databricks

Automate Production Workflows

Agenda
Automate Production Workloads

Lesson Name	Lesson Name
Lecture: Introduction to REST API and CLI	ADE 6.3 – Follow Along Demo – Using the Databricks API
ADE 6.1 – Follow Along Demo – Generate Tokens	ADE 6.4 – Follow Along Demo – Deploy a Pipeline with the CLI
ADE 6.2 – Follow Along Demo – Using the Databricks CLI	ADE 6.5L – Deploy Workloads with the API Lab
Lecture: Deploy Batch and Streaming Jobs	Lecture: Working with Terraform

Introduction to REST API & CLI

Learning Objectives

By the end of this lesson, you should be able to:

1 Describe how the to use the CLI/API to interact with Databricks jobs

2 Programmatically deploy a workload using the REST API

3 Programmatically deploy a workload using the CLI

Databricks REST API
Overview of the REST API

- Most of the actions needed to be completed in the Databricks workspace can be accomplished with REST API
- 3rd party tools can be integrated with Databricks using REST API
- Databricks CLI depends on the REST API
- Users can use curl, Postman, Python or other tools.


```
GET  /api/2.1/jobs/list
```

```json
{
    "has_more": false,
    "jobs": [
        {
            "created_time": 1601370337343,
            "creator_user_name": "user.name@databrick
            "job_id": 11223344,
            "settings": {
                "format": "MULTI_TASK",
                "name": "A multitask job",
                "job_clusters": [ … ],
                "email_notifications": { … },
                "tags": { … },
                "timeout_seconds": 86400,
                "tasks": [ … ],
                "max_concurrent_runs": 10,
                "schedule": { … },
                "webhook_notifications": { … },
                "git_source": { … }
            }
        }
    ]
}
```

Databricks REST API
Configuring REST API

Requirements:

- Databricks Instance ID (part of the workspace URL). Example: **dbc-a1b2345c-d6e7**.cloud.databricks.com.

- Personal Access Token (PAT): A **Bearer** authentication token can be found under "User Settings"

- PAT can be stored in a **.netrc file** to be used in curl or can be passed in the header for HTTP requests.

```
CURL get Request (with .netrc file)

curl --netrc --get \
https://abc-d1e2345f-a6b2.cloud.databricks.com/api/2.0/clusters/get \
--data cluster_id=1234-567890-patch123
```

```
REST API Response

{
  "cluster_id": "1234-567890-patch123",
  "spark_context_id": 123456789012345678,
  "cluster_name": "job-239-run-1",
  "spark_version": "8.1.x-scala2.12",
  ...
}
```

Databricks REST API

REST API 2.0 Endpoints

Developer Automation Tasks

- **Workspace API:** download, upload, manage files in workspace
- **Jobs API:** create, manage, and trigger runs of jobs
- **Clusters API:** create, manage, edit clusters
- **DBFS API:** interact with Databricks File System
- **Libraries API:** upload and install libraries
- **Tokens API:** create and revoke tokens

Admin Tasks

- **Account Users/Groups:** Account API, Groups API, SCIM API
- **Security:** Permissions API, Secrets API, Token Management API, IP Access List API
- **Clusters/Compute:** Cluster Policies API, Instance Pools API, Global Init Scripts API, Instance Profiles API

Databricks CLI

There are two CLIs available. CLIs are built on top of the REST API and organized into command groups.

Databricks CLI

Used for Data Science & Engineering workspace assets such as cluster policies, clusters, file systems, groups, pools, jobs. Use Cases:

- Provision compute resources in Databricks workspaces
- Run data processing and data analysis tasks
- List, import, and export notebooks and folders in workspaces

Databricks SQL CLI

Used for SQL warehouses. Use cases:

- Run SQL queries on existing warehouses
- Run queries from **a query string**. Example: `dbsqlcli -e "SELECT * FROM default.diamonds LIMIT 2"`
- Run queries from **a text file**. Example: `dbsqlcli -e my-query.sql`
- Run queries in read–evaluate–print loop (**REPL**)

Databricks CLI

Install and Configure the CLI

- Install CLI using pip

- Setup authentication: Enter host name and Personal Access Token as prompted. (credentials are stored in the file `~/.databrickscfg` on Unix, Linux, or macOS, or `%USERPROFILE%\.databrickscfg` on Windows)

- Invoke commands

```
Install CLI

pip install databricks-cli
```

```
Configure CLI

databricks configure --token
```

```
Test Command

databricks workspace ls
/Users/<someone@example.com>
```

Demo:

