Arquitetando a Extensibilidade: Uma Análise Abrangente da Integração do Protocolo de Contexto de Modelo (MCP) no Gemini CLI

Seção 1: O Protocolo de Contexto de Modelo como a Camada de Extensibilidade para o Gemini CLI

Esta seção estabelece o conhecimento fundamental necessário para compreender o restante do relatório. Ela define o Protocolo de Contexto de Modelo (MCP) e explica seu papel crítico na arquitetura de agentes de IA modernos como o Gemini CLI.

1.1 Definindo o Protocolo de Contexto de Modelo (MCP)

O Protocolo de Contexto de Modelo (MCP) é um protocolo de código aberto projetado para padronizar a comunicação entre Modelos de Linguagem Grandes (LLMs), como o Gemini, e ferramentas e serviços externos.[1] Ele funciona como uma "ponte", permitindo que a IA acesse informações do mundo real e execute ações que vão além de seu conhecimento estático e incorporado.[1] Essa capacidade transforma o Gemini CLI de um simples assistente de codificação em um participante ativo e versátil no fluxo de trabalho de um desenvolvedor, capaz de ler arquivos, escrever código, executar comandos e automatizar tarefas complexas.[2] O suporte ao MCP tem sido uma característica central do Gemini CLI desde seu lançamento inicial, o que sinaliza sua importância estratégica para a funcionalidade da ferramenta.[5]

1.2 A Importância Arquitetural do MCP no Gemini CLI

A arquitetura do Gemini CLI emprega um ciclo de "raciocínio e ação" (ReAct), utilizando servidores MCP para completar casos de uso complexos, como corrigir bugs, criar novas funcionalidades ou melhorar a cobertura de testes.[4] Ao construir ou se conectar a um servidor MCP, um desenvolvedor está efetivamente criando um plugin personalizado, ensinando ao Gemini CLI novas habilidades.[1] Essa extensibilidade é o que eleva a ferramenta de um chatbot para um verdadeiro agente de IA.[2] Para facilitar esse processo, frameworks como o FastMCP para Python abstraem as complexidades de baixo nível da especificação do MCP, tornando mais acessível para os desenvolvedores a criação de suas próprias ferramentas.[6]

A integração do MCP representa uma mudança de paradigma na forma como os desenvolvedores interagem com suas ferramentas de linha de comando. Em vez de apenas gerar código, o Gemini CLI pode interagir com o sistema de arquivos local (usando ferramentas como read_file e write_file), a web (com web_fetch) e uma gama de serviços externos via MCP.[4] Quando servidores para plataformas como GitHub, Firebase e Netdata são adicionados, o usuário não está mais apenas "conversando" com uma IA; ele está usando linguagem natural para orquestrar um conjunto complexo de ferramentas locais e remotas.[5] Isso transforma a CLI em um hub de comando central, análogo a um shell de sistema operacional, mas que compreende a intenção do usuário em vez de apenas comandos explícitos.

Além disso, a natureza de código aberto do protocolo MCP é um catalisador estratégico para um ecossistema multiplataforma. Um servidor construído para o Gemini CLI poderia, teoricamente, funcionar com outro cliente compatível com MCP, como o Claude Code, da Anthropic.[11] Isso fomenta um cenário competitivo e inovador onde o valor reside no ecossistema de ferramentas, não apenas no cliente de IA. A existência de catálogos como

mcpservers.org é uma evidência direta desse ecossistema florescente e agnóstico de plataforma, o que implica que o investimento na construção de um servidor MCP tem uma utilidade mais ampla que transcende o Gemini CLI.[13]

Seção 2: Mecanismo Central: Configuração Programática com gemini mcp add

Esta seção constitui o núcleo do relatório, fornecendo uma análise definitiva do comando gemini mcp add. Este comando é apresentado como o método moderno e simplificado para a configuração de servidores, atendendo a uma solicitação de funcionalidade para uma

configuração programática.[14]

