



Analytics Lakehouse

Release Q3 2025

EnterpriseDB Corporation

Sep 25, 2025

1	Introduction	3
1.1	Navigating the analytics accelerator	3
1.2	Conceptual foundations	3
1.3	EDB core analytics technologies	4
1.4	Practical guidance and solutions	4
1.5	Product-specific implementations	4
1.6	Where to start	4
1.6.1	Level 100	5
1.6.2	Level 200	5
1.6.3	Level 300	6
2	Quick start	7
2.1	Introduction	7
2.1.1	What to know	7
2.1.2	Basic architecture	7
2.2	Getting started	8
2.2.1	Create a Lakehouse node	8
2.3	Operating a Lakehouse node	10
2.3.1	Connect to the node	10
2.3.2	Understand the constraints	10
2.4	Inspect the benchmark datasets	10
2.5	Query the benchmark datasets	11
3	Tiered Tables	13
3.1	What are Tiered Tables	13
3.2	Why Tiered Tables matter for EDB analytics	13
3.3	How EDB implements Tiered Tables	14
3.4	Common use cases	14
3.5	Role-based guidance	14
3.6	Learning paths	15
3.7	Related concepts	15
3.8	Next steps	15
4	Reference - EDB Postgres® AI Lakehouse	17
4.1	Benchmarking datasets	17
4.2	Queries	18
4.3	Delta Lake Table tools	18

4.3.1	Creating Delta Lake Tables	18
4.4	DirectScan and fallback modes	19
4.4.1	DirectScan	19
4.4.2	Using EXPLAIN to check the query plan	19
4.5	User management	20
4.6	Loading data	20
4.6.1	Bringing your own data	20
4.7	PGAA functions reference	21
4.7.1	Synopsis	21
4.7.2	Parameters	21
4.7.3	Returns	21
4.7.4	Synopsis	21
4.7.5	Parameters	21
4.7.6	Returns	21
4.8	PGAA functions reference	21
4.8.1	pgaa.create_storage_location	21
4.9	Supported AWS instances	22
5	Delta Lake	23
5.1	What is Delta Lake	23
5.2	Why Delta Lake matters for EDB analytics	23
5.3	How EDB leverages Delta Lake	24
5.4	Common use cases	24
5.5	Role-based guidance	24
5.6	Learning paths	24
5.7	Related concepts	24
5.8	Next steps	25
6	Apache Iceberg	27
6.1	What is Apache Iceberg	27
6.1.1	Key goals	27
6.2	Key features and benefits	27
6.3	Why Apache Iceberg matters for EDB analytics	28
6.4	How EDB solutions implement Iceberg support	28
6.4.1	PGAA extensions	28
6.4.2	Catalog connectivity	28
6.4.3	Writing to Iceberg tables	28
6.5	Related topics	29
7	EDB Postgres Lakehouse	31
7.1	What is the EDB Postgres Lakehouse	31
7.2	Why Lakehouse matters for EDB analytics	31
7.3	How EDB implements Lakehouse architecture	32
7.4	Common use cases	32
7.5	Role-based guidance	32
7.6	Learning paths	33
7.7	Related concepts	33
7.8	Next steps	33
8	Analytics Accelerator learning resources	35
8.1	Documentation categories	35
8.1.1	Structured learning paths	35
8.1.2	Conceptual explanations	35
8.1.3	Tutorials	36
8.1.4	How-to guides	36

8.1.5	Use cases and solutions	36
8.1.6	Persona-based guidance	36
8.1.7	Analytics Accelerator in Hybrid Manager (HM)	36
9	Analytics Accelerator learning resources	37
9.1	Analytics Accelerator concepts and technologies	37
9.2	Analytics Accelerator concepts and technologies	37
9.2.1	Foundational analytics and modern data architectures	37
9.3	Analytics Accelerator concepts and technologies	38
9.3.1	Deep dive into core Analytics Accelerator components	38
9.4	Analytics Accelerator concepts and technologies	38
9.4.1	Understanding analytics within EDB Hybrid Manager (HM)	38
9.4.2	AI/ML workloads and interoperability	38
9.5	Analytics Accelerator learning paths	49
9.5.1	How to use these learning paths	49
9.6	Analytics Accelerator learning paths	49
9.7	Analytics Accelerator learning paths	50
9.7.1	EDB training and certification	50
9.8	Analytics Accelerator learning paths	50
9.8.1	Next steps	50
9.9	a persona-based guide	52
9.9.1	Quickly find your role	52
9.9.2	Database administrator (DBA)	53
9.9.3	DevOps engineer / site reliability engineer (SRE)	53
9.9.4	Data scientist / data analyst	54
9.9.5	Application developer	55
9.10	Analytics Accelerator tutorials	55
9.10.1	Tutorial categories	56

Use the **Analytics Accelerator** to explore the analytical capabilities built on EDB Postgres®. This accelerator helps you understand core concepts, explore key technologies such as EDB Postgres® Lakehouse, and learn how to implement analytics with EDB Hybrid Manager (HM).

We integrate modern data architectures and open standards with the reliability and flexibility of Postgres to help you unlock valuable insights.

1.1 Navigating the analytics accelerator

The accelerator organizes content into four areas:

- **Conceptual foundations** Build your understanding of analytics principles and EDB’s approach.
- **EDB core analytics technologies** Learn about EDB solutions and technologies that power our analytics offerings.
- **Practical guidance and solutions** Find use cases, persona-based guides, how-to articles, and tutorials.
- **Product-specific implementations** Access documentation for how these analytics capabilities surface and are managed in EDB products, such as EDB Hybrid Manager.

1.2 Conceptual foundations

Understand the principles and strategies behind modern data analytics and EDB’s approach.

- *Generic analytics concepts*

Learn about data architectures (Data Warehouse, Data Lake, Lakehouse) and foundational technologies (columnar storage, vectorized engines, and others).

- *EDB analytics concepts*

Explore EDB’s vision for Postgres® analytics and how EDB leverages core technologies.

- Explained: Analytics

Review in-depth explanations of EDB analytical features, design choices, and advanced topics. (*Coming soon*)

1.3 EDB core analytics technologies

Learn about EDB's analytics technologies and how they extend Postgres®.

- *EDB Postgres Lakehouse*

Review the EDB Postgres® Lakehouse solution and its components for enabling analytics on object storage.

- *Apache Iceberg*

Understand how EDB solutions use Apache Iceberg to manage large analytical datasets.

- *Delta Lake*

Learn how EDB Postgres® interacts with Delta tables to enable reliable data lakes.

- *EDB Postgres Distributed (PGD) and Tiered Tables*

Manage data across storage tiers using EDB Postgres Distributed (PGD) and Lakehouse capabilities to optimize cost and performance.

1.4 Practical guidance and solutions

Apply EDB's analytics capabilities to meet your needs.

- *Analytics Accelerator for your role: a persona-based guide*

Follow learning paths for DBAs, DevOps engineers, data scientists, and application developers.

1.5 Product-specific implementations

Review how EDB analytics concepts and technologies are implemented in EDB products.

- *Introduction*

Access documentation for analytics features in EDB Hybrid Manager. This includes HM Lakehouse clusters, using Iceberg, Delta, and tiered tables in HM, and HM-specific tutorials.

1.6 Where to start

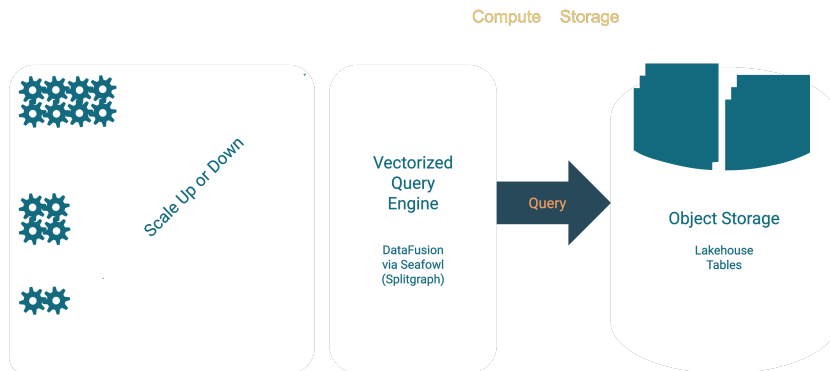
- Start with *Generic analytics concepts* and *EDB Postgres Lakehouse* to understand core ideas.
- Explore practical guidance when available.
- Use product-specific documentation when working with EDB Hybrid Manager.

Postgres Lakehouse is built using a number of technologies:

- PostgreSQL
- *Seafowl* , an analytical database
- *Apache DataFusion* , the query engine used by Seafowl
- *Delta Lake* (and specifically *delta-rs*), for implementing the storage and retrieval layer of Delta Tables

1.6.1 Level 100

The most important thing to understand about Postgres Lakehouse is that it separates storage from compute. This design allows you to scale them independently, which is ideal for analytical workloads where queries can be unpredictable and spiky. You wouldn't want to keep a machine mostly idle just to hold data on its attached hard drives. Instead, you can keep data in object storage (and also in highly compressible formats), and only provision the compute needed to query it when necessary.

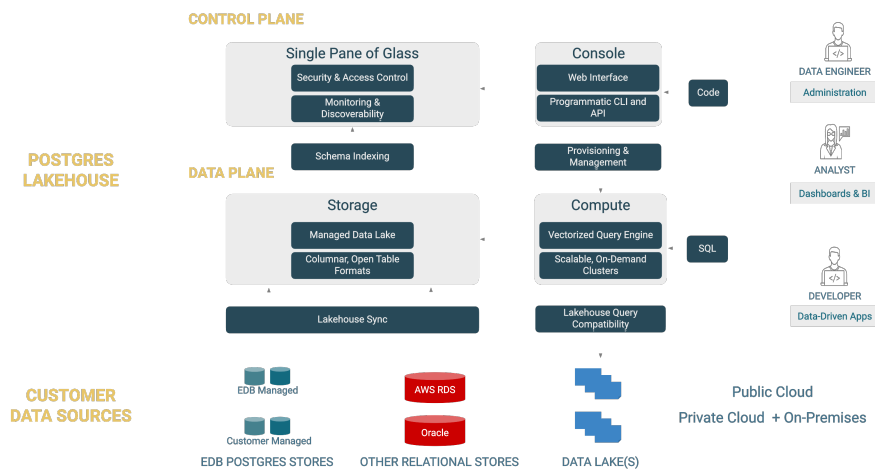


On the compute side, a vectorized query engine is optimized to query Lakehouse tables but still fall back to Postgres for full compatibility.

