# 

# Guia Completo de Instalação e Configuração RabbitMQ

## Implantação em Linux *Bare Metal* - Ambiente Corporativo B3

**Documento Técnico**  
**Versão:** 1.0  
**Data:** Setembro 2025  
**Ambiente:** RHEL 9

**B3**

30 de setembro de 2025

## Índice (Sumário)

Sumário

[Guia Completo de Instalação e Configuração RabbitMQ 1](#_Toc210141695)

[Implantação em Linux Bare Metal - Ambiente Corporativo 1](#_Toc210141696)

[Índice (Sumário) 2](#_Toc210141697)

[1. Instalação do Erlang/OTP (Offline) 3](#_Toc210141698)

[Instalar e validar Erlang 27.x (compatível com RabbitMQ 4.x) no RHEL 9 (air-gapped) 3](#_Toc210141699)

[2. Instalação do RabbitMQ (Offline) 5](#_Toc210141700)

[Instalar e validar RabbitMQ 4.1.0 no RHEL 9 (air-gapped) 5](#_Toc210141701)

[3. Configuração Inicial 8](#_Toc210141702)

[3.1 Criação de Usuários e Papéis (Tags) 8](#_Toc210141703)

[3.2 Criação de Virtual Hosts (vhosts) 8](#_Toc210141704)

[3.3 Permissões por vhost 8](#_Toc210141705)

[4. Configuração de Rede e Segurança 9](#_Toc210141706)

[4.1 Liberar portas 9](#_Toc210141707)

[4.2 Ativar TLS (opcional, recomendado) 9](#_Toc210141708)

[5. Configuração de Cluster 10](#_Toc210141709)

[5.1 Visão Geral 10](#_Toc210141710)

[5.2 Configuração de nomes e resolução de hosts 10](#_Toc210141711)

[5.3 Passos para formar o cluster 11](#_Toc210141712)

[5.4 Melhores práticas para alta disponibilidade 12](#_Toc210141713)

[6. Ativação de Plugins 14](#_Toc210141714)

[6.1 Plugins essenciais 14](#_Toc210141715)

[7. Configuração de Monitoramento 14](#_Toc210141716)

[7.1 Preparação do ambiente no Red Hat 8 14](#_Toc210141717)

[7.2 Como subir o Prometheus via Podman em um dos nodes RabbitMQ 14](#_Toc210141718)

[7.3 Observabilidade completa: Prometheus, Grafana e Alertas 16](#_Toc210141719)

[8. Testes de Funcionamento 17](#_Toc210141720)

[8.1 Passo pré-execução: Instalar rabbitmqadmin em todos os nodes 17](#_Toc210141721)

[8.2 Teste de Failover no Cluster RabbitMQ 18](#_Toc210141722)

[8.3 Testes básicos de funcionamento 19](#_Toc210141723)

[9. Setup de HAProxy 19](#_Toc210141724)

[9.1 Setup de HAProxy para RabbitMQ (Instalação Offline) 19](#_Toc210141725)

[9.2 Instalação do HAProxy (Offline) 19](#_Toc210141726)

[9.3 Configuração do HAProxy 22](#_Toc210141727)

[9.4 Inicialização do HAProxy 22](#_Toc210141728)

[9.5 Teste de funcionamento 22](#_Toc210141729)

[9.6 Alta disponibilidade do HAProxy (opcional) 22](#_Toc210141730)

[9.7 Observações 22](#_Toc210141731)

[9.8 Endereço único e alta disponibilidade 22](#_Toc210141732)

[9.9 Testando o balanceamento e failover do HAProxy com RabbitMQ 23](#_Toc210141733)

[Conclusão 24](#_Toc210141734)

[Pontos principais cobertos: 24](#_Toc210141735)

[Próximos passos recomendados: 25](#_Toc210141736)

## 1. Instalação do Erlang/OTP (Offline)

### Instalar e validar Erlang 27.x (compatível com RabbitMQ 4.x) no RHEL 9 (air-gapped)

#### 1.1 Pré-requisitos

* Usuário com sudo.
* Ambiente RHEL 9 (Enterprise Linux 9).
* Porta de upload de arquivos acessível via SCP/SFTP (MobaXterm ou similar).

#### 1.2 No Windows (PowerShell) — baixar arquivos

Crie/entre na pasta de staging:

cd C:\erlang-rhel9-offline

Baixe o pacote do **Erlang 27.3.4.2 (zero-dependency, EL9, x86\_64)**:

curl.exe -o erlang-27.3.4.2-1.el9.x86\_64.rpm "https://github.com/rabbitmq/erlang-rpm/releases/download/v27.3.4.2/erlang-27.3.4.2-1.el9.x86\_64.rpm"

(Se seu servidor for **ARM/aarch64**, baixe o arquivo correspondente da mesma release do repositório rabbitmq/erlang-rpm).

Confirme o tamanho (~25 MB):

Get-Item .\erlang-27.3.4.2-1.el9.x86\_64.rpm | Select-Object Length

#### 1.3 Upload para a VM

* **Painel SFTP (drag-and-drop)** → para /tmp
* ou **SCP no PowerShell**:

scp C:\erlang-rhel9-offline\erlang-27.3.4.2-1.el9.x86\_64.rpm usuario@IP\_DO\_SERVIDOR:/tmp/

⚠️ **Ponto de atenção — 2FA**: cada upload pode abrir **sub-sessão** → aprove no app **2FA** sempre que solicitado.

