

Building MCP Servers in Go

by Yash Mehrotra

Who Am I

Yash Mehrotra

Backend & Platform @
Flanksource

Prev: BlinkIt, MindTickle

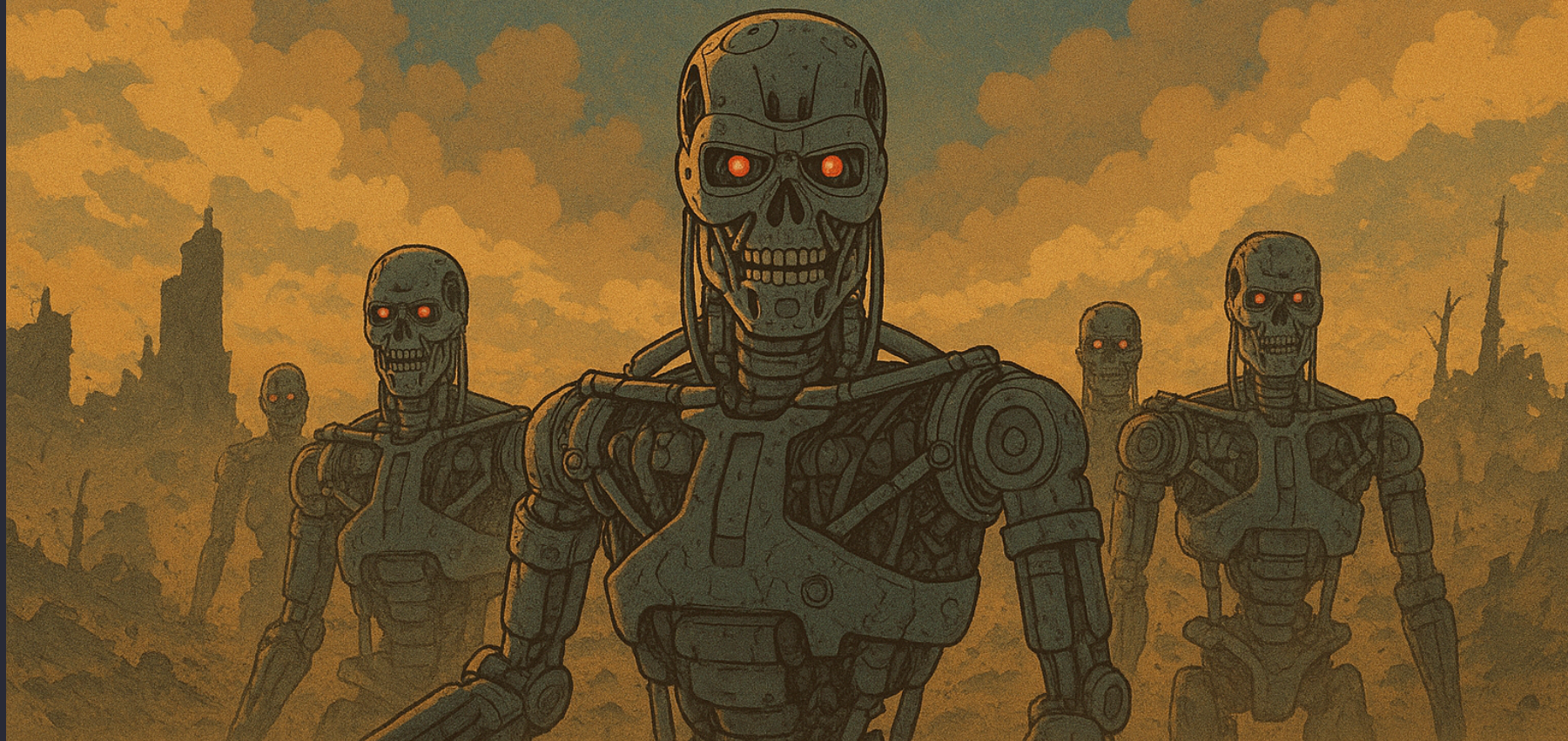
Enjoys  and 

yashmehrotra.com



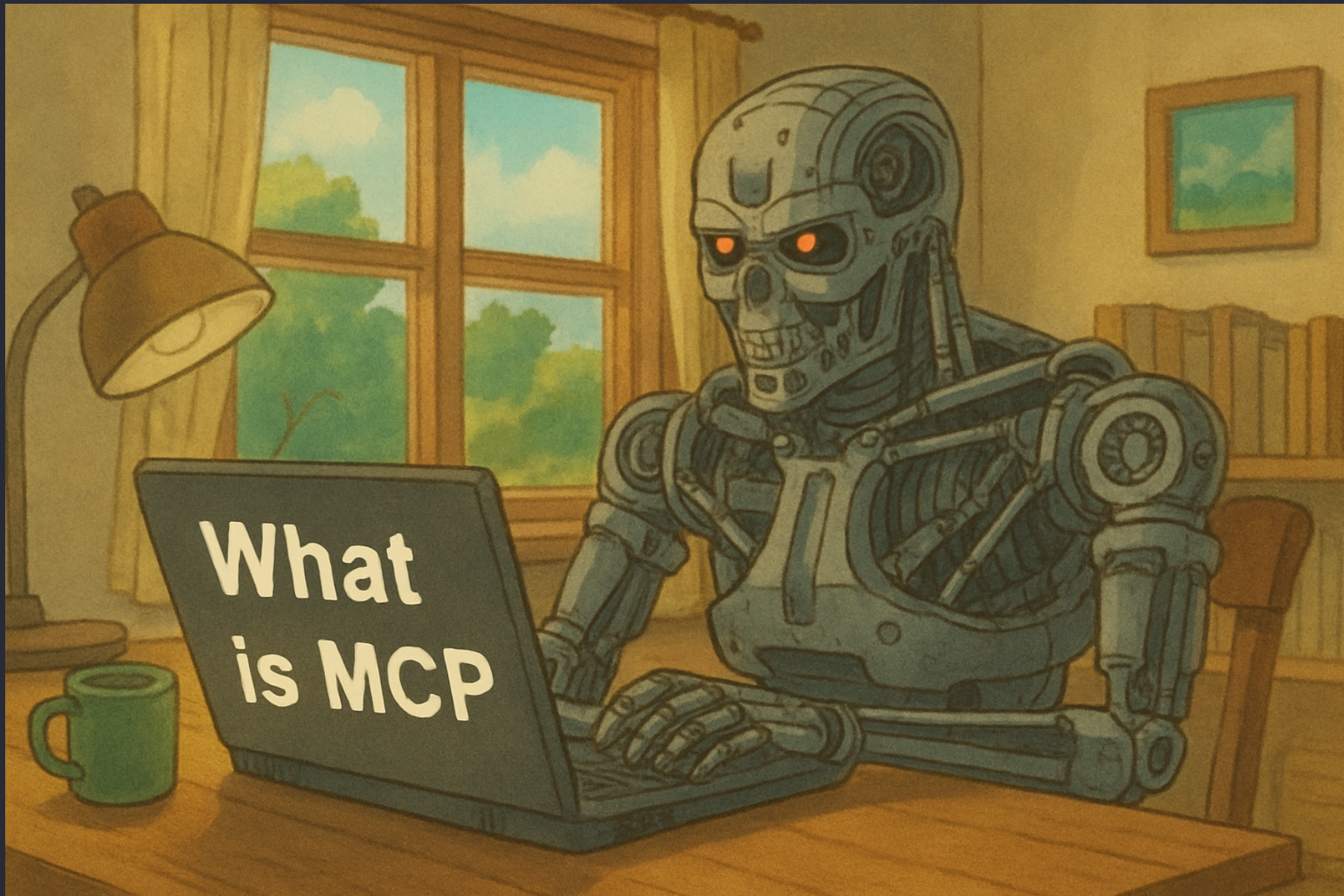
Why MCP ?

AGE OF AI IS UPON US



Why MCP ?

- Enable AI to perform real world tasks
- Standardized specification for LLM communication
- Running autonomous agents
- A new way for discovering and exposing RPC for any system

