluster cluster-3193'.

# Module 2: Running Dataproc Jobs

- Running Pig and Hive Jobs
- Separation of Storage and Compute
- Running Hadoop and Spark Jobs with Dataproc



# Running Pig and Hive Jobs

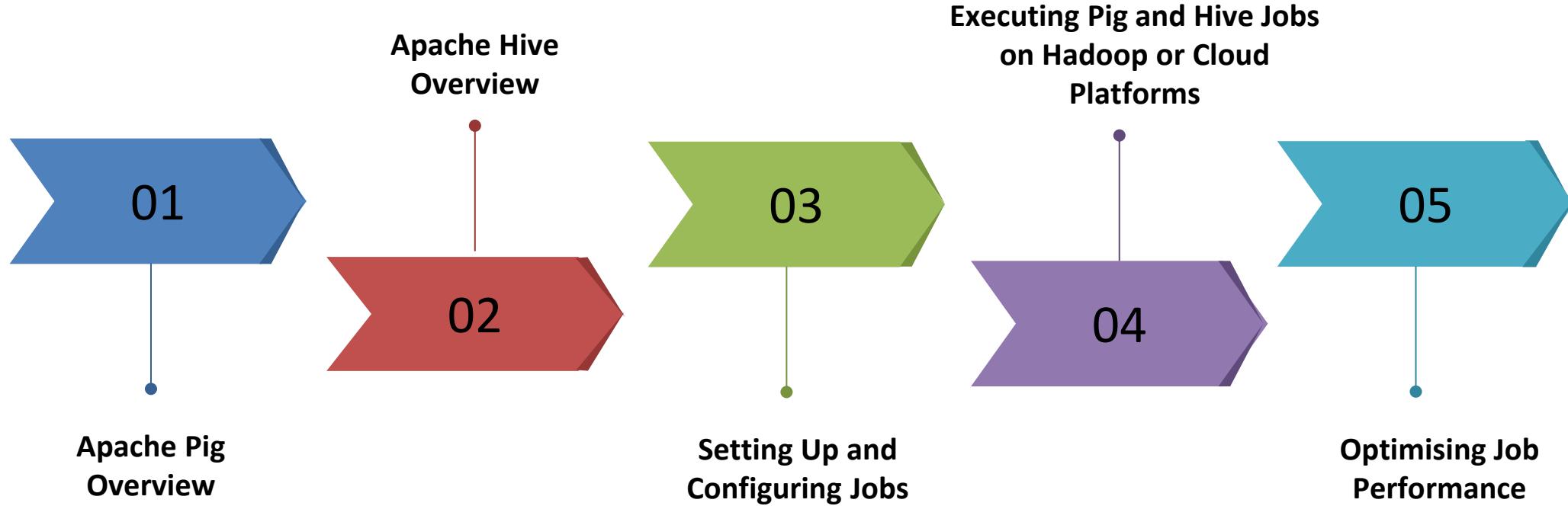
- ✓ Running Pig and Hive jobs on Apache Hadoop or in cloud environments like Google Cloud Platform allows for large-scale data processing and analytics.
- ✓ Tools offer high-level abstractions for managing and querying large datasets, making them essential components for handling big data workloads.
- ✓ Apache Pig, with its data flow language, and Apache Hive, with its SQL-like query language, provide flexible and scalable ways to process, analyse, and transform massive datasets stored in Hadoop's HDFS.
- ✓ These tools are ideal for data engineers and analysts working with complex data pipelines.



# Running Pig and Hive Jobs

(Continued)

- ✓ *The following are the key aspects for running pig and hive jobs:*



# Running Pig and Hive Jobs

## 1. Apache Pig Overview:

- Pig is a high-level platform that simplifies data processing tasks by providing a scripting language called Pig Latin.
- It is designed for parallel processing on large datasets, offering a more flexible, script-based alternative to traditional MapReduce.
- Running Pig jobs involves writing Pig scripts, which are then executed on the Hadoop cluster. Pig jobs are often used for ETL (Extract, Transform, Load) operations, such as filtering, joining, and transforming data.

## 2. Apache Hive Overview:

- Hive provides a SQL-like query interface on top of Hadoop, allowing users to query large datasets using a familiar relational model.
- It converts SQL queries into MapReduce jobs to process data stored in Hadoop HDFS or other compatible systems.

# Running Pig and Hive Jobs

### **3. Setting Up and Configuring Jobs:**

- Before running Pig or Hive jobs, you must configure the respective environment. For Pig, this includes setting up Pig properties (e.g., setting HDFS paths, specifying execution mode).
- For Hive, configurations may involve setting Hive Metastore, tuning execution settings (e.g., memory limits), and defining the Hadoop cluster settings to optimise the job execution.

### **4. Executing Pig and Hive Jobs on Hadoop or Cloud Platforms:**

- Jobs can be run in different environments, such as on a local Hadoop cluster or in cloud platforms like Google Cloud Dataproc.
- Both Pig and Hive jobs can be run using command-line tools or through integrated interfaces such as Hue or Apache Ambari for Hadoop.
- Additionally, cloud environments offer managed services to simplify the process of running and scaling these jobs.

# Running Pig and Hive Jobs

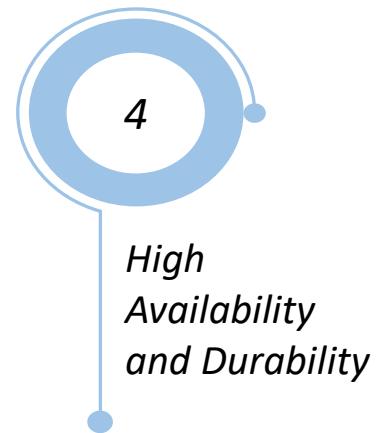
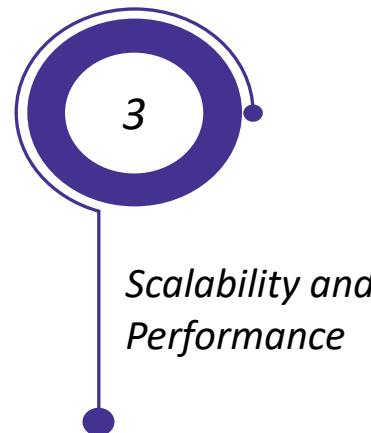
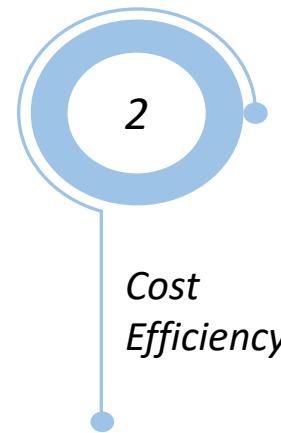
## 5. Optimising Job Performance:

- Optimising the performance of Pig and Hive jobs involves tuning the underlying Hadoop configuration, such as adjusting MapReduce settings, using compression to reduce disk usage, and fine-tuning data storage formats like Parquet or ORC for Hive.
- In Pig, performance can be optimised by using Pig UDFs (User Defined Functions) efficiently and reducing unnecessary data transformations.



# Separation of Storage and Compute

- ✓ The separation of storage and compute is a key architectural principle in modern cloud platforms, including Google Cloud. By decoupling these two components, it allows for more flexibility, scalability, and cost optimisation.
- ✓ In Dataproc, this approach means that storage and compute resources are managed independently, enabling users to scale them according to their specific needs without being bound by each other.
- ✓ ***The following are the key points for separation of storage and compute:***



# Running Hadoop and Spark Jobs with Dataproc

- ✓ Google Cloud Dataproc simplifies the process of running and managing both Hadoop and Spark jobs in the cloud, providing a fully managed, fast, and cost-effective service for large-scale data processing. Dataproc integrates seamlessly with other Google Cloud services, such as Cloud Storage, BigQuery, and Cloud Pub/Sub, making it ideal for big data workflows.
- ✓ *The following are the key points for running hadoop and spark jobs with dataproc:*



# Running Hadoop and Spark Jobs with Dataproc

## 1. Quick Cluster Provisioning and Configuration:

- Dataproc allows you to quickly provision Hadoop or Spark clusters on Google Cloud with just a few commands. The cluster can be easily customized by specifying the number and type of nodes, the software to be included (Hadoop, Spark, etc.), and other configurations based on the job's requirements.
- The ability to automatically configure clusters reduces setup time and enables you to focus more on data processing tasks rather than infrastructure management.

## 2. Support for Both Hadoop and Spark Workloads:

- Dataproc supports both Hadoop and Spark, making it versatile for different big data use cases. You can submit a variety of jobs, including batch processing (MapReduce jobs in Hadoop) or real-time processing (Spark Streaming), directly to the cluster.
- The flexibility to run either or both workloads on the same cluster provides a unified approach to big data processing.

# Running Hadoop and Spark Jobs with Dataproc

## 3. Integration with Google Cloud Services:

- Dataproc clusters are tightly integrated with Google Cloud's ecosystem, enabling you to use Cloud Storage for storing large datasets, BigQuery for running fast SQL queries, and Cloud Pub/Sub for real-time streaming data. The integration allows for seamless data flow between services.
- This integration boosts productivity and simplifies the orchestration of complex workflows, enabling faster insights from the data.

## 4. Autoscaling for Flexibility and Cost Efficiency:

- Dataproc supports autoscaling, where the cluster can automatically adjust the number of worker nodes based on workload demands. For instance, when a Spark or Hadoop job requires more compute resources, Dataproc can add nodes to the cluster, and scale them back down when the workload decreases.
- This dynamic scaling helps ensure efficient use of resources, lowering costs by only using what's necessary for processing.

# Running Hadoop and Spark Jobs with Dataproc

## 5. Job Submission via Multiple Methods:

- Jobs can be submitted to Dataproc clusters using various methods, including the Google Cloud Console, gcloud command-line tool, and REST API.
- You can also use the Dataproc Workflow Templates to define and run complex workflows with multiple stages and dependencies.
- The ability to submit jobs through different interfaces provides flexibility, catering to different operational preferences and automation needs.



# Module 3: Integrating Dataproc with Google Cloud Platform

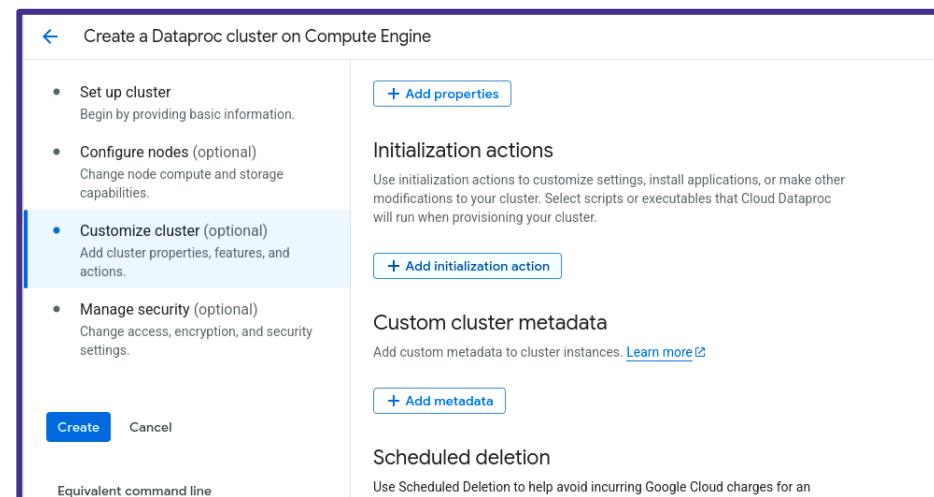
- Customize Cluster with Initialisation Activities
- BigQuery Support
- GCP Services



# Customize Cluster with Initialisation Activities

- ✓ Customizing clusters with initialisation activities in Google Cloud Dataproc allows you to run custom scripts or install additional software packages when the cluster is created or when new nodes are added.
- ✓ These initialisation actions are useful for pre-configuring your cluster with dependencies like libraries, software, or system configurations that are necessary for your workload.
- ✓ ***The following are the steps to customize cluster with initialisation activities:***

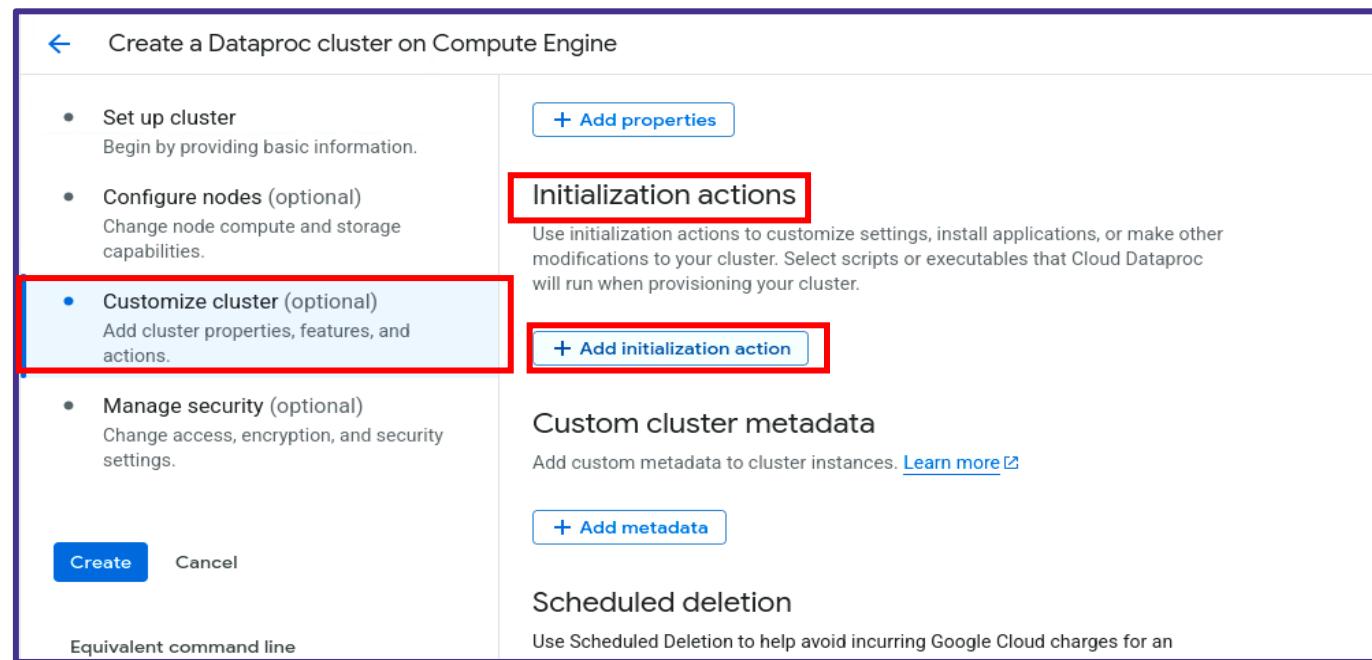
**Step 1:** Navigate to Create Cluster in the Dataproc section, you should be directed to the Cluster Configuration screen



# Customize Cluster with Initialisation Activities

**Step 2:** In the Cluster Configuration window, scroll down until you see the **Customize Cluster** section

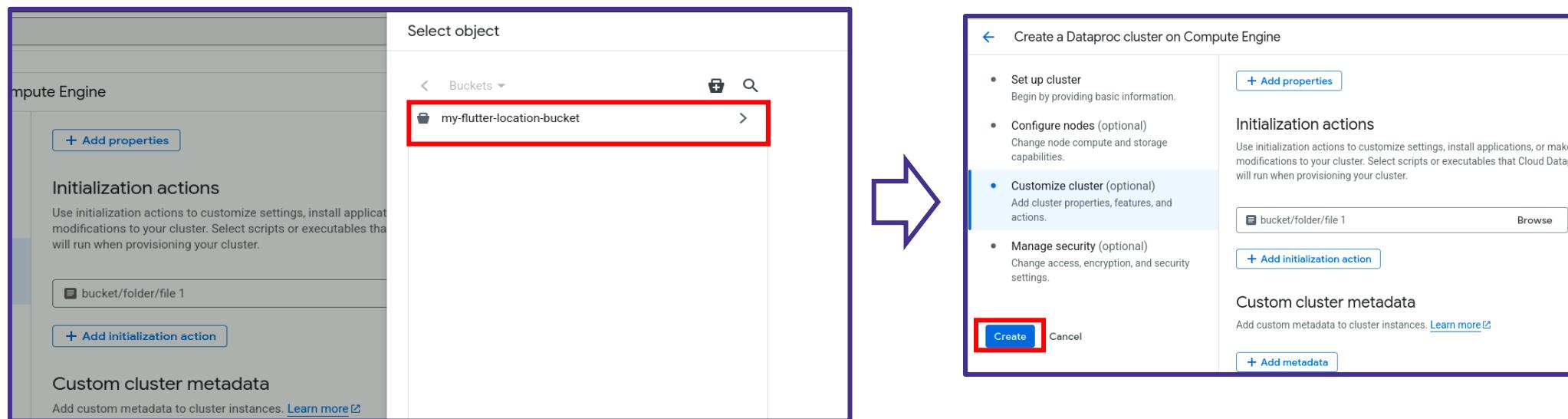
**Step 3:** You will find a subsection called **Initialisation actions**, click **Add initialization action** in this section to specify the location of your initialisation script



# Customize Cluster with Initialisation Activities

**Step 4:** Select the object in this section to specify the location of your initialisation script, click on **Create**

- ✓ You can add multiple scripts if you need to install different tools or software for different nodes



# BigQuery Support

- ✓ Dataproc's integration with BigQuery enables users to seamlessly use both systems together, processing large datasets on Dataproc clusters and then storing the processed results in BigQuery for easy analysis using SQL.
- ✓ You can use Dataproc for data engineering tasks such as ETL (Extract, Transform, Load) and then push the results to BigQuery for analytics or business intelligence (BI) applications.
- ✓ ***The following are the key benefits of BigQuery support in Dataproc:***
  1. **Seamless Data Movement:** Easily move data between Dataproc and BigQuery.
  2. **Advanced Analytics:** Use BigQuery's powerful SQL engine to query large datasets processed on Dataproc.
  3. **Cost Efficiency:** Leverage BigQuery's scalable storage and compute power, combined with Dataproc's cluster flexibility.
  4. **Integration with Google Cloud Services:** Tight integration with other Google Cloud services, making it easier to build and manage end-to-end data pipelines.

# GCP Services

- ✓ Google Cloud Platform (GCP) offers a comprehensive suite of cloud services that enable businesses and developers to build, deploy, and scale applications. GCP provides powerful tools and services for computing, data storage, machine learning, networking, and data analytics, among others.
- ✓ These services allow organisations to leverage Google's robust infrastructure, data security, and innovations in AI and machine learning, providing them with scalable solutions for both small and large-scale applications.
- ✓ Whether for processing big data, hosting virtual machines, or creating AI-driven applications, GCP services are designed to support a wide range of use cases.
- ✓ ***The following are the key aspects of GCP services:***



Compute



Storage & Database



Networking



Big Data



Developer Tools



Identity & Security



Internet of Things



Cloud AI



Management Tools



Data Transfer

# GCP Services

1. **Compute:** Services like Google Compute Engine and Kubernetes Engine for running virtual machines and containerised applications.
2. **Storage & Database:** Google Cloud Storage and databases like BigQuery for scalable, secure data storage and management.
3. **Networking:** Tools for managing network resources, including Cloud VPC and Cloud CDN.
4. **Big Data:** Services like Dataproc and Dataflow for large-scale data processing and analytics.
5. **Developer Tools:** Tools like Cloud Build and Cloud Source Repositories for automating development workflows.
6. **Identity & Security:** Services for managing users, roles, and data protection, including IAM and Cloud Security Command Center.



# GCP Services

7. **Internet of Things (IoT)**: Solutions for managing IoT devices and data streams, such as Cloud IoT Core.
8. **Cloud AI**: Machine learning and artificial intelligence tools, like AI Platform and TensorFlow, for building smart applications.
9. **Management Tools**: Monitoring, logging, and cost management tools for maintaining and optimising GCP resources.
10. **Data Transfer**: Services like Cloud Storage Transfer and BigQuery Data Transfer for moving large datasets to and from GCP.



# Module 4: Unstructured Data with Google's Machine Learning APIs

- Machine Learning APIs
- Use Cases of ML
- Adding Machine Learning Capabilities to Big Data



# Machine Learning APIs

- ✓ Google Cloud's Machine Learning APIs provide developers with easy access to advanced machine learning models without the need for building custom algorithms. These APIs are pre-trained, making them accessible for a wide range of tasks such as image recognition, natural language processing, speech recognition, and translation.
- ✓ The APIs leverage Google's powerful AI and machine learning technologies, enabling users to integrate machine learning capabilities into their applications with minimal effort.
- ✓ ***The following are the key aspects of google cloud machine learning APIs:***



Pre-trained  
Models



Wide Range  
of Use Cases



Ease of Use



Scalable and  
Cost-Effective



Security and  
Privacy



No ML Expertise  
Required



Integration with Other  
Google Cloud Services

# Machine Learning APIs

## 1. Pre-trained Models:

- Google's Machine Learning APIs provide pre-trained models, meaning you don't have to train a model from scratch. You can easily integrate AI capabilities into your apps and services.
- Examples include Vision AI for image recognition and Cloud Natural Language API for text analysis.

## 2. Wide Range of Use Cases:

- **Image Analysis:** APIs like Cloud Vision API allow you to detect objects, text, and faces in images.
- **Text Analysis:** The Cloud Natural Language API helps with sentiment analysis, entity recognition, and syntax analysis of text.
- **Speech Recognition:** With the Cloud Speech-to-Text API, you can convert audio recordings into text with high accuracy.
- **Translation:** The Cloud Translation API supports real-time language translation, helping bridge communication gaps.

# Machine Learning APIs

## 3. Ease of Use:

- These APIs are designed to be simple to integrate into your application using RESTful endpoints or client libraries in various programming languages (Python, Java, Node.js, etc.).
- The APIs are scalable and can be integrated into existing systems with minimal configuration.

## 4. Scalable and Cost-Effective:

- Google Cloud's Machine Learning APIs are designed to scale with your needs, allowing you to process large volumes of data without worrying about infrastructure.
- Pricing is based on usage, making it a cost-effective solution for businesses of all sizes.

## 5. Security and Privacy:

- The APIs are built with Google's security best practices, ensuring that data is processed securely.

# Machine Learning APIs

## 6. No ML Expertise Required:

- These APIs abstract the complexity of machine learning, allowing non-experts to easily use powerful AI models in their applications.
- By using the APIs, developers can add AI functionalities without the need for extensive knowledge in machine learning or data science.

## 7. Integration with Other Google Cloud Services:

- Google's Machine Learning APIs integrate seamlessly with other Google Cloud services like BigQuery, Cloud Storage, and Google Kubernetes Engine, allowing for a streamlined data processing and analysis pipeline.
- You can combine these APIs with tools like Google Data Studio to visualize the outputs of machine learning tasks.

# Use Cases of ML

- ✓ Machine Learning (ML) has a wide range of applications across various industries. Its ability to analyse vast amounts of data, detect patterns, and make predictions makes it incredibly useful in solving complex problems and enhancing business operations.
- ✓ ***The following are the key use cases of machine learning that demonstrate its versatility and power in different sectors:***

1. *Image and Object Recognition*

2. *Natural Language Processing*

3. *Predictive Analytics*

4. *Speech Recognition and Voice Assistants*

5. *Recommendation Systems*

# Use Cases of ML

## 1. Image and Object Recognition:

- ML is widely used in image processing for tasks like facial recognition and object detection.
- Applications like Google Vision API can automatically identify and categorise objects, faces, and text in images, making it useful for security, retail, and social media platforms.

## 2. Natural Language Processing (NLP):

- NLP uses machine learning to analyse and generate human language. Use cases include sentiment analysis, chatbots, and language translation, helping businesses understand customer feedback and automate customer service interactions.

## 3. Predictive Analytics:

- Machine learning models analyse historical data to predict future trends. This is valuable in sectors like finance (e.g., stock market prediction), healthcare (e.g., patient diagnosis prediction), and marketing (e.g., customer behavior forecasting).

# Use Cases of ML

## 4. Speech Recognition and Voice Assistants:

- ML is used to convert speech to text and interpret voice commands.
- Voice assistants like Google Assistant and Siri rely on ML to understand natural language commands and improve over time by learning from user interactions.

## 5. Recommendation Systems:

- ML algorithms analyse user behavior to provide personalised recommendations.
- Services like Netflix, Amazon, and YouTube use ML to suggest content or products tailored to individual preferences, improving user experience and engagement.



# Adding Machine Learning Capabilities to Big Data

- ✓ Integrating machine learning (ML) capabilities with big data enables organisations to unlock insights and patterns from vast amounts of data, improving decision-making processes and predictive analytics.
- ✓ By applying ML algorithms to big data, businesses can automate processes, make more accurate predictions, and discover hidden trends that are not immediately obvious.
- ✓ *The following are the key aspects of adding machine learning capabilities to big data:*



# Adding Machine Learning Capabilities to Big Data

## 1. Data Preprocessing:

- Before applying machine learning models to big data, preprocessing is essential to clean, transform, and normalise the data. This step ensures the quality of the data and prepares it for effective training of ML models.
- Techniques such as feature extraction, data imputation, and dimensionality reduction are commonly used to handle large datasets efficiently.

## 2. Distributed Machine Learning:

- Big data tools like Apache Spark and Hadoop enable distributed machine learning, where the data is split across multiple nodes to allow for parallel processing. This makes it feasible to train machine learning models on massive datasets in a scalable and cost-effective manner.
- Distributed ML techniques like MapReduce and MLLib in Apache Spark allow for running machine learning algorithms on large datasets.

# Adding Machine Learning Capabilities to Big Data

## 3. Real-Time Analytics:

- Machine learning models can be applied to big data streams for real-time analysis. For example, using Apache Kafka or Google Cloud Pub/Sub, data can be ingested and analysed as it is generated, enabling real-time predictions and decision-making.
- This is useful in applications like fraud detection, predictive maintenance, and real-time recommendation systems.

## 4. Predictive Analytics and Forecasting:

- ML models can analyse historical big data and predict future trends. This can be applied in fields like financial services for stock market predictions, healthcare for disease outbreak predictions, or e-commerce for forecasting demand and inventory management.
- Techniques like time-series forecasting, regression models, and deep learning can be used to forecast outcomes based on historical data.

# Adding Machine Learning Capabilities to Big Data

## 5. Automated Data Labeling and Classification:

- Big data often includes unstructured data, such as text, images, and video.
- ML algorithms, such as deep learning, are used to automatically classify, label, and organise unstructured data into meaningful categories.
- This is particularly useful in industries like healthcare (for medical image classification), retail (for customer sentiment analysis), and social media (for content moderation and classification).



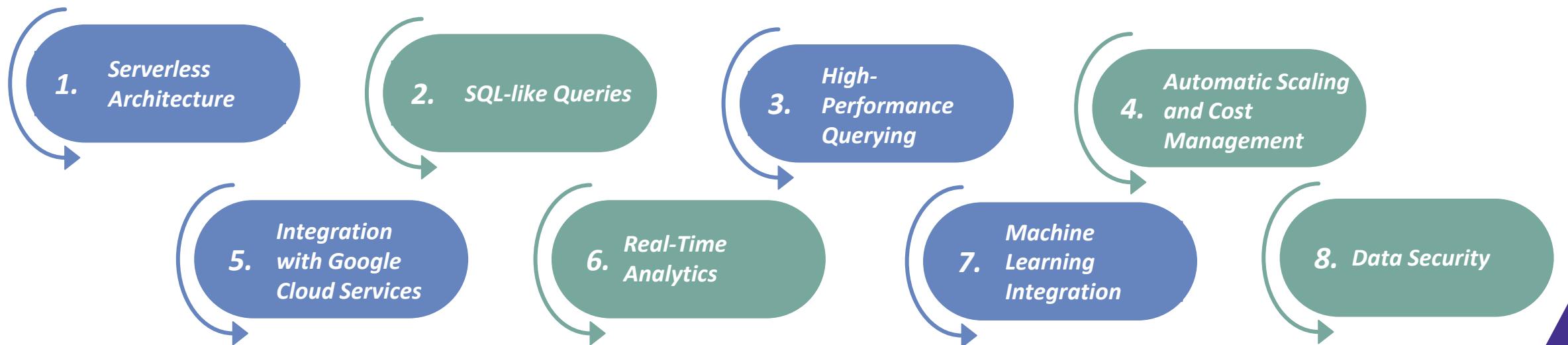
# Module 5: Serverless Data Analysis with BigQuery

- BigQuery Overview
- Functions and Queries
- Writing Queries in BigQuery



# BigQuery Overview

- ✓ Google BigQuery is a fully-managed, serverless data warehouse designed to handle massive datasets with high-speed querying and real-time analytics. Built on top of Google Cloud's scalable infrastructure, BigQuery allows businesses to store, analyse, and query large volumes of data quickly and cost-effectively.
- ✓ With BigQuery, users can run SQL-like queries on structured and semi-structured data without worrying about managing servers or infrastructure. It provides advanced capabilities such as automatic scaling, high availability, and integrated machine learning for seamless, large-scale data analytics.
- ✓ ***The following are the key aspects of BigQuery:***



# BigQuery Overview

## 1. Serverless Architecture:

- BigQuery's serverless model removes the need for users to manage infrastructure, making it easy to scale automatically based on the amount of data being processed. This allows for faster query execution without worrying about provisioning or maintaining hardware.

## 2. SQL-like Queries:

- BigQuery uses a variant of SQL, enabling users to run complex analytical queries on structured and semi-structured data. It provides a familiar environment for data analysts who are accustomed to SQL-based tools.

## 3. High-Performance Querying:

- BigQuery utilises Google's distributed architecture to perform high-speed analytics on massive datasets. It can process petabytes of data efficiently, enabling users to derive insights quickly, even from large-scale datasets.

# BigQuery Overview

## 4. Automatic Scaling and Cost Management:

- BigQuery scales automatically to handle data growth, ensuring that performance remains optimal as workloads increase. Pricing is based on the amount of data queried, so businesses only pay for what they use, making it a cost-efficient solution for both small and large organisations.

## 5. Integration with Google Cloud Services:

- BigQuery seamlessly integrates with other Google Cloud services, such as Google Cloud Storage, Google Data Studio, and Google Sheets, allowing for smooth data workflows and easy access to your data for analysis and reporting.

## 6. Real-Time Analytics:

- BigQuery supports real-time analytics by enabling users to query live data, which is ideal for time-sensitive use cases like monitoring systems, fraud detection, and IoT analytics.

# BigQuery Overview

## 7. Machine Learning Integration:

- BigQuery integrates with BigQuery ML, allowing users to build and deploy machine learning models directly within BigQuery using SQL queries.
- This simplifies the process of applying machine learning to your data without moving it to other systems.

## 8. Data Security:

- BigQuery offers robust security features, including IAM (Identity and Access Management), data encryption at rest and in transit, and fine-grained access control to protect sensitive information.



# Functions and Queries

- ✓ BigQuery allows users to perform powerful data analysis using SQL-like queries and a variety of functions for manipulating, transforming, and aggregating data.
- ✓ These functions are essential for tasks like data cleaning, statistical analysis, text processing, and more. By using BigQuery's built-in functions and writing efficient SQL queries, users can extract meaningful insights from large datasets quickly and effectively.
- ✓ ***The following are the key aspects of functions and queries in BigQuery:***

1

*Standard SQL  
Queries*

2

*Built-in Functions*

3

*User-Defined  
Functions*

4

*Subqueries and  
Nested Queries*

5

*Window Functions*

6

*Join Operations*

7

*Query Optimisation*

8

*Handling Large  
Datasets*

# Functions and Queries

## 1. Standard SQL Queries:

- BigQuery supports Standard SQL which is similar to SQL used in traditional relational databases. This allows users to execute complex queries using familiar syntax for filtering, sorting, joining, and aggregating data.

## 2. Built-in Functions:

- BigQuery provides a rich set of built-in functions that help users to perform various operations on their data. These include:
  - **Aggregate Functions:** COUNT(), SUM(), AVG(), MAX(), MIN(), etc., for summarising data.
  - **String Functions:** CONCAT(), LOWER(), UPPER(), REGEXP\_EXTRACT(), etc., for working with text data.
  - **Mathematical Functions:** ROUND(), FLOOR(), CEIL(), ABS(), etc., for numerical operations.
  - **Date and Time Functions:** CURRENT\_DATE(), DATE\_DIFF(), EXTRACT(), etc., for manipulating dates and timestamps.

# Functions and Queries

## 3. User-Defined Functions (UDFs):

- BigQuery allows users to write custom functions to handle specific tasks that are not covered by built-in functions. These are User-Defined Functions (UDFs), which can be written in JavaScript or SQL.

## 4. Subqueries and Nested Queries:

- BigQuery supports subqueries (queries within queries) to break down complex problems. Subqueries can be used in the SELECT, FROM, WHERE, and HAVING clauses.

## 5. Window Functions:

- BigQuery also supports window functions, which perform calculations across a set of rows related to the current row, without collapsing them into a single result.

## 6. Join Operations:

- BigQuery supports different types of JOINS: INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, etc., allowing users to combine multiple tables based on common columns.

# Functions and Queries

## 7. Query Optimisation:

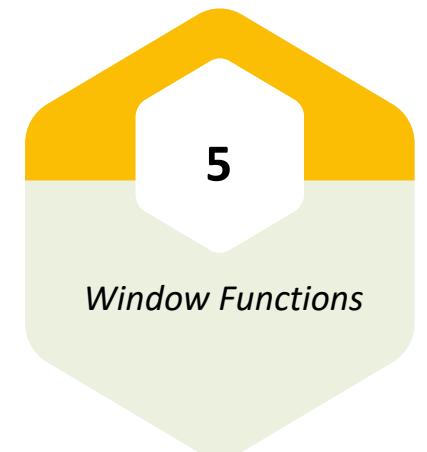
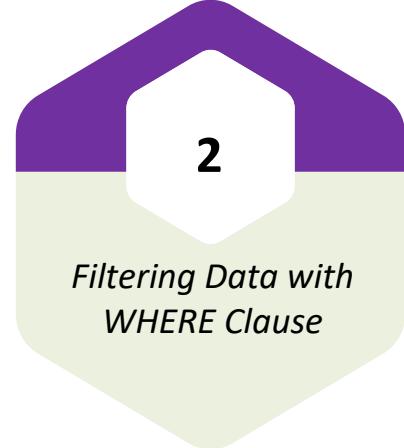
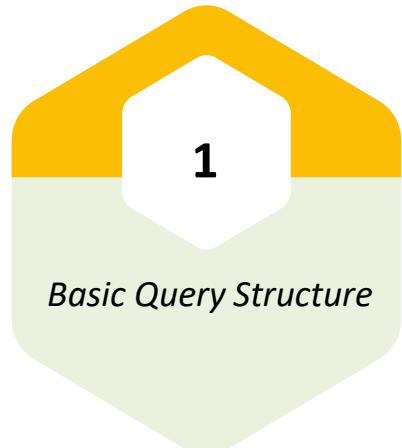
- BigQuery automatically optimises queries for performance, but users can also leverage best practices like partitioning, clustering, and denormalisation to improve query speed and reduce costs.
- Partitioning splits large tables into smaller, manageable chunks, while clustering organises data within partitions for faster querying.

## 8. Handling Large Datasets:

- BigQuery is designed for high-performance querying of massive datasets, and it allows users to run queries on petabytes of data.
- The system automatically handles the distribution of data across multiple nodes, providing parallel processing capabilities to enhance query performance.

# Writing Queries in BigQuery

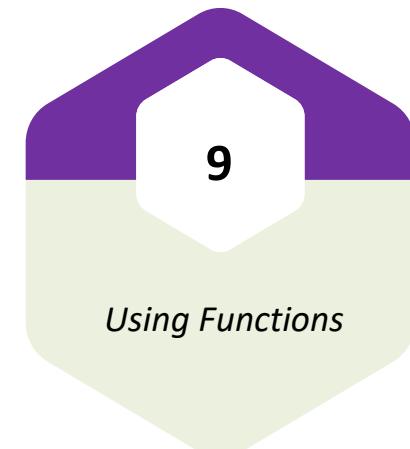
- ✓ Writing queries in BigQuery allows users to interact with their data efficiently using SQL-like syntax. BigQuery supports powerful querying capabilities, including filtering, aggregating, and transforming large datasets with minimal latency.
- ✓ By using optimised query strategies, you can process petabytes of data in seconds, making it a valuable tool for big data analytics and real-time decision-making.
- ✓ ***The following are the key aspects of writing queries in BigQuery:***



# Writing Queries in BigQuery

(continued)

- ✓ *The following are the key aspects of writing queries in BigQuery:*



# Writing Queries in BigQuery

(continued)

- ✓ *The following are the key aspects of writing queries in bigquery:*

## 1. Basic Query Structure:

- A basic query in BigQuery follows the standard SQL format with a SELECT statement. You can specify columns to retrieve from your dataset, filter data, and apply sorting or grouping operations.

```
SELECT name, age  
FROM `my_project.my_dataset.my_table`  
WHERE age > 25  
ORDER BY age DESC;
```

# Writing Queries in BigQuery

## 2. Filtering Data with WHERE Clause:

- The WHERE clause filters data based on conditions. It supports logical operators like AND, OR, NOT, and comparison operators like =, <, >, <=, >=, BETWEEN, IN, and LIKE.

