DOI: 10.58168/ModInfSys2026_32-39

УДК 004.413

СРАВНИТЕЛЬНЫЙ АНАЛИЗ СТРАТЕГИЙ РАЗВЁРТЫВАНИЯ МИКРОСЕРВИСОВ: ROLLING UPDATE, BLUE-GREEN И CANARY

В.Э. Буздин, Е.А. Аникеев

ФГБОУ ВО «Воронежский государственный лесотехнический университет имени Г.Ф. Морозова»

Аннотация. Проведён сравнительный анализ трёх стратегий развёртывания микросервисов — последовательной замены, переключения окружений и канареечного выката. Предложена модель риска выката, оценивающая ожидаемую долю затронутых дефектом запросов. По данным учебного стенда стратегии сопоставлены по доле пользователей под риском, времени отката и избыточности ресурсов. Показано, что канареечный выкат даёт наименьший радиус поражения при умеренных затратах.

Ключевые слова: развёртывание, микросервисы, canary, blue-green, rolling update, выкат, откат, риск, надёжность поставки, обработка информации.

COMPARATIVE ANALYSIS OF MICROSERVICE DEPLOYMENT STRATEGIES: ROLLING UPDATE, BLUE-GREEN AND CANARY

V.E. Buzdin, E.A. Anikeev

Voronezh State University of Forestry and Technologies named after G.F. Morozov

Abstract. A comparative analysis of three microservice deployment strategies — rolling update, environment switching (blue-green), and canary release — is carried out. A rollout risk model is proposed that estimates the expected share of requests affected by a defect. Using a test bench, the strategies are compared by the share of users at risk, rollback time, and resource overhead. It is shown that the canary release provides the smallest blast radius at moderate cost.

Keywords: deployment, microservices, canary, blue-green, rolling update, rollout, rollback, risk, delivery reliability, information processing.

© Буздин В. Э., Аникеев Е. А., 2026

Введение

Частые выпуски — одно из главных преимуществ микросервисной архитектуры: каждый сервис можно обновлять отдельно и многократно в течение дня. Но обратной стороной частоты становится риск: каждый выкат — это потенциальный источник сбоя в работающей системе, поэтому контроль качества переносится и на этап выпуска [2]. Поэтому важно не только как часто обновляется сервис, но и как именно новая версия выводится в эксплуатацию, то есть какая применяется стратегия развёртывания [1].

На практике распространены три подхода. Последовательная замена (rolling update) постепенно заменяет старые экземпляры новыми. Переключение окружений (blue-green) держит два параллельных окружения и в нужный момент переводит трафик с одного на другое. Канареечный выкат (canary) сначала направляет на новую версию малую долю трафика и наращивает её, только убедившись в отсутствии проблем; управление долями трафика при этом опирается на механизмы балансировки нагрузки [5]. Эти стратегии по-разному распределяют риск во времени и по-разному расходуют ресурсы, однако выбор между ними часто делается по привычке, а не по расчёту.

Цель работы — сравнить три стратегии развёртывания на единой основе. Для этого предложена модель риска выката, оценивающая, какая доля запросов пострадает, если новая версия окажется дефектной, и проведено количественное сравнение стратегий по этому и сопутствующим показателям. Вопросы планирования и поэтапного выполнения работ в изменяющихся условиях, близкие к управлению выкатом, рассматриваются в [4], а роль непрерывного мониторинга для своевременного решения — в [6].

Стратегии развёртывания и критерии сравнения

Механизмы трёх стратегий показаны на рисунке 1. При последовательной замене экземпляры обновляются по одному или небольшими группами; в каждый момент часть трафика обслуживается уже новой версией, поэтому при дефекте под удар попадает та доля пользователей, что пришлась на обновлённые экземпляры. При переключении окружений новая версия заранее развёрнута и проверена в резервном окружении, а трафик переводится целиком; это обеспечивает мгновенный откат (возврат на прежнее окружение), но в момент после переключения дефект затрагивает сразу всех. Канареечный выкат вводит новую

версию малыми долями (например, 10 %, затем 25 %, 50 %, 100 %), что ограничивает радиус поражения, но растягивает выкат во времени.

