Instalacja i konfiguracja

W Dockerze i bez Dockera, kreator, reverse proxy, logowanie przez Active Directory.

- Instalacja Gitea w Dockerze
- Instalacja Gitea bez Dockera
- Pierwsze uruchomienie: kreator, administrator, reverse proxy
- Logowanie kontem z Active Directory

Instalacja Gitea w Dockerze

Typ: procedura **Dotyczy: Gitea 1.27.3, obraz docker.gitea.com/gitea:1.27.3**

Czas: ~30 min

Efekt: działający kontener Gitea z bazą PostgreSQL, dostępny po WWW na porcie 3000 i po SSH na porcie 2222, gotowy do pierwszej konfiguracji przez przeglądarkę.

Skróty użyte na tej stronie (3)

Skrót	Co znaczy
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

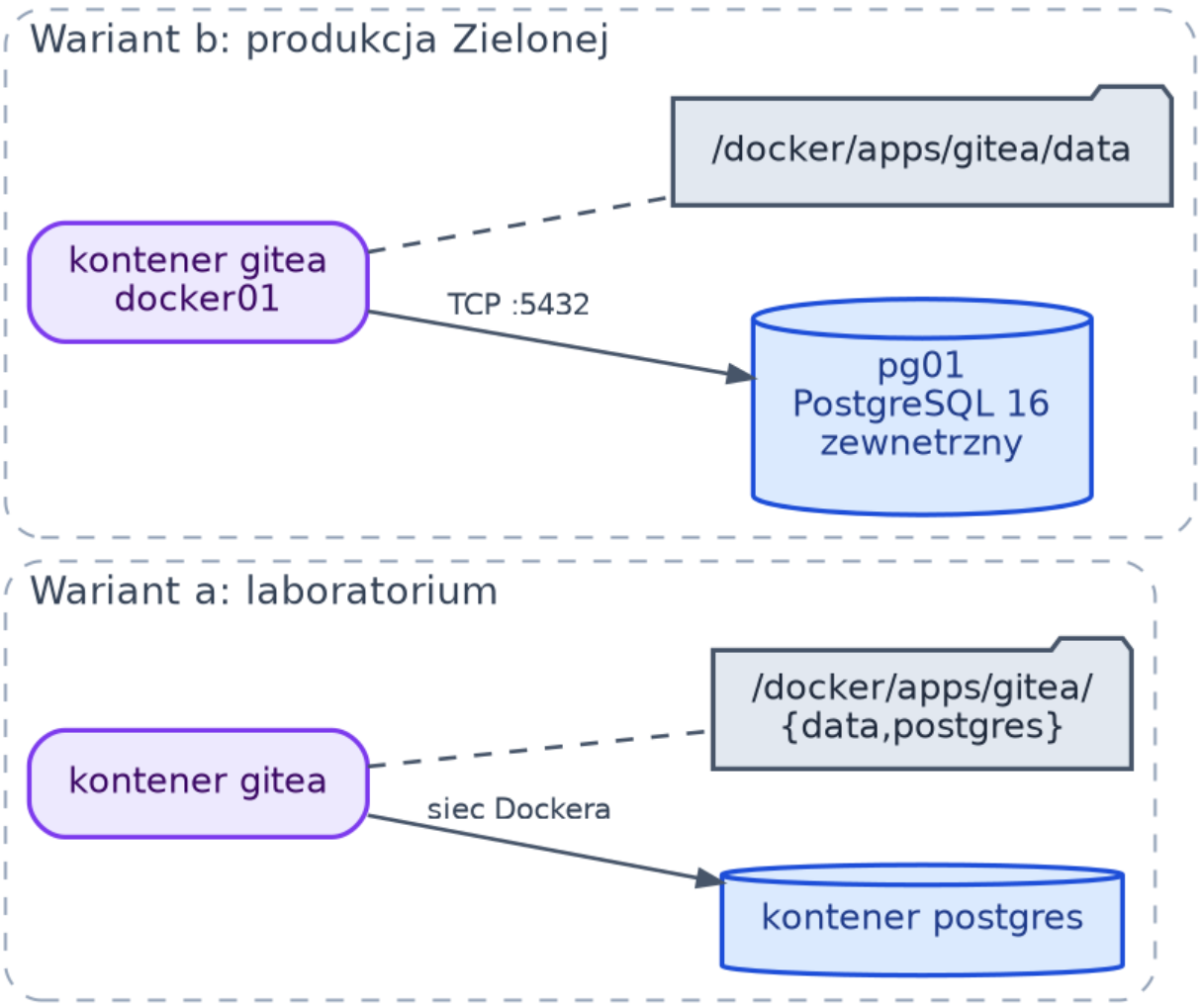
Zanim zaczniemy

Komendy wykonujemy z CLI bash jako user z uprawnieniami sudo albo root, na maszynie z zainstalowanym Dockerem i wtyczką Compose. Poniżej dwa warianty: laboratoryjny, z własnym kontenerem PostgreSQL obok Gitea, oraz produkcyjny Zielonej, z bazą na osobnym serwerze `pg01`. Wybieramy jeden, zależnie od tego, czy mamy już gdzieś działającą bazę PostgreSQL do współdzielenia.

> W naszym środowisku dane aplikacji trzymamy w `/docker/apps/<nazwa>`, a plik Compose w `/docker/compose/<nazwa>/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`
Wolne porty `3000` i `2222` na hoście	port `2222`, nie `22`, patrz niżej
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 3 wariantu b

Uważaj: obraz zawsze przypinamy do konkretnej wersji (`1.27.3`), nigdy do `latest`. Aktualizacja to świadoma zmiana numeru w `compose.yaml`, nie przypadkowe pobranie nowszej wersji przy kolejnym `docker compose pull`.



