



Proces ciągłej integracji Git + Gerrit + Jenkins

autor : Maciej Pieszala

Plan prezentacji

- definicja procesu ciągłej integracji
- zalecane praktyki
- plusy i minusy procesu ciągłej integracji
- narzędzia:
 - git
 - gerrit
 - jenkins
- podsumowanie
- bibliografia

definicja procesu ciągłej integracji

problemy w pracy nad projektami:

- duże repozytorium kodu,
- wielu programistów równolegle pracujących nad projektem,
- jak wdrożyć skończony projekt,
- szybki rozwój aplikacji
a duża liczba błędów,
- automatyzacja procesów

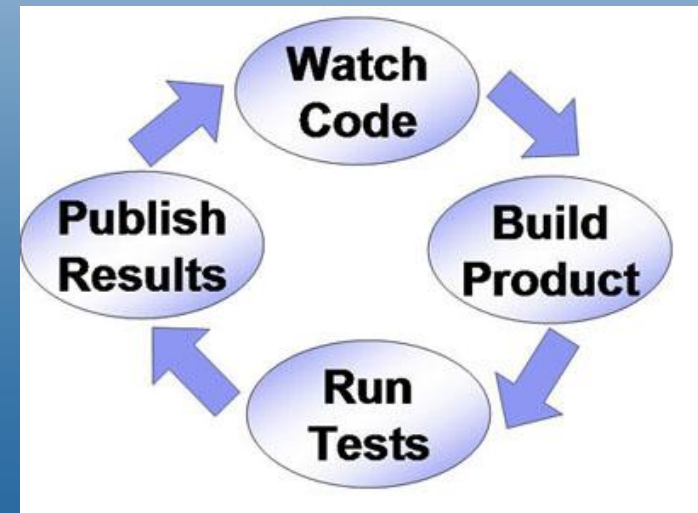


definicja procesu ciągłej integracji

Ciągła integracja to praktyka tworzenia oprogramowania, w której członkowie zespołu często integrują swoją pracę z całością projektu/aplikacji, zazwyczaj co najmniej raz dziennie - co prowadzi do wielu integracji dziennie.

Każda integracja jest weryfikowana przez automatyczne budowania (w tym testy oraz review kodu) w celu wykrycia błędów integracji, tak szybko jak to możliwe.

W wielu zespołach okazuje się, że podejście to prowadzi do znacznego zmniejszenia problemów z integracją i pozwala na tworzenie spójnego oprogramowania w szybszym tempie eliminując jednocześnie wiele błędów.



zalecane praktyki

- review kodu
- utrzymanie kodu w repozytorium
- dbanie o stan repozytorium
- automatyczne wykonywanie builda
- uruchamianie testów jednostkowych i funkcjonalnych
- codzienna synchronizacja kodu
- każda zmiana powinna podlegać kompilacji
- środowisko testowe odwzorowujące środowisko produkcyjne
- publiczne pokazywanie wyników kompilacji
- automatyczne wdrażanie



zalety

- po wykryciu błędu istnieje łatwy sposób powrotu do poprawnego kodu, bez marnowania czasu, debugowania,
- programiści wykrywają i rozwiązują problemy integracji w sposób ciągły – unikając chaosu w momencie wdrożenia,
- wczesne ostrzeganie o złym / niezgodnym kodzie,
- wczesne ostrzeganie przed konfliktami w zmianach,
- natychmiastowe i automatyczne wykonywanie testów jednostkowych na wszystkich zmianach,
- łatwe znalezienie zmiany, w której jest błąd i wyeliminowanie go,
- stała dostępność najaktualniejszej zbudowanej wersji,
- natychmiastowe powiadamianie programistów o wpływie jaki wywiera ich zmiana na całość systemu,
- wymaganie częstej kontroli kodu zmusza programistów do tworzenia modularnego, mniej skomplikowanego kodu,
- statystyki automatycznie generowane w procesie zachęcają programistów do większej dbałości o jakość kodu.

wady

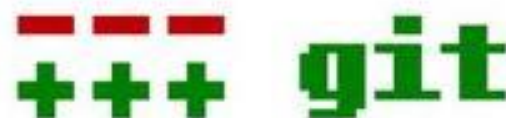
- nakład pracy wymagany do wstępnej konfiguracji narzędzi, utrzymanie i zmiany konfiguracji w narzędziach,
- wymagane dobrze rozwinięte archiwum testów, duże pokrycie kodu testami jednostkowymi, testy funkcjonalne, wysoka jakość testów,
- refaktoryzacja dużej funkcjonalności systemu może być kłopotliwa ze względu na szybko zmieniający się kod,
- koszt sprzętu wymaganego do automatycznego budowania i testowania aplikacji.

