Red Hat AMQ 7.2

Installing and Managing AMQ Online on OpenShift Container Platform

For use with AMQ Online 1.0

Last Updated: 2019-03-11

Red Hat AMQ 7.2 Installing and Managing AMQ Online on OpenShift Container Platform

For use with AMQ Online 1.0

Legal Notice

Abstract

This guide describes how to install and manage AMQ Online.

Table of Contents

CHAPTER 1. INTRODUCTION

1.1. AMQ ONLINE OVERVIEW

Red Hat AMQ Online is an OpenShift-based mechanism for delivering messaging as a managed service. With Red Hat AMQ Online, administrators can configure a cloud-native, multi-tenant messaging service either in the cloud or on premise. Developers can provision messaging using the AMQ Online console. Multiple development teams can provision the brokers and queues from the console, without requiring each team to install, configure, deploy, maintain, or patch any software.

AMQ Online can provision different types of messaging depending on your use case. A user can request messaging resources by creating an address space. AMQ Online currently supports two address space types, standard and brokered, each with different semantics. The following diagrams illustrate the high-level architecture of each address space type:

Figure 1.1. Standard address space

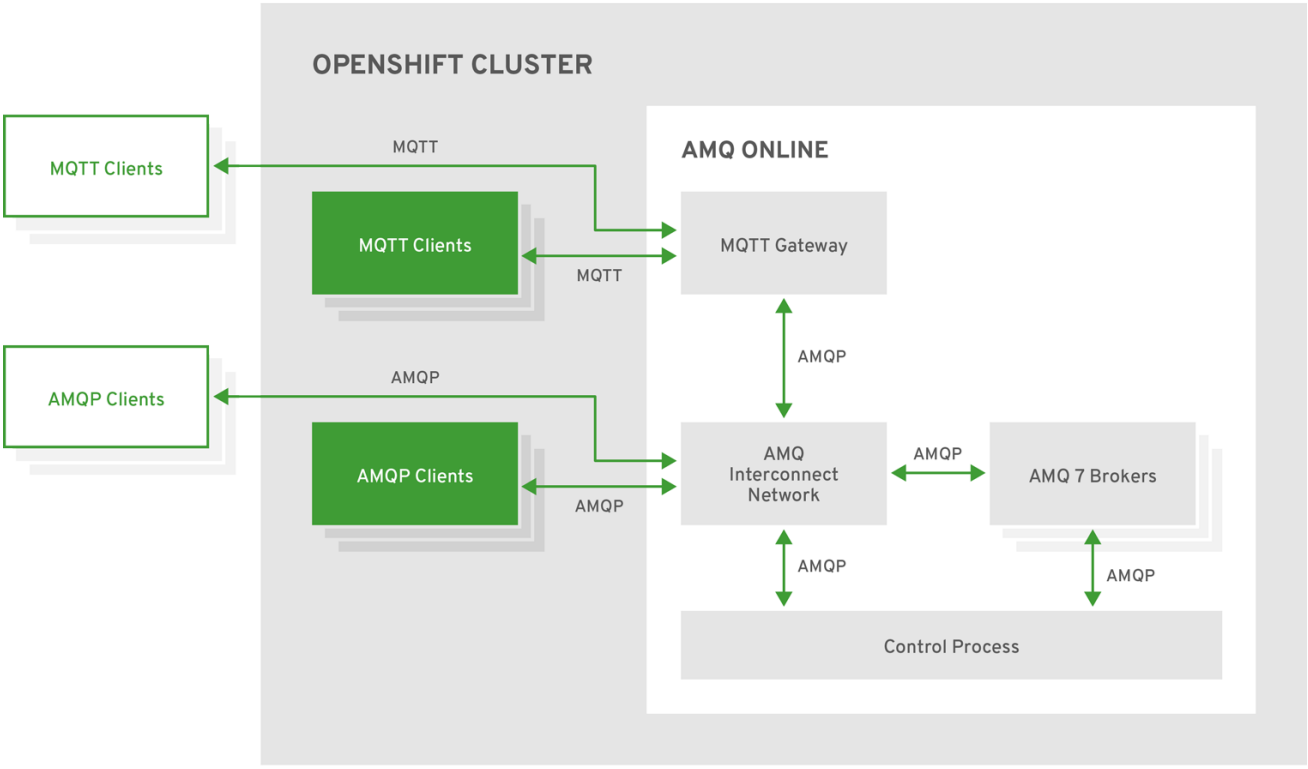
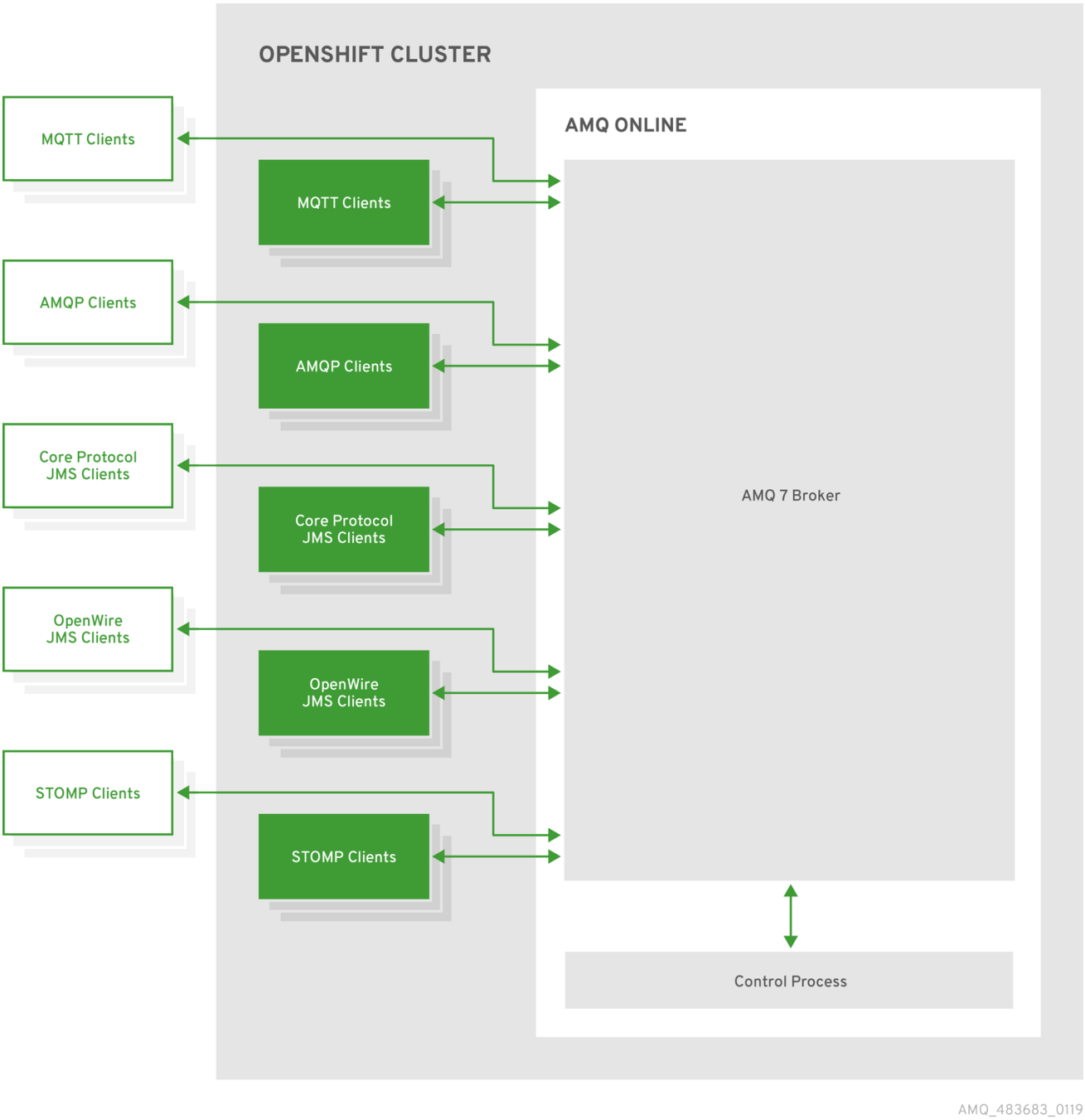


Figure 1.2. Brokered address space



1.2. SUPPORTED FEATURES

The following table shows the supported features for AMQ Online 1.0:

Table 1.1. Supported features reference table

Feature		Brokered address space	Standard address space
Address type	Queue	Yes	Yes
	Topic	Yes	Yes

Feature		Brokered address space	Standard address space
	Multicast	No	Yes
	Anycast	No	Yes
	Subscription	No	Yes
Messaging protocol	AMQP	Yes	Yes
	MQTT	Yes	Technology preview only
	CORE	Yes	No
	OpenWire	Yes	No
	STOMP	Yes	No
Transports	TCP	Yes	Yes
	WebSocket	Yes	Yes
Durable subscriptions	JMS durable subscriptions	Yes	No
	"Named" durable subscriptions	No	Yes
JMS	Transaction support	Yes	No
	Selectors on queues	Yes	No
	Message ordering guarantees (including prioritization)	Yes	No
Scalability	Scalable distributed queues and topics	No	Yes

1.3. AMQ ONLINE USER ROLES

AMQ Online users can be defined broadly in terms of two user roles: service administrator and messaging tenant. Depending on the size of your organization, these roles might be performed by the same person or different people.

