

**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

|  |
| --- |
| **РТУ МИРЭА** |
|  |
| **Институт кибербезопасности и цифровых технологий (ИКБ)** |
|  |
| КБ-2 «Информационно-аналитические системы кибербезопасности» |

**ОТЧЕТ О ВЫПОЛНЕНИИ ИНДИВИДУАЛЬНОГО ЗАДАНИЯ №6**

# **В РАМКАХ ДИСЦИПЛИНЫ «ТЕХНОЛОГИИ ХРАНЕНИЯ В СИСТЕМАХ КИБЕРБЕЗОПАСНОСТИ»**

Выполнил:

Студент 3-ого курса

Учебной группы БИСО-02-22

Зубарев В.С.

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

Файл docker-compose:

version: '3.8'

networks:

redis:

name: redis

driver: bridge

services:

redis-stack:

image: redis/redis-stack:latest

container\_name: redis\_zvs

networks:

- redis

ports:

- "6305:6379"

- "60005:8001"

environment:

REDIS\_ARGS: "--requirepass adminzvs"

restart: unless-stopped

redis-insight:

image: redislabs/redisinsight:latest

container\_name: insight\_zvs

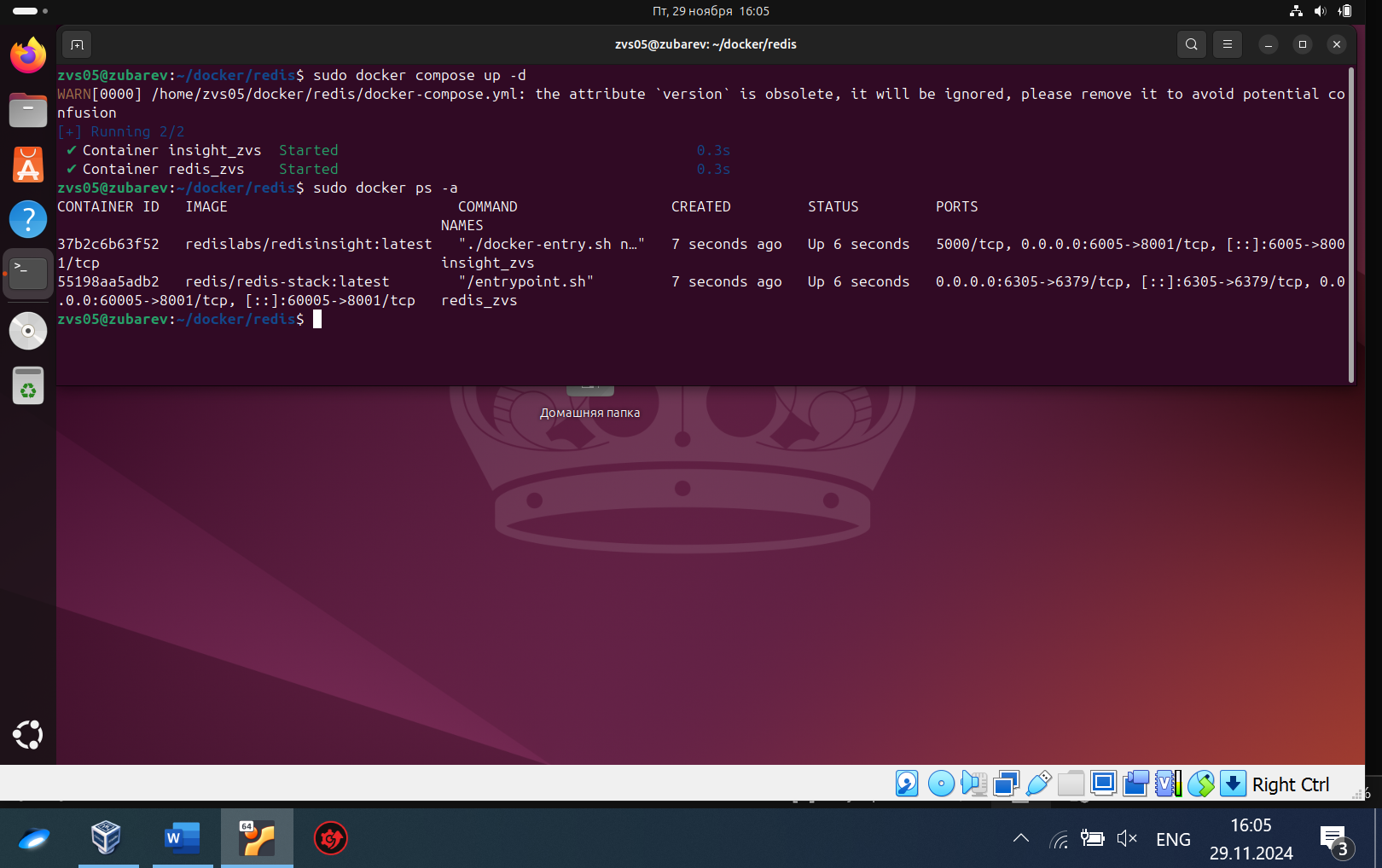
networks:

