МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

СЕРГИЕНКО ЛЕВ ЭДУАРДОВИЧ

Отчет по
ЛАБОРАТОРНАЯ РАБОТА 2
ПО ДИСЦИПЛИНЕ
«Непрерывное интегрирование и сборка программного обеспечения»

Коммуникация в многоконтейнерных системах

Преподаватель

Давидовская М.И.
Филиппов М.А.

2025

1. Цель работы

Проектирование распределенных систем и реализация коммуникации между их компонентами.

2. Вариант задания

10	Формирует файл **/var/result/data.txt** из первых строк всех файлов каталога **/var/data**	Ищет наименьшее число из файла **/var/data/data.txt** и сохраняет его третью степень в **/var/result/result.txt**

3. Код приложений, конфигурационных файлов

Задание 1. Разработка простейшего распределенного приложения

Worker 1

```go
package main

import (
    "bufio"
    "fmt"
    "io/fs"
    "os"
    "path/filepath"
    "strconv"
    "strings"
)

const (
    dataDir   = "/var/data"
    resultDir = "/var/result"
)

func main() {
    fmt.Println("Worker1: Начинаю обработку файлов...")

    if err := os.MkdirAll(resultDir, 0755); err != nil {
        fmt.Printf("Ошибка создания каталога %s: %v\n", resultDir, err)
```

```
        os.Exit(1)
    }

    resultFile, err := os.Create(filepath.Join(resultDir, "data.txt"))
    if err != nil {
        fmt.Printf("Ошибка создания файла результата: %v\n", err)
        os.Exit(1)
    }
    defer resultFile.Close()

    err = filepath.WalkDir(dataDir, func(path string, d fs.DirEntry, err
error) error {
        if err != nil {
            return err
        }

        if d.IsDir() {
            return nil
        }

        if strings.HasSuffix(strings.ToLower(path), ".txt") {
            fmt.Printf("Обрабатываю файл: %s\n", filepath.Base(path))

            file, err := os.Open(path)
            if err != nil {
                fmt.Printf("Ошибка открытия файла %s: %v\n", path, err)
                return err
            }
            defer file.Close()

            // Читаем первую строку
            scanner := bufio.NewScanner(file)
            if scanner.Scan() {
                line := strings.TrimSpace(scanner.Text())
                if _, err := strconv.Atoi(line); err == nil {
                    _, err := resultFile.WriteString(line + "\n")
                    if err != nil {
                        fmt.Printf("Ошибка записи в файл результата: %v\n",
err)
                        return err
                    }
                    fmt.Printf("Добавлено число: %s\n", line)
                } else {
                    fmt.Printf("Пропускаю нечисловую строку: %s\n", line)
                }
```

```
            }

            if err := scanner.Err(); err != nil {
                fmt.Printf("Ошибка чтения файла %s: %v\n", path, err)
                return err
            }
        }

        return nil
    })
    if err != nil {
        fmt.Printf("Ошибка при обходе каталога %s: %v\n", dataDir, err)
        os.Exit(1)
    }

    fmt.Println("Worker1: Обработка завершена успешно!")
    fmt.Printf("Результат сохранен в файл: %s\n", filepath.Join(resultDir,
"data.txt"))
}
```

```
FROM golang:1.24-alpine AS builder

WORKDIR /app

COPY main.go .

RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o worker1
main.go

FROM alpine:latest


# Создаем необходимые директории
RUN mkdir -p /var/data /var/result && \
    chown -R appuser:appgroup /var/data /var/result

COPY --from=builder /app/worker1 /usr/local/bin/worker1

RUN chmod +x /usr/local/bin/worker1


WORKDIR /var

CMD ["worker1"]
```

