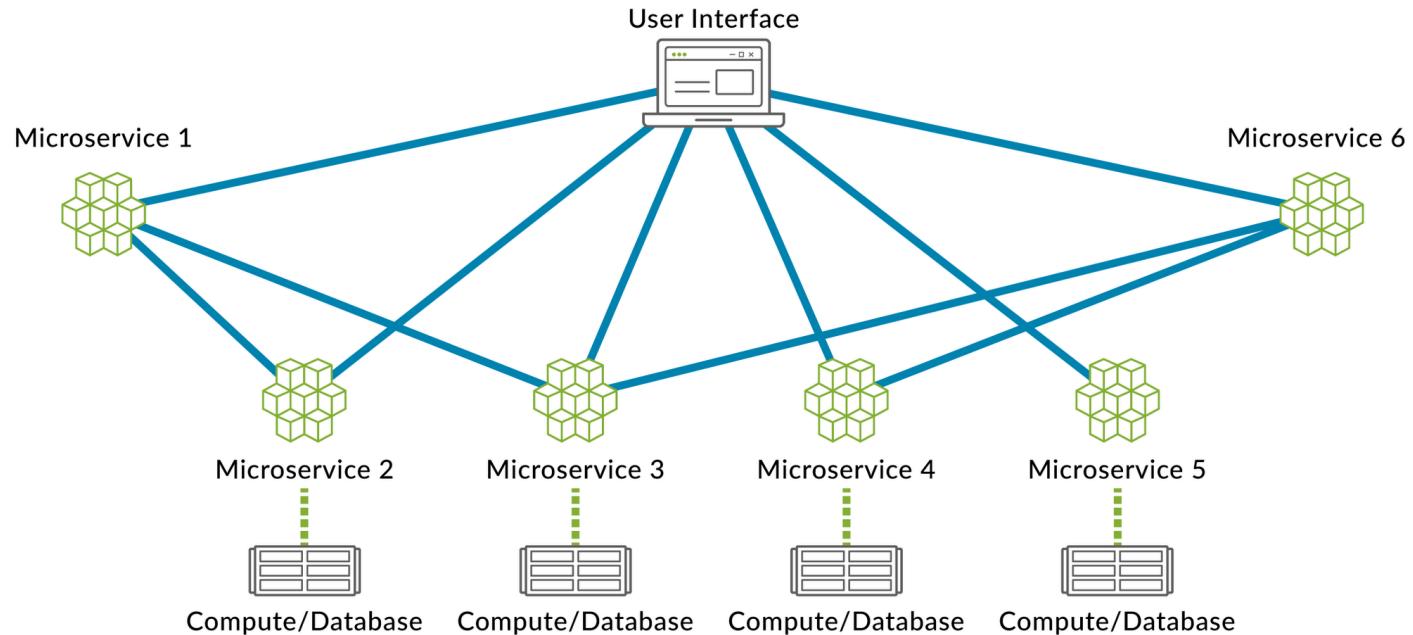


8119
LMT DAY

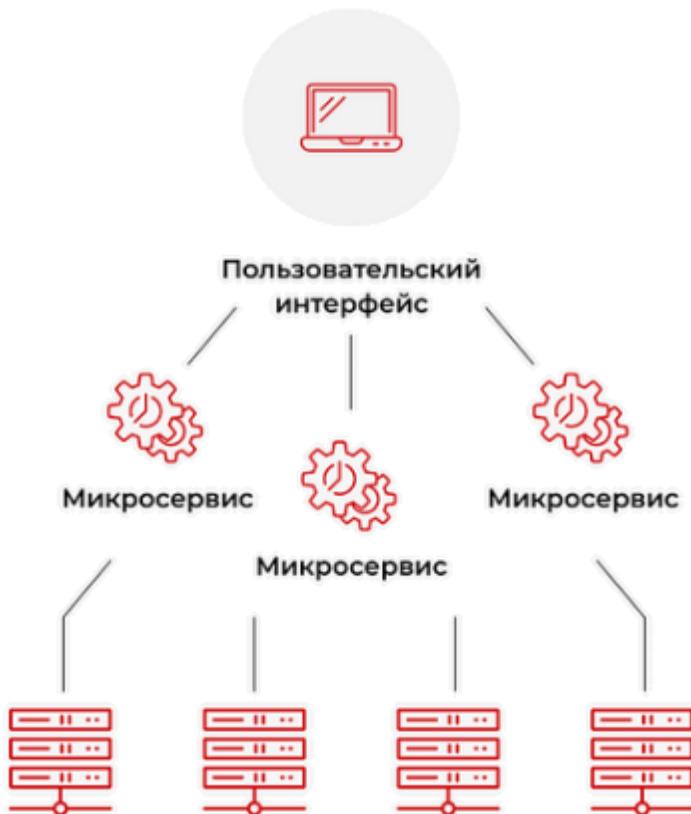
Sell 100
8119
LMT DAY

Микросервисы 2

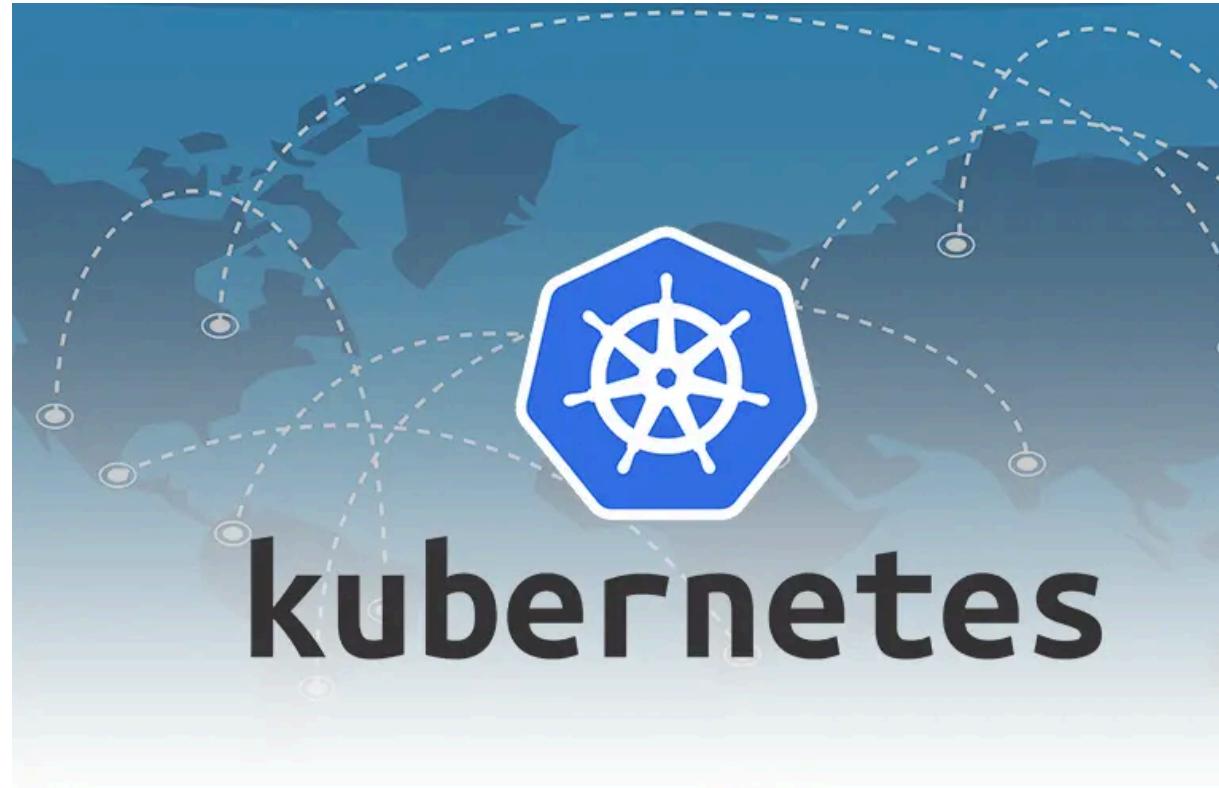
Microservices Cloud Architecture



О чём эта лекция?

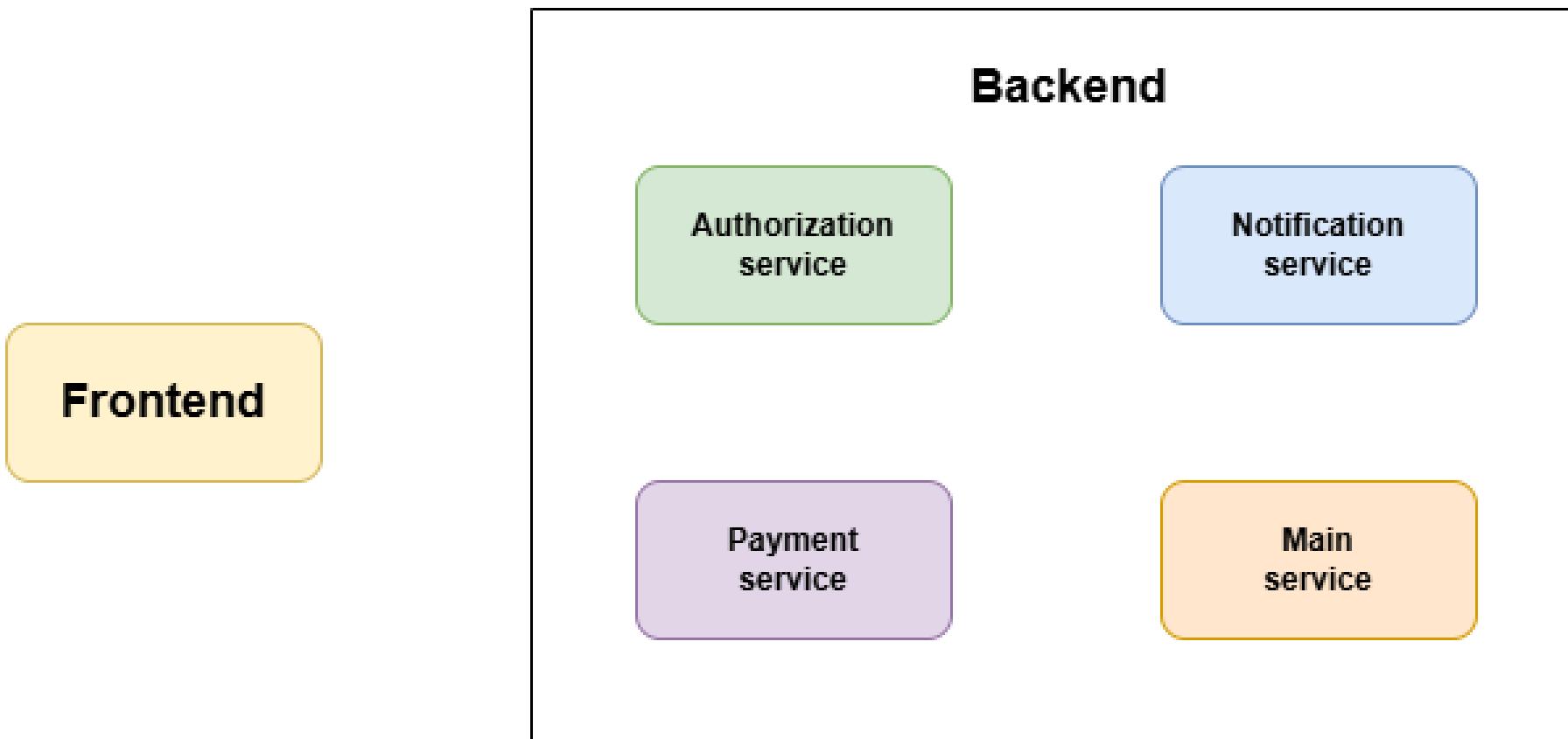


- 1 Сложности микросервисов
- 2 Что такое оркестрация?
- 3 Инструменты оркестрации
- 4 Что такое Kubernetes?
- 5 Задачи Kubernetes
- 6 Объекты Kubernetes
- 7 Что такое CI/CD?
- 8 Continuous Integration и его преимущества
- 9 Непрерывное тестирование
- 10 Continuous Delivery и Continuous Deployment и их отличие
- 11 Преимущества CD

