DOI 10.37882/2223–2966.2021.06.24

ПРАКТИЧЕСКИЙ ОПЫТ ВНЕДРЕНИЯ КУЛЬТУРЫ И ИНСТРУМЕНТОВ DEVOPS В ПРОЕКТ НА ПЛАТФОРМЕ PEGA В КРУПНОЙ ОРГАНИЗАЦИИ

Лапин Алексей Александрович
Специалист, Национальный исследовательский ядерный университет «МИФИ» (Москва); старший программист, ООО «Люксофт Профешнл», Москва
so.yazzzie@gmail.com

Аннотация. Настоящая статья описывает опыт, полученный в процессе внедрения инструментов и практик DevOps в крупной организации на проекте, использующем платформу Pega BPM. В начале статьи изложена исходная ситуация на проекте и существующие проблемы. Далее, приведено описание произведенных изменений как технических, так и организационных. В заключении представлены результаты и сделаны выводы об эффективности реализованных нововведений.

Ключевые слова: Pega BPM, DevOps, автоматизация, непрерывная интеграция, непрерывная поставка, непрерывное развертывание.

PRACTICAL EXPERIENCE OF IMPLEMENTING DEVOPS CULTURE AND TOOLS ON A PROJECT USING THE PEGA PLATFORM IN A LARGE ORGANIZATION

A. Lapin

Summary. This article describes the practical experience gained while implementing DevOps tools and practices in a large organization on a project using the Pega BPM platform. At the beginning of the article, the initial situation on the project and the existing problems are described. Then, a description of the changes made, both technical and organizational is given. In the conclusion, the results are presented and conclusions are drawn about the effectiveness of the implemented innovations.

Keywords: Pega BPM, DevOps, automation, continuous integration, continuous delivery, continuous deployment.

1. Введение

Данная статья описывает опыт проектирования и реализации методологии ведения разработки на платформе Pega BPM и инфраструктуры для автоматизации процессов сборки, тестирования и доставки конечного дистрибутива системы в одном из крупнейших банков России, далее Организация. Автор ставит своей целью осветить ряд важных, по его мнению, особенностей внедрения практик и инструментов DevOps в крупной организации без углубления в специфику её деятельности.

Автор статьи являлся сотрудником Организации и принимал непосредственное участие в проектировании и разработке большинства компонентов инфраструктуры.

Все работы происходили в рамках проекта по автоматизации бизнес-процессов, связанных с кредитованием, далее Проект. Они заключались в создании единого рабочего места для пользователей (сотрудников Организации) различных ролей, принимающих участие на разных этапах предоставления услуг по кредитным продуктам.

Использование платформы Pega BPM [6] на Проекте обусловлено тем, что продукт является современным, комплексным решением для моделирования и автоматизации бизнес-процессов, хорошо зарекомендовал себя на рынке и успешно проявил себя во многих крупных банках и не финансовых организациях по всему миру [5]. Лицензия на использование платформы была приобретена Организацией, а сама платформа уже эксплуатировалась на нескольких смежных проектах. Но несмотря на длительное использование Pega в Организации, процесс доставки дистрибутива системы до контура промышленной эксплуатации был неэффективным и мог занимать, в большинстве случаев, продолжительное время вплоть до недели. Кроме прочего, этап доставки релиза включал обязательное написание подробнейшей инструкции по установке и требовал значительных объемов ручных действий специалистов из нескольких подразделений.

2. Описание Проекта

Поводом для пересмотра подхода к разработке и выстраиванию конвейера поставки продукта к заказчику стала полномасштабная трансформация Организации. Изменения заключались в модификации структурных

