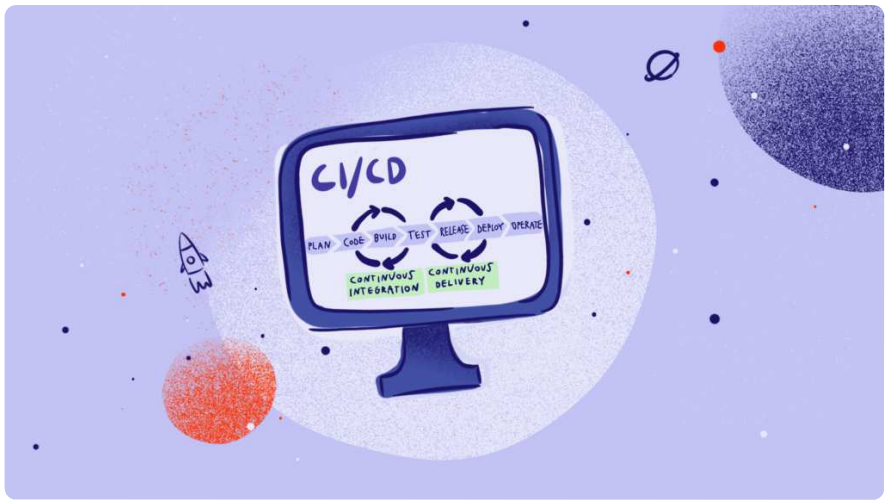
← GitLab

Korzyści biznesowe z CI/CD w GitLab



GITLAB

18 września 2019 · 21 min czytania

Radoslaw Kosiec

Wstęp

Co to Continuous Integration i Delivery?

Różnice pomiędzy continuous integration i continuous delivery

Workflow w CI/CD w GitLab

Zalety continuous integration i continuous delivery

Wpływ CI/CD na biznes

Co to Continuous Integration i Delivery?

W dzisiejszych czasach technologia rozwija się w niesamowicie szybkim tempie, jednocześnie zwracając się w kierunku automatyzacji wszelkich procesów. Rozwój ten obejmuje również podstawową

jednostkę obszaru IT, jaką jest produkcja oprogramowania. Systemy informatyczne są wykorzystywane w każdej działalności, firmie czy biznesie – niezależnie od branży, a automatyzacja jest kluczem do optymalizacji pracy i podniesienia wydajności zespołów. W przypadku programistów narzędziem usprawniającym tworzenie oprogramowania są procesy Continuous Integration i Continuous Delivery. Razem z Continuous Deployment stanowią elementy metodyki DevOps, a ich zadaniem jest automatyzacja procesu integracji i dostarczania zawsze działających wersji oprogramowania.

Różnice pomiędzy continuous integration i continuous delivery

Chociaż te pojęcia zazwyczaj występują w parze, nie są ze sobą tożsame i często są mylone, nawet przez programistów i osoby, które wykorzystują te funkcje w codziennej pracy. Zazwyczaj CI kojarzone jest z pracą nad programowaniem, natomiast CD to raczej zadanie administratora. W rzeczywistości są to kolejne, następujące po sobie zautomatyzowane etapy procesu tworzenia oprogramowania, który obejmuje nie tylko samo kodowanie, ale także przeniesienie aplikacji do środowiska testowego czy dostarczenie działającej aplikacji klientowi.

Przyjrzyjmy się zatem bliżej różnicom między procesami continuous integration i continuous delivery.

Co to jest continuous integration?

Continuous integration, czyli ciągła integracja to nieustanne, automatyczne buildowanie programu na dedykowanym serwerze po każdym commicie. Ta praktyka DevOps jest chętnie wykorzystywana w procesie tworzenia oprogramowania, ponieważ pozwala zespołowi programistów na szybkie wykrycie oraz ustalenie lokalizacji błędów, konfliktów i innych potencjalnych problemów. Najprościej mówiąc ciągła integracja to automatyczne wyzwalanie kompilacji.

W myśl zasad CI programista powinien stale integrować zmiany w kodzie ze zmianami wprowadzanymi przez innych członków z resztą zespołu – przy manualnym buildowaniu proces ten zabierałby bardzo

dużo czasu, a przy tym byłby bardzo obciążający dla środowiska programowania. Przy wykorzystaniu repozytorium GIT, automatyczna integracja rozpoczyna się po podaniu komendy „git push”. Następnie na dedykowanym serwerze następuje zautomatyzowany proces buildowania aplikacji. Później uruchamiany jest zestaw testów zapisanych w kodzie, których zadaniem jest potwierdzenie, że wprowadzone do kodu modyfikacje nie są przyczyną błędów w działaniu całego programu.

