Google ADK Integration Guide

Google ADK Integration Guide

Overview

Implementation guide for integrating Google's Python Agent Development Kit (ADK) v1.0+ with MCP Vacuum for production-ready agent deployment.

Agent Architecture

Base Agent Implementation

```python
from google.adk import BaseAgent
from typing import Protocol

class MCPDiscoveryAgent(BaseAgent):
    """Production MCP discovery agent with ADK integration."""

    def __init__(self) -> None:
        super().__init__()
        self.discovery_service = DiscoveryService()
        self.auth_manager = AuthManager()
        self.converter = SchemaConverter()

    async def execute_discovery(self) -> list[KagentSchema]:
        """Main discovery workflow."""
        servers = await self.discovery_service.scan_network()
        authenticated = await self.auth_manager.authenticate_batch(servers)
        return await self.converter.convert_to_kagent(authenticated)
```

Hierarchical Agent Composition

```python
class OrchestrationAgent(BaseAgent):
    """Parent agent coordinating specialized child agents."""

    def __init__(self) -> None:
        super().__init__()
        self.discovery_agent = DiscoveryAgent()
        self.auth_agent = AuthenticationAgent()
        self.conversion_agent = ConversionAgent()

    async def coordinate_workflow(self) -> WorkflowResult:
        """Coordinate multi-agent workflow execution."""
        discovery_result = await self.discovery_agent.discover()
        auth_result = await self.auth_agent.authenticate(discovery_result)
        return await self.conversion_agent.convert(auth_result)
```

Vertex AI Integration

Production Configuration

```
vertex_config = {
```

```
    "project_id": "your-gcp-project",
    "location": "us-central1",
    "agent_runtime": "python-3.12",
    "scaling": {
        "min_instances": 1,
        "max_instances": 10,
        "target_cpu_utilization": 70
    }
}
```

Event-Driven Communication

```python
class AgentEventBus:
    """Type-safe inter-agent communication."""

    def __init__(self) -> None:
        self.subscribers: dict[str, list[Callable]] = {}

    async def publish(self, event: AgentEvent) -> None:
        """Publish event to subscribers."""
        handlers = self.subscribers.get(event.type, [])
        await asyncio.gather(*[handler(event) for handler in handlers])
```

Next Steps

1. Setup ADK Environment
2. Implement Base Agents
3. Add Monitoring
4. Deploy to Vertex AI