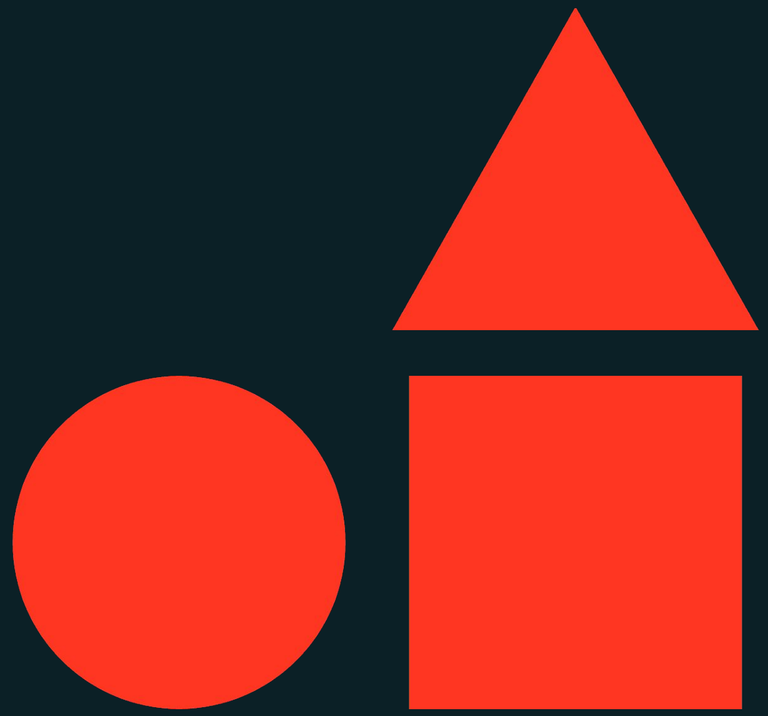
databricks

Automated Deployment with Databricks Asset Bundles (DABs)



Databricks Academy

Course Learning Objectives

- Review the core principles of DevOps, including continuous integration and continuous deployment (CI/CD) and testing, and how they apply to Databricks projects.
- Evaluate the tools available for continuous deployment (CD) of Databricks projects, including the Databricks REST API, SDK, and CLI.
- Explore the benefits of Databricks Asset Bundles for continuous deployment and their folder structure and main components.
- Learn how to validate, deploy, and run Databricks Asset Bundles using the Databricks CLI to multiple target environments using different configurations.
- At a high level, learn how to automate deployment pipelines for Databricks projects using GitHub Actions with Databricks Asset Bundles to streamline the CI/CD process.
- Get a quick introduction into how to use Visual Studio Code for developing, testing, and deploying Databricks Asset Bundles.

Agenda

Course Name
• DevOps and CI/CD Review
• Deployment with Databricks Asset Bundles (DABs)
• Doing More with Databricks Asset Bundles

Course Prerequisites (REQUIRED)

Proficient Knowledge of the Databricks Platform

- Databricks Workspaces
- Apache Spark
- Delta Lake and the Medallion Architecture
- Unity Catalog
- Delta Live Tables
- Workflows
- Git Folders

Course Prerequisites (REQUIRED)

Proficient Knowledge of the Databricks Platform

- Databricks Workspaces
- Apache Spark
- Delta Lake and the Medallion Architecture
- Unity Catalog
- Delta Live Tables
- Workflows
- Git Folders

Experience Ingesting and Transforming Data

- Familiarity with PySpark for data processing and DataFrame manipulations.
- Experience in writing intermediate-level queries using SQL
- Proficient in writing Python, including functions and classes.

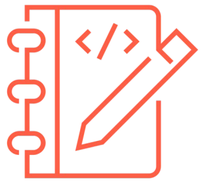
Course Prerequisites (REQUIRED)

Proficient Knowledge of the Databricks Platform

- Databricks Workspaces
- Apache Spark
- Delta Lake and the Medallion Architecture
- Unity Catalog
- Delta Live Tables
- Workflows
- Git Folders

Experience Ingesting and Transforming Data

- Familiarity with PySpark for data processing and DataFrame manipulations.
- Experience in writing intermediate-level queries using SQL
- Proficient in writing Python, including functions and classes.



Knowledge of DevOps and CI/CD Principles

- Familiarity with DevOps practices and the principles behind continuous integration and continuous delivery/deployment (CI/CD).
- Knowledge of Git version control.
- DevOps Essentials for Data Engineering Course

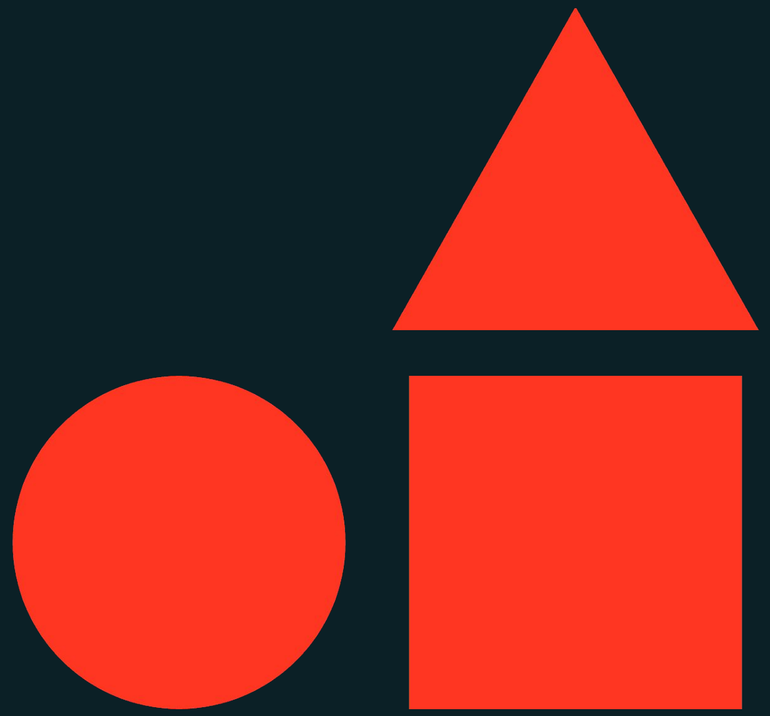
Lab Exercise Environment

Technical Details

- Your lab environment is provided by Vocareum.
- It will open in a new tab.
- It has been configured with the permissions and resources required to accomplish the tasks outlined in the lab exercise.
- Third party cookies must be enabled in your browser for Vocareum's user experience to work properly.
- Make sure to enable pop ups!

databricks

DevOps and CI/CD Review



Automated Deployment with Databricks Asset Bundles

Section Learning Objectives

- Review the core concepts and benefits of CI/CD in modern software development, with an emphasis on automation, continuous integration, and continuous delivery of data pipelines.
- Evaluate tools for continuous deployment of Databricks projects, including the Databricks REST API, SDK, and CLI, and understand their use in automating deployment processes.

Agenda

DevOps and CI/CD Review
• DevOps Review
• Continuous Integration and Continuous Deployment/Delivery (CI/CD) Review
• Course Setup and Authentication

databricks

DevOps and CI/CD Review

LECTURE

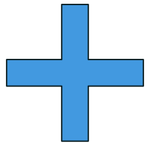
DevOps Review

DevOps Review

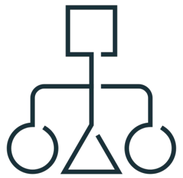
What is DevOps?

DevOps

Software Engineering Best Practices



IT Operations

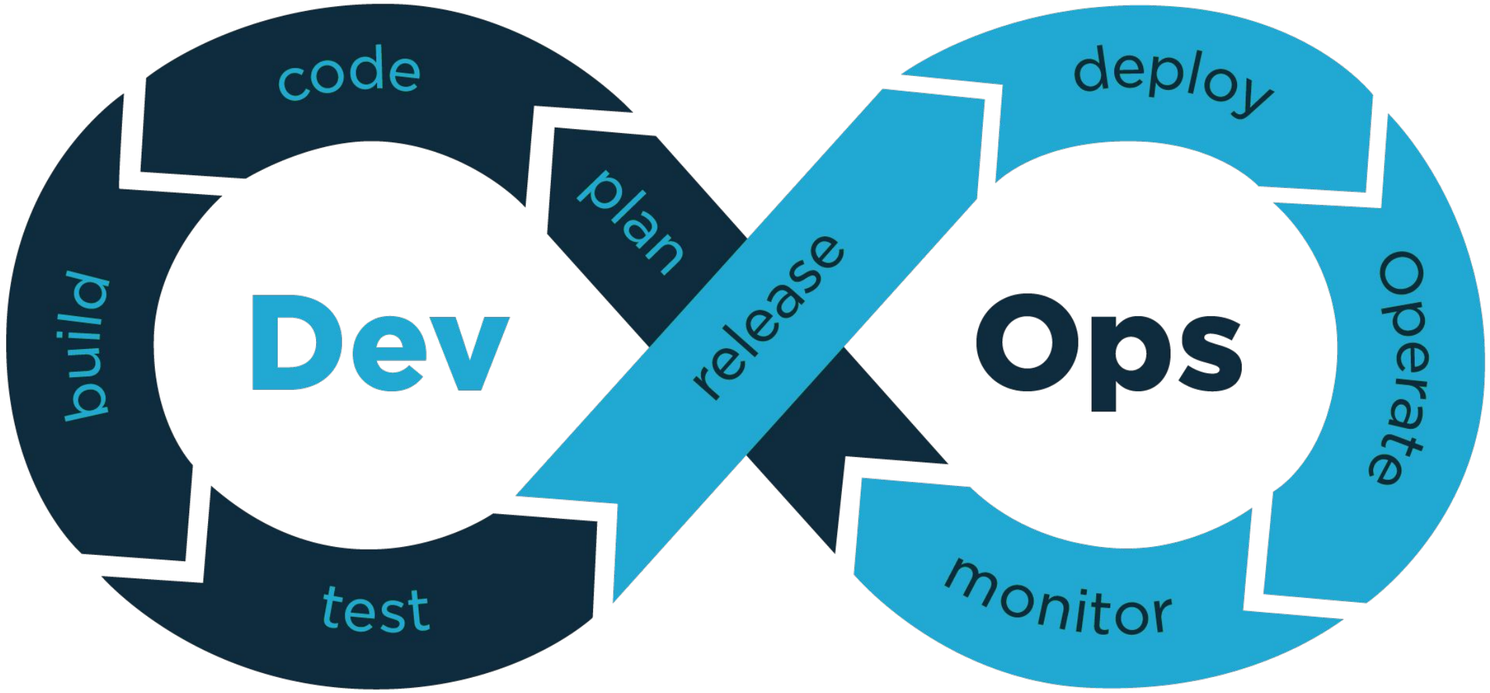


Fosters **collaboration** between development and operations teams to **automate workflows** and **streamline processes**.

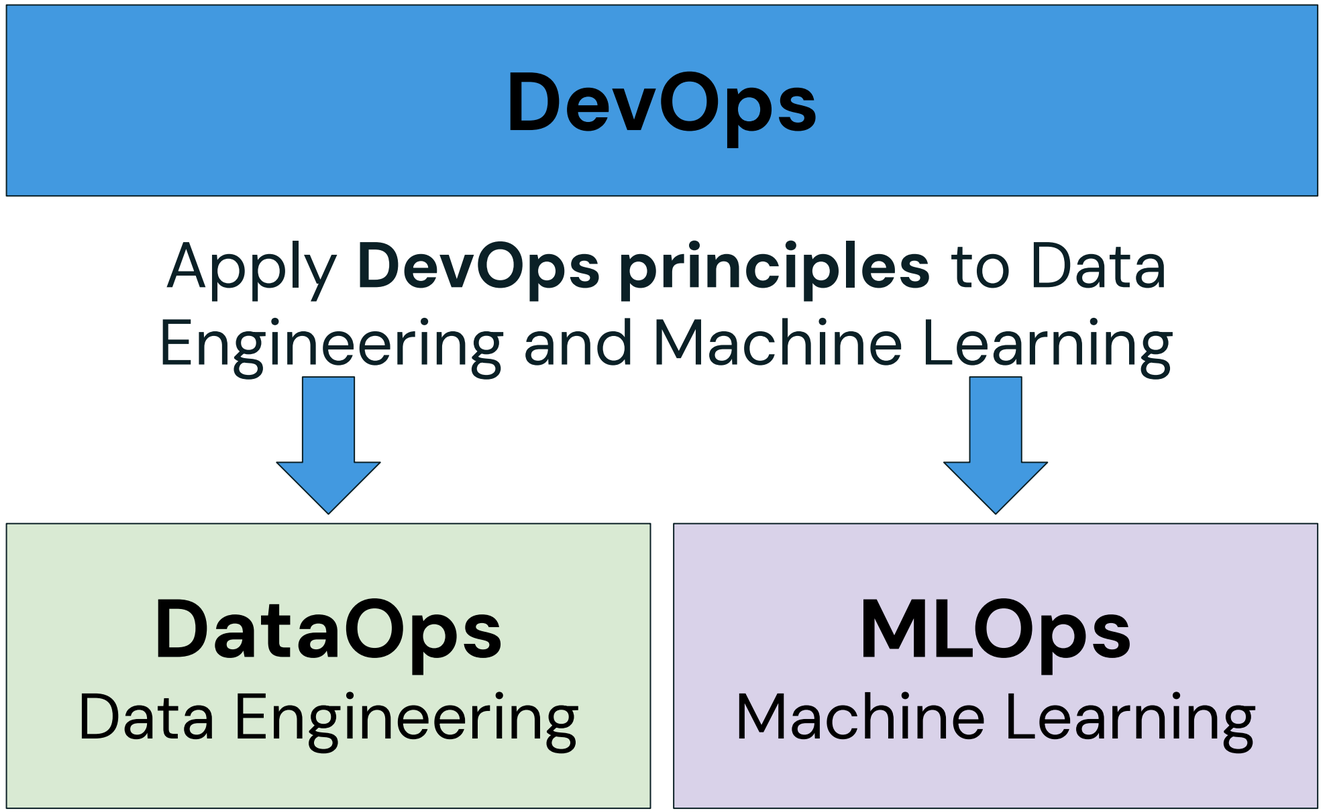
Key benefits include

- faster deployments
- improved collaboration
- enhanced reliability
- better scalability

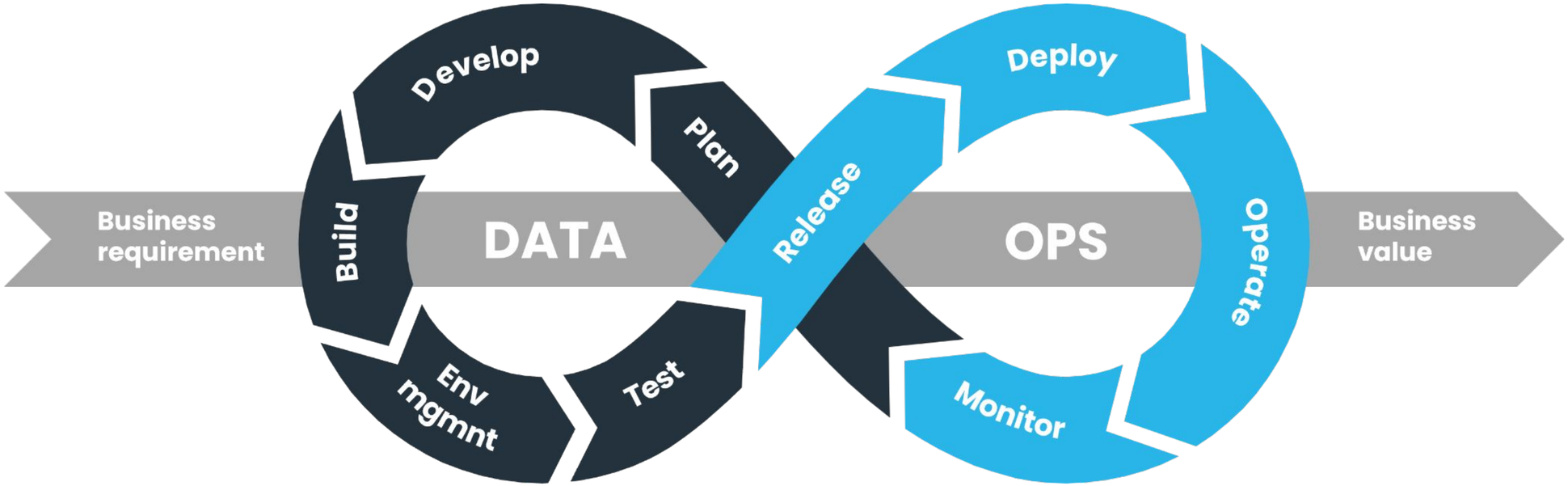
DevOps is a process for **continuously** integrating, testing and deploying your code.



DevOps For Data Engineering and Machine Learning



DataOps = DevOps for Data Engineering



You want to think about how **DevOps** principles and culture can be applied to your **Data Engineering pipelines**.

databricks

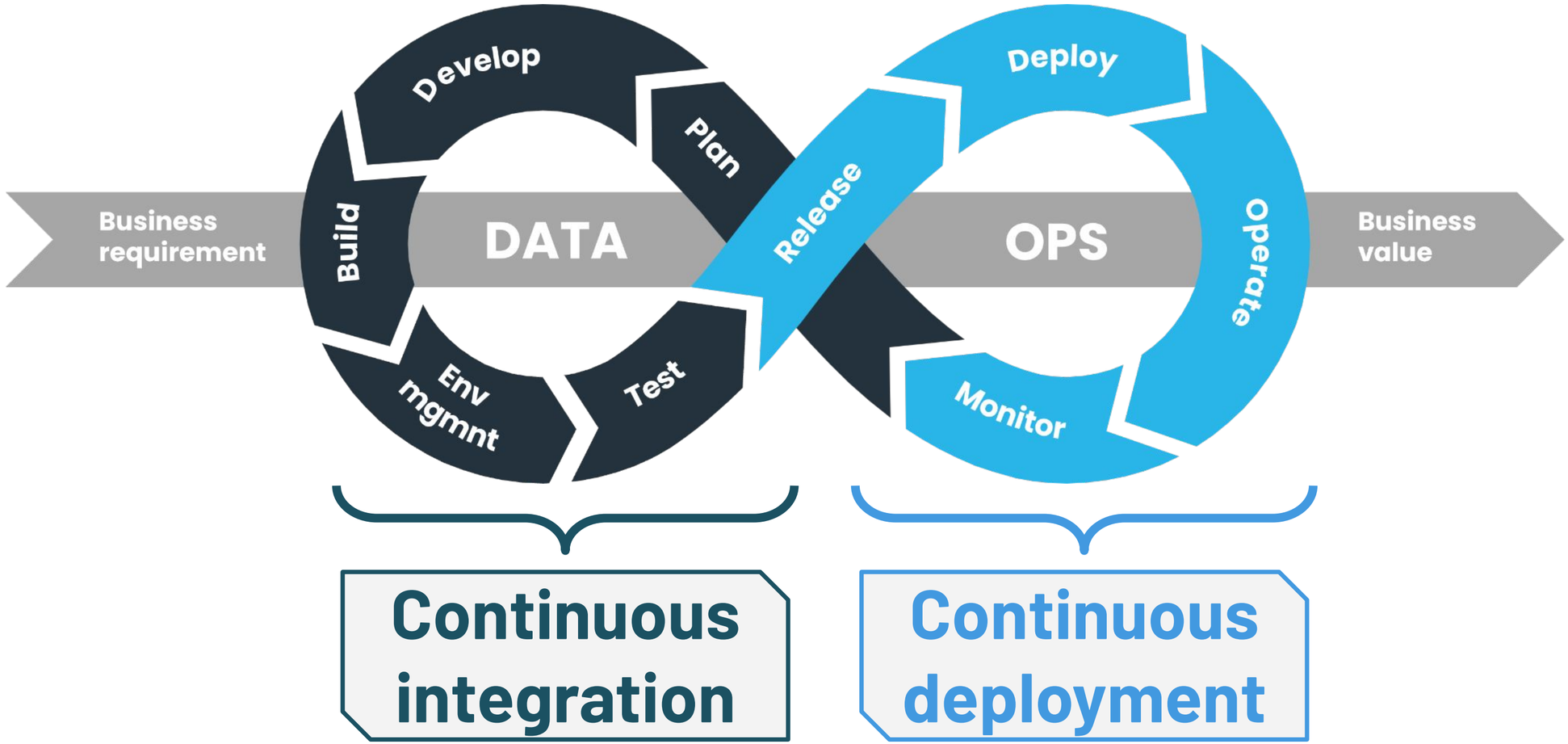
DevOps and CI/CD Review

LECTURE

Continuous Integration and Continuous Deployment/Delivery (CI/CD) Review

Review the Role of CI/CD in DevOps

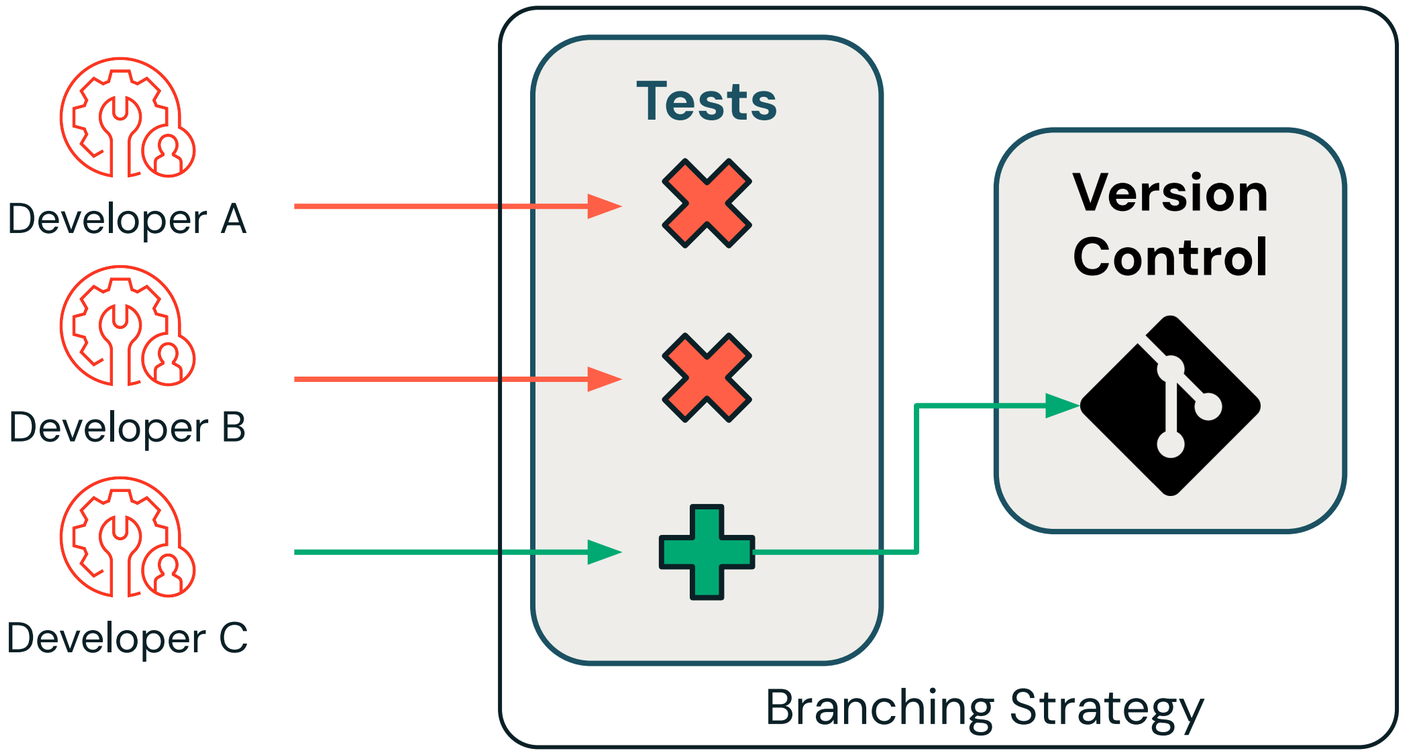
Continuous Integration (CI) and Continuous Deployment/Delivery (CD)



Review the Role of CI/CD in DevOps

Continuous Integration (CI) High Level Overview

CI involves regularly **merging code** changes from **multiple contributors** into a **central repository** and running **automated tests** to ensure code quality.

