Cloud Computing Homework4 -- Message Queue

Name	ID	Class
Qing Liu	6130116184	Class 164 of Computer Science and Technology

Run the rabbitmq in docker

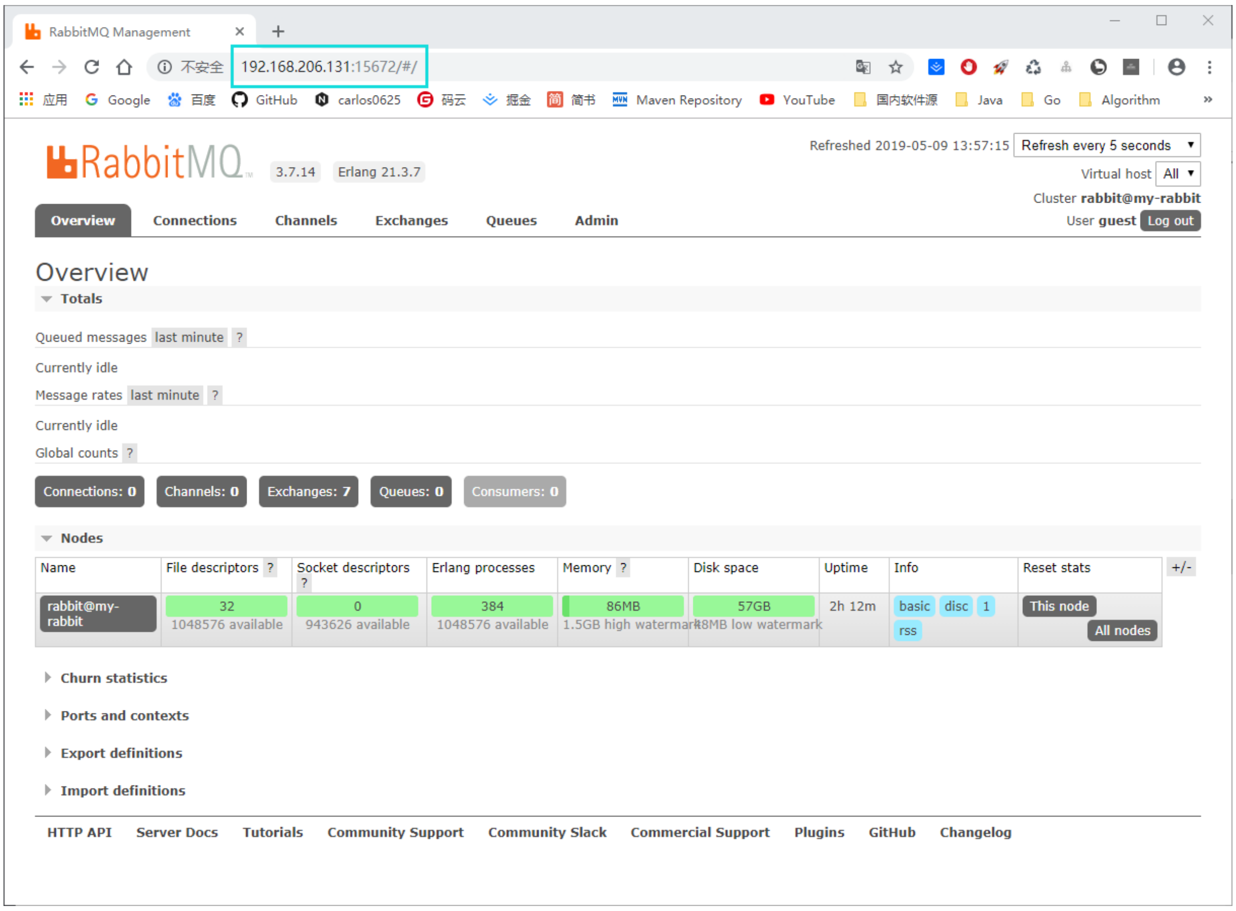
- Pull the image from docker hub

```
docker pull rabbitmq:management
```

- Run a rabbitmq container

```
docker run -d --hostname my-rabbit \
-p 5671:5671 -p 5672:5672 -p 4369:4369 \
-p 25672:25672 -p 15671:15671 \
-p 15672:15672  \
--name okong-rabbit -d rabbitmq:management
```

- Visit the `<ip>:15672` in chrome of host


RabbitMQ Management
192.168.206.131:15672/#/
应用 Google 百度 GitHub carlos0625 码云 掘金 简书 Maven Repository YouTube 国内软件源 Java Go Algorithm
RabbitMQ 3.7.14 Erlang 21.3.7
Overview Connections Channels Exchanges Queues Admin

Refreshed 2019-05-09 13:57:15 Refresh every 5 seconds
Virtual host All
Cluster rabbit@my-rabbit
User guest Log out

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@my-rabbit	32 1048576 available	0 943626 available	384 1048576 available	86MB 1.5GB high watermark	57GB 48MB low watermark	2h 12m	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

Write the go scripts, build image and run the containers

- Write the script sending messages in go programming language and dockerfile

`send.go`:

```go
// send.go

package main

import (
  "fmt"
  "github.com/streadway/amqp"
  "log"
  "strconv"
)

func failOnError(err error, msg string) {
  if err != nil {
    // Exit the program.
    panic(fmt.Sprintf("%s: %s", msg, err))
  }
}

func main() {
  //create the connection
  conn, err := amqp.Dial("amqp://guest:guest@192.168.206.131:5672/")
```

```
    failOnError(err, "Error connecting to the broker")
    defer conn.Close()

    //get the channel
    ch, err := conn.Channel()
    failOnError(err, "Failed to open a channel")
    defer ch.Close()


    q, err := ch.QueueDeclare(
      "hello",
      false,
      false,
      false,
      false,
      nil,
      )
    failOnError(err, "Failed to declare a queue")

    // publisher
    for i := 0; i < 100; i++ {
      body := "hello" + strconv.Itoa(i)
      err = ch.Publish(
        "",
        q.Name,
        false,
        false,
        amqp.Publishing{
          ContentType: "text/plain",
          Body: []byte(body),
        })
      log.Printf(" [x] Sent %s", body)
    }

    failOnError(err, "Failed to publish a message")
  }
```

`Dockerfile`:

```
FROM golang:latest

WORKDIR /go/src/send
COPY . /go/src/send

RUN go get -v github.com/streadway/amqp
RUN go build .

EXPOSE 8000

ENTRYPOINT ["./send"]
```