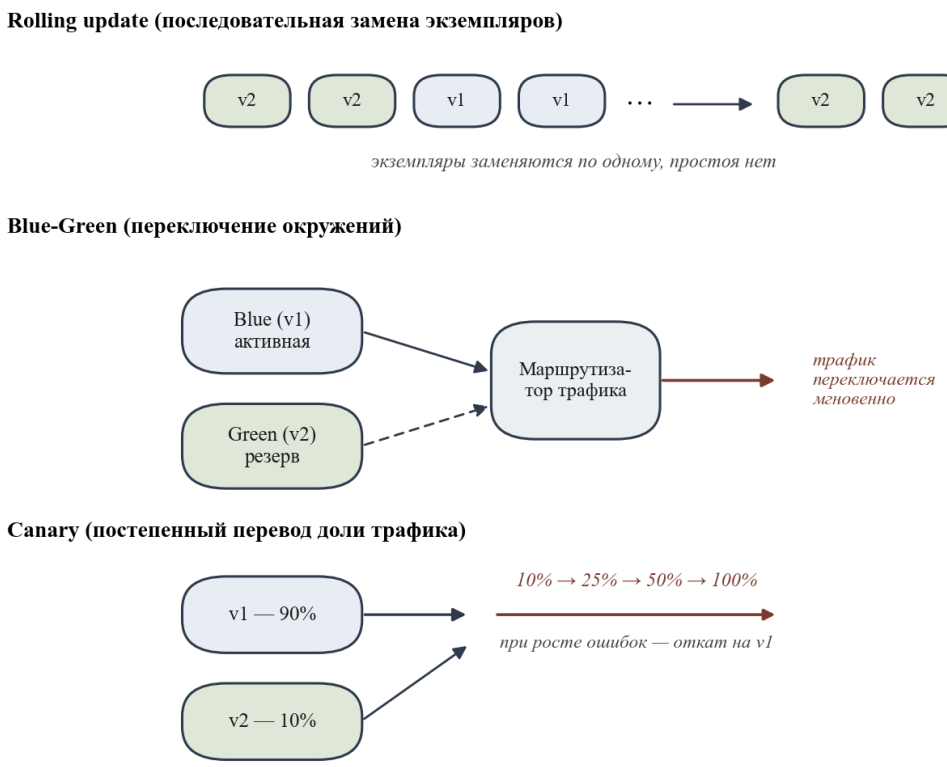


Рисунок 1 – Механизмы стратегий развёртывания: последовательная замена, переключение окружений, канареечный выкат

Для сравнения выбраны четыре критерия, важных при эксплуатации. Первый — доля пользователей под риском при сбое: какая часть запросов пострадает, прежде чем дефект будет обнаружен и устранён. Второй — время отката: насколько быстро можно вернуться к исправной версии. Третий — избыточность ресурсов: сколько дополнительных вычислительных мощностей требует стратегия во время выката. Четвёртый — сложность и предсказуемость процедуры. Первые три поддаются числовой оценке, четвёртый оценивается качественно.

Модель риска выката

Ключевой характеристикой выката примем радиус поражения — ожидаемое число запросов, обслуженных дефектной версией до момента обнаружения и отката. Если выкат разбит на этапы, на каждом из которых на новую версию приходится доля трафика s_k в течение времени τ_k, то при интенсивности запросов λ радиус поражения оценивается выражением:

$$A = \lambda \cdot \Sigma_k \, s_k \cdot \tau_k \qquad (1)$$

Выражение (1) имеет наглядный смысл: это площадь под графиком доли трафика, направленного на новую версию, до момента отката, умноженная на интенсивность запросов. Стратегия выгодна тогда, когда удерживает эту площадь малой. У переключения окружений доля s_k сразу равна единице, поэтому радиус поражения определяется только временем обнаружения дефекта. У канареечного выката доля наращивается постепенно, и если дефект проявляется на ранних этапах с малой s_k, площадь, а значит и радиус поражения, оказывается небольшой.

Отсюда видно, почему для канареечного выката критично быстрое обнаружение проблем: чем раньше по контролируемым метрикам (доля ошибок, задержка) замечен дефект на этапе с малой долей трафика, тем меньше слагаемое в сумме (1) и тем меньше пострадавших. Поэтому канареечный выкат разумно сочетать с автоматическим контролем метрик и правилом отката.

Количественное сравнение

Стратегии сравнивались на учебном стенде из контейнерного кластера с балансировщиком. Развёртывался сервис из четырёх реплик; дефектная версия вносилась намеренно, а время обнаружения определялось по росту доли ошибочных ответов. Результаты по трём числовым критериям приведены на рисунке 2, а сводное сравнение, включая качественные признаки, — в таблице 1. Переключение окружений показало наименьшее время отката (около 5 с), поскольку возврат сводится к обратному переключению трафика, но наибольший радиус поражения (до 100 % запросов в момент после переключения) и наибольшую избыточность — фактически требуется второе полное окружение. Последовательная замена дала средние значения по всем показателям: умеренную избыточность и умеренный радиус поражения, но самое долгое возвращение (около 90 с), так как откат требует обратной замены экземпляров. Канареечный выкат обеспечил наименьшую долю пользователей под риском (около 10 %) при небольшой избыточности, ценой более длительной процедуры выката.

