Nix

Guia de uso

Instalação, configuração e uso diário

O Nix conecta suas pastas de notas em Markdown aos seus agentes de desenvolvimento. Encontre ideias, leia e atualize notas com ferramentas que rodam na sua máquina.

Suas notas trabalham com você.

Uma ponte entre o conhecimento que você já organizou e o agente que ajuda a construir suas ideias.

Mais do que palavras

A busca híbrida combina busca semântica e léxica para encontrar ideias, nomes e termos nas suas notas.

Do agente para a wiki

Leia, crie e atualize notas em Markdown sem sair do seu fluxo de desenvolvimento.

Local por padrão

Embeddings e banco vetorial rodam localmente. Você decide quando sincronizar o índice.

Pronto para começar?

Windows (PowerShell): irm https://raw.githubusercontent.com/paulocesaaars/nix/main/install.ps1 | iex

Linux / macOS: curl -fsSL https://raw.githubusercontent.com/paulocesaaars/nix/main/install.sh | sh

Você precisa de uma pasta para suas notas em Markdown e de um sistema compatível. Crie uma pasta vazia ou use uma pasta com arquivos .md; ela será sua wiki. O binário da release já inclui o runtime: não é necessário instalar Python.

Disponível para Windows amd64, Linux amd64 ou arm64 e macOS Apple Silicon. A primeira sincronização baixa o modelo padrão, de cerca de 220 MB.

```
irm https://raw.githubusercontent.com/paulocesaaars/nix/main/install.ps1 | iex
```

Ao terminar, execute nix para abrir a interface. Se o comando não funcionar, feche o terminal, abra outro e tente de novo. Ver instruções completas

Configure e faça sua primeira busca.

Uma wiki é uma pasta de arquivos Markdown (.md). O índice é uma cópia preparada para busca; suas notas continuam na pasta original.

Inicialize a wiki

No terminal, execute nix init. Informe o caminho completo da pasta e escolha o modelo quando solicitado. Use lulu para começar. A primeira wiki recebe o nome Default.

```
nix init
```

Para informar o caminho diretamente: nix init --wiki "C:/Notas/MinhaWiki". Adapte o caminho ao seu sistema. A pasta deve existir.

Confira e crie o índice

Execute os comandos abaixo em sequência. A primeira sincronização baixa o modelo; aguarde a conclusão.

```
nix doctor / nix sync / nix status
```

Se a pasta estiver vazia, acrescente um arquivo .md e rode nix sync novamente.

Encontre uma nota na interface

Execute nix ui, abra Busca e pesquise uma palavra ou ideia presente nas notas. A interface também oferece Notas, Tags e Insights. Encerre pelo botão Encerrar ou com Ctrl+C no terminal.

Conecte um agente

Na interface, abra Integração Conectar cliente e copie as instruções do seu cliente. Use o caminho exibido por nix doctor, salve a configuração e reinicie ou recarregue o cliente MCP. Depois peça: “Busque nas minhas notas por arquitetura”.

O cliente inicia o servidor MCP; mantenha a wiki inicializada. A conexão usa o processo local, não uma URL HTTP.

Consulte, registre e mantenha suas notas.

Busque e leia

Na interface, use Busca, Notas, Tags e Insights. No agente, peça para buscar uma ideia, ler a nota e informar o caminho da fonte.

Crie ou altere com o agente

Peça para criar uma nota, acrescentar conteúdo, editar tags ou mover arquivos. Escritas pelo MCP atualizam o índice durante a operação; exclusões e substituições exigem confirmação.

Sincronize alterações externas

Após editar arquivos fora do Nix, execute nix sync ou peça ao agente para sincronizar. nix sync --dry-run apresenta uma prévia; nix status mostra a situação.

Administre e recupere

Na interface, use Wikis para cadastrar pastas, Configuração e Histórico para ajustar opções, e Backups para comparar ou restaurar uma versão anterior. Use Diagnóstico ou nix doctor quando algo falhar.

Mais possibilidades para seu conhecimento.

