Gitea i BookStack przez MCP: bezpieczne spięcie krok po kroku

Poziom: zaawansowane **Dotyczy: Gitea, BookStack, MCP, Claude Code, Antigravity CLI**

Czas: ~50 min

Spinamy z agentem dwie usługi, które w Zielonej trzymają najwięcej wiedzy: repozytoria w Gitea i bazę wiedzy w BookStacku. Zaczynamy od kont serwisowych z najmniejszą rolą, token chowamy w magazynie poświadczeń, serwer Gitea uruchamiamy w trybie tylko do odczytu, a wersję z zapisem dokładamy dopiero osobno, z ochroną gałęzi i potwierdzeniem człowieka.

Skróty użyte na tej stronie (10)

Skrót	Po angielsku	Po polsku i co znaczy
AI	Artificial Intelligence	sztuczna inteligencja: programy, które naśladują część ludzkiego rozumowania i rozmawiają w języku naturalnym
CLI	Command Line Interface	interfejs wiersza poleceń: praca przez wpisywanie komend w terminalu
DELETE	HTTP DELETE method	metoda HTTP: usuwa zasób
GET	HTTP GET method	metoda HTTP: prosi serwer o odczyt zasobu bez zmieniania czegokolwiek
HTTP	Hypertext Transfer Protocol	protokół przesyłania hipertekstu: język, którym przeglądarki i programy rozmawiają z serwerami WWW
MCP	Model Context Protocol	protokół kontekstu modelu: otwarty standard podłączania narzędzi i danych do modeli AI
OAuth	Open Authorization	otwarta autoryzacja: standard, w którym jedna usługa dostaje ograniczony dostęp do drugiej przez token, bez poznawania hasła
PATCH	HTTP PATCH method	metoda HTTP: zmienia część zasobu
POST	HTTP POST method	metoda HTTP: wysyła dane do serwera, na przykład żeby utworzyć nowy zasób
PUT	HTTP PUT method	metoda HTTP: zastępuje zasób nową zawartością

Leksykon: nazwy i pojęcia z tej strony (26)

Nazwa	Co to jest
agent	Model z dostępem do narzędzi, który w pętli planuje, wywołuje narzędzia, czyta wyniki i decyduje o kolejnym kroku. Tym różni się od czatu, że jego działania mogą zmieniać stan systemów.
allowlist	Lista dozwolonych: wszystko, czego na niej nie ma, jest zabronione. Jest bezpieczniejsza od listy zakazanych, bo o nowych zagrożeniach nie trzeba pamiętać.
Antigravity CLI	Narzędzie Google do pracy z agentami kodującymi, w wersji uruchamianej z terminala. Konfigurację serwerów MCP trzyma w pliku w profilu użytkownika.
BookStack	Nasza baza wiedzy w przeglądarce, ułożona w półki, książki, rozdziały i strony. Ma API, przez które agent może czytać strony.
Claude	Rodzina modeli firmy Anthropic.
Claude Code	Agent kodujący firmy Anthropic działający w terminalu. Czyta CLAUDE.md, ma system uprawnień i hooki.
czat	Rozmowa z modelem, w której on tylko odpowiada tekstem. Sam niczego nie uruchamia ani nie zmienia, dopóki nie damy mu narzędzi.
FastMCP i MCPServer	Klasa z pakietu mcp do pisania serwerów MCP w Pythonie. W wersji 1 pakietu nazywa się FastMCP, w wersji 2 MCPServer.
Git	System kontroli wersji, w którym przechowujemy kod i dokumenty z historią zmian.
Gitea	Własny serwer repozytoriów Git z API. Jego serwer MCP (gitea-mcp) podłączamy do agenta w trybie tylko do odczytu.
keyring	Biblioteka Pythona, która zapisuje i odczytuje sekrety w systemowym magazynie. W Windows jest to Menedżer poświadczeń.
pip i PyPI	pip to instalator pakietów Pythona, a PyPI to publiczne składowisko tych pakietów.
PowerShell	Powłoka i język skryptowy Windows. W nim wykonujemy polecenia z naszych stron dla Windowsa.
Python	Język programowania, w którym piszemy klienty, serwery MCP i skrypty.
sekret i poświadczenie	Informacja, która otwiera dostęp: hasło, klucz API, token. Trzymamy ją w magazynie sekretów, a nie w czacie, pliku konfiguracyjnym ani repozytorium.
serwer MCP	Program, który udostępnia agentowi narzędzia i dane w jednolity sposób, według protokołu MCP. Działa lokalnie jako podproces albo zdalnie przez HTTP.
śmiertelna trójka	Nazwa Simona Willisona (czerwiec 2025) dla trzech zdolności, które razem otwierają drogę do kradzieży danych: dostępu do danych prywatnych, kontaktu z niezaufaną treścią i możliwości wysyłania danych na zewnątrz. Zabezpieczenie polega na odjęciu jednej z nich.
stdio	Transport MCP, w którym klient uruchamia serwer jako własny podproces i rozmawia z nim przez standardowe wejście i wyjście. Poświadczenia serwer dostaje ze środowiska, a nie z sieci.
test akceptacyjny	Sprawdzenie opisane z góry, które faza musi przejść, żeby uznać ją za skończoną.

Nazwa	Co to jest
token	Słowo o dwóch znaczeniach. W modelu to kawałek tekstu (słowo, jego część albo znak), na które model dzieli wszystko, co czyta i pisze; od liczby tokenów zależą szybkość i koszt. W dostępie to ciąg znaków, który zastępuje hasło i otwiera drzwi do usługi.
tylko do odczytu	Tryb, w którym narzędzie może czytać, ale nie może zmieniać. Każde nowe podłączenie zaczynamy od niego.
uwierzytelnianie i autoryzacja	Uwierzytelnianie sprawdza, kim jesteś. Autoryzacja sprawdza, co ci wolno.
wrapper	Mały program, który pobiera sekret z magazynu i uruchamia właściwy program, przekazując sekret przez środowisko, a nie przez argumenty wiersza poleceń.
zakres tokenu	Lista uprawnień przypisana tokenowi, czyli co wolno nim zrobić, na przykład tylko odczyt repozytorium.
Zielona	Fikcyjna firma, w której dzieją się nasze przykłady; opisuje ją strona „Świat Zielonej”. Tomek jest młodszym administratorem, Ola programistką, pani Halina główną księgową.
zmienna środowiskowa	Nazwana wartość przekazywana procesowi przy starcie. Tak serwer MCP typu stdio dostaje token, ale nie jest to tajemnica przed innymi programami tego samego konta.

□□ ZMIENIA SYSTEM

Opakowanie serwera Gitea, które czyta token z magazynu

Serwer `gitea-mcp` przyjmuje token we fladze albo w zmiennej `GITEA_ACCESS_TOKEN`. Flaga jest widoczna na liście procesów, więc opakowanie wstawia token do środowiska procesu potomnego, a w argumentach zostają tylko ustawienia jawne.

```
python -m pip install keyring
python -c "import getpass, keyring; keyring.set_password('gitea-ro', 'token',
getpass.getpass('Token Gitea (nie będzie widoczny): '))"
claude mcp add --scope user --env GITEA_HOST=https://git.zielona.example --transport
stdio gitea-ro -- python C:\AI\mcp\gitea_ro.py
```

□□ **claude mcp add** = polecenie Claude Code dopisujące serwer MCP do konfiguracji · **--scope user** = zakres: serwer dostępny we wszystkich projektach tego użytkownika, zapisany w jego pliku konfiguracji · **--transport stdio** = serwer uruchamiany jako proces potomny, który rozmawia z agentem przez standardowe wejście i wyjście · **--env** = ustawienie zmiennej środowiskowej dla uruchamianego serwera

□□ TYLKO ODCZYT

Sprawdzenie, że token nie jest widoczny w argumentach procesu

Polecenie wypisuje linie poleceń uruchomionych serwerów. Szukamy w nich flag, a nie tokena; jeśli token tam jest, serwer jest uruchomiony niewłaściwie.

```
Get-CimInstance Win32_Process -Filter "Name like 'gitea-mcp%'" | Select-Object CommandLine
```

