**Проблема больших монолитных систем**

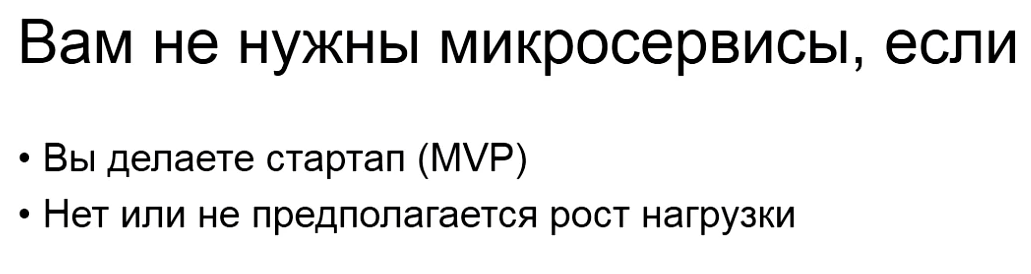
* Плохое горизонтальное масштабирование
* Плохая отказоустойчивость (из-за того что сложно горизонтально масштабироваться)
* Сложность внедрения новых технологий (БД, языки, …)
* Сложность рефакторинга legacy (из за того что сложно заменить технологию, то усложняется поддержка старых технологий, а так же новые разработчики не будут хотеть у нас рабоать)

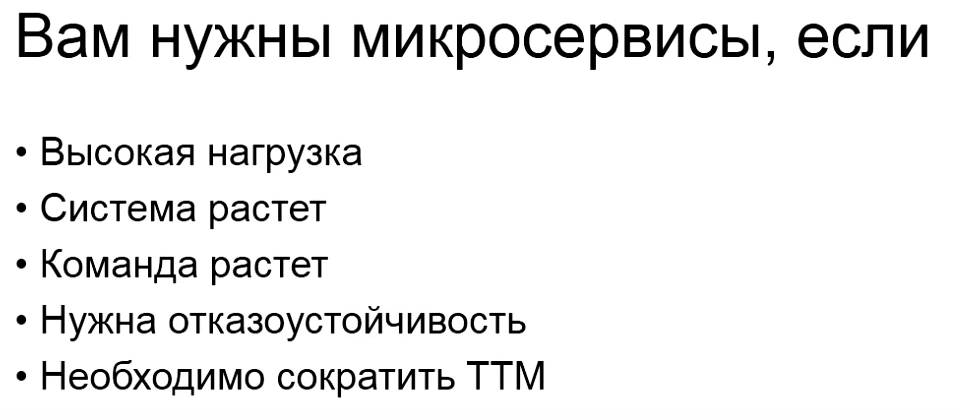
**Плюсы микросервисов**

* Горизонтальное масштабирование
* Отказоустойчивость (можно сделать несколько копий микросервиса, и если один откажет, то будет работать его копия)
* Масштабирование команд
* Переиспользование (один микросервис можно использовать много раз – например сервис отправки писем можно использовать в разных приложениях)
* Гибкость стека

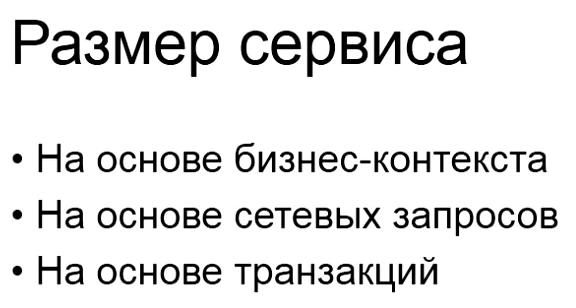
Минусы микросервиса

* Сложность (если раньше была одна система мониторинга – в монолите, то сейчас на каждый микросервис нужно весить мониторинги, так же разбираться с коммуникацией микросервисов и тд)
* Дорого разрабатывать
* Не согласованные данные

