CLICKUPFY

Guia completo do usuário

Planeje. Execute. Comprove.

Do ClickUp ao terminal e ao agente, com escopo, contexto e evidência.

Edição v1.2.3
Gerado em 20 de agosto de 2026
Fonte editorial: docs/user/

Sumário

Guia completo do usuário

O ClickUpfy conecta o trabalho mantido no ClickUp ao terminal e aos agentes de código. Um único executável configura vários perfis, percorre a hierarquia do ClickUp, administra tarefas, checklists, Sprints, comentários e time tracking, instala skills no projeto e oferece as mesmas operações por MCP.

Este guia ensina essa jornada em sequência. Você começa instalando o CLI e associando uma API key a um workspace. Depois encontra os IDs do projeto, executa uma tarefa completa pelo terminal e prepara um MCP restrito para que o agente trabalhe somente na List autorizada.

Leia online ou como ebook

Estas páginas também formam o **ClickUpfy — Guia completo do usuário**. A pasta do ebook publica a versão vigente em PDF e EPUB e registra os hashes que comprovam a origem comum dos formatos.

O PDF favorece leitura, compartilhamento e impressão. O EPUB se adapta ao tamanho de fonte e ao leitor digital. Os dois são reconstruídos sempre que uma página desta documentação muda.

Percurso pedagógico

Siga esta ordem na primeira leitura:

1. Entenda o ClickUpfy e os limites entre ClickUp, CLI, MCP e skills.
2. Instale o CLI pelo npm ou por um executável standalone.
3. Configure perfis e credenciais sem colocar a API key no repositório.
4. Consulte a referência de perfis e hierarquia sempre que precisar de parâmetro, efeito persistente ou variação de comando.
5. Prepare o primeiro projeto e confira o escopo antes de permitir escrita.
6. Navegue pela hierarquia até encontrar Space, Folder, List e Sprint Folder.
7. Trabalhe com tarefas e checklists, incluindo subtarefas, Markdown, datas e comentários.
8. Consulte a referência de tarefas, checklists, comentários e tempo para parâmetros, formatos e confirmações de escrita.
9. Planeje Sprints e acompanhe avanço por tarefas e Sprint Points.

10. Consulte a referência de Sprints para os parâmetros de relatório, associação e Sprint Points.
11. Registre tempo no item em execução.
12. Instale e use as skills que acompanham criação, implementação e release.
13. Conecte um agente por MCP com IDs fixos e opção read-only.
14. Consulte a referência de Docs, skills e servidor MCP para os recursos de documentação dentro do ClickUp e a integração no projeto.
15. Use a referência MCP de contexto e hierarquia para conhecer cada argumento aceito pelas ferramentas de navegação.
16. Consulte a referência MCP de Docs e administração para o workspace, Docs, páginas e ferramentas condicionais do servidor.
17. Consulte a referência MCP de trabalho para operações de tarefas, Sprints, checklists, comentários e tempo.
18. Consulte a referência do CLI para comandos, argumentos e formatos de saída.
19. Resolva falhas comuns de autenticação, escopo, status e integração.

O modelo mental em uma frase

O arquivo global guarda credenciais. Os arquivos do projeto guardam somente o perfil e os IDs autorizados. O `.mcp.json` atende clientes que usam JSON; o `.codex/config.toml` atende o Codex.

Essa divisão permite usar o mesmo ClickUpfy em vários projetos sem duplicar a API key. Um projeto pode apontar para a List de um produto, enquanto outro aponta para a List de um cliente. O MCP de cada raiz recusa IDs diferentes dos que foram fixados.

Comece por entender o papel de cada camada.

Entenda o ClickUpfy

O problema que a ferramenta resolve

O ClickUp continua sendo a fonte do trabalho: tarefa, descrição, status, checklist, comentário, Sprint e registro de tempo permanecem no workspace. O ClickUpfy não cria um gerenciador paralelo. Ele oferece uma interface previsível para consultar e alterar essa fonte pelo terminal ou por um agente.

Sem essa camada, cada agente precisa descobrir como autenticar, quais endpoints usar, em qual List trabalhar e como representar uma tarefa grande. O ClickUpfy centraliza essas decisões em três contratos:

- o perfil local associa uma API key a um workspace
- o projeto fixa o perfil e a hierarquia autorizada no `.mcp.json` ou no `.codex/config.toml`
- as skills descrevem quando ler, planejar, comentar, medir tempo e concluir.

As quatro camadas

ClickUp

É o sistema remoto e a fonte de verdade. O ClickUpfy usa a API pública para ler workspaces, Spaces, Folders, Lists, tarefas, checklists, comentários, Sprints e time entries. Permissões e ClickApps continuam sendo administrados no ClickUp.

CLI

É a interface humana e automatizável. A sintaxe segue `recurso ação`, como `task get`, `sprint current` e `comment create`. Sem `--json`, a saída é compacta. Com `--json`, o CLI preserva a resposta completa necessária para scripts e diagnóstico.

Servidor MCP

É a interface para agentes. Ele usa transporte stdio e expõe ferramentas com schemas explícitos. Uma configuração por projeto pode fixar account, workspace, Space, Folder, List e Sprint Folder. A List é obrigatória porque define o destino das consultas, buscas e criações de tarefa.

Skills

São instruções operacionais entregues junto ao pacote. Elas não substituem o MCP: orientam o agente a usar as ferramentas na ordem adequada, preservar contexto, registrar progresso e comprovar a conclusão.

Perfis e projetos são coisas diferentes

Um **account** do ClickUpfy é um perfil local. Ele contém um nome, a API key, o usuário autenticado e o workspace associado. O arquivo vive fora dos repositórios, em `~/clickupfy/config.json`.

Um **projeto** é uma raiz de código que pode conter `.mcp.json` e `.codex/config.toml`. Esses arquivos selecionam um account e fixam IDs da hierarquia no formato esperado por cada cliente. Eles podem ser versionados quando os IDs não forem considerados sensíveis pela equipe, pois nunca contêm a API key.

O mesmo account pode servir mais de um projeto. Ambientes de clientes ou workspaces diferentes devem receber perfis separados.

Leitura compacta e leitura executável

`task get` não retorna apenas a tarefa original. O ClickUpfy monta uma propriedade `execution` que achata a tarefa principal, subtarefas de qualquer nível e checklist items em uma fila ordenada.

Cada item informa:

- uma `key` estável para a leitura atual
- o tipo do item
- a relação com o pai
- a profundidade na árvore
- o estado aberto ou concluído
- a ação equivalente no CLI e no MCP.

Essa fila ajuda o agente a selecionar um item pendente sem perder a hierarquia. Quando o contexto precisa ser lido como documento, `task get --markdown` produz metadados, descrição, resumo e checkboxes em um único texto.

Limites importantes

O ClickUpfy não cria uma Sprint nem habilita o Sprint ClickApp, porque a API pública não oferece um endpoint próprio para essas operações. Crie o Sprint Folder e as Sprints na interface do ClickUp e use o CLI para consulta, associação de tarefas e Sprint Points.

O CLI também não decide sozinho qual status representa conclusão. Use `list get <list-id>` para ler os status configurados na List. Skills de implementação devem escolher um status terminal existente, não inventar uma grafia.

Agora siga para a instalação.

Instale o ClickUpfy

Requisitos

Para instalar pelo npm ou desenvolver, use Node.js `22.12.0` ou superior. Para usar um executável standalone, o Node.js não é necessário porque o runtime já está incorporado.

Você também precisa de uma API key pessoal do ClickUp com acesso a pelo menos um workspace. A chave só será usada na etapa de configuração.

Instale pelo npm

O pacote oficial instala o comando global:

```
npm install --global @promovaweb/clickupfy
clickupfy --version
clickupfy --help
```

O launcher seleciona o executável compilado em Linux x64, macOS Intel e macOS Apple Silicon. Em outras plataformas, inclusive Windows, usa o build JavaScript incluído no pacote.

Uma versão válida na primeira linha confirma que o pacote e o launcher foram encontrados. Se o terminal não localizar `clickupfy`, confira o diretório global do npm:

```
npm prefix --global
npm bin --global
```

O segundo comando pode não existir em versões novas do npm. Nesse caso, o diretório de executáveis costuma ser `bin` dentro do prefixo em Linux e macOS.

Atualize a instalação npm

Quando o ClickUpfy foi instalado globalmente pelo npm, o próprio comando inicia a instalação da versão nova e confirma o launcher que ficará disponível no terminal:

```
clickupfy upgrade
```

O alvo padrão é `latest`. Para testar um canal ou fixar uma versão, use `clickupfy upgrade next` ou `clickupfy upgrade 0.5.0`. O comando não substitui um executável standalone; nesse caso, baixe o archive correspondente na GitHub Release, confira `SHA256SUMS` e substitua o arquivo no `PATH`.

Instale um executável standalone

Cada GitHub Release distribui archives para Linux x64, macOS Intel, macOS Apple Silicon e Windows x64. Baixe o archive da plataforma, extraia `clickupfy` ou `clickupfy.exe` e mova o arquivo para uma pasta presente no `PATH`.

No Linux ou macOS:

```
chmod +x clickupfy
./clickupfy --version
```

Antes de usar um binário baixado, confira o arquivo `SHA256SUMS` da mesma release. O nome e a versão do archive precisam corresponder à plataforma escolhida.

No macOS, o Gatekeeper pode impedir a abertura de um executável sem notarização. Confirme a origem no repositório oficial e siga a política de segurança da máquina. Não desative proteções globais para contornar o aviso.

Instale para desenvolvimento

Clone o repositório independente do ClickUpfy e execute:

```
npm install
npm run build
npm link
clickupfy --version
```

`npm link` aponta o comando global para o checkout atual. Use esse modo apenas quando precisar desenvolver ou validar uma mudança local. Para voltar ao pacote publicado, remova o link e reinstale a versão do npm.

Confirme a instalação

Execute três leituras sem credenciais:

```
clickupfy --version
clickupfy --help
clickupfy agent skill list
```

O catálogo deve listar `clickupfy-dev`, `clickup-issue-create`, `clickup-issue-implement` e `clickupfy-release`. A presença das skills confirma que os assets foram empacotados junto ao CLI.

Ainda não execute comandos remotos. Primeiro configure um perfil.

Configure perfis e credenciais

Crie a API key

Use uma API key pessoal criada nas configurações do ClickUp. O ClickUpfy valida a chave antes de salvar, consulta o usuário autenticado e lista os workspaces permitidos.

Não cole a chave em issue, comentário, arquivo `.env` versionado, `.mcp.json`, `.codex/config.toml` ou comando compartilhado no histórico da equipe. Em um terminal interativo, prefira o setup com prompt oculto:

```
clickupfy install
```

Escolha um nome local para o perfil e selecione o workspace. O nome vira um identificador normalizado, como `promovaweb` ou `cliente-a`.

Automatize o setup quando necessário

Em uma automação isolada, o setup aceita parâmetros:

```
clickupfy install \
  --api-key "pk_..." \
  --name "Promovaweb" \
  --workspace "123456" \
  --non-interactive
```

Evite colocar a chave diretamente em um script. Leia o valor de um secret manager ou de uma variável protegida e apague o ambiente temporário depois da execução.

Entenda o arquivo local

Por padrão, a configuração fica em:

```
~/clickupfy/config.json
```

O diretório recebe permissão `0700` e o arquivo `0600`. A API key precisa estar no JSON porque o CLI autentica chamadas futuras. Essa proteção limita a leitura ao usuário local, mas não substitui os cuidados com backup, malware ou pastas sincronizadas.

O schema permite vários accounts:

```
{
  "version": 1,
  "activeAccount": "promovaweb",
  "accounts": {
    "promovaweb": {
      "name": "Promovaweb",
      "apiKey": "pk_...",
      "user": {
        "id": 123,
        "username": "Desenvolvedor"
      },
      "workspace": {
        "id": "456",
        "name": "Engenharia"
      },
      "createdAt": "2026-07-29T12:00:00.000Z",
      "updatedAt": "2026-07-29T12:00:00.000Z"
    }
  }
}
```

Gerencie accounts

Use comandos que mascaram a API key:

```
clickupfy account list
clickupfy account show
clickupfy status
clickupfy whoami
```

`status` mostra o caminho da configuração, o account resolvido e o workspace. `whoami` faz uma chamada autenticada e confirma se a chave ainda é válida.

Para conferir o setup local, as permissões do arquivo, o schema e os arquivos de integração do projeto, execute:

```
clickupfy doctor
```

Esse diagnóstico não faz uma chamada ao ClickUp. Use `clickupfy whoami` quando precisar validar a API key remotamente.

Para trocar o perfil ativo:

```
clickupfy account use cliente-a
```

Para usar outro account em um único comando, sem alterar o ativo:

```
clickupfy --account cliente-a task get 86abc123
```

A variável `PROMOVAWEB_CLICKUPFY_ACCOUNT` oferece a mesma seleção em automações. A variável `PROMOVAWEB_CLICKUPFY_CONFIG` aponta para outro JSON e é útil em testes ou ambientes efêmeros.

Troque o workspace associado

Liste os workspaces autorizados e escolha um deles:

```
clickupfy workspace list
clickupfy workspace use 123456
```

Essa ação altera o workspace do account atual. Configurações de projeto que fixam `--workspace` precisam continuar correspondendo ao perfil.

Remova um perfil

```
clickupfy account remove cliente-antigo
```

O CLI pede confirmação. Em automação, `--yes` confirma sem prompt. A remoção apaga o perfil local, não revoga a API key no ClickUp. Revogue a chave no ClickUp quando ela não for mais necessária.

Com o perfil validado, prepare o primeiro projeto.

Referência do CLI: perfis, autenticação e hierarquia

Opções globais

Todas as ações, exceto a configuração inicial, podem receber estas opções imediatamente depois de `clickupfy`:

OPÇÃO	TIPO	EFEITO
`--account <perfil>`	texto	Seleciona um perfil para esta execução sem trocar o perfil ativo.
`--json`	boolean	Imprime o payload completo em JSON, adequado para automação e inspeção.

O perfil ativo só muda com `account use`. A ordem importa: escreva `clickupfy --account cliente-a task list`, e não coloque `--account` depois do subcomando.

`clickupfy upgrade`

Atualiza a instalação global do ClickUpfy pelo npm e relê o launcher global depois da instalação. Sem alvo, instala `@promovaweb/clickupfy@latest`.

PARÂMETRO	OBRIGATÓRIO	USO
`[alvo]`	não	`latest`, `next` ou uma versão SemVer, como `0.5.0`. `latest` é o padrão.

O comando instala a nova versão com `npm install --global`, sem alterar a configuração de accounts, e confirma a versão retornada pelo launcher global. Ele exige o npm disponível no `PATH` e permissões para alterar o prefixo global. Não substitui executáveis standalone baixados da GitHub Release.

Exemplos:

```
clickupfy upgrade
```

```
clickupfy upgrade latest
```

```
clickupfy upgrade next
```

```
clickupfy upgrade 0.5.0
```

```
clickupfy upgrade v0.5.0
```

O quinto exemplo normaliza o prefixo `v` antes de chamar o npm. Se o npm falhar ou o launcher não retornar uma versão SemVer, o comando encerra com erro e não declara a atualização como concluída.

`clickupfy doctor`

Verifica o setup local sem fazer chamadas à API do ClickUp. O diagnóstico confere o caminho `~/clickupfy/config.json` ou o caminho definido por `PROMOVAWEB_CLICKUPFY_CONFIG`, as permissões `0700` do diretório e `0600` do arquivo, o JSON, o schema, os accounts e o account ativo. Quando executado na raiz de um projeto, também verifica o gerenciador `skills`, as quatro skills do ClickUpfy, o idioma dos arquivos instalados e lê `.mcp.json` e `.codex/config.toml` se existirem. API keys nunca aparecem na saída.

Os arquivos de projeto são opcionais. A ausência deles gera `skipped` porque o MCP só é necessário quando o projeto usa agentes. Um arquivo presente, mas inválido, gera `error`. O comando retorna código `1` quando existe uma verificação com erro. `warning` e `skipped` não reprovam o diagnóstico. Se o gerenciador `skills` não estiver instalado, ou se uma skill instalada estiver fora do padrão de português do Brasil, o diagnóstico registra essa situação para correção.

Exemplos:

```
clickupfy doctor
```

```
clickupfy --json doctor
```

```
PROMOVAWEB_CLICKUPFY_CONFIG=/tmp/clickupfy-config.json clickupfy doctor
```

```
cd ~/projetos/produto && clickupfy doctor
```

```
clickupfy --json doctor | jq '.checks[] | select(.estado == "error")'
```

O diagnóstico confirma a configuração local e os arquivos usados pelos agentes, mas não testa se a API key continua válida no ClickUp. Para essa verificação remota, use `clickupfy whoami`.

`clickupfy install`

Cria ou atualiza um perfil local e associa a API key a um workspace autorizado. Sem `--non-interactive`, o comando pergunta pelos valores ausentes. A API key fica no arquivo de configuração do usuário, cujo caminho aparece em `status`; ela não vai para o `.mcp.json` nem para o `.codex/config.toml` do projeto.

PARÂMETRO	OBRIGATÓRIO	USO
`--api-key <chave>`	não	API key pessoal do ClickUp. `--token` é um alias compatível.
`--name <nome>`	não	Nome legível do perfil local.
`--workspace <id>`	não	Workspace que será associado ao perfil.
`--non-interactive`	não	Recusa prompts; use junto dos dados necessários para automação.

Exemplos:

```
clickupfy install
```

```
clickupfy install --name "Produto"
```

```
clickupfy install --name "Cliente A" --workspace 123456
```

```
clickupfy install --api-key "$CLICKUP_API_KEY" --name "Automação"
```

```
clickupfy install \
  --api-key "$CLICKUP_API_KEY" \
  --name "Produto" \
  --workspace 123456 \
  --non-interactive
```

Quando a API key for inválida, execute `whoami` para conferir a autenticação. Quando o workspace informado não pertencer à chave, selecione um workspace da lista devolvida pelo ClickUp. O comando nunca deve aparecer em histórico de shell com uma chave escrita diretamente; prefira uma variável de ambiente temporária ou a entrada interativa.

