

# Итак, что нужно учесть при горизонтальном масштабировании

## 1. Stateless архитектура:

- **Основа:** Прежде всего, нужен единый сервер аутентификации (с хранилищем сессий), чтобы в случае смены сервера (например запросы от клиента были к фронтенд1, а теперь будут к фронтенд2) сохранялась вся информация о сессии клиента - и ему не приходилось заново перезаходить в систему.
- **JWT и OAuth:** Используйте эти технологии, чтобы реализовать данную возможность. И можно хранить сессии на стороне клиента в LocalStorage браузера.

## 2. Балансировка нагрузки:

- **Основа:** Нужен балансировщик нагрузки, чтобы распределять входящие запросы между различными фронтенд/бэкенд серверами.
- **Алгоритмы Балансировки:** Рассмотрите различные алгоритмы балансировки (Round Robin, Least Connections и т.д.) в зависимости от требований вашего приложения.

## 3. Микросервисы:

- **Независимое Масштабирование:** Каждый микросервис можно масштабировать независимо.
- **Командная Работа:** Это также облегчает параллельную разработку различными командами.

## Принципы, которых стоит придерживаться:

1. **Data Partitioning (Разделение данных):** Системы, которые хранят большие объемы данных, должны уметь эффективно их разделять (шардировать) между разными физическими или виртуальными устройствами для оптимальной загрузки.
2. **Redundancy (Избыточность):** Горизонтальное масштабирование часто включает в себя добавление избыточных узлов, что повышает отказоустойчивость системы.  
Например - резервный сервис в пару к действующему на отдельной машине.

3. **Репликация**: копирование данных в отдельные базы данных (в том числе в избыточные сервисы) для поддержания согласованности данных.
4. **Consistency (Согласованность)**: В распределенных системах большое значение имеет обеспечение согласованности данных. Принципы, как CAP-теорема, помогают понять компромиссы между согласованностью, доступностью и устойчивостью к разделением сети.
5. **Auto-scaling (Автомасштабирование)**: Эффективное горизонтальное масштабирование часто подразумевает возможность автоматического добавления или удаления ресурсов в зависимости от текущей нагрузки.
6. **Monitoring and Metrics (Мониторинг и метрики)**: Для успешного масштабирования необходима система мониторинга, которая может в реальном времени отслеживать состояние приложения и использование ресурсов.
7. **Geographic Distribution (Географическое распределение)**: В некоторых случаях (например, для сокращения времени отклика или из-за законодательных ограничений) ресурсы распределяются географически.
8. **Idempotency (Идемпотентность)**: Важность этого принципа заключается в том, чтобы система могла корректно обрабатывать повторные запросы, что особенно актуально при распределенной обработке.
9. **Rate Limiting (Ограничение скорости)**: Этот принцип помогает контролировать поток запросов к каждому из микросервисов или узлов, предотвращая их перегрузку.