



Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

Комп'ютерний практикум №4

Автоматизація роботи з ресурсами AWS засобами мови Python

предмет «Хмарні технології обробки даних»

Роботу виконав:

Студент 3 курсу ФТІ, групи ФІ-91
Цибульник Антон Владиславович

Приймали:

Шелестов Андрій Юрійович
Колотій Андрій Всеволодович

Мета

Ознайомитись з основами керування ресурсами AWS засобами Python SDK.

Завдання

- Розробити Python-скрипт для автоматичного створення та видалення хмарної інфраструктури з мінімальною конфігурацією (необхідно передбачити у функціях додаткові виключення для коректної роботи у нестандартних ситуаціях (наприклад, виводити відповідне повідомлення при створення бакету з вже існуючим ім'ям, при читанні з S3 файлу, якого там немає);
- Для результатів Лабораторної роботи №2 розробити bash-скрипт, який скопіює код з попередньо створеного git-репозиторію та встановить потрібні залежності `pip`.

Хід виконання роботи

1. Автоматизація роботи з обчислювальними ресурсами EC

Підготовчий етап

Повторимо усі кроки Лабораторної роботи №1 стосовно встановлення інстансу й роботи з ним, проте тепер не через інтерфейс AWS, а засобами Python SDK. Перш за все маємо виконати деякі підготовчі кроки на власному локальному середовищі (тобто на ноутбукі чи комп'ютері):

1. **Встановити мову Python3:** для операційної системи Ubuntu 20.04 (попередньо оновивши дерево пакетів) це можна зробити за допомогою команди

```
sudo apt update  
sudo apt install python3
```

Для керування програмними пакетами мови Python3 наступним кроком слід встановити інструмент `pip`:

```
sudo apt install python3-pip
```

2. **Встановити бібліотеку Boto3:** для цього виконуємо одну просту команду

```
pip3 install boto3
```

Результат пункту зображений на Рис. 1.

3. **Мати обліковий запис на AWS:** акаунт на AWS був створений ще у Лабораторній роботі №1. Зараз постає завдання підключитися до цього акаунту з локального середовища. Скористаємося інструментами AWS Command Line Interface (AWS CLI). Встановимо власне засоби AWS CLI командою

```
sudo apt install awscli
```

Одразу хочу зауважити, що при некоректній роботі щойно встановлених інструментів (та і просто для профілактики, як показав досвід другої лабораторної роботи, здійснимо превентивне виправлення несправностей, так би мовити) виконаємо рядок

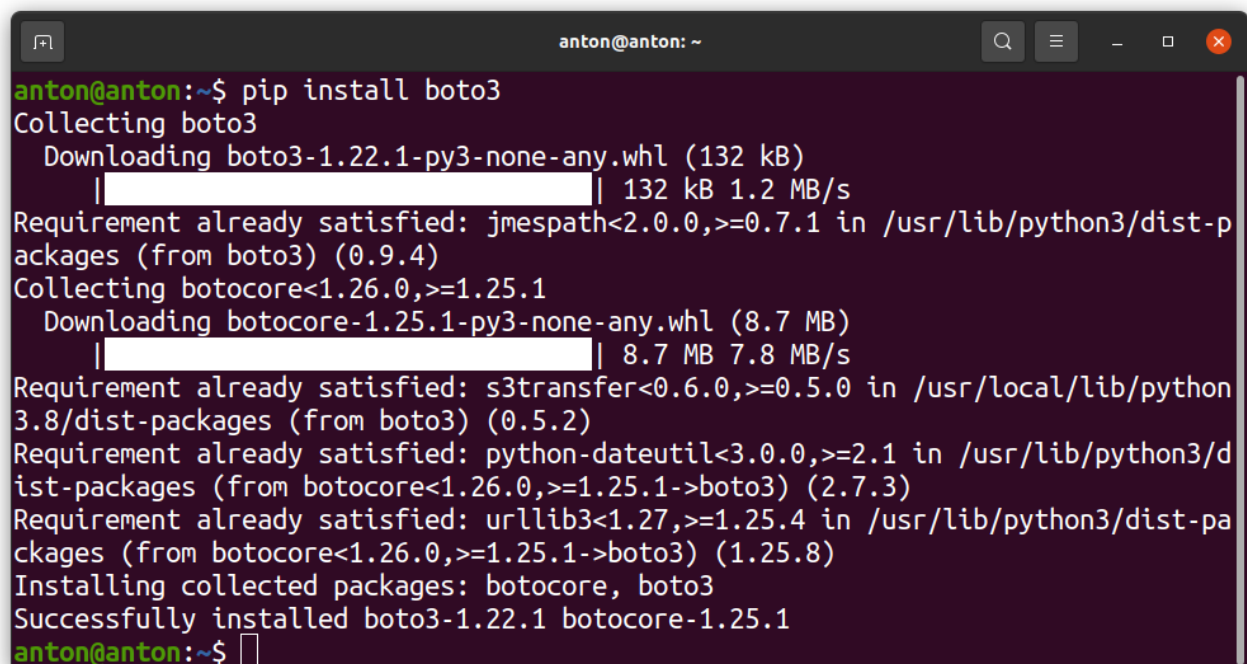
```
pip3 install --upgrade awscli
```

Усі проміжні результати на цей момент показані на Рис. 2.

Ну і наостанок власне здійснимо саме підключення до користувача AWS. Як це було виконано у Лабораторній роботі №2, виконуємо команду

```
aws configure
```

й коректно заповнюємо усі параметри, скориставшись [інструкцією](#). На Рис. 3 можемо бачити успішне підключення до AWS.



```
anton@anton: ~  
anton@anton:~$ pip install boto3  
Collecting boto3  
  Downloading boto3-1.22.1-py3-none-any.whl (132 kB)  
    | 132 kB 1.2 MB/s  
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3/dist-p  
ackages (from boto3) (0.9.4)  
Collecting botocore<1.26.0,>=1.25.1  
  Downloading botocore-1.25.1-py3-none-any.whl (8.7 MB)  
    | 8.7 MB 7.8 MB/s  
Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in /usr/local/lib/python  
3.8/dist-packages (from boto3) (0.5.2)  
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/d  
ist-packages (from botocore<1.26.0,>=1.25.1->boto3) (2.7.3)  
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3/dist-pa  
ckages (from botocore<1.26.0,>=1.25.1->boto3) (1.25.8)  
Installing collected packages: botocore, boto3  
Successfully installed boto3-1.22.1 botocore-1.25.1  
anton@anton:~$
```

