

RABBIT-MQ

Vista sistemica

Claudio Biancalana

Fonti e riferimenti

- <https://www.rabbitmq.com/tutorials/amqp-concepts.html>
- Advanced Message Queuing Protocol AMQP: Upgrader's Guide – Gerald Blokdyk - 2017
- RabbitMQ in Depth – Gavin M. Roy – 2017
- RabbitMQ in Action: Distributed Messaging for Everyone – Alvaro Videla e Jason J. W. Williams - 2012

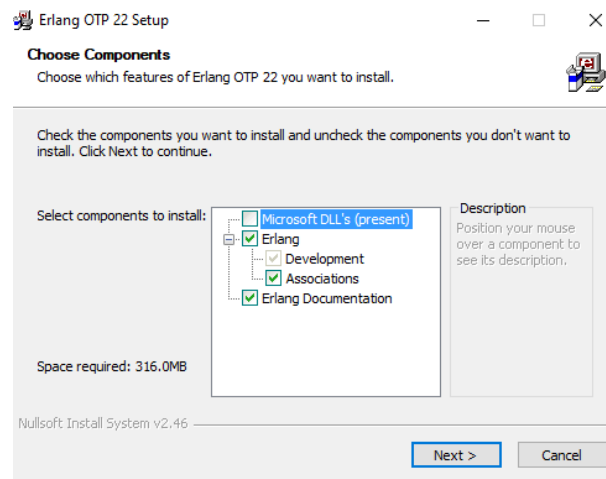
RabbitMQ

panoramica delle caratteristiche

- RabbitMQ comprende numerose caratteristiche che supportano il deployment la gestione e la configurazione delle singole istanze del server:
 - Supporta protocolli multipli (AMQP, STOMP, MQTT)
 - Funzionalità avanzate di routing
 - Supporto per molti linguaggi di programmazione lato client
 - Consegna affidabile
 - Clustering
 - Federazione
 - Alta disponibilità
 - Gestione e monitoraggio
 - Autenticazione e controllo di accesso
 - Architettura a plug-in

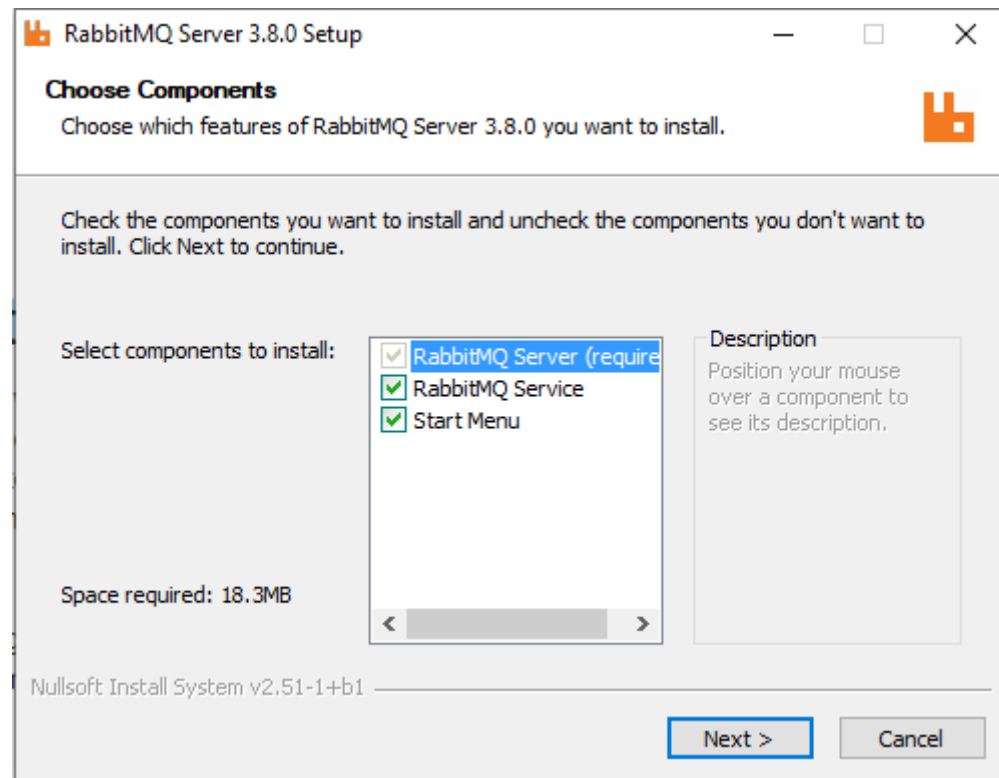
Installazione Windows

- Download dell'installer di RabbitMQ per windows:
 - <https://github.com/rabbitmq/rabbitmq-server/releases/download/v3.8.0/rabbitmq-server-3.8.0.exe>
- RabbitMQ funziona su macchine 32/64 bit e gira in ambiente Erlang.
- Pertanto prima è necessario installare il runtime di Erlang:
 - 64 bit: http://erlang.org/download/otp_win64_22.1.exe
 - 32 bit: http://erlang.org/download/otp_win32_22.1.exe



Installazione

- A seguito dell'installazione di Erlang sarà possibile installare i binary di RabbitMQ



Comandi

- Nell'installazione di RabbitMQ sono contenuti i comandi necessari per lo start/stop e status (sotto la cartella sbin)
 - `rabbitmqctl.bat status`
 - `rabbitmqctl.bat stop`
 - `rabbitmqctl.bat start_app`
- Per attivare la web console è necessario abilitare il relativo plugin
 - `rabbitmq-plugins.bat enable rabbitmq_management`
- E riavviare il server
 - `rabbitmq-server.bat start_app`

Installazione Linux (Ubuntu 18.04)

Installazione Erlang

- `echo "deb https://packages.erlang-solutions.com/ubuntu bionic contrib" | sudo tee /etc/apt/sources.list.d/rabbitmq.list`
- `wget -O- https://packages.erlang-solutions.com/ubuntu/erlang_solutions.asc | sudo apt-key add -`
- `sudo apt update`
- `sudo apt -y install erlang`

Installazione Linux (Ubuntu 18.04)

Installazione rabbitmq

- `wget -O- https://dl.bintray.com/rabbitmq/Keys/rabbitmq-release-signing-key.asc | sudo apt-key add -`
- `wget -O- https://www.rabbitmq.com/rabbitmq-release-signing-key.asc | sudo apt-key add`
- `echo "deb https://dl.bintray.com/rabbitmq/debian $(lsb_release -sc) main" | sudo tee /etc/apt/sources.list.d/rabbitmq.list`
- `sudo apt update`
- `sudo apt -y install rabbitmq-server`

Attivazione interfaccia di amministrazione web

- `sudo rabbitmq-plugins enable rabbitmq_management`

Porte utilizzate

(dalla documentazione di RabbitMQ)

- **4369**: epmd, a peer discovery service used by RabbitMQ nodes and CLI tools
- **5672, 5671**: used by AMQP 0-9-1 and 1.0 clients without and with TLS
- **25672**: used for inter-node and CLI tools communication (Erlang distribution server port) and is allocated from a dynamic range (limited to a single port by default, computed as AMQP port + 20000). Unless external connections on these ports are really necessary (e.g. the cluster uses federation or CLI tools are used on machines outside the subnet), these ports should not be publicly exposed. See networking guide for details.
- **35672-35682**: used by CLI tools (Erlang distribution client ports) for communication with nodes and is allocated from a dynamic range (computed as server distribution port + 10000 through server distribution port + 10010). See networking guide for details.
- **15672**: HTTP API clients, management UI and rabbitmqadmin (only if the management plugin is enabled)
- **61613, 61614**: STOMP clients without and with TLS (only if the STOMP plugin is enabled)
- **1883, 8883**: (MQTT clients without and with TLS, if the MQTT plugin is enabled)
- **15674**: STOMP-over-WebSockets clients (only if the Web STOMP plugin is enabled)
- **15675**: MQTT-over-WebSockets clients (only if the Web MQTT plugin is enabled)
- **15692**: Prometheus metrics (only if the Prometheus plugin is enabled)

Web Console

- URL: `http://localhost:15672/`
- User: guest - Password: guest

The screenshot displays the RabbitMQ Web Console interface. At the top, the RabbitMQ logo is shown alongside the version number 3.8.0 and the Erlang version 22.1. Below the logo is a navigation bar with tabs for Overview, Connections, Channels, Exchanges, Queues, and Admin. The Overview tab is selected, showing a summary of the system's status. The Overview section includes a 'Totals' section with metrics for Queued messages, Currently idle, Message rates, and Global counts. Below this, a row of buttons displays the current counts: Connections: 0, Channels: 0, Exchanges: 7, Queues: 0, and Consumers: 0. The 'Nodes' section is expanded, showing a table with details for the node 'rabbit@LAPTOP-PD8QJBC7'. The table includes columns for Name, File descriptors, Socket descriptors, Erlang processes, Memory, Disk space, Uptime, Info, and Reset stats. The node is currently running with 0 file descriptors, 0 socket descriptors, 444 Erlang processes, 103MiB of memory, and 3.9GiB of disk space. The uptime is 8m 52s. The Info column shows links for basic, disc, 1, and rss. The Reset stats column has buttons for 'This node' and 'All nodes'. Below the table, there are links for Churn statistics, Ports and contexts, Export definitions, and Import definitions. At the bottom, a footer contains links for HTTP API, Server Docs, Tutorials, Community Support, Community Slack, Commercial Support, Plugins, GitHub, and Changelog.