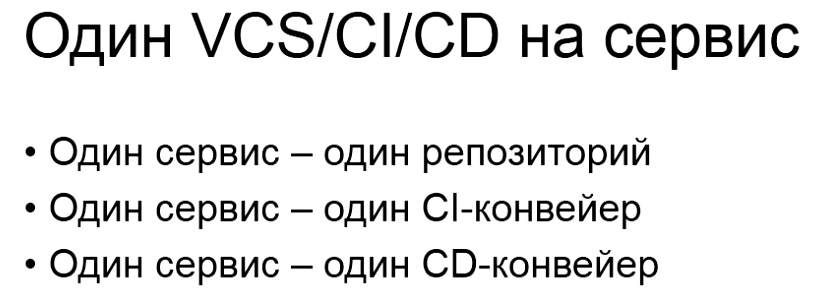




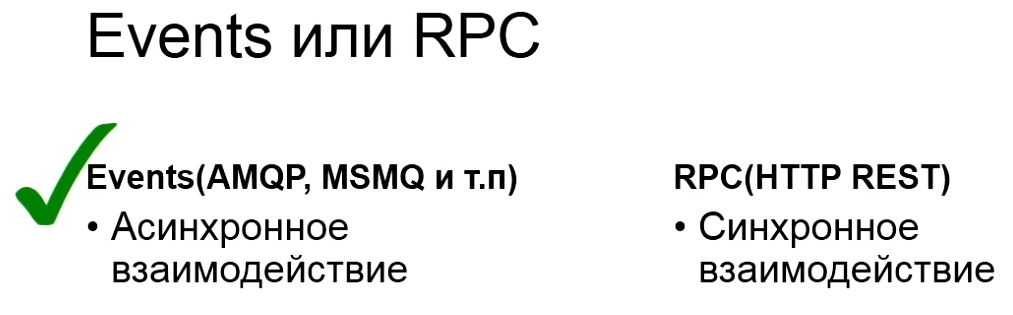
 ПО СЛОЯЕМ ИМЕЕТСЯ ВВИДУ СЕРВИС С FRONTEND, СЕРВИС С ЛОГИКОЙ И СЕРВИС С БД

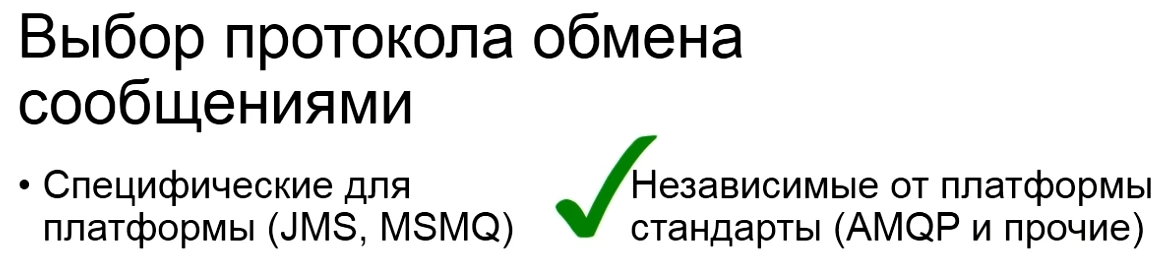


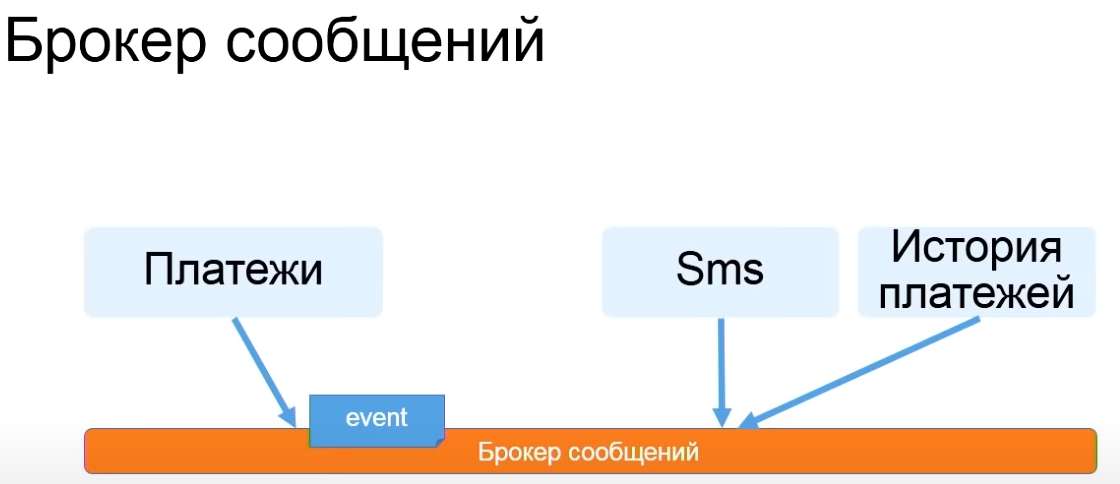
На основе сетевых запросов – это если 2 микросервиса постоянно обмениваются информацией, то лучше их объеденить в один

Что бы в случае изменений данных в 1 сервисе, не обновлялись остальные



 Для коммуникации микросервисом между собой используем Брокеров Сообщений



В каждый сервис уже инкапуслирована вся логика, и в случае получения евента, он знает как ему себя повести