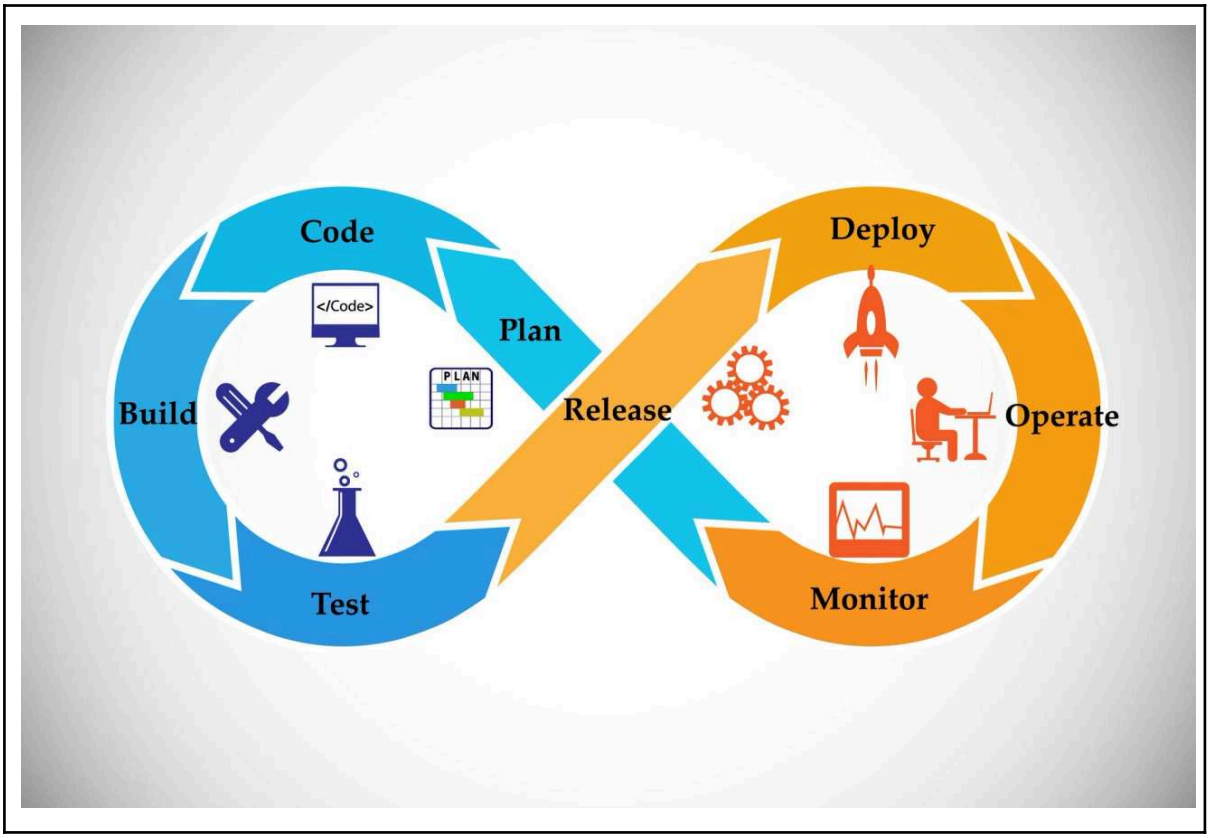
Developer’s Workflow

A developer’s workflow refers to the series of steps, practices, and tools that a developer or a team of developers follows to write, test, collaborate on, and deploy code effectively. This workflow is designed to streamline the software development process, enhance productivity, reduce errors, and ensure high quality code output.



A well defined developer’s workflow incorporates various stages of development, from coding to deployment, and often integrates tools for version control, code review, testing, and continuous integration & deployment (CI/CD).

Key Stages

---

1. **Coding:** Developers work on their local machine.
   Steps:
   - Pull latest code from main branch
   - Create a feature branch
   - Write code:
     - Follow coding standards
     - Write unit tests
     - Run app locally
   - Ensure:
     - Code builds
     - Tests pass
     - No lint errors

   Output: Working feature in a separate branch

2. **Version Control (Git):** Used to track changes and collaborate.
   Key Practices:
   - Small, meaningful commits
   - Push branch to remote
   - git pull origin main --rebase

   Why does this matter:
   - History is traceable
   - Easy rollback
   - Parallel development

3. **Pull Request / Merge Request:** Developer opens a PR (GitHub/GitLab/Bitbucket).

   PR includes:

   - Description of changes
   - Linked ticket/issue
   - Screenshots (if UI)
   - Test results

   Purpose:

   - Enable review
   - Trigger automated checks

4. **Code Review:** *Other developers review the PR.*

   Review focuses on:

   - Correctness
   - Readability & maintainability
   - Performance
   - Security
   - Test coverage

   Possible outcomes:

   - ✅ Approved
   - 🔁 Changes requested
   - ❌ Rejected

   Best practices:

   - No direct commits to main
   - At least 1–2 reviewers
   - Constructive feedback

5. **Continuous Integration (CI):** Automatically triggered when PR is opened or updated.

   CI Pipeline runs:

   - Build
   - Linting
   - Unit tests
   - Integration tests
   - Static analysis (SonarQube, CodeQL)

   Example tools:

   - GitHub Actions
   - GitLab CI
   - Jenkins
   - CircleCI

   Rule:
   ❌ If CI fails → PR cannot be merged

6. **Merge to Main Branch:**

   Once:

   - Code review is approved
   - CI passes

   PR is merged into:

   - `main` / `develop`

   Strategies:

   - Squash merge
   - Rebase merge
   - Merge commit

   Result: Code becomes part of the official codebase

7. **Continuous Deployment / Delivery (CD):** After merge, deployment happens automatically or manually.

   Typical environments:

   - Development
   - Staging / QA
   - Production

   CD steps:

   - Build artifacts (Docker image, binary)
   - Run migrations
   - Deploy to servers/cloud
   - Smoke tests / health checks

   Tools:

   - Docker
   - Kubernetes
   - AWS / GCP / Azure
   - ArgoCD, Flux
   - GitHub Actions

8. **Monitoring & Feedback:**

   After Deployment:

   - Monitor logs & metrics
   - Track errors (Sentry)
   - Observe performance
   - Collect user feedback

   If issues occur:

   - Rollback
   - Hotfix branch
   - Patch release

Benefits of a Developer's Workflow

Improved Collaboration: Clear workflows and processes facilitate better collaboration among team members, making it easier to understand who is responsible for what.

Higher Code Quality: Code reviews, automated tests, and CI/CD practices help maintain a high standard of code quality.

Reduced Errors: Automated testing and deployment reduce the likelihood of human error, ensuring more reliable releases.

Continuous Feedback Loop: Regular monitoring and feedback help teams quickly identify and address issues, continuously improving the product.