The service administrator performs the initial installation and any subsequent upgrades. The service administrator might also deploy and manage the messaging infrastructure, such as monitoring the

routers, brokers, and administration components; and creating the address space plans and address plans. *Installing and Managing AMQ Online on OpenShift Container Platform* provides information about how to set up and manage AMQ Online as well as configure the infrastructure and plans as a service administrator.

The messaging tenant can request messaging resources, using both cloud-native APIs and tools. The messaging tenant can also manage the users and permissions of a particular address space within the messaging system as well as create address spaces and addresses. For more information about how to manage address spaces, addresses, and users, see *Using AMQ Online on OpenShift Container Platform*.

1.4. SUPPORT STATEMENT AND RESOURCES

The AMQ Online 1.0 release is provided as a service to customers who want to try the AMQ Online features and work with support when problems are encountered. Support for Beta releases is limited to commercially reasonable effort and non-production use cases, and all support cases must be opened with a severity of 4. Patches will not be provided, but bug fixes might be incorporated in future releases. To contact support, visit Open a Support Case.

1.5. SUPPORTED CONFIGURATIONS

For more information about AMQ Online supported configurations see Red Hat AMQ 7 Supported Configurations.

1.6. DOCUMENT CONVENTIONS

1.6.1. Variable text

This document contains code blocks with variables that you must replace with values specific to your installation. In this document, such text is styled as italic monospace.

For example, in the following code block, replace ***my-namespace*** with the namespace used in your installation:

```
sed -i 's/amq-online-infra/my-namespace/' install/bundles/enmasse-with-
standard-authservice/*.yaml
```

CHAPTER 2. INSTALLING AMQ ONLINE

AMQ Online can be installed using automated Ansible playbooks, the OpenShift template, or the manual steps.

Prerequisites

- To install AMQ Online, the OpenShift client tools are required.
- An OpenShift cluster is required.
- A user on the OpenShift cluster with **`cluster-admin`** permissions is required to set up the required cluster roles and API services.

2.1. DOWNLOADING AMQ ONLINE

Procedure

- Download and extract the **`amq-online-install.zip`** file from the AMQ Online download site.

NOTE

Although container images for AMQ Online are available in the Red Hat Container Catalog, we recommend that you use the YAML files provided instead.

2.2. INSTALLING AMQ ONLINE USING A YAML BUNDLE

The simplest way to install AMQ Online is to use the predefined YAML bundles.

Procedure

1. Log in as a user with **`cluster-admin`** privileges:

```
oc login -u system:admin
```

2. (Optional) If you want to deploy to a namespace other than **`amq-online-infra`** you must run the following command and substitute **`amq-online-infra`** in subsequent steps:

```
sed -i 's/amq-online-infra/my-namespace/' install/bundles/amq-
online/*.yaml
```

3. Create the project where you want to deploy AMQ Online:

```
oc new-project amq-online-infra
```

4. Deploy using the **`amq-online`** bundle:

```
oc apply -f install/bundles/amq-online
```

2.3. INSTALLING AMQ ONLINE USING OPENSHIFT TEMPLATE

Installing AMQ Online using the OpenShift template is useful for evaluating AMQ Online. For a production setup, it is recommended to use one of the following installation methods instead:

- Installing AMQ Online using a YAML bundle
- Installing AMQ Online using Ansible
- Installing AMQ Online manually

Procedure

1. Log in as a user with **`cluster-admin`** privileges:

```
oc login -u system:admin
```

2. Create the project where you want to deploy AMQ Online:

```
oc new-project amq-online-infra
```

3. Deploy using **`amq-online.yaml`** template:

```
oc process -f install/templates/amq-online.yaml NAMESPACE=amq-
online-infra | oc apply -f -
```

2.4. INSTALLING AMQ ONLINE USING ANSIBLE

Installing AMQ Online using Ansible requires creating an inventory file with the variables for configuring the system. Example inventory files can be found in the **`ansible/inventory`** folder.

An example inventory file that enables both the API server and service broker integration:

```
[enmasse]
localhost ansible_connection=local

[enmasse:vars]
namespace=enmasse-infra
enable_rbac=False
api_server=True
service_catalog=True
register_api_server=True
keycloak_admin_password=admin
authentication_services=["standard"]
enable_monitoring=False
```

The following Ansible configuration settings are supported:

Table 2.1. Ansible configuration settings

Name	Description	Default value	Required
namespace	Specifies the namespace where AMQ Online is installed.	Not applicable	yes

Name	Description	Default value	Required
enable_rbac	Specifies whether to enable RBAC authentication of REST APIs	True	no
service_catalog	Specifies whether to enable integration with the Service Catalog	False	no
authentication_services	Specifies the list of authentication services to deploy. Supported values are **none** and **`standard`**.	**none**	no
keycloak_admin_password	Specifies the admin password to use for the **`standard`** authentication service Red Hat Single Sign-On instance	Not applicable	yes (if **`standard`** authentication service is enabled)
api_server	Specifies whether to enable the REST API server	True	no
register_api_server	Specifies whether to register the API server with OpenShift master	False	no
secure_api_server	Specifies whether to enable mutual TLS for the API server	False	no
enable_monitoring	Specifies whether to enable the monitoring services	Not applicable	no
smtp_server	Specifies the SMTP server used to send alert emails	Not applicable	no
smtp_username	Specifies the username used to authenticate to the SMTP server	Not applicable	no
smtp_password	Specifies the password used to authenticate to the SMTP server	Not applicable	no

Name	Description	Default value	Required
smtp_from_address	Specifies the from address displayed in alert emails	Not applicable	no
sysadmin_email	Specifies the email address to which alerts are sent	Not applicable	no

Procedure

1. (Optional) Create an inventory file.
2. Run the Ansible playbook:

```
ansible-playbook -i inventory-file
ansible/playbooks/openshift/deploy_all.yml
```

2.5. INSTALLING AMQ ONLINE MANUALLY

The manual deployment procedure can be performed on any platform supporting the OpenShift client.

2.5.1. Creating the project for AMQ Online

Procedure

- Create the ***amq-online-infra*** project:

```
oc new-project amq-online-infra
```

2.5.2. Deploying authentication services

AMQ Online requires at least one authentication service to be deployed:

- **standard** (Red Hat Single Sign-On) or
- **external** (not managed by AMQ Online).

2.5.2.1. Deploying the standard authentication service

Procedure

1. (Optional) If you want to deploy to a namespace other than **amq-online-infra** you must run the following command:

```
sed -i 's/amq-online-infra/my-namespace/'
install/components/standard-authservice/*RoleBinding*.yaml
```

2. Create the **standard** authentication service:

```
oc create -f ./install/components/standard-authservice
```

2.5.3. Deploying the Address Space Controller

The Address Space Controller is responsible for creating the infrastructure used by address spaces.

Procedure

1. (Optional) If you want to deploy to a namespace other than **amq-online-infra** you must run the following command:

```
sed -i 's/amq-online-infra/my-namespace/'
install/components/address-space-controller/*.yaml
```

2. Deploy the Address Space Controller

```
oc apply -f install/components/address-space-controller
```

2.5.4. Deploying the API server

The API server provides a REST API for creating address spaces and addresses. It can also serve as an aggregated API server if it is registered as an API service.

