ХМАРНІ ТЕХНОЛОГІЇ ОБРОБКИ ДАНИХ

Chapter · May 2022

CITATIONS
0

READS
53

3 authors:

Andrey Yu. Shelestov
National Technical University of Ukraine Kyiv Polytechnic Institute
181 PUBLICATIONS **5,159** CITATIONS
SEE PROFILE

Andrii Kolotii
National Academy of Sciences of Ukraine
64 PUBLICATIONS **907** CITATIONS
SEE PROFILE

Hanna Yailymova
National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”
72 PUBLICATIONS **303** CITATIONS
SEE PROFILE

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ХМАРНІ ТЕХНОЛОГІЇ ОБРОБКИ ДАНИХ

Лабораторний практикум

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньо-науковою програмою «Математичні методи моделювання,
розпізнавання образів та комп'ютерного зору»
спеціальності 113 «Прикладна математика»*

Київ
КПІ ім. Ігоря Сікорського
2022

Хмарні технології обробки даних. Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 113 «Прикладна математика» / А. Ю. Шелестов, А.В. Колотій; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12 531 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2022. – 53 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 5 від 26.05.2022 р.) за поданням Вченої ради НН ФТІ Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (протокол № 4 від18.04.2022 р.)

Електронне мережне навчальне видання

ХМАРНІ ТЕХНОЛОГІЇ ОБРОБКИ ДАНИХ

Лабораторний практикум

Автори:

Шелестов Андрій Юрійович, д. техн. наук, проф.
Колотій Андрій Всеволодович, к. техн. наук

Відповідальний редактор *Смирнов С.А.,* к.ф.-м.н., доц.

Рецензент *Лавренюк А.М.*., канд. техн. наук, доц. кафедри математичного моделювання та аналізу даних НН ФТІ Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського"

Навчальний посібник «Хмарні технології обробки даних. Лабораторний практикум» присвячено ознайомленню з підходами до використання хмарних платформ за спеціальністю 113 «Прикладна математика» та може бути корисним для здобувачів вищої освіти інших технічних спеціальностей. Метою навчальної дисципліни є формування у студентів здатностей засвоєння принципів використання сучасних інформаційних технологій хмарних платформ та їх застосування для розв’язанні прикладних задач, оволодіння практичними навичками використання цих інструментів. Посібник містить необхідний теоретичний матеріал, приклади програм, а також завдання для виконання лабораторного практикуму.

ЗМІСТ

Лабораторна робота №1
Використання AWS Management Console

Мета роботи: отримати базові навички по використанню AWS Management Console та створити власний мікро-сервер для подальшого використання.

1.1. План виконання

- Реєстрація в AWS
- Створення власного віртуального мікро-сервера
- Отримання віддаленого доступу через SSH
- Вивчення елементів моніторингу серверу та налаштування
- Документування зробленої роботи у вигляді деталізованого протоколу зкоментарями

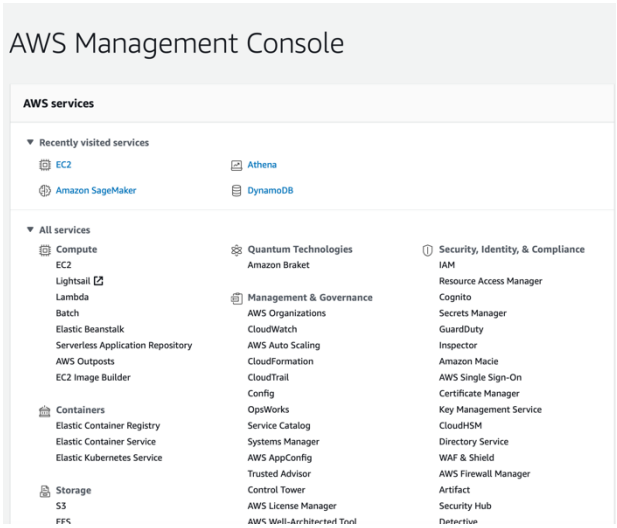
Довідкова інформація: розділ 3 книги Andreas Wittig, Michael Wittig "Amazon WebServices in Action"

1.2. Порядок виконання роботи

Для початку потрібно зареєструватись у AWS за посиланням https://portal.aws.amazon.com/billing/signup#/start

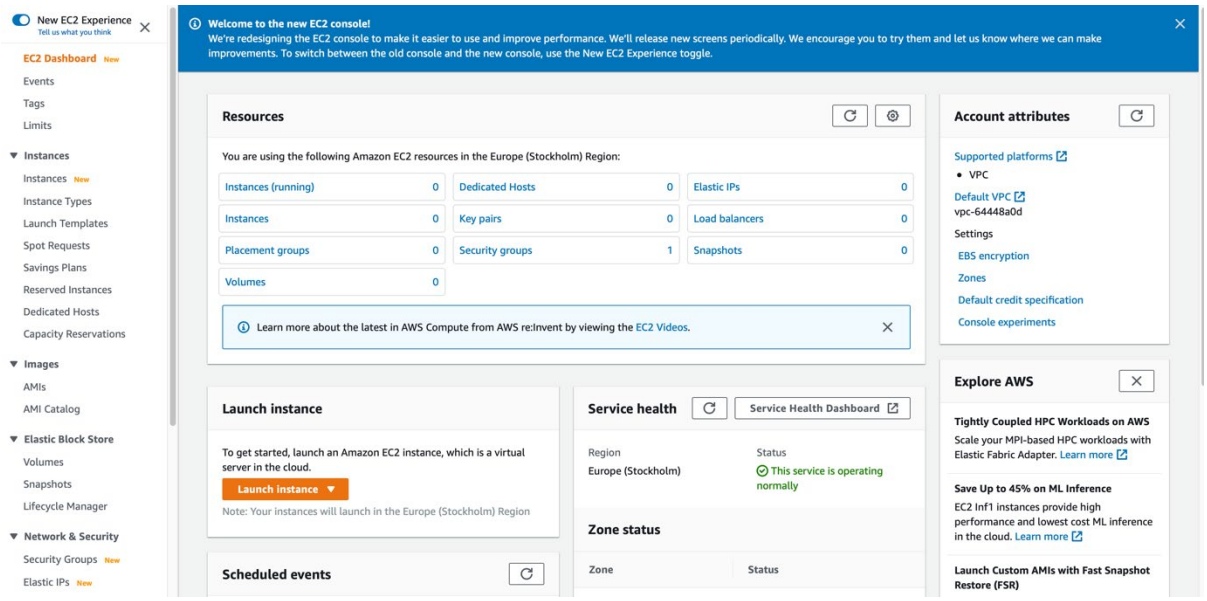
Увага: для реєстрації потрібно вказати банківську картку, з якої Amazon спише та поверне еквівалент 1$

Після реєстрації має бути доступною Amazon Management Console з цілим спектром сервісів. В даній роботі нас цікавить EC2 (Elastic Computing Service)



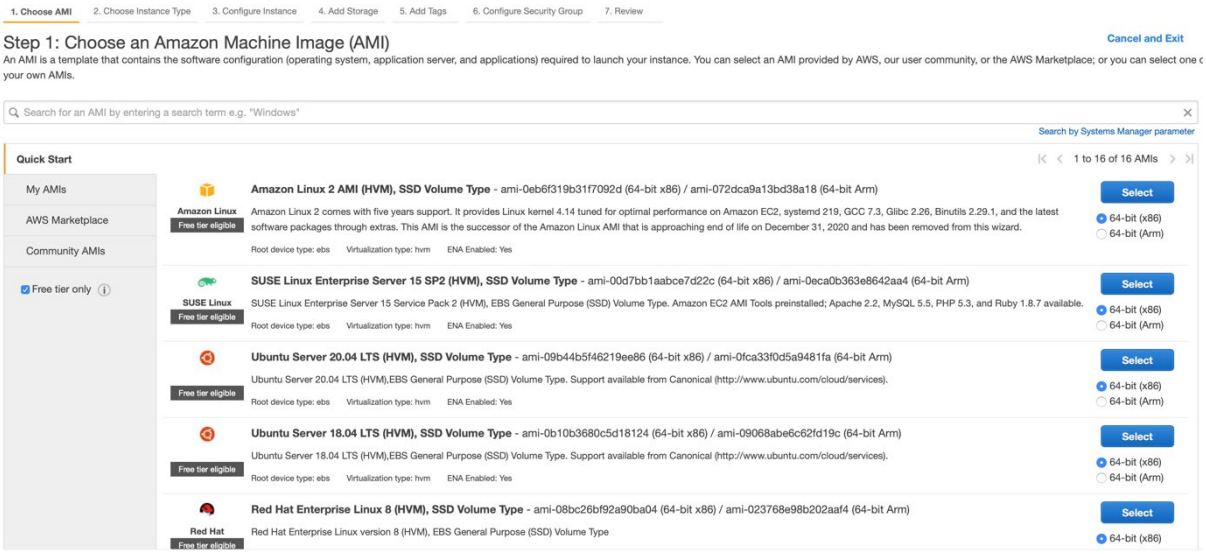
AWS Management Console

З сервісу EC2 маємо можливість запустити новий віртуальний сервер (інстанс) – помаранчева кнопка Launch Instance.



Elastic Compute Service

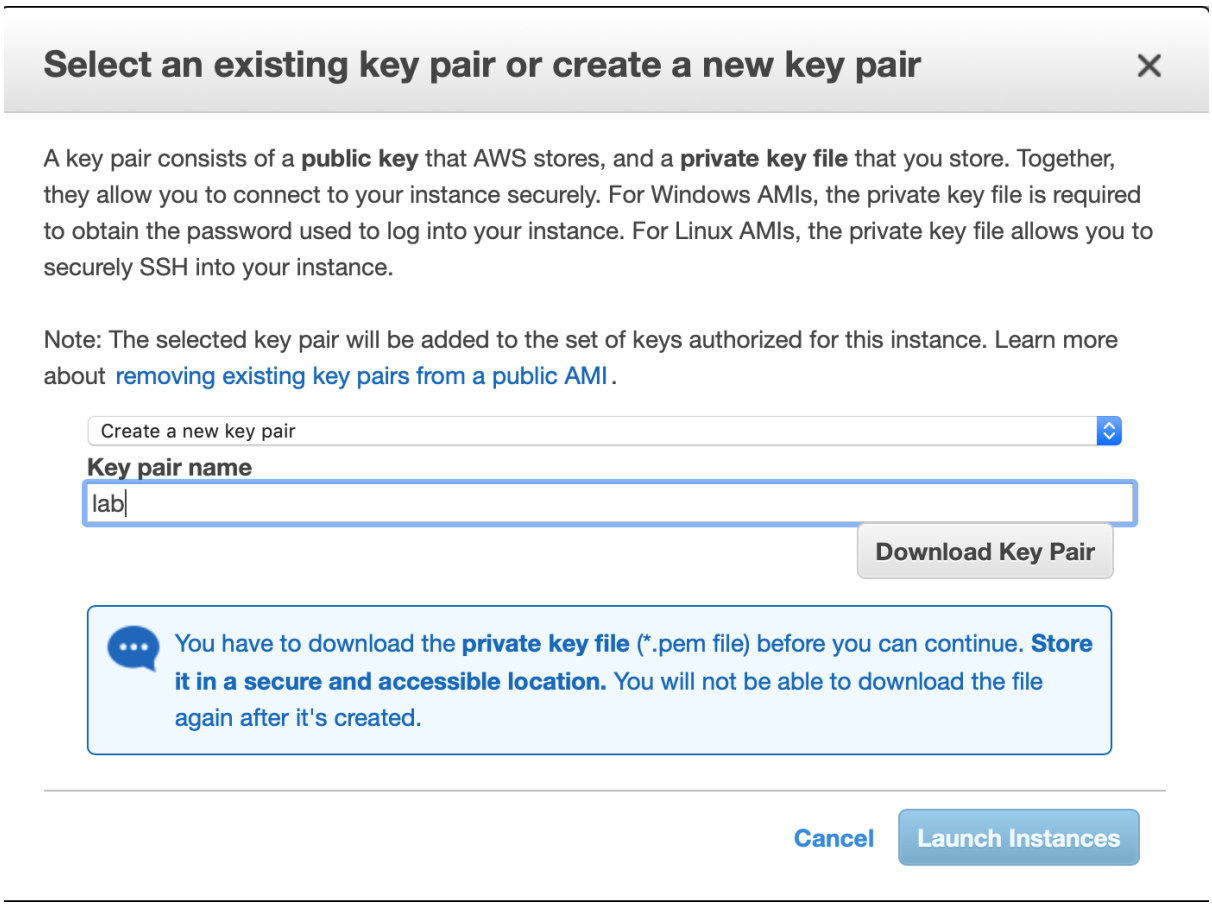
Важливо: при створенні вказуйте опцію Free tier Only щоб уникнути плати за ресурси



Створення інстансу

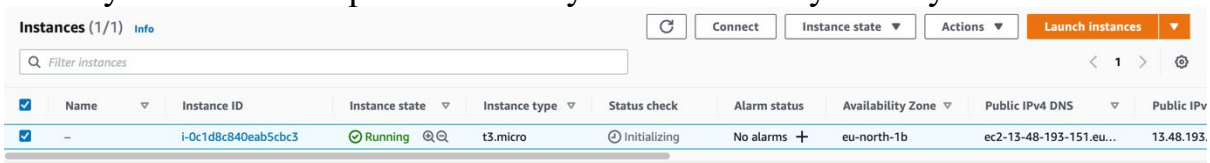
Детально цей процесс описано у розділі 3 книги Andreas Wittig, Michael Wittig “Amazon Web Services in Action”.

Після завершення створення інстансу Вам буде запропоновано створити пару ключей RSA чи додати вже існуючі та зберегти їх.



Створення ключів для віддаленого доступу

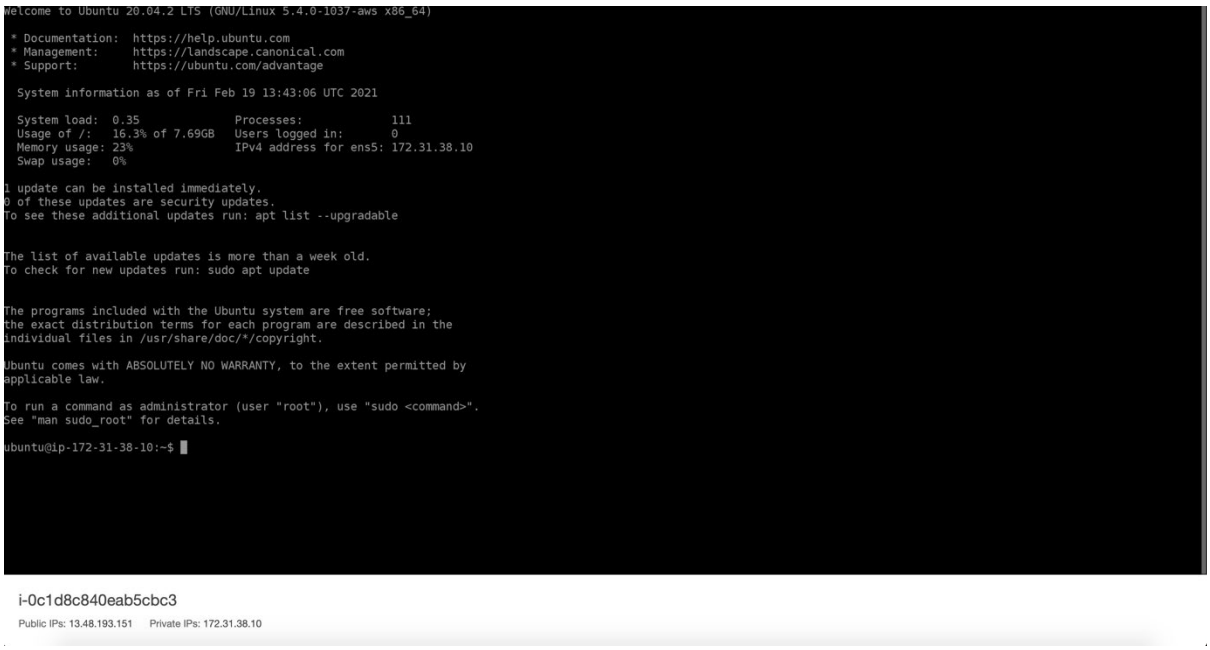
Після успішного створення інстансу він з'явиться у списку активних.