Рис. 1: Завантаження бібліотеки boto3

```
anton@anton: ~  
anton@anton:~$ sudo apt install awscli  
[sudo] password for anton:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
awscli is already the newest version (1.18.69-1ubuntu0.20.04.1).  
The following packages were automatically installed and are no longer required:  
  libfwupdplugin1 libllvm13 libllvm13:i386  
Use 'sudo apt autoremove' to remove them.  
0 upgraded, 0 newly installed, 0 to remove and 37 not upgraded.  
anton@anton:~$
```

```
anton@anton: ~  
anton@anton:~$ pip3 install --upgrade awscli  
Collecting awscli  
  Downloading awscli-1.23.1-py3-none-any.whl (3.8 MB)  
    | 3.8 MB 1.3 MB/s  
Requirement already satisfied, skipping upgrade: rsa<4.8,>=3.1.2 in /usr/lib/python3/dist-packages (from awscli) (4.0)  
Requirement already satisfied, skipping upgrade: colorama<0.4.5,>=0.2.5 in /usr/lib/python3/dist-packages (from awscli) (0.4.3)  
Requirement already satisfied, skipping upgrade: PyYAML<5.5,>=3.10 in /usr/lib/python3/dist-packages (from awscli) (5.3.1)  
Collecting docutils<0.16,>=0.10  
  Downloading docutils-0.15.2-py3-none-any.whl (547 kB)  
    | 547 kB 11.8 MB/s  
Requirement already satisfied, skipping upgrade: s3transfer<0.6.0,>=0.5.0 in /usr/local/lib/python3.8/dist-packages (from awscli) (0.5.2)  
Requirement already satisfied, skipping upgrade: botocore==1.25.1 in ./local/lib/python3.8/site-packages (from awscli) (1.25.1)  
Requirement already satisfied, skipping upgrade: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/dist-packages (from botocore==1.25.1->awscli) (2.7.3)  
Requirement already satisfied, skipping upgrade: urllib3<1.27,>=1.25.4 in /usr/lib/python3/dist-packages (from botocore==1.25.1->awscli) (1.25.8)  
Requirement already satisfied, skipping upgrade: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3/dist-packages (from botocore==1.25.1->awscli) (0.9.4)  
Installing collected packages: docutils, awscli  
Successfully installed awscli-1.23.1 docutils-0.15.2  
anton@anton:~$
```

Рис. 2: Встановлення інструментів AWS CLI

```
anton@anton: ~  
anton@anton:~$ aws configure  
AWS Access Key ID [None]: AKIAZ3CGCDYGHZFSTNX5  
AWS Secret Access Key [None]: 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  
Default region name [None]: us-east-1  
Default output format [None]: json  
anton@anton:~$
```

Рис. 3: Підключення до акаунту AWS

Створення пари ключів для доступу до EC2-інстансу

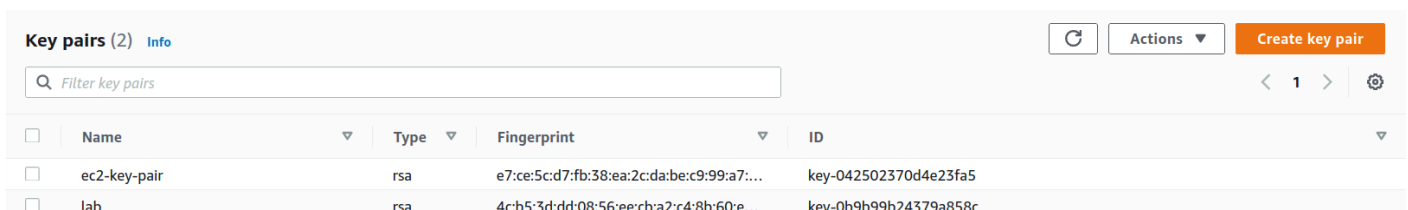
Виконавши усі підготовчі кроки, створюємо файл відповідного розширення (наприклад, `lab4.py`). Надалі будемо поповнювати цей файл кодом й запускатити його в терміналі командою

```
python3 lab4.py
```

Отже, напишемо функцію створення пари ключів (Лістинг 1). Рядком 6 задамо параметри інстансу та регіону, рядком 7 – ім'я для ключа, ну і 9 рядком запишемо назву файлу розширення `.pem`. Фактично, однією функцією одразу виконали усі кроки, які ми виконували для відповідного завдання у першій лабораторній роботі. Останнім рядком викликаємо функцію `create_key_pair()`. Запустивши файл на виконання, отримуємо бажаний результат (Рис. 4).

Лістинг 1: Створення пари ключів

```
1 import boto3
2 import botocore
3 import os
4
5 def create_key_pair():
6     ec2_client = boto3.client("ec2", region_name="us-east-1")
7     key_pair = ec2_client.create_key_pair(KeyName="ec2-key-pair2")
8     private_key = key_pair["KeyMaterial"]
9     with os.fdopen(os.open("aws_lab4.pem", os.O_WRONLY |
10        os.O_CREAT, 0o400), "w+") as handle:
11         handle.write(private_key)
12
13 create_key_pair()
```



Key pairs (2) Info			
Filter key pairs			
<input type="checkbox"/>	Name	Type	Fingerprint
<input type="checkbox"/>	ec2-key-pair	rsa	e7:ce:5c:d7:fb:38:ea:2c:da:be:c9:99:a7:...
<input type="checkbox"/>	lab	rsa	4c:b5:3d:dd:08:56:ee:cb:a2:c4:8b:60:e:...

Рис. 4: Успішно створена пара ключів

Створення EC2-інстансу

Наступним кроком створимо власне сам інстанс (Лістинг 2). Напишемо відповідну функцію `create_instance()` й вкажемо усі необхідні параметри, серед яких – ім'я щойно записаного ключа. Стосовно параметра «ImageId», то на Рис. 5 показано, що його можна знайти на першому кроці при стробі створити інстанс через інтерфейс AWS (сам параметр виділено сірим кольором). Рядком 25 забезпечуємо вивід «InstanceId» після виклику функції.

Запускаємо файл на виконання й переконуємося, що новий інстанс є у переліку усіх інстансів (Рис. 6).

