ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Ассистент		Д.А. Кочин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

НЕПРЕРЫВНАЯ ИНТЕГРАЦИЯ В ОБЛАКЕ С ИСПОЛЬЗОВАНИЕМ GITHUB ACTIONS

по курсу: ОПЕРАЦИОННЫЕ СИСТЕМЫ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4232		М.Д. Порохняк
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

1. Цель работы

Изучение принципов организации непрерывной интеграции приложений с использованием облачных технологий.

2. Задание

Добавить в репозиторий лабораторной работы №5 файл конфигурации .github/workflows/tests.yml, который должен автоматически запускать сборку файла, разработанного для ОС Linux (т.е. client.cpp, см Приложение А). Задачу (job) следует назвать test. Сборка должна осуществляться без ошибок.

При выполнении данной лабораторной работы требуется добавлять соответствующие задачи в раздел issues. Коммиты должны ссылаться на данные задачи.

3. Ход работы

В результате выполнения данной лабораторной был получен файл tests.yml со следующим содержимым (рис 3.1):

C: > Users > ad_ag > Documents > laba5 > os-task5-metaarsenit > .github > workflows > tests.yml

```yaml
name: test

on: [push, workflow_dispatch]

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Install required tools
        run: sudo apt-get update && sudo apt-get install -y build-essential

      - name: Compile file
        run: |
          mkdir build
          g++ -o build/client.out client.cpp

      - name: Saving compiled file
        uses: actions/upload-artifact@v3
        with:
          name: client-artifact
          path: build/
```

Рисунок 3.1 Файл tests.yml

Файл выполняет следующие шаги:

- устанавливает требуемые зависимости (Install required tools)

- создает каталог build (Compile file)

- в этом каталоге при помощи g++ компилирует файл client.cpp в файл client.out (Compile file)

- создает артефакт для скачивания (Saving compiled file)

Мы можем убедиться в успешном пройденном тесте, выполнив коммит и перейдя в раздел actions (рисунок 3.2)

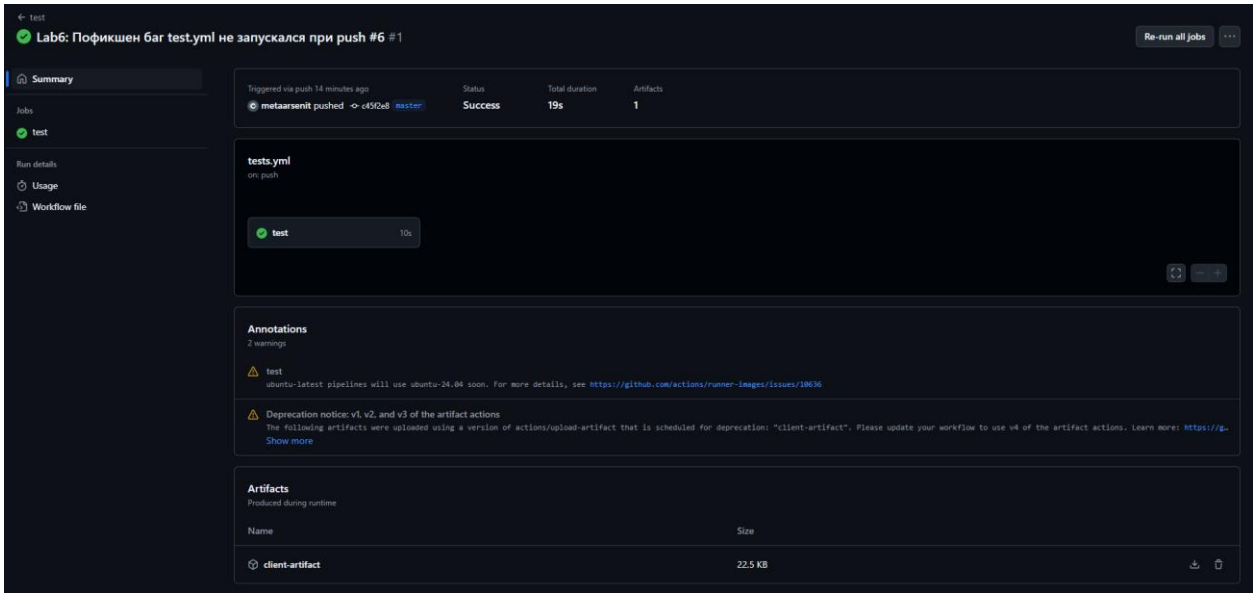

Рисунок 3.2 Успешно пройденный тест

Скачаем артефакт и дополнительно убедимся, что код скомпилирован успешно. Результат продемонстрирован на рисунке 3.3