RabbitMQ 3.8.0 Erlang 22.1

Overview Connections Channels Exchanges Queues Admin

Overview

▼ Totals

Queued messages [last minute](#) ?

Currently idle

Message rates [last minute](#) ?

Currently idle

Global counts ?

Connections: 0 Channels: 0 Exchanges: 7 Queues: 0 Consumers: 0

▼ Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats	+/-
rabbit@LAPTOP-PD8QJBC7	0 65536 available	0 58890 available	444 1048576 available	103MiB 6.4GiB high watermark	3.9GiB 48MiB low watermark	8m 52s	basic disc 1 rss	This node All nodes	

► Churn statistics

► Ports and contexts

► Export definitions

► Import definitions

HTTP API Server Docs Tutorials Community Support Community Slack Commercial Support Plugins GitHub Changelog

Variabili di ambiente per UNIX

Nome	Location
RABBITMQ_BASE	Variabile non usata per UNIX
RABBITMQ_CONFIG_FILE	\${install_prefix}/etc/rabbitmq/rabbitmq
RABBITMQ_LOGS	\$RABBITMQ_LOG_BASE/\$RABBITMQ_NODENAME.log
RABBITMQ_LOG_BASE	\${install_prefix}/var/log/rabbitmq
RABBITMQ_MNESIA_BASE	\${install_prefix}/var/lib/rabbitmq/mnesia
RABBITMQ_MNESIA_DIR	\$RABBITMQ_MNESIA_BASE/\$RABBITMQ_NODENAME
RABBITMQ_PLUGINS_DIR	\$RABBITMQ_HOME/plugins
RABBITMQ_SASL_LOG	\$RABBITMQ_LOG_BASE/\$RABBITMQ_NODENAMEsasl.log

Variabili di ambiente per Windows

Nome	Location
RABBITMQ_BASE	%APPDATA%\RabbitMQ
RABBITMQ_CONFIG_FILE	%RABBITMQ_BASE%\rabbitmq
RABBITMQ_LOGS	%RABBITMQ_LOG_BASE%\%RABBITMQ_NODENAME%.log
RABBITMQ_LOG_BASE	%RABBITMQ_LOG_BASE%\log
RABBITMQ_MNESIA_BASE	%RABBITMQ_BASE%\db
RABBITMQ_MNESIA_DIR	%RABBITMQ_MNESIA_BASE%\%RABBITMQ_NODENAME%
RABBITMQ_PLUGINS_DIR	%RABBITMQ_BASE%\plugins
RABBITMQ_SASL_LOG	%RABBITMQ_LOG_BASE%\%RABBITMQ_NODENAME%-sasl.log

Amministrare RabbitMQ

La potenza è nulla senza il controllo

Amministrare RabbitMQ

- start_app/stop dell'istanza
- Aggiungere/rimuovere/modificare/ispezionare utenti, virtual host, exchange, code e bind
- Backup e restore del database
- Settare un database differente per i messaggi persistenti
- Sicurezza
- Ispezionare log per errori
- Ottimizzare l'utilizzo delle risorse, il tuning e monitoraggio del broker
- Configurare le variabili di ambiente, parametri di configurazioni e policy
- Utilizzo delle API REST per la gestione del broker (esposte con un plugin di gestione)

Configurazione

- La configurazione di RabbitMQ è abbastanza importante per assicurare performance, alta disponibilità e scalabilità.
 - Variabili d'ambiente
 - File di configurazione
 - Parametri a Runtime
- Verificare dal log, la locazione dei file di configurazione:

```
2019-10-03 11:15:58.464 [info] <0.264.0>
Starting RabbitMQ 3.8.0 on Erlang 22.1
Copyright (C) 2007-2019 Pivotal Software, Inc.
Licensed under the MPL. See https://www.rabbitmq.com/
2019-10-03 11:15:58.465 [info] <0.264.0>
node           : rabbit@LAPTOP-PD8QJBC7
home dir       : C:\WINDOWS\system32\config\systemprofile
config file(s) : c:/Users/Claudio/AppData/Roaming/RabbitMQ/advanced.config
cookie hash    : 1bi6MrrYcQaaKMY98eHtA==
log(s)         : C:/Users/Claudio/AppData/Roaming/RabbitMQ/log/rabbit@LAPTOP-PD8QJBC7.log
                : C:/Users/Claudio/AppData/Roaming/RabbitMQ/log/rabbit@LAPTOP-PD8QJBC7_upgrade.log
database dir   : c:/Users/Claudio/AppData/Roaming/RabbitMQ/db/rabbit@LAPTOP-PD8QJBC7-mnesia
```

Amministrare i componenti RabbitMQ

- I vari componenti di RabbitMQ possono essere gestiti nelle seguenti modalità:
 - Dall'interfaccia web del plugin di gestione di RabbitMQ
 - A linea di comando con gli script di rabbitmqctl (dentro la cartella sbin)
 - Dalle API REST del plugin di gestione di RabbitMQ

Aggiungere utenti

- Per aggiungere utenti non amministrativi, senza associazione a vhost:
 - `rabbitmqctl add_user $USER $PASSWORD`
- Promuoverlo ad amministratore
 - `rabbitmqctl set_user_tags $USER administrator`
- Esempio:
 - `rabbitmqctl add_user claudio claudio2019`
 - `rabbitmqctl set_user_tags claudio administrator`

Users

▼ All users

Filter: ☐ Regex ?

Name	Tags	Can access virtual hosts	Has password
claudio	administrator	No access	•
guest	administrator	/	•

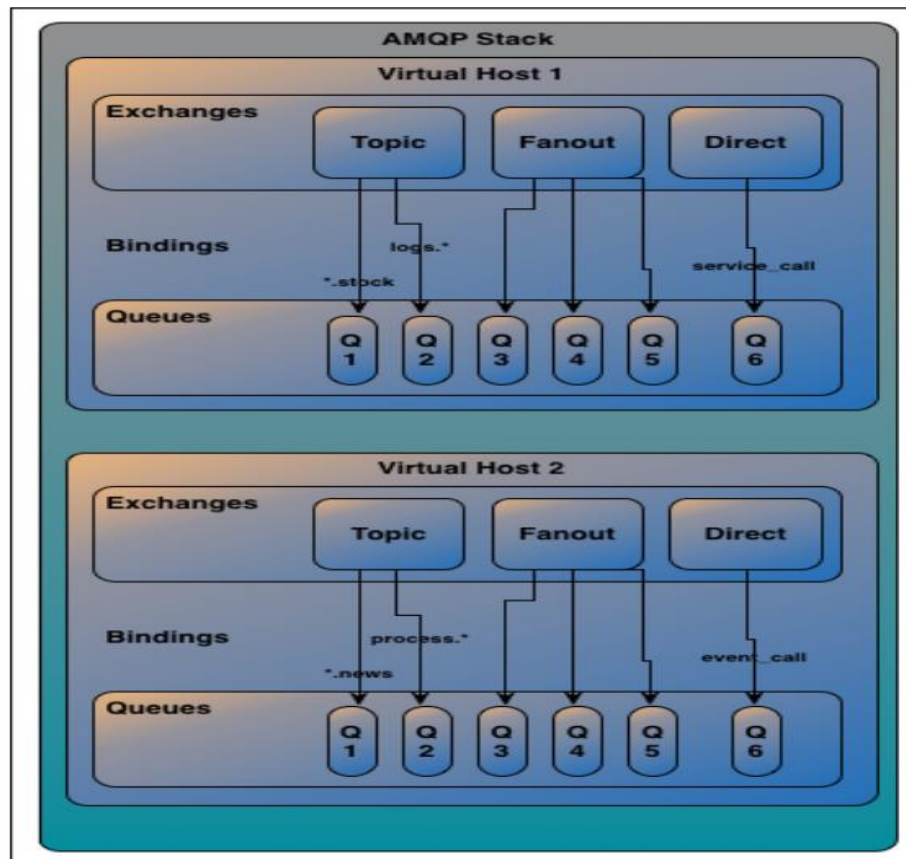
?

Gestire utenti

- Lista
 - `rabbitmqctl list_users`
- Cambio password
 - `rabbitmqctl change_password $USER $PASSWORD`
- Cancellazione utenza
 - `rabbitmqctl delete_user $USER`

Gestire vhosts (1)

- I virtual host (vhosts) sono usati per separare logicamente l'istanza di broker in domini multipli, ognuno dei quali ha un insieme di exchange, code e binding.



Gestire vhosts (2)

- Aggiungere vhost:
 - rabbitmqctl.bat add_vhost chat
 - rabbitmqctl.bat add_vhost events

Virtual Hosts

▼ All virtual hosts

Filter: ☐ Regex ?

Overview			Messages			Network		Message rates		+/-
Name	Users ?	State	Ready	Unacked	Total	From client	To client	publish	deliver / get	
/	guest	■ running	NaN	NaN	NaN					
chat	No users	■ running	NaN	NaN	NaN					
events	No users	■ running	NaN	NaN	NaN					

► Add a new virtual host

Gestire vhost (3)

- Lista
 - `rabbitmqctl.bat list_vhosts`
- Cancellazione
 - `rabbitmqctl.bat delete_vhost events`

E' possibile gestire i virtual host anche da interfaccia web.

