OTUS

# Microservice Architecture

Best Practice по разработке архитектуры программного обеспечения

Длительность курса: 122 академических часа

# 1 ИНФРАСТРУКТУРНЫЕ ПАТТЕРНЫ

## 1 Плюсы и минусы микросервисной архитектуры

**Цели занятия:**

рассмотреть плюсы и минусы монолитов;
рассмотреть плюсы и минусы микросервисной архитектуры;
рассмотреть сновные паттерны микросервисной архитектуры.

**Краткое содержание:**

архитектура и роль архитектора
про боли с монолитом
про боли с микросервисами
основные паттерны в микросервисной архитектуре

---

## 2 Основы работы с Docker

**Цели занятия:**

обсудить контейнеризацию;
рассмотреть компоненты docker.

**Краткое содержание:**

контейнеризация: краткий обзор;
компоненты docker: engine, cli, registry;
сборка контейнеров, dockerfile;
практика: build, run, up, down, pull, push.

---

| | | |
|---|---|---|
| 3 | **Инфраструктурные паттерны** | **Цели занятия:**<br><br>рассмотрим инфраструктурные паттерны;<br>обсудим основные инфраструктурные проблемы;<br><br>**Краткое содержание:**<br><br>CI/CD;<br>VM vs сервер приложений vs containers;<br>конфигурирование приложений;<br>Паттерны деплоя;<br>Service Discovery;<br>Health check. |
| 4 | **Основы работы с Kubernetes (часть 1)** | **Цели занятия:**<br><br>рассмотреть архитектуру и базовые сущности Kubernetes: Pod, Deployment, ReplicaSet<br><br>**Краткое содержание:**<br><br>основы Kubernetes. |
| 5 | **Основы работы с Kubernetes (часть 2)** | **Цели занятия:**<br><br>рассмотреть архитектуру и базовые сущности Kubernetes: ConfigMap, Persistent Volume, Persistent Volume Claim.<br><br>**Краткое содержание:**<br><br>Pod. Docker;<br>REPLICASET;<br>DEPLOYMENT;<br>SERVICEService discovery;<br>SKAFFOLD;<br>HEALTH CHECK;<br>BLUEGREEN;<br>INGRESS;<br>STATEFULSETS;<br>CONFIGURATION; |

HELM;
HELM-DEP.

**Домашние задания**

1 Основы работы с Kubernetes (Часть 2)

Цель: В этом ДЗ вы научитесь создавать минимальный сервис.

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 могут быть указаны Liveness, Readiness пробы.
Количество реплик должно быть не меньше 2.
Image контейнера должен быть указан с Dockerhub.

Хост в ингрессе должен быть arch.homework. В итоге после применения манифестов GET запрос на http://arch.homework/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).

Задание со звездой* (+5 баллов):
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.

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

**Цели занятия:**

рассмотреть архитектуру и базовые сущности Kubernetes: ConfigMap, Job.

**Краткое содержание:**

Helm 3 и шаблонизация манифестов;
Configmap, Secrets
Job

**Домашние задания**

1 Инфраструктурные паттерны

Цель: В этом ДЗ вы создадите простейший RESTful CRUD.

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.
Конфигурация приложения должна хранится в Configmaps.
Доступы к БД должны храниться в Secrets.
Первоначальные миграции должны быть оформлены в качестве Job-ы, если это требуется.
Ingress-ы должны также вести на url arch.homework/ (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:
+5 балла за шаблонизацию приложения в helm чартах

---

7 **Kubernetes. QA**

**Цели занятия:**

рассмотреть основные инфраструктурные паттерны.

**Краткое содержание:**

обсуждение вопросов.

---

8 **Мониторинг и алертинг**

**Цели занятия:**

рассмотреть мониторинг и алертинг.

**Краткое содержание:**

мониторинг и алертинг;
USE, RED и Four Golden Signals;
SLI, SLO, SLA и бюджет на ошибки;
паттерны для сбора метрик;
управление инцидентами.

---

9 **Prometheus. Grafana**

**Цели занятия:**

prometheus;
grafana;
alertManager;
promQL.

**Краткое содержание:**

метрики Prometheus;
PromQL;
service Monitors;
demo.

**Домашние задания**

1 Prometheus. Grafana

Цель: В этом ДЗ вы научитесь инструментировать сервис.

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.

Альтернативное задание на 1 балл (если не хочется самому ставить prometheus в minikube) - https://www.katacoda.com/schetinnikov/scenarios/prometheus-client

---

10 **Service mesh на примере Istio**

**Цели занятия:**

ответить на вопрос, что такое Service mesh?
рассказать об устройстве Istio;
рассказать о возможностях Istio и способах их настройки.

**Краткое содержание:**

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: В этом ДЗ вы научитесь:
- разворачивать 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

---

11 **Авторизация и аутентификация в микросервисной архитектуре**

**Цели занятия:**

рассмотреть основные паттерны аутентификации и авторизации, JWT токены.