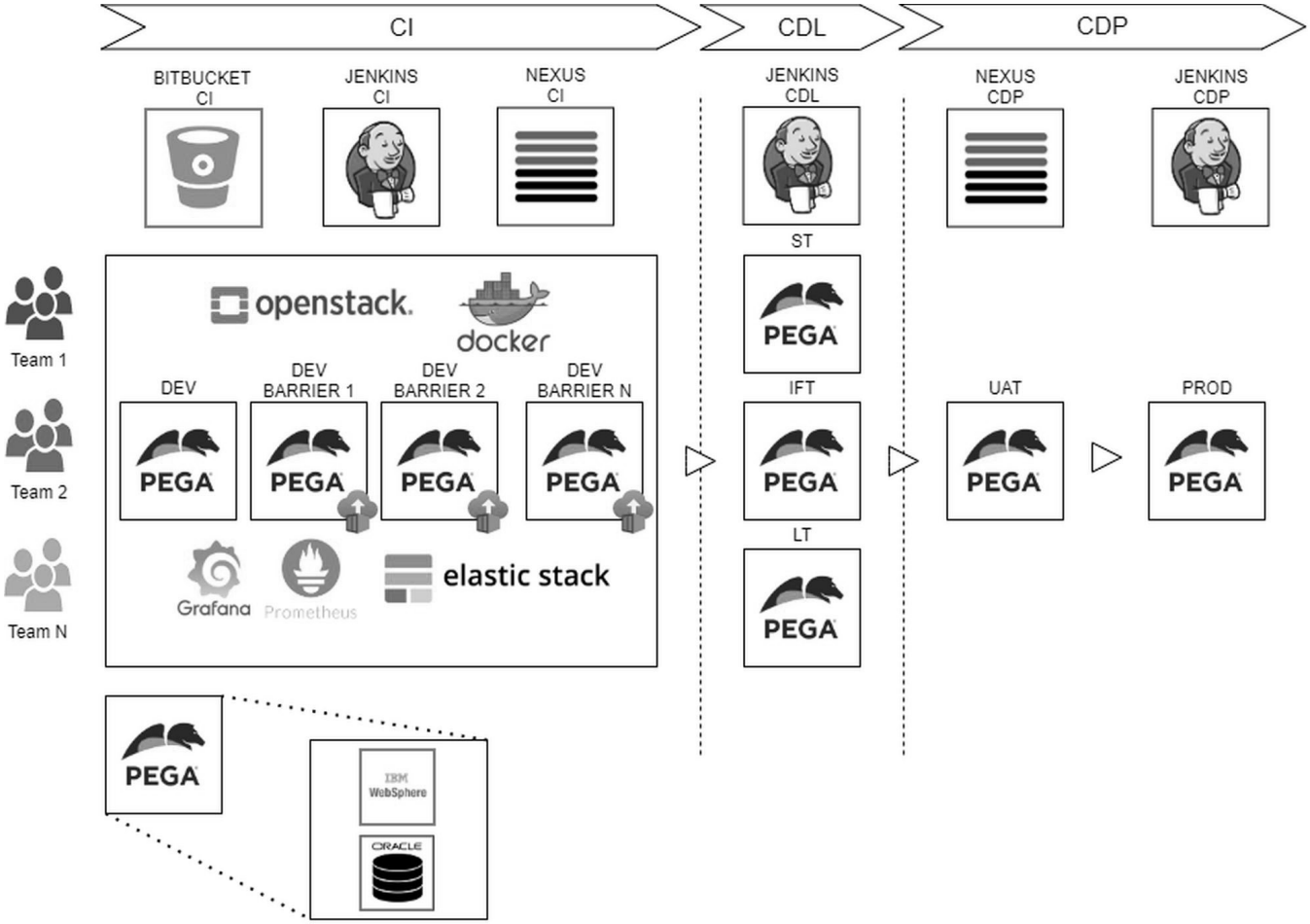

Рис. 1. Верхнеуровневая схема DevOps архитектуры на Проекте

подразделений, найме иностранных экспертов для курирования нововведений, а также создании и внедрении подходящей для Организации производной Agile [10] методологии для разработки программного обеспечения.

К моменту преобразований, человеческие ресурсы Проекта включали разработчиков, аналитиков, тестировщиков, менеджеров и бизнес-экспертов. В ходе трансформации все специалисты были распределены в 6 команд по 8–9 человек, каждая из которых включала представителей всех перечисленных компетенций. Команды сосредоточились на реализации отдельных составляющих общего бизнес-процесса на платформе Pega. В то же время, одна из них, та в которую входил автор, направила все усилия на автоматизацию инфраструктуры для платформы.

В рамках Agile-трансформации предполагалось внедрение культуры, практик и инструментов DevOps [11]. Требовалось провести масштабную работу по применению практик непрерывной интеграции [8], непрерывной доставки [7] и непрерывного развертывания [9]. Другими словами, автоматизированию подлежало большое количество ручных действий, которые раньше выполнялись в процессе релизного цикла программного обеспечения. По данному пути прошли многие компании из сферы информационных технологий, о чем свидетельствуют всевозможные источники [1–4].