Worker 2

```
package main

import (
    "bufio"
    "fmt"
    "math"
    "os"
    "path/filepath"
    "strconv"
    "strings"
)

const (
    dataFile  = "/var/data/data.txt"
    resultDir = "/var/result"
)

func main() {
    fmt.Println("Worker2: Начинаю поиск минимального числа...")

    if err := os.MkdirAll(resultDir, 0755); err != nil {
        fmt.Printf("Ошибка создания каталога %s: %v\n", resultDir, err)
        os.Exit(1)
    }

    file, err := os.Open(dataFile)
    if err != nil {
        fmt.Printf("Ошибка открытия файла %s: %v\n", dataFile, err)
        os.Exit(1)
    }
    defer file.Close()

    scanner := bufio.NewScanner(file)
    minNumber := math.MaxInt64
    var numbers []int

    for scanner.Scan() {
        line := strings.TrimSpace(scanner.Text())
        if line == "" {
            continue
        }

        number, err := strconv.Atoi(line)
        if err != nil {
```

```
            fmt.Printf("Пропускаю нечисловую строку: %s\n", line)
            continue
        }

        numbers = append(numbers, number)
        if number < minNumber {
            minNumber = number
        }
    }

    if err := scanner.Err(); err != nil {
        fmt.Printf("Ошибка чтения файла: %v\n", err)
        os.Exit(1)
    }

    if len(numbers) == 0 {
        fmt.Println("В файле не найдено ни одного числа!")
        os.Exit(1)
    }

    cube := minNumber * minNumber * minNumber

    fmt.Printf("Найденные числа: %v\n", numbers)
    fmt.Printf("Минимальное число: %d\n", minNumber)
    fmt.Printf("Третья степень минимального числа: %d\n", cube)

    resultFile, err := os.Create(filepath.Join(resultDir, "result.txt"))
    if err != nil {
        fmt.Printf("Ошибка создания файла результата: %v\n", err)
        os.Exit(1)
    }
    defer resultFile.Close()

    _, err = resultFile.WriteString(fmt.Sprintf("%d\n", cube))
    if err != nil {
        fmt.Printf("Ошибка записи в файл результата: %v\n", err)
        os.Exit(1)
    }

    fmt.Printf("Результат сохранен в файл: %s\n", filepath.Join(resultDir,
"result.txt"))
    fmt.Println("Worker2: Обработка завершена успешно!")
}
```

```
FROM golang:1.24-alpine AS builder
```

```
WORKDIR /app

COPY main.go .

RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o worker2
main.go

FROM alpine:latest

RUN mkdir -p /var/data /var/result && \
    chown -R appuser:appgroup /var/data /var/result

COPY --from=builder /app/worker2 /usr/local/bin/worker2

RUN chmod +x /usr/local/bin/worker2

WORKDIR /var

CMD ["worker2"]
```

Docker compose

```
services:
    # Первое приложение - формирует data.txt из первых строк всех файлов
каталога /var/data
    worker1:
        build:
            context: ./worker1
            dockerfile: Dockerfile
        container_name: worker1-container
        volumes:
            # Монтируем каталог ./data в /var/data для чтения исходных
данных
            - ./data:/var/data:ro
            # Монтируем каталог ./result-1 в /var/result для записи
промежуточных данных
            - ./result-1:/var/result
        networks:
            - app-network
        restart: "no"

    # Второе приложение - находит минимальное число и возводит в 3 степень
    worker2:
        build:
            context: ./worker2
```

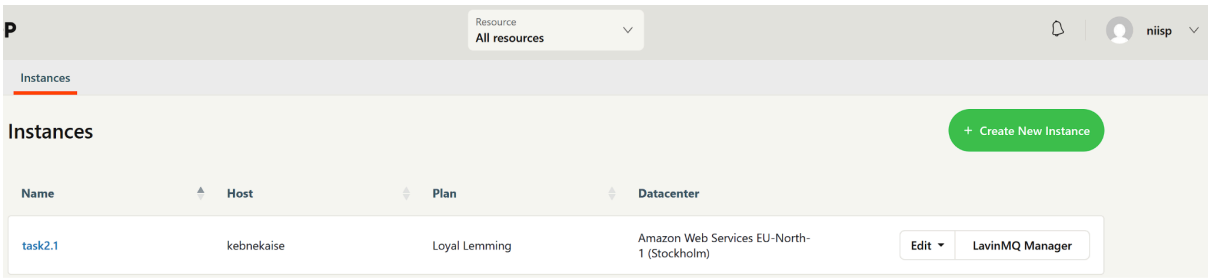
```
            dockerfile: Dockerfile
        container_name: worker2-container
        volumes:
            # Монтируем каталог ./result-1 в /var/data для чтения
промежуточных данных
            - ./result-1:/var/data:ro
            # Монтируем каталог ./result в /var/result для записи
финального результата
            - ./result:/var/result
        networks:
            - app-network
        # Зависимость от worker1 - worker2 запустится только после
завершения worker1
        depends_on:
            - worker1
        restart: "no"

networks:
    app-network:
        driver: bridge
```

