# Connecting Tools to AI

MGMT 675: Generative AI for Finance

Kerry Back

# The Problem: Isolated AI

Claude Desktop is powerful—but out of the box, it can only read and write text. It cannot control your browser, run terminal commands, or interact with other software.

- Want Claude to fill out a web form? It can't reach your browser.
- Want Claude to run a shell command? It has no terminal access.
- Want Claude to query a database? It has no connection.
- **MCP servers solve this problem.**

# What is MCP?

The **Model Context Protocol (MCP)** is an open standard that lets AI applications connect to external tools and data sources through a uniform interface.

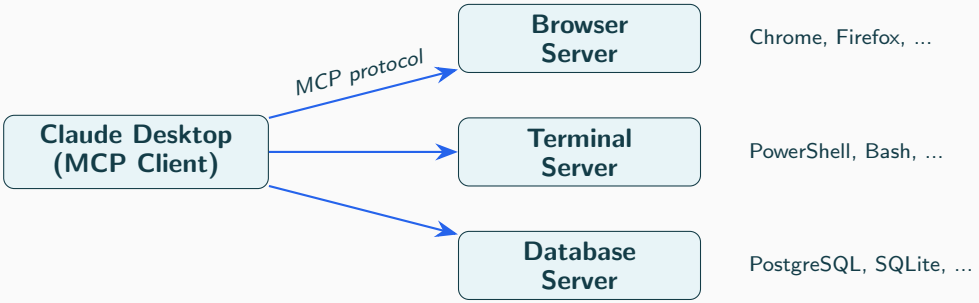
## The Idea

- A standard "plug-in" protocol
- AI app connects to MCP servers
- Each server provides specific tools
- Tools appear automatically in the AI

## Analogy

- USB is a standard for peripherals
- Plug in a keyboard, it just works
- MCP is a standard for AI tools
- Connect a server, tools just appear

# How MCP Works




- Claude Desktop is the **client**; each external service runs as a **server**
- Servers expose **tools** (functions the AI can call) via a standard protocol
- You can connect multiple servers at once

# Connecting an MCP Server in Claude Desktop

# Two Ways to Install MCP Servers

## Desktop Extensions (Easy)

1. Open Claude Desktop
2. Go to **Settings** → **Extensions**
3. Click **Browse extensions**
4. Find the server you want
5. Click **Install**

One-click install. Anthropic-reviewed.

## Manual Configuration

1. Install prerequisites (next slide)
2. Edit a JSON config file
3. Specify the server command
4. Restart Claude Desktop
5. Tools appear automatically

More control. Works with any server.

# Prerequisites: Node.js and uv

MCP servers are distributed as Node.js or Python packages. You need one or both installed depending on which servers you use.

## Node.js (for `npx` servers)

- Download the **LTS** installer from `https://nodejs.org`
- Run the installer (includes `npm` and `npx`)
- Verify: open a terminal and type `node --version`

Used by: DesktopCommander, FMP

## uv (for `uvx` servers)

- **Mac**: run in Terminal:
  `brew install uv`
- **Windows**: run in PowerShell:
  `winget install astral-sh.uv`
- Verify: `uvx --version`

Used by: Browser-Use, Alpha Vantage

**Install these before configuring any MCP server.** `uv` also installs Python if you don't have it.

# Config File Location

For manual installation, edit the Claude Desktop configuration file:

**macOS**

```
~/Library/Application Support/
Claude/claude_desktop_config.json
```

**Windows**

```
%APPDATA%\Claude\
claude_desktop_config.json
```

- Create the file if it doesn't exist
- The file defines which MCP servers to launch when Claude Desktop starts
- After editing, **fully quit and restart** Claude Desktop

# Config File Structure

## Example: Adding Two MCP Servers

```
{
  "mcpServers": {
    "my-server-1": {
      "command": "npx",
      "args": ["-y", "some-mcp-package"]
    },
    "my-server-2": {
      "command": "uvx",
      "args": ["another-mcp-package"]
    }
  }
}
```

- Each server has a name, a command, and arguments
- `npx` runs Node.js packages; `uvx` runs Python packages

# Verifying the Connection

1. Restart Claude Desktop after editing the config file
2. Look for the **hammer icon** in the chat input area
3. Click it to see the list of available tools from your servers
4. If a server fails to start, check the logs:
   - macOS: `~/Library/Logs/Claude/mcp*.log`
   - Windows: `%APPDATA%\Claude\logs\mcp*.log`

When Claude wants to use a tool, it will ask for your **permission** before executing. You stay in control.

# Browser-Use: AI-Driven Browser Automation

# What is Browser-Use?

**Browser-Use** is an MCP server that lets Claude autonomously control a web browser. You describe a task in plain English, and it handles the navigation, clicking, and typing.

