MCP w Claude Code i Antigravity CLI: konfiguracja, zakresy i uprawnienia

Poziom: zaawansowane **Dotyczy: Claude Code, Antigravity CLI, MCP** **Czas: ~40 min**

Serwer MCP trzeba nie tylko podłączyć, ale też powiedzieć agentowi, co wolno mu z niego wywołać bez pytania, co tylko po zgodzie, a czego wcale. Pokazujemy, gdzie leżą pliki konfiguracji w Claude Code i Antigravity CLI, jak działają zakresy i zatwierdzanie serwerów z repozytorium, jak pisać reguły dla narzędzi MCP i dlaczego same reguły nie są jeszcze granicą bezpieczeństwa.

Skróty użyte na tej stronie (7)

Skrót	Po angielsku	Po polsku i co znaczy
AI	Artificial Intelligence	sztuczna inteligencja: programy, które naśladują część ludzkiego rozumowania i rozmawiają w języku naturalnym
CI	Continuous Integration	ciągła integracja: automatyczne budowanie i testowanie kodu po każdej zmianie
CLI	Command Line Interface	interfejs wiersza poleceń: praca przez wpisywanie komend w terminalu
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
SDK	Software Development Kit	zestaw programistyczny: biblioteki i narzędzia do budowania programów dla danej platformy

Leksykon: nazwy i pojęcia z tej strony (27)

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.
Antigravity CLI	Narzędzie Google do pracy z agentami kodującymi, w wersji uruchamianej z terminala. Konfigurację serwerów MCP trzyma w pliku w profilu użytkownika.
Bash	Domyślna powłoka większości systemów Linux.
Bearer	Schemat nagłówka HTTP Authorization, w którym po słowie Bearer idzie token. Kto go okaże, zostaje wpuszczony, więc token trzeba chronić jak hasło.
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.
Gemini	Rodzina modeli i czat firmy Google.
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.
hook	Skrypt uruchamiany przez Claude Code w określonym momencie, na przykład przed wywołaniem narzędzia (PreToolUse). Może zablokować akcję niezależnie od tego, co postanowił model.
piaskownica	Odgrodzone środowisko (kontener, osobne konto, maszyna wirtualna), w którym agent może coś zepsuć, nie dotykając reszty systemu.
Python	Język programowania, w którym piszemy klienty, serwery MCP i skrypty.
reguły permissions	W Claude Code reguły allow, ask i deny, które rozstrzygają, co agent zrobi bez pytania, po pytaniu albo wcale. Reguły dla poleceń powłoki nie są granicą bezpieczeństwa same w sobie.
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.
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.
wywołanie narzędzia	Prośba modelu o uruchomienie funkcji, na przykład odczytu pliku albo zapytania do BookStacka. Model tylko prosi, a wykonuje program, który może odmówić.
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.

□□ PLIK KONFIGURACYJNY

Plik `.mcp.json` w repozytorium: serwer bez sekretów

Plik współdzielony przez zespół zawiera tylko polecenie uruchomienia serwera i ustawienia jawne. Przy pierwszym użyciu Claude Code pyta każdą osobę o zgodę na uruchomienie serwera z tego pliku.

```
{
  "mcpServers": {
    "bookstack-ro": {
      "command": "python",
      "args": ["C:\\AI\\mcp\\bookstack_ro.py"],
      "env": {
        "BOOKSTACK_URL": "https://wiedza.zielona.example",
        "BOOKSTACK_KSIAZKI": "19,829"
      }
    }
  }
}
```

PLIK KONFIGURACYJNY

Reguły uprawnień dla narzędzi MCP w `.claude/settings.json`

Odczyty z naszej bazy wiedzy pozwalamy bez pytania, wszystko z Gitea wymaga zgody, a pobieranie stron z sieci wyłączamy, bo agent czytający obce treści nie powinien mieć drogi na zewnątrz.

```
{
  "permissions": {
    "allow": ["mcp__bookstack-ro__szukaj", "mcp__bookstack-ro__czytaj_strone"],
    "ask": ["mcp__gitea__*"],
    "deny": ["WebFetch"]
  }
}
```

