Apache spark

Apache Spark is a lightning fast, unified analytics engine for big data processing and machine learning.

Spark was built from the ground up to address the shortcomings of Hadoop.

Hadoop was slow and inefficient for interactive and iterative computing jobs, and it was too complex

to learn and develop. On the other hand, Spark offers a much simpler, faster and easier APIs to develop on.

Spark can be 100 X faster than Hadoop, for large scale data processing by exploiting In-memory computing and other optimizations.

* Similar to most other big data engines, Spark runs on a distributed computing platform.
* Spark has an unified engine to support varying workloads. For example, it uses a single engine for streaming and batch workloads.
* It doesn't have separate one for each of those.
* It comes packaged with high level libraries, including support for SQL queries, streaming data, machine learning and graph processing.
* These standard libraries increase developer productivity and can be seamlessly combined to create complex workflows

High Level Architecture of Spark

**Spark Core:** The Spark Core takes care of scheduling tasks, memory management, fault recovery, communication with storage systems, etc. It's also home to Sparks main programming Abstraction API called RDD or Resilient Distributed Datasets.

What’s RDD?

RDDs are a collection of items distributed across various compute nodes, in the cluster that can be processed in parallel. Spark Core provides the APIs to create and manipulate these RDD collections.

It was difficult to use for complex operations and it was difficult to, optimize for Spark and mainly

down to the developer to write the optimized code.

**Spark SQL Engine:** In order to optimize the workload,

Spark introduced the SQL engine. It includes the Catalyst Optimizer, which takes care of converting a computational query to a highly efficient execution plan and the Tungsten Project, which is responsible for memory management and CPU efficiency.

The higher level abstraction such as Spark SQL and the Dataset and the DataFrame APIs, make it easier

to develop applications and also benefit from the optimizations from the SQL engine. So the recommended approach to develop applications in Spark, is to use these higher level APIs rather than the RDD API.

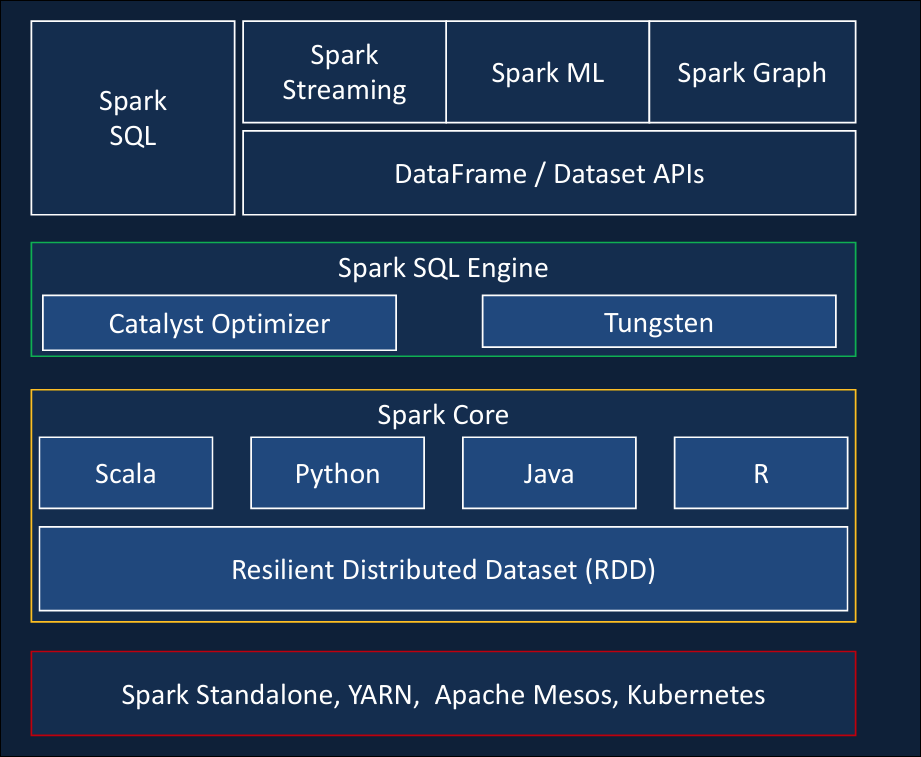
The Dataset and the DataFrame APIs can be invoked from any of the domain specific languages such as

Scala, Python, Java, or R. On top of this, we have the set of libraries such as Spark Structured

Streaming for streaming, ML Live for machine learning and also Graphics for graph processing.

Also, Spark comes with its standalone resource manager, but you can choose other resource managers

such as YARN, Apache Mesos and Kubernetes. Combining all of these, Spark provides the unified platform for doing streaming, batch, machine learning and graph processing workloads using a single execution engine and a standard set of APIs.



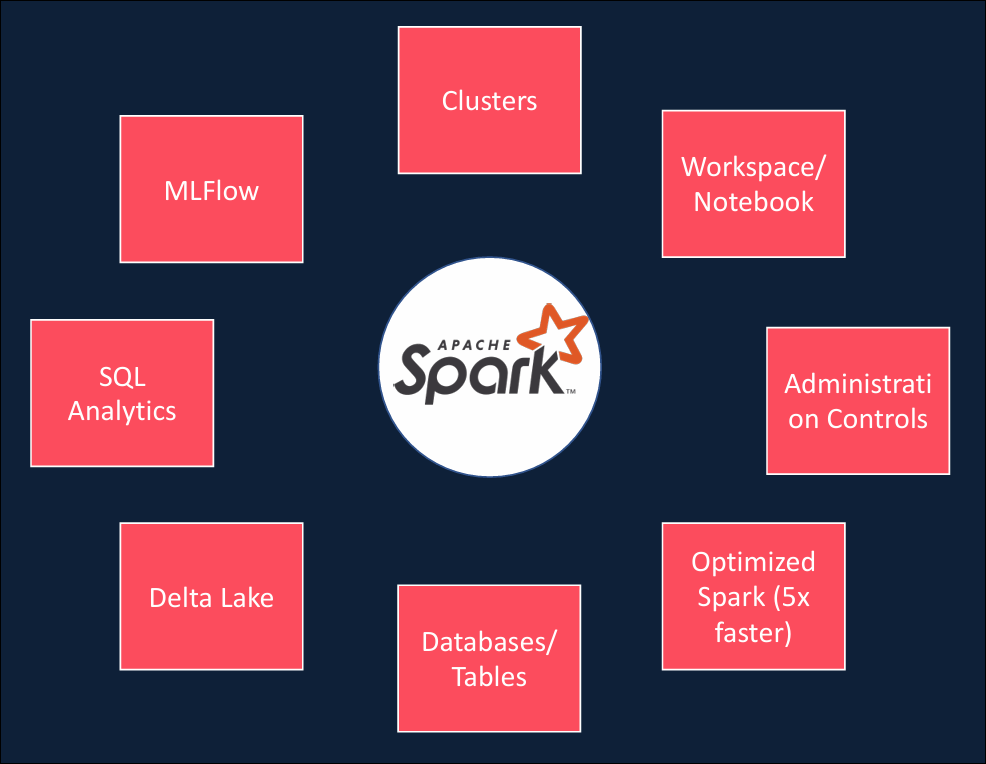
Now we know Spark is a fast execution engine with an easy to use set of higher level APIs. But, in order to work with Spark, we have to set up our own clusters, manage security, and also use third party products to write our programs. That's where Databricks comes in.

**Databricks**

Databricks is a cloud-based platform designed for big data processing, machine learning, and analytics. It provides an integrated environment for working with data using tools like Apache Spark, SQL, and other popular data science and machine learning libraries.

In simple terms, Databricks helps companies process large amounts of data quickly and efficiently, perform advanced analytics, and build machine learning models, all within a collaborative workspace. It's often used to scale and streamline data engineering and data science workflows in the cloud.

