INFRASTRUCTURE

AGENDA

- On Demand
- Solution – Architecture - Problem Solving
- Technologies
- Benchmark

On Demand

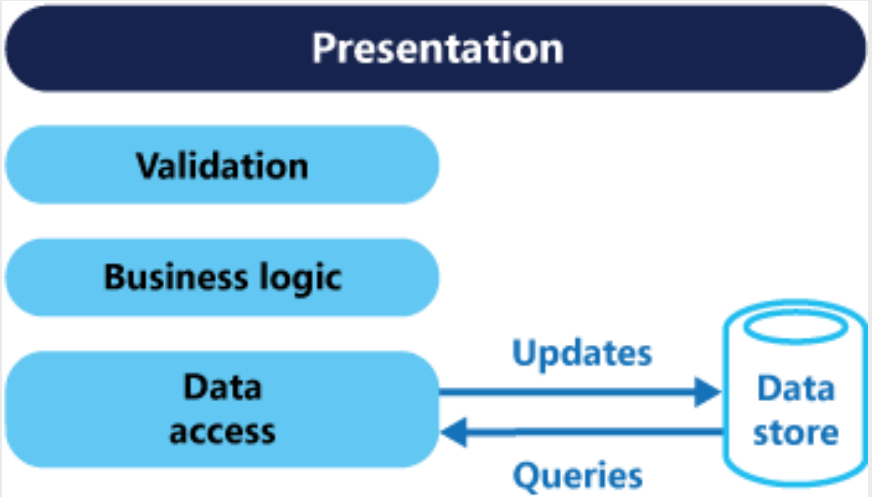
- Real-time
- High Performance
- Concurrency
- Clustering
- Load Balancing
- Robust
- Scalability
- Security
- Maintenance

Solution – Architecture - Problem Solving

- Command and Query Responsibility Segregation (CQRS) Pattern
- Advanced Message Queuing Protocol (AMQP) Pattern
  - Use RabbitMQ

Command and Query Responsibility Segregation(CQRS)

- Traditional CRUD architecture

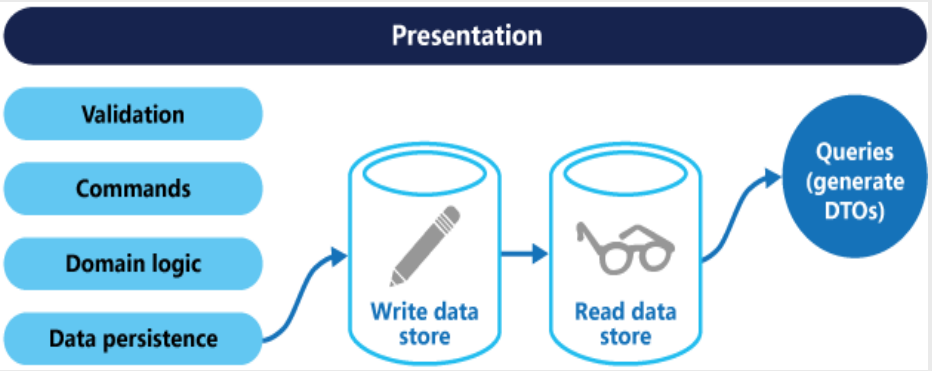


Has some disadvantages

- Limited business logic
- Mismatch between the read and write representations of data
- It risks encountering data contention in a collaborative domain
- Both commands and queries are executed in single data repository
- Update conflicts caused by concurrent updates when optimistic locking is used
- Bad performance with big traffic on request and response

Command and Query Responsibility Segregation(CQRS)