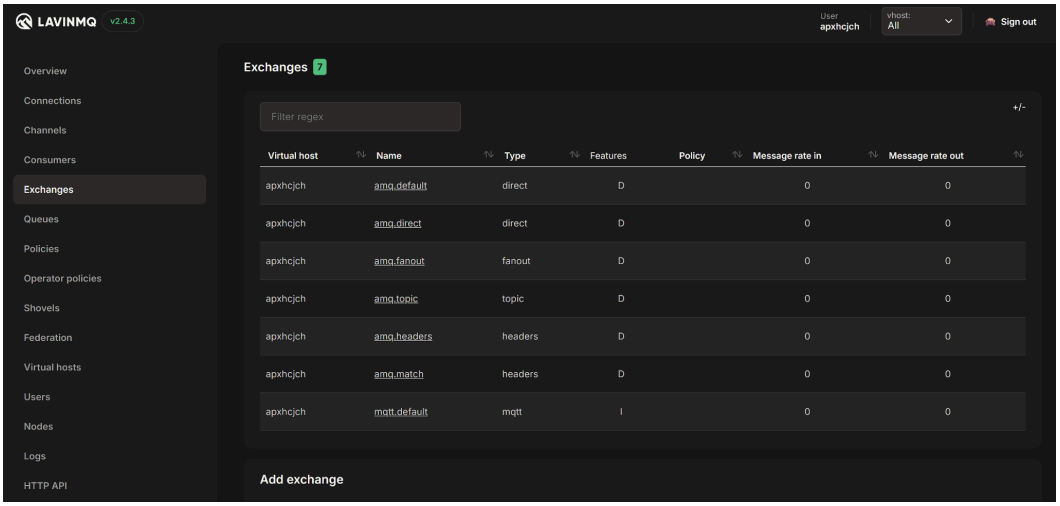
Результат

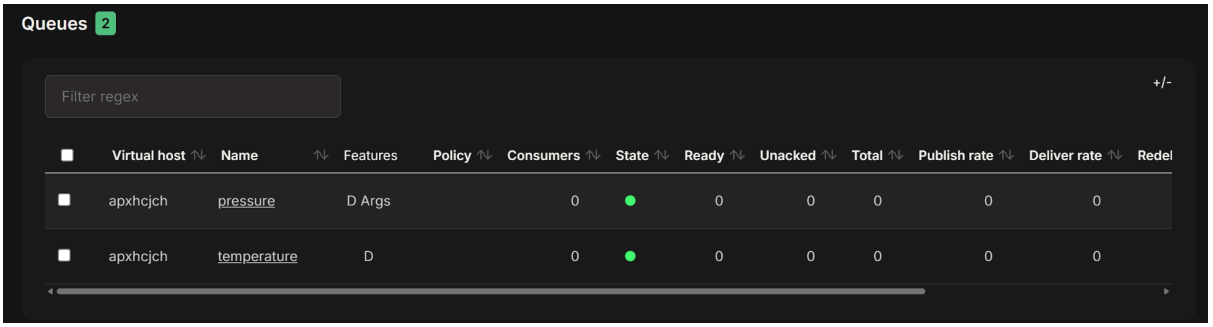
```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP\task1> docker compose up
 => [worker2] resolving provenance for metadata file                                0.0s
[+] Running 3/3
 ✓ Network task1_app-network    Created                                            0.1s
 ✓ Container worker1-container  Created                                            0.1s
 ✓ Container worker2-container  Created                                            0.1s
Attaching to worker1-container, worker2-container
worker1-container  | Worker1: Начинаю обработку файлов...
worker1-container  | Обрабатываю файл: file1.txt
worker1-container  | Добавлено число: 15
worker1-container  | Обрабатываю файл: file2.txt
worker1-container  | Добавлено число: 8
worker1-container  | Обрабатываю файл: file3.txt
worker1-container  | Добавлено число: 42
worker1-container  | Worker1: Обработка завершена успешно!
worker1-container  | Результат сохранен в файл: /var/result/data.txt
worker1-container exited with code 0
worker2-container  | Worker2: Начинаю поиск минимального числа...
worker2-container  | Найденные числа: [15 8 42]
worker2-container  | Минимальное число: 8
worker2-container  | Третья степень минимального числа: 512
worker2-container  | Результат сохранен в файл: /var/result/result.txt
worker2-container  | Worker2: Обработка завершена успешно!
worker2-container exited with code 0
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP\task1>
```

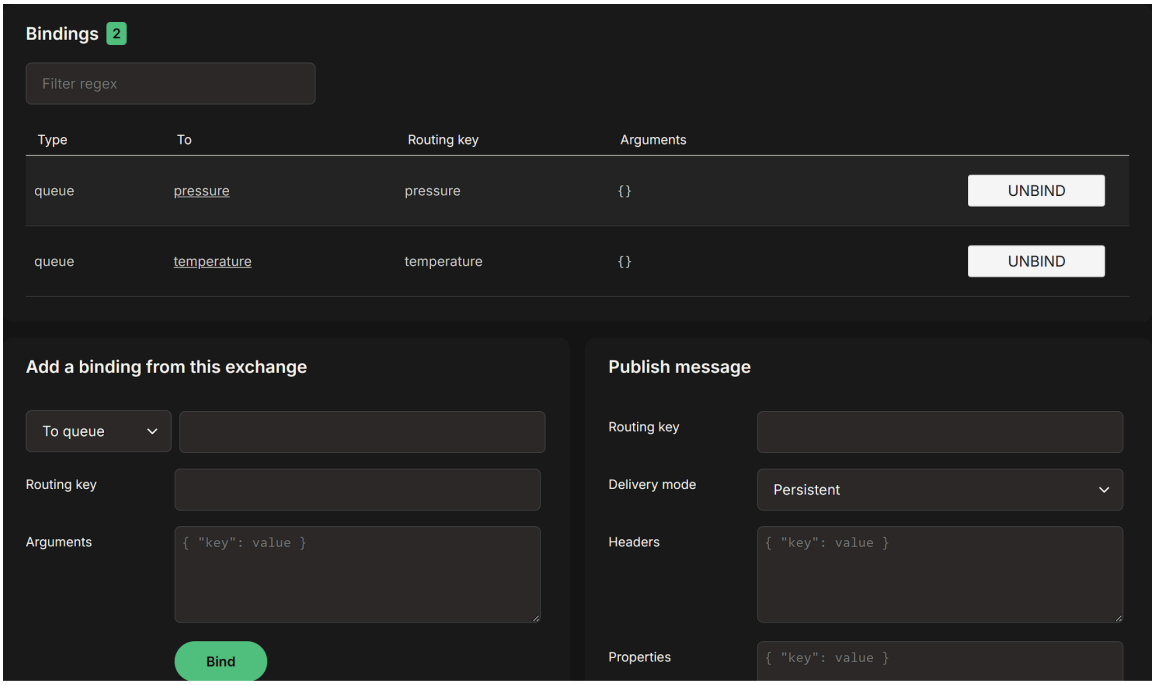
Задание 2. Асинхронная коммуникация в распределенных системах на основе брокера сообщений

