Apollo MCP Server

Apollo MCP Server Quickstart

Create and run an MCP server in minutes with Apollo

Ask AI a question about this page

Ask with ChatGPT

Apollo MCP Server is a Model Context Protocol server that exposes your GraphQL API operations as MCP tools.

This guide walks you through the process of creating, running and configuring an MCP server with Apollo.

Prerequisites

- Rover CLI v0.36 or later. We'll use Rover to initialize a project and run the MCP server. Follow the instructions for installing and authenticating Rover with a GraphOS account.
- Node.js v18 or later (for `mcp-remote`)
- Claude Desktop or another MCP-compatible client

Step 1: Create an MCP server

Run the interactive initialization command:

Bash

```bash
rover init --mcp
```

The CLI wizard guides you through several prompts.

Select **Create MCP tools from a new Apollo GraphOS project** and **Apollo graph with Connectors (connect to REST services)** as your starting point.

You'll also need to select your organization and give your project a name and ID.

The wizard shows all files that will be created, including:

- MCP server configuration files
- GraphQL schema and operations
- Docker setup for (optional) deployment

Type `Y` to confirm and create your project files.

Step 2: Run your MCP Server

You can start your MCP server locally with `rover dev`.

1. Choose the environment-specific command to load environment variables from the provided `.env` file and start the MCP server.

   Linux / MacOS Windows Powershell

   terminal

   ```
   set -a && source .env && set +a && rover dev --supergraph-config su
   ```

2. You should see some output indicating that the GraphQL server is running at `http://localhost:4000` and the MCP server is running at

Latest Announcements from GraphQL Summit

Search Apollo content (Cmd+K or /)

terminal

```
npx @modelcontextprotocol/inspector http://127.0.0.1:8000/mcp --tra
```

4. This will automatically open your browser to `http://127.0.0.1:6274`.

5. Click **Connect**, then **List Tools** to see the available tools.

Step 3: Connect to an MCP client

Apollo MCP Server works with any MCP-compatible client. Choose your favorite client and follow the instructions to connect.

Claude Desktop (recommended)

Open the `claude_desktop_config.json` file in one of the following paths:

- Mac OS: `~/Library/Application\ Support/Claude/claude_desktop_config.json`
- Windows: `%APPDATA%\Claude\claude_desktop_config.json`
- Linux: `~/.config/Claude/claude_desktop_config.json`

Copy the configuration:

JSON

```json
{
  "mcpServers": {
    "mcp-My API": {
      "command": "npx",
      "args": [
        "mcp-remote",
        "http://127.0.0.1:8000/mcp"
      ]
    }
  }
}
```

Claude Code

Cursor

Feedback Edit on GitHub Ask Community

> Cline (VS Code Extension)

> OpenCode

> Windsurf

1. Restart your MCP client.
2. Test the connection by asking: "What MCP tools do you have available?".
3. Verify GraphQL operations are listed as available tools.
4. Test a query using one of your configured operations.

Step 4: Define MCP tools

MCP tools are defined as GraphQL operations. The project template currently uses operation collections as the source of its tools.

> **NOTE**
>
> See Define MCP Tools for other ways to define MCP tools.

1. Navigate to Sandbox at http://localhost:4000.
2. Click the Bookmark icon to open Operation Collections.
3. Click **Sandbox** beside "Showing saved operations for your Sandbox, across all endpoints" and select your graph. This represents the graph name and ID you used when creating your project.
4. You'll see an operation collection called "Default MCP Tools".
5. Create a new operation in the middle panel:

GraphQL
```graphql
# Retrieves product information
query GetProducts {
  products {
    id
    name
```

```
      description
    }
  }
```

6. Click the **Save** button and give it the name `GetProducts`.
7. Select the `Default MCP Tools` collection and click **Save**.
8. Restart your MCP client and test the connection by asking: "What MCP tools do you have available?". You should see the `GetProducts` tool listed. You can also test this with MCP Inspector.

Step 5: Deploy your MCP server

Apollo MCP Server can run in any container environment.

Using the Apollo Runtime Container

Your project includes a pre-configured `mcp.Dockerfile` for easy deployment. This container includes:

- Apollo Router for serving your GraphQL API
- Apollo MCP Server for MCP protocol support
- All necessary dependencies

1. Build the container:

Bash
```bash
docker build -f mcp.Dockerfile -t my-mcp-server .
```

2. Run locally:

Bash
```bash
docker run -p 4000:4000 -p 8000:8000 \
  -e APOLLO_KEY=$APOLLO_KEY \
  -e APOLLO_GRAPH_REF=$APOLLO_GRAPH_REF \
  -e MCP_ENABLE=1 \
my-mcp-server
```

3. Deploy to your platform. The container can be deployed to any platform supporting Docker, such as: AWS ECS/Fargate, Google Cloud Run, Azure Container Instances, Kubernetes, Fly.io, Railway, Render.

4. Ensure these variables are set in your deployment environment:

Variable	Description	Required
`APOLLO_KEY`	Your graph's API key	Yes
`APOLLO_GRAPH_REF`	Your graph reference	Yes
`APOLLO_MCP_TRANSPORT__PORT`	MCP server port (default: 8000)	No
`APOLLO_ROUTER_PORT`	Router port (default: 4000)	No

For more deployment options, see the Deploy the MCP Server page.

Update client configuration

After deploying, update your MCP client configuration to use the deployed URL:

{} JSON

```json
{
  "mcpServers": {
    "my-api": {
      "command": "npx",
      "args": [
        "mcp-remote",
        "https://your-deployed-server.com/mcp"
      ]
    }
  }
}
```

Troubleshooting

Client doesn't see tools:

- Ensure you restarted your MCP client after configuration
- Verify the Apollo MCP Server is running (`rover dev` command)
- Check port numbers match between server and client config

Connection refused errors:

- Confirm the server is running on the correct port
- Verify firewall settings allow connections to localhost:8000
- For remote connections, ensure the host is set to `0.0.0.0` in your config

Authentication issues:

- Verify environment variables are properly set
- Check that your GraphQL endpoint accepts the provided headers
- When using `rover dev` you can test your GraphQL endpoint using Sandbox at http://localhost:4000

Additional resources

- Tutorial: Getting started with MCP and GraphQL
- Tutorial: Agentic GraphQL: MCP for the Enterprise
- Blog: Getting Started with Apollo MCP Server

Getting help

If you're still having issues:

- Check Apollo MCP Server GitHub issues
- Join the Apollo community forums
- Contact your Apollo representative for direct support

PREVIOUS NEXT

Apollo MCP Server

Define MCP Tools