* In order for Spark to do its distributed computing, we need to spin up Clusters and install the software.
* It provides a Jupyter Notebook style IDE with additional capabilities to create your application. Collaborate with your other colleagues and also integrate with configuration management tools such as Git.
* It provides administration controls that you can use to restrict or provide access to your users, to the workspace, Clusters, etc...
* On top of this, Databricks provides the Spark runtime, which is highly optimized for the Databricks platform and known to be up to 5x faster than the Vannila Apache Spark.
* With the use of high metastore, Databricks also provides the ability to create databases and tables.
* In order to provide ACID transaction capability,Databricks also comes with the Open Source project Delta Lake,
* a recent addition to Databricks is the SQL Analytics, which provides the data analyst a SQL based analytics environment. This allows the analyst to explore data, create dashboards, schedule a regular refresh of the dashboard, etc..
* Also, it comes with managed ML flow on Databricks, which allows us to manage the machine learning lifecycle, including experimentation, deployment, model registry, etc..



But Azure's integration is deeper than others, Databricks is a first party service on Azure. What that means is, on Azure you will be buying Databricks directly from Microsoft and all support requests

are handled by Microsoft.

As a result, it provides a unified Azure Portal for Databricks and a single unified bill for all your Azure

services, including Databricks.

Azure Databricks leverages, Azure security and seamlessly integrates with Azure Active Directory

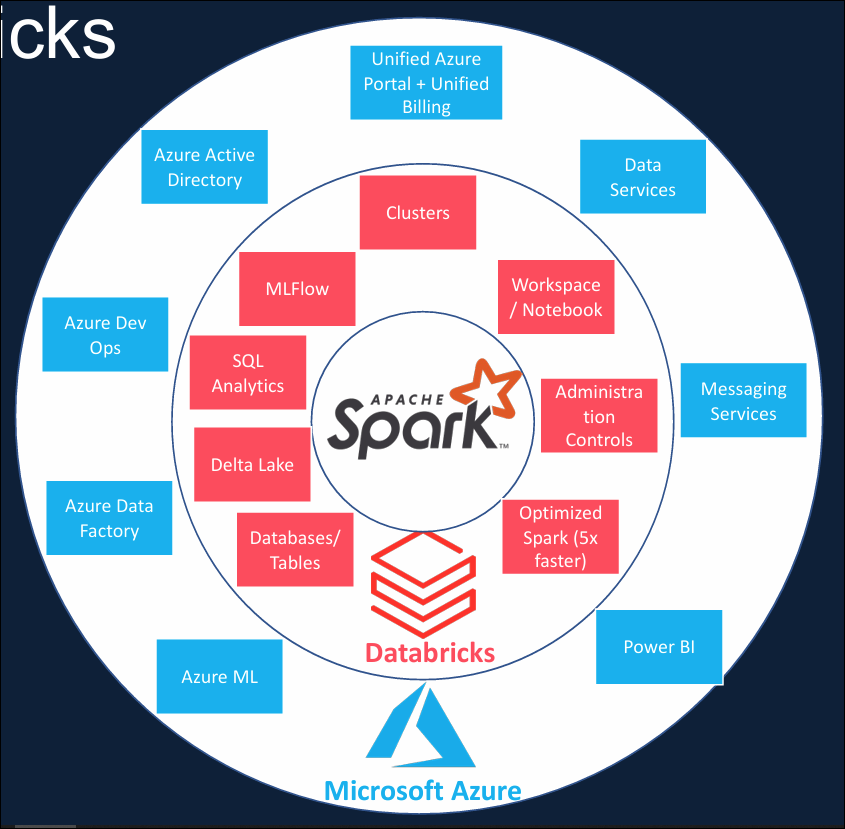
and single sign on.

It provides seamless integration and high-speed connectors between various Azure data services such as Azure Data Lake, Blob Storage, Cosmos DB, SQL DB and Synapse. Messaging services such as Event Hub and IoT Hub, Power BI and Azure ML, you can seamlessly run Databricks notebooks from Azure Data Factory and integrate with the rest of the data workflow in your data project.

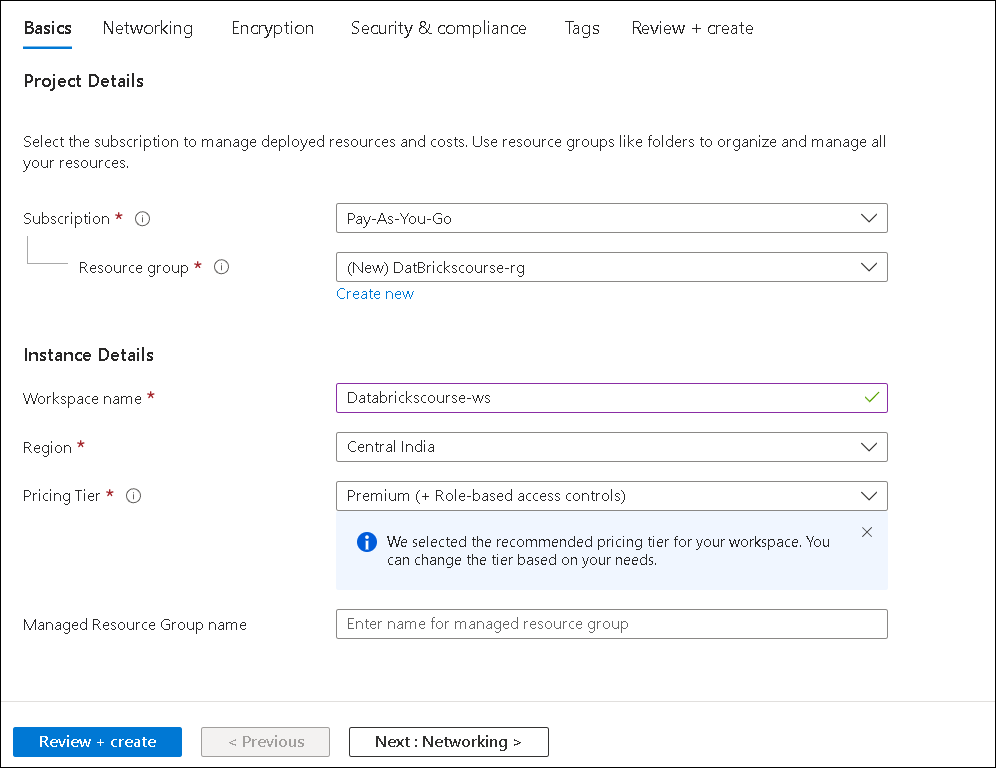
And finally, Databricks also connects with Azure Dev Ops to enable continuous integration and continuous

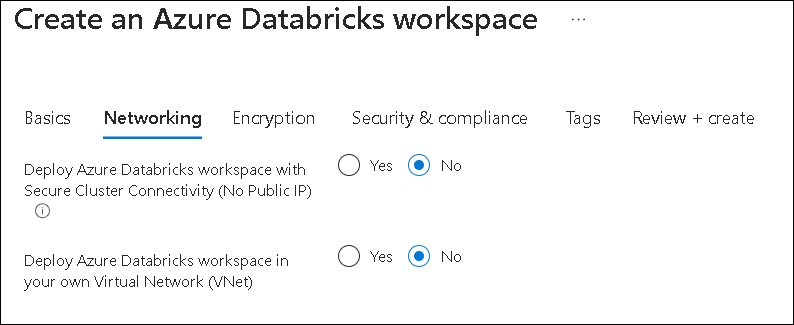
deployment.

So just to summarize, Azure Databricks is a spark based unified data analytics, platform as a service offering, that's optimized for the Microsoft Azure Cloud.

