



do your thing

# External processing, internal leverage MCP tool calls to REST with Envoy

# Providing superior customer experience through our scalable technologies



Jens Kat – Senior software engineer  
for ING's service mesh

 **2.000+**  
microservices in  
production

**1/3** of 60.000  
works in tech

 Located over  
**35 countries**

We serve around  
**39 million**  
customers

Over  
**100**  
countries

Annually  
**4.5 billion**  
payments



do your thing

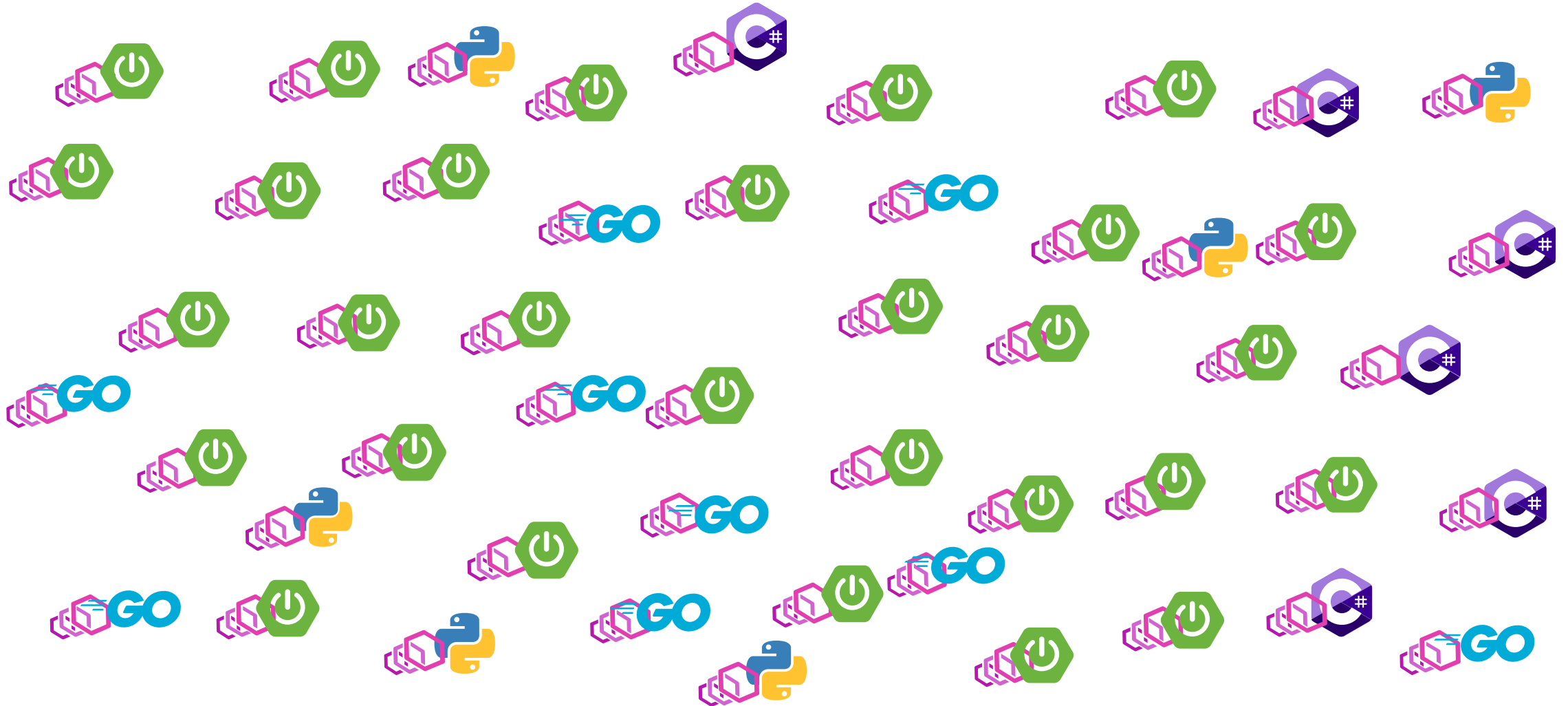


# Agenda

## Extending Envoy to support AI use-cases

1. ING service mesh