Get-CimInstance = polecenie PowerShella, które odczytuje obiekty systemu przez Common Information Model · **Win32_Process** = klasa systemu Windows opisująca uruchomione procesy, razem z ich linią poleceń · **Select-Object** = polecenie PowerShella wybierające tylko wskazane pola wyniku

OSTROŻNIE · NIEODWRACALNE

Tryb zapisu serwera Gitea

Uruchomienie serwera bez flagi `-r` i z tokenem mającym zakres zapisu pozwala agentowi zakładać zgłoszenia, zmieniać pliki i uruchamiać zadania Actions. Robimy to wyłącznie na osobnym koncie, z ochroną gałęzi `main` i z pytaniem o zgodę przy każdym narzędziu.

Zanim zaczniemy

Mamy dwie usługi i dwa pytania do każdej: kto jest agentem po stronie usługi i co dokładnie może zobaczyć lub zmienić. W Gitea odpowiedzią jest konto serwisowe z tokenem o zakresach tylko do odczytu, a w BookStacku konto serwisowe z rolą bez praw zapisu i naszym własnym serwerem MCP z poprzedniej strony. Nie używamy kont ludzi, żeby w logach było widać, kto co zrobił, i żeby wyciek tokena nie dawał dostępu do wszystkiego, do czego ma dostęp człowiek.

Potrzebujemy zainstalowanego `gitea-mcp` w aktualnej wersji, Pythona z biblioteką `keyring` i serwera BookStacka z poprzedniej strony, Własny serwer MCP w Pythonie: BookStack tylko do odczytu. Zasady przechowywania tokenów opisuje Gdzie trzymać token: sekrety agenta poza czatem i poza plikiem, a konfigurację agentów MCP w Claude Code i Antigravity CLI: konfiguracja, zakresy i uprawnienia.

Z ŻYCIA WZIĘTE

Ola prosi, żeby agent w jej terminalu sam przeglądał zgłoszenia w Gitea i podsumowywał, co zostało do zrobienia przed wydaniem. Zakładamy konto `agent-odczyt`, dajemy mu prawo odczytu do trzech repozytoriów zespołu i wystawiamy token z zakresami odczytu

> repozytoriów i zgłoszeń. Tydzień później Ola chce, żeby agent dodawał komentarze do zgłoszeń. Tworzymy osobne konto z jednym dodatkowym zakresem i zostawiamy oba serwery obok siebie, a nie jeden z szerszym tokenem.

Krok 1: konto serwisowe i token w Gitea

W Gitea zakładamy zwykłego użytkownika `agent-odczyt` i nadajemy mu prawo odczytu tylko do repozytoriów, które agent ma widzieć, najwygodniej przez zespół z uprawnieniem tylko do odczytu. Konto nie dostaje praw administratora ani członkostwa w organizacjach, których nie potrzebuje. Repozytoria prywatne spoza listy pozostają dla niego niewidoczne, więc próba ich otwarcia kończy się odpowiedzią „nie znaleziono”.

Token tworzymy w ustawieniach konta, w sekcji aplikacji, wybierając zakresy. Gitea dzieli zakresy na kategorie, takie jak `repository`, `issue`, `organization`, `user` i `admin`, a każda ma wariant `read:` i `write:`. Zakres odczytu obejmuje trasy GET, zakres zapisu obejmuje trasy POST, PUT, PATCH i DELETE razem z odczytem, a zakres `all` daje wszystkie zakresy zapisu [1]. Dla agenta czytającego wybieramy `read:repository` i `read:issue`, a kategorię `admin` zostawiamy niezaznaczoną.

Datę wygaśnięcia i nazwę tokena wpisujemy tak samo jak w BookStacku, według zasad ze strony o sekretach. Token zapisujemy prosto do ukrytego pola z fiszki, bez notatnika pośredniego.

