

# IBM watsonx.data

Technical Hands-on Lab



Kelly Schlamb ([kschlamb@ca.ibm.com](mailto:kschlamb@ca.ibm.com))  
Worldwide Technology Enablement

## Contents

1. Introducing watsonx.data .....	4
2. About this Lab .....	6
3. Getting Help .....	7
4. Prerequisites & Getting Started .....	8
4.1. Provision a watsonx.data Environment from TechZone .....	9
4.2. WireGuard VPN Installation .....	13
4.3. Download and Import the VPN Certificate .....	14
4.4. Web Interface URLs .....	17
4.5. Command Line Access .....	18
4.5.1. ssh (Secure Shell) .....	18
4.5.2. Remote Desktop .....	19
4.6. watsonx.data Infrastructure Components .....	21
4.7. Stopping and Starting watsonx.data .....	23
4.7.1. Stopping watsonx.data .....	23
4.7.2. Starting watsonx.data .....	24
5. Exploring the watsonx.data User Interface .....	26
5.1. Starting the watsonx.data User Interface .....	26
5.2. Infrastructure Manager Page .....	30
5.3. Data Manager Page .....	36
5.4. Query Workspace Page .....	46
5.5. Query History Page .....	54
5.6. Access Control Page .....	57
6. Working with Presto .....	65
6.1. Presto Command Line Interface (CLI) .....	65
6.2. Presto Web Interface .....	71
7. Working with MinIO .....	75
7.1. Introduction to Object Storage .....	75
7.2. Exploring MinIO Object Storage .....	76
8. Data Ingest .....	80
9. Federated Queries .....	86

10. Offloading Data from a Data Warehouse .....	96
11. Time Travel and Rollback.....	100
12. Summary.....	105
Appendix A. Troubleshooting.....	106
watsonx.data Console:.....	106
MinIO Console:.....	107
Presto Console: .....	107
Miscellaneous:.....	107
Appendix B. Acknowledgements .....	108

## 1. Introducing watsonx.data

Watsonx.data is a core component of **watsonx**, IBM's enterprise-ready AI and data platform designed to multiply the impact of AI across an enterprise's business.

The watsonx platform comprises three powerful components: the **watsonx.ai** studio for new foundation models, generative AI and machine learning; the **watsonx.data** fit-for-purpose data store that provides the flexibility of a data lake with the performance of a data warehouse; plus the **watsonx.governance** toolkit, to enable AI workflows that are built with responsibility, transparency, and explainability.

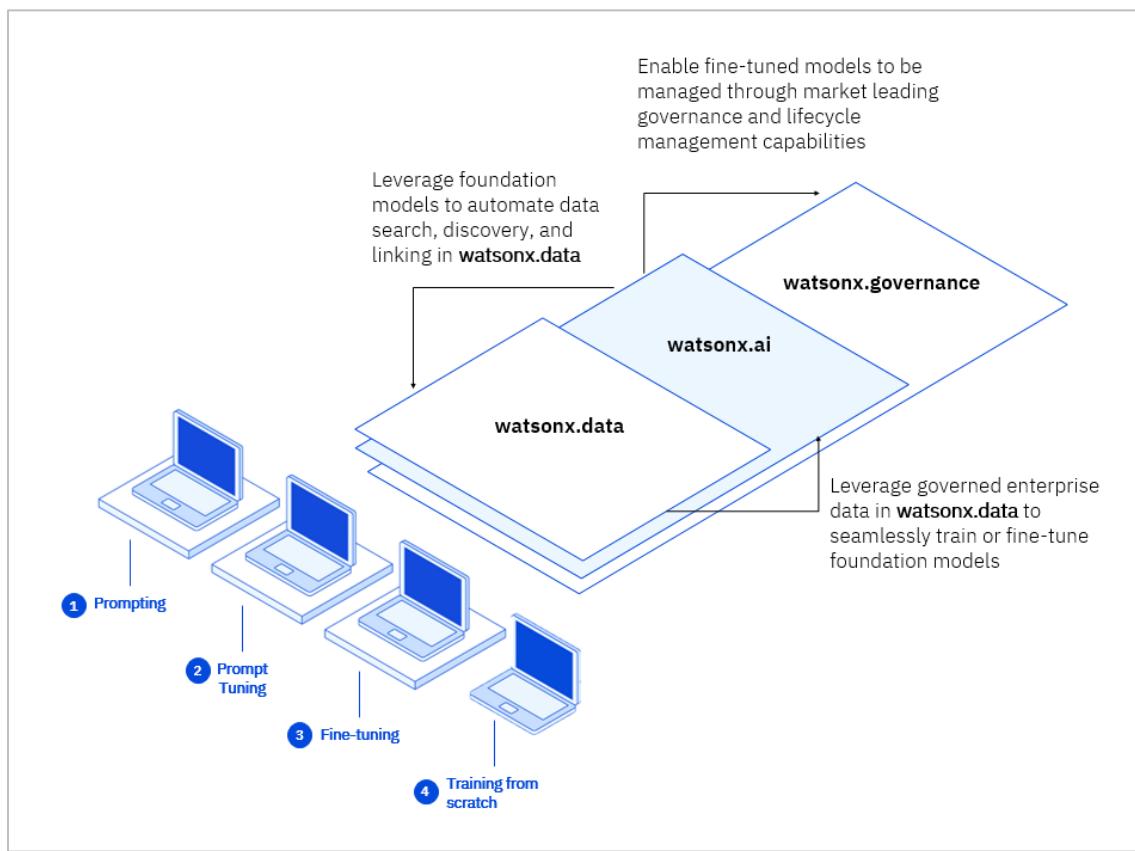


Figure 1: IBM watsonx platform

The watsonx.data component (the focus of this lab) makes it possible for enterprises to scale analytics and AI with a data store built on an open lakehouse architecture, supported by querying, governance, and open data and table formats, to access and share data. With watsonx.data, enterprises can connect to data in minutes, quickly get trusted insights, and reduce their data warehouse costs.

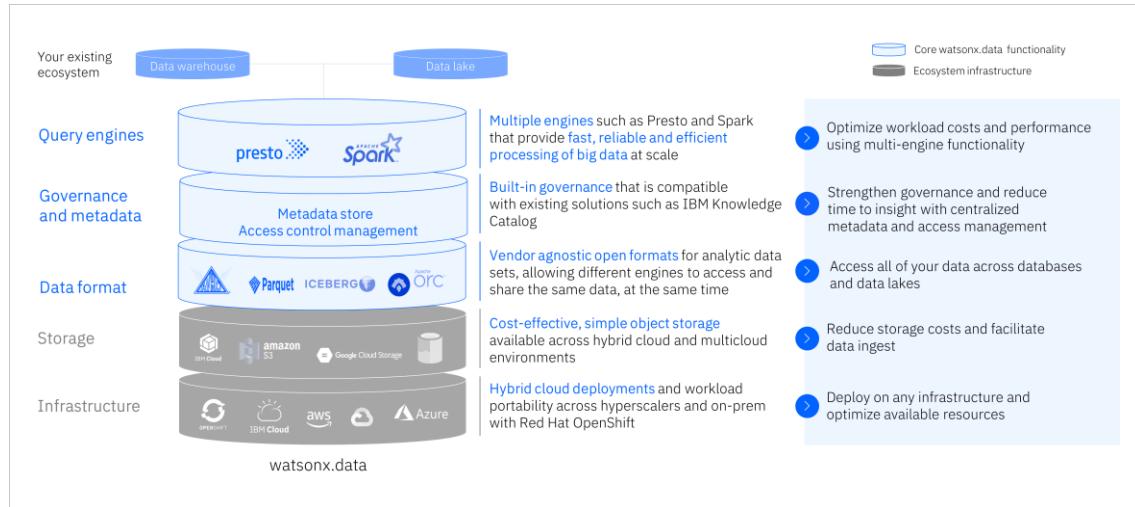


Figure 2: IBM watsonx.data's lakehouse architecture and value proposition

## 2. About this Lab

This IBM watsonx.data hands-on lab introduces you to some of the core components and capabilities of IBM watsonx.data. By completing this lab, you will gain the experience needed to demo these capabilities to clients.

Specifically, you will get hands-on experience in the following areas:

- The watsonx.data web-based user interface (UI), including infrastructure management, data management, running SQL statements, and managing user access
- The Presto web interface and the Presto command line interface (CLI)
- MinIO object storage
- Ingesting data into watsonx.data
- Creating schemas and tables
- Running queries that combine data from multiple data sources (data federation)
- Offloading tables from Db2 into watsonx.data
- Rolling back a table to a previous point in time

This lab requires that you provision an environment from IBM Technology Zone (TechZone). The image used comes pre-configured with watsonx.data Developer Edition, additional database systems including Db2 and PostgreSQL, and sample data sets. Provisioning details are provided in the [Prerequisites & Getting Started](#) section of this lab guide.

Some of the lab sections require that you copy commands or SQL statements and paste them into your browser or a command line terminal. If you have problems copying and pasting from this document, all of the commands and SQL statements can be found in this [text document](#). Download this file and copy the text from there instead.

Watsonx.data is being developed and released in an agile manner. In addition to new capabilities being added, the web interface is also likely to change over time. Therefore, the screenshots used in this lab may not always look exactly like what you see.

### 3. Getting Help

**Lab guide help:** If you require assistance in interpreting any of the steps in this lab, please post your questions to the [#data-ai-demo-feedback](#) Slack channel (IBMer only). Business Partners can request help at the [Partner Plus Support](#) website.

**Techzone environment:** If you are encountering issues regarding the Techzone environment being used in this lab, please see the [Techzone Help](#) page.

**watsonx.data:** Assistance with the watsonx.data product itself is available in the [#watsonx-data-techbites](#) Slack channel (IBMer only). Additionally, please refer to the watsonx.data documentation as needed ([SaaS](#), [software](#)).

**Additional troubleshooting:** See [Appendix A. Troubleshooting](#) at the end of this lab guide for guidance on commonly encountered issues.

## 4. Prerequisites & Getting Started

Clients can deploy watsonx.data in a number of different ways:

- As software-as-a-service (SaaS) on IBM Cloud and Amazon Web Services (AWS)
- As a cartridge with Cloud Pak for Data
- As standalone hybrid-cloud software that can be installed on Red Hat OpenShift (on-premises or in the cloud)
- As a simple, single-node Developer Edition installation

Watsonx.data is currently available in a *Standard Edition*, with an *Enterprise Edition* planned for the future. For the developer and partner community, IBM also offers an entry-level *Developer Edition*, which can be used to get familiar with the watsonx.data console and environment. The Developer Edition has the same code base as the Standard Edition, but some features are restricted, and it is not intended for production use.

This lab utilizes a pre-installed Developer Edition virtual machine (VM) environment that can be easily provisioned from IBM Technology Zone (TechZone). Provisioning instructions are included in the section that follows this one. WireGuard VPN is required to access the environment after it is provisioned. Instructions for configuring WireGuard are also included in this lab.

A number of software components are included as part of the VM. In this lab, you will be primarily interfacing with the following components:

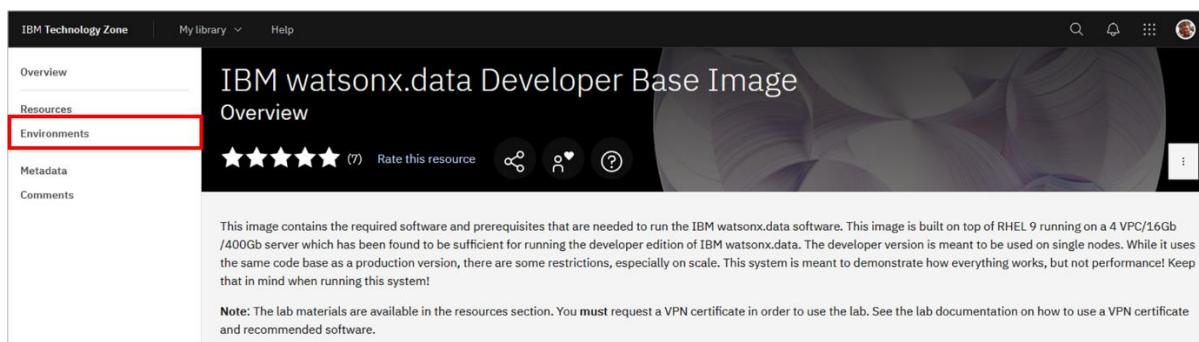
- watsonx.data (which includes a graphical console)
- Presto (a SQL query engine used in watsonx.data; it includes its own graphical console)
- MinIO (local object storage; it includes its own graphical console)

The Developer Edition comes pre-configured with a number of watsonx.data *infrastructure components*, including a Presto query engine, two catalogs, and two object storage buckets.

## 4.1. Provision a watsonx.data Environment from TechZone

Follow the steps below to provision the watsonx.data VM lab environment from IBM Technology Zone (TechZone). This environment is available to IBM employees and IBM Business Partners, but not clients.

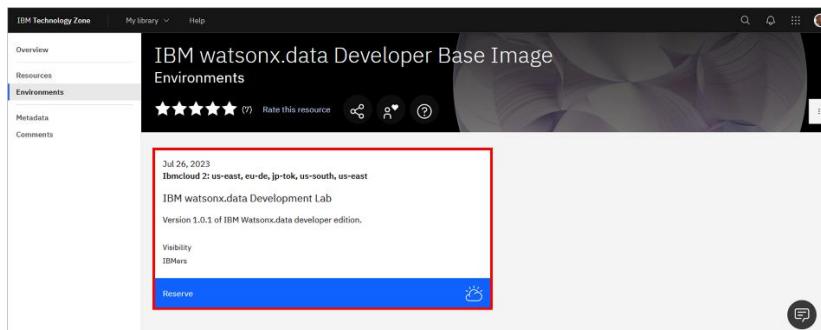
1. Open the **IBM watsonx.data Developer Base Image** collection in **IBM Technology Zone** at: <https://techzone.ibm.com/collection/ibm-watsonxdata-developer-base-image>. Sign in with your IBMid and accept any terms and conditions you are presented with.
2. Select the **Environments** tab in the left-side menu.



The screenshot shows the 'IBM watsonx.data Developer Base Image Overview' page in the IBM Technology Zone. The left sidebar has tabs for 'Overview', 'Resources', 'Environments' (which is highlighted with a red box), 'Metadata', and 'Comments'. The main content area displays the title 'IBM watsonx.data Developer Base Image Overview' and a five-star rating with '(7) Rate this resource'. Below the title is a descriptive text block: 'This image contains the required software and prerequisites that are needed to run the IBM watsonx.data software. This image is built on top of RHEL 9 running on a 4 VPC/16Gb /400Gb server which has been found to be sufficient for running the developer edition of IBM watsonx.data. The developer version is meant to be used on single nodes. While it uses the same code base as a production version, there are some restrictions, especially on scale. This system is meant to demonstrate how everything works, but not performance! Keep that in mind when running this system!' At the bottom of this block is a note: 'Note: The lab materials are available in the resources section. You must request a VPN certificate in order to use the lab. See the lab documentation on how to use a VPN certificate and recommended software.'

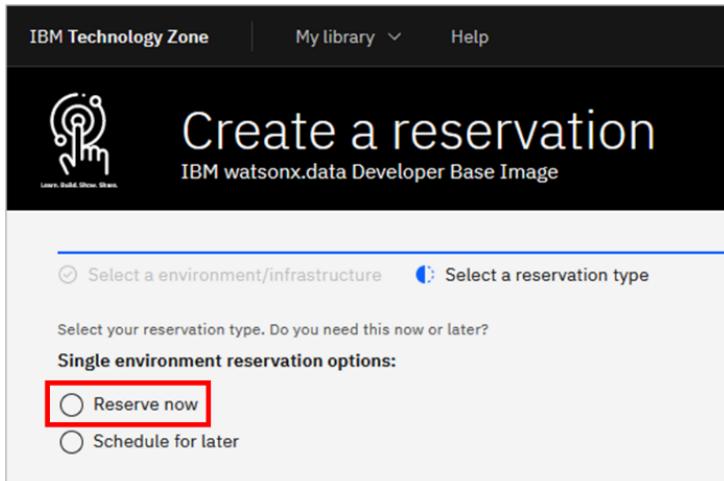
**Note:** This Techzone Collection includes its own lab guide (in the **Resources** tab) section. To be clear, that lab is *not* the same as this lab that you are currently going through. There are some common elements between the two labs, so consider taking it to reinforce what you are learning in this one, but that other lab is *not* a part of the watsonx.data for Technical Sales Level 3 learning plan and does not need to be done to complete this learning plan.

3. Click the **IBM watsonx.data Development Lab** tile.



The screenshot shows the 'IBM watsonx.data Developer Base Image Environments' page. The left sidebar has tabs for 'Overview', 'Resources', 'Environments' (selected), 'Metadata', and 'Comments'. The main content area features a large blue button labeled 'IBM watsonx.data Development Lab'. Above the button, there is descriptive text: 'Jul 26, 2023 Ibmcloud 2: us-east, eu-de, jp-tok, us-south, us-east IBM watsonx.data Development Lab Version 1.0.1 of IBM Watsonx.data developer edition.' Below the button, there is a 'Visibility' section with 'IBMer' selected and a 'Reserve' button at the bottom.

4. For the reservation type, select the **Reserve now** radio button.



5. Accept the default for the reservation **Name**, or provide a name of your choosing. For the **Purpose** of the reservation, select **Practice / Self-Education**.

The screenshot shows the 'Fill out your reservation' step of the process. It has four tabs at the top: 'Select a environment/infrastructure', 'Select a reservation type' (selected), 'Fill out your reservation' (current tab), and 'Complete'. In the 'Name' section, the input 'IBM watsonx.data Development Lab' is shown and highlighted with a red box. Below it, a note says 'Name this reservation. This will help identify it in your reservation list.' In the 'Purpose' section, there are four options: 'Customer Demo' (description: Showcasing a pre-built, standardized product use case with minimal customization needed on the environment for a client demonstration. Aligns with 'Custom Demo' in IBM Sales Cloud.), 'Practice / Self-Education' (selected, highlighted with a red box; description: Gaining experience with specific technology, product, or solution.), 'Proof-of-Technology' (description: Customizing an environment to set up and personalize beyond the pre-built option to showcase a specific customer use case. Typically these environments are needed for a longer period of time. Aligns with 'Proof of Value' in IBM Sales Cloud.), and 'Test' (description: Need to test a specific function, configuration, or customization.). A note at the bottom states: 'Please ensure to select the correct purpose as this can NOT be updated or changed after this reservation has been created. Review the [Reservation Duration Policy](#) to understand default durations allowed for specific infrastructures based on purpose.'

- Fill in the **Purpose description** box (you may have to scroll down to see it) with the reason you are making the reservation. Then, scroll further down and select your **Preferred geography** based on your location.

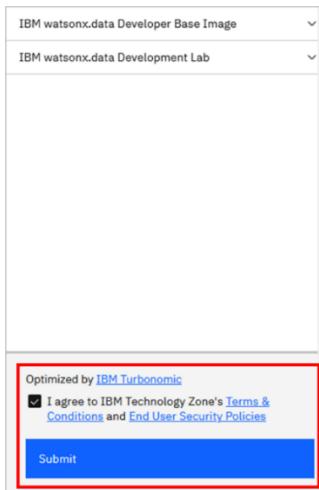
The screenshot shows two sections of a reservation form. The top section is labeled 'Purpose description' and contains a text input field with the placeholder 'Self-education on watsonx.data to earn the watsonx.data Technical Sales Intermediate badge.' Below this is a note asking 'What are you doing? Why do you need this? What are you trying to accomplish?'. The bottom section is labeled 'Preferred Geography' and contains a dropdown menu showing 'AMERICAS - us-east region - wdc06 datacenter'.

- Scroll down further. Adjust the reservation's **End date and time** as needed (by default it's two days (48 hours) from now; it can't exceed two days initially, but you can extend the reservation by two days, up to two times, before it expires). A VPN is needed to access the server, so change the **VPN Access** dropdown to **Enable**.

**Note:** If you do not enable VPN access then you will not be able to use the environment once provisioned. Ensure you have it enabled before continuing.

The screenshot shows two sections of a reservation form. The top section is labeled 'End date and time' and includes fields for 'Select a date' (set to 07/29/2023), 'Select a time' (set to 10:00 AM), and a dropdown for 'America/Toronto'. Below these fields is a note about reservation policy: 'Reservation policy: Recommended 2 days, but can be reserved up to 2 days on this reservation form. Extend later for 2 days increments up to 4 days total. Max time 6 days total.' The bottom section is labeled 'VPN Access' and contains a dropdown menu showing 'Enable'.

8. On the right-side panel, follow the links to read the **Terms & Conditions** and the **End User Security Policies** documents. Then, select the checkbox to agree to those terms. Finally, click **Submit**.



A message in the upper-right corner will briefly appear stating that the reservation has been created. You may also be presented with an opportunity to provide feedback on the process. Feel free to share your feedback.

Shortly after, you will receive an email from **IBM Technology Zone** acknowledging receipt of the request and you will receive another email when the provisioning is complete. Provisioning may be as quick as 15 minutes, or it may take an hour or more. *If provisioning fails, it may be due to a lack of resources in the geography specified. Try again with the same geography or specify a different one.*

Reservation status is available at <https://techzone.ibm.com/my/reservations>.

## 4.2. WireGuard VPN Installation

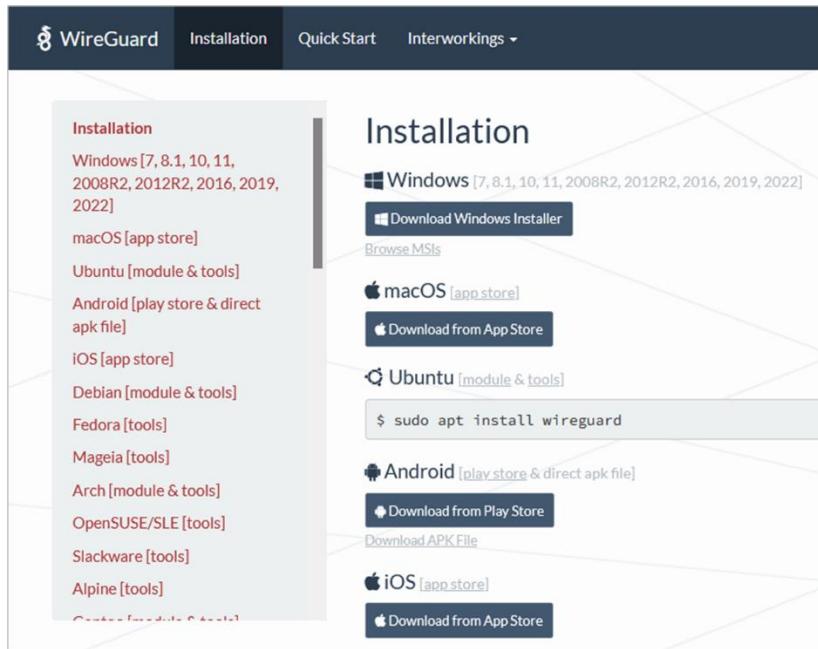
WireGuard VPN is needed to be able to access the watsonx.data environment you are provisioning in TechZone. The WireGuard VPN tunnel uses state-of-the-art cryptography to provide access to your VM through an isolated network.

To be clear, this is *not* the Cisco AnyConnect VPN client that IBMers have installed on their computers. This is a separate application that must be installed for the purpose of secured access to TechZone-provisioned VMs.

In this section you will install the WireGuard VPN client. When the provisioning of your TechZone environment is complete, you will be provided with a VPN certificate that can be imported into the WireGuard VPN client.

This installation section can be skipped if you already have the WireGuard VPN client installed.

1. Go to the WireGuard installation page at: <https://www.wireguard.com/install>.
2. Download and install the relevant software package for your computer.



The WireGuard VPN client may automatically start after installation. You can keep it open in the background or close it.

### 4.3. Download and Import the VPN Certificate

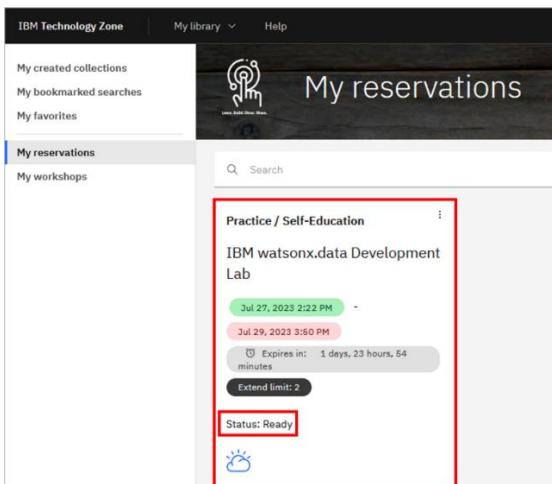
**Note:** You must now wait until the environment has been provisioned to complete this section.

Continue with the steps below once you've received the **Reservation Ready on IBM Technology Zone** email from IBM Technology Zone.

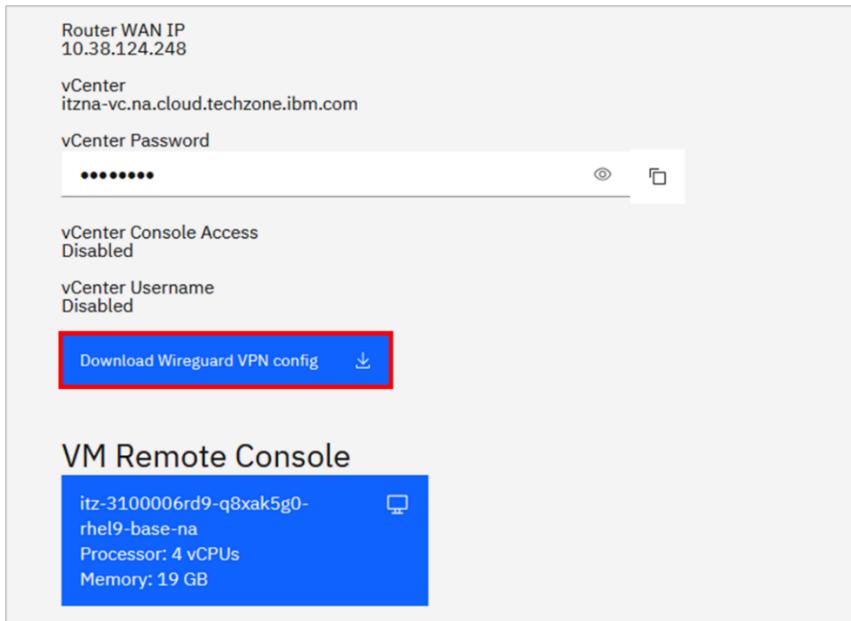
1. Open the **Reservation Ready on IBM Technology Zone** email.
2. Click the **View My Reservations** button to view your TechZone reservations (you may have to login again).



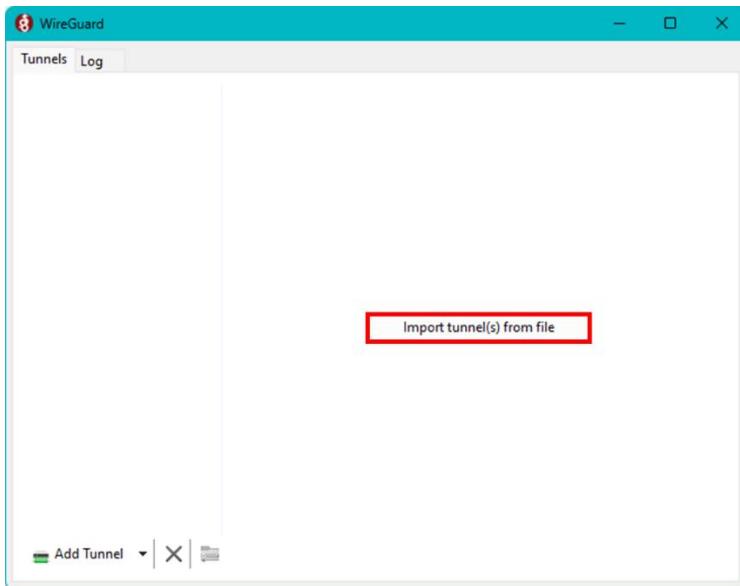
3. The tile associated with your reservation opens and the tile should say **Status: Ready**. Click this tile.



