# ****Prefect Orchestration: Detailed Technical Documentation with Azure Databricks Integration****

## ****1. Introduction****

This documentation outlines the design and implementation of data pipeline orchestration using **Prefect**, an open-source orchestration framework. The goal is to integrate Prefect with **Azure Databricks**, handle advanced orchestration features (branching, retries, error handling), and manage deployments using prefect.yml and GitHub version control.

## ****2. Core Prefect Components Explained****

### ****2.1 Flow****

A **Flow** is the core unit of orchestration in Prefect. It defines a directed graph of **Tasks**, managing the execution order and dependencies.

* Analogous to an end-to-end DAG in Airflow.
* Includes logic like branching, retries, timeouts, and parallelism.

python

CopyEdit

from prefect import flow

@flow

def my\_etl\_flow():

...

### ****2.2 Task****

A **Task** is a single operation or unit of work, defined by decorating a Python function with @task.

* Supports retries, caching, and timeout policies.
* Can be shared or reused across flows.

python

CopyEdit

from prefect import task

@task(retries=3)

def extract\_data():

...

### ****2.3 Deployment****

A **Deployment** is a registered version of a flow that can be scheduled, triggered via API, or used in CI/CD. It connects the flow to infrastructure via:

* A defined entrypoint (flow\_file:flow\_name)
* Parameters
* Work pool configuration

### ****2.4 Agent****

The **Agent** is a background process that pulls work from a work pool and executes it. It acts as the “runtime” for Prefect flows.

* Deploy on local, VM, container, or Kubernetes
* Start via CLI: prefect agent start -p <pool-name>

### ****2.5 Work Pool & Work Queue****

* **Work Pool**: Logical group that associates agents with flows
* **Work Queue**: A sub-category to organize flows (e.g., by type, priority)

Example:

bash

CopyEdit

prefect work-pool create default-pool -t process

### ****2.6 Block****

A **Block** is a reusable, encrypted storage for credentials, tokens, and configurations.

* Examples: AWS credentials, Azure tokens, GitHub tokens, Databricks configs
* Created and managed in Prefect UI or code

python

CopyEdit

from prefect.blocks.system import Secret

Secret(value="my-token").save("databricks\_token")

## ****3. Advanced Orchestration Concepts****

### ****3.1 Branching****

Used to conditionally run parts of the flow based on parameters or results.

python

CopyEdit

@flow

def conditional\_branching(condition: bool):

if condition:

task\_a()

else:

task\_b()

### ****3.2 Retries****

Configure automatic retries for transient failures using the @task(retries=n) decorator.

python

CopyEdit

@task(retries=2, retry\_delay\_seconds=30)

def flaky\_task():

...

### ****3.3 Error Handling****

Use try/except in tasks or Prefect’s on\_failure to handle exceptions gracefully.

python

CopyEdit

@task

def safe\_task():

try:

return api\_call()

except Exception as e:

logger.warning(f"Task failed: {e}")

return None

## ****4. Azure Databricks Integration****

### ****4.1 Authentication with Databricks****

Use a **DatabricksCredentials block** to securely authenticate:

python

CopyEdit

from prefect\_databricks import DatabricksCredentials

DatabricksCredentials(

databricks\_instance="https://<region>.azuredatabricks.net",

token="your-pat-token"

).save("my-dbx-creds")

### ****4.2 Run Notebook on Pre-Existing Cluster (Job Type 1)****

python

CopyEdit

@task

def run\_existing\_cluster\_job(cluster\_id: str):

creds = DatabricksCredentials.load("my-dbx-creds")

return run\_notebook\_and\_wait\_for\_completion(

notebook\_path="/Users/xyz@company.com/job1",

databricks\_credentials=creds,

cluster\_id=cluster\_id,

parameters={"input": "123"}

)

### ****4.3 Run Notebook with Autoscaling Cluster (Job Type 2)****

python

CopyEdit

@task

def run\_autoscaling\_job():

creds = DatabricksCredentials.load("my-dbx-creds")

return run\_notebook\_and\_wait\_for\_completion(

notebook\_path="/Users/xyz@company.com/job2",

databricks\_credentials=creds,

new\_cluster={

"spark\_version": "11.3.x-scala2.12",

"node\_type\_id": "Standard\_DS3\_v2",

"autoscale": {

"min\_workers": 2,

"max\_workers": 5

},

"spark\_conf": {

"spark.databricks.delta.preview.enabled": "true"

}

},

parameters={"input": "abc"}

)

## ****5. Flow Composition with Dependency and Logic****

python

CopyEdit

@flow

def databricks\_orchestration(run\_autoscale: bool, cluster\_id: str = None):

if run\_autoscale:

run\_autoscaling\_job()

else:

if cluster\_id:

run\_existing\_cluster\_job(cluster\_id)

## ****6. Deployment via**** prefect.yml

### ****Sample: Autoscaling Deployment****

yaml

CopyEdit

name: autoscaling-dbx

entrypoint: flows/dbx\_flow.py:databricks\_orchestration

description: Trigger Databricks notebook with new cluster

parameters:

run\_autoscale: true

work\_pool:

name: default-pool

work\_queue\_name: default

infra\_overrides:

env:

DATABRICKS\_TOKEN: "{{ prefect.blocks.secret.databricks\_token }}"

### ****Sample: Fixed Cluster Deployment****

yaml

CopyEdit

name: existing-cluster-dbx

entrypoint: flows/dbx\_flow.py:databricks\_orchestration

description: Trigger Databricks notebook on existing cluster

parameters:

run\_autoscale: false

cluster\_id: "0719-020255-breeze194"

work\_pool:

name: default-pool

work\_queue\_name: default

## ****7. GitHub Version Control & PR Workflow****

### ****7.1 Suggested Repo Structure****

CopyEdit

project-root/

├── flows/

│ └── dbx\_flow.py

├── prefect.yml

├── .gitignore

├── README.md

### ****7.2 Version Control Workflow****

1. Create a feature branch:

bash

CopyEdit

git checkout -b feature/dbx-prefect

1. Add and commit code:

bash

CopyEdit

git add .

git commit -m "Add Databricks flow with autoscaling support"

1. Push and raise a Pull Request:

bash

CopyEdit

git push origin feature/dbx-prefect

1. In PR:
   * Add reviewers
   * Ensure correct parameter handling
   * Validate prefect.yml
   * Confirm error handling logic

## ****8. Best Practices****

* **Secrets**: Use blocks for all credentials
* **Logging**: Add detailed logs inside tasks
* **Flow IDs**: Use unique deployment names
* **Branching**: Prefer explicit branching over nested ifs
* **Testing**: Use local testing via python flow.py

## ****9. References****

* Prefect Docs
* Prefect Databricks Plugin
* [Azure Databricks API](https://learn.microsoft.com/en-us/azure/databricks/dev-tools/api/)