Plan wdrożenia TeamCity w Zielonej

Typ: procedura **Dotyczy: TeamCity 2026.2 (On-Premises), serwer ci01**

Czas: ~25 min lektury

Ten plan prowadzi zespół Oli od pustego serwera `ci01` do działającego TeamCity, obok istniejącego Jenkinsa, bez przerywania niczego, co dziś już działa. Ustalamy tu cel, koszt, kolejność etapów i to, co zostaje po staremu.

Skróty użyte na tej stronie (1)

Skrót	Co znaczy
HSQLDB	HyperSQL DataBase: baza danych wbudowana w sam TeamCity, zapisywana na dysku obok reszty danych serwera, przeznaczona wyłącznie do testów i ewaluacji

Zanim zaczniemy

Cel. Zespół Oli dostaje drugi serwer budowania, obok Jenkinsa, z gotowym interfejsem do klikania nowych projektów bez uczenia się Groovy i wtyczek od zera. Pierwszym celem jest jeden działający projekt: repozytorium `zielona/cennik` budowane i publikowane przez TeamCity, z widocznym statusem w Gitea.

Zakres. Plan obejmuje instalację serwera i agentów na `ci01`, bazę na `pg01`, logowanie przez Active Directory na `dc01` i pierwszy projekt z Gitea. Nie obejmuje przeniesienia całego Jenkinsa naraz: migrację zadań traktujemy jako osobny, ostatni etap, zadanie po zadaniu, nie jako jednorazowy skok.

Wymaganie	Uwaga
Serwer `ci01`, Rocky Linux 9	dziś już z Jenkinsem, TeamCity staje obok
Baza PostgreSQL na `pg01`	osobna baza `teamcity`, oddzielona od bazy `gitea`
Konto techniczne w Active Directory	do logowania przez LDAP, patrz Logowanie kontem z Active Directory

Wymaganie	Uwaga
Repozytorium `zielona/cennik` w Gitea	to samo, które zna książka o Gitea
Zgoda Oli na kolejność etapów niżej	migracja z Jenkinsa jest ostatnim, nie pierwszym krokiem

Licencja: co jest za darmo

Zanim ustalimy koszt, prostujemy jedno założenie. Darmowa edycja TeamCity, nazwana Professional, nie ogranicza liczby kont ani osób logujących się do systemu: dziesięć czy dwadzieścia osób loguje się do tej samej darmowej instalacji bez żadnej dodatkowej opłaty. Ogranicza za to trzy inne rzeczy naraz, sprawdzone w dokumentacji licencyjnej JetBrains 27 września 2026 roku.

