

Narzędzia CI



O prowadzącej

Natalia Kniat

CI Lead / testerka / integratorka
w Nokia



certyfikowana testerka
ISTQB



trenerka Nokia Academy
(testowanie, telekomunikacja, Python)



zorganizowana mama
z zamiłowaniem do wyrobów własnych



Narzędzia CI



Przed CI



CI & CD
różnice



CI & CD
narzędzia



TravisCI
vs Jenkins

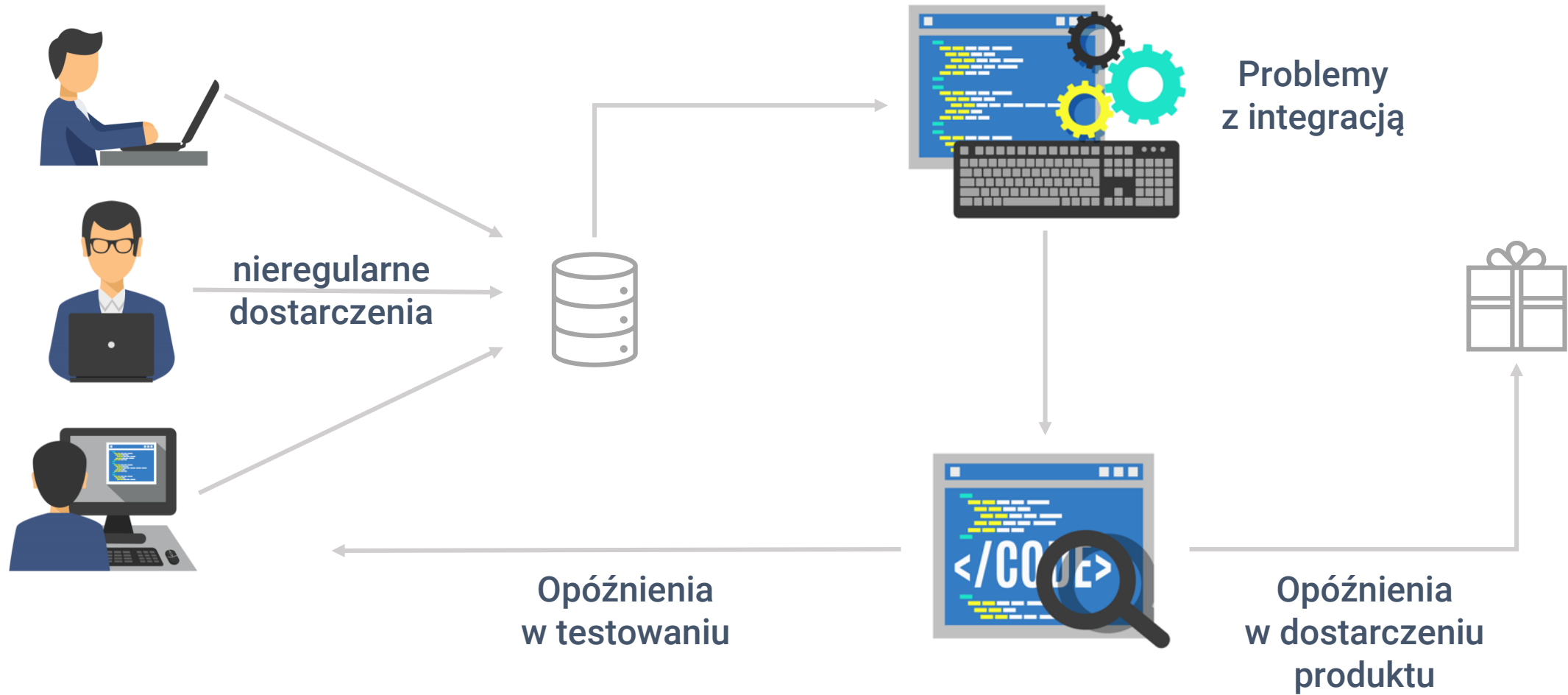


Jenkins
Konfiguracja



Pipeline

Przed CI



Przed CI



Problemy

- nieregularne commity
- developer czeka, aż całe oprogramowanie będzie dostarczone, aby przetestować i znaleźć błędy
- brak „iterative improvement”, wolne dostarczenia



Rezultat

- opóźnienia projektu
- obwinianie się nawzajem

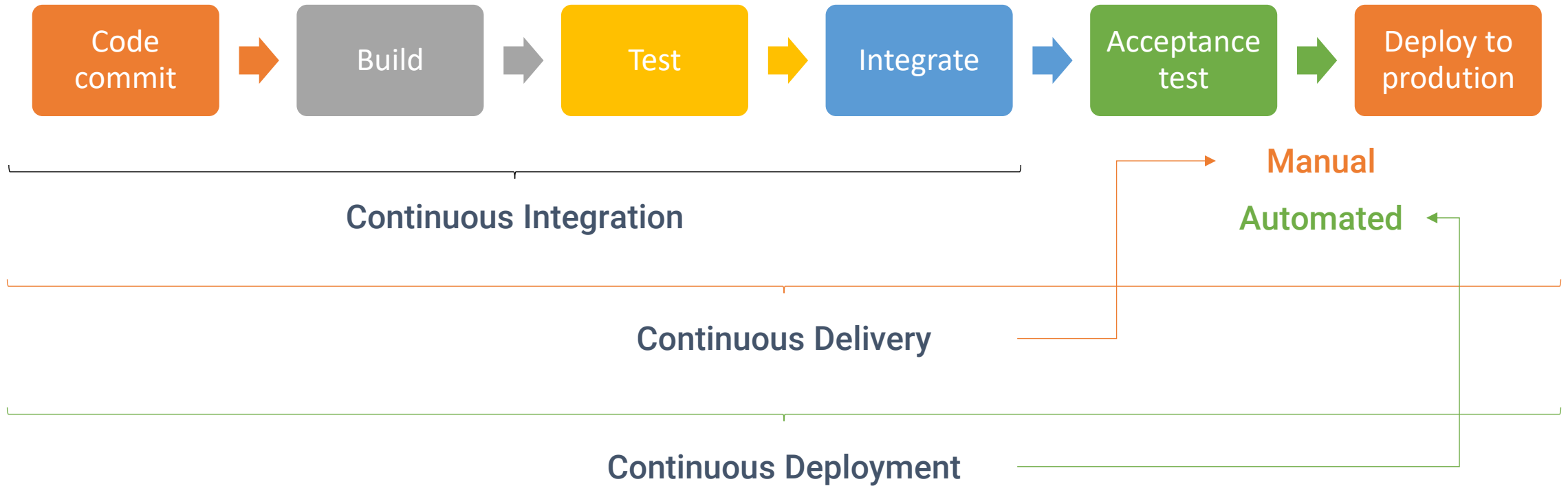


Wniosek

Jenkins wie, jak zaadresować wymienione problemy

CI vs CD vs CD

Continuous Integration vs Delivery vs Deployment



CI vs CD vs CD

Różnice

- **Podejście:**
 - CI - **automatyczne testowanie** każdej zmiany w bazie kodu
 - Continuous Delivery - **zmiany** nowych funkcji, konfiguracji i poprawki błędów
 - Continuous Deployment - tworzenia oprogramowania w krótkim cyklu.
- **Kiedy?**
 - CI jest wykonywane natychmiast **po komicie** programisty.
 - W Continuous Delivery opracowywany kod jest stale dostarczany do momentu, gdy programista uzna, że jest gotowy do wysyłki,
 - W Continuous Deployment programiści wdrażają kod bezpośrednio na etap produkcyjny, gdy jest on opracowywany.
- **Jakie testy?**
 - CI: **testy jednostkowe**,
 - Continuous Delivery: **testy logiki biznesowej**
 - Continuous Deployment: dowolna strategia testowania.
- **Fokus:**
 - CI odnosi się do wersjonowania kodu źródłowego,
 - Continuous Delivery odnosi się do logicznej ewolucji CI,
 - Continuous Deployment odnosi się do zautomatyzowanych implementacji kodu źródłowego.

Continuous Integration

Zalety & Wady

- Pomaga tworzyć oprogramowanie lepszej jakości
- Umożliwia przeprowadzanie powtarzalnych testów.
- CI umożliwia twórcom oprogramowania niezależną, równoległą pracę nad funkcjami.
- Może zwiększyć widoczność i umożliwić lepszą komunikację.
- Proces CI pomaga zwiększyć liczbę pracowników i wydajność dostaw zespołów inżynierskich.
- Ciągła integracja pomaga w opracowaniu produktu, który potencjalnie może zostać dostarczony po w pełni zautomatyzowanej kompilacji.
- Pomaga zmniejszyć ryzyko dzięki szybszemu i bardziej przewidywalnemu wdrażaniu
- Natychmiastowa informacja zwrotna po pojawieniu się problemu.
- Unika zamieszania w ostatniej chwili w dniu releasu,
- Zmniejsza ryzyko i sprawia, że proces wdrażania jest bardziej przewidywalny.
- CI zapewnia natychmiastową informację zwrotną w przypadku wystąpienia problemu.
- Możesz zobaczyć proces integracji w czasie rzeczywistym.
- Pozwala uniknąć kłopotów w ostatniej chwili w datach wydania.
- Aktualna wersja jest stale dostępna.
- Regularnie dostarcza produkty nadające się do wysyłki.
- Stosunkowo łatwo jest znaleźć historię kompilacji oprogramowania.
- CI zapewnia stabilność kodu.

Continuous Integration

Zalety & Wady

- Wstępna konfiguracja i szkolenie jest wymagane do zapoznania się z serwerem CI
- Dobrze zaprogramowana suita-testowa wymaga wielu zasobów dla serwera CI.
- Wymaga dodatkowych serwerów i środowisk.
- Potrzebujesz konwersji znanych procesów w jednym projekcie.
- To oznacza czekanie, gdy wielu programistów integruje swój kod w tym samym czasie.
- Twój zespół powinien pisać automatyczne testy dla każdej nowej funkcji lub poprawki błędu.
- Potrzebujesz serwera CI, który monitoruje główne repozytorium i uruchamia testy nowych komitów kodu.
- Deweloperzy powinni jak najczęściej scalać swoje zmiany.
- Procedura unit testów powinna przejść do deploymentu.

Continuous Delivery

Zalety & Wady

- Automatyzacja procesu wydawania oprogramowania, aby dostarczanie było wydajniejsze, szybsze i bezpieczniejsze.
- Praktyki związane z CD zwiększają produktywność, uwalniając programistów od pracy ręcznej i złożonych zależności.
- Pomaga wykryć błędy oprogramowania na wczesnym etapie procesu dostarczania.
- CD pomaga zespołowi biznesowemu w natychmiastowym i częstym dostarczaniu aktualizacji klientom.
- Zapewnia, że oprogramowanie jest zawsze gotowe do uruchomienia produkcji.
- Możesz częściej wydawać oprogramowanie, co pozwala szybko uzyskać informacje zwrotne od klientów.
- Mniejsza jest presja na decyzje dotyczące małych zmian.