TYLKO ODCZYT

Podgląd stanu serwerów i reguł

Te polecenia niczego nie zmieniają. Pierwsze pokazuje serwery i ich stan, drugie szczegóły jednego serwera. Obowiązujące reguły przeglądamy w interfejsie agenta poleceniem `/permissions`.

```
claude mcp list
claude mcp get bookstack-ro
```

claude mcp list = polecenie Claude Code, które wypisuje skonfigurowane serwery MCP i ich stan · **claude mcp get** = polecenie Claude Code, które pokazuje szczegóły jednego serwera · **/permissions** = polecenie wpisywane w sesji Claude Code, które pokazuje obowiązujące reguły zgody

Zanim zaczniemy

Mamy już serwer z poprzedniej strony, Własny serwer MCP w Pythonie: BookStack tylko do odczytu, i token w magazynie poświadczeń. Teraz ustawiamy, w którym miejscu agent o nim wie i co z niego wolno. Zakładamy, że zespół Oli pracuje w Claude Code, a my z Tomkiem sprawdzamy to samo w Antigravity CLI, więc opisujemy oba narzędzia obok siebie.

Konfiguracja ma trzy poziomy, które łatwo pomylić. Pierwszy to samo istnienie serwera: gdzie jest zapisany i kto go widzi. Drugi to zatwierdzenie: kto zgodził się na uruchomienie polecenia z cudzego pliku. Trzeci to uprawnienia: które narzędzia serwera agent może wywołać bez pytania, a które nie.

> **Z ŻYCIA WZIĘTE**
>
> Ola dodaje do repozytorium plik `.mcp.json` z serwerem do Gitea, żeby cały zespół miał te same narzędzia. Tomek klonuje repozytorium i przy pierwszym uruchomieniu agenta widzi pytanie o zgodę na serwer z tego pliku. Zamiast kliknąć „tak”, otwiera plik i pyta Olę, co to za polecenie, a okazuje się, że ścieżka w nim prowadziła do jej własnego katalogu, którego Tomek nie ma. Dzięki pytaniu poprawiamy plik przed pierwszym użyciem, a nie po.

Zakresy w Claude Code

Claude Code przechowuje serwery MCP w trzech zakresach, które różnią się tym, kto je widzi. Zakres lokalny jest domyślny: serwer jest dostępny tylko w jednym projekcie i tylko dla nas, a jego definicja leży w pliku `~/.claude.json`. Zakres użytkownika też leży w `~/.claude.json`, ale obowiązuje we wszystkich projektach na naszym koncie. Zakres projektu to plik `.mcp.json` w katalogu repozytorium, który trafia do gita i widzi go każdy, kto repozytorium sklonuje [1].

Zakres	Gdzie zapisany	Kto widzi	Kiedy używamy
lokalny (domyślny)	`~/.claude.json`, przypisany do projektu	tylko my, w tym projekcie	próby, serwery z danymi dostępowymi
użytkownika	`~/.claude.json`	tylko my, we wszystkich projektach	osobiste narzędzia, na przykład nasz serwer BookStacka
projektu	`.mcp.json` w repozytorium	cały zespół, przez git	serwery potrzebne wszystkim, bez żadnych sekretów

Zakres wybieramy opcją `--scope` w poleceniu `claude mcp add`. Przy zakresie projektu pilnujemy jednej rzeczy: w pliku mogą stać tylko zmienne `${NAZWA}` i wartości jawne, a każda tajna wartość pochodzi z magazynu, jak na stronie Gdzie trzymać token: sekrety agenta poza czatem i poza plikiem.

Zatwierdzanie serwerów z repozytorium

Plik `.mcp.json` ze sklonowanego repozytorium jest cudzym kodem: definiuje polecenie, które agent uruchomi na naszym komputerze. Dlatego w sesji interaktywnej Claude Code pyta o zgodę przed użyciem każdego serwera z takiego pliku. Decyzje można wyczyścić poleceniem `claude mcp reset-project-choices` [1].