Gestire i permessi

- Una volta creati utenti e virtual host, possiamo creare le relative associazioni mediante l'utilizzo dei permessi:
 - L'esempio seguente mostra come configurare i permessi di lettura e scrittura a tutte le risorse dell'vhost 'chat' per l'utente 'claudio'
 - `rabbitmqctl.bat set_permissions -p chat claudio ".*" ".*" ".*"`
- Lista dei permessi:
 - `rabbitmqctl.bat list_permissions -p chat`

```
Listing permissions for vhost "chat" ...
user      configure      write    read
claudio   .*                  .*      .*
```

- Rimuovere permessi:
 - `rabbitmqctl.bat clear_permissions -p chat claudio`

Gestire exchange

- E' possibile gestire gli exchange da interfaccia web o da linea di comando mediante rabbitmqadmin.py
 - rabbitmqadmin.py declare exchange name=logs type=fanout
- Cancellare l'exchange
 - rabbitmqadmin.py -V chat delete exchange name=logs
- Listare gli exchange nel virtual host 'chat'
 - rabbitmqadmin.py -V chat list exchanges
- Listare gli exchange all'interno del default vhost
 - rabbitmqadmin.py list exchanges

Gestire code

- E' possibile gestire gli le code da interfaccia web o da linea di comando mediante rabbitmqadmin.py
 - rabbitmqadmin.py declare queue name=error_logs durable=false
- Cancellare la coda
 - rabbitmqadmin.py -V chat delete queue name=error_logs
- Listare le code all'interno del dominio di default
 - rabbitmqadmin.py list queues

Gestire binding

- E' possibile gestire i binding da interfaccia web o da linea di comando mediante rabbitmqadmin.py
- Esempio
 - rabbitmqadmin.py declare binding source=logs destination=error_logs

Gestire le policy politica dead-letter

- Le policy permettono di definire (e cambiare!) alcune proprietà degli exchange e delle code a runtime.
- La policy deve essere unica per exchange/coda, all'interno possono quindi esserci settaggi multipli.
- Consideriamo i seguenti scenari:
 - Decidiamo di limitare la capacità di una coda; se viene ecceduta i messaggi possono essere 'droppati' oppure rediretti in un exchange alternativo (dead-lettered).
 - Decidiamo di settare un limite di tempo di sosta di un messaggio all'interno di una coda, se viene ecceduto il limite di tempo si può scegliere se scartare i messaggi o inserirli in dead-lettered.
 - Vogliamo definire un exchange 'dead-letter' che riceve messaggi 'dead-letter' da una o più code

Gestire policy

- Per limitare la capacità delle coda error_logs nel default host a 200.000 bytes, possiamo applicare la seguente policy
 - rabbitmqadmin.py declare policy name=max-queue-len pattern=error_logs definition="{\"max-length-bytes\":200000}\" apply-to=queues
- Per limitare la capacità della coda error_logs nel default host ad un numero di messaggi massimo
 - rabbitmqadmin.py declare policy name=max-queue-len pattern=error_logs definition="{\"max-length\":200000}\" apply-to=queues
- Per cancellare policy:
 - rabbitmqadmin.py delete policy name=max-queue-len

Gestire il database di RabbitMQ

- Il database di riferimento per RabbitMQ è Mnesia, nel quale vengono memorizzate numerose informazioni relative al funzionamento del broker.
- Per il backup sono necessarie i seguenti passaggi
 1. Stop del broker
 2. Copia ed archivio della cartella di Mnesia
 3. Restart del broker
- La procedura di restore è intuibile 😊
- E' necessario considerare il fatto che se un messaggio non è configurato in modo da essere persistente, non sarà possibile ripristinare la situazione con la procedura di restore.
- Per rendere il messaggio persistente, è necessario configurare in modalità '**durevole**' l'exchange e la coda, inoltre il messaggio deve essere marcato come **persistente**

Utilità a linea di comando

- Per gestire al meglio l'operatività a linea di comando è opportuo avvalersi del comando
 - `rabbitmqadmin.py`
- Per poterne usufruire è necessario avere un interprete python
- Il file è reperibile qui: <http://localhost:15672/cli/>
- Salvarlo e inserirlo nella cartella sbin rinominandolo `rabbitmqadmin.py`
- Export metadata: `rabbitmqadmin.py export broker.json`
- Import metadata: `rabbitmqadmin.py import broker.json`

Installare plugin

- Come è accaduto per le funzionalità di amministrazione, è possibile installare altri plugin che aggiungono feature a RabbitMQ.
- Procedura installazione:
 - Copiare il file del plugin, con estensione ez (Erlang ZIP file) all'interno della cartella plugin
 - Listare ed abilitare il plugin coi seguenti comandi:
 - `rabbitmq-plugin list`
 - `rabbitmq-plugins enable $NOME_PLUGIN`
- Procedura disinstallazione:
 - `rabbitmq-plugins disable $NOME_PLUGIN`
 - Rimozione del file ez del plugin da disinstallare

Configurare le istanze di RabbitMQ

- RabbitMQ può essere configurato in diverse modalità:
 - Settando le variabili di ambiente
 - Modificando i file di configurazione
 - Definendo i parametri di runtime e le policy che possono essere modificate a runtime

Variabili di ambiente

- Esempi
 - **RABBITMQ_NODE_IP_ADDRESS**: l'indirizzo IP dell'interfaccia di rete con la quale deve essere legato il broker. E' utile per avere più broker con indirizzi diversi nella stessa macchina. Vuoto significa che si lega a tutti gli indirizzi.
 - **RABBITMQ_NODE_PORT**: la porta TCP dal quale il broker ascolta
 - **RABBITMQ_NODENAME**: il nome dell'istanza del broker.

API

- RabbitMQ può essere gestito con chiamate ad API REST
- L'interfaccia delle API è esposta da questa radice:
 - `http://localhost:15672/api/`
- Esempio:
 - `http://localhost:15672/api/vhosts`
 - Fornisce un JSON esplicativo dei virtual host configurati nel broker (è necessaria autenticazione basic-auth come administrator!)

```
:: Windows
C:\> curl -i -u guest:guest http://localhost:15672/api/vhosts

# Unix
$ curl -i -u guest:guest http://localhost:15672/api/vhosts

HTTP/1.1 200 OK
cache-control: no-cache
content-length: 196
content-security-policy: default-src 'self'
content-type: application/json
date: Mon, 02 Sep 2019 07:51:49 GMT
server: Cowboy
vary: accept, accept-encoding, origin

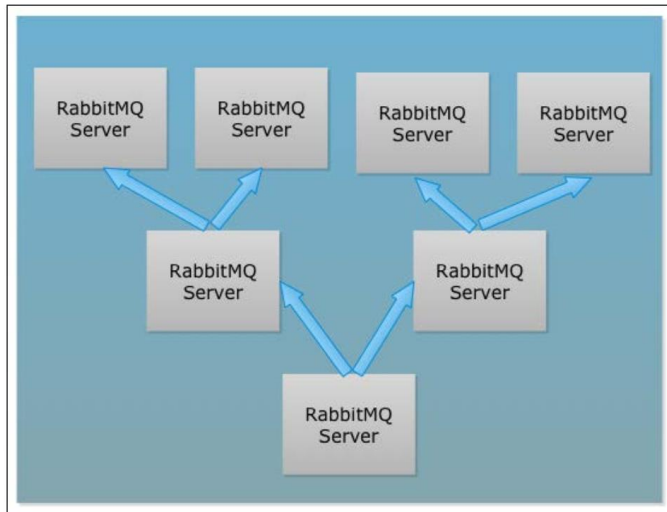
[{"cluster_state":{"rabbit@localhost":"running"},"description":"Default virtual host" ... (remainder elided)]
```

Aggiornare RabbitMQ

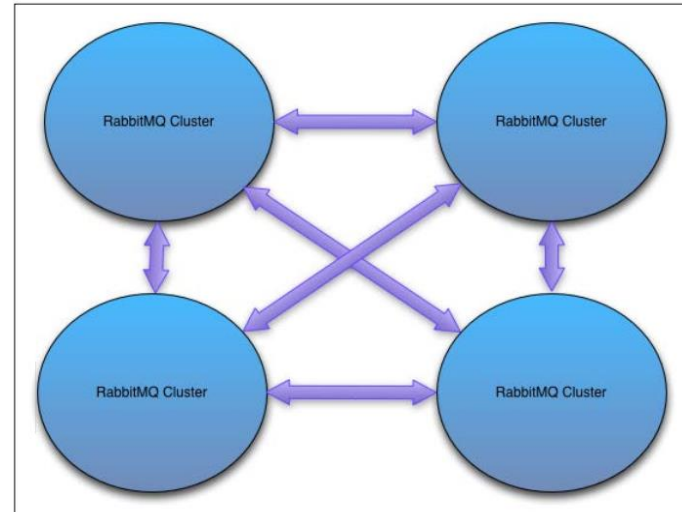
- L'aggiornamento di RabbitMQ coinvolge 2 oggetti:
 - L'aggiornamento dell'installazione di Erlang
 - L'aggiornamento dell'installazione del broker
- Tipicamente un aggiornamento del broker, preserva dati e configurazioni... In ogni caso è buona norma procedere con un full-backup prima di procedere ad un aggiornamento!
- Se l'ambiente è clusterizzato, è importante fermare tutti nodi del cluster ed usare la stessa versione di RabbitMQ per tutti gli aggiornamenti di tutti i nodi.

