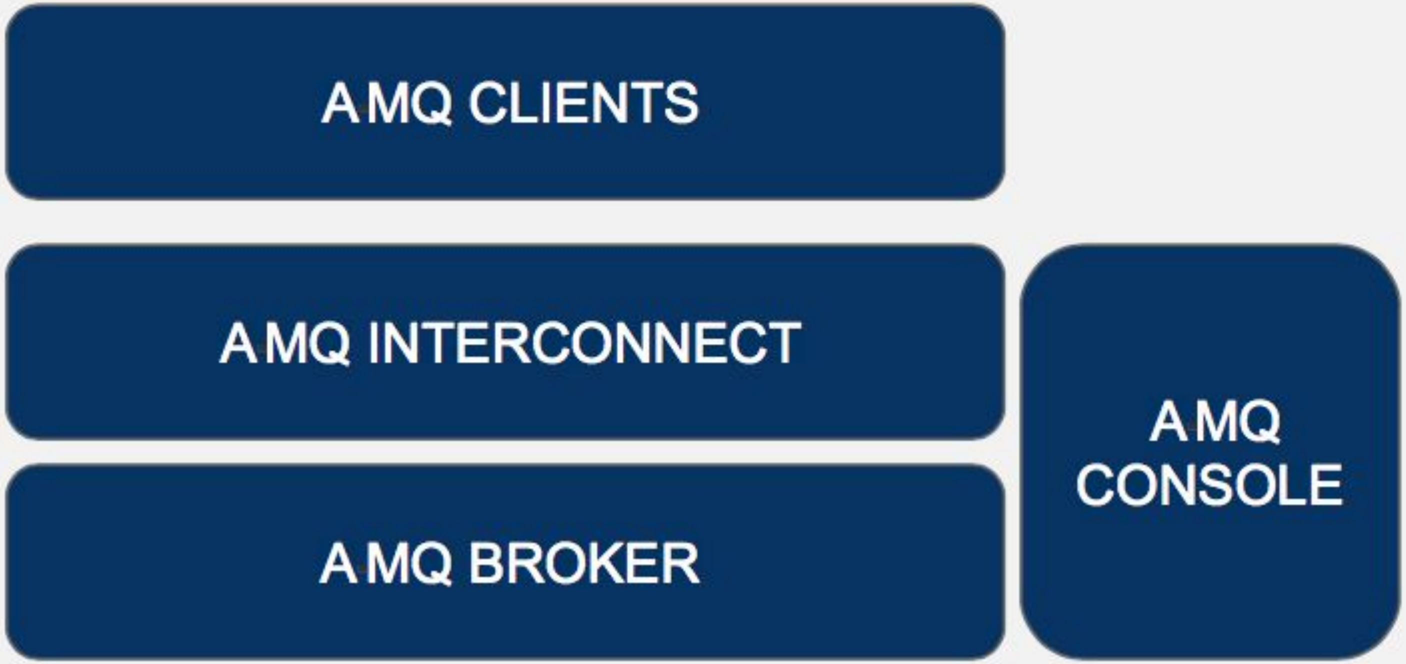
redhat.

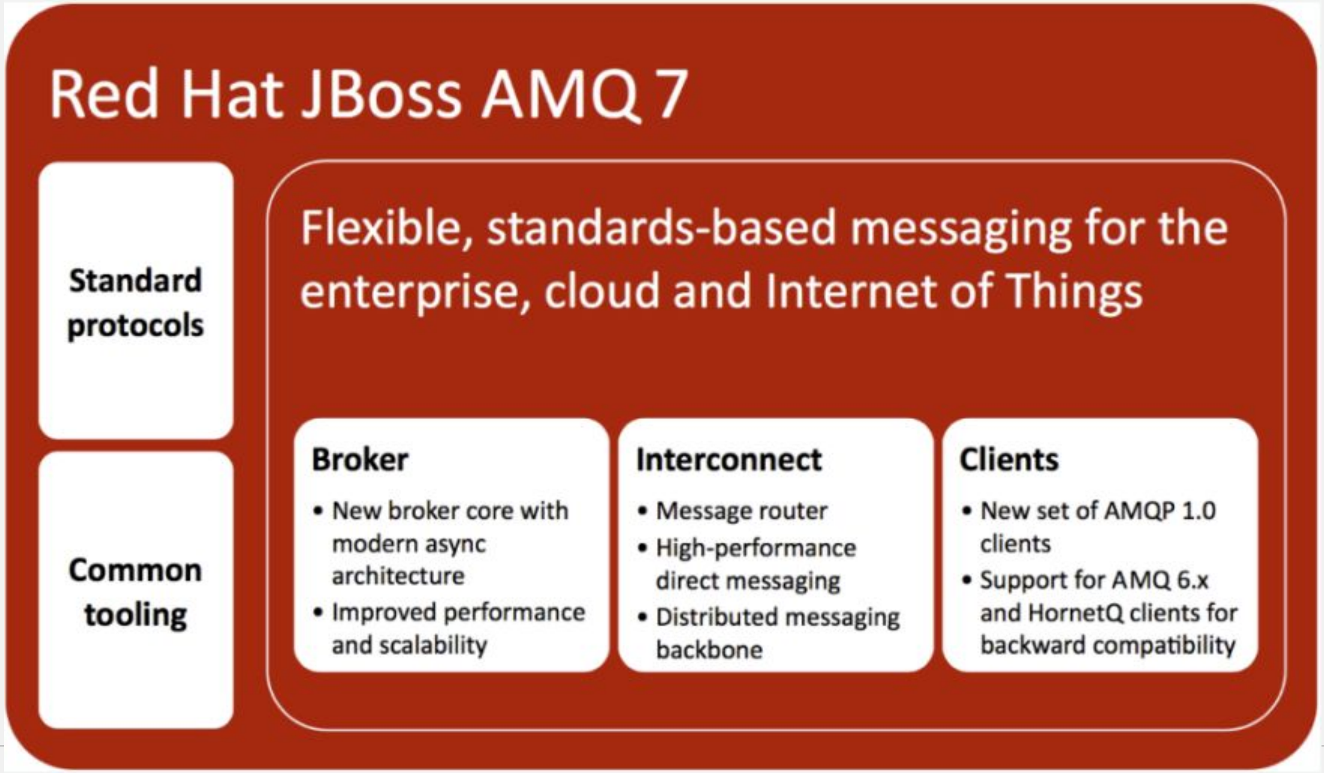
RED HAT® AMQ 7

Stelios Kousouris
Senior Applications Development Architect

AMQ 7 Components



AMQ 7 Features



Red Hat JBoss AMQ 7

Standard protocols

Common tooling

Flexible, standards-based messaging for the enterprise, cloud and Internet of Things

Broker

- New broker core with modern async architecture
- Improved performance and scalability

