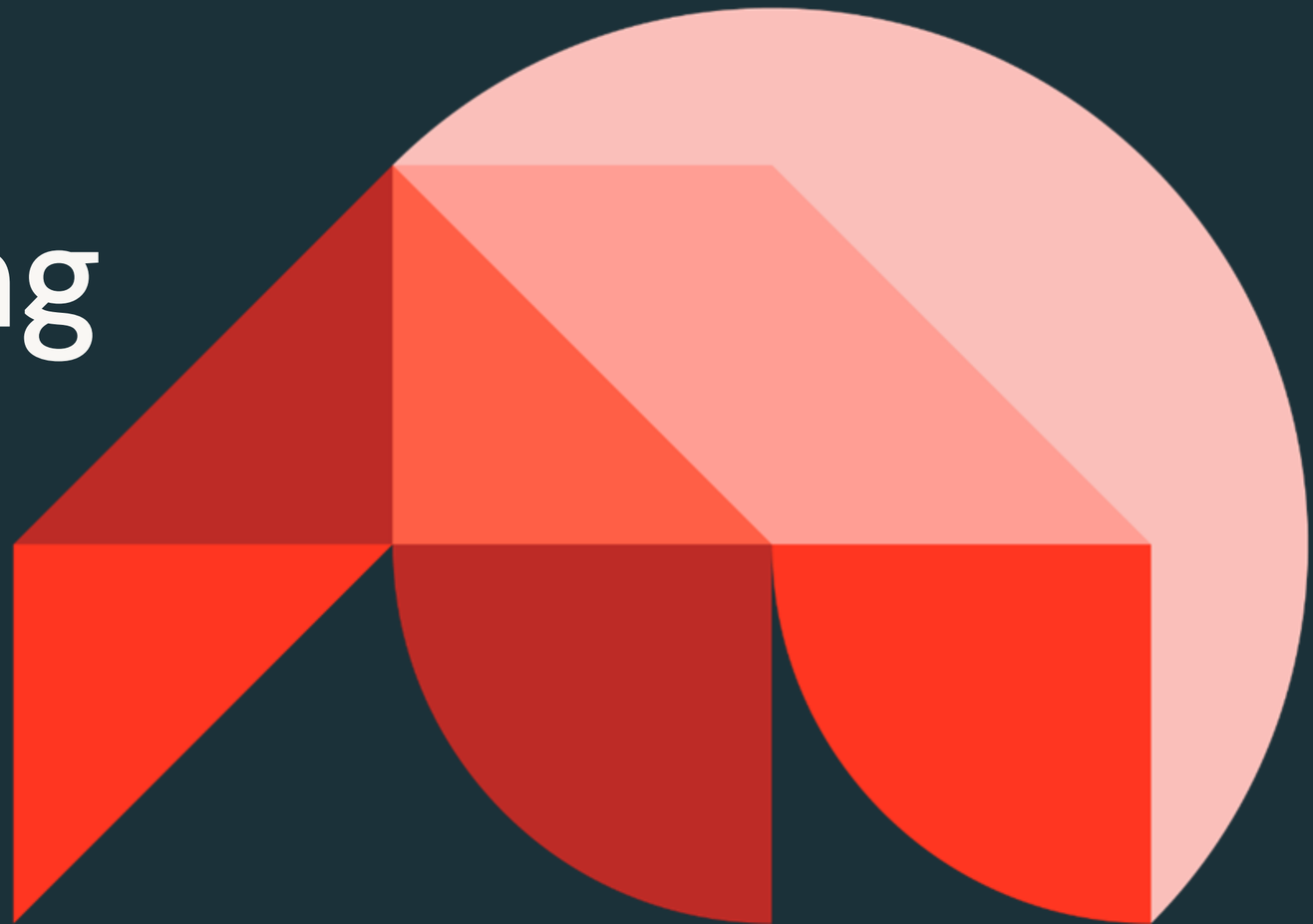
databricks

DevOps Essentials for Data Engineering



Databricks Academy

February, 2025

Course Learning Objectives

- Explain the core principles of software engineering best practices, including code quality, version control, documentation, and testing.
- Explain the principles of DevOps including core components, benefits, and the role of continuous integration and continuous delivery (CI/CD) in DevOps.
- Apply principles of modularity to organize PySpark code into reusable functions and components.
- Design and implement effective unit tests and integration tests for your Databricks data pipeline.
- Apply basic Git operations in Databricks using Git Folders to implement continuous integration practices effectively.
- Explain the different methods for deploying Databricks assets, including REST API, CLI, SDK, and Databricks Asset Bundles (DABs).

Agenda

Course Sections
• Software Engineering Best Practices, DevOps, and CI/CD Fundamentals
• Continuous Integration (CI)
• Introduction to Continuous Deployment (CD)

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
- Basic Git Knowledge

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
- Basic Git Knowledge

Experience Ingesting and Transforming Data

- Familiarity with PySpark for data processing and DataFrame manipulations.
- Experience in writing intermediate-level queries using SQL

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
- Basic Git Knowledge

Experience Ingesting and Transforming Data

- Familiarity with PySpark for data processing and DataFrame manipulations.
- Experience in writing intermediate-level queries using SQL

Knowledge of Python Programming

- Proficient in writing intermediate-level Python, including functions and classes.
- Ability to create, import, and utilize Python packages effectively.

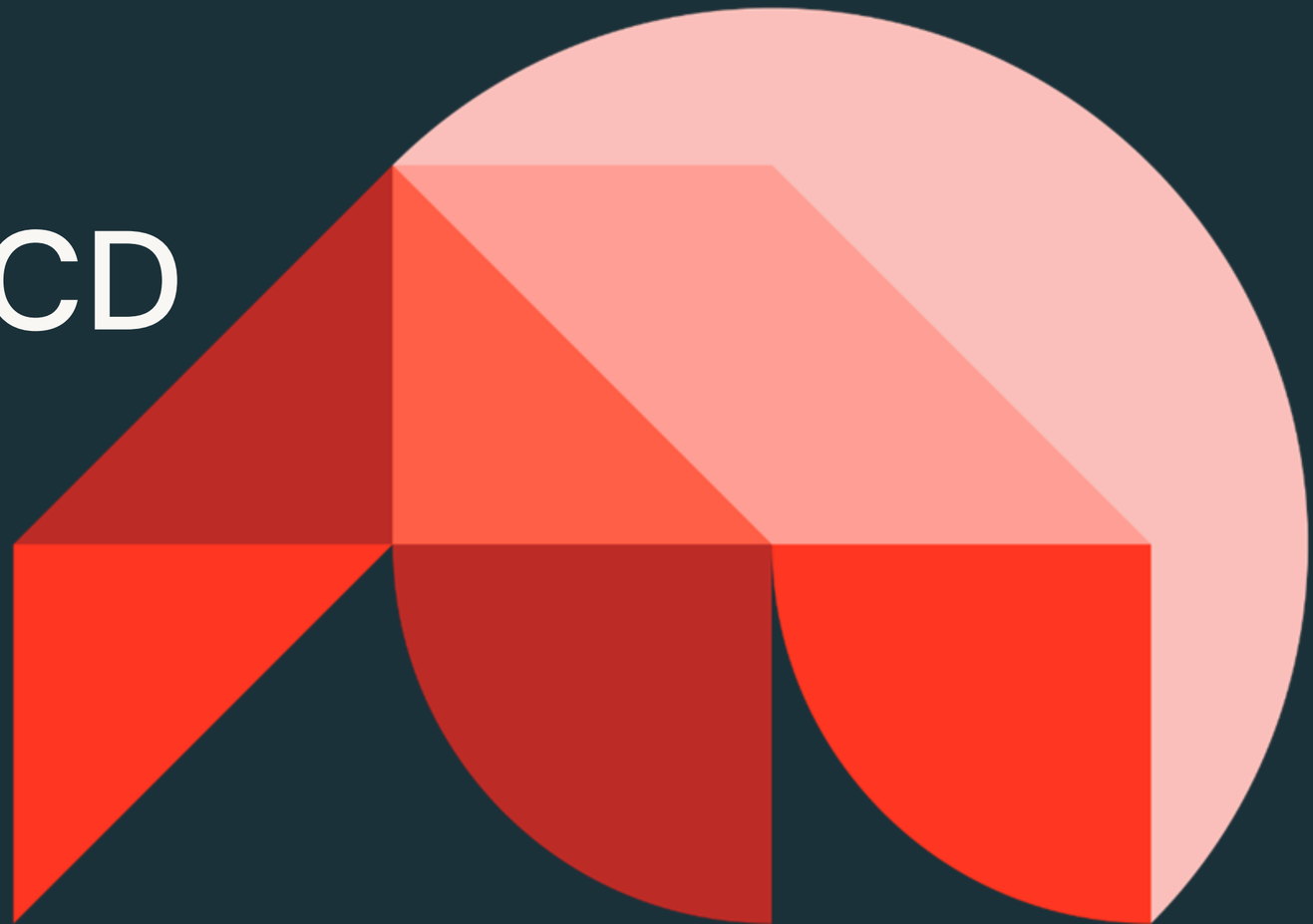
Lab Exercise Environment

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

Software Engineering, DevOps, and CI/CD Fundamentals



DevOps Essentials for Data Engineering

Section Learning Objectives

- Identify key software engineering practices like version control, testing, and code reviews.
- Understand how to modularize PySpark code into reusable and maintainable modules.
- Explain how DevOps and DataOps principles integrate development, operations, and data workflows for smoother collaboration and faster delivery.
- Understand the core concepts and benefits of CI/CD in modern software development, focusing on automation, continuous integration, and continuous delivery of data pipelines.

Agenda

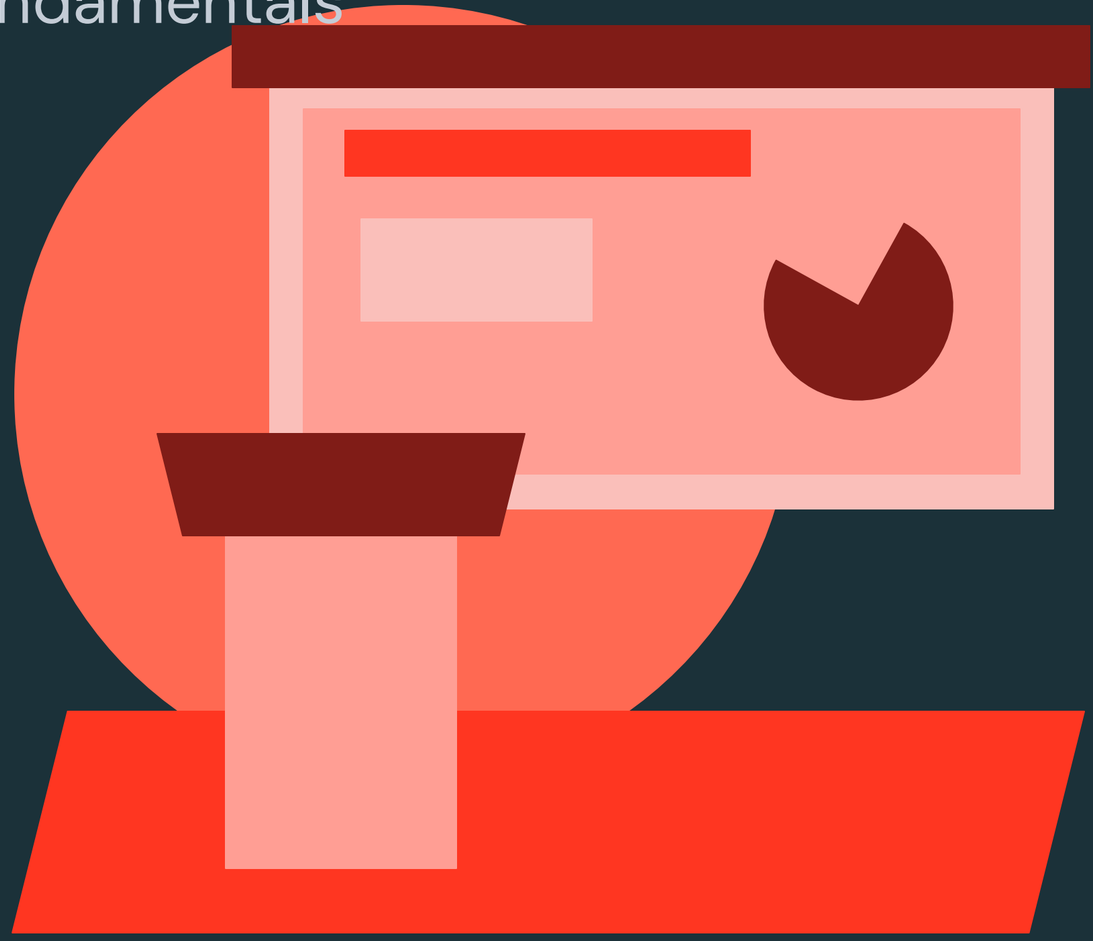
Software Engineering, DevOps, and CI/CD Fundamentals
• Introduction to Software Engineering (SWE) Best Practices
• Introduction to Modularizing PySpark Code
• DevOps Fundamentals
• The Role of CI/CD in DevOps
• Knowledge Check/Discussion

databricks

Software Engineering, DevOps, and CI/CD Fundamentals

LECTURE

Introduction to Software Engineering (SWE) Best Practices



Introduction to SWE Best Practices

To build reliable data pipelines, we can learn from software engineering best practices

Software Engineering Best Practices

Building Data Pipelines

Following **best practices** in development ensures that your data pipelines are efficient, scalable, and maintainable

Coding Practices

You can use code **linting** tools to help aid in your coding practices.

Code Readability

Write code that is **easy** to understand, navigate, and maintain.

Naming Conventions

Use descriptive, **consistent** names for variables, functions, and classes.

Modular Design

Break down software into **smaller, reusable** components (functions)

Coding Practices

Document Code

- Improves Code Maintainability
- Enhances Collaboration
- Facilitates Knowledge Transfer

Coding Practices

Document Code

Automated Testing

1. Unit Tests

Verifies the functionality of a **single unit** in isolation

2. Integration Tests

Tests how **different** components or systems work together.

Coding Practices | Document Code | Automated Testing | **Version Control and Code Review**

Version Control

Tools like Git are essential for tracking changes, collaborating with team members, and maintaining a history of the codebase.

Code Review

Help catch bugs early and ensure adherence to coding standards.

Coding Practices | Document Code | Automated Testing | Version Control and Code Review | **CI/CD**

Continuous Integration (CI)

A practice where developers frequently **commit**, **build**, **test** and **release code** to a shared repository.

Main focus of this course

Continuous Deployment (CD)

Automating the **release** of code to production after passing automated tests.

Coding Practices | Document Code | Automated Testing | Version Control and Code Review | CI/CD | **Isolated Environments**

Workspaces

Utilizing multiple workspaces, one for each environment

DEV

STAGE

PROD

Catalogs

Utilizing multiple catalogs, one for each environment

Introduction to SWE Best Practices

Best Practices

Coding Practices	Document Code	Automated Testing	Version Control and Code Review	CI/CD	Isolated Environments

The focus is on **writing high-quality code**, **testing** it, and ensuring **scalability** and **maintainability**.

Software Engineering with Databricks

Tools Overview

Develop code and run unit tests in a Databricks Workspaces or locally using **Notebooks** or **Files (SQL, Python, Scala, etc.)**.

Utilize Databricks **Git folders** to provide version control and significantly improve the workflow.

Focus on using **Unity Catalog** within a single **Workspace** or **multiple Workspaces** to isolate your environments securely, providing the necessary data access.

