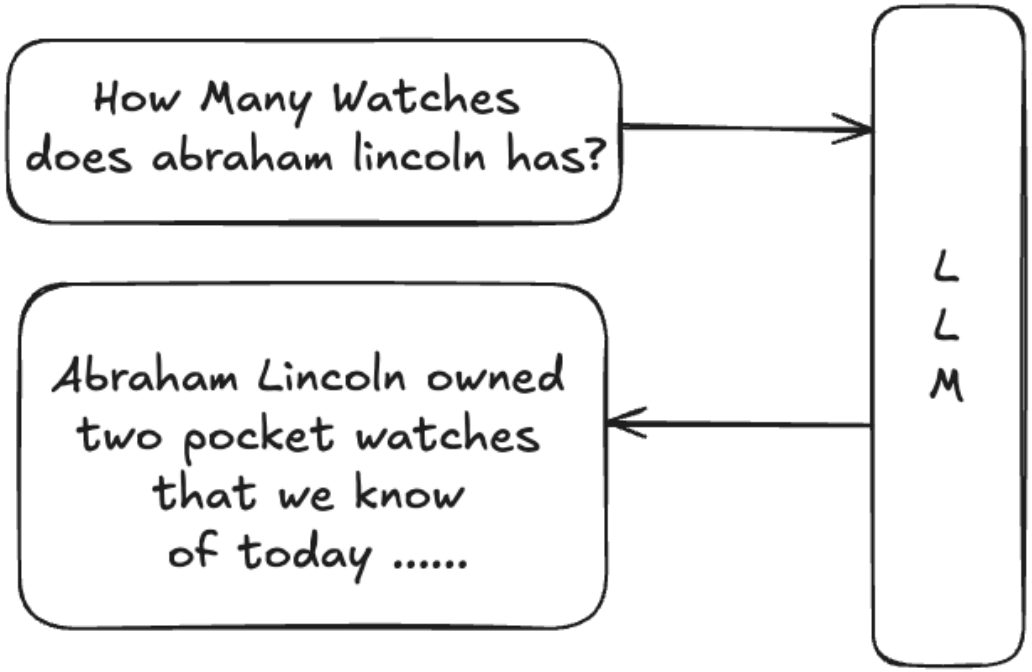
Model Context Protocol (MCP) Internals, Security, and Cloud Deployments Using Spring AI

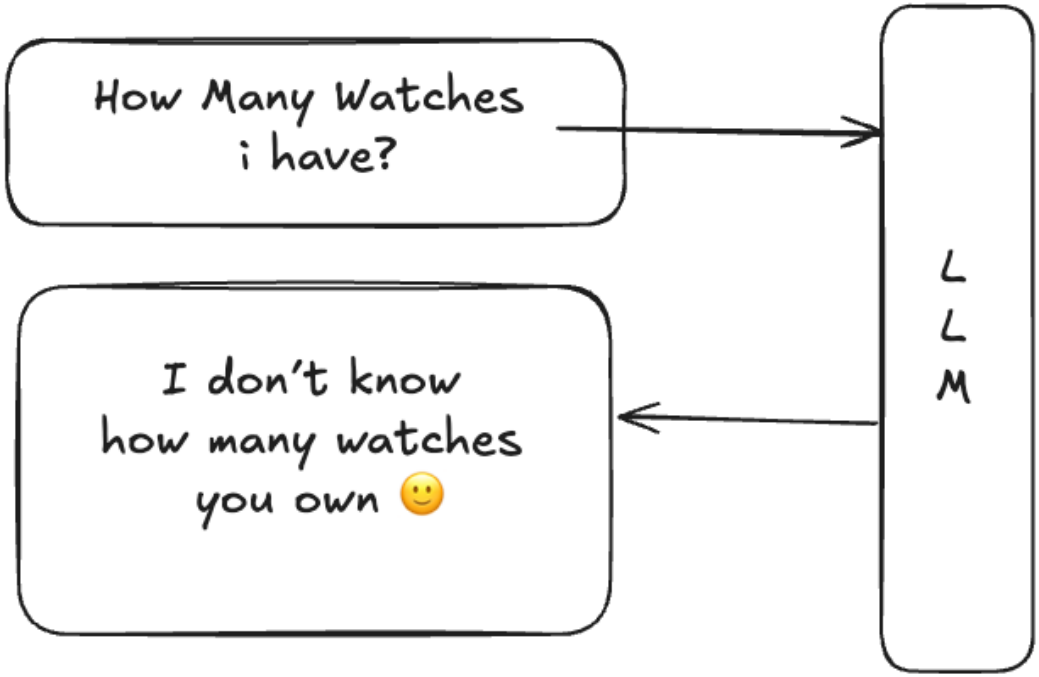
Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Internals

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)



Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

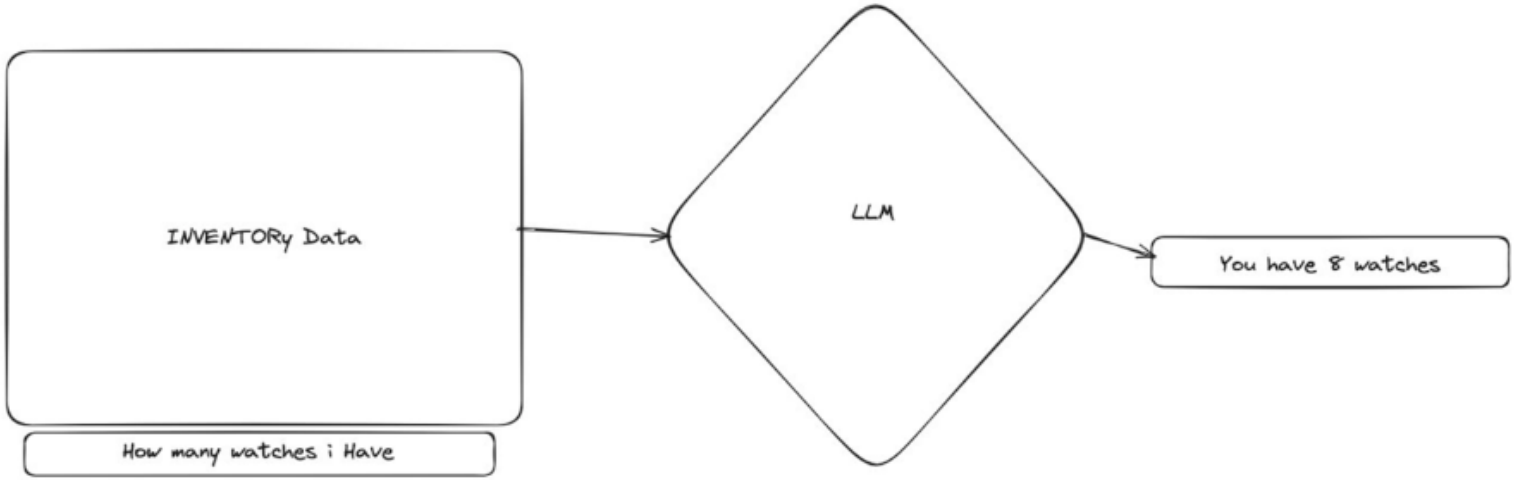


Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

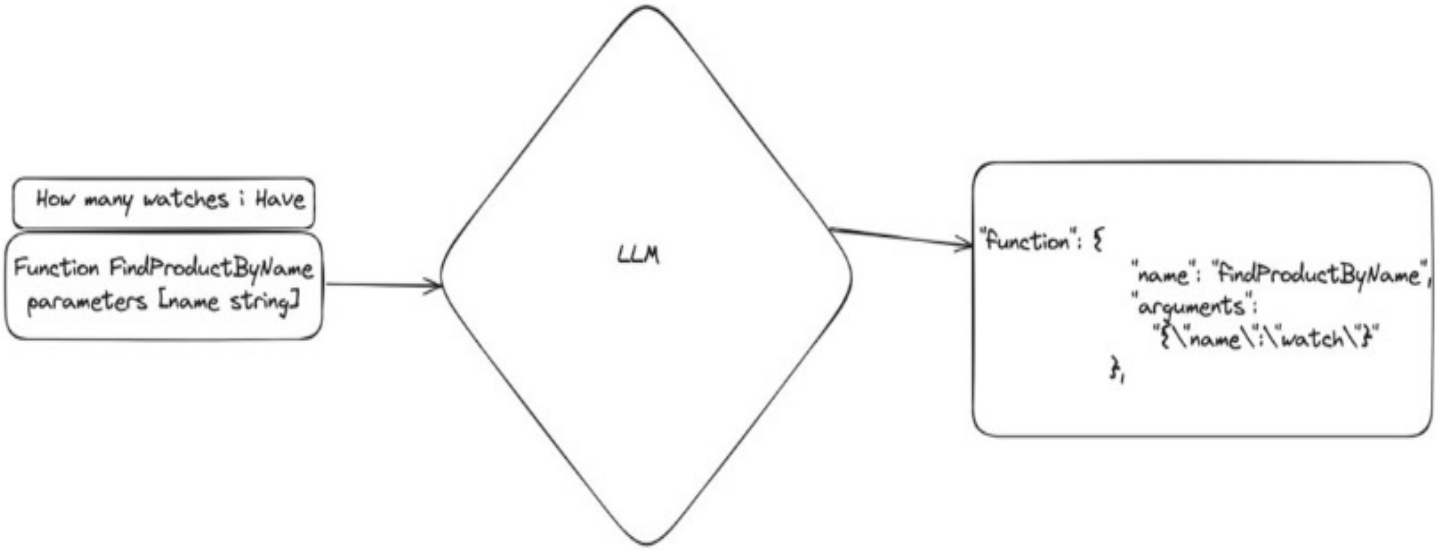
"A large language model
knows only what it has been trained on;
beyond that, only
educated guesses- and hallucinations..."