Continuous Delivery

Zalety & Wady

- Powinieneś znać praktyki ciągłej integracji, zanim zdecydujesz się na ciągłe dostarczanie.
- Wdrożenia są nadal ręczne, dlatego dostarczenie oprogramowania zajmuje dużo czasu.
- Testy automatyczne powinny być napisane i działać prawidłowo.
- Błędne testy mogą prowadzić do uszkodzeń podczas testowania jakości.
- Wymaga koordynacji zespołowej, ponieważ zmiany w kodzie powinny być regularnie gromadzone w efektywny sposób.
- Ciągłe dostarczanie wymaga niezawodnego i silnego serwera integracyjnego na potrzeby kosztownych testów automatyzacji.

Continuous Deployment

Zalety & Wady

- Pomaga zautomatyzować powtarzające się zadania.
- CD sprawia, że wdrożenie przebiega bezbłędnie bez narażania bezpieczeństwa.
- Łatwe skalowanie od pojedynczej aplikacji do portfolio IT przedsiębiorstwa.
- Możesz dostarczać aplikacje natywne dla chmury oraz aplikacje tradycyjne.
- Daje pojedynczy widok we wszystkich środowiskach i aplikacjach.
- Możesz połączyć istniejące narzędzia i skrypty DevOps w odpowiedni przepływ pracy.
- CD umożliwia zwiększenie ogólnej wydajności.
- Możesz integrować procesy i zespoły za pomocą ujednoliconego potoku.

Continuous Deployment

Zalety & Wady

- Twoja kultura testowania powinna być dobra, ponieważ jakość suity decyduje o tym, jak dobre są wydania oprogramowania.
- Dokumentacja musi nadążać za tempem wdrażania.
- Wprowadzanie znaczących zmian wymaga zatwierdzenia przez działy marketingu, pomocy i wsparcia oraz inne działy.

CI vs CD vs CD

Wyzwania

Continuous Integration

- Spowalnia proces tworzenia.
- Eksponuje problemy i dzielenie się problemami.
- Może prowadzić do braku utrzymania kontroli wersji.
- Może zmusić cię do radzenia sobie z problemami.
- Trudności w budowaniu automatycznego repozytorium kodu.
- Nieprzetestowany lub popsuty kod nie może być dostarczony

Continuous Delivery

- Musisz utrzymać CD wydajną
- Musisz radzić sobie z napiętymi terminami planu wydania.
- Słaba komunikacja zespołów dotycząca konkretnego produktu może prowadzić do zmian, a także opóźnień we wdrażaniu.
- Menagement powinien dysponować budżetem na infrastrukturę niezbędną do tworzenia bardziej imponującego oprogramowania.
- Dane/informacje z monitoringu powinny być wykorzystywane przez zespół badawczo-rozwojowy.
- Organizacja powinna upewnić się, że oprogramowanie typu open source pasuje do bieżącego sposobu pracy.

Continuous Deployment

- CD wymaga ciągłego planowania, aby osiągnąć częste i szybkie wydania.
- Zapewnienie zgodności między wymaganiami kontekstu biznesowego a rozwojem aplikacji.
- Szybka dostawa nie może być odizolowana wyłącznie od procesu tworzenia oprogramowania.
- Przepływ powinien być zgodny z całym cyklem tworzenia oprogramowania.
- Eksperymentalne działania muszą być stale powiązane z planem działania oprogramowania.

Jenkins

Narzędzie do ciągłej integracji,
pozwalające na ciągły rozwój, testowanie i wdrażanie nowo wytworzonego kodu



Dwa sposoby, aby zapewnić continuous development:

1. Nocne buildy
2. Continuous integration

Zmiana standardowego podejścia do wytwarzania oprogramowania.

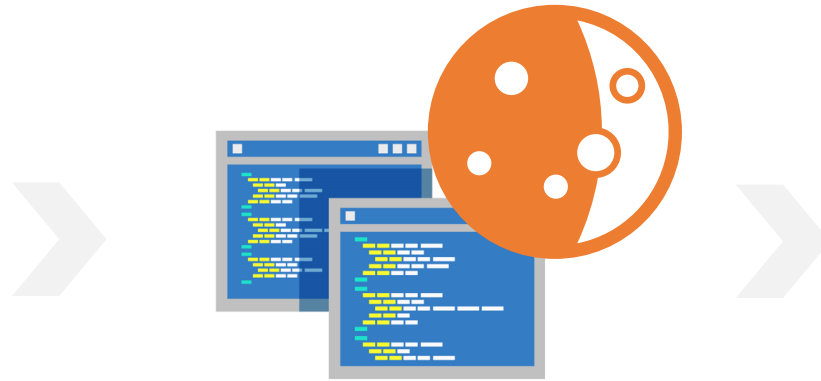
Jenkins

Nocne buildy (nightly builds)



Commit

Dostarczenie zmian do kodu źródłowego



Pull

Cały kod zostanie zaciągnięty w nocy



Build

Wszystkie zmiany w kodzie wchodzą razem i zostaną połączone w całość

Jenkins

Ciągłe budowanie (Countinous build)



Commit

Dostarczenie zmian do głównego programu



Pull

- Włączenie testów i weryfikacji środowiska
- Testowanie przed dostarczeniem

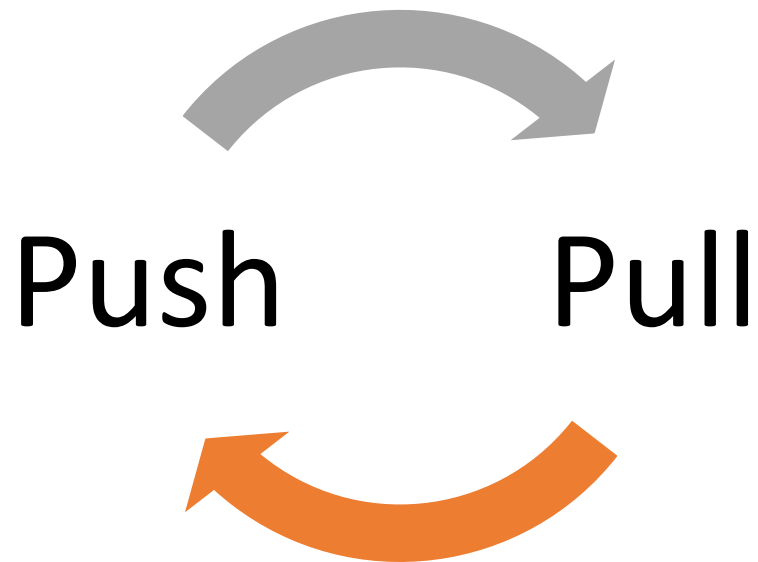


Build

Wszystkie zmiany w kodzie są ciągle przebudowywane

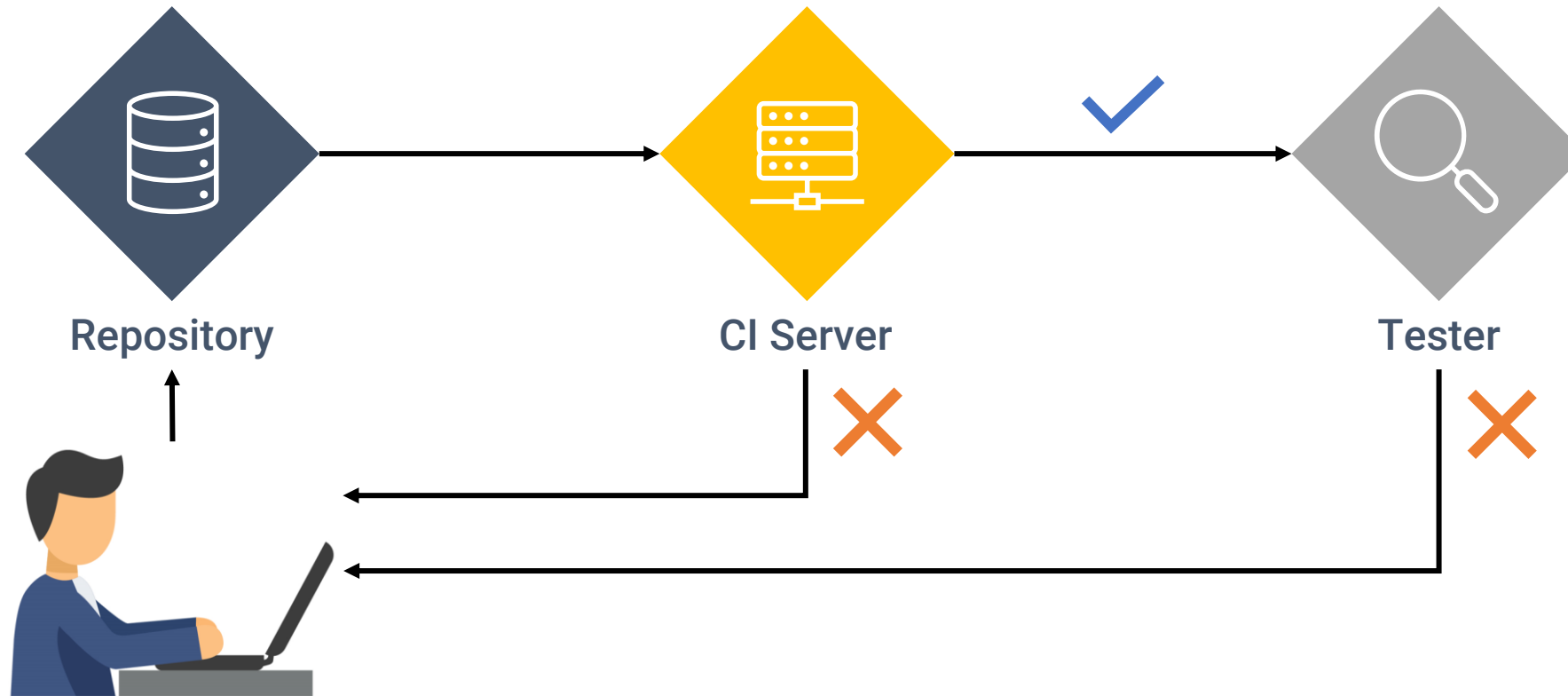
Jenkins

Pozwala na pobranie repozytorium (pull) w dowolnym momencie
oraz dostarczyć zmiany (push) w dowolnym momencie