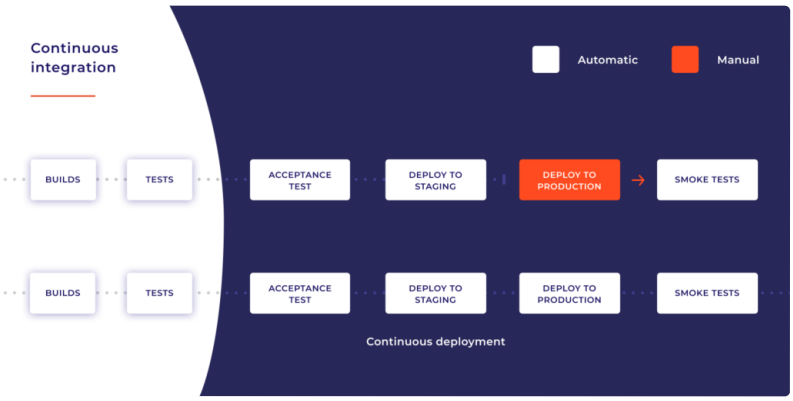
Jeśli podczas wykonywania zadań CI integracja nie powiedzie się, oznacza to, że wprowadzone zmiany wymagają priorytetowej naprawy, aby program mógł się zbuildować. Nowe funkcje i kolejne zmiany powinny być dodawane tylko do działającego programu. Nie tylko prędkość systemu jest ważna – liczy się również jakość.

Cały proces integracji można podzielić na trzy główne fazy. Pierwszy etap to Push – kiedy wprowadzone zmiany są wysyłane na serwer. Kolejny etap to Test – jak wspomniane zostało wcześniej, ułatwia on lokalizację błędów. Ostatnim etapem Continuous Integration jest Fix, czyli naprawa wykrytych problemów i uszkodzonych funkcji.

Co to jest continuous delivery?

Continuous Delivery jest kolejnym krokiem w automatyzacji procesu produkcji oprogramowania i następuje po pozytywnym przejściu fazy testów. Ciągłe dostarczanie polega na automatycznym dostarczaniu działającej wersji programu do ostatniego środowiska przed produkcją. Automatyczne dostarczenie wyzwalane jest po udanym merge’u gałęzi kodu. Mówiąc o działającej wersji aplikacji mamy oczywiście na myśli aplikację, która jest w stanie się zbudować, ale niekoniecznie posiada wszystkie wymagane funkcjonalności, a jej działanie nie musi być do końca prawidłowe.

Ciągłe dostarczanie to etap procesu wytwarzania oprogramowania, który poprzedza Continous Deployment, czyli ciągłe wdrażanie na produkcję. Do produkcji powinny trafiać tylko w pełni funkcjonujące wersje aplikacji, stąd konieczność sprawdzenia jej na wcześniejszych etapach.