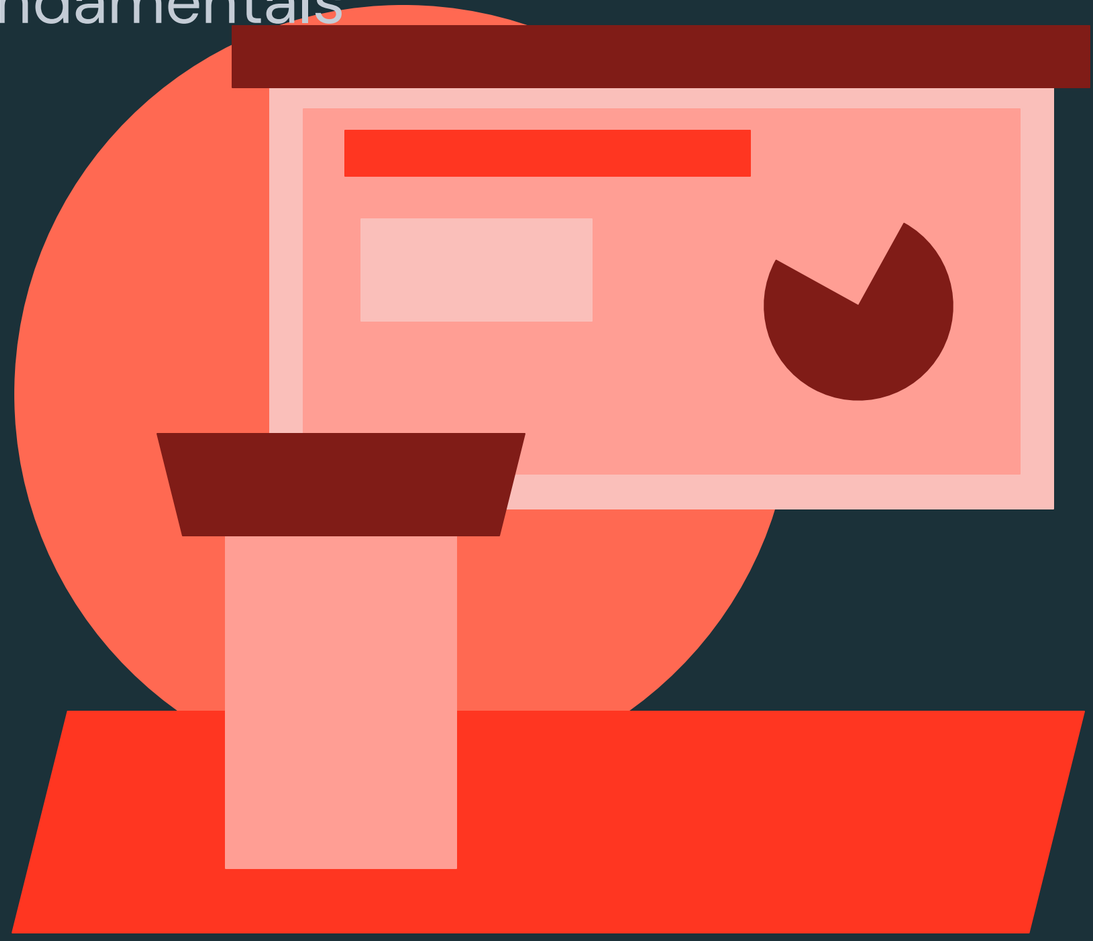
Get code **tested** & **deployed** via CI/CD pipelines using Databricks deployment tools to deploy to your desired environment automatically.

databricks

Software Engineering, DevOps, and CI/CD Fundamentals

LECTURE

Introduction to Modularizing PySpark Code



Modularizing PySpark Code

Non-Modularized Code (Before)

```
# Load data
df = (spark
      .read
      .csv("health.csv",
           header=True,
           inferSchema=True))

# Create column
df = (
 df
 .withColumn("NewColumn",
  when(col("Column") == 0, 'Normal')
  .otherwise('Unknown'))
)
```

Issues

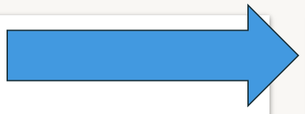
- Everything is in **one block**, making it harder to modify or test specific parts (e.g., loading data, adding new columns).

- Code **duplication** could occur if the same operations are needed elsewhere in the project.

Modularizing PySpark Code

Modularized Code (After)

```
# Load data
df = (spark
      .read
      .csv("health.csv",
           header=True,
           inferSchema=True))

# Create column
df = (
 df
 .withColumn("NewColumn",
  when(col("Column") == 0, 'Normal')
  .otherwise('Unknown'))
)
```



```
def load_data(file_path):
    return (spark
            .read
            .csv(file_path,
                 header=True,
                 inferSchema=True))


def add_new_col(df, new, s_col):
    return (df
            .withColumn(new,
               when(col(s_col) == 0, 'Normal')
               .otherwise('Unknown')))
```

Modularizing PySpark Code

Modularized Code Benefits

Benefits

- **Easier Maintenance** by updating only specific functions without changing the entire script.
- **Reuse** functions in different projects.
- **Testing** individual functions through unit tests to ensure code reliability.

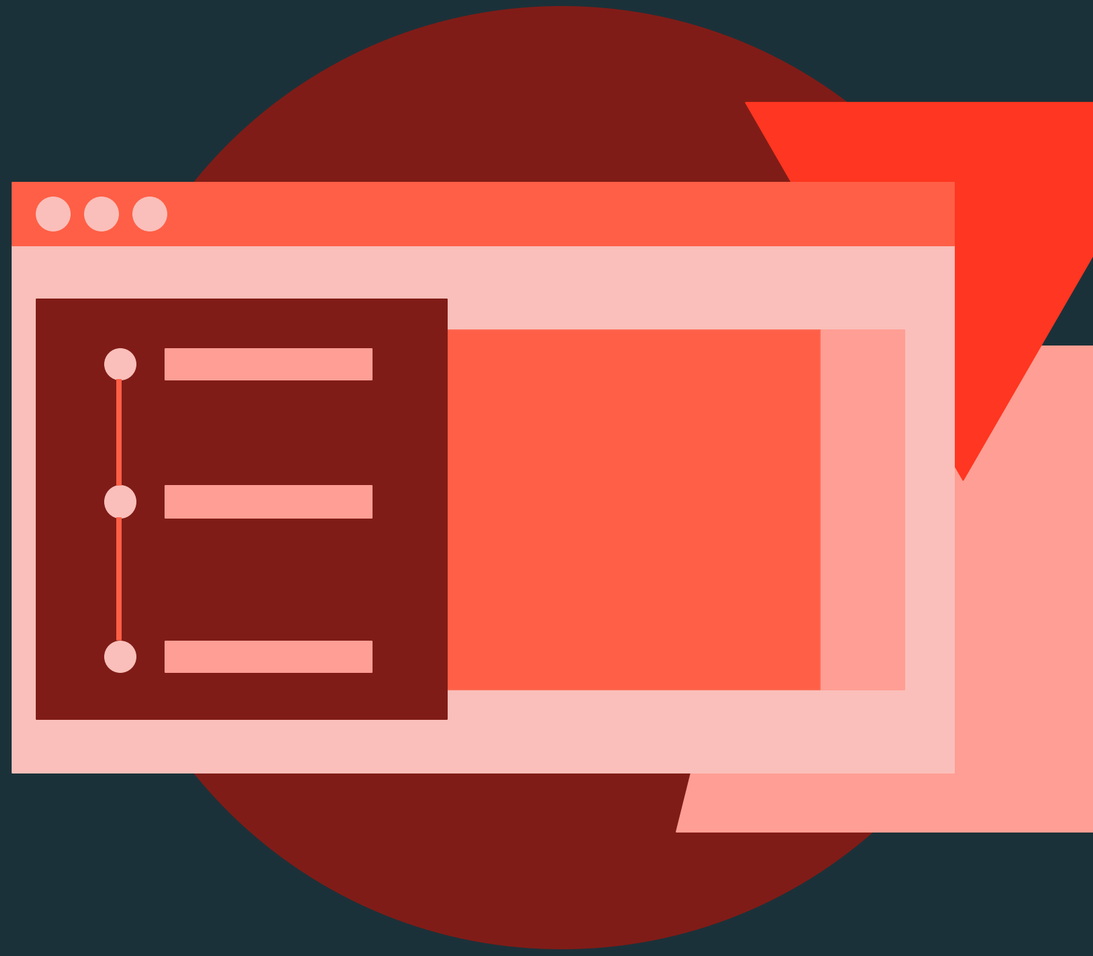
```
def load_data(file_path):
    return (spark
            .read
            .csv(file_path,
                 header=True,
                 inferSchema=True))


def add_new_col(df, new, s_col):
    return (df
            .withColumn(new,
               when(col(s_col) == 0, 'Normal')
               .otherwise('Unknown')))
```

databricks

Continuous Integration (CI)

DEMONSTRATION

Modularizing PySpark Code



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

Notebook: Course Notebooks/M02 - CI/2.1 - Modularizing PySpark Code

databricks

Continuous Integration (CI)

LAB EXERCISE

Modularize PySpark Code

Notebook: Course Notebooks/M02 - CI/2.2L - Modularize PySpark Code

databricks

Software Engineering, DevOps, and CI/CD Fundamentals

LECTURE

DevOps Fundamentals

DevOps Fundamentals

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

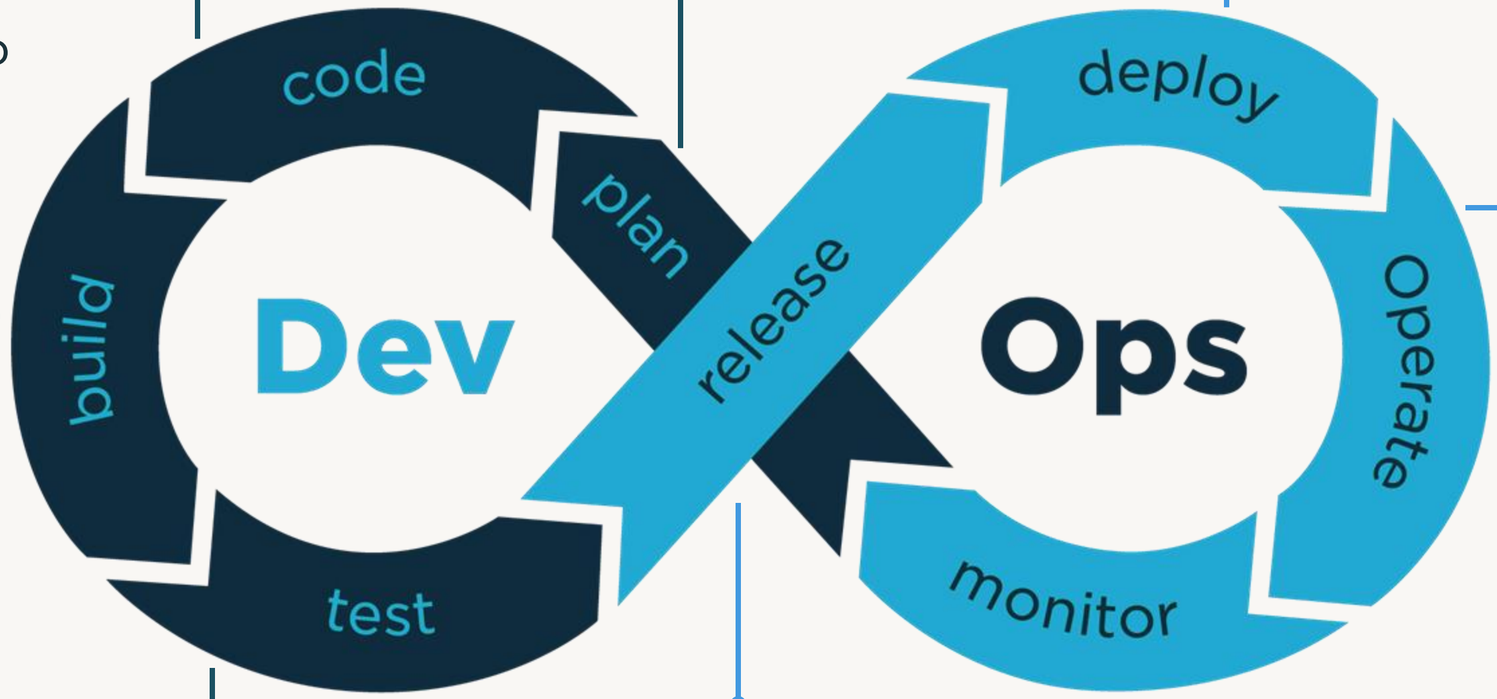
- faster deployments
- improved collaboration
- enhanced reliability
- better scalability

CODE the project

PLAN the project's features and requirements

DEPLOY the project to production environments

BUILD the code into executable files



OPERATE the released application

TEST the code works as expected by running **automated** tests

RELEASE the packaged application application ensuring it's ready for roll out

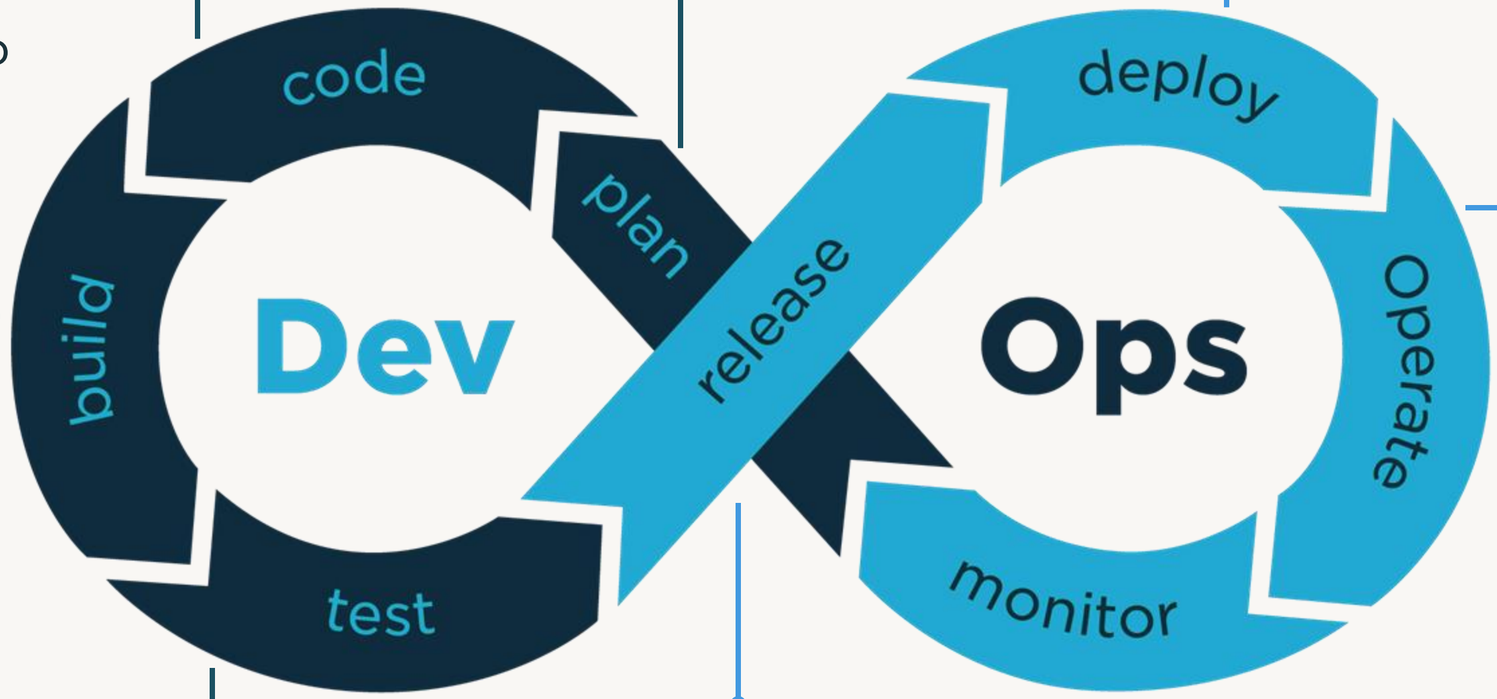
MONITOR the performance and gather feedback

