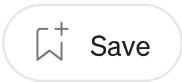




Francesco Bonizzi [Follow](#)

Sep 25, 2019 · 3 min read · [Listen](#)



# RabbitMQ with Docker on Windows in 30 minutes

How to set up a RabbitMQ instance with Docker on Windows and start sending messages.



Want to quickly spin up [RabbitMQ](#) on your Windows development environment? Maybe you just want to try the most deployed message broker and you don't want to try it on Azure or ask your IT crew to deploy a new server just for that: [Docker](#) is your solution.

1. [Install Docker for Windows](#): just download it and let the installer do the rest



426



6

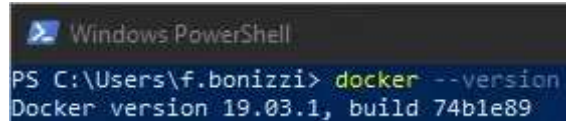
2. Start Docker and wait for its the initialization. You will see an icon of a whale in the taskbar

3. Open Powershell and type:

```
1 docker --version
```

medium-article-docker1.ps1 hosted with ❤️ by GitHub

[view raw](#)



```
Windows PowerShell
PS C:\Users\f.bonizzi> docker --version
Docker version 19.03.1, build 74b1e89
```

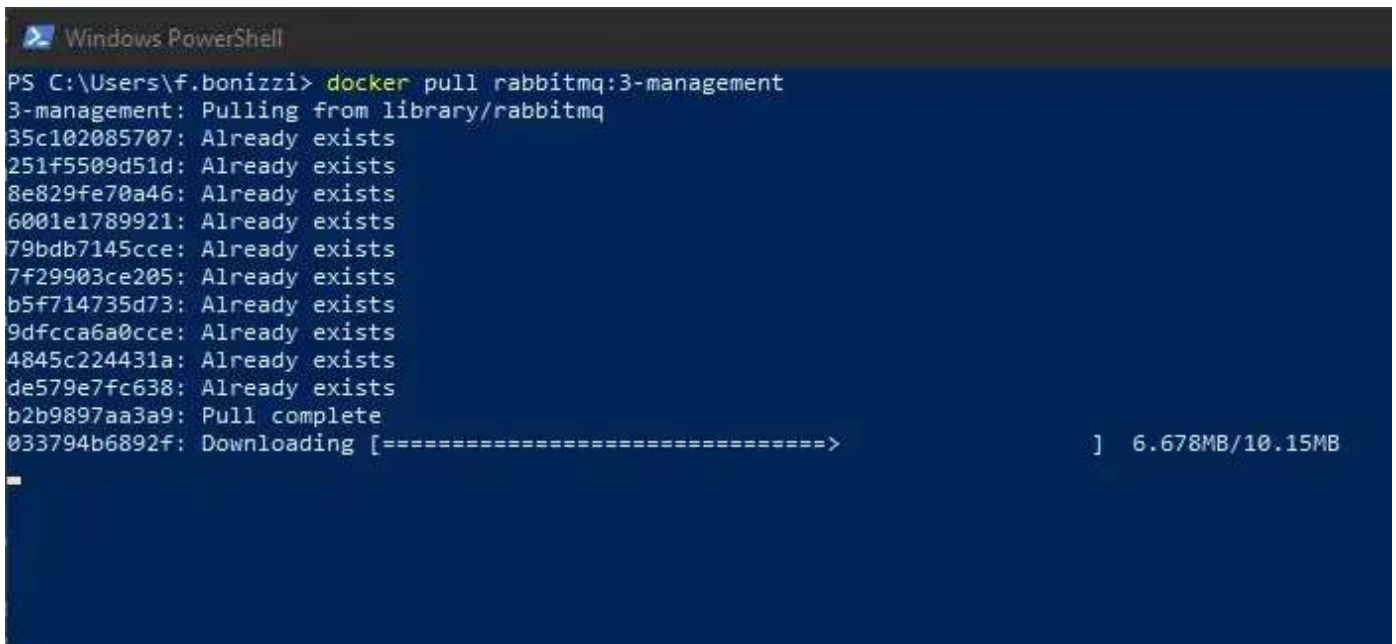
If you don't see any errors, and you see a Docker version, it is correctly installed.

4. Now we have to download the RabbitMQ image. In this case, we will download the image with a management plugin in order to be able to view the RabbitMQ server-side GUI. The command to download or update a Docker image is `pull`

```
1 docker pull rabbitmq:3-management
```

medium-article-docker2.ps1 hosted with ❤️ by GitHub

[view raw](#)



```
Windows PowerShell
PS C:\Users\f.bonizzi> docker pull rabbitmq:3-management
3-management: Pulling from library/rabbitmq
35c102085707: Already exists
251f5509d51d: Already exists
8e829fe70a46: Already exists
6001e1789921: Already exists
79bdb7145cce: Already exists
7f29903ce205: Already exists
b5f714735d73: Already exists
9dfcca6a0cce: Already exists
4845c224431a: Already exists
de579e7fc638: Already exists
b2b9897aa3a9: Pull complete
033794b6892f: Downloading [=====>] 6.678MB/10.15MB
```

5. Now we will start the RabbitMQ Docker image with a simple command:

```
1 docker run -d -p 15672:15672 -p 5672:5672 --name rabbit-test-for-medium rabbitmq:3-management
```

medium-article-docker3.ps1 hosted with ❤️ by GitHub

[view raw](#)

With the `p` argument we are mapping RabbitMQ ports to Docker container ports. 15672 is the default port for RabbitMQ GUI, 5672 for RabbitMQ message broker. With the `name` argument we are giving a name to our container, in order to identify it in a more readable way than using the generated GUID. This will allow us to easily stop, remove, and manage our containers. At the end, we specify the docker image to run, in this case the one we pulled before.

