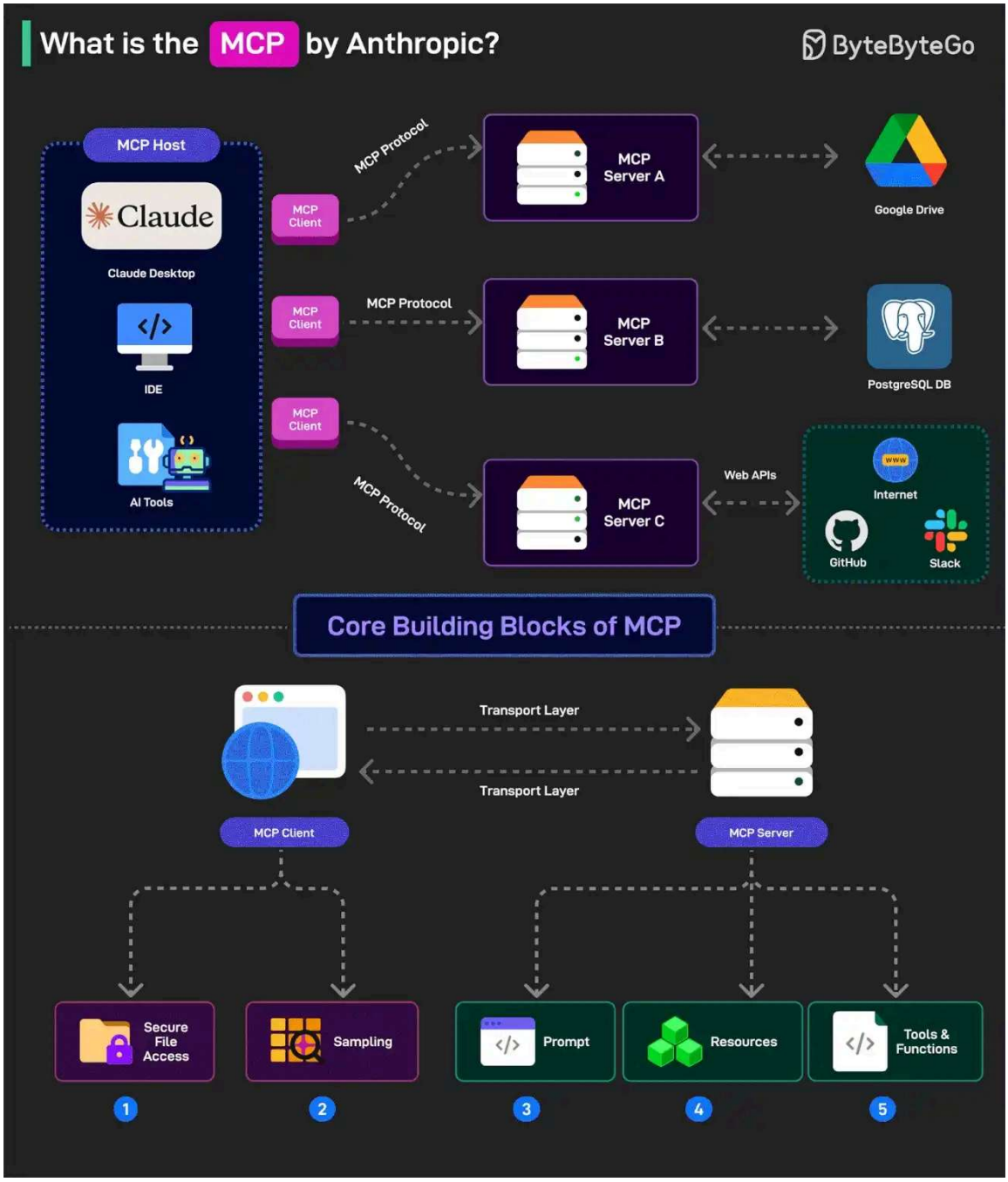


What is the Model Context Protocol (MCP)?

MCP (Model Context Protocol) is an open-source standard for connecting AI applications to external systems.

