



# Cocktail of Environments

How to Mix Test and Development Environments and Stay Alive



**@aatarasoff**



**@aatarasoff**



**@aatarasoff**



**@aatarasoff**



**@dmitryi.ulianov**



**@d-ulyanov**



**@dmitrii-ulianov**

# Prologue: When CTO Comes to You

# Initial state



Dev



Prod

# Typical Environments

**Dev**

**Stage**

**Prod**

miro

## Goals

- **Always stable testing env**
- **Minimize Dev vs QA gap**
- **Unblock parallel testing**
- **Try to keep it simple**

# Atypical Environments

**Dev**

**Stage**

**Prod**

miro



# One Cluster - Several Environments



miro

# Stable Dev

RC Dev

Branch Dev

Stable Dev

**Always contains all the services with the same versions as in production**

miro

# Stable Dev

RC Dev

Branch Dev

Stable Dev

Always contains all the services with the same versions as in production

Default routes come to it

# Stable Dev

RC Dev

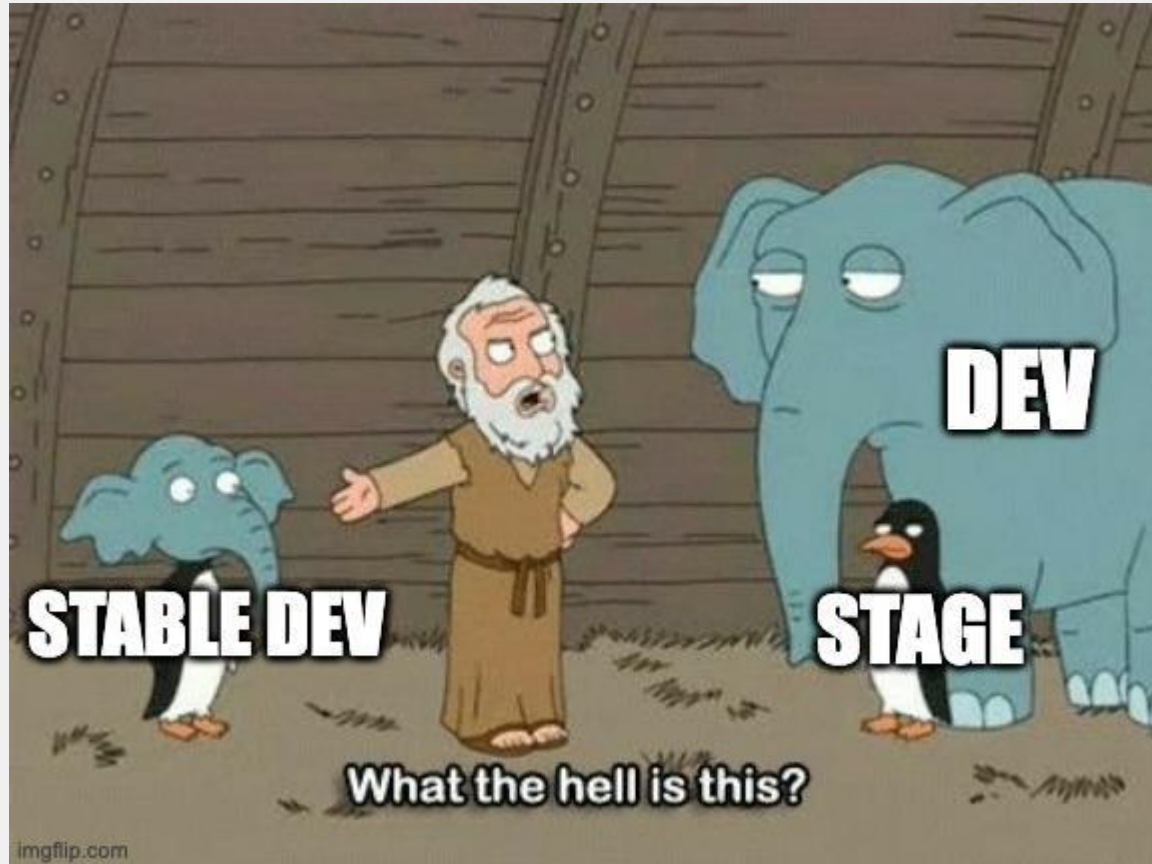
Branch Dev

Stable Dev

**Foundation for developing, testing, and staging**

miro

# Stable Dev Explained



# Branch Dev



miro

# Release Candidates Dev

Every new release is deployed as a candidate first

RC Dev

Branch Dev

Stable Dev

miro

## Issues to address

- **Routing aka Service Mesh**
- **Event Routing**
- **Data Isolation**

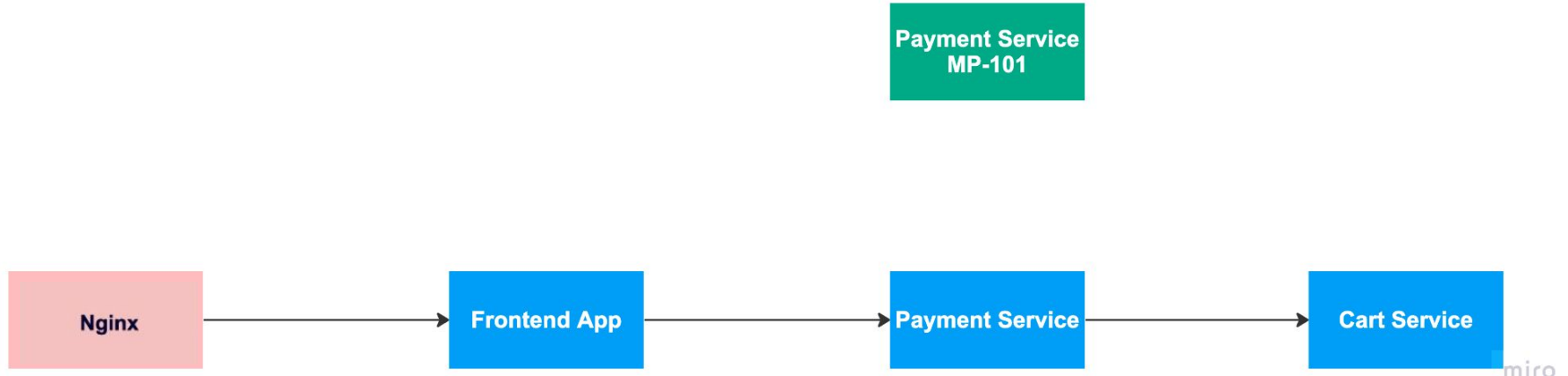


# Chapter 1: Service Mesh

## **Issues to address**

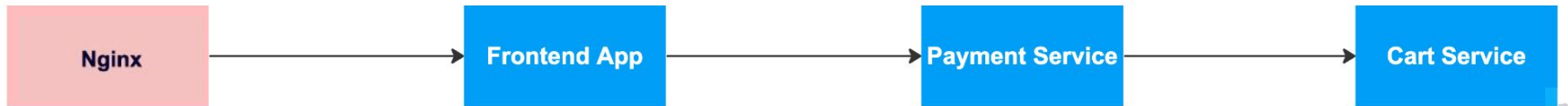
- **Test a release candidate in a call chain**
- **Test a branch version**

# Service Injection



# Service Injection

As a developer I created new branch: feature/mp-101-bla-bla



miro

# Service Injection

Deploy each branch to dev cluster

Pipeline Needs Jobs 6 Tests 0

Group jobs by

validate

✓ validate-branch ↻

test

✓ test ↻

build

✓ build-docker ↻

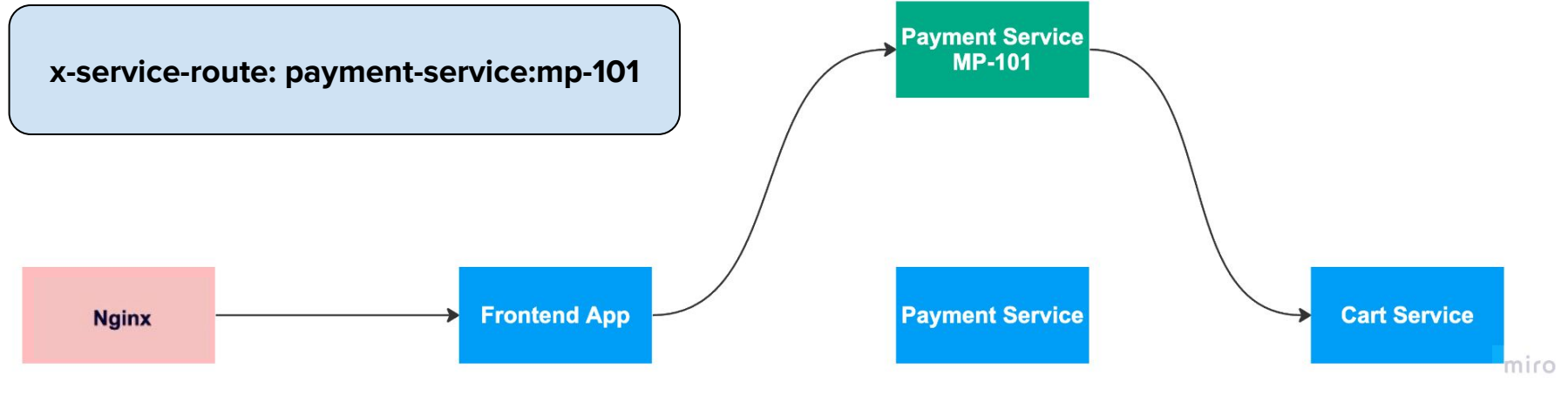
✓ build-go ↻

deploy

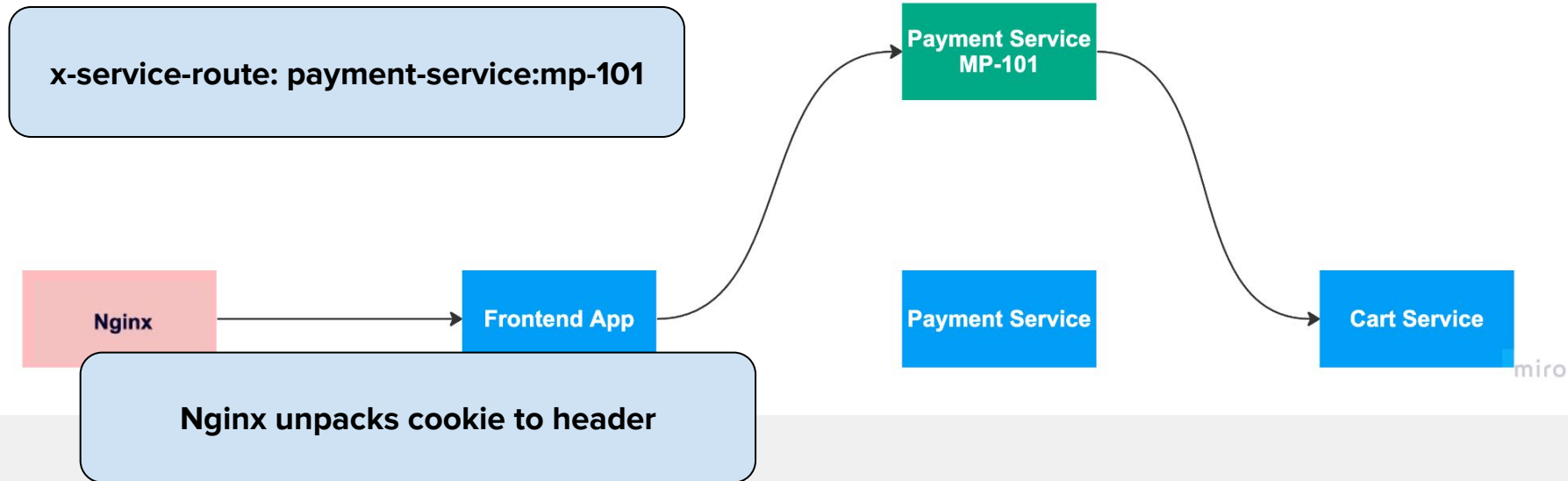
✓ deploy-branch-to-dev ↻

⚙️ destroy-dev ■

# Service Injection

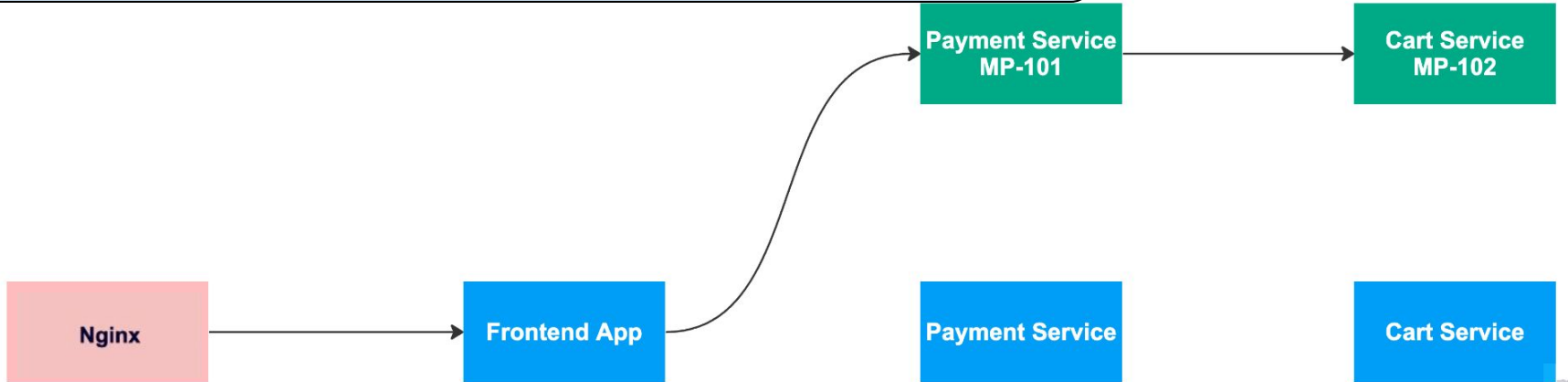


# Service Injection



# We Need More Branches

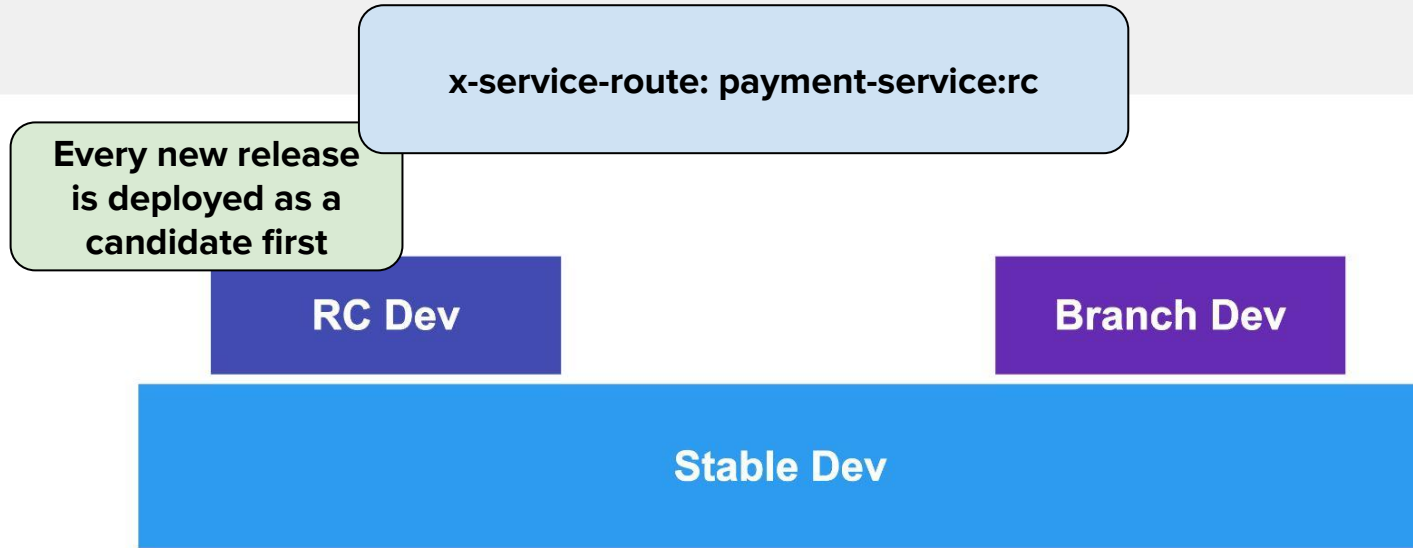
`x-service-route: payment-service:mp-101::cart-service:mp-102`



miro



# Release Candidates Testing



miro

# Istio Virtual Service

```
---
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: payment-service
spec:
  ...
  http:
  - name: stable
    route:
  - destination:
      host: payment-service.services.svc.cluster.local
```

# Istio Virtual Service

```
---
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: payment-service
spec:
  ...
  http:
  - name: stable
    route:
  - destination:
      host: payment-service.services.svc.cluster.local
```

Deploy for every stable version  
via Helm chart

# Route to a Branch

---

```
apiVersion: networking.istio.io/v1beta1
```

```
kind: VirtualService
```

```
metadata:
```

```
  name: payment-service
```

```
spec:
```

```
  ...
```

```
  http:
```

```
  - name: payment-service-mp-101
```

```
    match:
```

```
    - headers:
```

```
      x-service-route:
```

```
        regex: ^(payment-service:mp-101.*|.*::payment-service:mp-101.*)$
```

```
  - name: stable
```

```
    route:
```

```
    - destination:
```

```
      host: payment-service.services.svc.cluster.local
```