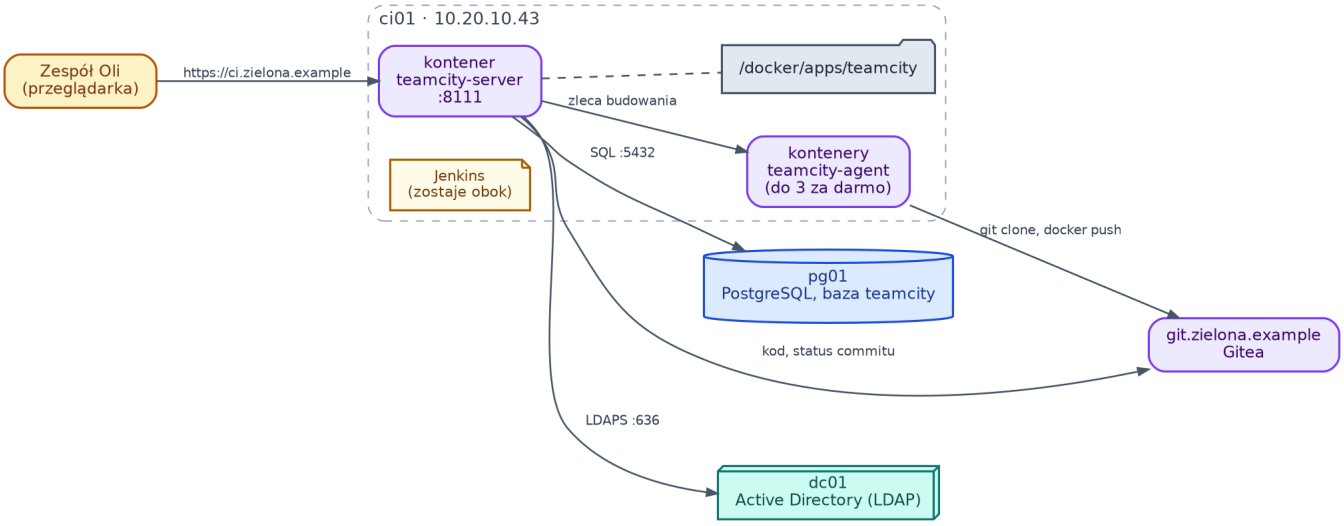
Co ogranicza Professional	Darmowy limit	Co robi dokupiona licencja agenta
Agenci budujący	3	każda dokupiona licencja dodaje jednego agenta
Konfiguracje budowania, razem z zarchiwizowanymi	100 naraz	każda dokupiona licencja agenta podnosi ten limit o 10
Pipeline'y, razem z zarchiwizowanymi	10 naraz	limit rośnie razem z limitem konfiguracji
Wsparcie i aktualizacje	dokumentacja i społeczność	edycja Enterprise dodaje roczną subskrypcję z priorytetowym wsparciem

Po przekroczeniu limitu konfiguracji albo pipeline'ów TeamCity nie usuwa nic samo z siebie: nowe buildy po prostu przestają się uruchamiać, dopóki nie usuniemy albo nie zarchiwizujemy części starszych konfiguracji, albo nie dokupimy licencji. Edycja Enterprise, płatna, znosi limit konfiguracji i pipeline'ów całkowicie, zostawiając tylko limit agentów, powiększany dalej przez dokupione licencje.

Dla zespołu Oli, trzech osób i jednego repozytorium na start, trzy darmowe agenty i sto miejsc na konfiguracje starczą na wiele miesięcy praktyki, zanim temat płatnej licencji w ogóle stanie się aktualny.

> TeamCity Cloud, wersja hostowana przez samego JetBrains, rządzi się zupełnie innym modelem opłat: nie liczy agentów ani konfiguracji, tylko osoby wysyłające co najmniej dziesięć commitów miesięcznie („committerów"), a do tego dolicza miesięczny limit miejsca na dysku, transferu i tak zwanych kredytów budowania. Ten model nie dotyczy Zielonej: instalujemy wersję On-Premises, opisaną w tym planie, we własnej serwerowni.

Architektura docelowa



Architektura TeamCity w Zielonej: serwer, agenci, baza na pg01, Gitea i Active Directory, obok Jenkinsa (kliknięcie otwiera pełny rozmiar)