Alta disponibilità

- Alla luce di risolvere problemi di scalabilità e affidabilità, RabbitMQ fornisce la possibilità di configurare più nodi.
 - Una modalità è la **federazione**, che ha come scopo semplicemente la trasmissione dei messaggi fra broker.
 - Un'altra modalità è il **clustering**, che ha come scopo il coordinamento di più nodi.



Federation



Clustering

Federazione


- La federazione è uno dei modi più potenti per gestire molti messaggi usando RabbitMQ.
- Lo scopo principale della federazione è trasmettere messaggi fra i broker senza la necessità di clustering.
- Motivazioni:
 - Basso accoppiamento
 - Scalabilità
 - Specificità (un broker può contenere componenti federati o locali)
- Il plugin della federazione è disponibile come standard all'interno dell'installazione di RabbitMQ Server.
- Si può abilitare col seguente comando:

```
(sudo) rabbitmq-plugins enable rabbitmq_federation
```

Federazione

- Inoltre è possibile usare il plugin di gestione, attivandolo col seguente comando:
 - `rabbitmq-plugins enable rabbitmq_federation_management`
- La configurazione è memorizzata nel database RabbitMQ.
- Per la federazione serve definire tre parametri:
 - **Upstream**: definisce la connessione ad un altro RabbitMQ
 - **Upstream sets**: definisce i gruppi di upstream
 - **Policies**: insieme di regole per la federazione

Federazione in RabbitMQ

 3.8.0 Erlang 22.1

Refreshed 2019-10-07 17:09:49 Refresh every 5 seconds Virtual host All Cluster rabbit@LAPTOP-PD8QJBC7 User claudio Log out

OverviewConnectionsChannelsExchangesQueuesAdmin

Federation Upstreams

Upstreams

... no upstreams ...

Add a new upstream

General parameters

Virtual host: /

Name: *

URI: ? *

Prefetch count: ?

Reconnect delay: ? s

Acknowledgement Mode: ? On confirm

Trust User-ID: ? No

Federated exchanges parameters

Exchange: ?

Max hops: ?

Expires: ? ms

Message TTL: ? ms

HA Policy: ?

Federated queues parameter

Queue: ?

Consumer tag: ?

Add upstream

Users

Virtual Hosts

Feature Flags

Policies

Limits

Cluster

Federation Status

Federation Upstreams

Clustering in RabbitMQ

- Il clustering è la soluzione principale per la gestione delle richieste dei client sul server.
- Con il clustering viene effettuata una replica di tutti i dati / stati attraverso tutti i nodi al fine di sostenere affidabilità e scalabilità.
- La struttura generale dei cluster viene modificata in modo dinamico, in base all'aggiunta o alla rimozione di eventuali cluster dai sistemi.
- Inoltre, RabbitMQ tollera l'errore di ciascun nodo.
- Esistono 2 tipi di nodo:
 - RAM-Node: memorizza il suo stato in memoria.
 - Disk-Node: memorizza il suo stato su memoria e disco.

Creazione del cluster

- Copiare la chiave della cache erlang su tutti i nodi:
 - **Ubuntu: /var/lib/rabbitmq/.erlang.cookie**
- Per ogni nodo, dal 2 in poi, lanciare il comando:
 - **instance-2\$ rabbitmqctl stop_app**
 - **instance-2\$ rabbitmqctl join_cluster rabbit@instance-1**
 - **instance-2\$ rabbitmqctl start_app**
- dalla console rileviamo lo stato dei singoli nodi del cluster:
 - **rabbitmq1\$ rabbitmqctl cluster_status**
 - **rabbitmq2\$ rabbitmqctl cluster_status**
 - **rabbitmq3\$ rabbitmqctl cluster_status**
 - **rabbitmq4\$ rabbitmqctl cluster_status**

Cluster da interfaccia web



Overview

[Connections](#)[Channels](#)[Exchanges](#)[Queues](#)[Admin](#)

Overview

▼ Totals

Queued messages [last minute](#) [?](#)

Currently idle

Message rates [last minute](#) [?](#)

Currently idle

Global counts [?](#)

Connections: 0 Channels: 0 Exchanges: 7 Queues: 0 Consumers: 0

▼ Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats	+/-
rabbit@instance-1	40 1024 available	0 829 available	426 1048576 available	78MiB 1.4GiB high watermark	7.7GiB 48MiB low watermark	3h 32m	basic disc 2 rss	This node All nodes	
rabbit@instance-2	40 32768 available	0 29399 available	426 1048576 available	78MiB 1.4GiB high watermark	7.9GiB 48MiB low watermark	21m 20s	basic disc 2 rss	This node All nodes	
rabbit@instance-3	41 32768 available	0 29399 available	431 1048576 available	81MiB 1.4GiB high watermark	7.9GiB 48MiB low watermark	9m 7s	basic disc 2 rss	This node All nodes	
rabbit@instance-4	40 32768 available	0 29399 available	428 1048576 available	77MiB 1.4GiB high watermark	7.9GiB 48MiB low watermark	13m 15s	basic disc 2 rss	This node All nodes	

Cambiare tipo di nodo

- E' possibile cambiare il tipo di nodo (disco/RAM) del cluster.
 - Posizionarsi sul nodo da cambiare (esempio nodo 3)
 - Stoppare il server:
 - `rabbitmqctl stop_app`
 - Cambiare il tipo
 - `rabbitmqctl change_cluster_node_type ram`

```
root@instance-3:~# rabbitmqctl stop_app
Stopping rabbit application on node rabbit@instance-3 ...
root@instance-3:~# rabbitmqctl change_cluster_node_type ram
Turning rabbit@instance-3 into a ram node
root@instance-3:~# rabbitmqctl start_app
```

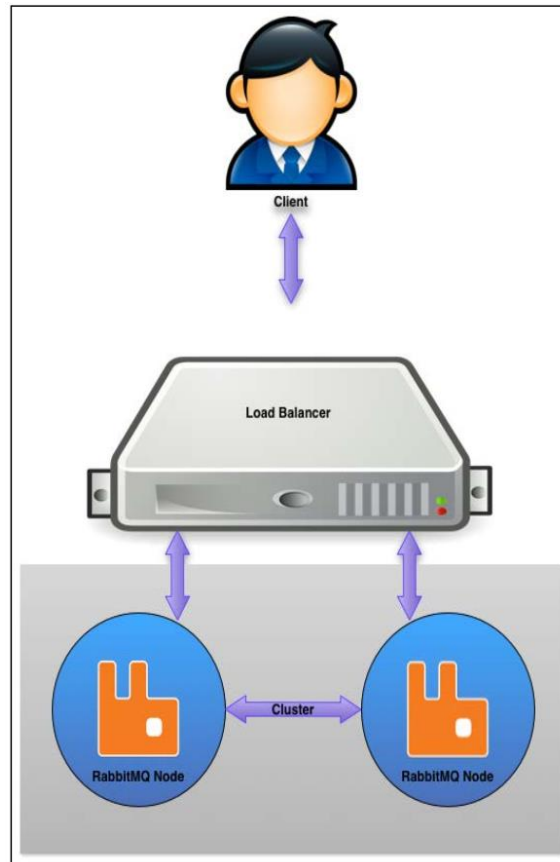
Rimuovere un nodo dal cluster

- A volte è importante avere il controllo del cluster sui singoli nodi.
- Per ragioni sistemiche potrebbe essere necessario rimuovere un nodo dal cluster
- Per farlo è necessario posizionarsi sul nodo, fermare l'app e resettare la configurazione:
 - `rabbitmqctl stop_app`
 - `rabbitmqctl reset`
 - `rabbitmqctl start_app`
- In alternativa è possibile rimuovere un nodo (ad esempio: instance-x) da 'remoto'
 - `rabbitmqctl forget_cluster_node rabbit@instance-x`

RabbitMQ

Alta affidabilità + prestazioni

- E' possibile adottare soluzione in alta affidabilità con cluster e bilanciatore di carico... vediamo come.



Aggiungere un bilanciatore SW (nell'esempio HAProxy)

- Installare HAProxy
 - `sudo apt install -y haproxy`
- Editare il file di configurazione
 - `sudo vim /etc/haproxy/haproxy.cfg`

```
global
    daemon

defaults
    mode tcp
    maxconn 10000
    timeout connect 5s
    timeout client 100s
    timeout server 100s

listen ha-proxy
    bind *:5672
    mode tcp
    balance roundrobin
    server instance-4 <IP-1>:5672 check
    server instance-1 <IP-2>:5672 check
    server instance-2 <IP-3>:5672 check
    server instance-3 <IP-4>:5672 check

listen stats
    bind *:80
    mode http
    stats enable
    stats uri /haproxy?stats
```

HAProxy

Console web di monitoraggio

- Per controllare la situazione posizionarsi sul seguente url
 - `http://<IP-HAProxy>/haproxy?stats`

HAProxy version 1.8.8-1ubuntu0.4, released 2019/01/24

Statistics Report for pid 2497

> General process information

pid = 2497 (process #1, nbproc = 1, nbthread = 1)
 uptime = 0d 0h00m22s
 system limits: memmax = unlimited; ulimit-n = 4035
 maxsock = 4035; maxconn = 2000; maxpipes = 0
 current conns = 2; current pipes = 0/0; conn rate = 2/sec
 Running tasks: 1/10; idle = 100 %