`clickupfy status`

Mostra o caminho de configuração, o perfil resolvido, se ele está ativo, o usuário e o workspace associado. A chave aparece mascarada. Use este comando como inspeção local, não como teste definitivo de uma chave recém-revogada.

Exemplos:

```
clickupfy status
```

```
clickupfy --json status
```

```
clickupfy --account produto status
```

```
clickupfy --account cliente-a --json status
```

```
clickupfy status > /tmp/clickupfy-status.txt
```

O quarto exemplo consulta o perfil `cliente-a` sem alterar o perfil que será usado pela próxima chamada sem `--account`. A redireção do último exemplo é segura porque a saída mascara a chave, mas ainda pode conter nomes de pessoas e workspaces.

Perfis locais

Um perfil é um conjunto local de API key, identidade autenticada e workspace selecionado. Cada perfil tem um identificador, como `produto` ou `cliente-a`. O identificador é usado com `--account` e pelo `--account` do servidor MCP.

`clickupfy account list`

Lista todos os perfis e marca o perfil ativo com `*`. Não revela API keys. `clickupfy account ls` é o alias equivalente.

Exemplos:

```
clickupfy account list
```

```
clickupfy account ls
```

```
clickupfy --json account list
```

```
clickupfy --account produto account list
```

```
clickupfy account list | less
```

Mesmo com `--account`, a lista continua mostrando todos os perfis; a opção só tem efeito quando uma ação precisa resolver uma conta. Use a tabela para conferir o identificador exato antes de chamar `account show`, `account use` ou um comando de trabalho.

`clickupfy account show [perfil]`

Exibe metadados seguros de um perfil. Sem argumento, resolve o perfil ativo ou o que foi indicado pela opção global `--account`. O argumento posicional tem precedência sobre a opção global quando os dois aparecem.

PARÂMETRO	OBRIGATÓRIO	USO
`[perfil]`	não	Identificador local do perfil.

Exemplos:

```
clickupfy account show
```

```
clickupfy account show produto
```

```
clickupfy --account cliente-a account show
```

```
clickupfy --json account show produto
```

```
clickupfy --account cliente-a account show produto
```

No último caso, a saída é do perfil `produto`, pois o argumento posicional foi informado. Se o perfil não existir, o CLI interrompe a chamada sem iniciar uma requisição ao ClickUp.

`clickupfy account use <perfil>`

Define o perfil ativo no arquivo de configuração. É uma alteração local e não muda nada no ClickUp. Evite usar este comando em automações ou quando dois projetos usam contas diferentes ao mesmo tempo; nesses casos, informe `--account` em cada chamada ou fixe o perfil no MCP de cada projeto.

PARÂMETRO	OBRIGATÓRIO	USO
`<perfil>`	sim	Identificador local já configurado.

Exemplos:

```
clickupfy account use produto
```

```
clickupfy account use cliente-a
```

```
clickupfy account use homologacao
```

```
clickupfy account use suporte
```

```
clickupfy account use produto && clickupfy status
```

O último exemplo confirma a mudança persistida. Quando o comando faz parte de um script, prefira não depender do perfil ativo: `clickupfy --account produto ...` deixa o destino explícito e não interfere no restante da máquina.

`clickupfy account remove <perfil>`

Remove somente o perfil local. Não revoga a API key no ClickUp, não remove usuários e não exclui tarefas. Em terminal interativo, pede confirmação. Em automação, `--yes` confirma a remoção sem prompt.

PARÂMETRO	OBRIGATÓRIO	USO
`<perfil>`	sim	Identificador local a remover.
`--yes`	não	Confirma sem interação.

Exemplos:

```
clickupfy account remove conta-antiga
```

```
clickupfy account remove sandbox --yes
```

```
clickupfy account remove cliente-encerrado
```

```
clickupfy account remove homologacao --yes
```

```
clickupfy account remove temporario --yes && clickupfy account list
```

Se você remover o perfil ativo, o ClickUpfy torna ativo o primeiro perfil que sobrar ou deixa a configuração sem perfil ativo quando não houver outro. Leia `account list` após a remoção, especialmente em uma máquina compartilhada.

Workspace e identidade autenticada

O perfil tem um workspace associado, mas uma API key pode ter permissão para mais de um workspace. `workspace list` descobre as opções; `workspace use` persiste a escolha no perfil; `whoami` consulta a API e confirma o usuário.

`clickupfy workspace list`

Lista os workspaces que a API key do perfil consegue acessar. O marcador `*` indica o workspace atualmente associado ao perfil. `workspace ls` é um alias.

Exemplos:

```
clickupfy workspace list
```

```
clickupfy workspace ls
```

```
clickupfy --account produto workspace list
```

```
clickupfy --account cliente-a --json workspace list
```

```
clickupfy workspace list | tee /tmp/workspaces.txt
```

O resultado autoritativo vem da API. Caso o workspace esperado não apareça, verifique a permissão da API key no ClickUp em vez de tentar adivinhar um ID.

`clickupfy workspace use <workspace-id>`

Associa um workspace autorizado ao perfil resolvido. O CLI consulta a API e só salva o ID quando ele aparece na lista permitida, portanto um número copiado de outro perfil é recusado.

PARÂMETRO	OBRIGATÓRIO	USO
`<workspace-id>`	sim	ID devolvido por `workspace list`.

Exemplos:

```
clickupfy workspace use 123456
```

```
clickupfy --account cliente-a workspace use 987654
```

```
clickupfy --account produto workspace use 123456
```

```
clickupfy workspace use 654321 && clickupfy status
```

```
clickupfy --account suporte workspace use 246810
```

A alteração afeta o perfil indicado, não todos os perfis. Um servidor MCP com `--workspace` fixo ainda recusará um workspace diferente, mesmo que o perfil local seja alterado depois.

`clickupfy whoami`

Valida a API key em uma chamada ao ClickUp e devolve o usuário autenticado junto do perfil e do workspace resolvidos. Use-o depois de `setup`, após uma troca de chave ou ao investigar erro de autorização.

Exemplos:

```
clickupfy whoami
```

```
clickupfy --json whoami
```

```
clickupfy --account produto whoami
```

```
clickupfy --account cliente-a --json whoami
```

```
clickupfy whoami && clickupfy workspace list
```

Uma resposta de `status` não substitui este teste: ela lê a configuração local. `whoami` é a chamada que confirma se a chave ainda funciona no serviço remoto.

Descoberta da hierarquia

O ClickUp organiza trabalho em workspace, Space, Folder e List. Uma List pode existir diretamente dentro de um Space, sem Folder. IDs não são intercambiáveis: um `space-id` não serve no lugar de `folder-id`, por exemplo. Comece no workspace selecionado e preserve os IDs retornados em um local seguro.

`clickupfy space list`

Lista Spaces do workspace associado ao perfil. `--archived` acrescenta Spaces arquivados; sem a opção, o resultado contém apenas os ativos.

PARÂMETRO	OBRIGATÓRIO	USO
`--archived`	não	Inclui Spaces arquivados.

Exemplos:

```
clickupfy space list
```

```
clickupfy space ls
```

```
clickupfy space list --archived
```

```
clickupfy --account produto --json space list
```

```
clickupfy --account cliente-a space list --archived
```

Escolha o Space pelo ID e pelo nome retornados, não pela posição da linha. Spaces arquivados podem ser úteis em migração ou consulta histórica, mas não devem ser usados como destino de novas tarefas sem autorização explícita.

`clickupfy folder list --space <id>`

Lista Folders de um Space. O parâmetro `--space` é obrigatório; o CLI não usa um Space implícito porque um perfil pode operar em vários projetos.

PARÂMETRO	OBRIGATÓRIO	USO
`--space <id>`	sim	ID do Space retornado por `space list`.
`--archived`	não	Inclui Folders arquivados.

Exemplos:

```
clickupfy folder list --space 1001
```

```
clickupfy folder ls --space 1001
```

```
clickupfy folder list --space 1001 --archived
```

```
clickupfy --account produto --json folder list --space 1001
```

```
clickupfy --account cliente-a folder list --space 2001 --archived
```

Uma resposta vazia não significa necessariamente falta de acesso: o Space pode guardar Lists diretamente. Execute `list list --space <id>` para cobrir esse ramo da hierarquia.

`clickupfy list list --folder <id> | --space <id>`

Lista Lists dentro de um Folder ou Lists que pertencem diretamente a um Space. Informe uma das duas opções. O CLI exige pelo menos uma; quando ambas forem fornecidas, a chamada deve apontar para o nível real que você pretende ler, mantendo um único destino por consulta.

PARÂMETRO	OBRIGATÓRIO	USO
`--folder <id>`	condicional	ID do Folder que contém as Lists.
`--space <id>`	condicional	ID do Space para Lists sem Folder.
`--archived`	não	Inclui Lists arquivadas.

Exemplos:

```
clickupfy list list --folder 2001
```

```
clickupfy list ls --folder 2001
```

```
clickupfy list list --folder 2001 --archived
```

```
clickupfy list list --space 1001
```

```
clickupfy --account produto --json list list --space 1001 --archived
```

Consulte cada List pelo comando seguinte para obter os status configurados. Em projetos com MCP, o ID da List escolhido aqui será o valor obrigatório de `agent install --list` ou de `mcp serve --list`.

`clickupfy list get <list-id>`

Obtém a List e os status aceitos pelas tarefas dela. A grafia retornada para um status é a mesma que deve aparecer em `task create --status` e `task update --status`; não use uma tradução ou suposição local.

PARÂMETRO	OBRIGATÓRIO	USO
`<list-id>`	sim	ID da List retornado por `list list`.

Exemplos:

```
clickupfy list get 3001
```

```
clickupfy --json list get 3001
```

```
clickupfy --account produto list get 3001
```

```
clickupfy --account cliente-a --json list get 4001
```

```
clickupfy list get 3001 > /tmp/list-3001.json
```

O último exemplo deve incluir `--json` quando o arquivo for consumido por outro programa. A tabela compacta é adequada para leitura, mas não é uma API estável de máquina. Continue em tarefas e checklists para usar os status e IDs encontrados aqui.

Prepare o primeiro projeto

O que você vai preparar

Neste percurso, um repositório será ligado a uma List do ClickUp. O resultado são `.mcp.json` e `.codex/config.toml`, cada um no formato do cliente que os consome, iniciando o ClickUpfy com um account e IDs fixos, além das quatro skills instaladas no projeto.

Antes de escrever qualquer tarefa, você vai:

1. validar o perfil
2. descobrir Space, Folder e List
3. instalar o MCP em read-only para conferir o destino
4. liberar escrita somente depois da revisão.

Valide o perfil

```
clickupfy status
clickupfy whoami
```

Confirme o nome do account, o usuário e o workspace. Se qualquer dado estiver errado, volte à configuração.

Descubra a hierarquia

```
clickupfy workspace list
clickupfy space list
clickupfy folder list --space <space-id>
clickupfy list list --folder <folder-id>
```

Uma List pode pertencer diretamente ao Space. Nesse caso:

```
clickupfy list list --space <space-id>
```

Guarde os IDs confirmados. Se o projeto usa Sprints, localize também o Sprint Folder que contém as Lists temporais.

Inicialize o agente

Na raiz do repositório:

```
clickupfy --account promovaweb agent install \
  --workspace 123 \
  --space 10 \
  --folder 20 \
  --list 30
```

`space` e `list` são obrigatórios. Workspace e Folder podem ser omitidos quando não forem necessários ao escopo. Acrescente `--sprint-folder <id>` se o projeto usa Sprints.

O comando instala as skills e mescla o servidor no `.mcp.json` e no `.codex/config.toml` existentes sem remover outros servidores ou configurações. No JSON, a entrada fica assim:

```
{
  "mcpServers": {
    "promovaweb-clickupfy": {
      "command": "clickupfy",
      "args": [
        "mcp",
        "serve",
        "--account",
        "promovaweb",
        "--workspace",
        "123",
        "--space",
        "10",
        "--folder",
        "20",
        "--list",
        "30"
      ]
    }
  }
}
```

No Codex, a entrada equivalente fica na tabela `mcp_servers`:

```
[mcp_servers."promovaweb-clickupfy"]
command = "clickupfy"
args = ["mcp", "serve", "--account", "promovaweb", "--workspace", "123", "--space",
"10", "--folder", "20", "--list", "30"]
```

Antes de abrir o cliente MCP pela primeira vez, acrescente `--read-only` ao fim de `args`. O `agent install` prepara o servidor completo. Essa edição consciente reduz a superfície durante a conferência inicial.

Confira no agente

Reinicie ou recarregue o cliente MCP para que ele leia a nova configuração. Peça uma consulta ao contexto:

```
Use clickupfy_mcp_context e mostre o perfil, o workspace e a hierarquia deste projeto.
```

Compare o resultado com os IDs anotados. Depois liste as tarefas:

```
Use clickupfy_tasks_list sem informar listId.
```

O servidor deve aplicar a List configurada. Uma tentativa de consultar outra List deve ser recusada.

Libere escrita

Quando o destino estiver correto, remova apenas `--read-only` da entrada gerenciada. Reabra o cliente MCP e confirme que as ferramentas de escrita aparecem em `tools/list`.

Crie uma tarefa de teste somente se o projeto e a equipe autorizarem a mutação. Caso contrário, use uma tarefa real já existente para validar leitura, comentários e checklists.

Primeira leitura de uma tarefa

No terminal:

```
clickupfy task get <task-id> --markdown
```

No agente:

```
Leia <task-id> com clickupfy_task_get usando markdown: true. Não altere a tarefa.
```

Confira descrição, subtarefas, checklists e fila executável. Agora você já pode aprofundar a hierarquia ou começar a trabalhar com tarefas.

Navegue pela hierarquia do ClickUp

A ordem dos recursos

O ClickUp organiza o trabalho nesta cadeia:

```
Workspace
└── Space
    ├── List sem Folder
    └── Folder
        └── List
```

Sprints também são Lists, mas vivem normalmente dentro de um Sprint Folder e possuem `start_date` e `due_date`.

O account já conhece um workspace. Por isso, `space list` não exige esse ID. A partir do Space, os comandos precisam do pai para evitar uma busca ambígua.

Liste workspaces e Spaces

```
clickupfy workspace list
clickupfy space list
clickupfy space list --archived
```

O asterisco na saída compacta indica o workspace associado ao account. Use `--archived` somente quando precisar localizar uma estrutura antiga.

Liste Folders e Lists

```
clickupfy folder list --space <space-id>
clickupfy list list --folder <folder-id>
```

Para Lists que pertencem diretamente ao Space:

```
clickupfy list list --space <space-id>
```

Folder e Space são alternativas no comando de Lists. Não informe os dois ao mesmo tempo.

Confira os status da List

```
clickupfy list get <list-id>
```

Essa leitura é importante antes de atualizar uma tarefa. Os nomes e tipos de status são configuráveis no ClickUp. Uma equipe pode usar `feito`, outra `concluído`, e uma terceira pode manter mais de um estado terminal.

Registre o ID da List e escolha um status terminal existente. Não assuma que `done` ou `complete` será aceito.

Liste e busque tarefas

```
clickupfy task list --list <list-id>
clickupfy task search --query "autenticação"
```

`task list` consulta uma List conhecida. `task search` procura no workspace, mas o MCP de projeto restringe a busca à List fixada para impedir vazamento de contexto entre projetos.

Use `--json` quando precisar inspecionar campos não exibidos na tabela:

```
clickupfy --json task list --list <list-id>
```

A opção global vem antes do grupo do comando.

Escolha o menor escopo suficiente

Um MCP de projeto sempre exige List. Space e Folder ajudam as ferramentas de navegação, mas não ampliam o destino de criação. Sprint Folder só deve ser configurado quando as Sprints do projeto estiverem dentro dele.

Para um projeto com List diretamente no Space:

```
clickupfy --account produto agent install \
  --space 10 \
  --list 30
```

Para um projeto com Folder e Sprints:

```
clickupfy --account produto agent install \
  --space 10 \
  --folder 20 \
  --list 30 \
  --sprint-folder 40
```

Continue em tarefas e checklists.

Trabalhe com tarefas, subtarefas e checklists

Leia antes de alterar

Comece pela tarefa completa:

```
clickupfy task get <task-id> --markdown
```

Esse formato reúne metadados, descrição, subtarefas e checklist items em um documento. Para automação estruturada, use:

```
clickupfy task get <task-id> --json
```

Para obter somente a resposta original da API, sem a fila `execution`:

```
clickupfy task get <task-id> --raw
```

`--markdown`, `--json` e `--raw` são mutuamente exclusivos.

Entenda a fila executável

O objeto `execution` transforma a árvore em itens ordenados. Uma chave como `task:86abc123` identifica uma tarefa ou subtarefa. Uma chave como `checklist:check-1:item-1` identifica um item de checklist.

O resumo informa totais abertos e concluídos. Cada item inclui `parentKey` e `depth`, por isso um agente consegue trabalhar em uma subtarefa aninhada sem tratá-la como item independente.

A fila é uma projeção da leitura atual, não um segundo estado. Sempre releia a tarefa depois de uma mutação.

Crie uma tarefa

Use Markdown na descrição quando o ClickUp ClickApp correspondente estiver disponível:

```
clickupfy task create \
  --list <list-id> \
  --name "Implementar autenticação" \
  --markdown-content "Adicionar o fluxo de login e os testes." \
  --start-date 2026-08-01 \
  --due-date 2026-08-05
```

Datas aceitam o formato `AAAA-MM-DD`. Antes de criar, confira o fuso e a política de datas da equipe.