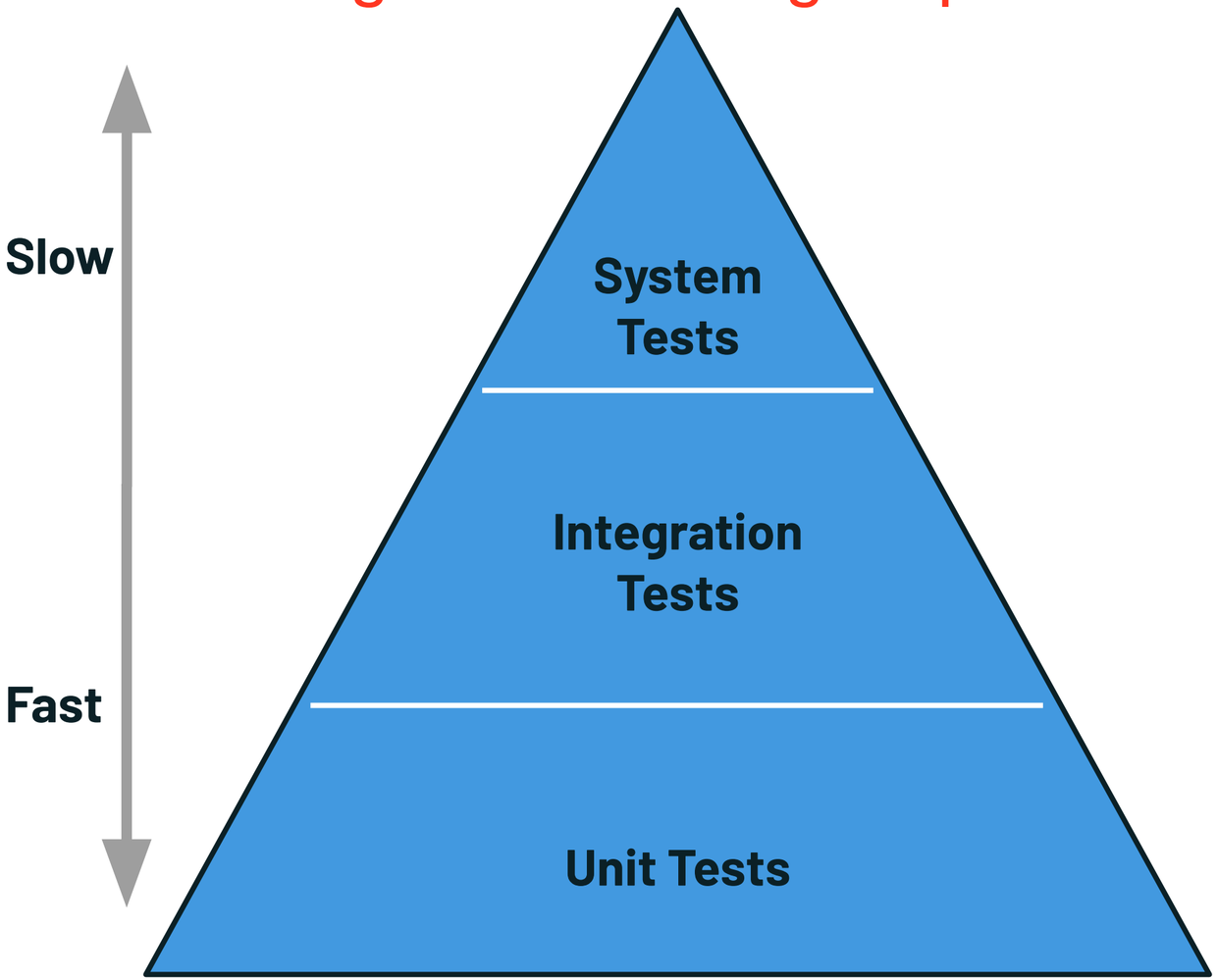


Benefits

- Early Detection of Issues
- Faster Development Cycle
- Improved Collaboration and Code Quality
- Automated Testing and Validation

Review the Role of CI/CD in DevOps

Review High-level Testing Steps



- Test the entire application, ensuring that all parts function together in a real-world scenario
- **Ex:** End to end data pipeline in a Workflow

- Test the interaction between different components or systems
- **Ex:** Notebooks / DLT/ Jobs interactions

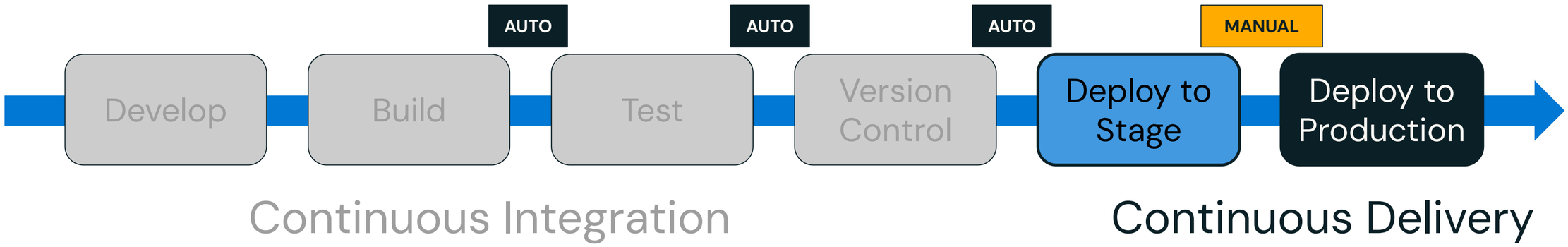
- Test **individual** functions or methods in isolation
- Fast, low cost, high coverage, and automated
- **Ex:** Custom pyspark functions

Review the Role of CI/CD in DevOps

Continuous Delivery/Deployment (CD) Overview

Continuous Delivery (CD):

Automatically pushing changes to staging/pre-production environments **with the ability to manually deploy to production** at any time.



Review the Role of CI/CD in DevOps

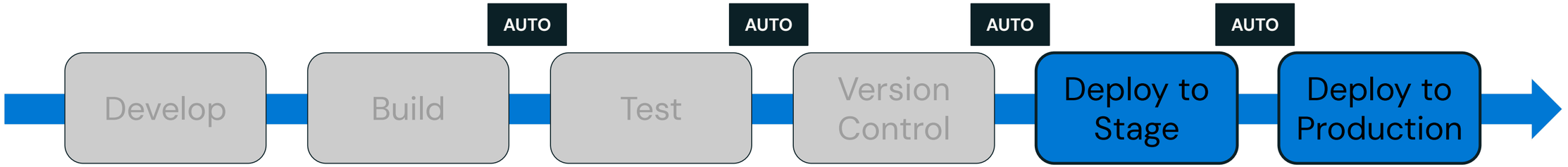
Continuous Delivery/Deployment (CD) Overview

Continuous Delivery (CD):

Automatically pushing changes to staging/pre-production environments with the ability to manually deploy to production at any time.

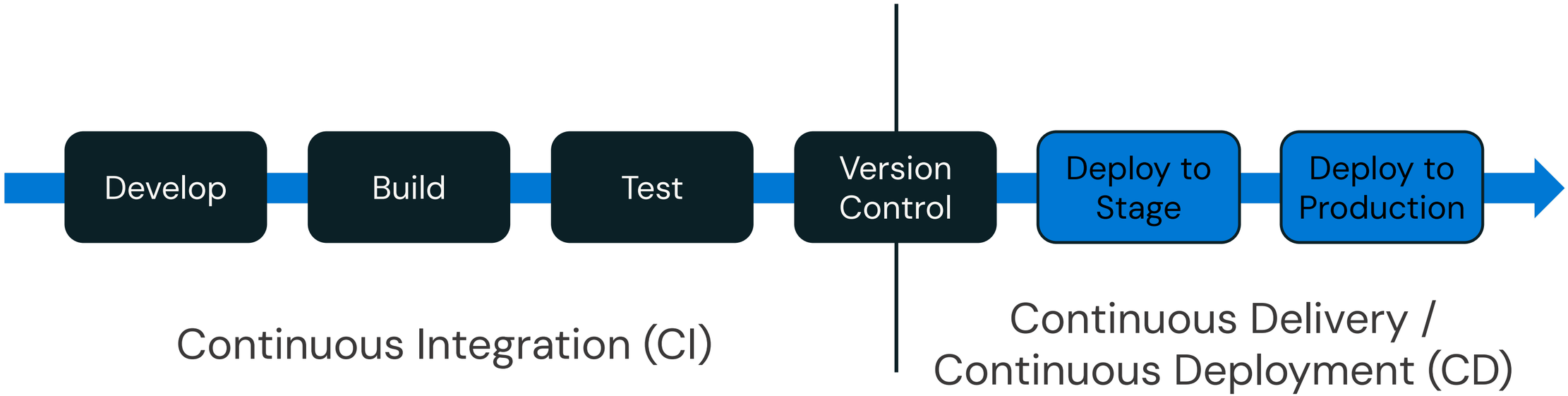
Continuous Deployment (CD):

Fully automated process where each change passing tests is immediately deployed to **production**.



Review the Role of CI/CD in DevOps

High-Level CI/CD Workflow Overview



Isolating Environments for CI/CD

Workspaces

Utilizing multiple workspaces, one for each environment

DEV

STAGE

PROD

Catalogs

Utilizing multiple catalogs, one for each environment

Setting Up Your Data for CI/CD

Dev Data

- Often a Small Static Subset of Production Data
- Can be Anonymized or Synthetic Datasets
- Supports Rapid Development and Testing
- Ensures Privacy and Data Integrity

Stage Data

- Staging Data Mirrors Production Structure & Volume, Typically Static
- Can be Anonymized or Scrubbed Sensitive Information
- Ensures Realistic Testing and Validation

Prod Data

- Production Data: Live & Fully Operational
- Contains Real User Data
- Continuously Updated
- Requires High Security, Privacy, & Compliance Standards

Deployment Tools in Databricks Overview

REST API

- Provides direct access to Databricks functionality through HTTP requests
- Requires **manual construction** of HTTP requests and handling responses

Databricks SDK

SDKs for:

- Python
- Java
- Go
- R

Method used in this course

Databricks CLI

- Easy-to-use interface for automation from terminal, command prompt, or bash scripts.
- Use **Databricks Asset Bundles (DABs)** within the CLI to write infrastructure as code for Databricks

Using the Databricks CLI

Connection and Authentication

Web Terminal

- You can use the Databricks web terminal to run **Databricks CLI** commands
- Enables you to run **shell commands** in Databricks
- Uses the **latest version** of the Databricks CLI by default
- **Authentication** based on the current user
- Must be **enabled**

Using the Databricks CLI

Connection and Authentication

Web Terminal

- You can use the Databricks web terminal to run **Databricks CLI** commands
- Enables you to run **shell commands** in Databricks
- Uses the **latest version** of the Databricks CLI by default
- **Authentication** based on the current user
- Must be **enabled**

VS Code

- You can use VSCode to install and use the **Databricks CLI**
- You must **authenticate** to Databricks
- You can use the **Databricks VSCode Extension** for additional features

Using the Databricks CLI

Connection and Authentication

Web Terminal

- You can use the Databricks web terminal to run **Databricks CLI** commands
- Enables you to run **shell commands** in Databricks
- Uses the **latest version** of the Databricks CLI by default
- **Authentication** based on the current user
- Must be **enabled**

VS Code

- You can use VSCode to install and use the **Databricks CLI**
- You must **authenticate** to Databricks
- You can use the **Databricks VSCode Extension** for additional features

Databricks Notebook

- You can run shell commands with **%sh** in a Databricks notebook.
- You can install and use the Databricks CLI
- You must **authenticate** with a token to utilize the CLI

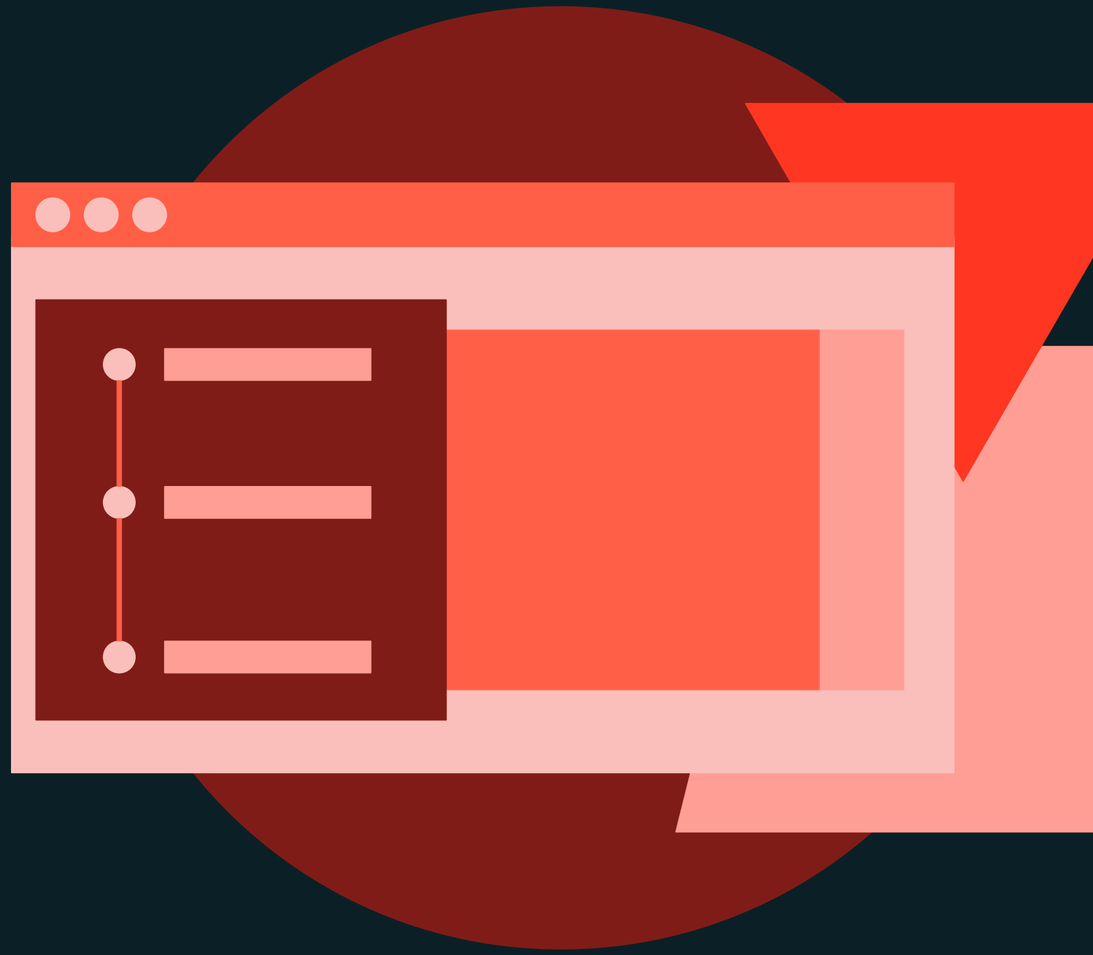
Main method used in this course

databricks

DevOps and CI/CD Review

DEMONSTRATION

Course Setup and Authentication

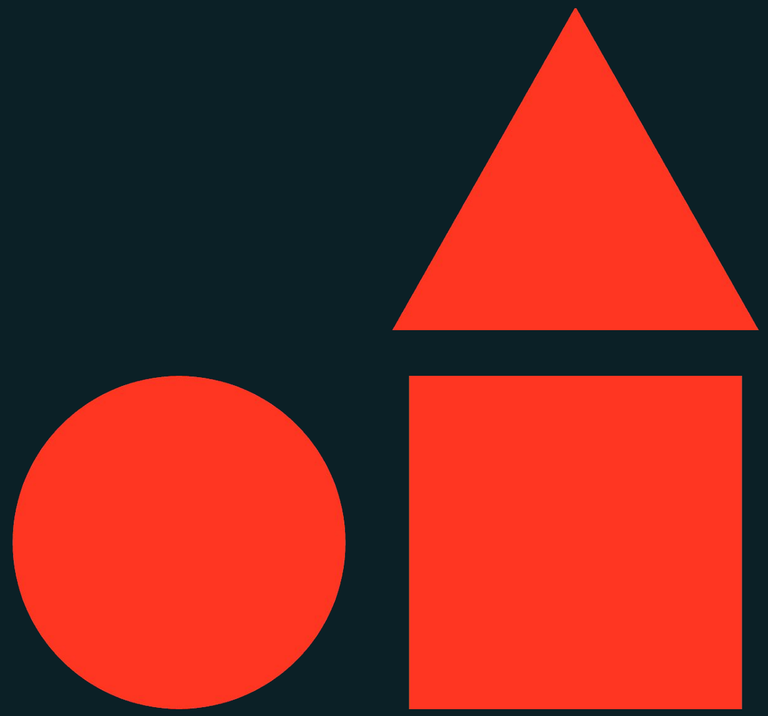


You must execute the following notebook to set up your environment for the remaining demonstrations and labs.

Notebook: ./0 - REQUIRED - Course Setup and Authentication

databricks

Deployment with Databricks Asset Bundles (DABs)



Automated Deployment with Databricks Asset Bundles

Section Learning Objectives

- Discuss the components of Databricks projects and how they are integrated into a CI/CD pipeline for seamless deployment and management.
- Understand the benefits and core components of Databricks Asset Bundles (DABs).
- Learn how to validate, deploy, and execute Databricks Asset Bundles effectively.
- Explore variable substitutions and environment configuration overrides based on the target environment.
- Learn about Databricks Asset Bundle templates.
- Apply a CI/CD pipeline using a Databricks project with Databricks Asset Bundles for streamlined deployment.

databricks

Deployment with Databricks Asset Bundles (DABs)

LECTURE

Deploying Databricks Projects

Deploying Databricks Projects

Typical Databricks Projects

Consist of a variety of components

Code
Notebooks, Python .whl, JAR, dbt, etc.

Execution Environment
Databricks Workspace, compute configuration

Resources:
Databricks Workflows, MLflow Tracking Server and Registry, Delta Live Tables…

Projects produce a variety of data products

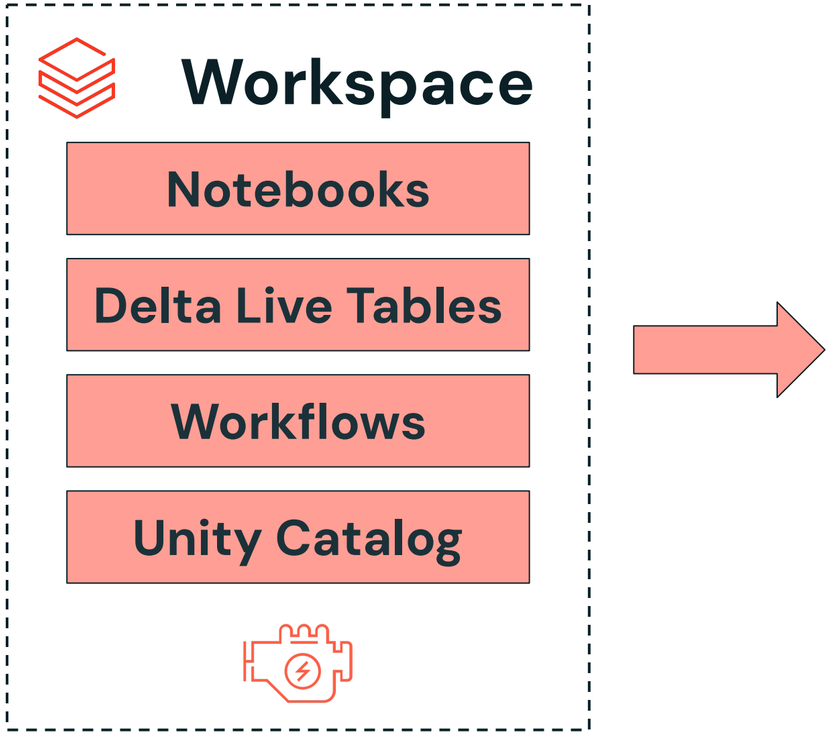
- tables
- pipelines
- jobs
- machine learning models
- dashboards
- call external services
- Etc.

The deliverable determines the components

- A **simple report** might consist of a notebook running on single node compute
- A **full MLOps pipeline** would require MLflow, Feature Store, and Model Serving components

Deploying Databricks Projects

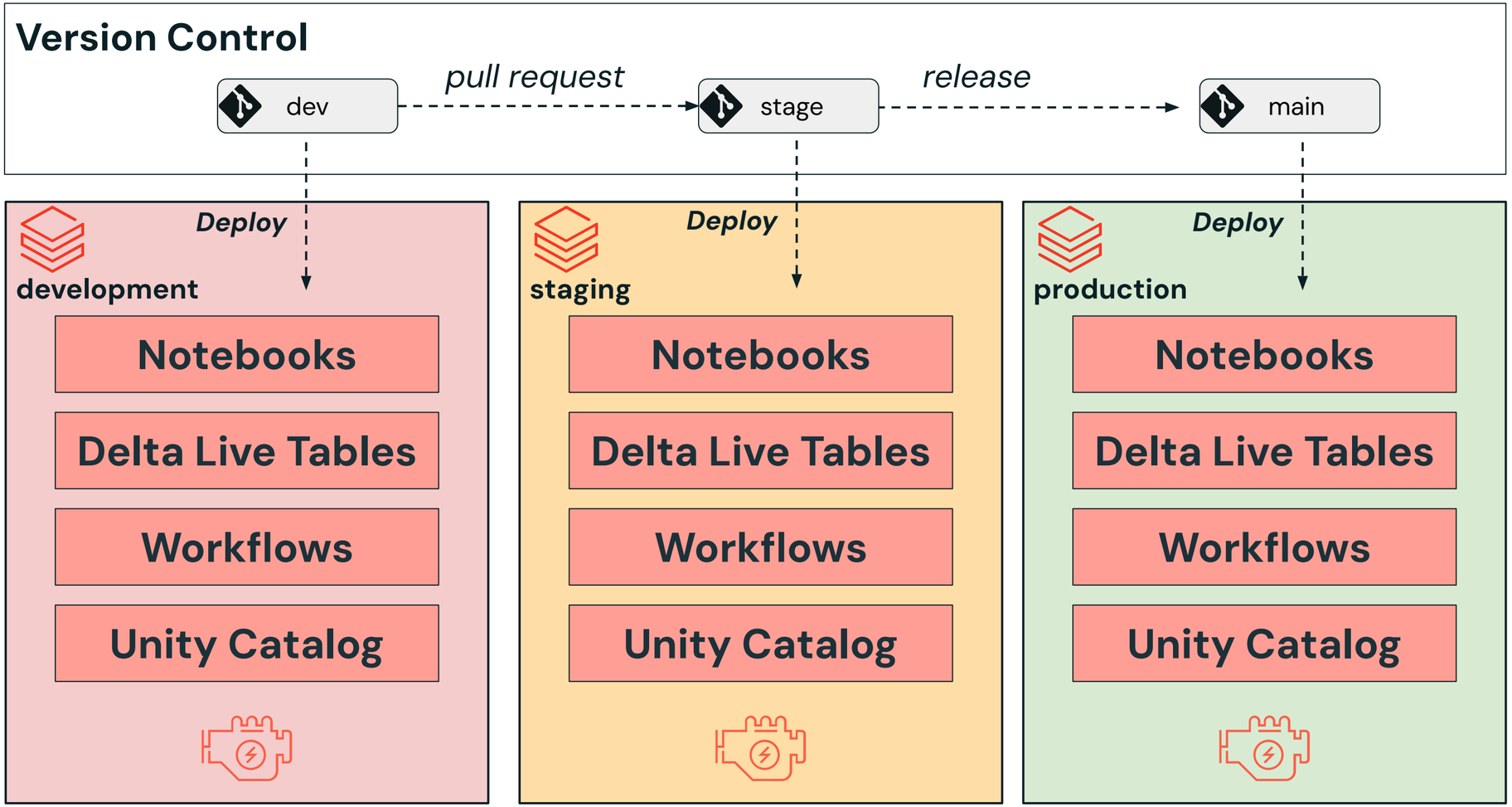
Simple Example Project Components



- **Code**
  - Python/SQL/R notebooks
- **Resources**
  - Workflow
  - Delta Live Tables (DLT)
- **Execution Environment**
  - Databricks Workspace
  - Unity Catalog
  - Compute configuration(s)

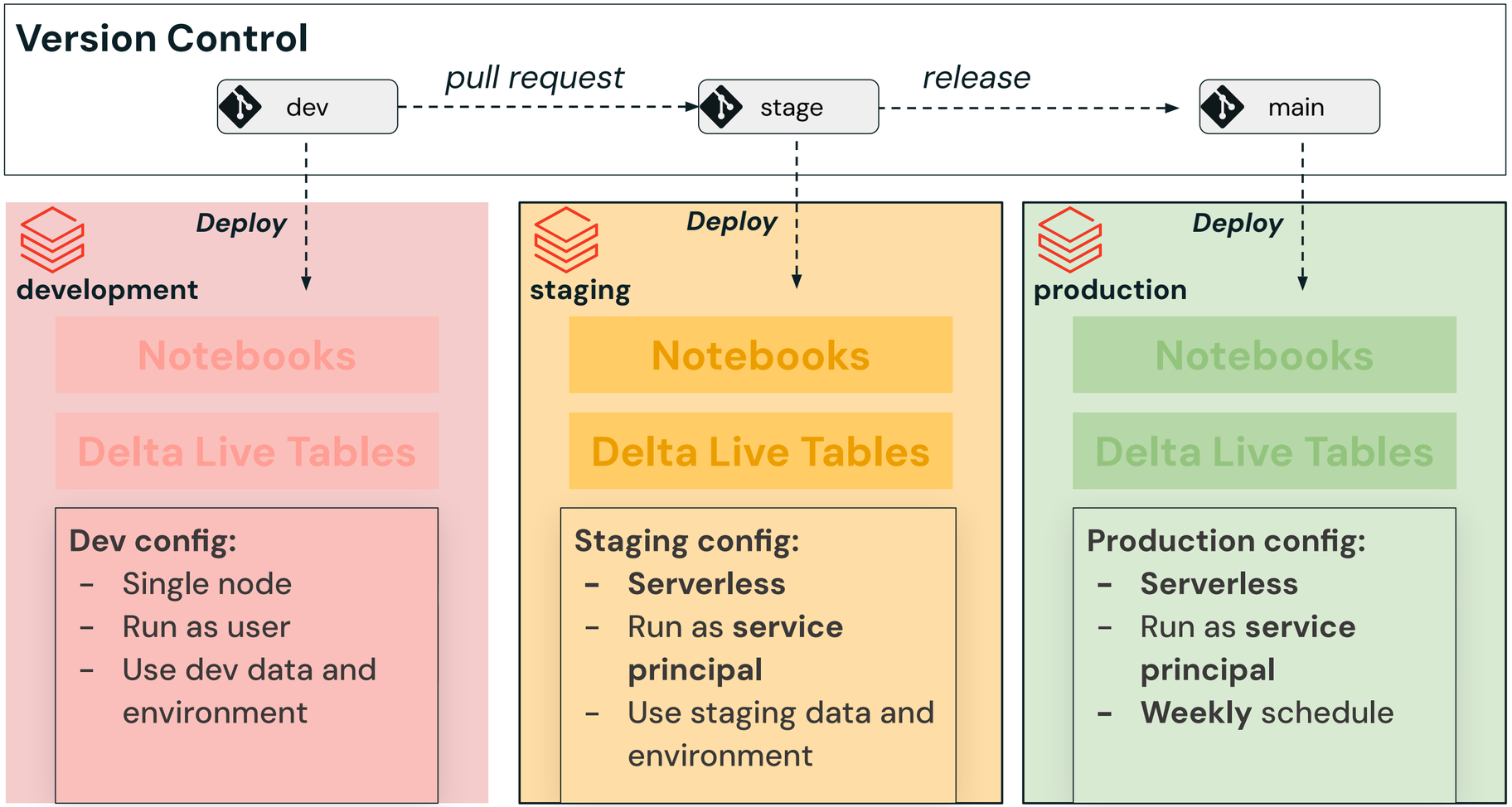
Deploying Databricks Projects

CI/CD Journey to Production



Deploying Databricks Projects

CI/CD Journey to Production



Deploying Databricks Projects

How do we orchestrate this CI/CD journey?