■ active UP ■ backup UP
■ active UP, going down ■ backup UP, going down
■ active DOWN, going up ■ backup DOWN, going up
■ active or backup DOWN ■ not checked
■ active or backup DOWN for maintenance (MAINT)
■ active or backup SOFT STOPPED for maintenance

Display option:

- Scope :
- [Hide 'DOWN' servers](#)
- [Refresh now](#)
- [CSV export](#)

External resources:

- [Primary site](#)
- [Updates \(v1.8\)](#)
- [Online manual](#)

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

http_front

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend				2	2	-	2	2	2 000	2			0	0	0	0	0					OPEN									

http_back

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
instance-1	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	22s UP	L4OK in 0ms	1	Y	-	0	0	0s	-
instance-2	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	22s UP	L4OK in 0ms	1	Y	-	0	0	0s	-
instance-3	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	22s UP	L4OK in 0ms	1	Y	-	0	0	0s	-
instance-4	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	22s UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0		0	0		0	0	200	0	0	?	0	0	0	0		0	0	0	0	22s UP		4	4	0		0	0s	

Abilitare e disabilitare Plugin

- Per la gestione dei plugin si utilizza il comando `rabbitmq-plugins` grazie al quale è possibile listare, abilitare e disabilitare i plugin.
 - `rabbitmq-plugins list`
 - `rabbitmq-plugins enable <plugin-name>`
 - `rabbitmq-plugins disable <plugin-name>`