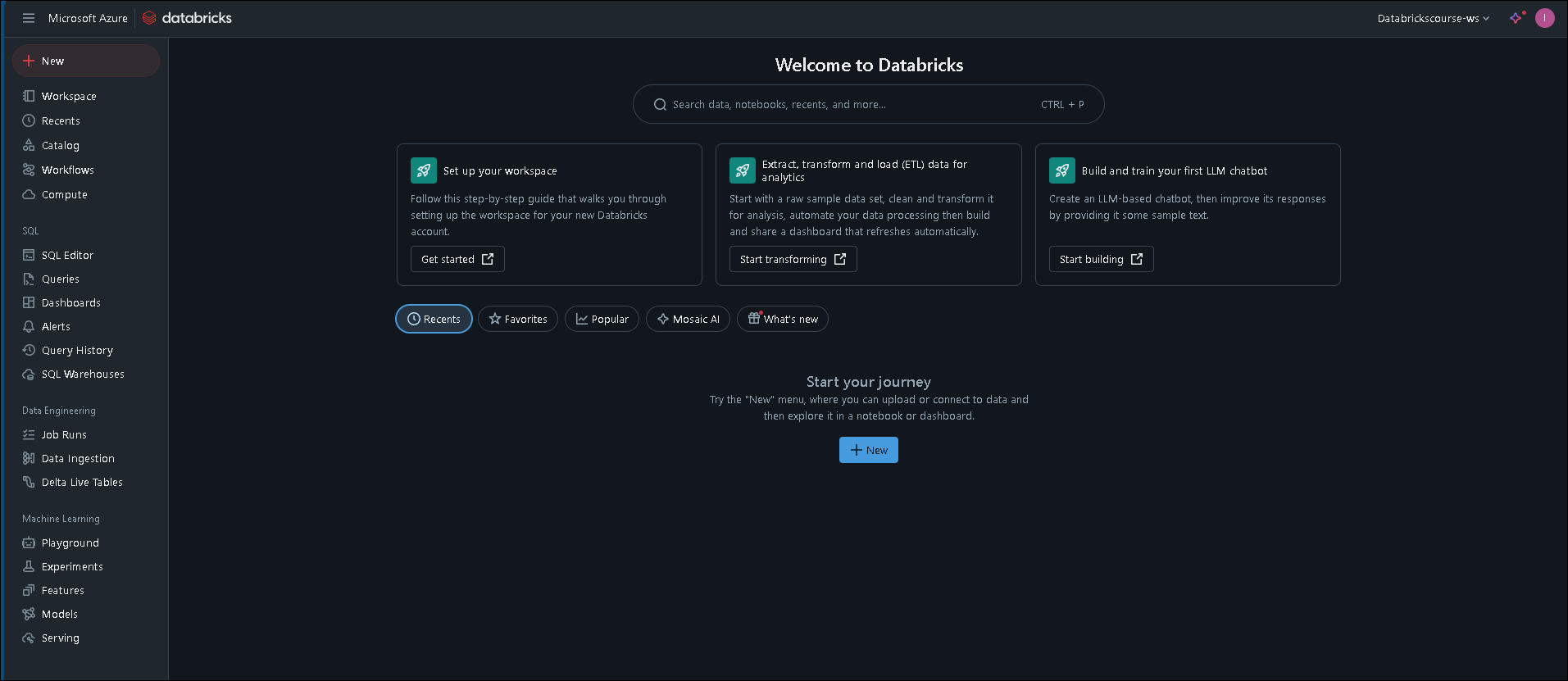


Create Data bricks workspace





There after just move to review and create pin it to dashboard and launch work space.



As we saw in the overview, Databricks offers multiple products to help with data warehousing, data engineering and machine learning.

This menu has also been designed to represent these product areas.

The section on the top of the navigation bar is the important one. It has the menu for accessing common functionalities across all three product areas we discussed.

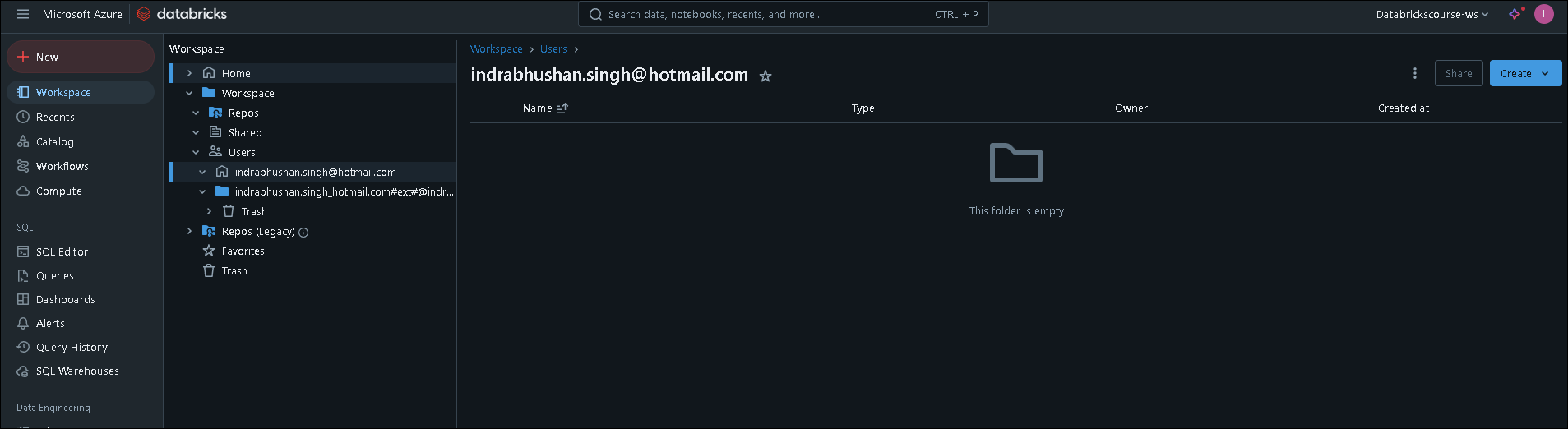
All the menu items you see under SQL relates specifically to Databricks SQL, which is Databricks' offering

to build SQL data warehouses on the Lakehouse platform.

The next section is data engineering, which includes specific products related to data engineering.

Similarly, the machine learning section includes products that are specific for machine learning workloads.

You can collapse these sections as you wish.



Workspace is basically a container for holding a set of folders, libraries, and files.

By default, each user has their own workspace and also there is a shared workspace which you can use to share assets amongst other users in this Databricks workspace. By right clicking on the workspace, you can create folders,

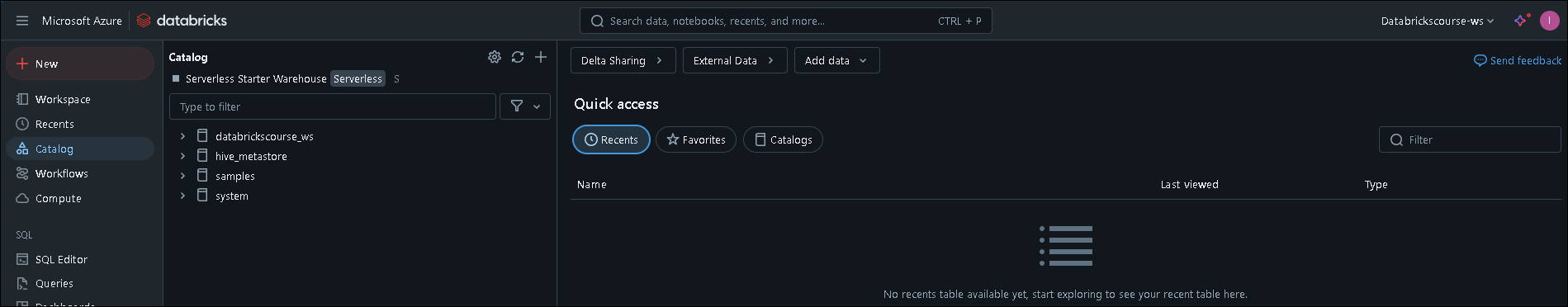
libraries, MLflow experiments, notebooks, et cetera. Also, you can import one of these type of assets from elsewhere. You also have the option to export your Databricks notebooks or folders into a Databricks file format called DBC or into source specific formats, for example, Python, SQL, Scala, et cetera.