Manually

Using the **UI**

PROS:

- Easy to learn, high level

CONS:

- Time consuming
- Error prone
- Not a viable option for the CI/CD process

Programmatically

Databricks **REST API** or **SDK**

PROS:

- Low level control

CONS:

- Lots of APIs to have to learn
- Lots of classes to learn
- Time consuming to automate the entire CI/CD process

Terraform

Databricks **Terraform** provider

PROS:

- Very powerful and expressive
- Admin tool of choice for configuration

CONS:

- Can be challenging to learn for data scientists and engineers

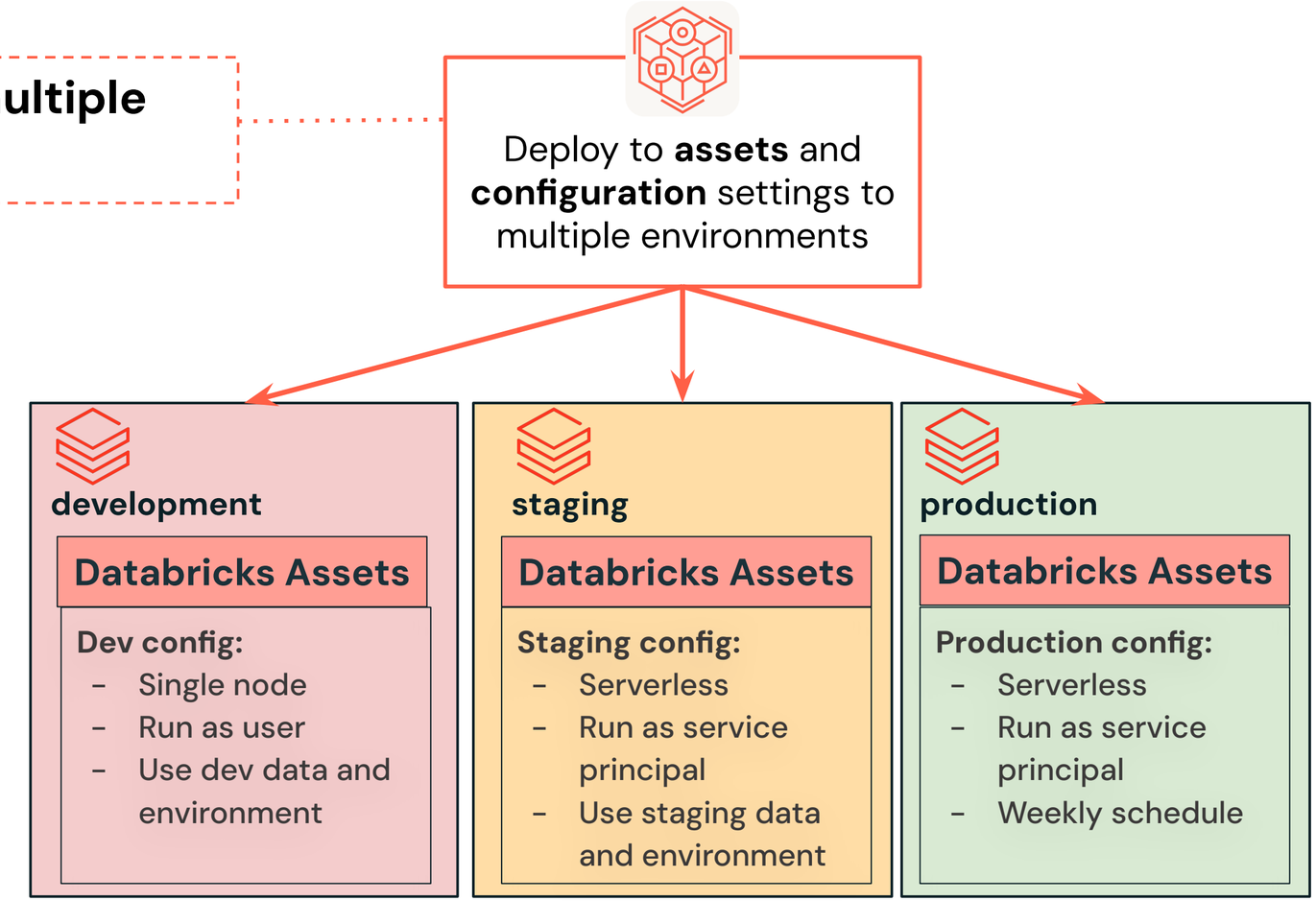
Deploying Databricks Projects

How Can the CI/CD Process be Simplified?

How to write code once, and then deploy to multiple environments easily?

What if we could:

- Co-version code with all configurations in a simple, easy to understand format like YAML?
- Define Databricks resources using existing REST API parameters?
- Ensure user isolation during deployment?
- Specify environment-based overrides and variables?



Deploying Databricks Projects

Introducing Databricks Asset Bundles (DABs)!

Databricks Asset Bundles (DABs)

A tool to facilitate the adoption of software engineering best practices, including source control, code review, testing, and **continuous integration and delivery (CI/CD), for your data and AI projects**

databricks

Deployment with Databricks Asset Bundles (DABs)

LECTURE

Introduction to Databricks Asset Bundles (DABs)

Introduction to DABs

Write code once, deploy everywhere

What are Databricks Asset Bundles (DABs)?

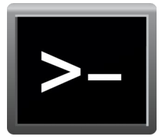
DABs use **YAML files** to specify the artifacts, resources, and configurations of a Databricks project.

Introduction to DABs

Write code once, deploy everywhere

What are Databricks Asset Bundles (DABs)?

DABs use **YAML files** to specify the artifacts, resources, and configurations of a Databricks project.



How do Databricks Asset Bundles work?

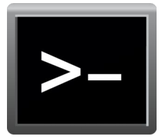
The new **databricks CLI** has specific bundle commands to **validate**, **deploy** and **run** Databricks Asset Bundles using a bundle YAML file

Introduction to DABs

Write code once, deploy everywhere

What are Databricks Asset Bundles (DABs)?

DABs use **YAML files** to specify the artifacts, resources, and configurations of a Databricks project.



How do Databricks Asset Bundles work?

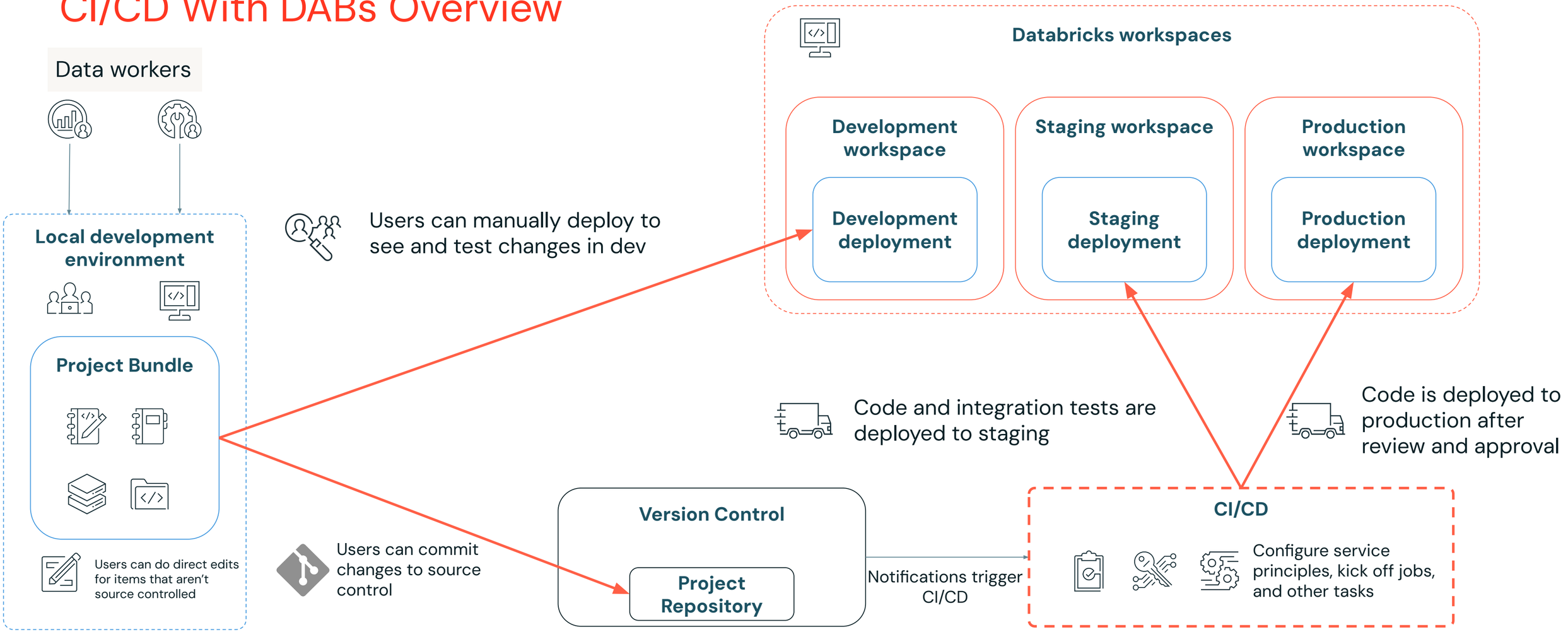
The new **databricks CLI** has specific bundle commands to **validate**, **deploy** and **run** Databricks Asset Bundles using a bundle YAML file

Where are Databricks Asset Bundles used?

Bundles are extremely useful during **development and CI/CD processes** to deploy Databricks assets to specified environments

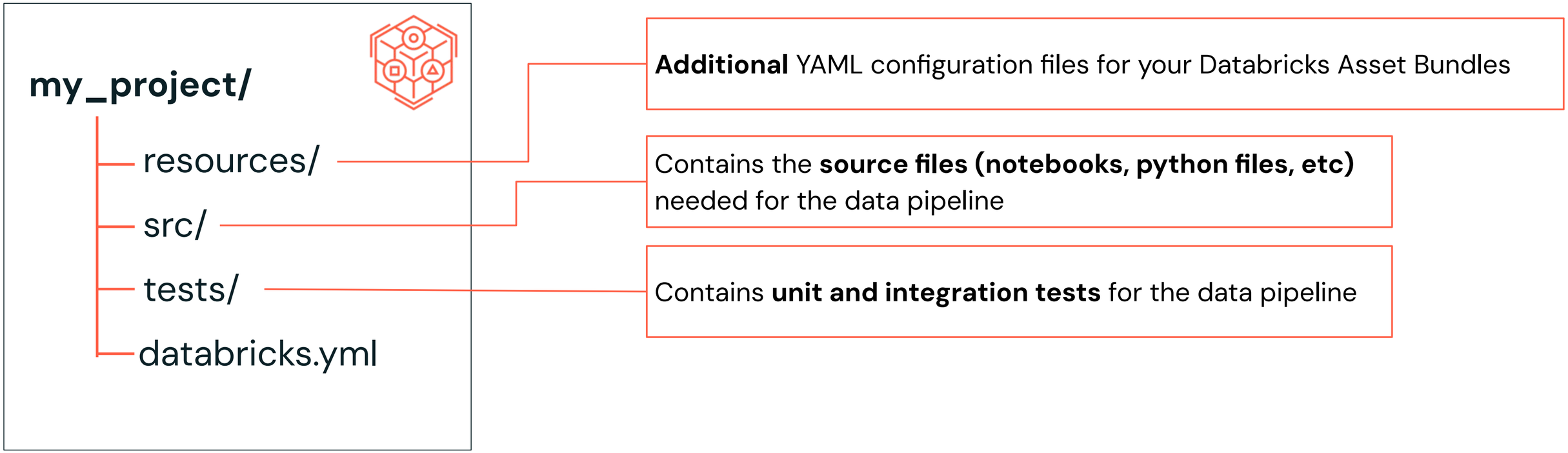
Introduction to DABs

CI/CD With DABs Overview



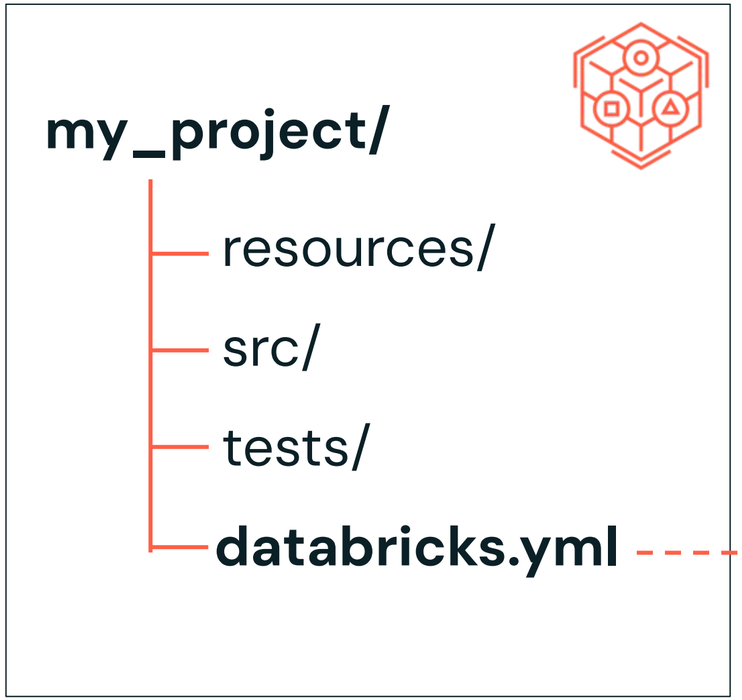
Introduction to DABs

Simple Project Structure Overview



Introduction to DABs

Simple Project Structure Overview



REQUIRED bundle configuration file that must:

- be expressed in **YAML format**
- contain at minimum the **top-level bundle mapping**
- contain at minimum one (and only one) bundle configuration file named **databricks.yml**

Top Level Mappings

The **databricks.yml** configuration top level mappings include:

- **bundle**
- **resources**
- **targets**
- variables
- workspace
- permissions
- artifacts
- include
- sync

databricks.yml file example

```
bundle:
    name: demo01_bundle

resources:
    jobs:
        l1_simple_dab:
            name: my_job_name_l1_simple_dab
            tasks:
                - task_key: create_bronze_table
                  notebook_task:
                      notebook_path: ./src/create_bronze_table.py
                      source: WORKSPACE
            . . .
targets:
    development:
        mode: development
        default: true
        workspace:
            host: https://dev.cloud.databricks.com/

    production:
        mode: production
        workspace:
            host: https://test.cloud.databricks.com/
```

The **bundle** top level mapping declares a required bundle **name**, and can use other optional configurations

The top-level mapping of **resources** defines the databricks resources used by the bundle, including:

- Jobs
- DLT pipelines
- MLflow
- And more

Resources are defined using the corresponding **Databricks REST API**.

databricks.yml file example

```
bundle:
    name: demo01_bundle

resources:
    jobs:
        l1_simple_dab:
            name: my_job_name_l1_simple_dab
            tasks:
                - task_key: create_bronze_table
                  notebook_task:
                      notebook_path: ./src/create_bronze_table.py
                      source: WORKSPACE
            ...
targets:
    development:
        mode: development
        default: true
        workspace:
            host: https://dev.cloud

    production:
        mode: production
        workspace:
            host: https://test.cloud.databricks.com/
```

Job key

Job Name

Starting December 20, 2024, the default format for new notebooks is now .ipynb format.

Make sure to specify the correct notebook extension.

The top-level mapping for **targets** sets specific environments and environment configurations, including:

- **Mode** types
- **Default** target environment
- Various other **configurations** and configuration **overrides**

Includes two environments: **development** and **production**, each with unique configurations and overrides.

databricks.yml file example

```
bundle:
    name: demo01_bundle

resources:
    jobs:
        l1_simple_dab:
            name: my_job_name_l1_simple_dab
            tasks:
                - task_key: create_bronze_table
                  notebook_task:
                      notebook_path: ./src/create_bronze_table.py
                      source: WORKSPACE
            ...
targets:
    development:
        mode: development
        default: true
        workspace:
            host: https://dev.cloud.databricks.com/

    production:
        mode: production
        workspace:
            host: https://prod.cloud.databricks.com/
```

Validate, Deploy and Run Your DAB

Using the Databricks CLI

1

```
databricks bundle validate
```

Returns **warnings** if unknown resource properties are found in bundle configuration files.

2

```
databricks bundle deploy -t development
```

Specifies which environment to **deploy** your bundle into. In this example, the bundle will be deployed to the **development** environment.

3

```
databricks bundle run -t development
l1_simple_dab
```

Specifies to run your bundle in the environment. You must specify the **job key name** to run the bundle job.

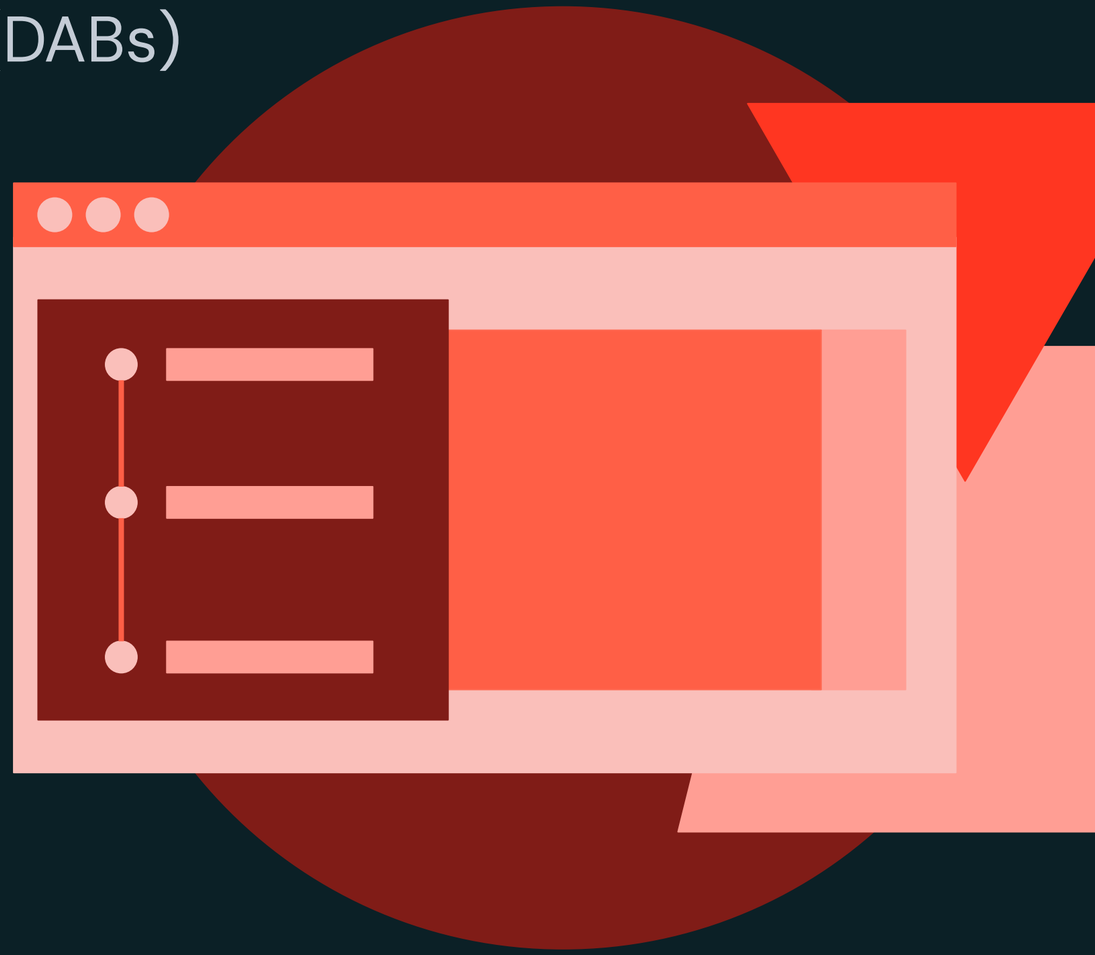
```
bundle:
    name: demo01_bundle

resources:
  jobs:
    l1_simple_dab:
        name: my_job_name_l1_simple_dab
        tasks:
          - task_key: create_bronze_table
            notebook_task:
              notebook_path: ./src/create_bronze_table.py
              source: WORKSPACE
        ...
targets:
    development:
        mode: development
        default: true
        workspace:
            host: https://dev.cloud.databricks.com/

    production ...
```

databricks

Deployment with Databricks Asset Bundles (DABs)

DEMONSTRATION

Deploying a Simple DAB



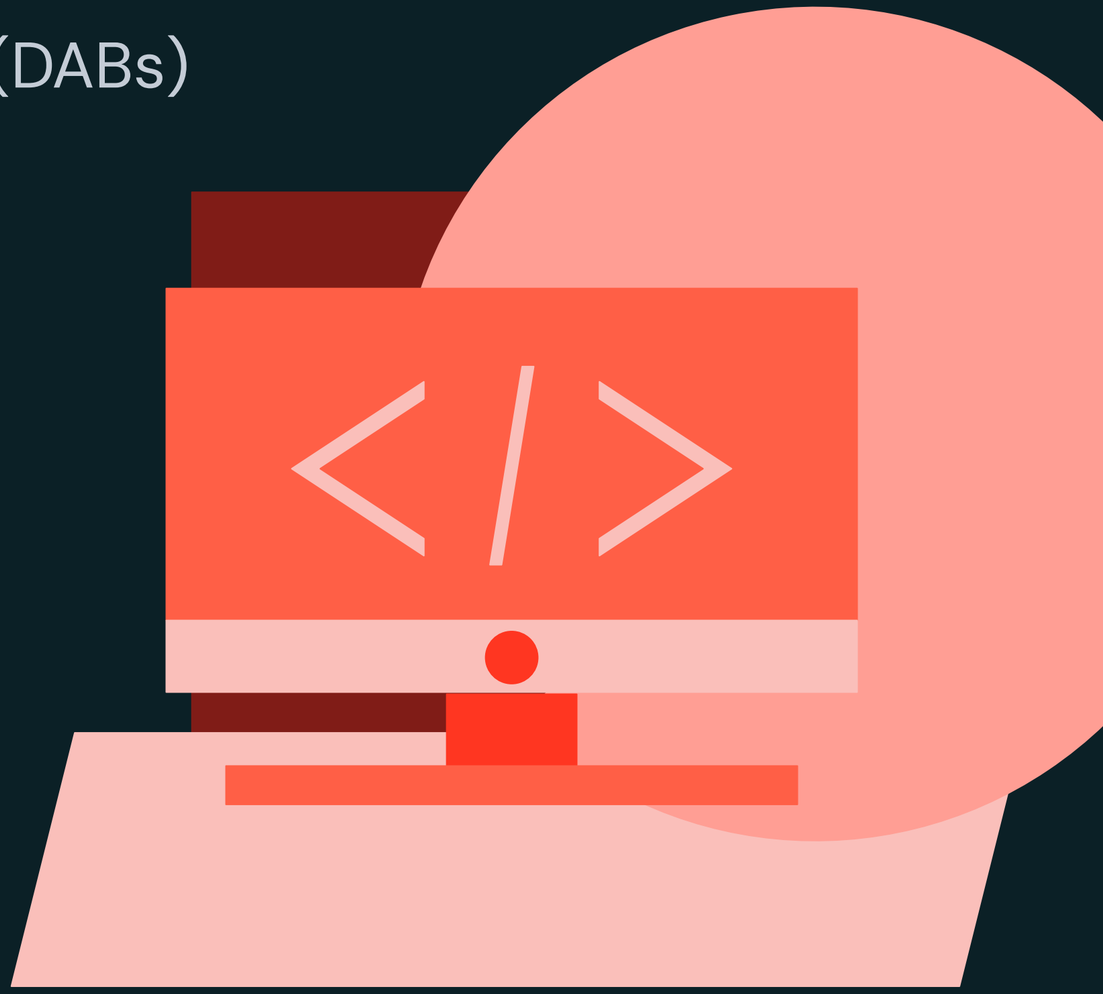
Notebook: /01 - Deploying a Simple DAB/Demo - Deploying a Simple DAB

databricks

Deployment with Databricks Asset Bundles (DABs)