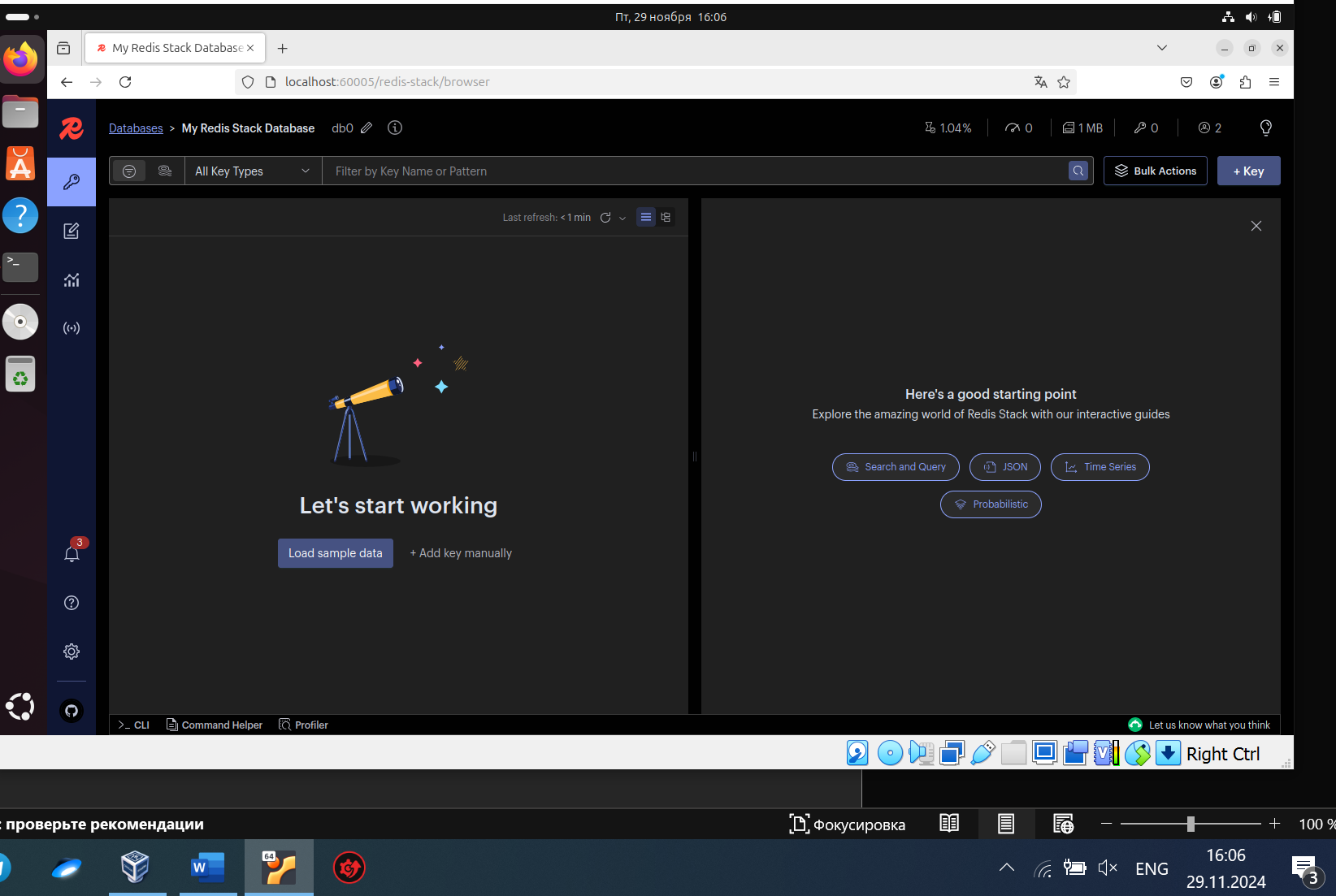
- redis

ports:

- "6005:8001"

