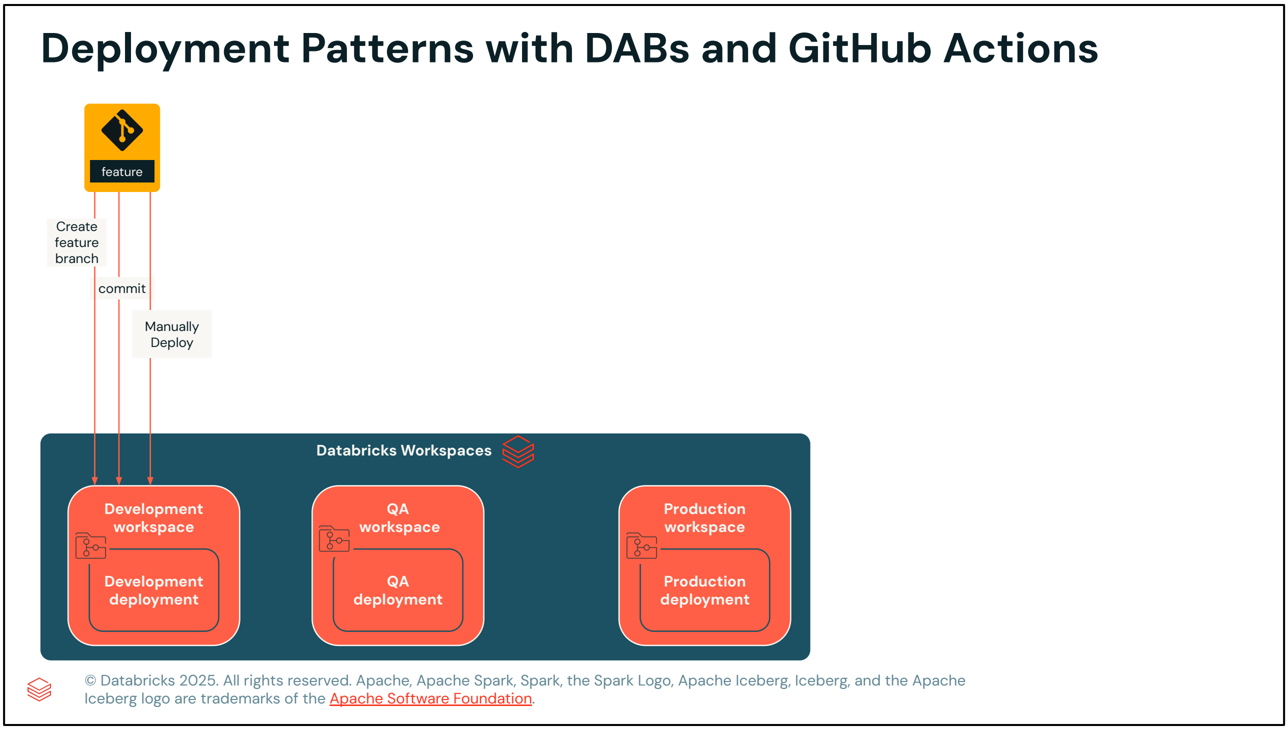
databricks

Doing More with DABs

LECTURE

Next Steps: Automated Deployment with GitHub Actions

In this lecture, we cover automated deployment using GitHub Actions with Databricks Asset Bundles (DABs) and an overview of Git integration in Databricks.