Using the CLI

Demo:
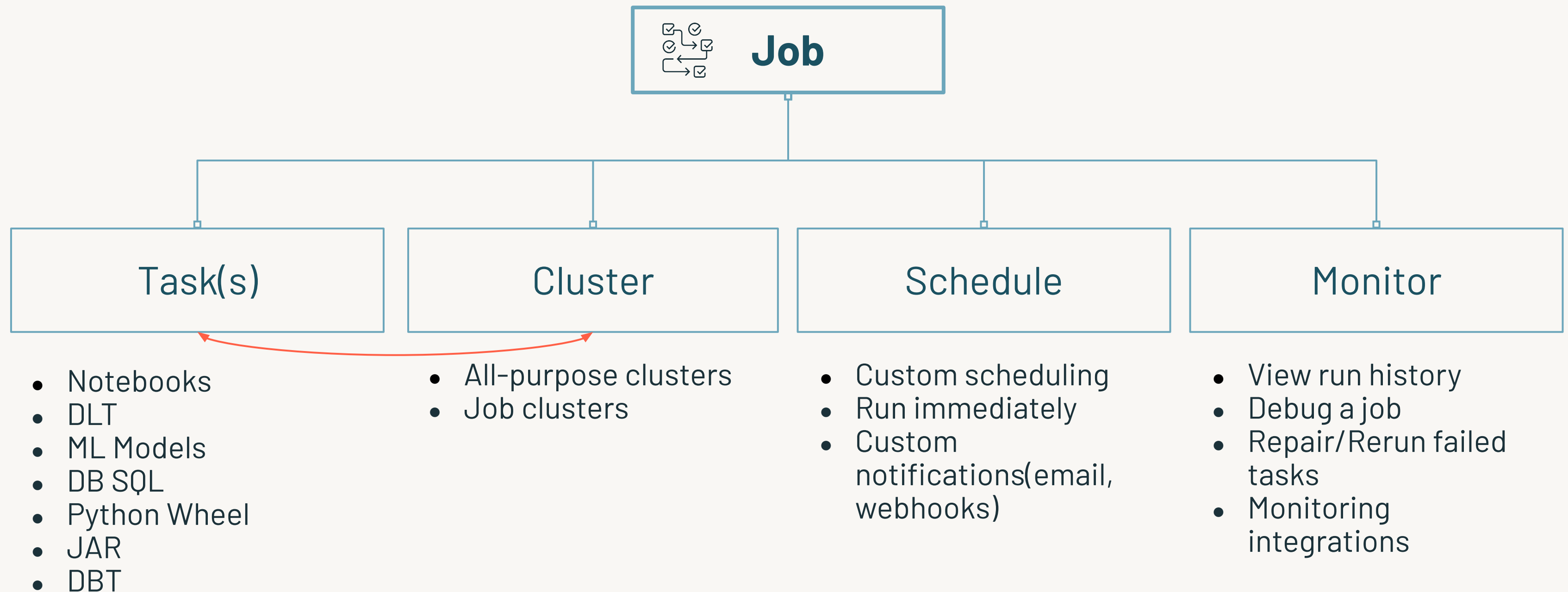
Using the Databricks API
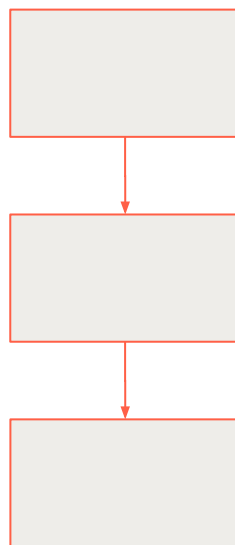
Deploy Batch and Streaming Jobs

Workflows Revisited

Main Components of a Job

Job

Task(s)	Cluster	Schedule	Monitor

Task(s)
- Notebooks
- DLT
- ML Models
- DB SQL
- Python Wheel
- JAR
- DBT

Cluster
- All-purpose clusters
- Job clusters

Schedule
- Custom scheduling
- Run immediately
- Custom notifications(email, webhooks)

Monitor
- View run history
- Debug a job
- Repair/Rerun failed tasks
- Monitoring integrations
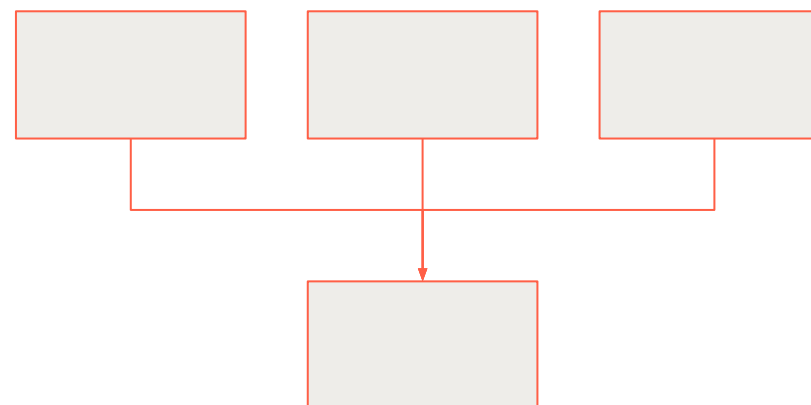
Workflows Revisited
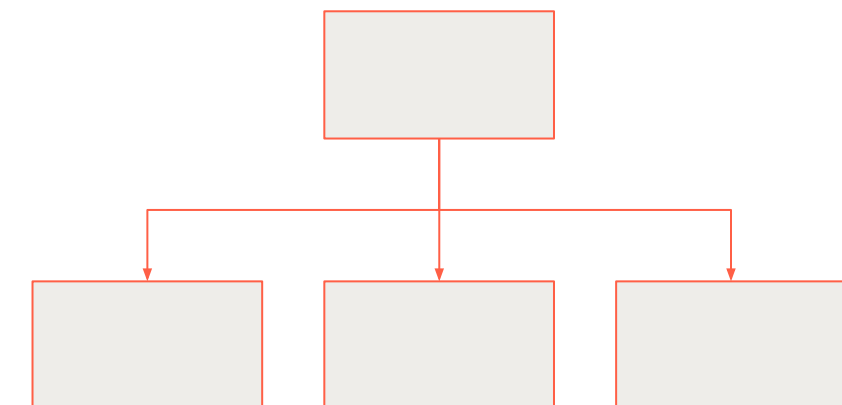## Common Workflow Job Patterns