Interconnect

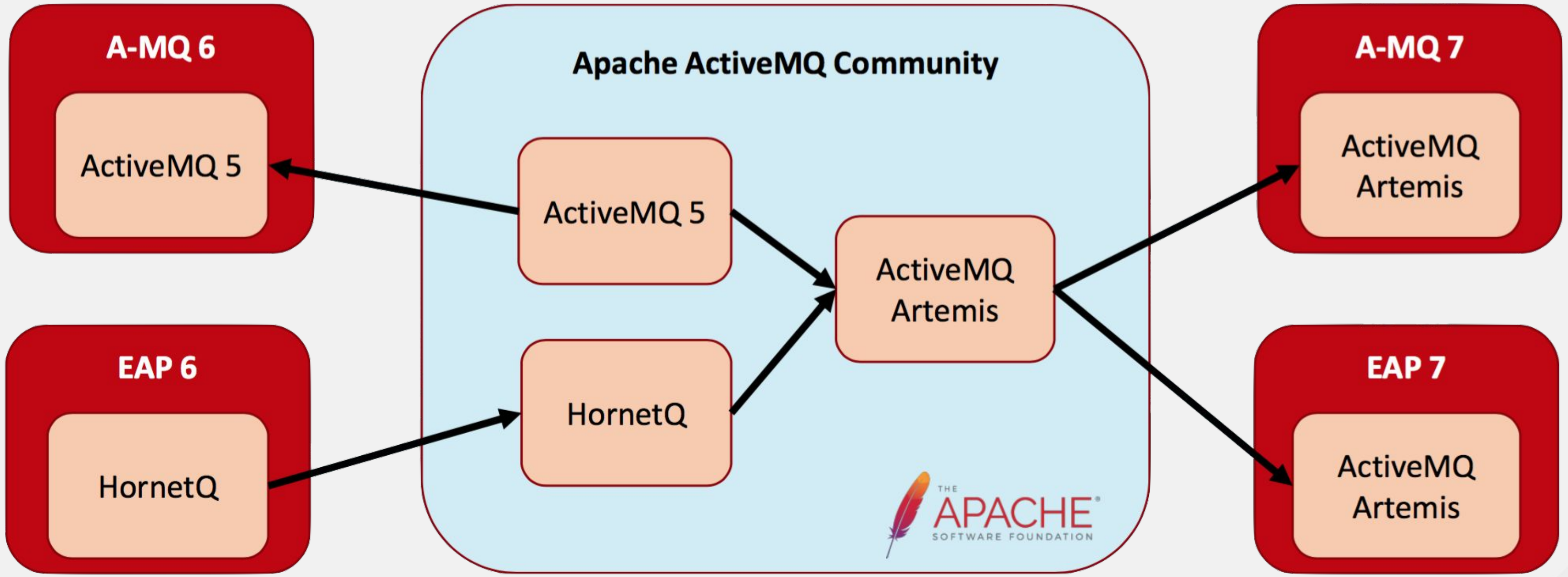
- Message router
- High-performance direct messaging
- Distributed messaging backbone

Clients

- New set of AMQP 1.0 clients
- Support for AMQ 6.x and HornetQ clients for backward compatibility

