DevOops

2023

Cocktail of Environments

How to Mix Test and Development Environments and Stay Alive

@aatarasoff

@aatarasoff

@aatarasoff

@aatarasoff


LINKERD HERO

Минздрав предупреждает

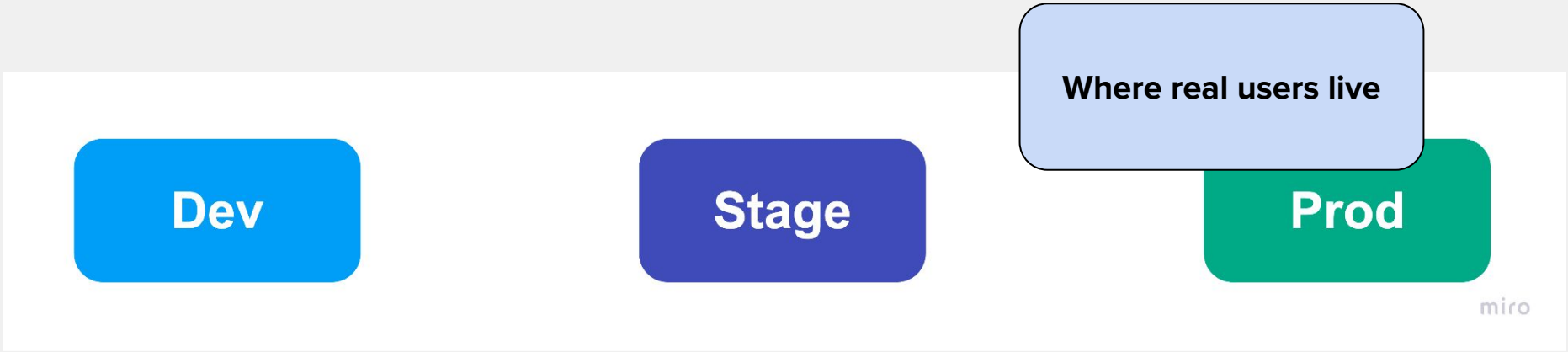
Мнение докладчика может не совпадать с официальной позицией его работодателя, коллег или других специалистов.

Все представленные в докладе сведения, примеры, выводы и другую информацию вы можете использовать на свой страх и риск. За все ваши действия ответственность несёте только вы сами.