2. AI agents came along
3. Envoy and extension points
4. Using ext\_proc
5. Open-sourcing
6. Wrap-up

# We have an HTTP API based service mesh



# And then AI agents came along



AI



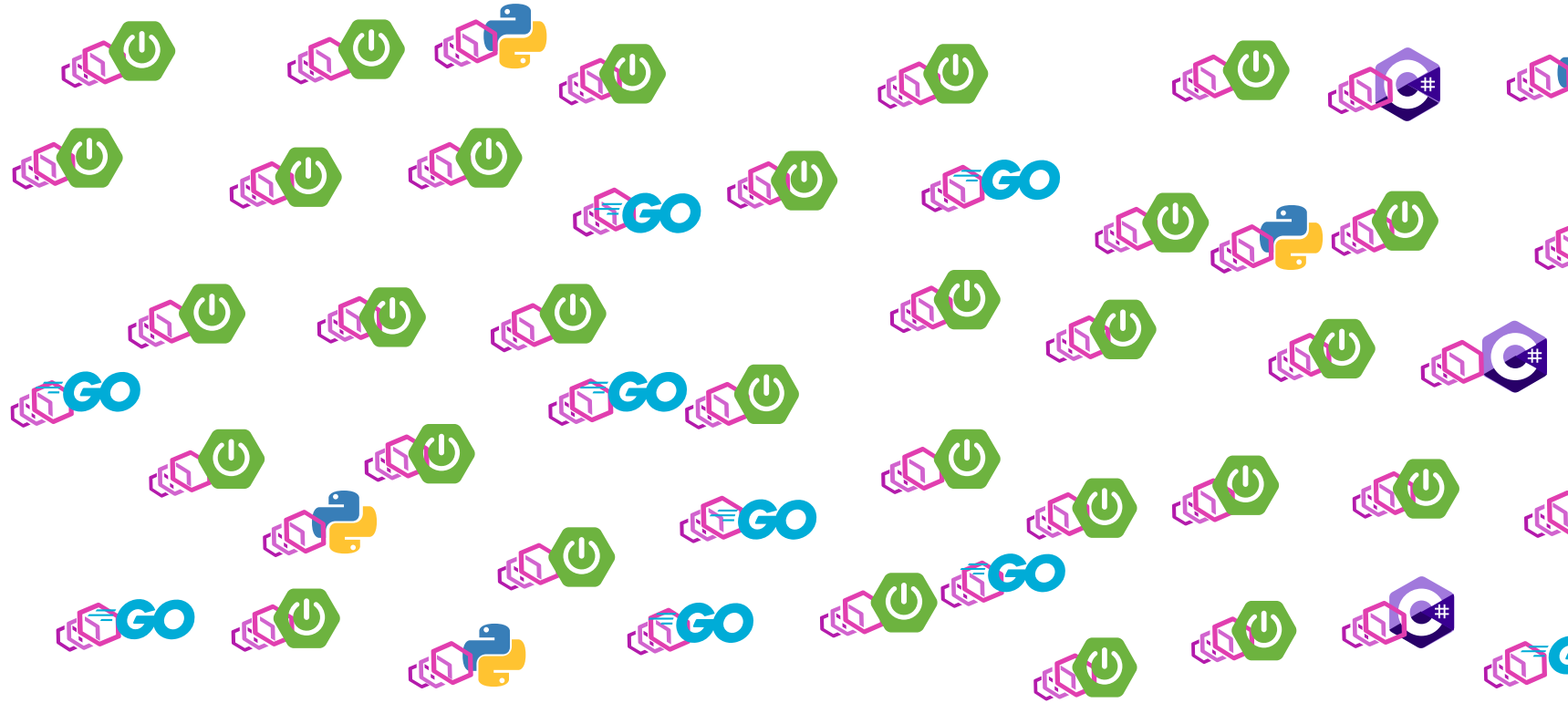
AI



AI

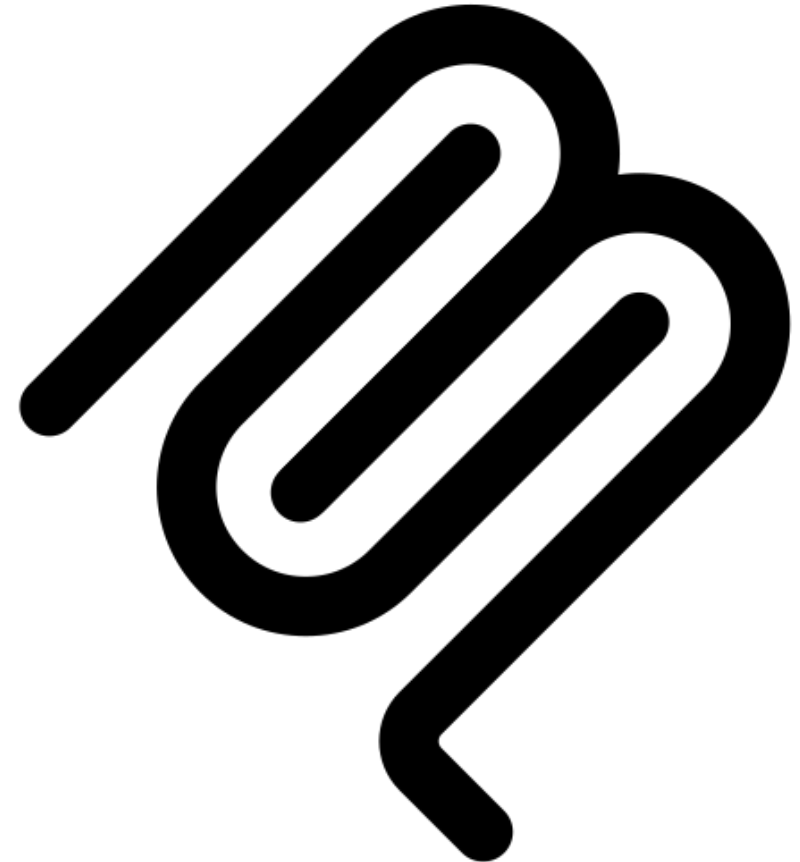
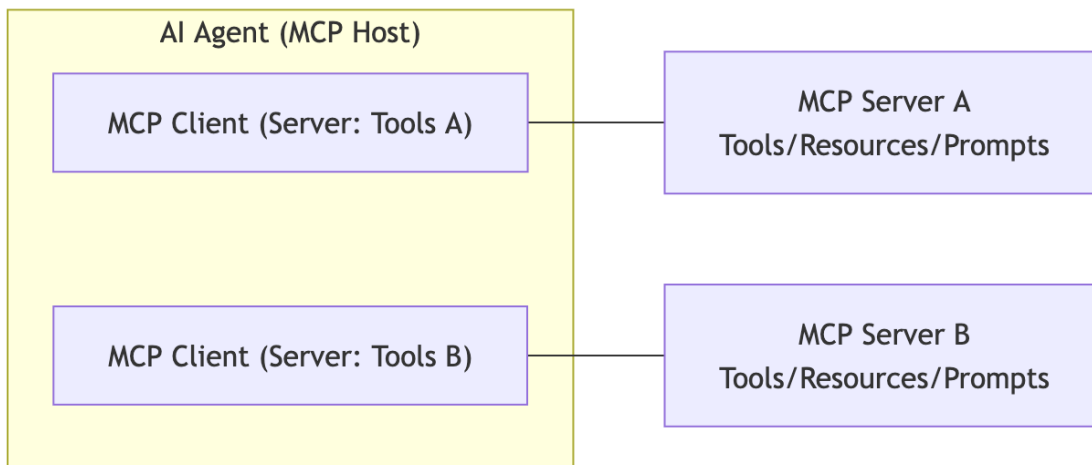


AI agent



# Connect with model context protocol (MCP) - a standard for connecting AI with external systems

- MCP is an open-standard to integrate and share data with systems using JSON-RPC
- Started by Anthropic in 2024 (and now adopted by many more)
- 16 months 'old' 🎉



