Отлично! Рассмотрим настройку CI/CD для Java-проекта с использованием Gradle и GitHub Actions.

Шаг 1: Создание workflow файла

Создадим файл .github/workflows/build-and-deploy.yaml в вашем репозитории:

```
name: CI/CD Pipeline for Java Project

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'adoptium'
      - name: Validate Gradle wrapper
        run: gradle wrapper --gradle-version 7.5
      - name: Build with Gradle
        run: ./gradlew clean build
      - name: Test with Gradle
        run: ./gradlew test
      - name: Upload artifacts
        uses: actions/upload-artifact@v3
        with:
          name: build-output
          path: build/libs/*.jar

  deploy:
    needs: build
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main'
    steps:
      - uses: actions/checkout@v3
      - name: Download artifacts
        uses: actions/download-artifact@v3
        with:
          name: build-output
      - name: Setup SSH key
        env:
          SSH_PRIVATE_KEY: ${{ secrets.SSH_PRIVATE_KEY }}
          SSH_KNOWN_HOSTS: ${{ secrets.SSH_KNOWN_HOSTS }}
        run: |
```

```
          mkdir -p ~/.ssh/
          echo "$SSH_PRIVATE_KEY" > ~/.ssh/id_rsa
          chmod 600 ~/.ssh/id_rsa
          echo "$SSH_KNOWN_HOSTS" > ~/.ssh/known_hosts
          chmod 644 ~/.ssh/known_hosts
      - name: Deploy to Server
        run: |
          scp -o StrictHostKeyChecking=no build/libs/*.jar
user@server:/path/to/project/
          ssh user@server "cd /path/to/project && java -jar your-app.jar &"
```

Описание шагов:

1. **Build job**:
   - Использует последнюю версию Ubuntu.
   - Проверяет исходный код из репозитория.
   - Устанавливает JDK 17.
   - Валидирует Gradle wrapper и устанавливает нужную версию Gradle.
   - Компилирует и собирает проект с помощью ./gradlew clean build.
   - Запускает тесты с помощью ./gradlew test.
   - Загружает собранные артефакты (в данном случае JAR-файл) в хранилище артефактов GitHub Actions.
2. **Deploy job**:
   - Выполняется только после успешного завершения build job.
   - Проверяет, что текущая ветвь — main, чтобы избежать ненужных развертываний.
   - Скачивает загруженный ранее артефакт.
   - Настраивает SSH-ключи для безопасного доступа к серверу.
   - Копирует собранный JAR-файл на сервер с помощью scp.
   - Запускает приложение на сервере с помощью команды java -jar your-app.jar &.

Шаг 2: Добавление секретов

Как и раньше, добавьте секреты SSH_PRIVATE_KEY и SSH_KNOWN_HOSTS в репозиторий.

Шаг 3: Тестирование и проверка

Теперь каждый раз, когда вы делаете коммит или пул-реквест в ветку main, GitHub Actions запустит вашу CI/CD пайплайн. Вы сможете следить за прогрессом выполнения в разделе Actions вашего репозитория.

Если все настроено правильно, после успешного завершения сборки и тестирования, изменения будут автоматически развернуты на вашем сервере.

Дополнительные возможности

Вы также можете добавить другие этапы, такие как статический анализ кода, линтеры, сканеры безопасности и многое другое. GitHub Actions предлагает огромное количество готовых действий, которые помогут расширить функциональность вашей пайплайна.

Надеюсь, этот гайд помог вам начать работу с CI/CD для Java-проекта с использованием Gradle и GitHub Actions!