- Write the script receiving messages in go programming language and dockerfile

  `receive.go`:

```go
// receive.go

package main

import (
  "fmt"
  "github.com/streadway/amqp"
  "log"
)

func failOnError(err error, msg string) {
  if err != nil {
    // Exit the program.
    panic(fmt.Sprintf("%s: %s", msg, err))
  }
}

func main() {
  //create the connection
  conn, err := amqp.Dial("amqp://guest:guest@192.168.206.131:5672/")
  failOnError(err, "Error connecting to the broker")
  defer conn.Close()

  //get the channel
  ch, err := conn.Channel()
  failOnError(err, "Failed to open a channel")
  defer ch.Close()

  q, err := ch.QueueDeclare(
    "hello",
    false,
    false,
    false,
    false,
    nil)
  failOnError(err, "Failed to declare a queue")

  //consumer
  msgs, err := ch.Consume(
    q.Name,
    "",
    true,
    false,
    false,
    false,
    nil)
  failOnError(err, "Failed to register a consumer")

  forever := make(chan bool)
```

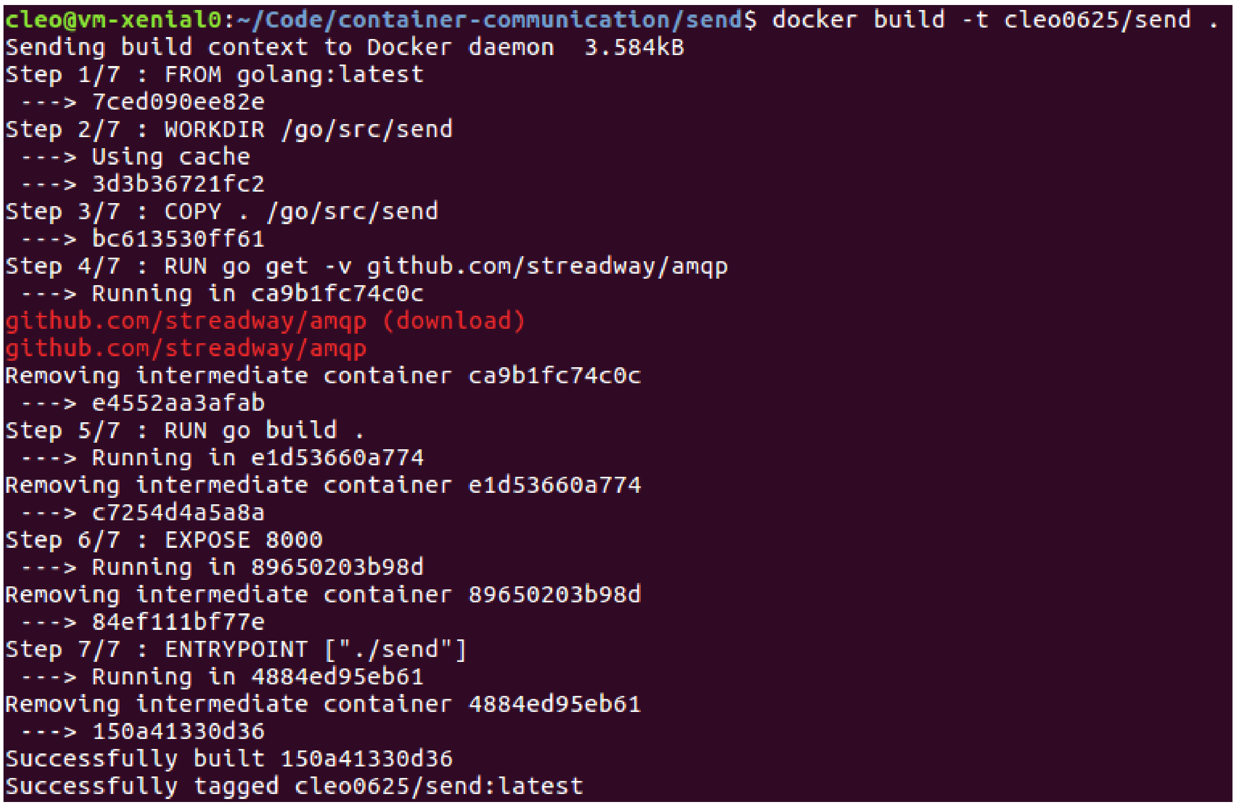
```
    go func() {
      for d := range msgs {
        log.Printf("Received a message: %s", d.Body)
      }
    }()

    log.Printf(" [*] Waiting for messages. To exit press CTRL+C")
    <-forever
  }
```

`dockerfile`:

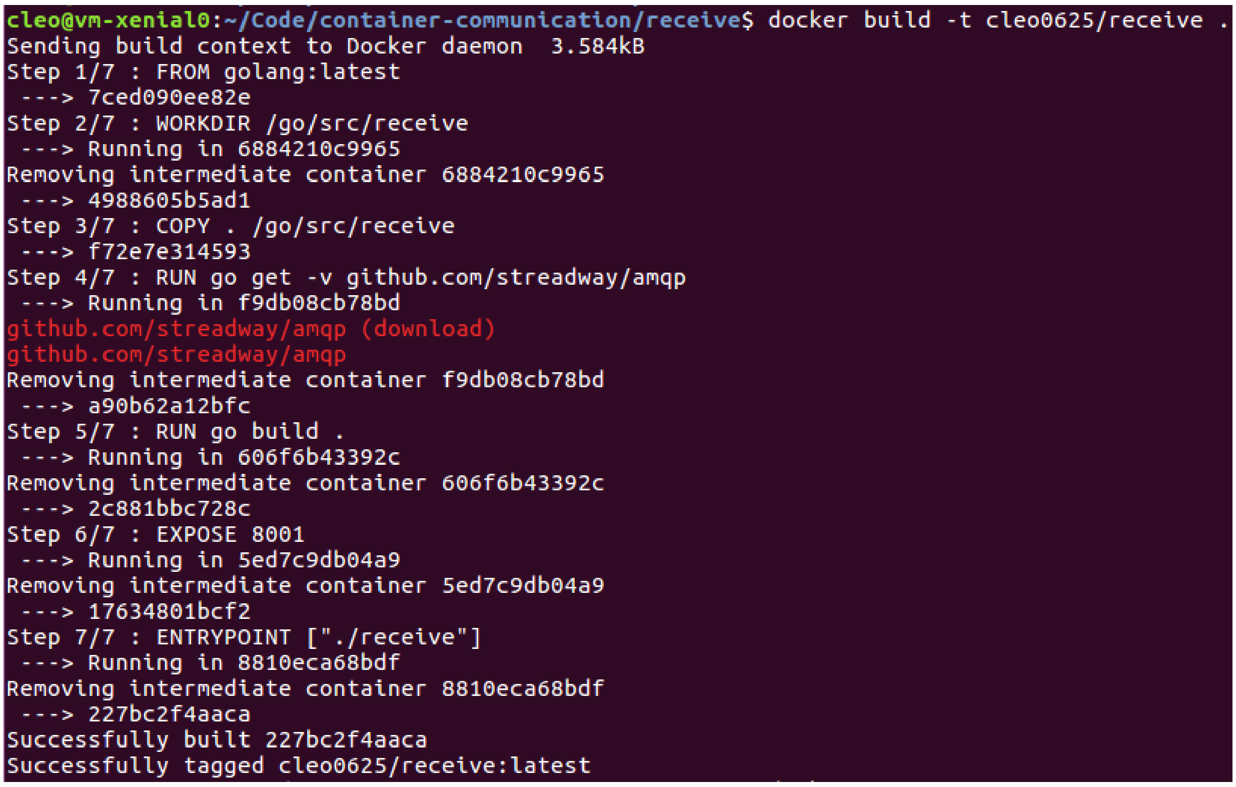
```
FROM golang:latest

WORKDIR /go/src/receive
COPY . /go/src/receive

RUN go get -v github.com/streadway/amqp
RUN go build .

EXPOSE 8001

ENTRYPOINT ["./receive"]
```