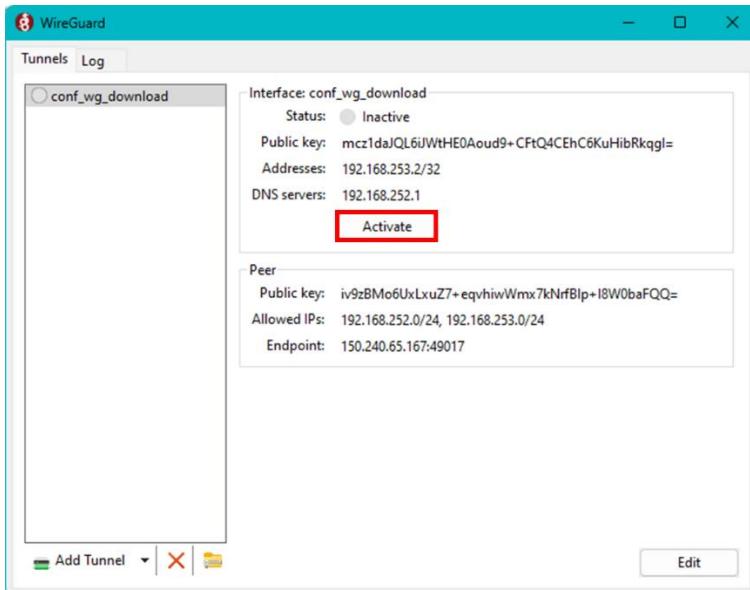
4. Scroll to the bottom of the page. Click the **Download Wireguard VPN config** button to download the VPN certificate file (**conf\_wg\_download.conf**) to your computer. Make note of the download folder location.



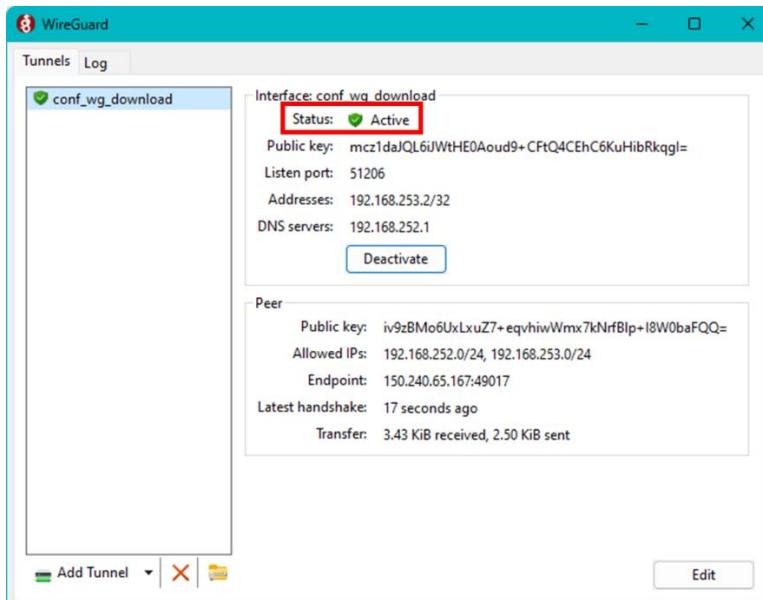
5. Start the WireGuard VPN client on your computer (if you don't already have it open).
6. Click the **Import tunnel(s) from file** button. (Windows-based screenshots are shown; the process is very similar for macOS.)



7. Navigate to the location where you downloaded the `conf_wg_download.conf` certificate file and select it for import.
8. With the certificate imported and a tunnel created, click the **Activate** button.



The **Status** of the tunnel should change from **Inactive** to **Active** as in the image below.



**Note:** Your interactions with the environment will always use the **192.168.253.2** address. Copy and paste this IP address somewhere such that you can reference it easily throughout this lab.

## 4.4. Web Interface URLs

In this lab you will interact with the graphical web interfaces (consoles) for watsonx.data, Presto, and the MinIO Object Store. The URLs, usernames, and passwords for these consoles will be provided in context of the instructions throughout the lab guide. However, they are listed here as well for reference purposes:

- **watsonx.data:** <https://192.168.252.2:9443>
  - Username: ibmlhadmin
  - Password: password
- **Presto:** <http://192.168.252.2:8080/>
  - No credentials needed
- **MinIO:** <http://192.168.252.2:9001/>
  - Access Key and Secret Key are unique to your environment. Instructions to extract these keys will be provided later.

## 4.5. Command Line Access

Some of the activities performed in this lab require that you run commands within the VM. There are two ways that this can be done. The preferred approach is to use ssh (Secure Shell) from your computer into the VM. The second approach is to bring up the VM's graphical desktop on your computer and open a terminal command window from within that VM desktop.

**Note:** Instructions in this lab assume you are using ssh, but again, you have the choice of using a terminal window within the remote desktop as well.

### 4.5.1. ssh (Secure Shell)

ssh is a secured network protocol that allows command-line execution against remote servers.

It is likely the case that ssh is already installed on your computer (Windows, macOS, and Linux). If not, you will need to download an ssh client for your specific operating system (such as PuTTY on Windows or Hyper on Mac).

To ssh into your watsonx.data VM, run the following command from your operating system's command prompt/terminal (the command takes the form of: `ssh <username>@<hostname-or-ipaddress>`). When prompted for the user's password, enter **watsonx.data**.

```
ssh watsonx@192.168.252.2
```

The first time that you run the command you may receive a message stating that the authenticity of the host can't be established. In this case, respond stating that you want to proceed anyway.

You are logging in with the **watsonx** username. When you need to run commands as the **root** user (you will be instructed when that is the case), enter the following command to switch to the **root** user. If prompted for the password of the watsonx user, enter **watsonx.data**.

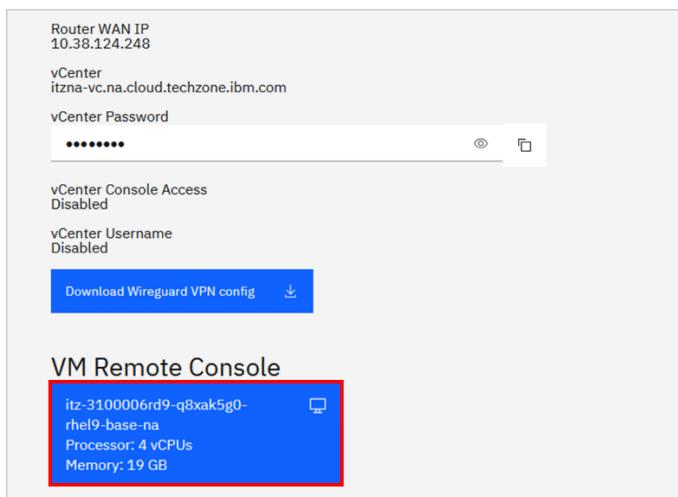
```
sudo su -
```

#### 4.5.2. Remote Desktop

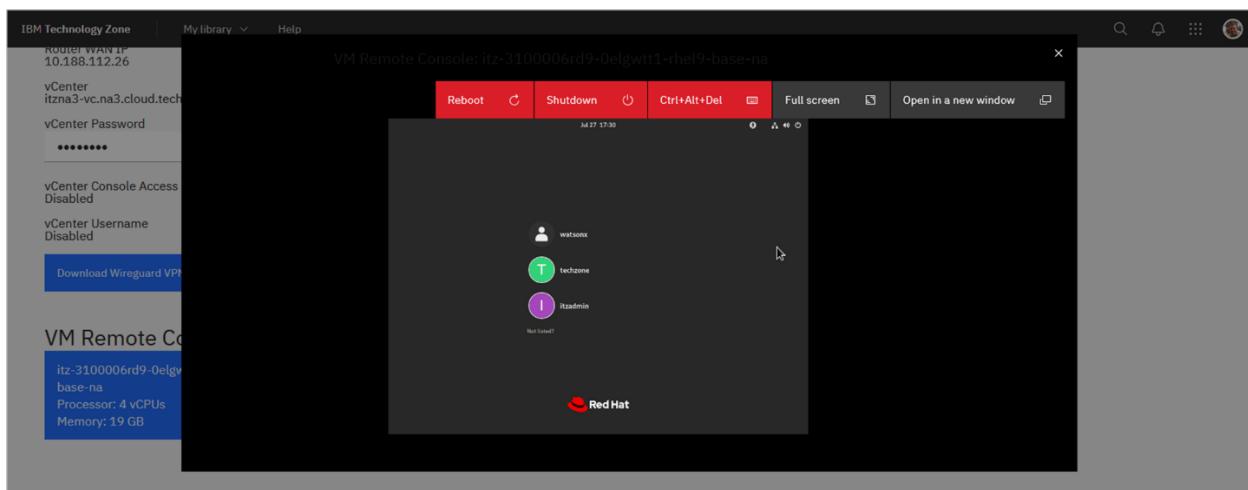
There are actually two approaches to remotely accessing the VM's desktop. One way is to use a locally installed VNC (Virtual Network Computing) client on your computer, such as TigerVNC or UltraVNC. The instructions for doing this are beyond the scope of this lab.

The other way is to use TechZone's **VM Remote Console** capability.

To use this, go to your TechZone reservations page, scroll to the bottom of the page, and click the **VM Remote Console** button.



The VM's desktop is shown. You may need to move your mouse pointer into the desktop area to have the login screen appear.



To login as the **watsonx** user, click on the **watsonx** username and enter **watsonx.data** for the password. Note that the remote console interface can be expanded to the full screen, or you can open it in its own browser window.

You may find the remote console difficult to use (it can be slow, and you may not see the mouse pointer at times) and that is why ssh is the recommended approach to running commands remotely.

## 4.6. watsonx.data Infrastructure Components

There are four infrastructure components that can be configured in watsonx.data:

- **Engines:** A query engine is used to run workloads against data in watsonx.data. Watsonx.data supports multiple engines; this lab will use Presto as the query engine.
- **Catalogs:** Metadata catalogs are used to manage table schemas and metadata for the data residing in watsonx.data.
- **Buckets:** Watsonx.data stores data in object storage. Specifically, data is stored in buckets, which are identified storage areas within object storage, similar to file folders. AWS Simple Storage Service (S3), IBM Cloud Object Storage (COS), and MinIO object storage are supported.
- **Databases:** External databases can be registered and used in watsonx.data. Schemas and metadata for objects within a database are stored in a watsonx.data catalog.

**Note:** The term *schema* was used above – and this term will be used often in this lab. It has multiple meanings; in this context, a schema refers to the definition of a table, including the name and data type for each column. Metadata for each table is maintained in watsonx.data (specifically in the catalogs within the metadata store), including the table's schema and where its data files can be found in object storage, among other things.

This lab uses watsonx.data *Developer Edition*, which includes an embedded instance of *MinIO*, an open-source S3-compatible object store. However, you can also register and use external buckets that you have created elsewhere (such as in IBM Cloud or Amazon Web Services).

Developer Edition comes pre-configured with the following components:

- **presto-01:** This is a Presto query engine. It is used to interact with data in the lakehouse.
- **iceberg\_data:** This is an Iceberg catalog, residing within watsonx.data's embedded Hive Metastore (HMS). It manages tables that have been created with the Iceberg open table format. This catalog is associated with the iceberg-bucket object storage bucket.
- **hive\_data:** This is a Hive catalog, also residing within the embedded HMS. This catalog is intended for use with non-Iceberg tables, where data is stored in files (such as Parquet, ORC, or CSV), but they are not using the Iceberg table format. This catalog is associated with the hive-bucket object storage bucket.

- **iceberg-bucket:** This is a bucket in the embedded MinIO object store. The table data stored here is associated with the iceberg\_data catalog.
- **hive-bucket:** This is a bucket in the embedded MinIO object store. The table data stored here is associated with the hive\_data catalog.

In this demo environment, you are limited in what you can add, edit, or delete. Because Developer Edition is being used, some tasks are prohibited. For example, you can't delete any of the pre-configured components and you can't add new query engines (Presto or other types).

When using a non-Developer Edition of watsonx.data, you are required to create/add these different components before you can proceed. This applies to both SaaS (the managed service) and user-managed deployments of watsonx.data.

You are likely asking yourself why the environment is set up with two different catalogs (Hive and Iceberg). Generally speaking, tables that you create and populate within watsonx.data should use the Iceberg open table format because they inherit beneficial capabilities like transactional consistency, schema and partition evolution, snapshots for time travel queries and rollback, and improved query efficiency. Iceberg was built to handle huge datasets and analytics workloads.

However, you may need to land existing data into watsonx.data's object storage that is in supported data file formats (like Parquet or ORC) but aren't using the Iceberg table format. In this case, you can land your data into a Hive catalog-managed bucket and then create a non-Iceberg format table on top of it. You can then create a new Iceberg version of the table (managed by an Iceberg catalog) with the same table definition and populate it with the data from the original table. This is made easy by a common SQL statement called *CREATE TABLE AS SELECT* (or CTAS for short), which is supported by Presto. You will see this in action later in this lab. For large-scale conversion jobs, Spark can also be used.

## 4.7. Stopping and Starting watsonx.data

This lab environment is configured so that watsonx.data is started automatically. You should not need to perform any manual steps yourself to use watsonx.data.

However, if you find that you need to stop and restart the watsonx.data software, for whatever reason, you can follow the instructions below to do so. This is purely for reference, so *DO NOT DO THIS NOW.*

### 4.7.1. Stopping watsonx.data

1. Open an SSH session and connect to the watsonx.data VM with the **watsonx** user ID. If you receive a message stating that the authenticity of the host can't be established, respond stating that you want to proceed anyway.

```
ssh watsonx@192.168.252.2
```

2. When prompted for the password, enter **watsonx.data**.
3. Switch to the **root** user (you may be prompted to enter the password for the **watsonx** user; if so then enter **watsonx.data**).

```
sudo su -
```

4. Change the directory to the watsonx.data product binaries (or alternatively, prefix the stop and/or start commands shown later with this path).

```
cd /root/ibm-lh-dev/bin
```

5. Stop watsonx.data.

```
./stop
```

You will see output similar to the text below (this is an excerpt and not the full output).

```
Inspecting docker image lhconsole-ui before removal:  
lhconsole-ui still running. Removing lhconsole-ui...  
lhconsole-ui  
  
Inspecting docker image lhconsole-nodeclient before removal:  
lhconsole-nodeclient still running. Removing lhconsole-  
nodeclient...  
lhconsole-nodeclient-svc  
  
Inspecting docker image lhconsole-javaapi before removal:  
lhconsole-javaapi still running. Removing lhconsole-javaapi...  
lhconsole-javaapi-svc  
...
```

6. Check the status of watsonx.data. If all of the watsonx.data components have been stopped then you will see no output to this command.

```
./status --all
```

#### 4.7.2. Starting watsonx.data

If you don't currently have a terminal open as the root user, follow Steps 1-4 from above to do so.

1. Start watsonx.data by running the following two commands.

```
export LH_RUN_MODE=diag
```

```
./start
```

It will take a few minutes for the various component containers to start. Output from the start command should look similar to what's shown below (this is an excerpt and not the full output):

```
-- starting container for ibm-lh-minio...
96ad814705c43487ea02044d8b4e5e682901caa39734258c929510b53a74f15e
-- starting container for ibm-lh-postgres...
6f20a833ae469089f63cae7c34419161e1cf0170bd24f354c4a4823e7356f981
Checking container: ibm-lh-postgres
Container ID for ibm-lh-postgres is: 6f20a833ae46
Postgres status is:
localhost:5432 - no response localhost:5432 - accepting
connections

-- starting container for ibm-lh-control-plane-prereq...
...
```

## 2. Check the status of watsonx.data:

```
./status --all
```

If watsonx.data has started successfully then you will see a number of containers running.

ibm-lh-hive-metastore	running
ibm-lh-minio	running
ibm-lh-postgres	running
ibm-lh-presto	running
lhconsole-api	running
lhconsole-javaapi-svc	running
lhconsole-nodeclient-svc	running
lhconsole-ui	running

**Note:** Even though the containers are running, the software components within those containers might still be initializing. For example, you may need to wait a few more minutes before you'll be able to bring up the watsonx.data graphical interface in a web browser.

## 5. Exploring the watsonx.data User Interface

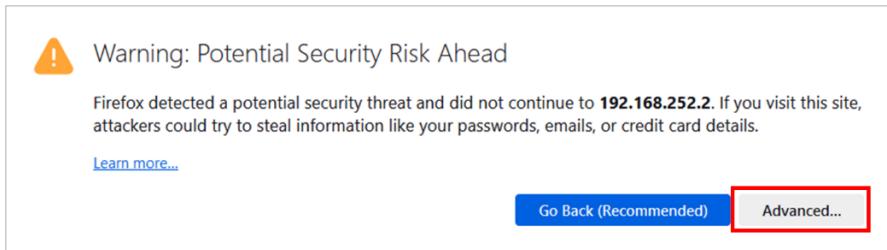
### 5.1. Starting the watsonx.data User Interface

Administration of the watsonx.data environment is primarily done with the watsonx.data user interface (also known as a console).

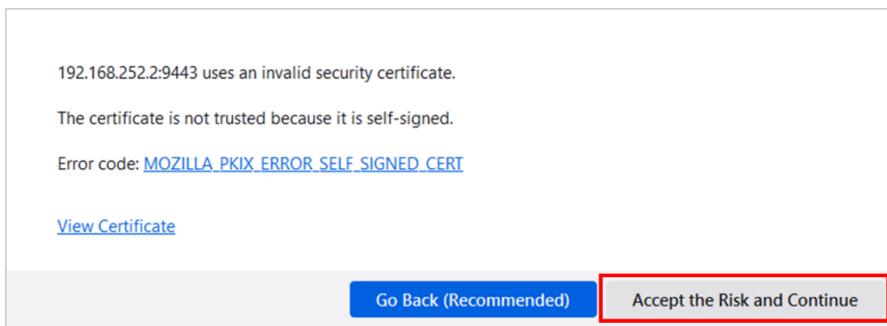
1. From your computer, open the watsonx.data console in your browser (<https://192.168.252.2:9443>).

For users of the Firefox browser (steps 2 & 3):

2. You might receive a warning about a potential security risk. If so, click the **Advanced** button.

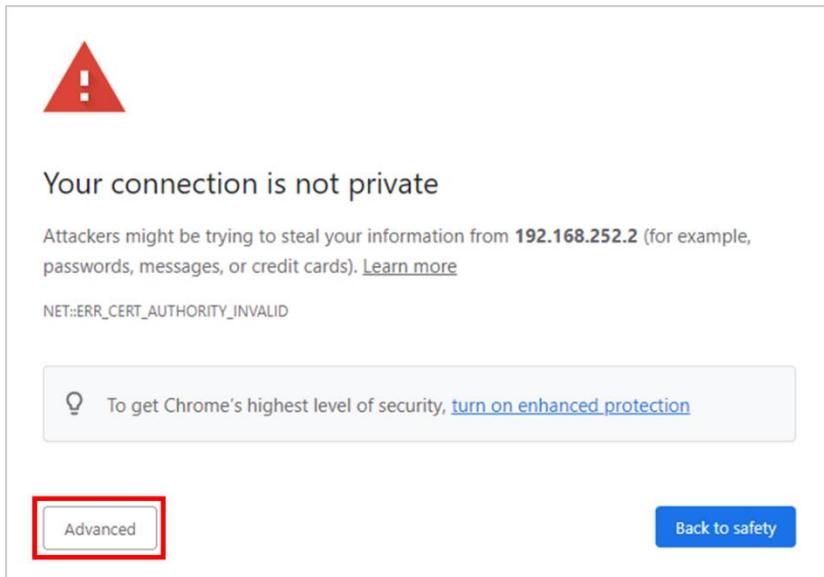


3. Scroll down the page and click the **Accept the Risk and Continue** button. Note that after doing this you will no longer see the warning when you open the console in the future.

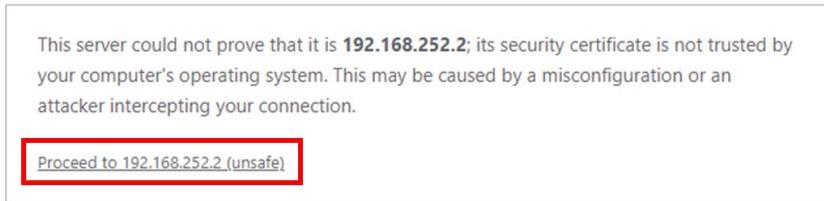


For users of the Chrome browser (steps 4 & 5):

4. You might receive a warning about the connection not being private. If so, click the **Advanced** button.



5. Scroll down the page and click the **Proceed to <URL> (unsafe)** link. Note that after doing this you will no longer see the warning when you open the console in the future.



All users should continue with the following steps:

6. In the IBM watsonx.data login screen, enter the following credentials:

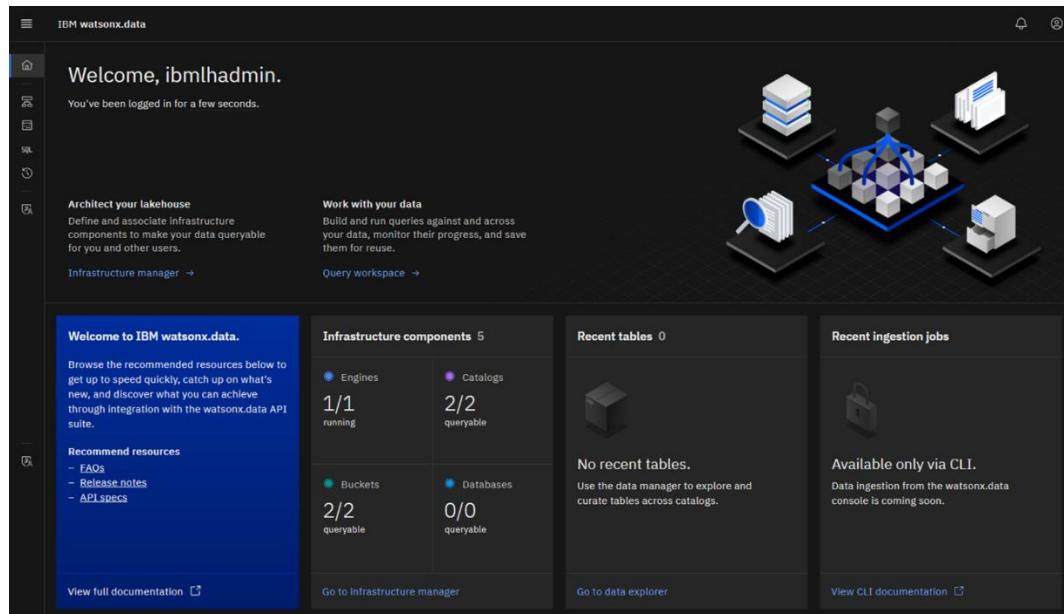
**Username:** ibmlhadmin

**Password:** password

7. Click the **Log in** button.



8. You will immediately start in the **Home** screen for watsonx.data. Scroll down and explore the contents of the page.



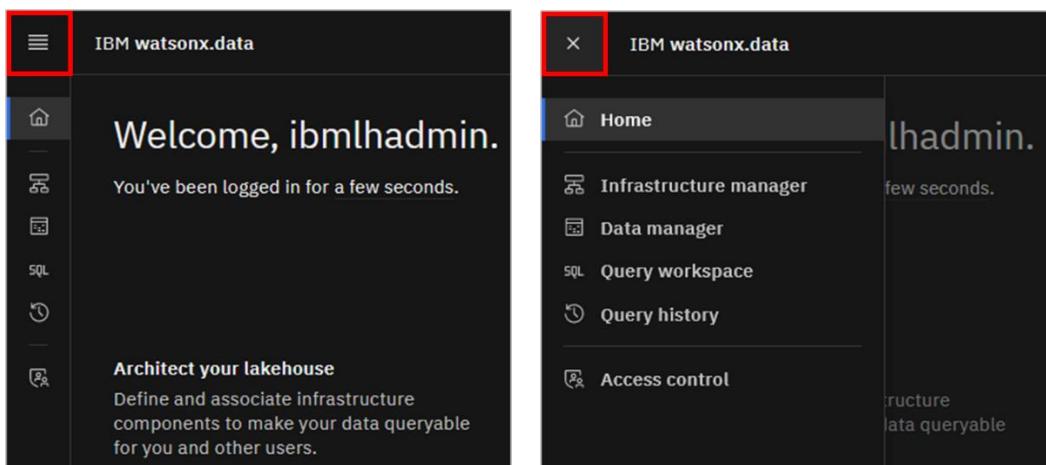
Included on the page are the following panels:

- **Welcome:** Introductory information, including a link to documentation
- **Infrastructure components:** A summary of the engines, catalogs, buckets, and databases that are registered with watsonx.data
- **Recent tables:** Tables that have recently been explored
- **Recent ingestion jobs:** Jobs that have recently moved data into watsonx.data
- **Saved worksheets:** Frequently run queries saved as worksheets, for easier reuse
- **Recent queries:** Queries that have recently been run or are in the process of being run

9. Note the left-side menu. Hover your mouse pointer over the various icons to see what actions or console pages they refer to.

-  **Home:** Returns you to the main Home screen
-  **Infrastructure manager:** Define and associate infrastructure components
-  **Data manager:** Browse schemas and tables by engine
-  **Query workspace:** Build and run queries against the data stored in the environment
-  **Query history:** Audit current and past queries across engines
-  **Access control:** Manage who can access infrastructure components and data

10. Alternatively, click on the **hamburger** icon in the upper left to expand the left-side menu such that you can see the name beside each icon. To collapse the menu back to the default view, click the **X** in the upper left.

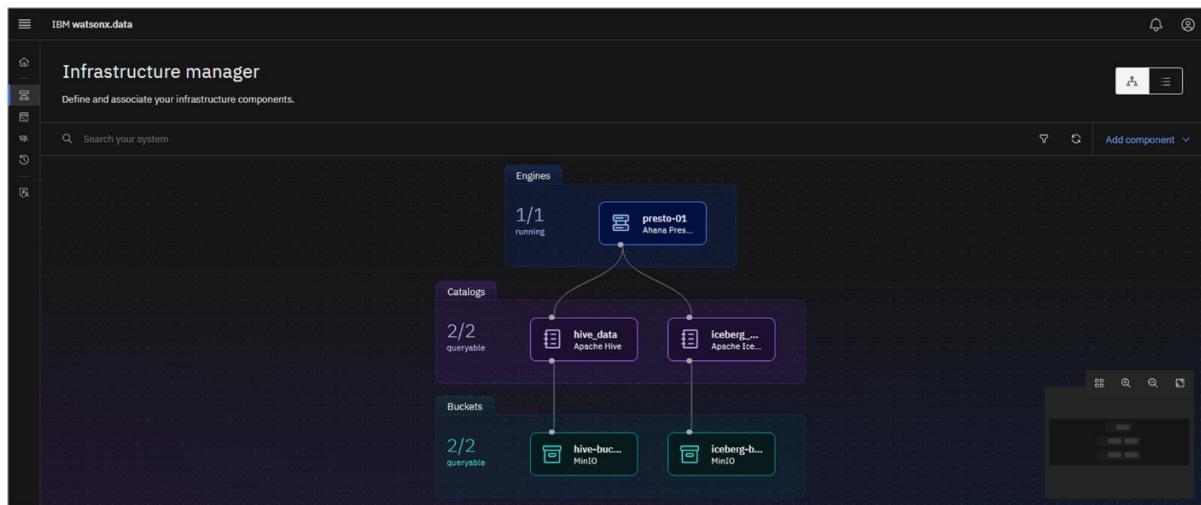


## 5.2. Infrastructure Manager Page

1. Select the **Infrastructure manager** icon (grid) from the left-side menu.

The **Infrastructure manager** page opens with a graphical canvas view of the different infrastructure components currently defined in this watsonx.data environment: Engines (blue layer), Catalogs (purple layer), Buckets (green layer), and Databases (also blue, but not shown).

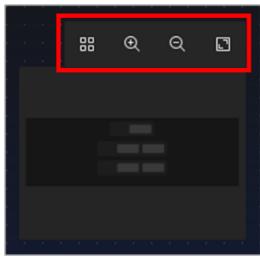
**Note:** Watsonx.data Developer Edition comes pre-configured with a Presto query engine, two catalogs, and two object storage buckets.



