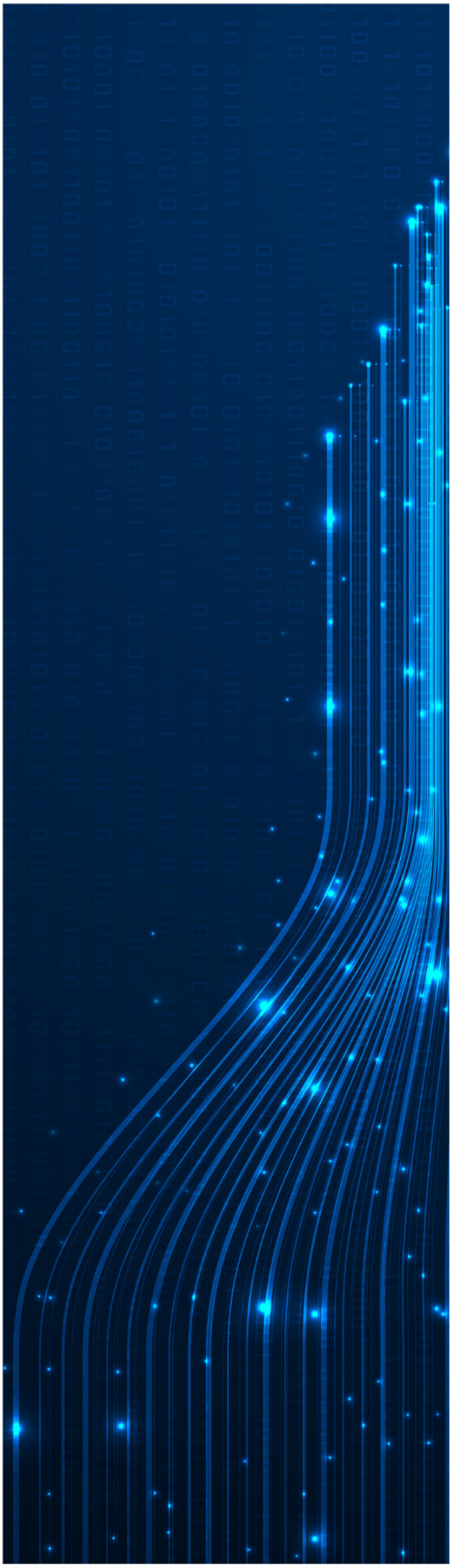
WISSEN



Streamlining

ML CODE MANAGEMENT & DEPLOYMENT IN DATABRICKS

Accelerate ML Development with Seamless Code Movement with DevOps CI/CD

www.wissen.com

Table of Contents

Abstract

Databricks is a popular cloud-based analytics platform that enables data scientists and engineers to build and deploy machine learning (ML) models at scale. However, managing and moving ML code across different environments, teams, and stages of the development lifecycle can be challenging. This whitepaper discusses the current problems associated with ML code movement in Databricks and proposes a solution using DevOps Continuous Integration/Continuous Deployment (CI/CD) practices for efficient code management and version control.

Who Should Read This

This whitepaper is intended for data scientists, machine learning engineers, software developers, and DevOps professionals who work with Databricks and are involved in the development, deployment, and management of machine learning models. It is also relevant for project managers and decision-makers who oversee ML projects and want to leverage DevOps CI/CD practices for improved code management, version control, and deployment efficiency in Databricks. The whitepaper provides insights, strategies, and best practices to streamline ML code movement and enhance collaboration, scalability, and reliability in Databricks using DevOps CI/CD methodologies.

Written By

Udaya Bhanu Koneni Reddy

Reviewed By

Sameer Pathare

Introduction

Background and Context

Databricks is a cloud-based unified analytics platform that is widely used for machine learning (ML) development and deployment. It provides an integrated environment that combines data engineering, data science, and collaborative features to facilitate efficient ML workflows, enabling data scientists and engineers to harness the potential of big data, distributed computing, and collaborative workflows

As a platform, Databricks offers several key features and capabilities for ML development:

Unified Workspace	• Databricks provides a unified workspace where data scientists and engineers can work collaboratively on ML projects. It includes notebooks for interactive coding, data exploration, and experimentation
Scalable Data Processing	• Databricks leverages the power of Apache Spark for scalable data processing. It enables distributed computing across clusters, allowing ML models to handle large volumes of data efficiently
Integrated ML Libraries	• Databricks provides a rich set of integrated ML libraries and frameworks, such as MLlib and TensorFlow, for building and training ML models. These libraries offer a wide range of algorithms and tools for data preparation, feature engineering, and model training
Managed Spark Clusters	• Databricks simplifies the management of Spark clusters by providing a fully managed environment. It takes care of cluster provisioning, configuration, and scaling, allowing users to focus on ML development rather than infrastructure management
AutoML and Model Management	• Databricks provides AutoML capabilities that automate the process of model selection and hyperparameter tuning. It also offers tools for model management, allowing users to track, deploy, and monitor ML models in production
Integration with Data Sources and Services	• Databricks seamlessly integrates with various data sources, including data lakes, data warehouses, and streaming platforms. It also provides connectors to popular data storage systems, such as Amazon S3, Azure Data Lake Storage, and Apache Kafka, enabling easy data ingestion and integration

Problem Statement

The process of handling ML code can be quite difficult and can create a lot of problems that slow down progress, make collaboration challenging, and make it harder to deploy ML models effectively. This can lead to decreased productivity, longer timeframes for development, increased risks during deployment, and difficulty in reproducing and tracing previous ML projects. To overcome these challenges, we need a solution that combines DevOps practices like Continuous Integration and Continuous Deployment, which are focused on efficiently managing and controlling code versions in Databricks. By implementing such a solution, we can enhance productivity, streamline development cycles, reduce deployment risks, and ensure that ML projects are easily traceable and reproducible.

Current Problems in ML Code Movement:

The current problems associated with ML code movement in Databricks can be summarized as follows:

Lack of Version Control:

- Databricks can benefit from further enhancements in version control capabilities to streamline change tracking, code version management, and collaboration among data scientists and development teams. Strengthening version control would help ensure a comprehensive history of code modifications, facilitate comparison between versions, and enable easy reversion to previous iterations when necessary.

Code Movement and Deployment Challenges:

- Efficiently deploying ML code from development to production environments in Databricks requires further advancements to minimize manual processes and associated inconsistencies or errors. Streamlined deployment mechanisms and automated workflows are crucial to ensure smooth transitions across different stages of the development lifecycle. The current absence of such mechanisms poses challenges like configuration drift and scalability concerns, especially when dealing with numerous ML models. Enhancements in this area would greatly improve the efficiency and reliability of deploying ML code in Databricks.

By leveraging DevOps CI/CD practices, data scientists and ML engineers can automate code integration, testing, and deployment processes. This enables faster feedback loops, ensures consistent code quality, facilitates collaboration among teams, and provides reliable versioning and traceability.

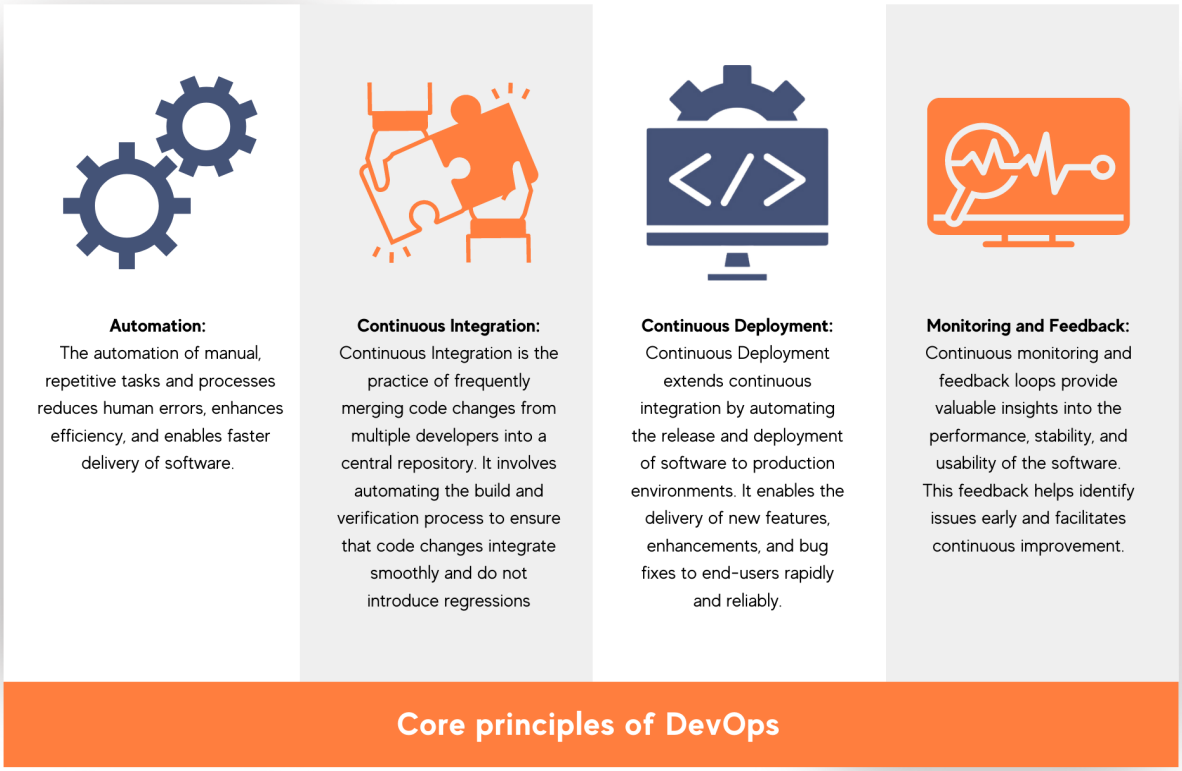
Introduction to DevOps CI/CD:

DevOps (Development and Operations) is a set of practices that combines software development (Dev) and IT operations (Ops) to enhance collaboration, streamline processes, and deliver high-quality software products more efficiently. Continuous Integration/Continuous Deployment (CI/CD) is a key component of the DevOps methodology, focusing on automation, rapid feedback, and continuous improvement throughout the software development lifecycle.

Understanding DevOps

DevOps emphasizes breaking down silos between development teams and operations teams, fostering a culture of collaboration, shared responsibilities, and automation. It promotes close cooperation and communication between different stakeholders involved in software development, including developers, testers, system administrators, and operations personnel.

The core principles of DevOps include:



Continuous Integration (CI)

Continuous Integration focuses on automating the process of integrating code changes from multiple developers into a shared repository.

Key aspects of CI include:

a. **Code Versioning and Repository Management:** Developers use a version control system, such as Git, to manage code versions, track changes, and collaborate effectively.

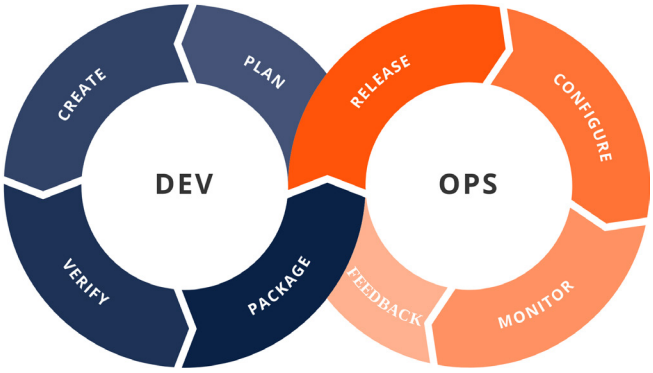
b. **Automated Build and Test:** CI tools automatically build the software and run tests to verify the integrity of the codebase. This helps catch integration issues and regressions early inthe development cycle.

c. **Code Quality and Static Analysis:** CI pipelines often include code quality checks and static analysis tools to enforce coding standards, identify potential issues, and maintain a high level of code quality.

Continuous Deployment (CD)

Continuous Deployment (CD) automates the release and deployment of software changes to production environments.

Key aspects of CD include:



a. **Automated Deployment Pipelines:** CD pipelines define the steps and actions required to deploy software changes, such as packaging, configuration management, infrastructure provisioning, and application deployment.

b. **Environment Management:** CD enables the management of different environments (e.g., development, testing, staging, production) with consistent configurations and reproducibility.

c. **Rollbacks and Roll-forwards:** CD pipelines should include mechanisms to rollback or rollforward changes in case of issues or failures during the deployment process.

d. **Release Orchestration:** CD ensures a smooth and coordinated release process, minimizing downtime and reducing the impact on end-users.

By implementing DevOps CI/CD practices, organizations can accelerate software delivery, reduce errors, enhance collaboration, and achieve a higher level of code stability and reliability. When applied to ML code movement in Databricks, DevOps CI/CD enables seamless integration, testing, deployment, and version control of ML models, resulting in increased productivity, improved collaboration, and efficient deployment processes.