- Build the two programs into images

```
cd ~/Code/container-communication/send
docker build -t cleo0625/send .
cd ../receive
docker build -t cleo0625/receive .
```



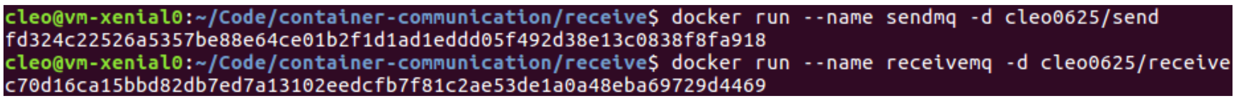
```
cleo@vm-xenial0:~/Code/container-communication/send$ docker build -t cleo0625/send .
Sending build context to Docker daemon  3.584kB
Step 1/7 : FROM golang:latest
 ---> 7ced090ee82e
Step 2/7 : WORKDIR /go/src/send
 ---> Using cache
 ---> 3d3b36721fc2
Step 3/7 : COPY . /go/src/send
 ---> bc613530ff61
Step 4/7 : RUN go get -v github.com/streadway/amqp
 ---> Running in ca9b1fc74c0c
github.com/streadway/amqp (download)
github.com/streadway/amqp
Removing intermediate container ca9b1fc74c0c
 ---> e4552aa3afab
Step 5/7 : RUN go build .
 ---> Running in e1d53660a774
Removing intermediate container e1d53660a774
 ---> c7254d4a5a8a
Step 6/7 : EXPOSE 8000
 ---> Running in 89650203b98d
Removing intermediate container 89650203b98d
 ---> 84ef111bf77e
Step 7/7 : ENTRYPOINT ["./send"]
 ---> Running in 4884ed95eb61
Removing intermediate container 4884ed95eb61
 ---> 150a41330d36
Successfully built 150a41330d36
Successfully tagged cleo0625/send:latest
```



```
cleo@vm-xenial0:~/Code/container-communication/receive$ docker build -t cleo0625/receive .
Sending build context to Docker daemon  3.584kB
Step 1/7 : FROM golang:latest
 ---> 7ced090ee82e
Step 2/7 : WORKDIR /go/src/receive
 ---> Running in 6884210c9965
Removing intermediate container 6884210c9965
 ---> 4988605b5ad1
Step 3/7 : COPY . /go/src/receive
 ---> f72e7e314593
Step 4/7 : RUN go get -v github.com/streadway/amqp
 ---> Running in f9db08cb78bd
github.com/streadway/amqp (download)
github.com/streadway/amqp
Removing intermediate container f9db08cb78bd
 ---> a90b62a12bfc
Step 5/7 : RUN go build .
 ---> Running in 606f6b43392c
Removing intermediate container 606f6b43392c
 ---> 2c881bbc728c
Step 6/7 : EXPOSE 8001
 ---> Running in 5ed7c9db04a9
Removing intermediate container 5ed7c9db04a9
 ---> 17634801bcf2
Step 7/7 : ENTRYPOINT ["./receive"]
 ---> Running in 8810eca68bdf
Removing intermediate container 8810eca68bdf
 ---> 227bc2f4aaca
Successfully built 227bc2f4aaca
Successfully tagged cleo0625/receive:latest
```

- Run send container and receive container

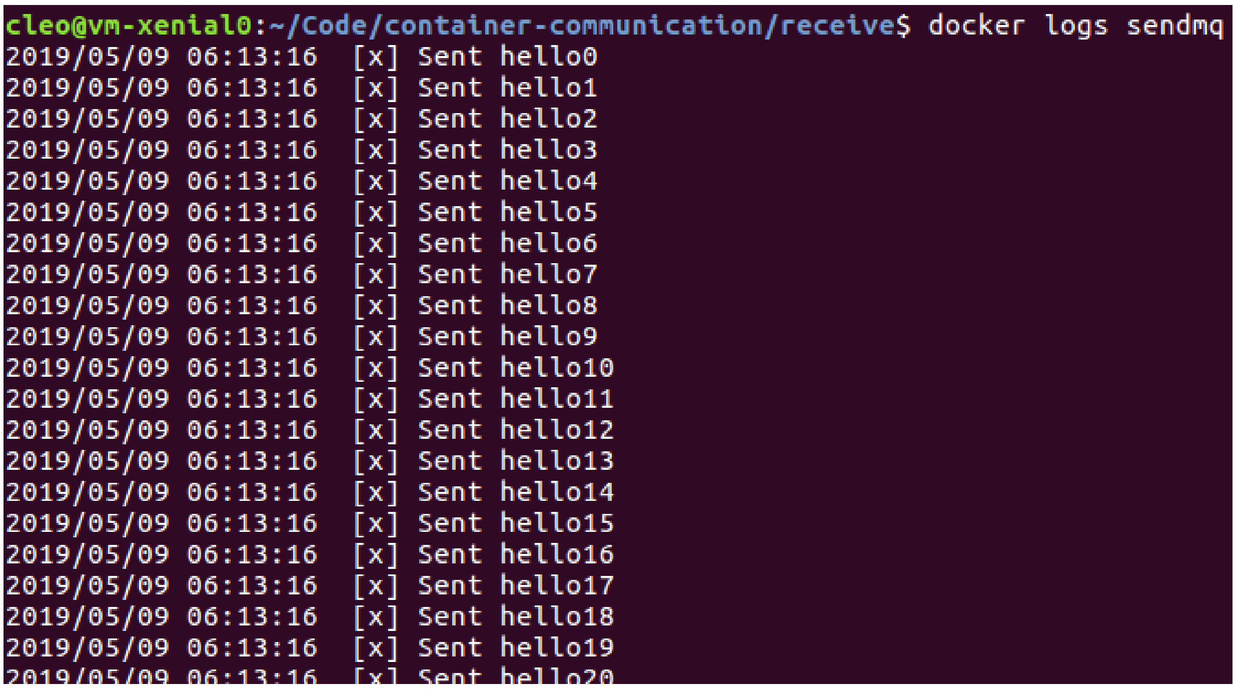
```
docker run --name sendmq -d cleo0625/send
docker run --name receivemq -d cleo0625/receive
```