Deployment Patterns with DABs and GitHub Actions

feature

Create feature branch

commit

Manually Deploy

Databricks Workspaces

Development workspace

Development deployment

QA workspace

QA deployment

Production workspace

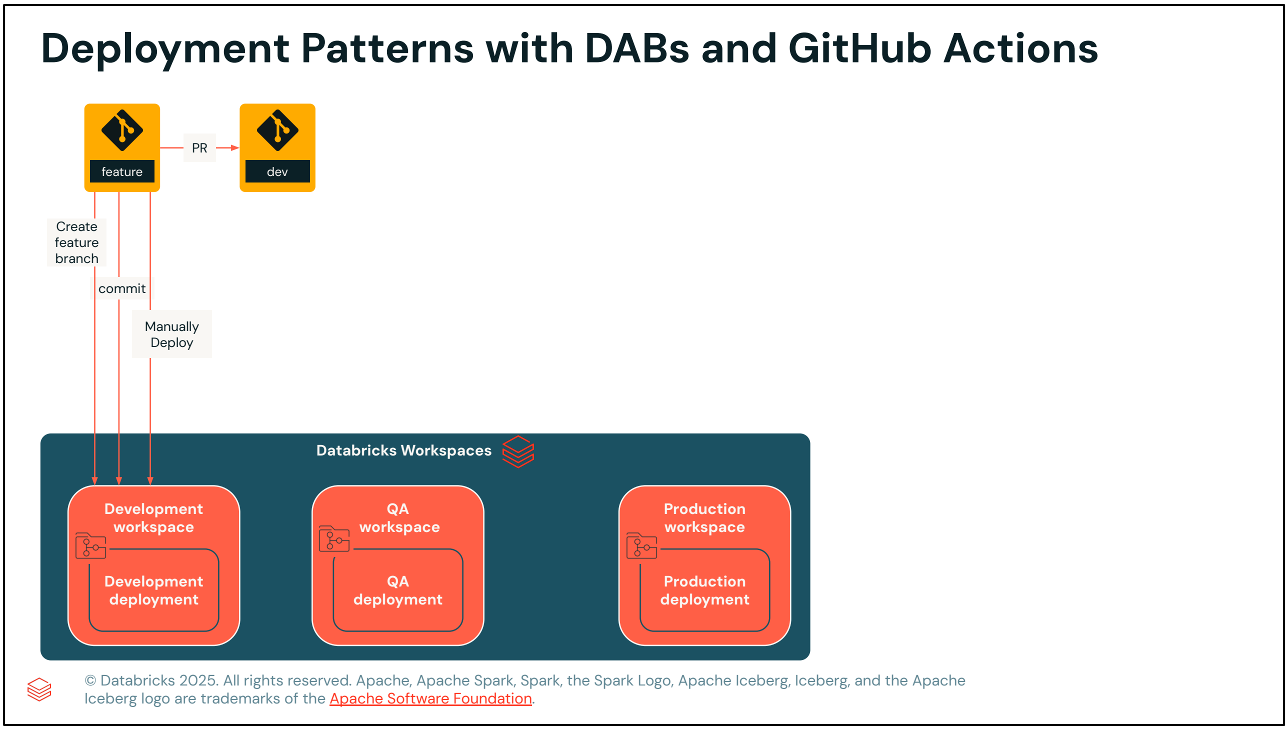
Production deployment

Introduction:

- Let's walk through a high-level overview of how we integrate GitFlow (which is an alternative Git branching model that involves the use of feature branches and multiple primary branches) with Databricks Asset Bundles, or DABs. This workflow streamlines the process of developing, testing, and deploying Databricks projects by leveraging GitFlow's structured branching model and GitHub Actions for automation.

Creation of Feature Branches:

- The process begins on the very left hand side with the developer creating a feature branch from the `dev` branch. This is where individual features or bug fixes are developed in isolation. Developers can deploy code directly to the DEV Databricks workspace using the databricks-cli