Są trzy sytuacje, w których pytanie się nie pojawia. Tryb nieinteraktywny `claude -p`, SDK agenta i sesje w chmurze nie mają kogo zapytać, więc ładują serwery projektu bez pytania. W takich przypadkach serwer można zablokować wpisem `disabledMcpjsonServers`, opcją `--strict-mcp-config` albo ograniczeniem źródeł ustawień do ustawień użytkownika (`--setting-sources user`) [2]. Repozytorium nie może też samo zatwierdzić swoich serwerów: wpis w jego `.claude/settings.json` nie działa w katalogu, któremu nie zaufaliśmy [1].

> **Cudzy `.mcp.json` jest cudzym poleceniem.** Pytanie o zgodę dotyczy serwera z pliku, więc przed kliknięciem czytamy sam plik i polecenie, które w nim stoi. W automatyzacjach i potokach CI, gdzie nikt nie klika, repozytorium uruchamiane przez agenta powinno mieć zablokowane serwery, których nie wskazaliśmy sami.

Reguły uprawnień dla narzędzi MCP

Reguły w sekcji `permissions` mają trzy listy: `deny`, `ask` i `allow`. Claude Code sprawdza je w tej kolejności, a pierwsza pasująca decyduje, więc reguła zezwalająca nie może zrobić wyjątku od reguły zakazującej [2]. Dla narzędzi MCP nazwa reguły składa się z przedrostka `mcp__`, nazwy serwera i nazwy narzędzia: `mcp__bookstack-ro__szukaj` pasuje do jednego narzędzia, a `mcp__gitea__*` do wszystkich narzędzi serwera `gitea`.

Kilka szczegółów składni często zaskakuje. Reguły zezwalające akceptują gwiazdkę tylko po literalnym przedrostku `mcp__<serwer>__`, więc `mcp__*` jako reguła `allow` jest pomijane z ostrzeżeniem, a jako `deny` blokuje wszystkie narzędzia MCP wszystkich serwerów. Kropki i spacje w nazwie serwera zamieniają się w `_`, więc serwer `my.server` zapisujemy jako `mcp__my_server__*`. Reguła `mcp__` z nawiasami jest pomijana przy wczytywaniu ustawień, a do zakazania narzędzia z konkretnym parametrem służy opcja `--disallowedTools` [2].

Reguły to nie jest granica

Dokumentacja Claude Code mówi wprost, że reguła zakazu dla Bash obejmuje formę polecenia, którą agent zwykle zapisuje, i nie jest granicą bezpieczeństwa wokół programu. Zakaz `curl` nie pokrywa tego samego programu uruchomionego inaczej, na przykład przez `sh -c`. Dlatego uprawnienia i piaskownica to dwie warstwy, które się uzupełniają: reguły decydują, o co agent pyta, a piaskownica ogranicza na poziomie systemu, do czego jego polecenia faktycznie sięgną [2].

Do kontroli, która nie zależy od tekstu polecenia, służy też hak PreToolUse: skrypt uruchamiany przed każdym narzędziem, który może je zatrzymać. Hak nie obchodzi reguł, bo zakaz i pytanie zachowują pierwszeństwo nawet wtedy, gdy hak zwróci zgodę. Tryb `bypassPermissions`, który wyłącza pytania, dokumentacja zaleca wyłącznie w odizolowanych środowiskach, takich jak kontenery i maszyny wirtualne [2].

Antigravity CLI

W Antigravity CLI serwery leżą w jednym z dwóch plików: globalnym `~/.gemini/config/mcp_config.json` i projektowym `.agents/mcp_config.json`. Wpis ma pole `command` dla serwera stdio albo `serverUrl` dla serwera HTTP, a do tego opcjonalne `args`, `env`, `cwd` i `headers`. Pole `disabled` wyłącza serwer bez usuwania wpisu, a `disabledTools` ukrywa wskazane narzędzia przed modelem, więc odpowiednik reguły zakazu mamy w samej konfiguracji serwera [3].

