Continuous integration and continuous delivery

Introduction to CI/CD

Combines Continuous Integration (CI) + Continuous Delivery (CD)

Enables faster, safer, and automated software development and delivery

Continuous Integration (CI)

Developers merge code frequently—at least once a day

Automated builds and tests validate each integration.

Early bug detection and quick feedback

Continuous Delivery (CD)

Automates the delivery of applications to staging/testing environments

Ensures deployable builds at any time

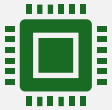
Optional: Can extend to Continuous Deployment (automatic production release)

Advantages of Implementing CI/CD

- Accelerated development cycles.
- Improved code quality and reliability.
- Enhanced collaboration among teams.

Key Components of a CI/CD Pipeline

Source Control: Version management of code.

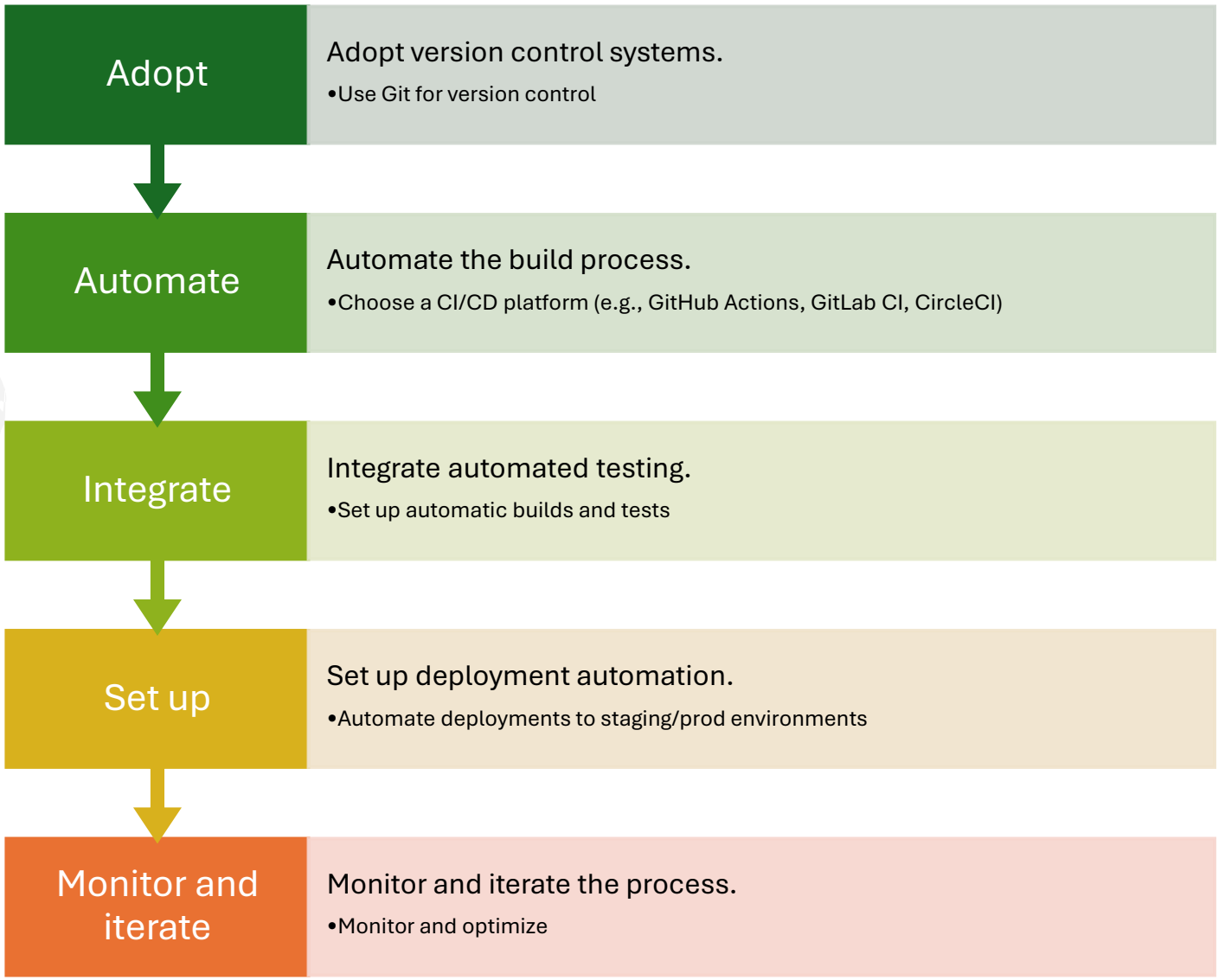


Build Automation: Compiling and packaging code.