LAB EXERCISE

Deploy a Simple DAB



Notebook: /02L - Deploy a Simple DAB/Lab - Deploy a Simple DAB

databricks

Deployment with Databricks Asset Bundles (DABs)

LECTURE

Variable Substitutions in DABs

Variables in Databricks Asset Bundles (DABs)

Overview

Default Substitutions

By default there are a variety of **variable substitutions** available:

- `${bundle.name}`
- `${bundle.target}`
- `${workspace.file_path}`
- `${workspace.root_path}`
- `${resources.jobs.<job-name>.id}`
- `${resources.models.<model-name>.name}`
- `${resources.pipelines.<pipeline-name>.name}`

Variables in Databricks Asset Bundles (DABs)

Overview

Default Substitutions

By default there are a variety of **variable substitutions** available:

- `${bundle.name}`
- `${bundle.target}`
- `${workspace.file_path}`
- `${workspace.root_path}`
- `${resources.jobs.<job-name>.id}`
- `${resources.models.<model-name>.name}`
- `${resources.pipelines.<pipeline-name>.name}`

Simple Custom Variables

- You can define **simple custom variables** in your bundle to enable dynamic retrieval of values needed for many scenarios.
- Custom variables are declared in your bundle configuration files within the **variables** mapping

```
variables:
  my_lab_user_name:
    description: Your user name
    default: labuser23904
```

Variables in Databricks Asset Bundles (DABs)

Overview

Default Substitutions

By default there are a variety of **variable substitutions** available:

- `${bundle.name}`
- `${bundle.target}`
- `${workspace.file_path}`
- `${workspace.root_path}`
- `${resources.jobs.<job-name>.id}`
- `${resources.models.<model-name>.name}`
- `${resources.pipelines.<pipeline-name>.name}`

Simple Custom Variables

- You can define **simple custom variables** in your bundle to enable dynamic retrieval of values needed for many scenarios.
- Custom variables are declared in your bundle configuration files within the **variables** mapping

```
variables:
  my_lab_user_name:
    description: Your user name
    default: labuser23904
```

Complex Variables

- A custom variable is assumed to be of type string unless you define it as a **complex variable**.
- Define a custom variable by setting the type to **complex**

```
variables:
  my_cluster:
    description: "My cluster"
    type: complex
    default:
      spark_version: "15.4.x-scala2.11"
      node_type_id: "Standard_DS3_v2"
      num_workers: 2
```

Creating Simple Custom Variables

The bundles settings file can contain one top-level **variables** mapping where custom variables are defined

Create a variable named **my_lab_user_name** with the default value of *labuser23904*

Create variables named **catalog_dev** and **catalog_prod** uses the **my_lab_user_name** value and appends *values*

databricks.yml file example

```
. . .
variables:
    my_lab_user_name:
        description: Your user name
        default: labuser23904

    catalog_dev:
        description:Development catalog reference
        default: ${var.my_lab_user_name}_1_dev

    catalog_prod:
        description: Production catalog reference
        default: ${var.my_lab_user_name}_3_prod
. . .
```

Reference a custom variables

catalog_dev = ***labuser23904_1_dev***

catalog_prod = ***labuser23904_3_prod***

Defining a Complex Variable

A variable is assumed to be of type string unless you define it as a **complex** variable.

databricks.yml file example

```
bundle:
    name: demo03_bundle

...
variables:

    my_cluster:
        description: "My cluster"
        type: complex
        default:
            spark_version: "15.4.x-scala2.11"
            node_type_id: "Standard_DS3_v2"
            num_workers: 2

...
```

Lookup Variables

Dynamically Retrieve an Object's Value

For specific object types you can **define a lookup** for your custom variable to retrieve the object's ID

Create the **new** variable

Lookup the cluster name to retrieve the **cluster ID**

```
. . .
variables:

  my_cluster_id:
    description: "Get the cluster ID using a lookup variable"
    lookup:
      cluster: myclustername

. . .
```

Lookup Variables

Can Define Lookups for the a Variety of Environment Configurations

- alert
- cluster_policy
- cluster
- dashboard
- instance_pool
- job
- metastore
- notification_destination
- pipeline
- query
- service_principal
- warehouse

Target Environment Variable Overrides

Dynamically **modify variable values** within each target environment using the **target** top-level mapping.

When deploying to **development**, the **`target_catalog`** variable will use the **`catalog_dev`** value.

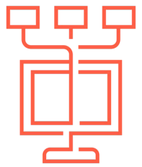
When deploying to **production**, the **`target_catalog`** variable will use the **`catalog_prod`** value.

```
...

targets:

    development:
        ...
        variables:
            target_catalog: ${var.catalog_dev}

    production:
        ...
        variables:
            target_catalog: ${var.catalog_prod}
```

A default value for the variable **must be defined in the variables top level mapping** in order for an override to work.

Benefits of Using Variables

Databricks Asset Bundles



Customizable for Different Environments

Easily modify configurations (e.g., database connections, file paths, etc) for development, staging, and/or production environments.

Reusability Across Databricks Projects

You can use the same asset bundle across multiple teams or workspaces by adjusting only variable values.

Easy Maintenance & Updates

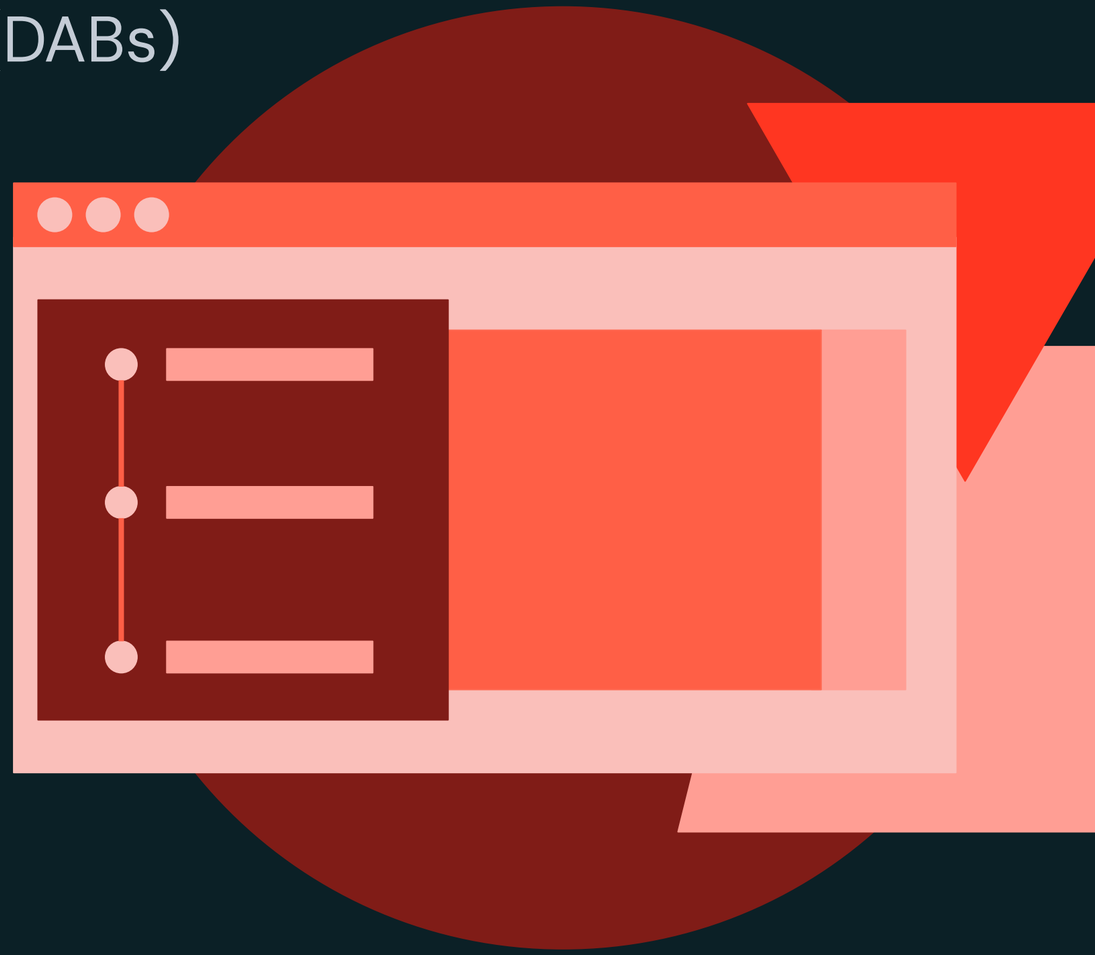
Quickly update assets by modifying variables to ensure, consistency and reduce errors.

databricks

Deployment with Databricks Asset Bundles (DABs)

DEMONSTRATION

Deploying a DAB to Multiple Environments



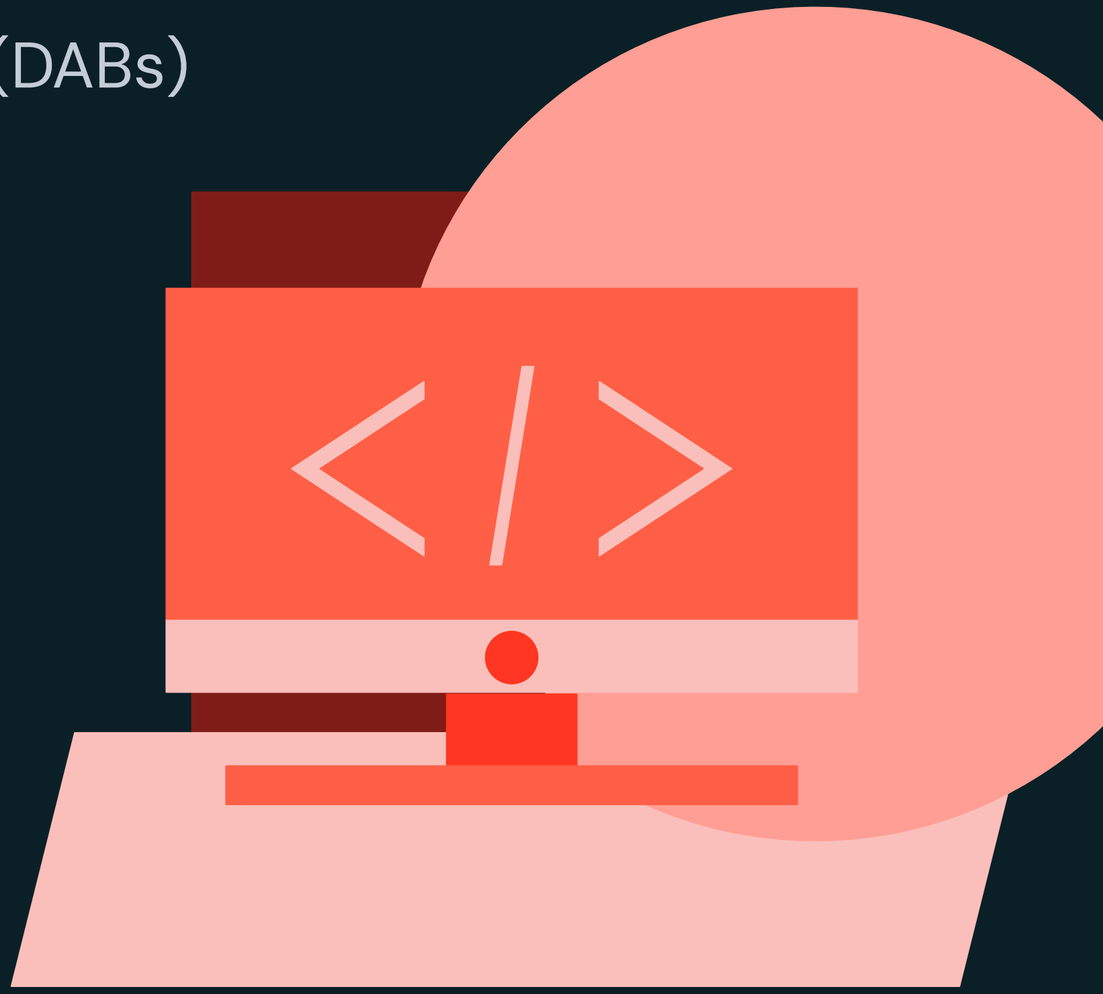
Notebook: /03 - Deploying a DAB to Multiple Environments/Demo - Deploying a DAB to Multiple Environments

databricks

Deployment with Databricks Asset Bundles (DABs)

LAB EXERCISE

Deploy a DAB to Multiple Environments



Notebook: /04L - Deploy a DAB to Multiple Environments/Lab - Deploy a DAB to Multiple Environments

databricks

Deployment with Databricks Asset Bundles (DABs)

LECTURE

DAB Project Templates Overview

Databricks Asset Bundle Project Templates

Overview

Default Bundle Template

Use a Databricks **default** bundle template to create your bundle

- Templates:
  - **default-python**
  - default-sql
  - dbt-sql
  - Mlops-stacks

```
databricks bundle init default-python
```

Custom Bundle Template

Databricks Asset Bundle Project Templates

Overview

Default Bundle Template

Use a Databricks default bundle template to create your bundle

- Templates:
  - **default-python**
  - default-sql
  - dbt-sql
  - Mlops-stacks

Custom Bundle Template

- At a minimum, it must have **databricks_template_schema.json** and **databricks.yml.tmpl**
- You can add user prompts, specific folder structure and more
- To use a custom bundle template, pass its local path or remote URL to the Databricks CLI bundle init command.

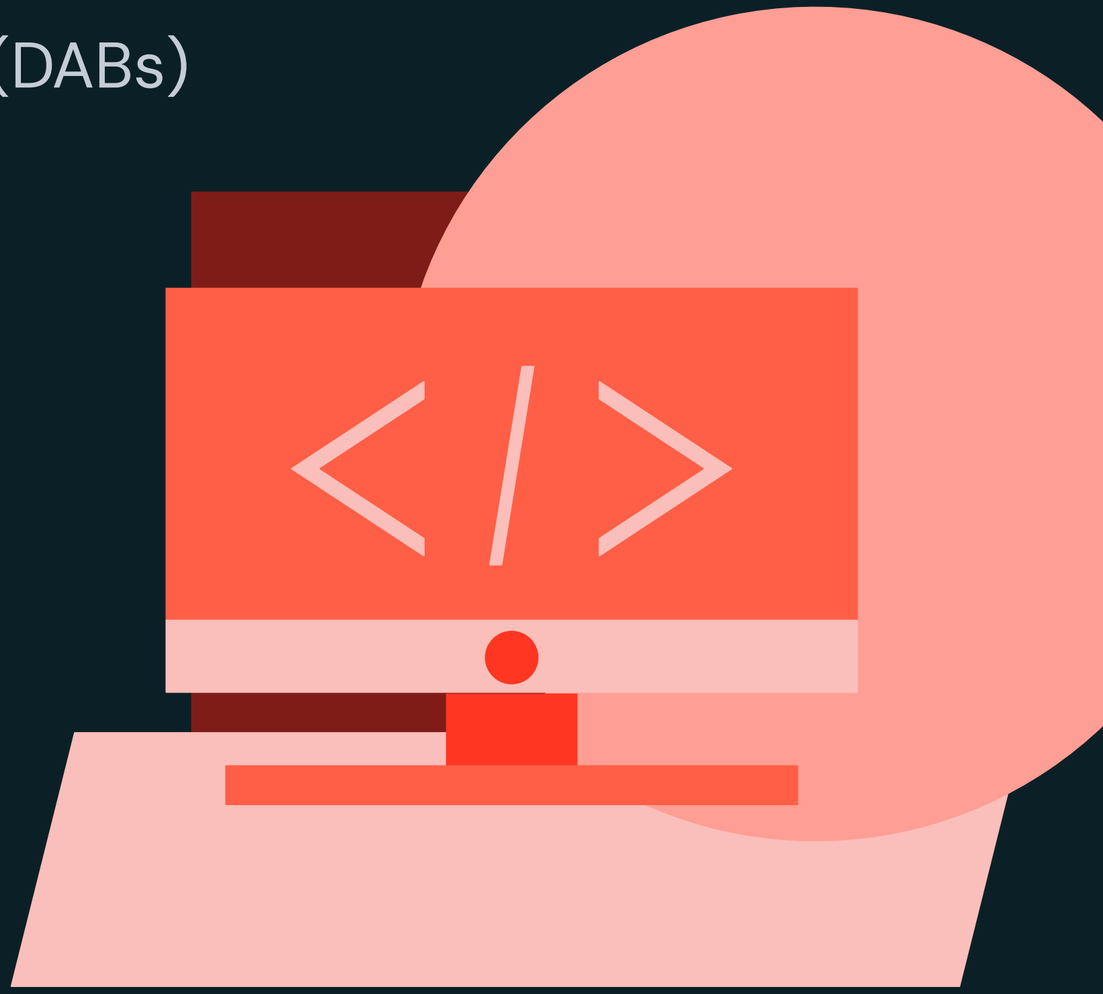
```
databricks bundle init
/projects/templates/test-template
```

databricks

Deployment with Databricks Asset Bundles (DABs)

LAB EXERCISE

Use a Databricks Default DAB Template



Notebook: /05L - Use a Databricks Default DAB Template/Lab - Use a Databricks Default DAB Template

databricks

Deployment with Databricks Asset Bundles (DABs)

LECTURE

CI/CD Project Overview with DABs

Planning the Project

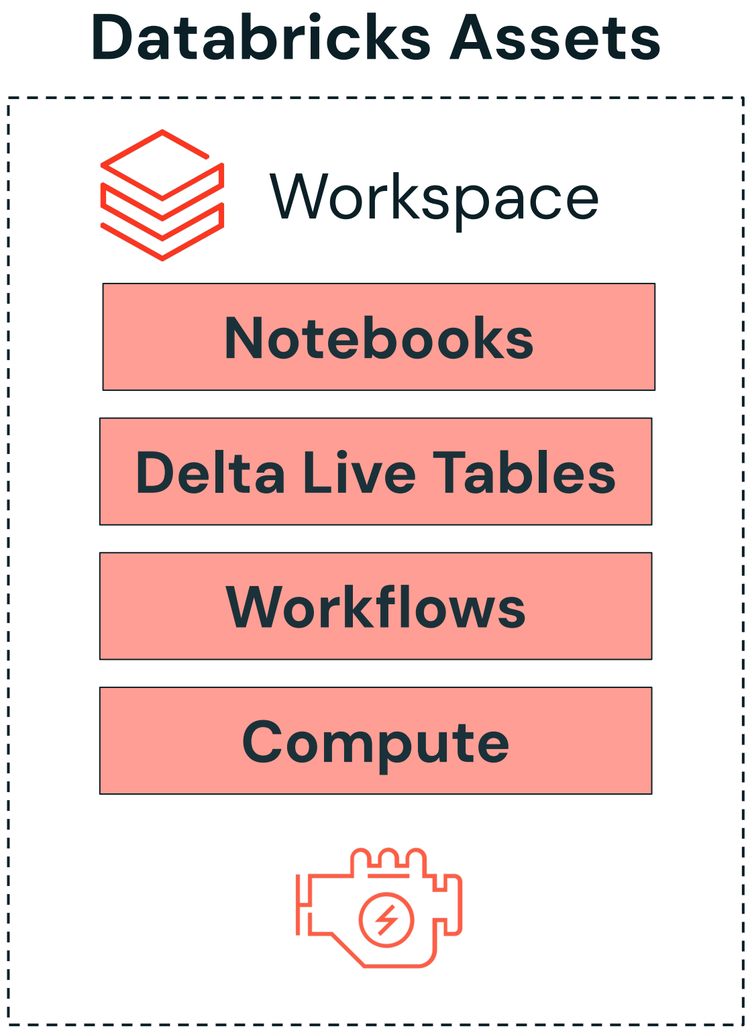
Requirements

Deliverable

- Visualize Health Data

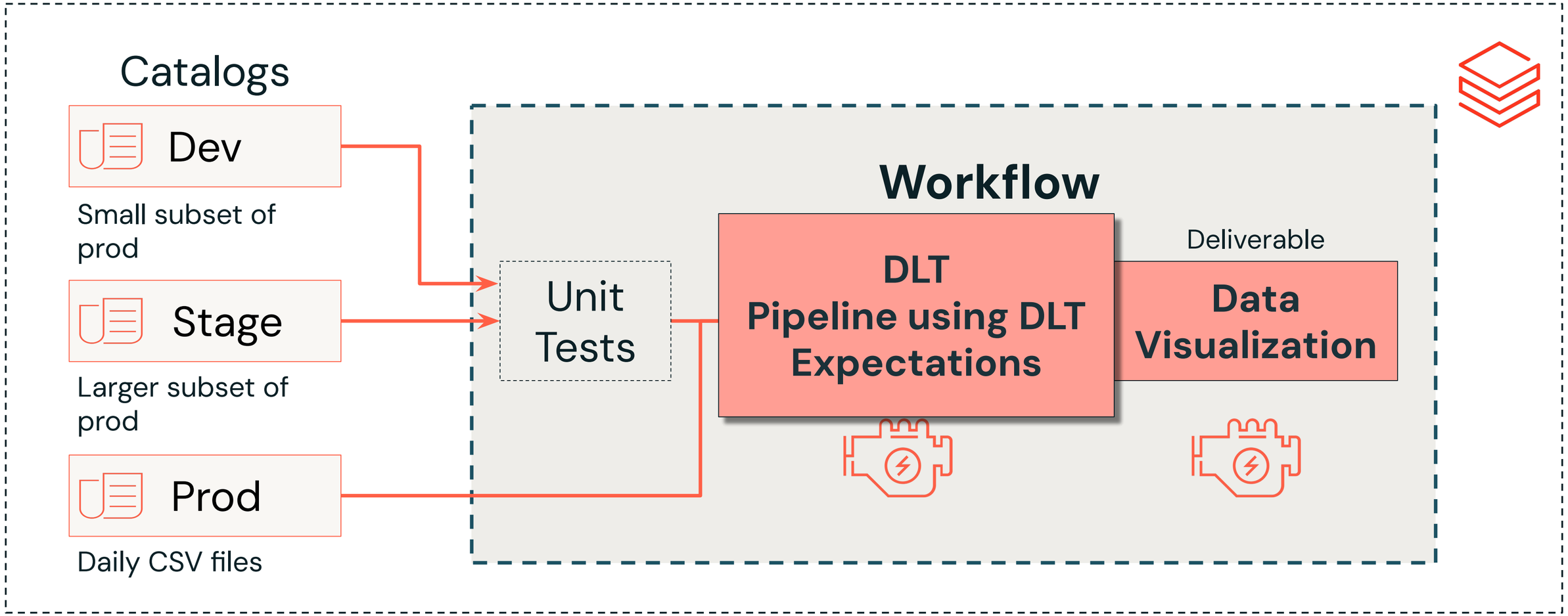
Tasks

- Ingest daily incremental CSV files to a bronze table
- Create a clean silver table
- Create gold tables to share with consumers



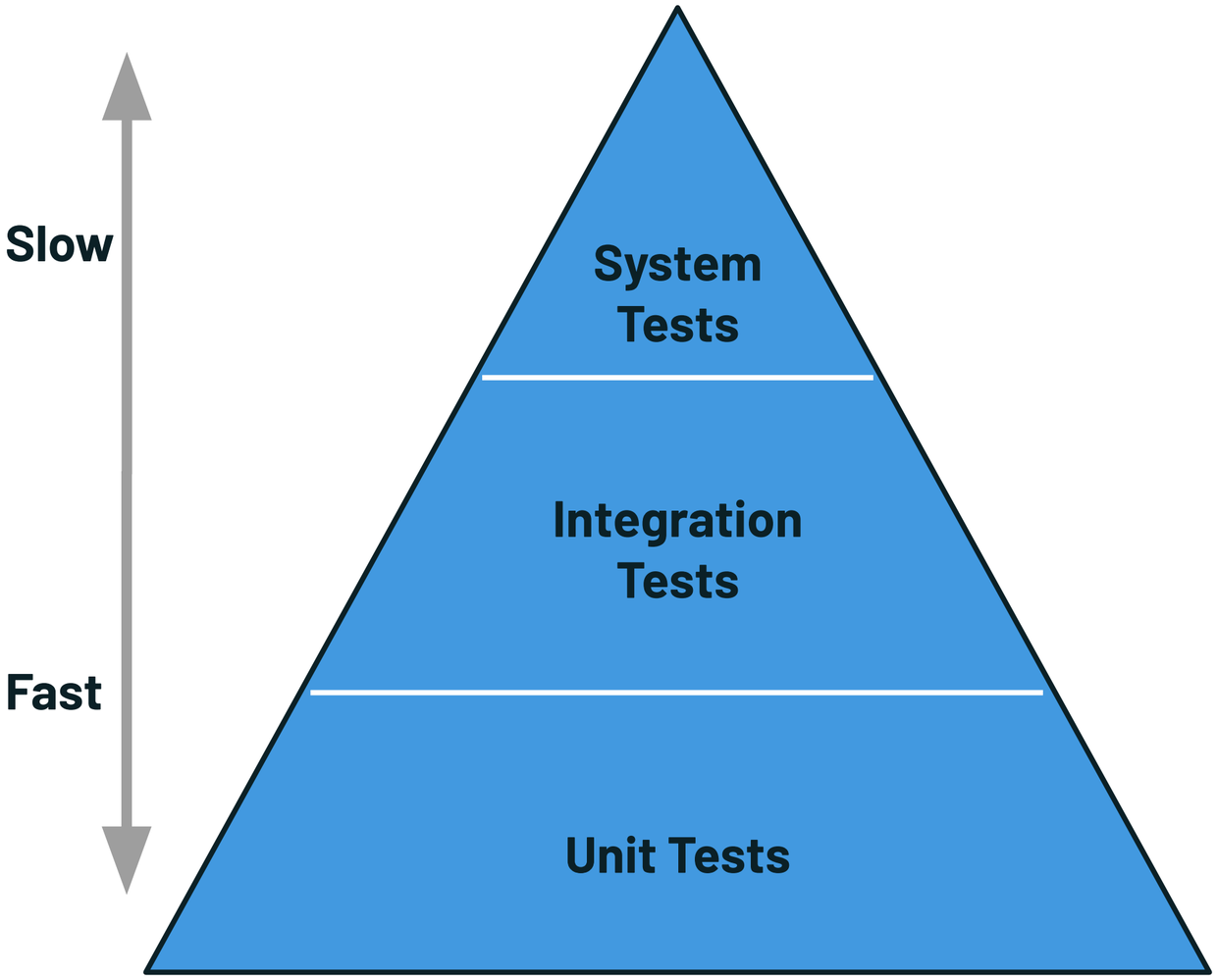
Planning the Project

Course Project Architecture



Review the Role of CI/CD Testing

High-level Testing Steps



- Test the entire application, ensuring that all parts function together in a real-world scenario
- **Ex:** End to end data pipeline in a Workflow

- Test the interaction between different components or systems
- **Ex:** Notebooks / DLT/ Jobs interactions

- Test **individual** functions or methods in isolation
- Fast, low cost, high coverage, and automated
- **Ex:** Custom pyspark functions

Review of Unit Testing with pytest

Pytest -is popular testing framework for Python that makes it easy to write simple and scalable test cases.