On the storage side, Lakehouse tables are stored using highly compressible columnar storage formats optimized for analytics.

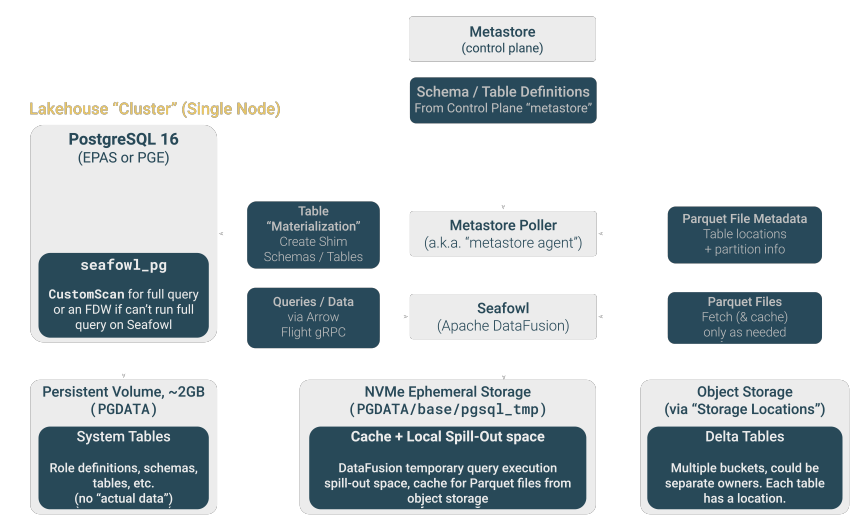
1.6.2 Level 200

Here's a slightly more comprehensive diagram of how these services fit together:



1.6.3 Level 300

Here’s the more detailed, zoomed-in view of “what’s in the box”:



In this quick start, you will:

1. Create a Lakehouse node.
2. Connect to the node with your preferred Postgres client.
3. Query sample data (TPC-H, TPC-DS, Clickbench, or 1BRC) in object storage.

For more details and advanced use cases, see [PGAA functions reference](#) .

2.1 Introduction

Postgres Lakehouse is a new type of Postgres cluster (it's really just one node) that you can provision with on the EDB Postgres® AI Platform. It includes a vectorized query engine based on Apache [DataFusion](#) for fast queries over columnar Lakehouse tables in object storage (using the [Delta Lake protocol](#)).

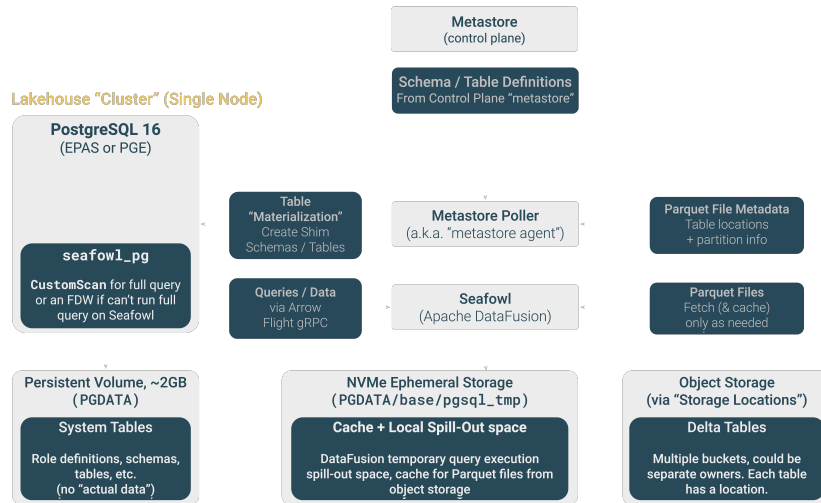
2.1.1 What to know

For more details, see [Related concepts](#) and [PGAA functions reference](#) , but these are the main points to know:

- Postgres Lakehouse nodes are ephemeral.
- Postgres Lakehouse nodes are preconfigured to query benchmark datasets in object storage.
- Postgres Lakehouse is generally read-only but otherwise supports all valid Postgres queries.
- System tables are writable, so you can create users and roles. These tables are stored on the attached block storage, so they persist between instance restarts and are saved as part of backup/restore operations. Otherwise, Lakehouse tables aren't part of backups, since they're ultimately stored in object storage.

2.1.2 Basic architecture

The figure shows what's in the box of a Lakehouse node:



2.2 Getting started

Once you've logged in to your HCP portal and created a project, you can create a cluster.

2.2.1 Create a Lakehouse node

On your project page, select **Create New > Lakehouse Analytics**:

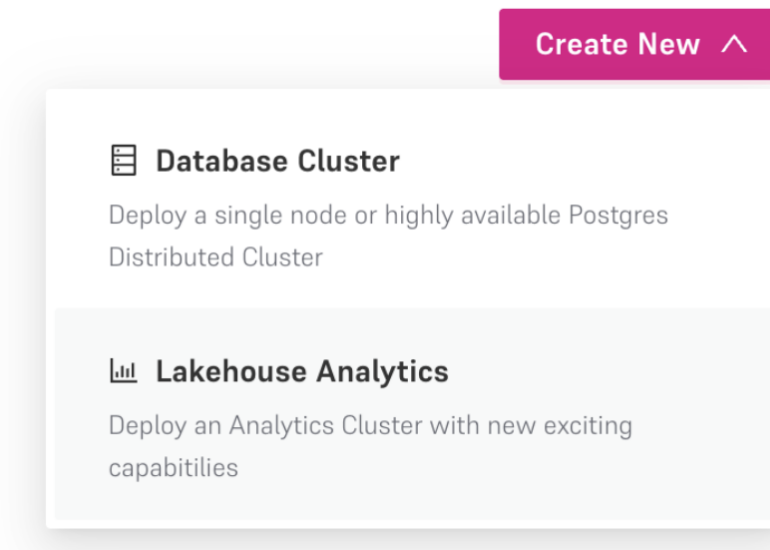


Fig. 1: Create Lakehouse Node Dropdown

The **Create Analytical Cluster** page opens, where you can choose between a templated build or a custom build:

Select **Custom Build**, and then select **Start from Scratch**. The **Create Analytical Cluster** page opens, where you can choose the cluster size and configuration:

Back

Create Analytical Cluster

Support
EDB Postgres® AI Docs

Choose Path

Choose Your Path

Use a template or build from scratch.

No Templates Available, Yet...

No templates are available, but you can create your own! Start building a new template below.

Create New Template

Custom Build

Customize every aspect of your cluster, from primary settings to backups. Best for advanced users needing flexibility.

Start from Scratch

Fig. 2: Create Lakehouse Node Wizard Step 1

Back

Create Analytical Cluster

Support
EDB Postgres® AI Docs

Cluster Settings
Additional Settings

Cluster Name

Enter a name for your cluster

Password

Passwords must be at least 12 characters.

Enter a password

We do not store your password and cannot retrieve it. If you forget your password, you can simply change it on the "Edit Cluster" page or from Clusters Action menu.

Tags

Add Tag

Deployment Location

Location

managed-devspatcher

Database Type

Please reference [this page](#) for more information on database types.

Postgres Type

PostgreSQL EDB Postgres Extended Server EDB Postgres Advanced Server

Postgres Version Software Image

Select... Select...

16 is the minimum supported version for Lakehouse clusters

Instance Size

CPU (0.1-15 Cores) Memory (0.1-10 Gi)

Cores Gi

Cluster Summary

Cluster Type	Single Node
Cluster Nodes	1 node
Cluster Name	--
Tags	--
Deployment Location	managed-devspatcher
Postgres Type	--
Postgres Version	--
CPU Cores	--
Memory	--
Networking	Private
Backups Retention	30 Days
Scheduled Backup Time	21:49

Fig. 3: Create Lakehouse Node Wizard Step 2

You then return to the **Clusters** page, where you can see the status of your cluster as it's created. Provisioning the cluster can take 10-15 minutes.

A few notes:

- You must provide your own password. Create a secure password and save it to your password manager. HCP doesn't save passwords.
- You can choose EDB Postgres Advanced Server or EDB Postgres Extended Server.
- All data on the NVMe drives is lost on cluster shutdown.
- Persistent data in system tables (users, roles, and so on) is stored in an attached block storage device and will survive a restart or backup/restore cycle.
- Only Postgres 16 is supported.

2.3 Operating a Lakehouse node

2.3.1 Connect to the node

You can connect to the Lakehouse node with any Postgres client in the same way that you connect to any other cluster from EDB Postgres AI Platform. Navigate to the cluster detail page and copy its connection string.

For example, you might copy the `.pgpass` blob into `~/.pgpass` (making sure to replace `$YOUR_PASSWORD` with the password you provided when launching the cluster). Then you can copy the connection string and use it as an argument to `psql` or `pgcli`.

In general, you can connect to the database with any Postgres client. We expect all introspection queries to work, and if you find one that doesn't, then that's a bug.

2.3.2 Understand the constraints

- Every cluster uses EDB Postgres Advanced Server or EDB Postgres Extended Server. So expect to see boilerplate tables from those flavors in the installation when you connect.
- Queryable data (like the benchmarking datasets) is stored in object storage as Delta Tables. Every cluster comes preloaded to point to a storage bucket with benchmarking data inside (TPC-H, TPC-DS, Clickbench) at scale factors from 1 to 1000.
- Only AWS is currently supported. Bring-your-own-account (BYOA) isn't supported.
- You can deploy a cluster in any region that's activated in your EDB Postgres AI account. Each region has a bucket with a copy of the benchmarking data. So when you launch a cluster, it uses the benchmarking data in the location closest to it.
- The cluster is ephemeral. None of the data is stored on the hard drive, except for data in system tables, for example, roles and users and grants. If you restart the cluster or back up the cluster and then restore it, it restores these system tables. But the data in object storage remains untouched.
- The cluster supports read-only queries of the data in object storage, but it supports write queries to system tables for creating users and so on. You can't write directly to object storage. You can't create tables.
- If you want to load your own data into object storage, see [Reference - Bring your own data](#).