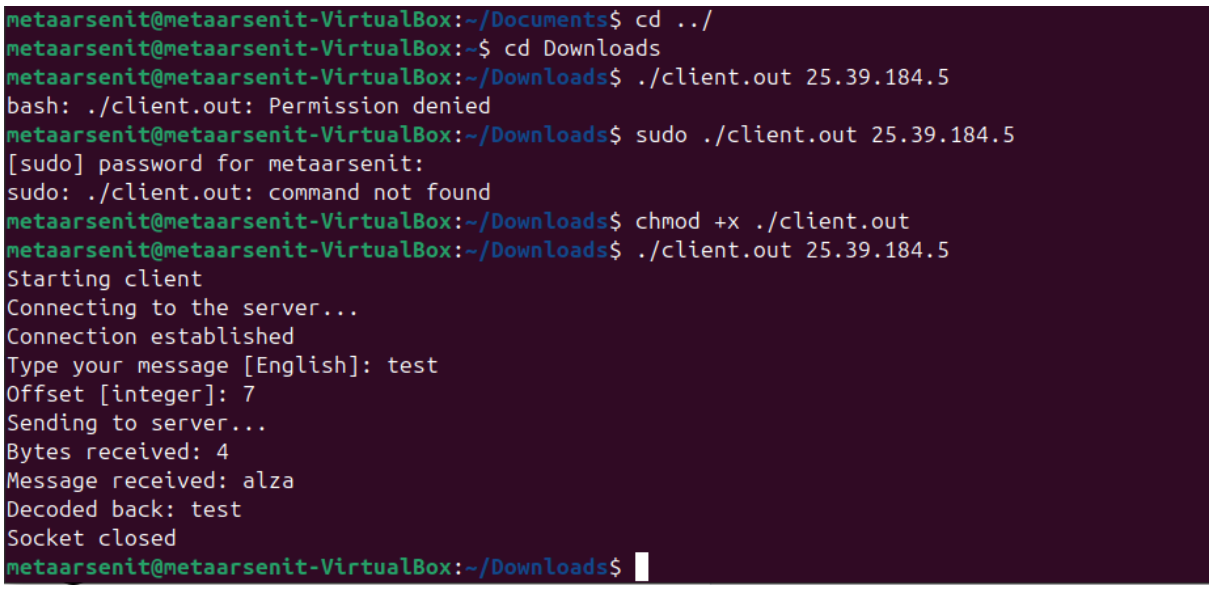

Рисунок 3.2 Использование артефакта

4. Выводы

В результате выполнения данной лабораторной работы были получены навыки и умения организации непрерывной интеграции приложений с использованием облачных технологий. Был создан файл test.yml, который автоматически запускает сборку файла client.cpp.

ПРИЛОЖЕНИЕ А

client.cpp

```
// Copyright 2025 metaarsenit
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<cstring>
#include<string>
#include<iostream>

char* decode(const char* word, int N, int length) {
    if (word == nullptr)
        return nullptr;

    char* answer = new char[length + 1];
    for (int i = 0; i < length; i++) {
        int value = static_cast<int>(word[i]);
        // 65-90 A-Z
        if (65 <= value && value <= 90) {
            answer[i] = (abs((value - 65 + N * 25) % 26)) + 65;
        } else if (97 <= value && value <= 122) {
            // 97-122 a-z
            answer[i] = (abs((value - 97 + N * 25) % 26)) + 97;
        } else {
            // special characters are unchanged
            answer[i] = static_cast<char>(value);
        }
    }
```

```
    answer[length] = '\0';
    return answer;
}



using std::cout, std::cin;
using std::string, std::to_string;

int main(int argc, char* argv[]) {
    cout << "Starting client\n";

    int clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    sockaddr_in serverAddress;
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(8080);
    serverAddress.sin_addr.s_addr = inet_addr(argv[1]);

    cout << "Connecting to the server...\n";
    connect(clientSocket,
            (struct sockaddr*)&serverAddress,
            sizeof(serverAddress));
    cout << "Connection established\n";

    cout << "Type your message [English]: ";
    string _message;
    cin >> _message;
    cout << "Offset [integer]: ";
    unsigned int _offset;
    cin >> _offset;
```

```
    const char* offset = to_string(_offset).c_str();
    cout << "Sending to server...\n";
    const char* message = _message.c_str();
    send(clientSocket, message, strlen(message), 0);
    send(clientSocket, offset, strlen(offset), 0);

    int buflen = 1024;
    char recvbuf[1024];

    int iResult = recv(clientSocket, recvbuf, buflen, 0);
    if (iResult > 0) {
        cout << "Bytes received: " << iResult << "\n";
        recvbuf[iResult] = '\0';
        cout << "Message received: " << recvbuf  << "\n";
        cout << "Decoded back: ";
        cout << decode(recvbuf, _offset, strlen(recvbuf)) << "\n";
    }
    close(clientSocket);
    cout << "Socket closed\n";

    return 0;
}
```