Uses Simple Syntax

Minimal syntax, just define functions starting with **test_**

Provides Assertions

Use **assert** statements to provide detailed error messages on failure

Automatic Discovery

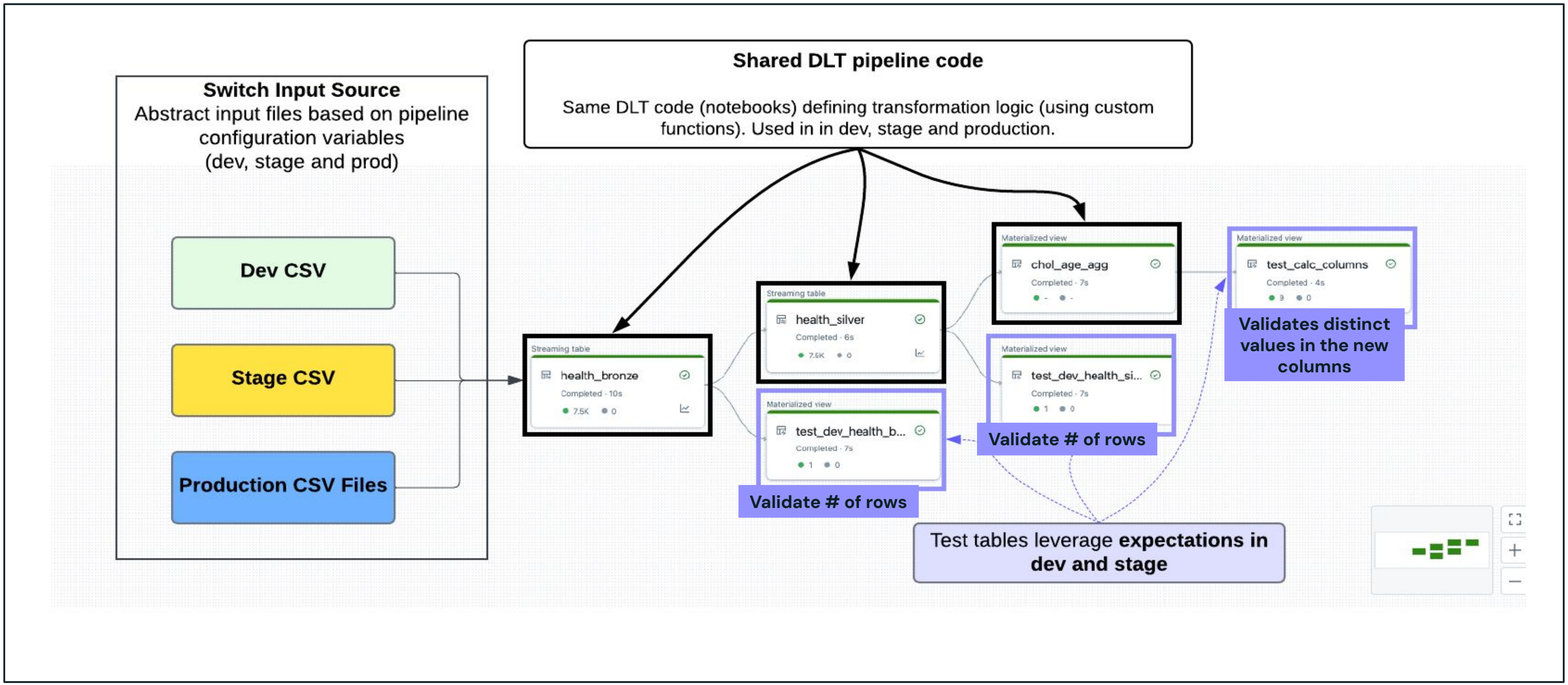
Finds and runs all tests **automatically** with a simple configuration

Rich Ecosystem

Extend functionality with plugins for coverage, parallel tests, and more

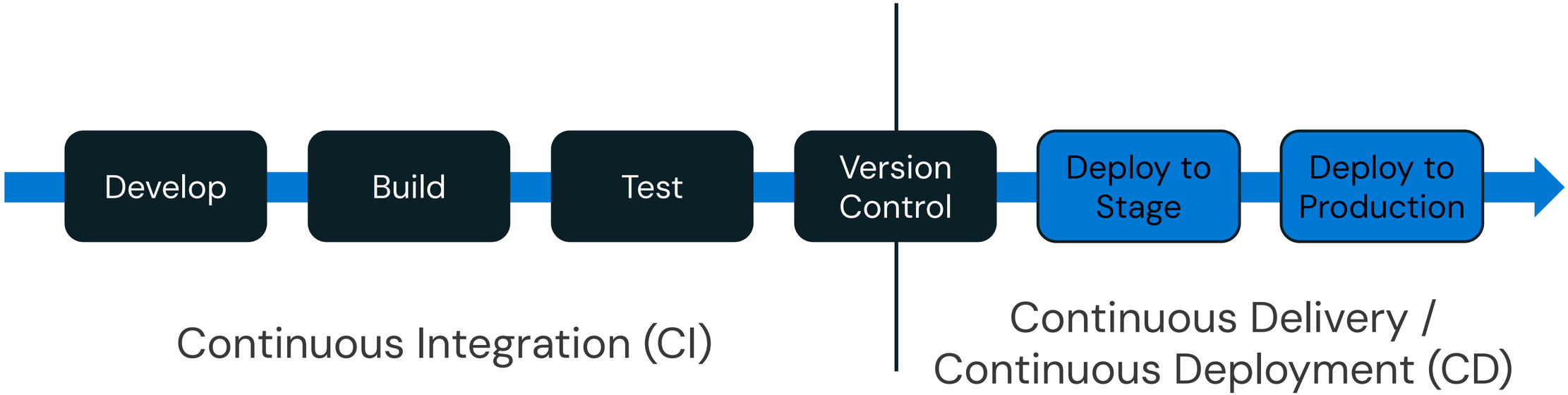
This course provides a simple introduction to **pytest**. There are many testing frameworks available, select the one that best meets your organization's needs.

Review of DLT Expectations for Integration Tests



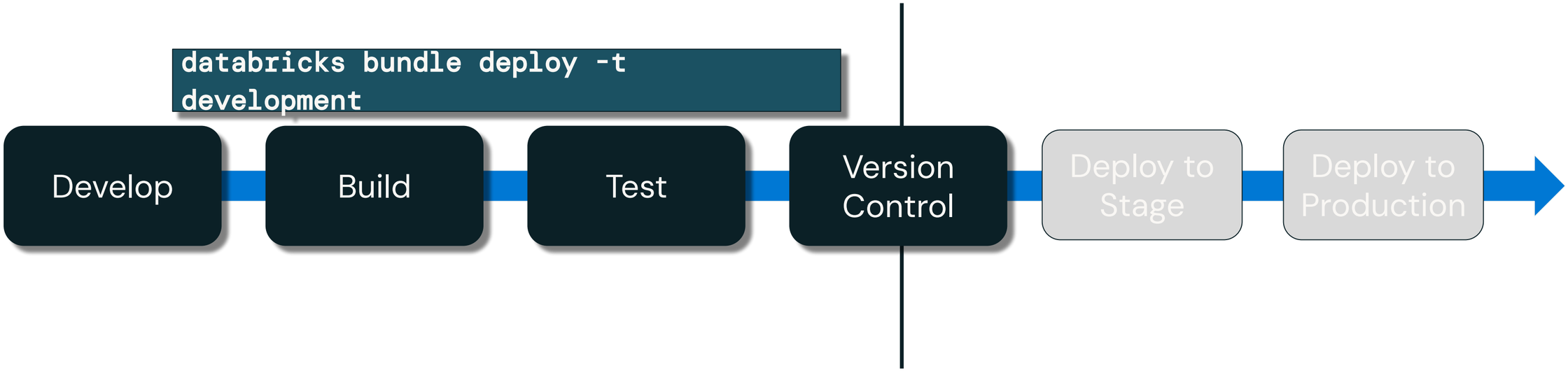
The Role of CI/CD in DevOps

High-Level CI/CD Workflow Overview



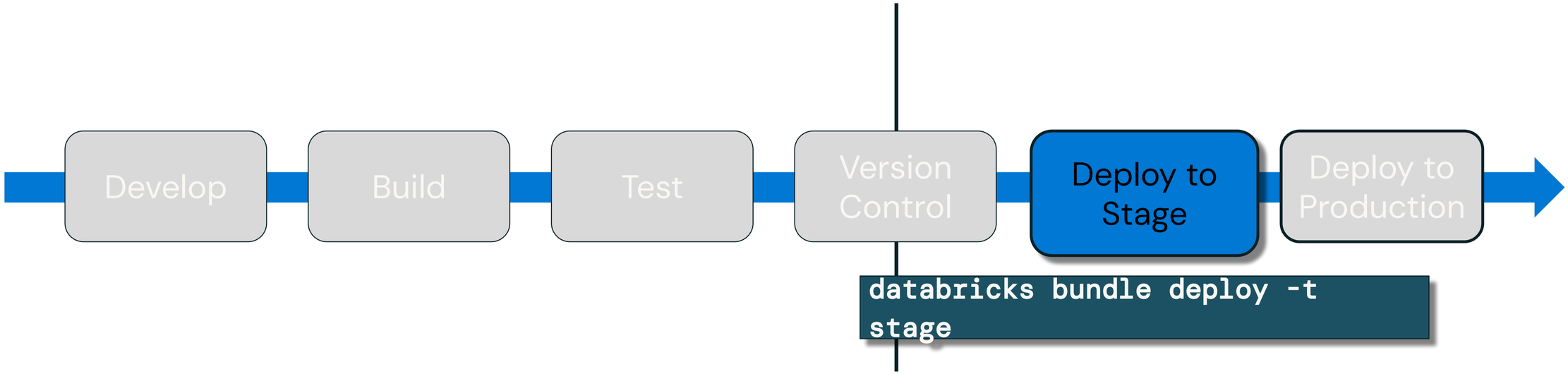
CI/CD with DABs

High-Level CI/CD Workflow Overview



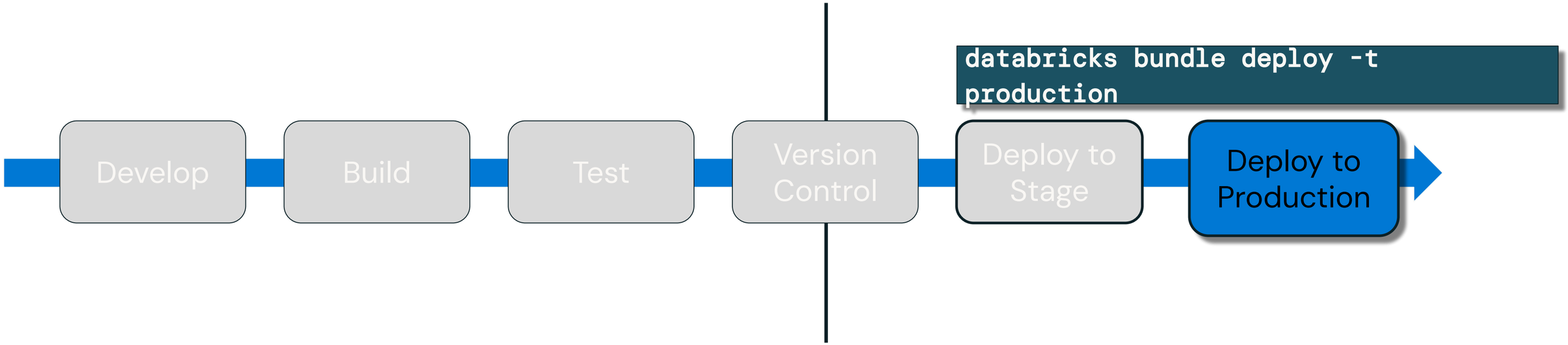
CI/CD with DABs

High-Level CI/CD Workflow Overview



CI/CD with DABs

High-Level CI/CD Workflow Overview

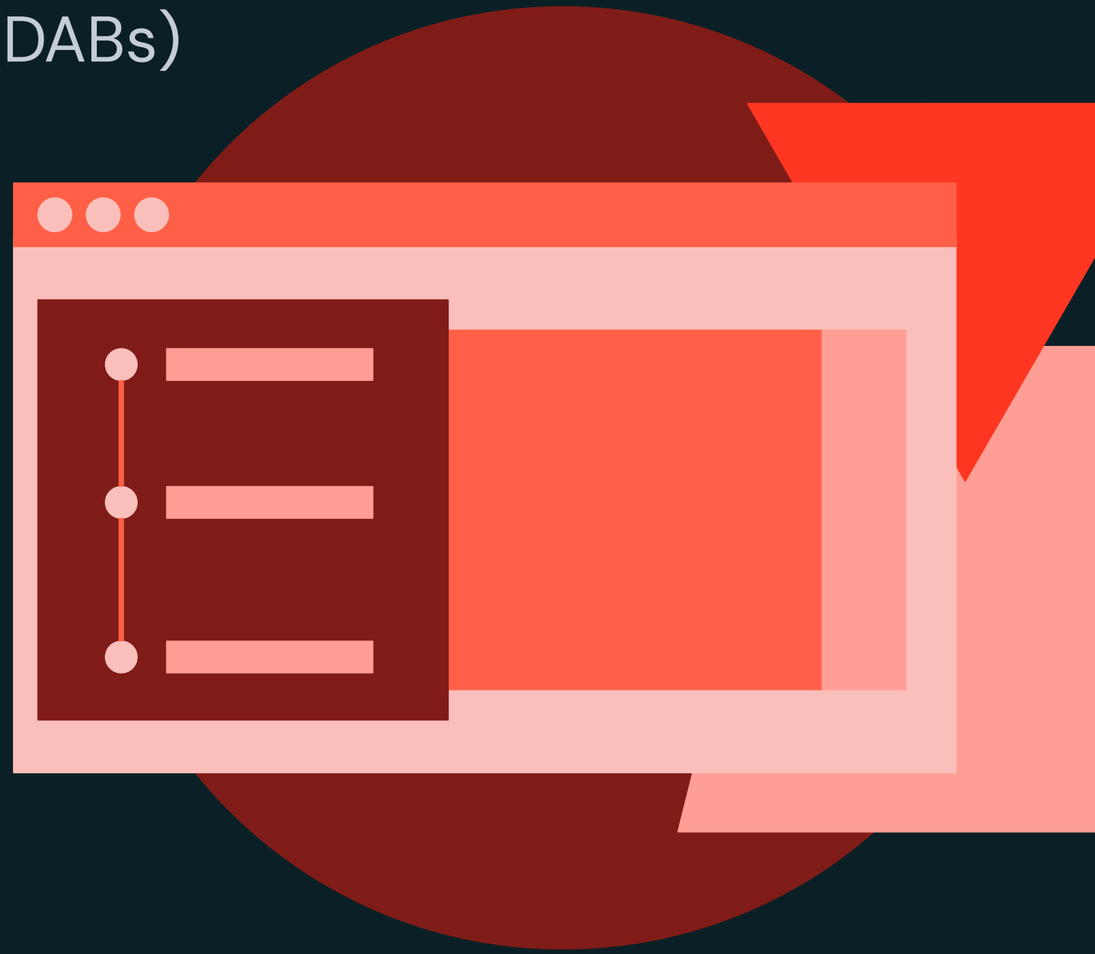


databricks

Deployment with Databricks Asset Bundles (DABs)

DEMONSTRATION

Continuous Integration and Continuous Deployment with DABs



Notebook: /06- Continuous Integration and Continuous Deployment with DABs/Demo - CICD with DABs

