MCP-Vacuum: Autodiscovery Agent Architecture

MCP-Vacuum: Autodiscovery Agent Architecture

Project Overview

MCP-Vacuum is a Google Python ADK agent that automatically discovers MCP (Model Context Protocol) servers on the local network, authenticates using modern OAuth 2.1 flows, and converts configurations to Kagent-compatible format.

🎯 Project Goals

Primary Objectives

- **Automatic Discovery**: Detect MCP servers using multiple network protocols (mDNS, SSDP, ARP)
- **Secure Authentication**: Implement OAuth 2.1 + PKCE with dynamic client registration
- **Schema Conversion**: Bidirectional mapping between MCP and Kagent formats
- **Production Deployment**: Google Cloud integration with Vertex AI Agent Engine
- **DevOps Integration**: CI/CD pipelines, monitoring, and observability

Technical Requirements

- Python 3.12+ with modern asyncio patterns
- Google ADK v1.0+ with hierarchical agent architecture
- Type-safe implementation following PEP 8 guidelines
- Comprehensive testing with 95%+ coverage
- Production-ready security and performance

🏗️ System Architecture

Component Overview

```
┌─────────────────────────────────────────────────────────────────┐
│                    MCP-Vacuum Agent System                      │
├─────────────────────────────────────────────────────────────────┤
│  Orchestration Agent (Coordinator)                             │
├────────────────┬────────────────┬────────────────┬───────────┤
│ Discovery Agent │ Authentication │ Conversion      │ MCP Client│
│                 │ Agent          │ Agent           │ Agent     │
├────────────────┼────────────────┼────────────────┼───────────┤
│ • mDNS/DNS-SD   │ • OAuth 2.1    │ • Schema        │ • JSON-RPC│
│ • SSDP/UPnP     │ • PKCE         │   Validation    │   2.0     │
│ • ARP Scanning  │ • Dynamic      │ • Bidirectional │ • Multiple│
│ • Filtering     │   Registration │   Mapping       │   Transpts│
└────────────────┴────────────────┴────────────────┴───────────┘
```

Agent Hierarchy

OrchestrationAgent (Parent)

- Coordinates workflow execution
- Manages agent lifecycle and state
- Handles error recovery and circuit breaking

DiscoveryAgent (Child)

- Multi-protocol network scanning
- Service classification and filtering
- Connection pooling and result caching

AuthenticationAgent (Child)

- OAuth 2.1 + PKCE implementation
- Dynamic client registration (RFC 7591)
- Credential management and rotation

ConversionAgent (Child)

- MCP ↔ Kagent schema transformation
- Validation pipeline with error reporting
- Metadata preservation and semantic integrity

MCPClientAgent (Child)

- JSON-RPC 2.0 protocol implementation
- Multi-transport support (STDIO, SSE, HTTP)
- Tool enumeration and invocation

🔧 Technical Stack

Core Technologies

- **Python 3.12+**: Modern language features and performance
- **Google ADK v1.0+**: Agent framework with Vertex AI integration
- **Pydantic V2**: Type-safe data validation and serialization
- **AsyncIO**: Concurrent network operations and agent communication
- **UV Package Manager**: Fast dependency resolution and virtual environments

Integration Points

- **Vertex AI Agent Engine**: Managed runtime with auto-scaling
- **Google Cloud Storage**: Configuration caching and persistence
- **BigQuery**: Discovery logs and analytics
- **Cloud Monitoring**: Observability and alerting

Development Tools

- **pytest + pytest-asyncio**: Comprehensive testing framework
- **ruff + black**: Code formatting and linting
- **mypy**: Static type checking
- **codecov**: Coverage tracking and reporting

📋 Project Status

Current Phase: Foundation & Planning

- ✅ Project architecture defined
- ✅ Task breakdown completed (22 tasks across 4 priority levels)
- ✅ Jira project setup with Epic and core tasks

- ✅ Research documentation compiled
- 🔄 Foundation implementation in progress

Priority Breakdown

- **P0 (Critical)**: 5 tasks - Core functionality foundation
- **P1 (High)**: 8 tasks - Essential features and testing
- **P2 (Medium)**: 7 tasks - Enhanced capabilities and deployment
- **P3 (Low)**: 2 tasks - Documentation and examples

🚀 Getting Started

Prerequisites

- Python 3.12+
- Google Cloud Account with ADK access
- UV package manager
- Git and Docker

Quick Setup

```
# Clone and setup project
git clone https://github.com/your-org/mcp-vacuum.git
cd mcp-vacuum

# Install dependencies with UV
uv sync --dev

# Run tests
uv run pytest

# Start development server
uv run python -m mcp_vacuum.cli discover --help
```

📚 Documentation Structure

Technical Documentation

- **Architecture Overview** - Detailed system design
- **API Reference** - Component interfaces and protocols
- **Development Guide** - Setup and contribution guidelines
- **Deployment Guide** - Production deployment patterns

Research & Background

- **Research Documentation** - Technology analysis and decisions
- **MCP Protocol Analysis** - Protocol implementation details
- **Google ADK Integration** - Framework usage patterns

🔗 Related Links

- **Jira Project**: MV - mcp-vacuum
- **Epic Ticket**: MV-1 - MCP Server Autodiscovery & Kagent Integration Agent
- **GitHub Repository**: (TBD)

- **Google ADK Docs**: https://google.github.io/adk-docs/
- **MCP Specification**: https://modelcontextprotocol.io/specification/

📞 Contact & Support

For questions, issues, or contributions:

- **Project Lead**: DevOps Engineer
- **Jira Issues**: Create tickets in the MV project
- **Documentation**: Update pages in this Confluence space

Last Updated: June 24, 2025
Version: 1.0.0