

Development with RabbitMQ

- [Bash](#)
- [Python](#)
- [PHP](#)
- [dotNet](#)
- [AMQP Client](#)

Bash

curl

```
curl -u login:pass -i -H "content-type:application/json" -X POST
http://localhost:15672/api/exchanges/%2Fvhost/exchange/publish \
-d '{"properties":{"routing_key":"","payload":"you message","payload_encoding":"string"}}'
```

rabbitmqadmin

`rabbitmqadmin` uses HTTP API authentication mechanism (basic HTTP authentication).

- [Command Line Tools](#)
- [Management Command Line Tool](#)

Install

```
# Install the dependency
yum install python3

# Download the script from the RabbitMQ Server
wget http://<rabbitmq-server-hostname>:15672/cli/rabbitmqadmin

#
chmod 0755 rabbitmqadmin
mv rabbitmqadmin /usr/local/bin
rabbitmqadmin -h

# Verify the connection via HTTP API
rabbitmqadmin -H <rabbitmq-server-hostname> -u <username> -p <password> list vhosts
```

Basic Operation

```
## 🐧 Linux 🐧
# 🐧 queue
rabbitmqadmin -H <rabbitmq-server-hostname> -u <username> -p <password> -V <vhost-name> declare
queue name=my-testq durable=true
```

```
# 1000
```

```
rabbitmqadmin -H <rabbitmq-server-hostname> -u <username> -p <password> -V <vhost-name> publish  
exchange=amq.default routing_key=my-testq payload="This is Alang"
```

```
# 1000
```

```
rabbitmqadmin -H <rabbitmq-server-hostname> -u <username> -p <password> -V <vhost-name> get  
queue=my-testq ackmode=ack_requeue_false
```

```
# 1000, 1000 tsv
```

```
rabbitmqadmin -H <rabbitmq-server-hostname> -u <username> -p <password> -V <vhost-name> -f tsv get  
queue=my-testq ackmode=ack_requeue_false
```

```
# 1000 5 10
```

```
rabbitmqadmin -H <rabbitmq-server-hostname> -u <username> -p <password> -V <vhost-name> get  
queue=my-testq count=5 ackmode=ack_requeue_false
```

```
while read -r line; do
```

```
    echo $line | rabbitmqadmin publish exchange=amq.default routing_key=my_queue ;
```

```
done < messages
```

```
rabbitmqadmin publish exchange=amq.default routing_key=test payload="hello, world"
```

```
# With parallel
```

```
cat messages | parallel -j 100 \
```

```
    ./rabbitmqadmin -H $RABBITMQ_HOST \
```

```
        -u $RABBITMQ_USERNAME \
```

```
        -p $RABBITMQ_PASSWORD \
```

```
        publish exchange=amq.default \
```

```
        routing_key=myqueue \
```

```
        payload="{ }"
```

Python

Tutorials

- [python rabbitmq Code Example \(codegrepper.com\)](#)
- [Part 2.3: Getting started with RabbitMQ and Python - CloudAMQP](#)
- [??Python??????RabbitMQ????? | ????? \(codertw.com\)](#)
- [Python ?? RabbitMQ ????????? Python ???_Python_AlwaysBeta_InfoQ????](#)
- [RedHat AMQ Python Client](#)

RedHat AMQ Python Client

Install via RHN

```
[root@dotnetdev ~]# dnf repolist all | grep amq-client
```

amq-clients-2-for-rhel-8-x86_64-debug-rpms	disabled
amq-clients-2-for-rhel-8-x86_64-rpms	disabled
amq-clients-2-for-rhel-8-x86_64-source-rpms	disabled
amq-clients-2.9-for-rhel-8-x86_64-debug-rpms	disabled
amq-clients-2.9-for-rhel-8-x86_64-rpms	disabled
amq-clients-2.9-for-rhel-8-x86_64-source-rpms	disabled

```
[root@dotnetdev ~]# subscription-manager repos --enable=amq-clients-2-for-rhel-8-x86_64-rpms
Repository 'amq-clients-2-for-rhel-8-x86_64-rpms' is enabled for this system.
```

```
[root@dotnetdev ~]# dnf repolist
```

Updating Subscription Management repositories.

repo id	repo name
amq-clients-2-for-rhel-8-x86_64-rpms	Red Hat AMQ Clients 2 for RHEL 8 x86_64 (RPMs)
rhel-8-for-x86_64-appstream-rpms	Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
rhel-8-for-x86_64-baseos-rpms	Red Hat Enterprise Linux 8 for x86_64 - BaseOS (RPMs)

```
[root@dotnetdev ~]# yum install python3-qpid-proton python-qpid-proton-docs
```