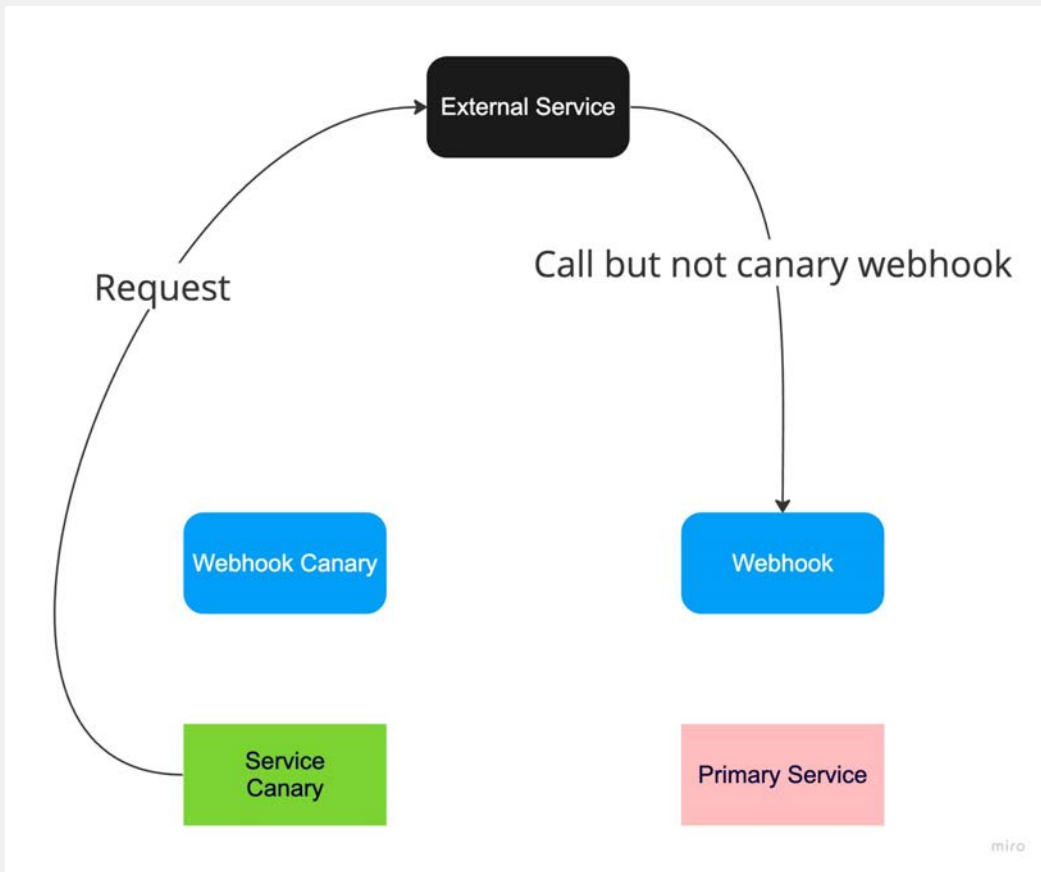
**We cannot add it with  
Helm chart**

# Virtual Service Merge Operator

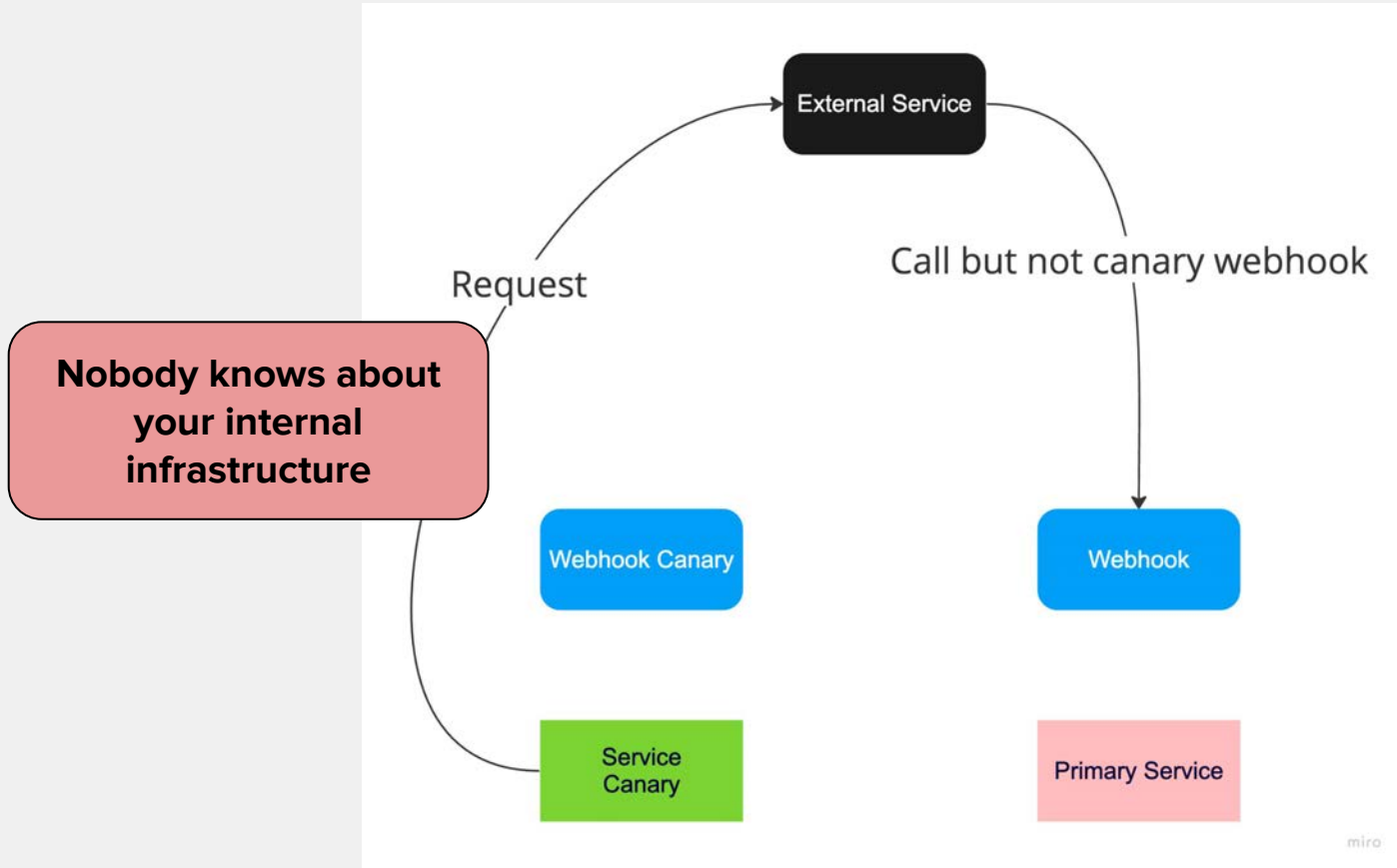
```
---
apiVersion: istiomerge.monime.sl/v1alpha1
kind: VirtualServiceMerge
metadata:
  name: payment-service-mp-101
spec:
  patch:
    http:
      - name: payment-service-mp-101
        match:
          - headers:
              x-service-route:
                regex: ^(payment-service:mp-101.*|.*::payment-service:mp-101.*)$
  target:
    name: payment-service
```

Deploy for every branch  
via Helm chart

# Tricky Case: Webhooks

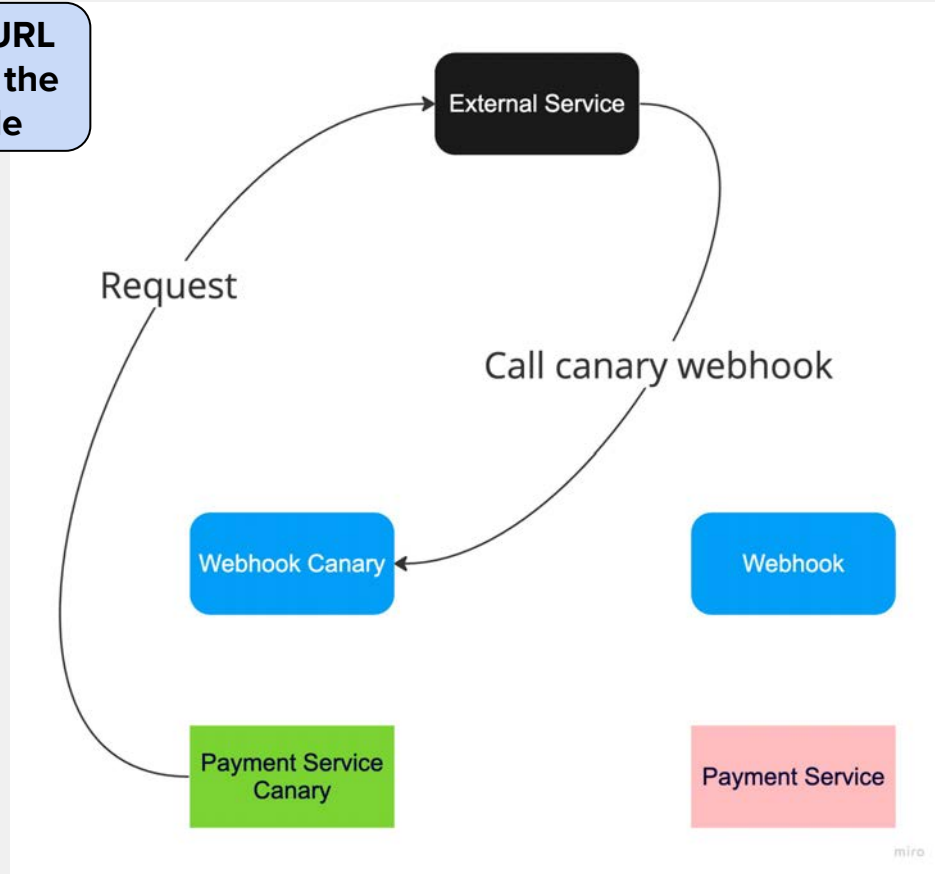


# Tricky Case: Webhooks



# Solution #1: Webhooks

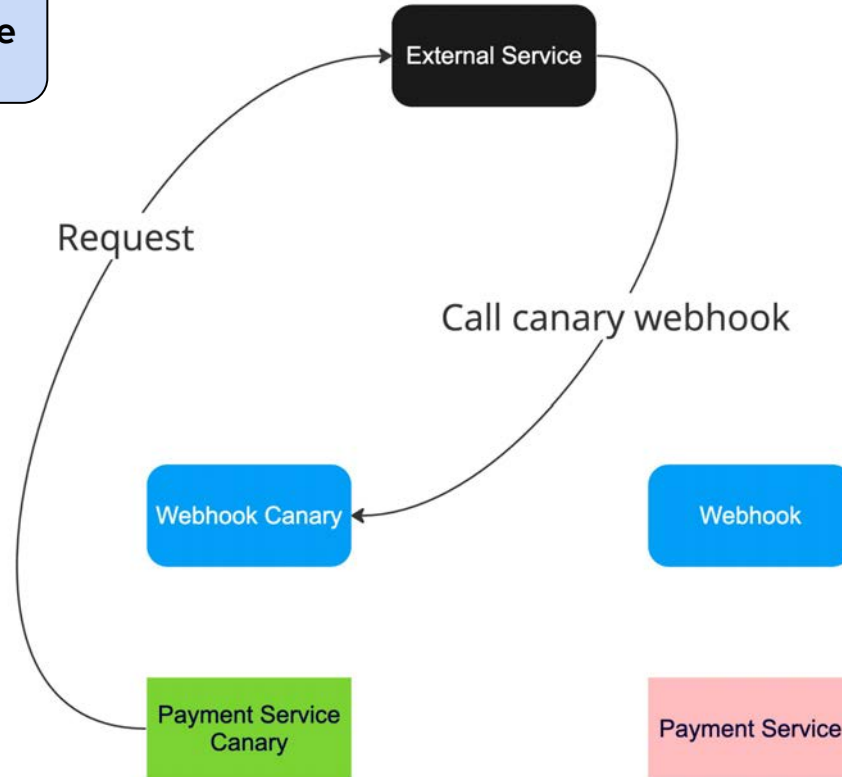
We can switch URL for webhook on the third-party side





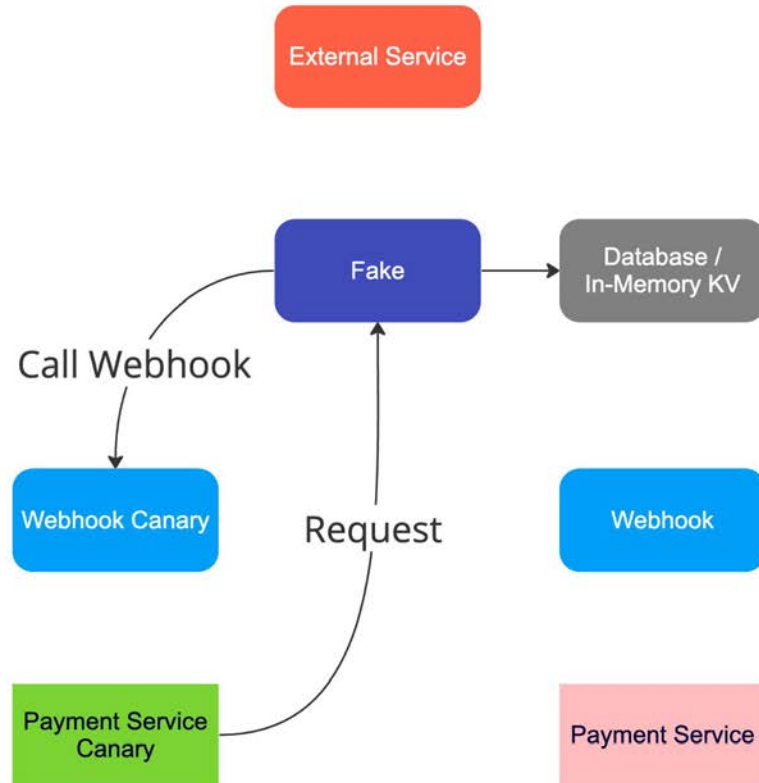
# Solution #1: Webhooks

**We can switch URL for webhook on the third-party side**



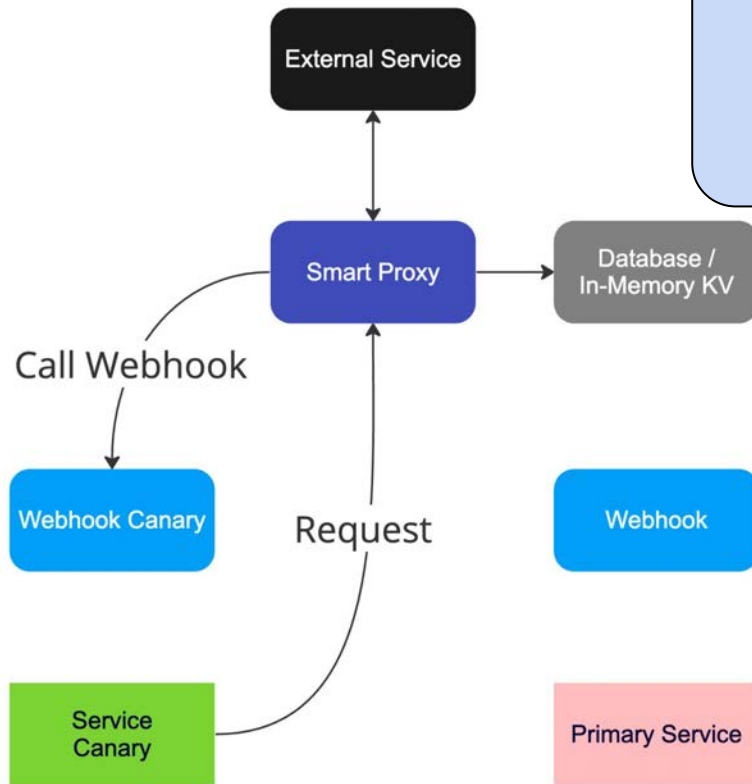
**However, we affect stable version**

# Solution #2: Webhooks



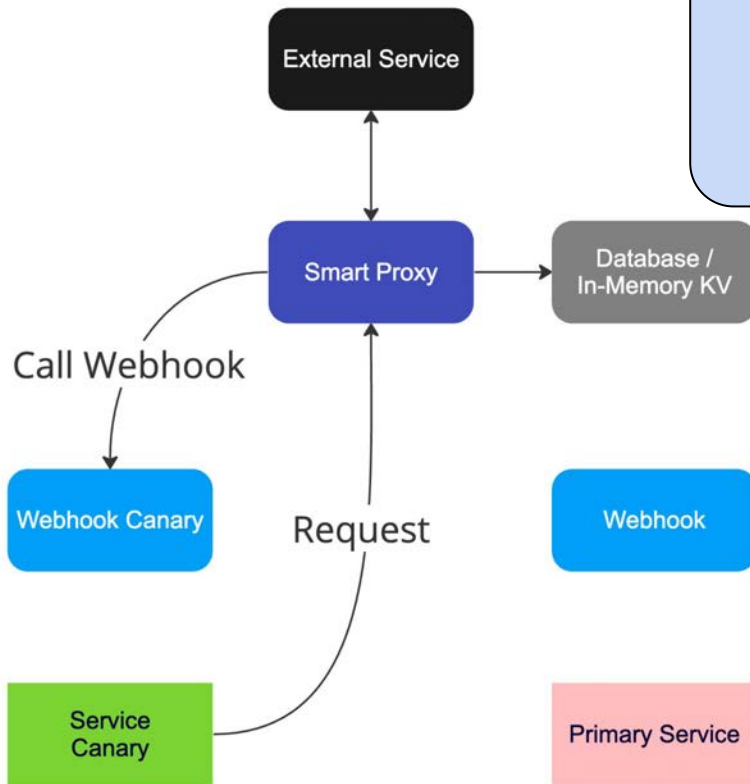
**Create advanced mock or Fake to get rid of External Service dependency at all**