- Using MCP, AI applications like Claude or ChatGPT can connect to data sources (e.g. local files, databases), tools (e.g. search engines, calculators) and workflows (e.g. specialized prompts)—enabling them to access key information and perform tasks.
- Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect electronic devices, MCP provides a standardized way to

connect AI applications to external systems.

What can MCP enable?

- Agents can access our Google Calendar and Notion, acting as a more personalized AI assistant.
- Claude Code can generate an entire web app using a Figma design.
- Enterprise chatbots can connect to multiple databases across an organization, empowering users to analyze data using chat.
- AI models can create 3D designs on Blender and print them out using a 3D printer.

Why does MCP matter?

Depending on where we sit in the ecosystem, MCP can have a range of benefits.

- Developers: MCP reduces development time and complexity when building, or integrating with, an AI application or agent.
- AI applications or agents: MCP provides access to an ecosystem of data sources, tools and apps which will enhance capabilities and improve the end-user experience.
- End-users: MCP results in more capable AI applications or agents which can access your data and take actions on your behalf when necessary.

🧩 Key Components of MCP Architecture

Here are the main building blocks in a typical MCP-based agentic AI system:

Component	Role / Responsibility
MCP Client	The part of the system (often the agent or agent host) that requests contextual information or actions. It formulates queries to MCP servers, decides what tools/data it needs, sends standardized requests, and receives responses.
MCP Server(s)	Servers that provide capabilities — e.g. APIs, data sources, memory stores, tool wrappers. They receive MCP-requests, perform the needed operations (authentication, data retrieval, tool invocation), then return results in a standardized format.
MCP Host / Orchestrator	The runtime or environment that ties together clients and servers, orchestrates workflows, manages sessions, state, possibly multiple agents. It ensures context flows, agents collaborate, tool discovery, and policy enforcement.
Protocol / Transport Layer	Defines how messages are passed between clients and servers — for example, JSON-RPC (commonly mentioned), streaming, or server-sent events for live updates. Handles framing, request/response linking, and stateful sessions.

Component	Role / Responsibility
Memory / State / Context Store	Shared or agent-specific memory (short-term or long-term), storing context, intermediate results, history, and environment config. The agent can reuse past information; context retention across steps.

🖥️ MCP Server

An MCP Server is a service or component that provides specific functions, data, or tools to the agentic AI system through the MCP protocol.

It receives standardized requests from the MCP Client (the agent), performs the necessary operations (like fetching data or running an action), and returns structured responses back to the client.

⚙️ Core Server Features

Servers provide functionality through **three building blocks**:

Feature	Explanation	Examples	Who Controls It
Tools	Functions that your LLM can actively call, and decide when to use them based on user requests. Tools can write to databases, call external APIs, modify files, or trigger other logic.	• Search flights • Send messages • Create calendar events	**Model**
Resources	Passive data sources that provide read-only access to information for context, such as file contents, database schemas, or API documentation.	• Retrieve documents • Access knowledge bases • Read calendars	**Application**
Prompts	Pre-built instruction templates that tell the model to work with specific tools and resources.	• Plan a vacation • Summarize my meetings • Draft an email	**User**

🖥️ MCP Client

The MCP Client is the component or agent in the MCP (Model Context Protocol) architecture that interacts with MCP Servers. Its main role is to request information,

invoke tools, and manage context for the AI agent. Think of it as the "brains" that decides what to ask and how to use the responses.

🧩 Core Client Features

In addition to using context provided by servers, **MCP Clients** may provide features that allow servers to build richer interactions:

Feature	Explanation	Example
Sampling	Allows servers to request LLM completions through the client, enabling an agentic workflow. The client maintains control of user permissions and security measures.	A server for booking travel may send a list of flights to an LLM and request that the LLM pick the best flight for the user.
Roots	Allows clients to specify which directories servers should focus on, communicating intended scope through a coordination mechanism.	A server for booking travel may be given access to a specific directory, from which it can read a user's calendar.
Elicitation	Enables servers to request specific information from users during interactions, providing a structured way to gather information on demand.	A server booking travel may ask for the user's preferences on airplane seats, room type, or contact number to finalize a booking.