**Краткое содержание:**

паттерны аутентификации в монолитах;
Identity Provider-ы и OIDC;
Token-based аутентификация и JWT;
Auth proxy;
паттерны межсервисной аутентификации.

---

12 **Backend for frontends. Apigateway**

**Цели занятия:**

рассмотреть паттерны BFF и APIgateway.

**Краткое содержание:**

API Gateway;
BFF;
Паттерны аутентификации в API Gateway;
Curcuit Breaker, Retry;
демо.

**Домашние задания**

1 Backend for frontends. Apigateway

Цель: В этом ДЗ вы научитесь добавлять в приложение аутентификацию и регистрацию пользователей.

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**

**Цели занятия:**

рассмотреть основные типы межсервисного взаимодействия;
обсудить версионирование API;
рассмотреть описание API.

**Краткое содержание:**

асинхронный и синхронный API;
оркестрация и хореография
версионирование API
IDL, API design first
Anemic API vs Rich API

---

2 **Event Driven Architecture**

**Цели занятия:**

рассмотреть основы Event Driven Architecture.

**Краткое содержание:**

события и сообщения;
паттерны проектирования событий;
паттерны использования событий.

---

3 **Распределенные очереди сообщений на примере Kafka**

**Цели занятия:**

rabbitmq;
Kafka.

**Краткое содержание:**

распределение очередей:
Kafka.

---

4 **Паттерны поддержания консистентности**

**Цели занятия:**

**данных (Stream processing)**

рассмотреть паттерн Transactional Log;
рассмотреть инструменты Change Data Capture.

**Краткое содержание:**

Stream Processing;
Event Sourcing;
Change Data Capture.

**Домашние задания**

1 Stream processing

Цель: В этом ДЗ вы научитесь реализовывать сервис заказа.

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 **GraphQL. gRPC**

**Цели занятия:**

рассмотреть основные паттерны использования GraphQL. Основные паттерны использования gRPC.

**Краткое содержание:**

GraphQL;
gRPC.

---

6 **RESTful**

**Цели занятия:**

рассмотреть GraphQL, Odata.

**Краткое содержание:**

REST. Maturity Levels;
Anemic API vs Rich API;
RESTful Patterns;
JsonScheme, OpenAPI;
GraphQL.

---

7 **Идемпотентость и коммутативность API в HTTP и очередях**

**Цели занятия:**

рассмотреть два кейса по декомпозиции сервисов;
на их примере выделим типичные проблемы при декомпозиции сервисов;
рассмотреть два паттерна и два антипаттерна.

**Краткое содержание:**

идемпотентность и коммутативность API;
Idempotent reciever.

**Домашние задания**

1 Идемпотентость и коммутативность API в HTTP и очередях

Цель: В этом ДЗ вы создадите сервис "Заказ" (или научитесь использовать сервис из прошлого занятия) и для одного из его методов, например, "создание заказа" сделаете идемпотетным.

На выходе должно быть:
0) описание того, какой паттерн для реализации идемпотентности использовался
1) команда установки приложения (из helm-а или из манифестов). Обязательно указать в каком namespace нужно устанавливать и команду создания namespace, если это важно для сервиса.
2) тесты в postman

В тестах обязательно
- использование домена arch.homework в качестве initial значения {{baseUrl}}

8 **Тестирование микросервисов (часть 1)**

**Цели занятия:**

рассмотреть ценарное нагрузочное тестирование;
рассмотреть Consumer based contracts.

**Краткое содержание:**

примеры;
плюсы и минусы serverless решений; кейсы использования.

---

9 **Тестирование микросервисов (часть 2)**

**Цели занятия:**

объяснить, что такое serverless вычисления;
рассмотреть виды serverless услуг.

**Краткое содержание:**

примеры;
плюсы и минусы serverless решений;
кейсы использования.

# 3 РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ

1 **DDD и модульные монолиты (часть 1)**

**Цели занятия:**

рассмотрим основы DDD и применение к ООП;
поймем как DDD помогает в построении архитектуры.

**Краткое содержание:**

ООП;
DDD.

---

2 **DDD и модульные монолиты (часть 2)**

**Цели занятия:**

рассмотрим основы DDD и применение к ООП.

**Краткое содержание:**

ООП;
DDD.

---

3 **Паттерны декомпозиции микросервисов**

**Цели занятия:**

объяснить как декомпозировать на сервисы.

**Краткое содержание:**

пользовательский сценарий;
модель предметной области;
модель на основе ООП;
функциональная модель;
разбиение на сервисы.

**Домашние задания**

1 Паттерны декомпозиции микросервисов

Цель: В этом ДЗ вы разделите ваше приложение на несколько микросервисов с учетом будущих изменений.

Попробуйте сделать несколько вариантов разбиений и попробуйте их оценить. Выберите вариант, который вы будете реализовывать.

На выходе вы должны предоставить
1) Пользовательские сценарии
2) Общую схему взаимодействия сервисов.
3) Для каждого сервиса опишите назначение сервиса и его зону ответственности.
4) Опишите контракты взаимодействия сервисов друг с другом.