2.4 Inspect the benchmark datasets

Inspect the benchmark datasets. Every cluster has some benchmarking data available out of the box. If you're using `pgcli`, you can run `\dn` to see the available tables.

The available benchmarking datasets are:

- TPC-H, at scale factors 1, 10, 100, and 1000
- TPC-DS, at scale factors 1, 10, 100, and 1000
- ClickBench, 100 million rows
- 1 Billion Row Challenge

For more details on benchmark datasets, see [Reference - Available benchmarking datasets](#).

2.5 Query the benchmark datasets

You can try running some basic queries:

```
edb_admin> select count(*) from clickbench.hits;
+-----+
| count |
+-----+
| 99997497 |
+-----+
SELECT 1
Time: 0.945s

edb_admin> select count(*) from brc_1b.measurements;
+-----+
| count |
+-----+
| 1000000000 |
+-----+
SELECT 1
Time: 0.651s
```

!!!note

See [PGAA functions reference](#) for more caveats and information about syntax/query compatibility. !!!

Tiered Tables enable EDB Postgres® to manage large, time-based datasets efficiently by automatically moving “cold” data to cost-effective object storage, while keeping “hot” data in primary transactional storage.

This pattern optimizes both performance and cost while preserving full analytical access to the entire dataset.

For details on how Tiered Tables are implemented and managed within Hybrid Manager (HM), see [Tiered Tables](#) .

3.1 What are Tiered Tables

Tiered Tables are a native capability of EDB Postgres Distributed (PGD), supported by the Analytics Accelerator architecture.

They automatically offload older partitions of time-partitioned tables from PGD to object storage (Apache Iceberg format) using:

- PGD AutoPartition for automated partitioning and lifecycle control
- PGAA and PGFS for querying and accessing offloaded data
- Optional Iceberg catalogs for governance and interoperability

The result: seamless, transparent access to data across hot (PGD) and cold (object storage) tiers.

Related concept: [Data tiering](#)

3.2 Why Tiered Tables matter for EDB analytics

Tiered Tables help organizations:

- Keep PGD operational storage lean and performant
- Lower storage costs by offloading old data to object storage
- Maintain unified query access to the full dataset
- Support both OLTP and OLAP use cases on Postgres

- Implement lakehouse architectures for historical analysis

Related concept: *Analytics Accelerator concepts*

3.3 How EDB implements Tiered Tables

Core components:

- PGD AutoPartition:
- Creates new time-based partitions automatically
- Defines `analytics_offload_period` to control offload timing
- PGFS:
- Provides access to object storage for offloaded data
- PGAA:
- Enables unified querying across PGD and Iceberg tiers
- Creates an offloaded view (`table_offloaded`) for cold data only
- Optional Iceberg catalog:
- Supports governance and cross-platform interoperability

Query behavior:

- Queries on the parent PGD table automatically access both hot and cold data.
- The PGD query planner pushes WHERE clauses to optimize access across storage tiers.

Related concepts:

- *Separation of storage and compute*
- *ETL / ELT*
- *OLAP / OLTP*

3.4 Common use cases

Use case	Tiered Tables + Analytics Accelerator
IoT and telemetry	Manage large time-series datasets with automated offload
Regulatory and financial data retention	Cost-efficient storage of historical data with full auditability
Analytical reporting on historical data	Use Lakehouse nodes to query offloaded data at scale
Hybrid OLTP / OLAP patterns	Keep current data fast on PGD, analyze large history in Iceberg

3.5 Role-based guidance

Database administrators (DBAs) *Analytics Accelerator for your role: DBA*

Data scientists / analysts *Analytics Accelerator for your role: Data scientist / analyst*

DevOps / SRE *Analytics Accelerator for your role: DevOps / SRE*

Application developers *Analytics Accelerator for your role: Application developer*

3.6 Learning paths

Analytics Accelerator 101: Foundational concepts

Analytics Accelerator 201: Practical application and core solutions

Analytics Accelerator 301: Advanced techniques and optimization

3.7 Related concepts

- *Data tiering*
- *Separation of storage and compute*
- *ETL / ELT*
- *OLAP / OLTP*

3.8 Next steps

For Hybrid Manager users [Tiered Tables](#)

How-To guides [Configure PGFS storage for Tiered Tables](#)

[Configure PGD node group for analytics offload](#)

[Configure BDR AutoPartition with analytics offload](#)

[Query Tiered Tables from PGD and Lakehouse](#)

[Explore more in the Analytics Accelerator learning guide](#) .

Reference - EDB Postgres® AI Lakehouse

HCP Lakehouse is a powerful tool for running analytical queries on your data. It's built on top of Postgres, so you can use your existing Postgres knowledge and tools.

To get the best experience with Lakehouse, follow the [Quick start](#) to query benchmarking data. Then you can try loading your own data and [query the delta lake tables](#).

This section details some of the important things to know:

- [Benchmarking datasets](#)
- [Queries in Lakehouse](#)
- [Delta Table tools](#)
- [DirectScan and fallback modes](#)
- [Users](#)
- [Loading data](#)
- [PGAA functions reference](#)

For information on the PGFS functions for managing storage location, see the AI Factory PGFS documentation.

4.1 Benchmarking datasets

When you provision a Lakehouse node, it comes preconfigured to point to a public S3 bucket in AWS us-east-1 that contains sample benchmarking datasets.

Note

Accessing these datasets can incur costs for data transfer and be subject to cross-region latencies.

You can query tables in these datasets by referencing them with their schema name.

Schema name	Dataset
<i>tpcds_sf_1</i>	TPC-DS, Scale Factor 1
<i>tpcds_sf_10</i>	TPC-DS, Scale Factor 10
<i>tpcds_sf_100</i>	TPC-DS, Scale Factor 100
<i>tpcds_sf_1000</i>	TPC-DS, Scale Factor 1000
<i>tpch_sf_1</i>	TPC-H, Scale Factor 1
<i>tpch_sf_10</i>	TPC-H, Scale Factor 10
<i>tpch_sf_100</i>	TPC-H, Scale Factor 100
<i>tpch_sf_1000</i>	TPC-H, Scale Factor 1000
<i>clickbench</i>	ClickBench, 100 million rows
<i>brc_1b</i>	Billion row challenge

!!!note “Notes about ClickBench data”

Data columns (`EventData`) are integers, not dates.

You must quote ClickBench column names because they contain uppercase letters, but unquoted identifiers in Postgres are case-insensitive. For example:

```
❑ select "Title" from clickbench.hits;
❑ select Title from clickbench.hits; !!!
```

4.2 Queries

Read-only queries are supported. You can’t write directly to object storage. This includes all Postgres built-in functions, statements, and types. It also includes any of those provided by EDB Postgres Advanced Server or EDB Postgres Extended Server, depending on the distribution you choose to deploy.

You can’t insert, update, delete, or otherwise modify data. You can use `CREATE TABLE` but only to create a normal Postgres table on the node. The node is ephemeral and will be destroyed when you terminate it. You can also use `CREATE TABLE... USING PGAA` to create an external table that references Delta Tables in S3-compatible object storage.

You must load data into the bucket out-of-band, either with your own ETL scripts or using a third-party ETL provider that supports writing to Delta Tables or Iceberg tables.

See [Loading data](#) for more details.

One exception is Postgres system tables, such as those used for storing users, roles, and grants. These tables are stored on the local block device, which is included in backups and restores. So you can use `CREATE USER` , `CREATE ROLE` , or `GRANT USAGE` , and these users/roles/grants will survive restarts and restores.

4.3 Delta Lake Table tools

4.3.1 Creating Delta Lake Tables

Using the `deltalake` Python library

You can use the `deltalake` Python library to create Delta Tables and write to the bucket.

Using the lakehouse-loader utility

You can also use the `lakehouse-loader` utility that EDB created for this task to export data from an arbitrary Postgres instance to Lakehouse tables in a storage bucket.

For example, with the `lakehouse-loader` utility:

```
export PGPASSWORD="..."
export AWS_ACCESS_KEY_ID="..."
export AWS_SECRET_ACCESS_KEY="..."

# export other AWS envvars

./lakehouse-loader postgres-to-delta postgres://test-user@localhost:5432/test-db -q
↪ "SELECT * FROM some_table" s3://my-bucket/my_schema/my_table
```

This code exports the data from the `some_table` table in the `test-db` database to a Delta Table in the `my_schema/my_table` path in the `my-bucket` bucket.

You can then query this table in the Lakehouse node by creating an external table that references the Delta Table in the `my_schema/my_table` path. See [Querying Delta Lake Tables](#) for details on how to do this.

4.4 DirectScan and fallback modes

4.4.1 DirectScan

Postgres Lakehouse is fastest when it can “push down” your entire query to DataFusion, the vectorized query used for handling queries when possible. (In the future, you will have more fine-grained control as we add support for partial pushdowns.)

Postgres Lakehouse can execute your query in two modes. First, it attempts to run the entire query using Seafoal, a dedicated columnar database based on DataFusion. If Seafoal can’t run the entire query, for example, because it uses PostgreSQL-specific operations like JSON, then Postgres Lakehouse falls back to using the PostgreSQL executor, with Seafoal streaming full table contents to it.

If your query is extremely slow, it’s possible that’s what’s happening.

