
Filtering and scaling of Eiffel

Mattias Linnér, Ericsson

Magnus Bäck, Axis Communications

Agenda layout

- (Mattias) ..., 45m?
- (Magnus?) ..., time?
 - Routing key vs Binding key and filtering (?)
 - Pitfalls of ER dependencies when consuming events (?)
- ~~(Tobias) Eiffel Collector, 15m~~
 - Propose to put this as a separate session earlier in the agenda

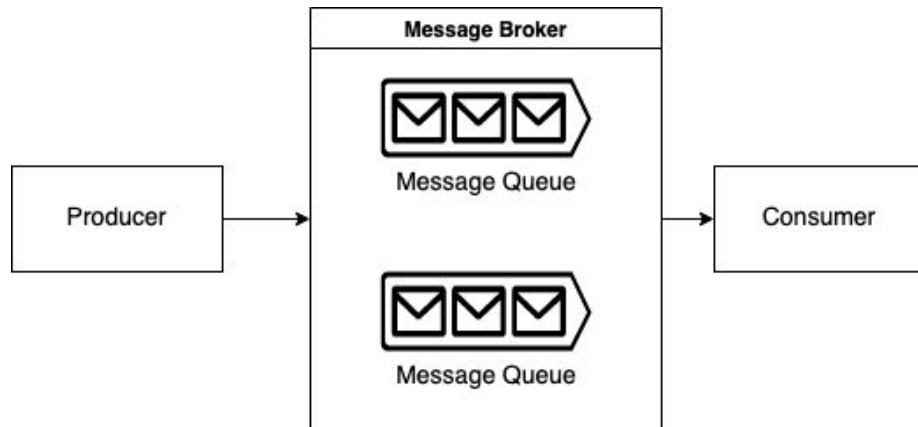
Agenda

- Message brokers
- RabbitMQ routing keys
- Scaling
- Reflections on Ericsson setup
- Reflections on Axis's setup
- Useful patterns when processing events

Message brokers

Choice of Message Broker

- Transports the message from the publisher to the consumer
- Different brokers different setups
- Traditionally RabbitMQ
 - Was widely available when we started



RabbitMQ – lock-in?

No

- Eiffel protocol as such does not stipulate transport
- PR to show how to use Eiffel on CloudEvents

Yes

- Eiffel Sepia talks about RabbitMQ
- Some tools in the Eiffel community does use RabbitMQ

RabbitMQ – how much do you need?

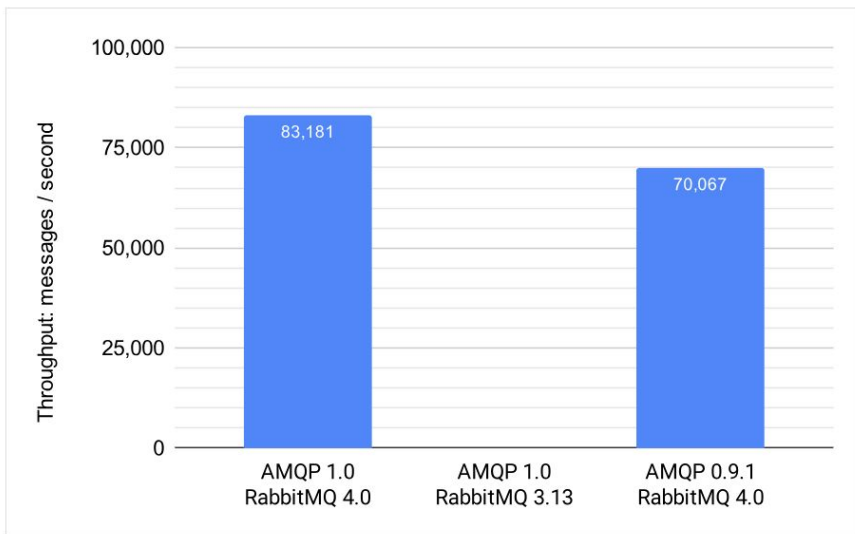


Figure 2: Quorum queue end-to-end message rate

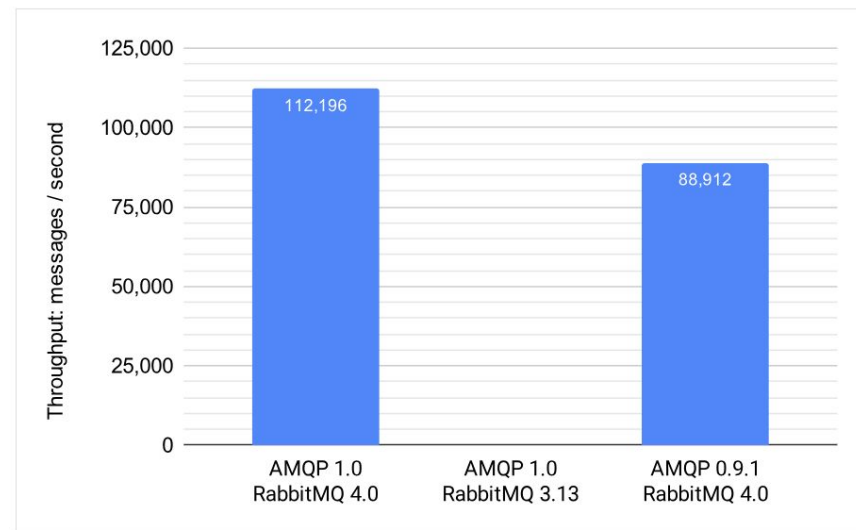


Figure 3: Stream end-to-end message rate

Modern RabbitMQ handles 70k - 110k messages/second
(on a single node 8 cores, 32GB RAM)

Alternatives – RabbitMQ vs Kafka

- RabbitMQ and Kafka offer high-performance message transmission
 - Kafka outperforms RabbitMQ in message transmission capacity.
- Kafka can send millions of messages per second via sequential disk I/O
- RabbitMQ need multiple brokers to reach millions of messages per second
 - Might slow down if RabbitMQ's queues are congested.

RabbitMQ – pros

- Complex routing architecture
 - RabbitMQ provides flexibility for clients with vague requirements or complex routing scenarios.
- Language and protocol support
 - RabbitMQ supports legacy protocols such as MQTT and STOMP
 - RabbitMQ supports a broader range of programming languages compared to Kafka.