O Nix oferece ferramentas para trabalhar com suas notas e administrar o ambiente local.

Até 100 wikis

Cadastre wikis nomeadas e busque em uma, em várias ou em todas. Cada escrita acontece em uma única wiki.

Administre pelo navegador

Execute nix ui para consultar notas, sincronizar, diagnosticar e configurar o Nix na sua máquina.

Continue em outro agente

Com o MCP da Nix no Claude e no Codex, peça para registrar onde parou e, no outro, para continuar.

Você escolhe quando

Alterações externas entram no índice com nix sync; escritas feitas pelo Nix atualizam o índice na operação.

Escrita com cuidado

Operações destrutivas exigem confirmação e versões anteriores são guardadas em backups antes de sobrescrever.

Escolha o embedding

O padrão é o modelo local lulu. O spitz é uma alternativa maior; trocar de modelo exige nix sync --full.

Todos os comandos.

Cada comando abaixo é de uso no terminal. nix hook, nix banner e nix brain work existem só para o processo interno.

Geral

```
nix
```

No terminal, abre a interface. Sem terminal, o cliente MCP recebe o servidor.

```
nix --version
```

Mostra a versão e sai.

```
nix init
```

Cadastra a primeira wiki e o modelo de embedding. --wiki é a pasta; sem ela, o terminal pergunta. --name batiza a wiki. --embedding-model escolhe lulu ou spitz. --force regrava a configuração, sem alterar as notas.

```
nix ui
```

Abre a interface no navegador. Exige a wiki já cadastrada. --no-browser só imprime a URL. --port fixa a porta.

```
nix doctor
```

Confere ambiente, configuração, o comando que o MCP deve chamar e o índice. --json entrega o relatório.

Índice

```
nix sync
```

Atualiza o índice com o que mudou na wiki fora do Nix. --full reconstrói tudo. --dry-run só mostra. --json entrega o relatório. --wiki escolhe uma ou várias. --all percorre todas.

```
nix status
```

Mostra se o índice está atrás da wiki. Aceita --json, --wiki e --all.

Wikis

```
nix wiki list
```

Lista as wikis, a pasta e qual é a padrão. --json entrega a lista.

```
nix wiki add NOME CAMINHO
```

Cadastra outra wiki. O limite é 100.

```
nix wiki rename ANTIGO NOVO
```

Troca o nome no cadastro. Não move a pasta nem o índice.

```
nix wiki remove NOME
```

Tira a wiki do cadastro. Recusa a padrão e a última. A pasta de notas permanece.

```
nix wiki default NOME
```

Escolhe a wiki usada quando o nome é omitido.

```
nix wiki path NOME CAMINHO
```

Aponta a wiki para outra pasta.

```
nix wiki set NOME CAMPO VALOR
```

Grava um ajuste daquela wiki: include, exclude, follow_symlinks, default_new_note_folder, default_frontmatter ou longterm_folder.

Configuração

```
nix config list
```

Lista os campos globais, o valor que vale e a origem. --json entrega a lista.

```
nix config get CAMPO
```

Mostra o valor de um campo, no formato secao.chave.

```
nix config set CAMPO VALOR
```

Grava o campo no settings.db. Uma variável NIX_SECAO__CAMPO continua ganhando do banco.

```
nix config unset CAMPO
```

Apaga o valor gravado e volta ao padrão.

```
nix config history
```

Lista as alterações recentes. --limit muda a quantidade.

```
nix config undo [ID]
```

Restaura o valor anterior. Sem id, desfaz a última.