4.4.2 Using EXPLAIN to check the query plan

You can check which mode is being used by running `EXPLAIN` on the query and making sure that the top-most query node is `DirectScan`. For example:

```
explain select count from (select count(*) from tpch_sf_1.lineitem);

QUERY PLAN

-----
↪
↪
↪
↪
Aggregate (cost=167.52..167.55 rows=1 width=8)
  -> Append (cost=0.00..165.01 rows=1001 width=0)
    -> CompatScan on "16529" lineitem_2 (cost=100.00..150.00 rows=1000 width=0)
        SeafoalPlan: logical_plan
            TableScan: tpch_sf_1.lineitem projection=[l_orderkey, l_partkey, l_
↪suppkey, l_linenum, l_quantity, l_extendedprice, l_discount, l_tax, l_returnflag,
```

(continues on next page)

(continued from previous page)

```

→ l_linestatus, l_shipdate, l_commitdate, l_receiptdate, l_shipinstruct, l_shipmode,
→ l_comment]
(6 rows)

```

In this case, the query is executed by PostgreSQL, and Seafoal is involved only when scanning the table (see `CompatScan` at the bottom). The fix in this case is to explicitly name the inner `COUNT(*)` column, since Seafoal gives it an implicit name `count(*)`, whereas PostgreSQL calls it `count`:

```

edb_admin=> explain select count from (select count(*) as count from tpch_sf_1.
→lineitem);

                        QUERY PLAN
-----
DirectScan: logical_plan
  Projection:  COUNT(*) AS count
    Aggregate:  groupBy=[[]], aggr=[[COUNT(UInt8(1)) AS COUNT(*)]]
      TableScan:  tpch_sf_1.lineitem projection=[]
(4 rows)

```

Here, you can see the `DirectScan` at the top, which means that Seafoal is running the entire query.

If you're having trouble rewording your query to make it run fully on Seafoal, open a support ticket.

4.5 User management

When you provision a Lakehouse node, you must provide a password. You need the password to log in as the `edb_admin` user. The `edb_admin` user is a superuser account that lets you create users, roles, and grants.

HCP doesn't save this password.

You can either share the credentials for `edb_admin` or use it to create other users and distribute those credentials.

4.6 Loading data

4.6.1 Bringing your own data

It's possible to point your Lakehouse node at an arbitrary S3 bucket with Delta Tables inside of it. However, there are some major caveats to doing so. (These will be resolved in future releases.)

Caveats

- The tables must be stored as [Delta Lake Tables](#) within the location.
- A *Delta Lake Table* (or *Delta Table*) is a folder of Parquet files along with some JSON metadata.

Loading data into your bucket

You can use the `lakehouse-loader` utility to export data from an arbitrary Postgres instance to Delta Tables in a storage bucket. See [Delta Lake Table tools](#) for more information on how to get and use that utility.

For more details, see [querying delta tables](#).

4.7 PGAA functions reference

4.7.1 Synopsis

Lists all PGAA tables on the instance, their storage location, and path.

4.7.2 Parameters

None.

4.7.3 Returns

Column	Type	Description
<i>schema_name</i>	<i>text</i>	The schema of the table
<i>table_name</i>	<i>text</i>	The name of the table
<i>storage_location_name</i>	<i>text</i>	The name of the storage location for the table
<i>path</i>	<i>text</i>	The path to the table in the storage location
<i>managed_by</i>	<i>text</i>	The name of the user who created the table
<i>format</i>	<i>text</i>	The format of the table (currently <i>delta</i> or <i>iceberg</i>)

pgaa.lakehouse_table_stats()

4.7.4 Synopsis

Returns the size in bytes of the latest version of a Lakehouse table and the total size of all versions.

4.7.5 Parameters

Parameter	Type	Description
<i>relation</i>	<i>regclass</i>	The name of the table to get the size of

4.7.6 Returns

Column	Type	Description
<i>latest_snapshot_size</i>	<i>bigint</i>	The size in bytes of the latest snapshot of the table
<i>total_size</i>	<i>bigint</i>	The total size in bytes of all versions of the table

4.8 PGAA functions reference

4.8.1 pgaa.create_storage_location

Synopsis

Creates a new storage location that references an S3-compatible object storage location.

Parameters

Parameter	Type	Description
<i>name</i>	<i>text</i>	The name of the storage location
<i>uri</i>	<i>text</i>	The URI of the S3-compatible object storage location, for example, <i>s3://bucket-name</i> or <i>s3://bucket-name/prefix</i>
<i>options</i>	<i>json</i>	Optional settings for the storage location

Options

Option	Type	Description
<i>access_key_id</i>	<i>text</i>	The access key ID for the storage location
<i>secret_access_key</i>	<i>text</i>	The secret access key for the storage location
<i>session_token</i>	<i>text</i>	The session token for the storage location
<i>region</i>	<i>text</i>	The region for the storage location
<i>endpoint</i>	<i>text</i>	The endpoint for the storage location
<i>bucket</i>	<i>text</i>	The bucket for the storage location
<i>use_http</i>	<i>boolean</i>	Use HTTP instead of HTTPS for the storage location
<i>skip_signature</i>	<i>boolean</i>	Skip signature verification for the storage location

4.9 Supported AWS instances

When deploying a Lakehouse node, you must choose an instance type from the `m6id` family of instances. Importantly, these instances come with NVMe drives attached to them.

Instances are ephemeral. These NVMe drives are used only for spill-out space *while processing queries, and for caching Delta Tables on disk. All data on the NVMe drives will be lost when the cluster is shutdown.

System tables are persisted. Persistent data in system tables (users, roles, *etc.) is stored in an attached block storage device, and will survive a pause/resume cycle.

Supported instances

API Name	Memory	vCPUs	Cores	Storage
<i>m6id.large</i>	8.0 GiB	2 vCPUs	1	118 GB NVMe SSD
<i>m6id.xlarge</i>	16.0 GiB	4 vCPUs	2	237 GB NVMe SSD
<i>m6id.2xlarge</i>	32.0 GiB	8 vCPUs	4	474 GB NVMe SSD
<i>m6id.4xlarge</i>	64.0 GiB	16 vCPUs	8	950 GB NVMe SSD
<i>m6id.8xlarge</i>	128.0 GiB	32 vCPUs	16	1900 GB NVMe SSD
<i>m6id.12xlarge</i>	192.0 GiB	48 vCPUs	24	2850 GB (2 * 1425 GB NVMe SSD)
<i>m6id.16xlarge</i>	256.0 GiB	64 vCPUs	32	3800 GB (2 * 1900 GB NVMe SSD)
<i>m6id.24xlarge</i>	384.0 GiB	96 vCPUs	48	5700 GB (4 * 1425 GB NVMe SSD)
<i>m6id.32xlarge</i>	512.0 GiB	128 vCPUs	64	7600 GB (4 * 1900 GB NVMe SSD)

Delta Lake is an open-source table format and transaction layer that enhances modern data lakes with ACID guarantees, schema enforcement, time travel, and scalable performance.

EDB enables first-class support for querying Delta Lake tables as part of the Analytics Accelerator — allowing Postgres SQL to operate on large-scale lakehouse data.

For details on how Delta Lake is used and managed within Hybrid Manager (HM), see [Delta Lake](#) .

5.1 What is Delta Lake

Delta Lake adds database-like reliability and consistency to object storage systems such as S3, GCS, and Azure Data Lake Storage.

Key characteristics:

- ACID transactions with strong data consistency
- Schema enforcement and schema evolution support
- Time travel for querying historical versions of data
- Batch and streaming support
- Open format built on [Apache Parquet](#) with a transaction log (`_delta_log`)

Related concept: [Open table formats](#)

5.2 Why Delta Lake matters for EDB analytics

Delta Lake support enables the Analytics Accelerator to:

- Query existing Delta Lake tables in place using Postgres SQL
- Support interoperable data architectures across Postgres, Spark, Trino, Presto, and more
- Eliminate unnecessary data duplication or movement between systems

- Provide governed, versioned data for analytical queries within Lakehouse clusters

Related concept: *EDB Postgres Lakehouse*

5.3 How EDB leverages Delta Lake

EDB Postgres Lakehouse clusters provide access to Delta Lake data through the PGAA extension:

- Define Delta Lake external tables using `CREATE TABLE ... USING PGAA WITH (pgaa.format = 'delta', ...)`
- Efficiently query Parquet-backed Delta Lake tables via vectorized execution with [Apache DataFusion](#)
- Benefit from Delta Lake features including schema evolution and time travel

Current primary support: querying existing Delta Lake tables. PGD offload currently targets Iceberg; Delta read support enables integration with existing Delta-based data lakes.

5.4 Common use cases

Use case	Delta Lake + Analytics Accelerator
Business intelligence reporting	Query Delta Lake tables using Postgres SQL and BI tools
Data science and machine learning	Access Delta tables for model training and feature engineering
Data lake governance	Utilize Delta's ACID guarantees with Lakehouse SQL access
Cross-platform interoperability	Query Delta Lake data alongside Spark, Trino, and Postgres Lakehouse

5.5 Role-based guidance

- *Database administrator (DBA)*
- *Data scientist / analyst*
- *DevOps / SRE*
- *Application developer*

5.6 Learning paths

- *Analytics Accelerator 101: Foundational concepts*
- *Analytics Accelerator 201: Practical application and core solutions*
- *Analytics Accelerator 301: Advanced techniques and optimization*

5.7 Related concepts

- *Data lakehouse*
- *Columnar storage formats*
- *Separation of storage and compute*

- *Open table formats*

5.8 Next steps

For Hybrid Manager users

- *Delta Lake*

How-To guides

- *Query Delta Lake tables*
- *Configure PGFS for Delta Lake*
- *Performance tuning for Delta Lake queries*

Explore more in the Analytics Accelerator learning guide .

Apache Iceberg is a high-performance, open table format designed for data lakes built on object storage. It provides scalable metadata management, schema evolution, and ACID transactions.

EDB enables Iceberg support within the Analytics Accelerator, allowing Postgres to query and manage large data lake tables efficiently.

For implementation in Hybrid Manager (HM), see [Working with Apache Iceberg in EDB HM](#).

6.1 What is Apache Iceberg

Iceberg is a table format designed for data lakes. It defines how data files (Parquet, ORC, Avro) are organized and tracked, adding many database-like capabilities to object storage.