If the operation goes well, you will see a GUID as output.

6. Test the image by opening “<http://localhost:15672/#/>” in your browser. The default login is `guest` `guest`: you should see the RabbitMQ management GUI.



The screenshot shows the RabbitMQ management interface. At the top, there's a navigation bar with tabs for Overview, Connections, Channels, Exchanges, Queues, and Admin. The Overview page displays several key metrics: Overview, Totals, Queued messages (1000000), Currently idle, Message rates (last minute), and Global counts (Connections: 0, Channels: 0, Exchanges: 7, Queues: 0, Consumers: 0). Below these, there's a 'Nodes' section with a table of node statistics. The table has columns for Name, File descriptors, Socket descriptors, Erlang processes, Memory, Disk space, Uptime, Info, and Reset stats. The first node is 'rabbit@15672f7d1a832d1' with 31 file descriptors, 9 socket descriptors, 166 Erlang processes, 33716 Memory, 14200 Disk space, 7m 4s Uptime, and is in a 'running' state. Below the table are links for Chain statistics, Ports and contacts, Export definitions, and Import definitions. At the bottom, there's a footer with links for HTTP API, Server Docs, Tutorials, Community Support, Community Slack, Commercial Support, Plugins, GitHub, and Changelog.

The RabbitMQ management GUI hosted by Docker

If you see this page correctly, **the most important part is done!**

7. Now we are ready to send messages to our RabbitMQ endpoint! In this phase the choice is yours: I work daily with .NET so I will show you a C# example. Just to remind you, the default values to connect to the endpoint with **your preferred language**:

- HostName: "localhost"
- UserName: "guest"
- Password: "guest"
- Port: 5672

In my case, I will create a new Windows Console Application, install [RabbitMQ.Client](#) Nuget package and write some lines of code to create a queue and send a message every 500 milliseconds:

```
1 using RabbitMQ.Client;
2 using System;
3 using System.Text;
4 using System.Threading.Tasks;
5
6 namespace TestRabbit
7 {
8     internal static class Program
9     {
10         static async Task Main(string[] args)
11         {
12             const string queueName = "testqueue";
13
14             try
15             {
16                 var connectionFactory = new ConnectionFactory()
17                 {
18                     HostName = "localhost",
19                     UserName = "guest",
20                     Password = "guest",
21                     Port = 5672,
22                     RequestedConnectionTimeout = 3000, // milliseconds
23                 };
24
25                 using (var rabbitConnection = connectionFactory.CreateConnection())
26                 {
27                     using (var channel = rabbitConnection.CreateModel())
28                     {
29                         // Declaring a queue is idempotent
30                         channel.QueueDeclare(
31                             queue: queueName,
32                             durable: false,
33                             exclusive: false,
34                             autoDelete: false,
35                             arguments: null);
36
37                         while (true)
38                         {
39                             string body = $"A nice random message: {DateTime.Now.Ticks}";
40                             channel.BasicPublish(
41                                 exchange: string.Empty,
42                                 routingKey: queueName,
43                                 basicProperties: null,
44                                 body: Encoding.UTF8.GetBytes(body));
45
46                             // ... (rest of the code)
47                         }
48                     }
49                 }
50             }
51         }
52     }
53 }
```

```

45
46         Console.WriteLine("Message sent");
47         await Task.Delay(500);
48     }
49 }
50 }
51 }
52 catch (Exception ex)
53 {
54     Console.ForegroundColor = ConsoleColor.Red;
55     Console.WriteLine(ex.ToString());
56     Console.ForegroundColor = ConsoleColor.White;
57 }
58
59 Console.WriteLine("End");
60 Console.Read();
61 }
62 }
63 }

```

medium-article-docker4.cs hosted with ❤️ by GitHub

[view raw](#)

Message rates **last minute** ?



Publish	2.0/s
Deliver (manual ack)	0.00/s
Deliver (auto ack)	0.00/s

RabbitMQ management GUI charts when sending messages to a queue

Also, you can click on the `Get messages` menu to grab a message from the queue and read its body:

## ▼ Get messages

Warning: getting messages from a queue is a destructive action. ?

Ack Mode:  ▼

Encoding:  ▼ ?

Messages:

**Get Message(s)**

Message 1

The server reported 141 messages remaining.

Exchange	(AMQP default)
Routing Key	testqueue
Redelivered	•
Properties	
Payload	A nice random message: 637042498285032163
<small>41 bytes</small>	
<small>Encoding: string</small>	

The body of a RabbitMQ message in our example

I don't have the experience to say how Docker is on production environments, but in test/study environments, I think that Docker is really a **time saving infrastructure!**

Docker

Rabbitmq

Programming

Csharp

Open Source

---

## Get an email whenever Francesco Bonizzi publishes.

By signing up, you will create a Medium account if you don't already have one.

Review our [Privacy Policy](#) for more information about our privacy practices.

 **Subscribe**

[About](#) [Help](#) [Terms](#) [Privacy](#)

Get the Medium app