Sequence

- Data transformation/processing/cleaning
- Bronze/Silver/Gold

Funnel

- Multiple data sources
- Data collection

Fan-out

- Single data source
- Data ingestion and distribution
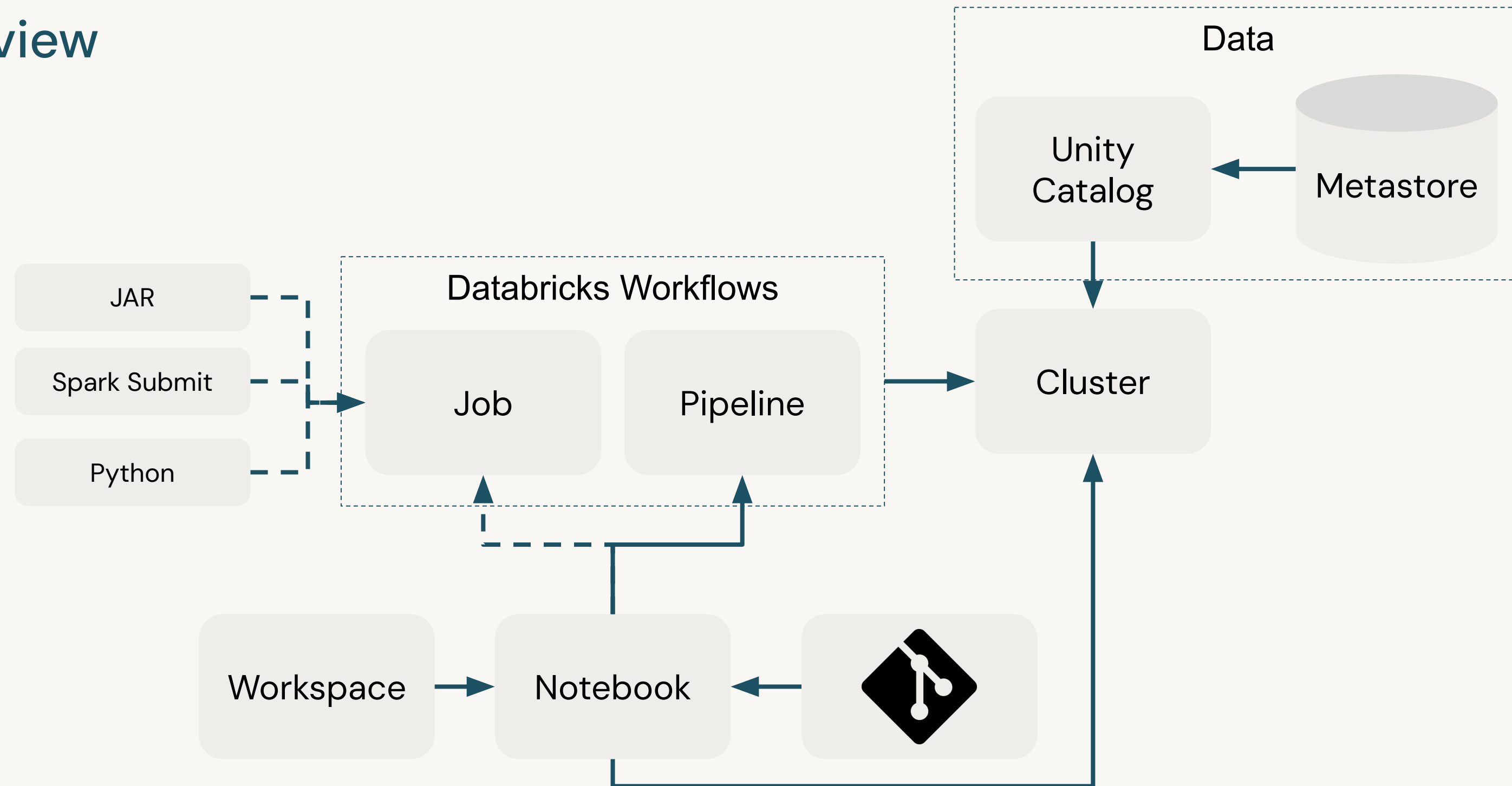
Workflow Automation & CICD Integration ⊟

