Проектирование и разработка технологического стека для обеспечения релизного процесса в рамках облачной разработки для крупного корпоративного решения

М. В. Старкин, e-mail: starkin_mihail@mail.ru

Воронежский государственный университет, г. Воронеж, Россия

Аннотация. *На основе текущего технологического стека, который используется в корпоративном решении, были разработаны дополнительные инструменты, с помощью которых было налажено взаимодействие между командами для обеспечения релизного процесса.*

Ключевые слова: *релиз, автоматизация, пайплайн.*

Введение

В процессе разработки на одном большом проекте могут участвовать множество команд, такие как разработчики, инженеры по тестированию, DevOps и релиз-инженеры. В процессе поддержки и разработки программного продукта требуется фиксировать определенные версии продукта, необходимые для различных команд. Это могут быть новые релизные версии, версии, несущие в себе срочные исправления, версии, имеющие в себе новый определенный набор изменений. Если продукт является микросервисным, и микросервисов в его составе достаточно много, процесс ручного отведения релиза и сопутствующих операций, например, контроль версий, становится очень затратным по таким критериям, как время и ресурсы. Если релизную версию проекта требуется отводить раз в определенный достаточно большой промежуток времени, и это можно реализовать с помощью ручного труда релиз-инженеров, то версии с исправлениями, наоборот, должны выпускаться постоянно, и в последствии они должны устанавливаться на продуктовое оборудование, оборудование инженеров по тестированию, разработчиков.

Чтобы наладить процесс передачи новых версий продукта от разработчиков к инженерам по тестированию для последующего тестирования и сделать его боле прозрачным, была поставлена задача спроектировать и разработать систему, которая позволяла бы

отслеживать поток разработанных, протестированных, а также выданных заказчику версий.

1. Текущее состояние системы

Для дальнейшего описания требуется определить некоторые основные понятия и технологии, которые будут использоваться в дальнейшем.

Пайплайн – последовательность выполнения стадий, каждая из которых включает несколько задач [1]. Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Релиз – версия продукта, которая является итогом некоторого отрезка разработки, содержащая в себе обновление кодовой базы продукта.

Перед тем как начать проектирование новой системы, необходимо провести анализ текущего состояния системы и определить основные проблемы.

Разработка и конфигурация продукта происходит в рамках трех основных этапов, уровней.

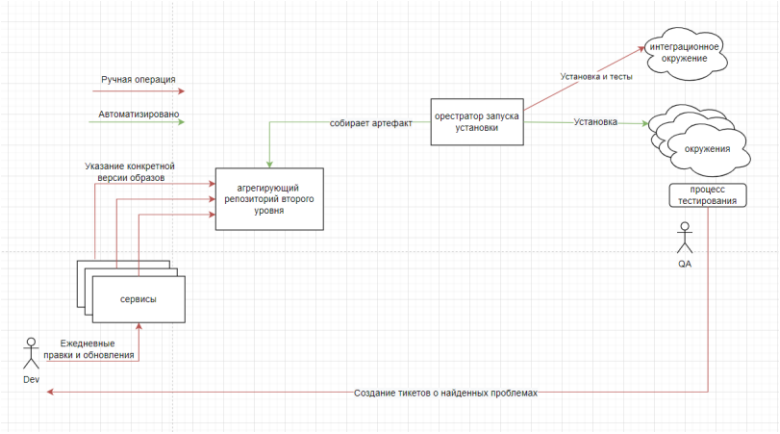

Рис. 1. Схематичное упрощенное представление первых двух уровней

Первый уровень – уровень репозитория, то есть непосредственно кодовая база микросервисов. В результате их сборки через удаленный инстанс Jenkins происходит генерация Docker-образов, использующихся в дальнейшем. Далее идет второй уровень - проектный. Это также репозиторий в используемой системой управления репозиториями кода, который содержит в себе указания на все микросервисы, из которых

состоит продукт. В текущем варианте используется прямое указание на версии используемых микросервисов – на конкретные версии Docker-образов. Это один из недостатков текущей системы, так как при каждом изменении в сервисе необходимо обновить Docker-образ в данном репозитории. Но с другой стороны это предоставляет определенный плюс в том, что сборка с определенного коммита всегда даст одинаковый результат, тем самым, обеспечивая воспроизводимость определенного состояния [1]. Этот метод предоставляет возможность определения, какие образы были использованы путем простого просмотра файлов внутри репозитория. Результатом сборки этого агрегирующего сервиса является новая версия манифеста приложения-продукта, который и будет устанавливаться через удаленный инстанс Jenkins в некоторое PaaS окружение. Третий уровень – сборка итогового артефакта, который будет выдан заказчику. Включает в себя информацию о версии продукта, то есть манифест, который был получен на втором уровне, а также все версии инфраструктурных сервисов. Инфраструктурная часть продукта не является темой данной работы.

