VMware Tanzu RabbitMQ

Reliability and Patterns for Edge Deployments

Dan Buchko, VMware Tanzu Solutions Engineer

July 23, 2020

Reliability

Dealing With Failure

Broker, Clients and Infrastructure

Broker

Will a message survive

- partial broker failure?
- complete broker failure?

Client

Has a message been

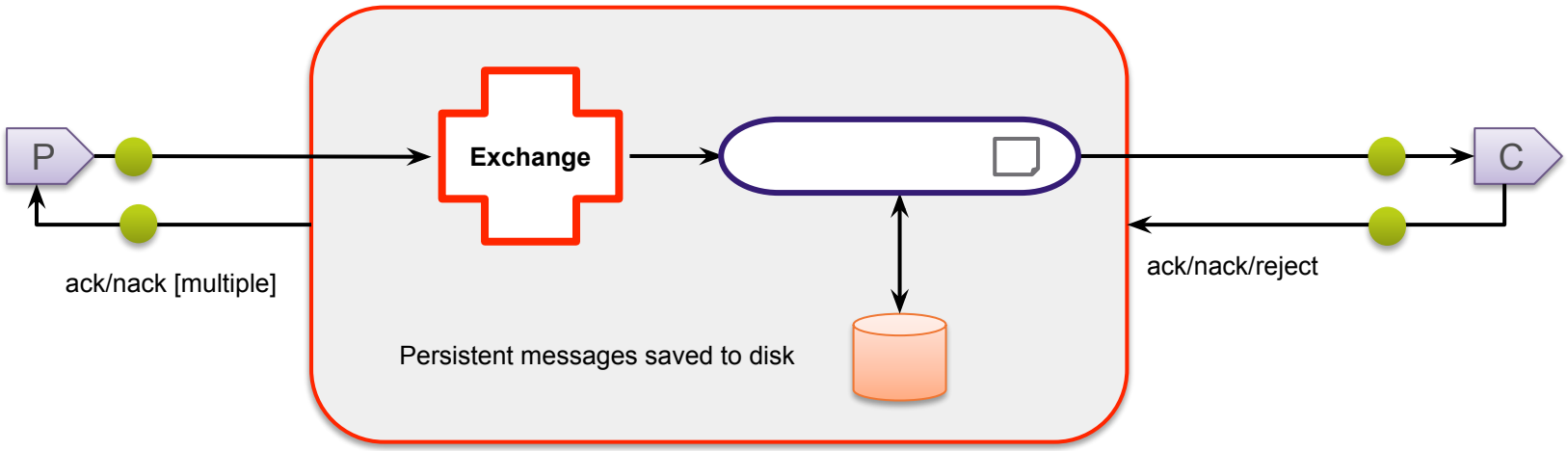
- published correctly?
- consumed?

Infrastructure

What happens if the network fails?

Message Flow

Points Of Failure



- Network buffers

Lots of places for things to break!

AMQP Reliability

AMQP offers several mechanisms to provide guarantees to producers and consumers

- basic.ack
  - Acknowledges publish/consume of one or more messages
- basic.reject
  - AMQP means to reject a single message
- basic.return
  - Returns a message to the sender with a reply code and text
  - Issued by the broker to publishers when mandatory=true
- transactions
  - Wrap around basic.ack/reject/return
  - Only atomic across a single queue
  - Use RabbitMQ Publisher Confirms instead

AMQP Reliability - Longevity

AMQP has two concepts for longevity

- Durable
  - Applies to **exchanges** and **queues**
  - Means the ***definitions, not the contents***
- Persistent
  - Applies to **messages** published to durable queues
  - Messages are saved to disk