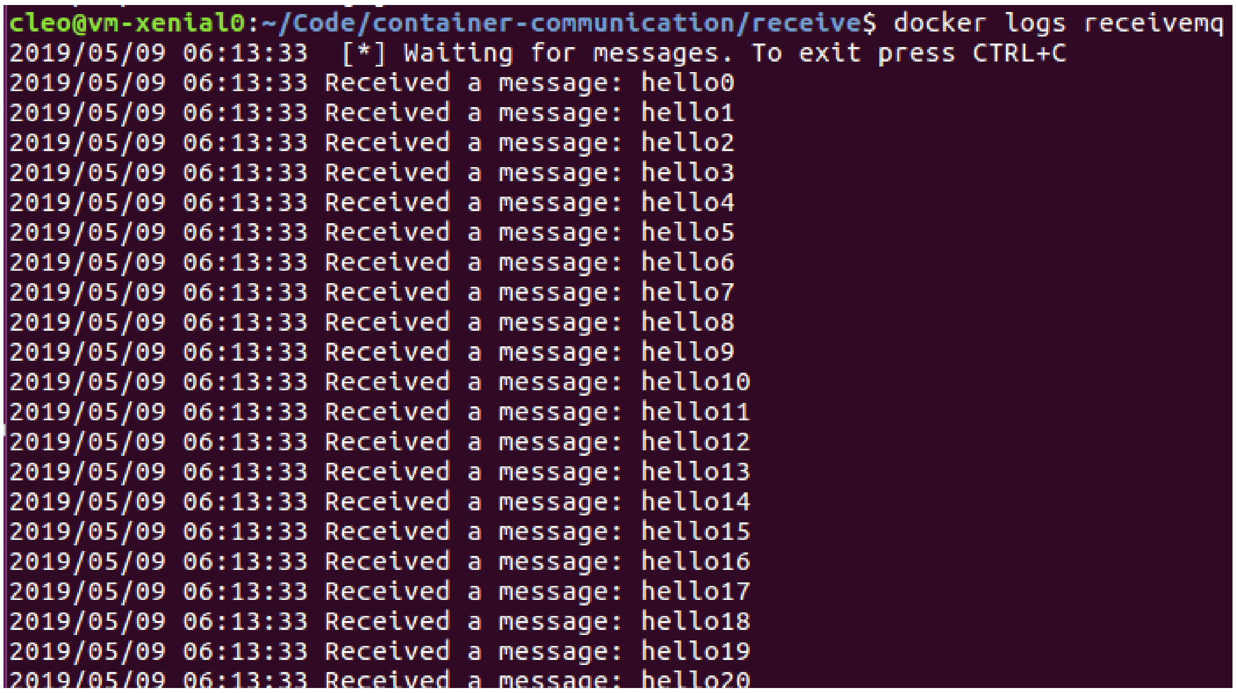
```
cleo@vm-xenial0:~/Code/container-communication/receive$ docker run --name sendmq -d cleo0625/send
fd324c22526a5357be88e64ce01b2f1d1ad1eddd05f492d38e13c0838f8fa918
cleo@vm-xenial0:~/Code/container-communication/receive$ docker run --name receivemq -d cleo0625/receive
c70d16ca15bbd82db7ed7a13102eedcfb7f81c2ae53de1a0a48eba69729d4469
```

- Check the logs of those two containers

```
docker logs sendmq
docker logs receivemq
```



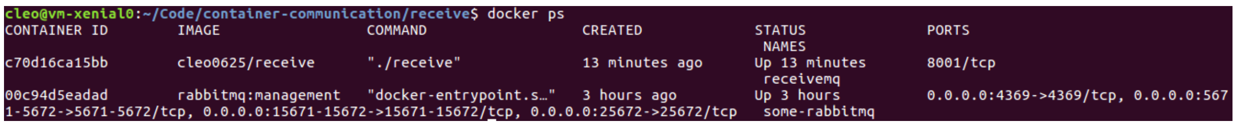
```
cleo@vm-xenial0:~/Code/container-communication/receive$ docker logs sendmq
2019/05/09 06:13:16  [x] Sent hello0
2019/05/09 06:13:16  [x] Sent hello1
2019/05/09 06:13:16  [x] Sent hello2
2019/05/09 06:13:16  [x] Sent hello3
2019/05/09 06:13:16  [x] Sent hello4
2019/05/09 06:13:16  [x] Sent hello5
2019/05/09 06:13:16  [x] Sent hello6
2019/05/09 06:13:16  [x] Sent hello7
2019/05/09 06:13:16  [x] Sent hello8
2019/05/09 06:13:16  [x] Sent hello9
2019/05/09 06:13:16  [x] Sent hello10
2019/05/09 06:13:16  [x] Sent hello11
2019/05/09 06:13:16  [x] Sent hello12
2019/05/09 06:13:16  [x] Sent hello13
2019/05/09 06:13:16  [x] Sent hello14
2019/05/09 06:13:16  [x] Sent hello15
2019/05/09 06:13:16  [x] Sent hello16
2019/05/09 06:13:16  [x] Sent hello17
2019/05/09 06:13:16  [x] Sent hello18
2019/05/09 06:13:16  [x] Sent hello19
2019/05/09 06:13:16  [x] Sent hello20
```



```
cleo@vm-xenial0:~/Code/container-communication/receive$ docker logs receivemq
2019/05/09 06:13:33  [*] Waiting for messages. To exit press CTRL+C
2019/05/09 06:13:33 Received a message: hello0
2019/05/09 06:13:33 Received a message: hello1
2019/05/09 06:13:33 Received a message: hello2
2019/05/09 06:13:33 Received a message: hello3
2019/05/09 06:13:33 Received a message: hello4
2019/05/09 06:13:33 Received a message: hello5
2019/05/09 06:13:33 Received a message: hello6
2019/05/09 06:13:33 Received a message: hello7
2019/05/09 06:13:33 Received a message: hello8
2019/05/09 06:13:33 Received a message: hello9
2019/05/09 06:13:33 Received a message: hello10
2019/05/09 06:13:33 Received a message: hello11
2019/05/09 06:13:33 Received a message: hello12
2019/05/09 06:13:33 Received a message: hello13
2019/05/09 06:13:33 Received a message: hello14
2019/05/09 06:13:33 Received a message: hello15
2019/05/09 06:13:33 Received a message: hello16
2019/05/09 06:13:33 Received a message: hello17
2019/05/09 06:13:33 Received a message: hello18
2019/05/09 06:13:33 Received a message: hello19
2019/05/09 06:13:33 Received a message: hello20
```

- After running that two containers, check the containers' status

```
docker ps
```

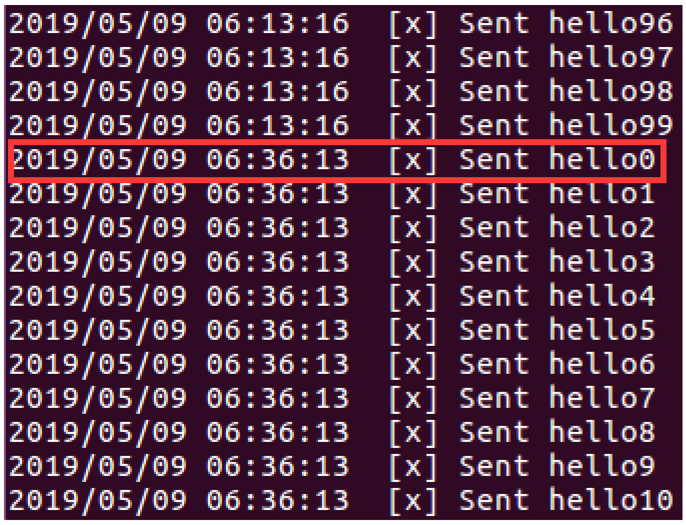


```
cleo@vm-xenial0:~/Code/container-communication/receive$ docker ps
CONTAINER ID        IMAGE                 COMMAND                  CREATED             STATUS              PORTS
                                                                                        NAMES
c70d16ca15bb        cleo0625/receive      "./receive"              13 minutes ago      Up 13 minutes       8001/tcp
                                                                                        receivemq
00c94d5eadad        rabbitmq:management   "docker-entrypoint.s…"   3 hours ago         Up 3 hours          0.0.0.0:4369->4369/tcp, 0.0.0.0:567
1-5672->5671-5672/tcp, 0.0.0.0:15671-15672->15671-15672/tcp, 0.0.0.0:25672->25672/tcp   some-rabbitmq
```

There are only two containers running in docker. One is rabbitmq container, and another one is receivemq container. It is waiting for sending messages.

We can running a new sending messages container to send the new messages to rabbitmq:

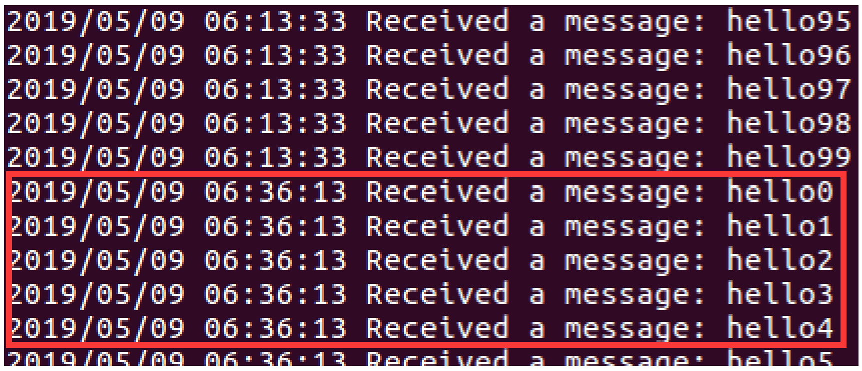
```
docker start sendmq
docker logs sendmq
```



```
2019/05/09 06:13:16  [x] Sent hello96
2019/05/09 06:13:16  [x] Sent hello97
2019/05/09 06:13:16  [x] Sent hello98
2019/05/09 06:13:16  [x] Sent hello99
2019/05/09 06:36:13  [x] Sent hello0
2019/05/09 06:36:13  [x] Sent hello1
2019/05/09 06:36:13  [x] Sent hello2
2019/05/09 06:36:13  [x] Sent hello3
2019/05/09 06:36:13  [x] Sent hello4
2019/05/09 06:36:13  [x] Sent hello5
2019/05/09 06:36:13  [x] Sent hello6
2019/05/09 06:36:13  [x] Sent hello7
2019/05/09 06:36:13  [x] Sent hello8
2019/05/09 06:36:13  [x] Sent hello9
2019/05/09 06:36:13  [x] Sent hello10
```

We can see that new messages had already been sent. And we can log out the logs of receivemq container.

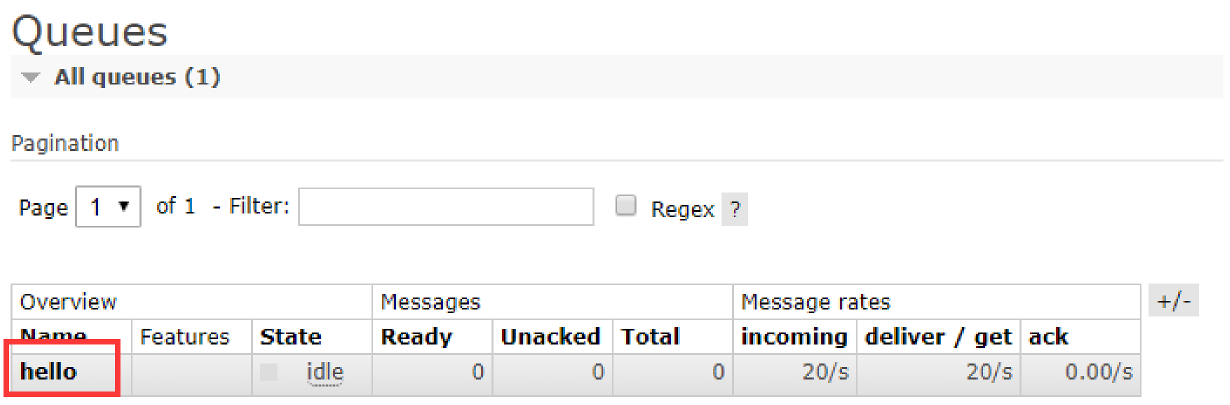
```
docker logs receivemq
```



```
2019/05/09 06:13:33 Received a message: hello95
2019/05/09 06:13:33 Received a message: hello96
2019/05/09 06:13:33 Received a message: hello97
2019/05/09 06:13:33 Received a message: hello98
2019/05/09 06:13:33 Received a message: hello99
2019/05/09 06:36:13 Received a message: hello0
2019/05/09 06:36:13 Received a message: hello1
2019/05/09 06:36:13 Received a message: hello2
2019/05/09 06:36:13 Received a message: hello3
2019/05/09 06:36:13 Received a message: hello4
2019/05/09 06:36:13 Received a message: hello5
```

We can see that the receivemq container had received new messages.