How to connect to local MCP servers

Learn how to extend Claude Desktop with local MCP servers to enable file system access and other powerful integrations

Model Context Protocol (MCP) servers extend AI applications' capabilities by providing secure, controlled access to local resources and tools. Many clients support MCP, enabling diverse integration possibilities across different platforms and applications.

This guide demonstrates how to connect to local MCP servers using Claude Desktop as an example, one of the many clients that support MCP. While we focus on Claude Desktop's implementation, the concepts apply broadly to other MCP-compatible clients. By the end of this tutorial, Claude will be able to interact with files on your computer, create new documents, organize folders, and search through your file system—all with your explicit permission for each action.

Prerequisites

Before starting this tutorial, ensure you have the following installed on your system:

1. Claude Desktop

Download and install Claude Desktop for your operating system. Claude Desktop is currently available for macOS and Windows.

2. Node.js (node --version)

The Filesystem Server and many other MCP servers require Node.js to run. Verify your Node.js installation by opening a terminal or command prompt and running:

Understanding MCP Servers

MCP servers are programs that run on your computer and provide specific capabilities to Claude Desktop through a standardized protocol. Each server exposes tools that Claude can use to perform actions, with your approval. The Filesystem Server we'll install provides tools for:

- Reading file contents and directory structures
- Creating new files and directories
- Moving and renaming files
- Searching for files by name or content

All actions require your explicit approval before execution, ensuring you maintain full control over what Claude can access and modify.

Installing the Filesystem Server

The process involves configuring Claude Desktop to automatically start the Filesystem Server whenever you launch the application. This configuration is done through a JSON file that tells Claude Desktop which servers to run and how to connect to them.

1. **Open Claude Desktop Settings**

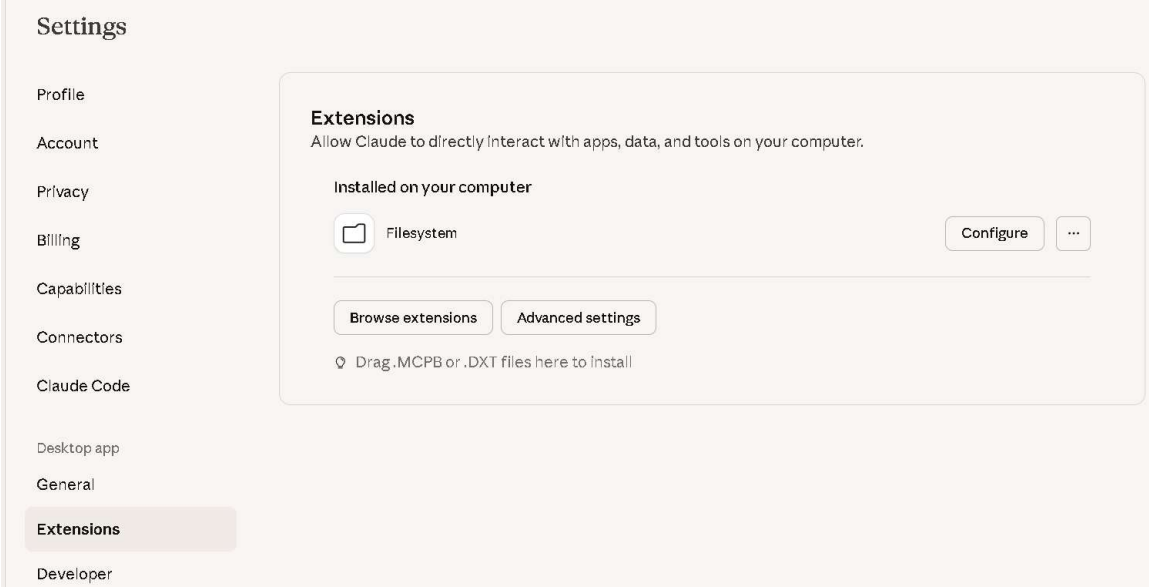
Start by accessing the Claude Desktop settings. Click on the Claude menu in your system's menu bar (not the settings within the Claude window itself) and select "Settings..."

