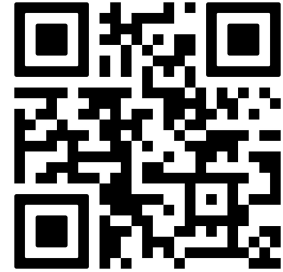
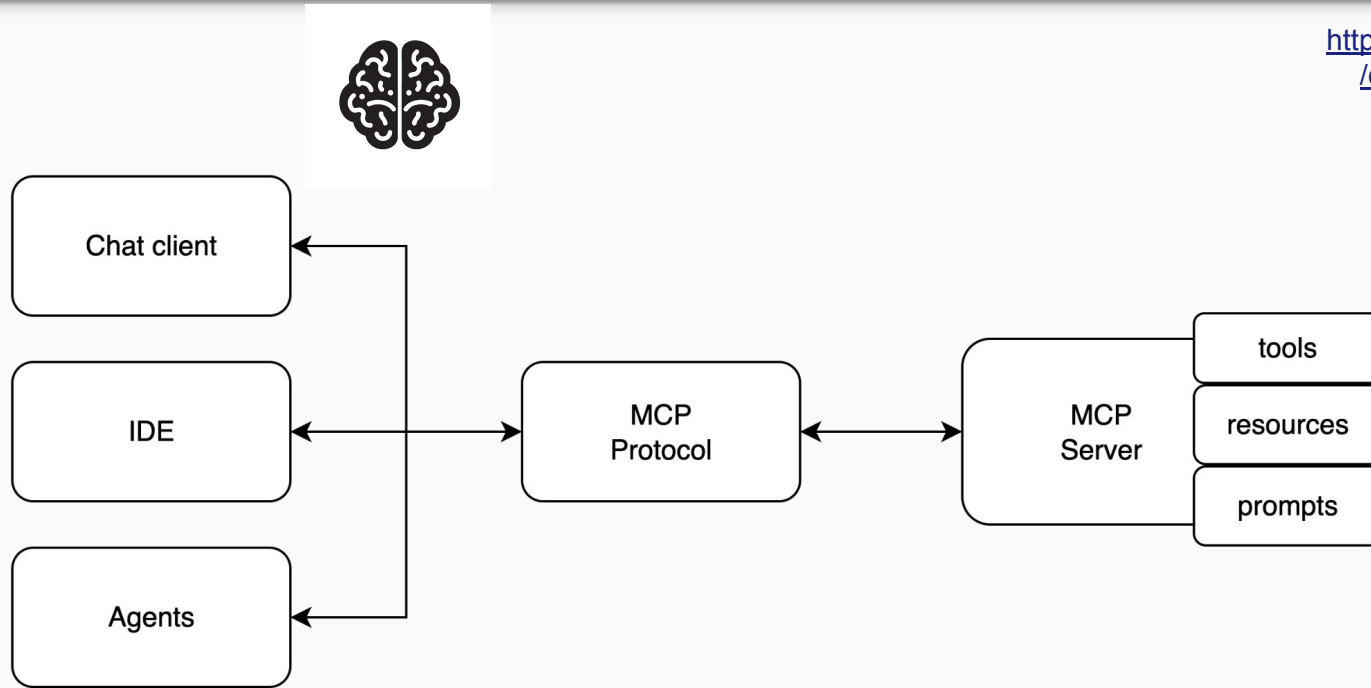