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

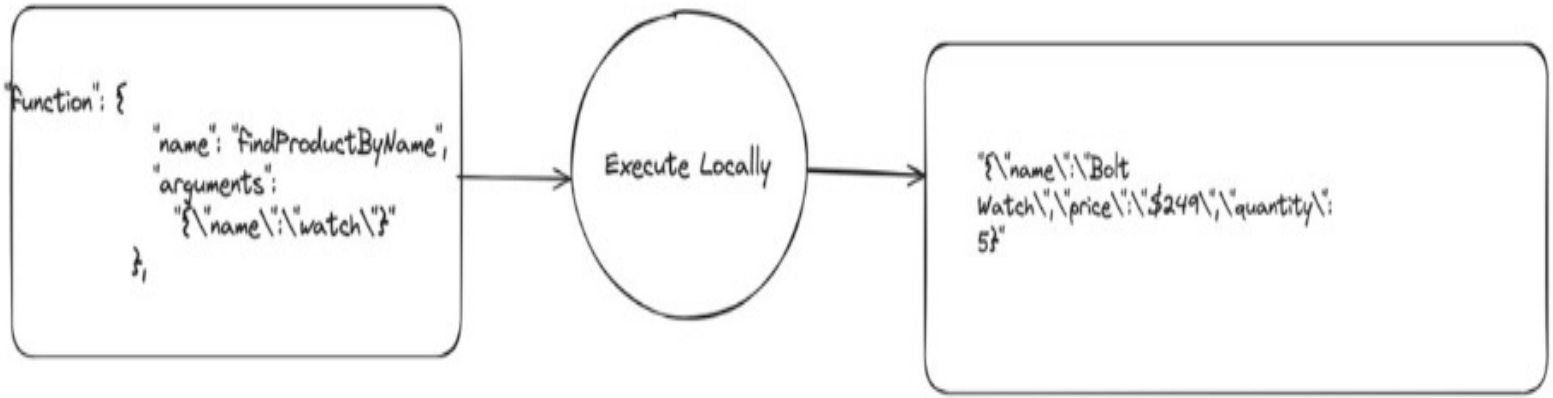
How to make Context - Aware



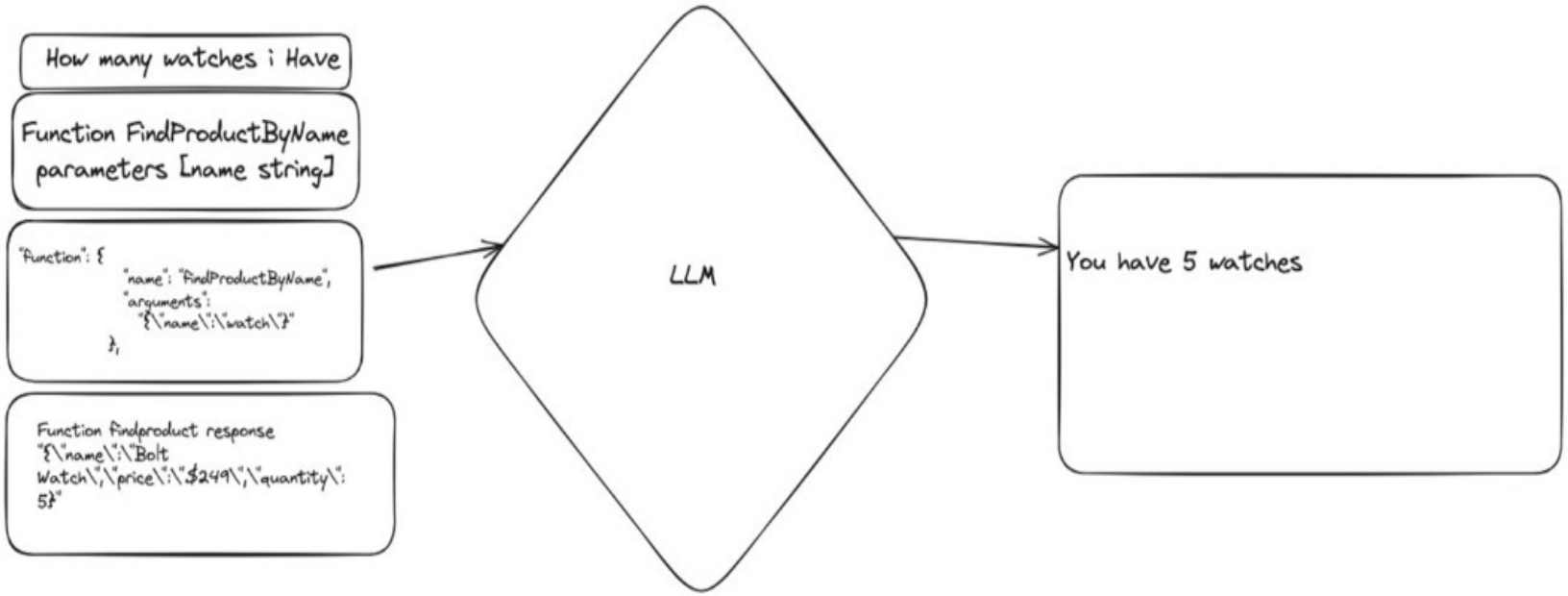
Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)



Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

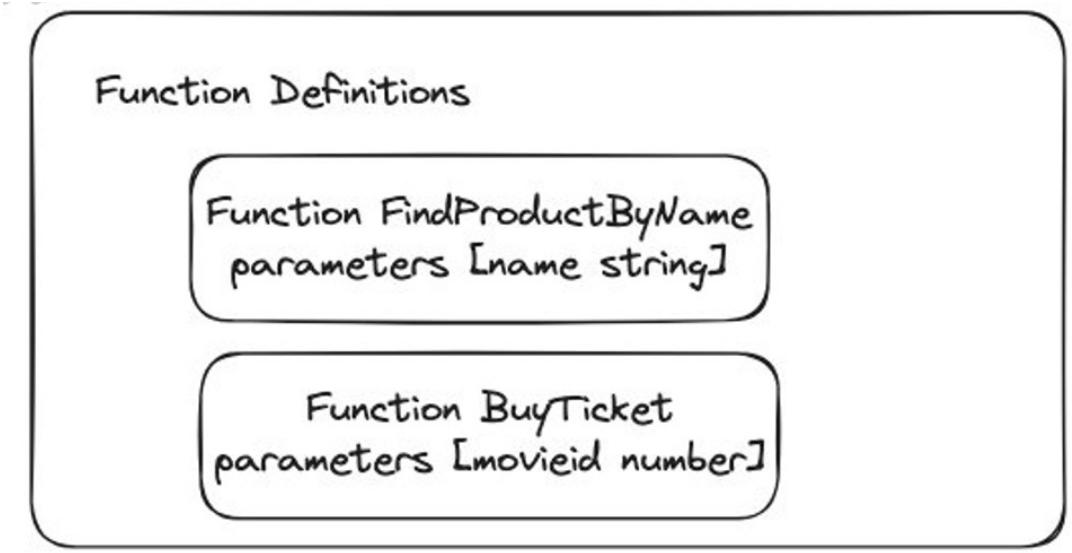


Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

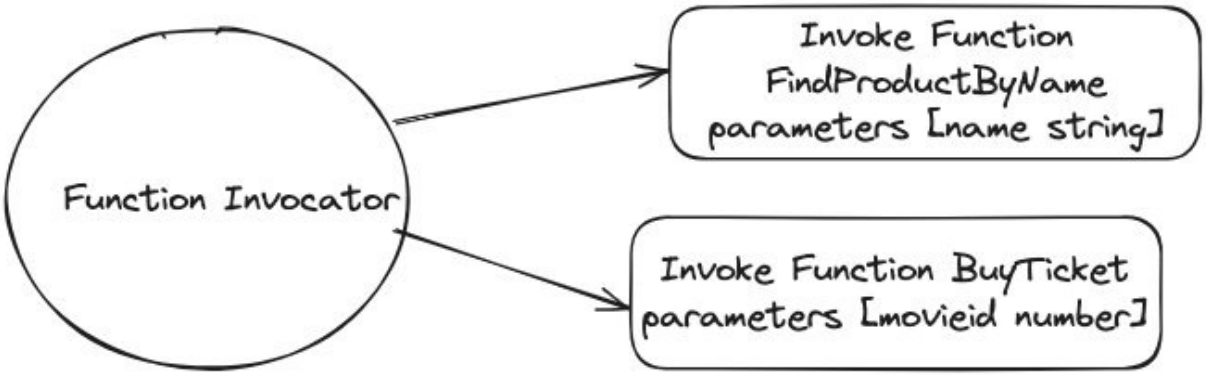


Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

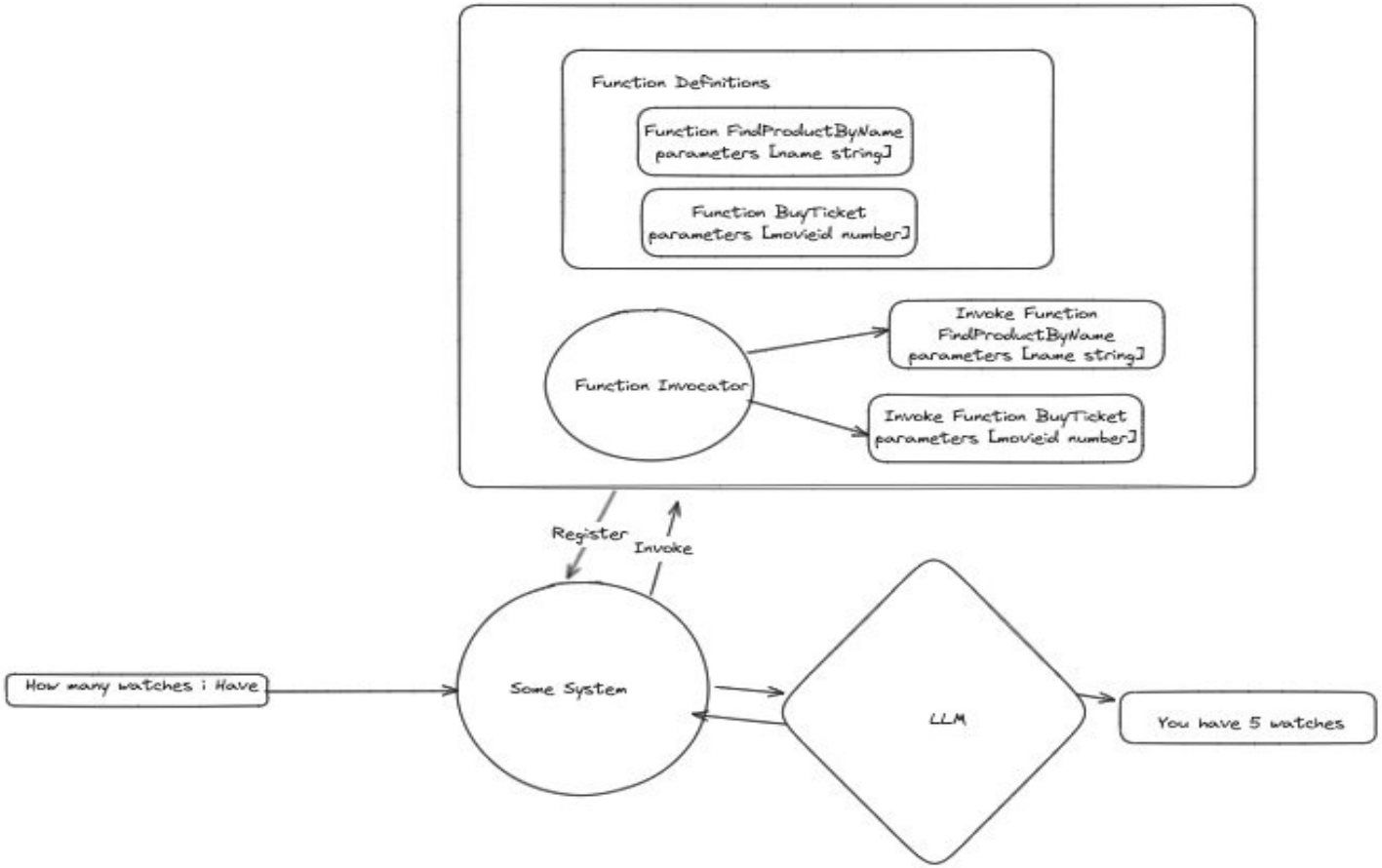
Split Tool Calling



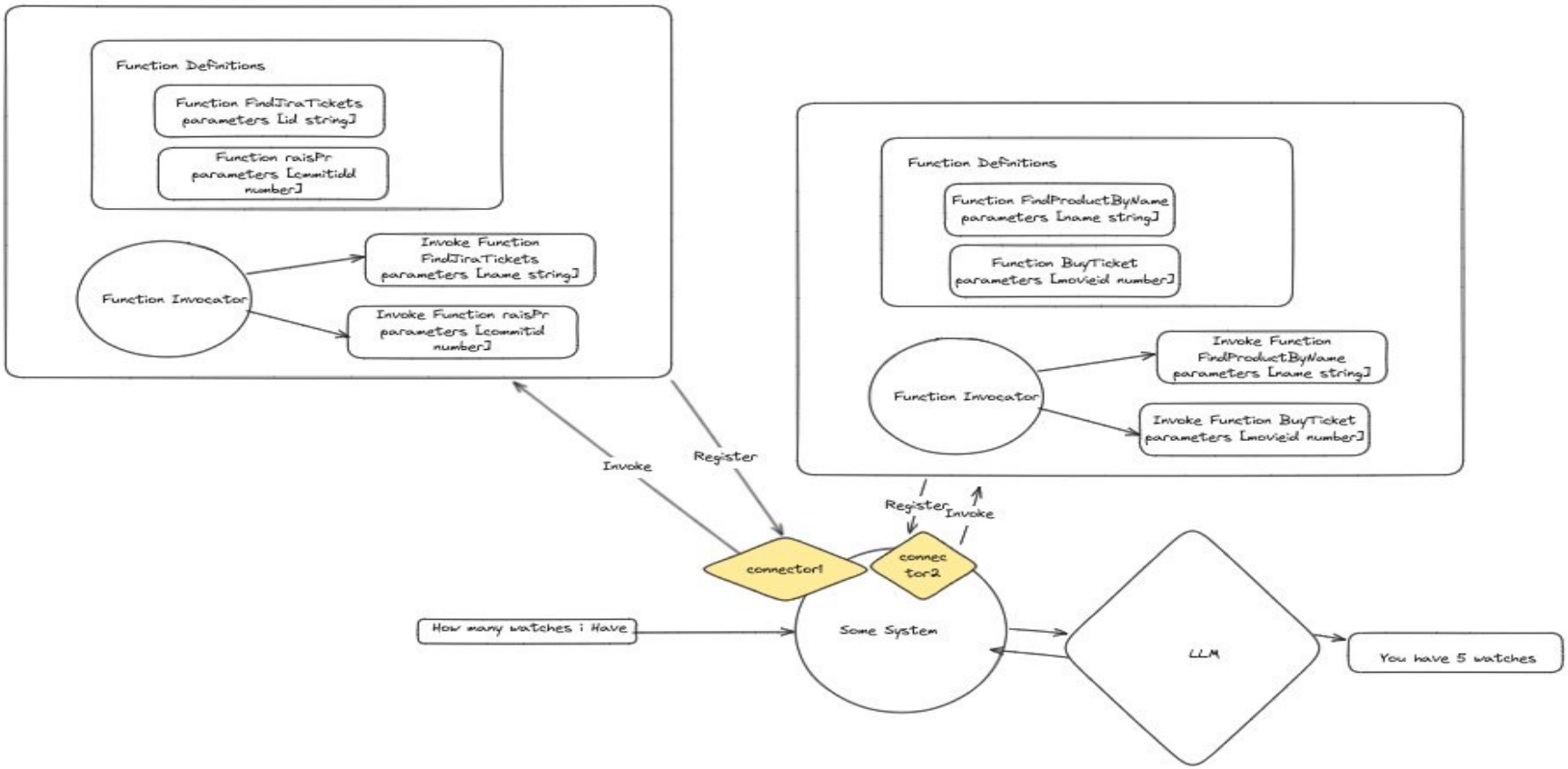
Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)



Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

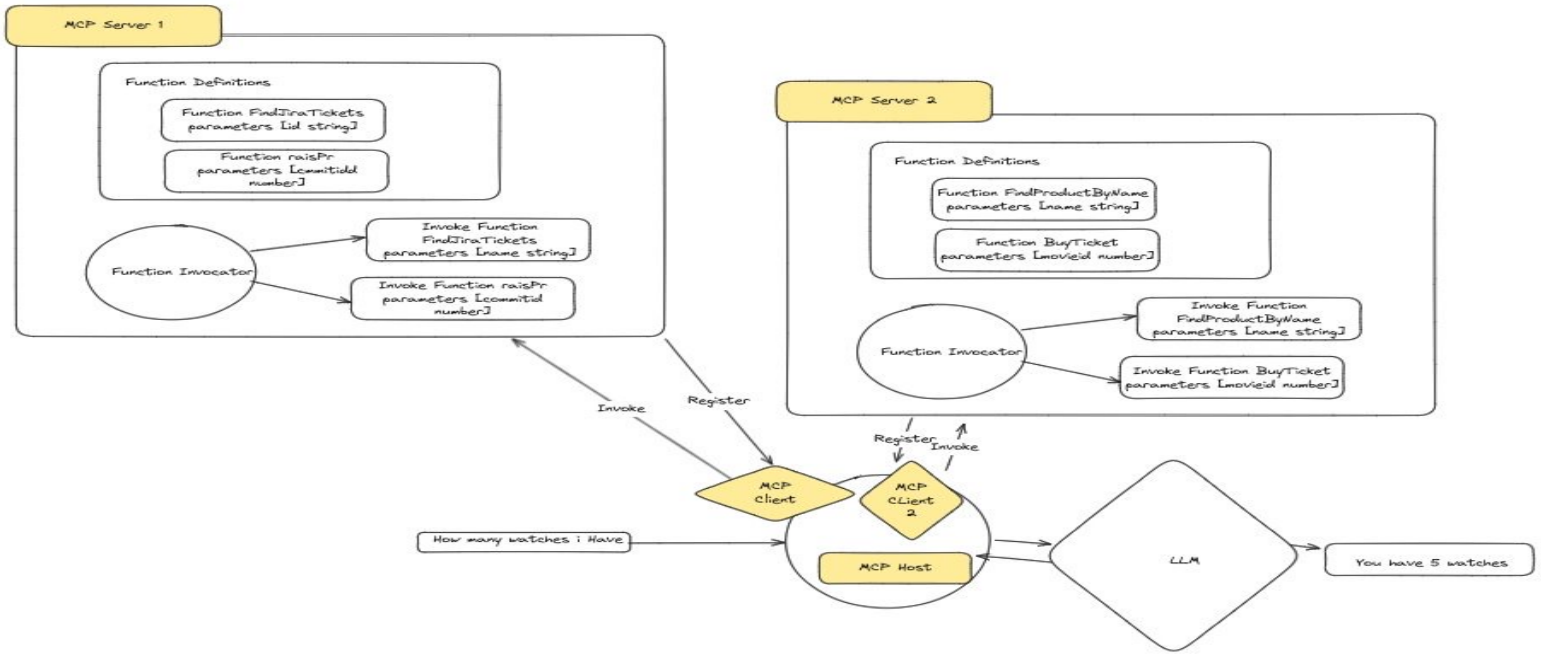


Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)