Full Project Outline

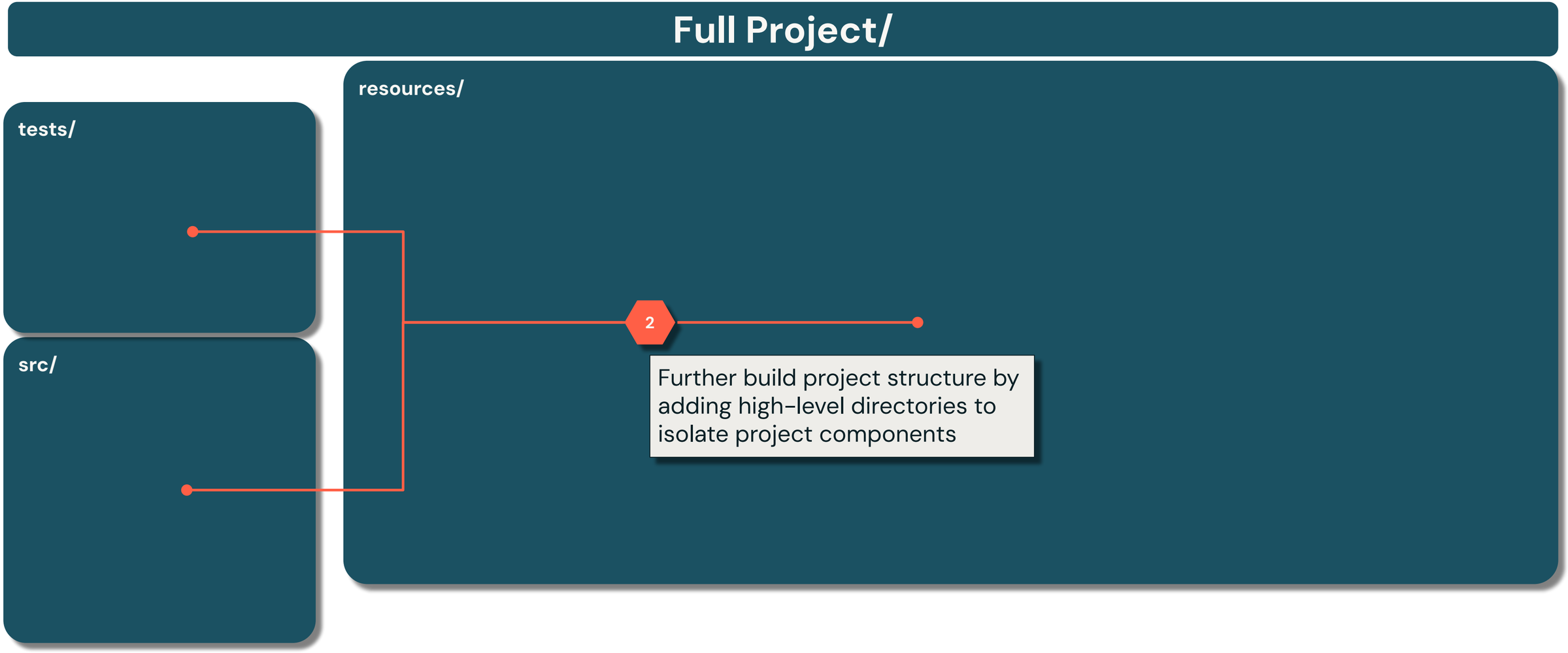
Folder Architecture

Full Project/

1. Create a project folder.

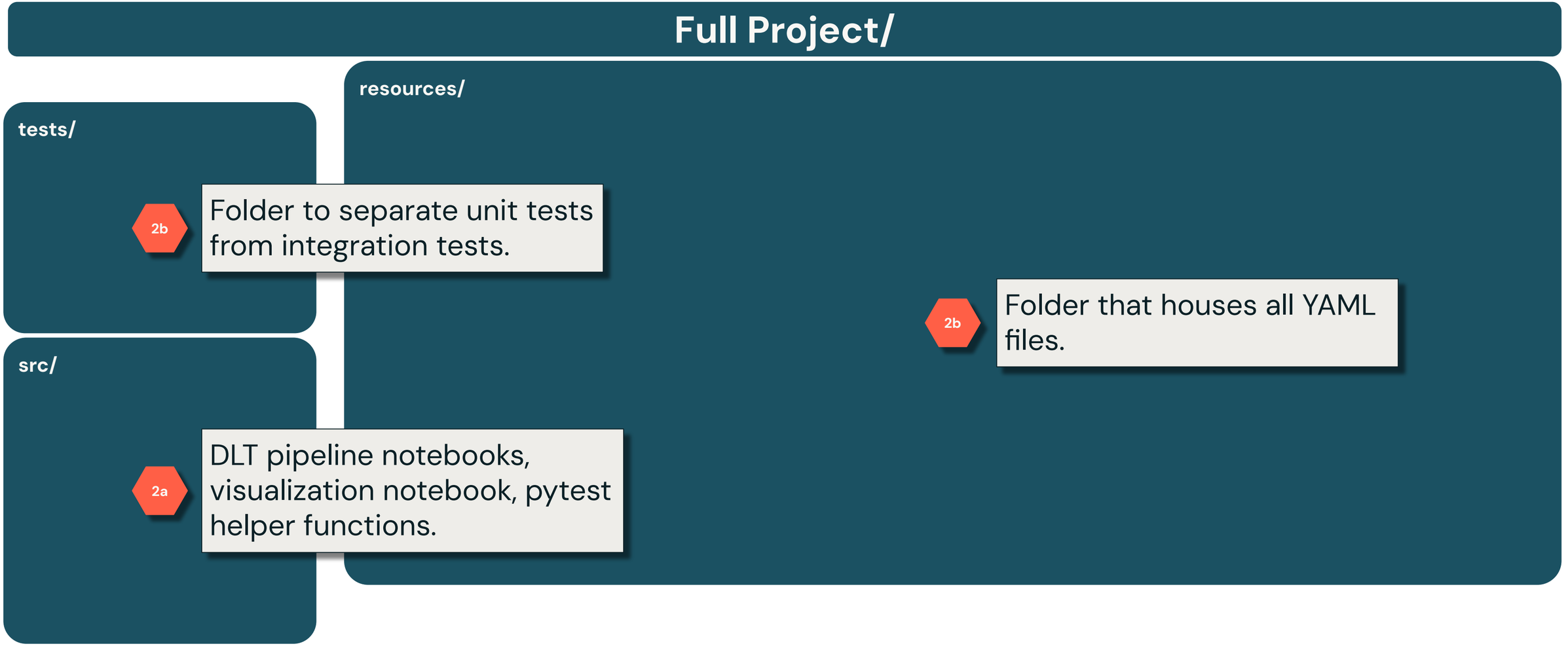
Full Project Outline

Folder Architecture



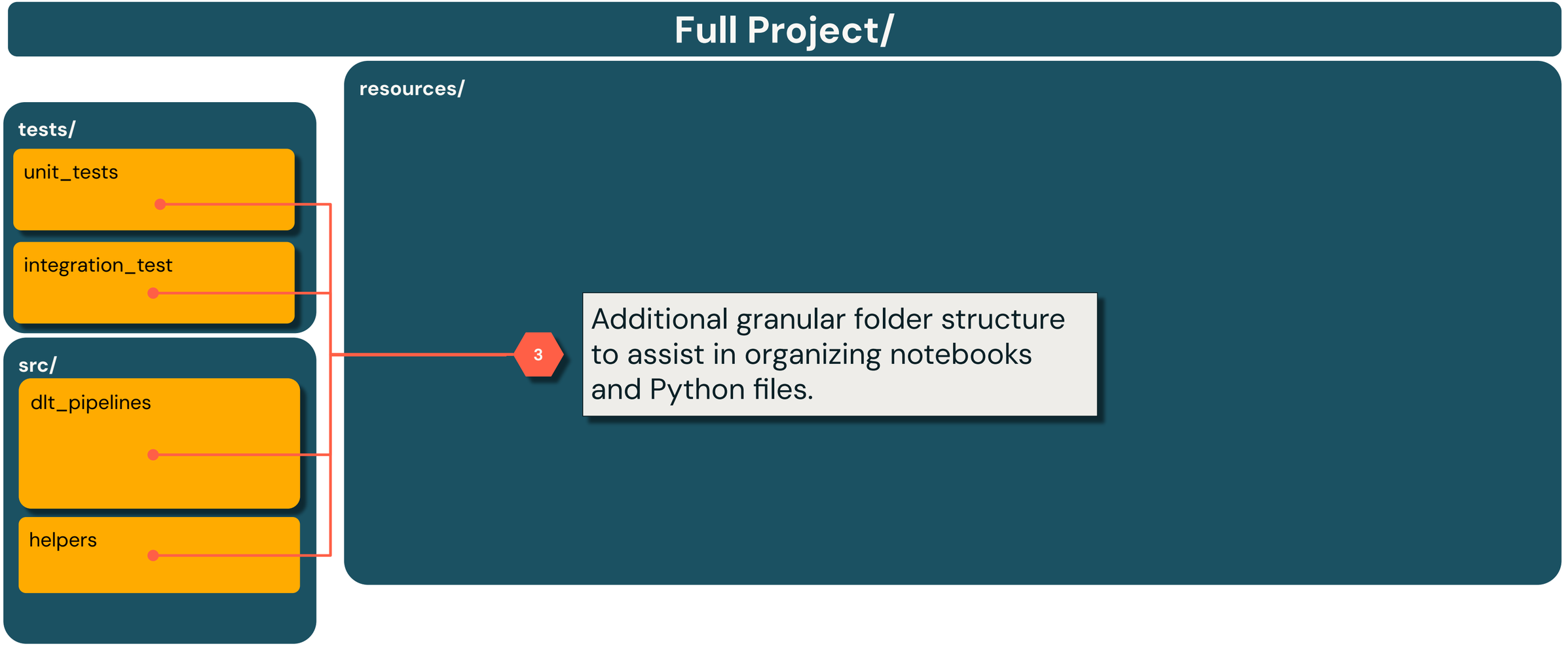
Full Project Outline

Folder Architecture



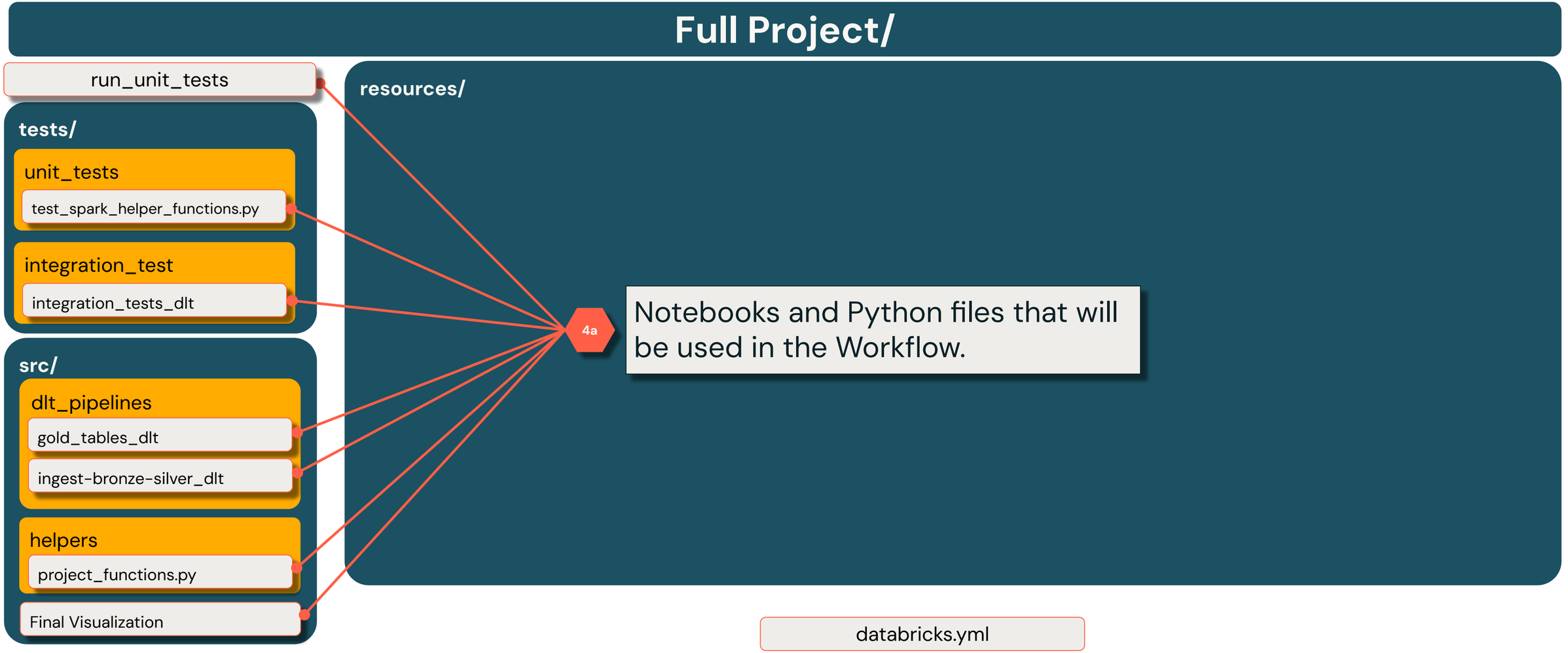
Full Project Outline

Folder Architecture



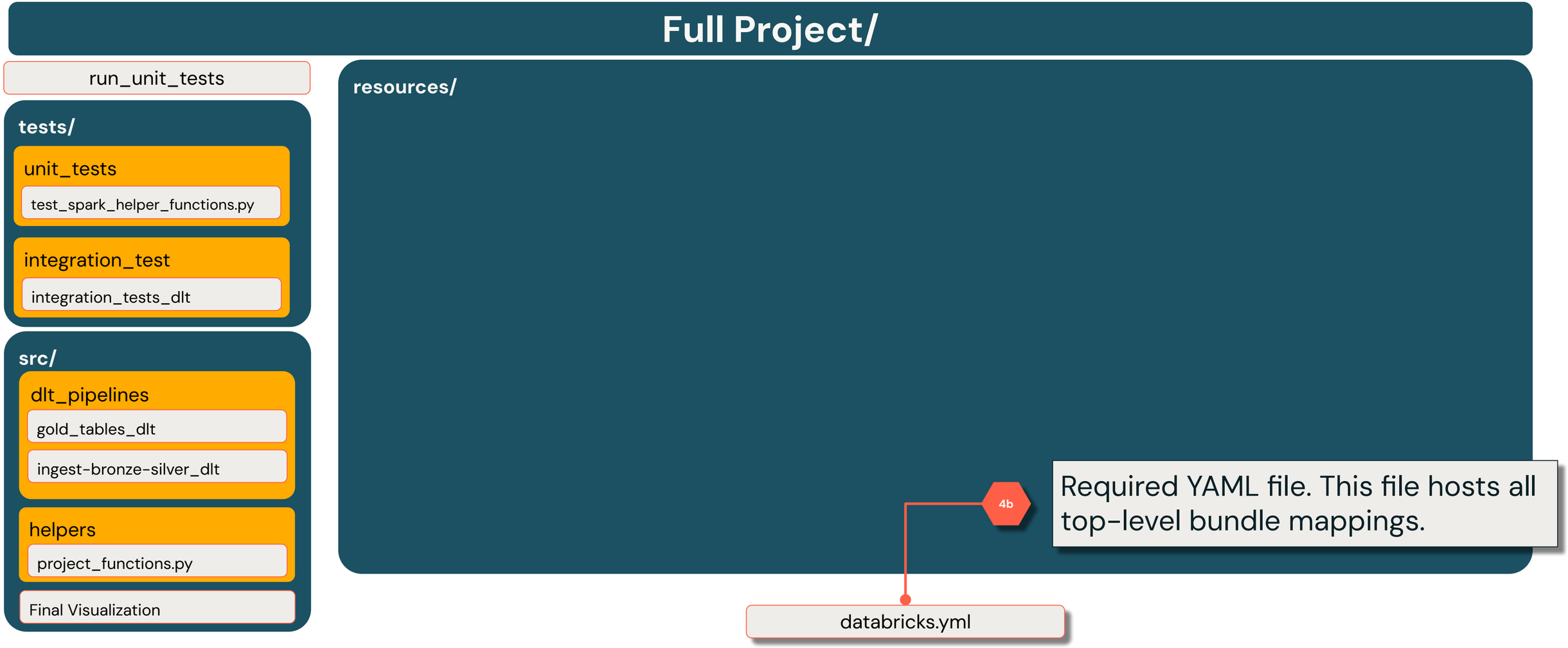
Full Project Outline

Folder Architecture



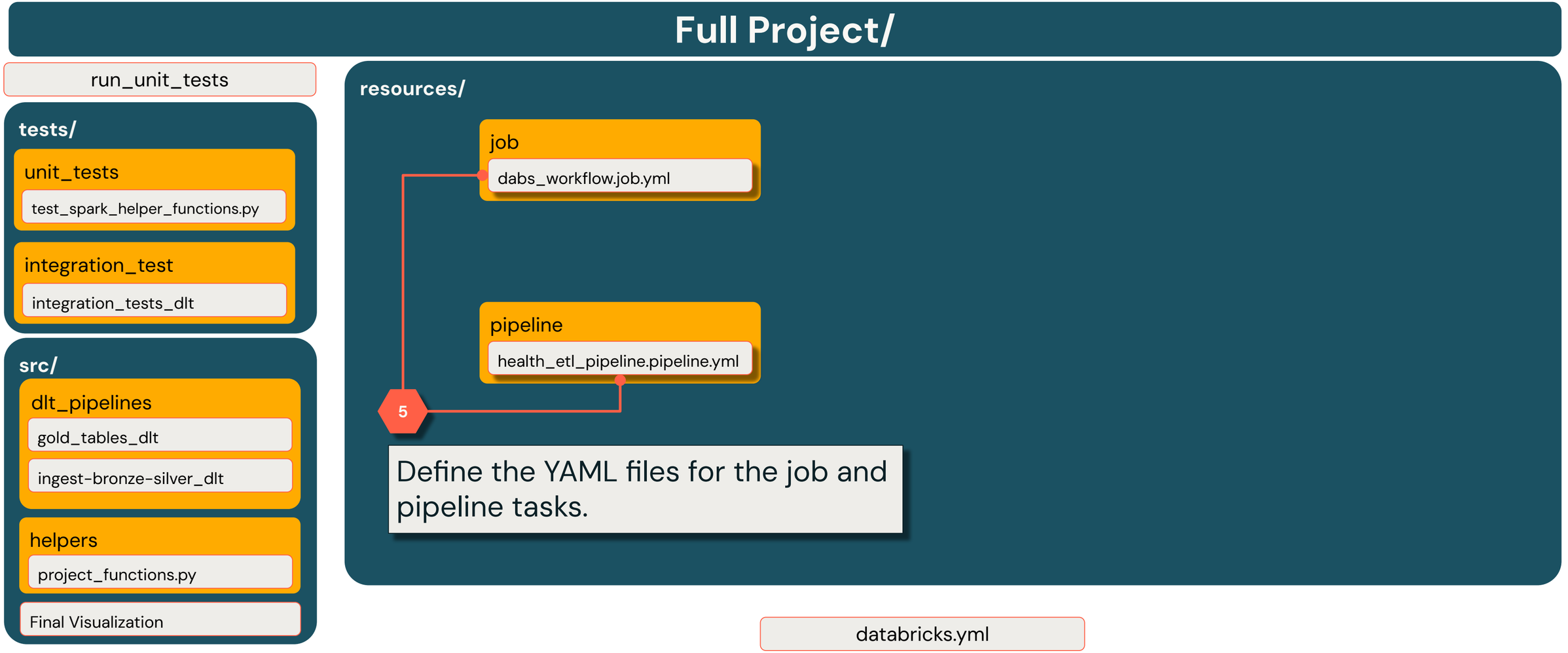
Full Project Outline

Folder Architecture



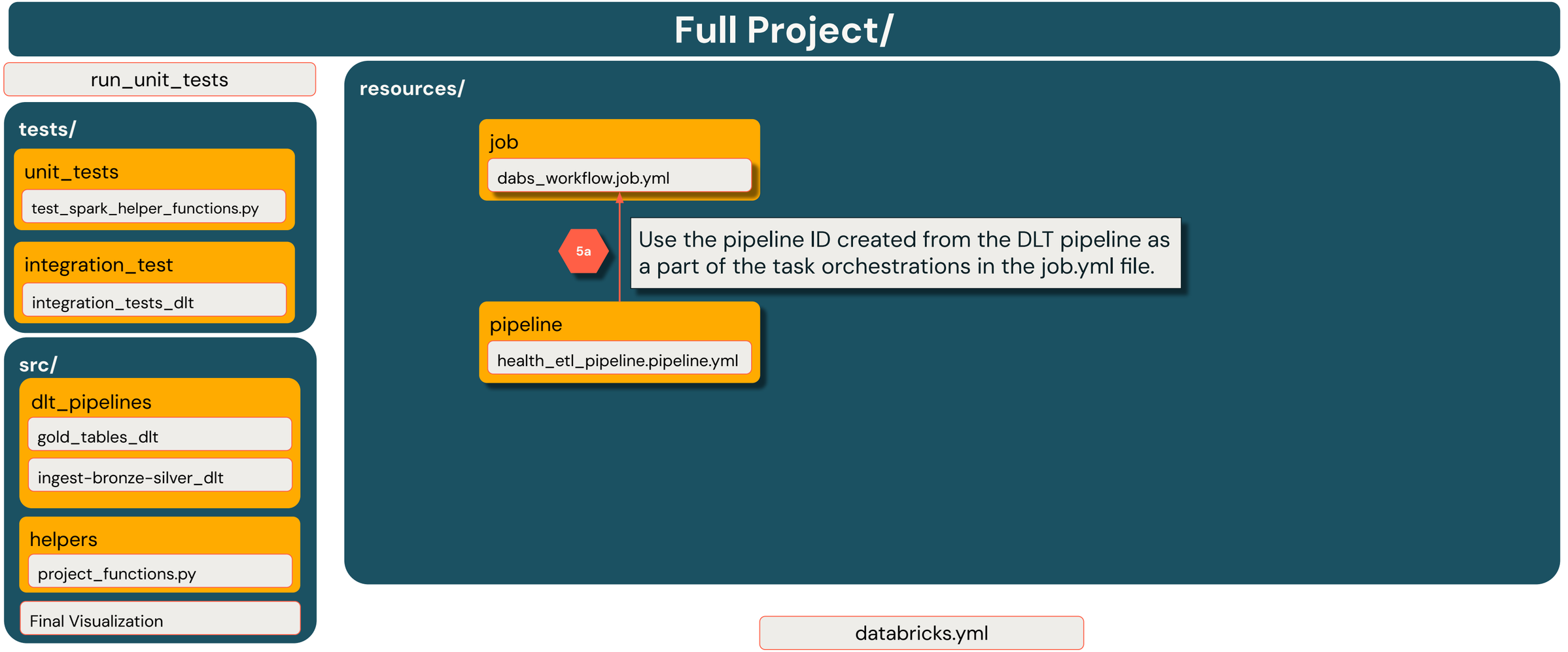
Full Project Outline

Folder Architecture



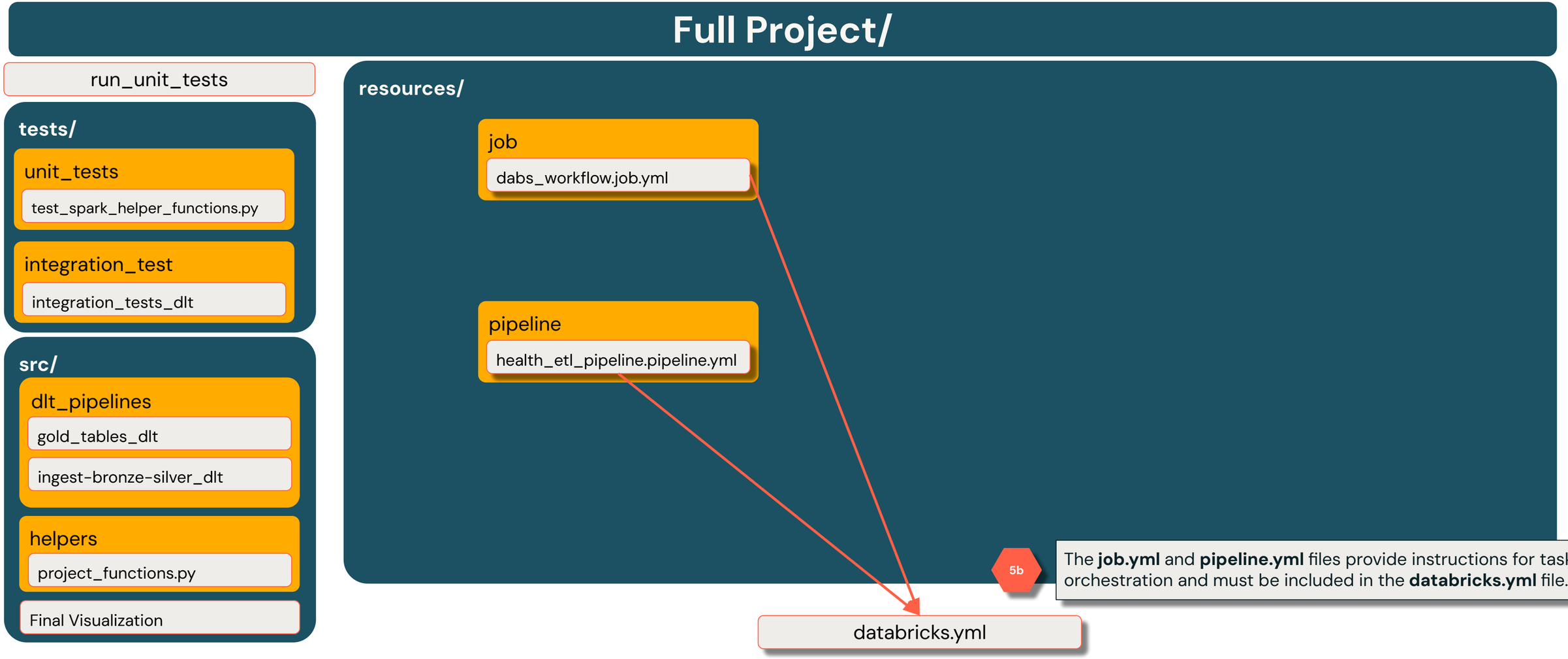
Full Project Outline

Folder Architecture



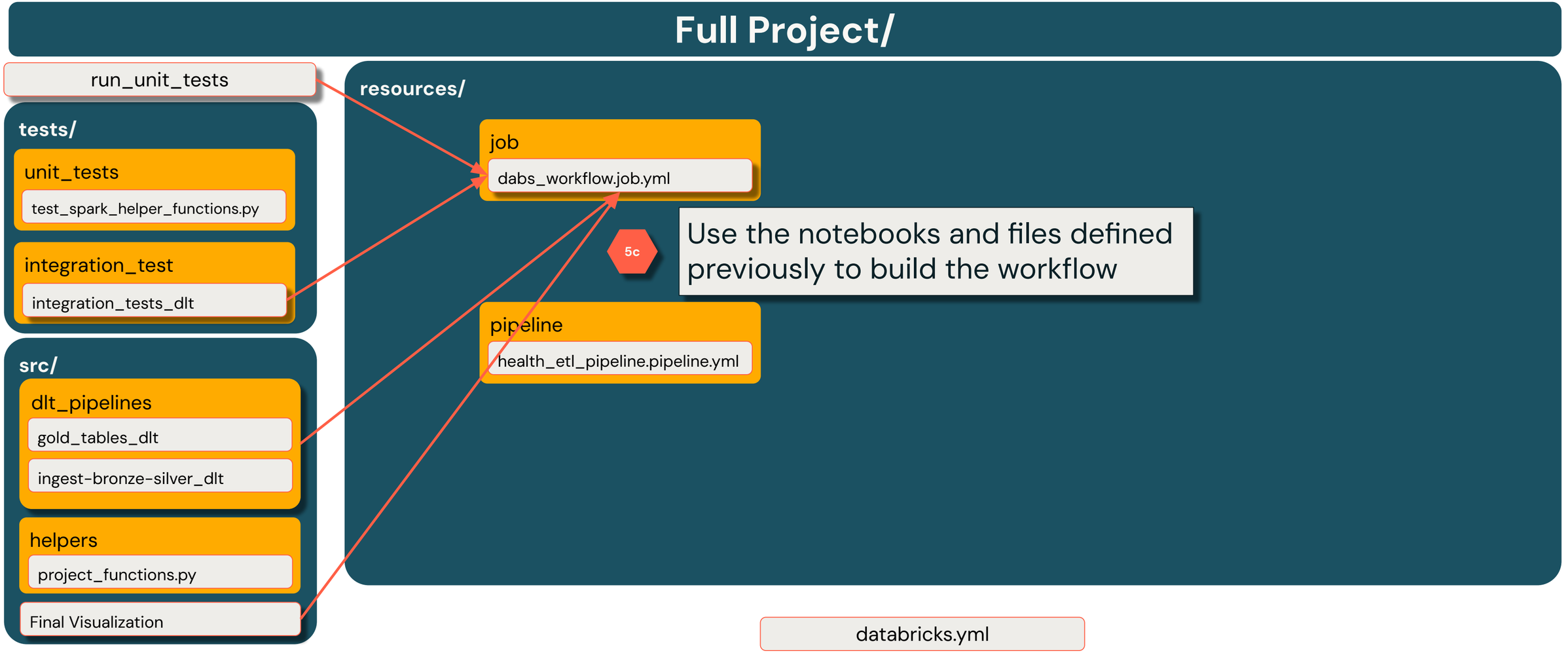
Full Project Outline

Folder Architecture



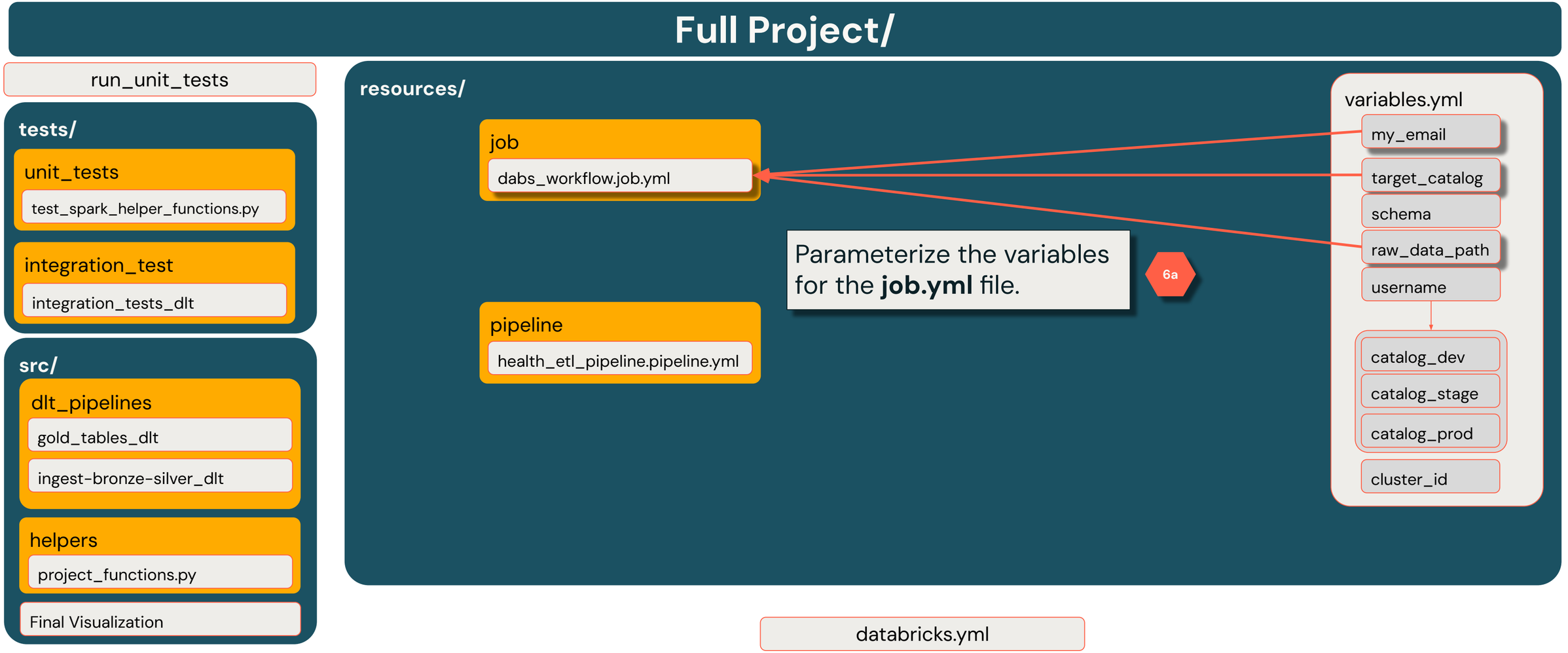
Full Project Outline

Folder Architecture



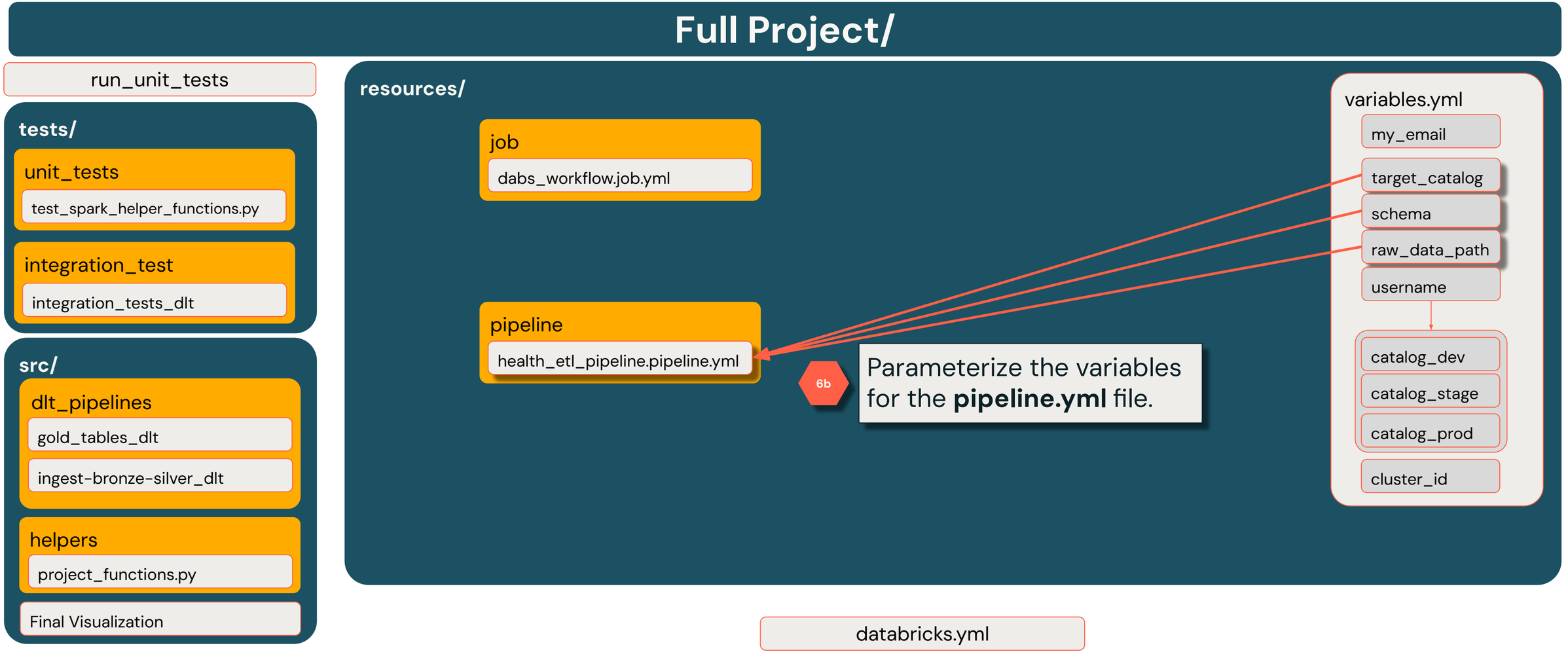
Full Project Outline

Folder Architecture



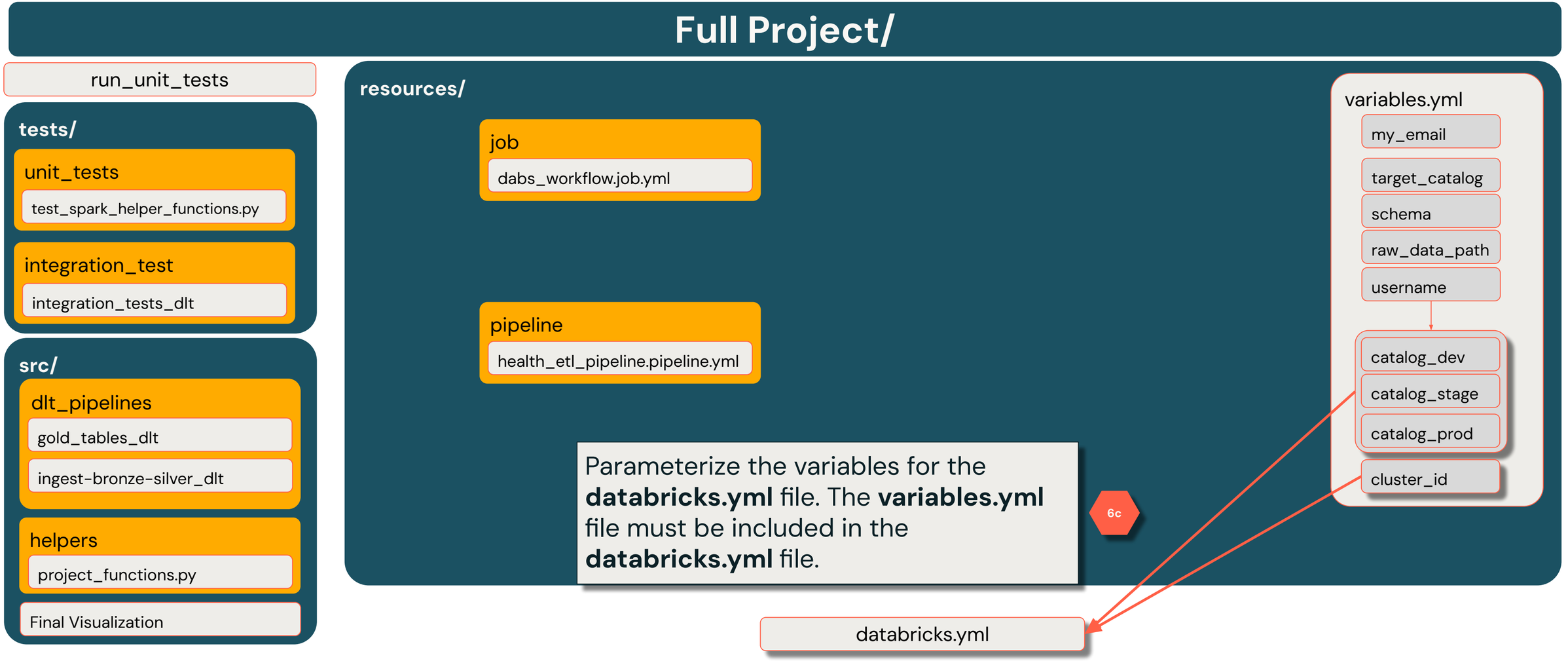
Full Project Outline

Folder Architecture



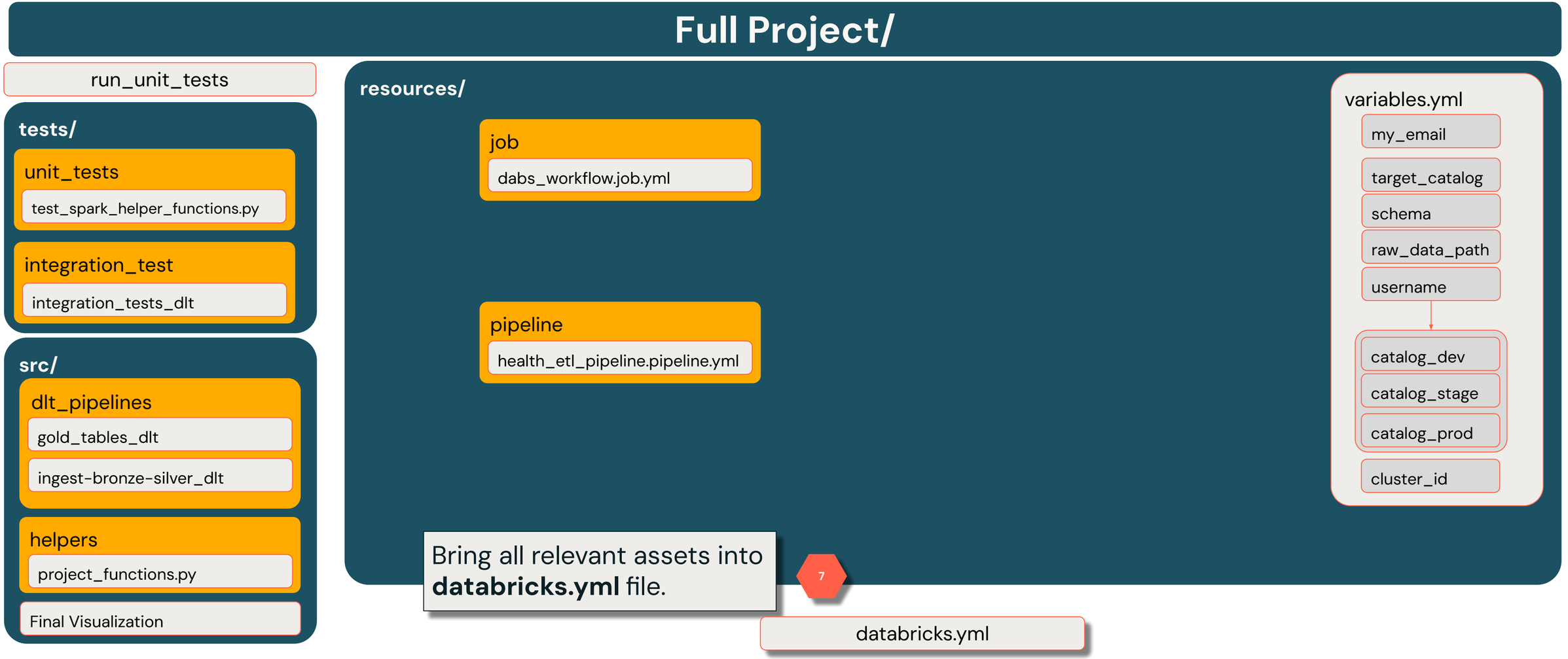
Full Project Outline

Folder Architecture



Full Project Outline

Folder Architecture

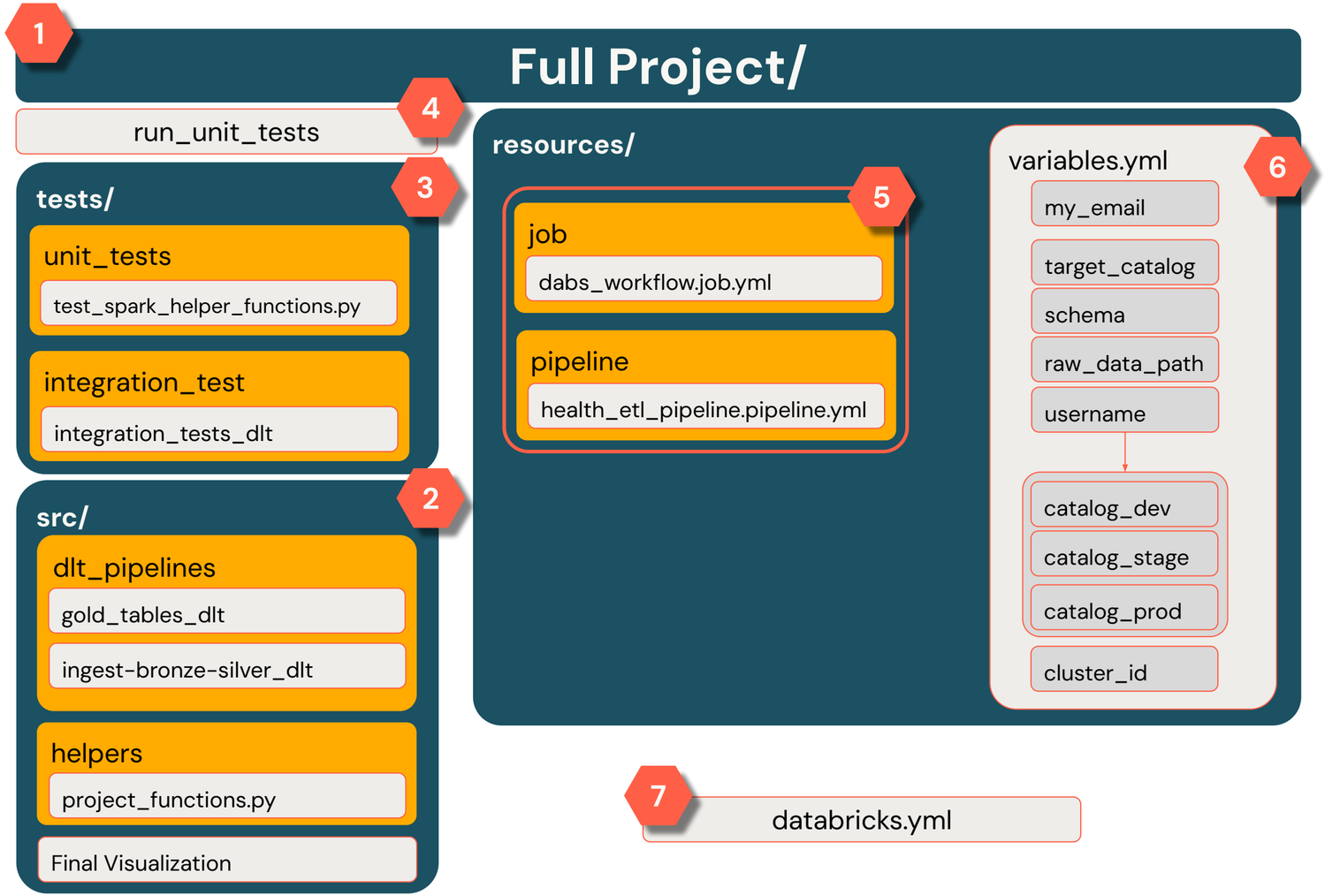


Full Project Outline

Summary

CI/CD with DABs Summary

1. Define root directory for the project
2. Incorporate tested notebooks – these notebooks should be tested in isolation prior to migrating to DABs
3. Set up a folder for isolating different tests, e.g. unit tests and integration tests
4. Define any other notebooks
5. Define YAML files for jobs and DLT pipeline tasks
6. Parameterize all YAML files with variables.yml if possible
7. Define databricks.yml file

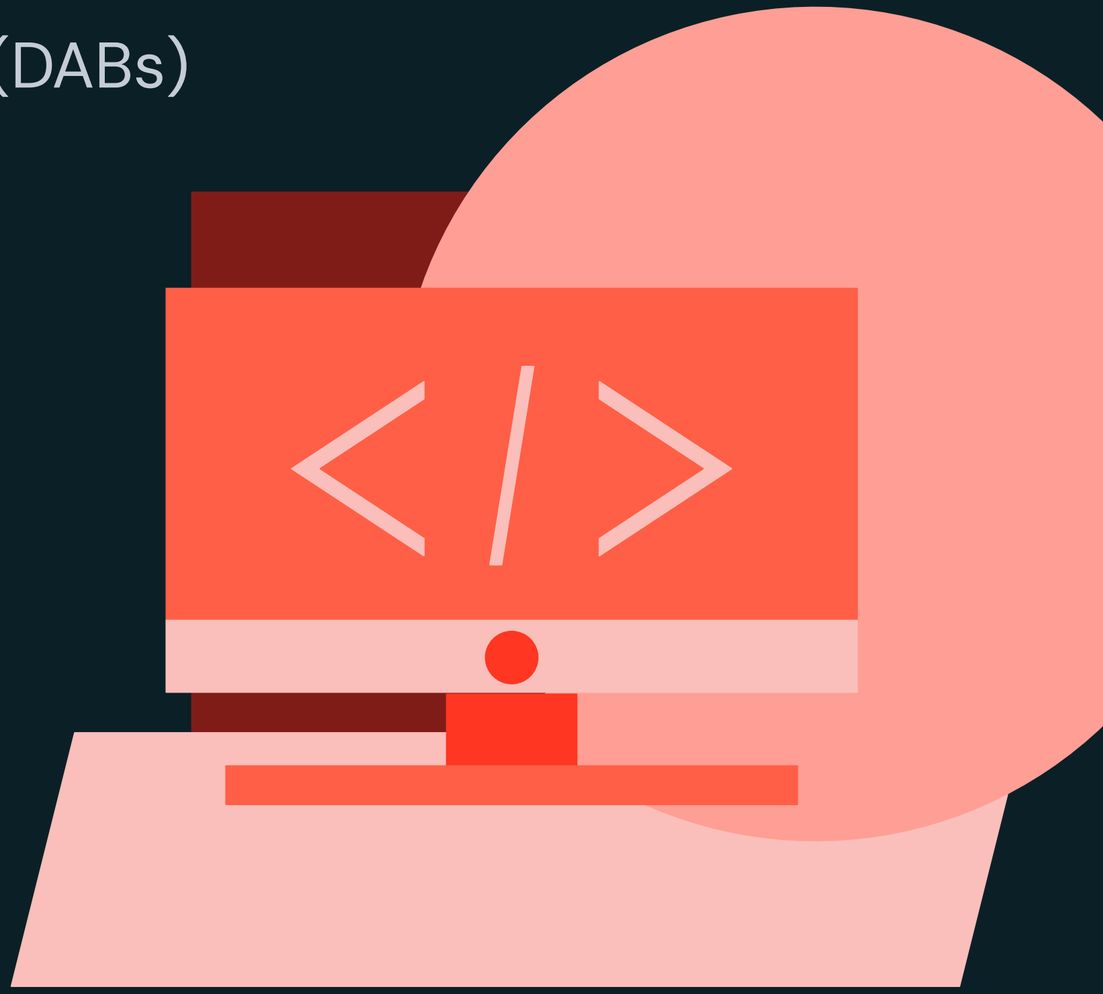


databricks

Deployment with Databricks Asset Bundles (DABs)

LAB EXERCISE

Adding ML to Engineering Workflows with DABs



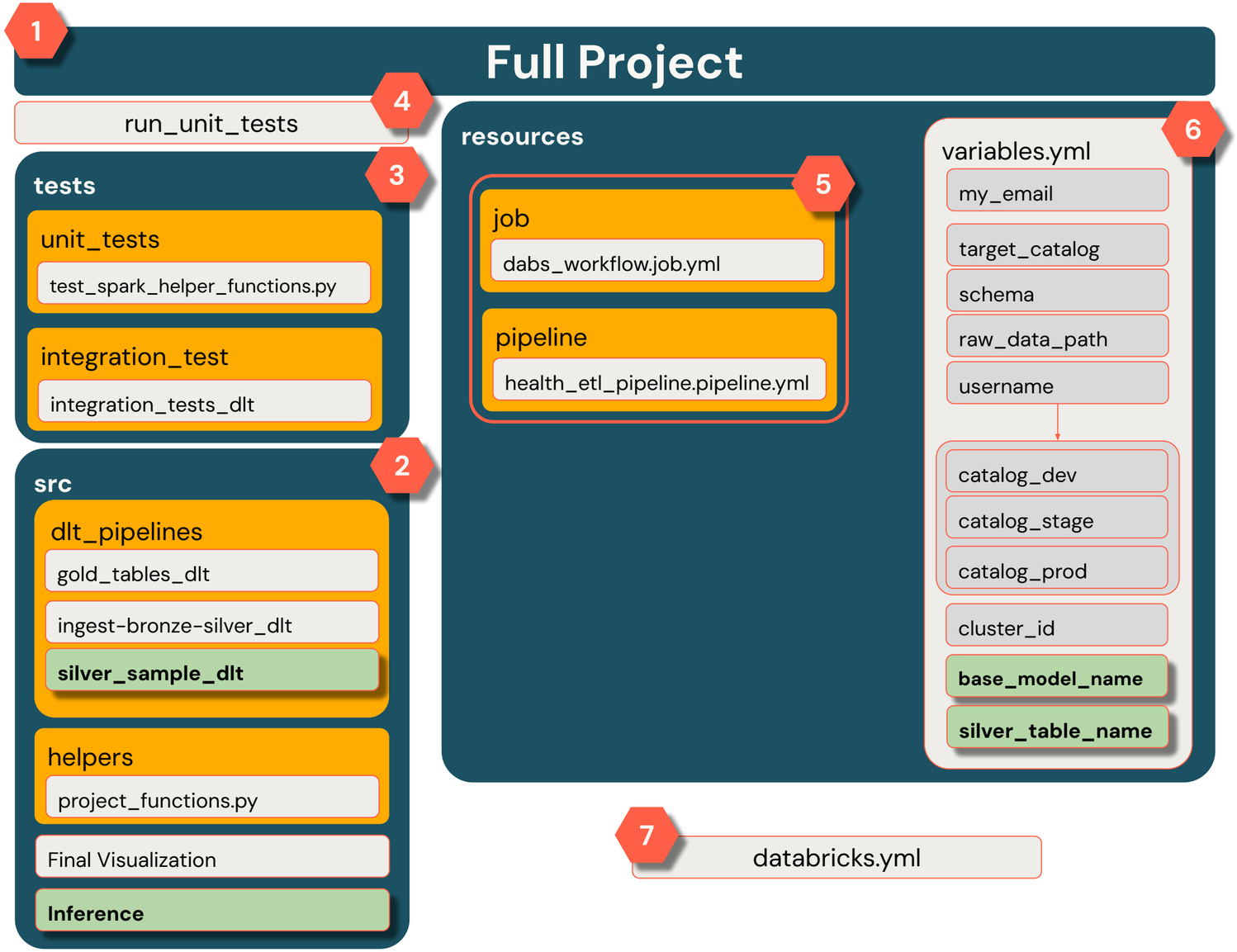
Notebook: /07L - Adding ML to Engineering Workflows with DABs/Lab - Make a Machine Learning Task

Lab Outline

Incorporate an ML Task

CI/CD with DABs Summary

1. Define root directory for the project
2. Incorporate tested notebooks - these notebooks should be tested in isolation prior to migrating to DABs
3. Set up a folder for isolating different tests, e.g. unit tests and integration tests
4. Define any other notebooks
5. Define YAML files for jobs and DLT pipeline tasks
6. Parameterize all YAML files with variables.yml if possible
7. Define databricks.yml file



databricks

Doing More with DABs