## 3. Aggregating Data with GROUP BY:

- The GROUP BY clause allows you to group rows that have the same values in specified columns and apply aggregate functions (e.g., COUNT(), SUM(), AVG()).

```
SELECT name, city  
FROM `my_project.my_dataset.users`  
WHERE city = 'New York' AND age >= 30;
```

```
SELECT city, AVG(age) AS avg_age  
FROM `my_project.my_dataset.users`  
GROUP BY city;
```

# Writing Queries in BigQuery

## 4. Join Operations:

- You can use JOIN to combine data from two or more tables based on common columns. BigQuery supports different types of joins such as INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN.

## 5. Window Functions:

- Window functions allow you to perform calculations across a set of rows related to the current row. These functions do not reduce the number of rows returned, unlike aggregate functions.

```
SELECT u.name, s.salary  
FROM `my_project.my_dataset.users` u  
INNER JOIN  
`my_project.my_dataset.salaries` s  
ON u.user_id = s.user_id;
```

```
SELECT name, age,  
 RANK() OVER (PARTITION BY city  
 ORDER BY age DESC) AS age_rank  
FROM `my_project.my_dataset.users`;
```

# Writing Queries in BigQuery

## 6. Subqueries:

- Subqueries are queries embedded within other queries. You can use them in the SELECT, FROM, or WHERE clauses.

## 7. Handling Nested and Repeated Data:

- BigQuery supports nested and repeated fields, which are useful for storing complex data types like JSON arrays or objects. You can query nested data using STRUCT and ARRAY functions.

```
SELECT name, age  
FROM `my_project.my_dataset.users`  
WHERE age > (SELECT AVG(age) FROM  
`my_project.my_dataset.users`);
```

```
SELECT name, hobbies[OFFSET(0)] AS  
first_hobby  
FROM `my_project.my_dataset.users`;
```

# Writing Queries in BigQuery

## 8. Optimising Queries:

- BigQuery automatically optimises queries for performance, but you can enhance query efficiency by:
  - **Partitioning tables:** Splitting large datasets into smaller, more manageable partitions.
  - **Clustering tables:** Organising data within partitions to speed up query execution.
  - **Limiting data:** Use the LIMIT clause to restrict the number of rows returned, improving query speed.

## 9. Using Functions:

- BigQuery provides a variety of built-in functions for text manipulation (CONCAT, LOWER), mathematical calculations (ROUND, ABS), and date and time operations (CURRENT\_DATE, DATE\_DIFF).

```
SELECT name, CONCAT('Hello, ', name) AS greeting  
FROM `my_project.my_dataset.users`;
```

# Writing Queries in BigQuery

## 10. Writing Queries with BigQuery ML:

- BigQuery ML allows you to create and execute machine learning models directly within BigQuery using SQL-like syntax. You can write queries to build and train models, such as linear regression or k-means clustering.

```
CREATE OR REPLACE MODEL  
`my_project.my_dataset.linear_model`  
OPTIONS(model_type='linear_reg') AS  
SELECT feature1, feature2, label  
FROM `my_project.my_dataset.training_data`;
```

# Module 6: Serverless, Autoscaling Data Pipelines with Dataflow

- Beam Programming Model Overview
- Data Pipelines
- Writing a Dataflow Pipeline
- Scalable Big Data Processing Using Beam
- MapReduce in Dataflow



# Module 6: Serverless, Autoscaling Data Pipelines with Dataflow

- Phases in MapReduce in Dataflow
- Steps of Dataflow
- Incorporating Additional Data
- Side Inputs in Apache Beam
- Querying Multiple Tables



# Beam Programming Model Overview

- ✓ The Beam Programming Model simplifies the development of data processing pipelines for both batch and streaming data.
- ✓ Apache Beam abstracts execution engines, allowing users to write a pipeline once and run it on different backends like Google Cloud Dataflow, Apache Flink, and Apache Spark.
- ✓ It supports various data processing tasks, including transformation, enrichment, aggregation, and windowing.
- ✓ This makes Beam a powerful tool for creating scalable and distributed data workflows.
- ✓ With Beam, developers can seamlessly process large datasets in real-time or batch modes without worrying about the underlying infrastructure.



# Data Pipelines

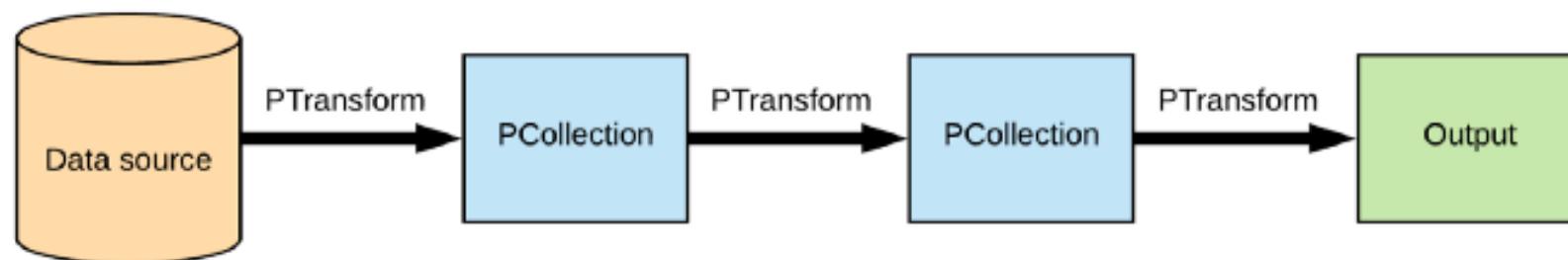
- ✓ A data pipeline refers to a series of data processing steps that ingest, transform, and output data. In the context of Google Cloud Dataflow, a data pipeline is a workflow built using Apache Beam that processes data in either batch or streaming mode.
- ✓ Data pipelines allow organisations to automate data workflows, providing a robust method to move, process, and analyse large datasets efficiently and reliably. Google Cloud Dataflow automates the scaling of resources, making it easy to handle big data workloads while minimising management overhead.
- ✓ ***To create a pipeline, instantiate the pipeline object, eventually pass some options and declaring the steps/transforms of the pipeline.***

```
import apache_beam as beam
from apache_beam.options.pipeline_options import
PipelineOptions

options = PipelineOptions()
p = beam.Pipeline(options=options)
```

# Writing a Dataflow Pipeline

- ✓ Writing a Dataflow pipeline in Google Cloud Dataflow involves using the Apache Beam programming model, which allows you to create both batch and stream data processing pipelines.
- ✓ These pipelines consist of a series of steps (called transforms) that process data. You can write a Dataflow pipeline in Python or Java, and it will run on Google Cloud Dataflow, a fully managed service for running Apache Beam pipelines at scale.
- ✓ ***The following is the data flow of the pipeline:***



# Writing a Dataflow Pipeline

## 1. Data Source (Input):

- Data source refers to where the data originates. This can be any external storage or database, such as Google Cloud Storage, BigQuery, Kafka, or any other supported data source. The data is ingested into the pipeline for processing.

## 2. PTransform (Transformations):

- PTransform stands for Parallel Transform in Apache Beam, which defines the operations applied to the data. A PTransform takes an input PCollection (a distributed dataset) and performs some operation (such as filtering, mapping, aggregating, etc.) on the data.
- In the diagram, there are multiple PTransform steps applied in sequence, where each transformation processes the data and outputs another PCollection.

# Writing a Dataflow Pipeline

## 3. PCollection (Data Container):

- PCollection is the fundamental abstraction in Apache Beam. It represents a distributed dataset. Data flowing through the pipeline is contained within PCollections, which are processed in parallel across different nodes or machines.
- In the image, each PCollection receives transformed data and can be passed to the next transform. This ensures that data is processed at scale across different resources.

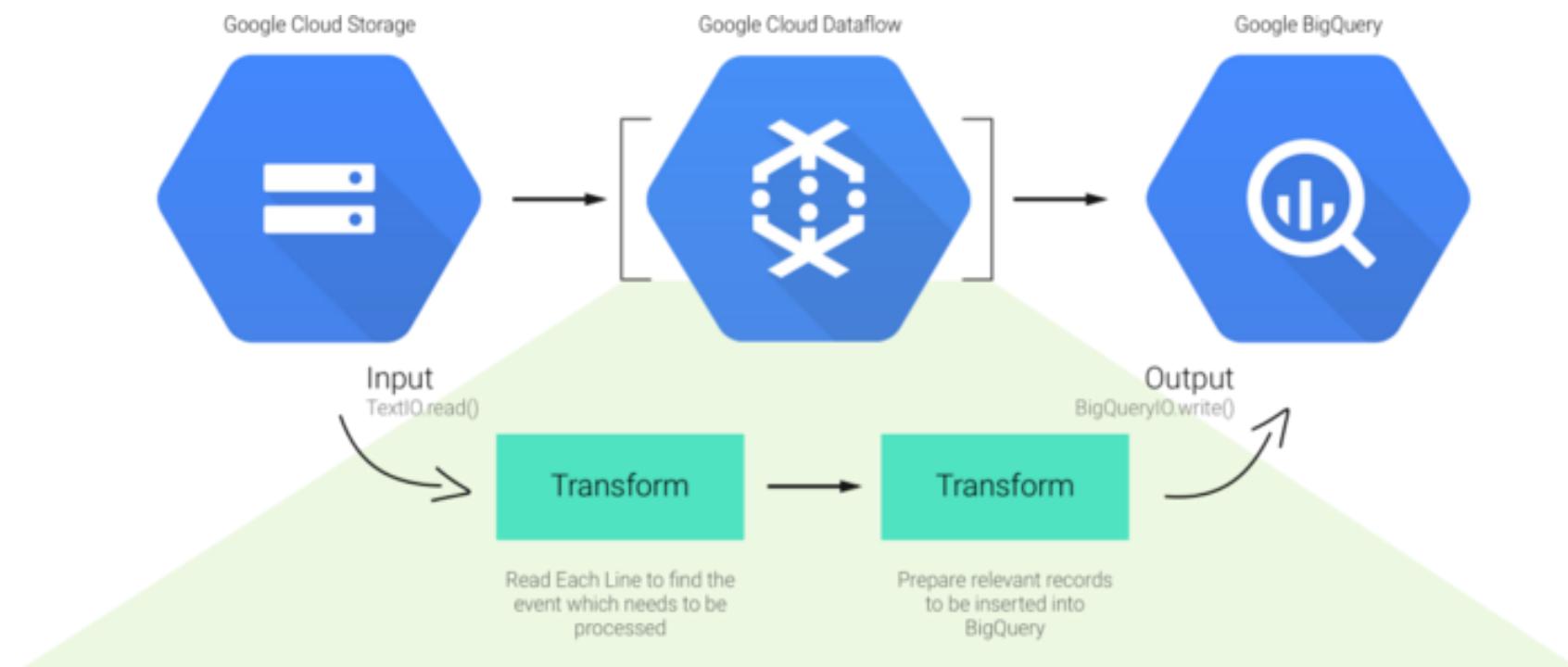
## 4. Output:

- The Output represents the final result of the data pipeline. After all the transformations are applied, the resulting data is written to an output sink. This could be a database, cloud storage, or another system.
- The output is the final step of the pipeline where the processed data is stored or made available for further use, such as in Google Cloud Storage or BigQuery.

# Writing a Dataflow Pipeline

(Continued)

- ✓ *The following is the basic flow of the pipeline:*



# Writing a Dataflow Pipeline

## 1. Data Source (Text File):

- The data source in this pipeline is a text file (shown as a Google Cloud Storage bucket in the diagram). This could be any file stored in Google Cloud Storage (GCS) or another compatible source.
- `Text.read()` is used to read the content of the text file line by line. Each line represents a data record that will be processed.

## 2. Transform (First Transformation Step):

- Transform 1 represents the first step in processing the data. In this case, the data is read line by line.
- Each line is then processed to identify the event that needs to be handled. This might involve parsing or filtering data to isolate specific events (for example, logs, customer transactions, etc.) that require further processing.

# Writing a Dataflow Pipeline

## 3. Apache Beam Pipeline:

- After reading the data, the pipeline applies transforms (operations or processing steps) to the data to clean, filter, or manipulate it. In this case:
  - The first transform reads each line and looks for specific data or events that need to be processed.
  - The second transform prepares the relevant records and formats them for insertion into BigQuery.

## 4. Output (BigQuery):

- After processing the data through the transforms, the BigQuery component is used to load the transformed data into a BigQuery table. This final output step ensures that the processed data is available for analytics or reporting.
- `BigQueryIO.write()` is used to insert data into BigQuery from the pipeline. This allows the processed data to be queried and analysed using SQL-based analytics in BigQuery.

# Scalable Big Data Processing Using Beam

- ✓ Apache Beam is a powerful, unified model for building scalable and flexible data processing pipelines. It enables both batch and streaming data processing in a distributed and parallelised manner, which is essential for handling big data at scale.
- ✓ Beam abstracts the complexities of managing infrastructure, allowing you to focus on building and managing data processing workflows.
- ✓ It supports various execution engines (runners) like Google Cloud Dataflow, Apache Spark, and Apache Flink, enabling scalable and efficient big data processing in a cloud environment.
- ✓ *The following are the key stages in the end-to-end data pipeline process:*



# Scalable Big Data Processing Using Beam

- 1. Capture (Data Ingestion at Any Scale):** Capture refers to the process of ingesting data from various sources (e.g., databases, APIs, IoT devices). This step is crucial for collecting data at any scale, whether from real-time streams or batch processes.
- 2. Process (Reliable Streaming Data Pipeline):** Process involves transforming and cleaning the raw data. In a streaming pipeline, data is processed in real-time as it is captured, enabling continuous data flow and transformation before storage.
- 3. Store (Data Lake and Data Warehousing):** Store focuses on saving data in data lakes (unstructured storage) or data warehouses (structured storage). This step ensures data is efficiently stored for future retrieval and analysis.
- 4. Analyse (Data Warehousing):** Analyse refers to the process of querying and examining the stored data in a data warehouse. It involves using SQL and other querying tools to extract meaningful insights from the data.
- 5. Use (Advanced Analytics):** Use represents the final step, where advanced analytics (e.g., machine learning, AI, and predictive modeling) are applied to the analysed data to generate actionable insights and drive decision-making.

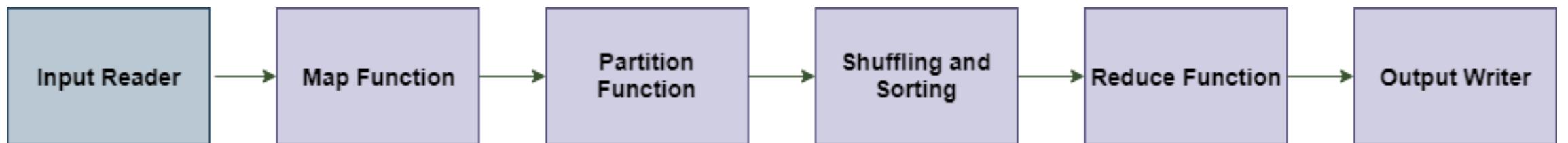
# MapReduce in Dataflow

- ✓ MapReduce is a powerful data processing model that handles large-scale datasets by dividing tasks into two phases: Map and Reduce.
- ✓ The Map phase processes data in parallel, while the Reduce phase aggregates the results. In Google Cloud Dataflow, MapReduce enables efficient parallel processing of large datasets at scale.
- ✓ Dataflow, built on the Apache Beam programming model, provides a distributed, serverless environment for executing MapReduce operations.
- ✓ This allows users to perform data processing without needing to manage underlying infrastructure, simplifying scalability and reducing management overhead.



# Phases in MapReduce in Dataflow

- ✓ In Google Cloud Dataflow, which is based on the Apache Beam programming model, MapReduce operations are broken down into several key phases: Map, Shuffle, and Reduce.
- ✓ These phases are crucial for efficiently processing large datasets in parallel and distributed environments.
- ✓ The entire process allows Dataflow to handle both batch and streaming data, enabling real-time analytics and large-scale data processing.
- ✓ ***The following are the key phases in MapReduce in dataflow:***



# Phases in MapReduce in Dataflow

- 1. Input Reader:** The input reader reads the upcoming data and splits it into the data blocks of the appropriate size (64 MB to 128 MB). Each data block is associated with a Map function. Once input reads the data, it generates the corresponding key-value pairs. The input files reside in HDFS.
- 2. Map Reduce:** The map function process the upcoming key-value pairs and generated the corresponding output key-value pairs. The map input and output type may be different from each other.
- 3. Partition Function:** The partition function assigns the output of each Map function to the appropriate reducer. The available key and value provide this function. It returns the index of reducers



# Phases in MapReduce in Dataflow

4. **Shuffling and Sorting:** The data are shuffled between/within nodes so that it moves out from the map and get ready to process for reduce function. Sometimes, the shuffling of data can take much computation time.
  - The sorting operation is performed on input data for Reduce function. Here, the data is compared using comparison function and arranged in a sorted form
5. **Reduce Function:** The Reduce function is assigned to each unique key. These keys are already arranged in sorted order. The values associated with the keys can iterate the Reduce and generates the corresponding output
6. **Output Writer:** Once the data flow from all the above phases, Output writer executes. The role of Output writer is to write the Reduce output to the stable storage.



# Steps of Dataflow

- ✓ Google Cloud Dataflow is a fully managed, serverless data processing service that allows users to build and execute data processing pipelines. These pipelines can handle both batch and streaming data efficiently.
- ✓ The process of building and running a Dataflow pipeline involves several key steps that transform raw data into valuable insights. Here's an overview of the key steps involved in a Dataflow pipeline.
- ✓ ***The following are the key steps of dataflow:***
  1. **Define the Pipeline:**
    - The first step is to define the pipeline using the Apache Beam SDK. A pipeline in Dataflow consists of a series of transforms (operations) applied to the data. The pipeline is designed to specify how data should be read, processed, and written.
    - Example: In Python, you create the pipeline with `beam.Pipeline()` and define the various transforms like `ParDo`, `GroupByKey`, or `Map`.

# Steps of Dataflow

## 2. Input Data Collection (Ingestion):

- In the data ingestion step, the pipeline reads data from a source. Data can come from various sources such as Google Cloud Storage, BigQuery, Cloud Pub/Sub, or even external systems (e.g., databases, APIs).
- Example: Using `beam.io.ReadFromText()` to read data from Cloud Storage or `beam.io.ReadFromBigQuery()` to query and read data from BigQuery.

## 3. Data Transformation:

- Once the data is ingested, the next step is data transformation. This is where you apply various transforms to process and manipulate the data. Examples of common transforms include:
  - **ParDo**: For applying a user-defined function to each element.
  - **GroupByKey**: To group data by key for aggregation.
  - **Windowing**: To manage unbounded (streaming) data and perform time-based processing.
- These transformations help in cleaning, filtering, aggregating, or enriching the data as needed.

# Steps of Dataflow

## 4. Windowing and Triggers (for Streaming Data):

- Windowing is an essential step for processing streaming data in Dataflow. It divides the continuous stream into fixed, sliding, or session-based windows, allowing you to process the data in chunks based on time.
- Triggers control when the results for a window are emitted. They are particularly useful for handling late-arriving data in real-time processing.
- Example: `beam.WindowInto(beam.window.FixedWindows(10))` to define a fixed time window for stream processing.

## 5. Aggregation and Grouping:

- After the transformation phase, data may need to be aggregated or grouped. You can use `GroupByKey`, `Combine`, or `Reduce` operations to aggregate data across different keys.
- Example: Summing values, finding averages, or calculating maximum/minimum over a dataset.

# Steps of Dataflow

## 6. Output Data (Sink):

- After processing the data, the next step is to write the transformed data to an output location (sink). Dataflow supports various output formats and destinations such as: Google Cloud Storage (for saving files), BigQuery (for structured data storage), Cloud Pub/Sub (for messaging systems), Datastore (for NoSQL databases).
- Example: Using `beam.io.WriteToBigQuery()` to write results into a BigQuery table or `beam.io.WriteString()` to write data to a text file in Cloud Storage.

## 7. Execute the Pipeline:

- After defining the pipeline with all necessary transforms and outputs, you execute it using the `run()` method. This triggers the pipeline to start processing the data.
- Example: `p.run().wait_until_finish()` will run the pipeline and ensure it completes successfully.

# Steps of Dataflow

## 8. Monitor and Debug the Pipeline:

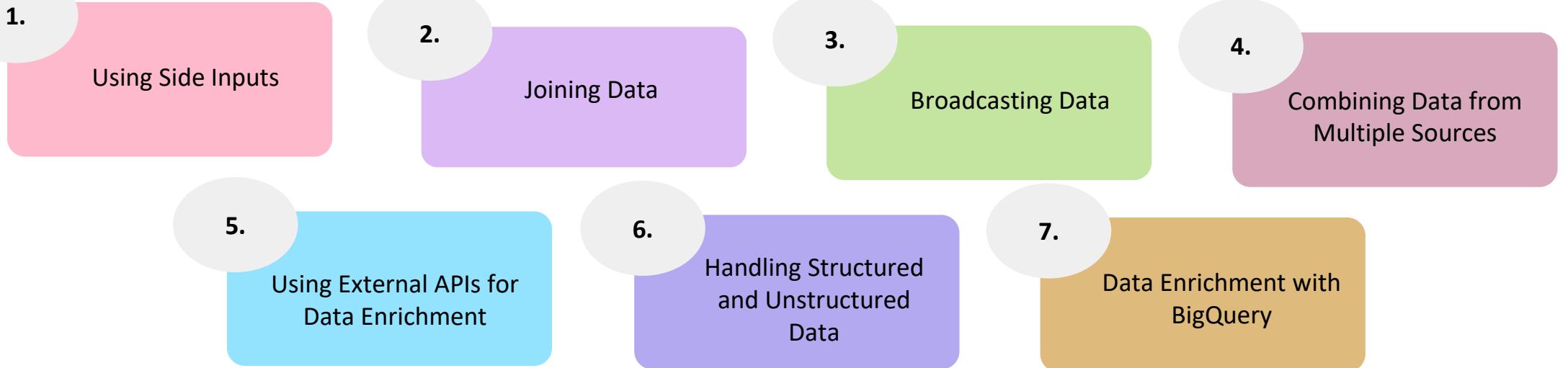
- Once the pipeline is running, you can monitor its progress in the Google Cloud Console. Dataflow provides detailed metrics and logs about the execution, which helps identify performance bottlenecks or errors.
- The Dataflow Monitoring UI provides insights into the pipeline's execution, with real-time data about running stages, resource usage, and completion status.

## 9. Fault Tolerance and Recovery:

- Fault tolerance is a key feature in Dataflow. The system automatically recovers from errors and retries failed tasks, ensuring data is processed reliably without loss.
- If failures occur during the pipeline run, Dataflow ensures that tasks are rescheduled, and the pipeline continues processing without interruption.

# Incorporating Additional Data

- ✓ Incorporating additional data into a Google Cloud Dataflow pipeline is a common requirement when processing complex datasets. Data often needs to be enriched by merging it with other datasets, applying external data sources, or augmenting it with additional information.
- ✓ Dataflow, using the Apache Beam programming model, provides several ways to join or combine data from multiple sources. This allows for building more sophisticated data pipelines that can process multiple inputs and produce rich, aggregated outputs.
- ✓ ***The following are the key aspects of incorporating additional data in dataflow:***



# Incorporating Additional Data

## 1. Using Side Inputs:

- Side inputs allow you to inject additional data (e.g., reference data or configuration data) into the pipeline. This data is typically small, static, and needs to be available throughout the pipeline.
- Side inputs are useful for enriching data or applying lookups.

## 2. Joining Data (Combining Multiple Sources):

- In many data processing tasks, you need to join data from multiple sources. This can be done using GroupByKey, CoGroupByKey, or other join operations in Apache Beam.
- CoGroupByKey is used to join data from two or more datasets that are grouped by the same key, allowing you to combine them based on matching keys.

# Incorporating Additional Data

## 3. Broadcasting Data:

- For large data sources, it is often not feasible to send the entire dataset as a side input. Instead, you can use broadcasting to send the additional data to each worker node without having to replicate it multiple times.
- Broadcasting is useful when you want to share the same reference data across all workers in the pipeline, such as a static lookup table.
- Apache Beam provides efficient ways to broadcast side inputs to workers by serializing them and ensuring only the required data is sent.

## 4. Combining Data from Multiple Sources:

- When working with large datasets, you may need to merge data from multiple sources. This can be achieved using transforms such as Flatten to merge multiple PCollections into one.

# Incorporating Additional Data

## 5. Using External APIs for Data Enrichment:

- Dataflow pipelines can incorporate additional data by calling external APIs or services within the pipeline to enrich the data as it flows through the pipeline.

## 6. Handling Structured and Unstructured Data:

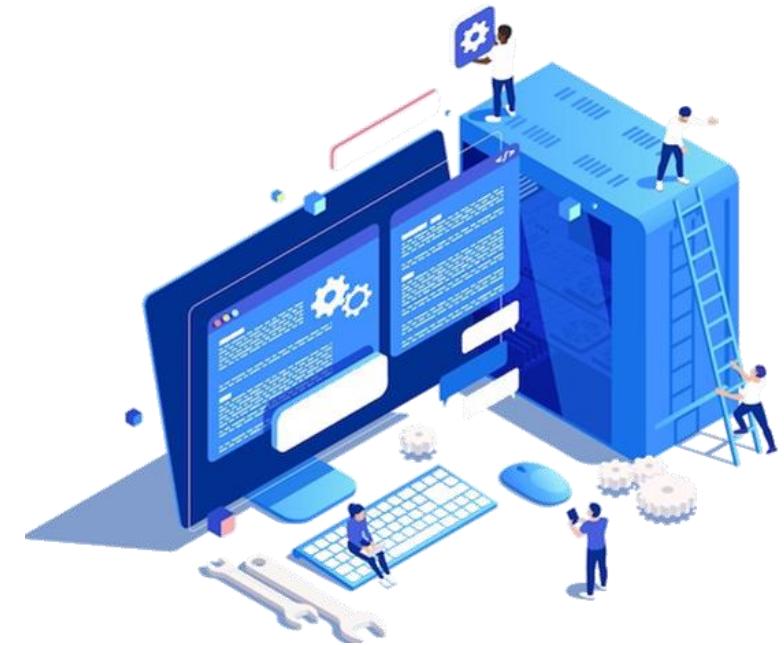
- You can incorporate both structured and unstructured data into the pipeline. Apache Beam supports reading and writing various formats, such as JSON, Avro, Parquet, and CSV. This flexibility allows for combining data from both structured databases and unstructured sources like log files.

## 7. Data Enrichment with BigQuery:

- You can also enrich your data by querying BigQuery within a Dataflow pipeline. For example, you might join real-time data with historical data stored in BigQuery to provide more context or analysis.

# Side Inputs in Apache Beam

- ✓ Side inputs are a powerful feature in Apache Beam that allow additional, small datasets to be injected into the main data processing pipeline.
- ✓ These inputs are typically used to provide lookup tables, configuration data, or other reference data that enrich the main data stream.
- ✓ Side inputs are useful when you need to use static or small data sets during the processing of large, dynamic datasets in a pipeline.
- ✓ Unlike the main data, which is typically large and processed in parallel across multiple workers, side inputs are usually small and do not require parallel processing.
- ✓ They are broadcast to all workers, ensuring that each worker can access the data during the execution of the pipeline.



# Side Inputs in Apache Beam

(Continued)

✓ ***The following are the key types of side inputs:***

1. **Iterable Side Inputs:** These are side inputs that can be iterated through. They are typically small datasets (e.g., lookup tables) that are broadcast to each worker. You can pass these as an Iterable and access them inside the DoFn.
2. **Value Side Inputs:** In some cases, you may only need a single value (e.g., a configuration parameter) for each element in the main data. These can be passed as a single value side input.

✓ ***The following are the use cases of side inputs:***

1. **Lookup Data:** Side inputs are often used for lookup tables where you need to enrich or filter the main data using small reference data.
2. **Configurations:** You can pass configurations, thresholds, or parameters to the pipeline that need to be accessed by all workers.

# Querying Multiple Tables

- ✓ In Google Cloud Dataflow, which is built on Apache Beam, querying multiple tables is a common requirement when combining data from different sources for analysis or transformation. Dataflow supports reading from multiple sources and performing operations like joins or merging of datasets to get a unified view.
- ✓ When working with data from multiple tables, you can efficiently apply transformations like joins, lookups, and aggregation to combine and process the data within the pipeline.
- ✓ ***The following are the key aspects of querying multiple tables in dataflow:***

## 1. Reading Data from Multiple Sources:

- Dataflow allows you to read data from multiple tables, whether stored in BigQuery, Google Cloud Storage, or other supported sources (like SQL databases or Apache Kafka).
- You can use the appropriate I/O transforms to read data from different tables.

# Querying Multiple Tables

## 2. Performing Joins Between Tables:

- A common use case for querying multiple tables is performing joins. You can use GroupByKey or CoGroupByKey in Apache Beam to join data from two different tables based on a shared key (e.g., user\_id, product\_id).

## 3. Combining Data from Multiple Tables:

- You can use Flatten to combine multiple datasets into one. This is useful if you want to merge data from multiple tables into a single collection for further processing.

## 4. Aggregating Data Across Tables:

- After joining or merging data from multiple tables, you may want to perform aggregation operations like sum, average, or count. You can use GroupByKey, Combine, or Reduce to perform these operations across the combined data.

# Querying Multiple Tables

## 5. Handling Unstructured and Structured Data:

- Dataflow can query both structured (e.g., BigQuery tables) and semi-structured (e.g., JSON, CSV files) data from multiple tables. This flexibility enables the combination of different types of data sources within the pipeline.
- You can apply transformations to convert unstructured data into a structured format (e.g., parsing JSON or CSV data) before performing the join or aggregation operations.

## 6. Handling Schema Differences Between Tables:

- When querying multiple tables, it's important to ensure that the schemas are compatible, especially if the tables have different structures.
- Dataflow allows for schema mapping and transformations to align the data formats before performing operations like joins.

# Querying Multiple Tables

## 7. Query Optimization for Multiple Tables:

- When querying multiple tables, it's essential to optimize the pipeline for performance.
- For example, you can filter data early in the pipeline to reduce the amount of data that is joined or aggregated. You can also partition or cluster the tables in BigQuery to optimize query performance.

## 8. Writing the Results to a Destination:

- After querying and processing the data from multiple tables, you can write the results to an output sink, such as BigQuery, Google Cloud Storage, or Cloud Pub/Sub.



# Module 7: Getting Started with Machine Learning

- What is Machine Learning (ML)?
- Types of Machine Learning
- Explore and Create ML Datasets



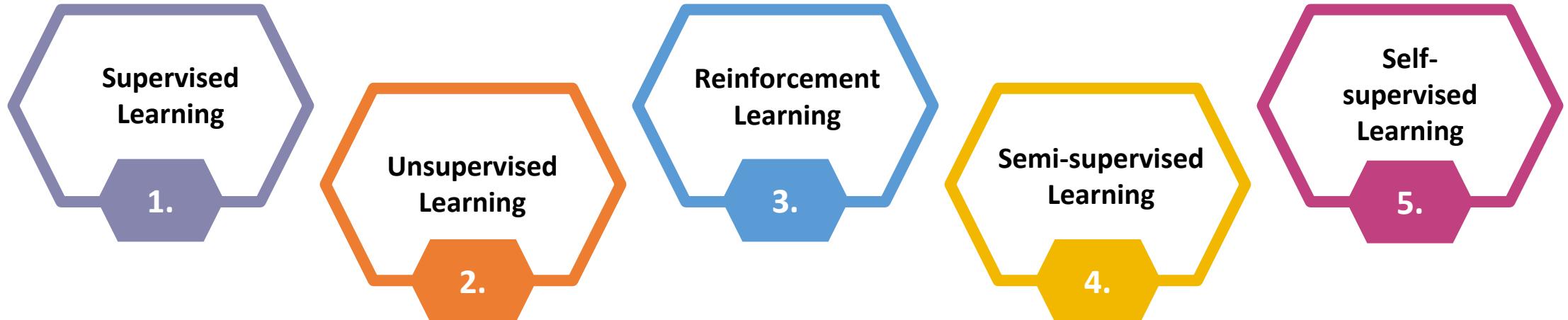
# What is Machine Learning (ML)?

- ✓ Machine Learning (ML) is a subset of artificial intelligence (AI) that involves the development of algorithms and models that enable computers to learn from and make predictions or decisions based on data, without being explicitly programmed for each task.
- ✓ The goal of machine learning is to enable systems to automatically improve their performance by learning from experience, thereby making them more efficient over time.
- ✓ ML algorithms are designed to detect patterns in data, allowing them to adapt to new inputs. This flexibility is essential for handling tasks like image recognition, natural language processing, and real-time decision-making.
- ✓ As more data is collected, ML models can be trained to become more accurate, enabling systems to make increasingly sophisticated predictions and decisions.



# Types of Machine Learning

- ✓ Supervised learning is the most commonly used type of machine learning. It involves training a model on labeled data, where both the input and the correct output are known. The model learns by making predictions on the input data and adjusting its internal parameters based on the error in its predictions, typically using a loss function.
- ✓ *The following are the types of machine learning:*



# Types of Machine Learning

## 1. Supervised Learning:

- Supervised learning is the most commonly used type of machine learning. It involves training a model on labelled data, where both the input and the correct output are known.
- The model learns by making predictions on the input data and adjusting its internal parameters based on the error in its predictions, typically using a loss function.
  - **Use Case:** Spam email detection, where the model learns from labeled examples (spam and non-spam) to classify new emails.
  - **Algorithms:** Linear regression, Logistic regression, Decision trees, Support vector machines (SVM), Neural networks.
- **Key Points:**
  1. The model is provided with labeled data. The goal is to predict the output for new, unseen data

# Types of Machine Learning

## 2. Unsupervised Learning:

- Unsupervised learning deals with data that does not have labels. The algorithm attempts to learn the structure or patterns from the input data without guidance on the expected output. It is primarily used for tasks like clustering, anomaly detection, and association.
  - **Use Case:** Customer segmentation, where a model groups customers based on purchasing behavior without knowing predefined categories.
  - **Algorithms:** K-means clustering, Hierarchical clustering, DBSCAN, Principal Component Analysis (PCA).
- **Key Points:**
  1. The data is unlabeled.
  2. The goal is to uncover hidden patterns or structures in the data.

# Types of Machine Learning

### 3. Reinforcement Learning:

- Reinforcement learning (RL) involves training models to make decisions through a system of rewards and penalties. The model (agent) interacts with an environment and learns by performing actions that maximise cumulative rewards. It is widely used in robotics, gaming, and decision-making tasks.
  - **Use Case:** Training self-driving cars, where the car (agent) learns to navigate traffic by receiving feedback from its actions (rewards for safe driving, penalties for collisions).
  - **Algorithms:** Q-learning, Deep Q Networks (DQN), Policy Gradient methods, Actor-Critic methods.
- **Key Points:**
  1. The model learns by interacting with an environment.
  2. The goal is to maximise the long-term reward through trial and error.

# Types of Machine Learning

## 4. Semi-supervised Learning:

- This type of learning lies between supervised and unsupervised learning. It uses both labeled and unlabeled data for training. Typically, the amount of labeled data is small, and the goal is to make use of a large amount of unlabeled data to improve model performance.
  - **Use Case:** Image classification where only a few images are labeled, and the rest are unlabeled. The model learns from both labeled and unlabeled data.
  - **Algorithms:** Semi-supervised support vector machines, Self-training algorithms.
- **Key Points:**
  1. Combines labeled and unlabeled data.
  2. Often used when labeling data is expensive or time-consuming.

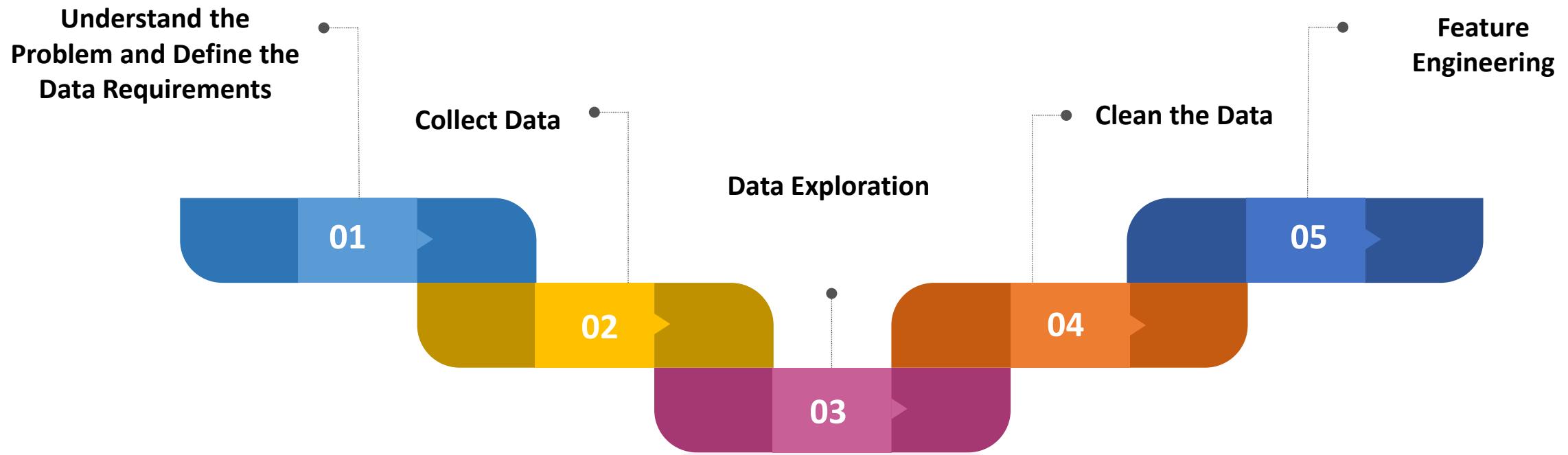
# Types of Machine Learning

## 5. Self-supervised Learning:

- A subcategory of unsupervised learning, self-supervised learning allows the model to generate labels from the data itself. The model learns by predicting parts of the input data from other parts, effectively creating its own "labels" without human intervention.
  - **Use Case:** Natural language processing (NLP) models like BERT, where the model learns from unstructured text data by predicting missing words based on surrounding context.
  - **Algorithms:** Contrastive learning, Masked Language Models (e.g., BERT).
- **Key Points:**
  1. The model generates its own labels from the input data.
  2. It is a way to leverage large amounts of unlabeled data for learning.