2.1 A Evolução da Edição Manual para um Comando Dedicado

Historicamente, a configuração de servidores MCP era realizada através da edição manual do arquivo ~/.gemini/settings.json.[1] O comando

gemini mcp add foi introduzido como uma alternativa mais amigável e programável que elimina a necessidade de manipulação direta de JSON, alinhando-se a uma tendência mais ampla de melhorar a experiência do desenvolvedor em ferramentas de linha de comando.[14] No entanto, observa-se uma defasagem entre a implementação dessa funcionalidade e sua documentação generalizada; a maioria dos tutoriais oficiais e de terceiros ainda detalha o método de edição manual.[1] A fonte mais definitiva para a sintaxe do comando

gemini mcp add permanece sendo um registro de issue no GitHub [14], tornando este relatório um recurso crucial para preencher essa lacuna de conhecimento.

2.2 Conectando a Servidores Locais (transporte stdio)

O mecanismo de transporte stdio (entrada/saída padrão) é utilizado para processos locais, onde o Gemini CLI se comunica diretamente com o executável do servidor através de seus fluxos de entrada e saída padrão. Este é o método principal para integrar scripts ou binários locais.

- **Sintaxe:** A sintaxe oficial é gemini mcp add <name> <command> [args...].[14]
- **Detalhamento dos Parâmetros:**
  - <name>: O identificador único para o servidor dentro do Gemini CLI.
  - <command>: O comando executável a ser executado (ex: npx, node, /usr/sbin/nd-mcp).
  - [args...]: Argumentos opcionais passados para o comando.
  - -e VAR=VALUE: Um sinalizador para definir variáveis de ambiente para o processo do servidor, crucial para passar segredos como chaves de API.[14]
  - --: Um separador para distinguir as opções do comando gemini mcp add do comando e argumentos do servidor.[14]
- **Exemplos Práticos:**
  - **Servidor Local Genérico:** gemini mcp add my-server -e API_KEY=123 -- /path/to/server arg1 arg2.[14]

- **Netdata:** gemini mcp add netdata /usr/sbin/nd-mcp ws://YOUR_NETDATA_IP:19999/mcp?api_key=NETDATA_MCP_API_KEY (Sintaxe derivada de [9]).
- **Firebase:** gemini mcp add firebase npx -y firebase-tools@latest experimental:mcp (Sintaxe derivada de [10]).

2.3 Conectando a Servidores Remotos (transporte HTTP)

O transporte HTTP é usado para conectar-se a servidores MCP expostos através de uma URL web padrão. Este é o método comum para se conectar a APIs de terceiros e serviços gerenciados, sendo o transporte recomendado para implantações baseadas na web.[6]

- **Sintaxe:** A sintaxe oficial é gemini mcp add --transport http <name> <url>.[14]
- **Detalhamento dos Parâmetros:**
  - --transport http: O sinalizador que especifica o protocolo.
  - <name>: O identificador do servidor.
  - <url>: A URL completa do endpoint MCP.
  - --header "Key: Value": Uma opção crucial para passar tokens de autenticação ou outros cabeçalhos necessários.[14]
- **Exemplos Práticos:**
  - **Servidor HTTP Genérico:** gemini mcp add --transport http http-server https://example.com/mcp.[14]
  - **Servidor GitHub Autenticado:** gemini mcp add --transport http github https://api.githubcopilot.com/mcp/ --header "Authorization: Bearer <YOUR_GITHUB_PAT>" (Sintaxe derivada de [5]).

2.4 Conectando a Servidores de Streaming (transporte SSE)

O transporte Server-Sent Events (SSE) permite que um servidor envie dados para o cliente de forma contínua, sendo útil para serviços que fornecem atualizações em tempo real.