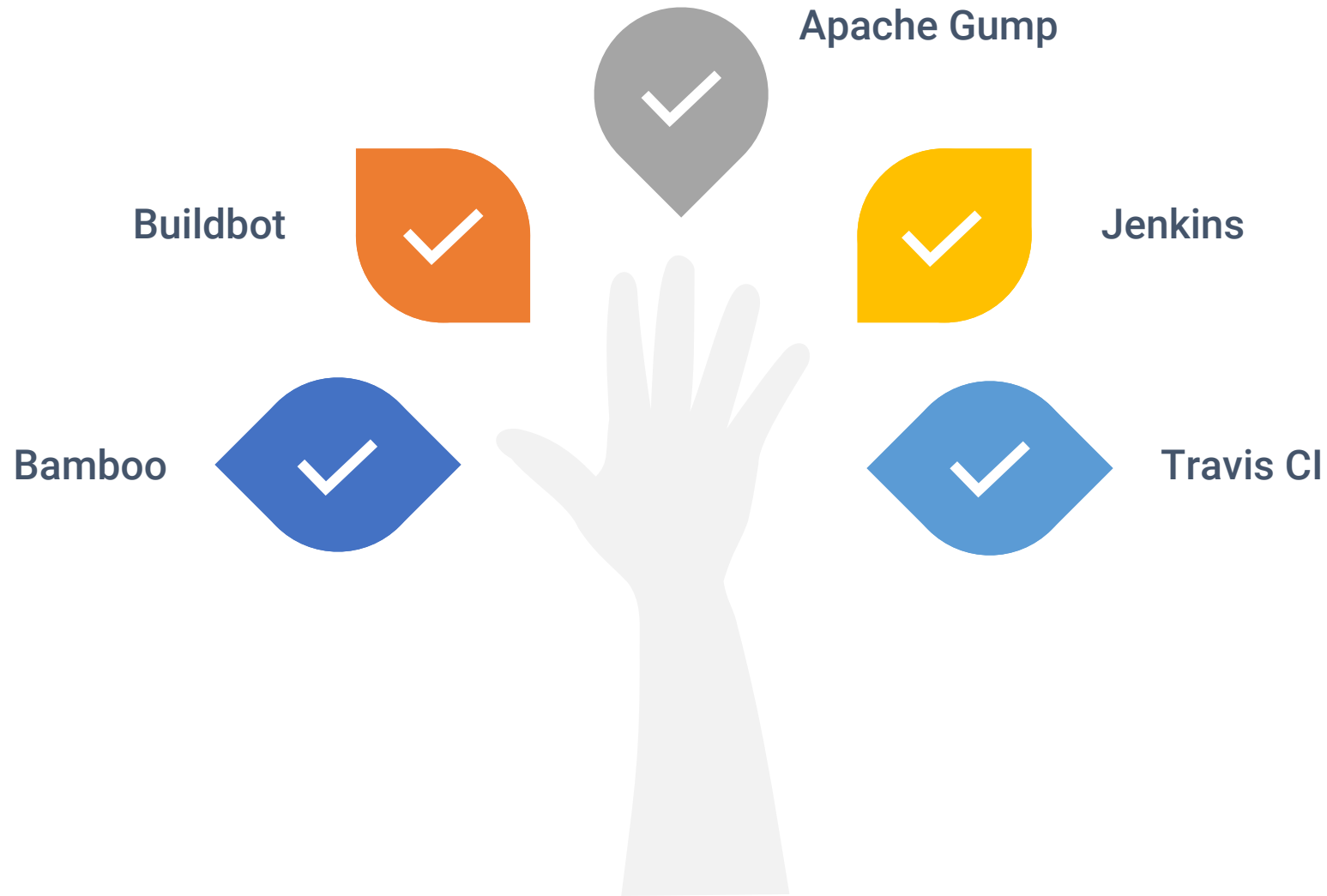
Continuous integration

„Continuous Integration Server” odpowiada za validację i testy nowego kodu.
Gdy test nie przejdzie – commit wraca do developera. Developer może uruchomić testy u siebie.
Dzięki temu można szybciej dostarczać.



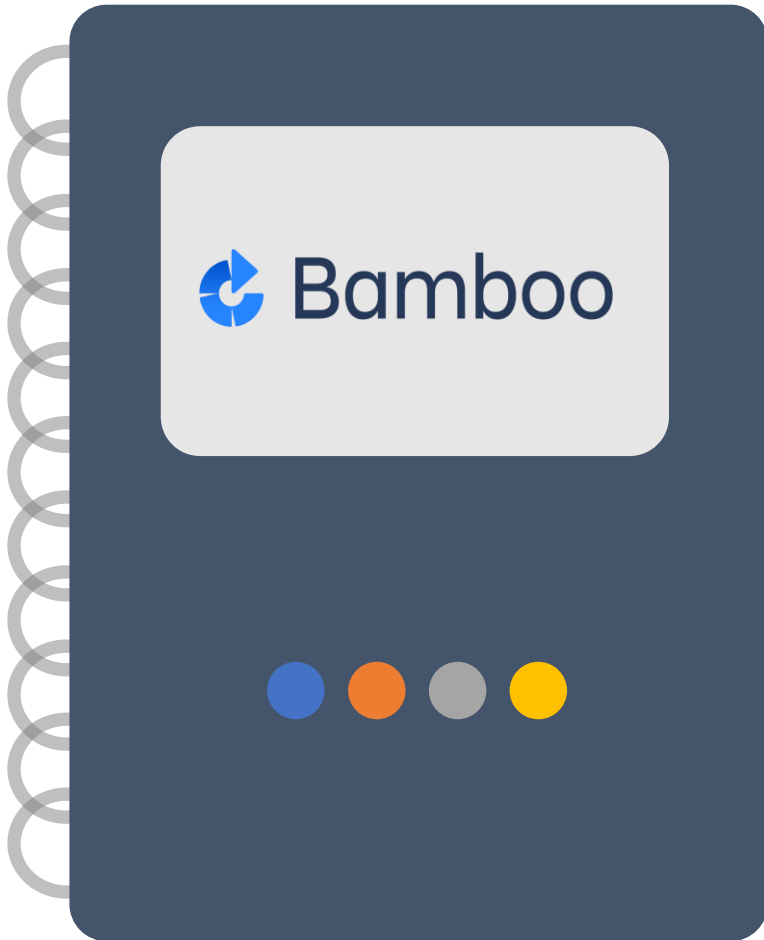
Continuous integration

Narzędzia Open Source oraz płatne



Continuous integration

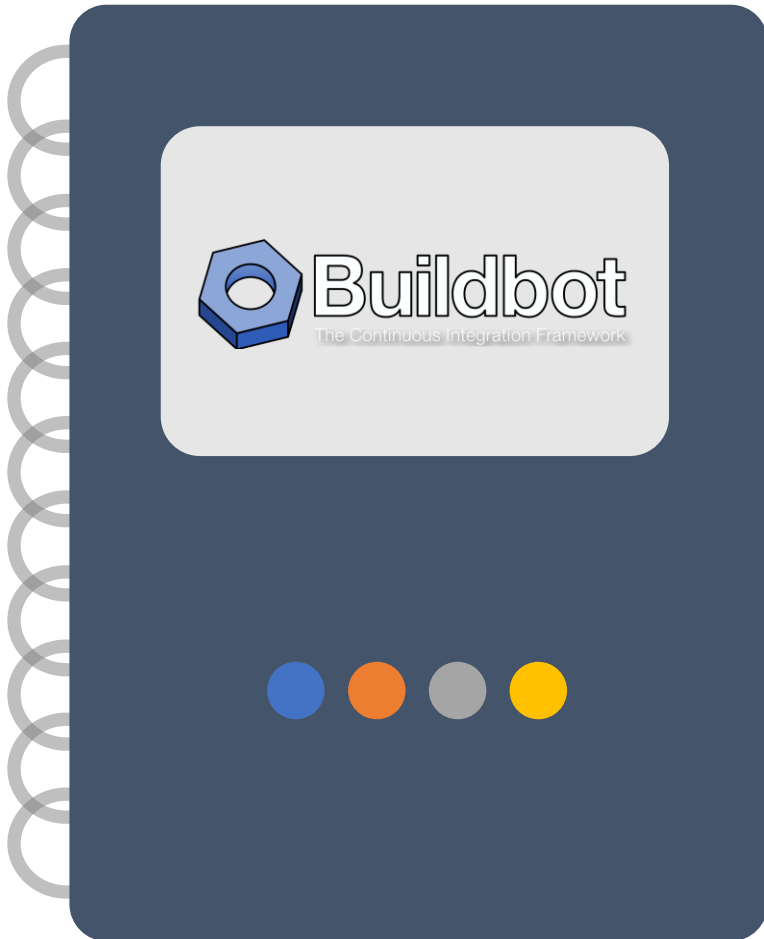
Narzędzia płatne



- pozwala na uruchomienie wielu buildów równolegle dla szybszej kompilacji
- gdy mamy kilka wersji naszego oprogramowania możemy uruchomić kompilację równolegle. Dzięki temu można szybko przetestować, jak działają różne wersje kodu.
- wbudowane funkcje odzyskiwania awaryjnego
- zintegrowane z Bitbucket (Git) i Jira - ten sam dostawca
- <https://www.atlassian.com/pl/software/bamboo>

Continuous integration

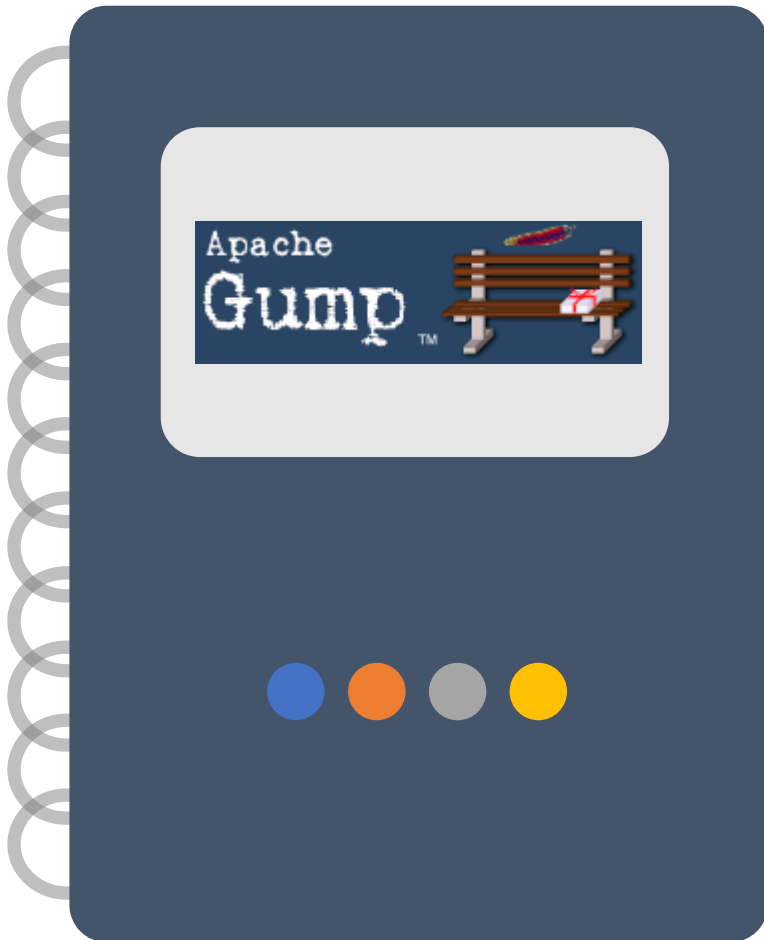
Narzędzia Open Source



- automatyzacja wytwarzania oprogramowania, testowania i wydawania,
- wspiera działanie rozproszone na różnych platformach
- napisany w Python
- w przeciwieństwie do Jenkinsa nie ma struktury gotowych do użycia aplikacji, przez co dokładniej można go dostosować do własnych potrzeb.
- <https://buildbot.net/>