RabbitMQ – recommendations

- RabbitMQ blog: *“A happy Rabbit is an empty Rabbit.”*
 - RabbitMQ provides the best throughput and lowest latency when queues are empty.
- Gotchas
 - Don't open and close connections or channels repeatedly.
 - Don't have queues that are too large or too long.
 - Don't use old RabbitMQ/Erlang versions or RabbitMQ clients/libraries.
- RabbitMQ happier with 10 queue à 1000 messages than 1000 queue à 10 messages

<https://www.rabbitmq.com/blog/2020/06/18/cluster-sizing-and-other-considerations>

<https://www.cloudamqp.com/blog/part4-rabbitmq-13-common-errors.html>

Questions?

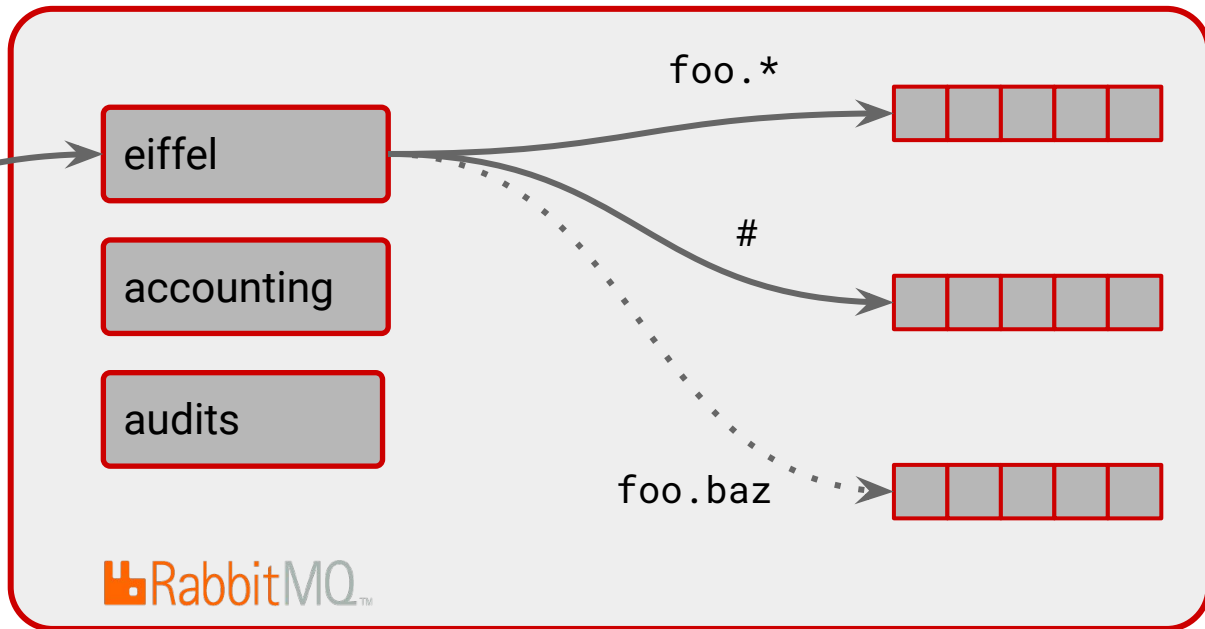
RabbitMQ routing keys

What's a routing key?

Send to "eiffel"
exchange



Routing key: foo.bar



Standard routing key schema for Eiffel events

`eiffel.artifact.EiffelArtifactPublishedEvent.anytag.anydomain`

Fixed prefix

Implementation-
defined family

Type

Implementation-
defined tag

Eiffel
domain of
event

([Full definition](#))

Examples

Producer

```
eiffel._.EiffelArtifactPublishedEvent._._
```

Consumer

```
eiffel.*.EiffelArtifactPublishedEvent.#
```

Compatibility

- Go SDK
- .NET SDK
- eiffel-pythonlib
- Eiffel REMReM Publish
- eiffel-broadcaster
- events-gerrit

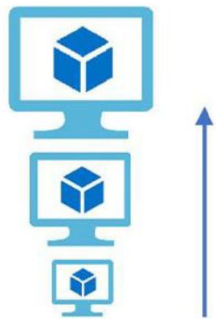
Scaling

Scaling

- Infrastructure scaling
- Scaling through partitioning

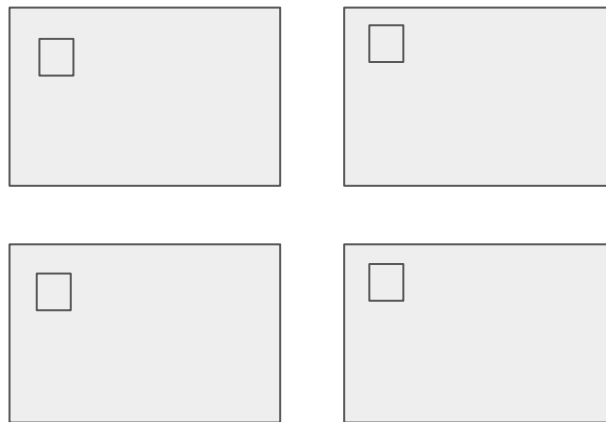
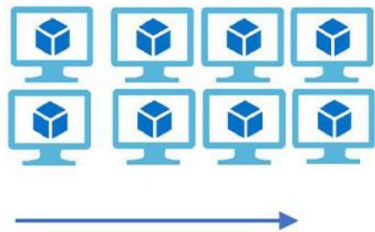
Vertical Scaling

(Increase size of instance (RAM , CPU etc.))



Horizontal Scaling

(Add more instances)

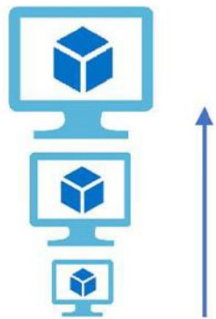


Scaling

- Infrastructure scaling
- Scaling through partitioning

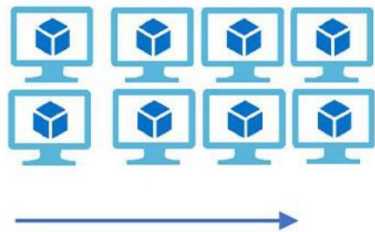
Vertical Scaling

(Increase size of instance (RAM , CPU etc.))



Horizontal Scaling

(Add more instances)



RabbitMQ – scaling

- RabbitMQ can expand its message-handling capacity
 - Vertically - allocate more compute resources to RabbitMQ's server
 - Horizontally

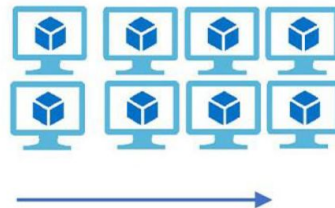
Vertical Scaling

(Increase size of instance (RAM , CPU etc.))



