

# Варианты развертывания приложения в продакшн (часть Continuous Deployment)

Если клиенты уже пользуются вашим приложением, то вы должны подумать, как вам реализовывать каждое обновление. Ведь может быть итак, что клиенты заходят и видят надпись "Ведутся технические работы. Вернитесь через два часа". Это именно из-за того что владельцы продукта решили определённым образом обновлять приложение.

Вот все варианты развёртывания:

## 1. Развертывание на месте (In-Place Deployment):

- **Реализация:** Системные администраторы или инженеры по развертыванию обновляют код приложения непосредственно на работающем сервере. Это может включать в себя обновление зависимостей, выполнение миграций базы данных и перезапуск сервисов.
- **Пример:** Администратор может зайти на сервер через SSH и выполнить команды обновления.
- **Плюсы:** Минимизация использования ресурсов; более быстрое развертывание.
- **Минусы:** Скорей всего ваше приложение будет недоступно какое-то время; риск повреждения текущей среды.

## 2. Скользящее развертывание (Rolling Deployment):

- **Реализация:** Новая версия постепенно раскатывается на отдельные экземпляры, пока все экземпляры не будут обновлены. Балансировщик нагрузки направляет трафик только на активные экземпляры.
- **Пример:** Представьте кластер из 10 серверов. Один за другим, каждый сервер останавливается для обновления, в это время весь трафик перенаправляется на остальные активные серверы.
- **Плюсы:** Уменьшение недоступности; гибкость в управлении версиями.
- **Минусы:** Риск несовместимости версий; возможность ошибок во время перехода.

## 3. Сине-зеленое развертывание (Blue-Green Deployment):

- **Реализация:** При этом методе создают две идентичные среды — "синюю" (старую версию) и "зеленую" (новую версию). После успешного тестирования на пред-продакшн(еще называют стейдж stage среда) новой версии, продакшн трафик

перенаправляется на зеленую среду с использованием балансировщика нагрузки или изменения записей DNS.

- **Пример:** Имеется синяя среда с текущей версией и зеленая среда с новой версией. После проверки зеленой среды, балансировщик нагрузки или DNS переключают трафик на зеленую среду.
- **Плюсы:** Минимизация недоступности; возможность быстрого отката.
- **Минусы:** Потребление больше ресурсов; сложность управления двумя средами.

#### 4. Красно-черное развертывание (Red-Black Deployment):

- **Реализация:** Схоже с сине-зеленым, но переключение трафика происходит мгновенно. Это обычно достигается путем изменения конфигурации балансировщика нагрузки.
- **Пример:** Красная среда служит текущей версией, а черная – новой. После успешного тестирования и подготовки, администратор мгновенно переключает балансировщик нагрузки с красной среды на черную.
- **Плюсы:** Нет недоступности в принципе; быстрый откат.
- **Минусы:** Требуется больше ресурсов; возможность ошибок при мгновенном переключении.

#### 5. Канареечное Развертывание (Canary Release):

- **Реализация:** При канареечном развертывании, новая версия (канарейка) вводится в эксплуатацию постепенно. Начинается с того, что лишь малая часть пользователей направляется на новую версию, позволяя мониторить поведение системы и получать отзывы от реальных пользователей. По мере уверенности в стабильности новой версии, процент трафика, направляемый на канарейку, увеличивается, пока все пользователи не будут переведены на новую версию. Используется балансировщик нагрузки для управления распределением трафика между старой и новой версиями.
- **Пример:** Есть старая версия приложения и новая версия, или "канарейка". Сначала лишь небольшой процент пользователей перенаправляется на канарейку, позволяя наблюдать за работой новой версии в реальных условиях. По мере положительной обратной связи и отсутствия критических ошибок, все больше пользователей направляется на новую версию, пока полностью не завершится переход.

- **Плюсы:** Риски минимизированы, поскольку новая версия вводится постепенно; быстрый откат; возможность получения быстрой обратной связи от клиентов (проведение тестов какой-то функциональности).
- **Минусы:** Сложности управления и мониторинга двух версий одновременно; возможность ошибок при мгновенном переключении.

## 6. Неизменяемое развертывание (Immutable Deployment):

- **Реализация:** При каждом обновлении создается новый сервер или контейнер с новой версией приложения, а старые серверы выключаются после успешного развертывания.
- **Пример:** Вместо обновления текущих серверов или контейнеров, для каждого обновления создаются новые экземпляры. После проверки нового экземпляра трафик перенаправляется на новые серверы, и старые серверы отключаются.
- **Плюсы:** Предотвращение конфигурационных ошибок; стабильность и надежность.
- **Минусы:** Большие затраты ресурсов; сложность управления множеством серверов.

Во всех методах важную роль играет балансировщик нагрузки, который эффективно распределяет входящий сетевой трафик по нескольким серверам, и изменение DNS, чтобы перенаправить трафик с одного сервера на другой или с одной среды на другую.