Domyślnie narzędzia MCP, których nie skonfigurowano, działają w trybie pytania, czyli wymagają naszej zgody przed każdym wywołaniem. Reguły polityki mają postać `mcp(serwer/narzędzie)` dla jednego narzędzia, `mcp(serwer/*)` dla całego serwera i `mcp(*)` dla wszystkich narzędzi wszystkich serwerów [3]. Tokeny OAuth są zapisywane w pliku `~/.gemini/antigravity/mcp_oauth_tokens.json`.

```
{
  "mcpServers": {
    "bookstack-ro": {
      "command": "python",
      "args": ["C:\\AI\\mcp\\bookstack_ro.py"],
      "env": {
        "BOOKSTACK_URL": "https://wiedza.zielona.example",
        "BOOKSTACK_KSIAZKI": "19,829"
      },
      "disabledTools": []
    }
  }
}
```

Przykład z dokumentacji Antigravity wpisuje token wprost w polu `headers` jako `Bearer YOUR_TOKEN`. U nas tej drogi nie używamy: dla serwerów zdalnych wybieramy OAuth, a dla własnych serwer stdio, który czyta sekret sam. W dokumentacji nie znaleźliśmy opisu rozwijania zmiennych w polu `env`, więc przy Antigravity nie liczymy na to, że `${NAZWA}` zostanie podstawiona [3].

Sprawa	Claude Code	Antigravity CLI
plik globalny	`~/.claude.json`	`~/.gemini/config/mcp_config.json`
plik projektu	`.mcp.json`	`.agents/mcp_config.json`
zmienne w konfiguracji	`${NAZWA}` i `${NAZWA:-wartość}` w `command`, `args`, `env`, `url`, `headers`	brak opisu w dokumentacji
ukrycie narzędzia	reguła `deny` z `mcp__serwer__narzędzie`	`disabledTools` w konfiguracji serwera
domyślne zachowanie	pyta przed użyciem serwerów z repozytorium i przed narzędziami bez reguły	narzędzia MCP bez konfiguracji działają w trybie pytania
reguły polityki	`permissions.allow`, `ask`, `deny`	`mcp(serwer/narzędzie)`, `mcp(serwer/*)`, `mcp(*)`
tokeny OAuth	zapisane przez klienta, dokumentacja nie podaje miejsca	`~/.gemini/antigravity/mcp_oauth_tokens.json`

Problemy i rozwiązania

Objaw	Przyczyna	Rozwiązanie
`claude mcp list` pokazuje „Pending approval”	serwer z `.mcp.json` czeka na zgodę	uruchamiamy `claude` interaktywnie i zatwierdzamy, po przeczytaniu polecenia
serwer nie startuje po `claude mcp add`	opcja `--env` stoi tuż przed nazwą serwera i nazwa została odczytana jako para zmiennej	wstawiamy między nie `--transport stdio`
reguła z nawiasami nie działa	reguły `mcp__` z nawiasami są pomijane	używamy reguły bez nawiasów albo opcji `--disallowedTools`
reguła `allow` z `mcp__*` nic nie zatwierdza	gwiazdka w regule zezwalającej musi stać po `mcp__<serwer>__`	wskazujemy serwer: `mcp__bookstack-ro__*`
reguła nie działa dla serwera o nazwie z kropką	kropka w nazwie zamienia się w `_`	zapisujemy `mcp__nazwa_serwera__*`
`claude -p` nie potrafi się zalogować do serwera zdalnego	w trybie nieinteraktywnym nie ma uwierzytelniania OAuth	najpierw logujemy się w sesji interaktywnej
serwer w Antigravity nie widzi zmiennej z `${NAZWA}`	brak udokumentowanego rozwijania zmiennych w `env`	wpisujemy wartość jawną albo przechodzimy na serwer czytający magazyn

Test akceptacyjny

Sprawdzamy pięć zachowań, każde w osobnej sesji. Po pierwsze `claude mcp list` pokazuje serwer jako połączony. Po drugie prośba o wyszukanie strony kończy się odpowiedzią bez pytania o zgodę, bo narzędzia `szukaj` i `czytaj_strone` są na liście `allow`.