- CQRS architecture

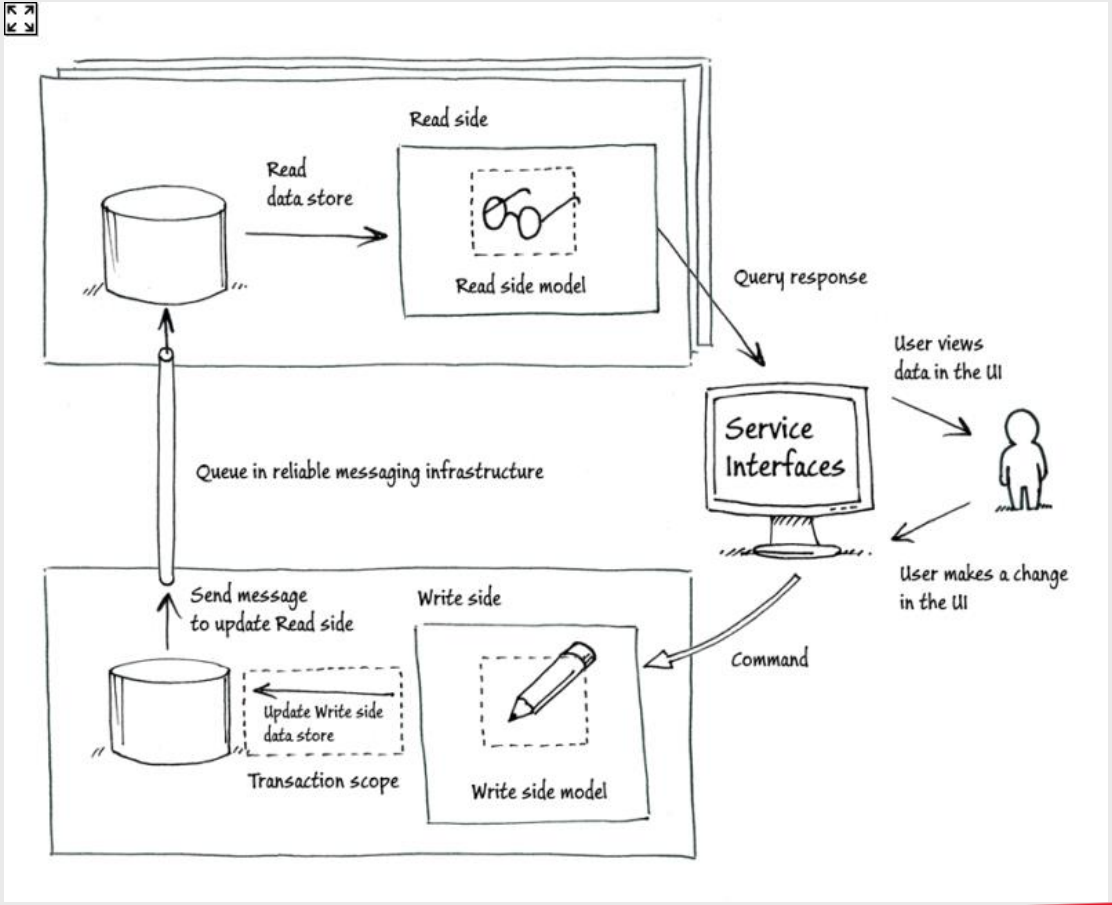


Advantages

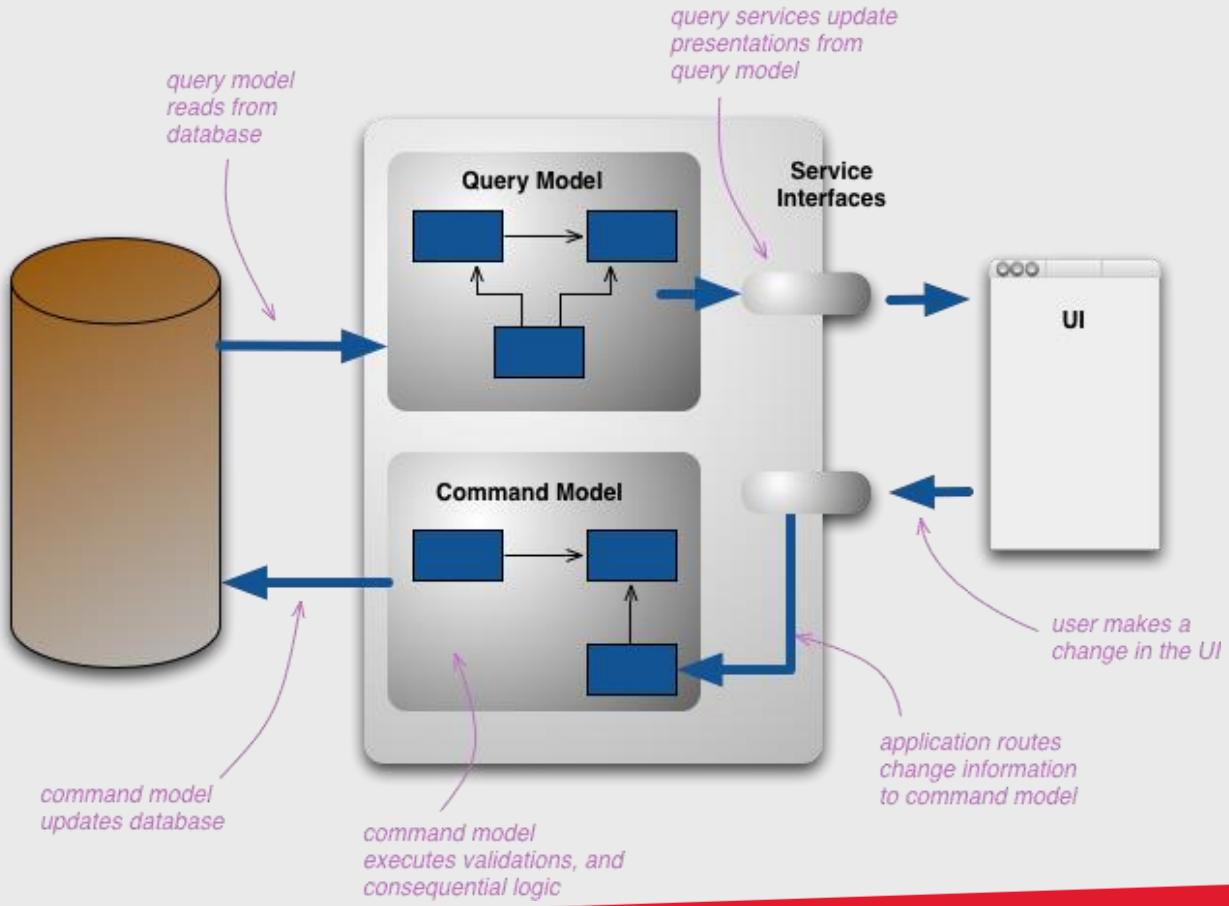
- Split out read and write operations
- Maximum performance read-only data storage
- Have very fast responses to queries for data
- Minimize resource utilization.
- Minimize latency
- Minimize costs
- No more mapping object to tables
- Suitable database for query operation and for write data storage
- Distributed systems capabilities