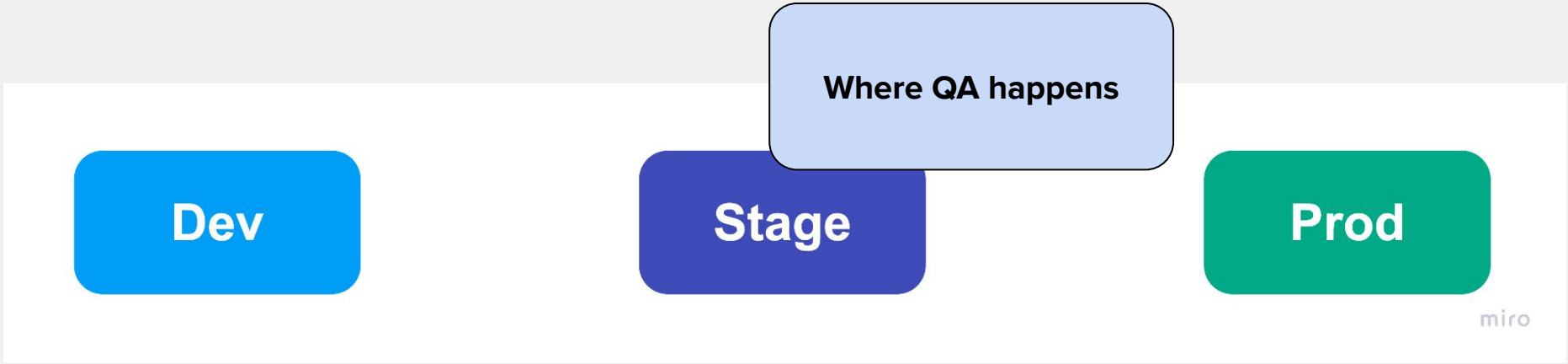
Prologue: No Good Solutions

Typical Environments

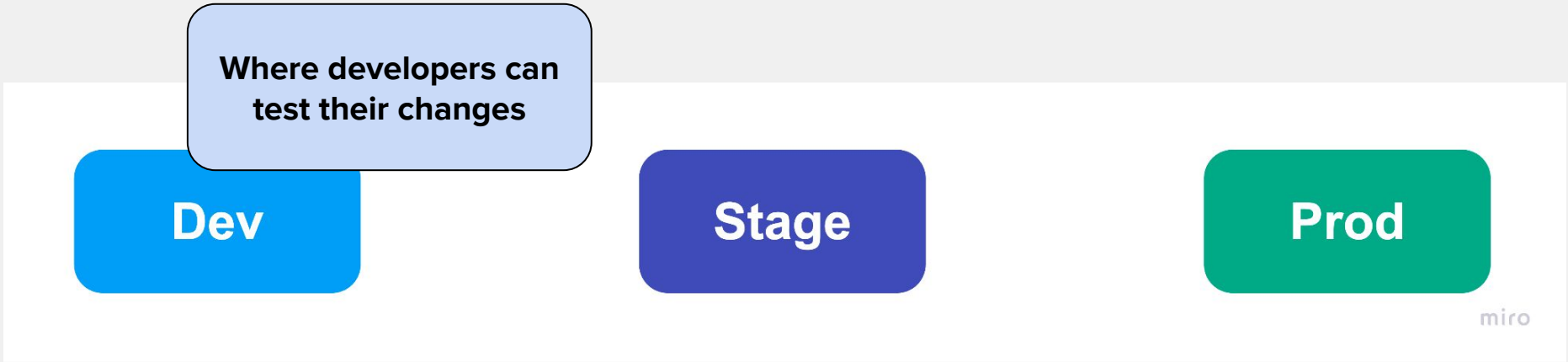
Typical Environments



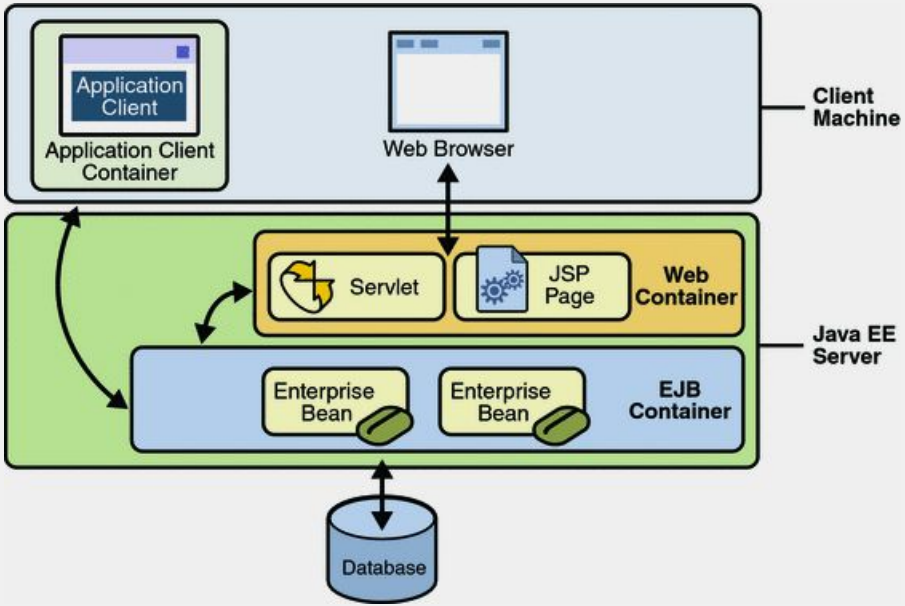
Typical Environments



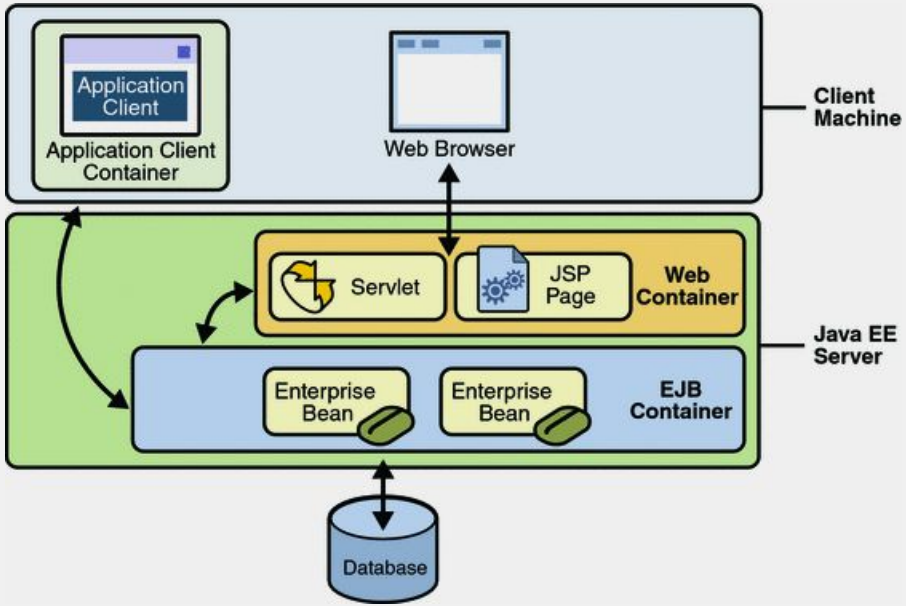
Typical Environments



No microservices - no problems

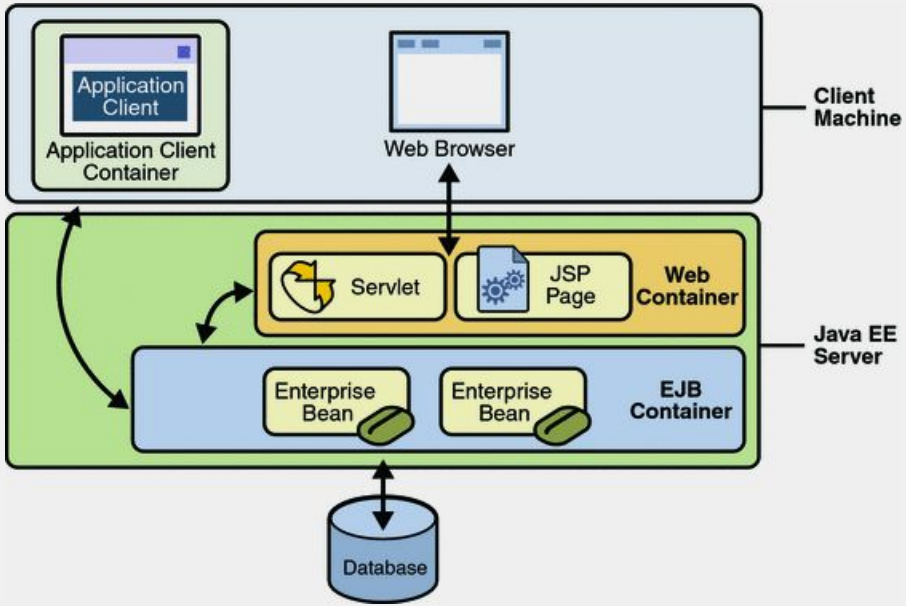


No microservices - no problems



Testing locally - easy

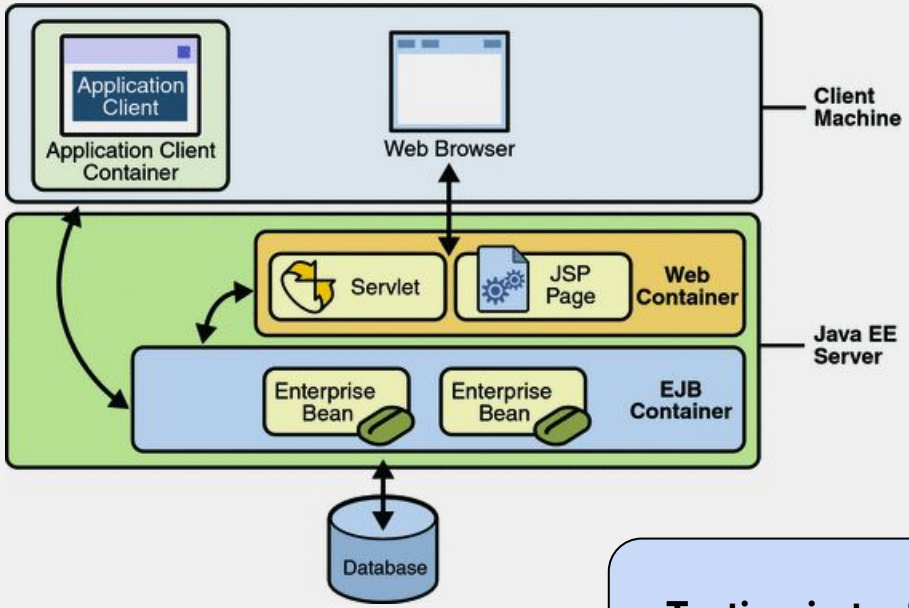
No microservices - no problems



Testing locally - easy

Testing in dev environment - easy

No microservices - no problems

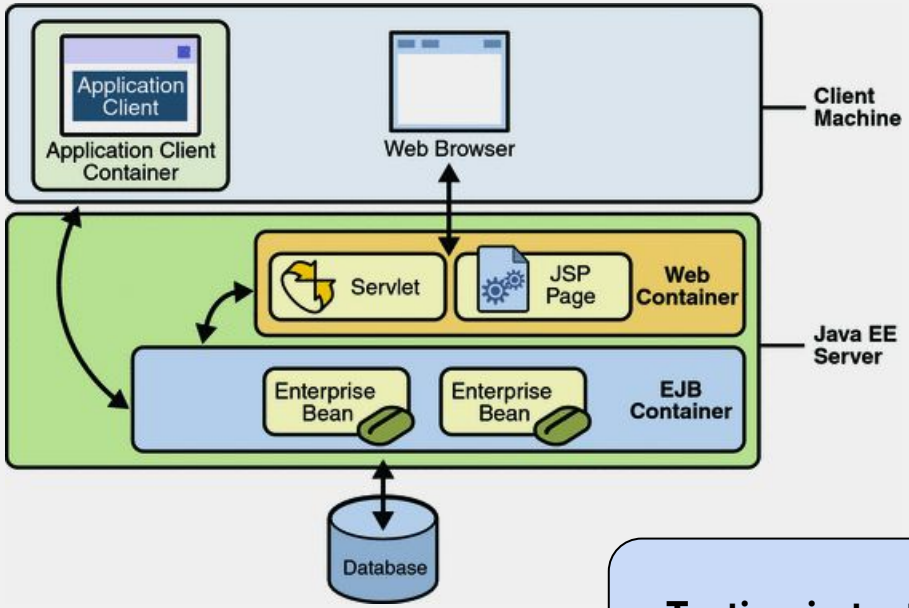


Testing locally - easy

Testing in dev environment - easy

Testing in test/stage environment - easy

No microservices - no problems



Testing locally - easy

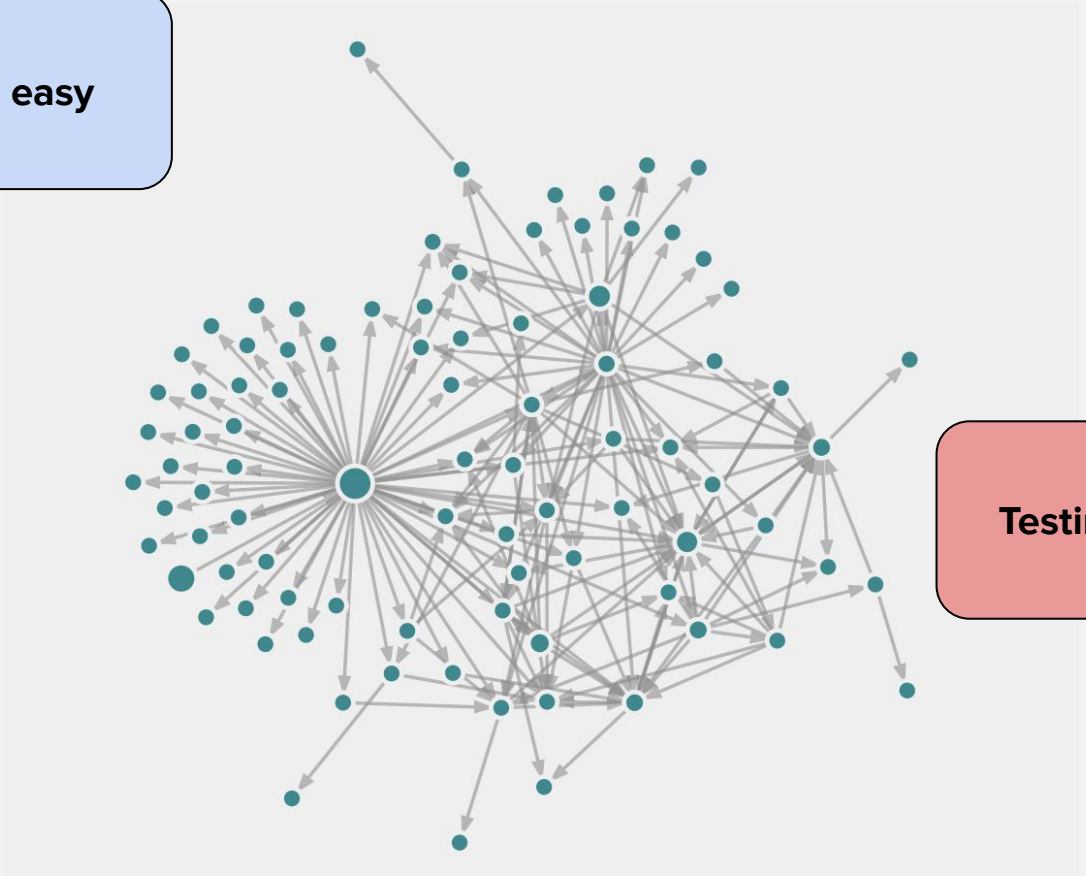
Testing in dev environment - easy

Testing in test/stage environment - easy


MICROSERVICES
MICROSERVICES EVERYWHERE
imgflip.com

Testing one - easy

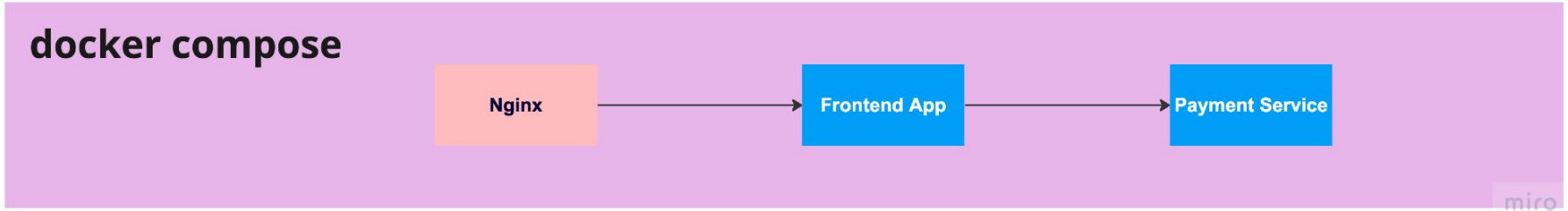
Testing one - easy



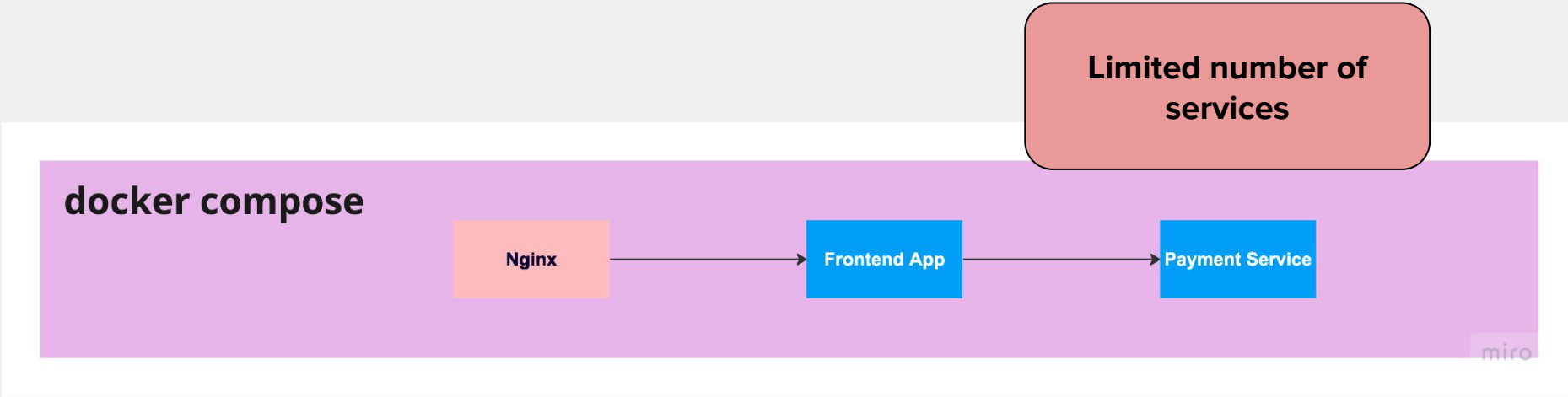
Testing many - hard

Chapter 1: Baking Dev Environment

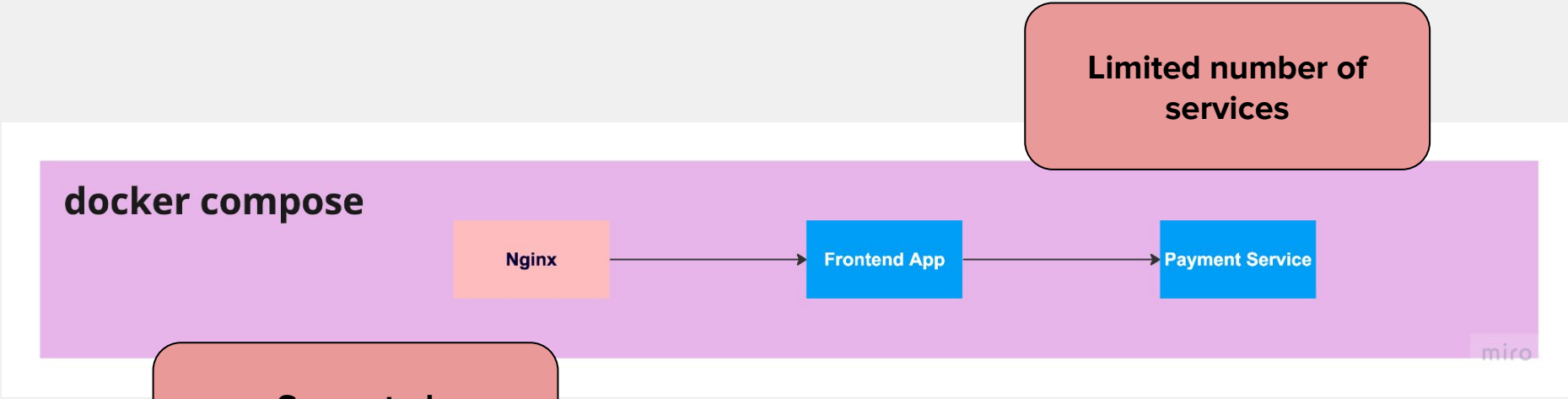
Work on my machine



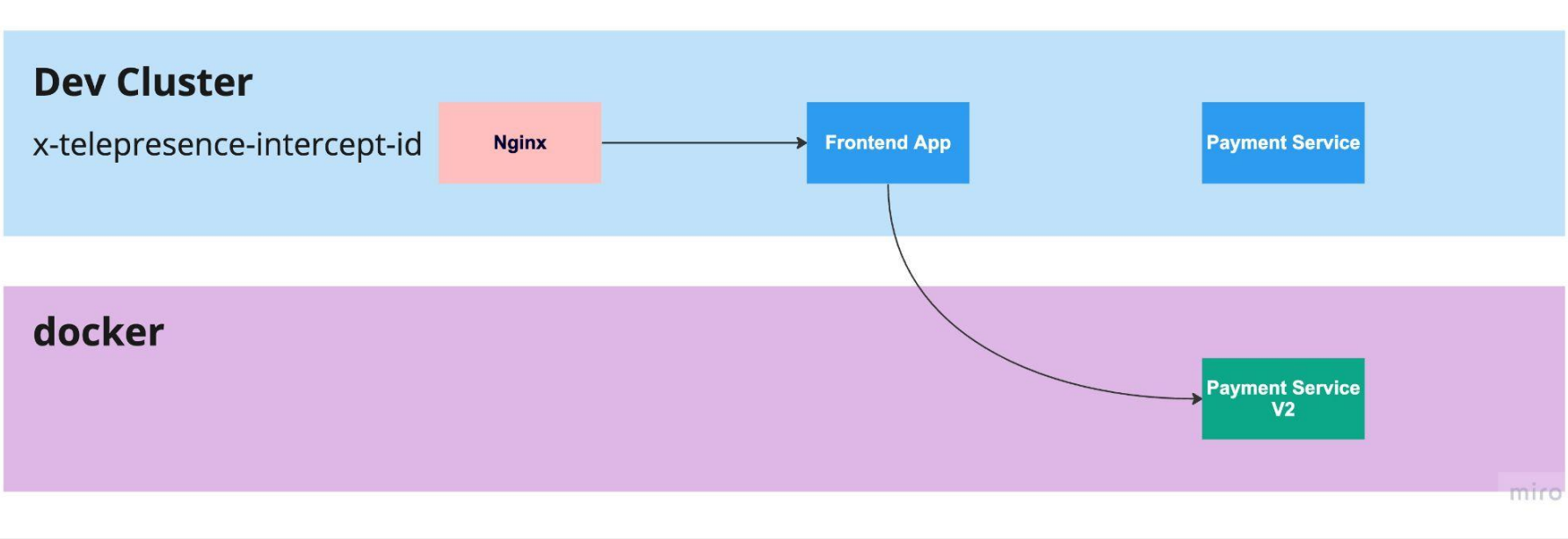
Work on my machine



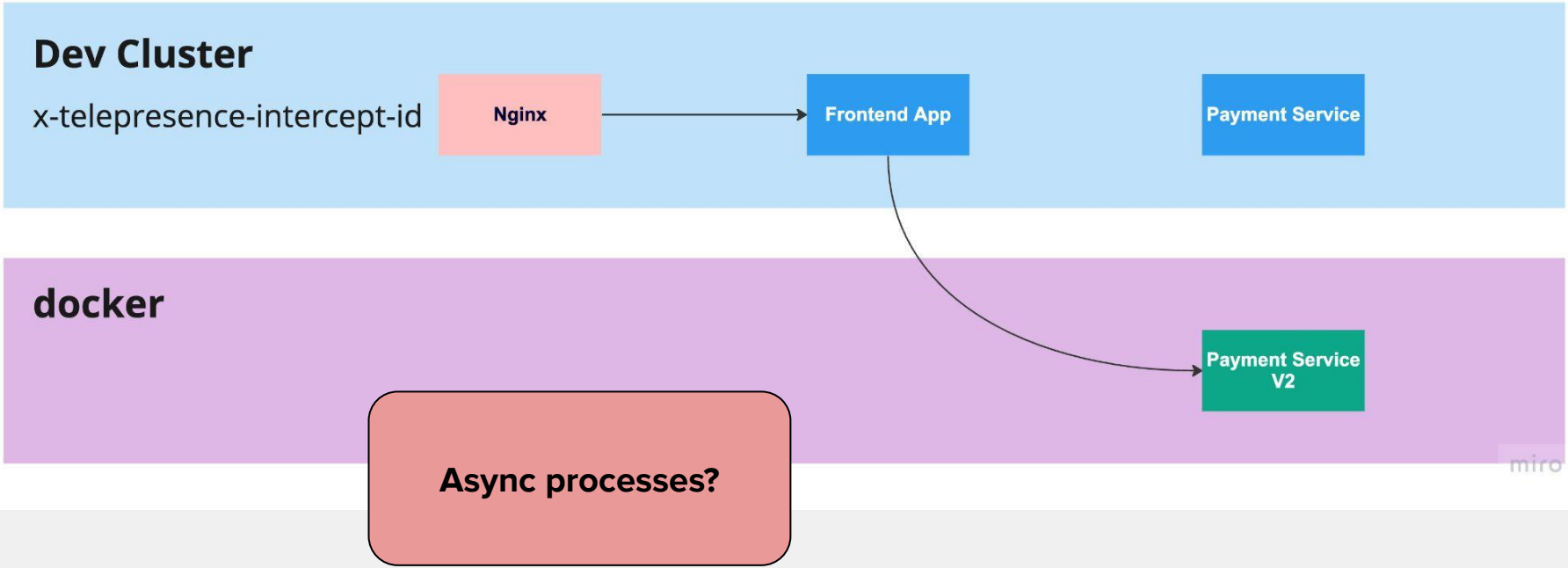
Work on my machine



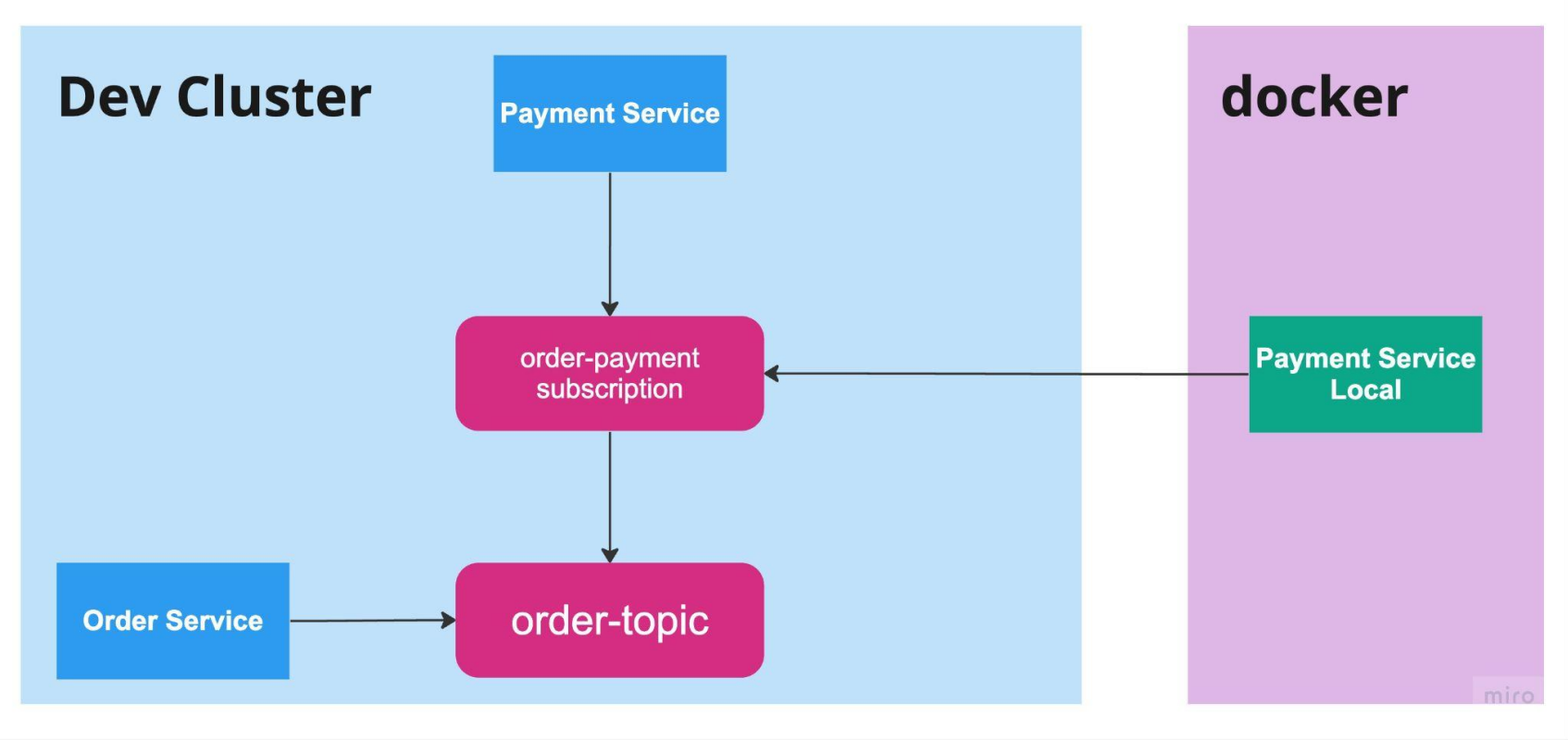
Telepresence



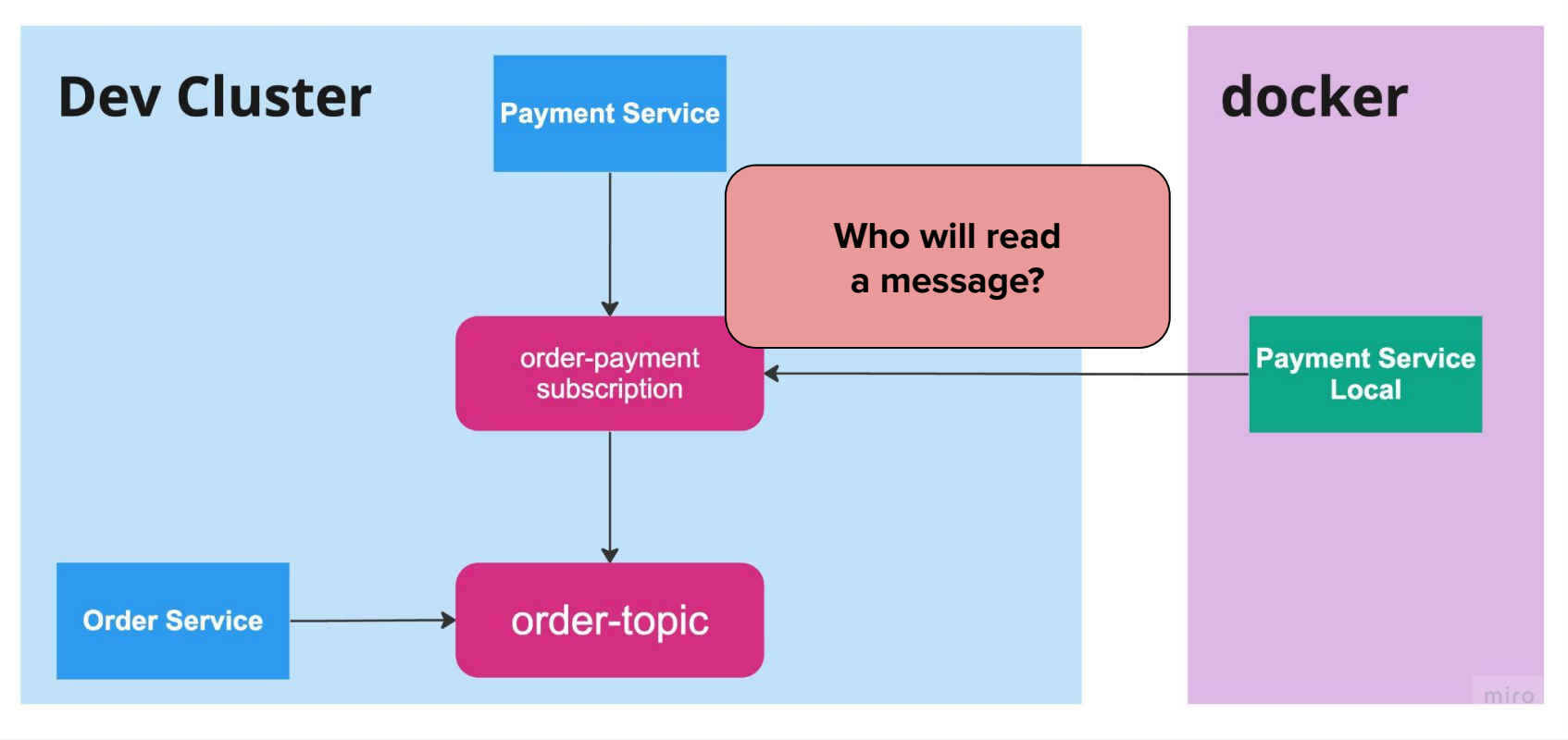
Telepresence



Local Async Process Testing



Local Async Process Testing



Local Only

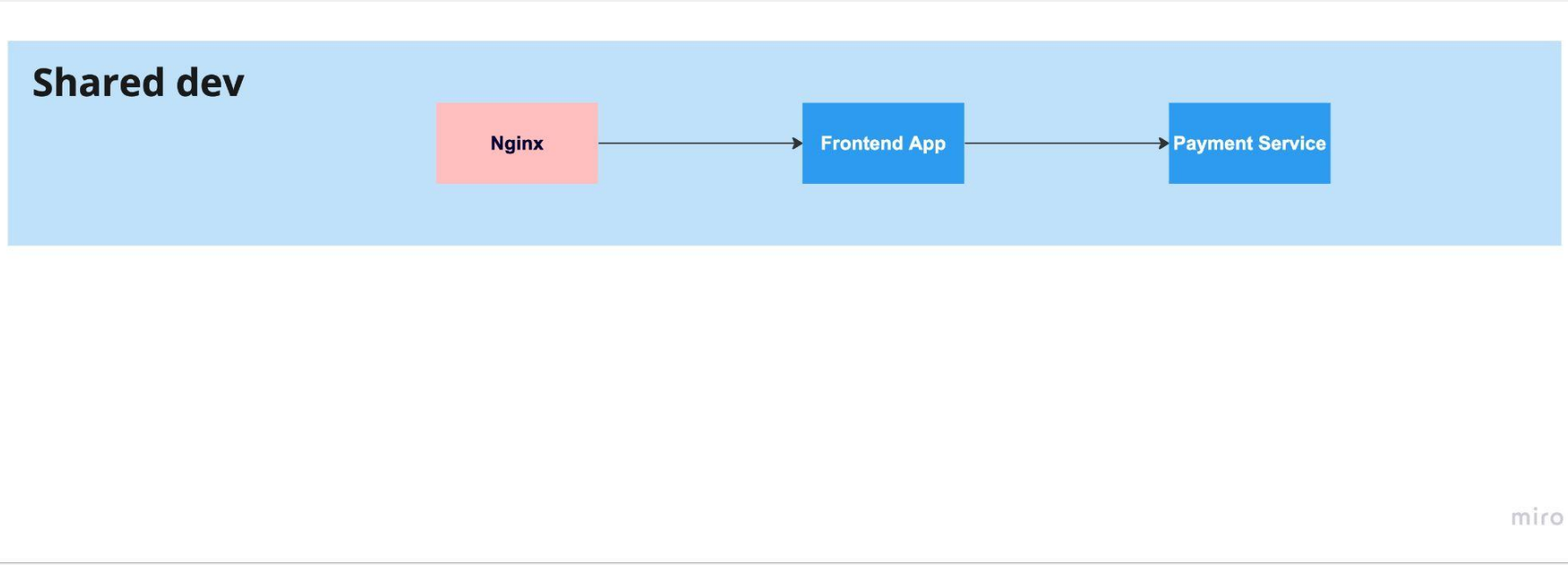
Benefits

- Fast feedback loop
- Good for simple usecases

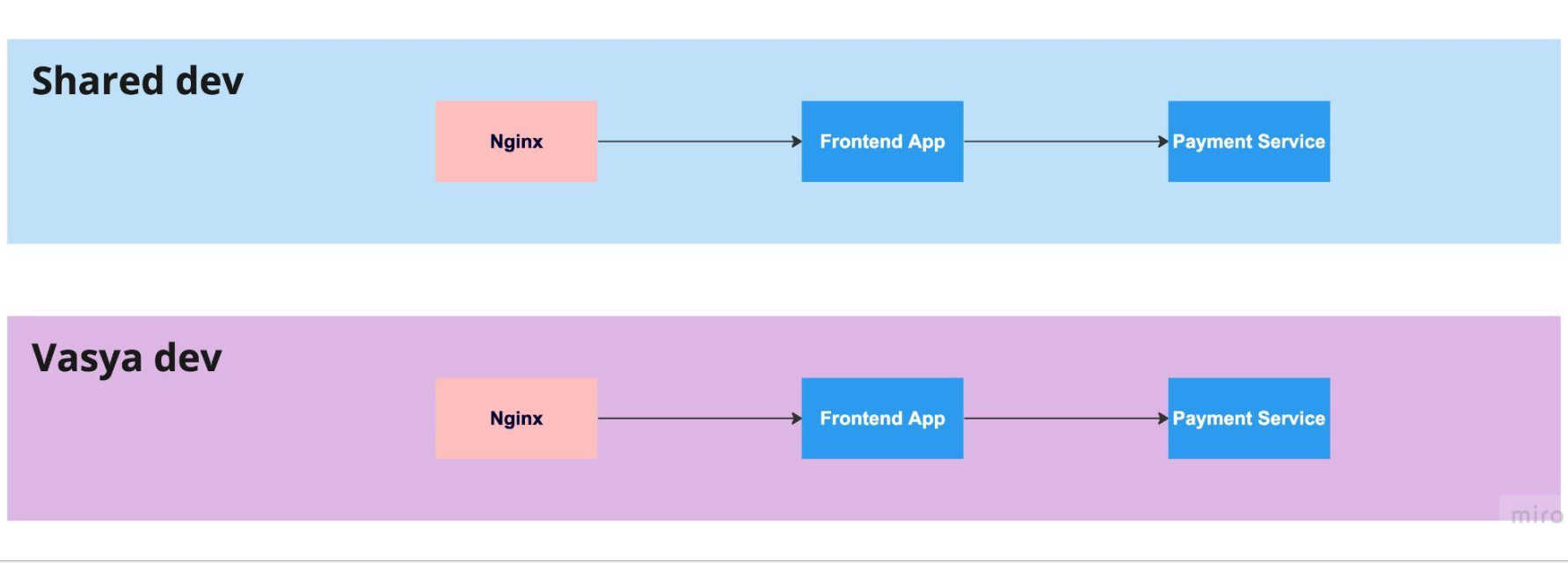
Drawbacks

- Hard to test complex scenarios
- Hard to collaborate
  - QA
  - other developers

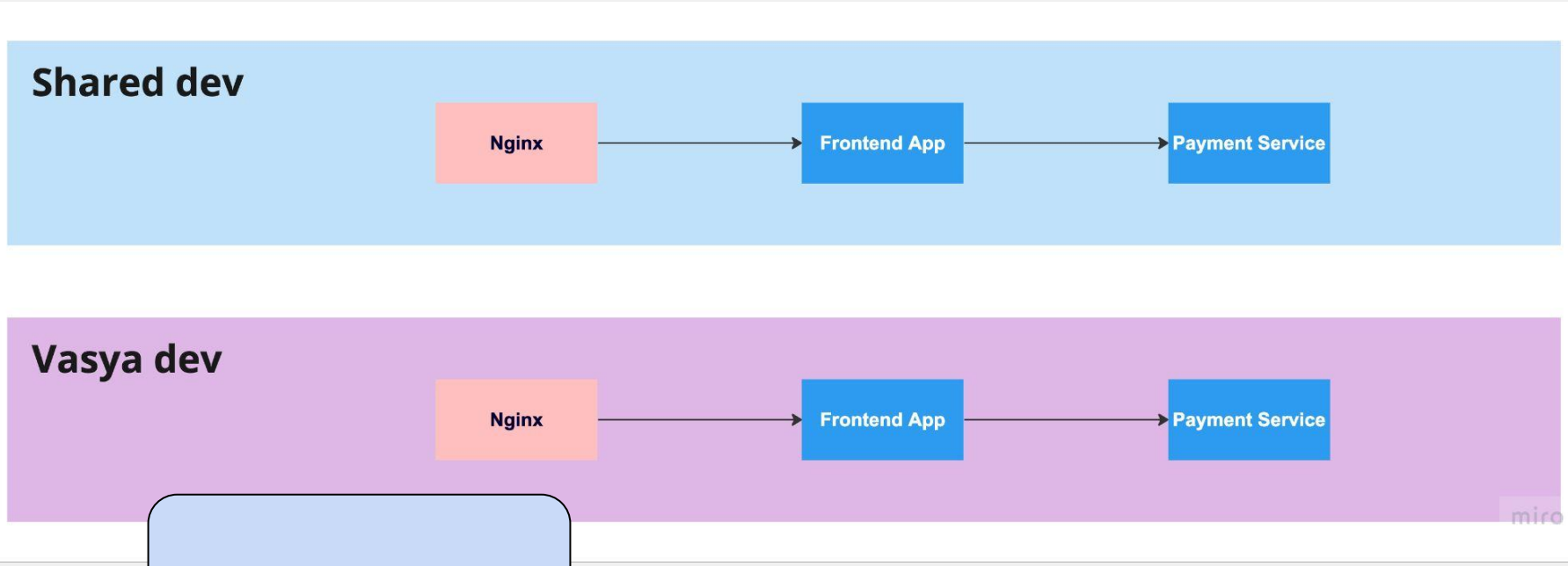
Full Copy for each Developer



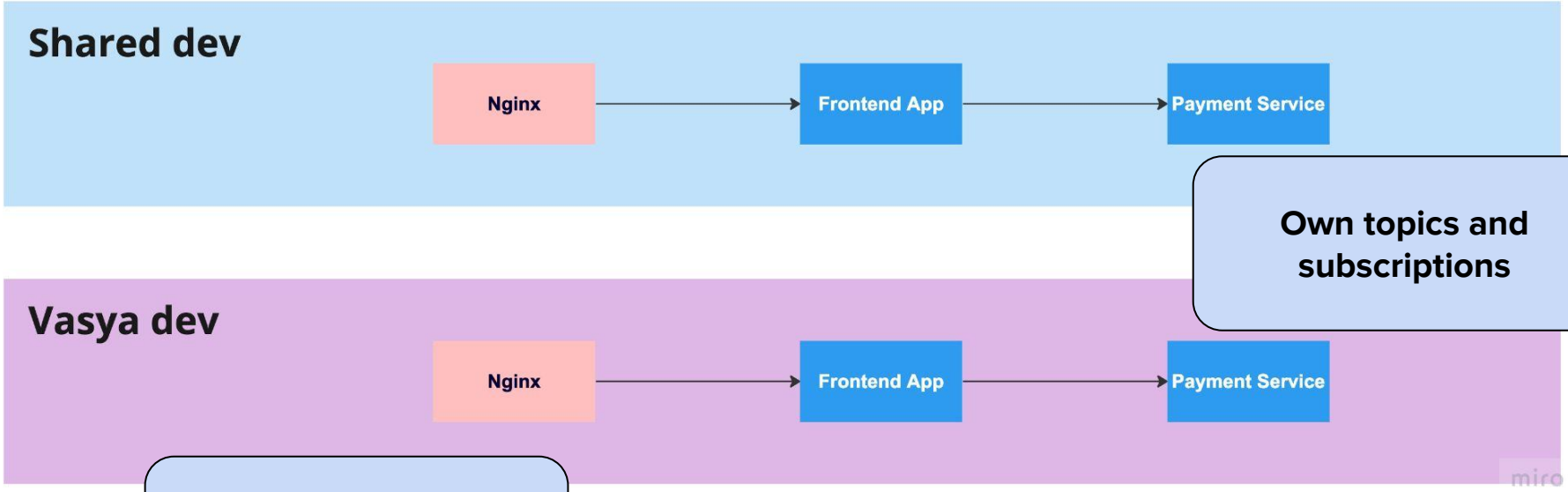
Full Copy for each Developer



Full Copy for each Developer



Full Copy for each Developer



- Tool to establish “Vasya” env
  - 10-15 minutes max

Issues to address

Issues to address

- Tool to establish “Vasya” env
  - 10-15 minutes max
- Handle the load
  - 10k workloads and much more

Issues to address

- Tool to establish “Vasya” env
  - 10-15 minutes max
- Handle the load
  - 10k workloads and much more
- Support separated infra components
  - dbs in containers
  - emulators for queues

- Part copy instead of full
  - core services first
  - specific services on-demand

Optimizations

Optimizations

- Part copy instead of full
  - core services first
  - specific services on-demand
- Shutdown every night

Optimizations

- Part copy instead of full
  - core services first
  - specific services on-demand
- Shutdown every night
- Env per squad, not developer

Optimizations

- Part copy instead of full
  - core services first
  - specific services on-demand
- Shutdown every night
- Env per squad, not developer
- Get rid of X
  - do not use service mesh
  - do not keep logs

Full Copy for each Developer

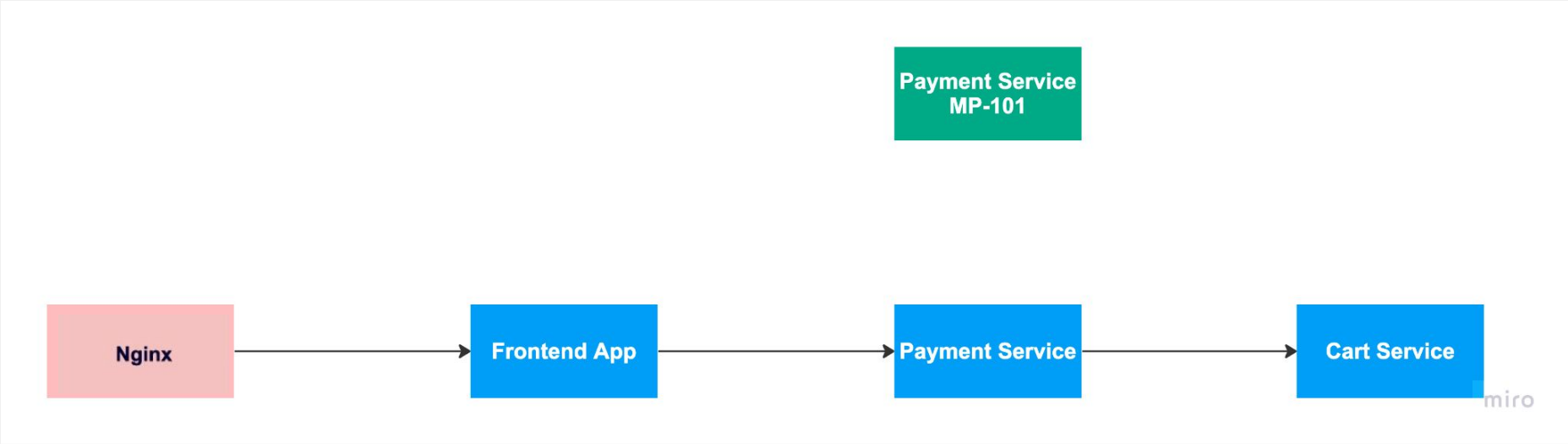
Benefits

- Full isolation
- The cognitive load is low

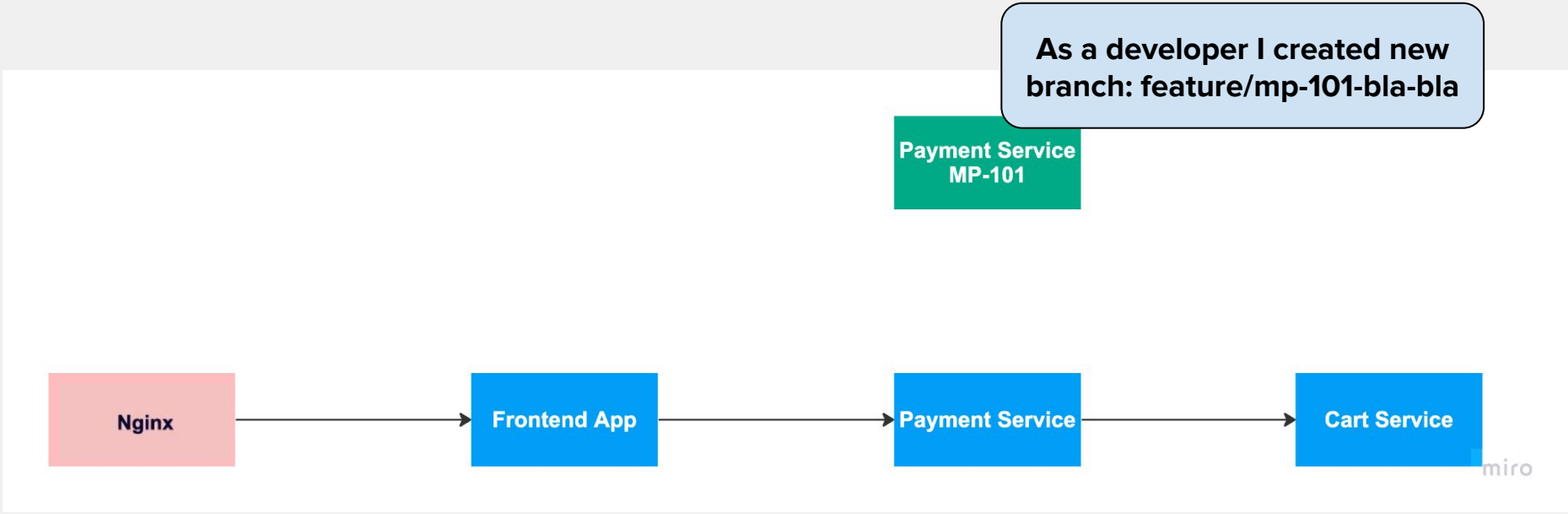
Drawbacks

- Custom configuration
- High resources consumption
- You own - you troubleshooting

Service Injection

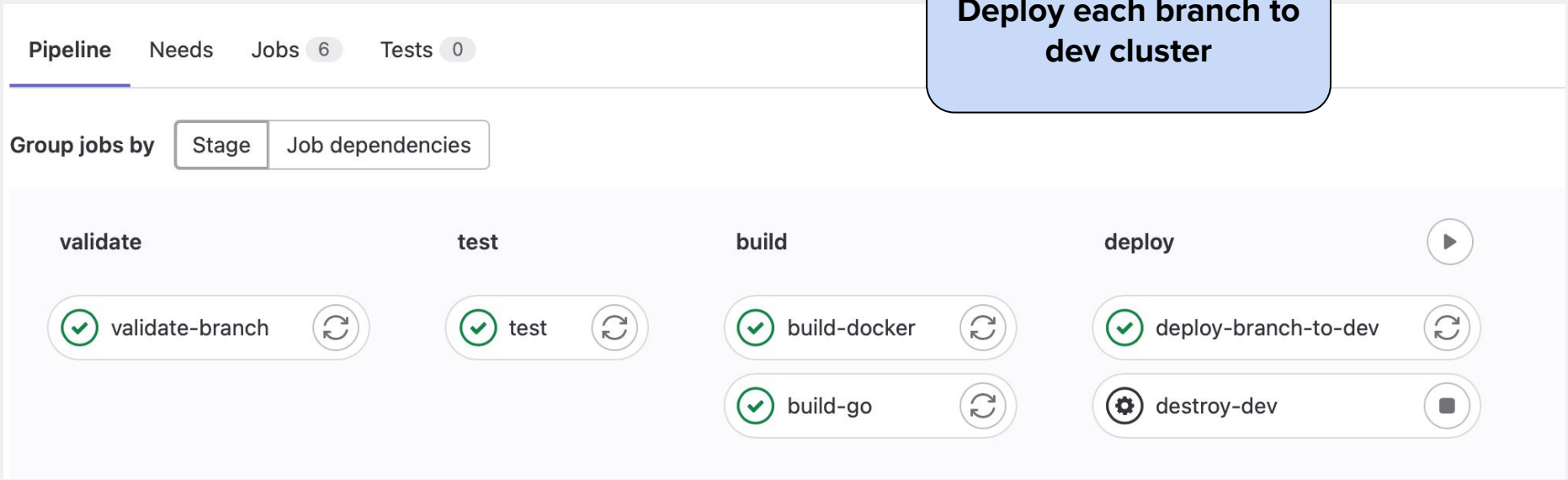


Service Injection

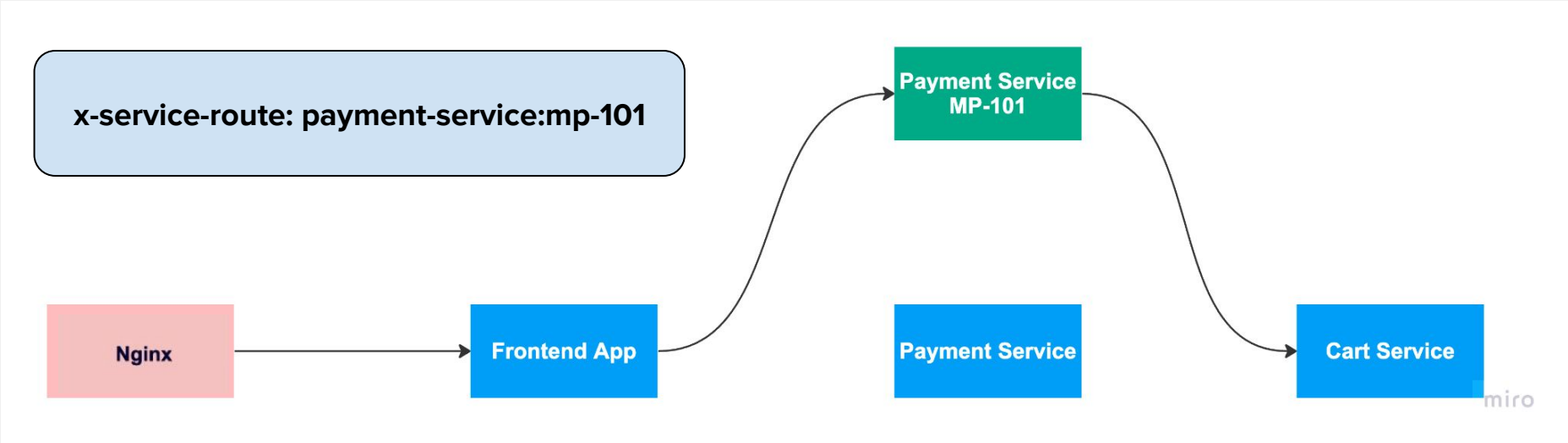


Service Injection

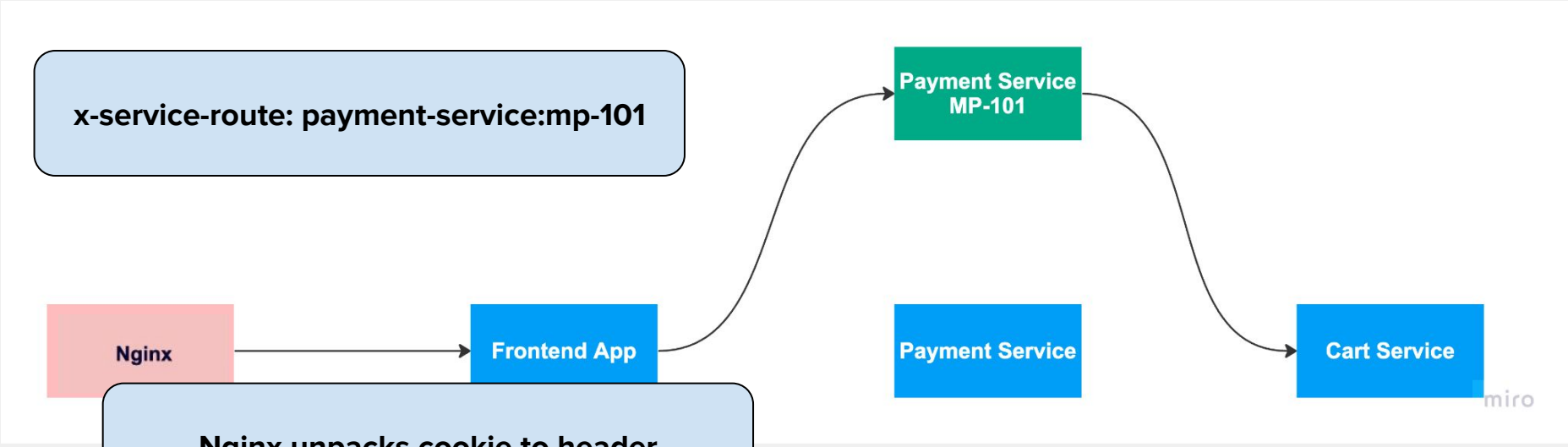
Deploy each branch to dev cluster



Service Injection

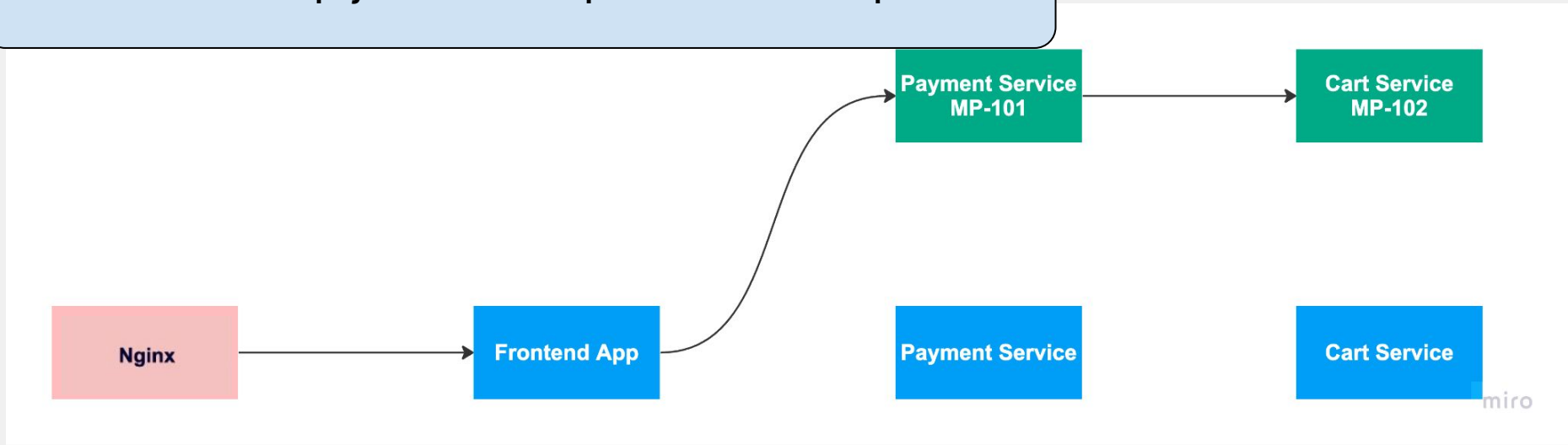


Service Injection

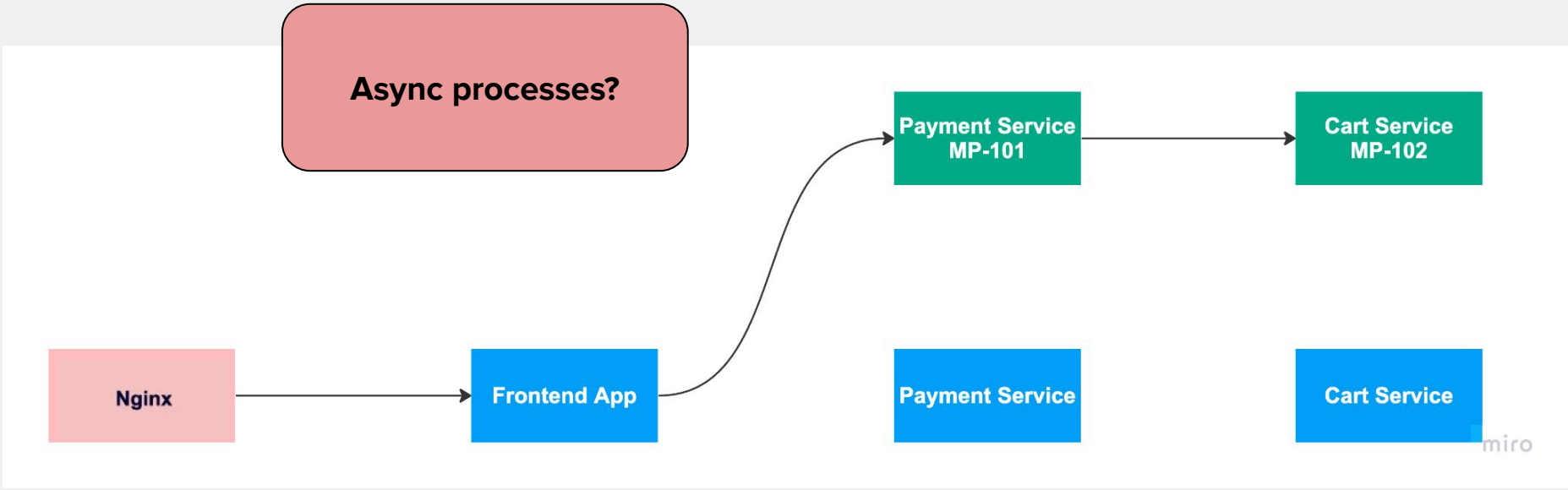


We Need More Branches

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



We Need More Branches



Service Injection

Benefits

- Low resources consumption
- Less troubleshooting required
- Convinient collaboration

Drawbacks

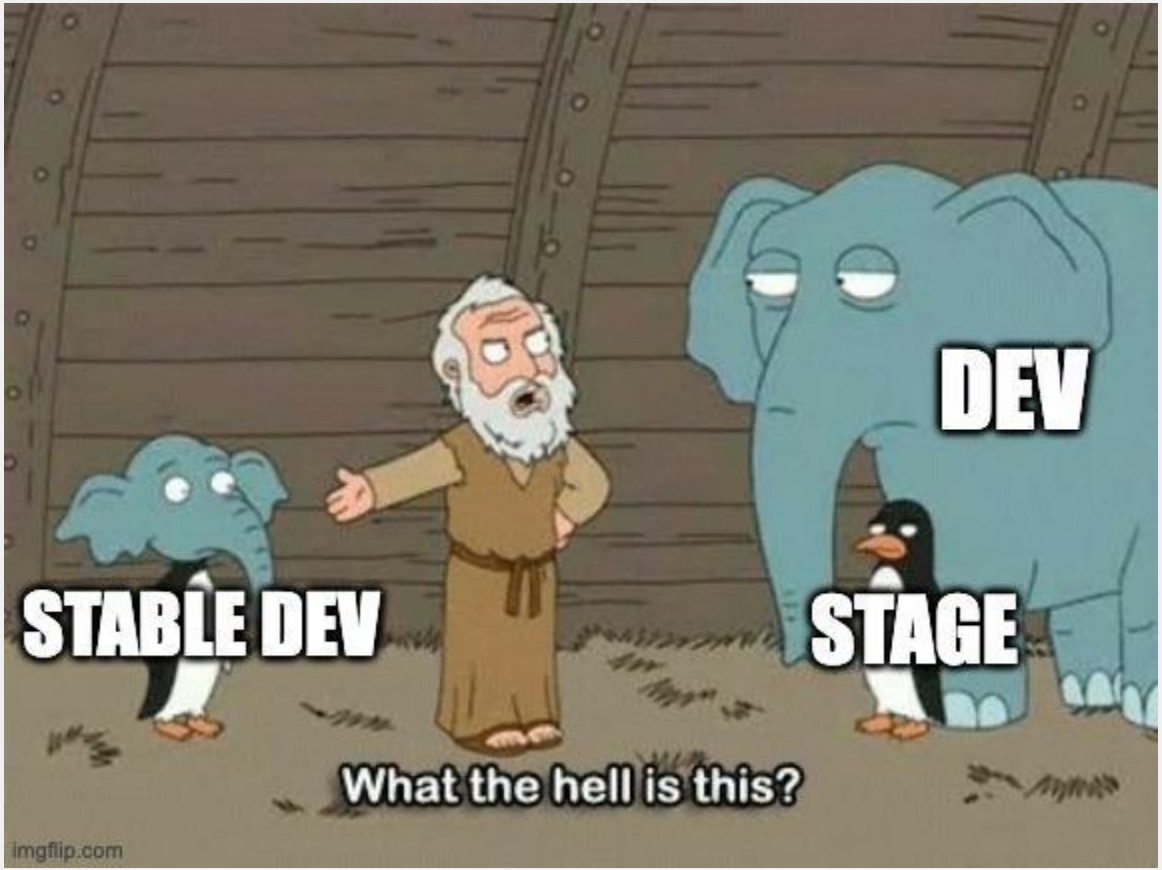
- Shared resources - poor isolation
- Hard to test async processes

Chapter 2: What are You, Stable Dev?

