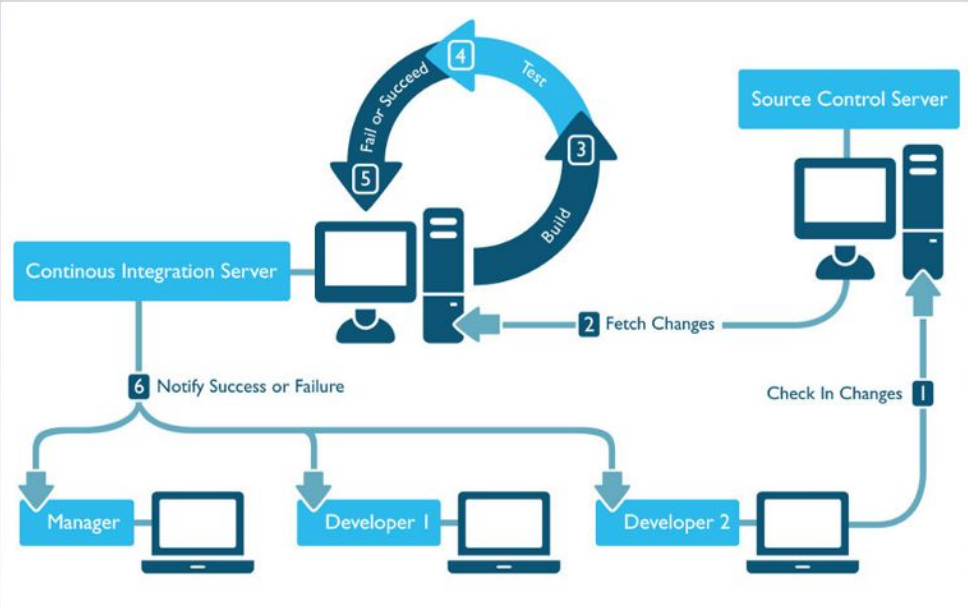
CI/CD

Continuous Integration Continuous Delivery

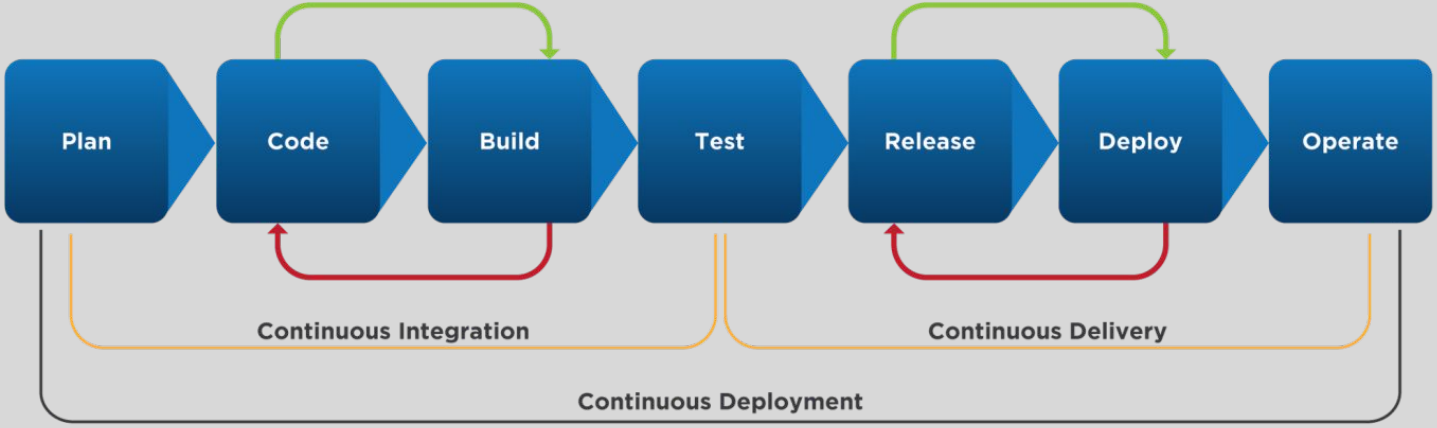
What is Continuous Integration

Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. It's a primary **DevOps best practice**, allowing developers to frequently merge code changes into a central repository where builds and tests then run. Automated tools are used to assert the new code's correctness before integration. It's all about code and "Dev" workflow



What is Continuous Deployment

Continuous deployment is a strategy in software development where code changes to an application are released automatically into the production environment. It's all about deployments and “Ops” workflow



Planning
- Requirement finalization
- Updates & new changes
- Architecture & design
- Task assignment
- Timeline finalization

Code
- Development
- Configuration finalization
- Check-in source code
- Static-code analysis
- Automated review & peer review