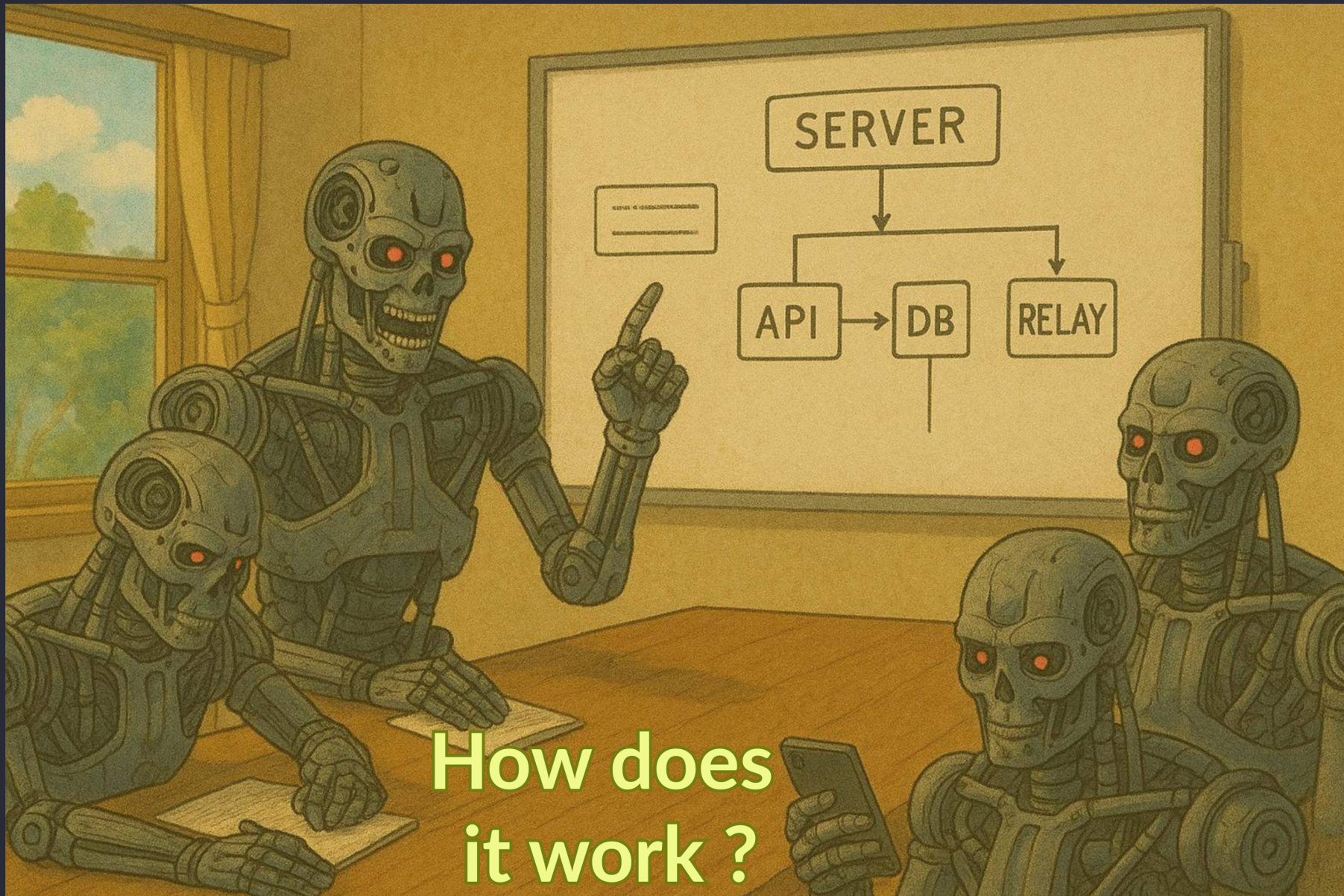


What is MCP ?