#### 1.4 Instalar na VM (offline)

##### 1.4.1 Remover Erlang existente (se já houver)

rpm -qa | grep -i erlang  
sudo dnf remove -y erlang

##### 1.4.2 Instalar Erlang 27.x

cd /tmp  
sudo dnf localinstall -y erlang-27.3.4.2-1.el9.x86\_64.rpm

#### 1.5 Validar a instalação

##### 1.5.1 Checar versão

erl -version

##### 1.5.2 Checar OTP release

erl -noshell -eval 'io:format("~s~n",[erlang:system\_info(otp\_release)]), halt().'

Saída esperada: 27

##### 1.5.3 Metadados do pacote

rpm -qi erlang

#### 1.6 Teste rápido — Hello World em Erlang

cat > hello.erl <<'EOF'  
-module(hello).  
-export([start/0]).  
  
start() ->  
 io:format("Hello, world!~n").  
EOF  
  
erlc hello.erl  
erl -noshell -eval 'hello:start(), halt().'

Saída esperada:

Hello, world!

#### 1.7 Troubleshooting rápido

* **Arquivo inválido** → confira com file erlang-27\*.rpm se é RPM válido.
* **Dependências quebradas** → baixe o pacote correto da release rabbitmq/erlang-rpm (zero-dependency).
* **Versão incorreta** → rode erl -noshell -eval 'io:format("~s~n",[erlang:system\_info(otp\_release)]), halt().' para confirmar que está em 27.

## 2. Instalação do RabbitMQ (Offline)

### Instalar e validar RabbitMQ 4.1.0 no RHEL 9 (air-gapped)

#### 2.1 Pré-requisitos

* Erlang 27 já instalado e validado (compatível com RabbitMQ 4.x).
* Usuário com sudo.
* Porta **5672/tcp** (AMQP) e **15672/tcp** (Management UI) liberadas no firewall, caso queira acesso externo.

#### 2.2 No Windows (PowerShell) — baixar arquivos

Crie/entre na pasta de staging:

cd C:\erlang-rhel9-offline

Baixe o pacote do **RabbitMQ Server 4.1.0** (RPM para EL8, compatível com RHEL 9):

curl.exe -o rabbitmq-server-4.1.0-1.el8.noarch.rpm "https://github.com/rabbitmq/rabbitmq-server/releases/download/v4.1.0/rabbitmq-server-4.1.0-1.el8.noarch.rpm"

Confirme o tamanho (~20 MB):

Get-Item .\rabbitmq-server-4.1.0-1.el8.noarch.rpm | Select-Object Length

Se precisar também de dependências (socat, logrotate), baixe dos mirrors CentOS Stream 9 com o mesmo padrão de comando:

curl.exe -o socat-1.7.4.1-5.el9.x86\_64.rpm "http://mirror.stream.centos.org/9-stream/AppStream/x86\_64/os/Packages/socat-1.7.4.1-5.el9.x86\_64.rpm"  
curl.exe -o logrotate-3.18.0-3.el9.x86\_64.rpm "http://mirror.stream.centos.org/9-stream/BaseOS/x86\_64/os/Packages/logrotate-3.18.0-3.el9.x86\_64.rpm"

#### 2.3 Upload para a VM

Mesmo processo usado no Erlang:

* **Painel SFTP (drag-and-drop)** → para /tmp
* ou **SCP no PowerShell**:

scp C:\erlang-rhel9-offline\rabbitmq-server-4.1.0-1.el8.noarch.rpm usuario@IP\_DO\_SERVIDOR:/tmp/  
scp C:\erlang-rhel9-offline\socat-1.7.4.1-5.el9.x86\_64.rpm usuario@IP\_DO\_SERVIDOR:/tmp/  
scp C:\erlang-rhel9-offline\logrotate-3.18.0-3.el9.x86\_64.rpm usuario@IP\_DO\_SERVIDOR:/tmp/

⚠️ **Ponto de atenção — 2FA**: cada upload pode abrir **sub-sessão** → aprove no app **2FA** sempre que solicitado.

#### 2.4 Instalar na VM (offline)

Na VM:

cd /tmp  
sudo dnf localinstall -y socat-\*.rpm logrotate-\*.rpm rabbitmq-server-\*.rpm

Verifique:

rpm -qi rabbitmq-server

#### 2.5 Habilitar e iniciar o serviço

sudo systemctl enable rabbitmq-server  
sudo systemctl start rabbitmq-server  
sudo systemctl status rabbitmq-server

#### 2.6 Configurar usuário admin e Management UI

Ative o plugin de console:

sudo rabbitmq-plugins enable rabbitmq\_management

Crie usuário admin (⚠️ use aspas simples para evitar erro de expansão de histórico no bash):

sudo rabbitmqctl add\_user admin 'StrongPass!123'

Defina privilégios:

sudo rabbitmqctl set\_user\_tags admin administrator  
sudo rabbitmqctl set\_permissions -p / admin ".\*" ".\*" ".\*"

#### 2.7 Firewall (validação inicial vs produção)

##### 2.7.1 Abrir portas para teste/validação (se firewalld estiver ativo)

sudo firewall-cmd --permanent --add-port=5672/tcp  
sudo firewall-cmd --permanent --add-port=15672/tcp  
sudo firewall-cmd --reload

##### 2.7.2 Caso apareça a mensagem **“FirewallD is not running”**

Isso significa que o servidor não usa firewalld localmente.  
Nessa situação:  
- As portas já estarão acessíveis conforme a política de rede.  
- O controle de acesso provavelmente é feito em **firewalls externos** ou regras corporativas de rede.  
- Para produção, alinhar com o time de segurança/rede se as portas devem ser expostas diretamente ou publicadas via proxy/VPN.