- **Sintaxe:** A sintaxe oficial é gemini mcp add --transport sse <name> <url>.[14]
- **Detalhamento dos Parâmetros:**
  - --transport sse: O sinalizador que especifica o protocolo SSE.
  - --header "Key: Value": Semelhante ao HTTP, permite o uso de cabeçalhos personalizados.
- **Exemplos Práticos:**

- **Servidor SSE Genérico:** gemini mcp add --transport sse sse-server https://example.com/sse-endpoint.[14]
- **Servidor SSE Autenticado:** gemini mcp add --transport sse api-server https://api.example.com/sse --header "X-API-Key: your-key".[14]

O suporte a múltiplos modelos de transporte é uma decisão arquitetural fundamental. Ele permite que o Gemini CLI atue como uma ponte segura entre a máquina local de um desenvolvedor e a internet pública. Um desenvolvedor pode criar uma ferramenta privada e altamente segura que interage com seu sistema de arquivos local e registrá-la via stdio, enquanto simultaneamente se conecta a uma poderosa API pública, como a do GitHub, via http. Este modelo híbrido oferece um nível de flexibilidade e segurança que um agente puramente baseado em nuvem ou puramente local não conseguiria alcançar, concedendo ao agente tanto um contexto local profundo quanto um amplo alcance externo.

Seção 3: Um Catálogo Global de Serviços MCP Compatíveis e Modelos de Integração

Esta seção fornece um recurso prático e valioso: um catálogo selecionado de servidores MCP conhecidos e compatíveis com o Gemini CLI. Para cada serviço, é apresentado um modelo de integração completo.

3.1 Integrações Oficiais e de Primeira Parte

- **GitHub:**
  - **Descrição:** O servidor MCP oficial do GitHub permite a interação com repositórios, issues e pull requests.[5]
  - **Detalhes de Conexão:** É um servidor remoto. A URL é https://api.githubcopilot.com/mcp/. Requer um cabeçalho Authorization com um Token de Acesso Pessoal (PAT).[5]
  - **Exemplo:** gemini mcp add --transport http github https://api.githubcopilot.com/mcp/ --header "Authorization: Bearer <YOUR_GITHUB_PAT>"
- **Firebase:**
  - **Descrição:** O servidor MCP do Firebase fornece ferramentas para gerenciar projetos do Firebase, Firestore, Auth, Data Connect e mais.[10]
  - **Detalhes de Conexão:** É um servidor local invocado via npx que se comunica por stdio.[10]

    - **Exemplo:** gemini mcp add firebase npx -y firebase-tools@latest experimental:mcp

3.2 Servidores de Terceiros e da Comunidade

- **Netdata:**
    - **Descrição:** Conecta o Gemini CLI ao agente de monitoramento Netdata para observabilidade de infraestrutura, permitindo consultas sobre desempenho do sistema, anomalias e consumo de recursos.[9]
    - **Detalhes de Conexão:** É um servidor local que atua como uma ponte. Requer o binário nd-mcp e se conecta via stdio. Os argumentos incluem a URL WebSocket do agente Netdata e uma chave de API.[9]
    - **Exemplo:** gemini mcp add netdata /path/to/nd-mcp ws://YOUR_NETDATA_IP:19999/mcp?api_key=NETDATA_MCP_API_KEY
- **Bright Data:**
    - **Descrição:** Fornece capacidades avançadas de web scraping e extração de dados, superando as limitações da ferramenta web_fetch nativa.[15]
    - **Detalhes de Conexão:** Um servidor local invocado via npx. Requer que um API_TOKEN seja passado como variável de ambiente.[15]
    - **Exemplo:** gemini mcp add brightData -e API_TOKEN=<YOUR_BRIGHT_DATA_API_KEY> -- npx -y @brightdata/mcp
- **O Ecossistema Mais Amplo:**
    - Existem listas selecionadas como mcpservers.org e a lista oficial no GitHub que servem como recursos para descobrir centenas de outros servidores construídos pela comunidade para diversas tarefas, como interação com bancos de dados, automação do Google Workspace e geração de imagens.[13]

3.3 Tabela: Tabela Abrangente de Integração de Servidores MCP

A tabela a seguir sintetiza informações de integração fragmentadas, consolidando pesquisas de várias fontes em um formato único, acionável e de fácil leitura.