This opens the Claude Desktop configuration window, which is separate from your Claude account settings.

2. **Access Developer Settings**

In the Settings window, navigate to the "Developer" tab in the left sidebar. This section contains options for configuring MCP servers and other developer features.

Click the "Edit Config" button to open the configuration file:



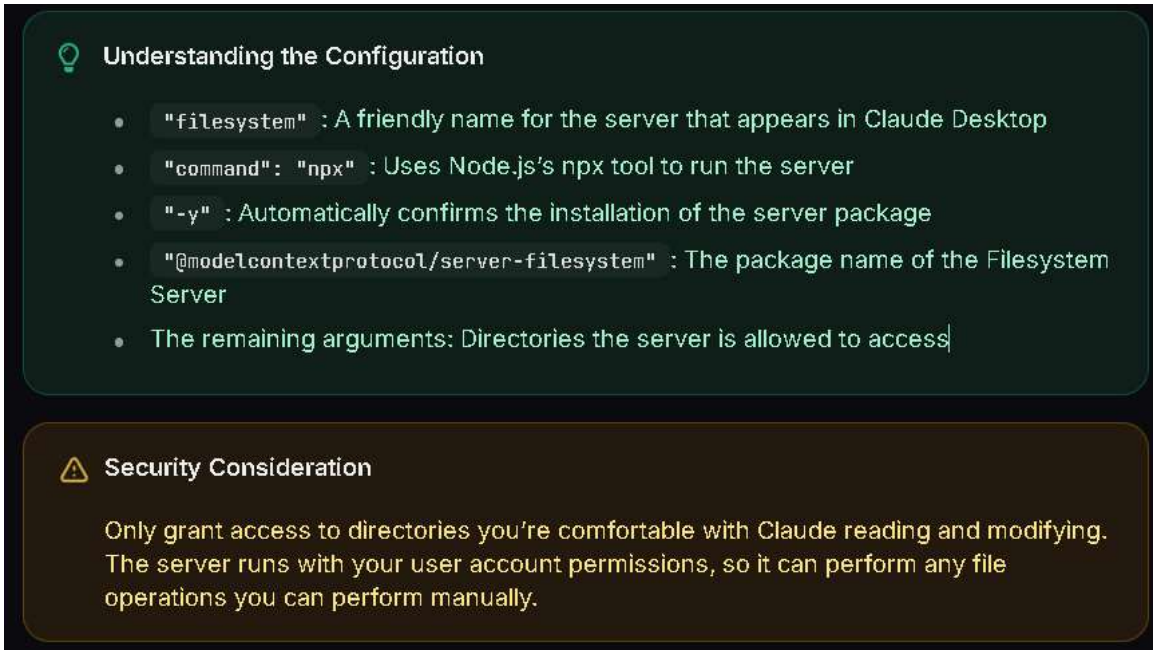
This action creates a new configuration file if one doesn't exist, or opens your existing configuration.

3. **Configure the Filesystem Server**

Replace the contents of the configuration file with the following JSON structure. This configuration tells Claude Desktop to start the Filesystem Server with access to specific directories:

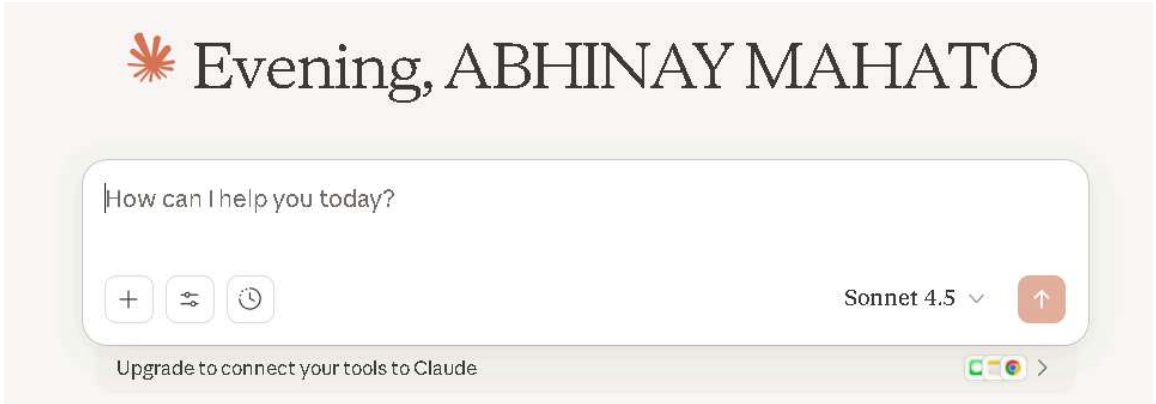
In [ ]:
```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-filesystem",
        "/Users/username/Desktop",
        "/Users/username/Downloads"
      ]
    }
  }
}
```

Replace username with your actual computer username. The paths listed in the args array specify which directories the Filesystem Server can access. You can modify these paths or add additional directories as needed.



4. Restart Claude Desktop

After saving the configuration file, completely quit Claude Desktop and restart it. The application needs to restart to load the new configuration and start the MCP server. Upon successful restart, you'll see an MCP server indicator in the bottom-right corner of the conversation input box:



Using the Filesystem Server

With the Filesystem Server connected, Claude can now interact with your file system. Try these example requests to explore the capabilities:

File Management Examples

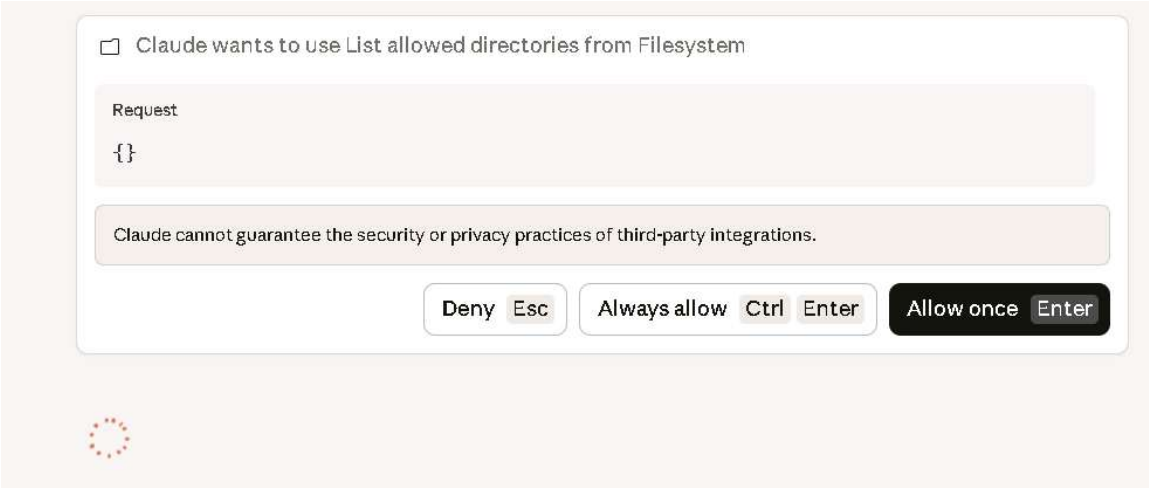
"Can you write a poem and save it to my desktop?" - Claude will compose a poem and create a new text file on your desktop

"What work-related files are in my downloads folder?" - Claude will scan your downloads and identify work-related documents

"Please organize all images on my desktop into a new folder called 'Images'" - Claude will create a folder and move image files into it

How Approval Works

Before executing any file system operation, Claude will request your approval. This ensures you maintain control over all actions:



Review each request carefully before approving. You can always deny a request if you're not comfortable with the proposed action.

Now that you've successfully connected Claude Desktop to a local MCP server.

In [ ]:

In [ ]: