# **11.4 Очереди RabbitMQ - Воронин Сергей**

### **Задание 1. Установка RabbitMQ**

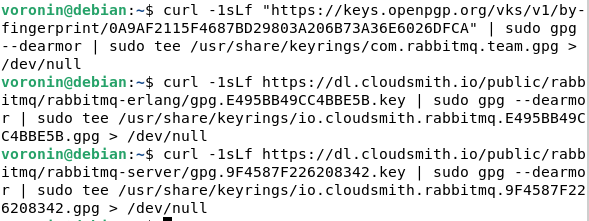
Используя Vagrant или VirtualBox, создайте виртуальную машину и установите RabbitMQ. Добавьте management plug-in и зайдите в веб интерфейс.

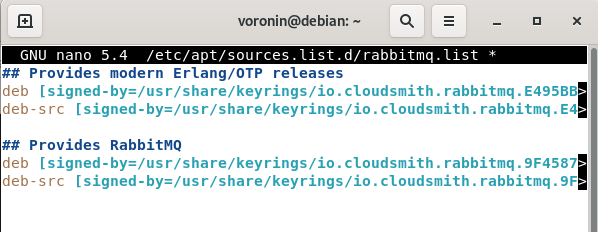
*Итогом выполнения домашнего задания будет приложенный скриншот веб интерфейса RabbitMQ.*