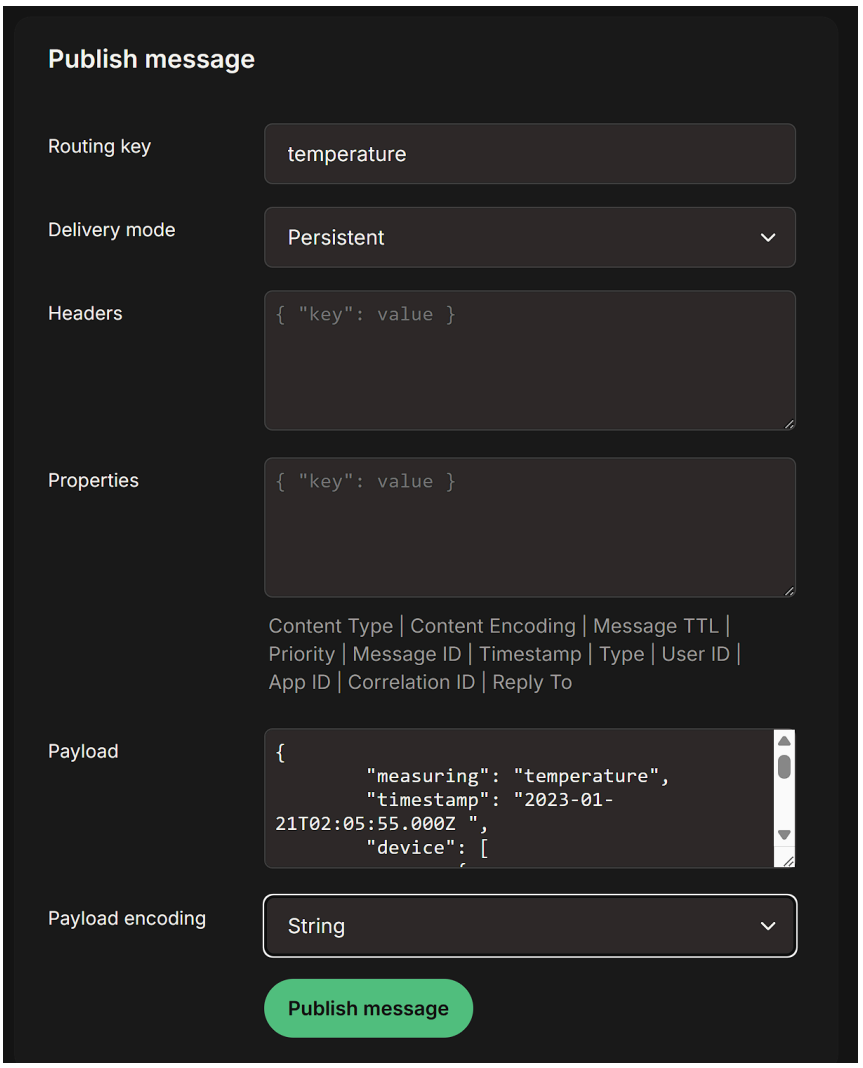
Упражнение 2.1

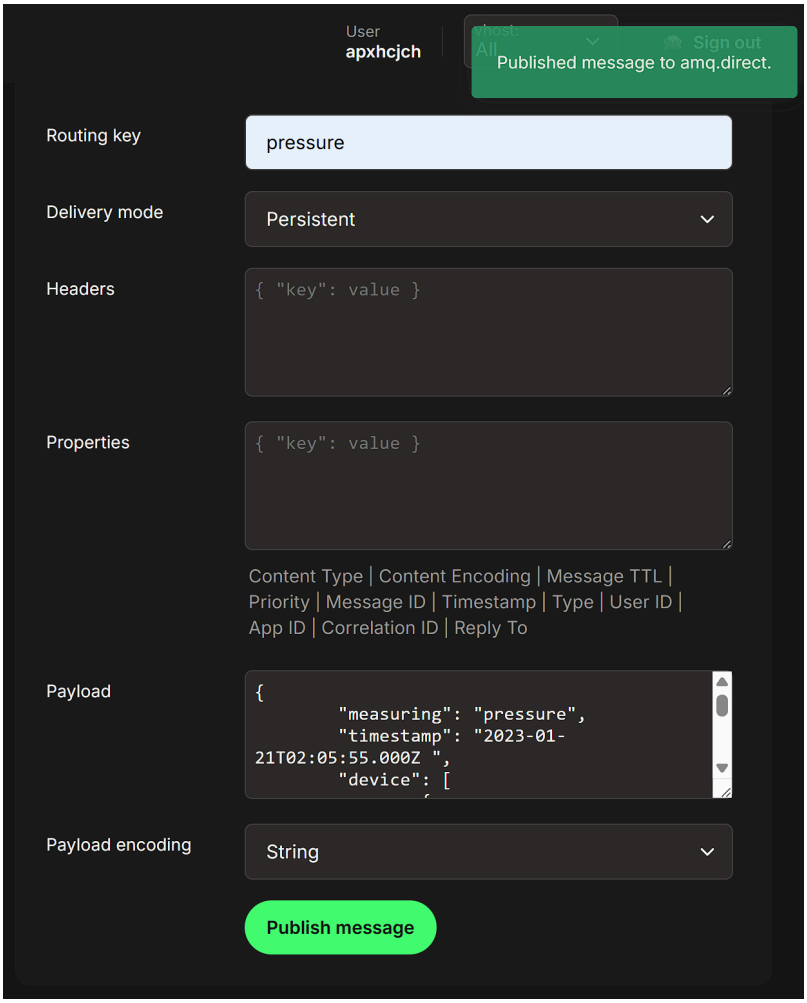


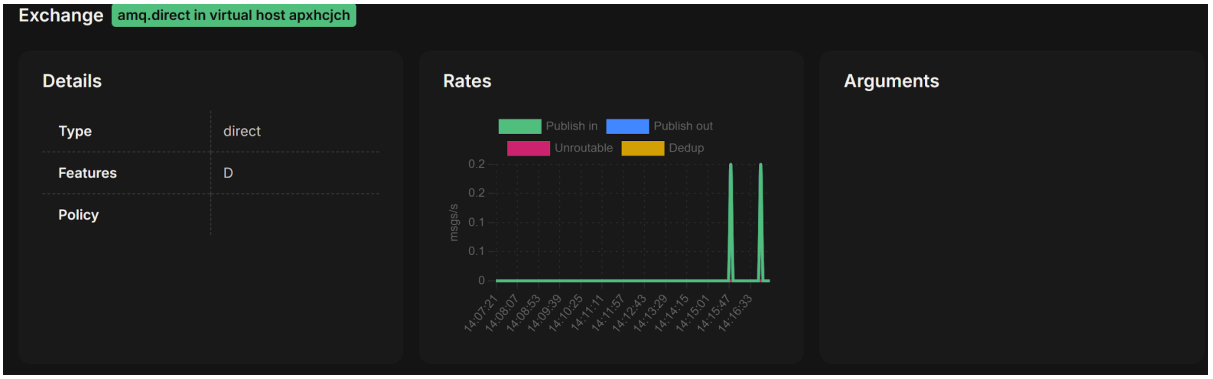



Queues 2
Filter regex
+/-
Virtual host
Name
Features
Policy
Consumers
State
Ready
Unacked
Total
Publish rate
Deliver rate
Redel
apxhcjch
pressure
D Args
0
0
0
0
0
0
apxhcjch
temperature
D
0
0
0
0
0
0


Bindings 2
Filter regex
Type
To
Routing key
Arguments
queue
pressure
pressure
{}
UNBIND
queue
temperature
temperature
{}
UNBIND
Add a binding from this exchange
To queue
Routing key
Arguments
{ "key": value }
Bind
Publish message
Routing key
Delivery mode
Persistent
Headers
{ "key": value }
Properties
{ "key": value }


Publish message
Routing key
temperature
Delivery mode
Persistent
Headers
{ "key": value }
Properties
{ "key": value }
Content Type | Content Encoding | Message TTL | Priority | Message ID | Timestamp | Type | User ID | App ID | Correlation ID | Reply To
Payload
{
"measuring": "temperature",
"timestamp": "2023-01-21T02:05:55.000Z ",
"device": [
Payload encoding
String
Publish message


User
apxhcjch
Vhost:
All
Sign out
Published message to amq.direct.
Routing key
pressure
Delivery mode
Persistent
Headers
{ "key": value }
Properties
{ "key": value }
Content Type | Content Encoding | Message TTL | Priority | Message ID | Timestamp | Type | User ID | App ID | Correlation ID | Reply To
Payload
{
"measuring": "pressure",
"timestamp": "2023-01-21T02:05:55.000Z ",
"device": [
Payload encoding
String
Publish message


Exchange
amq.direct in virtual host apxhcjch
Details
Type
direct
Features
D
Policy
Rates
Publish in
Publish out
Unroutable
Dedup
msg/s
0.2
0.2
0.1
0.1
0
14:07:21
14:08:07
14:08:53
14:09:39
14:10:25
14:11:11
14:11:57
14:12:43
14:13:29
14:14:15
14:15:01
14:15:47
14:16:33
Arguments