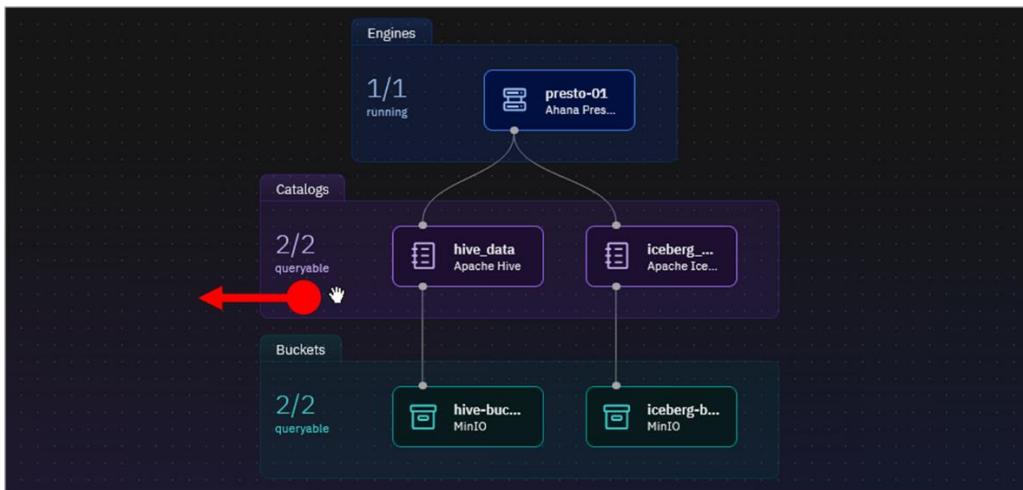
Each bucket is associated with a catalog (with a 1:1 mapping). When a bucket is added to watsonx.data, a catalog is created for it at the same time, based on input from the user. Likewise, if a database connection is added (for federation purposes), a catalog is created for that database connection as well. Both of these activities will be shown later in the lab.

Each catalog is then associated with one or more engines. An engine can't access data in a bucket or a remote database unless the corresponding catalog is associated with the engine.

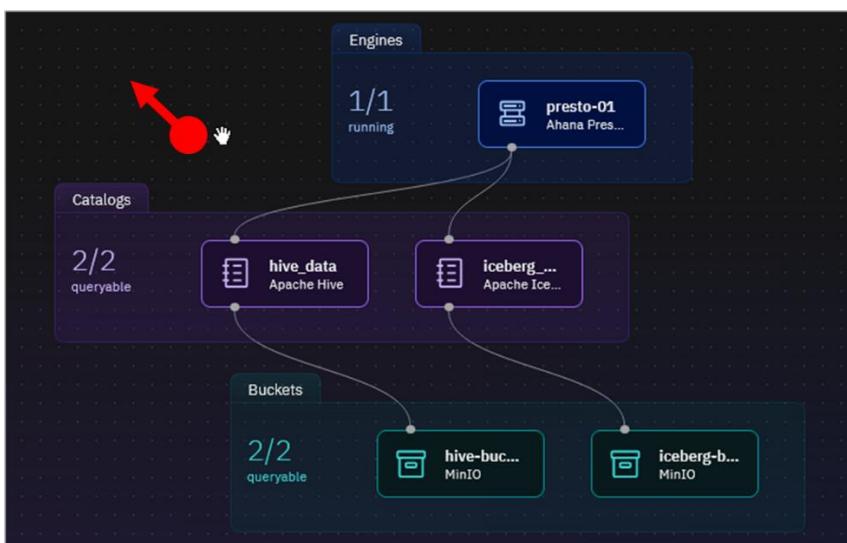
2. Note the mini-map in the lower left corner. This topology view is rather simple at this point, but as the number of infrastructure components grows, this control widget gives you a handy way to zoom in and out, auto-arrange the components, or fit the topology diagram to the screen. Try each of the icons to see how they affect the view.



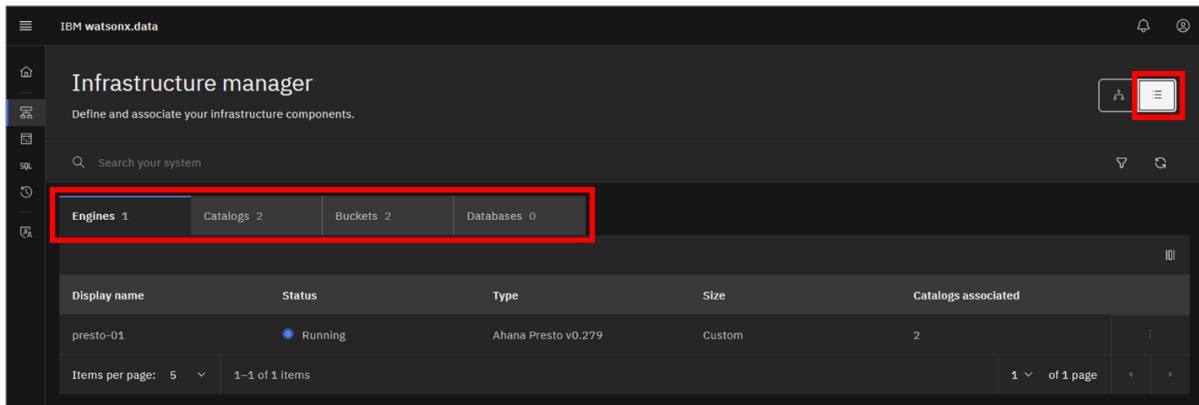
3. Additionally, you can drag and drop the different infrastructure layers across the canvas. Click within the purple **Catalogs** area, hold the mouse button down, and move the catalogs to a different spot on the canvas.



4. Finally, you can pan across the canvas as a whole. Click somewhere in the black background of the canvas, hold the mouse button down, and move your mouse to drag the canvas.



5. In addition to the graphical topology view, infrastructure components can be listed in a table format. Click the **List view** icon in the upper-right to switch to this alternate view.



IBM watsonx.data

Infrastructure manager

Define and associate your infrastructure components.

Search your system

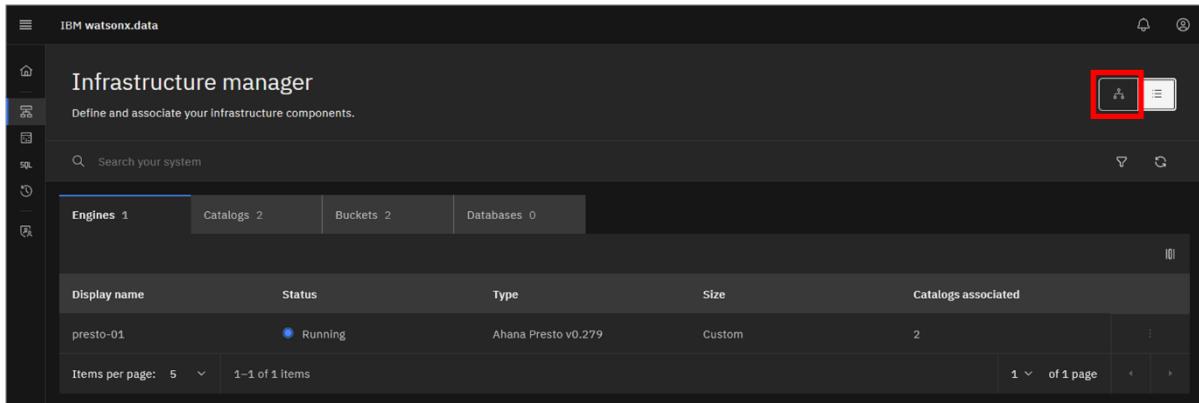
Engines 1 Catalogs 2 Buckets 2 Databases 0

Display name	Status	Type	Size	Catalogs associated
presto-01	Running	Ahana Presto v0.279	Custom	2

Items per page: 5 1–1 of 1 items

Tabs exist for each of **Engines**, **Catalogs**, **Buckets**, and **Databases**. Explore the different tabs to see what information can be found there.

6. Click the **Topology view** icon (the icon to the left of the **List view** icon you just clicked) to switch back to the original graphical view.



IBM watsonx.data

Infrastructure manager

Define and associate your infrastructure components.

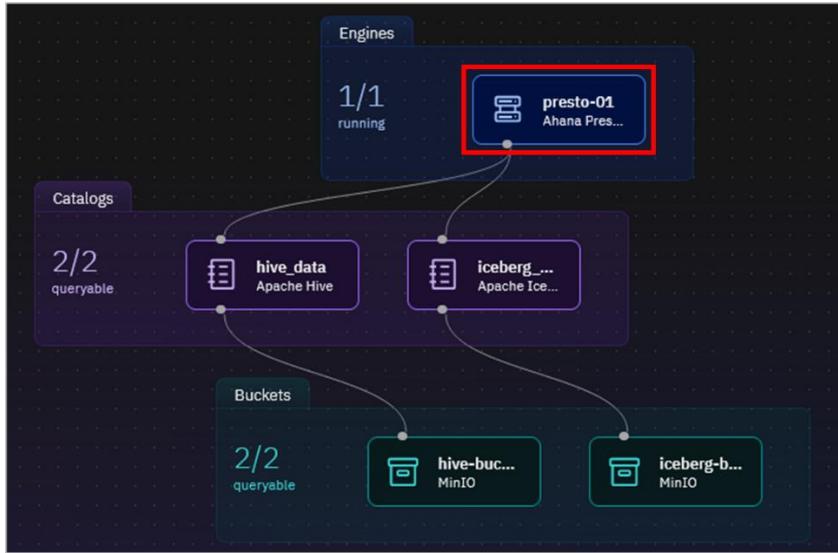
Search your system

Engines 1 Catalogs 2 Buckets 2 Databases 0

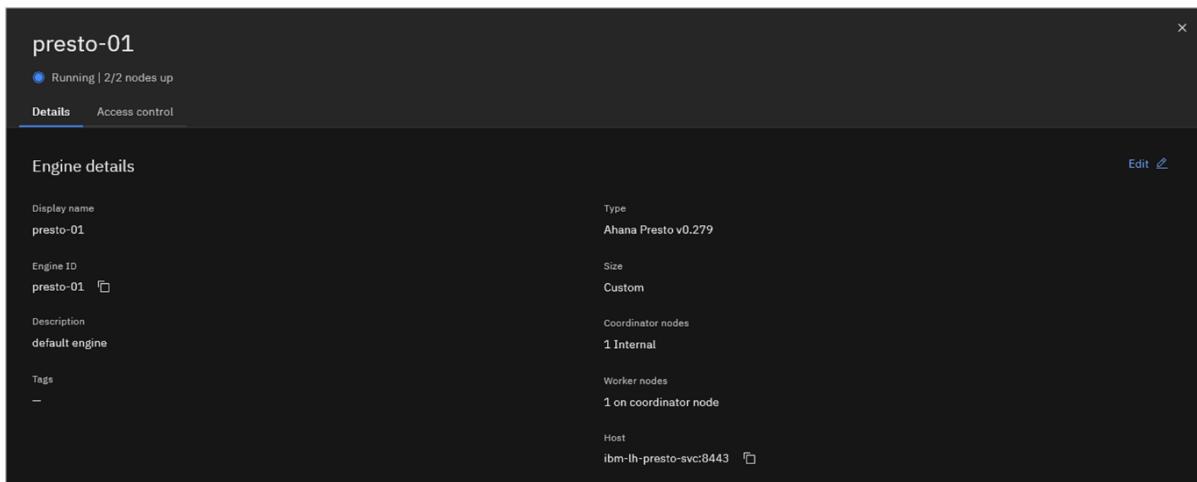
Display name	Status	Type	Size	Catalogs associated
presto-01	Running	Ahana Presto v0.279	Custom	2

Items per page: 5 1–1 of 1 items

7. You can view details associated with each component. Click the **presto-01** engine tile.

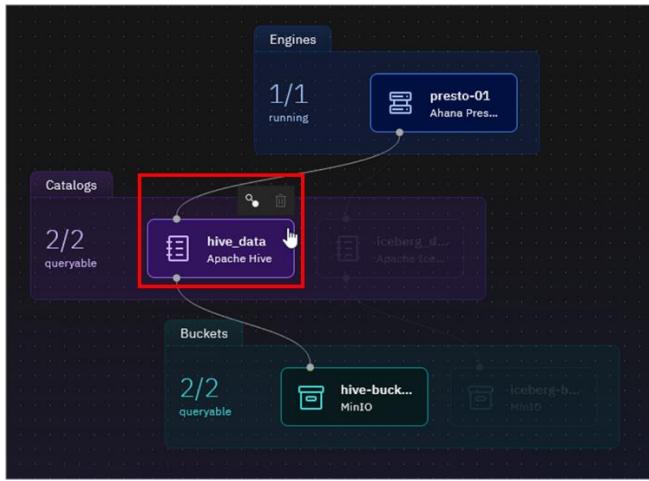


Details including the Presto software version, the number of coordinator nodes, number of worker nodes, size, and host name are shown. Some of the values are editable (by clicking the **Edit** button – but don't click this now). However, as this is an instance of the watsonx.data Developer Edition, you can't do more than just change the name and description of this Presto engine.



8. Click the **X** in the upper right corner of the pane to return to the topology view.
9. Repeat the previous two steps for each of the catalogs and buckets, to see what information is available for them.

10. Hover your mouse pointer over the **hive\_data** catalog tile – but don't click on it.



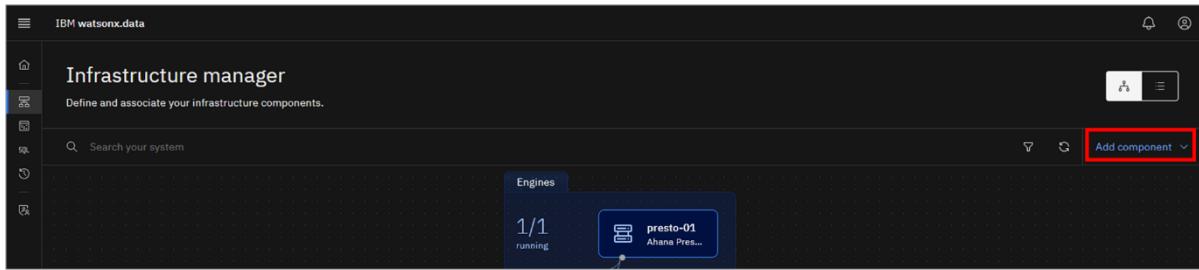
The catalog tile is highlighted, and icons appear above the tile. In this case there are two icons: **Manage associations** and **Delete**. The icons shown depend on the operations that the user can perform, which is based on the state of the component and the permissions that the user has on it.

**Note:** Because the **hive\_data** catalog is a default, system-managed catalog, you are not allowed to remove it and the **Delete** icon is grayed out.

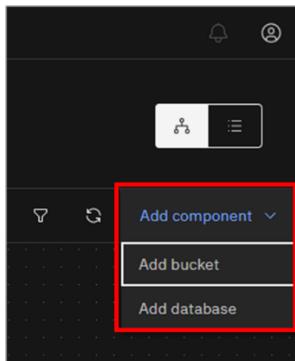
11. As the topology gets more complex, it may be difficult to find the components of interest. The console makes this easy by offering a search facility and the ability to filter what is shown (based on component type and/or the state of the component). Click the **Filter** icon to see the filter options available. Click the **Filter** icon again to close the filter options menu.

The screenshot shows the Watsonx Data topology interface with a search bar ('Search your system') and a filter panel open on the right. The filter panel is titled 'Filter' and includes sections for 'Status' (with 'Running (1)' checked and 'Queryable (4)' unchecked), 'Type' (with 'Engines (1)', 'Catalogs (2)', and 'Buckets (2)' listed), and a 'Reset' button. The main topology view shows the same components as the previous screenshot, with the 'hive\_data' catalog tile highlighted.

12. Click the **Add component** dropdown menu.



As this is the Developer Edition of watsonx.data, you are limited in what additional infrastructure components can be modified or added. You are not permitted to add additional engines, but you can add buckets and databases. The act of adding a new bucket or a database connection also adds an associated catalog, and so there is currently no explicit option for adding a catalog.



13. You will go through the process of adding a new bucket later in this lab. For now, simply close the **Add component** dropdown menu.

### 5.3. Data Manager Page

The **Data manager** page can be used to explore and curate your data. It includes a data objects navigation pane on the left side of the page with a navigable hierarchy of *engine > catalog > schema > table*.

Earlier in this lab guide, the term *schema* was used to describe the definition of a table (including column names and types). Here, schema has a different, albeit related, meaning.

Presto (and many other database systems) organizes tables, views, and other database objects in schemas. Think of a schema as a logical collection or container of related database objects. For example, sales tables might be contained in one schema and marketing tables might be contained in another.

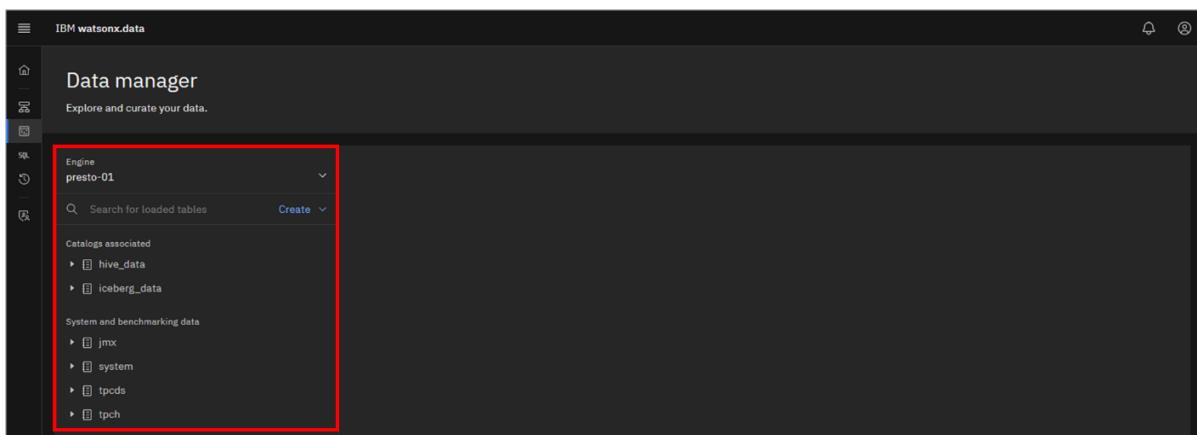
In Presto, tables can be identified in queries by specifying the catalog, schema, and table name (separated by periods). For example:

```
SELECT COUNT(*) FROM DATA_CATALOG.SALES_SCHEMA.CUSTOMER_TABLE
```

In this section you will explore the Data manager page and create your own schema and table.

1. Select the **Data manager** icon (☰) from the left-side menu.

The **Data manager** page opens with a data objects navigation pane on the left side:



The top-level navigation point is the query engine. You start by selecting an engine that is associated with the catalog and bucket you want to manage. As there is only one engine in this environment (**presto-01**), it is selected by default. If this was an environment with multiple engines defined, you would have the choice of selecting any one of the engines you have set up (as the administrator) or that you've been given access to (as a non-administrator).

With the engine selected, you can now navigate through the catalogs associated with the selected engine (the catalogs are listed in the **Catalogs associated** section on the left). Currently this includes the two default catalogs, but this is also where you would see any catalogs you explicitly associate with the engine.

**Note:** The web interface may automatically expand catalogs and schemas, as well as remember when catalogs and schemas were expanded previously. To simplify the screenshots in this lab, catalogs and schemas unrelated to the activity at hand are typically collapsed. So, what you see may not look exactly like what the screenshots in the lab show.

2. Expand each of the **hive\_data** and **iceberg\_data** catalogs by clicking on them.

The screenshot shows the Watsonx Data interface. At the top, it says "Engine" and "presto-01". Below that is a search bar with "Search for loaded tables" and a "Create" button. Under "Catalogs associated", there are two entries: "hive\_data" and "iceberg\_data", both of which are highlighted with red rectangular boxes.

What do you see in these catalogs? The **iceberg\_data** catalog is empty and the **hive\_data** catalog has some schemas (with tables) in it. These pre-defined catalogs (and any catalogs you create) are empty until you create schemas and tables within them. However, some data has been added to this lab environment to provide you with interesting data to play with. These datasets are specific to this environment and are not included when you install `watsonx.data` yourself.

The screenshot shows the Watsonx Data interface with the "hive\_data" catalog expanded. Under "hive\_data", there are three schemas: "gosalesdw", "ontime", and "taxi", all of which are highlighted with red rectangular boxes.

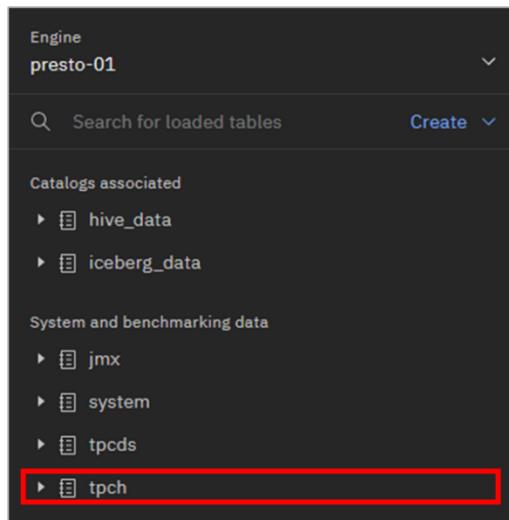
These three datasets are currently included:

1. **gosalesdw**: Sales data for the fictional Great Outdoors Company ([description](#), [schema](#)).
2. **ontime**: Airline reporting on-time performance dataset ([details](#)).
3. **taxis**: A subset of a Chicago taxi public dataset ([details](#)).

Additionally, Presto itself (and by extension watsonx.data) includes the *TPC-H* data set. TPC-H is a decision support benchmark maintained by the Transaction Processing Council (TPC). It is intended to mimic a real-world business workload involving a number of ad-hoc queries running while concurrent modifications are made to the data. The dataset that supports this benchmark includes information on customers, suppliers, orders, part numbers, and more. Datasets of different scale (size) can be generated to test workloads at different scale.

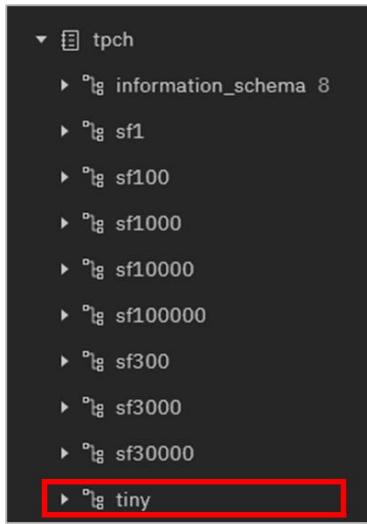
The TPC-H sample data (and other sample data) can be found in the **System and benchmarking data** section, which is below the **Catalogs associated** section.

3. Expand the **tpch** dataset by clicking on it.



Once expanded, the next level down is the schema. In the case of the TPC-H (tpch) data, each schema corresponds to a different scale factor of the dataset. The tables in the **sf100000** schema are 10x the size of the tables in the **sf10000** schema, which are 10x the size of the tables in the **sf1000** schema, and so on. The **tiny** schema is the smallest version of the dataset.

4. Expand the **tiny** schema.



5. Note the tables within this schema. Select the **customer** table. Information about this table is shown in the panel on the right. By default, the **Table schema** (table definition) tab is shown.

A screenshot of the IBM Watsonx.data Data manager interface. The left sidebar shows the engine (presto-01) and a list of loaded tables: sf3000, sf30000, tiny, lineitem, nation, orders, part, partsupp, region, and supplier. The customer table from the tiny schema is selected and highlighted with a red box. The main panel displays the schema for the customer table, also with a red box around it. The schema details are as follows:

Name	Data type	Nullable	Comment
custkey	bigint	YES	
name	varchar(25)	YES	
address	varchar(40)	YES	
nationkey	bigint	YES	

6. Select the **Data sample** tab to see a sample of the data in the customer table.

The screenshot shows the WatsonX Data workspace interface. On the left, there is a navigation pane with a dropdown menu set to 'presto-01'. Below it, a search bar says 'Search for loaded tables' and a 'Create' dropdown menu is open, with 'Create schema' highlighted. The main area shows a table named 'customer' from the 'tpch' catalog with a 'Schema: tiny' label. There are three tabs at the top of the table view: 'Table schema', 'Data sample' (which is selected and highlighted with a red box), and 'DDL'. Below the tabs is a search bar 'Search for records'. The table has columns: custkey, name, address, nationkey, phone, acctbal, and mktsegment. Five rows of data are listed:

	custkey	name	address	nationkey	phone	acctbal	mktsegment
1		Customer#000000001	IVhzIApeRbot,c,E	15	25-989-741-2988	711.56	BUILDING
2		Customer#000000002	XSTf4,NCwDVaWNe6tEgwwfmRchLXak	13	23-768-687-3665	121.65	AUTOMOBILE
3		Customer#000000003	MG9kdTD2WBHm	1	11-719-748-3364	7498.12	AUTOMOBILE
4		Customer#000000004	XxVSJsLAGtn	4	14-128-190-5944	2866.83	MACHINERY
5		Customer#000000005	KvpyuHCpirB84WgAiGV6sYpZq7Tj	3	13-750-942-6364	794.47	HOUSEHOLD

There are different ways that schemas and tables can be created in Presto. One way is through the use of SQL by running CREATE SCHEMA and CREATE TABLE SQL statements (which could be done in Presto's command line interface (CLI) or in watsonx.data's **Query workspace** page). Another approach is to use a third-party database management tool, such as [DBeaver](#).

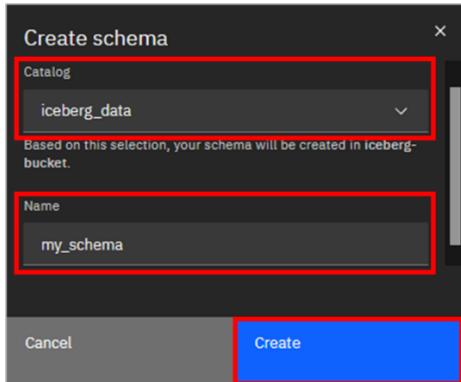
You can also use watsonx.data's **Data manager** page, which allows you to create a schema and upload a data file to define and populate it.

7. Go to the top of the left navigation pane and click the **Create** dropdown menu. Select **Create schema**.

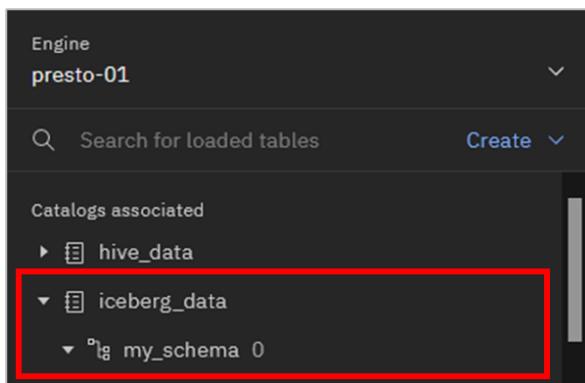
The screenshot shows the WatsonX Data workspace interface. On the left, there is a navigation pane with a dropdown menu set to 'presto-01'. Below it, a search bar says 'Search for loaded tables' and a 'Create' dropdown menu is open, with 'Create schema' highlighted. The main area shows a section titled 'Catalogs associated' with two entries: 'hive\_data' and 'iceberg\_data'. At the bottom of the left pane, there is a section titled 'System and benchmarking data'.

8. In the **Create schema** pop-up window, select/enter the following information, and then click the **Create** button.

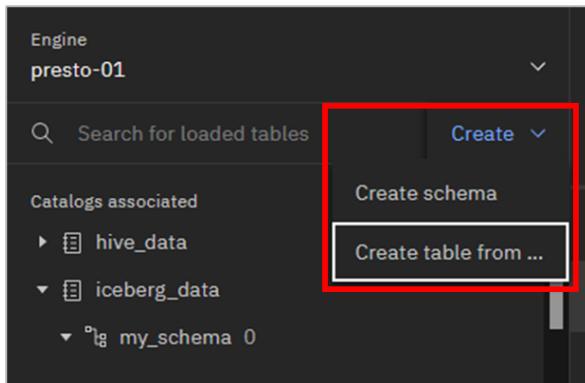
- **Catalog:** iceberg\_data
- **Name:** my\_schema



9. Expand the **iceberg\_data** catalog. The new schema should be listed (but contains no tables).



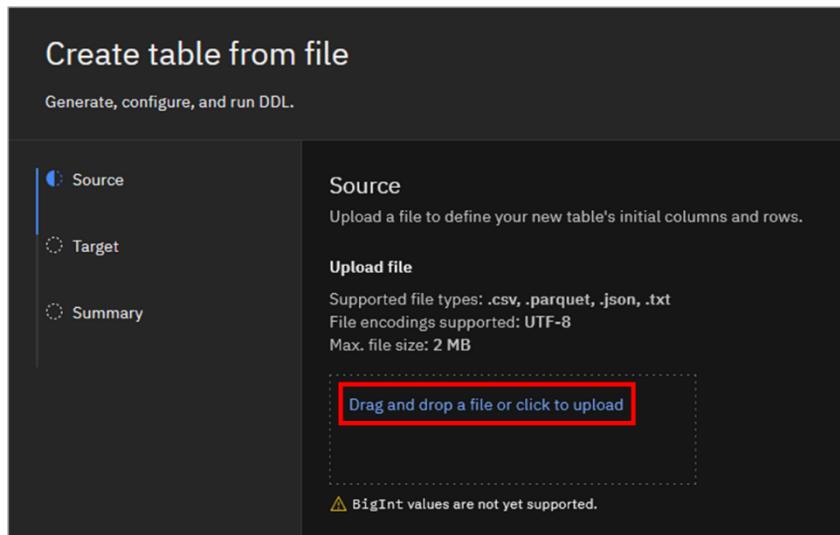
10. Click the **Create** dropdown menu again but this time select **Create table from file**.