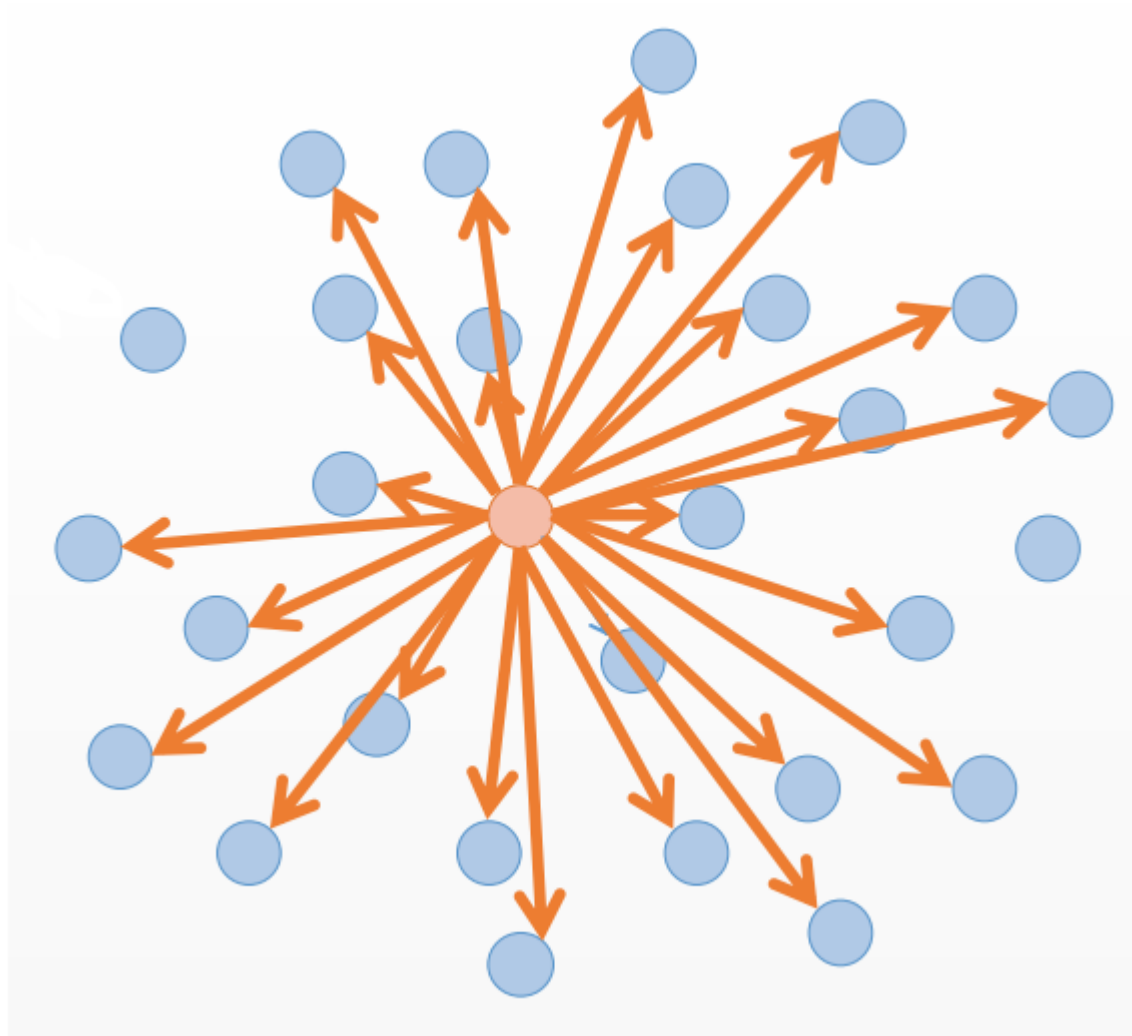
Aggiungere e rimuovere nodi nel cluster

- Metodo Direct Mail



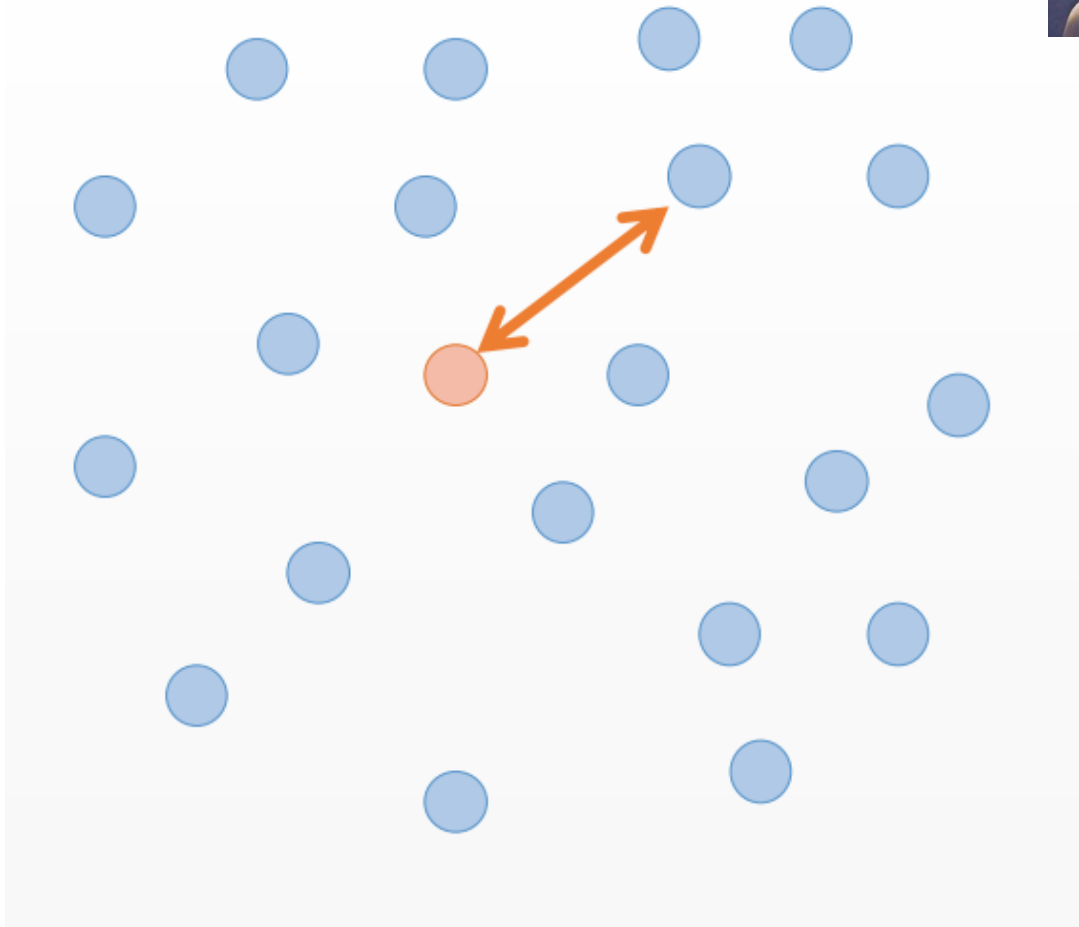
Aggiungere e rimuovere nodi nel cluster

- Metodo Direct Mail



Aggiungere e rimuovere nodi nel cluster

- Metodo Anti Entropy



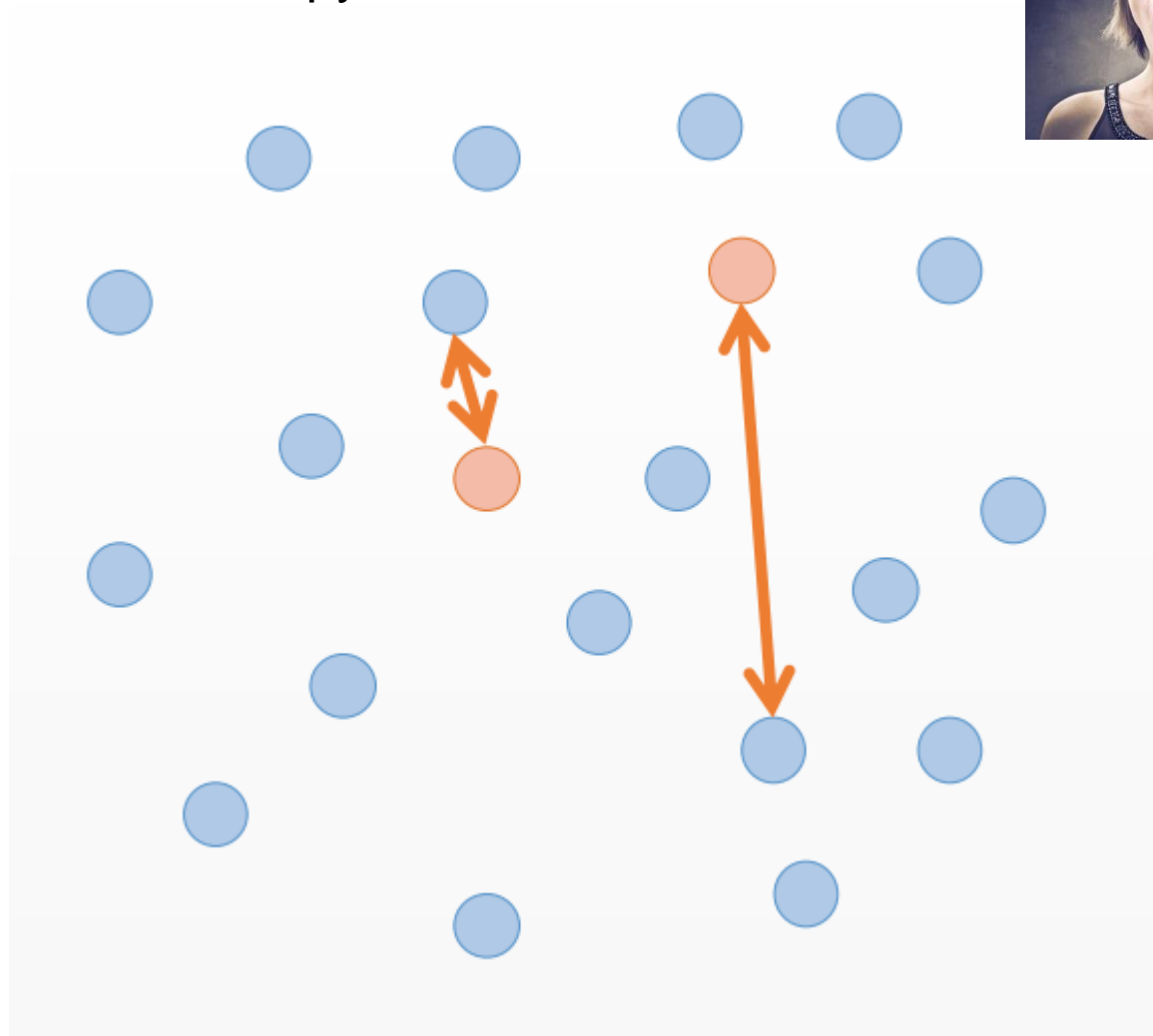
Aggiungere e rimuovere nodi nel cluster

- Metodo Anti Entropy



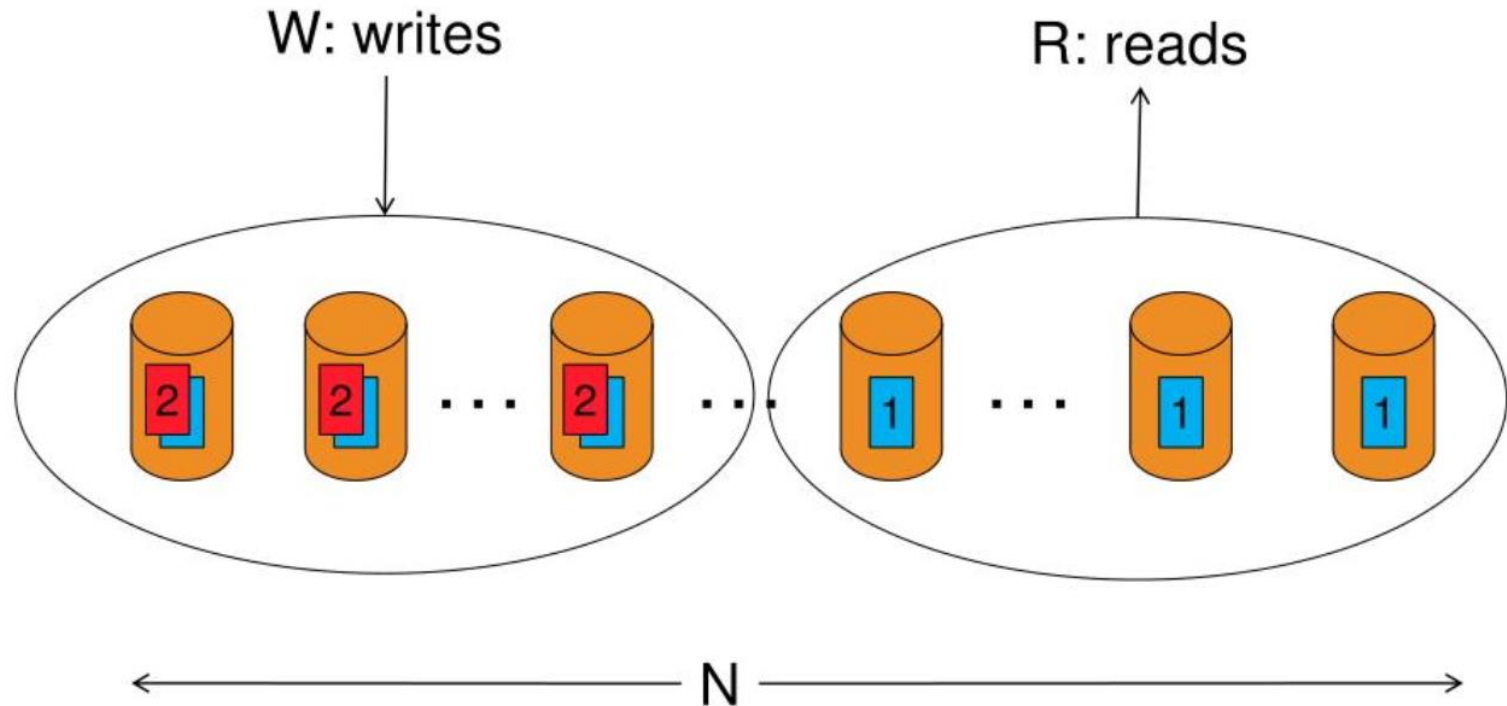
Aggiungere e rimuovere nodi nel cluster

- Metodo Anti Entropy



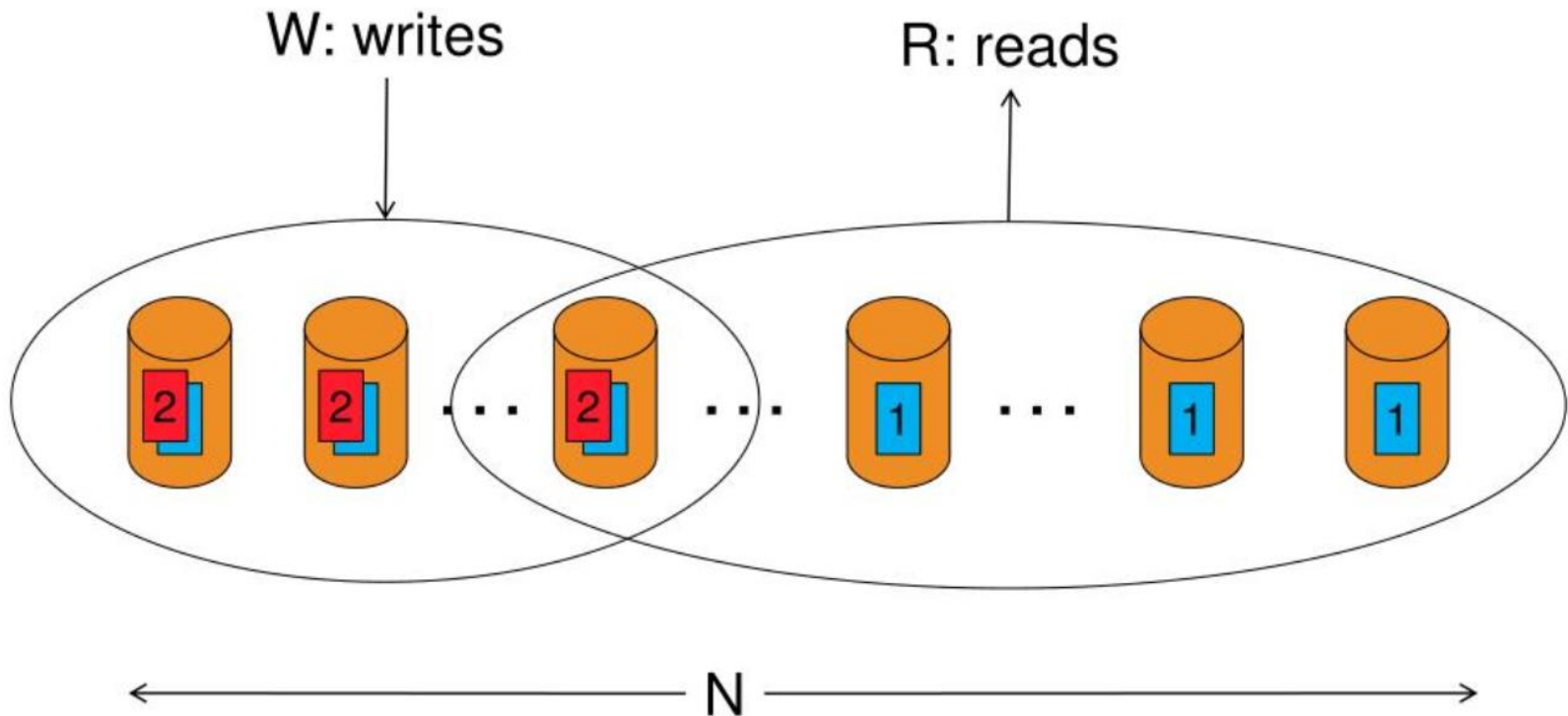
Quorum per lettura (R) e scrittura (W) su (N) nodi

