УДК 004.021

DOI 10.21661/r-558847

Невский АА.

НЕПРЕРЫВНАЯ ИНТЕГРАЦИЯ И НЕПРЕРЫВНАЯ ДОСТАВКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Аннотация*: в статье рассмотрена комбинация непрерывной интеграции и непрерывной доставки программного обеспечения в процессе разработки. Непрерывная интеграция и непрерывная доставка объединяет разработку, тестирование и развертывание программных приложений. Результат включает в себя описание процесса интеграции, развертывания и доставки приложений, а также лучшие промышленные практики использования такой комбинации.*

Ключевые слова*: непрерывная интеграция, непрерывное развертывание, непрерывная доставка, разработка программного обеспечения, процесс разработки, программное обеспечение, интеграция, доставка приложений, развертывание приложений.*

Актуальность проблемы.

Современный процесс разработки и доставки программного обеспечения настолько многогранен и порой так сложен, что без хорошего процесса и комбинации непрерывной интеграции, непрерывной доставки и развертывания крайне сложно обойтись. Когда системы создаются второпях, когда увеличение штата программистов – единственный способ продолжать выпускать новые версии, и когда чистоте кода или дизайну уделяется минимум внимания и времени (или же вообще не уделяется), такой процесс рано или поздно приведет к краху работы над программным продуктом [2].

Комбинация непрерывной интеграции (continuous integration), непрерывного развертывания (continuous deployment) и непрерывной доставки (continuous delivery) призвана обеспечить в том числе гарантированный стандарт внесения изменений при разработке программного обеспечения.

Рис. 1. Комбинация концепции Continuous Integration (слева)
и Continuous Deployment (справа)

Главная проблема, с которой сталкиваются организации, в которых отсутствует хорошая практика CI / CD, состоит в том, что обнаружение дефектов, повышение производительности и обеспечение более быстрых циклов выпуска программных продуктов и артефактов, значительно осложнено и затруднено и требует больших человеческих затрат. Умение писать чистый код – тяжелая работа [3]. Традиционно, когда набор обновлений программного обеспечения интегрировался в один большой пакет перед развертыванием более новой версии, он был трудоемок, занимал продолжительное время и был подвержен ошибкам, в том числе человеческим [1]. В наши дни эту проблему призвана решить автоматизированная комбинация непрерывной интеграции, непрерывной доставки и развертывания программного обеспечения [6].

Описание предложенного решения.

Предложим схему взаимодействия командой разработки программного обеспечения с различными автоматизированными средствами и сервисами, призванными решить проблему непрерывной интеграции, непрерывной доставки и развертывания программного обеспечения.

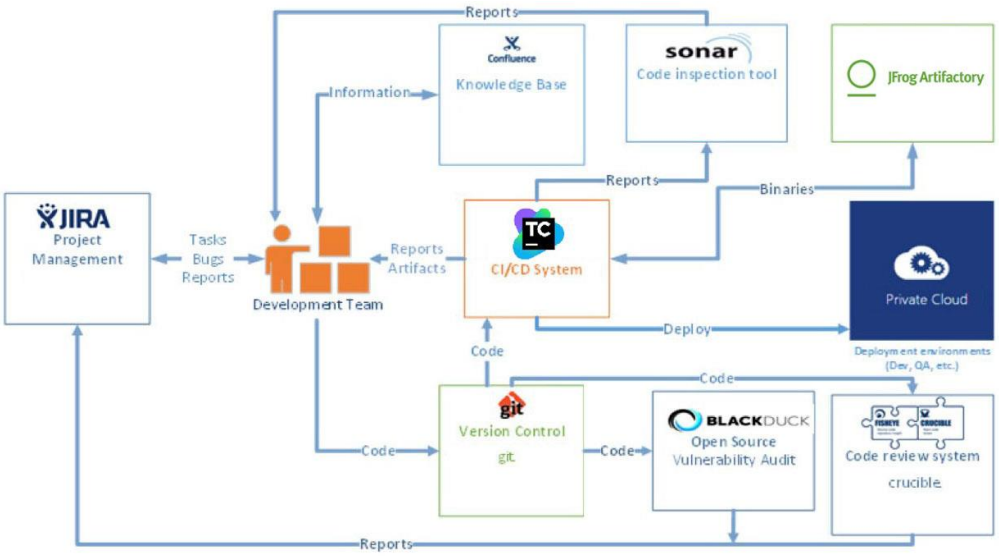

Рис. 2. Схема взаимодействия командой разработки
с различными сервисами в процессе разработки программного обеспечения

Команда разработки использует систему Jira для управления проектными отчетами и дефектами, загружает и получает информацию в базу знаний о проекте – систему Confluence. Разработанный программный код продукта загружается в систему управления исходным кодом, например, такую как Git. После этого этапа критически важно выстроить правильную работу комбинации непрерывной интеграции, непрерывного развертывания и непрерывной доставки [4].

Представим и опишем схему этой комбинации.