---

4 **От монолита к микросервису**

**Цели занятия:**

рассмотреть паттерн Strangler;
рассмотреть паттерны работы с данными при рефакторинге.

**Краткое содержание:**

трассировка (tracing);
монолит в микросервисы.

# 4 ДЕЦЕНТРАЛИЗОВАННЫЕ СИСТЕМЫ

1 **Введение в распределенные системы**

**Цели занятия:**

рассмотреть распределенные системы.

**Краткое содержание:**

виды распределенных систем;
CAP и PACELC теоремы;
BASE и ACID

---

2 **Распределенные транзакции**

**Цели занятия:**

выбрать стратегию реализации согласованности в распределённой архитектуре;
использовать двухфазные коммиты и паттерн «Сага»;
использовать паттерны транзакционной отправки сообщений.

**Краткое содержание:**

проблемы распределенных транзакций;
паттерн Сага.

**Домашние задания**

1 Распределенные транзакции

Цель: В этом ДЗ вы научитесь реализовывать распределенную транзакцию.

Можно использовать приведенный ниже сценарий для интернет-магазина или придумать свой.

Дефолтный сценарий:
Реализовать сервисы "Платеж", "Склад", "Доставка".

Для сервиса "Заказ", в рамках метода "создание заказа" реализовать механизм распределенной транзакции (на основе Саги или двухфазного коммита).
Во время создания заказа необходимо:

1) в сервисе "Платеж" убедиться, что платеж прошел
2) в сервисе "Склад" зарезервировать конкретный товар на складе
3) в сервисе "Доставка" зарезервировать курьера на конкретный слот времени.

Если хотя бы один из пунктов не получилось сделать, необходимо откатить все остальные изменения.

На выходе должно быть:
0) описание того, какой паттерн для реализации распределенной транзакции использовался
1) команда установки приложения (из helm-а или из манифестов). Обязательно указать в каком namespace нужно устанавливать и команду создания namespace, если это важно для сервиса.
2) тесты в postman

В тестах обязательно
- использование домена arch.homework в качестве initial значения {{baseUrl}}

---

3 **Паттерны кэширования и основные принципы**

**Цели занятия:**

использовать основные паттерны кэширования;
решать типичные проблемы, связанные с кэшированием;
выбирать инструмент кэширования под задачу.

**Краткое содержание:**

архитектурные паттерны кэширования;
различные алгоритмы кэширования.

---

4 **Шардирование**

**Цели занятия:**

рассмотреть виды шардинга;
проанализировать стратегии шардирования;
рассмотреть консистентное шардирование.

**Краткое содержание:**

кейсы шардирования (поиск, вычисления, хранение);
как правильно делить данные;
распределенный поиск (пример со sphinx)
map-reduce (можно на примере mongodb)
основные ошибки при шардировании (шардирование по плохому критерию, неравномерная нагрузка)
кто координирует (клиент, координатор, прокси)

---

5 **CP системы**

**Цели занятия:**

рассмотреть проблему византийских генералов;
рассмотреть алгоритмы консенсуса, Raft.
обсудить пример реализации своей CP системы.

**Краткое содержание:**

синхронизация изменений;
алгоритмы согласования;
Paxos;
Raft;
Zab.

---

6 **AP системы**

**Цели занятия:**

рассмотреть алгоритм GOSSIP (Scuttlebut);
обсудить пример реализации своей AP системы.

**Краткое содержание:**

AP системы;
проблемы master-master репликации;
Gossip: Scuttlebutt;
репликация без master’а (dynamo-подобные БД).

# 5 АРХИТЕКТОР

1 **Роль архитектора**

**Цели занятия:**

обсудить: кто такой архитектор? какая у него роль?

**Краткое содержание:**

архитерктор.

---

2 **Стоимость архитектуры. Артефакты архитектуры**

**Цели занятия:**

обсудить стоимость архитектуры;
научиться управлять рисками;
презентовать свои решения.

**Краткое содержание:**

Governance as a Code;
эволюционное развитие архитектуры;
процесс управлением архитектурными изменениями.

# 6 ИТОГОВЫЙ ПРОЕКТ

1 **Консультация по проектам и домашним заданиям**

**Цели занятия:**

получить ответы на вопросы по проекту, ДЗ и по курсу.

**Краткое содержание:**

вопросы по улучшению и оптимизации работы над проектом;
затруднения при выполнении ДЗ;
вопросы по программе.

**Домашние задания**

1 Проектная работа

Цель: В этом дз необходимо выбрать и утвердите в чате с преподавателем темы проекта, разработать и презентовать проект.

1. Выбрать тему
2. Утвердить темы в чате по дз
3. Презентовать проект

---

2 **Защита проектных работ**

**Цели занятия:**

защитить проект и получить рекомендации экспертов.

**Краткое содержание:**

презентация проектов перед комиссией;
вопросы и комментарии по проектам.