Para criar uma subtarefa, informe o pai:

```
clickupfy task create \
  --list <list-id> \
  --name "Criar testes de autenticação" \
  --parent <task-id> \
  --markdown-content "Cobrir sucesso, credencial inválida e limite."
```

Subtarefas podem receber novas subtarefas. A leitura executável preserva todos os níveis.

Monte checklists verificáveis

Crie o checklist na tarefa e depois os itens:

```
clickupfy checklist create <task-id> --name "Validação"
clickupfy checklist item-create <checklist-id> --name "Executar testes unitários"
clickupfy checklist item-create <checklist-id> --name "Executar build"
```

Os itens são criados abertos. Marque um item somente depois de executar sua verificação:

```
clickupfy checklist set \
  <task-id> <checklist-id> <item-id> \
  --resolved
```

Para reabrir:

```
clickupfy checklist set \
  <task-id> <checklist-id> <item-id> \
  --open
```

O ClickUpfy confirma primeiro que o item pertence à tarefa, grava o novo estado e relê a tarefa. O comando só comunica sucesso quando a API devolve o valor esperado.

Atualize campos

```
clickupfy task update <task-id> --status "em andamento"
clickupfy task update <task-id> --priority 2
clickupfy task update <task-id> --start-date 2026-08-01
clickupfy task update <task-id> --due-date 2026-08-05
clickupfy task update <task-id> --points 5
```

As prioridades seguem a API do ClickUp:

VALOR	PRIORIDADE
`1`	urgente
`2`	alta
`3`	normal
`4`	baixa

Consulte `list get` antes de definir status. Para remover datas:

```
clickupfy task update <task-id> --clear-start-date
clickupfy task update <task-id> --clear-due-date
```

Registre progresso em comentários

```
clickupfy comment list --task <task-id>
clickupfy comment create \
  --task <task-id> \
  --text "Testes focais passaram. Iniciando a regressão."
```

Um comentário útil registra um evento verificável: início, impedimento, checkpoint de teste ou conclusão. Evite publicar raciocínio interno, secrets, logs extensos ou promessas sem evidência.

Exclua apenas com alvo confirmado

```
clickupfy task delete <task-id> --yes
```

A exclusão é permanente na interface do CLI e exige confirmação. Leia a tarefa, confira List e nome e confirme se a solicitação autoriza exclusão antes de usar `--yes`.

Quando a equipe trabalha por ciclos, continue em Sprints e Sprint Points.

Referência do CLI: tarefas, checklists, comentários e tempo

Leitura de tarefas

`clickupfy task list --list <id>`

Lista tarefas de uma List em uma página. `--list` é obrigatório porque o CLI não lê o `.mcp.json` do projeto. `--status` aceita um ou mais nomes de status; `--assignee` recebe um ou mais IDs de responsáveis; `--page` começa em zero.

PARÂMETRO	OBRIGATÓRIO	USO
`--list <id>`	sim	List consultada.
`--status <status...>`	não	Restringe aos status informados.
`--assignee <ids...>`	não	Restringe aos responsáveis.
`--include-closed`	não	Inclui tarefas concluídas.
`--page <n>`	não	Página, iniciando em `0`.

Exemplos:

```
clickupfy task list --list 3001
```

```
clickupfy task list --list 3001 --status "em andamento"
```

```
clickupfy task list --list 3001 --status aberta "em andamento" --assignee 42
```

```
clickupfy task list --list 3001 --include-closed --page 0
```

```
clickupfy --account produto --json task list --list 3001 --page 2
```

A saída comum é uma tabela com ID, nome, status, prioridade, pontos, pessoas e link. Use `--json` quando um script precisar de todos os campos retornados pela API. Para encontrar uma tarefa fora de uma List conhecida, use `task search`.

`clickupfy task get <task-id>`

Obtém uma tarefa e prepara uma fila chamada `execution`, que inclui tarefa, subtarefas e checklist items em uma ordem estável. Sem opção de formato, imprime uma tabela compacta da fila. `--json` mostra a estrutura; `--markdown` junta metadados, descrição e itens em um documento; `--raw` mostra apenas a resposta original da API. Escolha exatamente um desses formatos.

PARÂMETRO	OBRIGATÓRIO	USO
`<task-id>`	sim	Tarefa raiz a consultar.
`--raw`	não	Retorna somente a resposta original do ClickUp.
`--markdown`	não	Renderiza descrição e fila em Markdown.
`--json`	não	Opção global que devolve tarefa e fila estruturada.

Exemplos:

```
clickupfy task get 86abc123
```

```
clickupfy task get 86abc123 --markdown
```

```
clickupfy task get 86abc123 --raw
```

```
clickupfy --json task get 86abc123
```

```
clickupfy --account produto task get 86abc123 --markdown
```

`execution.items` usa `task:<id>` para tarefas e subtarefas e `checklist:<checklist-id>:<item-id>` para itens de checklist. `parentKey` e `depth` preservam a relação de parentesco. Não tente combinar `--raw`, `--markdown` e `--json`: o CLI recusa a ambiguidade de formato.

`clickupfy task search`

Busca tarefas em todo o workspace associado ao perfil. É uma consulta paginada, limitada por `--max-pages`, que começa em uma e aceita no máximo cem páginas. O texto de `--query` procura nome, descrição ou ID.

PARÂMETRO	OBRIGATÓRIO	USO
`--query <texto>`	não	Termo de busca no nome, descrição ou ID.

PARÂMETRO	OBRIGATÓRIO	USO
`--status <status...>`	não	Filtra por status.
`--assignee <ids...>`	não	Filtra por responsáveis.
`--include-closed`	não	Inclui tarefas concluídas.
`--max-pages <n>`	não	Limite entre `1` e `100`; o padrão é `100`.

Exemplos:

```
clickupfy task search --query "autenticação"
```

```
clickupfy task search --query 86abc123
```

```
clickupfy task search --query "API" --status "em andamento"
```

```
clickupfy task search --assignee 42 --include-closed --max-pages 5
```

```
clickupfy --account cliente-a --json task search --query "checkout" --max-pages 10
```

Se mais de uma tarefa corresponder ao pedido, apresente os IDs e nomes para a pessoa escolher a correta antes de executar uma escrita. Em MCP de projeto, use `clickupfy_tasks_search`: essa ferramenta é limitada à List fixa, enquanto este comando percorre o workspace inteiro.

Criação e atualização de tarefas

`clickupfy task create`

Cria tarefa ou subtarefa na List indicada. `--list` e `--name` são obrigatórios. Use `--description` para texto simples ou `--markdown-content` quando a descrição precisa preservar Markdown. O CLI não impõe uma regra de prioridade, status ou pontos do seu time; ele transmite valores válidos para a API.

PARÂMETRO	OBRIGATÓRIO	USO
`--list <id>`	sim	List de destino.
`--name <nome>`	sim	Nome da tarefa.
`--description <texto>`	não	Descrição em texto.

PARÂMETRO	OBRIGATÓRIO	USO
`--markdown-content <markdown>`	não	Descrição em Markdown.
`--status <status>`	não	Status inicial existente na List.
`--priority <n>`	não	`1` urgente, `2` alta, `3` normal, `4` baixa.
`--assignee <ids...>`	não	IDs numéricos das pessoas responsáveis.
`--parent <task-id>`	não	Cria como subtarefa da tarefa informada.
`--start-date <AAAA-MM-DD>`	não	Data de início.
`--due-date <AAAA-MM-DD>`	não	Data de entrega.
`--points <n>`	não	Sprint Points iguais ou maiores que zero.

Exemplos:

```
clickupfy task create --list 3001 --name "Corrigir retorno da API"
```

```
clickupfy task create \
  --list 3001 \
  --name "Documentar autenticação" \
  --description "Explicar o fluxo de login."
```

```
clickupfy task create \
  --list 3001 \
  --name "Criar testes" \
  --markdown-content "## Casos\n\n- Login válido\n- Senha inválida"
```

```
clickupfy task create \
  --list 3001 \
  --name "Implementar refresh token" \
  --status "em andamento" \
  --priority 2 \
  --assignee 42 57 \
  --start-date 2026-08-10 \
  --due-date 2026-08-14 \
  --points 5
```

```
clickupfy --account produto task create \
  --list 3001 \
  --parent 86abc123 \
  --name "Cobrir expiração de sessão" \
  --markdown-content "Adicionar casos de teste para sessão expirada."
```

Consulte `clickupfy list get <list-id>` para copiar o status exato. Depois da criação, guarde o ID devolvido e chame `task get` para confirmar pai, datas, responsáveis e conteúdo. Não use `--parent` para mover uma tarefa existente; ele só define o pai na criação.

`clickupfy task update <task-id>`

Atualiza campos de uma tarefa existente. Pelo menos um campo é obrigatório. `--clear-start-date` e `--clear-due-date` removem as respectivas datas; quando uma flag de remoção e uma nova data aparecem juntas, a remoção prevalece.

PARÂMETRO	OBRIGATÓRIO	USO
`<task-id>`	sim	Tarefa a alterar.
`--name <nome>`	não	Novo nome.
`--description <texto>`	não	Nova descrição textual.
`--markdown-content <markdown>`	não	Nova descrição em Markdown.
`--status <status>`	não	Status existente na List.
`--priority <n>`	não	Prioridade de `1` a `4`.
`--start-date <AAAA-MM-DD>`	não	Nova data de início.
`--clear-start-date`	não	Remove a data de início.
`--due-date <AAAA-MM-DD>`	não	Nova data de entrega.
`--clear-due-date`	não	Remove a data de entrega.

PARÂMETRO	OBRIGATÓRIO	USO
`--points <n>`	não	Sprint Points não negativos.

Exemplos:

```
clickupfy task update 86abc123 --status "em andamento"
```

```
clickupfy task update 86abc123 --name "Corrigir autenticação por token"
```

```
clickupfy task update 86abc123 --priority 1 --points 8
```

```
clickupfy task update 86abc123 --start-date 2026-08-10 --due-date 2026-08-14
```

```
clickupfy --account produto task update 86abc123 --markdown-content "## Entrega\n\nPublicar a documentação atualizada." --clear-due-date
```

`--description` e `--markdown-content` representam alternativas de conteúdo; envie apenas a forma que você quer persistir. Uma atualização de status precisa usar a grafia devolvida por `list get` . Releia a tarefa depois da escrita e compare cada campo solicitado com o estado devolvido.

`clickupfy task delete <task-id>`

Exclui permanentemente uma tarefa. Em terminal interativo, pede confirmação; em automação, exige `--yes` . Leia nome, List e relação com Sprint antes de executar. A exclusão não é uma forma de retirar a tarefa de uma Sprint: use `sprint remove-task` para remover apenas a associação.

PARÂMETRO	OBRIGATÓRIO	USO
`<task-id>`	sim	Tarefa a excluir.
`--yes`	não	Confirma a exclusão sem prompt.

Exemplos:

```
clickupfy task delete 86abc123
```

```
clickupfy task delete 86abc123 --yes
```

```
clickupfy --account produto task delete 86abc123
```

```
clickupfy --account cliente-a task delete 77def456 --yes
```

```
clickupfy task get 86abc123 && clickupfy task delete 86abc123 --yes
```

O último exemplo lê a tarefa no mesmo terminal, mas ainda exige que você compare o resultado e tenha autorização explícita. No MCP, a ferramenta equivalente exige o campo literal `confirm: true`.

Checklists e comentários

`clickupfy checklist create <task-id> --name <nome>`

Cria um checklist vazio em uma tarefa. Os itens são criados pelo comando seguinte. Use um nome que indique a finalidade, como `Testes`, `Publicação` ou `Revisão manual`.

PARÂMETRO	OBRIGATÓRIO	USO
`<task-id>`	sim	Tarefa proprietária.
`--name <nome>`	sim	Nome do checklist.

Exemplos:

```
clickupfy checklist create 86abc123 --name "Testes"
```

```
clickupfy checklist create 86abc123 --name "Revisão de segurança"
```

```
clickupfy checklist create 77def456 --name "Publicação"
```

```
clickupfy --account produto checklist create 86abc123 --name "Aceite"
```

```
clickupfy --json checklist create 86abc123 --name "Regressão"
```

Guarde o ID retornado: `item-create` usa o ID do checklist, não o ID da tarefa. Crie o checklist na tarefa que realmente possui os itens; uma subtarefa pode ter checklist próprio e não deve receber os itens da tarefa pai por acidente.

`clickupfy checklist item-create <checklist-id> --name <nome>`

Acrescenta um item aberto a um checklist. `--assignee` é opcional e recebe um ID numérico. O comando não aceita o ID da tarefa porque o checklist já define seu proprietário.

PARÂMETRO	OBRIGATÓRIO	USO
`<checklist-id>`	sim	Checklist que receberá o item.
`--name <nome>`	sim	Texto do item.
`--assignee <id>`	não	Pessoa responsável pelo item.

Exemplos:

```
clickupfy checklist item-create check-1 --name "Executar testes unitários"
```

```
clickupfy checklist item-create check-1 --name "Executar build"
```

```
clickupfy checklist item-create check-1 --name "Revisar documentação" --assignee 42
```

```
clickupfy --account produto checklist item-create check-2 --name "Conferir changelog"
```

```
clickupfy --json checklist item-create check-3 --name "Validar publicação" --assignee 57
```

Um item deve descrever uma verificação observável, não uma promessa genérica. Depois de criar itens, execute `task get --json` para localizar o `item-id` e confirmar que todos começam abertos.

`clickupfy checklist set <task-id> <checklist-id> <item-id>`

Marca um item como concluído com `--resolved` ou o reabre com `--open`. Escolha exatamente uma flag. O ClickUpfy primeiro confirma que o item pertence à tarefa raiz informada, grava o estado e relê a tarefa para confirmar o resultado.

PARÂMETRO	OBRIGATÓRIO	USO
`<task-id>`	sim	Tarefa raiz usada para validar a associação.
`<checklist-id>`	sim	Checklist proprietário.
`<item-id>`	sim	Item a alterar.
`--resolved`	condicional	Marca como concluído.
`--open`	condicional	Reabre o item.

Exemplos:

```
clickupfy checklist set 86abc123 check-1 item-1 --resolved
```

```
clickupfy checklist set 86abc123 check-1 item-1 --open
```

```
clickupfy --account produto checklist set 86abc123 check-2 item-4 --resolved
```

```
clickupfy --json checklist set 77def456 check-3 item-2 --resolved
```

```
clickupfy checklist set 77def456 check-3 item-2 --open
```

Não use `--resolved --open` nem omita ambas: o CLI recusa as duas situações. Um `item-id` de outro checklist também é recusado pela releitura. Marque o item somente depois da validação correspondente.

`clickupfy comment list --task <id>`

Lista os comentários de uma tarefa com autoria, data e texto. A API pode representar o conteúdo em formatos diferentes; o ClickUpfy o normaliza para a coluna `text` na tabela compacta.

Exemplos:

```
clickupfy comment list --task 86abc123
```

```
clickupfy comment ls --task 86abc123
```

```
clickupfy --json comment list --task 86abc123
```

```
clickupfy --account produto comment list --task 86abc123
```

```
clickupfy --account cliente-a --json comment list --task 77def456
```

Leia os comentários antes de atualizar status ou publicar outro progresso, pois podem conter uma orientação nova. Nunca publique credenciais, dados pessoais, URLs assinadas ou logs extensos em um comentário.

`clickupfy comment create --task <id> --text <texto>`

Publica um comentário em uma tarefa. `--notify-all` pede ao ClickUp que notifique todos os participantes. Use o comando para registrar início, resultado de teste, mudança de escopo ou conclusão, sempre com informação verificável.

PARÂMETRO	OBRIGATÓRIO	USO
`--task <id>`	sim	Tarefa que receberá o comentário.
`--text <texto>`	sim	Conteúdo do comentário.
`--notify-all`	não	Notifica participantes da tarefa.

Exemplos:

```
clickupfy comment create --task 86abc123 --text "Iniciei a análise da tarefa."
```

```
clickupfy comment create --task 86abc123 --text "Os testes unitários passaram."
```

```
clickupfy comment create --task 86abc123 --text "Aguardando acesso ao ambiente de homologação." --notify-all
```

```
clickupfy --account produto comment create --task 86abc123 --text "Documentação e ebook foram atualizados."
```

```
clickupfy --json comment create --task 77def456 --text "Validação final concluída." --notify-all
```

Reler `comment list` confirma que o comentário ficou na tarefa pretendida e que o texto chegou completo. O comando não modifica status, timer ou checklist.

Time tracking

`clickupfy time current`

Consulta o time entry em execução no workspace associado ao perfil. Chame este comando antes de `time start` para não iniciar outro registro sem saber qual tarefa já está em andamento.

Exemplos:

```
clickupfy time current
```

```
clickupfy --json time current
```

```
clickupfy --account produto time current
```

```
clickupfy --account cliente-a --json time current
```

```
clickupfy time current && clickupfy task get 86abc123
```

Se houver entrada ativa para outra tarefa, pare e peça orientação antes de usar `time stop`. O comando é somente leitura e não altera status ou comentários.

`clickupfy time start --task <id>`

Inicia um time entry associado à tarefa. A descrição é opcional e deve dizer o trabalho em curso. O registro pertence ao usuário autenticado pelo perfil e ao workspace desse perfil.

PARÂMETRO	OBRIGATÓRIO	USO
`--task <id>`	sim	Tarefa associada ao tempo.
`--description <texto>`	não	Descrição curta do trabalho.

Exemplos:

```
clickupfy time start --task 86abc123
```

```
clickupfy time start --task 86abc123 --description "Implementação"
```

```
clickupfy time start --task 86abc123 --description "Testes de regressão"
```

```
clickupfy --account produto time start --task 86abc123 --description "Revisão de documentação"
```

```
clickupfy --account cliente-a time start --task 77def456 --description "Correção da integração"
```

Depois da chamada, execute `time current` para confirmar o item ativo. Iniciar um timer não muda o status da tarefa nem avisa participantes; essas ações usam `task update` e `comment create` separadamente.