Nome do Serviço	Descrição	Tipo de Conexão	Transporte	Exemplo de Comando gemini mcp add	Snippet settings.json

GitHub	Interage com repositórios, issues e PRs.	Remoto	HTTP	gemini mcp add --transport http github https://api.githubcopilot.com/mcp/ --header "Authorization: Bearer <YOUR_PAT>"	{ "httpUrl": "https://api.githubcopilot.com/mcp/", "headers": { "Authorization": "Bearer <YOUR_PAT>" } }
Firebase	Gerencia projetos do Firebase, Firestore, Auth.	Local	stdio	gemini mcp add firebase npx -y firebase-tools@latest experimental:mcp	{ "command": "npx", "args": ["-y", "firebase-tools@latest", "experimental:mcp"] }
Netdata	Acessa dados de observabilidade de infraestrutura.	Local	stdio	gemini mcp add netdata /path/to/nd-mcp ws://<IP>:19999/mcp?api_key=<KEY>	{ "command": "/path/to/nd-mcp", "args": }
Bright Data	Web scraping avançado e extração de dados.	Local	stdio	gemini mcp add brightData -e API_TOKEN=<KEY> -- npx -y	{ "command": "npx", "args": ["-y", "@brightdata/mcp"],

				@brightdata/mcp	"env": { "API_TOKEN": "<KEY>" } }
Servidor Python Customizado	Uma ferramenta local criada pelo usuário via FastMCP.	Remoto	HTTP	gemini mcp add --transport http my-tool http://127.0.0.1:8080/mcp/	{ "httpUrl": "http://127.0.0.1:8080/mcp/" }

Seção 4: Configuração Avançada e Manual via settings.json

Esta seção detalha o método manual de edição do arquivo ~/.gemini/settings.json, proporcionando uma compreensão mais profunda da estrutura de configuração subjacente.

4.1 A Estrutura do settings.json

O arquivo de configurações do Gemini CLI pode ser encontrado em ~/.gemini/settings.json para configurações globais ou em .gemini/settings.json para configurações específicas do projeto. As configurações do projeto têm precedência sobre as globais.[19] Dentro deste arquivo, o objeto

mcpServers contém um mapa de chave-valor onde cada chave é o nome do servidor e o valor é seu objeto de configuração.[1]

4.2 Detalhamento dos Parâmetros de Configuração

A seguir, uma lista abrangente dos parâmetros possíveis para um objeto de servidor:

- command (string, obrigatório para stdio): O comando a ser executado.[5]
- args (array de strings, opcional): Argumentos para o comando.[5]
- httpUrl (string, obrigatório para http/sse): A URL do endpoint.[5]
- headers (objeto, opcional): Mapa de chave-valor de cabeçalhos HTTP.[5]
- env (objeto, opcional): Variáveis de ambiente para o processo do servidor.[15]
- timeout (número, opcional): Tempo limite da requisição em milissegundos.[5]

4.3 Exemplos Paralelos: settings.json vs. gemini mcp add

A comparação lado a lado abaixo demonstra como o comando gemini mcp add simplifica o processo de configuração.

- **Exemplo para GitHub:**
  - **settings.json:**
    JSON
    ```
    {
      "mcpServers": {
        "github": {
          "httpUrl": "https://api.githubcopilot.com/mcp/",
          "headers": { "Authorization": "Bearer <YOUR_GITHUB_PAT>" },
          "timeout": 5000
        }
      }
    }
    ```

    (De [5])
  - **gemini mcp add:** gemini mcp add --transport http github https://api.githubcopilot.com/mcp/ --header "Authorization: Bearer <YOUR_GITHUB_PAT>"

Seção 5: Verificação, Solução de Problemas e Melhores Práticas

Esta seção equipa o usuário com o conhecimento para validar suas configurações e resolver problemas comuns.