Лістинг 2: Створення інстансу

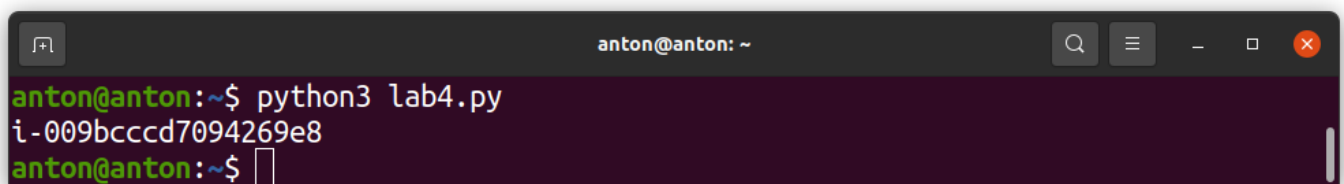
```

15 def create_instance():
16     ec2_client = boto3.client("ec2", region_name="us-east-1")
17     instances = ec2_client.run_instances(
18         ImageId="ami-08895422b5f3aa64a",
19         MinCount=1,
20         MaxCount=1,
21         InstanceType="t2.micro",
22         KeyName="ec2-key-pair2"
23     )
24
25     print(instances["Instances"][0]["InstanceId"])
26
27 create_instance()

```

Step 1: Choose an Amazon Machine Image (AMI)

Рис. 5: Параметр «ImageId»



Instances (2) Info

Refresh

Connect

Instance state ▾

Actions ▾

Launch instances

▾

Q Search

< 1 >

⚙

<input type="checkbox"/>	Name ▾	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability Zone ▾	Public IPv4 DNS ▾
<input type="checkbox"/>	demo	i-009bcccd7094269e8	<div>✔ Running</div> <div>🔍🔍</div>	t2.micro	–	No alarms <div>+</div>	us-east-1a	ec2-3-239-208-221.co...
<input type="checkbox"/>	first	i-03c0d5c1d9a8fb1ad	<div>⏸ Stopped</div> <div>🔍🔍</div>	t2.micro	–	No alarms <div>+</div>	us-east-1b	–

Рис. 6: Створення інстансу

Опісля можемо виконати один додатковий крок – перевірити, які інстанси зараз запущені на нашому акаунті AWS. Сконструємо доволі громіздку функцію `get_running_instances()`, де у рядках 30-39 фільтром вкажемо тип інстансів, які ми хочемо бачити (конкретно 34 рядок – хочемо виводити саме запущені інстанси). Змінивши параметри фільтру, можна вивидити, наприклад, і зупинені інстанси.

А рядками 41-47 для більшої інформативності напишемо код виведення параметрів «instance_id», «instance_type», «public_ip» та «private_ip». Результати на Рис. 7 збігаються із результатами на Рис. 6 (ба більше, ще й маємо публічні та приватні IP адреси).

Лістинг 3: Моніторинг активних сеансів

```
29 def get_running_instances():
30     ec2_client = boto3.client("ec2", region_name="us-east-1")
31     reservations = ec2_client.describe_instances(Filters=[
32         {
33             "Name": "instance-state-name",
34             "Values": ["running"],
35         },
36         {
37             "Name": "instance-type",
38             "Values": ["t2.micro"]
39         }
40     ]).get("Reservations")
41
42     for reservation in reservations:
43         for instance in reservation["Instances"]:
44             instance_id = instance["InstanceId"]
45             instance_type = instance["InstanceType"]
46             public_ip = instance["PublicIpAddress"]
47             private_ip = instance["PrivateIpAddress"]
48             print(f"{instance_id}, {instance_type}, {public_ip}, {private_ip}")
49
50 get_running_instances()
```

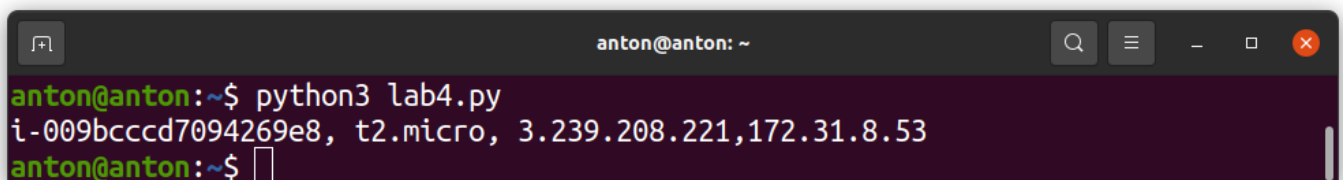


Рис. 7: Перелік активних сеансів

Зупинка й видалення інстансу

На Лістингу 4 та 5 прописаний код зупинки й видалення інстансу. Ключовим параметром є ID інстансу, який ми отримали ще на етапі Рис. 6. Як результат – відповідні позначки «Instance state» для інстансу «demo» (Рис. 8 та Рис. 9).

Лістинг 4: Зауника активного інстансу

```
51 def stop_instance(instance_id):
52     ec2_client = boto3.client("ec2", region_name="us-east-1")
53     response = ec2_client.stop_instances(InstanceIds=[instance_id])
54     print(response)
55
56 stop_instance("i-009bcccd7094269e8")
```



```
anton@anton: ~$ python3 lab4.py
{'StoppingInstances': [{'CurrentState': {'Code': 64, 'Name': 'stopping'}, 'InstanceID': 'i-009bcccd7094269e8', 'PreviousState': {'Code': 16, 'Name': 'running'}}], 'ResponseMetadata': {'RequestId': '948f0ee9-8803-4e4d-a063-a991e28a5d3e', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': '948f0ee9-8803-4e4d-a063-a991e28a5d3e', 'cache-control': 'no-cache, no-store', 'strict-transport-security': 'max-age=31536000; includeSubDomains', 'content-type': 'text/xml; charset=UTF-8', 'content-length': '579', 'date': 'Wed, 27 Apr 2022 14:41:56 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}
```

Instances (2) Info

Q Search

Refresh

Connect

Instance state ▾

Actions ▾

Launch instances

1

⚙

<input type="checkbox"/>	Name ▾	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability Zone ▾	Public IPv4 DNS ▾
<input type="checkbox"/>	demo	i-009bcccd7094269e8	⊖ Stopped 🔍	t2.micro	–	No alarms +	us-east-1a	–
<input type="checkbox"/>	first	i-03c0d5c1d9a8fb1ad	⊖ Stopped 🔍	t2.micro	–	No alarms +	us-east-1b	–

Рис. 8: Зупинений інстанс

Лістинг 5: Видалення зупиненого інстансу

```
58 def terminate_instance(instance_id):
59     ec2_client = boto3.client("ec2", region_name="us-east-1")
60     response = ec2_client.terminate_instances(InstanceIds=[instance_id])
61     print(response)
62
63 terminate_instance("i-009bcccd7094269e8")
```

```
anton@anton: ~$ python3 lab4.py
{'TerminatingInstances': [{'CurrentState': {'Code': 48, 'Name': 'terminated'}, 'InstanceID': 'i-009bcccd7094269e8', 'PreviousState': {'Code': 80, 'Name': 'stopped'}}], 'ResponseMetadata': {'RequestId': 'ccb68f32-104b-4ec9-8868-af9d5e7f8dbc', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'ccb68f32-104b-4ec9-8868-af9d5e7f8dbc', 'cache-control': 'no-cache, no-store', 'strict-transport-security': 'max-age=31536000; includeSubDomains', 'vary': 'accept-encoding', 'content-type': 'text/xml; charset=UTF-8', 'transfer-encoding': 'chunked', 'date': 'Wed, 27 Apr 2022 15:09:37 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}
```