Horizontal Scaling

(Add more instances)



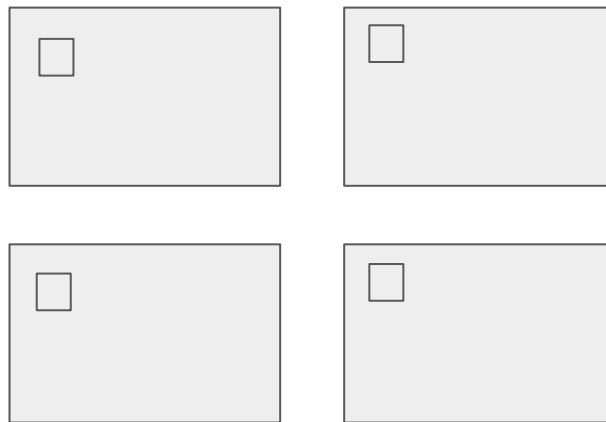
- Technique called RabbitMQ consistent hash exchange

<https://aws.amazon.com/compare/the-difference-between-rabbitmq-and-kafka/>

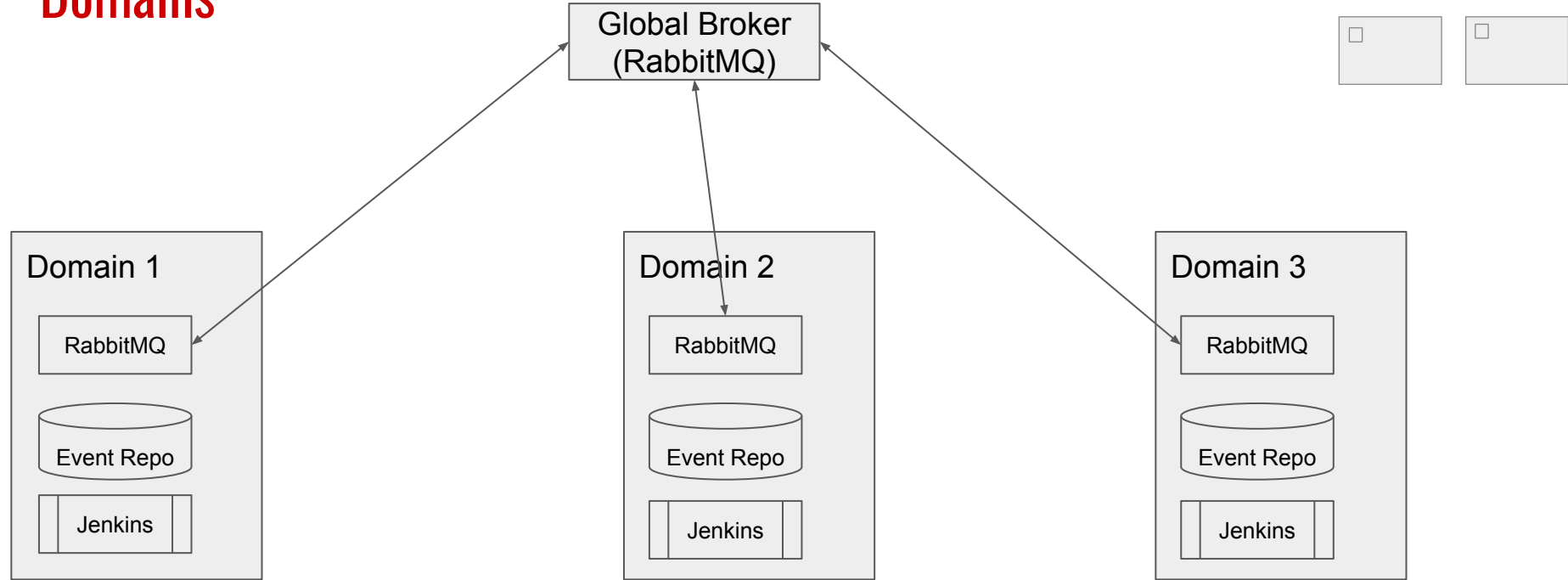
<https://www.alibabacloud.com/tech-news/a/rabbitmq/qu0eyrdxq4-scaling-rabbitmq-with-consistent-hashing>

Scaling

- Infrastructure scaling
- Scaling through partitioning

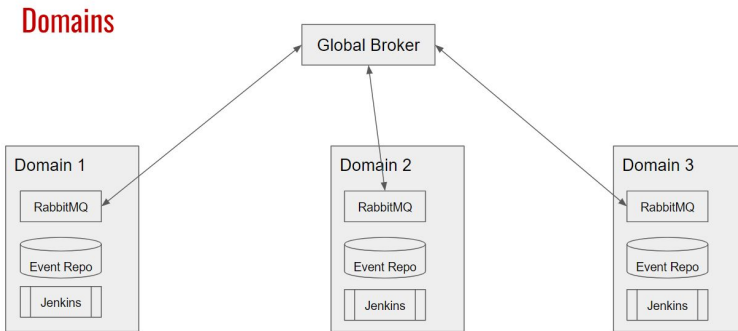


Domains



Domains – partitioning

- Group components
 - Upgrade components independently
- Can span several sites
- Maps to common sense
 - Like a group of people working with a product
 - All the events connected to a “product”
- A domain decides
 - whom to share events with
 - Can restrict what events will be federated
 - Who will access broker and database