#### 2.8 Validar

##### 2.8.1 Validar serviço

sudo systemctl status rabbitmq-server

##### 2.8.2 Validar console de administração (UI)

Descubra o nome da máquina (hostname) e/ou IP para montar o link de acesso externo:

hostname # nome curto  
hostname -f # FQDN (se configurado)  
hostname -I # IP(s) da máquina

Exemplo de acesso pelo browser (do seu desktop):

http://NOME-DA-MAQUINA:15672  
ou  
http://IP-DA-MAQUINA:15672

→ login: admin / StrongPass!123

#### 2.9 Troubleshooting rápido

* **Erro de dependência no localinstall** → baixe os RPMs EL9 faltantes e instale junto (sudo dnf localinstall -y \*.rpm).
* **Serviço não sobe** → veja logs:
* sudo journalctl -u rabbitmq-server -f
* **Permissões negadas** → confirme que está rodando como sudo e que o diretório /var/lib/rabbitmq/ pertence ao usuário rabbitmq.

## 3. Configuração Inicial

### 3.1 Criação de Usuários e Papéis (Tags)

O RabbitMQ utiliza “tags” para definir papéis de usuários, como administrator (administração total), monitoring (somente leitura/monitoramento), entre outros. Não há grupos nativos, mas as tags cumprem esse papel.

#### Exemplo: Usuários administrativos e de monitoramento

# Usuário admin (acesso total)  
sudo rabbitmqctl add\_user admin 'SenhaForte!123'  
sudo rabbitmqctl set\_user\_tags admin administrator  
  
# Usuário somente leitura (monitoramento)  
sudo rabbitmqctl add\_user viewer 'SenhaLeitura!123'  
sudo rabbitmqctl set\_user\_tags viewer monitoring

### 3.2 Criação de Virtual Hosts (vhosts)

Virtual hosts permitem isolar ambientes, aplicações ou times dentro do mesmo cluster RabbitMQ. Bons exemplos de uso:

# VHOST para produção  
sudo rabbitmqctl add\_vhost prod\_app  
  
# VHOST para homologação  
sudo rabbitmqctl add\_vhost staging\_app  
  
# VHOST para integração de sistemas  
sudo rabbitmqctl add\_vhost integracao

### 3.3 Permissões por vhost

Defina permissões específicas para cada usuário em cada vhost:

# Permissões totais para admin em todos os vhosts  
sudo rabbitmqctl set\_permissions -p prod\_app admin ".\*" ".\*" ".\*"  
sudo rabbitmqctl set\_permissions -p staging\_app admin ".\*" ".\*" ".\*"  
sudo rabbitmqctl set\_permissions -p integracao admin ".\*" ".\*" ".\*"  
  
# Permissões de leitura para viewer apenas em produção  
sudo rabbitmqctl set\_permissions -p prod\_app viewer "^$" ".\*" ".\*"

**Dica:** - Use nomes de vhost que representem ambientes reais (ex: prod\_app, staging\_app, integracao). - Crie usuários com papéis distintos para separar administração de monitoramento. - Sempre defina permissões mínimas necessárias para cada usuário/vhost.

## 4. Configuração de Rede e Segurança

### 4.1 Liberar portas

Se firewall ativo:

sudo ufw allow 5672  
sudo ufw allow 15672

### 4.2 Ativar TLS (opcional, recomendado)

#### 4.2.1 Gerar certificados autoassinados (para testes/lab)

**IMPORTANTE:** Ao rodar o comando openssl, quando solicitado o “Common Name (CN)”, informe o nome da máquina (hostname ou FQDN) que será usado para acessar o RabbitMQ via HTTPS. **Exemplo:** para acessar https://node00:15671, informe node00 como CN. Para domínio, use o FQDN.

Crie os arquivos diretamente na pasta esperada:

sudo mkdir -p /etc/rabbitmq/  
cd /etc/rabbitmq/  
sudo openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out cert.pem  
# Para ambiente de teste, use o próprio cert.pem como ca.pem:  
sudo cp cert.pem ca.pem  
# Ajustar permissões dos arquivos:  
sudo chown rabbitmq:rabbitmq /etc/rabbitmq/\*.pem  
sudo chmod 600 /etc/rabbitmq/\*.pem

#### 4.2.2 Editar configuração do RabbitMQ

Se o comando abaixo retornar “command not found”, instale o nano com:

sudo dnf install -y nano

Depois edite o arquivo:

sudo nano /etc/rabbitmq/rabbitmq.conf

Adicionar no arquivo:

# Listeners AMQP  
listeners.tcp.default = 5672  
listeners.ssl.default = 5671  
ssl\_options.cacertfile = /etc/rabbitmq/ca.pem  
ssl\_options.certfile = /etc/rabbitmq/cert.pem  
ssl\_options.keyfile = /etc/rabbitmq/key.pem  
ssl\_options.verify = verify\_peer  
ssl\_options.fail\_if\_no\_peer\_cert = true  
  
# Management UI  
management.tcp.port = 15672  
management.ssl.port = 15671  
management.ssl.cacertfile = /etc/rabbitmq/ca.pem  
management.ssl.certfile = /etc/rabbitmq/cert.pem  
management.ssl.keyfile = /etc/rabbitmq/key.pem

#### 4.2.3 Reiniciar serviço

sudo systemctl restart rabbitmq-server

**Nota:** - Em ambiente corporativo, utilize certificados e CA oficiais fornecidos pela empresa. - Para testes, o próprio cert.pem pode ser usado como ca.pem, mas isso não é recomendado para produção. - Assegure-se de que os arquivos .pem pertençam ao usuário rabbitmq e tenham permissão restrita (600) para evitar falhas de inicialização. - O Common Name (CN) do certificado deve ser igual ao hostname ou FQDN usado no acesso HTTPS. - O acesso ao Management UI via HTTPS será feito pela porta 15671 (https://nomedamaquina:15671).