By checking the management web page, we can see the `hello` queue's message rate:


Queues

All queues (1)

Pagination

Page 1 of 1 - Filter: Regex ?

Overview			Messages			Message rates		
Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
hello		idle	0	0	0	20/s	20/s	0.00/s

Run IBM cphtestp

- Clone cphtestp from github

```
git clone https://github.com/ibm-messaging/cphtestp.git
```

- Download the MQ Client seperately and copy the necessary file into root directory

  This step need to create a new IBM account. Selcet `9.1.0.0-IBM-MQC-UbuntuLinuxX64.tar.gz` and download in http mode.

```
cd Downloads
tar -xzvf 9.1.0.0-IBM-MQC-UbuntuLinuxX64.tar.gz
ls
```

  Here is the result of command `ls`

```
9.1.0.0-IBM-MQC-UbuntuLinuxX64.tar.gz  ibmmq-msg-ko_9.1.0.0_amd64.deb
copyright                              ibmmq-msg-pl_9.1.0.0_amd64.deb
ibmmq-bcbridge_9.1.0.0_amd64.deb       ibmmq-msg-pt_9.1.0.0_amd64.deb
ibmmq-client_9.1.0.0_amd64.deb         ibmmq-msg-ru_9.1.0.0_amd64.deb
ibmmq-gskit_9.1.0.0_amd64.deb          ibmmq-msg-zh-cn_9.1.0.0_amd64.deb
ibmmq-java_9.1.0.0_amd64.deb           ibmmq-msg-zh-tw_9.1.0.0_amd64.deb
ibmmq-jre_9.1.0.0_amd64.deb            ibmmq-runtime_9.1.0.0_amd64.deb
ibmmq-man_9.1.0.0_amd64.deb            ibmmq-samples_9.1.0.0_amd64.deb
ibmmq-msg-cs_9.1.0.0_amd64.deb         ibmmq-sdk_9.1.0.0_amd64.deb
ibmmq-msg-de_9.1.0.0_amd64.deb         ibmmq-sfbridge_9.1.0.0_amd64.deb
ibmmq-msg-es_9.1.0.0_amd64.deb         lap
ibmmq-msg-fr_9.1.0.0_amd64.deb         licenses
ibmmq-msg-hu_9.1.0.0_amd64.deb         mqlicense.sh
ibmmq-msg-it_9.1.0.0_amd64.deb         Packages.gz
ibmmq-msg-ja_9.1.0.0_amd64.deb         READMES
```

Then, copy `lap/`, `mqlicense.sh`, `ibmmq-client_9.1.0.0_amd64.deb`, `ibmmq-runtime_9.1.0.0_amd64.deb` to root directory

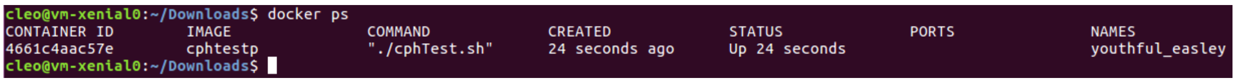
```
cd /home/cleo/Code/cphtestp
cp -r /home/cleo/Downloads/lap .
cp /home/cleo/Downloads/mqlicense.sh .
cp /home/cleo/Downloads/ibmmq-client_9.1.0.0_amd64.deb .
cp /home/cleo/Downloads/ibmmq-runtime_9.1.0.0_amd64.deb .
cp /home/cleo/Downloads/ibmmq-gskit_9.1.0.0_amd64.deb .
```

- Build the image of `cphtestp`

```
docker build --tag cphtestp .
```

- Run the container in network host mode

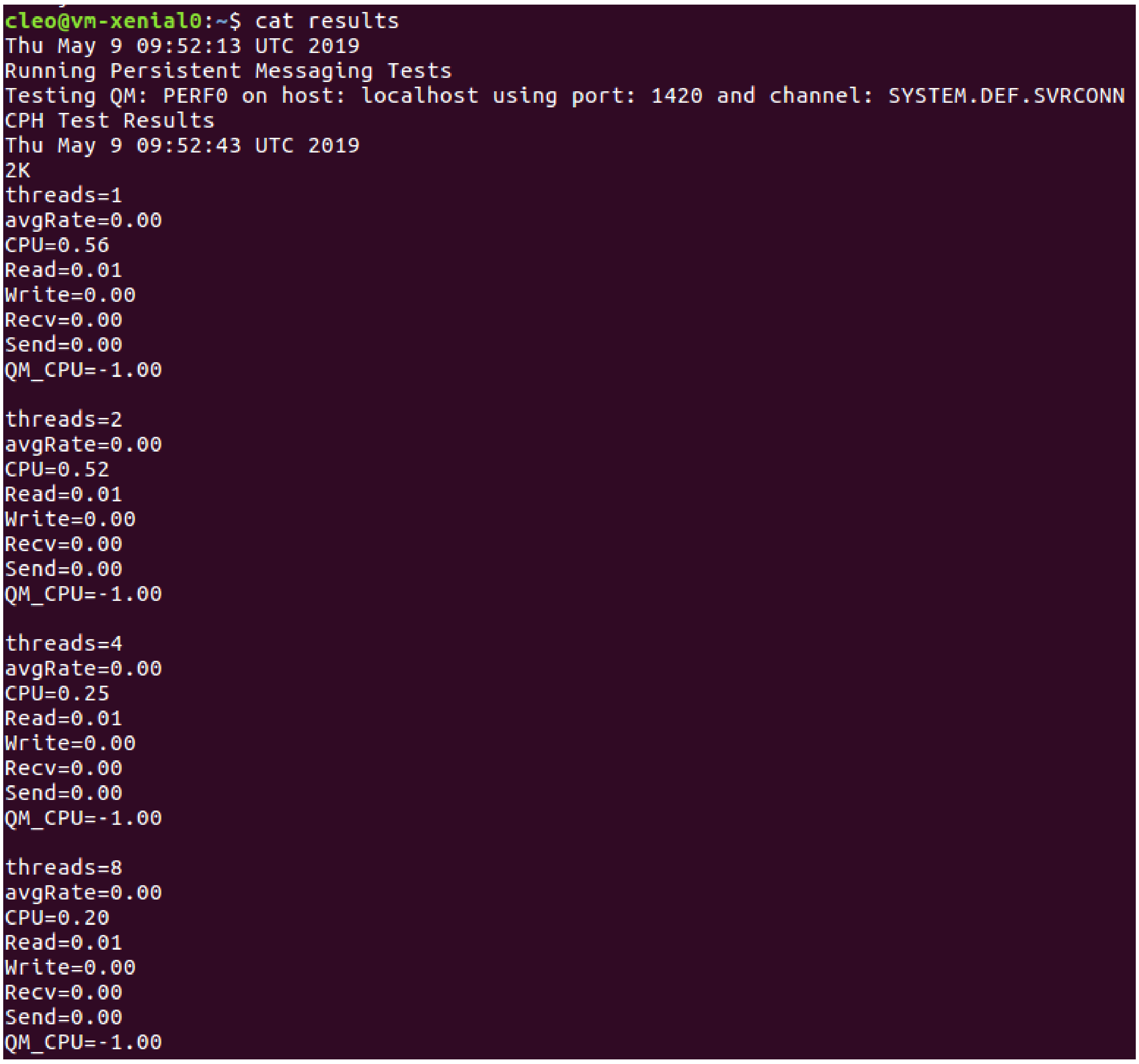
```
docker run -itd --net="host" cphtestp
```



```
cleo@vm-xenial0:~/Downloads$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
4661c4aac57e        cphtestp            "./cphTest.sh"      24 seconds ago      Up 24 seconds                           youthful_easley
cleo@vm-xenial0:~/Downloads$
```