# Explore and Create ML Datasets

- ✓ Datasets are the backbone of Machine Learning (ML). To build robust models, it is essential to have high-quality datasets that accurately represent the problem you're trying to solve. Exploring and creating ML datasets involves understanding the data, processing it, and preparing it for model training.
- ✓ *The following are the key steps to explore and create ML datasets:*



# Explore and Create ML Datasets

## 1. Understand the Problem and Define the Data Requirements:

- **Define the Goal:** Understand the problem you want to solve (e.g., classification, regression, clustering) and determine the type of data needed.
- **Identify Features and Labels:** Features are the input variables, and labels are the output variables (in supervised learning). For unsupervised learning, the focus will be on the patterns within the features.

## 2. Collect Data:

- **Data Sources:** Data can be sourced from public repositories (e.g., Kaggle, UCI Machine Learning Repository), company databases, APIs, web scraping, or sensors.
- **Data Types:** Identify the types of data needed for the problem (e.g., structured data, unstructured data like text or images, time-series data).

# Explore and Create ML Datasets

## 3. Data Exploration:

- **Examine the Structure:** Explore the dataset's structure by inspecting columns, rows, data types, and sample values. This can be done using pandas (Python) or other data manipulation libraries.
- **Summary Statistics:** Use descriptive statistics like mean, median, variance, and distribution of values to understand the dataset better.
- **Visualisation:** Visualise the data using charts (histograms, scatter plots, box plots) to identify trends, patterns, and outliers.

## 4. Clean the Data:

- **Handle Missing Values:** Fill missing values with mean/median values, or drop rows/columns with too many missing values, depending on the dataset's importance.

# Explore and Create ML Datasets

(Continued)

- **Remove Duplicates:** Check and remove any duplicate rows that might affect model accuracy.
- **Data Transformation:** Normalise or standardise features, especially when the dataset contains variables with different units of measurement.

## 5. Feature Engineering:

- **Create New Features:** Combine or derive new features from existing ones that can improve the model's performance (e.g., combining "height" and "weight" to create a "BMI" feature).
- **Encoding Categorical Data:** Convert categorical features (e.g., country, gender) into numerical values using techniques like one-hot encoding or label encoding.
- **Feature Selection:** Select the most relevant features based on correlation analysis or domain knowledge to reduce noise and improve model efficiency.

# Module 8: Building ML Models with TensorFlow

- Understand TensorFlow
- TensorFlow Graphs and Loops
- Use Low-Level TensorFlow
- Monitoring ML Training
- Charts and Graphs of TensorFlow



# Understand TensorFlow

- ✓ TensorFlow is an open-source machine learning framework developed by Google, designed to support the creation, training, and deployment of machine learning and deep learning models at scale.
- ✓ It provides a flexible ecosystem of tools, libraries, and community resources that allow developers and data scientists to build models for tasks like image recognition, natural language processing, predictive analytics, and recommendation systems.
- ✓ At its core, TensorFlow uses a computational graph approach, where data flows through nodes representing mathematical operations, making it efficient for both small-scale experiments and large-scale distributed systems.
- ✓ It supports multiple programming languages such as Python, C++, and JavaScript, but Python remains the most widely used due to its simplicity.



# Understand TensorFlow

(Continued)

✓ ***The following are the key aspects of tensorflow:***

1. **Flexible and Scalable Architecture:** TensorFlow offers a flexible architecture, allowing developers to deploy models on different platforms, including CPUs, GPUs, and TPUs. It scales well from research prototypes to large-scale production systems.
2. **Comprehensive Ecosystem:** TensorFlow provides a full ecosystem for machine learning, including tools for model building, training, deployment, and monitoring. Key components include TensorFlow Lite (for mobile and embedded devices), TensorFlow.js (for browser-based ML), and TensorFlow Extended (TFX) for production pipelines.
3. **High-Level APIs (Keras):** Keras is a high-level API in TensorFlow that simplifies the process of building deep learning models. It offers easy-to-use layers and tools, making neural networks more accessible to developers and researchers.

# Understand TensorFlow

4. **Support for Deep Learning and Machine Learning:** TensorFlow supports both deep learning (e.g., neural networks, CNNs, RNNs) and traditional machine learning techniques (e.g., linear regression, clustering). This versatility allows it to be used for a wide range of tasks like image recognition, NLP, and predictive analytics.
5. **Distributed Computing:** TensorFlow enables distributed training across multiple devices or machines, making it ideal for large datasets and resource-intensive applications. It also includes tools like TensorFlow Distributed for efficient multi-node training.
6. **Automatic Differentiation (Autograd):** TensorFlow supports automatic differentiation, allowing it to compute gradients automatically during backpropagation. This feature is crucial for training deep learning models through gradient descent.
7. **TensorFlow Hub and Model Garden:** TensorFlow Hub provides a library for reusable machine learning modules, allowing developers to easily integrate pre-trained models into their applications. The TensorFlow Model Garden offers a collection of state-of-the-art models and implementations.

# Understand TensorFlow

8. **Cross-Platform Deployment:** TensorFlow models can be deployed across a wide variety of platforms, from mobile devices (via TensorFlow Lite) to web browsers (via TensorFlow.js) and cloud-based environments, making it versatile for different use cases.
9. **Robust Community and Ecosystem:** TensorFlow has a large and active community of developers and researchers. It is supported by extensive documentation, tutorials, and resources, and integrates well with other Google Cloud services for AI development.
10. **Optimized for Performance:** TensorFlow is optimised for performance, supporting accelerated computation on GPUs and TPUs. The framework can handle large-scale data processing efficiently, making it suitable for high-performance computing environments.



# TensorFlow Graphs and Loops

- ✓ In TensorFlow, one of the most powerful features is the ability to create computational graphs. These graphs represent mathematical operations as a series of nodes (operations) and edges (data dependencies) that allow you to build machine learning models efficiently.
- ✓ Understanding graphs and loops in TensorFlow is essential for creating flexible, reusable models, especially when working with large-scale data and complex architectures.
- ✓ ***The following are the key aspects of tensorflow graphs and loops:***

## 1. TensorFlow Computational Graph:

- A computational graph is a representation of the sequence of operations that occur in the TensorFlow framework. In a graph, nodes represent operations (e.g., addition, multiplication, matrix operations) and edges represent the data flowing between these operations (tensors).
- TensorFlow 1.x required explicitly defining a graph using the `tf.Graph()` API, where all operations and variables were defined first, and then run in a session.

# TensorFlow Graphs and Loops

## 2. Tensors:

- Tensors are the fundamental building blocks in TensorFlow. They represent data in the form of arrays (multidimensional matrices). The edges of the computational graph are the tensors that flow through the graph and are processed by the nodes.

## 3. Building a Graph:

- In TensorFlow 2.x, you can define models using high-level APIs like Keras. However, if you need to optimise performance, you can convert parts of the model into a graph using `tf.function`. This allows TensorFlow to optimise the computation by executing the operations as a graph.

## 4. Loops in TensorFlow:

- Loops are used in TensorFlow for iterative operations, such as training epochs, or when applying a function repeatedly on batches of data.

# TensorFlow Graphs and Loops

(Continued)

- While TensorFlow does not use traditional loops in the Python sense, it uses `tf.while_loop` and `tf.for_loop` to create loops in the graph for iterative tasks, like training a model or generating sequences.
- `tf.while_loop` is used to create loops for control flow in TensorFlow graphs. It's used for tasks that require iteration with condition checks.
- `tf.cond` is used to evaluate conditions inside a graph and decide which part of the graph to execute.

## 5. Eager Execution vs Graph Execution:

- Eager execution is enabled by default in TensorFlow 2.x, meaning operations execute immediately, and you can inspect and debug the results in real-time. However, for performance, especially with complex models or large datasets, it's beneficial to convert the code into a graph using `tf.function`.
- Graph execution allows TensorFlow to optimise and parallelise the computation for better performance. This is especially useful when training large models or deploying to production.

# TensorFlow Graphs and Loops

## 6. Control Flow Operations:

- TensorFlow provides operations like `tf.while_loop`, `tf.for_loop`, and `tf.cond` for defining control flows within the graph.
- These are particularly useful for complex models where you need to make dynamic decisions based on data, or iterate over sequences of data efficiently.

## 7. Optimising with TensorFlow Graphs:

- Graphs in TensorFlow can be optimised for performance. TensorFlow automatically applies optimisations like operation fusion (combining operations) and memory optimisation (reducing memory footprint).
- TensorFlow also supports XLA (Accelerated Linear Algebra), a compiler that optimises graph execution for hardware acceleration, such as GPUs and TPUs.

# Use Low-Level TensorFlow

- ✓ In TensorFlow, you can build models using both high-level APIs (like Keras) and low-level APIs, which provide more control over the computation and allow you to create custom models, optimisers, and layers.
- ✓ Low-level TensorFlow gives you the flexibility to manually define and optimise the underlying operations and computations in the model, making it ideal for more complex tasks or when you need full control over the machine learning process.
- ✓ Using low-level TensorFlow means working with the TensorFlow core API to define operations, layers, models, and computations explicitly. It allows for greater customisation and optimisation but requires a deeper understanding of TensorFlow's internals and how computational graphs work.
- ✓ ***The following are the key aspects of using low-level tensorflow:***
  1. **Tensors and Operations:** Tensors are the core data structure in TensorFlow, representing multi-dimensional arrays. Operations are applied to tensors in a computational graph, where the tensors flow through various transformations.

# Use Low-Level TensorFlow

2. **Manual Graph Construction:** Low-level TensorFlow requires explicit definition of the computational graph, where each operation and tensor flow is manually set up, unlike high-level APIs like Keras that automate this process.
3. **Custom Layers and Models:** You can subclass `tf.Module` or `tf.keras.layers.Layer` in low-level TensorFlow to build custom models and layers, providing flexibility to define specific behaviors and structures.
4. **Optimisers and Custom Training Loops:** Low-level TensorFlow allows you to create custom optimisers and training loops, giving full control over backpropagation, gradient calculation, and optimisation, unlike high-level Keras APIs.
5. **Gradient Tape:** TensorFlow's `GradientTape` API helps track tensor operations and compute gradients during backpropagation, enabling fine-grained control over the optimisation process in custom training loops.
6. **Control Flow and Dynamic Computation:** Low-level TensorFlow enables dynamic computation graphs, allowing the structure of the graph to change during runtime using control flow operations like `tf.while_loop`, `tf.cond`, or `tf.function`.

# Use Low-Level TensorFlow

7. **Eager Execution vs Graph Execution:** Eager execution (default in TensorFlow 2.x) runs operations immediately for easier debugging, while graph execution (`tf.function`) optimises performance by converting functions into static computation graphs.
8. **Memory Management:** Low-level TensorFlow allows explicit control over tensor memory allocation and lifetime, optimising memory usage for large models and high-performance tasks like distributed training.
9. **Distributed Training:** TensorFlow supports distributed training across multiple devices or machines using `tf.distribute.Strategy`, enabling scaling of models on multiple GPUs or servers.



# Monitoring ML Training

- ✓ Monitoring machine learning (ML) training is essential to ensure that the model is being trained correctly, efficiently, and effectively.
- ✓ It helps in tracking key metrics, detecting issues early, and improving the model's performance. Whether you are training on a small dataset or large-scale data, monitoring the training process provides insights into the model's learning behavior, resource utilisation, and potential bottlenecks.
- ✓ Effective monitoring can prevent overfitting, underfitting, or training on incorrect data, ensuring that the model generalises well to unseen data.
- ✓ ***The following are the key aspects of monitoring ML training:***
  1. **Tracking Loss and Accuracy:** Loss and accuracy are the most common metrics tracked during model training. The loss indicates how well or poorly the model is performing, while accuracy measures how often the model's predictions match the actual values.

# Monitoring ML Training

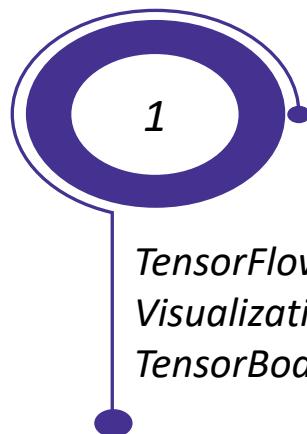
2. **Real-Time Monitoring:** Real-time monitoring allows you to track model training progress as it happens. Tools like TensorBoard or Google Cloud AI Platform enable you to visualize metrics like loss, accuracy, and even model weights or gradients during training.
3. **Training Time and Epoch Monitoring:** Monitoring training time helps you assess the efficiency of the training process. Long training times might indicate inefficient models or resource allocation.
4. **Monitoring Resource Utilisation:** During training, it is essential to monitor the hardware resources being used, such as CPU, GPU, memory, and disk space. Inefficient use of resources or resource bottlenecks can slow down training or cause crashes.
5. **Gradient and Weight Histograms:** Gradient and weight histograms help track the magnitude of updates during training. If the gradients are too large (gradient explosion) or too small (gradient vanishing), it can hinder the learning process.
6. **Overfitting and Underfitting Detection:** Overfitting occurs when the model performs well on the training data but poorly on the validation or test data, indicating the model has learned noise or specific patterns that don't generalise.

# Monitoring ML Training

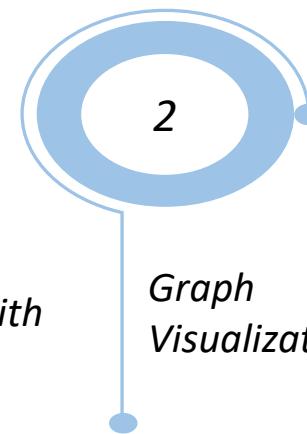
7. **Hyperparameter Tuning Monitoring:** Hyperparameters such as learning rate, batch size, and the number of layers significantly affect model performance. Monitoring and tuning these hyperparameters during training can improve model accuracy.
8. **Model Checkpoints:** Model checkpoints save the model weights at regular intervals during training, which is useful in case of interruptions (e.g., crashes or power failures). It also allows you to save and load the best-performing model based on validation metrics.
9. **Logging and Debugging:** Logging is crucial to track training progress and capture any issues or errors during training. TensorFlow, PyTorch, and other libraries allow you to log metrics, model parameters, and any exceptions during training.
10. **Post-Training Evaluation:** After training is complete, evaluate the model on a test dataset that was not seen during training or validation. This helps ensure that the model generalises well to new data.

# Charts and Graphs of TensorFlow

- ✓ In TensorFlow, charts and graphs are integral for visualizing model architecture, training progress, and performance metrics. These visual representations help data scientists and machine learning engineers better understand model behavior, debug issues, and optimise training processes.
- ✓ TensorFlow provides various tools, such as TensorBoard, to visualize computational graphs, track training metrics, and inspect internal layers of a model.
- ✓ ***The following are the key features of charts and graphs of Tensorflow:***



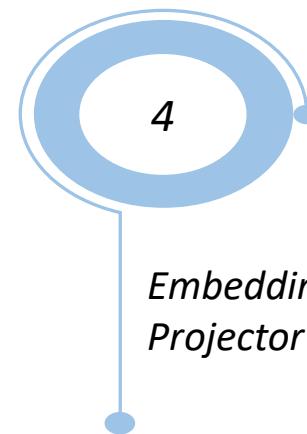
*TensorFlow  
Visualization with  
TensorBoard*



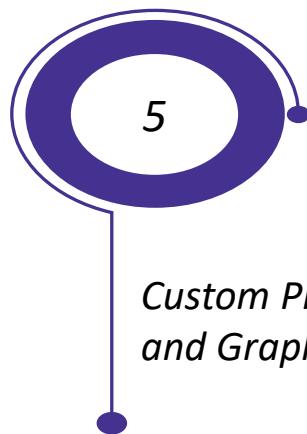
*Graph  
Visualization*



*Tracking Model  
Weights and  
Gradients*



*Embedding  
Projector*



*Custom Plotting  
and Graphs*

# Charts and Graphs of TensorFlow

## 1. TensorFlow Visualization with TensorBoard:

- TensorBoard is the primary tool for visualizing TensorFlow training metrics. It provides interactive visualizations of loss, accuracy, and other metrics, helping to understand the training process and monitor model performance.
- TensorBoard allows users to visualize scalars, histograms, and distributions of training data in real time, aiding in performance analysis and debugging.

## 2. Graph Visualization:

- TensorFlow allows you to visualize the computational graph with TensorBoard, showing how operations and data flow within the model.
- This feature is useful for understanding the structure of complex models, identifying bottlenecks, and ensuring the model's architecture is built as expected.

# Charts and Graphs of TensorFlow

## 3. Tracking Model Weights and Gradients:

- TensorFlow provides the ability to visualize weights and gradients during training to help with understanding the learning process.
- By visualizing these charts, users can detect issues like vanishing gradients or overfitting and make necessary adjustments to the model.

## 4. Embedding Projector:

- The Embedding Projector feature in TensorFlow allows the visualization of high-dimensional data, like word embeddings, by projecting them onto lower-dimensional spaces.
- This helps in exploring how different features or tokens are represented in the model, making it easier to analyse patterns or relationships in the data.

# Charts and Graphs of TensorFlow

## 5. Custom Plotting and Graphs:

- TensorFlow integrates with libraries like Matplotlib to create custom charts and graphs outside of TensorBoard, offering greater flexibility in visualizing training data and results.
- This enables users to plot detailed line graphs, scatter plots, or confusion matrices, providing deeper insights into the model's performance.
- In addition to standard plots, users can create interactive visualizations for exploring model predictions, performance over epochs, or comparing different training runs.



# Module 9: Scaling ML Models with CloudML

- Cloud ML Overview
- TensorFlow Model
- Running of ML Model



# Cloud ML Overview

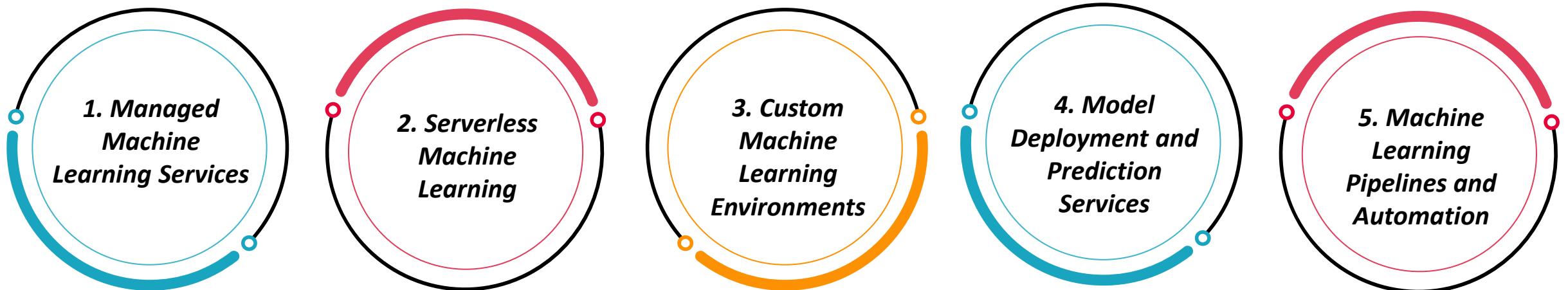
- ✓ Cloud ML refers to machine learning services and tools offered by cloud providers that enable users to develop, train, and deploy machine learning models without the need to manage the underlying infrastructure.
- ✓ These services provide scalable, high-performance environments for machine learning, allowing data scientists, developers, and businesses to leverage powerful compute resources on-demand.
- ✓ Cloud ML platforms abstract away the complexity of infrastructure management, enabling users to focus on building and refining models while ensuring that they can scale as needed to handle large datasets and complex computations.
- ✓ These platforms also offer seamless integration with other cloud services, such as data storage, processing, and visualization tools, making it easier to create end-to-end machine learning workflows.



# Cloud ML Overview

(Continued)

- ✓ *The following are the key types of cloud ML services:*



# Cloud ML Overview

1. **Managed Machine Learning Services:** Managed services, like Google Cloud AI Platform, provide a fully managed environment for training, deploying, and serving machine learning models.
2. **Serverless Machine Learning:** Serverless ML services, like Google Cloud AutoML, offer a simplified interface for users who may not have extensive machine learning experience.
3. **Custom Machine Learning Environments:** Cloud ML platforms allow users to create custom environments to train machine learning models using any framework or library.
4. **Model Deployment and Prediction Services:** Cloud ML services also provide tools to deploy models into production environments for real-time prediction or batch inference.
5. **Machine Learning Pipelines and Automation:** Cloud ML tools like Kubeflow help automate and streamline the end-to-end ML lifecycle, including data ingestion, model training, evaluation, and deployment.

# TensorFlow Model

- ✓ A TensorFlow model refers to a computational graph that represents a machine learning model built using the TensorFlow framework. TensorFlow is an open-source platform widely used for building and training machine learning models.
- ✓ It provides both high-level APIs like Keras for simplicity and low-level APIs for flexibility and customisation. TensorFlow models are designed to work with large-scale datasets, allowing users to create, train, and deploy models for various machine learning tasks, including classification, regression, image processing, and natural language processing.
- ✓ ***The following are the key aspects of tensorflow model:***

**1. Building a TensorFlow Model**

**2. Model Training**

**3. TensorFlow Model Layers**

**4. Model Evaluation**

**5. Model Saving and Loading**

# TensorFlow Model

## 1. Building a TensorFlow Model:

- A TensorFlow model typically consists of layers of computations (represented as nodes in a computational graph) connected by tensors (data flowing through the graph). Models can be built using Keras, TensorFlow's high-level API for neural networks, or with low-level TensorFlow operations for more control over model architecture.

## 2. Model Training:

- Training a TensorFlow model involves feeding data through the model, calculating the loss (error), and updating the model's weights using an optimiser. The training process adjusts the weights of the model to minimise the loss function.

## 3. TensorFlow Model Layers:

- Layers are the fundamental building blocks of a neural network in TensorFlow. Layers define how data flows through the model. Common layers include Dense (fully connected layer), Conv2D (convolutional layer for image processing), LSTM (long short-term memory layer for sequential data), and more.

# TensorFlow Model

## 4. Model Evaluation:

- After training, the model is evaluated on a validation or test dataset to check its performance.
- Common evaluation metrics include accuracy, precision, recall, and F1 score.

## 5. Model Saving and Loading:

- Once a model is trained, it can be saved to disk and later loaded for inference or further training.
- TensorFlow provides functions like `model.save()` and `tf.keras.models.load_model()` for saving and loading the model.



# Running of ML Model

- ✓ Running a machine learning (ML) model involves the process of training, evaluating, and making predictions using the model. Once an ML model is built, the next step is to run it using data, either for training (learning from data) or inference (predicting on new data).
- ✓ This process can be done locally on a machine, on cloud services, or on specialised hardware like GPUs/TPUs for large-scale models.
- ✓ The success of running an ML model depends on various factors such as the quality of data, model parameters, and computational resources available.
- ✓ ***The following are the key aspects of running an ML model:***
  1. **Training the Model:** Training a model involves feeding data into the model, performing forward passes through the network, calculating the loss (or error), and using optimisation algorithms (like gradient descent) to adjust the model weights.

# Running of ML Model

2. **Model Evaluation:** After training, the model is evaluated using a separate validation or test dataset to measure how well it generalises to new, unseen data. Common evaluation metrics include accuracy, precision, recall, and F1 score.
3. **Prediction (Inference):** After training and evaluation, the model is used to make predictions on new, unseen data (inference). The model applies what it has learned to predict the output for new input data. Running predictions is often the final step in deploying an ML model in production or for providing real-time predictions.
4. **Optimising Model Performance:** To improve performance, you can fine-tune hyperparameters, adjust the model architecture, or apply techniques like regularisation and dropout to prevent overfitting.
5. **Deploying the Model:** After the model has been trained and evaluated, it needs to be deployed for use in a production environment. Deployment involves serving the model for real-time predictions or batch processing, and it can be done using TensorFlow Serving, TensorFlow Lite (for mobile), or Cloud ML APIs for scalability.

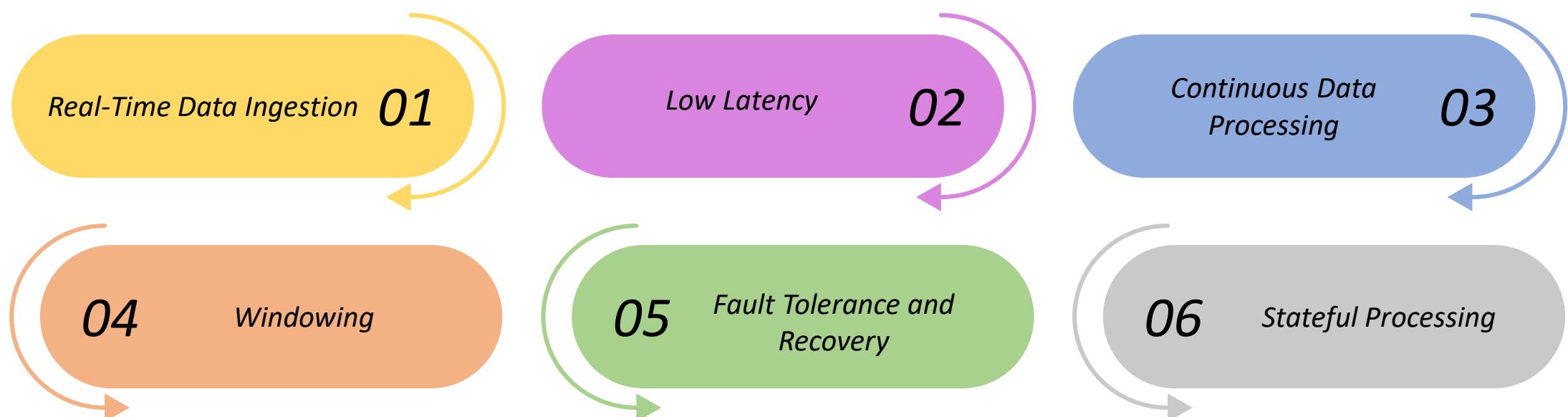
# Module 10: Architecture of Streaming Analytics Pipelines

- Understand Stream Data Processing
- Handling Variable Data Volumes
- Designing a Streaming Pipeline



# Understand Stream Data Processing

- ✓ Stream data processing refers to the continuous processing of real-time data as it arrives. Unlike batch processing, where data is collected over a period and processed in chunks, stream processing involves processing data in real-time or in small time windows.
- ✓ This is essential for applications that require timely insights and decisions based on live data streams, such as real-time analytics, IoT sensor data, user activity tracking, fraud detection, and social media analysis.
- ✓ ***The following are the key aspects of stream data processing:***



# Understand Stream Data Processing

## 1. Real-Time Data Ingestion:

- Stream processing begins with the real-time ingestion of data from various sources such as Apache Kafka, Google Cloud Pub/Sub, or AWS Kinesis. These services provide reliable, low-latency message delivery to stream processing systems.

## 2. Low Latency:

- A defining feature of stream processing is its ability to process data with minimal latency. This means that as soon as new data arrives, it is processed and ready for analysis in near real-time.

## 3. Continuous Data Processing:

- Unlike batch processing, which works on fixed datasets, stream processing works on a continuous flow of data, processing each incoming piece of data immediately. This continuous nature allows the system to handle unbounded datasets (streams that don't have a defined start or end).

# Understand Stream Data Processing

4. **Windowing:** To perform operations on real-time data, stream processing systems often use windowing to divide the stream into finite chunks (or windows). Common types of windows are:

- **Fixed Window:** A window of a fixed size (e.g., 5 seconds).
- **Sliding Window:** A window that slides over the stream at regular intervals.
- **Session Window:** A window that dynamically adjusts based on activity.

5. **Fault Tolerance and Recovery:**

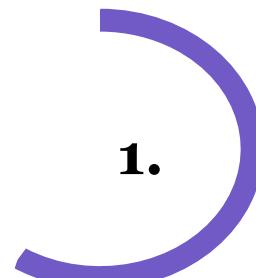
- Stream data processing systems need to be fault-tolerant, ensuring that data is not lost if failures occur. Most stream processing systems, like Apache Kafka and Google Cloud Dataflow, provide built-in mechanisms to handle data recovery in case of system crashes.

6. **Stateful Processing:**

- In stream processing, the system often needs to keep track of state across multiple events (e.g., user sessions, financial transactions). This is called stateful processing.

# Handling Variable Data Volumes

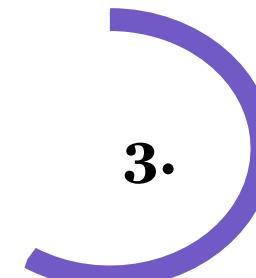
- ✓ In data engineering, handling variable data volumes refers to efficiently processing datasets that fluctuate in size or arrival rate. These variations can occur due to spikes in traffic, changing data patterns, or seasonal demand.
- ✓ Proper handling of variable data volumes is critical for maintaining system performance, ensuring data accuracy, and meeting operational requirements. The ability to manage fluctuating data volumes efficiently is essential for real-time data processing, batch processing, and maintaining scalability in production environments.
- ✓ ***The following are the key aspects of handling variable data volumes:***



Auto-Scaling



Sharding



Buffering



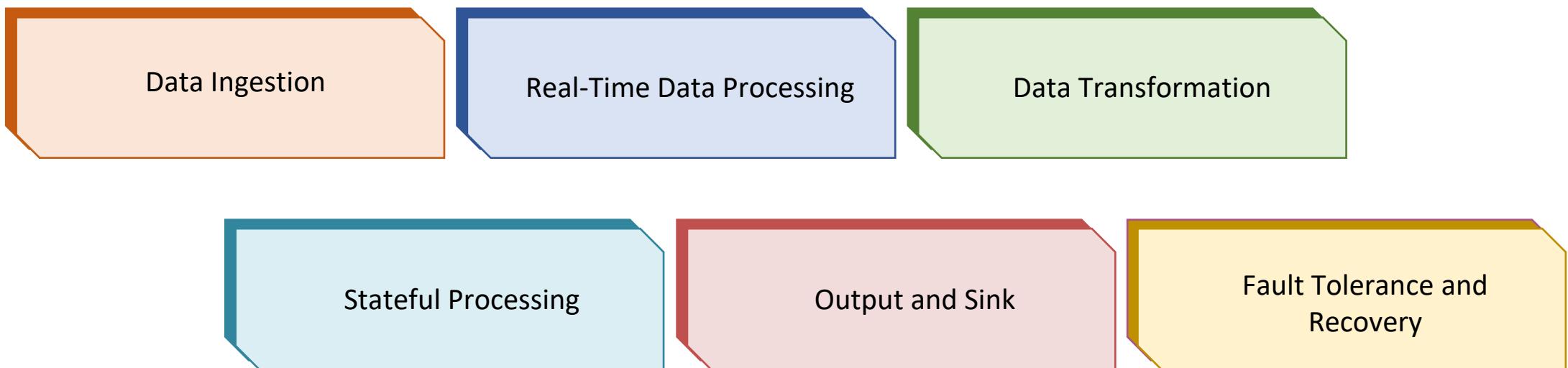
Data Partitioning

# Handling Variable Data Volumes

1. **Auto-Scaling:** Auto-scaling refers to automatically adjusting computational resources based on the volume of incoming data. In cloud environments, auto-scaling ensures that resources like CPU, memory, and storage scale up during high data volumes and scale down when data volume decreases, optimising both performance and costs.
2. **Sharding:** Sharding involves dividing the data into smaller, manageable pieces (called **shards**) that can be processed independently. Each shard contains a subset of the total data, and this allows parallel processing of data across multiple resources or nodes, improving scalability.
3. **Buffering:** Buffering is used to store incoming data temporarily in memory or storage when the system is processing data at a slower rate than it is being ingested. This helps to smooth out fluctuations in the incoming data stream.
4. **Data Partitioning:** Data partitioning refers to the strategy of splitting data into distinct segments based on certain criteria (such as time, geography, or user ID). Partitioning helps manage large datasets and ensures that specific subsets of data are processed independently, reducing bottlenecks.

# Designing a Streaming Pipeline

- ✓ Designing a streaming pipeline involves creating a system that continuously processes data in real-time as it is ingested. Streaming data pipelines are essential in scenarios where data flows constantly, such as in real-time analytics, sensor data collection, fraud detection, or live user activity tracking.
- ✓ The goal of a streaming pipeline is to capture data as it arrives, perform transformations, and output meaningful insights or actions in real time.
- ✓ ***The following are the key aspects of designing a streaming pipeline:***



# Designing a Streaming Pipeline

1. **Data Ingestion (Real-Time Data Sources):** The first step in any streaming pipeline is ingesting real-time data from various sources. Common streaming data sources include Apache Kafka, Google Cloud Pub/Sub, AWS Kinesis, or real-time data from sensors and devices.
2. **Real-Time Data Processing:** The core of the streaming pipeline is the real-time processing of data. Streaming frameworks like Apache Flink, Apache Beam, and Google Cloud Dataflow allow you to define transformations that can process data as it flows through the pipeline.
3. **Data Transformation:** After data ingestion, the next step is data transformation, which involves cleaning, enriching, filtering, or aggregating data. Common transformations include:
  - **Filtering:** Removing unwanted or irrelevant data.
  - **Aggregation:** Summing, counting, or averaging values over specific time windows.
  - **Enrichment:** Adding contextual information, such as adding user details to event data.

# Designing a Streaming Pipeline

4. **Stateful Processing:** Stateful processing is required when you need to track information across multiple events (e.g., keeping track of a user's session or detecting fraud based on patterns in recent transactions).
  - In streaming systems, maintaining state across events can be complex. Systems like Apache Flink and Google Cloud Dataflow provide built-in mechanisms for managing state efficiently and ensuring fault tolerance.
5. **Output and Sink (Real-Time Data Delivery):** After processing and transforming the data, the results need to be written to a sink or destination, such as BigQuery, Google Cloud Storage, HDFS, Datastore, or real-time dashboards.
  - Real-time delivery to sinks allows the results of the streaming data pipeline to be immediately available for analysis, storage, or further processing.
6. **Fault Tolerance and Recovery:** Fault tolerance ensures that data is not lost or corrupted in case of failures, and the pipeline can continue processing. In streaming pipelines, data needs to be processed exactly once or at least once to ensure data integrity.

# Module 11: Ingesting Variable Volumes

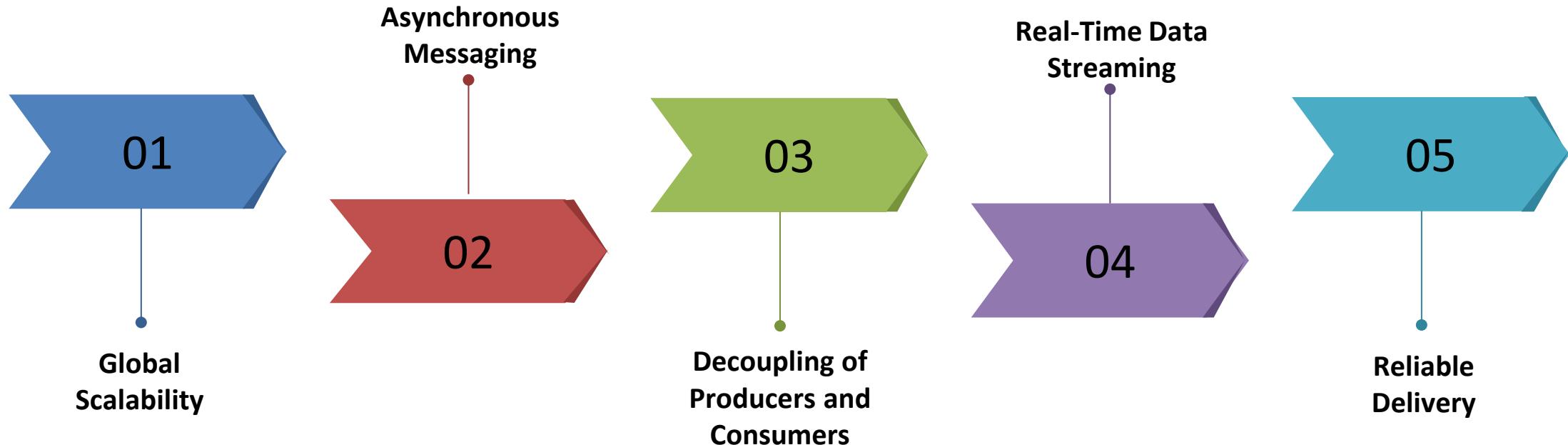
- Cloud Pub/Sub Overview
- Working of Cloud Pub/Sub



# Cloud Pub/Sub Overview

- ✓ Cloud Pub/Sub is a fully managed messaging service provided by Google Cloud that enables asynchronous communication between different applications, services, or components.
- ✓ It facilitates real-time data streaming and event-driven architectures by decoupling senders and receivers, ensuring efficient and scalable communication.