Workflows provide multiple options to automate and deploy with CI/CD pipelines

- **Databricks Asset Bundle**: Databricks Asset Bundles or DABs are a collection of Databricks artifacts. Using the Databricks CLI these bundles can be materialized across multiple workspaces like dev ,staging and production.
- **Python SDK**: The Jobs Python API allows you to create, edit, and delete jobs.
- **Terraform**: databricks_job resource can be used to create and manage jobs using Terraform
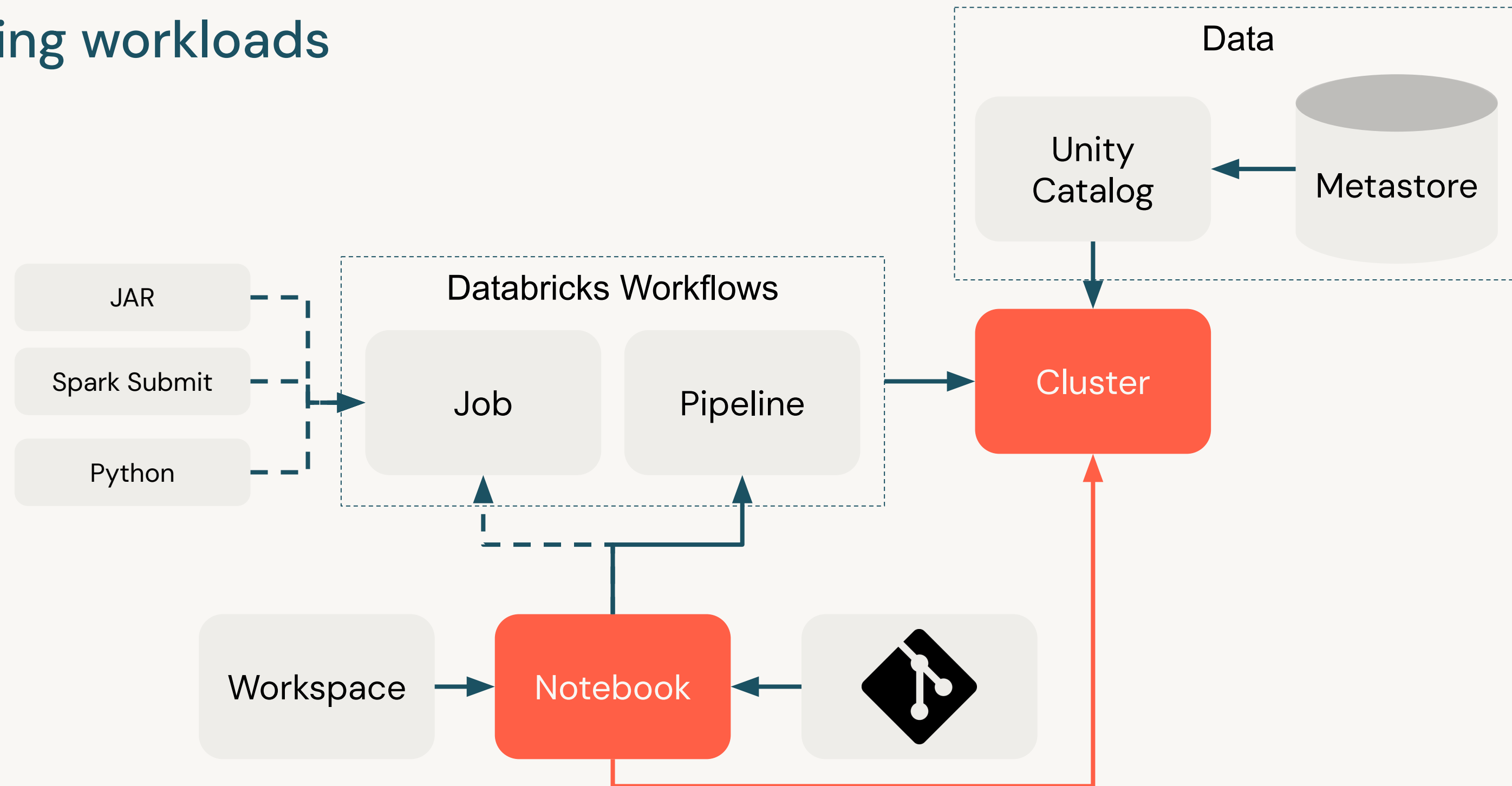- **Rest API**: The Jobs REST API allows to manage Databricks Jobs programmatically.
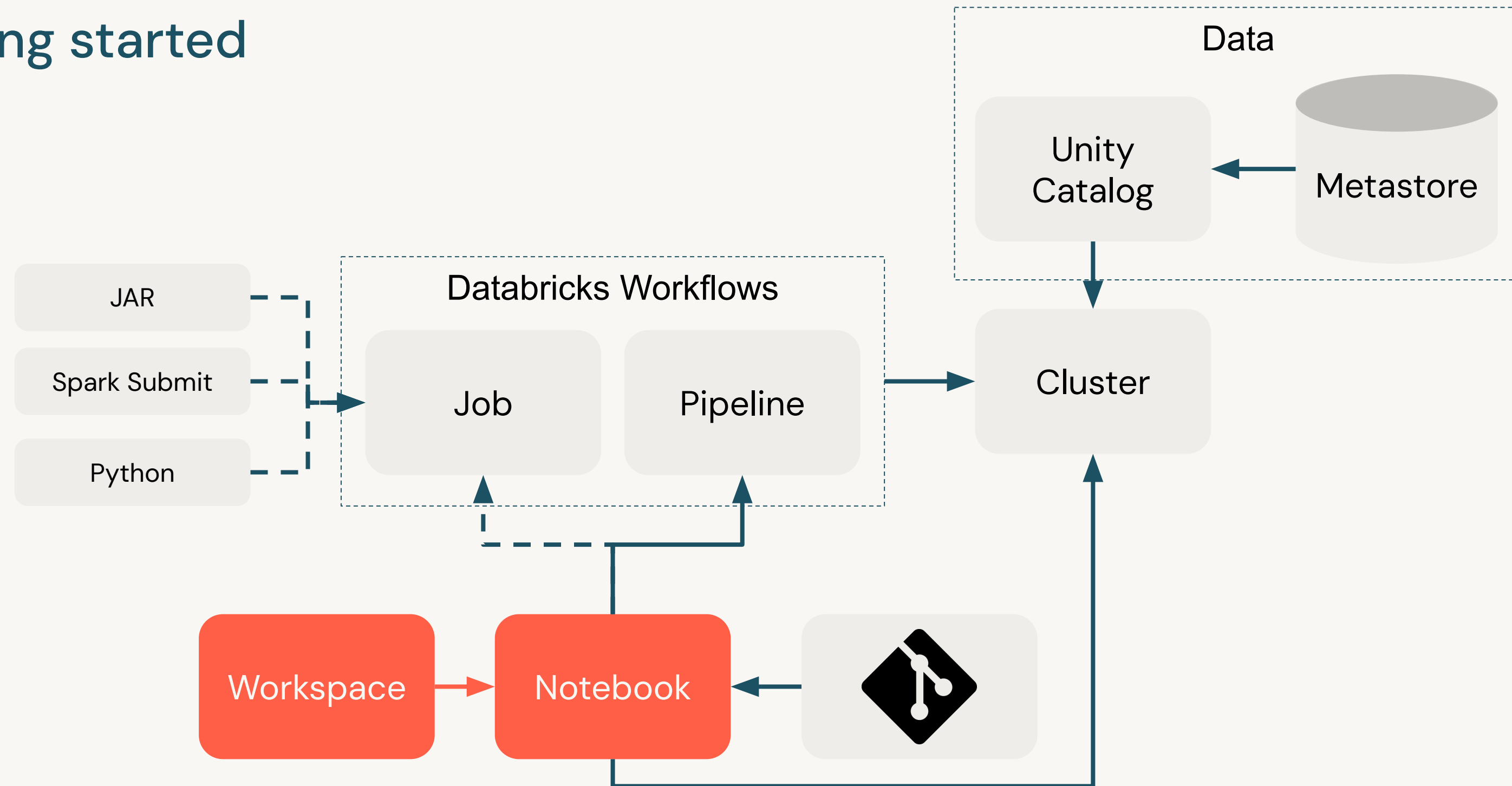
Workflow
Overview

Workflow
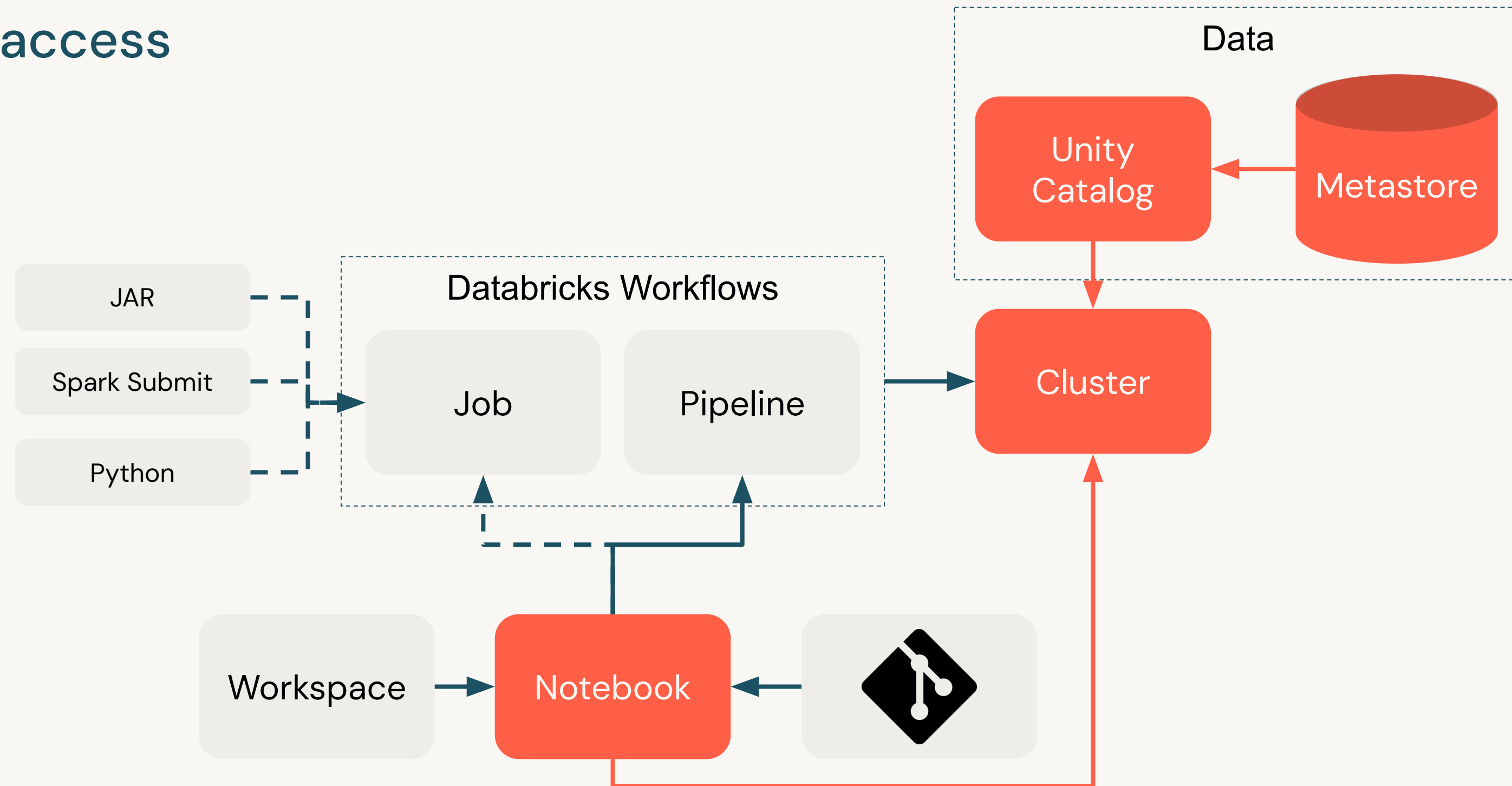
Running workloads

Workflow

Getting started

JAR

Spark Submit

Python

Databricks Workflows

Job

Pipeline

Cluster

Data

Unity Catalog

Metastore

Workspace

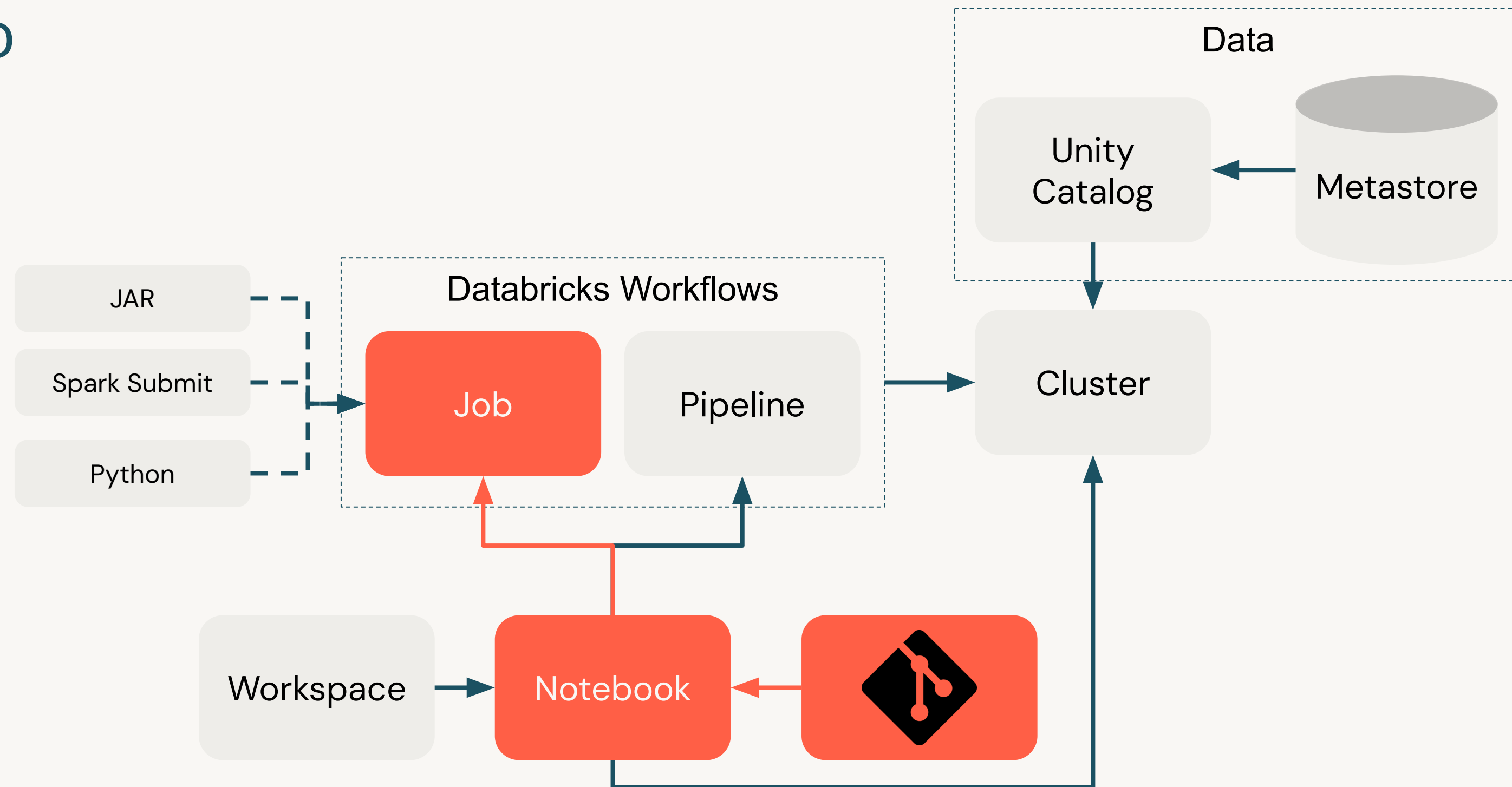Notebook

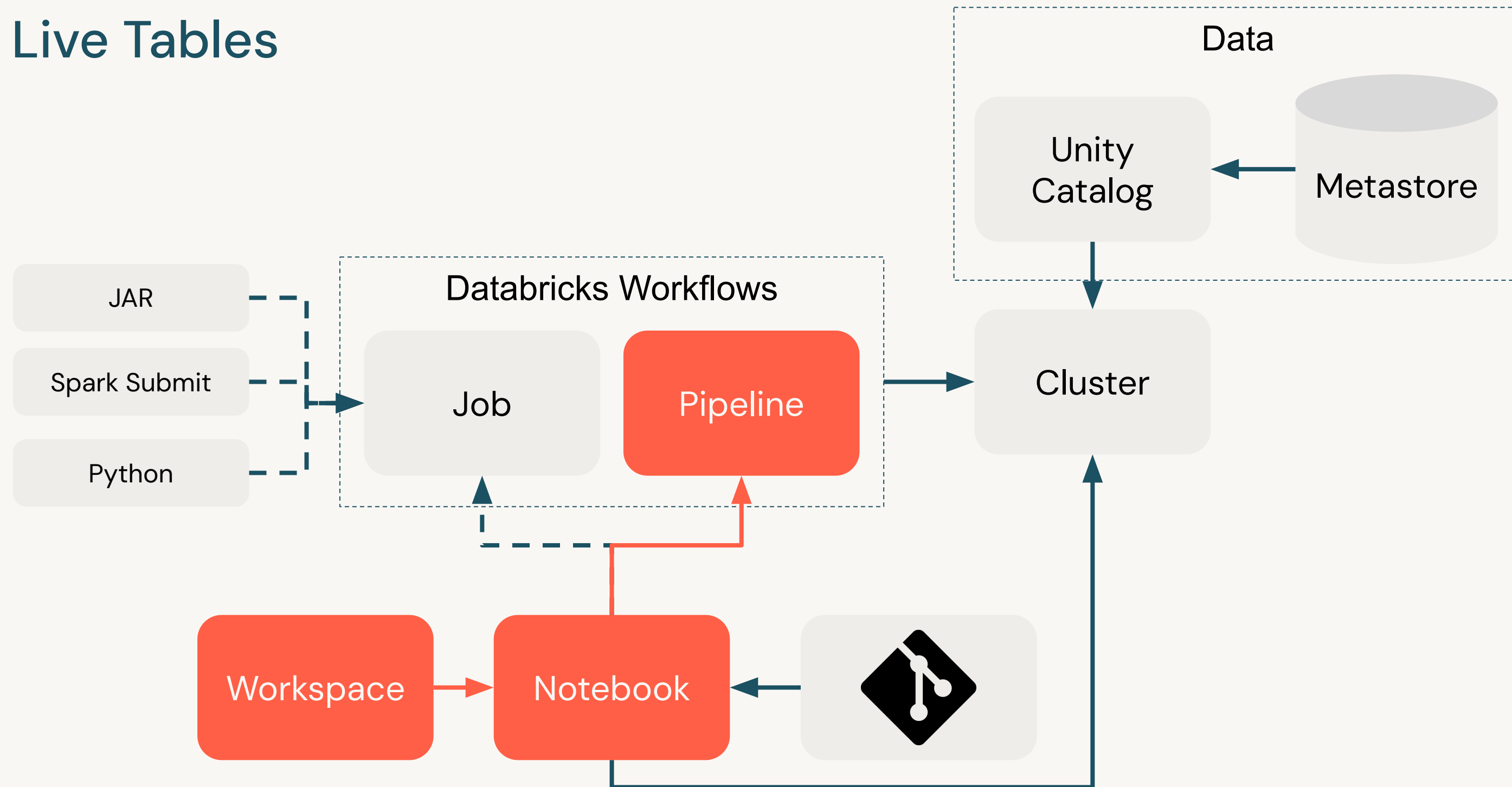19

Workflow

Data access
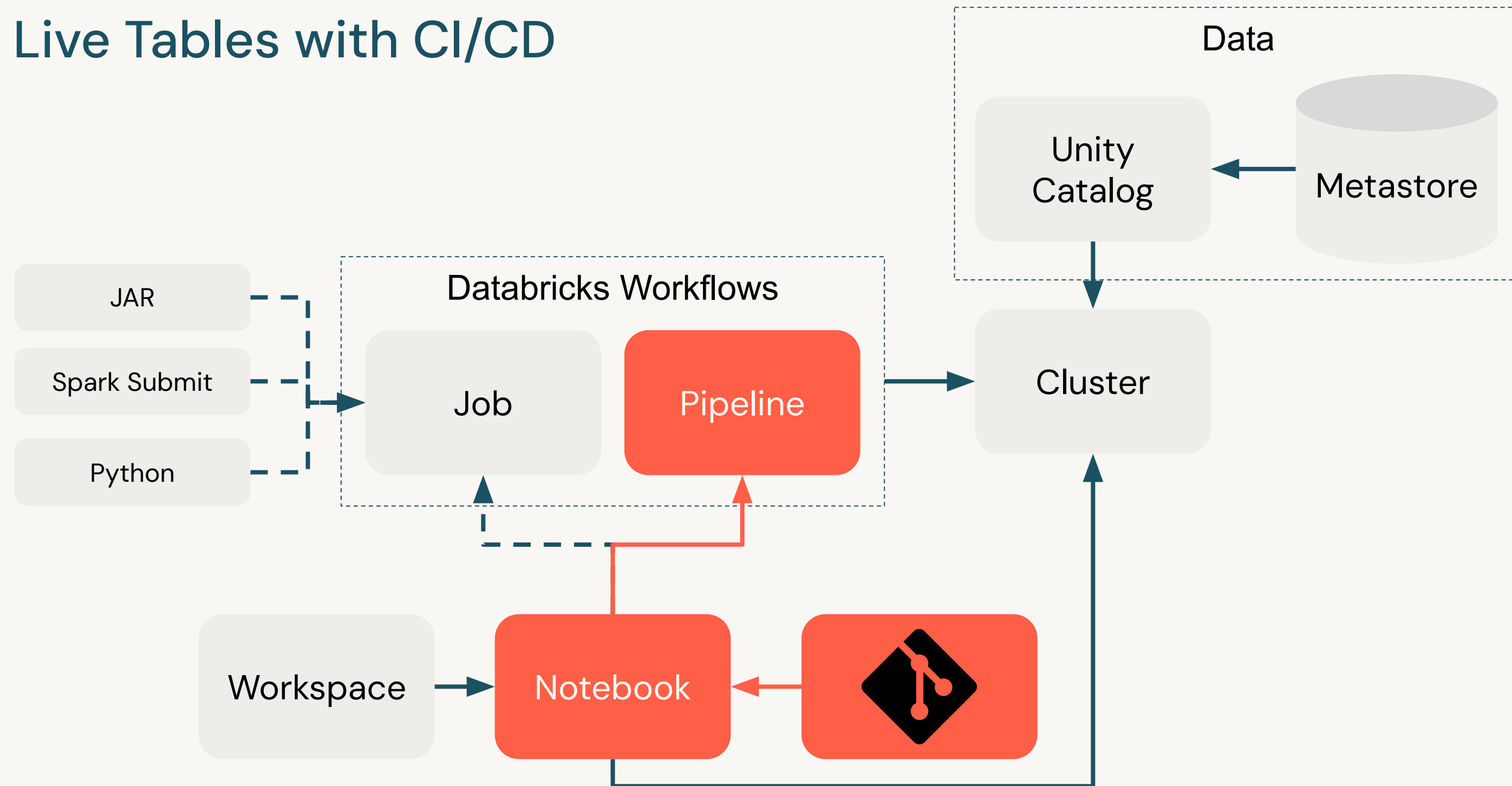
Workflow

CI/CD

Workflow
Delta Live Tables

Workflow

Delta Live Tables with CI/CD

Demo:

Deploy a Pipeline with the CLI

Lab:

Deploy Workloads with the CLI

Working with Terraform

Learning Objectives

By the end of this lesson, you should be able to:

1 Describe how to use Terraform to provision a Databricks workspace, configure data, and deploy a multi-task job

Working with Terraform

Terraform Databricks Integration

- Terraform is an **Infrastructure as a Code** tool

- It allows us to codify the infrastructure and make it **repeatable, shareable, and auditable**.

- **Databricks Terraform Provider:**

  - Manage Databricks workspaces and associated cloud infrastructure.

  - Support automation of deployment and management.