Krok 2: opakowanie czytające token z magazynu

Serwer `gitea-mcp` przyjmuje host i token jako flagi albo zmienne środowiskowe, przy czym flagi mają pierwszeństwo [2]. Flagę widać na liście procesów, więc token wpisujemy do zmiennej `GITEA_ACCESS_TOKEN`, ale nie w pliku konfiguracji agenta, tylko w opakowaniu, które sięga po niego do magazynu poświadczeń i uruchamia serwer z tym jednym dodatkowym wpisem w środowisku. Opakowanie jest krótkie i też mieści się w jednym pliku.

```
#!/usr/bin/env python3
"""Opakowanie dla gitea-mcp: token z magazynu poświadczeń trafia do serwera przez środowisko,
a nie przez argumenty, bo argumenty widać na liście procesów. Tryb tylko do odczytu włączamy
flagą -r."""
import os
```

```
import subprocess
import sys

import keyring

HOST = os.environ.get("GITEA_HOST", "https://git.zielona.example")

token = keyring.get_password("gitea-ro", "token")
if not token:
    sys.exit("Brak tokena gitea-ro w magazynie poświadczeń")

srodowisko = {**os.environ, "GITEA_ACCESS_TOKEN": token.strip()}
polecenie = ["gitea-mcp", "-t", "stdio", "-r", "--host", HOST]
sys.exit(subprocess.call(polecenie, env=srodowisko))
```

Flaga `-r` włącza tryb tylko do odczytu: według dokumentacji narzędzia z dostępem do zapisu są wtedy ukryte [2]. To warstwa po stronie serwera MCP, więc nie zastępuje ograniczenia po stronie Gitea: token ma zakresy tylko do odczytu, a serwer dodatkowo nie wystawia narzędzi zapisujących. Opakowanie sprawdziliśmy z atrapą programu `gitea-mcp`: argumenty nie zawierały tokena, środowisko go miało, a dane na standardowym wejściu przechodziły bez zmian.

Krok 3: rejestracja w agentach

W Claude Code używamy `claude mcp add` z fiszki: w poleceniu stoi tylko adres Gitea i ścieżka do opakowania. W Antigravity CLI wpis trafia do pliku konfiguracji serwerów, a opakowanie jest poleceniem serwera stdio.

```
{
  "mcpServers": {
    "gitea-ro": {
      "command": "python",
      "args": ["C:\\AI\\mcp\\gitea_ro.py"],
      "env": { "GITEA_HOST": "https://git.zielona.example" }
    }
  }
}
```

W regułach uprawnień Claude Code pozwalamy na `mcp__gitea-ro__*`, bo serwer jest uruchomiony w trybie odczytu, a token ma zakresy odczytu. Gdyby ktoś usunął flagę `-r`, drugą warstwą pozostają

zakresy tokena. Dla serwera z zapisem ta sama reguła byłaby wpisem `ask`, tak jak w przykładzie ze strony o konfiguracji.

Krok 4: BookStack

Dla BookStacka używamy własnego serwera z poprzedniej strony i konta serwisowego z rolą bez praw zapisu. Zalety są takie same jak przy Gitea: znamy każdą linię kodu, token czytamy z magazynu, a lista dozwolonych książek jest drugim ograniczeniem po stronie serwera. Rejestrację wykonujemy poleceniem ze strony o sekretach.

Gdy w grę wchodzi cudzy serwer BookStacka, na przykład znaleziony w rejestrze serwerów, traktujemy go jak każdy inny kod, który dostanie token. Przechodzimy listę kontrolną z następnej sekcji i dopiero potem decydujemy, czy wdrażać go u nas, czy zostać przy własnym.

Lista kontrolna przed użyciem cudzego serwera MCP

Cudzy serwer MCP jest programem uruchamianym z naszymi uprawnieniami i z dostępem do tokena, więc oceniamy go jak każdy inny program tej klasy. Poniższa lista zawiera sześć pytań i do każdego dopisujemy, na czym opieramy odpowiedź.