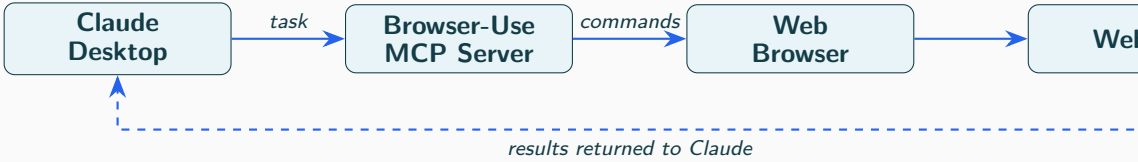
## Example Tasks

- "Fill out the application form on this page with my information"
- "Download all linked CSV files from this page"
- "Search for AAPL on Yahoo Finance and get the P/E ratio"

## Key Features

- Plain-language task descriptions
- Navigates pages autonomously
- Fills forms, clicks buttons
- Extracts data from web pages
- Works on Mac and Windows

# How Browser-Use Works




- Claude sends a high-level task (e.g., "fill out this form")
- Browser-Use translates the task into browser actions (click, type, scroll)
- Results (extracted text, confirmation, screenshots) return to Claude

# Browser-Use: Two Deployment Options

## Cloud (Hosted)

- No local setup required
- Browser runs on their servers
- **10-step limit per task**
- Requires a Browser-Use API key
- Persistent login profiles available

## Local (Self-Hosted)

- Browser runs on your machine
- No step limits
- Requires your own LLM API key
- Full privacy—data stays local
- **Recommended for most users**

Docs: https://docs.browser-use.com/customize/integrations/mcp-server

# Installing Browser-Use

## Add to Claude Desktop Config

```
"browser-use": {
  "command": "uvx",
  "args": ["--from", "browser-use[cli]",
           "browser-use", "--mcp"],
  "env": {
    "ANTHROPIC_API_KEY": "sk-ant-..."
  }
}
```

- **Prerequisite**: `uv` installed and Chrome/Chromium on your machine
- The `[cli]` extra is required—without it, the MCP server won't start
- `ANTHROPIC_API_KEY`: Browser-Use calls an LLM to decide how to navigate pages, so it needs its own API key (Anthropic or OpenAI)
- Restart Claude Desktop after editing the config; a browser window will open when Claude uses the tool

# Browser-Use: Strengths and Limitations

## Strengths

- Natural language task descriptions
- No need to write selectors or scripts
- Form filling is a primary use case
- Handles multi-step workflows
- Cross-platform (Mac, Windows, Linux)

## Limitations

- AI-driven: can be unpredictable on complex pages
- Local version needs an LLM API key (adds cost per action)
- Newer project, smaller community
- May struggle with heavily dynamic sites

# DesktopCommanderMCP: Terminal Control

# What is DesktopCommanderMCP?

**DesktopCommanderMCP** is an MCP server that gives Claude Desktop the ability to execute terminal commands, manage files, and control processes on your computer.

## Example Tasks

- "Run `pip install pandas`"
- "Show me all Python files in this folder"
- "Start a Jupyter notebook server"
- "Check if port 8080 is in use"

## Key Features

- Execute terminal commands
- Read, write, and search files
- Manage running processes
- Diff-based file editing
- Works on Mac, Windows, and Linux

# Why Give Claude a Terminal?

- **Software installation**: "Install the Python libraries I need for this project"
- **Data processing**: "Convert all the .xlsx files in this folder to .csv"
- **Git operations**: "Initialize a git repo, commit my changes, and push"
- **System administration**: "Show disk usage" or "List running processes"
- **Development**: "Run my test suite and tell me what failed"

With DesktopCommander, Claude Desktop can do much more than chat—it becomes an active tool on your computer.

# Installing DesktopCommanderMCP

## Add to Claude Desktop Config

```
"desktop-commander": {
  "command": "npx",
  "args": ["-y", "desktop-commander-mcp"]
}
```

- **Prerequisite**: Node.js (LTS) installed
- `npx -y` downloads and runs the package automatically—no separate install step
- No API keys needed—everything runs locally
- Restart Claude Desktop after editing the config
- Claude will ask permission before running each command

GitHub: https://github.com/wonderwhy-er/DesktopCommanderMCP

# DesktopCommanderMCP: Tools Available

| Tool | Description |
|---|---|
| execute_command | Run a terminal command |
| read_file | Read file contents |
| write_file | Create or overwrite a file |
| edit_block | Make targeted edits using diff blocks |
| search_files | Search file contents with regex |
| list_directory | List files and folders |
| get_file_info | Get file metadata (size, dates) |
| list_processes | Show running processes |
| kill_process | Stop a running process |

# DesktopCommanderMCP: Strengths and Limitations

## Strengths

- Full terminal access
- File operations built in
- Process management
- Well-established, large community
- Cross-platform (Mac, Windows, Linux)
- No API keys needed (runs locally)

