УДК 621.326

Луцків А.М. доцент, канд.техн.наук, Шевчук А.М., студент, гр. СІм-52
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ШЛЯХИ ВДОСКОНАЛЕННЯ ПРОЦЕСУ НЕПЕРЕРВНОЇ ІНТЕГРАЦІЇ В ХОДІ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Lutskiv A. M. Assoc. Prof, Ph.D., Shevchyk A.M. student, group СІм-52

WAYS TO IMPROVE THE PROCESS OF CONTINUOUS INTEGRATION IN THE DEVELOPMENT OF SOFTWARE

Процес створення програмного забезпечення має низку етапів життєвого циклу. З метою підвищення якості процесу розроблення програмного забезпечення у 1999 р., Мартіном Фаулером було запропоновано використання концепції неперервної інтеграції[1] (англ. Continuous Integration) — розроблення програмного забезпечення, яке полягає у виконанні частих(періодичних) автоматизованих збирань/компіляцій (build) проекту для якнайшвидшого виявлення та вирішення інтеграційних проблем. Дана концепція передбачає слідування певним правилам та використання спеціалізованих засобів. На сьогодні до найпопулярніших засобів неперервної інтеграції належать:

- Jenkins/Hudson (проект для неперервної інтеграції з відкритим вихідним кодом, написаний на Java);
- TeamCity (серверне програмне забезпечення від компанії JetBrains, написане на мові Java, білд-сервер для забезпечення неперервної інтеграції);
- Travis CI (розподілений веб-сервіс для складання та тестування програмного забезпечення, що використовує GitHub в якості хостингу коду);
- IBM Bluemix (реалізація відкритої хмарної архітектури IBM, заснована на Cloud Foundry, яка дозволяє швидко створювати, розгортати і адмініструвати хмарні додатки).

У рамках дослідження використовується система Jenkins для забезпечення розроблення web-сервісу у рамках технології JavaEE. Яка має можливість шляхом використання різноманітних розширень (plugins) розширювати свій функціонал.

Сформулюємо перелік рекомендацій, яких має дотримуватись адміністратор-розробник (DevOps), який займається організацією системи неперервної інтеграції[2]:

1. Виконувати збирання проекту кожного разу при зміні в репозиторії (виконанні git push). Такий підхід дає змогу забезпечити отримання протестованого і готового до роботи програмного забезпечення при кожній зміні в репозиторії.
2. Внесення змін у вихідний код програми після виконання кожного завдання. Це дозволить уникнути помилок побудови та складних інтеграційних помилок, які виникають при внесенні змін одразу для декількох завдань.
3. Створити шаблон автоматизації всіх робіт по створенню програмного забезпечення з вихідного коду, виключити ручне налаштування, створити сценарії, які відокремлені від IDE. Такий підхід дозволить полегшити процес постійних збирань і дозволить уникнути численних ручних налаштувань.
4. Централізувати всі залежні бібліотеки. Такий підхід дозволить уникнути повторення одних і тих же бібліотек від проекту до проекту.
5. Створити просту, але чітко визначену структуру каталогів для оптимізації програмного забезпечення і збільшення крос-проектної передачі знань.
6. Створити шаблон віддаленого розгортання проекту для запуску в різних середовищах і на різних платформах. Такий підхід дозволить полегшити процес

запуску проекту, дозволить уникнути необхідності розгортання безпосередньо для кожного окремого середовища.

7. Налаштувати автоматизований моніторинг роботи сервера неперервної інтеграції командою розробників. Такий підхід дозволить команді розробників реагувати миттєво на проблеми, які виникають під час розроблення програмного забезпечення. Такий моніторинг може здійснюватися розробкою за допомогою: e-mail, RSS, SMS, X10, Monitors, Web Notifiers, Campfire, Slack, HipChat.

8. Виконувати виправлення помилок, які виникли під час збирання, одразу після виникнення. Такий підхід дозволить не допустити виникнення цих помилок в подальших збірках і дозволить уникнути накопичення несправностей, що може вплинути на погіршення якості програмного забезпечення.

9. Генерувати документацію розробника через відповідні проміжки часу для перевірених змін у вихідному коді. Такий підхід дозволить уникнути періодичного ручного генерування документації, що може бути складним і трудомістким процесом.

10. Переконатися, що всі побудови і розгортання можна запустити однією командою. При такому підході користувачу необхідно набрати одну команду, а всі дії будуть виконуватися системою неперервної інтеграції. Такий підхід дозволяє максимально спростити процес розгортання, а, відповідно, й зробити його максимально надійним, зокрема уникнути ручного копіювання файлів, зміни конфігураційних файлів, перезавантаження сервера, встановлення паролів та інших потенційно помилкових дій.

11. Виконувати побудову проекту на окремому віддаленому комп'ютері або хмарному сервісі. Такий підхід виключає будь-які залежності програмного забезпечення від середовища.

12. Доступ до репозиторію має тільки авторизований і кваліфікований персонал. Такий підхід дозволить створити відносно захищену конфігурацію.

13. Створити полегшену версію бази даних (з мінімально достатньою кількістю записів, щоб перевірити функціональність). Використовувати цю ж версію, або її копії для всіх розробників. Використовувати цю базу даних в середовищах розробки для прискорення виконання тестів. Даний підхід дозволить уникнути використання реальних (production) баз даних і знизить ризик пошкодження даних.

14. Створити автоматизовані тести для кожного блоку програмного забезпечення. Даний підхід дозволить уникнути ручного тестування, регресійних тестів.

Використання систем неперервної інтеграції є особливо актуальним при використанні гнучких методологій розроблення ПЗ, зокрема при TDD-методології. Вона може використовуватись як при створенні комерційного закритого програмного забезпечення, так і написанні проектів з відкритим вихідним кодом. Може бути використана при створенні різних типів програм: мобільних, веб-додатків, десктоп і вбудованих. Система неперервної інтеграції може бути використана для різноманітних мов програмування, як інтерпретовних так і тих, які передбачають формування виконуваного бінарного або байт-коду компілятором.

Література

1. Martin Fowler. Continuous Integration // MartinFowler.com [Электронний ресурс]. URL: https://martinfowler.com/articles/continuousIntegration.html (дата звернення – 20.04.2017)

2. Paul Duvall. Continuous Integration: Patterns and Anti-Patterns // DZone [Электронний ресурс]. URL: https://dzone.com/refcardz/continuous-integration (дата звернення – 24.04.2017)