Procedure

1. (Optional) If you want to deploy to a namespace other than **amq-online-infra** you must run the following command and substitute **amq-online-infra** in subsequent steps:

```
sed -i 's/amq-online-infra/my-namespace/' install/components/api-
service/*.yaml
sed -i 's/amq-online-infra/my-namespace/' install/components/api-
server/*.yaml
```

2. Register API service

```
oc apply -f install/components/api-service
```

3. Deploy the API Server

```
oc apply -f install/components/api-server/
```

2.5.5. (Optional) Deploying the service broker

The service broker provides an implementation of the Open Service Broker API that integrates with the OpenShift Service Catalog.

Prerequisites

- The service broker requires the **`standard`** authentication service to be deployed.

Procedure

1. (Optional) If you want to deploy to a namespace other than **`amq-online-infra`** you must run the following command:

```
sed -i 's/amq-online-infra/my-namespace/'
install/components/service-broker/*.yaml
sed -i 's/amq-online-infra/my-namespace/'
install/components/cluster-service-broker/*.yaml
```

2. Deploy the Service Broker:

```
oc apply -f install/components/service-broker
```

3. Register the Service Broker with the OpenShift Service Catalog:

```
oc apply -f install/components/cluster-service-broker
```

CHAPTER 3. CONFIGURING AMQ ONLINE

3.1. ADDRESS SPACE PLANS

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.

AMQ Online includes a default set of plans that are sufficient for most use cases.

Plans are configured as custom resources. The following example shows a plan for the standard address space:

```
apiVersion: admin.enmasse.io/v1beta1
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
  max: 2.0
- name: broker
  max: 2.0
- name: aggregate
  max: 2.0
```

The following fields are required:

- **`metadata.name`**
- **`resources`**
- **`addressPlans`**
- **`addressSpaceType`**

The other fields are used by the AMQ Online Console UI. Note the annotation **`enmasse.io/defined-by`**, which points to an infrastructure configuration that must exist when an address space using this plan is created. For more information about infrastructure configurations, see Infrastructure configuration.

3.2. CREATING ADDRESS SPACE PLANS

Procedure

1. Log in as a service admin:

```
oc login -u system:admin
```

2. Select the project where AMQ Online is installed:

```
oc project amq-online-infra
```

3. Create an address space plan definition:

```
apiVersion: admin.enmasse.io/v1beta1
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
  max: 2.0
- name: broker
  max: 2.0
- name: aggregate
  max: 2.0
```

4. Create the address space plan:

```
oc create -f restrictive-plan.yaml
```

5. Verify that schema has been updated and contains the plan:

```
oc get addressspaceschema standard -o yaml
```

3.3. 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.

AMQ Online includes a default set of address plans that are sufficient for most use cases.

In the Address space plans section, the address space plan references two address plans: **`small-queue`** and **`small-anycast`**. These address plans are stored as custom resources and are defined as follows:

```
apiVersion: admin.enmasse.io/v1beta1
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`**
- **`requiredResources`**
- **`addressType`**

A single router can support five instances of addresses and broker can support three instances of addresses with this plan. If the number of addresses with this plan increases to four, another broker is created. If it increases further to six, 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.

The **`small-anycast`** plan does not consume any broker resources, and can provision two routers at the expense of not being able to create any brokers:

```
apiVersion: admin.enmasse.io/v1beta1
kind: AddressPlan
metadata:
  name: small-anycast
  labels:
    app: enmasse
displayName: Small anycast plan
displayOrder: 0
shortDescription: A plan for small anycast addresses
longDescription: A plan for small anycast addresses that consume little
```

```
resources
uuid: cb61f440-1184-11e8-adda-507b9def37d9
addressType: anycast
requiredResources:
- name: router
  credit: 0.2
```

With this plan, up to 10 addresses can be created.

3.4. CREATING ADDRESS PLANS

Procedure

1. Log in as a service admin:

```
oc login -u system:admin
```

2. Select the project where AMQ Online is installed:

```
oc project amq-online-infra
```

3. Create an address plan definition:

```
apiVersion: admin.enmasse.io/v1beta1
kind: AddressPlan
metadata:
  name: small-anycast
  labels:
    app: enmasse
displayName: Small anycast plan
displayOrder: 0
shortDescription: A plan for small anycast addresses
longDescription: A plan for small anycast addresses that consume
little resources
uuid: cb61f440-1184-11e8-adda-507b9def37d9
addressType: anycast
requiredResources:
- name: router
  credit: 0.2
```

4. Create the address plan:

```
oc create -f small-anycast-plan.yaml
```

5. Verify that schema has been updated and contains the plan:

```
oc get addressspaceschema standard -o yaml
```

3.5. INFRASTRUCTURE CONFIGURATION

AMQ Online creates infrastructure components such as routers, brokers, and consoles. These components can be configured while the system is running, and AMQ Online automatically updates the

components with the new settings. The AMQ Online service operator can edit the AMQ Online default infrastructure configuration or create new configurations.

Infrastructure configurations can be referred to from one or more address space plans. For more information about address space plans, see Address space plans.

Infrastructure configuration can be managed for both **`brokered`** and **`standard`** infrastructure using **`BrokeredInfraConfig`** and **`StandardInfraConfig`** resources.

3.5.1. Brokered infrastructure configuration

`BrokeredInfraConfig` resources are used to configure infrastructure deployed by **`brokered`** address spaces. The brokered infrastructure configuration is referenced by address space plans using a **`enmasse.io/defined-by`** annotation. For more information, see Address space plans.

```
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: brokered-infra-config-example
spec:
  version: 0.26
  admin:
    resources:
      memory: 256Mi
  broker:
    resources:
      memory: 2Gi
      storage: 100Gi
    addressFullPolicy: PAGE
```

The **`version`** field specifies the AMQ Online version used. When upgrading, AMQ Online uses this field to determine whether to upgrade the infrastructure to the requested version.

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 the existing broker storage size.

3.5.2. Standard infrastructure configuration

`StandardInfraConfig` resources are used to configure infrastructure deployed by **`standard`** address spaces. The standard infrastructure configuration is referenced by address space plans using a **`enmasse.io/defined-by`** annotation. For more information, see Address space plans.

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: myconfig
spec:
  version: 0.26
  admin:
    resources:
      memory: 256Mi
  broker:
```

```
    resources:
      memory: 2Gi
      storage: 100Gi
    addressFullPolicy: PAGE
  router:
    resources:
      memory: 256Mi
    linkCapcity: 1000
    minReplicas: 1
```

The **version** field specifies the AMQ Online version used. When upgrading, AMQ Online uses this field to determine whether to upgrade the infrastructure to the requested version.

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 the existing broker storage size.

The **router** object specifies the settings you can configure for the **router** components.

3.6. APPLYING INFRASTRUCTURE CONFIGURATION

You can edit existing configurations or create new ones.

Prerequisites

- An infrastructure configuration change that you want to apply. For more information, see Infrastructure configuration.