$$W + R \leq N$$



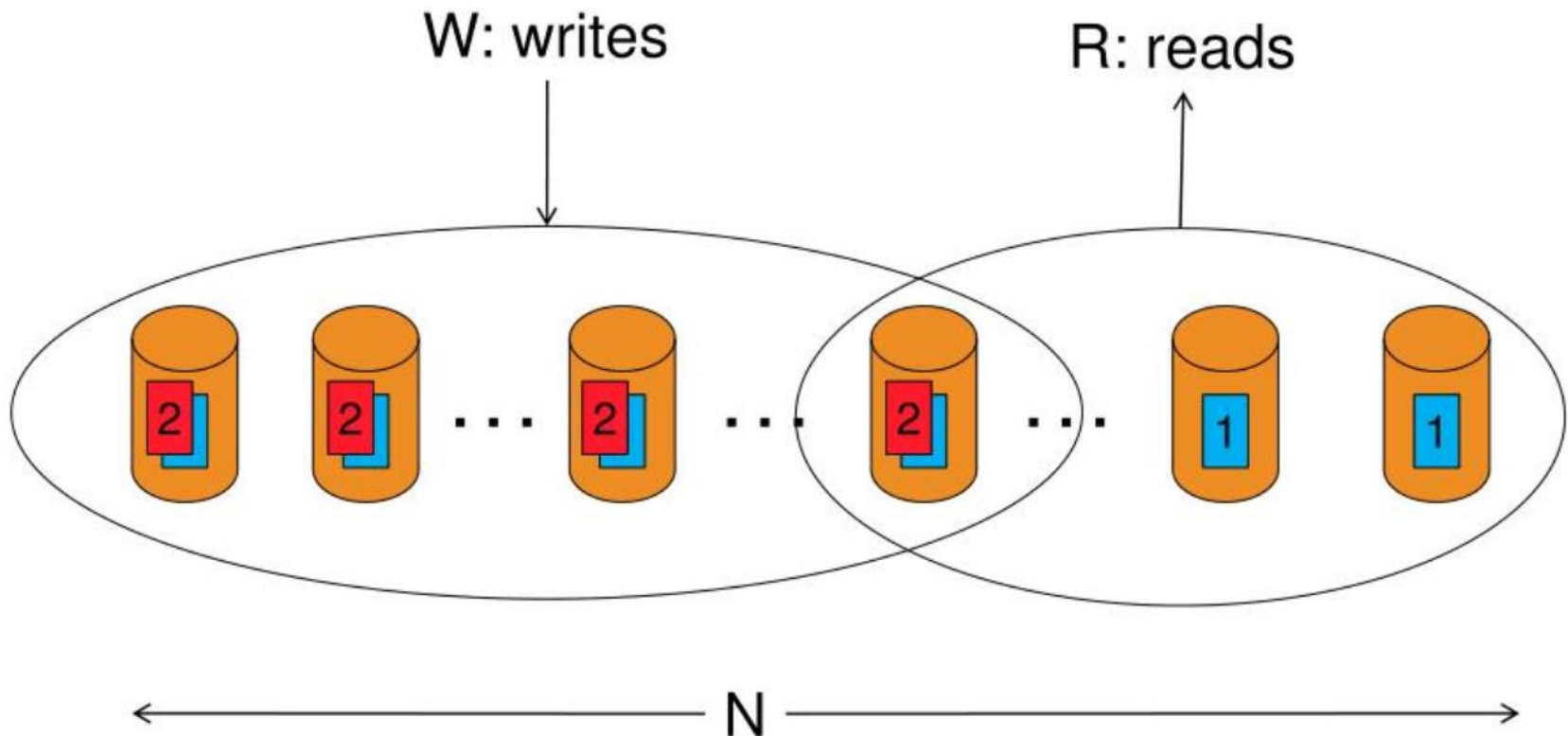
Quorum per lettura (R) e scrittura (W) su (N) nodi

$$W + R > N$$



Quorum per lettura (R) e scrittura (W) su (N) nodi

$$W + R > N$$



Quorum per lettura (R) e scrittura (W) su (N) nodi

- Configurazione code da web console

Queues

▼ All queues (2)

Pagination

Page 1 ▼ of 1 - Filter:

Overview

Virtual host	Name	Node
chat	hello	rabbit@instance-2
chat	task_queue	rabbit@instance-4 +3

▼ Add a new queue

Virtual host:

Type:

Name:

Node:

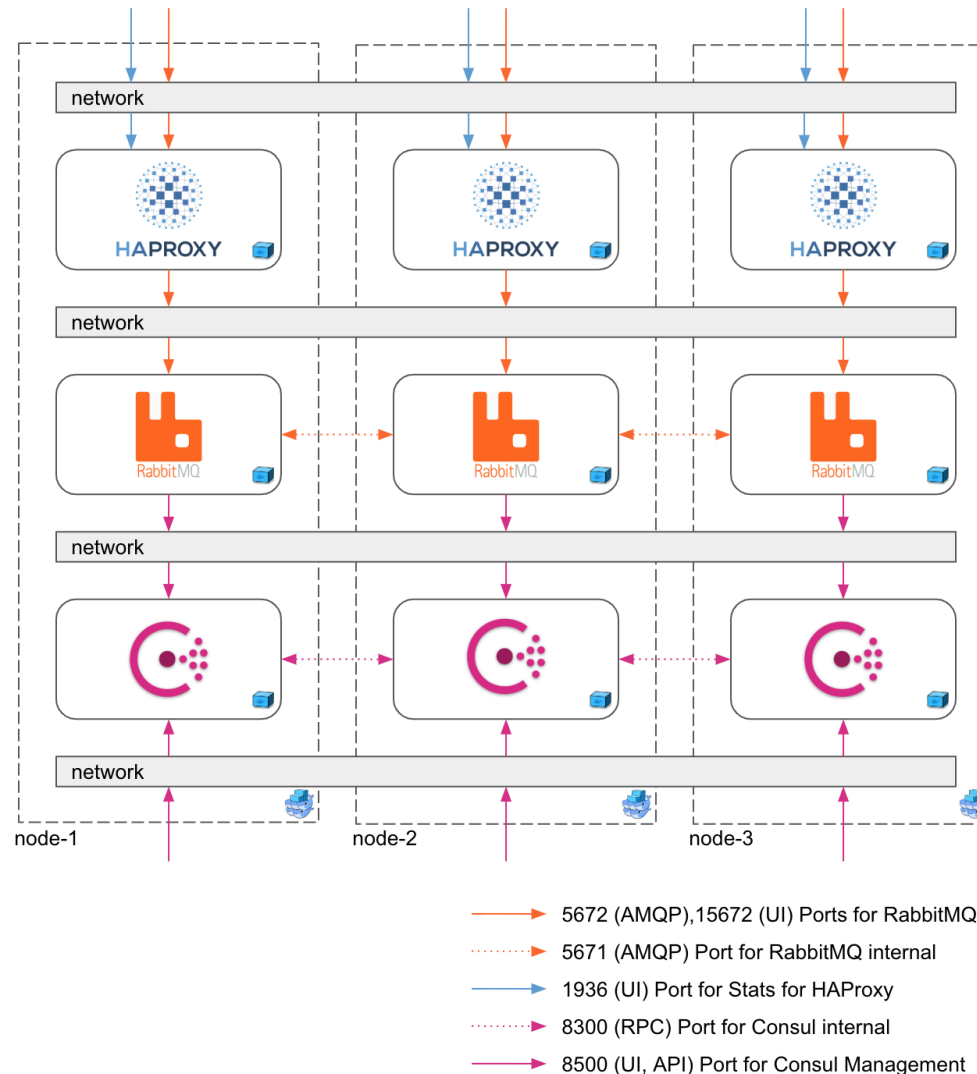
Arguments: =

Add Max length ? | Max length ? | Dead letter exchange ? | D

Configurazione avanzata

Configurazione attesa (3 nodi) swarm-node-X

<https://medium.com/hepsiburadatech/implementing-highly-available-rabbitmq-cluster-on-docker-swarm-using-consul-based-discovery-45c4e7919634>



Installazione docker – Su tutti i nodi! (Ubuntu)

- Installazione docker:
 - `curl -fsSL https://get.docker.com/ | sh`
- Controllo di versione:
 - `docker -v`

```
root@swarm-node-1:~# docker -v
Docker version 19.03.3, build a872fc2f86
root@swarm-node-1:~#
```