PHP

Tutorials

- [how to call a model from rabbitmq php consumer's callback in codeigniter? \(google.com\)](#)

dotNet

Tutorials

- [GitHub] [AMQP.Net Lite](#)
- [RedHat AMQ .Net Client](#)
- [C# ??? RabbitMQ Cluster - ?? RabbitMQ .Net Client ? EasyNetQ](#)
- [GitHub] [EasyNetQ](#)

Case: Console RabbitMQ Client

[ConsoleRabbitMQ.zip](#)

Package Required:

- RabbitMQ.Client

Target Framework:

- net6.0

Platform Supported:

- Windows 64
- RedHat Linux 8 64 (net6 runtime is required)

AMQP Client

Terms & Concepts

Queue

- prefetch: Consumer ?????? consumer ?????? PUSH ??????????????????
consumer ? PUSH ??????????(????????)?????? consumer
???????????????????? Unacked ?????? Unacked ??????????????????
?????????consumer ???? prefetch ?????????? consumer ??????
- Lazy Queue: Queue ?????????(?)???????????? queue????????????????????

Consume Message

- Consumer ?????????? Push (??) ? Poll (?) ?????
 - [????RabbitMQ\(?\)——???????QOS?C#?? - ?????? - ??? \(cnblogs.com\)](#)

Publish Message

- ??? nodes ? Cluster
???????????????????????????? nodes????????????????
- Publish Confirm:

TTL (Time to Live)

```

????????????????????????????????????? TTL?? RabbitMQ ? TTL ???? Message ? Queue
???
```

Message TTL

- `??: ??????????????????`
- `?? Message ?????? Queues ?????? Queue ? TTL ?????????????????????? TTL ? Queue???????????????? ?`
- `???: Millisecond (60 seconds = 60000)`

???: Policy

```
rabbitmqctl set_policy TTL ".*" '{"message-ttl":60000}' --apply-to queues
```

???: Queue ??

- x-message-ttl: 60000

Sample codes in C#

```
var args = new Dictionary<string, object>();
args.Add("x-message-ttl", 60000);
model.QueueDeclare("myqueue", false, false, false, args);
```

Queue TTL

- ?? : ???????? Queue?? Queue ??????
- ??? : Millisecond (30 mins = 1800000)

??? : Policy

```
rabbitmqctl set_policy expiry ".*" '{"expires":1800000}' --apply-to queues
```

??? : Queue ??

- x-expires: 1800000

Sample codes in Java

```
Map<String, Object> args = new HashMap<String, Object>();
args.put("x-expires", 1800000);
channel.queueDeclare("myqueue", false, false, false, args);
```

rabbitmqadmin

- [Management Command Line Tool — RabbitMQ](#)

Usage

```
# Publish a message
rabbitmqadmin -H <rabbitmq-server-ip> -u <user-name> -p <secret> -V <virtual-server> publish
exchange=amq.default routing_key=my-testq payload="This is Alang"

# Consume/Get a message
rabbitmqadmin -H <rabbitmq-server-ip> -u <user-name> -p <secret> -V <virtual-host> get queue=my-testq
ackmode=ack_requeue_false
```

amqp-tools

A CLI tool is built-in Ubuntu.

Install

```
sudo apt update
sudo apt install amqp-tools
```

Usage

```
# Declare a queue
amqp-declare-queue --url="amqp://<user-name>:<secret>@<rabbitmq-server-ip>:<rabbitmq-server-
port>/<virtual-server>" -d -q "my-testq"

# Publish a message
amqp-publish --url="amqp://<user-name>:<secret>@<rabbitmq-server-ip>:<rabbitmq-server-port>/<virtual-
server>" --routing-key="my-testq" -b "Hello,World"

# Get the messages (Poll mode)
amqp-get --url="amqp://<user-name>:<secret>@<rabbitmq-server-ip>:<rabbitmq-server-port>/<virtual-
server>" --queue="my-testq"

# Get the messages (Push mode)
amqp-consume --url="amqp://<user-name>:<secret>@<rabbitmq-server-ip>:<rabbitmq-server-port>/<virtual-
server>" --queue="my-testq" -p 2 ./show.sh
```

show.sh:

```
#!/usr/bin/env bash
read line
echo "Message: $line"
sleep 1
```