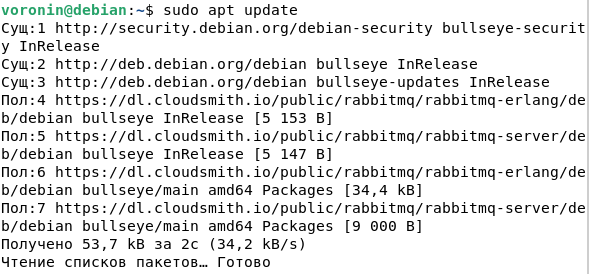
Установка

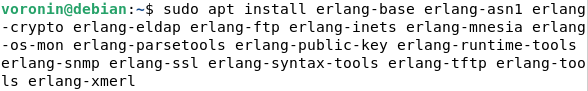




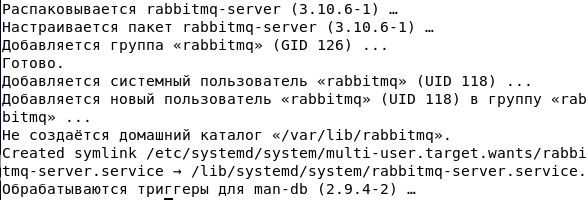




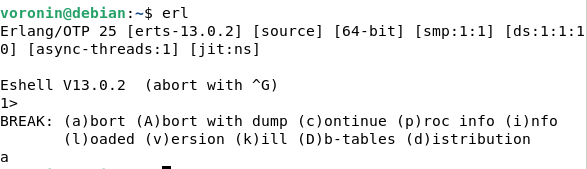




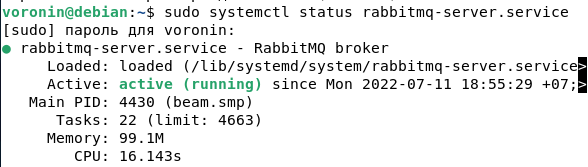




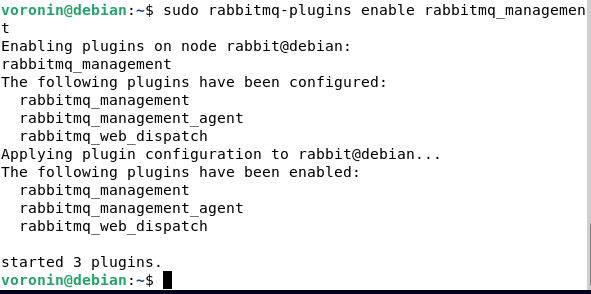
**Erlang есть**



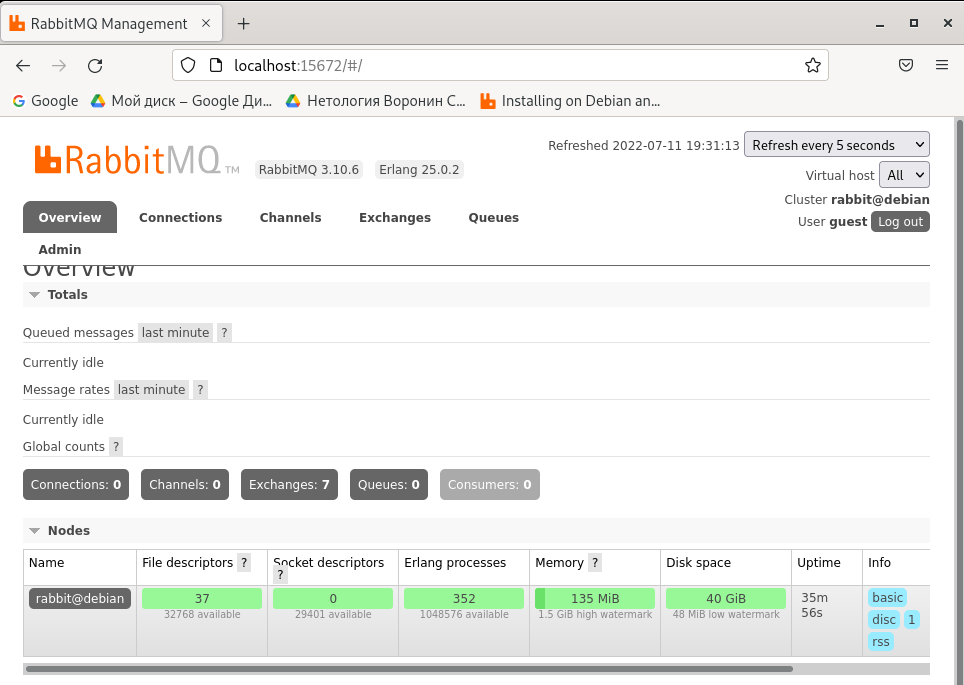
**Rabbitmq запущен**



**Запуск плагина rabbitmq\_management для веб-интерфейса**



**Заходим в веб-интерфейс localhost:15672, guest:guest**

****

### **Задание 2. Отправка и получение сообщений.**

Используя приложенные скрипты, проведите тестовую отправку и получение сообщения. Для отправки сообщений необходимо запустить скрипт producer.py

Для работы скриптов вам необходимо установить Python версии 3 и библиотеку pika. Также в скриптах нужно указать ip адрес машины, на которой запущен RabbitMQ, заменив localhost на нужный ip.

$ pip install pika

Зайдите в веб интерфейс, найдите очередь под названием hello и сделайте скриншот. После чего запустите второй скрипт consumer.py и сделайте скриншот результата выполнения скрипта

*В качестве решения домашнего задания приложите оба скриншота, сделанных на этапе выполнения.*

Для закрепления материала можете попробовать модифицировать скрипты, чтобы поменять название очереди и отправляемое сообщение.

**IP**

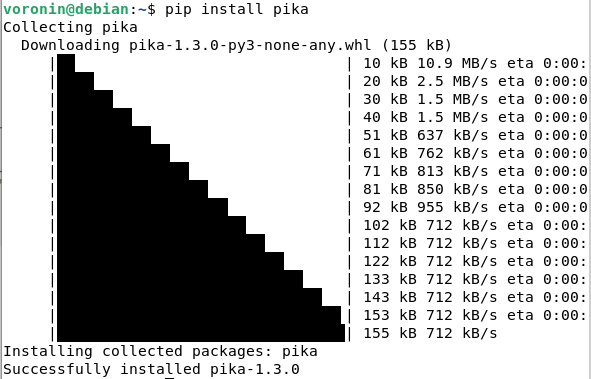


**Устанавливаем pip**





**Устанавливаем pika**

****

**Создаем producer.py**

#!/usr/bin/env python3

# coding=utf-8

import pika

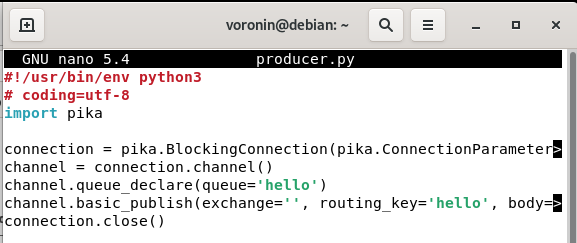
connection = pika.BlockingConnection(pika.ConnectionParameters('127.0.0.1'))

channel = connection.channel()

channel.queue\_declare(queue='hello')

channel.basic\_publish(exchange='', routing\_key='hello', body='Hello Netology from Voronin')

connection.close()