Continuous integration

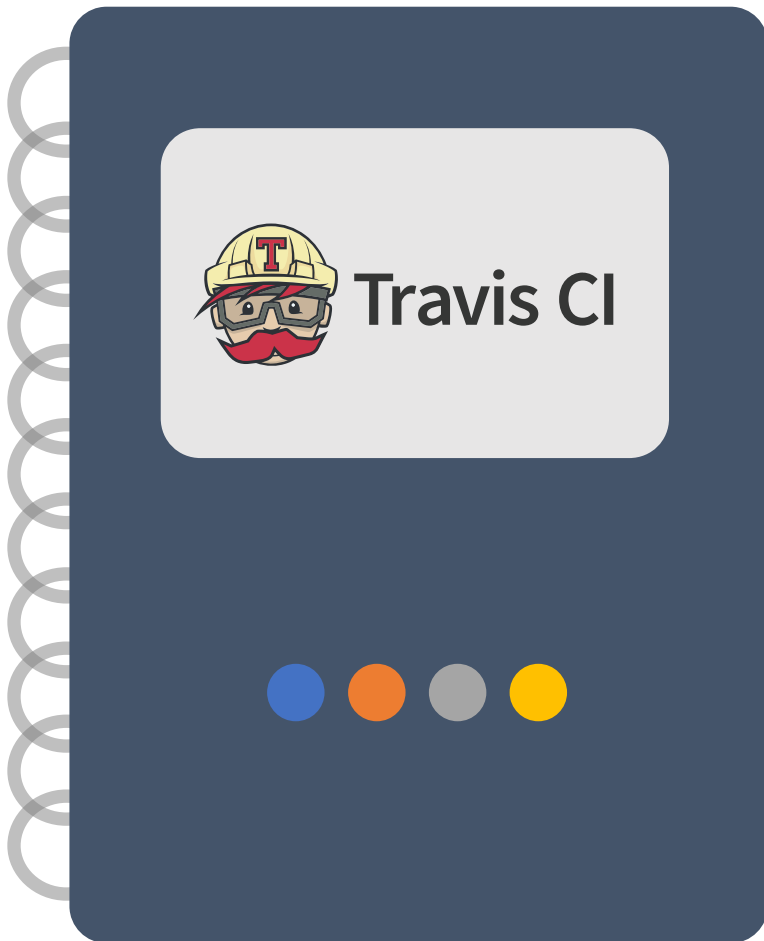
Narzędzia Open Source



- dla projektów napisanych w Java
- napisany w Python
- <https://gump.apache.org/>

Continuous integration

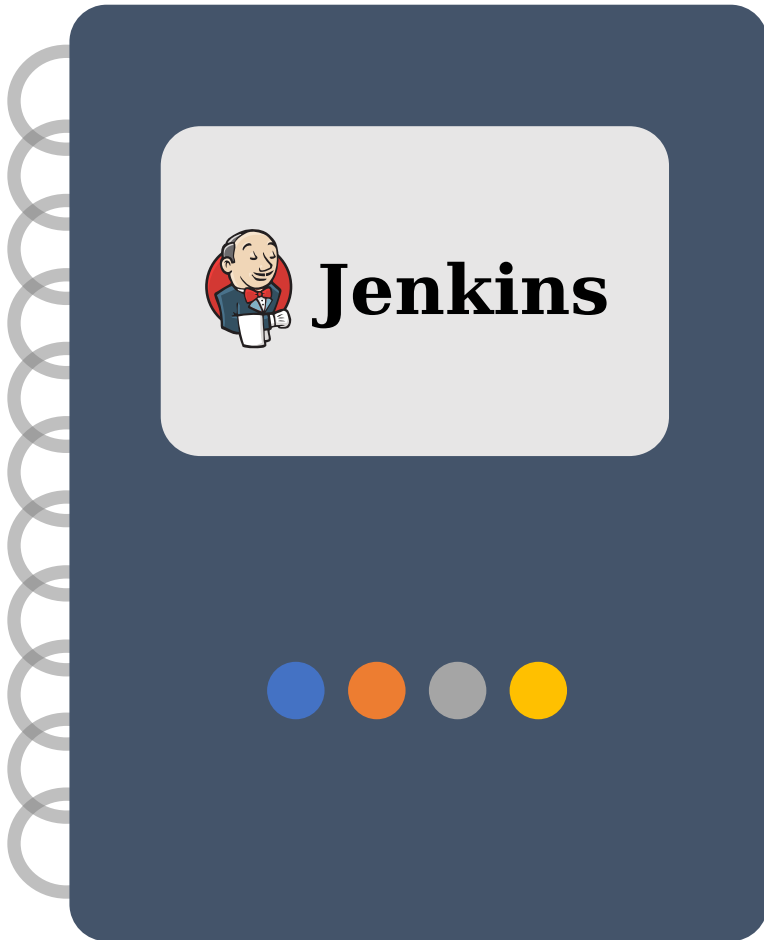
Narzędzia płatne



- narzędzie dla projektów w Github
- wspiera 20 języków
- automatyzację procesu wdrażania nowego kodu na serwery
- <https://travis-ci.org/>

Continuous integration

Narzędzia Open Source



- napisany w Javie
- zcentralizowane narzędzie do wszystkich projektów
- może być zainstalowany za pomocą natywnych paczek systemowych, kontenerów (Docker) lub na dowolnym komputerze z zainstalowanym środowiskiem Java
- <https://www.jenkins.io/>

Jenkins

Jakie posiada funkcje?

Łatwa konfiguracja

Prosta konfiguracja z interfejsu webowego, który sprawdza błędy i posiada wbudowaną pomoc

Łatwa instalacja

Samowystarczalny program oparty na Java gotowy do uruchomienia z Windows, Mac OS X i Unix OS

Wtyczki

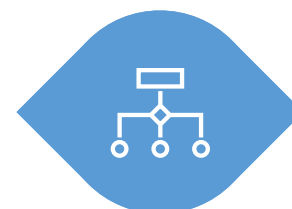
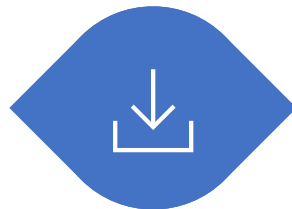
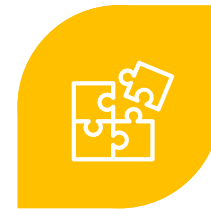
Posiada setki wtyczek dostępnych w Update Center, dzięki czemu integruje się z każdym innym narzędziem CI/CD

Rozszerzalność

Rozszerzany za pomocą wtyczek, zapewni nieskończenie wiele możliwości

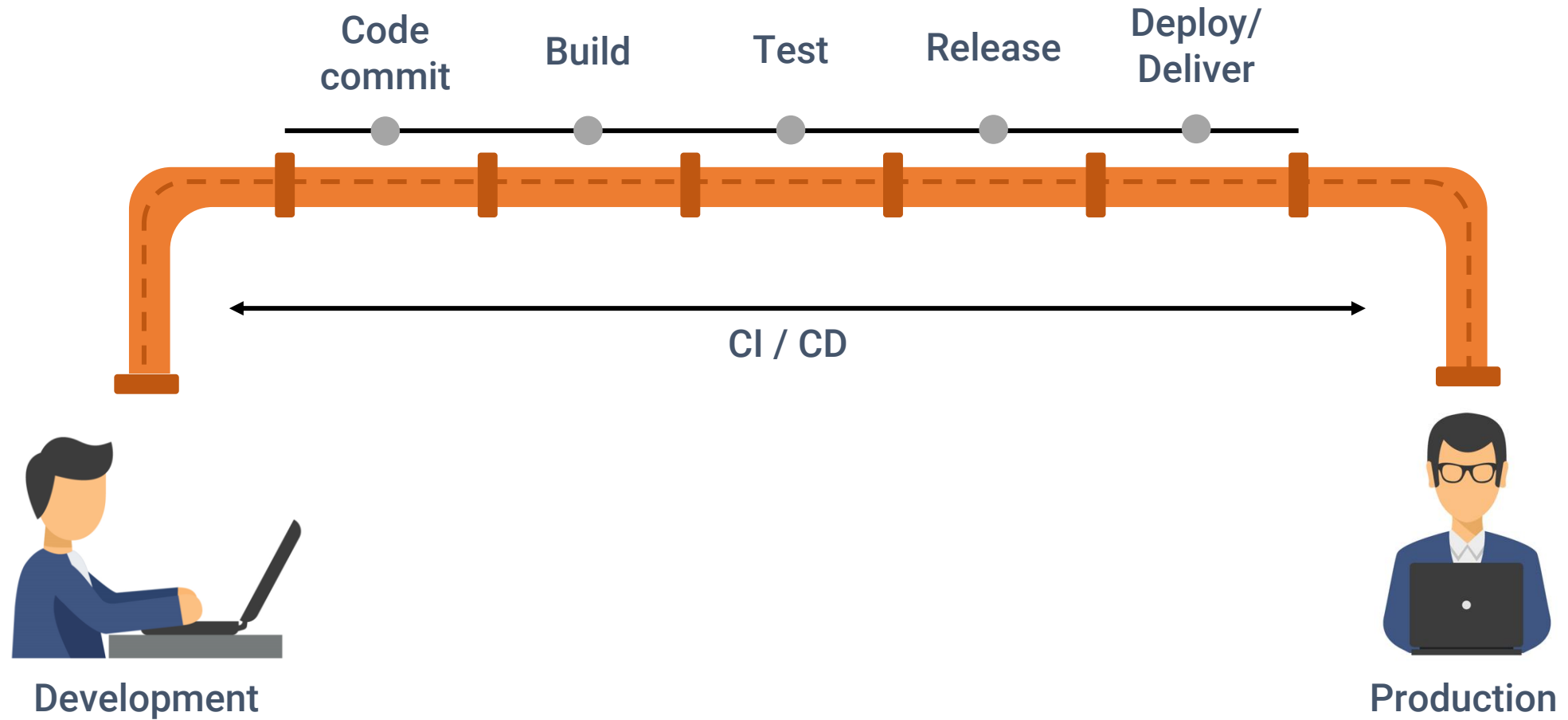
Dystrybucja

W prosty sposób rozdziela pracę na wiele maszyn, pomagając w szybszej kompilacji, testowaniu i dostarczaniu na różne platformy