Reliable Producer and Consumer Design

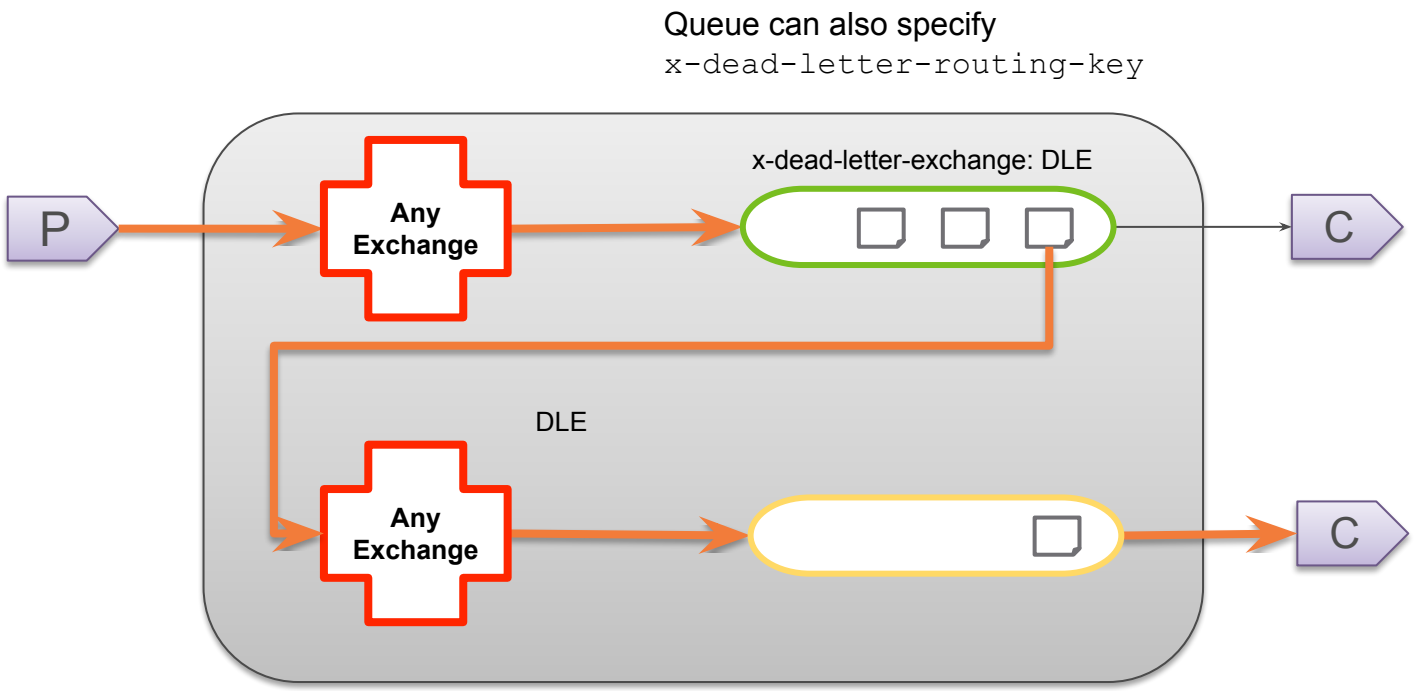
- Allow for failure - Idempotent design
  - Consumer acknowledgements
    - “Best effort” pattern
    - Consumer reads message, processes it, and acknowledges it
    - RabbitMQ only deletes the message from the queue after it has received the acknowledgement
  - Consumers can check for ‘redelivered = true’
    - Client **may** have seen the message before
    - Consumers **must** be idempotent and handle duplicate messages in case of redelivery
  - Producers use confirms
    - Analagous to consumer acknowledgements, but for the producer
    - Retransmit any non-confirmed messages on recovery from a channel or connection failure
    - Use ‘mandatory’ flag on basic.publish to receive a basic.return if the message could not be queued

RabbitMQ Extensions - Message Lifecycle

- Per-message TTL
  - Per-message when published
- Per-queue message TTL
  - Declare a queue with x-message-ttl or by setting message-ttl policy
  - TTL sets maximum time a message may be in the queue
  - Shortest (message vs queue) is applied
- Queue TTL
  - Queue will be deleted if unused for the TTL
  - Declare at queue creation or by policy
- Queue Length
  - Declare queue with x-max-length or by policy
  - Messages dropped/dead-lettered from front of queue once length exceeded
- Dead-letter Exchanges
  - Re-route messages when discarded or expired

‘Dead-Letter’ Exchanges

Any Exchange Can Be A Dead-Letter Exchange



Messages are dead-lettered when:

- basic.reject or basic.nack
  [requeue=false]
- TTL expired
- Queue length exceeded

