Zarządzanie konfiguracją

(configuration management tools)

Wojciech Sołtysiak 228111

Agenda

1. Wstęp do Zarządzania Konfiguracją
2. Zarządzanie zmianą
3. Zarządzanie wersjami
4. Git Flow
5. Budowanie systemu
6. Zarządzanie wydaniami
7. Continuous Integration, Continuous Deployment
8. Zadania

Wstęp do Zarządzania Konfiguracją

Każde oprogramowanie podlega ciągłym zmianom. Jeśli spojrzymy na system jak na zbiór wersji, z których każda jest zarządzana i wspierana to głównym celem zarządzania konfiguracją jest odpowiedź na pytania:

- Jakie zmiany wprowadza konkretna wersja systemu?
- Jakie wersje komponentów są elementami konkretnej wersji systemu?

Zarządzanie konfiguracją jest ściśle związane z narzędziami pozwalającymi na zarządzanie zmianami wprowadzanymi do zmiennego systemu.

Zarządzanie zmianą

Jest to pierwszy etap zarządzania konfiguracją. To właśnie w tym miejscu śledzimy zgłoszenia potrzeby zmiany naszego systemu, określamy koszty i wpływ ewentualnej zmiany. Po tych krokach możemy zdecydować czy zmiana zostanie zaimplementowana, czy jednak odrzucona.

Zarządzanie zmianą

Przydatne oprogramowanie:

Nazwa	Cena	Szczegóły
Mantis	Darmowy, z możliwością wykupienia hostingu	Open-source, webowy interfejs, możliwość rozbudowy o dodatki, Mobile-Friendly
GitHub (funkcjonalność **issues**)	Darmowy dla publicznych repozytoriów, płatny dla prywatnych ($7-$21 na miesiąc)	Zespolony z repozytorium na GitHubie, prosty system zgłaszania i oznaczania zmian
IBM Rational ClearQuest	Uzgadniana z klientem, zależna od wielu czynników. Liczona raczej w tysiącach dolarów.	Profesjonalny “kombajn” z wieloma funkcjonalnościami, raportami i wykresami