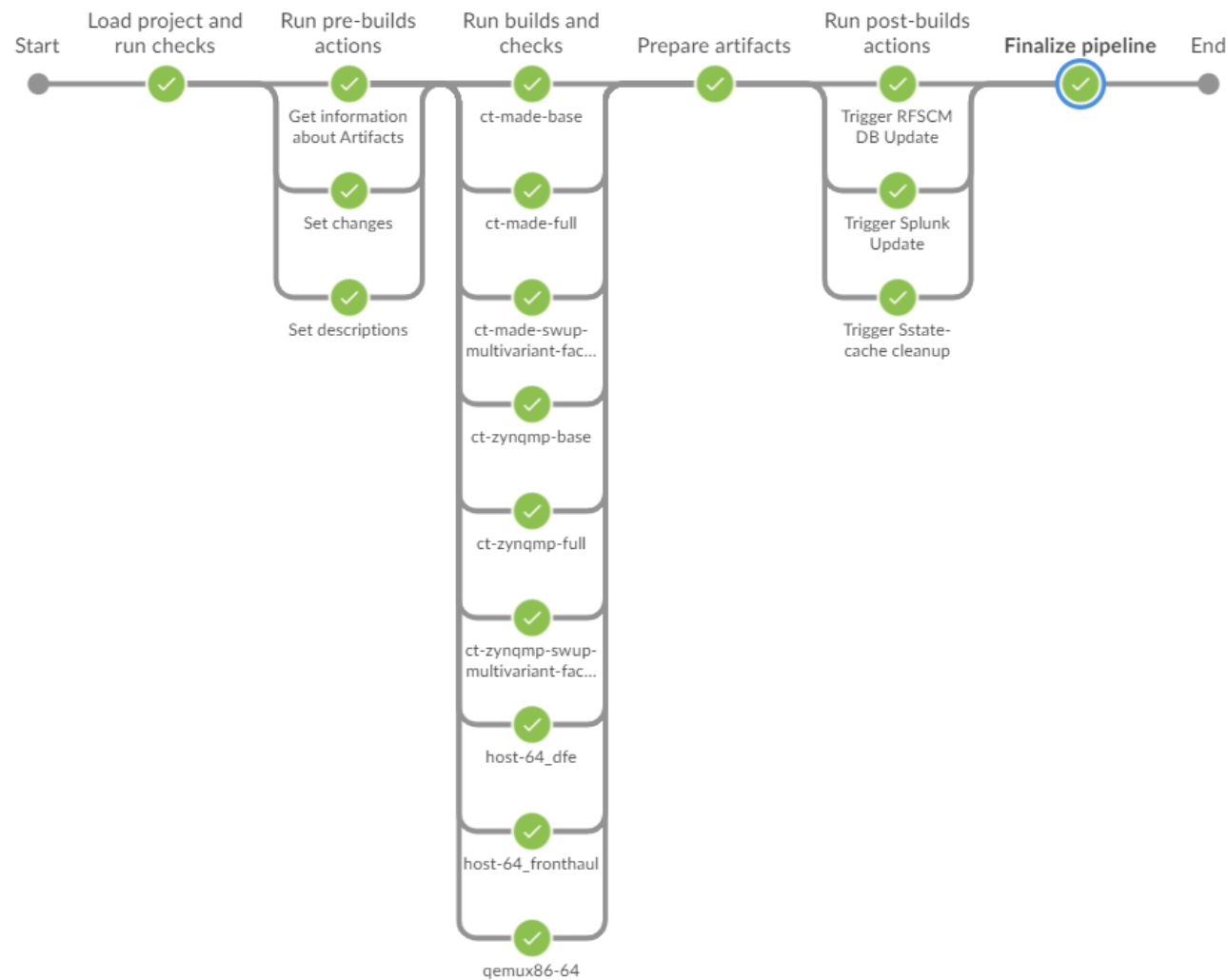
Jenkins

Pipeline



Jenkins





Przykład (Nokia)



Jenkins

Jaka dzisiaj pogoda?

Production jobs

S	W	Name ↓	Last Success	Last Failure	Last Duration
		PROD_CTA6_release	1 hr 16 min - #5809	10 days - #5555	2 min 51 sec
		PROD_CTA6_release_supervisor	1 hr 13 min - #5777	13 hr - #5755	22 min
		PROD_CTA6_worker	1 hr 12 min - #117864	13 hr - #117590	17 min

Branches

S	W	Name ↓	Last Success	Last Failure	Last Duration
		CI_Megatron_PROD_CTA6_22R2_release_AQQL_Acceptance_Tests_rexIO	1 day 12 hr - #16	4 days 13 hr - #15	3 hr 42 min 
		CI_Megatron_PROD_CTA6_22R2_release_AQQL_Acceptance_Tests_standalone	1 day 11 hr - #12	1 day 12 hr - #11	1 hr 1 min 
		CI_Megatron_PROD_CTA6_22R2P7_release_AQQL_Acceptance_Tests_rexIO	N/A	N/A	N/A 

Jenkins

Jaka dzisiaj pogoda?

 	CHECK RFSW INTERNAL draco-common_rel	21 days - #11	13 days - #16	1 hr 2 min	-
 	CHECK RFSW INTERNAL draco-configs	1 hr 10 min - #6179	12 hr - #6165	47 min	-
 	CHECK RFSW INTERNAL draco-configs_rel	1 day 10 hr - #112	5 days 7 hr - #110	59 min	-
 	CHECK RFSW INTERNAL draco-features-made	2 mo 19 days - #2	N/A	2 min 44 sec	-
 	CHECK RFSW INTERNAL draco-features-made_rel	N/A	N/A	N/A	-
 	CHECK RFSW INTERNAL draco-features-nahka	8 days 12 hr - #232	N/A	31 min	-
 	CHECK RFSW INTERNAL draco-features-nahka_rel	N/A	N/A	N/A	-
 	CHECK RFSW INTERNAL draco-ifc	10 hr - #2215	2 hr 18 min - #2223	52 min	-
 	CHECK RFSW INTERNAL draco-ifc_rel	8 days 11 hr - #31	11 days - #29	1 hr 27 min	-
 	CHECK RFSW INTERNAL draco-platform-asmr	1 yr 2 mo - #1	N/A	46 sec	-
 	CHECK RFSW INTERNAL draco-platform-asmr_rel	N/A	N/A	N/A	-
 	CHECK RFSW INTERNAL draco-platform-chile	2 days 7 hr - #1071	1 day 10 hr - #1075	45 min	-
 	CHECK RFSW INTERNAL draco-platform-chile_rel	6 days 0 hr - #21	6 days 5 hr - #20	29 min	-

Jenkins

Case study



Oszczędność

Problem:

- Systemy automotive stają się coraz bardziej wyspecjalizowane i złożone
- Wiele funkcji pojazdów to rozwiązania softwarowe.

Firma BOSCH znalazła potrzebę pomocy swoim software engineerom w szybszym produkowaniu i dostarczaniu dobrej jakości oprogramowania.

Wyzwanie:

- zarządzanie rozwojem coraz bardziej skomplikowanego oprogramowania z pomocą CI i CD, aby skrócić proces produkcji i dostarczeń.

Użyto platformy CloudBees Jenkins. Pozwoliło to zredukować liczbę kroków wykonywanych manualnie oraz wykryć i wyeliminować duplikującą się pracę w procesie budowania, wdrażania i testowania.

Rezultat:

- Proces budowania paczki zredukowano z 3 dni do 3 godzin!

TravisCI vs Jenkins

Co to jest?



TravisCI vs Jenkins

Różnice

Co robi Travis?

- Możesz monitorować projekty GitHub
- Uruchamia test i szybko generuje wyniki. Możliwe jest równoległe wykonanie testu.
- Tworzy artefakty i sprawdza jakość kodu
- Łatwe wdrażanie do usług w chmurze
- Może identyfikować zarówno małe, jak i duże zmiany w kodzie.
- Deweloperzy mogą używać Travis CI do oglądania testów, gdy są uruchomione.
- Narzędzie integruje się ze Slack, HipChat, Email itp.

Co robi Jenkins?

- Pozwala zautomatyzować tworzenie, testowanie i wdrażanie zadań. Narzędzie zapewnia obsługę różnych systemów operacyjnych, takich jak Windows, Mac OSX i Linux.
- Umożliwia szybkie budowanie i testowanie kodu, aby uzyskać wczesną informację zwrotną, czy jest gotowy do produkcji, czy nie. W większości przypadków Jenkins będzie wymagał kilku modyfikacji zgodnie z niestandardowymi wymaganiami Twojego zespołu.

TravisCI vs Jenkins

Różnice

Funkcje Trávisa:

- Automatyczna integracja z GitHub
- Dostęp do repozytorium w celu kompilacji pull requestów
- Obsługa 21 języków, takich jak Android, C, C#, C++, Java, JavaScript (z Node.js), Perl, PHP, Python, R, Ruby itp.
- Wstępnie zainstalowane narzędzia do budowania i testowania
- Dostępne usługi – bazy danych, kolejki wiadomości itp.
- Wdrożenie do wielu usług w chmurze
- Szyfrowanie, bezpieczne zmienne środowiskowe lub pliki
- Maszyny wirtualne odtwarzane po każdej kompilacji
- Klient CLI i API do obsługi skryptów
- Zawiera darmowy hosting w chmurze, który nie wymaga konserwacji ani administracji.

