## Python+Docker

## Python+Docker: використання docker&docker-compose у python проектах.

|  |  |
| --- | --- |
| **Кількість уроків:** | 5 |
|  |  |
| **Тривалість у годинах:** | 10 годин |

**Опис курсу**

Курс “Python+Docker” буде корисним усім, хто має просунуті знання мови Python і бажає довести володіння ним до розвинутого рівня. Курс готує до подальшого освоєння складних тем, пов’язаних з багатосервісною архітектурою проектів і є необхідним для подальших курсів «Python+PostgreSQL» і може бути корисним при створенні власного курсового проекту і його розгортанні. Викладання курса побудовано на постійній комбінації теорії і практики: теоретичні матеріали отримані протягом заняття одразу використовуються на практиці під час створення інфраструктури і застосунка – телеграм-бота.

Під час вивчення курсу буде створений учбовий проект чат-боту на базі фреймворка aiogram, який буде побудований як багатосервісний застосунок із розгортанням кожного сервіса в окремому контейнері і налаштуванні взаємодії цих сервісів. В якості БД буде використовуватись postgrеSQL, протягом побудови застосунку студенти ознайомляться з основами роботи із змінними середовища, з структуруванням проекту на окремі модулі.

Отримані протягом курсу знання та навички є абсолютно практичними і відповідають поточним потребам ринку праці за фахом python backend developer\python software engeneer.

**План**

# Лекція 1 – створення базового проекту і встановлення необхідних сервісів.

1. Загальна постановка завдання на проект.
2. Реєстрація бота в Telegram і створення файла .env
3. Формування загальної структури проекту і розуміння складових частин.
4. Створення settings.py і знайомство з модулем python-dotenv.
5. Написання коду ехо-бота і знайомство з модулем aiogram.
6. Запуск першого варіанта бота і тестування його роботи.
7. Встановлення платформи docker і перевірка коректності встановлення.

# Лекція 2: знайомство з платформою docker.

1. Теорія: що таке віртуалізація\контейнерізація і для чого це використовується.
2. Віртуалізація: розуміння і використання.
3. Контейнерізація: розуміння і використання.
4. Що таке docker: загальне розуміння.
5. Архітектура docker.
6. Знайомство із складовими частинами платформи docker.
7. Як працю docker. Теорія.
8. Як створити docker-образ. Dockerfile
9. Структура Dockerfile і основні інструкції Dockerfile.
10. Пишемо Dockerfile для нашого проекта.
11. Домашнє завдання – робота з основними командами CLI Docker.

# Лекція 3: розвиток нашого проекту. Як працюють декілька контейнерів в одному проекті і для чого.

1. Розбираємо помилки і питання при роботі з CLI Docker.
2. Працюємо «в середині контейнера». Як зберігати секрети і конфидеційну інформацію.
3. Плануємо розвиток нашого проекту: додаємо роботу з БД postgreSQL. Модуль psycopg2.
4. Docker, теорія: аспекти сумісної роботи декількох контейнерів:
5. Співставлення портів: ports
6. Співставлення томів і створення постіних томів для зберігання даних: volumes
7. Docker networking: як спілкуються контейнери
8. Bridge\Host\None\власні мережі
9. DNS
10. Консолідуємо теорію для вирішення завдань розвитку проекта. DockerHub.
11. Docker compose: розуміння. CLI docker-compose.
12. Файл docker-compose.yaml – розуміння і загальна структура, основні інструкції.
13. Створюємо docker-compose.yaml для нашого проекту**.**

# Лекція 4: розвиток проекту і додаткові інструменти.

1. Розбираємо помилки і питання при роботі з CLI docker-compose і при створенні docker-compose.yaml файла.
2. Будуємо багатоконтейнерний проект. Основні дії – збірка, запуск, зупинка, видалення.
3. Розгортаємо БД postgreSQL в контейнері.
4. Планування найближчих цілей проекта: створення БД, створення таблиць, підключення до БД із основго застосунка
5. Створення БД, практична робота з CLI docker-compose.
6. Написання коду для роботи з БД.
7. Робота з CLI docker-compose.
8. Розгортання в додатковому контейнері інструментального ПЗ – pgAdmin:
9. Модифікація docker-compose.yaml для створення додаткового контейнера
10. Розуміння профілів запуску і створення dev-профіля.
11. Підключення pgAdmin до нашої БД, базові можливості.

# Лекція 5: створення підключення до БД і таблиць, модифікація бота для роботи з розгорнутою в контейнері БД

1. Планування структури даних під сформульовані раніше завдання для проекту. Структуризація коду для роботи з БД.
2. Створення з’єднання з БД – файл connection.py
3. Код для створення таблиці необхідної структури. Файл create\_table.py
4. Робота з CLI docker-compose для запуска скриптів і створення таблиць. Використання pgAdmin для контролю стану бази даних.
5. Модифікація коду бота для роботи з БД (echobot.py)
6. Робота з CLI docker-compose для перестворення образів, запуска контейнерів, роботи з логами контейнерів, зупинки\старта сервисів, видалення контейнерів, образів, томів даних.
7. Запуск створеного застосунка і тестування його роботи.
8. Висновки.