Zero to Production: Building Secure, Scalable MCP Servers and AI Agents with Open-Source Templates

Red Hat
Data and AI Team

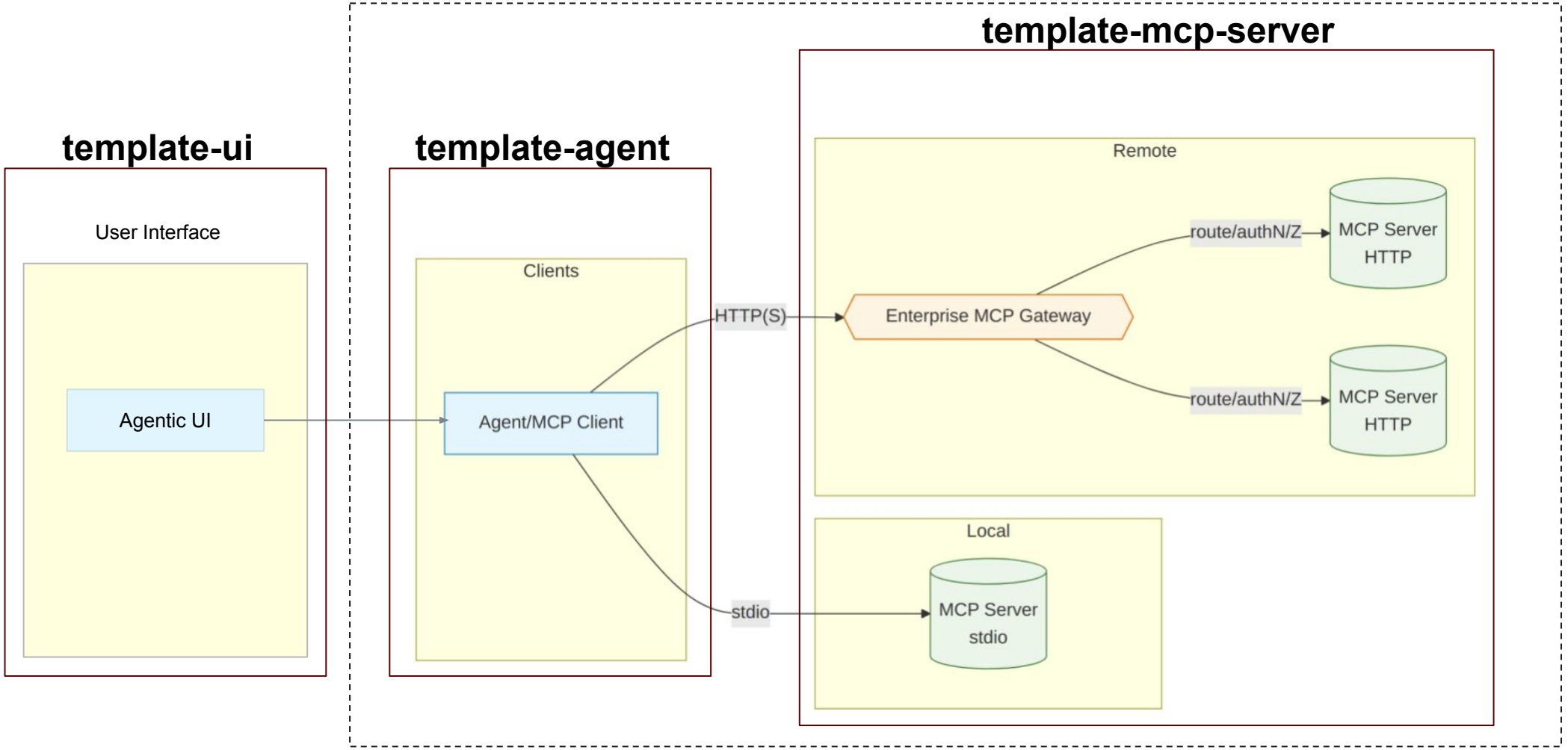
Workshop Overview

- **MCP Server**
  FastMCP based Model Context Protocol server
- **Agent**
  LangGraph-based AI agent with streaming
- **UI Application**
  React + Fastify chat interface

By the end, you'll have:

- A working MCP server with custom tools
- An AI agent that uses your MCP server
- A production-ready web UI for user interaction

Architecture & Deployment (IBM + Anthropic)



https://www.ibm.com/downloads/documents/us-en/1443d5dd174f42e6
https://newsroom.ibm.com/2025-10-07-2025-ibm-and-anthropic-partner-to-advance-enterprise-software-development-with-proven-security-and-governance

Template MCP Server

What is an MCP Server?

Model Context Protocol (MCP) server provides **tools, prompts** and **resources** that AI agents can use.

