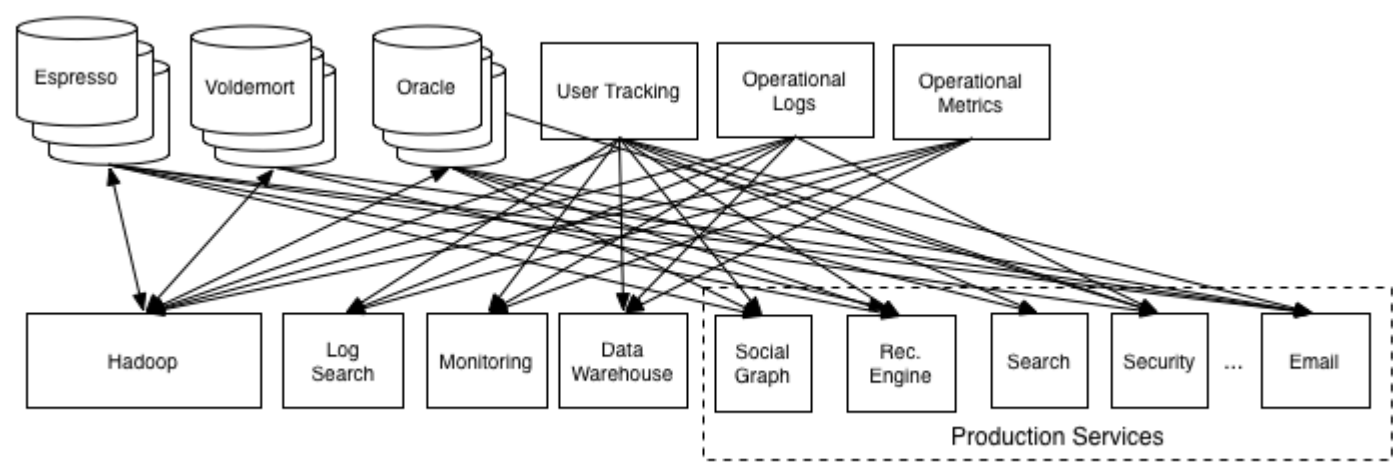


Самый важный антипаттерн перехода к микросервисам - распределённый монолит

Распределённый монолит — это антипаттерн в микросервисной архитектуре, при котором, несмотря на физическое разделение компонентов на отдельные сервисы, система сохраняет характеристики монолитной архитектуры. То есть ситуация, когда микросервисы не могут выполнить какую-то функцию не сходяв за многими данными в другие микросервисы. Вот как это выглядело в LinkedIn в 2010-х:



Основными проблемами распределённого монолита являются:

Сильная связанность

Сервисы так сильно зависят друг от друга, что не могут функционировать независимо. Изменение в одном сервисе зачастую требует изменений в других, что уменьшает главное преимущество микросервисов — независимость разработки и развертывания.

Сложность развертывания

Поскольку сервисы тесно связаны, они часто должны быть развернуты совместно. Это уменьшает возможности для независимого масштабирования и развертывания, а также усложняет процедуры отката.

Сложность управления

Каждый сервис может требовать своего набора инструментов для мониторинга, логирования и тестирования. При тесной связанности эти инструменты становятся менее эффективными, поскольку они должны учитывать взаимозависимости между сервисами.

Проблемы с производительностью

При распределённом монолите часто возникают проблемы с производительностью из-за чрезмерного количества межсервисных вызовов. Сетевая задержка и сложность агрегации данных из разных источников могут существенно снизить производительность системы.

Затруднение масштабирования

Масштабирование становится проблематичным, поскольку при изменении масштаба одного сервиса зачастую необходимо менять масштаб смежных сервисов.

Распределённый монолит объединяет недостатки монолитной и микросервисной архитектур, не давая при этом преимуществ ни одной из них.

Поэтому при проектировании микросервисов важно избегать избыточного разбиения и фокусироваться на создании независимых, хорошо определённых сервисов.

Используйте принципы низкой связанности и высокой сплоченности для создания микросервисной архитектуры. Сервисы должны быть как можно менее зависимыми друг от друга, при этом каждый сервис должен предоставлять функциональность, логически связанную внутри себя.