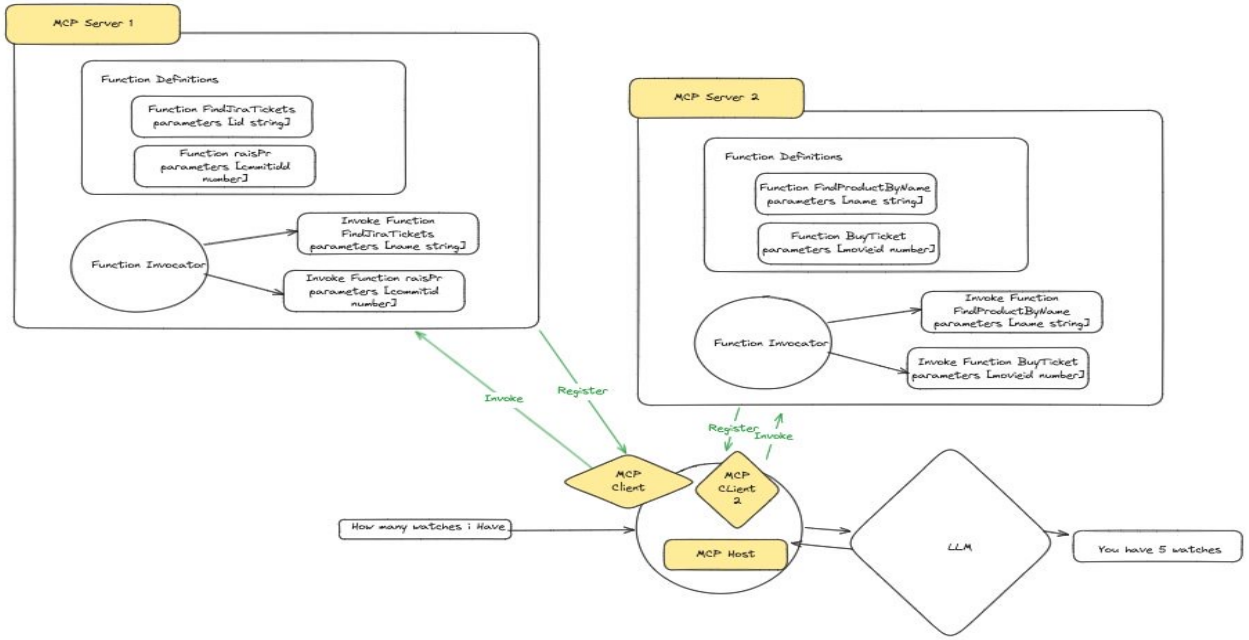
Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Model context protocol (MCP)



Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

How all these connections Exchange



Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

The Protocol – JSON RPC

- JSON-RPC = method + params + id → result or error
- Response is either result or error, never both
- Clean, minimal structure for tool invocation

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

The Protocol – JSON RPC

Request

```
{
  "jsonrpc": "2.0",
  "method": "subtract",
  "params": [42, 23],
  "id": 1
}
```

Response

```
{
  "jsonrpc": "2.0",
  "result": 19,
  "id": 1
}
```

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

The Protocol – JSON RPC

Request

```
{
  "jsonrpc": "2.0",
  "method": "buyMovieTicket",
  "params": {
    "movieName": "Inception",
    "showTime": "2025-06-17T19:00:00",
    "seats": ["A1", "A2"],
    "userId": "user_123"
  },
  "id": 101
}
```

Response

```
{
  "jsonrpc": "2.0",
  "error": {
    "code": -32601,
    "message": "Method not found"
  },
  "id": 101
}
```

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

MCP Protocol - The Three Core Calls

- initialize → Returns version, capabilities
- getTools → Lists all tool definitions
- invokeTool → Executes tool with input

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Initialize

Request

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "initialize",
  "params": {
    "protocolVersion": "2025-03-26",
    "capabilities": {
      "roots": {
        "listChanged": true
      },
      "sampling": {}
    },
    "clientInfo": {
      "name": "Visual Studio Code",
      "version": "1.101.1"
    }
  }
}
```

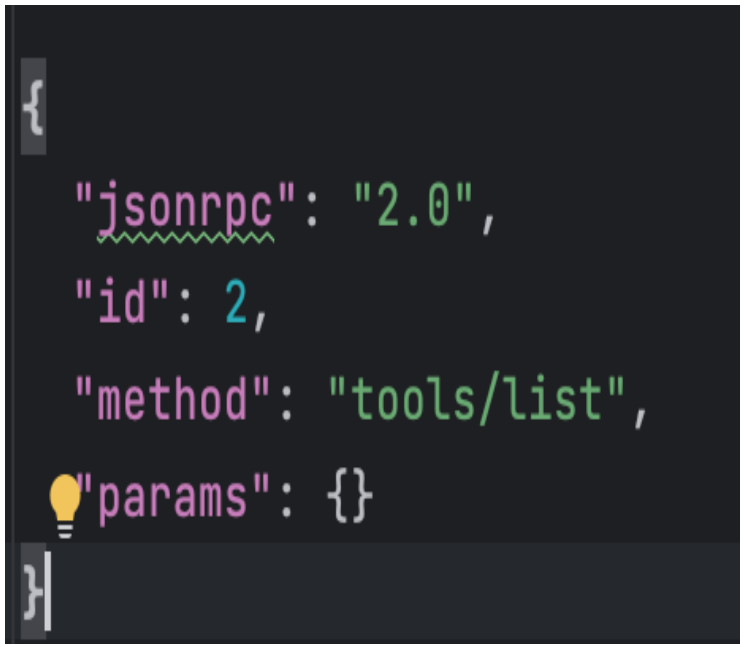
Response

```
{
  "jsonrpc": "2.0",
  "result": {
    "protocolVersion": "2024-11-05",
    "serverInfo": {
      "name": "BrowserServer",
      "version": "0.1.0"
    },
    "capabilities": {
      "tools": {
        "listChanged": true
      }
    },
    "instructions": "This server provides browser-related functionalities.
  },
  "id": 1
}
```

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Discovery

Request



```
{
  "jsonrpc": "2.0",
  "id": 2,
  "method": "tools/list",
  "params": {}
}
```

Response

```
{
  "jsonrpc": "2.0",
  "result": {
    "tools": [
      {
        "name": "open_google_chrome",
        "description": "Open Google Chrome with the default URL
        "inputSchema": {
          "type": "object",
          "properties": {},
          "required": []
        }
      }
    ]
  },
  "id": 2
}
```

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Execute

Request

```
{
  "jsonrpc": "2.0",
  "id": 3,
  "method": "tools/call",
  "params": {
    "name": "open_google_chrome",
    "arguments": {}
  }
}
```

Response