Instances (2) Info

🔄

Connect

Instance state ▾

Actions ▾

Launch instances

▾

🔍 Search

< 1 >

⚙️

<input type="checkbox"/>	Name ▾	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability Zone ▾	Public IPv4 DNS ▾
<input type="checkbox"/>	demo	i-009bcccd7094269e8	⊖ Terminated 🔍🔍	t2.micro	–	No alarms +	us-east-1a	–
<input type="checkbox"/>	first	i-03c0d5c1d9a8fb1ad	⊖ Stopped 🔍🔍	t2.micro	–	No alarms +	us-east-1b	–

Рис. 9: Видалений інстанс

2. Автоматизація роботи з сховищем S3

Створення бакета S3

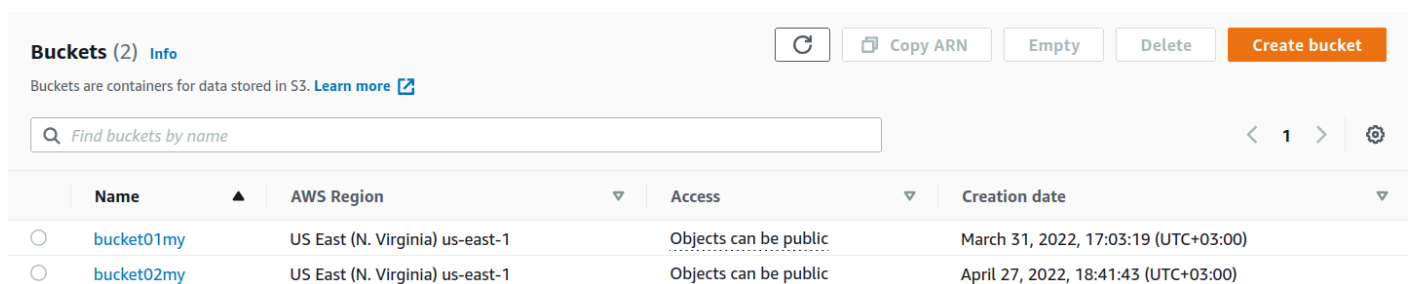
Для створення бакету реалізуємо функцію `create_bucket()`, якій передамо параметри імені нового бакету та регіону. Аналогічно до об'єкту `ec2_client` у попередньому розділі, на цей раз всі необхідні методи для керування бакетом можемо отримати через об'єкт `s3_client`.

Лістинг 6: Створення бакету S3

```
65 def create_bucket(bucket_name, region):
66     s3_client = boto3.client("s3", region_name=region)
67
68     try:
69         response = s3_client.create_bucket(Bucket=bucket_name)
70         print(response)
71     except botocore.exceptions.ClientError:
72         print("Wrong name!")
73     except s3_client.exceptions.BucketAlreadyExists:
74         print("Such bucket already exists!")
75
76 create_bucket("bucket02my", "us-east-1")
```



```
anton@anton: ~$ python3 lab4.py
{'TerminatingInstances': [{'CurrentState': {'Code': 48, 'Name': 'terminated'}, 'InstanceId': 'i-009bcccd7094269e8', 'PreviousState': {'Code': 48, 'Name': 'terminated'}}], 'ResponseMetadata': {'RequestId': 'aa9ace88-ed34-40cd-85e1-0b2e91f7a647', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'aa9ace88-ed34-40cd-85e1-0b2e91f7a647', 'cache-control': 'no-cache, no-store', 'strict-transport-security': 'max-age=31536000; includeSubDomains', 'vary': 'accept-encoding', 'content-type': 'text/xml; charset=UTF-8', 'transfer-encoding': 'chunked', 'date': 'Wed, 27 Apr 2022 15:41:40 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}
{'ResponseMetadata': {'RequestId': 'XHH0AAHG9MP02A10', 'HostId': 'SVEr7akkezlSvl6SDXsRwQRwN4HgR3eA0H9SAUr7ctDZac22MsS+yYEVPGsbz8RkoBZA9wRG08E=', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amz-id-2': 'SVEr7akkezlSvl6SDXsRwQRwN4HgR3eA0H9SAUr7ctDZac22MsS+yYEVPGsbz8RkoBZA9wRG08E=', 'x-amz-request-id': 'XHH0AAHG9MP02A10', 'date': 'Wed, 27 Apr 2022 15:41:43 GMT', 'location': '/bucket02my', 'server': 'AmazonS3', 'content-length': '0'}, 'RetryAttempts': 0}, 'Location': '/bucket02my'}
anton@anton: ~$
```



Buckets (2) Info					Refresh	Copy ARN	Empty	Delete	Create bucket
Buckets are containers for data stored in S3. Learn more									
<input type="text" value="Find buckets by name"/>									
	Name	AWS Region	Access	Creation date					
<input type="radio"/>	bucket01my	US East (N. Virginia) us-east-1	Objects can be public	March 31, 2022, 17:03:19 (UTC+03:00)					
<input type="radio"/>	bucket02my	US East (N. Virginia) us-east-1	Objects can be public	April 27, 2022, 18:41:43 (UTC+03:00)					

Рис. 10: Створений бакет


У Лістингу 6 конструкція `try: except` аналогічна конструкції `try: catch` в інших мовах програмування й покликана «зловити» виняткові ситуації помилок. Наприклад, у рядку 71 «ловимо» помилку неприйнятної довжини для імені, а у рядку 73 – проблему неунікального імені бакету.

Між іншим, назва помилки `botocore.exceptions.ClientError` була витягнута безпосередньо з консолі терміналу. Отже, після уникнення помилок хибних імен бакету, маємо остаточний результат, який показано на Рис. 10.

Аналогічно до етапу створення інстансу, останнім кроком цього пункту можемо написати код моніторингу активних бакетів (Лістинг 7). Як результат – маємо бажаний перелік (Рис. 11).

Лістинг 7: Активні бакети

```
78 def printing_buckets():
79     s3_client = boto3.client("s3")
80     response = s3_client.list_buckets()
81
82     print("Existing buckets:")
83     for bucket in response["Buckets"]:
84         print(f" {bucket['Name']}")
85
86 printing_buckets()
```



```
anton@anton:~$ python3 lab4.py
Existing buckets:
bucket01my
bucket02my
anton@anton:~$
```

Рис. 11: Перелік існуючих бакетів

Завантаження й читання даних

Завантажимо на новостворений бакет файл з курсом валют, який ми використувували у Лабораторній роботі №2. До речі, необхідний нам файл `request.csv` досі лежить на попередньому бакеті (Рис. 11), створеному у тій же другій лабораторній. Тож зайшовши на бакет `bucket01my`, завантажую на локальне сховище шукані дані. Отже, передаю у функцію `upload()` назву оригінального файлу та ім'я, яке він матиме на бакеті `bucket02my`.

