# 3. Morpheus Nonlinear Routing: 비선형 파이프라인 및 라우팅 제어

## 프로젝트 개요

Morpheus의 고급 기능인 **비선형 파이프라인 구축**과 \*\*RouterStage\*\*를 활용하여 데이터 메시지를 동적으로 분기하는 로직을 실습했습니다. 이는 서로 다른 조건의 데이터(예: 정상 vs. 비정상)를 각기 다른 모델이나 분석 로직으로 보내는 데 필수적인 기술입니다.

## ⚙️ 주요 기술 및 역량

* **기술:** Morpheus Pipeline (vs. LinearPipeline), RouterStage, key\_fn (라우팅 함수), MRC Broadcast.
* **역량:**
  1. **비선형 구조 이해:** Pipeline 클래스를 사용하여 메시지 흐름이 일방적이지 않고 복잡하게 분기/결합되는 구조를 정의하고 시각화(pipeline.visualize)했습니다.
  2. **조건부 라우팅:** RouterStage를 활용하여 메시지의 페이로드(user 상태)를 검사하고, '성공 로그인' 메시지는 브랜치 A로, '실패 로그인' 메시지는 브랜치 B로 분기하는 **동적 제어 로직**을 구현했습니다.
  3. **내부 그래프 구축:** 커스텀 스테이지 내부에 MRC 노드를 병렬 또는 직렬로 연결하는 **그래프 구축 능력**을 입증했습니다.

## 🛠️ 주요 구현 내용 (Router.ipynb 중심)

1. **라우팅 키 정의:** 메시지 데이터 내 status 컬럼 값(success/failure)을 기준으로 \*\*라우팅 키(key)\*\*를 반환하는 router\_key\_fn 함수를 작성.
2. **Router Stage:** RouterStage에 라우팅 키 함수와 가능한 출력 키("route\_a", "route\_b")를 전달하고, 라우터의 출력 포트를 사용하여 각기 다른 다운스트림 스테이지에 연결했습니다.
3. **메시지 태깅:** 라우팅된 메시지가 최종 싱크에 도착했을 때, 메시지에 **어떤 브랜치**를 통과했는지 확인하는 메타데이터(processing\_branch)를 추가하여 라우팅 성공 여부를 검증했습니다.

## 📌 주요 코드 로직 (라우팅 함수 및 연결 추상화)

from morpheus.stages.general.router\_stage import RouterStage  
from morpheus.messages import ControlMessage, MessageMeta  
  
# 1. 라우팅 키 함수 정의  
def router\_key\_fn(message: ControlMessage) -> str:  
 # ControlMessage의 페이로드(DataFrame)에 접근하여 조건 확인  
 df = message.payload().get\_data()  
   
 # 예시: 'status' 컬럼의 첫 번째 값이 'success'인지 확인  
 if df['status'].iloc[0] == 'success':  
 return "success\_route"  
 else:  
 return "failure\_route"  
  
# 2. 파이프라인 연결 시:  
router = pipeline.add\_stage(RouterStage(config,   
 keys=["success\_route", "failure\_route"],   
 key\_fn=router\_key\_fn))  
  
# 3. 라우터 출력 연결 (비선형 분기)  
pipeline.add\_edge(router.output\_ports[0], success\_sink)  
pipeline.add\_edge(router.output\_ports[1], failure\_sink)