Procedure

1. Log in as a service operator:

```
oc login -u developer
```

2. Select the project where AMQ Online is installed:

```
oc project enmasse
```

3. Create infrastructure configuration:

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: myconfig
spec:
  version: 0.26
  admin:
    resources:
      memory: 256Mi
  broker:
    resources:
      memory: 2Gi
      storage: 100Gi
```

```
    addressFullPolicy: PAGE
  router:
    resources:
      memory: 256Mi
    linkCapcity: 1000
    minReplicas: 1
```

4. Apply the configuration changes:

```
oc apply -f standard-infra-config-example.yaml
```

5. Monitor the pods while they are restarted:

```
oc get pods -w
```

The configuration changes will be applied within a couple of minutes.

CHAPTER 4. UPGRADING AMQ ONLINE

AMQ Online supports upgrades between minor versions using cloud native tools and the same mechanism used to apply configuration changes. When upgrading, the updated infrastructure configuration of the new version will trigger the upgrade to start.

Upgrading AMQ Online is done by applying the YAML files for the new version.

4.1. UPGRADING AMQ ONLINE USING A YAML BUNDLE

Prerequisites

- A new release of AMQ Online. For more information, see Downloading AMQ Online.

Procedure

1. Log in as a service operator:

```
oc login -u system:admin
```

2. Select the project where AMQ Online is installed:

```
oc project amq-online-infra
```

3. Apply the new release bundle:

```
oc apply -f install/bundles/amq-online
```

4. Monitor pods while they are restarted:

```
oc get pods -w
```

   The upgrade should cause all pods to be restarted within a couple of minutes.

4.2. UPGRADING AMQ ONLINE USING THE RELEASE TEMPLATE

Prerequisites

- A new release of AMQ Online. For more information, see Downloading AMQ Online.

Procedure

1. Log in as a service operator:

```
oc login -u system:admin
```

2. Select the project where AMQ Online is installed:

```
oc project amq-online-infra
```

3. Apply the new release template:

```
oc process -f install/templates/amq-online.yaml NAMESPACE=amq-
online-infra | oc apply -f -
```

4. Monitor the pods while they are restarted:

```
oc get pods -w
```

The upgrade should cause all pods to be restarted within a couple of minutes.

4.3. UPGRADING AMQ ONLINE USING ANSIBLE

Prerequisites

- A new release of AMQ Online. For more information, see Downloading AMQ Online.

Procedure

1. Log in as a service operator:

```
oc login -u system:admin
```

2. Run the Ansible playbook from the new release:

```
ansible-playbook -i inventory-file
ansible/playbooks/openshift/deploy_all.yml
```

3. Monitor pods while they are restarted:

```
oc get pods -w
```

The pods restart and become active within several minutes.

CHAPTER 5. MONITORING AMQ ONLINE

You can monitor AMQ Online by deploying built-in monitoring tools or using your pre-existing monitoring infrastructure.

5.1. DEPLOYING MONITORING

Monitoring services run frequent health checks on AMQ Online and send alerts when health checks fail. Health is assessed using Prometheus and **`kube-state-metrics`**. Alerting is implemented with **`Alertmanager`**. Grafana is also configured to provide a dashboard of the current status of health checks.

5.1.1. Deploying Prometheus

Procedure

1. Replace the namespace with the namespace AMQ Online is currently deployed to:

```
sed -i 's/amq-online-infra/_my-namespace_/'
install/components/prometheus/*.yaml
```

2. Create the Prometheus deployment:

```
oc apply -f ./install/components/prometheus
```

5.1.2. Deploying kube-state-metrics

Procedure

1. Replace the namespace with the namespace AMQ Online is currently deployed to:

```
sed -i 's/amq-online-infra/_my-namespace_/' install/components/kube-
state-metrics/*.yaml
```

2. Create the **`kube-state-metrics`** deployment:

```
oc apply -f ./install/components/kube-state-metrics
```

5.1.3. Deploying Alertmanager

Prerequisites

- You must have already installed Prometheus and **`kube-state-metrics`**.

Procedure

1. Replace the namespace with the namespace AMQ Online is currently deployed to:

```
sed -i 's/amq-online-infra/_my-namespace_/'
install/components/alertmanager/*.yaml
```

2. Create the **`Alertmanager`** deployment:

```
oc apply -f ./install/components/alertmanager
```

3. (Optional) Configure **`Alertmanager`** to send emails using a custom configmap with the relevant SMTP details:

```
oc apply -f ./alertmanager-configmap.yaml
```

An example **`Alertmanager`** config map:

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: enmasse
  name: alertmanager-config
data:
  alertmanager.yml: |
    global:
      resolve_timeout: 5m
      smtp_smarthost: localhost
      smtp_from: alerts@localhost
      smtp_auth_username: admin
      smtp_auth_password: password
    route:
      group_by: ['alertname']
      group_wait: 60s
      group_interval: 60s
      repeat_interval: 1h
      receiver: 'sysadmins'
    receivers:
    - name: 'sysadmins'
      email_configs:
      - to: sysadmin@localhost
    inhibit_rules:
      - source_match:
          severity: 'critical'
        target_match:
          severity: 'warning'
        equal: ['alertname']
```

5.1.4. Deploying Grafana

Prerequisites

- You must have already installed Prometheus and **`kube-state-metrics`**.

Procedure

1. Replace the namespace with the namespace AMQ Online is currently deployed to:

```
sed -i 's/amq-online-infra/_my-namespace_/'
install/components/grafana/*.yaml
```

2. Create the Grafana deployment:

```
oc apply -f ./install/components/grafana
```

5.2. MONITOR AMQ ONLINE WITH EXISTING INFRASTRUCTURE

AMQ Online can also be monitored using an existing Prometheus server and optionally **`kube-state-metrics`** for the relevant OpenShift infrastructure.

5.2.1. Exposed metrics

The Address Space Controller and API Server export Prometheus metrics about AMQ Online's health on port 8080. These metrics can be scraped using the services created during AMQ Online installation and the following sample Prometheus scrape configuration.

Prerequisites

- Prometheus' service account must have View privileges to the AMQ Online namespace.

Procedure

1. Add the following job to your Prometheus scrape configuration:

```
- job_name: <component-health>
  kubernetes_sd_configs:
    - role: service
      namespaces:
        names:
          - <namespace>
  metrics_path: "/metrics"
  relabel_configs:
  - action: keep
    regex: health.*
    source_labels: [__meta_kubernetes_service_port_name]
```

5.2.2. Alerts

You can enable alerts regarding the health of AMQ Online resources.

Procedure

1. Add the following rules to your Prometheus Rules configuration:

```
- alert: Component_Health
  annotations:
    description: '{{ $labels.summary }}'
    value: "{{ $value }}"
    severity: warning
  expr: health == 1
  for: 60s
- alert: Address_Space_Health
  annotations:
```

```
      description: '{{ $labels.summary }}'
      value: "{{ $value }}"
      severity: warning
    expr: address_spaces_not_ready_total > 0
    for: 300s
  - alert: Address_Health
    annotations:
      description: '{{ $labels.summary }}'
      value: "{{ $value }}"
      severity: warning
    expr: addresses_not_ready_total > 0
    for: 300s
```

5.3. RESTARTING A COMPONENT

Procedure

- Delete the component's pod:

```
oc delete pod -l name=component
```

The component will be automatically restarted in a new pod.

5.4. VIEWING ROUTER LOGS

Procedure

1. List all router pods and choose the pod for the relevant address space:

```
oc get pods -l name=qdrouterd -o go-template --template '{{range
.items}}{{.metadata.name}}{{"\t"}}
{{.metadata.annotations.addressSpace}}{{"\n"}}{{end}}'
```

2. Display the logs for the pod:

```
oc logs pod -c router
```

5.5. VIEWING BROKER LOGS

Procedure

1. List all broker pods and choose the pod for the relevant address space:

```
oc get pods -l role=broker -o go-template --template '{{range
.items}}{{.metadata.name}}{{"\t"}}
{{.metadata.annotations.addressSpace}}{{"\n"}}{{end}}'
```

2. Display the logs for the pod:

```
oc logs pod
```

5.6. USING QDSTAT

You can use **qdstat** to monitor the AMQ Online service.

5.6.1. Viewing router connections using qdstat

You can view the router connections using **qdstat**.