Queue / pressure in virtual host apxhcjch

Details

Features	D
State	running
Consumers	0
Policy	
Operator policy	apxhcjch-max-length
Effective policy definition	max-length: 20000

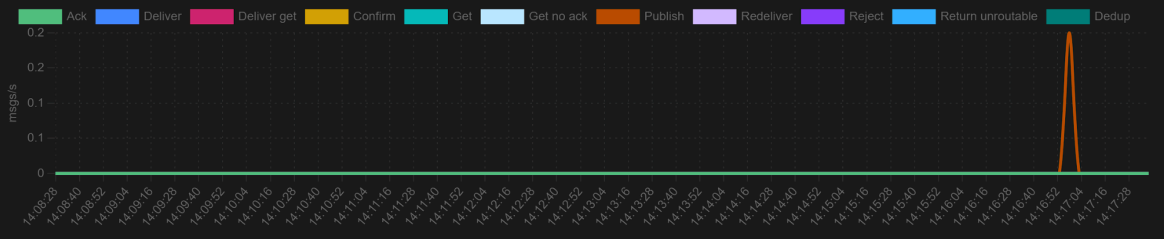
Messages

	Count	Bytesize	Average
Total	1	403B	403B
Ready	1	403B	403B
Unacked	0	0B	0B

Arguments

- x-max-priority: 5

Rates



Message 1

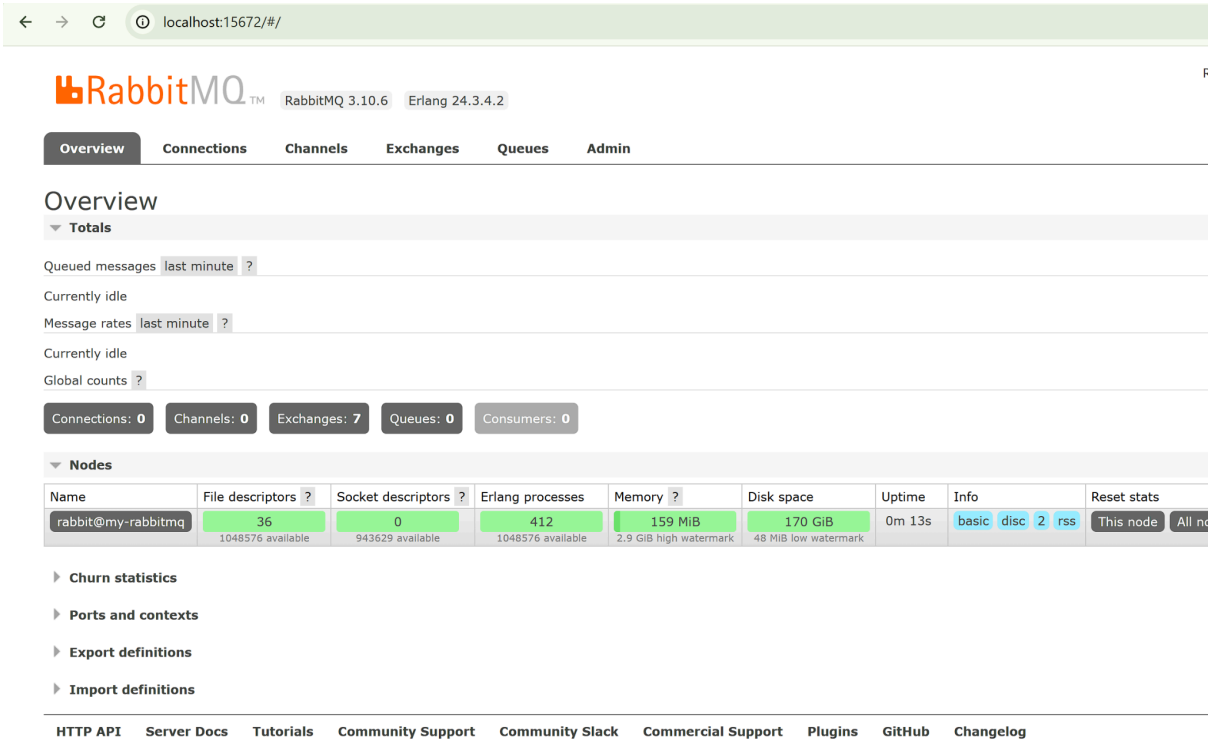
The server reported 0 messages remaining.