## ***Key Features***



# Cloud Pub/Sub Overview

## 1. Global Scalability:

- Cloud Pub/Sub is designed to handle massive amounts of data, offering seamless scalability across global regions.
- This means that whether you are sending a few messages or handling millions of events per second, the service can grow with your needs.
- It automatically scales to accommodate spikes in traffic without requiring manual intervention, ensuring consistent performance even under heavy loads.
- The global infrastructure of Google Cloud ensures low-latency message delivery, providing real-time data distribution regardless of geographical location.



# Cloud Pub/Sub Overview

## 2. Asynchronous Messaging:

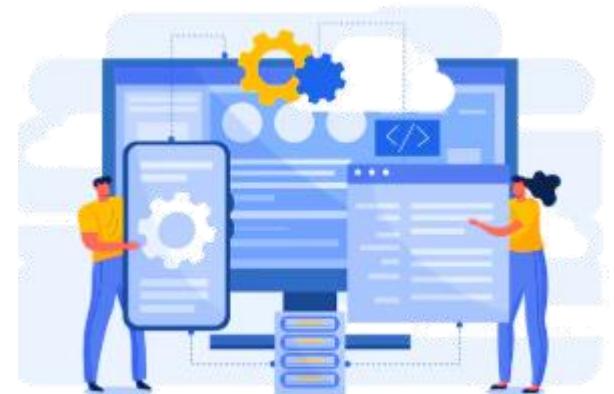


- Cloud Pub/Sub enables asynchronous messaging, which allows applications to send messages without waiting for an immediate response.
- This decoupling of message production and consumption makes systems more responsive and scalable, as producers can continue operating without waiting for the consumer to process the data.
- By leveraging asynchronous communication, applications can handle peak traffic and process messages at their own pace.
- This also minimises bottlenecks, as slow consumers do not block fast producers. Asynchronous messaging is particularly beneficial in cloud-native architectures, where services need to operate independently.

# Cloud Pub/Sub Overview

### 3. Decoupling of Producers and Consumers:

- Cloud Pub/Sub provides an architecture where producers (publishers) and consumers (subscribers) operate independently, without needing to be directly connected.
- Publishers send messages to a topic without needing to know who will consume the messages, and subscribers pull or receive messages from a subscription without needing to know the publishers.
- This decoupling enhances system flexibility, allowing for easier scaling and maintenance.
- It also enables a more fault-tolerant system since any component can fail independently without affecting the entire flow.



# Cloud Pub/Sub Overview

## 4. Real-Time Data Streaming:



- Cloud Pub/Sub supports real-time data streaming, making it ideal for event-driven architectures.
- This allows applications to react instantly to changes or updates in data, which is crucial in industries like finance, healthcare, and IoT.
- With real-time message delivery, Cloud Pub/Sub enables systems to provide up-to-date information continuously, supporting a wide range of use cases such as live data monitoring, alert systems, and dynamic content delivery.
- By enabling seamless data flow between distributed services, it enhances the ability to process and analyse data as it is generated.

# Cloud Pub/Sub Overview

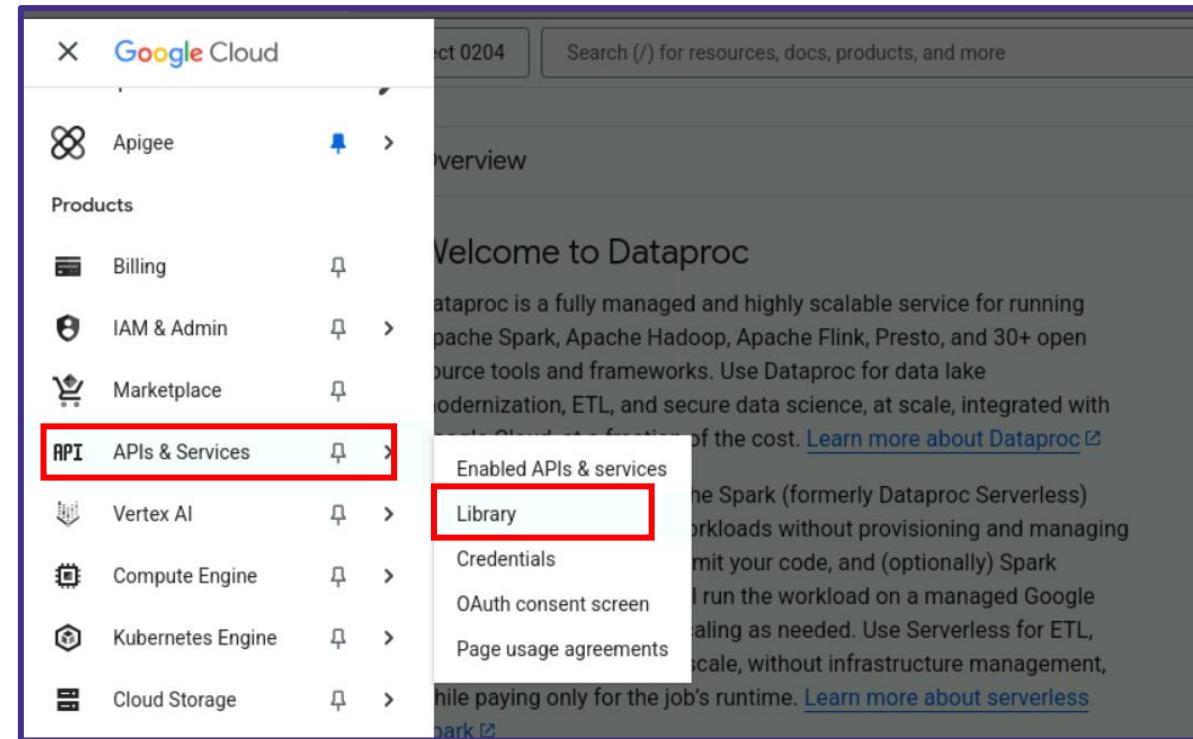
## 5. Reliable Delivery:

- Cloud Pub/Sub guarantees at least once delivery of messages, ensuring that no messages are lost, even in the event of failures.
- This reliability is achieved through message persistence, which stores messages in durable storage until they are successfully acknowledged by subscribers.
- The system also includes built-in retry mechanisms that automatically attempt to resend failed messages, ensuring that consumers have a chance to process the messages.
- Additionally, Cloud Pub/Sub supports dead-letter topics, where messages that cannot be processed within a defined retry limit are redirected for further analysis or troubleshooting.

# Working of Cloud Pub/Sub

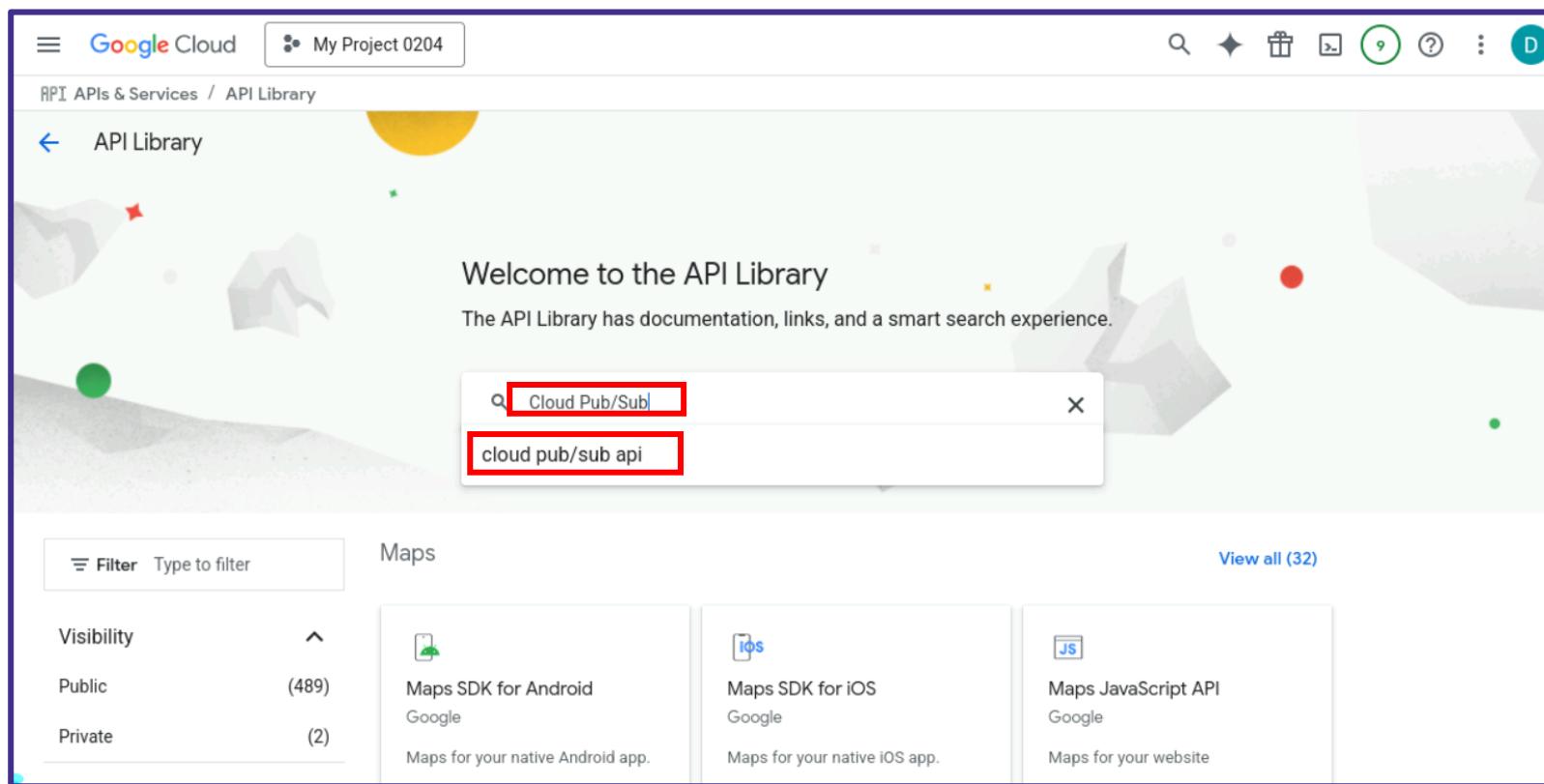
- ✓ *The following are the steps for working with Cloud Pub/Sub:*

**Step 1:** Navigate to Google Cloud console and then click on **APIs & Services > Library**



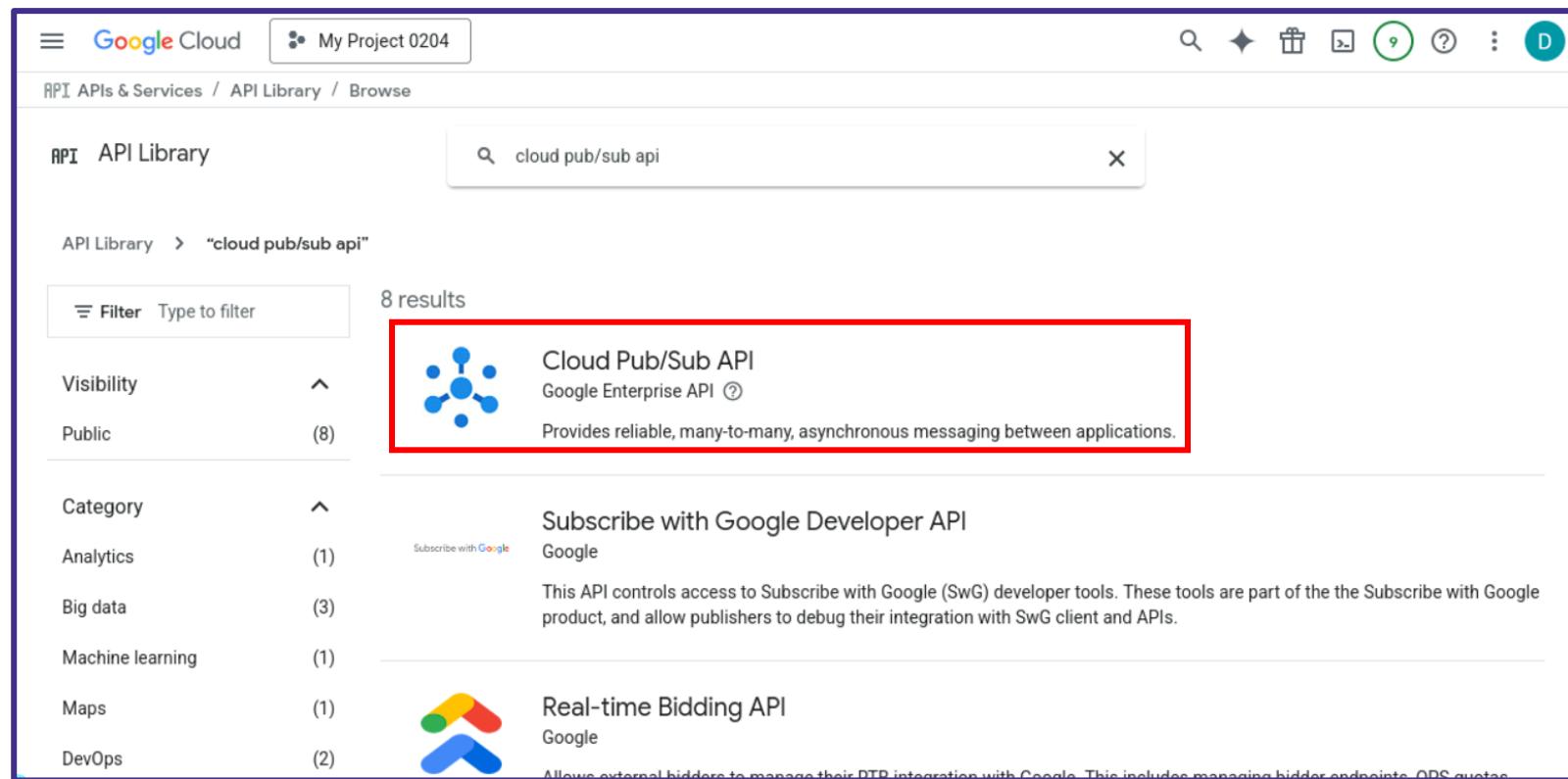
# Working of Cloud Pub/Sub

**Step 2:** Now, search for **Cloud Pub/Sub** and then click on it



# Working of Cloud Pub/Sub

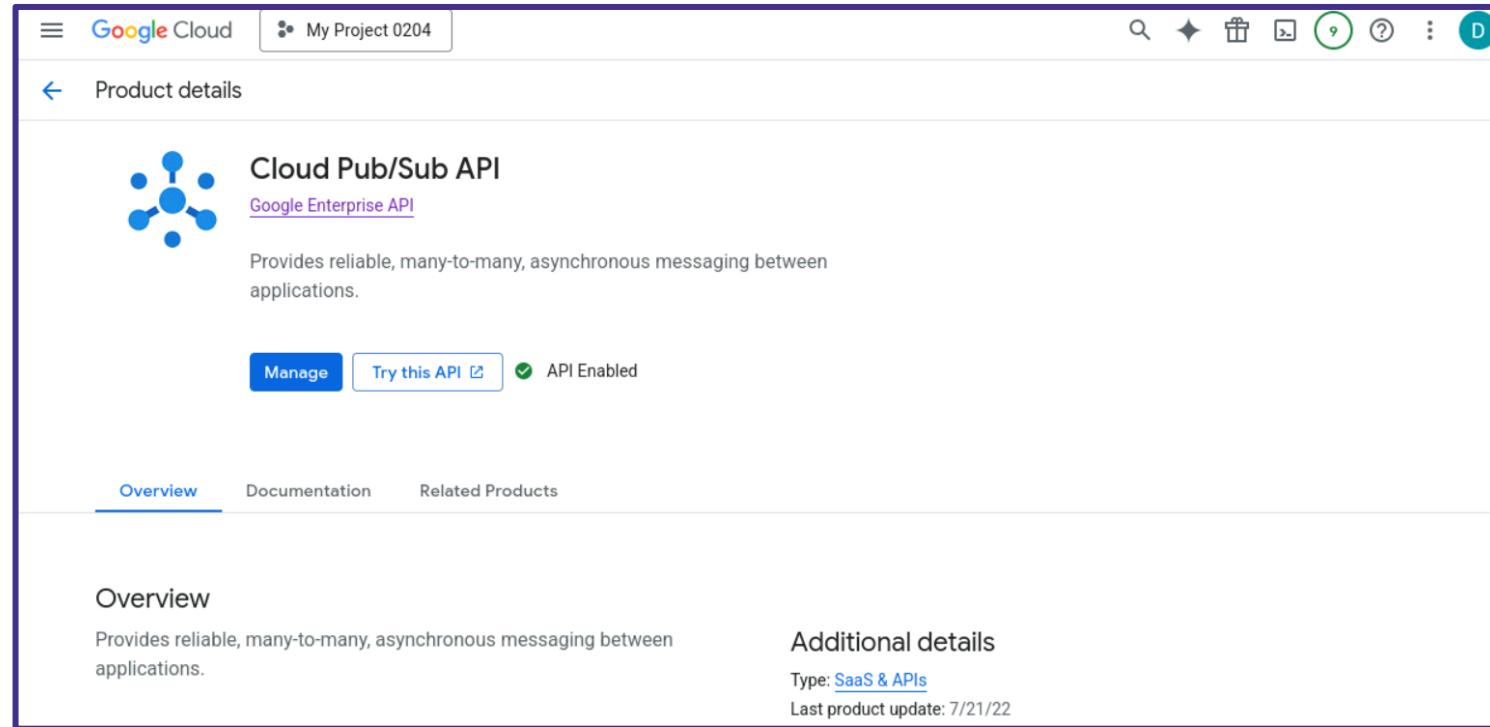
## Step 3: Click on Cloud Pub/Sub API



# Working of Cloud Pub/Sub

(Continued)

- ✓ Make sure that the Cloud Pub/Sub API is enabled.



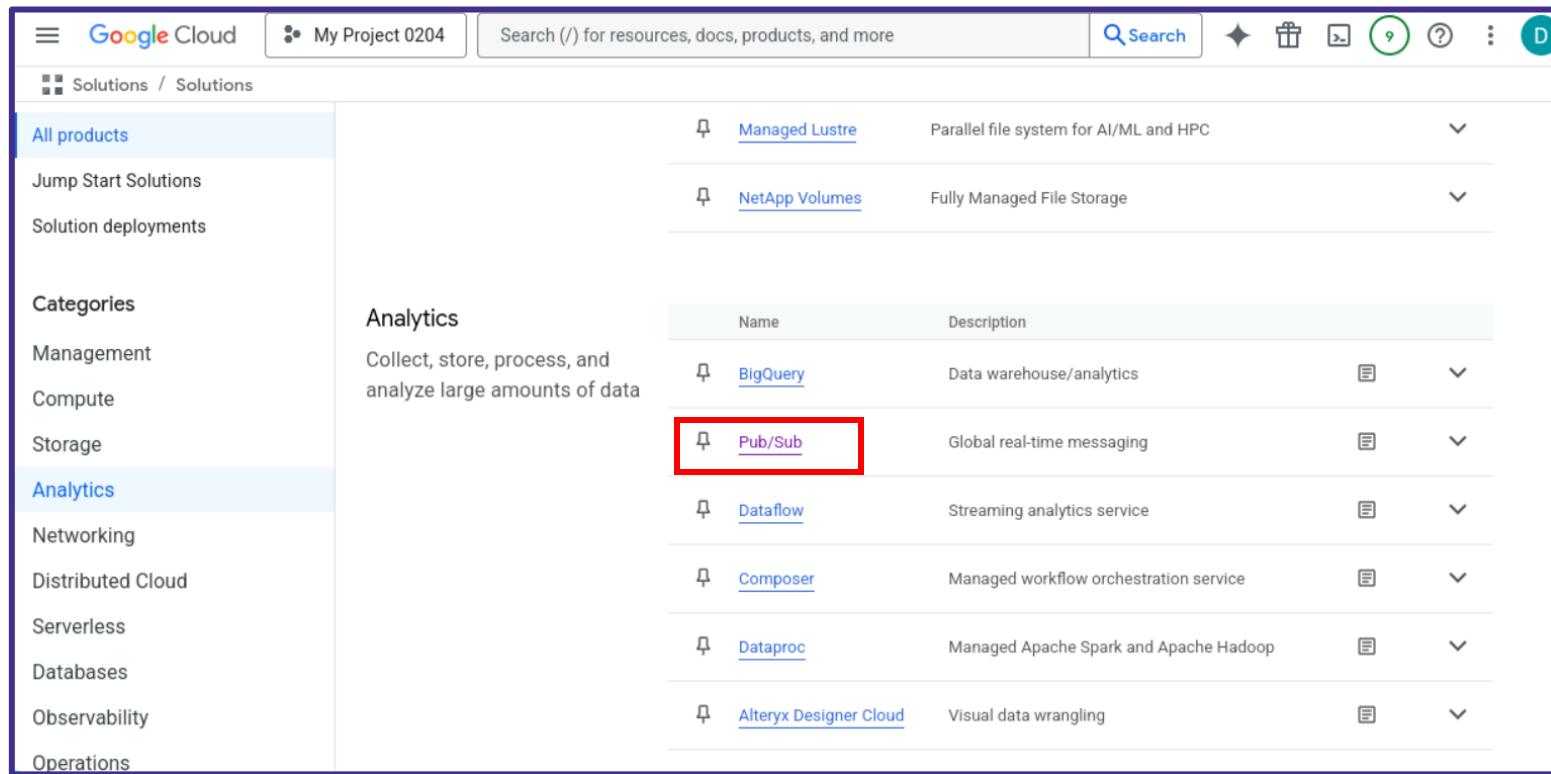
# Working of Cloud Pub/Sub

**Step 4:** In the search bar, search for **All Products** and then click on **All products**

The screenshot shows the Google Cloud search interface. At the top, there is a navigation bar with the Google Cloud logo, a project selector for 'My Project 0204', and a search bar containing the text 'All Products'. The search bar is highlighted with a red box. Below the navigation bar is a search bar labeled 'Search' with a placeholder 'Search'. The main content area has tabs for 'ALL', 'DOCUMENTATION & TUTORIALS', 'RESOURCES', and 'MARKETPLACE & APIs'. The 'ALL' tab is selected and highlighted in blue. On the left, there is a sidebar titled 'Filter by' with checkboxes for 'Product or Page', 'Documentation or tutorial', 'Marketplace and APIs', 'Organization', 'Folder', 'Project', and 'Resources'. Below this is a section for 'Resource filters' with dropdown menus for 'Project, folder, or org' set to 'My Project 0204' and 'Resource type' set to 'Any'. The main search results are displayed under the heading 'Search results'. It says 'Showing 30 of 96 results for "All Products".' A red box highlights the 'All products' link in the results. The results show two items: 'All products' (a Solutions item) and 'Chalk' (a Marketplace Product). The 'All products' item has details: Type: Product or Page, Product: Solutions. The 'Chalk' item has details: Type: Marketplace Product, Producer: Chalk. To the right of the search results, there is a sidebar with tips: 'Speed up your search with these quick tips', 'Press / to search from anywhere in the console', 'Find 111 resource types and more by using the new filters on this page', 'Find an API key by entering its key string in search', and 'Search for top AWS or Azure offerings and get suggestions for similar Google Cloud products.'

# Working of Cloud Pub/Sub

Step 5: Click on **Pub/Sub** under Analytics section



The screenshot shows the Google Cloud Solutions interface. The left sidebar has a 'Categories' section with 'Analytics' selected, indicated by a blue background. The main area shows the 'Analytics' category with a sub-description: 'Collect, store, process, and analyze large amounts of data'. Below this is a table listing various services:

Name	Description
BigQuery	Data warehouse/analytics
Pub/Sub	Global real-time messaging
Dataflow	Streaming analytics service
Composer	Managed workflow orchestration service
Dataproc	Managed Apache Spark and Apache Hadoop
Alteryx Designer Cloud	Visual data wrangling

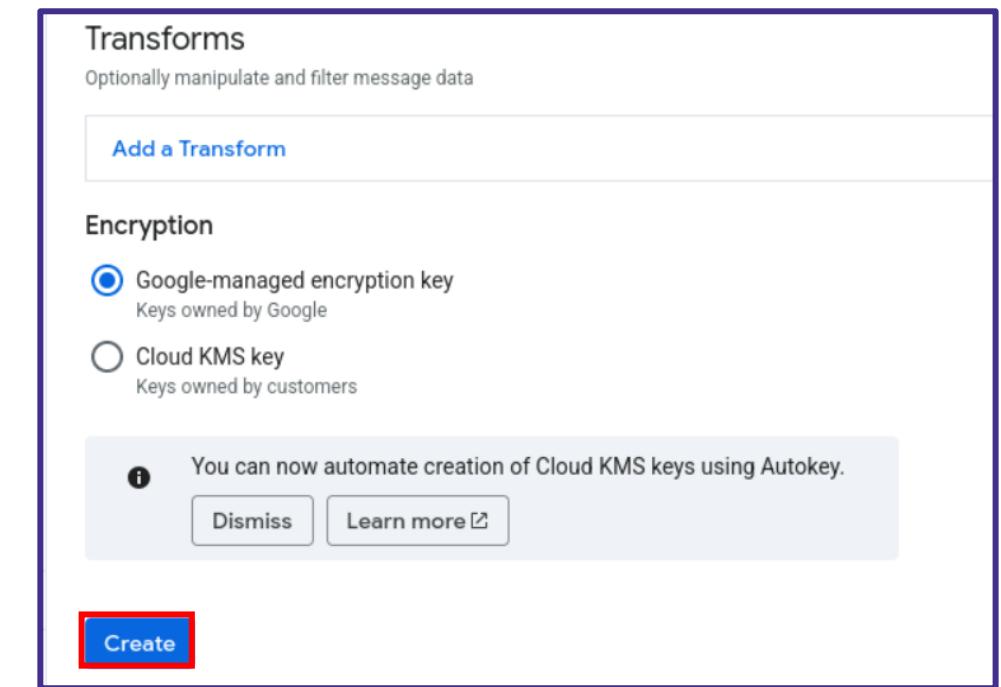
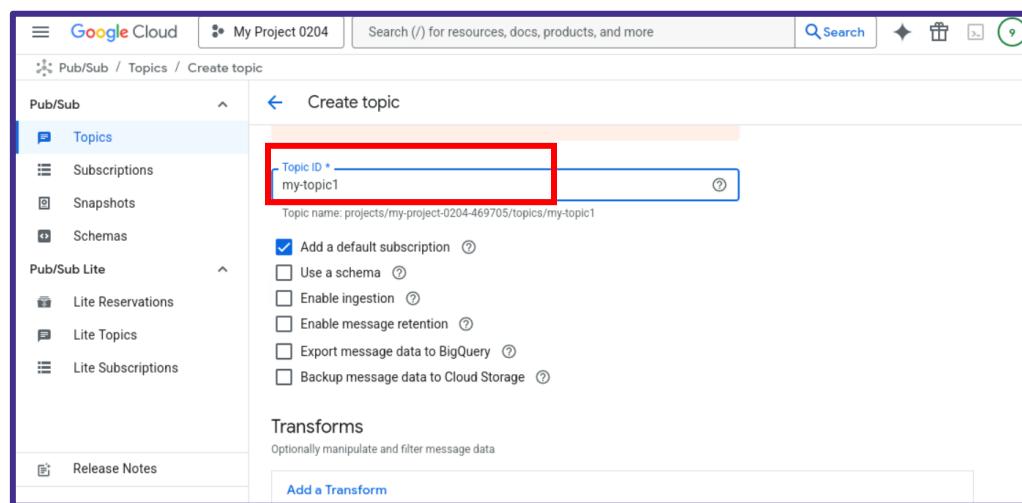
# Working of Cloud Pub/Sub

Step 6: Click on Topics > Create topic

The screenshot shows the Google Cloud Platform interface for the Pub/Sub service. The top navigation bar includes the Google Cloud logo, the project name "My Project 0204", a search bar, and various navigation icons. On the left, a sidebar menu under "Pub/Sub" has "Topics" selected, indicated by a red box. At the top center, there is a "Topics" section with a "Create topic" button highlighted by a red box. Below this, there are two promotional cards: one about SMTs for streaming pipelines and another about simplifying data lake pipelines with Cloud Storage subscriptions. The main content area shows a table with columns for "List", "Metrics", "Filter topics", "Permissions", "Labels", and "Storage policy". A message at the bottom right says "Please select at least one resource."

# Working of Cloud Pub/Sub

Step 7: Enter Topic ID name and then click on Create

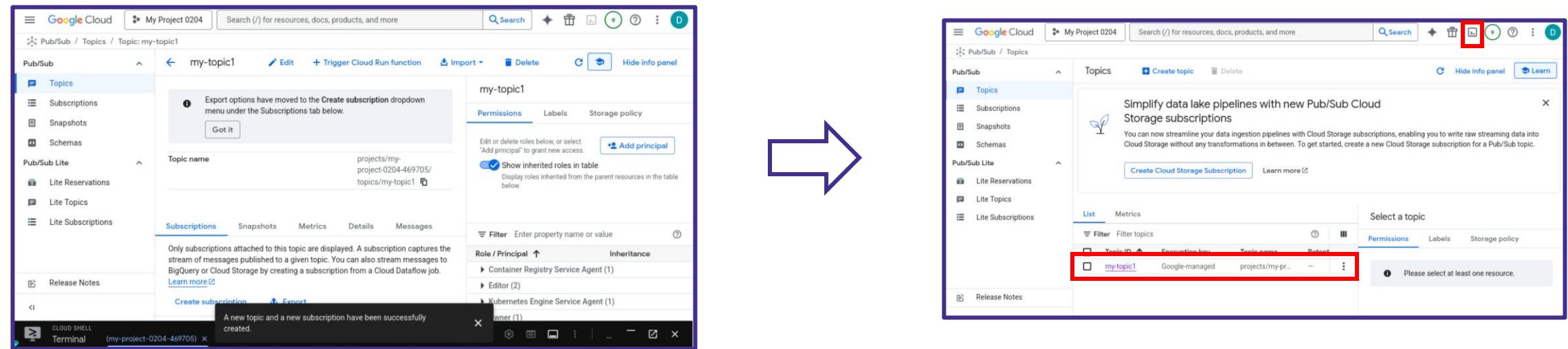


# Working of Cloud Pub/Sub

(Continued)

- ✓ A new topic and subscription has been created successfully.

## Step 8: Click on Google Cloud Shell



# Working of Cloud Pub/Sub

**Step 9:** Use the following **gcloud** command to publish a message to the topic

```
delegatetwelve@cloudshell:~ (my-project-0204-469705)$ gcloud pubsub topics publish my-topic1 --message "Hello, Cloud Pub/Sub!"  
messageIds:  
- '15968358626363666'  
delegatetwelve@cloudshell:~ (my-project-0204-469705)$ █
```

# Working of Cloud Pub/Sub

Step 10: Navigate to the **Subscriptions** section and click on the newly created subscription

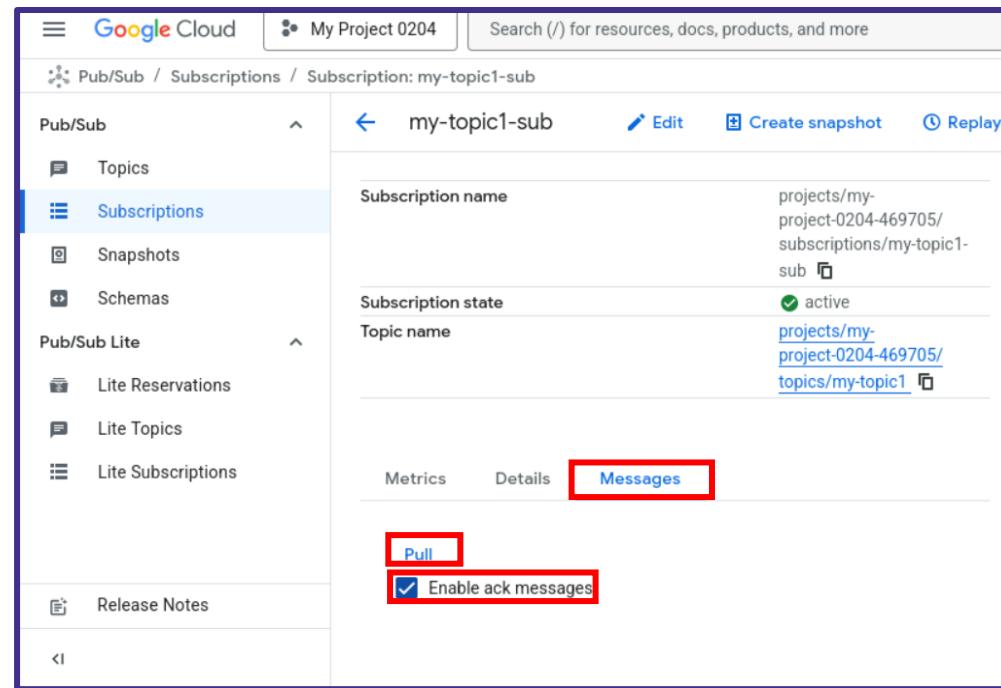
The screenshot shows the Google Cloud Platform interface for managing Pub/Sub subscriptions. The left sidebar has sections for Topics, Snapshots, Schemas, and Lite Reservations, Lite Topics, and Lite Subscriptions. The main area is titled 'Subscriptions' and contains a table with the following data:

State	Subscription ID	Delivery type
<input type="checkbox"/>	<a href="#">my-topic1-sub</a>	Pull

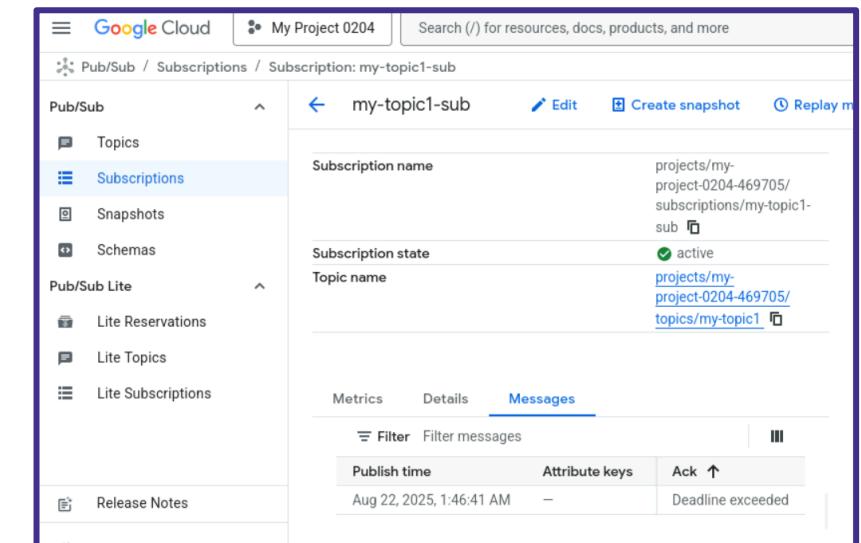
# Working of Cloud Pub/Sub

**Step 11:** Click on **Messages**, then **Enable ack messages** and then click on **Pull**

- ✓ In this way you can work with cloud pub/sub.



This screenshot shows the Google Cloud Pub/Sub Subscriptions page for a project named 'My Project 0204'. The left sidebar has 'Subscriptions' selected. The main area shows a subscription named 'my-topic1-sub'. The 'Messages' tab is highlighted with a red box. Below it, there are two buttons: 'Pull' and 'Enable ack messages', with 'Enable ack messages' checked. The 'Subscription name' and 'Subscription state' sections are also visible.



This screenshot shows the same Google Cloud Pub/Sub Subscriptions page after performing the steps. The 'Messages' tab is still selected. A single message is listed in the table: 'Publish time: Aug 22, 2025, 1:46:41 AM', 'Attribute keys: -', and 'Ack: Deadline exceeded'.

# Module 12: Implementing Streaming Pipelines

- Stream Processing Overview
- Watermark
- Triggers



# Stream Processing Overview

- ✓ Stream processing refers to the continuous processing of data in real time as it is generated or received, rather than in batches. This approach allows systems to process, analyse, and act on data immediately as it flows through the pipeline, enabling real-time insights and actions.