Typical Environments

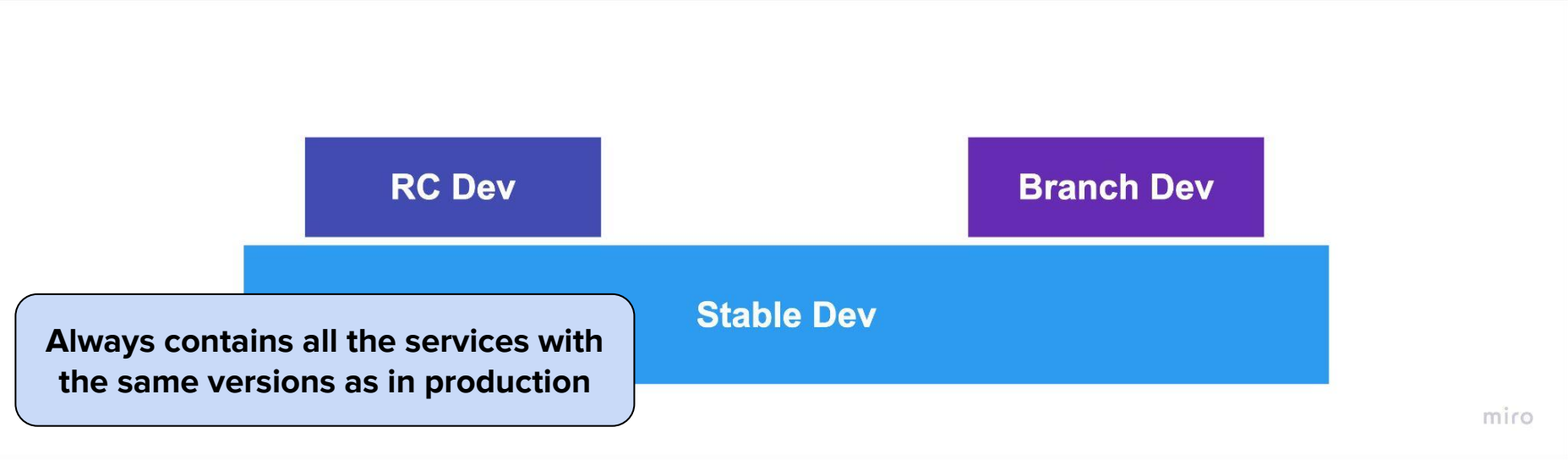
Atypical Environments

Atypical Environments

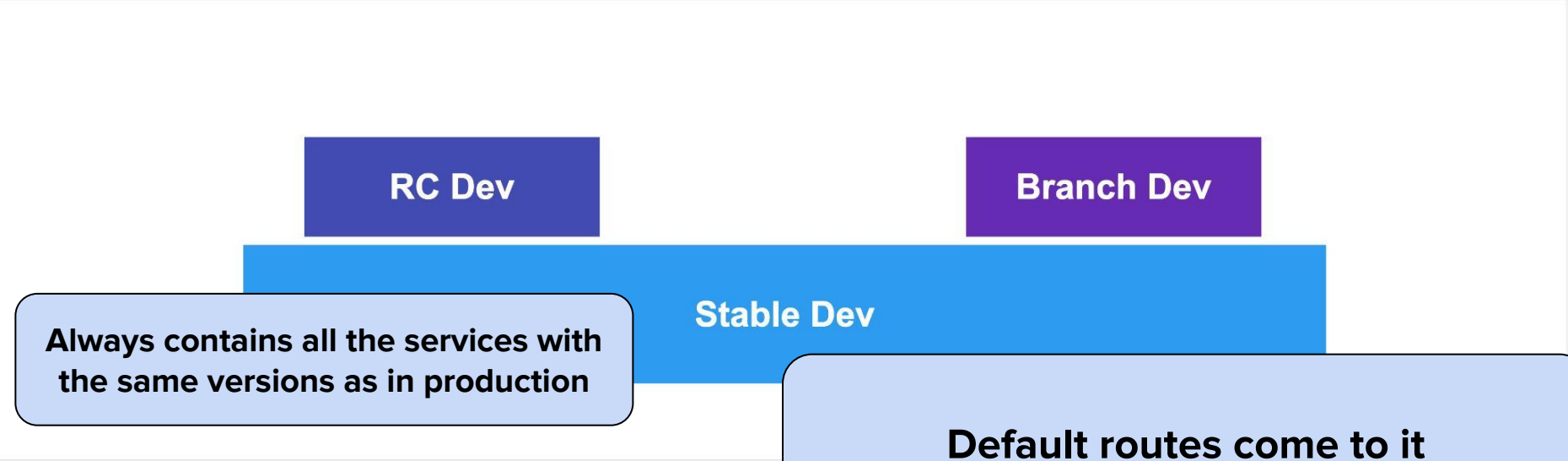


One Cluster - Several Environments

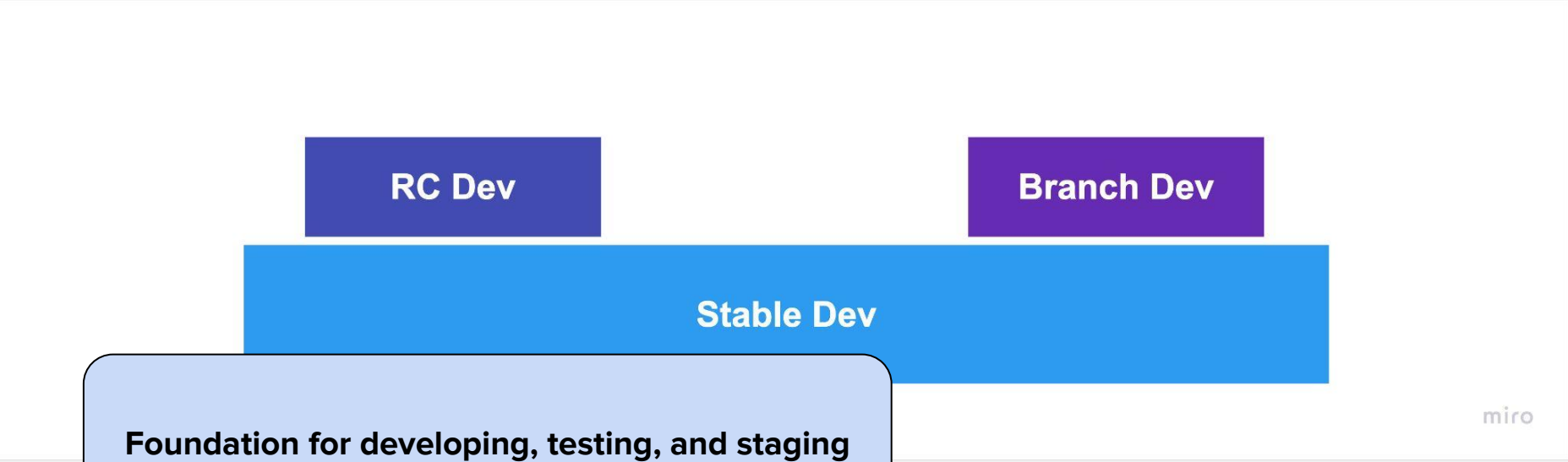
Stable Dev


RC Dev
Branch Dev
Stable Dev
Always contains all the services with the same versions as in production
miro

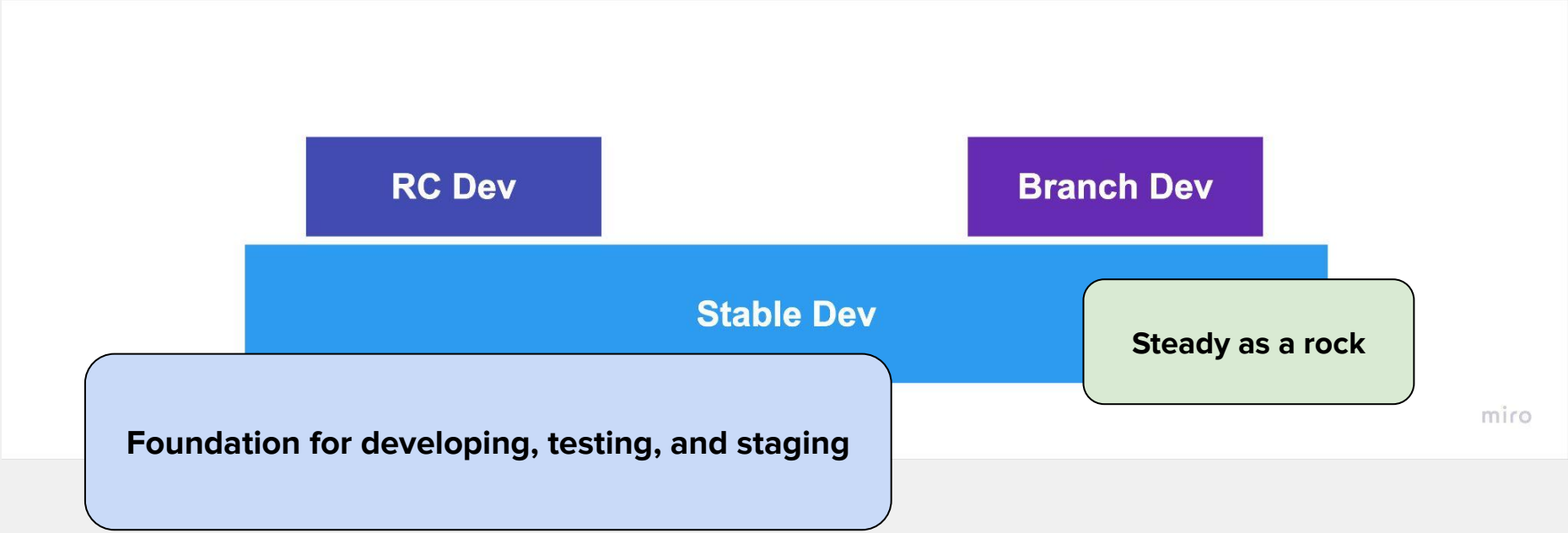
Stable Dev


RC Dev
Branch Dev
Stable Dev
Always contains all the services with the same versions as in production
Default routes come to it

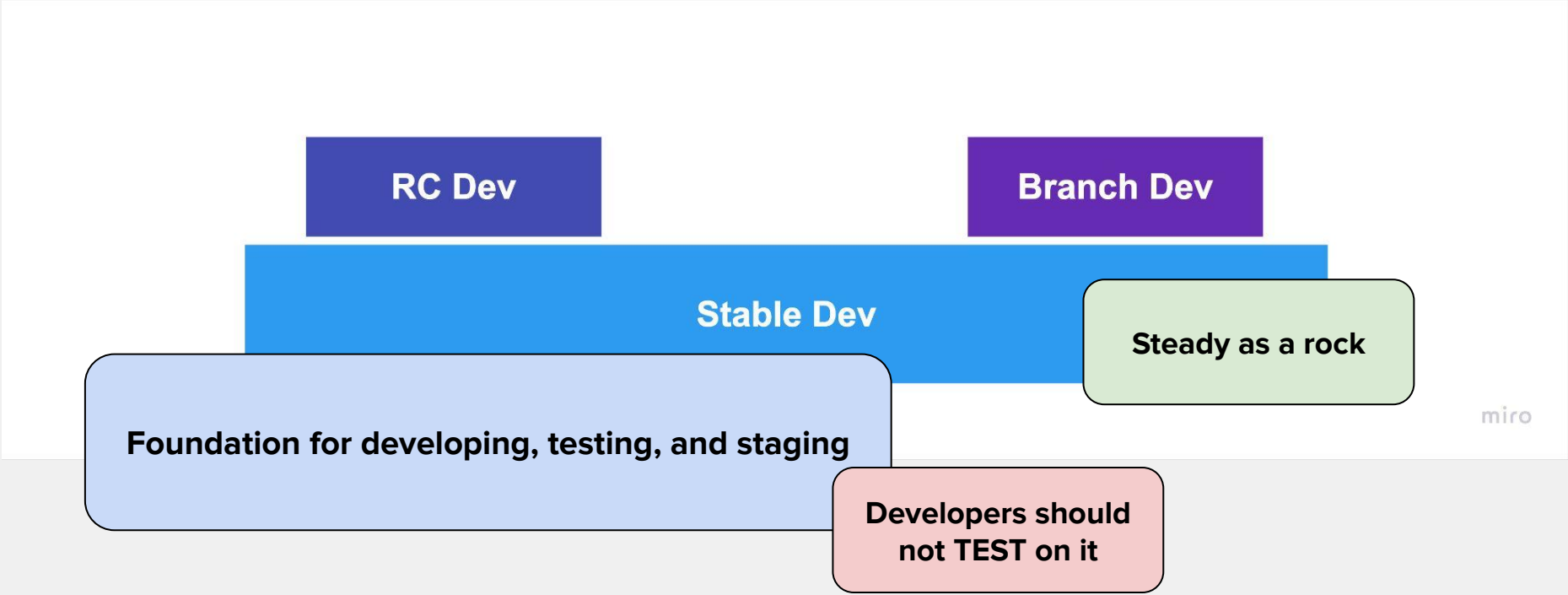
Stable Dev


RC Dev
Branch Dev
Stable Dev
Foundation for developing, testing, and staging
miro

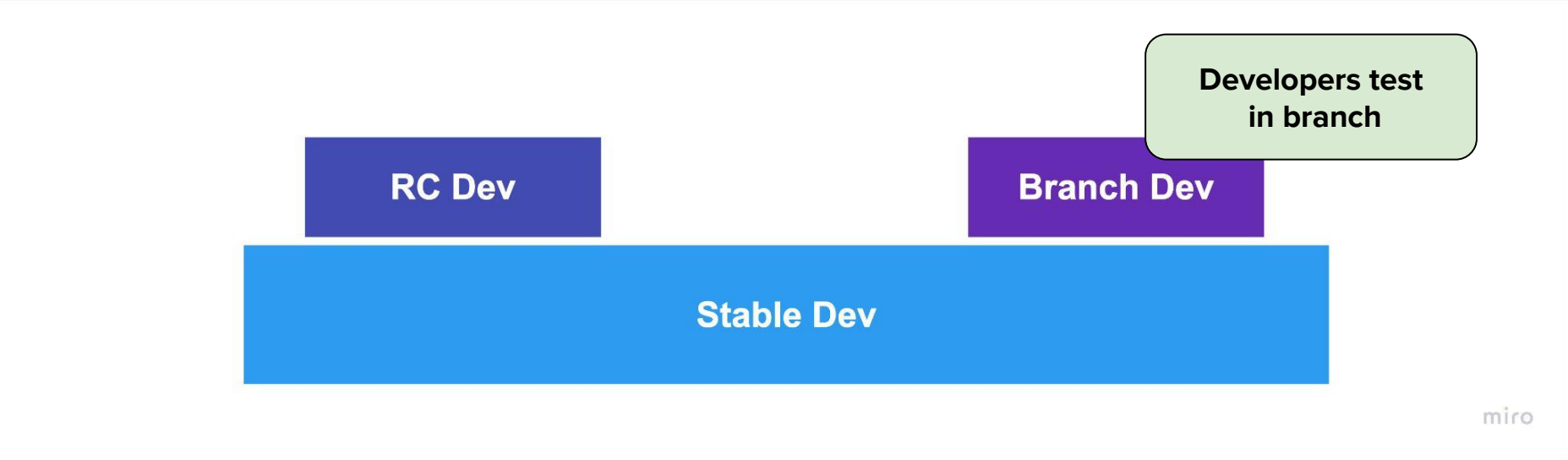
Stable Dev


RC Dev
Branch Dev
Stable Dev
Steady as a rock
Foundation for developing, testing, and staging
miro

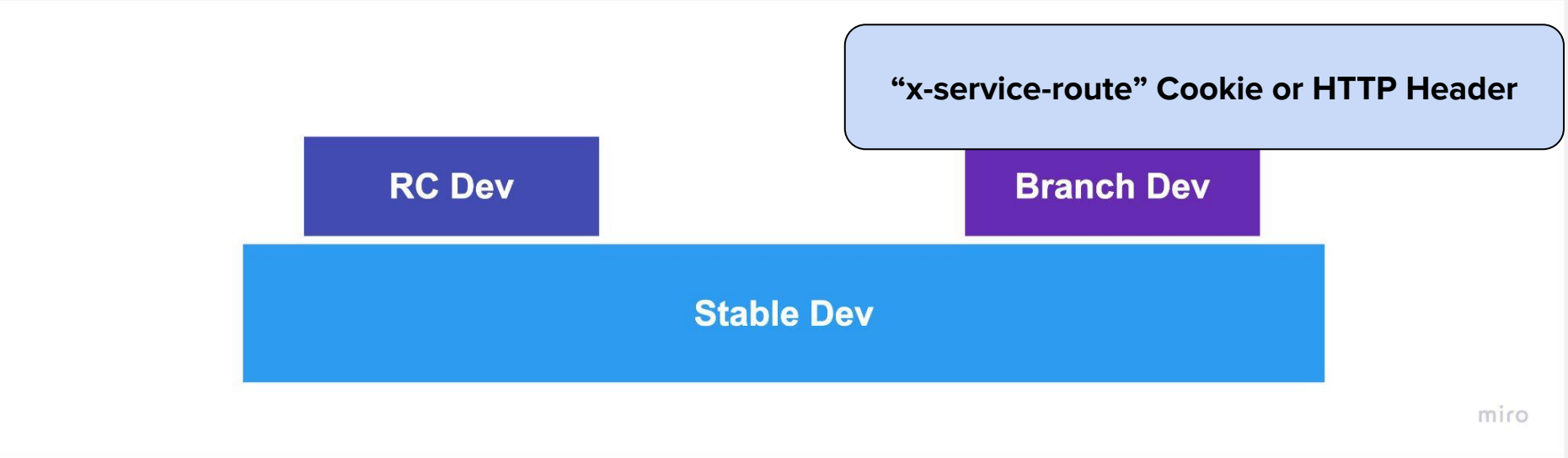
Stable Dev


RC Dev
Branch Dev
Stable Dev
Steady as a rock
Foundation for developing, testing, and staging
Developers should not TEST on it
miro

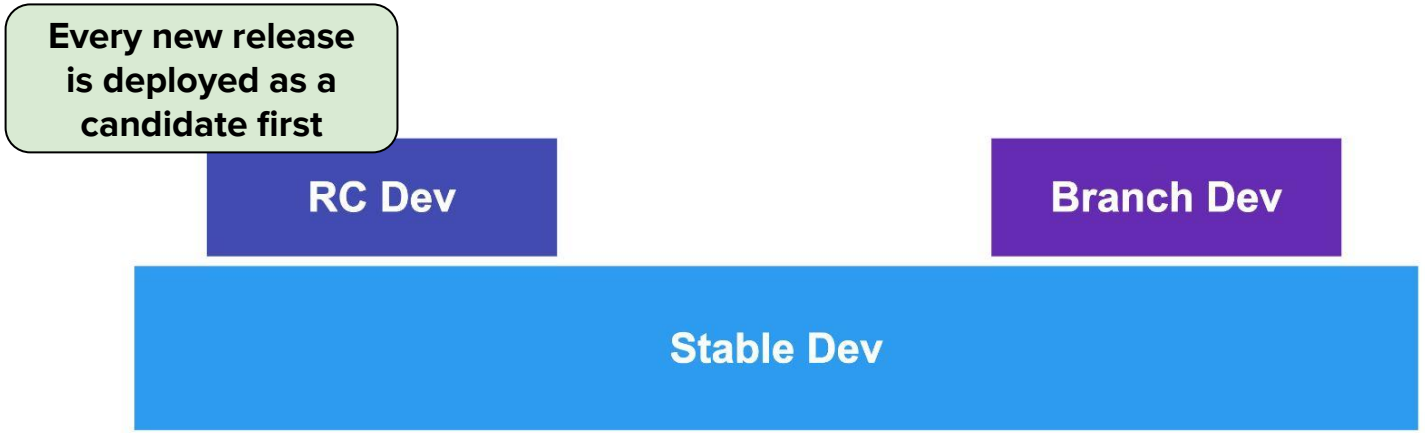
Branch Dev



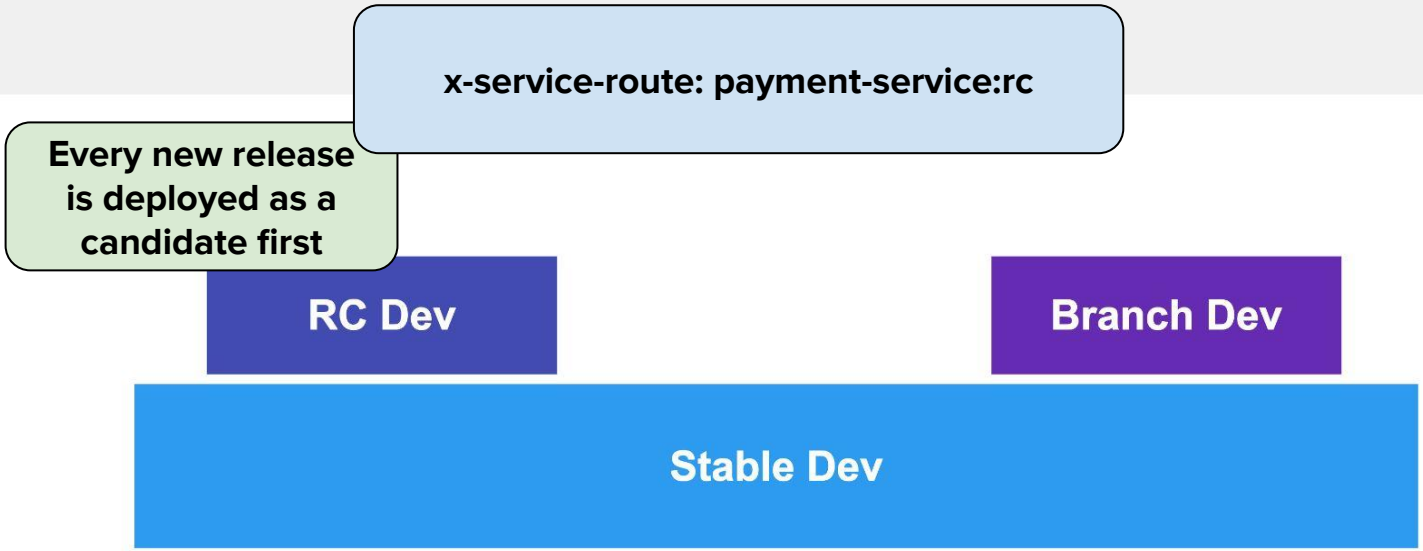
Branch Dev


“x-service-route” Cookie or HTTP Header
RC Dev
Branch Dev
Stable Dev
miro

Canary Dev


Every new release is deployed as a candidate first
RC Dev
Branch Dev
Stable Dev
miro

Canary Dev


x-service-route: payment-service:rc
Every new release is deployed as a candidate first
RC Dev
Branch Dev
Stable Dev
miro


LEFT
EXIT 12
Staging
Stable
Dev
Why did we turn this way?
imgflip.com

What if not?

- **No dogfooding**
  - A silo between QA and developers
  - Developers are pushed to fix the stage

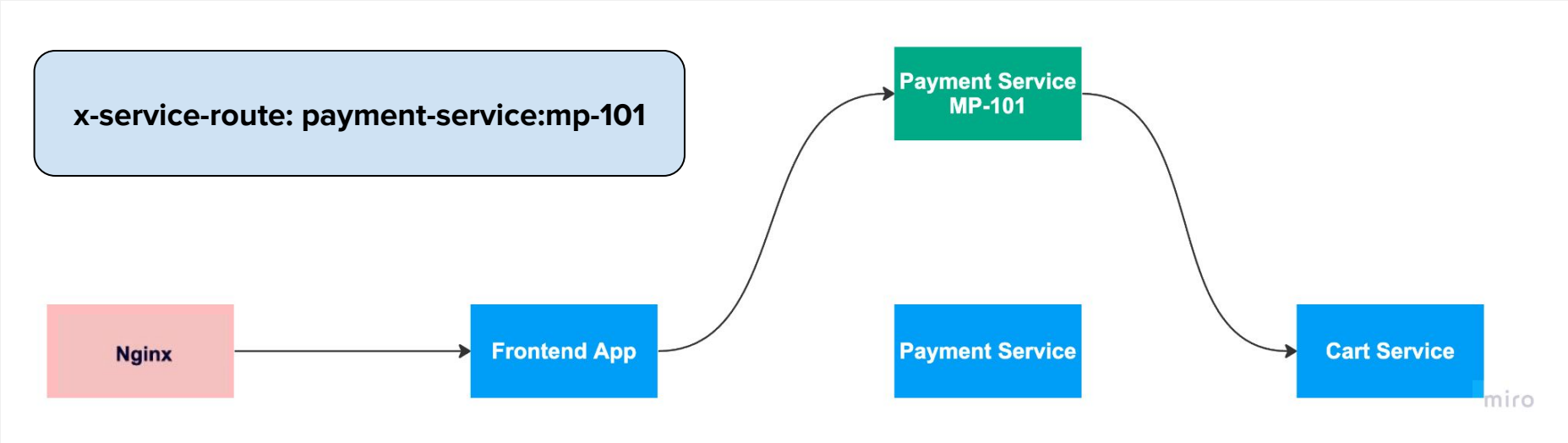
What if not?

- **No dogfooding**
  - A silo between QA and developers
  - Developers are pushed to fix the stage
- **We should keep stable two environments instead of one**
  - Staging should be stable by design
  - The development environment should be stable too
    - If the authorization service doesn’t work - developer cannot test their branch

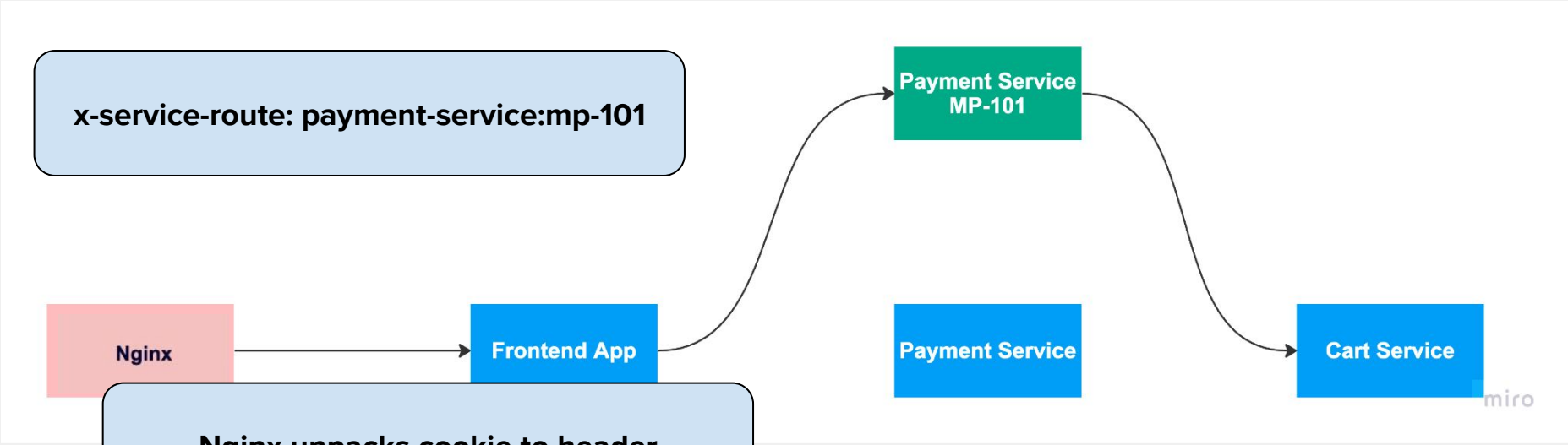
Chapter 3: Make Some Code

Traffic Routing

Service Injection



Service Injection



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
```

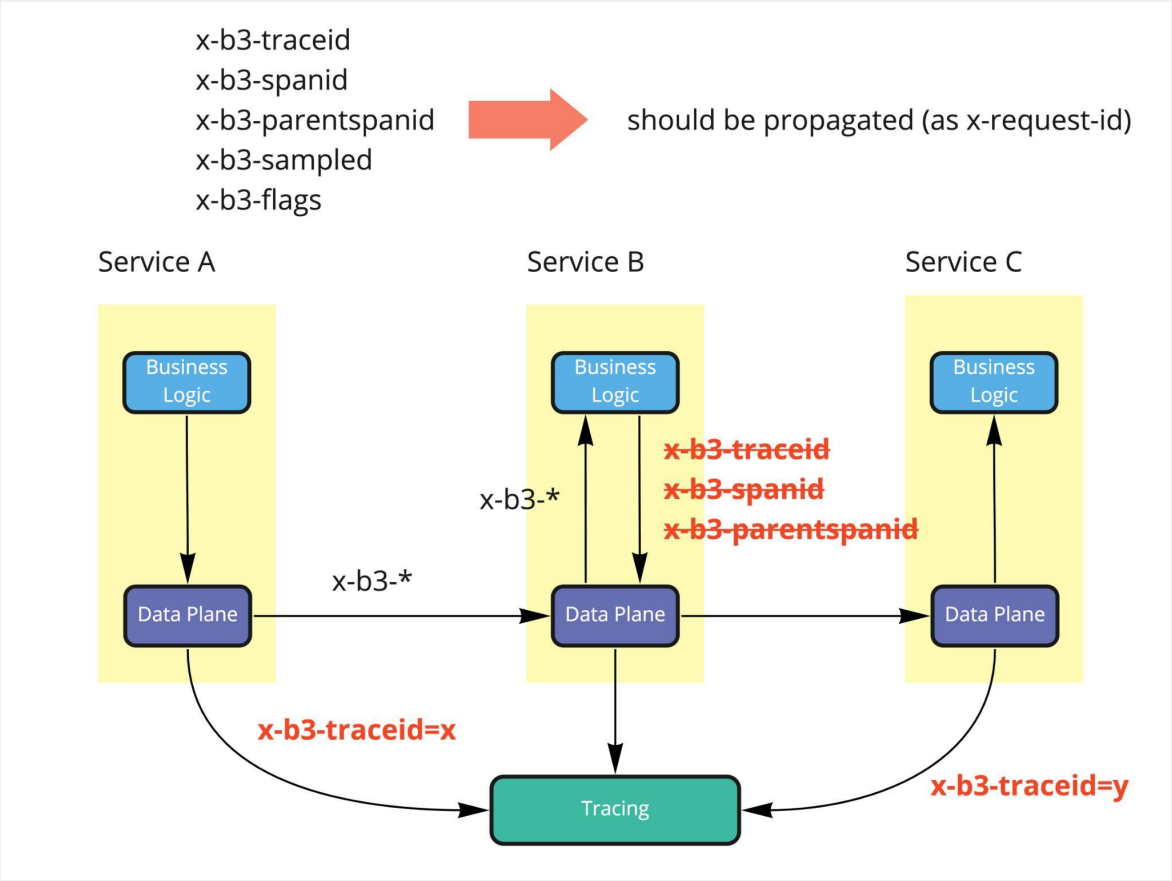
We cannot add it with Helm chart

Virtual Service Merge Operator

```
---
apiVersion: istiomerger.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