Лістинг 8: Завантаження файлу на бакет

```
88 def upload(file_name, bucket_name, s3_obj_name):
89     s3_client = boto3.client("s3")
90     response = s3_client.upload_file(Filename=file_name, Bucket=bucket_name,
91                                     Key=s3_obj_name)
92     print(response)
93
94 upload("request.csv", "bucket02my", "data.csv")
```

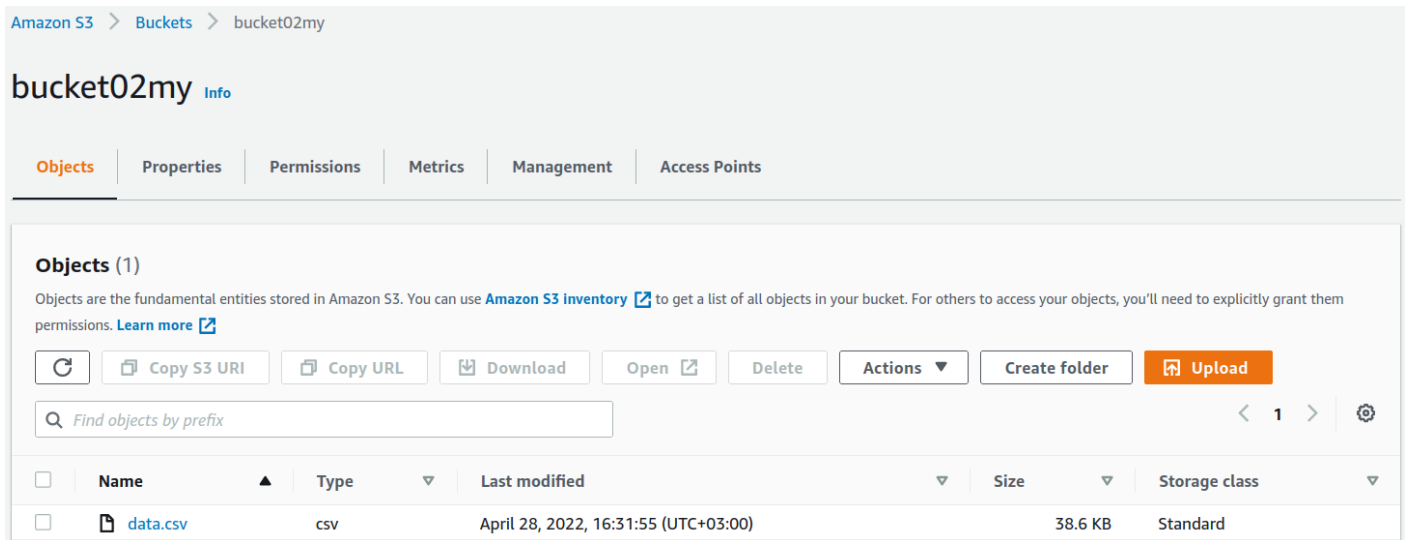


Рис. 12: Вивантажений на бакет файл

Як бачимо, завантаження пройшло успішно (Рис. 12). Тепер зчитуємо із цього файлу інформацію про перші рядки таблиці. Для візуалізації даних окремо встановимо відповідну бібліотеку `pandas` командою

```
pip3 install pandas
```

Наразі приймемося до написання функції `printing_data_from_s3()`. Основним є 101 рядок (Лістинг 9). А рядками 105 та 107 «ловимо» й інформуємо користувача про помилки звернення до бакету чи файлу з неіснуючим іменем.

Лістинг 9: Зчитування даних з бакету

```

95 import pandas
96
97 def printing_data_from_s3(file_name, bucket_name):
98     s3_client = boto3.client("s3")
99
100     try:
101         obj = s3_client.get_object(Bucket = bucket_name, Key = file_name)
102         data = pandas.read_csv(obj["Body"])
103         print("Printing the data frame...")
104         print(data.head())
105     except s3_client.exceptions.NoSuchBucket:
106         print("Проблемка. з. іменем. бакету")
107     except s3_client.exceptions.NoSuchKey:
108         print("Проблемка. з. іменем. файлу")
109
110 printing_data_from_s3("data.csv", "bucket02my")

```

У рядках 106 та 108 текст помилок не виділений відповідним кольором виключно через технічні складнощі середі \LaTeX у відображенні в блоках коду символів нелатинського алфавіту. Результати помилкових запитів до бакету й власне успішне читання даних зображені на Рис. 13.

```
anton@anton: ~$ python3 lab4.py
Проблемка з іменем файлу
anton@anton: ~$ python3 lab4.py
Проблемка з іменем бакету
anton@anton: ~$ python3 lab4.py
Printing the data frame...
   r030      txt      rate  cc  exchangedate
0    36  Австралійський долар  21.8659  AUD   10.01.2021
1   124  Канадський долар  22.2610  CAD   10.01.2021
2   156   Юань Женьмінбї  4.3944  CNY   10.01.2021
3   191      Куна  4.6098  HRK   10.01.2021
4   203  Чеська крона  1.3286  CZK   10.01.2021
anton@anton: ~$
```

Рис. 13: Читання з файлу

Видалення бакету S3

Ну і наостанок видалимо бакет з іменем `bucket02my`. Щоправда мушу зазначити, що метод `s3_client.delete_bucket()` видаляє лише порожні бакети, тому спершу слід видалити усі зайві файли (як це зроблено для файлу `data.csv` у рядку 116 Лістингу 10). Тож викликавши файл на виконання, переконаємося, що бакет справді видалений, за допомогою раніше написаної функції `printing_buckets()`. Результати успішного виконання – на Рис. 14.

Лістинг 10: Видалення бакету

```
112 def destroy_bucket(bucket_name):
113     s3_client = boto3.client("s3")
114
115     # First of all delete an existing file
116     s3_client.delete_object(Bucket = bucket_name, Key = "data.csv")
117
118     response = s3_client.delete_bucket(Bucket=bucket_name)
119     print(response)
120
121 destroy_bucket("bucket02my")
```

```
anton@anton: ~$ python3 lab4.py
Existing buckets:
bucket01my
anton@anton: ~$
```

Рис. 14: Перевірка видалення бакету

3. Робота з git-репозиторієм

Наведу ще раз умову завдання цього етапу: для результатів Лабораторної роботи №2 розробити bash-скрипт, який скопіює код з попередньо створеного git-репозиторію та встановить потрібні залежності `pip`. Тож з постановки завдання можна виокремити декілька підкроків:

- Підготувати на локальному сховищі такі два файли:

<code>lab2.py</code>	код Лабораторної роботи №2
<code>requirements.txt</code>	перелік бібліотек та пакетів <code>pip</code>

- Відкрити git-репозиторій та зберегти зазначені вище файли на ньому;
- На локальному сховищі написати необхідний bash-скрипт;
- Перекинути bash-скрипт на новостворений інстанс й запустити його.