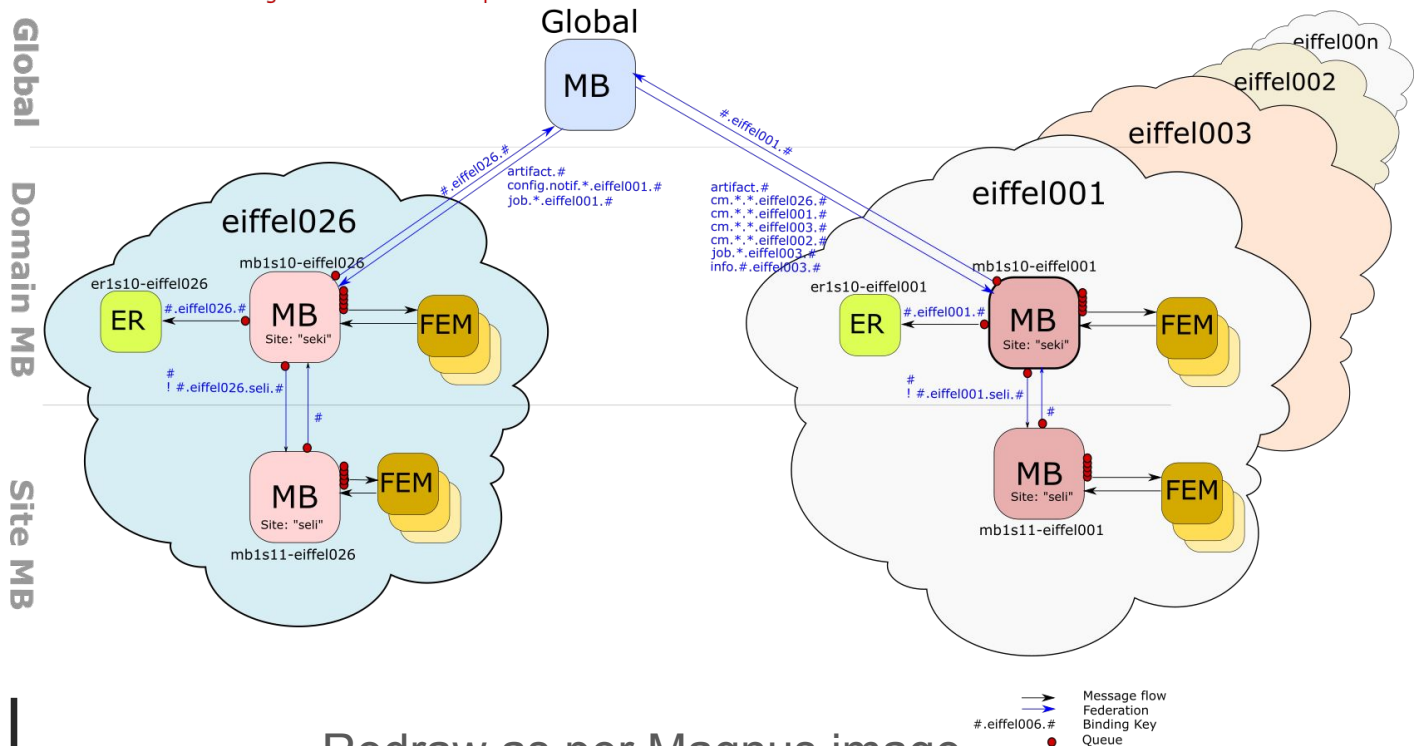


Domain setup – example

Check if we can use this

Drawing is for illustration only!

Hostnames and config is not the real setup



RabbitMQ – federation

- Federation plugin - transmit messages between brokers without requiring clustering
 - they may be hosted in different data centers, potentially on different continents
 - they may have different users, virtual hosts, permissions and purpose
 - they may run on different versions of RabbitMQ and Erlang
 - they may be of different sizes
- Types of federations
 - Exchange federation: for replicating a flow of messages through an exchange to a remote cluster
 - Queue federation: to create a "logical queue" across N clusters that will move messages where consumers are (if there are no local consumers)

Questions?

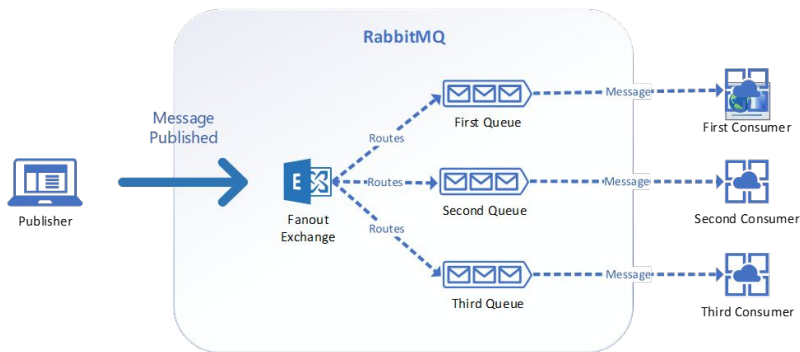
Reflections on Ericsson's setup

Domains – Where are we today in Ericsson?

- Designed years ago
 - Works well with about 4 million events/day
 - Probably not the setup if we done it today
- Original 3 domains
- Today 100+ domains (i.e. 100+ brokers)
- Need of governance
 - If you don't give a direction the engineers will create one
- Issues with syncing upgrades - then did not have 0 downtime/seamless upgrades
 - Therefore each user had their own domain

Domains – Where are we today

- A lot of users on old RabbitMQ with some not consuming lead to problems
 - Connection with Fanout using old fashion queues



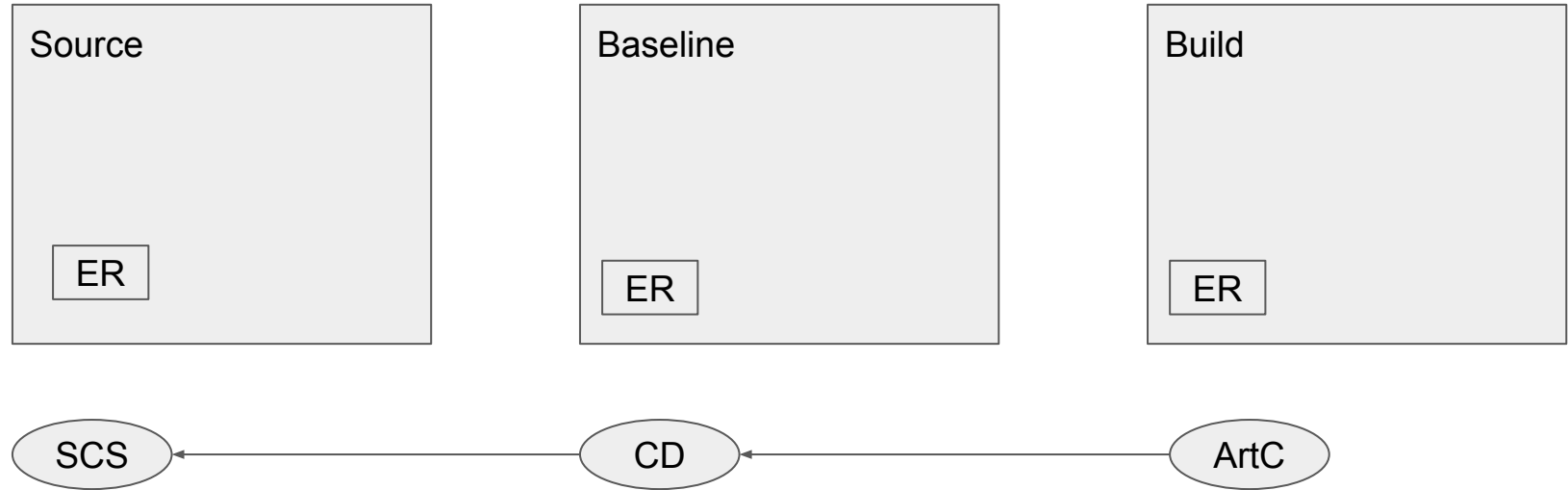
Domains – good and tricky

- Logical grouping valuable for understanding, reasoning, and filtering
 - E.g. separate staging and production environments