# MCP support with frameworks scattered and duplicated in our service mesh



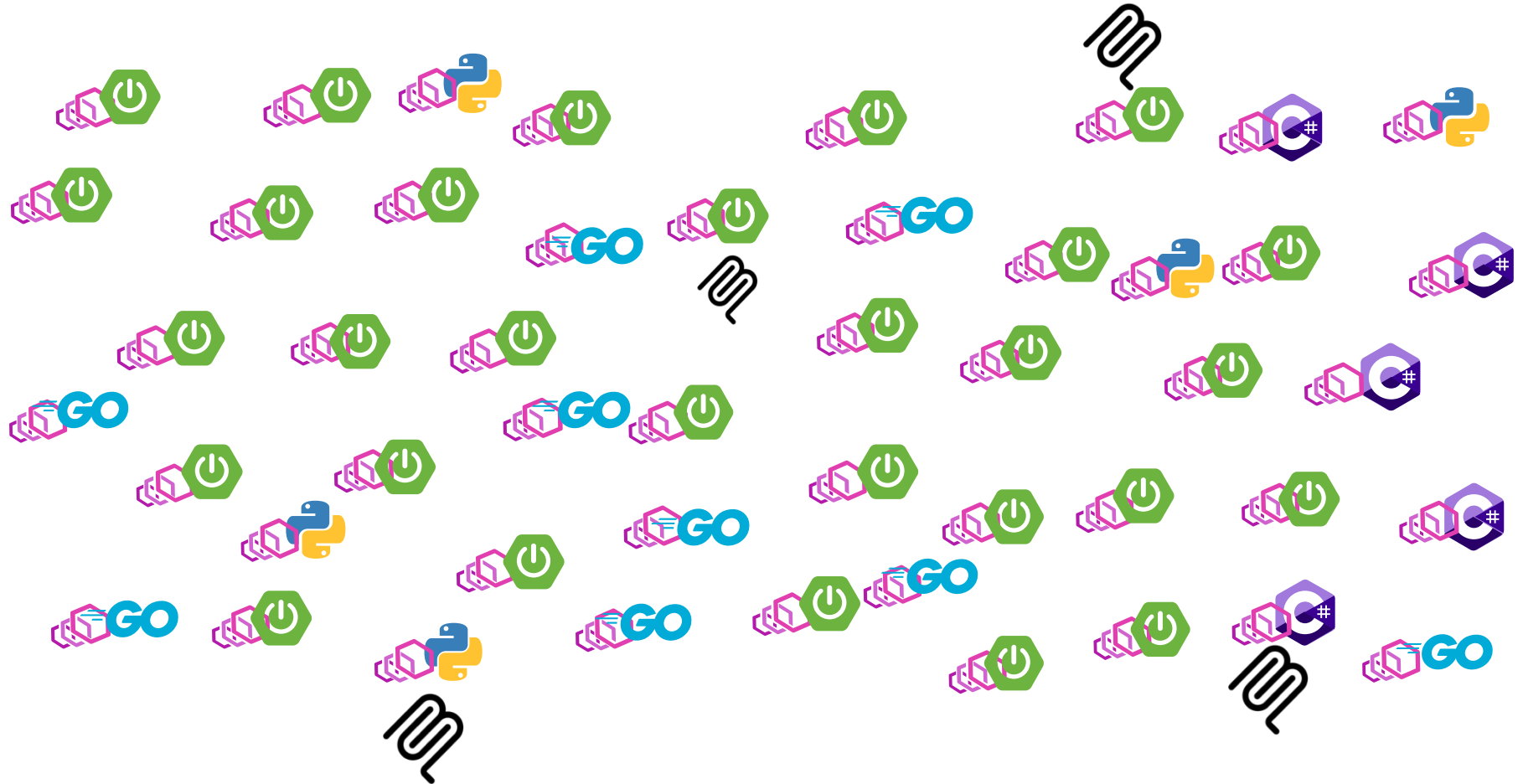
AI agent



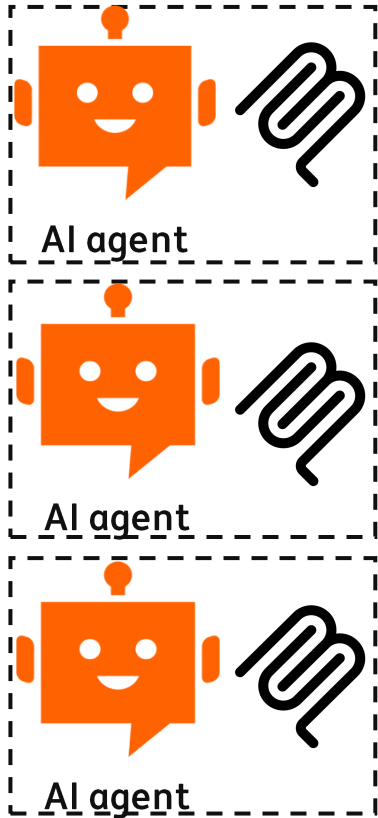
AI agent



AI agent