The **Create table from file** workflow allows you to upload a small (maximum 2 MB file size) .csv, .parquet, .json, or .txt file to define and populate a new table.

11. Download the sample **cars.csv** file to your desktop (link to file: <https://ibm.box.com/v/data-cars-csv>).

12. For the **Source**, click **Drag and drop a file or click to upload**. Locate the **cars.csv** file you downloaded in the previous step and select it for upload (or simply drag and drop the file into this panel).



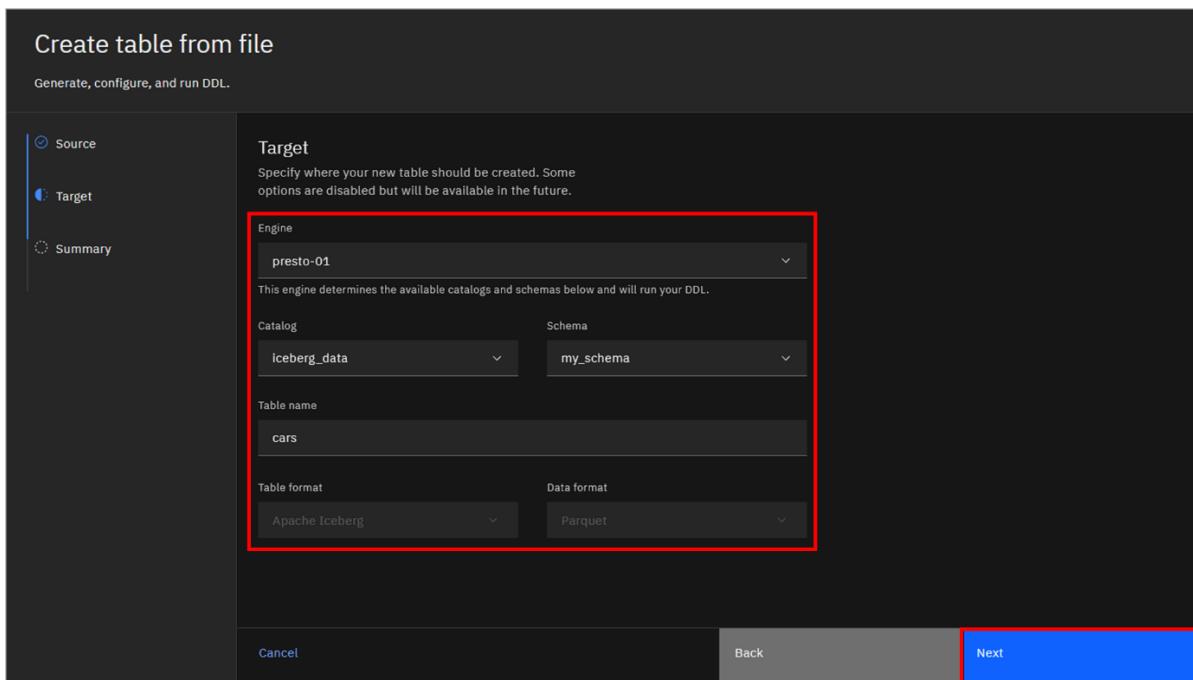
13. Scroll down to view a sample of the data uploaded. The schema of the table is inferred from the data in the file. Click **Next**.

The screenshot shows the 'Table schema configuration' screen. On the left, the sidebar shows 'Source' selected. The main area displays the 'cars.csv' file with its schema. The schema table has columns: Car, MPG, Cylinders, Displacement, Horsepower, Weight, Acceleration, and Model Year. The data table below shows four rows of car information. At the bottom, there are 'Cancel', 'Back', and 'Next' buttons, with 'Next' being highlighted with a red box.

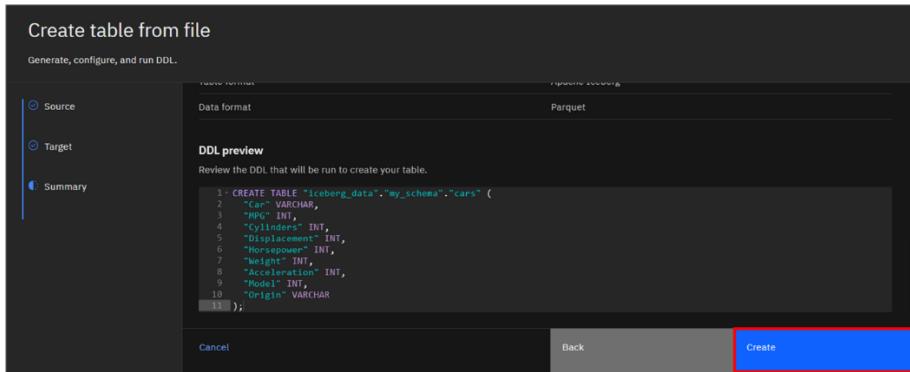
Car	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year
Chevrolet Chevelle Malibu	18	8	307	130	3504	12	70
Buick Skylark 320	15	8	350	165	3693	11.5	70
Plymouth Satellite	18	8	318	150	3436	11	70
AMC Rebel SST	16	8	304	150	3433	12	70

14. For the **Target**, select/enter the following information (some fields are pre-populated and cannot be changed). Once filled in, click **Next**.

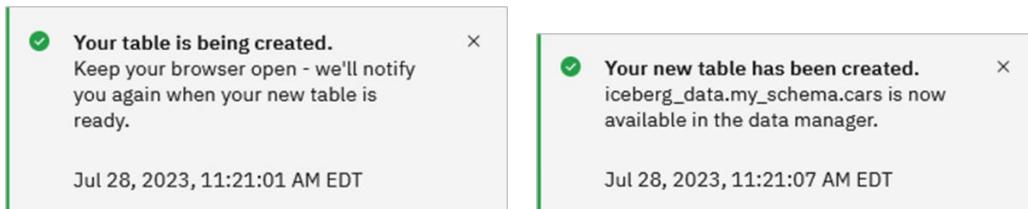
- **Engine:** presto-01
- **Catalog:** iceberg\_data
- **Schema:** my\_schema
- **Table name:** cars
- **Table format:** Apache Iceberg
- **Data format:** Parquet



15. Scroll down to review the **Summary**, which includes the Data Definition Language (DDL) that will be used to create the table. You have an opportunity to alter the DDL statement if you wish, but do not change anything for this lab. Click **Create** to create the table.



You may see a pop-up message in the upper-right corner stating that the table is being created and another one after it completes. Not to worry if you miss them. Clicking on the **Notification** icon (looks like a bell) in the upper-right corner shows past notifications, including those related to creating tables – this icon will have a red dot if there is an unread notification.



16. Navigate to your new table: `iceberg_data > my_schema > cars`.

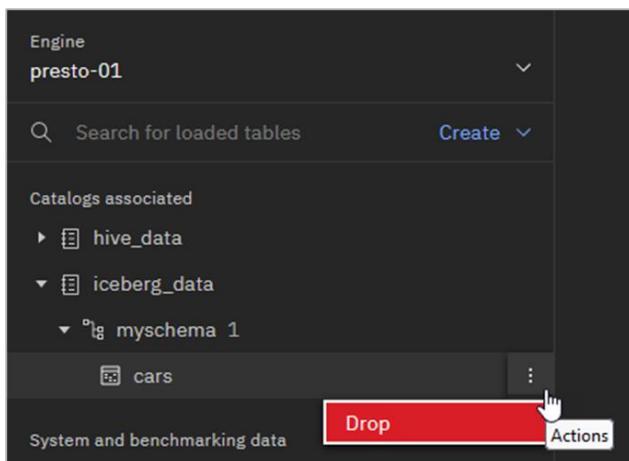
The screenshot shows the 'Data manager' interface. On the left, there's a sidebar with 'Catalogs associated' and a list including 'hive\_data' and 'iceberg\_data'. Under 'iceberg\_data', there's a folder named 'my\_schema 1' which contains a table named 'cars'. This entire path ('iceberg\_data > my\_schema 1 > cars') is highlighted with a red box. To the right, there's a detailed view of the 'cars' table schema:

Name	Data type	Nullable	Comment
car	varchar	YES	
mpg	integer	YES	
cylinders	integer	YES	
displacement	integer	YES	

At the bottom, there are pagination controls: 'Items per page: 25', '1–9 of 9 items', and '1 of 1 page'.

17. Explore the **Table schema**, **Data sample**, and **DDL** tabs. Notice a new tab called **Time travel** that you did not see with the TPC-H dataset earlier. You may view this now, but don't do anything in there. This topic will be covered later.

18. It is not immediately obvious, but there are additional menu options available for the catalogs, schemas, and tables in the navigation pane. Hover your mouse pointer at the far right of the line for the **cars** table. An **overflow menu** icon (vertical ellipses) appears. Click the **overflow menu** icon to see the option to drop the table (don't do that now). Click the **overflow menu** icon again to close it.



## 5.4. Query Workspace Page

Databases and query engines such as Presto have multiple ways that users can interact with the data. For example, there is usually an interactive command line interface (CLI) that lets users run SQL statements from a command terminal. Also, remote applications can use JDBC (Java Database Connectivity) to connect to the data store and run SQL statements.

The watsonx.data user interface includes an SQL interface for building and running SQL statements. This is called the **Query workspace**. Users can write or copy in their own SQL statements, or they can use templates to assist in building new SQL statements.

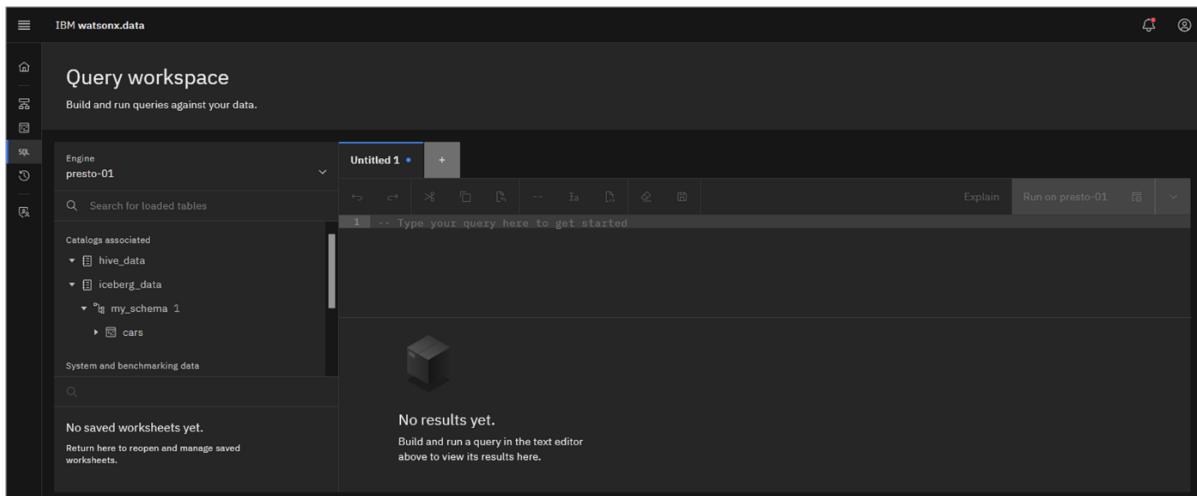
**Note:** Using Presto in watsonx.data, you can use SQL to insert new records into tables, but you aren't currently able to delete or update rows. This is currently possible with Spark SQL, though.

In addition to one-time execution of SQL statements, SQL statements that need to be run repeatedly can be saved to a *worksheet* and be run as often as needed later.

1. Select the **Query workspace** () icon from the left-side menu.

The **Query workspace** page opens with a data objects navigation pane on the left side and a SQL editor (workspace) pane on the right side.

Like in the **Data manager** page, the top of the navigation pane directs you to select an engine to use. It is this engine that will be used to run the SQL statements entered here. The only engine that exists in this lab environment is the **presto-01** Presto engine, which is selected by default.



2. Copy and paste the following text into the **SQL worksheet**. Note that the table you are about to query is identified by a 3-part name that includes the catalog, schema, and table name. Click **Run on presto-01**.

```
select car, avg(mpg) as avg_mpg from iceberg_data.my_schema.cars
group by car order by car;
```

The screenshot shows the Presto Query workspace. At the top, there is a toolbar with various icons. Below the toolbar, the SQL query is written in the editor. A red box highlights the "Run on presto-01" button. The results panel below the editor shows a message: "No results yet. Build and run a query in the text editor above to view its results here." There is also a small icon of a 3D cube.

The query result is displayed at the bottom of the panel.

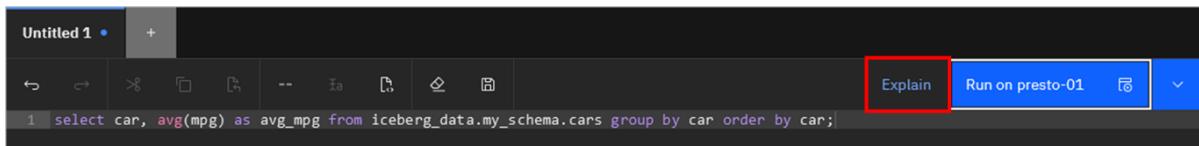
The screenshot shows the Presto Query workspace after the query has been run. The results panel now displays the output of the query. The output is a table with two columns: "car" and "avg\_mpg". The table contains three rows: "AMC Ambassador Brougham" with an average mpg of 13, "AMC Ambassador DPL" with an average mpg of 15, and "AMC Ambassador SST" with an average mpg of 17. A red box highlights the entire results table. In the top right corner of the results panel, it says "Run time: 1.086s" with a green checkmark. The toolbar at the top of the workspace is visible again.

car	avg_mpg
AMC Ambassador Brougham	13
AMC Ambassador DPL	15
AMC Ambassador SST	17

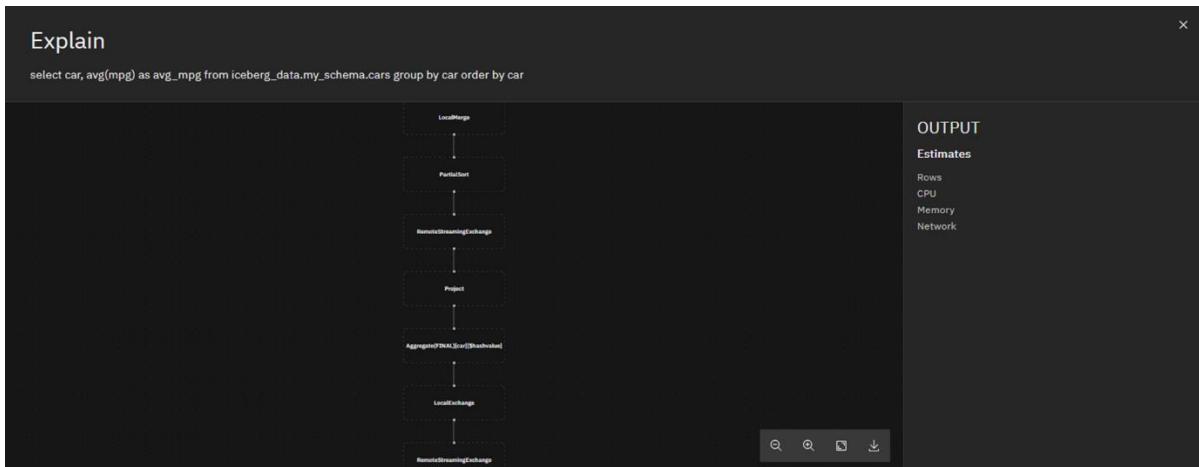
When people are developing or tuning an SQL query, they often want to understand all of the underlying work that is involved in running it – this is commonly referred to as the query's *execution plan*. Presto includes an *explain* facility that shows how Presto breaks up and distributes tasks needed to run a query. A graphical representation of a query's execution plan – what's commonly referred to as *visual explain* – is available from within the Query workspace

page. Watsonx.data uses an EXPLAIN SQL statement on the given query to create the corresponding graph, which can be used to analyze and further improve the efficiency of the query.

3. Click the **Explain** button.



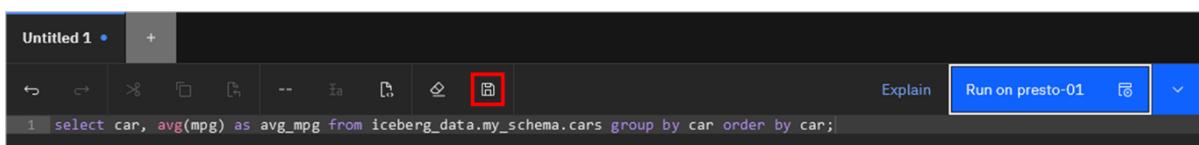
This shows a graphical representation of the query's execution plan. For this example, the query execution plan is relatively simple. Feel free to zoom in and explore the different elements of it. Clicking on a particular stage may show additional information in the pane on the right.



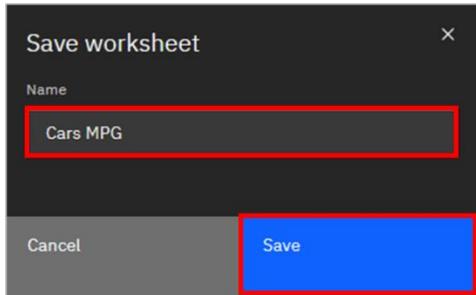
4. Click the X in the upper-right corner of the **Explain** window to close it.

Let's assume this is a query you will want to run time and time again in the future. To do this, you can save it as a worksheet. While this example just has a single SQL statement, imagine a scenario where you have multiple statements in the workspace editor and in a worksheet.

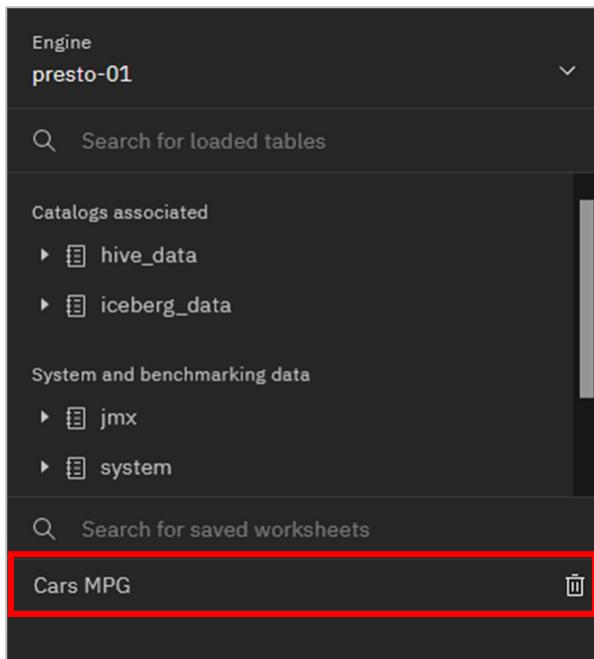
5. Click the **Save** icon (looks like a disk) in the editor menu above the SQL statement.



6. In the **Save worksheet** pop-up window, enter **Cars MPG** for the **Name** and then click **Save**.



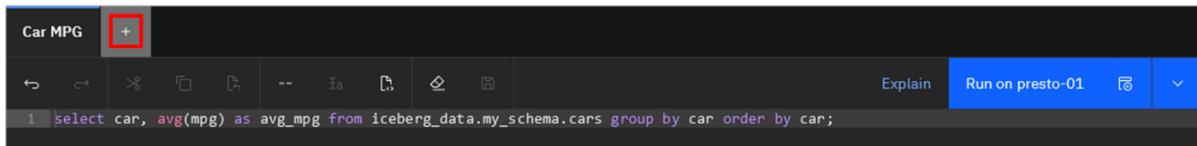
The worksheet is now displayed at the bottom of the left-side navigation pane.



The worksheet can be opened and run in the future as needed, simply by clicking on it (it will open a new SQL tab with the name of the worksheet; if you try to do it now, though, nothing will happen because the tab for this worksheet is already open).

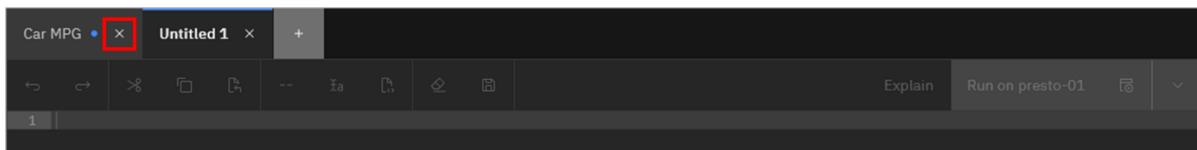
Any worksheet can be deleted at any time by clicking on the **Delete** icon (looks like a garbage can) to the right of the worksheet name.

7. Click the **+** (New worksheet) icon at the top of the current worksheet to create a new, empty worksheet.



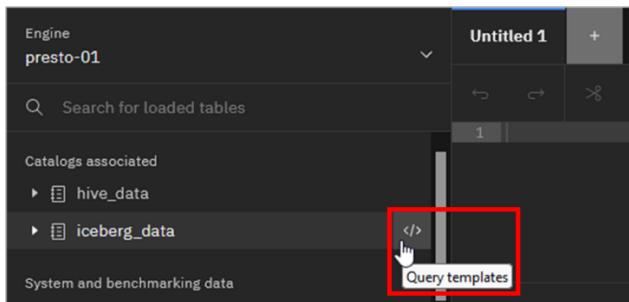
```
1 select car, avg(mpg) as avg_mpg from iceberg_data.my_schema.cars group by car order by car;
```

8. Click the **X** in the **Car MPG** tab to close that worksheet. If asked to confirm closing, click **Close**.

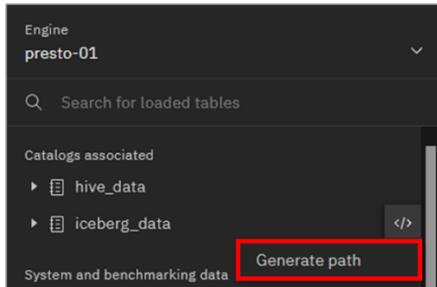


In addition to writing queries from scratch or copying and pasting queries from elsewhere, the Query workspace interface can also assist in generating SQL for tables that are in `watsonx.data`. Let's see what options are available here.

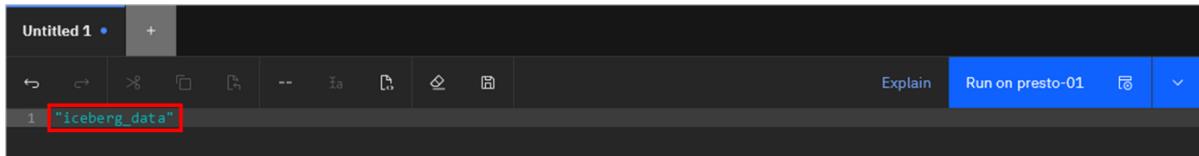
9. On the left-side navigation pane, hover your mouse pointer at the far right of the line for the `iceberg_data` catalog until the **Query templates (</>)** icon appears. When you see this icon, click it.



10. The only template for catalogs and schemas is **Generate path**. Click **Generate path**.

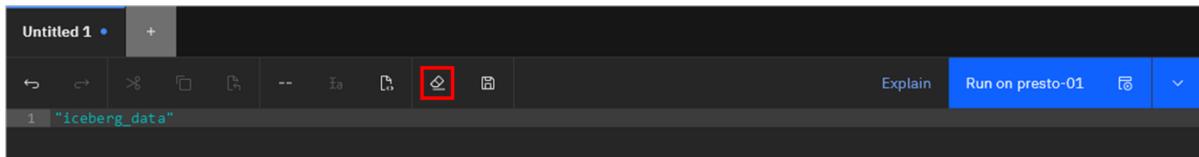


Notice how the name of the catalog was entered into the worksheet. The same thing can be done for schemas, and for schemas the text that gets entered into the worksheet is in the form of “catalog”.schema”.



A screenshot of a SQL worksheet titled "Untitled 1". The query field contains the text "1 \"iceberg\_data\"". The "Run on presto-01" button is highlighted in blue at the top right. A red box highlights the text "iceberg\_data" in the query field.

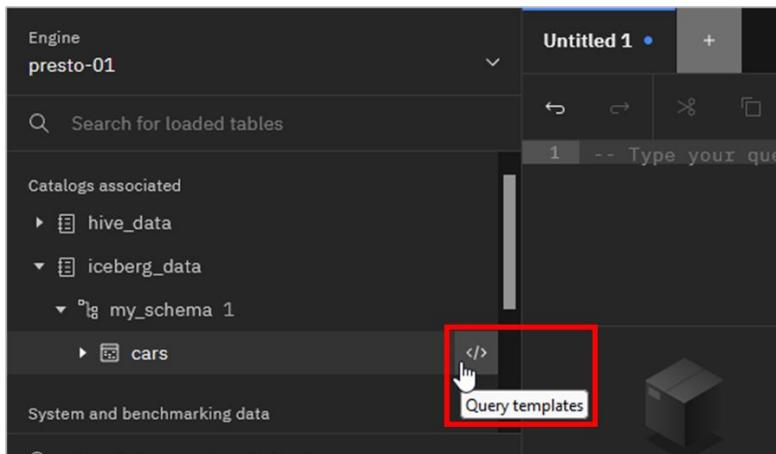
11. Click the **Clear** icon (looks like an eraser) in the menu above the SQL statement. This clears the text that was previously entered.



A screenshot of a SQL worksheet titled "Untitled 1". The query field contains the text "1 \"iceberg\_data\"". The "Run on presto-01" button is highlighted in blue at the top right. A red box highlights the clear icon (eraser symbol) in the toolbar above the query field.

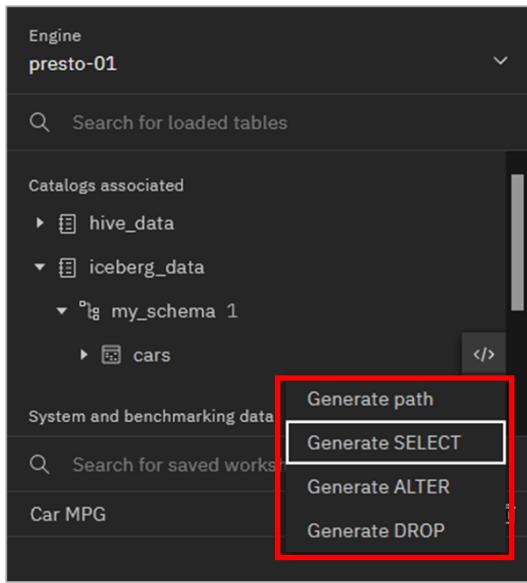
**Note:** If at any point in this lab you are instructed to copy and paste a SQL statement into a worksheet and there is already text there from a previous step you took, the implication is that you should clear the old text out first.

12. Navigate to **iceberg\_data > my\_schema > cars**. Hover your mouse pointer at the far right of the line for the **cars** table until you see the **Query templates (</>)** icon appear. When you see this icon, click it.



A screenshot of the Presto UI interface. On the left, under "Catalogs associated", there is a tree view: "hive\_data" (closed), "iceberg\_data" (open), and "my\_schema" (closed). Under "my\_schema", the "cars" table is listed. To the right of the table list, there is a "Query templates" button with a hand cursor icon pointing at it. A red box highlights this button. The main workspace is titled "Untitled 1" and contains the placeholder text "1 -- Type your query".

13. As you can see, more templates are available when working with tables than what you saw with the catalog:



Selecting **Generate path** generates the table name, as a 3-part name: "catalog"."schema"."table". The other three options can be used to generate SELECT, ALTER, and DROP SQL statements.

14. Click **Generate SELECT**.

15. A basic SELECT statement querying from the cars table is entered into the worksheet. This default statement returns 10 rows of the table including all columns. Click **Run on presto-01**.

The screenshot shows the Apache Presto worksheet titled 'Untitled 1'. It contains the following SQL query:

```
1 SELECT
2 *
3 FROM
4   "iceberg_data"."my_schema"."cars"
5 LIMIT
6 10;
```

The 'Run on presto-01' button at the top right of the worksheet is highlighted with a red box.

As before, the result of the query is shown at the bottom of the screen.