Procedure

1. On the command line, run the following command to obtain the **podname** value needed in the following step:

```
oc get pods
```

2. On the command line, run the following command:

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b
127.0.0.1:7777 -c

Connections
  id   host                 container
role    dir  security                              authentication
tenant

=================================================================
=================================================================
=================
  3    172.17.0.9:34998     admin-78794c68c8-9jdd6
normal  in   TLSv1.2(ECDHE-RSA-AES128-GCM-SHA256)
CN=admin,O=io.enmasse(x.509)
  12   172.30.188.174:5671  27803a14-42d2-6148-9491-a6c1e69e875a
normal  out  TLSv1.2(ECDHE-RSA-AES128-GCM-SHA256)  x.509
  567  127.0.0.1:43546      b240c652-82df-48dd-b54e-3b8bbaef16c6
normal  in   no-security                           PLAIN
```

5.6.2. Viewing router addresses using qdstat

You can view the router addresses using **qdstat**.

Procedure

1. On the command line, run the following command to obtain the **podname** value needed in the following step:

```
oc get pods
```

2. Run the following command:

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b
127.0.0.1:7777 -a

Router Addresses
```

```
  class     addr                  phs  distrib       in-proc
local  remote  cntnr  in     out    thru  to-proc  from-proc

=================================================================
=================================================
  local     $_management_internal       closest       1        0
0       0       0       0       0     588       588
  link-in   $lwt                        linkBalanced  0        0
0       0       0       0       0     0         0
  link-out  $lwt                        linkBalanced  0        0
0       0       0       0       0     0         0
  mobile    $management                 0     closest       1        0
0       0       601     0       0     601       0
  local     $management                 closest       1        0
0       0       2,925   0       0     2,925     0
  local     qdhello                     flood         1        0
0       0       0       0       0     0         5,856
  local     qdrouter                    flood         1        0
0       0       0       0       0     0         0
  topo      qdrouter                    flood         1        0
0       0       0       0       0     0         196
  local     qdrouter.ma                 multicast     1        0
0       0       0       0       0     0         0
  topo      qdrouter.ma                 multicast     1        0
0       0       0       0       0     0         0
  local     temp.VTXOKyyWsq7OEei        balanced      0        1
0       0       0       0       0     0         0
  local     temp.k2RGQNPe6sDMvz4        balanced      0        1
0       0       0       3,511   0     0         3,511
  local     temp.xg+y8I_Tr4Y94LA        balanced      0        1
0       0       0       5       0     0         5
```

5.6.3. Viewing router links using qdstat

You can view the router links using **`qdstat`**.

Procedure

1. On the command line, run the following command to obtain the **podname** value needed in the following step:

```
oc get pods
```

2. On the command line, run the following command:

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b
127.0.0.1:7777 -l

Router Links
  type      dir  conn id  id  peer  class   addr
phs  cap  undel  unsett  del   presett  psdrop  acc   rej  rel  mod
admin    oper

=================================================================
=================================================================
```

```
============
  endpoint  in   3        8
250  0      0       3829  0        0       3829  0    0    0
enabled  up
  endpoint  out  3        9         local   temp.k2RGQNPe6sDMvz4
250  0      0       3829  3829     0       0     0    0    0
enabled  up
  endpoint  in   12       10
250  0      0       5     0        0       5     0    0    0
enabled  up
  endpoint  out  12       11        local   temp.xg+y8I_Tr4Y94LA
250  0      0       5     5        0       0     0    0    0
enabled  up
  endpoint  in   645      26        mobile  $management           0
50   0      0       1     0        0       1     0    0    0
enabled  up
  endpoint  out  645      27        local   temp.0BrHJ1O+fi6whyg
50   0      0       0     0        0       0     0    0    0
enabled  up
```

5.6.4. Viewing link routes using qdstat

You can view the link routes using **`qdstat`**.

Procedure

1. On the command line, run the following command to obtain the **podname** value needed in the following step:

```
oc get pods
```

2. On the command line, run the following command:

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b
127.0.0.1:7777 --linkroutes

Link Routes
  address  dir  distrib       status
  ====================================
  $lwt     in   linkBalanced  inactive
  $lwt     out  linkBalanced  inactive
```

CHAPTER 6. UNINSTALLING AMQ ONLINE

Procedure

1. Log in as a user with **`cluster-admin`** privileges:

```
oc login -u system:admin
```

2. Delete the rolebinding in the **`kube-system`** namespace:

```
oc delete rolebindings -l app=enmasse -n kube-system
```

3. Delete "cluster level" resources:

```
oc delete clusterrolebindings -l app=enmasse
oc delete crd -l app=enmasse
oc delete clusterroles -l app=enmasse
oc delete apiservices -l app=enmasse
oc delete oauthclients -l app=enmasse
```

4. (Optional) Delete the service catalog integration:

```
oc delete clusterservicebrokers -l app=enmasse
```

5. Delete the project where AMQ Online is deployed:

```
oc delete project amq-online-infra
```

APPENDIX A. REST API REFERENCE

A.1. ENMASSE REST API

A.1.1. Overview

This is the EnMasse API specification.

A.1.1.1. Version information

Version : 1.0.0

A.1.1.2. URI scheme

Schemes : HTTPS

A.1.1.3. Tags

- addresses : Operating on Addresses.
- addressplans : Operating on AddressPlans.
- addressspaceplans : Operating on AddressSpacePlans.
- addressspaces : Operate on AddressSpaces
- brokeredinfraconfigs : Operating on BrokeredInfraConfigs.
- messagingusers : Operating on MessagingUsers.
- standardinfraconfigs : Operating on StandardInfraConfigs.

A.1.1.4. External Docs

Description : Find out more about EnMasse
URL : http://enmasse.io

A.1.2. Paths

A.1.2.1. POST /apis/admin.enmasse.io/v1beta1/namespaces/{namespace}/addressspaceplans

A.1.2.1.1. Description

create an AddressSpacePlan

A.1.2.1.2. Parameters

Type	Name	Description	Schema

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.admin.v1beta1.AddressSpacePlan

A.1.2.1.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.admin.v1beta1.AddressSpacePlan
201	Created	io.enmasse.admin.v1beta1.AddressSpacePlan
401	Unauthorized	No Content

A.1.2.1.4. Consumes

- `application/json`

A.1.2.1.5. Produces

- `application/json`

A.1.2.1.6. Tags

- addressspaceplan
- admin
- enmasse_v1beta1

A.1.2.2. GET /apis/admin.enmasse.io/v1beta1/namespaces/{namespace}/addressspaceplans

A.1.2.2.1. Description

list objects of kind AddressSpacePlan

A.1.2.2.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Query	**labelSelector** *optional*	A selector to restrict the list of returned objects by their labels. Defaults to everything.	string

A.1.2.2.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.admin.v1beta1.AddressSpacePlanList
401	Unauthorized	No Content

A.1.2.2.4. Produces

- `application/json`

A.1.2.2.5. Tags

- addressspaceplan
- admin
- enmasse_v1beta1

A.1.2.3. GET /apis/admin.enmasse.io/v1beta1/namespaces/{namespace}/addressspaceplans/{name}

A.1.2.3.1. Description

read the specified AddressSpacePlan

A.1.2.3.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpacePlan to read.	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

A.1.2.3.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.admin.v1beta1.AddressSpacePlan
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.3.4. Consumes

- **`application/json`**

A.1.2.3.5. Produces

- **`application/json`**

A.1.2.3.6. Tags

- addressspaceplan
- admin
- enmasse_v1beta1

A.1.2.4. PUT /apis/admin.enmasse.io/v1beta1/namespaces/{namespace}/addressspaceplans/{name}

A.1.2.4.1. Description

replace the specified AddressSpacePlan

A.1.2.4.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpacePlan to replace.	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.admin.v1beta1.AddressSpacePlan

A.1.2.4.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.admin.v1beta1.AddressSpacePlan
201	Created	io.enmasse.admin.v1beta1.AddressSpacePlan
401	Unauthorized	No Content

A.1.2.4.4. Produces

- `application/json`

A.1.2.4.5. Tags

- addressspaceplan
- admin
- enmasse_v1beta1

A.1.2.5. DELETE /apis/admin.enmasse.io/v1beta1/namespaces/{namespace}/addressspaceplans/{name}

A.1.2.5.1. Description

delete an AddressSpacePlan

A.1.2.5.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpacePlan to delete.	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

A.1.2.5.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.5.4. Produces

- `application/json`

A.1.2.5.5. Tags

- addressspaceplan
- admin
- enmasse_v1beta1

A.1.2.6. POST /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses

A.1.2.6.1. Description

create an Address

A.1.2.6.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.v1beta1.Address

A.1.2.6.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.Address
201	Created	io.enmasse.v1beta1.Address

HTTP Code	Description	Schema
401	Unauthorized	No Content

A.1.2.6.4. Consumes

- `application/json`

A.1.2.6.5. Produces

- `application/json`

A.1.2.6.6. Tags

- addresses
- enmasse_v1beta1

A.1.2.7. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses

A.1.2.7.1. Description

list objects of kind Address

A.1.2.7.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Query	**labelSelector** *optional*	A selector to restrict the list of returned objects by their labels. Defaults to everything.	string

A.1.2.7.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressList
401	Unauthorized	No Content