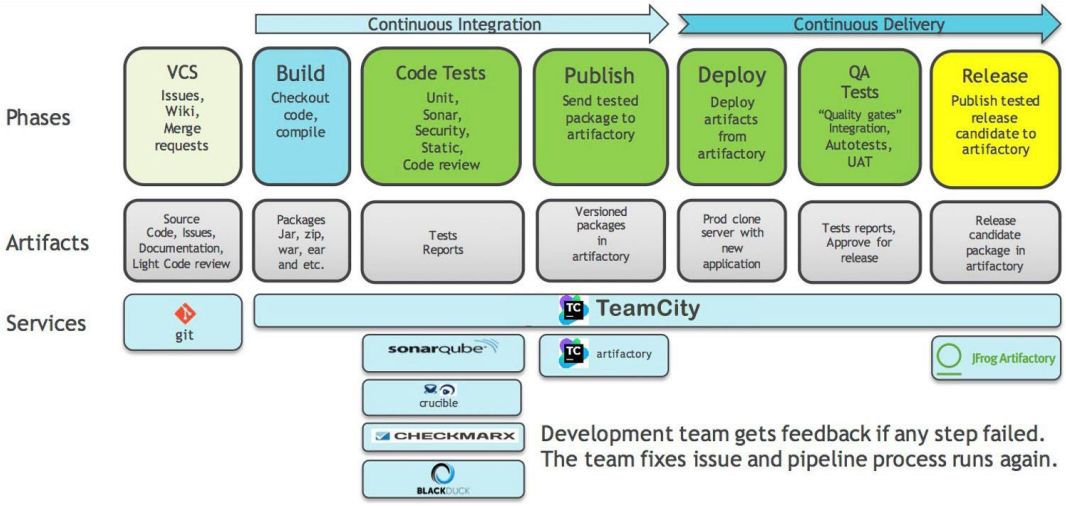

Рис. 3. Схема непрерывной интеграции, непрерывной доставки
и развертывания программного обеспечения

Опишем фазы работы процесса CI / CD и взаимодействие его компонентов:

– фаза 1: отправка исходного кода в систему управления исходным кодом, создание запроса на изменение кода;

– фаза 2: компиляция и сборка исходного кода;

– фаза 3: проверка исходного кода автоматизированными системами проверки получившегося конечного пакета на предмет прохождения базовых тестов, соответствия всем конвенциям, отсутствие потенциальных уязвимостей и прочих недостатков;

– фаза 4: публикация успешно прошедшего все проверки исходного кода и программного пакета;

– фаза 5: развертывание программного пакета на одном из тестовых серверов;

– фаза 6: «ворота качества» или проверка на основе ручного и автоматического тестирования новой версии приложения;

– фаза 7: доставка прошедшего все проверки артефакта с новым исходным кодом приложения.

На любом из данных этапов, если одна из фаз завершилась неудачно или не соответствует установленным стандартам качества и надежности, происходит автоматическая блокировка и возврат к предыдущей версии с предоставлением подробного отчета в результате одной или нескольких проверок, после чего следует как можно быстрее исправить обнаруженные недостатки [5] и повторить процесс заново, тем самым обеспечив гарантированный стандарт и высокое качество конечного программного продукта.

Вывод.

Представленная комбинация непрерывной интеграции, непрерывного развертывания и непрерывной доставки исходного кода позволяет команде разработки контролировать и управлять процессом внесения изменений, обеспечивает полный контроль и заданный уровень качества конечного продукта, предоставляет автоматический отчет в случае нахождения любых несоответствий, а также позволяет вести статистику удачных и неудачных попыток изменения исходного кода, предупреждает по мере своих возможностей о потенциальных уязвимостях

и программных ошибках. Для любой команды, разрабатывающей программный продукт, следование данным рекомендациям критически важно практически в каждом случае, если эта команда не хочет столкнуться с трудоемким ручным процессом внесения изменений и хаосом из большого количества дефектов, несоответствий и уязвимостей в разрабатываемом командой программным продуктом.

Список литературы

1. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка / М. Клеппман. – СПб.: Питер, 2020. – 30 с.

2. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. – СПб.: Питер, 2022. – 24 с.

3. Мартин Р. Чистый код. Создание, анализ и рефакторинг / Р. Мартин. – СПб.: Питер, 2019. –24 с.

4. Adkins H. Building Secure & Reliable Systems / H. Adkins, B. Beyer, P. Blankinship [et al.]. – CA, US.: O'Reilly Media, 2020. – 304 c.

5. Fowler M. Continuous Integration / M. Fowler [Electronic resource]. – Access mode: https://martinfowler.com/articles/continuousIntegration.html (date of access: 08.01.2023).

6. Red Hat – Topics. What is CI/CD? [Electronic resource]. – Access mode: https://www.redhat.com/en/topics/devops/what-is-ci-cd (date of access: 08.01.2023).

Невский Алексей Александрович – бакалавр, ведущий инженер-программист, AWS Certified Architect, Glovo, Беларусь, Минск.