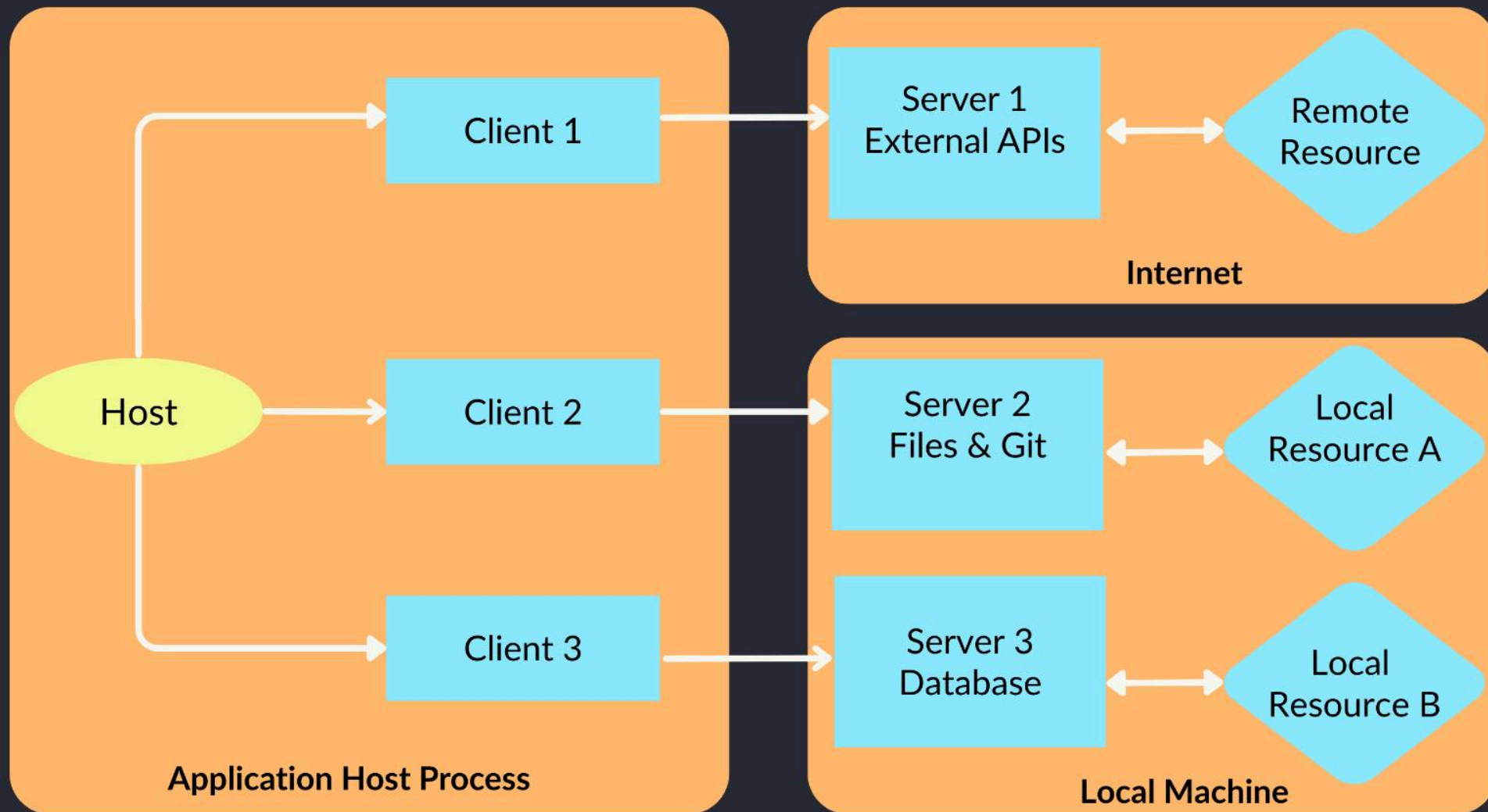
- Model Context Protocol
- Open protocol to standardize how applications provide context to large language models (LLMs)
- Standardized way to connect AI models to different data sources and tools

Model Context Protocol 🤝

- **Host:** coordinates and manages one or multiple MCP clients
- **Client:** maintains a connection to an MCP server and obtains context from an MCP server for the MCP host to use
- **Server:** provides context and data to MCP clients



How does
it work ?



Transport Layer

- Uses JSON RPC 2.0
- Transport
 - StdIO
 - Streamable HTTP transport

Transport Layer

```
{  
  "jsonrpc": "2.0",  
  "id": 3,  
  "method": "tools/call",  
  "params": {  
    "name": "weather_current",  
    "arguments": {  
      "location": "Jaipur",  
      "units": "metric"  
    }  
  }  
}
```

Model Context Protocol



Resources



Tools



Prompts

Resources

- Provide structured access to information that the host application can retrieve as context
- Expose data from files, APIs, databases, or any other source that an AI needs to understand
- Use URI-based identification
- Examples:
 - `file:///Documents/Travel/Passport.pdf`
 - `user://{username}`

Tools

- Perform actions through server-implemented functions
- Schema-defined interfaces that LLMs can invoke
- Each tool defines a specific operation with typed inputs and outputs
- Examples:
 - `update_replica_count(name: string, count: int)`
 - `search_flights(src: string, dst: string)`

Prompts

- Reusable templates that guide model interactions
- Allow authors to provide parameterized input

```
{  
  "name": "plan-vacation",  
  "description": "Guide through vacation planning process",  
  "arguments": [  
    {"name": "destination", "type": "string", "required": true },  
    {"name": "duration", "type": "number", "description": "days" },  
  ]  
}
```


Building Block	Purpose	Who Controls It	Real-World Example
Tools	For AI actions	Model-controlled	Search flights, send messages
Resources	For context data	Application-controlled	Documents, calendars
Prompts	For interaction templates	User-controlled	Set meetings, Draft email

Let's build
our server



Filesystem MCP

```
import (  
    ...  
    "github.com/mark3labs/mcp-go/mcp"  
    "github.com/mark3labs/mcp-go/server"  
)
```

```
s := server.NewMCPServer(  
    "file-manager",  
    "1.0.0",  
    server.WithLogging(),  
)
```

Filesystem MCP

```
// Register the create_file tool
s.AddTool(
    mcp.NewTool("create_file",
        mcp.WithDescription("Create a new file with specified content"),
        mcp.WithString("path",
            mcp.Description("Path where file should be created"),
            mcp.Required()),
        mcp.WithString("content",
            mcp.Description("Content to write to the file")),
    ),
    handleCreateFile
)
```