## 5. Configuração de Cluster

### 5.1 Visão Geral

O cluster RabbitMQ será composto por **5 nós** (node00 a node04), conforme descrito em infra.md. Todos os nós já devem ter executado os passos de instalação e configuração inicial (capítulos 1, 2, 3, 4).

* **Alta disponibilidade:** Todos os nós participam do cluster, suportando mirrored queues e quorum queues para ambientes críticos.
* **Disc nodes:** Em versões recentes, todos os nós podem ser disc nodes (recomendado para resiliência).

### 5.2 Configuração de nomes e resolução de hosts

#### 5.2.1 Resolução de nomes dos nós

**Recomendado para ambientes corporativos:** - Utilize DNS interno para garantir que todos os nós consigam resolver os nomes uns dos outros (ex: node00, node01, node02, node03, node04). - O uso de DNS permite mudanças de IP sem impacto para o cluster.

Todos os nós devem conseguir resolver os nomes dos demais para garantir comunicação correta.

### 5.3 Passos para formar o cluster

#### 5.3.1 Parar o aplicativo RabbitMQ nos nós secundários

Execute nos nós que vão se juntar ao cluster (node01, node02, node03, node04):

sudo rabbitmqctl stop\_app

#### 5.3.2 Juntar os nós ao cluster

Execute nos nós secundários, sempre apontando para o nó inicial do cluster (**node00**).

**Checklist prático antes do join\_cluster:**

1. **Conferir hostname e resolução:**
   * No node01, rode:
   * hostname  
     ping NOME\_DO\_NODE00
   * O ping deve responder e o nome do node00 deve ser o mesmo que aparece em rabbitmqctl status no node00.
2. **Conferir nome do nó RabbitMQ:**
   * No node00, rode:
   * sudo rabbitmqctl status | grep 'Node'
   * Use exatamente esse nome (ex: rabbit@nodeName00c ou rabbit@nodeName00c.internalenv.corp, conforme aparece no status). **Na prática, normalmente é apenas o hostname curto, como rabbit@nodeName00c.**
3. **Padronizar o arquivo .erlang.cookie:**
   * O conteúdo do arquivo /var/lib/rabbitmq/.erlang.cookie deve ser idêntico em todos os nós.
   * No node00:
   * sudo cat /var/lib/rabbitmq/.erlang.cookie
   * Copie esse conteúdo para os demais nós (node01 a node04):
   * sudo nano /var/lib/rabbitmq/.erlang.cookie  
     sudo chown rabbitmq:rabbitmq /var/lib/rabbitmq/.erlang.cookie  
     sudo chmod 400 /var/lib/rabbitmq/.erlang.cookie
   * Reinicie o serviço rabbitmq-server nos nós onde o cookie foi alterado:
   * sudo systemctl restart rabbitmq-server
4. **Firewall/Portas:**
   * Confirme que a porta 25672/TCP está liberada entre os nós:
   * nc -zv NOME\_DO\_NODE00 25672
5. **Serviço RabbitMQ:**
   * O serviço rabbitmq-server deve estar rodando no node00 (nó principal) e parado nos nós que vão se juntar.

**Exemplo de comando join\_cluster:**

# No node01  
sudo rabbitmqctl stop\_app  
sudo rabbitmqctl join\_cluster rabbit@nodeName00c  
sudo rabbitmqctl start\_app

Todos os nós secundários devem se juntar ao cluster iniciado pelo node00, usando o nome exato do nó principal. Não utilize rabbit@node01, rabbit@node02, etc. como destino. Dica: O nome do nó principal geralmente é apenas o hostname curto (ex: rabbit@nodeName00c). Use exatamente o que aparece em rabbitmqctl status no node00.

##### Checklist de troubleshooting para erro de conexão (ex: {erpc,noconnection})

Se aparecer erro de conexão ao rodar o join\_cluster, revise todos os itens do checklist acima. Os problemas mais comuns são: - Nome do nó incorreto (use sempre o nome exato do node00) - .erlang.cookie diferente entre os nós - Falha de resolução de nomes ou firewall bloqueando porta 25672

Corrija qualquer inconsistência e tente novamente o comando join\_cluster.

#### 5.3.3 Iniciar o aplicativo RabbitMQ nos nós secundários

sudo rabbitmqctl start\_app

#### 5.3.4 Verificar status do cluster

Em qualquer nó:

sudo rabbitmqctl cluster\_status

### 5.4 Melhores práticas para alta disponibilidade

* **Mirrored Queues:**
  + Use políticas para replicar filas críticas entre os nós.
  + Exemplo de política para todas as filas (pode ser executado em qualquer nó do cluster):
  + sudo rabbitmqctl set\_policy ha-all ".\*" '{"ha-mode":"all"}' --priority 1 --apply-to queues
  + Para verificar se a política foi aplicada:
  + sudo rabbitmqctl list\_policies
  + O resultado deve mostrar a política ha-all aplicada.
  + Mirrored queues são recomendadas para compatibilidade, mas para novos projetos prefira quorum queues.
* **Quorum Queues:**
  + São nativamente distribuídas e tolerantes a falhas, recomendadas para novos projetos.
  + Não dependem de política global: cada fila deve ser criada explicitamente como tipo quorum.
  + Exemplo de criação via CLI (rabbitmqadmin):
  + sudo rabbitmqadmin declare queue name=nome\_da\_fila type=quorum durable=true
  + Exemplo via Management UI:
    - Ao criar uma fila, selecione o tipo “quorum” no campo “Type”.
  + Para listar filas quorum já criadas:
  + sudo rabbitmqctl list\_queues name type
  + O tipo da fila deve aparecer como quorum.
* **Sincronização de horário:**
  + Todos os nós devem estar com NTP ativo para evitar problemas de cluster.