```
{
  "jsonrpc": "2.0",
  "result": {
    "content": [
      {
        "type": "text",
        "text": " \n"
      }
    ]
  },
  "id": 3
}
```

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Transports

1. STDIO
2. SSE (Depc)
3. Streamable HTTP
4. Stateless Streamable HTTP

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Let's Kickstart!!

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Pre-Requisite

1. Java 21
2. Docker
3. Git
4. Node.js 18+
5. VS Code or IntelliJ IDEA
6. Github Copilot
7. AWS CLI, gcloud CLI, and Azure CLI (anyone)

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

```
npm i -g @go-task/cli
npm i -g @muthuishere/spinx
```

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Creating MCP Server Using Spring AI

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Download the below Repository

https://github.com/shaamam/jf-ch-1

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Security

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

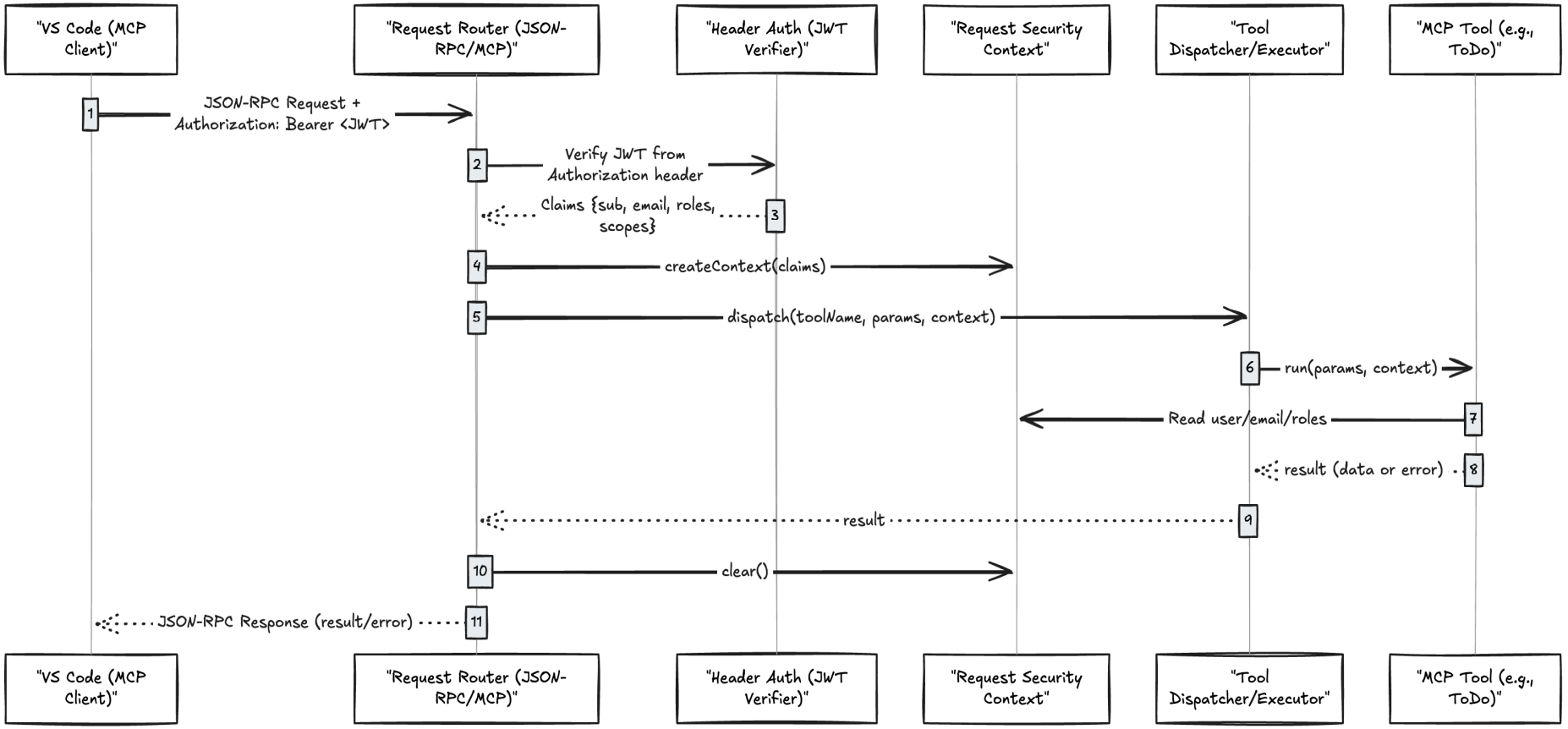
What is Bearer Token ?

- A bearer token proves authentication. Whoever holds it gains access. Protect it strictly.
- A token string (often JWT) sent as Authorization: Bearer <token> after login.
- Access is granted by possession; server verifies signature, issuer, audience, and expiry.
- Handle like a password: HTTPS only, short-lived, rotate/revoke on suspicion.

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

How Bearer Token Auth Works ?

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)



Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

MCP Security Implementation with Bearer token

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

https://firebasejwt.muthuishere.site/

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Download the below Repository

https://github.com/shaamam/jf-ch-2

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Dynamic Client Registration (DCR)

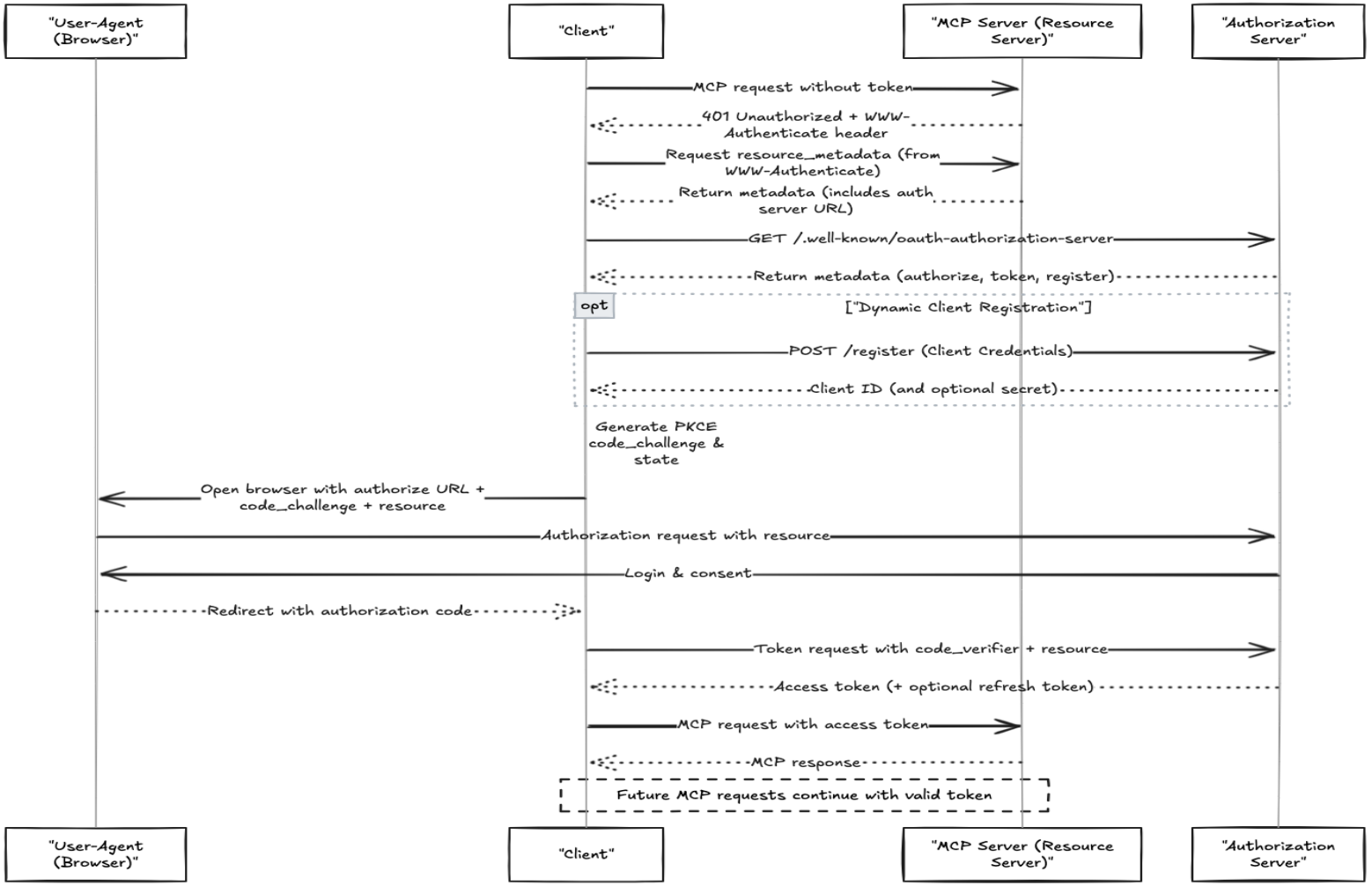
- Based on **OAuth 2.0 Dynamic Client Registration (RFC 7591)**.
- Allows MCP clients to obtain credentials dynamically from the AS.
- Common endpoint: /register.
- Client receives a **client_id** (and possibly a **secret**) automatically.
- Reduces friction: no need for manual pre-registration.

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

What is /.well-known?

- /.well-known is a standard path for publishing discovery metadata.
- It allows clients to auto-detect configuration without hardcoded URLs.