Mega Brain

```
nix brain enable --wiki NOME
```

Liga o Mega Brain e escolhe a wiki das páginas de sessão. O estado fica gravado.

```
nix brain status
```

Mostra se está ligado, qual é a wiki e o estado dos hooks deste projeto.

```
nix brain hooks install
```

Instala os hooks de captura. No terminal, pergunta o cliente e se o alcance é a conta ou só o repositório. Há prévia e confirmação. --scope, --agent, --all, --dry-run e --yes dispensam as perguntas.

```
nix brain hooks remove --scope ESCOPO
```

Tira os hooks da Nix daquele escopo. --agent escolhe o cliente.

```
nix brain hooks status
```

Mostra, por agente, se o arquivo e o consentimento estão alinhados.

```
nix brain handoff create --text "..."
```

Abre o handoff do projeto: onde a sessão parou. Só pode haver um aberto.

```
nix brain handoff list
```

Lista os handoffs do projeto, do mais recente ao mais antigo.

```
nix brain handoff accept --agent NOME
```

Reivindica o handoff aberto uma única vez e mostra o texto.

```
nix brain handoff cancel
```

Cancela o handoff aberto.

```
nix brain profile list
```

Lista as frases ativas do perfil.

```
nix brain profile rebuild
```

Reaprende o perfil a partir dos pedidos gravados, sem modelo de linguagem.

```
nix brain profile apply
```

Regrava na wiki as páginas das entradas ativas.

```
nix brain profile review
```

Mostra assuntos que ainda não apareceram em projetos suficientes.

```
nix brain profile forget CAMINHO
```

Esquece uma entrada. Ela não volta na próxima passada.

Agentes

```
nix run AGENTE
```

Sobe Cursor, VS Code (copilot), Claude Code (claude) ou Codex (codex). Instala o hook e o MCP do projeto se faltarem. --no-autowire não mexe nesses arquivos. --project escolhe o repositório. Para Claude Code e Codex, deixa o histórico nativo anterior em NIX_PREVIOUS_SESSION_HISTORY até o agente terminar.

O que dá para fazer.

Os casos abaixo usam só o que a Nix já faz hoje.

Abrir a interface

Execute nix. Se o comando não existir, feche o terminal, abra outro e tente de novo. nix ui abre a mesma interface quando a wiki já está cadastrada.

Cadastrar a primeira wiki

nix init grava a pasta e o modelo. A instalação não faz mais essa pergunta.

Achar uma ideia no editor

Com o MCP da Nix ligado, peça ao agente para buscar nas notas. A busca usa o índice da wiki.

Registrar uma nota no editor

Peça ao agente para criar ou atualizar a nota. A escrita passa pelo MCP e o índice daquela nota é atualizado na hora.

Trazer o que mudou fora do Nix

nix sync atualiza o índice. --dry-run mostra o que mudaria. --full reconstrói tudo, e é obrigatório depois de trocar o modelo.

Usar mais de uma wiki

nix wiki add Trabalho C:/Notas/Trabalho cadastra a segunda. nix wiki default Trabalho passa a usá-la quando o nome é omitido.

Descobrir por que não conecta

nix doctor confere o comando, a configuração e o índice.

Guardar memória entre agentes

nix brain enable --wiki Default liga o Mega Brain. nix brain hooks install --scope global --all faz os clientes chamarem o Nix a cada evento. O VS Code não tem arquivo global: a captura dele fica parcial.

Continuar no Codex o que estava no Claude

Os dois precisam do MCP da Nix, do Mega Brain ligado e da mesma pasta do projeto. No Claude, peça para deixar registrado onde a sessão parou: ele chama brain_handoff_create. Abra o Codex e diga para continuar o que estava fazendo no Claude: ele chama brain_handoff_accept com o agente codex e, se precisar do resto, brain_query ou brain_read_page. Sem o MCP, o mesmo texto sai no terminal com nix brain handoff create e nix brain handoff accept --agent codex. O hook, sozinho, só entrega no início da sessão um briefing com os últimos pedidos.

Deixar um recado para o outro agente

Com o Mega Brain ligado, brain_message_send coloca uma mensagem na fila do projeto. O outro agente pega com brain_message_pop, uma única vez.

Ler o que o Mega Brain já guardou

brain_query busca na wiki do Mega Brain e nas observações. brain_recent_pages lista as páginas de sessão. brain_read_page lê uma pelo caminho. O texto é histórico.