5.1 Verificação e Descoberta

O comando /mcp dentro do Gemini CLI é a principal ferramenta de diagnóstico para o subsistema MCP. Ele lista todos os servidores configurados, seu status de conexão e as ferramentas disponíveis.[5] Subcomandos como

/mcp desc e /mcp schema fornecem informações mais detalhadas.[9] O uso deste comando deve ser o primeiro passo em qualquer processo de solução de problemas, pois ele pode diferenciar instantaneamente entre um erro de sintaxe na configuração, um problema de rede ou uma falha no processo do servidor.

5.2 Cenários Comuns de Solução de Problemas

- **Erros de Autenticação:** Chaves de API ou PATs incorretos e credenciais expiradas são causas comuns de falha. É essencial verificar os cabeçalhos e as variáveis de ambiente.[9]
- **Problemas de Conexão e Caminho:** URLs incorretas, firewalls de rede bloqueando o acesso ou caminhos incorretos para binários locais (command not found) podem impedir a conexão. Recomenda-se o uso de caminhos absolutos ou garantir que o binário esteja no PATH do sistema.[9]
- **Incompatibilidades de Protocolo:** Existe um problema conhecido em que algumas versões do Gemini CLI se recusavam a conectar a um servidor que não fornecia o campo opcional instructions em sua definição, mesmo que o servidor estivesse em conformidade com a especificação do MCP.[23] Isso representa uma armadilha crítica para desenvolvedores que constroem seus próprios servidores.
- **Problemas de Dependência:** Para servidores executados via npx ou scripts locais, é vital garantir que todas as dependências (como Node.js, Python ou pacotes específicos) estejam instaladas corretamente.[1]

O ritmo acelerado do ecossistema de servidores de terceiros pode, por vezes, superar a estabilidade do cliente Gemini CLI. Problemas de conexão podem não ser culpa do servidor, mas sim de um bug ou de uma interpretação estrita da especificação pelo cliente.[23] Portanto, verificar os issues oficiais do repositório do Gemini CLI no GitHub deve ser um passo padrão

na solução de problemas.

5.3 Melhores Práticas de Segurança

- **Gerenciamento de Credenciais:** É fortemente desaconselhado codificar segredos como chaves de API ou PATs diretamente no settings.json. A prática recomendada é usar variáveis de ambiente referenciadas na configuração (ex: "env": { "API_KEY": "$API_KEY" }) ou ferramentas de gerenciamento de segredos de nível de sistema.
- **Confiança nos Servidores:** Servidores MCP locais podem executar código arbitrário na máquina do usuário.[25] Os usuários devem adicionar apenas servidores de fontes confiáveis e revisar a configuração antes de habilitá-los.

Conclusão

O Protocolo de Contexto de Modelo (MCP) é a espinha dorsal da extensibilidade do Gemini CLI, transformando-o de uma ferramenta de codificação em um agente de IA versátil e poderoso. O comando gemini mcp add representa o método moderno e preferencial para configurar essas extensões, simplificando um processo que antes exigia a edição manual de arquivos JSON.

A análise revela um ecossistema rico e crescente de servidores MCP de primeira e terceira partes, capazes de conectar o Gemini CLI a uma vasta gama de serviços, desde o gerenciamento de repositórios no GitHub até a observabilidade de infraestrutura com o Netdata. A capacidade de suportar transportes locais (stdio) e remotos (http, sse) confere ao Gemini CLI uma flexibilidade arquitetural única, equilibrando segurança e alcance.