CODE the project

PLAN the project's features and requirements

DEPLOY the project to production environments

BUILD the code into executable files



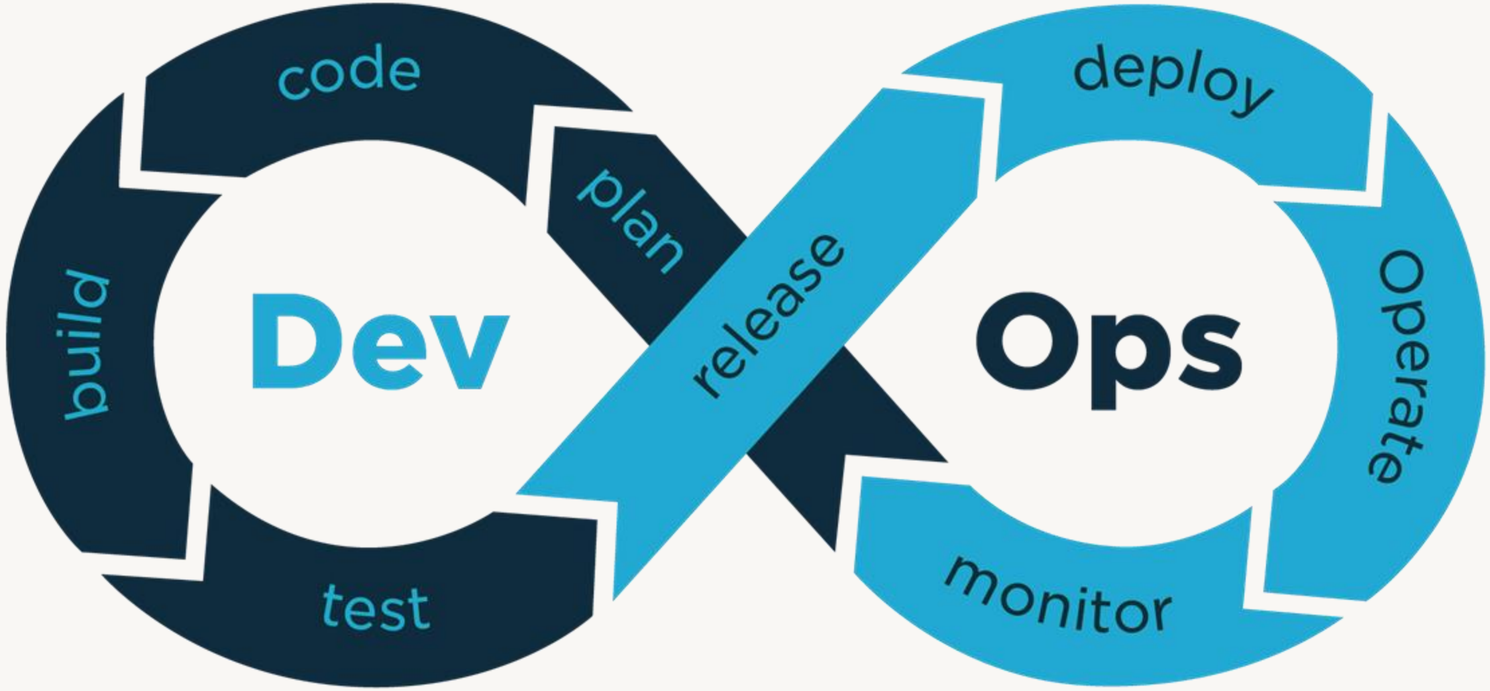
OPERATE the released application

TEST the code works as expected by running **automated** tests

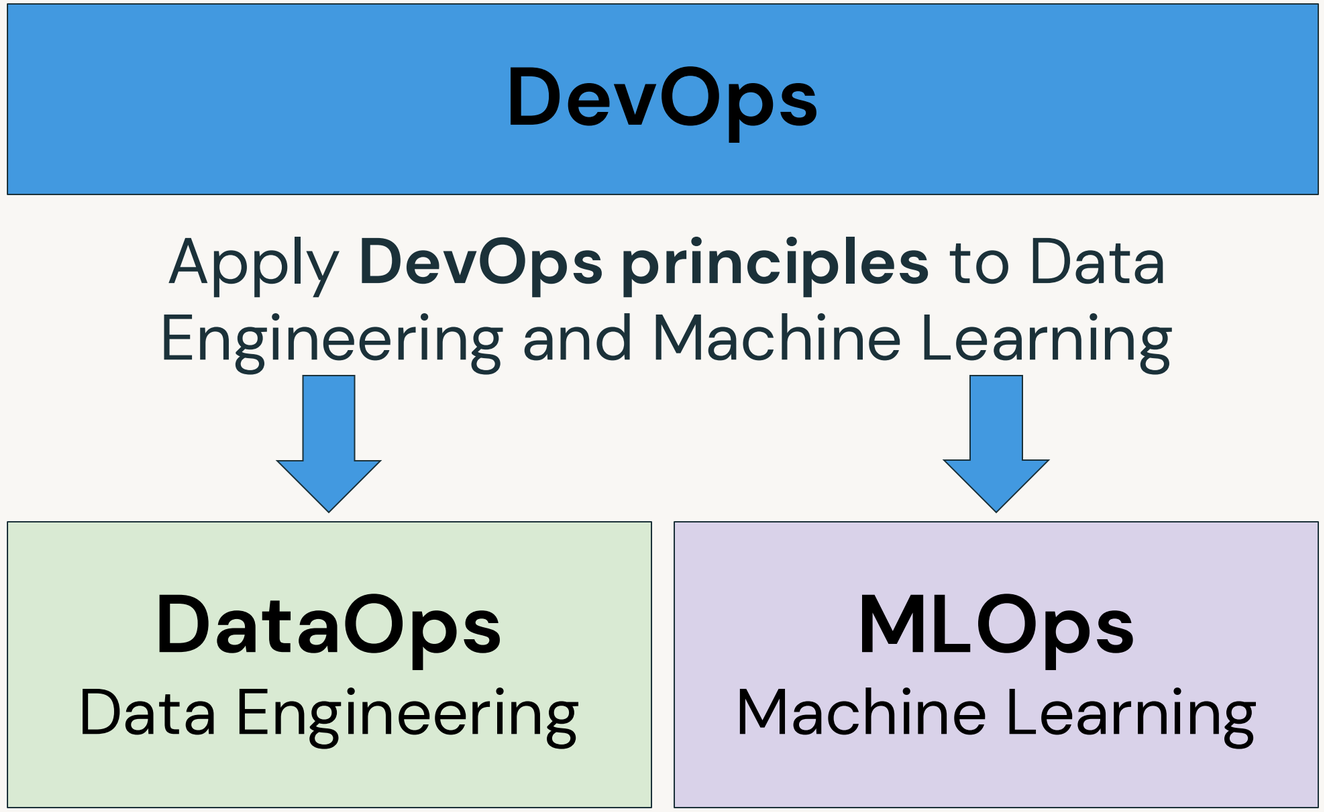
RELEASE the packaged application application ensuring it's ready for roll out

MONITOR the performance and gather feedback

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



DevOps For Data Engineering and Machine Learning



DevOps, DataOps and MLOps

A set of practices, processes, and technologies

DevOps

Software development and IT operations

- Automate CI/CD
- Enable continuous code testing
- Version control
- Establish Production-grade workflows
- Orchestration & Automation
- Monitor system performance

DataOps

Building quality data pipeline processes

- Optimize data processing
- Centralize data discovery, management, and governance
- Establish traceable data lineage and monitoring
- Enhance collaboration across teams
- Monitor data quality

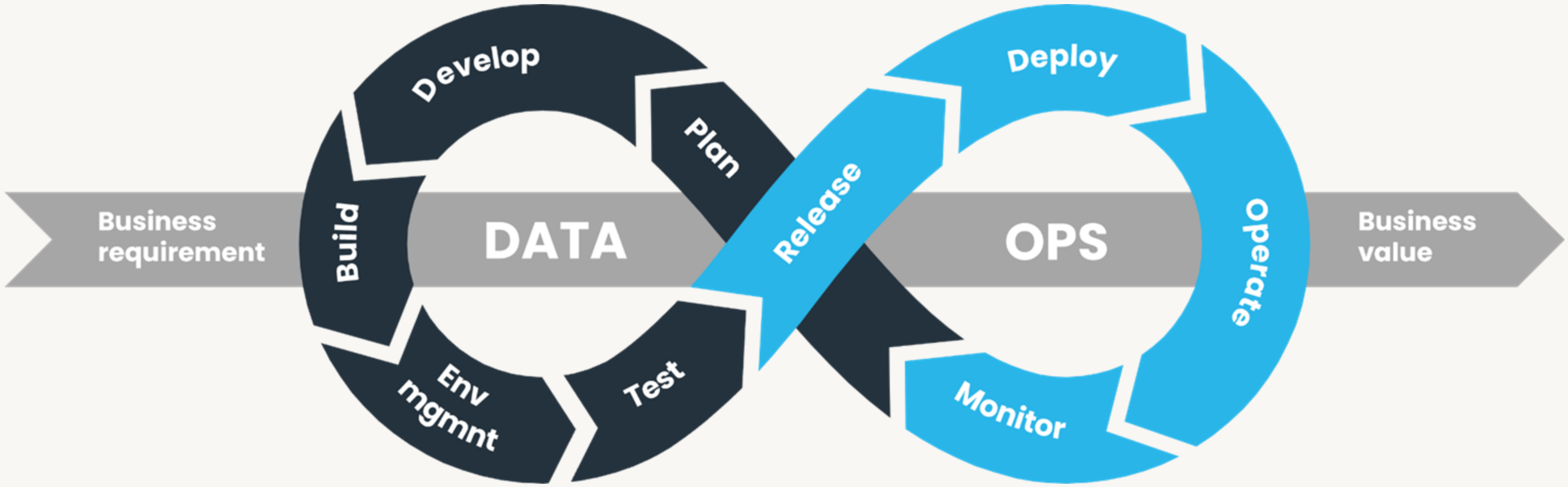
MLOps

ML model development and deployment

- Treating model code as software
- Treating models as data
- Manage the model lifecycle
- Monitor Model Performance

Outside the scope of this course.

DataOps = DevOps for Data Engineering



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

databricks

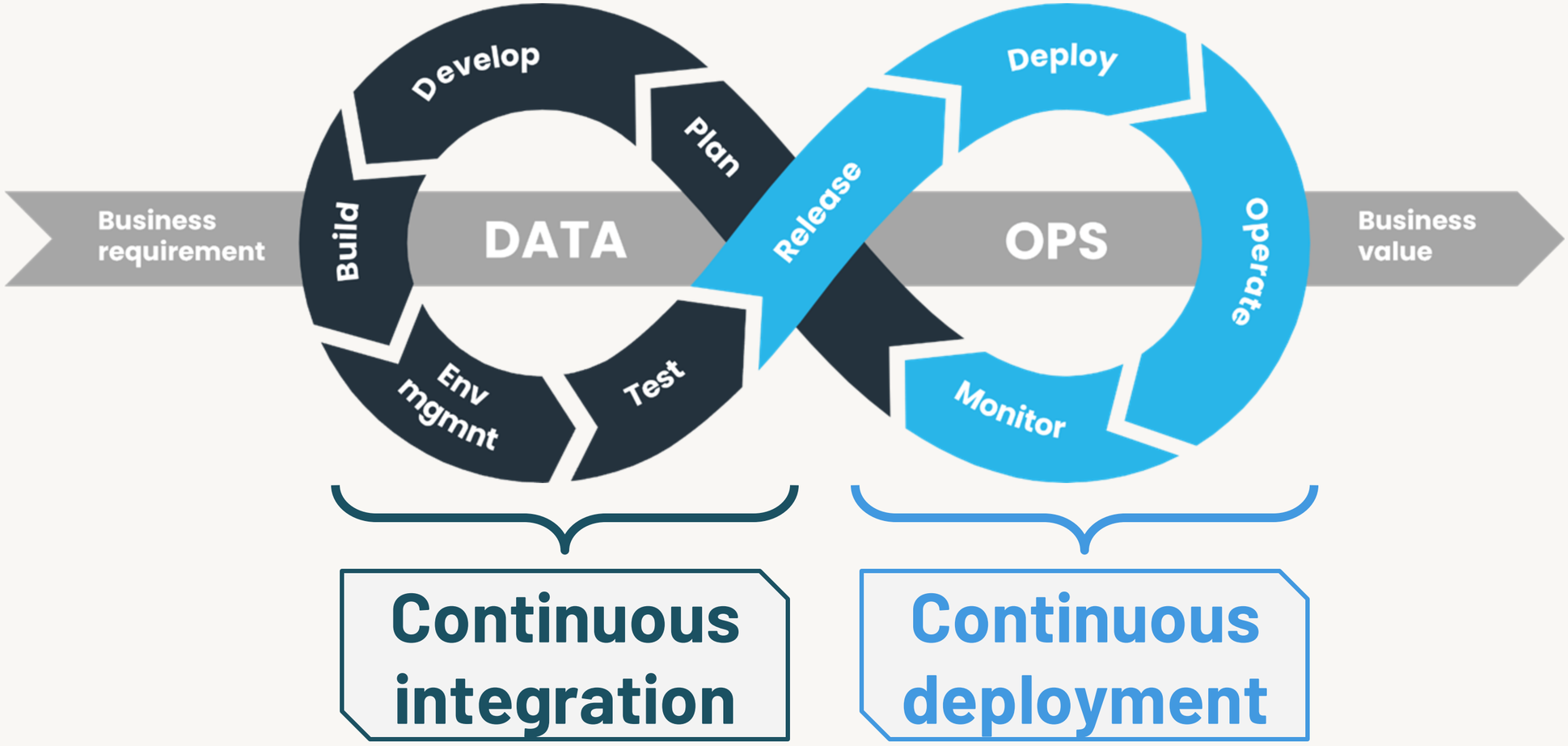
Software Engineering, DevOps, and CI/CD Fundamentals

LECTURE

The Role of CI/CD in DevOps

The Role of CI/CD in DevOps

Continuous Integration (CI) and Continuous Deployment (CD)



The Role of CI/CD in DevOps

Overview

CI/CD Overview

- **Automates** and **streamlines** software development processes.
- **Improves** code quality, speed, and reliability.
- Code is deployed to production through an **automated process**.

CI/CD Process

- Enables **development** and **delivery** of software in short, frequent cycles.
- Uses **automated** pipelines to ensure faster **deployment** and consistency.

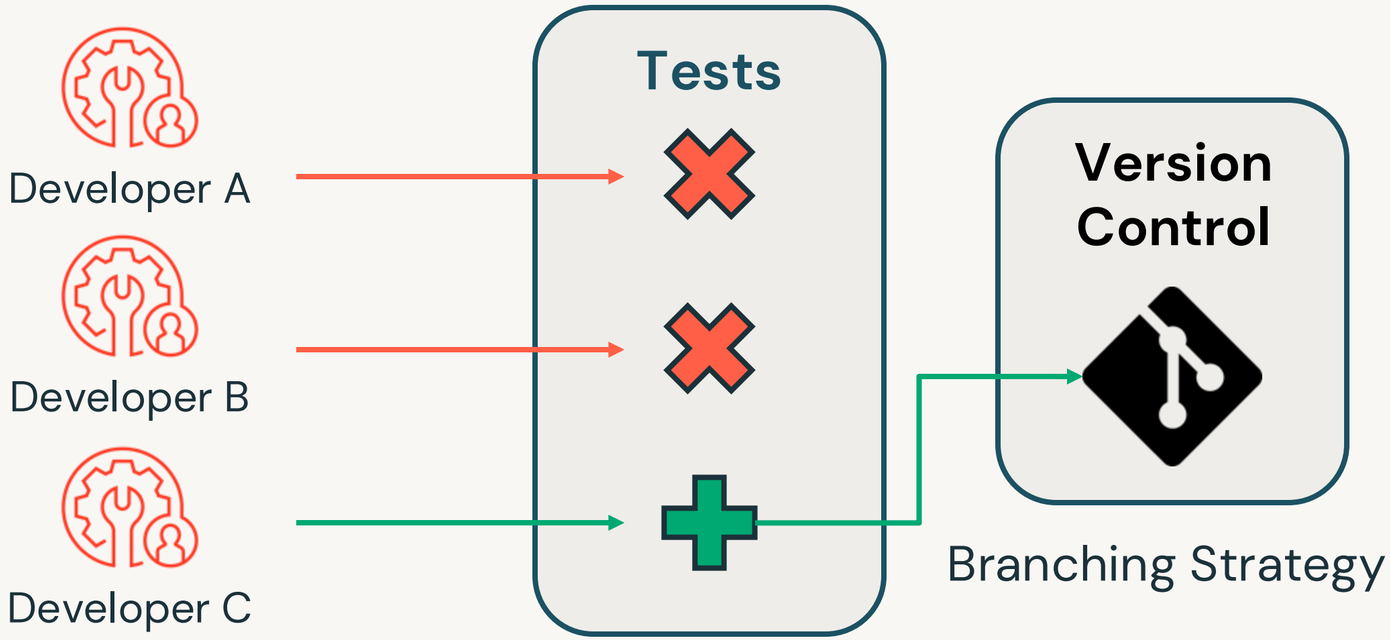
CI/CD Adoption

- Common practice in software development.
- **Growing importance** in data engineering and data science.

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.