Рассмотрим основные проблемы, которые есть в системе. Первая из них - отсутствие прозрачности передачи версии от разработчиков к инженерам по тестированию. Проблема заключается в том, что в данный момент процесс взаимодействия является таковым, что инженеры по тестированию устанавливали на свои окружения не конкретную версию продукта, которую им указали, а состояние агрегирующего репозитория в какой-то момент, и не всегда она совпадала по версиям микросервисов с версией продукта, которую разработчики считали стабильной. Также, как следствие этой проблемы, отсутствовал стабильный процесс передачи фиксированной протестированной версии дальше на третий уровень.

Второй основной проблемой является отсутствие автоматизации по отведению релиза для данного продукта. Данная проблема накладывает дополнительную нагрузку на DevOps-инженера проекта по отведению новых релизных веток для сервисов, обновление конфигурационного репозитория и другие операции, которые необходимо провести на данном технологическом стеке.

2. Проектирование новой системы

При анализе были выявлены основные две проблемы – отсутствие прозрачности выдачи версий продукта разработчиками инженерам по тестированию, и также отсутствие автоматизации по релизным процессам, что приводило к дополнительной нагрузке DevOps-инженера.

Была спроектирована и разработана следующая система. Ее схематичное упрощенное представление с компонентами, процессами и действующими пользователями, которые будут взаимодействовать с данной системой, представлена на рисунке.

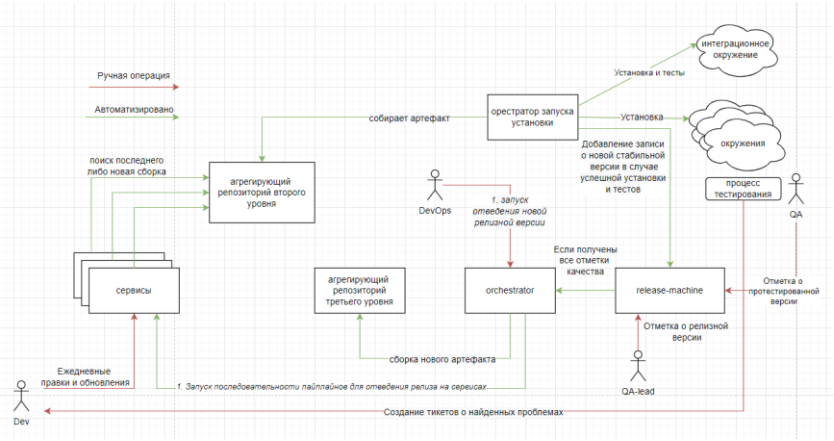

Рис. 2. Схематичное упрощенное представление новой системы

Для стабилизации выдачи новых версий на проекте была введена ежедневная ночная установка всего продуктового решения в тестовое окружение. На первом этапе это ввелось при помощи базовых возможностей распределенной системы хранения кода Gitlab, которая предоставляет возможность запуска пайплайнов в определенный отложенный момент времени при помощи расписания [1]. Пайплайн запускался на определенном сервисе, который был разработан ранее. Он является одной из конфигурационных точек установки продукта, а также выполняет роль оркестратора установки. Также при каждой ночной установке после успешного разворота запускались автотесты на данном окружении для проверки тесткейсов, которые были автоматизированы. Если установка проходила все этапы успешно, версию продукта, которая была установлена, можно было считать стабильной и выдавать инженерам по тестированию для полного цикла тестирования, также включающее в себя мануальные шаги. Это было принято считать первой качественной проверкой. Необходимо было где-то сохранять такие стабильные версии. Для того, чтобы хранить информацию о том, что версия продукта прошла какие-либо стадии, было принято решение разработать компоненту, которая могла бы хранить эту информацию. Для первой версии была использована

упрощенная схема. Была создана дополнительная компонента, которая содержала в себе файл, который содержал в себе записи о версиях продукта, а также их качественных отметках. Часть полей в этих записях проставлялось самой системой в автоматическом режиме, часть же полей должно было проставляться определенными пользователями. Если версия получала отметки во всех полях, которые отвечали за качественные проверки, то запускалась автоматически сборка артефакта третьего уровня с включенным в него версией продукта.

Для решения проблемы автоматизации релизных процессов были использованы следующие инструменты компании, разработанные ранее:

1. Оркестратор – сервис, задачей которого является для списка сервисов построение дерева зависимостей и в порядке от листов к корню этого дерева запускать пайплайны для сервисов.
2. Unified-service-pipeline – компонента, содержащая реализацию большинства процессов жизненного цикла для микросервиса.

Заключение

Проведенный анализ, проектирование и разработка позволяют повысить эффективность системы, ввести определенные пункты качества, зафиксировать роли пользователей, повысить управляемость процессов за счет ускорения и автоматизации процессов, а также введения непрерывного релизного процесса на проекте.

Дальнейшее развитие можно продолжать в следующих направлениях: увеличение количества автоматизируемых процессов и дальнейшее развитие компоненты, хранящей информацию о версиях продукта, в сторону повышения надежности, например, за счет использования базы данных.

Литература

1. Документация Gitlab, Gitlab CI/CD Pipeline Configuration [сайт]. URL: https://docs.gitlab.com/ee/ci/yaml/README.html