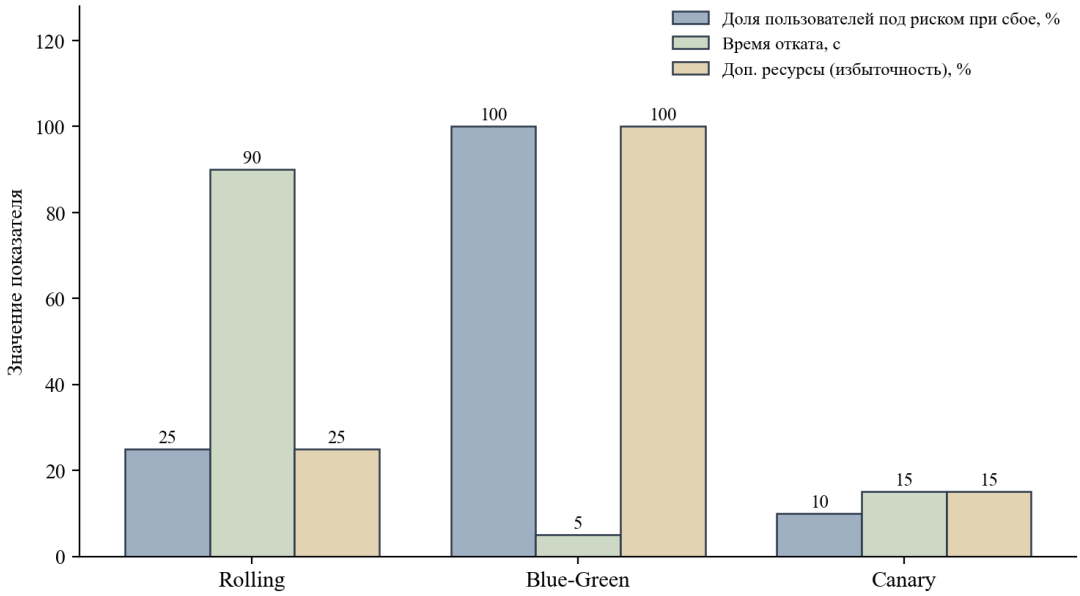


Рисунок 2 – Сравнение стратегий по доле пользователей под риском, времени отката и избыточности ресурсов

Таблица 1 – Сводное сравнение стратегий развёртывания

Критерий	**Rolling**	**Blue-green**	**Canary**
Пользователей под риском	средне	высоко	низко
Время отката	долго	мгновенно	быстро
Избыточность ресурсов	умеренная	высокая	низкая
Скорость выката	высокая	высокая	низкая
Предсказуемость	средняя	высокая	высокая

Выбор стратегии и автоматический откат

Сравнение показывает, что универсально лучшей стратегии нет — выбор зависит от приоритета. Если важнее всего мгновенный откат и есть ресурсы на второе окружение, оправдано переключение окружений; оно удобно для редких, но крупных обновлений. Если приоритет — минимальный радиус поражения при ограниченных ресурсах, предпочтителен канареечный выкат; он особенно уместен для часто меняющихся пользовательских сервисов. Последовательная замена остаётся разумным выбором по умолчанию для некритичных сервисов, где не нужны ни мгновенный откат, ни тонкий контроль доли трафика.

Независимо от стратегии радиус поражения по выражению (1) тем меньше, чем быстрее обнаружен дефект. Поэтому на стенде выкат дополнялся автоматическим правилом отката, реализованным средствами автоматизации поставки [3]: при превышении доли ошибочных ответов или задержки сверх порога на текущем этапе выкат останавливался и трафик возвращался на исправную версию без участия человека. Такое правило особенно усиливает канареечный подход, превращая раннее наращивание доли в управляемый, самоостанавливающийся процесс.

Обсуждение

Предложенная модель риска позволяет сравнивать стратегии не по общим словам, а по сопоставимым числам, и объясняет наблюдаемые различия через одно выражение. Главный практический вывод состоит в том, что выбор стратегии — это выбор того, как распределить риск: сосредоточить его в коротком, но полном переключении или растянуть в виде малых порций. Числовая оценка радиуса поражения делает этот выбор осознанным.