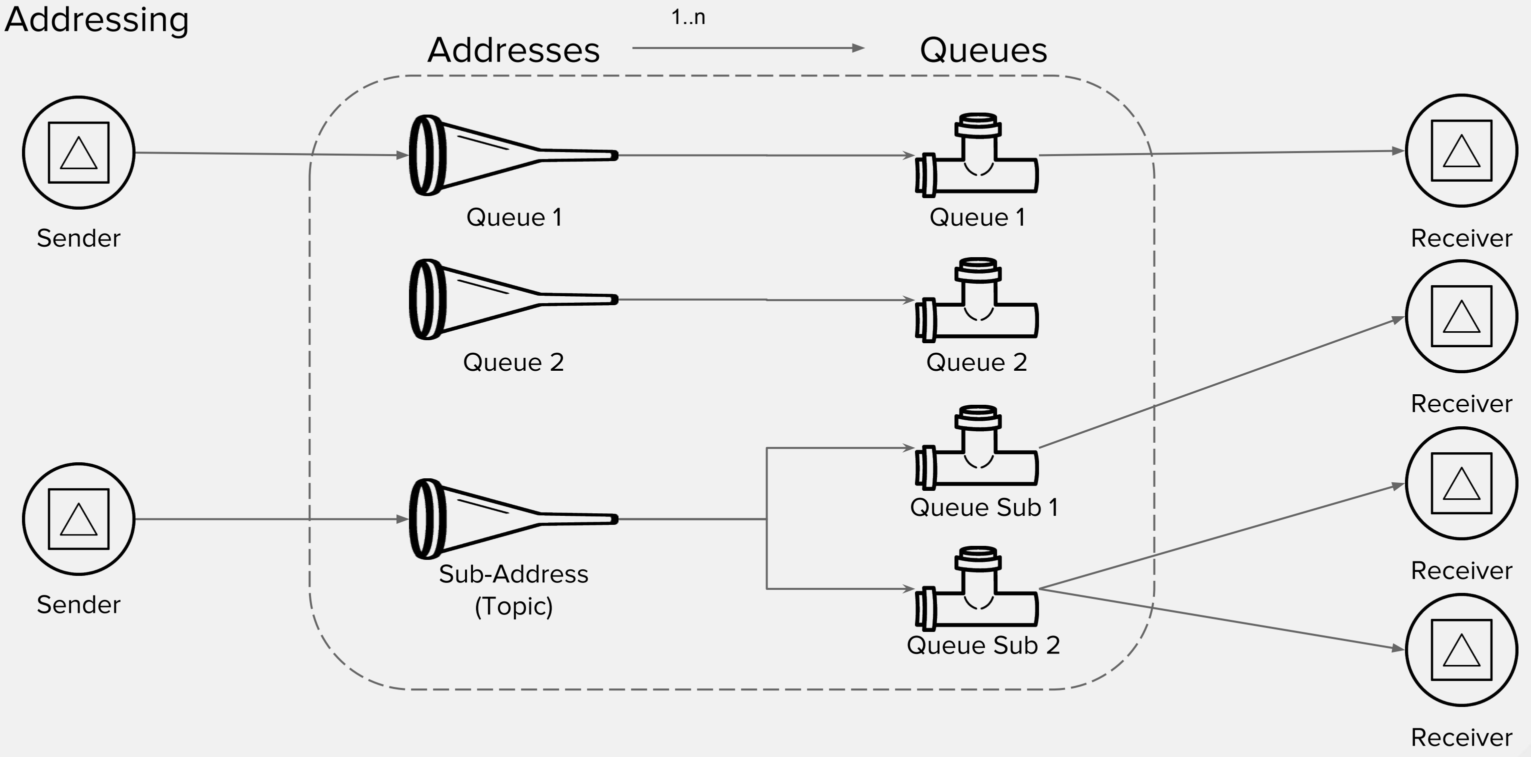
ABOUT AMQ-7 AND AMQP

Broker consolidation

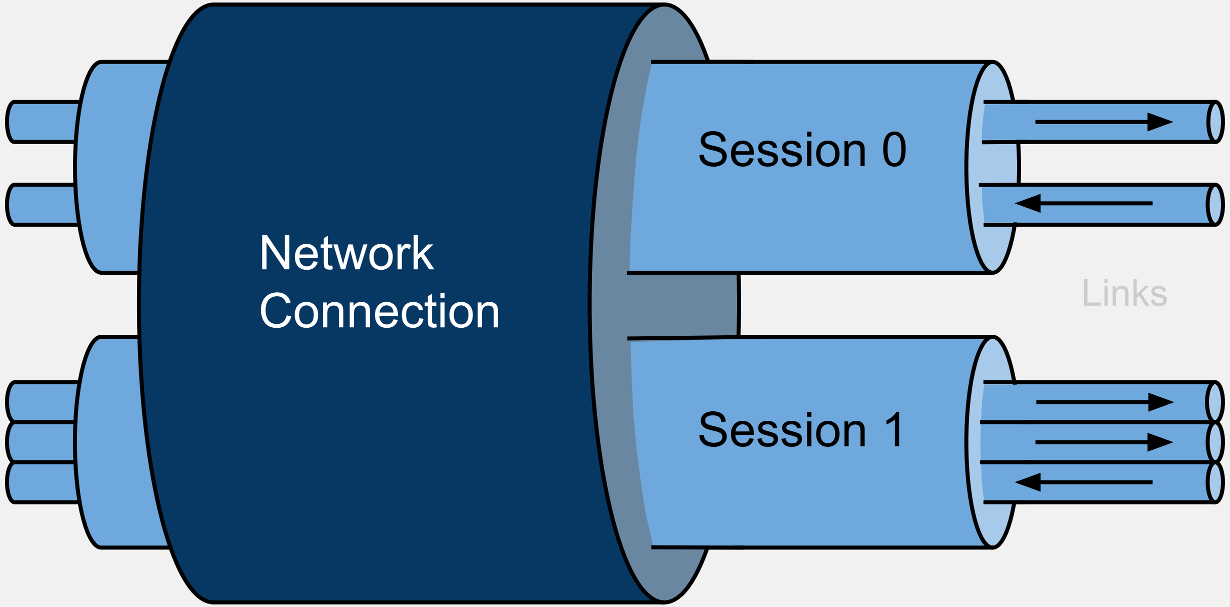


General Concepts

Addressing

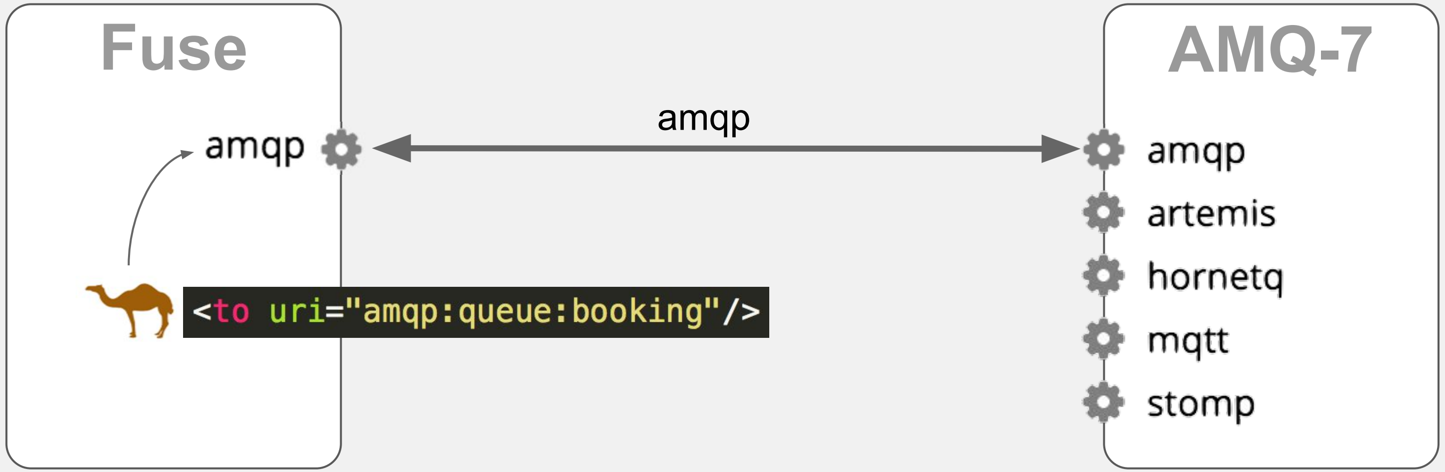


Focus on AMQP



AMQP protocol

Fuse example: the Camel AMQP component is based on the JMS QPid Proton library.

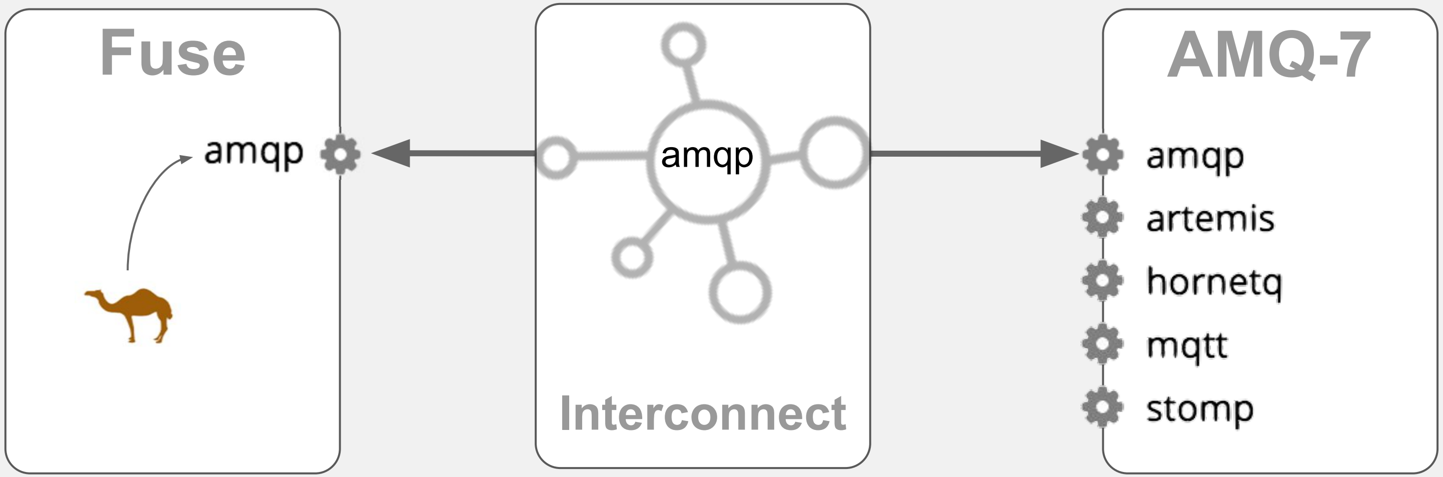


```
<to uri="amqp:queue:booking"/>
```

Enables Interconnect (included in AMQ-7)

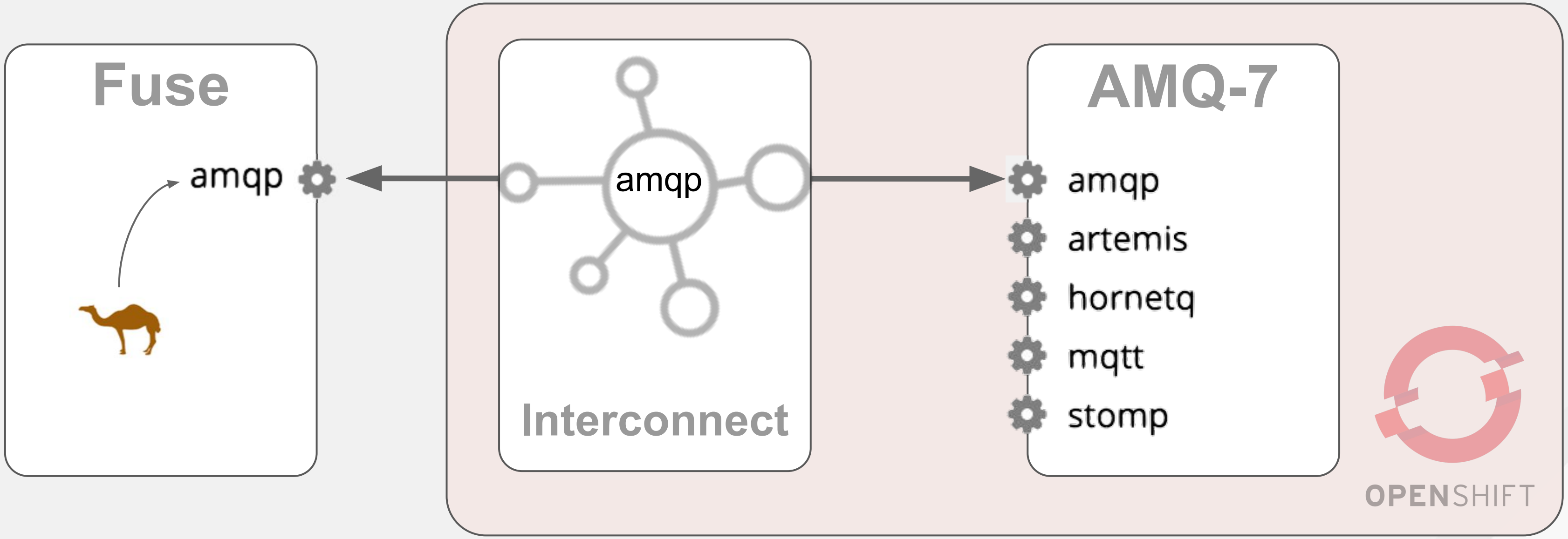
AMQP Communication allows putting Interconnect in the mix