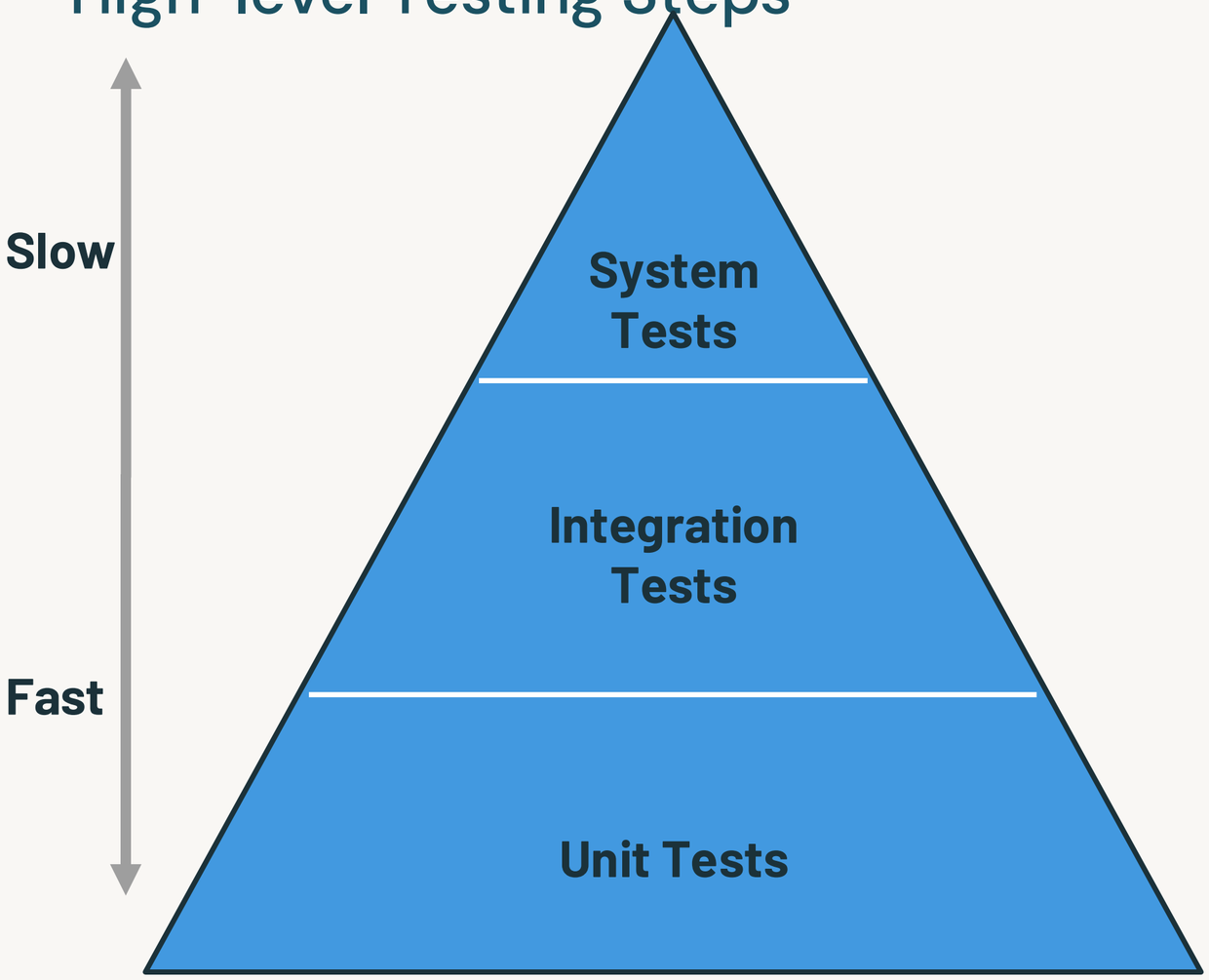


Benefits

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

The Role of CI/CD in DevOps

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

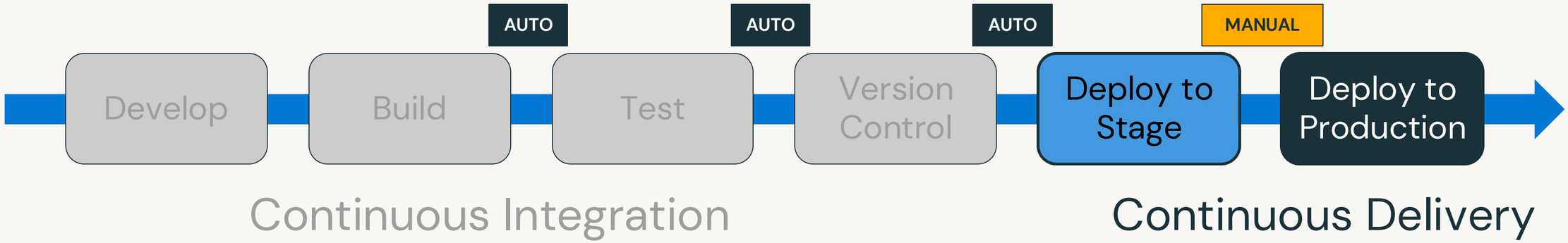
- Test **individual** functions or methods in isolation
- Fast, low cost, high coverage, and automated
- **Ex:** Custom pyspark functions

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.



The Role of CI/CD in DevOps

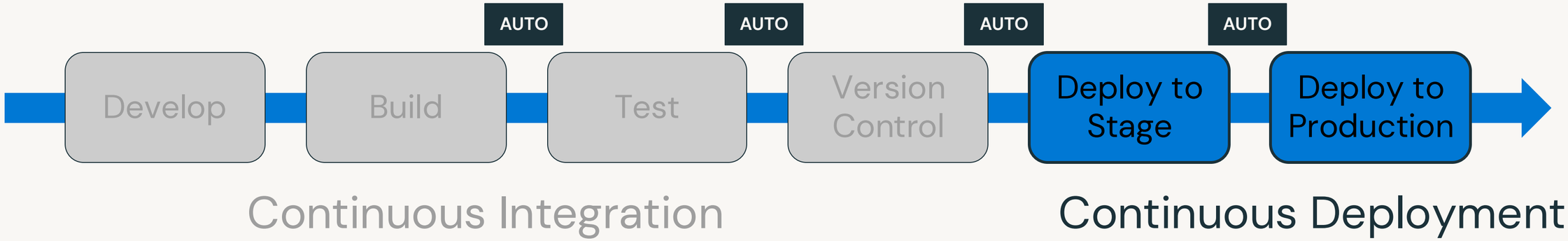
Continuous Delivery/Deployment (CD) Overview

Continuous Delivery (CD):

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

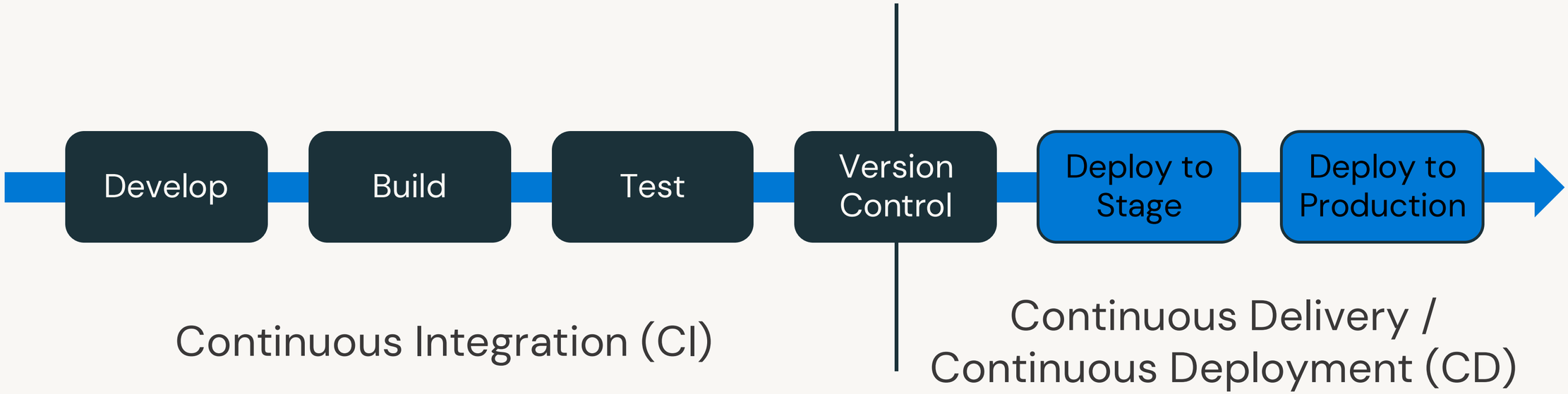
Continuous Deployment (CD):

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



The Role of CI/CD in DevOps

High-Level CI/CD Workflow Overview



Knowledge Check

Knowledge check

Think about this question and volunteer an answer

Which of the following best describes the primary goal of DevOps?

A. To automate the software development lifecycle and streamline collaboration between development and operations teams.

B. To ensure that the development team focuses solely on writing code without worrying about deployment or infrastructure.

C. To prioritize long release cycles in order to maintain stability.

D. To eliminate the need for quality assurance teams by automating all testing.

Knowledge check

ANSWER

Which of the following best describes the primary goal of DevOps?

A. **To automate the software development lifecycle and streamline collaboration between development and operations teams.**

B. To ensure that the development team focuses solely on writing code without worrying about deployment or infrastructure.

C. To prioritize long release cycles in order to maintain stability.

D. To eliminate the need for quality assurance teams by automating all testing.

Knowledge check

Think about this question and volunteer an answer

What is the primary purpose of continuous integration (CI) in a DevOps pipeline?

A. To automate the deployment of code to production

B. To ensure that code changes are tested and integrated frequently to detect errors early

C. To monitor the system performance in real-time

D. To manage infrastructure provisioning through code

Knowledge check

ANSWER

What is the primary purpose of continuous integration (CI) in a DevOps pipeline?

A. To automate the deployment of code to production

B. **To ensure that code changes are tested and integrated frequently to detect errors early**

C. To monitor the system performance in real-time

D. To manage infrastructure provisioning through code

Knowledge check

Think about this question and volunteer an answer

Which of the following is a key characteristic of Continuous Deployment (CD) in a DevOps pipeline

A. Code changes are automatically deployed to a staging environment after passing tests
B. Code is deployed to production without manual approval once automated tests pass
C. Developers must manually push changes to production after review
D. Code is deployed to multiple environments with no automated testing

Knowledge check

ANSWER

Which of the following is a key characteristic of Continuous Deployment (CD) in a DevOps pipeline

A. Code changes are automatically deployed to a staging environment after passing tests

B. **Code is deployed to production without manual approval once automated tests pass**

C. Developers must manually push changes to production after review

D. Code is deployed to multiple environments with no automated testing

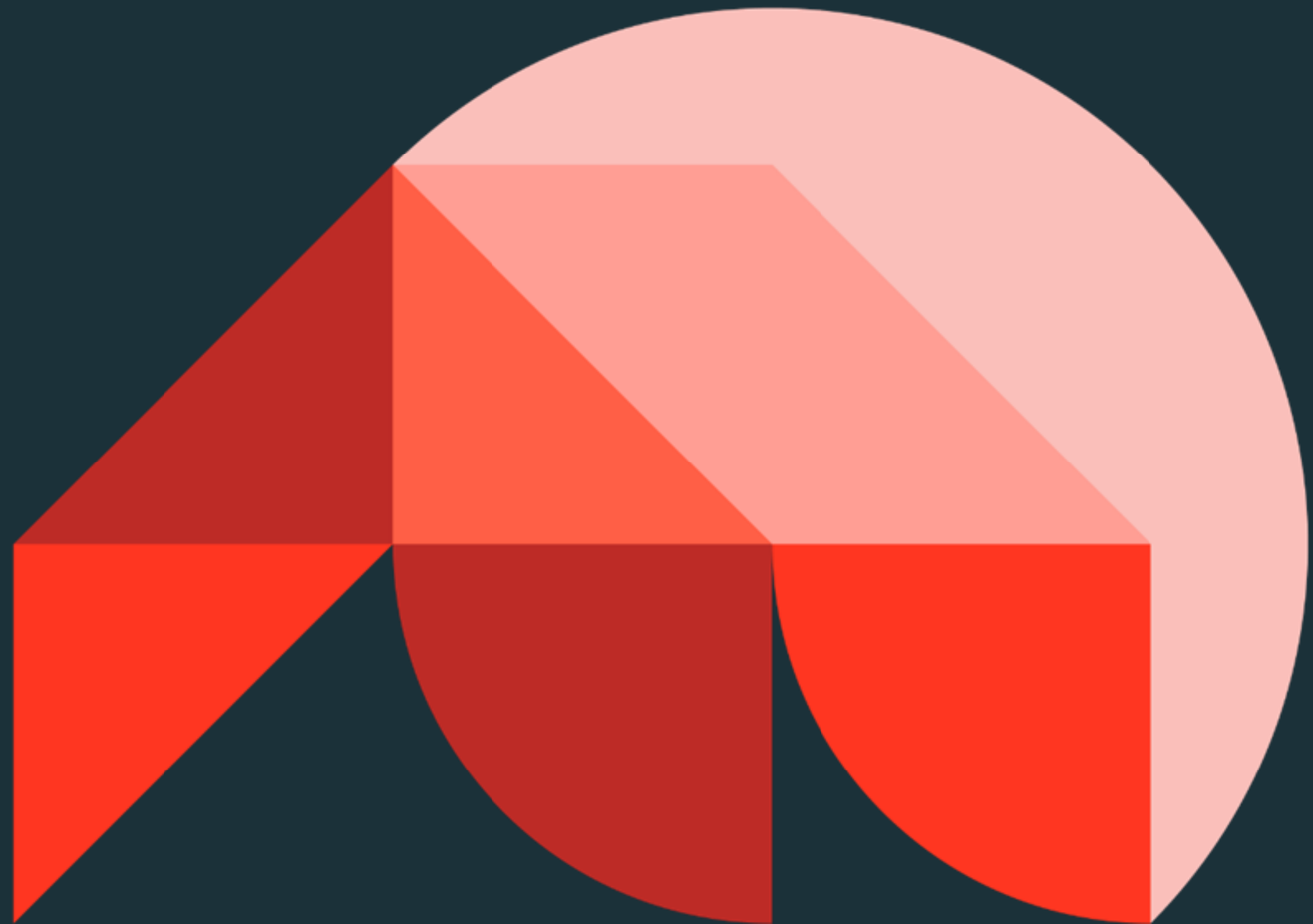
Discussion Question

Think about this question and volunteer an answer

Do you currently use any DevOps practices in your organization? If so, which ones have been most effective, and what challenges have you faced?

databricks

Continuous Integration (CI)



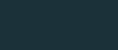
DevOps Essentials for Data Engineering

Section Learning Objectives

- Learn to plan and structure a data engineering project, defining key components and isolation steps for successful execution.
- Learn to write unit tests for PySpark code to ensure functionality and catch errors early.
- Apply the pytest framework in Databricks to execute unit tests and analyze the test results for errors.
- Learn to run integration tests using DLT and workflows to validate data pipeline functionality.

Agenda

Continuous Integration
• Planning the Project
• Introduction to Unit Tests with Pyspark
• Executing Integration Tests with DLT and Workflows
• Version Control with Git Overview



databricks

Continuous Integration (CI)

LECTURE

Planning the Project

Planning the Project

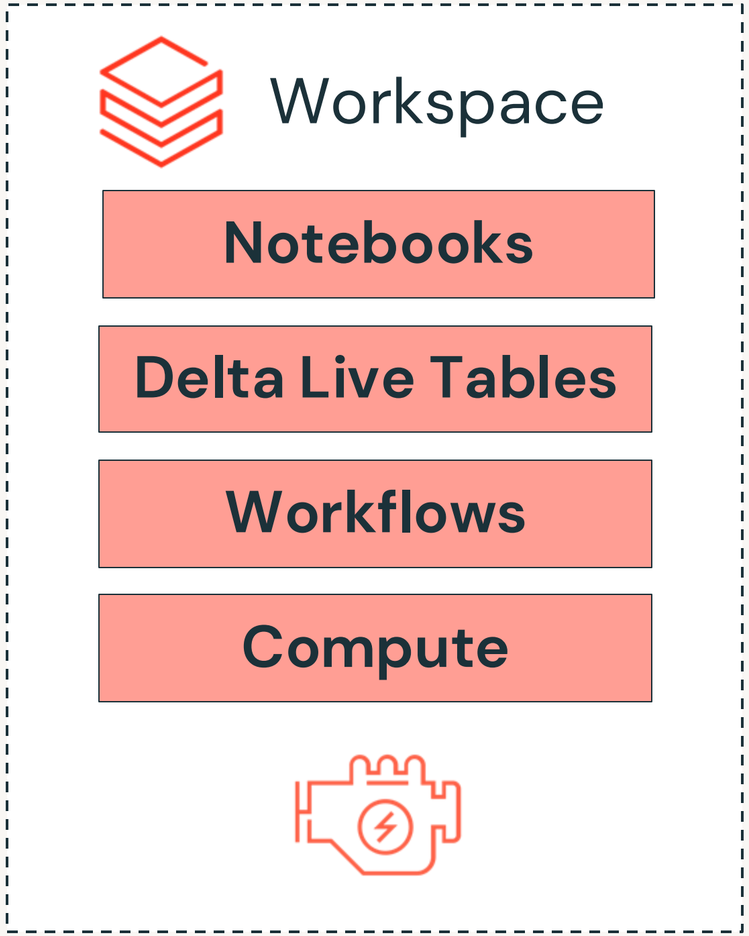
Requirements

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

Setting Up Your Data Environments

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

Planning the Project

Isolating Environments

Workspaces

Utilizing multiple workspaces, one for each environment

DEV

STAGE

PROD

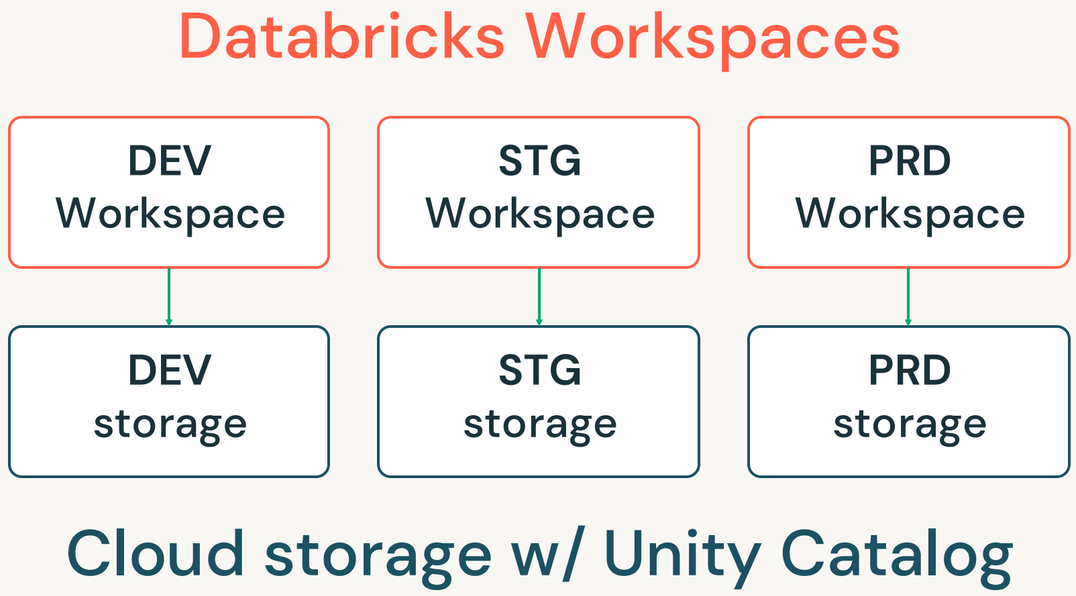
Catalogs

Utilizing multiple catalogs, one for each environment

Planning the Project

Workspace Isolation Overview

You can **isolate** your dev, stage and prod environments at the Workspace and storage level.



