Setting up CI/CD Pipeline

Process Overview

CI/CD (Continuous Integration/Continuous Delivery) is the practice of automating application building, testing, and deployment. Let's look at the step-by-step setup.

What We're Setting Up

1. **Automation on Repository Push:**
    - Running tests
    - Code quality checks
    - Application build
    - Docker image creation
    - Server deployment
2. **Where We Configure:**
    - GitHub Actions for CI/CD
    - Docker Hub for image storage
    - Your server for deployment
3. **How It Works:**
    - Push code → GitHub triggers Actions
    - Actions runs all checks
    - On success, code deploys to server

Configuration Files

1. Version Control (GitHub)

```
# .github/workflows/main.yml
name: CI/CD Pipeline
on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]
```

2. Build and Test

```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'
      - run: npm ci
      - run: npm test
      - run: npm run build
```

3. Code Quality

```
      - name: Lint Check
        run: npm run lint
      - name: Type Check
        run: npm run type-check
      - name: Security Scan
        uses: snyk/actions/node@master
```

4. Docker Build

```
      - name: Build Docker
        run: |
          docker build -t myapp:${{ github.sha }} .
          docker tag myapp:${{ github.sha }} myapp:latest
```

5. Deploy

```
      - name: Deploy
        if: github.ref == 'refs/heads/main'
        run: |
          echo ${{ secrets.DOCKER_PASSWORD }} | docker login -u ${{
secrets.DOCKER_USERNAME }} --password-stdin
          docker push myapp:latest
```

Best Practices

- Branch Protection
- Environment Secrets
- Automated Testing
- Semantic Versioning
- Rollback Strategy

Benefits

- Automated Deployments
- Consistent Quality
- Faster Releases
- Reduced Human Error

Tips

1. Start Small

2. Automate Everything Possible
3. Monitor Pipeline Performance
4. Keep Security in Mind
5. Document Thoroughly

Tools Used

- GitHub Actions
- Docker
- Node.js
- Jest
- ESLint
- TypeScript