restart: unless-stopped





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

− set

− get

− getset

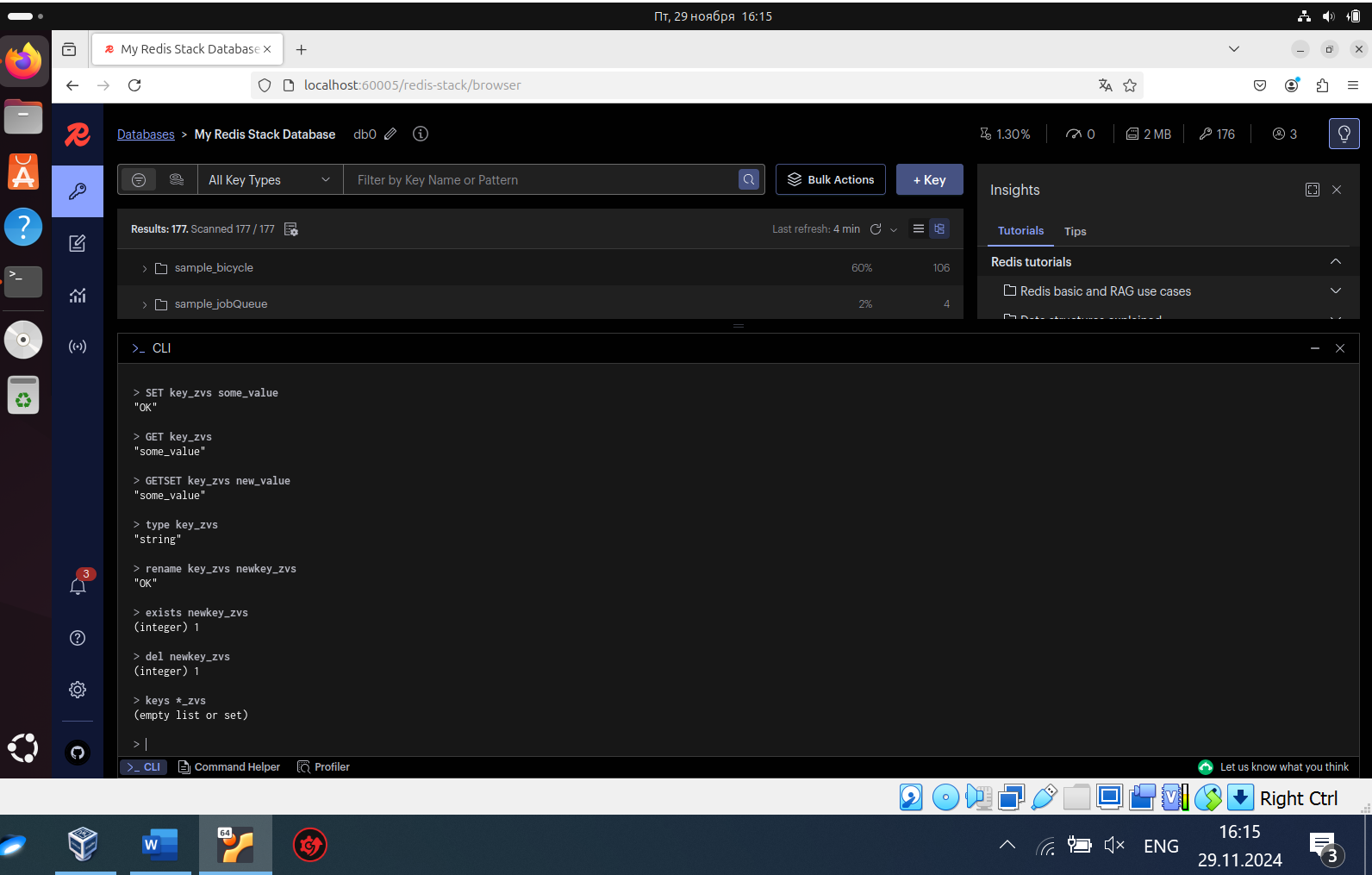
− type

− rename

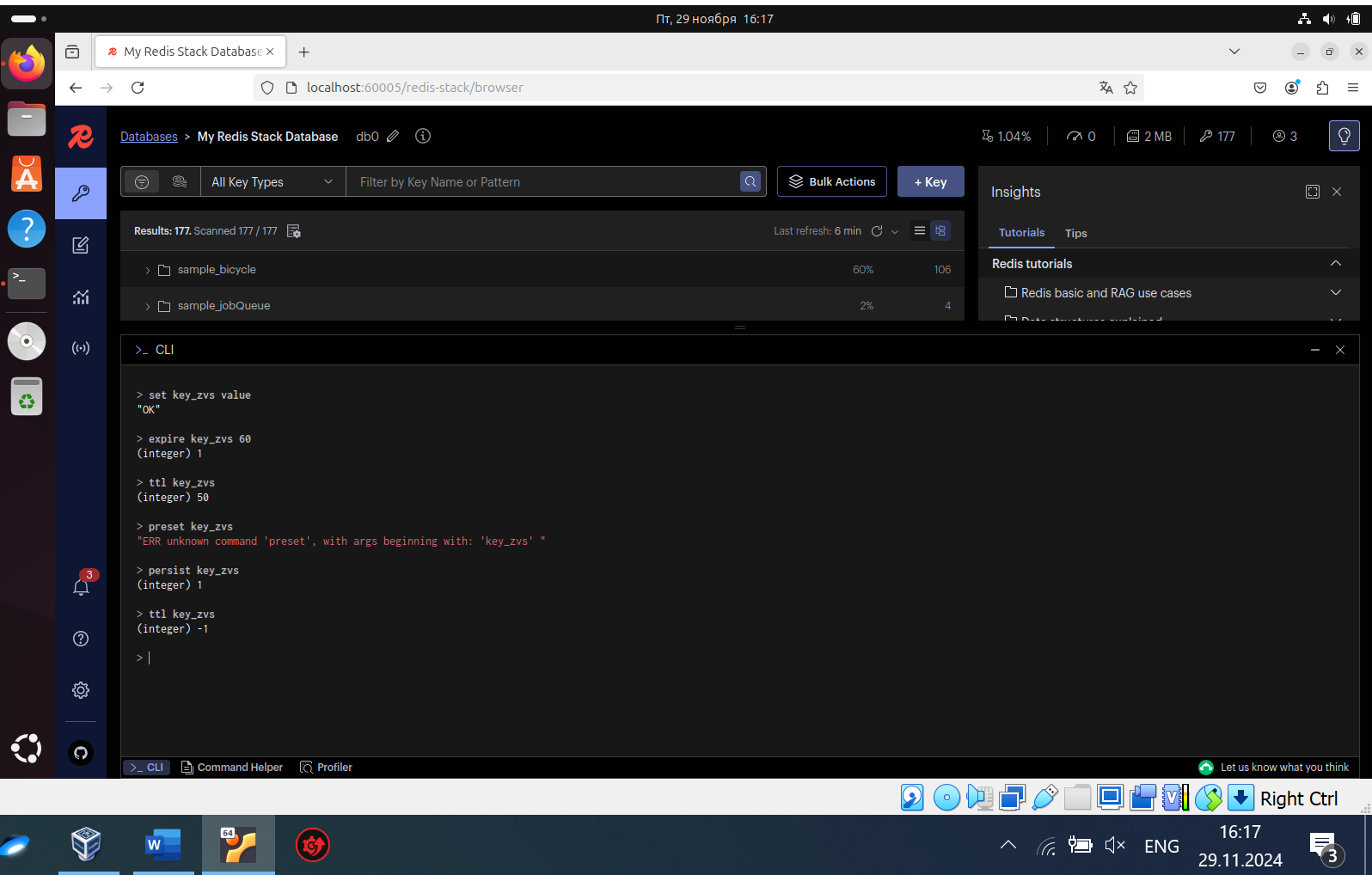
− exists

− del

− keys



Изучите возможность использования параметра времени жизни объекта (TTL). Изучите команды ttl и expire