Automated Testing: Validating code functionality.

Deployment Automation: Releasing code to environments.

Steps to Implement CI/CD



Real-World Example

Code pushed to GitHub triggers CI pipeline

Jenkins builds and tests the app

If successful, CD pushes to staging

Manual approval sends it to production

Embracing CI/CD for Modern Development

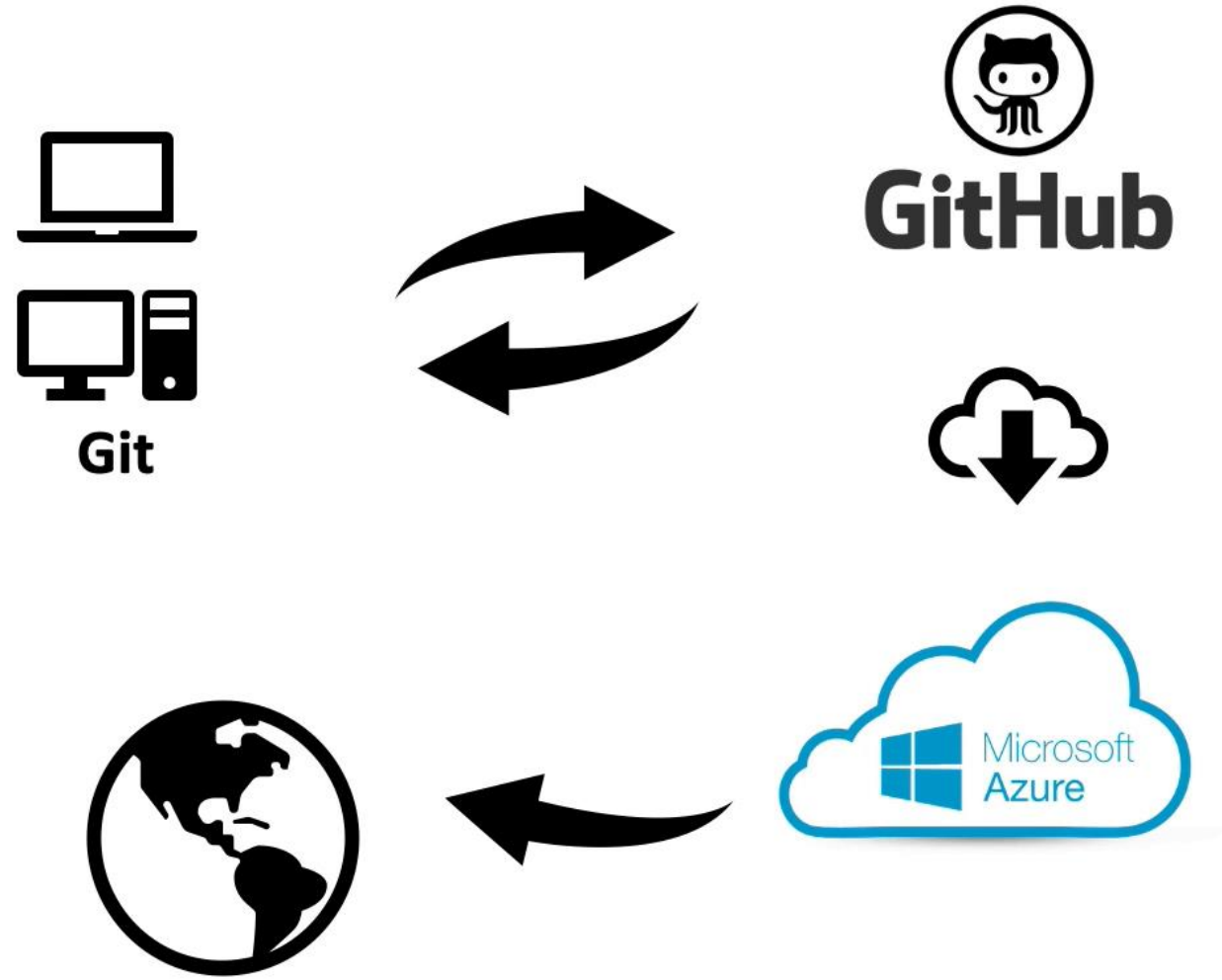
CI/CD enhances development efficiency.

Promotes high-quality, reliable software delivery.

Essential for agile and DevOps practices.


Windows Azure
practical workflow

Website Creation with GitHub and Microsoft Azure



Create static website