Reaprender o perfil

nix brain profile rebuild relê os pedidos gravados. list mostra as frases, forget esquece uma e apply regrava as páginas na wiki.

Corrigir uma configuração

nix config history mostra a alteração. nix config undo volta o valor anterior.

Painel, fila e perfil.

O Mega Brain fica desligado até nix brain enable --wiki NOME. O projeto é a raiz Git da pasta em que o agente trabalhou, ou a própria pasta se não houver repositório. O nome é o da pasta, ou o campo project de um .nix.toml.

Um cartão por projeto

O cartão aparece depois da primeira sessão capturada. Mostra quantas sessões existem, quais agentes passaram e a última atividade. Abrir o projeto mostra visão geral, agentes, tokens, sessões e conhecimento.

Recado entre agentes

A caixa de entrada não é o handoff. brain_message_send deixa um texto para um agente ou para qualquer um do projeto. O estado fica esperando. brain_message_pop entrega o mais antigo uma única vez e marca entregue. Cancelar tira da fila antes disso. A tela conserva os já entregues.

Preferência repetida

Sem modelo de linguagem. A Nix procura “sempre use”, “prefiro”, “nunca use” e o equivalente em inglês, seguidos de uma palavra. Ferramentas conhecidas vão para profile/tools/. O resto vai para profile/habits/. O assunto só entra com pelo menos dois projetos. A frase mais nova prevalece. nix brain profile rebuild reaprende. forget vira lápide e não volta.

A página da sessão vai para projects/<projeto>/sessions/ quando a sessão encerra. O handoff e o recado ficam na base. No início da sessão o briefing inclui o perfil ativo, salvo [profile] consume = false.

O agente não roda o comando.

Você pede em linguagem normal. O cliente chama uma ferramenta. As de wiki ficam sempre disponíveis. As brain_* só entram com o Mega Brain ligado; reinicie o cliente depois de ligar. Os nomes atuais são list_wikis e wiki_insights.

O mesmo efeito do terminal

```
sync_index
```

nix sync. Peça para sincronizar o índice ou trazer o que mudou fora do Nix.

```
index_status
```

nix status. Peça se o índice está atrasado.

```
list_wikis
```

nix wiki list. Peça quais wikis estão cadastradas.

```
brain_status
```

nix brain status. Peça como está o Mega Brain neste projeto.

```
brain_handoff_create
```

nix brain handoff create. Peça para registrar onde pararam.

```
brain_handoff_list
```

nix brain handoff list. Peça se há handoff aberto.

```
brain_handoff_accept
```

nix brain handoff accept. Peça para continuar o que estava fazendo no outro agente.

```
brain_handoff_cancel
```

nix brain handoff cancel. Peça para cancelar o handoff.

Só pelo agente

```
search_notes
```

Procura nas notas.

```
read_note
```

Abre a nota. get_note_outline e read_note_lines leem o outline ou um trecho.

```
list_notes
```

Lista as notas. list_all_tags lista as tags.

```
get_linked_notes
```

Mostra o que aponta para a nota.

```
create_note
```

Cria, e append_to_note, update_note, update_frontmatter e manage_tags alteram.

```
move_note
```

Move a nota. move_file move o anexo.

```
delete_note
```

Apaga a nota. Exige confirmação.

```
wiki_insights
```

Notas órfãs, duplicatas ou sugestão de links.

```
remember
```

Grava um fato duradouro na pasta de memória da wiki.

```
brain_query
```

Busca o que o Mega Brain guardou.

```
brain_read_page
```

Lê uma página. brain_recent_pages lista as recentes.

```
brain_briefing
```

Monta o briefing. brain_read_session_observations lê a sessão.

```
brain_write_page
```

Anota regra, decisão ou perfil. brain_patch_page e brain_delete_page alteram ou apagam.

```
brain_feedback
```

Registra se a página foi útil.

```
brain_message_send
```

Deixa um recado. brain_message_list, brain_message_pop e brain_message_cancel listam, pegam ou cancelam.

nix, nix init, nix doctor, nix ui, nix run, nix config, o restante de nix wiki, nix brain enable, nix brain hooks e nix brain profile ficam no terminal. O hook dispara sozinho.