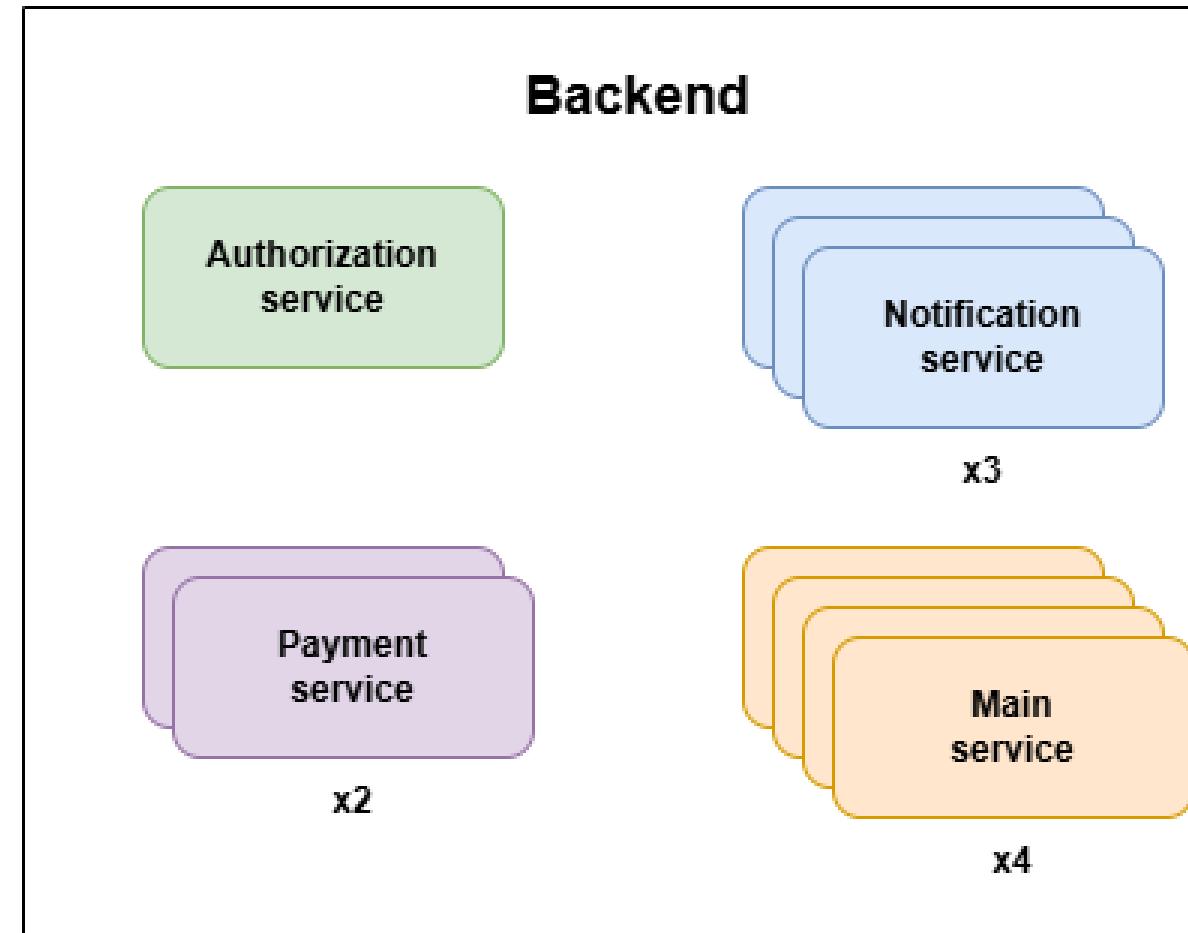




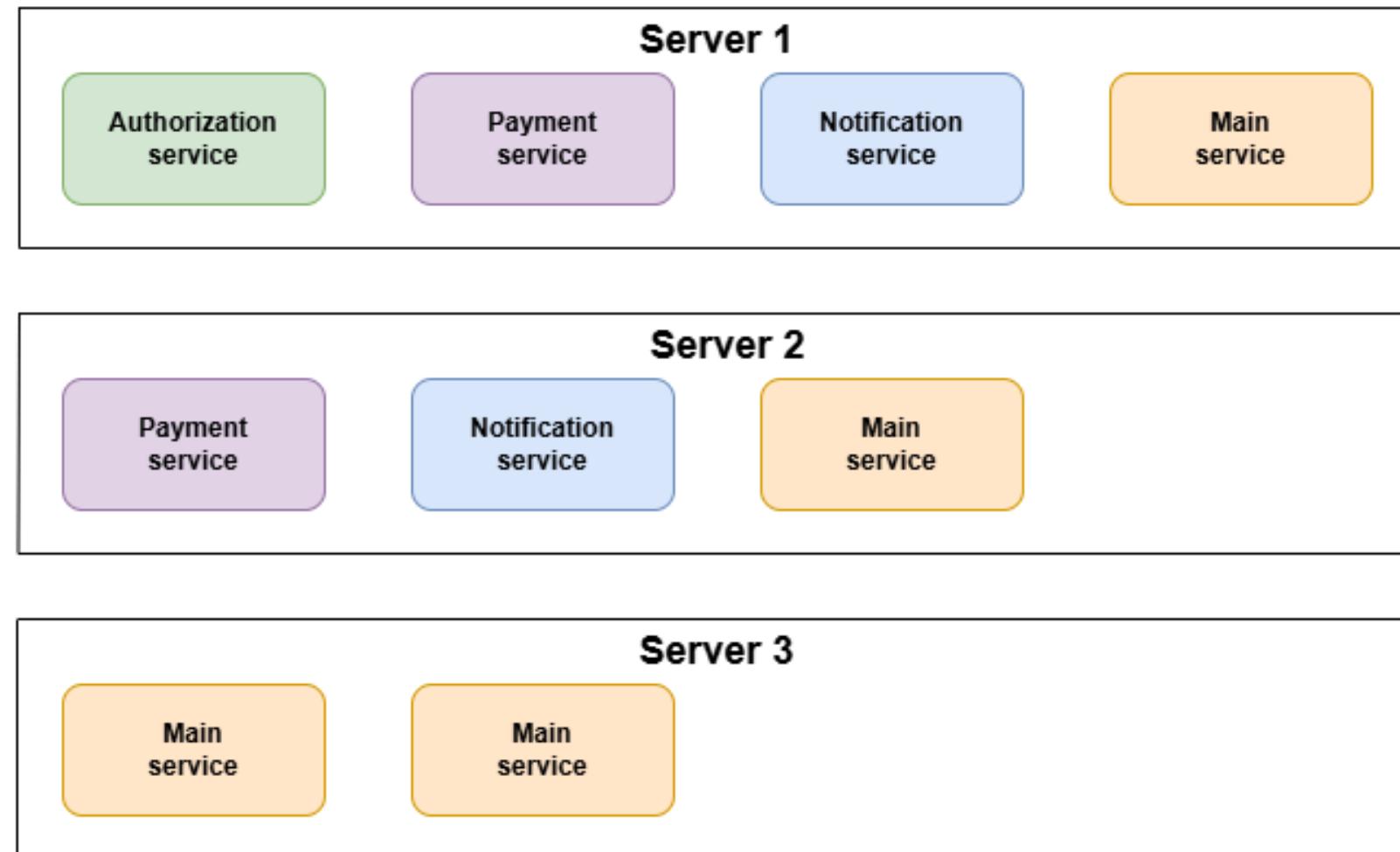
Frontend и backend



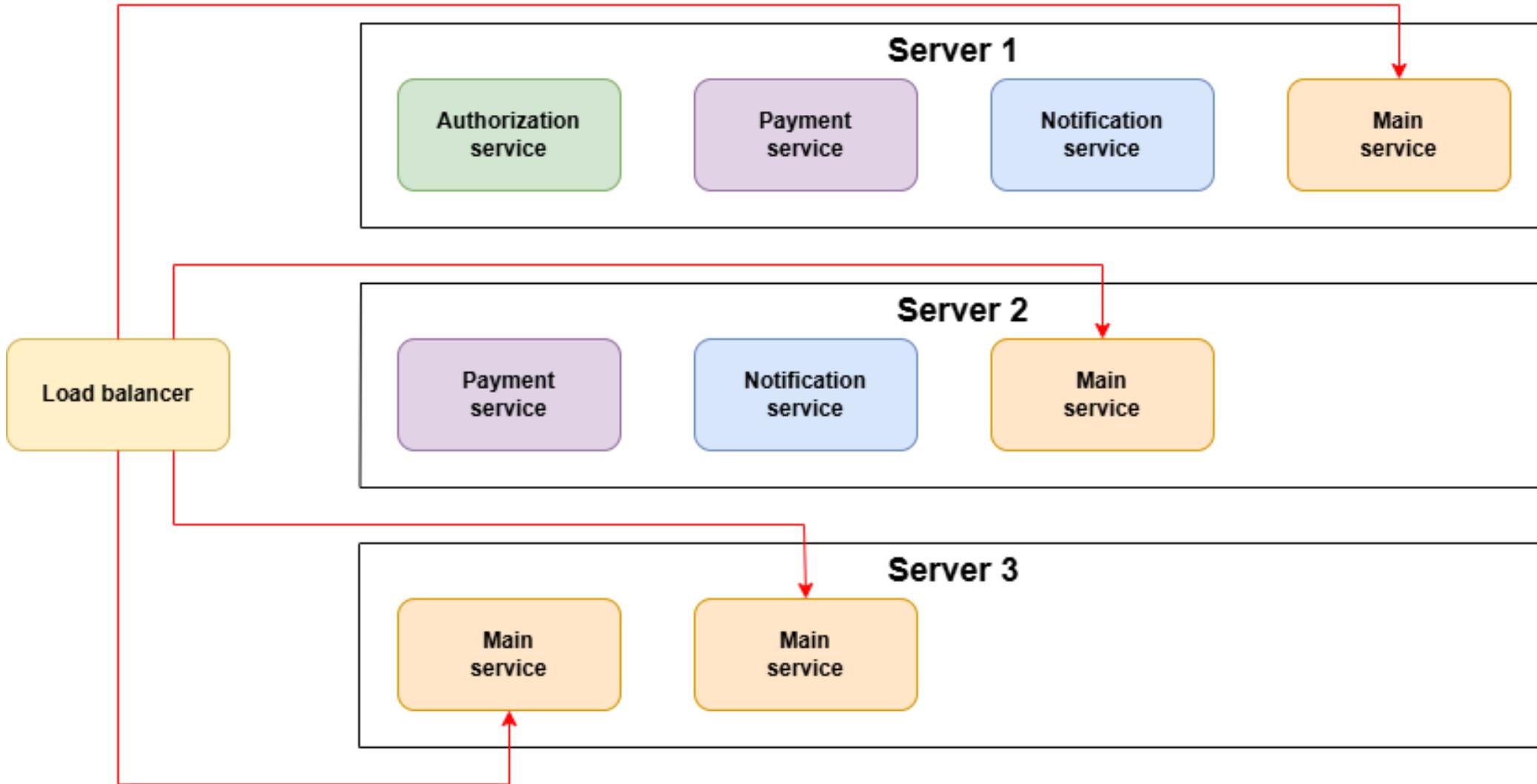
Масштабирование сервисов



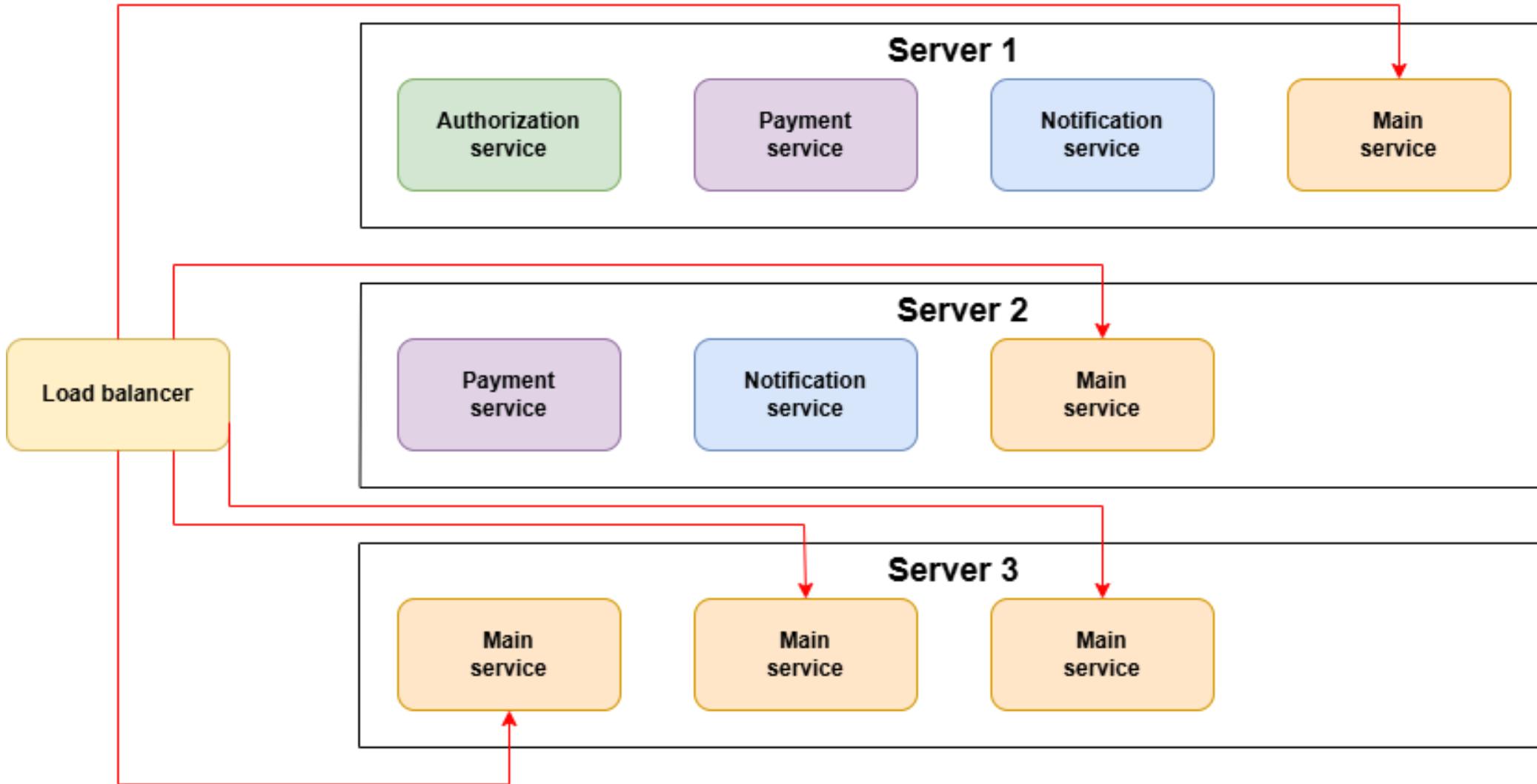
Горизонтальное масштабирование и распределение нагрузки