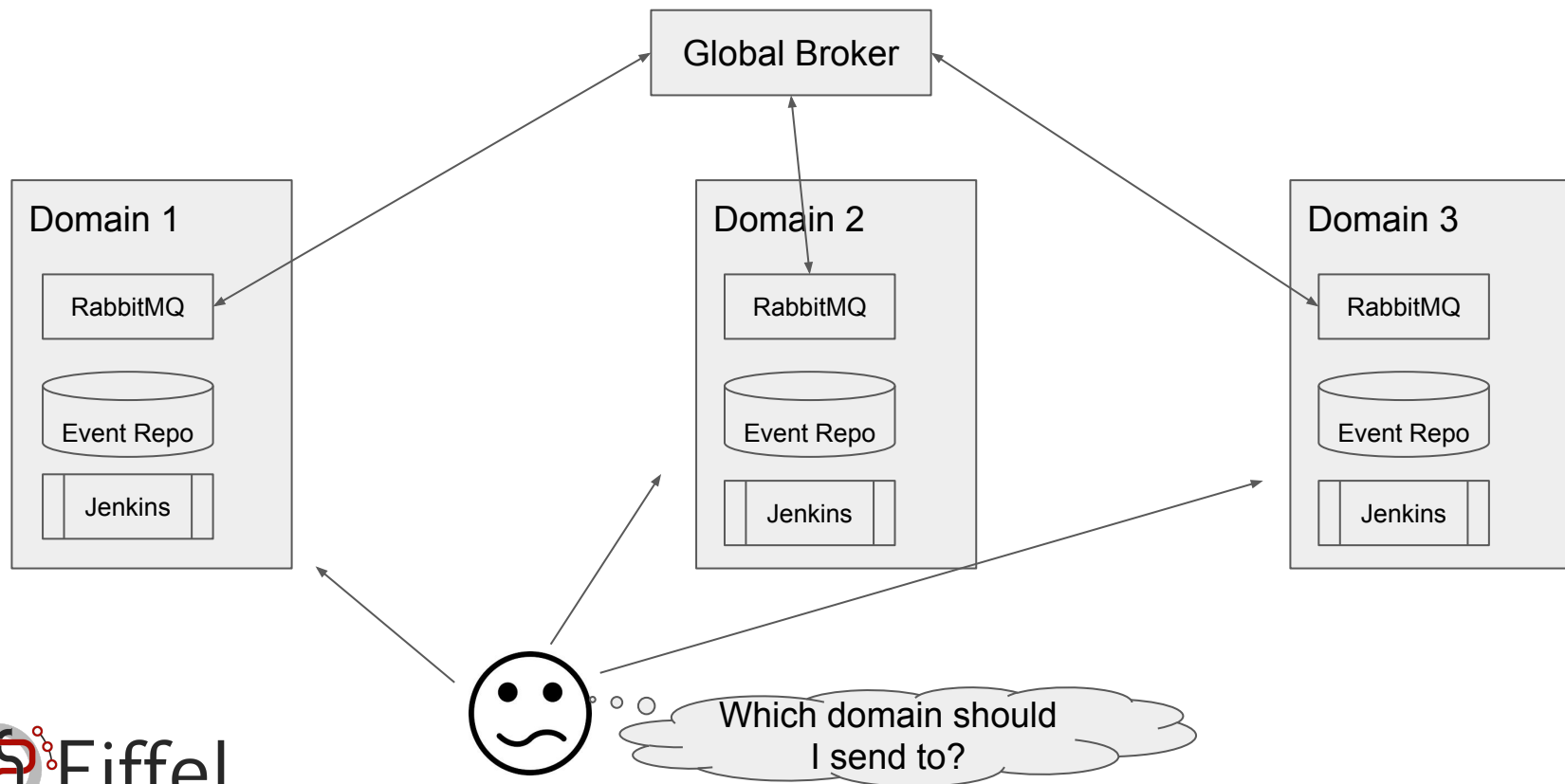
Domains – good and tricky

- Logical grouping valuable for understanding, reasoning, and filtering
 - E.g. separate staging and production environments
- Physical grouping
 - Tricky