Para os usuários, dominar a integração com o MCP é fundamental para desbloquear todo o potencial do Gemini CLI. Isso requer não apenas compreender a sintaxe do comando gemini mcp add, mas também estar ciente da defasagem documental, utilizar o comando /mcp como uma ferramenta de diagnóstico primária e abordar a solução de problemas com a compreensão de que podem surgir problemas tanto no servidor quanto no cliente. Ao seguir as melhores práticas de segurança e aproveitar o ecossistema de ferramentas disponíveis, os desenvolvedores podem personalizar e expandir o Gemini CLI para criar um assistente de IA verdadeiramente adaptado aos seus fluxos de trabalho específicos.

Perfeito, Rogerio! Bora fechar esse ciclo no mesmo padrão dos anteriores. Aqui estão os comandos gemini mcp add para os MCPs novos que você pediu:

Testes e Automação de Navegador

gemini mcp add playwright -- npx -y @microsoft/playwright-mcp

Automatiza interações de navegador (cliques, formulários, testes E2E) via MCP.

Crawling e Scraping

gemini mcp add firecrawl -- npx -y @mendableai/firecrawl-mcp

Raspagem estruturada de sites (Markdown/JSON), ideal para dar contexto real ao LLM.

Gemini CLI via MCP

gemini mcp add gemini -- npx -y @blesscat/gemini-cli-mcp

Expõe comandos Gemini (chat, generate, list-models) como ferramentas MCP.

Monitoramento e Observabilidade

gemini mcp add sentry -- env SENTRY_AUTH_TOKEN=SUA_CHAVE_DE_API SENTRY_ORG=SEU_ORG SENTRY_PROJECT=SEU_PROJETO npx -y @sentry/mcp-server

Acessa issues, eventos e insights de performance do Sentry direto pelo agente.

```
#!/usr/bin/env bash

# Script de integração Gemini MCP para Archon AI

# Preencha os placeholders antes de rodar 🛠️

# Context7 MCP — documentação sempre atualizada via Context7

gemini mcp add context7 -- npx -y @upstash/context7-mcp

# Controle de Versão e Análise de Código

gemini mcp add git -- uvx mcp-server-git --repository /caminho/para/seu/repo

gemini mcp add github -- env GITHUB_PERSONAL_ACCESS_TOKEN=SEU_TOKEN_DE_ACESSO
npx -y @modelcontextprotocol/server-github

gemini mcp add gitlab -- env GITLAB_PERSONAL_ACCESS_TOKEN=SEU_TOKEN_DE_ACESSO
npx -y @modelcontextprotocol/server-gitlab


# Cloud e Infraestrutura (IaaS/PaaS)

gemini mcp add firebase -- npx -y firebase-tools@latest experimental:mcp

gemini mcp add kubernetes -- env KUBECONFIG=/caminho/para/seu/kubeconfig npx -y
mcp-server-kubernetes

gemini mcp add docker -- uvx docker-mcp

gemini mcp add aws-kb -- env AWS_ACCESS_KEY_ID=SEU_ACCESS_KEY
AWS_SECRET_ACCESS_KEY=SEU_SECRET_KEY npx -y
@modelcontextprotocol/server-aws-kb-retrieval


# Bancos de Dados e APIs

gemini mcp add postgres -- npx -y @modelcontextprotocol/server-postgres
postgresql://USUARIO:SENHA@HOST:5432/SEU_BANCO

gemini mcp add redis -- npx -y @modelcontextprotocol/server-redis
redis://USUARIO:SENHA@HOST:6379/0

gemini mcp add clickhouse -- npx -y clickhouse-mcp --dsn
```

"clickhouse://USUARIO:SENHA@HOST:8443/SEU_DB"

Monitoramento e Observabilidade

gemini mcp add sentry -- env SENTRY_AUTH_TOKEN=SUA_CHAVE_DE_API SENTRY_ORG=SEU_ORG SENTRY_PROJECT=SEU_PROJETO npx -y @sentry/mcp-server

gemini mcp add grafana -- env GRAFANA_URL=https://SEU_DOMINIO GRAFANA_API_KEY=SUA_CHAVE_DE_API npx -y @grafana/mcp-grafana

gemini mcp add datadog -- env DATADOG_API_KEY=SUA_CHAVE_DE_API DATADOG_SITE=SEU_SITE npx -y @datadog/mcp-server