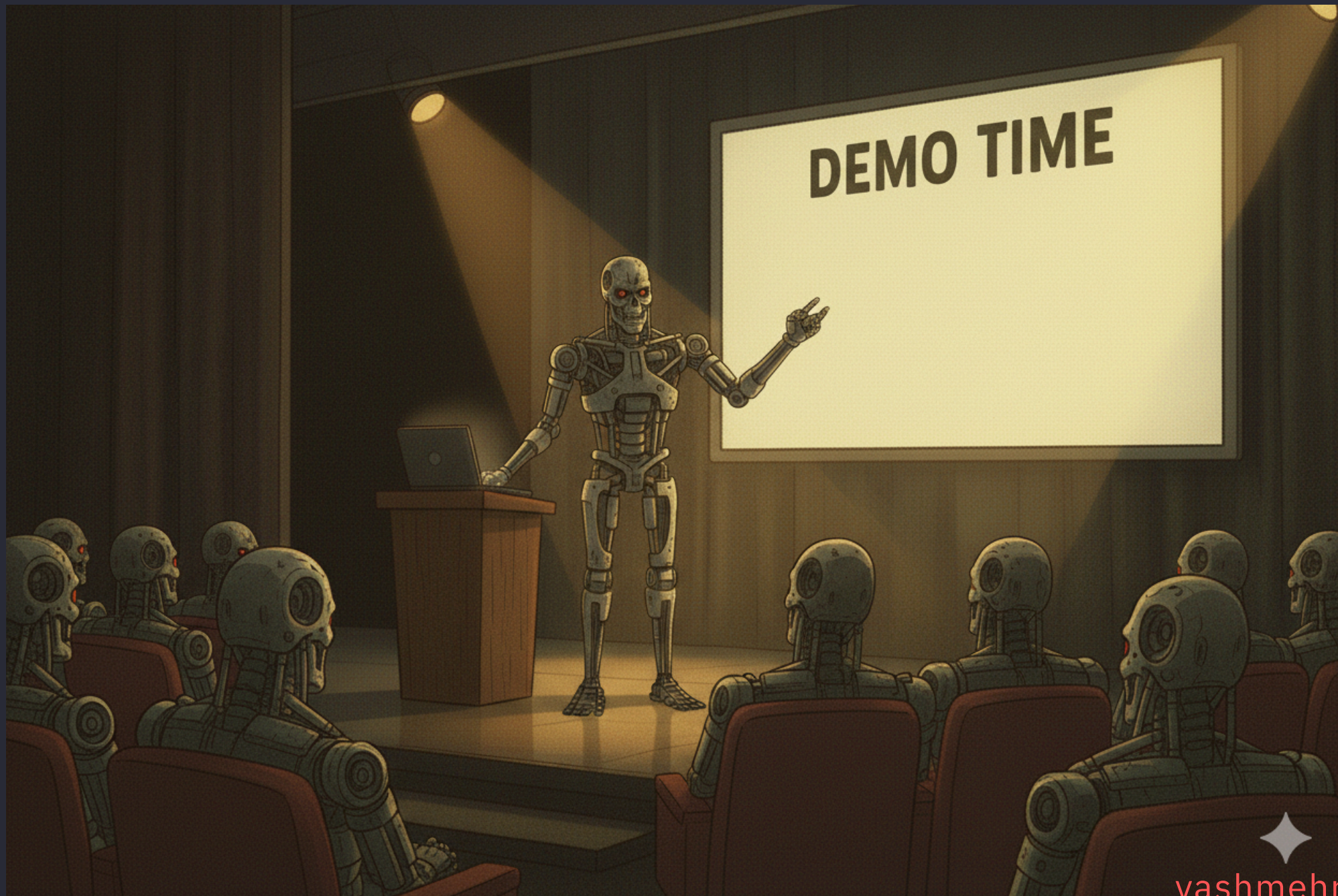

Filesystem MCP

```
func handleCreateFile(ctx context.Context, request mcp.CallToolRequest) (*mcp.CallToolResult, error) {
    path := request.GetString("path", "")
    if path == "" {
        return ...
    }
    content := request.GetString("content", "")

    dir := filepath.Dir(path)
    _ = os.MkdirAll(dir, 0755)

    if err := os.WriteFile(path, []byte(content), 0644); err != nil {
        return ...
    }

    return ...
}
```



Integrating MCP with HTTP

- MCP works out of the box with HTTP
- `server.NewStreamableHTTPServer` creates an HTTP Server instance
- Server implements `ServeHTTP` method to support the standard `http.Handler` interface

