УДК 004.415.2

А. А. Золотов

РТУ МИРЭА, г. Москва

АВТОМАТИЗАЦИЯ РАЗВЁРТЫВАНИЯ И МОНИТОРИНГА ПРОГРАММНЫХ СИСТЕМ В CI/CD-КОНВЕЙЕРАХ

Аннотация. В статье рассматриваются методы автоматизации развёртывания и мониторинга программных систем в рамках CI/CD-конвейеров. Проведён анализ типовых этапов конвейера непрерывной интеграции и доставки, а также инструментальных средств их реализации. Описаны подходы к контейнеризации приложений, оркестрации развёртывания и организации наблюдаемости (observability) системы. Предложена модель конвейера, обеспечивающая автоматический откат при деградации ключевых метрик. Апробация проведена на примере веб-сервиса с непрерывным циклом поставки.

Ключевые слова: CI/CD, непрерывная интеграция, непрерывная доставка, развёртывание, контейнеризация, Docker, мониторинг, наблюдаемость, автоматический откат, DevOps.

Введение

Практики непрерывной интеграции и непрерывной доставки (CI/CD) стали стандартом в современной разработке программного обеспечения, позволяя сократить время выхода продукта на рынок и повысить надёжность процесса поставки [1]. Автоматизация сборки, тестирования и развёртывания снижает вероятность ошибок, вызванных ручным вмешательством, и обеспечивает воспроизводимость результата.

Вместе с тем практическое построение CI/CD-конвейеров сопряжено с рядом нетривиальных задач: обеспечением изолированности окружений, управлением секретами, организацией надёжного отката при неудачных развёртываниях и построением системы наблюдаемости, позволяющей своевременно обнаруживать деградацию качества обслуживания [2].

В настоящей статье предложена модель CI/CD-конвейера, включающая этапы автоматической сборки, многоуровневого тестирования, контейнеризованного развёртывания и мониторинга с автоматическим откатом, а также описаны инструментальные решения для каждого из этапов.

1. Этапы непрерывной интеграции

Этап непрерывной интеграции включает автоматический запуск сборки и тестирования при каждом коммите в репозиторий. Типовая цепочка CI состоит из следующих шагов: статический анализ кода (линтинг), компиляция или сборка

артефакта, запуск модульных тестов, запуск интеграционных тестов, формирование отчёта о покрытии [1]. Все шаги выполняются в изолированном контейнерном окружении, что исключает влияние состояния машины-исполнителя на результат сборки.

Для организации параллельного выполнения независимых шагов применяется матричная стратегия запуска, позволяющая одновременно тестировать несколько версий среды исполнения. Это сокращает общее время прохождения конвейера без потери полноты проверок.

1.1. Этапы непрерывной доставки

Этап непрерывной доставки охватывает сборку контейнерного образа, его публикацию в реестре образов и развёртывание в целевой среде. Контейнеризация приложения на основе Docker обеспечивает идентичность окружений разработки, тестирования и производственной эксплуатации [2]. Для минимизации размера образов применяется многоэтапная сборка (multi-stage build), при которой в финальный образ включаются только артефакты времени выполнения.

Стратегия развёртывания определяется требованиями к доступности сервиса. Для сервисов с высокими требованиями к непрерывности применяется сине-зелёное развёртывание или канареечный выкат, при которых новая версия постепенно принимает нагрузку под контролем метрик качества обслуживания.

2. Три столпа наблюдаемости

Концепция наблюдаемости (observability) основана на трёх категориях телеметрических данных: метриках, журналах (логах) и трассировках [3]. Метрики обеспечивают агрегированное представление о производительности системы и позволяют задавать пороговые условия для срабатывания алертов. Журналы сохраняют детальную запись событий, необходимую для диагностики инцидентов. Трассировки позволяют отслеживать путь запроса через распределённые компоненты системы.

Для централизованного сбора и хранения телеметрии применяется стек на основе открытых инструментов: Prometheus для метрик, Loki или Elasticsearch для журналов, Jaeger или Tempo для распределённых трассировок. Визуализация агрегируется в единой панели Grafana.

2.1. Автоматический откат при деградации метрик

Ключевым элементом предложенной модели является механизм автоматического отката развёртывания при деградации ключевых показателей качества обслуживания [4]. После выкатки новой версии конвейер переходит в режим наблюдения: в течение заданного временного окна (как правило, от 5 до 15 минут) оцениваются целевые метрики — частота ошибок, медианное время отклика и доля успешных проверок

работоспособности. При превышении пороговых значений инициируется автоматический откат к предыдущей стабильной версии без участия оператора.

3. Практическая реализация

Описанная модель конвейера реализована с использованием GitHub Actions в качестве оркестратора CI/CD-процессов. Сборка и публикация контейнерных образов осуществляется в реестр, доступный среде развёртывания. Развёртывание выполняется инструментом Kamal, обеспечивающим нулевое время простоя при замене контейнеров на целевом хосте.

Мониторинг производительности после каждого развёртывания реализован через опрос Prometheus-эндпоинта приложения. Решение об откате принимается на основе сравнения текущих значений метрик с базовыми показателями предшествующей версии, зафиксированными перед началом выкатки. Данный подход исключает зависимость от абсолютных пороговых значений, что повышает применимость механизма в условиях переменной нагрузки.

Заключение

В статье описана модель CI/CD-конвейера, охватывающая этапы непрерывной интеграции, контейнеризованного развёртывания и мониторинга с автоматическим откатом. Рассмотрены инструментальные решения для каждого этапа и обоснованы архитектурные решения, обеспечивающие надёжность и воспроизводимость процесса поставки.

Предложенный механизм автоматического отката на основе сравнения метрик с базовыми показателями предшествующей версии позволяет минимизировать время воздействия регрессий на пользователей без необходимости постоянного дежурства операционной команды. Дальнейшим направлением развития является интеграция прогнозной аналитики для упреждающего обнаружения деградации до её проявления на уровне пользовательского опыта.

Литература

1. Humble J., Farley D. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. New York : Addison-Wesley, 2010. 512 p. : электронный ресурс. URL: https://continuousdelivery.com (дата обращения: 08.09.2025).

2. Shahin M., Babar M. A., Zhu L. Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices // IEEE Access. 2017. Vol. 5. P. 3909–3943 : электронный ресурс. URL: https://ieeexplore.ieee.org (дата обращения: 14.09.2025).

3. Beyer B. et al. Site Reliability Engineering: How Google Runs Production Systems. Sebastopol : O'Reilly Media, 2016. 552 p. : электронный ресурс. URL: https://sre.google/sre-book (дата обращения: 19.09.2025).

4. Schermann G. et al. Continuous Experimentation: Challenges, Implementation Techniques, and Current Knowledge // IEEE Software. 2018. Vol. 35, № 2. P. 26–31 : электронный ресурс. URL: https://ieeexplore.ieee.org (дата обращения: 25.09.2025).

Информация об авторах

Золотов Александр Алексеевич — магистрант 2-го года обучения кафедры КБ-9 «Предметно-ориентированные информационные системы» РТУ МИРЭА, г. Москва. E-mail: zolotov.a.a1@edu.mirea.ru