Planning the Project

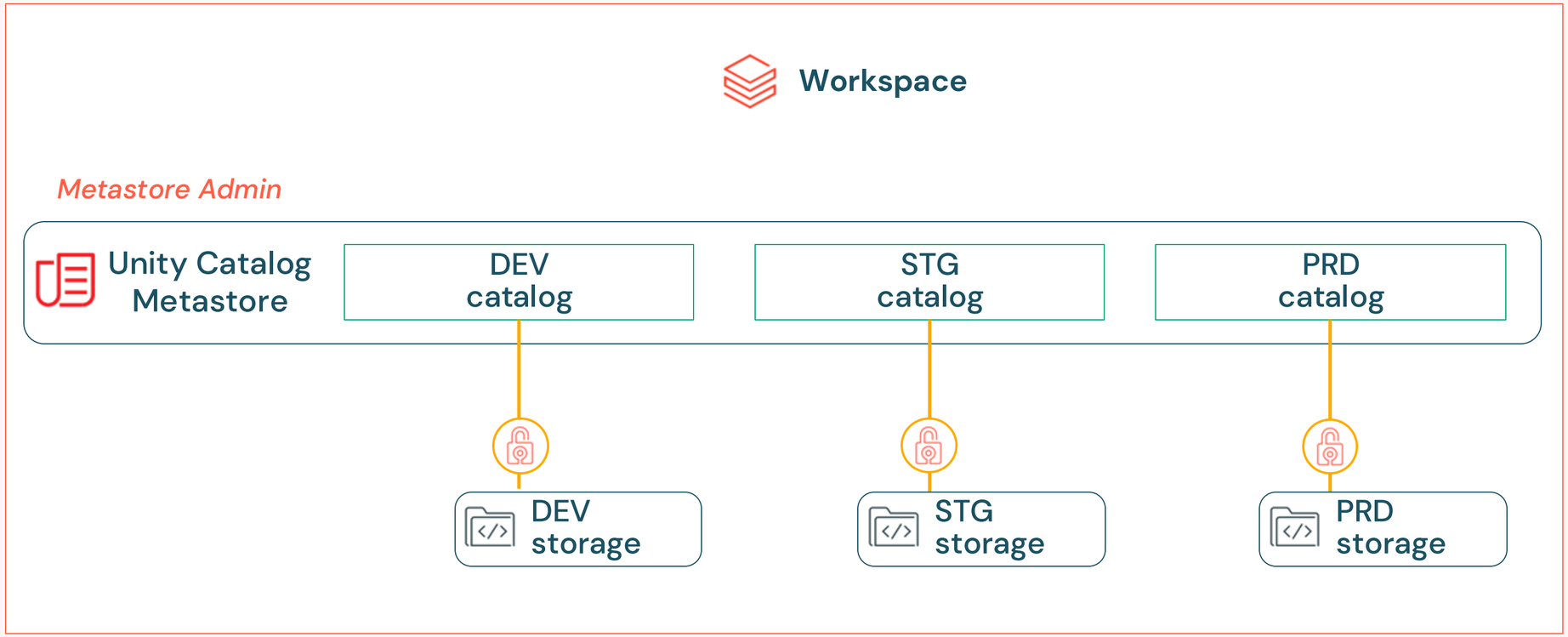
Unity Catalog Isolation

Storage isolation

This example separates the storage locations on catalog level

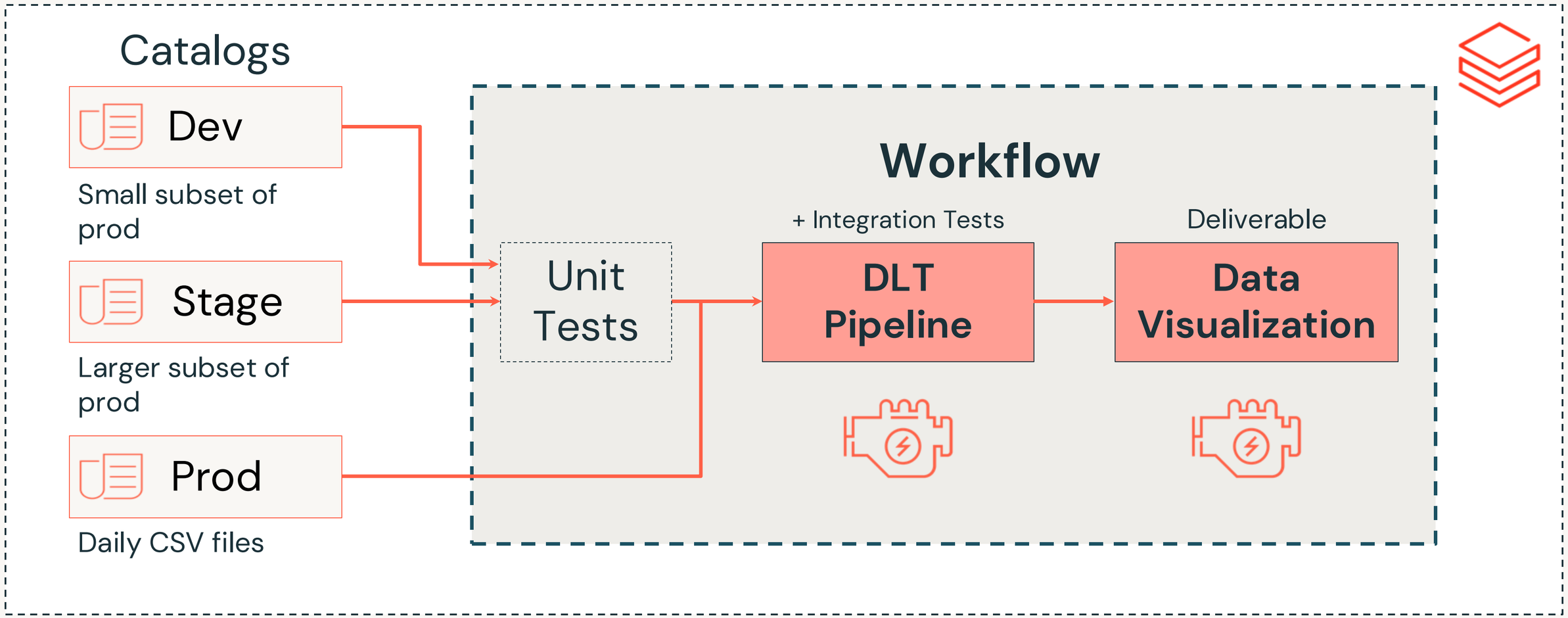
UC Access Control

Users should only gain access to data/ metadata based on agreed access rules



Planning the Project

Course Project Architecture

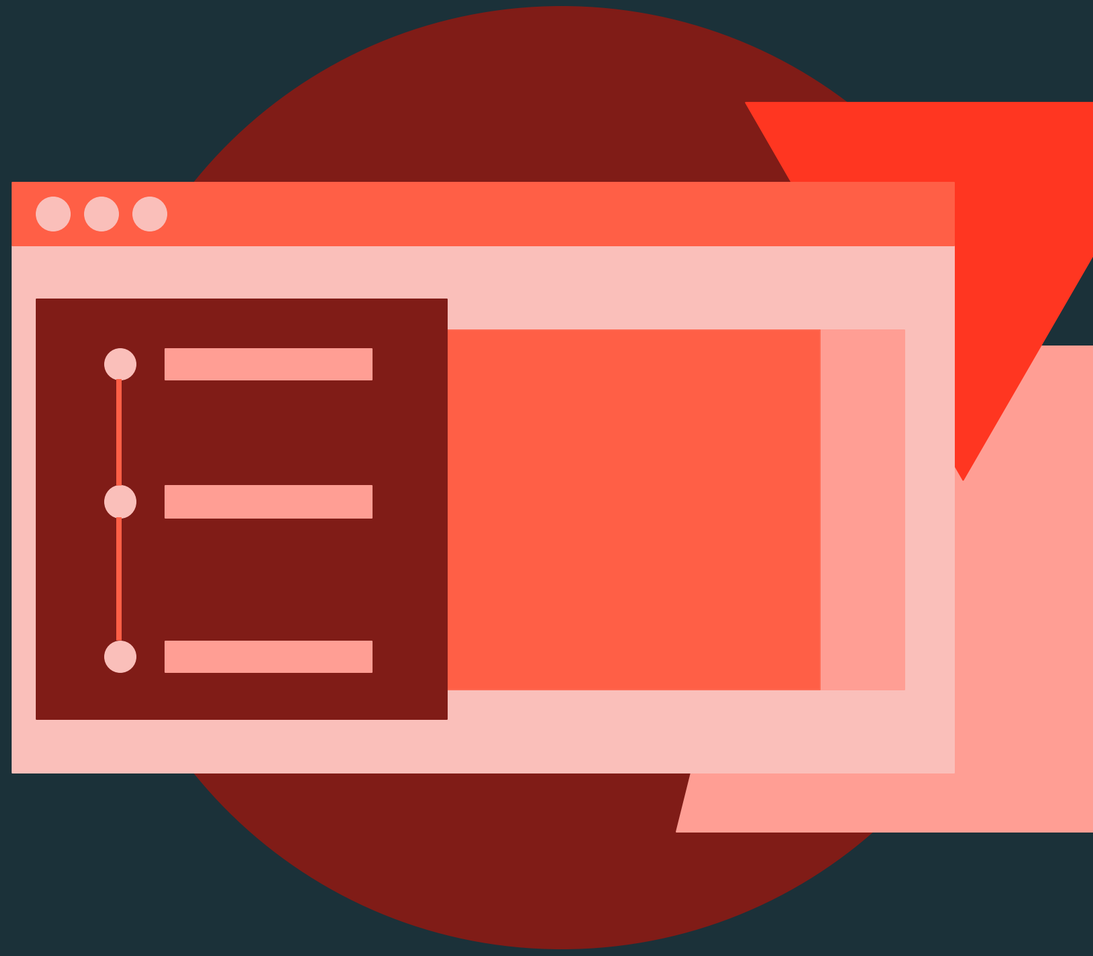


databricks

Continuous Integration (CI)

DEMONSTRATION

Project Setup Exploration



Notebook: Course Notebooks/M02 - CI/2.3 - Project Setup Exploration

databricks

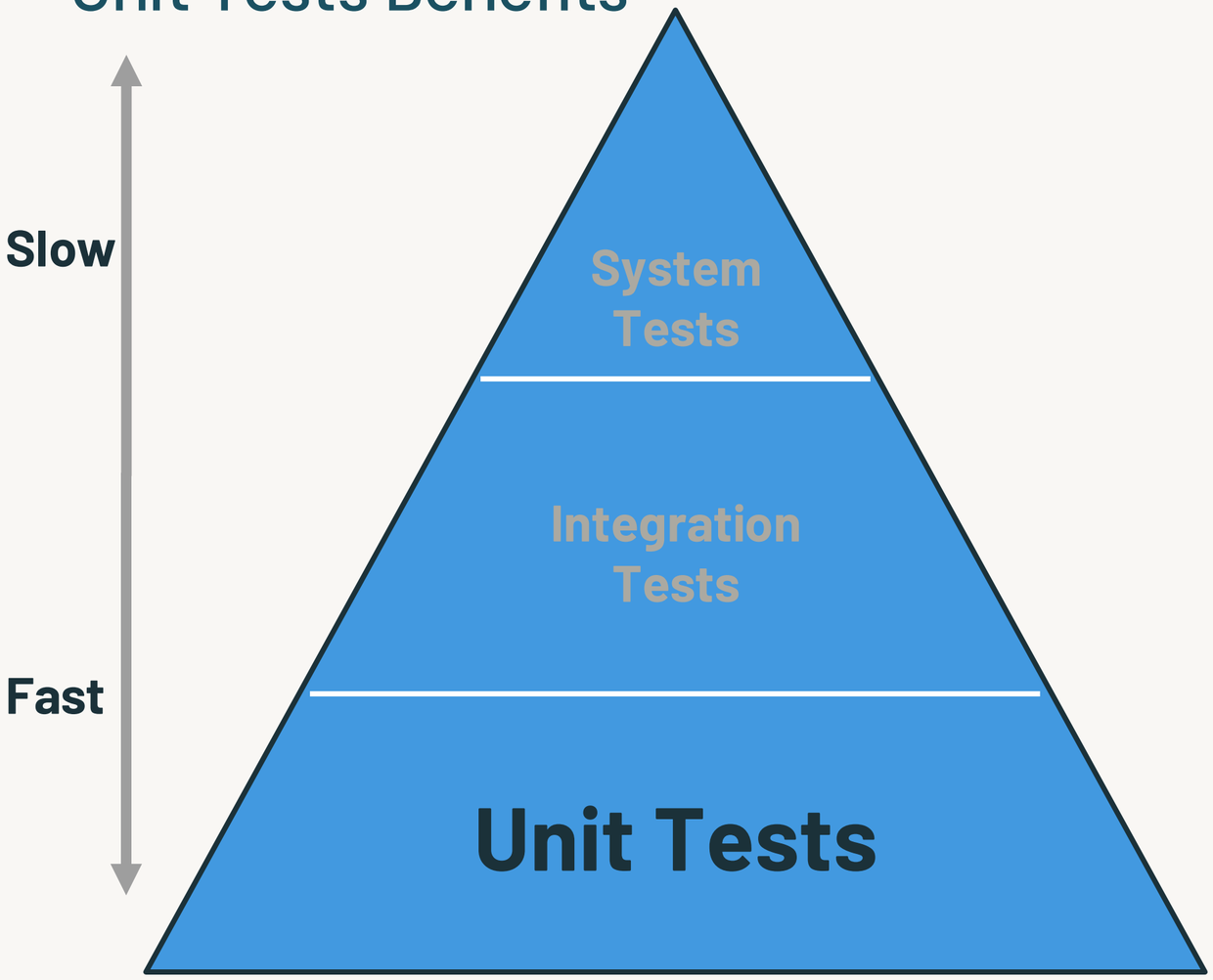
Continuous Integration (CI)

LECTURE

Introduction to Unit Tests for PySpark

Introduction to Unit Tests for PySpark

Unit Tests Benefits



UNIT TESTS Benefits

- Test only **one specific** function, on small amount of data
- Catch bugs before you deploy them in your project
- Refactoring your code is easier
- Tests make your debugging easier

Introduction to Unit Tests for PySpark

Pyspark.testing.utils Testing Functions

- **pyspark.testing.utils** provides helper functions to make unit testing in PySpark easier.
  - **assertDataFrameEqual(actual, expected[, ...])**
  - **assertSchemaEqual(actual, expected)**

There are a variety of other methods to test your unit tests, we will focus on the pyspark testing utils.

Unit Test Example

1. You have the following function to create a column

```
from pyspark.sql.functions import col, when
def add_new_col(df, new, s_col):
 return (df
         .withColumn(new,
            when(col(s_col) == 0, 'Normal')
            .otherwise('Unknown')))
```

2. Your desired results

original	desired
0	Normal
1	Unknown
-1	Unknown
null	Unknown

Unit Test Goal

Compare the **actual result** of the function with a **defined expected result**

3. Create the unit test

```
def test_add_new_col():
    data = [(0,), (1,), (-1,),(None,)]
    columns = ["value"]
    df = spark.createDataFrame(data, columns)

    actual_df = add_new_col(df, "new_value", "value")

    expected_data = [(0, 'Normal'), (1, 'Unknown'),
                     (-1, 'Unknown'), (None, 'Unknown')]
    expected_df = spark.createDataFrame(expected_data,
                                        ["value", "new_value"])

    assertDataFrameEqual(actual_df, expected_df)
```

Create a unit test function to test the function. Name the unit test function accordingly.

Create a sample DataFrame to test any use cases you can think of.

Execute your function on the sample data and store the result.

Create an expected result DataFrame using the sample data.

Check the two DataFrames. If they are not identical an error will be returned.