Integrating MCP with HTTP



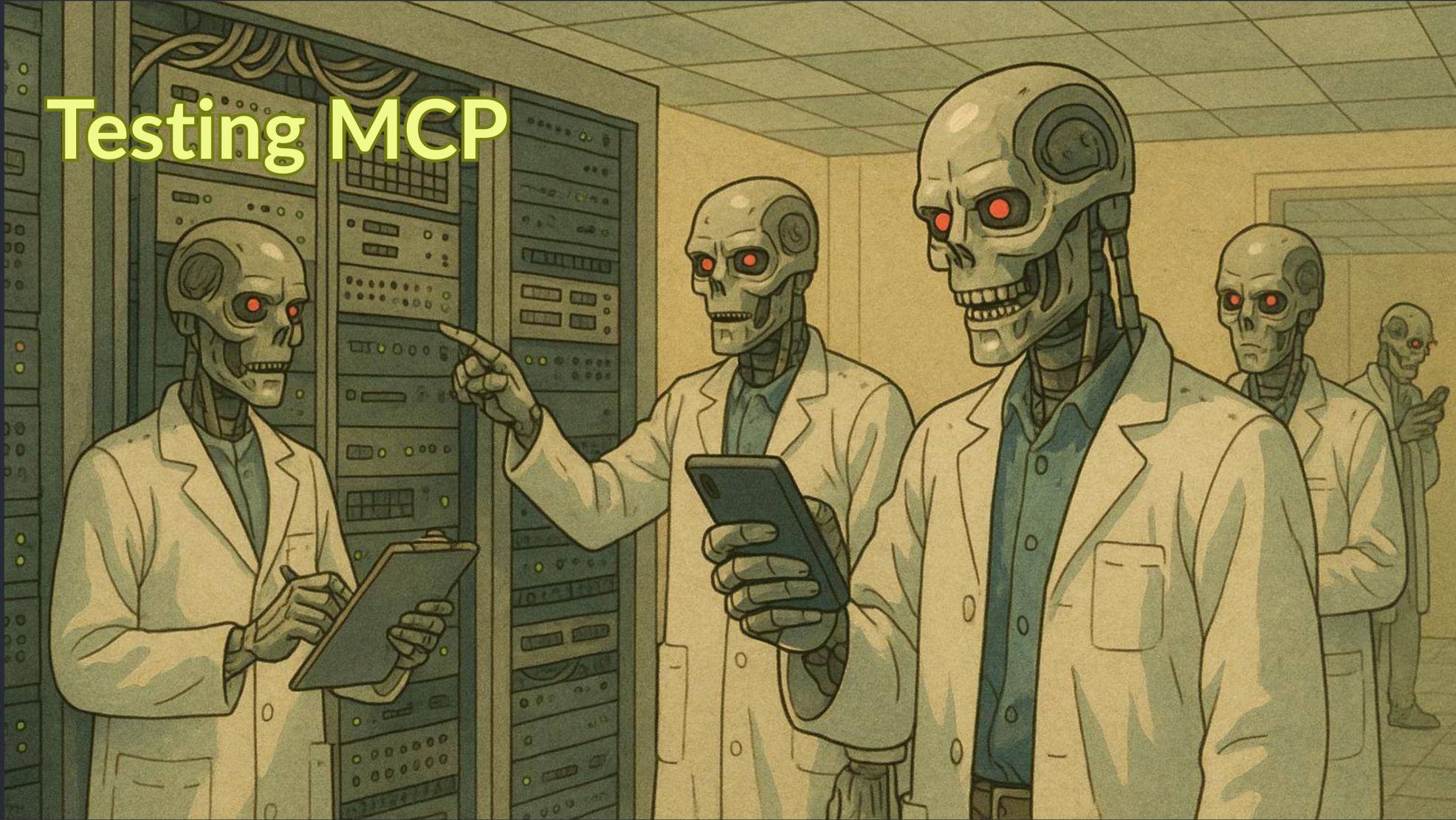
```
s := server.NewMCPServer("name", "version")
httpServer := server.NewStreamableHTTPServer(s)

// Start server at any port
httpServer.Start(":port")

// Use this server with echo
e.POST("/mcp", echo.WrapHandler(httpServer))

// Use this server with gin
g.POST("/mcp", gin.WrapH(httpServer))
```