A.1.2.7.4. Produces

- `application/json`

A.1.2.7.5. Tags

- addresses
- enmasse_v1beta1

A.1.2.8. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}

A.1.2.8.1. Description

read the specified Address

A.1.2.8.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of Address to read	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

A.1.2.8.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.Address
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.8.4. Consumes

- `application/json`

A.1.2.8.5. Produces

- `application/json`

A.1.2.8.6. Tags

- addresses
- enmasse_v1beta1

A.1.2.9. PUT /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}

A.1.2.9.1. Description

replace the specified Address

A.1.2.9.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of Address to replace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.v1beta1.Address

A.1.2.9.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.Address
201	Created	io.enmasse.v1beta1.Address
401	Unauthorized	No Content

A.1.2.9.4. Produces

- `application/json`

A.1.2.9.5. Tags

- addresses
- enmasse_v1beta1

A.1.2.10. DELETE /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}

A.1.2.10.1. Description

delete an Address

A.1.2.10.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of Address to delete	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

A.1.2.10.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.10.4. Produces

- `application/json`

A.1.2.10.5. Tags

- addresses
- enmasse_v1beta1

A.1.2.11. POST /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces

A.1.2.11.1. Description

create an AddressSpace

A.1.2.11.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.v1beta1.AddressSpace

A.1.2.11.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressSpace
201	Created	io.enmasse.v1beta1.AddressSpace
401	Unauthorized	No Content

A.1.2.11.4. Consumes

- `application/json`

A.1.2.11.5. Produces

- `application/json`

A.1.2.11.6. Tags

- addressspaces
- enmasse_v1beta1

A.1.2.12. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces

A.1.2.12.1. Description

list objects of kind AddressSpace

A.1.2.12.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Query	**labelSelector** *optional*	A selector to restrict the list of returned objects by their labels. Defaults to everything.	string

A.1.2.12.3. Responses

HTTP Code	Description	Schema

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressSpaceList
401	Unauthorized	No Content

A.1.2.12.4. Produces

- `application/json`

A.1.2.12.5. Tags

- addressspaces
- enmasse_v1beta1

A.1.2.13. POST /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{addressSpace}/addres

A.1.2.13.1. Description

create Addresses in an AddressSpace

A.1.2.13.2. Parameters

Type	Name	Description	Schema
Path	**addressSpace** *required*	Name of AddressSpace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*	AddressList object	io.enmasse.v1beta1.AddressList

A.1.2.13.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content

HTTP Code	Description	Schema
404	Not found	No Content

A.1.2.13.4. Consumes

- `application/json`

A.1.2.13.5. Produces

- `application/json`

A.1.2.13.6. Tags

- addressspace_addresses
- enmasse_v1beta1

A.1.2.14. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{addressSpace}/addres

A.1.2.14.1. Description

list objects of kind Address in AddressSpace

A.1.2.14.2. Parameters

Type	Name	Description	Schema
Path	**addressSpace** *required*	Name of AddressSpace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

A.1.2.14.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressList
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.14.4. Produces

- `application/json`

A.1.2.14.5. Tags

- addressspace_addresses
- enmasse_v1beta1

A.1.2.15. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{addressSpace}/addres

A.1.2.15.1. Description

read the specified Address in AddressSpace

A.1.2.15.2. Parameters

Type	Name	Description	Schema
Path	**address** *required*	Name of Address	string
Path	**addressSpace** *required*	Name of AddressSpace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

A.1.2.15.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.Address
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.15.4. Produces

- `application/json`

A.1.2.15.5. Tags

- addressspace_addresses

- enmasse_v1beta1

A.1.2.16. PUT /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{addressSpace}/addres

A.1.2.16.1. Description

replace Address in an AddressSpace

A.1.2.16.2. Parameters

Type	Name	Description	Schema
Path	**address** *required*	Name of address	string
Path	**addressSpace** *required*	Name of AddressSpace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*	Address object	io.enmasse.v1beta1.Address

A.1.2.16.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.Address
201	Created	io.enmasse.v1beta1.Address
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.16.4. Consumes

- `application/json`

A.1.2.16.5. Produces

- `application/json`

A.1.2.16.6. Tags

- addressspace_addresses
- enmasse_v1beta1

A.1.2.17. DELETE /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{addressSpace}/addres

A.1.2.17.1. Description

delete an Address in AddressSpace

A.1.2.17.2. Parameters

Type	Name	Description	Schema
Path	**address** *required*	Name of Address	string
Path	**addressSpace** *required*	Name of AddressSpace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

A.1.2.17.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.17.4. Produces

- **`application/json`**

A.1.2.17.5. Tags

- addressspace_addresses
- enmasse_v1beta1

A.1.2.18. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

A.1.2.18.1. Description

read the specified AddressSpace

A.1.2.18.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpace to read	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

A.1.2.18.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressSpace
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.18.4. Consumes

- `application/json`

A.1.2.18.5. Produces

- `application/json`

A.1.2.18.6. Tags

- addressspaces
- enmasse_v1beta1

A.1.2.19. PUT /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

A.1.2.19.1. Description

replace the specified AddressSpace

A.1.2.19.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpace to replace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.v1beta1.AddressSpace

A.1.2.19.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressSpace
201	Created	io.enmasse.v1beta1.AddressSpace
401	Unauthorized	No Content

A.1.2.19.4. Produces

- `application/json`

A.1.2.19.5. Tags

- addressspaces
- enmasse_v1beta1

A.1.2.20. DELETE /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

A.1.2.20.1. Description

delete an AddressSpace

A.1.2.20.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpace to delete	string

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

A.1.2.20.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.20.4. Produces

- `application/json`

A.1.2.20.5. Tags

- addressspaces
- enmasse_v1beta1

A.1.2.21. POST /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers

A.1.2.21.1. Description

create a MessagingUser

A.1.2.21.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.user.v1beta1.MessagingUser

A.1.2.21.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.user.v1beta1.MessagingUser
201	Created	io.enmasse.user.v1beta1.MessagingUser
401	Unauthorized	No Content

A.1.2.21.4. Consumes

- `application/json`

A.1.2.21.5. Produces

- `application/json`

A.1.2.21.6. Tags

- auth
- enmasse_v1beta1
- user

A.1.2.22. GET /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers

A.1.2.22.1. Description

list objects of kind MessagingUser

A.1.2.22.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Query	**labelSelector** *optional*	A selector to restrict the list of returned objects by their labels. Defaults to everything.	string

A.1.2.22.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.user.v1beta1.MessagingUserList
401	Unauthorized	No Content

A.1.2.22.4. Produces

- `application/json`

A.1.2.22.5. Tags

- auth
- enmasse_v1beta1
- user

A.1.2.23. GET /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

A.1.2.23.1. Description

read the specified MessagingUser

A.1.2.23.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of MessagingUser to read. Must include addressSpace and dot separator in the name (that is, 'myspace.user1').	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

A.1.2.23.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.user.v1beta1.MessagingUser

HTTP Code	Description	Schema
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.23.4. Consumes

- `application/json`

A.1.2.23.5. Produces

- `application/json`

A.1.2.23.6. Tags

- auth
- enmasse_v1beta1
- user

A.1.2.24. PUT /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

A.1.2.24.1. Description

replace the specified MessagingUser

A.1.2.24.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of MessagingUser to replace. Must include addressSpace and dot separator in the name (that is, 'myspace.user1').	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.user.v1beta1.MessagingUser

A.1.2.24.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.user.v1beta1.MessagingUser
201	Created	io.enmasse.user.v1beta1.MessagingUser
401	Unauthorized	No Content

A.1.2.24.4. Produces

- `application/json`

A.1.2.24.5. Tags

- auth
- enmasse_v1beta1
- user

A.1.2.25. DELETE /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

A.1.2.25.1. Description

delete a MessagingUser

A.1.2.25.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of MessagingUser to delete. Must include addressSpace and dot separator in the name (that is, 'myspace.user1').	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

A.1.2.25.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content
404	Not found	No Content

A.1.2.25.4. Produces

- `application/json`

A.1.2.25.5. Tags

- auth
- enmasse_v1beta1
- user

A.1.3. Definitions

A.1.3.1. ObjectMeta

ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.

Name	Schema
name *required*	string
namespace *optional*	string

A.1.3.2. Status

Status is a return value for calls that do not return other objects.