Command and Query Responsibility Segregation(CQRS)

- CQRS with Message



Command and Query Responsibility Segregation(CQRS)

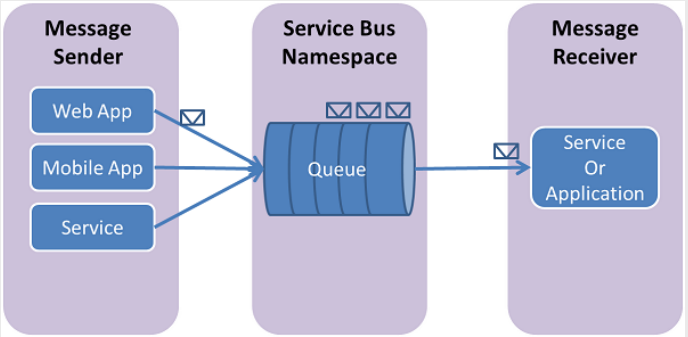


Advanced message Queuing Protocol (AMQP)

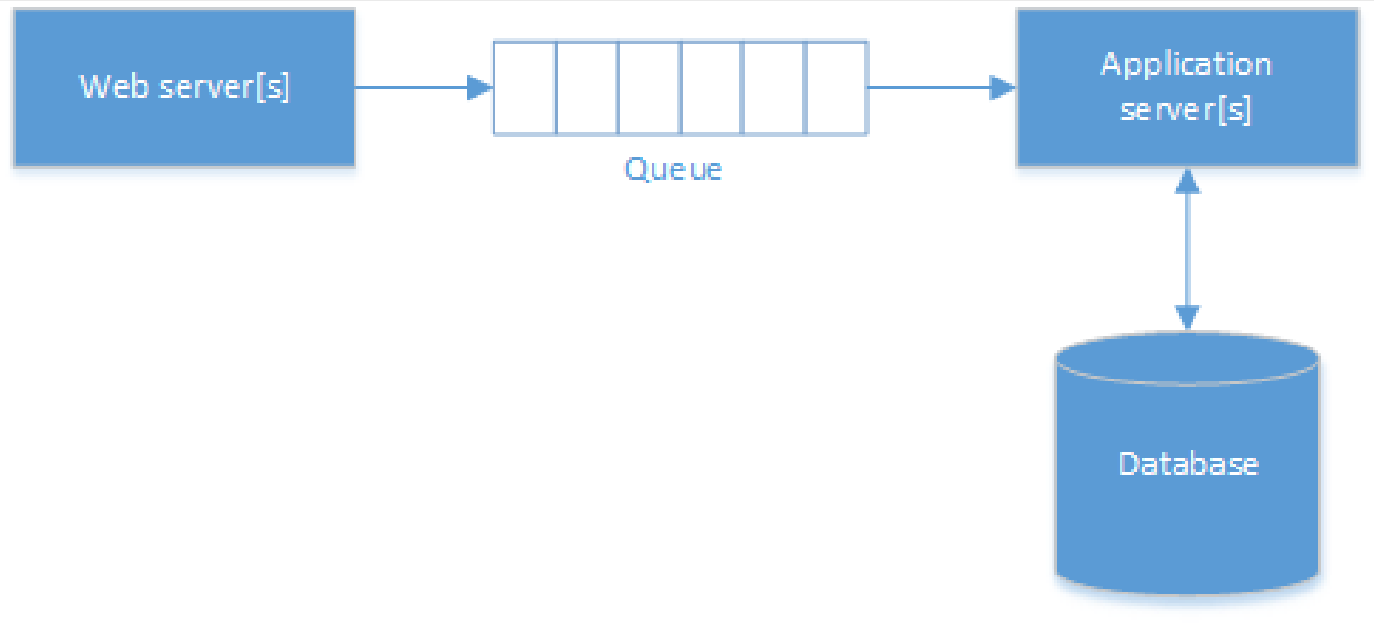
- Message-based middleware system
- Asynchrony
- Routing
- Scalability
- Resiliency
- Decoupling

Objectives

- Understand the need message queuing?
- Basic understanding about Advance Message Queuing Protocol (AMQP)
- Explore common message queue patterns
- Benchmark



