УДК 004.4

Куліш В. В. магістрант
Науковий керівник: Єфіменко А. А., канд. техн. наук, доцент,
завідувач кафедри КІ та КБ
Державний університет «Житомирська політехніка»

МЕТОДОЛОГІЯ CI/CD ТА ЇЇ ВПЛИВ НА СУЧАСНУ РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Безперервна інтеграція (Continuous Integration, CI) і безперервне постачання (Continuous Delivery, CD) є культурою, набором принципів і практик, які дозволяють розробникам частіше і надійніше розгортати зміни програмного забезпечення. Обидва терміни дуже тісно пов'язані з професією DevOps і являються основними термінами їх еволюції.

У різних сферах – від банківських та фінансових послуг до розд-рібної торгівлі, від державного управління та муніципального управління до туризму і розваг – розробка ПЗ стає найважливішою частиною роботи багатьох організацій. Доступ до товарів та послуг та їх підтримка здійснюються за допомогою додатків та онлайнсервісів, а внутрішні комп'ютерні системи потрібні для безперебійної роботи.

Термін DevOps поєднує у собі розробку (DEVelopment) та операційну діяльність (OPerationS). Це наголошує на необхідності інтеграційної роботи всіх команд для ефективної доставки працюючого ПЗ. Для створення коду, готового до розгортання у виробничому середовищі, розробники повинні мати можливість відстежувати всі проміжні кроки від розробки до релізу. Необхідно подолати роз'єднаність та організувати спільну роботу з командами, що відповідають за QA, безпеку та інфраструктуру з урахуванням їхнього вкладу в загальний процес [1]. Основні принципи CI/CD наведені нижче: сегрегація відповідальності зацікавлених сторін; зниження ризику; короткий цикл зворотнього зв'язку; реалізація середовища.

Перший пункт списку вирішує проблему комунікації та розставляє чіткі зони відповідальності між учасниками. Розробники і дизайнери проектують бізнес-логіку, а також забезпечують позитивний досвід взаємодії з готовою системою. Інженери за якістю вводять наскрізні функції та приймальні тести, DevOps-інженери організують логістику коду, а користувачі дають зворотний зв'язок за результатами використання системи.

Принцип зниження ризику потребує щоб кожна група учасників розробки мінімізувала всі можливі ризики при проходженні продукту через стадії життєвого циклу (контроль цілісності бізнес-логіки, користувальницького досвіду, оптимізація зберігання та обробки даних, міграції та ін.).

Принцип циклу зворотнього зв'язку дає змогу вирішити інциденти при виникненні помилок, чи проблем з новим функціоналом. Щоб додавати в продукт новий функціонал швидше за конкурентів, необ-хідно прагнути до автоматизації складання та тестування коду. Однак, у ситуаціях, коли для вирішення потрібна участь людини, автоматизація може лише нашкодити. Для таких ситуацій рекомендується скорочувати кількість інформаційних посередників, забезпечуючи короткий цикл зворотного зв'язку.

За реалізацію середовища відповідають DevOps інженери. Також команді розробки потрібне єдине робоче оточення для контролю версій і побудови допоміжних гілок для контролю якості, прийнятності, масштабованості та відмовостійкості виробленого коду. [2]

Основними інструментами для реалізації CI/CD можуть бути: GitLab, Docker, Jenkins, Maven, Ansible, Kubernetes і т.д. Підводячи підсумки можна виділити певні переваги та недоліки CI/CD підходу:

Безперечно такі методи забезпечуть оперативність виведення нового функціоналу продукту. Як правило, це лічені дні чи тижні. У той же час, при класичному підході до розробки клієнтського ПЗ на це може бути витрачено рік і навіть більше.

Якість продукту підвищується за допомогою паралельного тестування функціональних блоків майбутньої системи. Вузькі місця та критичні моменти фіксуються та відпрацьовуються ще на ранніх ста-діях циклу.

Проте керівники проектів помилково приймають методологію як панацею і прагнуть запровадити її у всі свої розробки. Нестача досвіду призводить до ускладнення робіт з IT-продуктами компанії.

Отже, CI/CD можна назвати кращою методикою розробки, яка відповідає задачам сьогодення і поступово стає звичним терміном в усіх IT компаніях. Розглядаючи весь процес розробки та доставки ПЗ в цілому та оптимізуючи кожен його етап, можна швидше створювати продукт і та отримувати зворотний зв'язок. Тим самим забезпечується розвиток та покращення продукту.

Список використаних джерел

1. The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems/Thomas A. Limoncelli , Strata R. Chalup , Christina J. Hogan, 2014. – 560 с. – («InformIT telecom»).
2. Achieving DevOps: A Novel About Delivering the Best of Agile, DevOps, and Microservices/ Dave Harrison, Knox Lively, 2019 - 496 с. – (Apress, Berkeley, CA).