Configurazione SWARM (1)

- Sul nodo1
 - `docker swarm init --advertise-addr <IP-Interno>`
- Collegare il cluster sul nodo2 e sul nodo3
 - `docker swarm join --token <Token> <IP-Interno>:2377`
- Promuovere tutti i nodi a manager
 - `docker node promote swarm-node-<X>`
- Controllare i nodi del cluster
 - `docker node ls`

```
root@swarm-node-1:~# docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
xt83hl214iebne64114ltavcx *	swarm-node-1	Ready	Active	Leader	19.03.3
z1lqt6dsc58zvz8da5ngnygfj	swarm-node-2	Ready	Active	Reachable	19.03.3
chp5gv2xyggqdcqjry1jssbz	swarm-node-3	Ready	Active	Reachable	19.03.3

```
root@swarm-node-1:~#
```

Configurazione SWARM (2)

- Creare un overlay network
 - `docker network create --driver=overlay --attachable prod`
- Controllare il network
 - `docker network ls`

```
root@swarm-node-1:~# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
f146d5c126a6        bridge              bridge              local
c594r19367bx        consul_consul       overlay             swarm
1a2215fd6f4d        docker_gwbridge     bridge              local
351567ad0d21        host                host                local
ie81ju1dz4ye4        ingress             overlay             swarm
c41363cc57c2        none                null                local
q367ftc0ynnx        prod                overlay             swarm
root@swarm-node-1:~#
```

Consul

```
wget https://raw.githubusercontent.com/olgac/consul/master/docker-compose.yml  
docker node update --label-add consul=true swarm-node-1  
docker stack deploy -c docker-compose.yml consul
```

#Attesa per l'elezione del «Consul Leader»

```
sleep 10
```

#Aggiunta dei nodi Consul sul nodo2 e nodo3

```
docker node update --label-add consul=true swarm-node-2  
docker node update --label-add consul=true swarm-node-3
```

#Controllo del Consul Leader

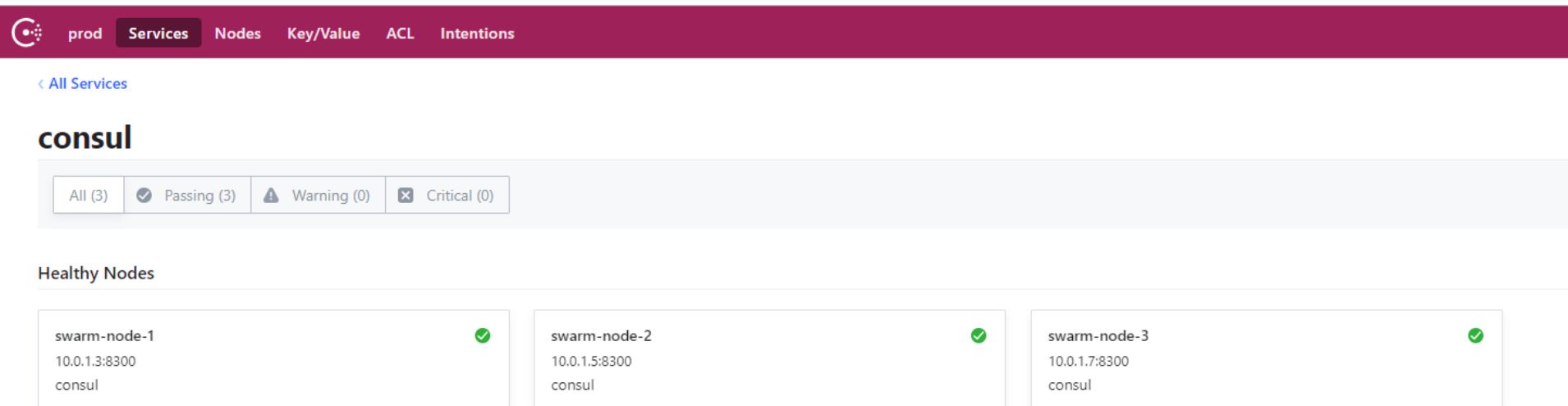
```
curl <IP-Leader>:8500/v1/status/leader
```

#Controllo dei Consul Peers

```
curl <IP-Leader>:8500/v1/status/peers
```

Consul interfaccia web

http://<IP-Leader>:8500/



The screenshot displays the Consul web interface. At the top, a dark red navigation bar contains the Consul logo and several tabs: 'prod', 'Services' (which is selected), 'Nodes', 'Key/Value', 'ACL', and 'Intentions'. Below the navigation bar, a blue link '< All Services' is visible. The main content area is titled 'consul' in a large, bold font. Below this title, there is a summary bar with four status indicators: 'All (3)', 'Passing (3)' (with a green checkmark icon), 'Warning (0)' (with a yellow warning triangle icon), and 'Critical (0)' (with a red 'X' icon). Underneath the summary bar, the section 'Healthy Nodes' is displayed. This section contains three separate boxes, each representing a healthy node. Each box lists the node name, its IP address and port, and the service it is running. All three nodes are marked with a green checkmark icon, indicating they are in a 'Passing' state.

Node Name	IP Address	Port	Service	Status
swarm-node-1	10.0.1.3	8300	consul	Passing
swarm-node-2	10.0.1.5	8300	consul	Passing
swarm-node-3	10.0.1.7	8300	consul	Passing

Docker compose

- `wget`
`https://raw.githubusercontent.com/olgac/rabbitmq/master/docker-compose.yml`
- `docker node update --label-add rabbitmq1=true node-1`
- `docker node update --label-add rabbitmq2=true node-2`
- `docker node update --label-add rabbitmq3=true node-3`
- `docker stack deploy -c docker-compose.yml rabbitmq`

HAProxy

(Su tutti i nodi!)

- `wget https://raw.githubusercontent.com/olgac/haproxy-for-rabbitmq/master/docker-compose.yml`
- `docker network create --driver=overlay --attachable prod`
- `docker stack deploy -c docker-compose.yml haproxy`

Verifica nodi su interfaccia web

- RabbitMQ
 - `http://<IP>:15672`
 - User: admin
 - Password: Passw0rd

RabbitMQ 3.7.8 Erlang 20.3.8.5

Overview Connections Channels Exchanges Queues Admin

Overview

▼ Totals

Queued messages **last minute** ?

Currently idle

Message rates **last minute** ?

Currently idle

Global counts ?

Connections: 0 **Channels: 0** **Exchanges: 7** **Queues: 0** **Consumers: 0**

▼ Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime
rabbit@rabbitmq-01	30 1048576 available	0 943626 available	398 1048576 available	83MB 2.9GB high watermark	7.0GB 4.7GB low watermark	3d 4h
rabbit@rabbitmq-02	31 1048576 available	0 943626 available	400 1048576 available	82MB 2.9GB high watermark	7.3GB 4.7GB low watermark	3d 3h
rabbit@rabbitmq-03	28 1048576 available	0 943626 available	399 1048576 available	82MB 2.9GB high watermark	7.0GB 4.7GB low watermark	3d 3h

▼ Ports and contexts

Verifica nodi su interfaccia web

- HAProxy
 - `http://<IP>:1936`

HAProxy

Statistics Report for pid 6

> General process information

pid = 0 (process #1, nbproc = 1, nbthread = 1)
uptime = 3d 4h00m34s
system limits: memmax = unlimited; ulimit-n = 8211
maxsock = 8211; maxconn = 4096; maxpipes = 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec
Running tasks: 1/11; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
backup UP
backup UP, going down
backup DOWN, going up
not checked
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

stats	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Status
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	
Frontend				1	1	-	1	1	2 000	0			42 548	2 310 718	0	0	5					OPEN
Backend	0	0		0	0		0	0	200	0	0s		42 548	2 310 718	0	0		0	0	0	0	3d4h UP

rabbitmq	Queue			Session rate			Sessions					Bytes		Denied		Errors		Warnings		Status		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp		Retr	Redis
Frontend				0	1	-	0	1		2 000	0		172	541	0	0	0					OPEN
rabbitmq-01	0	0	-	0	1		0	1		-	2	2d4h	16	16	0	0	0	0	0	0	0	3d4h UP
rabbitmq-02	0	0	-	0	1		0	1		-	2	16h29m	16	509	0	0	0	0	0	0	0	3d3h UP
rabbitmq-03	0	0	-	0	1		0	1		-	2	1d11h	140	16	0	0	0	0	0	0	0	3d4h UP
Backend	0	0		0	1		0	1		200	0	16h29m	172	541	0	0	0	0	0	0	0	3d4h UP

rabbitmq-ui	Queue			Session rate			Sessions					Bytes		Denied		Errors		Warnings		Status		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp		Retr	Redis
Frontend				0	1	-	0	1		2 000	15		132 194	348 678	0	0	0	0	0	0	0	OPEN
rabbitmq-01	0	0	-	0	2		0	1		-	103	103	1h42m	48 579	175 645	0	0	0	0	0	0	3d4h UP
rabbitmq-02	0	0	-	0	2		0	1		-	89	88	1h42m	39 083	83 819	0	0	1	0	3	0	3d3h UP
rabbitmq-03	0	0	-	0	2		0	1		-	103	103	1h42m	48 532	89 214	0	0	0	0	0	0	3d4h UP
Backend	0	0		0	5		0	1		200	292	292	1h42m	132 194	348 678	0	0	1	0	3	0	3d4h UP