TeamCity: budowanie kodu od JetBrains

Poziom: podstawy **Dotyczy: TeamCity 2026.2 (On-Premises)** **Czas: ~15 min**

Ta strona otwiera książkę o TeamCity: skąd wziął się ten program, czym różni się od Jenkinsa i od Gitea Actions, i co czeka na kolejnych stronach. TeamCity stanie w Zielonej obok Jenkinsa, nie zamiast niego.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
YAML	YAML Ain't Markup Language: format plików konfiguracyjnych oparty na wcięciach

Skąd się wzięło

> **Pojęcie: JetBrains.** Firma programistyczna założona w 2000 roku w Pradze, znana przede wszystkim ze środowiska programistycznego IntelliJ IDEA i z języka Kotlin, który sama stworzyła. TeamCity to jeden z jej starszych produktów, rozwijany od dwudziestu lat równolegle z resztą rodziny narzędzi dla programistów.

> **Pojęcie: TeamCity.** Serwer do ciągłej integracji: przyjmuje zdarzenie z repozytorium kodu, zleca budowanie i testowanie agentowi, a wynik pokazuje w rozbudowanym interfejsie webowym z raportami, wykresami i historią. Pierwsza wersja ukazała się 2 października 2006 roku.

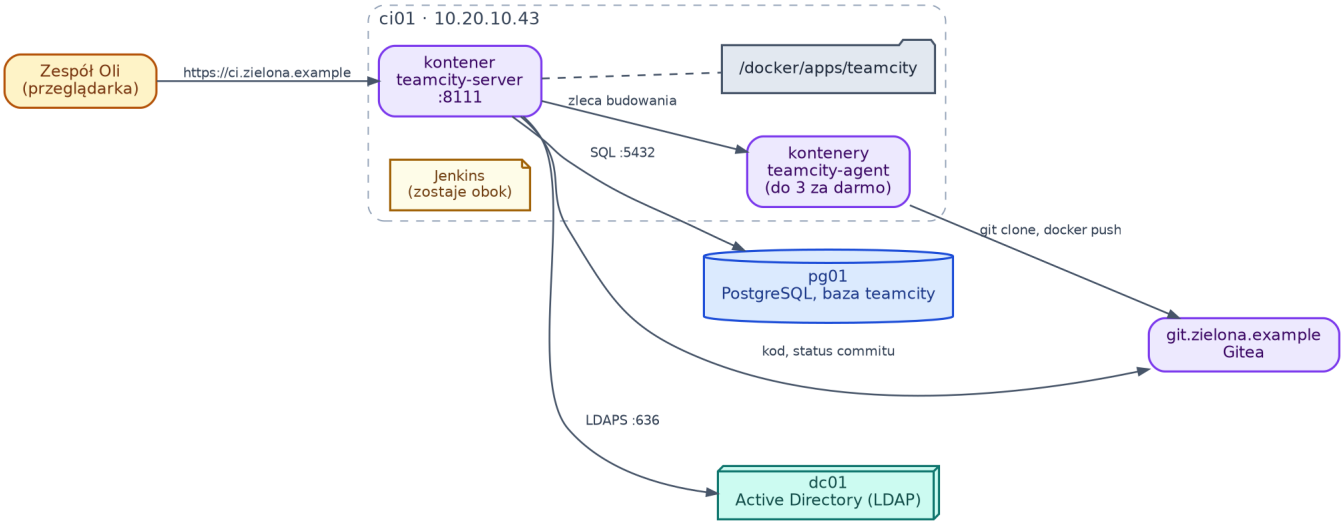
TeamCity powstał z potrzeby, którą JetBrains zauważył we własnych zespołach. Sergey Dmitriev, ówczesny prezes firmy, tłumaczył to wprost: zespoły programistów grzęzły w ręcznym budowaniu, testowaniu i wersjonowaniu, czyli w rzeczach, które dało się zautomatyzować. JetBrains chciał dać całym zespołom to, co jego środowiska programistyczne od lat dawały pojedynczym programistom: mniej czasu na proces, więcej na sam kod. Powstał więc program celujący najpierw w zespoły Javy i .NET, rok po tym, jak w świecie Javy zadebiutował Hudson, program, z którego kilka lat później, po sporze o markę, odłączy się Jenkins.

Ciekawostka: ten sam JetBrains, który w 2006 roku napisał TeamCity, stworzył pięć lat później język Kotlin. Dziś to właśnie w Kotlinie zapisujemy konfigurację TeamCity jako kod, jeśli nie chcemy trzymać jej wyłącznie w interfejsie, patrz strona Pierwszy projekt: cennik z Gitea budowany w TeamCity. Firma używa więc własnego języka do konfigurowania własnego narzędzia.

TeamCity od zawsze miał darmowy próg: instalacja bez opłat, ograniczona liczbą agentów budujących i liczbą równocześnie istniejących konfiguracji, a dopiero powyżej tego progu trzeba kupić licencję Enterprise. Szczegóły tego progu, ważne dla Zielonej, opisuje strona Plan wdrożenia TeamCity w Zielonej.