## **Key Aspects**

1. **Real-Time Data:** Stream processing handles data that is continuously generated from sources such as sensors, user activity, or financial transactions, and processes it as it arrives.
2. **Event-Driven Architecture:** It is often used in event-driven systems where actions are triggered by incoming data streams (e.g., alerts when a specific threshold is met).
3. **Low Latency:** Stream processing minimises the delay between data arrival and processing, ensuring immediate responses to events or changes in data.
4. **Distributed Systems:** Stream processing is often distributed across multiple systems or nodes, allowing it to scale efficiently and handle large volumes of data.

# Stream Processing Overview

## Benefits

- 1. Timely Insights:** It enables businesses to respond instantly to changing conditions.
- 2. Scalability:** Stream processing systems are highly scalable and can handle vast amounts of incoming data in real-time.
- 3. Fault Tolerance:** It is built to be resilient, often incorporating mechanisms for recovering from errors without losing data.
- 4. Enhanced Decision-Making:** Real-time processing leads to faster and more informed decisions based on up-to-the-minute data.



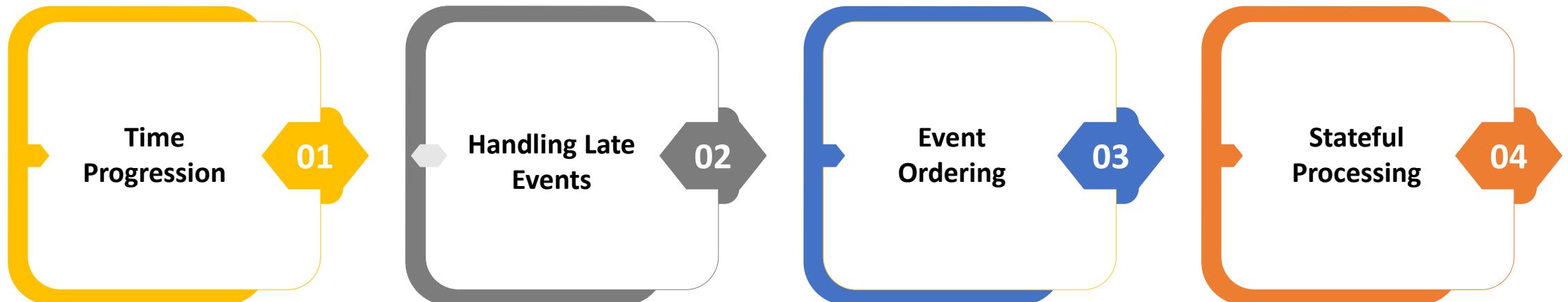
# Watermark

- ✓ A watermark in stream processing is a concept used to track the progress of data within an event stream.
- ✓ It marks the point in the stream up to which the data has been processed and serves as a way to handle out-of-order events in real-time processing.
- ✓ In a distributed system, data events can arrive at different times, and watermarks help manage and ensure accurate processing despite event latency.
- ✓ By using watermarks, systems can handle late-arriving data without holding up the entire process, ensuring that earlier events are processed first.
- ✓ Watermarks help maintain the consistency and integrity of results in real-time data pipelines. It is crucial for systems that require time-windowed operations, like aggregations or joining streams based on time.

# Watermark

(Continued)

## ***Key Aspects of Watermarks***



# Watermark

## 1. Time Progression:

- Watermarks are used to represent the passage of time in an event stream, helping the system track which events have been processed up to a particular timestamp.
- By marking this point, it allows the system to process the next batch of data efficiently.
- Watermarks ensure that the data is processed in the correct order, maintaining a logical sequence for time-based operations.
- This progress tracking ensures that later events do not interfere with the timely processing of earlier ones. They also help avoid processing the same data multiple times, improving system efficiency.



# Watermark

## 2. Handling Late Events:



- In stream processing, late-arriving data can disrupt the flow of processing. Watermarks help to manage such events by marking a point in time beyond which late events are treated differently.
- These events can be dropped, ignored, or sent to a special processing stream for later handling.
- This threshold allows for efficient data processing without waiting indefinitely for late-arriving messages.
- By setting a limit on how late data can be, watermarks prevent bottlenecks and keep the system moving forward.

# Watermark

### 3. Event Ordering:

- In stream processing, data often arrives out of order due to network latencies or other issues. Watermarks help maintain the correct order of events by marking the point in time that the system has reached.
- This ensures that all events up to a specific timestamp have been processed before moving on to the next batch.
- Watermarks allow the system to handle out-of-order data effectively, ensuring that time-based operations such as aggregations or joins are performed correctly.
- They guide the system in determining when it is safe to process new data and when all necessary events for a particular window are complete.



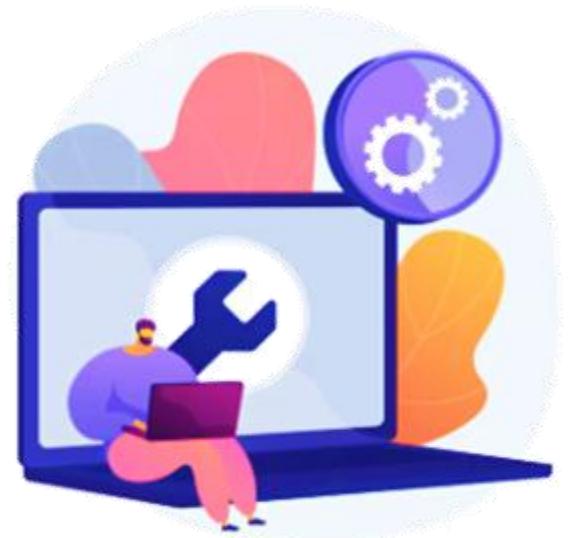
# Watermark

## 4. Stateful Processing:

- When dealing with stateful operations like aggregations or time-based joins, watermarks play a crucial role in maintaining the integrity of computations.
- They help to identify when it is safe to emit results, ensuring that all relevant events have been processed within a given time window.
- This is essential for operations where the system needs to track ongoing state, such as calculating running totals or combining streams of data.
- Watermarks allow the system to process the data progressively, ensuring consistency in calculations and results.

# Triggers

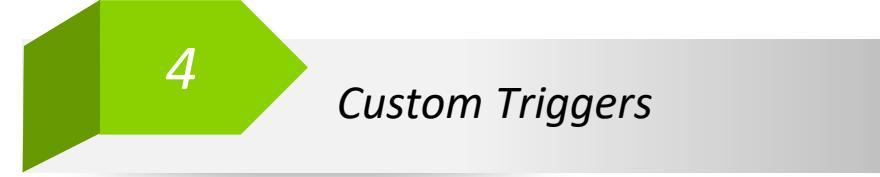
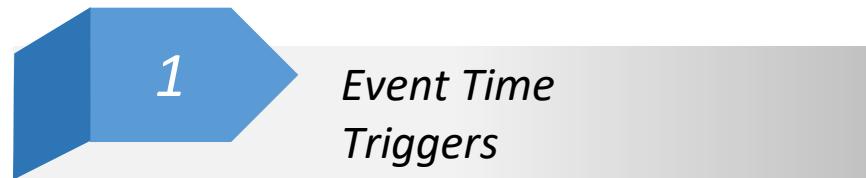
- ✓ A trigger in stream processing is a mechanism used to determine when the results of a time-windowed operation should be emitted.
- ✓ Triggers are essential for managing when to produce output based on the incoming data, especially when processing data in real-time or handling time-based windows.
- ✓ Without triggers, systems would not know when it is appropriate to output results or continue processing the stream.
- ✓ Triggers help ensure that results are emitted at the correct time and in an efficient manner, improving system performance and accuracy.
- ✓ Different types of triggers allow for flexibility in stream processing, including those based on event arrival, window completion, or specific data thresholds.



# Triggers

(Continued)

## ***Key Aspects of Triggers***



# Triggers

1. **Event Time Triggers:** Triggers based on event time ensure that the system emits results when the window of events defined by a specific timestamp has been fully processed. These triggers wait until all events within a particular window have arrived before firing, guaranteeing correct and complete output.
2. **Processing Time Triggers:** In contrast to event time, processing time triggers fire based on when the data is processed, rather than when the event actually occurred. This approach is useful when you need to handle data as it arrives, regardless of event timestamps.
3. **Watermark-Based Triggers:** Watermark-based triggers fire when the watermark reaches a certain threshold, signaling that no more events with a timestamp earlier than that watermark will be processed. This ensures that events are processed in the correct order, and late-arriving data is handled appropriately.
4. **Custom Triggers:** Stream processing frameworks allow developers to define custom triggers based on specific criteria or business logic. These triggers can fire when certain data patterns, thresholds, or conditions are met, providing flexibility in how and when results are emitted.

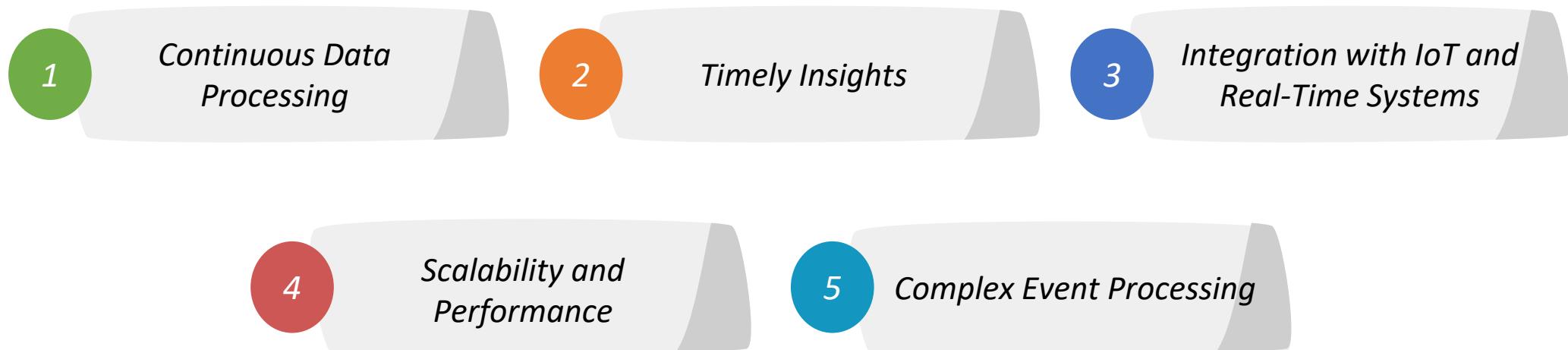
# Module 13: Streaming Analytics and Dashboards

- Streaming Analytics Overview
- Querying Streaming Data with BigQuery
- Google Data Studio Overview



# Streaming Analytics Overview

- ✓ Streaming analytics refers to the real-time processing and analysis of data as it is ingested, enabling businesses to gain insights from live data streams. This approach is crucial for applications where immediate responses and data-driven decisions are required.
- ✓ ***The following are the key points of streaming analytics:***



# Streaming Analytics Overview

## 1. Continuous Data Processing:

- Streaming analytics involves the constant ingestion and processing of data streams in real time, enabling immediate analysis and response.
- This allows businesses to make data-driven decisions as events unfold, reducing the delay between data generation and actionable insights.

## 2. Timely Insights:

- By processing data as it arrives, businesses can derive actionable insights instantly, which is crucial for industries like finance, healthcare, and e-commerce.
- This real-time capability helps organisations stay ahead of competitors and address issues as soon as they arise.

# Streaming Analytics Overview

## 3. Integration with IoT and Real-Time Systems:

- Streaming analytics plays a vital role in real-time monitoring and decision-making, especially when dealing with Internet of Things (IoT) devices and operational systems.
- This integration allows organisations to track assets, monitor conditions, and automate responses based on real-time data.

## 4. Scalability and Performance:

- The ability to process large volumes of data in parallel across multiple systems ensures that streaming analytics can scale as data grows, without compromising performance.
- It can handle data streams from diverse sources while maintaining low-latency processing and high availability.

# Streaming Analytics Overview

## 5. Complex Event Processing:

- Streaming analytics allows for the detection of patterns and correlations in real-time, enabling businesses to respond to events and anomalies as they occur.
- By leveraging event processing engines, organisations can create complex rules for monitoring and managing workflows effectively.
- By utilising machine learning algorithms alongside event processing engines, businesses can improve the accuracy of event detection, predict future trends, and automate decision-making processes in real-time.



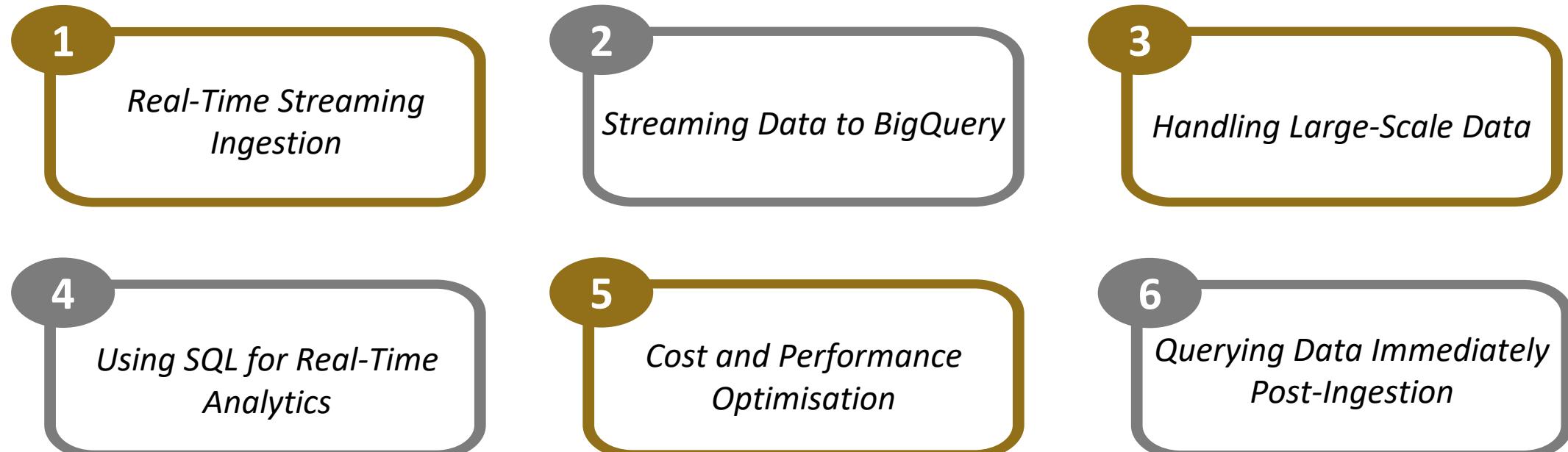
# Querying Streaming Data with BigQuery

- ✓ BigQuery, Google Cloud's fully-managed data warehouse, supports real-time analytics and allows users to query streaming data directly.
- ✓ By ingesting data streams via Google Cloud Pub/Sub or other methods, BigQuery provides a seamless interface for querying data in real-time.
- ✓ It eliminates the need for complex infrastructure management, allowing organisations to focus on analytics and insights.
- ✓ BigQuery's ability to handle massive datasets in real-time makes it ideal for applications that require instant data processing and decision-making.
- ✓ Additionally, with its serverless architecture, users can scale seamlessly based on their streaming data volume without worrying about resource provisioning.

# Querying Streaming Data with BigQuery

(Continued)

- ✓ *The following are the key aspects of querying streaming data with BigQuery:*



# Querying Streaming Data with BigQuery

## 1. Real-Time Streaming Ingestion:

- BigQuery supports real-time data streaming, enabling users to query data as it arrives. Data can be ingested through Pub/Sub or Cloud Storage in JSON, Avro, or Parquet formats, making it suitable for live analytics.

## 2. Streaming Data to BigQuery:

- Data streams are inserted into BigQuery using the Streaming API. The API allows users to push individual data rows to a table, offering low-latency insertion without requiring batch processing.

## 3. Handling Large-Scale Data:

- BigQuery can scale to handle massive volumes of streaming data. By utilising distributed architecture, it allows for querying large datasets while maintaining performance, even with high-throughput data streams.

# Querying Streaming Data with BigQuery

## 4. Using SQL for Real-Time Analytics:

- Once the data is ingested, users can execute SQL queries directly on the streaming data. BigQuery supports SQL-based querying, including window functions, aggregations, and filtering, enabling advanced analysis on streaming data.

## 5. Cost and Performance Optimisation:

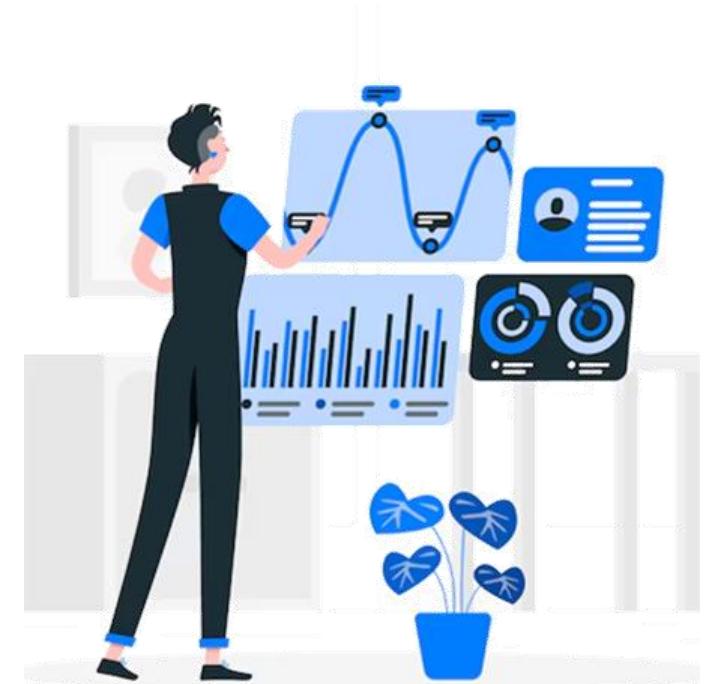
- BigQuery allows users to optimise costs by partitioning tables and using clustering techniques. Streaming data can be queried efficiently with minimal cost, especially when optimised queries are used.

## 6. Querying Data Immediately Post-Ingestion:

- Unlike batch processing, BigQuery allows users to query data seconds after it is ingested, which is critical for applications requiring immediate insights, such as fraud detection and real-time monitoring.

# Google Data Studio Overview

- ✓ Google Data Studio is a powerful, free data visualisation and reporting tool that allows users to transform raw data into interactive and customisable reports and dashboards.
- ✓ Integrated with various Google services like Google Analytics, Google Ads, and BigQuery, it enables users to create real-time, shareable insights.
- ✓ With an intuitive drag-and-drop interface, users can easily connect to multiple data sources, manipulate data, and present it in an engaging format.
- ✓ As a cloud-based tool, Google Data Studio fosters collaboration and enables users to make data-driven decisions from anywhere, making it ideal for businesses of all sizes.



# Google Data Studio Overview

(Continued)

- ✓ *The following are the key aspects of Google Data Studio:*



# Google Data Studio Overview

## 1. Ease of Use:

- Google Data Studio's intuitive interface and drag-and-drop functionality make it easy for users, even those with no coding experience, to build custom reports and dashboards.
- The platform also offers pre-built templates, helping users get started quickly, while still allowing for extensive customisation for more advanced users.

## 2. Integration with Google Products:

- Seamless integration with Google services such as Google Analytics, Google Ads, BigQuery, and Sheets enables easy access to data from multiple sources, streamlining the reporting process.
- Additionally, Data Studio supports third-party data connectors, allowing users to link to external sources and expand the scope of their reports.

# Google Data Studio Overview

## 3. Customisable Reports and Dashboards:

- Users can create highly customized reports with various charts, graphs, and controls to fit their specific needs, providing a visual representation of key performance indicators (KPIs) and metrics.
- The platform also offers dynamic filtering options, enabling viewers to interact with the data and view personalised insights.

## 4. Real-Time Collaboration:

- As a cloud-based tool, Google Data Studio allows multiple users to collaborate on reports and dashboards in real-time, ensuring that teams can work together to derive insights quickly.
- The ability to leave comments directly within reports ensures that feedback and suggestions are easily integrated into the workflow.

# Google Data Studio Overview

## 5. Sharing and Publishing:

- Reports and dashboards can be easily shared with stakeholders or embedded on websites, facilitating transparent communication and decision-making based on up-to-date data insights.
- With flexible permission settings, users can control who can view, edit, or comment on the reports, ensuring data security and access control.



# Module 14: High Throughput and Low-Latency with Bigtable

- Cloud Spanner Overview
- Designing Bigtable Schema
- Ingesting into Bigtable
- Streaming into Bigtable



# Cloud Spanner Overview

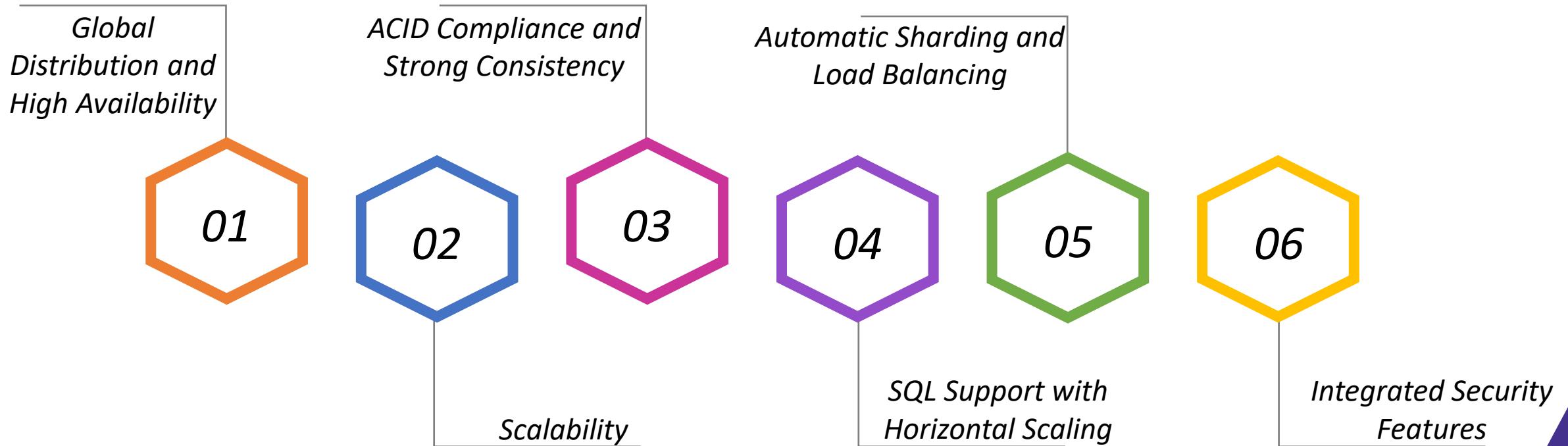
- ✓ Cloud Spanner is a fully managed, scalable, relational database service by Google Cloud, designed for high availability and strong consistency across distributed systems.
- ✓ It combines the benefits of traditional relational databases with the scalability of NoSQL systems.
- ✓ Ideal for applications requiring both high throughput and low-latency performance, Cloud Spanner ensures global distribution of data while maintaining transactional consistency.
- ✓ It provides built-in replication, automatic scaling, and seamless integration with other Google Cloud services, making it a powerful choice for mission-critical applications.



# Cloud Spanner Overview

(Continued)

- ✓ *The following are the key features of cloud spanner:*



# Cloud Spanner Overview

## 1. Global Distribution and High Availability:

- Cloud Spanner automatically replicates data across multiple regions and data centers, ensuring high availability and disaster recovery.
- This global distribution enables low-latency access to data, no matter where the users are located.

## 2. Scalability:

- Designed for massive scalability, Cloud Spanner can handle large-scale transactional workloads. It provides horizontal scaling without the need for manual sharding, enabling seamless growth without performance degradation.

## 3. ACID Compliance and Strong Consistency:

- Cloud Spanner ensures full ACID (Atomicity, Consistency, Isolation, Durability) compliance, guaranteeing transactional integrity. It provides strong consistency across distributed systems, ensuring that all transactions are accurate and reliable.

# Cloud Spanner Overview

## 4. SQL Support with Horizontal Scaling:

- Cloud Spanner supports SQL for querying, including advanced features such as joins, indexes, and foreign keys. This allows users to leverage relational database features while benefiting from the scalability of NoSQL.

## 5. Automatic Sharding and Load Balancing:

- Cloud Spanner automatically manages the sharding of data and load balancing across resources, ensuring that data is distributed efficiently. This provides consistent high performance even during large traffic spikes or variable workloads.

## 6. Integrated Security Features:

- Cloud Spanner offers built-in encryption, both in transit and at rest, to secure data. It integrates with Google Cloud's Identity and Access Management (IAM) for fine-grained access control, ensuring that only authorised users can access sensitive data.

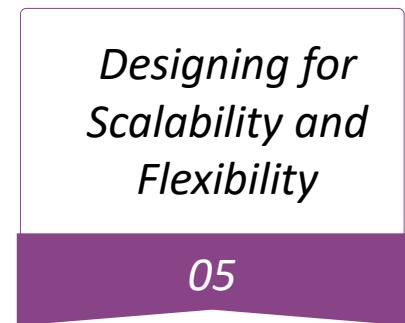
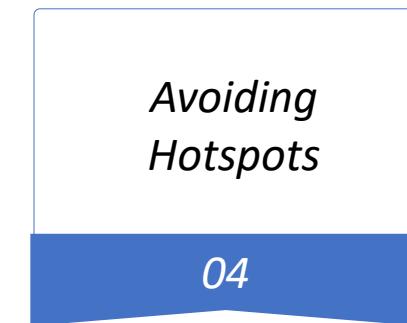
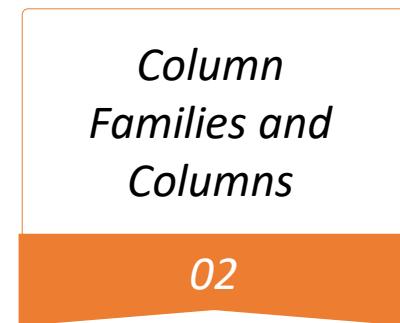
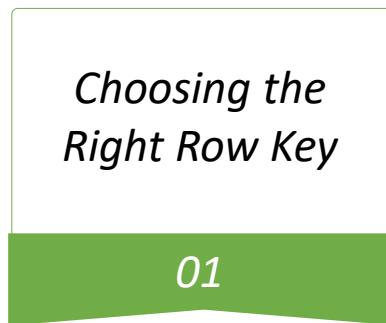
# Designing Bigtable Schema

- ✓ Designing an effective schema for Google Cloud Bigtable is crucial for achieving optimal performance, scalability, and manageability.
- ✓ Bigtable is a NoSQL, distributed database that handles large-scale workloads, and its schema design directly impacts query efficiency and cost.
- ✓ Unlike traditional relational databases, Bigtable's schema design focuses on managing data in rows and columns, with an emphasis on how data is distributed across clusters to minimise latency and maximise throughput.
- ✓ Efficient schema design in Bigtable can lead to faster access, reduced query times, and cost savings, especially when dealing with large datasets.
- ✓ By considering factors like row key design, column families, and data access patterns, users can create a schema that scales seamlessly with their data needs.

# Designing Bigtable Schema

(Continued)

- ✓ *The following are the key aspects of designing Bigtable schema:*



# Designing Bigtable Schema

## 1. Choosing the Right Row Key:

- The row key is the most important element in Bigtable schema design. It determines how data is stored and distributed across nodes.
- A well-designed row key allows for efficient queries by minimising hotspots and ensuring even data distribution. It is recommended to avoid monotonically increasing or time-based keys, as they can create uneven data distribution.

## 2. Column Families and Columns:

- Bigtable stores data in column families, which are groups of related columns. Column families should be designed based on access patterns.
- It's essential to minimise the number of column families to avoid performance bottlenecks, as each column family is stored separately. Column families can be optimised by grouping frequently accessed columns together.

# Designing Bigtable Schema

## 3. Efficient Data Access Patterns:

- When designing the schema, it's crucial to align it with the access patterns of the application. Consider the most frequent queries and ensure that the schema is optimised for those use cases.
- For example, queries that filter on specific columns should be designed with those columns in mind to reduce unnecessary read operations.

## 4. Avoiding Hotspots:

- Bigtable distributes data across multiple nodes using row keys. If many rows have the same prefix in their row keys (e.g., timestamps or sequential IDs), they may end up being stored on the same node, causing hotspots.
- To avoid this, it's important to design row keys that evenly distribute data across all nodes, such as by using randomisation or by using a combination of multiple attributes for the row key.

# Designing Bigtable Schema

## 5. Designing for Scalability and Flexibility:

- Bigtable is designed for horizontal scaling, but schema design should still account for future growth.
- Data types, access patterns, and potential schema changes should be considered.
- It's important to avoid hard-coding any assumptions in the schema to ensure scalability and flexibility as the application evolves.



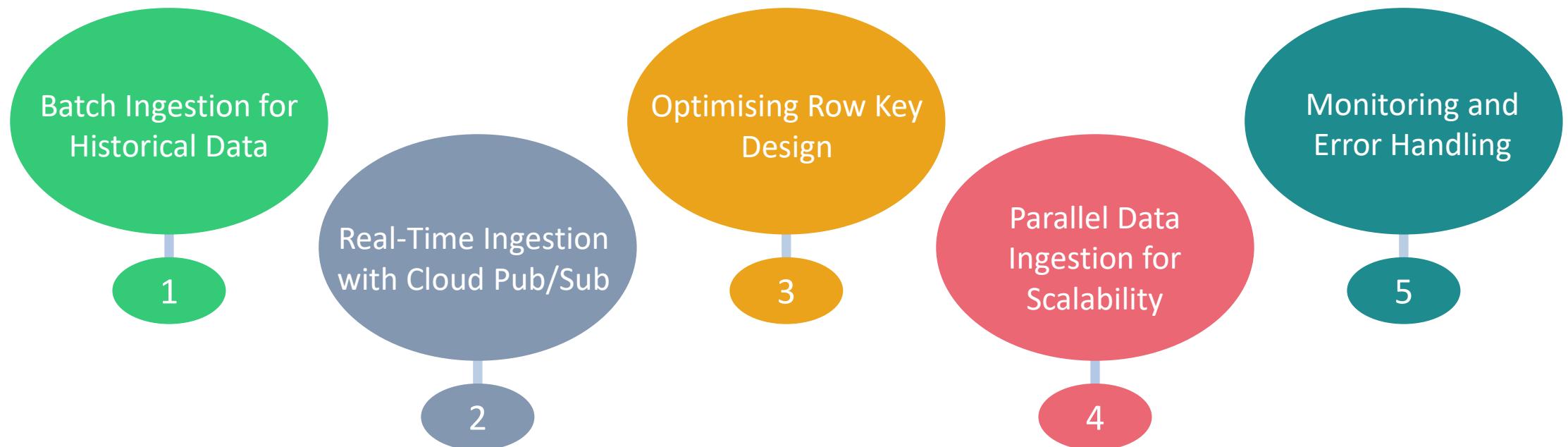
# Ingesting into Bigtable

- ✓ Ingesting data into Google Cloud Bigtable involves loading large volumes of data into the distributed NoSQL database in a scalable and efficient manner.
- ✓ Bigtable provides various methods for ingesting data, each suited for different use cases, ranging from batch loading to real-time streaming.
- ✓ The choice of ingestion method directly impacts the speed of data processing, performance, and cost.
- ✓ Understanding the best practices for data ingestion helps ensure that Bigtable is used effectively for both high-throughput and low-latency applications.
- ✓ By leveraging the appropriate ingestion techniques, users can optimise resource usage and ensure that Bigtable can handle both large datasets and time-sensitive data seamlessly.
- ✓ This flexibility allows businesses to adapt Bigtable to a variety of data workloads, from historical analysis to real-time monitoring.

# Ingesting into Bigtable

(Continued)

- ✓ *The following are the key strategies for ingesting into Bigtable:*



# Ingesting into Bigtable

## 1. Batch Ingestion for Historical Data:

- For large datasets or historical data, batch ingestion using tools like Cloud Dataflow or Apache Beam is ideal. This method processes and loads data in bulk, minimising overhead and reducing ingestion time for large volumes of data.

## 2. Real-Time Ingestion with Cloud Pub/Sub:

- For real-time data processing, Cloud Pub/Sub is a powerful solution. It enables the ingestion of streaming data, such as IoT sensor data, transaction logs, or real-time application logs, into Bigtable with minimal latency, ensuring that data is available for immediate analysis.

## 3. Optimising Row Key Design:

- Effective row key design plays a critical role in the ingestion process. By ensuring that data is distributed evenly across the Bigtable nodes, you can prevent hotspots, which can degrade performance. Strategies include using compound row keys or shuffling sequential data to achieve even distribution.

# Ingesting into Bigtable

## 4. Parallel Data Ingestion for Scalability:

- To handle large-scale ingestion, distribute the load across multiple machines or threads. Parallelising ingestion helps speed up the process, especially when dealing with high-throughput use cases.
- This can be achieved by utilising tools like Cloud Dataflow with parallel processing pipelines.

## 5. Monitoring and Error Handling:

- Implement robust monitoring systems to track data ingestion status and performance. Using tools like Stackdriver for error detection and health monitoring ensures smooth ingestion processes.
- Set up alerts for failures or bottlenecks and use retries or dead-letter queues to handle errors efficiently.

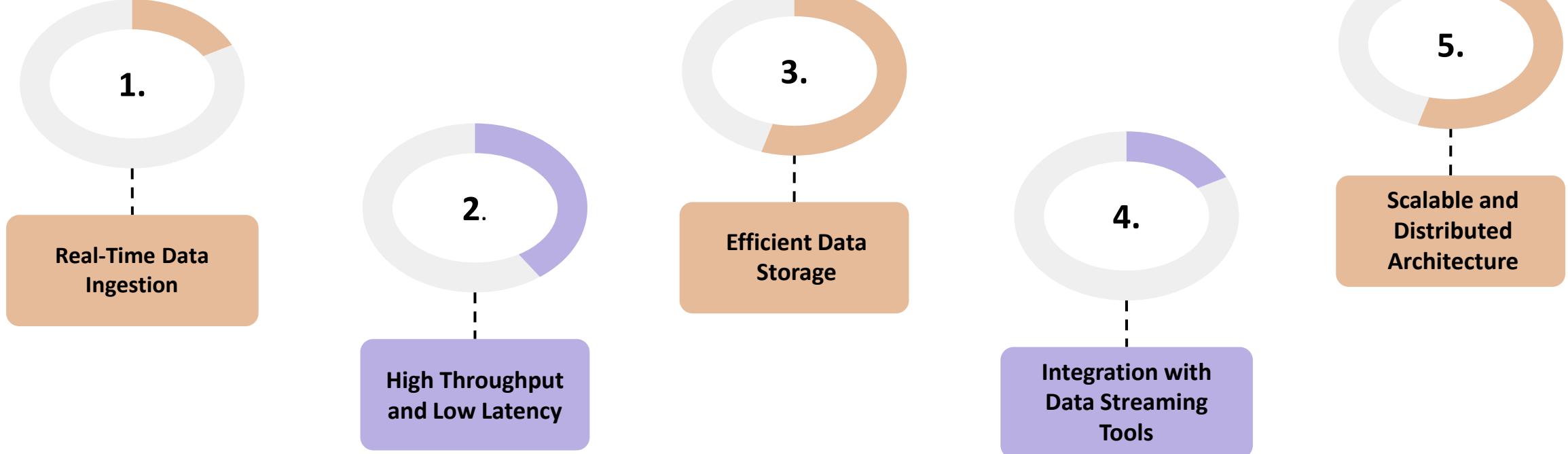
# Streaming into Bigtable

- ✓ Google Cloud Bigtable is a NoSQL, high-performance database designed for applications that require low-latency and high-throughput storage.
- ✓ It is ideal for handling massive amounts of data across a large number of rows and columns, making it perfect for time-series data, real-time analytics, and other large-scale workloads.
- ✓ Streaming into Bigtable refers to the process of continuously writing real-time data into Bigtable, which can then be processed, analysed, or used for further storage.
- ✓ Streaming data into Bigtable enables businesses to collect data from various real-time sources (e.g., sensors, IoT devices, or web traffic) and store it for immediate or future analysis.
- ✓ Bigtable's architecture allows it to handle high-throughput writes with low latency, making it an ideal choice for time-sensitive applications.



# Streaming into Bigtable

## *Key Aspects of Streaming into Bigtable*



# Streaming into Bigtable

- Real-Time Data Ingestion:** Streaming data into Bigtable enables the continuous ingestion of large volumes of real-time data, such as logs or sensor readings, with immediate availability for further processing and analysis.
- High Throughput and Low Latency:** Bigtable ensures high-throughput writes and minimal latency, making it ideal for applications that require quick and efficient storage of large-scale, time-sensitive data.
- Efficient Data Storage:** Bigtable uses a column-family format for efficient storage and retrieval, particularly beneficial for sparse datasets and time-series data where each column represents events over time.
- Integration with Data Streaming Tools:** Bigtable integrates with data streaming tools like Cloud Pub/Sub, Apache Kafka, and Dataflow, enabling seamless ingestion of real-time data streams into Bigtable for processing.
- Scalable and Distributed Architecture:** Bigtable's horizontally scalable architecture automatically splits and replicates data across nodes, ensuring efficient data handling and processing as data throughput increases.

