Outlook

(No subject)

From Bahl, Vivek (Cognizant) <Vivek.Bahl@cognizant.com>
Date Mon 8/11/2025 10:46 AM
To Bahl, Vivek (Cognizant) <Vivek.Bahl@cognizant.com>

# MCP Local Cluster Project - Newcomer's Guide

[[memory:5725148]] [[memory:5724900]] [[memory:5724841]]

## 🎯 What This Project Does

This project creates a **secure, local development environment** for building AI applications that need to interact with external systems safely. Think of it as a **"safe sandbox"** where your AI agents can access files, databases, git repositories, and web APIs without risking your actual system.

### **The Problem It Solves**
Imagine you want to build an AI assistant that can:
- Read and write files on your computer
- Query databases for information
- Make git commits and push code
- Call external APIs like GitHub or Slack

Without this project: Your AI would have direct, unrestricted access to everything - potentially dangerous!

With this project: Your AI goes through a secure gateway that monitors, controls, and protects every interaction.

## 🏗️ What You Get

### **1. Secure AI Gateway**
- Acts like a security guard for your AI
- Blocks malicious requests automatically
- Logs everything for audit trails
- Rate limits to prevent abuse

### **2. Pre-built AI Tools**
- **File Operations**: Read/write files safely in designated folders
- **Database Access**: Query PostgreSQL with SQL injection protection
- **Git Operations**: Commit, push, pull with safety controls
- **Web APIs**: Call external services with domain restrictions

3. Visual Testing Interface
- Web-based tool to test your AI integrations
- See exactly what tools are available

- Debug connections and messages
- Monitor security in real-time

4. Production-Ready Security
- Container isolation (each service runs separately)
- Threat detection (blocks malicious AI behavior)
- Secrets management (API keys stored securely)
- Comprehensive logging (audit trail of all actions)

## 🚀 Steps to Use This for AI App Development

Phase 1: Setup (One-time)

#### Step 1: Install Prerequisites
```powershell
# You need:
# - Docker Desktop (free download)
# - Windows 10/11 with PowerShell (built-in)
# - 4GB+ RAM, 2GB+ disk space
```

#### Step 2: Get the Project
```powershell
# Navigate to the project directory
cd C:\vbahl\MCPSetup

# Check if everything is ready
.\pre-setup-check.ps1
```

#### Step 3: Configure Environment
```powershell
# Set up configuration files and directories
.\setup-environment.ps1

# Edit .env file with your API keys (optional for basic testing)
notepad .env
```

#### Step 4: Launch the Cluster
```powershell
# Build and start all components (takes 5-10 minutes first time)
.\start-cluster.ps1 -Build

# Verify everything is working
.\verify-cluster.ps1
```

Phase 2: Testing & Learning

#### Step 5: Explore with MCP Inspector

```
1. Open http://localhost:5173 in your browser
2. Connect to ws://localhost:8811
3. Click "Initialize" to establish connection
4. Browse available tools and test them
```

Example Tools You'll See:
- `filesystem.read_file` - Read files from workspace
- `database.query` - Run SQL queries safely
- `git.status` - Check git repository status
- `web.http_get` - Make HTTP requests to approved domains

#### Step 6: Test Basic Operations
```javascript
// In the MCP Inspector, try these:

// Read a file safely
{
  "method": "tools/call",
  "params": {
    "name": "filesystem.read_file",
    "arguments": { "path": "/workspace/sample-data.txt" }
  }
}

// Query database
{
  "method": "tools/call",
  "params": {
    "name": "database.query",
    "arguments": {
      "sql": "SELECT * FROM tasks WHERE status = ?",
      "params": ["pending"]
    }
  }
}
```

Phase 3: AI Application Development

#### Step 7: Connect Your AI Application

For Python AI Apps:
```python
import websocket
import json

# Connect to MCP Gateway
ws = websocket.create_connection("ws://localhost:8811")
```

```
# Initialize connection
init_message = {
    "jsonrpc": "2.0",
    "id": 1,
    "method": "initialize",
    "params": {
        "capabilities": {
            "tools": {"listChanged": True},
            "resources": {"subscribe": True}
        }
    }
}
ws.send(json.dumps(init_message))
response = json.loads(ws.recv())
print("Connected:", response)
```