**Notas:** - O comando join\_cluster só é executado nos nós que vão se juntar ao cluster (node01 a node04). O node00 é o nó inicial. - Todos os nós devem ter as mesmas configurações de usuário, vhost e permissões para garantir consistência. - Para ambientes críticos, monitore o cluster e configure alertas para falhas de nó ou de sincronização de filas. - A ativação dos plugins de gerenciamento e monitoramento está detalhada nos capítulos 6 e 7 deste guia. - **Importante:** O suporte a alta disponibilidade (mirrored queues e quorum queues) depende de políticas e configurações aplicadas após o cluster estar formado. A etapa de cluster garante apenas a comunicação entre os nós; a replicação e tolerância a falhas das filas é definida posteriormente, conforme exemplos acima.

## 6. Ativação de Plugins

### 6.1 Plugins essenciais

Execute em todos os nós do cluster:

sudo rabbitmq-plugins enable rabbitmq\_management  
sudo rabbitmq-plugins enable rabbitmq\_prometheus

## 7. Configuração de Monitoramento

### 7.1 Preparação do ambiente no Red Hat 8

#### 7.1.1 Verifique/instale o Podman

No Red Hat 8, o Podman pode ser instalado com:

sudo dnf install -y podman

Verifique se o comando está disponível:

podman --version

#### 7.1.2 Crie o diretório para configuração do Prometheus

sudo mkdir -p /opt/prometheus  
sudo chown $(whoami) /opt/prometheus

**Observação:** O parâmetro :Z no comando -v do podman é necessário para compatibilidade com SELinux (ativo por padrão no RHEL 8).

### 7.2 Como subir o Prometheus via Podman em um dos nodes RabbitMQ

#### 7.2.1 Ative o plugin prometheus em todos os nodes RabbitMQ

Em cada node do cluster:

sudo rabbitmq-plugins enable rabbitmq\_prometheus

Cada node irá expor métricas em http://<NOME\_DO\_NODE>:15692/metrics.

#### 7.2.2 Escolha um node para rodar o Prometheus

Você pode rodar o Prometheus em qualquer node do cluster (ou em um servidor dedicado, se preferir). Para ambientes de teste/lab, rodar em um node RabbitMQ é aceitável.

#### 7.2.3 Crie o arquivo prometheus.yml (apenas no node onde rodará o Prometheus)

No node nodeName01c (onde o Prometheus será instalado), crie o arquivo de configuração:

sudo nano /opt/prometheus/prometheus.yml

Cole o conteúdo abaixo, ajustando se necessário:

global:  
 scrape\_interval: 15s  
  
scrape\_configs:  
 - job\_name: 'rabbitmq'  
 static\_configs:  
 - targets: ['nodeName00c:15692', 'nodeName01c:15692', 'nodeName02c:15692', 'nodeName03c:15692', 'nodeName04c:15692']

Salve e feche o arquivo.

**Importante:** O arquivo prometheus.yml é único e fica apenas no node nodeName01c (Prometheus). Não precisa replicar nos outros nodes.

#### 7.2.4 Baixe a imagem oficial do Prometheus

##### Download offline da imagem Prometheus

###### 7.2.4.1 No Windows (PowerShell) — baixar imagem

No seu computador com acesso à internet:

podman pull docker.io/prom/prometheus:latest  
podman save -o prometheus-latest.tar docker.io/prom/prometheus:latest

###### 7.2.4.2 Upload para o servidor

Transfira o arquivo prometheus-latest.tar para o servidor (ex: usando MobaXterm/SCP/SFTP) para a pasta /tmp.

###### 7.2.4.3 No servidor (offline) — importar imagem

No node nodeName01c:

cd /tmp  
sudo podman load -i prometheus-latest.tar

#### 7.2.5 Rode o container Prometheus

sudo podman run -d \  
 --name prometheus \  
 -p 9090:9090 \  
 -v /opt/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml:Z \  
 docker.io/prom/prometheus:latest

#### 7.2.6 Acesse a interface web do Prometheus

Abra no navegador:

http://<IP\_DO\_NODE>:9090

#### 7.2.7 (Opcional) Parar/remover o container

podman stop prometheus  
podman rm prometheus

**Resumo:** - Ative o plugin prometheus em todos os nodes RabbitMQ. - O arquivo prometheus.yml é único e fica apenas no node onde o Prometheus roda. - O Prometheus coleta métricas de todos os nodes, centralizando a observabilidade.

### 7.3 Observabilidade completa: Prometheus, Grafana e Alertas

#### 7.3.1 Ativação do plugin Prometheus

* O plugin rabbitmq\_prometheus deve ser ativado em **todos os nodes** do cluster:
* sudo rabbitmq-plugins enable rabbitmq\_prometheus
* Cada node irá expor métricas em http://<NOME\_DO\_NODE>:15692/metrics.