# Module 15: Introduction to Data Engineering

- Data Science vs Data Engineering
- Data Engineering Infrastructure and Data Pipelines
- Monitoring Pipeline



# Data Science vs Data Engineering

- ✓ Data Science is the field that deals with analysing and interpreting complex data to help organisations make decisions.
- ✓ It combines skills in statistics, machine learning, data analysis, and programming to extract valuable insights from structured and unstructured data.
- ✓ Data Engineering focuses on designing, building, and maintaining the infrastructure and pipelines required for data collection, storage, and processing.
- ✓ Data engineers ensure that data is clean, accessible, and properly formatted for analysis.



# Data Science vs Data Engineering

(Continued)

- ✓ ***The following are the key roles and responsibilities:***

## 1. Data Science

- **Data Analysis:** Extract meaningful insights and patterns from data using statistical methods, machine learning models, and algorithms.
- **Model Building:** Design, train, and validate machine learning models that can be used to predict future outcomes or automate decision-making.
- **Data Exploration:** Explore data to identify trends, patterns, and relationships to inform business decisions.
- **Visualization:** Communicate data insights effectively through dashboards, charts, and presentations.

# Data Science vs Data Engineering

## 2. Data Engineer

- **Data Pipeline Design:** Create robust data pipelines that automate the collection, storage, and processing of large datasets.
- **Data Storage:** Design data storage solutions, ensuring data is stored efficiently and securely (e.g., data lakes, data warehouses).
- **Data Integration:** Integrate data from different sources (APIs, databases, external data providers) into a central system.
- **Optimisation:** Ensure that data processing is efficient and scalable, optimising the speed and cost of data operations.



# Data Science vs Data Engineering

(Continued)

- ✓ ***The following are the key differences between data science and data engineering:***

Aspect	Data Science	Data Engineering
<b>Primary Focus</b>	Extracting insights and knowledge from data	Building and managing infrastructure to support data processes
<b>Core Responsibilities</b>	Data analysis, machine learning, statistical modelling	Data pipeline construction, data storage, ETL processes
<b>Skills Required</b>	Python, R, Machine Learning, Data Analytics, Statistical Modelling	SQL, Hadoop, Spark, Kafka, ETL Tools, Data Warehousing

# Data Science vs Data Engineering

(Continued)

Aspect	Data Science	Data Engineering
Tools & Technologies	Jupyter Notebooks, TensorFlow, Scikit-learn, Pandas	Apache Spark, Apache Kafka, Google Cloud Dataflow, Airflow
Focus Area	Predictive modelling, machine learning, AI	Data processing, data storage, automation of data workflows
Typical Output	Reports, dashboards, data visualizations, predictive models	Data pipelines, databases, infrastructure for data storage

# Data Engineering Infrastructure and Data Pipelines

## ***Data Engineering Infrastructure***

- ✓ Data engineering infrastructure refers to the hardware, software, and tools used to collect, store, process, and analyse data. It involves setting up scalable systems for handling large volumes of data and ensuring the seamless flow of information across different stages of the data lifecycle.
- ✓ Effective data engineering infrastructure enables efficient data pipelines, real-time analytics, and supports machine learning and business intelligence applications.
- ✓ ***The following are the components of data engineering:***

### **1. Storage Systems:**

- **Cloud Storage:** Cloud storage services like Google Cloud Storage, Amazon S3, and Azure Blob Storage provide scalable and secure storage for large amounts of data.
- **Data Lakes:** A Data Lake is a central repository that allows you to store all structured, semi-structured, and unstructured data at any scale. Amazon S3, Azure Data Lake, and Google Cloud Storage are popular choices.

# Data Engineering Infrastructure and Data Pipelines

(Continued)

- **Databases:** Structured data is often stored in relational databases like Google BigQuery, Amazon Redshift, or SQL Server, while unstructured data might reside in NoSQL databases such as MongoDB or Cassandra.

## 2. Compute Resources:

- **Cloud Compute:** Virtual machines (VMs), clusters, and serverless computing services like Google Dataproc, AWS EC2, and Azure VMs are used for running data processing workloads.
- **Data Processing Engines:** Tools like Apache Spark, Hadoop MapReduce, and Apache Flink are commonly used for distributed data processing. Cloud services such as Google Cloud Dataflow and AWS Glue also enable serverless data processing.

## ***Data Pipelines***

- ✓ A Data Pipeline is a series of processes through which data is collected, processed, and moved from one place to another, often from data sources to storage systems, and then to analytical or operational systems.

# Data Engineering Infrastructure and Data Pipelines

## Lambda Architecture

- ✓ Lambda refers to a serverless compute service provided by cloud platforms like AWS Lambda, enabling users to run code without managing servers.
- ✓ It automatically scales depending on the volume of incoming requests and executes code in response to events such as file uploads, database changes, or HTTP requests.
- ✓ Lambda functions are stateless and are typically short-lived, making them ideal for event-driven architectures.
- ✓ This serverless model eliminates the need for provisioning or maintaining servers, allowing developers to focus on writing and deploying code efficiently

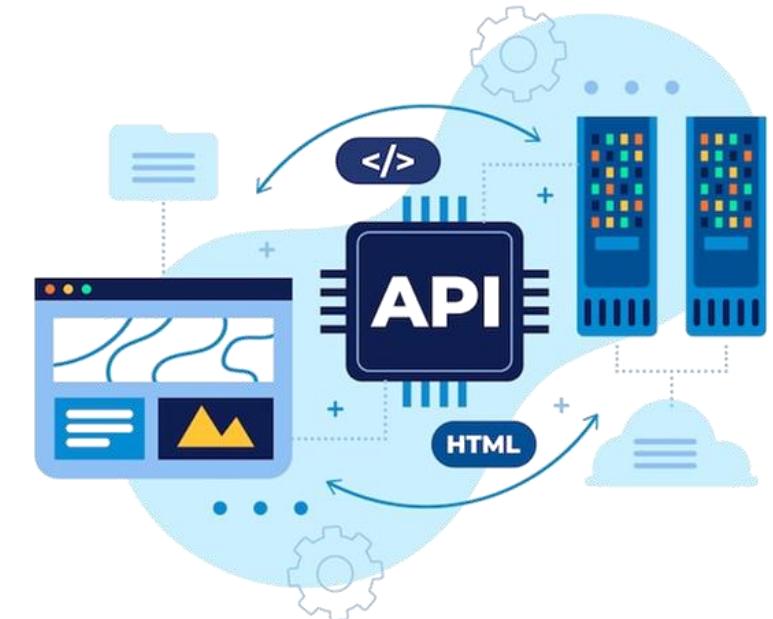


# Data Engineering Infrastructure and Data Pipelines

(Continued)

- ✓ ***The following are the layers of lambda architecture:***

- 1. Batch Layer:** Processes large datasets in batches using tools like Hadoop or Spark and stores raw data.
- 2. Speed Layer:** Handles real-time data processing using tools like Apache Kafka or Flink.
- 3. Serving Layer:** Merges the processed data from both layers and serves it for querying, typically stored in BigQuery or Hadoop HDFS.
- 4. Workflow:** Data is processed in both the batch and real-time layers simultaneously and then merged for analysis.



# Data Engineering Infrastructure and Data Pipelines

(Continued)

## ***Kappa Architecture***

- ✓ Kappa Architecture is a data processing architecture designed for real-time streaming applications, focusing on simplicity and scalability. Unlike the traditional Lambda architecture, it eliminates the need for batch processing by using a single stream processing pipeline for both real-time and historical data.
- ✓ The architecture relies on a distributed streaming platform, like Apache Kafka, to store all data in a single, immutable log. This design simplifies maintenance, ensures consistency, and supports fault-tolerant, high-throughput processing.
- ✓ ***The following are the layers of kappa architecture:***
  1. **Single Streaming Pipeline:** Processes both real-time and historical data through stream processing platforms like Apache Kafka or Google Cloud Dataflow.
  2. **Data Storage:** Stores processed data in systems like BigQuery or Cloud Storage.

# Data Engineering Infrastructure and Data Pipelines

(Continued)

## ***Streaming Big Data Architectures***

- ✓ Streaming Big Data Architectures are designed to handle real-time data processing and provide quick insights from large streams of continuously arriving data. These architectures are often used in scenarios where quick, actionable insights from live data are needed.
- ✓ ***The following are the key components of streaming big data architecture:***
  - Streaming data is continuous, real-time data generated by sources like sensors, logs, financial data, and user interactions.
  - Data Ingestion: Collects and streams data using tools like Apache Kafka, Google Cloud Pub/Sub, and Amazon Kinesis.
  - Stream Processing: Processes data in real-time with Apache Flink, Spark Streaming, or Google Cloud Dataflow.

# Monitoring Pipeline

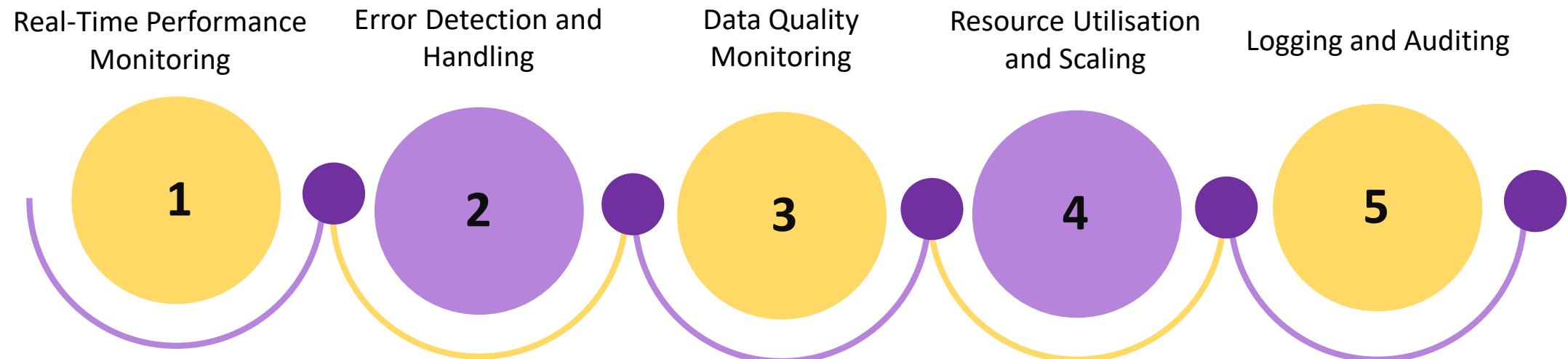
- ✓ In Data Engineering, monitoring data pipelines is essential for ensuring that data flows smoothly from sources to destinations, and that data processing tasks are executed efficiently and reliably.
- ✓ Pipeline monitoring provides visibility into the health, performance, and accuracy of data processing workflows, allowing data engineers to detect and resolve issues quickly.
- ✓ Monitoring is especially critical in environments with large-scale, real-time, or complex data flows, where potential problems (e.g., data delays, failures, or bottlenecks) can have significant business impacts.
- ✓ Effective monitoring helps to ensure that pipelines are running as expected, provides insights into system performance, and helps with troubleshooting when things go wrong.



# Monitoring Pipeline

(Continued)

- ✓ *The following are the key aspects of monitoring pipeline:*



# Monitoring Pipeline

1. **Real-Time Performance Monitoring:** Monitoring tools track the performance of pipelines in real-time, offering insights into throughput, latency, and resource usage.
2. **Error Detection and Handling:** Proactive error detection helps identify failures or issues in the pipeline early, allowing engineers to address them before they escalate.
3. **Data Quality Monitoring:** Monitoring the quality of data is essential for ensuring that only accurate and clean data flows through the pipeline. Automated checks for data integrity, completeness, and consistency can be implemented to prevent the propagation of faulty data.
4. **Resource Utilisation and Scaling:** Monitoring resource usage (e.g., CPU, memory, disk) ensures that the pipeline has adequate capacity to handle the data load.
5. **Logging and Auditing:** Detailed logs and audit trails are essential for tracking data flow and understanding the lifecycle of the data as it passes through the pipeline.

# Module 16: Big Data Tools

- Apache Hadoop
- Apache Pig Features
- Apache Hive Features



# Apache Hadoop

- ✓ Apache Hadoop is an open-source framework that allows for the distributed processing of large datasets across clusters of computers using simple programming models. It is designed to scale up from a single server to thousands of machines, offering high fault tolerance and reliability.

## ***Distributed Framework***

- ✓ **What is it?**
  - The distributed framework in Hadoop allows data to be stored across multiple machines in a network, with tasks distributed among many nodes. This parallel processing reduces the time required to process large datasets, providing scalability and fault tolerance.
- ✓ **How does it work?**
  - Hadoop's distributed framework is built on the Hadoop Distributed File System (HDFS) and the MapReduce programming model. These components work together to process large datasets efficiently in a distributed manner.

# Apache Hadoop

(Continued)

## ***HDFS (Hadoop Distributed File System)***

### ✓ **What is HDFS?**

- HDFS is a distributed file system designed to store large volumes of data across multiple nodes in a cluster. It splits files into blocks and stores them on various machines to ensure reliability and fault tolerance. It also replicates data blocks to avoid data loss.

### ✓ **How it works?**

- Data is split into fixed-size blocks (default 128MB). Each block is replicated across different nodes in the cluster to ensure redundancy. The NameNode controls the metadata of the file system, and the DataNodes store the actual data blocks.

# Apache Hadoop

(Continued)

## ***MapReduce***

### ✓ **What is MapReduce?**

- MapReduce is a programming model for processing and generating large datasets. It divides the processing into two main tasks:
  - **Map:** The input data is divided into smaller chunks, processed in parallel across the cluster.
  - **Reduce:** The output from the map tasks is aggregated and reduced to a final output.

### ✓ **How it works:**

- The Mapper takes the input and converts it into key-value pairs. The Reducer takes these pairs and processes them to produce the final result, such as aggregating data or sorting.

# Apache Hadoop

(Continued)

## ***YARN (Yet Another Resource Negotiator)***

### ✓ **What is YARN?**

- YARN is the resource management layer for Hadoop. It manages and schedules resources across the cluster, handling workloads from various applications, including MapReduce and Spark. YARN enables multiple data processing frameworks to run on top of Hadoop concurrently, making it more efficient and flexible.

### ✓ **How it works:**

- The NodeManager monitors resource usage on individual nodes and reports back to the ResourceManager.
- YARN allows for dynamic resource allocation and efficient task management, improving the overall performance of the cluster.

# Apache Pig Features

- ✓ Apache Pig is a high-level platform built on top of Hadoop for processing and analysing large datasets. It is used for the development of complex data processing workflows, abstracting the complexity of writing low-level MapReduce code.
- ✓ Pig provides a scripting language, called Pig Latin, that allows for data transformation, loading, and manipulation, enabling users to write data processing tasks in a more intuitive way.
- ✓ ***The following are the features of apache pig:***
  1. **High-Level Abstraction for Hadoop:** Pig simplifies complex MapReduce programming with Pig Latin scripts, providing a simpler syntax, especially for non-Java users.
  2. **Flexible Data Processing:** Pig handles structured, semi-structured, and unstructured data, supporting diverse formats like CSV, JSON, Avro, and HDFS files.
  3. **Extensibility:** Pig allows the creation of custom User Defined Functions (UDFs) in Java, enhancing its built-in functionality for specialised operations.

# Apache Pig Features

4. **Optimised Query Execution:** The Pig Optimiser automatically optimises queries, applying techniques like filtering and partitioning to improve performance and reduce resource use.
5. **Support for Complex Data Types:** Pig supports maps, tuples, bags, and flat files, enabling sophisticated processing of semi-structured and nested data formats.
6. **Execution Modes:** Pig can run in local mode for small datasets or in MapReduce mode on Hadoop clusters for larger data processing tasks.
7. **Interoperability with Hadoop Ecosystem:** Pig integrates with HDFS, HBase, and Hive, facilitating seamless data management and processing within the Hadoop ecosystem.
8. **Parallel Execution:** Pig leverages MapReduce to execute data transformations across multiple cluster nodes, improving scalability and speed for large datasets.
9. **Built-in Operations:** Pig provides operators for joins, filters, grouping, sorting, and aggregation, simplifying complex data manipulation tasks.

# Apache Hive Features

- ✓ Apache Hive is a data warehouse infrastructure built on top of Hadoop. It provides a high-level abstraction over Hadoop's MapReduce and enables querying of large datasets stored in Hadoop using a SQL-like language called HiveQL.
- ✓ Hive simplifies the process of querying and managing large-scale data, and it can handle structured and semi-structured data. It's widely used for data analysis, reporting, and ETL tasks.
- ✓ ***The following are the key features of apache hive as follows:***
  1. **SQL-Like Query Language (HiveQL):** HiveQL simplifies querying large datasets by offering an SQL-like syntax, making it easy for users familiar with SQL to work with Hadoop without complex MapReduce code.
  2. **Scalability:** Hive scales horizontally across a Hadoop cluster, processing large datasets and handling petabytes of data without performance degradation.

# Apache Hive Features

3. **Data Storage with HDFS:** Hive stores data in HDFS, ensuring high fault tolerance and scalability for data storage, enabling efficient access and processing.
4. **Extensibility:** Hive supports custom User Defined Functions (UDFs), allowing users to extend functionality and incorporate custom transformations within queries.
5. **Data Partitioning and Bucketing:** Partitioning and bucketing help optimise query performance by organising large datasets into manageable parts, reducing scan times.
6. **Integration with Hadoop Ecosystem:** Hive integrates with Hadoop tools like MapReduce, HBase, and Spark, facilitating seamless data processing and storage within the ecosystem.
7. **Support for Complex Data Types:** Hive supports complex data types like arrays, maps, and structs, enabling the storage and processing of semi-structured data for advanced analytics.
8. **Metadata Management:** Hive uses a metastore to efficiently store and manage metadata for tables, partitions, and schemas, simplifying data organisation and querying.

# Module 17: Apache Spark and Kafka

- Spark Components
- Spark Executions – Session
- RDD
- Spark DataFrames
- Spark Datasets
- Spark SQL
- Spark MLlib
- Spark Streaming
- Introduction to Apache Kafka



# Spark Components

- ✓ Apache Spark is a fast, in-memory data processing engine with elegant APIs for distributed computing. It was designed for large-scale data processing tasks and provides high-level APIs in Java, Scala, Python, and R.
- ✓ It can handle a variety of workloads, including batch processing, real-time streaming, machine learning, and graph processing. Apache Spark is built on top of Hadoop, and it can work with HDFS, Amazon S3, HBase, and other storage systems.

## ***Key Components***



*Spark Core*



*Spark SQL*



*Spark Streaming*



*MLlib (Machine Learning Library)*

# Spark Components

## 1. Spark Core:

- The Spark Core is the foundation of the entire Spark ecosystem. It provides the basic functionality for task scheduling, memory management, fault recovery, and interacting with storage systems.
- Spark Core manages the execution of jobs, distribution of tasks across the cluster, and scheduling tasks. It is responsible for interacting with storage systems and handling low-level operations like data shuffling and job scheduling.

## 2. Spark SQL:

- Spark SQL provides an interface for working with structured and semi-structured data using SQL queries. It allows you to run SQL queries on data stored in various formats like JSON, Parquet, Hive, and more.
- With Spark SQL, you can leverage the power of SQL to perform data analysis, filtering, and transformations on large datasets without having to write low-level code.

# Spark Components

## 3. Spark Streaming:

- Spark Streaming is an extension of Spark that allows for real-time stream processing. It ingests real-time data streams, processes them in micro-batches, and returns results with low latency.
- Spark Streaming makes it easy to process live data streams from sources like Kafka, Flume, or socket connections. It supports the same operations as batch processing, such as map, filter, and reduce, but it processes data in near real-time.

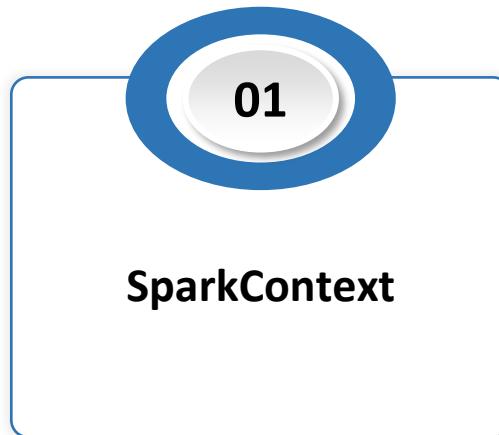
## 4. MLLib (Machine Learning Library):

- MLLib is a built-in library for machine learning in Apache Spark. It provides scalable algorithms for classification, regression, clustering, collaborative filtering, and dimensionality reduction.
- MLLib allows you to build machine learning models that can scale to large datasets. You can perform data analysis, create predictive models, and execute machine learning algorithms directly on Spark clusters, benefiting from its distributed processing power.

# Spark Executions – Session

- ✓ Apache Spark is a distributed data processing engine designed to handle large-scale data processing tasks. One of the most important concepts in Spark is the Spark Session. It serves as the entry point to work with Spark's functionalities, including reading data, transforming data, and performing computations.
- ✓ A Spark Session allows users to interact with various components of Spark such as Spark SQL, DataFrames, RDDs (Resilient Distributed Datasets), and Streaming.

**Spark session components and their functions as follows:**



# Spark Executions – Session

## 1. **SparkContext:**

- The SparkContext is responsible for connecting to a Spark cluster. It manages the resources and interacts with the cluster manager (such as YARN, Mesos, or Kubernetes). It is used for running Spark jobs on a cluster, reading data, and performing transformations on RDDs.

## 2. **SQLContext:**

- The SQLContext provides the interface to work with structured data in Spark, allowing you to execute SQL queries and work with DataFrames. It abstracts the complexity of Spark's underlying architecture and makes it easier to interact with data using SQL queries.

## 3. **HiveContext:**

- The HiveContext enables Spark to work with Apache Hive for querying and managing data in Hive. It integrates Hive's SQL capabilities into Spark, making it possible to use Hive functions, access Hive tables, and run queries within Spark jobs.

# Spark Executions – Session

## ***How Spark Session Works***

- ✓ When you create a Spark Session, it initialises the core components needed for execution, such as:
  1. **Cluster Connection:** The session connects to the cluster manager (such as YARN or Kubernetes) to allocate resources for Spark tasks.
  2. **Resource Allocation:** The session manages how resources (like CPU, memory, and storage) are allocated across the cluster for executing Spark jobs.
  3. **Data Handling:** The session allows you to load data from various sources like HDFS, S3, JDBC, and Hive, and then work with it using Spark SQL or DataFrame API.
  4. **Execution of Tasks:** The session coordinates the execution of tasks, stages, and jobs across multiple Spark workers. It schedules tasks based on the Spark DAG (Directed Acyclic Graph) and ensures the efficient execution of queries.

# RDD

- ✓ In Apache Spark, RDD (Resilient Distributed Dataset) is the fundamental data structure that represents a distributed collection of objects, which can be processed in parallel across a cluster of computers. RDDs are the core abstraction that Spark uses to distribute data and computations across nodes in a cluster.
- ✓ RDDs allow for parallel computation, which makes them suitable for large-scale data processing. They also enable users to work with low-level data manipulations, offering both fine-grained control and high-level optimisations.

## ***Key Features of RDDs***

01

**Immutable**

03

**Distributed**

02

**Fault Tolerant**

04

**In-memory Computation**

# RDD

1. **Immutable:** Once an RDD is created, it cannot be modified. Any transformation applied to an RDD results in the creation of a new RDD. This immutability property ensures fault tolerance, as previous data states are preserved in case of failure.
2. **Fault Tolerant:** If a partition of an RDD is lost (e.g., due to a node failure), Spark can recompute that partition using lineage information, which is the sequence of operations that created the RDD. This makes RDDs resilient to failures, ensuring data consistency and reliability in distributed environments.
3. **Distributed:** RDDs are partitioned across multiple nodes in a cluster. Each partition can be processed in parallel. This enables efficient data processing on large datasets, as tasks can be distributed and computed across different nodes.
4. **In-memory Computation:** RDDs can store data in memory across the cluster (in contrast to disk-based storage). This allows Spark to perform computations much faster, as accessing in-memory data is quicker than reading data from disk.

# RDD

## ***RDD Operations***

- ✓ RDDs support two types of operations:

### **1. Transformations:**

- These operations are applied to an existing RDD to produce a new RDD. Transformations are lazy, meaning they are not executed immediately.

### **2. Actions:**

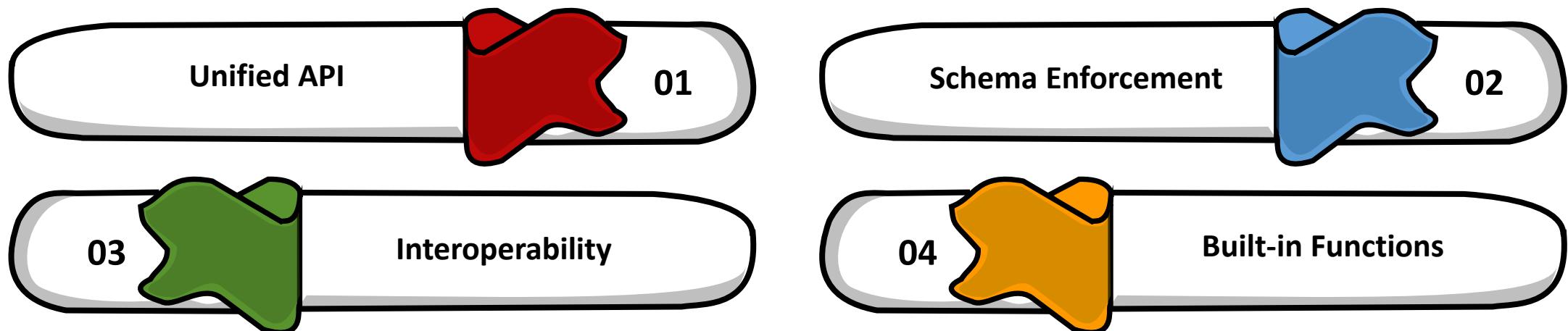
- These operations trigger the actual execution of the transformations applied on RDDs and return results.



# Spark DataFrames

- ✓ In Apache Spark, a DataFrame is a distributed collection of data organised into named columns. It is conceptually equivalent to a table in a relational database, but it can handle more complex data types such as nested structures.
- ✓ DataFrames are a key abstraction for structured data processing in Spark, providing a higher-level API for querying, transforming, and manipulating large datasets.

## *Key Features of Spark DataFrames*



# Spark DataFrames

1. **Unified API:** DataFrames in Spark are available in multiple programming languages, including **Scala**, **Python** (PySpark), and R, providing a unified interface for interacting with data across different platforms.
2. **Schema Enforcement:** DataFrames enforce schema (structure) on data, making it easy to handle structured data with defined columns and data types. Unlike RDDs, which can hold arbitrary data, DataFrames provide type safety and allow for structured operations like SQL queries.
3. **Interoperability:** DataFrames can be easily created from RDDs, HDFS, JSON, Parquet, CSV, JDBC, and Hive tables, making them highly flexible and adaptable to various data formats.
4. **Built-in Functions:** Spark DataFrames come with a rich set of built-in functions that can be used for data transformation, aggregation, filtering, and window operations, along with SQL functions for querying data.

# Spark Datasets

- ✓ In Apache Spark, a Dataset is a distributed collection of data that provides both object-oriented programming (OOP) and functional programming features.
- ✓ Datasets are an abstraction layer over DataFrames and are designed to provide the best of both RDDs and DataFrames.
- ✓ DataFrames are essentially Datasets with a specific type of data (i.e., rows and columns, like a table in a relational database).
- ✓ Datasets provide the strong typing and compile-time type safety that RDDs offer, along with the optimisations available in DataFrames (like Catalyst Optimiser for query optimisation).
- ✓ Datasets were introduced in Spark 1.6 to bridge the gap between RDDs and DataFrames, giving developers the ability to perform operations in a type-safe manner while still benefiting from Spark SQL optimisations.

# Spark Datasets

## *Key Advantages of Datasets*

1. **Type-Safety:** Unlike DataFrames, Datasets maintain type safety, meaning you can catch errors at compile time.
2. **Interoperability:** Datasets provide a combination of both the functional programming style of RDDs and the declarative query style of DataFrames.
3. **Optimised Execution:** Datasets benefit from Catalyst Optimiser, the query optimisation engine used by DataFrames, ensuring better performance.



# Spark Datasets

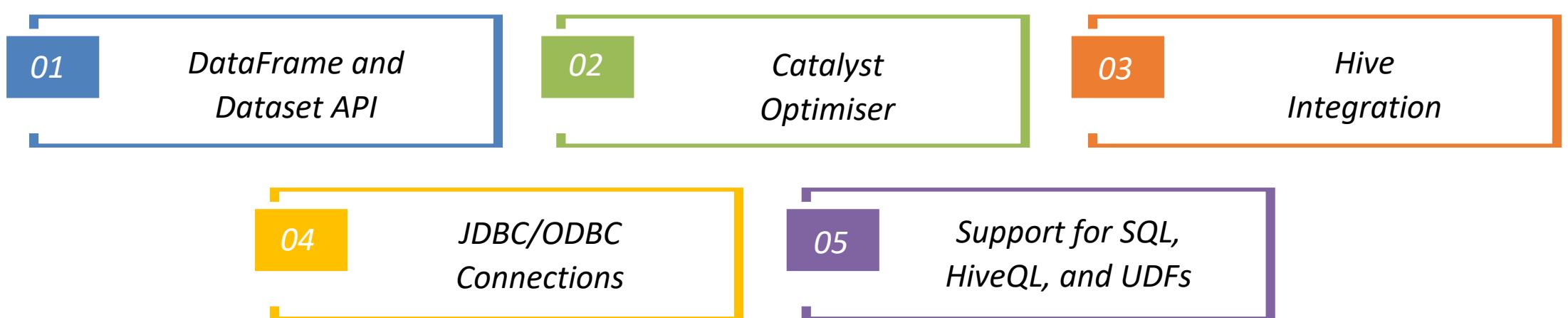
## ***Key Features of Datasets***

1. **Strong Typing:** Datasets offer the ability to specify the type of data (e.g., String, Integer) at compile time, making the code more type-safe and reducing runtime errors.
2. **Interoperability with DataFrames:** Datasets are an extension of DataFrames. They provide a high-level API that can be easily converted to DataFrames and vice versa, enabling seamless transitions between structured and unstructured data processing.
3. **Spark SQL Optimisation:** Like DataFrames, Datasets are optimised using Catalyst Optimiser and Tungsten execution engine, which improves performance and execution efficiency.
4. **Works Well with Complex Types:** Datasets support complex types like Array, Map, and Struct, enabling easier handling of nested data structures.

# Spark SQL

- ✓ Spark SQL is a component of Apache Spark that allows you to run SQL queries on structured data. It provides an interface for querying data using SQL, as well as the ability to execute SQL-like operations on DataFrames and Datasets.
- ✓ Spark SQL enables seamless integration between structured data and Spark's powerful data processing capabilities. It leverages the Catalyst Optimiser for query optimisation, which improves the performance of queries, and the Tungsten execution engine for efficient physical execution.

## ***Key Features of Spark SQL***



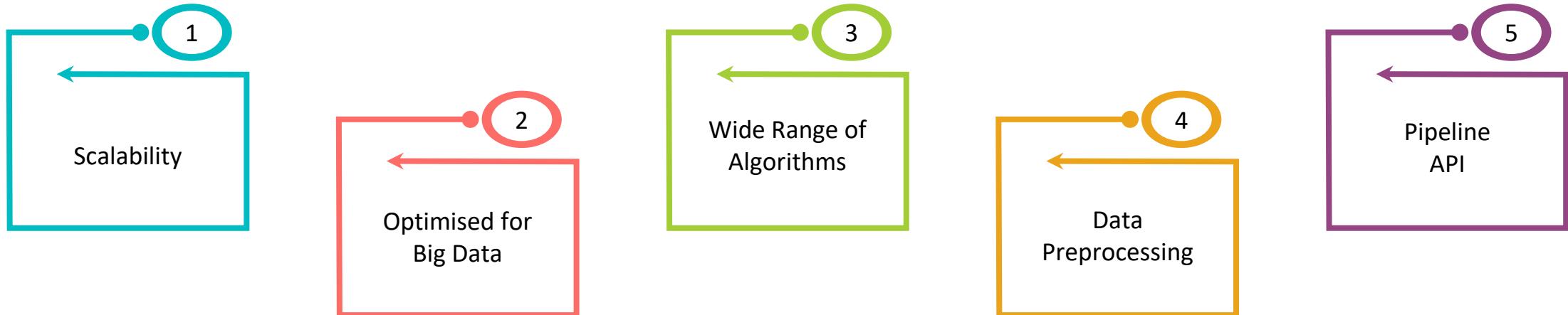
# Spark SQL

1. **DataFrame and Dataset API:** The DataFrame API and Dataset API in Spark are both built on top of Spark SQL. These APIs allow users to perform transformations and queries on distributed data while leveraging SQL capabilities for querying.
2. **Catalyst Optimiser:** The Catalyst Optimiser is a query optimisation engine used by Spark SQL. It applies various optimisation techniques like predicate pushdown, constant folding, and join reordering to optimise queries and improve performance.
3. **Hive Integration:** Spark SQL can read and write data to Hive, which makes it compatible with existing Hive tables and allows users to run HiveQL queries within Spark.
4. **JDBC/ODBC Connections:** Spark SQL supports connections via JDBC and ODBC, which allows for easy integration with traditional databases.
5. **Support for SQL, HiveQL, and UDFs:** Spark SQL supports standard SQL queries, HiveQL queries, and User Defined Functions (UDFs), enabling custom transformations and logic within queries.

# Spark MLlib

- ✓ Spark MLlib is Apache Spark's scalable machine learning library, designed to simplify the creation of machine learning workflows at scale.
- ✓ It provides a wide range of algorithms and tools for both classification, regression, clustering, and other machine learning tasks. The main objective of Spark MLlib is to allow you to work with big data and apply machine learning models efficiently.

## ***Key Features of Spark MLlib***

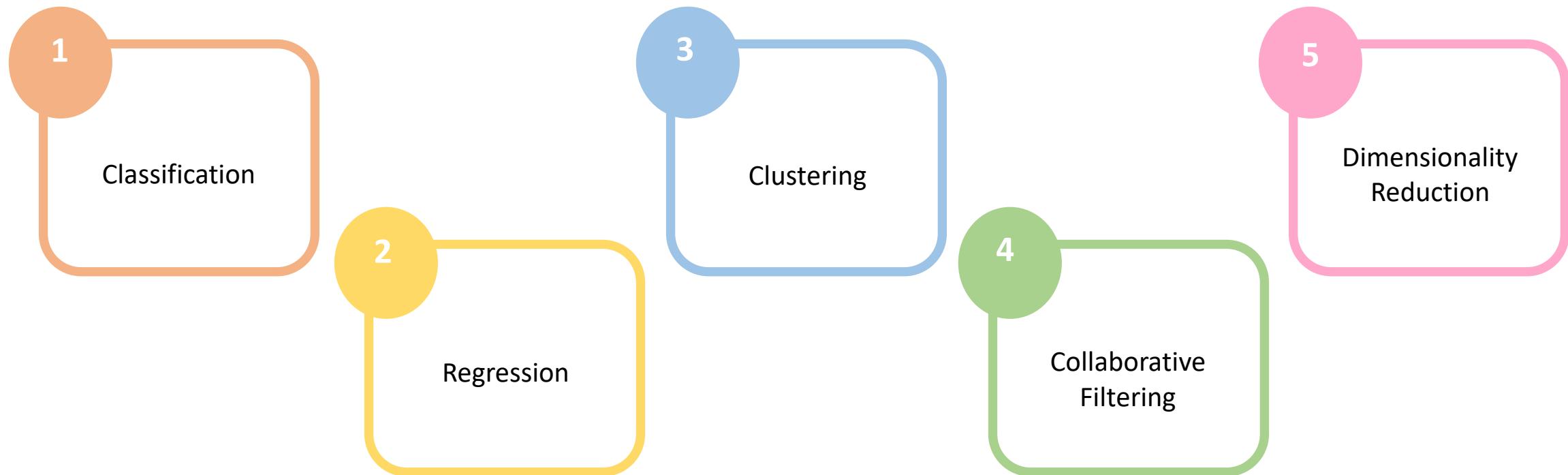


# Spark MLlib

1. **Scalability:** MLlib can handle large datasets by distributing the computations across a cluster.
2. **Optimised for Big Data:** Works seamlessly with large datasets in a distributed environment.
3. **Wide Range of Algorithms:** Offers algorithms for classification, regression, clustering, recommendation, and more.
4. **Data Preprocessing:** Includes tools for data transformation, such as normalisation, scaling, encoding, etc.
5. **Pipeline API:** A high-level API for building machine learning workflows, enabling easier tuning and model deployment.