`clickupfy time stop`

Encerra o time entry em execução no workspace do perfil. Não recebe argumento de tarefa: ele para o registro atual. Por isso a consulta anterior é necessária em trabalhos paralelos ou quando mais de uma pessoa usa o mesmo perfil local.

Exemplos:

```
clickupfy time stop
```

```
clickupfy --account produto time stop
```

```
clickupfy --account cliente-a time stop
```

```
clickupfy time stop && clickupfy time current
```

```
clickupfy --json time stop
```

O quarto exemplo verifica que não há entrada em execução após o encerramento. Parar o tempo não fecha tarefa, não altera seus pontos e não resolve checklist items. Registre a conclusão do trabalho nas interfaces próprias se elas também forem autorizadas.

Planeje Sprints e Sprint Points

Como o ClickUpfy reconhece uma Sprint

No ClickUp, uma Sprint é uma List dentro de um Sprint Folder. O ClickUpfy considera Sprint uma List que possua `start_date` e `due_date`. Lists comuns do mesmo Folder ficam fora do resultado, salvo quando `--include-regular` for usado.

O CLI não cria Sprints nem habilita o Sprint ClickApp. Essas operações continuam na interface do ClickUp.

Liste os ciclos

```
clickupfy sprint list --folder <sprint-folder-id>
```

Para diagnóstico de um Folder misto:

```
clickupfy sprint list \
  --folder <sprint-folder-id> \
  --include-regular
```

Confira nome, ID, início e término. Um período ausente indica uma List comum ou uma Sprint ainda não configurada corretamente.

Encontre a Sprint atual

```
clickupfy sprint current --folder <sprint-folder-id>
```

O comando espera uma única Sprint ativa na data atual. Nenhuma Sprint ativa ou mais de uma sobreposta produz uma falha explícita. Corrija o calendário no ClickUp antes de automatizar a seleção.

Leia o relatório

```
clickupfy sprint get <sprint-id>
clickupfy sprint tasks <sprint-id>
clickupfy sprint tasks <sprint-id> --open-only
```

O relatório percorre todas as páginas e inclui tarefas concluídas. O avanço é calculado por quantidade de tarefas e por Sprint Points. Uma tarefa sem Points participa da quantidade de itens, mas não acrescenta peso ao cálculo por pontos.

Use `--open-only` para a fila de trabalho corrente, não para o relatório final da Sprint.

Associe uma tarefa sem mover a List principal

```
clickupfy sprint add-task <sprint-id> <task-id>
```

A API associa a tarefa à Sprint e preserva sua List principal. Isso permite manter backlog e ciclo como dimensões diferentes.

Para remover a associação:

```
clickupfy sprint remove-task <sprint-id> <task-id>
```

Essa ação não exclui a tarefa.

Defina Sprint Points

```
clickupfy sprint set-points <task-id> 5
```

O mesmo campo pode ser atualizado pelo grupo de tarefas:

```
clickupfy task update <task-id> --points 8
```

Use a escala adotada pela equipe. O ClickUpfy transmite o número, mas não define se ele representa complexidade, esforço ou tamanho relativo.

Configure o MCP para Sprints

Acrescente o Folder ao projeto:

```
clickupfy --account produto agent install \
  --space 10 \
  --folder 20 \
  --list 30 \
  --sprint-folder 40
```

As ferramentas de Sprint podem omitir `folderId` quando esse valor está fixado. Um ID diferente é recusado. Consultas permanecem disponíveis em read-only. Associação e Points aparecem apenas no servidor com escrita.

Depois do planejamento, aprenda a registrar o tempo.

Referência do CLI: Sprints e Sprint Points

Consulta de ciclos

`clickupfy sprint list --folder <id>`

Lista Sprints identificadas pelo período. `--folder` é obrigatório. Use `--include-regular` para diagnosticar um Folder misto e `--archived` para incluir Lists arquivadas. `--at` recebe uma data `AAAA-MM-DD` para calcular o estado do ciclo naquela data.

PARÂMETRO	OBRIGATÓRIO	USO
`--folder <id>`	sim	Sprint Folder consultado.
`--archived`	não	Inclui Lists arquivadas.
`--include-regular`	não	Inclui Lists sem período configurado.
`--at <AAAA-MM-DD>`	não	Data usada no cálculo do estado.

Exemplos:

```
clickupfy sprint list --folder 4001
```

```
clickupfy sprint list --folder 4001 --at 2026-08-10
```

```
clickupfy sprint list --folder 4001 --include-regular
```

```
clickupfy sprint list --folder 4001 --archived --include-regular
```

```
clickupfy --account produto --json sprint list --folder 4001 --at 2026-09-01
```

Uma List sem início ou término aparece somente com `--include-regular`; ela não deve ser tratada como Sprint. Guarde o ID de uma Sprint retornada para os comandos de relatório e associação.

`clickupfy sprint current --folder <id>`

Encontra a única Sprint cujo período inclui a data atual ou a data indicada por `--at`. A consulta falha quando não existe ciclo ativo ou quando dois ciclos se sobrepõem; essa falha protege o planejamento contra uma seleção arbitrária.

PARÂMETRO	OBRIGATÓRIO	USO
`--folder <id>`	sim	Sprint Folder consultado.
`--at <AAAA-MM-DD>`	não	Data usada para localizar a Sprint.

Exemplos:

```
clickupfy sprint current --folder 4001
```

```
clickupfy sprint current --folder 4001 --at 2026-08-10
```

```
clickupfy sprint current --folder 4001 --at 2026-09-01
```

```
clickupfy --account produto sprint current --folder 4001
```

```
clickupfy --account cliente-a --json sprint current --folder 5001 --at 2026-10-15
```

Corrija datas sobrepostas no ClickUp quando o comando falhar. Não passe uma List comum para `sprint get` como alternativa: o relatório depende do período da Sprint.

`clickupfy sprint get <sprint-id>`

Mostra período, estado temporal, total de tarefas, distribuição de status e progresso por quantidade e por Sprint Points. `--at` muda a leitura temporal, sem alterar a Sprint nem suas tarefas.

PARÂMETRO	OBRIGATÓRIO	USO
`<sprint-id>`	sim	Sprint a analisar.
`--at <AAAA-MM-DD>`	não	Data usada no cálculo do estado.

Exemplos:

```
clickupfy sprint get sprint-10
```

```
clickupfy sprint get sprint-10 --at 2026-08-10
```

```
clickupfy sprint get sprint-11
```

```
clickupfy --account produto sprint get sprint-10
```

```
clickupfy --account cliente-a --json sprint get sprint-22 --at 2026-09-01
```

Uma tarefa sem Points entra no total de tarefas, mas não adiciona peso ao progresso por pontos. Consulte `sprint tasks` para inspecionar as tarefas que compõem esses números.

`clickupfy sprint tasks <sprint-id>`

Lista tarefas associadas à Sprint. Por padrão, inclui tarefas concluídas. `--open-only` omite itens fechados e serve para a fila de trabalho atual, não para conferir o resultado final do ciclo.

PARÂMETRO	OBRIGATÓRIO	USO
`<sprint-id>`	sim	Sprint consultada.
`--open-only`	não	Omite tarefas concluídas.

Exemplos:

```
clickupfy sprint tasks sprint-10
```

```
clickupfy sprint tasks sprint-10 --open-only
```

```
clickupfy sprint tasks sprint-11
```

```
clickupfy --account produto sprint tasks sprint-10 --open-only
```

```
clickupfy --account cliente-a --json sprint tasks sprint-22
```

O retorno é um resumo de tarefas. Para obter descrição, subtarefas e checklist items de uma tarefa, execute `clickupfy task get <task-id>`.

Alterações no planejamento

`clickupfy sprint add-task <sprint-id> <task-id>`

Associa uma tarefa existente a uma Sprint sem trocar sua List principal. Isso permite manter a tarefa no backlog e no ciclo atual ao mesmo tempo. A operação não cria cópia, não move a tarefa e não modifica status.

PARÂMETRO	OBRIGATÓRIO	USO
`<sprint-id>`	sim	Sprint de destino.
`<task-id>`	sim	Tarefa que será associada.

Exemplos:

```
clickupfy sprint add-task sprint-10 86abc123
```

```
clickupfy sprint add-task sprint-10 77def456
```

```
clickupfy --account produto sprint add-task sprint-10 86abc123
```

```
clickupfy --account cliente-a sprint add-task sprint-22 55ghi789
```

```
clickupfy --json sprint add-task sprint-10 99jkl012
```

Reler `sprint tasks <sprint-id>` confirma a associação. Só execute essa ação quando o planejamento da Sprint estiver no escopo autorizado.

`clickupfy sprint remove-task <sprint-id> <task-id>`

Remove a associação entre tarefa e Sprint. A tarefa continua existindo na List principal e não perde seus comentários, checklist, tempo ou histórico.

PARÂMETRO	OBRIGATÓRIO	USO
`<sprint-id>`	sim	Sprint cuja associação será removida.
`<task-id>`	sim	Tarefa associada.

Exemplos:

```
clickupfy sprint remove-task sprint-10 86abc123
```

```
clickupfy sprint remove-task sprint-10 77def456
```

```
clickupfy --account produto sprint remove-task sprint-10 86abc123
```

```
clickupfy --account cliente-a sprint remove-task sprint-22 55ghi789
```

```
clickupfy --json sprint remove-task sprint-10 99jkl012
```

Para excluir uma tarefa, use `task delete` apenas com autorização explícita. Para confirmar que a associação saiu, releia a Sprint e a tarefa depois desta operação.

`clickupfy sprint set-points <task-id> <points>`

Define Sprint Points de uma tarefa. O número deve ser igual ou maior que zero. O ClickUpfy não interpreta a escala: a equipe define se `1`, `2`, `3`, `5` ou outros valores representam tamanho, esforço ou complexidade.

PARÂMETRO	OBRIGATÓRIO	USO
`<task-id>`	sim	Tarefa que receberá pontos.
`<points>`	sim	Número não negativo.

Exemplos:

```
clickupfy sprint set-points 86abc123 0
```

```
clickupfy sprint set-points 86abc123 1
```

```
clickupfy sprint set-points 86abc123 3
```

```
clickupfy --account produto sprint set-points 86abc123 5
```

```
clickupfy --account cliente-a --json sprint set-points 77def456 8
```

`task update <task-id> --points <n>` atualiza o mesmo campo. Use apenas uma das interfaces por alteração e releia a tarefa para confirmar o valor salvo.

Registre tempo de trabalho

Consulte antes de iniciar

```
clickupfy time current
```

Essa leitura evita iniciar um segundo registro enquanto outro item permanece em execução. Se houver um time entry ativo, confirme a tarefa e decida se ele deve continuar ou ser encerrado.

Inicie um registro

```
clickupfy time start \
  --task <task-id> \
  --description "Implementação e testes"
```

Use uma descrição curta e observável. O registro pertence ao usuário autenticado pelo account selecionado.

Em um fluxo com agente, inicie o tempo somente quando a implementação realmente começar. Leitura inicial, esclarecimento e espera por autorização não devem ser registrados automaticamente sem uma regra explícita da equipe.

Encerre o registro

```
clickupfy time stop
```

Depois, consulte novamente:

```
clickupfy time current
```

A ausência de registro em execução confirma o encerramento.

Use outro account em uma operação

```
clickupfy --account cliente-a time current
clickupfy --account cliente-a time start \
  --task <task-id> \
  --description "Correção e regressão"
```

Essa seleção não altera o account ativo. É útil quando dois projetos usam workspaces diferentes na mesma máquina.

Relação com comentários e status

Time tracking não muda status nem publica comentário. As três ações representam evidências diferentes:

- status indica a etapa no fluxo
- comentário registra um evento legível pela equipe
- time entry mede o período atribuído ao trabalho.

A skill de implementação coordena essas ações, mas cada mutação continua explícita e verificável.

Continue em skills para agentes.

Use as skills para agentes

O catálogo incluído

O pacote distribui cinco skills:

SKILL	RESPONSABILIDADE
`clickupfy-setup`	Configuração de perfil, skills e MCP do projeto.
`clickupfy-dev`	Consulta e atualização do trabalho diário de software.
`clickup-issue-create`	Criação de tarefa, subtarefas e checklists sem executar a implementação.
`clickup-issue-implement`	Execução acompanhada por plano, comentários, tempo, testes e status.
`clickupfy-release`	Preparação e acompanhamento de versões e artefatos.

As skills são instruções. Elas usam o CLI ou o MCP disponível, mas não contêm credenciais.

Inspecione antes de instalar

```
clickupfy agent skill list
clickupfy agent skill show clickupfy-dev
```

`list` apresenta o catálogo. `show` imprime a fonte empacotada da skill, útil para revisar permissões e fluxo antes da instalação. As cinco skills são mantidas em português do Brasil.

Configure o projeto

Use `$clickupfy-setup` para preparar um repositório novo ou revisar uma integração existente. A skill executa `clickupfy doctor`, orienta o perfil com `clickupfy auth add` quando necessário, instala as skills por `npx skills add` e cria o MCP somente depois de confirmar os IDs do ClickUp.

Instale no projeto

Na raiz do repositório, o ClickUpfy delega a instalação ao gerenciador `skills`:

```
clickupfy agent skill install
```

As skills entram em `.agents/skills/` e o gerenciador registra a origem em `skills-lock.json`. Esse modo mantém a configuração próxima ao projeto e permite versionar as instruções.

Para instalar no catálogo global do usuário, em `~/.codex/skills/`:

```
clickupfy agent skill install --global
```

Prefira a instalação local quando projetos usam versões ou regras diferentes. Consulte a instalação efetiva com:

```
npx skills list
npx skills list --global
```

`--force` é aceito por compatibilidade. A atualização passa pelo próprio `npx skills add`, que mantém o lockfile e os caminhos canônicos:

```
clickupfy agent skill install --force
```

Crie uma issue sem implementá-la

Peça explicitamente a skill de criação:

```
Use $clickup-issue-create para transformar esta solicitação em uma tarefa no
ClickUp. Crie subtarefas e checklists de teste, mas não implemente o trabalho.
```

A skill deve obter os status da List, redigir a descrição em Markdown, criar a tarefa principal e materializar subtarefas e checklists recursivos. A criação não autoriza editar código nem marcar validações como concluídas.

Implemente uma issue existente

```
Use $clickup-issue-implement para executar a tarefa <task-id> até a validação
final.
```

O fluxo lê a tarefa inteira, monta um plano visível, registra início, datas e time tracking quando aplicável, percorre subtarefas e checklist items e publica checkpoints. Um item de teste só é resolvido depois do comando correspondente passar.

A conclusão exige releitura do estado remoto, regressão aplicável e um status terminal existente na List.

Use a skill diária

`clickupfy-dev` é adequada para consultas e mutações pontuais:

```
Use $clickupfy-dev para ler a tarefa <task-id>, publicar o resultado dos testes
e marcar apenas o checklist correspondente.
```

Escopo e autorização continuam limitados pelo pedido. A skill não transforma uma consulta em implementação completa.

Prepare uma release

`clickupfy-release` lê a documentação de release do próprio repositório e acompanha versão, changelog, validações e artefatos. Ela não deve ser instalada como regra genérica em projetos que apenas consomem o CLI.

Agora veja como essas skills acessam o ClickUp pelo servidor MCP.

Referência do CLI: Docs, skills e servidor MCP

Docs

`clickupfy doc list`

Busca Docs no workspace associado ao perfil. `--max-pages` controla o número de páginas de cursor consultadas e aceita inteiros de `1` a `50`; o padrão é `50`. `--parent-id` deve ser usado junto de `--parent-type` para identificar o local.

PARÂMETRO	OBRIGATÓRIO	USO
`--query <texto>`	não	Filtro pelo nome ou ID do Doc.
`--parent-id <id>`	não	Local pai do Doc.
`--parent-type <n>`	não	`4` Space, `5` Folder, `6` List, `7` Everything, `12` tarefa.
`--deleted`	não	Inclui Docs excluídos.
`--archived`	não	Inclui Docs arquivados.
`--creator <id>`	não	ID numérico da pessoa criadora.
`--max-pages <n>`	não	Cursor de `1` a `50`; padrão `50`.

Exemplos:

```
clickupfy doc list
```

```
clickupfy doc list --query "API"
```

```
clickupfy doc list --parent-id 3001 --parent-type 6
```

```
clickupfy doc list --archived --deleted --creator 42 --max-pages 10
```

```
clickupfy --account produto --json doc list --query "Manual" --max-pages 5
```

`--parent-type` sem `--parent-id` não cria uma consulta útil, pois não há local para classificar. A saída inclui os metadados necessários para chamar `doc get` ou navegar pelas páginas.

`clickupfy doc get <doc-id>`

Obtém metadados de um Doc no workspace atual. Ele não devolve automaticamente o conteúdo das páginas; use `doc page list` para conteúdo completo ou `doc page get` para uma página determinada.

Exemplos:

```
clickupfy doc get doc-1
```

```
clickupfy --json doc get doc-1
```

```
clickupfy --account produto doc get doc-1
```

```
clickupfy --account cliente-a --json doc get doc-22
```

```
clickupfy doc get doc-1 > /tmp/doc-1.txt
```

O último exemplo guarda a tabela compacta. Para processamento por outro programa, acrescente `--json` e preserve os dados do workspace como informação interna da equipe.

`clickupfy doc create --name <nome>`

Cria um Doc no workspace. Quando `--parent-id` é informado, `--parent-type` é obrigatório. Os tipos são `4` Space, `5` Folder, `6` List, `7` Everything e `12` tarefa. `--visibility` aceita o valor definido pela API, normalmente `PRIVATE` ou `PUBLIC`; `--create-page` cria a primeira página em branco.