narzędzia



git

Git – rozproszony system kontroli wersji. Stworzył go Linus Torvalds jako narzędzie wspomagające rozwój jądra Linux. Git stanowi wolne oprogramowanie i został opublikowany na licencji GNU GPL w wersji 2.



git vs svn - różnice

- Git jest znacznie szybszy niż Subversion,
 - Subversion pozwala sprawdzić tylko poddrzewa repozytorium
- Git wymaga klonu całego repozytorium (w tym historii) i utworzenia kopii roboczej, która odzwierciedla co najmniej podzbiór elementów pod kontrolą wersji,
- Git repozytoria są znacznie mniejsze niż Subversion (projektu Mozilla, 30x mniejsze),
 - Git został zaprojektowany, aby być w pełni rozproszony od samego początku, pozwalając jednocześnie programiście na pełną kontrolę w lokalnym repo,
 - branche Git są prostsze i wymagają mniej zasobów niż ciężkie Subversion,
 - branche Git posiadają całą historię zmian,

git vs svn - różnice

- Git zapewnia lepszą kontrolę nad łączeniem zmian,
- formaty plików w Gitcie są prostsze, co ułatwia naprawę repo i powoduje że błędy zdarzają się bardzo rzadko,
- tworzenie kopii zapasowych repozytoriów Subversion jest potencjalnie prostsze (centralne repo),
- Subversion UI jest bardziej dojrzały niż w Git,
- przeglądanie wersji w Subversion jest prostsze, ponieważ wykorzystuje kolejne numery wersji (1,2,3,..); Git używa hashy SHA-1. Cofanie wersji w Git jest łatwe dzięki możliwości użycia "^" w składni, ale nie jest łatwo iść do przodu.

git vs svn – plusy gita

- charakter rozproszony
- kontrola dostępu
- praca w branchach
- wydajność (szybkość wykonywania operacji)
- wymaganie mniejszej przestrzeni dyskowej
- automatyczna konwersja zakończenia linii

git vs svn – plusy svna

- dojrzałość interfejsu użytkownika,
- pojedyncze repozytorium,
- możliwość pobrania i pracy na części repozytorium,
- krótsze i przewidywalne numery wersji

gerrit

Gerrit to internetowy system przeglądu kodu, ułatwia osobom dokonującym przeglądu pracę z projektami wykorzystującymi system kontroli wersji Git.

Gerrit ułatwia wyświetlanie zmian w sposób side-by-side, pozwala na umieszczanie komentarzy w kodzie oraz ocenę zmiany przez sprawdzającego.

Gerrit umożliwia kontrolę rezultatów pracy w dużym zespole programistów, poprzez możliwość wymuszenia zatwierdzenia zmiany przez opiekuna danej funkcjonalności. Pozwala na bardziej scentralizowane wykorzystanie Gita.



gerrit

All | **My** | **Admin** | **Documentation**
[Changes](#) | [Drafts](#) | [Watched Changes](#) | [Starred Changes](#)

★ Change Ic8474648: **11-7359** | **11-7359**

Change-Id:	Ic8474648b874d44f7efdabb6ba9a748aeef34d 
Owner:	Pieszala, Maciej
Project:	geppo
Branch:	master
Topic:	
Uploaded:	May 9, 2011 6:17 PM
Updated:	May 12, 2011 1:47 PM
Status:	Merged


[Permalink](#) 

Reviewer	Verified	Code Review	Jira Check	Syntax Check	Review Check	
Pieszala, Maciej						
Gerrit			✓	✓	✓	Jira verified; Syntax verified; Review verified
Norris, Chuck	✓					Verified
[REDACTED]	✓	✓				Verified; Looks good to me, approved

► **Included in**

► **Dependencies**

▼ **Patch Set 1** [c8474648b874d44f7efdabb6ba9a748aeef34d](#) ([gitweb](#))

Author	Maciej Pieszala <maciej.pieszala@allegro.pl> May 9, 2011 6:17 PM
Committer	Maciej Pieszala <maciej.pieszala@allegro.pl> May 9, 2011 6:17 PM
Download	checkout pull cherry-pick patch SSH HTTP git fetch ssh://[REDACTED]:[REDACTED] geppo refs/changes/64/364/1 && git cherry-pick FETCH_HEAD 

[Review](#) | [Diff All Side-by-Side](#) | [Diff All Unified](#)

	File Path	Comments	Size	Diff	Reviewed
►	Commit Message			Side-by-Side Unified	
M	[REDACTED]		+1, -1	Side-by-Side Unified	✓
			+1, -1		

Comments [Expand Recent](#) | [Expand All](#) | [Collapse All](#)

Gerrit Patch Set 1: Syntax verified	May 9
Gerrit Patch Set 1: Jira verified	May 9
Norris, Chuck Patch Set 1: Build Started ...	May 9
Norris, Chuck Patch Set 1: Verified http://[REDACTED]:[REDACTED]@allegro-gerrit/397/	May 9

jenkins

Jenkins, znany wcześniej jako Hudson, jest oprogramowaniem open source ciągłej integracji (CI), narzędzie napisane w Javie. Projekt zmienił nazwę po sporze z Oracle, który rości sobie prawo do znaku towarowego i nazwy Hudson.

