CI/CD Pipeline

07 April 2025 16:11

How CI/CD works

1. **Continuous integration (CI)**: Developers frequently merge code changes into a central repository
2. **Continuous delivery (CD)**: Code changes are tested and deployed fully automatically

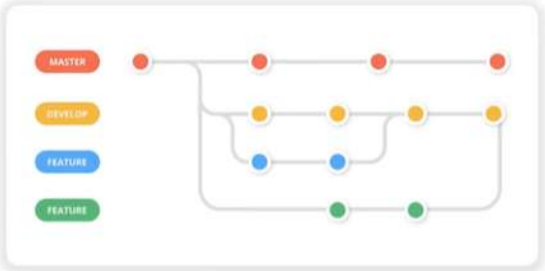
CI/CD can be implemented using tools like: Jenkins, Jira, and GitHub Actions.

CI/CD pipelines can help prevent errors that result from manual processes. They can also allow for rapid product iterations and provide consistent feedback during the development process.

Continuous Integration

Continuous Integration (CI):

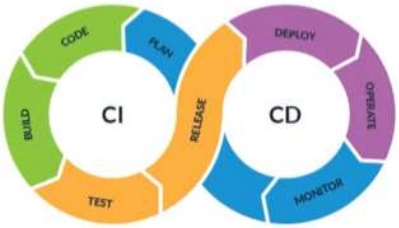
- **Definition:** CI is a development practice where developers integrate code changes frequently into a shared repository.
- **Goal:** To detect errors quickly, ensuring that code changes work well with existing code.
- **Key Benefits:**
  - Reduces integration issues by catching bugs early.
  - Encourages frequent and smaller commits, making code management easier.
  - Allows teams to deliver code updates more reliably.



Continuous Deployment

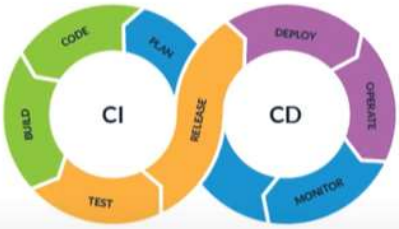
Continuous Deployment (CD):

- **Definition:** CD automates the release of software updates from a repository to production environments.
- **Goal:** To provide a streamlined and automated path from development to deployment.
- **Key Benefits:**
  - Ensures a faster and more reliable release process.
  - Reduces the time to market for new features.
  - Minimizes manual errors and enhances the stability of releases.



CI / CD Workflow

1. **Code Commit:** Developers commit code changes to a shared repository (e.g., GitHub).
2. **Build Process:**
   - Automated builds are triggered to compile the code and ensure it's error-free.
   - Unit tests and static code analysis are performed to maintain code quality.
3. **Testing:**
   - Automated tests, such as integration and end-to-end tests, are run to verify functionality.
   - If tests fail, notifications are sent to developers to fix issues before progressing further.
4. **Deployment:**
   - Approved builds are automatically deployed to staging or production environments.
   - Monitoring systems ensure the deployed changes work as expected.
5. **Feedback & Monitoring:**
   - Real-time feedback from users and monitoring tools helps identify issues.
   - Rollback mechanisms enable a swift response to any deployment problems.



CI / CD Tools

GitHub Actions

Jenkins

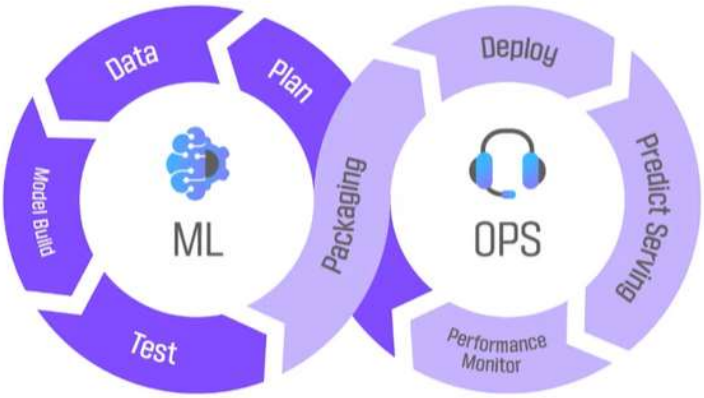
Bitbucket Pipelines

AWS CodePipeline

Azure DevOps

GITLAB CI/CD

CI / CD Workflow



Practical steps to perform CI/CD

- Create a repository on Github
- Create a new python project with new virtual env
- Activate the virtual environment .venv/Scripts/activate
- Clone the repository using git clone Link

- Enter into that git cloned project
- Create src dir for main python file
- Create tests file for creating test cases
- Create requirements.txt to store libraries
- Create a blank Conftest.py file (No use just to make sure that the code runs)
- Then run the test cases using Pytest/tests
- Then git add . ----> to send file in the dir
- Git status to check the files
- git commit -m "base code"
- git push origin main
- Go onto Github and click on actions
- Choose python function
- Create the folder in pycharm .github workflows and the yml and paste the code yml code in pycharm and also paste the deploy code
- Then again repeat the git actions git add, status, commit and git push origin main
- Then it'll check for the test cases and then deploy