Bước 3 (A2A) — Multi-Agent on-prem (Shopping ↔ Merchant)

Mục tiêu: Tạo hệ **đa tác nhân** gồm **Shopping Agent** (đại diện người dùng) và **Merchant Agent** (đại diện bên bán) giao tiếp theo phong cách **A2A**. Ở bước này ta dùng một **A2A-lite shim** (HTTP + JSON envelope + HMAC) để chạy on-prem, dễ debug/K8s. Khi cần, bạn có thể **thay shim bằng SDK A2A chính thức** (mapping 1:1 theo verb/payload/envelope).

Sau bước này bạn có: - 2 process/Pod tách rời: `shopping_agent` (ADK+LLM) và `merchant_agent` (FastAPI + nghiệp vụ + MCP từ Bước 2). - Giao tiếp A2A kiểu **request/response** có **idempotency** và **HMAC ký thông điệp**. - Các **flow** hoàn chỉnh: `quote` (báo giá), `reserve` (giữ hàng), `create_order` (chuẩn bị cho AP2 ở Bước 4).

---

1) Kiến trúc & luồng

```
[User]
  │ chat
  ▼
Shopping Agent (ADK + LLM)
  │  A2A-lite (HTTP, HMAC, Envelope)
  ▼
Merchant Agent (FastAPI)
  │   ├─ gọi MCP tools: get_product / calc_shipping / reserve_stock
  │   └─ nghiệp vụ quote / reserve / create_order
  ▼
MCP Server (Bước 2)
```

Flow chính (quote → reserve → create_order): 1) User: “Mua SKU002 số lượng 2, báo giá & ship về Hanoi.” 2) Shopping Agent gọi **A2A verb** `quote` → Merchant Agent. 3) Merchant gọi **MCP**: `get_product`, `calc_shipping` → trả **Quote** (đơn giá, ship, tổng, tồn kho). 4) User chấp thuận → Shopping Agent gọi `reserve`. Merchant **giảm stock** qua MCP. 5) Merchant trả **Reservation** (reserved qty, stock mới). Tuỳ chọn: `create_order` (giỏ hàng chuẩn bị cho AP2 ở Bước 4).

---

2) Cấu trúc dự án

```
adk-a2a-step3/
├─ shared/
│  ├─ __init__.py
│  ├─ schemas.py        # pydantic models: Envelope, Quote/Reserve/Order
```

```
│  └─ security.py       # HMAC sign/verify, idempotency helpers
├─ mcp_server/          # (từ Bước 2)
│  └─ server.py
├─ merchant_agent/
│  ├─ __init__.py
│  └─ server.py         # FastAPI nhận A2A envelope, gọi MCP, trả phản hồi
└─ shopping_agent/
   ├─ __init__.py
   └─ agent.py          # ADK agent, tool gửi A2A request tới Merchant
```

---

3) Khai báo message schema (shared/schemas.py)

```python
# shared/schemas.py
from __future__ import annotations
from pydantic import BaseModel, Field
from typing import Optional, Literal, Dict, Any
from datetime import datetime

# ——— A2A envelope chung ———
class A2AEnvelope(BaseModel):
    message_id: str
    correlation_id: Optional[str] = None
    timestamp: datetime
    sender: str  # agent_id hoặc url
    receiver: str
    verb: Literal["quote", "reserve", "create_order"]
    payload: Dict[str, Any]
    # security
    signature: Optional[str] = None   # HMAC hex
    key_id: Optional[str] = None      # id của shared key

# ——— Payloads ———
class QuoteRequest(BaseModel):
    sku: str
    qty: int
    destination: str

class QuoteLine(BaseModel):
    sku: str
    unit_price: float
    currency: str
    qty: int
    shipping_fee: float
    total: float
```

```
    stock: int

class QuoteResponse(BaseModel):
    status: Literal["success", "error"]
    message: Optional[str] = None
    quote: Optional[QuoteLine] = None

class ReserveRequest(BaseModel):
    sku: str
    qty: int

class ReserveResponse(BaseModel):
    status: Literal["success", "error"]
    message: Optional[str] = None
    sku: Optional[str] = None
    reserved: Optional[int] = None
    stock: Optional[int] = None

class CreateOrderRequest(BaseModel):
    sku: str
    qty: int
    unit_price: float
    currency: str
    destination: str
    shipping_fee: float

class Order(BaseModel):
    order_id: str
    sku: str
    qty: int
    currency: str
    unit_price: float
    shipping_fee: float
    total: float

class CreateOrderResponse(BaseModel):
    status: Literal["success", "error"]
    message: Optional[str] = None
    order: Optional[Order] = None
```

---

4) Ký/kiểm chữ ký A2A (shared/security.py)

```
# shared/security.py
import hmac, hashlib, json, time
```

```python
from typing import Tuple

# Demo: shared secret đơn giản (prod: mTLS/KMS)
SHARED_KEY_ID = "k1"
SHARED_SECRET = b"change_me_strong_secret"

def canonicalize(envelope: dict) -> bytes:
    # loại signature trước khi ký
    data = {k: v for k, v in envelope.items() if k not in ("signature",)}
    # sort keys để ổn định
    raw = json.dumps(data, sort_keys=True, separators=(",", ":"))
    return raw.encode("utf-8")

def sign_envelope(envelope: dict) -> dict:
    buf = canonicalize(envelope)
    sig = hmac.new(SHARED_SECRET, buf, hashlib.sha256).hexdigest()
    envelope["signature"] = sig
    envelope["key_id"] = SHARED_KEY_ID
    return envelope

def verify_envelope(envelope: dict) -> Tuple[bool, str]:
    sig = envelope.get("signature", "")
    buf = canonicalize(envelope)
    exp = hmac.new(SHARED_SECRET, buf, hashlib.sha256).hexdigest()
    ok = hmac.compare_digest(sig, exp)
    return ok, ("ok" if ok else "bad-signature")
```

---

5) Merchant Agent (FastAPI + MCP) — merchant_agent/server.py

```python
# merchant_agent/server.py
import uuid, asyncio
from datetime import datetime, timezone
from fastapi import FastAPI, HTTPException, Request
from pydantic import BaseModel
from shared.schemas import (
    A2AEnvelope, QuoteRequest, QuoteResponse, QuoteLine,
    ReserveRequest, ReserveResponse, CreateOrderRequest, CreateOrderResponse,
Order
)
from shared.security import verify_envelope

# —— Dev mode: import trực tiếp MCP server funcs (Bước 2) ——
#  (Prod có thể dùng mcp client thay vì import)
from mcp_server.server import get_product, calc_shipping, reserve_stock
```

```
app = FastAPI(title="merchant_agent")

class A2AIn(BaseModel):
    envelope: A2AEnvelope

@app.post("/a2a/messages")
async def handle_a2a(inmsg: A2AIn):
    env = inmsg.envelope.model_dump()
    ok, reason = verify_envelope(env)
    if not ok:
        raise HTTPException(status_code=401, detail=reason)

    verb = env["verb"]
    payload = env["payload"]

    if verb == "quote":
        req = QuoteRequest(**payload)
        prod = await get_product(req.sku)
        if prod.get("status") != "success":
            return _wrap(env, QuoteResponse(status="error",
message="product not found"))
        p = prod["product"]
        ship = await calc_shipping(req.destination, float(req.qty))  # demo:
weight≈qty
        fee = ship["shipping_fee"]
        unit = float(p["price"])
        total = round(unit * req.qty + fee, 2)
        quote = QuoteLine(
            sku=req.sku, qty=req.qty, unit_price=unit, currency=p["currency"],
            shipping_fee=fee, total=total, stock=int(p.get("stock", 0))
        )
        return _wrap(env, QuoteResponse(status="success", quote=quote))

    elif verb == "reserve":
        req = ReserveRequest(**payload)
        r = await reserve_stock(req.sku, int(req.qty))
        if r.get("status") != "success":
            return _wrap(env, ReserveResponse(status="error",
message=r.get("error_message", "reserve fail")))
        return _wrap(env, ReserveResponse(status="success", sku=req.sku,
reserved=r["reserved"], stock=r["stock"]))

    elif verb == "create_order":
        req = CreateOrderRequest(**payload)
        total = round(req.unit_price * req.qty + req.shipping_fee, 2)
        order = Order(
            order_id=str(uuid.uuid4()), sku=req.sku, qty=req.qty,
```

```
            currency=req.currency, unit_price=req.unit_price,
            shipping_fee=req.shipping_fee, total=total,
        )
        return _wrap(env, CreateOrderResponse(status="success", order=order))

    else:
        raise HTTPException(status_code=400, detail="unknown verb")


def _wrap(env: dict, data: BaseModel):
    # response envelope (giữ correlation_id để shopping agent match)
    return {
        "message_id": str(uuid.uuid4()),
        "correlation_id": env.get("message_id"),
        "timestamp": datetime.now(timezone.utc).isoformat(),
        "sender": "merchant_agent",
        "receiver": env.get("sender"),
        "verb": env.get("verb"),
        "payload": data.model_dump(),
    }

# uvicorn merchant_agent.server:app --host 0.0.0.0 --port 8082
```