## Limitations

- Powerful = potentially dangerous
- A bad command could delete files or break your system
- Always review commands before approving
- No built-in sandboxing

**Always review terminal commands before clicking "Allow"—Claude asks permission for a reason.**

# Financial Data Servers

# MCP Servers for Financial Data

## Alpha Vantage

- 115+ tools: stock prices, options, forex, crypto, economic indicators, 50+ technical indicators
- Free API key at https://www.alphavantage.co
- **Free tier**: 25 requests/day, 5/minute, last 100 data points per request
- Paid: from $49.99/month

## Financial Modeling Prep

- 253+ tools: financial statements, fundamentals, SEC filings, earnings, ESG scores, insider trading
- Free API key at https://financialmodelingprep.com
- **Free tier**: 250 requests/day, 5 years of annual data, 500 MB/month
- Some endpoints are paid-only

# Installing Financial Data Servers

## Alpha Vantage

```
"alphavantage": {
  "command": "uvx",
  "args": ["av-mcp", "YOUR_API_KEY"]
}
```

**Prerequisite**: `uv` installed. Get a free API key at https://www.alphavantage.co/support/#api-key

## Financial Modeling Prep

```
"financial-modeling-prep": {
  "command": "npx",
  "args": ["-y", "@houtini/fmp-mcp"],
  "env": { "FMP_API_KEY": "YOUR_API_KEY" }
}
```

# Putting It All Together

# Combining Multiple MCP Servers

You can connect multiple MCP servers at once. Claude sees all their tools and can use them together in a single conversation.

## Example Combination

- **Browser-Use**: navigate to a website and download data
- **DesktopCommander**: run a Python script to process the data
- **Result**: end-to-end automation with no manual steps

## Other Popular Servers

- **Filesystem**: sandboxed file access
- **PostgreSQL/SQLite**: database queries
- **GitHub**: pull requests, issues
- **Slack**: send messages
- Browse all: `https://github.com/modelcontextprotocol/servers`

# Example: Multiple Servers in One Config

**All entries go inside `"mcpServers": { ... }`**

```
{
  "mcpServers": {
    "browser-use": {
      "command": "uvx",
      "args": ["--from", "browser-use[cli]",
               "browser-use", "--mcp"],
      "env": {"ANTHROPIC_API_KEY": "sk-ant-..."}
    },
    "desktop-commander": {
      "command": "npx",
      "args": ["-y", "desktop-commander-mcp"]
    },
    "alphavantage": {
      "command": "uvx",
      "args": ["av-mcp", "YOUR_AV_KEY"]
    }
```

# Troubleshooting MCP Servers

MCP servers can be finicky to set up. If tools don't appear after restarting Claude Desktop, work through this checklist:

1. **Validate your JSON**: a missing comma or extra comma will silently break the config. Paste your config into `https://jsonlint.com` to check.
2. **Check prerequisites**: run `node --version` and `uvx --version` in a terminal. If either is "not found," install it first.
3. **Fully quit Claude Desktop**: on Mac, `Cmd+Q`; on Windows, right-click the system tray icon and **Quit**. Simply closing the window is not enough.
4. **Look for the hammer icon**: after restarting, click the hammer icon in the chat box to see if tools loaded.
5. **Check the logs**:
   - Mac: `~/Library/Logs/Claude/mcp*.log`
   - Windows: `%APPDATA%\Claude\logs\mcp*.log`
6. **Test the command manually**: copy the `command` and `args` from your config and run them in a terminal to see if the server starts.

# Looking Ahead: Built-In Alternatives to MCP

DesktopCommander and Browser-Use add terminal and browser control to Claude Desktop's **Chat** mode. But Claude Desktop also has other modes—available on paid plans—that have these capabilities built in. We will cover them later in the course.

| | Chat | Code tab | Cowork tab |
|---|---|---|---|
| Terminal / files | **MCP needed** | Built in | Built in |
| Browser control | **MCP needed** | Chrome ext | Chrome ext |
| Available on | All plans | Pro+ | Pro+ |

- **Code tab**: a coding environment built into Claude Desktop with terminal and file access
- **Cowork tab**: an agent that works with your local files in a sandboxed environment
- **Claude in Chrome**: a browser extension—built-in alternative to Browser-Use

# Summary

- **MCP** is a standard protocol that connects AI apps to external tools
- **MCP servers** are plug-ins—each one adds specific capabilities
- Install via **Desktop Extensions** (easy) or **config file** (flexible)
- **Browser-Use**: autonomous web browsing—forms, downloads, data extraction
- **DesktopCommanderMCP**: terminal access—commands, files, processes
- **Alpha Vantage** and **Financial Modeling Prep**: financial data with free tiers
- Combine servers for end-to-end workflows

**MCP turns Claude from a conversation partner into a tool that takes action on your computer.**