Funkcje Jenkinsa:

- Łatwa instalacja, aktualizacja i konfiguracja
- Rozproszone kompilacje
- Monitorowanie zewnętrznych jobów
- Ponad 600 wtyczek do dostosowywania środowiska Jenkins
- Ponad 1000 publicznych repozytoriów na Github, ponad 500 współtwórców, silna aktywność związana z zatwierdzaniem
- Wsparcie dla różnych metod uwierzytelniania, systemów kontroli wersji, powiadomień itp.
- Jenkins zapewnia zdalny dostęp do API i jego funkcjonalności.
- Zapewnij potężne narzędzie CI/CD dla dużych projektów
- Obsługuje różne modele pracy, takie jak Freestyle, Pipeline itp.,
- Umożliwia programistom dodawanie rozszerzeń
- Kompatybilny z Docker, Libvirt, Kubernetes i wieloma innymi programami

TravisCI vs Jenkins

Porównanie

Parametr	Jenkins	TravisCI
Koszt	Darmowy + koszt serwera dedykowanego	Płatny dla przedsiębiorstw
Czas ustawienia	Skomplikowana konfiguracja = dużo czasu	Jeden plik konfiguracyjny = mało czasu
Wydajność	Nieograniczone możliwości dostosowania	Dobry dla projektów open source
Użycie	Łatwe	Elastyczne
Github	Dobry	Doskonały
Wsparcie	Szerokie wsparcie społeczności	Ograniczone wsparcie społeczności
Serwer	Oparty na serwerze	Oparty na chmurze

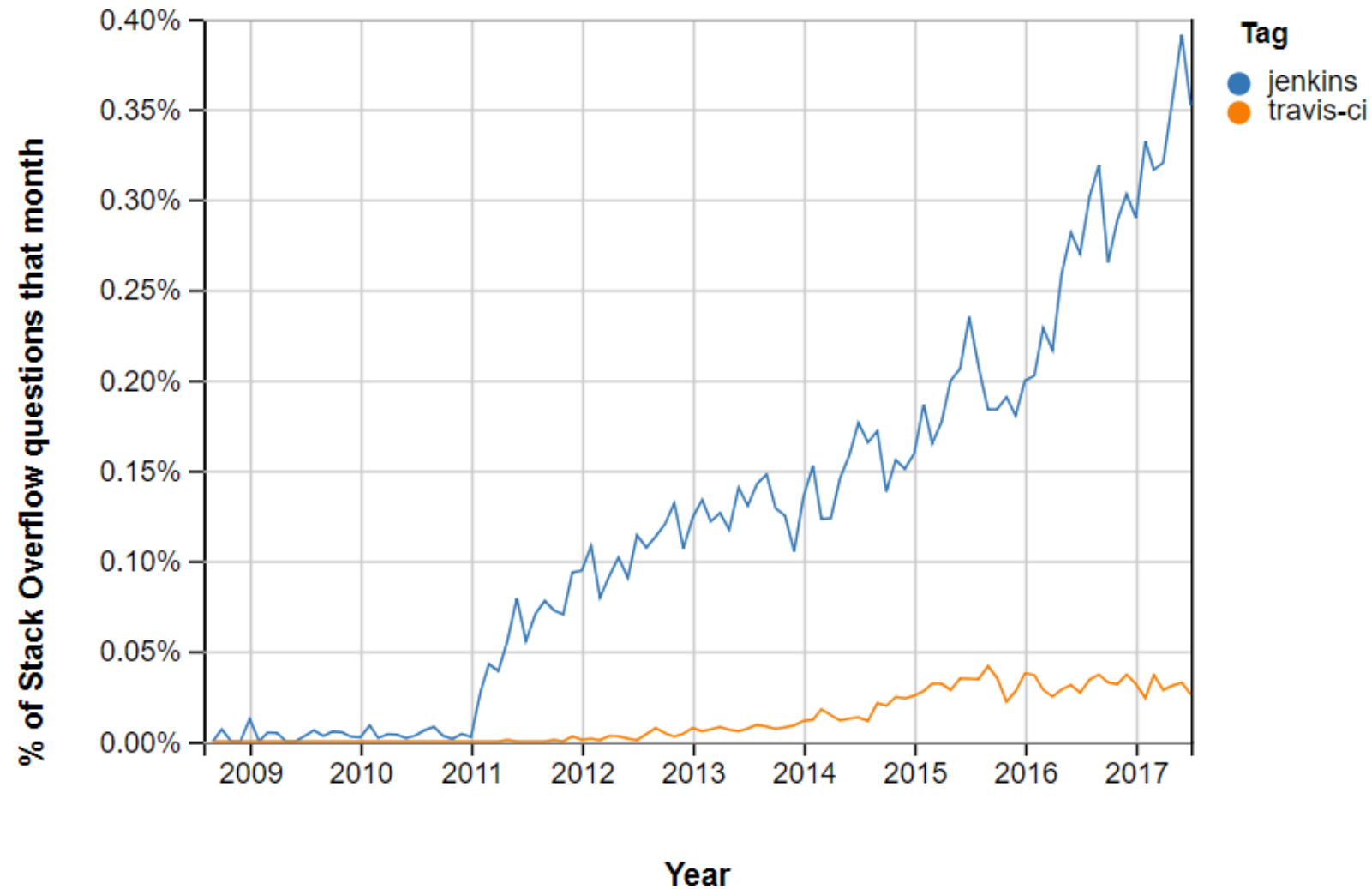
TravisCI vs Jenkins

Porównanie

	Jenkins	TravisCI
Plusy	Personalizacja	Integracja z GitHub i chmurą
	Rozbudowane archiwum wtyczek	Nieograniczone projekty open source z pełną funkcjonalnością
	Możesz także dodać nowe funkcje, takie jak uwierzytelnianie, alerty i poświadczenia.	Rozbudowana konfiguracja projektu za pomocą pliku .travis.yml
		Umożliwia testy klastrowe i uruchamianie ich równolegle
Minusy	Wysoce konfigurowalny = brak gotowej konfiguracji	Płatny
		Nie nadaje się do projektów o wysokim poziomie bezpieczeństwa

TravisCI vs Jenkins

Popularność



Jenkins

Narzędzie do ciągłej integracji,
pozwalające na ciągły rozwój, testowanie i wdrażanie nowo wytworzonego kodu



1. Wejdź na <https://vlabs.wsb.wroclaw.pl/portal>
2. Uruchom maszynę 2. Programowanie
3. Uruchom plik „Jenkins”, znajdujący się na Pulpicie

```
C:\Users\vdi-belfer\Desktop>java -jar C:\Users\Public\Documents\jenkins.war  
Running from: C:\Users\Public\Documents\jenkins.war  
webroot: $user.home/.jenkins
```

4. Poczekać, aż konsola zakomunikuje poprawny start

```
INFO jenkins.InitReactorRunner$1#onAttained: Prepared all plugins  
INFO jenkins.InitReactorRunner$1#onAttained: Started all plugins  
INFO jenkins.InitReactorRunner$1#onAttained: Augmented all extensions  
INFO jenkins.InitReactorRunner$1#onAttained: System config loaded  
INFO jenkins.InitReactorRunner$1#onAttained: System config adapted  
INFO jenkins.InitReactorRunner$1#onAttained: Loaded all jobs  
INFO jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated  
INFO hudson.model.AsyncPeriodicWork#lambda$doRun$1: Started Download metadata  
INFO hudson.util.Retrier#start: Attempt #1 to do the action check updates server  
INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization  
INFO hudson.WebAppMain$3#run: Jenkins is fully up and running
```

Jenkins

Narzędzie do ciągłej integracji,
pozwalające na ciągły rozwój, testowanie i wdrażanie nowo wytworzonego kodu



Witamy w Jenkinsie!

student

●●●●●●●●

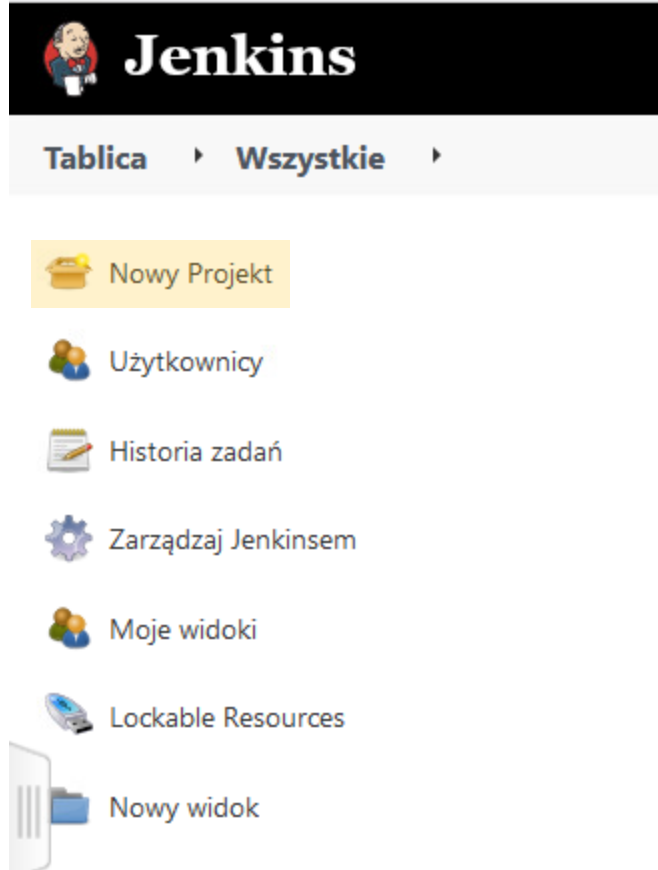
Zaloguj

☐ Zapamiętaj mnie

1. W dowolnej przeglądarce wejdź na adres:
localhost:8080
2. Zaloguj się:
login: student
hasło: Pa\$\$w0rd

Jenkins

Tworzenie joba



1. Z menu z lewej strony wybierz „Nowy Projekt”
2. Wpisz nazwę projektu, np. „Hello world”
3. Wybierz „Ogólny projekt”
4. Kliknij OK

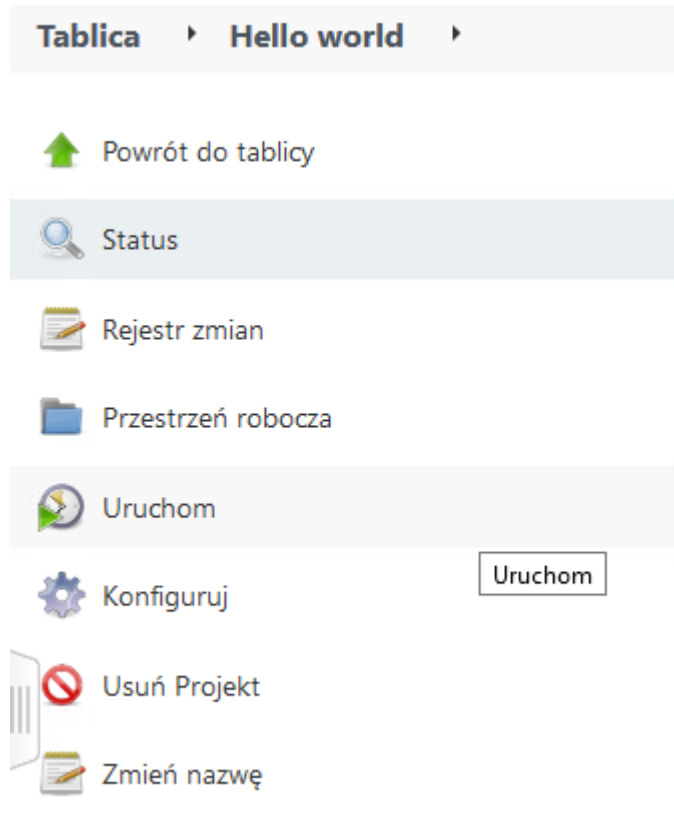
The image shows the 'Podaj nazwę projektu' dialog box. At the top, there's a title bar. Below it is a text input field containing 'Hello world'. A small note below the field says '» Pole wymagane'. Below the input field is a list of project types, each with an icon and a description:

- Ogólny projekt**: To jest podstawowa funkcja Jenkinsa. Jenkins stworzy projekt łączący dowolny SCM z dowolnym systemem budującym, może to być również wykorzystane do czegoś innego niż budowanie oprogramowania.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.

