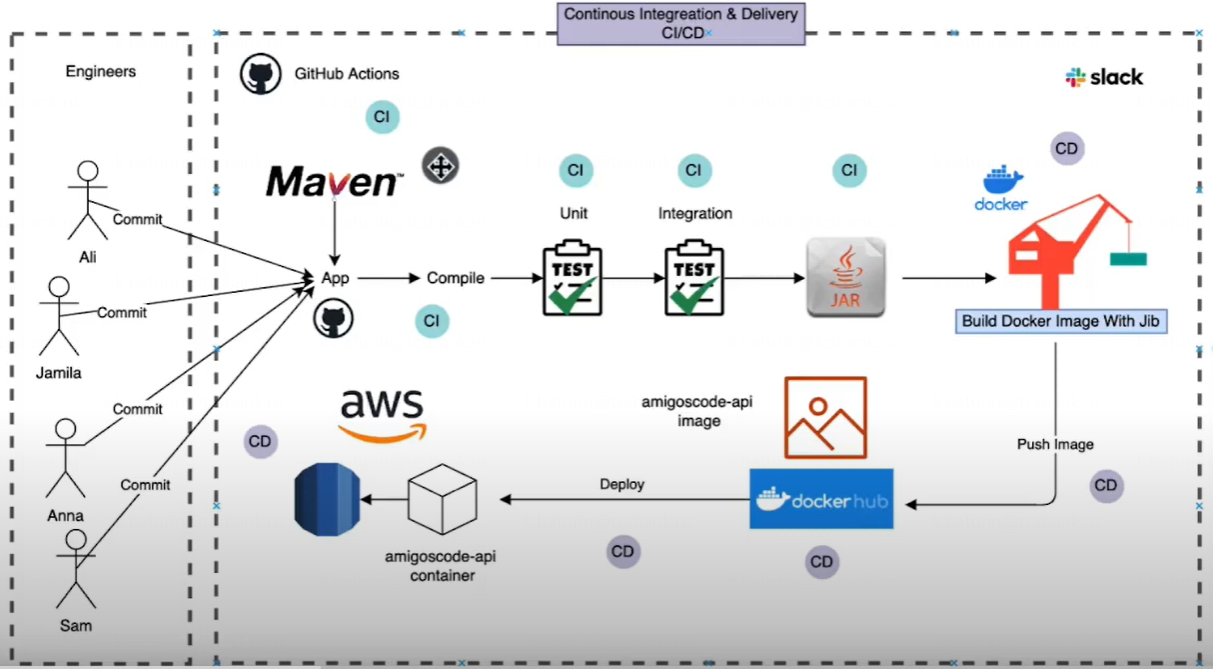
***Continuous Integration & Continuous Delivery*** – это способ интеграции изменений кода и доставки его на прод. Он позволяет автоматизировать процесс обновления, тестирования кода и деплоя новой версии приложения.



Вручную запаковать *jar*-файл можно командой ***mvn package***

Вручную запустить jar-файл можно командой ***java -jar .\target\spring-boot-example-0.0.1-SNAPSHOT.jar***

Остановить работу jar-файла можно комбинацией ***Ctrl+C***

Для того чтобы автоматически скомпилировать, протестировать и собрать проект в jar-файл, в *pom*-файл проекта нужно добавить ***spring-boot-maven-plugin***. Все эти команды можно запустить через готовые скрипты *Maven* в *Idea*.

*<build>*

*<plugins>*

*<plugin>*

*<groupId>org.springframework.boot</groupId>*

*<artifactId>spring-boot-maven-plugin</artifactId>*

*</plugin>*

*</plugins>*

*</build>*

В качестве инструмента для сборки и загрузки во вне докер-образа можно использовать либо [*dokerfile* + скрипт *maven*](https://docs.spring.io/spring-boot/docs/current/maven-plugin/reference/htmlsingle/#build-image.build-image-goal) на сборку образа, либо плагин [jib](https://github.com/GoogleContainerTools/jib) от google.

Пример *jib*-плагина в *pom*-файле:

*<build>*

*<plugins>*

*<plugin>*

*<groupId>com.google.cloud.tools</groupId>*

*<artifactId>jib-maven-plugin</artifactId>*

*<version>3.3.1</version>*

*<configuration>*

*<from>*

*<image>eclipse-temurin:17</image> // версия java sdk*

*<platforms>*

*<platform> // тип ядра и ОС*

*<architecture>arm64</architecture>*

*<os>linux</os>*

*</platform>*

*<platform>*

*<architecture>amd64</architecture>*

*<os>linux</os>*

*</platform>*

*</platforms>*

*</from>*

*<to> // загрузка в dockerHub (название докер-образа и теги)*

*<image>docker.io/${docker.username}/${project.artifactId}:${project.version}</image>*

*<tags>*

*<tag>latest</tag>*

*</tags>*

*</to>*

*</configuration>*

*</plugin>*

*</plugins>*

*</build>*

Для сборки проекта и загрузки образа в *dockerhub* нужно ввести команду:

*mvn clean package jib:build*

Для построения *CI/CD* можно также использовать ***GitActions***. Большинство повторяющихся действий уже описано другими пользователями. Их можно найти на <https://github.com/marketplace> .

Рассмотрим пример создания простого *CI* процесса. Для этого в корне проекта создадим папку *.github\workflows* и новый файл-*workflow* с именем ***backend-ci.yml*** :

*name: CI - Build Backend*

*on:*

*pull\_request: // workflow срабатывает при pull\_request*

*branches:*

*- main // в ветку main*

*paths:*

*- backend/\*\* // с обновлением любых файлов по этому пути*

*jobs:*

*build-container-job: // имя job может быть любым*

*runs-on: ubuntu-latest // работа будет выполняться на ВМ с ubuntu*

*services:*

*postgres: // создаем и настраиваем сервис с тестовой БД*

*image: postgres:14.1*

*env:*

*POSTGRES\_USER: straigt*

*POSTGRES\_PASSWORD: password*

*POSTGRES\_DB: customer*

*ports:*

*- 5332:5432*

*options: >- // проверка работоспособности тестовой БД*

*--health-cmd pg\_isready*

*--health-interval 10s*

*--health-timeout 5s*

*--health-retries 5*

*defaults: // указываем папку с проектом*

*run:*

*working-directory: ./backend*

*steps: // список шагов*

*- uses: actions/checkout@v3 // проверка, получение доступа и клонирование репозитория*

*- uses: actions/setup-java@v3 // установка JDK temurin 17 с кешированием зависимостей*

*with:*

*distribution: 'temurin'*

*java-version: '17'*

*cache: 'maven'*

*- name: Build and run Unit/Integration Tests with Maven*

*run: mvn -ntp -B verify // запуск скрипта maven на проверку (без логов зависимостей)*