Name	Description	Schema
code *optional*	Suggested HTTP return code for this status, 0 if not set.	integer (int32)

A.1.3.3. io.enmasse.admin.v1beta1.AddressPlan

Name	Schema
addressType *required*	string
apiVersion *required*	enum (admin.enmasse.io/v1beta1)
displayName *required*	string
displayOrder *required*	integer
kind *required*	enum (AddressPlan)
longDescription *optional*	string
metadata *required*	ObjectMeta
requiredResources *required*	< requiredResources > array
shortDescription *required*	string
uuid *required*	string

requiredResources

Name	Schema
credit *required*	number
name *required*	string

A.1.3.4. io.enmasse.admin.v1beta1.AddressPlanList

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta1)
items *required*	< io.enmasse.admin.v1beta1.AddressPlan > array
kind *required*	enum (AddressPlanList)

A.1.3.5. io.enmasse.admin.v1beta1.AddressSpacePlan

Name	Schema
addressPlans *optional*	< string > array
addressSpaceType *required*	string
apiVersion *required*	enum (admin.enmasse.io/v1beta1)
displayName *required*	string
displayOrder *required*	integer
kind *required*	enum (AddressSpacePlan)
longDescription *optional*	string
metadata *required*	ObjectMeta
resources *required*	< resources > array
shortDescription *required*	string
uuid *required*	string

resources

Name	Schema
max *required*	number
name *required*	string

A.1.3.6. io.enmasse.admin.v1beta1.AddressSpacePlanList

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta1)
items *required*	< io.enmasse.admin.v1beta1.AddressSpacePlan > array
kind *required*	enum (AddressSpacePlanList)

A.1.3.7. io.enmasse.admin.v1beta1.BrokeredInfraConfig

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta1)
kind *required*	enum (BrokeredInfraConfig)
metadata *required*	ObjectMeta
spec *required*	io.enmasse.admin.v1beta1.BrokeredInfraConfigSpec

A.1.3.8. io.enmasse.admin.v1beta1.BrokeredInfraConfigList

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta1)

Name	Schema
items *required*	< io.enmasse.admin.v1beta1.BrokeredInfraConfig > array
kind *required*	enum (BrokeredInfraConfigList)

A.1.3.9. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpec

Name	Schema
admin *optional*	io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecAdmin
broker *optional*	io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecBroker
networkPolicy *optional*	networkPolicy
version *required*	string

networkPolicy

Name	Schema
egress *optional*	< io.k8s.api.networking.v1.NetworkPolicyEgressRule > array
ingress *optional*	< io.k8s.api.networking.v1.NetworkPolicyIngressRule > array

A.1.3.10. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecAdmin

Name	Schema
resources *optional*	resources

resources

Name	Schema
memory *optional*	string

A.1.3.11. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecBroker

Name	Schema
addressFullPolicy *optional*	enum (PAGE, BLOCK, FAIL)
resources *optional*	resources
storageClassName *optional*	string
updatePersistentVolumeClaim *optional*	boolean

resources

Name	Schema
memory *optional*	string
storage *optional*	string

A.1.3.12. io.enmasse.admin.v1beta1.StandardInfraConfig

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta1)
kind *required*	enum (StandardInfraConfig)
metadata *required*	ObjectMeta
spec *required*	io.enmasse.admin.v1beta1.StandardInfraConfigSpec

A.1.3.13. io.enmasse.admin.v1beta1.StandardInfraConfigList

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta1)
items *required*	< io.enmasse.admin.v1beta1.StandardInfraConfig > array
kind *required*	enum (StandardInfraConfigList)

A.1.3.14. io.enmasse.admin.v1beta1.StandardInfraConfigSpec

Name	Schema
admin *optional*	io.enmasse.admin.v1beta1.StandardInfraConfigSpecAdmin
broker *optional*	io.enmasse.admin.v1beta1.StandardInfraConfigSpecBroker
networkPolicy *optional*	networkPolicy
router *optional*	io.enmasse.admin.v1beta1.StandardInfraConfigSpecRouter
version *required*	string

networkPolicy

Name	Schema
egress *optional*	< io.k8s.api.networking.v1.NetworkPolicyEgressRule > array
ingress *optional*	< io.k8s.api.networking.v1.NetworkPolicyIngressRule > array

A.1.3.15. io.enmasse.admin.v1beta1.StandardInfraConfigSpecAdmin

Name	Schema
resources *optional*	resources

resources

Name	Schema
memory *optional*	string

A.1.3.16. io.enmasse.admin.v1beta1.StandardInfraConfigSpecBroker

Name	Schema
addressFullPolicy *optional*	enum (PAGE, BLOCK, FAIL)
resources *optional*	resources
storageClassName *optional*	string
updatePersistentVolumeClaim *optional*	boolean

resources

Name	Schema
memory *optional*	string
storage *optional*	string

A.1.3.17. io.enmasse.admin.v1beta1.StandardInfraConfigSpecRouter

Name	Schema
linkCapacity *optional*	integer

Name	Schema
minReplicas *optional*	integer
resources *optional*	resources

resources

Name	Schema
memory *optional*	string

A.1.3.18. io.enmasse.user.v1beta1.MessagingUser

Name	Schema
apiVersion *required*	enum (user.enmasse.io/v1beta1)
kind *required*	enum (MessagingUser)
metadata *required*	ObjectMeta
spec *required*	io.enmasse.user.v1beta1.UserSpec

A.1.3.19. io.enmasse.user.v1beta1.MessagingUserList

Name	Schema
apiVersion *required*	enum (user.enmasse.io/v1beta1)
items *required*	< io.enmasse.user.v1beta1.MessagingUser > array
kind *required*	enum (MessagingUserList)

A.1.3.20. io.enmasse.user.v1beta1.UserSpec

Name	Schema
authentication *optional*	authentication
authorization *optional*	< authorization > array
username *required*	string

authentication

Name	Description	Schema
federatedUserid *optional*	User id of the user to federate when 'federated' type is specified.	string
federatedUsername *optional*	User name of the user to federate when 'federated' type is specified.	string
password *optional*	Base64 encoded value of password when 'password' type is specified.	string
provider *optional*	Name of provider to use for federated identity when 'federated' type is specified.	string
type *required*		enum (password, federated)

authorization

Name	Schema
addresses *optional*	< string > array
operations *required*	< enum (send, receive, view, manage) > array

A.1.3.21. io.enmasse.v1beta1.Address

Name	Schema
apiVersion *required*	enum (enmasse.io/v1beta1)
kind *required*	enum (Address)
metadata *required*	ObjectMeta
spec *required*	io.enmasse.v1beta1.AddressSpec
status *optional*	io.enmasse.v1beta1.AddressStatus

A.1.3.22. io.enmasse.v1beta1.AddressList

Name	Description	Schema
apiVersion *required*	**Default** : `"enmasse.io/v1beta1"`	enum (enmasse.io/v1beta1)
items *required*		< io.enmasse.v1beta1.Address > array
kind *required*		enum (AddressList)

A.1.3.23. io.enmasse.v1beta1.AddressSpace

Name	Schema
apiVersion *required*	enum (enmasse.io/v1beta1)
kind *required*	enum (AddressSpace)
metadata *required*	ObjectMeta

Name	Schema
spec *required*	io.enmasse.v1beta1.AddressSpaceSpec
status *optional*	io.enmasse.v1beta1.AddressSpaceStatus

A.1.3.24. io.enmasse.v1beta1.AddressSpaceList

Name	Description	Schema
apiVersion *required*	**Default** : `"enmasse.io/v1beta1"`	enum (enmasse.io/v1beta1)
items *required*		< io.enmasse.v1beta1.AddressSpace > array
kind *required*		enum (AddressSpaceList)