Key Features in the template MCP:

- FastAPI-based with multiple transport protocols (HTTP/Streamable-HTTP, SSE)
- Modular tool system for easy extension
- Comprehensive testing and container support
- OAuth integration and PostgreSQL storage

MCP Server - Quick Start

```
# Clone the template
git clone https://github.com/redhat-data-and-ai/template-mcp-server.git
cd template-mcp-server

# One command setup - starts everything!
make local
```

Test it works

```
curl http://localhost:5001/health
```

MCP Server - Adding Custom Tools

Add a new tool file

```
# template_mcp_server/src/tools/my_tool.py

def get_sales_data(territory: str, quarter: str) -> Dict[str, Any]:
    """Retrieve sales data for a specific territory and quarter."""
    ... business logic here
```

Register the new tool

```
# template_mcp_server/src/mcp.py
from template_mcp_server.src.tools.my_tool import get_sales_data

def _register_mcp_tools(self) -> None:
    """Register all MCP tools."""
    self.mcp.tool()(get_sales_data)  # ← Add your tool here
```

Template Agent

What is the Template Agent?

A production-ready **LangGraph-based AI agent** with streaming capabilities.

Key Features in the template Agent:

- Real-time streaming responses (SSE)
- Thread persistence with PostgreSQL
- Enterprise-ready (health checks, logging, error handling)
- Langfuse tracing for observability

Template Agent - Quick Start

```
# Clone the template
git clone https://github.com/redhat-data-and-ai/template-agent.git
cd template-agent

# One command setup - starts everything!
make local
```

Test it works

```
curl http://localhost:5002/health
```

Template Agent - Streaming API

```
curl -X POST "http://localhost:5002/v1/stream" \
  -H "Content-Type: application/json" \
  -d '{
    "message": "What is 25 times 42?",
    "thread_id": "thread_123",
    "user_id": "user_456",
    "stream_tokens": true
  }'
```

Sample Response

```
{"type": "message", "content": {"type": "ai", "content": "", "tool_calls": [...]}}
{"type": "token", "content": "The"}
{"type": "token", "content": " answer"}
{"type": "token", "content": " is"}
{"type": "token", "content": " 1050"}
{"type": "message", "content": {"type": "ai", "content": "The answer is 1050"}}
[DONE]
```

Template Agent - Integration with MCP

```
curl -X POST "http://localhost:5002/v1/stream" \
  -H "Content-Type: application/json" \
  -d '{
    "message": "What is 25 times 42?",
    "thread_id": "thread_123",
    "user_id": "user_456",
    "stream_tokens": true
  }'
```

Sample Response

```
{"type": "message", "content": {"type": "ai", "content": "", "tool_calls": [...]}}
{"type": "token", "content": "The"}
{"type": "token", "content": " answer"}
{"type": "token", "content": " is"}
{"type": "token", "content": " 1050"}
{"type": "message", "content": {"type": "ai", "content": "The answer is 1050"}}
[DONE]
```

Template UI

What is the Template UI?

A modern **React + Fastify** chat interface for your AI agent.

Tech Stack and Key Features:

- React with Typescript.
- Fastify (Node v22) web server
- OAuth2/SSO Authentication
- Tailwind CSS v4 for styling
- Real Time streaming support

Template Agent - Quick Start

```
# Clone the template
git clone https://github.com/redhat-data-and-ai/template-ui.git
cd template-ui

# One command setup - starts everything!
make local
```

Test it works

```
curl http://localhost:5173/
```

Production Considerations

Security

- Enable OAuth/SSO authentication in UI
- Configure SSL/TLS certificates
- Use secure cookie signing
- Implement rate limiting
- Add input validation

Scalability

- Use managed PostgreSQL service
- Configure horizontal pod autoscaling
- Set up load balancing
- Implement caching strategies

Monitoring

- Set up Langfuse for agent tracing
- Configure logging aggregation
- Add health check monitoring
- Set up alerting

Best Practices

MCP Server

- Keep tools focused and single-purpose
- Include comprehensive error handling
- Add detailed logging
- Write tests for each tool
- Document tool parameters clearly

Agent

- Use streaming for better UX
- Implement conversation persistence
- Add tracing for debugging
- Configure appropriate timeouts
- Handle tool failures gracefully

UI

- Implement proper loading states
- Show streaming responses
- Handle authentication errors
- Add error boundaries
- Optimize bundle size

Contributions are Welcome!!

- MCP Server

  https://github.com/redhat-data-and-ai/template-mcp-server

- Agent

  https://github.com/redhat-data-and-ai/template-agent

- UI Application

  https://github.com/redhat-data-and-ai/template-ui

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


x.com/RedHat

Red Hat
Data and AI Team