# Solution #3: Webhooks



**Use Smart-Proxy  
and some correlation ID  
for matching requests  
from the external service**


# Solution #3: Webhooks



**Use Smart-Proxy and some correlation ID for matching requests from the external service**

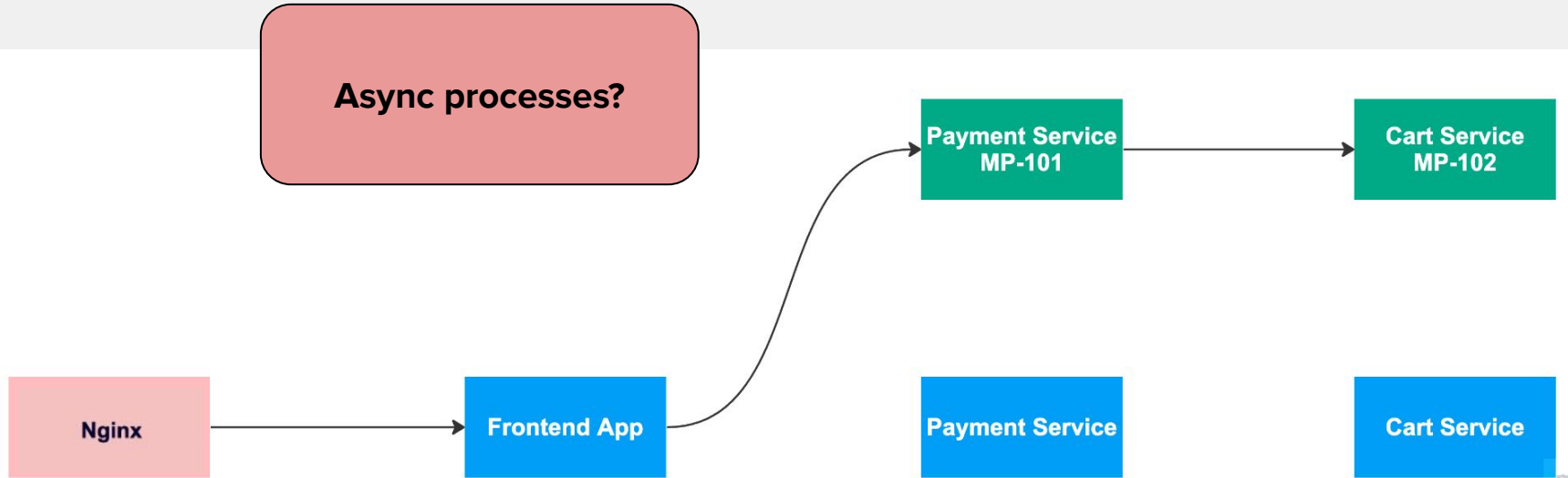
**We chose this approach**

**Recall  
the issues  
to address**

- **Routing aka Service Mesh** 
- **Event Routing**
- **Data Isolation**

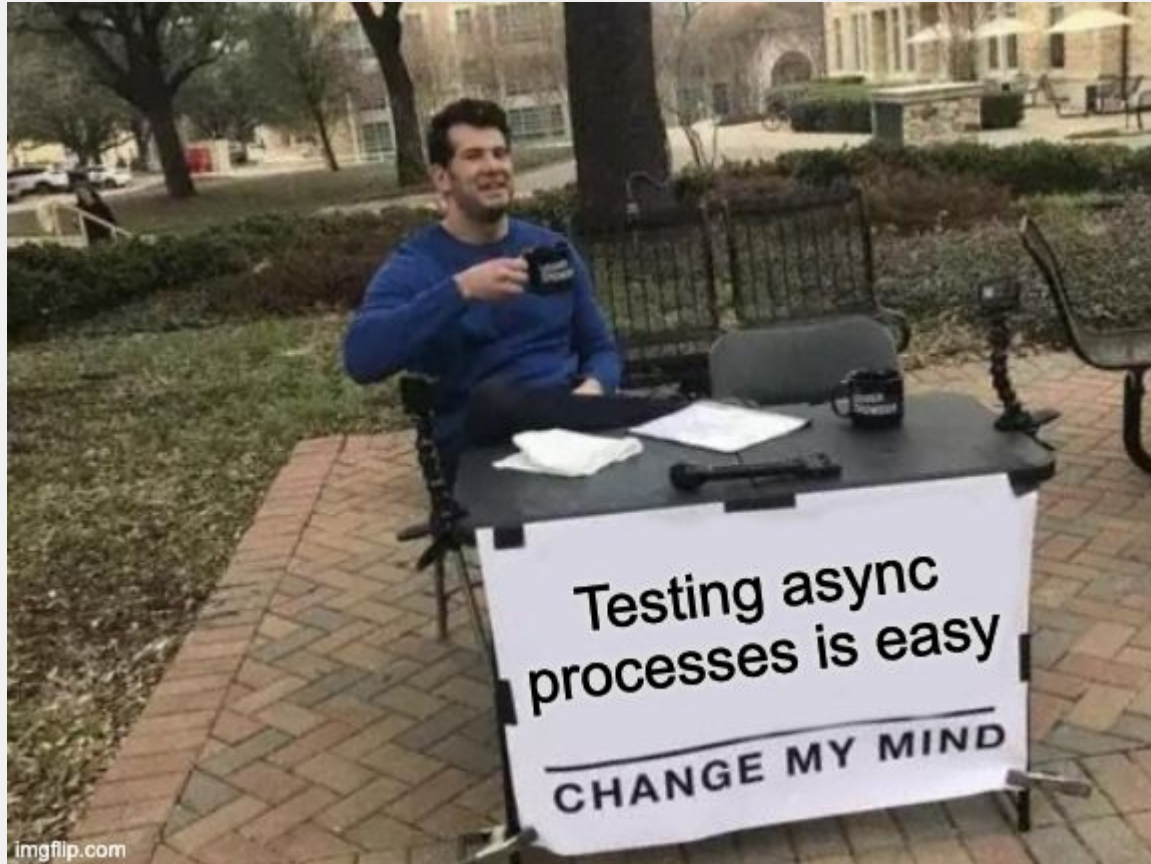
# Chapter 2: Event Routing

# What About Event-Driven?



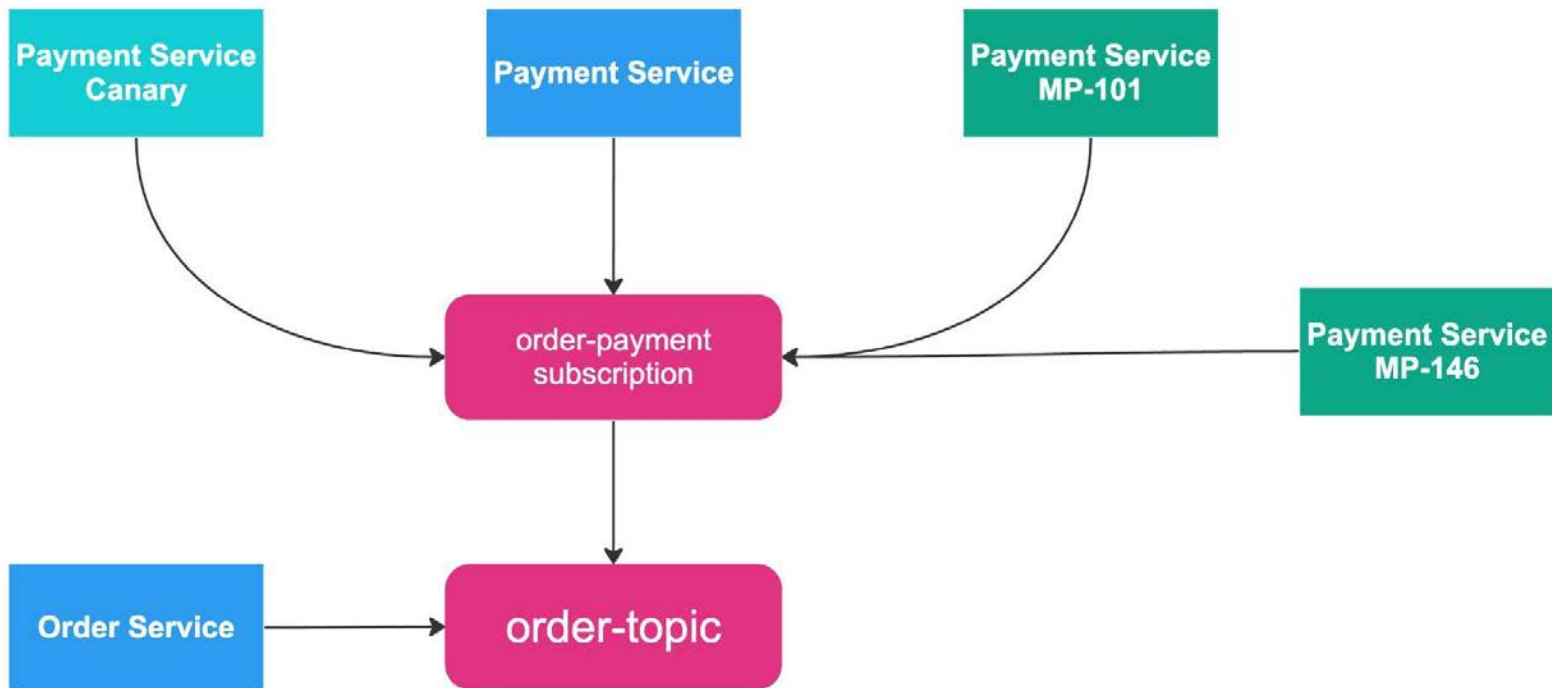
miro

# Unblocking Async Scenarios



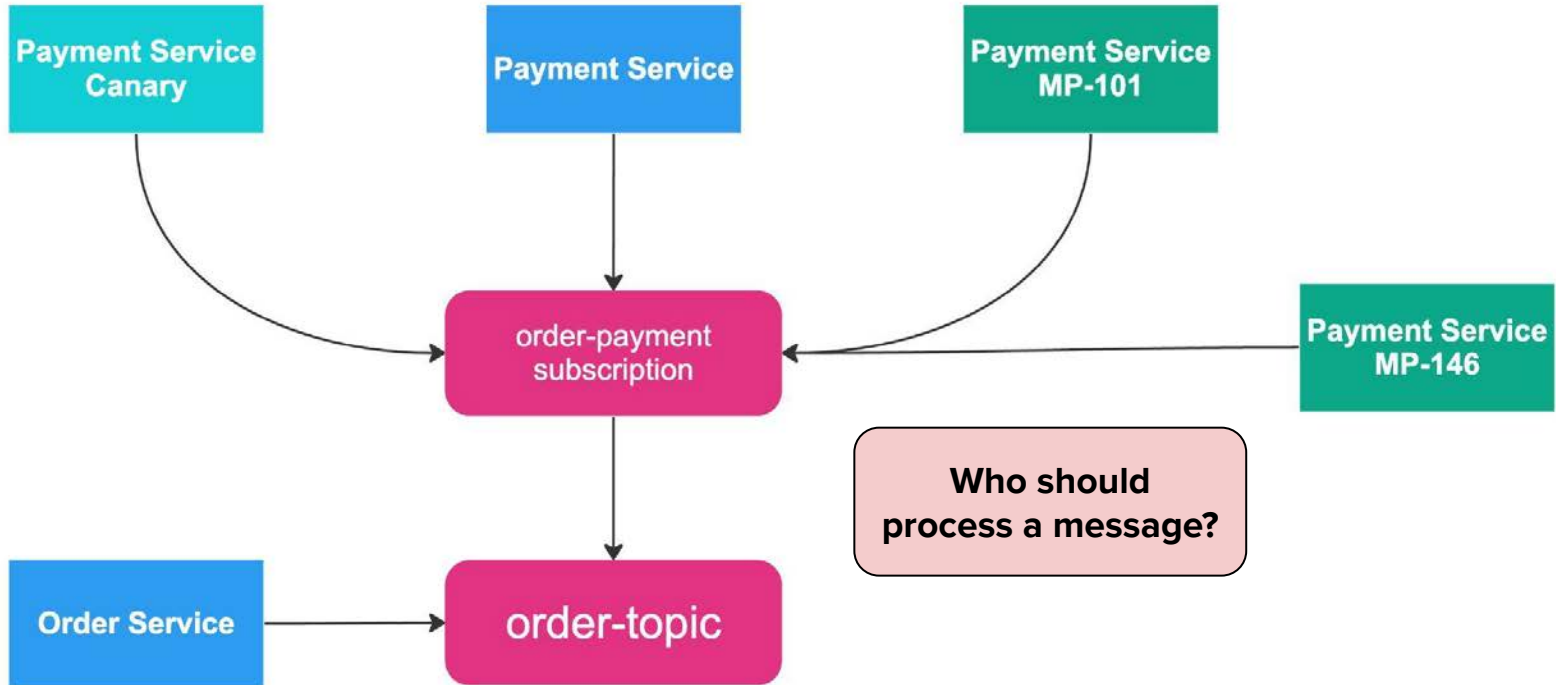


# Async Issues



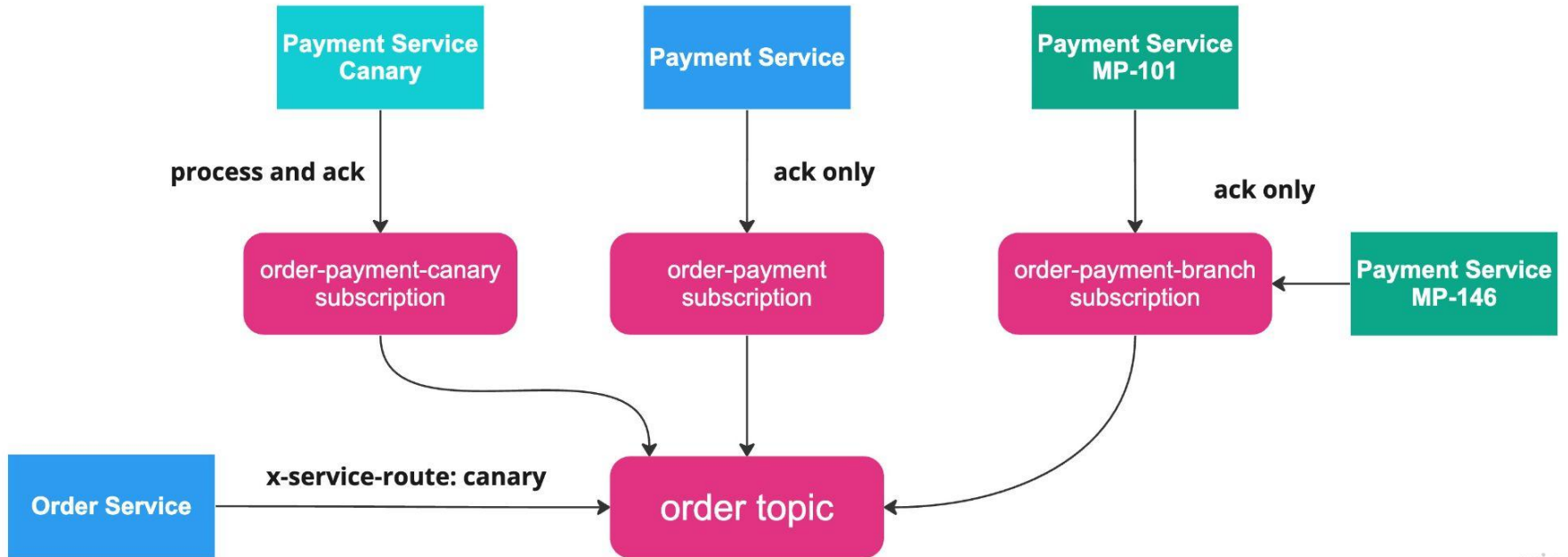
miro

# Async Issues



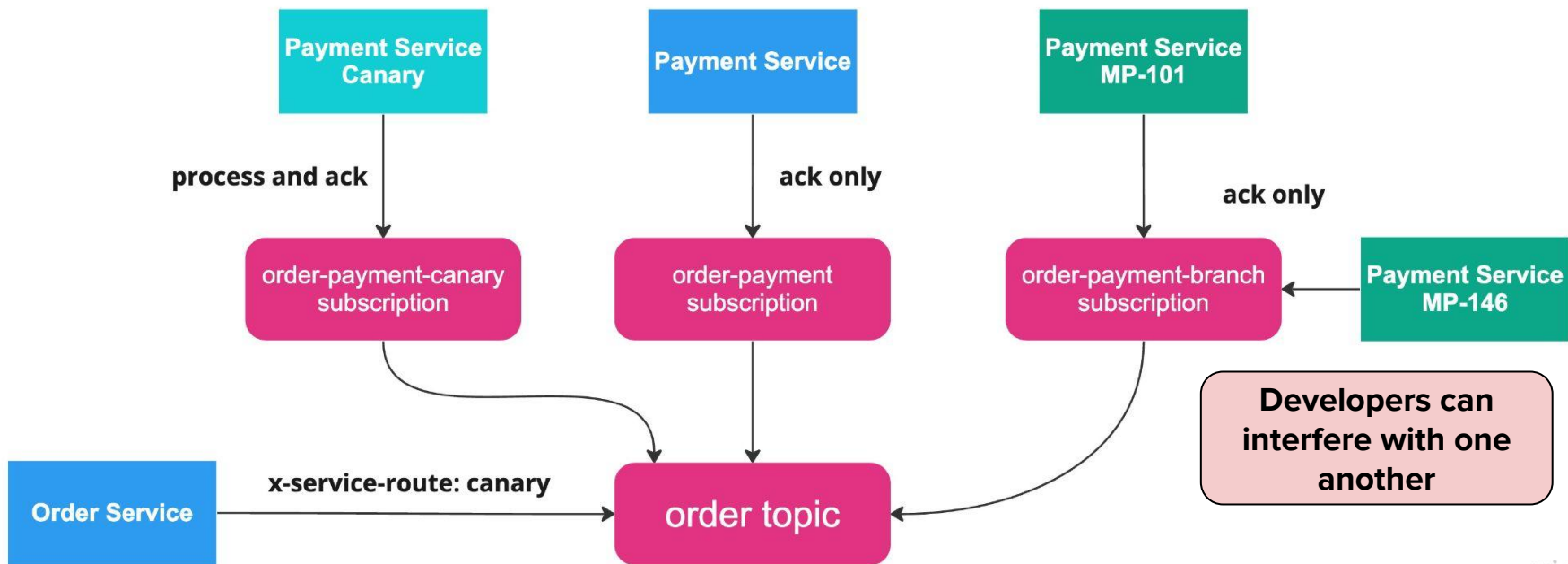
miro

# Let's Use Event Routing



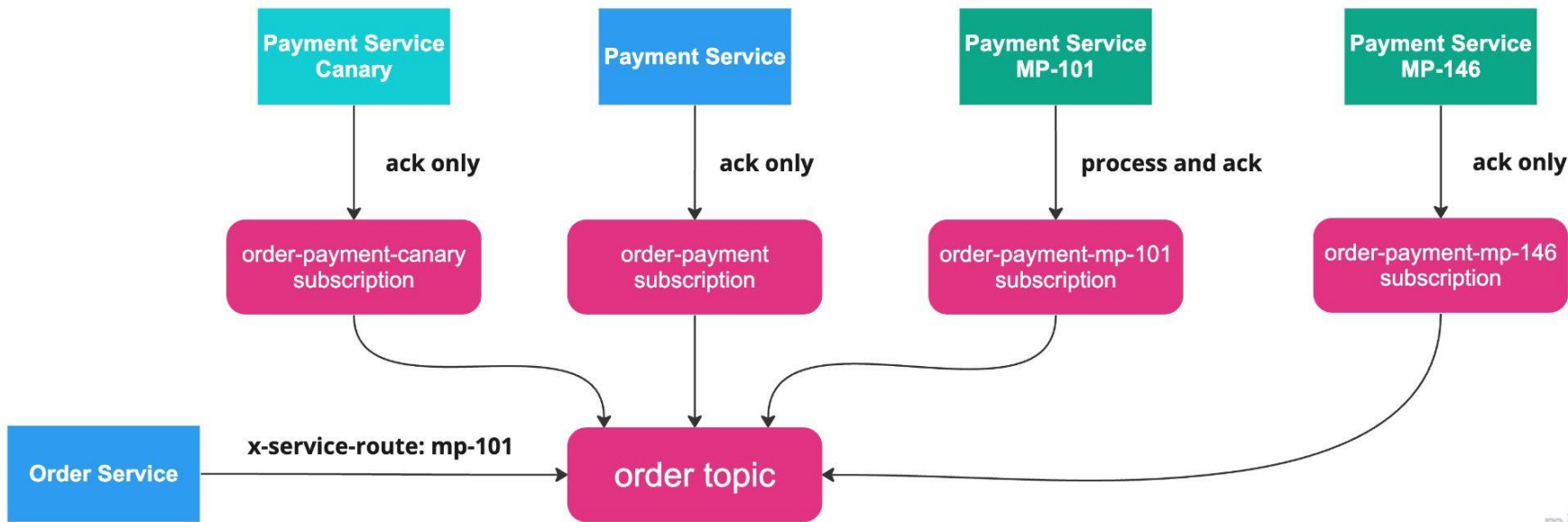