На старте, для всех проектов был утвержден единый набор таких инструментов и сервисов, как система версионирования исходного кода и хостинг для него, сервер автоматизации, хранилище артефактов и другие. Также были утверждены практики, применяемые в релизном цикле программного обеспечения: автоматизированная сборка, тестирование и развертывание. Для проектов Организации на Java, адаптация практик проходила быстрее и эффективнее, чем на Pega. Так было благодаря наличию большого количества готовых к применению инструментов и опыту в этой сфере как у сообщества в целом, так и Организации в частности. Применение же вышеперечисленных инструментов и практик в проекте на Pega потребовало значительных усилий. Во многих аспектах Проект автора данной ста-

тьи стал первопроходцем, а разработанные средства впоследствии были успешно заимствованы другими проектами на платформе Pega.

3. Pega и особенности сборки дистрибутива

Платформа Pega представляет собой web-приложение, написанное на языке программирования Java с применением трёхзвенной архитектуры. Таким образом, Pega использует сервер приложений, базу данных и предоставляет графический пользовательский интерфейс через браузер. В Организации уже осуществлялась эксплуатация продуктов IBM WebSphere и Oracle DB, поэтому вопрос выбора сервера и базы данных не стоял. Разработка на платформе осуществляется через специализированный портал разработчика в браузере. Там же происходит получение артефакта для установки на другие стенды и сама установка.

В период, предшествующий внедрению каких-либо DevOps практик, сборка дистрибутива системы осуществлялась вручную через пользовательский интерфейс Pega ответственным членом одной из команд. Первым шагом на пути уменьшения ручных действий стало применение инструмента от производителя платформы, позволяющего взаимодействовать с экземпляром Pega по Rest интерфейсу и представляющего из себя Java приложение и Ant обертку для него. Инструмент имел функционал по сборке дистрибутива и его установке из командной строки. Впрочем, возможностей данного инструмента было недостаточно для покрытия всех потребностей по автоматизированному переносу дистрибутива системы между средами. Дополнительно требовалось переносить изменения схемы базы данных вне артефакта, выдаваемого платформой при сборке, и конфигурации сервера приложений WebSphere.

Усилия автора в составе команды автоматизации позволили реализовать новый инструмент, основанный на инструменте от вендора и предоставляющий больше контроля над процессом сборки.

Написание Maven-плагина позволило инкапсулировать механизм взаимодействия с платформой через новый инструмент в процесс сборки Maven-проекта. В дополнение, члены продуктовых команд смогли декларативно описывать состав дистрибутива и модулей системы, и хранить их в отдельных git-репозиториях в Bitbucket.

4. Частное облако и Ansible

Для упрощения создания и перераспределения вычислительных мощностей, Проекту была предоставлена квота ресурсов в Openstack. Это позволило значительно ускорить развёртывание новых виртуальных машин для целей Проекта командой автоматизации без взаимодействия со специалистами инфраструктурного обеспечения. Таким образом, на мощностях Openstack был развернут ресурсоемкий стенд разработки (DEV), обеспечивающий процесс разработки 6 команд, и стенды проверки сборок (DEV-BARRIER). В последних, по необходимости, автоматически создавались Docker-контейнеры из образов предыдущих релизов, в которых происходило инсталляционное тестирование сборок компонент и системы текущего релиза. Кроме того, там же были развернуты средства для сбора, хранения и визуализации логов (Elastic Stack) и сбора и отображения метрик (Prometheus + Grafana). Верхнеуровневая схема DevOps архитектуры проекта представлена на Рис. 1.

Для установки дистрибутива были написаны Ansible Playbook'и и Ansible роли, ввиду того что Ansible был принят Организацией как обязательный инструмент, используемый при развертывании систем. Playbook по установке принимал в качестве параметра путь к дистрибутиву и, с помощью специализированных ролей, запускал установку составляющих дистрибутива, таких как артефакты с кодовой базой платформы, артефакты с Liquibase скриптами миграций базы данных и артефакты с Jython-скриптами конфигурации WebSphere.