Scenario 1

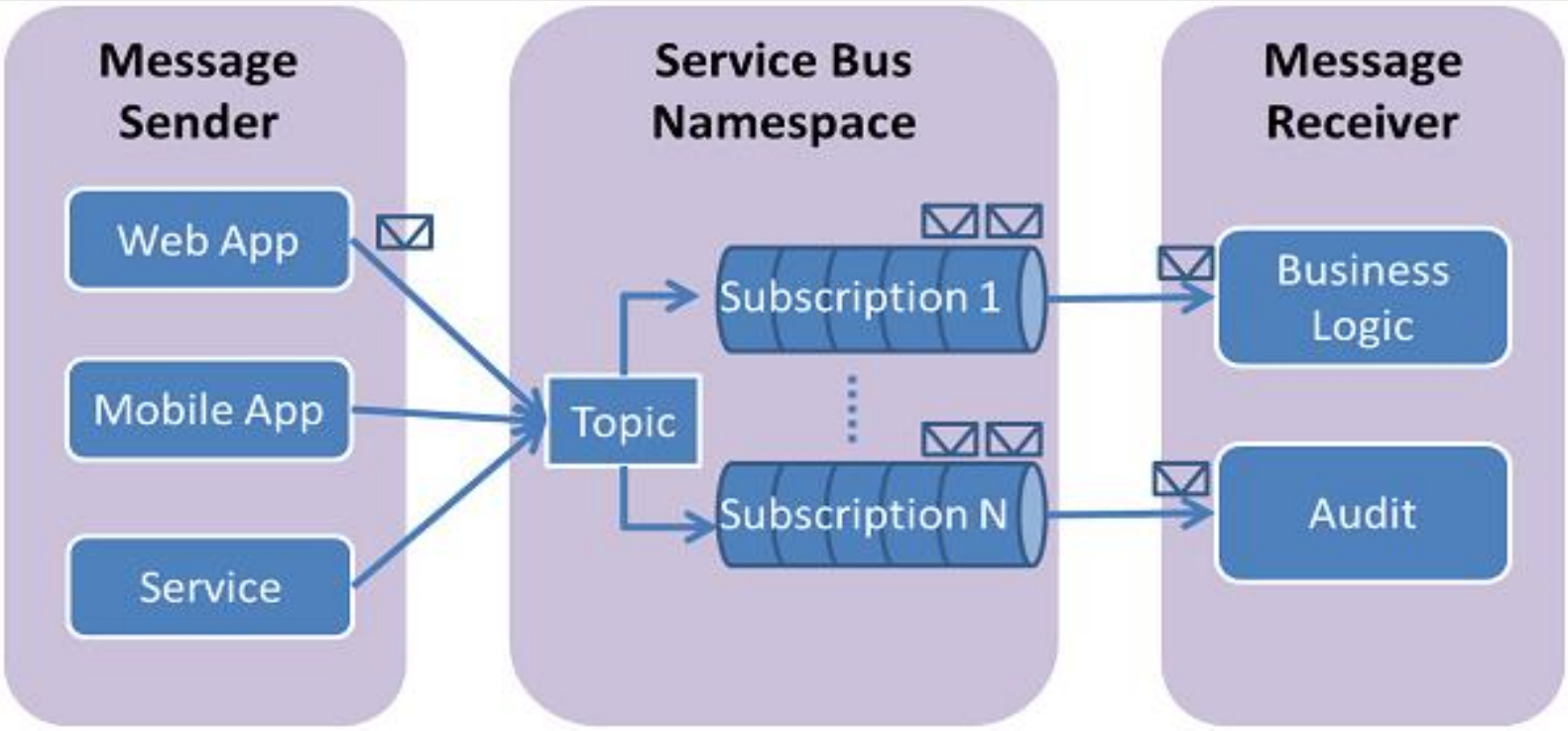


Decoupling heavy-weight processing from a live user request

Why not use database?

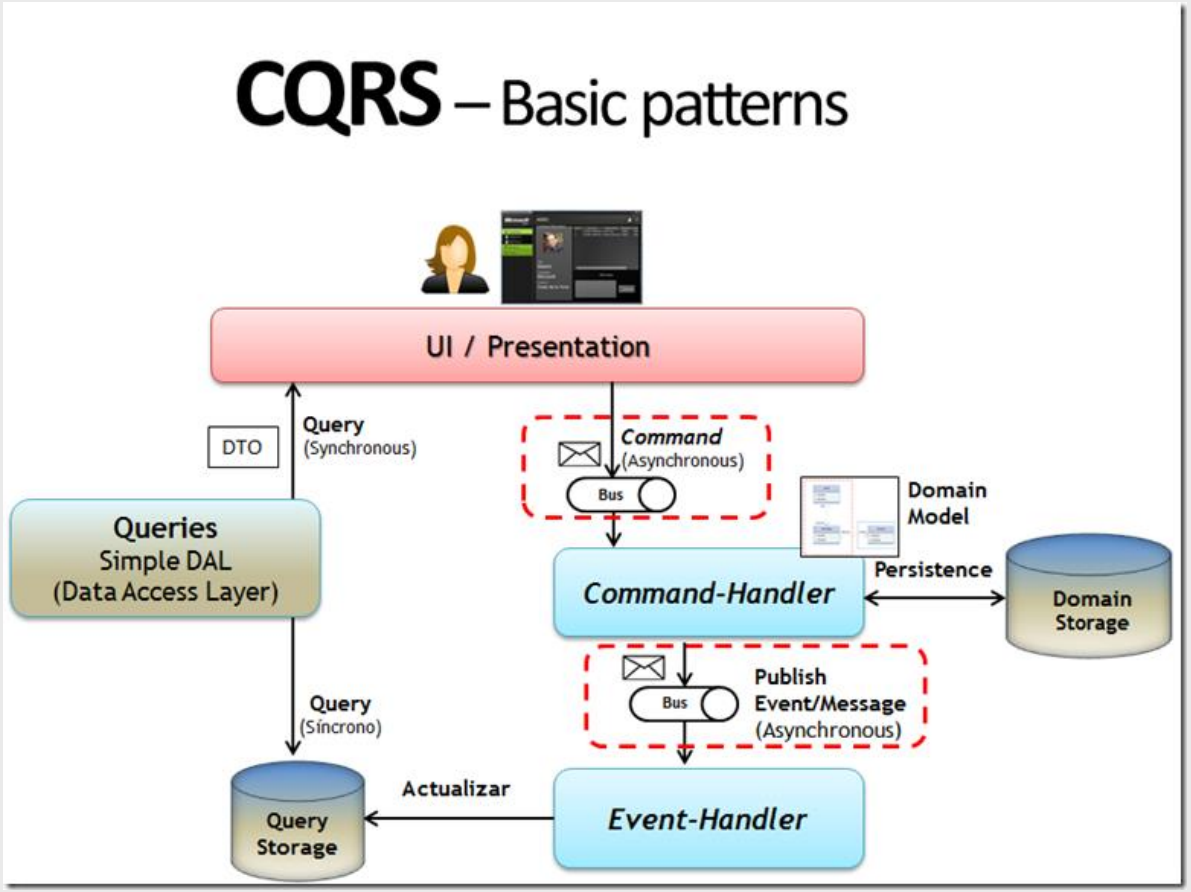
- Efficiency
  - Polling
  - Concurrency
- Manual clean up
- Scaling Won’t Be Easy