Deployment Patterns with DABs and GitHub Actions

feature

PR

dev

Create feature branch

commit

Manually Deploy

Databricks Workspaces

Development workspace

Development deployment

QA workspace

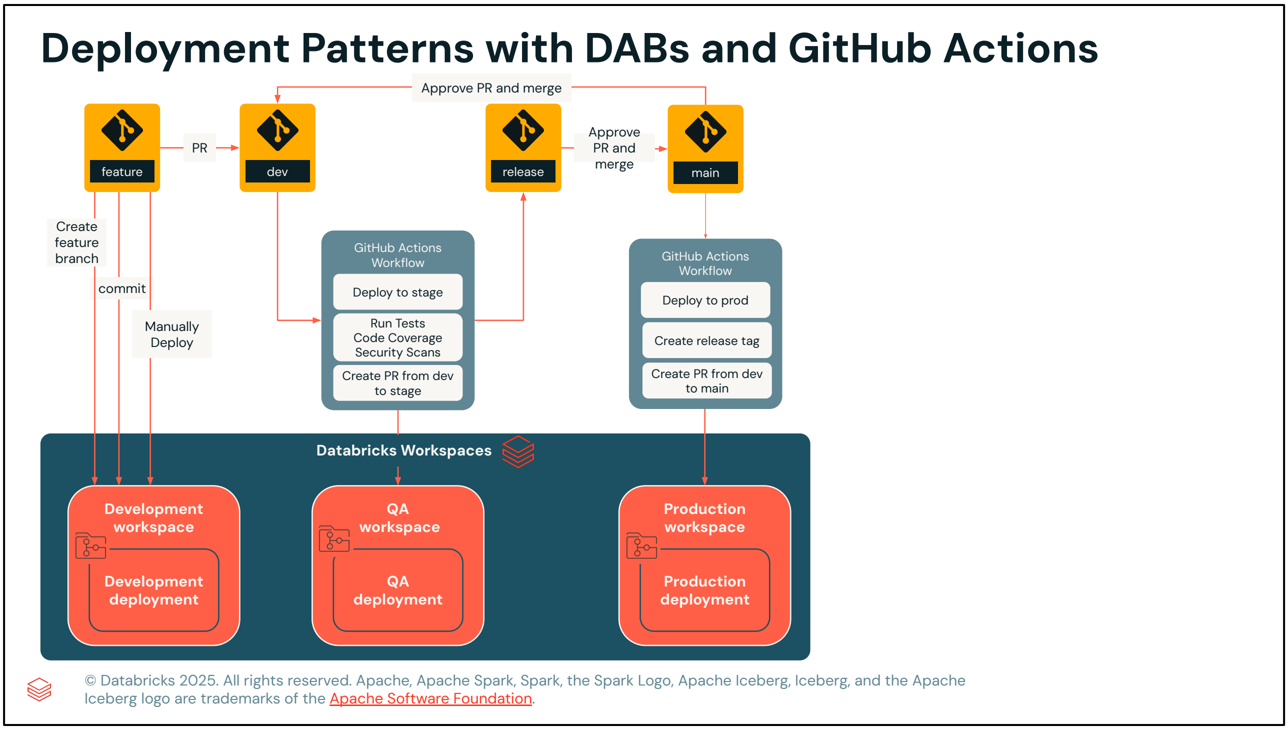
QA deployment

Production workspace

Production deployment

Pull Request Workflow:

- Once the development on a feature branch is complete, a pull request is opened. After the code is peer-reviewed, it is merged into the develop branch.