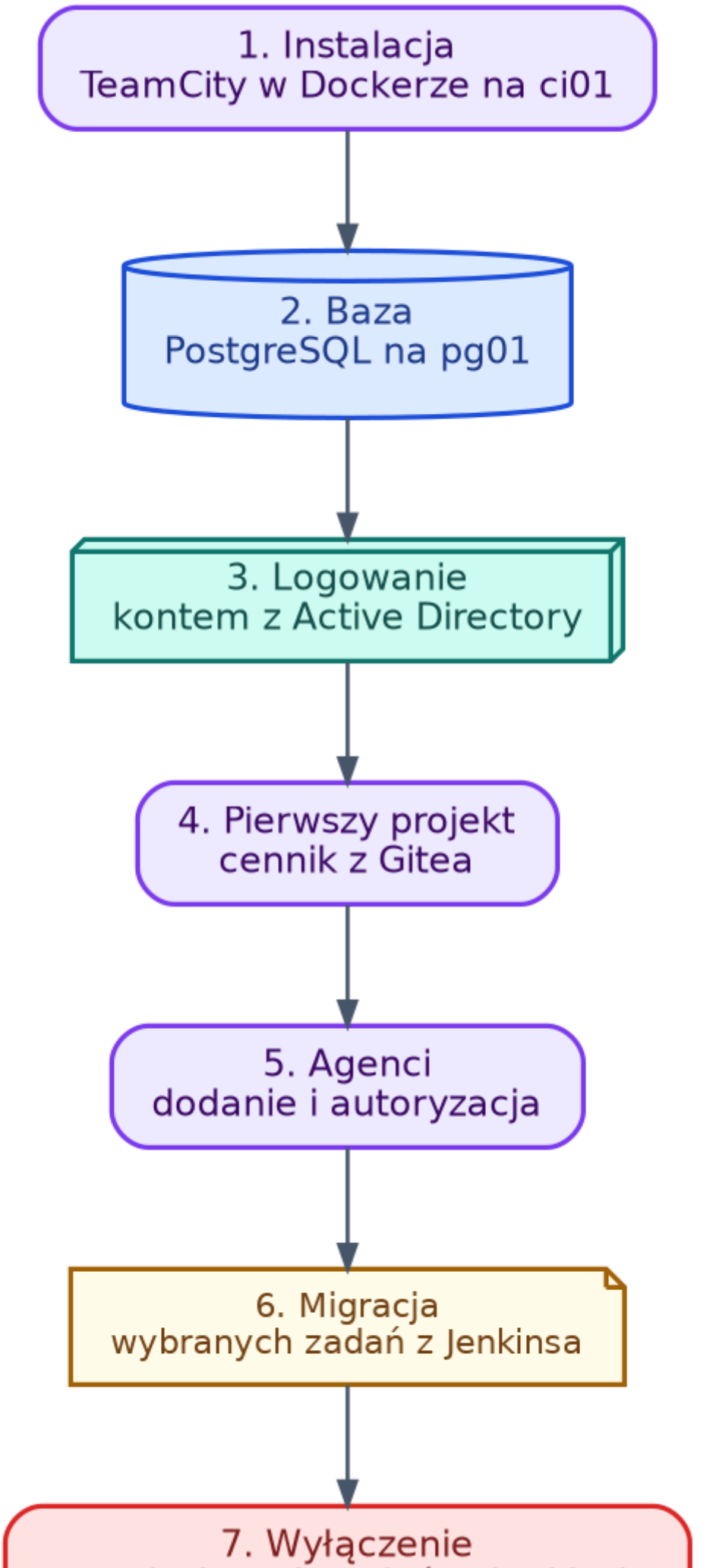
Serwer TeamCity i jego agenci stają na `ci01`, tym samym serwerze co dzisiejszy Jenkins, jako osobne kontenery. Konfiguracja i historia buildów trafiają do bazy PostgreSQL na `pg01`, logowanie do Active Directory na `dc01`, a kod i wynikowe obrazy do Gitea na `docker01`. Żaden z tych elementów nie zastępuje odpowiednika po stronie Jenkinsa: oba systemy mają własną bazę i własnych agentów, nawet jeśli dzielą ten sam serwer fizyczny.

Wymagania sprzętowe

Element	Wymaganie	Uwaga dla Zielonej
Serwer TeamCity	co najmniej 4 rdzenie CPU na produkcję, 16 gigabajtów pamięci RAM starcza na sto równoległych buildów i dwustu użytkowników online	zespół Oli zmieści się na ułamku tego zasobu, `ci01` ma zapas
Agent budujący	około 500 megabajtów pamięci RAM ponad to, co zużywa sam build	trzy darmowe agenty to około 1,5 gigabajta RAM dodatkowo
Baza danych	PostgreSQL na produkcję, wbudowana baza HSQLDB tylko do testów, patrz niżej	współdzielimy `pg01` z Gitea, osobna baza `teamcity`
Dysk	rośnie z historią buildów i logami, nie z samą instalacją	katalog `/docker/apps/teamcity` na wolumenie `vgDocker/lvDocker`