PARÂMETRO	OBRIGATÓRIO	USO
`--name <nome>`	sim	Nome do Doc.
`--parent-id <id>`	não	Local onde o Doc será criado.
`--parent-type <n>`	condicional	Tipo do local quando há `parent-id`.
`--visibility <valor>`	não	Visibilidade, como `PRIVATE` ou `PUBLIC`.
`--create-page`	não	Cria uma primeira página vazia.

Exemplos:

```
clickupfy doc create --name "Manual da API"
```

```
clickupfy doc create --name "Guia do produto" --create-page
```

```
clickupfy doc create --name "Notas da List" --parent-id 3001 --parent-type 6
```

```
clickupfy doc create --name "Documento interno" --parent-id 2001 --parent-type 5 --
visibility PRIVATE
```

```
clickupfy --account produto --json doc create --name "Plano de release" --parent-id
86abc123 --parent-type 12 --create-page
```

Uma criação bem-sucedida devolve o ID do Doc. Releia seus metadados e crie as páginas pelo grupo `doc page`. A API pública usada pelo ClickUpfy não expõe exclusão de Docs, mudança de permissões ou reordenação da árvore.

`clickupfy doc page tree <doc-id>`

Mostra somente a árvore de páginas, sem o corpo. É a leitura mais econômica para descobrir um `page-id` e a relação pai-filho antes de criar uma subpágina ou atualizar conteúdo.

Exemplos:

```
clickupfy doc page tree doc-1
```

```
clickupfy --json doc page tree doc-1
```

```
clickupfy --account produto doc page tree doc-1
```

```
clickupfy --account cliente-a --json doc page tree doc-22
```

```
clickupfy doc page tree doc-1 > /tmp/doc-1-tree.txt
```

O retorno não é cópia de segurança do conteúdo. Use `doc page list` quando o objetivo for ler todas as páginas em Markdown.

`clickupfy doc page list <doc-id>`

Lista páginas com conteúdo. `--max-page-depth` limita a profundidade de subpáginas retornada. `--content-format` aceita `text/md`, o padrão, ou `text/plain` para texto sem a representação Markdown da API.

PARÂMETRO	OBRIGATÓRIO	USO
`<doc-id>`	sim	Doc cujas páginas serão lidas.
`--max-page-depth <n>`	não	Profundidade máxima de subpáginas.
`--content-format <valor>`	não	`text/md` ou `text/plain`.

Exemplos:

```
clickupfy doc page list doc-1
```

```
clickupfy doc page ls doc-1 --max-page-depth 1
```

```
clickupfy doc page list doc-1 --max-page-depth 3
```

```
clickupfy doc page list doc-1 --content-format text/plain
```

```
clickupfy --account produto --json doc page list doc-1 --content-format text/md
```

Use uma profundidade pequena quando só precisa dos capítulos principais. Para uma página conhecida, `doc page get` reduz a resposta e evita transmitir o conteúdo de outras páginas a um agente.

`clickupfy doc page get <doc-id> <page-id>`

Obtém uma página específica e seu conteúdo. `--content-format` tem os mesmos valores de `doc page list`: `text/md` por padrão e `text/plain` como opção.

Exemplos:

```
clickupfy doc page get doc-1 page-1
```

```
clickupfy doc page get doc-1 page-1 --content-format text/plain
```

```
clickupfy --json doc page get doc-1 page-1
```

```
clickupfy --account produto doc page get doc-1 page-2 --content-format text/md
```

```
clickupfy --account cliente-a --json doc page get doc-22 page-8
```

O ID da página deve pertencer ao Doc informado. Localize-o com `doc page tree` ou `doc page list`, não com tentativa e erro.

`clickupfy doc page create <doc-id> --name <nome>`

Cria página ou subpágina em um Doc. `--parent-page` indica uma página pai; `--orderindex` escolhe a posição entre páginas irmãs. O conteúdo inicial e o subtítulo são opcionais. Não use `orderindex` como substituto de uma revisão da árvore existente: leia a árvore antes de inserir conteúdo.

PARÂMETRO	OBRIGATÓRIO	USO
`<doc-id>`	sim	Doc proprietário.
`--name <nome>`	sim	Título da página.
`--content <texto>`	não	Conteúdo inicial.
`--sub-title <texto>`	não	Subtítulo.
`--parent-page <id>`	não	Página pai de uma subpágina.
`--orderindex <n>`	não	Posição entre páginas irmãs.
`--content-format <valor>`	não	`text/md` ou `text/plain`.

Exemplos:

```
clickupfy doc page create doc-1 --name "Introdução"
```

```
clickupfy doc page create doc-1 --name "Autenticação" --content "Use uma API key pessoal."
```

```
clickupfy doc page create doc-1 --name "Detalhes" --sub-title "Campos e respostas" --content "## Parâmetros"
```

```
clickupfy doc page create doc-1 --name "Erros" --parent-page page-1 --orderindex 2
```

```
clickupfy --account produto --json doc page create doc-1 --name "Integração" --content "# MCP" --content-format text/md
```

Leia a página criada com `doc page get` e a árvore com `doc page tree` para confirmar conteúdo e posição. A criação não atualiza outras páginas nem cria um Doc novo.

`clickupfy doc page update <doc-id> <page-id>`

Atualiza título, subtítulo e/ou conteúdo de uma página. Pelo menos um campo é necessário. `--content-edit-mode` controla o conteúdo: `replace` substitui, `append` acrescenta ao fim e `prepend` insere no começo. Sem a flag, o modo é `replace`.

PARÂMETRO	OBRIGATÓRIO	USO
`<doc-id>`, `<page-id>`	sim	Doc e página a atualizar.
`--name <nome>`	não	Novo título.
`--sub-title <texto>`	não	Novo subtítulo.
`--content <texto>`	não	Conteúdo enviado à API.
`--content-edit-mode <modo>`	não	`replace`, `append` ou `prepend`.
`--content-format <valor>`	não	`text/md` ou `text/plain`.

Exemplos:

```
clickupfy doc page update doc-1 page-1 --name "Visão geral"
```

```
clickupfy doc page update doc-1 page-1 --sub-title "Instalação e configuração"
```

```
clickupfy doc page update doc-1 page-1 --content "# Novo conteúdo"
```

```
clickupfy doc page update doc-1 page-1 --content "\n## Changelog" --content-edit-mode append
```

```
clickupfy --account produto --json doc page update doc-1 page-2 --content "# Aviso\n\n" --content-edit-mode prepend --content-format text/md
```

Leia a página depois da escrita para confirmar o modo aplicado. A API pública usada pelo ClickUpfy não oferece exclusão de página nem movimentação na árvore.

Skills e projeto de agente

`clickupfy agent skill list`

Lista as skills embaladas: `clickupfy-dev`, `clickup-issue-create`, `clickup-issue-implement` e `clickupfy-release`. Não instala nada e não precisa de perfil configurado. O estado da instalação é responsabilidade do gerenciador `npx skills`; use `npx skills list` no projeto ou `npx skills list --global`.

Exemplos:

```
clickupfy agent skill list
```

```
clickupfy agent skill list | sort
```

```
clickupfy agent skill list > /tmp/clickupfy-skills.txt
```

```
clickupfy agent skill list | wc -l
```

```
clickupfy agent skill list && clickupfy agent skill show clickupfy-dev
```

O catálogo é incorporado ao pacote ou executável. `list` não verifica se uma skill já está instalada no projeto; ele apenas mostra o que a versão instalada do ClickUpfy pode distribuir. As instruções e os metadados distribuídos estão em português do Brasil.

`clickupfy agent skill show <skill>`

Imprime a fonte da skill embarcada. Use-o para revisar instruções e permissões antes da instalação. O argumento precisa ser um dos quatro nomes do catálogo.

Exemplos:

```
clickupfy agent skill show clickupfy-dev
```

```
clickupfy agent skill show clickup-issue-create
```

```
clickupfy agent skill show clickup-issue-implement
```

```
clickupfy agent skill show clickupfy-release
```

```
clickupfy agent skill show clickupfy-dev > /tmp/clickupfy-dev-skill.md
```

Uma skill inexistente é recusada. O comando só lê o asset distribuído; ele não abre uma cópia local já instalada, que pode ter sido alterada pelo projeto.

`clickupfy agent skill install [skills...]`

Instala todas as skills quando nenhum nome é informado, ou somente as skills posicionais escolhidas, chamando `npx skills add promovaweb/clickupfy`. Sem `--global`, o destino é `.agents/skills/` da pasta atual e o gerenciador atualiza `skills-lock.json`. `--global` usa `~/.codex/skills`. A instalação sempre usa `--agent codex`, `--copy` e `--yes`.

PARÂMETRO	OBRIGATÓRIO	USO
`[skills...]`	não	Uma ou mais skills do catálogo.
`--global`	não	Instala no catálogo global do usuário.
`--force`	não	Permite repetir a instalação.

Exemplos:

```
clickupfy agent skill install
```

```
clickupfy agent skill install clickupfy-dev
```

```
clickupfy agent skill install clickup-issue-create clickup-issue-implement
```

```
clickupfy agent skill install --global clickupfy-dev
```

```
clickupfy agent skill install clickupfy-release --force
```

`--target` não é aceito porque os caminhos canônicos pertencem ao gerenciador `npx skills`. Revise a fonte com `agent skill show` e consulte `npx skills list` antes de atualizar regras de um projeto.

`clickupfy agent install`

Instala as skills e mescla um servidor `promovaweb-clickupfy` no `.mcp.json` e no `.codex/config.toml` do projeto. `--space` e `--list` são obrigatórios. Os arquivos gerados guardam perfil e IDs, mas nunca a API key. O JSON usa `mcpServers`; o Codex usa `[mcp_servers."promovaweb-clickupfy"]`. Acrescente `--read-only` manualmente aos argumentos das entradas quando o primeiro uso só pode consultar dados.

PARÂMETRO	OBRIGATÓRIO	USO
`--global`	não	Instala skills globalmente.
`--workspace <id>`	não	Workspace esperado pelo MCP.
`--space <id>`	sim	Space fixado para o projeto.
`--folder <id>`	não	Folder fixado.
`--list <id>`	sim	List fixa e obrigatória.
`--sprint-folder <id>`	não	Sprint Folder do projeto.
`--force`	não	Atualiza skills existentes.

Exemplos:

```
clickupfy agent install --space 1001 --list 3001
```

```
clickupfy --account produto agent install --workspace 123456 --space 1001 --folder 2001 --list 3001
```

```
clickupfy agent install --space 1001 --list 3001 --sprint-folder 4001
```

```
clickupfy agent install --global --space 1001 --list 3001 --force
```

```
clickupfy --account cliente-a agent install --workspace 987654 --space 5001 --list 7001 --sprint-folder 8001 --force
```

O comando preserva outros servidores e configurações nos dois arquivos. Reabra o cliente MCP e chame `clickupfy_mcp_context` para conferir o destino. O CLI não lê esses arquivos em chamadas normais: eles só são usados pelo cliente para iniciar `mcp serve`.

`clickupfy mcp serve --list <id>`

Inicia o servidor MCP em stdio. A saída padrão é reservada ao protocolo JSON-RPC; mensagens operacionais são escritas em stderr. `--list` é obrigatório. Os outros IDs fixam os níveis superiores; `--read-only` remove ferramentas de escrita de `tools/list`.

PARÂMETRO	OBRIGATÓRIO	USO
`--account <perfil>`	não	Perfil fixado.
`--workspace <id>`	não	Workspace esperado.
`--space <id>`	não	Space fixado.
`--folder <id>`	não	Folder fixado.
`--list <id>`	sim	List fixa.
`--sprint-folder <id>`	não	Sprint Folder fixado.
`--read-only`	não	Expõe somente ferramentas de leitura.

Exemplos:

```
clickupfy mcp serve --list 3001
```

```
clickupfy mcp serve --account produto --list 3001
```

```
clickupfy mcp serve --workspace 123456 --space 1001 --folder 2001 --list 3001
```

```
clickupfy mcp serve --list 3001 --sprint-folder 4001 --read-only
```

```
clickupfy mcp serve --account cliente-a --workspace 987654 --space 5001 --list 7001
--sprint-folder 8001
```

Normalmente o cliente MCP executa esse comando a partir do `.mcp.json` ou do `.codex/config.toml`; não o inicie em um terminal que também precise de saída humana. A referência MCP detalha cada ferramenta exposta pelo servidor e as diferenças entre leitura e escrita.

Conecte agentes pelo servidor MCP

Inicie manualmente

O servidor usa transporte stdio e reserva a saída padrão para as mensagens JSON-RPC trocadas com o cliente MCP:

```
clickupfy mcp serve --list 30
```

A List é obrigatória porque limita consultas e mutações ao destino do projeto. O comando abaixo também fixa os níveis superiores e o Sprint Folder:

```
clickupfy mcp serve \
  --account promovaweb \
  --workspace 123 \
  --space 10 \
  --folder 20 \
  --list 30 \
  --sprint-folder 40
```

Na prática, deixe o cliente MCP iniciar esse processo pelo `.mcp.json` ou pelo `.codex/config.toml` quando o cliente for o Codex. O comando `agent install` gera e atualiza os dois formatos.

Ative read-only

```
clickupfy mcp serve --list 30 --read-only
```

O modo read-only não registra ferramentas de escrita. Elas deixam de aparecer em `tools/list`, portanto a restrição é aplicada na superfície do servidor e não depende apenas da obediência do agente.

Use esse modo para descoberta, auditoria, demonstração ou qualquer conversa que não autorize alterações. Confira `tools/list` no cliente para confirmar que as ferramentas de escrita não foram registradas.

Entenda o isolamento

O MCP recebe IDs no processo. Quando uma ferramenta omite um ID, o servidor usa o valor fixado. Quando recebe um valor diferente, recusa a operação. Essa comparação protege estes níveis:

- account
- workspace
- Space
- Folder
- List
- Sprint Folder.

A busca de tarefas também permanece dentro da List do projeto. Assim, dois clientes MCP podem operar em paralelo sem compartilhar destino.

Ferramentas de contexto e navegação

`clickupfy_mcp_context` mostra perfil e hierarquia sem expor a API key. `clickupfy_list_get` retorna os status válidos da List.

Catálogo disponível em read-only

FERRAMENTA	FUNÇÃO
`clickupfy_mcp_context`	Mostra o account e os IDs fixados pelo projeto.
`clickupfy_accounts_list`	Lista accounts locais sem revelar API keys.
`clickupfy_whoami`	Valida o account e retorna o usuário autenticado.
`clickupfy_workspaces_list`	Lista workspaces autorizados.
`clickupfy_spaces_list`	Lista Spaces do workspace do account.
`clickupfy_folders_list`	Lista Folders do Space fixado ou informado.
`clickupfy_lists_list`	Lista Lists de um Folder ou diretamente de um Space.
`clickupfy_list_get`	Lê a List do projeto e seus status.
`clickupfy_tasks_list`	Lista tarefas da List autorizada.
`clickupfy_tasks_search`	Busca tarefas dentro da List autorizada.
`clickupfy_task_get`	Lê uma tarefa como resposta original, fila `execution` ou Markdown.
`clickupfy_comments_list`	Lê os comentários de uma tarefa.

FERRAMENTA	FUNÇÃO
`clickupfy_sprints_list`	Lista Sprints do Sprint Folder.
`clickupfy_sprint_current`	Obtém a Sprint ativa na data de referência.
`clickupfy_sprint_get`	Produz o relatório de uma Sprint.
`clickupfy_sprint_tasks`	Lista tarefas associadas à Sprint.
`clickupfy_time_current`	Consulta o time entry em execução.
`clickupfy_docs_list`	Busca Docs do workspace do account.
`clickupfy_doc_get`	Lê metadados de um Doc.
`clickupfy_doc_page_tree`	Lê a hierarquia de páginas do Doc, sem conteúdo.
`clickupfy_doc_pages_list`	Lê as páginas do Doc com conteúdo em Markdown.
`clickupfy_doc_page_get`	Lê uma página específica com conteúdo.

Catálogo acrescentado no modo com escrita

FERRAMENTA	FUNÇÃO
`clickupfy_account_use`	Altera o account ativo quando o servidor não fixa um account.
`clickupfy_workspace_use`	Altera o workspace quando account e workspace não estão fixados.
`clickupfy_task_create`	Cria tarefa ou subtarefa na List autorizada.
`clickupfy_task_update`	Atualiza campos de uma tarefa.
`clickupfy_checklist_create`	Cria um checklist em uma tarefa.
`clickupfy_checklist_item_create`	Acrescenta um item ao checklist.
`clickupfy_checklist_item_set`	Marca ou reabre um item e confirma o novo estado.
`clickupfy_sprint_add_task`	Associa uma tarefa à Sprint.
`clickupfy_sprint_remove_task`	Remove a associação sem excluir a tarefa.
`clickupfy_sprint_set_points`	Define Sprint Points.
`clickupfy_task_delete`	Exclui uma tarefa mediante `confirm: true`.
`clickupfy_comment_create`	Publica um comentário na tarefa.
`clickupfy_time_start`	Inicia um time entry associado à tarefa.
`clickupfy_time_stop`	Encerra o time entry atual.
`clickupfy_doc_create`	Cria um Doc no workspace, opcionalmente vinculado a um local.

FERRAMENTA	FUNÇÃO
`clickupfy_doc_page_create`	Cria uma página, ou sub-página, em um Doc.
`clickupfy_doc_page_update`	Atualiza título, subtítulo e/ou conteúdo de uma página.

Na criação e na atualização, informe apenas os campos que devem ser enviados. O servidor recusa IDs que não correspondem ao escopo fixado. A skill de implementação consulta `clickupfy_time_current` antes de iniciar outro registro.