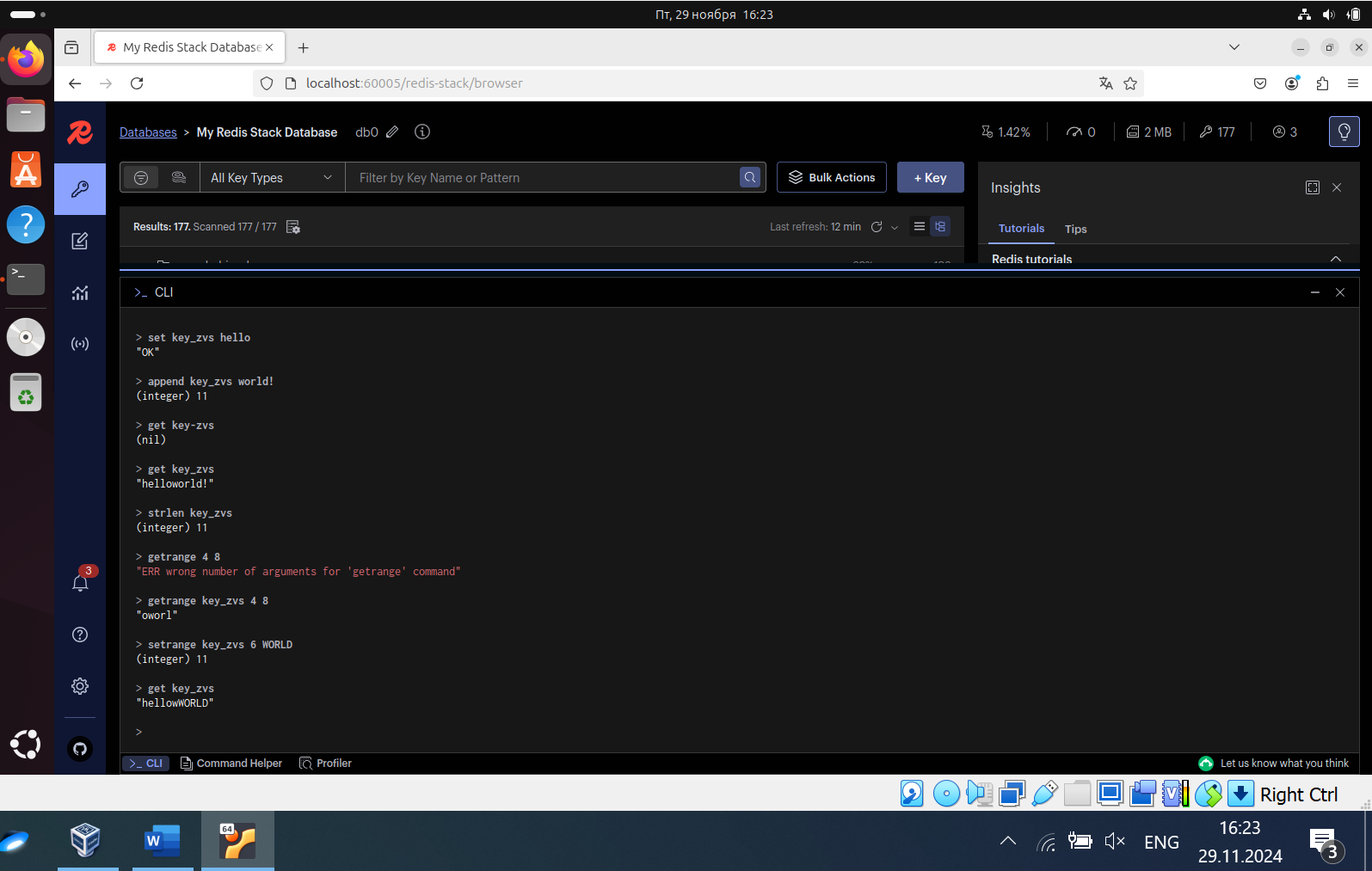
Изучите основные строковые операции:

− append

− strlen

− getrange

− setrange



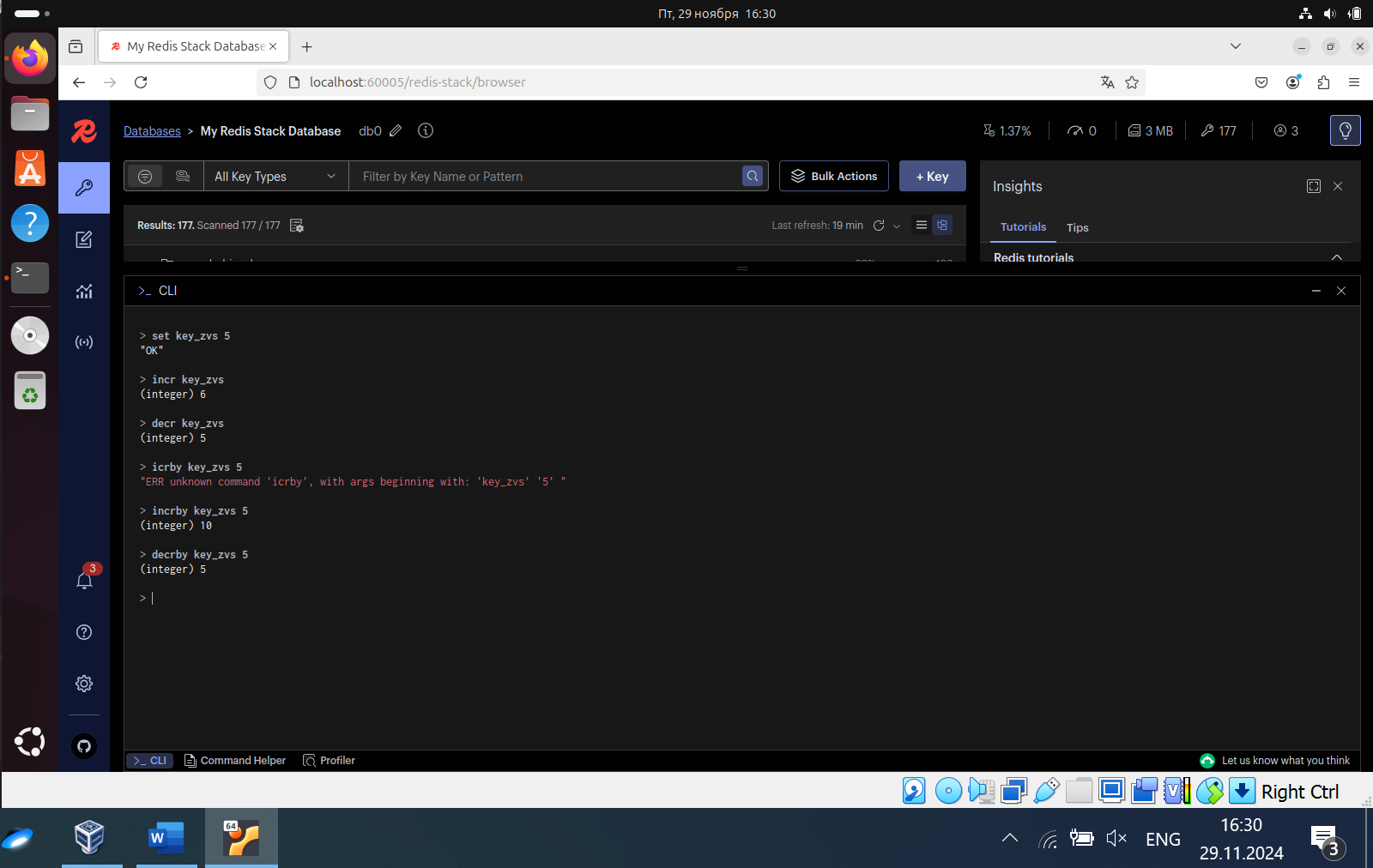
Изучите операции над числами:

− incr

− decr

− incrby

− decrby



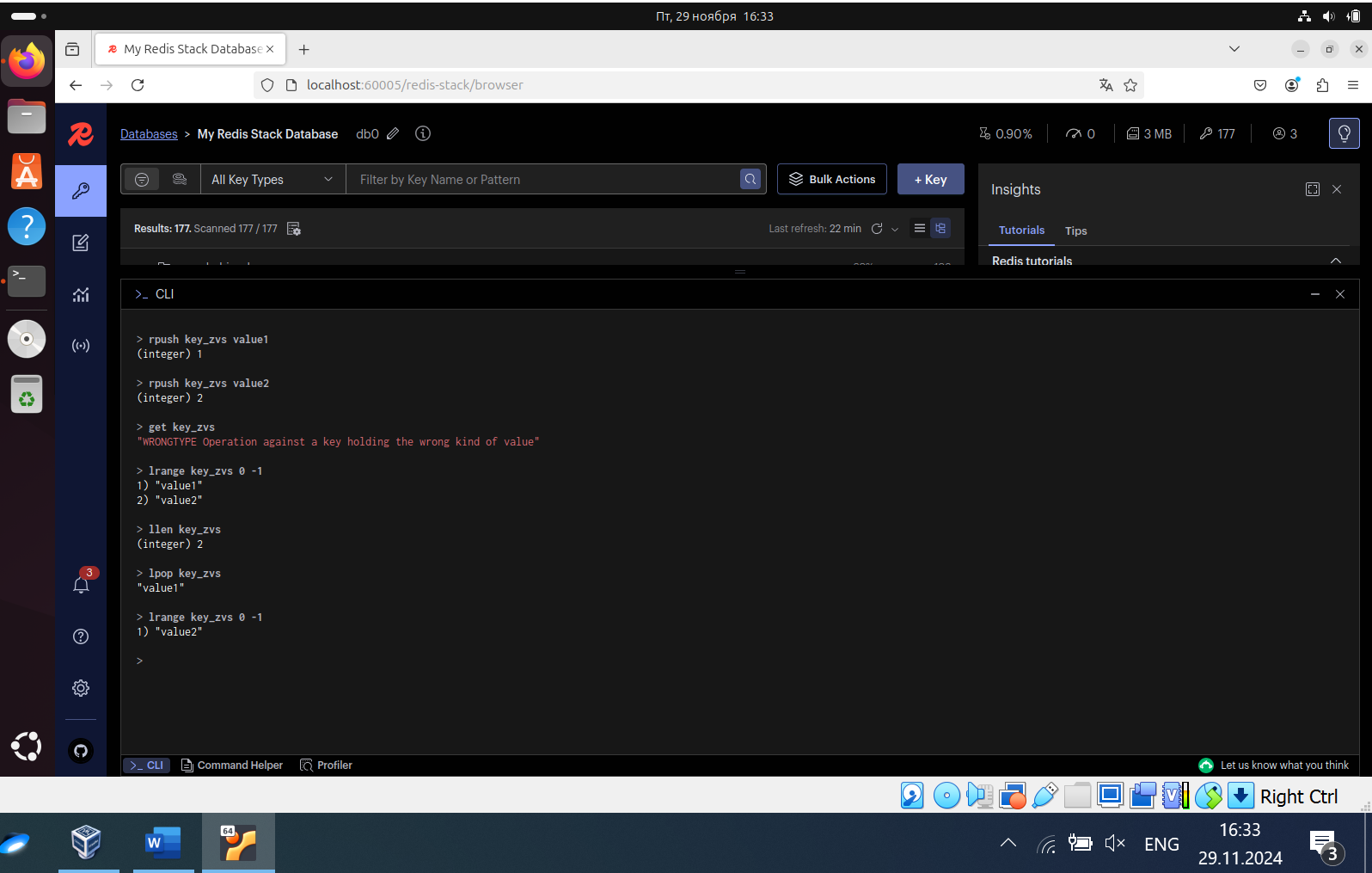
Изучите основные операции над списками:

− rpush

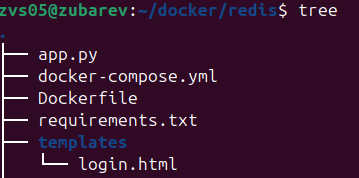
− lrange

− llen

− lpop



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.



Листинг app.py

from flask import Flask, render\_template, request

import redis

import time

app = Flask(\_\_name\_\_)

# Настройки для подключения к Redis

redis\_host = "redis\_zvs" # Имя контейнера Redis из docker-compose.yml

redis\_port = 6379

redis\_password = "adminzvs"

# Создаем подключение к Redis

r = redis.StrictRedis(host=redis\_host, port=redis\_port, password=redis\_password, decode\_responses=True)

# Данные для авторизации

VALID\_USERNAME = "zvs\_05"

VALID\_PASSWORD = "12345"

# Функция для получения количества неудачных попыток

def get\_failed\_attempts(username):

return int(r.get(f"failed\_attempts\_{username}") or 0)

# Функция для увеличения счетчика неудачных попыток

def increment\_failed\_attempts(username):

failed\_attempts = get\_failed\_attempts(username)

r.set(f"failed\_attempts\_{username}", int(failed\_attempts) + 1, ex=60) # Сохраняем на 60 секунд

# Функция для сброса счетчика неудачных попыток

def reset\_failed\_attempts(username):

r.delete(f"failed\_attempts\_{username}")

# Функция для регистрации удачной попытки

def register\_successful\_login(username):

timestamp = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())

r.rpush("successful\_logins", f"{timestamp}:{username}") # Добавляем запись в список

# Функция для проверки, заблокирован ли пользователь

def is\_blocked(username):

failed\_attempts = get\_failed\_attempts(username)

if failed\_attempts >= 3:

# Добавляем пользователя в список заблокированных

r.sadd("blocked\_users", username)

return True

return False

# Функция для проверки, заблокирован ли пользователь в Redis

def is\_user\_blocked(username):

return r.sismember("blocked\_users", username)

# Главная страница - форма авторизации

@app.route("/", methods=["GET", "POST"])

def login():

error = None

username = request.form.get("username")

password = request.form.get("password")

# Проверка на заблокированного пользователя

if username and is\_user\_blocked(username):

error = "Ваш аккаунт заблокирован. Попробуйте позже."

return render\_template("login.html", error=error)

if request.method == "POST":

# Проверка правильности логина и пароля

if username == VALID\_USERNAME and password == VALID\_PASSWORD:

# Успешный вход - сброс неудачных попыток и регистрация удачной попытки

reset\_failed\_attempts(username)

register\_successful\_login(username)

return "Успешный вход!"

else:

# Неверный логин или пароль

increment\_failed\_attempts(username)

error = "Неверный логин или пароль."

return render\_template("login.html", error=error)

@app.route("/blocked")

def blocked\_users():

blocked = r.smembers("blocked\_users")

return f"Заблокированные пользователи: {', '.join(blocked)}"

@app.route("/successful")

def successful\_logins():

logins = r.lrange("successful\_logins", 0, -1)

return f"Успешные попытки входа:<br>" + "<br>".join(logins)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host="0.0.0.0", port=5000, debug=True)

Листинг docker-compose.yml

version: '3.8'

networks:

redis:

driver: bridge

services:

redis-stack:

image: redis/redis-stack:latest

container\_name: redis\_zvs

networks:

- redis

ports:

- "6305:6379"

- "60005:8001"

environment:

REDIS\_ARGS: "--requirepass adminzvs"

restart: unless-stopped

redis-insight:

image: redislabs/redisinsight:latest

container\_name: insight\_zvs

networks:

- redis

ports:

- "6005:8001"

restart: unless-stopped

web:

build: .

container\_name: web\_auth

ports:

- "5000:5000"

environment:

- REDIS\_URL=redis://redis\_zvs:6379

- REDIS\_PASSWORD=adminzvs

depends\_on:

- redis-stack

restart: unless-stopped

networks:

- redis

Листинг Dockerfile

# Используем официальный Python образ

FROM python:3.9-slim

# Устанавливаем рабочую директорию

WORKDIR /app

# Копируем все файлы в контейнер

COPY . /app

# Устанавливаем зависимости

RUN pip install --no-cache-dir -r requirements.txt

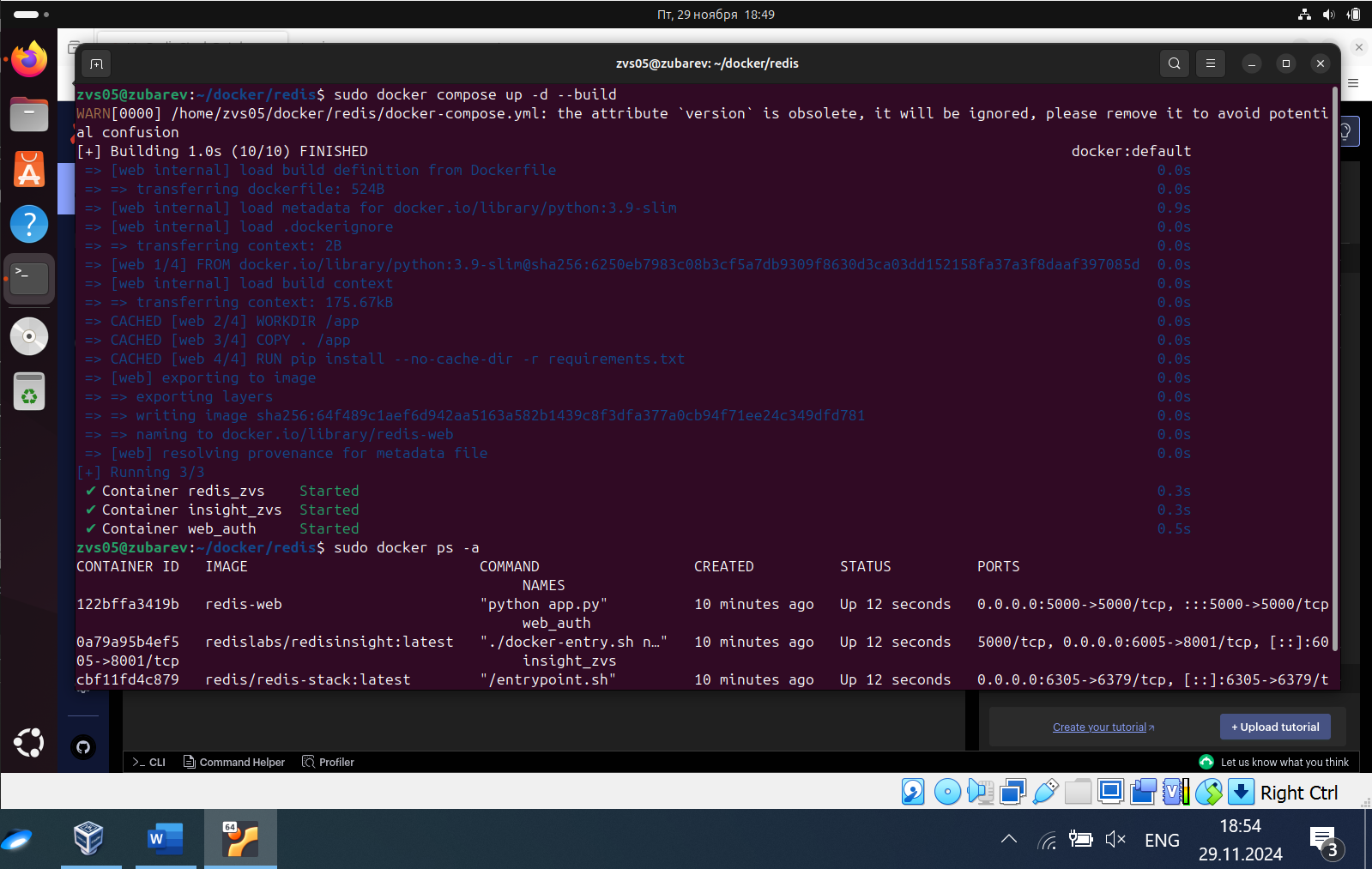
# Открываем порт для Flask-приложения

EXPOSE 5000

# Запускаем приложение

CMD ["python", "app.py"]