Dwa warianty instalacji Gitea w Dockerze: laboratorium z własnym kontenerem PostgreSQL i produkcja Zielonej z bazą na pg01 (kliknięcie otwiera pełny rozmiar)

Przebieg

Wariant a: laboratorium, Gitea z własnym kontenerem PostgreSQL

Zakładamy katalogi na dane, zgodnie z konwencją ścieżek:

```
sudo mkdir -p /docker/apps/gitea/data /docker/apps/gitea/postgres
sudo mkdir -p /docker/compose/gitea
```

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

```
GITEA_DB_PASSWORD=HasloDoBazy
```

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

```
services:
  server:
    image: docker.gitea.com/gitea:1.27.3
    container_name: gitea
    restart: always
    environment:
      - USER_UID=1000
      - USER_GID=1000
      - GITEA__database__DB_TYPE=postgres
      - GITEA__database__HOST=db:5432
      - GITEA__database__NAME=gitea
      - GITEA__database__USER=gitea
      - GITEA__database__PASSWD=${GITEA_DB_PASSWORD}
    volumes:
      - /docker/apps/gitea/data:/data
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    ports:
      - "3000:3000"
      - "2222:22"
    depends_on:
      - db

  db:
    image: postgres:16
```

```
    container_name: gitea-postgres
    restart: always
    environment:
      - POSTGRES_USER=gitea
      - POSTGRES_PASSWORD=${GITEA_DB_PASSWORD}
      - POSTGRES_DB=gitea
    volumes:
      - /docker/apps/gitea/postgres:/var/lib/postgresql/data
```

Pierwsze uruchomienie:

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

Wariant b: produkcja Zielonej, baza na zewnętrznym pg01

Katalog na dane potrzebny jest tylko dla samej Gitea, bez podkatalogu na bazę:

```
sudo mkdir -p /docker/apps/gitea/data
sudo mkdir -p /docker/compose/gitea
```

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

Plik `/docker/compose/gitea/.env`:

```
GITEA_DB_PASSWORD=HasloDoBazy
```

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

```
services:
  server:
    image: docker.gitea.com/gitea:1.27.3
    container_name: gitea
    restart: always
    environment:
      - USER_UID=1000
      - USER_GID=1000
      - GITEA__database__DB_TYPE=postgres
      - GITEA__database__HOST=pg01:5432
      - GITEA__database__NAME=gitea
      - GITEA__database__USER=gitea
      - GITEA__database__PASSWD=${GITEA_DB_PASSWORD}
    volumes:
      - /docker/apps/gitea/data:/data
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    ports:
      - "3000:3000"
      - "2222:22"
```

Wartości zmiennych środowiskowych w postaci `GITEA__sekcja__KLUCZ` odpowiadają dokładnie sekcjom i kluczom pliku `app.ini`, opisanego na stronie Pierwsze uruchomienie: kreator, administrator, reverse proxy: `GITEA__database__HOST` nadpisuje `HOST` w sekcji `[database]`, i tak dalej. Pierwsze uruchomienie:

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

Dlaczego port 2222, nie 22

Kontener nasłuchuje wewnątrz na standardowym porcie SSH `22`, ale na hoście mapujemy go na `2222`. Port `22` hosta jest zwykle już zajęty przez systemowy `sshd`, którym administrujemy samą maszyną `docker01`, niezależnie od Gitea. Rozdzielenie portów pozwala też ograniczyć na zaporze dostęp do administracyjnego SSH hosta osobno od SSH używanego przez programistów do operacji `git clone`/`push`/`pull`.

Logi pierwszego startu

Sprawdzamy, czy kontener wstał poprawnie i połączył się z bazą:

```
docker compose logs -f server
```

Przykład:

```
gitea  | 2026/09/27 10:02:14 ...s/setting/setting.go:320:NewContext() [I] Custom config:
/data/gitea/conf/app.ini
gitea  | 2026/09/27 10:02:15 ...ute/routes/common/db.go:32 [I] ORM engine initialization
successful!
gitea  | 2026/09/27 10:02:15 ...ute/setting/git.go:213:newGit() [I] Git Version: 2.45.0
gitea  | 2026/09/27 10:02:16 ...able/server/main.go:120 [I] Listen: http://0.0.0.0:3000
gitea  | 2026/09/27 10:02:16 ...ute/graceful/server.go:75:NewServer() [I] Starting new web
server: tcp:0.0.0.0:3000 on PID: 1
```

Weryfikacja

Sprawdzamy, że kontenery działają:

```
docker compose ps
```

Przykład:

```
NAME              IMAGE                              STATUS
gitea             docker.gitea.com/gitea:1.27.3      Up 2 minutes
gitea-postgres    postgres:16                        Up 2 minutes
```

Sprawdzamy, że interfejs WWW odpowiada, zanim otworzymy go w przeglądarce:

```
curl -I http://localhost:3000
```

Przykład:

```
HTTP/1.1 200 OK
Set-Cookie: i_like_gitea=...
Set-Cookie: _csrf=...
```

🖼 ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: pierwszy ekran otwarty pod `http://docker01:3000`, czyli kreator instalacji Gitea („Install Gitea") z widoczną sekcją „Database Settings" i typem bazy „PostgreSQL".

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/gitea
```

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
Kontener `server` restartuje się w pętli	baza jeszcze nie wystartowała (wariant a) albo `pg01` nieosiągalny (wariant b)	`docker compose logs db` albo sprawdzamy `telnet pg01 5432` z hosta `docker01`
`curl -I http://localhost:3000` zwraca `Connection refused`	kontener jeszcze się uruchamia albo port zajęty przez inną usługę	`docker compose logs server`, `ss -tlnp`
Błąd `password authentication failed for user gitea`	hasło w `.env` inne niż w bazie na `pg01`	poprawiamy hasło w `.env` albo w bazie przez `ALTER USER gitea WITH PASSWORD`
Push po SSH nie działa, mimo że WWW działa	firewall blokuje port `2222` albo klient łączy się na `22`	sprawdzamy `firewall-cmd --list-ports` i adres w konfiguracji klienta Git

Powiązane: Instalacja Gitea bez Dockera · Pierwsze uruchomienie: kreator, administrator, reverse proxy · Konwencja Dockera w Zielonej

Źródła: docs.gitea.com, rozdział „Installation with Docker" · docs.gitea.com, rozdział „Config Cheat Sheet" (zmienne `GITEA__`)

Instalacja Gitea bez Dockera

Typ: procedura **Dotyczy: Gitea 1.27.3, binarka linux amd64, Rocky Linux 9**

Czas: ~25 min

Efekt: Gitea uruchomiona jako usługa systemd na dedykowanym koncie systemowym, bez Dockera, z firewalld i SELinux dopasowanymi do nowego portu.

Skróty użyte na tej stronie (4)

Skrót	Co znaczy
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
DNS	Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP
SELinux	Security-Enhanced Linux: dodatkowa warstwa uprawnień w jądrze RHEL
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. Serwer musi rozwiązywać nazwy DNS i mieć ruch wychodzący do `dl.gitea.com`, żeby pobrać binarkę.

Wymaganie	Uwaga
Rocky Linux 9 albo RHEL 9	polecenia SELinux i firewalld poniżej dotyczą tej rodziny systemów
Ruch wychodzący do `dl.gitea.com`	pobranie binarki i sumy kontrolnej
Wolny port `3000` (WWW) i `22` albo osobny port SSH	bez Dockera nie ma potrzeby przesuwać portu SSH, o ile Gitea nie dzieli maszyny z inną usługą na `22`
Baza danych osiągalna z tego serwera	PostgreSQL na `pg01` albo wbudowany SQLite na start w laboratorium

Kiedy wybieramy ten wariant zamiast Dockera:

Sytuacja	Docker	Bez Dockera
Mamy już standard Dockera w firmie	tak, spójnie z resztą aplikacji	dodatkowy wyjątek do utrzymania

Sytuacja	Docker	Bez Dockera
Serwer ma niewiele zasobów albo politykę „bez kontenerów"	kontener to dodatkowy narzut	mniejszy narzut, jeden proces
Aktualizacja ma być podmianą jednego pliku, bez rejestru obrazów	trzeba śledzić tagi obrazu	podmiana binarki, prościej do audytu
Zespół zna dobrze systemd, SELinux i firewalld	mniej istotne, izolacja z kontenera	pełna kontrola, ale więcej ręcznej pracy

Przebieg

1. Utworzenie konta systemowego git

```
sudo useradd --system --shell /bin/bash --comment "Git Version Control" --create-home --home-dir /home/git git
```

Konto `git` uruchamia usługę i jest właścicielem repozytoriów na dysku, bez prawa logowania hasłem.

2. Pobranie binarki z weryfikacją sumy kontrolnej

```
curl -L -o gitea https://dl.gitea.com/gitea/1.27.3/gitea-1.27.3-linux-amd64
```

Pobieramy też sumę kontrolną publikowaną obok binarki:

```
curl -L -o gitea.sha256 https://dl.gitea.com/gitea/1.27.3/gitea-1.27.3-linux-amd64.sha256
```

Weryfikujemy zgodność, zanim nadamy plikowi prawo wykonywania:

```
sha256sum -c gitea.sha256 --ignore-missing
```

Przykład:

```
gitea-1.27.3-linux-amd64: OK
```

Uważaj: jeśli suma się nie zgadza, nie uruchamiamy pobranego pliku. Oznacza to uszkodzone pobieranie albo, w gorszym wypadku, podmienioną binarkę.

3. Instalacja binarki

```
chmod +x gitea
sudo mv gitea /usr/local/bin/gitea
```

4. Utworzenie katalogów danych i konfiguracji

```
sudo mkdir -p /var/lib/gitea/custom /var/lib/gitea/data /var/lib/gitea/log
sudo chown -R git:git /var/lib/gitea
sudo chmod -R 750 /var/lib/gitea
```

Katalog na `app.ini`, czytelny dla konta `git`, ale nie dla pozostałych użytkowników systemu:

```
sudo mkdir -p /etc/gitea
sudo chown root:git /etc/gitea
sudo chmod 770 /etc/gitea
```

5. Jednostka systemd

Plik `/etc/systemd/system/gitea.service`:

```
[Unit]
Description=Gitea
After=network.target

[Service]
RestartSec=2s
Type=simple
User=git
Group=git
WorkingDirectory=/var/lib/gitea/
ExecStart=/usr/local/bin/gitea web --config /etc/gitea/app.ini
Restart=always
Environment=USER=git HOME=/home/git GITEA_WORK_DIR=/var/lib/gitea
```

```
[Install]
WantedBy=multi-user.target
```

Włączamy i uruchamiamy usługę:

```
sudo systemctl daemon-reload
sudo systemctl enable --now gitea
```

Sprawdzamy status:

```
sudo systemctl status gitea
```

Przykład:

```
● gitea.service - Gitea
     Loaded: loaded (/etc/systemd/system/gitea.service; enabled; preset: disabled)
     Active: active (running) since Sun 2026-09-27 10:14:02 UTC; 4s ago
   Main PID: 20211 (gitea)
      Tasks: 7
     Memory: 68.4M
```

6. SELinux i firewalld na Rocky 9

Domyślna polityka SELinux nie blokuje binarki uruchomionej z `/usr/local/bin`, ale warto sprawdzić kontekst pliku po przeniesieniu:

```
ls -Z /usr/local/bin/gitea
```

Jeśli kontekst różni się od oczekiwanego typu wykonywalnego, przywracamy domyślny:

```
sudo restorecon -v /usr/local/bin/gitea
```

Otwieramy port WWW na firewalld:

```
sudo firewall-cmd --add-port=3000/tcp --permanent
sudo firewall-cmd --reload
```

Weryfikujemy otwarte porty:

```
sudo firewall-cmd --list-ports
```

Przykład:

```
3000/tcp
```

Weryfikacja

```
curl -I http://localhost:3000
```

Przykład:

```
HTTP/1.1 200 OK
Set-Cookie: i_like_gitea=...
```

Sprawdzamy logi usługi, gdyby coś odbiegało od oczekiwań:

```
sudo journalctl -u gitea -n 50
```

Wycofanie zmian

```
sudo systemctl disable --now gitea
sudo rm -f /etc/systemd/system/gitea.service
sudo systemctl daemon-reload
sudo rm -rf /var/lib/gitea /etc/gitea /usr/local/bin/gitea
sudo userdel -r git
```

Gdy coś nie zadziała

Objaw	Przyczyna	Naprawa
`systemctl status gitea` pokazuje `failed` zaraz po starcie	zły właściciel `/var/lib/gitea` albo brak pliku `app.ini` przy wskazanej ścieżce	`journalctl -u gitea`, poprawiamy `chown` i ścieżkę w `ExecStart`
`curl -I http://localhost:3000` zwraca `Connection refused`, mimo że usługa działa	firewalld blokuje port	`firewall-cmd --list-ports`, dodajemy `3000/tcp`
Usługa działa lokalnie, ale nie widać jej z innego hosta	port otwarty tylko na `localhost` w `app.ini` (`HTTP_ADDR`)	ustawiamy `HTTP_ADDR = 0.0.0.0` w sekcji `[server]`

Objaw	Przyczyna	Naprawa
`sha256sum -c` zgłasza `FAILED`	uszkodzone pobieranie albo zmieniona wersja pliku na serwerze	pobieramy ponownie, porównujemy z sumą podaną na stronie wydania

Powiązane: Instalacja Gitea w Dockerze · Pierwsze uruchomienie: kreator, administrator, reverse proxy

Źródła: docs.gitea.com, rozdział „Installation from binary" · `man 5 systemd.service` · `man 8 firewall-cmd`

Pierwsze uruchomienie: kreator, administrator, reverse proxy

Poziom: podstawy **Dotyczy: Gitea 1.27** **Czas: ~25 min**

Po pierwszym starcie kontenera albo usługi Gitea czeka kreator konfiguracji w przeglądarce. Ta strona prowadzi przez kreator, przez założenie pierwszego konta administratora i przez postawienie Nginksa przed Giteą, żeby nie wystawiać portu 3000 na świat wprost.

Skróty użyte na tej stronie (4)

Skrót	Co znaczy
HTTP	Hypertext Transfer Protocol: protokół stron WWW
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera
TLS	Transport Layer Security: szyfrowanie połączeń, np. w HTTPS
URL	Uniform Resource Locator: adres strony lub zasobu

Fiszka

TYLKO ODCZYT

Otwarcie kreatora instalacji: adres w przeglądarce

```
http://docker01:3000
```

Przy pierwszym uruchomieniu, dopóki `INSTALL_LOCK` nie jest ustawiony na `true`, Gitea pokazuje zamiast strony logowania kreator instalacji.

□□ ZMIENIA SYSTEM

Nadanie pierwszych uprawnień administratora z CLI: gitea admin user create

```
docker exec -u git gitea gitea admin user create --username ola.wisniewska --password HasloAdministratora --email ola.wisniewska@zielona.example --admin
```

Alternatywa dla konta administratora zakładanego w samym kreatorze, przydatna przy automatyzacji albo gdy kreator zdążył już zablokować się bez utworzenia konta.

□□ **CLI** = Command Line Interface: praca przez wpisywanie komend w terminalu · **docker** = narzędzie do kontenerów

□□ TYLKO ODCZYT

Weryfikacja odpowiedzi Nginksa po podłączeniu Gitea: curl -I

```
curl -I https://git.zielona.example
```

Sprawdzamy, że reverse proxy w ogóle przekazuje ruch do kontenera, zanim zaczniemy szukać błędu po stronie samej Gitea.

Przykład:

```
HTTP/2 200
server: nginx
content-type: text/html; charset=UTF-8
set-cookie: i_like_gitea=...
```

□□ **curl** = client URL: pobieranie i wysyłanie przez HTTP z terminala

Z życia wzięte

□□ Z ŻYCIA WZIĘTE

Tomek, drugi tydzień pracy, dostaje zadanie postawienia Gitea w laboratorium. Przechodzi kreator, klika przez wszystkie pola bez czytania i zostawia `ROOT_URL` z domyślnym adresem `http://localhost:3000`. Dzień później powiadomienia e-mail o nowych pull requestach

> przychodzą z linkami wskazującymi na `localhost`, bezużytecznymi dla nikogo poza samym Tomkiem.

Kreator instalacji pole po polu

> **Pojęcie: kreator instalacji.** Formularz w przeglądarce, przez który Gitea przy pierwszym uruchomieniu zbiera minimalny zestaw ustawień i zapisuje go od razu do pliku `app.ini`. Po zatwierdzeniu kreator sam ustawia `INSTALL_LOCK = true`, więc powtórne wejście na ten adres pokazuje już zwykły ekran logowania.

Pole kreatora	Co wpisujemy w Zielonej	Dlaczego
Database Type	PostgreSQL	zgodnie z bazą przygotowaną na `pg01` albo w kontenerze `db`
Host	`pg01:5432` (produkcja) albo `db:5432` (laboratorium)	adres bazy z kroku instalacji
Username / Password	`gitea` / hasło z `.env`	konto założone wcześniej w PostgreSQL
Database Name	`gitea`	nazwa bazy założonej wcześniej
Site Title	`Gitea Zielona`	nazwa widoczna w interfejsie i w tytule karty przeglądarki
Repository Root Path	pozostawiamy domyślne `/data/git/repositories`	ścieżka wewnątrz kontenera, mapowana przez wolumen na `/docker/apps/gitea/data`
SSH Server Domain	`git.zielona.example`	adres, pod jakim Gitea podaje polecenia klonowania przez SSH
SSH Server Port	`2222`	port zmapowany na hoście, patrz Instalacja Gitea w Dockerze
Gitea Base URL (ROOT_URL)	`https://git.zielona.example/`	rzeczywisty adres, pod jakim użytkownicy otwierają Giteę, nie adres wewnętrzny kontenera
Optional Settings → Admin Account	konto Oli jako pierwszego administratora	pierwsza osoba z pełnymi uprawnieniami w systemie

Uważaj: pole `Gitea Base URL` odpowiada kluczowi `ROOT_URL` w `app.ini`. Jeśli zostanie tam adres wewnętrzny albo `localhost`, wszystkie linki generowane przez Giteę, w mailach, w webhookach, w podpowiedziach klonowania, wskażą na adres bezużyteczny dla kogokolwiek poza tą jedną maszyną.

ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran kreatora instalacji Gitea, sekcja „General Settings", z widocznymi polami „Site Title" ustawionym na „Gitea Zielona" i „Gitea Base URL" ustawionym na `https://git.zielona.example/`.

Pierwsze konto administratora

Konto założone w sekcji „Admin Account Settings" kreatora dostaje od razu najwyższy poziom uprawnień w całej instalacji: zarządzanie użytkownikami, organizacjami, źródłami uwierzytelniania i globalną konfiguracją widoczną z poziomu interfejsu. W Zielonej to konto Oli, jako prowadzącej zespół programistów.

ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran „Site Administration” po zalogowaniu kontem Oli, zakładka „Dashboard”, z widoczną wersją Gitea i liczbą zarejestrowanych użytkowników.

app.ini: najważniejsze sekcje

> **Pojęcie: app.ini.** Główny plik konfiguracyjny Gitea. Kreator zapisuje go przy pierwszym uruchomieniu, a dalsze zmiany, których kreator nie obejmuje, wprowadzamy edytując go ręcznie i restartując usługę, albo nadpisując wybrane klucze zmiennymi środowiskowymi `GITEA__sekcja__KLUCZ`.

Sekcja [server]. `ROOT_URL` to adres, pod jakim instancja jest widoczna z zewnątrz. `DOMAIN` to sama nazwa hosta, bez protokołu i ścieżki, używana między innymi przy generowaniu ciasteczek. `SSH_DOMAIN` i `SSH_PORT` to adres i port, jakie Gitea podaje w podpowiedziach `git clone` przez SSH, niezależnie od tego, pod jakim portem SSH faktycznie nasłuchuje kontener wewnątrz.

```
[server]
DOMAIN     = git.zielona.example
ROOT_URL   = https://git.zielona.example/
SSH_DOMAIN = git.zielona.example
SSH_PORT   = 2222
```

Sekcja [service]. `DISABLE_REGISTRATION` ustawione na `true` wyłącza samodzielną rejestrację kont z poziomu strony logowania: w Zielonej konta zakłada się przez import z Active Directory albo ręcznie przez administratora, nie przez otwarty formularz.

```
[service]
DISABLE_REGISTRATION = true
```

Sekcja [security]. `INSTALL_LOCK` ustawiony na `true` blokuje ponowne uruchomienie kreatora. Bez tego ustawienia dowolna osoba ze świeżym dostępem do adresu instalacji mogłaby przejść przez kreator ponownie i nadpisać konfigurację produkcyjną.

```
[security]
INSTALL_LOCK = true
```

Sekcja [mailer]. Włącza wysyłkę powiadomień e-mail, na przykład o nowych pull requestach albo wzmiankach w komentarzu. Bez tej sekcji Gitea działa normalnie, tylko bez powiadomień.

```
[mailer]
ENABLED  = true
SMTP_ADDR = smtp.zielona.example
SMTP_PORT = 587
FROM      = gitea@zielona.example
USER      = gitea@zielona.example
PASSWD    = HasloSkrzynki
```

Reverse proxy przed Giteą

Gitea sama w sobie serwuje ruch HTTP na porcie 3000, bez obsługi TLS. W Zielonej certyfikat kończy Nginx stojący na tym samym hoście `docker01`, opisany szerzej na stronie Reverse proxy i równoważenie obciążenia: tutaj tylko konkretny blok gotowy do skopiowania.

