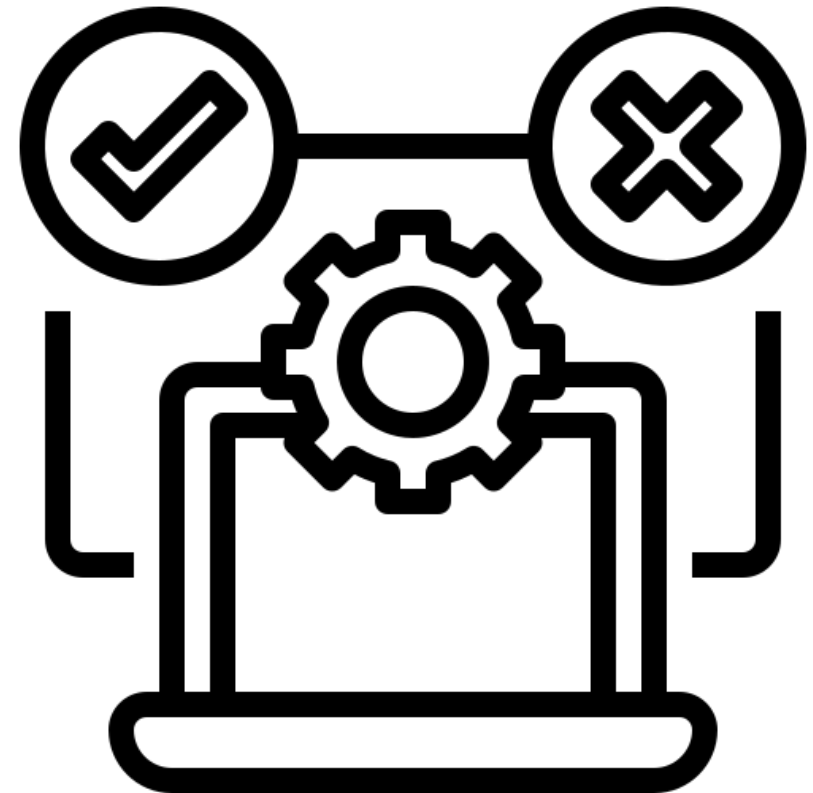


# АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ

## Лекция 6: Пайплайны

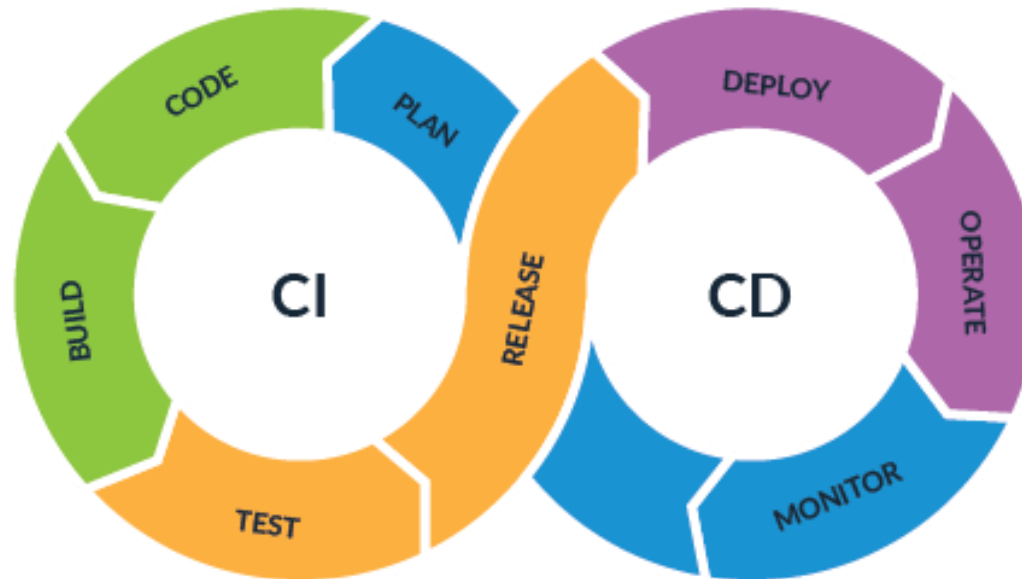
БАШАРИНА ЕКАТЕРИНА АЛЕКСАНДРОВНА



# CI/CD

- *Continuous integration / continuous deliver*

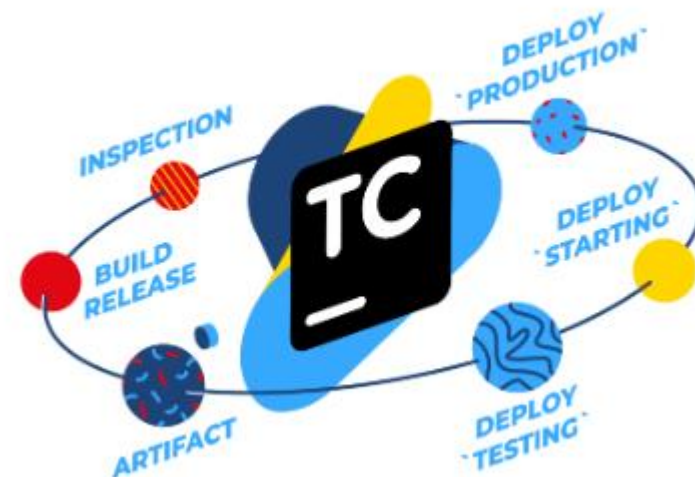
Частое внесение  
небольших  
изменений в код  
проекта,  
постоянное  
тестирование по  
ходу дела



Частый  
автоматизированный  
релиз новых  
стабильных версий  
приложения с  
небольшими  
изменениями

# CI/CD инструменты

- Jenkins job builder
- Gitlab CI/CD
- GitHub Pull Request buider
- CI/CD TeamCity
- И другие

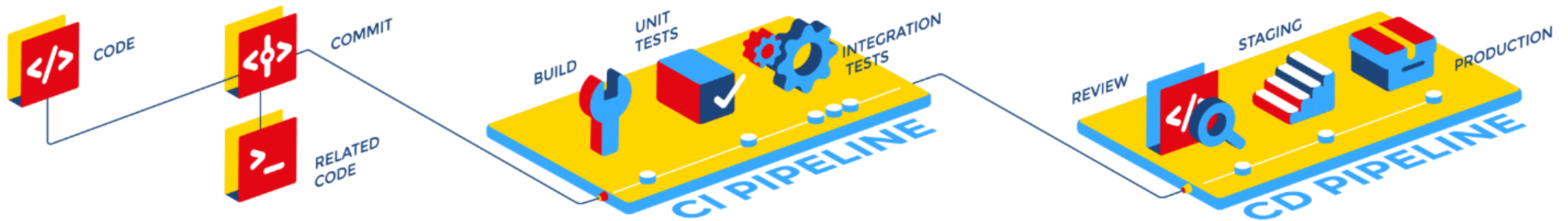


# Основная идея

- **Стратегическая задача CI/CD — поставить процесс тестирования и запуска ПО на регулярную основу, вплоть до нескольких раз в день.**
- **Поэтому необходимо максимально автоматизировать процесс, чтобы при завершении каждого этапа автоматически запускался следующий или выдавалось уведомление об ошибке.**

# Пайплайн (pipeline)

- концепция автоматизации процесса сборки, тестирования и развертывания программного обеспечения*



# Из чего состоит пайплайн?

- Конвейер, который состоит из нескольких этапов (stage) и задач (step), выполняемых последовательно
- Этапы могут включать в себя сборку кода, тестирование, анализ, публикация артефактов и т.д.