Unit Testing Framework – 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

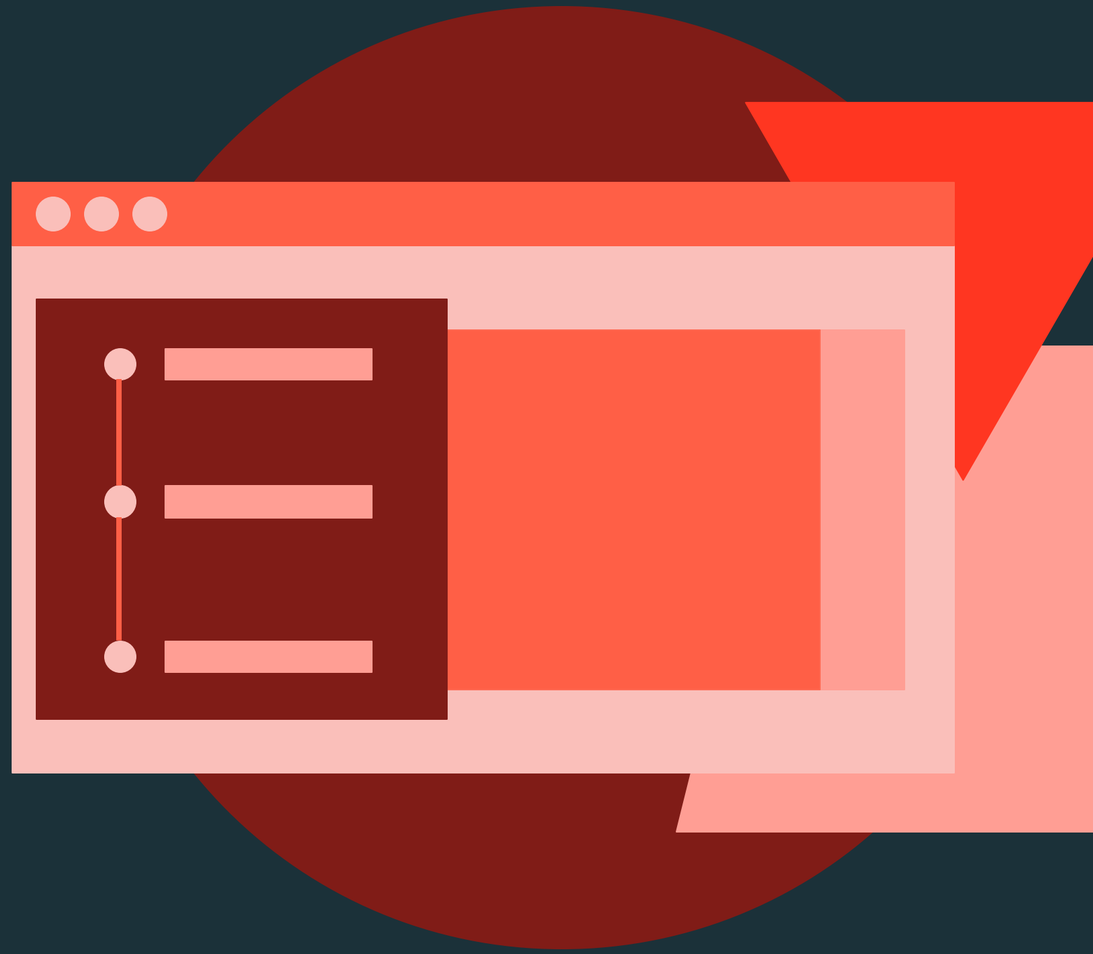
This course provides a simple introduction to **pytest**. There are many testing frameworks available, select the one that best meets your organization's needs.

databricks

Continuous Integration (CI)

DEMONSTRATION

Creating and Executing Unit Tests



Notebook: Course Notebooks/M02 - CI/2.4 - Creating and Executing Unit Tests

databricks

Continuous Integration (CI)

LAB EXERCISE

Create and Execute Unit Tests

Notebook: Course Notebooks/M02 - CI/2.5L - Create and Execute Unit Tests

databricks

Continuous Integration (CI)

LECTURE

Executing Integration Tests with DLT and Workflows

Executing Integration Tests

With Delta Live Tables (DLT) or Workflows

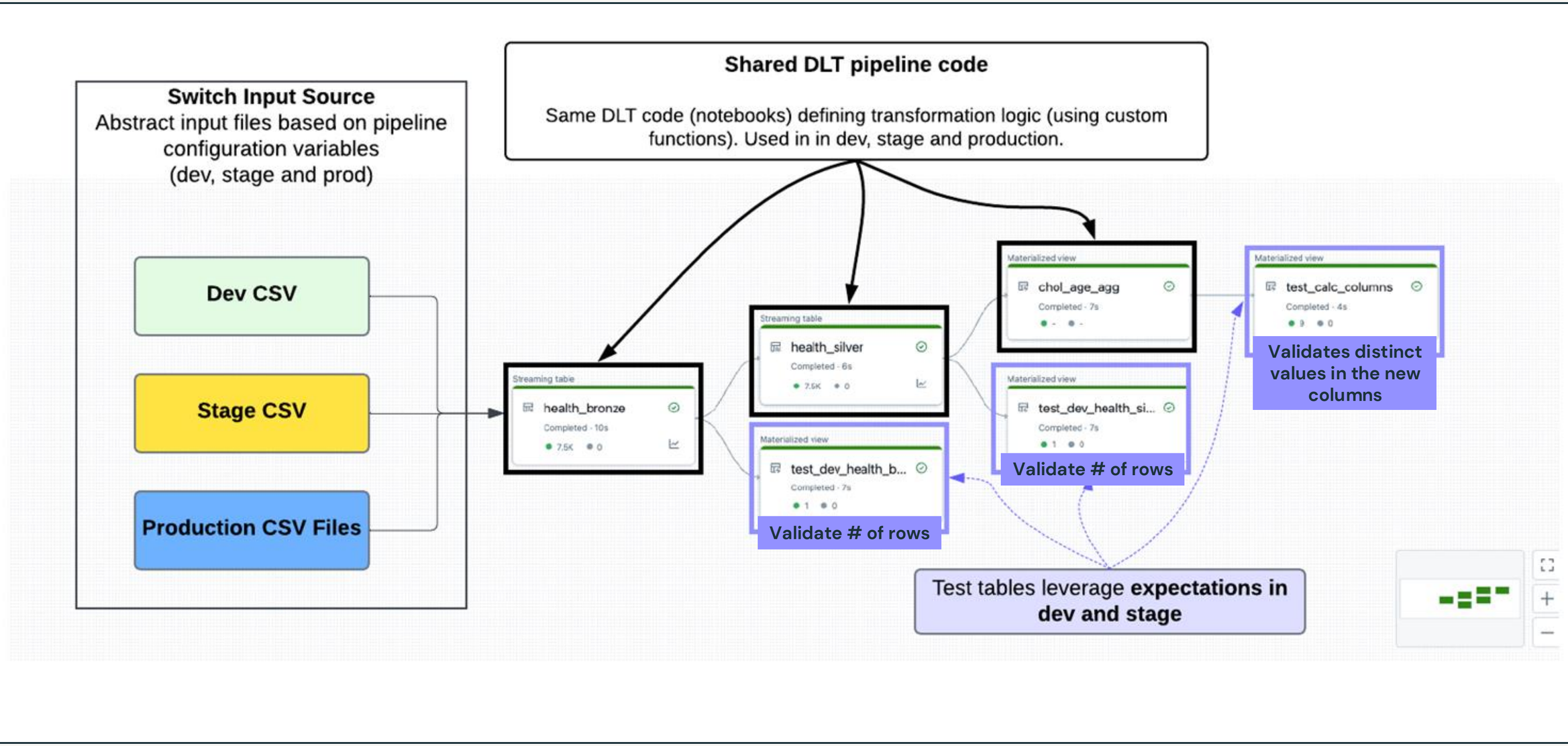
Delta Live Tables (DLT)

- Use **DLT expectations** to check pipeline's results (demo technique).

Workflows

- Implement it as a **Databricks Workflow with multiple tasks** - similarly what is typically done for non-DLT code.

Delta Live Tables – Method 1 – Expectations

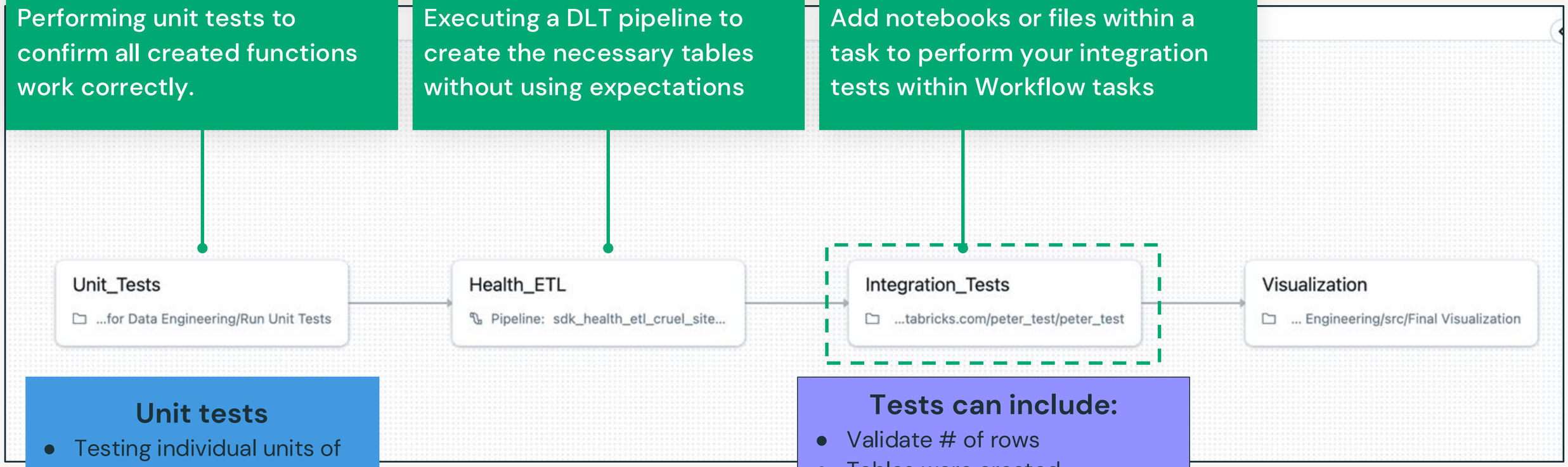


Databricks Workflow – Method 2 – Tasks

Performing unit tests to confirm all created functions work correctly.

Executing a DLT pipeline to create the necessary tables without using expectations

Add notebooks or files within a task to perform your integration tests within Workflow tasks



Unit tests

- Testing individual units of code in isolation
- If the unit test pass continue on into the Workflow

Tests can include:

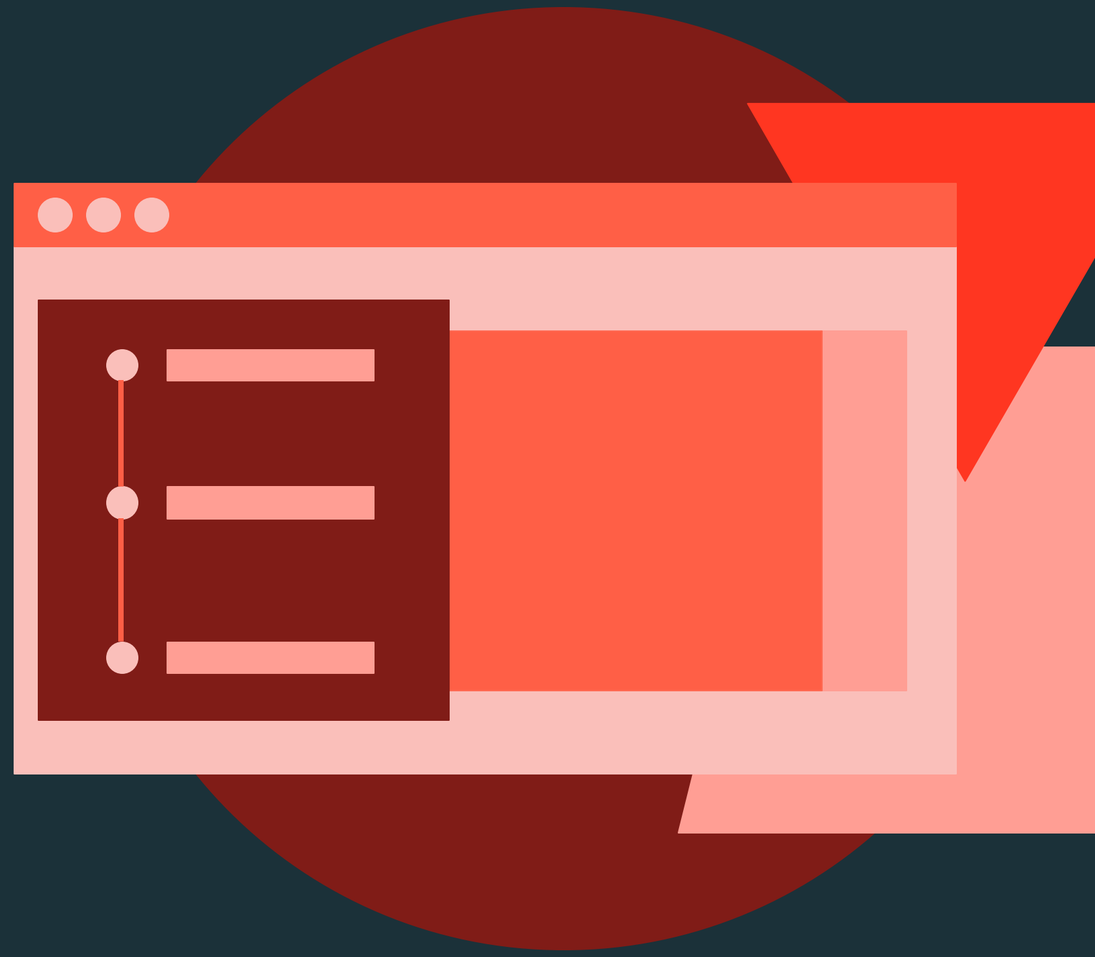
- Validate # of rows
- Tables were created successfully
- Columns contain correct distinct values
- Columns contain values between a specific range
- Columns contain no duplicates

databricks

Continuous Integration (CI)

DEMONSTRATION

Performing Integration Tests with DLT and Workflows



Notebook: Course Notebooks/M02 - CI/2.6- Performing Integration Tests

Section Learning Objectives

- Organizational challenges with version control
- Have a high-level understanding of how Git-based repositories work on Databricks
- Explain source code challenges with Branching
- Define git Operations, common tools, and understand how Databricks supports Git providers
- Understand how to integrate GitHub repositories with Databricks
- Explain how Git Folders can be used for development
- Understand which file types are supported with repos

databricks

Continuous Integration (CI)

LECTURE

Version Control with Git Overview

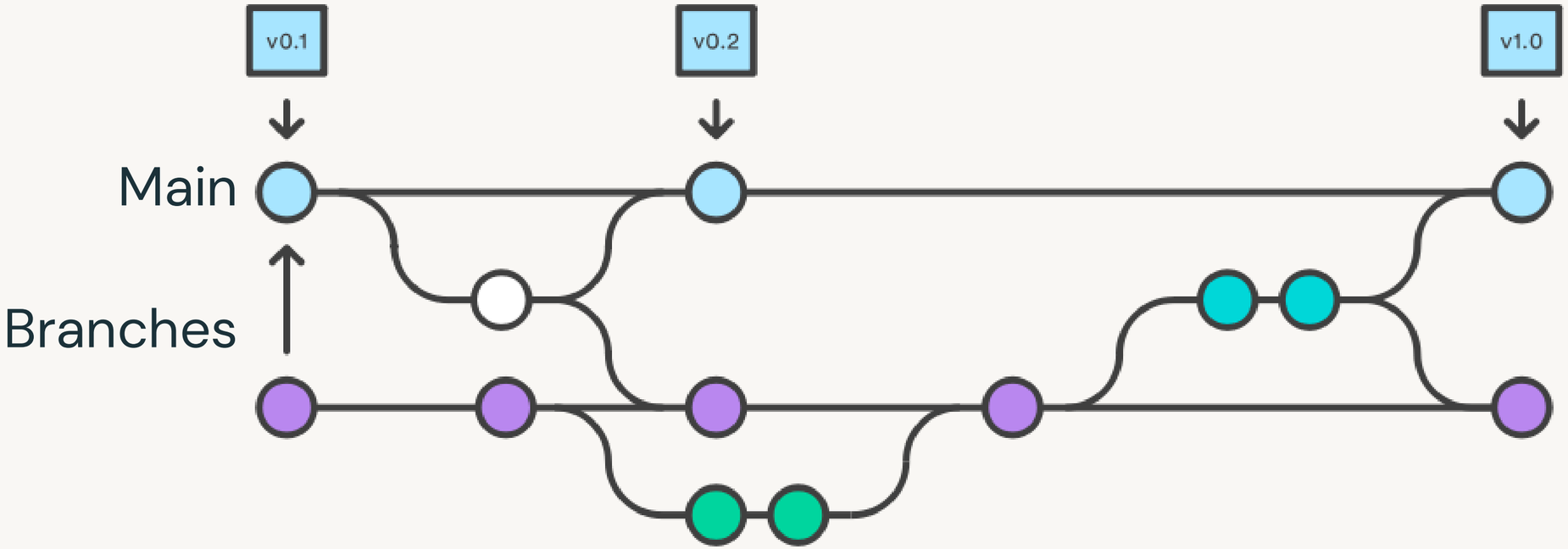
Complications with Version Control

Organizational Challenges

- Silos form over time that isolate development and team operations.
- Independent development leads to lower quality of software development due to lack of centralized version control (dpublicate code, inconsistent standards, code reviewing, etc.)
- Unstable versioning - it can become difficult to track, revert, and audit changes .
- Without proper version control, managing frequent updates increases risk.
- Branching, merging, and CI/CD integration can become difficult, which diminishes the ability to scale development.

