← GitLab

GitLab w rozwoju oprogramowania – część 2: Code Review

GITLAB

21 maja 2019 · 13 min czytania

Armin Orlik

Wstęp
Pierwszy Commit
Merge Request
WIP MR
Przegląd
Buduj, testuj, wdrażaj
Integracja Koding
Przykłady użycia
Feedback: Analiza Cykli

Pierwszy Commit

Już w samym opisie commita możemy umieścić numer zgłoszenia, który będzie z nim powiązany. Tworzymy w ten sposób połączenie pomiędzy dwoma etapami procesu tworzenia oprogramowania: zgłoszeniem samym w sobie oraz commitem z przypisanym zgłoszeniem.

Jeśli zgłoszenie i kod znajdują się w tym samym projekcie, możesz po prostu dodać #xxx do opisu commita, zastępując xxx numerem zgłoszenia. W innym przypadku możesz dodać do zgłoszenia cały URL (https://gitlab.com/<nazwa_uzytkownika>/<projekt>/zgloszenia/<xxx>).

Gitlab.com oczywiście należy zastąpić URL-em Twojej instancji GitLab.

Mała uwaga: *Połączenie pierwszego commita ze zgłoszeniem znacznie poprawi Twoje możliwości śledzenia procesu przy pomocy narzędzia GitLab Cycle Analytics. Narzędzie to zmierzy czas spędzony na planowaniu wdrożenia, czyli czas pomiędzy powstaniem zgłoszenia a pierwszym commitem.*

Merge Request

Kiedy prześlemy dane do funkcjonalnego brancha, GitLab rozpozna zmiany i poprosi o stworzenie Merge Request'a (MR). Każdy MR ma tytuł (podsumowujący wdrożenie) i opis wspierany przez Markdown. W opisie możemy zwięźle przedstawić jakie zmiany wnosi dany MR, a także wspomnieć o powiązanych zgłoszeniach i MR-ach (tworząc między nimi połączenie).

Jeśli commit lub MR rozwiązuje problem zgłoszenia, możemy stworzyć wzory zamknięcia zgłoszeń. Zastosowanie opisanego rozwiązania pozwoli na automatyzację tego procesu w sytuacji, gdy wspomniany commit lub MR łączy się z domyślnym branchem w projekcie.

Rozważmy poniższy przykład:

MR o takim opisie wykona następujące operacje:

po wykonaniu merge'a, zamknie zgłoszenia #xxx oraz
https://gitlab.com/<uzytkownik>/<projekt>/zgloszenia/<xxx>,
wyświetli obrazek,
wyśle mailem powiadomienie użytkownikom @Maria, @Janina, i @Janusz.

Możemy przypisać MR do siebie, dopóki nie zakończymy pracy, a następnie przypisać osobę, która napisze recenzję. Możemy również wielokrotnie przypisywać MR-a do różnych osób, na przykład gdy potrzebujemy kilku recenzji.

MR-y można także oznaczać i dodawać do milestone'ów, aby łatwiej było je organizować i nadawać priorytety.

Gdy dodamy lub edytujemy plik, a następnie wykonamy commit na nowego brancha z interfejsu zamiast z wiersza poleceń, nadal mamy możliwość stworzenia nowego merge request'a. Wystarczy zaznaczyć pole **start a new merge request with these changes**, by GitLab automatycznie stworzył nowego MR-a po wykonaniu commita.

Mała uwaga: *Musimy pamiętać o dodaniu wzoru zamknięcia zgłoszeń do naszego MR-a, aby mieć możliwość śledzenia procesu przy pomocy narzędzia GitLab Cycle Analytics. Bada ono etap "kodowania", czyli czas pomiędzy push'em w pierwszym commicie a merge request'em, który jest związany z tym commitem.*

Nowością jest obecnie rozwijane narzędzie Review Apps, które pozwala umieścić aplikację w dynamicznym środowisku, dając możliwość podglądu zmian na bazie nazwy brancha dla danego merge requesta. Zapraszam do zapoznania się z przykładem wykorzystania.