Repos option here gives you a visual Git client within Databricks. Databricks lets you integrate this workspace with Git repositories offered by most of the Git providers, such as GitHub, Bitbucket, Azure DevOps Services, et cetera.

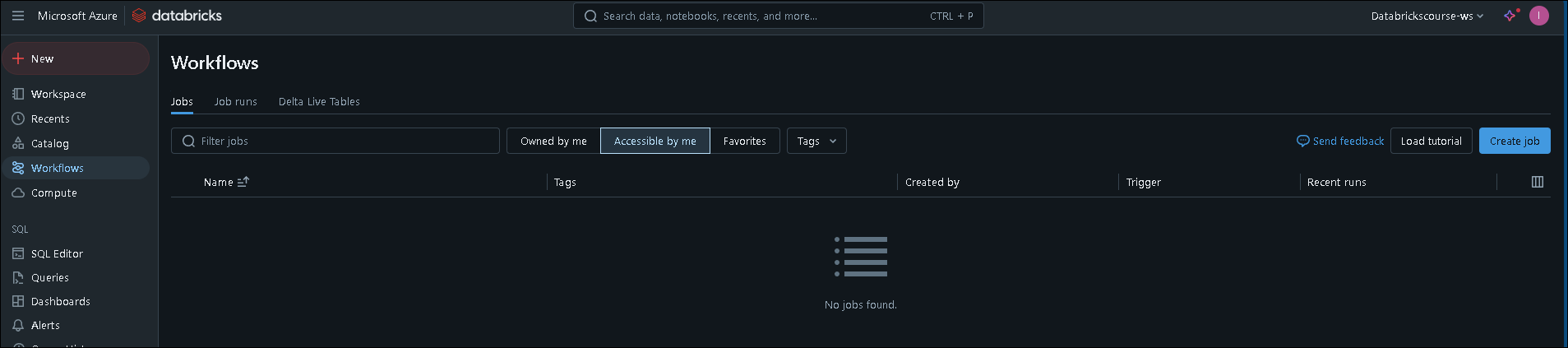
And it supports standard Git operations such as commit, push, pull, et cetera from a Git repository.

And finally deleted notebooks or files are kept in trash up to 30 days.

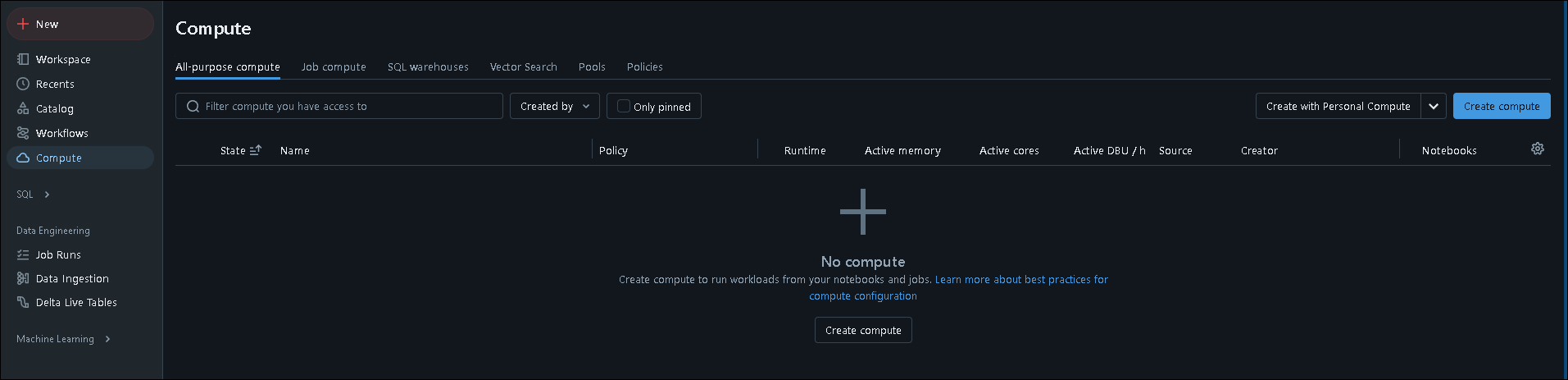
and any recently modified files will be listed under recents menu, so you can get quick and easy access to them.



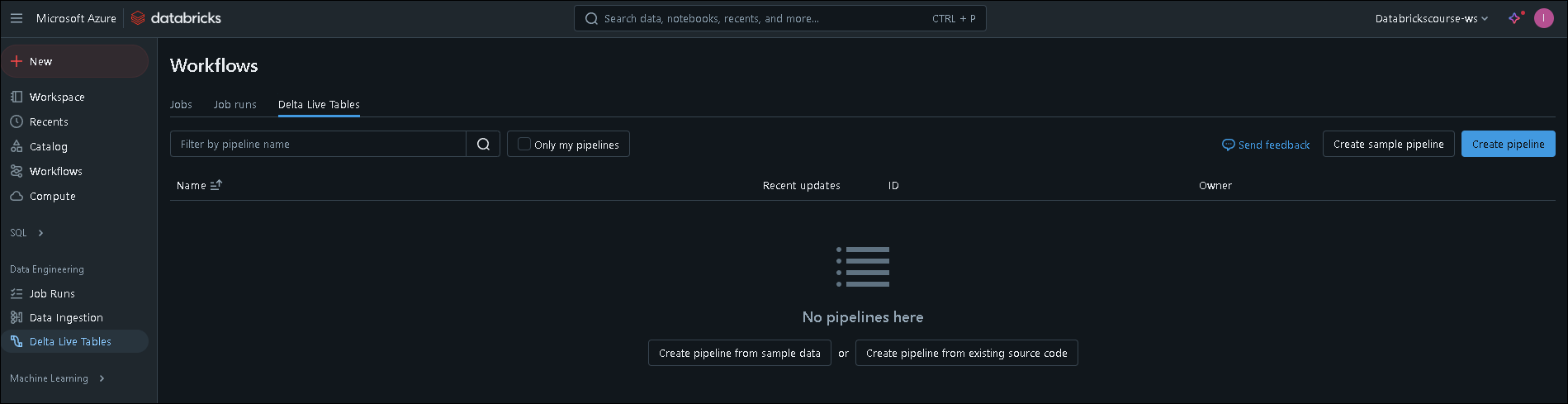
Catalog menu here lets us interact with any tables of use that you've already createdand also lets you create new tables in Databricks.Please note that this menu used to be called data,rather than catalog.



Workflows menu contains jobs, job runs and Delta Live Tables. Databricks jobs basically lets you schedule notebooks periodically via a scheduling system. You can create them here and monitor them under job runs. Delta Live Tables, or DLT, is a new offering by Databricks. This is an ETL framework that uses declarative approach to building data pipelines with automated testing and it is still evolving.



Compute menu here lets us create clusters, cluster pools and SQL data warehouses. You can either create an all-purpose or a job compute cluster.