Testing MCP



Testing MCP Server

MCP Client is used to call the server

```
import "github.com/mark3labs/mcp-go/client"

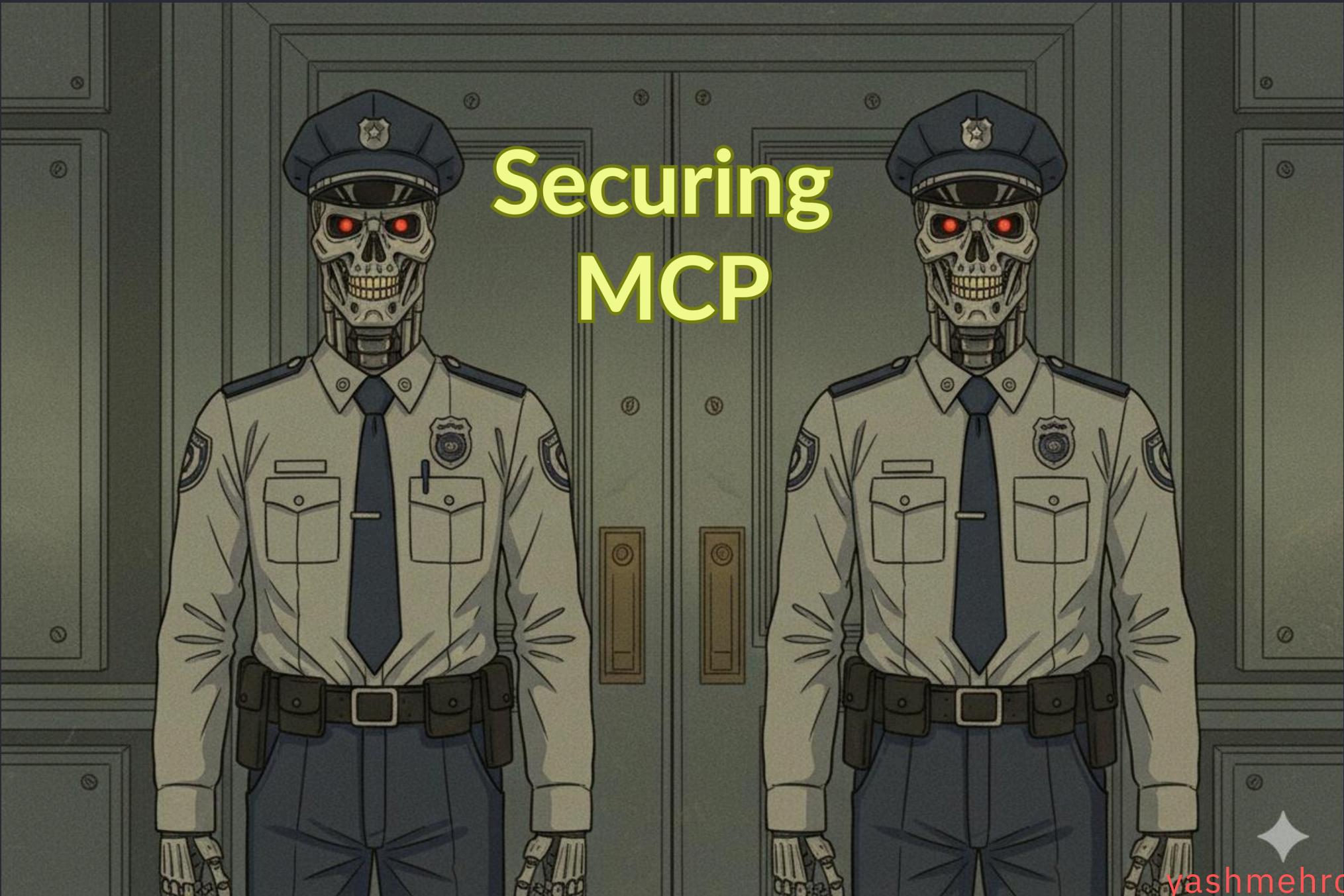
e := echo.New()
e.POST("/mcp", echov4.WrapHandler(mcpHTTPServer.ServeHTTP))
testServer = httptest.NewServer(e)

mcpClient, err := client.NewStreamableHttpClient(
    testServer.URL + "/mcp",
)
```

Testing MCP Server

```
func TestFileMCP(t *testing.T) {  
    result, err := mcpClient.CallTool(DefaultContext,  
        mcp.CallToolRequest{  
            Params: mcp.CallToolParams{  
                Name: "create_file",  
                Arguments: map[string]any{  
                    "path": "/tmp/new_file",  
                },  
            },  
        })  
    if _, err := os.Stat("/tmp/new_file"); err != nil {  
        t.Fatalf("file not found")  
    }  
}
```


Securing MCP



Security (Do's)

- ✓ Always check if user is authorised to call the tools and resources
- ✓ Use tokens for authentication
- ✓ Log everything & maintain audit trails