Pojęcie: HSQLDB. Baza danych wbudowana w sam TeamCity, zapisywana jako pliki obok reszty danych serwera, bez osobnej instalacji. Dokumentacja JetBrains wprost odradza jej używania poza testami i ewaluacją: brak łatwej kopii zapasowej na żywo i ograniczona wydajność przy większej liczbie buildów sprawiają, że każda instalacja produkcyjna przechodzi na zewnętrzną bazę, u nas na PostgreSQL na `pg01`, jeszcze przed pierwszym prawdziwym projektem.

Przebieg: kolejność etapów


1. Instalacja
TeamCity w Dockerze na ci01
2. Baza
PostgreSQL na pg01
3. Logowanie
kontem z Active Directory
4. Pierwszy projekt
cennik z Gitea
5. Agenci
dodanie i autoryzacja
6. Migracja
wybranych zadań z Jenkinsa
7. Wyłączenie

Kolejność etapów wdrożenia TeamCity w Zielonej (kliknięcie otwiera pełny rozmiar)

Etap 1: instalacja

Stawiamy serwer TeamCity w Dockerze na `ci01`, zgodnie ze stroną Instalacja TeamCity w Dockerze. **Kryterium zakończenia:** kreator pierwszego uruchomienia otwiera się w przeglądarce pod `https://ci.zielona.example` i pokazuje ekran wyboru bazy danych.

Etap 2: baza

Podłączamy zewnętrzną bazę PostgreSQL na `pg01`, zamiast wbudowanej HSQLDB z kreatora. **Kryterium zakończenia:** serwer TeamCity restartuje się bez błędu połączenia z bazą, a w PostgreSQL na `pg01` widać wypełnione tabele schematu TeamCity.

Etap 3: LDAP

Podłączamy logowanie kontem z Active Directory, zgodnie ze stroną Logowanie kontem z Active Directory. **Kryterium zakończenia:** Tomek loguje się do TeamCity swoim kontem domenowym, bez konta lokalnego założonego ręcznie.

Etap 4: pierwszy projekt z Gitea

Zakładamy projekt budujący repozytorium `zielona/cennik`, zgodnie ze stroną Pierwszy projekt: cennik z Gitea budowany w TeamCity. **Kryterium zakończenia:** push do gałęzi `main` w Gitea kończy się zielonym buildem w TeamCity i widocznym statusem commitu w samej Gitea.

Etap 5: agenci

Dodajemy i autoryzujemy agentów potrzebnych do równoległej pracy zespołu, zgodnie ze stroną Agenci TeamCity: dodanie, autoryzacja, pule. **Kryterium zakończenia:** dwa buildy uruchomione jeden po drugim nie czekają w kolejce na tego samego, jedynego agenta.

Etap 6: migracja wybranych zadań z Jenkinsa

Przenosimy do TeamCity te zadania z Jenkinsa, które najbardziej korzystają na czytelnym interfejsie i prostszej konfiguracji, jedno po drugim, nie hurtowo. **Kryterium zakończenia:** przeniesione zadanie wykonało w TeamCity co najmniej pięć udanych buildów z rzędu, na tych samych danych wejściowych co jego odpowiednik w Jenkinsie.

Etap 7: wyłączenie przeniesionych zadań w Jenkinsie

Dopiero gdy TeamCity potwierdził parytet w etapie 6, wyłączamy odpowiadające zadanie w Jenkinsie, zamiast je usuwać. **Kryterium zakończenia:** zadanie w Jenkinsie ma status wyłączonego (`disabled`), nie usuniętego, przez co najmniej miesiąc, zanim rozważymy jego skasowanie.