Exchange	amq.direct
Routing key	pressure
Redelivered	true
Properties	{"headers":{},"delivery_mode":2}

Payload
360 bytes
Encoding: string


```
{
        "measuring": "pressure",
        "timestamp": "2023-01-21T02:05:55.000Z ",
        "device": [
                {
                        "device_id": 4123,
                        "value": 9.67,
                        "status": "OK"
                },
                {
                        "device_id": 587,
                        "value": 1.03,
                        "status": "ERROR"
                },
                {
                        "device_id": 1524,
                        "value": 6.34,
                        "status": "OK"
                },
                {
                        " device_id": 97,
                        "value": 0.21,
                        "status": "ERROR"
                }
        ]
}
```


Упражнение 2.2

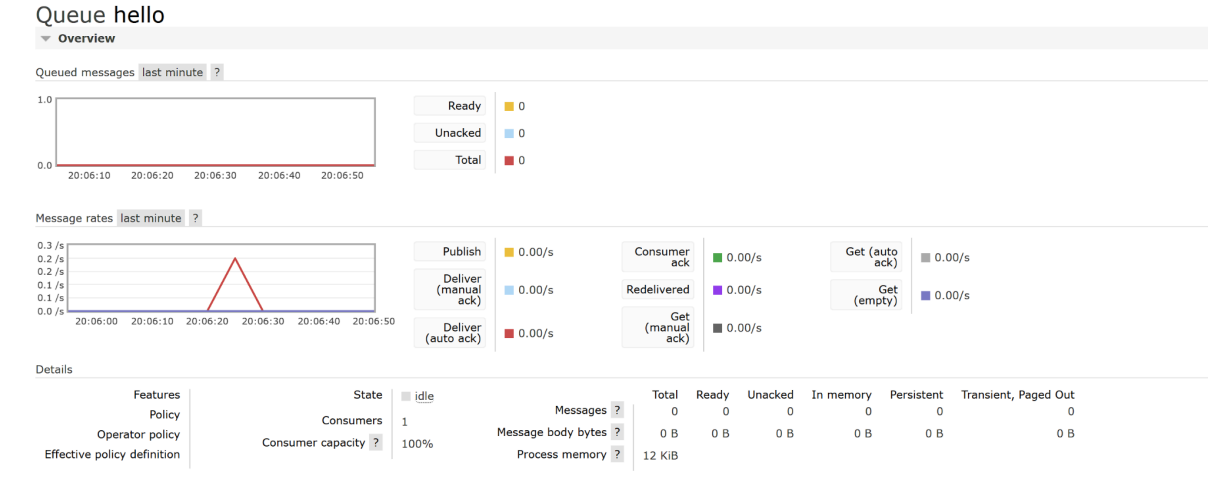


Доступ к интерфейсу администрирования

Урок 1

```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP\task2-2\task1> go run .\receive.go
2025/09/29 20:06:23  [*] Waiting for messages. To exit press CTRL+C
2025/09/29 20:06:26 Received a message: Hello World!
```

```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP\task2-2\task1> go run .\send.go
2025/09/29 20:06:26  [x] Sent Hello World!
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP\task2-2\task1>
```



Урок 2

```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP\task2-2\task2\new_task> go run . wrfrfrffrfrfl
2025/09/30 01:02:14  [x] Sent wrfrfrffrfrfl
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP\task2-2\task2\new_task>
```

```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP\task2-2\task2\worker> go run .
2025/09/30 01:02:09  [*] Waiting for messages. To exit press CTRL+C
2025/09/30 01:02:14 Received a message: wrfrfrffrfrfl
2025/09/30 01:02:14 Done
```

Урок 3

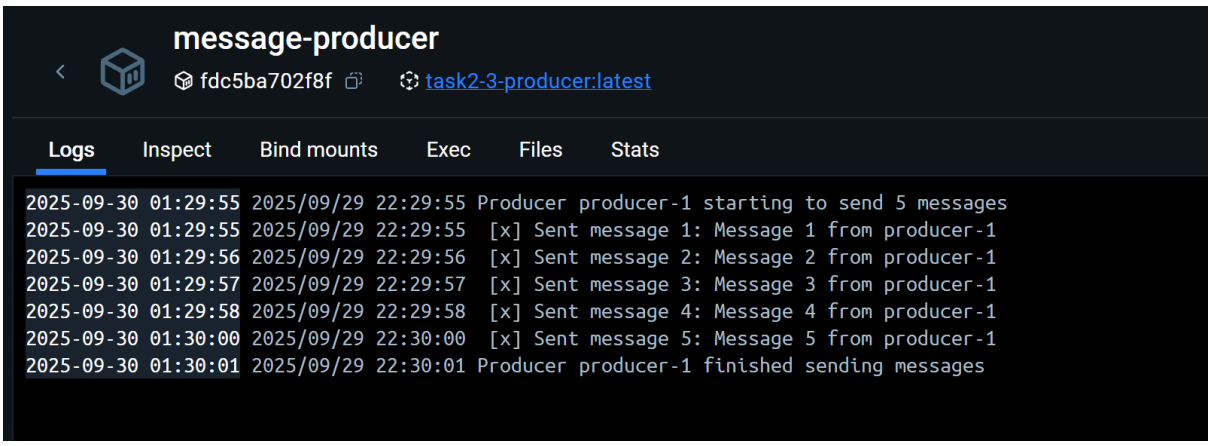
```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP> cd task2-2/task3/emit_log
>> go run . "Your log message here"
2025/09/30 01:04:10  [x] Sent Your log message here
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP\task2-2\task3\emit_log>
```

```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP> cd task2-2/task3/receive_logs
>> go run .
2025/09/30 01:04:06  [*] Waiting for logs. To exit press CTRL+C
2025/09/30 01:04:10  [x] Your log message here
```