Interconnect enables great elasticity & scalability



Enables Message-as-a-Service

AMQP messaging leverages your systems to be future-ready



AMQP

Advanced Message Queuing Protocol v 1.0

Open, general-purpose message transfer standard protocol that enables cross-platform apps.

International Standard OASIS and ISO/IEC 19464

Efficient - binary, connection-oriented protocol

Reliable - range of guarantees, fire-&-forget to reliable, exactly-once delivery

Portable data representation - cross-platform, full-fidelity message exchange

Sophisticated flow control - credit-based flow control & channel multiplexing

Flexible - client-client, client-broker & broker-broker topologies

Broker-model independent - no requirements on broker internals

ISO/IEC 19464:2014

http://www.amqp.org

AMQP 1.0 client libraries

for A-MQ 7 Broker and Interconnect

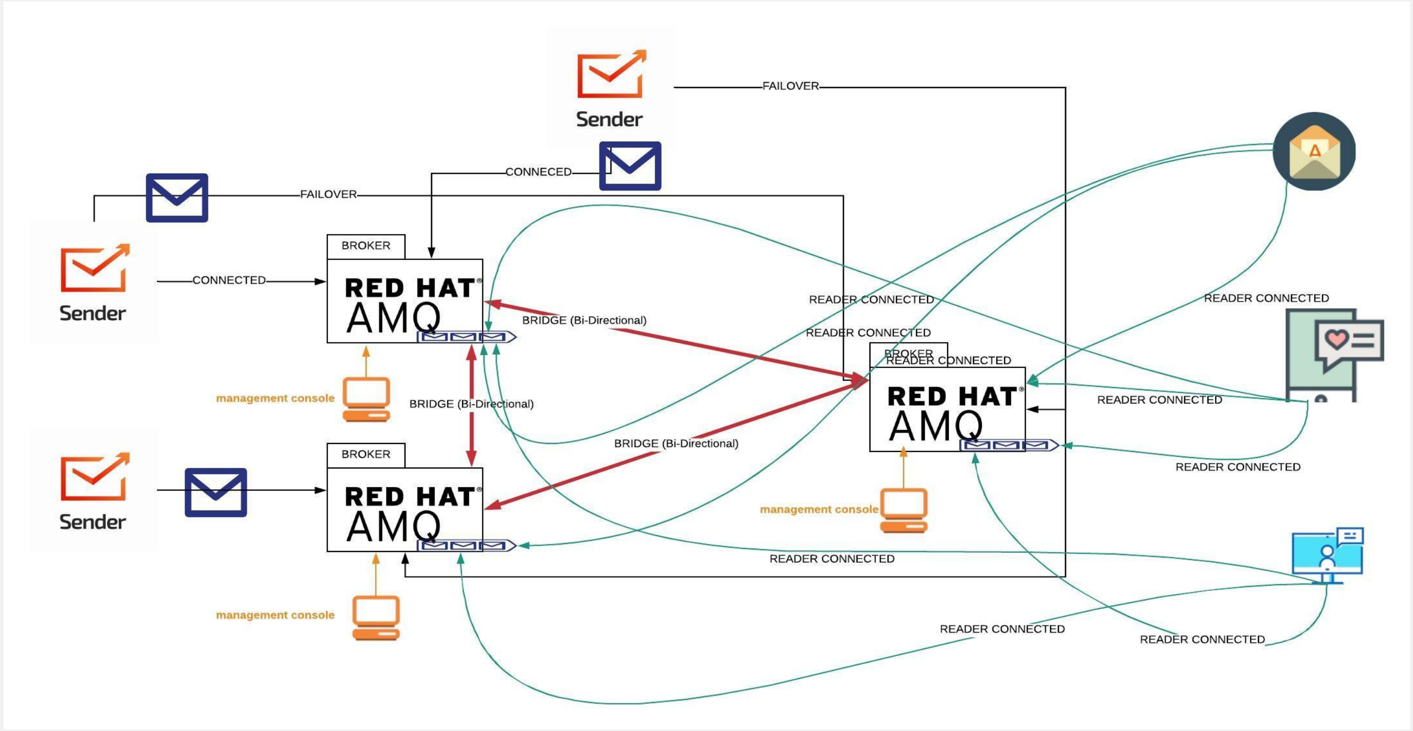
- Java JMS 1.1 client (Apache Qpid JMS based on Qpid Proton)
- Reactive C++ client (Apache Qpid Proton)
- Reactive Python client (Apache Qpid Proton)
- Reactive pure JavaScript client w support for Node.js (GitHub Rhea)
- Fully-featured .NET library (GitHub AMQP .NET Lite)

