CI/CD Pipelines with Jenkins, GitLab, and GitHub Actions

Modern software development practices prioritize speed, reliability, and consistency. Continuous Integration (CI) and Continuous Deployment/Delivery (CD) help teams automate testing, building, and deployment of applications. Let’s explore how Jenkins, GitLab CI/CD, and GitHub Actions enable this automation.

⚙️ Jenkins

Overview:

- **Language:** Java-based
- **Setup:** Self-hosted, requires configuration
- **Flexibility:** Highly customizable with plugins

Example Pipeline (Jenkinsfile):

```
pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps {
        git 'https://github.com/user/repo.git'
      }
    }
    stage('Build') {
      steps {
        sh './build.sh'
      }
    }
    stage('Test') {
```

```
      steps {
        sh './run-tests.sh'
      }
    }
    stage('Deploy') {
      steps {
        sh './deploy.sh'
      }
    }
  }
}
```

Pros:

- Highly extensible (1500+ plugins)
- Mature and widely adopted

Cons:

- Complex setup and maintenance
- UI/UX not as modern

⚡ GitHub Actions

Overview:

- Integrated into **GitHub**
- Define workflows using YAML in `.github/workflows/`

Example Workflow:

```
name: CI Pipeline

on:
  push:
    branches: [main]

jobs:
```

```
  build-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up Node.js
        uses: actions/setup-node@v3
        with:
          node-version: 18
      - run: npm install
      - run: npm test
```

Pros:

- Native GitHub integration
- Marketplace for reusable actions
- Easier setup for open-source projects

Cons:

- Limited flexibility for complex pipelines
- Workflow logs could be more user-friendly

✅ Best Practices for CI/CD

- **Fail fast:** Run tests early in the pipeline
- **Parallelization:** Split tests/builds across jobs
- **Artifacts:** Use caching and artifacts between jobs
- **Secrets Management:** Use environment variables or secret stores
- **Rollback Plans:** Automate rollback on failed deployments

---

🔚 Conclusion

Whether you're a team looking for a fully integrated DevOps platform (GitLab), already invested in GitHub (GitHub Actions), or need a highly customizable solution (Jenkins), choosing the right CI/CD tool

hinges on your development workflow, hosting preference, and integration needs.