- OAuth uses it to expose authorization and resource server details.
- This ensures interoperability across different clients and platforms.

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)



Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

MCP Security Implementation with DCR

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Download the below Repository

https://github.com/shaamam/jf-ch-3

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Cloud Deployments

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Deploying MCP Servers

- MCP servers communicate over **streaming HTTP** (long-lived responses).
- Perfect match for modern containerized deployments.
- We'll explore:
  - Containers & registries
  - Serverless container platforms
  - Typical IaC challenges
  - The lightweight deployment workflow

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Serverless Container Platforms

- **AWS Fargate** — managed ECS tasks per container.
- **GCP Cloud Run** — scales HTTP services to zero.
- **Azure Container Apps** — KEDA-based autoscaling.
- All handle **HTTP streaming** transparently.

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

What is Spinx?

- Spinx lets you deploy once and run anywhere across AWS, GCP, and Azure.
- It uses simple YAML configs for unified, human-friendly multi-cloud deployments.
- Manage setup, deploy, logs, and cleanup securely with one consistent CLI command.

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Spinx Deployment Pattern

- Build the MCP server container.
- Push to registry.
- Deploy to Fargate / Cloud Run / ACA.
- Tail logs while streaming requests.
- Tear down cleanly when done.

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Deploying MCP Server on Serverless Cloud Platforms

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Pre-Requisite

Bash:

aws configure

or

gcloud auth login

or

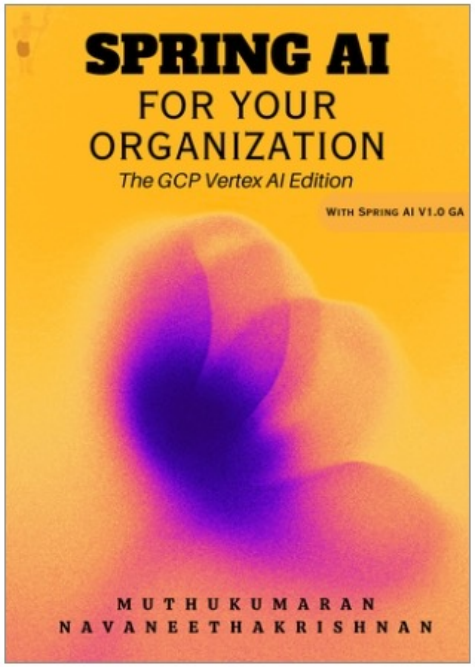
az login

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Download the below Repository

https://github.com/shaamam/jf-ch-4

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)



References

- Book:
  - ✓ Spring AI For your Organization (The GCP Vertex Edition)
    by Muthukumaran Navaneethakrishnan
    https://leanpub.com/springai

- Code:
  - ✓ https://github.com/Shaamam/javafest-mcp-internals-security-cd

https://www.linkedin.com/in/shaama-m-03011523?
https://www.linkedin.com/in/muthuishere/

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)

Q&A

Muthukumaran(muthuishere@gmail.com) &
Shaama(shaama0704@gmail.com)