  - Support all Databricks REST APIs

Working with Terraform

Reusable infrastructure definition as a code
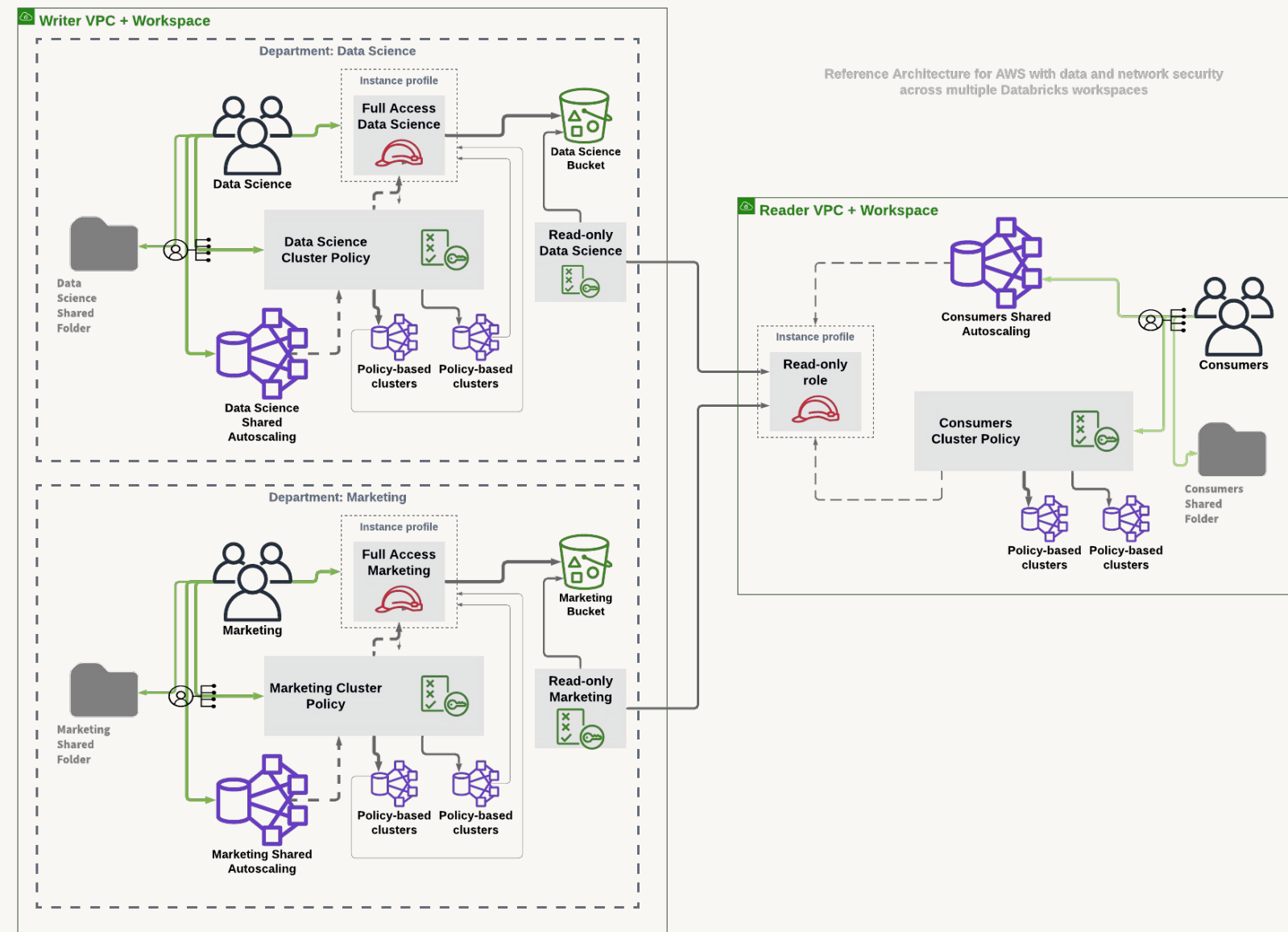

```
Databricks Terraform Provider

provider "databricks" {
}

data "databricks_current_user" "me" {}
data "databricks_spark_version" "latest" {}
data "databricks_node_type" "smallest" {
  local_disk = true
}

resource "databricks_notebook" "this" {
  path     = "${data.databricks_current_user.me.home}/Terraform"
  language = "PYTHON"
}

resource "databricks_job" "this" {
  name = "Terraform Demo
(${data.databricks_current_user.me.alphanumeric})"

  new_cluster {
    num_workers   = 1
    spark_version = data.databricks_spark_version.latest.id
    node_type_id  = data.databricks_node_type.smallest.id
  }
}

resource "databricks_catalog" "sandbox" {
  metastore_id = databricks_metastore.this.id
  name         = "sandbox"
  comment      = "this catalog is managed by terraform"
  properties = {
    purpose = "testing"
  }
}
```

Working with Terraform

Why infrastructure as Code (Terraform)?

Enable Expansion

- Allow non-technical people to set up fully consistently configured workspace in no time.

- Require less effort to maintain configurations across different teams and deployments.

Automate Infrastructure Deployment

- Make deploying and managing the infrastructure much easier.

- Have smaller, cleaner, expressive, integrated and reliable codebase.

Control Cloud Environment

- Better align with corporate cloud deployment standards.

- Ensure the necessary cost and security controls.

Working with Terraform

Cloud provider integrations

AWS

- AWS IAM EC2 instance profiles
- AWS S3 bucket mounts
- Common IAM policies helpers
- AWS KMS integration
- AWS VPC integration

Azure

- Native Azure CLI Authentication
- Azure Service Principal Authentication
- Azure Key Vault integration
- Azure Storage mounts

GCP

- Native OIDC default application credentials authentication
- GCP workspace creation

Knowledge Check

A data engineer wants to create a single refined table out of raw data from several streaming sources. Which of the following data ingestion patterns can be used to complete this task?

Select one response

A. Sequence
B. Filter
C. Fan-out
D. Funnel

Which of the following components of a job are associated with viewing run history and debugging?

Select one response

A. Task
B. Monitor
C. Cluster
D. Schedule

In what order do the following commands need to be run in order to install and configure the Databricks CLI in the Databricks workspace at the location stored in the variable `path`?

```
1. databricks configure –token
2. pip install databricks-cli
3. databricks workspace ls $path
```

Select one response

A. 3, 2, 1

B. 1, 3, 2

C. 2, 1, 3

D. 2, 3, 1

Which of the following APIs are used only by administrators to manage account users and groups?

Select one response

A. Jobs API, Clusters API, DBFS API

B. Workspace API, Libraries API, Tokens API

C. Account API, Groups API, SCIM API

D. Cluster Policies API, Instance Pools API, Global Init Scripts API, Instance Profiles API

Which of the following features are available with Databricks Terraform Integration?

Select two responses

a. Management of Databricks workspaces and associated cloud infrastructure

b. Generation of personal access tokens (PATs) for service principals

c. Support for automated deployment and management

d. Support for all account-level administrator tasks