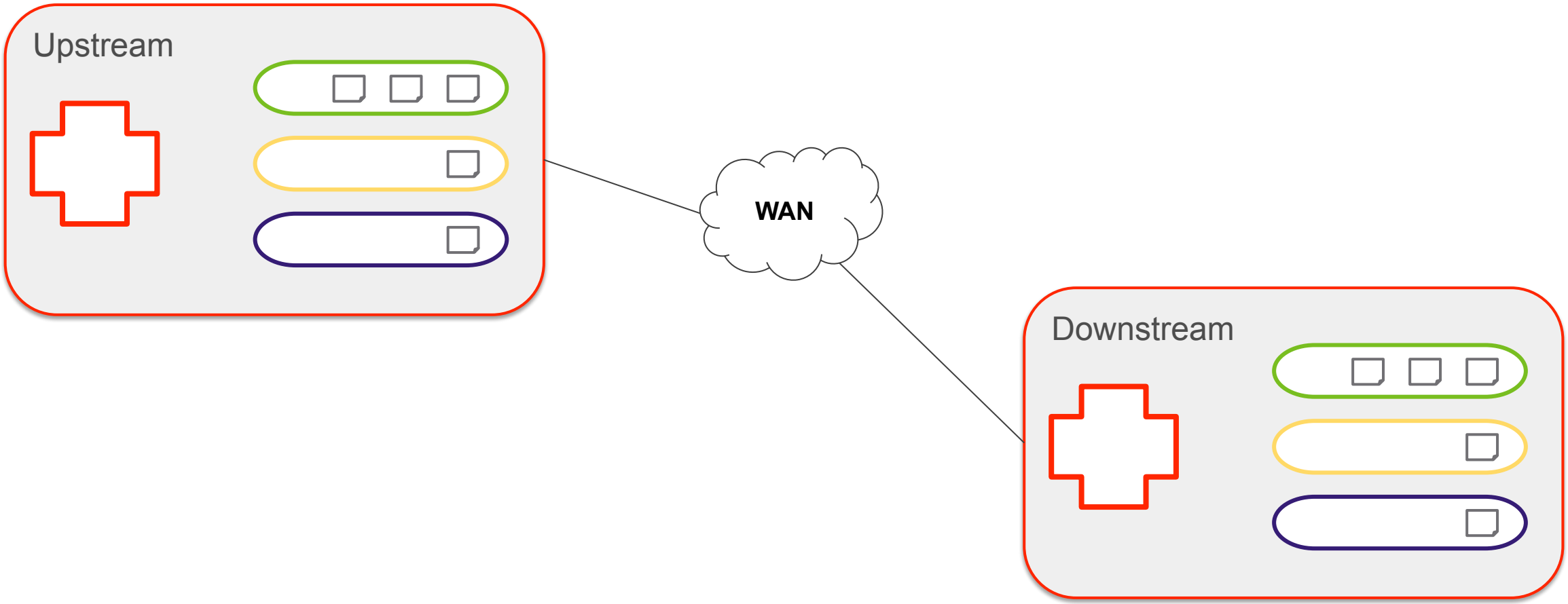
x-death header

```
reason:
queue:
exchange:
time:
routing-keys:
```

Edge Deployments

Connecting Brokers

Terminology



The Shovel

A Tool For Moving Messages

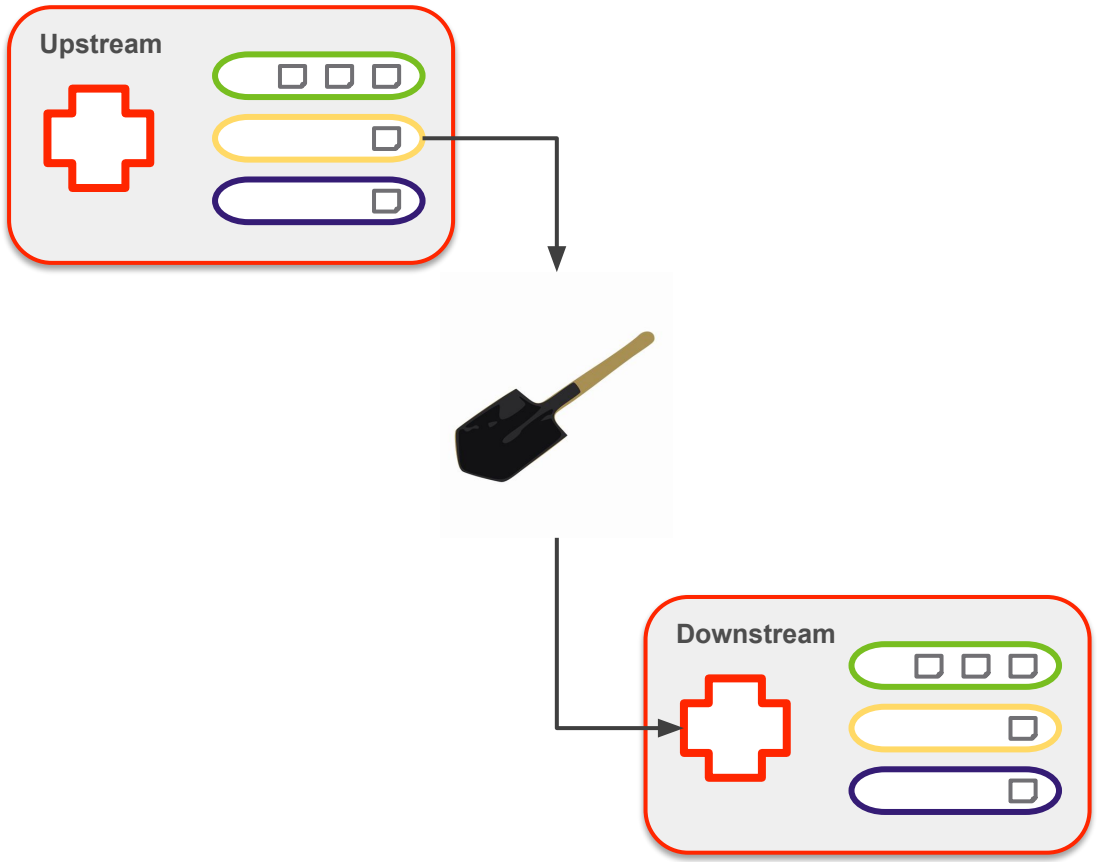
Consumes messages from a remote queue and republishes them to a local exchange