AMQ 7 Usage Modes

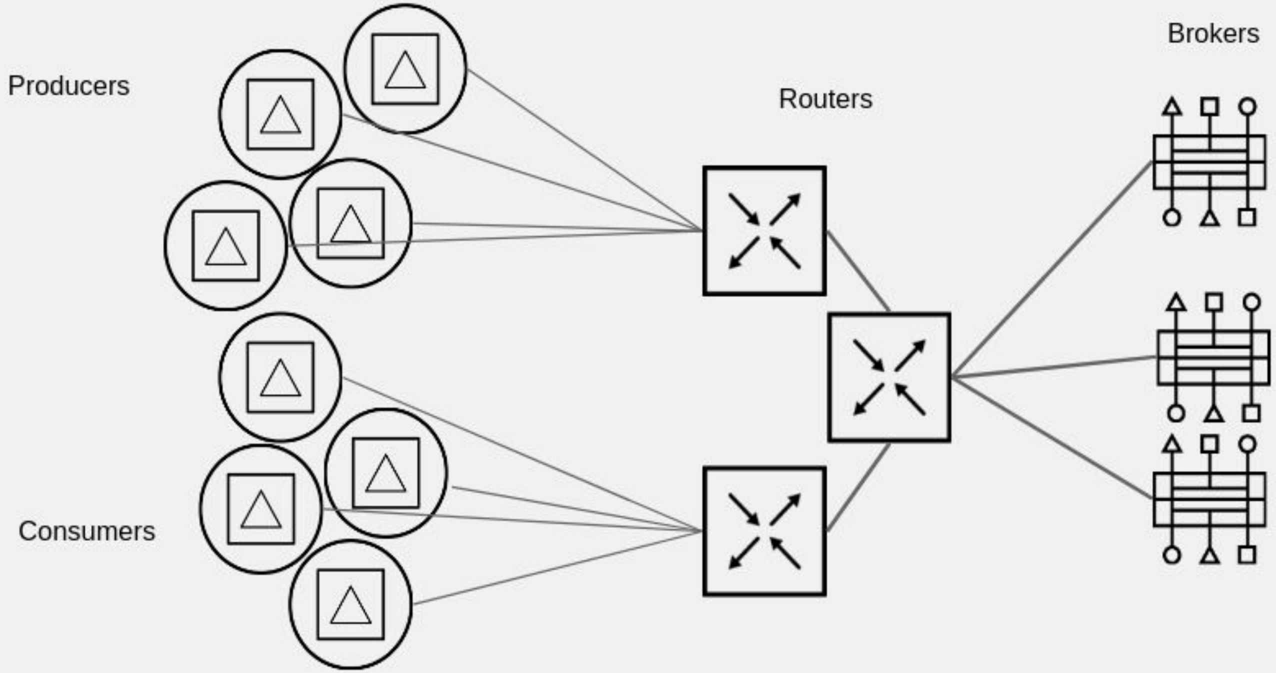
AMQ 7 Usage Modes

- AMQ 7 Brokers Only
- Self-managed Interconnect and Brokers
- AMQ Online

AMQ 7 Brokers Only

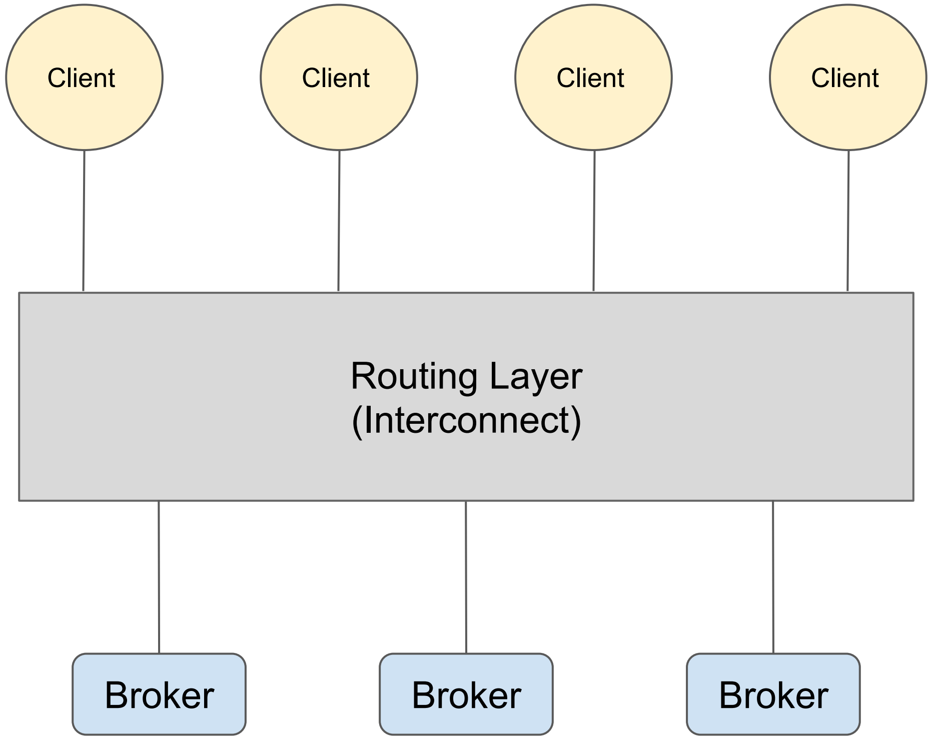


Self Managed Interconnect & Brokers



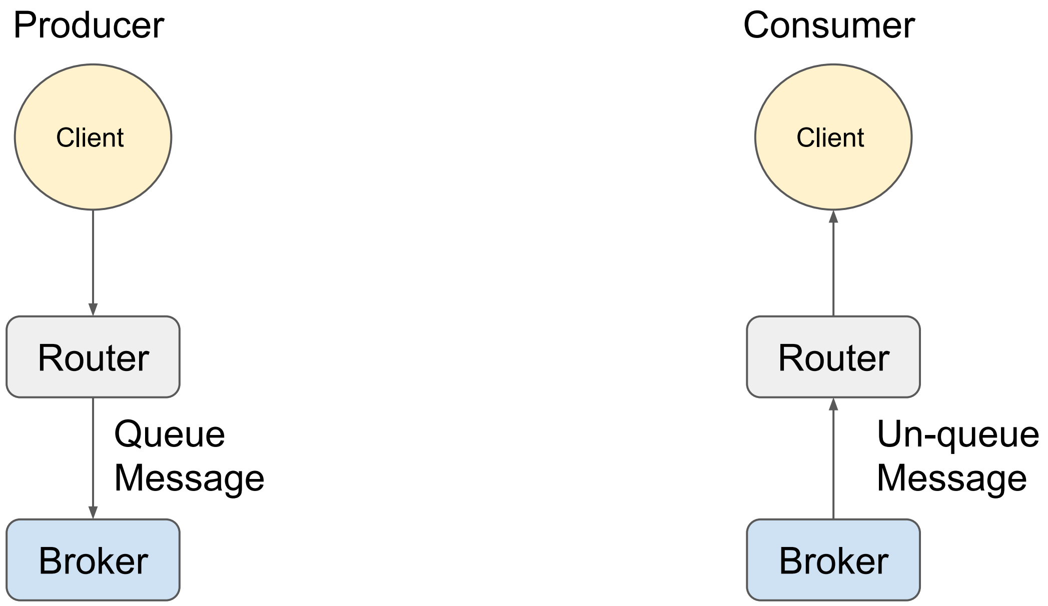
Routing: abstraction layer

Scaling Up...

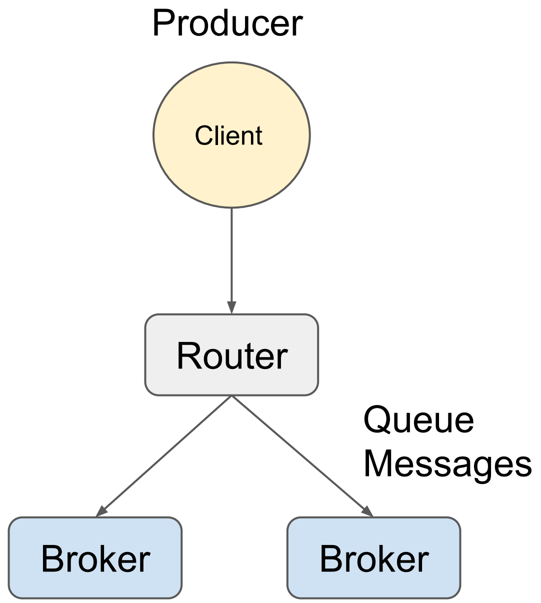


Scaling Down...