Scenario 2



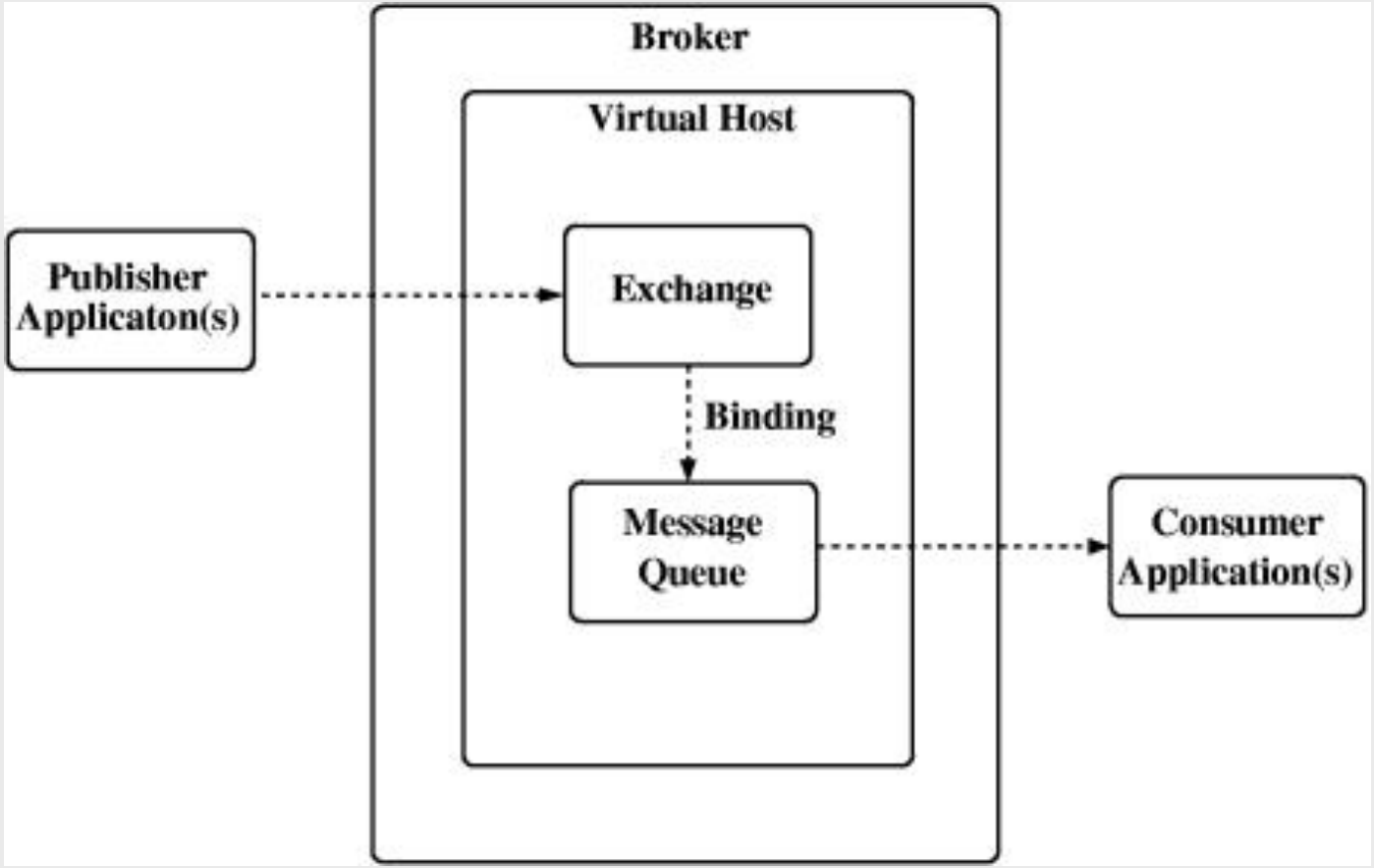
Decoupling senders and receivers

Scenario 2


CQRS – Basic patterns
UI / Presentation
Query
(Synchronous)
DTO
Command
(Asynchronous)
Bus
Domain
Model
Queries
Simple DAL
(Data Access Layer)
Command-Handler
Persistence
Domain
Storage
Publish
Event/Message
(Asynchronous)
Bus
Query
(Síncrono)
Actualizar
Query
Storage
Event-Handler