# Configure dedicated MCP servers with specific tools and bridge JSON-RPC to HTTP in one place



# Two options for connecting AI agents to an existing service mesh using MCP

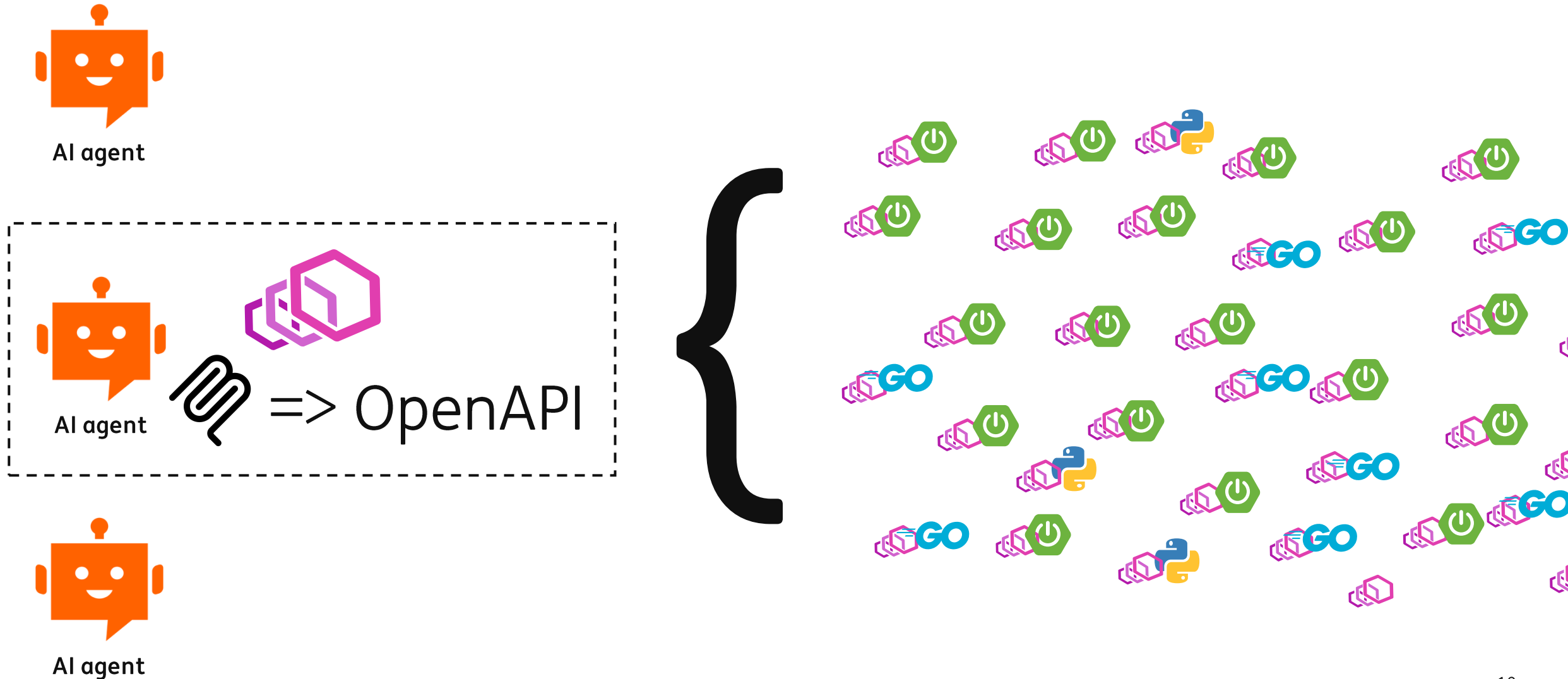
## Off-the-shelf framework(s)