miro

# Subscription for All Branches



miro

# Subscription per Branch



miro

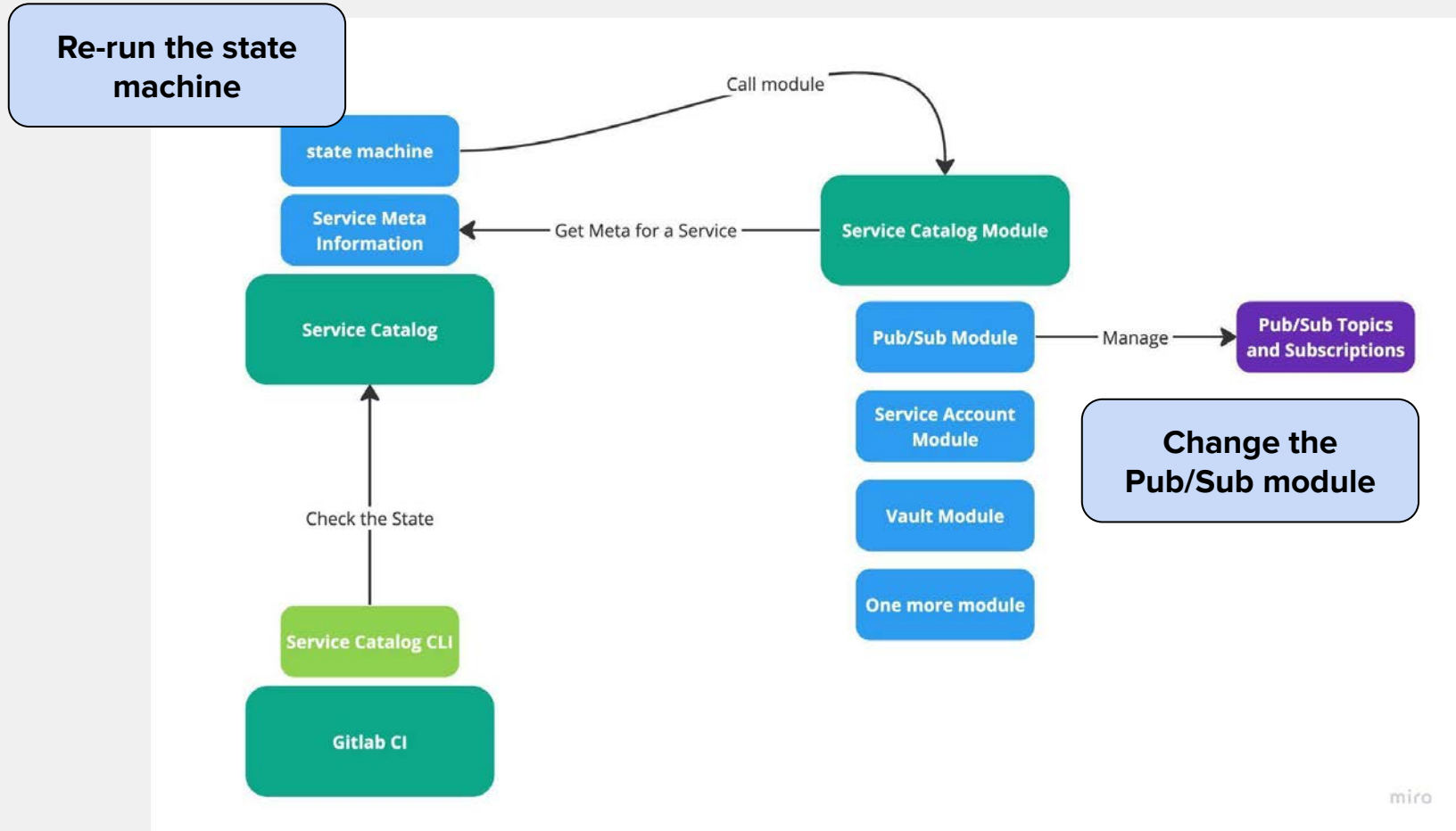
**To implement it  
we need**

- **Static subscription for RC**
- **Dynamic subscriptions for branches**
- **Common library**
  - **context propagation**
  - **message skip logic**

To implement it  
we need

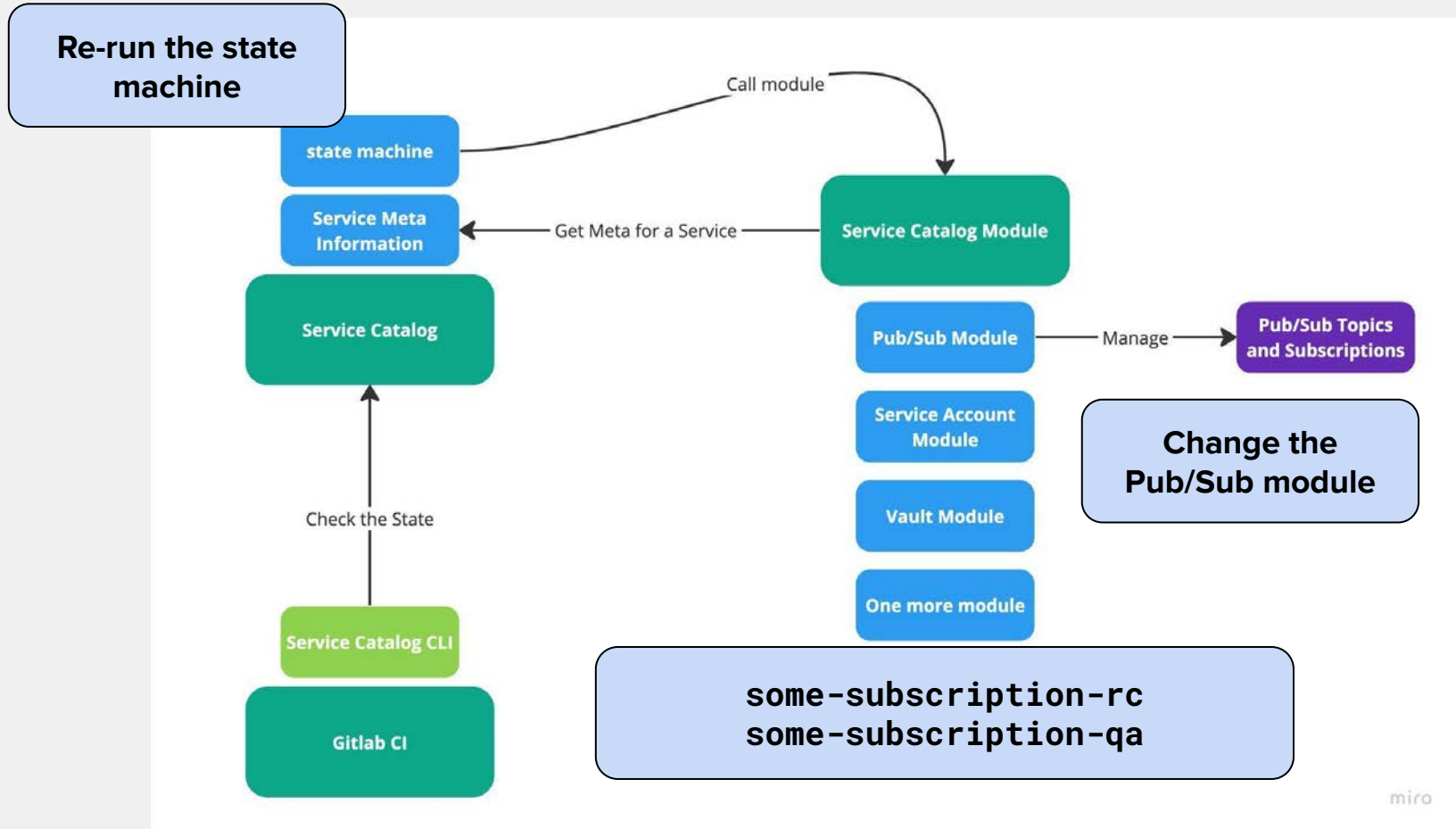
- **Static subscription for RC**
- **Dynamic subscriptions for branches**
- **Common library**
  - **context propagation**
  - **message skip logic**

# Static Subscriptions





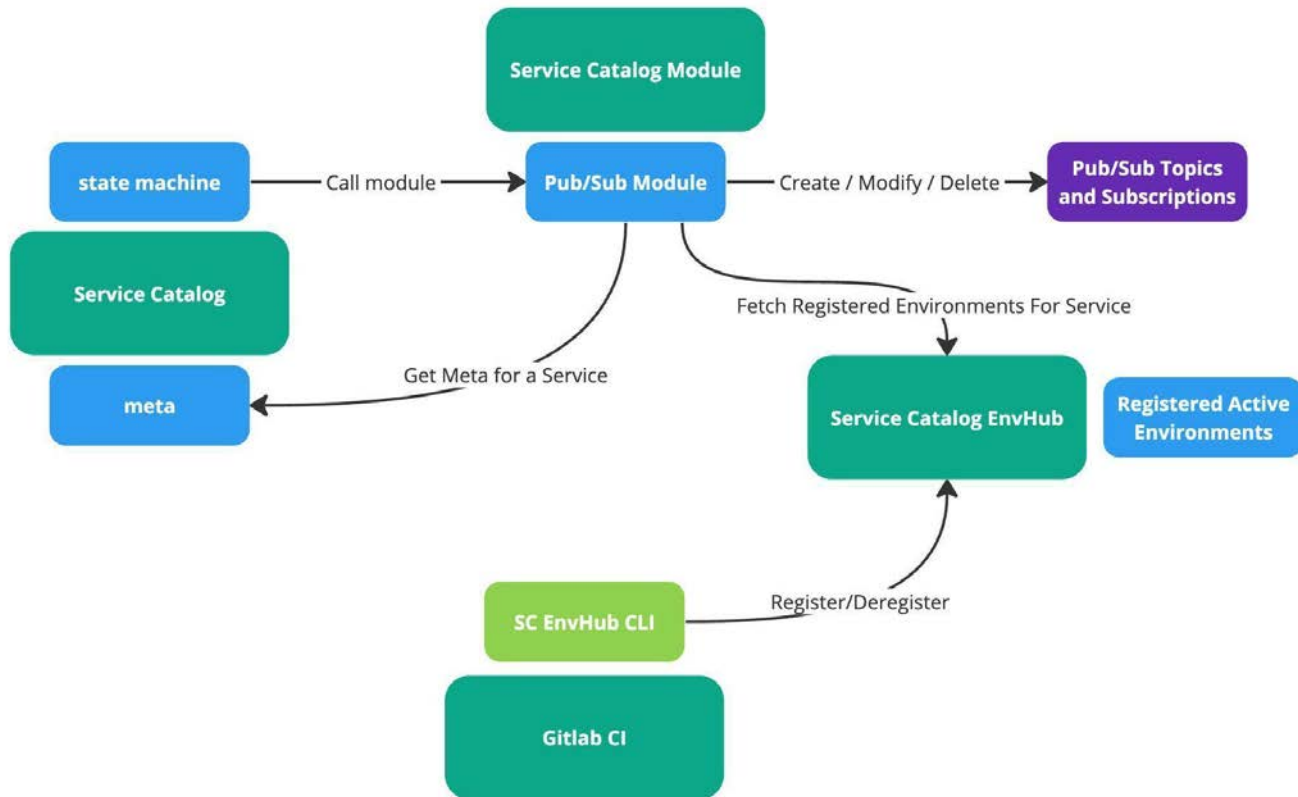
# Static Subscriptions



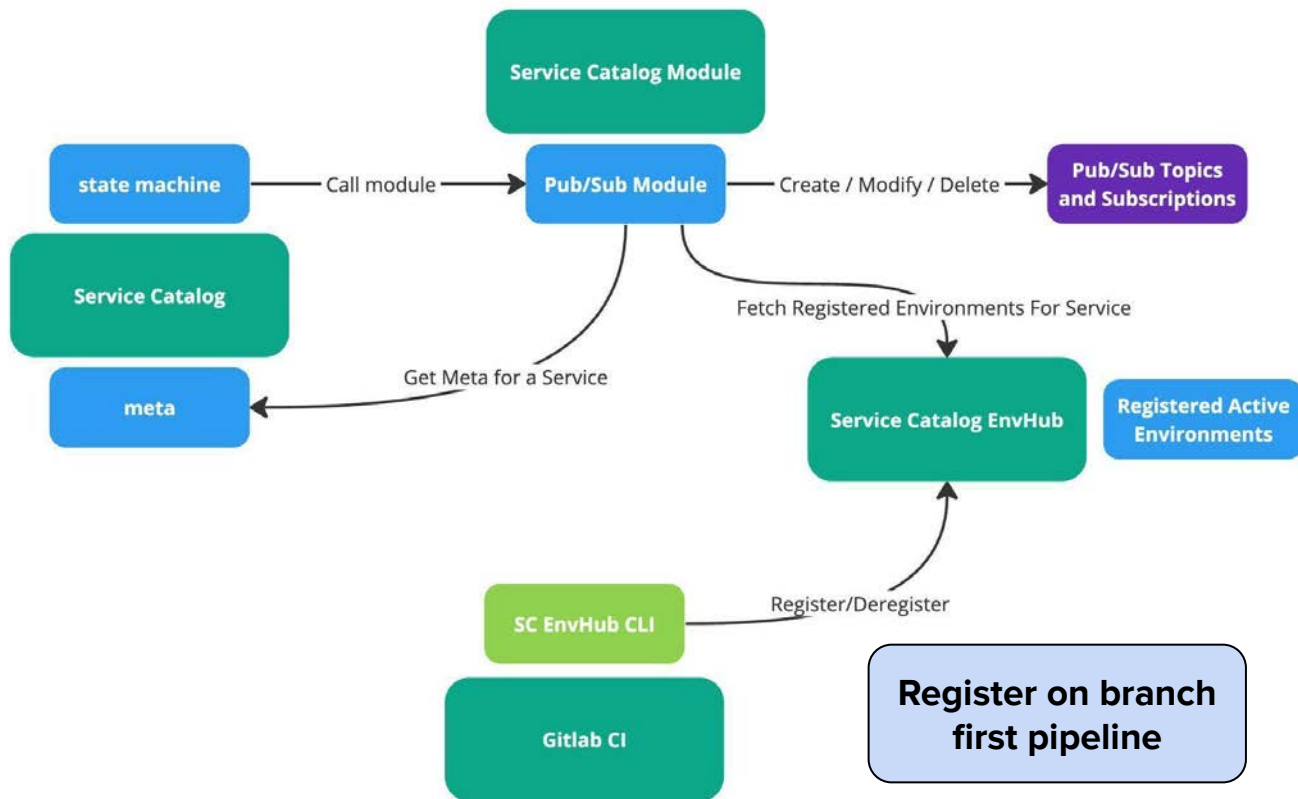
## Issues to address

- **Static subscription for canary**
- **Dynamic subscriptions for branches**
- **Common library**
  - **context propagation**
  - **message skip logic**