AMQP Broker implementations

AMQP Model


Broker
Virtual Host
Publisher Applicaton(s)
Exchange
Binding
Message Queue
Consumer Application(s)

Exchange

- A message routing agent: it will receives messages from publisher and route them to Queue
- Default exchange (without a name)

  routing messages to a queue (routing key = name queue)
- Direct exchange

  routing message to a queue based on routing key (not necessary queue name, routing key = bind key)
- Fanout exchange

  routing message to more queue (pub/sub) and not use a routing key
- Topic exchange

  routing message to a queue based on routing key like a topic (routing key match pattern)
- Header exchange

  routing message to queue based on header filters

Exchange

- Important attributes
  - Name
  - Durability: Durable exchanges last until they are deleted. Temporary exchanges last until the server shuts-down
  - Auto delete: exchanges last until they are no longer used
  - Arguments

Binding

- An exchange accepts messages from a producer application and routes these to message queues according to pre-arranged criteria.
- These criteria are called "bindings“

Queue

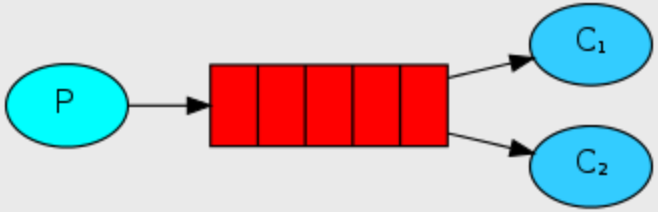
- **Name:**
- **Durable**: if set, the message queue remains present and active when the server restarts. It may lose transient messages if the server restarts.
- Exclusive: if set, the queue belongs to the current connection only, and is deleted when the connection closes.
- Auto delete: Auto-deleted message queues last until they are no longer used
- Arguments

Message

- Message is the atomic unit of processing
- A message consists of a content header, holding a set of properties, and a content body, holding an opaque block of binary data.
- Broker never modifies content body

Common messages queue patterns

Worker Queue

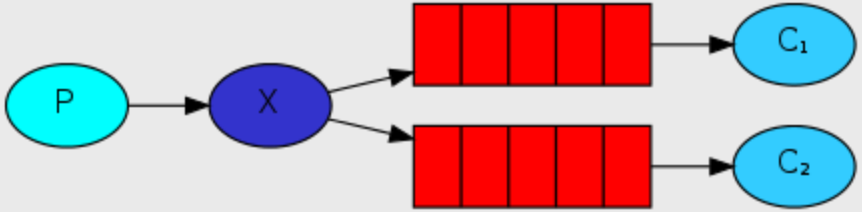


- Worker queues are used to distribute time-consuming task among multiple workers
- Especially useful in web application where it's impossible to handle complex or resource-intensive tasks
- Default exchange: direct exchange with no name (empty string) pre-declared by the broker and automatically bound to queue by queue name

Worker Queue

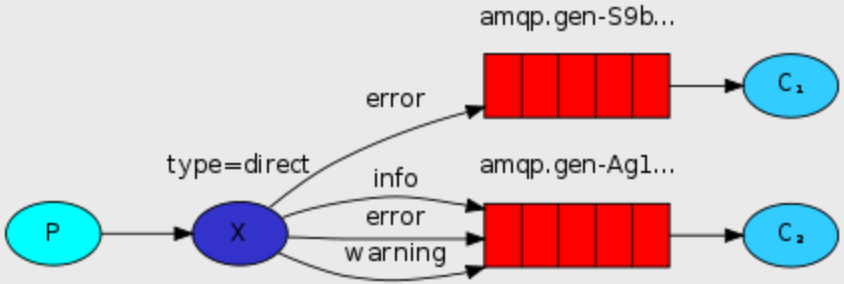
- Round-robin dispatching
- Message acknowledgement
- Message durability
- Fair dispatching

Publish/Subscribe



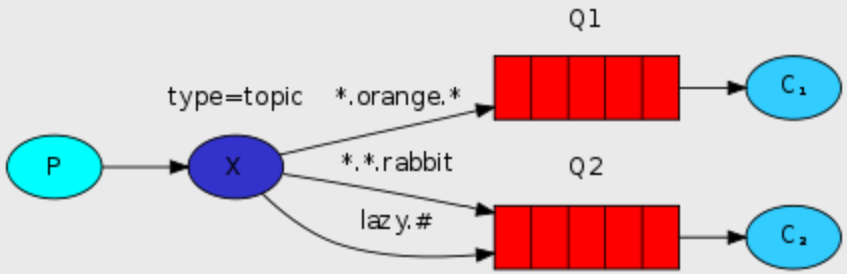
- Delivery messages to multiple consumers
- Fanout exchange: broadcasts all the messages it receives to all the queues it knows
- Binding: the relationship between exchange and queue