- Dedicated library / server for specific business logic
- Use Envoy AI gateway (NOT-PLANNED [#1450](#))
- New tech stack with unknown failure modes
- Re-implement observability, resiliency and security (or sidecar)

## Extending Envoy

- Known technology
- Many extension points to choose from that can transform requests
- MCP support in-progress by community in open-source project

# Use Envoy to expose service mesh data to AI agents



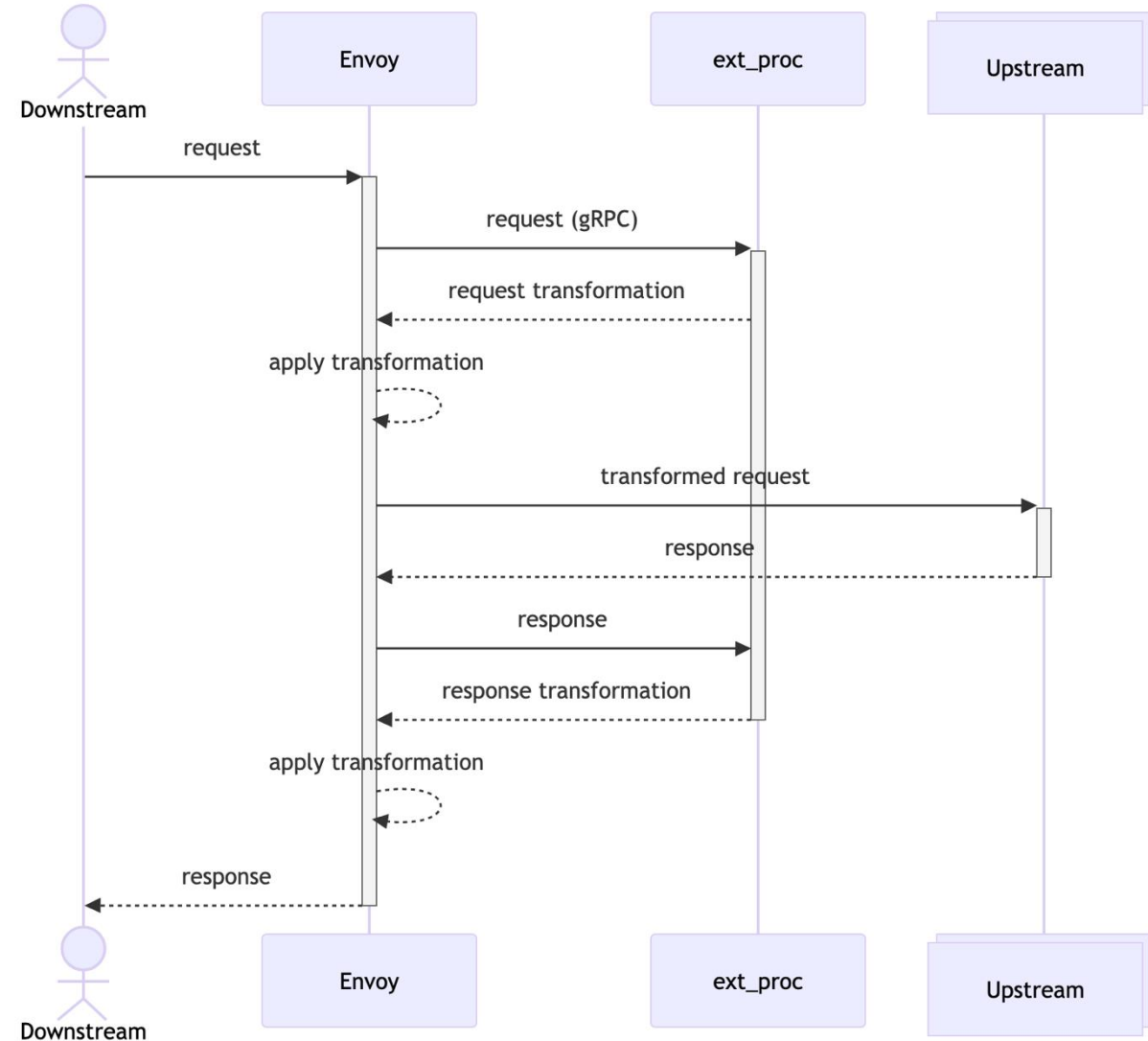
# Extending Envoy - Options to implement MCP