WIP MR

WIP MR czyli **Work in Progress Merge Request** (wersja robocza Merge Requesta) to sposób GitLab'a na zapobieganie przedwczesnym merge'om MR-ów. Wystarczy dodać prefiks WIP przed nazwą MR-a, a nie będzie on mergowany do momentu jego usunięcia.

Gdy Twoje zmiany są gotowe do merge'a usuń prefiks WIP. Możesz to zrobić ręcznie edytując zgłoszenie albo użyć skrótu dostępnego pod opisem MR-a.

Nowością jest prefiks WIP, który można szybko dodać do merge requesta przy pomocy komendy /wip. Wystarczy wpisać ją w komentarzu lub w opisie MR-a.

Przegląd

Po stworzeniu merge requesta możemy zebrać feedback od członków zespołu lub współpracowników. Interfejs GitLab'a wyposażony jest w opcje umożliwiające dodawanie komentarzy w linii kodu, odpowiadanie oraz ich zamykanie.

Możemy również uzyskać link do pojedynczej linii kodu klikając na jej numer.

Historia commitów jest dostępna z poziomu interfejsu i umożliwia nam również śledzenie zmian pomiędzy wersjami danego pliku.

Jeśli trafimy na jakikolwiek konflikt podczas wykonywania merge'a, możemy go łatwo rozwiązać z poziomu interfejsu użytkownika albo wyedytować plik ręcznie:

Buduj, testuj, wdrażaj

GitLab CI to zaawansowany mechanizm Continous Integration, Continuous Deployment oraz Continuous Delivery, którego możemy używać do wykonywania skryptów w pożądany przez nas sposób.

Sercem narzędzia jest plik Yaml o nazwie .gitlab-ci.yml, znajdujący się w repozytorium projektu. Aby skorzystać z szablonów CI, wystarczy dodać w interfejsie webowym nowy plik o nazwie .gitlab-ci.yml. Uzyskamy wówczas dostęp do rozwijanej listy z wieloma szablonami o

różnych zastosowaniach, z której możemy wybrać najlepszy dla naszego projektu.

Integracja Koding

Opcja Koding integration, pozwala uruchomić w chmurze całe środowisko produkcyjne. Takie rozwiązanie daje nam możliwość wykonania check out'u lub merge request'a, a cała operacja odbywa się w całkowicie zintegrowanym środowisku.

Przykłady użycia

Narzędzie GitLab CI możemy wykorzystać w następujący sposób:

budowa dowolnych generatorów stron statycznych oraz wdrażanie stron przy pomocy GitLab Pages,
wdrażanie strony na środowiska pre-produkcyjne i produkcyjne,
budowa aplikacji na system iOS,
budowa i wdrożenie Docker Image przy użyciu narzędzia GitLab Container Registry.

Więcej przykładowych projektów GitLab CI prezentujemy w tym miejscu.

Feedback: Analiza Cykli

Jeśli korzystamy z GitLab Workflow, mamy możliwość zbierania informacji zwrotnej przy pomocy narzędzia GitLab Cycle Analytics. Mierzy ono czas jakiego potrzebował zespół na przejście od pomysłu do produkcji, uwzględniając poszczególne etapy całego procesu:

zgłoszenie: czas między powstaniem zgłoszenia a przypisaniem go do milestone'a albo do listy na tablicy zgłoszeń,
planowanie: czas mierzony od końca zgłoszenia do pierwszego commita,

kodowanie: czas od końca planowania do powstania merge requesta,

testy: czas wykonania całego procesu CI dla danego merge requesta,

przegląd: czas pomiędzy powstaniem merge requesta a mergem,

staging: czas pomiędzy mergem a wdrożeniem do produkcji,

produkcja: suma wszystkich etapów – czas pomiędzy powstaniem zgłoszenia a wdrożeniem kodu do produkcji.

źródło: https://about.gitlab.com/2016/10/25/gitlab-workflow-an-overview/

Armin Orlik

Inżynier i konsultant w obszarach DevOps, Automation, Cloud, konteneryzacja. Bogate doświadczenie we wdrażaniu i utrzymaniu praktyk DevOps. Obecnie pasjonat rozwiązań chmurowych w szczególności bazując na AWS. Interesuje się szeroko pojętą automatyzacją oraz projektowaniem mikroserwisów i aplikacji serverless oprogramowania.