Як результат: завдяки встановленню необхідних бібліотек та пакетів з файлу `requirements.txt` код `lab2.py` з Лабораторної роботи №2 має запуститися й успішно виконатися на новому інстансі.

Крок 1: підготовка програмних файлів

Перш за все я віднайшов файл з кодом `lab2.py`, суть якого полягала у візуалізації даних про курс валют протягом 2021 року. Маючи інструмент `pip` встановленим, наступним кроком командою

```
pip3 freeze > requirements.txt
```

я формую текстовий файл із назвами відповідних пакетів та бібліотек. На Рис. 15 показано процес створення відповідного файлу та перевірки його наповнення (демонструється лише початок файлу).

Крок 2: відкриття git-репозиторію

Створивши акаунт на <https://github.com> (або маючи вже створений), відкриваємо новий репозиторій. Усі подальші кроки можна виконувати як інструментами `git` через термінальний рядок, так і за допомогою інтерфейсу самого сайту. Я виконуватиму роботу через другий варіант: тож обираємо «uploading an existing file» та завантажуюмо підготовані на Кроці 1 файли. Результати й процес виконання зображені на Рис 16.

Назагал, нам ще зустрінеться робота з утилітою `git`, тож для попереднього тестування й ознайомлення цей інструмент можна завантажити на своє локальне сховище завдяки команді `sudo apt install git`.

```
ubuntu@ip-172-31-81-119: ~  
ubuntu@ip-172-31-81-119:~$ pip3 freeze > requirements.txt  
/home/ubuntu/.local/lib/python3.8/site-packages/pkg_resources/__init__.py:122: PkgResourcesDeprecationWarning: 0.1.36ubuntu1 is an invalid version and will not be supported in a future release  
  warnings.warn(  
/home/ubuntu/.local/lib/python3.8/site-packages/pkg_resources/__init__.py:122: PkgResourcesDeprecationWarning: 0.23ubuntu1 is an invalid version and will not be supported in a future release  
  warnings.warn(  
ubuntu@ip-172-31-81-119:~$ cat requirements.txt  
argon2-cffi==21.3.0  
argon2-cffi-bindings==21.2.0  
asttokens==2.0.5  
attrs==21.4.0  
Automat==0.8.0  
awscli==1.22.86  
backcall==0.2.0  
beautifulsoup4==4.10.0  
bleach==4.1.0  
blinker==1.4  
boto3==1.21.31  
botocore==1.24.31  
certifi==2019.11.28
```

Рис. 15: Підготовка необхідних файлів

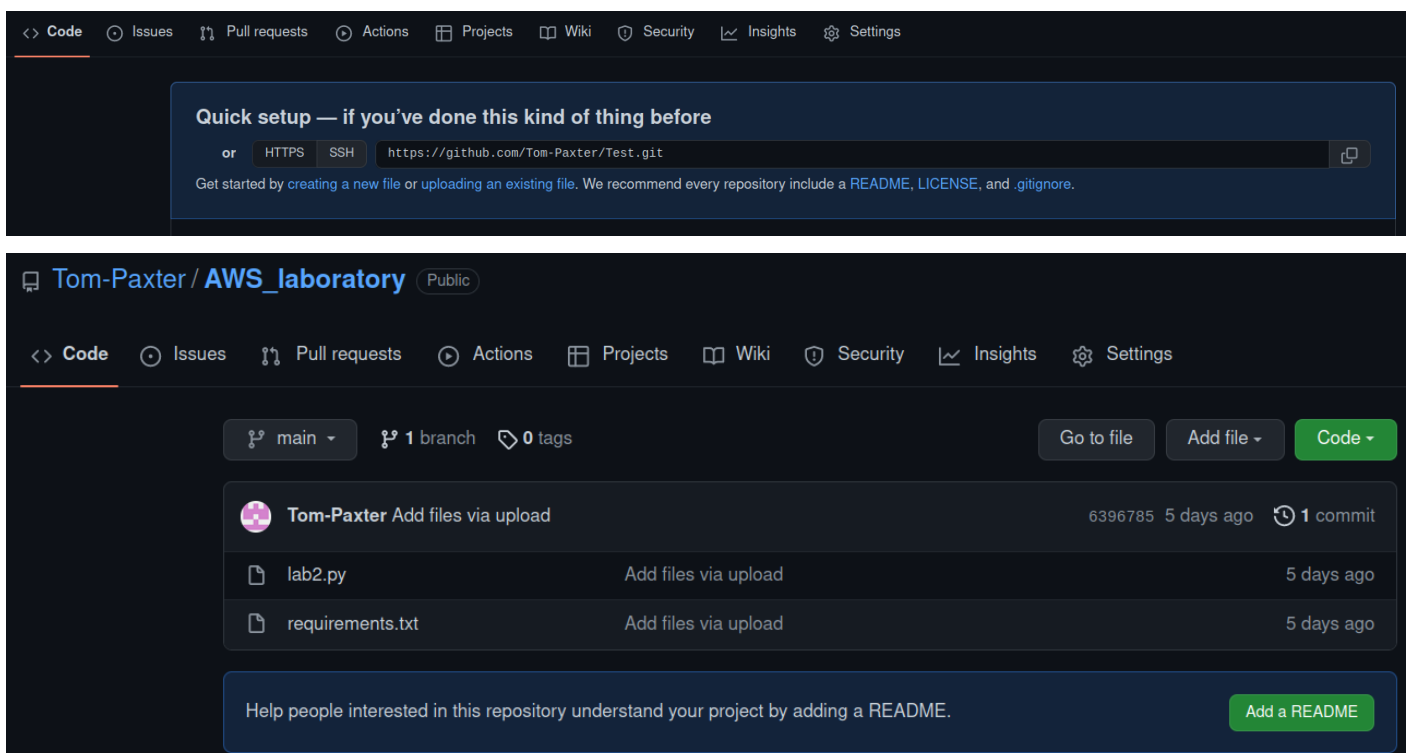


Рис. 16: Завантаження файлів на репозиторій

Крок 3: створення bash-скрипта

Bash-скрипт являє собою файл з набором термінальних команд. Тож створимо файл `commands.sh` приблизно такого змісту:

```
1  #!/bin/bash
2
3  sudo apt update
4  sudo apt install git
5  git clone https://github.com/Tom-Paxter/AWS_laboratory.git
6  sudo apt install python3-pip
7  while read requirement; do pip3 install $requirement; done <
    AWS_laboratory/requirements.txt
```

Завдання скрипта полягає у тому, щоб при його запуску на новому інстансі були встановлені усі необхідні інструменти для виконання програмного файлу `lab2.txt`. Кожен рядок коду виконує таку функцію:

- `sudo apt update` оновлення дерева пакетів для подальшого завантаження наступних інструментів;
- `sudo apt install git` встановлення термінальної утиліти `git`;
- `git clone` копіювання із репозиторію (url репозиторію) файлів `lab2.py` та `requirements.txt`;
- `sudo apt install python3-pip` завантаження менеджера `pip`;
- `while read requirement;` завантаження необхідних пакетів `pip` з вказаного файлу `requirements.txt`

Замість команди в рядку 7 спершу була використана більш зрозуміла на вигляд команда

```
pip3 install -r AWS_laboratory/requirements.txt
```

Проте її виконання призводило до певних помилок, тому її довелося модифікувати (детальніше про цю проблему в заключному розділі).