A simple way to move messages over a WAN

An Erlang AMQP application

- Can declare exchanges and queues on both the source and destination
- Good control over acks, prefetch etc.
- Essentially a well-written client

Can run anywhere - doesn't have to be on either broker



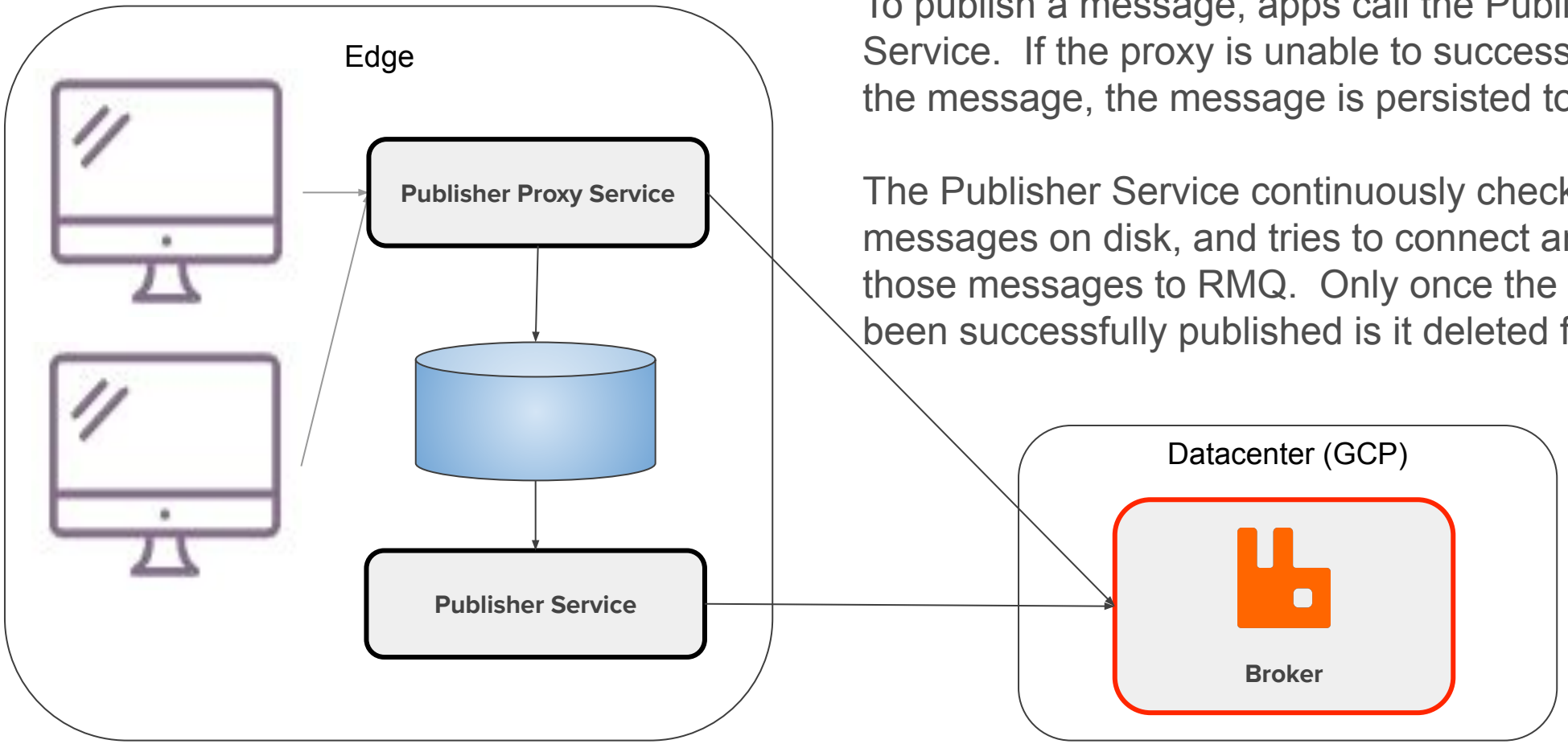
Edge Deployments

A Couple of Options...

