Python Pro Sample: Event-Driven Order Processor Microservice

This document contains an extremely long, highly informative Python code sample that simulates a production-grade event-driven microservice. This example covers advanced features such as async programming, data validation with Pydantic, logging, background task queues; retry mechanisms, and metrics exposure via Prometheus.

Code:

```
import asyncio
import logging
import signal
from typing import Optional

from aiohttp import web
from pydantic import BaseModel, Field, ValidationError
from pydantic_settings import BaseSettings
from prometheus_client import Counter, generate_latest, CONTENT_TYPE_LATEST

import aiohttp
import random

# ---- Configuration ----

class Settings(BaseSettings):
    service_name: str = "order_processor"
    host: str = "0.0.0.0"
    port: int = 8080
    external_inventory_api: str = "https://fake-inventory.example.com/api/check"
    max_retries: int = 5
    retry_backoff: float = 0.5  # seconds

    class Config:
        env_file = ".env"


settings = Settings()
```

```
# ---- Logging ----

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s %(levelname)s [%(name)s] %(message)s",
)
logger = logging.getLogger(settings.service_name)

# ---- Metrics ----

ORDERS_RECEIVED = Counter("orders_received_total", "Total orders received")
ORDERS_PROCESSED = Counter("orders_processed_total", "Total orders processed")
ORDERS_FAILED = Counter("orders_failed_total", "Total orders failed")

# ---- Models ----

class Order(BaseModel):
    id: str
    user_id: str
    product_id: str
    quantity: int = Field(gt=0)


# ---- Middleware ----

@web.middleware
async def metrics_middleware(request, handler):
    try:
        response = await handler(request)
        return response
    except Exception as e:
        logger.exception("Unhandled exception")
        raise web.HTTPInternalServerError()


# ---- Retry Mechanism ----

async def retry_with_backoff(coro_func, max_retries=3, base_delay=0.5):
    for attempt in range(max_retries):
```

```
        try:
            return await coro_func()
        except Exception as e:
            logger.warning(f"Retry {attempt + 1}/{max_retries} failed: {e}")
            await asyncio.sleep(base_delay * (2 ** attempt))
    raise Exception(f"All {max_retries} retries failed.")


# ---- External Inventory Check ----

async def check_inventory(product_id: str, quantity: int) -> bool:
    async def _check():
        # Simulated flaky inventory API
        if random.random() < 0.3:
            raise Exception("Inventory API timeout")
        return random.choice([True, False])

    return await retry_with_backoff(_check, max_retries=settings.max_retries,
base_delay=settings.retry_backoff)


# ---- Order Processing ----

async def process_order(order: Order):
    logger.info(f"Processing order {order.id} for user {order.user_id}")
    inventory_ok = await check_inventory(order.product_id, order.quantity)
    if not inventory_ok:
        raise Exception(f"Product {order.product_id} out of stock")
    logger.info(f"Order {order.id} processed successfully")
    ORDERS_PROCESSED.inc()


# ---- HTTP Handlers ----

async def handle_order(request: web.Request):
    ORDERS_RECEIVED.inc()
    try:
        data = await request.json()
        order = Order(**data)
        await process_order(order)
```

```
        return web.json_response({"status": "ok"}, status=200)
    except ValidationError as e:
        logger.warning(f"Validation error: {e}")
        return web.json_response({"error": "Invalid order format"}, status=400)
    except Exception as e:
        logger.error(f"Order failed: {e}")
        ORDERS_FAILED.inc()
        return web.json_response({"error": str(e)}, status=500)


async def handle_metrics(request):
    return web.Response(
        body=generate_latest(),
        content_type=CONTENT_TYPE_LATEST
    )


# ---- Graceful Shutdown ----

async def on_shutdown(app):
    logger.info("Shutting down service...")


# ---- App Setup ----

def create_app():
    app = web.Application(middlewares=[metrics_middleware])
    app.add_routes([
        web.post("/order", handle_order),
        web.get("/metrics", handle_metrics),
    ])
    app.on_shutdown.append(on_shutdown)
    return app


# ---- Entrypoint ----

def main():
    app = create_app()
    web.run_app(app, host=settings.host, port=settings.port)
```

```
if __name__ == "__main__":
    main()
```