****

**Создаем consumer.py**

#!/usr/bin/env python3

# coding=utf-8

import pika

connection = pika.BlockingConnection(pika.ConnectionParameters(‘127.0.0.1’))

channel = connection.channel()

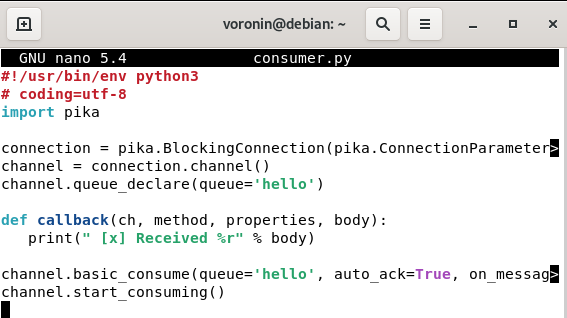
channel.queue\_declare(queue='hello')

def callback(ch, method, properties, body):

print(" [x] Received %r" % body)

channel.basic\_consume(queue='hello', auto\_ack=True, on\_message\_callback=callback)

channel.start\_consuming()

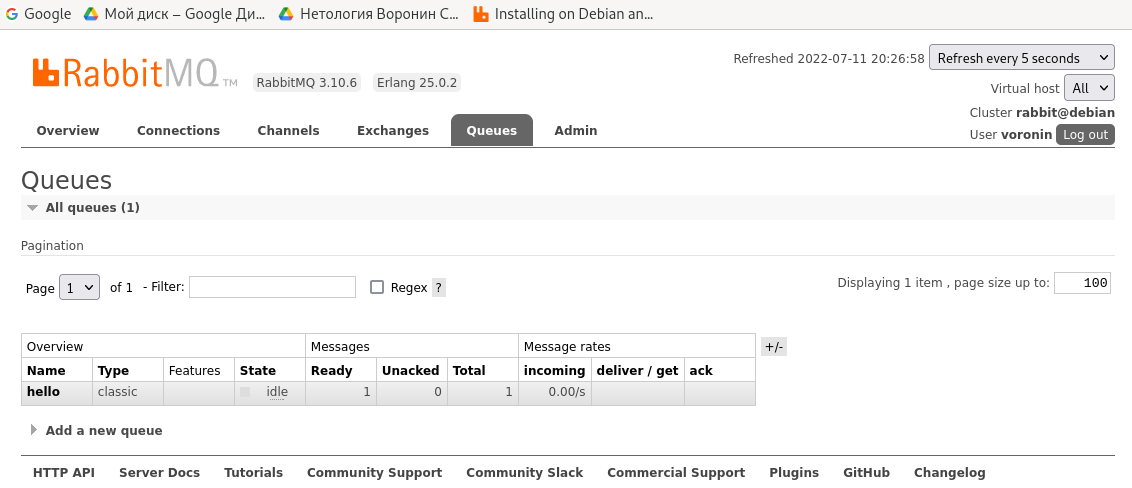


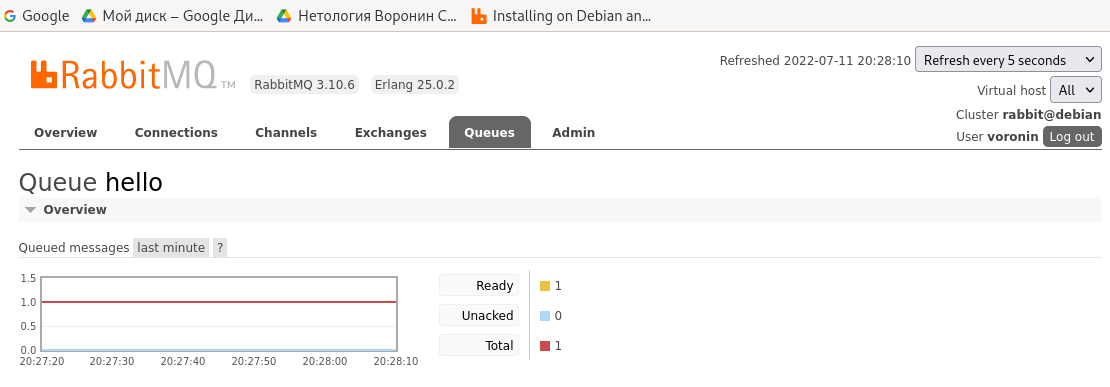
**Делаем файл скрипта исполняемым:**



**Запускаем producer.py**

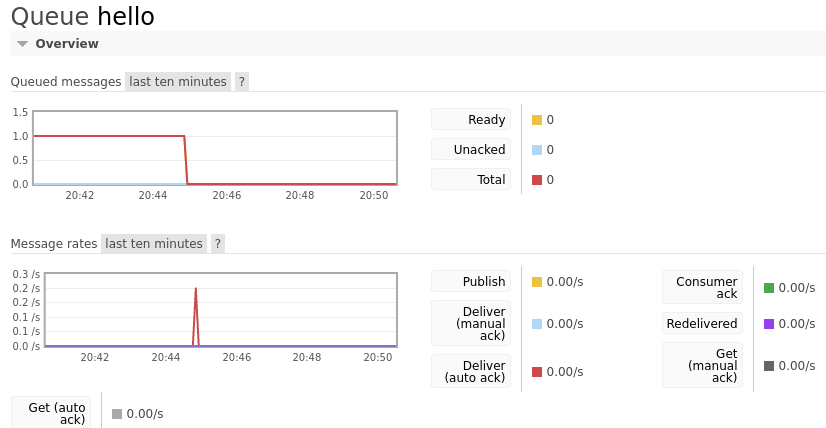






**Запускаем consumer.py, получаем:**





**Кстати, если запускать скрипт из методички, то будут сотни ошибок, начиная с env python (у меня заработало с env python3), заканчивая channel.basic\_consume(callback, queue='hello', no\_ack=True) (заработало с channel.basic\_consume(queue='hello',auto\_ack=True,on\_message\_callback=callback)). Еще повезло, что это я такой офигенно умный, был бы тупой - было бы грустно…**

### **Задание 3. Подготовка HA кластера**