5. Сервер автоматизации

В качестве сервера автоматизации в Организации был выбран Jenkins. Этот инструмент являлся ключевым элементом инфраструктуры непрерывной интеграции и, в тоже время, входной точкой ко всем автоматизированным действиям, выполняемым с составляющими системы. Для большинства Job (задач) в Jenkins использовался подход Infrastructure as a Code (IaaS) с применением Pipeline-плагина и написанием Groovy-скриптов, находящихся под версионным контролем в Bitbucket. Автором были написаны Groovy Pipeline скрипты и переиспользуемые Jenkins Shared Libraries (библиотеки) для сборки компонентов системы, сборки общего дистрибутива системы, установки общего дистрибутива системы на целевой стенд, бэкапа базы данных и многие другие.

Сборка компоненты системы представляла из себя многоступенчатый процесс и включала несколько этапов:

- клонирование репозитория компоненты;
- запуск Maven для создания архива, содержащего все необходимые для функционирования компоненты составляющие;

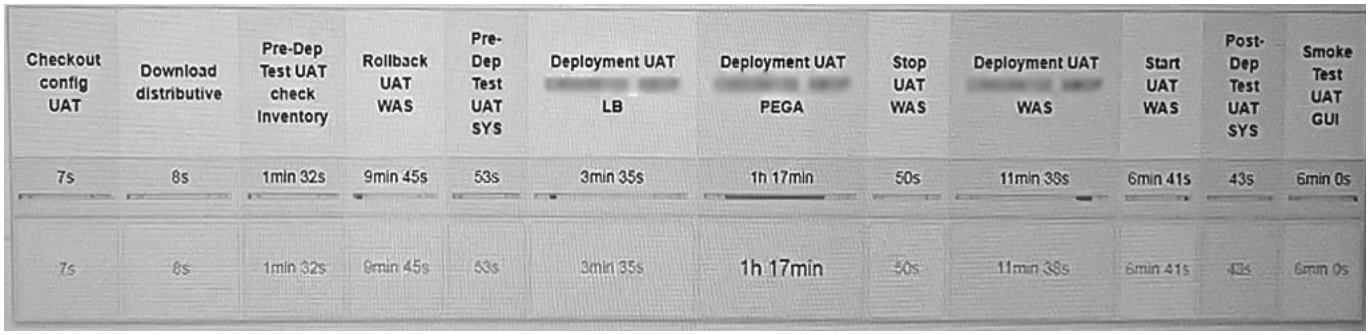

Рис. 2. Пример набора этапов pipeline установки дистрибутива на стенд приёмо-сдаточных испытаний

- запуск совокупности Docker-контейнеров с версией системы, соответствующей установленной на промышленный контур (prod-like);
- установка компоненты в контейнеризированное приложение, выполнение различного рода тестов из репозитория компоненты как Unit, так и автоматизированных GUI тестов;
- в случае их успешного прохождения — загрузка полученного артефакта компоненты в Nexus;
- удаление сборочных контейнеров.

Сборки компонент осуществлялись командами-владельцами по мере необходимости, путём запуска соответствующей Job в меню сервера автоматизации Jenkins, либо по коммиту в репозиторий компоненты в Bitbucket. В среднем, время сборки составляло 10–15 минут, но могло варьироваться в зависимости от компоненты, количества изменений и тестов.

Этапы сборки общего дистрибутива системы были сходны с этапами сборки компоненты, а общесистемный git-репозиторий содержал Maven-проект с перечнем компонент-зависимостей.

Сборки общего дистрибутива системы осуществлялись каждую ночь по расписанию и по мере необходимости, путём запуска соответствующей Job в меню сервера автоматизации Jenkins. Среднее время сборки составляло от 15 минут до 1 часа, и, аналогично времени сборки компоненты, могло варьироваться в зависимости от количества и состава компонент.

Большинство Job в Jenkins были запрограммированы на отсылку уведомлений по почте, что позволяло оперативно получать обратную связь о статусе выполняемых задач и своевременно предпринимать действия по устранению ошибок при их возникновении. Job в Jenkins были предметом постоянных оптимизаций и доработок, повышающих стабильность сборок и уменьшающих время их выполнения.

В Организации была применена схема контроля качества программного обеспечения, подразумевающая сохранение флагов прохождения различных проверок, так называемых Quality Gates. При этом, при сборке общего дистрибутива системы, результирующий артефакт загружался в Nexus, а по мере прохождения Quality Gates артефакт дополнялся файлами со строго заданными именами (флагами). Далее, при установке дистрибутива на стенд проверялось наличие обязательных для данного стенда флагов и, в случае их отсутствия, установка прекращалась, выдавая ошибку о том, что устанавливаемая версия дистрибутива не отвечает необходимым требованиям качества.