Routing



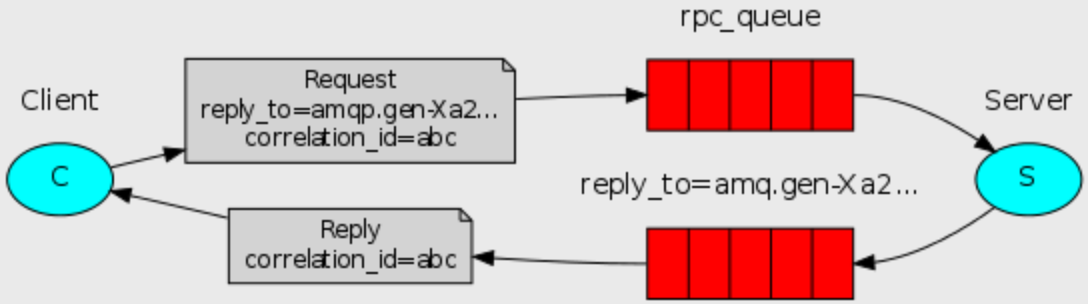
- Subscribe to only a subset of messages
- Direct exchange: a message goes to the queues whose binding key exactly matches the routing key of the message

Topic



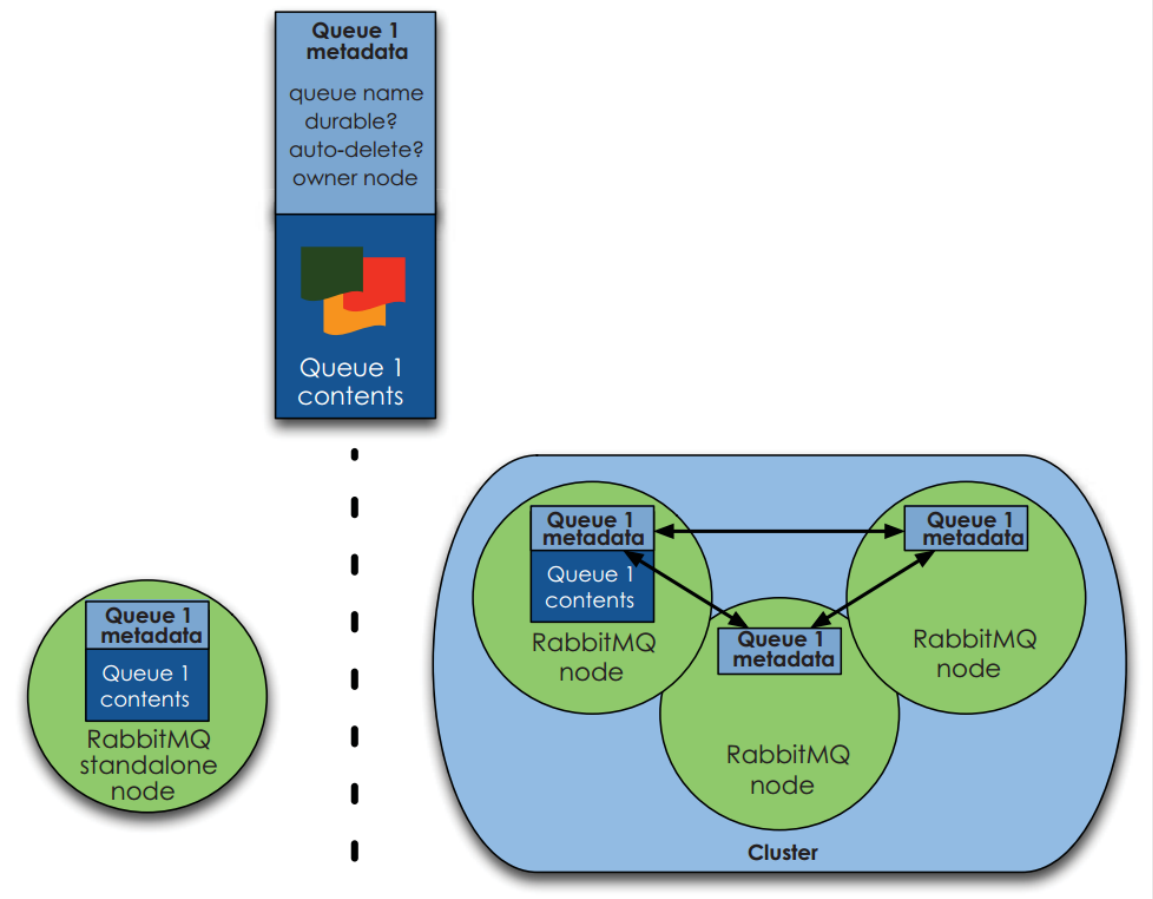
- Routing based on multiple criteria
- Topic exchange: matching binding key
- * (star) can substitute for exactly one word.
- # (hash) can substitute for zero or more words.