Крок 4: перевірка результатів завдання

Отже, наостанок залишилося створити новий інстанс, пернекинути на нього файл bash-скрипта та перевірити виконання завдання. Перш за все створюємо новий інстанс (Рис. 17), потім перекидаємо файл (Рис. 18) й перевіряємо його наявність на новому інстансі (Рис. 19).

Instances (2) [Info](#)

Search

< 1 >

Refresh

Connect

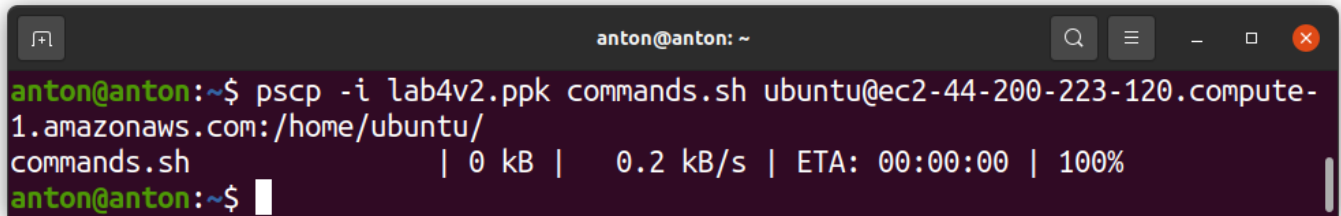
Instance state ▾

Actions ▾

Launch instances

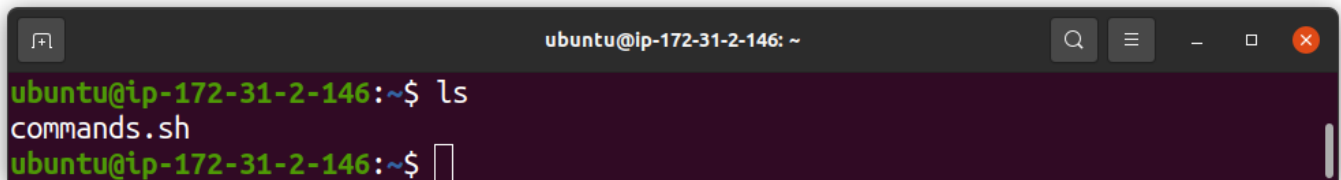
<input type="checkbox"/>	Name ▾	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability Zone ▾	Public IPv4 DNS ▾
<input type="checkbox"/>	first	i-03c0d5c1d9a8fb1ad	Running	t2.micro	2/2 checks passed	No alarms <div>+</div>	us-east-1b	ec2-18-207-204-243.co...
<input type="checkbox"/>	demo2	i-08190f5c50e7c6a4d	Running	t2.micro	-	No alarms <div>+</div>	us-east-1c	ec2-18-215-161-22.co...

Рис. 17: Новий інстанс «demo2»



```
anton@anton: ~  
anton@anton:~$ pscp -i lab4v2.ppk commands.sh ubuntu@ec2-44-200-223-120.compute-1.amazonaws.com:/home/ubuntu/  
commands.sh | 0 kB | 0.2 kB/s | ETA: 00:00:00 | 100%  
anton@anton:~$
```

Рис. 18: Копіювання файлу на новий істанс



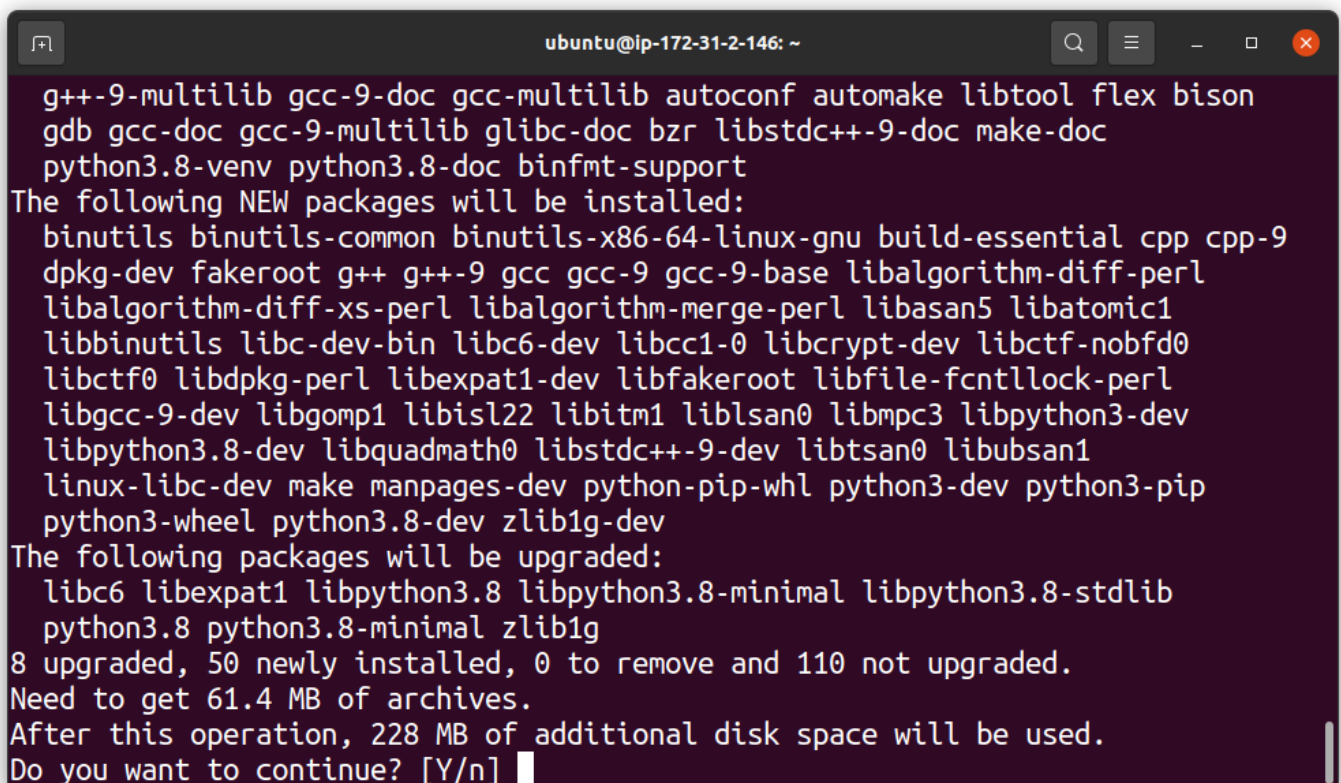
```
ubuntu@ip-172-31-2-146: ~  
ubuntu@ip-172-31-2-146:~$ ls  
commands.sh  
ubuntu@ip-172-31-2-146:~$
```

Рис. 19: Перевірка успішного копіювання

Усі підготовчі кроки виконано, тож нарешті підключаємося до інстансу й запускаємо bash-скрипт простою термінальною командою

```
bash commands.sh
```

Якщо все виконано правильно, то на новий порожній інстанс почнеться завантаження необхідного програмного забезпечення (показано на Рис 20 та Рис. 21).



```
ubuntu@ip-172-31-2-146: ~  
g++-9-multilib gcc-9-doc gcc-multilib autoconf automake libtool flex bison  
gdb gcc-doc gcc-9-multilib glibc-doc bzip libstdc++-9-doc make-doc  
python3.8-venv python3.8-doc binfmt-support  
The following NEW packages will be installed:  
binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-9  
dpkg-dev fakeroot g++ g++-9 gcc gcc-9 gcc-9-base libalgorithm-diff-perl  
libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1  
libbinutils libc-dev-bin libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0  
libctf0 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl  
libgcc-9-dev libgomp1 libisl22 libitm1 liblsan0 libmpc3 libpython3-dev  
libpython3.8-dev libquadmath0 libstdc++-9-dev libtsan0 libubsan1  
linux-libc-dev make manpages-dev python-pip-whl python3-dev python3-pip  
python3-wheel python3.8-dev zlib1g-dev  
The following packages will be upgraded:  
libc6 libexpat1 libpython3.8 libpython3.8-minimal libpython3.8-stdlib  
python3.8 python3.8-minimal zlib1g  
8 upgraded, 50 newly installed, 0 to remove and 110 not upgraded.  
Need to get 61.4 MB of archives.  
After this operation, 228 MB of additional disk space will be used.  
Do you want to continue? [Y/n]
```

Рис. 20: Завантаження менеджера pip

```
ubuntu@ip-172-31-2-146: ~  
Selecting previously unselected package libc6-dev:amd64.  
Preparing to unpack .../14-libc6-dev_2.31-0ubuntu9.7_amd64.deb ...  
Unpacking libc6-dev:amd64 (2.31-0ubuntu9.7) ...  
Selecting previously unselected package gcc-9-base:amd64.  
Preparing to unpack .../15-gcc-9-base_9.4.0-1ubuntu1~20.04.1_amd64.deb ...  
Unpacking gcc-9-base:amd64 (9.4.0-1ubuntu1~20.04.1) ...  
Selecting previously unselected package libisl22:amd64.  
Preparing to unpack .../16-libisl22_0.22.1-1_amd64.deb ...  
Unpacking libisl22:amd64 (0.22.1-1) ...  
Selecting previously unselected package libmpc3:amd64.  
Preparing to unpack .../17-libmpc3_1.1.0-1_amd64.deb ...  
Unpacking libmpc3:amd64 (1.1.0-1) ...  
Selecting previously unselected package cpp-9.  
Preparing to unpack .../18-cpp-9_9.4.0-1ubuntu1~20.04.1_amd64.deb ...  
Unpacking cpp-9 (9.4.0-1ubuntu1~20.04.1) ...  
Selecting previously unselected package cpp.  
Preparing to unpack .../19-cpp_4%3a9.3.0-1ubuntu2_amd64.deb ...  
Unpacking cpp (4:9.3.0-1ubuntu2) ...  
Progress: [ 21%] [#####.....]
```

Рис. 21: Процес завантаження необхідного ПЗ

Остаточно переконуємося в наявності бажаних файлів (Рис 22) й запускаємо на виконання `lab2.py`. Як результат – отримуємо поповнення в дереві файлів: щонайменше малюнок курсу валют `exchange_rate.png` (Рис. 23).

```
ubuntu@ip-172-31-2-146: ~/AWS_laboratory  
ubuntu@ip-172-31-2-146:~$ ls  
AWS_laboratory  commands.sh  
ubuntu@ip-172-31-2-146:~$ cd AWS_laboratory  
ubuntu@ip-172-31-2-146:~/AWS_laboratory$ ls  
lab2.py  requirements.txt  
ubuntu@ip-172-31-2-146:~/AWS_laboratory$
```

Рис. 22: Перевірка наявності файлів

```
ubuntu@ip-172-31-2-146: ~/AWS_laboratory  
ubuntu@ip-172-31-2-146:~/AWS_laboratory$ python3 lab2.py  
ubuntu@ip-172-31-2-146:~/AWS_laboratory$ ls  
exchange_rate.png  lab2.py  request.csv  request.json  requirements.txt  
ubuntu@ip-172-31-2-146:~/AWS_laboratory$
```

Рис. 23: Фінальна перевірка завдання

4. Перелік проблем протягом виконання роботи

1. Аналогічно до проблеми Лабораторної роботи №2 зіткнувся з питанням некоректного завантаження інструментів AWS CLI. Вдалося вирішити непорозуміння завдяки додатковому оновленню утиліт командою

```
pip3 install -upgrade awscli
```

2. Назагал, робота з git-репозиторієм є для мене новим досвідом. Створений акаунт я вже мав, проте реальної потреби зберігати на ньому певний код не виникло. Через відсутність практичних навичок були певні труднощі в опануванні термінального менеджера `git` – не вдалося створити репозиторії й перекинути туди бажані файли. Тому вирішив піти іншим шляхом (як не прикро, але шляхом новачка) – взаємодія з інтерфейсом сайту <https://github.com>.
3. При створенні `bash`-скрипта, як це вже зазначалося раніше, виникли проблеми із способом завантаження пакетів `pip` через команду

```
pip3 install -r AWS_laboratory/requirements.txt
```

Зіткнувся з аналогічною проблемою, яка зазначена [тут](#). У цій же статті віднайшов коректний спосіб оминати завантаження «проблемних» для терміналу бібліотек.