С реализацией возможности установки на промышленный контур, было достигнуто согласованное функционирование всех инструментов инфраструктуры и выстроен конвейер по автоматизированной доставке дистрибутива на все контуры системы.

6. Заключение

Активное участие автора статьи в проектировании, разработке и внедрению культуры, практик и инструментов DevOps, позволило значительно уменьшить объем ручных действий, необходимых для прохождения всего процесса переноса продукта со стенда разработки до стенда промышленной эксплуатации, при этом сократив время с нескольких дней до 3 часов. Появилась возможность релиза на стенд промышленной эксплуатации по запросу, для чего требовалось выполнить всего несколько кликов мыши, произведя запуск соответствующей задачи на сервере автоматизации. Внедрение практик непрерывной интеграции, непрерывной поставки и непрерывного развертывания значительно повысило качество системы, при этом количество критических дефектов сократилось на 35%, сократилось время вывода новой функциональности заказчику до двухнедельных итераций, а разработчикам позволило максимально сосредоточиться на ре-

ализации бизнес-требований. В подведении итогов также следует отметить, что внутренние отчеты Организации показали значительную финансовую выгоду от проделанной работы.

Несмотря на успешное достижение целей Agile-трансформации на проекте, несомненно, оставался простор для дальнейшего развития и улучшения инфраструктуры автоматизации. Одним из предложений был отказ от некоторых составляющих, например, Ant, ввиду наличия большого объема логики по сборке в Ant-скриптах, которые неудобны и неэффективны в тестировании. Другим — сокращение логики по установке артефактов в Ansible и Jenkins Pipleline скриптах по тем же причинам. Оптимальным решением стало бы создание отдельного инструмента, включающего указанную функциональность, на языке Java.

ЛИТЕРАТУРА

1. Бейер Б. Site Reliability Engineering. Надежность и безотказность как в Google / Б. Бейер, Д. Петофф, К. Джоунс [и др.]; — Санкт-Петербург: Питер — 592 с.
2. Ким Д. Проект «Феникс». Роман о том, как DevOps меняет бизнес к лучшему / Д. Ким, Д. Спаффорд, К. Бер. — Москва: Бомбора, 2015. — 410 с.
3. Ким Д. Руководство по DevOps. Как добиться гибкости, надежности и безопасности мирового уровня в технологических компаниях / Д. Ким, Д. Уиллис, П. Дебуа [и др.]; — Москва: Манн, Иванов и Фербер (МИФ) — 550 с.
4. Нейгард М. Release it! Проектирование и дизайн ПО для тех, кому не всё равно / М. Нейгард. — Санкт-Петербург: Питер — 320 с.
5. All Customers | Pega [Электронный ресурс]. — URL: https://www.pega.com/customers (дата обращения: 28.10.2020).
6. Customer engagement solutions on a unified CRM platform | Pega [Электронный ресурс]. — URL: https://www.pega.com/products/pega-platform (дата обращения: 28.10.2020).
7. Fowler M. ContinuousDelivery / M. Fowler // ContinuousDelivery [Электронный ресурс]. — URL: https://martinfowler.com/bliki/ContinuousDelivery.html (дата обращения: 01.11.2020).
8. Fowler M. Continuous Integration / M. Fowler // Continuous Integration [Электронный ресурс]. — URL: https://martinfowler.com/articles/continuousIntegration.html (дата обращения: 01.11.2020).
9. Fowler M. DeploymentPipeline / M. Fowler // DeploymentPipeline [Электронный ресурс]. — URL: https://martinfowler.com/bliki/DeploymentPipeline.html (дата обращения: 01.11.2020).
10. Manifesto for Agile Software Development [Электронный ресурс]. — URL: https://agilemanifesto.org/ (дата обращения: 28.10.2020).
11. Wilsenach R. DevOpsCulture / R. Wilsenach // DevOpsCulture [Электронный ресурс]. — URL: https://martinfowler.com/bliki/DevOpsCulture.html (дата обращения: 01.11.2020).

Журнал «Современная наука: актуальные проблемы теории и практики»