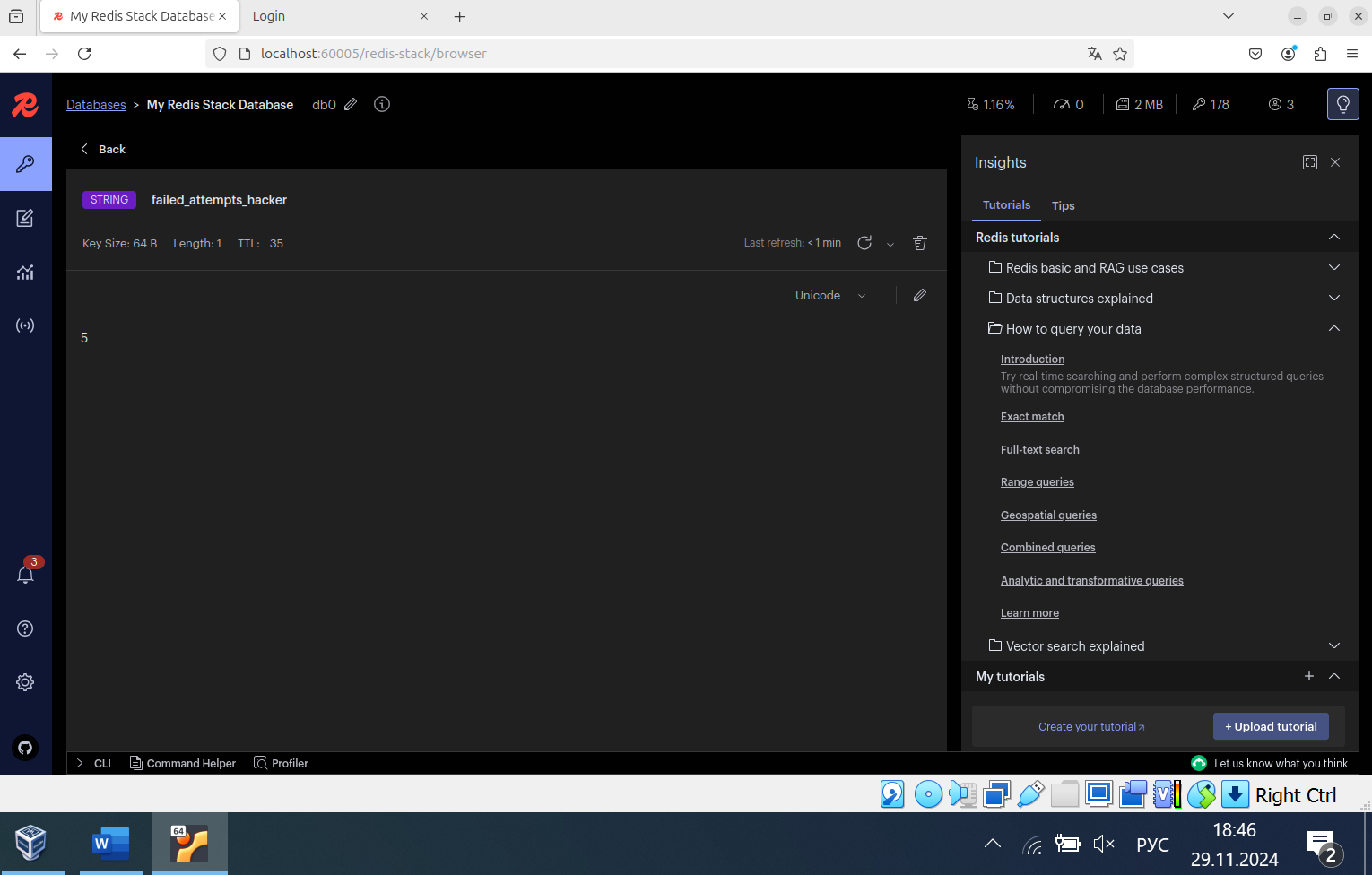
Работа контейнеров

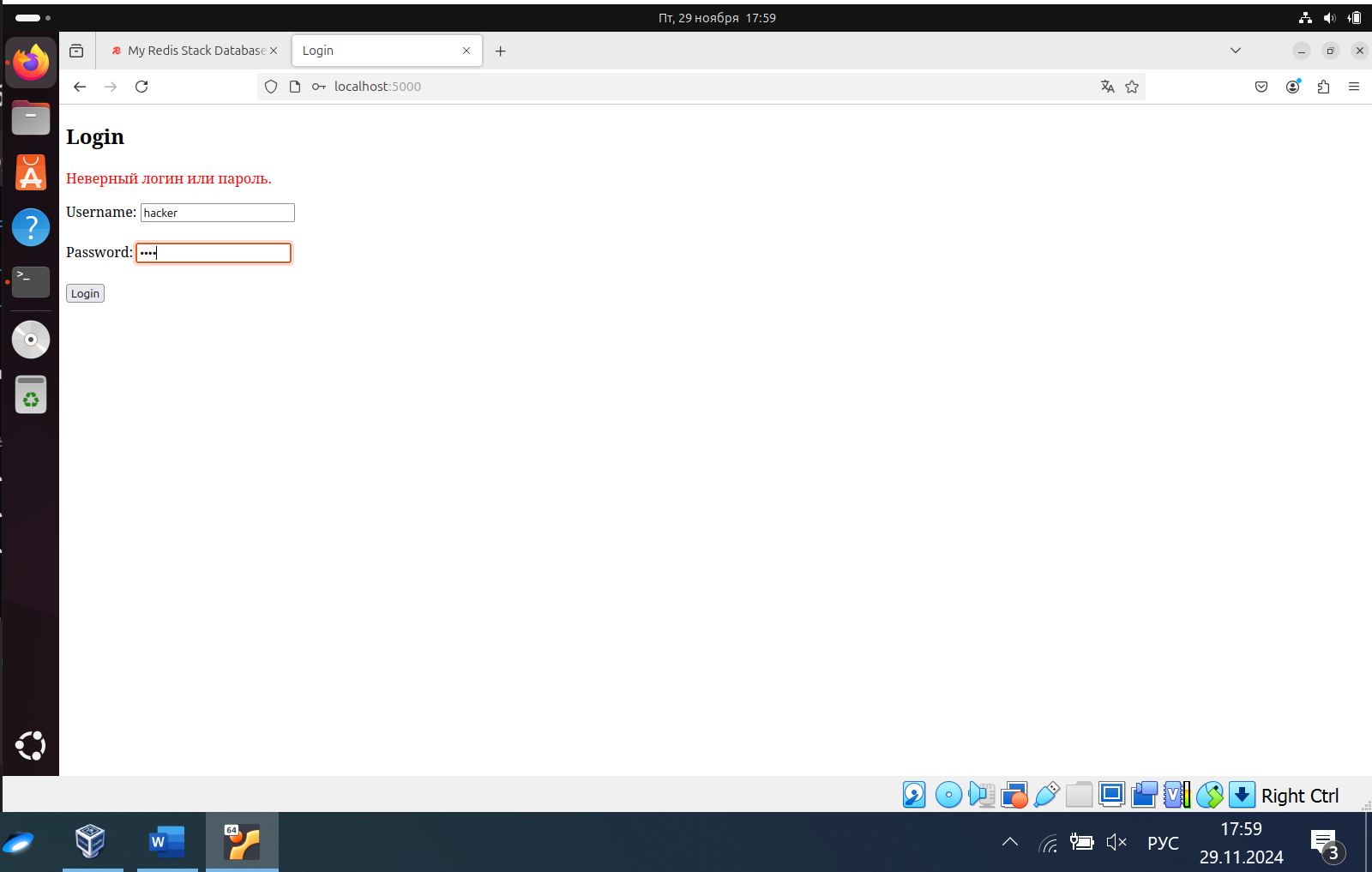


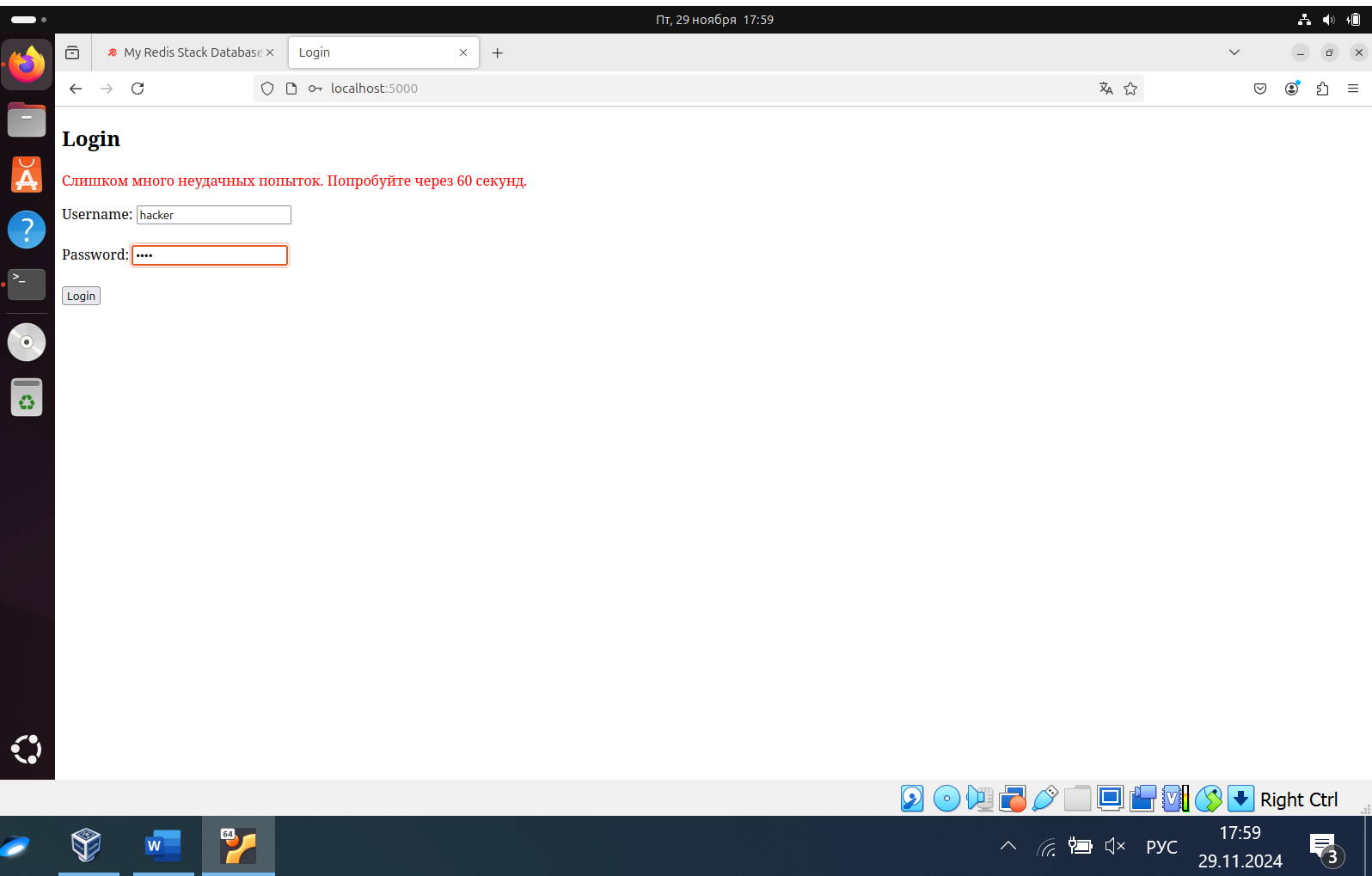
Протестируйте работу сервиса. Проанализируйте данные в Redis с помощью RedisInsight для трех случаев:

− неудачная попытка авторизации;

− превышено количество попыток авторизации;







− успешная авторизация.

