



# Get Started with Databricks Data Science & Engineering Workspace

---

Module 01



# Module Objectives

## Get Started with Databricks Data Science and Engineering Workspace

1. Describe the core components of the Databricks Lakehouse Platform.
2. Navigate the Databricks Data Science & Engineering Workspace UI.
3. Create and manage clusters using the Databricks Clusters UI.
4. Develop and run code in multi-cell Databricks notebooks using basic operations.
5. Integrate git support using Databricks Repos.

# Module Overview

## Get Started with Databricks Data Science and Engineering Workspace

### Databricks Workspace and Services

Navigate the Workspace UI

### Compute Resources

DE 1.1 – Create and Manage Interactive Clusters

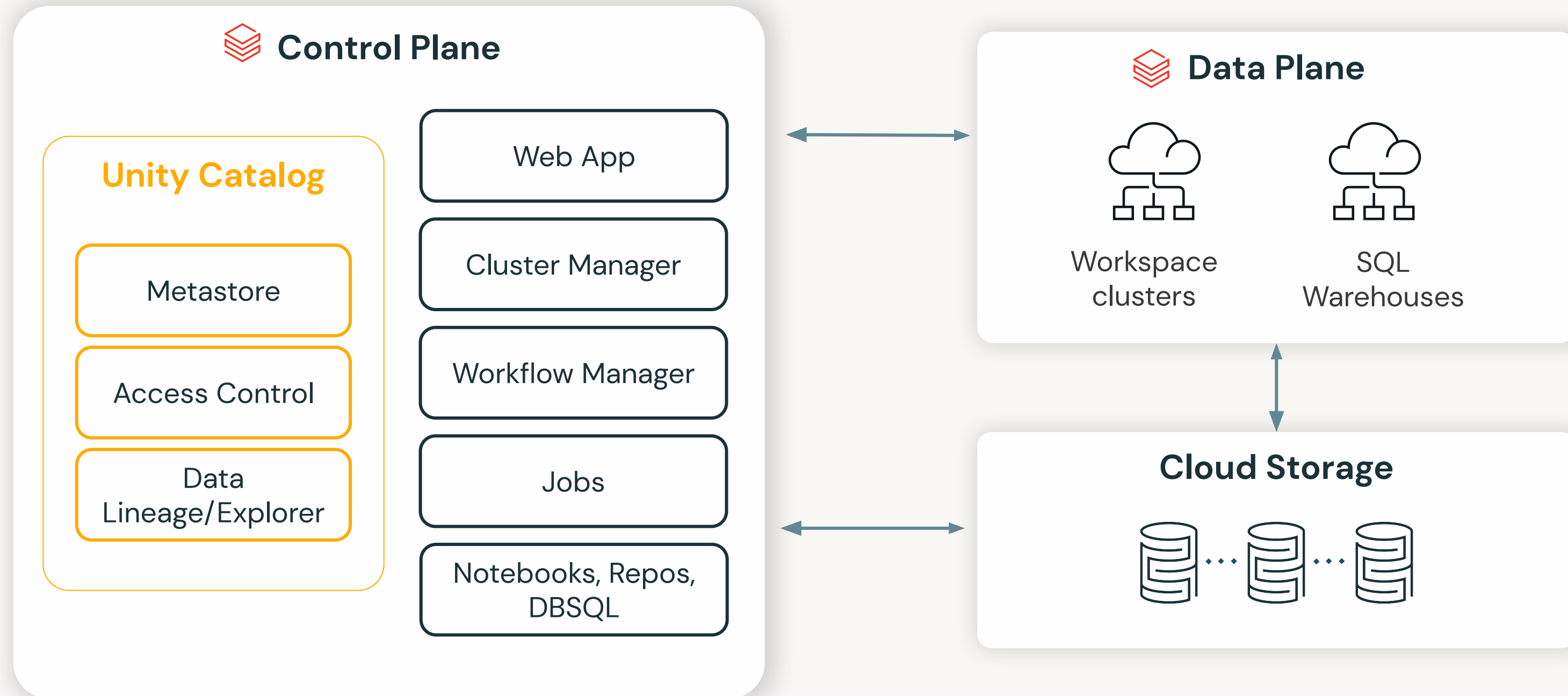
### Develop Code with Notebooks & Repos

DE 1.2 – Databricks Notebook Operations

DE 1.3L – Get Started with the Databricks Lakehouse Platform Lab

# Databricks Workspace and Services

# Databricks Workspace and Services



# Demo: Navigate the Workspace UI

# Compute Resources

# Clusters

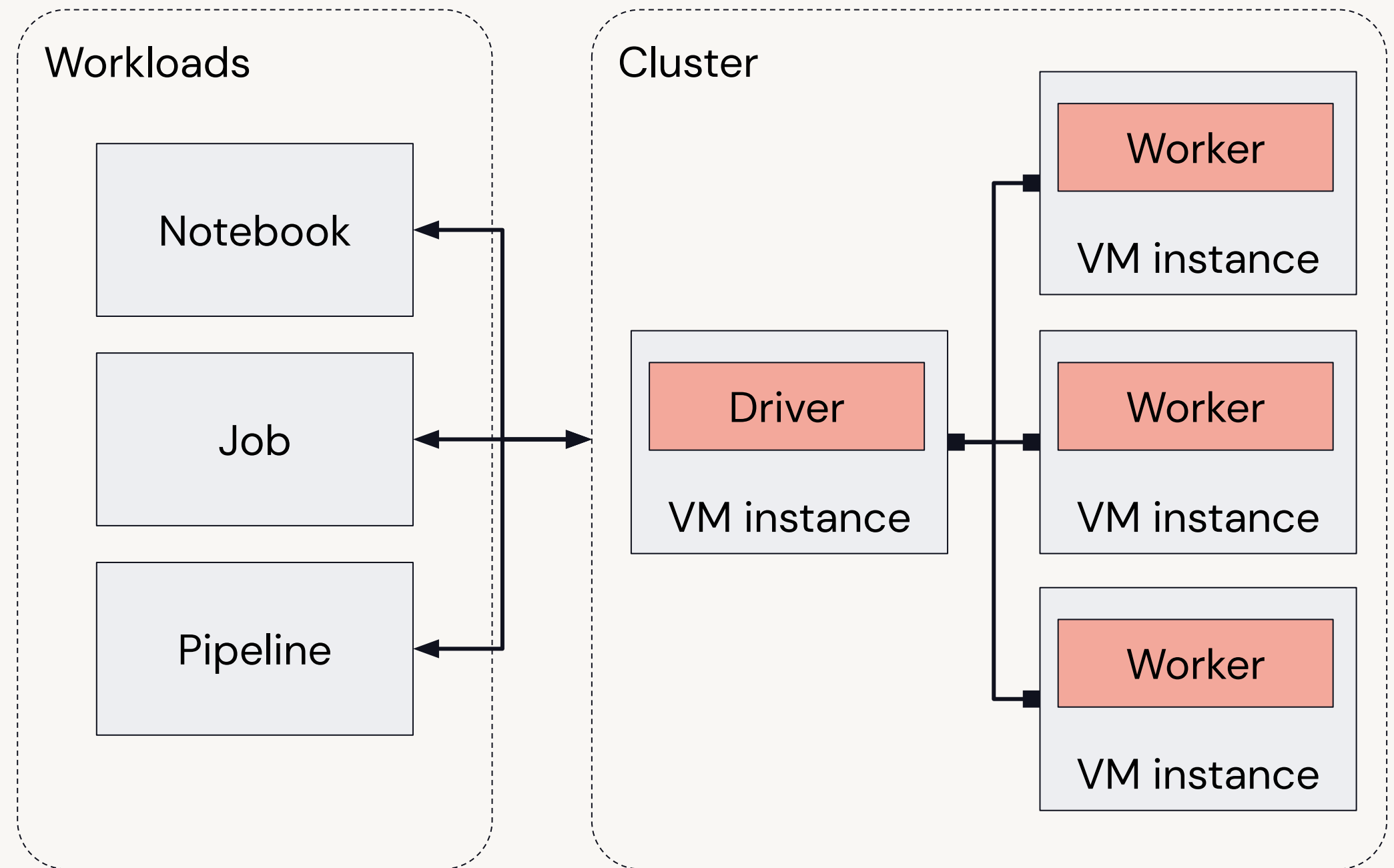
## Overview

Collection of VM instances

Distributes workloads  
across workers

Two main types:

1. **All-purpose** clusters for interactive development
2. **Job** clusters for automating workloads





# Cluster Types

## All-purpose Clusters

Analyze data collaboratively using **interactive** notebooks

Create clusters from the Workspace or API

Configuration information retained for up to 70 clusters for up to 30 days

## Job Clusters

Run **automated** jobs

The Databricks job scheduler creates job clusters when running jobs

Configuration information retained for up to 30 most recently terminated clusters

# Cluster Configuration

# Cluster Mode

## Standard (Multi Node)

Default mode for workloads developed in any supported language (requires at least two VM instances)

## Single node

Low-cost single-instance cluster catering to single-node machine learning workloads and lightweight exploratory analysis

# Databricks Runtime Version

## Standard

Apache Spark and many other components and updates to provide an optimized big data analytics experiences

## Machine learning

Adds popular machine learning libraries like TensorFlow, Keras, PyTorch, and XGBoost.

## Photon

An optional add-on to optimize SQL workloads

# Access Mode

Access mode dropdown	Visible to user	Unity Catalog support	Supported languages
Single user	Always	Yes	Python, SQL, Scala, R
Shared	Always (Premium plan required)	Yes	Python (DBR 11.1+), SQL
No isolation shared	Can be hidden by enforcing user isolation in the admin console or configuring account-level settings	No	Python, SQL, Scala, R
Custom	Only shown for existing clusters <i>without</i> access modes (i.e. legacy cluster modes, Standard or High Concurrency); not an option for creating new clusters.	No	Python, SQL, Scala, R

# Cluster Policies

Cluster policies can help to achieve the following:

- Standardize cluster configurations
- Provide predefined configurations targeting specific use cases
- Simplify the user experience
- Prevent excessive use and control cost
- Enforce correct tagging

# Cluster Access Control

	No Permissions	Can Attach To	Can Restart	Can Manage
Attach notebook		✓	✓	✓
View Spark UI, cluster metrics, driver logs		✓	✓	✓
Start, restart, terminate			✓	✓
Edit				✓
Attach library				✓
Resize				✓
Change permissions				✓



# DE 1.1: Create and Manage Interactive Clusters

Use the Clusters UI to configure and deploy a cluster  
Edit, terminate, restart, and delete clusters