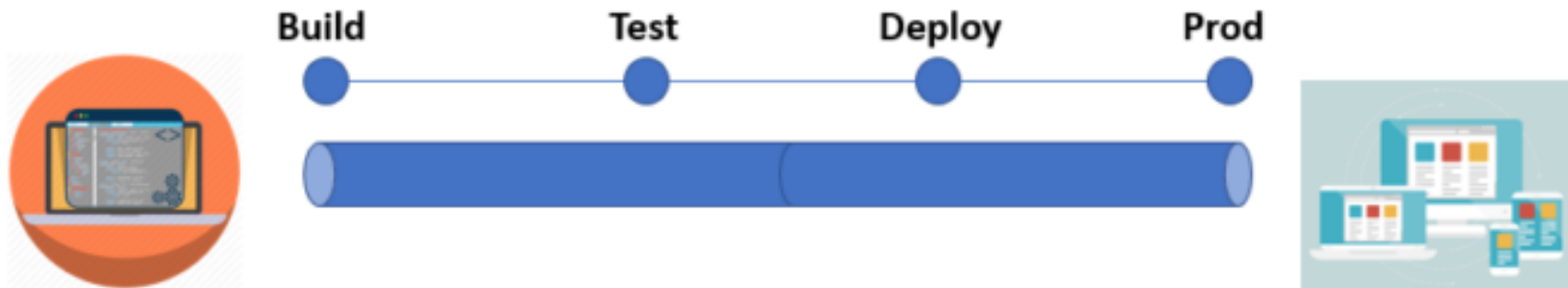


# Этапы пайплайна

- Коммит в ветку master или любую другую ветку
- Сборка или первый набор юнит-тестов
- Выполнение автоматических тестов
- Развертывание ПО в тестовых средах
- Внесение изменений в режиме реального времени

# Пайплайн в Jenkins

- Набор плагинов, позволяющий определить жизненный цикл сборки и доставки приложения как код
- Представляет из себя Groovy-скрипт





# Groovy

- ООП-язык, разработанный для платформы Java как дополнение к языку Java
- Использует Java-подобный синтаксис с динамической компиляцией в JVM-байткод
- Абсолютно всё в Groovy является объектами! (даже примитивные типы)



# Groovy как DSL (Domain-specific language)

- **Гроову может быть использован как скриптовый язык для написания сценариев (наподобие bat или sh)**
- **Хорош тем, что легко интегрируется с Java**


```
println "Groovy script"  
multi = {  
    num1, num2 -> num1*num2  
}  
multi(4,4)
```

# Настройка пайплайна в Jenkins


- Создаём New item -> Pipeline

**Enter an item name**

*» Required field*

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

# Настройка пайплайна в Jenkins

- В разделе Pipeline пишем скрипт на Groovy, описывающий необходимую последовательность шагов
- Для начала можно использовать sample Pipelines



The screenshot shows the Jenkins Pipeline configuration page. At the top, there's a 'Pipeline' header. Below it, a 'Definition' section contains a text area labeled 'Pipeline script'. Underneath, a 'Script ?' section displays a sample Groovy script for a pipeline. The script defines a pipeline with an 'any' agent, a 'Hello' stage, and an 'echo' step.

```
1 pipeline {  
2   agent any  
3  
4   stages {  
5     stage('Hello') {  
6       steps {  
7         echo 'Hello World'  
8       }  
9     }  
10  }  
11 }  
12
```

