Задание #1

- Зайти на Gitlab кафедры ВС. Далее будет подразумеваться именно этот Gitlab – https://git.csc.sibsutis.ru
- Создать репозиторий. Дальнейшая разработка будет вестись в этом репозитории.
- Написать простое приложение, которое запускает http-сервер и отдает контент. Контент может быть любым – строка, текст, случайное число и тп. Язык программирования – любой.
- Написать Dockerfile для данного приложения. Результат сборки – docker image, при запуске docker-контейнера из которого в контейнере запускается разработанное приложение. Файл разметить вместе с проектом в репозитории.

Задание #2

- Доработать Dockerfile – Для компилируемых языков сделать multistage сборку. Этап 1 – компиляция/сборка; Этап 2 – копирование артефакта компиляции и запуск приложения. Проанализировать как изменится размер итогового образа.
- Написать конфигурацию Gitlab CI со следующим функционалом
    a. Сборка образа (Build, tag)
    b. Загрузка образа в Gitlab Registry (push)
    c. Удаление образа с Gitlab Runner-а
- Запуск CI должен выполняться автоматически при установке тега для коммита. Так же для любого коммита должна быть возможность запуска «по кнопке» (manual). Тегом получившегося docker-образа является хэш коммита или git tag
- Добавить в проект .dockerignore файл, включив в него список файлов, не требуемых для сборки проекта (Синтаксис аналогичен файлу .gitignore). Проанализировать как изменится размер итогового образа и время сборки (Для измерения времени можно создать временный большой файл и добавить/убрать его из .dockerignore)

Задание #3

- Разработать docker-compose файл со следующим функционалом:
    a. Запуск образа приложения, написанного в задании #2
    b. Запуск готового docker-приложения с веб-сервером (например – traefik/whoami)
    c. Запуск основного прокси-сервера по варианту. Прокси-сервер должен реализовать отдачу контента двух приложений при обращении по домену/IP адресу на порт 80 хоста. Домен – любой. При обращении на /app запрос должен проксироваться приложению из пункта b. На остальные запросы должно отвечать приложение из пункта a. Варианты основных прокси-серверов:
        1. Nginx
        2. Apache
        3. Traefik
- Docker-compose файл, дополнительные файлы конфигурации разместить так же в репозитории. Файлы конфигурации поместить в отдельной директории.

Задание #4

- Доработать Dockerfile таким образом, чтобы приложение (Разработанное в задании #1) запускалось от имени непривилегированного пользователя (Не root). Необходимо в процессе сборки:
    a. Создать пользователя.
    b. Используя инструкцию USER установить для запуска созданного пользователя
    c. Альтернатива: Используя entrypoint-скрипт, инструкцию ENTRYPOINT и gosu (https://github.com/tianon/gosu) при старте контейнера переключиться в контекст непривилегированного пользователя
- *Посмотреть и ответить, в чем различие в runtime при использовании инструкции USER и использовании gosu.

Задание #5

- Добавить в Docker-compose экземпляр базы данных по варианту
    a. Mariadb
    b. Postgresql
    c. Mongodb
- Доработать приложение для работы с базой данных.
    a. Необходимо разработать модель данных (Например, User с полями id, name ,email)
    b. При запуске контейнера БД должны создаваться необходимые для этой модели таблицы в базе данных.
    c. Реализовать в приложении набор следующих endpoint-ов (Хотя бы 2, на примере ./users)
        1. GET /users – вывести список всех пользователей из БД (можно реализовать с поддержкой параметров start-end или иным вариантом пагинации: /users?_start=0&_end=5)
        2. PUT /users – создать пользователя с переданными параметрами
        3. GET /users/{:id} – вывести данные пользователя с указанным id
        4. DELETE /users/{:id} – удалить пользователя с указанным id
    d. Модель данных, передаваемых для put,post,delete оформляется в json, возвращаемые данные также в json.
    e. Продемонстрировать взаимодействие с БД
- Параметры подключения к базе данных необходимо передавать приложению при запуске через ключи командной строки или переменные окружения (Environment)
- Дополнительные файлы конфигурации разместить в отдельной директории.

РГЗ

- Установить minikube (https://kubernetes.io/ru/docs/tasks/tools/install-minikube/)
- Запустить kubernetes на базе minikube

1. Для оценки 3
    a. Развернуть приложение в minikube. Приложение может быть как из задания #1, так и на выбор (Например traefik/whoami). Развертывание при помощи сущности kubernetes - Deployment. Продемонстрировать работу приложения в kubernetes.
2. Для оценки 4
    a. Пункт 1.
    b. Добавить Readiness/Liveness probe для приложения
    c. С помощью Helm/Helmfile установить веб-приложение в minikube. (Например echo-server https://ealenn.github.io/Echo-Server/)
3. Для оценки 5
    a. Пункт 2.
    b. Создать сущность kubernetes – Service, для обращения к приложению, развернутому в Deployment.
    c. Включить nginx-ingress-controller в minikube
    d. Создать сущность kubernetes – Ingress, разрешающую обращение по определенному домену к сервису, созданному ранее. Домен может быть любым
    e. Добавить домен для приложения из пункта 2.c при помощи helm