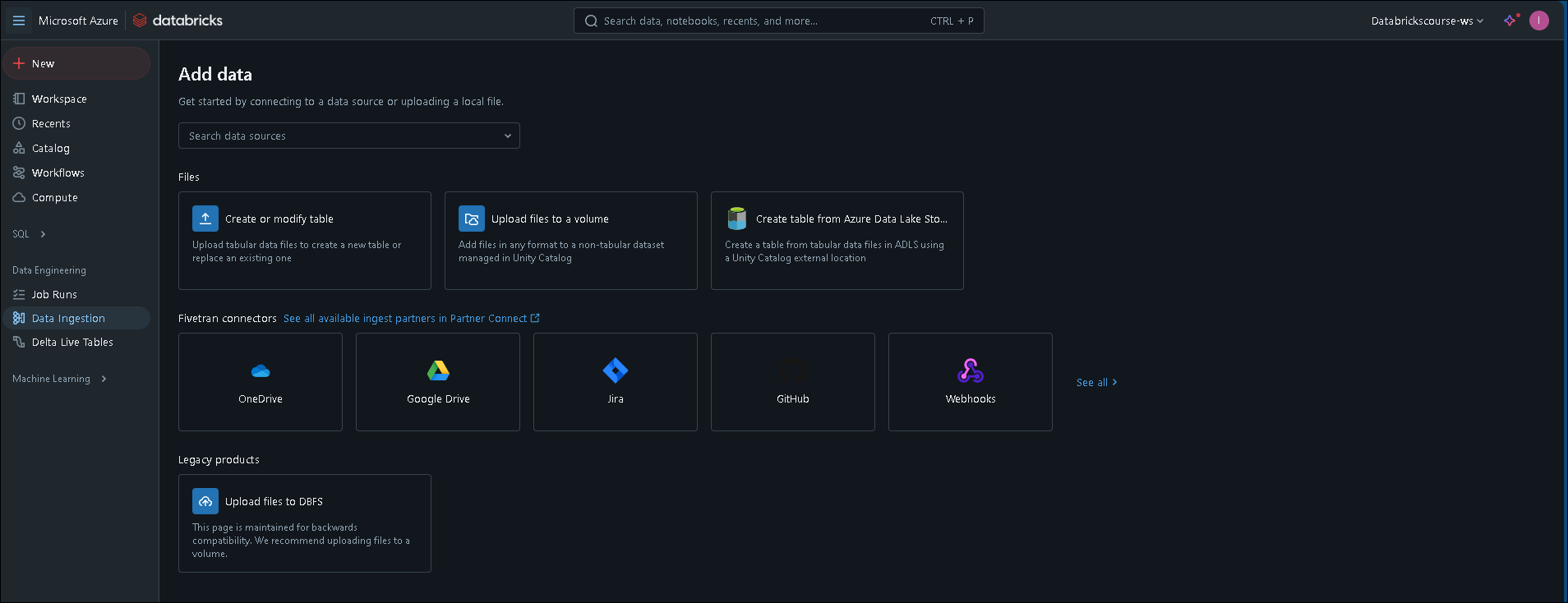


Let's now switch our focus to the data engineering section. To be honest, most of this is redundant and we've already seen them under workflows. Let me navigate to workflows. As you can see, you can navigate to job runs and Delta Live Tables from here. And the default page for workflows is jobs.

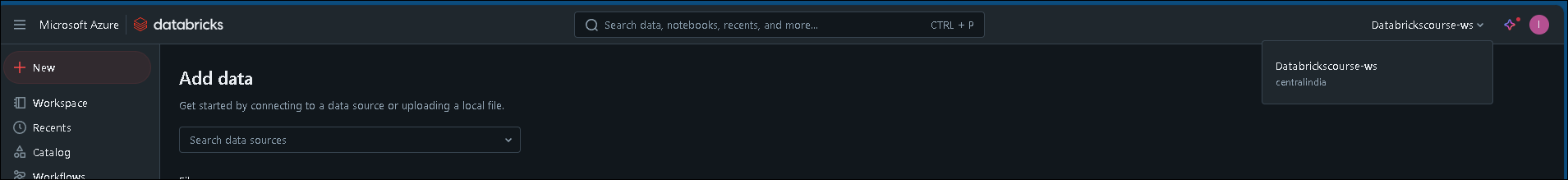
The job runs and Delta Live Tables here under the data engineering also takes us to this page. But when you click on job runs, the default page is job runs, and when you click on Delta Live Tables, the default page is Delta Live Tables but it is exactly the same as what you see under workflows.

And also, you may notice that when you come and click on Delta Live Tables, the title here still says workflows.

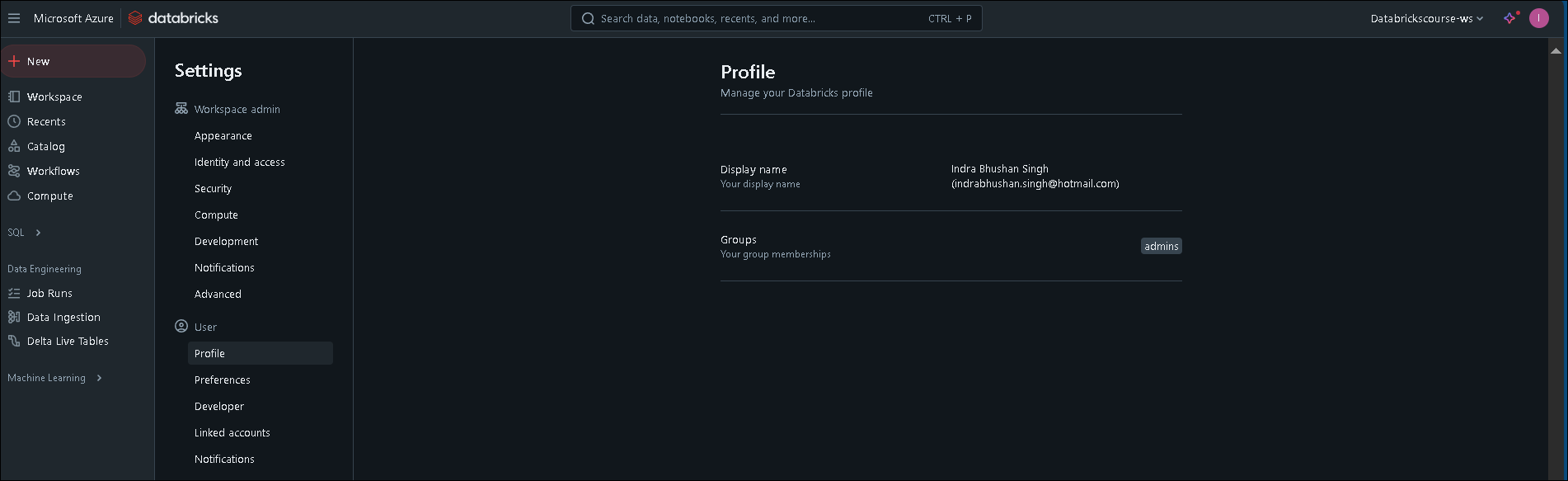
So it is basically a duplicate of what you see under workflows



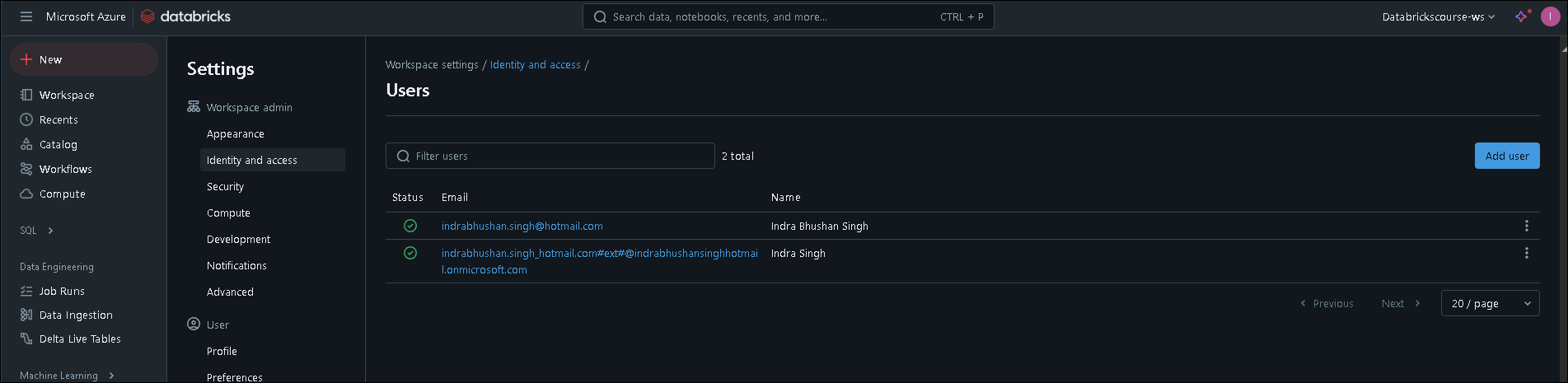
The menu data ingestion lets you create tables from data stored in your local computer. Also, you can get some start notebooks to ingest data from one of these sources and you can find some sample data here too. It also lets us develop starter ingestion pipelines using the third-party partner called Fivetran. But please note that Fivetran is outside of the scope of this course and it's a separate product all together.



The search bar at the top is very powerful and it lets you search within the entire workspace. By clicking on the advanced search, you can also choose to search within specific Databricks assets, such as notebooks, jobs, folders, et cetera. The menu option here lets you search to the other Databricks workspaces. We're currently in the Databricks course workspace. From here, we can switch to different workspaces without going to the Azure portal.

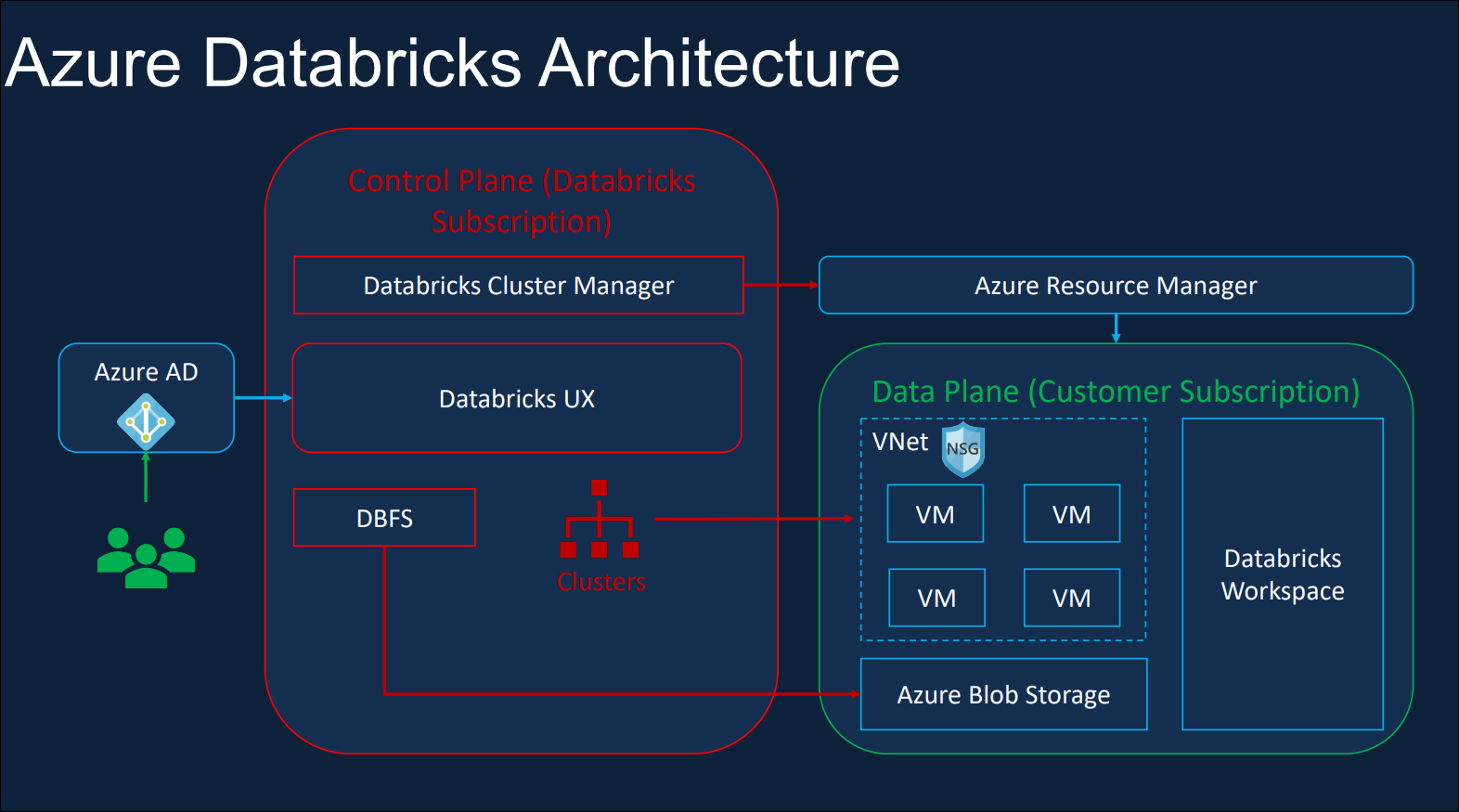


User settings lets us set the preferred language and also, you can set the developer settings here. You can also enable Git integration for this workspace



and the identity and access here gives us the ability to manage users and groups, as well as workspaces and warehouse settings.

**Azure Databricks Architecture**



Databricks Architecture is basically split into two parts, one called the Control Plane and another one called the Data Plane. Control plane is located in Databricks own subscription. This contains the Databricks UX and also the Cluster Manager. It's also home to the Databricks File System (DBFS) and also metadata about Clusters, Files mounted, etc. Data Plane is located in the customer subscription.

When you create a Databricks service in Azure, there are four resources created in your subscription, a Virtual Network and Network Security Group for the Virtual Network. Azure Blob Storage for the default storage and also a Databricks Workspace.

We've just created the Databricks Service.

So let's switch over to the Azure Portal and have a look at those.The Databricks uses such as Data Engineers, Data Scientist and Data Analyst, will use Active Directory Single Sign On, to access the Databricks service. When a user request for a Cluster to be created, the Databricks Cluster Manager will create the required virtual machines in our subscription via the Azure Resource Manager.

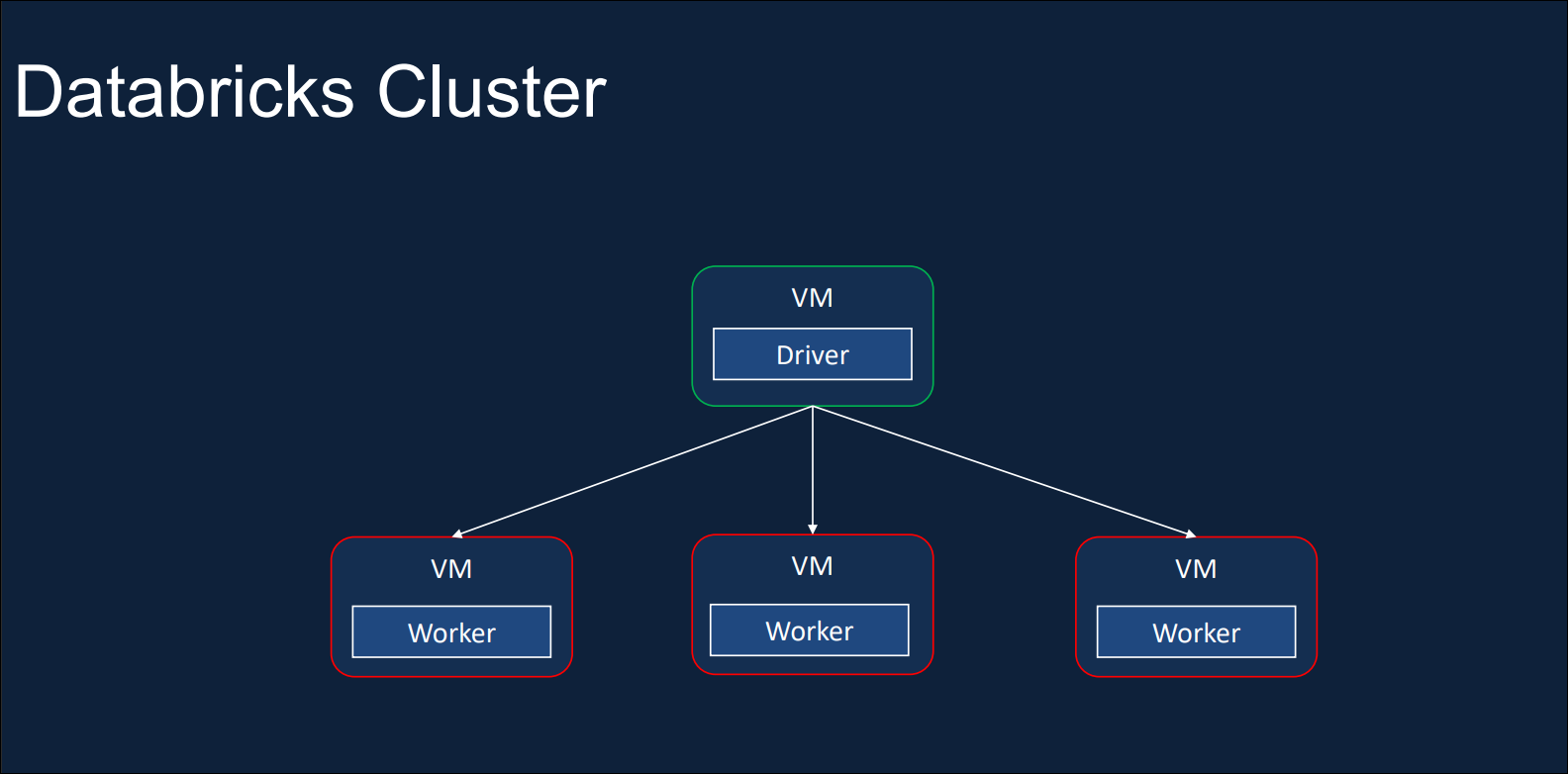
So none of the customer data leaves a subscription. Temporary outputs such as running a display command or data for manage tables, are stored in the Azure Blob Storage, and the processing also happens within the VNet in our subscription.

Blob Storage, and the processing also happens within the VNet in our subscription.

The Azure Blob Storage we have shown here is the default storage or otherwise called the DBFS a route,

and it's not recommended as a permanent data storage.

**Databricks Clusters**



A Cluster is basically a collection of Virtual Machines. In a Cluster, there is usually a Driver node,

which orchestrates the tasks performed by one or more worker nodes.

Clusters allow us to treat this group of computers, as a single compute engine via the Driver node.

Databricks Clusters enable us to run different types of workloads, such as ETL for Data Engineering,

Data Science and Machine Learning workloads.

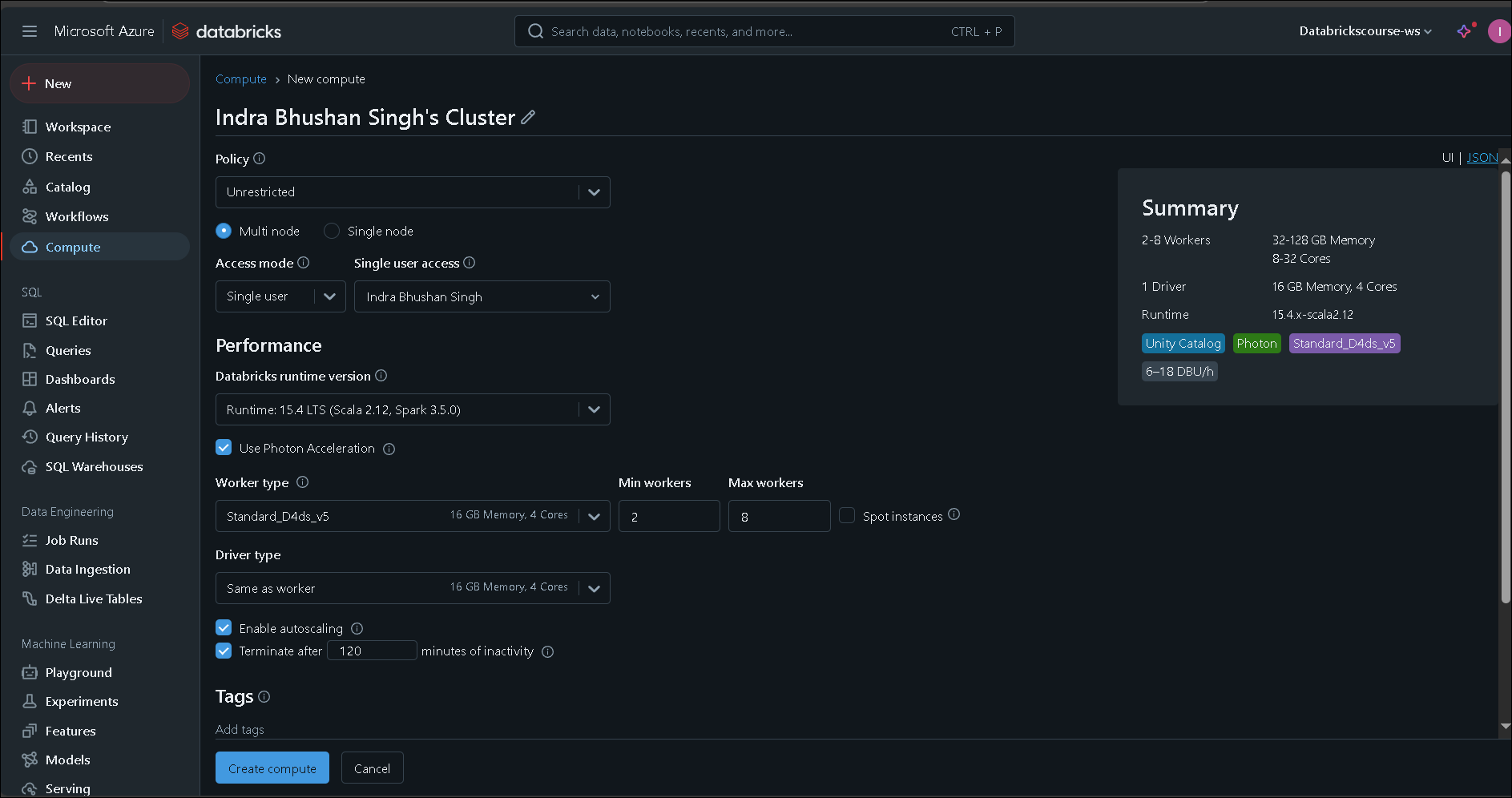
Databricks offers two types of Clusters.