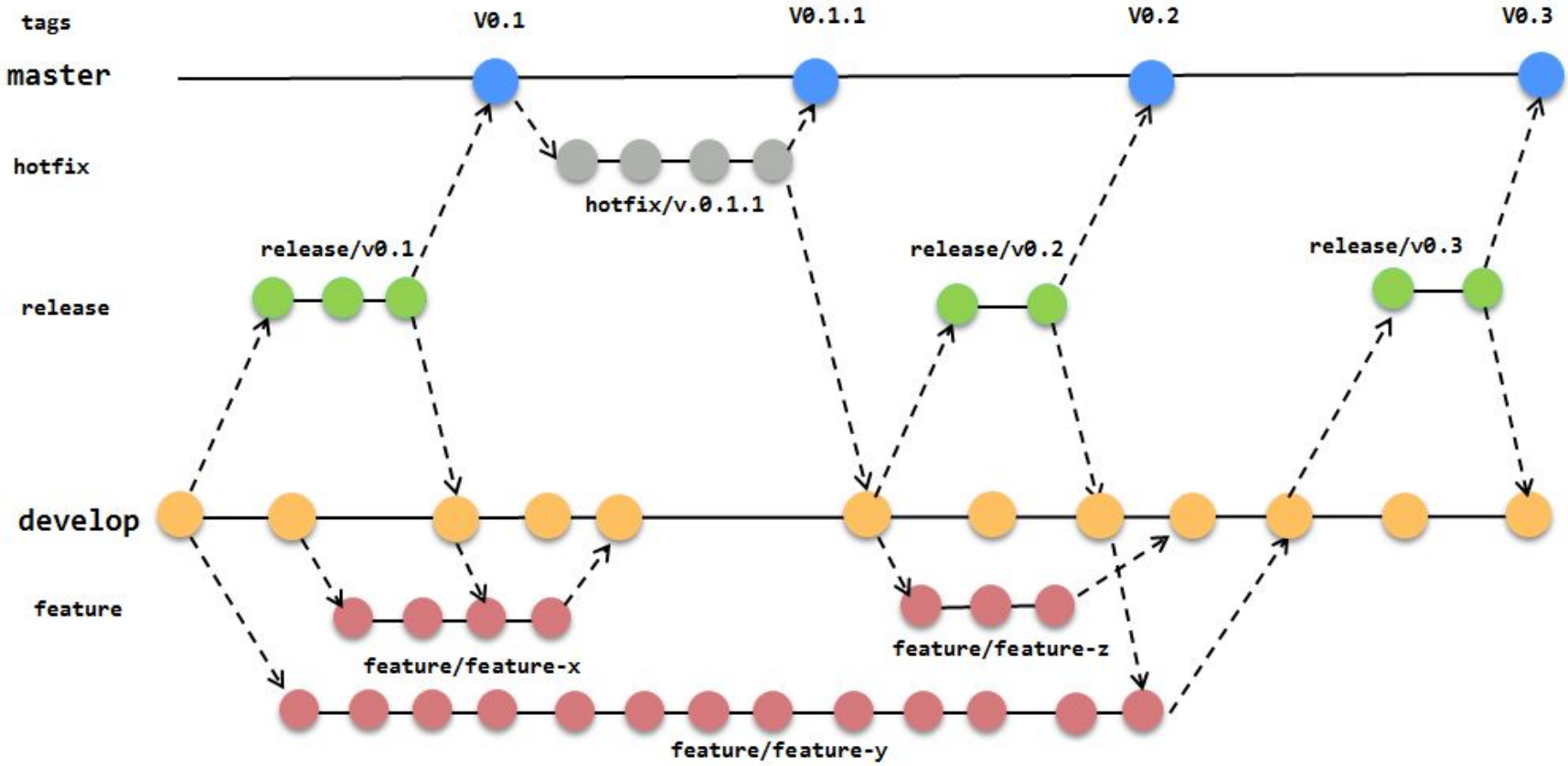
Zarządzanie wersjami

Drugi etap zarządzania konfiguracją. W tym miejscu zajmujemy się śledzeniem zmian w systemie i tworzeniem wersji. Jest to pomocne zarówno dla developerów jak i dla klientów. Klient wie, jaką aktualnie ma wersję systemu, a developer wie na której aktualnie pracuje. Narzędzia do zarządzania wersjami pozwalają na wygodną pracę programistów.

Zarządzanie wersjami

Przydatne oprogramowanie:

Nazwa	Cena	Szczegóły
Git	Darmowy	Rozproszony i aktualnie najbardziej popularny system kontroli wersji
SVN (Subversion)	Darmowy	Scentralizowany, wyparty przez Gita
Mercurial	Darmowy	Rozproszony, powstawał równolegle z Gitem, ma podobne założenia, jednak nie zdobył takiej popularności


tags
V0.1
V0.1.1
V0.2
V0.3
master
hotfix
hotfix/v.0.1.1
release/v0.1
release/v0.2
release/v0.3
release
develop
feature
feature/feature-x
feature/feature-z
feature/feature-y

Budowanie systemu

Trzeci etap zarządzania konfiguracją. Jest to proces łączenia komponentów programu, danych i bibliotek do postaci wynikowej. W zależności od wybranej technologii w której tworzony jest system mamy do wyboru wiele, zazwyczaj darmowych narzędzi.

Zarządzanie wydaniami, CD i CI

Ostatni etap, który musimy wykonać, żeby dostarczyć klientowi zaimplementowane zmiany. Przygotowujemy oprogramowanie do wydania na produkcję i śledzimy wersje, które dostarczamy klientom.

Continuous Integration - automatyczne uruchamianie kompilacji i testów (np. podczas wypushowania kodu albo o ustalonej godzinie) i ewentualne przygotowanie do wdrożenia.

Continuous Deployment - wgranie wydania na konkretny serwer (testowy, preprodukcyjny, produkcyjny) np. w momencie wmergowania gałęzi do mastera.

Zarządzanie wydaniami

Przydatne oprogramowanie:

Nazwa	Cena	Szczegóły
JIRA (funkcja **release**)	$10 na miesiąc do 10 użytkowników $7 na miesiąc za każdego użytkownika (11-100 użytkowników) Cena do negocjacji dla większych zespołów	Przygotowanie releasów na podstawie epiców, tasków i bugów w JIRA
GitHub (funkcja **releases**)	Darmowy dla publicznych repozytoriów, płatny dla prywatnych ($7-$21 na miesiąc)	Zespolony z repozytorium na GitHubie, prosty system tworzenia wydań
GitLab + GitLab Runner	Darmowy (wersja CE), werjse EE (na rok): $39, $199 + możliwość negocjacji w przypadkach większych potrzeb	Bogate możliwości przeprowadzania releasów, wsparcie dla CI i CD
Jenkins	Darmowy	Oprogramowanie oferujące CI i CD na serwery testowe, preprodukcyjne, produkcyjne itd.

Zadanie 1

1. Utwórz lokalnie repozytorium Git
2. Stwórz plik README.md, uzupełnij go i zacommituj
3. Załóż **puste**, publiczne (zdalne) repozytorium na platformie GitHub (https://www.github.com)
4. Połącz zdalne repozytorium z lokalnym
5. Wypushuj zmiany

Zadanie 2

1. Utwórz gałąź **develop** wychodzącą z gałęzi master (pamiętaj, aby ją wypushować)
2. W sekcji **Issues** dodaj zagadnienie “Dodanie hello world”
3. Utwórz gałąź featurową **hello-world** z brancha **develop**
4. Dodaj Hello World w dowolnym języku programowania, zacommituj i wypushuj
5. Załóż Pull Request brancha **hello-world** do brancha **develop** na platformie GitHub
6. Dodaj komentarz do swoich zmian (w Pull Request)
7. Zmerguj branch featurową do **develop**
8. Zamknij issue “Dodanie hello world”

Zadanie 3

1. Domegruj zmiany z **developa** do **mastera** (nie rób pull requesta!)
2. Utwórz nowy release z brancha **master** i oznacz go numerem 201801-1