Where to configure, parse and convert JSON-RPC to HTTP requests?

- Implement logic in filter
  - C++, Go, WASM or Lua
  - Dynamic modules with Rust / Go SDK
- ext\_proc filter to offload processing to a separate service
  - Rich set of transformations for headers, body and trailers on both request as response
  - Implemented with a separate gRPC server

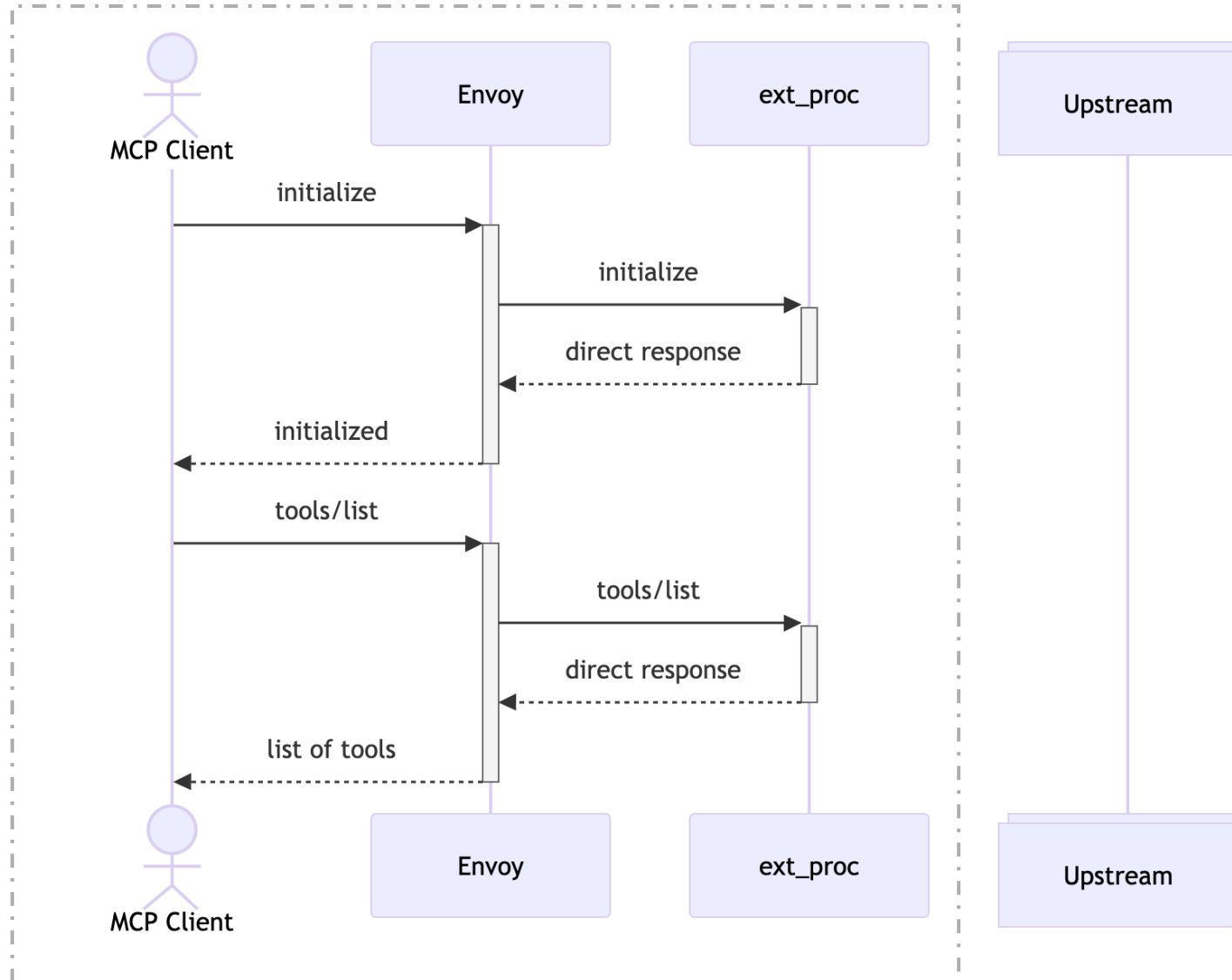
# External processing - a short overview

