### ****Тема 1: Введение в CI/CD****

#### Урок 1: Что такое CI и CD

**Теория:**

**CI (Continuous Integration)** — процесс автоматического объединения кода разработчиков в одну ветку с постоянной проверкой.

**CD (Continuous Delivery / Deployment)** — автоматическая доставка кода до сервера или окружения (staging/production).

**Преимущества:**

* Быстрая проверка изменений
* Минимизация багов
* Автоматизация сборки и деплоя

**Цель:** как можно быстрее и безопаснее доставлять изменения пользователям.

**Тест:**

1. Что означает CI?  
   **Ответ:** continuous integration
2. Что автоматизирует CD?  
   **Ответ:** деплой

#### Урок 2: Типичный CI/CD-процесс

**Теория:**

Типовой CI/CD процесс:

1. Разработчик делает **push** в Git.
2. Тесты запускаются автоматически (CI).
3. Если всё успешно, код проходит **сборку** и **деплой** (CD).
4. Уведомления приходят в Slack/Telegram.

**Пример пайплайна:**

* checkout → build → test → deploy

**Инструменты:** GitHub Actions, GitLab CI, Jenkins, CircleCI и др.

**Тест:**

1. Что происходит первым в CI-процессе?  
   **Ответ:** push
2. Назовите одну популярную CI/CD платформу:  
   **Ответ:** github actions
3. Какой этап обычно идёт после тестов?  
   **Ответ:** деплой

### ****Тема 2: Git и GitHub как основа CI****

#### 🔹 Урок 1: Основы Git и команды для CI

**Теория:**

Git — система контроля версий. Основные команды:

git init # инициализация

git add . # добавить файлы

git commit -m "msg" # зафиксировать

git push # отправить

В CI важно:

* Работать с **ветками**
* Использовать **pull requests**
* Поддерживать **чистую историю коммитов**

**Тест:**

1. Команда для добавления всех файлов:  
   **Ответ:** git add .
2. Как называется запрос на слияние кода?  
   **Ответ:** pull request
3. Что такое Git?  
   **Ответ:** cистема контроля версий

#### 🔹 Урок 2: Работа с GitHub репозиториями

**Теория:**

**GitHub** — хостинг для Git-репозиториев с поддержкой CI/CD.

* Репозиторий — проект с кодом.
* .github/workflows — место для CI-скриптов.
* Actions — единицы CI/CD, выполняющие действия по триггеру.

Можно:

* Запустить тесты при push
* Задеплоить на сервер
* Прислать уведомление

**Тест:**

1. Где хранятся workflows в GitHub?  
   **Ответ:** .github/workflows
2. Что такое Action в GitHub CI/CD?  
   **Ответ:** Операция/действие
3. Какой файл определяет CI процесс?  
   **Ответ:** yaml-файл

### ****Тема 3: GitHub Actions / GitLab CI****

#### 🔹 Урок 1: Основы GitHub Actions

**Теория:**

**GitHub Actions** — встроенный CI/CD-инструмент GitHub, позволяющий запускать автоматические действия по событиям в репозитории.

**Структура workflow:**

yaml

КопироватьРедактировать

name: CI Pipeline

on: [push]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: Установка Node.js

uses: actions/setup-node@v3

with:

node-version: '18'

- name: Установка зависимостей

run: npm install

- name: Запуск тестов

run: npm test

**Ключевые элементы:**

* on — триггер (push, pull\_request и др.)
* jobs — список задач
* steps — действия внутри задачи

**Тест:**

1. Какой файл описывает GitHub Actions workflow?  
   **Ответ:** YAML
2. Какой ключ указывает, когда запускается workflow?  
   **Ответ:** on
3. Где обычно размещается workflow?  
   **Ответ:** .github/workflows

#### 🔹 Урок 2: Основы GitLab CI

**Теория:**

**GitLab CI/CD** — мощный инструмент, встроенный в GitLab, для автоматизации процессов сборки, тестирования и деплоя.

Файл .gitlab-ci.yml описывает pipeline.

**Пример:**

stages:

- build

- test

- deploy

build\_job:

stage: build

script:

- npm install

test\_job:

stage: test

script:

- npm test

deploy\_job:

stage: deploy

script:

- echo "Deploying..."

**Элементы:**

* stages — этапы пайплайна
* script — команды для выполнения
* before\_script, after\_script — общие команды

**Тест:**

1. Как называется файл конфигурации GitLab CI?  
   **Ответ:** .gitlab-ci.yml
2. Назовите этапы пайплайна (ключ):  
   **Ответ:** stages
3. Где указывается код, который должен выполняться в GitLab CI?  
   **Ответ:** script

### ****Тема 4: Docker и автоматизация в CI/CD****

#### 🔹 Урок 1: Docker как часть CI

**Теория:**

Docker — ключевой инструмент в CI/CD. Позволяет:

* Упаковать приложение с зависимостями в **контейнер**
* Гарантировать, что оно будет одинаково работать везде

**Dockerfile** описывает, как собрать контейнер.

Пример:

FROM node:18

WORKDIR /app

COPY . .

RUN npm install

CMD ["npm", "start"]

Контейнер используется:

* В GitHub Actions
* В GitLab CI
* В Jenkins

**Тест:**

1. Что описывает Dockerfile?  
   **Ответ:** Как собрать контейнер
2. Какой командой запускают сборку контейнера?  
   **Ответ:** docker build
3. Почему контейнеры удобны для CI?  
   **Ответ:** Идентичность окружения

#### 🔹 Урок 2: Автоматизация сборки и деплоя Docker-контейнеров

**Теория:**

В CI можно автоматически:

1. Собрать контейнер:

docker build -t myapp .

1. Отправить в Docker Hub:

docker tag myapp username/myapp:latest

docker push username/myapp:latest

1. Задеплоить на сервер:

docker run -d -p 80:3000 username/myapp

Все эти шаги можно автоматизировать в GitHub Actions или GitLab CI.

**Тест:**

1. Команда для отправки контейнера в Docker Hub:  
   **Ответ:** docker push
2. Какой CI инструмент может собирать и пушить Docker?  
   **Ответ:** GitHub Actions
3. Какой порт обычно используется для HTTP?  
   **Ответ:** 80