Secure code changes through branching

- Complementary concept to CI/CD → Enables effective CI
- Version control changes and run through quality control before merging to main branch and deploying
- Example: Gitflow



Source: Openclipart.org

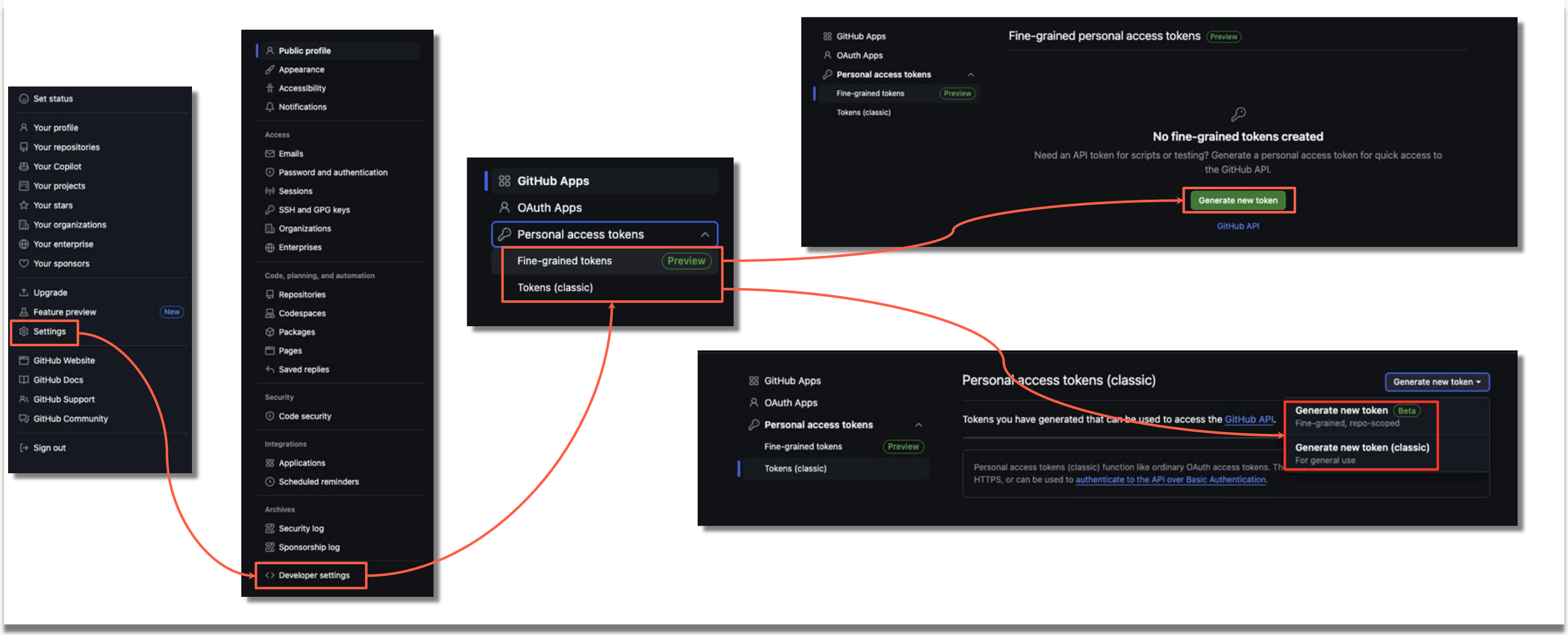
Overview of Git with Databricks

Definitions

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

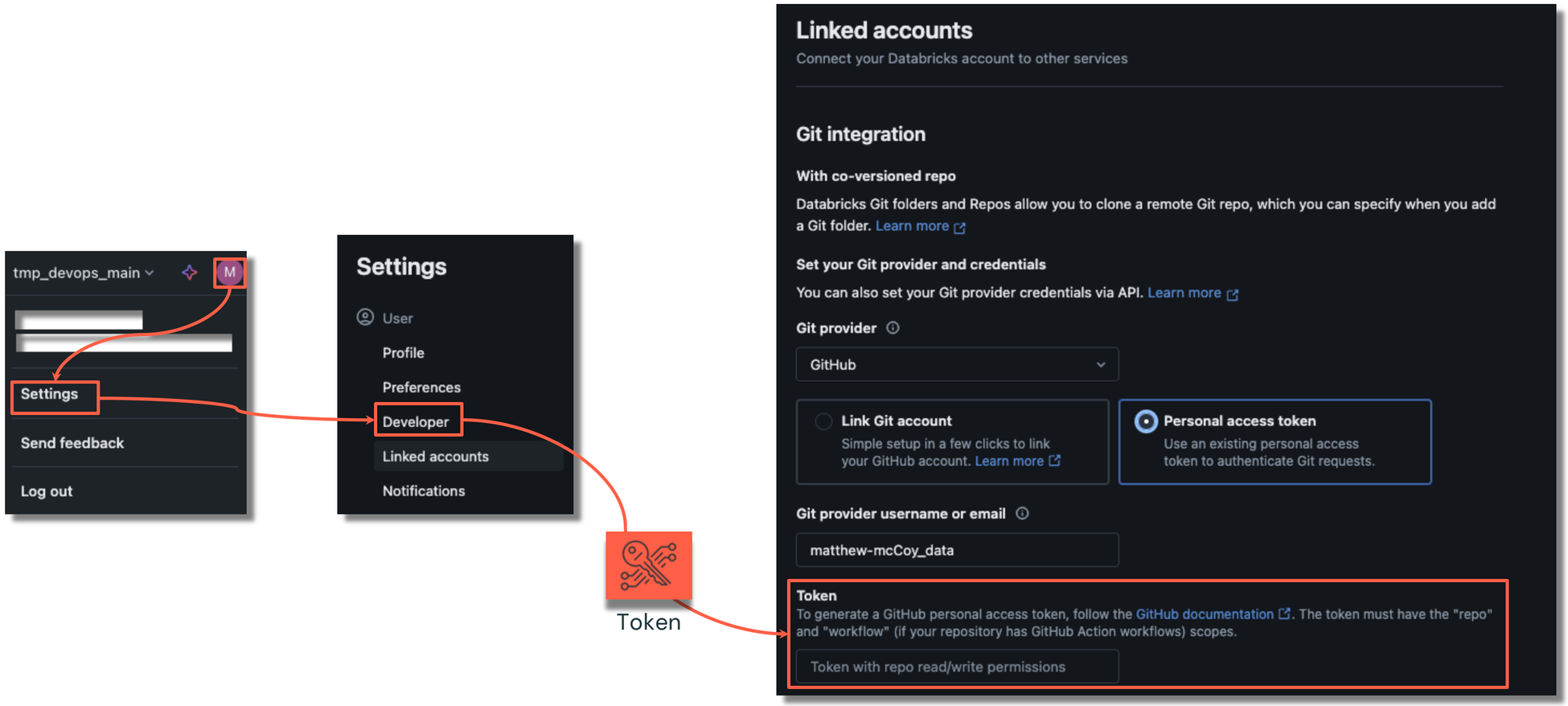
Generating GitHub Personal Access Token(PAT)

Authentication

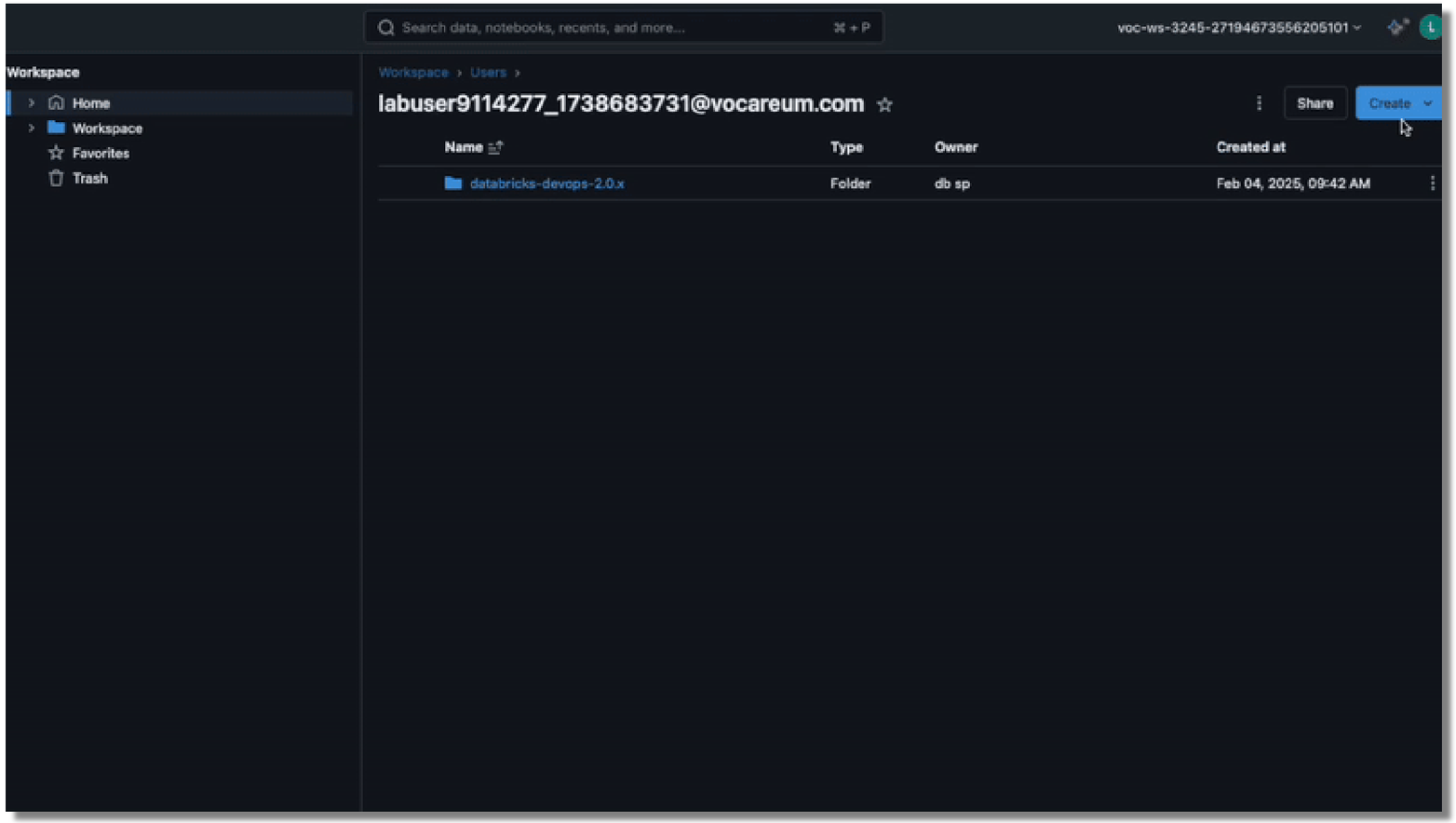


Connecting to Databricks with GitHub PAT

Seamless Integration



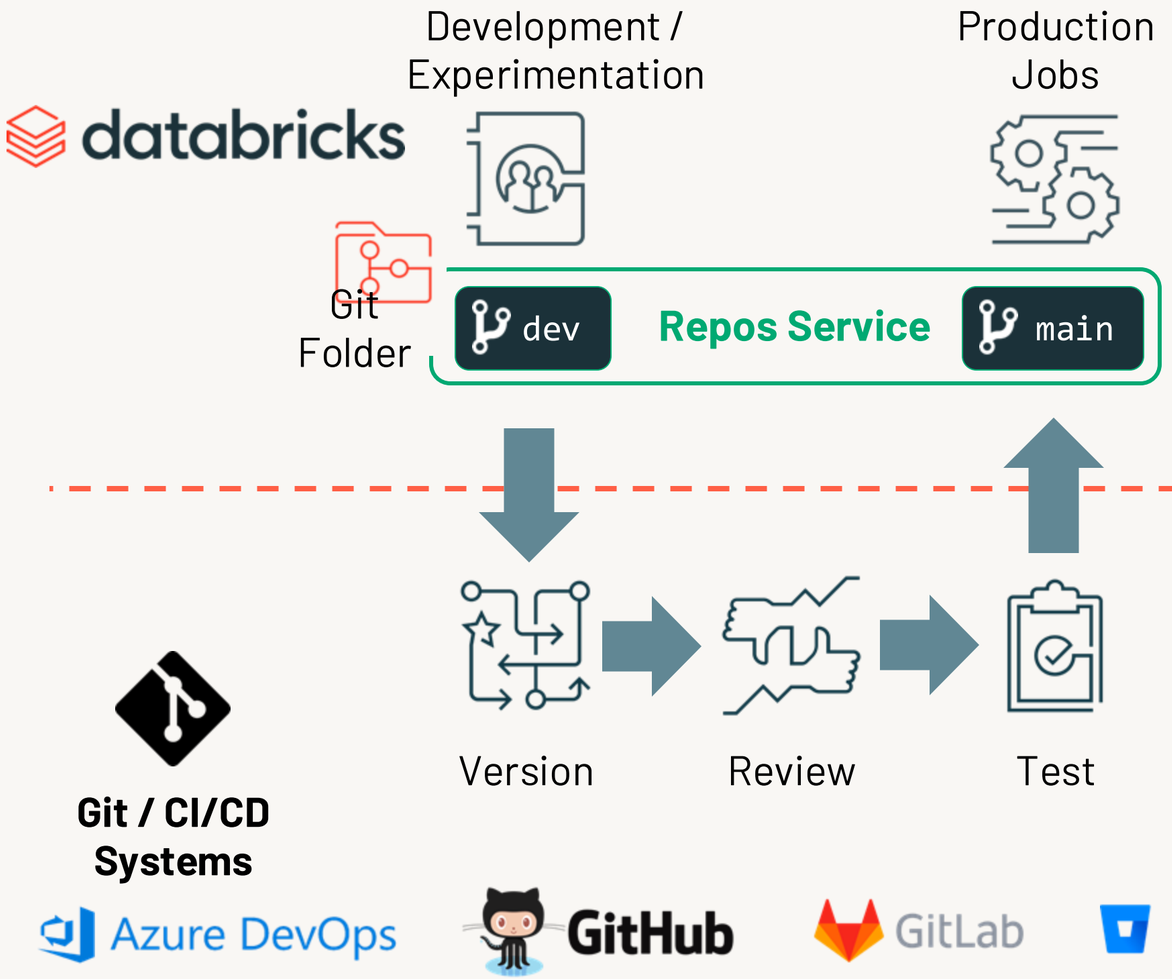
Databricks Git Folders



1. Clone remote repositories
2. Commit and push changes using UI
3. Pull branch updates
4. Branch Management
5. Visual validation when committing

Git-based Repos in Databricks

Repos API to automate CI/CD



Programmatically manage Repos

E.g., set up automation to automatically create Databricks Repos for all of your repositories

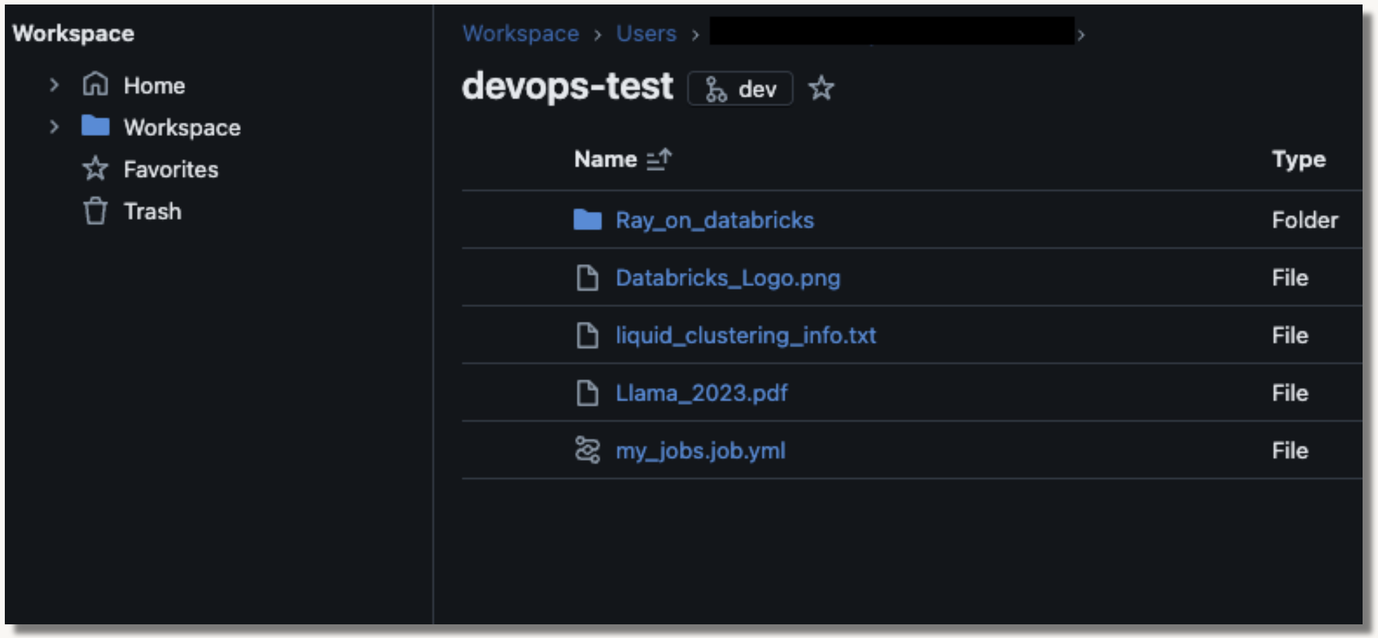
Perform Git operations on Repos

E.g., set up automation to always keep Repos in a production folder at a specified version

Specify Git branch, tag, hash for Jobs

E.g., update the Repo to a specific branch and commit and run Jobs against the notebooks in the repo

Arbitrary files support in Repos



.py

- Portability of code
- Library files - use Python/R files as packages

.txt
.yml

- Environment specification portability
- Build packages from the same repo

.csv

- Small data ease of use
- Relative imports

...

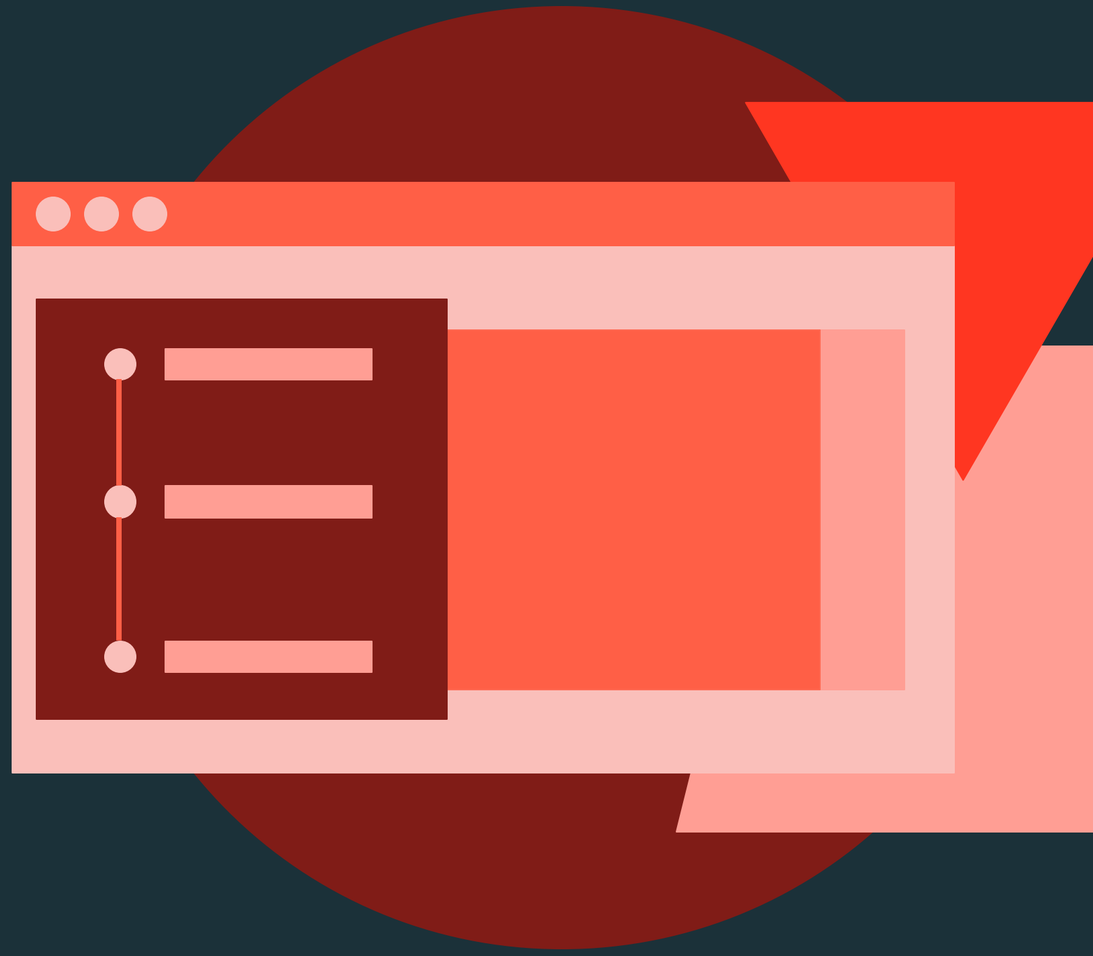
- Whatever you can do with files "just works"

databricks

Continuous Integration (CI)

DEMONSTRATION

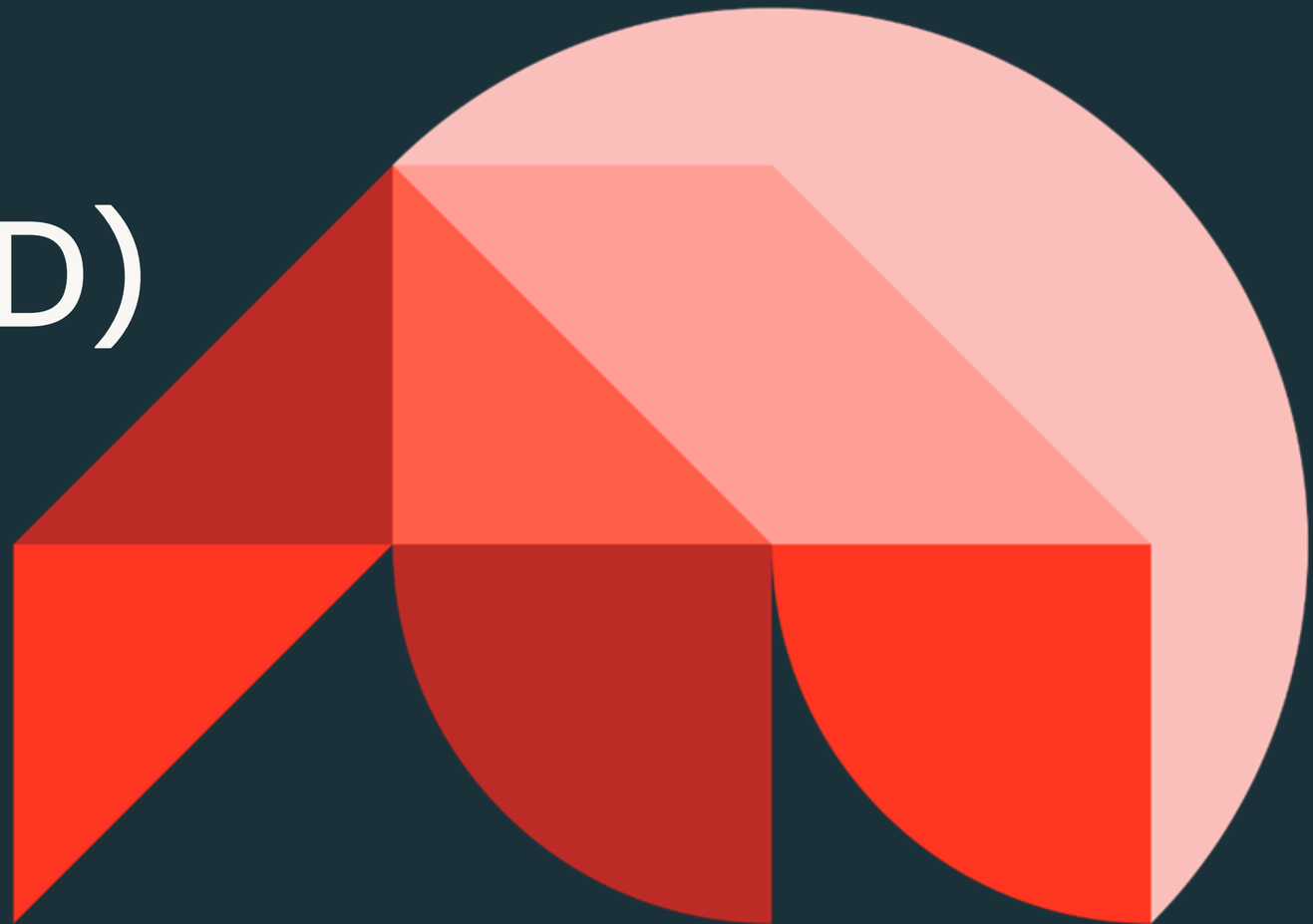
Version Control with Databricks Git Folders



Notebook: Course Notebooks/M02 - CI/2.7L - Version Control with Databricks Git Folders and GitHub

databricks

Introduction to Continuous Deployment (CD)



DevOps Essentials for Data Engineering

Section Learning Objectives

- Understand and demonstrate the difference between use-cases of the Databricks REST API, CLI, and SDK
- Understand how software engineering best practices are supported with DABs
- Understand how DABs are used for CI/CD

Agenda

Introduction to Continuous Deployment (CD)
• Deploying Databricks Assets Overview
• Deploying the Project
• Next Steps

databricks

Introduction to Continuous Deployment (CD)

LECTURE

Deploying Databricks Assets Overview

Deploying Databricks Assets Overview

Deployment Options

REST API

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

Databricks CLI

Command-line interface that wraps the REST API

Ideal for one-off tasks, experimentation, and shell scripting

Databricks SDKs

- Available for multiple programming languages (Python, Java, Go, R)
- Allows development of applications, custom Databricks workflows, and robust error-handling
- Programmatic way to interact with Databricks resources

Databricks Asset Bundles (DABs)

Databricks recommends Databricks Asset Bundles for creating, developing, deploying, and testing jobs and other Databricks resources

Databricks Asset Bundles

Version Control

The practice of tracking and managing changes to code and other development artifacts over time.

Code Review

Systematic examination of source code with the goal of identifying and squashing bugs, improve quality, and enforce coding standards.

Testing

The process of validating expected output from relevant functions and adhering to predetermine requirements.

Continuous Integration

The process of automating development, testing, and deployment to ensure reliability.

Software Engineering Practices

Databricks Asset Bundles (DABs)

What are DABs?

Write code once, deploy everywhere

YAML files that specify the **artifacts**, **resources**, and **configurations** of a Databricks project. This leads to easy configuration of complex notebook and pipeline interactions and **reproducibility** of your workflows.

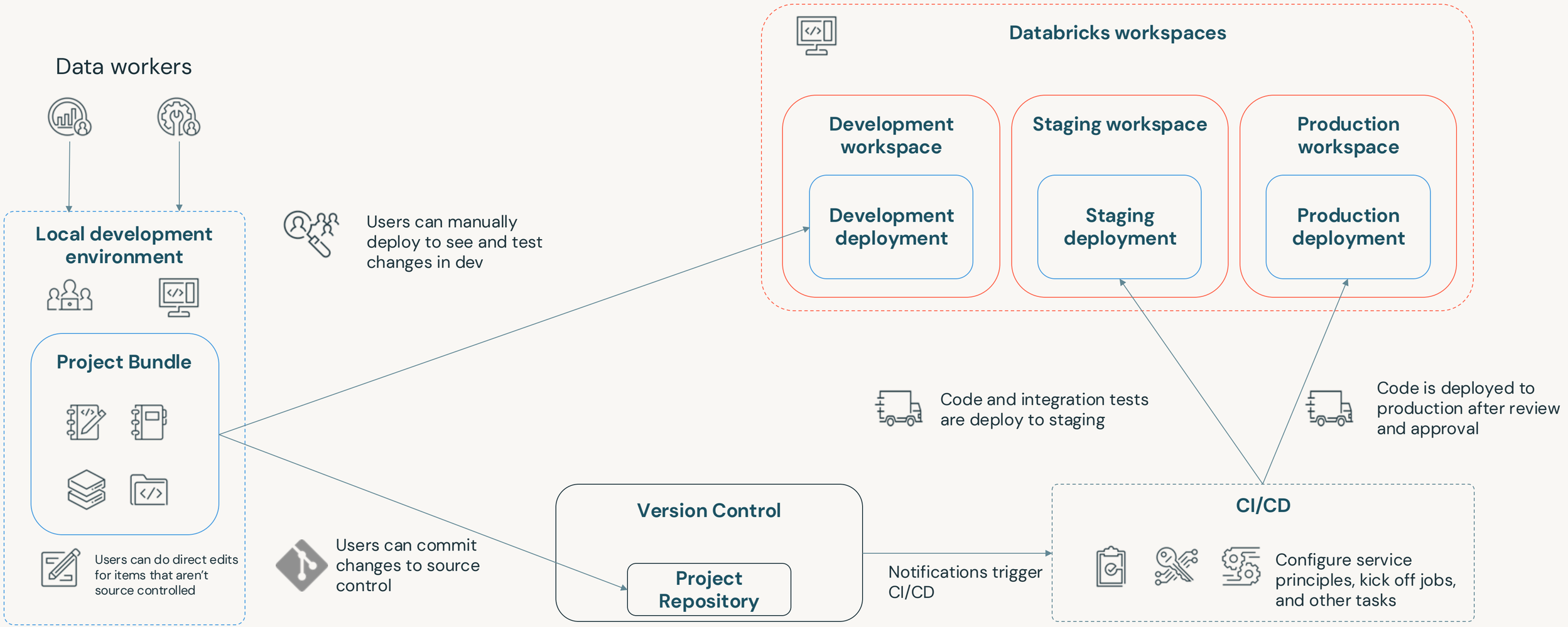
What are Databricks Asset Bundles?

DABs are a tool designed to streamline this process for **Databricks projects**. These bundles encapsulate all necessary configurations and artifacts.

How do bundles work?

Bundles provide an exact **definition** of Databricks resources that are to be used within your project with support for **validation** and **deployment** instructions.

Development and CI/CD with DABs

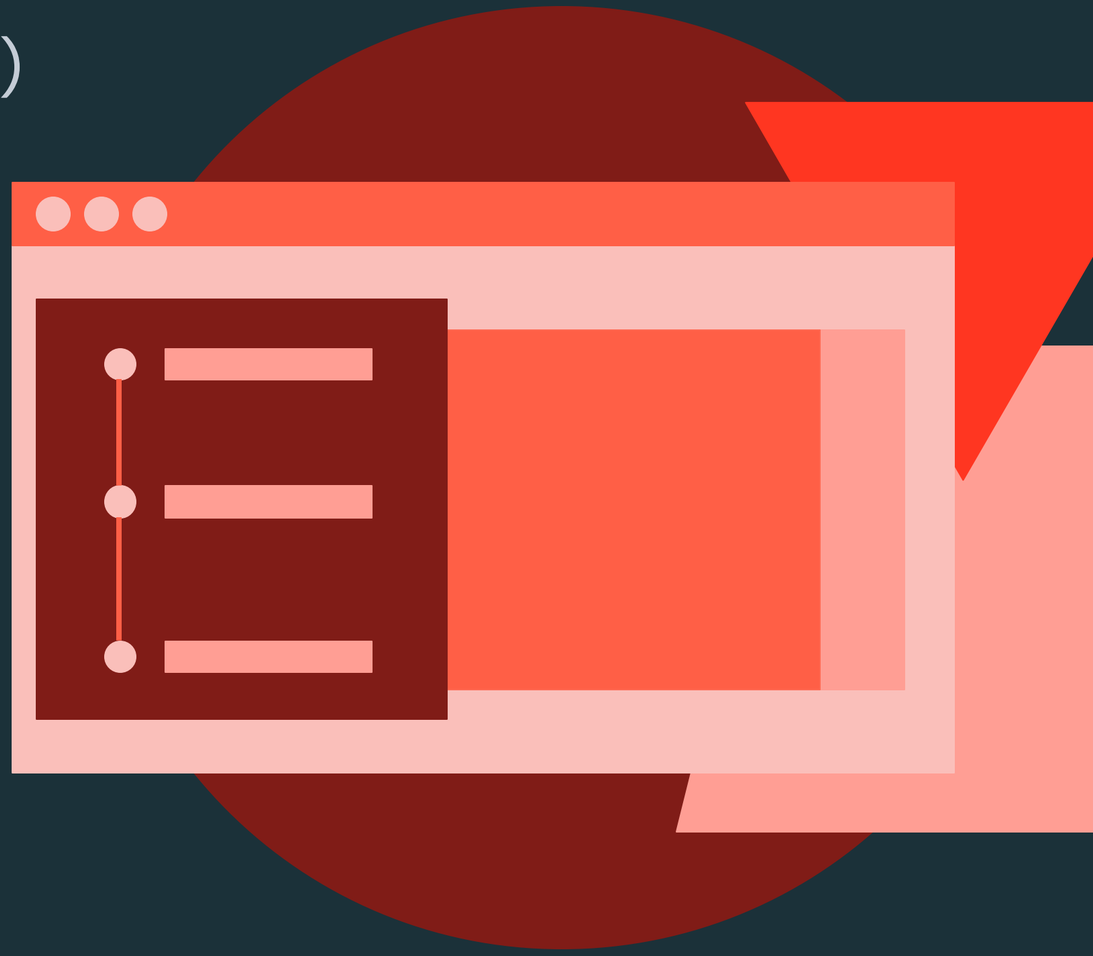


databricks

Introduction to Continuous Deployment (CD)

DEMONSTRATION

Deploying the Project



Notebook: Course Notebooks/M03 - CD/3.1 - Deploying the Databricks Assets

databricks

Introduction to Continuous Deployment (CD)

LECTURE

Next Steps

Next Steps

Additional resources for continuing the learning journey.

More Ops Course Offerings from Databricks

- (ILT/SP) Machine Learning Operations with Databricks
  - Associate
  - Professional

Further Documentation on Databricks Asset Bundles

- DABs documentation
- Bundles with Pipelines Tutorial
- MLOps Staks