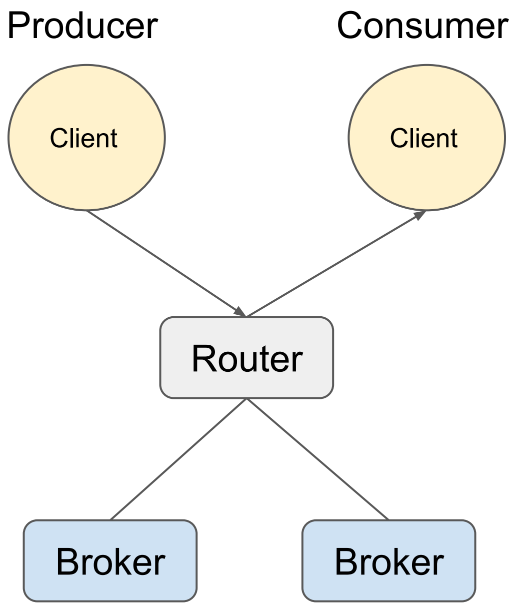
Routing: abstraction layer



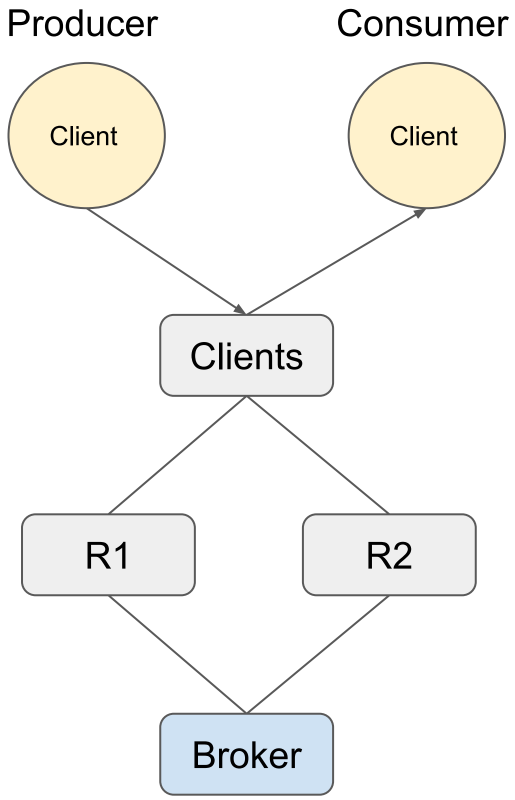
Routing: queue sharding



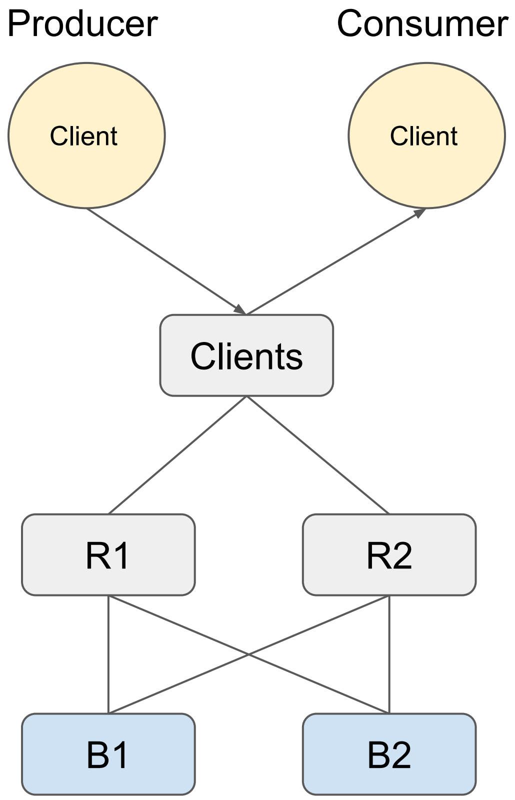
Routing: clients unaware of interruptions



Routing: high availability with multiple paths

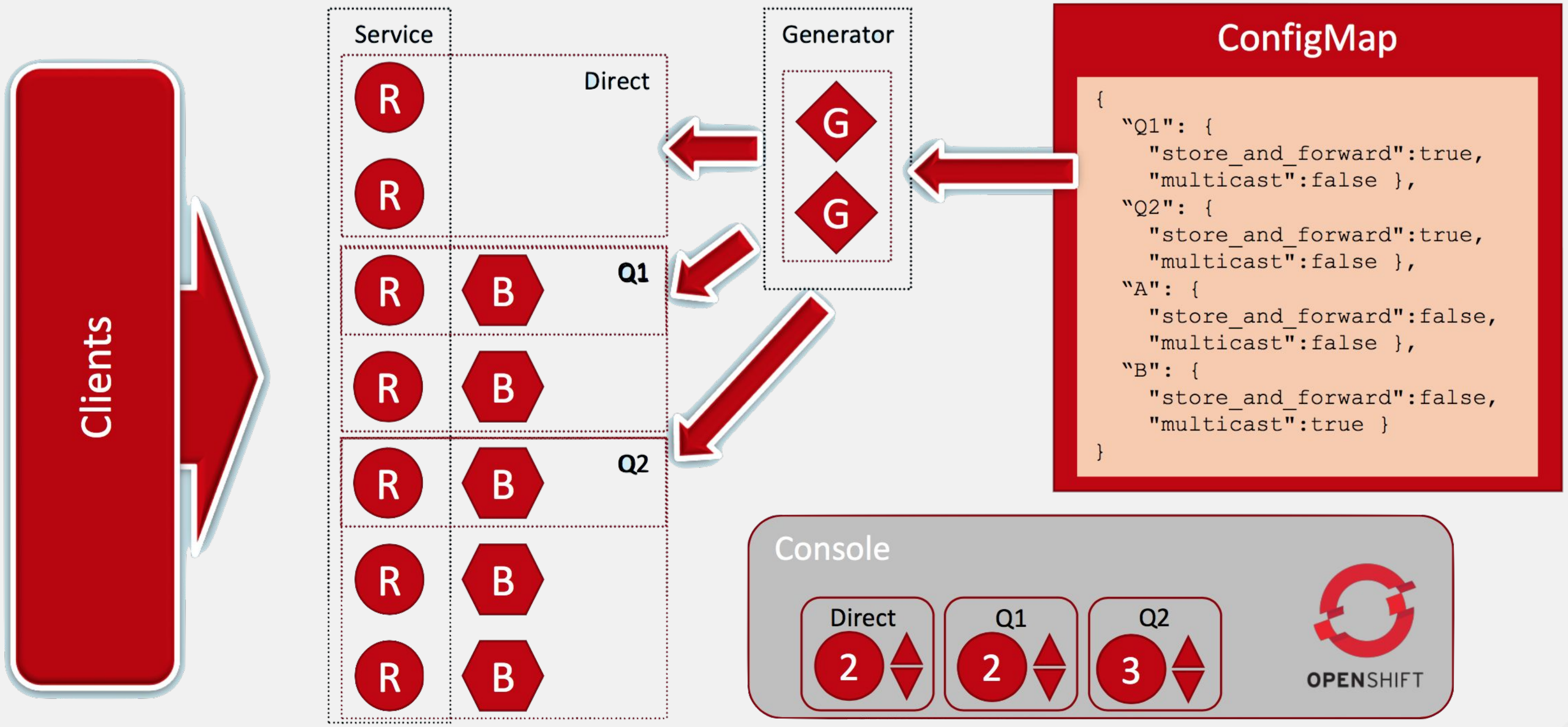


Routing: high availability with multiple paths



ABOUT AMQ Online

AMQ-7 on OpenShift



```
{
  "Q1": {
    "store_and_forward":true,
    "multicast":false },
  "Q2": {
    "store_and_forward":true,
    "multicast":false },
  "A": {
    "store_and_forward":false,
    "multicast":false },
  "B": {
    "store_and_forward":false,
    "multicast":true }
}
```

AMQ-Online - Infrastructure Components

- Routers,
- Brokers,
- Consoles

AMQ Online service operator can edit the AMQ Online default infrastructure configuration or create new configurations.