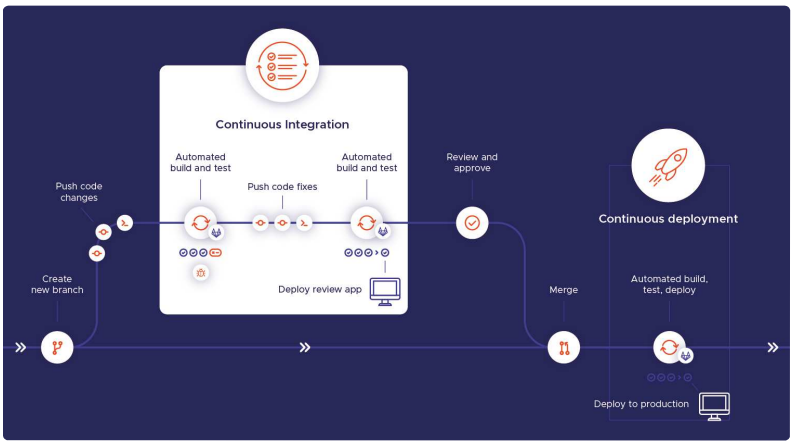


Workflow w CI/CD w GitLab

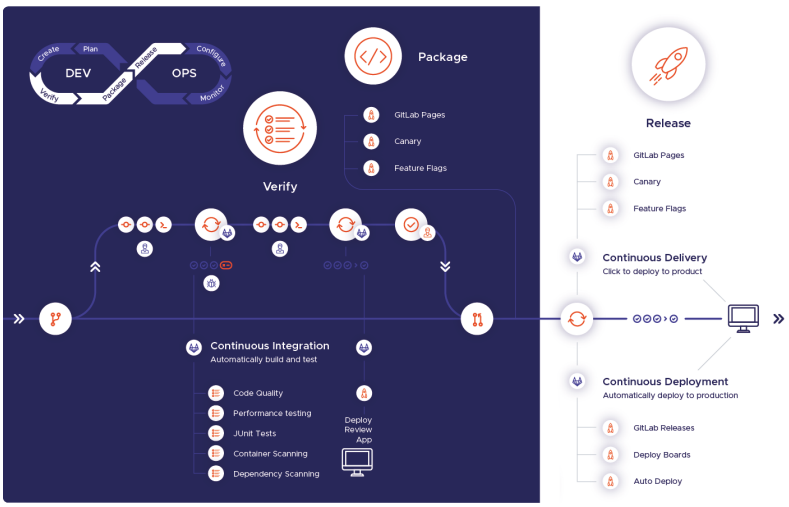
Aby cały proces był rzeczywiście zautomatyzowany, niezbędne jest wdrożenie odpowiednich narzędzi DevOps. Dobrze dobrane narzędzie usprawni pracę programistów i podniesie wydajność całego zespołu, co przełoży się także na szybsze ukończenie projektu. Takim narzędziem jest GitLab. Usługa CI/CD w GitLab dobrze pasuje do standardowych przepływów pracy zespołów programistycznych.

Rozważmy następującą sytuację: po podjęciu decyzji o sposobie implementacji kodu programista rozpoczyna wprowadzanie zmian w lokalnym repozytorium. Po zacommitowaniu zmian w odpowiedniej gałęzi (przekazaniu ich do zdalnego repozytorium) GitLab uruchamia automatyczny potok (ang. pipeline) CI/CD dla projektu. W ten sposób GitLab uruchamia skrypty budujące i testujące aplikację, umożliwia przegląd zmian według merge'ów, tak samo jak w przypadku lokalnego hosta. Po udanej implementacji użytkownik otrzymuje sprawdzony i zaakceptowany kod, może zmerge'ować gałąź funkcjonalności z gałęzią główną, natomiast po merge'u usługa CI/CD w Gitlab automatycznie dostarcza działającą wersję aplikacji do środowiska testowego, a jeżeli aplikacja pomyślnie przejdzie testy to GitLab automatycznie dodaje zmiany do środowiska produkcyjnego.

Oczywiście jeżeli pojawią się błędy to zespół może łatwo powrócić do poprzedniego stanu i wprowadzić odpowiednie poprawki.



Jeśli przyjrzymy się bliżej podstawowemu przepływowi pracy, można zobaczyć bardziej szczegółowo wszystkie funkcje dostępne w GitLab na każdym etapie cyklu życia DevOps (weryfikacja, pakiet, wydanie). Pełny cykl procesu DevOps ilustruje poniższy schemat:



DevOps to bardzo złożony mechanizm, który obejmuje wiele funkcji i czynności na każdym etapie produkcji oprogramowania.

1. Weryfikacja (verify):
   automatyczne buildowanie i testowanie programu (Continuous Integration),
   analiza jakości kodu źródłowego (GitLab Code Quality),
   określenie wpływu wprowadzonych zmian w kodzie (Browser Performance Testing),
   wykonanie serii testów (Containter Scanning, Dependency Scanning, JUnit),
   wdrożenie zmian za pomocą Review Apps, aby uzyskać podgląd zmian na każdej gałęzi.

2. Artefakty (package):
   przechowywanie obrazów Docker (Container Registry),
   przechowywanie pakietów NPM (NPM Registry),
   przechowywanie artefaktów Maven (Maven Repository).
3. Wydanie (release):
   automatyczne wdrożenie aplikacji na produkcję (Continuous Deployment),
   funkcja Continuous Delivery i manualne wdrożenie aplikacji na produkcję,
   wdrażanie statycznych stron internetowych (GitLab Pages),
   udostępnianie nowych funkcji tylko w części podów i udostępnianie części użytkowników czasowo wdrożonych funkcji (Canary Deployments),
   wdrożenie funkcji z Feature Flags,
   możliwość dodania opisów wydań (release notes) do każdego repozytorium Git (GitLab Releases),
   podgląd aktualnego stanu każdego środowiska CI działającego w klastrze Kubernetes (Deploy Boards),
   wdrożenie aplikacji na produkcję w klastrze Kubernetes (Auto Deploy).