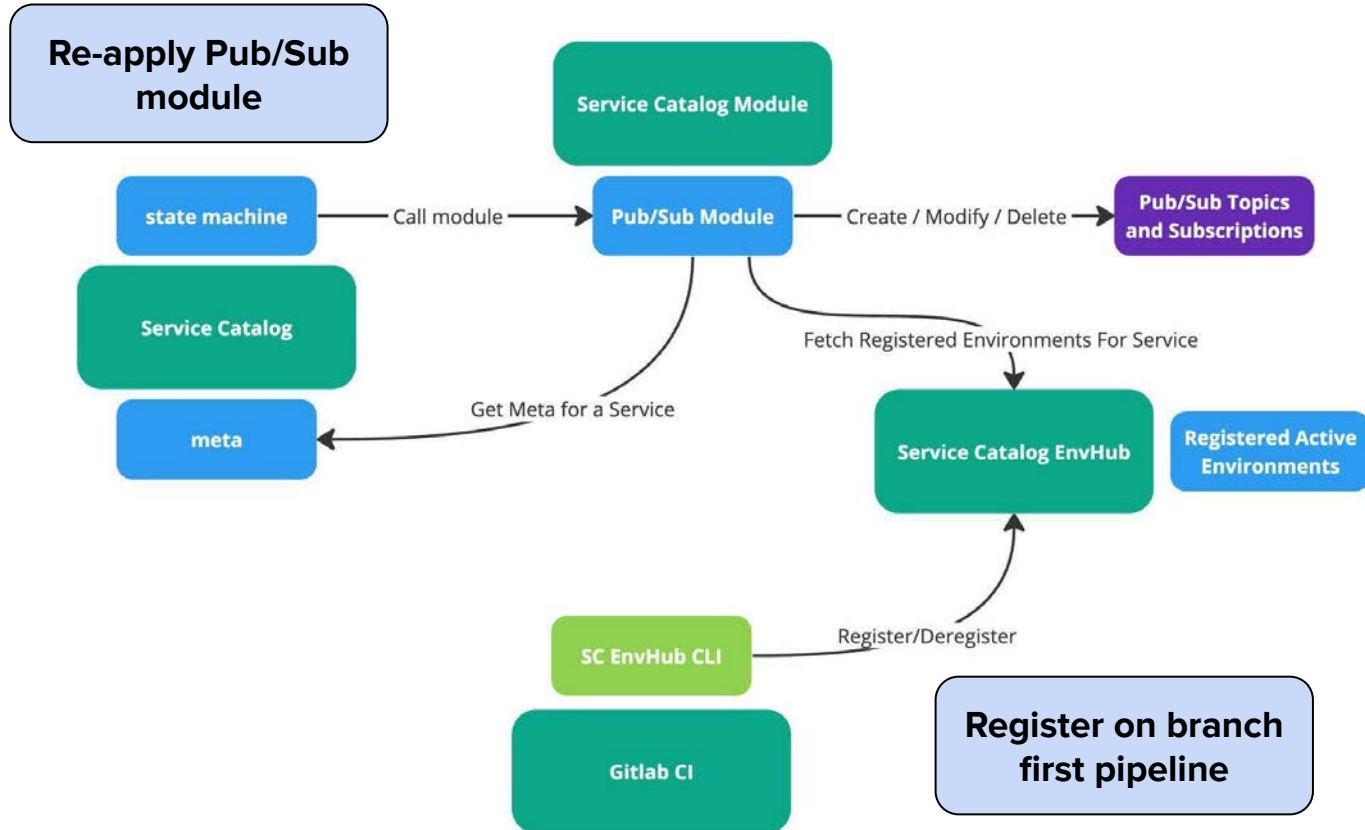
# Dynamic Subscriptions



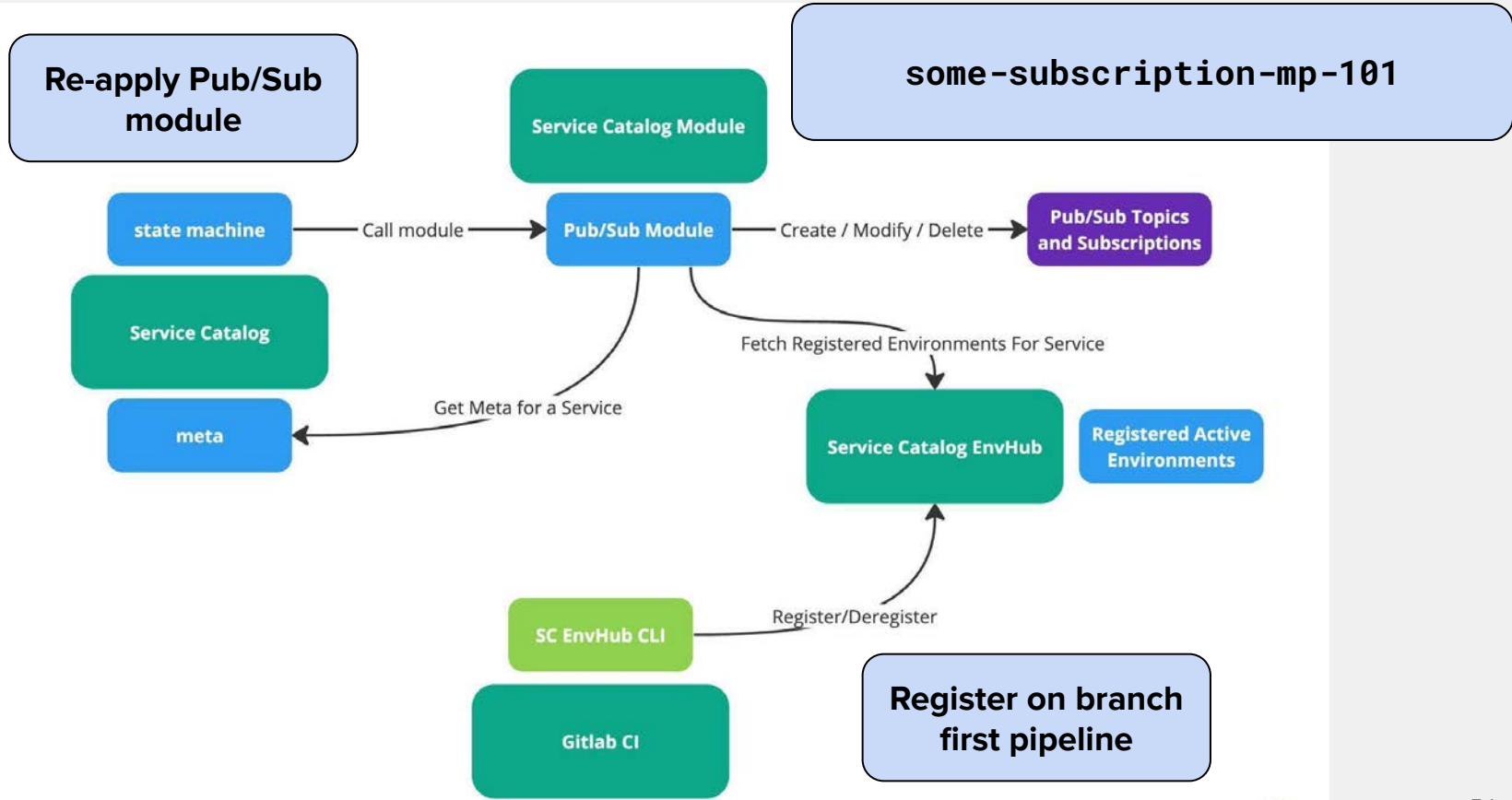
# Dynamic Subscriptions



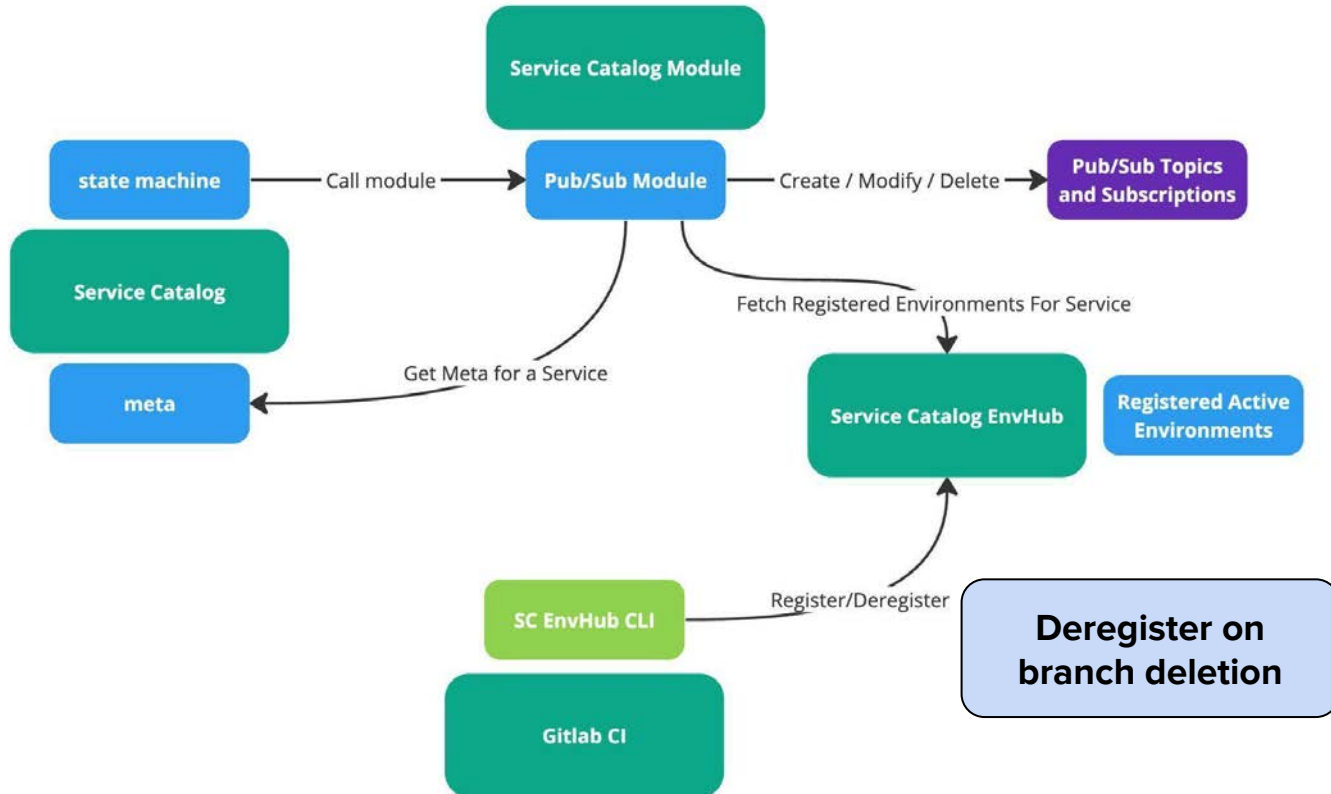
# Dynamic Subscriptions



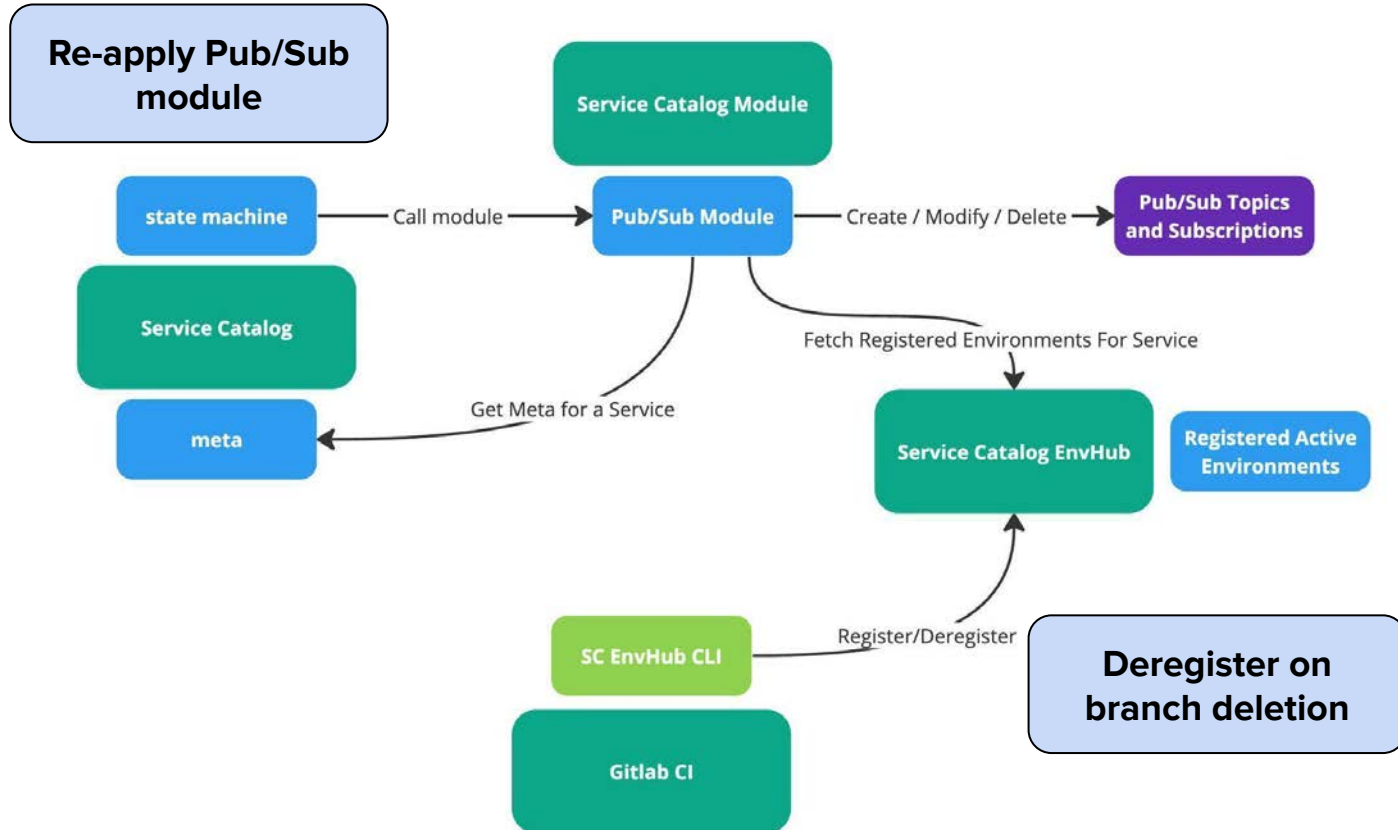
# Dynamic Subscriptions



# Dynamic Subscriptions

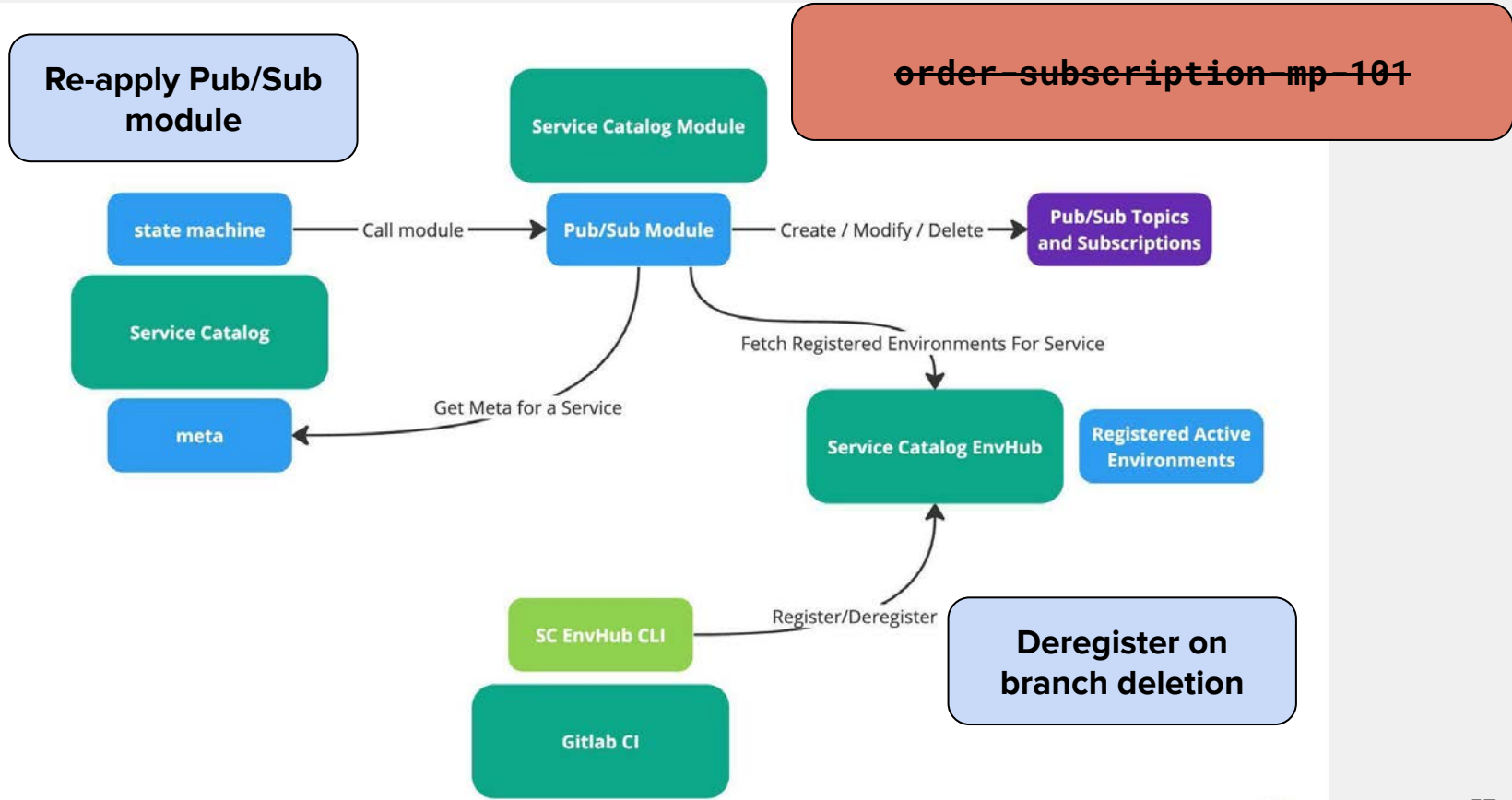


# Dynamic Subscriptions






# Dynamic Subscriptions




# Deployment Process


## test


test 