As ferramentas de Docs não seguem o isolamento por Space, Folder ou List: elas sempre operam no workspace do account resolvido, porque Docs não pertencem a uma List. `clickupfy_doc_create` exige `parentType` sempre que `parentId` for informado, seguindo os mesmos tipos de local da API do ClickUp (`4` Space, `5` Folder, `6` List, `7` Everything, `12` tarefa).

Use Markdown para contexto longo

`clickupfy_task_get` aceita `markdown: true`. Nesse modo, a ferramenta retorna um texto único, sem envolver o documento em uma string JSON escapada. Essa forma reduz ruído para agentes e preserva checkboxes hierárquicos.

Esta solicitação usa a leitura em Markdown e informa ao agente que nenhuma mutação está autorizada:

> `Leia a tarefa 86abc123 com clickupfy_task_get e markdown: true. Resuma o objetivo, liste os itens pendentes e não faça mutações.`

Diagnostique a conexão

Se o cliente não listar ferramentas, siga a conexão desde o comando local até o recarregamento do processo:

1. execute o comando de `args` manualmente no terminal
2. confira se o account existe
3. confirme que `--list` possui valor
4. valide o JSON
5. reinicie o cliente MCP
6. leia os logs sem imprimir a API key.

A referência do CLI reúne os comandos e as flags usados para reproduzir o servidor manualmente quando o diagnóstico exigir uma comparação.

Referência MCP: contexto, perfis e hierarquia

Ferramentas de identificação

`clickupfy_mcp_context`

Mostra o perfil resolvido, o workspace associado e a hierarquia fixada pelo processo MCP. Execute-a como primeira chamada de um agente: ela informa quais IDs podem ser omitidos nas ferramentas seguintes e se o projeto tem `sprintFolderId`. O único argumento aceito é `account`, opcional.

CAMPO	TIPO	OBRIGATÓRIO	EFEITO
`account`	string	não	Perfil a resolver quando o processo não o fixou.

Exemplos:

```
{}
```

```
{"account":"produto"}
```

```
{"account":"cliente-a"}
```

```
{"account":"homologacao"}
```

```
{"account":"suporte"}
```

Uma chamada sem argumento não seleciona outro perfil: ela usa o perfil fixado ou o perfil ativo do arquivo local. Guarde os valores de `scope` durante a sessão. O campo `listId` sempre existe porque `mcp serve --list` é obrigatório.

`clickupfy_accounts_list`

Lista todos os perfis configurados na máquina, com nome, usuário, workspace e marca do perfil ativo. Ela não recebe argumentos e não faz requisição à API do ClickUp. Use-a para localizar o identificador de um perfil, nunca para obter uma credencial.

Exemplos:

```
{}
```

```
{}
```

```
{}
```

```
{}
```

```
{}
```

Os cinco exemplos são idênticos porque a ferramenta não possui parâmetro. Em um cliente com chamada nomeada, a variação fica no nome da ferramenta, não nos argumentos: `clickupfy_accounts_list` sempre recebe o objeto vazio. Esse fato é importante para agentes: não invente filtros como `workspace`, `query` ou `includeArchived`, pois o schema os rejeita.

`clickupfy_whoami`

Valida o perfil escolhido na API do ClickUp e retorna o usuário autenticado, além do workspace associado ao perfil. É a confirmação remota para uma chave que pode ter sido revogada ou cujas permissões acabaram de mudar.

CAMPO	TIPO	OBRIGATÓRIO	EFEITO
`account`	string	não	Perfil que será autenticado.

Exemplos:

```
{}
```

```
{"account":"produto"}
```

```
{"account":"cliente-a"}
```

```
{"account":"homologacao"}
```

```
{"account":"suporte"}
```

Quando `account` for omitido, o mesmo resolvedor usado por `clickupfy_mcp_context` escolhe o perfil. Uma resposta bem-sucedida prova que a chave funciona naquele momento; não prova permissão para uma List específica. Consulte `clickupfy_mcp_context` e `clickupfy_list_get` para conferir o destino do projeto e os status permitidos.

`clickupfy_workspaces_list`

Lista os workspaces autorizados pela API key do perfil. Ela não recebe ID de workspace porque consulta todos os workspaces acessíveis e o servidor usa o perfil para autenticar a chamada.

CAMPO	TIPO	OBRIGATÓRIO	EFEITO
`account`	string	não	Perfil cuja chave será usada na consulta.

Exemplos:

```
{}
```

```
{"account":"produto"}
```

```
{"account":"cliente-a"}
```

```
{"account":"homologacao"}
```

```
{"account":"suporte"}
```

Essa ferramenta apenas lê os workspaces. Para alterar a associação do perfil, `clickupfy_workspace_use` só fica disponível quando o servidor tem escrita e não foi iniciado com `--account` nem `--workspace`. O valor retornado aqui deve ser usado literalmente nessa ferramenta de escrita ou no CLI.

Ferramentas de navegação

`clickupfy_spaces_list`

Lista Spaces do workspace associado ao perfil. A ferramenta usa o workspace do perfil ou o que foi fixado no processo. `archived` é opcional e, quando verdadeiro, acrescenta Spaces arquivados.

CAMPO	TIPO	OBRIGATÓRIO	EFEITO
`account`	string	não	Perfil usado para a consulta.
`archived`	booleano	não	Inclui Spaces arquivados quando vale `true`.

Exemplos:

```
{}
```

```
{"archived":true}
```

```
{"account":"produto"}
```

```
{"account":"produto","archived":true}
```

```
{"account":"cliente-a","archived":false}
```

O valor `false` é diferente de omitir o campo apenas para deixar a intenção explícita no registro do agente; os dois retornam somente recursos ativos. Escolha um `id` da resposta para chamar `clickupfy_folders_list` ou `clickupfy_lists_list`.

`clickupfy_folders_list`

Lista Folders de um Space. Quando o servidor foi iniciado com `--space`, omita `spaceId` para usar o valor fixado. Quando não há Space fixo, o campo é necessário. Um ID diferente do fixado é recusado para preservar o isolamento do projeto.

CAMPO	TIPO	OBRIGATÓRIO	EFEITO
`account`	string	não	Perfil usado para a consulta.
`spaceId`	string	condicional	Space a consultar; obrigatório sem Space fixo.
`archived`	booleano	não	Inclui Folders arquivados.

Exemplos:

```
{}
```

```
{"spaceId":"1001"}
```

```
{"spaceId":"1001","archived":true}
```

```
{"account":"produto","spaceId":"1001"}
```

```
{"account":"cliente-a","spaceId":"2001","archived":false}
```

O primeiro exemplo só funciona quando o processo MCP já recebeu `--space`. Uma resposta vazia pode significar que o Space guarda Lists diretamente; ela não autoriza assumir que o projeto não tem Lists. Nesse caso, consulte `clickupfy_lists_list` com `spaceId`.

`clickupfy_lists_list`

Lista Lists de um Folder ou Lists criadas diretamente em um Space. O servidor resolve `folderId` e `spaceId` contra os IDs fixados. Quando um Folder é resolvido, ele tem precedência e o Space não é enviado à API. Sem Folder e sem Space, a ferramenta recusa a chamada porque não há nível da hierarquia para consultar.

CAMPO	TIPO	OBRIGATÓRIO	EFEITO
`account`	string	não	Perfil usado para a consulta.
`folderId`	string	condicional	Folder que contém as Lists.
`spaceId`	string	condicional	Space usado quando as Lists não pertencem a Folder.
`archived`	booleano	não	Inclui Lists arquivadas.

Exemplos:

```
{}
```

```
{"folderId":"2001"}
```

```
{"folderId":"2001","archived":true}
```

```
{"spaceId":"1001"}
```

```
{"account":"produto","spaceId":"1001","archived":false}
```

A primeira chamada exige que o processo tenha iniciado com `--folder` ou `--space`. Não passe IDs de ambos os níveis para tentar ampliar a resposta. A separação permite que o agente descubra a hierarquia sem atravessar o destino do projeto.

`clickupfy_list_get`

Obtém os metadados de uma List e os status aceitos por suas tarefas. A List fixada no processo é a escolha normal: omita `listId` para usá-la. Só informe o campo em um MCP sem List fixa, situação que não ocorre no comando público `mcp serve`, ou para repetir exatamente o ID fixado.

CAMPO	TIPO	OBRIGATÓRIO	EFEITO
`account`	string	não	Perfil usado para a consulta.
`listId`	string	não no MCP do projeto	List a obter; omita para usar a List fixada.

Exemplos:

```
{}
```

```
{"listId":"3001"}
```

```
{"account":"produto"}
```

```
{"account":"produto","listId":"3001"}
```

```
{"account":"cliente-a","listId":"4001"}
```

Leia os nomes dos status retornados antes de chamar uma ferramenta que crie ou atualize tarefa. `status` não é um conjunto global do ClickUp: uma grafia como `em revisão` pode existir em uma List e ser recusada em outra. Esta ferramenta não altera o status de nenhuma tarefa.

Referência MCP: Docs e administração de perfil

Leitura de Docs

`clickupfy_docs_list`

Busca Docs do workspace do perfil. `maxPages` aceita inteiros de `1` a `50`. Quando `parentId` for informado, `parentType` informa o tipo do local: `4` Space, `5` Folder, `6` List, `7` Everything ou `12` tarefa.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`query`	string	não	Nome ou ID do Doc.
`parentId`	string	não	Local pai do Doc.
`parentType`	inteiro	não	Tipo de `parentId`.
`deleted`, `archived`	booleano	não	Inclui Docs excluídos ou arquivados.
`creator`	inteiro	não	Pessoa criadora.
`maxPages`	inteiro	não	Cursor de `1` a `50`.

Exemplos:

```
{}
```

```
{"query":"API"}
```

```
{"parentId":"3001","parentType":6}
```

```
{"archived":true,"deleted":true,"creator":42,"maxPages":10}
```

```
{"account":"produto","query":"Manual","maxPages":5}
```

Use o ID retornado em `clickupfy_doc_get`, `clickupfy_doc_page_tree` ou uma ferramenta de página. A busca não devolve automaticamente todo conteúdo das páginas e não se limita à List fixada pelo MCP.

`clickupfy_doc_get`

Obtém metadados de um Doc. O corpo pertence às páginas, por isso esta ferramenta é indicada para confirmar nome, local e estado do Doc antes de consultar ou alterar uma página.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`docId`	string	sim	Doc a obter.

Exemplos:

```
{"docId":"doc-1"}
```

```
{"docId":"doc-22"}
```

```
{"account":"produto","docId":"doc-1"}
```

```
{"account":"cliente-a","docId":"doc-22"}
```

```
{"account":"homologacao","docId":"doc-33"}
```

Se o Doc não pertencer ao workspace do perfil, a API recusa a chamada. Não substitua `docId` por `pageId`: os dois identificadores pertencem a recursos distintos.

`clickupfy_doc_page_tree`

Retorna a hierarquia de páginas de um Doc sem trazer seus conteúdos. É a escolha adequada para localizar `pageId`, parentesco e profundidade antes de criar uma subpágina ou atualizar uma página existente.

Exemplos:

```
{"docId":"doc-1"}
```

```
{"docId":"doc-22"}
```

```
{"account":"produto","docId":"doc-1"}
```

```
{"account":"cliente-a","docId":"doc-22"}
```

```
{"account":"homologacao","docId":"doc-33"}
```

Essa ferramenta não é uma cópia de segurança do texto. Use `clickupfy_doc_pages_list` para ler muitas páginas ou `clickupfy_doc_page_get` para ler uma página conhecida.

`clickupfy_doc_pages_list`

Lista páginas de um Doc com conteúdo. `maxPageDepth` limita o retorno de subpáginas. `contentFormat` aceita a forma usada pela API, normalmente `text/md` ou `text/plain`; quando omitido, a API usa `text/md`.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`docId`	string	sim	Doc cujas páginas serão lidas.
`maxPageDepth`	inteiro	não	Profundidade máxima de subpáginas.
`contentFormat`	string	não	Forma do conteúdo, como `text/md`.

Exemplos:

```
{"docId":"doc-1"}
```

```
{"docId":"doc-1","maxPageDepth":1}
```

```
{"docId":"doc-1","maxPageDepth":3}
```

```
{"docId":"doc-1","contentFormat":"text/plain"}
```

```
{"account":"produto","docId":"doc-1","contentFormat":"text/md"}
```

Leia somente a profundidade necessária para reduzir conteúdo irrelevante na conversa do agente. A ferramenta não modifica o Doc nem a árvore.

`clickupfy_doc_page_get`

Obtém uma página específica e seu conteúdo. O `pageId` deve pertencer ao Doc informado; localize-o pela árvore ou pela listagem de páginas.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`docId`	string	sim	Doc proprietário.
`pageId`	string	sim	Página a obter.
`contentFormat`	string	não	Forma do conteúdo.

Exemplos:

```
{"docId":"doc-1","pageId":"page-1"}
```

```
{"docId":"doc-1","pageId":"page-1","contentFormat":"text/plain"}
```

```
{"docId":"doc-1","pageId":"page-2","contentFormat":"text/md"}
```

```
{"account":"produto","docId":"doc-1","pageId":"page-1"}
```

```
{"account":"cliente-a","docId":"doc-22","pageId":"page-8"}
```

Escrita de Docs

As três ferramentas seguintes não aparecem em read-only. A API pública usada pelo ClickUpfy não expõe ferramentas para excluir Docs ou páginas, mudar permissões de um Doc existente ou reordenar a árvore depois da criação.

`clickupfy_doc_create`

Cria Doc no workspace. `name` é obrigatório. Quando `parentId` é informado, `parentType` também é obrigatório. `visibility` pode ser `PRIVATE` ou `PUBLIC`; `createPage` cria uma página em branco junto do Doc.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`name`	string	sim	Nome do Doc.
`parentId`	string	não	Local pai.
`parentType`	inteiro	condicional	Tipo do local pai.
`visibility`	string	não	Visibilidade do novo Doc.

CAMPO	TIPO	OBRIGATÓRIO	USO
`createPage`	booleano	não	Cria a primeira página vazia.

Exemplos:

```
{"name":"Manual da API"}
```

```
{"name":"Guia do produto","createPage":true}
```

```
{"name":"Notas da List","parentId":"3001","parentType":6}
```

```
{"name":"Documento interno","parentId":"2001","parentType":5,"visibility":"PRIVATE"}
```

```
{"account":"produto","name":"Plano de
release","parentId":"86abc123","parentType":12,"createPage":true}
```

Chame `clickupfy_doc_get` depois da criação e use o ID devolvido para criar as páginas. Quando houver `parentId` sem `parentType`, o servidor recusa a chamada antes de tocar na API.

`clickupfy_doc_page_create`

Cria página ou subpágina em um Doc. `name` é obrigatório. `parentPageId` define o pai; `orderindex` indica a posição entre páginas irmãs. O conteúdo e o subtítulo são opcionais.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`, `docId`	string	`docId` sim	Perfil e Doc proprietário.
`name`	string	sim	Título.
`content`, `subTitle`	string	não	Corpo e subtítulo iniciais.
`parentPageId`	string	não	Página pai.
`orderindex`	número	não	Posição entre páginas irmãs.
`contentFormat`	string	não	Forma do conteúdo.

Exemplos:

```
{"docId":"doc-1","name":"Introdução"}
```

```
{"docId":"doc-1","name":"Autenticação","content":"Use uma API key pessoal."}
```

```
{"docId":"doc-1","name":"Detalhes","subTitle":"Campos e respostas","content":"## Parâmetros"}
```

```
{"docId":"doc-1","name":"Erros","parentPageId":"page-1","orderindex":2}
```

```
{"account":"produto","docId":"doc-1","name":"Integração","content":"# MCP","contentFormat":"text/md"}
```

`clickupfy_doc_page_update`

Atualiza título, subtítulo e/ou conteúdo. O schema recusa chamada sem campo de alteração. `contentEditMode` aceita `replace`, `append` ou `prepend`; o padrão é `replace`. O modo afeta somente `content` e não o título ou subtítulo.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`, `docId`, `pageId`	string	Doc e página sim	Recurso a alterar.
`name`, `subTitle`, `content`	string	ao menos um	Campos a persistir.
`contentEditMode`	enum	não	`replace`, `append` ou `prepend`.
`contentFormat`	string	não	Forma do conteúdo.

Exemplos:

```
{"docId":"doc-1","pageId":"page-1","name":"Visão geral"}
```

```
{"docId":"doc-1","pageId":"page-1","subTitle":"Instalação e configuração"}
```

```
{"docId":"doc-1","pageId":"page-1","content":"# Novo conteúdo"}
```

```
{"docId":"doc-1","pageId":"page-1","content":"\n##
Changelog","contentEditMode":"append"}
```

```
{"account":"produto","docId":"doc-1","pageId":"page-2","content":"#
Aviso\n\n","contentEditMode":"prepend","contentFormat":"text/md"}
```

Leia a página com `clickupfy_doc_page_get` após atualizar. Não envie um objeto vazio para testar a conexão: use uma das ferramentas de leitura, pois uma atualização sem conteúdo é recusada.

Administração condicionada ao servidor

`clickupfy_account_use`

Define o perfil ativo local. Só aparece quando `mcp serve` não recebeu `--account` e o servidor não está em read-only. É a equivalência MCP de `clickupfy account use`; em projetos isolados, prefira fixar o perfil na configuração do processo para evitar troca global durante o trabalho.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	sim	Perfil local já configurado.

Exemplos:

```
{"account":"produto"}
```

```
{"account":"cliente-a"}
```

```
{"account":"homologacao"}
```

```
{"account":"suporte"}
```

```
{"account":"desenvolvimento"}
```

O retorno informa `activeAccount`. Esta ação muda a configuração da máquina, logo não deve ser usada para uma consulta pontual de outro perfil.