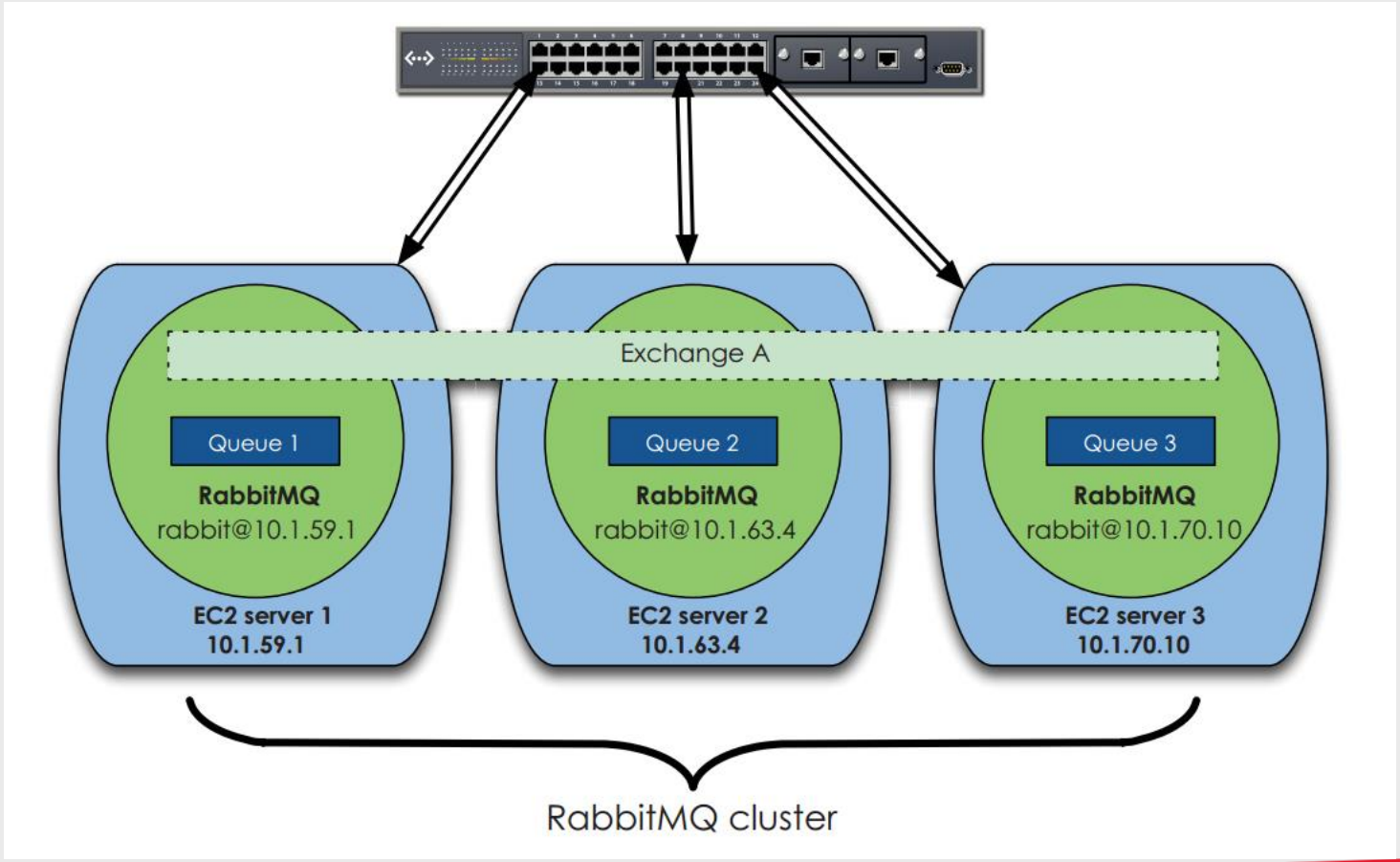
Request/Response or RPC



- Run a function on a remote computer and wait for the result

Performance and clustering




Exchange A
Queue 1
RabbitMQ
rabbit@10.1.59.1
EC2 server 1
10.1.59.1
Queue 2
RabbitMQ
rabbit@10.1.63.4
EC2 server 2
10.1.63.4
Queue 3
RabbitMQ
rabbit@10.1.70.10
EC2 server 3
10.1.70.10
RabbitMQ cluster

RabbitMQ

User: queue
RabbitMQ 3.2.3, Erlang R15B01
Log out

Overview | Connections | Channels | Exchanges | Queues | Admin

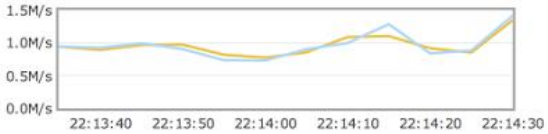
Overview

Totals

Queued messages (chart: last minute) (?)

Ready	2,343 msg
Unacknowledged	0 msg
Total	2,343 msg

Message rates (chart: last minute) (?)



Publish	1,345,531/s
Deliver (noack)	1,413,840/s

Global counts (?)

Connections: 12690 | Channels: 12690 | Exchanges: 194 | Queues: 186 | Consumers: 10304

Nodes

Name	File descriptors (?)	Socket descriptors (?)	Erlang processes	Memory	Disk space	Uptime	Type
rabbit@b-rabbitmq-queue-1te2	445 262144 available	395 235837 available	4594 1048576 available	252MB 12GB high watermark	6.2GB 48MB low watermark	1h 33m	RAM

THANK YOU