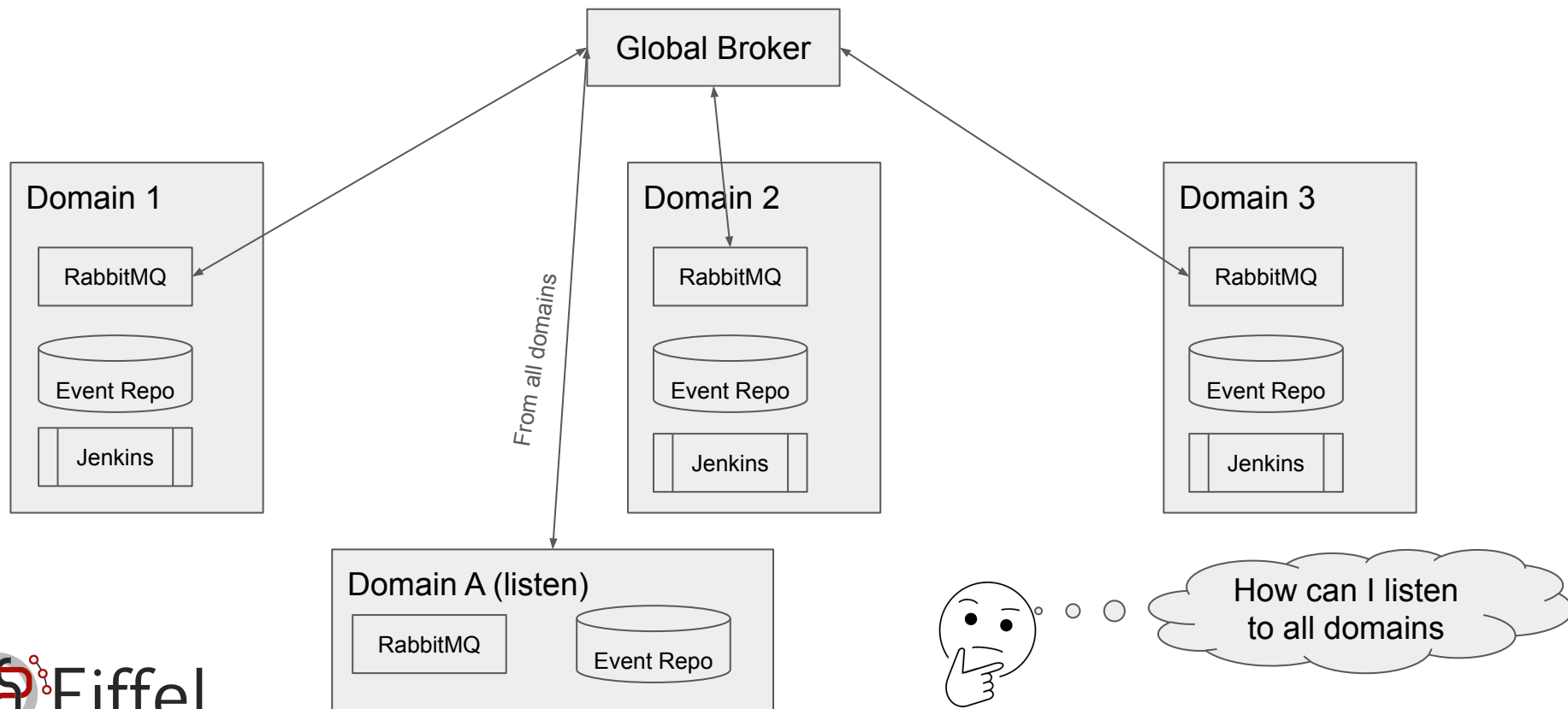
Traverse several databases



Where to send



Listening domain



Domains – good and tricky

- Logical grouping valuable for understanding, reasoning, and filtering
 - E.g. separate staging and production environments
- Physical grouping
 - Tricky
 - causes problems when traversing over several databases
 - Where should I send my event - may need access to send in many domains
 - Eg. when using a test framework used by many products need to send in their own domain
 - Sometime we need to create domains to listen in many domains
 - Many separate brokers
 - Ericsson: Upgrades responsible on many,
 - Infrastructure cost - many domains many brokers

Domains – good and tricky

- Logical grouping valuable for understanding, reasoning, and filtering
 - E.g. separate staging and production environments
- Physical grouping
 - Tricky
 - causes problems when traversing over several databases
 - Where should I send my event - may need access to send in many domains
 - Eg. when using a test framework used by many products need to send in their own domain
 - Sometime we need to create domains to listen in many domains
 - Many separate brokers
 - Ericsson: Upgrades responsible on many,
 - Infrastructure cost - many domains many brokers
 - Good
 - Makes possible to create secure zones if needed
 - Can from RabbitMQ 3.9 restrict on topic

Domains – good and tricky

- Logical grouping valuable for understanding, reasoning, and filtering
 - E.g. separate staging and production environments
- Physical grouping
 - Tricky
 - causes problems when traversing over several databases
 - Where should I send my event - may need access to send in many domains
 - Eg. when using a test framework used by many products need to send in their own domain
 - Sometime we need to create domains to listen in many domains
 - Many separate brokers
 - Ericsson: Upgrades responsible on many,
 - Infrastructure cost - many domains many brokers
 - Good
 - Makes possible to create secure zones if needed
 - Can from RabbitMQ 3.9 restrict on topic

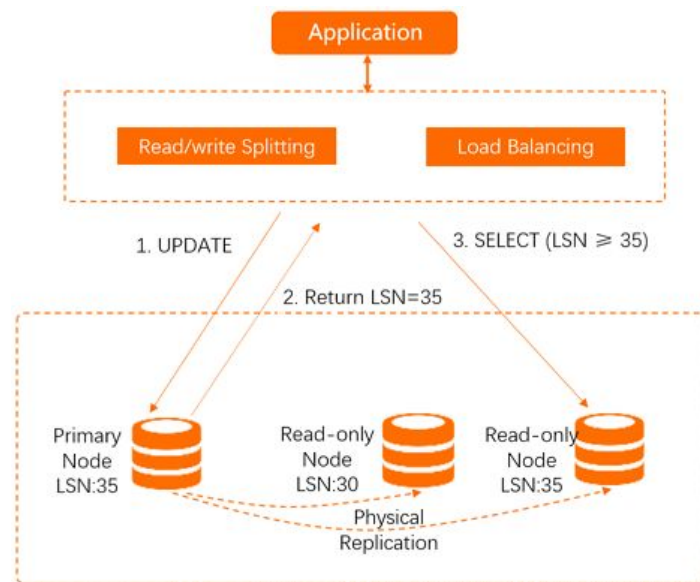
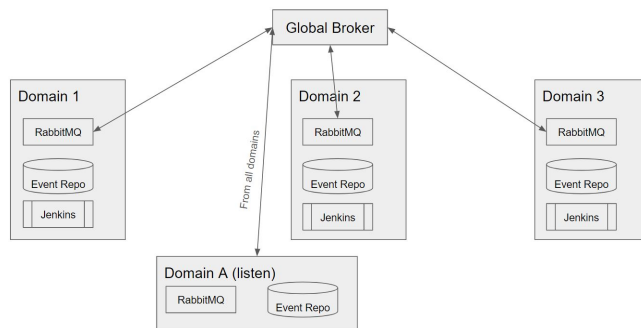
Don't introduce domain just because of organization, but when you really need it e.g. because of security

Security to motivate separation

- Event Content Tampering
 - Think zero trust
 - Sign event to guard against tampering instead of relying on broker authentication
- Sensitive information in events
 - Don't put sensitive information in the events link to it instead
- Read access
 - Hidden? -> Domain for it

Separate databases (re-hash)

- Security applications could motivate it
 - Read access
- Hard to find chains over several databases
 - Need aggregating event from many domains
- Cost?
- Performance reading?