Используя Vagrant или VirtualBox, создайте вторую виртуальную машину и установите RabbitMQ. Добавьте в файл hosts название и ip адрес каждой машины, чтобы машины могли видеть друг друга по имени.

Пример содержимого hosts файла:

$ cat /etc/hosts

192.168.0.10 rmq01

192.168.0.11 rmq02

После этого ваши машины могут пинговаться по имени.

Затем объедините две машины в кластер и создайте политику ha-all на все очереди.

*В качестве решения домашнего задания приложите скриншоты из веб интерфейса с информацией о доступных нодах в кластере и включенной политикой.*

Также приложите вывод команды с двух нод:

$ rabbitmqctl cluster\_status

Для закрепления материала снова запустите скрипт producer.py и приложите скриншот выполнения команды на каждой из нод:

$ rabbitmqadmin get queue='hello'

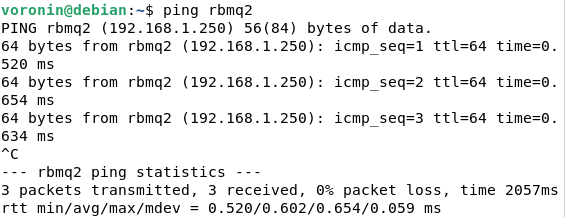
После чего попробуйте отключить одну из нод, желательно, ту к которой подключались из скрипта, затем поправьте параметры подключения в скрипте consumer.py на вторую ноду и запустите его.

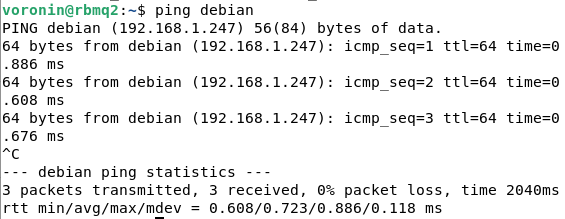
*Приложите скриншот результата работы второго скрипта.*