Quando a empresa bloqueia o executável

Se o executável, o PowerShell ou os downloads do instalador estiverem bloqueados, use o código-fonte com Python no Git Bash. Essa opção não instala o nix.exe.

Baixe o código

Na última release , baixe Source code (zip). Se o GitHub estiver bloqueado nessa máquina, baixe em outra e copie o arquivo.

Descompacte e instale

Abra o Git Bash na pasta descompactada. Com Python 3.11 ou superior, execute:

```
python scripts/bootstrap.py
```

O script não pede a wiki. Ao terminar, execute nix. Se o comando não funcionar, feche o terminal, abra outro e tente de novo. O pip precisa alcançar o PyPI.

Registre o cliente MCP

Aponte o comando para o Python do ambiente criado e passe -P -m nix como argumentos. Ajuste o caminho à pasta onde você descompactou o projeto:

```
{ / "mcpServers": { / "nix": { / "command": "C:/Git/nix/.venv/Scripts/python.exe", / "args": ["-P",
"-m", "nix"] / } / } / }
```

Abra um novo terminal e execute nix doctor. Se python não existir no Git Bash, tente py ou python3.

Troque a versão sem apagar os dados.

Encerre nix e nix-hook, apague só esses dois binários e rode o instalador de novo. A wiki, o settings.db e a pasta .nix ficam. O desinstalador abaixo também apaga settings.db e .nix, salvo --keep-data.

Windows

No PowerShell, $root é NIX_HOME ou %LOCALAPPDATA%\nix.

```
Stop-Process -Name nix,nix-hook -Force -ErrorAction SilentlyContinue / $root = if ($env:NIX_HOME) {
$env:NIX_HOME } else { Join-Path $env:LOCALAPPDATA "nix" } / Remove-Item -Force (Join-Path $root
"bin\nix.exe"), (Join-Path $root "bin\nix-hook.exe") / irm
https://raw.githubusercontent.com/paulocesaaars/nix/main/install.ps1 | iex
```

Feche e abra de novo o cliente MCP para ele carregar o binário novo.

Linux ou macOS

O instalador substitui os arquivos. Encerre os processos e rode o instalador de novo.

```
pkill -x nix || true / pkill -x nix-hook || true / curl -fsSL
https://raw.githubusercontent.com/paulocesaaars/nix/main/install.sh | sh
```

Remova a instalação quando precisar.

Escolha o comando correspondente à forma como instalou. A pasta da wiki não é alterada. Para só trocar a versão, use a seção anterior.

Instalado pelo PowerShell

Baixe e execute o desinstalador no PowerShell:

```
irm https://raw.githubusercontent.com/paulocesaaars/nix/main/uninstall.ps1 -OutFile
"$env:TEMP\uninstall-nix.ps1" / & "$env:TEMP\uninstall-nix.ps1"
```

Responda s à confirmação. Se nix.exe ou nix-hook.exe estiver aberto, o desinstalador para antes de alterar o PATH e avisa que o arquivo está em uso. Encerre os processos e rode de novo:

```
Stop-Process -Name nix,nix-hook -Force -ErrorAction SilentlyContinue / &
"$env:TEMP\uninstall-nix.ps1"
```

Linux ou macOS

```
curl -fsSL https://raw.githubusercontent.com/paulocesaaars/nix/main/uninstall.sh | sh
```

Instalado a partir do código ou zip

Na pasta descompactada ou do clone, execute:

```
python scripts/uninstall.py
```

Use --yes para dispensar a confirmação e --keep-data para preservar settings.db e o índice. Abra um novo terminal e remova também o servidor Nix da configuração MCP do editor. O cache compartilhado do modelo pode permanecer.

Leve o guia de uso com você.

Baixe o guia de uso da Nix em PDF para consultar quando precisar.

Há mais para descobrir.

Os comandos e os casos de uso estão nesta página. O repositório guarda o detalhe de cada recurso.