# Spark MLlib

## ***Spark MLlib Components***



# Spark MLlib

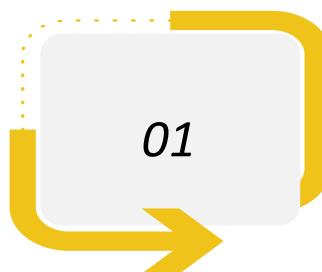
1. **Classification:** Algorithms like Logistic Regression, Decision Trees, Random Forests, Naive Bayes, and Gradient Boosting for predicting discrete labels.
2. **Regression:** Algorithms like Linear Regression, Decision Trees, Random Forests, and Gradient Boosting for predicting continuous values.
3. **Clustering:** Algorithms like K-means and Gaussian Mixture Models (GMM) for grouping similar data points.
4. **Collaborative Filtering:** Techniques like ALS (Alternating Least Squares) for building recommendation systems.
5. **Dimensionality Reduction:** Techniques like Principal Component Analysis (PCA) for reducing feature space.



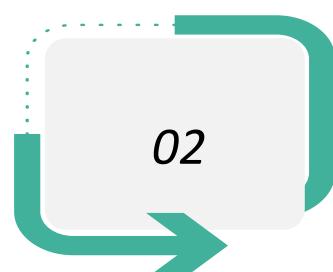
# Spark Streaming

- ✓ Spark Streaming is an extension of Apache Spark that allows you to process and analyse real-time streaming data.
- ✓ It processes data in small batches (micro-batches) and enables real-time analytics using the full capabilities of Spark's core architecture.
- ✓ With Spark Streaming, you can ingest data from sources like Kafka, HDFS, Flume, Kinesis, and TCP/IP sockets, and then perform operations like aggregation, filtering, and windowing on the streaming data.

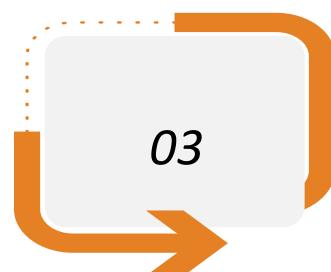
## *Key Aspects of Spark Streaming*



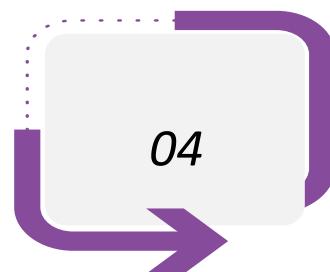
*Micro-Batch Processing*



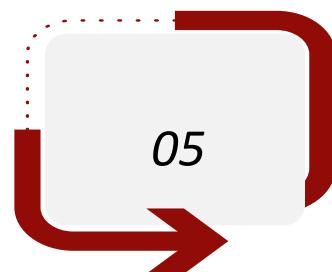
*Dstreams*



*Stream Processing with Spark SQL*



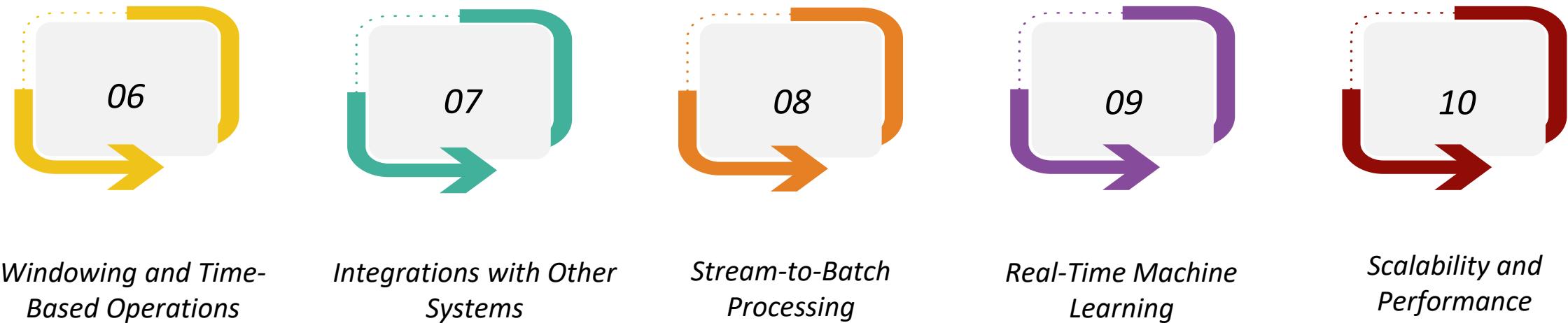
*Fault Tolerance and Reliability*



*Stateful Stream Processing*

# Spark Streaming

(Continued)



# Spark Streaming

## 1. Micro-Batch Processing:

- Spark Streaming processes streaming data in small micro-batches (typically from 1 second to 60 seconds).
- This approach provides a balance between the low latency of true stream processing and the reliability of batch processing.
- Data is collected in batches and processed through the same Spark engine that handles batch data.

## 2. Dstreams:

- A DStream (Discretized Stream) is the basic abstraction in Spark Streaming. It represents a continuous stream of data and can be created from various data sources (like Kafka, HDFS, etc.)

# Spark Streaming

## 3. Stream Processing with Spark SQL:

- With Spark SQL, you can process streaming data using SQL queries.
- Structured Streaming (introduced in Spark 2.0) provides a high-level API to work with streaming data as DataFrames and Datasets, allowing for SQL-like querying of streams.

## 4. Fault Tolerance and Reliability:

- **Checkpointing:** Spark Streaming ensures fault tolerance by periodically saving the state of the DStream. If a failure occurs, the stream can be recovered from the last checkpoint.
- **Replication:** DStreams can be replicated across multiple nodes in the cluster to ensure data redundancy and avoid data loss.
- **Exactly Once Semantics:** This ensures that each record is processed only once, avoiding duplicate processing in case of failures.

# Spark Streaming

## 5. Stateful Stream Processing:

- Spark Streaming supports stateful transformations, such as windowing and tracking the state of data over time (e.g., for applications like fraud detection or real-time monitoring).
- `MapWithState` and `UpdateStateByKey` are commonly used operations for stateful stream processing.

## 6. Windowing and Time-Based Operations:

- Spark Streaming allows users to define windowed operations that apply to data over a specific time period (e.g., the last 5 minutes).
- This feature is useful for scenarios such as sliding windows, where you need to perform aggregation or computation on recent data.

# Spark Streaming

**7. Integrations with Other Systems:** Spark Streaming integrates seamlessly with popular data sources like:

- **Kafka** for event-driven streaming.
- **Amazon Kinesis** for cloud-based data streams.
- **HDFS** and **HBase** for distributed storage.
- **Flume** for log data collection.
- **TCP sockets** for custom data sources.

**8. Stream-to-Batch Processing:**

- Structured Streaming provides a unified batch and streaming API, which makes it easier to write applications that process both batch and real-time data with the same codebase

# Spark Streaming

## 9. Real-Time Machine Learning:

- Spark Streaming integrates with MLlib, allowing real-time training and prediction of machine learning models on streaming data.
- For example, you can apply a trained model to incoming data for real-time classification, recommendation, or anomaly detection.

## 10. Scalability and Performance:

- Spark Streaming can scale to handle massive streams of data by distributing the load across multiple nodes in a cluster.
- It can handle data with very high throughput and perform complex operations in real-time with low latency

# Introduction to Apache Kafka

- ✓ Apache Kafka is an open-source distributed event streaming platform used for building real-time data pipelines and streaming applications.
- ✓ It was originally developed by LinkedIn and later open-sourced as part of the Apache Software Foundation. Kafka is designed to handle high throughput, fault tolerance, scalability, and low-latency communication for large-scale distributed systems.
- ✓ Kafka is primarily used for real-time data streaming and is commonly employed in use cases like event-driven architecture, data integration, and real-time analytics.
- ✓ It allows for the publication, subscription, storage, and processing of streams of records in real time. Kafka's architecture is designed to handle vast amounts of data while ensuring reliability and scalability.



# Module 18: Introduction to Apache Airflow

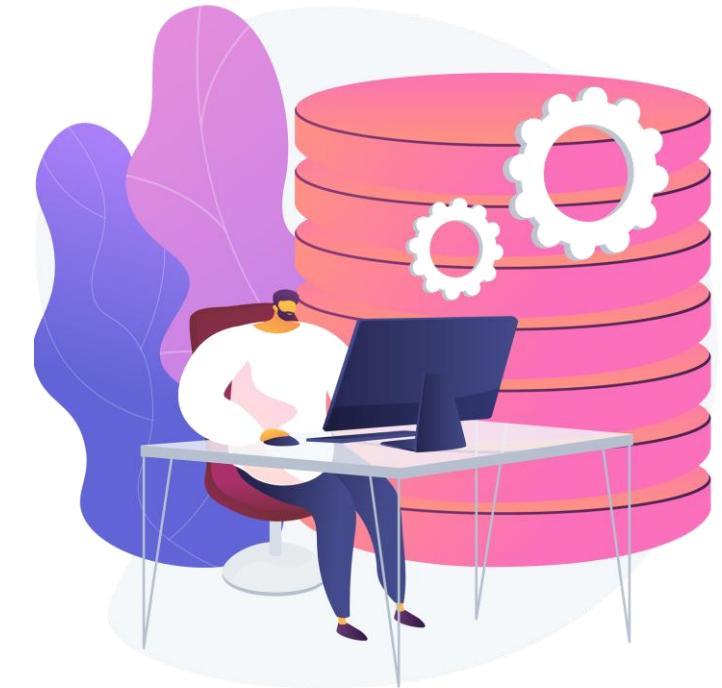
- Building Pipelines in Apache Airflow
- Production Data Pipeline



# Building Pipelines in Apache Airflow

## *Understanding Data Pipelines*

- ✓ In Apache Airflow, a pipeline is a directed acyclic graph (DAG) that defines how data flows through a sequence of tasks.
- ✓ Each task represents a discrete unit of work such as extracting data, transforming it, or loading it into a destination.
- ✓ Pipelines allow engineers to manage dependencies and execution order clearly, ensuring complex workflows run reliably.



# Building Pipelines in Apache Airflow

## ***Defining a DAG***

- ✓ A DAG is the backbone of any pipeline in Airflow. It is created using Python scripts, where you specify:
  - DAG ID and schedule (e.g., hourly, daily, custom cron).
  - Default arguments such as retries, start dates, or timeouts.
  - Operators and tasks that form the building blocks of the workflow.

## ***Tasks and Operators***

- ✓ Pipelines are constructed using operators, which define the type of work to be done. Common operators include:
  - BashOperator for executing shell commands.
  - PythonOperator for running Python functions.
  - Sensors for waiting on certain conditions or external events.

# Building Pipelines in Apache Airflow

## ***Pipeline Scheduling***

- ✓ Airflow pipelines are typically scheduled at regular intervals. The scheduling mechanism is highly flexible:
  - Cron expressions allow precise time-based schedules.
  - Catch-up mechanism lets Airflow rerun pipelines for past dates if required.
  - This scheduling ensures pipelines align with business or data refresh cycles.

## ***Monitoring Pipelines***

- ✓ Once built, pipelines are monitored via the Airflow UI. Users can check task statuses (success, failed, running, skipped), logs, and execution history. This transparency is vital for debugging and improving pipeline reliability.

# Building Pipelines in Apache Airflow

(Continued)

- ✓ *The following are the steps for building pipelines:*

**Step 1:** In one terminal, run the commands and make sure airflow is running. Keep this terminal open. The UI is at <http://localhost:8080>

```
training@training:~$ source ~/airflow_venv/bin/activate
(airflow_venv) training@training:~$
(airflow_venv) training@training:~$ airflow standalone
standalone | Starting Airflow Standalone
standalone | Checking database is initialized
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
WARNI [unusual_prefix_cf3733d39fa1a8d6b1e3a19f83ba7f7ab1289116_example_python_operator] The virtualenv_python example task requires virtualenv, please install it
.
```

The screenshot shows the Apache Airflow web interface at [localhost:8080/home](http://localhost:8080/home). The top navigation bar includes links for Airflow, DAGs, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. A timestamp of 06:35 UTC is shown on the right. Below the header, there are two yellow warning messages: "Do not use SQLite as metadata DB in production – it should only be used for dev/testing. We recommend using Postgres or MySQL. Click here for more information." and "Do not use the SequentialExecutor in production. Click here for more information." The main section is titled "DAGs" and displays a table of five DAG entries. Each entry includes a thumbnail icon, the DAG name, owner, number of runs, schedule, last run, next run, and recent tasks. The "All" tab is selected, showing 54 DAGs in total, with 0 Active and 54 Paused.

All	Active	Paused
54	0	54

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
Params Trigger UI	[example] params	airflow	None			
Params UI tutorial	[example] params	airflow	None			
Sample DAG with Display Name	[example]	airflow	None			
conditional_dataset_and_time_based_timetable	[dataset-time-based-timetable]	airflow	Dataset or 0 1 * * 3	2025-08-13, 01:00:00		

# Building Pipelines in Apache Airflow

**Step 2:** Open a second terminal, activate your venv, and create a DAG

**Step 3:** Enter this **minimal ETL pipeline** (uses the @task API so no extra providers needed). Save & exit (Ctrl+O, Enter, Ctrl+X)

```
training@training: ~
training@training:~$ source ~/airflow_venv/bin/activate
(airflow_venv) training@training:~$ 
(airflow_venv) training@training:~$ mkdir -p ~/airflow/dags
(airflow_venv) training@training:~$ 
(airflow_venv) training@training:~$ nano ~/airflow/dags/pipeline_demo.py
(airflow_venv) training@training:~$ 
```

```
training@training: ~
GNU nano 4.8      /home/training/airflow/dags/pipeline_demo.py  Modified

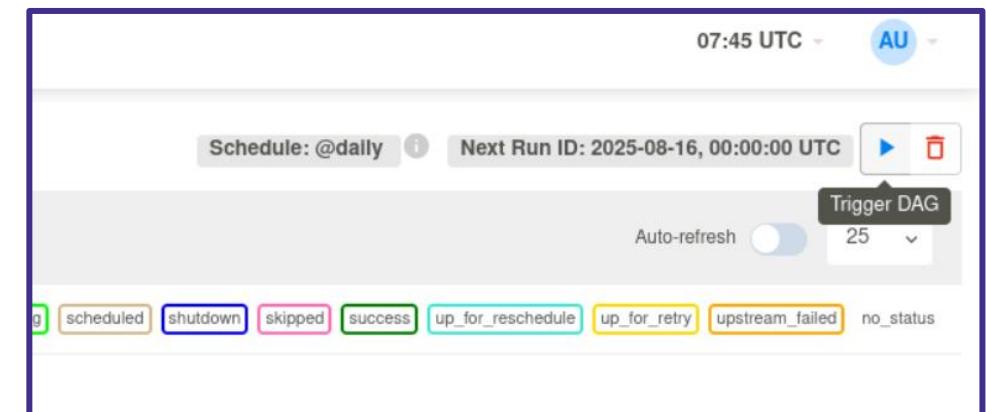
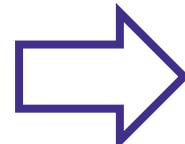
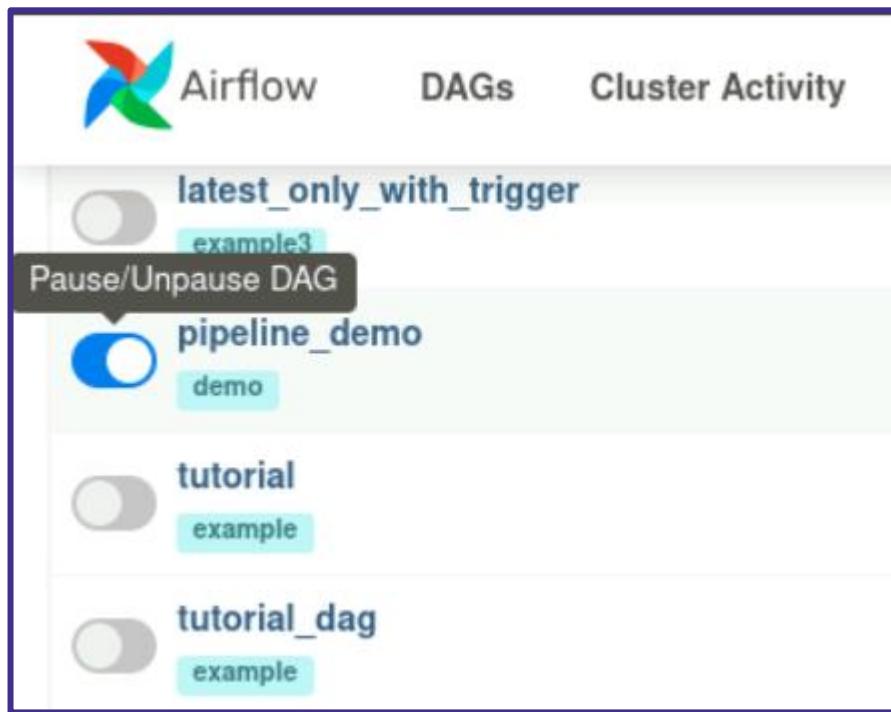
from datetime import datetime, timedelta
from airflow import DAG
from airflow.decorators import task

default_args = {
    "owner": "you",
    "retries": 2,
    "retry_delay": timedelta(minutes=2),
}
```

# Building Pipelines in Apache Airflow

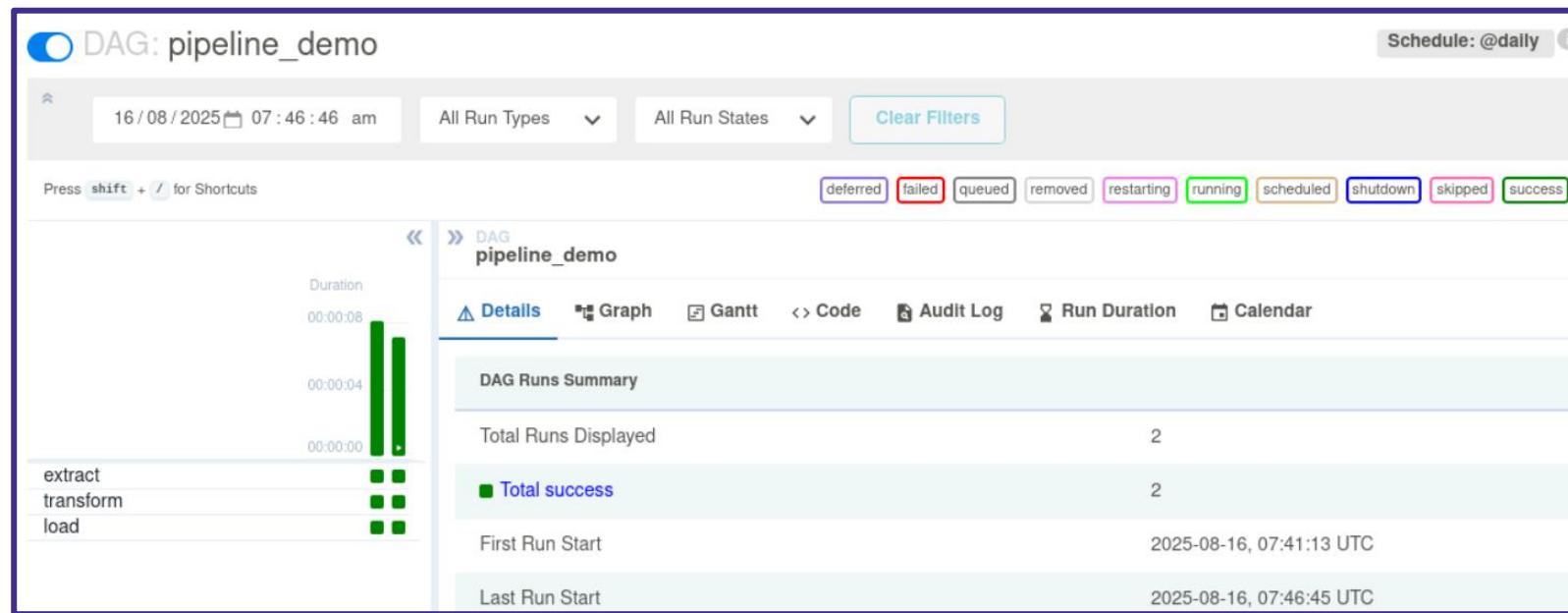
**Step 4:** Open the UI → DAGs. Toggle **pipeline\_demo** to On

**Step 5:** Click on the DAG to open it and then click on Trigger DAG button



# Building Pipelines in Apache Airflow

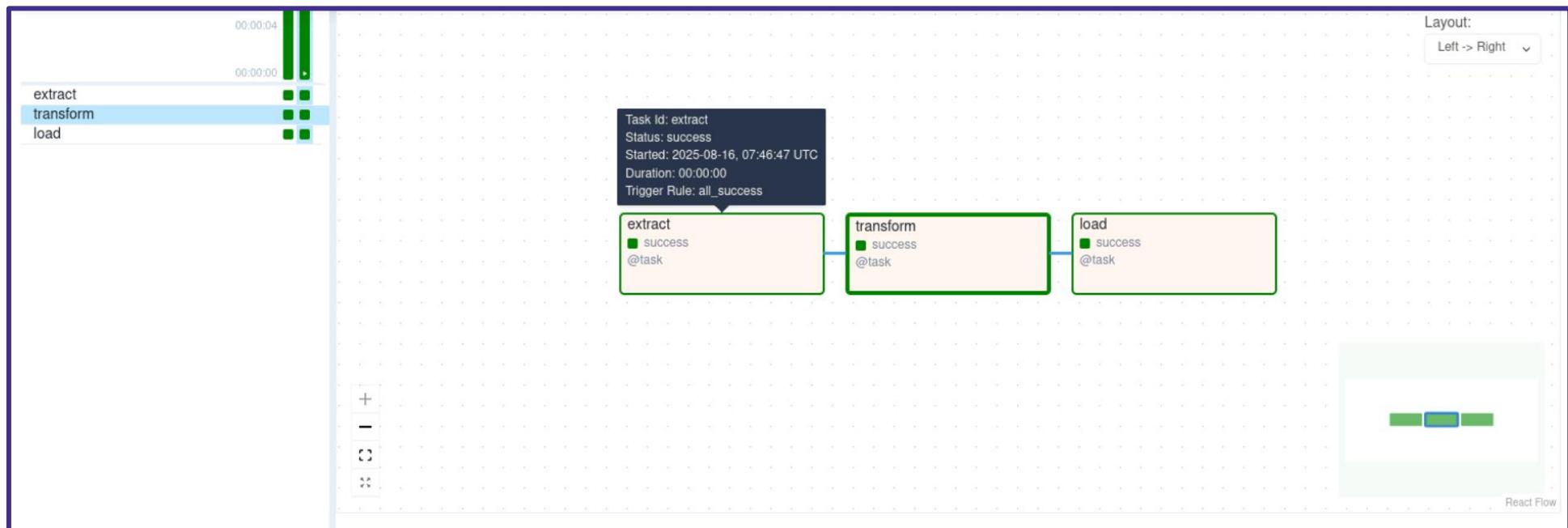
## Step 6: Go to Graph View



# Building Pipelines in Apache Airflow

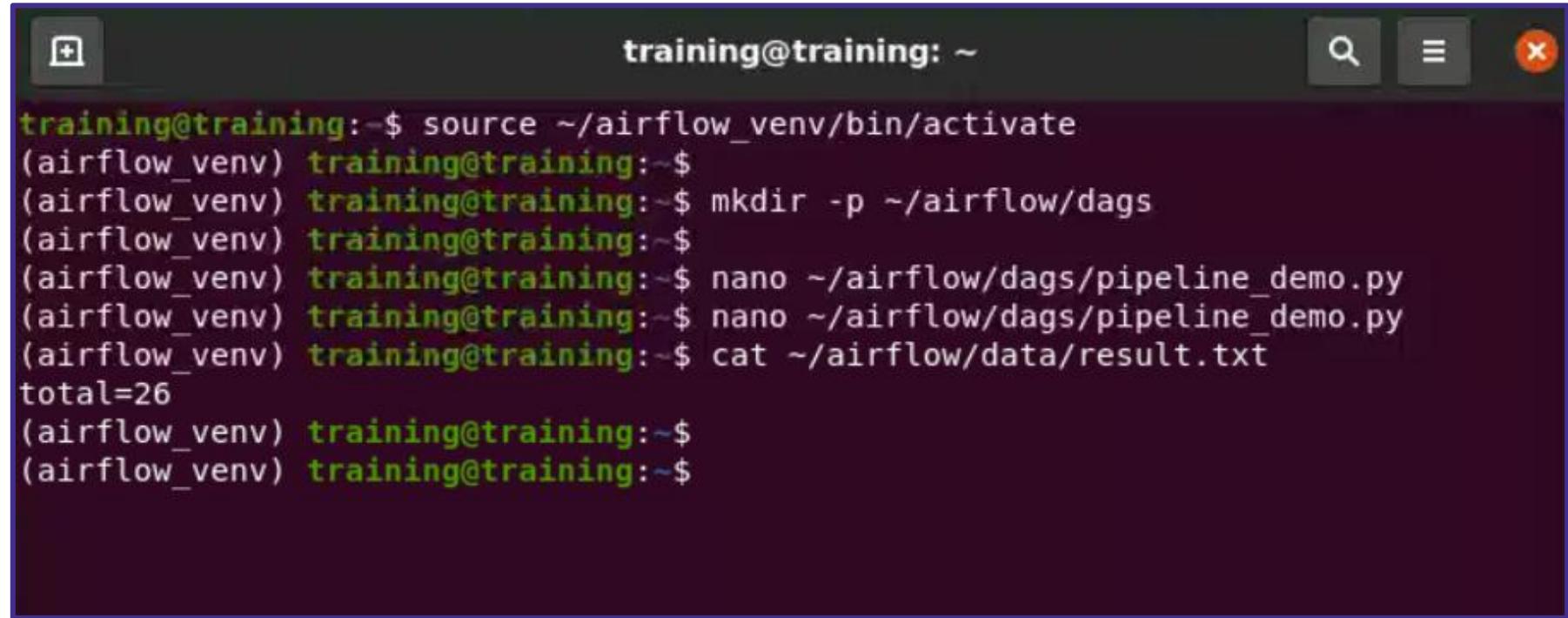
(Continued)

- ✓ The tasks (extract → transform → load) will change from grey → **running** → **green**. Click each task → **Log** to see details and outputs.



# Building Pipelines in Apache Airflow

**Step 7:** In a terminal (with venv active), check the output file



```
training@training: ~
training@training:~$ source ~/airflow_venv/bin/activate
(airflow_venv) training@training:~$ mkdir -p ~/airflow/dags
(airflow_venv) training@training:~$ nano ~/airflow/dags/pipeline_demo.py
(airflow_venv) training@training:~$ nano ~/airflow/dags/pipeline_demo.py
(airflow_venv) training@training:~$ cat ~/airflow/data/result.txt
total=26
(airflow_venv) training@training:~$
(airflow_venv) training@training:~$
```

# Production Data Pipeline

- ✓ A production data pipeline is a fully automated system that moves and processes data from one or more sources to a destination for further analysis or action. These pipelines typically handle large volumes of data and must operate without interruption or failure.

## ***Key Characteristics of a Production Data Pipeline***

1. **Automated:** The entire process (from ingestion to transformation and storage) is automated with minimal human intervention.
2. **Scalable:** The pipeline can handle increasing amounts of data as the business grows.
3. **Fault Tolerant:** It includes mechanisms for error handling and recovery in case of failures.
4. **Monitored:** Monitoring systems are set up to alert teams when something goes wrong.

# Production Data Pipeline

## *Components of a Production Data Pipeline*

1. **Data Sources:** These are the places where data is coming from, such as databases, APIs, files, streaming services, etc.
2. **Data Ingestion:** The process of collecting data from different sources and loading it into a staging area or directly into processing.
3. **Data Processing:** This includes transformations, cleaning, validation, and other operations on the data, often using ETL (Extract, Transform, Load) processes.
4. **Data Storage:** The processed data is stored in data warehouses, data lakes, or other storage systems for further analysis or use.
5. **Orchestration/Automation:** Tools like Apache Airflow are used to automate the entire pipeline, ensuring that tasks run in the correct order and at the appropriate times.

# Module 19: Data Lakes and Data Warehouses

- OLTP vs OLAP
- Databases vs Data Lakes vs Data Warehouses
- Data Lakehouse
- Data Fabric, Data Mesh, Data Mart, Delta Lake
- Choosing the Right Storage Option
- Data Lake Cloud Offerings
- Cloud Data Warehouse Services



# OLTP vs OLAP

- ✓ ***The following are the key difference between OLTP and OLAP:***

Aspect	OLTP (Online Transaction Processing)	OLAP (Online Analytical Processing)
Purpose	Designed for managing real-time transactional data.	Designed for analytical querying and reporting.
Data Structure	Data is stored in a highly normalised form (usually relational).	Data is stored in a multidimensional format (star/snowflake schema).
Use Cases	E-commerce, banking, customer relationship management (CRM), etc.	Business intelligence, data analysis, decision support, etc.
Transactions	Handles day-to-day operations with frequent insert, update, and delete operations.	Handles complex queries for decision-making with limited write operations.
Data Volume	Smaller data size per transaction, but large numbers of transactions.	Larger volumes of historical data for analysis.

# OLTP vs OLAP

(Continued)

- ✓ ***The following are the key difference between OLTP and OLAP:***

Aspect	OLTP (Online Transaction Processing)	OLAP (Online Analytical Processing)
<b>Query Complexity</b>	Simple queries with low computational complexity (e.g., SELECT, INSERT, UPDATE).	Complex queries with high computational complexity (e.g., aggregations, multi-dimensional analysis).
<b>Response Time</b>	Fast response times for transactional queries.	Slower response times due to the complexity of analytical queries.
<b>Data Integrity</b>	High focus on data accuracy and integrity in real-time.	Focuses on summarising and aggregating data for analysis, may involve approximations.
<b>Read/Write Ratio</b>	High write-to-read ratio (frequent updates to database records).	High read-to-write ratio (queries are more frequent than updates).
<b>Data Source</b>	Real-time data from user actions or business operations.	Historical data aggregated over time for reporting and analysis.
<b>Examples</b>	Transaction-based systems like ATM, order entry systems, etc.	Data warehouses, BI tools, reporting dashboards, etc.

# Databases vs Data Lakes vs Data Warehouses

- ✓ *The following are the key difference between Databases, Data Lakes and Data Warehouses:*

Aspect	Databases	Data Lakes	Data Warehouses
Purpose	Designed for managing structured transactional data.	Designed to store large volumes of raw, unstructured, and semi-structured data.	Designed for analytical querying and business intelligence.
Data Structure	Primarily structured data, stored in rows and columns (tables).	Raw data in its native format (e.g., JSON, XML, Parquet, text files).	Structured data optimised for analysis (e.g., star or snowflake schema).
Data Types	Structured data (tables with rows and columns).	Structured, semi-structured, and unstructured data (e.g., logs, social media, IoT).	Primarily structured and processed data, often from OLTP systems.
Data Storage	Data is usually stored in relational databases (e.g., MySQL, PostgreSQL).	Data is stored in its raw format without transformation (e.g., Hadoop, Amazon S3).	Data is pre-processed, cleaned, and optimised for queries (e.g., Redshift, Google BigQuery).

# Databases vs Data Lakes vs Data Warehouses

(Continued)

Aspect	Databases	Data Lakes	Data Warehouses
<b>Use Cases</b>	Day-to-day operations (e.g., transactional systems, CRM).	Big data analytics, machine learning, real-time data processing.	Data analysis, reporting, business intelligence, decision support.
<b>Data Loading</b>	Fast and frequent transactional data insertions (write-heavy).	Batch and real-time data ingestion, including large-scale unstructured data.	Data is loaded in batches after transformation from various sources.
<b>Query Complexity</b>	Simple queries focused on real-time transactions (e.g., SELECT, INSERT, UPDATE).	Complex queries for big data processing, often using distributed systems (e.g., MapReduce, Spark).	Complex analytical queries with high computational overhead (e.g., aggregations, data mining).
<b>Performance</b>	Optimised for fast transactional read/write operations.	Performance can vary depending on data size and processing tools.	Optimised for fast querying and reporting on large datasets.

# Data Lakehouse

- ✓ A Data Lakehouse is an emerging architecture that combines the best features of Data Lakes and Data Warehouses. It leverages the scalability and flexibility of a Data Lake with the data management and performance optimisation of a Data Warehouse.
- ✓ A data lakehouse allows organisations to store all their structured, semi-structured, and unstructured data in a single platform, providing both analytical and operational processing in a unified system.



Data Warehouse

Data Lake

Lake House

# Data Lakehouse

## ***Key Aspects of Data Lakehouse***

### **1. Unified Data Platform:**

- A Data Lakehouse combines the benefits of both data lakes and data warehouses. It allows organisations to store all types of data—structured, semi-structured, and unstructured—in a unified platform.
- This makes it easier to access and analyse all data, whether it's transactional (structured) or raw (unstructured), without having to move data between multiple systems.

### **2. Schema-on-Read and Schema-on-Write:**

- A Data Lakehouse supports both schema-on-read (like data lakes) and schema-on-write (like data warehouses). This allows users to ingest raw data (schema-on-read) and apply structured schemas for querying and analysis as needed (schema-on-write).

# Data Lakehouse

## 3. Cost-Effectiveness:

- Data Lakehouses are typically more cost-effective than traditional data warehouses, as they combine the low-cost storage of data lakes (e.g., Hadoop or cloud storage like Amazon S3) with the efficiency and performance of data warehouses.
- This reduces the need for data duplication and the cost of moving data between storage systems.

## 4. Performance and Query Optimisation:

- While Data Lakes can be inefficient for fast analytics, Data Lakehouses address this by integrating advanced query optimisation techniques, indexing, and data management.
- They leverage technologies like Delta Lake, Apache Hudi, or Apache Iceberg, which allow for ACID transactions, data versioning, and time travel queries for better consistency and performance.

# Data Lakehouse

## 5. Advanced Analytics and Machine Learning:

- Data Lakehouses support both operational analytics (e.g., business intelligence) and advanced analytics (e.g., machine learning, deep learning). By storing both structured and unstructured data in one place, they enable data scientists to perform machine learning and predictive analytics directly on the data lakehouse.
- This reduces data silos and makes it easier to analyse all data for insights, model training, and predictive modeling.

## 6. Real-Time Data Processing:

- Data lakehouses can handle real-time streaming data, enabling businesses to perform real-time analytics and make immediate, data-driven decisions.
- This makes data lakehouses well-suited for use cases like fraud detection, recommendation systems, and real-time monitoring.

# Data Lakehouse

## 7. Data Governance and Security:

- One of the challenges of Data Lakes is managing data quality and governance. Data Lakehouses provide built-in governance features like data auditing, metadata management, and data lineage, similar to traditional data warehouses.
- This ensures that the data stored is consistent, secure, and compliant with regulations, while also providing the flexibility of a data lake.

## 8. Integration with Existing Data Systems:

- Data lakehouses are designed to integrate seamlessly with existing systems in the data stack, such as ETL tools, BI tools, and data warehousing solutions.
- This helps organisations continue to use their existing workflows and tools without requiring major changes to their infrastructure.

# Data Lakehouse

## 9. Scalability:

- Just like Data Lakes, Data Lakehouses can scale to handle huge volumes of data, thanks to the distributed nature of the architecture.
- This scalability is critical as data continues to grow in size and complexity, especially for big data and machine learning applications.

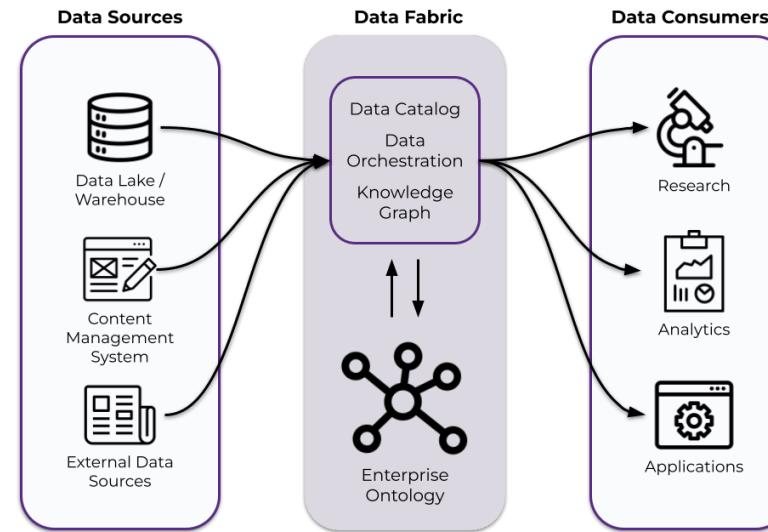
## 10. ACID Transactions and Time Travel:

- ACID transactions ensure that the data remains consistent, even in the event of system failures, by providing full transaction support (Atomicity, Consistency, Isolation, Durability).
- Time Travel allows users to query the historical state of the data, making it easier to perform operations like auditing and recovering previous versions of data.

