Instalacja TeamCity w Dockerze

Typ: procedura

Dotyczy: TeamCity 2026.2, obrazy jetbrains/teamcity-server:2026.2 i jetbrains/teamcity-agent:2026.2

Czas: ~35 min

Efekt: działający serwer TeamCity z jednym agentem, dostępny po WWW na porcie 8111, z bazą PostgreSQL na `pg01`, gotowy do przejścia kreatora w przeglądarce.

Skróty użyte na tej stronie (5)

Skrót	Co znaczy
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
HSQLDB	HyperSQL DataBase: baza danych wbudowana w sam TeamCity, zapisywana na dysku obok reszty danych serwera, przeznaczona wyłącznie do testów i ewaluacji
JDBC	Java Database Connectivity: standardowy sposób, w jaki program w Javie łączy się z bazą danych
LDAP	Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA)
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

Zanim zaczniemy

Komendy wykonujemy z CLI bash jako user z uprawnieniami sudo albo root, na `ci01`, z zainstalowanym Dockerem i wtyczką Compose. Poniżej dwa warianty: laboratoryjny, z wbudowaną bazą HSQLDB do samego zapoznania się z interfejsem, oraz produkcyjny Zielonej, z bazą na osobnym serwerze `pg01`. Do jakiejkolwiek pracy zespołu wybieramy wariant b: wariant a nie nadaje się do niczego poza pierwszym rzutem oka na ekran.

> W naszym środowisku dane aplikacji trzymamy w `/docker/apps/teamcity`, a plik Compose w `/docker/compose/teamcity/compose.yaml`. To konwencja, nie wymóg Dockera: można ją

> zmienić, byle w całej firmie była jedna. Katalog `/docker` to osobny wolumen `vgDocker/lvDocker`, żeby pełny dysk aplikacji nie zatrzymał systemu.

Wymaganie	Uwaga
Docker Engine i wtyczka Compose	sprawdzamy `docker compose version`
Wolny port `8111` na hoście	interfejs WWW serwera
Wariant b: osiągalny `pg01` na porcie `5432`	baza tworzona ręcznie przed pierwszym startem
Wariant b: konto i baza założone w PostgreSQL na `pg01`	patrz krok 2 wariantu b

Uważaj: obraz zawsze przypinamy do konkretnej wersji (`2026.2`), nigdy do `latest`. Aktualizacja to świadoma zmiana numeru w `compose.yaml`, opisana na stronie Kopia zapasowa i aktualizacja TeamCity, nie przypadkowe pobranie nowszej wersji przy kolejnym `docker compose pull`.

Przebieg

Katalogi danych, zgodnie z konwencją ścieżek

```
sudo mkdir -p /docker/apps/teamcity/data /docker/apps/teamcity/logs
/docker/apps/teamcity/agent-conf
sudo mkdir -p /docker/compose/teamcity
```

Katalog `data` trzyma całą konfigurację serwera, w tym poświadczenia i klucz szyfrujący: to jest odpowiednik `JENKINS_HOME` po stronie TeamCity, opisanego w książce o Jenkinsie na stronie Architektura Jenkinsa, kontroler, agenci, workspace.

Wariant a: laboratorium, wbudowana baza HSQLDB

`/docker/compose/teamcity/compose.yaml`:

```
services:
  server:
    image: jetbrains/teamcity-server:2026.2
```

```
    container_name: teamcity-server
    restart: always
    ports:
      - "8111:8111"
    volumes:
      - /docker/apps/teamcity/data:/data/teamcity_server/datadir
      - /docker/apps/teamcity/logs:/opt/teamcity/logs

  agent:
    image: jetbrains/teamcity-agent:2026.2
    container_name: teamcity-agent-1
    restart: always
    environment:
      - SERVER_URL=http://server:8111
      - AGENT_NAME=ci01-agent-1
    volumes:
      - /docker/apps/teamcity/agent-conf:/data/teamcity_agent/conf
    depends_on:
      - server
```

Pierwsze uruchomienie:

```
cd /docker/compose/teamcity
docker compose up -d
```

Uważaj: ten wariant zapisuje bazę na dysku wewnątrz `datadir`, bez osobnego serwera bazy danych. Dokumentacja JetBrains wprost ogranicza wbudowaną bazę do testów i ewaluacji: brak wygodnej kopii zapasowej na żywo i słabsza wydajność przy większej liczbie buildów wykluczają ją z jakiejkolwiek pracy produkcyjnej.

Wariant b: produkcja Zielonej, baza na zewnętrznym pg01

Zanim uruchomimy kontener, zakładamy bazę i użytkownika na `pg01`. Łączymy się tam jako administrator PostgreSQL:

```
psql -h pg01 -U postgres
```

W sesji `psql` tworzymy bazę i konto aplikacji:

```
CREATE DATABASE teamcity WITH ENCODING 'UTF8';
CREATE USER teamcity WITH ENCRYPTED PASSWORD 'HasloDoBazy';
GRANT ALL PRIVILEGES ON DATABASE teamcity TO teamcity;
ALTER DATABASE teamcity OWNER TO teamcity;
```

TeamCity nie zawiera w obrazie sterownika JDBC do PostgreSQL: pobieramy go raz, do katalogu danych, gdzie serwer sam go znajdzie przy starcie.

```
sudo mkdir -p /docker/apps/teamcity/data/lib/jdbc
sudo curl -L -o /docker/apps/teamcity/data/lib/jdbc/postgresql-42.7.9.jar \
  https://jdbc.postgresql.org/download/postgresql-42.7.9.jar
```