Infrastructure configuration can be managed for both ***brokered*** and ***standard*** infrastructure using

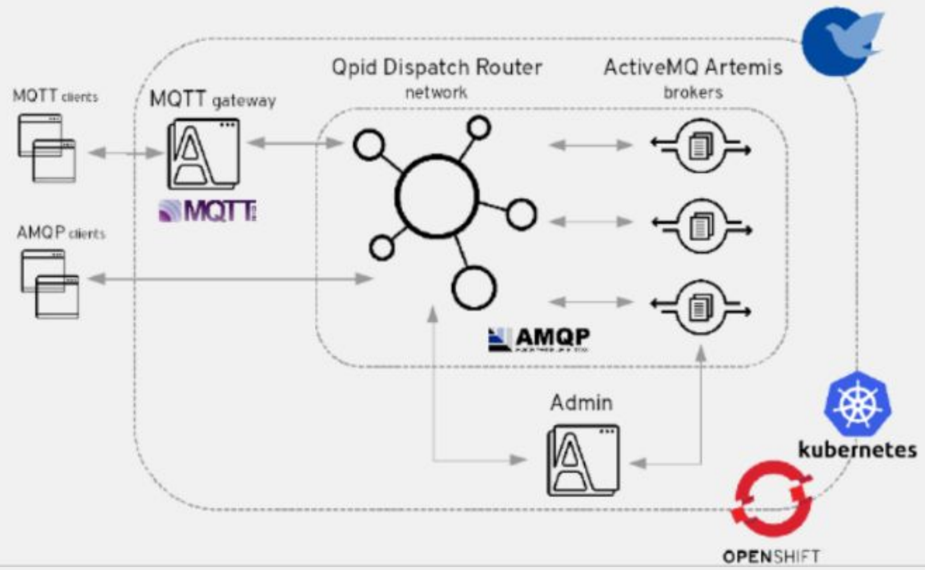
- ***BrokeredInfraConfig*** and
- ***StandardInfraConfig***

resources.

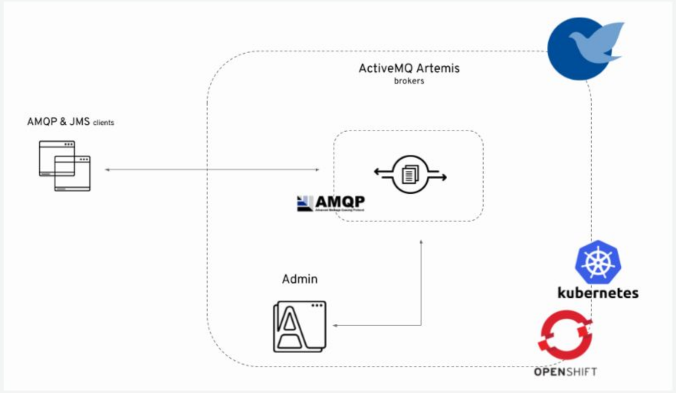
AMQ-Online - Address Space Types

Brokered vs Standard

Standard address space type



Brokered Address Architecture



AMQ-Online - Space Infra Configuration

```yaml
apiVersion: admin.enmasse.io/v1alpha1
kind: BrokeredInfraConfig
metadata:
  name: brokered-infra-config-example
spec:
  version: 0.23.0
  admin:
    resources:
      memory: 256Mi
  broker:
    resources:
      memory: 2Gi
      storage: 100Gi
    addressFullPolicy: PAGE
```

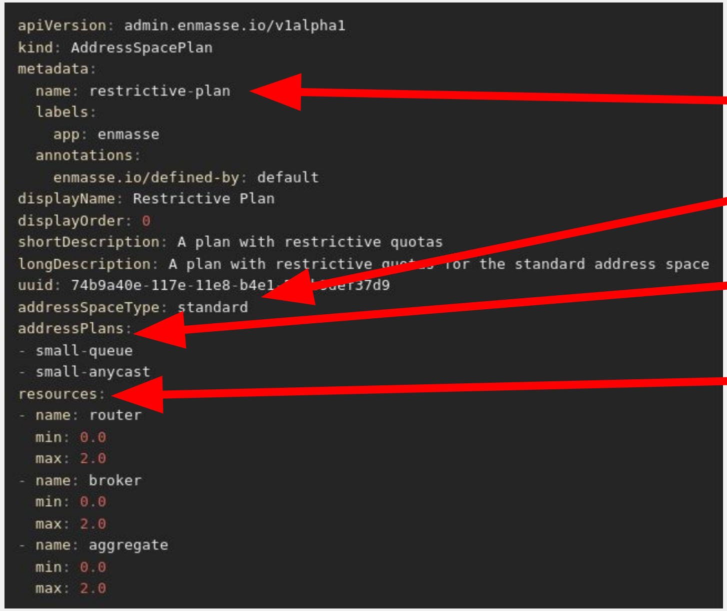
```yaml
apiVersion: admin.enmasse.io/v1alpha1
kind: StandardInfraConfig
metadata:
  name: myconfig
spec:
  version: 0.23.0
  admin:
    resources:
      memory: 256Mi
  broker:
    resources:
      memory: 2Gi
      storage: 100Gi
    addressFullPolicy: PAGE
  router:
    resources:
      memory: 256Mi
    linkCapcity: 1000
```

- The ***version*** field specifies the AMQ Online version used (used whether to upgrade the infrastructure to the requested version during an upgrade)
- The ***admin*** object specifies the settings you can configure for the admin components.
- The ***broker*** object specifies the settings you can configure for the broker components. Changing the ***.broker.resources.storage*** setting does not configure existing broker storage size.
- The ***router*** object specifies the settings you can configure for the router components.

AMQ-Online - Address Space Plans

Brokered vs Standard

Address space plans are used to configure quotas and control the resources consumed by address spaces. Address space plans are configured by the AMQ Online service operator and are selected when creating an address space.



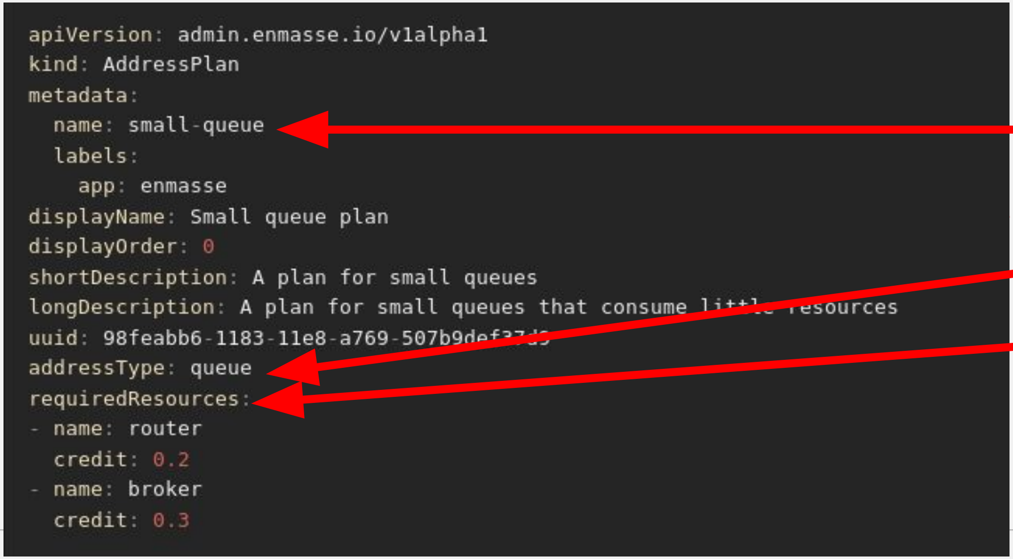
```
apiVersion: admin.enmasse.io/v1alpha1
kind: AddressSpacePlan
metadata:
  name: restrictive-plan
  labels:
    app: enmasse
  annotations:
    enmasse.io/defined-by: default
displayName: Restrictive Plan
displayOrder: 0
shortDescription: A plan with restrictive quotas
longDescription: A plan with restrictive quotas for the standard address space
uuid: 74b9a40e-117e-11e8-b4e1-[illegible]3der37d9
addressSpaceType: standard
addressPlans:
- small-queue
- small-anycast
resources:
- name: router
  min: 0.0
  max: 2.0
- name: broker
  min: 0.0
  max: 2.0
- name: aggregate
  min: 0.0
  max: 2.0
```

