OTUS

Software Architect

Best Practice по разработке архитектуры программного обеспечения

Длительность курса: 124 академических часа

1 Инфраструктурные паттерны

1	**Плюсы и минусы микросервисной архитектуры.**	Плюсы и минусы монолитов. Плюсы и минусы микросервисной архитектуры. Основные паттерны микросервисной архитектуры.
2	**Основы работы с Docker.**	
3	**Инфраструктурные паттерны**	
4	**Основы работы с Kubernetes (часть 1)**	архитектура и базовые сущности Kubernetes: Pod, Deployment, ReplicaSet, StatefulSet, Service
5	**Основы работы с Kubernetes (часть 2)**	Архитектура и базовые сущности Kubernetes: ConfigMap, Persistent Volume, Persistent Volume Claim Домашние задания 1 Основы работы с Kubernetes (часть 2) Создать минимальный сервис, который 1) отвечает на порту 8000 2) имеет http-метод GET /health/ RESPONSE: {"status": "OK"}

Собрать локально образ приложения в докер.
Запушить образ в dockerhub

Написать манифесты для деплоя в k8s для этого сервиса.

Манифесты должны описывать сущности Deployment, Service, Ingress.
В Deployment обязательно должны быть указаны Liveness, Readiness пробы.
Количество реплик должно быть не меньше 2. Image контейнера должен быть указан с Dockerhub.

В Ingress-е должно быть правило, которое форвардит все запросы с /otusapp/{student name}/* на сервис с rewrite-ом пути. Где {student name} - это имя студента.

Хост в ингрессе должен быть arch.homework. В итоге после применения манифестов GET запрос на http://arch.homework/otusapp/{student name}/health должен отдавать {“status”: “OK”}.

На выходе предоставить
0) ссылку на github с манифестами. Манифесты должны лежать в одной директории, так чтобы можно было их все применить одной командой kubectl apply -f .
1) url, по которому можно будет получить ответ от сервиса (либо тест в postmane).

---

6 **Основы работы с Kubernetes (часть 3)**

Домашние задания

1 Инфраструктурные паттерны

Сделать простейший RESTful CRUD по созданию, удалению, просмотру и обновлению пользователей.
Пример API - https://app.swaggerhub.com/apis/otus55/users/1.0.0

Добавить базу данных для приложения.
Базу данных установить helm-ом из одного из репозиториев чартов (желательно официальных).
Конфигурация приложения должна хранится в Configmaps.
Доступы к БД должны храниться в Secrets.
Первоначальные миграции должны быть оформлены в качестве Job-ы.
Ingress-ы должны также вести на url arch.homework/otusapp/{student-name}/* (как и в прошлом задании)

На выходе должны быть предоставлена
1) ссылка на директорию в github, где находится директория с манифестами кубернетеса
2) инструкция по запуску приложения.
- команда установки БД из helm, вместе с файлом values.yaml.
- команда применения первоначальных миграций
- команда kubectl apply -f, которая запускает в правильном порядке манифесты кубернетеса
3) Postman коллекция, в которой будут представлены примеры запросов к сервису на создание, получение,

изменение и удаление пользователя. Важно: в postman коллекции использовать базовый url - arch.homework.

Задание со звездочкой:
+3 балла за шаблонизацию приложения в helm 3 чартах
+2 балла за использование официального helm чарта для БД и подключение его в чарт приложения в качестве зависимости.

---

7 **Мониторинг и алертинг.**

Мониторинг и алертинг
USE, RED и Four Golden Signals
SLI, SLO, SLA и бюджет на ошибки
Паттерны для сбора метрик
Управление инцидентами

---

8 **Prometheus. Grafana.**

Prometheus.
Grafana.
AlertManager.
PromQL

Домашние задания

1 Prometheus. Grafana

Инструментировать сервис из прошлого задания метриками в формате Prometheus с помощью библиотеки для вашего фреймворка и ЯП.

Сделать дашборд в Графане, в котором были бы метрики с разбивкой по API методам:
1. Latency (response time) с квантилями по 0.5, 0.95, 0.99, max
2. RPS
3. Error Rate - количество 500ых ответов

Добавить в дашборд графики с метрикам в целом по сервису, взятые с nginx-ingress-controller:
1. Latency (response time) с квантилями по 0.5, 0.95, 0.99, max
2. RPS
3. Error Rate - количество 500ых ответов

Настроить алертинг в графане на Error Rate и Latency.