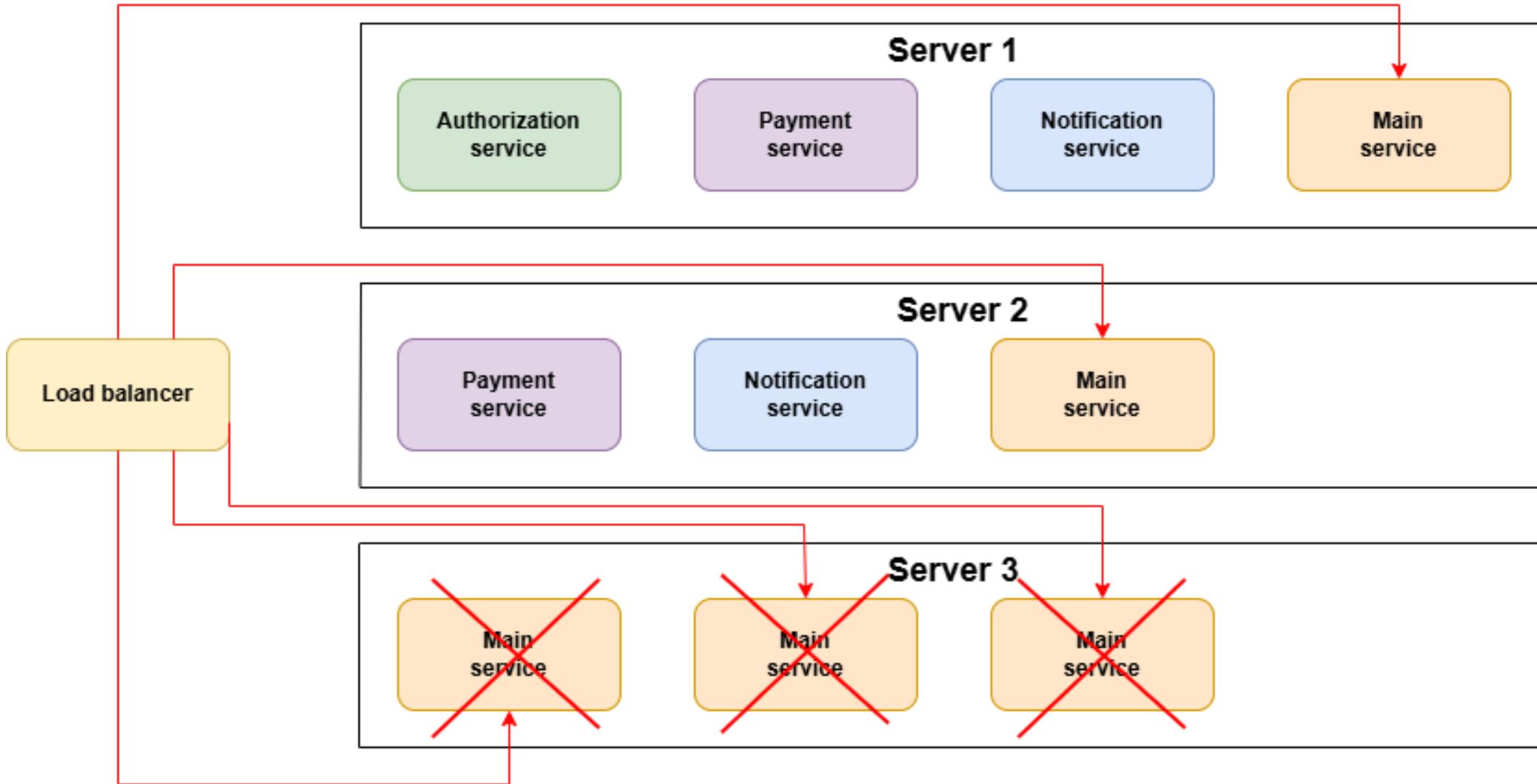
Балансирующий нагрузки



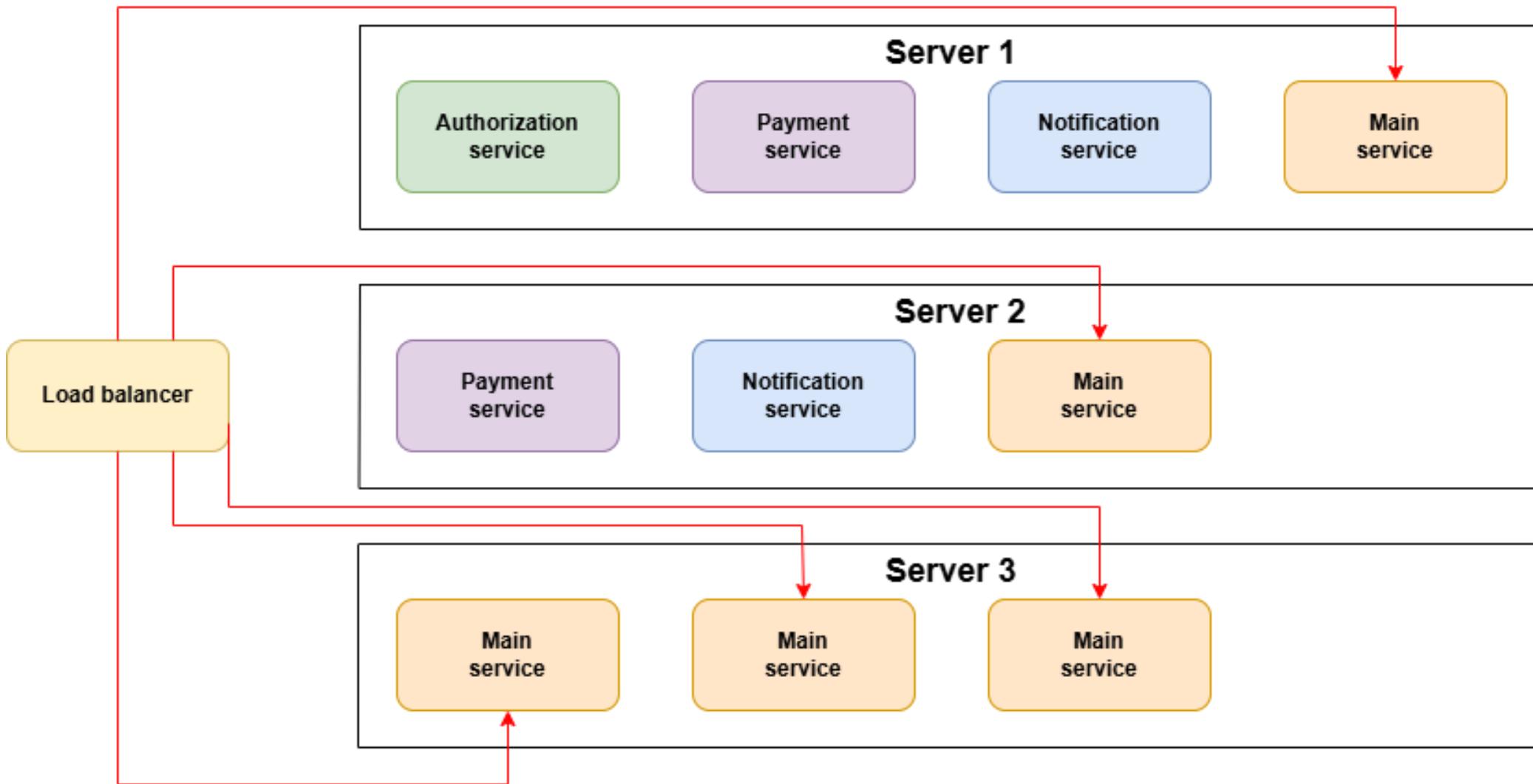
Добавление нового экземпляра сервиса



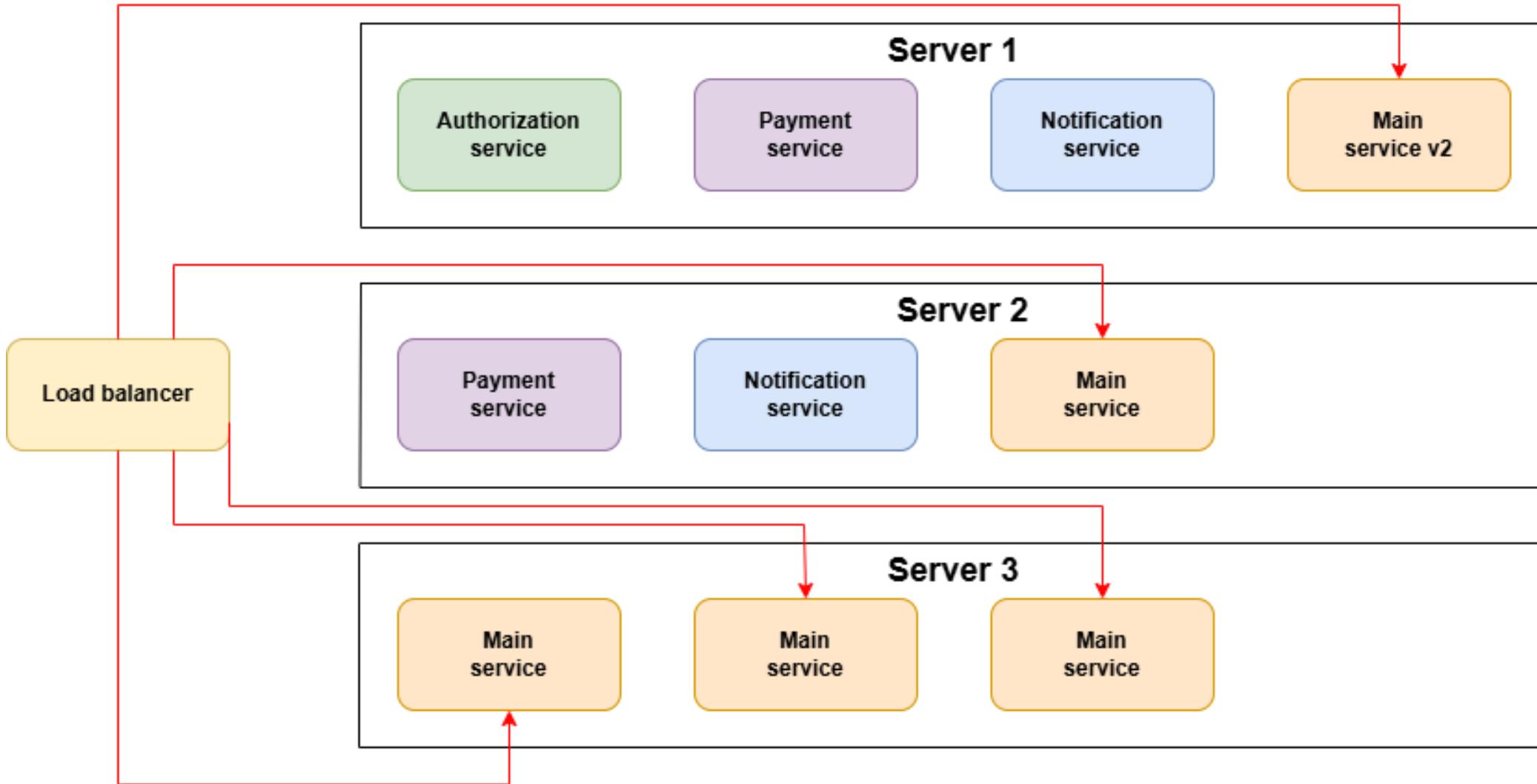
Падение сервисов на одном из серверов



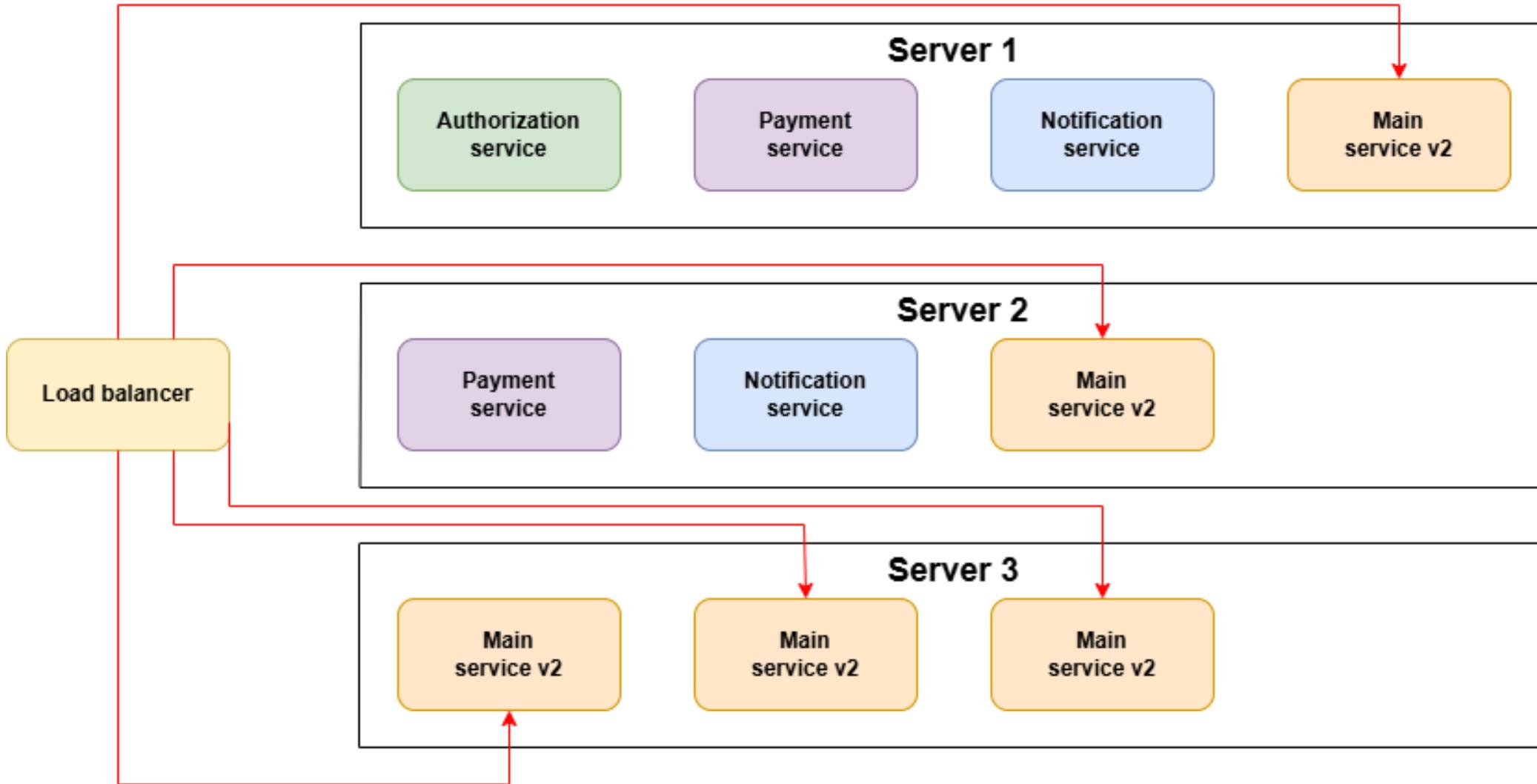
Поднятие сервисов



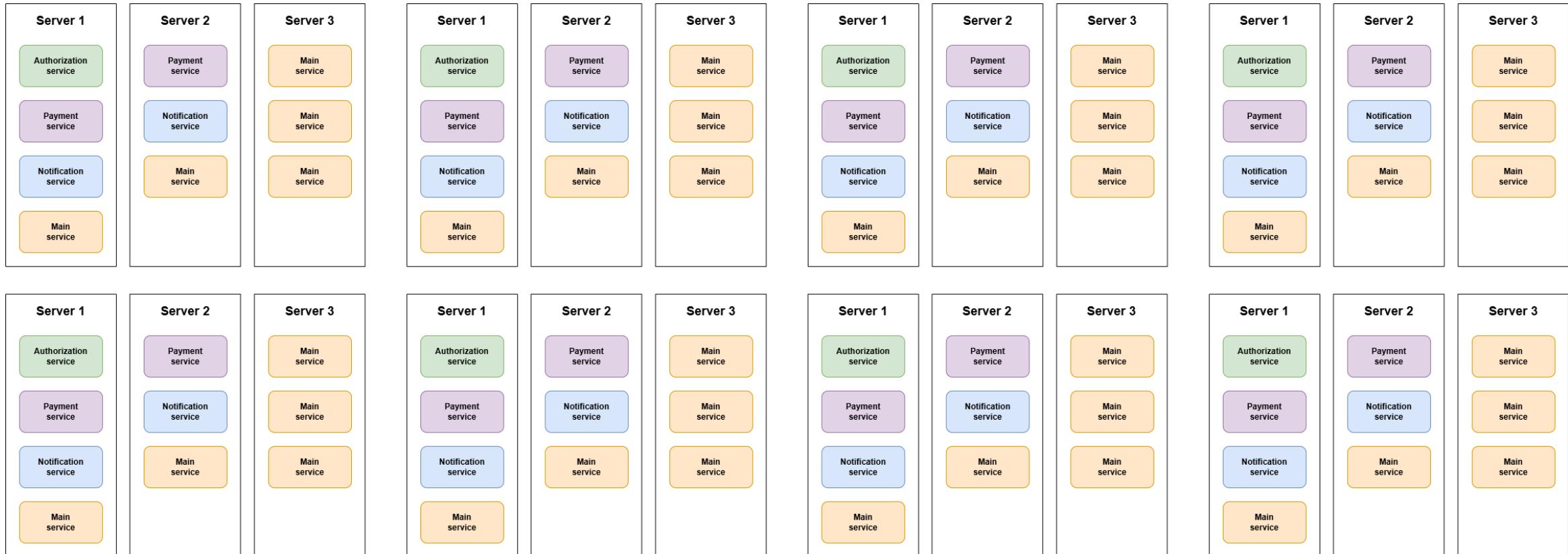
Обновление одного экземпляра сервиса



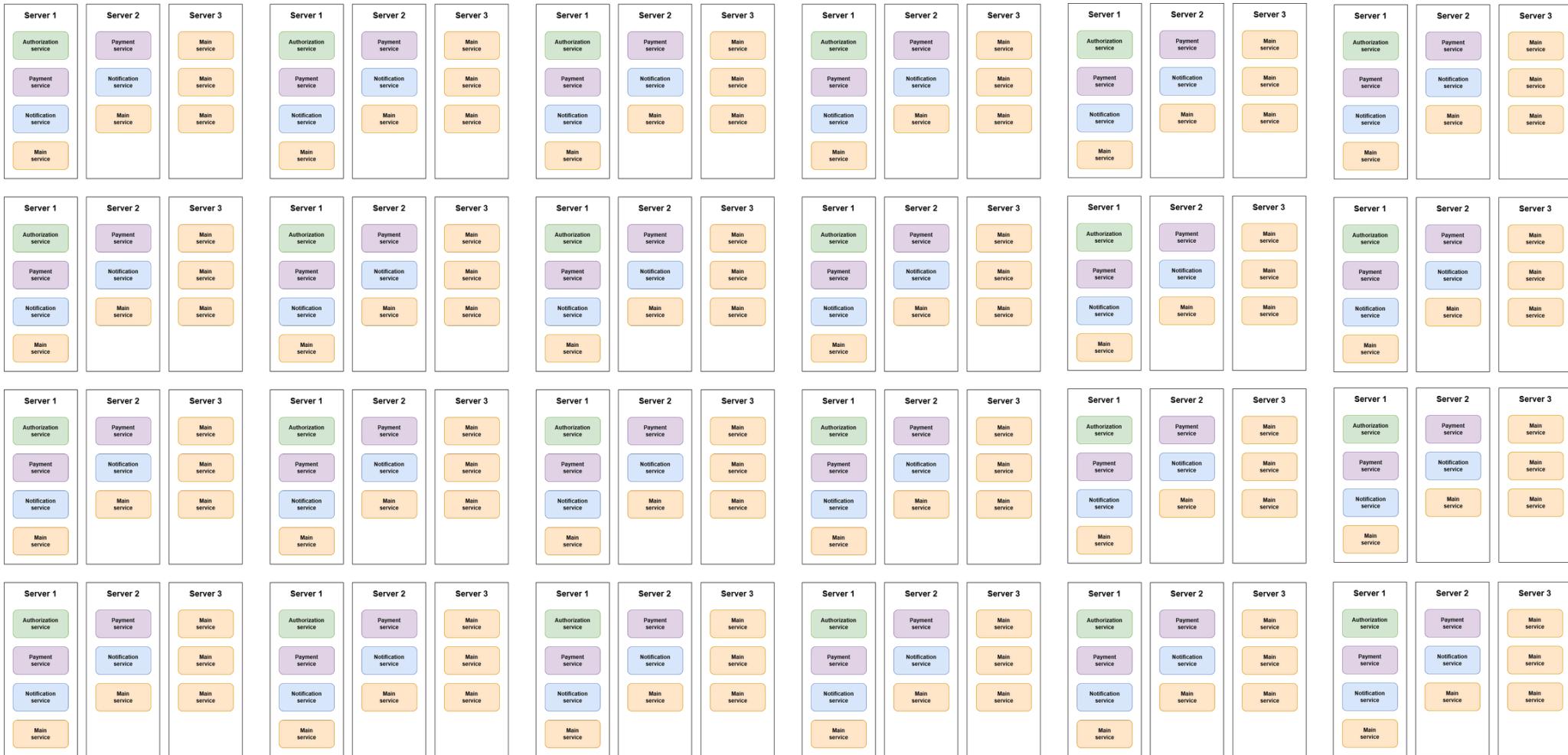
Обновление всех экземпляров сервиса



Сервисы в средней компании



Сервисы в крупной компании



Сложности микросервисов



Усложнение инфраструктуры

С ростом количества сервисов нужна автоматизация запуска, связи и управления ими.



Работа с большим количеством контейнеров

Требуется инструмент, который централизованно управляет контейнерами контейнерами на множестве машин.



Балансировка нагрузки

Нужно равномерно распределять трафик между живыми инстансами сервисов.



Сетевые проблемы