---

6) Shopping Agent (ADK + tool gửi A2A) — shopping_agent/agent.py

```
# shopping_agent/agent.py
import uuid, httpx
from datetime import datetime, timezone
from google.adk.agents import LlmAgent
from google.adk.models.lite_llm import LiteLlm
from shared.schemas import A2AEnvelope
from shared.security import sign_envelope

MERCHANT_URL = "http://localhost:8082/a2a/messages"

# ——— Tool: gửi A2A envelope tới Merchant ———
def a2a_request(verb: str, payload: dict) -> dict:
    env = A2AEnvelope(
        message_id=str(uuid.uuid4()),
        timestamp=datetime.now(timezone.utc),
        sender="shopping_agent",
        receiver="merchant_agent",
        verb=verb,
        payload=payload,
```

```
    ).model_dump()
    env = sign_envelope(env)
    with httpx.Client(timeout=15.0) as client:
        res = client.post(MERCHANT_URL, json={"envelope": env})
        res.raise_for_status()
        return res.json()  # response envelope

# ——— Model on-prem (vLLM/OpenAI-compat) ———
model = LiteLlm(
    model="google/gemma-2-9b-it",
    api_base="http://localhost:8000/v1"
)

root_agent = LlmAgent(
    model=model,
    name="shopping_agent",
    instruction=(
        "You represent the buyer. For product/price/shipping/stock, call the
a2a_request tool with the right verb: \n"
        "- quote: {sku, qty, destination}\n"
        "- reserve: {sku, qty}\n"
        "- create_order: {sku, qty, unit_price, currency, destination,
shipping_fee}\n"
        "Always summarize the merchant's response for the user in Vietnamese."
    ),
    tools=[a2a_request]
)

# Chạy: adk web   → chọn shopping_agent
# Sau khi Merchant Agent đã chạy cổng 8082
```

---

7) Cách chạy & kiểm thử

1) **Chạy MCP server** (Bước 2):

```
python mcp_server/server.py   # stdio mode nếu gọi trực tiếp; ở đây ta import
trực tiếp trong merchant (dev mode)
```

> Ở dev mode, Merchant import trực tiếp các hàm MCP (đơn giản hoá). Khi deploy production, chuyển sang **MCP client** để gọi qua stdio/SSE.

2) **Chạy Merchant Agent**

```
uvicorn merchant_agent.server:app --host 0.0.0.0 --port 8082
```

3) **Chạy Shopping Agent (ADK)**

```
adk web
# mở http://127.0.0.1:8000 → chọn "shopping_agent"
```

Ví dụ hội thoại: - "Báo giá SKU002 số lượng 2 ship về Hanoi." → tool `a2a_request(verb="quote", payload={...})` - "OK, giữ hàng giúp mình." → `reserve` - "Tạo đơn luôn." → `create_order`

Test cURL trực tiếp (bỏ qua LLM):

```
curl -s -X POST http://localhost:8082/a2a/messages
  -H 'Content-Type: application/json'
  -d '{
    "envelope": {
      "message_id": "123",
      "timestamp": "2025-09-18T04:00:00Z",
      "sender": "shopping_agent",
      "receiver": "merchant_agent",
      "verb": "quote",
      "payload": {"sku":"SKU002","qty":2,"destination":"Hanoi"},
      "key_id": "k1",
      "signature": "will-be-overwritten-if-using-tool"
    }
  }'
```

---

8) Definition of Done (PASS Bước 3)

- [ ] **Hai tiến trình/POD tách rời**: shopping (ADK) & merchant (FastAPI) chạy độc lập.
- [ ] **Giao tiếp A2A-lite OK**: `quote → reserve → create_order` trả về envelope có `correlation_id` khớp.
- [ ] **Bảo mật cơ bản**: yêu cầu **HMAC signature** hợp lệ mới xử lý; thử thay payload → 401.
- [ ] **Idempotency**: dùng `message_id` / `correlation_id` để chống xử lý lặp (có ghi TODO/impl sơ bộ).
- [ ] **MCP được gọi** ở merchant (ít nhất cho `quote` và `reserve`).
- [ ] **Quan sát được**: log input/output, có thể gắn Prometheus sau.

---

9) Nâng cấp/K8s gợi ý

- **Service mesh/mTLS** cho kênh A2A giữa Pods.
- **Ingress** riêng cho Merchant, hạn chế IP nguồn (Shopping) hoặc JWT.
- **Prometheus metrics**: `a2a_requests_total{verb=...}`, `a2a_latency_seconds_bucket`, `a2a_errors_total`.
- **Retry + backoff** ở Shopping khi Merchant lỗi mạng; thêm **idempotency key** để không trừ stock 2 lần.
- **MCP client chuẩn** trong Merchant thay vì import trực tiếp server.

---

10) Mapping sang SDK A2A chính thức

- **Envelope** ↔ **Message**; `message_id` / `correlation_id` ↔ **traceId/conversationId** (tuỳ SDK).
- **verb** ↔ **action/capability**; có thể gắn **Agent Card** mô tả: supports `quote/reserve/create_order`.
- **HMAC signature** ↔ **Credential/JWT/mTLS**; có thể thay bằng **Verifiable Presentation** khi gom chung với AP2.
- Khi chuyển, chỉ cần thay `a2a_request()` ở Shopping và `/a2a/messages` ở Merchant bằng **send/receive** của SDK A2A; payload/schema giữ nguyên.

---

11) Bước tiếp theo (đi vào Bước 4 — AP2)

- Biến `CreateOrderResponse.order` thành **Cart** hợp chuẩn, thêm `line_items`, `merchant_id`, `nonce`, `timestamp`.
- Sinh **Intent/Cart Mandate** (ký số) và đính kèm vào thông điệp A2A → Payment Agent.
- Thêm **Payment Agent** (service thứ 3) để mô phỏng AP2: nhận cart + mandate, kiểm chứng chữ ký, phát hành **Payment Mandate**.