На выходе должно быть:
0) скриншоты дашборды с графиками в момент стресс-тестирования сервиса. Например, после 5-10 минут нагрузки.
1) json-дашборды.

Задание со звездочкой (+5 баллов)
Используя существующие системные метрики из кубернетеса, добавить на дашборд графики с метриками:
1. Потребление подами приложения памяти
2. Потребление подами приолжения CPU

Инструментировать базу данных с помощью экспортера для prometheus для этой БД.
Добавить в общий дашборд графики с метриками работы БД.

---

9 **Service mesh на примере Istio**

Что такое service mesh
Плюсы, минусы и подводные камни service mesh.
Архитектура service mesh на примере Istio

Домашние задания

1 Развернуть в кластере две версии приложения и настроить балансировку трафика между ними

Цель: После выполнения ДЗ студент сможет:
- Развернуть Istio в кластере Kubernetes
- Настраивать балансировку трафика между разными версиями приложения

Инструкция к заданию и его описание находится по ссылке https://github.com/izhigalko/otus-homework-istio

---

10 **Авторизация и аутентификация в микросервисной архитектуре.**

Основные паттерны аутентификации и авторизации.
JWT токены.

---

11 **Backend for frontends. Apigateway.**

Паттерны BFF и APIgateway.

Домашние задания

1 Backend for frontends. Apigateway

Добавить в приложение аутентификацию и регистрацию пользователей.

Реализовать сценарий "Изменение и просмотр данных в профиле клиента".
Пользователь регистрируется. Заходит под собой и по определенному урлу получает данные о своем профиле. Может поменять данные в профиле. Данные профиля для чтения и редактирования не должны быть доступны другим клиентам (аутентифицированным или нет).

На выходе должны быть
0) описание архитектурного решения и схема взаимодействия сервисов (в виде картинки)
1) команда установки приложения (из helm-а или из манифестов). Обязательно указать в каком namespace нужно устанавливать.
1*) команда установки api-gateway, если он отличен от nginx-ingress.
2) тесты постмана, которые прогоняют сценарий:
- регистрация пользователя 1
- проверка, что изменение и получение профиля пользователя недоступно без логина

- вход пользователя 1
- изменение профиля пользователя 1
- проверка, что профиль поменялся
- выход* (если есть)
- регистрация пользователя 2
- вход пользователя 2
- проверка, что пользователь2 не имеет доступа на чтение и редактирование профиля пользователя1.

В тестах обязательно
- наличие {{baseUrl}} для урла
- использование домена arch.homework в качестве initial значения {{baseUrl}}
- использование сгенерированных случайно данных в сценарии
- отображение данных запроса и данных ответа при запуске из командной строки с помощью newman.

2 КОММУНИКАЦИОННЫЕ ПАТТЕРНЫ

1	**Асинхронный и синхронный API**	Основные типы межсервисного взаимодействия. Версионирование API Описание API.
2	**Event Driven Architecture.**	Основы Event Driven Architecture.
3	**Распределенные очереди сообщений на примере Kafka.**	
4	**Паттерны поддержания консистентности данных (Stream processing)**	Паттерн Transactional Log. Инструменты Change Data Capture.

Домашние задания

1 Stream processing

Реализовать сервис заказа. Сервис биллинга. Сервис нотификаций.

При создании пользователя, необходимо создавать аккаунт в сервисе биллинга. В сервисе биллинга должна быть возможность положить деньги на аккаунт и снять деньги.

Сервис нотификаций позволяет отправить сообщение на email. И позволяет получить список сообщений по методу API.

Пользователь может создать заказ. У заказа есть параметр - цена заказа.
Заказ происходит в 2 этапа:
1) сначала снимаем деньги с пользователя с помощью сервиса биллинга
2) отсылаем пользователю сообщение на почту с результатами оформления заказа. Если биллинг подтвердил платеж, должно отослаться письмо счастья. Если нет, то письмо горя.

