УДК 004.05

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ CI/CD ПРИ РАЗРАБОТКЕ ПРИЛОЖЕНИЯ «АВТОМАТИЗИРОВАННАЯ СИСТЕМА КОММУНИКАЦИЙ СТУДЕНТОВ, АДМИНИСТРАЦИИ И ПОТЕНЦИАЛЬНЫХ РАБОТОДАТЕЛЕЙ»

В. И. БАКЛАН
(Представлено: канд. техн. наук, доц. И.Б. БУРАЧЁНОК)

Рассмотрены особенности использования CI/CD, а также основные практики и подходы CI/CD непрерывной интеграции и доставки на примере разработанного приложения «Автоматизированная система коммуникаций студентов, администрации и потенциальных работодателей». Показаны преимущества применения данной технологии в процессе разработки программного продукта.

Современная модель получения высшего образования в нашей стране построена на базе коллективного обучения. Это обосновано тем, что студентов гораздо больше, чем преподавателей. В такой системе есть свои плюсы, но основным минусом является то, что она не предоставляет возможности студенту сфокусироваться на конкретном направлении, а взамен – ВУЗ выпускает специалиста широкого профиля, охватывая как можно больший спектр вакансий. Однако получив такого специалиста, работодатель должен затратить дополнительное время и средства на его развитие в зависимости от позиции. Улучшить имеющуюся модель образования можно включив в неё элемент индивидуального обучения (ИО) автоматизировать его. Цель автоматизации – упрощение взаимодействия преподавателя с множеством студентов, при сохранении элемента получения индивидуального образования, что, несомненно, является актуальным.

Таким образом, возникает необходимость создания автоматизированной системы (АИ) коммуникации студентов, администрации и потенциальных работодателей. Разрабатываемая система должна взять на себя решение части задач студента и преподавателя, а в последствии и потенциального работодателя. Причём, со стороны студента система должна: позволять указывать студенту приоритетные направления; мониторить прогресс; фиксировать достижения в той или иной области в результате чего будет формироваться карта обучения, в результате прохождения которой выпускник сможет найти позицию, подходящую приобретёнными знаниями и навыками. Со стороны преподавателя система должна: предоставлять возможность анализа прогресса каждого студента; создавать индивидуальные задачи и планы, на базе заметок; мониторить прогресс одного или группы обучающихся, что позволит реализовать частичное ИО при допустимых затратах сил и времени. Работодателю же, система должна предоставить возможность мониторинга интересующих студентов, просматривая карты обучения конкретного человека, а также возможность корректировки плана обучения на поздних этапах учебного процесса, с целью чтобы после выпуска из учебного заведения специалист был сразу готов работать на определенной позиции.

Непрерывная интеграция (CI, англ. Continuous Integration) – практика разработки программного обеспечения (ПО), которая заключается в постоянном слиянии рабочих копий в общую ветвь разработки и выполнения частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем.

В обычном проекте, где над разными частями системы разработчики трудятся независимо, стадия интеграции является заключительной. Она может непредсказуемо задержать окончание работ. Переход к непрерывной интеграции позволяет снизить трудоёмкость интеграции и сделать её более предсказуемой за счёт раннего обнаружения и устранения ошибок и противоречий. Основным преимуществом сокращения стоимости исправления дефект, за счёт раннего её выявления.

Далее подробнее остановимся на рассмотрении основных этапов CI процесса в приложении «Автоматизированная система коммуникации студентов, администрации и потенциальных работодателей»

1 Checkout в требуемую ветку (по умолчанию – master);
2 Сборка приложения;
3 Запуск unit-тестов;
4 Запуск integration-тестов;
5 Создание docker образа;
6 Выгрузка образа в хранилище.

CI процесс построен на базе GitHub для системы хранения и менеджмента кода, Github actions для автоматизации Integration стадии. Для контейнеризации приложения используется Docker Hub, в котором хранятся версионированные образы приложения.

Docker – это платформа, которая предназначена для разработки, развёртывания и запуска приложений в контейнерах. Слово «Docker» в последнее время стало чем-то вроде синонима слова «контейнеризация».

При сборке образа для master ветки создаётся два тега: latest и версионированный тег для того, чтобы развёртывание было проще, так как для данного приложения не критична возможность мгновенного отката к предыдущей версии приложения.