Deployment Patterns with DABs and GitHub Actions

feature

PR

dev

Approve PR and merge

release

Approve PR and merge

main

Create feature branch

commit

Manually Deploy

GitHub Actions Workflow

Deploy to stage

Run Tests
Code Coverage
Security Scans

Create PR from dev to stage

GitHub Actions Workflow

Deploy to prod

Create release tag

Create PR from dev to main

Databricks Workspaces

Development workspace

Development deployment

QA workspace

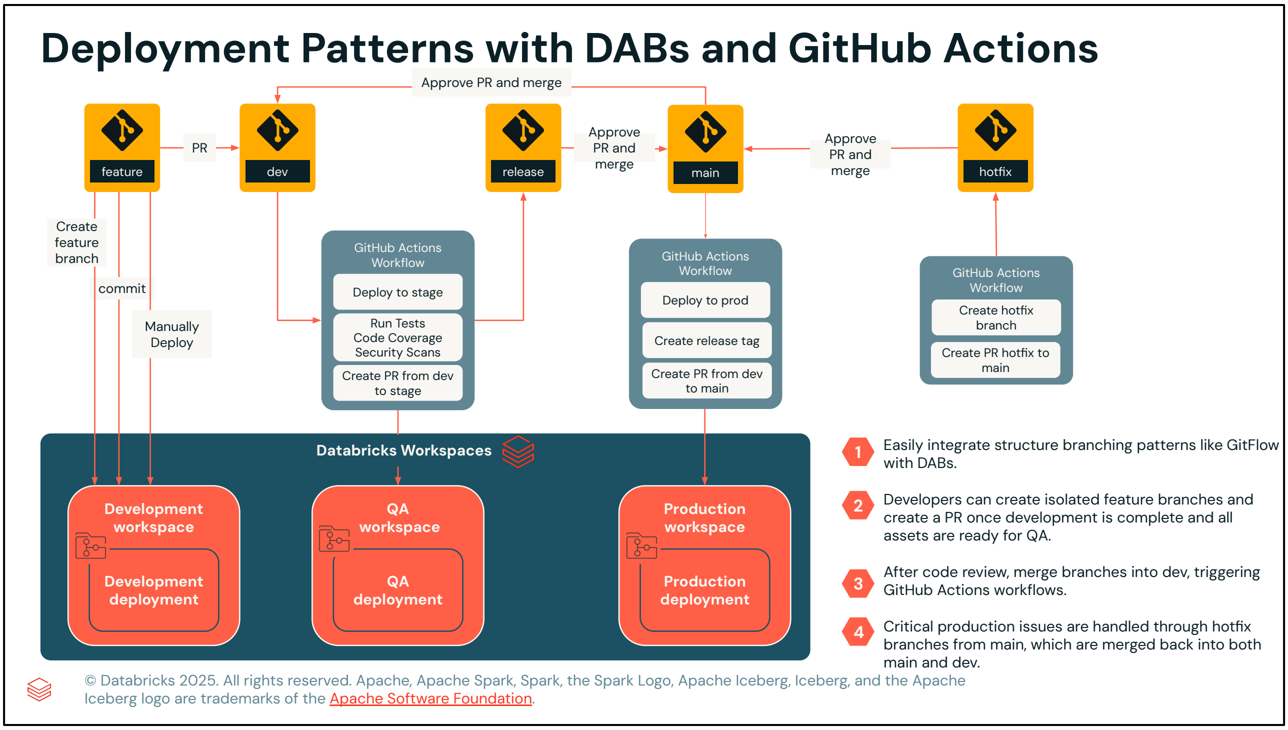
QA deployment

Production workspace

Production deployment

Deployment to QA and Production:

- After merging into develop, the next step involves triggering the GH Actions workflow for Drafting a release. This deploys the changes to the QA environment using DABs ,runs Tests, code coverage checks and security scans and finally creates the versioned release branch. It also creates a PR for the release branch to be merged into main.
- After the Release PR is closed, it triggers another GitHub actions workflow that deploys the bundle to PROD, creates a Release Tag and eventually, a PR for main branch to be merged into DEV.