Адреса сервисов постоянно меняются - нужен автоматический поиск и маршрутизация.



Масштабирование

Сервисы должны уметь быстро увеличивать или уменьшать количество инстансов в зависимости от нагрузки.



Высокая доступность

Система должна восстанавливать сервисы при сбоях без участия человека.



Хранение конфигураций и секретов

Необходимо безопасно хранить и передавать сервисам конфиги, пароли и ключи.



Обеспечение непрерывной работы и обновлений

Обновления должны проходить без простоя, с возможностью отката и постепенного раската.

EP
APL
RLX
APL
X.US.TNX

RLX
ACB
EUX
TR
XG
TF
SA
BDM
ATF
BDZ

Что такое оркестрация?

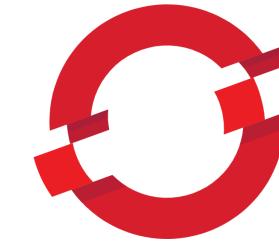
Оркестрация - это автоматизация развертывания, управления, масштабирования и мониторинга контейнерных приложений в кластере.



Инструменты оркестрации



kubernetes



OPENSHIFT

Что такое Kubernetes?

Kubernetes (часто сокращается до K8s) – это популярная платформа с открытым исходным кодом, предназначенная для автоматизации развертывания, масштабирования и управления контейнеризованными приложениями.



Факты о Kubernetes

- ✓ Изначально разработан в Google
- ✓ В середине 2014 опубликованы исходные коды проекта
- ✓ Изначально назывался *Project Seven*
- ✓ В 2015 Google передала проект в фонд Cloud Native Computing Foundation (CNCF)



Задачи Kubernetes



Deployment

Поднимает новые экземпляры приложения.



Масштабирование

Добавляет или уменьшает количество экземпляров приложения, в зависимости от нагрузки.



Самовосстановление

Следит за приложением, если упало, автоматически перезапускает его.



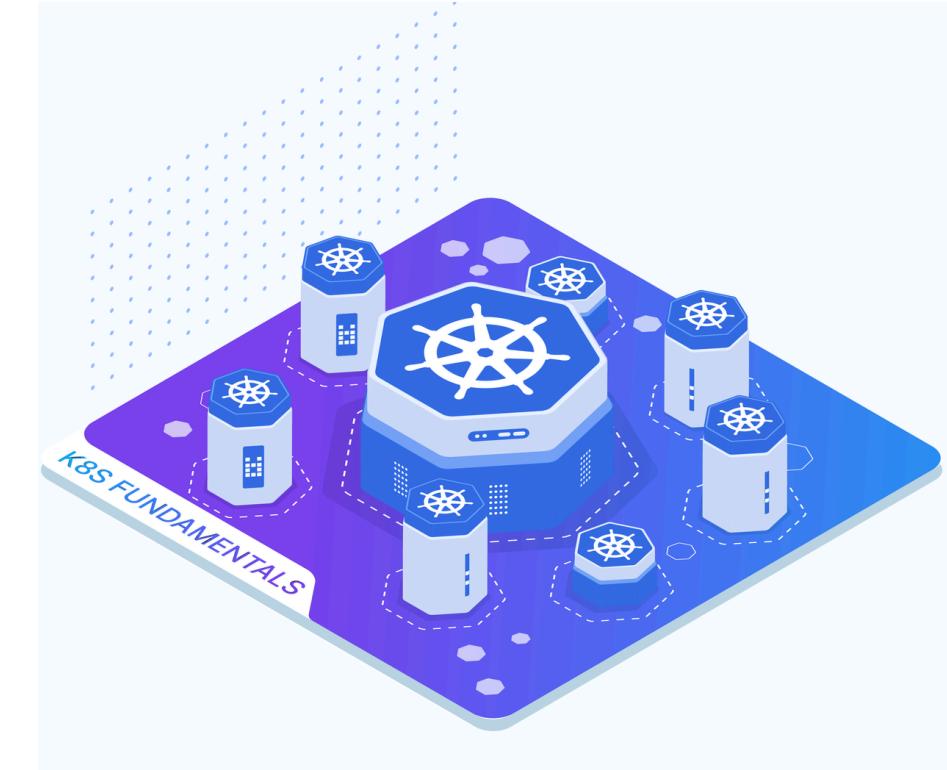
Балансировка трафика

Равномерно распределяет нагрузку, по экземплярам приложений.

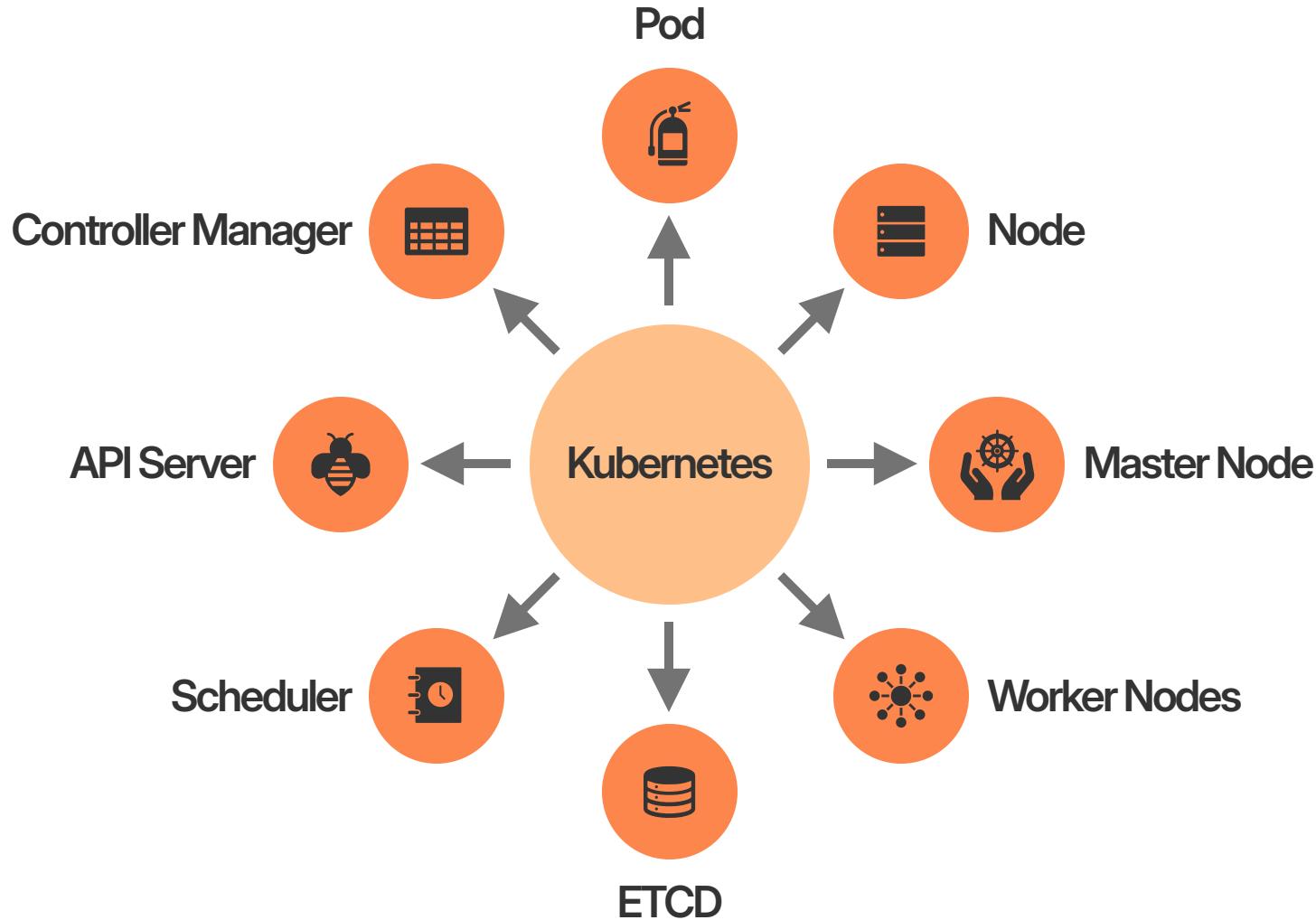


Управление

Контролирует жизненный цикл приложения.