Applying DevOps CI/CD to Databricks:

To optimize ML code movement in Databricks, organizations can leverage DevOps Continuous Integration/Continuous Deployment (CI/CD) practices. Implementing these practices helps streamline code management, version control, and deployment processes. Here are key steps for applying DevOps CI/CD to Databricks:

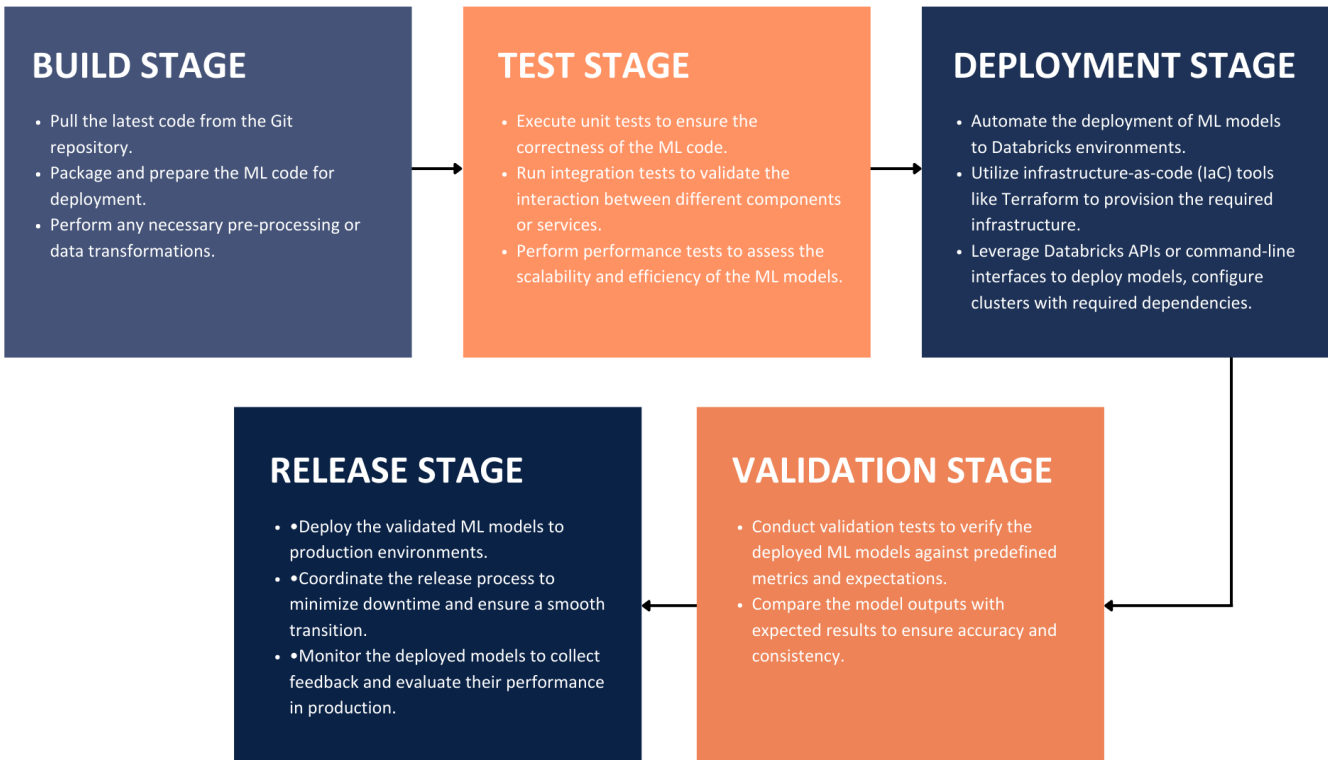
Version Control with Git

Integrate Databricks with a version control system like Git to enable efficient code versioning, branching, merging, and collaboration. Data scientists and engineers can leverage **Git** repositories to manage ML code, track changes, and work on different branches simultaneously. This integration allows for a clear history of modifications, easy collaboration, and the ability to revert to previous versions if necessary.

CI/CD Pipeline Setup

Set up a CI/CD pipeline to automate code integration, testing, and deployment processes. Several popular CI/CD tools like **Jenkins, Azure DevOps, or GitLab** can be used to orchestrate the pipeline. The pipeline can be triggered automatically whenever changes are pushed to the **Git** repository, ensuring continuous integration and rapid feedback.

The CI/CD pipeline for Databricks can include the following stages:

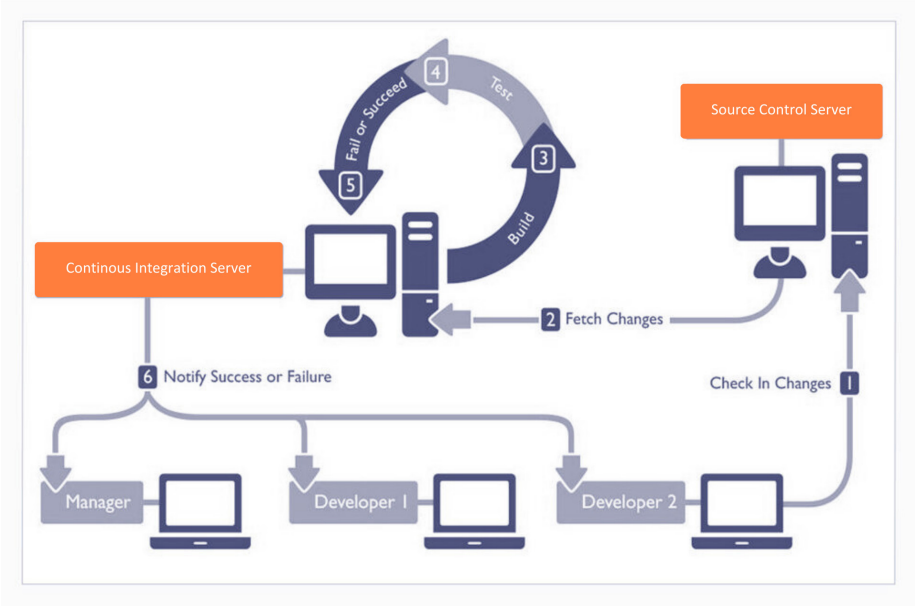


Unit Testing and Code Quality

Incorporate automated testing and code quality checks into the CI/CD pipeline to ensure the reliability and robustness of ML code in Databricks. Implement unit tests to verify the correctness of individual components, as well as integration tests to assess the functionality of the ML models within the broader system. Additionally, utilize code quality analysis tools to enforce coding standards, identify potential issues, and maintain a high level of code quality.

Deployment Automation

Automate the deployment process of ML models from development to production environments in Databricks. Leverage Databricks APIs and IaC tools like Terraform to define the infrastructure, cluster configurations, and dependencies required for model deployment. This automation streamlines the deployment process, reduces manual errors, and ensures consistent and reproducible deployments across different environments.



By implementing these DevOps CI/CD practices in Databricks,organizations can achieve several benefits, including:

- Faster development cycles and reduced time-to-market for ML models.
- Improved collaboration and version control among data scientists and development teams.
- Enhanced code quality and reliability through automated testing and code analysis.
- Streamlined and automated deployment processes with reduced manual intervention.
- Efficient scaling and reproducibility of ML models across different environments.

It is important to consider best practices for managing credentials, handling data dependencies, and aligning with Databricks workspace organization to ensure a smooth integration of DevOps.

Implementation Using Git and GitLab

Below are the implementation steps for DevOps CI/CD accelerator for Databricks ML using Git and GitLab.

1. **Setting up Git Repository:**
   - Create a new Git repository or use an existing one to store your ML code and Databricks notebooks.
   - Initialize the repository with a 'README.md' file and a '.gitignore' file to exclude unnecessary files from version control, also Clone the repository locally.

```bash
# Create a new repository on GitLab
# Initialize Git repository
git init
# Create a README file
echo "# My Databricks ML Project" >> README.md
# Commit the changes
git add README.md
git commit -m "Initial commit"
# Add GitLab repository as remote
git remote add origin <gitlab_repository_url>
# Push the code to GitLab repository
git push -u origin master
# Clone repository locally
git clone <repository_url>
```

2. **Configure GitLab Repository and CI/CD Pipeline:**
   - Create a new project in GitLab and associate it with your Git repository.
   - Configure the project settings, including repository access, permissions, and branch protection rules.
   - Set up the CI/CD pipeline in GitLab by creating a '.gitlab-ci.yml' file in the root of your repository.
   - Customize the pipeline stages and scripts based on your specific requirements and workflows.
   - Define the stages for the pipeline, such as build, test, deploy, and release.
   - Specify the jobs for each stage, including the necessary scripts, commands, and environment configurations.
   - Configure the pipeline triggers, such as triggering the pipeline on each commit or scheduled intervals.

```yaml
stages:
- build
- test
- deploy
- release

build:
stage: build
script:
- # Commands to build your ML code

test:
stage: test
script:
- # Commands to run tests on your ML code

deploy:
stage: deploy
script:
- # Commands to deploy your ML models to Databricks

release:
stage: release
script:
    - # Commands to release your validated ML models to production
environments
```

3. **Databricks Integration with Git and GitLab:**
   - Connect your Databricks workspace with the Git repository using the Git integration feature provided by Databricks.
   - Set up the necessary credentials and configurations to enable synchronization between Databricks and Git.