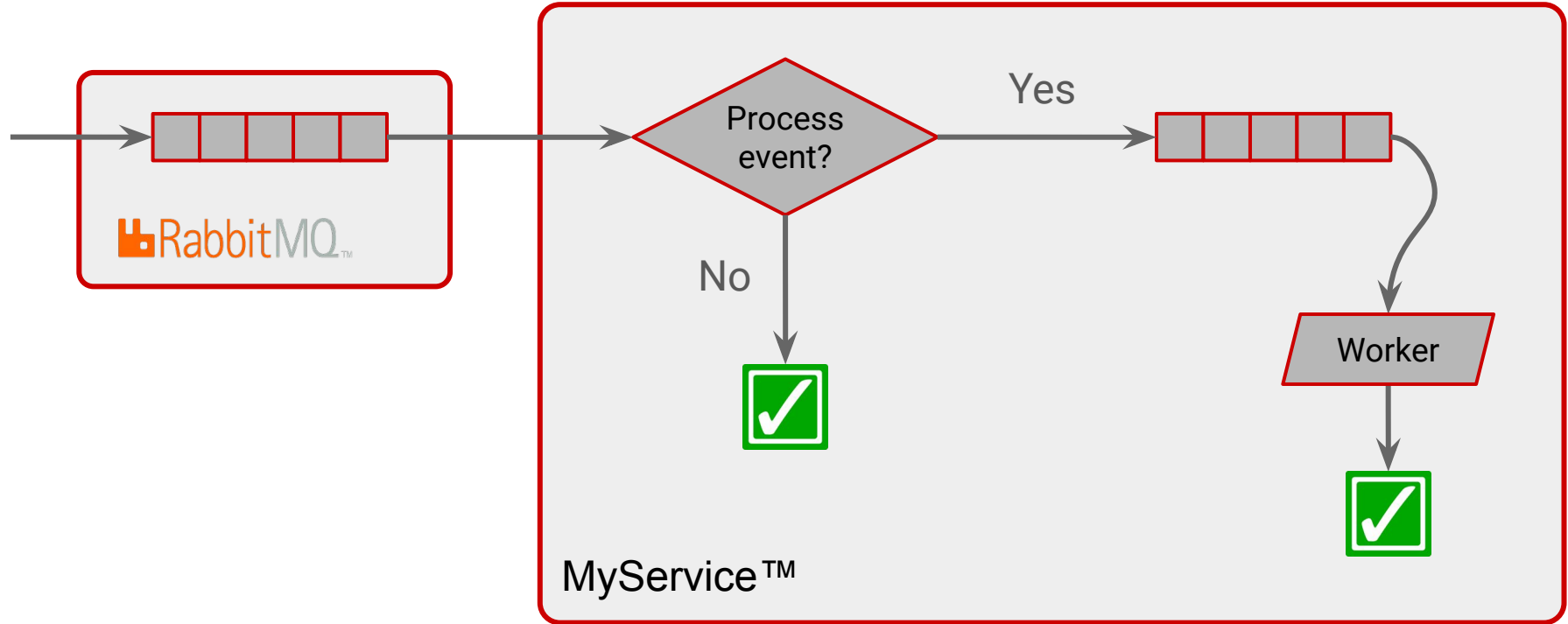
Takeaways

- Create a team to set the foundations, think platform
 - Have some central governance not to diverge too much
- Put the user first, many instances will make it harder for them
- Make sure to use the right queues
- Get the data of how much you actually need
- Enable seamless upgrades

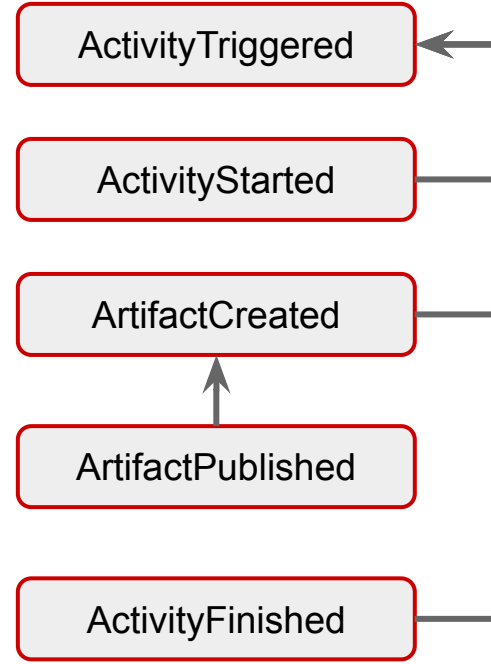
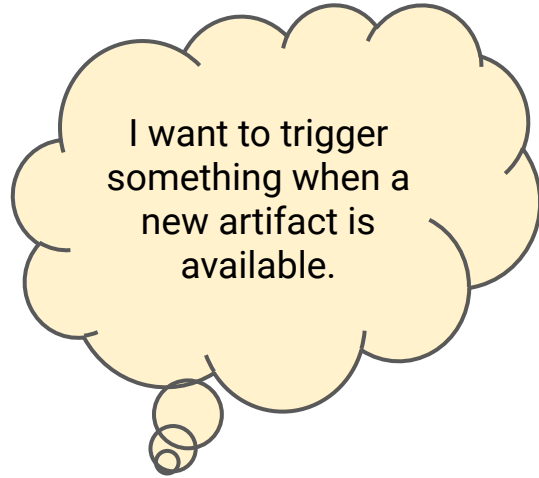
Questions?