# Настройка пайплайна в Jenkins

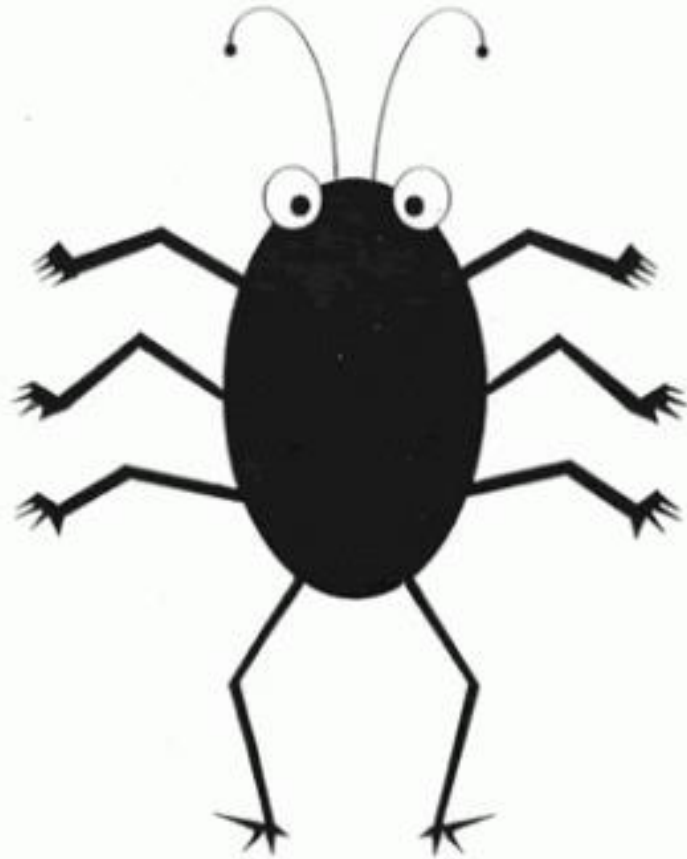
- Можно настроить дополнительные параметры – например, шаги или триггеры сборки
- Запуск пайплайна происходит так же, как и запуск джобы

- ☐ Pipeline speed/durability override ?
- ☐ Preserve stashes from completed builds ?
- ☐ This project is parameterized ?
- ☐ Throttle builds ?

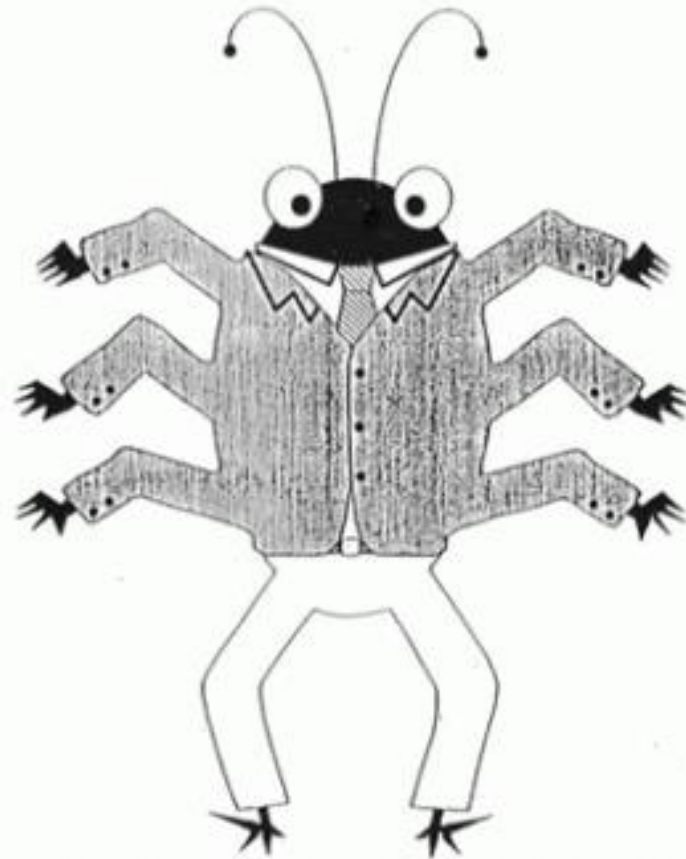
## Build Triggers

- ☐ GitHub hook trigger for GITScm polling ?
- ☐ Build periodically ?
- ☐ Build after other projects are built ?
- ☐ Poll SCM ?

# Спасибо за внимание!



**BUG**



**FEATURE**