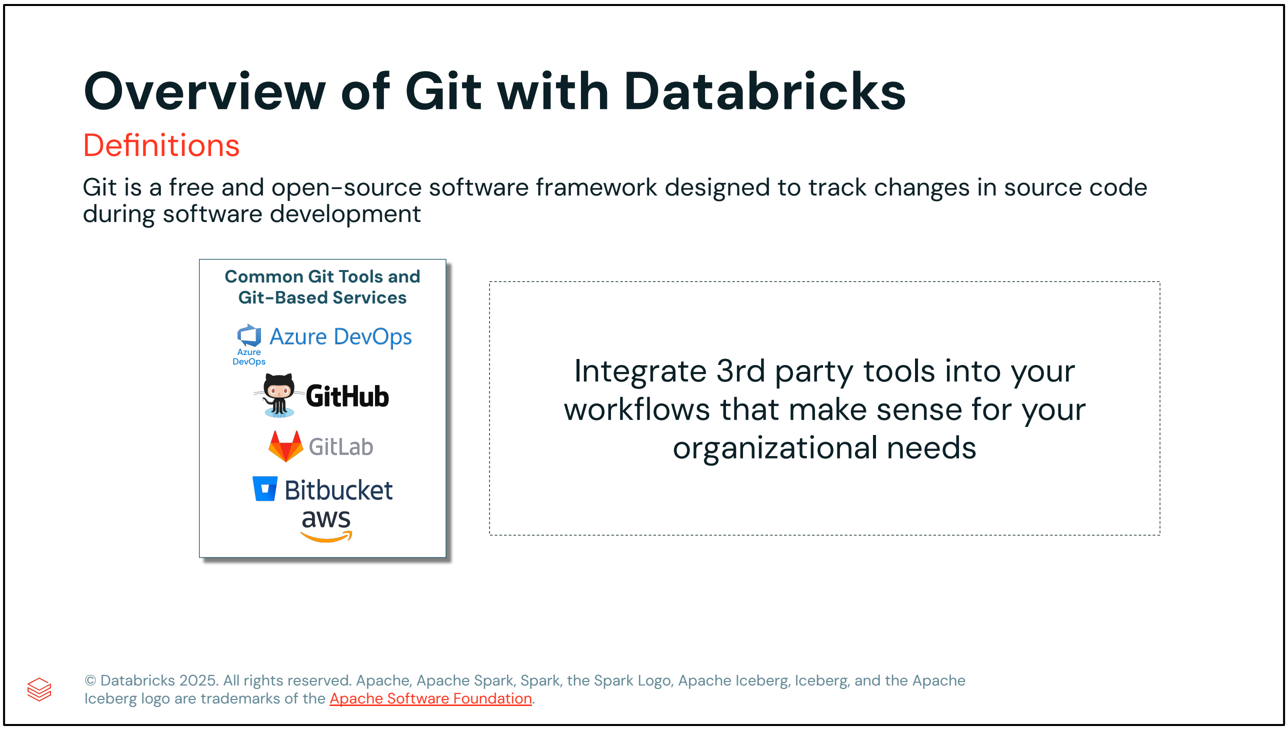
Deployment Patterns with DABs and GitHub Actions

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.

Release Branching and Hotfixes:

- Finally, If a critical issue arises in production, we handle it with a hotfix branch, which is branched from `main` and then merged back into both `main` and `dev` after the fix is applied. This ensures that our production environment remains stable while allowing ongoing development to continue.