**Первая машина 192.168.1.247 debian**

**Вторая машина 192.168.1.250 rbmq2**







**Содержимое файла /var/lib/rabbitmq/.erlang.cookie на debian должно совпадать с содержимым файла на rbmq2.**

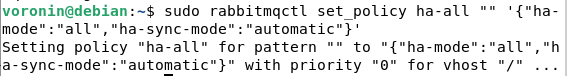






**Создание политики репликации**

rabbitmqctl set\_policy ha-all "" '{"ha-mode":"all","ha-sync-mode":"automatic"}'

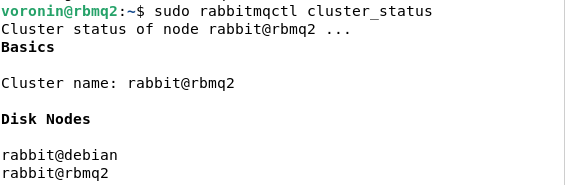


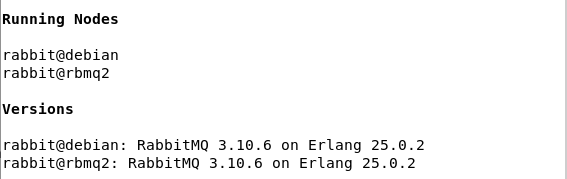
**Вторая нода (rbmq2)**

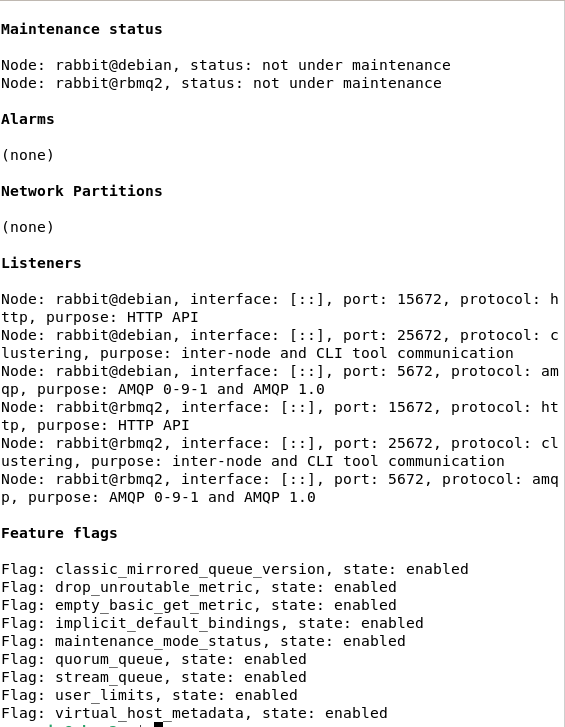


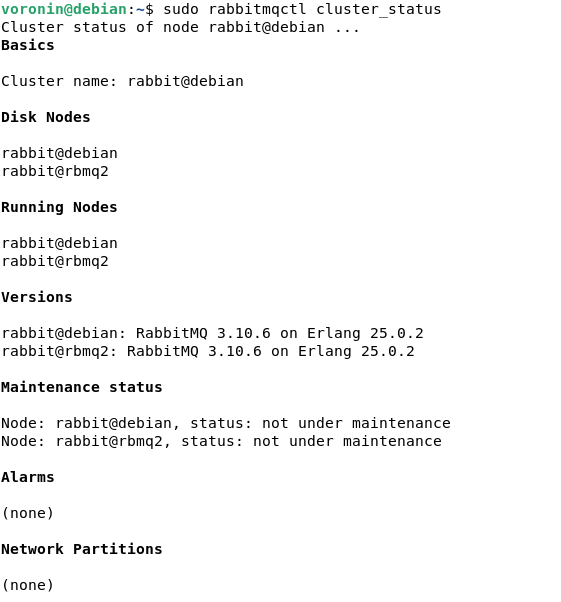


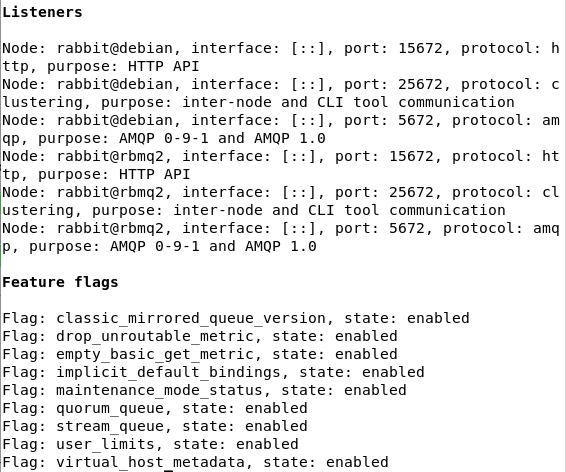




