Bước 2 (MCP) — Agent dùng MCP server on-prem

Mục tiêu: tách **năng lực nghiệp vụ** (tra cứu catalog, tính phí ship, tồn kho...) ra thành **MCP server** chuẩn hoá (protocol-based), và để **ADK agent** tiêu thụ các năng lực đó như **MCP tools**. Sau bước này bạn sẽ có:

- 1 **MCP server (Python)** expose các tool `get_product`, `calc_shipping`, (tuỳ chọn) `reserve_stock`.
- 1 **ADK agent** (Python) đóng vai **MCP client**, kết nối server qua **STDIO** (local) hoặc **SSE/HTTP** (sidecar/remote).
- Quy trình chạy/dev/test on-prem, sẵn nền cho Bước 3 (A2A) & Bước 4 (AP2).

---

1) Kiến trúc ngắn gọn

```
[User] → ADK Agent (LLM + policy)
              │
              │  (MCPToolset: list_tools / call_tool)
              ▼
         MCP Server (Python)
            ├─ get_product(query)
            ├─ calc_shipping(dest, weight)
            └─ reserve_stock(sku, qty)   # tuỳ chọn demo giao dịch
```

- **Server** đóng gói business logic, có thể thay thế/scale/hardening độc lập. - **Agent** chỉ giữ policy hội thoại và quyết định **khi nào** gọi tool.

---

2) Cấu trúc dự án

```
adk-mcp-step2/
├─ mcp_server/
│  ├─ __init__.py
│  └─ server.py             # MCP server (FastMCP)
└─ shop_agent_mcp/
   ├─ __init__.py
   └─ agent.py              # ADK agent dùng MCPToolset
```

---

3) Cài đặt

```
python -m venv .venv && source .venv/bin/activate
pip install google-adk "mcp[cli]" httpx python-dotenv
```

> *Bạn có thể giữ lại môi trường từ Bước 1. Nếu muốn on-prem tuyệt đối, tiếp tục dùng LiteLLM+vLLM/Ollama cho model như ở Bước 1.*

---

4) MCP Server (Python) — `mcp_server/server.py`

```python
# mcp_server/server.py
from typing import Any
from mcp.server.fastmcp import FastMCP

mcp = FastMCP("shop_mcp")

# --- dữ liệu giả lập (on-prem) ---
CATALOG = {
    "SKU001": {"sku": "SKU001", "name": "Anime Figure A", "price": 29.9,
"currency": "USD", "stock": 7},
    "SKU002": {"sku": "SKU002", "name": "Manga Volume 1", "price": 12.5,
"currency": "USD", "stock": 21},
}

@mcp.tool()
async def get_product(query: str) -> dict[str, Any]:
    """Tra cứu sản phẩm theo SKU hoặc tên (substring).
    Args:
      query: SKU hoặc tên gần đúng.
    Returns:
      {status: 'success'|'error', product?: {...}, error_message?: str}
    """
    q = query.strip().lower()
    # match SKU
    for k, v in CATALOG.items():
        if k.lower() == q:
            return {"status": "success", "product": v}
    # match name substring
    for p in CATALOG.values():
        if q in p["name"].lower():
            return {"status": "success", "product": p}
    return {"status": "error", "error_message": f"No product matches '{query}'"}
```

```
@mcp.tool()
async def calc_shipping(destination: str, weight_kg: float) -> dict[str, Any]:
    """Tính phí ship đơn giản: base 2.0 + zone + 0.8*weight_kg.
    Args:
      destination: thành phố (hanoi/hochiminh/khác)
      weight_kg: khối lượng
    """
    zone_fee = 0.0 if destination.lower() in ["hanoi", "hochiminh", "saigon"]
else 1.5
    fee = round(2.0 + zone_fee + 0.8 * float(weight_kg), 2)
    return {"status": "success", "shipping_fee": fee, "currency": "USD"}

@mcp.tool()
async def reserve_stock(sku: str, qty: int) -> dict[str, Any]:
    """Giảm tồn kho (demo), trả về tồn kho mới. Dùng cho kịch bản giao dịch.
    Args:
      sku: mã sản phẩm
      qty: số lượng cần giữ
    """
    item = CATALOG.get(sku)
    if not item:
        return {"status": "error", "error_message": f"SKU '{sku}' not found"}
    if qty <= 0:
        return {"status": "error", "error_message": "qty must be > 0"}
    if item["stock"] < qty:
        return {"status": "error", "error_message": "insufficient stock"}
    item["stock"] -= qty
    return {"status": "success", "sku": sku, "reserved": qty, "stock":
item["stock"]}

if __name__ == "__main__":
    # STDIO mode: server được host (spawn) bởi client/agent
    mcp.run(transport="stdio")
```

Ghi chú: - Dùng `FastMCP` + decorator `@mcp.tool()` để auto sinh schema cho tools. - Server **không tự listen HTTP** ở ví dụ này; nó chờ được **spawn qua STDIO** bởi "host" (ADK agent của bạn).