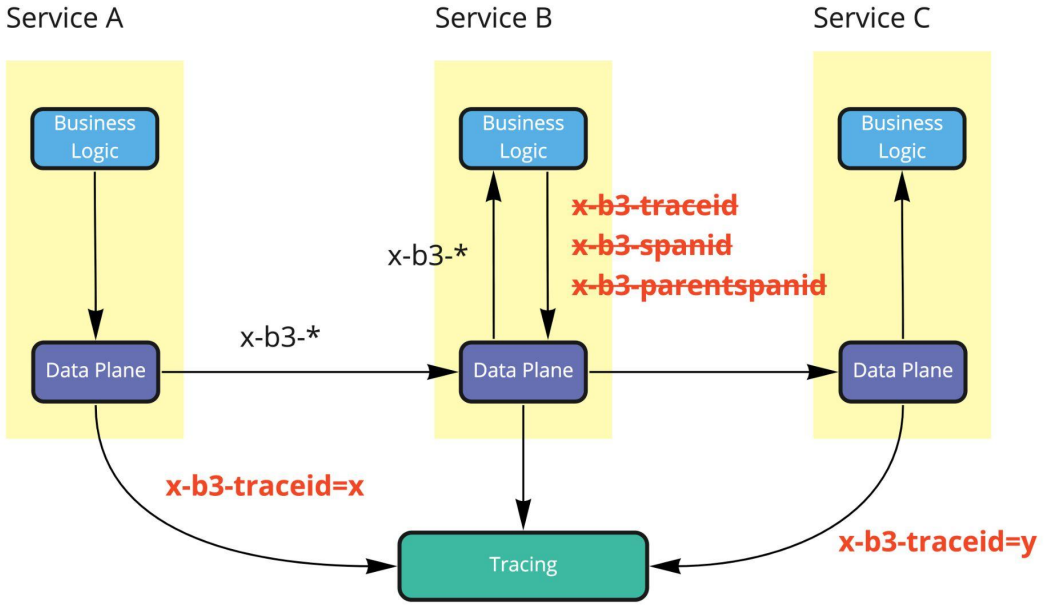
Propagation Problem



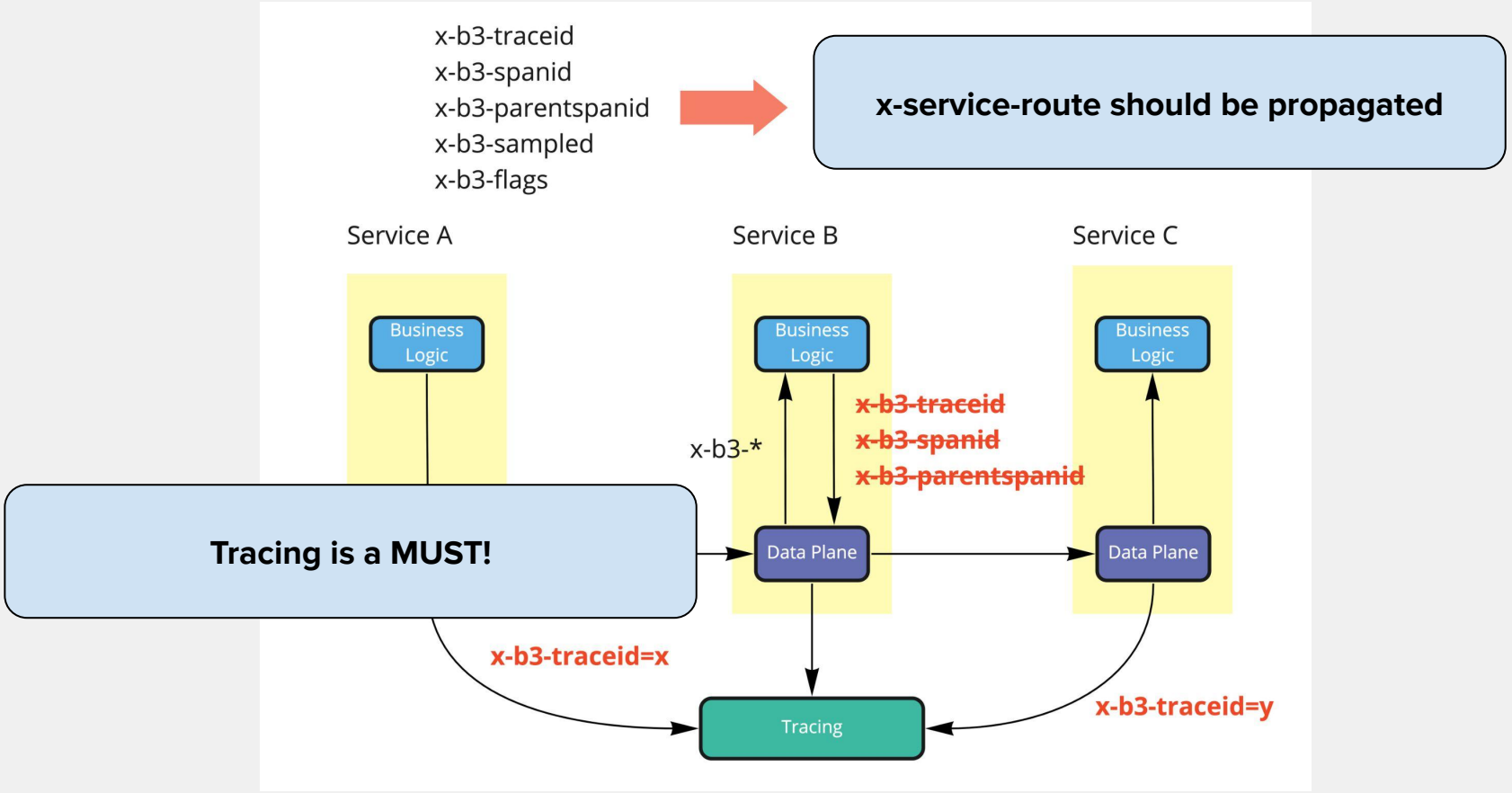
Propagation Problem

x-b3-traceid
x-b3-spanid
x-b3-parentspanid
x-b3-sampled
x-b3-flags

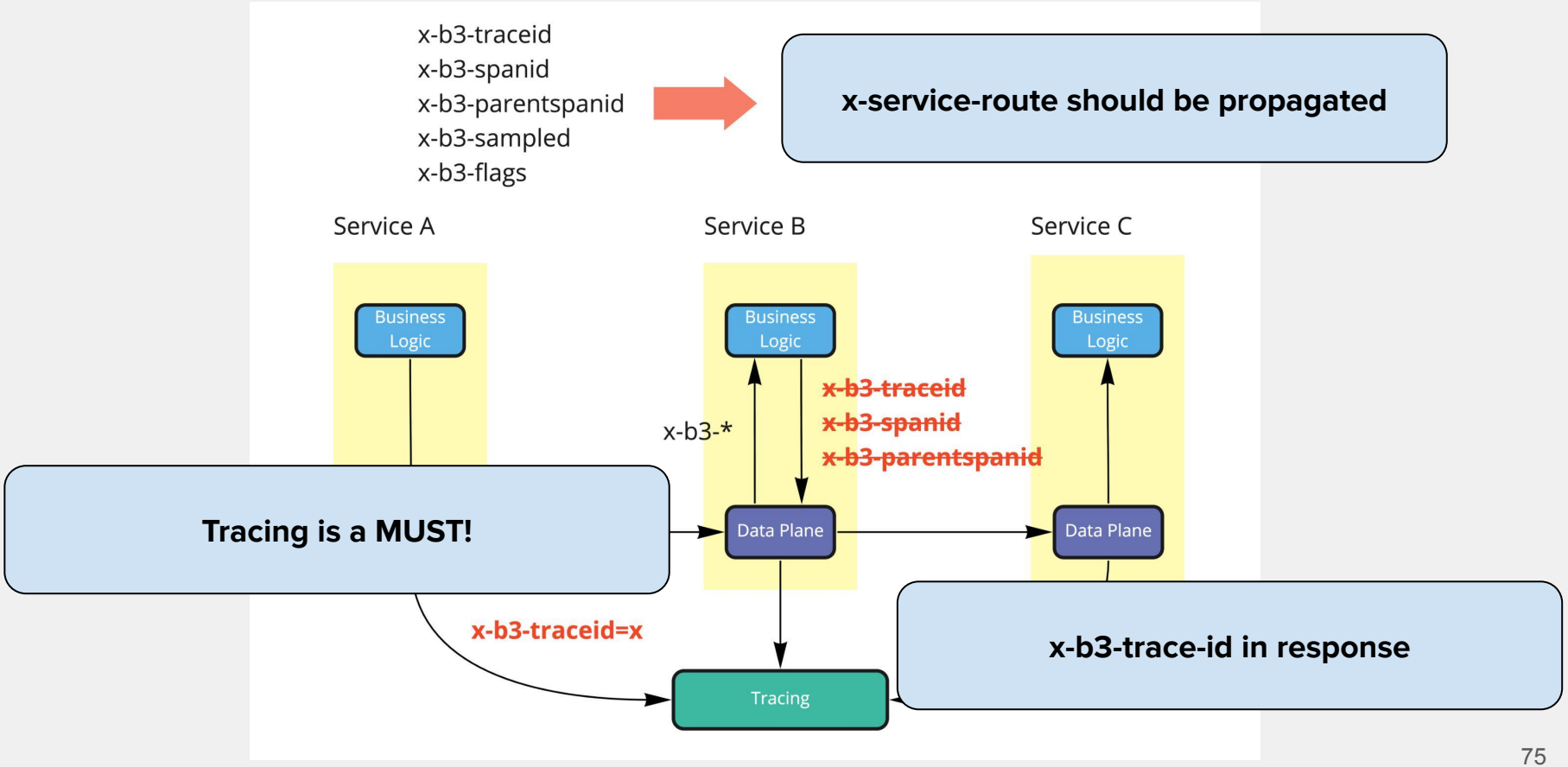
x-service-route should be propagated



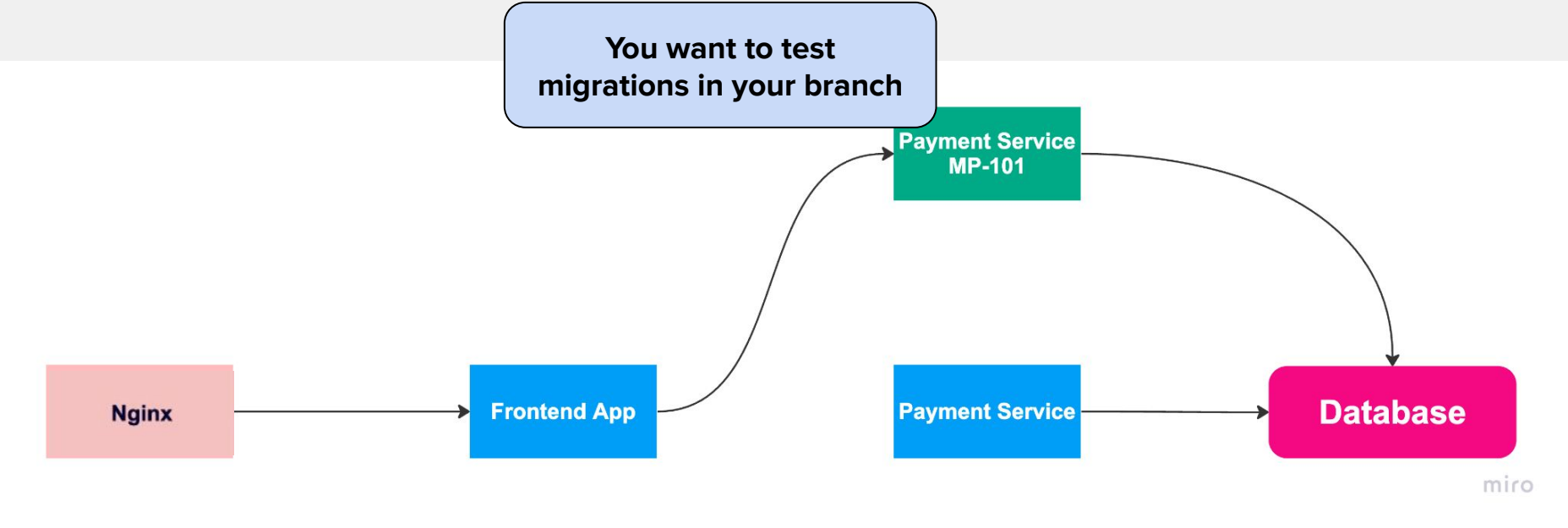
Propagation Problem



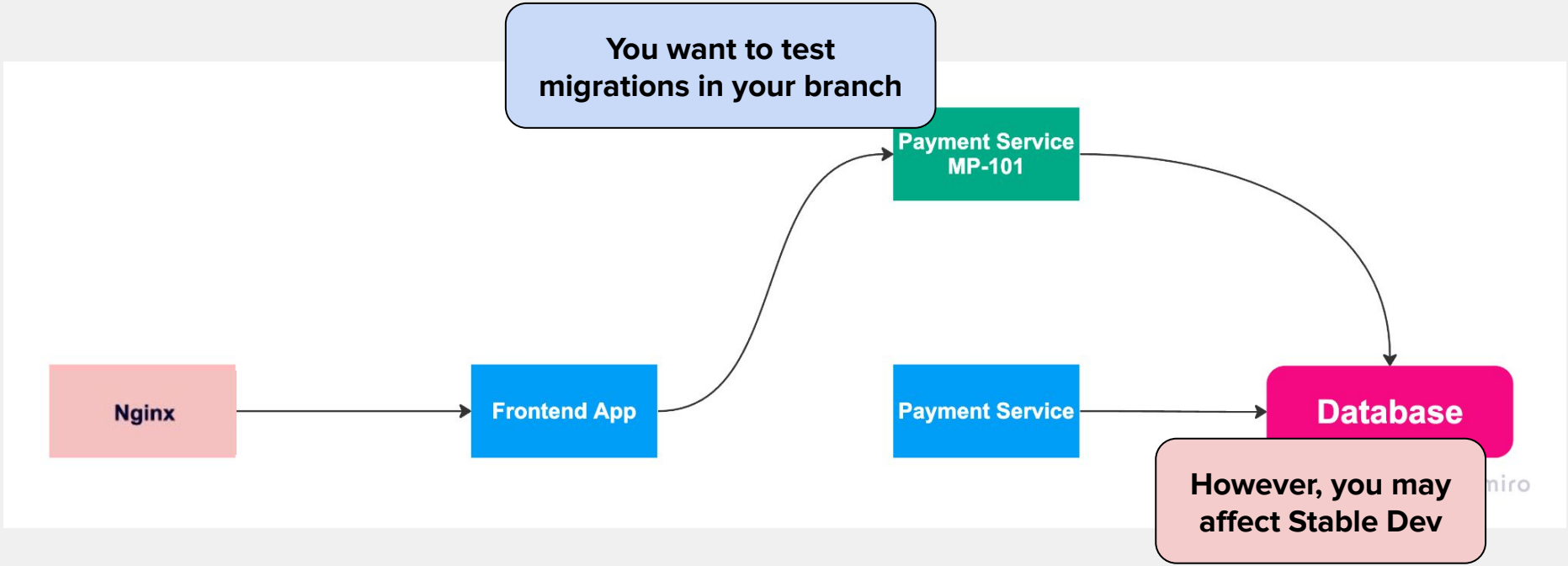
Propagation Problem



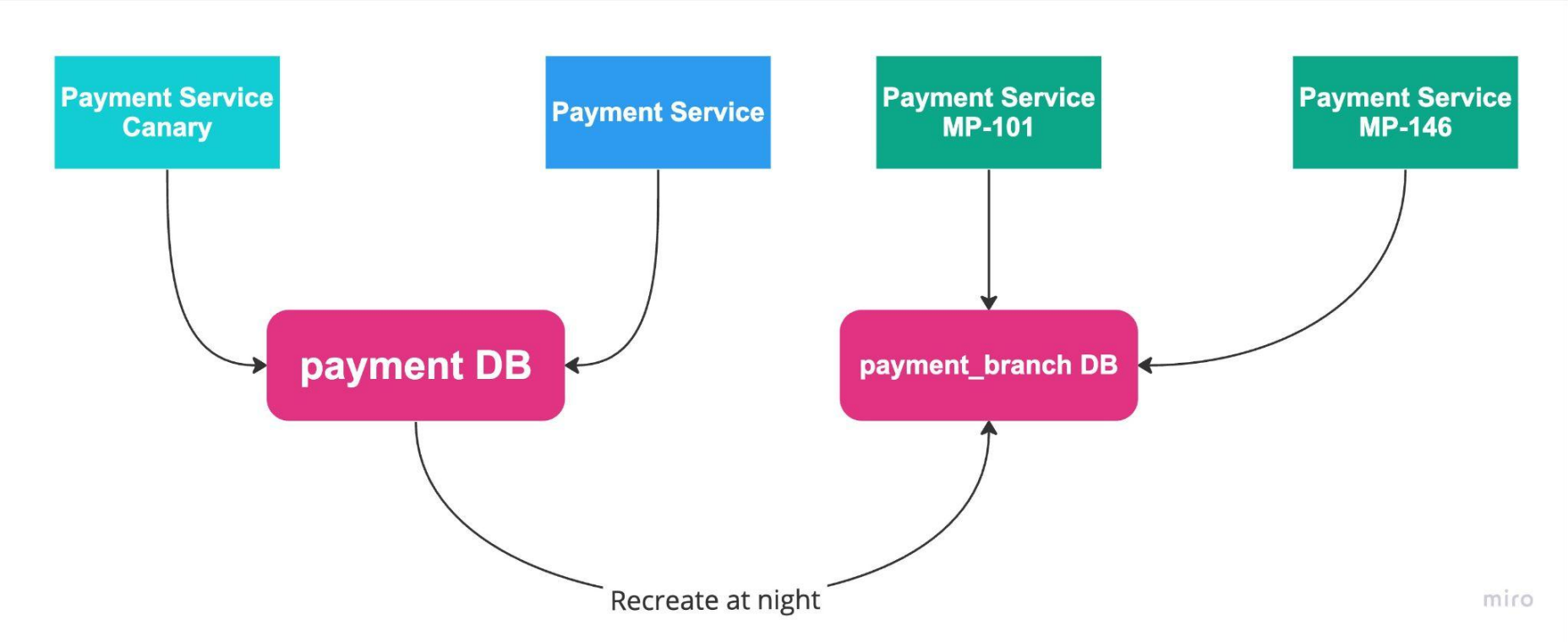
Tricky Case: Migrations that Break



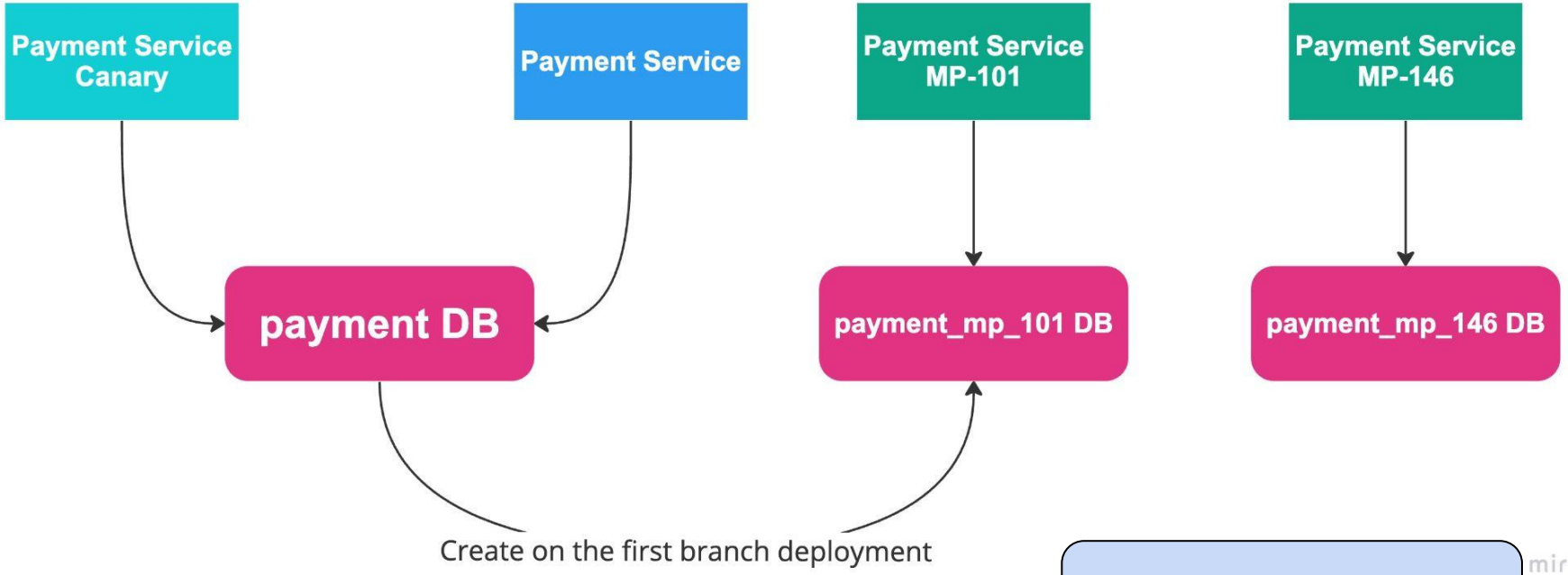
Tricky Case: Migrations that Break



Solution: Migrations that Break

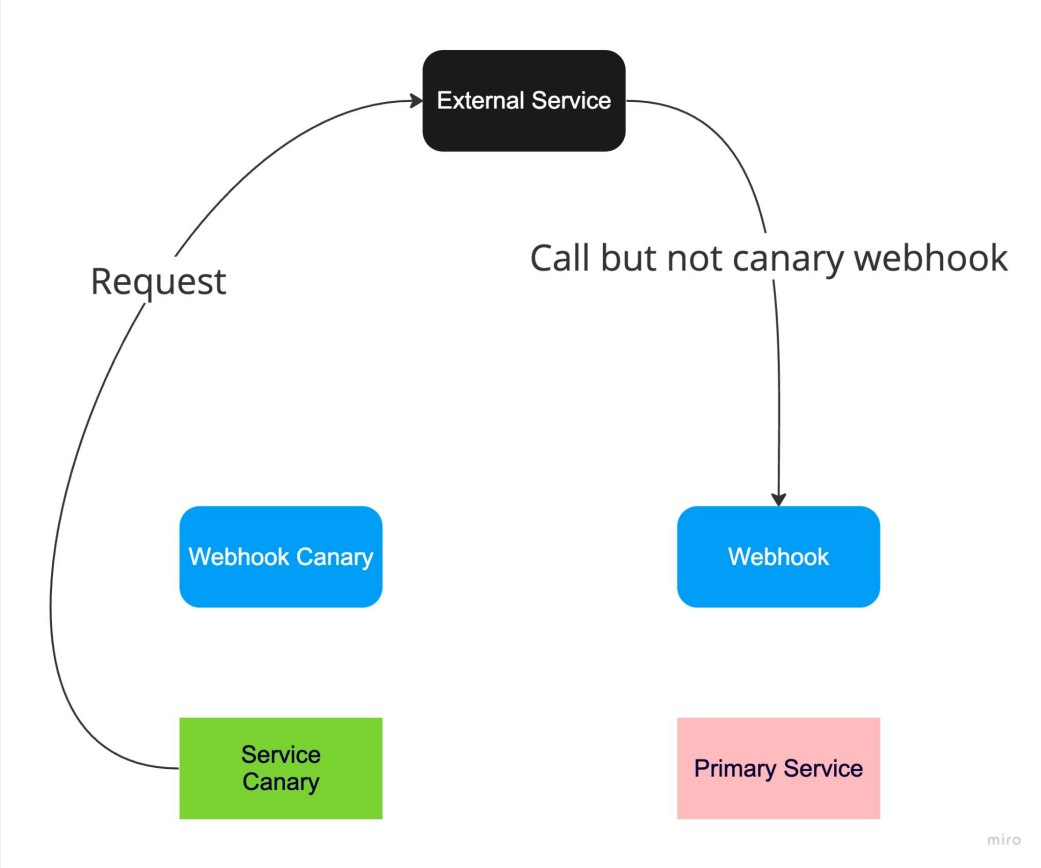


Solution: Migrations that Break

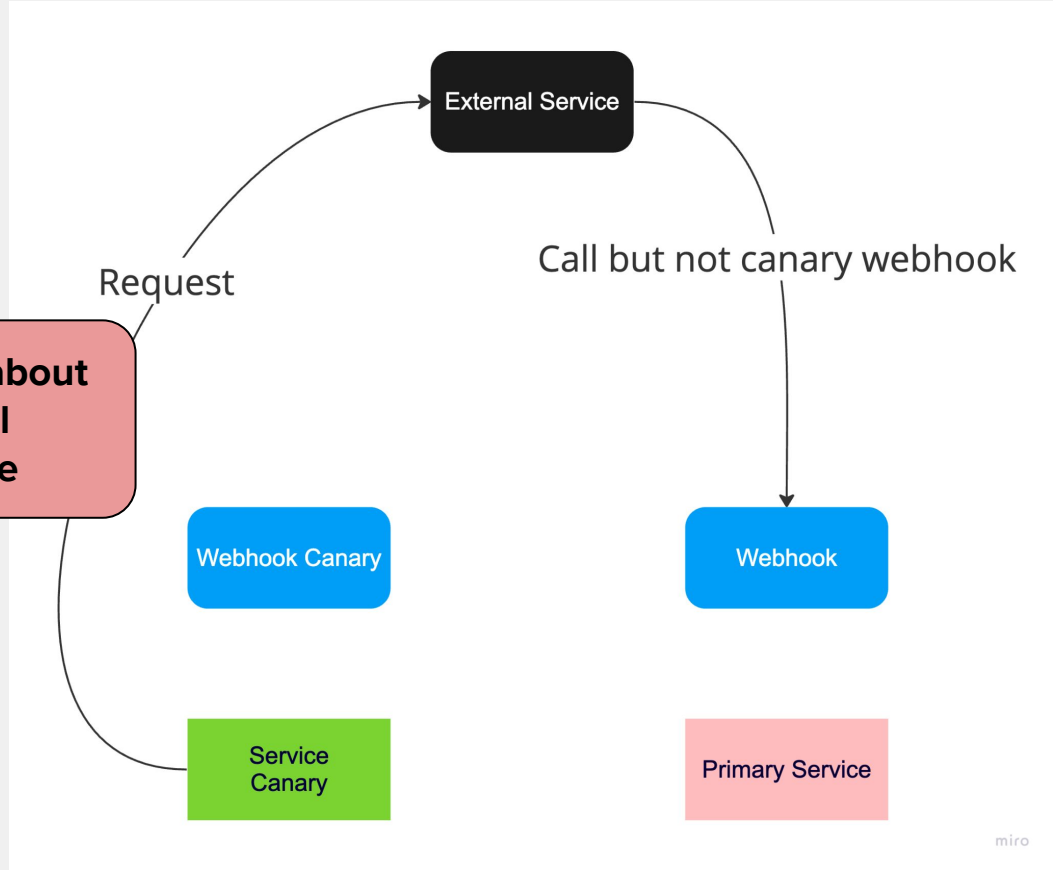


We chose this variant

Tricky Case: Webhooks

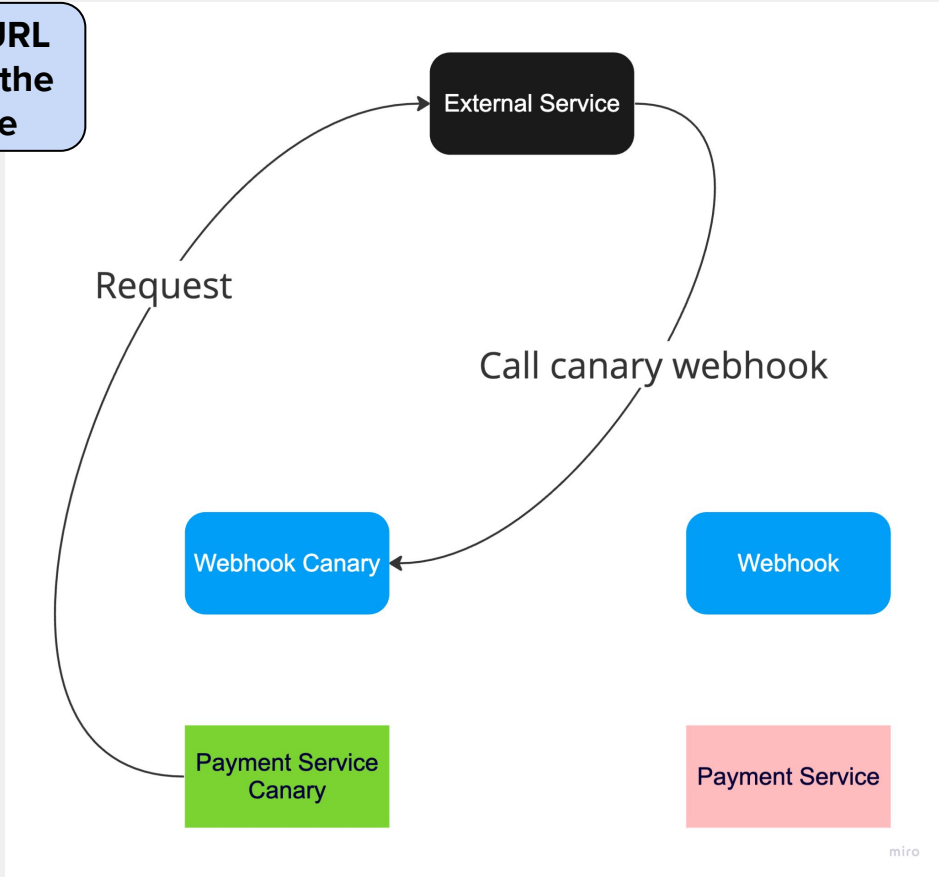


Tricky Case: Webhooks



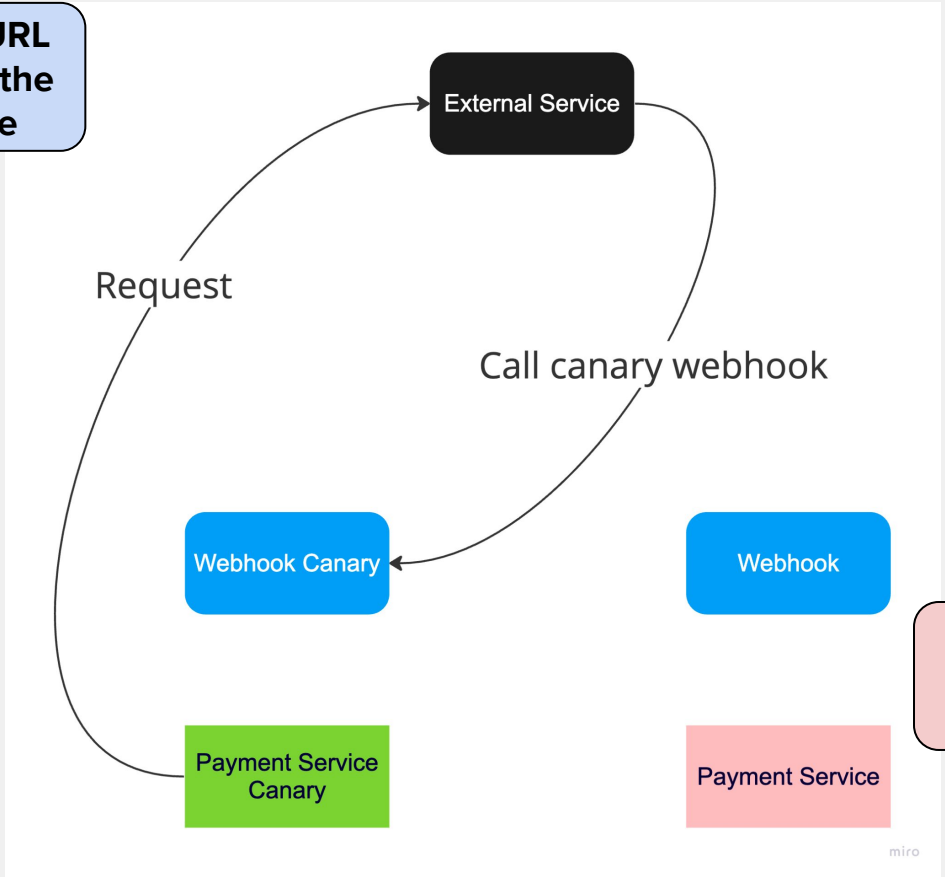
Solution #1: Webhooks

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



Solution #1: Webhooks

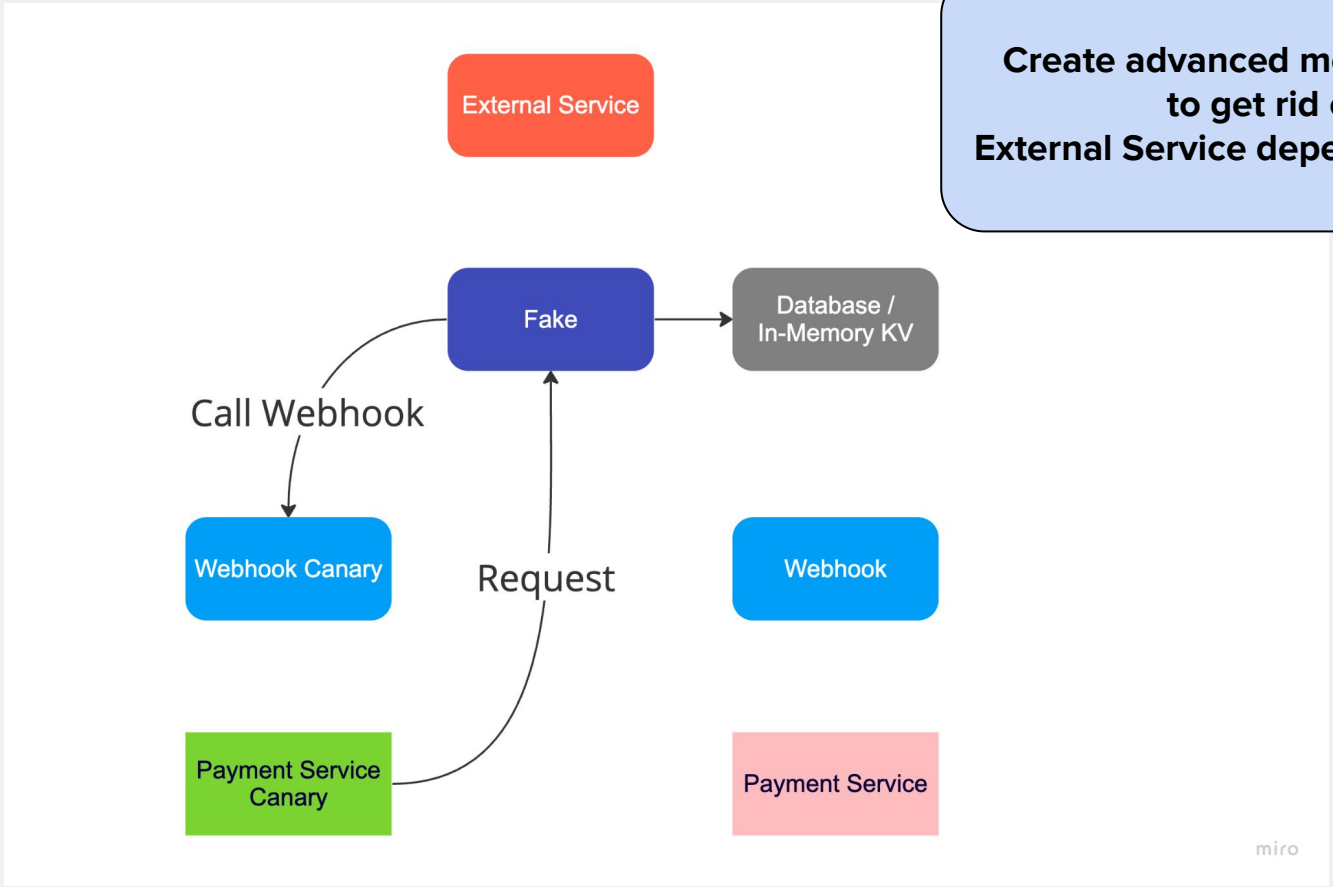
We can switch URL for webhook on the third-party side