## bake

build-docker 


build-go 


check-provisioned-resources 

fetch-service-catalog-info 

provision-resources 

## deploy

deploy-branch-to-dev 

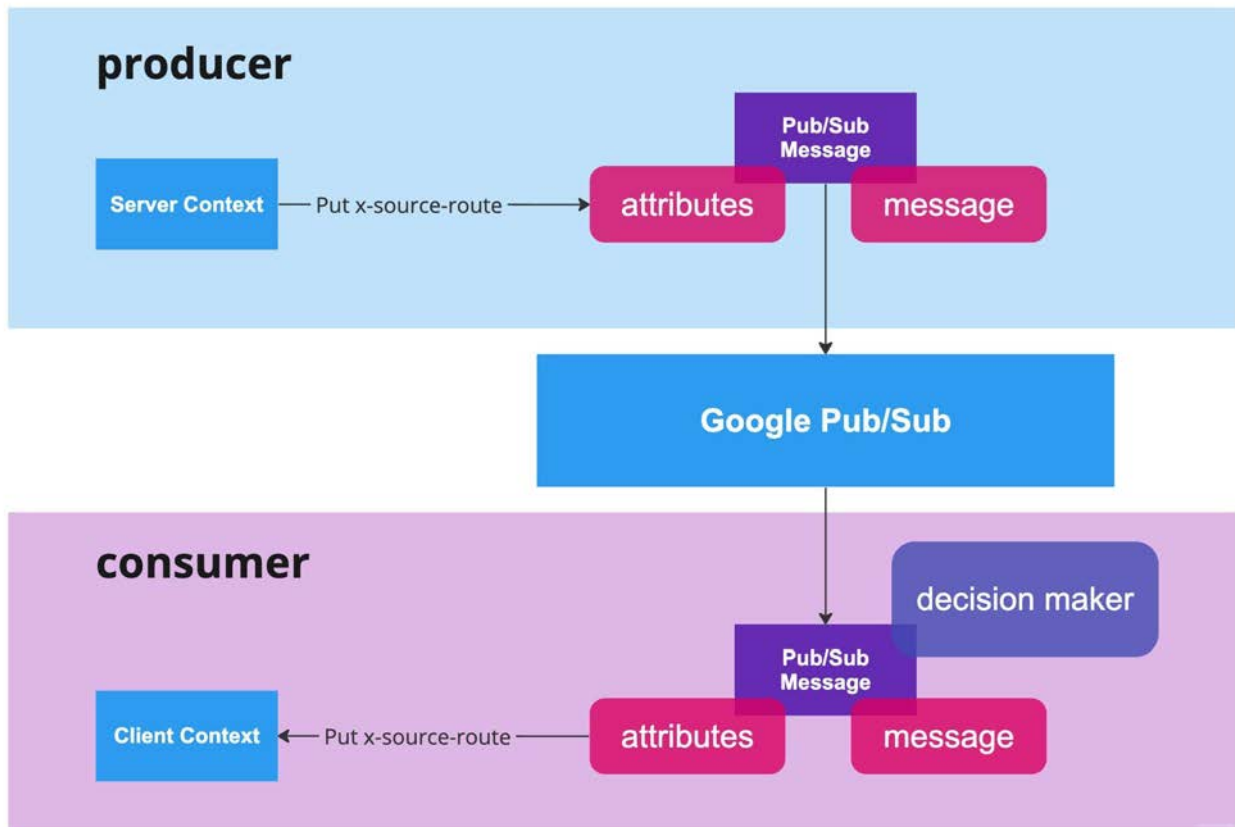
deprovision-resources 

destroy-dev 

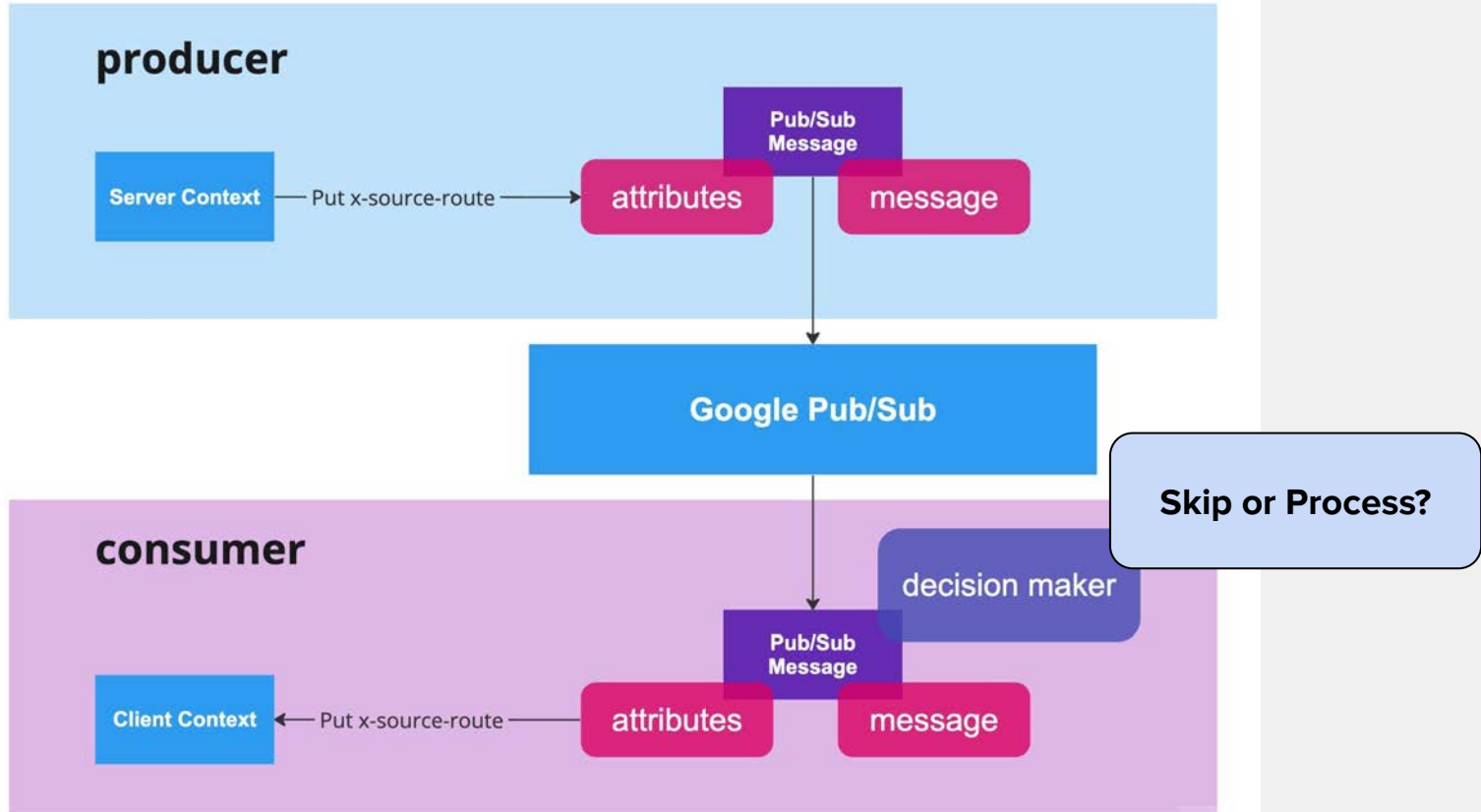
## Issues to address

- **Static subscription for canary**
- **Dynamic subscriptions for branches**
- **Common library**
  - **context propagation**
  - **message skip logic**