Usługa CI/CD w GitLab CI/CD pozwala dodatkowo na:

łatwe zaplanowanie całego cyklu produkcji aplikacji z usługą Auto DevOps,
wdrażanie aplikacji do różnych środowisk,
instalację własnego GitLab Runnera,
skonfigurowanie potoków,
sprawdzenie aplikacji pod kątem podatności (Security Test reports).

Zalety continuous integration i continuous delivery

Szybsze wdrażanie nowości = konkurencyjność na rynku

Firma, która korzysta z technologii CI/CD jest w stanie szybciej wdrożyć aplikację, niż firma, która nie stosuje tego rozwiązania. Automatyzacja zawsze korzystnie wpływa na rozwój biznesowy i technologiczny. Firmy, które rozumieją znaczenie CI/CD, wyznaczają tempo wdrażania innowacji na rynku dla wszystkich innych organizacji.

Wdrożony kod to generator zysków

Nie oszukujmy się – kod, który jest wdrożony na produkcję przynosi zyski, w przeciwieństwie do tego, który oczekuje w kolejce na manualną weryfikację. Firmy, które stosują technologię CI/CD mogą szybciej umieszczać kod w środowisku produkcyjnym, co przekłada się na wzrost zysków. Programiści nie muszą się martwić o jakość swojego kodu, ponieważ został on automatycznie sprawdzony w procesie ciągłego dostarczania i ciągłe wdrożenie będzie miało miejsce tylko w przypadku kodu o określonym standardzie. Minimalizuje to ryzyko błędów ludzkich i zmniejsza opóźnienia całego procesu.

Wysoka jakość kodu

Automatyzacja sprawia, że programiści mogą skupić się na programowaniu, a nie na środowisku produkcyjnym, w którym kod sprawdza się bez konieczności manualnego wyzwalania tego procesu. Ustalenie określonych standardów kodu sprawia, że kod złej jakości nie ma szans trafienia na produkcję lub do klienta.

Wpływ CI/CD na biznes

Zróżnicowane rozwiązania inżynieryjne

Wprowadzenie zróżnicowanych rozwiązań inżynieryjnych znacznie obniża koszty okazjonalne. Firmy mogą sobie pozwolić na określoną liczbę inżynierów jednocześnie, a niezróżnicowane systemy wymagające obszernych nakładów na utrzymanie to mniej inżynierów pracujących nad dochodowymi projektami. Zróżnicowanie oznacza także nieustanny rozwój.

Terminowa realizacja projektu

Automatyzacja procesów integracji, dostarczania i wdrażania oprogramowania, znacząco obniża koszty projektu i przyspiesza czas jego realizacji. Dzięki szybkiemu wykrywaniu błędów programiści mogą zareagować w odpowiednim czasie i nie muszą martwić się o wykonanie integracji, ponieważ w funkcji CI/CD integracja wykonuje się sama.

Lepsza produktywność zespołu

Każdy przestój w realizacji projektu to utracone przychody. Aby tego uniknąć programiści spędzają czas na pracy nad infrastrukturą i konfiguracją programu, zamiast poświęcić czas na dopracowanie logiki biznesowej. W obu przypadkach programiści są mniej produktywni i wykonują pracę, która wykracza poza ich kompetencje. Automatyzacja sprawia, że programiści nie muszą więcej zajmować się zadaniami inżynierów DevOps, a to podnosi motywację całego zespołu.

Czy chcecie dowiedzieć się więcej o możliwościach Auto DevOps z GitLab? Zapraszamy na webinar, w którym nasz inżynier DevOps

Radoslaw Kosiec

Wspieram Deviniti w wielu obszarach już od 2011 r. Jestem dumny z tego, że mogę wspierać firmę w zgodzie z zasadami, w które zawsze wierzyłem – w pracy z Klientem najważniejsza jest troska o jego rzeczywiste potrzeby. Dzięki takiemu podejściu zaufały nam tysiące firm z całego świata. Doradzam firmom w zakresie narzędzi dla DevOps, Service Desk, obsługi klienta we wszystkich kanałach komunikacji, zarządzania zadaniami i projektami w oparciu na narzędzia naszych partnerów oraz tworzone przez nasz zespół pod indywidualne potrzeby naszych Klientów.