* The first one is the **All purpose Cluster**, which is created manually, via the Graphical User Interface, the CLI or the API. Whereas, the **Job Clusters** are created when a job starts to execute, and the job has been configured to use a Job Cluster.
* All Purpose Clusters are persistent, they can be terminated and restarted at any point in time, whereas the Job Clusters are terminated at the end of the job. They cannot be restarted. So they're no longer usable once the job has been completed.
* All Purpose Clusters are suitable for interactive and ad-hoc Analysis workloads. On the other hand, Job Clusters are suitable for automated workloads, such as running an ETL pipeline or Machine Learning workflow at a regular interval.
* All Purpose Clusters can be shared among many users, and they are good for collaborative analysis, whereas the Job Clusters are isolated just for the job being executed.
* All Purpose Clusters are expensive to run compared to the Job Clusters.
* In summary, All Purpose Clusters are great for interactive analysis and ad-hoc work, whereas Job Clusters are great for repeated production workloads.

**Cluster Pool:** Cluster Pools give you the ability to set aside some ready to use compute capacity, so that when you create an All Purpose Cluster, it can be created quickly. Usually when you create a Cluster, it takes about 5 to 6 minutes to spin up a Cluster. In order to speed up that time, you can have a pool of resources waiting for you via Cluster Pools. And that's where Cluster Pools come in.  
  
**Cluster Policy:** When creating a Cluster, there is a ton of configuration options to specify.

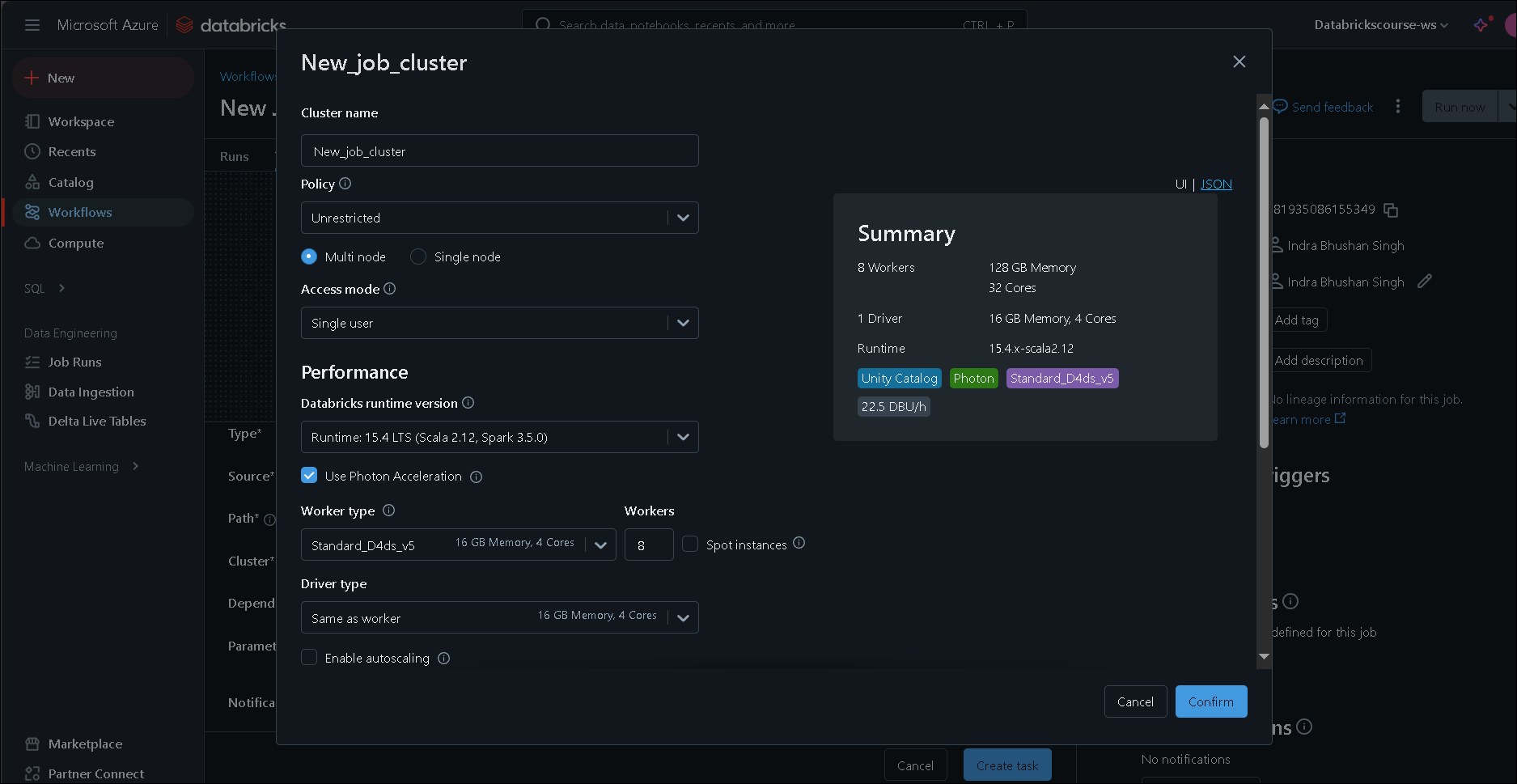
Cluster Policy helps us pre-configure some of these details, so that creating a Cluster becomes simpler,

and also, it helps restrict the maximum size of the Clusters being created to keep the cost under control.



As we said before, All Purpose Clusters can be created manually, and we do that by clicking the Create

compute button here, and we can specify the required configuration details here.



Let's now navigate to the Job's compute. As you can see, Job compute is missing the Create button.

That's because the Job Clusters cannot be created manually, Job Clusters are created when a job starts to run and it's destroyed as soon as the job completes. You can create Databricks Jobs, as part of workflows by selecting the Workflow icon here on the sidebar. As you can see, you can create Jobs and the Job Cluster by selecting the New Job Cluster menu here. As we said, the Cluster will be created when the job runs and destroyed as soon as it completes.

**Create Cluster**

When we come to create the Cluster, We will be presented with a number of configuration options as shown here. In this lesson, I'll walk you through each one of these in detail. First, we have the option to choose whether we want to create a Single Node or a Multi Node Cluster.

* Multi Node Cluster will have one Driver Node and one or more Worker Nodes. When you run a Spark Job against a Multi Node Cluster, the Driver Node will distribute the tasks to run on the Worker Nodes in parallel, and returns the result. They give us the ability to horizontally scale the Cluster depending on your workload. We can basically keep adding Worker Nodes as we need. These are the default type of Clusters used for Spark Jobs and suitable for large workloads.
* On the other hand, Single Node Cluster will have only one node, which is the Driver Node and there are no Worker Nodes. Even though, there are no Worker Nodes, Single Node Clusters also supports Spark workloads. When you run a Spark Job, the Driver Node acts as both the driver and the worker. As there are no Worker Nodes, the Single Node Clusters are not horizontally scalable, so they're not suitable for large ETL workloads. They're mainly targeted for lightweight Machine Learning and Data Analysis workloads which don't require, any distributed compute capacity.

**Create Access Mode**

We then need to define the Access Mode.

There are four different types of Access Modes available at the moment for the Cluster.

* As the name suggests, Single User access mode only allows a single user to access the Cluster. It supports all four languages Python, SQL, Scala, and R.
* Shared access mode allows the Cluster to be shared amongst more than one user, but it provides process isolation. Each process gets its environment, so one process can't see the data or the credential used by the other one. It's only available on premium workspaces. Also, it only supports Python and SQL workloads.
* No Isolation Shared also allows the Cluster to be shared amongst more than one user. It's available on both standard and premium workspaces. Also, it supports all four languages Python, Scala, SQL and R.
* The main difference between this and the Shard access mode is that, No Isolation Shared access mode doesn't provide any process isolation. So failure in one user's process may affect the others.
* Also, they don't offer any task preemption, so one running process may use all the resources and the others may fail. And most importantly, as everything is shared, it's considered less secure.
* Custom access mode is not an option,