For Node.js AI Apps:
```javascript
const WebSocket = require('ws');

const ws = new WebSocket('ws://localhost:8811');

ws.on('open', () => {
    // Initialize MCP connection
    ws.send(JSON.stringify({
        jsonrpc: '2.0',
        id: 1,
        method: 'initialize',
        params: {
            capabilities: {
                tools: { listChanged: true },
                resources: { subscribe: true }
            }
        }
    }));
});

ws.on('message', (data) => {
    const message = JSON.parse(data);
    console.log('Received:', message);
});
```

#### Step 8: Build AI Applications

Example: AI Code Assistant
```python
# Your AI can now safely:

# 1. Read project files
```

```
file_content = call_mcp_tool("filesystem.read_file", {
    "path": "/workspace/src/main.py"
})

# 2. Check git status
git_status = call_mcp_tool("git.status", {})

# 3. Query project database
issues = call_mcp_tool("database.query", {
    "sql": "SELECT * FROM issues WHERE status = 'open'"
})

# 4. Call GitHub API
github_data = call_mcp_tool("web.http_get", {
    "url": "https://api.github.com/repos/owner/repo/issues"
})

# AI processes this data and suggests code improvements
```

Example: Data Analysis Agent
```python
# AI can safely analyze business data:

# 1. Read CSV files
sales_data = call_mcp_tool("filesystem.read_file", {
    "path": "/workspace/data/sales.csv"
})

# 2. Query database for trends
trends = call_mcp_tool("database.query", {
    "sql": "SELECT month, SUM(revenue) FROM sales GROUP BY month"
})

# 3. Generate reports
report = ai_analyze(sales_data, trends)

# 4. Save results safely
call_mcp_tool("filesystem.write_file", {
    "path": "/workspace/reports/analysis.md",
    "content": report
})
```

### **Phase 4: Advanced Development**

#### Step 9: Add Custom Tools
```yaml
# Edit config/gateway.yaml to add new services
servers:
  - name: my-custom-service
```

    url: http://my-service:3000
    type: custom
    security:
      allowedOperations: ["read", "analyze"]
      maxRequestSize: 1048576

Step 10: Production Deployment
```powershell
# For production use:
# 1. Change all default passwords in .env
# 2. Add real API keys to secrets/ directory
# 3. Configure proper SSL certificates
# 4. Set up monitoring and alerting
# 5. Review security settings in config/gateway.yaml
```

## 🛡️ Built-in Safety Features

Automatic Threat Detection
- **Tool Poisoning**: Blocks malicious tool descriptions
- **MCP Rug Pull**: Prevents tools from changing behavior after approval
- **MCP Shadowing**: Detects conflicting or duplicate tools

### **Access Controls**
- **File System**: Only access `/workspace` directory
- **Database**: SQL injection prevention and query limits
- **Web APIs**: Domain allowlisting and rate limiting
- **Git**: Dangerous commands blocked (`rm -rf`, etc.)

### **Audit & Monitoring**
- Every AI action is logged to database
- Real-time security monitoring
- Performance metrics and health checks
- Complete audit trail for compliance

## 🎓 Learning Path for Newcomers

### **Week 1: Setup & Basics**
1. Install and run the cluster
2. Explore MCP Inspector interface
3. Test each tool type (filesystem, database, git, web)
4. Read the security logs to understand what's happening

### **Week 2: Simple AI Integration**
1. Connect a basic AI client (Python/Node.js)
2. Build a simple file-reading AI assistant
3. Create an AI that queries the sample database
4. Experiment with git operations

### **Week 3: Real Applications**

1. Build a code review AI using git + filesystem tools
2. Create a data analysis AI using database + web tools
3. Develop a documentation AI that reads/writes files
4. Add custom business logic

Week 4: Production Readiness
1. Add proper API keys and secrets
2. Configure security policies for your use case
3. Set up monitoring and alerting
4. Deploy to a server environment

## 💡 Key Benefits for AI Developers

1. **Safety First**: Your AI can't accidentally harm your system
2. **Standardized**: Works with any AI framework (OpenAI, Anthropic, local models)
3. **Observable**: See exactly what your AI is doing at all times
4. **Scalable**: Add new tools and services easily
5. **Production-Ready**: Enterprise security from day one

## 🔄 Typical Development Workflow

```
1. Start cluster: .\start-cluster.ps1
2. Test with Inspector: http://localhost:5173
3. Code your AI app: Connect to ws://localhost:8811
4. Test integration: Use MCP tools in your app
5. Monitor security: Check logs and health endpoints
6. Deploy safely: Production-ready security included
```

This project transforms AI development from **"hope it doesn't break anything"** to **"confidently build powerful AI applications"** with enterprise-grade security and monitoring built-in from day one.