---

5) ADK Agent dùng MCPToolset — `shop_agent_mcp/agent.py`

```
# shop_agent_mcp/agent.py
import os
from google.adk.agents import LlmAgent
from google.adk.models.lite_llm import LiteLlm   # on-prem (tuỳ chọn)
from google.adk.tools.mcp_tool.mcp_toolset import MCPToolset
```

```
from google.adk.tools.mcp_tool.mcp_session_manager import StdioConnectionParams,
SseConnectionParams
from mcp import StdioServerParameters

# --- Model cấu hình ---
# Option 1: Dùng Gemini (như Bước 1)
# model_name = "gemini-2.0-flash"
# model = model_name

# Option 2: On-prem tuyệt đối (vLLM/Ollama qua LiteLLM)
model = LiteLlm(
    model="google/gemma-2-9b-it",      # đổi theo model bạn đang serve
    api_base="http://localhost:8000/v1" # endpoint OpenAI-compatible tự host
)

# --- MCPToolset: STDIO pattern (spawn server process local) ---
mcp_tools = MCPToolset(
    connection_params=StdioConnectionParams(
        server_params=StdioServerParameters(
            command="python",
            args=[os.path.join(os.path.dirname(__file__), "..", "mcp_server",
"server.py")]
        )
    ),
    # tool_filter=["get_product", "calc_shipping"]  # nếu muốn chỉ expose 1 phần
)

root_agent = LlmAgent(
    model=model,
    name="shop_agent_mcp",
    instruction=(
        "You are a shopping assistant. Use MCP tools for product lookup,
shipping fee, and stock reservation. "
        "If a query mentions product, price, shipping, stock — prefer MCP
tools."
    ),
    tools=[mcp_tools]
)

# --- Tuỳ chọn: kết nối MCP remote (SSE/HTTP) thay vì STDIO ---
# mcp_tools = MCPToolset(
#     connection_params=SseConnectionParams(url="http://mcp-server:8081")
# )
```

Ý tưởng: - **STDIO**: nhanh nhất cho dev cục bộ — ADK spawn process `python server.py` và nối STDIO. - **SSE/HTTP**: dùng khi bạn muốn **sidecar**/remote (K8s) — để agent gọi qua mạng.

---

6) Chạy & kiểm thử

1) **Dev UI**

```
adk web
# Mở http://127.0.0.1:8000 → chọn "shop_agent_mcp"
```

- Vào **Events/Trace**: gõ “giá ‘Anime Figure’?” ⇒ thấy **call_tool → mcp:get_product**. - Gõ “tính ship về Hanoi nặng 1.5kg” ⇒ **mcp:calc_shipping**. - Gõ “giữ 2 cuốn SKU002” ⇒ **mcp:reserve_stock** (stock giảm).

2) **CLI** (tuỳ cách bạn tổ chức project)

```
adk run shop_agent_mcp
```

3) **API server** (chuẩn bị nối Django/Ingress)

```
adk api_server
# Gửi POST /v1/chat với {"agent": "shop_agent_mcp", "message": "Tra giá SKU001"}
```

---

7) Test nhanh (logic thuần Python, không phụ thuộc LLM)

Tạo `tests/test_mcp_tools.py` và dùng **mcp client** tối giản hoặc... test trực tiếp hàm trong server.

Cách đơn giản — import trực tiếp và gọi:

```
# tests/test_mcp_tools.py
import asyncio
from mcp_server.server import get_product, calc_shipping, reserve_stock

async def test_get_product_sku():
    out = await get_product("SKU001")
    assert out["status"] == "success" and out["product"]["sku"] == "SKU001"

async def test_calc_shipping_ok():
    out = await calc_shipping("Hanoi", 1.2)
    assert out["status"] == "success" and out["shipping_fee"] > 0

async def test_reserve_stock_fail():
    out = await reserve_stock("SKU001", 999)
    assert out["status"] == "error"
```

```
# pytest sẽ tự chạy event loop với pytest-asyncio, hoặc:
# pip install pytest pytest-asyncio
```

---

8) Definition of Done (PASS Bước 2)

- [ ] **ADK ↔ MCP hoạt động**: nhập câu hỏi trong Dev UI, agent **gọi MCP tools** (thấy rõ trong Events/Trace).
- [ ] **Tách biệt nghiệp vụ**: code nghiệp vụ chính nằm ở MCP server; agent chỉ làm nhiệm vụ điều phối/tool-calling.
- [ ] **Lỗi & biên**: `get_product` trả lỗi đẹp khi không khớp; `reserve_stock` kiểm tra tồn kho âm; `calc_shipping` validate input.
- [ ] **On-prem model**: chạy được với **LiteLLM+vLLM/Ollama** (không cần cloud).
- [ ] **STDIO & (tuỳ chọn) SSE**: STDIO chạy ổn; có README/flag để chuyển sang SSE/sidecar khi lên K8s.
- [ ] **Test**: `pytest` pass cho các hàm chính.

---

9) Troubleshooting nhanh

- **Server STDIO không spawn**: kiểm tra đường dẫn `server.py` là **absolute** hoặc build từ `__file__` như ví dụ.
- **Không thấy tool trong Dev UI**: thiếu quyền spawn process, thiếu `__init__.py`, hoặc lỗi import lib `mcp`.
- **Tool trả về object lạ**: đảm bảo return **JSON-serializable** (dict/list/str...), tránh object tuỳ biến.
- **Sidecar/HTTP**: nếu dùng SSE/HTTP, cấu hình **auth**/network và **tool_filter** để giới hạn quyền.

---

10) Nâng cấp (gợi ý nhẹ cho Bước 3/4)

- Thêm tool `quote_price(sku, qty)` & `create_order(cart)` để sau này **Shopping Agent ↔ Merchant Agent** (A2A) có thứ để thương lượng.
- Chuẩn hoá **payload** (idempotency key, currency, decimals) để map sang **AP2 mandates** ở Bước 4.
- Log/audit: ghi file JSON cho mỗi call_tool (sẵn tiện làm audit trail khi tích hợp AP2.