# Common Library

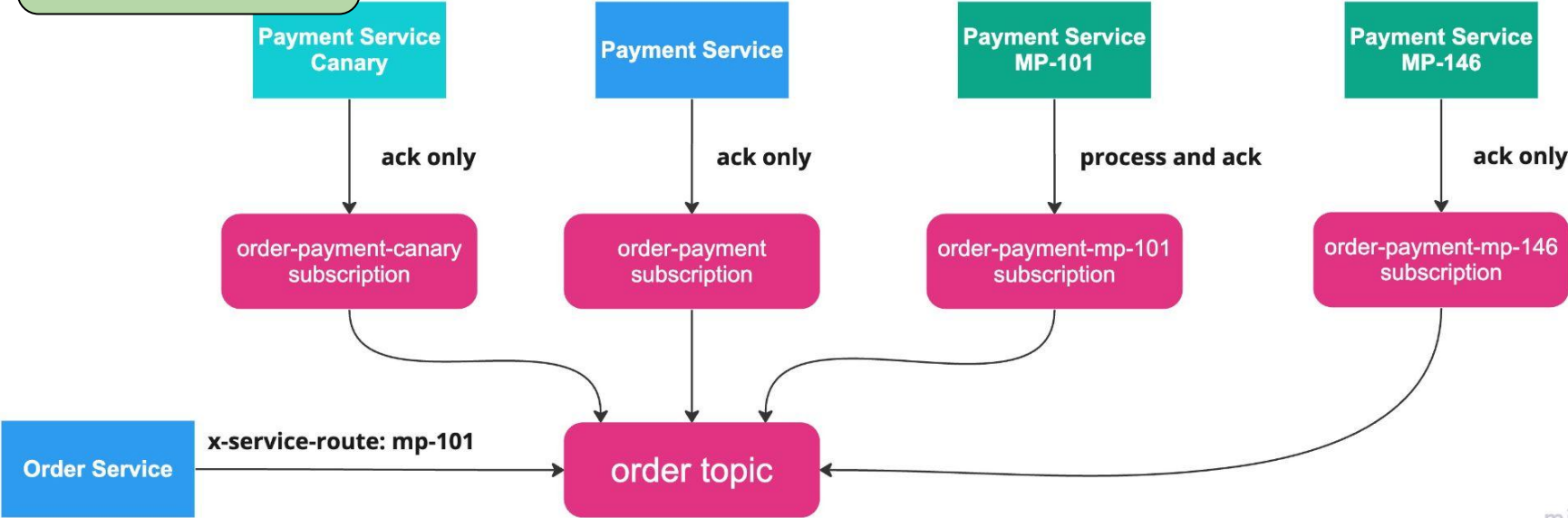


# Common Library

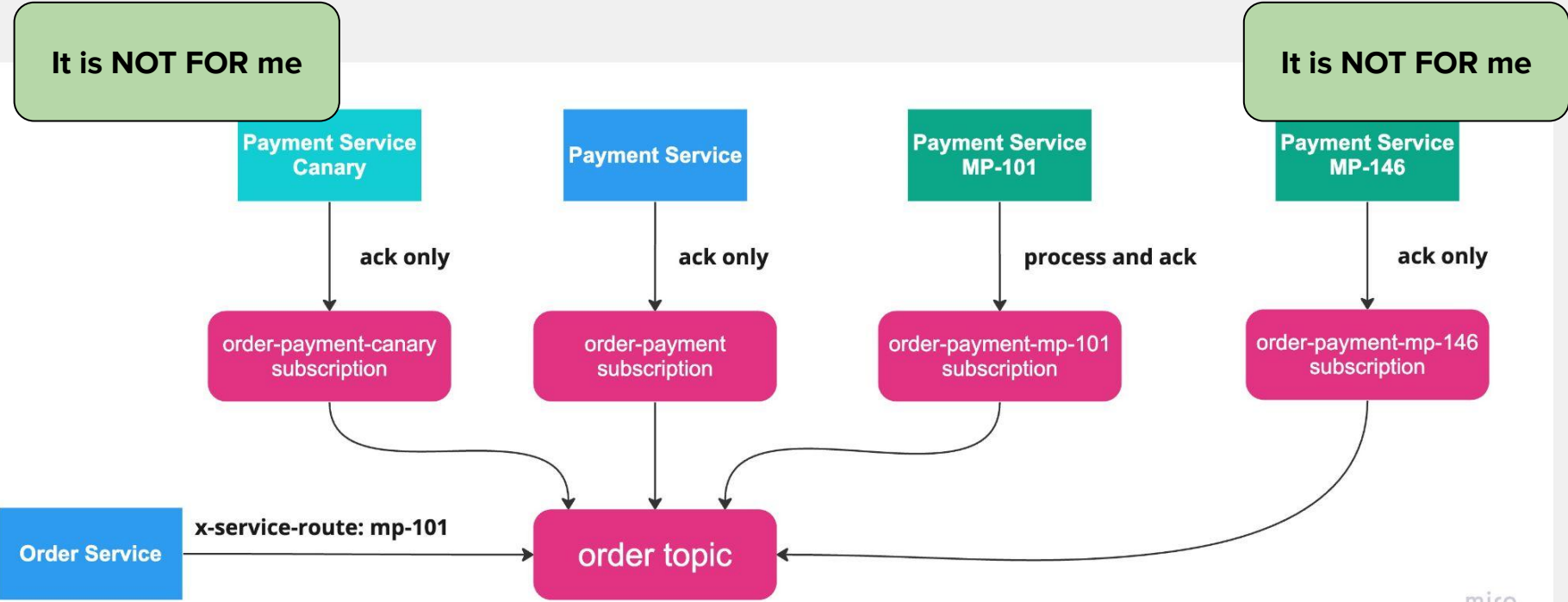


# Common Library: Decision Maker

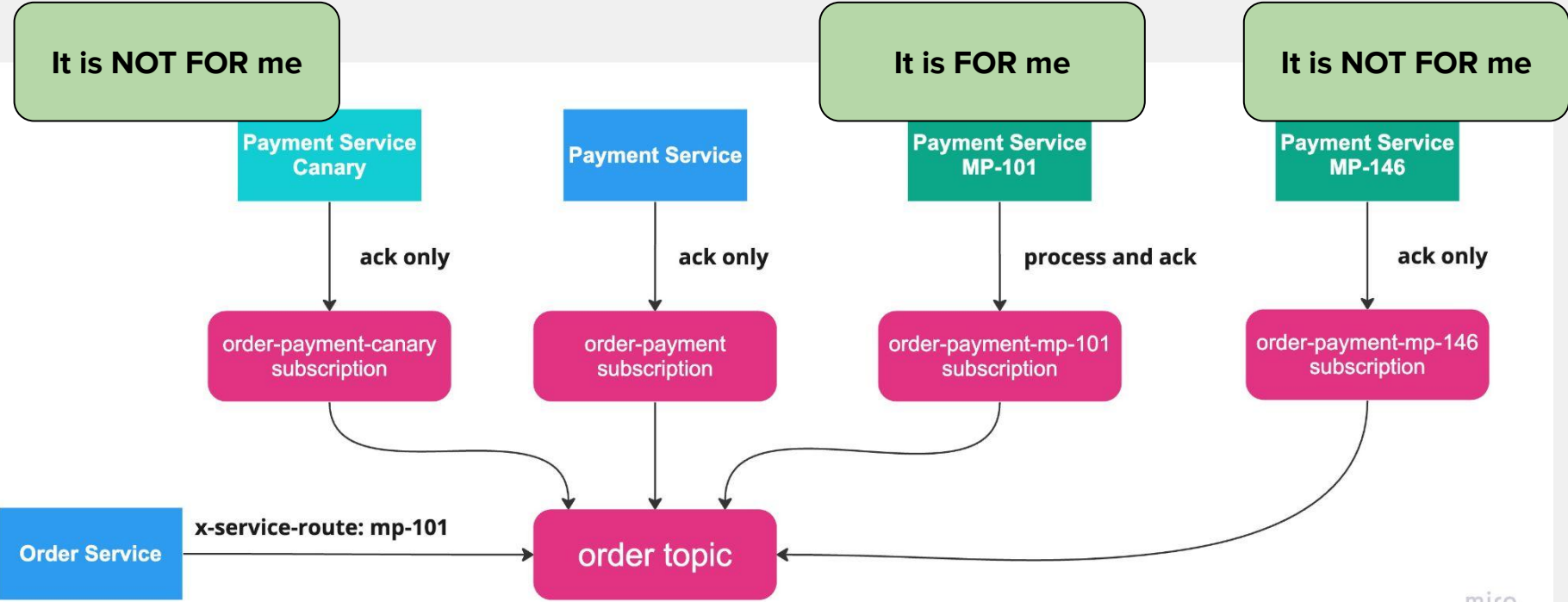
It is NOT FOR me



# Common Library: Decision Maker

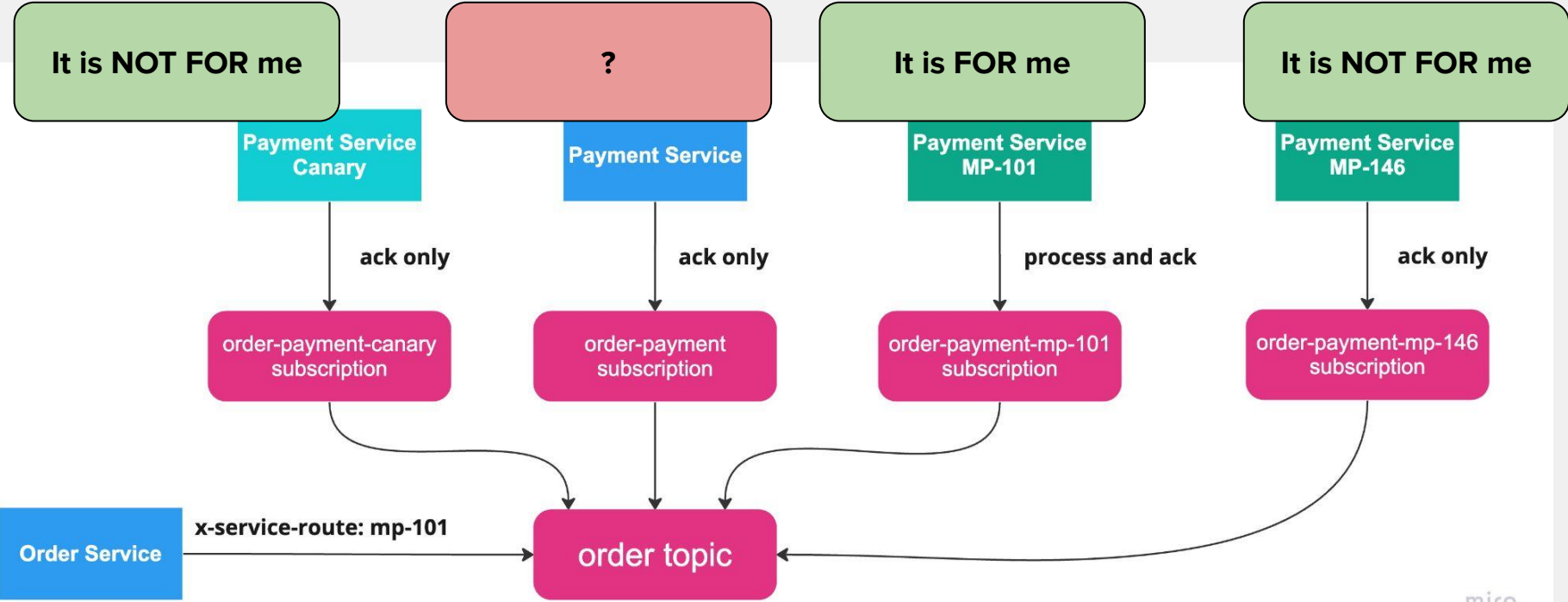


# Common Library: Decision Maker

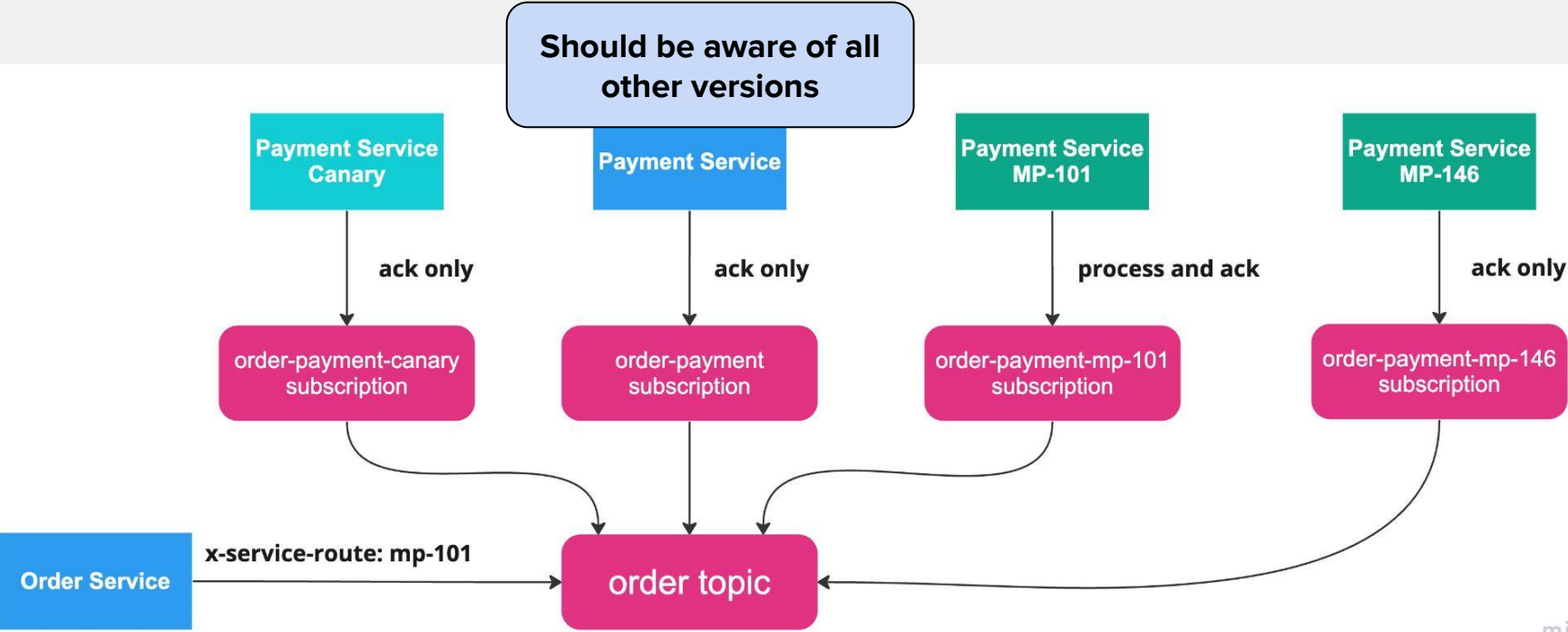




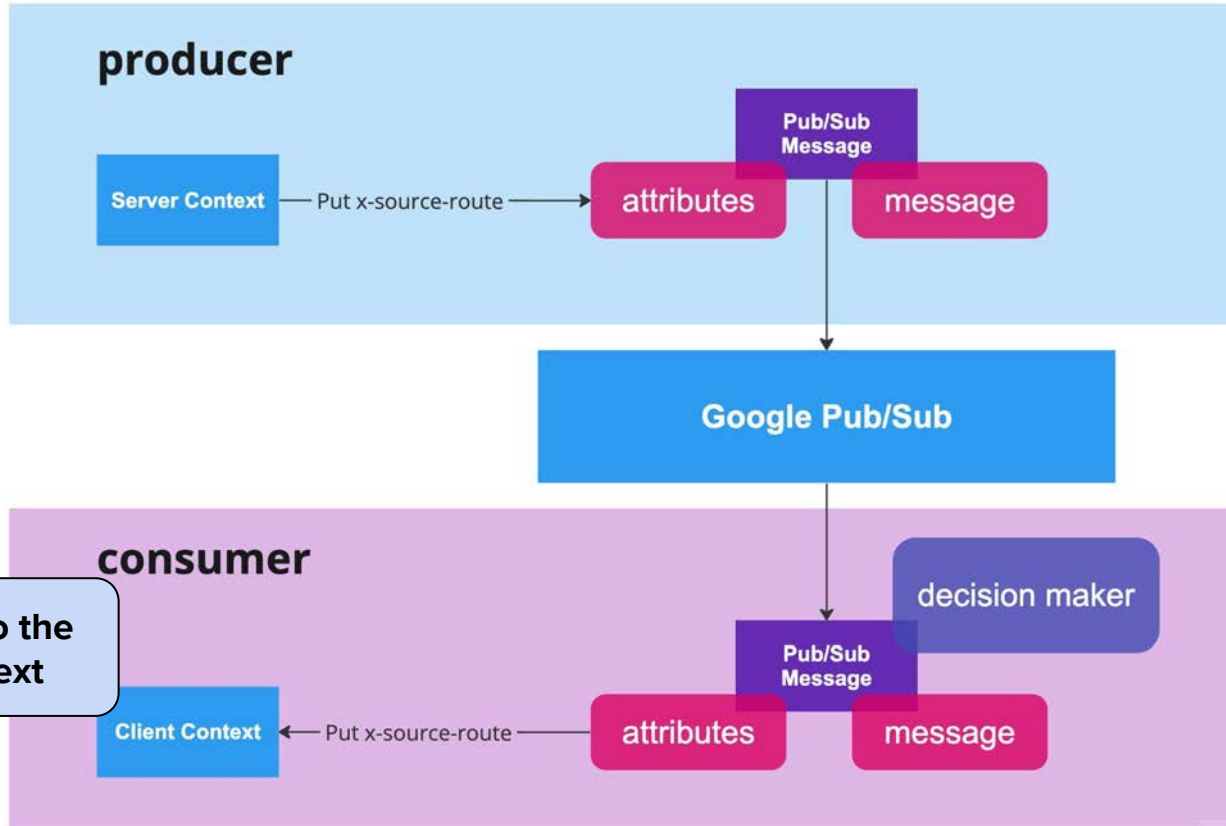
# Common Library: Decision Maker



# Common Library: Decision Maker



# Common Library

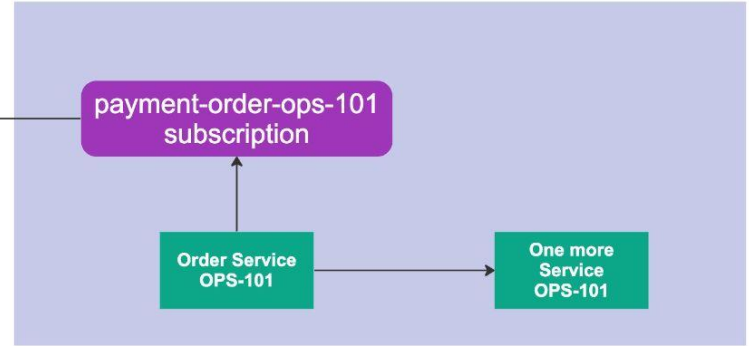


Put it back to the client context



# Async Scenarios are Unlocked

x-source-route: ops-101

stable dev



**Recall  
the issues  
to address**

- **Routing aka Service Mesh** 
- **Event Routing** 
- **Data Isolation**

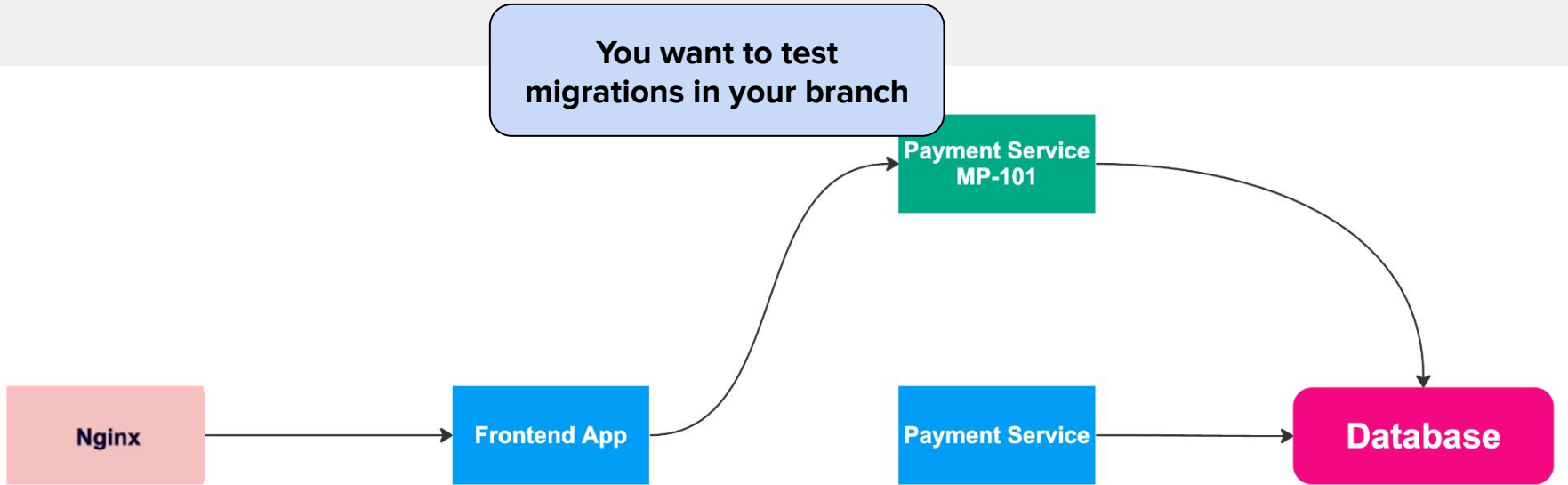
# Chapter 3: Data Isolation

**Make the Solution Safe**



# Migrations that Break

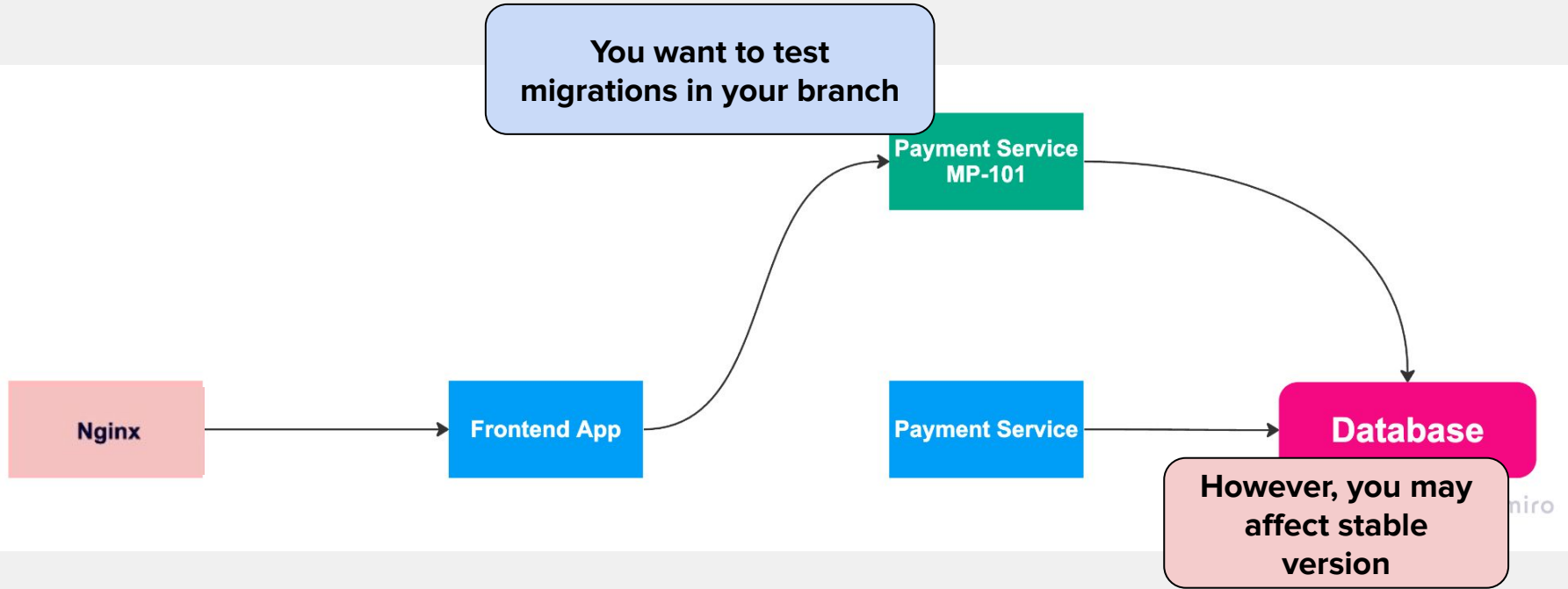
You want to test migrations in your branch



