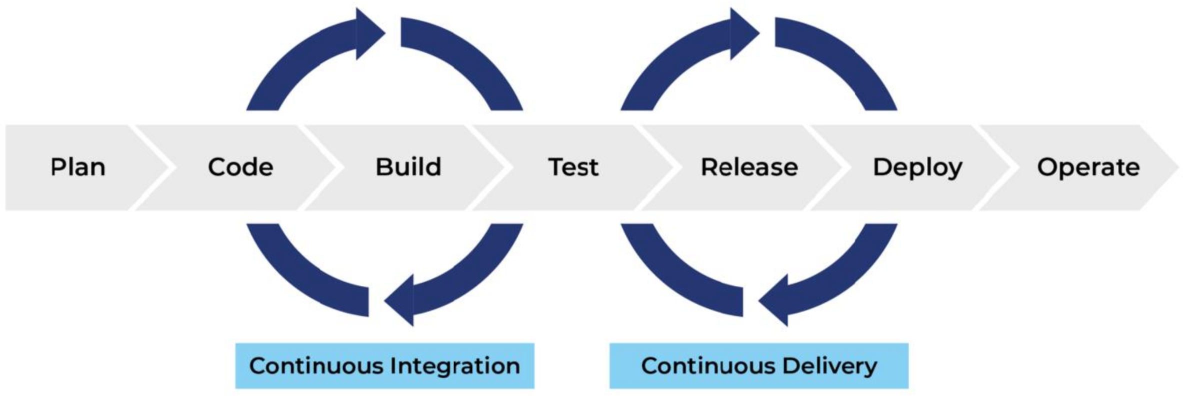
CI/CD Pipeline:

What is CI/CD?

CI/CD stands for:

- **Continuous Integration (CI):**
  Developers frequently merge code changes into a shared repository, followed by automated builds and tests.
- **Continuous Delivery (CD):**
  Automatically prepares code changes for release to production.
- **Continuous Deployment (CD):**
  Extends delivery by automatically deploying every validated change to production.

Goal: Automate the software delivery process, reduce manual errors, and ensure faster, reliable updates.



Why CI/CD?

- **Speed:** Faster integration and delivery of features.
- **Quality:** Early bug detection with automated tests.
- **Reliability:** Predictable and consistent deployments.
- **Collaboration:** Encourages small, frequent commits.
- **Reduced manual work:** Automated testing, builds, and deployments.

CI/CD Pipeline Stages

Stage 1: Source Stage

- Developers push code to a **Version Control System (VCS)** like Git.
- Triggers the pipeline.

Stage 2: Build Stage

- Code compiled and packaged.
- Dependency management handled.
- Tools: Maven, Gradle, npm.

Stage 3: Test Stage

- Automated unit tests, integration tests, and functional tests are executed.
- Tools: JUnit, Selenium, PyTest.

Stage 4: Deploy Stage

- Code is deployed to staging or production environments.
- May include containerization using Docker, orchestration with Kubernetes.

Stage 5: Monitor Stage

- Application performance and logs are monitored post-deployment.
- Rollbacks handled if issues detected.
- Tools: Prometheus, Grafana, ELK Stack.

CI/CD Pipeline Example:

Example Tools:

Pipeline Phase	**Tools**
VCS	Git, GitHub, GitLab
CI	Jenkins, GitLab CI, CircleCI
CD	Spinnaker, ArgoCD, GitHub Actions
Containers	Docker
Orchestration	Kubernetes
Monitoring	Prometheus, Grafana

Challenges in CI/CD

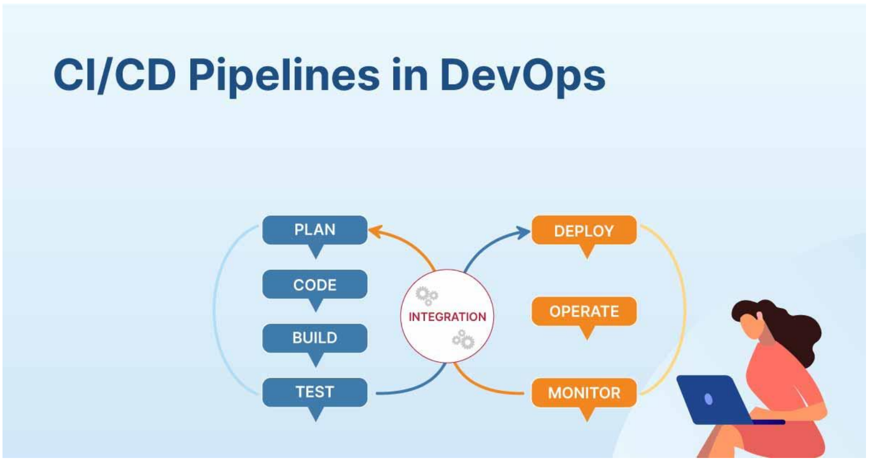
- Managing complex microservices pipelines.

- Handling database migrations automatically.
- Ensuring test reliability to prevent false positives/negatives.
- Securing pipelines and credentials.
- Rollback strategies for faulty deployments.

CI/CD in DevOps Culture

CI/CD is a **core DevOps practice**:

- Encourages **collaboration between developers and operations.**
- Aligns with **agile methodologies** for continuous delivery.
- Reduces **time to market** for new features.



Conclusion: Why CI/CD Matters

- Enables **automation and consistency** in delivering software.
- Minimizes **manual errors** and ensures **faster feedback loops**.
- Empowers teams to innovate with **safe, rapid deployments**.
- Becomes essential in **cloud-native, microservices, and agile environments**.