Pytanie	Czego szukamy
skąd pochodzi serwer	oficjalny projekt usługi albo znany autor, aktywne repozytorium, czytelny kod
jak przyjmuje token	zmienna, nagłówek albo OAuth; flaga w linii poleceń jest gorsza, bo widać ją na liście procesów
czy ma tryb odczytu albo filtr narzędzi	flaga lub zmienna, która ukrywa narzędzia zapisujące, i możliwość wskazania listy narzędzi
dokąd wysyła dane	adresy w kodzie poza naszą usługą, telemetria, wywołania do usług trzecich
co loguje	czy dziennik zawiera żądania, nagłówki albo treści, i dokąd trafiają logi
jak aktualizujemy	przypięta wersja, czytanie listy zmian przed aktualizacją

Serwer `gitea-mcp` ma ponadto przydatne pokrętła: w bieżącym README są filtr zakresu (`-S`), filtr listy narzędzi (`-O`) i wsparcie dla OAuth, a logi trafiają na stderr. Flaga `-d` włącza logowanie każdego żądania wysłanego do Gitea, więc włączamy ją tylko do diagnozy, a po niej usuwamy i

czyścimy log [2]. Pełną listę opcji pokazuje `gitea-mcp --help` w wersji, którą mamy zainstalowaną.

Wariant z zapisem: osobne konto, ochrona gałęzi, potwierdzenie

Wersję z zapisem budujemy jako drugi, osobny serwer z osobnym kontem, a nie przez rozszerzenie tokena do odczytu. Konto `agent-zapis` dostaje zakres `write:repository` albo `write:issue`, zależnie od zadania, i prawo zapisu tylko do repozytoriów, które tego wymagają. Na gałęzi `main` włączamy w Gitea ochronę: bezpośredni push jest zablokowany, a zmiany wchodzą przez pull request z zatwierdzeniem człowieka.

Narzędzia zapisu serwera `gitea-mcp` wymagają od wersji 1.8.0 podania nazwy gałęzi i komunikatu, więc agent nie zatwierdza po cichu zmian na gałęzi domyślnej [3]. Mimo to ochrona gałęzi po stronie Gitea pozostaje obowiązkowa, bo to ona działa także wtedy, gdy serwer zawiedzie. W regułach agenta każde narzędzie tego serwera ma wpis `ask`, a człowiek przy każdej zgodzie widzi, co agent chce zrobić.

> **Zgłoszenia z zewnątrz to obca treść.** Jeśli agent czyta zgłoszenia, które mogą napisać osoby spoza zespołu, a jednocześnie ma narzędzia zapisu, mamy komplet trzech nóg: dane prywatne, obcą treść i możliwość działania. Dlatego agent czytający zgłoszenia nie dostaje narzędzi zapisu, a agent piszący nie czyta obcych zgłoszeń bez pośrednika, jak opisuje strona Śmiertelna trójka: dane, obca treść i wyjście na zewnątrz.

Problemy i rozwiązania

Objaw	Przyczyna	Rozwiązanie
serwer kończy się komunikatem o braku tokena	wpis `gitea-ro` założono na innym koncie Windows albo pod inną nazwą	ponawiamy wpis na tym koncie, na którym działa agent
odpowiedź 401 z Gitea	token wygasł, został usunięty albo wpisano go z białymi znakami	wystawiamy nowy token i wpisujemy go ponownie
odpowiedź 403 przy próbie zapisu	token ma zakresy odczytu	to zamierzone działanie; zapis wymaga osobnego konta i serwera
odpowiedź „nie znaleziono” dla istniejącego repozytorium	konto serwisowe nie ma do niego dostępu	dodajemy konto do zespołu z prawem odczytu

Objaw	Przyczyna	Rozwiązanie
token widać na liście procesów	token podano we fladze `--token`	przechodzimy na opakowanie i zmienną środowiskową
uszkodzony protokół MCP, błędy parsowania po stronie agenta	w starszych wersjach `-d` pisało diagnostykę na stdout	aktualizujemy `gitea-mcp`, a diagnostykę oglądamy na stderr
dziennik serwera zawiera zapytania do Gitea	włączona flaga `-d`	wyłączamy flagę i usuwamy log

Test akceptacyjny

Prosimy agenta o listę otwartych zgłoszeń w repozytorium zespołu i oczekujemy poprawnej odpowiedzi. Prosimy o założenie zgłoszenia i oczekujemy, że agent powie, że takiego narzędzia nie ma. Jednorazowo uruchamiamy serwer bez flagi `-r` na koncie z tokenem odczytu i próbujemy to samo: Gitea powinna odpowiedzieć odmową, co potwierdza drugą warstwę ochrony.

