



МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«МИРЭА – Российский технологический университет»  
**РТУ МИРЭА**

**МАТЕРИАЛЫ ПРАКТИЧЕСКИХ ЗАНЯТИЙ**

**Технологии хранения в системах кибербезопасности**

	<i>(наименование дисциплины (модуля) в соответствии с учебным планом)</i>
Уровень	специалитет
	<i>(бакалавриат, магистратура, специалитет)</i>
Форма обучения	очная
	<i>(очная, очно-заочная, заочная)</i>
Направление(-я) подготовки	10.05.04 Информационно-аналитические системы безопасности
	<i>(код(-ы) и наименование(-я))</i>
Институт	Кибербезопасности и цифровых технологий (ИКБ)
	<i>(полное и краткое наименование)</i>
Кафедра	КБ-2 «Информационно-аналитические системы кибербезопасности»
	<i>(полное и краткое наименование кафедры, реализующей дисциплину (модуль))</i>
Лектор	к.т.н., Селин Андрей Александрович, Бугаев Александр Александрович
	<i>(сокращенно – ученая степень, ученое звание; полностью – ФИО)</i>
Используются в данной редакции с учебного года	2024/2025
	<i>(учебный год цифрами)</i>
Проверено и согласовано «___» _____ 2024 г.	А.А. Бакаев
	<i>(подпись директора Института/Филиала с расшифровкой)</i>

Москва 2024 г.

## ПРАКТИЧЕСКАЯ РАБОТА № 6

### «Знакомство с NoSQL-хранилищем типа «ключ-значение». Использование Redis»

**Цель работы** – получение практических навыков развертывания и использования СУБД Redis.

#### **Задание:**

1. Запустите Unix-подобную систему (например, Debian 12.6.0 64-bit<sup>1</sup>).

2. Создайте пользователя с именем формата **fio\_nn**,  
где **f** – первая буква фамилии на латинице;  
**i** – первая буква имени на латинице;  
**o** – первая буква отчества на латинице (при наличии),  
**nn** – двузначный номер по списку в группе.

Добавьте его в группу **sudo**. **Все дальнейшие действия необходимо выполнять от имени созданного пользователя.**

3. Запустите терминал и установите Docker и Docker Compose.

4. Создайте каталог для нового проекта (например, **redis**) и сформируйте файл **docker-compose.yml** для развертывания Redis и RedisInsight (рекомендуется использовать **Redis Stack** – <https://hub.docker.com/r/redis/redis-stack>).

#### **Требования к запускаемым сервисам:**

– последние 2 цифры номера порта, на котором будет развернут сервис, должны соответствовать номеру по списку в группе (например, для 15 – 12315, 8015, 9915 и т.п.);

– имя контейнера должно заканчиваться на символ подчеркивания и инициалы ФИО (например, для Иванова Петра Дмитриевича – **redis\_ipd**).

5. Разверните Redis и RedisInsight (Redis Stack) с помощью Docker Compose.

---

<sup>1</sup> Можно скачать готовый образ виртуальной машины по ссылке  
<https://sourceforge.net/projects/osboxes/files/v/vb/14-D-b/12.6.0/64bit.7z/download>

В пунктах 6-10 названия ключей должны заканчиваться на символ подчеркивания и инициалы ФИО (например, для Иванова Петра Дмитриевича – test\_key\_ipd, name\_ipd, size\_ipd и т.п.).

6. Изучите команды по созданию, выборке, модификации, удалению и получению базовой информации об объектах с использованием интерфейса командной строки (CLI):

- set
- get
- getset
- type
- rename
- exists
- del
- keys

Пример выполнения команд set и get:

The screenshot displays the Redis Stack Database interface. At the top, the breadcrumb navigation shows 'Databases > My Redis Stack Database db0'. Below this, there's a search bar with 'All Key Types' and a filter option 'Filter by Key Name or Pattern'. The main content area shows a list of keys with 'Total: 1' and 'Last refresh: < 1 min'. The key 'test\_key\_ipd' is highlighted, showing it is a 'STRING' type with 'No limit' and '72 B' size. To the right, a detailed view of the key shows 'test\_value' and metadata: 'Key Size: 72 B', 'Length: 10', and 'TTL: No limit'. At the bottom, the CLI window shows the execution of the following commands:

```
> _ CLI
> set test_key_ipd test_value
"OK"
> get test_key_ipd
"test_value"
>
```

The bottom status bar includes links for '> \_ CLI', 'Command Helper', and 'Profiler'.