6.1.1 Key goals

- **Reliability:** Ensures data consistency with concurrent operations
- **Performance:** Enables fast queries through partition pruning and metadata optimizations
- **Scalability:** Designed to manage petabyte-scale tables
- **Openness:** Apache Software Foundation project with broad ecosystem support

Related concepts: [Generic concepts - open table formats](#)

6.2 Key features and benefits

- **Schema evolution:** Safely add, drop, or modify columns without rewriting table data
- **Hidden partitioning:** Partitions data transparently, enabling efficient partition pruning
- **Time travel and versioning:** Supports querying historical table versions
- **ACID transactions:** Provides transactional guarantees using optimistic concurrency
- **Multiple file format support:** Manages Parquet, ORC, and Avro files

- **Efficient metadata management:** Uses a metadata tree structure for fast query planning and data skipping
- **Catalog integration:** Supports REST-based catalogs (Project Nessie, AWS Glue), Hive Metastore, and others

6.3 Why Apache Iceberg matters for EDB analytics

Iceberg support enables the Analytics Accelerator to:

- **Unify operational and analytical data:** Query Iceberg tables alongside traditional Postgres data
- **Leverage cost-effective object storage:** Store large historical datasets efficiently
- **Improve performance:** Use Iceberg metadata and partition pruning with vectorized engines
- **Support tiered storage:** PGD can offload data to Iceberg as part of a tiered table strategy
- **Enable interoperability:** Share data with tools like Spark, Presto, Trino, Flink, and Dremio

Related concepts: [Analytics Accelerator concepts - EDB Postgres Lakehouse](#)

6.4 How EDB solutions implement Iceberg support

6.4.1 PGAA extensions

The Postgres Analytical Appliance (PGAA) components within Lakehouse nodes and PGD nodes can read and interact with Iceberg metadata and data files.

6.4.2 Catalog connectivity

PGAA connects to Iceberg catalogs (such as Lakekeeper, AWS Glue, or REST catalogs) using:

```
SELECT pgaa.add_catalog(...);
SELECT pgaa.attach_catalog(...);
```

This enables discovery of tables, reading schemas, and querying data.

Querying Iceberg tables Users can define external tables in Postgres that reference Iceberg tables:

```
CREATE TABLE my_table USING PGAA WITH (
  pgaa.format = iceberg,
  pgaa.managed_by = my_catalog,
  pgaa.catalog_namespace = my_db,
  pgaa.catalog_table = my_table
);
```

Once defined, the table is fully queryable using Postgres SQL.

6.4.3 Writing to Iceberg tables

PGD can offload data to Iceberg format in object storage:

New Iceberg tables can be created

Existing Iceberg tables can be appended to

Offload operations typically follow tiered table strategies

Related concepts: Tiered Tables

6.5 Related topics

- *Learn Generic Analytics Concepts*
- *Learn Analytics Accelerator Concepts*
- *How-To: Configure an Iceberg REST catalog connection in Hybrid Manager*

EDB Postgres Lakehouse

EDB Postgres® Lakehouse brings modern lakehouse architecture to Postgres-based analytics.

It enables fast SQL-based analytics on data stored in object storage (S3-compatible), using open table formats such as Apache Iceberg and Delta Lake.

For implementation and management in Hybrid Manager (HM), see [EDB Postgres Lakehouse](#).

7.1 What is the EDB Postgres Lakehouse

EDB Postgres Lakehouse is an architecture pattern and set of capabilities that:

- Integrate Postgres with modern lakehouse patterns
- Provide fast, scalable analytics on data in object storage
- Use vectorized query execution and columnar storage formats
- Leverage open table formats for interoperability and data governance

Related concept: [Data lakehouse](#)

7.2 Why Lakehouse matters for EDB analytics

Lakehouse enables Analytics Accelerator users to:

- Run fast SQL analytics on object storage — no data movement required
- Implement separation of storage and compute for cost efficiency
- Support interoperability with external tools (Spark, Trino, Flink, data science frameworks)
- Query PGD Tiered Tables offloaded to Iceberg seamlessly
- Enable unified OLTP + OLAP architectures with Postgres at the center

Related concept: [Analytics Accelerator concepts](#)

7.3 How EDB implements Lakehouse architecture

Core components:

- **EDB Postgres Lakehouse Nodes:**
 - Stateless analytical compute nodes
 - Provisioned and managed via Hybrid Manager (HM) or self-managed
- **Vectorized query engine:**
 - Powered by [Apache DataFusion](#)
 - Processes data in Parquet and other columnar formats efficiently
- **PGAA:**
 - Postgres extensions enabling Lakehouse behavior:
 - External tables over Iceberg and Delta Lake
 - Unified access to PGD hot data + offloaded cold data
- **PGFS:**
 - Unified access layer to object storage
 - Supports S3, GCS, MinIO, Azure Data Lake Storage, and compatible systems
- **Open table formats:**
 - Apache Iceberg (full support including catalogs)
 - Delta Lake (read-only support)

Related concepts:

- *Separation of storage and compute*
- *Vectorized query engines*
- *Columnar storage formats*
- *Open table formats*

7.4 Common use cases

Use case	EDB Postgres Lakehouse capability
Business intelligence & reporting	Run fast, scalable SQL on data in object storage
Historical analytics	Seamlessly query offloaded PGD Tiered Tables
Data lake analytics	Query existing Iceberg and Delta Lake tables without ETL
Data science pipelines	Provide efficient access to training data and features
Hybrid architectures	Enable Postgres-centered OLTP + OLAP patterns

7.5 Role-based guidance

Database administrators (DBAs)

- *Analytics Accelerator for your role: DBA*

Data scientists / analysts

- *Analytics Accelerator for your role: Data scientist / analyst*

DevOps / SRE

- *Analytics Accelerator for your role: DevOps / SRE*

Application developers

- *Analytics Accelerator for your role: Application developer*

7.6 Learning paths

- *Analytics Accelerator 101: Foundational concepts*
- *Analytics Accelerator 201: Practical application and core solutions*
- *Analytics Accelerator 301: Advanced techniques and optimization*

7.7 Related concepts

- *Data lakehouse*
- *Separation of storage and compute*
- *Vectorized query engines*
- *Columnar storage formats*
- *Open table formats*

7.8 Next steps

For Hybrid Manager users

- *EDB Postgres Lakehouse*

How-To guides

- *Create a Lakehouse cluster*
- *Query existing Apache Iceberg tables*
- *Query Delta Lake tables*

Explore more in the Analytics Accelerator learning guide .

Analytics Accelerator learning resources

Use this guide to navigate all documentation for the Analytics Accelerator, built on Postgres®. Whether you want to understand core concepts, follow step-by-step tutorials, solve specific problems, or explore how analytics applies to your role, this guide helps you find the right resources.

The documentation is structured to help you learn effectively and find information efficiently.

8.1 Documentation categories

8.1.1 Structured learning paths

Follow structured learning paths to progress from foundational concepts to advanced implementations.

Explore learning paths: *Analytics Accelerator learning paths*

- *Analytics Accelerator 101: Foundational concepts*
- *Analytics Accelerator 201: Practical application and core solutions*
- *Analytics Accelerator 301: Advanced techniques and optimization*

8.1.2 Conceptual explanations

Build foundational knowledge of analytics principles, modern data architectures, and Analytics Accelerator technologies and strategies.

Explore: Explained: Analytics Accelerator concepts and technologies

Covers generic analytics concepts, the Analytics Accelerator vision, and deep dives into core technologies such as EDB Postgres Lakehouse, Apache Iceberg, Delta Lake, Tiered Tables, and architecture patterns.

8.1.3 Tutorials

Learn by doing through complete projects or scenarios. Build practical skills and apply analytics in your environment.

Explore: [Analytics Accelerator tutorials index](#)

8.1.4 How-to guides

Find concise, step-by-step instructions for common tasks related to the Analytics Accelerator.

Explore: [Analytics Accelerator how-to guides index](#)

8.1.5 Use cases and solutions

Discover how the Analytics Accelerator addresses business problems and supports architecture patterns across industries.

Explore: [Analytics Accelerator use cases](#)

8.1.6 Persona-based guidance

Access curated information and learning paths tailored to your technical role.

Explore: [Analytics Accelerator for your role](#)

8.1.7 Analytics Accelerator in Hybrid Manager (HM)

For users of EDB's managed services, this section provides documentation on how Analytics Accelerator concepts and technologies are implemented, provisioned, and managed in Hybrid Manager.

Explore: [Analytics in Hybrid Manager](#)

Analytics Accelerator learning resources

Use this learning guide to chart your path through the Analytics Accelerator documentation — whether you're starting with core concepts or going deep on advanced architectures.

9.1 Analytics Accelerator concepts and technologies

Use this section to build a clear understanding of the **core concepts**, **technologies**, **architectural patterns**, and **strategies** that define the Analytics Accelerator.

This section complements:

- [How-To Guides](#) — for step-by-step tasks
- [Analytics Terminology](#) — for key terms used in this space

For explanations specific to the EDB Hybrid Manager (HM) environment, see [Analytics in Hybrid Manager](#) .

9.2 Analytics Accelerator concepts and technologies

9.2.1 Foundational analytics and modern data architectures

Understand the **industry trends** and **architectural patterns** that shape today's analytics landscape.

Generic concepts

- [Analytics Accelerator generic concepts](#)

Foundational terms such as data warehouses, data lakes, lakehouse architecture, columnar storage, vectorized engines, and separation of storage and compute.

Additional articles (coming soon)

- The evolution to the data lakehouse
- The significance of open table formats (Iceberg, Delta Lake, Hudi)
- Benefits and trade-offs of columnar vs. row-oriented storage for analytics

9.3 Analytics Accelerator concepts and technologies

9.3.1 Deep dive into core Analytics Accelerator components

Learn about the **core components** that enable advanced analytics on EDB Postgres.

EDB Postgres Lakehouse

- EDB Postgres Lakehouse: An overview (coming soon)
- EDB Postgres Lakehouse: Detailed concepts and terminology (coming soon)

Open table formats with EDB Postgres

- Understanding Apache Iceberg with EDB solutions (coming soon)
- Understanding Delta Lake with EDB solutions (coming soon)

Data management and tiering with EDB Postgres Distributed (PGD)

- Understanding Tiered Tables with EDB Postgres (coming soon)

Underlying engine components

- The role of PGAA and PGFS in EDB's Lakehouse (coming soon)
- Vectorized query execution with Apache DataFusion in EDB Postgres (coming soon)

9.4 Analytics Accelerator concepts and technologies

9.4.1 Understanding analytics within EDB Hybrid Manager (HM)

How EDB's analytics technologies are **implemented and managed in Hybrid Manager**.