Основные объекты кластера Kubernetes





Nodes (ноды, узлы)



Nodes (ноды, узлы)

Физические или виртуальные машины, на которых развертываются и запускаются контейнеры с приложениями. Совокупность нод образует кластер Kubernetes.



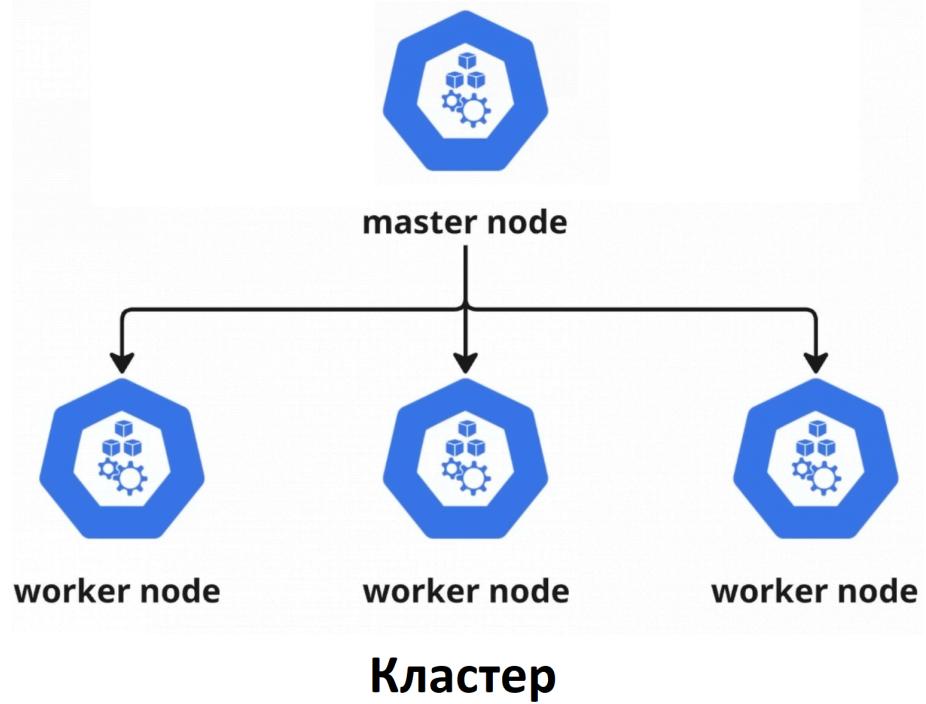
Master node

Узел, управляющий всем кластером. Следит за остальными нодами и распределяет между ними нагрузку.



Worker nodes

Узлы, на которых работают контейнеры. Чем больше рабочих узлов, тем больше приложений можно запустить.



Отказоустойчивость кластера Kubernetes



Отказоустойчивость кластера

1 мастер-нода не отказоустойчива, 3 - минимальное рекомендованное, 5 - повышенная устойчивость.



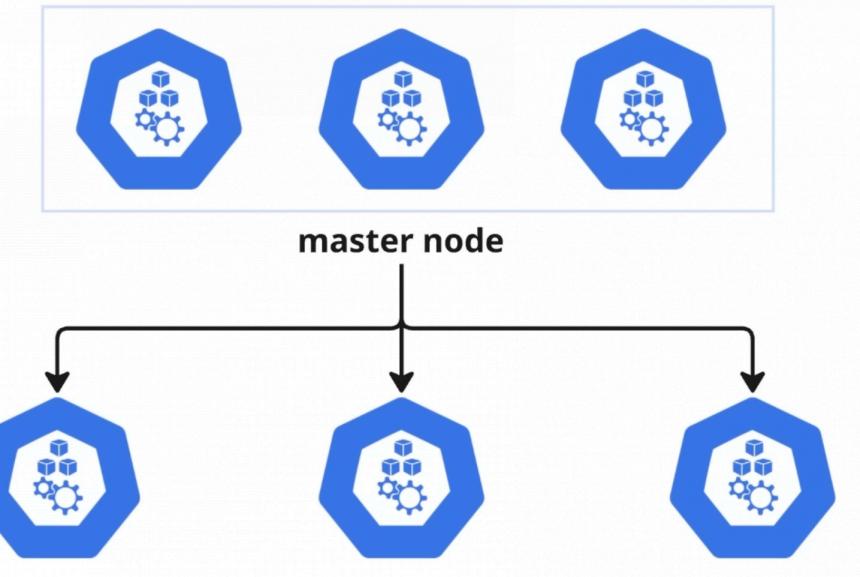
Кворум

Это большинство мастер-нод, которые должны быть доступны, чтобы кластер считался живым и мог принимать решения. С 3-мя мастер-нодами кворум = 2, с 5-ю = 3.



Нечетное кол-во нод

При четном кол-ве кластер теряет кворум при потере половины нод.



Кластер

Pods (Поды)

● Pod

Минимальная единица развертывания в Kubernetes. Один под может содержать один или несколько контейнеров.

● Связь контейнеров

Все контейнеры внутри одного пода используют общий IP-адрес и локальные порты, что позволяет им взаимодействовать максимально быстро и напрямую.

● Управление

Kubernetes не управляет контейнерами напрямую, а работает с подами как с единственным объектом, т.е. Kubernetes создает, перезапускает, масштабирует именно поды, а не отдельные контейнеры.

● Размещение

Поды запускаются на нодах кластера. Если нода выходит из строя, Kubernetes пересоздаёт под на другой доступной ноде. Это обеспечивает гибкость и отказоустойчивость.

Нода № 1

Под № 1



Под № 2



Под № 3



Контейнеры

Компоненты в master node

● ETCD

Это база данных формата «ключ - значение», которая содержит текущую конфигурацию и состояние кластера Kubernetes.

● Scheduler

Определяет, на какой из worker-node, будет деплоится приложение (т.е. решает на какой worker-node должен работать под, когда Kubernetes создает его). Scheduler работает с подами, он только выбирает ноду, но не запускает контейнеры. А уже Kubelet на выбранной ноде создаёт и запускает контейнеры внутри пода. А уже внутри контейнеров наши микросервисы.

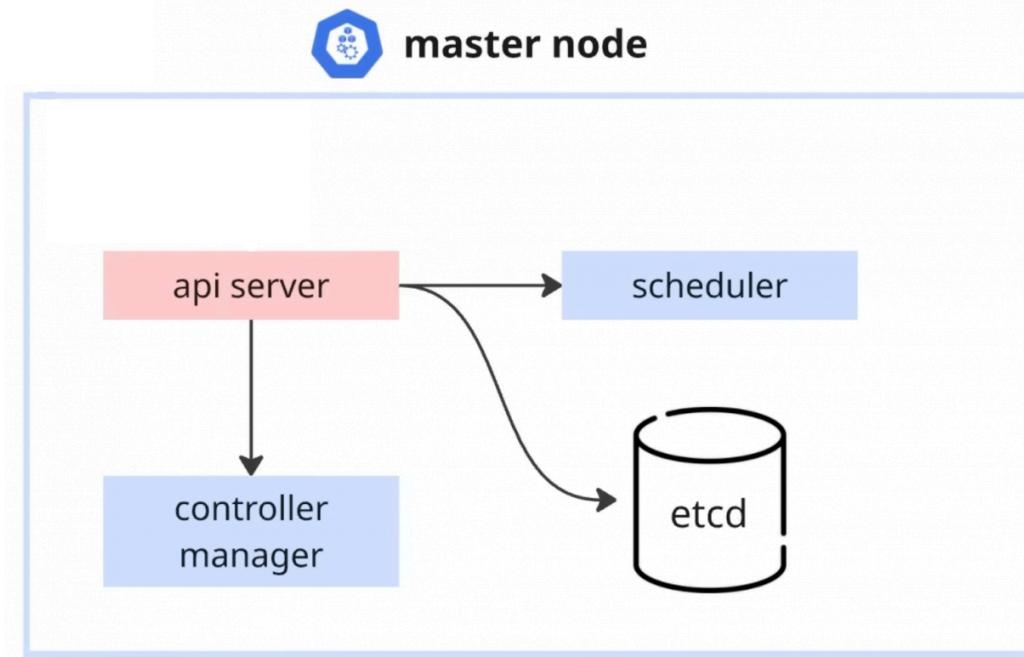
● API server

Сердце управления Kubernetes, единственная точка входа в кубер, представляет интерфейс в стиле REST. Он

- > принимает все запросы (создать под, удалить сервис, получить логи и др),
- > валидирует запросы, записывает состояние в etcd,
- > говорит остальным компонентам (scheduler, controller-manager), что нужно сделать.

● Controller manager

Следит за тем, чтобы все, что попросили, было сделано. Запускает разные контроллеры, которые следят, чтобы фактическое состояние кластера совпадало с желаемым.



Компоненты в worker node

- **Pods**

Минимальная единица развертывания в Kubernetes. Один под может содержать один или несколько контейнеров.

- **Containers**

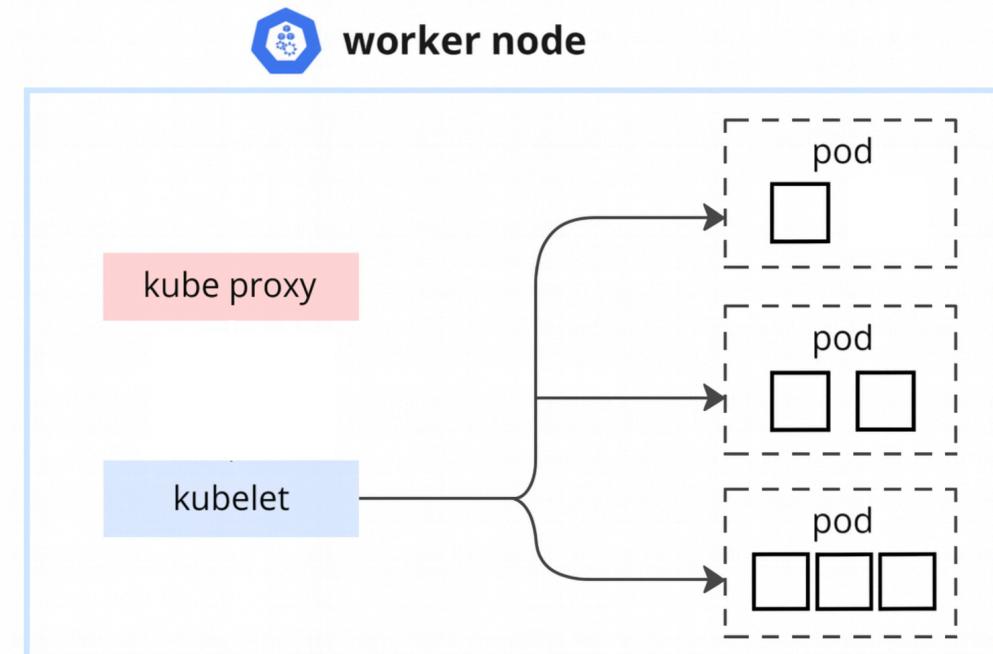
Это изолированные процессы с приложением и всеми его зависимостями, которые запускаются внутри пода.