Useful patterns when processing events

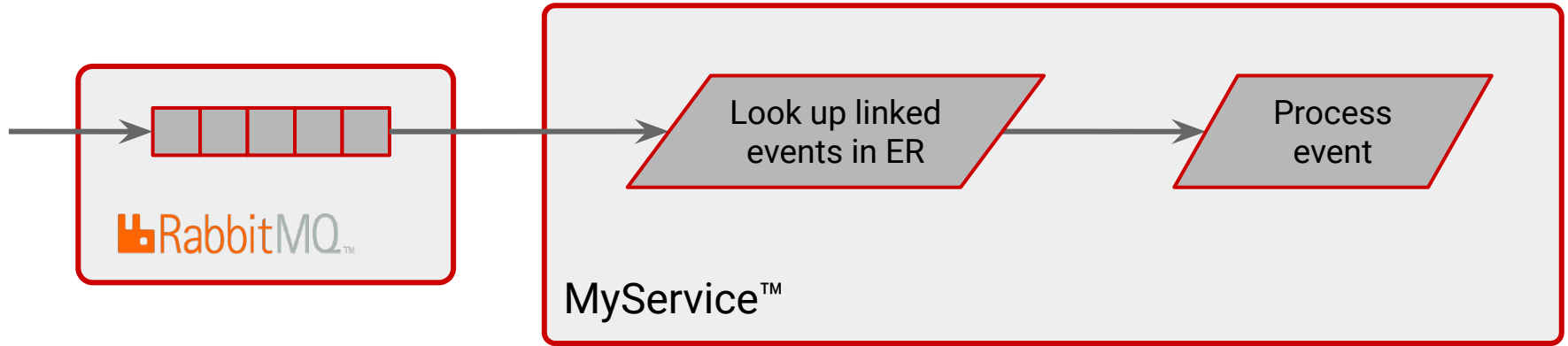
Use a queue inside your service



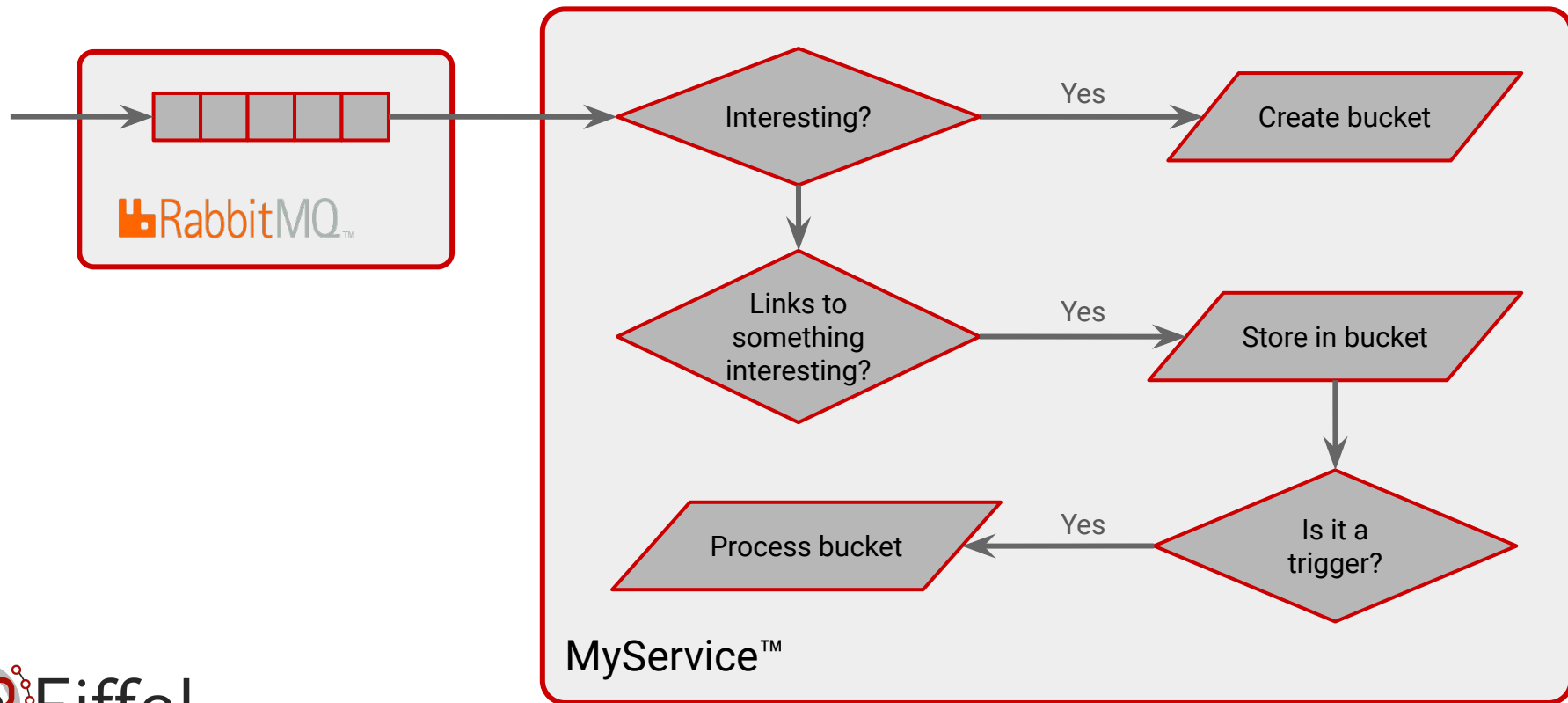
Avoiding event repository dependencies in consumers: Premise



Avoiding event repository dependencies in consumers: Not great

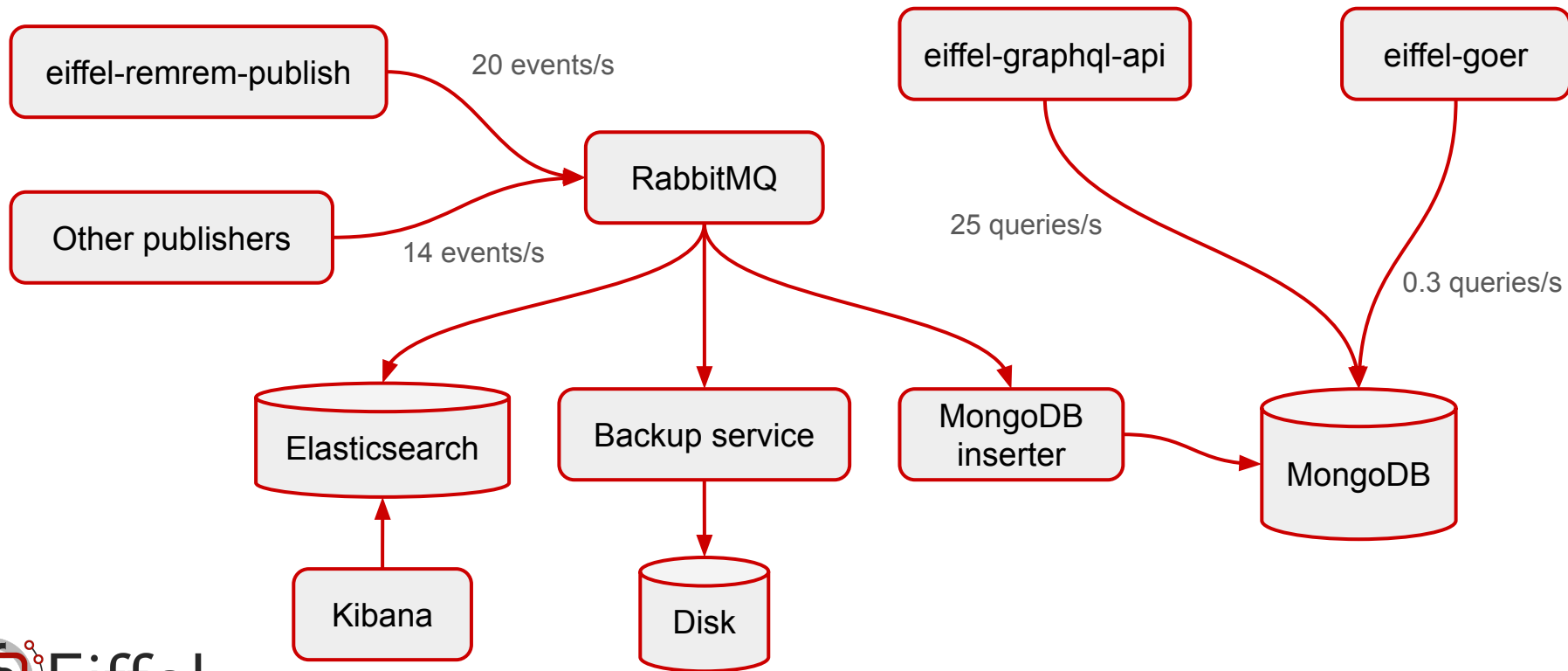


Avoiding event repository dependencies in consumers: Better



Reflections on Axis's setup

Axis's Eiffel infrastructure



Questions?