Список інстансів

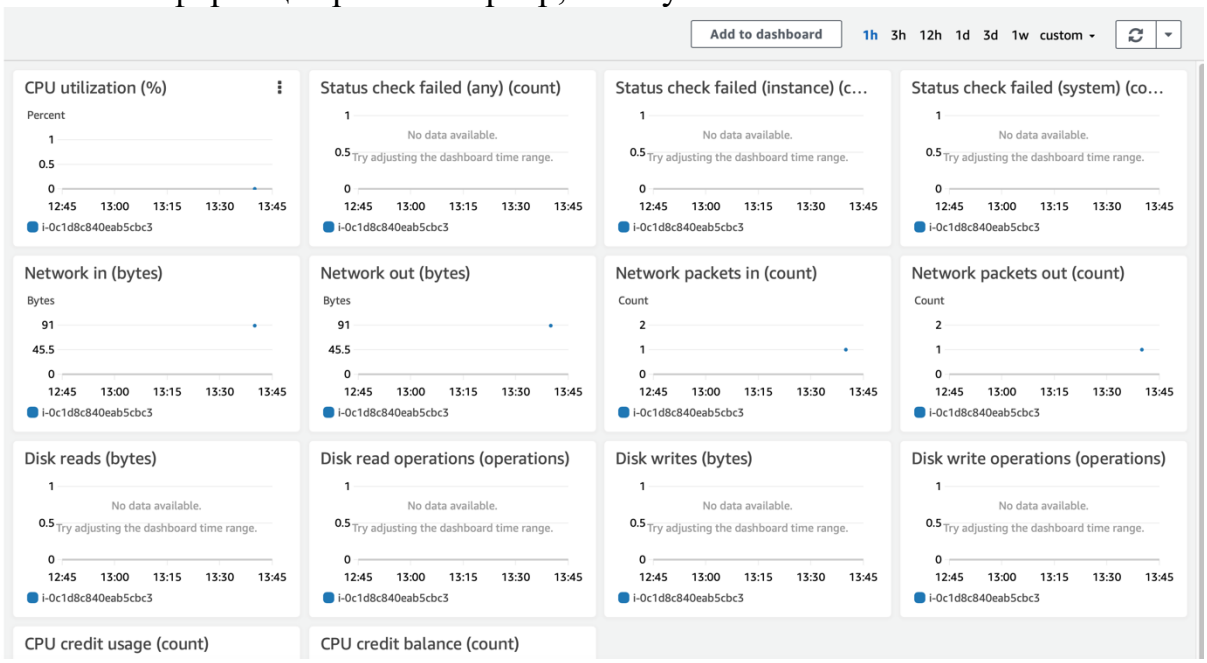
Доступ до інстансу можно отримати прямо з браузера, через ssm manager та SSH.

Спробуйте усі три в ході виконання роботи, в роботі рекомендуємо використовувати доступчерез SSH, як більш зручний.



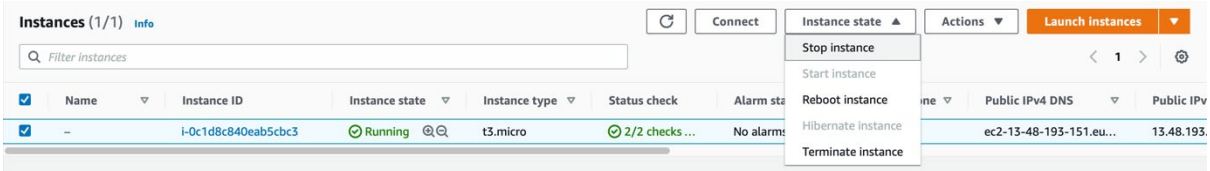
Доступ до bash інстансу з браузеру

Клацнувши на Instance Id у списку активних інстансів можна отримати більше інформаціїпро ваш сервер, в тому числі його навантаження.



Моніторинг навантаження на сервер

По завершенні роботи потрібно вимкнути інстанс – квота безкоштовного використання не є безмежною. Це можна зробити з меню Instance State для списку інстансів.



Вимкнення інстансу

1.3. Завдання

Вивчіть консольні способи моніторингу навантаження на сервер, доступні в Linux (використання диску, пам'яті, процесорного часу – оперативні та на базі логів). Результати самостійного опрацювання потрібно включити в протокол.

1. Зареєструватись в AWS
2. Створити мікро-інстанс
3. Отримати доступ до нього
4. Навчитись моніторити використання ресурсів
5. Навчитися завантажувати файли на інстанс (створити пустий файл *.txt та завантажити його на інстанс через термінал та за допомогою FileZilla)
6. Відкрити файл на інстансі за допомогою редактора Vim (додати текс у файл «Hello world!»)
7. Завантажити із інстанса змінений текстовий файл
8. Результати усіх кроків оформити у вигляді детального протоколу зі скріншотами
9. Навести перелік проблем, вирішення яких було складним в ході виконання роботи врозділі висновків до протоколу

Примітка: без пунктів 5-9 робота зарахована не буде.

1.4. Додаткові джерела інформації

1. https://aws.amazon.com/
2. https://www.youtube.com/watch?v=YB_qanudIzA
3. https://www.youtube.com/watch?v=8bIW7qlldLg

Лабораторна робота №2
Використанням AWS Simple Storage Service (S3)

Мета роботи: ознайомитись з використанням AWS Simple Storage Service (S3).

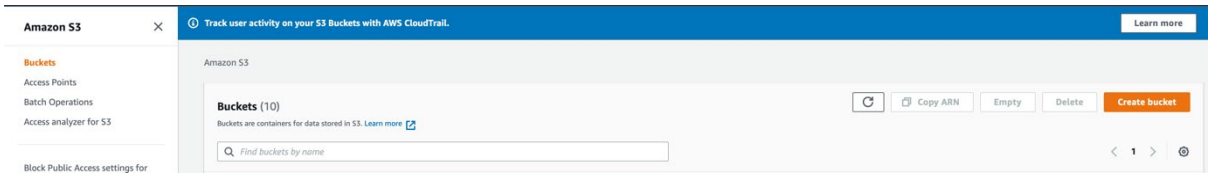
2.1. План виконання

- Створити бакет S3
- Налаштувати доступ до нього з інстансу, створеного у попередній лабораторній роботі
- Ознайомитись зі способами взаємодії з ним

Довідкова інформація: глави 7.1-7.5 книги Andreas Wittig, Michael Wittig "Amazon Web Services in Action".

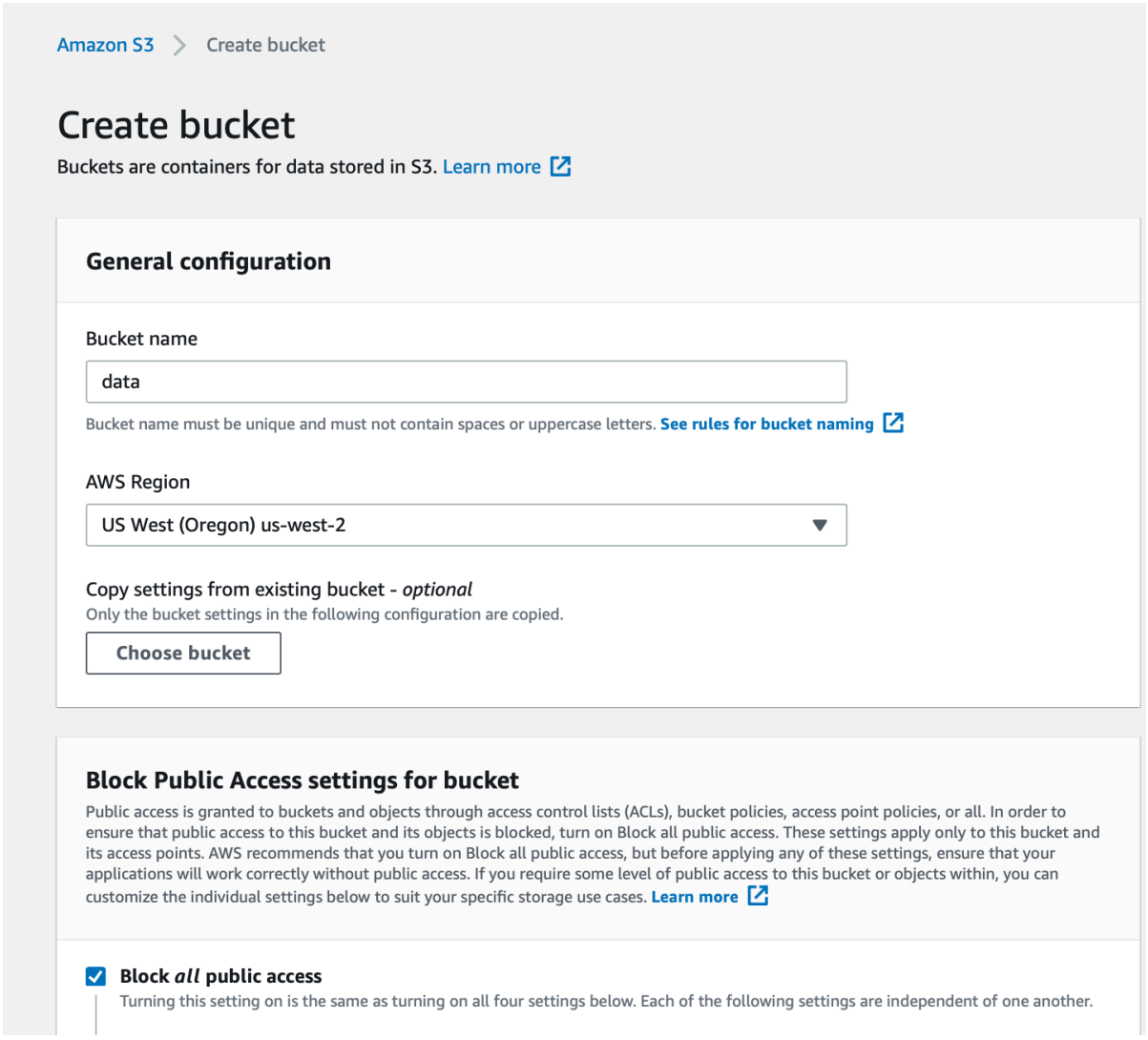
2.2. Порядок виконання роботи

В межах Free Tier для використання доступно 5 Gb сховища S3 (20 000 запитів на читання та 2000 запитів на запис).
Створення бакета S3 можливе різними способами: через AWS Management Console, через AWS CLI тощо.
Так, через сервіс S3 бакет може бути створений через процедуру Create Bucket.



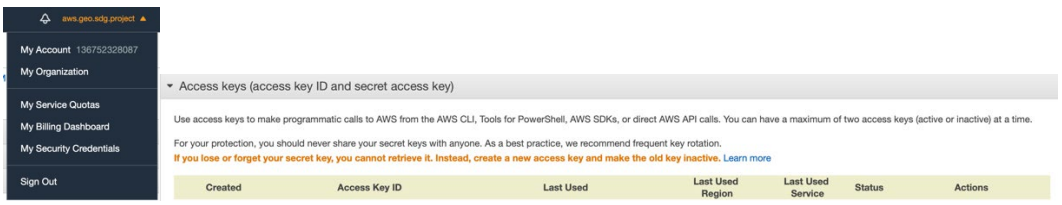
Створення бакету з веб-інтефейсу

Потрібно вказати ім'я бакету та регіон (той самий, в якому було раніше створено інстанс). При створенні варто заборонити публічний доступ до створеного бакета.



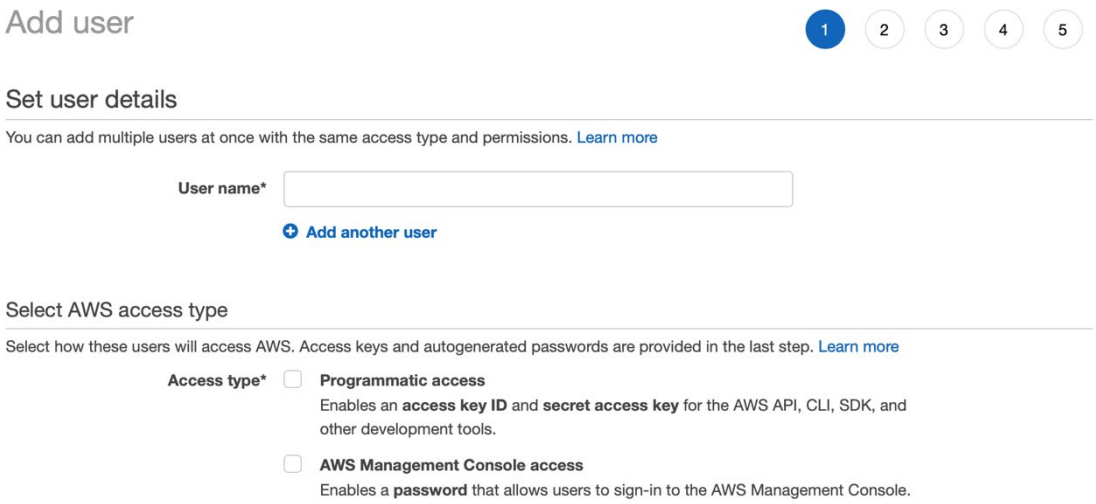
Параметри бакету

Для програматичної роботи з ресурсами AWS (через API) вам знадобиться раніше створений ключ користувача (Access key ID, Secret access key).
Їх можна отримати з профіля користувача (пункт My security Credentials), пункт Access keys (access key ID and secret access key), кнопка Create new Access key.



Створення ключів для програматичного доступу

Бажано створювати окремого користувача (не root). Це можна створити шляхом використання сервісу керування користувачами IAM (розділ Users, кнопка Add user). Важливо при цьому не забути надати користувачу програматичний доступ.



Для подальшої роботи потрібно **встановити** AWS CLI: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
Після встановлення потрібно **налаштувати** клієнт: https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html
aws configure

```
AWS Access Key ID [None]: отримано раніше
AWS Secret Access Key [None]: отримано раніше
Default region name [None]: зона інстансу
Default output format [None]: json
```

Детальніше про використання: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-using.html

Для опрацювання самостійно

- Ознайомитись з основами використання AWS CLI для роботи з S3 (робота з файлами, папками, копіювання даних з інстансу на бакет та навпаки) як через AWS CLI (в консольному режимі), так і у веб-інтерфейсі AWS Management Console. Для довідки - https://docs.aws.amazon.com/cli/latest/reference/s3/
- Вивчити елементи програматичного доступу до даних на S3 з Python API (boto3 бібліотека) - https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-examples.html (приклади для ознайомлення)

Для налаштування Python API потрібно встановити пакетний менеджер pip (враховуючи обмежений дисковий простір на мікро-інстансі не варто встановлювати повну версію Anaconda).

За потреби можна встановити Miniconda https://docs.conda.io/en/latest/miniconda.html) та власне сам Python SDK для AWS
pip install boto3 чи *conda install boto3* (в залежності від обраного способу)

Приклад коду для програматичного вивантаження файлів на s3:
https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-uploading-files.html
Приклад коду для програматичного завантаження файлів з s3:
https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-example-download-file.html

Як запустити jupyter notebook на інстансі та працювати з ним з локального робочого місця

Після розгортання інстансу знадобиться оновлення дерева пакетів (далі приклад для Ubuntu server 20.04 LTS)

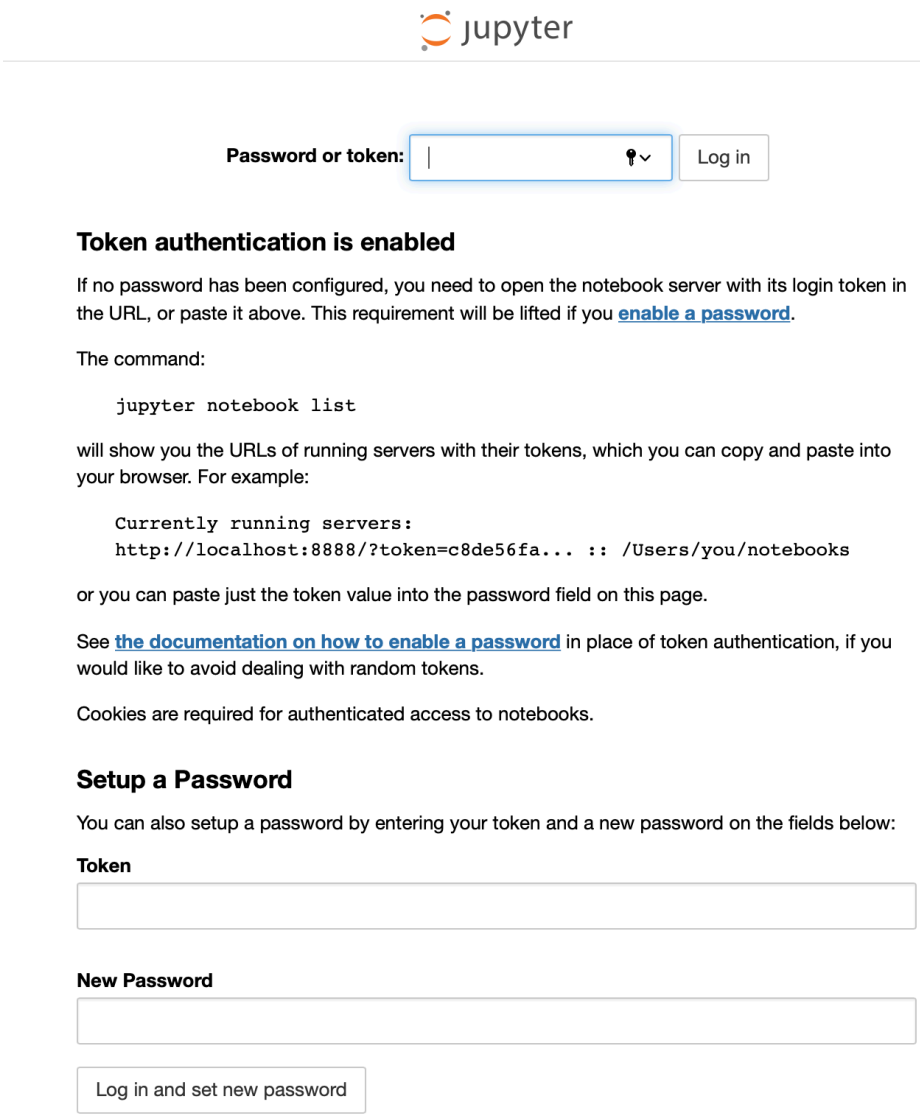
```
sudo apt install python3-pip
sudo pip3 install jupyter notebook
jupyter notebook --no-browser --port 8889
```

Далі для **Windows** завантажуємо plink для створення захищеного тунелю http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

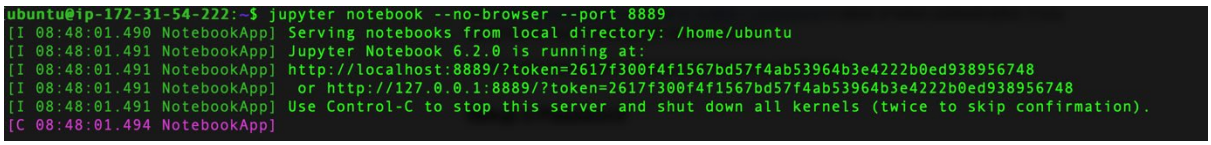
Та створюємо SSH-тунель
`plink.exe -ssh -N -L localhost:8888:localhost:8889 ubuntu@Public IPv4 adress` (його можна отримати у AWS Management Console для активного інстансу)

Для Linux
`ssh -i odc.pem -N -f -L localhost:8888:localhost:8889 ubuntu@54.71.56.14`

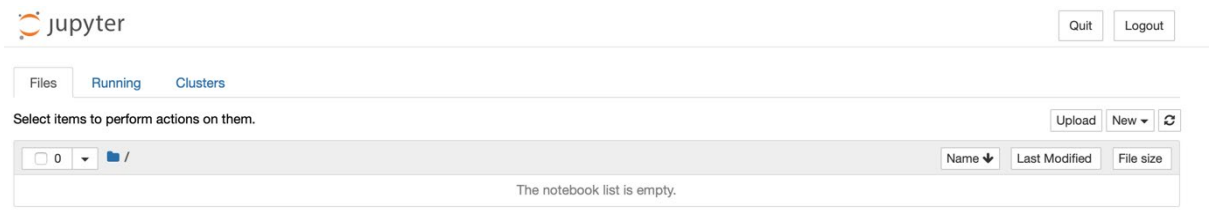
Цим самим ми перекинемо localhost на порту 8888 на віддалене з'єднання.
Вводимо в браузері http://localhost:8888/ та отримаємо запрошення для вводу токена авторизації



Токен є на консолі інстансу, де Ви попередньо запустили jupyter notebook



Вставляємо його в поле для токена і наш хмарний ноутбук готовий для роботи



2.3. Завдання

1. Отримати програматично дані щодо курсу гривні у JSON-форматі на інстанс (https://bank.gov.ua/ua/open-data/api-dev). Альтернативним варіантом є автоматизоване вивантаження даних про якість повітря для України у форматі .csv з джерела SaceEcoBot - https://www.saveecobot.com/maps#12/50.4270/30.5351/aqi/ .
2. Написати скрипт, що створить відповідний csv-файл з даними, конвертуючи отриманий json-файл з пункту 1. У випадку вибору даних SaceEcoBot цей пункт пропускаємо.
3. Створені csv-файли мають програматично вивантажуватись на S3
4. Розробити скрипт для читання файлів з бакету та візуалізації курсу валют (усереднених щоденних даних про якість повітря) засобами Python (наприклад у jupyter notebook - https://jupyter.org, ядро якого працюватиме на інстансі, а сам він буде працювати у браузері на вашому комп'ютері)
5. Побудувати графік із курсом гривні щодо іноземних валют (Долар США та Євро) для 2022 року. З даними SaceEcoBot – графік залежності якості повітря від часу доби.
6. Зберегти побудований графік на бакет та додати його до звіту
7. Результати усіх кроків оформити у вигляді детального протоколу зі скріншотами та командами в консолі які використовувалися
8. Навести перелік проблем, вирішення яких було складним в ході виконання роботи в розділі висновків до протоколу

Примітка: без пунктів 7-8 робота зарахована не буде.

2.4. Додаткові джерела інформації

1. Andreas Wittig, Michael Wittig "Amazon WebServices in Action"
2. https://www.youtube.com/watch?v=i4YFFWcyeFM

Лабораторна робота №3
Використання AWS DynamoDB

Мета роботи: ознайомитись з використанням AWS DynamoDB (serverless database).

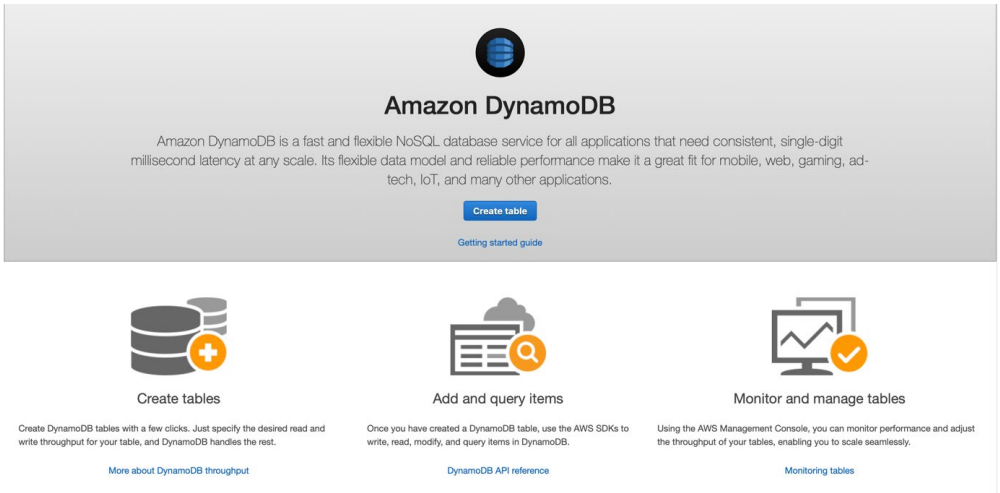
3.1. План виконання

- Спроектувати структуру даних (таблицю).
- Вивчити способи роботи з даними засобами DynamoDB.
- Виконати завдання відповідно до варіанту (в GUI, засобами AWS CLI та Python – останнє за бажанням).

Довідкова інформація: глава 10 книги Andreas Wittig, Michael Wittig "Amazon Web Services in Action" (Programming for the NoSQL database service: DynamoDB).

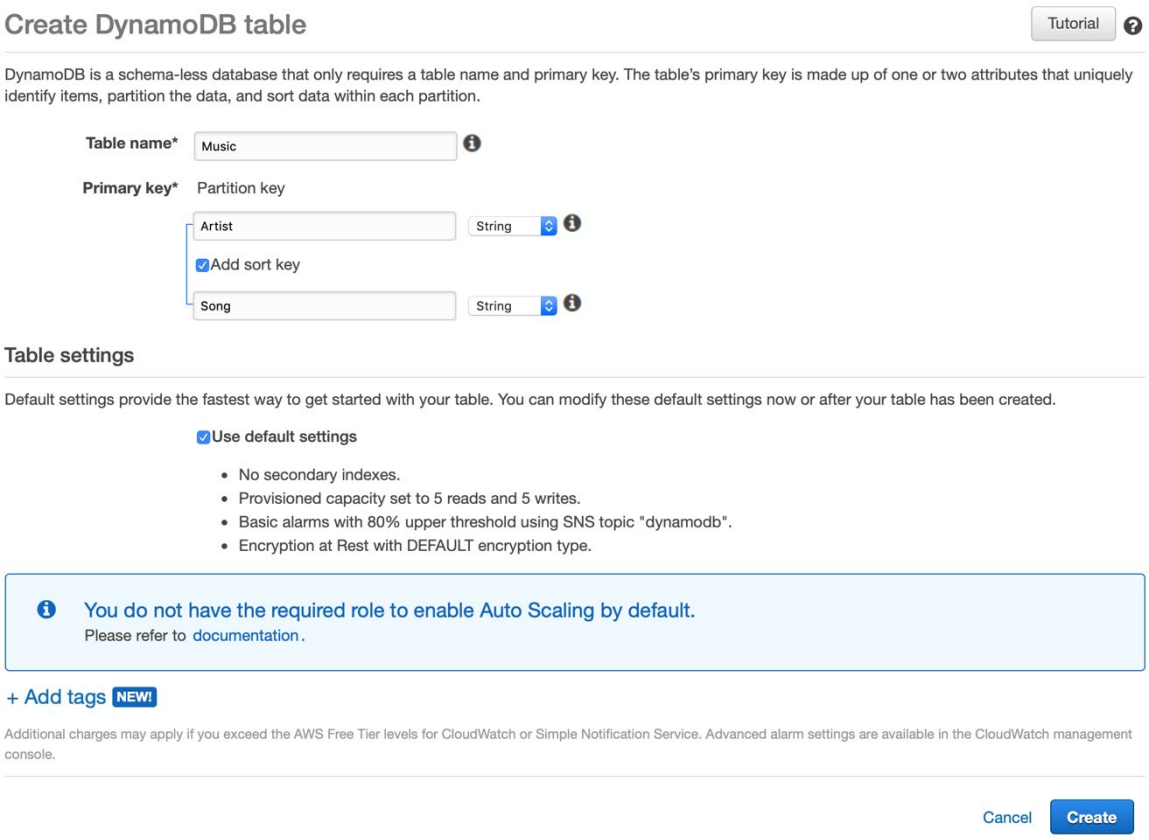
3.2. Порядок виконання роботи

Створення таблиці DynamoDb здійснюється траційно як через AWS CLI, так і засобами AWS Management Console.
Для створення таблиці засобами AWS Management Console оберіть сервіс DynamoDB (доступний у розділі Databases) та натисніть кнопку «Create Table»



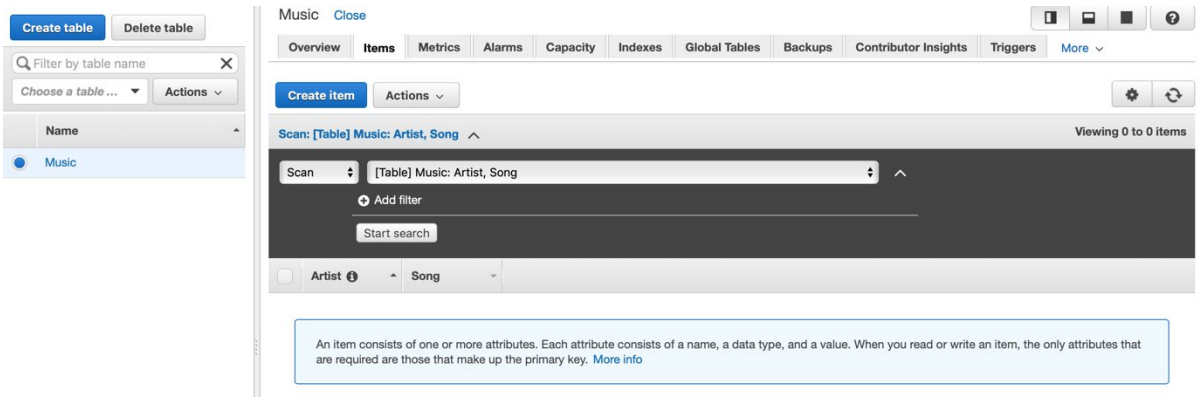
Сторінка запрошення сервісу DynamoDB

Створимо таблицю Music, додавши до неї в якості ключа текстове поле Artist та поле Song, за яким можна буде здійснювати пошук.



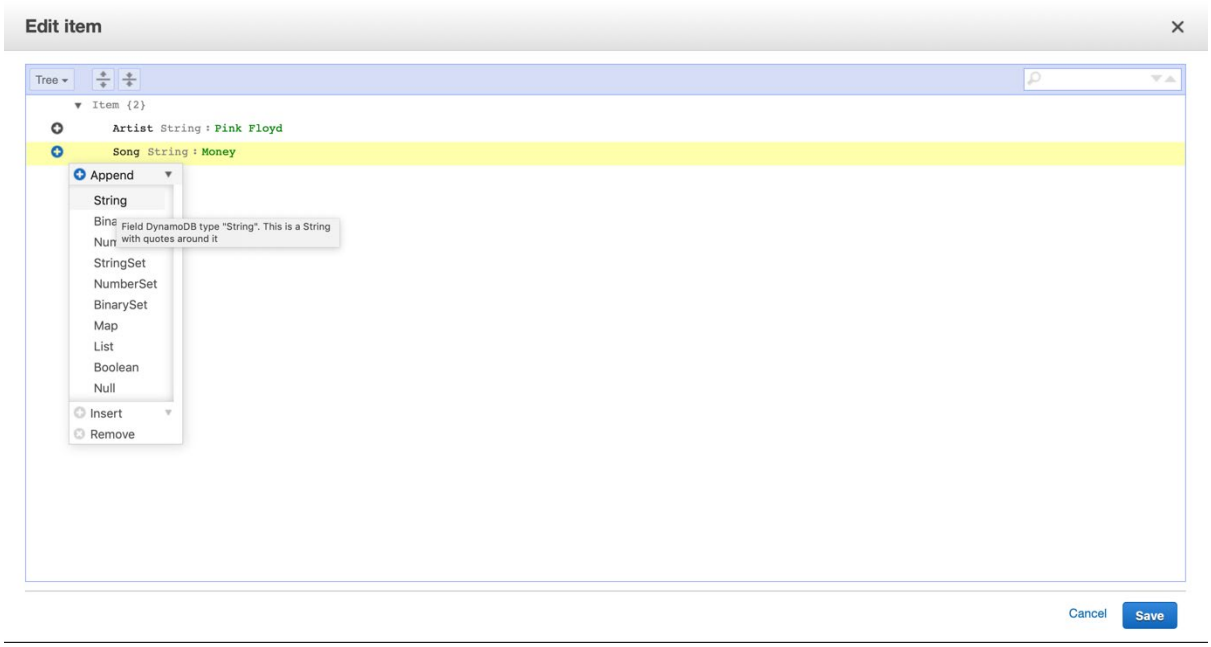
Створення таблиці DynamoDB

Створимо кілька записів для цієї таблиці, кожен запис (Item) містить кілька атрибутів (це можна зробити в розділі Items).



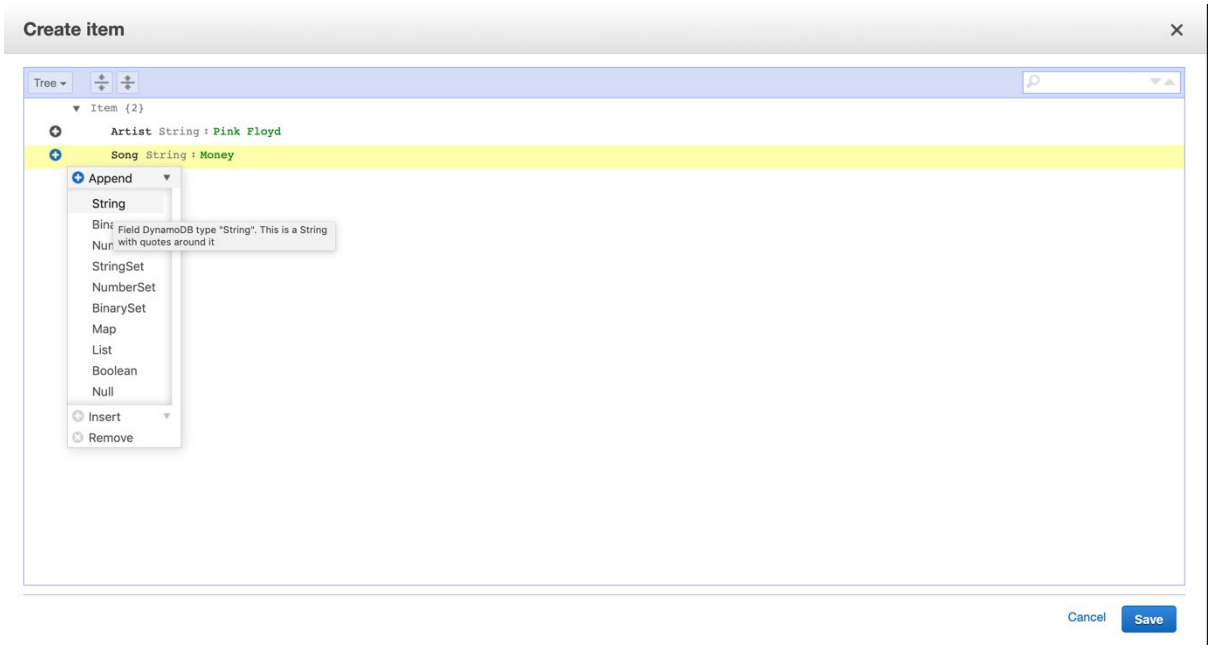
Внесення записів до таблиці DynamoDB

Внесемо основні частини запису через Create Item

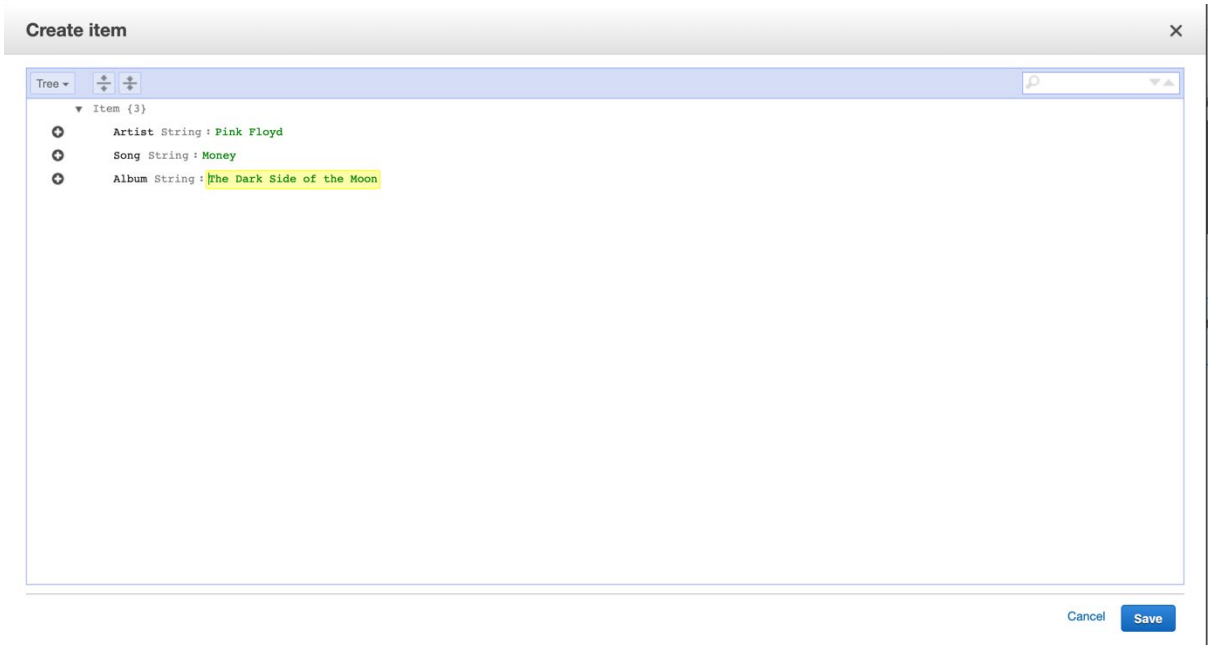


Конструювання структури запису

Та додамо додаткове текстове поле Album



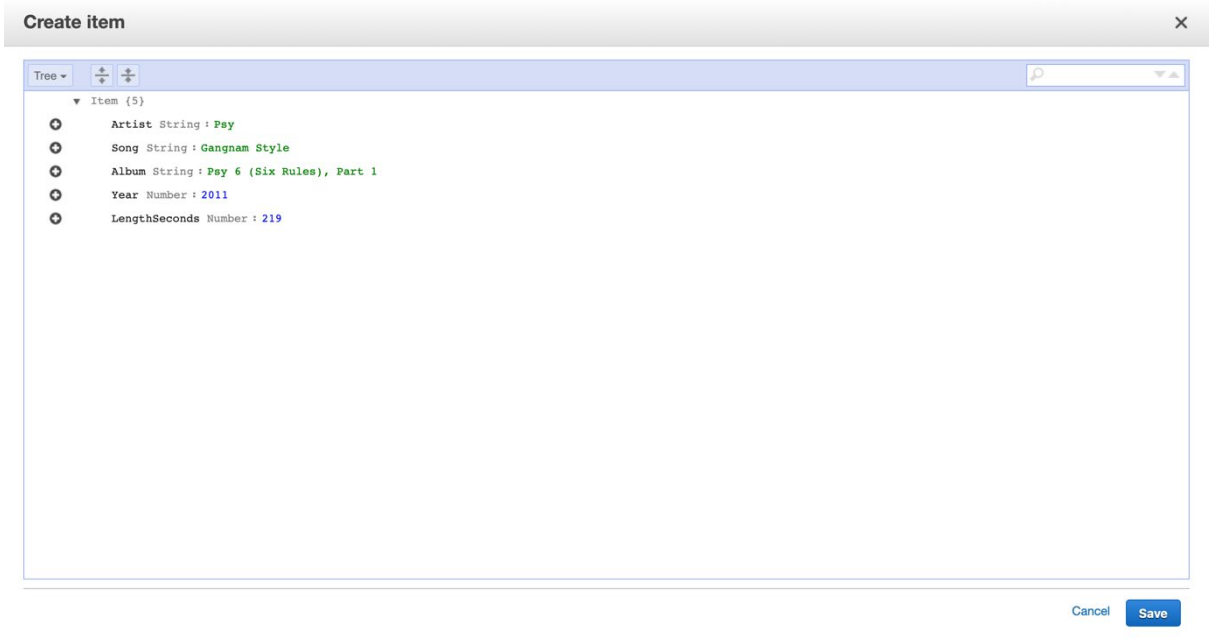
В результаті отримаємо Item такого вигляду



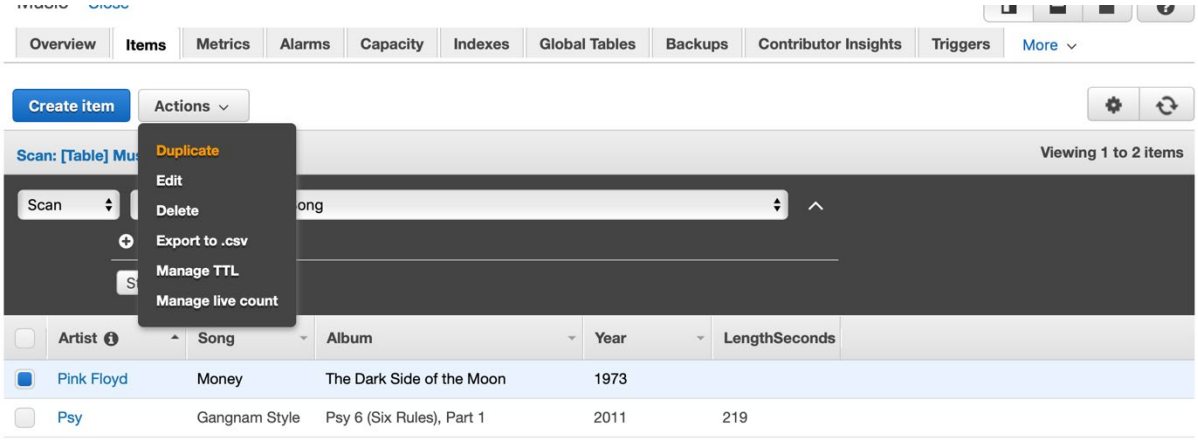
Кількість атрибутів може бути розширена досить гнучко – вона може бути іншою в сенсі додаткових полів – фактично є повна аналогія з представленням даних в JSON.

Тепер створимо новий об’єкт для альбому

Attribute Name	Attribute Type	Attribute Value
Artist	String	Psy
Song	String	Gangnam Style
Album	String	Psy 6 (Six Rules), Part 1
Year	Number	2011
LengthSeconds	Number	219

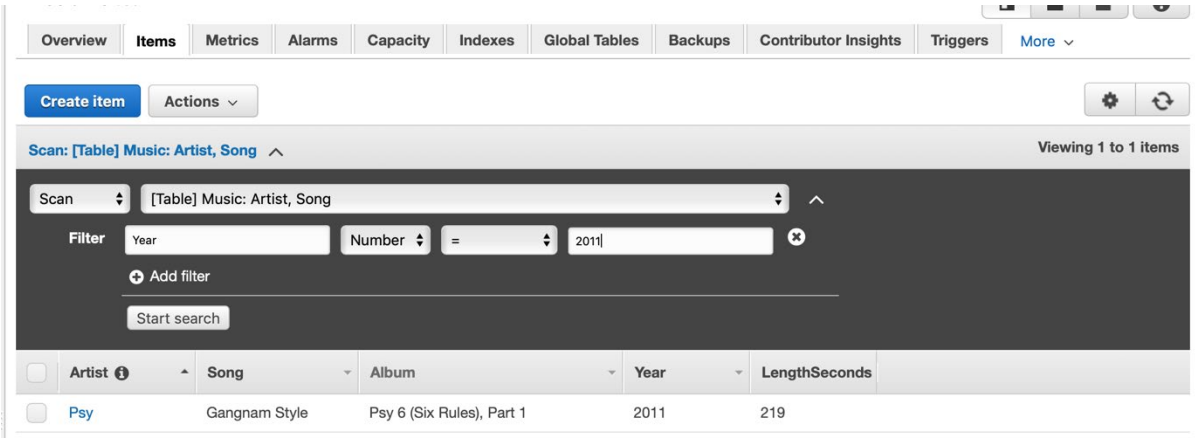


Об’єкти-записи можна змінювати, дублювати, видаляти, експортувати тощо.



Сторінка запрошення сервісу DynamoDB

Працюючи з фільтрами можна здійснювати пошук даних



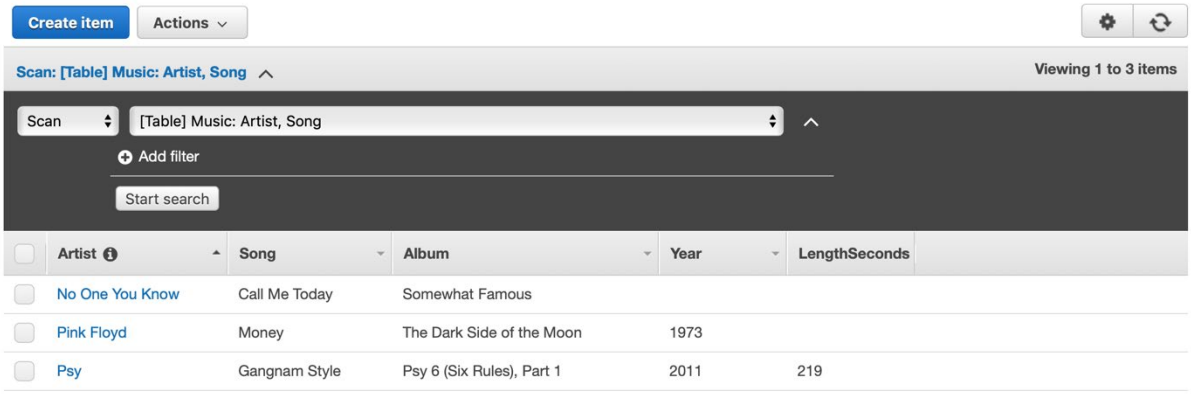
Пошук даних засобами DynamoDB

Значну частину операцій з dynamodb можна зробити і із використанням AWS CLI (https://docs.aws.amazon.com/cli/latest/reference/dynamodb/index.html#cli-aws-dynamodb).

```
aws dynamodb create-table --table-name Music --
attribute-definitions
AttributeName=Artist,AttributeType=S
AttributeName=SongTitle,AttributeType=S  --key-schema
AttributeName=Artist,KeyType=HASH
AttributeName=SongTitle,KeyType=RANGE --provisioned-
throughput ReadCapacityUnits=10,WriteCapacityUnits=5
```

Так для додавання запису можна використати метод put-item, передавши йому відповідний JSON

```
aws dynamodb put-item --table-name Music --item
'{"Artist": {"S": "No One You Know"}, "Song": {"S":
"Call Me Today"},"Album": {"S": "Somewhat Famous"} }' -
-return-consumed-capacity TOTAL
```



Поточний вміст таблиці DynamoDB

Пошук із збереженням параметрів у json-файлі

```
aws dynamodb query --table-name Music --key-condition-
expression "Artist = :v1" --expression-attribute-values
file://~/conf.json
{
    "Items": [
        {
            "Song": {
                "S": "Gangnam Style"
            },
            "Album": {
                "S": "Psy 6 (Six Rules), Part 1"
            },
            "LengthSeconds": {
                "N": "219"
            },
            "Artist": {
                "S": "Psy"
            },
            "Year": {
```

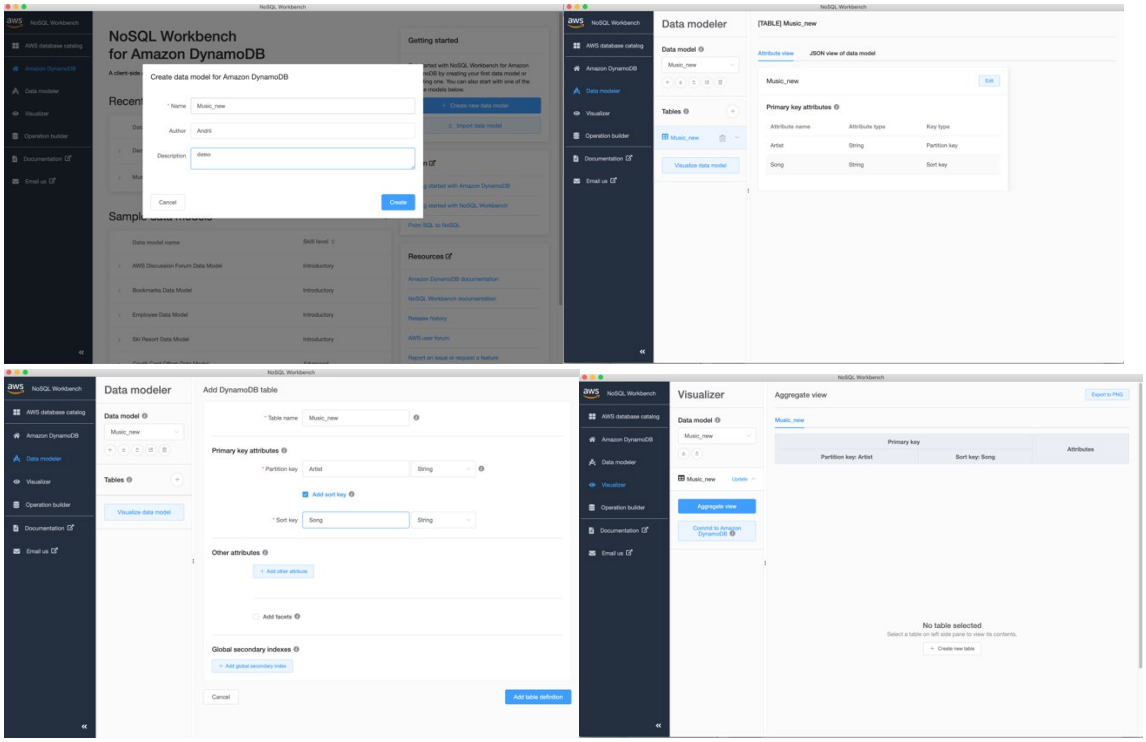
```
                "N": "2011"
            }
        }
    ],
    "Count": 1,
    "ScannedCount": 1,
    "ConsumedCapacity": null
}
```

Де у файлі, що зберігається у домашній директорії, надані параметри пошуку по ключу Artist (таблиця створена з цим ключовим полем)

```
cat conf.json
{
  ":v1": {"S": "Psy"}
}
```

Додаткові реперні інструкції та зразки можна знайти за посиланням https://docs.aws.amazon.com/cli/latest/userguide/cli-services-dynamodb.html.

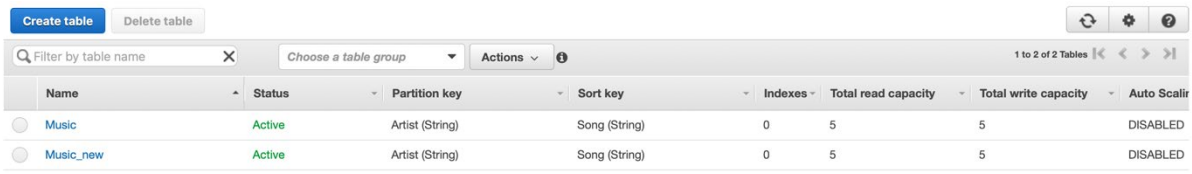
Ще одним зручним засобом роботи з DynamoDB є **NOSQL Workbench** - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/workbench.html. В ньому можна спроектувати структуру таблиці та перенести на AWS.



Операції у NOSQL Workbench

Для перенесення до AWS знадобиться зробити налаштування та зробити commit (знадобляться aws_access_key_id та aws_secret_access_key, отримані у попередній роботі)

Після успішного комміти таблиця буде додана до переліку таблиць DynamoDb



3.3. Завдання

1. Для даних, отриманих в роботі №2 спроектувати таблицю DynamoDB (засобами NOSQL Workbench, AWS CLI та AWS Management Console).
2. Відпрацювати додавання, видалення та пошук даних засобами AWS Management Console та AWS CLI.
3. За додаткові бали імплементувати ці операції засобами Python SDK (приклади для boto3 доступні за посиланням https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.html).

3.4. Додаткові джерела інформації

1. https://docs.aws.amazon.com/cli/latest/userguide/cli-services-dynamodb.html.
2. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/workbench.html.

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

Мета роботи: ознайомитись з основами керування ресурсами AWS засобами Python SDK.

Для виконання роботи необхідно наступне:

- Встановити Python3.
- Встановити бібліотеку Boto3 (pip install boto3).
- Мати вже створений обліковий запис на AWS.

4.1. Порядок виконання роботи

Автоматизація роботи з обчислювальними ресурсами EC

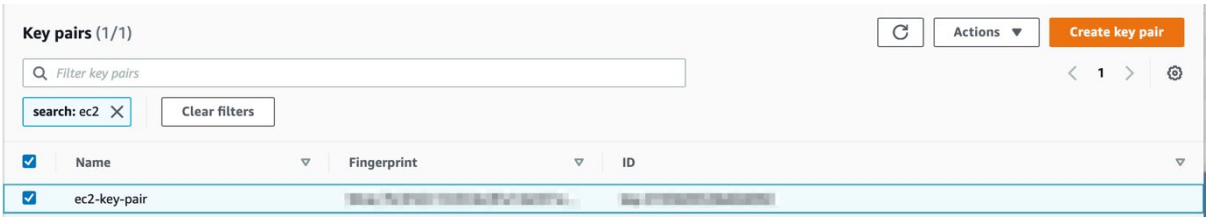
Крок 1 – створення ключової пари для доступу до EC2-інстансу

Пара ключів знадобиться для доступу через ssh-протокол до EC2-інстансу, який буде створено

```
import boto3
import os
def create_key_pair():
    ec2_client = boto3.client("ec2", region_name="us-west-2")
    key_pair    =    ec2_client.create_key_pair(KeyName="ec2-key-
pair")
    private_key = key_pair["KeyMaterial"]
    with os.fdopen(os.open("/tmp/aws_ec2_key.pem", os.O_WRONLY |
os.O_CREAT, 0o400), "w+") as handle:
        handle.write(private_key)
create_key_pair()
```

Функція create_key_pair() створює ключову пару ec2-key-pair з правами лише на читання (400 або r-- --- --- в стандартній Unix нотації) для ключа, що знадобиться для подальшого підключення до інстансу у файлі /tmp/aws_ec2_key.pem

Створений ключ буде зареєстровано в межах користувацького облікового запису AWS

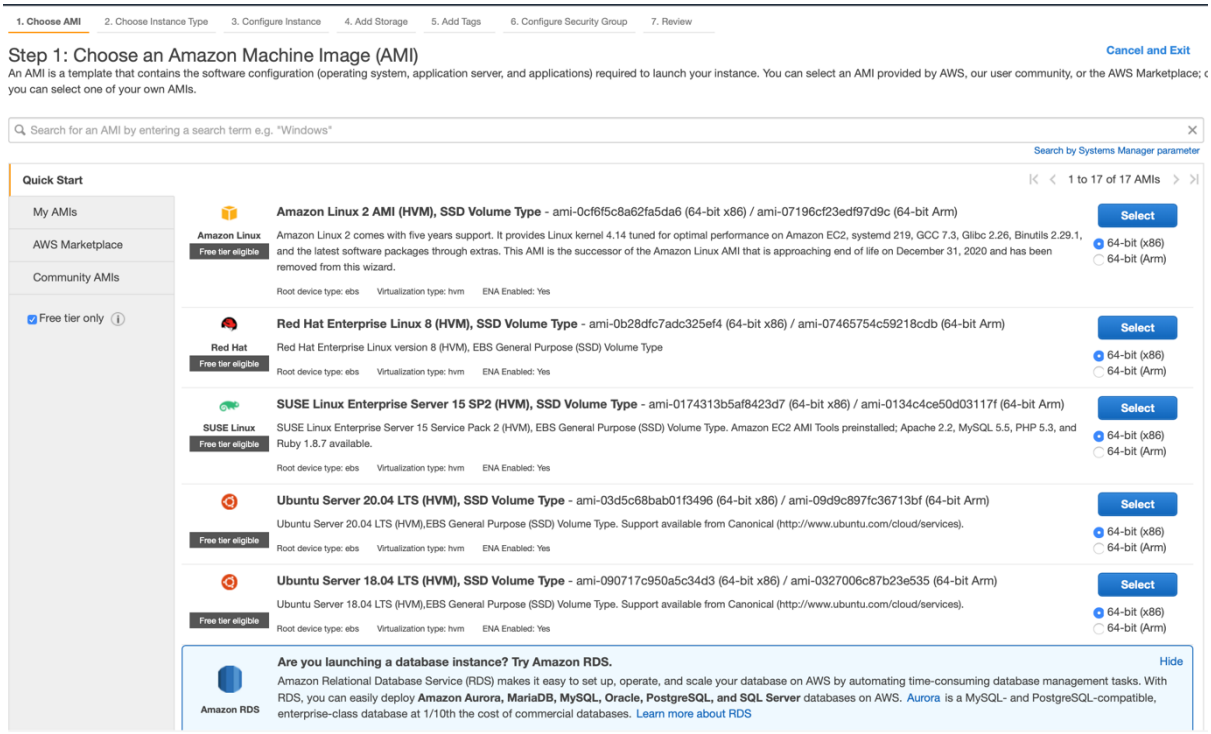


Крок 2 – створення EC2 інстансу засобами boto3

Для створення інстансу потрібно визначитися з образом операційної системи (Amazon Machine Image - AMI), який буде використовуватись при розгортанні (знати конкретний ImageID).

Можна використати як створені самостійно образи, так і взяти вже готові.

Переліг готових AMI можна отримати на початковому кроці розгортання системи (не забуваємо про Free tier only опцію).



В прикладі використаємо Amazon Linux 2 AMI (HVM), SSD Volume Type для архітектури 64-bit x86 з ідентифікатором **ami-0b0154d3d8011b0cd**.

Параметри функції:

- ***MinCount***: мінімальне число EC2 інстансів, що створюється
- ***MaxCount***: максимальне число EC2 інстансів, що створюється

- ***InstanceType***: тип інстансу EC2 (у прикладі - t4g.nano). Повний їх перелік - https://aws.amazon.com/ec2/instance-types/
- ***KeyName***: ім'я ключової пари, яка буде використовуватись для ssh-доступу до інстансу (якщо її не вказати, то доступу не буде)

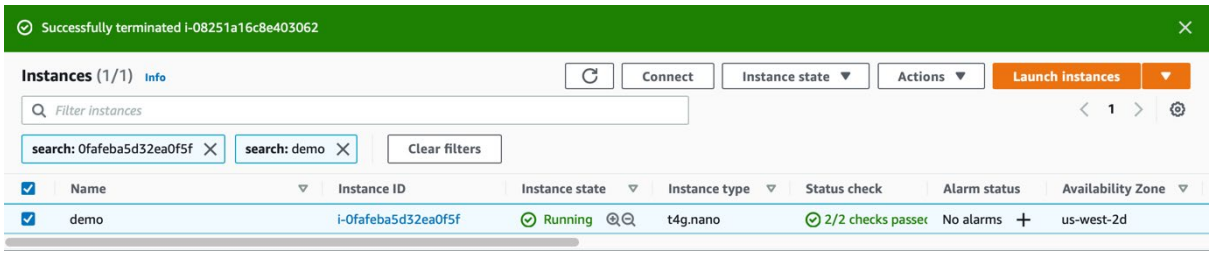
```
def create_instance():
    ec2_client = boto3.client("ec2", region_name="us-west-2")
    instances = ec2_client.run_instances(
        ImageId="ami-0b0154d3d8011b0cd",
        MinCount=1,
        MaxCount=1,
        InstanceType="t4g.nano",
        KeyName="ec2-key-pair"
    )
    print(instances["Instances"][0]["InstanceId"])
create_instance()
```

Результат виклику функції create_instance():

```
    i-0ce72b201a0580783
```

Цей ідентифікатор унікальним чином вказує на інстанс.

В результаті буде створено інстанс з певним id, який відобразиться у AWS Management console. Для зручності дамо йому ім'я demo.



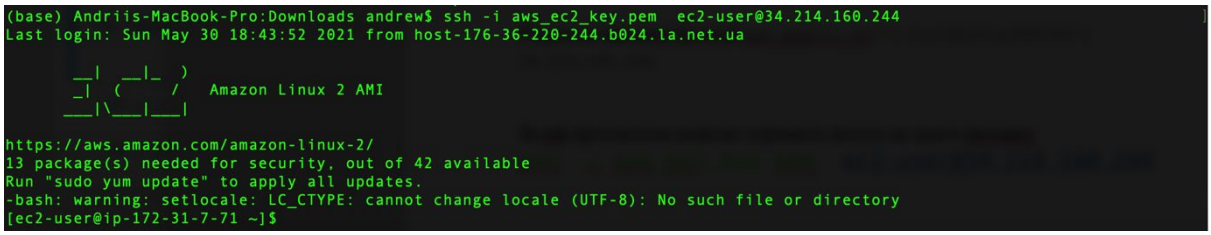
Для отримання ip-адреси створеного інстансу можна написати наступний код

```
def get_public_ip(instance_id):
    ec2_client = boto3.client("ec2", region_name="us-west-2")
    reservations                                                =
ec2_client.describe_instances(InstanceIds=[instance_id]).get("Re
servations")

    for reservation in reservations:
        for instance in reservation['Instances']:
            print(instance.get("PublicIpAddress"))
```

Результат виклику функції get_public_ip('i-0ce72b201a0580783')

```
34.214.160.244
```

За ssh-протоколом можемо отримати доступ до цього інстансу ssh -i aws_ec2_key.pem ec2-user@34.214.160.244



Крок 3 – автоматизація моніторингу активних інстансів

Зберемо інформація про всі інстанси, що працюють на мають тип t4.nano.

Використаємо метод describe_instances та передамо йому в якості фільтру необхідні параметри (стан та тип інстансу).

```
def get_running_instances():
    ec2_client = boto3.client("ec2", region_name="us-west-2")
    reservations = ec2_client.describe_instances(Filters=[
        {
            "Name": "instance-state-name",
            "Values": ["running"],
        },
        {
            "Name": "instance-type",
            "Values": ["t4g.nano"]
        }
    ]).get("Reservations")

    for reservation in reservations:
        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}")

get_running_instances()
```

Вихідний результат:

```
i-0ce72b201a0580783, t4g.nano, 34.214.160.244, 172.31.7.71
```

Крок 4 – зупинка інстансу

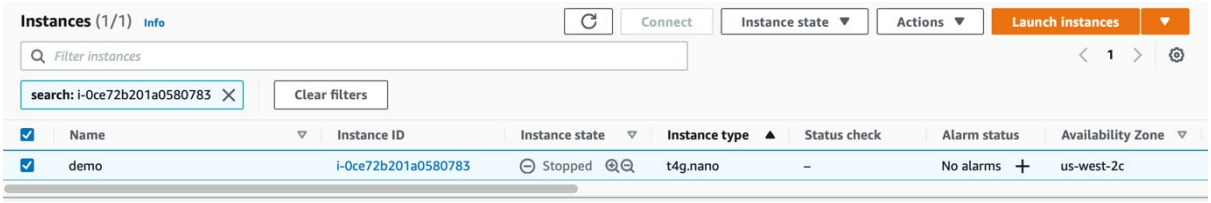
Використаємо метод stop_instances та передамо йому в якості параметру ідентифікатор інстансу

```
def stop_instance(instance_id):
    ec2_client = boto3.client("ec2", region_name="us-west-2")
    response =
ec2_client.stop_instances(InstanceIds=[instance_id])
    print(response)
stop_instance('i-0ce72b201a0580783')
```

Результат

```
{'StoppingInstances': [{'CurrentState': {'Code': 64,
'Name': 'stopping'}, 'InstanceId': 'i-
0ce72b201a0580783', 'PreviousState': {'Code': 16,
'Name': 'running'}}], 'ResponseMetadata': {'RequestId':
'28db8f2d-c87b-44f7-8aae-ad5c6dc949c9',
'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-
requestid': '28db8f2d-c87b-44f7-8aae-ad5c6dc949c9',
'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, 30 May 2021 19:02:32 GMT', 'server':
'AmazonEC2'}, 'RetryAttempts': 0}}
```

В AWS Management Console можемо для перевірки отримати такий результат



Крок 5 – видалення (термінація) непотрібного інстансу

Використаємо метод terminate_instances та передамо йому в якості параметру ідентифікатор інстансу

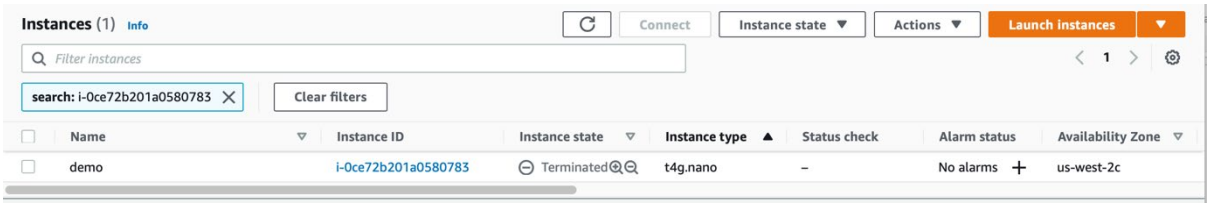
```
def terminate_instance(instance_id):
    ec2_client = boto3.client("ec2", region_name="us-west-2")
    response =
ec2_client.terminate_instances(InstanceIds=[instance_id])
```

```
    print(response)
terminate_instance('i-0ce72b201a0580783')
```

Результат

```
    {'TerminatingInstances': [{'CurrentState': {'Code': 48,
'Name': 'terminated'}, 'InstanceId': 'i-0ce72b201a0580783',
'PreviousState': {'Code': 80, 'Name': 'stopped'}}],
'ResponseMetadata': {'RequestId': '5ec6779d-9247-4a0c-9114-
1a4d8054dcd6', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amzn-
requestid': '5ec6779d-9247-4a0c-9114-1a4d8054dcd6', 'cache-
control': 'no-cache, no-store', 'strict-transport-security':
'max-age=31536000; includeSubDomains', 'content-type':
'text/xml;charset=UTF-8', 'transfer-encoding': 'chunked',
'vary': 'accept-encoding', 'date': 'Sun, 30 May 2021 19:06:53
GMT', 'server': 'AmazonEC2'}, 'RetryAttempts': 0}}
```

В AWS Management Console можемо для перевірки отримати такий результат – невдовзі інстанс перестане відображатись у списку.



Автоматизація роботи з сховищем S3

Крок 1 – створення бакета S3

Для створення бакету S3 використаємо метод create_bucket(), передавши йому назву бакету на регіон

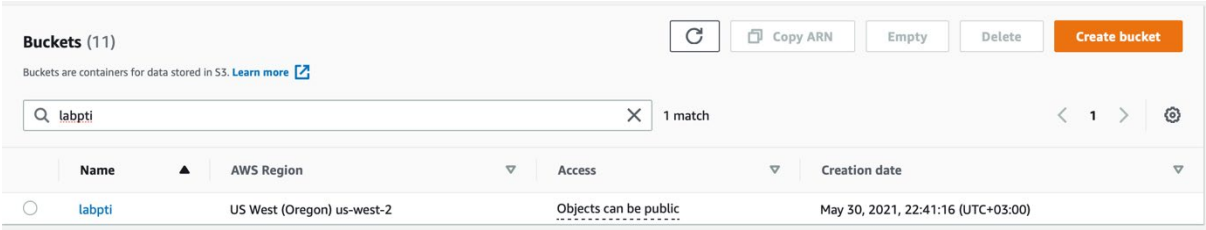
Важливо: простір імен бакетів є глобальним, а тому створити бакет з ім'ям, яке вже існує не вдасться.

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

create_bucket("labpti", "us-west-2")
```

Результат:

```
{'ResponseMetadata': {'RequestId': 'XZ2Y8FH4F204R1K5', 'HostId':
'uBXjA63Kh614Y/XHo3XU0SqwDEo3JgVZcDCb+OL+7T8oh4nKpicJ0rWNJQWP8Xe
l9eHXGIc7TsM=', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amz-
id-2':
'uBXjA63Kh614Y/XHo3XU0SqwDEo3JgVZcDCb+OL+7T8oh4nKpicJ0rWNJQWP8Xe
l9eHXGIc7TsM=', 'x-amz-request-id': 'XZ2Y8FH4F204R1K5', 'date':
'Sun, 30 May 2021 19:46:08 GMT', 'location':
'http://labpti.s3.amazonaws.com/', 'content-length': '0',
'server': 'AmazonS3'}, 'RetryAttempts': 0}, 'Location':
'http://labpti.s3.amazonaws.com/'}
```

В AWS Management Console можемо для перевірки отримати такий результат



Крок 2 – лістинг існуючих бакетів облікового запису

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

Результат:

```
Existing buckets:
  Labpti
```

Крок 3 – завантаження файлу

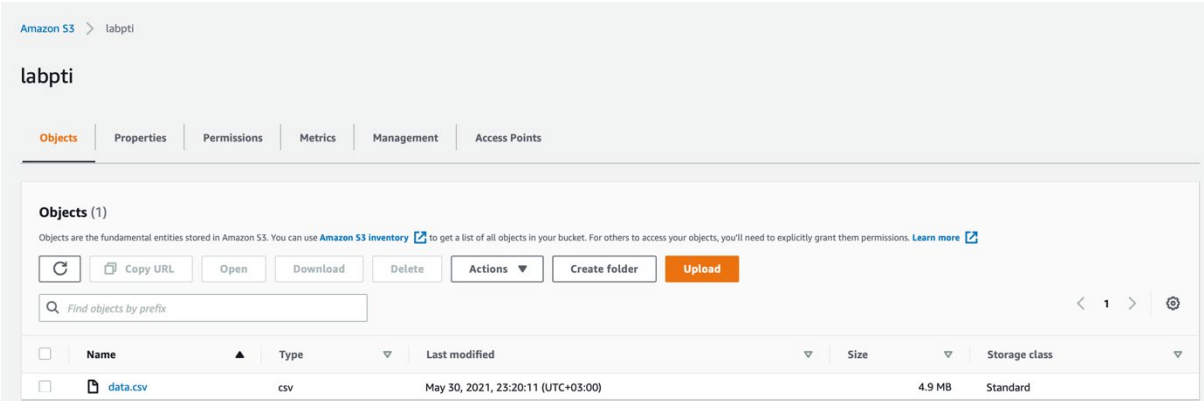
При створенні об’єкту на S3 його ключ (ідектифікатор) може бути іншим, ніж його ім’я в локальній файловій системі ПК. В даному випадку файл data.csv буде створено в корені бакету labpti

```
    def upload(file_name, bucket_name, s3_obj_name):
        s3_client = boto3.client('s3')
        responce    =    s3_client.upload_file(Filename=file_name,
Bucket=bucket_name, Key=s3_obj_name)
        print(response)
```

```
    upload('/Users/andrew/Downloads/annual-enterprise-survey-
2019-financial-year-provisional-csv.csv', 'labpti', 'data.csv')
```

Результат:


```
{'ResponseMetadata': {'RequestId': 'QAPTP6NWBJA5XW7S', 'HostId':
'HyR4ALFGO46TYEBCbBA2vOqN6bOXAr3z/piVRTsmod1AYvu2ThwWTss37LmVr5m
jTRlry0J99hQ=', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amz-
id-2':
'HyR4ALFGO46TYEBCbBA2vOqN6bOXAr3z/piVRTsmod1AYvu2ThwWTss37LmVr5m
jTRlry0J99hQ=', 'x-amz-request-id': 'QAPTP6NWBJA5XW7S', 'date':
'Sun, 30 May 2021 19:50:15 GMT', 'content-type':
'application/xml', 'transfer-encoding': 'chunked', 'server':
'AmazonS3'}, 'RetryAttempts': 0}, 'Buckets': [{'Name': 'aws-emr-
resources-136752328087-us-east-1', 'CreationDate':
datetime.datetime(2020, 2, 12, 15, 10, 44, tzinfo=tzutc())},
{'Name': 'aws-emr-resources-136752328087-us-west-2',
'CreationDate': datetime.datetime(2020, 2, 7, 15, 52, 6,
tzinfo=tzutc())}, {'Name': 'aws-logs-136752328087-us-east-1',
'CreationDate': datetime.datetime(2020, 2, 7, 12, 19, 26,
tzinfo=tzutc())}, {'Name': 'aws-logs-136752328087-us-west-2',
'CreationDate': datetime.datetime(2020, 2, 7, 15, 25, 51,
tzinfo=tzutc())}, {'Name': 'fire-project', 'CreationDate':
datetime.datetime(2020, 10, 26, 11, 3, 53, tzinfo=tzutc())},
{'Name': 'ikiexportdata', 'CreationDate':
datetime.datetime(2019, 12, 26, 16, 15, 51, tzinfo=tzutc())},
{'Name': 'labpti', 'CreationDate': datetime.datetime(2021, 5,
30, 19, 46, 9, tzinfo=tzutc())}, {'Name': 'nfdulanddegradation',
'CreationDate': datetime.datetime(2020, 12, 14, 17, 55, 36,
tzinfo=tzutc())}, {'Name': 'smart-city-aq', 'CreationDate':
datetime.datetime(2020, 10, 7, 17, 3, 58, tzinfo=tzutc())},
{'Name': 'ukr-data', 'CreationDate': datetime.datetime(2020, 10,
3, 16, 52, 52, tzinfo=tzutc())}, {'Name': 'ukrainian-data-cube',
'CreationDate': datetime.datetime(2019, 8, 30, 9, 11, 40,
tzinfo=tzutc())}], 'Owner': {'DisplayName':
'aws.geo.sdg.project', 'ID':
'ef5f3fbc3855742447c1637d75a04f609f47181ff0ef4d10d41c5ba0ae439bb
d'}}
```

Крок 4– читання даних з s3

Опишемо в obj інформацію про розміщення даних (бакет та ключ об’єкту на s3)

```
import pandas
obj = s3_client.get_object(
    Bucket = 'labpti',
    Key = 'data.csv'
)

# Read data from the S3 object
data = pandas.read_csv(obj['Body'])

# Print the data frame
print('Printing the data frame...')
print(data.head())
```

Результат виведення перших 5 рядків датафрейму data

```
    Printing the data frame...
       Year          Industry_aggregation_NZSIOC         Industry_code_NZSIOC
Industry_name_NZSIOC  \
    0  2019                        Level 1                    99999        All
industries
    1  2019                        Level 1                    99999        All
industries
    2  2019                        Level 1                    99999        All
industries
    3  2019                        Level 1                    99999        All
industries
    4  2019                        Level 1                    99999        All
industries

                  Units Variable_code  \
    0  Dollars (millions)           H01
    1  Dollars (millions)           H04
    2  Dollars (millions)           H05
    3  Dollars (millions)           H07
    4  Dollars (millions)           H08
```

```
                           Variable_name        Variable_category
\
    0                     Total income  Financial performance
    1  Sales, government funding, grants and subsidies  Financial
performance
    2                Interest, dividends and donations  Financial
performance
    3                            Non-operating income  Financial
performance
```

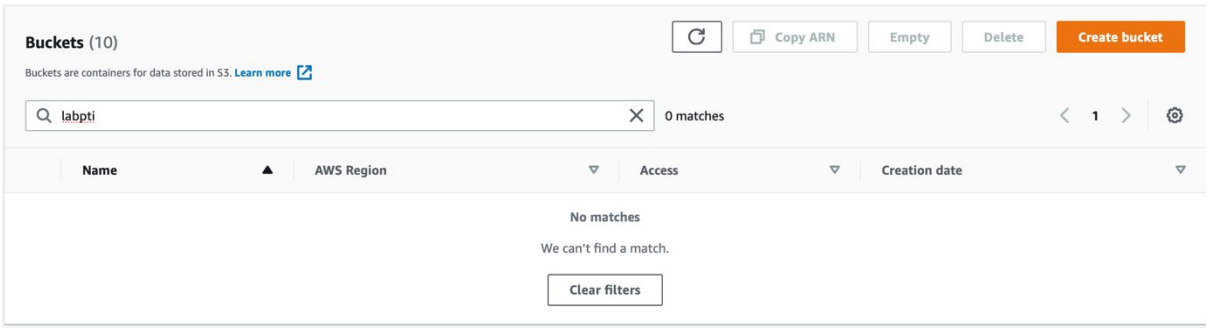
```
    4                                Total expenditure  Financial
performance

        Value                                Industry_code_ANZSIC06
    0  728,239  ANZSIC06 divisions A-S (excluding classes K633...
    1  643,809  ANZSIC06 divisions A-S (excluding classes K633...
    2   62,924  ANZSIC06 divisions A-S (excluding classes K633...
    3   21,505  ANZSIC06 divisions A-S (excluding classes K633...
    4  634,710  ANZSIC06 divisions A-S (excluding classes K633...
```

Крок 5– видалення непотрібного бакета (лише для пустого бакету)

```
def destroy_bucket(bucket_name):
    s3_client = boto3.client('s3')
    response = s3_client.delete_bucket(Bucket=bucket_name)
    print(response)

destroy_bucket("labpti")
```

Результат:

```
{'ResponseMetadata': {'RequestId': 'W65EEN9JZ9MVDCFK', 'HostId':
'WAG8610kCNPvN24OzxU9p8hZMyDctN3MZTpSklHPDX1428+yPaI9X8MyiAAeik6
4aunP/INi53A=', 'HTTPStatusCode': 204, 'HTTPHeaders': {'x-amz-
id-2':
'WAG8610kCNPvN24OzxU9p8hZMyDctN3MZTpSklHPDX1428+yPaI9X8MyiAAeik6
4aunP/INi53A=', 'x-amz-request-id': 'W65EEN9JZ9MVDCFK', 'date':
'Sun, 30 May 2021 19:56:37 GMT', 'server': 'AmazonS3'},
'RetryAttempts': 0}}
```



4.2. 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 клонуватими код з попередньо створеного git-репозиторію та встановить потрібні залежності (pip та необхідні заленості)
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

Примітка: без пунктів 3-4 робота зарахована не буде.

4.3. Додаткові джерела інформації

1. Щодо можливостей Boto3: https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html

Лабораторна робота № 5
Елементи машинного навчання у AWS Sagemaker

Мета роботи: ознайомитись з основами машинного навчання у AWS Sagemaker.

5.1. Порядок виконання роботи

Sagemaker – сервіс машинного навчання у хмарі AWS, який містить чимало вже реалізованих алгоритмів. Окрім наявних користувач може додати ті, які реалізує самостійно (принцип BYOM – Bring Your Own Model). Непогане оглядове відео - https://www.youtube.com/watch?v=Qv_Tr_BCFCQ.

Amazon SageMaker linear learner (https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner.html) – алгоритм для класифікації та лінійної регресії. В ході роботи спробуємо порівняти його із вже звичним для Python-based Data Science та основі SciKit-Learn (https://scikit-learn.org/stable/). Детальніше про Liner Learner за посиланням, однак фактично він є класичним представником методів градієнтного спуску.

В роботі використаємо дані із відритого репозиторію даних для методів машинного навчання UCI - https://archive.ics.uci.edu/ml/datasets/Abalone (спостереження за молюском морське вушко)

Так би мовити об’єкт дослідження))

Примітка: під час виконання роботи можна взяти будь-який відкритий датасет, який в достатній мірі відповідатиме вашому почуття прекрасного ;)

```
Name            Data Type  Meas.  Description
----           ---------  -----  -----------
Sex             nominal           M, F, and I (infant)
Length          continuous mm     Longest shell measurement
Diameter        continuous mm     perpendicular to length
Height          continuous mm     with meat in shell
Whole weight    continuous grams  whole abalone
Shucked weight continuous grams   weight of meat
Viscera weight continuous grams   gut weight (after bleeding)
Shell weight    continuous grams  after being dried
Rings           integer           +1.5 gives the age in years
```

Метою є оцінка віку молюску (фактично - число кілець + 1.5) за наявними ознаками

Імпортуємо потрібні бібліотеки Python (за потреби їх можна встановити засобами pip чи conda - залежить від налаштувань робочого середовища).

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split,
      cross_val_score
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.pipeline import make_pipeline
import matplotlib.pyplot as plt
%matplotlib inline
```

Та створимо датафрейм

```
column_names = ['sex', 'length', 'diameter', 'height',
'whole_weight', 'shucked_weight', 'viscera_weight',
'shell_weight', 'rings']
df = pd.read_csv('abalone.data', names=column_names)
df.head()
```

Структура датасету:

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   sex             4177 non-null   object
 1   length          4177 non-null   float64
 2   diameter        4177 non-null   float64
 3   height          4177 non-null   float64
 4   whole_weight    4177 non-null   float64
 5   shucked_weight  4177 non-null   float64
 6   viscera_weight  4177 non-null   float64
 7   shell_weight    4177 non-null   float64
 8   rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

```
df.describe()
```

	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

Проміжні висновки щодо даних:

- Немає відсутніх даних
- Є суміш категоріальних даних (object) та числових (присутні 2 типи – float64 та int64)

- Залишимо для простоти лише числові дані

```
#Read in the data and prepare the features and targets
column_names = ['sex', 'length', 'diameter', 'height',
'whole_weight', 'shucked_weight', 'viscera_weight',
'shell_weight', 'rings']
df = pd.read_csv('abalone.data', names=column_names)

numeric_features = list(df.select_dtypes([np.number]).columns)
X = df[numeric_features].copy()
X.drop(columns=['rings'], axis=1, inplace=True)
y = df['rings']

# Validation set -  5%
X, X_holdout, y, y_holdout = train_test_split(X,y,
test_size=0.05)


# Split to train and test
X_train, X_test, y_train, y_test = train_test_split(X,y,
test_size=0.2)

# Scale the numeric values
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Create the LinearRegression Model  - sklearn implementation
linreg = LinearRegression()

# fit, score, predict
linreg.fit(X_train_scaled, y_train)
score_scaled = linreg.score(X_test_scaled, y_test)
print("Testing Results")
print(f"R^2 Score: {score_scaled}")
y_pred = linreg.predict(X_test_scaled)

# compute the RMSE of our predictions
rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print(f"Test RSME: {rmse}")
```

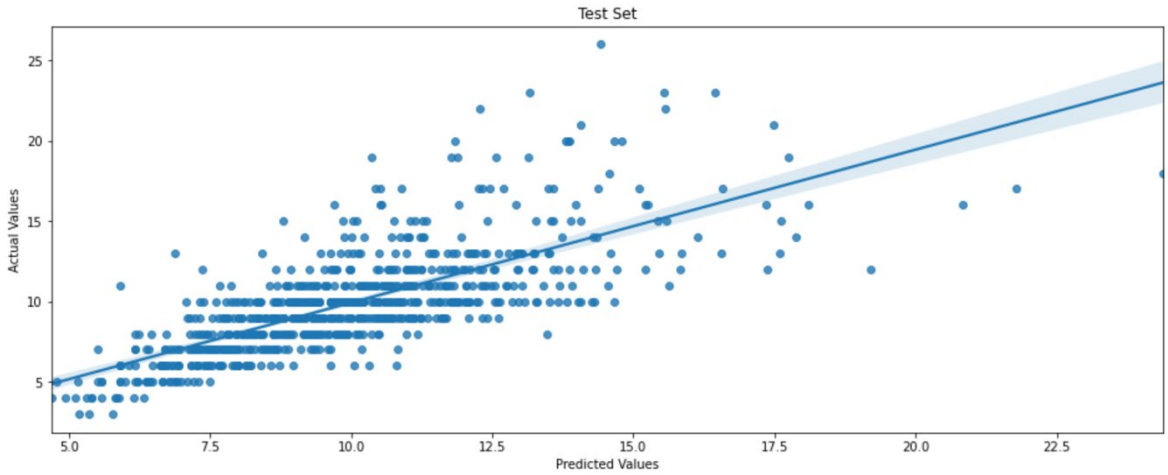
```
Testing Results
R^2 Score: 0.5397781200981733
Test RSME: 2.181369491963945
```

На тестовій вибірці картина є такою

```
# Use Test DataSet
print("Test DataSet")
X_test_scaled = scaler.transform(X_test)
y_pred_test = [round(x, 1) for x in
linreg.predict(X_test_scaled)]
test_prediction_df = pd.DataFrame({
        'actual rings': y_test,
        'predicted rings': y_pred_test
    })
test_prediction_df['rings diff'] = test_prediction_df['actual
rings'] - test_prediction_df['predicted rings']
test_prediction_df.head()
```

	actual rings	predicted rings	rings diff
4000	7	7.9	-0.9
2926	11	9.4	1.6
2750	8	8.2	-0.2
2177	14	16.1	-2.1
3995	5	5.6	-0.6

```
plt.figure(figsize=(16, 6))
plt.title("Test Set")
plt.ylabel("Actual Values")
plt.xlabel("Predicted Values")
fig = sns.regplot(y_pred,y_test.values)
plt.show(fig)
```



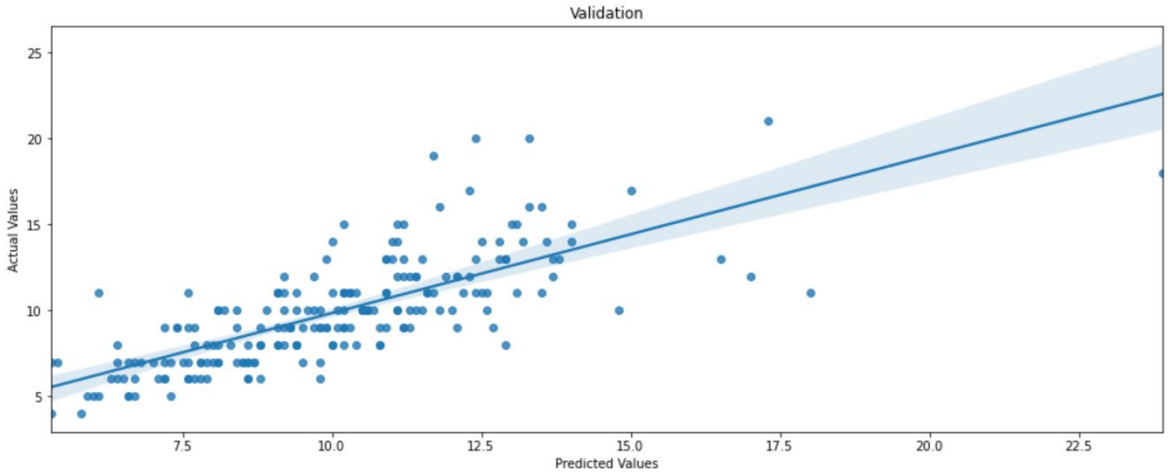
На тестовій вибірці:

```
print("Holdout DataSet")
X_holdout_scaled = scaler.transform(X_holdout)
y_pred_holdout = [round(x, 1) for x in
linreg.predict(X_holdout_scaled)]
prediction_df = pd.DataFrame({
        'actual rings': y_holdout,
        'predicted rings': y_pred_holdout
    })
prediction_df['rings diff'] = prediction_df['actual rings'] -
prediction_df['predicted rings']
prediction_df.head()
```

	actual rings	predicted rings	rings diff
1961	11	13.5	-2.5
2435	14	11.0	3.0
2633	6	8.6	-2.6
730	11	12.6	-1.6
399	11	10.9	0.1

```
plt.figure(figsize=(16, 6))
plt.title("Holdout")
plt.ylabel("Actual Values")
plt.xlabel("Predicted Values")
```

```
fig = sns.regplot(prediction_df['predicted
rings'].values,prediction_df['actual rings'].values)
plt.show(fig)
```



З локальною реалізацією завершено.

Коротко про **AWS Sagemaker** можна прочитати тут- https://aws.amazon.com/getting-started/hands-on/build-train-deploy-machine-learning-model-sagemaker/

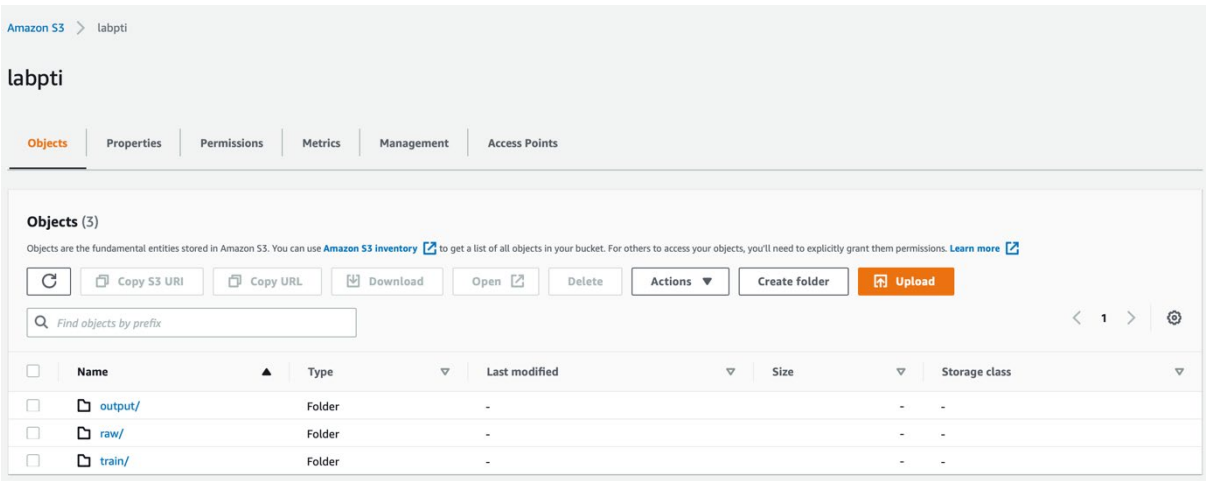
Фактично має місце такий робочий процес (workflow) при роботі з цим сервісом:

1. Створення ноутбучного інстансу
2. Підготовка даних
3. Навчання моделі на даних
4. Деплой моделі для подальшого використання
5. Оцінка ефективності роботи моделі на незалежних даних

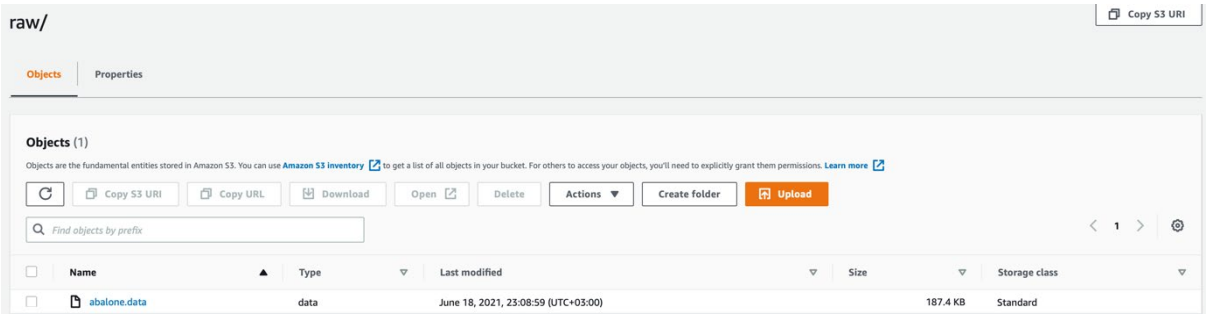
Linear Learner приймає дані в кількох форматах – в тому числі і у CSV (без заголовка)

Для подальшої роботи нам знадобиться S3 бакет – створимо його або використаємо вже існуючий.

На ньому створимо папки output, raw та train.

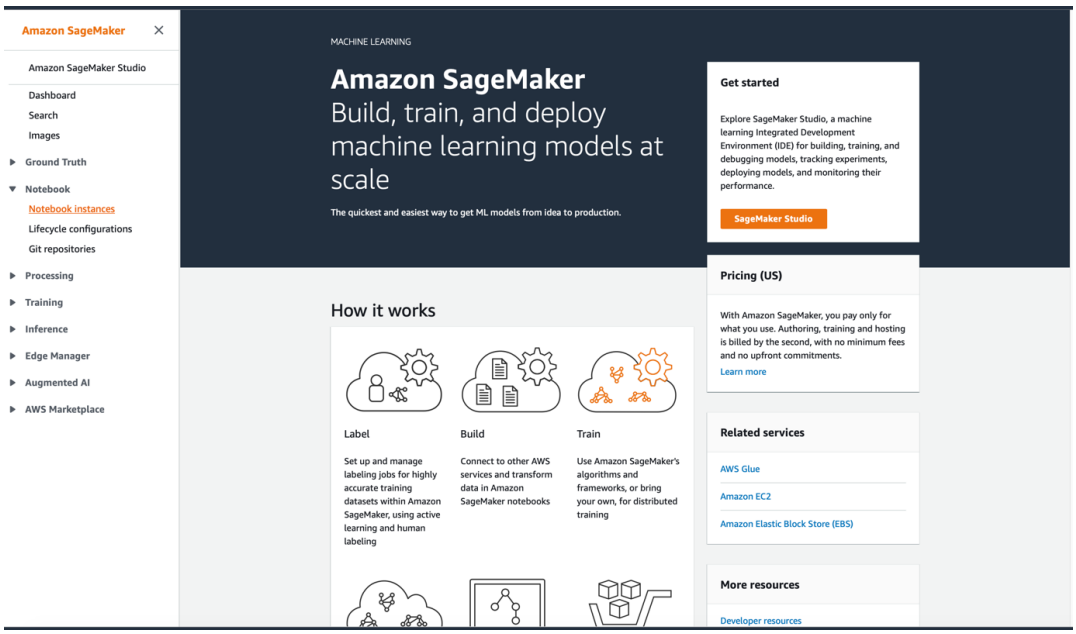


В папку raw помістимо оригінальний файл з даними abalone.data



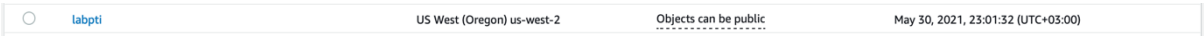
Папка train міститиме дані у RecordIO-форматі (сформуємо в коді нижче), output – результати.

Створимо Sagemaker notenook instance

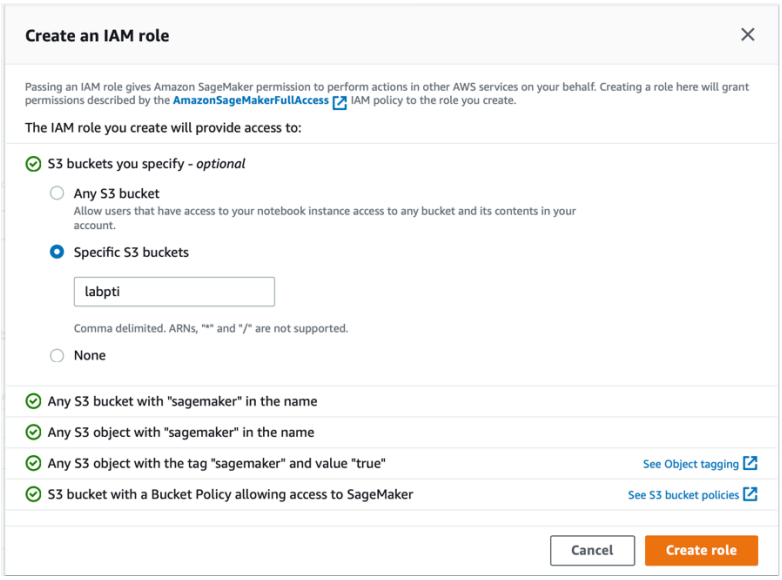


Бажано створюватим інстанс в тому ж регіоні, де і бакет

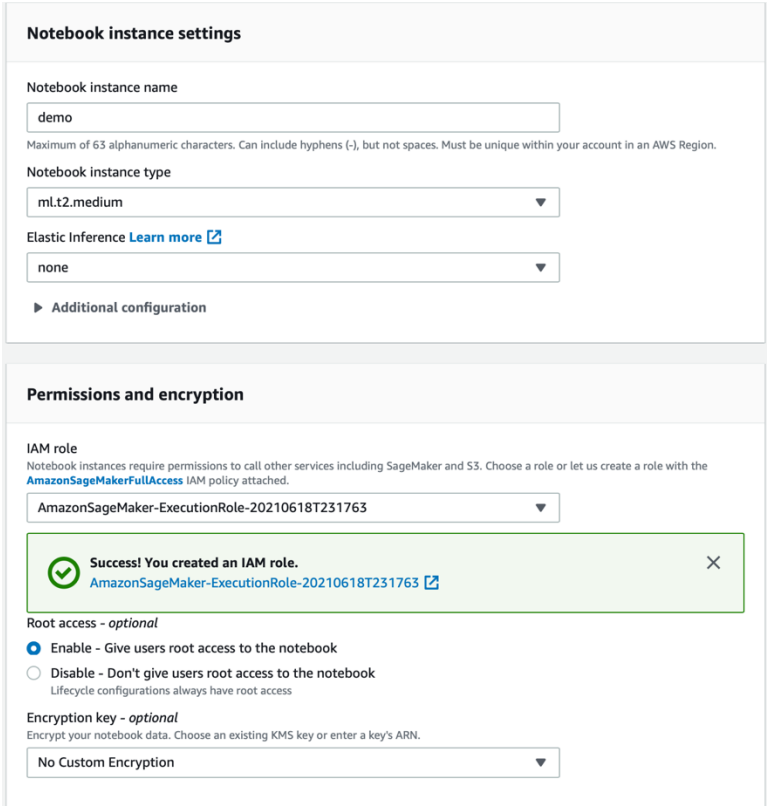
Наш бакет labpti створено у us-west-2 зоні, а тому у ноутбук для Sagemaker створимо в цій зоні.



Тип інстансу - базовий із Free Tier (ml.t2.medium) - його цілком достатньо для нескладних моделей.

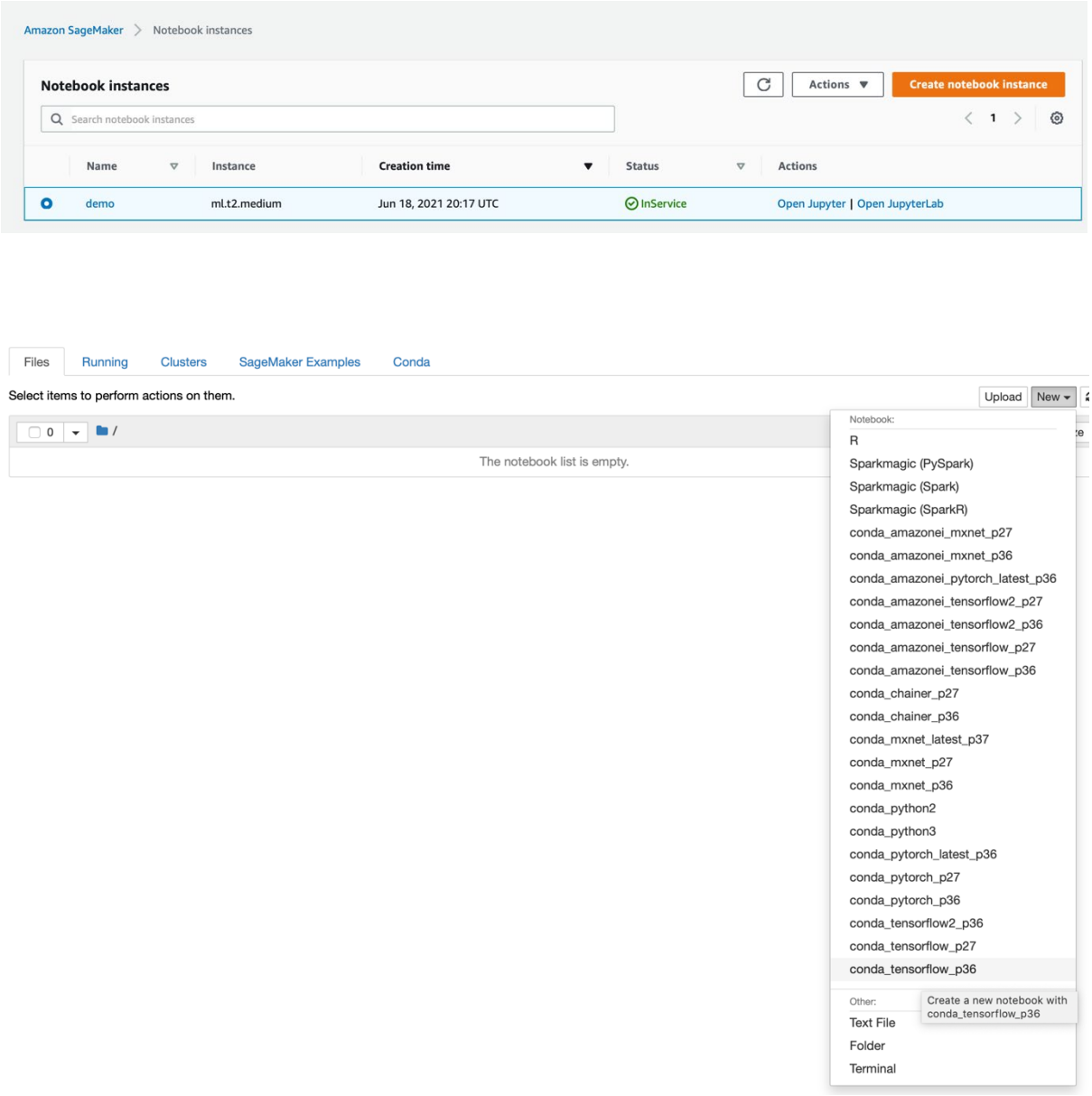


Роль, яка буде використовуватись для доступу до бакету із AWS Sagemaker інстансу

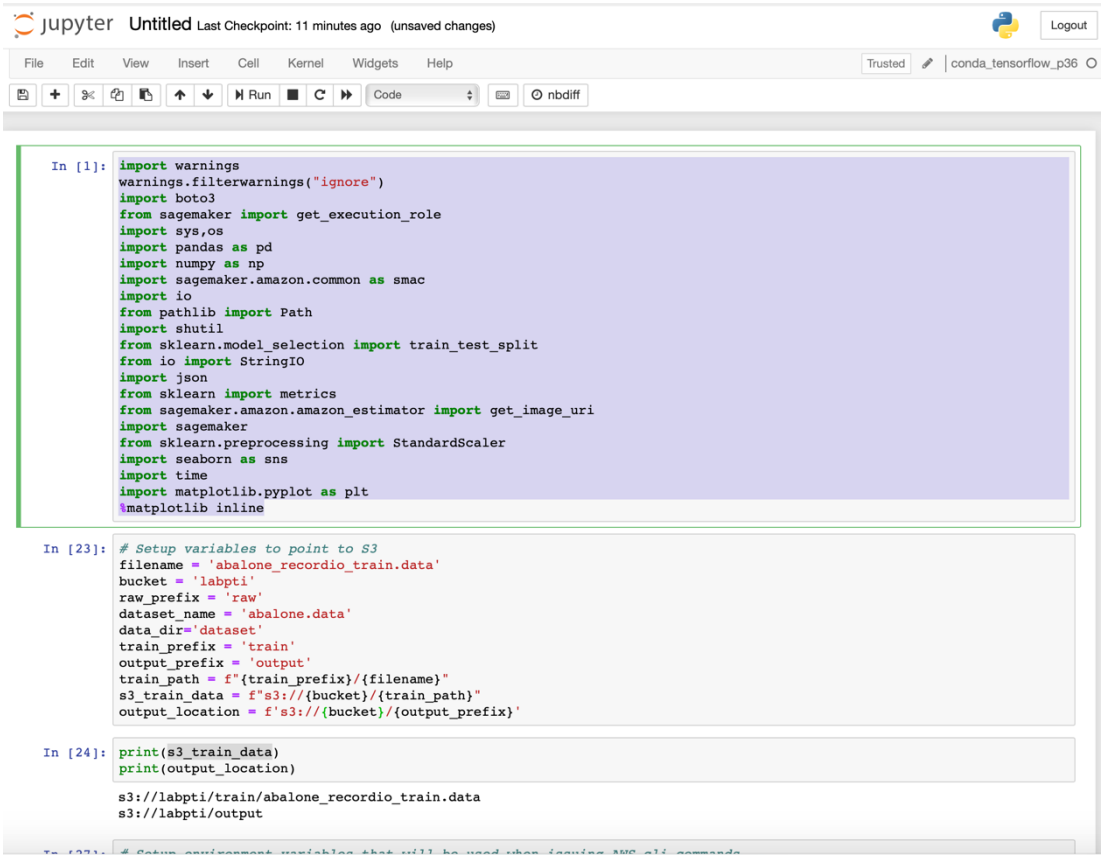


IAM роль передбачає доступ лише до бакету labpti

Ноутбучний інстанс готовий до використання і має статус InService у переліку Notebook Instances для AWS Sagemaker.



Створення jupyter notebook для подальшої роботи



Внесемо усі потрібні імпорти

```
import warnings
warnings.filterwarnings("ignore")
import boto3
from sagemaker import get_execution_role
import sys,os
import pandas as pd
import numpy as np
import sagemaker.amazon.common as smac
import io
from pathlib import Path
import shutil
from sklearn.model_selection import train_test_split
from io import StringIO
import json
from sklearn import metrics
from sagemaker.amazon.amazon_estimator import get_image_uri
import sagemaker
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import time
import matplotlib.pyplot as plt
%matplotlib inline
```

Вкажемо параметри взаємодії із S3:

```
filename = 'abalone_recordio_train.data'
bucket = 'labpti'
raw_prefix = 'raw'
dataset_name = 'abalone.data'
data_dir='dataset'
train_prefix = 'train'
output_prefix = 'output'
train_path = f"{train_prefix}/{filename}"
s3_train_data = f"s3://{bucket}/{train_path}"
output_location = f's3://{bucket}/{output_prefix}'
```

Перевіримо шляхи до даних та виходів моделі:

```
print(s3_train_data)
print(output_location)
```

```
    s3://labpti/train/abalone_recordio_train.data
    s3://labpti/output
```

Налаштуємо також змінні оточення для роботи з AWS CLI:

```
%env DATA_DIR=$data_dir
%env S3_DATA_BUCKET_NAME = $bucket/$raw_prefix
%env DATASET_NAME = $dataset_name
%env TRAINING_PATH = $bucket/$train_prefix
```

```
    env: DATA_DIR=dataset
    env: S3_DATA_BUCKET_NAME=labpti/raw
    env: DATASET_NAME=abalone.data
    env: TRAINING_PATH=labpti/train
```

Заберемо сирі дані з S3:

```
!aws s3 cp s3://$S3_DATA_BUCKET_NAME/$DATASET_NAME ./$DATA_DIR/
```

download: s3://labpti/raw/abalone.data to dataset/abalone.data

Тепер файл вже на інстансі та з ним можна працювати, як у локальному випадку:

```
column_names = ['sex', 'length', 'diameter', 'height',
'whole_weight', 'shucked_weight', 'viscera_weight',
'shell_weight', 'rings']
```

```
df = pd.read_csv('./dataset/abalone.data', names=column_names)
```

```
df.head()
```

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

```
df.shape
```

(4177, 9)

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   sex             4177 non-null   object 
 1   length          4177 non-null   float64
 2   diameter        4177 non-null   float64
 3   height          4177 non-null   float64
 4   whole_weight    4177 non-null   float64
 5   shucked_weight  4177 non-null   float64
 6   viscera_weight  4177 non-null   float64
 7   shell_weight    4177 non-null   float64
 8   rings           4177 non-null   int64  
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

```
df.describe()
```

	sex	length	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

Як можна побачити – всі операції працюють аналогічно.

Тепер, як і раніше, залишимо лише цифрові величини

```
numeric_features = list(df.select_dtypes([np.number]).columns)
X = df[numeric_features].copy()
X.drop(columns=['rings'], axis=1, inplace=True)
y = df['rings']
```

Розділимо датасет на вибірки – на тестову частину залишимо 5% даних (3968+209 = 4177 – загальна кількість зразків).

```
# create a holdout set, 5%
X_train, X_holdout, y_train, y_holdout = train_test_split(X,y,
test_size=0.05)


X_train.shape

    (3968, 7)


X_holdout.shape

    (209, 7)
```

Хоча алгоритми Sagemaker і працюють із CSV, значно ефективнішим для мережевого доступу є recordIO-формат. Засобами boto3 збережемо модифікований датасет на S3.

```
buf = io.BytesIO()

smac.write_numpy_to_dense_tensor(buf,
np.array(X_train).astype('float32'),
np.array(y_train).astype('float32'))
buf.seek(0)
boto3.resource('s3').Bucket(bucket).Object(f'{train_path}').uplo
ad_fileobj(buf)
```

AWS Sagemaker надає різні вже реалізовані методи машинного начання у вигляді AMI-образів – в даному випадку використаємо linear-learner контейнер

```
container = get_image_uri(boto3.Session().region_name, 'linear-
learner')
```

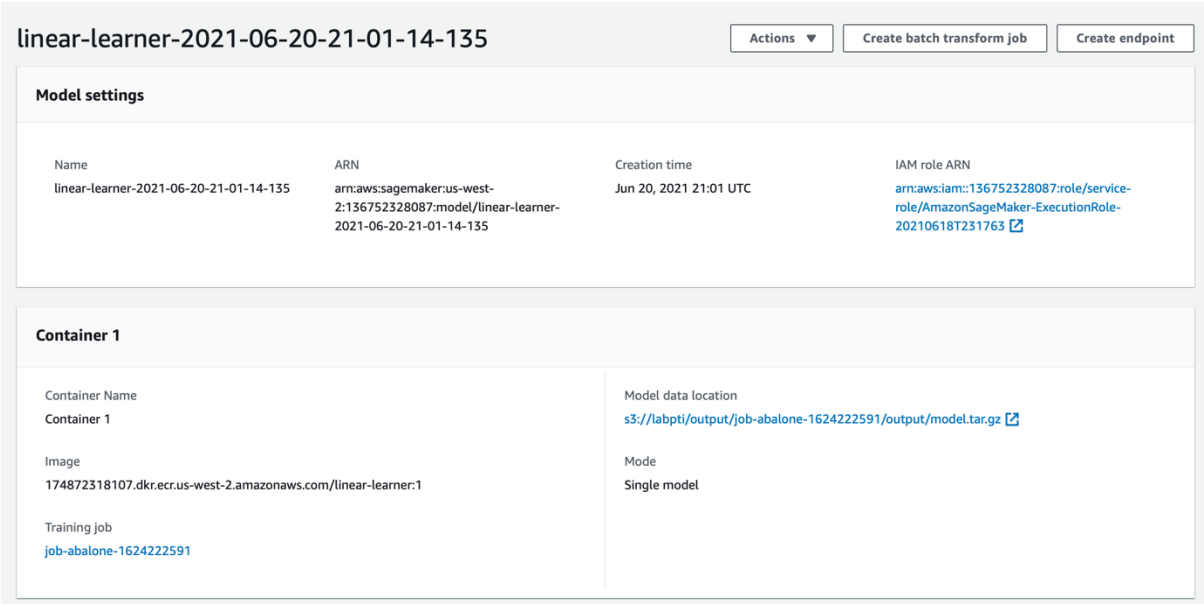
Тепер створимо сесію sagemaker та отримаємо IAM роль. Далі через Estimator API виберемо інстанс, який відповідає задачі та датасету. В секції з

гіперпаратмерами зробимо лише нормалізацію даних (feature_dim=7 – відповідає числу ознак у датасеті)

Власне сам процесс триває досить довго оскільки Sagemaker створює новий інстанс та відбирає оптимальну модель, яка буде збережена у output_path папку.

```
sess = sagemaker.Session()

role = get_execution_role()

linear = sagemaker.estimator.Estimator(container,
                                       role,
                                       train_instance_count=1,

train_instance_type='ml.c4.xlarge',
                                       output_path=output_location,
                                       sagemaker_session=sess)

linear.set_hyperparameters(feature_dim=7, epochs=20,
num_models=32, loss='absolute_loss',
                           predictor_type='regressor',
                           mini_batch_size=32,
                          normalize_data=True,
normalize_label=False)

linear.fit({'train': s3_train_data}, job_name=f"job-abalone-
{int(time.time())}")
```

Навчена модель на S3:



Тепер можна задеплоїти модель. Тепер вже використовується ml.t2.medium інстанс – модель вже навчена і для її використання потрібно менше ресурсів. Створюємо endpoint моделі, який забезпечує доступ до неї.

```
linear_predictor = linear.deploy(initial_instance_count=1,
                                 instance_type='ml.t2.medium',
endpoint_name="abalone-endpoint")
```

```
X_holdout.values
```

```
    array([[0.65  , 0.525 , 0.165 , ..., 0.647 , 0.2485, 0.3005],
           [0.58  , 0.42  , 0.14  , ..., 0.3285, 0.102 , 0.2255],
           [0.59  , 0.44  , 0.15  , ..., 0.387 , 0.215 , 0.245 ],
           ...,
           [0.335 , 0.26  , 0.09  , ..., 0.0875, 0.041 , 0.056 ],
           [0.46  , 0.345 , 0.11  , ..., 0.235 , 0.0885, 0.116 ],
           [0.67  , 0.55  , 0.19  , ..., 0.5425, 0.3035, 0.4   ]]
)
```

```
X_holdout.shape
```

```
    (209, 7)
```

Виористаємо метод predict для linear_predictor endpoint, щод отримати вихід моделі для тестового датасету. Метод predict поверне json з прогнозом.

```
from sagemaker.predictor import csv_serializer,
json_deserializer
linear_predictor.serializer = csv_serializer
linear_predictor.deserializer = json_deserializer
result = linear_predictor.predict(X_holdout.values)
```

Заберемо з отриманого json прогнозні значення та розрахуємо RMSE помилку.

```
predictions = [ x['score'] for x in result["predictions"]]
print(f"RSME:
{np.sqrt(metrics.mean_squared_error(y_holdout.values,
predictions))}")
```

```
    RSME: 2.1730089779732293
```

Результати є цілком порівняними із локальних варіантом для даного прикладу.

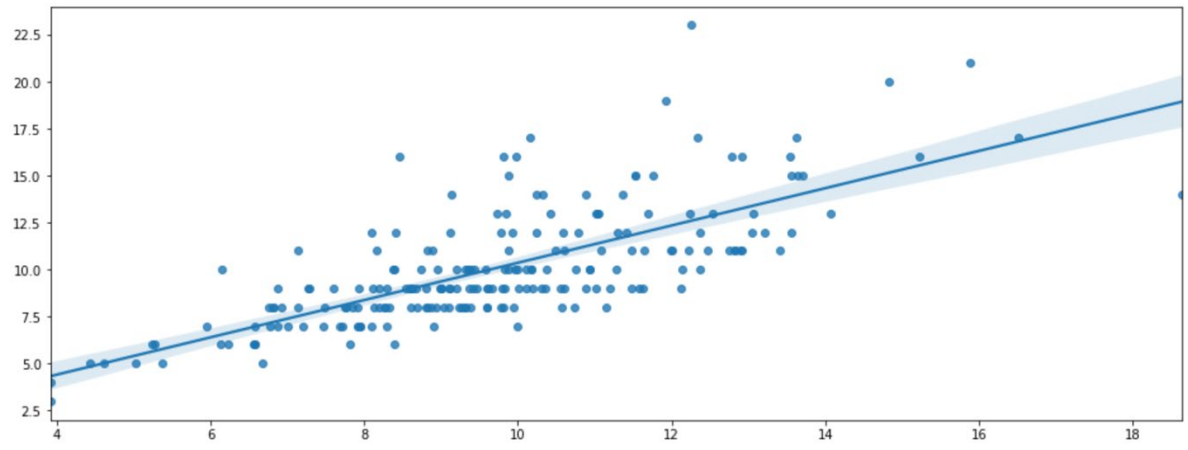
```
print("Holdout DataSet")

prediction_df = pd.DataFrame({
        'actual rings': y_holdout.values,
        'predicted rings': predictions
    })
prediction_df['rings diff'] = prediction_df['actual rings'] -
prediction_df['predicted rings']
prediction_df.head()
```

```
Holdout DataSet
```

	actual rings	predicted rings	rings diff
0	9	9.614385	-0.614385
1	9	9.827354	-0.827354
2	8	9.812040	-1.812040
3	8	9.266432	-1.266432
4	14	11.364495	2.635505

Ну і виведемо на графік фактичні та прогнозі значення віку молюсків.

```
plt.figure(figsize=(16, 6))
sns.regplot(np.array(predictions),np.array(y_holdout.values))
```



По завершенні роботи обов'язково потрібно видалити endpoint, щоб уникнути витрат.

```
import sagemaker
sagemaker.Session().delete_endpoint(linear_predictor.endpoint)
```

5.2. Завдання

1. Обрати датасет у репозиторії https://archive.ics.uci.edu/ml/index.php (варіанти датасету мають бути погоджені з викладачем та не перетинатися).
2. Вивчити його особливості.
3. Вирішити задачу класифікації / кластеризації за допомогою можливостей AWS Sagemaker.
4. Результати оформити протоколом.

5.3. Додаткові джерела інформації

Для самостійного опрацювання пропонується перелік реалізованих алгоритмів https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html (в роботі ми познайомилися лише з Linear Learner) та ресурс, на якому зібрані приклади роботи з іншими алгоритмами у AWS Sagemaker https://sagemaker-examples.readthedocs.io/en/latest/training/algorithms.html.

Перелік корисних посилань

References

1. Мережеве сховище AWS для доступу до даних https://registry.opendata.aws/sentinel-2-l2a-cogs/.
2. https://www.opendatacube.org/.
3. Інструкція для налаштування Anaconda - https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html.
4. Шелестов А. Ю. Методи глибинного навчання для геопросторового аналізу та задач спостереження Землі / Шелестов А. Ю., Лавренюк М. С., Яйлимов Б. Я., Ткаченко О. М. // К.: "Наукова думка" – 2019. – 228 с.
5. Andrii Shelestov, Mykola Lavreniuk, Vladimir Vasiliev, Leonid Shumilo, Andrii Kolotii, Bohdan Yailymov, Nataliia Kussul, Hanna Yailymova. Cloud Approach to Automated Crop Classification Using Sentinel-1 Imagery. IEEE Transactions on Big Data – 2020. – Vol. 6, No. 3. – 572-582 pp.
6. Куссуль Н.М., Скакун С.В., Шелестов А.Ю. Аналіз ризиків надзвичайних ситуацій на основі супутникових даних. Моделі і технології. К.: "Наукова думка" – 2014. – 184 с.
7. M. Hosseini, H. McNairn, et all. A Comparison between Support Vector Machine and Water Cloud Model for Estimating Crop Leaf Area Index. *Remote Sensing. – 2021. – Vol. 13, No. 7. – P. 1-20. DOI: 10.3390/rs13071348*
8. Shumilo, L., Lavreniuk, M., Skakun, S., & Kussul, N. (2021). Is Soil Bonitet an Adequate Indicator for Agricultural Land Appraisal in Ukraine? Sustainability, 13(21), 12096. DOI: 10.3390/su132112096
9. Shelestov, A., Lavreniuk, M., et al. (2019). Cloud approach to automated crop classification using Sentinel-1 imagery. IEEE Transactions on Big Data, 6(3), 572-582. DOI: 10.1109/TBDATA.2019.2940237
10. Kussul, N., Lavreniuk, M., et al. (2019). A workflow for Sustainable Development Goals indicators assessment based on high-resolution satellite data. International Journal of Digital Earth. DOI: 17538947.2019.1610807
11. Skakun, S., Justice, C. O., Kussul, N., Shelestov, A., & Lavreniuk, M. (2019). Satellite data reveal cropland losses in South-Eastern Ukraine under military conflict. Frontiers in Earth Science, 7, 305. DOI:10.3389/feart.2019.00305
12. N. Kussul, K. Deininger, L. Shumilo, M. Lavreniuk, D. Ayalew Ali., O. Nivievskyi Biophysical Impact of Sunflower Crop Rotation on Agricultural Fields. Sustainability. — 2022. – No. 14(7):3965 — pp. 125-132. https://doi.org/10.3390/su14073965.
13. Азарсков В.М., Блохин Л.Н., Житецкий Л.С., Куссуль Н.Н. Робастные методы оценивания, идентификации и адаптивного управления // К.: НАУ, 2004. — 500 с.
14. N. Kussul, A. Shelestov, M. Lavreniuk, I. Butko and S. Skakun, "Deep learning approach for large scale land cover mapping based on remote sensing data fusion," 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2016, pp. 198–201.

15. N. Kussul, N. Lavreniuk, A. Shelestov, B.Yailymov, I. Butko Land Cover Changes Analysis Based on Deep Machine Learning Technique.- Journal of Automation and Information Sciences. — 2016. — PP. 42–54.
16. Куссуль Н.Н., Шелестов А.Ю. Grid-системы для задач исследования Земли. Архитектура, модели и технологии // К.: “Наукова думка”, 2008. – 452 с.