# Develop Code with Notebooks & Repos

# Databricks Notebooks

Collaborative, reproducible, and enterprise ready

## Multi-language

Use Python, SQL, Scala, and R, all in one Notebook

## Collaborative

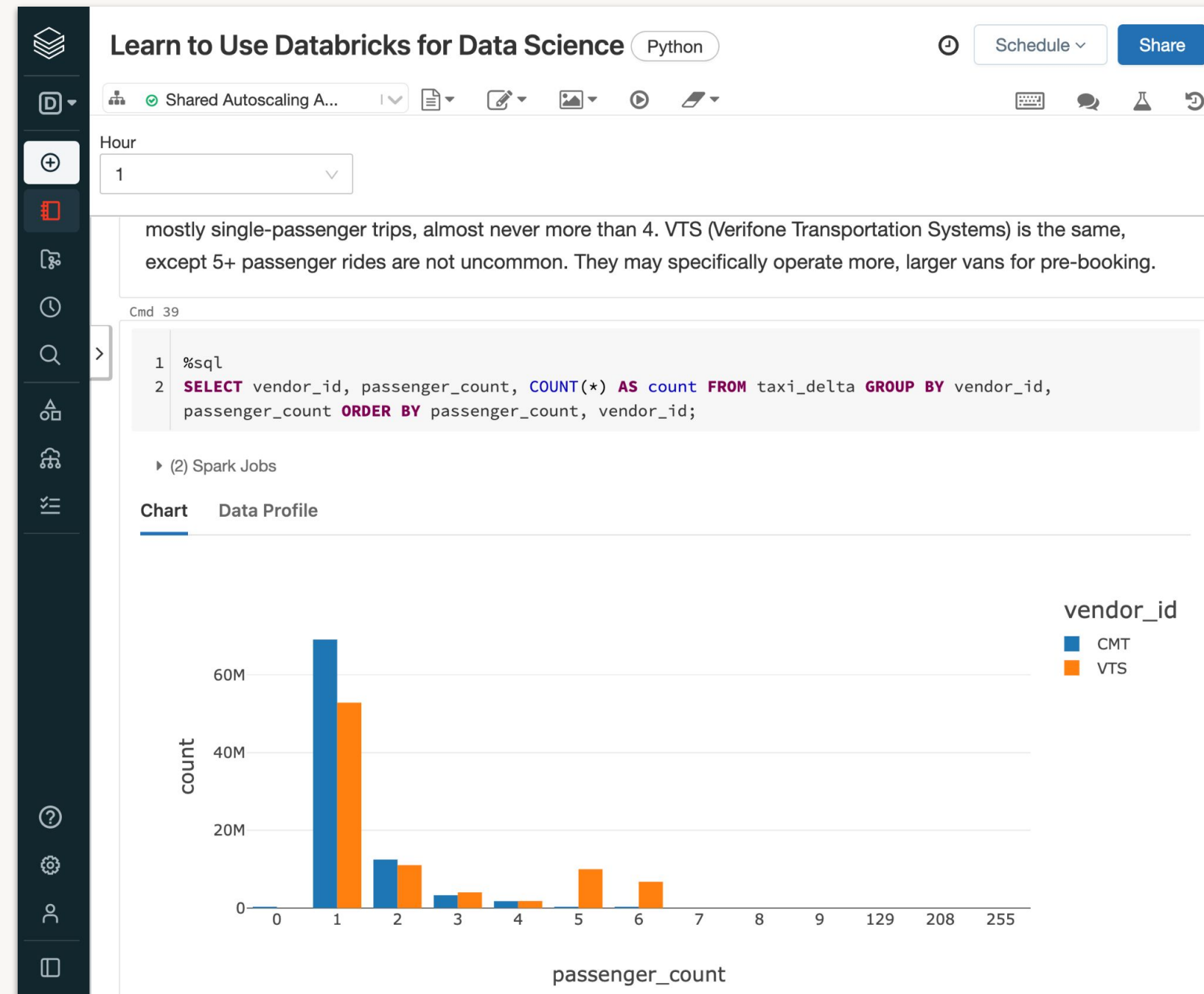
Real-time co-presence, co-editing, and commenting

## Ideal for exploration

Explore, visualize, and summarize data with built-in charts and data profiles

## Adaptable

Install standard libraries and use local modules



## Reproducible

Automatically track version history, and use git version control with Repos

## Get to production faster

Quickly schedule notebooks as jobs or create dashboards from their results, all in the Notebook

## Enterprise-ready

Enterprise-grade access controls, identity management, and auditability

# Notebook magic commands

Use to override default languages, run utilities/auxiliary commands, etc.

`%python, %r, %scala, %sql` Switch languages in a command cell

`%sh` Run shell code (only runs on driver node, not worker nodes)

`%fs` Shortcut for `dbutils` filesystem commands

`%md` Markdown for styling the display

`%run` Execute a remote notebook from a notebook

`%pip` Install new Python libraries

# dbutils (Databricks Utilities)

Perform various tasks with Databricks using notebooks

Utility	Description	Example
fs	Manipulates the Databricks filesystem (DBFS) from the console	<code>dbutils.fs.ls()</code>
secrets	Provides utilities for leveraging secrets within notebooks	<code>dbutils.secrets.get()</code>
notebook	Utilities for the control flow of a notebook	<code>dbutils.notebook.run()</code>
widgets	Methods to create and get bound value of input widgets inside notebooks	<code>dbutils.widget.text()</code>
jobs	Utilities for leveraging jobs features	<code>dbutils.jobs.taskValues.set()</code>

Available within Python, R, or Scala notebooks

# DE 1.2: Databricks Notebook Operations

Attach a notebook to a cluster to execute a cell in a notebook

Set the default language for a notebook

Describe and use magic commands

Create and run SQL, Python, and markdown cells

Export a single or collection of notebook

# DE 1.3L: Get Started with the Databricks Platform

Rename a notebook and change the default language

Attach a cluster

Use the %run magic command

Run Python and SQL cells

Create a Markdown cell