Security (Dont's)

- ✗ Never pass tokens through, only accept tokens issued specifically for the server and not for other APIs
- ✗ Validate and sanitize all the inputs
- ✗ Don't give agents complete access, be restrictive with what it is allowed to run

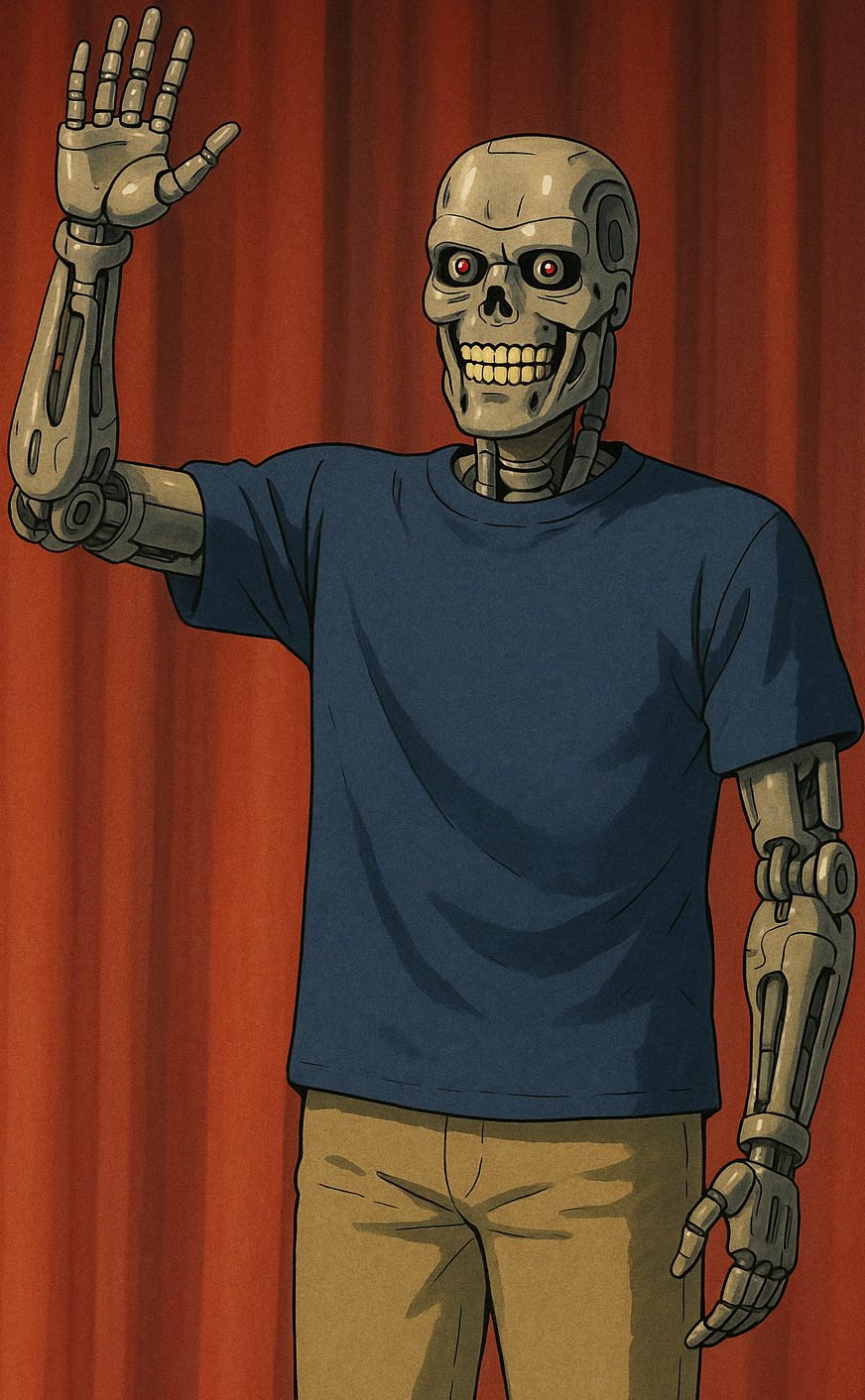
Bonus Tips

- Design around user intent, not API endpoints
- AI models rely entirely on what is told to them in descriptions and responses
- Use errors effectively in responses to guide the model
- Ecosystem is rapidly evolving, always keep learning

“ To err is human

”

“ To err is human, and handling `err` is what makes us gophers ”



Thanks 🙌



yashmehrotra.com



@yashmehrotra.com



@yashm95



github.com/yashmehrotra



in/yashmehrotra