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

«Харні технології»

Лабораторна робота №4
«Автоматизація роботи з ресурсами AWS засобами мови Python»

Виконала:
студентка групи ФБ-95
Гурджия Валерія Вахтангівна

Київ-2022

ЗАВДАННЯ

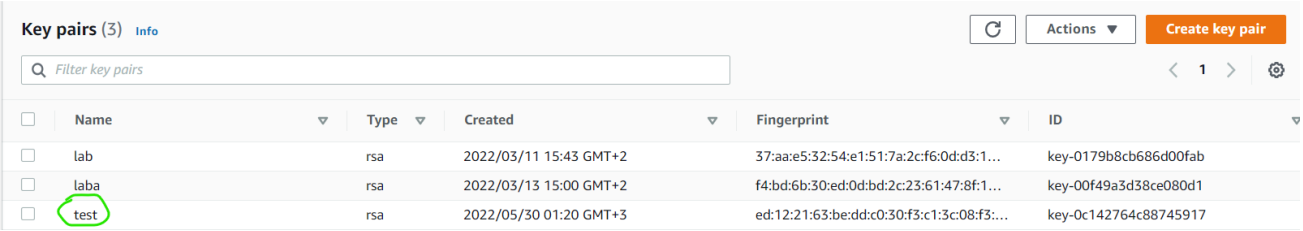
1. Розробити Python-скрипт для автоматичного створення та видалення хмарної інфраструктури з мінімальною конфігурацією (необхідно передбачити у функціях додаткові виключення для коректної роботи у нестандартних ситуаціях (наприклад, виводити відповідне повідомлення при створення бакету з вже існуючим ім'ям, при читанні з S3 файлу, якого там немає)
2. Для результатів лабораторної роботи 2 розробити bash-скрипт, який клонуватиме код з попередньо створеного git-репозиторію та встановить потрібні залежності (pip та необхідні залежності)
3. Результати усіх кроків оформити у вигляді детального протоколу зі скріншотами
4. Навести перелік проблем, вирішення яких було складним в ході виконання роботи в розділі висновків до протоколу

Автоматизація роботи з обчислювальними ресурсами EC

Створення ключової пари для доступу до EC2-інстансу.

```
import boto3
import os
import pandas
import botocore
from botocore.exceptions import ClientError
def create_key_pair(key_name, region_name):
    ec2_client = boto3.client("ec2", region_name)
    try:
        key_pair = ec2_client.create_key_pair(KeyName=key_name)
        private_key = key_pair["KeyMaterial"]
        print("The key pair", key_name, "successfully created!")

    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidKeyPair.Duplicate":
                print("The key pair", key_name, "already exist!")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("Endpoint Connection Error:", "Check your region name!")
        else:
            print("Error:", e)

create_key_pair("test", "us-west-1")
```

```
The key pair test successfully created!
```



Key pairs (3) Info

Actions | Create key pair

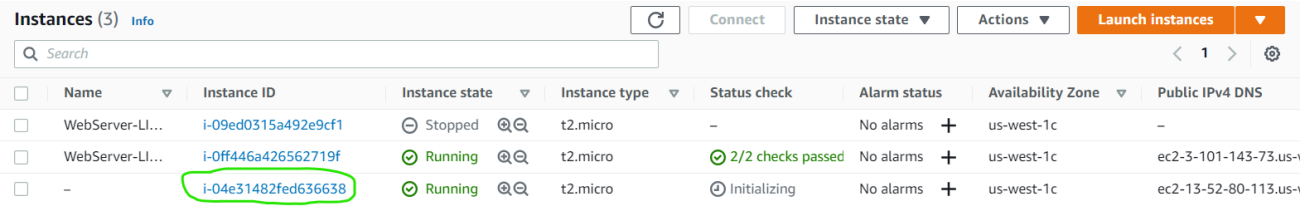
Filter key pairs

Name	Type	Created	Fingerprint	ID
lab	rsa	2022/03/11 15:43 GMT+2	37:aa:e5:32:54:e1:51:7a:2c:f6:0d:d3:1...	key-0179b8cb686d00fab
laba	rsa	2022/03/13 15:00 GMT+2	f4:bd:6b:30:ed:0d:bd:2c:23:61:47:8f:1...	key-00f49a3d38ce080d1
test	rsa	2022/05/30 01:20 GMT+3	ed:12:21:63:be:dd:c0:30:f3:c1:3c:08:f3:...	key-0c142764c88745917

Створення EC2 інстансу засобами boto3

```python
def create_instance(region_name, InstanceType, KeyName):
    ec2_client = boto3.client("ec2", region_name)
    try:
        instances = ec2_client.run_instances(
        ImageId="ami-0a8a24772b8f01294",
        MinCount=1,
        MaxCount=1,
        InstanceType=InstanceType,
        KeyName=KeyName
        )
        print("Instance successfully created!")
        print("Instance ID:", instances["Instances"][0]["InstanceId"])
    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidKeyPair.NotFound":
                print("key not found")
            elif e.response['Error']['Code'] == "InvalidAMIID.Malformed":
                print("Wrong ImageID")
            elif e.response['Error']['Code'] == "InvalidAMIID.NotFound":
                print("ImageID not found")
            elif e.response['Error']['Code'] == "InvalidParameterValue":
                print("Wrong instance type")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            print("Error:", e)
create_instance("us-west-1", "t2.micro", "test")
```

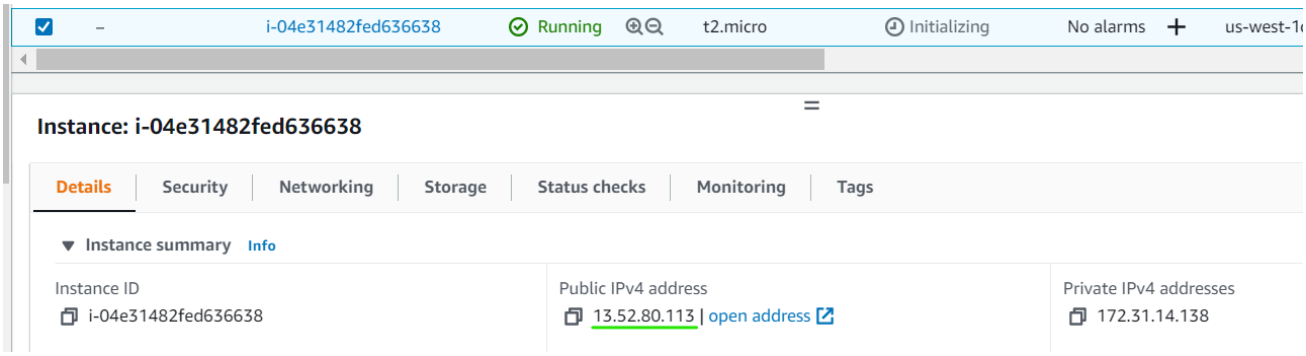
```
Instance successfully created!
Instance ID: i-04e31482fed636638
```



Отримання ip-адреси створеного інстансу

```
def get_public_ip(instance_id, region_name):
    ec2_client = boto3.client("ec2", region_name)
    try:
        reservations = ec2_client.describe_instances(InstanceIds=[instance_id]).get("Reservations")
        for reservation in reservations:
            for instance in reservation['Instances']:
                print("Public Ip Address:", instance.get("PublicIpAddress"))

    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidInstanceID.Malformed":
                print("Invalid instance ID!")
            elif e.response['Error']['Code'] == "InvalidInstanceID.NotFound":
                print("Instance ID does not exist!")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            print("Error:", e)

get_public_ip("i-04e31482fed636638", "us-west-1")
```

```
Public Ip Address: 13.52.80.113
```



Автоматизація моніторингу активних інстансів

```python
def get_running_instances(region_name):
    ec2_client = boto3.client("ec2", region_name)
    try:
        reservations = ec2_client.describe_instances(Filters=[
            {
                "Name": "instance-state-name",
                "Values": ["running"],
            },
            {
                "Name": "instance-type",
                "Values": ["t2.micro"]
            }
        ]).get("Reservations")

        count=0
        for reservation in reservations:
            count+=1
            for instance in reservation["Instances"]:
                instance_id = instance["InstanceId"]
                instance_type = instance["InstanceType"]
                public_ip = instance["PublicIpAddress"]
                private_ip = instance["PrivateIpAddress"]
                print(f"{instance_id}, {instance_type}, {public_ip}, {private_ip}")
        print("Count of running instances:", count)

    except Exception as e:
        if type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            print("Error:", e)

get_running_instances("us-west-1")
```

```
i-0ff446a426562719f, t2.micro, 3.101.143.73, 172.31.11.110
i-04e31482fed636638, t2.micro, 13.52.80.113, 172.31.14.138
Count of running instances: 2
```

	Name	Instance ID	Instance state
☐	WebServer-LI...	i-09ed0315a492e9cf1	Stopped
☐	WebServer-LI...	i-0ff446a426562719f	Running
☑	–	i-04e31482fed636638	Running

Зупинка інстансу

```python
def stop_instance(instance_id, region_name):
    ec2_client = boto3.client("ec2", region_name)
    try:
        response = ec2_client.stop_instances(InstanceIds=[instance_id])
        print("Instance was successfully stoped!")
        print(response)

    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidInstanceID.Malformed":
                print("Invalid instance ID!")
            elif e.response['Error']['Code'] == "InvalidInstanceID.NotFound":
                print("Instance ID does not exist!")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            print("Error:", e)

stop_instance("i-04e31482fed636638", "us-west-1")
```

```
Instance was successfully stoped!
{'StoppingInstances': [{'CurrentState': {'Code': 64, 'Name': 'stopping'}, 'InstanceId': 'i-04e31482fed636638', 'PreviousState':
{'Code': 16, 'Name': 'running'}}], 'ResponseMetadata': {'RequestId': 'a3a6f905-bd52-4dcd-91b8-d279f6bbd6ce', 'HTTPStatusCode':
200, 'HTTPHeaders': {'x-amzn-requestid': 'a3a6f905-bd52-4dcd-91b8-d279f6bbd6ce', 'cache-control': 'no-cache, no-store', 'strict
-transport-security': 'max-age=31536000; includeSubDomains', 'content-type': 'text/xml;charset=UTF-8', 'content-length': '579',
'date': 'Sun, 29 May 2022 22:35:23 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}
```

WebServer-LI...	i-09ed0315a492e9cf1	Stopped
WebServer-LI...	i-0ff446a426562719f	Running
–	i-04e31482fed636638	Stopped

Видалення (термінація) непотрібного інстансу

```python
def terminate_instance(instance_id, region_name):
    ec2_client = boto3.client("ec2", region_name)
    try:
        response = ec2_client.terminate_instances(InstanceIds=[instance_id])
        print("Instance was successfully terminated!")
        print(response)
    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidInstanceID.Malformed":
                print("Invalid instance ID!")
            elif e.response['Error']['Code'] == "InvalidInstanceID.NotFound":
                print("Instance ID does not exist!")
            elif e.response['Error']['Code'] == "OperationNotPermitted":
                print("Operation Not Permitted!")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            print("Error:", e)

terminate_instance("i-04e31482fed636638", "us-west-1")
```

```
Instance was successfully terminated!
{'TerminatingInstances': [{'CurrentState': {'Code': 48, 'Name': 'terminated'}, 'InstanceId': 'i-04e31482fed636638', 'PreviousSt
ate': {'Code': 80, 'Name': 'stopped'}}], 'ResponseMetadata': {'RequestId': 'dfb6881e-2771-48b9-9758-2a1f9357035d', 'HTTPStatusC
ode': 200, 'HTTPHeaders': {'x-amzn-requestid': 'dfb6881e-2771-48b9-9758-2a1f9357035d', 'cache-control': 'no-cache, no-store',
'strict-transport-security': 'max-age=31536000; includeSubDomains', 'vary': 'accept-encoding', 'content-type': 'text/xml;charse
t=UTF-8', 'transfer-encoding': 'chunked', 'date': 'Sun, 29 May 2022 22:37:47 GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}
```

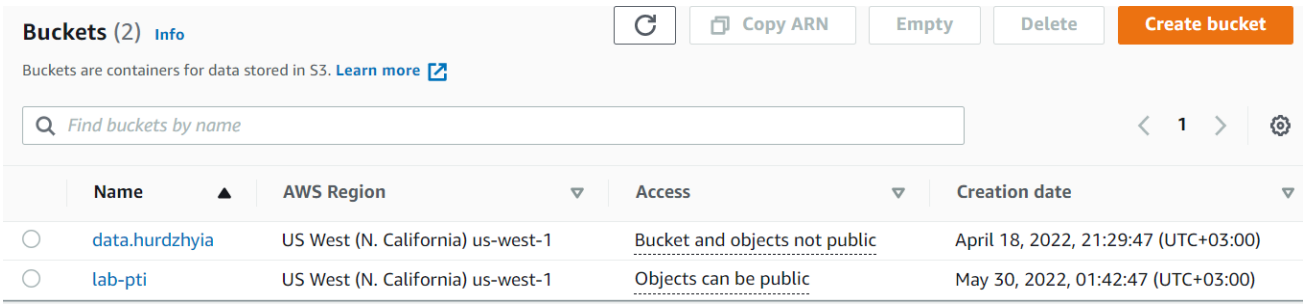
WebServer-LI...	i-09ed0315a492e9cf1	Stopped
WebServer-LI...	i-0ff446a426562719f	Running
–	i-04e31482fed636638	Terminated

Автоматизація роботи з сховищем S3

Створення бакета S3

```
def create_bucket(bucket_name, region):
    s3_client = boto3.client('s3', region_name=region)
    try:
        location = {'LocationConstraint': region}
        response = s3_client.create_bucket(Bucket=bucket_name, CreateBucketConfiguration=location)
        print("Bucket was successfully created!")
        print(response)
    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidBucketName":
                print("Invalid Bucket Name!")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            if e.response['Error']['Code'] == "BucketAlreadyOwnedByYou":
                print("You already have bucket with this name!")
            elif e.response['Error']['Code'] == "BucketAlreadyExists":
                print("Bucket with this name already exist!")
            else:
                print("Error:", e)
                return e
create_bucket("lab-pti", "us-west-1")
```

```
Bucket was successfully created!
{'ResponseMetadata': {'RequestId': 'VA6AWH2NAMAY3ST7', 'HostId': 'fMAmiqvCVz2cfWQ6GBxPXQ2cigL54SkdLD27wHRipyRbDsDITn5pIWcDomvPz
NeEn4BI2oqSdck=', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amz-id-2': 'fMAmiqvCVz2cfWQ6GBxPXQ2cigL54SkdLD27wHRipyRbDsDITn5pIWc
DomvPzNeEn4BI2oqSdck=', 'x-amz-request-id': 'VA6AWH2NAMAY3ST7', 'date': 'Sun, 29 May 2022 22:42:47 GMT', 'location': 'http://la
b-pti.s3.amazonaws.com/', 'server': 'AmazonS3', 'content-length': '0'}, 'RetryAttempts': 0}, 'Location': 'http://lab-pti.s3.ama
zonaws.com/'}
```



Лістинг існуючих бакетів облікового запису

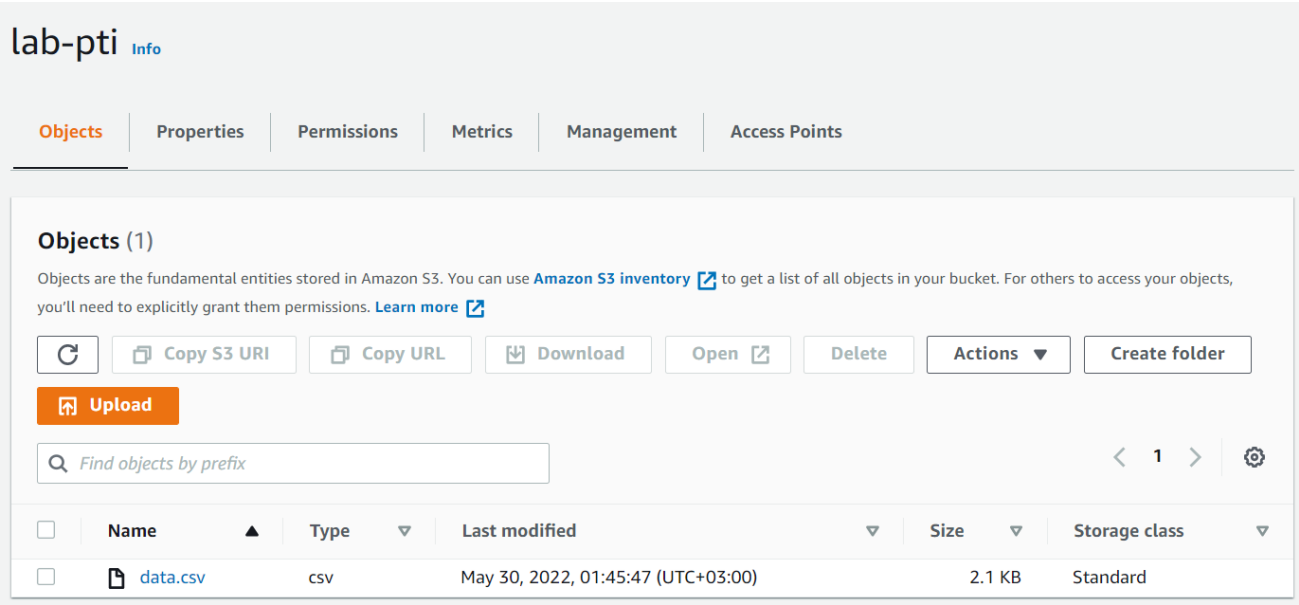
```
def show_existing_buckets():
    s3 = boto3.client('s3')
    response = s3.list_buckets()
    print("Existing buckets:")
    for bucket in response['Buckets']:
        print(f' {bucket["Name"]}')
show_existing_buckets()
```

```
Existing buckets:
 data.hurdzhyia
 lab-pti
```

Завантаження файлу

```python
def upload(file_name, bucket_name, s3_obj_name):
    s3_client = boto3.client('s3')
    try:
        response = s3_client.upload_file(Filename=file_name, Bucket=bucket_name, Key=s3_obj_name)
        print("file was successfully uploaded!")
        print(response)

    except FileNotFoundError:
        print("File not found!")

    except boto3.exceptions.S3UploadFailedError as e:
        print("No such bucket or access to this bucket denied")

    except Exception as e:
        print("Error:", e)

upload("data.csv", "lab-pti", "data.csv")
```

```
file was successfully uploaded!
None
```



Читання даних з s3

```
def show_context_of_bucket(bucket_name, key_name):
    s3_client = boto3.client('s3')
    try:
        obj = s3_client.get_object(Bucket = bucket_name, Key = key_name)
        # Read data from the S3 object
        data = pandas.read_csv(obj['Body'])
        # Print the data frame
        print('Printing the data frame...')
        print(data.head())

    except ClientError as e:
        print("Client Error:", end=' ')
        if e.response['Error']['Code'] == "NoSuchBucket":
            print("No Such Bucket")
        elif e.response['Error']['Code'] == "NoSuchKey":
            print("No Such File")
        elif e.response['Error']['Code'] == "AccessDenied":
            print("Access Denied")
        else:
            print("Error:", e)

show_context_of_bucket("lab-pti", "data.csv")
```

```
Printing the data frame...
    StartDate  TimeSign  CurrencyCode CurrencyCodeL  Units   Amount
0  01.01.2021         0            36           AUD      1  21.6852
1  01.01.2021         0           944           AZN      1  16.6439
2  01.01.2021         0           933           BYN      1  10.9477
3  01.01.2021         0           975           BGN      1  17.7571
4  01.01.2021         0           410           KRW    100   2.6015
```

Видалення непотрібного бакета (лише для пустого бакету)
Спробували видалити непорожній бакет

```
def destroy_bucket(bucket_name):
    s3_client = boto3.client('s3')
    try:
        response = s3_client.delete_bucket(Bucket=bucket_name)
        print("Bucket successfully deleted!")
        print(response)

    except ClientError as e:
        print("Client Error:", end=' ')
        if e.response['Error']['Code'] == "NoSuchBucket":
            print("No Such Bucket")
        elif e.response['Error']['Code'] == "BucketNotEmpty":
            print("Bucket is not Empty")
        elif e.response['Error']['Code'] == "AccessDenied":
            print("Access Denied")
        else:
            print("Error:", e)
destroy_bucket("lab-pti")
```

```
Client Error: Bucket is not Empty
```

Код програми

```
import boto3
import os
import pandas
import botocore
from botocore.exceptions import ClientError

def create_key_pair(key_name, region_name):
    ec2_client = boto3.client("ec2", region_name)
    try:
        key_pair = ec2_client.create_key_pair(KeyName=key_name)
        private_key = key_pair["KeyMaterial"]
        print("The key pair", key_name, "successfully created!")

    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidKeyPair.Duplicate":
                print("The key pair", key_name, "already exist!")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("Endpoint Connection Error:", "Check your region name!")
        else:
            print("Error:", e)

#create_key_pair("testik", "eu-west-1")



def create_instance(region_name, InstanceType, KeyName):
    ec2_client = boto3.client("ec2", region_name)
    try:
        instances = ec2_client.run_instances(
        ImageId="ami-0a8a24772b8f01294",
        MinCount=1,
        MaxCount=1,
        InstanceType=InstanceType,
        KeyName=KeyName
        )
        print("Instance successfully created!")
        print("Instance ID:", instances["Instances"][0]["InstanceId"])
    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidKeyPair.NotFound":
                print("key not found")
            elif e.response['Error']['Code'] == "InvalidAMIID.Malformed":
                print("Wrong ImageID")
            elif e.response['Error']['Code'] == "InvalidParameterValue":
                print("Wrong instance type")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            print("Error:", e)
#create_instance("us-west-1", "t2.micro", "testik")



def get_public_ip(instance_id, region_name):
```

```
    ec2_client = boto3.client("ec2", region_name)
    try:
        reservations =
ec2_client.describe_instances(InstanceIds=[instance_id]).get("Reservations")
        for reservation in reservations:
            for instance in reservation['Instances']:
                print("Public Ip Address:", instance.get("PublicIpAddress"))

    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidInstanceID.Malformed":
                print("Invalid instance ID!")
            elif e.response['Error']['Code'] == "InvalidInstanceID.NotFound":
                print("Instance ID does not exist!")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            print("Error:", e)

#get_public_ip("i-06864239172e16c52", "us-west-2")



def get_running_instances(region_name):
    ec2_client = boto3.client("ec2", region_name)
    try:
        reservations = ec2_client.describe_instances(Filters=[
            {
                "Name": "instance-state-name",
                "Values": ["running"],
            },
            {
                "Name": "instance-type",
                "Values": ["t2.micro"]
            }
        ]).get("Reservations")

        count=0
        for reservation in reservations:
            count+=1
            for instance in reservation["Instances"]:
                instance_id = instance["InstanceId"]
                instance_type = instance["InstanceType"]
                public_ip = instance["PublicIpAddress"]
                private_ip = instance["PrivateIpAddress"]
                print(f"{instance_id}, {instance_type}, {public_ip}, {private_ip}")
        print("Count of running instances:", count)

    except Exception as e:
        if type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            print("Error:", e)

#get_running_instances("us-west-1")



def stop_instance(instance_id, region_name):
```

```
    ec2_client = boto3.client("ec2", region_name)
    try:
        response = ec2_client.stop_instances(InstanceIds=[instance_id])
        print("Instance was successfully stoped!")
        print(response)

    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidInstanceID.Malformed":
                print("Invalid instance ID!")
            elif e.response['Error']['Code'] == "InvalidInstanceID.NotFound":
                print("Instance ID does not exist!")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            print("Error:", e)

#stop_instance("i-0f982df6b1f82b8b2", "us-west-1")




def terminate_instance(instance_id, region_name):
    ec2_client = boto3.client("ec2", region_name)
    try:
        response = ec2_client.terminate_instances(InstanceIds=[instance_id])
        print("Instance was successfully terminated!")
        print(response)
    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidInstanceID.Malformed":
                print("Invalid instance ID!")
            elif e.response['Error']['Code'] == "InvalidInstanceID.NotFound":
                print("Instance ID does not exist!")
            elif e.response['Error']['Code'] == "OperationNotPermitted":
                print("Operation Not Permitted!")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            print("Error:", e)

#terminate_instance("i-09ed0315a492e9cf1", "us-west-1")




def create_bucket(bucket_name, region):
    s3_client = boto3.client('s3', region_name=region)
    try:
        location = {'LocationConstraint': region}
        response = s3_client.create_bucket(Bucket=bucket_name,
CreateBucketConfiguration=location)
        print("Bucket was successfully created!")
        print(response)
```

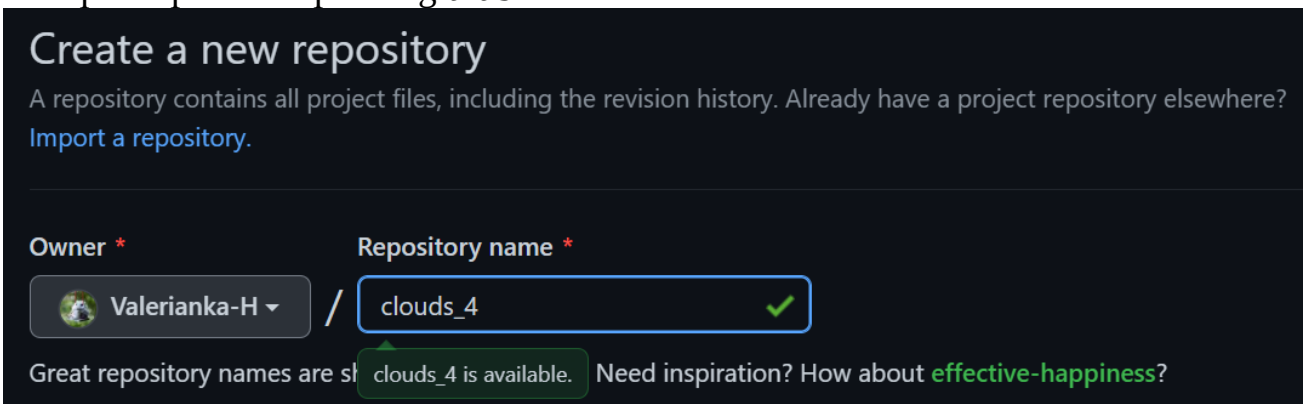
```
    except Exception as e:
        if type(e) == botocore.exceptions.ClientError:
            print("Client Error:", end=' ')
            if e.response['Error']['Code'] == "InvalidBucketName":
                print("Invalid Bucket Name!")
            else:
                print("Error:", e)
        elif type(e) == botocore.exceptions.EndpointConnectionError:
            print("EndpointConnectionError:", "Check your region name!")
        else:
            if e.response['Error']['Code'] == "BucketAlreadyOwnedByYou":
                print("You already have bucket with this name!")
            elif e.response['Error']['Code'] == "BucketAlreadyExists":
                print("Bucket with this name already exist!")
            else:
                print("Error:", e)
                return e
#create_bucket("lab-pti", "us-west-1")



def show_existing_buckets():
    s3 = boto3.client('s3')
    response = s3.list_buckets()
    print("Existing buckets:")
    for bucket in response['Buckets']:
        print(f' {bucket["Name"]}')
#show_existing_buckets()


def upload(file_name, bucket_name, s3_obj_name):
    s3_client = boto3.client('s3')
    try:
        response = s3_client.upload_file(Filename=file_name, Bucket=bucket_name,
Key=s3_obj_name)
        print("file was successfully uploaded!")
        print(response)

    except FileNotFoundError:
        print("File not found!")

    except boto3.exceptions.S3UploadFailedError as e:
        print("No such bucket or access to this bucket denied")

    except Exception as e:
        print("Error:", e)

#upload("data.csv", "lab--pti", "data.csv")



def show_context_of_bucket(bucket_name, key_name):
    s3_client = boto3.client('s3')
    try:
        obj = s3_client.get_object(Bucket = bucket_name, Key = key_name)
        # Read data from the S3 object
        data = pandas.read_csv(obj['Body'])
        # Print the data frame
        print('Printing the data frame...')
        print(data.head())

    except ClientError as e:
```

```
        print("Client Error:", end=' ')
        if e.response['Error']['Code'] == "NoSuchBucket":
            print("No Such Bucket")
        elif e.response['Error']['Code'] == "NoSuchKey":
            print("No Such File")
        elif e.response['Error']['Code'] == "AccessDenied":
            print("Access Denied")
        else:
            print("Error:", e)

#show_context_of_bucket("lab-pti", "data.csv")


def destroy_bucket(bucket_name):
    s3_client = boto3.client('s3')
    try:
        response = s3_client.delete_bucket(Bucket=bucket_name)
        print("Bucket successfully deleted!")
        print(response)

    except ClientError as e:
        print("Client Error:", end=' ')
        if e.response['Error']['Code'] == "NoSuchBucket":
            print("No Such Bucket")
        elif e.response['Error']['Code'] == "BucketNotEmpty":
            print("Bucket is not Empty")
        elif e.response['Error']['Code'] == "AccessDenied":
            print("Access Denied")
        else:
            print("Error:", e)
#destroy_bucket("lab-pti")


if __name__ == "__main__":
    while True:
        print("Amazon Web Service")
        print("1) Enter to EC2")
        print("2) Enter to S3")
        print("3) Exit")

        print("Your choice:", end=' ')
        choice = int(input())

        if choice == 1:

            while True:
                print("1) Create Key Pair")
                print("2) Create Instance")
                print("3) Get Public IP")
                print("4) Get Running Instances")
                print("5) Stop Instance")
                print("6) Terminate Instance")
                print("7) Exit")

                print("Your choice:", end=' ')
                choice = int(input())

                if choice == 1:
                    print("Enter key name:", end=' ')
                    key_name = input()
                    print("Enter region:", end=' ')
                    region_name = input()
```

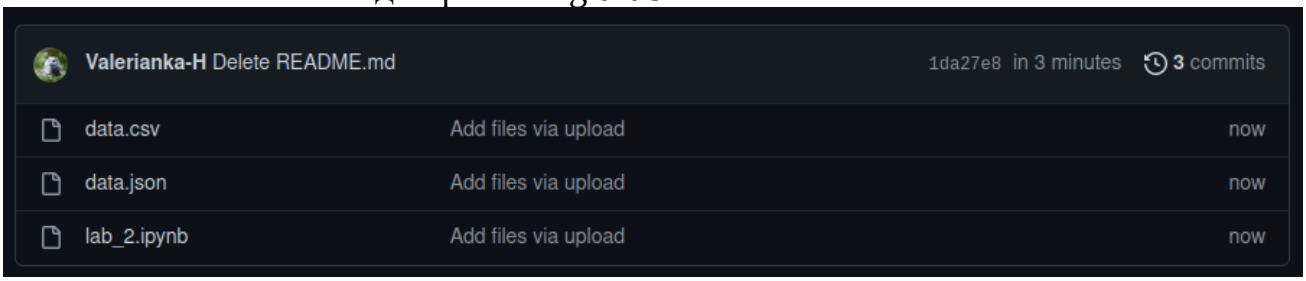
```
            create_key_pair(key_name, region_name)
        elif choice == 2:
            print("Enter region:", end=' ')
            region_name = input()
            print("Enter instance type:", end=' ')
            InstanceType = input()
            print("Enter key pair:", end=' ')
            KeyName = input()
            create_instance(region_name, InstanceType, KeyName)
        elif choice == 3:
            print("Enter instance ID:", end=' ')
            instance_id = input()
            print("Enter region:", end=' ')
            region_name = input()
            get_public_ip(instance_id, region_name)
        elif choice == 4:
            print("Enter region:", end=' ')
            region_name = input()
            get_running_instances(region_name)
        elif choice == 5:
            print("Enter instance ID:", end=' ')
            instance_id = input()
            print("Enter region:", end=' ')
            region_name = input()
            stop_instance(instance_id, region_name)
        elif choice == 6:
            print("Enter instance ID:", end=' ')
            instance_id = input()
            print("Enter region:", end=' ')
            region_name = input()
            terminate_instance(instance_id, region_name)
        elif choice == 7:
            print("bye")
            break
        else:
            print("Try again")


elif choice == 2:
    while True:
        print("1) Create Bucket")
        print("2) Show Existing Buckets")
        print("3) Upload File")
        print("4) Show Context Of Bucket")
        print("5) destroy_bucket")
        print("6) Exit")

        print("Your choice:", end=' ')
        choice = int(input())

        if choice == 1:
            print("Enter Bucket name:", end=' ')
            bucket_name = input()
            print("Enter region:", end=' ')
            region = input()
            create_bucket(bucket_name, region)
        elif choice == 2:
            show_existing_buckets()
        elif choice == 3:
            print("Enter File name:", end=' ')
            file_name = input()
            print("Enter Bucket name:", end=' ')
```

```
            bucket_name = input()
            print("Enter Object name:", end=' ')
            s3_obj_name = input()
            upload(file_name, bucket_name, s3_obj_name)
        elif choice == 4:
            print("Enter Bucket name:", end=' ')
            bucket_name = input()
            print("Enter Key name:", end=' ')
            key_name = input()
            show_context_of_bucket(bucket_name, key_name)
        elif choice == 5:
            print("Enter Bucket name:", end=' ')
            bucket_name = input()
            destroy_bucket(bucket_name)
        elif choice == 6:
            print("bye")
            break
        else:
            pritn("Try again")
elif choice == 3:
    print("Good Bye!")
    break;
else:
    print("Try again")
```

Створимо репозиторій на github



Завантажили всі необхідні фали на github



Bash-скрипт, який клонуватиме код з попередньо створеного git-репозиторію та встановить потрібні залежності

```
#! /bin/bash

echo "Installing..."
pip install boto3
sudo apt install python3-pip
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
sudo pip3 install pandas
sudo pip3 install matplotlib
sudo pip3 install jupyter notebook
echo "Done!"

echo "Cloning.."
git clone https://github.com/Valerianka-H/clouds_4
echo "Done!"
```

Запустимо bash-скрипт

```
[ec2-user@ip-172-31-11-110 ~]$ sh bash_script.sh
Installing...
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 202
hon 2.7 is no longer maintained. pip 21.0 will drop support for Python
out Python 2 support in pip can be found at https://pip.pypa.io/en/late
on-2-support pip 21.0 will remove support for this functionality.
```

```
[ec2-user@ip-172-31-11-110 ~]$ ls -l
total 119948
-rw-rw-r--  1 ec2-user ec2-user       74 Apr 23 19:13 a.json
drwxr-xr-x  3 ec2-user ec2-user       78 May 30 12:44 aws
-rw-rw-r--  1 ec2-user ec2-user 47057660 May 30 12:44 awscliv2.zip
-rw-r--r--  1 ec2-user ec2-user      386 May 30 12:42 bash_script.sh
drwxrwxr-x  3 ec2-user ec2-user       70 May 30 12:43 clouds_4
-rw-rw-r--  1 ec2-user ec2-user       25 Apr 26 16:58 conf.json
-rw-rw-r--  1 ec2-user ec2-user     5034 Apr 22 16:28 currency.png
-rw-rw-r--  1 ec2-user ec2-user     2117 Apr 22 16:27 data.csv
-rw-rw-r--  1 ec2-user ec2-user     7492 Apr 23 19:18 data.json
-rw-r--r--  1 ec2-user ec2-user     1876 Apr 25 15:35 example.json
-rw-rw-r--  1 ec2-user ec2-user        0 Mar 14 18:45 file1.txt
-rw-r--r--  1 ec2-user ec2-user      285 Apr 25 13:44 file.json
-rwxrwx---  1 ec2-user ec2-user       13 Mar 14 19:16 file.txt
-rw-rw-r--  1 ec2-user ec2-user    32473 Apr 22 16:28 lab_2.ipynb
-rw-rw-r--  1 ec2-user ec2-user    24423 May 29 23:04 lab_4.ipynb
drwxrwxr-x 16 ec2-user ec2-user      256 Apr 21 18:54 miniconda3
-rwxrwx--x  1 ec2-user ec2-user 75660608 Apr 20 14:55 Miniconda3-py39_4.11.0-Linux-
x86_64.sh
-rw-r--r--  1 ec2-user ec2-user     1005 Apr 26 13:38 students.json
[ec2-user@ip-172-31-11-110 ~]$
```

Висновок: під час виконання лабораторної роботи я навчилася працювати з такими ресурсами AWS, як EC2 та S3 за допомогою python скриптів, та більш детальніше ознайомилася з бібліотекою boto3.