При сборке других веток создаётся только один тег со случайным хеш-кодом, чтобы была возможность развёртывания промежуточных веток в процессе разработки и/или тестирования.

Непрерывная доставка (CD, англ. Continuous delivery) – это практика автоматизации всего процесса релиза ПО. Цель заключается в том, чтобы выполнять CI, плюс автоматически готовить и вести релиз к версии для конечного пользователя. При этом желательно добиться следующего: любой, кто обладает достаточными привилегиями для развёртывания нового релиза может выполнить развёртывание в любой момент, и это можно сделать в несколько кликов. Программист, избавившись практически от всей ручной работы, трудится продуктивнее [1].

Как правило, в процессе непрерывной доставки требуется выполнять вручную как минимум один этап: одобрить развёртывание в окружении конечного пользователя и запустить его. В сложных системах с множеством зависимостей конвейер непрерывной доставки может включать дополнительные этапы, выполняемые вручную либо автоматически.

В рассматриваемом приложении CD реализован на базе AWS Code Deploy, который позволяет разворачивать приложение полностью из заранее собранных контейнеров в несколько кликов. Ограничение привилегий достигается за счёт использования индивидуальных AWS IAM пользователей.

Само приложение развёртывается на базе AWS Web Services, в частности, AWS EC2 (elastic cloud containers) в качестве виртуального сервера, EBS в качестве физически доступной памяти, S3 для хранения ресурсов, RDS для менеджмента базы данных.

Отдельное внимание стоит уделить EC2, так как данный сервис непосредственно предоставляет сервера, на которых разворачиваются контейнеры. EC2 – это облачный сервис, предоставляющий виртуальные сервера (Amazon EC2 Instance), 2 вида хранилищ данных, а также балансировщик нагрузки (Load Balancer). EC2 позволяет запускать уже заранее сконфигурированные серверы с предустановленными ОС. Так же возможно создавать свои образы (AMI – Amazon Machine Image) и использовать любой Linux. Есть возможность настроить защиту доступа к серверам. EC2 инстансы объединяются в группы безопасности (Security Groups) с возможностью ограничения доступа по портам с IP или подсетей. Балансировка нагрузки и автомасштабирование являются очень важными функциями EC2. Можно создать правила, при которых станет возможно автоматически увеличить количество серверов, например, если один или несколько серверов не справляются с нагрузкой. Контроль за здоровьем серверов ведёт ещё один сервис AWS – Amazon Cloudwatch. С помощью данного сервиса можно создавать разного рода проверки – checks, при помощи которых контролируются важнейшие показатели работы ОС. При использовании EC2 осуществляется почасовая оплата, некоторые подсервисы, такие как EBS имеют биллинг с оплатой по месяцам. Для каждого подсервиса предусматривается свой отдельный биллинг по заведомо утверждённой цене в час или в месяц. Так же у EC2 инстансов существует так называемая резервация (Reservation) – оплачивается сразу 3-4 месяца работы сервера, после чего, час работы сервера стоит в ~1,5 раза дешевле. Резервации удобно использовать, если EC2 используется на постоянной основе – экономия на лицо.

Таким образом, использование связки CI/CD и AWS Web Services в данном приложении ускорило разработку и тестирование. Такой подход предоставляет возможность развёртывания новых версий в кратчайшее время, кроме этого, за счёт гибкого использования ресурсов данная связка экономит денежные затраты на каждом шаге интеграции и разработки. Использование сервисов, таких как Github Actions, Cloud Formation предоставляют возможность сохранять конфигурации CI/CD и, в случае заморозки проекта, позволяют в кратчайшие сроки развернуть аналогичную конфигурации спустя какое-то время без надобности реализации дополнительной конфигурации.

ЛИТЕРАТУРА

1. Architectural Styles and the Design. Диссертация Roy Thomas Fielding. [Электронный ресурс]. / ics.uci.edu. – Режим доступа: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf/. – Дата доступа: 20.09.2020.
2. Философия DevOps. Искусство управления IT // J.Davis – O'Reilly Media, Inc., 2017. – 213 с.
3. Zaharia M. Spark: The definitive guide // Zaharia M, Chambers B. – O'Reilly Media, Inc, 2018. – 606 с.