Produtividade e Documentação

gemini mcp add notion -- env NOTION_API_KEY=SUA_CHAVE_DE_API npx -y @modelcontextprotocol/server-notion

gemini mcp add confluence -- env ATLASSIAN_API_TOKEN=SEU_TOKEN_DE_ACESSO npx -y @modelcontextprotocol/server-confluence

gemini mcp add slack -- env SLACK_BOT_TOKEN=SEU_TOKEN_DE_ACESSO npx -y @modelcontextprotocol/server-slack

Testes e Qualidade

gemini mcp add playwright -- npx -y mcp-server-playwright

gemini mcp add jest -- npx -y mcp-server-jest

gemini mcp add cypress -- npx -y mcp-server-cypress

Desenvolvimento Mobile e Cross-platform

gemini mcp add expo -- npx -y expo-cli@latest mcp

gemini mcp add flutter -- flutter mcp

gemini mcp add android-studio -- studio.sh mcp

Compatibilidade e Testes de Ambiente

gemini mcp add browserstack -- env BROWSERSTACK_USERNAME=SEU_USUARIO BROWSERSTACK_ACCESS_KEY=SUA_CHAVE_DE_API npx -y mcp-server-browserstack

gemini mcp add saucelabs -- env SAUCE_USERNAME=SEU_USUARIO SAUCE_ACCESS_KEY=SUA_CHAVE_DE_API npx -y mcp-server-saucelabs

echo "Todas as integrações foram adicionadas (ou tentadas). Preencha os placeholders e execute o script com atenção."

Como usar:

1. Salve esse script como setup_archon_mcp.sh.

2. Preencha todos os SEU_... com os valores corretos (tokens, URLs, creds).

3. Dê permissão de execução:

chmod +x setup_archon_mcp.sh

4. Execute no terminal:

./setup_archon_mcp.sh

Isso já montará um ambiente robusto com suporte a versionamento, infra, DBs, observabilidade, produtividade e testes — tudo pela interface MCP, integrado ao Archon AI.

Referências citadas

1. How to Build a Coding Assistant with Gemini CLI, MCP and Go - Google Codelabs, acessado em agosto 15, 2025, https://codelabs.developers.google.com/cloud-gemini-cli-mcp-go
2. Gemini CLI: A comprehensive guide to understanding, installing, and leveraging this new Local AI Agent : r/GeminiAI - Reddit, acessado em agosto 15, 2025, https://www.reddit.com/r/GeminiAI/comments/1lkojt8/gemini_cli_a_comprehensive_guide_to_understanding/
3. Gemini CLI: A comprehensive guide to understanding, installing, and leveraging