- **Kubelet**

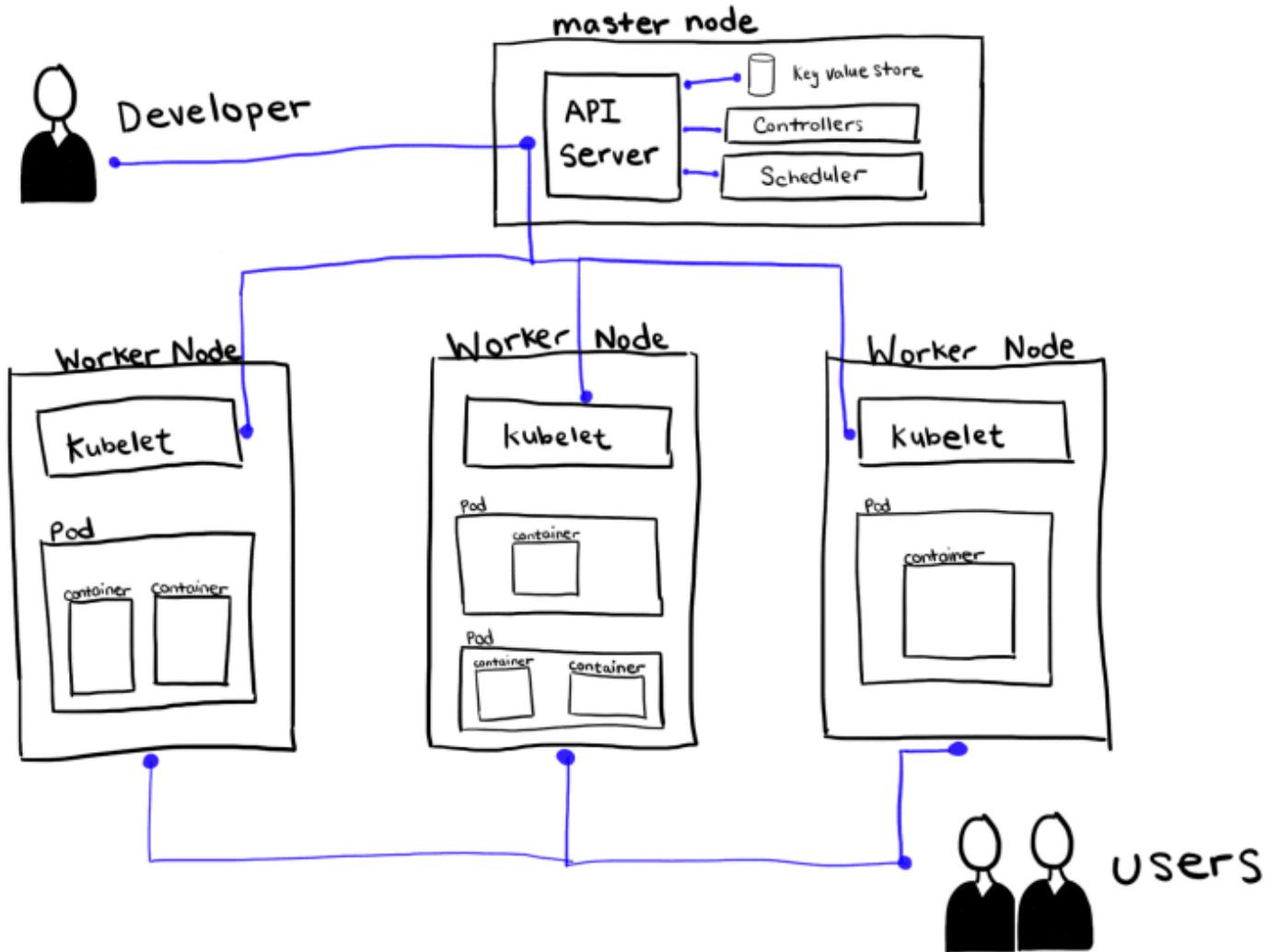
Агент, управляет состоянием ноды и следит за подами и контейнерами: запускает, перезапускает при падении, проверяет их состояние. Получает инструкции от мастер-ноды, через API-сервер и исполняет их.

- **Kube-proxy**

Компонент, который перехватывает запросы к подам и направляет их на нужный контейнер. Обеспечивает, чтобы трафик внутри кластера корректно доходил до нужного пода.



Кластер Kubernetes



CI/CD

A cloud of various technology-related acronyms and terms such as EP, APL, XUS, TTX, RLX, CDC, AGI, XAI, XG, TR, EUX, XTF, EX, SA, T, I, E, X, BDM, ATF, BDZ, CI/CD, and many more, all rendered in different fonts and sizes.



Как командам выпускать код чаще, не нарушая продакшена?

Проблема

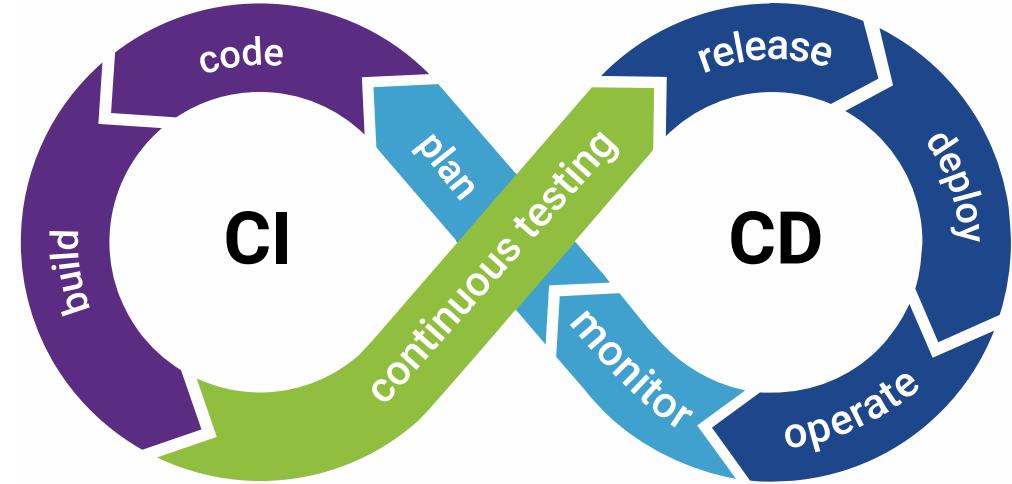


Что такое CI/CD?

CI/CD - это подход к разработке, при котором автоматизируются процессы разработки, тестирования и доставки программного обеспечения.

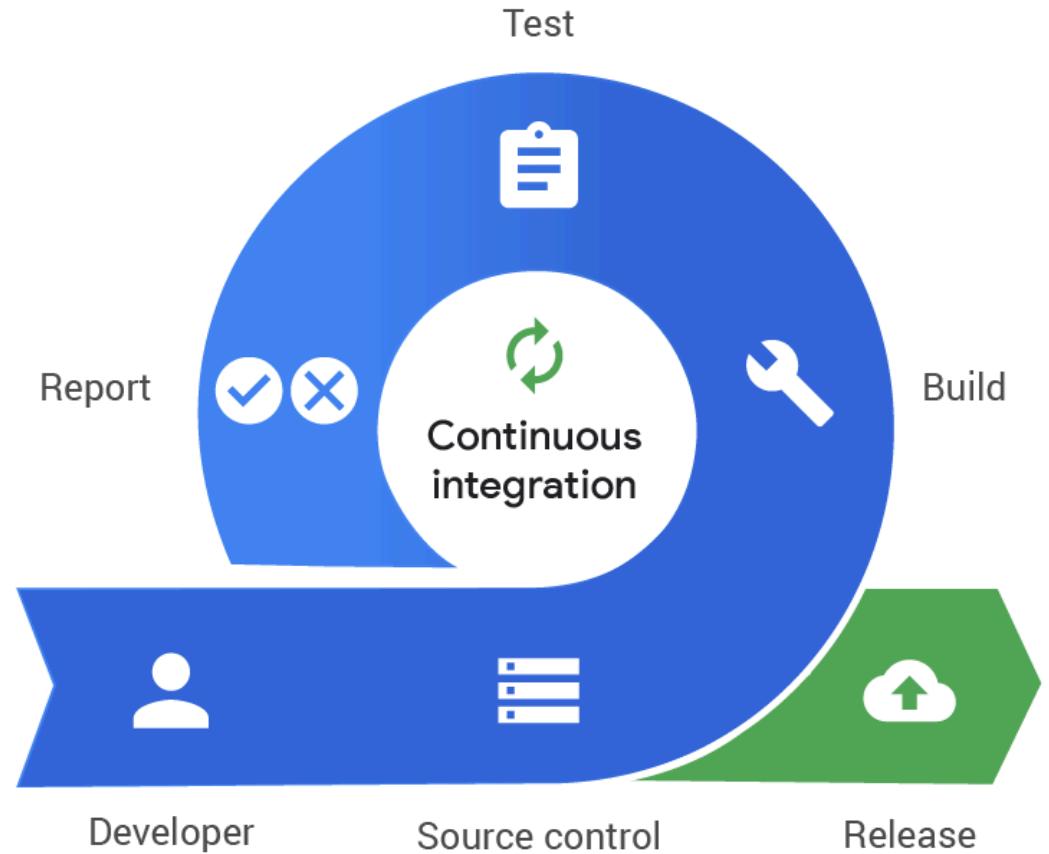
CI - Continuous Integration (непрерывная интеграция).

CD - Continuous Delivery (непрерывная поставка) или Continuous Deployment (непрерывное развертывание).