#### 7.3.2 Configuração do Prometheus

* O arquivo prometheus.yml é configurado **apenas no servidor onde o Prometheus roda** (não precisa replicar nos nodes RabbitMQ).
* Inclua todos os nodes RabbitMQ como targets:
* scrape\_configs:  
   - job\_name: 'rabbitmq'  
   static\_configs:  
   - targets: ['nodeName00c:15692', 'nodeName01c:15692', 'nodeName02c:15692', 'nodeName03c:15692', 'nodeName04c:15692']
* Inicie o Prometheus apontando para esse arquivo:
* ./prometheus --config.file=prometheus.yml

#### 7.3.3 Dashboards Grafana

* Instale o Grafana e adicione o Prometheus como data source.
* Importe dashboards prontos para RabbitMQ, como o oficial:
  + [RabbitMQ Overview (ID 10991)](https://grafana.com/grafana/dashboards/10991-rabbitmq-overview/)
* No Grafana: “+ Import” → cole o ID do dashboard → selecione o Prometheus como data source.

#### 7.3.4 Alertas (Prometheus + Alertmanager)

* Configure regras de alerta no Prometheus, por exemplo:
* groups:  
   - name: rabbitmq-alerts  
   rules:  
   - alert: RabbitMQNodeDown  
   expr: up{job="rabbitmq"} == 0  
   for: 1m  
   labels:  
   severity: critical  
   annotations:  
   summary: "RabbitMQ node está fora do ar"
* Configure o Alertmanager para enviar alertas por e-mail, Slack, etc.
  + [Guia oficial Alertmanager](https://prometheus.io/docs/alerting/latest/alertmanager/)

#### 7.3.5 Fluxo resumido

1. Ative o plugin prometheus em todos os nodes RabbitMQ.
2. Configure o prometheus.yml no servidor Prometheus com todos os nodes como targets.
3. Suba o Prometheus e valide a coleta.
4. Adicione o Prometheus como data source no Grafana e importe dashboards.
5. Configure alertas conforme necessidade.

**Dúvida frequente:** - O plugin prometheus deve ser ativado em cada node RabbitMQ. - O arquivo prometheus.yml é único e fica apenas no servidor Prometheus, centralizando a coleta de todos os nodes.

## 8. Testes de Funcionamento

### 8.1 Passo pré-execução: Instalar rabbitmqadmin em todos os nodes

#### 8.1.1 Download e instalação do rabbitmqadmin

##### 8.1.1.1 Em um node do cluster com Management UI ativo (ex: nodeName01c)

No próprio node, baixe o script:

curl -o rabbitmqadmin http://localhost:15672/cli/rabbitmqadmin  
chmod +x rabbitmqadmin  
sudo mv rabbitmqadmin /usr/local/bin/

##### 8.1.1.2 Transfira o arquivo para os demais nodes

Copie o arquivo rabbitmqadmin (agora em /usr/local/bin/) para os outros nodes do cluster usando SCP/SFTP/MobaXterm, ou baixe diretamente de cada node usando:

curl -o rabbitmqadmin http://nodeName01c:15672/cli/rabbitmqadmin  
chmod +x rabbitmqadmin  
sudo mv rabbitmqadmin /usr/local/bin/

O Management UI precisa estar ativo no node de origem (porta 15672 liberada na rede).

Assim, o comando rabbitmqadmin estará disponível em todos os nodes para os testes.

### 8.2 Teste de Failover no Cluster RabbitMQ

Siga o roteiro abaixo para validar o funcionamento do failover no cluster:

#### 8.2.1 Valide o status do cluster

sudo rabbitmqctl cluster\_status

#### 8.2.2 Crie uma fila quorum e publique mensagens

sudo rabbitmqadmin declare queue name=test\_quorum type=quorum durable=true  
sudo rabbitmqadmin publish queue=test\_quorum payload="Mensagem de teste 1"  
sudo rabbitmqadmin publish queue=test\_quorum payload="Mensagem de teste 2"

#### 8.2.3 Consuma mensagens da fila

sudo rabbitmqadmin get queue=test\_quorum requeue=false

#### 8.2.4 Identifique o node líder da fila

* No Management UI, veja em “Queues” qual node está como “Leader” da fila test\_quorum.

#### 8.2.5 Simule falha do node líder

* No node líder, pare o serviço RabbitMQ:

sudo systemctl stop rabbitmq-server

#### 8.2.6 Verifique failover

* No Management UI ou via CLI, veja se outro node assumiu como líder da fila.
* Tente consumir/publish na fila novamente:

sudo rabbitmqadmin get queue=test\_quorum requeue=false  
sudo rabbitmqadmin publish queue=test\_quorum payload="Mensagem após failover"

#### 8.2.7 Reinicie o node parado

sudo systemctl start rabbitmq-server

#### 8.2.8 Valide a reintegração do node ao cluster

sudo rabbitmqctl cluster\_status

**Dica:** Repita o teste para diferentes nodes e filas. Teste também o comportamento dos clientes conectados durante o failover.

### 8.3 Testes básicos de funcionamento

#### 8.3.1 Criar fila de teste

sudo rabbitmqadmin declare queue name=test\_queue durable=true

#### 8.3.2 Publicar mensagem

sudo rabbitmqadmin publish routing\_key=test\_queue payload="hello world"

#### 8.3.3 Consumir mensagem

sudo rabbitmqadmin get queue=test\_queue

## 9. Setup de HAProxy

### 9.1 Setup de HAProxy para RabbitMQ (Instalação Offline)

#### 9.1.1 Cenário

* Dois nodes HAProxy em alta disponibilidade (HA), atuando como balanceadores para o cluster RabbitMQ.
* Instalação offline, sem acesso à internet.
* HAProxy distribui conexões dos clientes entre os nodes RabbitMQ.