# Data Fabric, Data Mesh, Data Mart, Delta Lake

## ***Data Fabric***

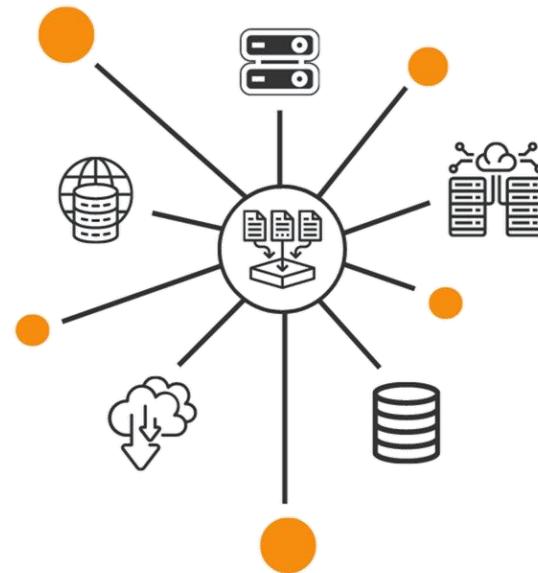
- ✓ Data Fabric is an architecture and set of data services that provide a unified, integrated, and intelligent data management layer across different environments. It enables seamless data integration, data governance, and data access across multiple platforms, whether on-premise, in the cloud, or in hybrid environments.



# Data Fabric, Data Mesh, Data Mart, Delta Lake

## *Data Mesh*

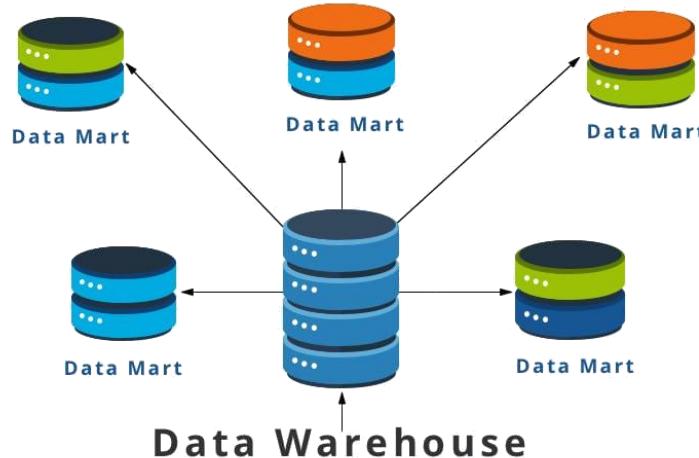
- ✓ Data Mesh is a decentralised approach to data architecture that focuses on distributed data ownership and responsibility. Rather than centralising data in a single repository like a data lake or data warehouse, Data Mesh decentralises data ownership to the teams responsible for the data domains. This approach encourages a domain-oriented design where individual teams manage, govern, and make decisions about their own data.



# Data Fabric, Data Mesh, Data Mart, Delta Lake

## ***Data Mart***

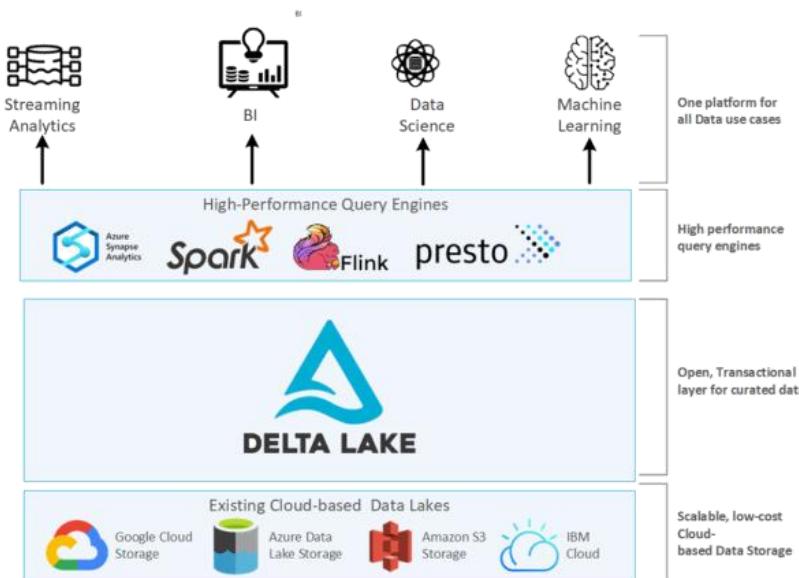
- ✓ A Data Mart is a subset of a Data Warehouse that is focused on a specific business area or department (e.g., sales, marketing, finance). It stores a specialised collection of data relevant to that department's needs and is typically smaller and more focused than a full Data Warehouse.



# Data Fabric, Data Mesh, Data Mart, Delta Lake

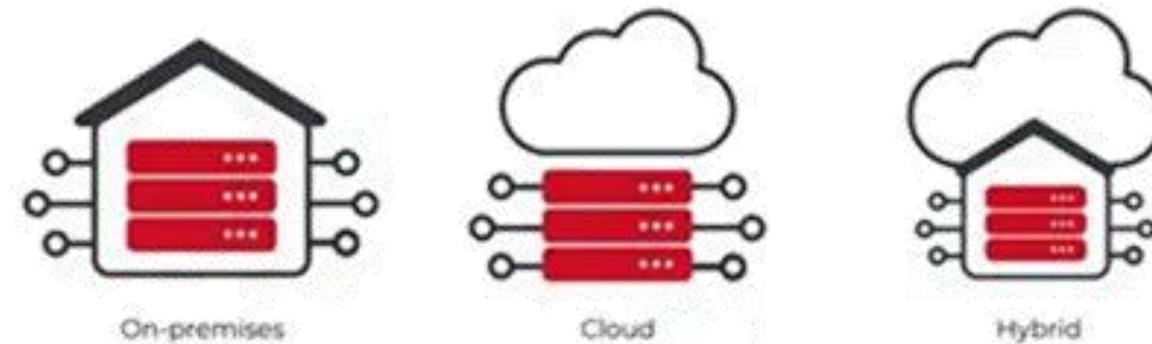
## *Delta Lake*

- ✓ Delta Lake is an open-source storage layer built on top of Apache Spark and Hadoop that provides ACID transactions, data versioning, and scalable metadata handling. It brings reliability and consistency to Data Lakes by adding the capabilities of Data Warehouses, such as schema enforcement, versioning, and the ability to handle batch and streaming data.



# Choosing the Right Storage Option

- ✓ Choosing the right storage solution is crucial for any data architecture. The storage option you select should align with your organisation's specific data needs, processing requirements, budget, and scalability goals.
- ✓ The data landscape today offers a variety of storage options such as Data Lakes, Data Warehouses, Databases, and Cloud Storage, each suited for different use cases.



# Choosing the Right Storage Option

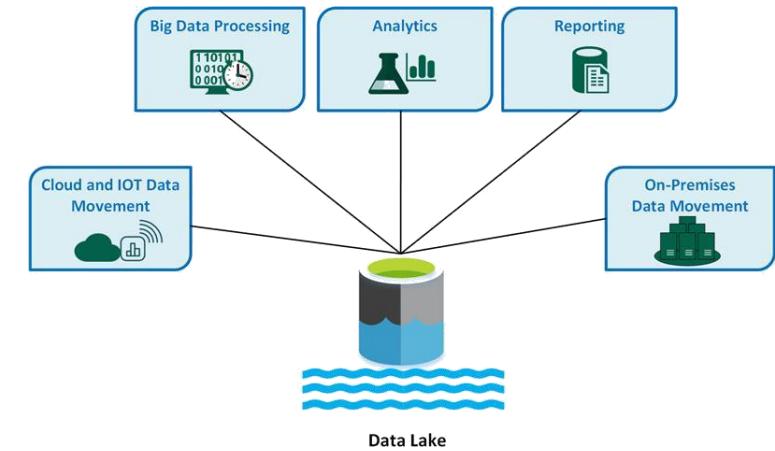
## *Types of Storage Options for Data:*

1. Relational Database Storage (RDBMS)
2. Data Warehouse Storage
3. Data Lake Storage
4. Cloud Storage
5. NoSQL Database Storage
6. Object Storage
7. Cold Storage
8. Hybrid Cloud Storage



# Data Lake Cloud Offerings

- ✓ A Data Lake is a centralised repository that allows you to store all your structured, semi-structured, and unstructured data at any scale.
- ✓ In recent years, cloud-based data lakes have become a popular choice for organisations, offering the flexibility, scalability, and cost-effectiveness needed to handle large volumes of data.
- ✓ Cloud Data Lake offerings allow businesses to leverage cloud infrastructure, without the need to build and maintain on-premise data lakes.
- ✓ Leading cloud providers offer specialised Data Lake solutions that integrate seamlessly with other cloud services, enabling businesses to store, manage, and analyse their data efficiently.



# Data Lake Cloud Offerings

(Continued)

- ✓ ***Below are some of the key cloud-based data lake solutions provided by major cloud platforms:***

## 1. Amazon S3 (AWS):

- Amazon S3 is one of the most popular cloud storage services that serves as the foundation for AWS's Data Lake offering. It provides object storage for large amounts of unstructured data and integrates with various AWS services for analytics, machine learning, and more.



# Data Lake Cloud Offerings

## 2. Google Cloud Storage (GCS):

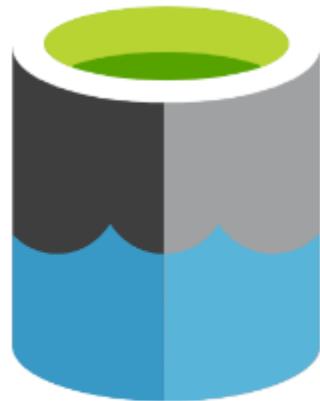
- Google Cloud Storage provides scalable, durable, and highly available object storage for data lakes. It offers seamless integration with other Google Cloud services, such as BigQuery for data analysis and Google Dataproc for distributed data processing.



# Data Lake Cloud Offerings

## 3. Azure Data Lake Storage:

- Azure Data Lake Storage is a scalable and secure data lake service for big data analytics on Azure. It's built on Azure Blob Storage and offers a comprehensive data lake solution with high integration with Azure Analytics Services.



Azure Data Lake Storage

# Data Lake Cloud Offerings

## 4. IBM Cloud Object Storage:

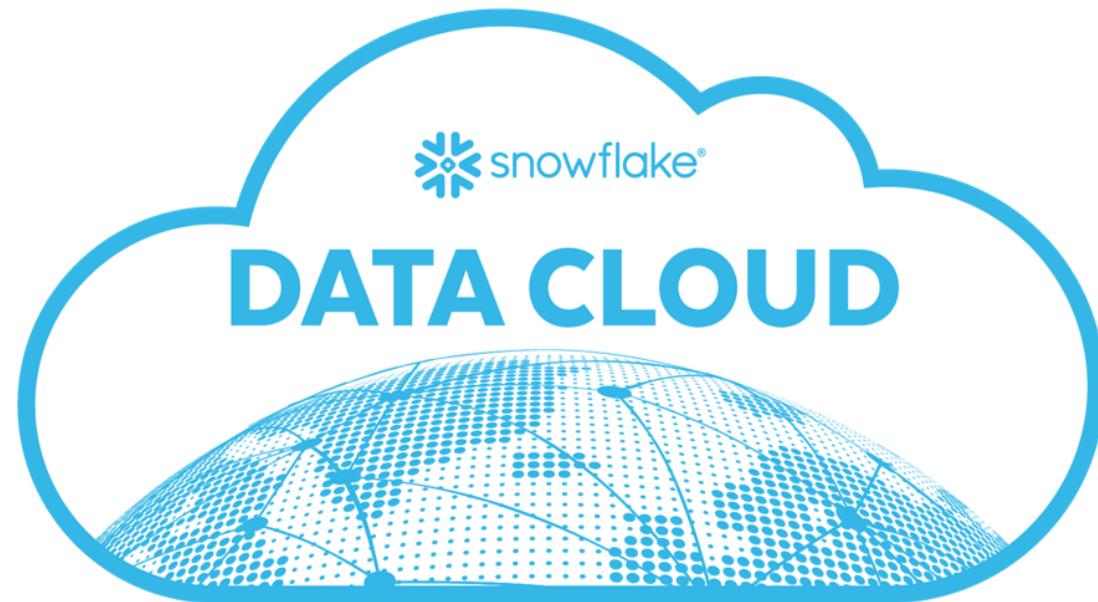
- IBM Cloud Object Storage provides scalable and secure cloud storage for data lakes. It is designed to handle massive datasets and integrates with IBM Watson for advanced data analytics and machine learning.



# Data Lake Cloud Offerings

## 5. Snowflake Data Cloud:

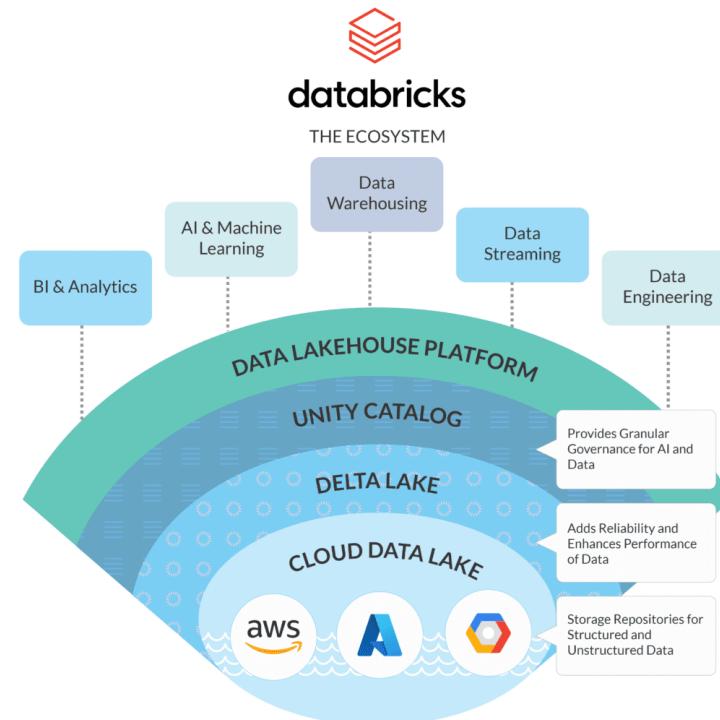
- Snowflake is a cloud-native data platform that supports data lakes and data warehouses in a single solution. It separates compute from storage, allowing for efficient scaling of both independently.



# Data Lake Cloud Offerings

## 6. Databricks Lakehouse:

- Databricks Lakehouse combines the features of data lakes and data warehouses into a unified platform, enabling scalable real-time analytics and AI workloads.



# Cloud Data Warehouse Services

- ✓ A Cloud Data Warehouse is a specialised system that allows businesses to store, manage, and analyse large volumes of structured data in the cloud.
- ✓ Unlike traditional on-premise data warehouses, cloud data warehouses are fully managed, scalable, and accessible over the internet. They are optimised for high-performance querying, complex analytics, and business intelligence (BI) tasks.
- ✓ Cloud data warehouse services are designed to support real-time and batch processing and allow organisations to efficiently perform analytical queries on large datasets.
- ✓ They offer numerous benefits, including scalability, elasticity, performance optimisation, and integration with other cloud-based services such as machine learning, business intelligence tools, and data lakes.

# Cloud Data Warehouse Services

*Here's an overview of the major cloud data warehouse services provided by leading cloud providers:*

## 1. Amazon Redshift (AWS):

- Amazon Redshift is one of the most widely used cloud data warehouse services, known for its performance, scalability, and integration with other AWS services.

### **Key Features**

- **Columnar Storage:** Uses a columnar storage model, which optimises read-heavy workloads and improves query performance.
- **Parallel Query Processing:** Amazon Redshift supports parallel query execution, enabling faster results for complex analytical queries.

# Cloud Data Warehouse Services

## 2. Google BigQuery (Google Cloud):

- BigQuery is Google Cloud's serverless, fully managed data warehouse, optimised for fast SQL queries and large-scale data analytics.

### ***Key Features:***

- **Serverless:** No need to provision infrastructure, as BigQuery automatically scales to accommodate workloads.
- **Real-Time Analytics:** Supports real-time data streaming, allowing you to ingest and analyse data as it arrives.
- **High Performance:** Offers high-speed querying with columnar storage and distributed execution of queries across Google's infrastructure.

# Cloud Data Warehouse Services

### 3. Snowflake:

- Snowflake is a cloud-native data warehouse that provides flexible, scalable data storage and computing capabilities.

#### ***Key Features***

- **Multi-Cloud:** Snowflake runs on AWS, Azure, and Google Cloud, making it suitable for multi-cloud environments.
- **Separation of Compute and Storage:** Unlike traditional data warehouses, Snowflake separates compute and storage. This allows you to scale storage and compute independently.
- **Automatic Scaling:** Snowflake automatically scales to handle varying workloads and allows users to pay only for the resources used.
- **Zero-Copy Cloning:** Enables the creation of clones of data without duplicating storage, allowing for quick testing and experimentation.

# Cloud Data Warehouse Services

## 4. Microsoft Azure Synapse Analytics:

- Azure Synapse Analytics (formerly SQL Data Warehouse) is a cloud-based analytics service that brings together big data and data warehousing in a unified platform.

### ***Key Features***

- **Unified Analytics Platform:** Integrates both data warehousing (SQL-based) and big data processing (Spark-based), enabling unified analytics.
- **On-demand Querying:** Supports serverless querying of data stored in Azure Data Lake.
- **Deep Integration with Azure Services:** Seamlessly integrates with Power BI, Azure Machine Learning, and Azure Data Factory for end-to-end analytics.
- **Security and Governance:** Offers advanced security features such as encryption, managed identity, and role-based access control.

# Cloud Data Warehouse Services

## 5. IBM Db2 Warehouse on Cloud:

- IBM Db2 Warehouse on Cloud is an enterprise-class cloud data warehouse designed for structured and unstructured data analytics.

### ***Key Features***

- **Flexible Storage Options:** Offers both cloud storage and on-premise deployment.
- **AI-Driven Analytics:** Integrates with IBM Watson for machine learning and AI-driven insights.
- **High Availability:** Provides built-in high availability and data redundancy features.
- **Optimised for Analytics:** Offers optimised storage and high-performance querying for real-time analytics.

# Module 20: Architecting Data Pipelines in GCP

- Designing Data Pipelines for Scalability and Efficiency
- Utilising GCP for Real-time Data Processing
- Integration of GCP with External Data Sources
- Security and Compliance in GCP Data Pipelines



# Designing Data Pipelines for Scalability and Efficiency

✓ *The following are the methods for designing data pipelines:*

## 1. Understand the Components of a Data Pipeline in GCP:

- Data Ingestion: Collecting data from external sources (e.g., APIs, databases, IoT devices).
- Data Processing: Transforming or cleaning the data using distributed processing frameworks.
- Data Storage: Storing the processed data in a data warehouse or data lake.
- Data Analysis: Querying and analysing data in real-time or batch mode.

## 2. Define Your Data Pipeline Requirements:

- Data Sources: Where is the data coming from (e.g., on-premise, other cloud providers, real-time streams)?
- Data Size: Will the data scale up significantly over time?

# Designing Data Pipelines for Scalability and Efficiency

(Continued)

- **Latency:** Do you need real-time or batch processing?
- **Performance:** What processing time is acceptable?
- **Cost:** What is the budget for processing and storing data?

### 3. Choose the Right GCP Services for Scalability and Efficiency:

- **Cloud Pub/Sub:** A messaging service for real-time event-driven data ingestion.
- **Cloud Dataflow:** A fully managed service for stream and batch data processing using Apache Beam.
- **BigQuery:** A fully-managed, scalable data warehouse for analysing large datasets.
- **Cloud Storage:** Object storage for storing raw or processed data.

# Designing Data Pipelines for Scalability and Efficiency

## 4. Design the Pipeline Architecture:

- **Data Ingestion:** Use Cloud Pub/Sub for real-time data ingestion (e.g., from IoT devices, logs, external systems). Use Cloud Storage for batch data ingestion (e.g., files uploaded periodically).
- **Data Processing:** Use Cloud Dataflow for real-time or batch processing of data. Dataflow allows you to create data pipelines using Apache Beam to handle transformations (ETL). Data can be read from Cloud Pub/Sub, Cloud Storage, or BigQuery and then processed.
- **Data Storage:** Store the processed data in BigQuery for structured, analytical queries, or in Cloud Storage for raw and semi-structured data.
- **Data Monitoring:** Use Cloud Monitoring and Cloud Logging to monitor and troubleshoot the pipeline. Set up alerts and dashboards for the health of your pipeline (e.g., if a task fails or data ingestion delays occur).

# Designing Data Pipelines for Scalability and Efficiency

## 5. Implement Scalability Features:

- **Scalable Data Ingestion:** Pub/Sub automatically scales to handle millions of messages per second, making it ideal for high-volume data ingestion. If using Cloud Storage, organise data into multiple buckets and use object lifecycle management to handle data archiving and deletion.
- **Scalable Data Processing:** Dataflow allows you to scale horizontally by adding more workers as data volumes grow. It also optimises resource usage based on the size and complexity of the data. Use autoscaling with Dataflow to adjust resources dynamically based on data processing demand.
- **Cost Management:** Use BigQuery's serverless architecture for querying large datasets without worrying about the infrastructure. Apply partitioning and clustering in BigQuery for better performance and cost management.

# Designing Data Pipelines for Scalability and Efficiency

## 6. Test, Deploy, and Monitor the Pipeline

- **Test the Pipeline:** Run the pipeline in a staging environment with test data to ensure everything works smoothly. Verify that Cloud Pub/Sub, Dataflow, and BigQuery are integrated correctly.
- **Deploy to Production:** Once the pipeline works correctly in the test environment, deploy it to production. Use Cloud Composer (Apache Airflow) to schedule and orchestrate the pipeline in production.
- **Monitor and Troubleshoot:** Use Cloud Monitoring and Cloud Logging to track the health of the pipeline and check logs for any errors. Set up alerts in Cloud Monitoring to notify you if the pipeline fails or if there are delays in data processing.

# Designing Data Pipelines for Scalability and Efficiency

## 7. Automate the Pipeline:

- **Schedule the Data Pipeline:** Use Cloud Composer to orchestrate and schedule your data pipeline. Airflow can automate the execution of the pipeline at regular intervals (e.g., hourly, daily).
- **Set up Retry Mechanisms:** Ensure that your pipeline has automatic retry mechanisms in place, especially for critical tasks (e.g., data ingestion or data processing).



# Utilising GCP for Real-time Data Processing

- ✓ In Google Cloud Platform (GCP), real-time data processing allows you to analyse and act on data as it arrives, making it suitable for applications that require immediate insights. This can include use cases like real-time analytics, monitoring, fraud detection, and event-driven applications.

## ***Key Components for Real-Time Data Processing***

1. **Cloud Pub/Sub:** Real-time messaging service that decouples data producers from data consumers, enabling the ingestion of real-time data streams.
2. **Cloud Dataflow:** Fully managed service for stream and batch data processing that supports Apache Beam.
3. **BigQuery:** A fully managed, serverless data warehouse that allows you to perform real-time analytics on large datasets.
4. **Cloud Functions:** Event-driven, serverless compute service for triggering actions in response to events.

# Utilising GCP for Real-time Data Processing

5. **Cloud Storage:** Object storage for handling raw data or intermediate processing stages.
6. **Cloud Monitoring and Logging:** For tracking and alerting on system health, performance, and errors

✓ *The following are the utilising GCP for real-time data processing:*

- Set Up Your GCP Project
- Set Up Cloud Pub/Sub for Data Ingestion
- Set Up Cloud Dataflow for Real-Time Processing
- Query Data in BigQuery
- Set Up Real-Time Monitoring and Alerts
- Scalability Considerations



# Integration of GCP with External Data Sources

- ✓ Integrating Google Cloud Platform (GCP) with external data sources allows organisations to enrich their data pipelines, facilitate data exchange, and perform analytics on diverse datasets.
- ✓ Whether you need to integrate data from external databases, APIs, on-premise systems, or other cloud services, GCP provides several tools and services to seamlessly ingest, process, and analyse data from a variety of sources.

## ***Key Tools and Services for Data Integration***

1. **Cloud Storage:** For storing and ingesting raw data files (e.g., CSV, JSON) from external sources.
2. **Cloud Pub/Sub:** For real-time messaging and streaming data integration.
3. **Cloud Dataflow:** For ETL (Extract, Transform, Load) processing from external systems.
4. **BigQuery:** For loading and analysing external structured data.

# Integration of GCP with External Data Sources

5. **Cloud Functions:** For triggering actions based on events, such as new data from an external API.
6. **Data Transfer Service:** For automating data transfers from external services like Amazon S3, Google Analytics, YouTube, and more.
7. **Cloud SQL / Cloud Spanner:** For integrating with external relational databases (e.g., MySQL, PostgreSQL).

✓ ***The following are the integration of GCP with External Data sources:***

1. Set Up Your GCP Project
2. Data Ingestion from External Sources
  - Ingesting Files from External Systems to Cloud Storage
  - Integrating Real-Time Data via Cloud Pub/Sub
  - Ingesting Data from External Databases (Cloud SQL / BigQuery)

# Integration of GCP with External Data Sources

3. Data Processing and Transformation
4. Store Data in BigQuery
5. Monitor and Automate Data Pipelines
6. Automate and Scale the Pipeline
  - Scaling Cloud Dataflow
  - Schedule Pipelines:



# Security and Compliance in GCP Data Pipelines

- ✓ When designing and deploying data pipelines in Google Cloud Platform (GCP), ensuring security and compliance is critical, especially when handling sensitive data. GCP provides various tools, services, and best practices to help safeguard your data and meet regulatory requirements.

## ***Key Principles of Data Pipeline Security and Compliance in GCP***

- 1. Data Encryption:** Ensure that data is encrypted both in-transit and at-rest to protect data from unauthorised access.
- 2. Identity and Access Management (IAM):** Control access to resources with precise permissions and roles.
- 3. Audit Logging:** Keep track of who accessed your data and what actions they performed.
- 4. Compliance:** Ensure your data pipelines meet relevant regulatory standards like GDPR, HIPAA, PCI DSS, and others.

# Security and Compliance in GCP Data Pipelines

(Continued)

- ✓ When building data pipelines in Google Cloud Platform (GCP), ensuring the security and compliance of data is crucial. GCP provides a wide range of tools and services to ensure that your data pipeline meets industry standards for privacy, security, and regulatory requirements.
- ✓ This topic focuses on securing your data pipelines and ensuring they comply with regulations such as GDPR, HIPAA, and ISO/IEC 27001

## ***Key Considerations for Security and Compliance in Data Pipelines***

1. **Data Privacy and Confidentiality:** Protecting the privacy of data, especially personally identifiable information (PII), and ensuring it is only accessible to authorised users.
2. **Data Integrity:** Ensuring that data remains accurate, unaltered, and secure during its journey through the pipeline.
3. **Regulatory Compliance:** Adhering to various regulatory requirements such as GDPR, HIPAA, SOC 2, and ISO 27001, depending on the region and the nature of the data.

# Module 21: Advanced Machine Learning with GCP

- Machine Learning Model Deployment in GCP
- AutoML: Leveraging Automated Machine Learning in GCP
- Building and Scaling ML Models with TensorFlow and GCP
- Advanced Analytics and ML Feature Engineering



# Machine Learning Model Deployment in GCP

- ✓ Deploying machine learning models in Google Cloud Platform (GCP) enables organisations to take their trained models from development to production, where they can serve predictions at scale.
- ✓ GCP offers several powerful tools and services for deploying, managing, and scaling machine learning models, allowing you to focus on model optimisation rather than infrastructure management.
- ✓ With the use of services like AI Platform and Vertex AI, you can easily integrate your models into production environments and deliver real-time or batch predictions efficiently.
- ✓ ***The following are the deployment machine learning models on GCP:***
  1. Before deployment, ensure you have a trained machine learning model. This model can be created using tools like: **TensorFlow, Scikit-learn, XGBoost, Keras, PyTorch**

# Machine Learning Model Deployment in GCP

2. GCP offers several ways to deploy machine learning models. Depending on your use case, you can choose from:
  - Vertex AI (Recommended for Full ML Lifecycle): Vertex AI is a managed service that provides end-to-end capabilities for building, training, deploying, and monitoring machine learning models. It supports both online predictions (real-time) and batch predictions (large-scale data processing).
  - AI Platform Prediction: Similar to Vertex AI but a bit more streamlined, AI Platform Prediction helps you serve your model on a scalable and secure platform.
  - Cloud Functions (for Lightweight Deployments): If you need to deploy a simple model for real-time predictions via API calls, you can use Cloud Functions. This is a serverless solution for lightweight models.
  - App Engine or Cloud Run: If you need to deploy a custom application that serves your model, use App Engine or Cloud Run to deploy the application as a containerised service.

# Machine Learning Model Deployment in GCP

3. Let's walk through the process of deploying a model using Vertex AI, which is recommended for most machine learning use cases on GCP:
  - Configure the Prediction Endpoint: Once deployed, Vertex AI provides an HTTP endpoint where your model can be accessed for online predictions. You will also get a REST API to integrate the deployed model into your applications.
4. Deploying Using AI Platform Prediction:
  - Deploy the Model on AI Platform: After creating the model, deploy it on AI Platform by creating a Version of the model. Choose Machine type (e.g., n1-standard-4) and specify other deployment configurations.
  - Access the Model Endpoint for Predictions: AI Platform will provide a URL for making REST API calls to the model for predictions.

# Machine Learning Model Deployment in GCP

5. Once the model is deployed, it's crucial to monitor and manage its performance in production:

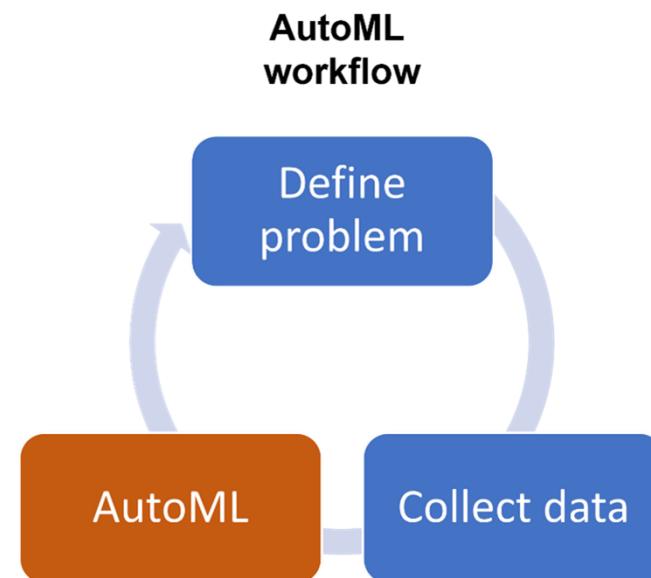
- Monitor with Vertex AI: Use the Vertex AI console to view detailed information about the model performance, including latency, accuracy, and request/response metrics.
- Logging and Monitoring: Set up Cloud Logging and Cloud Monitoring to track errors, latency, and utilisation. Set up alerts for anomalies in model predictions or performance issues.

6. Update or Retrain the Model:

- Model Retraining: As new data becomes available, you may need to retrain your model to improve its performance. You can either automate the retraining process using Cloud AI or Dataflow or manually upload the updated model.
- Update Deployed Model: You can create a new version of your model in Vertex AI or AI Platform to replace the old model.

# AutoML: Leveraging Automated Machine Learning in GCP

- ✓ AutoML (Automated Machine Learning) refers to the process of automating the end-to-end process of building, training, and deploying machine learning models, making it easier for non-experts to build high-quality models.
- ✓ Google Cloud Platform (GCP) provides a powerful suite of tools under AutoML to automate various aspects of machine learning, allowing developers to build and deploy models without requiring deep expertise in data science or machine learning.



# AutoML: Leveraging Automated Machine Learning in GCP

(Continued)

- ✓ ***The workflow depicted in the image for AutoML involves the following steps:***
- 1. **Define the Problem:** This is the initial stage where you determine the problem you are trying to solve with machine learning. It's crucial to clearly define what you want to predict or classify.
- 2. **Collect Data:** In this step, you gather the necessary data to train your machine learning model. The quality and quantity of data collected are vital as they directly influence the model's performance. This data could be images, text, numbers, or structured data.
- 3. **AutoML:** Once the data is collected, you use AutoML to build a model. AutoML automates the process of model selection, training, and hyperparameter tuning, eliminating the need for expert-level knowledge of machine learning.
- 4. **Loop Back:** After using AutoML to train the model, you might need to iterate over the process. If the model doesn't meet expectations, you can revisit the problem definition or the data collection phase, refine your approach, and retrain the model to improve results.

# AutoML: Leveraging Automated Machine Learning in GCP

(Continued)

✓ *The following are the key benefits of using automl in GCP:*

1. **Ease of Use:** GCP AutoML allows users with minimal machine learning expertise to build custom models by simply providing labeled data and specifying model parameters.
2. **Speed and Efficiency:** AutoML tools can significantly reduce the time it takes to train and deploy models, as they automate many of the time-consuming tasks typically handled by data scientists.
3. **Scalability:** AutoML solutions are highly scalable and can handle large datasets without compromising performance.
4. **Customisable Models:** Unlike pre-trained models, AutoML allows you to build customized models tailored specifically to your use case and data.
5. **State-of-the-Art Models:** AutoML leverages Google's best-in-class algorithms and infrastructure, including neural networks and deep learning, ensuring high model accuracy.

# Building and Scaling ML Models with TensorFlow and GCP

- ✓ TensorFlow provides a rich ecosystem for developing machine learning models, including tools for data manipulation, model building, and evaluation. When paired with GCP, TensorFlow can take advantage of cloud-based resources like Compute Engine, Cloud Storage, AI Platform, and TPUs (Tensor Processing Units) for accelerated training and deployment of models.
- ✓ ***The following are the key advantages of using tensorflow with GCP:***
  1. **Scalability:** GCP offers high-performance virtual machines and distributed computing, which allow you to scale your TensorFlow models efficiently.
  2. **Managed Services:** With AI Platform and Kubeflow, you can manage your ML workflows, ensuring that your TensorFlow models are continuously trained and deployed at scale.
  3. **High-speed Compute:** By using TPUs (specialised hardware for machine learning), you can significantly speed up the training of large models.
  4. **Integrated Data Storage:** GCP provides Cloud Storage for scalable data storage, making it easy to store large datasets and serve them to your TensorFlow models.

# Building and Scaling ML Models with TensorFlow and GCP

(Continued)

- ✓ ***The following are the methods for building and scaling ML models with TensorFlow and GCP:***

1. Set Up Your GCP Project
2. Set Up TensorFlow in a Virtual Machine (VM)
3. Train a TensorFlow Model on GCP
4. Scaling the Model Using GCP
5. Deploy Your TensorFlow Model on GCP
6. Monitor the Model and Improve



# Advanced Analytics and ML Feature Engineering

- ✓ Advanced Analytics involves using sophisticated tools and techniques to analyse large datasets and extract actionable insights. This can involve various statistical and machine learning methods, including:
  1. **Predictive Analytics:** Making predictions based on historical data (e.g., forecasting demand, predicting customer churn).
  2. **Prescriptive Analytics:** Providing recommendations for optimal decision-making.
  3. **Descriptive Analytics:** Understanding what has happened in the past.
  4. **Diagnostic Analytics:** Investigating the cause of an event.
  5. **Feature Engineering** is the process of selecting and transforming variables (features) from raw data into formats suitable for building machine learning models. It can dramatically improve model accuracy and robustness by providing the model with more meaningful and relevant inputs.

# Advanced Analytics and ML Feature Engineering

(Continued)

- ✓ ***The following are the key aspects of feature engineering in ML:***

## 1. Data Transformation:

- **Scaling:** Scaling numerical features to a standard range (e.g., using MinMaxScaler or StandardScaler) can help models converge faster and perform better.
- **Normalisation:** Rescaling values of features to a consistent scale (like z-score normalization) for models sensitive to varying scales.
- **Log Transformations:** Using log transformations to handle features with skewed distributions.

## 2. Categorical Feature Encoding:

- **One-Hot Encoding:** This converts categorical variables into binary vectors, where each category gets a unique vector.

# Advanced Analytics and ML Feature Engineering

(Continued)

- **Label Encoding:** Encoding categories with unique integers. It's important to apply it only when the categories have an ordinal relationship.
- **Target Encoding:** Calculating the mean target value for each category and encoding the categorical feature with these means.

### 3. Feature Creation:

- **Polynomial Features:** Creating higher-order features by combining original features (e.g.,  $x^2$ ,  $x^3$ ) to capture nonlinear relationships.
- **Interaction Features:** Combining two or more features to capture interactions between them (e.g., multiplying two features together).
- **Time-Based Features:** For time series data, creating features such as day of the week, month, or lagged variables.

# Advanced Analytics and ML Feature Engineering

## 4. Missing Data Imputation:

- **Mean/Median/Mode Imputation:** Filling missing values with statistical measures.
- **KNN Imputation:** Using the nearest neighbors to fill in missing values.
- **Advanced Imputation:** Using algorithms such as Random Forests or even deep learning models to predict missing values based on other features.

## 5. Feature Selection:

- **Filter Methods:** Using statistical tests (like chi-square, correlation coefficients) to select features that have the most impact on the target variable.
- **Wrapper Methods:** Using a machine learning algorithm (like Recursive Feature Elimination) to evaluate feature subsets.
- **Embedded Methods:** Feature selection embedded into the learning algorithm (e.g., Lasso regression, decision trees).



## The World's Largest Global Training Provider

✉ theknowledgeacademy.com

🌐 info@theknowledgeacademy.com

 /The.Knowledge.Academy.Ltd

 /TKA\_Training

 /the-knowledge-academy

 /TheKnowledgeAcademy