this new Local AI Agent - Reddit, acessado em agosto 15, 2025, https://www.reddit.com/r/GoogleGeminiAI/comments/1lkol0m/gemini_cli_a_comprehensive_guide_to_understanding/
4. Gemini CLI | Gemini for Google Cloud, acessado em agosto 15, 2025, https://cloud.google.com/gemini/docs/codeassist/gemini-cli
5. Gemini CLI Tutorial Series — Part 5 : Github MCP Server | by Romin Irani | Google Cloud, acessado em agosto 15, 2025, https://medium.com/google-cloud/gemini-cli-tutorial-series-part-5-github-mcp-server-b557ae449e6e
6. Gemini CLI Tutorial Series — Part 8: Building your own MCP Server | by Romin Irani | Google Cloud - Community | Aug, 2025 | Medium, acessado em agosto 15, 2025, https://medium.com/google-cloud/gemini-cli-tutorial-series-part-8-building-your-own-mcp-server-74d6add81cca
7. How to Build a Coding Assistant with Gemini CLI, MCP and Go - Google Codelabs, acessado em agosto 15, 2025, https://codelabs.developers.google.com/cloud-gemini-cli-mcp-go?hl=en
8. Gemini CLI: A Guide With Practical Examples - DataCamp, acessado em agosto 15, 2025, https://www.datacamp.com/tutorial/gemini-cli
9. Gemini CLI - Learn Netdata, acessado em agosto 15, 2025, https://learn.netdata.cloud/docs/ai-&-ml/devops-copilots/gemini-cli
10. Firebase MCP server | Firebase Documentation - Google, acessado em agosto 15, 2025, https://firebase.google.com/docs/cli/mcp-server
11. Claude Code | Learn Netdata, acessado em agosto 15, 2025, https://learn.netdata.cloud/docs/ai-&-ml/devops-copilots/claude-code
12. What MCPs is everyone using with Claude? : r/mcp - Reddit, acessado em agosto 15, 2025, https://www.reddit.com/r/mcp/comments/1lzfa9r/what_mcps_is_everyone_using_with_claude/
13. Awesome MCP Servers, acessado em agosto 15, 2025, https://mcpservers.org/
14. FR: Support command for configuring MCP servers (gemini mcp add) #3470 - GitHub, acessado em agosto 15, 2025, https://github.com/google-gemini/gemini-cli/issues/3470
15. Gemini CLI & Bright Data's Web MCP: Step-by-Step Guide, acessado em agosto 15, 2025, https://brightdata.com/blog/ai/gemini-cli-web-mcp
16. Give Gemini CLI the Ability to Generate Images and Video, Work with GitHub Repos, and Use Other Tools | by Dazbo (Darren Lester) | Google Cloud - Medium, acessado em agosto 15, 2025, https://medium.com/google-cloud/give-gemini-cli-the-ability-to-generate-images-and-video-work-with-github-repos-and-use-other-482172571f99
17. Gemini CLI Tutorial Series — Part 6 : More MCP Servers | by Romin Irani | Google Cloud, acessado em agosto 15, 2025, https://medium.com/google-cloud/gemini-cli-tutorial-series-part-6-more-mcp-servers-91422cadaa35
18. How to setup MCP SERVERS in GEMINI CLI | Complete Guide - YouTube, acessado em agosto 15, 2025,

https://m.youtube.com/watch?v=ODnjoFcdcwU&pp=0gcJCcEJAYcqIYzv
19. Google Gemini CLI Cheatsheet - Philschmid, acessado em agosto 15, 2025, https://www.philschmid.de/gemini-cli-cheatsheet
20. Gemini CLI Tutorial Series — Part 2 : Gemini CLI Command line parameters | by Romin Irani | Google Cloud - Medium, acessado em agosto 15, 2025, https://medium.com/google-cloud/gemini-cli-tutorial-series-part-2-gemini-cli-command-line-parameters-e64e21b157be
21. Use agentic chat as a pair programmer | Gemini Code Assist - Google for Developers, acessado em agosto 15, 2025, https://developers.google.com/gemini-code-assist/docs/use-agentic-chat-pair-programmer
22. Gemini CLI MCP Server - LobeHub, acessado em agosto 15, 2025, https://lobehub.com/mcp/diversioteam-gemini-cli-mcp
23. If an MCP doesn't supply instructions, Gemini CLI refuses to connect to the MCP. #2015, acessado em agosto 15, 2025, https://github.com/google-gemini/gemini-cli/issues/2015
24. Issue with MCP Server get_image and Google Gemini CLI - Figma Forum, acessado em agosto 15, 2025, https://forum.figma.com/report-a-problem-6/issue-with-mcp-server-get-image-and-google-gemini-cli-42183
25. Use MCP servers in VS Code, acessado em agosto 15, 2025, https://code.visualstudio.com/docs/copilot/chat/mcp-servers