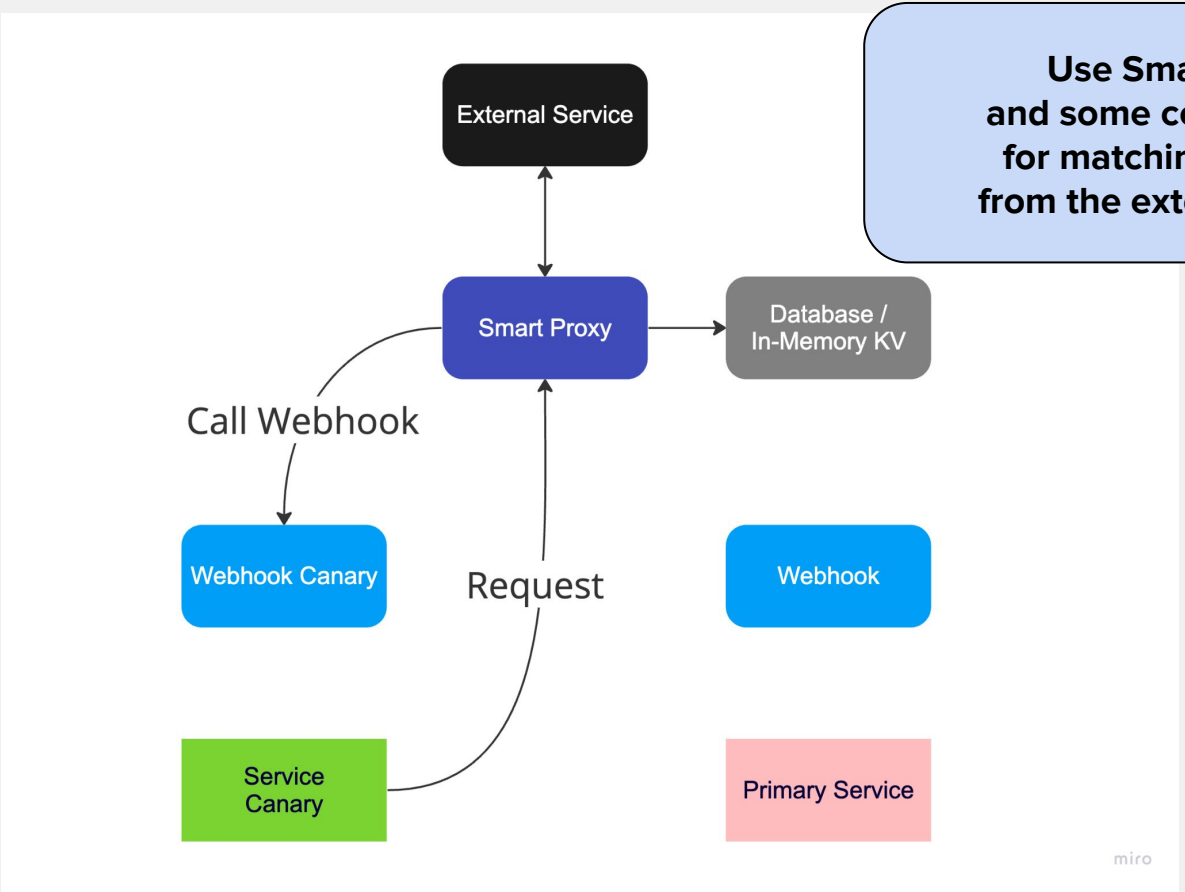
However, we affect Stable Dev

Solution #2: Webhooks

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

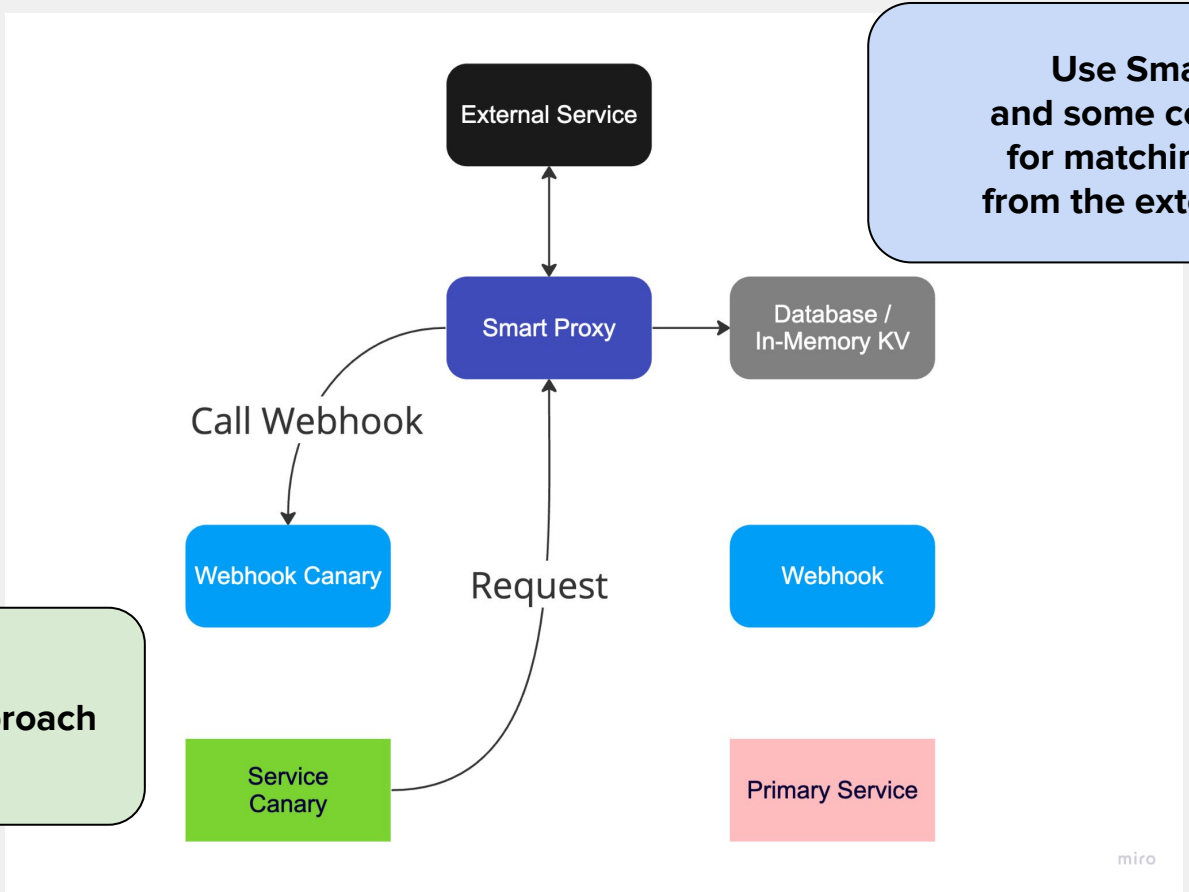


Solution #3: Webhooks



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

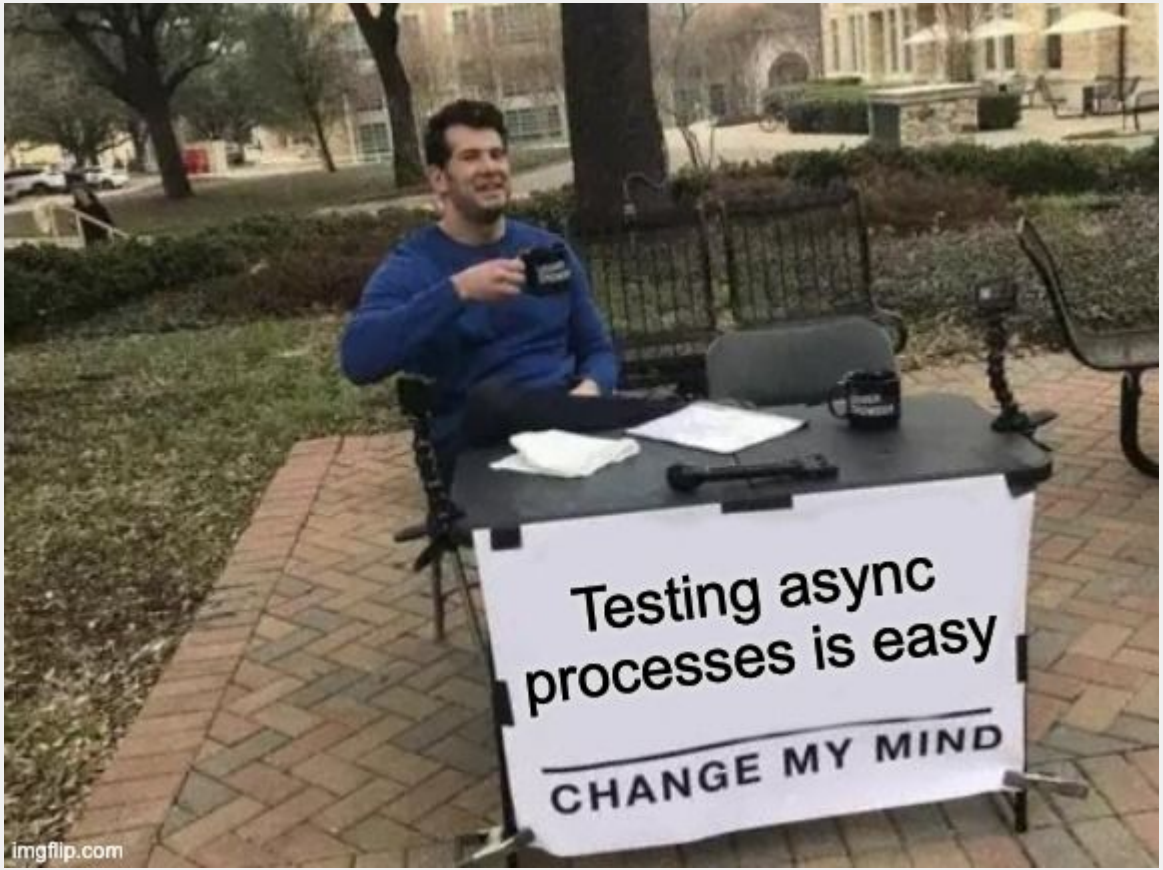
Solution #3: Webhooks



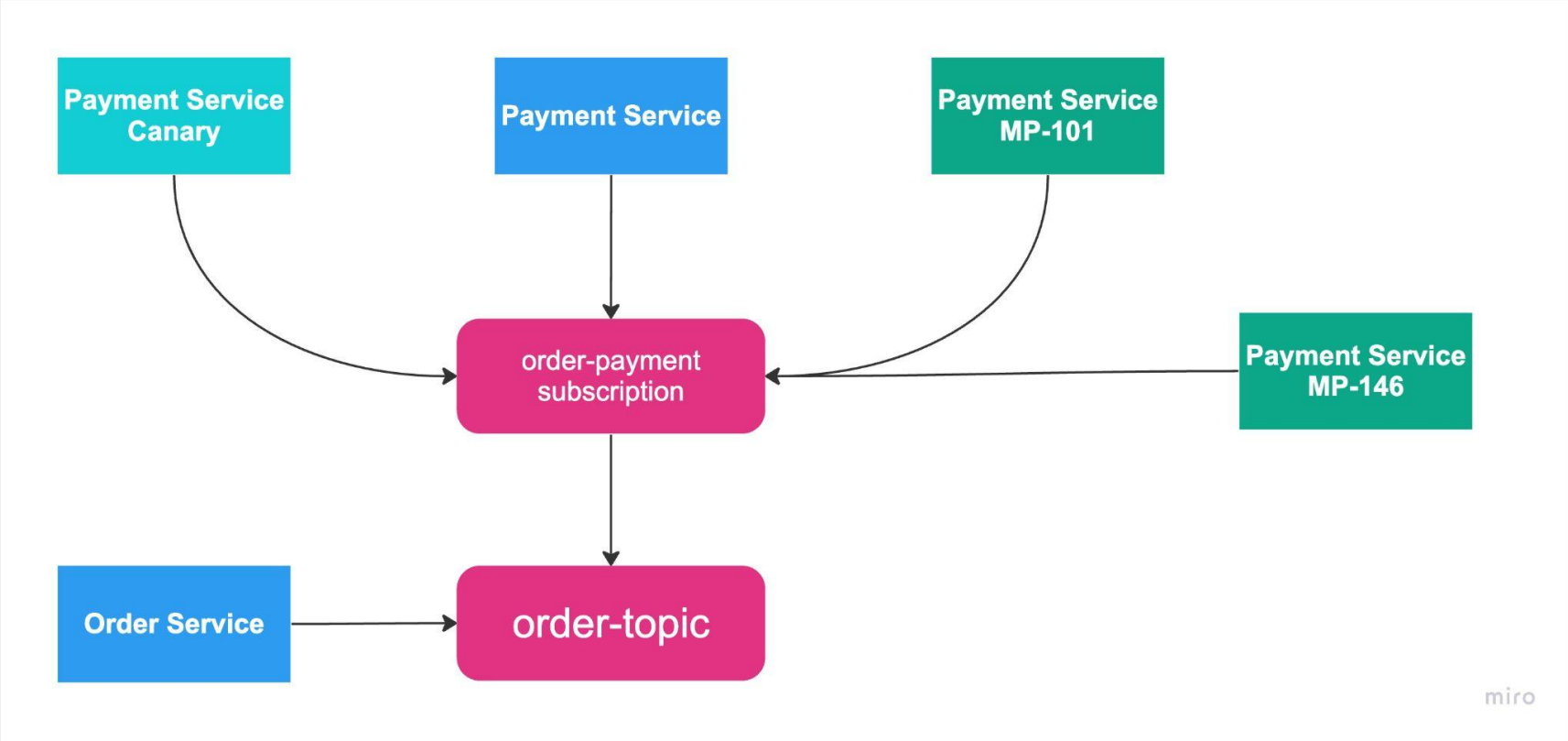
Use Smart-Proxy and some correlation ID for matching requests from the external service

We chose this approach

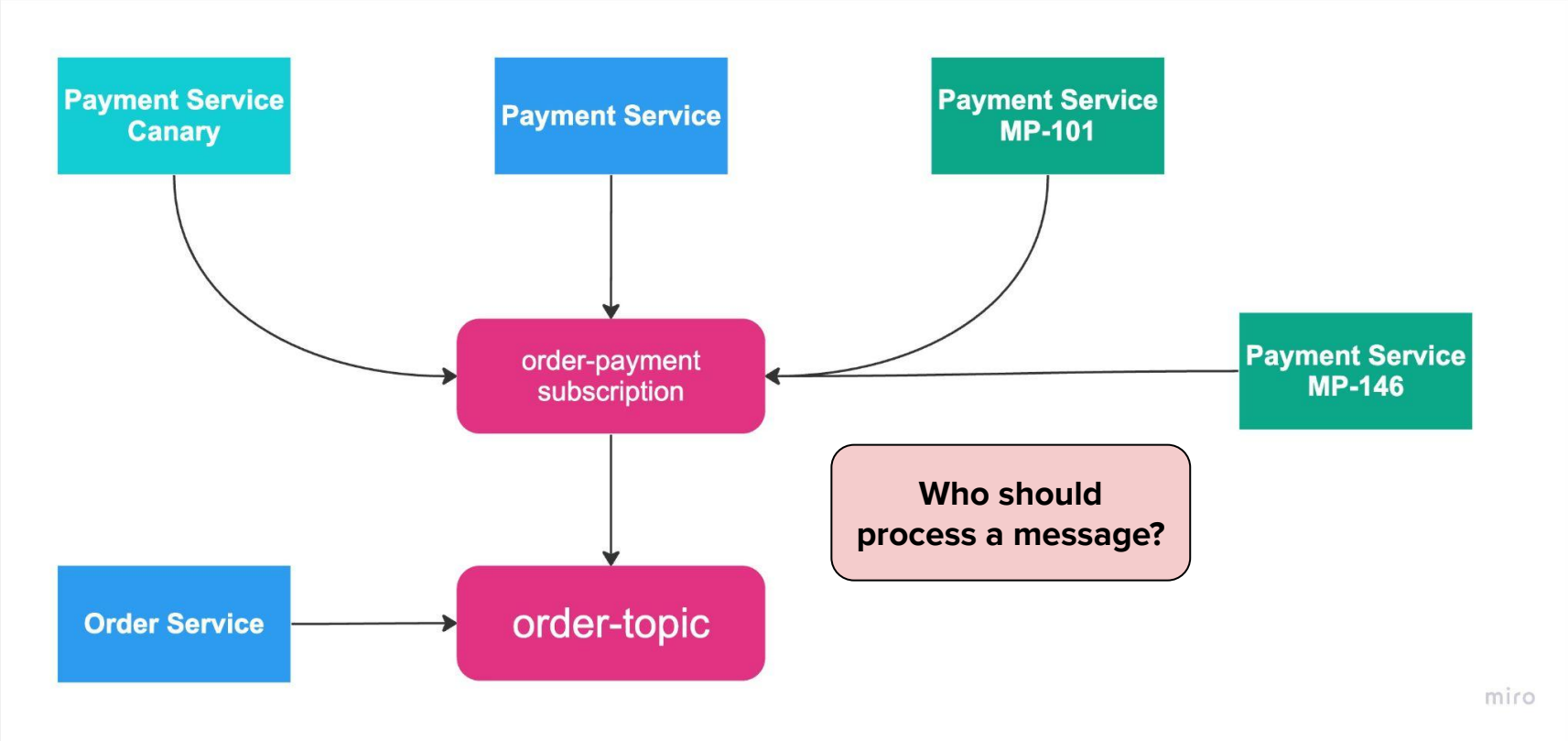
Unblocking Async Scenarios



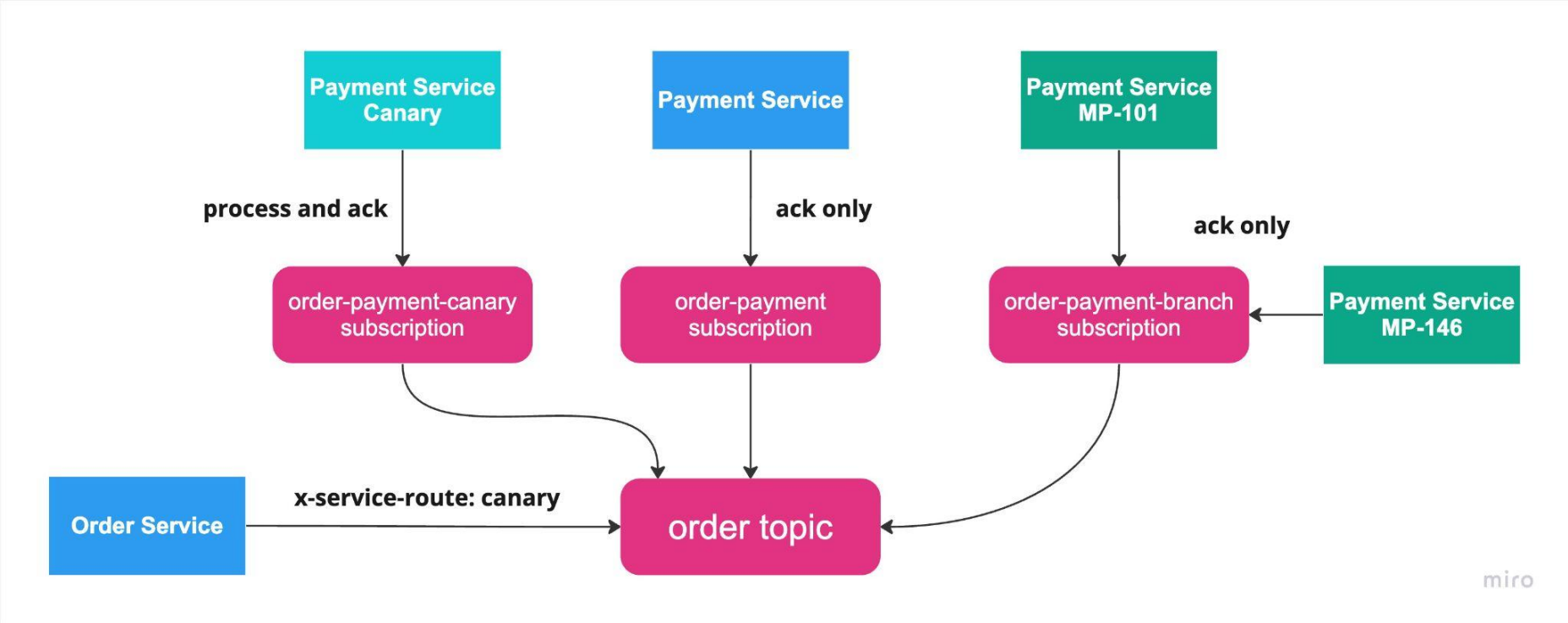
Async Issues



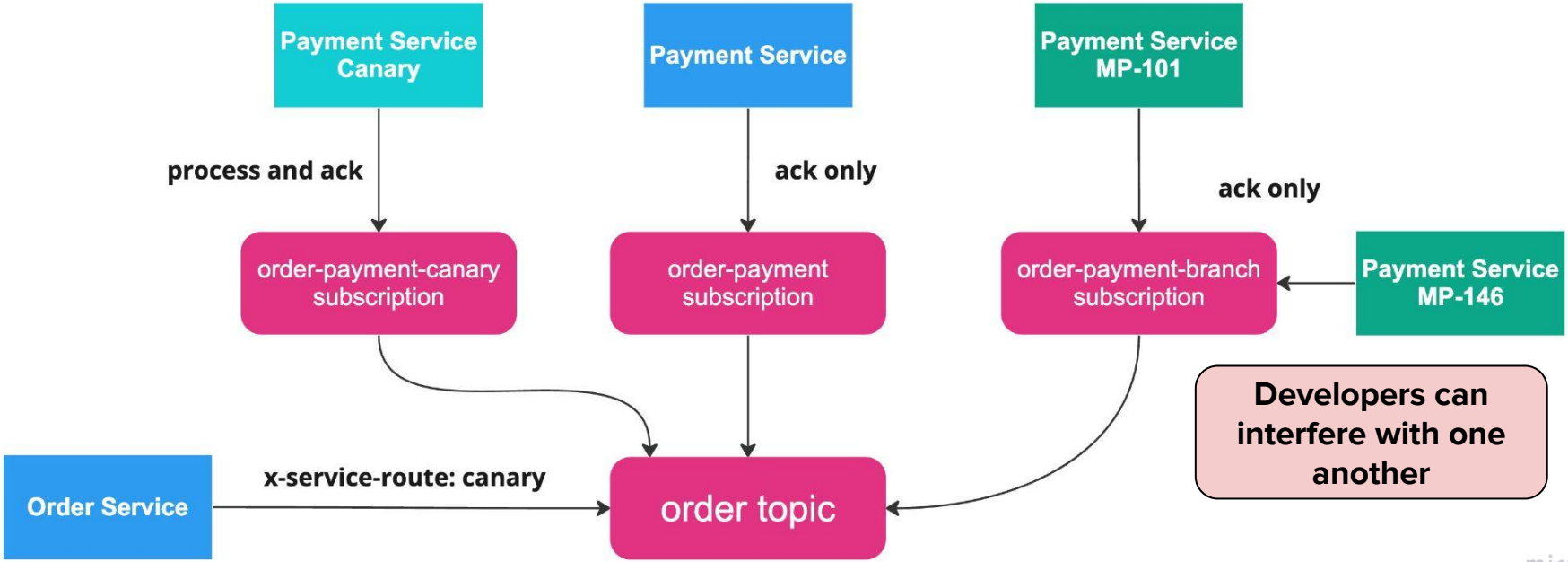
Async Issues



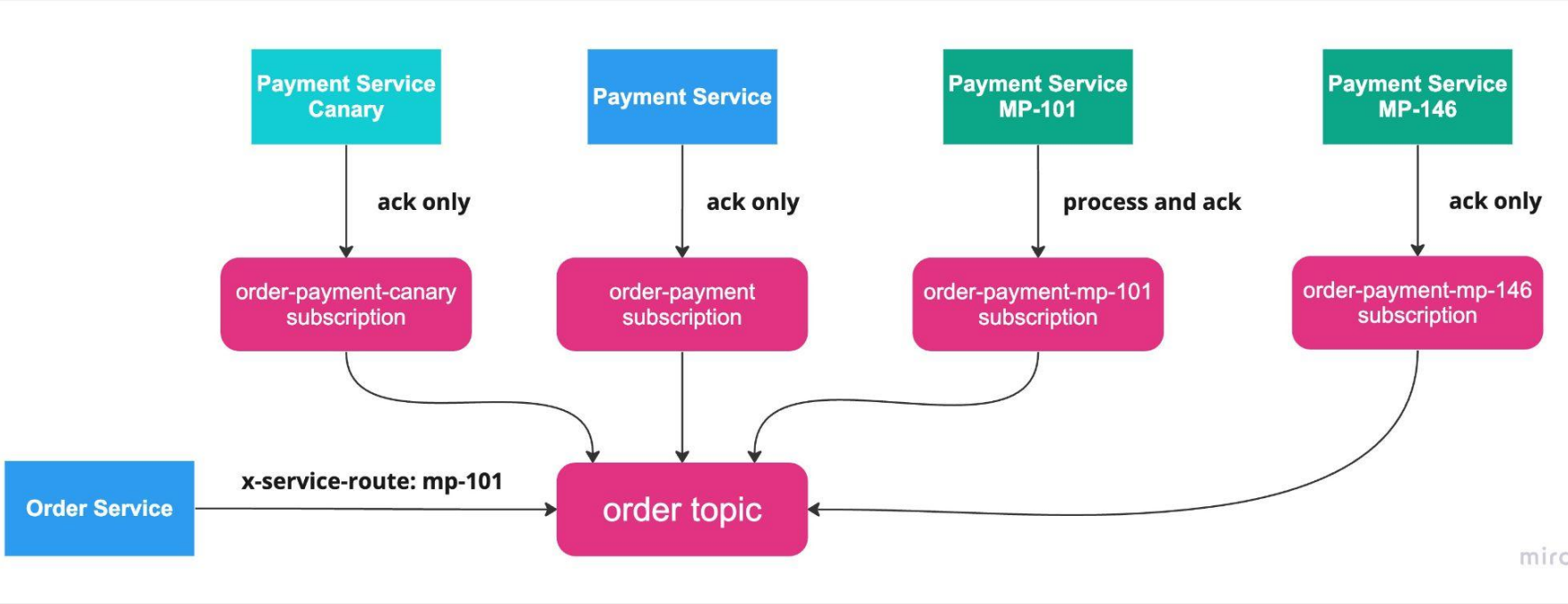
Let’s Use the Same Appoach



Let’s Use the Same Appoach



Subscription per Branch



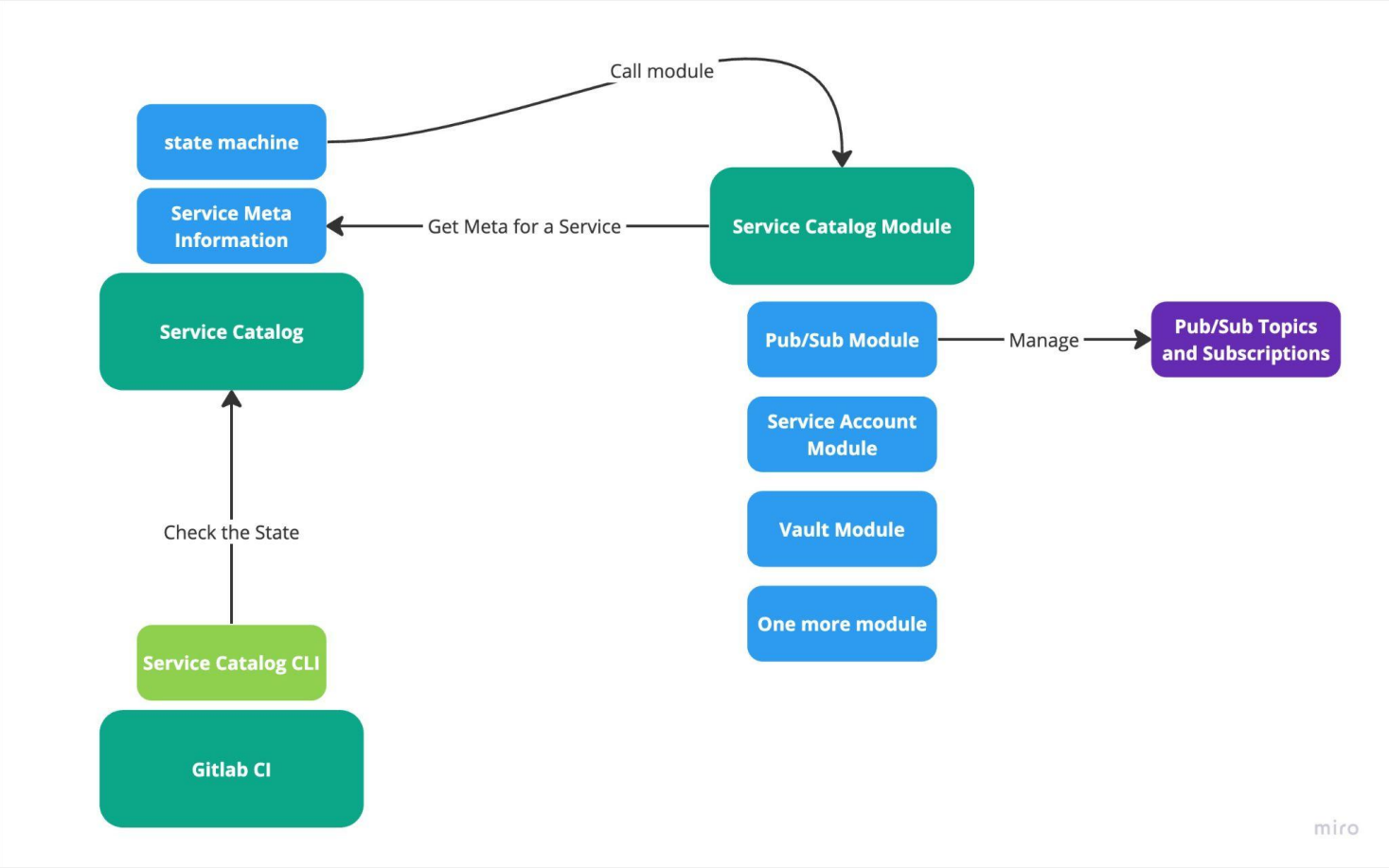
Issues to address

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

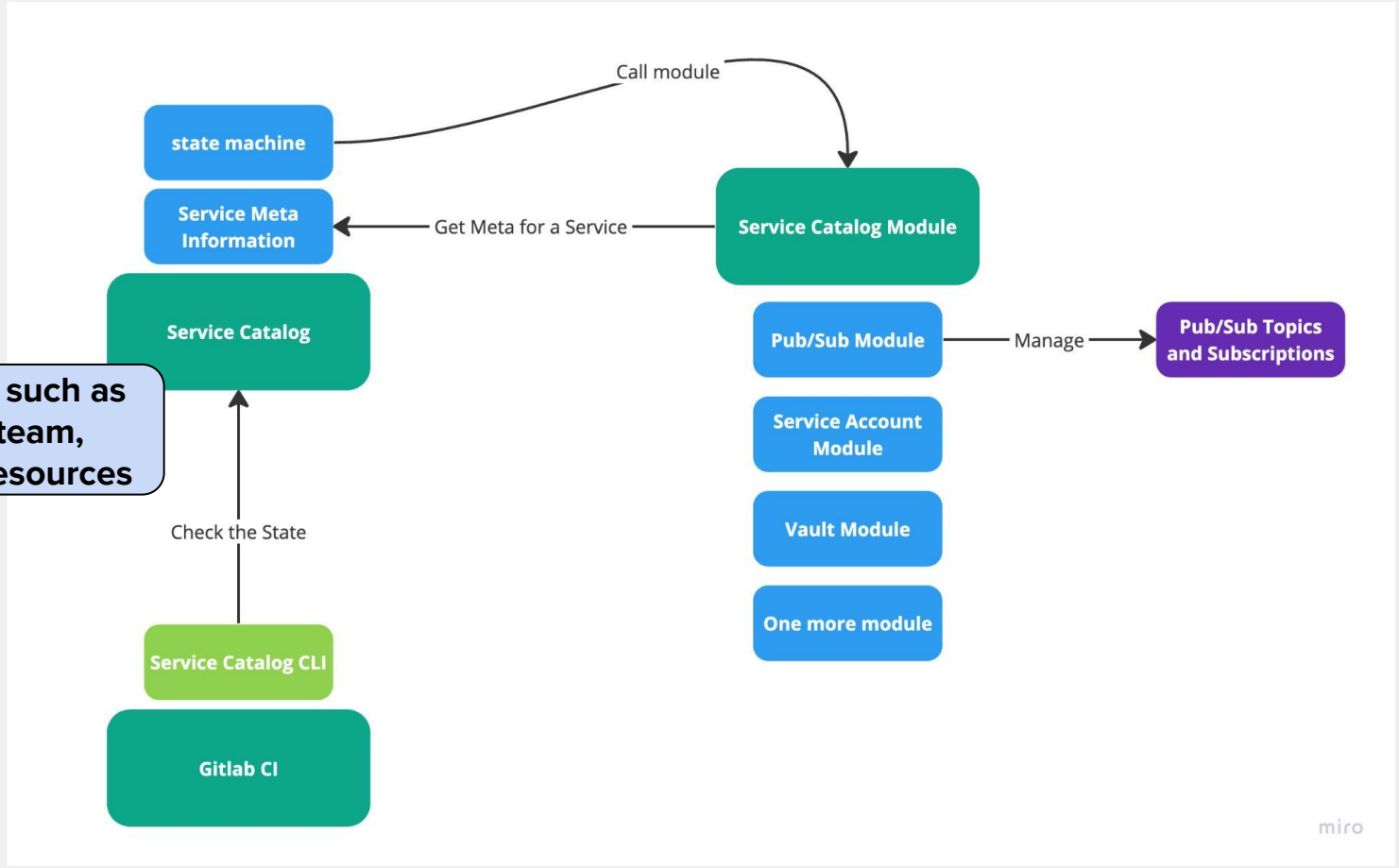
Issues to address

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

Service Catalog



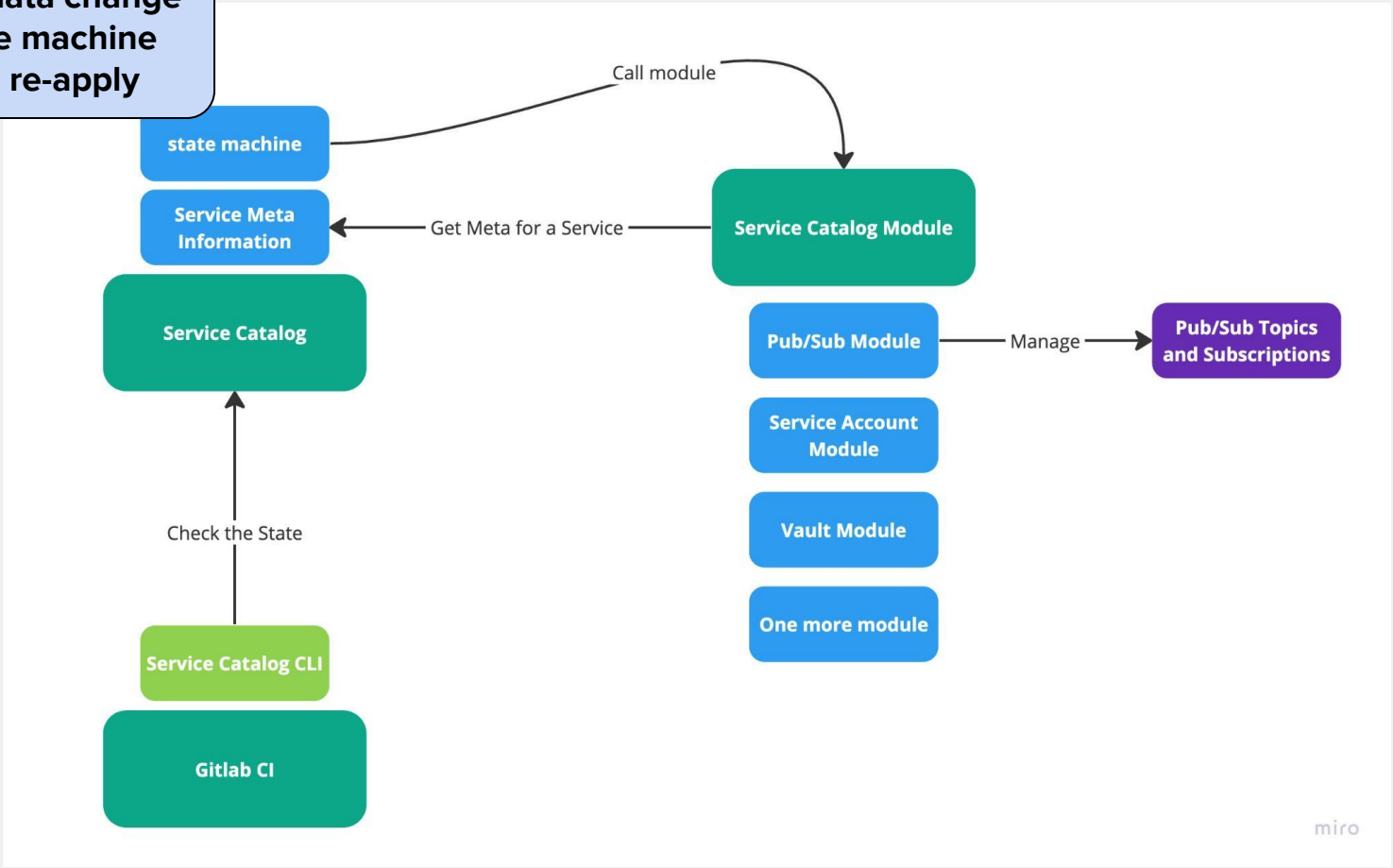
Service Catalog



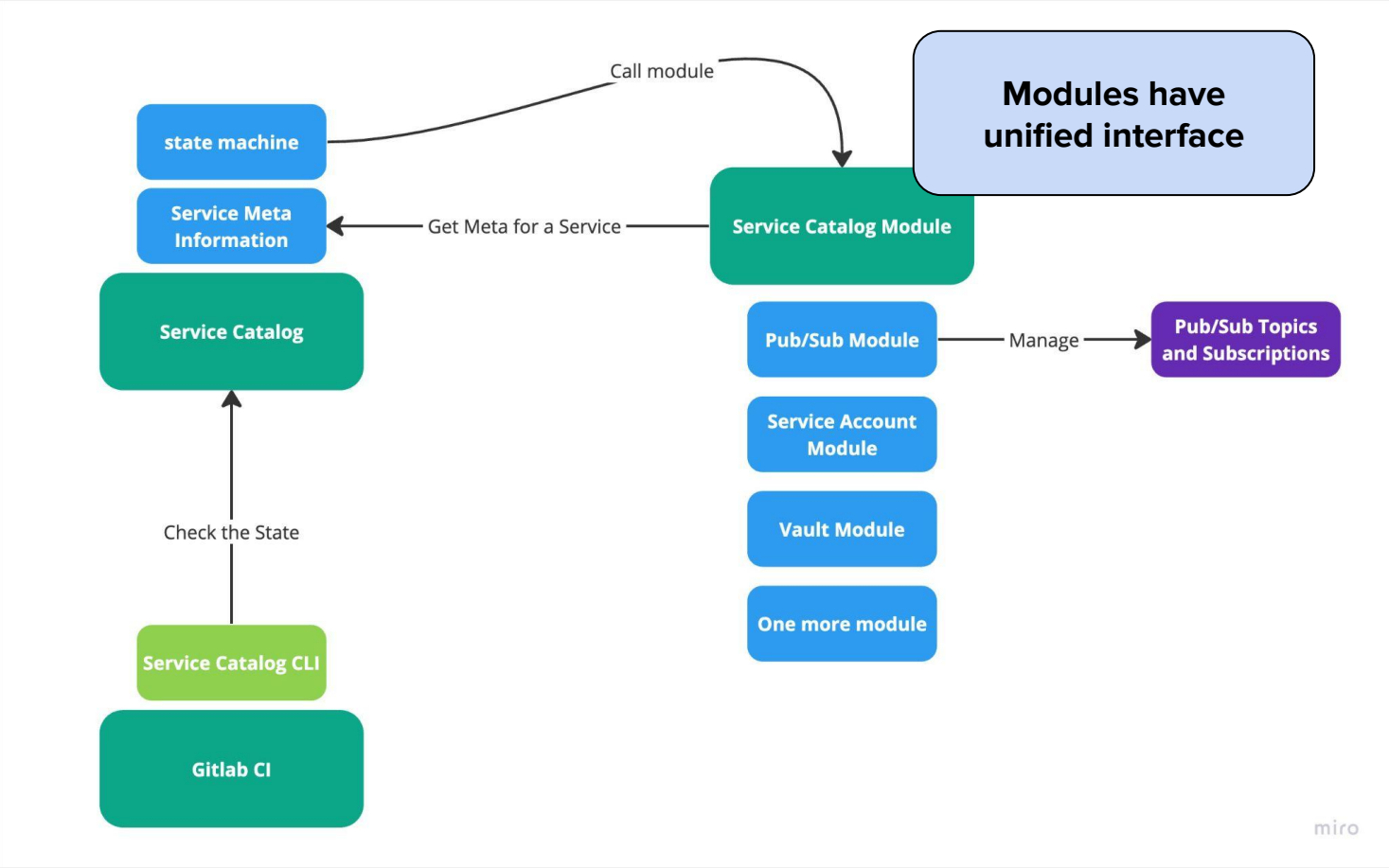
Metadata such as name, team, domain, resources

