

# MCP POC – Factory Monitoring

## Overview

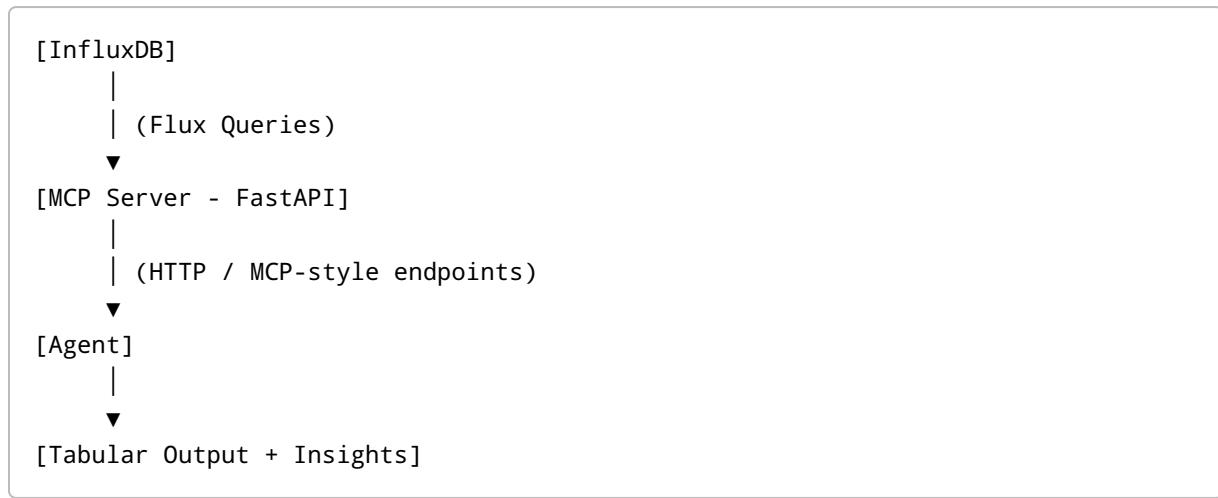
This repository contains a **Proof of Concept (POC)** demonstrating how **Model Context Protocol (MCP)** can be used to expose factory sensor data through a lightweight API and consume it via an agent for insights.

The POC focuses on:

- Exposing sensor statistics (temperature, PPFD, volts)
- Using a FastAPI-based MCP server
- Consuming MCP endpoints via a rule-based agent
- Following secure coding practices (no secrets committed to GitHub)

---

## Architecture



---

## Files in This Repository

File	Description
<code>mcp_poc.py</code>	MCP server implemented using FastAPI. Exposes sensor statistics endpoints.
<code>agent.py</code>	Agent that queries MCP endpoints and generates discussion-style insights.
<code>MCP_POC_Documentation.docx</code>	Detailed documentation explaining the POC design and flow.
<code>.gitignore</code>	Ensures secrets, caches, and unnecessary files are not committed.

## MCP Server (`mcp_poc.py`)

The MCP server: - Connects to **InfluxDB** - Queries sensor data using **Flux** - Calculates **min**, **max**, and **latest** values per topic - Exposes REST endpoints:

### Available Endpoints

Endpoint	Description
<code>/temperature</code>	Temperature statistics per sensor
<code>/ppfd</code>	PPFD (light intensity) statistics
<code>/volts</code>	Voltage statistics

## Agent (`agent.py`)

The agent: - Accepts natural language-like input from the user - Identifies sensor type and metric (min / max / latest) - Calls the appropriate MCP endpoint - Displays results in a table - Generates **human-readable insights** for discussion and analysis

Example questions: - "What is the max temperature?" - "Show latest volts" - "Lowest PPFD readings"

## Security & Configuration

### Secrets Handling

- No tokens or secrets are committed to GitHub
- InfluxDB token is read from an environment variable

Example:

```
export INFLUX_TOKEN=""
```

Or on Windows (PowerShell):

```
$env:INFLUX_TOKEN=""
```

## How to Run

### 1. Start the MCP Server

```
uvicorn mcp_poc:app --reload --port 5005
```

### 2. Run the Agent

```
python agent.py
```

---

## Purpose of This POC

This POC demonstrates:

- How MCP can decouple data access from consumers
- How agents can reason over MCP-exposed tools
- A secure, production-aware approach to API and agent design

It is intended for **learning, experimentation, and architectural validation**, not as a production deployment.

---

## Future Enhancements

- Replace rule-based agent with LLM-based reasoning
  - Add authentication and authorization
  - Integrate visualization tools (e.g., Grafana API)
  - Extend MCP tools for additional sensor types
- 

## Author

Kavita Khan  
MCP POC – Factory Monitoring