Plik `/docker/compose/teamcity/.env` z hasłem do bazy, obok pliku `compose.yaml`, nie w nim samym:

```
TEAMCITY_DB_PASSWORD=HasloDoBazy
```

Plik `/docker/compose/teamcity/compose.yaml`:

```
services:
  server:
    image: jetbrains/teamcity-server:2026.2
    container_name: teamcity-server
    restart: always
    ports:
      - "8111:8111"
    environment:
      - TEAMCITY_DB_URL=jdbc:postgresql://pg01:5432/teamcity
      - TEAMCITY_DB_USER=teamcity
      - TEAMCITY_DB_PASSWORD=${TEAMCITY_DB_PASSWORD}
    volumes:
      - /docker/apps/teamcity/data:/data/teamcity_server/datadir
      - /docker/apps/teamcity/logs:/opt/teamcity/logs

  agent:
    image: jetbrains/teamcity-agent:2026.2
    container_name: teamcity-agent-1
    restart: always
    environment:
      - SERVER_URL=http://server:8111
      - AGENT_NAME=ci01-agent-1
    volumes:
```

```
      - /docker/apps/teamcity/agent-conf:/data/teamcity_agent/conf
      - /var/run/docker.sock:/var/run/docker.sock
    depends_on:
      - server
```

> Wolumen `/var/run/docker.sock` daje kontenerowi agenta pełną władzę nad demonem Dockera hosta `ci01`, dokładnie z tym samym ryzykiem, co przy runnerze Gitea opisanym na stronie Gitea Actions: runner i pierwszy workflow: build wykonany przez tego agenta mógłby zatrzymać albo odczytać dane dowolnego innego kontenera na `ci01`. Montujemy to gniazdo tylko wtedy, gdy agent faktycznie ma budować obrazy Dockera, i tylko dla projektów, którym ufamy.

Pierwsze uruchomienie:

```
cd /docker/compose/teamcity
docker compose up -d
```

Kreator instalacji w przeglądarce

Otwieramy adres serwera:

```
http://ci01:8111
```

Wariant b, z bazą już wskazaną przez zmienne środowiskowe, pomija ekran wyboru bazy danych i przechodzi wprost do akceptacji licencji, a potem do założenia pierwszego konta administratora. Dalsze kroki kreatora, LDAP i pierwszy projekt, opisują kolejne strony tej książki.

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: pierwszy ekran otwarty pod `http://ci01:8111`, ekran „Database connection setup" z widoczną wartością „PostgreSQL" i statusem „Connection successful".

Weryfikacja

Sprawdzamy, że kontenery działają:

```
docker compose ps
```

Przykład:

```
NAME               IMAGE                                  STATUS
teamcity-server    jetbrains/teamcity-server:2026.2       Up 3 minutes
teamcity-agent-1   jetbrains/teamcity-agent:2026.2        Up 3 minutes
```

Sprawdzamy log serwera, szukając potwierdzenia połączenia z bazą:

```
docker compose logs --tail 50 server
```

Przykład:

```
teamcity-server  | [TeamCity] Using database: jdbc:postgresql://pg01:5432/teamcity
teamcity-server  | [TeamCity] Server startup finished
```

Sprawdzamy, że agent zdążył się już zarejestrować i czeka na autoryzację, zgodnie ze stroną Agenci TeamCity: dodanie, autoryzacja, pule:

```
Administracja → Agents → Unauthorized
```

Wycofanie zmian

Zatrzymanie i usunięcie kontenerów, bez kasowania danych z wolumenów:

```
docker compose down
```

Pełne usunięcie razem z danymi, tylko gdy naprawdę chcemy zacząć od zera:

```
docker compose down -v
sudo rm -rf /docker/apps/teamcity
```

Baza `teamcity` na `pg01` zostaje po tej operacji nietknięta: usuwamy ją osobno, tym samym poleceniem `DROP DATABASE teamcity`, tylko jeśli naprawdę nie planujemy wracać do tej instalacji.

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa

Kontener `server` restartuje się w pętli zaraz po starcie	brak sterownika JDBC w `data/lib/jdbc` albo `pg01` nieosiągalny	`docker compose logs server`, sprawdzamy plik `.jar` i `telnet pg01 5432` z hosta `ci01`
W logu serwera `FATAL: password authentication failed for user "teamcity"`	hasło w `.env` inne niż w bazie na `pg01`	poprawiamy hasło w `.env` albo w bazie przez `ALTER USER teamcity WITH PASSWORD`
Kreator w przeglądarce wisi na ekranie „Starting..." dłużej niż kilka minut	za mało pamięci przydzielonej kontenerowi serwera	sprawdzamy `docker stats teamcity-server`, podnosimy limit RAM hosta albo `TEAMCITY_SERVER_MEM_OPTS`
Agent w ogóle nie pojawia się na liście `Unauthorized`	zły adres w `SERVER_URL` albo agent nie widzi serwera po sieci Dockera	sprawdzamy `docker compose logs agent`, `SERVER_URL` musi wskazywać nazwę usługi `server`, nie `localhost`

Powiązane: Instalacja TeamCity bez Dockera · Agenci TeamCity: dodanie, autoryzacja, pule · Konwencja Dockera w Zielonej

Źródła: jetbrains.com/help/teamcity, rozdział „Installing and Configuring the TeamCity Server" · github.com/JetBrains/teamcity-docker-images, `dockerhub/teamcity-server/README.md` i `dockerhub/teamcity-agent/README.md` · jetbrains.com/help/teamcity, rozdział „Set up External Database"

Revision #31
Created 2026-09-27 11:19:32 UTC by Tomasz Konieczny
Updated 2026-10-26 14:17:42 UTC by Tomasz Konieczny