4. **Code Development and Version Control:**
   - Develop your ML code, notebooks, and scripts locally on your development machine.
   - Use Git commands ('git add', 'git commit', 'git push', etc.) to track and version control your code changes.
   - Create branches for different features or bug fixes, and merge them into the main branch using Git workflows such as Gitflow or feature branches

5. **Continuous Integration (CI) with GitLab:**
   - Every time a code change is pushed to the Git repository, the CI/CD pipeline in GitLab automatically triggers the CI process.
   - The CI process typically involves building and testing your ML code to ensure its correctness and quality.
   - Within the CI stage of the pipeline, you can use Databricks CLI or REST API to create clusters, run notebooks, and execute tests.

```yaml
build:
stage: build
script:
- # Commands to build your ML code
- databricks clusters create ...

test:
stage: test
script:
    - # Commands to run tests on your ML code
- databricks workspace import ...
- databricks notebooks run ...
```

6. **Continuous Deployment (CD) to Databricks:**
   - After the code passes the CI stage, the CD process deploys your ML models to Databricks for production use.
   - Use the Databricks CLI or REST API to interact with Databricks, import notebooks, create and configure clusters, and deploy ML models.
   - Customize the deployment scripts according to your ML model deployment requirement

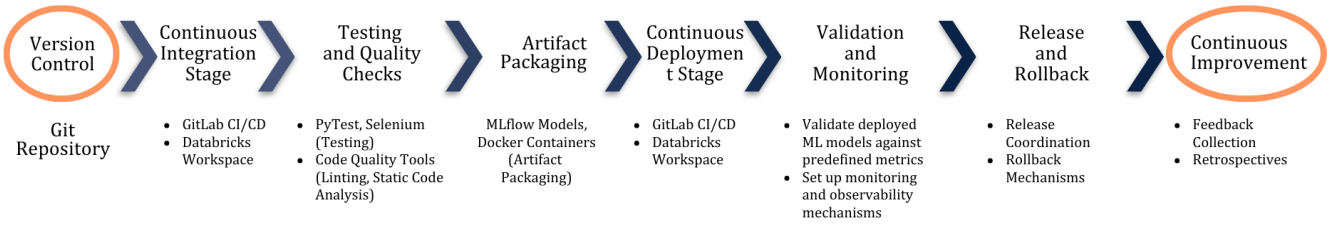
```yaml
deploy:
stage: deploy
script:
- # Commands to deploy your ML models to Databricks
- databricks workspace import ...
- databricks clusters create ...
- databricks notebooks run ...

release:
stage: release
script:
    - # Commands to configure the Databricks CLI and authenticate
    - # Additional setup steps, such as environment configuration
and dependency installation
- # Import the ML model notebooks or scripts

- databricks workspace import ...

- # Create or configure the production cluster
- databricks clusters create ...
    - # Create a job to execute the ML model notebooks or scripts
- databricks jobs create ...
```

   - The release stage is defined, and the corresponding script section contains the commands required to deploy the ML models to Databricks.
   - First, you need to configure the Databricks CLI and authenticate it with the necessary credentials for the deployment.
   - Next, you can perform any additional setup steps required for the deployment, such as configuring environment variables or installing dependencies.
   - Then, use the 'databricks workspace import' command to import the ML model notebooks or scripts to the Databricks workspace.
   - After that, use the 'databricks clusters create' command to create or configure the production cluster to host the deployed models.

- Finally, use the 'databricks jobs create' command to create a job that executes the ML model notebooks or scripts periodically or based on a trigger.
- By incorporating the release stage in your CI/CD pipeline, you can automate the deployment of your ML models to production, ensuring consistency, reliability, and controlled release processes.

7. Continuous Improvement:

- Collect feedback, monitor pipeline performance, and regularly update and improve the CI/CD pipeline and ML models.