Reference:
https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow



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

Common Git Tools and Git-Based Services

Azure DevOps

GitHub

GitLab

Bitbucket

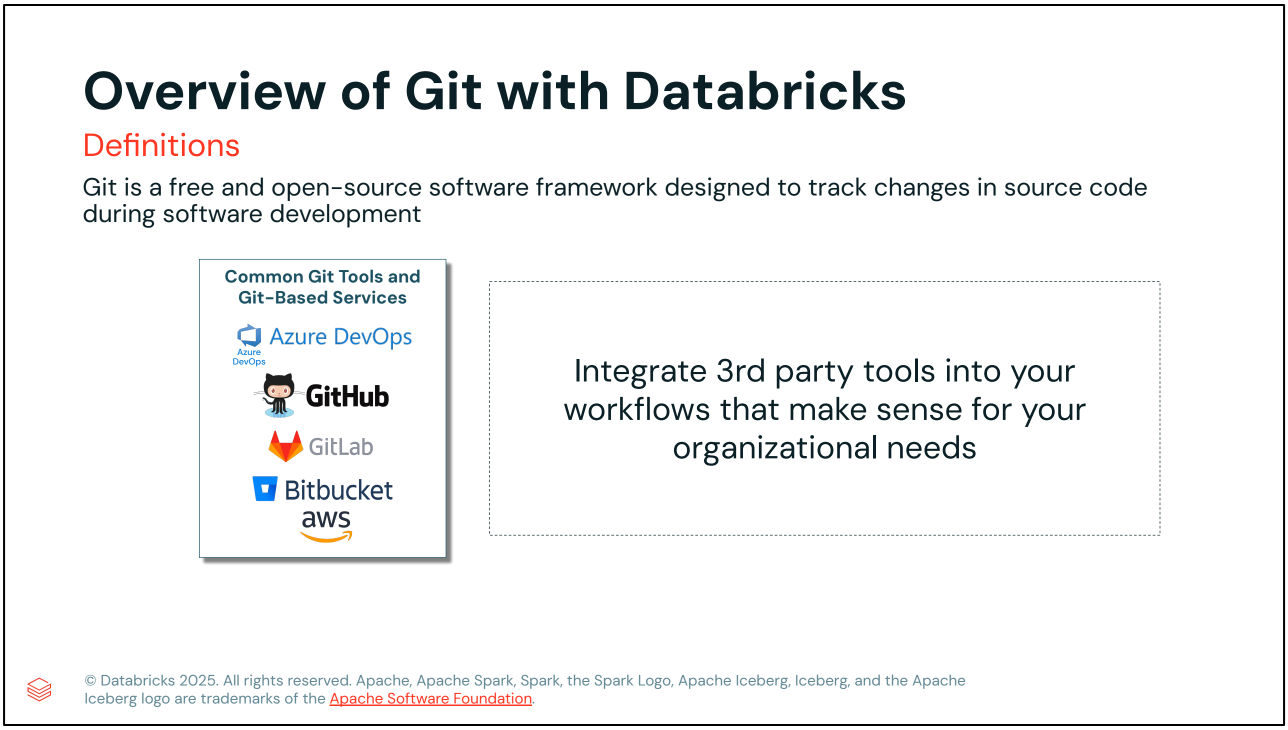
aws

Integrate 3rd party tools into your workflows that make sense for your organizational needs

Git Benefits: Version control enables tracking code changes, facilitating rollback and collaboration. Branching and merging allow multiple developers to work in parallel and integrate changes efficiently. A distributed workflow ensures each developer has a full local repository, enhancing flexibility and reliability. Git is optimized for high-performance handling of large projects, and its security features use cryptographic integrity checks to prevent data corruption.

Git Tools & Services

1. **GitHub, GitLab, Bitbucket, Azure DevOps** – Cloud-based repositories with CI/CD, issue tracking, and team collaboration.
2. **Git CLI & GUI Clients (e.g., SourceTree, GitKraken, VS Code Git Integration)** – Provides different interfaces for managing repositories.
3. **CI/CD Integration** – Automated testing and deployment pipelines.
4. **Code Review & Collaboration** – Features like pull requests and merge approvals streamline teamwork.
5. **Security & Access Control** – Role-based permissions and audit logs enhance repository security.



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

Common Git Tools and Git-Based Services

Azure DevOps

GitHub

GitLab

Bitbucket

aws

Integrate 3rd party tools into your workflows that make sense for your organizational needs

1. **Visual Git Client -** Databricks provides a user-friendly interface for common Git operations, which we will discuss about on the next slide.
2. **Seamless Integration -** Users can leverage remote Git repos while developing code inside Databricks notebooks
3. **CI/CD Capabilities -** The repos REST API enables integration of data and AI projects into CI/CD pipelines, allowing users to automate Git workflows

databricks

Thank you for completing this lesson and continuing your journey to develop your skills with us.