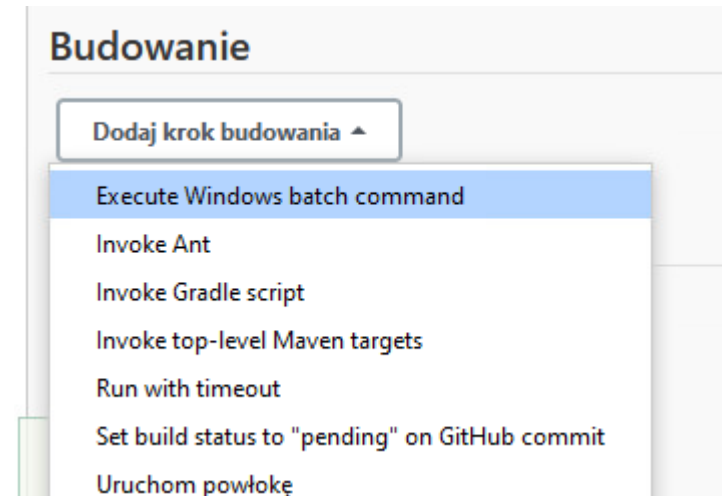
At the bottom of the dialog is an 'OK' button.

Jenkins

Tworzenie joba



1. Dodaj krótki opis projektu
2. Dodaj krok budowania: Execute Windows batch command:
echo Hello World



3. Zapisz zmiany
4. Uruchom projekt
5. Sprawdź logi

Jenkins

Modyfikacja joba

Projekt Projekt testowy

To zadanie wymaga parametrów:

Jakiś parametr

Run with last working build

Testlinia

TL 1 ▼

Buduj

1. Parametryzacja Projektu



1. Ustaw parametry typu 'Tekst wielolinijkowy'
2. Ustaw parametru typu 'Lista wyboru'
3. Uruchom projekt z parametrami.

2. Timeout

1. Dodaj krok budowania: Run with timeout
2. W kroku budowania wpisz nieskończoną pętlę.
3. Wykorzystaj zmienne z Jenkinsa.
4. Uruchom projekt z parametrami.

Pipeline Megatron_CTA6_RFSW_L1_Release

This build requires parameters:

slave	<div>Testline_CI1_10.34.188.124 ▾</div> <div>Controls which radio (slave) the test will be run on.</div>
rfsw_branch	<div>PROD_CTA6_release ▾</div> <div>Select the RFSW branch to use for update</div>
test_model	<div>TM3_1 ▾</div> <div>Test Model to use</div>
update_build_num	<div>0</div> <div>Branch-specific build number to install RFSW. This will be used only as a note if sw_update is set to false</div>
<div><input checked="" type="checkbox"/> sw_update</div> <div>Runs the SWupdate Stage with given branch/update_build_num parameters</div>	
<div><input checked="" type="checkbox"/> upload_results_to_COOP</div> <div>Select to upload results to COOP</div>	
<div><input type="checkbox"/> case1</div> <div>Run the NR100 case stage</div>	
<div><input checked="" type="checkbox"/> case2</div> <div>Run the NR20 case stage</div>	
<div><input checked="" type="checkbox"/> case3</div> <div>Run the NR50 case stage</div>	
<div><input checked="" type="checkbox"/> case4</div> <div>Run the Mix 2xNR100 case stage</div>	

Build

Jenkins

Timeout – co poszło nie tak?

Budowanie

Run with timeout

Time-out strategy

Absolute

Timeout minutes ?

1

Build Step

Execute Windows batch command

Command

```
@echo off
:x
echo Hello! This is build numer: %BUILD_NUMBER%!
goto x
```

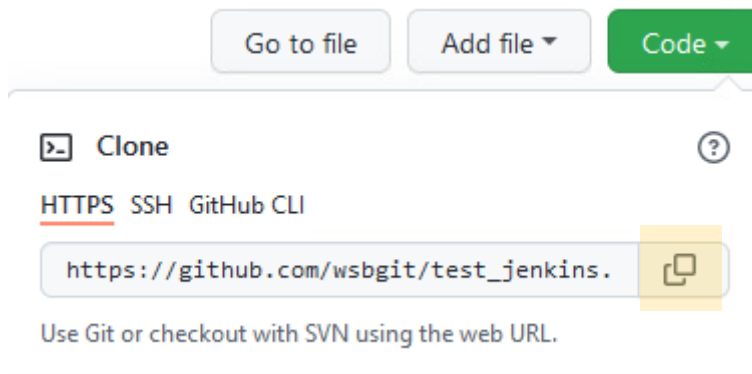
See [the list of available environment variables](#)

Zaawansowane...

Time-out actions

Fail the build

Add action ▾



Repozytorium kodu

☐ Brak

☒ Git

Repositories

Repository URL

`https://github.com/wsbgit/test_jenkins.git`

Credentials

- none -

Add

Jenkins


Połączenie z Github


1. Stwórz nowe publiczne repo na github z plikiem readme.md
2. Skopiuj link do repozytorium
3. Z menu z lewej strony wybierz: Hello world > Konfiguruj
4. Zaznacz repozytorium kodu Git i wklej adres URL
5. Zaznacz wyzwalacz zadania „Pobierz repozytorium kodu”
6. Zapisz zmiany, uruchom projekt, sprawdź logi


Jenkins


Połączenie z Github – co poszło nie tak?


Tablica ▶ **Hello world** ▶ **#2**


 Powrót do projektu


 Status


 Rejestr zmian


 Logi konsoli


 Edytuj informacje o zadaniu


 Usuń zadanie '#2'

 Poprzednie zadanie

 Następne zadanie

 **Zadanie #2 (22 mar 2022, 13:29:48)**

 Failed to determine ([log](#))

 Wystartowane przez użytkownika [student](#)

Jenkins

Połączenie z Github

✔ Zadanie #5 (22 mar 2022,



Changes

1. Update helloworld.py ([details](#) / [githubweb](#))



Wystartowane przez użytkownika [student](#)



git

Revision: 3c5f16cee9f5dedf188809b934ce252fd478f1b8

Repository: https://github.com/wsbgit/test_jenkins.git

• refs/remotes/origin/main

1. W swoim repozytorium dodaj plik helloworld.py
2. W pliku umieść prosty kod, np.
`print(Hello World! It is me!)`
3. Zakomituuj ;) zmiany
4. Zmień konfigurację swojego joba tak, by uruchomił się pobrany z repo plik.
5. Zapisz zmiany, uruchom projekt, sprawdź logi

```
C:\Users\vdi-student\.jenkins\workspace\Hello world>python helloworld.py  
Hello World! It is me!
```


Jenkins

Harmonogram

```
----- minute (0-59)
| ----- hour (0-23)
| | ----- day of the month (1-31)
| | | ----- month (1-12)
| | | | ----- day of the week (0-6) (Sunday to Saturday; 7 is also Sunday on some systems)
| | | | |
| | | | |
* * * * * <command>
```

<minuta> <godzina> <dzień-miesiąca> <miesiąc> <dzień tygodnia> <komenda>

crontab guru

The quick and simple editor for cron schedule expressions by Cronitor

<https://crontab.guru/>

Ustaw harmonogram na:

- Każdy dzień o godzinie 12:00
- Każda minuta od godziny 1 PM do godziny 1:05 PM w każdy dzień
- O 13:15 i o 13:34 w każdy poniedziałek miesiąca maja
- O 9:30 AM piętnastego dnia każdego miesiąca

Jenkins

Harmonogram

1. Zmodyfikuj swój projekt i nadaj mu cykliczne budowanie co 2 minuty
2. Zapisz zmiany, poczekaj, aż projekt sam się uruchomi.

Wyzwalacze zadania

☐ Wyzwalaj budowanie zdalnie (np. przez skrypt)

☒ Buduj cyklicznie

Harmonogram

* / 2 * * * *

Historia zadań		trend ^
szukaj		
✓ #7	22 mar 2022, 14:08	
✓ #6	22 mar 2022, 14:06	

✓ Zadanie #6 (22 mar 2022, 14:06:00)



No changes.



Uruchomione przez zegar



Revision: 3c5f16cee9f5dedf188809b934ce252fd478f1b8

Repository: https://github.com/wsbgit/test_jenkins.git

• refs/remotes/origin/main

Zarządzaj Jenkinsem

Dodaj powitanie na stronie głównej.

The screenshot shows the Jenkins web interface with the 'Konfiguracja' (Configuration) tab selected. The left sidebar contains navigation links: 'Nowy Projekt', 'Użytkownicy', 'Historia zadań', 'Zarządzaj Jenkinsem', 'Moje widoki', 'Lockable Resources', and 'Nowy widok'. Below these are sections for 'Kolejka zadań' (Queue) and 'Status wykonawców zadań' (Agent Status). The main configuration area includes the following settings:

- Katalog domowy** (Home directory): C:\Users\ydi-student\jenkins
- Komunikat systemowy** (System message): Witaj na moim osobistym CII!
- # of executors**: 2
- Labels**: (empty field)
- Plan wykorzystania** (Usage plan): Wykorzystuj ten węzeł tak bardzo, jak to tylko możliwe
- Quiet period**: 5
- SCM checkout retry count**: 0
- ☐ Restrict project naming

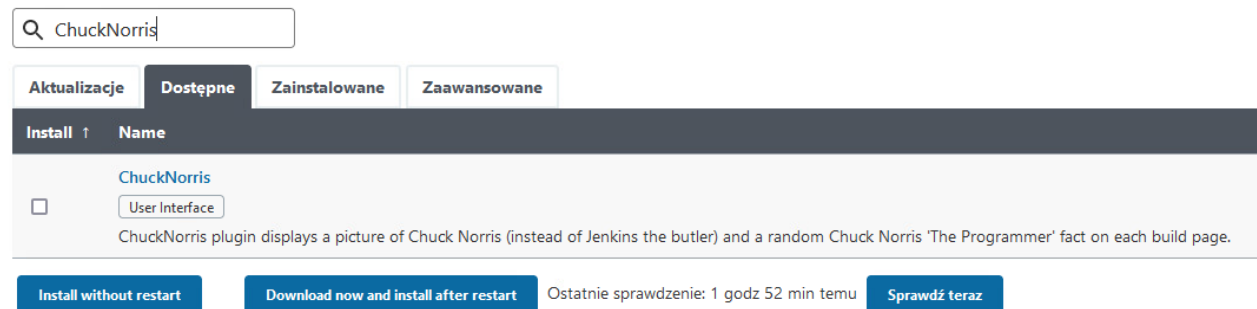
At the bottom of the configuration area are two buttons: 'Zapisz' (Save) and 'Zastosuj' (Apply).