Service Catalog

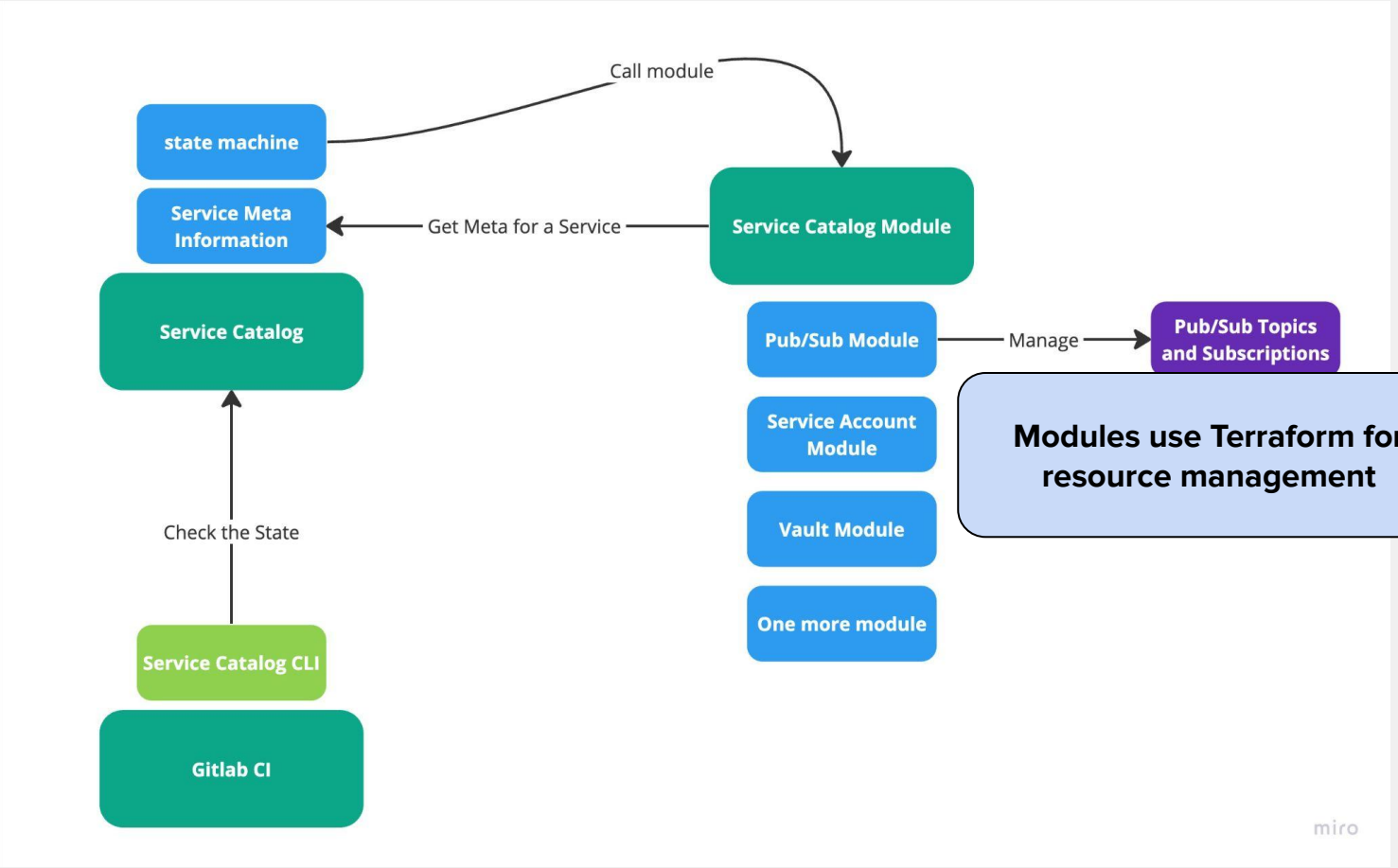
On metadata change the state machine tries to re-apply