#### 9.1.2 Pré-requisitos

* Pacote HAProxy disponível localmente (RPM ou tar.gz).
* Acesso root nos nodes HAProxy.
* IPs/hosts dos nodes RabbitMQ conhecidos.

### 9.2 Instalação do HAProxy (Offline)

#### 9.2.1 Preparação dos nodes dedicados ao HAProxy (03c e 04c)

Para usar os servidores 03c e 04c exclusivamente como HAProxy:

##### 9.2.1.1 Pare e desabilite o RabbitMQ nos nodes 03c e 04c

Execute nos próprios nodes 03c e 04c:

sudo systemctl stop rabbitmq-server  
sudo systemctl disable rabbitmq-server

##### 9.2.1.2 Remova os nodes 03c e 04c do cluster RabbitMQ

Com os serviços já parados, execute em um node ativo do cluster (ex: 00c):

sudo rabbitmqctl forget\_cluster\_node rabbit@nodeName03c  
sudo rabbitmqctl forget\_cluster\_node rabbit@nodeName04c

Isso garante que o cluster não acuse ausência desses nodes e mantenha o status saudável.

**Observação:** O comando forget\_cluster\_node só funciona corretamente se o serviço RabbitMQ nos nodes a serem removidos estiver parado.

##### 9.2.1.3 Instale o HAProxy offline nos nodes 03c e 04c

###### 9.2.1.3.1 Baixe o pacote RPM do HAProxy em um computador com acesso à internet

Versão recomendada: haproxy-2.4.22-1.el9.x86\_64.rpm URL oficial CentOS Stream 9:

# Exemplo em ambiente Windows (PowerShell):  
curl.exe -o haproxy-2.4.22-1.el9.x86\_64.rpm "http://mirror.stream.centos.org/9-stream/AppStream/x86\_64/os/Packages/haproxy-2.4.22-1.el9.x86\_64.rpm"

Confirme o tamanho do arquivo baixado:

Get-Item .\haproxy-2.4.22-1.el9.x86\_64.rpm | Select-Object Length

###### 9.2.1.3.2 Faça o upload do arquivo para o diretório /tmp de cada node HAProxy

Use o Painel SFTP (drag-and-drop via MobaXterm) ou SCP:

scp C:\caminho\haproxy-2.4.22-1.el9.x86\_64.rpm usuario@IP\_DO\_NODE\_HAPROXY:/tmp/

###### 9.2.1.3.3 Instale o pacote transferido

No node HAProxy, instale o pacote:

sudo dnf install /tmp/haproxy-2.4.22-1.el9.x86\_64.rpm

Se houver dependências, baixe e transfira também os pacotes necessários e instale todos juntos:

sudo dnf install /tmp/haproxy-2.4.22-1.el9.x86\_64.rpm /tmp/dependencia1.rpm /tmp/dependencia2.rpm

###### 9.2.1.3.4 Configure o HAProxy normalmente

Cole ao final do arquivo /etc/haproxy/haproxy.cfg os blocos abaixo, que apontam apenas para os nodes RabbitMQ ativos (00c, 01c, 02c):

**Bloco para AMQP (porta 5672)**

frontend rabbitmq\_front  
 mode tcp  
 bind \*:5672  
 default\_backend rabbitmq\_nodes  
  
backend rabbitmq\_nodes  
 mode tcp  
 balance roundrobin  
 server nodeName00c nodeName00c:5672 check  
 server nodeName01c nodeName01c:5672 check  
 server nodeName02c nodeName02c:5672 check

**Bloco para AMQP com SSL/TLS (porta 5671)**

frontend rabbitmq\_front\_ssl  
 bind \*:5671  
 mode tcp  
 default\_backend rabbitmq\_nodes\_ssl  
  
backend rabbitmq\_nodes\_ssl  
 mode tcp  
 balance roundrobin  
 server nodeName00c nodeName00c:5671 check  
 server nodeName01c nodeName01c:5671 check  
 server nodeName02c nodeName02c:5671 check

**Bloco para Management UI HTTP (porta 15672)**

frontend rabbitmq\_mgmt  
 bind \*:15672  
 default\_backend rabbitmq\_mgmt\_nodes  
  
backend rabbitmq\_mgmt\_nodes  
 balance roundrobin  
 server nodeName00c nodeName00c:15672 check  
 server nodeName01c nodeName01c:15672 check  
 server nodeName02c nodeName02c:15672 check

**Bloco para Management UI HTTPS (porta 15671)**

frontend rabbitmq\_mgmt\_ssl  
 bind \*:15671  
 mode tcp  
 default\_backend rabbitmq\_mgmt\_nodes\_ssl  
  
backend rabbitmq\_mgmt\_nodes\_ssl  
 mode tcp  
 balance roundrobin  
 server nodeName00c nodeName00c:15671 check  
 server nodeName01c nodeName01c:15671 check  
 server nodeName02c nodeName02c:15671 check

**Importante:** - Realize a desativação do RabbitMQ apenas nos nodes 03c e 04c. - Configure o HAProxy para escutar nas portas padrão. - As aplicações devem apontar para os IPs/hostnames dos nodes HAProxy (03c e 04c) nas portas padrão.

### 9.3 Configuração do HAProxy

### 9.4 Inicialização do HAProxy

sudo systemctl enable --now haproxy  
sudo systemctl restart haproxy # Reinicie após alterar o haproxy.cfg  
sudo systemctl status haproxy

### 9.5 Teste de funcionamento

* Conecte um cliente à porta 5672 do HAProxy e verifique se a conexão é distribuída entre os nodes RabbitMQ.
* Teste failover: pare um node RabbitMQ e veja se o HAProxy redireciona para os nodes ativos.