The screenshot shows a database query interface with the following details:

- Untitled 1**: The current tab name.
- Run on presto-01**: The selected execution cluster.
- Query Text:**

```
1 SELECT
2 *
3 FROM
4 "iceberg_data"."my_schema"."cars"
5 LIMIT
6 10;
```
- Result Set:** The query's output is displayed as a table. The columns are: car, mpg, cylinders, displacement, horsepower, weight, acceleration, model, and origin.
- Rows:** The table contains 10 rows of data, starting with:

car	mpg	cylinders	displacement	horsepower	weight	acceleration	model	origin
Chevrolet Chevelle Malibu	18	8	307	130	3504	12	70	US
Buick Skylark 320	15	8	350	165	3693	12	70	US
Plymouth Satellite	18	8	318	150	3436	11	70	US
Audi 100 L 4.2	16	8	264	150	3429	10	70	US
- Run time:** 1.715s

## 5.5. Query History Page

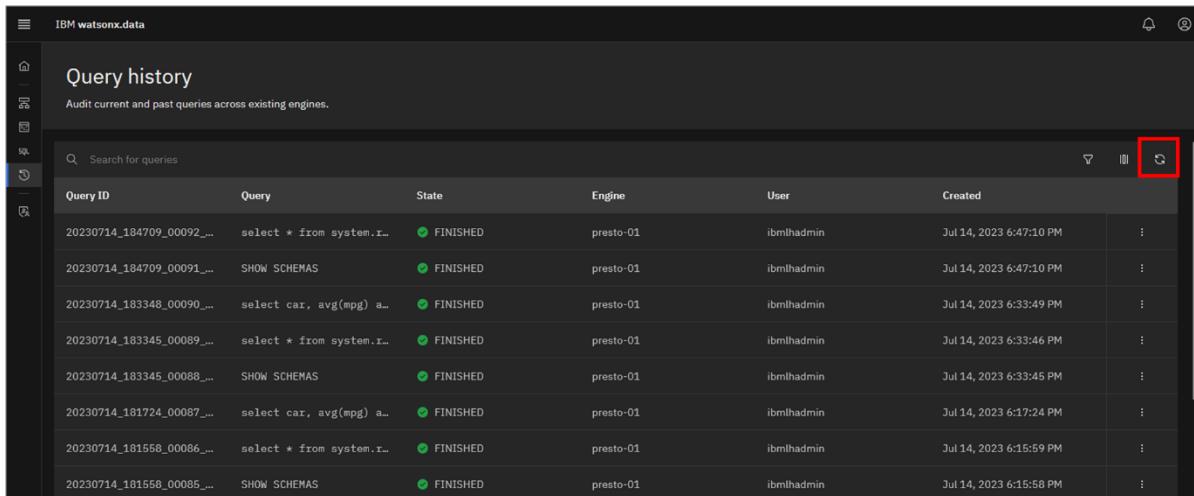
The **Query history** page lets you audit currently running queries and queries that ran in the past, across all the engines defined in the environment. This includes queries that users have explicitly run, as well as queries used in the internal management and function of the environment.

This section is intended to show some of the different ways you can interact with this page.

**Note:** What you see in your list of queries will not match the screenshots shown here.

1. Select the **Query history** (⌚) icon from the left-side menu.

The **Query history** page opens with a list of queries currently running and that ran in the past. If the list appears short or not current, click the **Refresh** icon in the upper-right.



Query ID	Query	State	Engine	User	Created
20230714_184709_00092...	select * from system.r...	FINISHED	presto-01	ibmihadmin	Jul 14, 2023 6:47:10 PM
20230714_184709_00091...	SHOW SCHEMAS	FINISHED	presto-01	ibmihadmin	Jul 14, 2023 6:47:10 PM
20230714_183348_00090...	select car, avg(mpg) a...	FINISHED	presto-01	ibmihadmin	Jul 14, 2023 6:33:49 PM
20230714_183345_00089...	select * from system.r...	FINISHED	presto-01	ibmihadmin	Jul 14, 2023 6:33:46 PM
20230714_183345_00088...	SHOW SCHEMAS	FINISHED	presto-01	ibmihadmin	Jul 14, 2023 6:33:45 PM
20230714_181724_00087...	select car, avg(mpg) a...	FINISHED	presto-01	ibmihadmin	Jul 14, 2023 6:17:24 PM
20230714_181558_00086...	select * from system.r...	FINISHED	presto-01	ibmihadmin	Jul 14, 2023 6:15:59 PM
20230714_181558_00085...	SHOW SCHEMAS	FINISHED	presto-01	ibmihadmin	Jul 14, 2023 6:15:58 PM

Included (at the top) is a search bar to find specific queries of interest. The list can also be filtered by the state of the query (for example, FAILED, FINISHED, and RUNNING), the engine that ran the query, and the user that submitted the query. Information including the query text, state, and when the query was run is shown, but you can customize the columns to show more or less information.

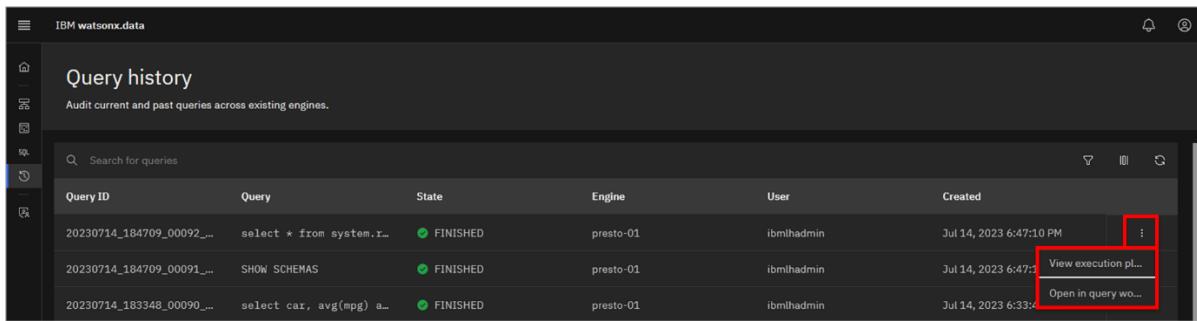
2. Click the **Filter** icon on the right (looks like a funnel). Note the filters available. Click the **Filter** icon again to close it.

Query ID	Query	State	Engine	User
20230714_184709_00092...	select * from system.t...	FINISHED	presto-01	ibmihadmin
20230714_184709_00091...	SHOW SCHEMAS	FINISHED	presto-01	ibmihadmin
20230714_183348_00090...	select car, avg(mpg) a...	FINISHED	presto-01	ibmihadmin
20230714_183345_00089...	select * from system.t...	FINISHED	presto-01	ibmihadmin
20230714_183345_00088...	SHOW SCHEMAS	FINISHED	presto-01	ibmihadmin
20230714_181724_00087...	select car, avg(mpg) a...	FINISHED	presto-01	ibmihadmin
20230714_181558_00086...	select * from system.t...	FINISHED	presto-01	ibmihadmin
20230714_181558_00085...	SHOW SCHEMAS	FINISHED	presto-01	ibmihadmin

3. Click the **Customize columns** icon (looks like a series of vertical lines; it's to the right of the **Filter** icon). Note the columns available to display. Click the **Customize columns** icon again to close it.

Query ID	Query	State	Engine	User	Source	Queued time	Analysis time	Created
20230714_184709_00092...	select * from system.t...	FINISHED	presto-01	ibmihadmin				
20230714_184709_00091...	SHOW SCHEMAS	FINISHED	presto-01	ibmihadmin				
20230714_183348_00090...	select car, avg(mpg) a...	FINISHED	presto-01	ibmihadmin				
20230714_183345_00089...	select * from system.t...	FINISHED	presto-01	ibmihadmin				
20230714_183345_00088...	SHOW SCHEMAS	FINISHED	presto-01	ibmihadmin				
20230714_181724_00087...	select car, avg(mpg) a...	FINISHED	presto-01	ibmihadmin				
20230714_181558_00086...	select * from system.t...	FINISHED	presto-01	ibmihadmin				
20230714_181558_00085...	SHOW SCHEMAS	FINISHED	presto-01	ibmihadmin				

4. Click the **overflow menu** icon (vertical ellipses) at the end of the row for any queries listed. Note the options available. You can view the explain plan for the query and you can bring up the query in the Query workspace. Click the **overflow menu** icon again to close the list.



Query ID	Query	State	Engine	User	Created
20230714_184709_00092...	select * from system.r...	FINISHED	presto-01	ibmihadmin	Jul 14, 2023 6:47:10 PM
20230714_184709_00091...	SHOW SCHEMAS	FINISHED	presto-01	ibmihadmin	Jul 14, 2023 6:47:2...
20230714_183348_00090...	select car, avg(mpg) a...	FINISHED	presto-01	ibmihadmin	Jul 14, 2023 6:33:4...

**Note:** The query history is retrieved directly from Presto, which stores this history information in its `system.runtime.queries` table. However, this table is cleared when Presto is restarted, which means that the query history is lost whenever the engine shuts down. If users want their query history to be maintained over Presto engine restarts, they can choose to create their own query history table and populate it periodically with data from the system table.

## 5.6. Access Control Page

The **Access control** page is used to manage infrastructure access and data access policies.

Security and access control within watsonx.data are based on roles. A role is a set of privileges that control the actions that users can perform. Authorization is granted by assigning a specific role to a user, or by adding the user to a group that has been assigned one or more roles.

Access control at the infrastructural level allows permissions to be granted on the engines, catalogs, buckets, and databases. Roles for these components include Admin, Manager, User, Writer, and Reader (depending on the component).

Access to the data itself is managed through data control policies. Policies can be created to permit or deny access to schemas, tables, and columns.

User account management and access management varies between the different deployment options for watsonx.data. For instance, in the managed cloud service (SaaS), the service owner would need to invite other users to the environment and give them appropriate service access. With the standalone software, users can be added within the console's Access control page. In the Developer Edition, users can be added using a command line tool.

In this section you will add a new user and provide them with privileges over the infrastructure and data. Note that it's not the intention of this lab's instructions to show the results of these privileges (you will not be logging in with other users), the intention is to highlight the process of how you would assign these privileges in the first place.

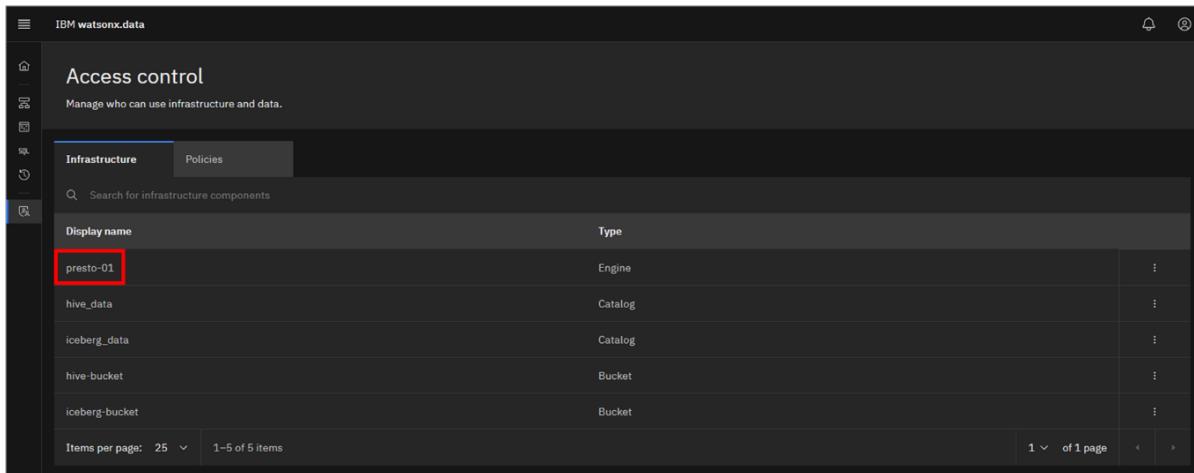
1. Open a terminal command window to the watsonx.data server as the **root** user (remember to use the `sudo` command to become the root user, or you will receive a permission denied error when running the command in the next step).
2. Run the following command to create a non-administrator user (**User** type) with the username **user1** and password **password1**.

```
/root/ibm-1h-dev/bin/user-mgmt add-user user1 User password1
```

3. In the watsonx.data user interface running in your browser, click the Access control (key) icon from the left-side menu.

The **Access control** page opens in the **Infrastructure** tab, with the currently defined engines (1), catalogs (2), and buckets (2) listed (when you add databases, they will be listed here as well).

4. Click on **presto-01** (it will turn into a hyperlink when you hover over it).

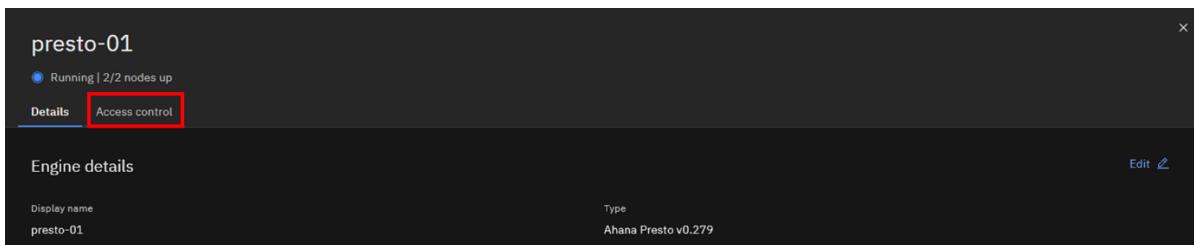


The screenshot shows the Access control interface with the Infrastructure tab selected. A search bar at the top right says "Search for infrastructure components". Below it is a table with columns "Display name" and "Type". The table contains the following data:

Display name	Type
presto-01	Engine
hive_data	Catalog
iceberg_data	Catalog
hive-bucket	Bucket
iceberg-bucket	Bucket

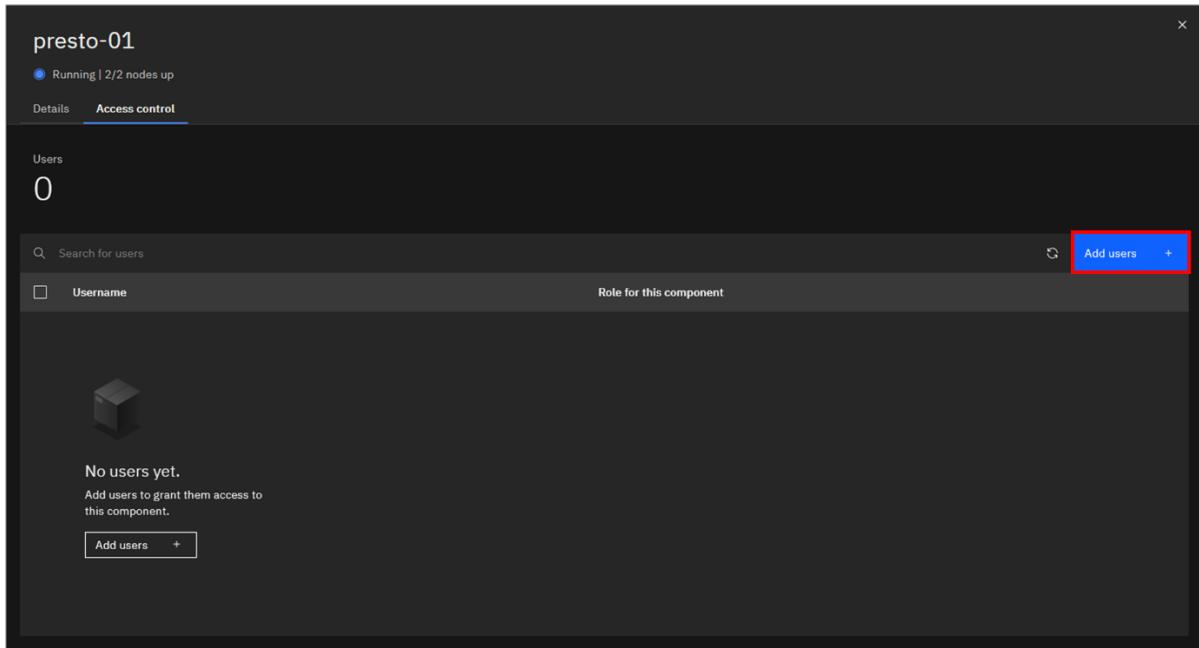
At the bottom, there are pagination controls: "Items per page: 25" and "1–5 of 5 items".

5. Select the **Access control** tab.



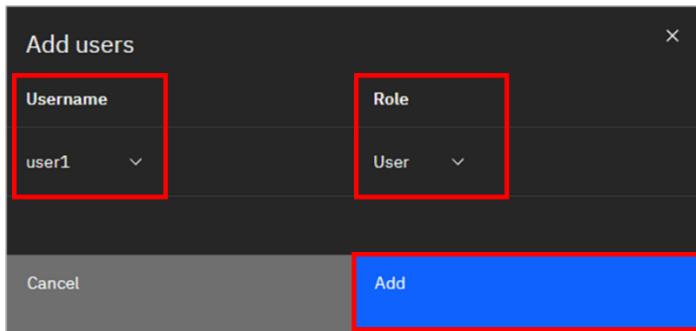
The screenshot shows the details for the presto-01 engine. At the top, it says "presto-01" and "Running | 2/2 nodes up". Below that, there are two tabs: "Details" (selected) and "Access control" (highlighted with a red box). Under "Engine details", it shows the display name "presto-01" and type "Ahana Presto v0.279". There is also an "Edit" button.

6. Click the blue **Add users** button on the right.



7. The **Add users** pop-up window is displayed. For **Username**, select **user1**. For **Role**, select **User**. Click **Add**.

**Note:** If user1 isn't listed then cancel the operation, refresh your browser tab (for instance, using <F5> in the Firefox browser), and repeat the steps.



8. Note how the user's role has been added. Click the X in the upper-right corner to close the access controls for the Presto engine.

presto-01

Running | 2/2 nodes up

Details Access control

Users 1

Search for users Add users

Username	Role for this component
user1	User

Items per page: 5 1-1 of 1 items

9. Optionally, repeat Steps 4 - 8 to assign the following permissions to **user1** (but rather than clicking on presto-01, you'll click on the component referenced below instead). (This is just intended to show the different roles that can be assigned to the other infrastructure component types.)

- iceberg\_data (Catalog): User
- iceberg-bucket (Bucket): Reader

In a real-world scenario where a user will be querying data from a table, that user will need to be given a minimum of **User access to an engine** (to be able to run the query), **User access for the catalog** associated with the data (to be able to see the schema information associated with the table), and **Reader access to the bucket** associated with the data (to be able to read the data from object storage).

Additionally, a policy has to be created to permit the user to access the table in question.

10. Select the **Policies** tab.

IBM watsonx.data

Access control

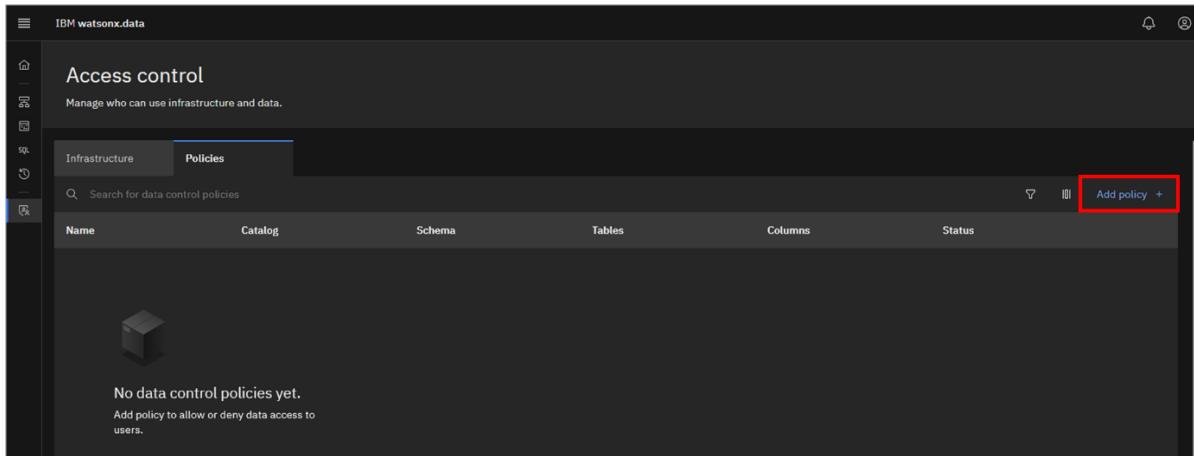
Manage who can use infrastructure and data.

Infrastructure Policies

Search for infrastructure components

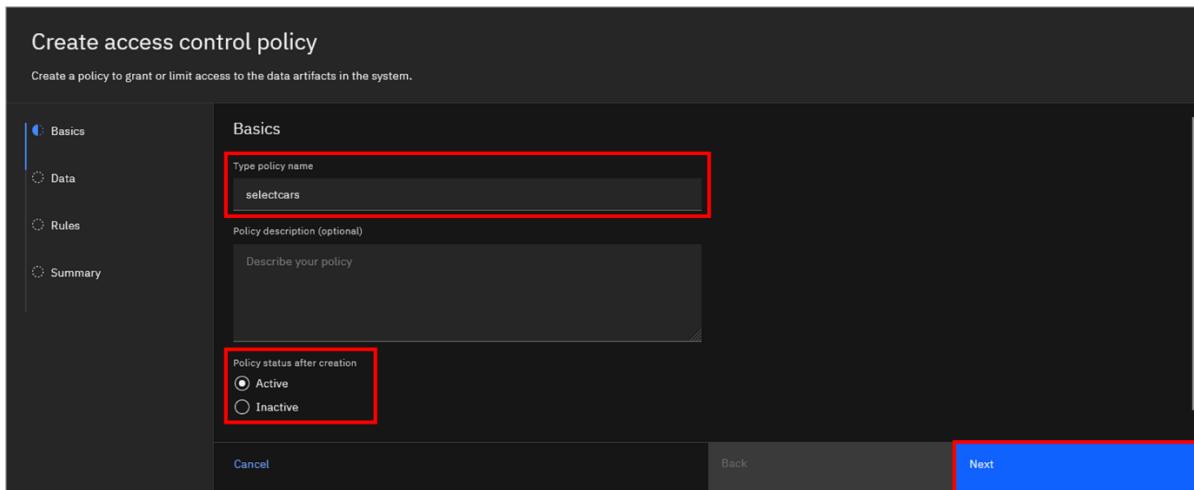
Display name	Type
--------------	------

11. Click **Add policy** on the right.



The screenshot shows the 'Access control' interface in the IBM WatsonX Data service. The 'Policies' tab is selected. A red box highlights the 'Add policy +' button in the top right corner of the main content area. Below the button, there is a message: 'No data control policies yet. Add policy to allow or deny data access to users.'

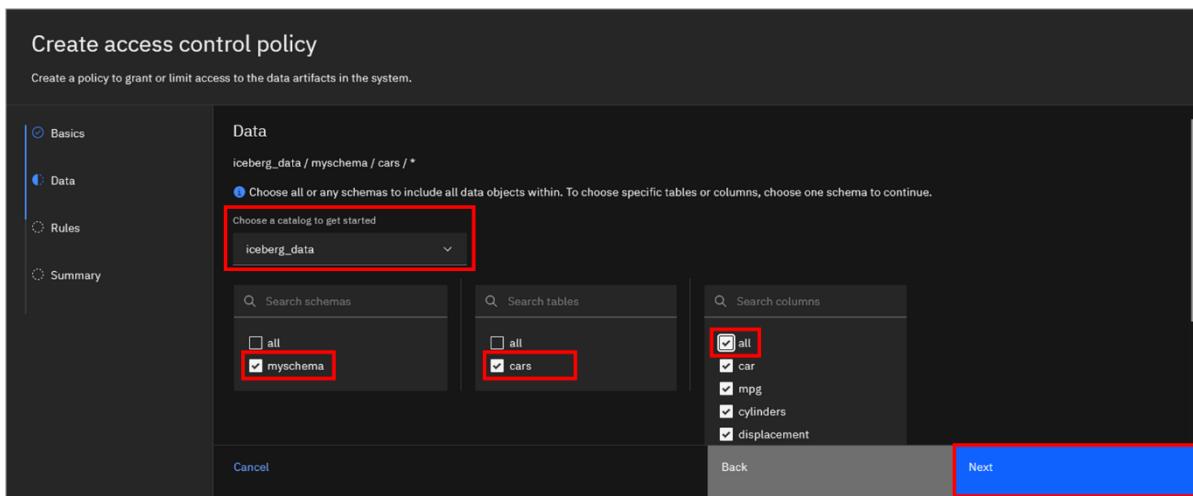
12. Enter **selectcars** in the **Type policy name** field, select **Active** for **Policy status after creation**, and then click **Next**.



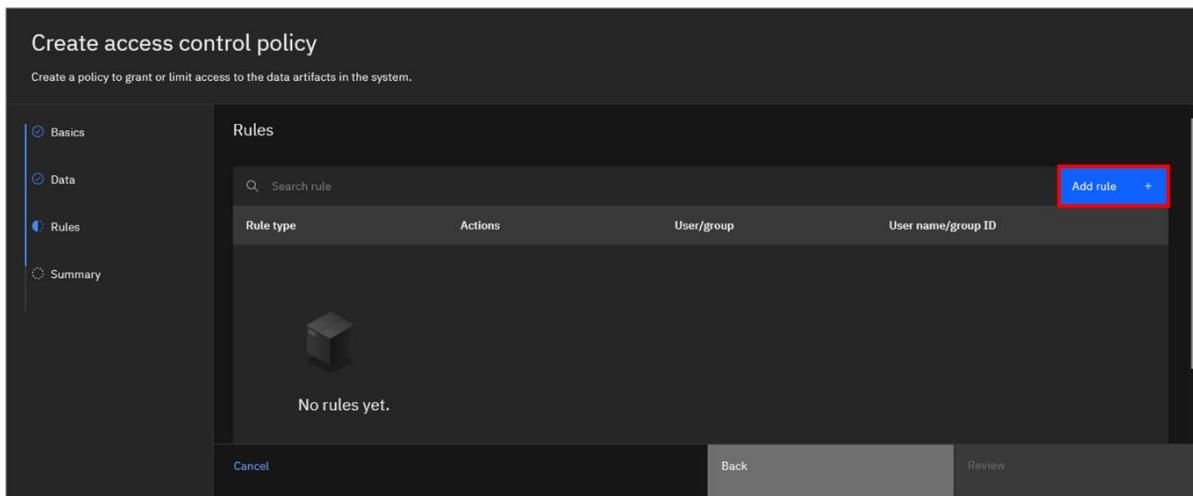
The screenshot shows the 'Create access control policy' dialog. The 'Basics' step is selected. The 'Type policy name' field contains 'selectcars'. The 'Policy status after creation' section shows 'Active' selected. The 'Next' button at the bottom right is highlighted with a blue box.

13. Click on the **Choose a catalog to get started** dropdown and select **iceberg\_data**. A list of schemas is then displayed.

- a. Select the checkbox for the **myschema** schema.
- b. A list of the tables in the schema is then displayed; select the checkbox for the **cars** table.
- c. A list of the columns in the table is then displayed; select the checkbox for **all** columns, which selects all columns.
- d. Click **Next**.

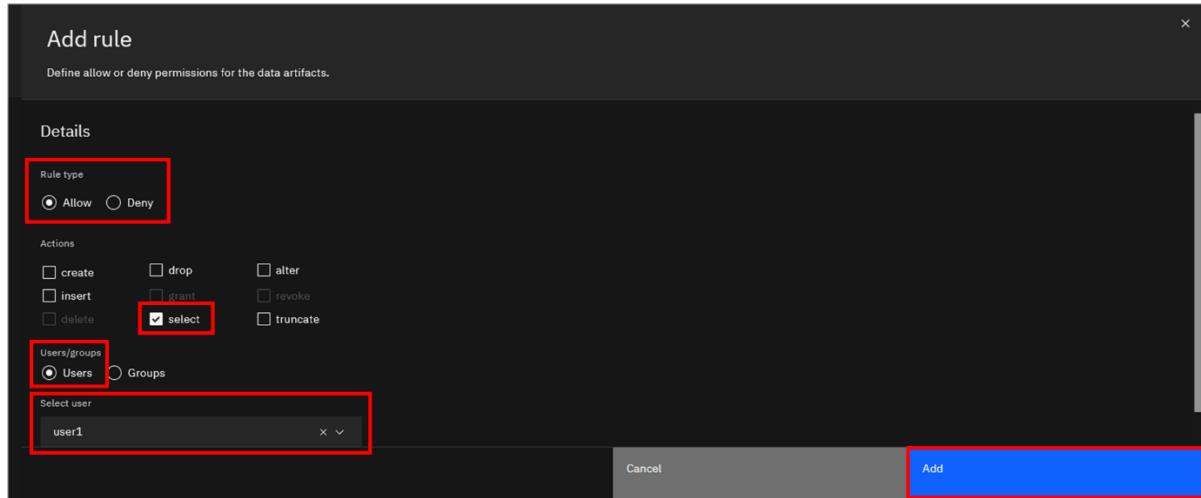


14. One or more rules can now be added to the policy. Click **Add rule** on the right.

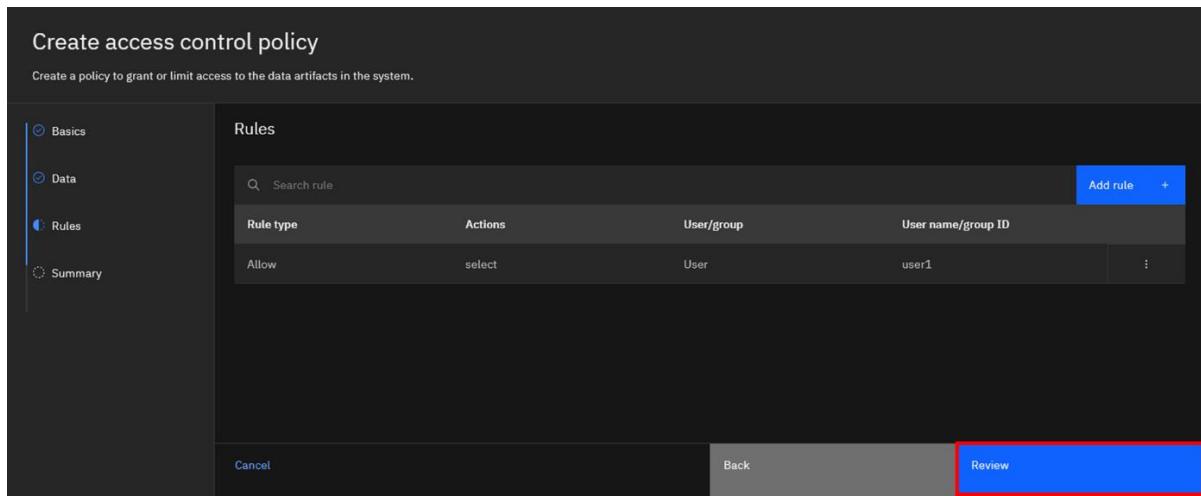


15. Make the following selections and then click the **Add** button:

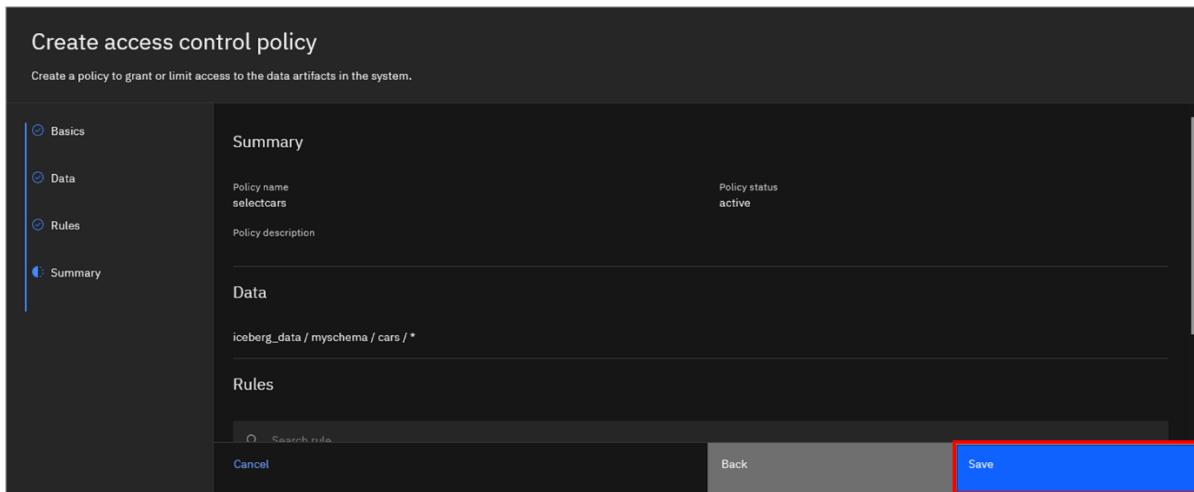
- a. For the **Rule type**, select **Allow**.
- b. For **Actions**, select the **select** checkbox.
- c. For **Users/group**, select **Users**.
- d. In the **Select user** dropdown, select **user1**.



16. With the rule added to the policy, click **Review**.



17. Review the policy details you've entered and click **Save**.



The policy should now be listed. Because you specified for the policy to be active upon creation, it is in fact active (as you can see by looking at the **Status** column).

The screenshot shows the 'Access control' interface with the 'Policies' tab selected. The table displays the following data:

Name	Catalog	Schema	Tables	Columns	Status
selectcars	iceberg_data	myschema	cars	All	Active

You have just given **user1** the permissions needed to query the data within the cars table.

## 6. Working with Presto

Up to this point, you have seen how Presto is integrated into watsonx.data and the watsonx.data user interface. However, you can also work directly with Presto.

For example, Presto includes a terminal-based interactive command line interface (CLI) that allows you to run SQL statements. Presto also has a web-based graphical user interface for getting information on query activity in the system.

Additionally, users can connect 3<sup>rd</sup> party management and analytics tools to the Presto server. Likewise, applications can use JDBC (Java Database Connectivity) to work with data in watsonx.data, by connecting to the Presto server (this topic is beyond the scope of this lab).

### 6.1. Presto Command Line Interface (CLI)

Presto CLI is a terminal-based interactive shell that can be used to run queries. You can connect to the Presto server either through Presto CLI installed as part of the *ibm-lh-client* package or through Presto CLI installed separately.

Presto CLI is pre-installed as part of watsonx.data Developer Edition. It is started by using the `presto-cli` command (located in `/root/ibm-lh-dev/bin`).

Presto uses a 3-part name to identify tables: catalog.schema.table. These identifiers can be enclosed in double quotes as needed (for example, "mycatalog"."myschema"."mytable"). Note that double quotes are needed if you're using any special characters in a name, like a hyphen.

1. Open a terminal command window to the watsonx.data server as the **root** user.
2. Run the following two commands to open a Presto CLI interactive terminal.

```
cd /root/ibm-lh-dev/bin
```

```
./presto-cli
```

**Note:** In the Presto CLI interactive terminal, you can use the cursor control (arrow) keys on your keyboard to scroll through previously entered statements, and edit the current statement you're about to run.

- Run the following command to list the catalogs that have been registered in Presto (Note: Presto requires that commands end with a semicolon.)

```
show catalogs;
```

The output should be similar to the text below.

```
Catalog
-----
hive_data
iceberg_data
jmx
system
tpcds
tpch
(6 rows)
```

- Run the following query. It should return a count of 1,500 rows.

```
select count(*) from tpch.tiny.customer;
```

Rather than explicitly including the catalog name and schema name for every statement, you can set the default catalog and schema for a session when you start the CLI. You will try this shortly.

You can also tell Presto which schema you want to work with by using the USE command.

- First, run the following command to see the schemas in the **tpch** catalog.

```
show schemas in tpch;
```

The output should be similar to the text below.

```
Schema
-----
information_schema
sf1
sf100
sf1000
sf10000
sf100000
sf300
sf3000
sf30000
tiny
(10 rows)
```

6. Run the following USE command to set the schema for the current session:

```
use tpch.tiny;
```

7. Run the following query. Note how the catalog and schema aren't required. Again, this should return a count of 1,500 rows.

```
select count(*) from customer;
```

Alternatively, you can provide the session catalog and schema (or just the catalog) when you start the Presto CLI. Note that even if you specify the catalog and schema when starting the Presto CLI or by running the **USE** command, you can still access tables in other schemas and catalogs. You just have to specify the full 3-part name for them (or USE the schema in question to change the default for the session).

8. Quit from the Presto CLI by running the following command.

```
quit;
```

9. Start the Presto CLI again, but this time specify the session catalog and schema as command line options.

```
./presto-cli --catalog tpch --schema tiny
```

10. Run the following query. Note how this time the catalog and schema aren't required. Again, this should return a count of 1,500 rows.

```
select count(*) from customer;
```

11. Quit from the Presto CLI by running the following command.

```
quit;
```

Next, you will create a new schema and table. When using Presto to create a schema in watsonx.data, you must specify the object storage bucket associated with the catalog.

12. Start the Presto CLI and specify that you intend to use the **iceberg\_data** catalog.

```
./presto-cli --catalog iceberg_data
```

13. Run the following SQL statement to create a new schema in the catalog.

```
create schema if not exists newschema  
with (location='s3a://iceberg-bucket/newschema');
```

14. Run the following command to see a list of the schemas in the catalog being used.

```
show schemas;
```

The **newschema** schema should be listed.

15. Run the following SQL statements to create a new table in this schema, populate the table with some data, and then query the table.

```
create table newschema.users (id int, name varchar, age int);
```

```
insert into newschema.users values (1, 'Robert', 54);
```

```
insert into newschema.users values (2, 'Susan', 37);
```

```
select * from newschema.users;
```

The output of the final SELECT statement should look similar to what is shown below.

id	name	age
1	Robert	54
2	Susan	37

(2 rows)

16. Run the following two commands to show the table in the new schema.

```
use newschema;
```

```
show tables;
```

The output of the SHOW TABLES statement should look similar to what is shown below.

Table
users

(1 row)

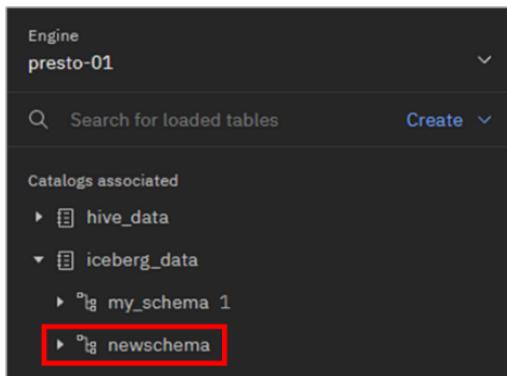
17. Quit from the Presto CLI by running the following command.

```
quit;
```

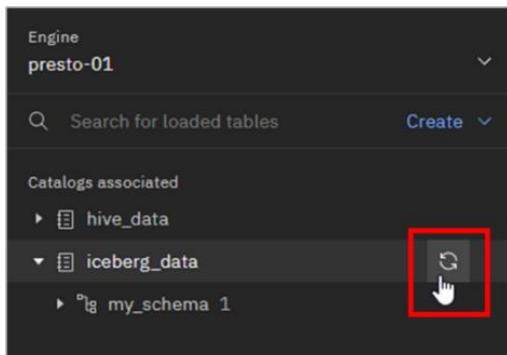
18. Open the watsonx.data user interface in a browser window (if you don't already have one open).

19. Select the **Data manager** (grid) icon from the left-side menu.

20. Expand the **iceberg\_data** catalog to see the new schema you created (**newschema**).



**Note:** If you don't see the schema listed then hover your mouse pointer over the far right of the line for the **iceberg\_data** catalog until you see the **Refresh** icon appear. Click the **Refresh** icon. You should now see the schema listed as in the above image.



21. Expand the **newschema** schema to see the new table you created (**users**).

The screenshot shows the Presto UI interface. At the top, it says "Engine" and "presto-01". Below that is a search bar "Search for loaded tables" and a "Create" dropdown. Under "Catalogs associated", there is a tree view. The "iceberg\_data" catalog has a child node "my\_schema 1". This node has a child node "newschema 1", which is highlighted with a red box. Inside "newschema 1" is a table named "users".

The table you created through the Presto CLI is visible through the watsonx.data user interface. This is one of the benefits of having a shared metastore. In the future, any query engines that get associated with the `iceberg_data` catalog would also be able to work with this table.

## 6.2. Presto Web Interface

Presto includes its own web interface (console) for monitoring and managing Presto queries. It's a great place to get information about running queries and completed queries. This includes the query text, query state, the name of the user that ran the query, and the percentage complete if it's still running.

Each query is assigned a unique Query ID and clicking on the ID brings up a Query Details page with additional information regarding the query.

1. From your computer, open the Presto console in your browser (<http://192.168.252.2:8080>). No credentials are required.

The screenshot shows the "CLUSTER OVERVIEW" page of the Presto console. At the top, it displays cluster statistics: VERSION 0.279, ENVIRONMENT PRODUCTION, UPTIME 2.62h, and a Logout button. Below this are four main metrics: RUNNING QUERIES (0), ACTIVE WORKERS (1), ROWS/SEC (0.00), QUEUED QUERIES (0), RUNNABLE DRIVERS (0.00), BYTES/SEC (0), and BLOCKED QUERIES (0), RESERVED MEMORY (B) (0), and WORKER PARALLELISM (0.00). At the bottom, the "QUERY DETAILS" section shows a table with columns for User, Source, Query ID, Resource Group, and Query Text. The table is currently empty with the message "No queries matched filters".

You immediately start on the **Cluster Overview** page. The upper portion of the page includes various metrics regarding the environment.

The **Query Details** section at the bottom of the page lists queries matching the **State** filter being applied. By default, only actively running queries are shown and so you probably don't see any queries listed here.

2. Select the **Finished** button to include queries that have finished running.

A screenshot of the 'QUERY DETAILS' section. At the top, there is a search bar labeled 'User, source, query ID, resource group, or query text'. Below it is a filter bar with 'State:' dropdown set to 'Running' (which is checked), and other options like 'Queued', 'Finished' (which is also checked and highlighted with a red border), 'Failed', 'Sort', 'Reorder Interval', and 'Show'. A message 'No queries matched filters' is displayed below the filter bar.

You should now see queries listed. You may recognize some as queries you ran, and others may have been run internally by the system.

A screenshot of the 'QUERY DETAILS' section showing two finished queries. The first query is from 'ibmadmin' at 1:29pm, with the command 'SHOW TABLES FROM "iceberg\_data"."newschema"'. The second query is from 'ibmadmin' at 1:23pm, with the command 'SHOW CREATE TABLE "iceberg\_data"."my\_schema".cars'. Both queries are listed under the 'FINISHED' status. The interface includes a search bar, a filter bar with 'State:' dropdown set to 'Running', and other buttons like 'Queued', 'Failed', 'Sort', 'Reorder Interval', and 'Show'.

3. Click the **Query ID** link for a query that looks interesting to you.

A screenshot of the 'QUERY DETAILS' section showing a single finished query selected. The query ID '20230801\_172316\_00065\_d7wm3' is highlighted with a red box. The query was run by 'ibmadmin' at 1:23pm. The command is 'SELECT \* FROM information\_schema.columns WHERE table\_catalog = \'iceberg\_data\' AND table\_schema = \'my\_schema\' AND table\_name = \'cars\''. The interface includes a search bar, a filter bar with 'State:' dropdown set to 'Running', and other buttons like 'Queued', 'Failed', 'Sort', 'Reorder Interval', and 'Show'.

4. This opens a new browser window with the **Query Details** page for that specific query. There are multiple tabs, and you start in the **Overview** tab by default.

QUERY DETAILS		VERSION	ENVIRONMENT	UPTIME	<a href="#">Logout</a>
20230801_172939_00068_d7wm3.s		0.279	PRODUCTION	• 2.90h	
		<a href="#">Overview</a>	<a href="#">Live Plan</a>	<a href="#">Stage Performance</a>	<a href="#">Splits</a>
					<a href="#">JSON</a>
<b>FINISHED</b>					
<b>Session</b>		<b>Execution</b>			
User	ibmihadmin 	Resource Group <b>global</b>			
Principal	IBMLHPrincipal [name=ibmihadmin, role=Administrator, groups=null]	Submission Time <b>2023-08-01 1:29pm</b>			
Source	presto-go-client	Completion Time <b>2023-08-01 1:29pm</b>			
Catalog	iceberg_data	Elapsed Time <b>833.11ms</b>			
Schema	newschema	Prerequisites Wait Time <b>7.82ms</b>			
Client Address	172.18.0.4	Queued Time <b>335.07us</b>			
Client Tags		Planning Time <b>242.56ms</b>			
Session Properties		Execution Time <b>607.73ms</b>			
Resource Estimates		Coordinator <b>172.18.0.6</b>			

5. Scroll down the page to familiarize yourself with the information available.
  6. Select the **Live Plan** tab at the top of the page.

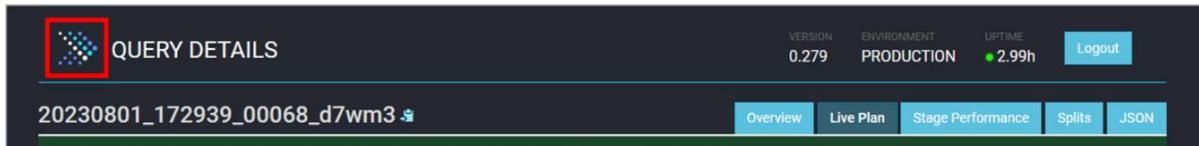
## QUERY DETAILS

---

Here you can see the query's execution plan and the various steps involved in running the query (your execution plan will look different as it depends on the query you chose). This output is similar to what you see in the visual explain output for watsonx.data.

The screenshot shows the DataRobot interface for a specific query. At the top, there's a navigation bar with 'QUERY DETAILS' on the left and 'VERSION 0.279 ENVIRONMENT PRODUCTION UPTIME 2.99h LOGOUT' on the right. Below this is a timestamp '20230801\_172939\_00068\_d7wm3.s'. The main area has tabs for 'Overview', 'Live Plan', 'Stage Performance', 'Splits', and 'JSON'. A large green box labeled 'FINISHED' contains a flowchart. The flowchart starts with 'Stage 0' (GPU: 0.0ms, Buffer: 0.0ms, Read: 0.0ms, Write: 0.0ms, Split: 0.0, F1: 1.0, Duration: 0.0s) which takes 'Input: 400 / 1 row' and outputs 'Input: 100 / 1 row' to 'Stage 1'. Stage 1 (GPU: 0.0ms, Buffer: 0.0ms, Read: 0.0ms, Write: 0.0ms, Split: 0.0, F1: 1.0, Duration: 0.0s) also takes 'Input: 400 / 1 row' and outputs 'Input: 100 / 1 row' to the final stage. The final stage consists of two parallel steps: 'LoadFeature [data\_name=ALL, LULL, LULL]' and 'Perturb [data\_name=ALL, LULL, LULL]'. A note at the bottom right says 'Scroll to zoom. Click stage to view additional statistics'.

7. Click the **Presto logo** in the top-left to return to the **Cluster Overview** page.



8. When you're done exploring the Presto console, close the browser window (and any other Presto console windows that are still open).

## 7. Working with MinIO

### 7.1. Introduction to Object Storage

Object storage is a type of storage architecture where data is managed as *objects*. Each object typically includes the data itself, a variable amount of metadata, and a globally unique identifier. The basic unit of storage is an object and objects are organized in *buckets* (think of a bucket like a high-level directory or folder).

Object storage systems allow retention of massive amounts of structured, semi-structured, and unstructured data in which data is written once and read once or many times. Object storage is low cost, with near unlimited capacity and scalability.

Cloud object storage vendors have designed their offerings with extremely high levels of durability and reliability. The most notable provider of object storage is Amazon S3 (Simple Storage Service). Most other vendors (including IBM Cloud) offer S3 API-compatible object storage. To be clear, S3 API-compatible doesn't mean that the object storage is running in Amazon. When the industry talks of object stores being S3 API-compatible (or simply S3-compatible), it means that they support the same API set that Amazon created around their S3 object storage. In doing so, applications and tools that support S3 should work against S3-compatible object storage as well.

Clients consider characteristics such as performance, price, and workload type when deciding upon a storage format for a particular type of data or workload. One of the main reasons for the rising popularity of the lakehouse architecture is its use of low-cost object storage. Data warehouses, on the other hand, have traditionally used block storage to store data, which offers high performance, but is significantly more costly.

A major appeal of watsonx.data is that vast amounts of data can be stored in object storage at a relatively low cost, with fit-for-purpose query engines (like Presto and Spark) accessing the data concurrently. This allows for offloading existing data from a client's enterprise data warehouse (EDW), where the performance requirements and/or frequency with which the data is accessed don't justify the costs of having that data in the warehouse (keep in mind that costs aren't limited to the data storage itself; there are costs in preparing and moving data into the warehouse, additional storage costs for larger backup images, the impact of running relatively low priority workloads at the same time as higher priority workloads, and so on). Additionally, with IBM Db2 and Netezza's tight integration with watsonx.data, data that needs to live in a data warehouse can do so, while the rest of the data lives in watsonx.data. Db2 and Netezza can access the data in object storage directly (due in part to the shared metadata store) and queries can combine data in the warehouse with the data in the lakehouse. This provides clients with complete flexibility in where they store their data.

## 7.2. Exploring MinIO Object Storage

Watsonx.data Developer Edition includes a local S3-compatible object store called *MinIO*. Rather than using an external S3 object store (which is certainly possible with watsonx.data), this lab uses the local MinIO object store.

You will need MinIO credentials to log into and use the MinIO console. This includes an *access key* (username) and a *secret key* (password). These values are specific to your environment and are stored as environment variables in the Presto container.

1. Open a terminal command window to the watsonx.data server as the **root** user.
2. Run the following command to extract and display the access key (username). Make note of this and keep it in a location you can refer to later.

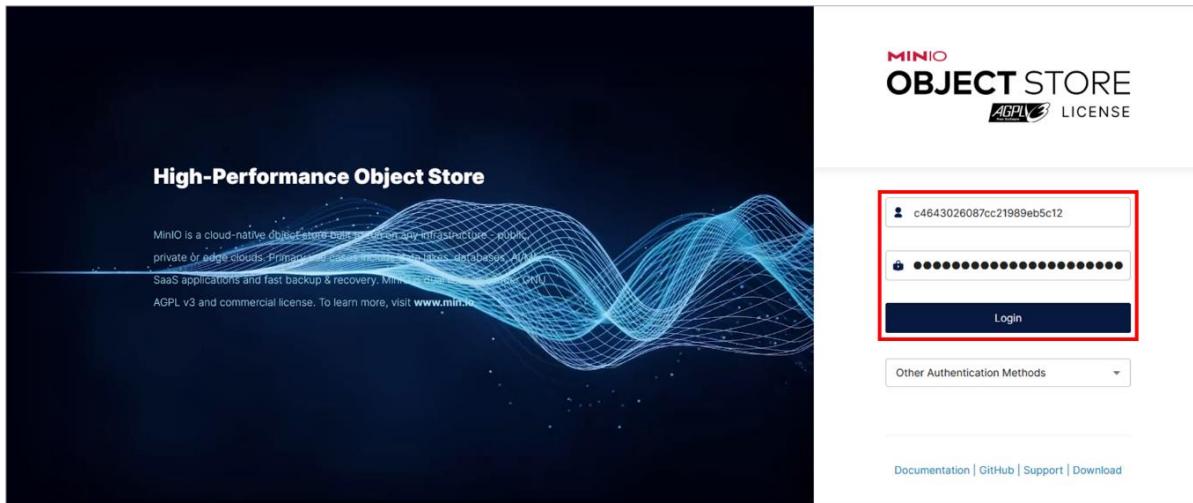
```
docker exec ibm-lh-presto printenv | grep LH_S3_ACCESS_KEY | sed  
's/.*=//'
```

3. Run the following command to extract and display the secret key (password). Make note of this and keep it in a location you can refer to later.

```
docker exec ibm-lh-presto printenv | grep LH_S3_SECRET_KEY | sed  
's/.*=//'
```

4. Open the MinIO console in a new browser window (<http://192.168.252.2:9001>). You may be warned about a potential security risk, in which case you can proceed as you did with the watsonx.data console earlier.

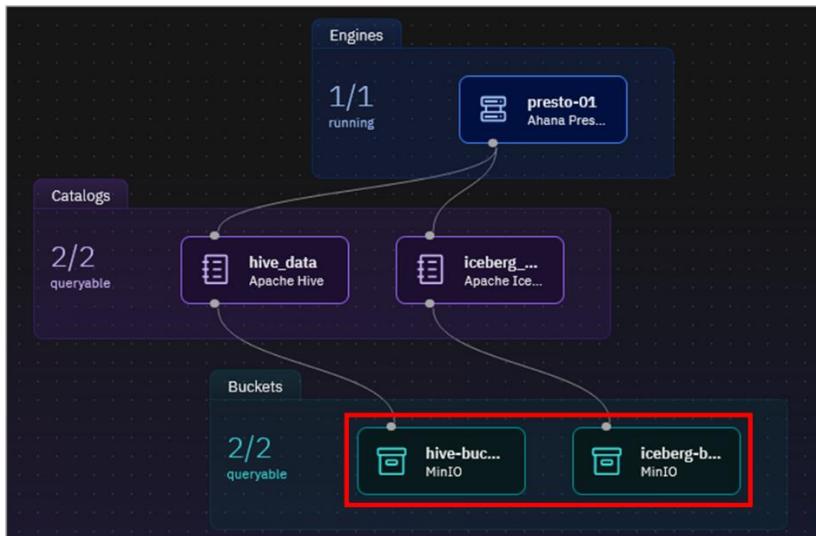
- Refer to the MinIO credentials you copied above and enter the **access key** for the **Username** and the **secret key** for the **Password**. Click the **Login** button.



This opens the **Object Browser** screen. There are two buckets that exist by default: **hive-bucket** and **iceberg-bucket**. There are a number of objects within these buckets already – the pre-populated tables in the hive-bucket and the tables you've created in the iceberg-bucket.

Name	Objects	Size	Access
hive-bucket	70	42.3 MiB	R/W
iceberg-bucket	16	49.4 KiB	R/W

- Refer back to the **Infrastructure manager** screen in the **watsonx.data console** and note how these two buckets were previously registered with watsonx.data.



All tables created in the **hive\_data** catalog have their files (objects) in the **hive-bucket** bucket. Likewise, all tables created in the **iceberg\_data** catalog have their files (objects) in the **iceberg-bucket** bucket.

- Go back to the MinIO console. Click the row for the **iceberg-bucket** bucket.

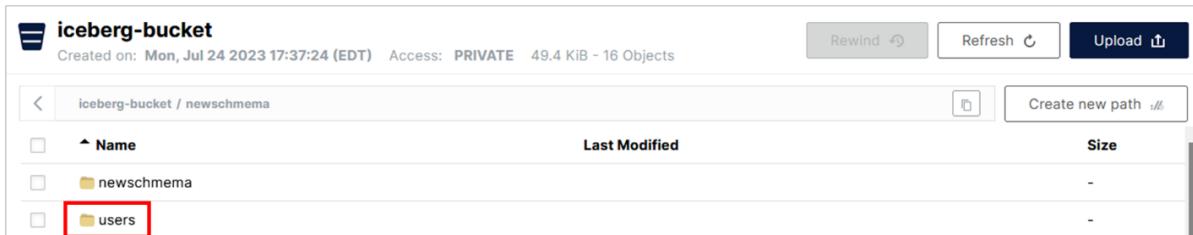
Name	Objects	Size	Access
hive-bucket	70	42.3 MiB	R/W
iceberg-bucket	15	49.3 KiB	R/W

There are two folders listed, one for each schema that has been created in the catalog associated with this bucket (**iceberg\_data**). Watsonx.data uses the schema name for the folder when you create it (such as when you created the schema **my\_schema** earlier in the lab). Recall that when you created the schema **newschema** using the Presto CLI, you specified the location as '`s3a://iceberg-bucket/newschema`', which matches what you see here.

The screenshot shows the MinIO console interface for the 'iceberg-bucket'. The bucket was created on Monday, July 24, 2023, at 17:37:24 (EDT) with PRIVATE access. It contains 16 objects with a total size of 49.4 KiB. The bucket list view shows two subfolders: 'my\_schema' and 'newschmema'. Both of these folder names are highlighted with a red border.

- Click the row for the **newschema** folder.

In the **newschema** folder is a sub-folder called **users**. When you created the **users** table in the Presto CLI earlier, it created this sub-folder with the same name as the table.

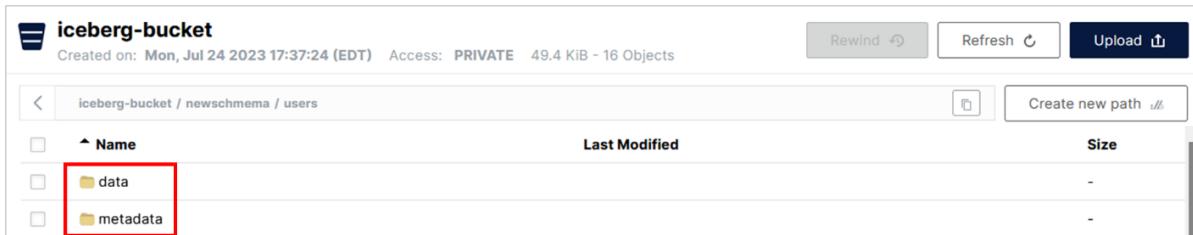


The screenshot shows the MinIO console interface. At the top, it displays the bucket name "iceberg-bucket" and some metadata: "Created on: Mon, Jul 24 2023 17:37:24 (EDT)", "Access: PRIVATE", and "49.4 KiB ~ 16 Objects". On the right, there are buttons for "Rewind", "Refresh", and "Upload". Below this is a breadcrumb navigation bar showing the path: "iceberg-bucket / newschmema". To the right of the path are icons for "Create new path" and a trash bin. The main area is a table listing folder contents. The columns are "Name", "Last Modified", and "Size". There are three entries: "newschmema" (with a minus sign in Size), "users" (with a minus sign in Size), and another "users" entry which is highlighted with a red box. All other rows have a checkbox column to their left.

Name	Last Modified	Size
newschmema	-	-
users	-	-
users	-	-

- Click the row for the **users** folder.

There are two sub-folders within the table's folder. The **data** folder contains the Parquet file(s) holding the actual data for the table. The **metadata** folder contains a series of metadata files (of varying data file formats) used by Iceberg.



The screenshot shows the MinIO console interface. The path in the breadcrumb navigation bar is now "iceberg-bucket / newschmema / users". The main area is a table listing folder contents. The columns are "Name", "Last Modified", and "Size". There are two entries: "data" (highlighted with a red box) and "metadata" (also highlighted with a red box). Both entries have a minus sign in the Size column.

Name	Last Modified	Size
data	-	-
metadata	-	-

- Review the files within the **data** and **metadata** folders. You will see a mix of Parquet and Avro files.

To return back up the folder hierarchy, use the **breadcrumbs** at the top of the navigation pane. Clicking the < icon brings you to the parent of the folder you are currently in.



The screenshot shows the MinIO console interface. The path in the breadcrumb navigation bar is now "iceberg-bucket / newschmema / users / metadata". The main area is a table listing folder contents. The columns are "Name", "Last Modified", and "Size". There is one entry: "data" (highlighted with a red box). It has a minus sign in the Size column.

Name	Last Modified	Size
data	-	-

- If you are proceeding to the next section (Data Ingest) then keep the MinIO console open. Otherwise, close the browser window.

## 8. Data Ingest

Data ingestion is the process of importing and loading data into watsonx.data. Multiple methods are available to ingest data into watsonx.data, including:

- The **Create table from file** option on the **Data manager** page (as you saw earlier in section [5.3. Data Manager Page](#)).
- The standalone, non-Developer Edition of watsonx.data includes an **Ingestion jobs** tab on the **Data manager** page, where data can be read from a source object storage bucket and ingested into a table. However, as this lab uses the Developer Edition, it is not available to use here.
- Installation of the **ibm-lh** command line tool and using commands to create a job to ingest files from S3-compatible object storage or a local file system.
- Loading the data file(s) for a table into an object storage bucket (if they don't already reside in object storage), registering the bucket with watsonx.data, and creating a table on top of the file(s) using SQL.

In this section you will use the MinIO console to upload a Parquet data file into object storage and then you will create a table on it using Presto.

1. Download the **aircraft.parquet** file from here: <https://ibm.box.com/v/ontime-aircraft-id>.
2. If you don't already have the MinIO console open, open it in a new browser window (<http://192.168.252.2:9001>). Use the credentials (access key and secret key) you made note of earlier.

Because you are uploading a data file that is not “wrapped” in the Iceberg table format, it needs to use a Hive catalog. You will use the existing **hive\_data** catalog which is associated with the **hive-bucket** bucket.

3. Click the row for the **hive-bucket** bucket.

Name	Objects	Size	Access
hive-bucket	70	42.3 MiB	R/W
iceberg-bucket	16	49.4 KiB	R/W

- In the **hive-bucket** panel, click the **Create new path** button.

The screenshot shows the 'hive-bucket' panel. At the top, there are buttons for 'Rewind', 'Refresh', and 'Upload'. Below that is a table with columns: 'Name', 'Last Modified', and 'Size'. There are four entries: 'gosalesdw', 'ontime', and 'taxi', each with a small folder icon. To the right of the table is a red box highlighting the 'Create new path' button, which has a plus sign icon and the text 'Create new path'.

- In the **Choose or create a new path** pop-up window, enter **myschema2/aircraft** for the **New Folder Path**. Then, click the **Create** button.

The screenshot shows a modal dialog titled ':// Choose or create a new path'. It has a 'Current Path:' label with the value '/hive-bucket'. Below it is a 'New Folder Path\*' input field containing 'myschema2/aircraft', which is also highlighted with a red box. At the bottom are two buttons: 'Clear' and 'Create', with 'Create' being highlighted by a red box.

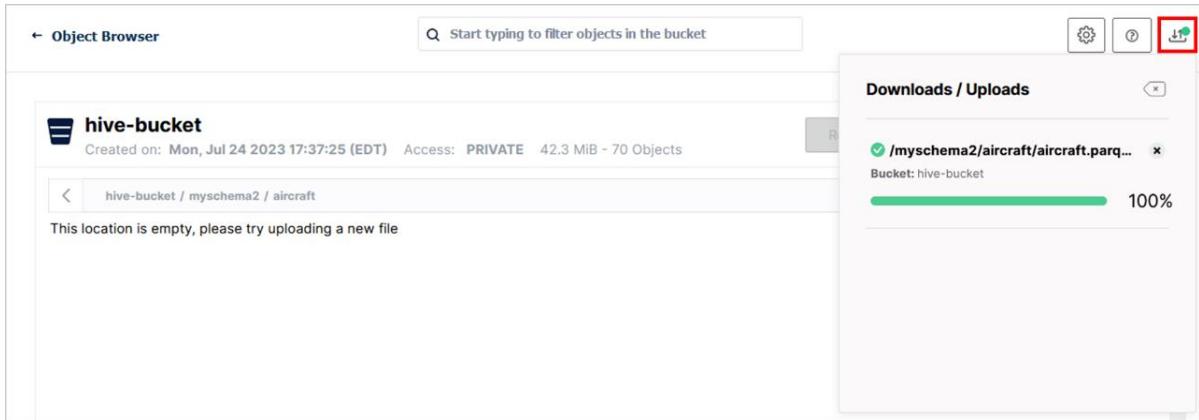
This is going to create nested folders under the current bucket. The top-level folder (**myschema2**) should match the name of the schema you are going to create later, and the next folder down (**aircraft**) should match the name of the table you are going to create later.

- You are instructed to upload a file. Click the **Upload** button in the upper-right and then click **Upload File**.

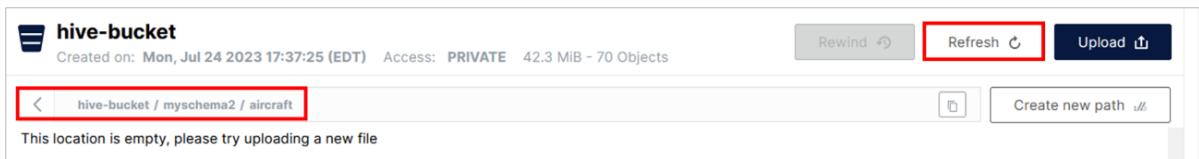
The screenshot shows the 'hive-bucket' panel again. The 'Upload' button at the top right is highlighted with a red box. Below it, a context menu is open with options: 'Upload File' (highlighted with a red box) and 'Upload Folder'.

**Note:** You are also given the option of uploading a folder. If you already have a set of files in folders organized by schema and table, you could upload the entire folder (in which case you wouldn't have to create a new path for the table as you did above, as the path is already reflected in the folder being uploaded).

- Locate, select, and upload the **aircraft.parquet** file you downloaded earlier.
- Click the **Downloads/Upserts** icon to close the **Downloads/Upserts** panel.



- If the folder appears empty, click the **Refresh** button in the upper-right. If the folder is still showing as empty, navigate up one level to the parent folder (**myschema2**) and then navigate back down to this folder (**aircraft**). If this still doesn't work, refresh the browser window (for instance, using **<F5>** in the Firefox browser).



The file should now be listed, as in the image below.

Name	Last Modified	Size
aircraft.parquet	Today, 18:49	129.0 KIB

- Close the MinIO console browser window as you no longer need to use it.

You will now create a schema and a table to match the file that you uploaded. You will use the **Presto CLI**, but alternatively you could also use the **Query workspace** page in the watsonx.data web interface.

- Open a terminal command window to the watsonx.data server as the **root** user.

12. Run the following command to open a Presto CLI interactive terminal.

```
/root/ibm-lh-dev/bin/presto-cli
```

13. Run the following SQL statement to create the **myschema2** schema (based on the catalog/bucket being used and the schema folder name you specified when uploading the file to object storage earlier; the schema name doesn't have to match, but it's easier to manage this way).

```
create schema hive_data.myschema2
with (location = 's3a://hive-bucket/myschema2');
```

14. Run the following SQL statement to create the **aircraft** table.

```
create table hive_data.myschema2.aircraft
(tail_number varchar,
 manufacturer varchar,
 model varchar)
with (format = 'Parquet',
      external_location='s3a://hive-bucket/myschema2/aircraft');
```

15. Validate that the **aircraft** table has been created correctly by querying the number of rows in it.

```
select count(*) from hive_data.myschema2.aircraft;
```

There should be 13,101 rows returned. Your table just read the data out of the file you uploaded into object storage!

16. Quit from the Presto CLI by running the following command.

```
quit;
```

17. Open the watsonx.data user interface in a browser window (if you don't already have it open).

18. Select the **Data manager** (grid) icon from the left-side menu.

19. Expand the **hive\_data** catalog to see the new schema you created (**myschema2**).

The screenshot shows the watsonx.data user interface. At the top, it says "Engine" and "presto-01". Below that is a search bar with "Search for loaded tables" and a "Create" button. Under "Catalogs associated", there is a dropdown menu for "hive\_data". Inside this dropdown, several schemas are listed: "gosalesdw 64", "myschema2" (which is highlighted with a red box), "ontime", "taxi", and "iceberg\_data".

**Note:** If you don't see the schema listed then hover your mouse pointer over the far right of the line for the **hive\_data** catalog until you see the **Refresh** icon appear. Click the **Refresh** icon. You should now see the schema listed as in the above image.

This screenshot is similar to the previous one, showing the watsonx.data interface. The "hive\_data" catalog is expanded, and the "refresh" icon next to it is highlighted with a red box. This indicates that the schema has been updated after a refresh.

20. Expand the **myschema2** schema to see the new table you created (**aircraft**).

This screenshot shows the watsonx.data interface. The "hive\_data" catalog is expanded, and its "myschema2" schema is also expanded. Inside "myschema2", there is a single table named "aircraft", which is highlighted with a red box.

The table you just created through the Presto CLI – which is based on a data file you uploaded into object storage – is visible through the watsonx.data user interface. This is one of the

benefits of having a shared metastore. In the future, any query engines that get associated with the `hive_data` catalog would also be able to work with this table.

In practice, a next step might be to convert this table to the Iceberg table format, so that it benefits from all the capabilities provided by Iceberg (such as transactional consistency, schema and partition evolution, snapshots for time travel queries and rollback, and improved query efficiency). You will see how easy it is to create a copy of a table using `CREATE TABLE AS SELECT` (CTAS) in section [\*10. Offloading Data from a Data Warehouse\*](#).

## 9. Federated Queries

Unlike traditional database systems, Presto doesn't have its own native database storage. Instead, Presto supports separation of compute and storage, with dozens of connectors that let Presto access data where it lives – which could be in relational databases, NoSQL databases, data warehouses, data lakes, data lakehouses, and more.

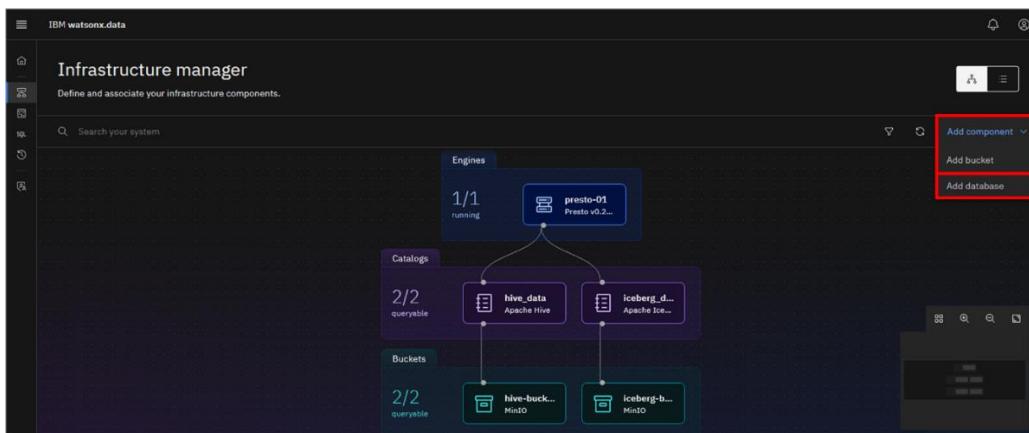
With Presto's *federated query* capability (frequently referred to as *data federation*, or simply *federation*), enterprises can combine data from their existing, diverse sources with new data that they have in watsonx.data, rapidly unlocking insights without the complexity and cost of duplicating or moving data.

Although Presto supports a wide range of connectors, watsonx.data officially only supports a subset of them. This is because IBM wants to ensure quality, performance, and security of connectors before adding support (which may require updating connector source code to do this). More connectors will be added over time.

As of 3Q 2023, Presto in watsonx.data can currently connect to IBM Db2, Netezza, Apache Kafka, MySQL, PostgreSQL, and MongoDB, among others. The most current list of connectors and the types SQL statements supported can be found [here](#). The list of supported connectors will grow over time.

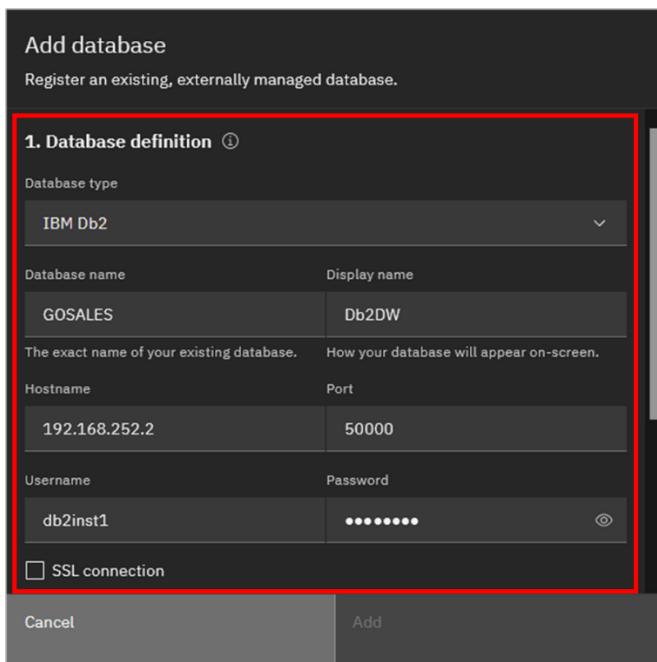
In this section you will combine data from watsonx.data's object storage with data in Db2 and PostgreSQL databases. To avoid you having to provision these databases yourself, they've been installed in the same VM as watsonx.data and are pre-populated with data.

1. Select the **Infrastructure manager** icon (粤港澳) from the left-side menu.
2. Click the **Add component** dropdown menu at the right-side of the screen. Select **Add Database**.



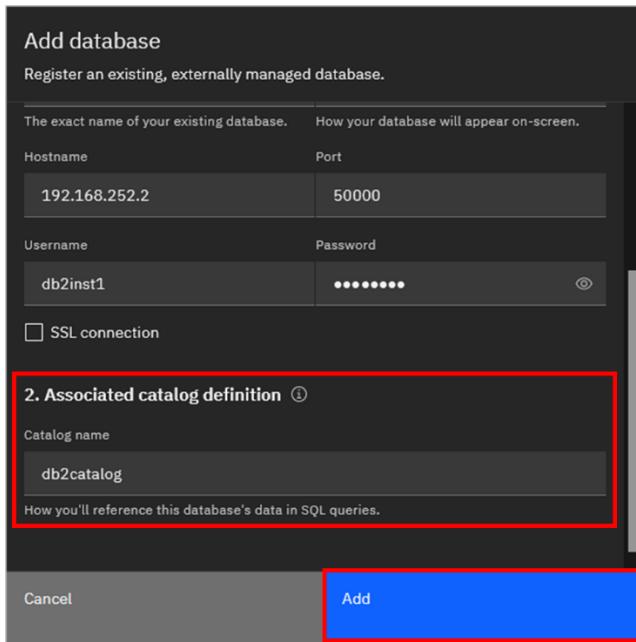
3. In the **Add database** pop-up window, select/enter the following pieces of information in the **1. Database definition** section:

- **Database type:** IBM Db2
- **Database name:** GOSALES
- **Display name:** Db2DW
- **Hostname:** 192.168.252.2
- **Port:** 50000
- **Username:** db2inst1
- **Password:** db2inst1
- **SSL connection:** <Uncheck>

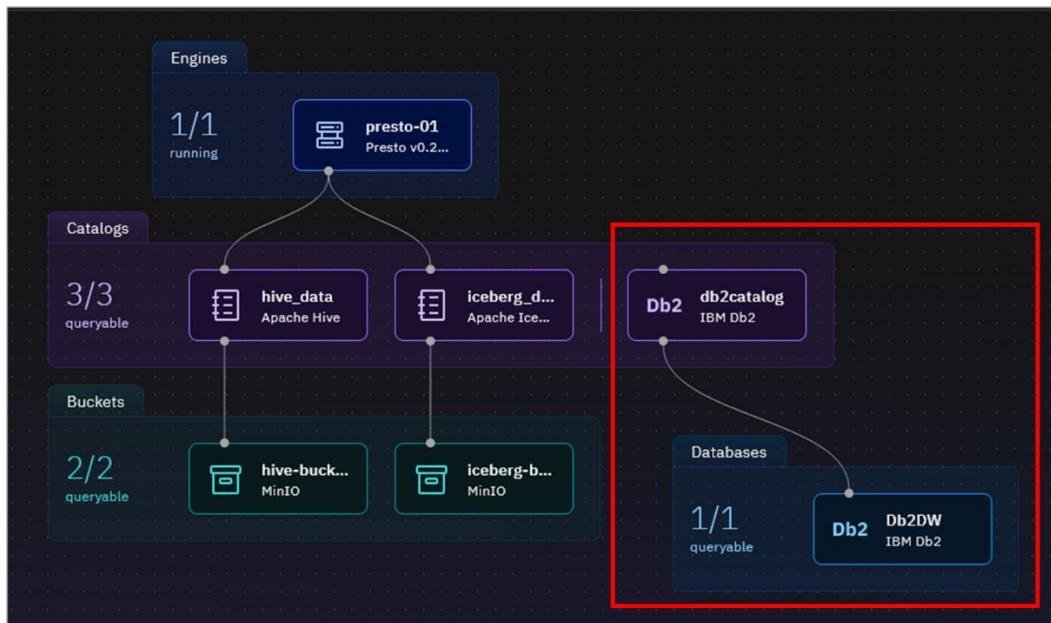


When a new bucket or database is added to watsonx.data, a new catalog is created as well.

4. Scroll down to section **2. Associated catalog definition**. Enter **db2catalog** for the **Catalog name**. Then, click **Add**.

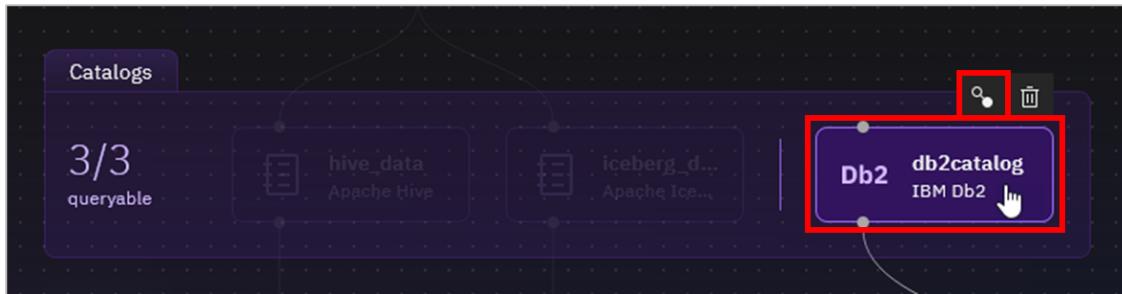


The **Db2DW** database and **db2catalog** catalog have been added to **watsonx.data** and are now reflected in the topology view of the infrastructure components.

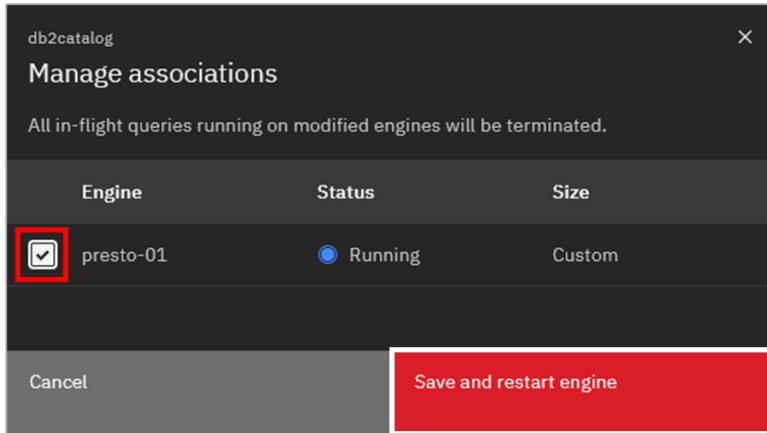


The **db2catalog** catalog is automatically associated with the **Db2DW** database, but to be able to query data from this database, the **db2catalog** catalog must also be associated with the **presto-01** engine (note how there currently is no line between **db2catalog** and **presto-01**).

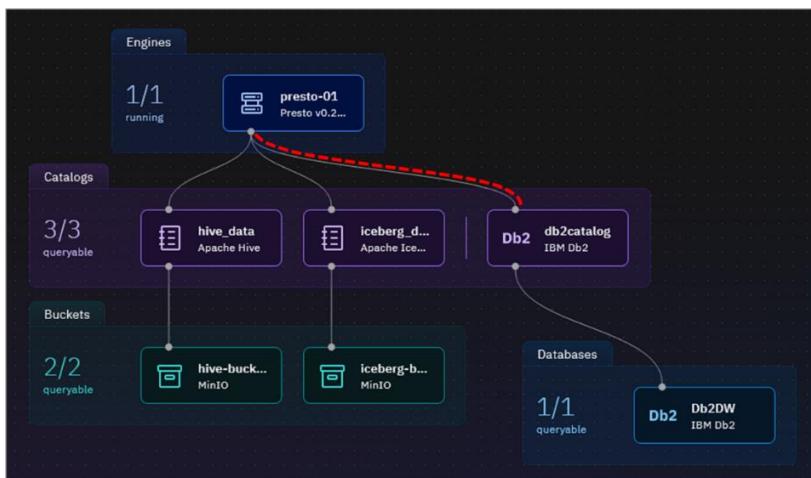
5. Hover your mouse pointer over the **db2catalog** catalog tile and the **Manage associations** icon will appear. Click the **Manage associations** icon.



6. In the **Manage associations** pop-up window, select the checkbox for the **presto-01** engine and then click **Save and restart engine**.



A line now connects **presto-01** with **db2catalog**, indicating that they're associated.



Next you will add the PostgreSQL database. The password for the PostgreSQL admin account is specific to your environment and the following steps will extract it for you.

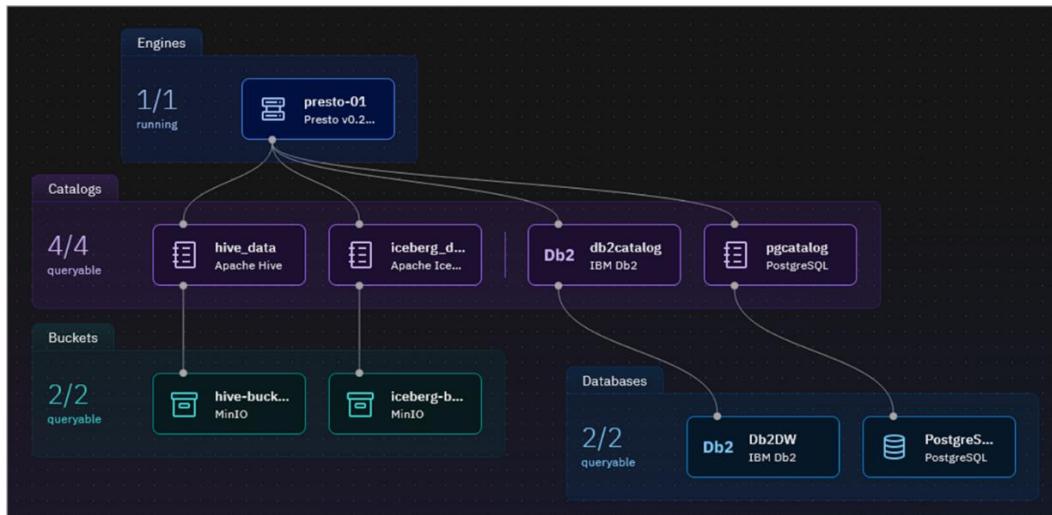
7. Open a terminal command window to the watsonx.data server as the **root** user.
8. The administrative user for the PostgreSQL database is **admin**. Run the following command to extract and display the **password**. Copy the value shown to a location you can refer to later.

```
docker exec ibm-lh-postgres printenv | grep POSTGRES_PASSWORD | sed  
's/.*=//'
```

9. Repeat Steps 2-6 from above to add the PostgreSQL database to watsonx.data. Use the following information:

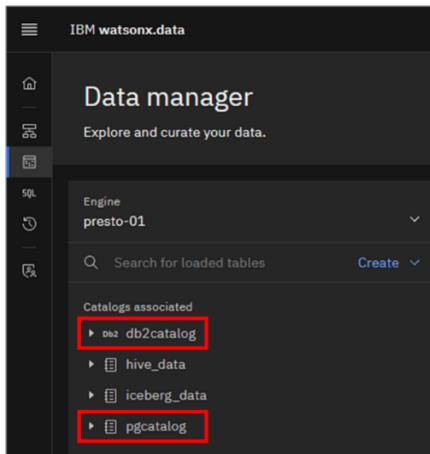
- **Database type:** PostgreSQL (it can be found under Other Databases)
- **Database name:** gosales
- **Display name:** PostgreSQLDB
- **Hostname:** 192.168.252.2
- **Port:** 5432
- **Username:** admin
- **Password:** <The password generated in the previous step>
- **Catalog name:** pgcatalog

With both the Db2 and PostgreSQL databases added, the topology should look like the image below.



10. Select the **Data manager** icon (grid) from the left-side menu.

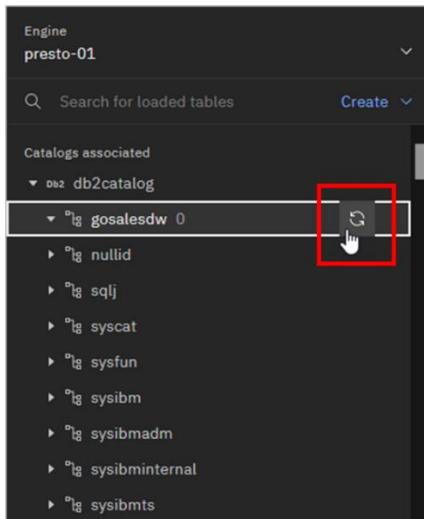
11. Expand both the **db2catalog** and **pgcatalog** catalogs.



Schema and table information are shown for both of these databases. Both include an identical **gosalesdw** schema with tables containing sales data for the fictional Great Outdoors company. Remember that there is also a copy of this same data in object storage, managed by the **hive\_data** catalog.

12. If you do not see any tables associated with the **gosalesdw** schema in either of the catalogs, hover your mouse pointer at the far right of the line for the catalog until you see the **Refresh** icon appear. When you see the icon, click it. Likewise, if you didn't actually see the **gosalesdw** schema itself under the catalog, refresh the catalog in the same way.

**Note:** If for some reason the list of tables does not refresh, you can still proceed. The table information is in the catalog and the tables can be queried.