# Lessons learned: MCP server implementation



# What is MCP? (Model Context protocol)

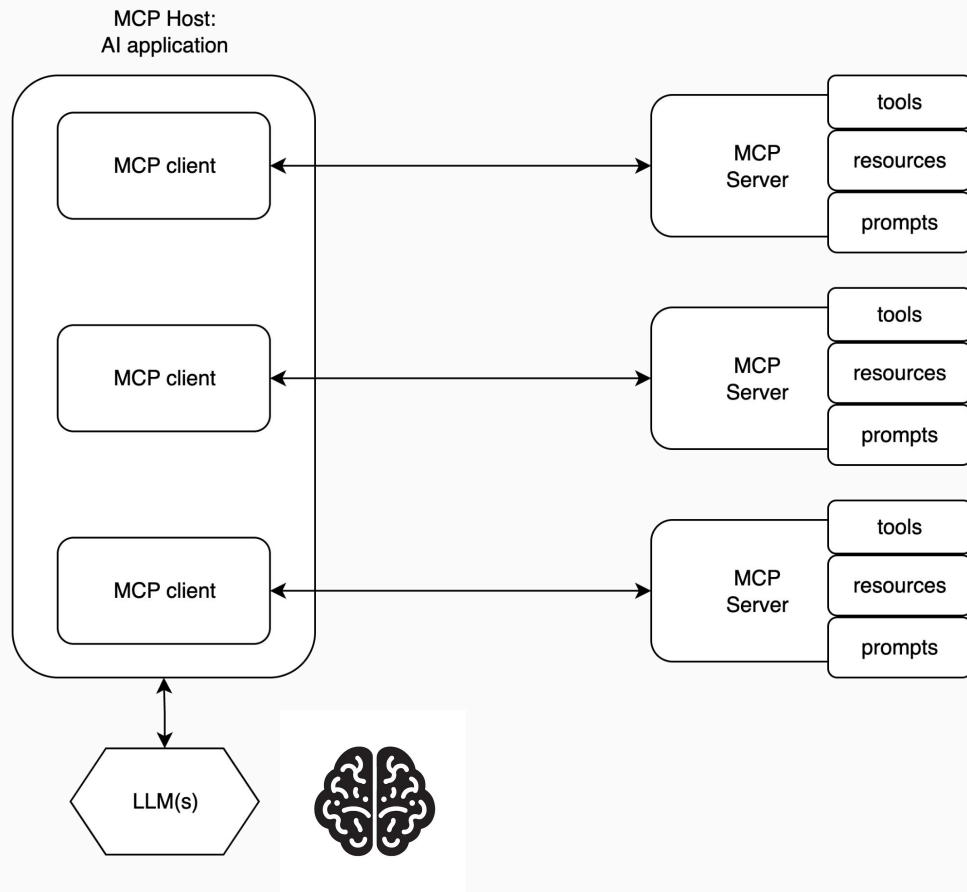
<https://modelcontextprotocol.io/docs/getting-started/intro>



MCP is an open source protocol for connecting AI applications to external systems

# MCP Protocol

- **Client-Server**
  - Transport Layer: stdio, streamable HTTP transport
  - Data Layer: JSON-RPC
  - OAuth
- **Discovery**
  - Client/Server capabilities
- **Tool/Resources/Prompts**
  - JSON schema + Description
  - Execution
  - Notification
- **SDKs**
  - Python, Go, Rust, Typescript, C#, Java, Kotlin, PHP, ...



# Example tool implementation

```
/// List all resource of a given group version kind using a given filter
#[tool(description = "List all resource of a given group, version, kind using a given filter.
    The meaning of the group, version and kind can be found through discovery")]
pub async fn list_resources(
    &self,
    Parameters(p: ListResourcesArgs): Parameters<ListResourcesArgs>,
) -> Result<Json<ListResourcesResult>, McpError> {

    tracing::info!("INVOKE MCP TOOL list resources {:?}", p);
    let uc: Arc<UnstructuredClient> = self.factory Arc<Factory>
        .get_unstructured_client();

    let gvk: GroupVersionKind = GroupVersionKind{
        group: p.gvk.group,
        version: p.gvk.version,
        kind: p.gvk.kind,
    };

    let mut ul: Unstructured = Unstructured::new(&gvk);

    uc.list(&mut ul, None).await.map_err(
        |e: ClientError| McpError::internal_error(format!("list failed {e}"), None)?;

    let items: Vec<serde_json::Value> = ul Unstructured
        .get_items()
        .iter()
        .map(|u: &Unstructured| u.to_json_value())
        .collect();

    Ok(Json(ListResourcesResult { resources: items }))
} fn list_resources
```

Description

Schema (JSON schema)  
input, output (optional)

Tool business logic

# Example client usage

```
{
  "mcpServers": {
    "c4o": {
      "url": "https://localhost:6443/mcp/sse",
      "headers": {
        "Authorization": "..."
      }
    }
  }
}
```

mcp.json

Cursor Settings X mcp.json

W wim.henderickx@... Free Plan

Search settings ⌘F

General

Chat

Tab

Models

Background Agents

MCP

Rules & Memories

Indexing & Docs

Network

Beta

Docs

MCP

Installed Servers

C c4o 3 tools enabled

+ New MCP Server Add a Custom MCP Server

Using the c4o tool

yes please export this to excel ?

I'll save the CSV to your Downloads folder so you can open it in Excel.

Ran command: printf

\$ printf 'name,tenant,category,class,mode,value,claimed,pool  
ipam.pool1.prefix-10\_0\_0\_0\_16,default,Prefix,IPAM,Static,10.0.0.0/16,10.0.0.0/16,ipam.pool1  
ipam.pool1.prefix-10\_0\_0\_0\_24,default,Prefix,IPAM,Static,10.0.0.1/24,10.0.0.1/24,ipam.pool1  
ipam.pool1.prefix-172\_0\_0\_0\_16,default,Prefix,IPAM,Static,172.0.0.0/16,172.0.0.0/16,ipam.pool1  
range1,default,Range,IPAM,Static,172.0.0.10-172.0.0.19,172.0.0.10-172.0.0.19,ipam.pool1  
' > "\$HOME/Downloads/ip\_claims.csv"

✓ Success

Saved. You can open this in Excel:

• ~/Downloads/ip\_claims.csv

Want me to include additional fields (e.g., creationTimestamp, uid) or filter differently?

Review Changes

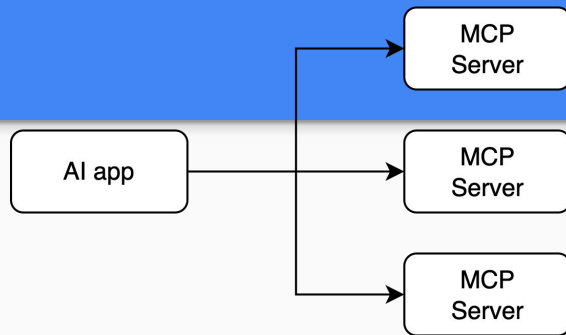
@ Add Context 11.5%

Write, @ for context, / for commands

ip\_claims

name	tenant	category	class	mode	value	claimed	pool
ipam.pool1.prefix-10_0_0_0_16	default	Prefix	IPAM	Static	10.0.0.0/16	10.0.0.0/16	ipam.pool1
ipam.pool1.prefix-10_0_0_0_24	default	Prefix	IPAM	Static	10.0.0.1/24	10.0.0.1/24	ipam.pool1
ipam.pool1.prefix-172_0_0_0_16	default	Prefix	IPAM	Static	172.0.0.0/16	172.0.0.0/16	ipam.pool1
range1	default	Range	IPAM	Static	172.0.0.10-172.0.0.19	172.0.0.10-172.0.0.19	ipam.pool1

# Why is it gaining so much traction?



Natural language

Natural way for humans to interact with a particular system

Open

A single AI application (chat, IDE, Agent) can interact with multiple MCP servers versus being contained to a given Server implementation

Abstraction

Avoids specific client implementation per server

## Lessons learned

- It's fairly easy to build a MCP server (see SDK list)
- Not all clients support all parts of the specification
- Session lifecycle is immature.
- Tool scale and tool information need to be handled properly
- Containing MCP context with the right info needs further work
- MCP server should be built on a proper API server if you want it to be exposed to multiple users
  - Authn/Authz - Identity, RBAC
  - Logging, Tracing, Stats, Rate limiting
- 



mcp-lessons-learned

THANK YOU