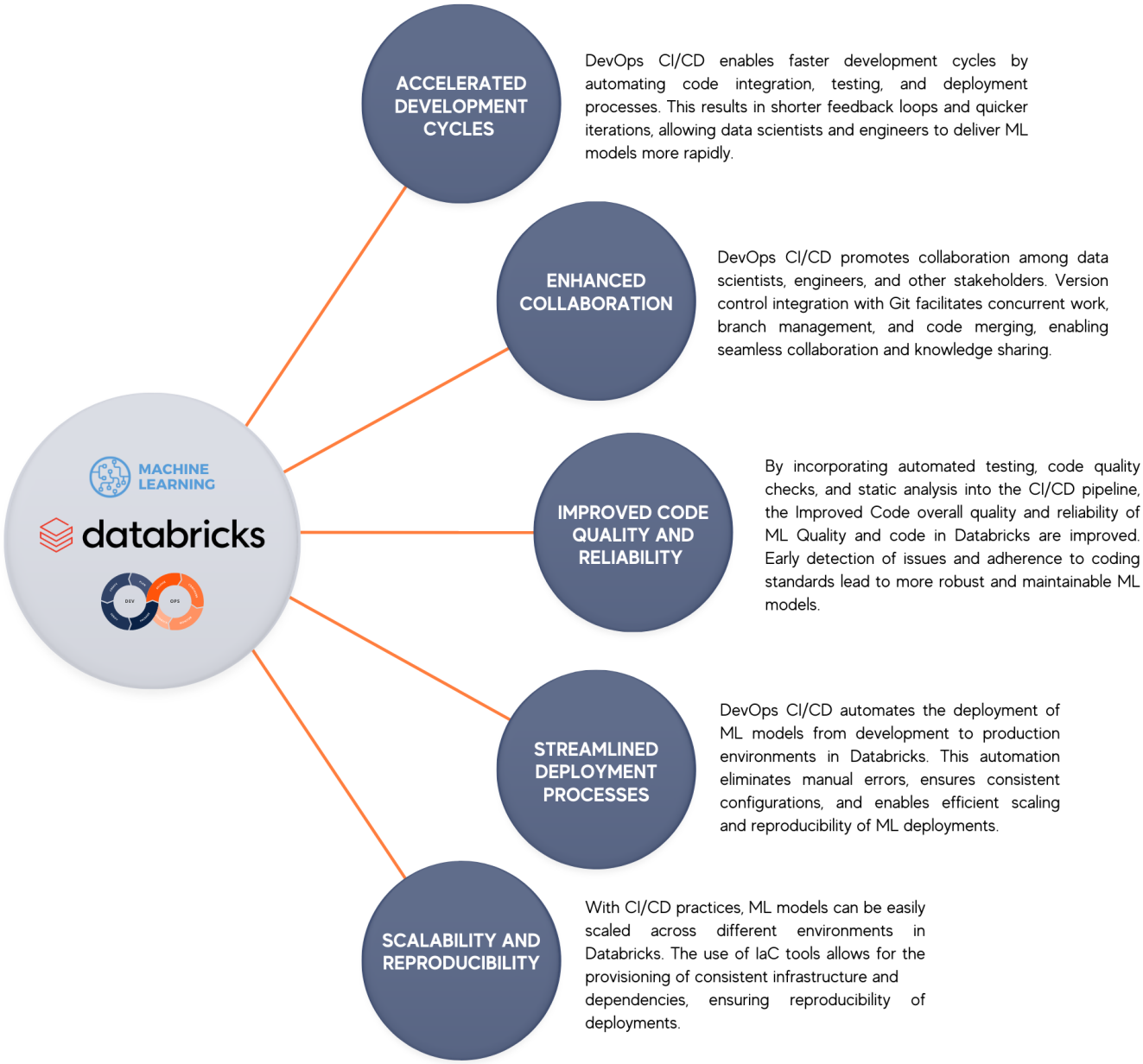


This process flow demonstrates the integration of Git, GitLab, and Databricks for implementing DevOps CI/CD in Databricks ML. The provided code snippets are examples, and you can customize them based on your specific project requirements and preferences.

Benefits and Considerations:

Implementing DevOps CI/CD practices for ML code movement in Databricks offers several benefits and brings about improvements in code management, version control, and deployment processes. However, it is essential to consider certain factors and best practices. Let's explore the benefits and considerations associated with applying DevOps CI/CD to Databricks:

Benefits of DevOps CI/CD in Databricks



Considerations and Best Practices

a. **Security and Credentials Management:** Ensure proper security measures are in place to protect sensitive information such as access tokens, authentication credentials, and data sources. Implement secure storage mechanisms and follow best practices for credential management to prevent unauthorized access.

b. **Data Dependencies:** Consider the dependencies and data sources required for ML models in Databricks. Ensure that the CI/CD pipeline takes into account the provisioning and management of data dependencies, such as datasets, databases, or data pipelines.

c. **Databricks Workspace Organization:** Establish a well-structured workspace organization in Databricks to facilitate efficient code management, collaboration, and deployment. Define clear guidelines for notebook organization, folder structures, and naming conventions to ensureconsistency and ease of navigation.

d. **Monitoring and Observability:** Implement monitoring and observability practices to gain insights into the performance and behavior of ML models in production. Set up logging, metrics collection, and alerting mechanisms to detect issues and anomalies, allowing for proactive remediation.

e. **Continuous Improvement:** Embrace a culture of continuous improvement by regularly reviewing and refining the CI/CD pipeline and development processes. Encourage feedback from the development team, collect metrics, and conduct retrospectives to identify areas for optimization and enhancement.

f. **Training and Documentation:** Provide training and documentation to the development team on CI/CD practices, Git integration, and Databricks-specific workflows. Ensure that team members are proficient in using the CI/CD tools and understand the established processes and best practices

Conclusion:

Summary

In this whitepaper, we explored the challenges associated with ML code movement in Databricks and how organizations can overcome them by applying DevOps CI/CD practices. We discussed the importance of version control, automation, collaboration, and code quality in ML development and deployment.

By integrating Databricks with Git for version control and setting up a CI/CD pipeline, organizations can streamline code management, testing, and deployment processes. Automated testing, code quality checks, and infrastructure-as-code tools enhance code reliability and reproducibility. The benefits of applying DevOps CI/CD in Databricks include accelerated development cycles, improved collaboration, and efficient deployment of ML model

Future Directions

Looking ahead, there are several areas organizations can focus on to further enhance ML development and deployment in Databricks using DevOps CI/CD:

- **Model Monitoring and Management:** Develop robust monitoring and management practices to track the performance of deployed ML models, detect anomalies, and enable efficient model updates and retraining.
- **Governance and Compliance:** Strengthen governance practices to ensure compliance with data privacy regulations, security standards, and organizational policies throughout the ML lifecycle.
- **Automated Data Pipelines:** Integrate automated data pipelines with the CI/CD process to enable efficient data ingestion, transformation, and feature engineering, supporting a streamlined end-to-end ML workflow.
- **Experiment Tracking and Reproducibility:** Leverage ML experimentation frameworks like MLflow to track and manage experiments, reproduce results, and facilitate collaboration among data scientists.
- **Continuous Deployment to Edge Devices:** Extend the CI/CD pipeline to support the deployment of ML models to edge devices and IoT devices, enabling real-time inferencing and intelligent edge computing.
- **Model Explainability and Interpretability:** Explore techniques and tools for model explainability and interpretability to enhance transparency, trust, and regulatory compliance of ML models.

By focusing on these future directions, organizations can further optimize ML development anddeployment in Databricks, enabling continuous improvement, scalability, and innovation in their machine learning initiatives.

In conclusion, by adopting DevOps CI/CD practices in Databricks and considering future directions, organizations can unleash the full potential of their ML development efforts, achieve faster timeto-market, and deliver high-quality ML solutions that drive business value.

References:

Here are some references that can provide further information on the topics discussed in this whitepaper:

- Official documentation for Databricks, covering various aspects of ML code management, deployment, and version control.
- Databricks Documentation
- "Continuous Integration, Delivery, and Deployment" - Martin Fowler
- "Version Control with Git" - Pro Git book by Scott Chacon and Ben Straub
- "Continuous Deployment to Azure Databricks using Azure DevOps" - Microsoft Documentation
- "Best Practices for Deploying Machine Learning Models" - Martin Görner and Ryan Gillard (Google Cloud)
- Offers detailed documentation on Jenkins, an open-source automation server widely used for CI/CD. Jenkins Documentation
- Provides official documentation for Git, a widely adopted version control system. Git Documentation:
- "Continuous Integration vs. Continuous Deployment vs. Continuous Delivery" by Atlassian:

About Wissen

Founded in 2000 in the US, Wissen is an esteemed Information Technology company headquartered in Bangalore, India. With a global presence spanning offices in the US, India, UK, Australia, Mexico, and Canada, we offer end-to-end solutions to companies across sectors such as Banking and Financial Services, Telecom, Healthcare, Manufacturing, and Energy. Leveraging state-of-the-art infrastructure and development facilities worldwide, we have successfully delivered over $1 billion worth of projects for more than 25 Fortune 500 companies. With a highly skilled workforce of 4500+ professionals,

Services We Offer

Application Development

Big Data and Analytics

Artificial Intelligence and Machine Learning

Robotic Process Automation

Agile and DevOps

Visualization and Business Intelligence

Cloud and Mobility

Quality Assurance and Test Automation

Infrastructure Management

contact@wissen.com

WISSEN

www.wissen.com

India | UK | USA | Australia | Canada | Mexico | Vietnam