Key HM analytics documentation entry points

- [Analytics in Hybrid Manager](#)
- [Analytics concepts within Hybrid Manager](#)

9.4.2 AI/ML workloads and interoperability

The Analytics Accelerator supports **general-purpose analytics** and can also serve as a platform component in AI/ML pipelines:

- Lakehouse nodes provide efficient access to large datasets used in model training.
- Tiered data patterns and ELT pipelines can stage data for AI/ML processing.

For AI/ML-specific concepts, see [AI Factory concepts](#) .

Analytics Accelerator concepts

This page explains how EDB delivers modern analytics capabilities through the **Analytics Accelerator** and **Hybrid Manager (HM)**.

It builds on established industry patterns and technologies. If you're new to the space, start with:

- *Analytics Terminology* — learn the key terms used
 - *Analytics Accelerator generic concepts* — understand common architectures and patterns
-

EDB's vision for analytics on Postgres

EDB implements modern analytics patterns through the Analytics Accelerator, enabling Postgres to serve as a unified platform for both operational and analytical workloads.

Key principles:

- Perform analytics **close to operational data**, reducing data movement and latency
- Use **open formats** and **vectorized query engines** for fast analytics on object storage
- Support **data tiering** to balance cost and performance
- Separate **compute and storage** to allow independent scaling
- Manage the entire stack via Hybrid Manager

Related concept: *Data lakehouse*

Key EDB solutions and technologies

EDB Postgres Lakehouse

Lakehouse architecture enables EDB Postgres to query data in object storage:

- Lakehouse nodes query data in **open table formats** (Iceberg, Delta Lake)
- Vectorized query execution powered by *Apache DataFusion*
- Columnar storage formats such as **Parquet**
- Separation of compute and storage

Related concepts:

- *Columnar storage formats*
 - *Vectorized query engines*
 - *Separation of storage and compute*
 - *Open table formats*
-

Vectorized query optimization

Analytics Accelerator embeds [Apache DataFusion](#) in Lakehouse nodes:

- Processes data as **columnar batches**
- Uses **SIMD** instructions to accelerate performance
- Optimized for **Parquet-formatted data**

Related concept: [Vectorized query engines](#)

EDB Postgres Distributed (PGD) and Tiered Tables

EDB Postgres Distributed (PGD) powers Tiered Tables:

- **BDR AutoPartition** manages time-based partitioning
- Cold data is automatically **offloaded to object storage** (Iceberg tables)
- Hot data remains in transactional PGD nodes

This balances operational performance with cost-efficient historical data access.

Related concepts:

- [Data tiering](#)
 - [ETL / ELT](#)
-

Advanced query optimization

Analytics Accelerator improves core Postgres capabilities for analytics:

- Enhanced parallel query processing
- Optimized join algorithms for analytical workloads
- Tight integration with vectorized engines for high-performance OLAP queries

Related concept: [OLAP / OLTP](#)

Analytics and AI/ML workloads

Analytics Accelerator supports traditional analytics and AI/ML data pipelines:

- Lakehouse nodes provide efficient access to large datasets used in model training
- Tiered data pipelines support cost-effective AI/ML staging patterns
- Open table formats enable integration with vector search and **retrieval augmented generation (RAG)** workflows

Related concepts:

- [Semantic search](#)
 - [Vector embeddings](#)
 - [Retrieval augmented generation \(RAG\)](#)
-

How EDB implements analytics across products

Hybrid Manager (HM)

Hybrid Manager acts as the control plane:

- Provision and manage **Lakehouse clusters**
- Configure and monitor **Tiered Tables**
- Manage **storage locations**, **catalog connections**, and **compute nodes**

EDB Postgres Advanced Server / Extended Server

EDB Postgres Advanced Server adds:

- Improved parallel query capabilities
- Advanced SQL features helpful for analytics
- Native support for PGAA (Postgres Generic Analytics Adapter) features

EDB Postgres Distributed (PGD)

PGD provides the foundation for:

- High-availability transactional workloads
 - Data tiering and offloading (Tiered Tables)
 - Scalable architectures that integrate with Lakehouse clusters
-

Explore next

- [Analytics Terminology](#)
- [Analytics Accelerator generic concepts](#)
- [Analytics How-To Guides](#)
- [Analytics Use Cases](#)

Analytics Accelerator generic concepts

This hub explains key industry concepts, architectures, and technologies that form the foundation of modern data analytics platforms.

Understanding these concepts helps you see how EDB's Analytics Accelerator builds on proven patterns. To learn how EDB implements these ideas, visit the [Analytics Accelerator concepts](#).

Architectures

Data warehouses

What: Centralized repositories optimized for structured, SQL-based analytics. They use schema-on-write and are designed for fast analytical queries.

Why: Business intelligence teams need trusted, consistent data for reporting and historical analysis.

How: Data is loaded through ETL processes into relational tables with strong typing and constraints.

Why use it: When you need high performance for known, consistent reporting needs and strong governance over data quality.

Learn how this influences [EDB Postgres Lakehouse](#) .

Data lakes

What: Flexible repositories for raw data (structured, semi-structured, unstructured). Schema-on-read.

Why: Data lakes let you store all your data at low cost, in its native format.

How: Data is loaded to object storage (e.g., S3, GCS) as files, often in Parquet or ORC format.

Why use it: When you need to capture all available data — even before knowing the full set of use cases — and run flexible processing later.

[More on data lakes](#)

Learn how this influences [EDB Postgres Lakehouse](#) .

Data lakehouse

What: A modern architecture that combines the flexibility of data lakes with the performance and reliability of data warehouses.

Why: Organizations want one system for both data science and BI, without moving large amounts of data.

How: Data is stored in the lake (object storage) using open table formats that add structure, transactions, and metadata.

Why use it: When you want scalable, low-cost storage but also need ACID guarantees and high-performance queries.

[Lakehouse architecture paper](#) (Armbrust et al.)

[Apache Iceberg](#)

[Delta Lake](#)

[Apache Hudi](#)

Learn how this influences [EDB Postgres Lakehouse](#) .

Key technologies

Columnar storage formats

What: Data formats that store columns together rather than rows.

Why: Most analytical queries only access a subset of columns. Reading columns directly improves performance and reduces I/O.

How: Popular formats are [Apache Parquet](#) and [Apache ORC](#) , which support compression and column pruning.

Why use it: When you run large, read-heavy analytical queries and want maximum scan speed.

See how this is used in [EDB Postgres Lakehouse](#) .

Vectorized query engines

What: Query engines that process data in columnar batches (vectors) instead of row by row.

Why: Columnar batches match CPU cache behavior and SIMD instructions, making analytics much faster.

How: Engines like [Apache DataFusion](#) and [Velox](#) implement vectorized execution.

Why use it: When your analytical workload is CPU-bound and performance matters.

See how this is used in [Vectorized query optimization](#) .

Separation of storage and compute

What: Architectures where compute (query engines) and storage (object storage) are decoupled.

Why: This allows you to scale them independently, optimizing cost and performance.

How: Systems like the Analytics Accelerator use object storage (e.g., S3) for data and stateless compute nodes for queries.

Why use it: When your compute needs are bursty or unpredictable, and you want to avoid over-provisioning.

See how this is used in [EDB Postgres Lakehouse](#) .

Open table formats

What: Formats that bring database features (ACID transactions, schema evolution) to data lakes.

Why: Without an open table format, lake data is a set of loose files. You need structure and governance.

How:

- [Apache Iceberg](#) uses metadata layers and manifest files
- [Delta Lake](#) uses transaction logs ([delta-rs](#))
- [Apache Hudi](#) focuses on stream-first updates

Why use it: When you want strong data management for lake-based data.

See how this is used in [EDB Postgres Lakehouse](#) .

Core concepts

OLAP / OLTP

What:

- OLAP: Analytical queries (aggregations, trends) on large datasets
- OLTP: High-volume, small transactions (updates, inserts)

Why: Different workloads have different performance profiles.

How: OLAP is optimized with columnar storage and parallel execution. OLTP uses row-based storage and transactional guarantees.

Why use it: To choose the right architecture for each type of workload.

See how this is used in [Advanced query optimization](#) .

ETL / ELT

What:

- ETL: Extract, Transform, Load
- ELT: Extract, Load, Transform (in the lake or warehouse)

Why: ELT takes advantage of modern lakehouses' ability to process raw data at scale.

How: Data is staged in object storage, then transformed using query engines or pipelines.

Why use it: To simplify data ingestion and make pipelines more flexible.

See how this is used in [Analytics Accelerator concepts](#) .

Data tiering

What: Classifying data into hot, warm, cold based on access patterns.

Why: Not all data needs fast storage — moving cold data to cheaper storage saves cost.

How: Data is physically separated across storage types (fast SSDs, object storage).

Why use it: To optimize storage cost vs. performance.

See how this is used in [Analytics Accelerator concepts](#) .

Semantic search

What: Search that understands meaning, not just keywords.

Why: Users expect smarter search experiences.

How: Data is represented as vector embeddings and compared using similarity search.

Why use it: To support search applications in AI and modern apps.

See how this is used in [Analytics Accelerator concepts](#) .

Vector embeddings

What: Numerical representations of data (text, images, audio) in high-dimensional space.

Why: Enable similarity comparisons that go beyond exact matches.

How: Libraries like [FAISS](#) , [Pinecone](#) , and many ML models generate and use embeddings.

Why use it: For AI/ML search and retrieval tasks.

See how this is used in [Analytics Accelerator concepts](#) .

Retrieval augmented generation (RAG)

What: AI pattern where an LLM retrieves documents from an external source before answering.

Why: Makes LLM responses more accurate and grounded in your data.

How: Vector search retrieves relevant embeddings; LLM combines them with the prompt.

Why use it: To improve factual accuracy in AI-driven apps.

[Intro to RAG](#)

See how this is used in [Analytics Accelerator concepts](#) .

Additional concepts

Multi-engine interoperability

What: Architecture where multiple query engines can access shared data in the lake.

Why: Organizations want flexibility — different tools for different use cases (SQL engines, data science notebooks, streaming engines).