Ограничения работы связаны с упрощениями модели. Выражение (1) предполагает, что дефект проявляется наблюдаемо и сразу после попадания трафика, тогда как часть ошибок носит отложенный характер и зависит от данных. Не учитывались также взаимодействия между несколькими одновременно обновляемыми сервисами и совместимость версий на стыке интерфейсов. Тем не менее даже простая модель отражает основные различия стратегий и даёт обоснование для выбора, а её применение не требует сложных вычислений.

Заключение

Выполнено сравнение трёх стратегий развёртывания микросервисов на единой основе. Предложена модель радиуса поражения, оценивающая ожидаемое число запросов, затронутых дефектной версией, и показано, что эта величина определяется тем, как стратегия распределяет долю трафика во времени. Количественное сравнение на стенде показало, что переключение окружений даёт мгновенный откат при высокой избыточности, последовательная замена — компромисс по всем критериям, а канареечный выкат — наименьший радиус поражения при умеренных затратах. Дополнение выката автоматическим правилом отката по метрикам уменьшает радиус поражения при любой стратегии.

Список литературы

1. Опарин, Г. А. Инструментальные средства автоматизации разработки и применения пакета прикладных микросервисов / Г. А. Опарин, В. Г. Богданова, А. А. Пашинин // Информационные и математические технологии в науке и управлении. – 2024. – № 2(34). – С. 155-168. – DOI 10.25729/ESI.2024.34.2.015. – EDN DNGVDT.

2. Крутяков, Д. В. Эволюция архитектур программного обеспечения: от монолита к микросервисам / Д. В. Крутяков // Вестник науки. – 2026. – Т. 1, № 2(95). – С. 730-736. – EDN OFVNYQ.

3. Арсеньев, Д. Г. Объяснимая runtime-детекция нарушений API-контрактов в микросервисах на основе T-GNN / Д. Г. Арсеньев, Д. Е. Баскаков // Промышленный искусственный интеллект (ПИИ'2025) : Сборник научных трудов Всероссийской научно-практической конференция с международным участием, Санкт-Петербург, 03–04 июля 2025 года. – Санкт-Петербург: ПОЛИТЕХ-ПРЕСС, 2025. – С. 513-519. – DOI 10.18720/SPBPU/2/id25-555. – EDN STGIEO.

4. Насонов, М. А. Модель управления потоками заданий ремонтным бригадам в условиях динамического планирования / М. А. Насонов // Моделирование систем и процессов. – 2025. – Т. 18, № 4. – С. 63-70. – DOI 10.12737/2219-0767-2025-18-4-63-70. – EDN MSAKEX.

5. Мочалов, В. П. Алгоритм динамического распределения и балансировки нагрузки в распределенных облачных вычислениях / В. П. Мочалов, Н. Ю. Братченко, Д. В. Гостева // Моделирование систем и процессов. – 2024. – Т. 17, № 1. – С. 92-102. – DOI 10.12737/2219-0767-2024-17-1-92-102. – EDN EWMPYM.

References

1. Oparin, G. A. Instrumental automation tools for the development and application of a package of applied microservices / G. A. Oparin, V. G. Bogdanova, A. A. Pashinin // Information and mathematical technologies in science and management. – 2024. – № 2(34). - St. 155-168. – DOI 10.25729/ESI.2024.34.2.015. – EDN DNGVDT.

2. Krutyakov, D. V. The evolution of software architecture: from monolith to microservices / D. V. Krutyakov // Bulletin of Science. – 2026. - Vol. 1, № 2(95). - St. 730-736. – EDN OFVNYQ.

3. Arsenyev, D. G. Explicable runtime detection of API contract violations in microservices based on T-GNN / D. G. Arsenyev, D. E. Baskakov // Industrial Artificial Intelligence (FDI'2025) : Proceedings of the All-Russian Scientific and Practical Conference with International Participation, St. Petersburg, July 03-04, 2025. - Saint Petersburg: POLYTECH PRESS, 2025. - St. 513-519. – DOI 10.18720/SPBPU/2/id25-555. – EDN STGIEO.

4. Nasonov, M. A. A model for managing task flows to repair teams under dynamic planning / M. A. Nasonov // Modeling of systems and processes. – 2025. – Vol. 18, No. 4. – pp. 63-70. – DOI 10.12737/2219-0767-2025-18-4-63-70. – EDN MSAKEX.

5. Mochalov, V. P. An algorithm for dynamic distribution and load balancing in distributed cloud computing / V. P. Mochalov, N. Yu. Bratchenko, D. V. Gosteva // Modeling of systems and processes. – 2024. – Vol. 17, No. 1. – pp. 92-102. – DOI 10.12737/2219-0767-2024-17-1-92-102. – EDN EWMPYM.