Po trzecie wywołanie dowolnego narzędzia `gitea` zatrzymuje się na pytaniu. Po czwarte prośba o pobranie strony z sieci jest odrzucana przez regułę `deny`.

Po piąte tworzymy w katalogu tymczasowym repozytorium z własnym `.mcp.json`, w którym stoi nieznane polecenie, i uruchamiamy w nim agenta. Oczekujemy pytania o zgodę, a po odmowie braku tego serwera na liście. Test zaliczamy, gdy wszystkie pięć zachowań zgadza się z opisem.

Ćwiczenia

1. W `permissions.allow` wpisano `mcp__*`. Co się stanie?

Odpowiedź

Nic nie zostanie zatwierdzone. Reguły zezwalające akceptują gwiazdkę tylko po literalnym przedrostku `mcp__<serwer>__`, więc taka reguła jest pomijana z ostrzeżeniem. Jako reguła `deny` ten sam zapis zablokowałby wszystkie narzędzia MCP wszystkich serwerów.

2. Reguła `allow` zezwala na `mcp__gitea__create_issue`, a reguła `deny` zakazuje `mcp__gitea__*`. Które narzędzie zadziała?

Odpowiedź

Żadne z serwera `gitea`. Reguły są sprawdzane w kolejności zakaz, pytanie, zgoda, a pierwsza pasująca decyduje, więc zgoda nie może zrobić wyjątku od zakazu.

3. Dlaczego w pipeline CI, w którym działa `claude -p`, samo `.mcp.json` w repozytorium jest ryzykiem?

Odpowiedź

W trybie nieinteraktywnym nie ma pytania o zgodę, więc serwery z `.mcp.json` ładują się bez zatwierdzenia. Ktokolwiek może zmienić ten plik w pull requeście, może więc dopisać polecenie, które uruchomi się w CI. Blokujemy takie serwery wpisem `disabledMcpjsonServers`, opcją `--strict-mcp-config` albo ograniczeniem źródeł ustawień.

Źródła i dalsze lektury

Przypisy

[1] Zakresy, plik `.mcp.json`, zatwierdzanie serwerów projektu, `claude mcp reset-project-choices`, `claude mcp add` z opcjami `--env`, `--transport`, `--scope`: Claude Code, MCP.

[2] Kolejność reguł zakaz, pytanie, zgoda, składnia reguł dla narzędzi MCP, pomijanie reguł z nawiasami, ograniczenia reguł dla Bash, piaskownica, haki PreToolUse, tryb `bypassPermissions`, zachowanie w trybie nieinteraktywnym: Claude Code, Configure permissions.

[3] Pliki `mcp_config.json`, pola konfiguracji, `disabledTools`, domyślny tryb pytania, wzorce `mcp(...)`, miejsce zapisu tokenów OAuth: Antigravity, MCP.

Lektury

Dobór reguł do konkretnych serwerów opisuje Gitea i BookStack przez MCP: bezpieczne spięcie krok po kroku, a uzasadnienie, dlaczego agent czytający obce treści nie dostaje drogi na zewnątrz, daje Śmiertelna trójka: dane, obca treść i wyjście na zewnątrz.

Nota metodyczna: czego nie wiemy

Opisujemy dokumentację z 20 października 2026 roku. Nazwy pól i opcji obu narzędzi zmieniają się szybko, więc przed wdrożeniem sprawdzamy je w bieżącej dokumentacji i poleceniem `claude mcp --help`. Nie uruchamialiśmy Antigravity CLI na własnym komputerze, więc jego opis opieramy na dokumentacji, a nie na własnym teście.

W dokumentacji Antigravity nie znaleźliśmy wskazania, który plik ma pierwszeństwo, gdy ten sam serwer jest zdefiniowany globalnie i w projekcie, więc unikamy podwójnych definicji. Nie

sprawdzaliśmy też ustawień zarządzanych przez administratora, takich jak narzucanie listy serwerów dla całej organizacji, i nie opisujemy ich tutaj.

Revision #3
Created 2026-10-20 08:59:12 UTC by Tomasz Konieczny
Updated 2026-10-20 09:05:26 UTC by Tomasz Konieczny