7. Изучите возможность использования параметра времени жизни объекта (TTL). Изучите команды **ttl** и **expire**.
8. Изучите основные строковые операции:
- **append**
  - **strlen**
  - **getrange**
  - **setrange**
9. Изучите операции над числами:
- **incr**
  - **decr**
  - **incrby**
  - **decrby**
10. Изучите основные операции над списками:
- **rpush**
  - **lrange**
  - **llen**
  - **lpop**
11. Изучите возможности RedisInsight.
12. Разверните тестовый сервис авторизации на веб-странице с хранением информации о количестве неудачных попыток в Redis. При 3 и более неудачных попытках авторизации для конкретного пользователя необходимо на 60 секунд приостанавливать возможность авторизации. Ниже приведен пример такого сервиса (пункты 12.1-12.5).
- 12.1. Создайте папку **app** в каталоге с вашим проектом со следующей структурой:

```
app
├── app.py
├── requirements.txt
├── templates
│   └── login.html
```

Содержимое файла **app.py** (код приложения с использованием Flask):

```
from flask import Flask, render_template, request, redirect, url_for
import redis

app = Flask(__name__)
r = redis.Redis(host='redis', port=6379, db=0)

@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        attempts_count = 0
        if (r.exists(username)):
            attempts_count = int(r.get(username))
            if (attempts_count > 2):
                r.set(username, attempts_count + 1, ex=60)
                return redirect(url_for('fail'))

        # Проверка имени пользователя и пароля
        if username == 'admin' and password == 'qwerty123':
            if (r.exists(username)):
                r.delete(username)
            return redirect(url_for('success'))
        else:
            attempts_count += 1
            r.set(username, attempts_count, ex=60)
            if (attempts_count > 2):
                return redirect(url_for('fail'))
            else:
                return render_template('login.html', error='Неверное имя пользователя или пароль')

    return render_template('login.html')

@app.route('/success')
def success():
    return 'Успешная авторизация!'

@app.route('/fail')
def fail():
    return 'Превышено количество попыток! Повторите через 1 минуту.'

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Необходимо указать свои параметры подключения к Redis, а также параметры авторизации (имя пользователя должно соответствовать инициалам ФИО и номеру по списку (например, ipd\_15)).

Содержимое файла **requirements.txt** (требуемые пакеты):

```
Flask==3.0.3
redis==5.2.0
```

Содержимое файла **login.html** (HTML-разметка страницы авторизации):

```
<!DOCTYPE html>
<html>
<head>
  <title>Login</title>
</head>
<body>
  <h2>Login</h2>
  {% if error %}
    <p style="color: red;">{{ error }}</p>
  {% endif %}
  <form method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required><br><br>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required><br><br>

    <input type="submit" value="Login">
  </form>
</body>
</html>
```

12.2. Создайте Dockerfile в каталоге с вашим проектом со следующим содержимым:

```
FROM python:3.9-alpine

WORKDIR /app

COPY ./app /app

RUN pip install -r requirements.txt

ENTRYPOINT ["python"]

CMD ["app.py"]
```

Пример итоговой структуры вашего проекта:

```
├── Dockerfile
├── app
│   ├── app.py
│   ├── requirements.txt
│   └── templates
│       └── login.html
└── docker-compose.yml
```

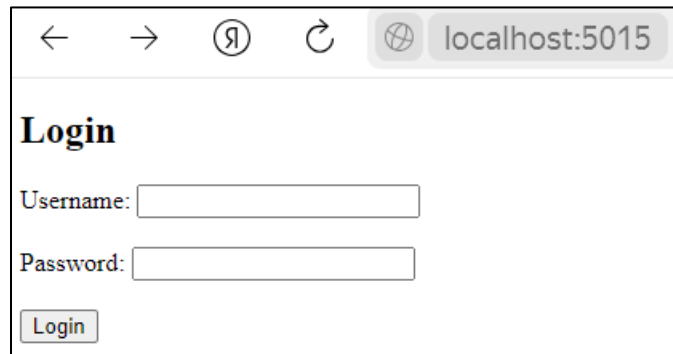
12.3. Сформируйте образ. Пример команды:

```
sudo docker build -t login-app .
```

12.4. Добавьте в ваш файл **docker-compose.yml** новый сервис на основе созданного образа. Пример:

```
login-app:
  image: login-app:latest
  container_name: login-app_ipd
  ports:
    - 5015:5000
  networks:
    - redis-net
```

12.5. Разверните тестовый сервис авторизации с помощью Docker Compose.



← → ↻ ↺ 🌐 localhost:5015

**Login**

Username:

Password:

Login

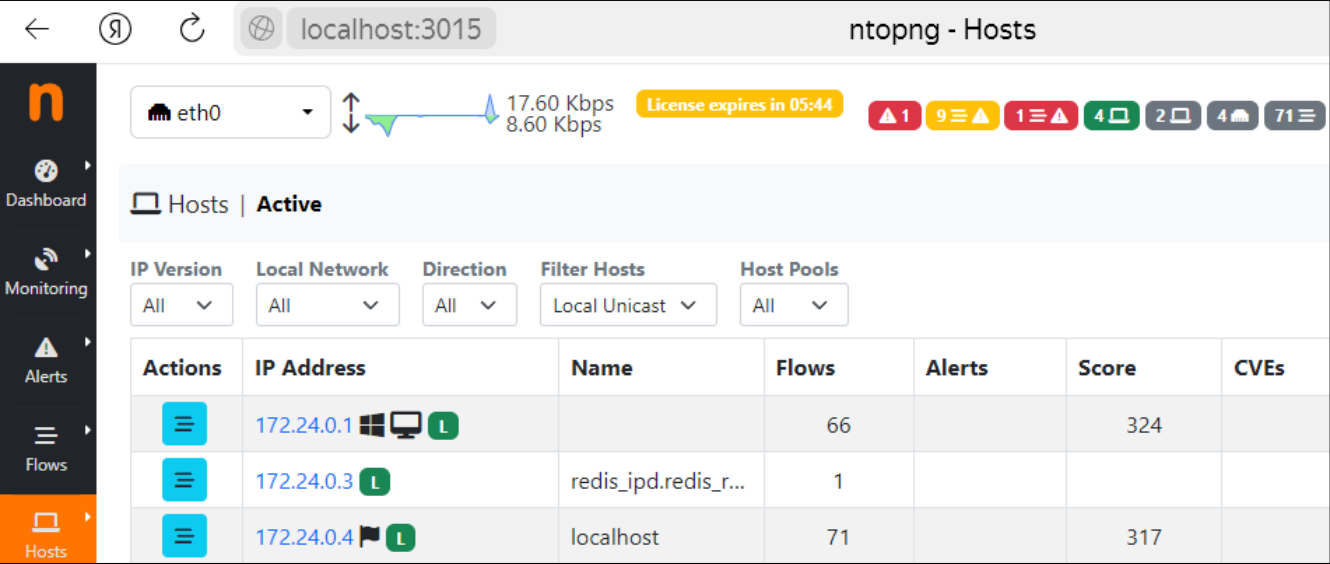
13. Протестируйте работу сервиса. Проанализируйте данные в Redis с помощью RedisInsight для трех случаев:

- неудачная попытка авторизации;
- превышено количество попыток авторизации;
- успешная авторизация.

14. Добавьте в ваш файл **docker-compose.yml** сервис для мониторинга трафика в компьютерной сети **ntopng** на основе образа **ntop/ntopng:stable** (<https://hub.docker.com/r/ntop/ntopng>). Пример:

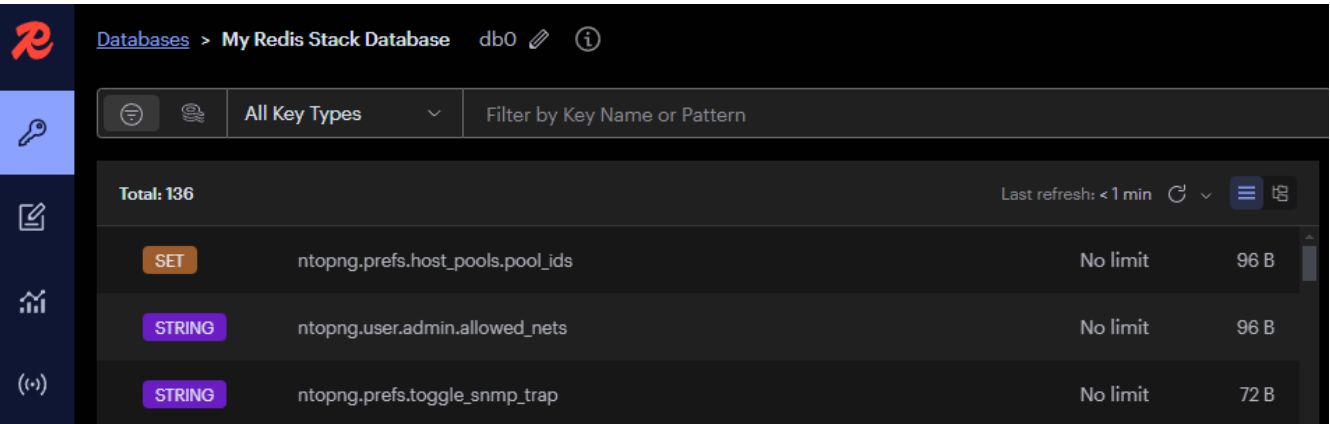
```
ntopng:
  image: ntop/ntopng:stable
  container_name: ntopng_ipd
  command: ["ntopng", "--redis", "redis", "--interface", "eth0"]
  volumes:
    - ./ntopng.license:/etc/ntopng.license:ro
  ports:
    - "3015:3000"
  networks:
    - redis-net
  depends_on:
    - redis
```

15. Запустите сервис **ntopng** с помощью Docker Compose.



16. Изучите возможности **ntopng**.

17. Проанализируйте с помощью RedisInsight, какие ключи (их типы и значения) формируются в Redis в результате работы ntopng.



18. Дополнительно изучите в Redis:

- основные операции над множествами;
- основные операции над упорядоченными множествами;
- транзакции;
- основные операции над хешами.