Jak to działa w jednym obrazku

Zespół Oli otwiera TeamCity w przeglądarce pod adresem `ci.zielona.example`, na serwerze `ci01`, tym samym, na którym od lat stoi Jenkins. Serwer TeamCity trzyma konfigurację projektów w bazie PostgreSQL na `pg01`, a logowanie sprawdza w Active Directory na `dc01`, dokładnie tak samo jak Gitea. Same kroki budowania, checkout kodu, kompilację, testy i budowanie obrazu, wykonują osobne procesy: agenci TeamCity, uruchomieni jako kontenery obok serwera.



Architektura TeamCity w Zielonej: przeglądarka, serwer, agenci, baza i Active Directory, obok Jenkinsa (kliknięcie otwiera pełny rozmiar)

Dlaczego Zielona

Z ŻYCIA WZIĘTE

Wtorek, po południu. Ola prowadzi rozmowę z panem Zbigniewem o tym, dlaczego zespół chce postawić obok Jenkinsa jeszcze jeden serwer budowania. Jenkins działa od lat i nikt nie planuje go wyłączać, ale każdy nowy pipeline w nim to najpierw dobór wtyczek, potem Groovy w Jenkinsfile, a na końcu i tak ręczne poprawki w interfejsie, żeby cokolwiek zadziałało. Ola chce dać zespołowi trzeciej osoby, Tomkowi, narzędzie z gotowym kreatorem projektu i czytelnymi raportami od pierwszego dnia, bez tygodnia nauki samego Jenkinsa.

TeamCity odpowiada na inną potrzebę niż Gitea Actions, opisane w książce o Gitea, i inną niż Jenkins. Gitea Actions wystarcza, gdy cały przepis na budowanie mieści się w jednym pliku YAML i nikt nie potrzebuje rozbudowanych raportów. Jenkins dźwiga najbardziej złożone, wieloletnie pipeline'y, kosztem tego, że każdy z nich trzeba było kiedyś ręcznie złożyć z wtyczek. TeamCity celuje pośrodku: konfiguracja projektu powstaje głównie klikaniem w interfejsie, z gotowymi krokami do budowania obrazów Dockera i publikowania ich w rejestrze, a historia buildów, wykresy i porównania między nimi są tam od razu, bez dodatkowych wtyczek.

Zespół Oli przenosi do TeamCity nowe zadania i te z Jenkinsa, które najbardziej korzystają na czytelnym interfejsie, a resztę zostawia tam, gdzie działa. Pan Zbigniew, jak zwykle, pyta najpierw o koszt: darmowy próg TeamCity starcza dla całego zespołu Oli na wiele miesięcy, więc odpowiedź na pierwsze pytanie brzmi po prostu „nic", z zastrzeżeniami opisanymi na kolejnej stronie.

Mapa tej książki

Książka o TeamCity ma dziesięć stron.

Strona	Na jakie pytanie odpowiada
TeamCity: budowanie kodu od JetBrains	Skąd wziął się TeamCity i po co stawiamy go obok Jenkinsa
Plan wdrożenia TeamCity w Zielonej	Co zrobimy, w jakiej kolejności, ile to kosztuje i co zostaje w Jenkinsie
Instalacja TeamCity w Dockerze	Jak postawić serwer i agenta w kontenerach, z bazą na pg01
Instalacja TeamCity bez Dockera	Jak postawić serwer jako usługę systemd, bez kontenerów
Agenci TeamCity: dodanie, autoryzacja, pule	Czym jest agent i pula, jak dodać kolejnego i co robi darmowy limit trzech
Agent TeamCity w Dockerze na maszynie fizycznej: Linux i Windows, agent, który przenosi się między hostami	Jak uruchomić agenta z GPU lub Windows w kontenerze i przenieść go na inny host bez zajmowania drugiej licencji
Logowanie kontem z Active Directory	Jak podłączyć logowanie przez LDAP zamiast kont lokalnych
Pierwszy projekt: cennik z Gitea budowany w TeamCity	Jak zbudować obraz z repozytorium Gitea i zobaczyć wynik jako status commitu

Strona	Na jakie pytanie odpowiada
Kopia zapasowa i aktualizacja TeamCity	Jak bezpiecznie zrobić backup i podnieść wersję
TeamCity: częste problemy	Co robić, gdy coś nie działa

Powiązane: Architektura Jenkinsa, kontroler, agenci, workspace · Gitea Actions: runner i pierwszy workflow · Active Directory: konta, grupy, jednostki organizacyjne

Źródła: en.wikipedia.org/wiki/TeamCity · blog.jetbrains.com, ogłoszenie TeamCity z 2 października 2006 · jetbrains.com/help/teamcity · blog.jetbrains.com/teamcity, wpis o wydaniu 2026.2 (7 września 2026)

Revision #32
Created 2026-09-27 11:19:28 UTC by Tomasz Konieczny
Updated 2026-10-26 14:17:39 UTC by Tomasz Konieczny