Jenkins kontra TeamCity kontra Gitea Actions: tabela decyzji

Typowe zadanie w Zielonej	Wybieramy	Dlaczego
Statyczna strona cennika, jeden Dockerfile	Gitea Actions	cały przepis mieści się w jednym pliku YAML obok kodu, bez osobnego serwera
Nowy, mały projekt zespołu Oli z testami i raportem pokrycia	TeamCity	gotowe raporty i wykresy trendu bez dodatkowych wtyczek, konfiguracja głównie klikaniem
Wieloletni, złożony pipeline z dziesiątkami wtyczek, już działający	Jenkins	działa, zespół zna go od lat, migracja bez wyraźnej korzyści to tylko ryzyko
Zadanie wymagające ręcznego zatwierdzenia przed wdrożeniem na produkcję	TeamCity albo Jenkins	oba to potrafią, wybór zależy od tego, gdzie żyje reszta powiązanych kroków
Szybki eksperyment na jednorazowym repozytorium	Gitea Actions	najmniej do skonfigurowania, najmniej do posprzątania po skończonym eksperymencie

Wycofanie zmian

Ryzyko: przekroczenie darmowego limitu konfiguracji. Zbyt entuzjastyczne zakładanie nowych projektów w TeamCity, bez porządkowania starych, kończy się zatrzymanymi nowymi

buildami po stu konfiguracjach. Zapobiegamy temu, archiwizując konfiguracje nieużywane od dawna, zanim w ogóle zbliżymy się do limitu.

Ryzyko: migracja zadania z Jenkinsa psuje coś, czego etap 6 nie przewidział. Zanim wyłączymy cokolwiek w Jenkinsie, etap 7 wymaga pięciu udanych buildów z rzędu w TeamCity: to jest właśnie zabezpieczenie przed tym ryzykiem, nie formalność do odhaczenia.

Ryzyko: baza `teamcity` obciąża współdzielony `pg01`. Baza TeamCity rośnie razem z historią buildów. Jeśli `pg01` zacznie odczuwać dodatkowe obciążenie, przenosimy bazę `teamcity` na własną instancję PostgreSQL, bez zmiany niczego po stronie samego TeamCity poza adresem w `database.properties`.

Plan wycofania całości. Dopóki żadne zadanie z Jenkinsa nie zostało wyłączone (etap 7 jeszcze się nie wydarzył), wycofanie całego wdrożenia to zatrzymanie i usunięcie kontenerów TeamCity oraz bazy `teamcity` na `pg01`: Jenkins nie zauważa tej zmiany, bo nigdy nie zależał od TeamCity. Po etapie 7 wycofanie oznacza też ponowne włączenie wyłączonych zadań w Jenkinsie, stąd zasada trzymania ich jako `disabled`, nie usuniętych, przez pierwszy miesiąc.

Co zostaje w Jenkinsie i dlaczego

Zespół Oli zostawia w Jenkinsie zadania, których migracja kosztowałaby więcej czasu, niż dałaby korzyści: wieloletnie pipeline'y ze skomplikowanym Jenkinsfile, korzystające z wtyczek bez prostego odpowiednika w TeamCity, oraz zadania uruchamiane tak rzadko, że nikt nie odczuwa niewygody obecnego interfejsu. Trzymanie ich w Jenkinsie chroni też darmowy limit stu konfiguracji w TeamCity: przenosimy tam to, co realnie korzysta na zmianie, nie wszystko, co da się przenieść.

Powiązane: TeamCity: budowanie kodu od JetBrains · Architektura Jenkinsa, kontroler, agenci, workspace · Gitea Actions: runner i pierwszy workflow · Agent TeamCity w Dockerze na maszynie fizycznej: Linux i Windows, agent, który przenosi się między hostami · TeamCity: częste problemy

Źródła: jetbrains.com/help/teamcity/licensing-policy.html (sprawdzone 27.09.2026) · jetbrains.com/help/teamcity/system-requirements.html · teamcity-support.jetbrains.com, artykuł o licencjonowaniu TeamCity Cloud

---

Revision #31
Created 2026-09-27 11:19:30 UTC by Tomasz Konieczny
Updated 2026-10-26 14:17:41 UTC by Tomasz Konieczny