SKILLFACTORY

DEVOPS-ИНЖЕНЕР

КОНСПЕКТ МОДУЛЯ

Контейнерная оркестрация. Docker Swarm. AWS Elastic Container Service (ECS)

Системы оркестрации нужны для автоматического управления некоторым множеством контейнеров.

Системы оркестрации контейнерами берут на себя роль управляющего дирижера, который координирует работу множества контейнеров, организует их правильное взаимодействие согласно описанным правилам.

Задачи контейнерных оркестраторов:

1. Инициализация и запуск контейнеров.
2. Обеспечение сетевого взаимодействия между контейнерами кластера.
3. Масштабирование контейнерных юнитов кластера.
4. Контроль за соответствием желаемого состояния действительному.
5. Контроль за распределением нагрузки.
6. Правила доставки новых версий и отката к предыдущим.

Микросервисная архитектура (MSA Micro Service Architecture) — это подход применяемый для создания гибкого, масштабируемого приложения путем разделения его составных частей на множество независимых юнитов разделенных по бизнес признаку или по любому другому.

Под **монолитным приложением** подразумевается решение, содержащее в себе всю логику составных частей приложения. Взаимодействие между ними происходит внутри единого процесса.

В основе подхода SOA лежит несколько базовых идей:

- переиспользование компонентов системы путём разделения их на составные части;
- использование единой шины (ESB — Enterprise Service Bus) для взаимодействия и маршрутизации между компонентами.

Микросервисный подход

Преимущества	Недостатки
Независимый деплой, частичное развёртывание. Важным преимуществом микросервисов является возможность независимого деплоя разных компонентов системы в разное время. Поэтому данный подход характеризуется слабой связанностью.	**Простота.** Микросервисная архитектура требует более тщательного подхода при проектировании и поддержке, так как взаимодействие между компонентами происходит посредством API.
Мультиплатформенность и гетерогенность. Поскольку компоненты системы разделены между собой, за командой остается выбор технологии для реализации конкретного компонента системы, а также платформы, на которой будет работать микросервис.	**Согласованность.** Это самый сложный момент при реализации микросервисной архитектуры. Из этого недостатка вытекает и достоинство: управлять микросервисом может ответственная именно за этот микросервис команда с соблюдением своих внутрикомандных стандартов, не затрагивающих остальные сервисы.
Доступность. Увеличение доступности осуществляется за счёт того, что сбой одного компонента системы не приводит к сбою системы в целом.	**Межмодульный рефакторинг.** Когда требуется произвести рефакторинг нескольких модулей, необходимо учесть вопросы взаимодействия между модулями. Кроме того, если мы хотим изменить функциональность и перенести её в другой сервис, мы должны чётко разделять границы микросервисов, чтобы сервис не вырос в нечто большее и остался в своём контексте.
Модульность. Микросервисы помогут поддерживать ваш проект в соответствии с правилами SOLID.	

КОНСПЕКТ МОДУЛЯ

Контейнерная оркестрация. Docker Swarm. AWS Elastic Container Service (ECS)

Принципы построения микросервисов:

1. Небольшой размер.
2. Автономность.
3. Контекст.
4. Логика взаимодействия сервисов осуществляется по сети.
5. Страховка от сбоев.
6. Децентрализованное управление данными.

Типы сетей в Docker:

1. Bridge;
2. Host;
3. Overlay;
4. Macvlan;
5. None;
6. Сетевые плагины.

Трафик в Docker Swarm разделен на два типа:

- служебный;
- пользовательский.

Amazon Elastic Container Service (Amazon ECS) — это сервис от Amazon для запуска контейнеров на базе Docker. Контейнеры крутятся, в свою очередь, в инстансах EC2 (облачных виртуальных машин). Кроме того, ECS предоставляет возможность объединения ECS-инстансов в кластер.

В самом простом варианте вы берете нужный вам образ, указываете сколько ресурсов ему нужно, и запускаете ваше приложение в облаке — это и есть ECS. Таким образом, можно сразу отметить ключевые преимущества: простота работы с этим сервисом и тесное взаимодействие с другими обширными возможностями AWS.

Подобную ECS функциональность мы с вами уже видели в решении Docker Swarm, но давайте рассмотрим поближе этот сервис.