- **gRPC interface** to give hints to Envoy to transform request headers, body, trailers and response headers, body, trailers
  - Can send a direct response
  - Can access dynamic metadata
- Stable since release 1.29
- gRPC server has state per request / stream lifecycle making state-management easy to reason about
- Envoy's HttpClient interacts with ext\_proc server
- Stats come out of the box
- Traceparent support for OpenTelemetry observability
- Unix Domain Sockets for inter-process communication prevents unnecessary serialization in TCP stack



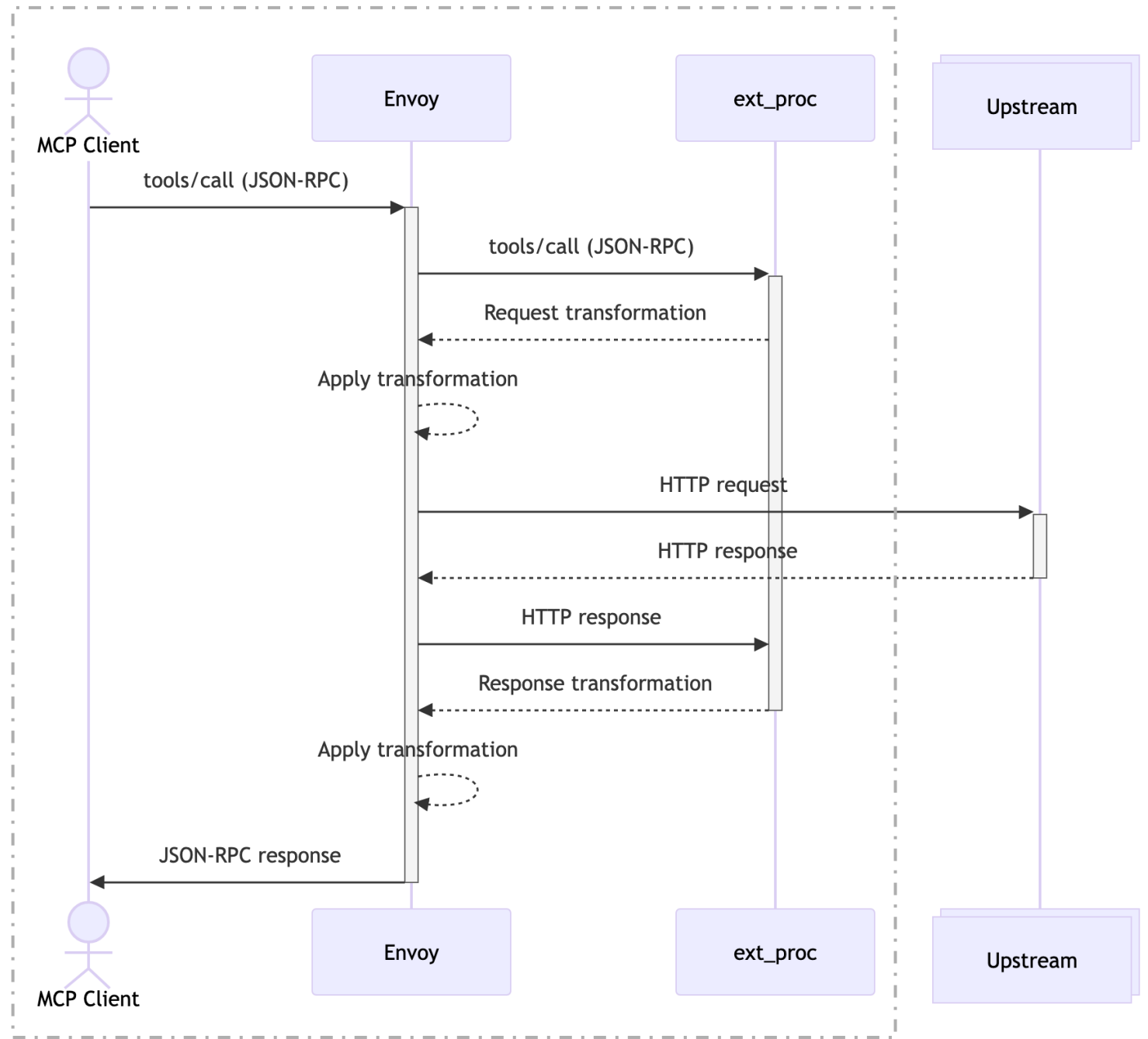
# Implementing MCP using ext\_proc (1)

