**Homework\_Lesson\_28**

"Цель: научиться работать с Jenkins Pipeline, Groovy, а также автоматизировать сборку, тестирование и развертывание приложения.

Основные шаги:

Создать Jenkins Pipeline с использованием декларативного синтаксиса.

Определить агент,который будет выполнять сборку

Добавить этапы (stages):

Build – сборка приложения с помощью Maven

Test – запуск тестов.

Deploy – развертывание приложения.

Определить шаги (steps) – команды, которые будут выполняться на каждом этапе (например, запуск команд в терминале sh, клонирование репозитория git).

Добавить условия выполнения (when) – запуск определенных этапов в зависимости от параметров (например, запуск только при изменениях в коде).

Использовать параметры (parameters) – например, выбор версии приложения или окружения.

Предусмотреть проверку на ошибки, чтобы в случае проблем процесс не зависал.

Задокументировать Pipeline – описать структуру, шаги и параметры в файле README.

**Для начала определим агент для выполнения всех этапов внутри Docker-контейнера.   
Буду использовать образ Maven для сборки приложения.**

Дополнительное задание (необязательно):

Написать Groovy-скрипт, который:

Клонирует репозиторий с веб-приложением.

Собирает приложение (Gradle/Maven).

Создает Docker-образ.

Запускает контейнер с приложением.

Проверяет, что приложение работает.

Автоматизация отчетов:

Создать DSL (мини-язык) для генерации отчетов.

Автоматизировать процесс сбора данных и формирование отчета.

Добавить обработку ошибок (например, если данных нет).

pipeline {

agent any

stages {

stage('Clone repository') {

steps {

git 'https://github.com/username/repo.git'

}

}

stage('Build application') {

steps {

sh 'gradle build' // или 'mvn clean install'

}

}

stage('Create Docker image') {

steps {

sh 'docker build -t my-app .'

}

}

stage('Deploy Docker container') {

steps {

sh 'docker run -d -p 8080:8080 --name my-app-container my-app'

}

}

stage('Check application') {

steps {

sh 'curl http://localhost:8080'

}

}

}

}

"