```
server {
    listen 443 ssl;
    server_name git.zielona.example;

    ssl_certificate     /etc/nginx/ssl/git.zielona.example.crt;
    ssl_certificate_key /etc/nginx/ssl/git.zielona.example.key;

    client_max_body_size 512m;
```

```
    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

server {
    listen 80;
    server_name git.zielona.example;
    return 301 https://$host$request_uri;
}
```

`client_max_body_size` podnosimy wyraźnie ponad domyślny 1MB: repozytoria z dużymi plikami binarnymi albo push z wieloma commitami naraz łatwo przekraczają domyślny limit, kończąc się błędem 413 bez żadnego komunikatu ze strony samej Gitea.

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: strona logowania Gitea otwarta pod adresem `https://git.zielona.example`, z widoczną kłódką certyfikatu w pasku adresu przeglądarki.

Częste błędy

Objaw	Przyczyna	Co robimy
Linki w mailach i webhookach wskazują na zły adres	`ROOT_URL` niezgodny z rzeczywistym adresem	poprawiamy `ROOT_URL` w `app.ini`, restartujemy kontener
Push przez SSH podaje w podpowiedzi zły port	`SSH_DOMAIN` albo `SSH_PORT` niezgodne z rzeczywistym mapowaniem portów	poprawiamy sekcję `[server]` na wartości pasujące do `docker compose.yaml`
Duży push kończy się błędem 413	brak albo zbyt niski `client_max_body_size` w Nginksie	podnosimy limit w bloku `server` przed Giteą
Kreator pokazuje się ponownie po restarcie	`INSTALL_LOCK` nie zapisał się (błąd zapisu do `app.ini`)	sprawdzamy uprawnienia do pliku `app.ini` dla konta, na którym działa Gitea

□□ Ćwiczenia

1. Wyjaśniamy, dlaczego pole `Gitea Base URL` w kreatorze ma znaczenie nawet wtedy, gdy Gitea jest dostępna wyłącznie z sieci wewnętrznej firmy.

Odpowiedź

To pole trafia do `ROOT_URL` i jest używane przy generowaniu każdego linku bezwzględnego: w mailach z powiadomieniami, w linkach webhooków, w podpowiedziach klonowania. Nawet w sieci wewnętrznej adres musi być tym, pod którym rzeczywiście łączą się użytkownicy, inaczej linki prowadzą donikąd.

1. Push dużego repozytorium kończy się błędem 413 mimo poprawnej konfiguracji samej Gitea. Wskazujemy, gdzie szukamy przyczyny i jak ją naprawiamy.

Odpowiedź

Błąd 413 pochodzi zwykle od reverse proxy, nie od Gitea: sprawdzamy `client_max_body_size` w konfiguracji Nginksa przed Giteą i podnosimy limit odpowiednio do rozmiaru repozytoriów.

Powiązane: Instalacja Gitea w Dockerze · Reverse proxy i równoważenie obciążenia · Logowanie kontem z Active Directory

Źródła: docs.gitea.com, rozdział „Config Cheat Sheet" · docs.gitea.com, rozdział „Initialization"

Logowanie kontem z Active Directory

Typ: procedura **Dotyczy: Gitea 1.27, Active Directory na dc01/dc02** **Czas: ~35 min**

Efekt: logowanie do Gitea kontem domenowym z Active Directory, z automatycznym nadaniem uprawnień administratora osobom z odpowiedniej grupy, bez zakładania kont lokalnych dla każdej nowej osoby.

Skróty użyte na tej stronie (7)

Skrót	Co znaczy
AD	Active Directory: katalog Microsoftu z kontami użytkowników, komputerów i grup w firmie
CA	Certificate Authority: urząd certyfikacji, który podpisuje certyfikaty
CLI	Command Line Interface: praca przez wpisywanie komend w terminalu
DN	Distinguished Name: pełny, jednoznaczny adres jednego obiektu w katalogu LDAP, na przykład konta albo grupy
LDAP	Lightweight Directory Access Protocol: protokół do odpytywania katalogu użytkowników (AD, FreeIPA)
LDAPS	LDAP szyfrowany TLS, zwykle port 636
OU	Organizational Unit: folder w Active Directory grupujący konta i komputery

Zanim zaczniemy

Z ŻYCIA WZIĘTE

Tomek dostaje dostęp do Gitea drugiego dnia pracy. Zamiast czekać, aż ktoś założy mu konto lokalne i wyśle hasło mailem, loguje się od razu tym samym kontem domenowym, którym loguje się do komputera i do poczty.

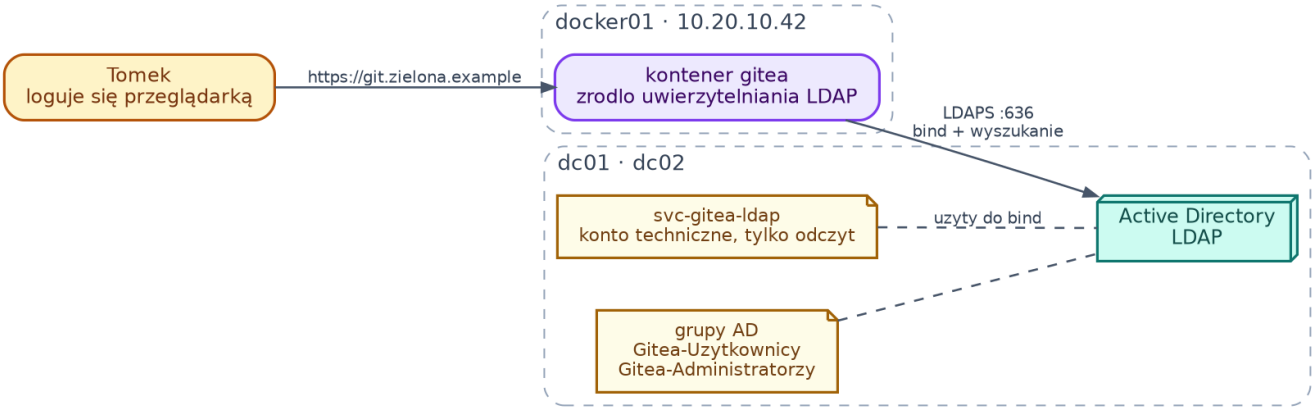
> **Pojęcie: LDAP.** Protokół odpytywania katalogu z kontami, grupami i innymi obiektami organizacyjnymi. Active Directory na `dc01` i `dc02` udostępnia swoje dane właśnie przez LDAP : Gitea, zamiast trzymać osobną bazę haseł, o każdym logowaniu pyta katalog: czy takie konto istnieje, czy podane hasło jest poprawne, i do jakich grup to konto należy.

> **Pojęcie: źródło uwierzytelniania LDAP.** Konfiguracja w Gitea opisująca, jak i gdzie łączyć się z katalogiem: adres serwera, poświadczenia konta technicznego do wyszukiwania, oraz filtry określające, które konta wolno logować i które automatycznie stają się administratorami.

Komendy CLI wykonujemy przez `docker exec` na koncie systemowym `git` wewnątrz kontenera, zgodnie z instalacją opisaną na stronie Instalacja Gitea w Dockerze. Konfigurację przez interfejs wykonujemy zalogowani kontem administratora Gitea.

Wymaganie	Uwaga
Konto techniczne w AD do bind LDAP	zakładamy w kroku 1, prawo tylko do odczytu katalogu
Dwie grupy w AD: użytkownicy i administratorzy Gitea	zakładamy w kroku 1
LDAPS na porcie 636 osiągalny z `docker01` do `dc01`/`dc02`	sprawdzamy zaporą, zanim zaczniemy konfigurację w Gitea
Certyfikat firmowego CA zaufany przez kontener Gitea	bez niego połączenie LDAPS kończy się błędem certyfikatu

Uważaj: konto techniczne wskazane przy bind LDAP powinno mieć wyłącznie prawo odczytu katalogu, na tej samej zasadzie najmniejszych uprawnień, na jakiej opieramy konta monitorujące bazy danych. Gitea nie potrzebuje żadnego prawa zapisu w Active Directory do samego uwierzytelniania użytkowników.



Gitea łączy się z Active Directory na dc01 po LDAPS, przez konto techniczne i dwie grupy uprawnień (kliknięcie otwiera pełny rozmiar)

Przebieg

1. Konto techniczne i grupy w Active Directory

Zakładamy konto techniczne do bind LDAP, z hasłem nieulegającym wygaśnięciu i bez prawa jego zmiany przez samo konto:

```
New-ADUser -Name "svc-gitea-ldap" `
  -SamAccountName "svc-gitea-ldap" `
  -UserPrincipalName "svc-gitea-ldap@ad.zielona.example" `
  -Path "OU=KontaTechniczne,DC=ad,DC=zielona,DC=example" `
  -AccountPassword (Read-Host -AsSecureString "Haslo konta") `
  -Enabled $true `
  -PasswordNeverExpires $true `
  -CannotChangePassword $true
```

Zakładamy dwie grupy: jedną dla wszystkich osób z dostępem do Gitea, drugą dla administratorów:

```
New-ADGroup -Name "Gitea-Uzytkownicy" -GroupScope Global -Path
"OU=Grupy,DC=ad,DC=zielona,DC=example"
New-ADGroup -Name "Gitea-Administratorzy" -GroupScope Global -Path
"OU=Grupy,DC=ad,DC=zielona,DC=example"
```

Dodajemy Olę do grupy administratorów:

```
Add-ADGroupMember -Identity "Gitea-Administratorzy" -Members "ola.wisniewska"
```

2. Konfiguracja przez interfejs

W Site Administration → Authentication Sources → Add Authentication Source wypełniamy formularz zgodnie z tabelą.

Pole	Wartość w Zielonej	Uwaga

Authentication Name	`AD Zielona`	dowolna, widoczna tylko w panelu administracyjnym
Authentication Type	LDAP (via BindDN)	odróżniamy od uproszczonego trybu simple bind
Security Protocol	LDAPS	port 636, ruch szyfrowany
Host	`dc01.ad.zielona.example`	drugi kontroler `dc02` jako zapasowy wpis, jeśli formularz na to pozwala
Port	`636`	standardowy port LDAPS
Bind DN	`CN=svc-gitea-ldap,OU=KontaTechniczne,DC=ad,DC=zielona,DC=example`	konto techniczne z kroku 1
Bind Password	hasło konta technicznego	to samo hasło co przy `New-ADUser`
User Search Base	`OU=Uzytkownicy,DC=ad,DC=zielona,DC=example`	jednostka organizacyjna z kontami pracowników
User Filter	`(&(objectClass=user)(sAMAccountName=%s)(memberOf:1.2.840.113556.1.4.1941:=CN=Gitea-Uzytkownicy,OU=Grupy,DC=ad,DC=zielona,DC=example))`	wpuszcza tylko członków grupy, także przez grupy zagnieżdżone
Admin Filter	`(memberOf:1.2.840.113556.1.4.1941:=CN=Gitea-Administratorzy,OU=Grupy,DC=ad,DC=zielona,DC=example)`	konto pasujące dostaje uprawnienia administratora automatycznie
Username Attribute	`sAMAccountName`	login w starym formacie, ten sam co do stacji Windows
First/Surname Attribute	`givenName` / `sn`	imię i nazwisko wyświetlane w interfejsie
E-mail Attribute	`mail`	adres do powiadomień
Enable User Synchronization	zaznaczone	odświeża listę użytkowników cyklicznie, patrz krok 4

Pojęcie: LDAP_MATCHING_RULE_IN_CHAIN. Specjalna reguła dopasowania LDAP o identyfikatorze `1.2.840.113556.1.4.1941`, dodawana po dwukropku w filtrze `memberOf:1.2.840.113556.1.4.1941:=`. Sprawdza przynależność do grupy również wtedy, gdy konto należy do niej pośrednio, przez inną grupę zagnieżdżoną w docelowej, zamiast tylko przez bezpośrednie członkostwo.

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran Site Administration → Authentication Sources po dodaniu źródła LDAP, widoczny wiersz „AD Zielona" ze statusem Aktywne i typem LDAP (via BindDN).

3. Konfiguracja przez CLI: gitea admin auth add-ldap

Ten sam efekt co krok 2, wykonany jedną komendą wewnątrz kontenera, przydatny przy automatyzacji wdrożenia:

```
docker exec -u git gitea gitea admin auth add-ldap \
  --name "AD Zielona" \
  --security-protocol ldaps \
  --host dc01.ad.zielona.example \
  --port 636 \
  --bind-dn "CN=svc-gitea-ldap,OU=KontaTechniczne,DC=ad,DC=zielona,DC=example" \
  --bind-password 'HasloKontaTechnicznego' \
  --user-search-base "OU=Uzytkownicy,DC=ad,DC=zielona,DC=example" \
  --username-attribute sAMAccountName \
  --firstname-attribute givenName \
  --surname-attribute sn \
  --email-attribute mail \
  --user-filter
"(&(objectClass=user)(sAMAccountName=%s)(memberOf:1.2.840.113556.1.4.1941:=CN=Gitea-
Uzytkownicy,OU=Grupy,DC=ad,DC=zielona,DC=example))" \
  --admin-filter "(memberOf:1.2.840.113556.1.4.1941:=CN=Gitea-
Administratorzy,OU=Grupy,DC=ad,DC=zielona,DC=example)" \
  --synchronize-users
```

Odmiana `add-ldap-simple`, zamiast `add-ldap`, łączy się prostszym trybem simple bind: konto loguje się bezpośrednio swoim DN i własnym hasłem, bez osobnego konta technicznego do wyszukiwania. Ma to sens w mniejszych katalogach LDAP bez struktury OU dopasowanej do wyszukiwania po atrybucie, ale w Active Directory Zielonej zostajemy przy pełnym trybie bind DN z tabeli wyżej.

4. Zaufanie do firmowego CA

Kontener Gitea musi ufać certyfikatowi, którym `dc01` i `dc02` podpisują LDAPS, inaczej połączenie kończy się błędem certyfikatu, mimo poprawnej reszty konfiguracji. Certyfikat CA kopiujemy do kontenera i dokładamy do systemowego magazynu zaufania:

```
docker cp CA-Zielona.crt gitea:/usr/local/share/ca-certificates/ca-zielona.crt
docker exec -u root gitea update-ca-certificates
docker restart gitea
```

> Po przenosinach urzędu według stronyPrzenosiny urzędu z dc01 bez przerwy: dwa urzędy równolegle i wygaszenie CA-Zielona dokładamy tu w ten sam sposób zaufanie do `Zielona-Root-CA`, a `CA-Zielona.crt` zostaje obok do wygaszenia starego urzędu.

Weryfikacja

Testujemy logowanie kontem domenowym na ekranie logowania Gitea, używając loginu `tomek.nowak` i hasła domenowego, zamiast konta lokalnego.

Sprawdzamy listę źródeł uwierzytelniania i ich stan z poziomu CLI:

```
docker exec -u git gitea gitea admin auth list
```

Przykład:

```
ID  Name        Type  Enabled
1   AD Zielona   LDAP  true
```

Synchronizacja użytkowników

> **Pojęcie: synchronizacja użytkowników.** Cykliczne zadanie porównujące listę kont w Gitea z listą kont pasujących do filtra w Active Directory: nowe konta w grupie `Gitea-Uzytkownicy` pojawiają się w Gitea automatycznie, a konta usunięte z grupy albo wyłączone w AD tracą dostęp bez ręcznej interwencji.

Wymuszamy pojedynczą synchronizację od razu, zamiast czekać na najbliższy cykl:

```
docker exec -u git gitea gitea admin auth update-ldap --id 1 --synchronize-users
```

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran Site Administration → Users po pierwszej synchronizacji, z widocznymi kontami zaimportowanymi z Active Directory i kolumną „Source" ustawioną na „AD Zielona".

Wycofanie zmian

Wyłączenie źródła LDAP bez usuwania importowanych kont, gdy chcemy tymczasowo wrócić do logowania lokalnego:

```
docker exec -u git gitea gitea admin auth update-ldap --id 1 --disable
```

Pełne usunięcie źródła:

```
docker exec -u git gitea gitea admin auth delete --id 1
```

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy
`Failed to connect: LDAP Result Code 49 Invalid Credentials`	zły Bind DN albo hasło konta technicznego	sprawdzamy DN i hasło komendą `ldapsearch` z hosta `docker01` przed poprawką w Gitea
`x509: certificate signed by unknown authority`	certyfikat firmowego CA niezaufany w kontenerze	wykonujemy krok 4, dokładamy CA do magazynu i restartujemy kontener
Konto istnieje w AD, ale logowanie odrzucone	filtr użytkownika bez `%s` albo konto poza grupą wskazaną w filtrze	sprawdzamy, czy `%s` jest w filtrze i czy konto rzeczywiście należy do `Gitea-Uzytkownicy`, także pośrednio
Konto loguje się, ale nie widzi uprawnień administratora mimo członkostwa w grupie	filtr administratora bez reguły `LDAP_MATCHING_RULE_IN_CHAIN` przy grupie zagnieżdżonej	dokładamy `:1.2.840.113556.1.4.1941:` do `memberOf` w Admin Filter

Powiązane: Pierwsze uruchomienie: kreator, administrator, reverse proxy · FreeIPA · Active Directory: konta, grupy, jednostki organizacyjne

Źródła: docs.gitea.com, rozdział „LDAP Authentication" · `gitea admin auth add-ldap --help` · Microsoft Learn, „Search Filter Syntax” (LDAP_MATCHING_RULE_IN_CHAIN)