ext\_proc handling direct responses



# Implementing MCP tool calls using ext\_proc (2)

Request / response transformations



# Transforming JSON-RPC requests to HTTP

## JSON-RPC (POST)

```
{  
  "id": 2,  
  "jsonrpc": "2.0",  
  "method": "tools/call",  
  "params": {  
    "name": "findPetsByStatus",  
    "arguments": {  
      "status": "available"  
    }  
  }  
}
```

## => HTTP (GET)

GET <https://petstore.swagger.io/v2/pet/findByStatus?status=available>  
HTTP/2.0

Host: petstore.swagger.io

# Transforming HTTP responses back to JSON-RPC

## JSON-RPC

```
{
  "id": 2,
  "jsonrpc": "2.0",
  "result": {
    "content": [{
      "text":
        "[{\"id\":0,\"category\":{\"id\":0,\"name\":\"string\"},\"name\": \"doggie\", \"status\": \"available\"}]",
      "type": "text"
    }],
    "structuredContent": [
      {
        "id": 0,
        "category": {
          "id": 0,
          "name": "string"
        }
      }
    ]
  }
}
```

## <= HTTP

```
[
  {
    "id": 0,
    "category": {
      "id": 0,
      "name": "string"
    },
    "name": "doggie",
    "status": "available"
  }
]
```

# Configuring Envoy to use an ext\_proc service

- name: envoy.filters.http.ext\_proc

typed\_config:

"@type": type.googleapis.com/envoy.extensions.filters.http.ext\_proc.v3.ExternalProcessor

grpc\_service:

envoy\_grpc:

cluster\_name: ext\_proc

mutation\_rules:

**allow\_all\_routing:**

**value: true**

processing\_mode:

request\_header\_mode: SEND

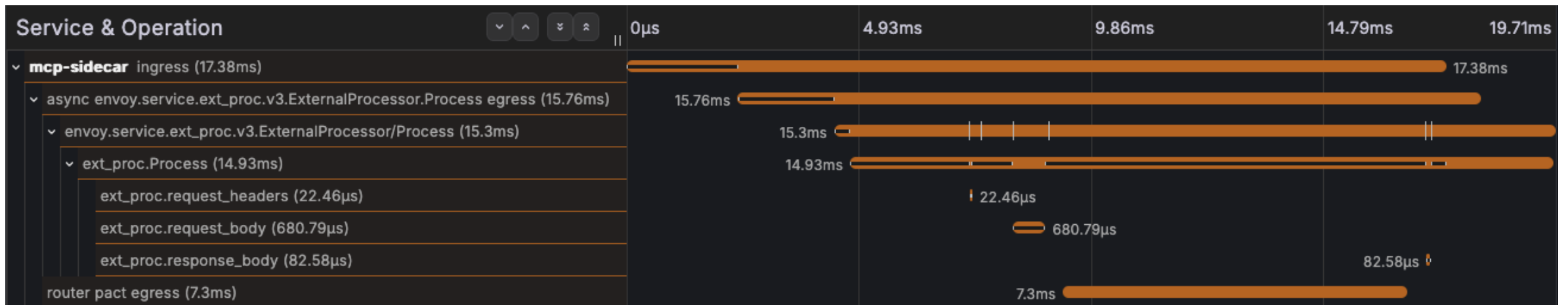
response\_header\_mode: SKIP

request\_body\_mode: BUFFERED

response\_body\_mode: BUFFERED

# Adding observability

Use `spawn_upstream_span: true` to visualize upstream processing time from `ext_proc`



# Envoy is ideal to implement an MCP server using ext\_proc for our use-case

- Extension point turned out ideal for our use-case
- Teams do not have to implement MCP for their HTTP endpoints
- Business logic can evolve faster than Envoy's backlog
  - Can be scaled down when Envoy's MCP feature set evolves
- Full integration with observability
- Proposal to **open-source** this to [github.com/ing-bank](https://github.com/ing-bank)

# Wrapping up

- AI agents can consume existing HTTP APIs via MCP tool calls and leveraging an existing service mesh does this in a secure, reliable and observable manner
- Envoy's extension points support use-cases not in the core feature-set of a service mesh proxy
- ext\_proc has a rich interface that lets you do *anything(!)* with requests/responses and fits our use-case
- This is running in production and supports several AI use-cases
- ING is open-sourcing this 'openapi mcp ext\_proc' server

Tune into our tech careers.



Find us at booth 799.