Automated Deployment with Databricks Asset Bundles

Section Learning Objectives

- Understand core components of developing locally with Visual Studio Code (VS Code)
- Understand how to use Databricks Connect with VS Code
- Discuss the Databricks Extension within VS Code
- Understand core principles of best practices for CI/CD
- Discuss common development patterns with GitHub Actions and common 3rd party tools

Agenda

Doing More with DABs
• Developing Locally with Visual Studio Code (VSCode)
• CI/CD Best Practices for Data Engineering
• Next Steps: Automated Deployment with GitHub Actions

databricks

Doing More with DABs

LECTURE

Developing Locally with Visual Studio Code (VSCode)

Developing Locally with (VSCode)

Developer Tools Overview

Databricks CLI

- Ideal for shell scripting & lightweight command-line tasks
- Useful in development and CI/CD processes
- Supports unified authentication (OAuth, PATs, etc.)
- More interactive debugging compared to Notebooks

Databricks Connect V2

Databricks VSCode Extension

Developing Locally with (VSCode)

Developer Tools Overview

Databricks CLI

- Ideal for shell scripting & lightweight command-line tasks
- Useful in development and CI/CD processes
- Supports unified authentication (OAuth, PATs, etc.)
- Limited interactive debugging compared to IDEs

Databricks Connect V2

- Runs Apache Spark code remotely on a Databricks cluster from a local environment
- Ideal for interactive debugging and remote Spark job execution
- Easy setup with pip install databricks-connect>= your version

Databricks VSCode Extension

Developing Locally with (VSCode)

Developer Tools Overview

Databricks CLI

- Ideal for shell scripting & lightweight command-line tasks
- Useful in development and CI/CD processes
- Supports unified authentication (OAuth, PATs, etc.)
- Limited interactive debugging compared to IDEs

Databricks Connect V2

- Runs Apache Spark code remotely on a Databricks cluster from a local environment
- Ideal for interactive debugging and remote Spark job execution
- Easy setup with pip install databricks-connect>=13.0.28

Databricks VSCode Extension

- Integrates Databricks into VS Code for batch and interactive development
- Simplifies setup with resource explorers and Databricks CLI
- Supports running and debugging Python files on Databricks clusters
- Keeps developers within VS Code, and provides DAB features

Databricks Extension for (VSCode)

Overview

Simple Setup

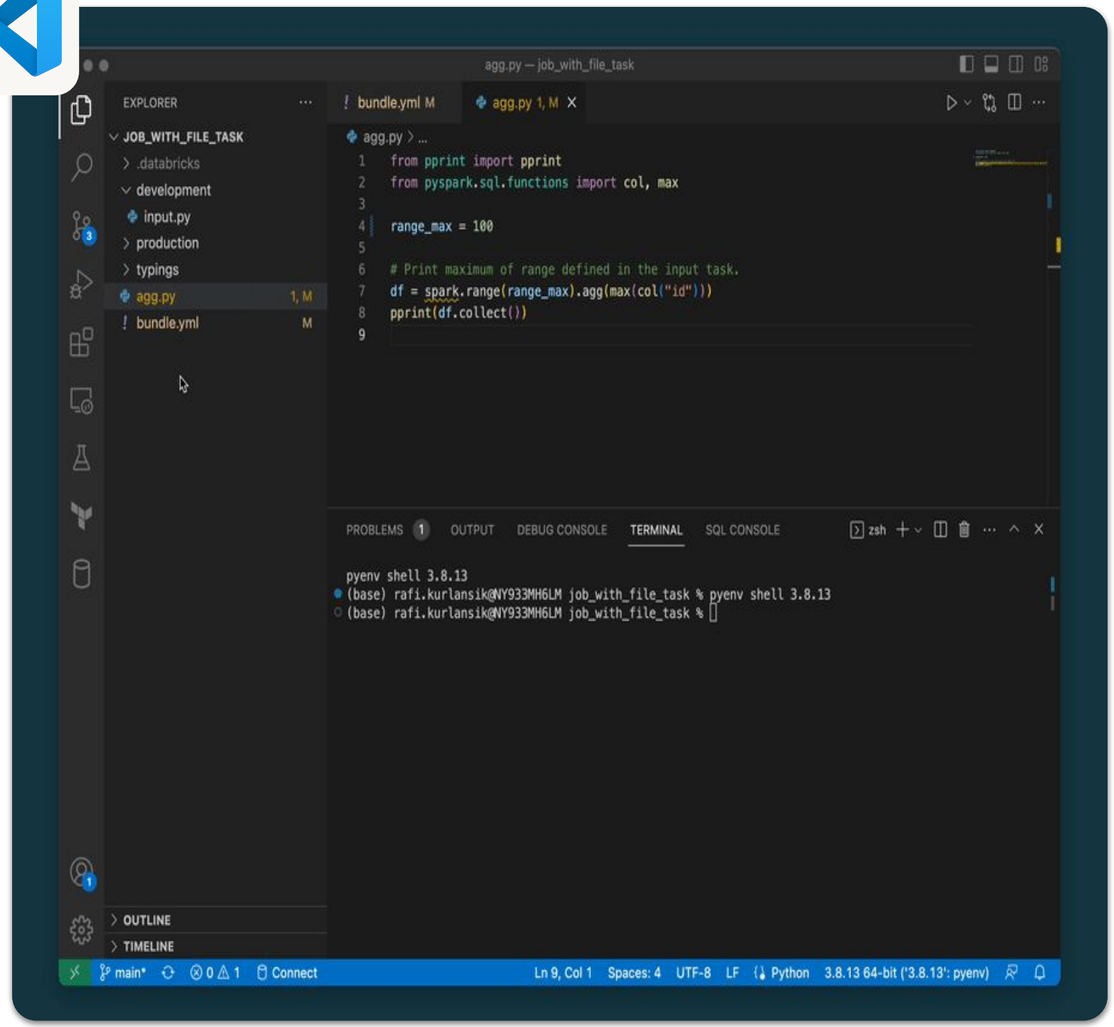
Find us on the VS Code Marketplace and get connected to compute in minutes