- Create a couple of microservices to persist messages on the client
- Leverage RabbitMQ plugins (such as Shovel or Federation) to replicate messages over the WAN

Edge Deployment

Option #1: Custom Publisher Proxy Service

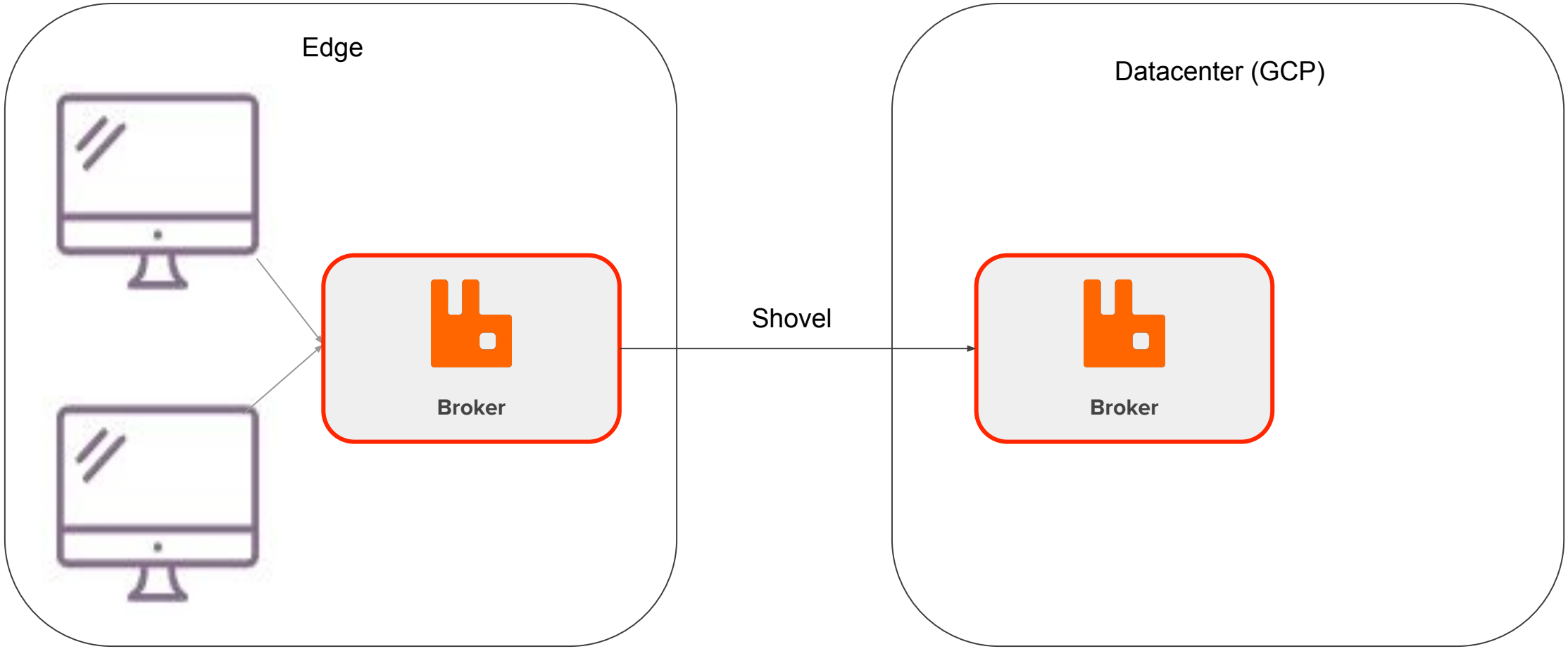


To publish a message, apps call the Publisher Proxy Service. If the proxy is unable to successfully publish the message, the message is persisted to local disk.

The Publisher Service continuously checks for messages on disk, and tries to connect and publish those messages to RMQ. Only once the message has been successfully published is it deleted from disk.

Edge Deployment

Option #2: Using RabbitMQ Shovel Plugin



Thank You!