Jenkins

Instalowanie wtyczek



1. Ze strony głównej przejdź do: Zarządzaj Jenkinsem > Zarządzaj wtyczkami
2. Zmień zakładkę na „Dostępne”
3. Wyszukaj wtyczki **ChuckNorris**



4. Zainstaluj z opcją restartu, a następnie uruchom ponownie Jenkinsa

Jenkins

Instalowanie i wykorzystanie wtyczek



1. Utwórz nowy Projekt typu ,Ogólny projekt'
2. Zadaż mu krótki opis i przejdź do sekcji Budowanie i wybierz ,Execute Windows batch command'.
3. Wpisz kod:

```
@echo off  
set racja=true  
  
:start  
if %racja% equ true echo Masz racje!  
if %racja% neq true echo Nie masz racji!  
pause
```
4. Dodaj akcję po zadaniu: Activate Chuck Norris
5. Zapisz i uruchom projekt. Sprawdź wyniki!

Ćwiczenie

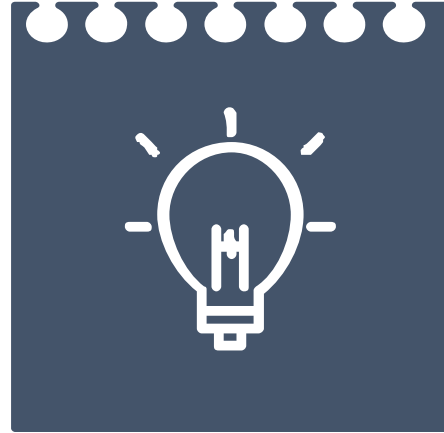
Zarządzanie wtyczkami

Zainstaluj
Green balls

01

Zainstaluj
Blue Ocean

02



03

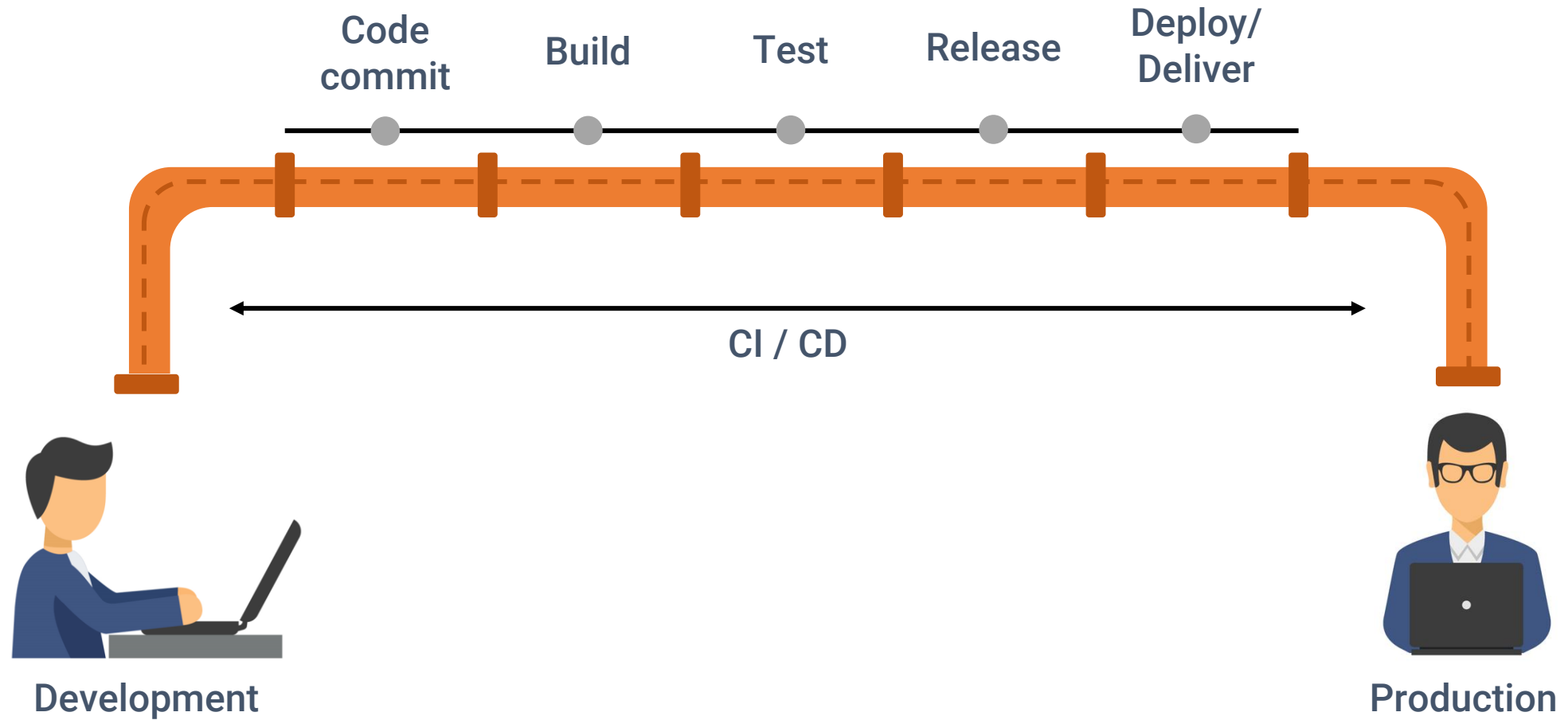
Sprawdź zainstalowane pluginy
wchodząc na adres:
Localhost:8080/script

Wykonaj script:

```
Jenkins.instance.pluginManager.plugins.each{  
    plugin -> println  
    ("${plugin.getShortName()}")  
}
```

Jenkins

Pipeline



Jenkins

Pipeline – co to jest?

Wtyczki

Zestaw wtyczek, który obsługuje implementację i integrację

Automatyzacja

Zautomatyzowany proces wytwarzania oprogramowania od repozytorium do kodu dla użytkownika

Osobny plik

Definicja Pipeline zapisana jest w pliku tekstowym (Jenkinsfile).
Można też konfigurować go z poziomu interfejsu webowego.

Rozszerzalność

Zestaw narzędzi do modelowania sposobu dostarczenia kodu.

Składnia

Obsługuje dwie składnie:

- Declarative pipeline
- Scripted pipeline

Jenkins

Tworzenie Pipeline

Pipeline

Definition

Pipeline script

Script

```
1 pipeline {  
2     agent any  
3  
4     stages {  
5         stage('Hello') {  
6             steps {  
7                 echo 'Hello World'  
8             }  
9         }  
10    }  
11 }  
12
```

1. Dodaj krótki opis. Zauważ, że dostępnych jest mniej pól.

Koncepcja Pipeline jest następująca:

1. Pipeline
2. Agent/node
3. Stage
4. Step

2. W sekcji Pipeline wykorzystaj gotowy kod „Hello World”

3. Zapisz i uruchom projekt.

Jenkins

Modyfikacja Pipeline



1. Dodaj jeszcze dwie sekcje ,stage'.
2. W pierwszej sklonuj swoje repozytorium.

```
stage('Git clone') {  
    steps {  
        git branch: 'main', url: 'https://github.com/wsbgit/test_jenkins.git'  
    }  
}
```

3. W drugiej zasymuluj testowanie.
4. W trzeciej zasymuluj dostarczenie.
2. Zapisz i uruchom projekt.

```
stage('Test') {  
    steps {  
        echo 'Testing...'  
    }  
}  
stage('Deploy') {  
    steps {  
        echo 'Deploying...'  
    }  
}
```

Stage View

	Declarative: Checkout SCM	5G-L1-LIB_checkout	PackageZip:	Velocity	Run dutControlServiceL1	run robot case	Publish Robot results	Publish Robot results to publish report	post robot build xml info to coop	post robot result info to coopDB	Declarative: Post Actions
Average stage times:	6s	27s	8min 3s	8s	4s	2h 11min	7s	15s	1min 16s	59s	23s
#3295 Mar 23 19:35 3 commits	3s	18s	7min 21s	5s	1s	in					
3294_70M_jenkins-PROD_CTA6_release-6237 Mar 23 17:12 4 commits	14s	49s	8min 11s	3s	1s	2h 10min	3s	13s	1min 17s	50s	21s
3293_60M_jenkins-PROD_CTA6_release-6237 Mar 23 14:22 4 commits	7s	24s	9min 28s	9s	6s	2h 33min	5s	14s	2min 46s	1min 8s	27s
3292_60M_jenkins-PROD_CTA6_release-6235 Mar 23 11:11 15 commits	11s	27s	8min 7s	3s	1s	2h 59min	7s	23s	2s	1min 8s	22s
3291_60M_jenkins-PROD_CTA6_release-6235 Mar 23 04:44 5 commits	8s	21s	8min 19s	6s	2s	1h 13min	3s	12s	5s	57s	11s

Stage View

		Initialize	Clone Git	Get ConfigDB	Setup Measurements	Setup TestModel & Activate Carriers	BF calibration check	Fault history log check	Cell_0 - allocation	Declarative: Post Actions
Average stage times: (Average <u>full</u> run time: ~24min 12s)		1s	806ms	0ms	1s	10min 42s	3min 6s	11s	8min 3s	58s
#3536	Mar 25 22:08 No Changes	925ms	672ms		1s	1min 46s				
22min 22s										
#3535	Mar 25 20:32 No Changes	1s	1s		1s	11min 35s	2min 38s	3s	8min 8s	58s
#3534	Mar 25 20:08 No Changes	1s	641ms		972ms	11min 19s	2min 44s	2s	8min 8s	58s
#3533	Mar 25 18:31 No Changes	1s	666ms		1s	11min 41s	2min 47s	3s	8min 11s	58s
#3532	Mar 25 18:07 No Changes	1s	659ms		1s	11min 45s	2min 44s	2s	8min 7s	58s
#3531	Mar 25 16:32 No Changes	1s	1s		993ms	11min 40s	2min 35s	6s	8min 5s	58s

Koniec

**Dziękuję
za uwagę**

