2025-07-OTUS-RABBITMQ

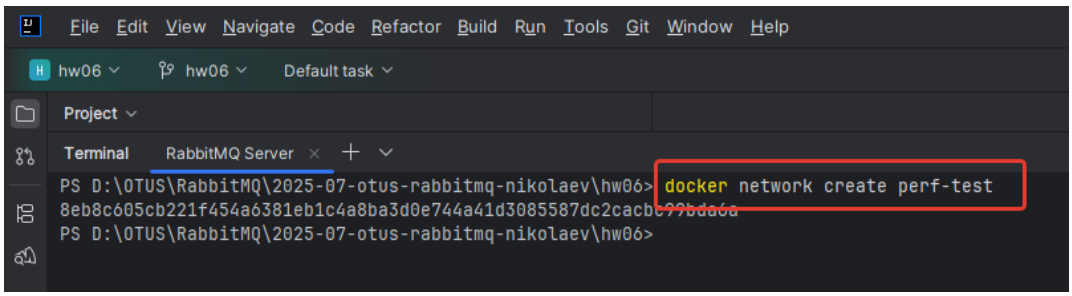
Домашнее задание №6

Александр Николаев

1. Тестирование классических очередей.

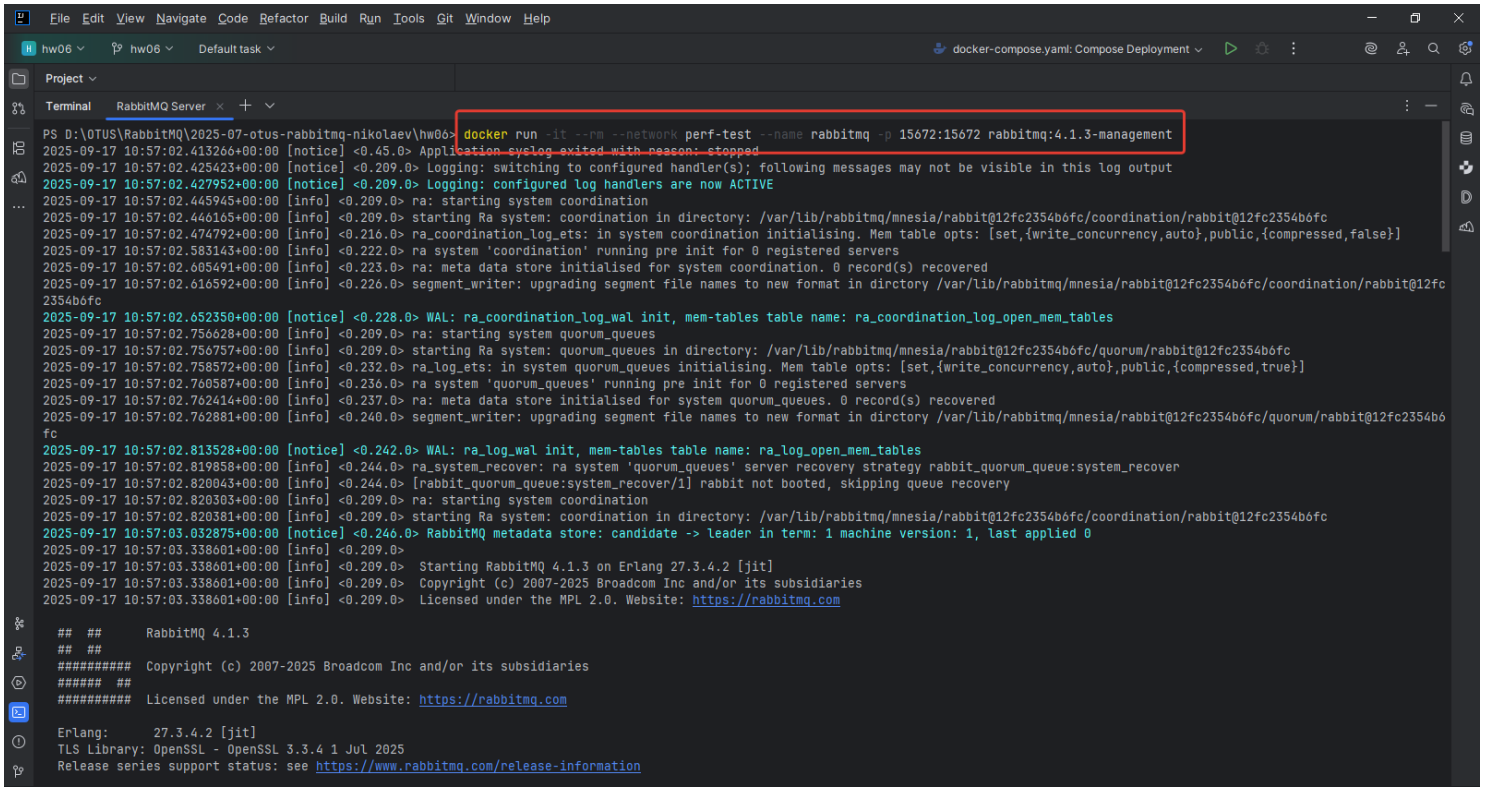
1.1. Создаем отдельную сеть в Docker для связи RabbitMQ сервера и stream-perf-test.

```
docker network create stream-perf-test
```



1.2. Создаем и запускаем докер контейнер с сервером RabbitMQ в созданной на предыдущем шаге сети.

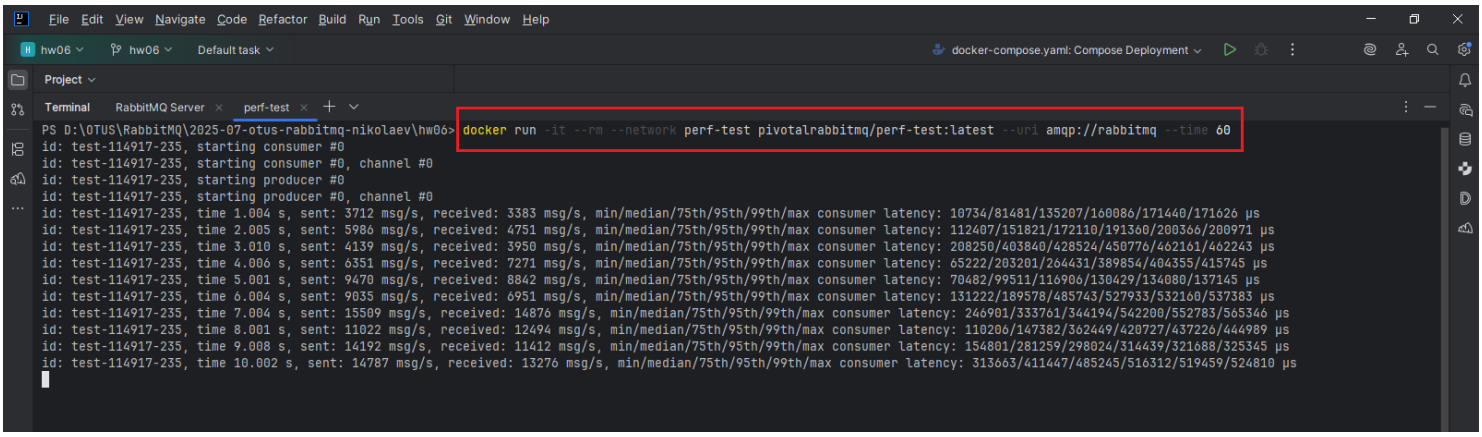
```
docker run -it --rm --network perf-test --name rabbitmq rabbitmq:4.1.3-management
```



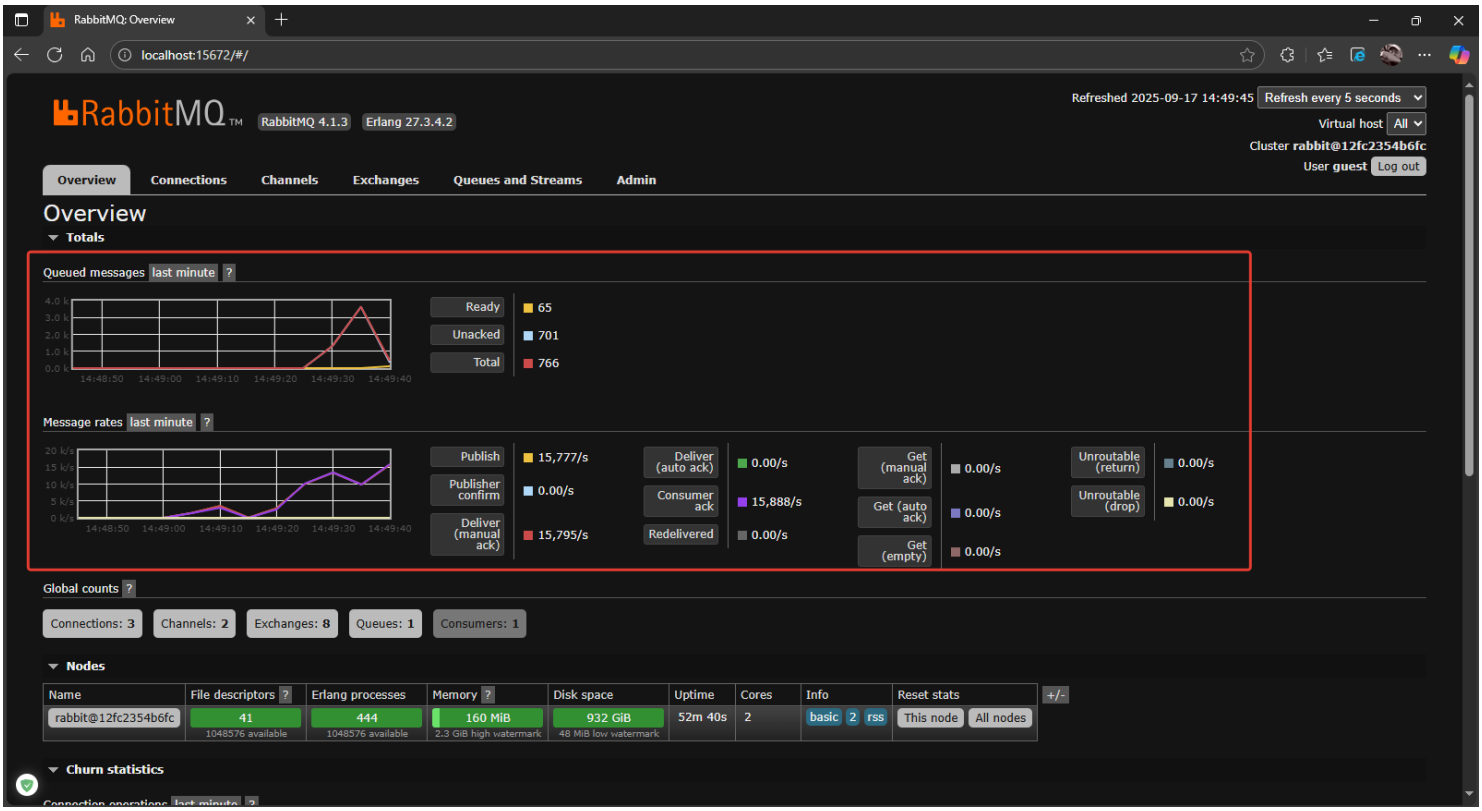
1.3. Создаем и запускаем докер контейнер с утилитой perf-test.jar для тестирования классических очередей (60 сек).

```
docker run -it --rm --network perf-test pivotalrabbitmq/perf-test:latest --uri amqp://rabbitmq --time 60
```

Тестирование проводим с 1 продюсером и 1 консьюмером. Размер сообщения по умолчанию – 12 байт.



1.4. Проверяем в management-ui, что нагрузка на сервер пошла.



1.5. Результат выполнения теста.

Зафиксируем результат – классические очереди, 1 продюсер, 1 консьюмер, размер сообщения – 12 байт.

Sending rate avg: 12136 msg/s, Receiving rate avg: 12126 msg/s

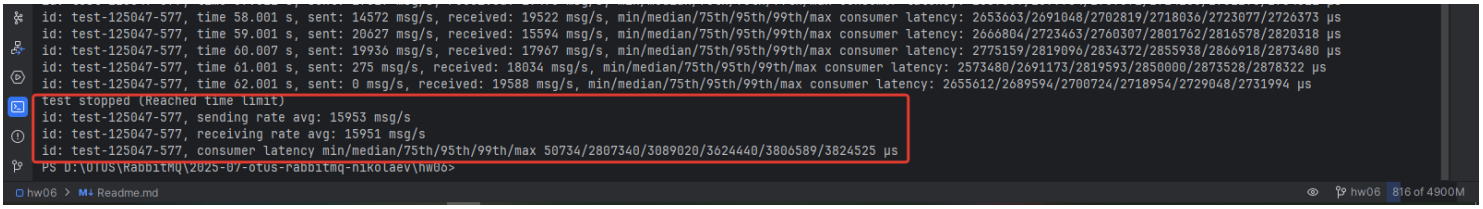
1.6. Произведем такое же тестирование, но введем для консьюмера prefetch (QoS) 500 сообщений.

```
docker run -it --rm --network perf-test pivotalrabbitmq/perf-test:latest --uri amqp://rabbitmq -q 500 --time 60
```

1.7. Результат выполнения теста после 60 секунд.

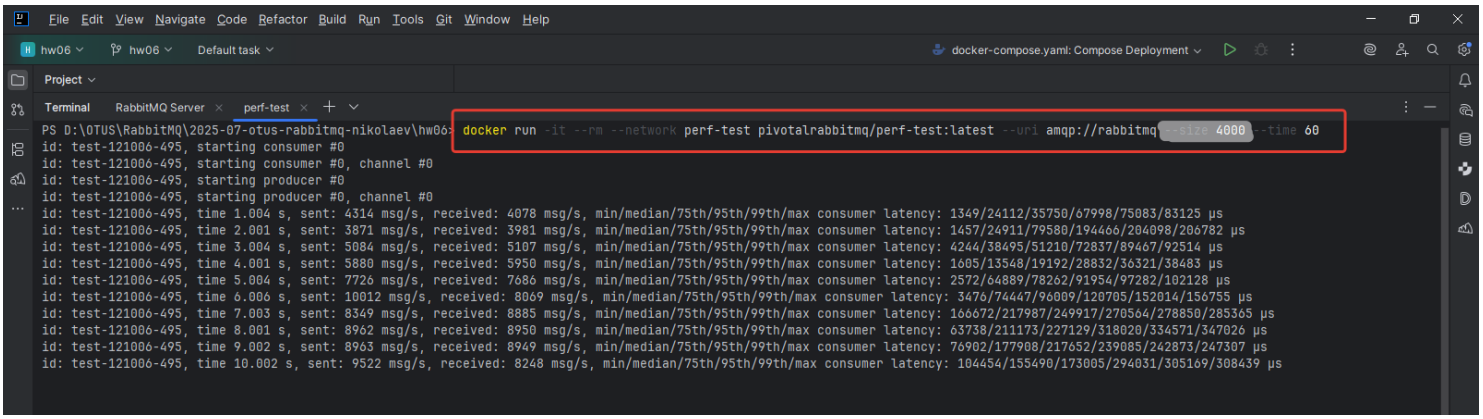
Зафиксируем результат – классические очереди, 1 продюсер, 1 консьюмер, размер сообщения – 12 байт, prefetch - 500

Sending rate avg: 15953 msg/s, Receiving rate avg: 15951 msg/s



1.8. Произведем тестирование без prefetch, но увеличим размер сообщения до 4 кб.

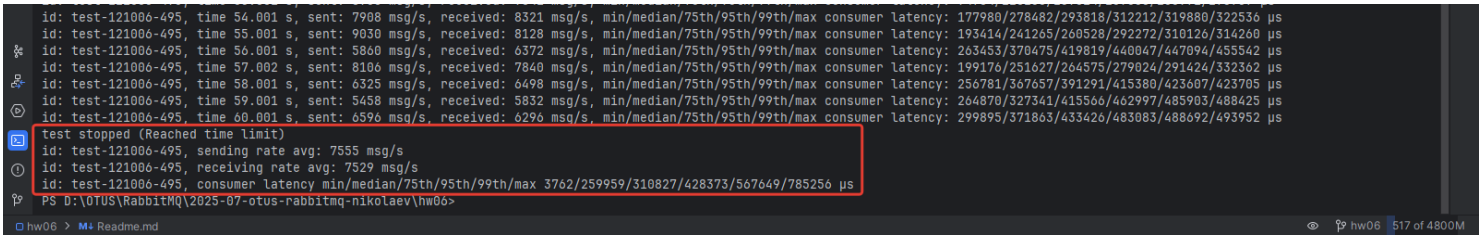
```
docker run -it --rm --network perf-test pivotalrabbitmq/perf-test:latest --uri amqp://rabbitmq
--size 4000 --time 60
```



1.9. Результат выполнения теста.

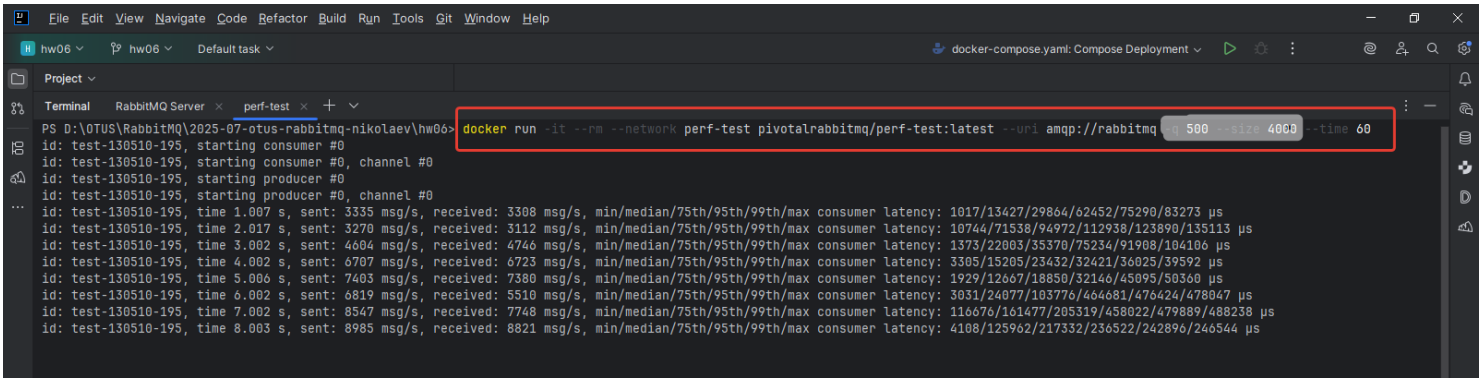
Зафиксируем результат – классические очереди, 1 продюсер, 1 консьюмер, размер сообщения – 4000 байт.

Sending rate avg: 7555 msg/s, Receiving rate avg: 7529 msg/s



1.10. Произведем тестирование, увеличим размер сообщения до 4 кб и добавим prefetch для консьюмера 500.

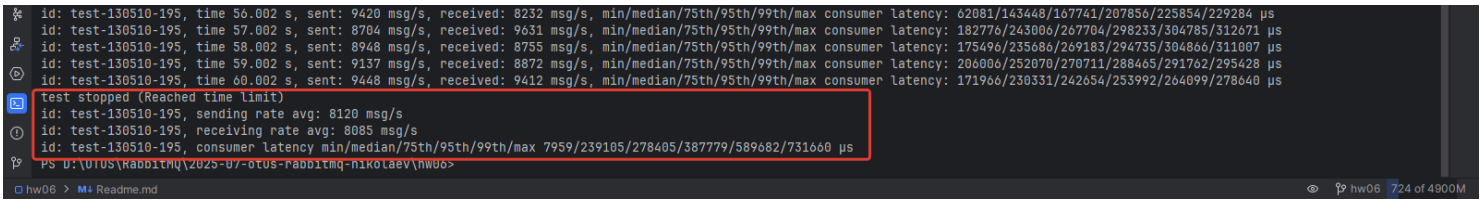
```
docker run -it --rm --network perf-test pivotalrabbitmq/perf-test:latest --uri amqp://rabbitmq -q 500
--size 4000 --time 60
```



1.11. Результат выполнения теста.

Зафиксируем результат – классические очереди, 1 продюсер, 1 консьюмер, размер сообщения – 4000 байт, prefetch - 500.

Sending rate avg: 8120 msg/s, Receiving rate avg: 8085 msg/s

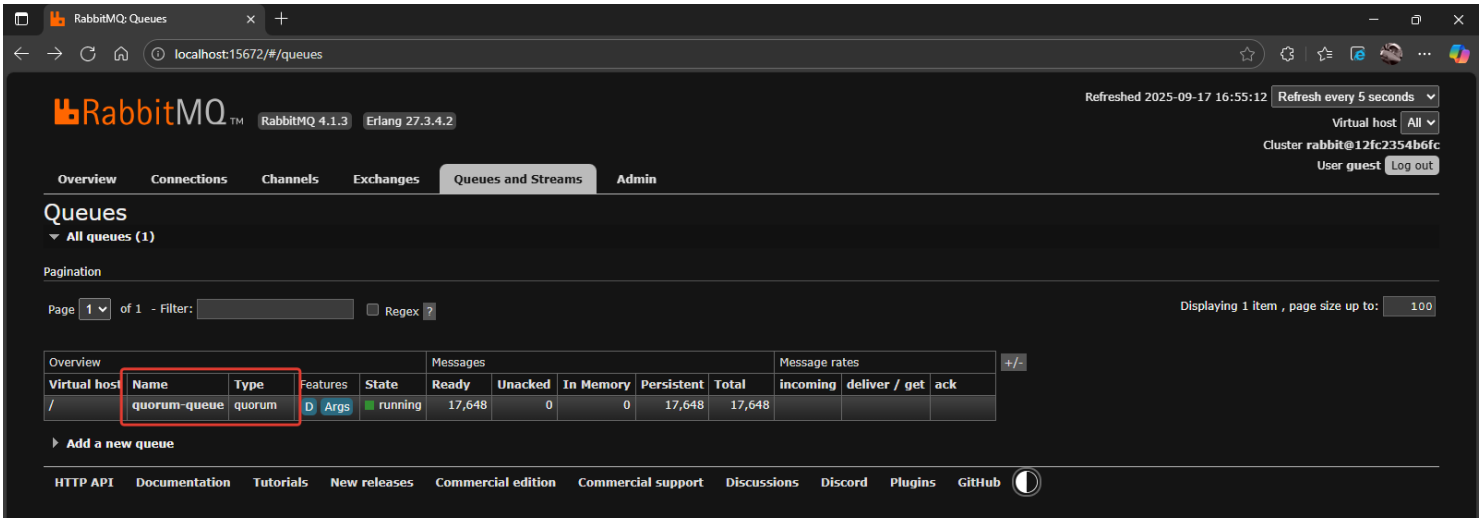


1.12. Общий результат тестирования классических очередей

Номер, п/п	Параметры теста, classic queue	Sending rate avg., msg/s	Receiving rate avg., msg/s
1	msg size – 12 bytes, no prefetch	12136	12126
2	msg size – 12 bytes, prefetch - 500	15953	15951
3	msg size – 4000 bytes, no prefetch	7555	7529
4	msg size – 4000 bytes, prefetch - 500	8120	8085

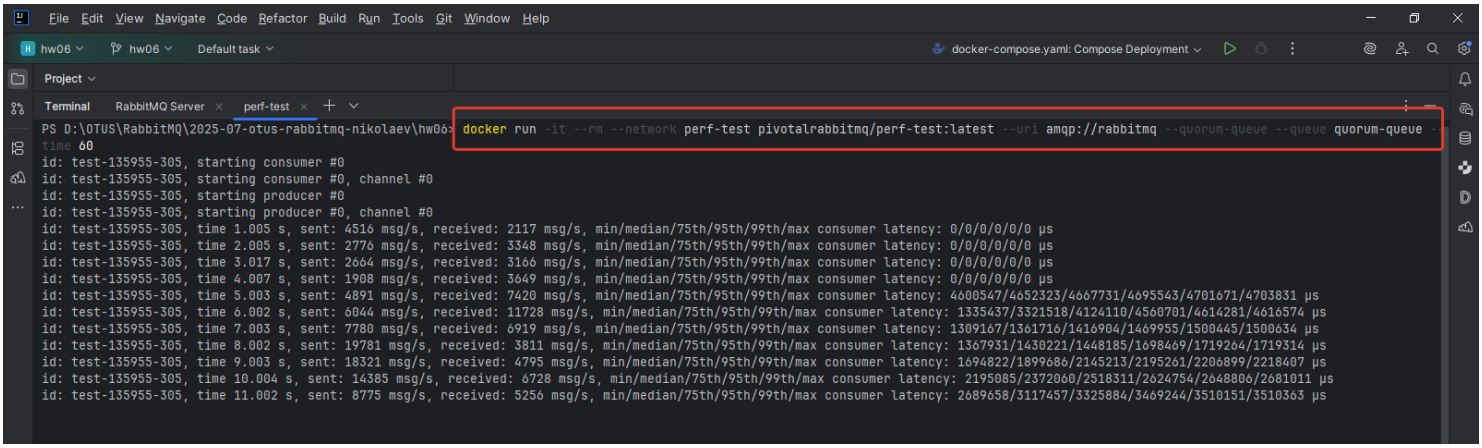
2. Тестирование кворумных очередей.

2.1. Для тестирования кворумных очередей создадим такую очередь.



2.2. Произведем тестирование кворумной очереди. Параметры теста дефолтные – 1 продюсер, 1 консьюмер, размер сообщения – 12 байт, время тестирования – 60 секунд.

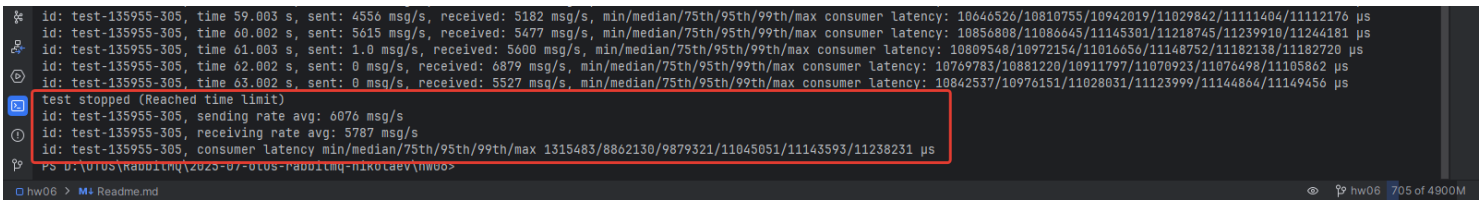
```
docker run -it --rm --network perf-test pivotalrabbitmq/perf-test:latest --uri amqp://rabbitmq
--quorum-queue --queue quorum-queue --time 60
```



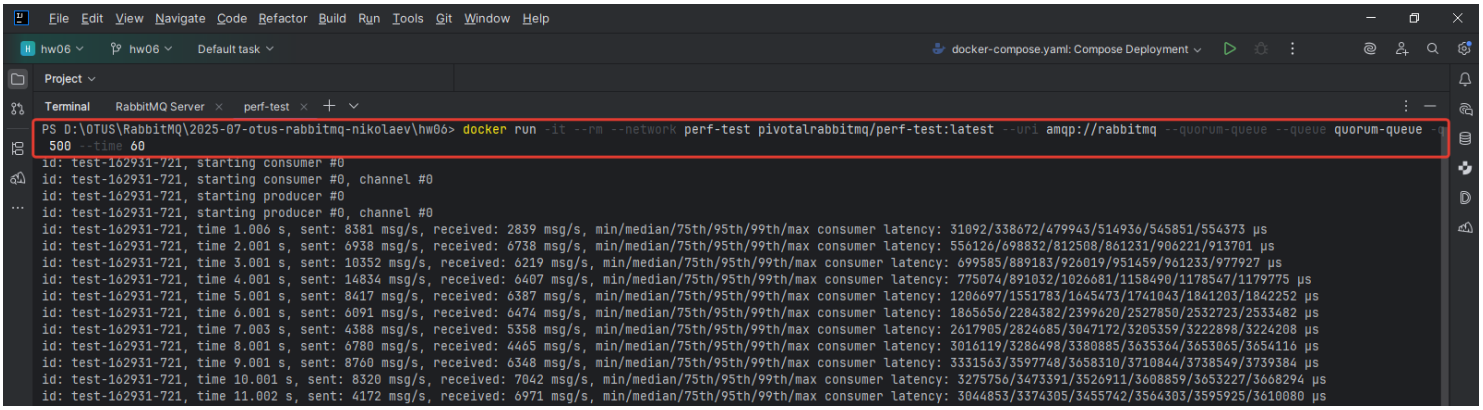
2.3. Результат выполнения теста.

Зафиксируем результат – кворумные очереди, 1 продюсер, 1 консьюмер, размер сообщения – 12 байт.

Sending rate avg: 8120 msg/s, Receiving rate avg: 8085 msg/s



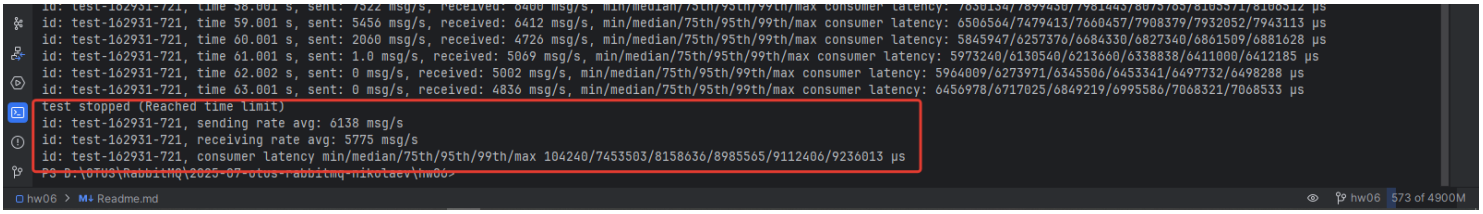
2.4. Произведем такое же тестирование, кворумные очереди, дефолтные значения, но введем для консьюмера prefetch (QoS) 500 сообщений.



2.5. Результат выполнения теста.

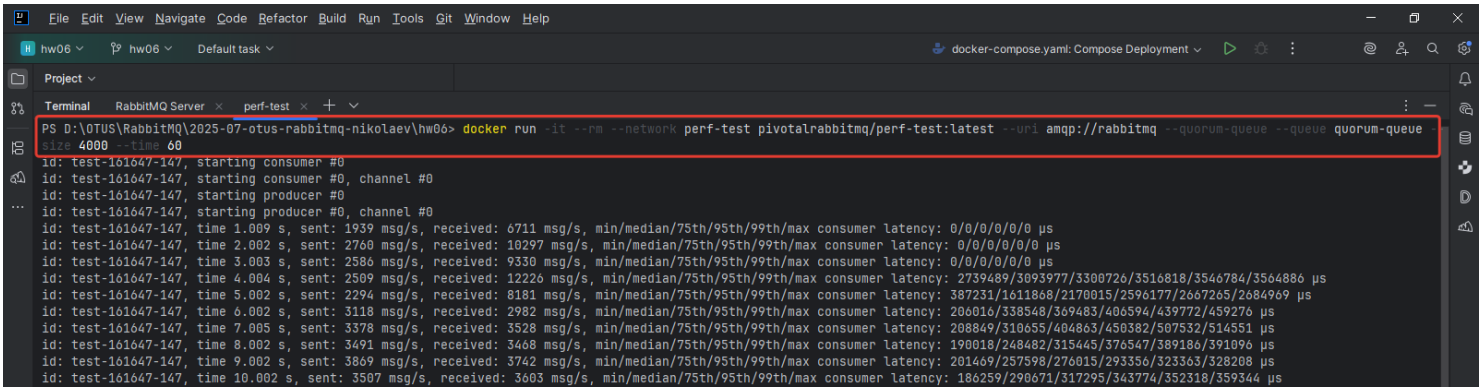
Зафиксируем результат – кворумные очереди, 1 продюсер, 1 консьюмер, размер сообщения – 12 байт, prefetch - 500.

Sending rate avg: 6138 msg/s, Receiving rate avg: 5775 msg/s



2.6. Произведем тестирование кворумной очереди изменив параметры теста. 1 продюсер, 1 консьюмер, размер сообщения – 4000 байт, prefetch – отсутствует, время тестирования – 60 секунд.

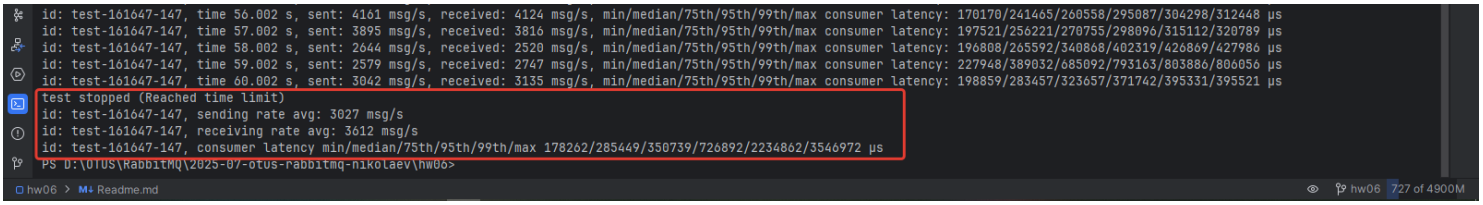
```
docker run -it --rm --network perf-test pivotalrabbitmq/perf-test:latest --uri amqp://rabbitmq
--quorum-queue --queue quorum-queue –size 4000 --time 60
```



2.7. Результат выполнения теста.

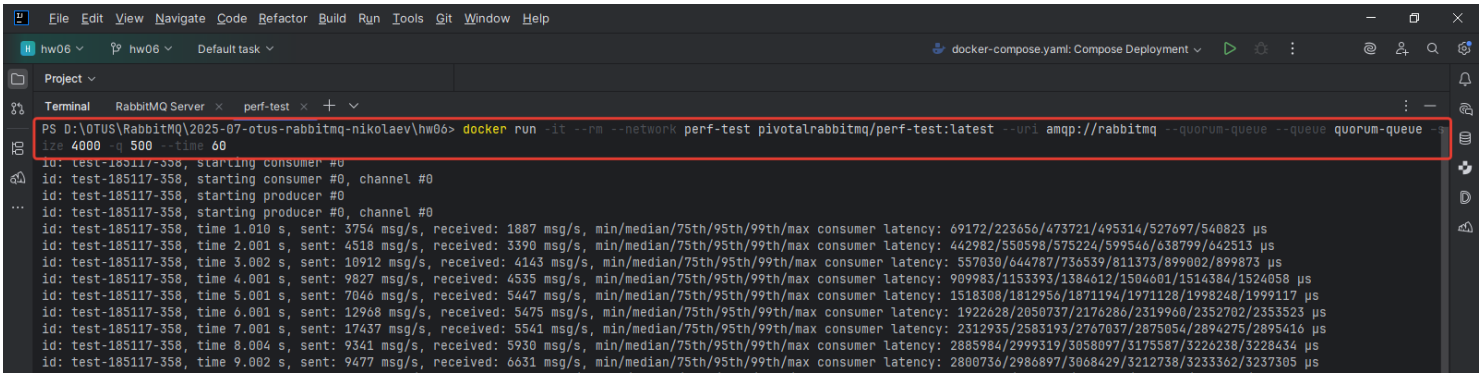
Зафиксируем результат – кворумные очереди, 1 продюсер, 1 консьюмер, размер сообщения – 4000 байт.

Sending rate avg: 3027 msg/s, Receiving rate avg: 3612 msg/s



2.8. Произведем тестирование кворумной очереди изменив параметры теста. 1 продюсер, 1 консьюмер, размер сообщения – 4000 байт, prefetch – 500, время тестирования – 60 секунд.

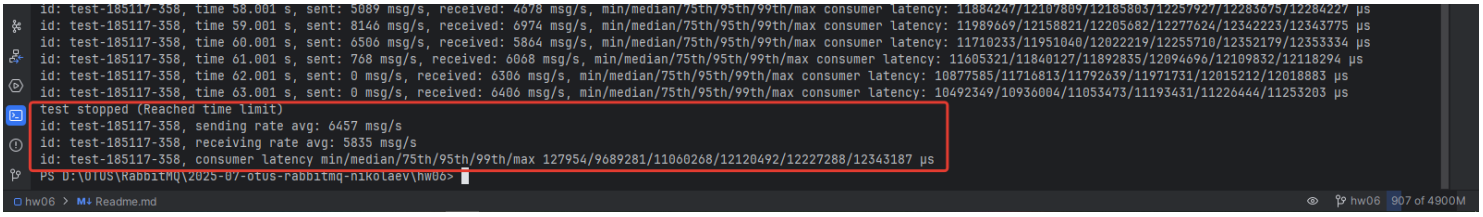
```
docker run -it --rm --network perf-test pivotalrabbitmq/perf-test:latest --uri amqp://rabbitmq
--quorum-queue --queue quorum-queue –size 4000 -q 500 --time 60
```



2.9. Результат выполнения теста.

Зафиксируем результат – кворумные очереди, 1 продюсер, 1 консьюмер, размер сообщения – 4000 байт, prefetch - 500.

Sending rate avg: 6457 msg/s, Receiving rate avg: 5835 msg/s



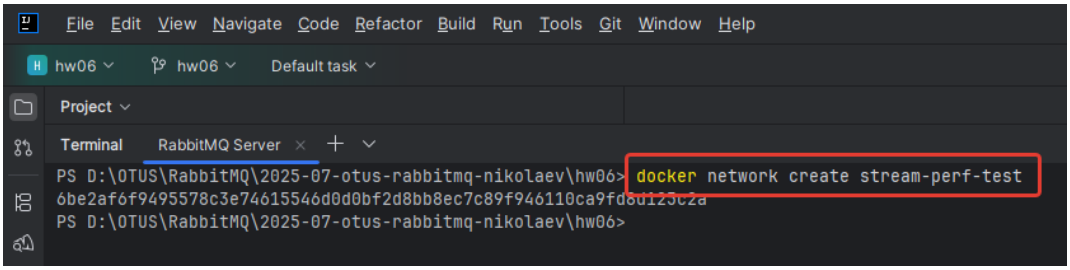
2.10. Общий результат тестирования кворумных очередей

Номер, п/п	Параметры теста, quorum queue	Sending rate avg., msg/s	Receiving rate avg., msg/s
1	msg size – 12 bytes, no prefetch	6076	5787
2	msg size – 12 bytes, prefetch - 500	6138	5775
3	msg size – 4000 bytes, no prefetch	3027	3612
4	msg size – 4000 bytes, prefetch - 500	6457	5835

3. Тестирование стримов.

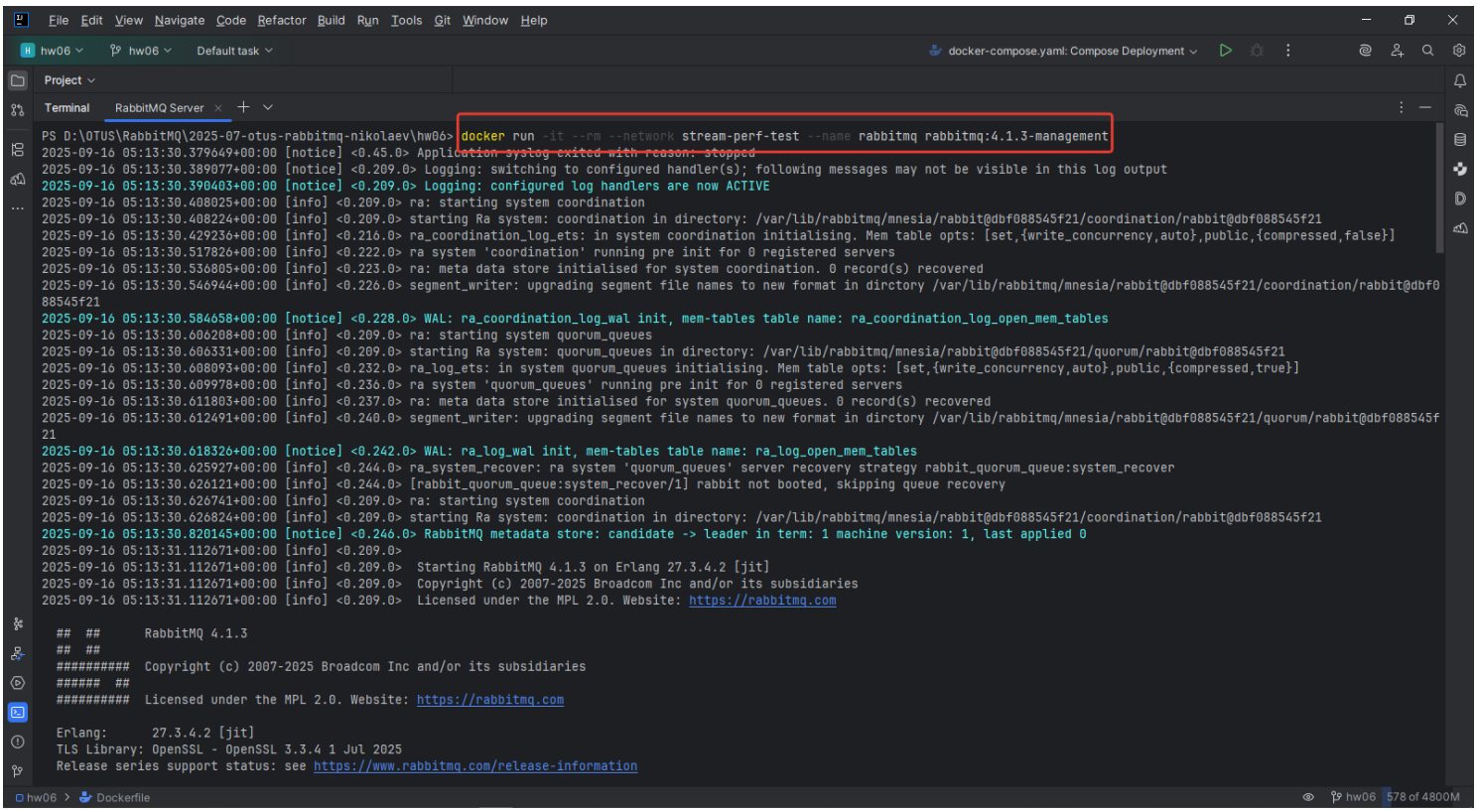
3.1. Создаем отдельную сеть в Docker для связи RabbitMQ сервера и stream-perf-test.

```
docker network create stream-perf-test
```



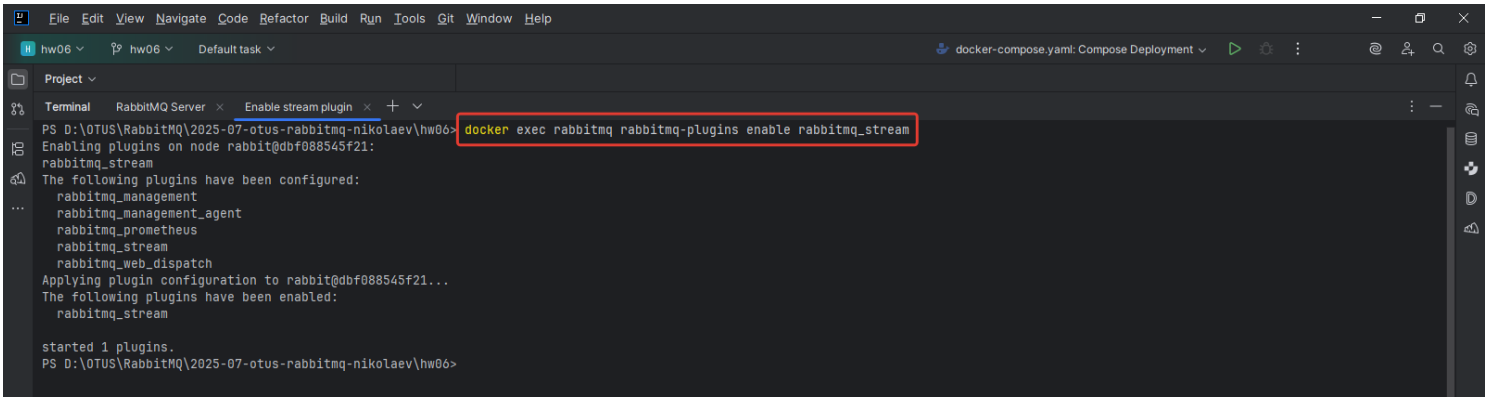
3.2. Создаем и запускаем новый докер контейнер с сервером RabbitMQ в созданной на предыдущем шаге сети.

```
docker run -it --rm --network stream-perf-test --name rabbitmq -p 15672:15672 rabbitmq:4.1.3-management
```



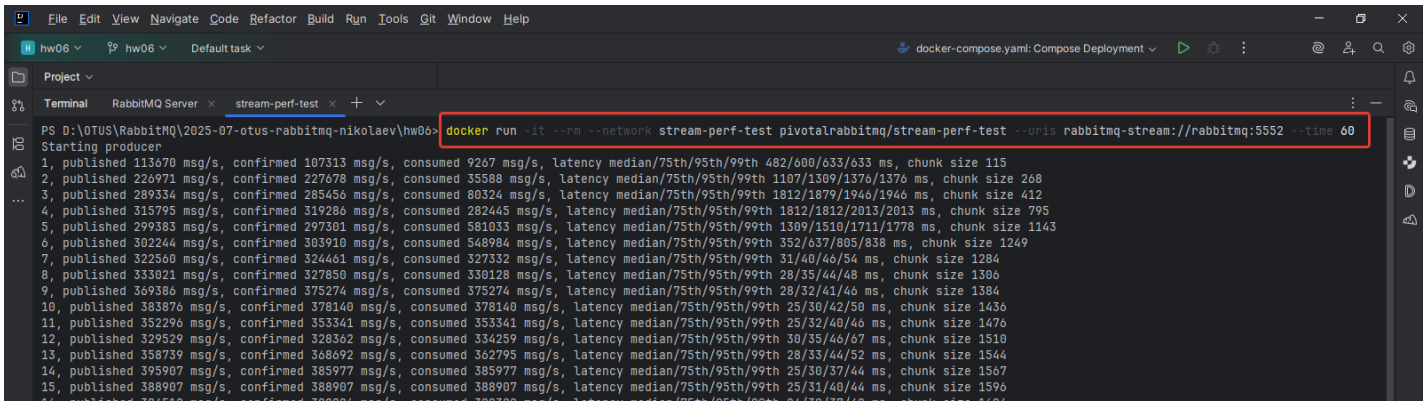
3.3. Включаем плагин rabbitmq_stream.

```
docker exec rabbitmq rabbitmq-plugins enable rabbitmq_stream
```

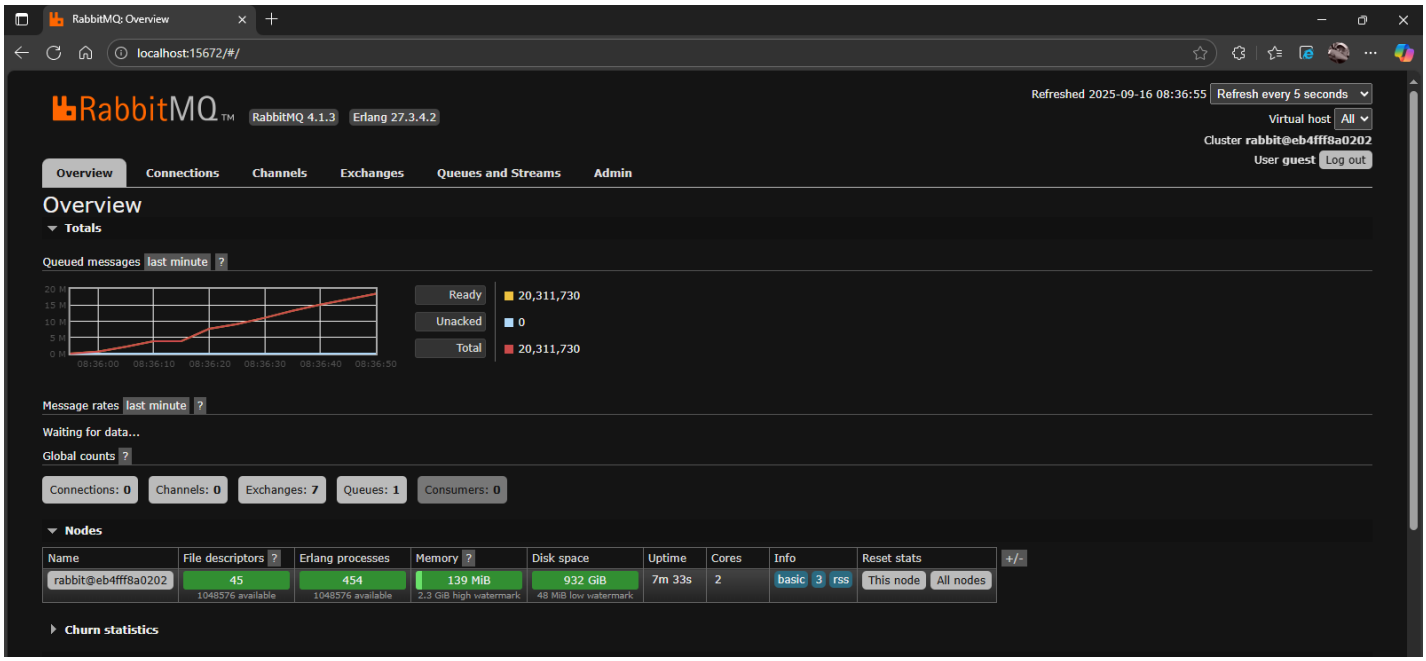


3.4. Произведем тестирование стрим очереди. Параметры теста дефолтные – 1 продюсер, размер сообщения – 12 байт, время тестирования – 60 секунд.

```
docker run -it --rm --network stream-perf-test pivotalrabbitmq/stream-perf-test:latest --uris rabbitmq-stream://rabbitmq:5552 --time 60
```



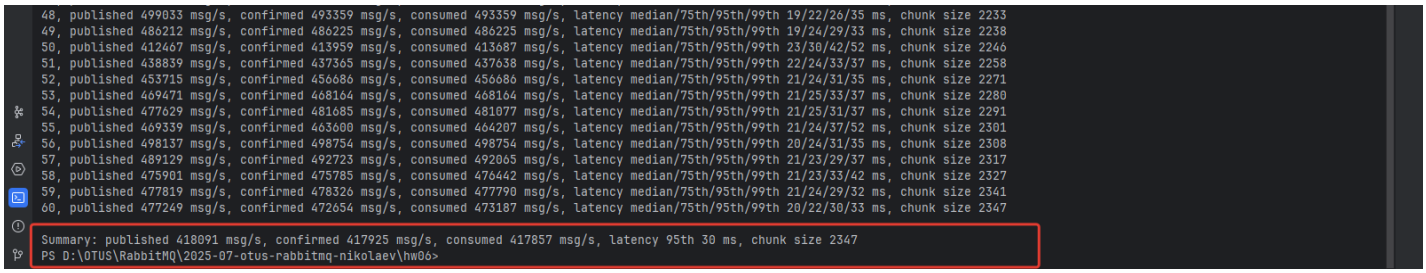
3.5. Проверяем в management-ui, что нагрузка на сервер пошла.



3.6. Результат выполнения теста.

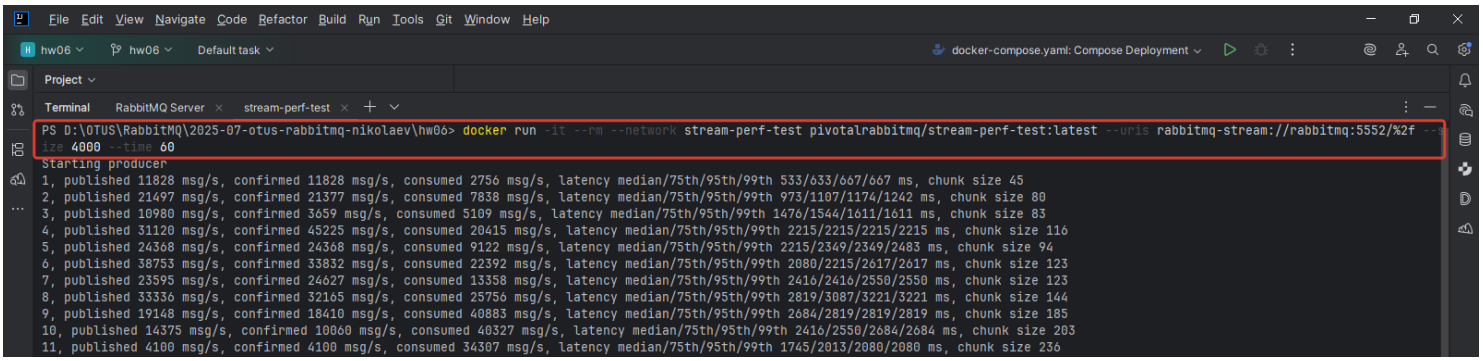
Зафиксируем результат – стрим очереди, 1 продюсер, размер сообщения – 12 байт.

Published 418091 msg/s, confirmed 417925 msg/s, consumed 417857 msg/s



3.7. Произведем тестирование стрим очереди изменив параметры теста. Параметры теста дефолтные – 1 продюсер, размер сообщения – 4000 байт, время тестирования – 60 секунд.

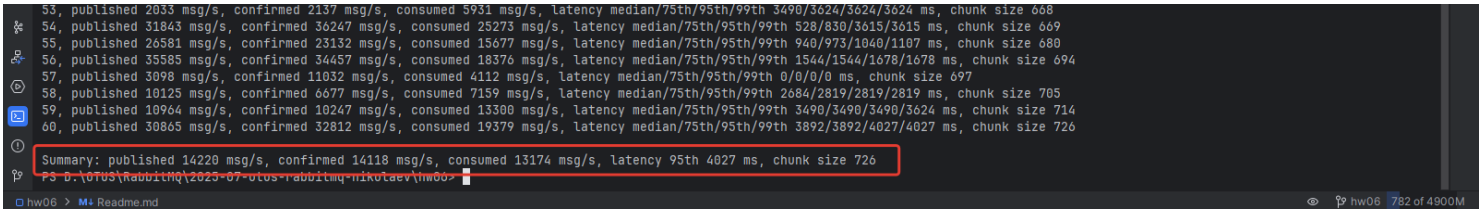
```
docker run -it --rm --network stream-perf-test pivotalrabbitmq/stream-perf-test:latest --uris rabbitmq-stream://rabbitmq:5552 -size 4000 --time 60
```



3.8. Результат выполнения теста.

Зафиксируем результат – стрим очереди, 1 продюсер, размер сообщения – 4000 байт.

Published 14220 msg/s, confirmed 14118 msg/s, consumed 13174 msg/s



3.9. Общий результат тестирования кворумных очередей

Номер, п/п	Параметры теста, stream queue	Published, msg/s	Confirmed, msg/s	Consumed, msg/s
1	msg size – 12 bytes	418091	417925	417857
2	msg size – 4000 bytes	14220	14118	13174

Выводы:

1. Размер сообщения сильно влияет на скорость передачи сообщения от продюсера к консьюмеру.
2. Кворумные очереди медленнее, чем классические, и тем более, чем стримы.
3. Стрим очереди самые быстрые, но надо заботится о количестве unconfirmed сообщений.
4. В общем случае предел предварительной выборки prefetch увеличивает скорость передачи, но этот параметр надо подбирать отдельно.