- Obtain the avaliavle results

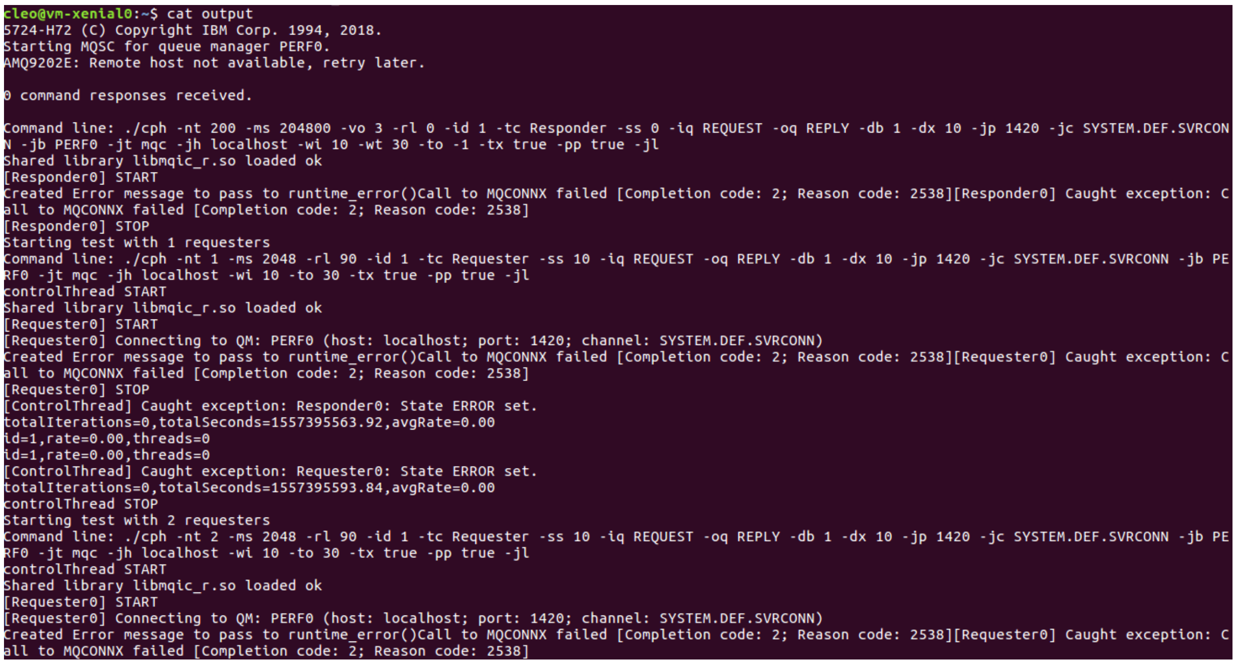
```
docker cp objective_euclid:/home/mqperf/cph/results .
cat result
```



```
cleo@vm-xenial0:~$ cat results
Thu May 9 09:52:13 UTC 2019
Running Persistent Messaging Tests
Testing QM: PERF0 on host: localhost using port: 1420 and channel: SYSTEM.DEF.SVRCONN
CPH Test Results
Thu May 9 09:52:43 UTC 2019
2K
threads=1
avgRate=0.00
CPU=0.56
Read=0.01
Write=0.00
Recv=0.00
Send=0.00
QM_CPU=-1.00

threads=2
avgRate=0.00
CPU=0.52
Read=0.01
Write=0.00
Recv=0.00
Send=0.00
QM_CPU=-1.00

threads=4
avgRate=0.00
CPU=0.25
Read=0.01
Write=0.00
Recv=0.00
Send=0.00
QM_CPU=-1.00

threads=8
avgRate=0.00
CPU=0.20
Read=0.01
Write=0.00
Recv=0.00
Send=0.00
QM_CPU=-1.00
```

- View the output from the running responder and requester processes

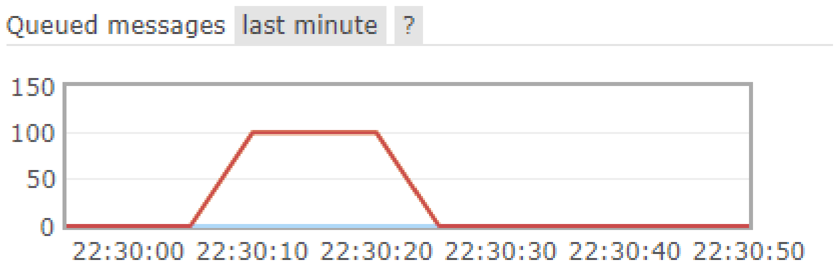
```
docker cp objective_euclid:/home/mqperf/cph/output .
cat output
```

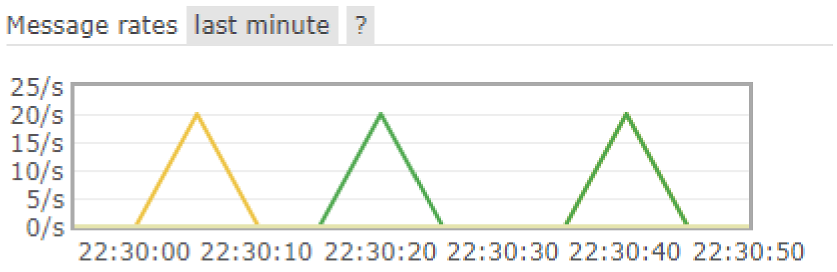


```
cleo@vm-xenial0:~$ cat output
5724-H72 (C) Copyright IBM Corp. 1994, 2018.
Starting MQSC for queue manager PERF0.
AMQ9202E: Remote host not available, retry later.

0 command responses received.

Command line: ./cph -nt 200 -ms 204800 -vo 3 -rl 0 -id 1 -tc Responder -ss 0 -iq REQUEST -oq REPLY -db 1 -dx 10 -jp 1420 -jc SYSTEM.DEF.SVRCON
N -jb PERF0 -jt mqc -jh localhost -wi 10 -wt 30 -to -1 -tx true -pp true -jl
Shared library libmqic_r.so loaded ok
[Responder0] START
Created Error message to pass to runtime_error()Call to MQCONNX failed [Completion code: 2; Reason code: 2538][Responder0] Caught exception: C
all to MQCONNX failed [Completion code: 2; Reason code: 2538]
[Responder0] STOP
Starting test with 1 requesters
Command line: ./cph -nt 1 -ms 2048 -rl 90 -id 1 -tc Requester -ss 10 -iq REQUEST -oq REPLY -db 1 -dx 10 -jp 1420 -jc SYSTEM.DEF.SVRCONN -jb PE
RF0 -jt mqc -jh localhost -wi 10 -to 30 -tx true -pp true -jl
controlThread START
Shared library libmqic_r.so loaded ok
[Requester0] START
[Requester0] Connecting to QM: PERF0 (host: localhost; port: 1420; channel: SYSTEM.DEF.SVRCONN)
Created Error message to pass to runtime_error()Call to MQCONNX failed [Completion code: 2; Reason code: 2538][Requester0] Caught exception: C
all to MQCONNX failed [Completion code: 2; Reason code: 2538]
[Requester0] STOP
[ControlThread] Caught exception: Responder0: State ERROR set.
totalIterations=0,totalSeconds=1557395563.92,avgRate=0.00
id=1,rate=0.00,threads=0
id=1,rate=0.00,threads=0
[ControlThread] Caught exception: Requester0: State ERROR set.
totalIterations=0,totalSeconds=1557395593.84,avgRate=0.00
controlThread STOP
Starting test with 2 requesters
Command line: ./cph -nt 2 -ms 2048 -rl 90 -id 1 -tc Requester -ss 10 -iq REQUEST -oq REPLY -db 1 -dx 10 -jp 1420 -jc SYSTEM.DEF.SVRCONN -jb PE
RF0 -jt mqc -jh localhost -wi 10 -to 30 -tx true -pp true -jl
controlThread START
Shared library libmqic_r.so loaded ok
[Requester0] START
[Requester0] Connecting to QM: PERF0 (host: localhost; port: 1420; channel: SYSTEM.DEF.SVRCONN)
Created Error message to pass to runtime_error()Call to MQCONNX failed [Completion code: 2; Reason code: 2538][Requester0] Caught exception: C
all to MQCONNX failed [Completion code: 2; Reason code: 2538]
```

Test the performance of MQ in docker container

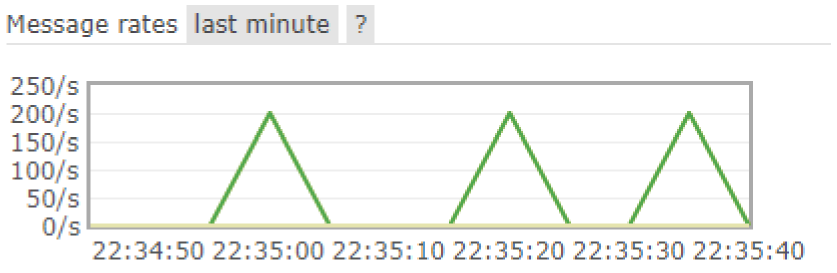
- 100 messages





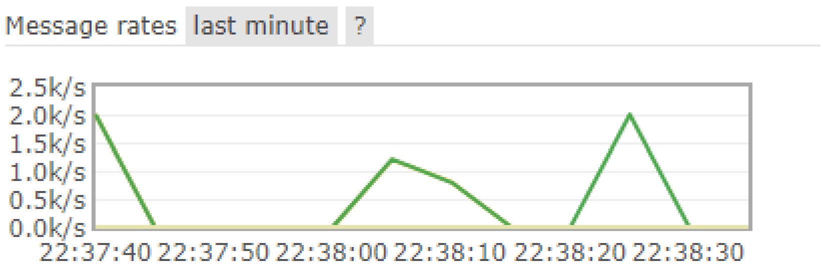
max: 20/s

- 1000 messages



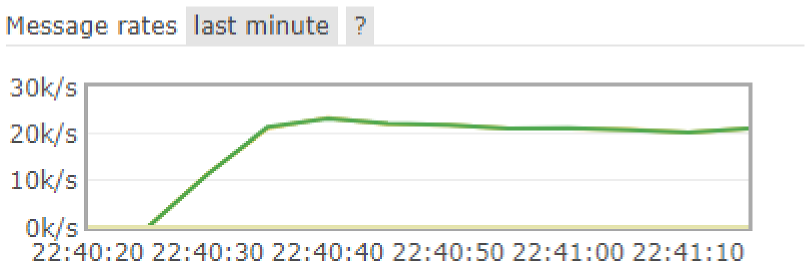
max: 200/s

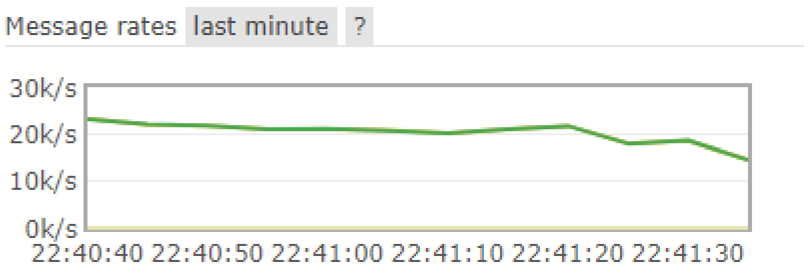
- 10000 messages

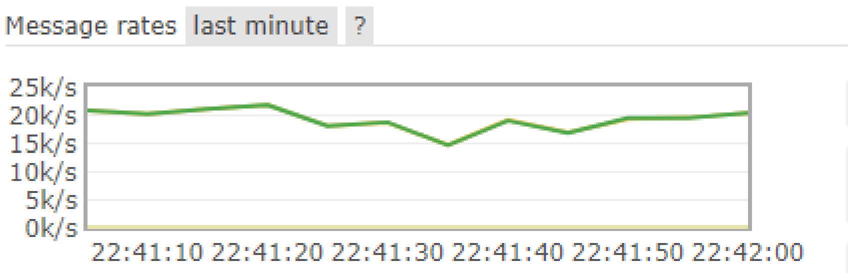


max: 2.0k/s

- infinitas messages







max: 21700/s