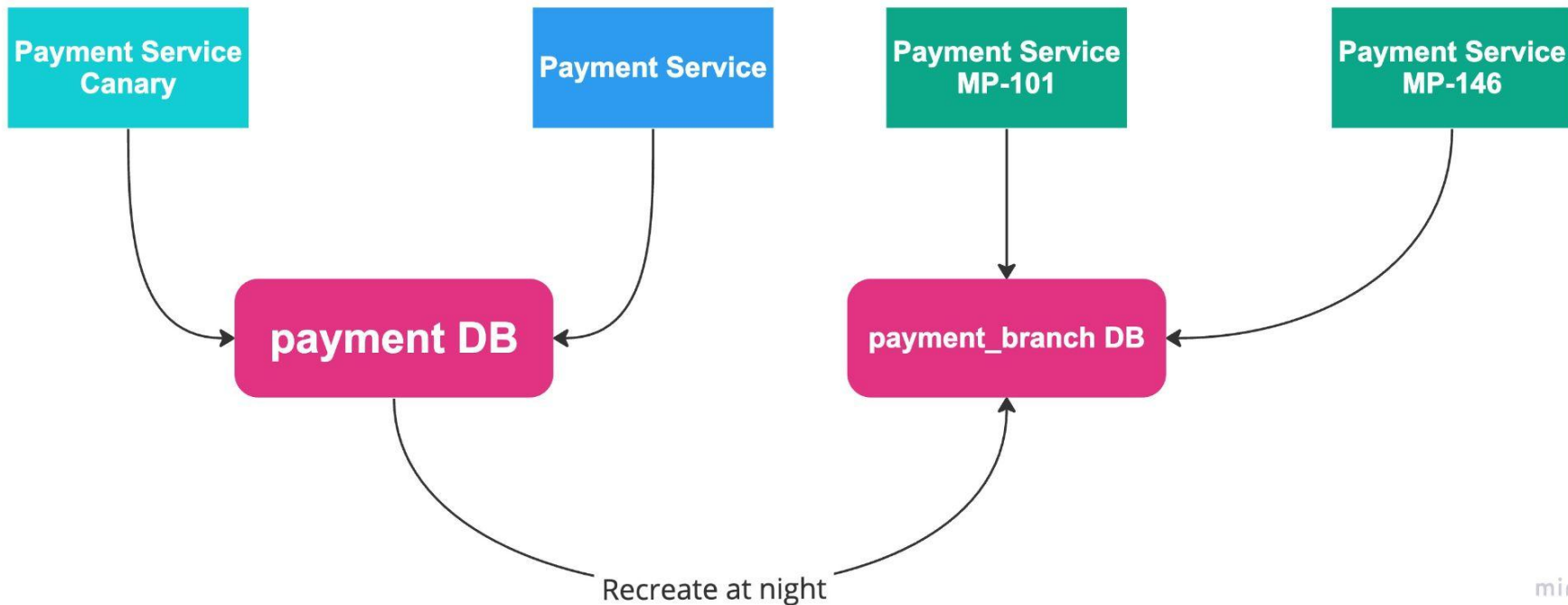
miro



# Migrations that Break

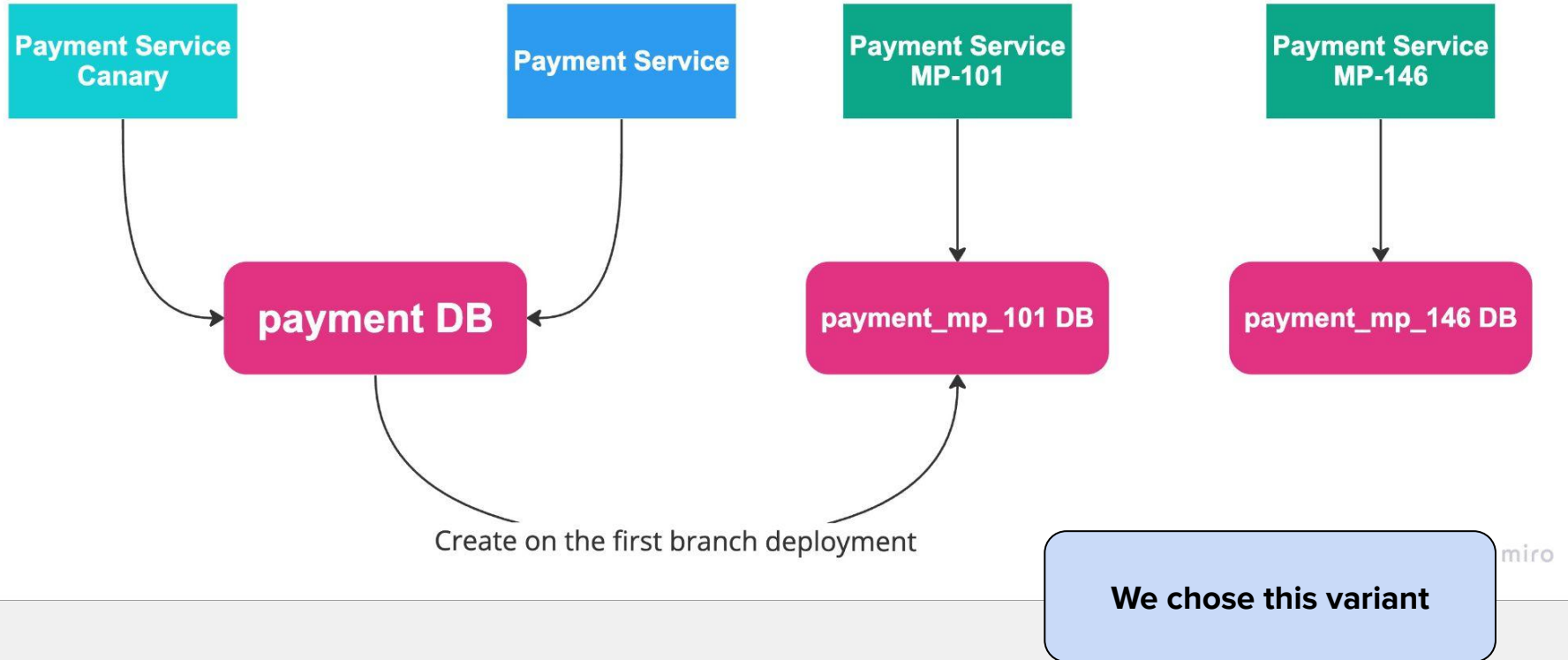


# Use Separated DB for All Branches



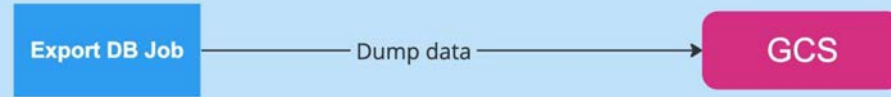
miro

# Use Separated DB per Branch



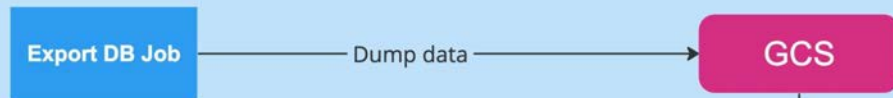
# Separated DBs Schema

nightly



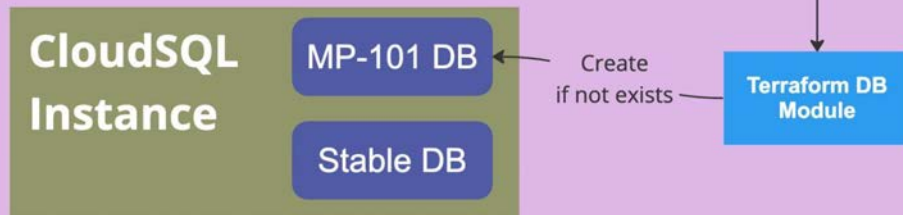
# Separated DBs Schema

**nightly**



Import data to DB




**on branch creation**



**The same  
issues to  
address**

- **Separated Redis for branches**
- **Separated DB for branches**
  - **the incomplete data**

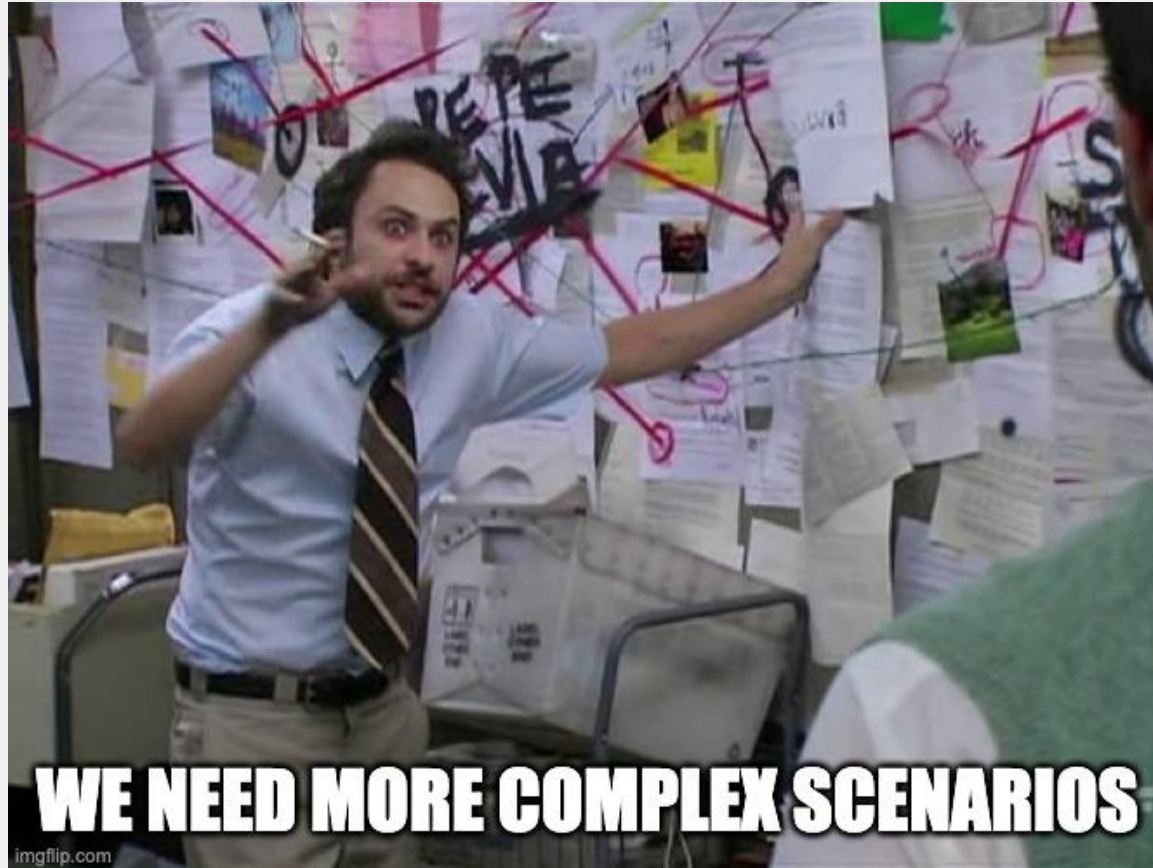
**Recall  
the issues  
to address**

- **Routing aka Service Mesh** 
- **Event Routing** 
- **Data Isolation** 

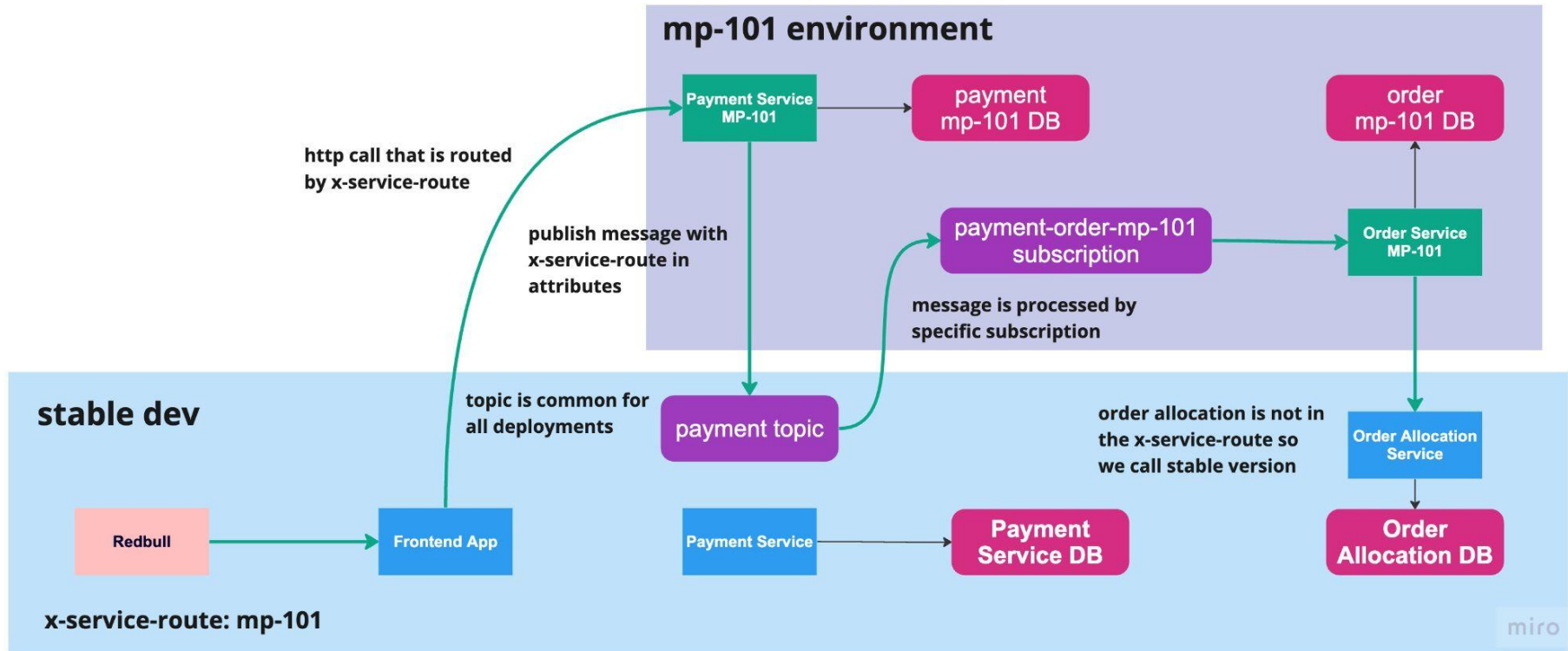
# Chapter 4: Ephemeral Environments



# Welcome to Real Life



# Welcome to Ephemeral Environments



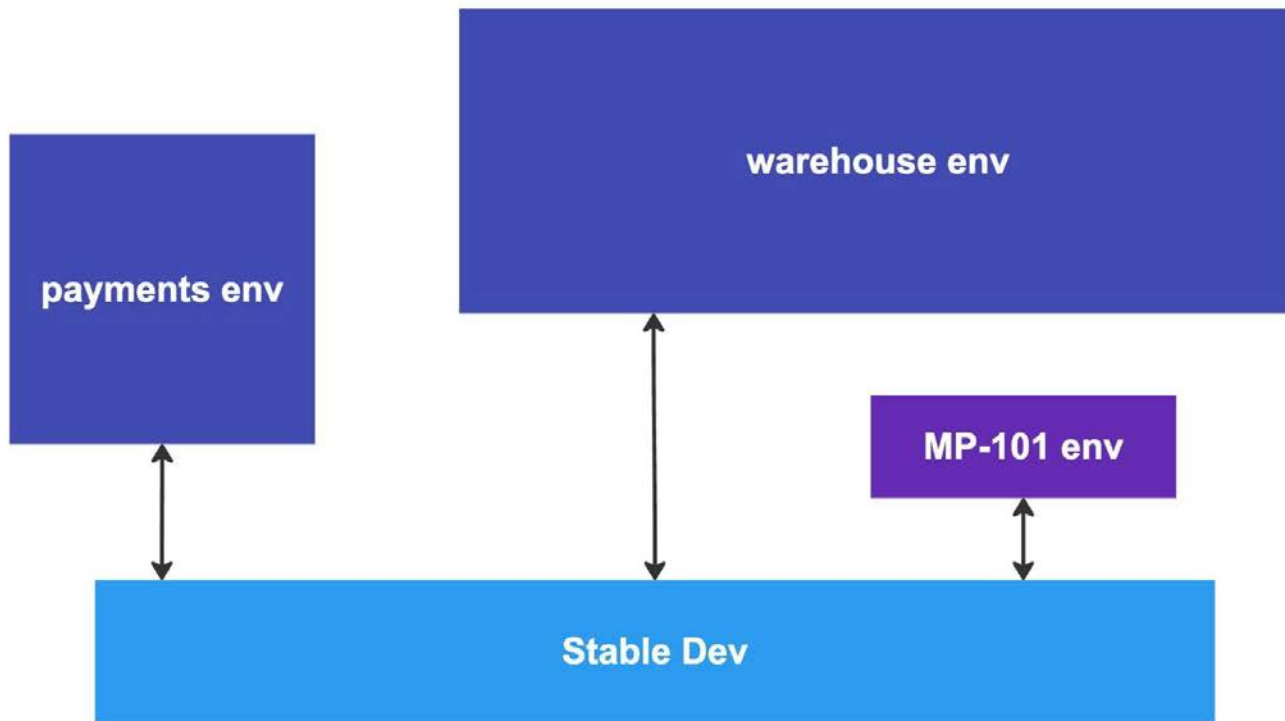
## Types of Ephemeral Environments

- **One service branch**
- **Several services branches**
  - **under one x-source-route**
  - **Jira-based**

## Types of Ephemeral Environments

- One service branch
- Several services branches
  - under one x-source-route
  - Jira-based
- Custom environments
  - for squad (payment-dev)
  - for domain (warehouse)

# Custom Ephemeral Environments



# Epilogue: Let's Reflect a Little

## **Benefits**

- **No silo between Dev and QA**
- **Low resources consumption**
- **Environments on-demand**

## **Drawbacks**

- **High cognitive load**
- **Time investments**
- **Not-fair isolation**



# Questions?



@aatarasoff



@d-ulyanov



@dmitrii-ulianov