Native Experience

Write code using the productivity features you love from VS Code

Run on Databricks

Execute batch workloads or start interactively debugging directly from your IDE

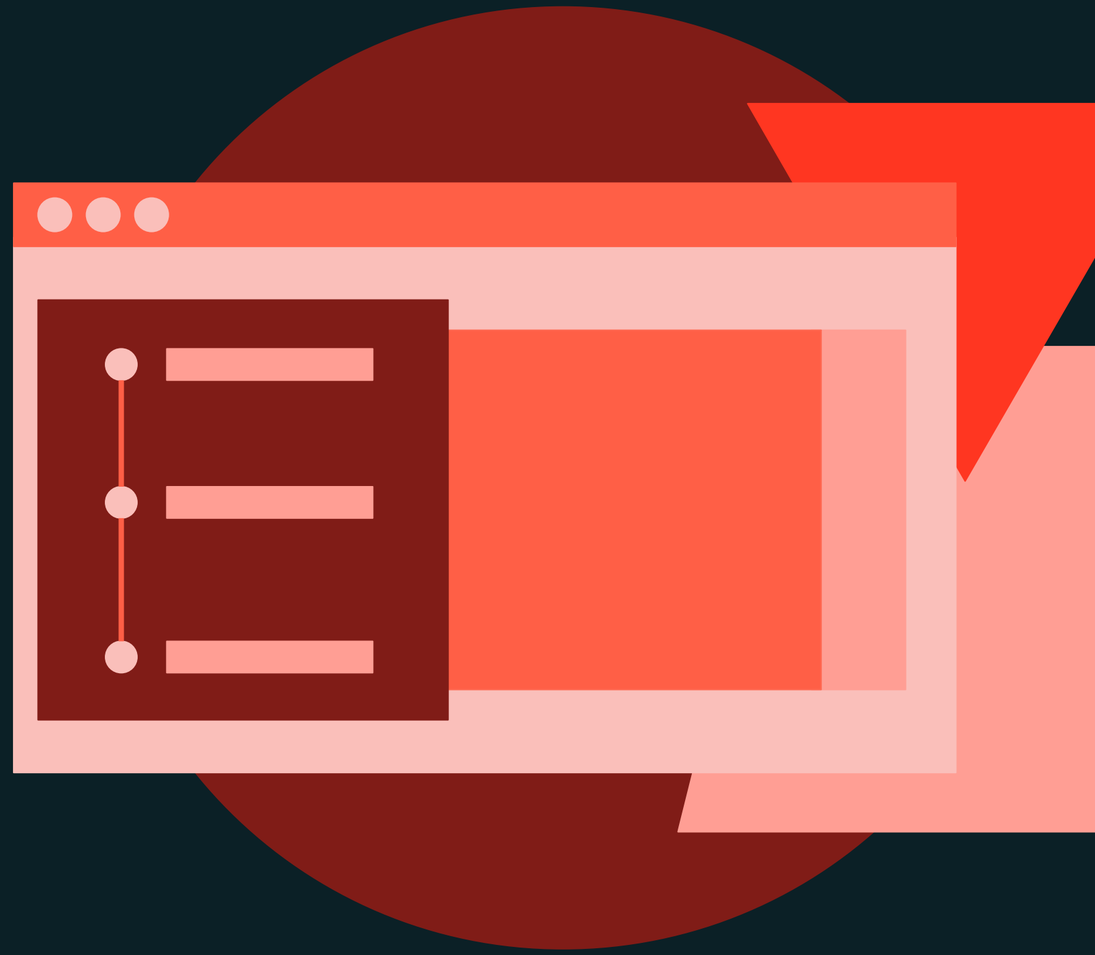


databricks

Module Name

DEMONSTRATION

Using VSCode with Databricks



Notebook: /08 Bonus - Using VSCode with Databricks/08 Bonus - Using VSCode with Databricks

databricks

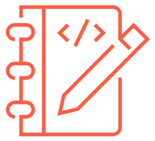
Doing More with DABs

LECTURE

CI/CD Best Practices for Data Engineering

CI/CD Best Practices for Data Engineering

Summary



Define a Clear Testing Strategy

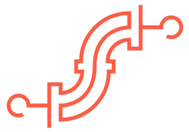
Establish and automate unit, integration and end to end tests

Automate Deployment with DABs

Use DABs for consistent, repeatable, and automated deployments in Databricks

Implement a Version Control Strategy

Use a version control system like git to manage code and define branching strategies



Monitor and Optimize CI/CD Pipelines

Continuously monitor the performance of your CI/CD pipelines

Establish a Code Review Process

Set up mandatory code reviews to ensure the quality of changes before they are merged

CI/CD Best Practices for Data Engineering

Summary

Implementing CI/CD

Requires alignment across teams and establishment of robust processes that ensure continuous improvement and quality

For more information check out the Best practices for operational excellence

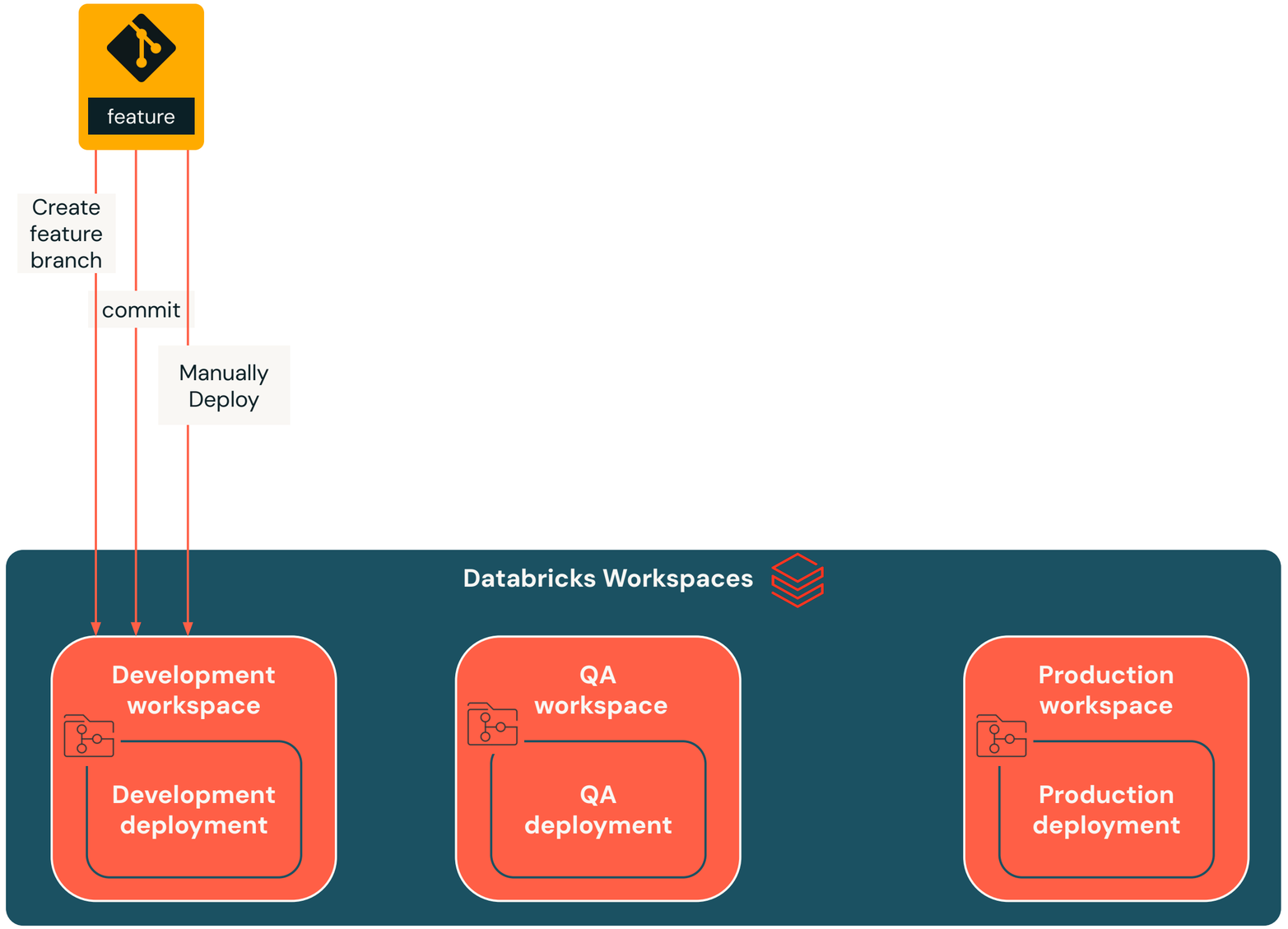
databricks

Doing More with DABs

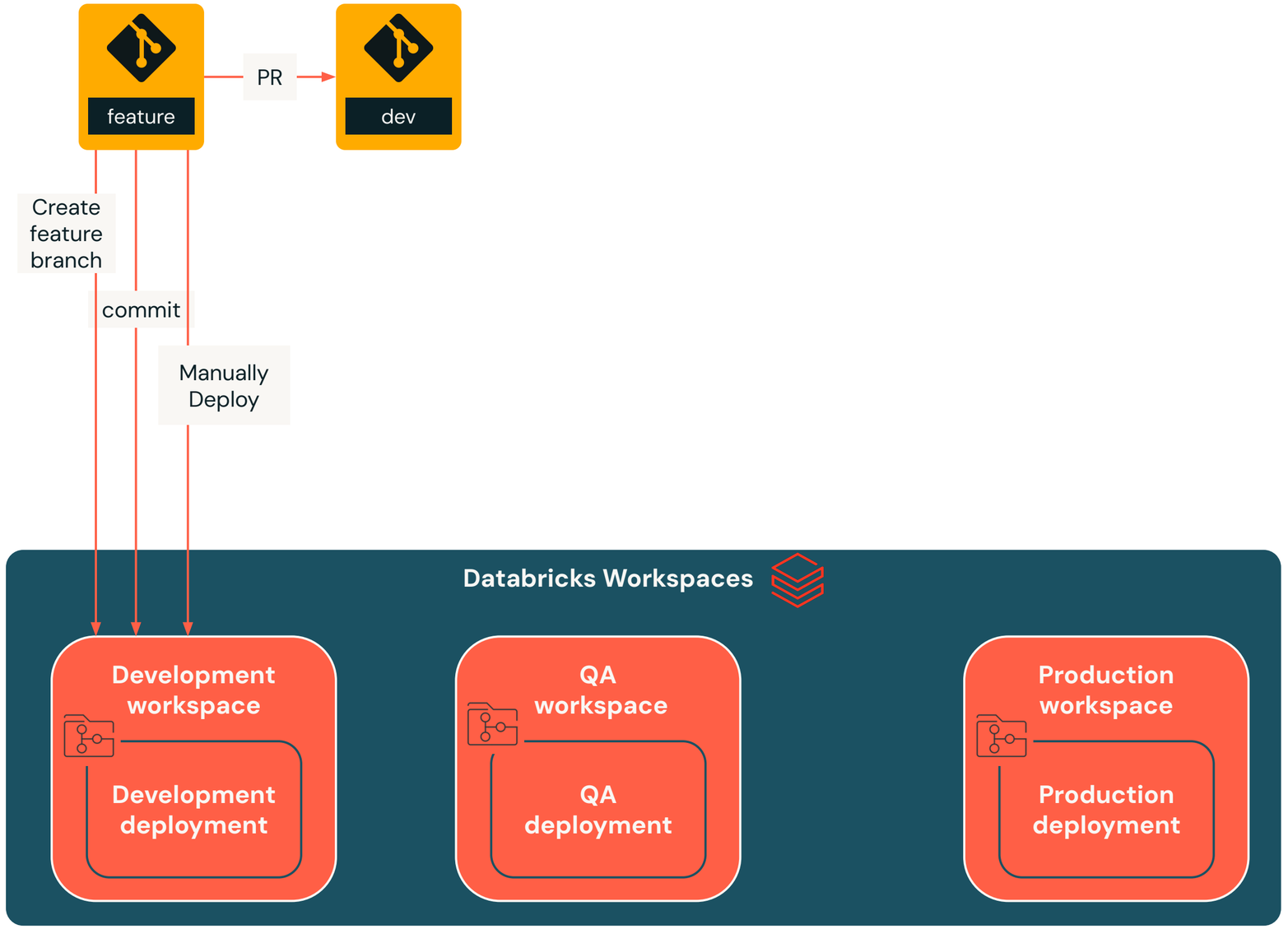
LECTURE

Next Steps: Automated Deployment with GitHub Actions

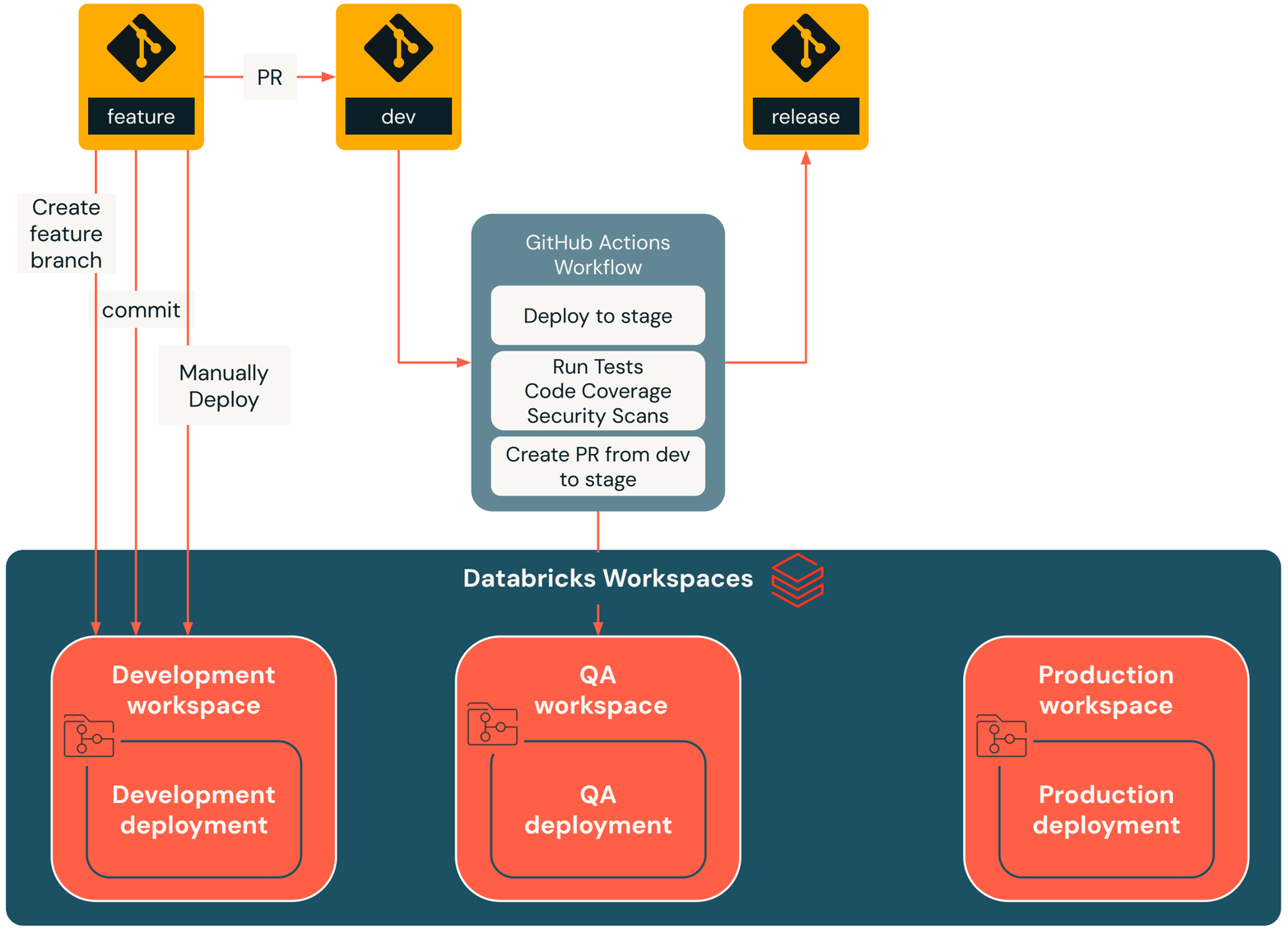
Deployment Patterns with DABs and GitHub Actions



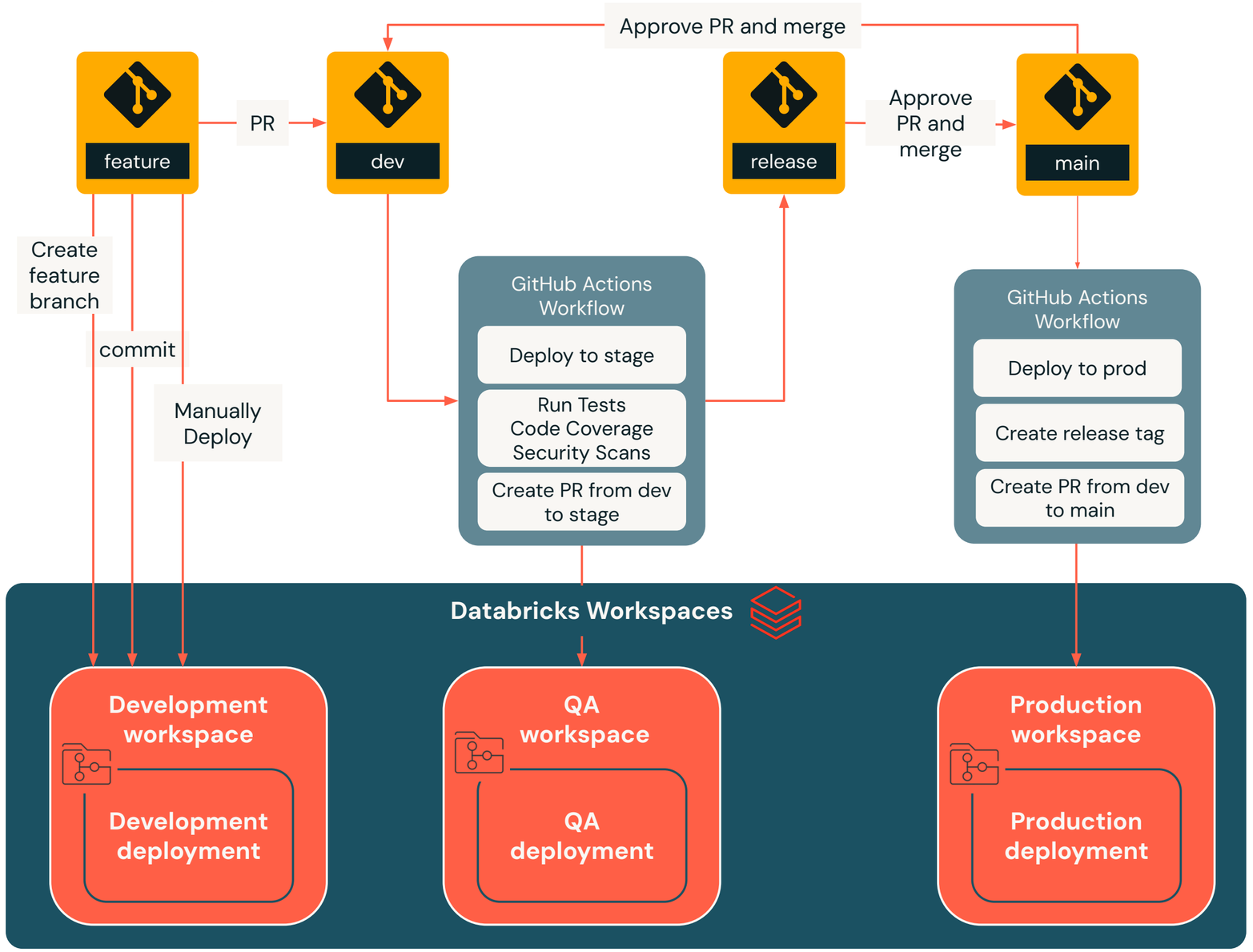
Deployment Patterns with DABs and GitHub Actions



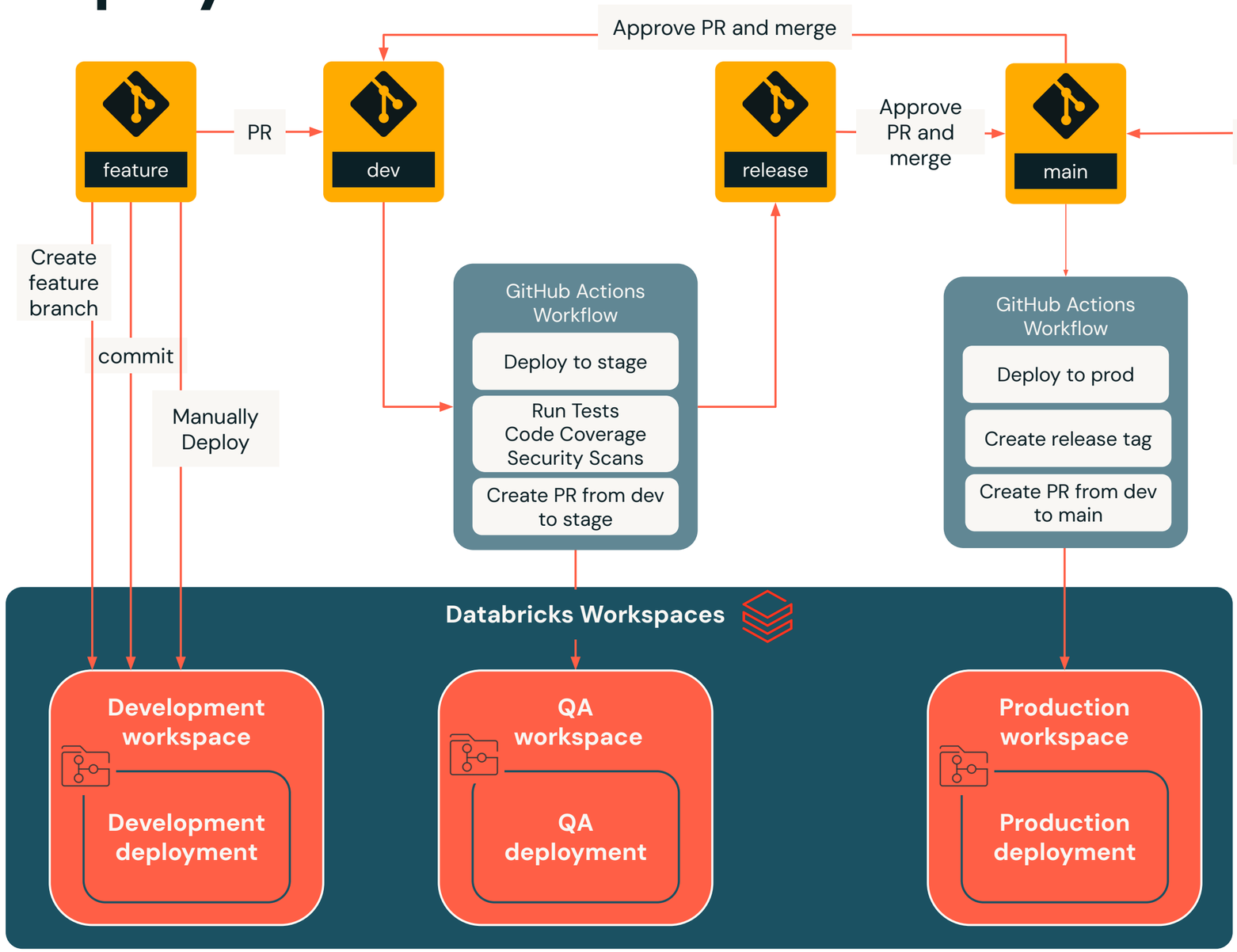
Deployment Patterns with DABs and GitHub Actions



Deployment Patterns with DABs and GitHub Actions



Deployment Patterns with DABs and GitHub Actions



1. Easily integrate structure branching patterns like GitFlow with DABs.
2. Developers can create isolated feature branches and create a PR once development is complete and all assets are ready for QA.
3. After code review, merge branches into dev, triggering GitHub Actions workflows.
4. Critical production issues are handled through hotfix branches from main, which are merged back into both main and dev.

Overview of Git with Databricks

Definitions

Git is a free and open-source software framework designed to track changes in source code during software development

databricks