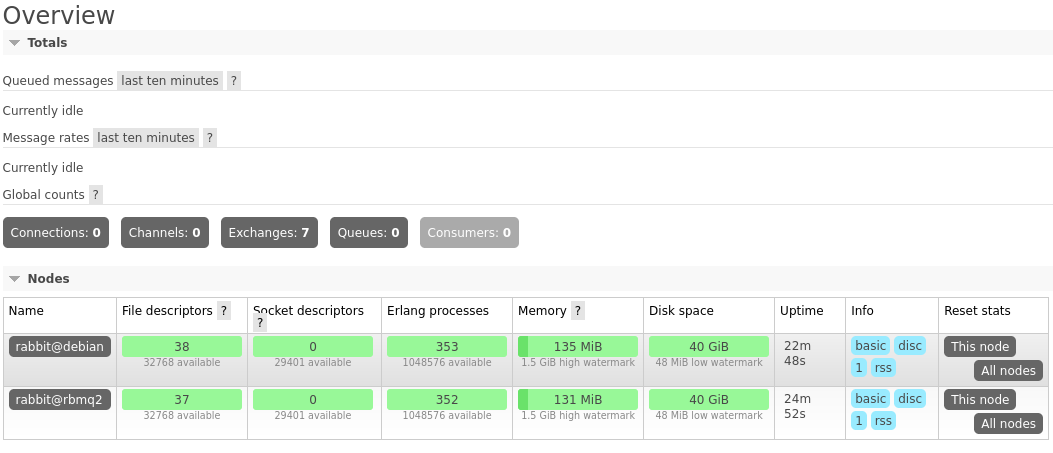


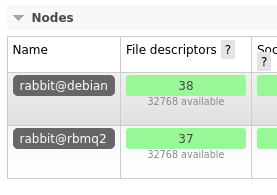






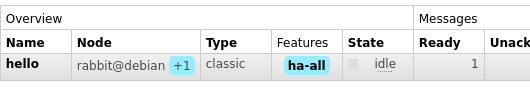
**Кластер с политикой создан**



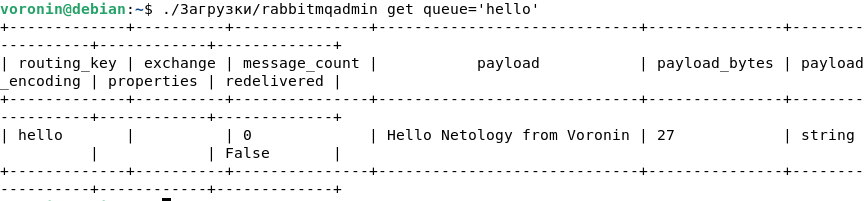


**Для закрепления материала снова запустите скрипт producer.py на ноде 1 (debian):**

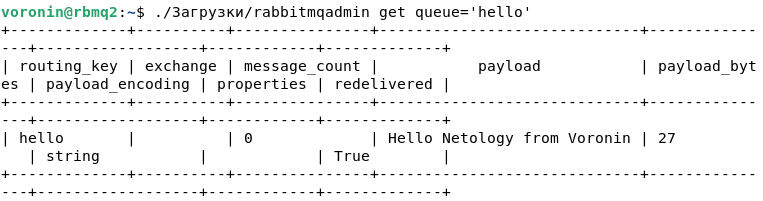




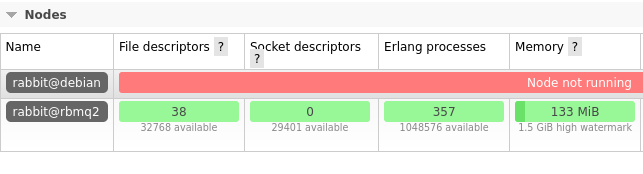
**На ноде1**



**На ноде2**



**Пробуем отключить одну из нод (первую)**



**На второй ноде (rbmq2) запускаем скрипт consumer.py:**



## **Дополнительные задания (со звездочкой\*)**