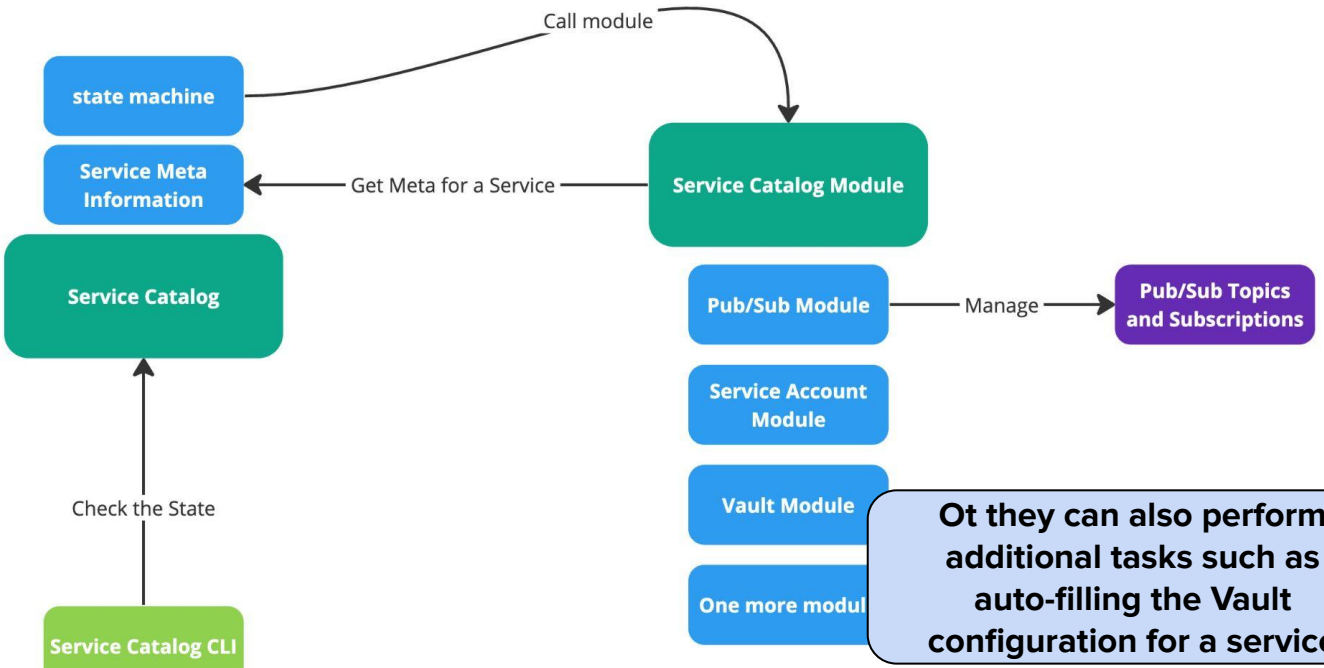
Service Catalog



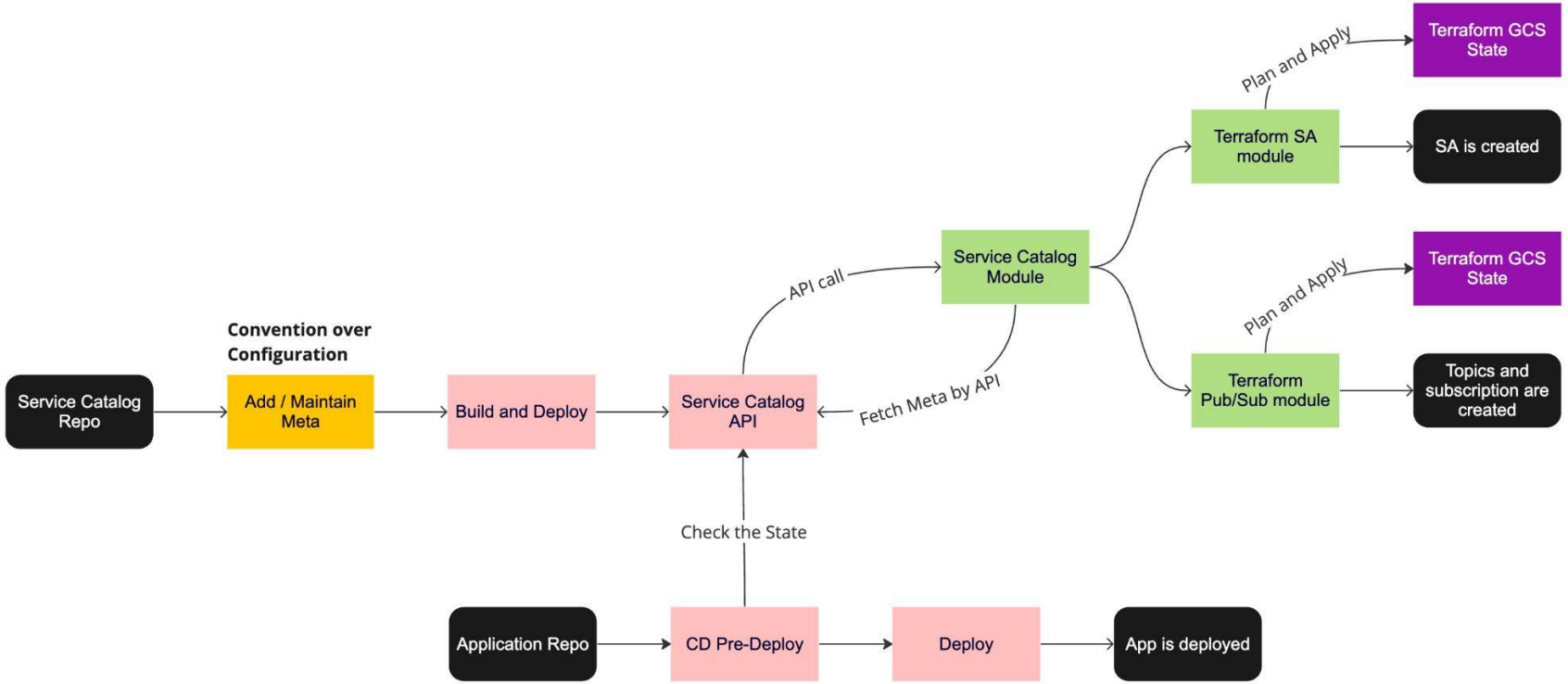
Service Catalog



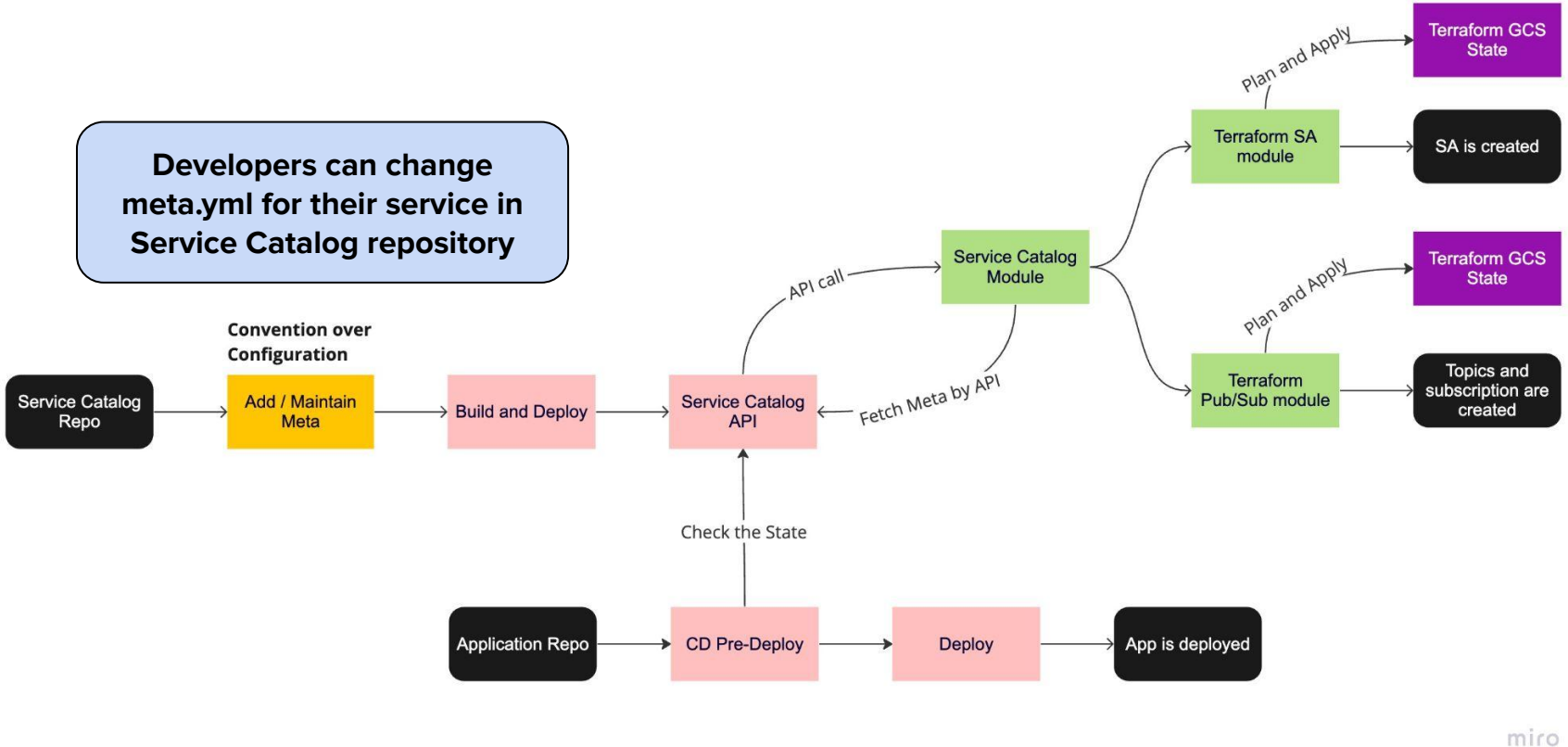
Service Catalog



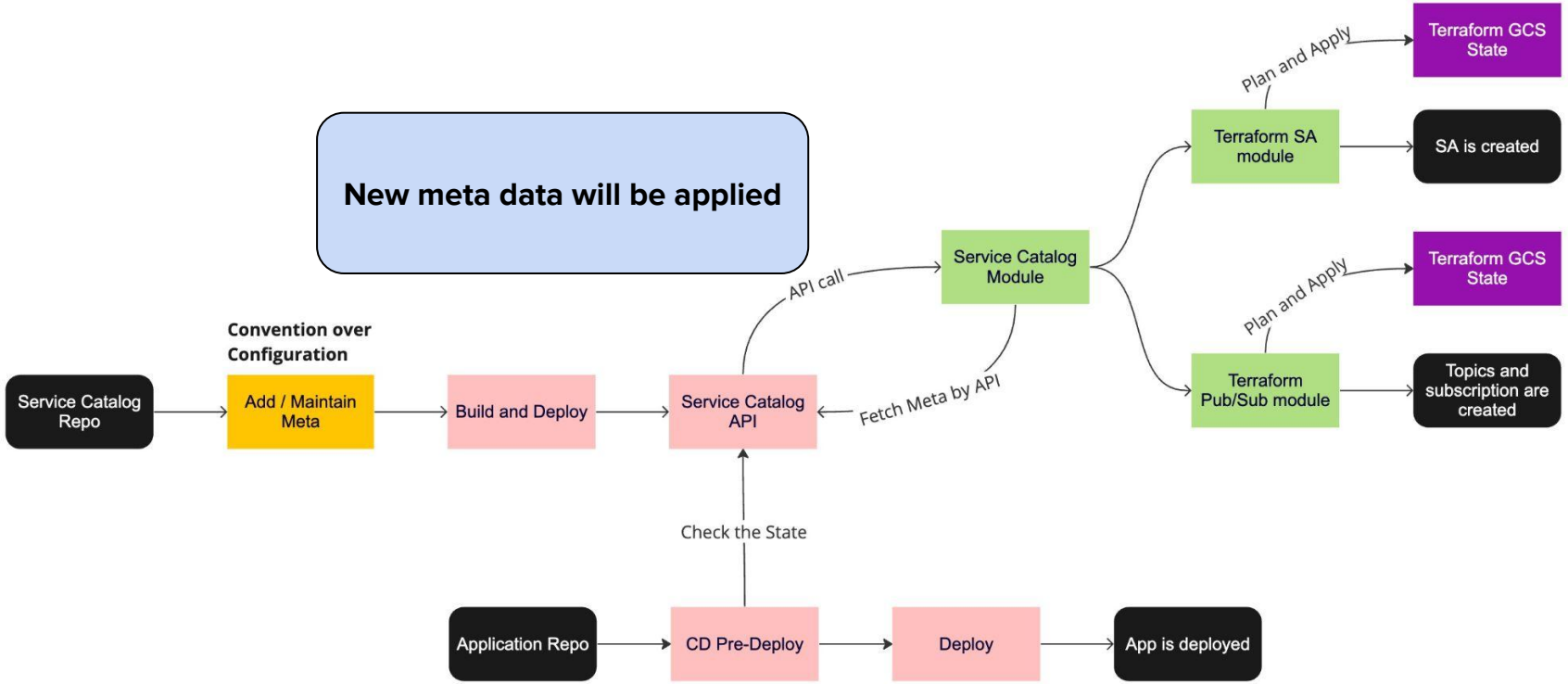
Deployment Process



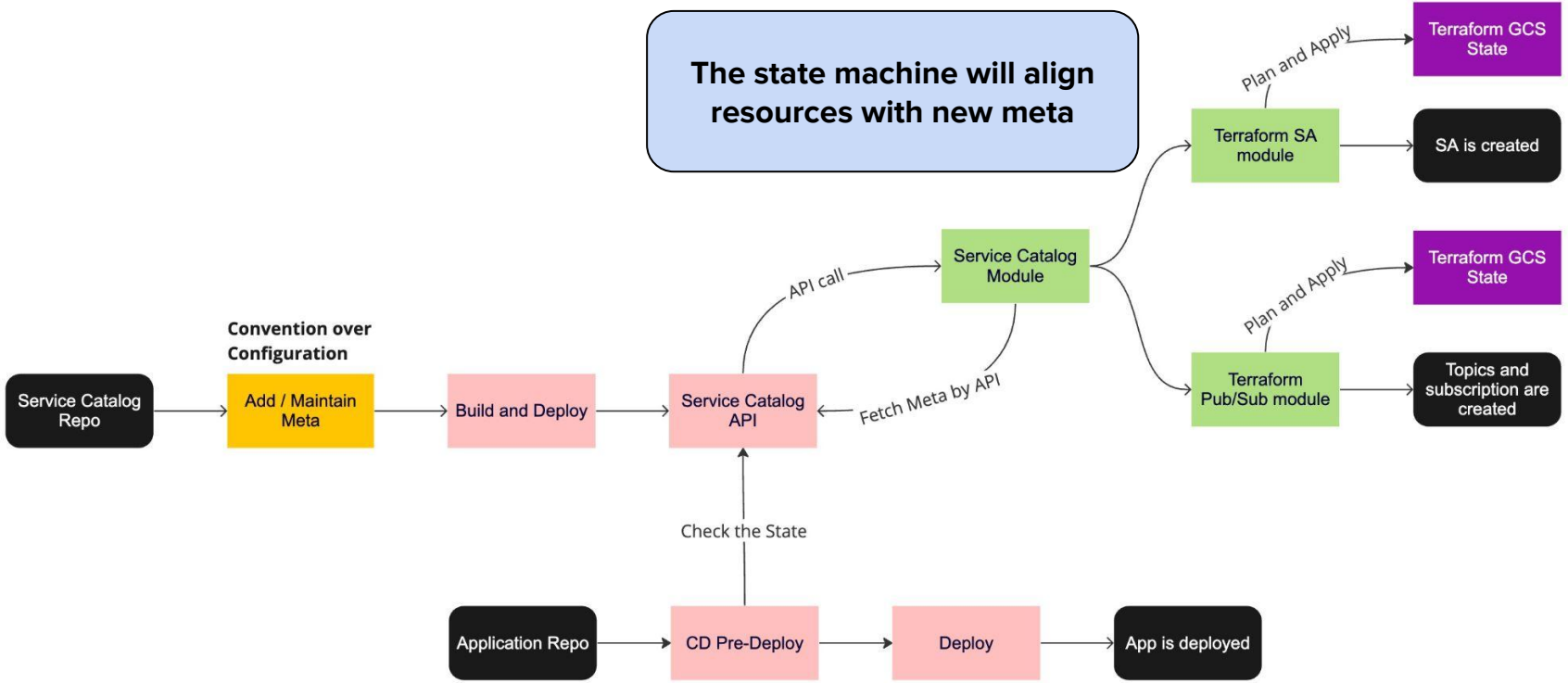
Deployment Process



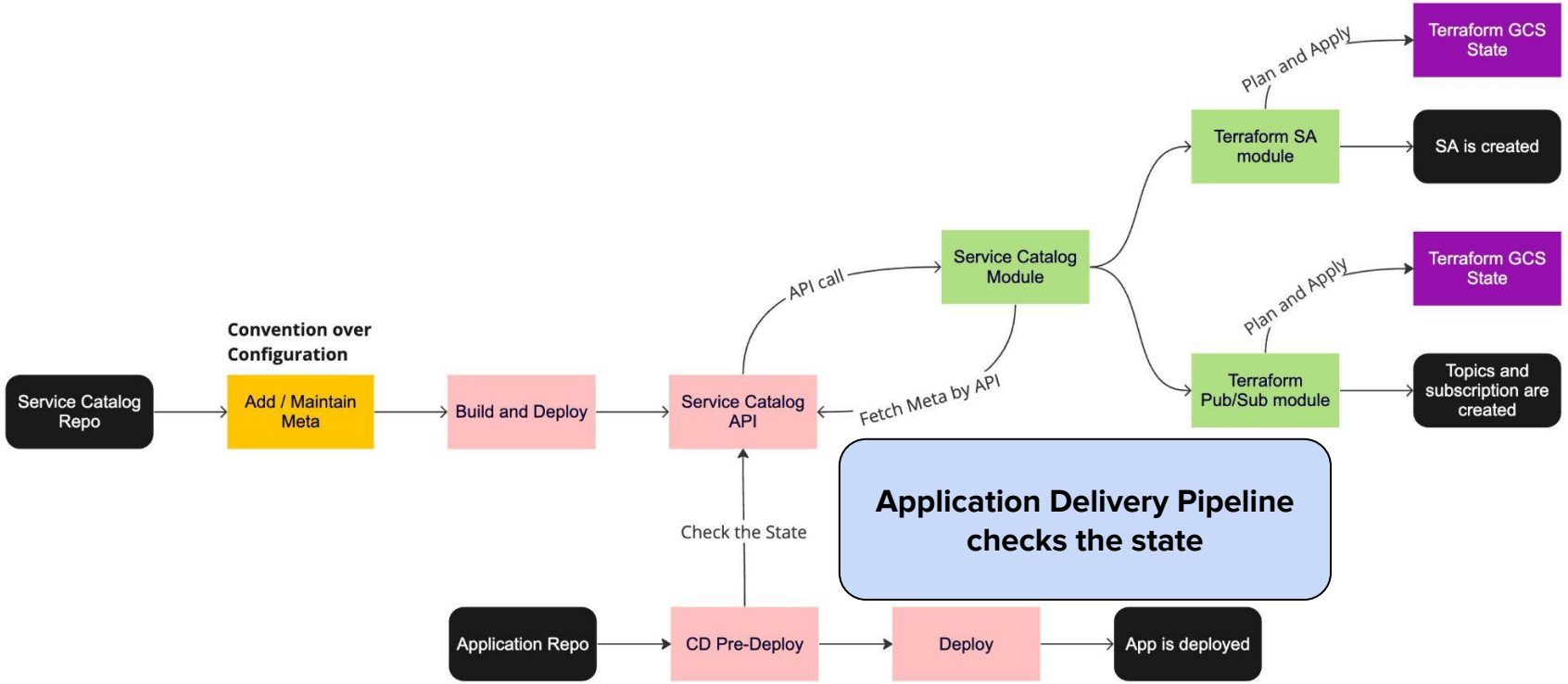
Deployment Process



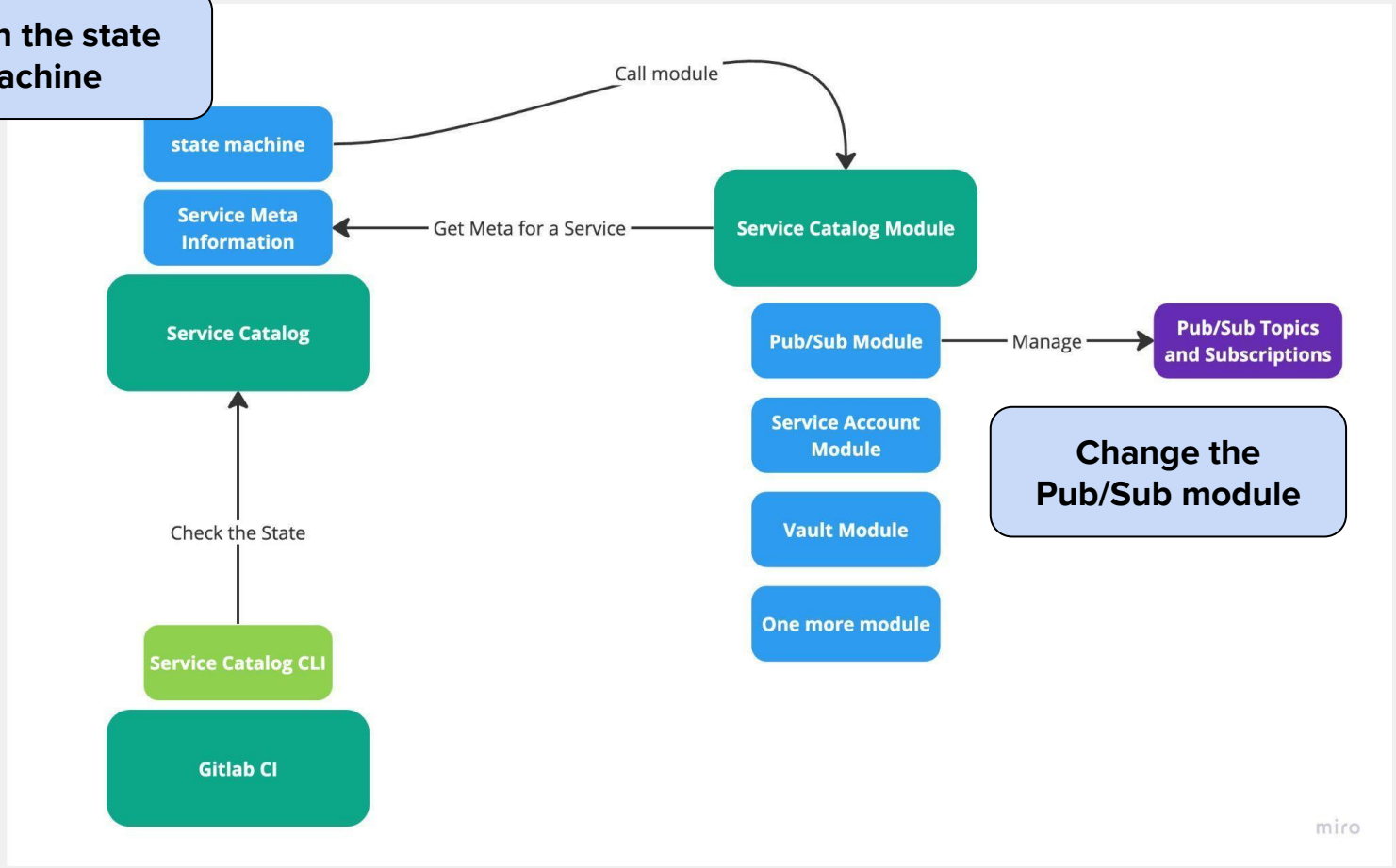
Deployment Process



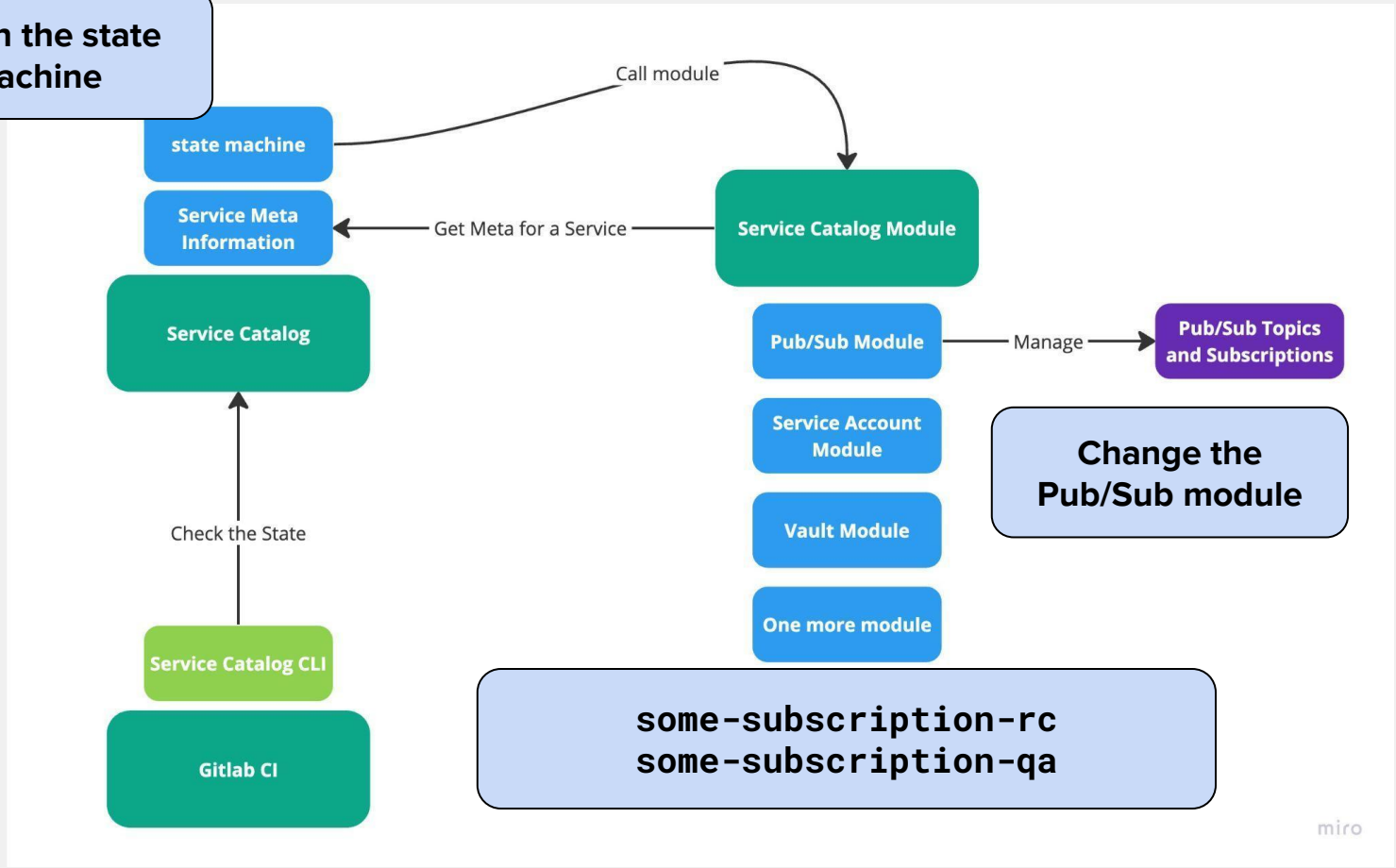
Deployment Process



Static Subscriptions



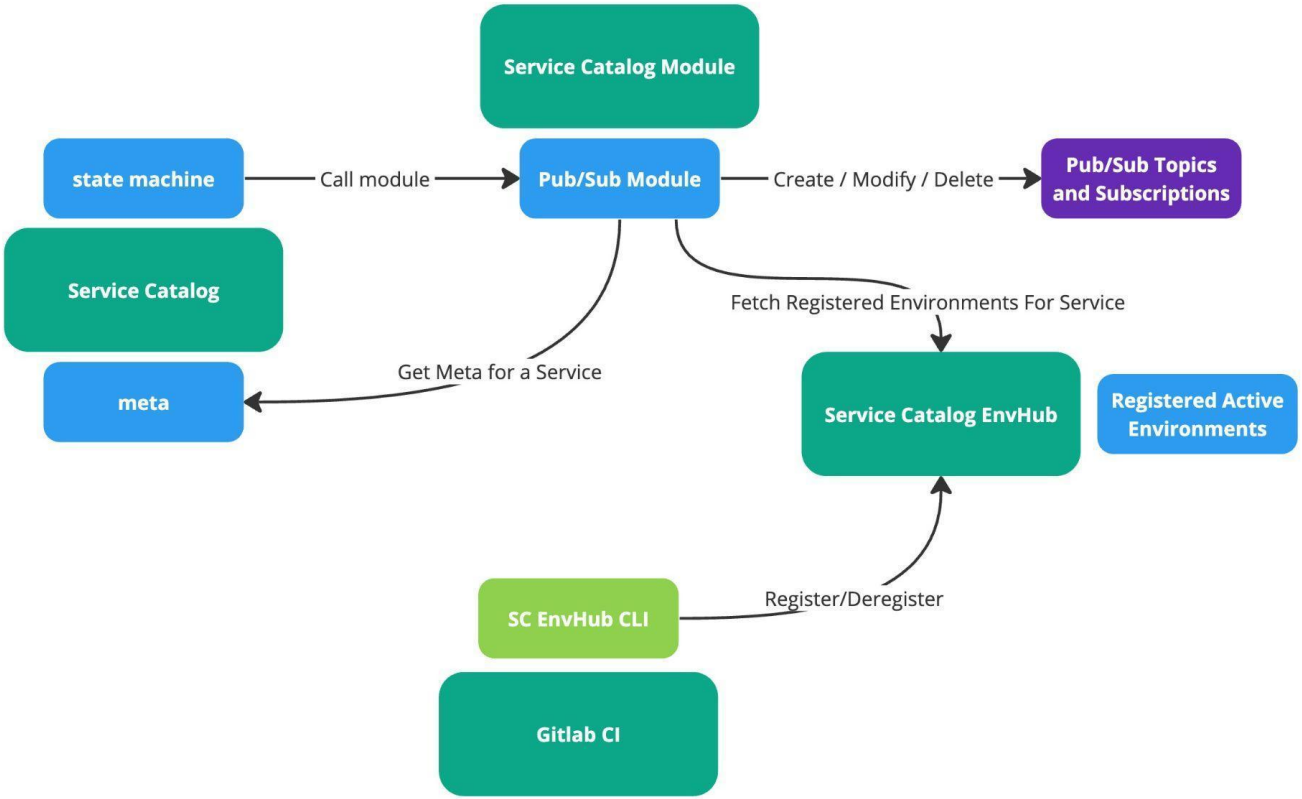
Static Subscriptions



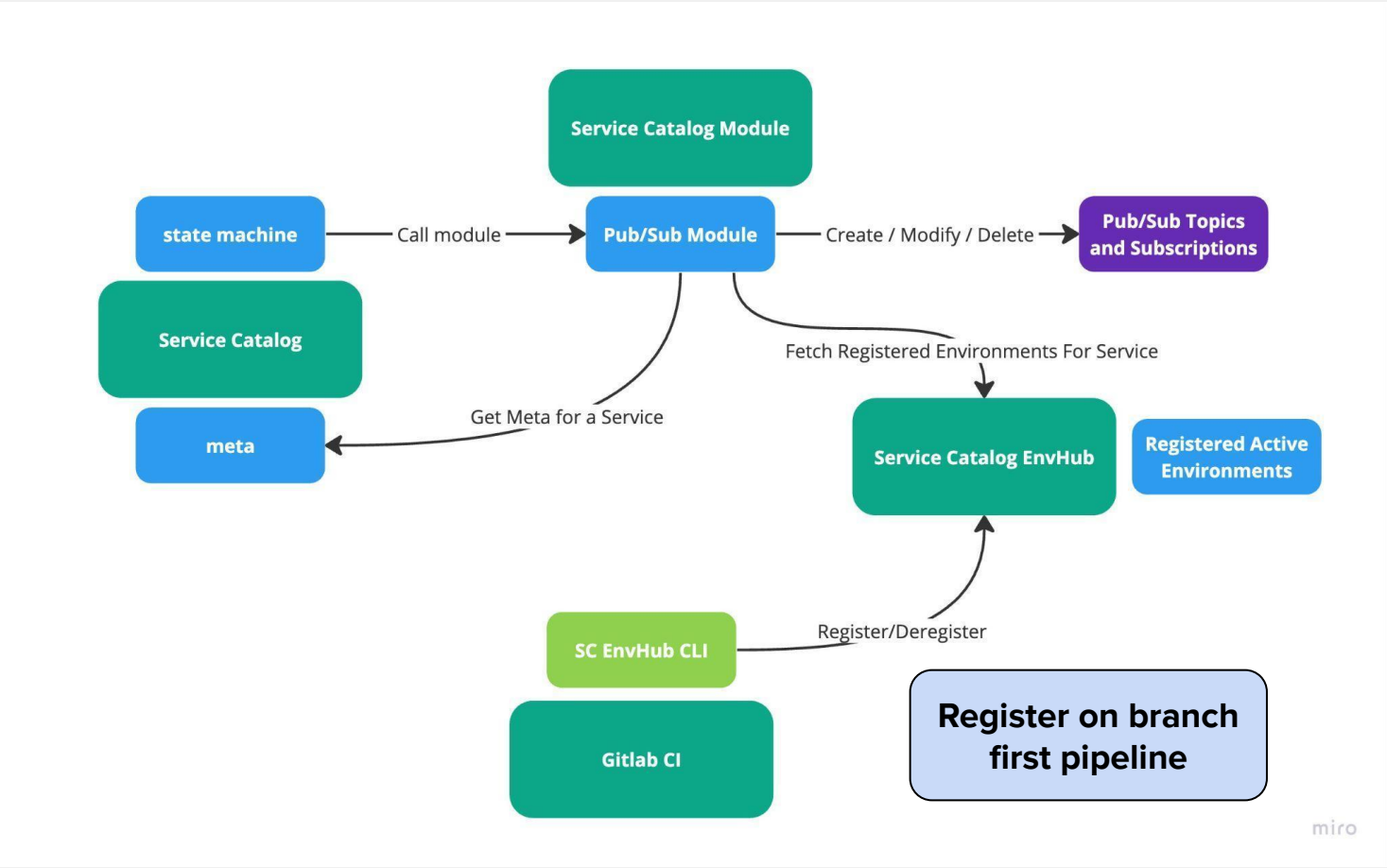
Issues to address

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

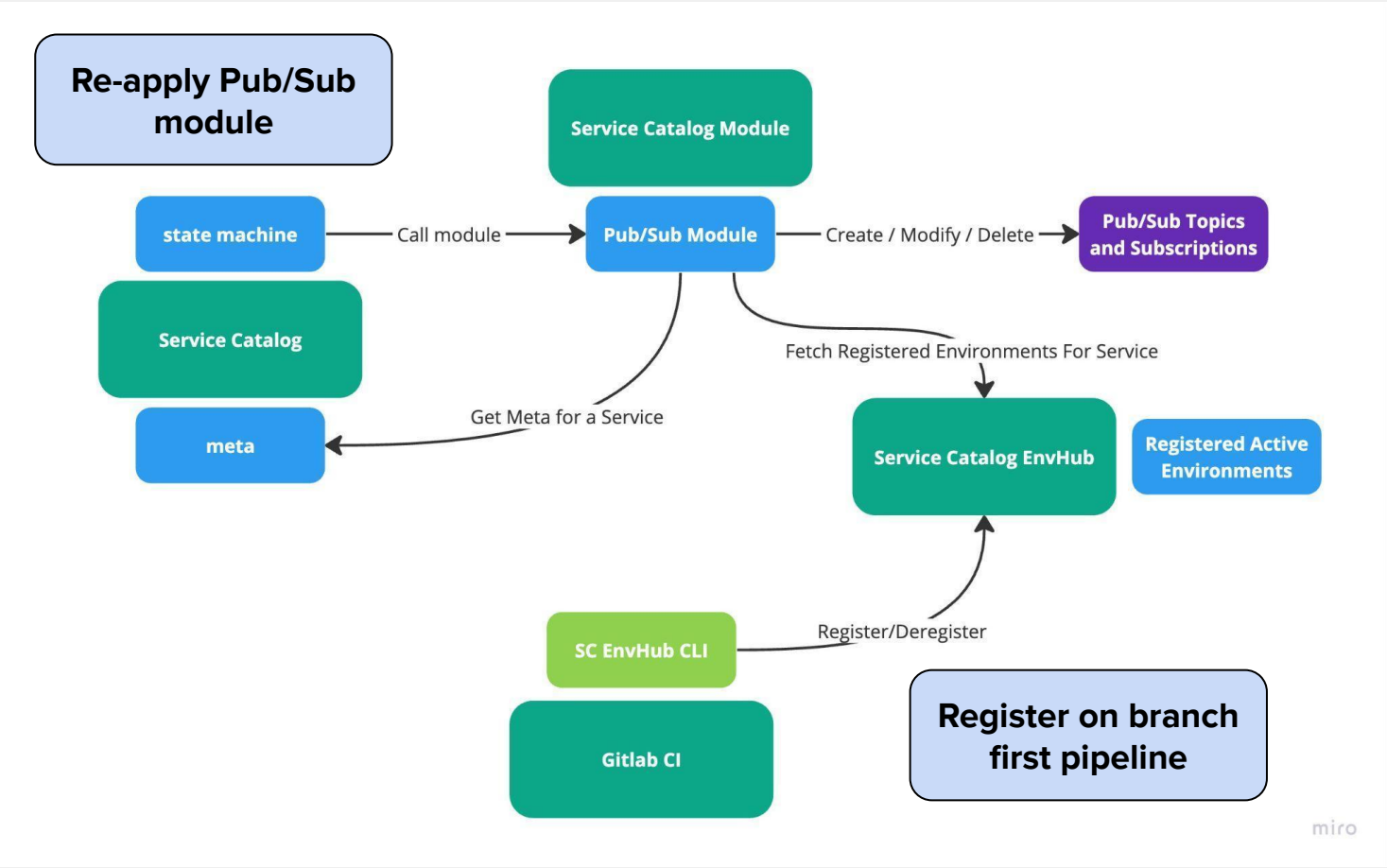
Dynamic Subscriptions



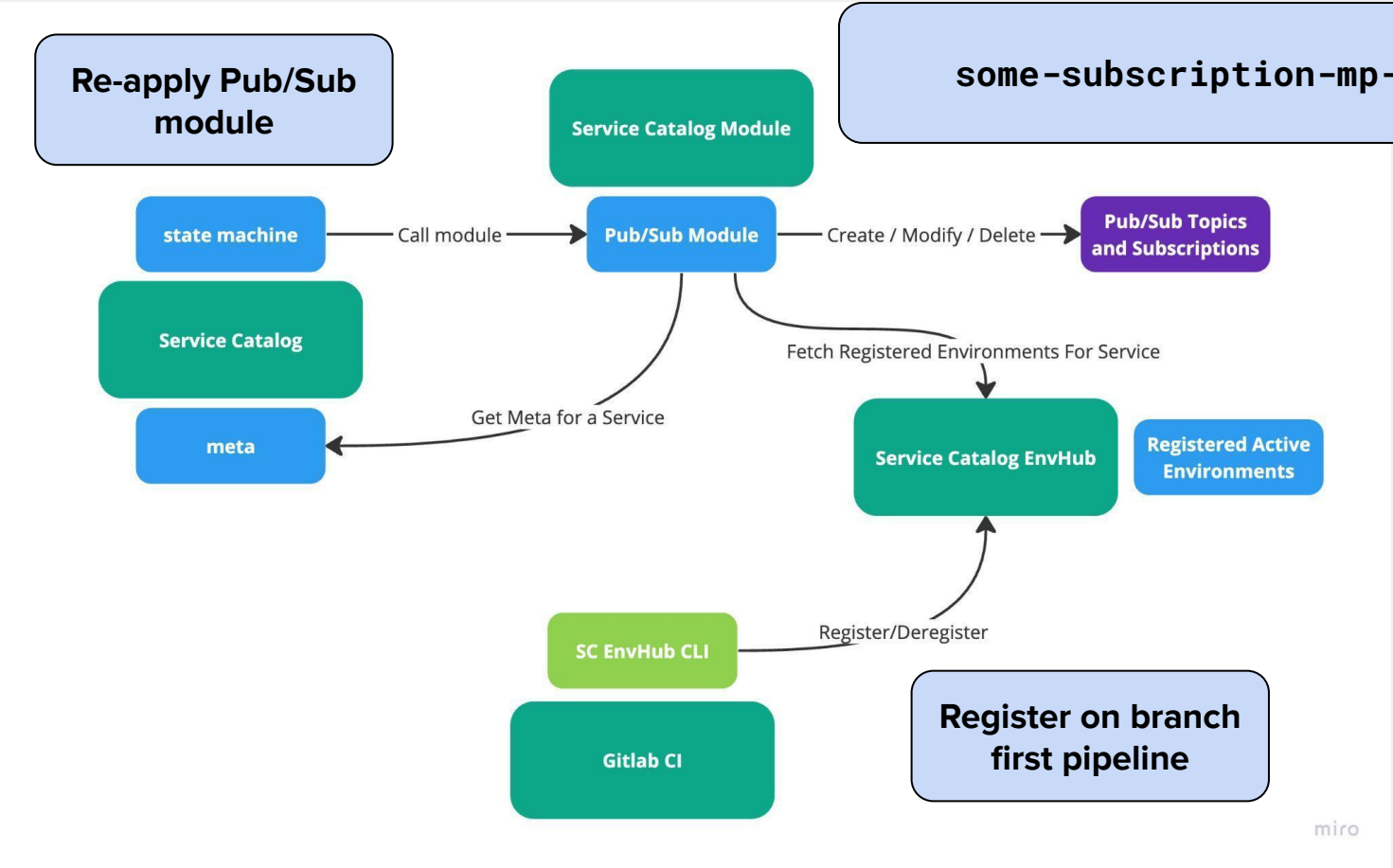
Dynamic Subscriptions



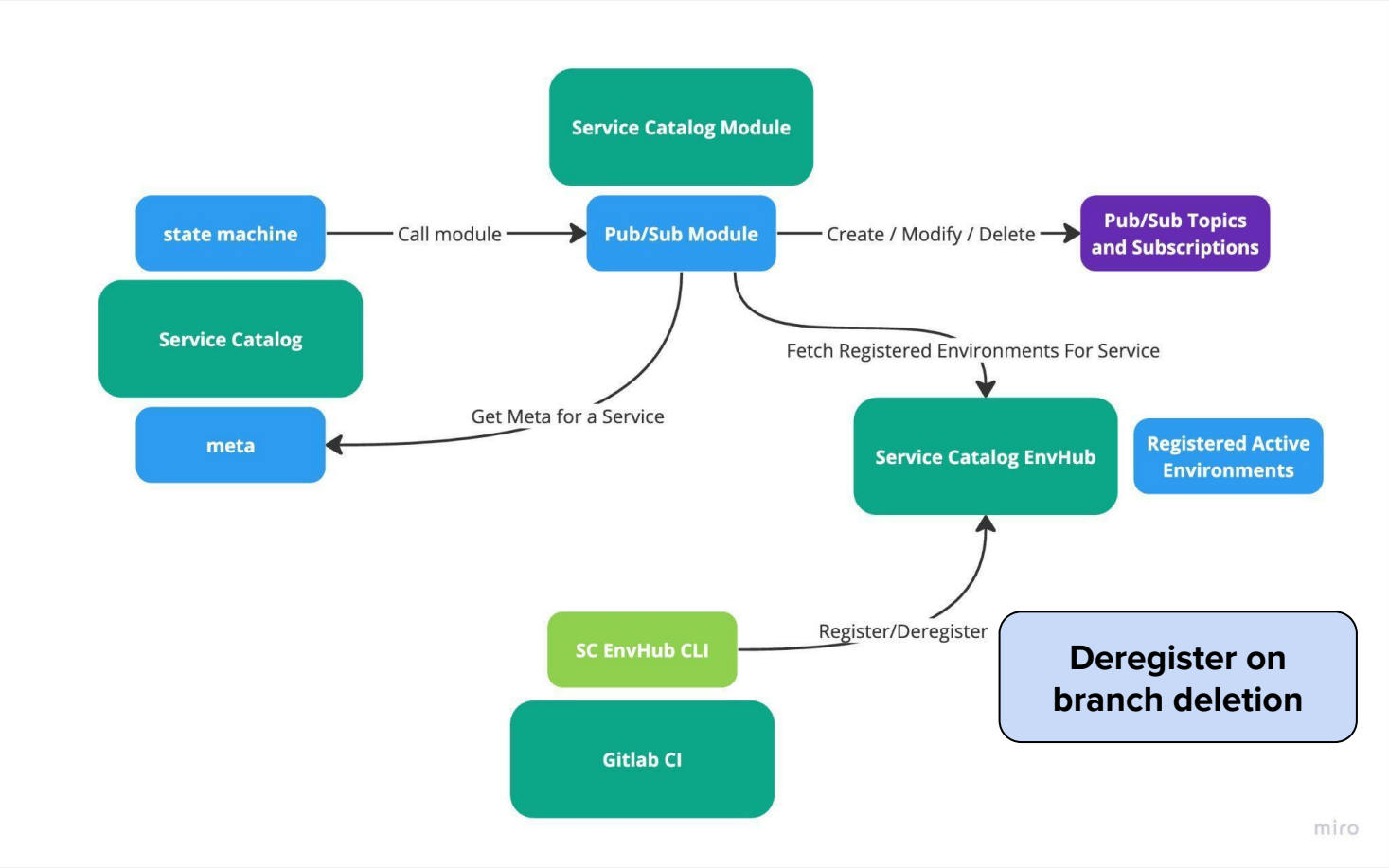
Dynamic Subscriptions



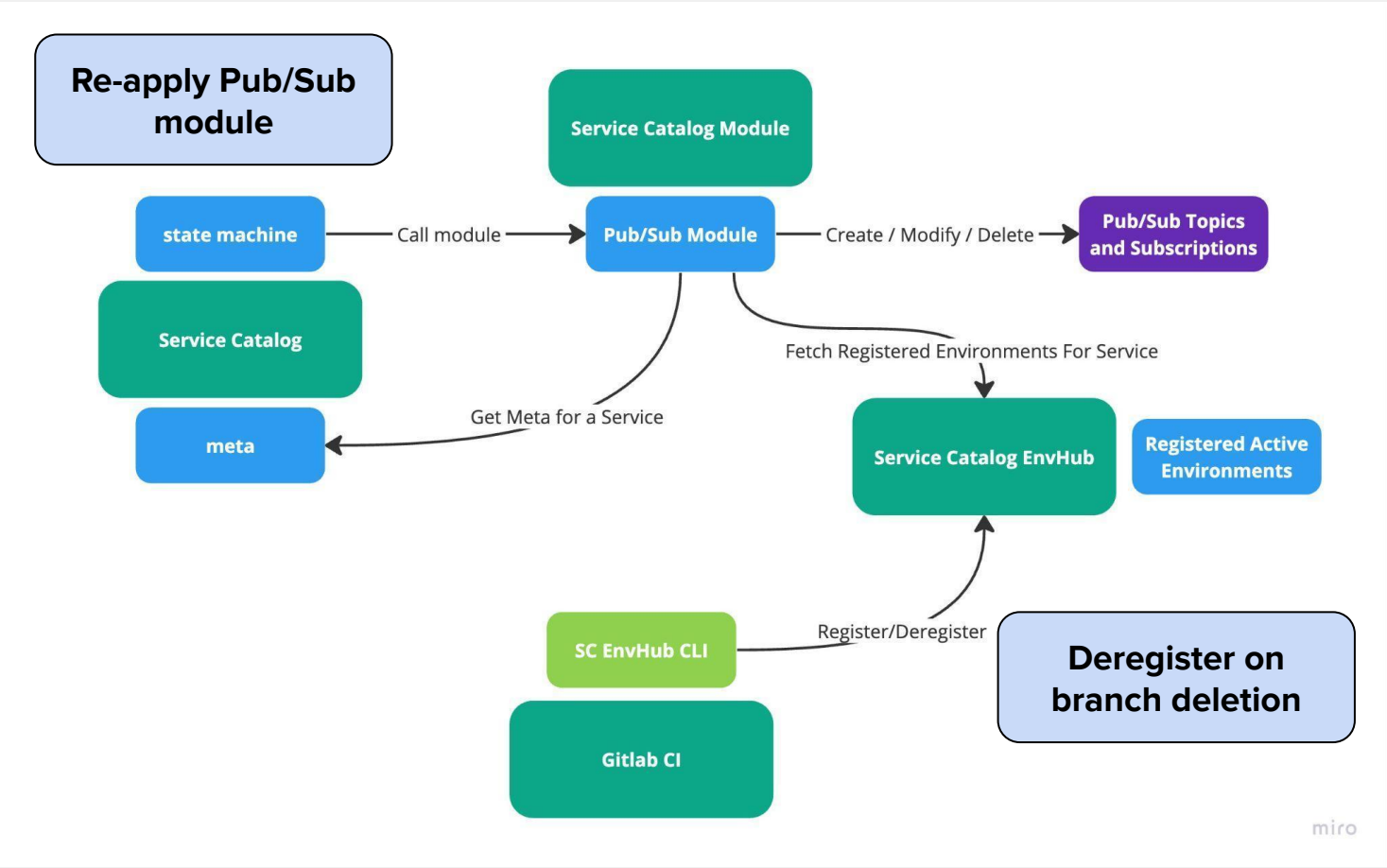
Dynamic Subscriptions



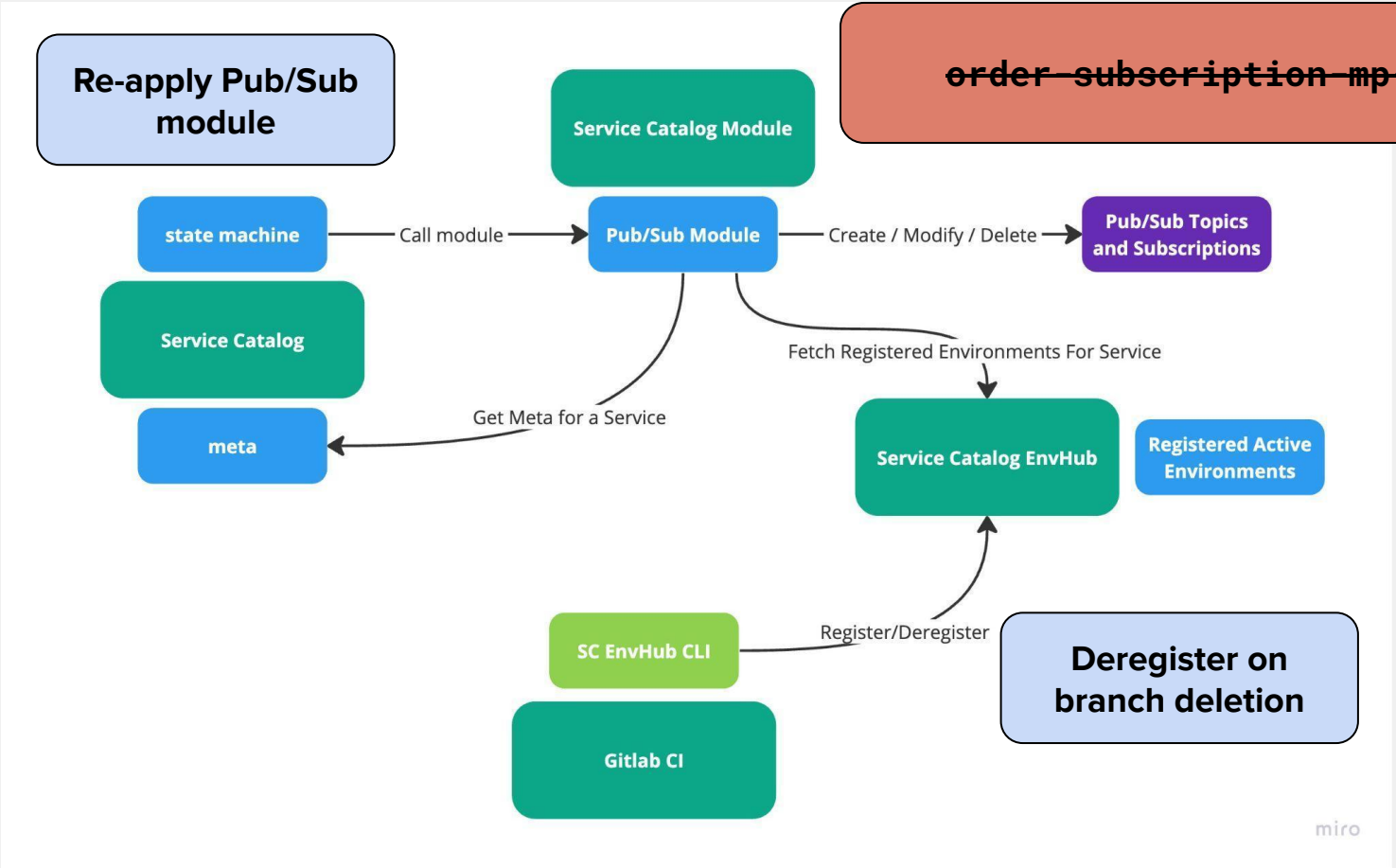
Dynamic Subscriptions



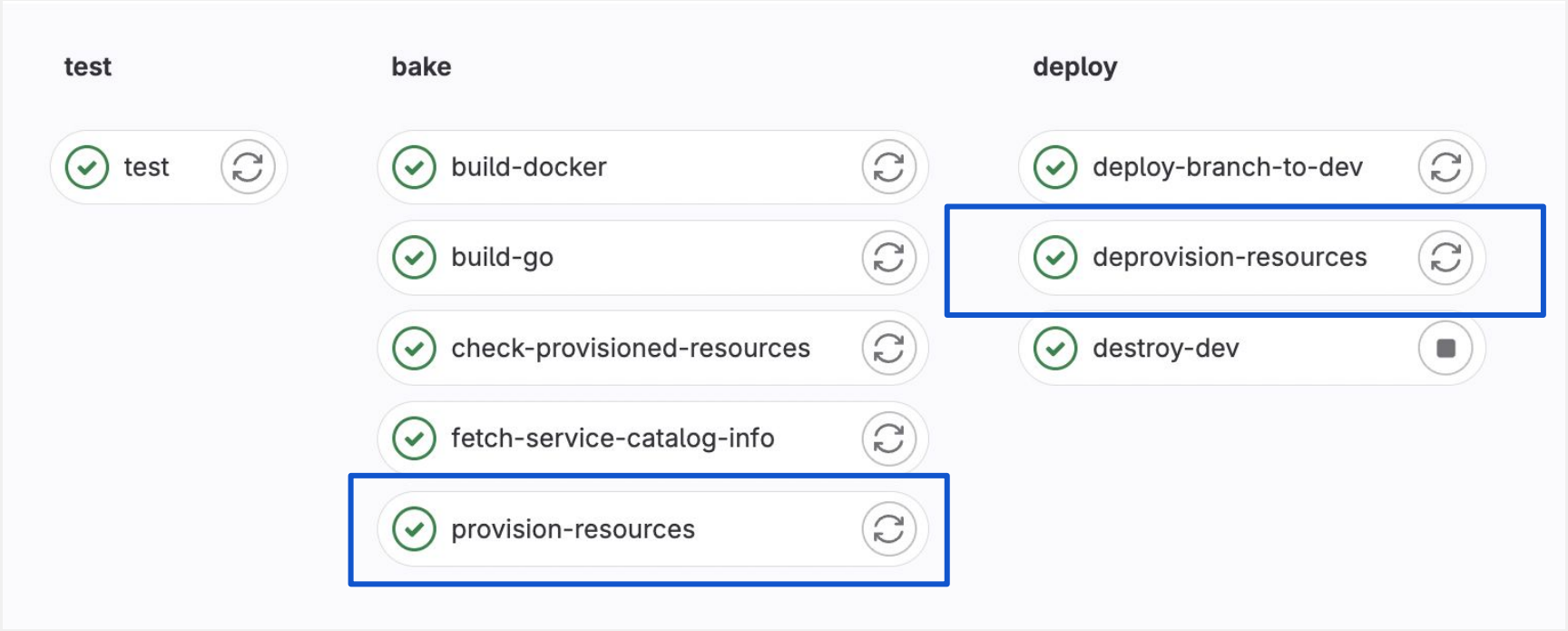
Dynamic Subscriptions



Dynamic Subscriptions



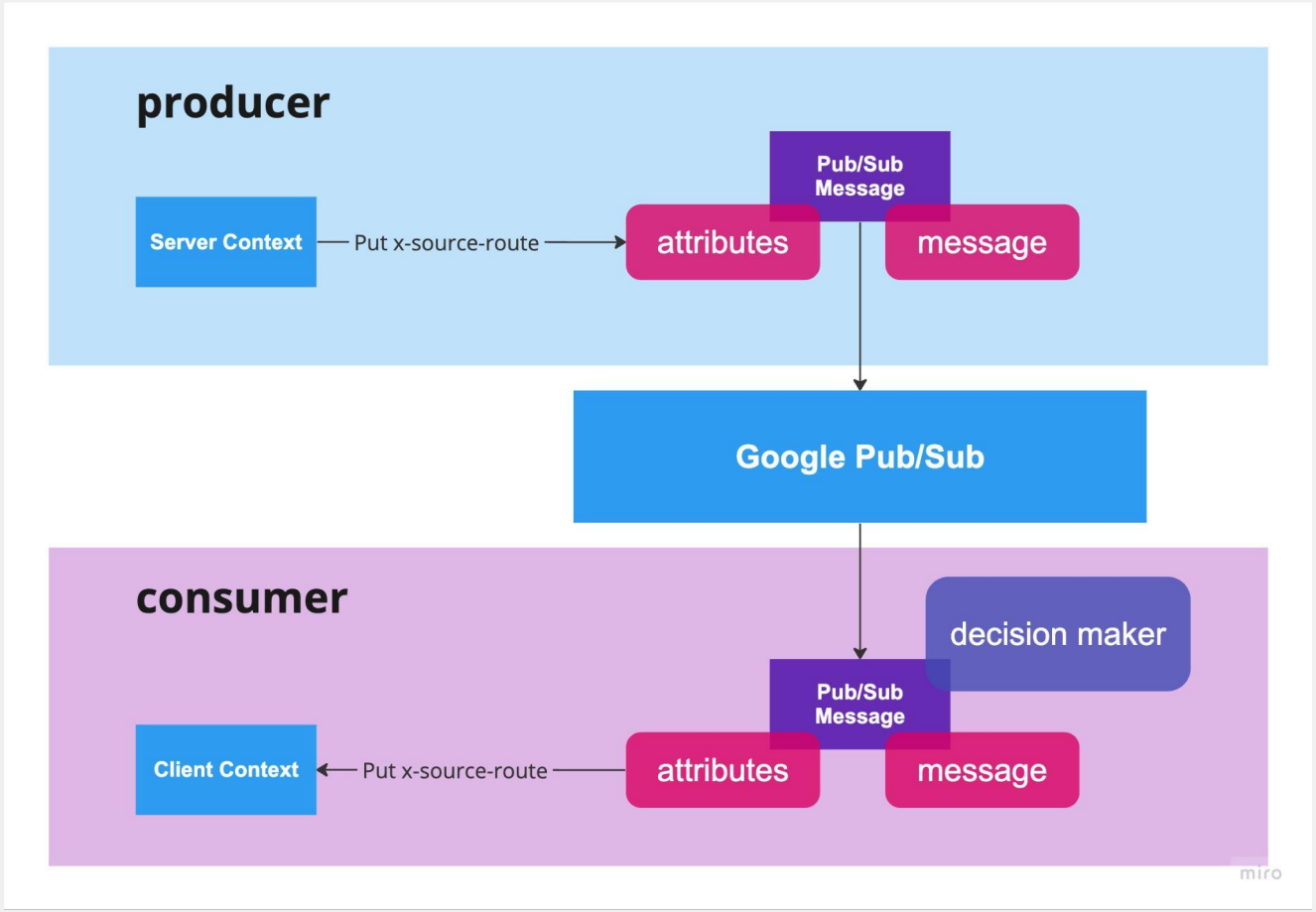
Deployment Process



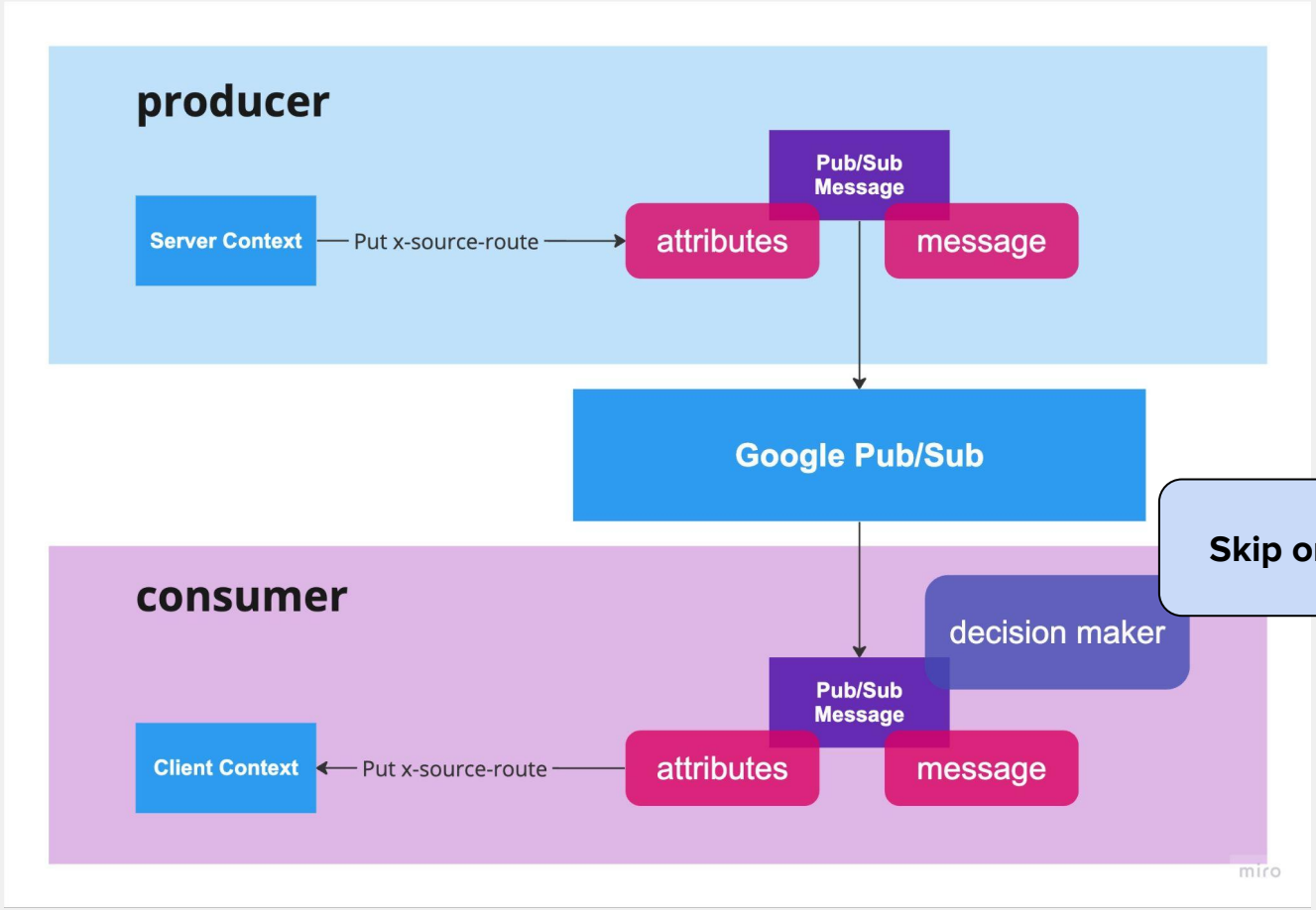
Issues to address

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

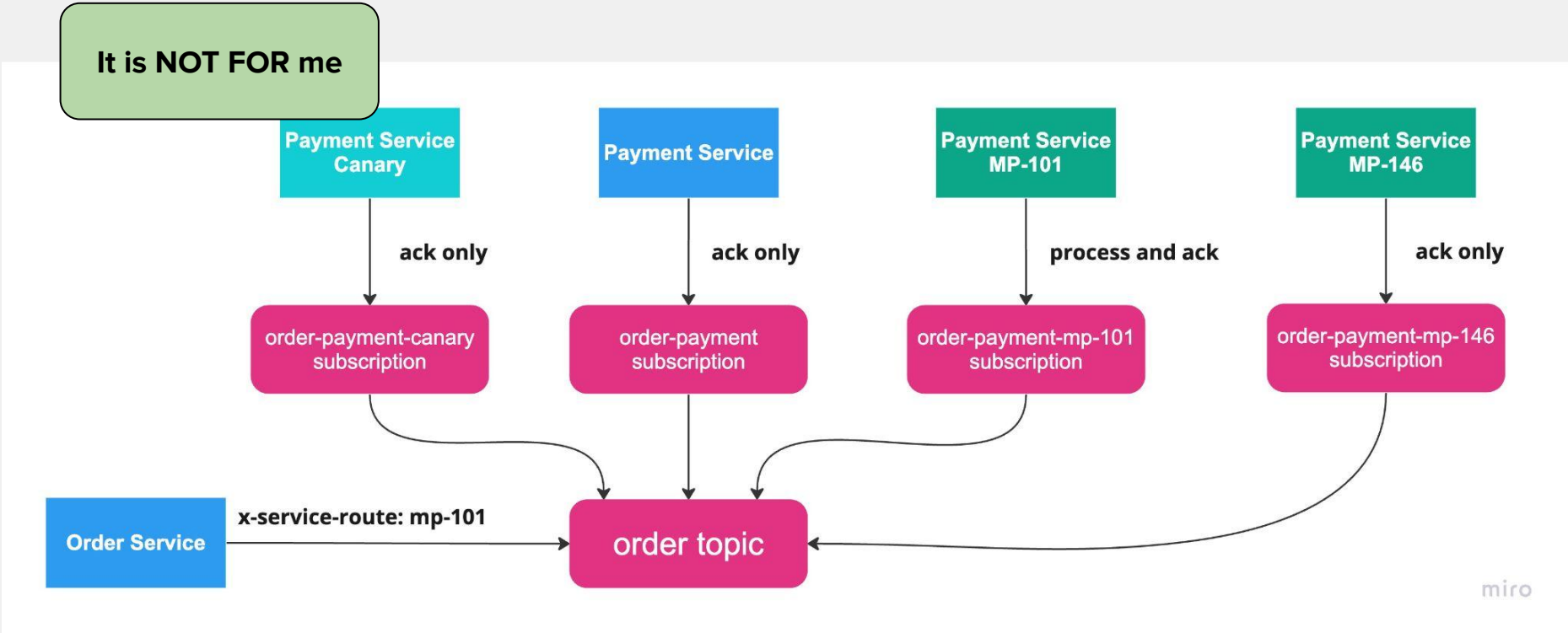
Common Library