### 9.6 Alta disponibilidade do HAProxy (opcional)

* Para HA real, configure um VIP (Virtual IP) com Keepalived ou Pacemaker entre os dois nodes HAProxy.
* O VIP garante que, se um HAProxy falhar, o outro assume automaticamente.
* Exemplo de configuração de Keepalived pode ser fornecido conforme necessidade.

### 9.7 Observações

* Não é necessário ajuste nos nodes RabbitMQ para uso com HAProxy.
* Garanta que as portas estejam liberadas no firewall dos nodes HAProxy e RabbitMQ.
* Documente os IPs/hosts usados no balanceamento para facilitar troubleshooting.

### 9.8 Endereço único e alta disponibilidade

Para que as aplicações utilizem um único endereço externo (IP ou hostname) e tenham alta disponibilidade, é necessário configurar um VIP (Virtual IP) entre os dois nodes HAProxy usando Keepalived ou Pacemaker.

**Exemplo de instrução para Keepalived:**

1. Instale o Keepalived em ambos os nodes HAProxy: bash sudo dnf install keepalived
2. Configure o arquivo /etc/keepalived/keepalived.conf em ambos os nodes, definindo o mesmo VIP (ex: 10.10.10.100) e prioridade diferente para cada node.
3. Inicie e habilite o serviço: bash sudo systemctl enable --now keepalived sudo systemctl status keepalived
4. As aplicações devem apontar para o VIP configurado, e não para os IPs individuais dos nodes HAProxy.

**Alternativas:** - Em ambientes mais avançados, pode-se usar Traefik, Consul ou Route53 para balanceamento, descoberta de serviços e failover dinâmico. - Essas soluções permitem automação, DNS dinâmico, health-checks e integração com cloud ou containers. - Para ambientes on-premise e simples, o VIP com Keepalived é a solução mais direta e robusta.

Se desejar instruções detalhadas para Traefik, Consul ou Route53, solicite conforme o contexto do seu ambiente.

**Dica:** Para ambientes offline, mantenha todos os pacotes necessários em um repositório local ou mídia removível.

### 9.9 Testando o balanceamento e failover do HAProxy com RabbitMQ

Esta seção orienta como validar que o HAProxy está distribuindo corretamente as conexões e garantindo alta disponibilidade para aplicações que utilizam filas quorum no RabbitMQ.

#### 9.9.1 Cenário de teste

* A aplicação deve apontar para o endereço de um dos nodes HAProxy (ex: 03c ou 04c) na porta 5672.
* A fila utilizada deve ser do tipo quorum (já criada e visível no admin do RabbitMQ).
* No admin do RabbitMQ, a fila quorum mostra o node líder (leader) e os membros online (members).

#### 9.9.2 Inicie o processamento normalmente

* Execute a aplicação para consumir/produzir mensagens na fila quorum.
* No admin do RabbitMQ, acompanhe o consumo e o status dos nodes.

#### 9.9.3 Identifique o node backend que está sendo consumido

Para saber qual node RabbitMQ está recebendo a conexão da aplicação via HAProxy:

##### a) Pelo admin do RabbitMQ

* Acesse o Management UI (porta 15672 ou 15671 via HAProxy).
* Vá em “Connections” e procure pela conexão da aplicação (pode filtrar pelo IP do HAProxy ou do cliente).
* Clique na conexão e veja o campo “Node” — este é o node backend que está atendendo a aplicação.

##### b) Pelo log do HAProxy

* No node HAProxy, rode: bash sudo journalctl -u haproxy -f
* Inicie a aplicação e observe qual backend está sendo selecionado para cada nova conexão.

##### c) Pela fila quorum

* No admin do RabbitMQ, ao clicar na fila quorum, veja o campo “Leader” e os “Members”. O leader é quem processa as escritas, mas as conexões podem ser atendidas por qualquer membro.

#### 9.9.4 Teste de failover

1. Com a aplicação processando normalmente, identifique o node backend que está atendendo a conexão.
2. Desligue (ou pare o serviço RabbitMQ) neste node específico: bash sudo systemctl stop rabbitmq-server
3. Observe se a aplicação reconecta automaticamente e se o HAProxy redireciona para outro node online.
4. No admin do RabbitMQ, verifique se a fila quorum permanece acessível e se outro node assume o papel de leader (em caso de escrita).

#### 9.9.5 Dicas adicionais

* Repita o teste desligando outros nodes para validar o balanceamento e a resiliência.
* Sempre monitore os logs do HAProxy e do RabbitMQ para identificar possíveis problemas de conexão ou failover.
* Certifique-se de que a aplicação está configurada para reconectar automaticamente em caso de falha de conexão.

## Conclusão

Este guia apresentou um processo completo de instalação e configuração do RabbitMQ em ambiente corporativo RHEL 9 air-gapped, abrangendo desde a instalação offline dos componentes até a configuração de alta disponibilidade com HAProxy.

### Pontos principais cobertos:

* **Instalação offline** do Erlang e RabbitMQ
* **Configuração de cluster** com 5 nós
* **Implementação de segurança** com TLS
* **Monitoramento** com Prometheus
* **Alta disponibilidade** com HAProxy
* **Testes de failover** e validação

### Próximos passos recomendados:

* Implementar backups regulares das configurações
* Configurar alertas proativos para monitoramento
* Documentar procedimentos operacionais específicos do ambiente
* Treinar equipe operacional nos procedimentos de manutenção

**Documento elaborado para ambiente corporativo**  
**Versão:** 1.0  
**Última atualização:** Setembro 2025