Упрощаем и считаем, что ничего плохого с сервисами происходить не может (они не могут падать и т.д.). Сервис нотификаций на самом деле не отправляет, а просто сохраняет в БД.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ (5 баллов):
0) Спроектировать взаимодействие сервисов при создании заказов. Предоставить варианты взаимодействий в следующих стилях в виде sequence диаграммы с описанием API на IDL:
- только HTTP взаимодействие
- событийное взаимодействие с использование брокера

		сообщений для нотификаций (уведомлений) - Event Collaboration стиль взаимодействия с использованием брокера сообщений - вариант, который вам кажется наиболее адекватным для решения данной задачи. Если он совпадает одним из вариантов выше - просто отметить это. ПРАКТИЧЕСКАЯ ЧАСТЬ (5 баллов): Выбрать один из вариантов и реализовать его. На выходе должны быть 0) описание архитектурного решения и схема взаимодействия сервисов (в виде картинки) 1) команда установки приложения (из helm-а или из манифестов). Обязательно указать в каком namespace нужно устанавливать. 2) тесты постмана, которые прогоняют сценарий: 1. Создать пользователя. Должен создаться аккаунт в биллинге. 2. Положить деньги на счет пользователя через сервис биллинга. 3. Сделать заказ, на который хватает денег. 4. Посмотреть деньги на счету пользователя и убедиться, что их сняли. 5. Посмотреть в сервисе нотификаций отправленные сообщения и убедиться, что сообщение отправилось 6. Сделать заказ, на который не хватает денег. 7. Посмотреть деньги на счету пользователя и убедиться, что их количество не поменялось. 8. Посмотреть в сервисе нотификаций отправленные сообщения и убедиться, что сообщение отправилось. В тестах обязательно - наличие {{baseUrl}} для урла - использование домена arch.homework в качестве initial значения {{baseUrl}} - отображение данных запроса и данных ответа при запуске из командной строки с помощью newman.
5	**RESTful**	GraphQL. Odata.
6	**GraphQL. gRPC**	
7	**Идемпотетность и коммутативность API в HTTP и очередях.**	Идемпотентность и коммутативность API. Idempotent reciever. Домашние задания 1 Идемпотетность и коммутативность API в HTTP и очередях
8	**Тестирование микросервисов (часть 1)**	Сценарное нагрузочное тестирование. Consumer based contracts.

9 **Тестирование микросервисов (часть 2)**

3 РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ

1	**DDD и модульные монолиты. Часть 1**	
2	**DDD и модульные монолиты. Часть 2**	
3	**Паттерны декомпозиции микросервисов.**	Декомпозиция про предметной области. Event Storming Домашние задания 1 Паттерны декомпозиции микросервисов
4	**От монолита к микросервису.**	Паттерн Strangler. Паттерны работы с данными при рефакторинге.

4 ДЕЦЕНТРАЛИЗОВАННЫЕ СИСТЕМЫ

1	**Введение в распределенные системы.**	BASE. ACID. Проблемы CAP теоремы. Основные проблемы.
2	**Распределенные транзакции.**	Проблемы распределенных транзакций. Паттерн Сага. Домашние задания 1 Распределенные транзакции
3	**Паттерны кэширования и основные принципы.**	Архитектурные паттерны кэширования. Различные алгоритмы кэширования и
4	**Шардирование.**	Виды шардинга Стратегии шардирования Консистентное шардирование
5	**CP системы.**	Проблема византийских генералов Алгоритмы консенсуса. Алгоритм Raft. Пример реализации своей CP системы
6	**AP системы.**	Алгоритм GOSSIP (Scuttlebut) Пример реализации своей AP системы

5 АРХИТЕКТОР

1	**Роль архитектора**	Домашние задания 1 Event Sourcing и CQRS
2	**Стоимость архитектуры. Артефакты архитектуры.**	Governance as a Code. Эволюционное развитие архитектуры. Процесс управлением архитектурными изменениями.

6 Итоговый проект

1	**Консультация по проектам и домашним заданиям**	получить ответы на вопросы по проекту, ДЗ и по курсу. Домашние задания 1Проект
2	**Защита проектных работ**	защитить проект и получить рекомендации экспертов.