**Note:** The number to the right of the schema name indicates the number of tables in the schema that the catalog is aware of (and if you aren't currently seeing any tables listed in one of these schemas, refreshing it should increase the number of tables from zero to a non-zero value).

You now have copies of the Great Outdoors company sales tables in object storage (hive\_data), Db2 (db2catalog), and PostgreSQL (pgcatalog). This isn't something you'd have in a real-world scenario (after all, the benefit of being able to federate access to multiple data sources is to avoid data duplication). However, it's being done here for the purposes of highlighting Presto's federation capabilities.

You will now see how you can run a federated query that combines data from all three of these data sources.

13. Select the **Query workspace** () icon from the left-side menu.

14. Copy and paste the query below into the **SQL worksheet**. Click **Run on presto-01**.

This sample query could be used by the fictional business to determine which purchasing method is associated with the largest orders. The query accesses five tables, one of which is in PostgreSQL (yellow), two are in Db2 (green), and two are in watsonx.data's object storage (pink).

```
select pll.product_line_en as product,
       md.order_method_en as order_method,
       sum(sf.quantity) as total
  from
    pgcatalog.gosalesdw.sls_order_method_dim as md,
    db2catalog.gosalesdw.sls_product_dim as pd,
    db2catalog.gosalesdw.sls_product_line_lookup as pll,
    hive_data.gosalesdw.sls_product_brand_lookup as pbl,
    hive_data.gosalesdw.sls_sales_fact as sf
 where
    pd.product_key = sf.product_key
    and md.order_method_key = sf.order_method_key
    and pll.product_line_code = pd.product_line_code
    and pbl.product_brand_code = pd.product_brand_code
 group by pll.product_line_en, md.order_method_en
 order by product, order_method;
```

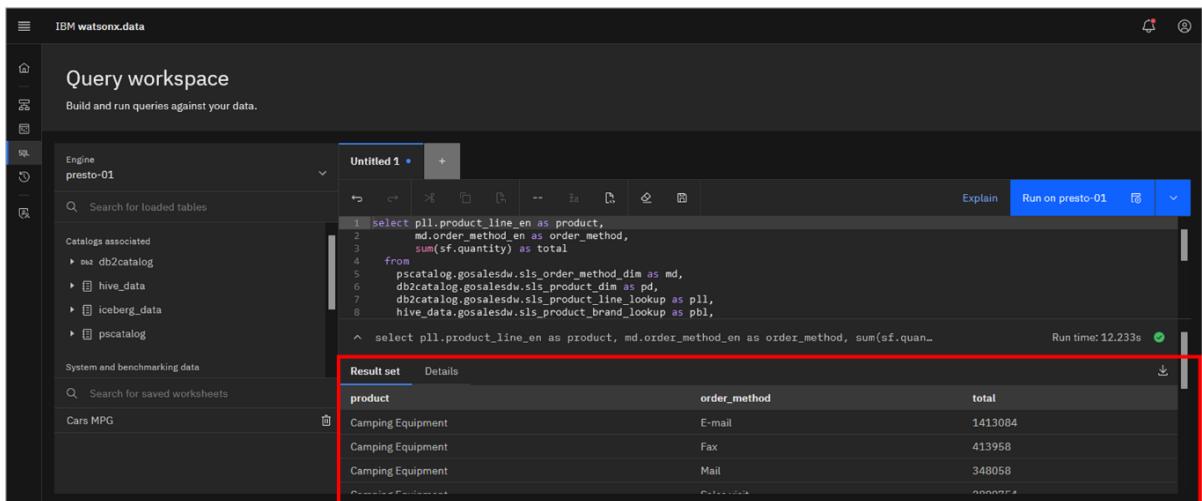
**Note:** If the SQL statement pastes into the worksheet as a single line, you can nicely format it by clicking the **Format worksheet** icon.



```
Untitled 1 • + ↵ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ Explain Run on presto-01 ⌂ ⌂
```

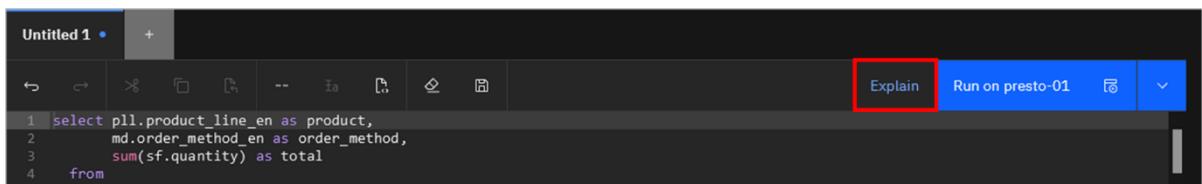
```
1 select pll.product_line_en as product, md.order_method_en as order_method, sum(sf.quantity) as total from pgcatalog.gosalesdw.sls_order
```

The **Result set** for the query is found at the bottom of the screen.



product	order_method	total
Camping Equipment	E-mail	1413084
Camping Equipment	Fax	413958
Camping Equipment	Mail	348058
Camping Equipment	Other	2000000

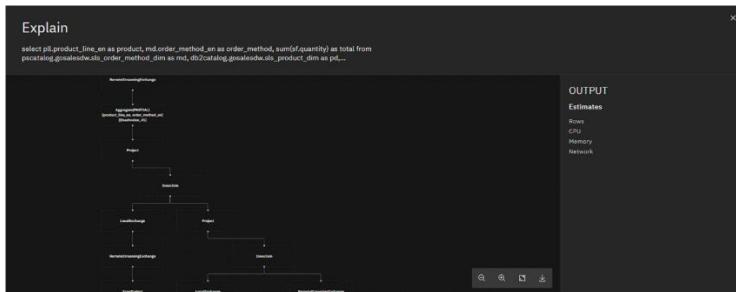
15. Click the **Explain** button in the menu bar of the worksheet.



```
Untitled 1 • + ↵ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ Explain Run on presto-01 ⌂ ⌂
```

```
1 select pll.product_line_en as product,
2     md.order_method_en as order_method,
3     sum(sf.quantity) as total
4   from
```

The visual explain output for this query looks a lot more interesting than the one shown earlier. If you scroll through the visual explain output, you'll see five **ScanProject** leaf nodes in the tree. These correspond to the five tables being read.



16. Click the X in the upper-right corner of the Explain window to close it.

**Optional step #1:** Here are two other queries you can try running as well, which combine data from the same three data sources.

1. The following query displays all Canadian and Mexican employees, along with their region and country. This is the kind of query that a reporting tool might generate, based on input from the user.

```
select distinct branch_region_dim.region_en region,
               branch_region_dim.country_en country,
               emp_employee_dim.employee_name employee
        from hive_data.gosalesdw.go_region_dim branch_region_dim,
             pgcatalog.gosalesdw.emp_employee_dim emp_employee_dim,
             db2catalog.gosalesdw.go_branch_dim go_branch_dim
       where branch_region_dim.country_en in ('Canada', 'Mexico') and
             branch_region_dim.country_code = go_branch_dim.country_code and
             emp_employee_dim.branch_code = go_branch_dim.branch_code
    order by region, country, employee;
```

2. In many businesses, departments (or organizations, as they're called in this dataset) are hierarchical, in that a department falls under another department, which in turn falls under another one. This query displays the two parent departments for a given set of departments. As in the previous query, this is the kind of query that a reporting tool might generate.

```
select gosalesdw.go_org_dim.organization_key,
       go_org_dim_1.organization_parent as org_level1_code,
       go_org_name_lookup_1.organization_name_en as org_level1_name,
       gosalesdw.go_org_dim.organization_parent as org_level2_code,
       go_org_name_lookup_2.organization_name_en as org_level2_name,
       gosalesdw.go_org_dim.organization_code as org_code,
       gosalesdw.go_org_name_lookup.organization_name_en as org_name
  from pgcatalog.gosalesdw.go_org_name_lookup go_org_name_lookup_2
    inner join
      hive_data.gosalesdw.go_org_dim
        inner join
          pgcatalog.gosalesdw.go_org_name_lookup
            on hive_data.gosalesdw.go_org_dim.organization_code =
               pgcatalog.gosalesdw.go_org_name_lookup.organization_code
            on go_org_name_lookup_2.organization_code =
               hive_data.gosalesdw.go_org_dim.organization_parent
              inner join
                pgcatalog.gosalesdw.go_org_name_lookup go_org_name_lookup_1
                  inner join
                    hive_data.gosalesdw.go_org_dim go_org_dim_1
                      on go_org_name_lookup_1.organization_code =
                         go_org_dim_1.organization_parent
                      on hive_data.gosalesdw.go_org_dim.organization_parent =
                         go_org_dim_1.organization_code
            where (hive_data.gosalesdw.go_org_dim.organization_code
                   between '1700' and '5730')
              order by org_code;
```

**Optional step #2:** Try running the two queries above from within the Presto CLI as well. You should get the same results in the Presto CLI as you did in the Query workspace.

## 10. Offloading Data from a Data Warehouse

The previous section described the federated query capability of watsonx.data and Presto. In addition to it making it easy to access data across the enterprise, federation also opens the door to significant cost-savings for clients.

Many enterprises maintain expensive data warehouses, whether they are on-premises or in the cloud. These warehouses support mission-critical workloads with low-latency and high-performance requirements that justify their high costs. However, there is also often data in these warehouses that are supporting less important or less stringent workloads. This could be infrequently accessed (cold) data being kept for audit purposes, or data used for business intelligence, reporting, and data science. It makes financial sense for this data to be offloaded to a lower-cost solution, but keeping it in the warehouse has just been the easier path to take. The result, though, is periodic scaling of the data warehouse to support growing workloads, which can be very expensive.

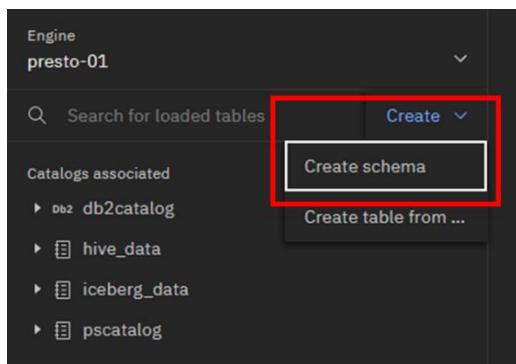
With watsonx.data, enterprises can offload data that really doesn't need to be in the warehouse to lower-cost object storage. With fit-for-purpose engines, data warehousing costs can be reduced by offloading workloads from a data warehouse to watsonx.data. Specifically, applications that need to access this data can query it through Presto (or Spark). This includes being able to combine the offloaded data with the data that remains in the warehouse. This section will show an example of how this can be done with Db2 (but the steps are equivalent for other data warehouse products).

Additionally, external engines that support the Iceberg open table format can also work directly with data in watsonx.data's object storage. For example, Db2 and Netezza have been enhanced to read from and write to the Iceberg table format (not all deployment options for Db2 and Netezza currently include this support; as of 3Q 2023 it is limited to Db2 Warehouse Gen3 on AWS and Netezza Performance Server as a Service on AWS). This means that Db2 and Netezza can participate in the watsonx.data ecosystem as well, accessing the lakehouse data in object storage directly (just as Presto and Spark can). This, however, is beyond the scope of this lab.

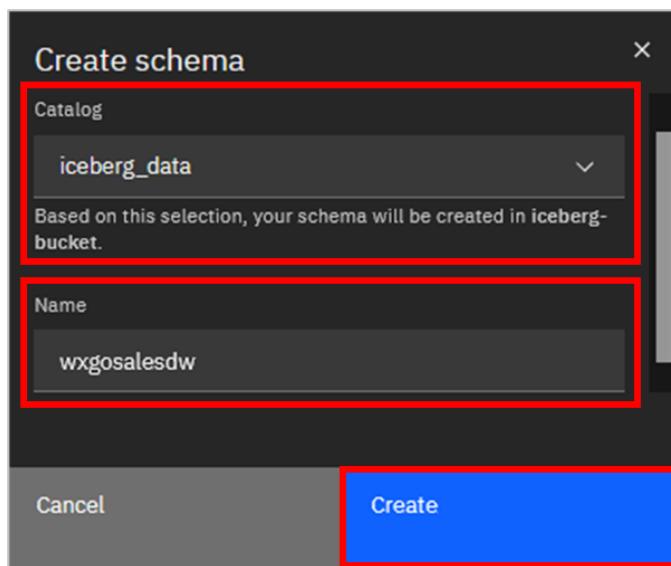
In the previous section you registered an existing Db2 environment with watsonx.data (calling it **Db2DW**) and created a catalog for it called **db2catalog**.

In this example scenario, you're going to "move" the **gosalesdw.sls\_sales\_fact** table from Db2 to watsonx.data's object storage. It will be created as an Iceberg table, in a new schema you create called **wxgosalesdw**, managed by the **iceberg\_data** catalog.

1. Select the **Data manager** (grid icon) icon from the left-side menu.
2. Go to the top of the navigation pane and click the **Create** dropdown menu. Select **Create schema**. (Note: These steps are using the web interface, but you can also create a schema through SQL.)



3. In the **Create schema** pop-up window, select/enter the following information, and then click the **Create** button:
  - **Catalog:** iceberg\_data
  - **Name:** wxgosalesdw



4. Expand the **iceberg\_data** catalog. The new schema should be listed.

The screenshot shows the 'Catalogs associated' section of the Presto UI. A dropdown menu at the top is set to 'presto-01'. Below it is a search bar and a 'Create' button. The 'Catalogs associated' list includes 'db2 db2catalog', 'hive\_data' (which is expanded), 'iceberg\_data' (which is expanded and highlighted with a red box), 'my\_schema\_1' (under iceberg\_data), 'wxgosalesdw' (under iceberg\_data), and 'pscatalog'.

5. Select the **Query workspace** ( icon) from the left-side menu.

To create a new table with the same table definition as the original table, you can use the `CREATE TABLE AS SELECT` (CTAS) SQL statement.

6. Copy and paste the following SQL into the **SQL worksheet**. Click **Run on presto-01**.

```
create table iceberg_data.wxgosalesdw.sls_sales_fact  
as select * from db2catalog.gosalesdw.sls_sales_fact;
```

The screenshot shows the SQL worksheet interface. The query `create table iceberg_data.wxgosalesdw.sls_sales_fact as select * from db2catalog.gosalesdw.sls_sales_fact;` is entered in the text area. The 'Run on presto-01' button is highlighted with a blue box.

The result shown at the bottom of the worksheet shows the number of rows that have been inserted from the source table to the new table.

The screenshot shows the results of the SQL query. The output table has two rows: 'Result set' and 'Details'. The 'Result set' row shows 'rows' with the value '446023'. The 'Details' row shows 'Run time: 11.586s' with a green checkmark. The 'rows' value is highlighted with a red box.

In a real-world scenario, you would then remove the table from the data warehouse. However, to keep the gosalesdw sample dataset intact in your environment, you won't do that here.

As a test, you can run a federated query that combines the new table in object storage with existing tables in Db2.

7. Copy and paste the following SQL into the **SQL worksheet**. Click **Run on presto-01**.

```
select pll.product_line_en as product,
       sum(sf.quantity) as total
  from
    db2catalog.gosalesdw.sls_product_dim as pd,
    db2catalog.gosalesdw.sls_product_line_lookup as pll,
    hive_data.gosalesdw.sls_sales_fact as sf
 where
   pd.product_key = sf.product_key
   and pll.product_line_code = pd.product_line_code
 group by pll.product_line_en
 order by product;
```

This business query calculates total sales for each of the high-level product lines. The output should be similar to the image below.

The screenshot shows a SQL worksheet interface. At the top, there is a code editor containing the provided SQL query. To the right of the code editor, the text "Run time: 11.821s" and a green checkmark icon are displayed. Below the code editor is a table titled "Result set". The table has two columns: "product" and "total". The data in the table is as follows:

product	total
Camping Equipment	27301149
Golf Equipment	5113701
Mountaineering Equipment	9900091
Outdoor Protection	12014445
Personal Accessories	34907705

At the bottom left of the table, it says "1–5 of 5 items". On the far right, there are navigation icons for a single page.

## 11. Time Travel and Rollback

The Iceberg open table format provides a number of benefits to users, including the ability to see a table as it existed at a point in the past. This *time travel* capability is useful in a number of different ways. For example, having the ability to query historical data is useful for auditing purposes. Or if an application corrupts table data in some way, you can imagine the value in being able to quickly reverse those changes by resetting the table to a known good state.

Iceberg uses *snapshots* to support this time travel capability. A snapshot represents the state of a table at some point in time. When data is modified in a table, such as inserting, updating, or deleting records, a new snapshot is created. There are maintenance operations that can be used to clean up older snapshots that are no longer needed.

As of 3Q 2023, time travel queries are not yet supported in watsonx.data. However, you can roll back a table to an earlier point in time using snapshots. You will go through that process in this section.

1. Select the **Query workspace** (SQL) icon from the left-side menu.
2. Copy and paste the following SQL statement into the **SQL worksheet**. Click **Run on presto-01**.

```
create table iceberg_data.my_schema.airport_id  
as select * from hive_data.ontime.airport_id;
```

The screenshot shows the WatsonX Query workspace interface. At the top, there is a toolbar with various icons. Below the toolbar, the SQL worksheet contains the following code:

```
1 create table iceberg_data.my_schema.airport_id  
as select * from hive_data.ontime.airport_id;
```

The "Run on presto-01" button is highlighted with a red box. The result set shows the output of the query:

Result set	Details
rows	
6250	

The status bar at the bottom right indicates a run time of 1.159s. The page footer includes copyright information and navigation links.

As shown in the output above after running the query, there are **6,250 rows** in this table.

3. Select the **Data manager** (grid icon) from the left-side menu.
4. Navigate to the **iceberg\_data > my\_schema > airport\_id** table (if you don't see the table, refresh the schema). Then, select the **Time travel** tab.

Snapshot ID	Operation	Total records	Added records	Total deleted files	Committed at
539029806241330...	append	6250	6250	0	2023-07-31T17:54...

This tab lists the snapshots associated with the table (identified by a **Snapshot ID**). At this point there is only one snapshot and there are 6,250 total records (matching what you saw earlier).

**Note:** For reference, you can get snapshot information through SQL as well. For example (and no need to run this now), this will query the snapshots available for the `airport_id` table:

```
select * from iceberg_data.my_schema."airport_id$snapshots"
order by committed_at;
```

5. Select the **Query workspace** (SQL icon) again from the left-side menu.
6. Copy and paste the following SQL statement into the **SQL worksheet** and click **Run on presto-01**.

```
insert into iceberg_data.my_schema.airport_id
values (10000, 'North Pole: Reindeer Field');
```

7. Copy and paste (replacing the SQL already there) the following SQL statement into the **SQL worksheet** and click **Run on presto-01**.

```
select * from iceberg_data.my_schema.airport_id
where code = 10000;
```

You should see the row you inserted in the previous step.

A screenshot of a SQL worksheet interface. At the top, there is a command line with the text: '^ select \* from iceberg\_data.my\_schema.airport\_id where code = 10000'. To the right of the command line, it says 'Run time: 1.231s' with a green checkmark. Below the command line, there are two tabs: 'Result set' and 'Details'. The 'Result set' tab is selected. The results table has two columns: 'code' and 'description'. There is one row: 'code' is 10000 and 'description' is 'North Pole: Reindeer Field'. This row is highlighted with a red border. At the bottom left of the results table, it says '1-1 of 1 items'. At the bottom right, there are navigation icons: a downward arrow, '1 of 1 page', and arrows for navigating between pages.

code	description
10000	North Pole: Reindeer Field

8. Copy and paste (replacing the SQL already there) the following SQL statement into the **SQL worksheet** and click **Run on presto-01**.

```
select count(*) from iceberg_data.my_schema.airport_id;
```

You should see a count of 6,251 rows.

9. Select the **Data manager** (grid icon) again from the left-side menu.

10. As before, navigate to the **iceberg\_data > my\_schema > airport\_id** table. Then, select the **Time travel** tab.

A screenshot of the Data manager interface. On the left, there is a sidebar with a dropdown 'Engine' set to 'presto-01'. Below it is a search bar 'Search for loaded tables' and a 'Create' button. Under 'Catalogs associated', there are entries for 'db2catalog', 'hive\_data', 'iceberg\_data', and 'wxgosalesdw'. Under 'iceberg\_data', there is a expanded section for 'my\_schema 2' containing the 'airport\_id' table, which is also highlighted with a red border. On the right, the main panel shows the 'airport\_id' table details. It says 'Catalog: iceberg\_data | Schema: my\_schema'. Below this, there are tabs: 'Table schema', 'Time travel' (which is selected and highlighted with a red border), 'Data sample', and 'DDL'. Under the 'Time travel' tab, there is a sub-header 'Search for snapshots'. A table lists two snapshots:

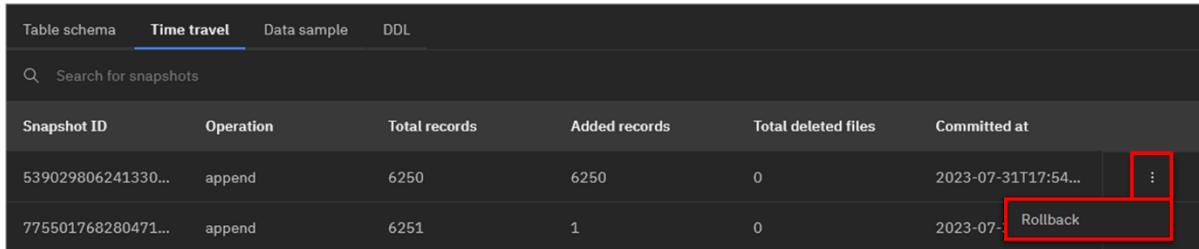
Snapshot ID	Operation	Total records	Added records	Total deleted files	Committed at
539029806241330...	append	6250	6250	0	2023-07-31T17:54...
775501768280471...	append	6251	1	0	2023-07-31T18:09...

Note how there are now two snapshots shown. If you do not see a second snapshot then refresh your browser (for instance, using **<F5>** in the Firefox browser) and repeat this step.

The second snapshot shows that there are 6,251 total rows, with 1 row having been added in this new version of the table.

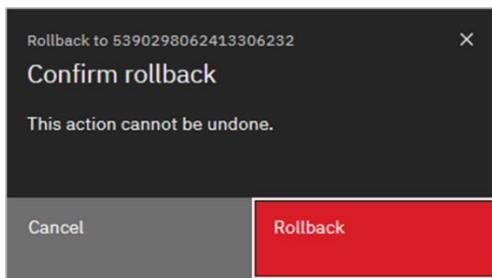
You are now going to roll the table back to the first snapshot, representing the initial state of the table. This version of the table did *not* have the row you added.

11. Click the **overflow menu** icon (vertical ellipses) at the end of the row for the original snapshot (the first snapshot shown, with the earlier **Committed at timestamp** and **6250 Added records**). Click **Rollback**.



Snapshot ID	Operation	Total records	Added records	Total deleted files	Committed at	⋮
539029806241330...	append	6250	6250	0	2023-07-31T17:54...	
775501768280471...	append	6251	1	0	2023-07-	<b>Rollback</b>

12. In the **Confirm rollback** pop-up window, click **Rollback**.



**Note:** For reference, the following SQL statement will perform the equivalent roll back operation (DO NOT RUN THIS NOW):

```
call iceberg_data.system.rollback_to_snapshot
    ('my_schema', 'airport_id', <snapshotID>);
```

13. Select the **Query workspace** () icon again from the left-side menu.

14. Copy and paste the following SQL statement into the **SQL worksheet** and click **Run on presto-01**.

```
select * from iceberg_data.my_schema.airport_id
where code = 10000;
```

The row you added earlier is now gone!

A screenshot of a SQL worksheet titled "Untitled 1". The query entered is:

```
1 select * from iceberg_data.my_schema.airport_id where code = 10000;
```

The result set shows 0 items. The entire result set area is highlighted with a red box.

Result set		Details
0–0 of 0 items		1 of 1 page

15. Copy and paste (replacing the SQL already there) the following SQL statement into the **SQL worksheet** and click **Run on presto-01**.

```
select count(*) from iceberg_data.my_schema.airport_id;
```

A screenshot of a SQL worksheet titled "Untitled 1". The query entered is:

```
1 select count(*) from iceberg_data.my_schema.airport_id;
```

The result set shows 1 item, labeled "\_col0" with the value 6250. The column header is highlighted with a red box.

Result set		Details
_col0		1 of 1 page
6250		

You should see a count of 6,250 rows, which matches the original count when you first created the table. The table is back at the state it was in when it was first created!

## 12. Summary

Congratulations on completing this lab! You gained hands-on experience in the following areas of watsonx.data:

- The watsonx.data web-based user interface, including infrastructure management, data management, running SQL statements, and managing user access
- The Presto web interface and the Presto command line interface (CLI)
- MinIO object storage
- Ingesting data into watsonx.data
- Creating schemas and tables
- Running queries that combine data from multiple data sources
- Offloading tables from Db2 into watsonx.data
- Rolling back a table to a previous point in time

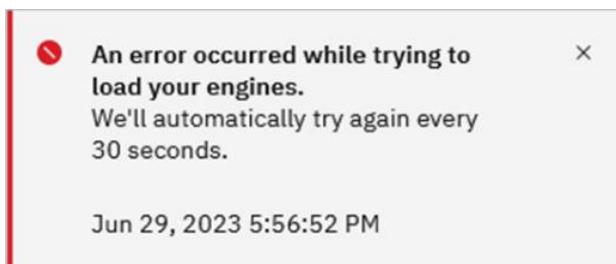
As a result, you should feel much more confident in your ability to demonstrate these capabilities to your clients, as well as be able to discuss the business value of watsonx.data with them.

## Appendix A. Troubleshooting

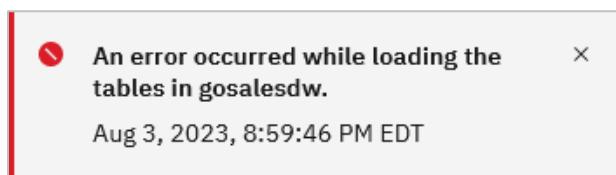
Unexpected errors, while rare, may occur. Resolutions to some of the more common issues encountered by users are included below:

### watsonx.data Web Interface:

1. **The watsonx.data web interface won't start:** If you cannot bring up the watsonx.data web interface (console) in your browser (unable to connect, connection has timed out, and so on) then the environment is likely still starting up. Wait at least 10 minutes and try again. If you are still not able to bring up the web interface after this time, then you can try restarting watsonx.data as described in section [4.7. Stopping and Starting watsonx.data](#).
2. **Engines, catalogs, buckets, or databases cannot be loaded:** If you see console errors stating that watsonx.data's engines, catalogs, buckets, or databases cannot be loaded (similar to the image below), this is likely due to watsonx.data still being started. Wait a minute and refresh the browser window.



3. **Table or schema creation fails:** It is infrequent, but when performing an action within the console, such as creating a schema or table, you might encounter an error. The action may have in fact completed, but if not then repeat the action.
4. **Loading gosalesdw tables fails:** You may see a message appear in the upper-right corner saying "*An error occurred while loading the tables in gosalesdw.*" This occurs when watsonx.data is unable to list the gosales tables under db2catalog in the Data management page. It can be safely ignored; queries that access these tables will still work.



## MinIO Web Interface:

1. **The MinIO web interface won't open in your browser:** If you cannot start the MinIO web interface but the watsonx.data web interface starts fine, then something in the environment is preventing http (non-https) connections from working. Please restart watsonx.data as described in section [4.7. Stopping and Starting watsonx.data](#) (ensure that LH\_RUN\_MODE=diag is set prior to starting watsonx.data).

## Presto Web Interface:

1. **The Presto web interface won't open in your browser:** If you cannot start the Presto web interface but the watsonx.data web interface starts fine, then something in the environment is preventing http (non-https) connections from working. Please restart watsonx.data as described in section [4.7. Stopping and Starting watsonx.data](#) (ensure that LH\_RUN\_MODE=diag is set prior to starting watsonx.data).

## Miscellaneous:

1. **Command/SQL statement fails:** If you are instructed to copy and paste a command or SQL statement from this lab guide and the command/statement fails, it may be because it was copied incorrectly. Try to copy and paste the command/statement again. Alternatively, all of the commands and SQL statements can also be found in this [text document](#). Download this file and copy the text from the file instead.

## Appendix B. Acknowledgements

A big “Thank you!” is owed to George Baklarz, who created the Techzone image used in this lab. There was a tremendous amount of work involved in installing and configuring all the software and populating the various data sources included in the image. George (and his lab material) was also a great source of information on the “how to’s” of watsonx.data and of the lab environment.