Jenkins zapewnia ciągłą integrację usług rozwoju oprogramowania, głównie w języku programowania Java. Jest to serwerowy system, który działa w kontenerze serwletów takim jak Apache Tomcat.

Integruje się z systemami kontroli wersji, w tym CVS, Subversion, Git i ClearCase. Może wykonywać skrypty związane z Apache Ant i Apache Maven, jak również dowolne skrypty powłoki i polecenia systemu Windows. Głównym twórcą Jenkins jest Kohsuke Kawaguchi . Opublikowane na Licencji MIT, Jenkins jest wolnym oprogramowaniem.

jenkins

Buildy można uruchomić za pomocą różnych środków, w tym wywołane przez commita w system kontroli wersji, planowania poprzez mechanizm typu cron, wyzwalanie gdy inne wersje zakończyły działanie oraz wywołując określony URL. Istnieje możliwość integracji np. z gerritem poprzez specjalne pluginy.

Około 2007 r. powstał projekt jako popularna alternatywa dla CruiseControl i innych open-source serwerów o podobnym charakterze. Na konferencji JavaOne w maju 2008 r. program został zwycięzcą Duke's Choice Award w kategorii Solutions Developer .

jenkins

Jenkins

Jenkins » [\[redacted\]](#) » #397

 [Back to Project](#)

 [Status](#)

 [Changes](#)

 [Console Output](#)

 [Configure](#)

 [Retriquer](#)

 [Parameters](#)

 [Git Build Data](#)

 [Test Result](#)

 [Code Coverage](#)

 [Previous Build](#)

 [Next Build](#)



Build #397 (May 9, 2011 6:17:49 PM)



Changes

1. [\[redacted\]](#) ([detail](#))



Triggered by Gerrit: [http://\[redacted\]:364](#)



Revision: c84746489b8f74d44f7efdabb6ba9a748aeef34d

•



[Test Result](#) (no failures)



Chuck Norris doesn't need an OS.

inne narzędzia

- AnthillPro - serwer ciągłej integracji Urbancode,
- Apache Continuum - serwer ciągłej integracji wsparcie Apache Maven i Apache Ant. Obsługa CVS, Subversion, Ant, Maven, i skryptów powłoki,
- Apache Gump - narzędzie ciągłej integracji Apache,
- Bamboo - serwer ciągłej integracji Atlassian Software Systems,
- Buildbota – oparty na Python / Twisted system ciągłej integracji,
- BuildMaster – narzędzie zarządzania cyklem życia aplikacji i ciągłej integracji firmy Inedo,
- CABIE – automatyczne cykliczne buildy oraz integracja środowisk - open source, napisany w języku Perl, współpracuje z CVS, Subversion, AccuRev, Bazaar i Perforce,
- Cascade - narzędzie ciągłej integracji, zapewnia możliwość tworzenia punktów kontrolnych do tworzenia i testowania zmian zanim zostaną one commitowane,
- CruiseControl – oparty na Java framework dla procesu ciągłego budowania,
- CruiseControl.NET – oparty na . NET automatyczny serwer ciągłej integracji,
- CruiseControl.rb - Lekki, oparty na Ruby serwer ciągłej integracji, który może budować kod, nie tylko Ruby, rozpowszechniony na licencji Apache 2.0,
- ElectricCommander – rozwiązanie dla procesu ciągłej integracji oraz zarządzania wdrożeniami firmy Electric Cloud,
- FinalBuilder Server - własne narzędzie do automatycznego budowania aplikacji oraz serwer ciągłej integracji firmy VSoft Technologies
- Go – narzędzie automatycznego budowania, integracji i wdrażania oparty na paradygmatach agile firmy Thoughtworks
- Software Configuration and Library Manager- oprogramowanie do zarządzania konfiguracją systemu z / OS firmy IBM Rational Software
- QuickBuild - serwer ciągłej integracji z darmową edycją, oferuje zarządzanie cyklem eksploatacji i weryfikacją pre-commit.
- Team Foundation Server - serwer ciągłej integracji i repozytorium kodu źródłowego firmy Microsoft
- Tinderbox - Mozilla produkt napisany w języku Perl

Bibliografia

<http://www.martinfowler.com/articles/continuousIntegration.html>

http://en.wikipedia.org/wiki/Continuous_integration

http://en.wikipedia.org/wiki/Comparison_of_Continuous_Integration_Software

<https://git.wiki.kernel.org/index.php/GitSvnComparison>

<http://jenkins-ci.org/>

<http://code.google.com/p/gerrit/>

Dziękuję za uwagę

pytania ???