`clickupfy_workspace_use`

Associa workspace autorizado ao perfil. Só aparece quando o processo não fixou `--account` nem `--workspace` e tem escrita. O servidor confere a autorização contra a API antes de salvar a associação no perfil local.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a alterar quando omitido usa o ativo.
`workspaceId`	string	sim	Workspace autorizado para o perfil.

Exemplos:

```
{"workspaceId":"123456"}
```

```
{"account":"produto","workspaceId":"123456"}
```

```
{"account":"cliente-a","workspaceId":"987654"}
```

```
{"account":"homologacao","workspaceId":"246810"}
```

```
{"account":"suporte","workspaceId":"135791"}
```

Use `clickupfy_workspaces_list` para descobrir um ID autorizado. Uma tentativa com workspace ausente da lista é recusada e não altera a associação existente.

Referência MCP: tarefas, Sprints, checklists, comentários e tempo

Tarefas: leitura e busca

`clickupfy_tasks_list`

Lista tarefas da List autorizada. Omitir `listId` usa a List fixada. Os filtros aceitam arrays: `status` é uma lista de nomes configurados na List e `assignees` contém IDs de pessoas. `page` deve ser inteiro maior ou igual a zero.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`listId`	string	não	List fixa do projeto quando omitido.
`status`	string[]	não	Filtra por um ou mais status.
`assignees`	string[]	não	Filtra por responsáveis.
`includeClosed`	booleano	não	Inclui tarefas concluídas.
`page`	inteiro	não	Página, iniciando em zero.

Exemplos:

```
{}
```

```
{"status":["em andamento"]}
```

```
{"assignees":["42","57"],"includeClosed":false}
```

```
{"includeClosed":true,"page":1}
```

```
{"account":"produto","listId":"3001","status":["aberta","em andamento"]}
```

O retorno é compacto, com os campos usados para localizar trabalho. Para obter descrição, subtarefas e checklist items de uma tarefa, chame `clickupfy_task_get` com o `id` devolvido. Nunca use um `listId` diferente do fixado pelo projeto.

`clickupfy_task_get`

Obtém a tarefa e acrescenta `execution`, uma fila endereçável que preserva subtarefas e checklist items. `raw` devolve somente a resposta original da API. `markdown` devolve um texto único com descrição e itens; os dois formatos são mutuamente exclusivos.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`taskId`	string	sim	Tarefa raiz.
`raw`	booleano	não	Retorna somente o payload do ClickUp.
`markdown`	booleano	não	Renderiza tarefa e fila em Markdown.

Exemplos:

```
{"taskId":"86abc123"}
```

```
{"taskId":"86abc123","markdown":true}
```

```
{"taskId":"86abc123","raw":true}
```

```
{"account":"produto","taskId":"86abc123"}
```

```
{"account":"cliente-a","taskId":"77def456","markdown":true}
```

Os itens usam chaves como `task:86abc123` e `checklist:check-1:item-1`. `parentKey` e `depth` preservam a árvore. A ação `complete` de cada item mostra o comando CLI e a chamada MCP adequados, mas a autorização para executar essa ação continua vindo do pedido recebido pelo agente.

`clickupfy_tasks_search`

Busca somente dentro da List fixada, diferentemente do comando CLI `task search`, que percorre o workspace. `maxPages` aceita inteiro entre `1` e `100`. Omitir `query` lista de acordo com os outros filtros, portanto inclua um termo quando a intenção for uma busca nominal.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`query`	string	não	Texto no nome, descrição ou ID.

CAMPO	TIPO	OBRIGATÓRIO	USO
`status`	string[]	não	Filtra por status.
`assignees`	string[]	não	Filtra por responsáveis.
`includeClosed`	booleano	não	Inclui tarefas concluídas.
`maxPages`	inteiro	não	Paginação, de `1` a `100`.

Exemplos:

```
{"query":"autenticação"}
```

```
{"query":"86abc123"}
```

```
{"query":"API","status":["em andamento"]}
```

```
{"assignees":["42"],"includeClosed":true,"maxPages":5}
```

```
{"account":"produto","query":"checkout","maxPages":10}
```

Se duas tarefas forem plausíveis, apresente ID e nome e aguarde a escolha da pessoa que solicitou o trabalho. A ferramenta não altera tarefa e não expande a busca para uma List não autorizada.

`clickupfy_comments_list`

Lê comentários de uma tarefa. O retorno normaliza usuário, data e texto para facilitar a leitura pelo agente. A ferramenta não aceita paginação nem filtro de autor; leia a lista completa retornada e não trate um comentário antigo como instrução mais recente sem conferir as datas.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`taskId`	string	sim	Tarefa cujos comentários serão lidos.

Exemplos:

```
{"taskId":"86abc123"}
```

```
{"taskId":"77def456"}
```

```
{"account":"produto","taskId":"86abc123"}
```

```
{"account":"cliente-a","taskId":"77def456"}
```

```
{"account":"homologacao","taskId":"55ghi789"}
```

Leia essa ferramenta antes de publicar mudança de status ou comentário de progresso. Ela não expõe anexos como conteúdo, não publica respostas e não altera o campo de status.

Sprints

`clickupfy_sprints_list`

Lista Sprints dentro do Sprint Folder fixado. Uma Sprint é uma List que possui `start_date` e `due_date`; `includeRegular` acrescenta Lists comuns ao retorno de diagnóstico. `at` recebe uma data `AAAA-MM-DD` usada para classificar o estado do ciclo.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`folderId`	string	não	Sprint Folder fixado quando omitido.
`archived`	booleano	não	Inclui Lists arquivadas.
`includeRegular`	booleano	não	Inclui Lists que não são Sprints.
`at`	string	não	Data de referência em `AAAA-MM-DD`.

Exemplos:

```
{}
```

```
{"at":"2026-08-10"}
```

```
{"archived":true,"includeRegular":true}
```

```
{"folderId":"4001","at":"2026-09-01"}
```

```
{"account":"produto","folderId":"4001","includeRegular":false}
```

O objeto vazio funciona somente se o servidor recebeu `--sprint-folder`. A ferramenta não cria Sprint nem altera datas: a API pública do ClickUp não oferece criação de Sprint por esse fluxo.

`clickupfy_sprint_current`

Obtém a única Sprint cujo período contém a data informada ou a data atual. Se não houver Sprint ativa, ou se dois períodos se sobrepuserem, a ferramenta retorna erro em vez de escolher uma List arbitrariamente.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`folderId`	string	não	Sprint Folder fixado quando omitido.
`at`	string	não	Data de referência em `AAAA-MM-DD`.

Exemplos:

```
{}
```

```
{"at":"2026-08-10"}
```

```
{"folderId":"4001"}
```

```
{"folderId":"4001","at":"2026-09-01"}
```

```
{"account":"produto","folderId":"4001","at":"2026-10-15"}
```

Não use a falha como motivo para anexar tarefa a qualquer List do Folder. Corrija datas e sobreposição no ClickUp, depois leia novamente a Sprint atual.

`clickupfy_sprint_get`

Produz relatório de uma Sprint informada. Inclui período, progresso por quantidade de tarefas, progresso por Sprint Points e distribuição de status. `at` muda somente o cálculo temporal da situação da Sprint.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`sprintId`	string	sim	Sprint a analisar.
`at`	string	não	Data de referência em `AAAA-MM-DD`.

Exemplos:

```
{"sprintId":"sprint-10"}
```

```
{"sprintId":"sprint-10","at":"2026-08-10"}
```

```
{"sprintId":"sprint-11"}
```

```
{"account":"produto","sprintId":"sprint-10"}
```

```
{"account":"cliente-a","sprintId":"sprint-22","at":"2026-09-01"}
```

Uma tarefa sem Points participa do total de tarefas, mas não adiciona peso ao cálculo por pontos. Leia `clickupfy_sprint_tasks` para obter os itens do relatório, não tente inferi-los apenas pelas porcentagens.

`clickupfy_sprint_tasks`

Lista tarefas associadas a uma Sprint, incluindo concluídas por padrão. `openOnly: true` remove tarefas já concluídas e é útil para a fila presente, mas não substitui a lista integral usada na revisão final do ciclo.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`sprintId`	string	sim	Sprint consultada.
`openOnly`	booleano	não	Omite tarefas concluídas quando vale `true`.

Exemplos:

```
{"sprintId":"sprint-10"}
```

```
{"sprintId":"sprint-10","openOnly":true}
```

```
{"sprintId":"sprint-11"}
```

```
{"account":"produto","sprintId":"sprint-10","openOnly":false}
```

```
{"account":"cliente-a","sprintId":"sprint-22","openOnly":true}
```

A ferramenta retorna resumos de tarefa. Para descrição, subtarefas e checklist items de um item específico, chame `clickupfy_task_get` com o ID da tarefa.

Ferramentas de escrita de trabalho

As ferramentas seguintes só aparecem fora de read-only. Informe apenas campos que devem mudar. Elas usam o mesmo perfil e regras de isolamento das consultas.

`clickupfy_task_create`

Cria tarefa ou subtarefa na List fixa. `name` é obrigatório. Datas devem ser timestamps inteiros em milissegundos, diferentemente do CLI, que aceita datas `AAAA-MM-DD`.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`, `listId`	string	não	Perfil e List fixa quando omitida.
`name`	string	sim	Nome da tarefa.
`description`, `markdownContent`, `status`, `parent`	string	não	Conteúdo, status e pai.
`priority`	inteiro	não	De `1` a `4`.
`assignees`	número[]	não	Responsáveis.
`startDate`, `dueDate`	inteiro	não	Timestamp em milissegundos.
`points`	número	não	Sprint Points não negativos.

Exemplos:

```
{"name":"Corrigir retorno da API"}
```

```
{"name":"Documentar login","description":"Explicar a sessão."}
```

```
{"name":"Criar testes","markdownContent":"## Casos\n\n- Login válido"}
```

```
{"name":"Implementar token","status":"em andamento","priority":2,"assignees":
[42,57],"startDate":1786320000000,"dueDate":1786665600000,"points":5}
```

```
{"account":"produto","listId":"3001","name":"Cobrir
expiração","parent":"86abc123","markdownContent":"Adicionar testes de sessão
expirada."}
```

Leia a tarefa criada com `clickupfy_task_get`. Para `listId`, omitir é o caminho normal no MCP do projeto; um valor diferente do valor fixo é recusado.

`clickupfy_task_update`

Atualiza campos de uma tarefa. `taskId` é obrigatório e pelo menos um campo de alteração deve estar presente. Para remover data, use `null` em `startDate` ou `dueDate`; não envie texto vazio.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`, `taskId`	string	`taskId` sim	Perfil e tarefa.
`name`, `description`, `markdownContent`, `status`	string	não	Campos textuais.
`priority`	inteiro	não	De `1` a `4`.
`startDate`, `dueDate`	inteiro ou `null`	não	Nova data ou remoção.
`points`	número	não	Sprint Points não negativos.

Exemplos:

```
{"taskId":"86abc123","status":"em andamento"}
```

```
{"taskId":"86abc123","name":"Corrigir autenticação por token"}
```

```
{"taskId":"86abc123","priority":1,"points":8}
```

```
{"taskId":"86abc123","startDate":1786320000000,"dueDate":1786665600000}
```

```
{"account":"produto","taskId":"86abc123","markdownContent":"## Entrega\n\nPublicar manual atualizado.","dueDate":null}
```

Depois da escrita, use `clickupfy_task_get`. A ferramenta devolve a tarefa atualizada, mas a releitura é importante quando outras alterações concorrem na mesma tarefa.

`clickupfy_checklist_create`

Cria checklist em uma tarefa. O retorno fornece o ID necessário em `clickupfy_checklist_item_create`. Itens não são criados automaticamente.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`taskId`	string	sim	Tarefa proprietária.
`name`	string	sim	Nome do checklist.

Exemplos:

```
{"taskId":"86abc123","name":"Testes"}
```

```
{"taskId":"86abc123","name":"Revisão de segurança"}
```

```
{"taskId":"77def456","name":"Publicação"}
```

```
{"account":"produto","taskId":"86abc123","name":"Aceite"}
```

```
{"account":"cliente-a","taskId":"77def456","name":"Regressão"}
```

`clickupfy_checklist_item_create`

Acrescenta item aberto a um checklist. `assignee`, quando informado, é um ID numérico. A ferramenta não recebe `taskId` porque o checklist já define a tarefa proprietária.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`checklistId`	string	sim	Checklist que receberá o item.
`name`	string	sim	Texto do item.
`assignee`	inteiro	não	Pessoa responsável pelo item.

Exemplos:

```
{"checklistId":"check-1","name":"Executar testes unitários"}
```

```
{"checklistId":"check-1","name":"Executar build"}
```

```
{"checklistId":"check-1","name":"Revisar documentação","assignee":42}
```

```
{"account":"produto","checklistId":"check-2","name":"Conferir changelog"}
```

```
{"account":"cliente-a","checklistId":"check-3","name":"Validar
publicação","assignee":57}
```

`clickupfy_checklist_item_set`

Marca ou reabre um item e confirma o estado por uma nova leitura da tarefa. `taskId` deve ser a tarefa raiz devolvida por `clickupfy_task_get`, `resolved: true` conclui e `resolved: false` reabre.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`taskId`	string	sim	Tarefa raiz usada na conferência.
`checklistId`	string	sim	Checklist proprietário.
`itemId`	string	sim	Item a atualizar.
`resolved`	booleano	sim	`true` conclui; `false` reabre.

Exemplos:

```
{"taskId":"86abc123","checklistId":"check-1","itemId":"item-1","resolved":true}
```

```
{"taskId":"86abc123","checklistId":"check-1","itemId":"item-1","resolved":false}
```

```
{"account":"produto","taskId":"86abc123","checklistId":"check-2","itemId":"item-4","resolved":tr
```

```
{"account":"cliente-
a","taskId":"77def456","checklistId":"check-3","itemId":"item-2","resolved":true}
```

```
{"taskId":"77def456","checklistId":"check-3","itemId":"item-2","resolved":false}
```

Não passe um item de outra tarefa. A ferramenta faz essa conferência e recusa a escrita quando a relação não corresponde à árvore lida.

`clickupfy_sprint_add_task`

Associa tarefa a Sprint sem trocar sua List principal. Ela não cria tarefa e não muda seu status. A ferramenta só aparece em servidor com escrita.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`sprintId`	string	sim	Sprint de destino.
`taskId`	string	sim	Tarefa a associar.

Exemplos:

```
{"sprintId":"sprint-10","taskId":"86abc123"}
```

```
{"sprintId":"sprint-10","taskId":"77def456"}
```

```
{"account":"produto","sprintId":"sprint-10","taskId":"86abc123"}
```

```
{"account":"cliente-a","sprintId":"sprint-22","taskId":"55ghi789"}
```

```
{"account":"homologacao","sprintId":"sprint-33","taskId":"99jkl012"}
```

Leia `clickupfy_sprint_tasks` depois da escrita. Só associe a tarefa quando o planejamento da Sprint estiver incluído no pedido.

`clickupfy_sprint_remove_task`

Remove uma associação com Sprint sem excluir a tarefa. Comentários, checklist, tempo e List principal permanecem na tarefa.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`sprintId`	string	sim	Sprint cuja associação será removida.
`taskId`	string	sim	Tarefa associada.

Exemplos:

```
{"sprintId":"sprint-10","taskId":"86abc123"}
```

```
{"sprintId":"sprint-10","taskId":"77def456"}
```

```
{"account":"produto","sprintId":"sprint-10","taskId":"86abc123"}
```

```
{"account":"cliente-a","sprintId":"sprint-22","taskId":"55ghi789"}
```

```
{"account":"homologacao","sprintId":"sprint-33","taskId":"99jkl012"}
```

Use `clickupfy_task_delete` somente quando a exclusão inteira estiver explicitamente autorizada.

`clickupfy_sprint_set_points`

Define Sprint Points de uma tarefa. `points` é um número maior ou igual a zero. O ClickUpfy transmite o valor; a escala pertence ao time.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`taskId`	string	sim	Tarefa que receberá pontos.
`points`	número	sim	Valor não negativo.

Exemplos:

```
{"taskId":"86abc123","points":0}
```

```
{"taskId":"86abc123","points":1}
```

```
{"taskId":"86abc123","points":3}
```

```
{"account":"produto","taskId":"86abc123","points":5}
```

```
{"account":"cliente-a","taskId":"77def456","points":8}
```

Leia a tarefa com `clickupfy_task_get` para confirmar o número persistido.

`clickupfy_task_delete`

Exclui permanentemente uma tarefa. O campo `confirm` só aceita o literal `true`; essa exigência impede uma exclusão causada por objeto incompleto ou ambíguo. A ferramenta não está disponível em read-only.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`taskId`	string	sim	Tarefa a excluir.
`confirm`	`true`	sim	Confirmação literal obrigatória.

Exemplos:

```
{"taskId":"86abc123","confirm":true}
```

```
{"taskId":"77def456","confirm":true}
```

```
{"account":"produto","taskId":"86abc123","confirm":true}
```

```
{"account":"cliente-a","taskId":"77def456","confirm":true}
```

```
{"account":"homologacao","taskId":"55ghi789","confirm":true}
```

Leia tarefa, nome e List e obtenha autorização explícita para exclusão antes de qualquer exemplo deste tipo. Para somente retirar a tarefa do ciclo, use `clickupfy_sprint_remove_task`.

`clickupfy_comment_create`

Publica comentário em uma tarefa. `notifyAll` é opcional; ele pede ao ClickUp para notificar participantes. A ferramenta não modifica status, timer ou checklist.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`taskId`	string	sim	Tarefa que receberá o comentário.
`text`	string	sim	Conteúdo do comentário.
`notifyAll`	booleano	não	Pede notificação aos participantes.

Exemplos:

```
{"taskId":"86abc123","text":"Iniciei a análise da tarefa."}
```

```
{"taskId":"86abc123","text":"Os testes unitários passaram."}
```

```
{"taskId":"86abc123","text":"Aguardando acesso ao ambiente de
homologação.","notifyAll":true}
```

```
{"account":"produto","taskId":"86abc123","text":"Documentação e ebook foram
atualizados."}
```

```
{"account":"cliente-a","taskId":"77def456","text":"Validação final
concluída.","notifyAll":true}
```

`clickupfy_time_current`

Consulta o time entry que está em execução no workspace. Não recebe filtros; omitir `account` usa o perfil fixado ou ativo.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil cujo workspace será consultado.

Exemplos:

```
{}
```

```
{"account":"produto"}
```

```
{"account":"cliente-a"}
```

```
{"account":"homologacao"}
```

```
{"account":"suporte"}
```

`clickupfy_time_start`

Inicia um time entry para uma tarefa. Consulte `clickupfy_time_current` antes, pois a ferramenta não encerra automaticamente um registro que já esteja ativo.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil a resolver.
`taskId`	string	sim	Tarefa associada ao tempo.
`description`	string	não	Descrição curta do trabalho.

Exemplos:

```
{"taskId":"86abc123"}
```

```
{"taskId":"86abc123","description":"Implementação"}
```

```
{"taskId":"86abc123","description":"Testes de regressão"}
```

```
{"account":"produto","taskId":"86abc123","description":"Revisão de documentação"}
```

```
{"account":"cliente-a","taskId":"77def456","description":"Correção da integração"}
```

`clickupfy_time_stop`

Encerra o time entry atual do workspace. Ela não recebe `taskId` e não modifica status nem comentários. Relê `clickupfy_time_current` depois da chamada quando o encerramento precisa ser confirmado.

CAMPO	TIPO	OBRIGATÓRIO	USO
`account`	string	não	Perfil cujo registro atual será encerrado.

Exemplos:

```
{}
```

```
{"account":"produto"}
```

```
{"account":"cliente-a"}
```

```
{"account":"homologacao"}
```

```
{"account":"suporte"}
```

Consulte a referência do CLI

Sintaxe global

```
clickupfy [options] [command]
```

O comando de manutenção da instalação npm usa esta forma:

```
clickupfy upgrade [alvo]
```

As opções globais selecionam o account e o formato da resposta antes que o CLI interprete o grupo e a ação:

OPÇÃO	EFEITO
`-V`, `--version`	Mostra a versão.
`--account <perfil>`	Usa outro account somente nessa execução.
`--json`	Imprime a resposta completa em JSON.
`-h`, `--help`	Mostra ajuda.

Coloque as opções globais antes do grupo. No exemplo abaixo, a resposta de `task get` usa o account `cliente-a` e sai em JSON:

```
clickupfy --account cliente-a --json task get 86abc123
```

Configuração e identidade

```
clickupfy install [options]
clickupfy doctor
clickupfy status
clickupfy whoami
clickupfy account list
clickupfy account show [perfil]
clickupfy account use <perfil>
clickupfy account remove <perfil>
clickupfy workspace list
clickupfy workspace use <workspace-id>
```

Para atualizar uma instalação global feita pelo npm, use `clickupfy upgrade`. Consulte a referência de perfis e hierarquia para parâmetros, canais aceitos, confirmação da versão e limites do comando.

O setup recebe a credencial, identifica o account local e associa o workspace. Estes parâmetros permitem executar o mesmo processo com ou sem prompts:

OPÇÃO	EFEITO
`--api-key <chave>`	Informa a API key pessoal do ClickUp.
`--token <chave>`	Mantém compatibilidade como alias de `--api-key`.
`--name <nome>`	Define o nome local do account.
`--workspace <id>`	Associa o account ao workspace indicado.
`--non-interactive`	Executa sem abrir prompts e exige os valores necessários por opção.

`account remove <perfil>` aceita `--yes` para confirmar a remoção sem prompt. As outras ações desse grupo não possuem opções próprias.

`clickupfy doctor` verifica localmente o JSON em `~/clickupfy/config.json`, as permissões, o schema, os accounts, o gerenciador `skills`, as skills do ClickUpfy e os arquivos `.mcp.json` e `.codex/config.toml` do projeto quando existirem. Use `--json` para automação. O comando não faz chamada ao ClickUp. Use `clickupfy whoami` para validar a API key remotamente.

Hierarquia

```
clickupfy space list [--archived]
clickupfy folder list --space <id> [--archived]
clickupfy list list --folder <id> [--archived]
clickupfy list list --space <id> [--archived]
clickupfy list get <list-id>
```

`space list`, `folder list` e `list list` aceitam `--archived`. O comando `folder list` exige `--space <id>`. Em `list list`, informe `--folder <id>` ou `--space <id>`, pois os dois destinos são mutuamente exclusivos e a resposta precisa pertencer a um único nível da hierarquia.

Tarefas

```
clickupfy task list --list <list-id>
clickupfy task search --query <texto>
clickupfy task get <task-id>
clickupfy task get <task-id> --markdown
clickupfy task get <task-id> --raw
clickupfy task create [options]
clickupfy task update <task-id> [options]
clickupfy task delete <task-id> [--yes]
```

Filtros de leitura

COMANDO	OPÇÃO	EFEITO
`task list`	`--list <id>`	Define a List consultada e é obrigatório.
`task list`	`--status <status...>`	Restringe a resposta aos status informados.
`task list`	`--assignee <ids...>`	Restringe a resposta aos responsáveis informados.
`task list`	`--include-closed`	Inclui tarefas concluídas.
`task list`	`--page <n>`	Consulta uma página, começando em zero.
`task search`	`--query <texto>`	Procura no nome, na descrição ou no ID.
`task search`	`--status <status...>`	Restringe a busca aos status informados.
`task search`	`--assignee <ids...>`	Restringe a busca aos responsáveis informados.
`task search`	`--include-closed`	Inclui tarefas concluídas.
`task search`	`--max-pages <n>`	Limita a paginação consultada entre uma e 100 páginas.
`task get`	`--raw`	Retorna a resposta original do ClickUp.
`task get`	`--markdown`	Concatena tarefa, subtarefas e checklists em Markdown.

`--raw` e `--markdown` não podem ser usados juntos. A opção global `--json` também é incompatível com esses dois formatos, e o CLI encerra a execução com uma mensagem de uso quando encontra a combinação.

Campos de criação

OPÇÃO	EFEITO
`--list <id>`	Define a List de destino e é obrigatório.
`--name <nome>`	Define o nome e é obrigatório.
`--description <texto>`	Envia uma descrição em texto.
`--markdown-content <markdown>`	Envia uma descrição preservada como Markdown.
`--status <status>`	Define o status inicial.
`--priority <n>`	Define prioridade de `1` a `4`.
`--assignee <ids...>`	Informa IDs numéricos dos responsáveis.
`--parent <task-id>`	Cria uma subtarefa ligada ao item informado.
`--start-date <AAAA-MM-DD>`	Define a data de início.
`--due-date <AAAA-MM-DD>`	Define a data de entrega.
`--points <n>`	Define Sprint Points com número igual ou maior que zero.

Escolha `--description` para texto ou `--markdown-content` para Markdown. Evite enviar os dois campos na mesma criação para não deixar a representação da descrição ambígua.

Campos de atualização

OPÇÃO	EFEITO
`--name <nome>`	Altera o nome.
`--description <texto>`	Substitui a descrição por texto.
`--markdown-content <markdown>`	Substitui a descrição por Markdown.
`--status <status>`	Altera o status.
`--priority <n>`	Altera a prioridade de `1` a `4`.
`--start-date <AAAA-MM-DD>`	Altera a data de início.
`--clear-start-date`	Remove a data de início.
`--due-date <AAAA-MM-DD>`	Altera a data de entrega.
`--clear-due-date`	Remove a data de entrega.
`--points <n>`	Altera os Sprint Points.

O comando exige pelo menos um campo. Quando uma data e sua opção `--clear-*` aparecem na mesma execução, a remoção prevalece. `task delete` usa `--yes` para confirmar a exclusão sem prompt.

Depois de escolher os campos, consulte a ajuda da versão instalada para comparar a referência com o contrato executável:

```
clickupfy task create --help
clickupfy task update --help
```

Checklists e comentários

```
clickupfy checklist create <task-id> --name <nome>
clickupfy checklist item-create <checklist-id> --name <nome>
clickupfy checklist set \
  <task-id> <checklist-id> <item-id> \
  --resolved
clickupfy checklist set \
  <task-id> <checklist-id> <item-id> \
  --open
clickupfy comment list --task <task-id>
clickupfy comment create --task <task-id> --text <texto>
```

`checklist item-create` aceita `--assignee <id>` para associar o item a um responsável. Em `checklist set`, escolha exatamente `--resolved` ou `--open` para que a releitura confirme o estado pretendido. `comment create` aceita `--notify-all` quando a atualização precisa notificar os participantes da tarefa.

Sprints

```
clickupfy sprint list --folder <folder-id>
clickupfy sprint current --folder <folder-id>
clickupfy sprint get <sprint-id>
clickupfy sprint tasks <sprint-id> [--open-only]
clickupfy sprint add-task <sprint-id> <task-id>
clickupfy sprint remove-task <sprint-id> <task-id>
clickupfy sprint set-points <task-id> <points>
```

Os comandos de Sprint aceitam opções de calendário, inclusão e estado para separar uma List comum do ciclo que será analisado:

COMANDO	OPÇÃO	EFEITO
`sprint list`	`--archived`	Inclui Lists arquivadas.
`sprint list`	`--include-regular`	Inclui Lists sem período do Sprint Folder.
`sprint list`	`--at <AAAA-MM-DD>`	Calcula o estado do ciclo na data informada.
`sprint current`	`--at <AAAA-MM-DD>`	Procura a Sprint ativa na data informada.
`sprint get`	`--at <AAAA-MM-DD>`	Calcula o relatório na data informada.
`sprint tasks`	`--open-only`	Omite tarefas concluídas.

`sprint list` e `sprint current` exigem `--folder <id>` para limitar a busca ao Sprint Folder selecionado.

Time tracking

```
clickupfy time current
clickupfy time start --task <task-id> [--description <texto>]
clickupfy time stop
```

Docs

```
clickupfy doc list [--query <texto>] [options]
clickupfy doc get <doc-id>
clickupfy doc create --name <nome> [options]
clickupfy doc page tree <doc-id>
clickupfy doc page list <doc-id> [options]
clickupfy doc page get <doc-id> <page-id> [--content-format <valor>]
clickupfy doc page create <doc-id> --name <nome> [options]
clickupfy doc page update <doc-id> <page-id> [options]
```

Docs usam a API v3 do ClickUp e sempre operam no workspace do account ativo, sem depender de Space, Folder ou List.

COMANDO	OPÇÃO	EFEITO
`doc list`	`--query <texto>`	Filtra pelo nome ou ID localmente.
`doc list`	`--parent-id <id>`	Restringe aos Docs criados sob este local.
`doc list`	`--parent-type <n>`	Tipo do local: `4` Space, `5` Folder, `6` List, `7` Everything, `12` tarefa.

COMANDO	OPÇÃO	EFEITO
`doc list`	`--deleted`	Inclui Docs excluídos.
`doc list`	`--archived`	Inclui Docs arquivados.
`doc list`	`--creator <id>`	Restringe ao ID numérico do criador.
`doc list`	`--max-pages <n>`	Limita a paginação por cursor, entre uma e 50 páginas.
`doc create`	`--parent-id <id>`	Local onde o Doc nasce; exige `--parent-type`.
`doc create`	`--parent-type <n>`	Tipo do local; exige `--parent-id`.
`doc create`	`--visibility <valor>`	`PRIVATE` ou `PUBLIC`.
`doc create`	`--create-page`	Cria também a primeira página em branco.
`doc page list`	`--max-page-depth <n>`	Limita a profundidade de sub-páginas retornadas.
`doc page list` / `doc page get`	`--content-format <valor>`	`text/md` (padrão) ou `text/plain`.
`doc page create`	`--content <texto>`	Conteúdo inicial da página.
`doc page create`	`--sub-title <texto>`	Subtítulo da página.
`doc page create`	`--parent-page <id>`	Cria como sub-página deste ID.
`doc page create`	`--orderindex <n>`	Posição entre as páginas irmãs.
`doc page update`	`--content-edit-mode <modo>`	`replace` (padrão), `append` ou `prepend`.

`doc page tree` mostra a hierarquia de páginas sem conteúdo, o caminho mais rápido para localizar um `page-id`. `doc page update` exige pelo menos um campo entre `--name`, `--sub-title` e `--content`.

A API pública do ClickUp não oferece endpoints para excluir Docs ou páginas, reordenar páginas na árvore, nem gerenciar permissões de compartilhamento.

Agentes e MCP

```
clickupfy agent skill list
clickupfy agent skill show <nome>
clickupfy agent skill install [--global] [--force]
clickupfy agent install [options]
clickupfy mcp serve --list <id> [options]
```

`agent skill install` aceita uma lista opcional de skills e chama `npx skills add promovaweb/clickupfy --agent codex --copy --yes`. Sem `--global`, o gerenciador usa `.agents/skills/` e cria ou atualiza `skills-lock.json` no projeto. Com `--global`, usa `~/.codex/skills/`. Para consultar o estado real, use `npx skills list` ou `npx skills list --global`; `clickupfy agent skill list` exibe somente o catálogo distribuído pelo ClickUpfy.

O comando não oferece `--target`: o gerenciador de skills controla os caminhos canônicos. `--force` permanece aceito por compatibilidade, mas a sincronização é feita pelo próprio `npx skills add`.

`agent install` exige `--space <id>` e `--list <id>`. Ele também aceita `--global`, `--workspace <id>`, `--folder <id>`, `--sprint-folder <id>` e `--force`. O comando atualiza o servidor `promovaweb-clickupfy` no `.mcp.json` e no `.codex/config.toml`, preservando as outras entradas de cada arquivo.

`mcp serve` exige `--list <id>` e aceita `--account <perfil>`, `--workspace <id>`, `--space <id>`, `--folder <id>`, `--sprint-folder <id>` e `--read-only`.

Escolha o formato de saída

A tabela compacta é adequada para leitura humana. Use `--json` quando um script precisar de campos completos ou quando a tabela não mostrar um detalhe da API.

Use `task get --markdown` para contexto textual. Use `--raw` apenas quando precisar comparar a resposta original do ClickUp com a projeção do ClickUpfy.

Ajuda é a referência da versão instalada

Esta documentação explica o contrato estável. Para flags exatas da versão presente na máquina:

```
clickupfy --help
clickupfy <grupo> --help
clickupfy <grupo> <ação> --help
```

Quando a ajuda estiver correta, mas a execução falhar, continue no guia de solução de problemas para conferir instalação, credencial, escopo e resposta da API.

Resolva falhas comuns

O comando não foi encontrado

Execute:

```
npm prefix --global
```

Confirme se a pasta de executáveis globais está no `PATH`. Em instalação standalone, confira permissão de execução e o nome do arquivo.

A API key foi recusada

Use:

```
clickupfy whoami
```

Se falhar, gere ou confira a chave no ClickUp e execute `clickupfy install` novamente. Uma chave revogada não pode ser recuperada pelo ClickUpfy.

Nunca publique a chave para pedir suporte. Informe apenas o account, o workspace mascarado quando necessário e a mensagem de erro sem headers.

O workspace está incorreto

```
clickupfy workspace list
clickupfy workspace use <workspace-id>
clickupfy status
```

Se o MCP fixa `--workspace`, atualize a configuração do projeto para o mesmo ID ou selecione o account correto.

Folder ou List não aparece

Confirme o Space e tente incluir arquivados:

```
clickupfy folder list --space <space-id> --archived
clickupfy list list --folder <folder-id> --archived
```

Para List sem Folder, use `--space`. Permissões da API key também podem ocultar recursos.

O status foi rejeitado

```
clickupfy list get <list-id>
```

Copie a grafia de um status retornado. Status são configuráveis por List.

O MCP não inicia

Execute manualmente o comando registrado no `.mcp.json` ou no `.codex/config.toml`. Os motivos mais comuns são:

- `clickupfy` ausente no `PATH` do cliente
- JSON inválido
- account inexistente
- `--list` sem valor
- workspace fixado diferente do perfil
- cliente não reiniciado depois da edição.

O MCP recusou um ID

Essa recusa é esperada quando a ferramenta recebe um ID diferente do escopo do projeto. Consulte:

```
clickupfy_mcp_context
```

Use o destino fixado ou altere o arquivo do cliente conscientemente. Não contorne a proteção repetindo a chamada com outro recurso.

Uma ferramenta de escrita não aparece

O servidor pode estar em read-only. Confira os argumentos. Remover `--read-only` amplia as permissões do cliente e deve ser uma decisão explícita.

A Sprint atual não foi encontrada

Liste o Folder com períodos:

```
clickupfy sprint list --folder <sprint-folder-id> --include-regular
```

Confirme se existe exatamente uma Sprint cujo período inclui a data atual. Corrija datas ou sobreposição na interface do ClickUp.

O item de checklist não mudou

Releia a tarefa:

```
clickupfy task get <task-id> --json
```

Confirme task ID, checklist ID e item ID. O comando `checklist set` rejeita um item que não pertença à tarefa informada.

O time entry já está em execução

```
clickupfy time current
clickupfy time stop
```

Confira o item antes de encerrar. Depois inicie o registro correto.

A saída compacta não mostra um campo

Repita com `--json`:

```
clickupfy --json task get <task-id>
```

Para uma tarefa longa consumida por agente, prefira `--markdown`.

Diagnóstico seguro

Ao relatar um problema, inclua:

- versão de `clickupfy --version`
- plataforma e forma de instalação
- grupo e ação executados
- mensagem de erro
- se o modo era CLI ou MCP
- escopo sem credenciais.

Remova API keys, tokens, headers, conteúdo de clientes e dados pessoais.

Volte ao início do guia ou consulte a referência do CLI.