A.1.3.25. io.enmasse.v1beta1.AddressSpaceSpec

Name	Schema
authenticationService *optional*	authenticationService
endpoints *optional*	< endpoints > array
networkPolicy *optional*	networkPolicy
plan *required*	string
type *required*	io.enmasse.v1beta1.AddressSpaceType

authenticationService

Name	Schema
details *optional*	object
type *optional*	string

endpoints

Name	Schema
cert *optional*	cert
expose *optional*	expose
name *optional*	string
service *optional*	string

cert

Name	Schema
provider *optional*	string
secretName *optional*	string
tlsCert *optional*	string
tlsKey *optional*	string

expose

Name	Schema
annotations *optional*	object

Name	Schema
loadBalancerPorts *optional*	< string > array
loadBalancerSourceRanges *optional*	< string > array
routeHost *optional*	string
routeServicePort *optional*	string
routeTlsTermination *optional*	string
type *optional*	enum (route, loadbalancer)

networkPolicy

Name	Schema
egress *optional*	< io.k8s.api.networking.v1.NetworkPolicyEgressRule > array
ingress *optional*	< io.k8s.api.networking.v1.NetworkPolicyIngressRule > array

A.1.3.26. io.enmasse.v1beta1.AddressSpaceStatus

Name	Schema
endpointStatuses *optional*	< endpointStatuses > array
isReady *optional*	boolean
messages *optional*	< string > array

endpointStatuses

Name	Schema
cert *optional*	string
externalHost *optional*	string
externalPorts *optional*	< externalPorts > array
name *optional*	string
serviceHost *optional*	string
servicePorts *optional*	< servicePorts > array

externalPorts

Name	Schema
name *optional*	string
port *optional*	integer

servicePorts

Name	Schema
name *optional*	string
port *optional*	integer

A.1.3.27. io.enmasse.v1beta1.AddressSpaceType

AddressSpaceType is the type of address space (standard, brokered). Each type supports different types of addresses and semantics for those types.

Type : enum (standard, brokered)

A.1.3.28. io.enmasse.v1beta1.AddressSpec

Name	Schema
address *required*	string
plan *required*	string
type *required*	io.enmasse.v1beta1.AddressType

A.1.3.29. io.enmasse.v1beta1.AddressStatus

Name	Schema
isReady *optional*	boolean
messages *optional*	< string > array
phase *optional*	enum (Pending, Configuring, Active, Failed, Terminating)

A.1.3.30. io.enmasse.v1beta1.AddressType

Type of address (queue, topic, …). Each address type support different kinds of messaging semantics.

Type : enum (queue, topic, anycast, multicast)

A.1.3.31. io.k8s.api.networking.v1.IPBlock

IPBlock describes a particular CIDR (Ex. "192.168.1.1/24") that is allowed to the pods matched by a NetworkPolicySpec's podSelector. The except entry describes CIDRs that should not be included within this rule.

Name	Description	Schema
cidr *required*	CIDR is a string representing the IP Block Valid examples are "192.168.1.1/24"	string
except *optional*	Except is a slice of CIDRs that should not be included within an IP Block Valid examples are "192.168.1.1/24" Except values will be rejected if they are outside the CIDR range	< string > array

A.1.3.32. io.k8s.api.networking.v1.NetworkPolicyEgressRule

NetworkPolicyEgressRule describes a particular set of traffic that is allowed out of pods matched by a NetworkPolicySpec's podSelector. The traffic must match both ports and to. This type is beta-level in 1.8

Name	Description	Schema
ports *optional*	List of destination ports for outgoing traffic. Each item in this list is combined using a logical OR. If this field is empty or missing, this rule matches all ports (traffic not restricted by port). If this field is present and contains at least one item, then this rule allows traffic only if the traffic matches at least one port in the list.	< io.k8s.api.networking.v1.NetworkPolicyPort > array
to *optional*	List of destinations for outgoing traffic of pods selected for this rule. Items in this list are combined using a logical OR operation. If this field is empty or missing, this rule matches all destinations (traffic not restricted by destination). If this field is present and contains at least one item, this rule allows traffic only if the traffic matches at least one item in the to list.	< io.k8s.api.networking.v1.NetworkPolicyPeer > array

A.1.3.33. io.k8s.api.networking.v1.NetworkPolicyIngressRule

NetworkPolicyIngressRule describes a particular set of traffic that is allowed to the pods matched by a NetworkPolicySpec's podSelector. The traffic must match both ports and from.

Name	Description	Schema
from *optional*	List of sources which should be able to access the pods selected for this rule. Items in this list are combined using a logical OR operation. If this field is empty or missing, this rule matches all sources (traffic not restricted by source). If this field is present and contains at least on item, this rule allows traffic only if the traffic matches at least one item in the from list.	< io.k8s.api.networking.v1.NetworkPolicyPeer > array
ports *optional*	List of ports which should be made accessible on the pods selected for this rule. Each item in this list is combined using a logical OR. If this field is empty or missing, this rule matches all ports (traffic not restricted by port). If this field is present and contains at least one item, then this rule allows traffic only if the traffic matches at least one port in the list.	< io.k8s.api.networking.v1.NetworkPolicyPort > array

A.1.3.34. io.k8s.api.networking.v1.NetworkPolicyPeer

NetworkPolicyPeer describes a peer to allow traffic from. Only certain combinations of fields are allowed

Name	Description	Schema
ipBlock *optional*	IPBlock defines policy on a particular IPBlock. If this field is set then neither of the other fields can be.	io.k8s.api.networking.v1.IPBlock

Name	Description	Schema
namespaceSelector *optional*	Selects Namespaces using cluster-scoped labels. This field follows standard label selector semantics; if present but empty, it selects all namespaces. If PodSelector is also set, then the NetworkPolicyPeer as a whole selects the Pods matching PodSelector in the Namespaces selected by NamespaceSelector. Otherwise it selects all Pods in the Namespaces selected by NamespaceSelector.	io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelector
podSelector *optional*	This is a label selector which selects Pods. This field follows standard label selector semantics; if present but empty, it selects all pods. If NamespaceSelector is also set, then the NetworkPolicyPeer as a whole selects the Pods matching PodSelector in the Namespaces selected by NamespaceSelector. Otherwise it selects the Pods matching PodSelector in the policy's own Namespace.	io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelector

A.1.3.35. io.k8s.api.networking.v1.NetworkPolicyPort

NetworkPolicyPort describes a port to allow traffic on

Name	Description	Schema
port *optional*	The port on the given protocol. This can either be a numerical or named port on a pod. If this field is not provided, this matches all port names and numbers.	io.k8s.apimachinery.pkg.util.intstr.IntOrString
protocol *optional*	The protocol (TCP or UDP) which traffic must match. If not specified, this field defaults to TCP.	string

A.1.3.36. io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelector

A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

Name	Description	Schema
matchExpressions *optional*	matchExpressions is a list of label selector requirements. The requirements are ANDed.	< io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelectorRequirement > array

Name	Description	Schema
matchLabels *optional*	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.	< string, string > map

A.1.3.37. io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelectorRequirement

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Name	Description	Schema
key *required*	key is the label key that the selector applies to.	string
operator *required*	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.	string
values *optional*	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.	< string > array

A.1.3.38. io.k8s.apimachinery.pkg.util.intstr.IntOrString

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

Type : string (int-or-string)

APPENDIX B. USING YOUR SUBSCRIPTION

AMQ Online is provided through a software subscription. To manage your subscriptions, access your account at the Red Hat Customer Portal.

Accessing your account

1. Go to access.redhat.com.
2. If you do not already have an account, create one.
3. Log in to your account.

Activating a subscription

1. Go to access.redhat.com.
2. Navigate to **My Subscriptions**.
3. Navigate to **Activate a subscription** and enter your 16-digit activation number.

Downloading zip and tar files

To access zip or tar files, use the Red Hat Customer Portal to find the relevant files for download. If you are using RPM packages, this step is not required.

1. Open a browser and log in to the Red Hat Customer Portal **Product Downloads** page at access.redhat.com/downloads.
2. Locate the **Red Hat AMQ Online** entries in the **JBOSS INTEGRATION AND AUTOMATION** category.
3. Select the desired AMQ Online product. The Software Downloads page opens.
4. Click the **Download** link for your component.

Registering your system for packages

To install RPM packages on Red Hat Enterprise Linux, your system must be registered. If you are using zip or tar files, this step is not required.

1. Go to access.redhat.com.
2. Navigate to **Registration Assistant**.
3. Select your OS version and continue to the next page.
4. Use the listed command in your system terminal to complete the registration.

To learn more see How to Register and Subscribe a System to the Red Hat Customer Portal.

Revised on 2019-03-11 19:09:50 UTC