Упражнение 2.3

```
"Starting RabbitMQ distributed system..."
docker compose up -d
[+] Running 6/6
 ✓ Network task2-3_rabbitmq_network  Created
 ✓ Volume "task2-3_rabbitmq_data"    Created
 ✓ Container rabbitmq-server         Healthy
 ✓ Container message-producer        Started
 ✓ Container message-consumer-1      Started
 ✓ Container message-consumer-2      Started
"System started! Management UI available at http://localhost:15672"
"Username: admin, Password: admin"
```

```
PS C:\Users\lev\Desktop\devops\devops-gr12b-lab2-foreverNP\task2-3> make status
"=== System Status ==="
docker compose ps
NAME                IMAGE                COMMAND                  SERVICE      CREATED          STATUS                    PORTS
message-consumer-1   task2-3-consumer-1   "./main"                 consumer-1   51 seconds ago   Up 21 seconds
message-consumer-2   task2-3-consumer-2   "./main"                 consumer-2   51 seconds ago   Up 21 seconds
rabbitmq-server      task2-3-rabbitmq     "docker-entrypoint.s…"   rabbitmq     51 seconds ago   Up 49 seconds (healthy)   4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, 15671/tcp, 15691-15
692/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp
""
"=== RabbitMQ Queues ==="
docker compose exec rabbitmq rabbitmqctl list_queues name messages consumers
Timeout: 60.0 seconds ...
Listing queues for vhost / ...
name    messages        consumers
task_queue      0       2
""
"=== RabbitMQ Exchanges ==="
docker compose exec rabbitmq rabbitmqctl list_exchanges name type
Listing exchanges for vhost / ...
name    type
amq.rabbitmq.trace      topic
amq.topic       topic
amq.direct      direct
amq.headers     headers
amq.match       headers
amq.fanout      fanout
        direct
""
"=== RabbitMQ Bindings ==="
docker compose exec rabbitmq rabbitmqctl list_bindings
Listing bindings for vhost /...
source_name     source_kind     destination_name        destination_kind        routing_key     arguments
        exchange        task_queue      queue   task_queue      []
```



4. Ответы на контрольные вопросы

1. Обмен сообщениями (messaging) - передача данных между приложениями/сервисами через посредника (broker) в виде сообщений, асинхронно и надёжно.
2. Брокер сообщений - сервер, который принимает, хранит, маршрутизирует и доставляет сообщения между приложениями (продюсерами и потребителями).
3. RabbitMQ - популярный брокер сообщений с открытым исходным кодом, реализующий очереди, обменники и механизмы маршрутизации/гарантий доставки.
4. Протокол в основе RabbitMQ - AMQP.
5. Сообщения, принятые от продюсера, записываются в exchange и затем (в зависимости от биндингов) попадают в одну или несколько очередей (queues).
6. Для простого распараллеливания обработки - не мультикаст, а обычная очередь (work queue): отправлять в одну очередь (через default/direct exchange) и запускать несколько потребителей; т.е.

использовать схему «одна очередь - несколько потребителей» (не fanout).

7. Обычно продюсер отправляет сообщение в exchange, часто указывая routing key.
8. Чтобы избежать потребления устаревших данных - назначить TTL для сообщений/очереди и/или использовать dead-letter exchange; дополнительно потребитель может проверять метку времени в сообщении.
9. Очереди в RabbitMQ по умолчанию устроены по принципу FIFO (первый пришёл - первый обработан).
10. Binding - связь между exchange и очередью, задающая условия маршрутизации.
11. Routing Key - ключ маршрутизации, который продюсер указывает при публикации; exchange использует его совместно с биндингами для определения, в какие очереди направить сообщение.
12. Точка обмена - компонент брокера, принимающий публикации от продюсеров и маршрутизирующий сообщения в очереди по заданным правилам (тип exchange определяет логику маршрутизации).
13. Процесс работы:
    - Продюсер подключается к брокеру и публикует сообщение в exchange с routing key;
    - Exchange сравнивает routing key с биндингами и выбирает целевые очереди;
    - Сообщение помещается в выбранные очереди (и при необходимости сохраняется на диск);
    - Потребители подключаются к очередям и получают (получают push/consume) сообщения;
    - Потребитель подтверждает (ack) или отклоняет (nack) сообщение; при nack/таймауте сообщение может быть переотправлено или отправлено в DLX.
14. Server в RabbitMQ (broker) - отвечает за приём, хранение, маршрутизацию и доставку сообщений; управляет соединениями, кластеризацией, персистентностью, авторизацией и интерфейсом администрирования.

15. Vhost (виртуальный хост) - логическая область в RabbitMQ для разделения ресурсов (очереди, exchange, биндинги) и прав доступа; используется для многоарендности и изоляции.