How: Open table formats (Iceberg, Delta Lake) enable multi-engine access:

- Spark writes data → Trino, Postgres Lakehouse, and other engines can query it.
- PGD offloads data → Lakehouse clusters and external engines can analyze it.

When to use: Build flexible data architectures without vendor lock-in.

Data freshness and latency

What: Ability to query data as soon as it lands in the lake, with minimal delay.

Why: Business users increasingly expect near-real-time access to new data.

How: Lakehouse architectures support this:

- ELT pipelines can continuously update lake data.
- Open table formats support atomic commits and transaction visibility.
- Vectorized Lakehouse engines can query updated data immediately.

When to use: Analytics on operational or streaming data, without requiring full warehouse ETL cycles.

Explore next

- [Analytics Accelerator concepts](#)
- [Analytics How-To Guides](#)
- [Analytics Terminology](#)

Analytics Terminology

This page defines key terms used in the Analytics Accelerator and Hybrid Manager (HM) analytics features.

Use it as a quick reference when exploring concepts, how-tos, and feature guides.

Apache Iceberg

An open table format for large analytic datasets stored in object storage. Provides schema evolution, time travel, and interoperability with many analytics engines (Spark, Trino, Flink, and Postgres).

[Apache Iceberg](#)

Delta Lake

An open table format and storage layer that adds ACID transactions and reliability to data lakes. Built on Parquet files with a `_delta_log` transaction log.

[Delta Lake](#)

Data Lakehouse

A modern architecture that combines elements of data lakes and data warehouses:

- Stores data in object storage
- Supports open table formats (Iceberg, Delta Lake)
- Provides fast SQL analytics using vectorized query engines
- Decouples storage from compute

Related concept: [Data lakehouse](#)

EDB Postgres Lakehouse Cluster

A managed analytical compute cluster provisioned by Hybrid Manager to run fast SQL queries on data stored in object storage.

- Supports Iceberg and Delta Lake formats
- Uses a vectorized query engine (Apache DataFusion)
- Separates storage and compute for scalability and cost efficiency

[EDB Postgres Lakehouse](#)

EDB Postgres Distributed (PGD)

An advanced, distributed version of Postgres:

- Provides high availability and multi-master replication
- Enables advanced data tiering patterns (Tiered Tables)
- Works seamlessly with Lakehouse clusters for scalable analytics

[Learn more](#)

Tiered Tables

A feature that uses PGD **AutoPartition** to manage large time-based datasets:

- **Hot data** stays in the PGD transactional cluster
- **Cold data** is offloaded to object storage as Apache Iceberg tables
- Queries can transparently access both hot and cold data

[EDB Postgres Distributed \(PGD\) and Tiered Tables](#)

PGAA (Postgres Generic Analytics Adapter)

An extension that enables Postgres to:

- Query open table formats (Iceberg, Delta Lake)
- Define **external tables** that map to object storage
- Power Lakehouse cluster queries
- Enable PGD Tiered Table offloading and querying

Example in use

PGFS (Postgres File System)

An extension that defines **storage locations** Postgres can use to access object storage.

- Used by PGAA and PGD for Iceberg and Delta Lake access
- Supports AWS S3, Google Cloud Storage, and compatible services
- Must be configured on each Lakehouse or PGD cluster that needs to query object storage

Example in use

AutoPartition

A capability of PGD BDR (Bi-Directional Replication):

- Automatically creates time-based partitions for a PGD table
- Automatically offloads older partitions to object storage when configured with an `analytics_offload_period`

Example in use

BDR Analytics Table

An internal PGD concept:

- The **analytics_table** view tracks which PGD tables are marked for Tiered Table offload
- Tracks the state of each table and offload progress

Useful for monitoring and validating Tiered Table behavior.

Example in use

Iceberg Catalog

A metadata service that tracks Iceberg table schemas, locations, and versions:

- Required for full interoperability across multiple engines (Spark, Trino, Postgres)
- Can be HM-managed (Lakekeeper) or external (AWS Glue, Nessie, Polaris, etc.)
- PGAA and PGD both support connecting to an Iceberg REST catalog

Example in use

Lakekeeper

An HM-managed Iceberg catalog service:

- Provides a central catalog for Iceberg tables
- Supports PGD Tiered Table offload with catalog integration
- Supports Lakehouse cluster queries through catalog-based table discovery
- Uses Iceberg REST Catalog API

Example in use

Open Table Formats

Standardized file layouts and metadata formats used for analytics:

- **Apache Iceberg**
- **Delta Lake**

Enable multi-engine access to the same data (Spark, Trino, Postgres Lakehouse, etc.).

Open table formats

Vectorized Query Engine

An analytics engine that processes data in **columnar batches** (instead of row-by-row):

- Accelerates analytics queries
- Makes use of modern CPU features (SIMD)
- Powers Lakehouse cluster query performance

EDB Lakehouse clusters embed [Apache DataFusion](#) for this.

Data Tiering

The practice of storing **hot**, **warm**, and **cold** data on different storage tiers:

- Hot → fast transactional PGD nodes
- Cold → object storage (Iceberg tables)
- Tiered Tables automate this pattern in Postgres + HM

Related concept: [Data tiering](#)

Separation of Storage and Compute

A pattern used by Lakehouse architectures:

- Store data in **object storage** (independently scaled)
- Scale analytical compute (Lakehouse clusters) independently of storage

Enables cost-efficient, elastic analytics architectures.

Related concept: [Separation of storage and compute](#)

Next steps

Now that you understand key terminology:

- [Learn Analytics Concepts](#)
- [EDB Postgres Lakehouse](#)
- [EDB Postgres Distributed \(PGD\) and Tiered Tables](#)
- [See How-To Guides](#)

9.5 Analytics Accelerator learning paths

Use these learning paths to build knowledge progressively — from foundational concepts to advanced implementations and optimization techniques.

9.5.1 How to use these learning paths

Select the level that matches your current knowledge and goals. Each level links to curated guides with conceptual explanations, how-to articles, and practical tutorials.

For advanced topics and certification, explore official EDB training programs.

9.6 Analytics Accelerator learning paths

Target audience: Users with foundational knowledge ready to apply core technologies practically. Suitable for DBAs, data engineers, and developers.

Focus: Hands-on use of EDB Postgres Lakehouse, PGD analytics (including Tiered Tables), and Iceberg/Delta Lake integration. Basic use cases and configurations.

What you will learn:

- Set up and query EDB Postgres Lakehouse clusters
- Work with Apache Iceberg and Delta Lake tables
- Implement basic Tiered Table configurations using EDB Postgres Distributed (PGD)
- Connect BI tools to the Analytics Accelerator
- Apply analytics to BI reporting, historical data analysis, and operational analytics

Continue your journey: [Analytics Accelerator concepts](#) + [How-To Guides](#)

9.7 Analytics Accelerator learning paths

9.7.1 EDB training and certification

For expert-led instruction and official certification, explore EDB’s training programs and certifications.

- [Analytics/Lakehouse \(Coming soon\)](#)
- [Analytics/Lakehouse \(Coming soon\)](#)

9.8 Analytics Accelerator learning paths

9.8.1 Next steps

- Build your foundation with [Analytics Accelerator generic concepts](#)
- Learn EDB’s approach in [Analytics Accelerator concepts](#)
- Apply your knowledge with [How-To Guides](#)
- Explore industry applications in [Analytics Accelerator use cases](#)

Use these learning paths to progress your Analytics Accelerator expertise — from foundational knowledge to advanced optimization and architecture.

Foundational concepts

Start here to learn the core ideas that underpin the Analytics Accelerator. This level is designed for those new to modern analytics concepts, or those seeking a refresher.

What you will learn

- Fundamental concepts of data warehouses, data lakes, and lakehouse architectures
- Columnar storage and vectorized query engines
- Open table formats: Apache Iceberg and Delta Lake
- The EDB Analytics Accelerator vision and core architecture
- The value of separating storage and compute for analytics

Before you begin

- [Analytics Accelerator generic concepts](#)
- [Analytics Accelerator concepts](#)

Suggested learning path

1. *Analytics Accelerator generic concepts*
2. *Analytics Accelerator concepts*
3. Understanding Apache Iceberg with EDB solutions (*section link — to be provided*)
4. Understanding Delta Lake with EDB solutions (*section link — to be provided*)
 - *EDB Postgres Lakehouse*

Next steps

Once you've completed 101, continue with:

Practical application and core solutions

Build on foundational knowledge and begin applying key Analytics Accelerator technologies in your environment.

What you will learn

- How to set up and query EDB Postgres Lakehouse clusters
- How to work with Apache Iceberg and Delta Lake tables
- How to implement tiered tables with PGD
- How to connect BI tools to the Analytics Accelerator
- How to apply these technologies to BI reporting, historical data analysis, and operational analytics

Before you begin

- *Analytics Accelerator concepts*

Suggested learning path

1. *Create a Lakehouse cluster*
2. *Configure an Iceberg REST catalog connection*
3. *Query existing Apache Iceberg tables*
4. *Query Delta Lake tables*
5. *Configure PGD node group for analytics offload*
6. *Configure BDR AutoPartition with analytics offload*
7. *Query Tiered Tables from PGD and Lakehouse*
8. *Connect BI tools to the Lakehouse (as appropriate)*

Next steps

Once you've completed 201, continue with:

□

Advanced techniques and optimization

For experienced users, architects, and engineers ready to deepen their expertise with advanced optimization and architecture patterns.

What you will learn

- Advanced configuration of EDB Postgres Lakehouse, PGAA, and PGFS
- Performance tuning for vectorized queries and object storage access
- Complex tiered table strategies for optimized cost and performance
- Managing advanced Iceberg catalogs and integrating third-party catalogs
- Architecting solutions that combine Analytics Accelerator with other data frameworks (Spark, streaming platforms)
- Troubleshooting complex analytical queries and data pipelines
- Security best practices for analytics environments

Before you begin

- [Analytics Accelerator concepts](#)

Suggested learning path

1. *Performance tuning for Delta Lake queries*
2. *Monitor Tiered Tables status and storage savings*
3. *Advanced Iceberg catalog management (section link — to be provided)*
4. *Architectural patterns for real-time vs. batch analytics (section link — to be provided)*
5. *Analytics use cases and industry solutions*
6. *Analytics Accelerator for your role*
7. *Security best practices for analytics environments (section link — to be provided)*

Next steps

Consider official training and certification:

- [EDB Training portal](#)

9.9 a persona-based guide

Use this guide to understand how to leverage EDB Postgres® AI and analytics capabilities for your specific role. EDB Postgres AI provides a powerful platform for a wide range of analytical needs. This guide helps you focus on the capabilities that align with your goals.