Następnie prosimy o otwarcie prywatnego repozytorium spoza listy i oczekujemy odpowiedzi „nie znaleziono”. Fiszką odczytu sprawdzamy listę procesów i potwierdzamy, że w linii poleceń nie ma tokena. Na koniec usuwamy token w Gitea i obserwujemy, że agent dostaje błąd autoryzacji, a nie zawiesza się; test zaliczamy, gdy wszystkie pięć obserwacji zgadza się z opisem.

Ćwiczenia

1. Dlaczego włączamy tryb `-r` w serwerze, skoro token ma już zakresy tylko do odczytu?

Odpowiedź

To dwie niezależne warstwy. Zakresy tokena egzekwuje Gitea i działają nawet wtedy, gdy serwer MCP zawiedzie albo ktoś usunie flagę. Flaga `-r` ukrywa narzędzia zapisujące przed modelem, więc agent w ogóle nie próbuje ich użyć, a człowiek nie dostaje pytań o zgodę na operacje, których i tak nie da się wykonać.

2. Zespół chce, żeby agent czytający zgłoszenia z zewnątrz mógł też od razu je zamykać. Co proponujemy zamiast jednego serwera z zapisem?

Odpowiedź

Dwa osobne serwery i dwa osobne konta. Agent czytający dostaje tylko odczyt i nie widzi narzędzi zapisu, a zamykanie zgłoszeń wykonuje drugi agent albo człowiek po przeczytaniu

streszczenia, które nie zawiera obcej treści w surowej postaci. Dzięki temu obca treść nigdy nie trafia do agenta, który ma narzędzia zapisu.

3. Dlaczego opakowanie przekazuje token w zmiennej środowiskowej procesu potomnego, a nie we fladze `--token`?

Odpowiedź

Flaga jest częścią linii poleceń procesu, którą widać na liście procesów dla innych programów na tym samym komputerze. Zmienna środowiskowa procesu jest trudniejsza do odczytania i nie pojawia się w opisie procesu. To nie jest ochrona doskonała, ale usuwa najłatwiejszą drogę wycieku.

Źródła i dalsze lektury

Przypisy

[1] Zakresy tokenów dostępu w Gitea (kategorie, warianty odczyt i zapis, `all`, `public-only`): zmiana „Redesign Scoped Access Tokens” (#24767).

[2] Flagi i zmienne `gitea-mcp`, tryb odczytu `-r` i `GITEA_READONLY`, filtry `-S` i `-O`, flaga `-d`, tryb HTTP i OAuth: README projektu gitea-mcp; wersja opisana na stronie to 1.10.0 z 18 października 2026 roku.

[3] Wymóg nazwy gałęzi i komunikatu w narzędziach zapisu plików, tryb OAuth, poprawki obsługi logów: lista wydań gitea-mcp.

Lektury

Dlaczego agent czytający obce treści nie dostaje narzędzi zapisu, tłumaczy Śmiertelna trójka: dane, obca treść i wyjście na zewnątrz, a jak podzielić role agentów w planie wdrożenia, pokazuje Agenci w planie wdrożenia: role, uprawnienia i granice.

Nota metodyczna: czego nie wiemy

Opisujemy `gitea-mcp` w wersji 1.10.0 według README z 20 października 2026 roku. Serwera nie uruchamialiśmy z prawdziwą instancją Gitea, więc zachowanie flagi `-r` i nazw narzędzi opisujemy według dokumentacji, a opakowanie sprawdziliśmy tylko z atrapą programu. Dokumentacja wspomina o filtrach `-S` i `-O`, ale nie wyjaśnia ich do końca, więc przed użyciem czytamy wynik `gitea-mcp --help`.

Nie sprawdzaliśmy, czy w Gitea istnieje zakres tokena ograniczający dostęp do pojedynczego repozytorium: ograniczenie realizujemy prawami konta serwisowego, a nie zakresem. Nazwy zakresów pochodzą z opisu zmiany z 2023 roku i w bieżącej wersji Gitea mogły zostać rozszerzone, więc przy wystawianiu tokena czytamy listę, którą pokazuje interfejs.

Revision #3
Created 2026-10-20 08:59:12 UTC by Tomasz Konieczny
Updated 2026-10-20 09:05:27 UTC by Tomasz Konieczny