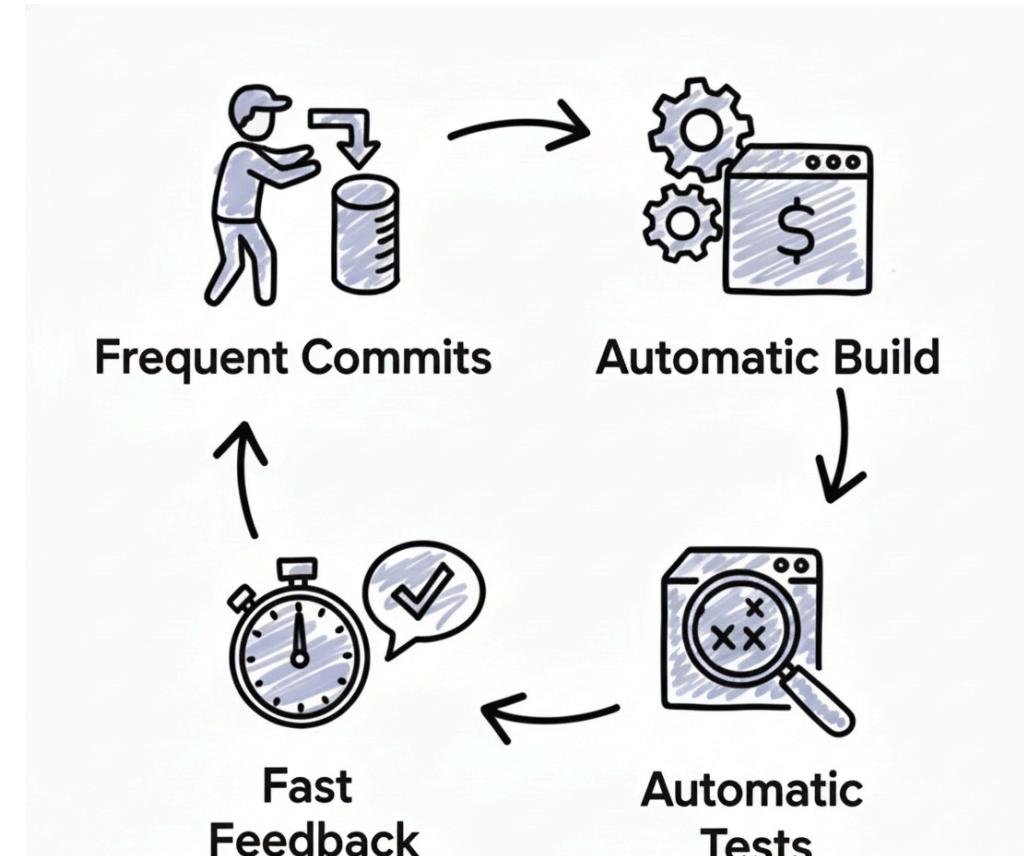
Continuous Integration

Continuous Integration (непрерывная интеграция) – методология, при которой в код вносятся небольшие изменения с частыми коммитами в общий репозиторий



Цикл Continuous Integration

- ✓ Частые коммиты
- ✓ Автоматическая сборка
- ✓ Автоматические тесты
- ✓ Быстрая обратная связь

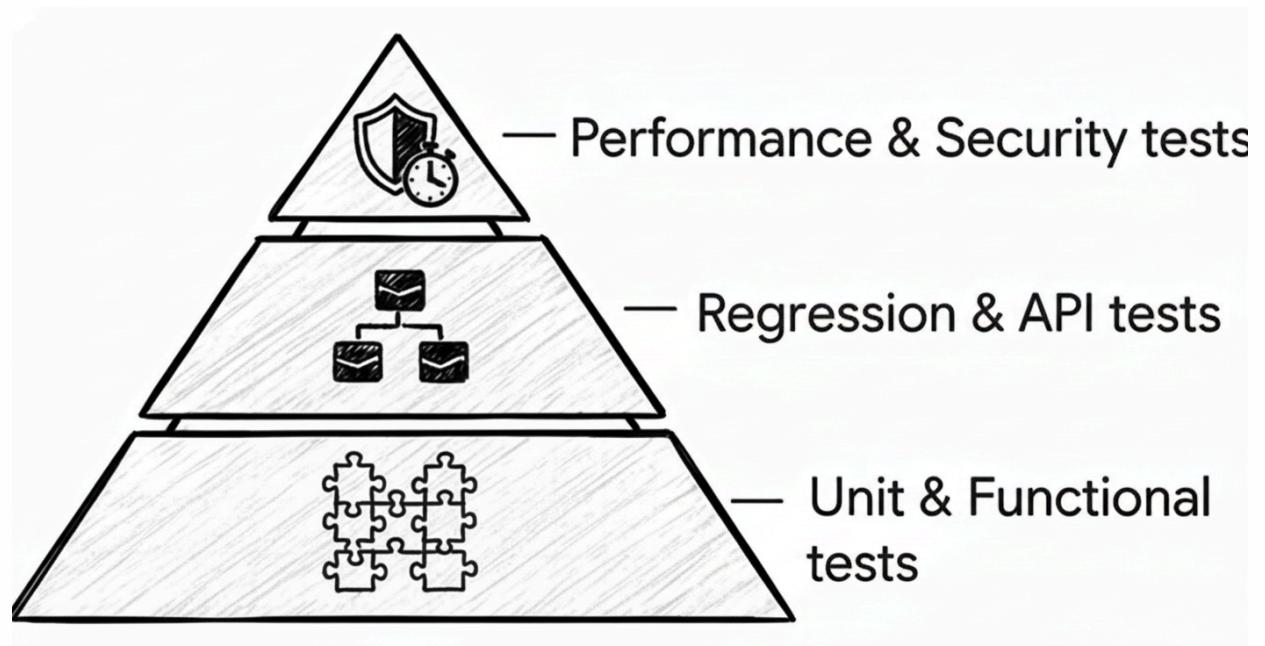


Преимущества Continuous Integration

- ✓ **Раннее обнаружение ошибок**
Каждый коммит автоматически проходит сборку и тесты. Ошибки выявляются сразу, а не на этапе релиза - исправлять их дешевле и легче.
- ✓ **Стабильная главная ветка (main/master)**
CI гарантирует, что в основную ветку попадает только рабочий, протестированный код. Это снижает риск «сломать прод» после очередного merge.
- ✓ **Быстрая обратная связь для разработчиков**
Разработчик мгновенно узнаёт: прошли ли тесты, сборка удалась или нет, что именно упало. Это ускоряет итерации и улучшает качество кода.
- ✓ **Более быстрый и частый выпуск фич**
Так как интеграция идёт постоянно и без ручных шагов, команда может быстрее двигаться к Release.

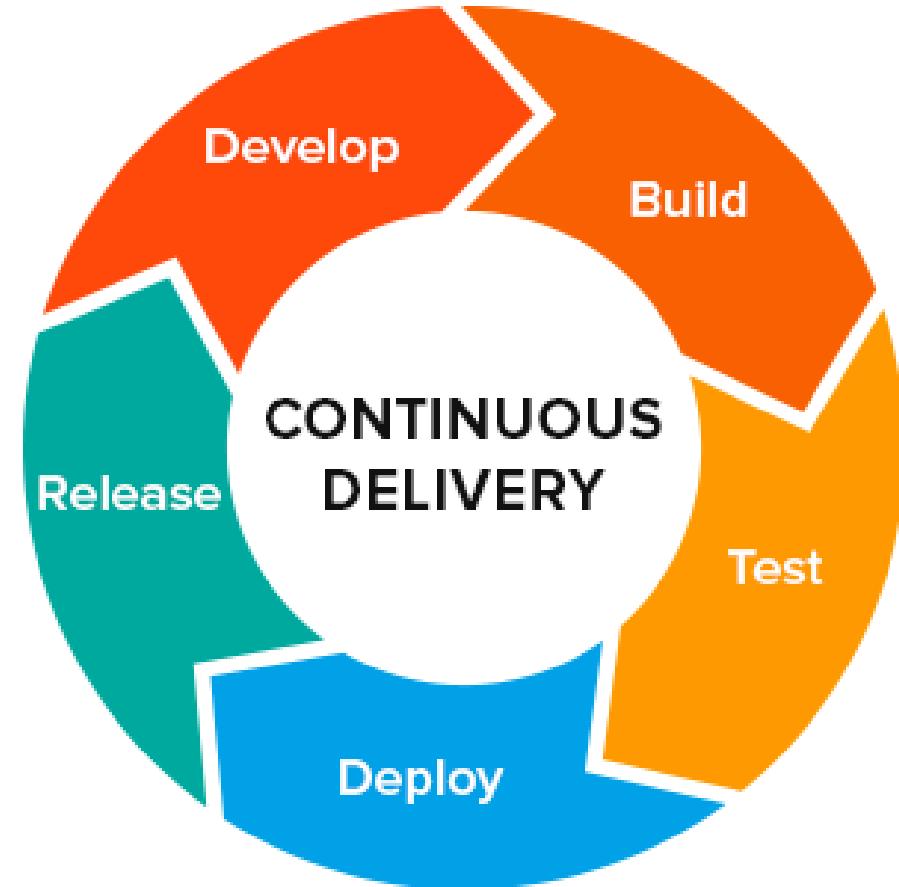
Непрерывное тестирование

- ✓ Unit & functional tests
- ✓ Regression & API tests
- ✓ Performance & security tests



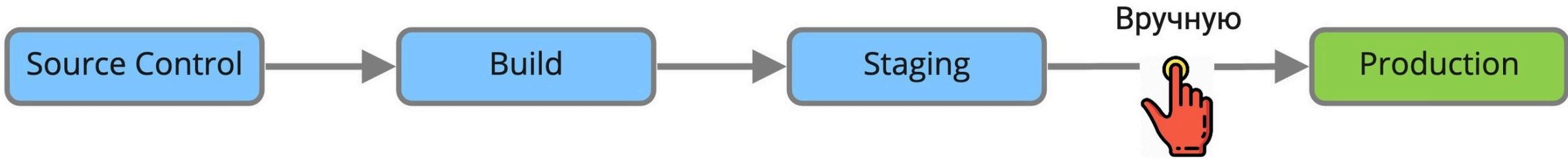
Continuous Delivery / Continuous Deployment

Continuous Delivery / Continuous Deployment (непрерывная поставка / непрерывное развертывание) – это автоматизированный процесс доставки и развертывания проверенного кода в разные окружения, включая production.

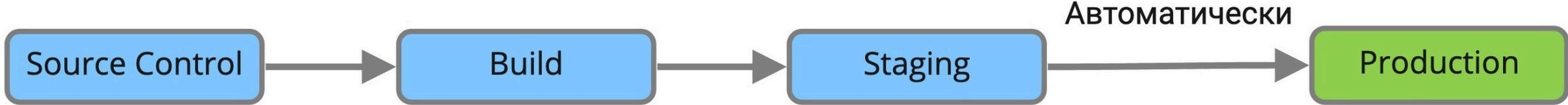


Continuous Delivery vs Continuous Deployment: в чем разница?

Continuous Delivery



Continuous Deployment



Delivery = автоматизация + ручной релиз

Deployment = автоматизация + автоматический релиз

Преимущества Continuous Delivery / Continuous Deployment

✓ Более стабильные и предсказуемые релизы

Автоматизация сборки, тестирования и развёртывания снижает риск ошибок и делает процесс поставки повторяемым.

✓ Быстрая проверка изменений в реальных окружениях

Код автоматически разворачивается в тестовой среде, что помогает выявлять проблемы до попадания в продакшен.

✓ Меньше ручной работы и человеческого фактора

Сокращает количество ручных операций, сборка, тесты, деплой занимают меньше времени и сил.

✓ Ускорение выпуска новых фич и улучшений

Разработчики могут итеративно выпускать изменения небольшими порциями, быстрее реагируя на требования бизнеса.

✓ Максимально быстрая доставка кода в продакшен (**Deployment**)

Каждый успешный коммит автоматически выкатывается без участия человека.

✓ Мгновенная обратная связь от реальных пользователей (**Deployment**)

Разработчики сразу видят, как работает новая функциональность в продакшене.

От коммита до продакшена



Вопросы