9.9.1 Quickly find your role

- *Database administrator (DBA)*
- *DevOps engineer / site reliability engineer (SRE)*
- *Data scientist / data analyst*

- *Application developer*

For details on how to implement and manage these solutions within EDB Hybrid Manager (HM), see Analytics Accelerator for your role: a persona-based guide .

9.9.2 Database administrator (DBA)

Focus on performance, reliability, cost efficiency, and manageability of your database infrastructure.

Key objectives and Analytics Accelerator solutions

Performance monitoring and tuning

Identify slow queries, bottlenecks, and inefficient resource usage. Use HM dashboards to monitor performance of EDB Postgres Lakehouse Nodes and PGD clusters. Analytics Accelerator for your role: a persona-based guide .

Storage management and cost optimization

Manage large databases and historical data cost effectively. Use tiered tables with PGD and EDB Postgres Lakehouse to offload cold data to object storage (as Iceberg tables). Analytics Accelerator for your role: a persona-based guide . Manage tiered tables via HM.

Capacity planning

Forecast storage and compute needs. Analyze historical growth patterns and current utilization via Lakehouse and HM dashboards. Leverage the separation of compute and storage in Lakehouse for flexible scaling.

Backup and recovery of analytical data

Protect data in object storage and associated metadata. Follow Iceberg catalog backup strategies and HM backup options. Analytics Accelerator for your role: a persona-based guide .

Ensuring data accessibility for analytics

Provide performant and secure access to data analysts and data scientists. Use Lakehouse Nodes and PGD offloading to minimize impact on transactional systems.

Recommended resources

- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide

9.9.3 DevOps engineer / site reliability engineer (SRE)

Focus on automation, infrastructure management, scalability, reliability, and observability.

Key objectives and Analytics Accelerator solutions

Automated provisioning and management

Use Hybrid Manager to provision Lakehouse clusters and related resources via the console, API, or CLI. Analytics Accelerator for your role: a persona-based guide .

Scalability and elasticity

Independently scale Lakehouse compute resources to meet demand.

Monitoring and observability

Gain visibility into performance, health, and cost of analytical systems. Use HM dashboards and logging. Analytics Accelerator for your role: a persona-based guide .

CI/CD for data pipelines and analytical applications

Integrate schema changes (Iceberg evolution) and data offloading via PGD into CI/CD pipelines.

Cost management for cloud resources

Track and optimize costs for object storage and analytical compute. Use HM resource insights.

Recommended resources

- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide

9.9.4 Data scientist / data analyst

Focus on extracting insights, building models, and answering business questions.

Key objectives and Analytics Accelerator solutions

Accessing large and diverse datasets

Query large datasets stored as Iceberg or Delta Lake tables via Lakehouse Nodes. Use Python, R, SQL clients, and BI tools.

Data preparation and exploration

Perform data filtering, aggregation, and transformation with SQL via Lakehouse Nodes.

Working with open table formats

Leverage schema evolution, time travel, and ACID transactions in Iceberg and Delta Lake tables. Analytics Accelerator for your role: a persona-based guide . Analytics Accelerator for your role: a persona-based guide .

Integration with data science tools and frameworks

Load Lakehouse query results into pandas, R data frames, or use with Spark, TensorFlow, or PyTorch.

Query performance

Accelerate analytical SQL queries with vectorized query engines in Lakehouse Nodes.

Recommended resources

- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide

9.9.5 Application developer

Focus on building data-driven applications and embedding insights.

Key objectives and Analytics Accelerator solutions

Building applications with analytical features

Use Lakehouse Nodes or PGD read replicas to provide reporting, dashboards, and data exploration in applications.

Efficiently handling large reporting queries

Offload large aggregation and historical data queries to Lakehouse Nodes. Use tiered tables to direct historical queries to an analytical tier.

Accessing data from a unified platform

Provide consistent SQL access to both operational and analytical data.

Developing AI/ML-enabled applications

Store and access ML model data using Lakehouse Nodes.

Recommended resources

- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide
- Analytics Accelerator for your role: a persona-based guide

Explore these resources for more detailed guidance. For HM-specific implementation steps, see [Analytics Accelerator for your role: a persona-based guide](#) .

9.10 Analytics Accelerator tutorials

Use this section to explore in-depth, step-by-step tutorials for building practical solutions with the Analytics Accelerator.

Tutorials are learning-oriented and guide you through achieving meaningful outcomes. They often cover multiple features and concepts working together to solve larger problems.

For concise, goal-oriented instructions on specific tasks, see the [How-To guides](#) . For tutorials specific to Hybrid Manager (HM), refer to the [HM analytics tutorials](#) .

9.10.1 Tutorial categories

Getting started and foundational projects

Tutorials for users new to EDB's advanced analytics capabilities or those building foundational projects.

Tutorials

**** Analytics/Lakehouse ** Level:** Beginner **Estimated time:** ~45 min **Products used:** Postgres Lakehouse, PGAA, PGFS **End result:** Stand up a Postgres Lakehouse node and query object storage from Postgres. **Related concepts:** [Data lakehouse](#) , [EDB Postgres Lakehouse](#)

**** Analytics/Lakehouse ** Level:** Beginner **Estimated time:** ~1 hr **Products used:** Postgres Lakehouse, PGFS, Delta Lake **End result:** Load raw data into object storage and query it as a Lakehouse table. **Related concepts:** [Separation of storage and compute](#)

**** Analytics/Lakehouse ** Level:** Intermediate **Estimated time:** ~1.5 hrs **Products used:** PGAA, PGFS, PGD, Postgres Lakehouse **End result:** Combine multiple data sources into a unified analytical view. **Related concepts:** [Analytics Accelerator concepts](#)

End-to-end EDB Postgres Lakehouse implementations

Projects that demonstrate full Lakehouse workflows — ingestion, storage, querying.

Tutorials

**** Analytics/Lakehouse ** Level:** Intermediate **Estimated time:** ~2 hrs **Products used:** Postgres Lakehouse, PGAA, Iceberg REST catalog, BI tools **End result:** Build an analytical pipeline and visualize Lakehouse data in BI tools. **Related concepts:** [Open table formats](#)

**** Analytics/Lakehouse ** Level:** Intermediate **Estimated time:** ~2-3 hrs **Products used:** Postgres Lakehouse, PGFS, Delta Lake **End result:** Migrate and analyze historical data sets using Delta format. **Related concepts:** [EDB Postgres Lakehouse](#)

**** Analytics/Lakehouse ** Level:** Advanced **Estimated time:** ~2-3 hrs **Products used:** Postgres Lakehouse, PGAA, BI tools (Tableau, PowerBI) **End result:** Build an interactive analytical dashboard on Lakehouse data. **Related concepts:** [Analytics Accelerator concepts](#)

Advanced PGD for analytics and tiered storage

Using PGD for tiering, offloading, and hybrid analytical architectures.

Tutorials

**** Analytics/Lakehouse ** Level:** Advanced **Estimated time:** ~3 hrs **Products used:** PGD, AutoPartition, Iceberg catalog, Postgres Lakehouse **End result:** Configure automatic tiering and offload partitions to Iceberg. **Related concepts:** [Data tiering](#)

**** Analytics/Lakehouse ** Level:** Advanced **Estimated time:** ~2-3 hrs **Products used:** PGD, PGAA, Postgres Lakehouse **End result:** Tune and test queries across tiered storage layers. **Related concepts:** [Analytics Accelerator concepts](#)

**** Analytics/Lakehouse ** Level:** Advanced **Estimated time:** ~2 hrs **Products used:** PGD, AutoPartition, Postgres Lakehouse **End result:** Implement data retention and lifecycle management policies for tiered tables. **Related concepts:** [Analytics Accelerator concepts](#)

Building AI-powered applications with Gen AI Builder and analytics

Using Gen AI Builder + Analytics Accelerator together.

For in-depth AI/ML concepts, see the [AI Factory concepts](#) .

Tutorials

**** Analytics/Lakehouse ** Level:** Intermediate **Estimated time:** ~1 hr **Products used:** Gen AI Builder, Griptape, Postgres Lakehouse **End result:** Build a Griptape structure that routes user inquiries to analytical data. **Related concepts:** [Analytics Accelerator concepts](#)

**** Analytics/Lakehouse ** Level:** Intermediate **Estimated time:** ~1 hr **Products used:** Gen AI Builder, Griptape, Postgres Lakehouse **End result:** Build a Griptape tool to query account balances from Lakehouse data. **Related concepts:** [Analytics Accelerator concepts](#)

**** Analytics/Lakehouse ** Level:** Advanced **Estimated time:** ~2 hrs **Products used:** Gen AI Builder, Griptape, Postgres Lakehouse, PGAA **End result:** Implement an AI assistant that can query Lakehouse data on demand. **Related concepts:** [Analytics Accelerator concepts](#)

Industry solution walkthroughs

End-to-end tutorials for solving real-world business problems with Analytics Accelerator.

Tutorials

**** Analytics/Lakehouse ** Level:** Advanced **Estimated time:** ~2-3 hrs **Products used:** PGD, PGAA, Lakehouse, streaming engine (Kafka, Flink) **End result:** Build a real-time fraud detection pipeline using Lakehouse + streaming. **Related concepts:** [Analytics Accelerator concepts](#)

**** Analytics/Lakehouse ** Level:** Intermediate **Estimated time:** ~2 hrs **Products used:** PGD, AutoPartition, Lakehouse, BI tools **End result:** Implement a reporting data mart architecture using tiered tables. **Related concepts:** [OLAP / OLTP](#)

This index will grow as additional tutorials are added. For Hybrid Manager (HM)-specific tutorials, see the [HM analytics tutorials](#) .