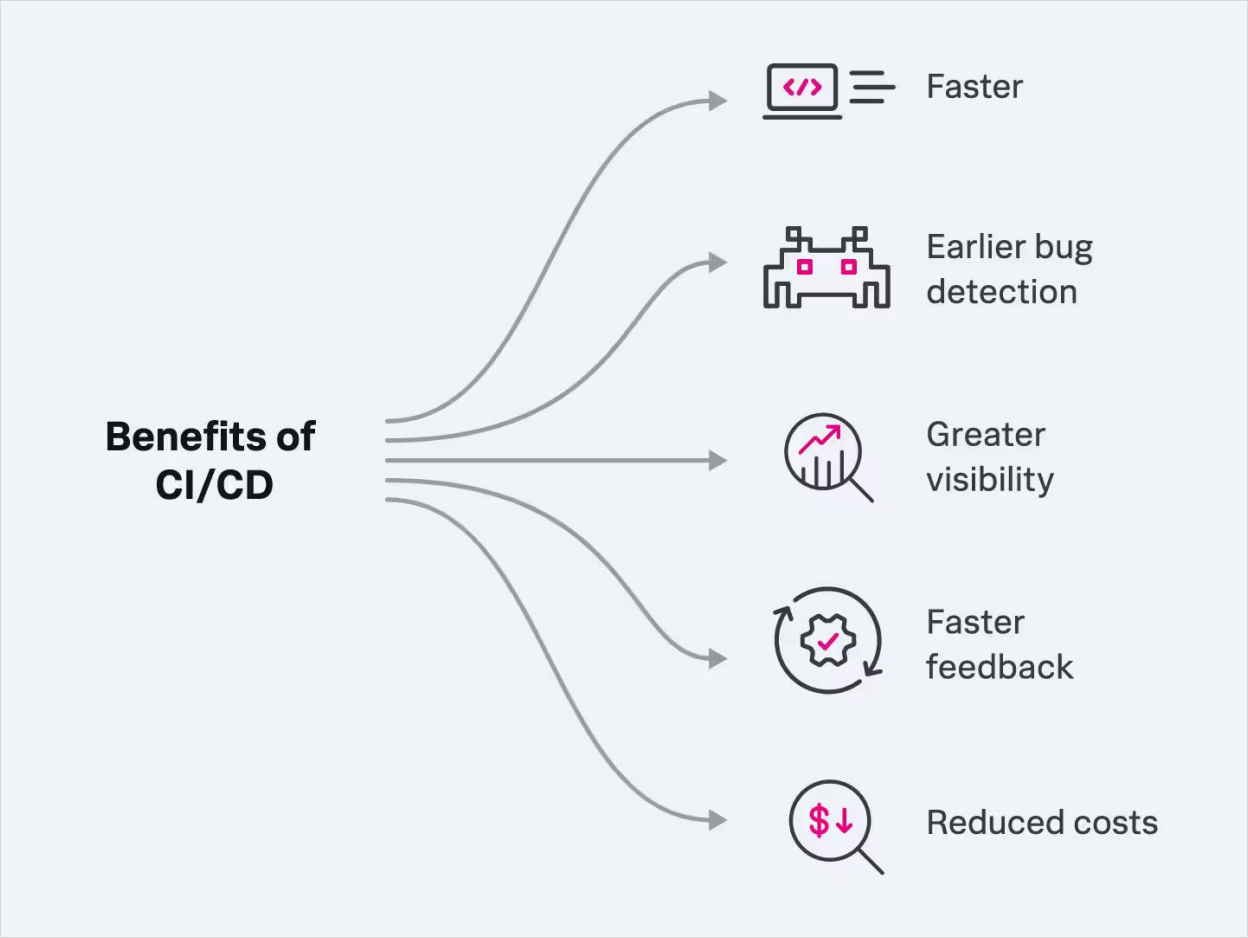
Build
- Compile code
- Unit testing
- Code-metrics
- Build container images or package
- Preparation or update in deployment templated
- Create or update monitor dashboards

Test
- Integration test with other component
- Load & stress test
- UI testing
- Penetration testing
- Requirement testing

Release
- Preparing release notes
- Version tagging
- Code freeze
- Feature freeze

Deploy
- Updating the infrastructure i.e staging, production
- Verification on deployment i.e smoke tests

Operate
- Monitor designed dashboard
- Alarm triggers
- Automatic critical events handler
- Monitor error logs


Benefits of CI/CD
Faster
Earlier bug detection
Greater visibility
Faster feedback
Reduced costs

Faster and more robust releases: With the automation of most testing and bug fixing, developers can increasingly focus on writing code and improving the quality of the application.

Earlier bug detection: Developers can identify bugs and other problems as they arise — with the majority detected in early stages — eliminating potentially devastating surprises during deployment.

Greater visibility: A CI/CD pipeline allows development teams to analyze builds, as well as test results and coding errors with great detail. This gives them a better understanding of what build changes led to problems and inefficiencies and prepares them to avoid these issues in the future.

Faster feedback loops: CI/CD sets up a constant user feedback loop that developers can use to their advantage to experiment with features and improve the app experience.

Reduced costs: Developers no longer have to spend time on mundane, manual tasks, which allows them to focus on improving code quality and making app improvements that boost ROI.

Increased customer satisfaction: More reliable releases and faster turnarounds for updates, bug fixes and new features incentivize customers to choose you over the competition.