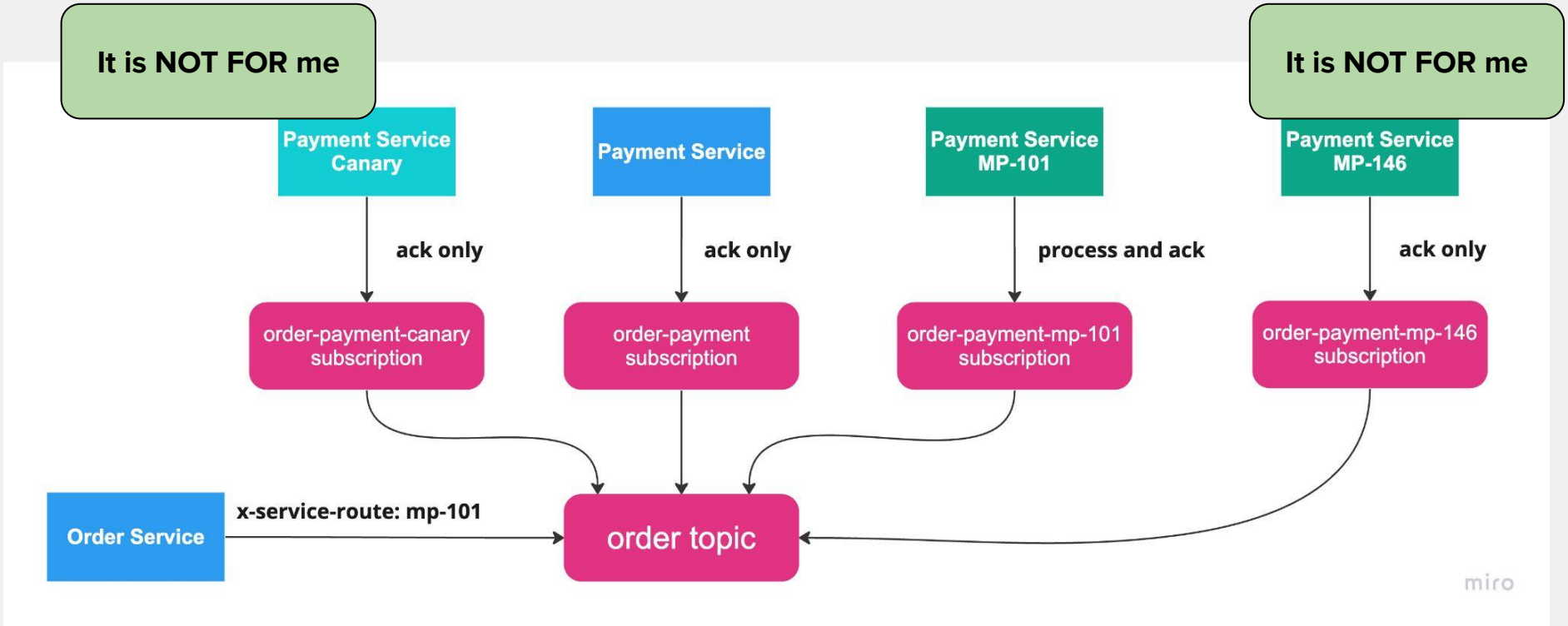
Common Library



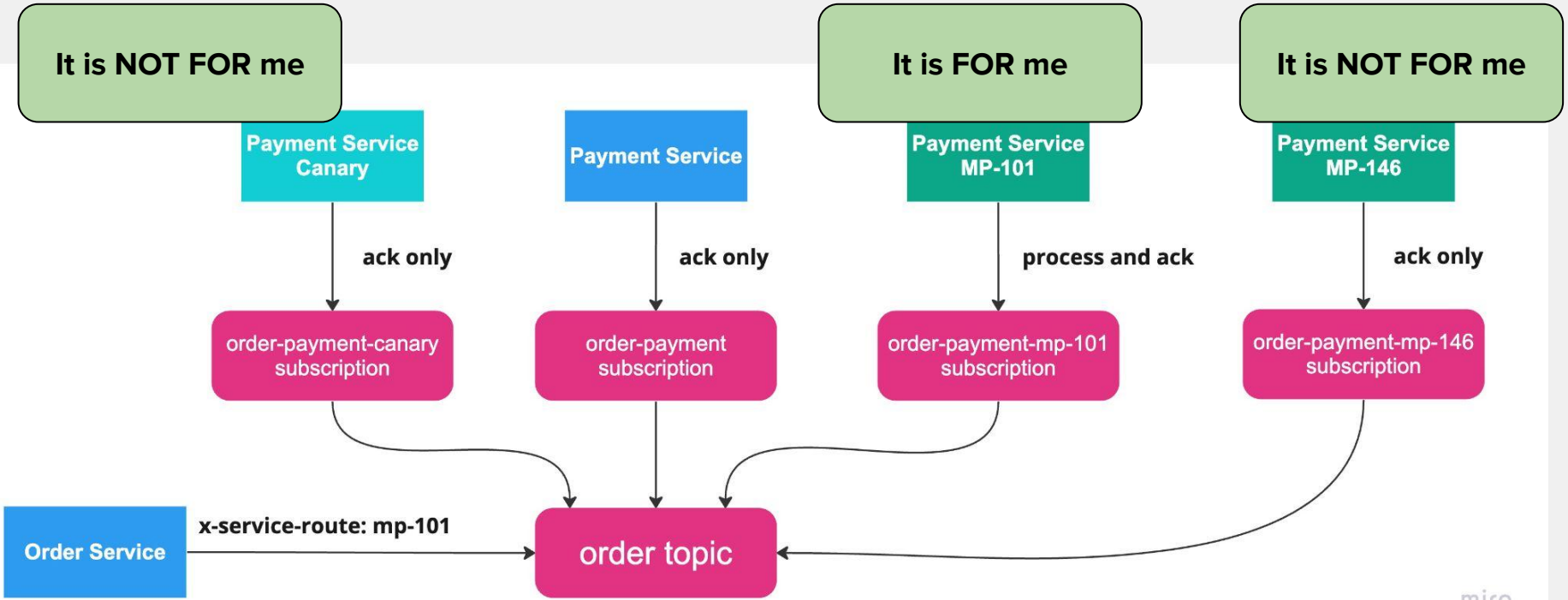
Common Library: Decision Maker



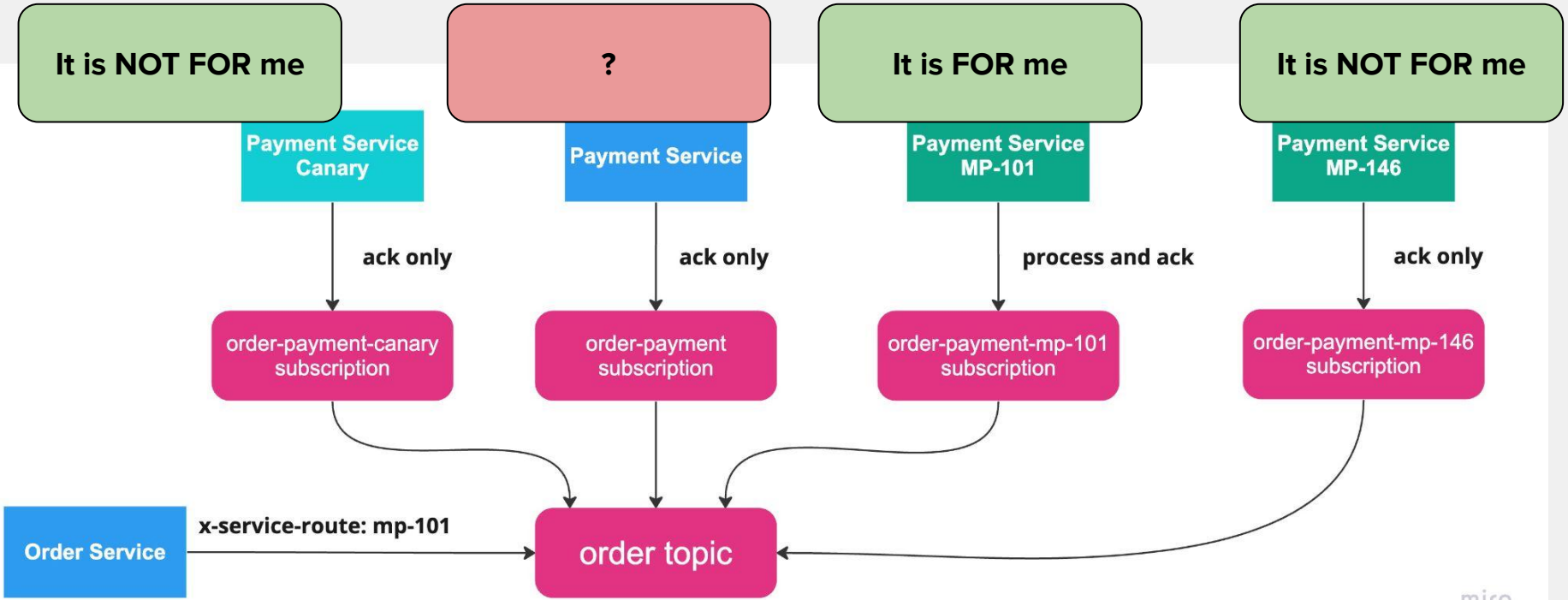
Common Library: Decision Maker



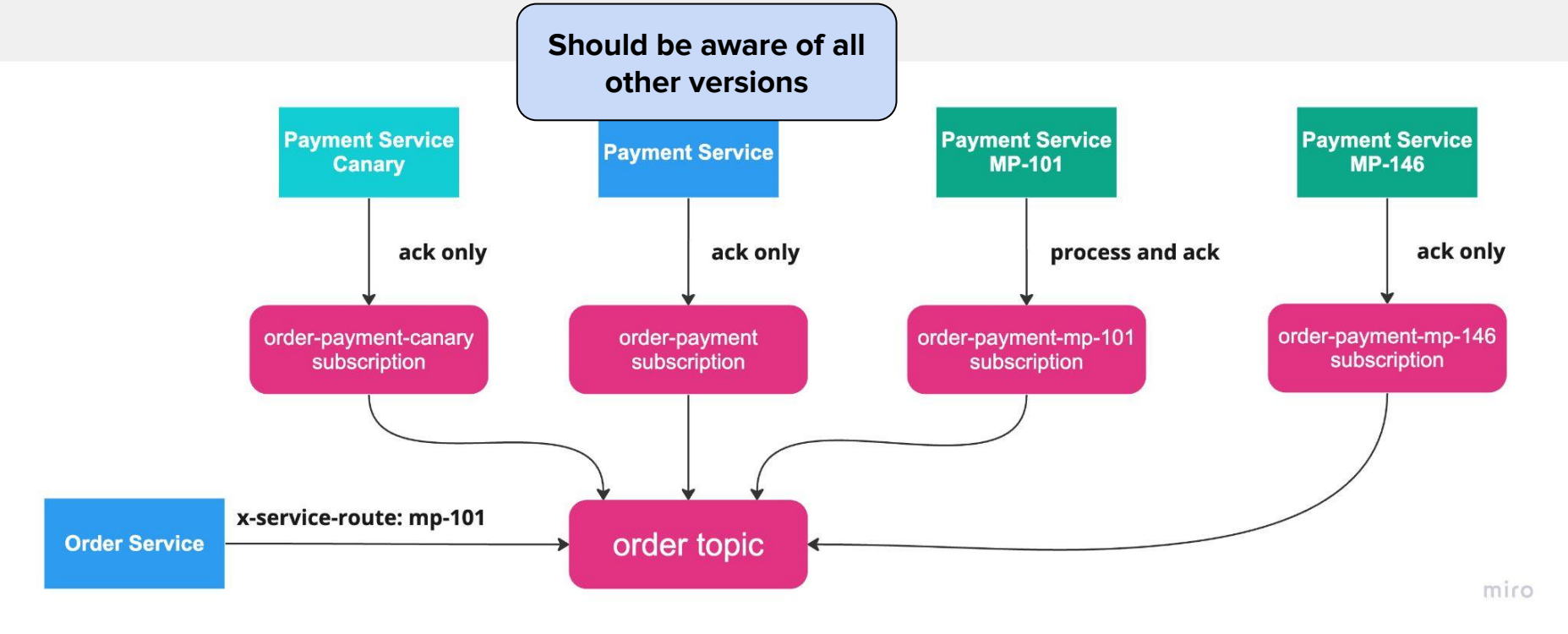
Common Library: Decision Maker



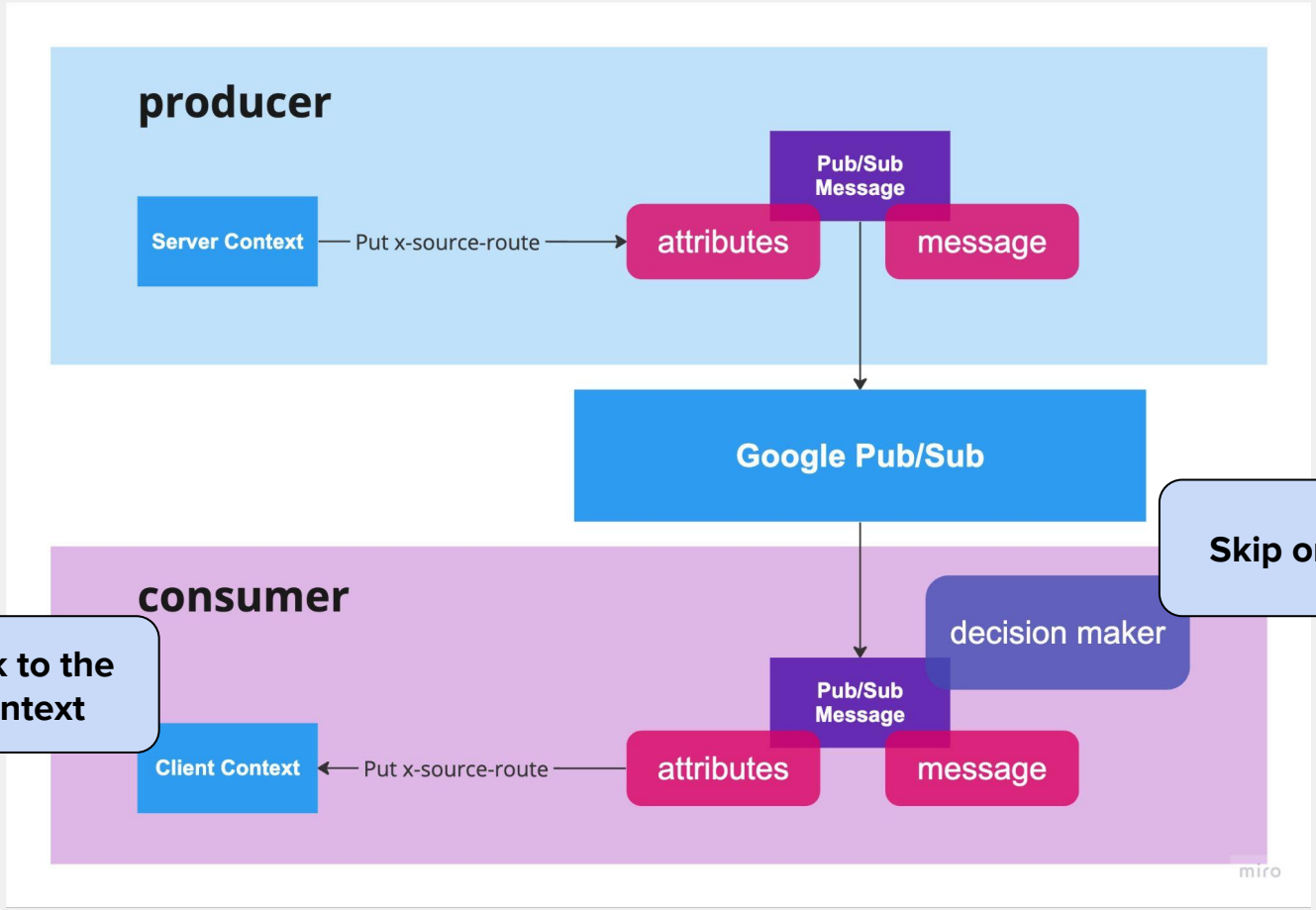
Common Library: Decision Maker



Common Library: Decision Maker



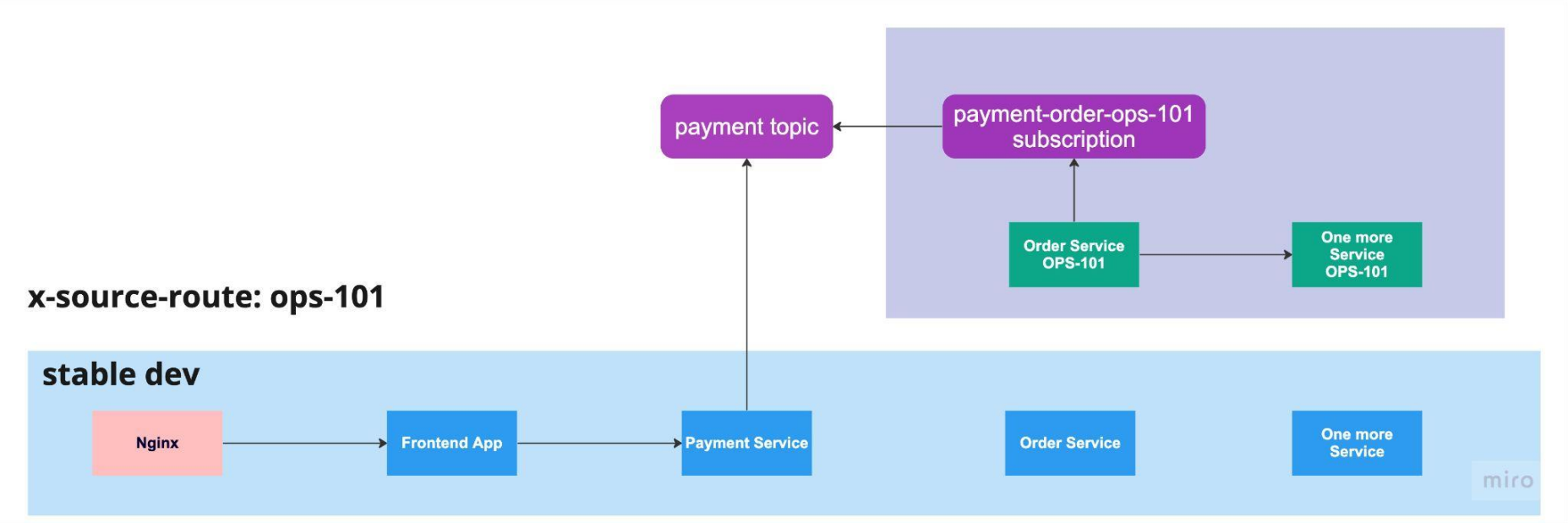
Common Library



Skip or Process?

Put it back to the client context

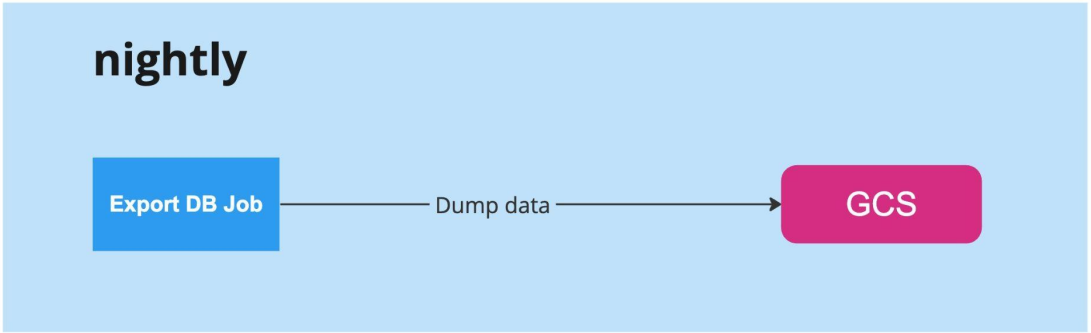
Complex Scenarios Supported



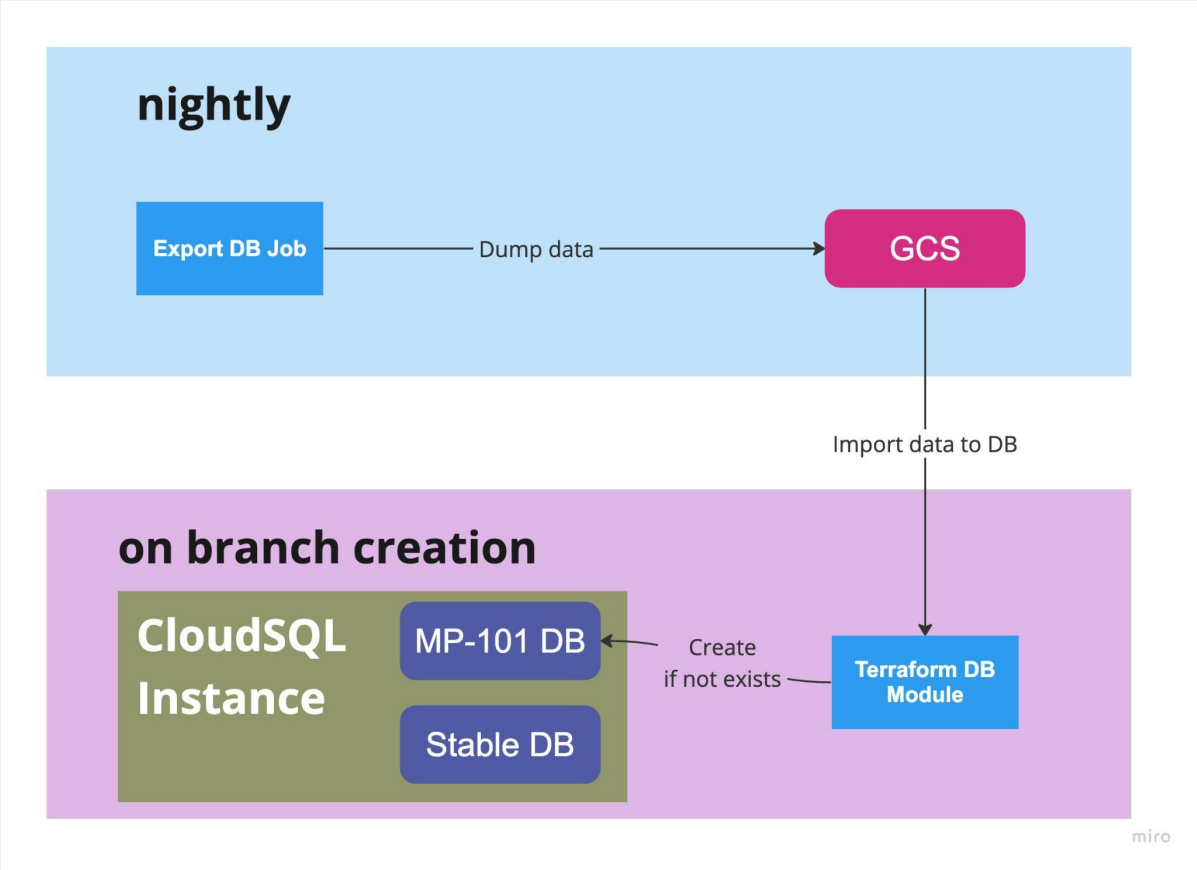
Issues to Address

- Separated DB for branches
  - the same approach as for subscriptions

Separated DBs Schema



Separated DBs Schema

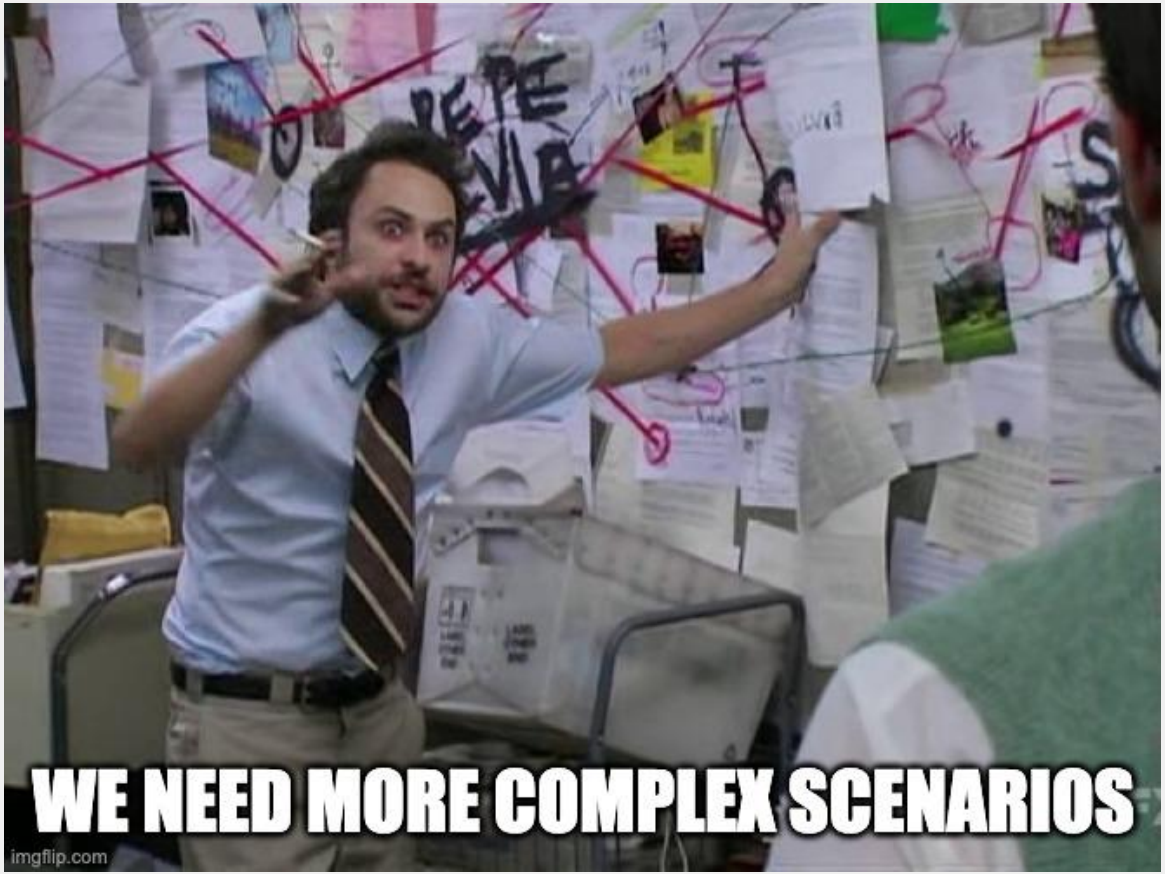


Issues to Address

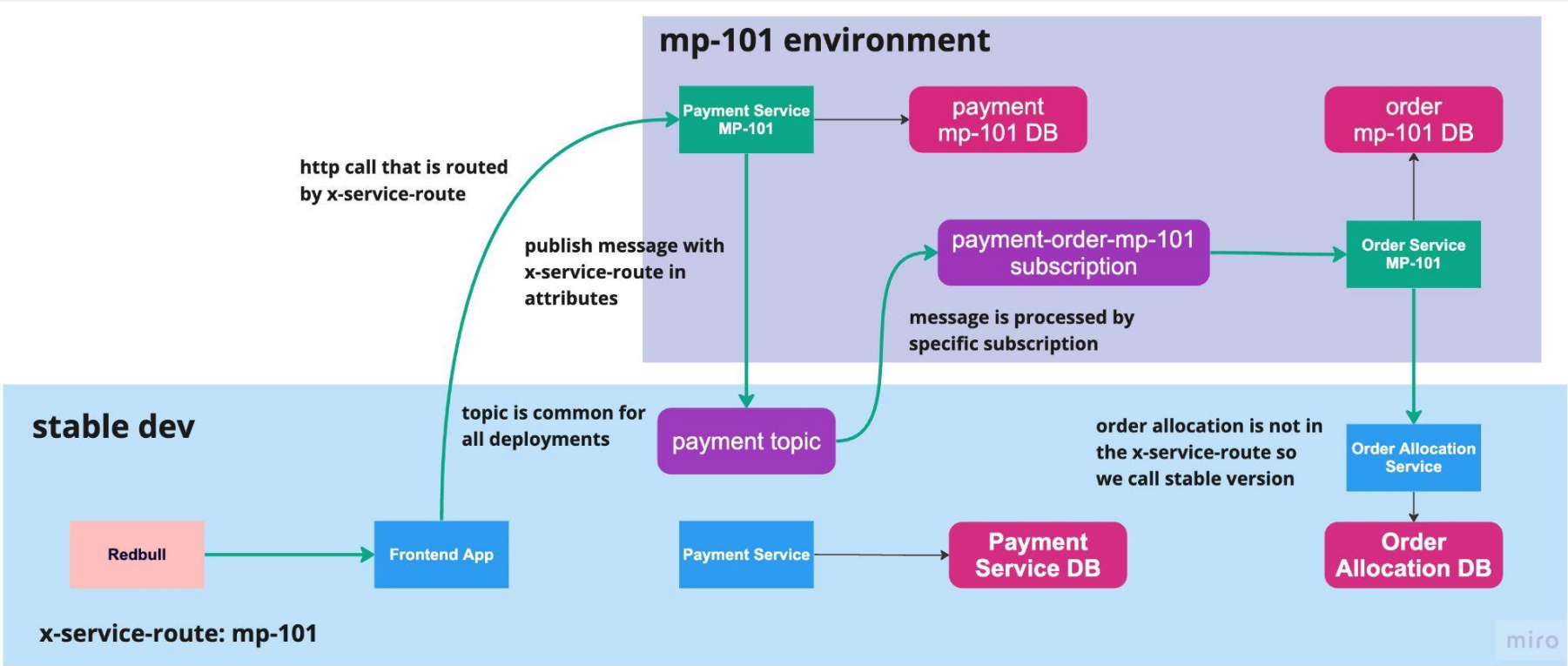
- Separated DB for branches
  - the same approach as for subscriptions
  - **the incomplete data**

Chapter 4: Ephemeral Environments

Welcome to Real Life



Welcome to Ephemeral Environments



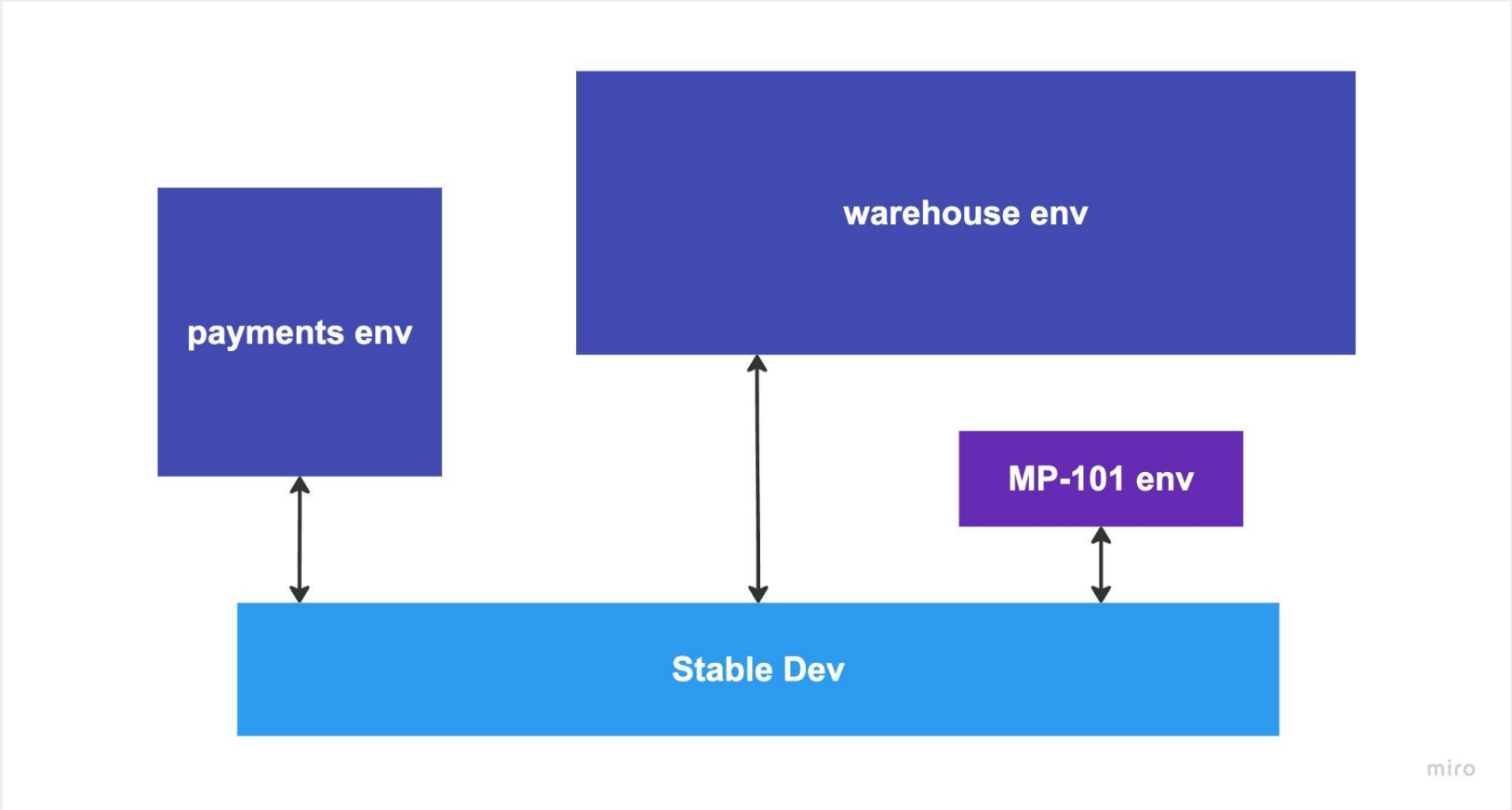
Types of Ephemeral Environments

- 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: Some Conclusions

Benefits

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

Drawbacks

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

Questions?

@aatarasoff

@aatarasoff

@aatarasoff

@aatarasoff