The following fields are required:

- metadata.name
- addressSpaceType
- addressPlans
- resources

AMQ-Online - Address Plans

- Address plans specify the expected resource usage of a given address.
- The sum of the resource usage for all resource types determines the amount of infrastructure provisioned for an address space.
- A single router and broker pod has a maximum usage of one.
- If a new address requires additional resources and the resource consumption is within the address space plan limits, a new pod will be created automatically to handle the increased load.
- Address plans are configured by the AMQ Online service operator and are selected when creating an address.

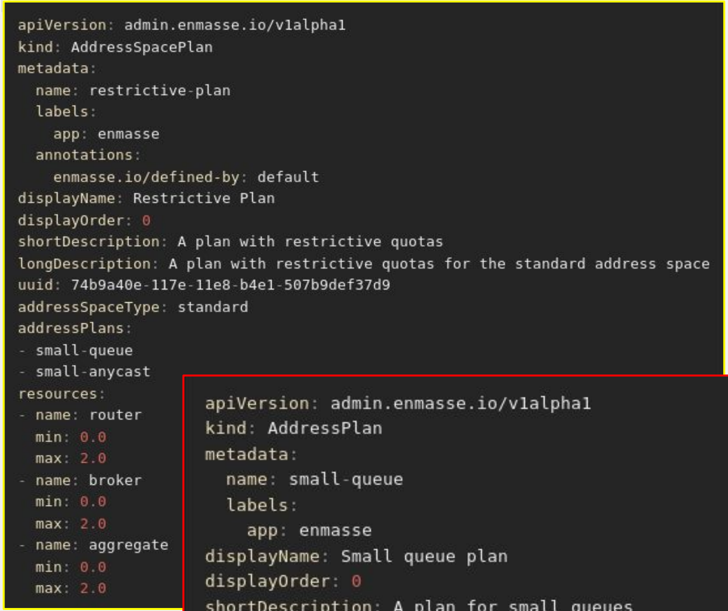


```yaml
apiVersion: admin.enmasse.io/v1alpha1
kind: AddressPlan
metadata:
  name: small-queue
  labels:
    app: enmasse
displayName: Small queue plan
displayOrder: 0
shortDescription: A plan for small queues
longDescription: A plan for small queues that consume little resources
uuid: 98feabb6-1183-11e8-a769-507b9def37d9
addressType: queue
requiredResources:
- name: router
  credit: 0.2
- name: broker
  credit: 0.3
```

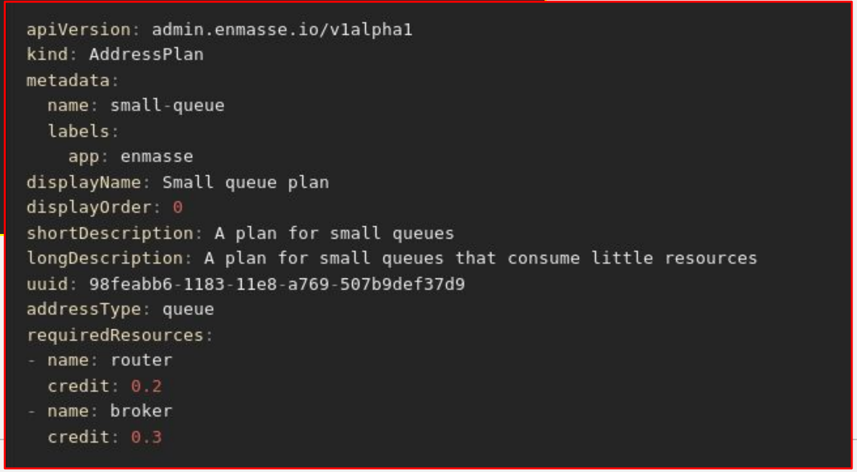
The following fields are required:

- metadata.name
- addressType
- requiredResources

AMQ-Online - Resource Usage



```
apiVersion: admin.enmasse.io/v1alpha1
kind: AddressSpacePlan
metadata:
  name: restrictive-plan
  labels:
    app: enmasse
  annotations:
    enmasse.io/defined-by: default
displayName: Restrictive Plan
displayOrder: 0
shortDescription: A plan with restrictive quotas
longDescription: A plan with restrictive quotas for the standard address space
uuid: 74b9a40e-117e-11e8-b4e1-507b9def37d9
addressSpaceType: standard
addressPlans:
- small-queue
- small-anycast
resources:
- name: router
  min: 0.0
  max: 2.0
- name: broker
  min: 0.0
  max: 2.0
- name: aggregate
  min: 0.0
  max: 2.0
```



```
apiVersion: admin.enmasse.io/v1alpha1
kind: AddressPlan
metadata:
  name: small-queue
  labels:
    app: enmasse
displayName: Small queue plan
displayOrder: 0
shortDescription: A plan for small queues
longDescription: A plan for small queues that consume little resources
uuid: 98feabb6-1183-11e8-a769-507b9def37d9
addressType: queue
requiredResources:
- name: router
  credit: 0.2
- name: broker
  credit: 0.3
```

AddressPlan:

- single router can support 5 instances of addresses (since 0.2 x 5 = 1 router)
- broker supports 3 instances of addresses with this plan (since 0.3 x 3 = 0.9 router)
- If the # of addresses with this plan increases to 4, another broker is created.
- If it increases further to 6, another router is created as well.

Note, however, that although the address space plan allows two routers and two brokers to be deployed, it only allows two pods to be deployed in total. This means that the address space is restricted to three addresses with the small-queue plan.

AMQ-Online - Address Spaces & Addresses

Address Space

- Group of addresses accessible through single connection
- Type: Defines the infrastructure to be deployed
- Plan: May define quota restrictions (i.e., maximum number of addresses)
- Available plans determined by the type of address space

Address

- Type: Defines semantics of address
- Plan: Affects what infrastructure is used or deployed for this address
- Has set of properties
- Available address types determined by the address space type
- Available plans determined by the address type

redhat.

AMQ-Online - Address Types

- Focused on scaling number of connections and throughput
- Supports AMQP and MQTT protocols
- Based on Apache Artemis and Apache Qpid Dispatch router projects
- Supports four address types:
  - Queue: Store-and-forward semantics
  - Topic: n:m publish and subscribe
  - Anycast: Direct address for sending to one consumer
  - Multicast: Direct address for sending to multiple consumers

AMQ-Online - Monitoring

Deploying Prometheus

Deploying kube-state-metrics

Deploying Alertmanager

https://access.redhat.com/documentation/en-us/red_hat_amq/7.2/html-single/using_amq_online_on_openshift_container_platform/#monitoring_amq_online

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos