Hurtownia danych w chmurze

Jerzy Niemczyk
Szymon Puszcz
Jacek Matczyński

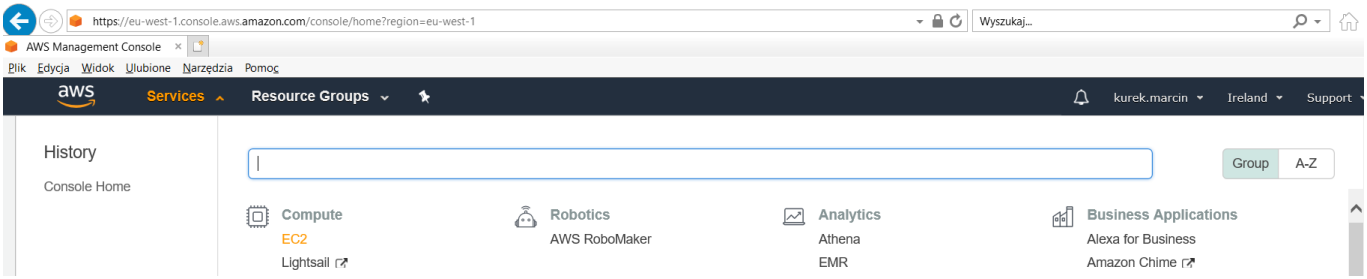
14.06.2019

Dokumentacja projektowa “Hurtownia danych w chmurze”

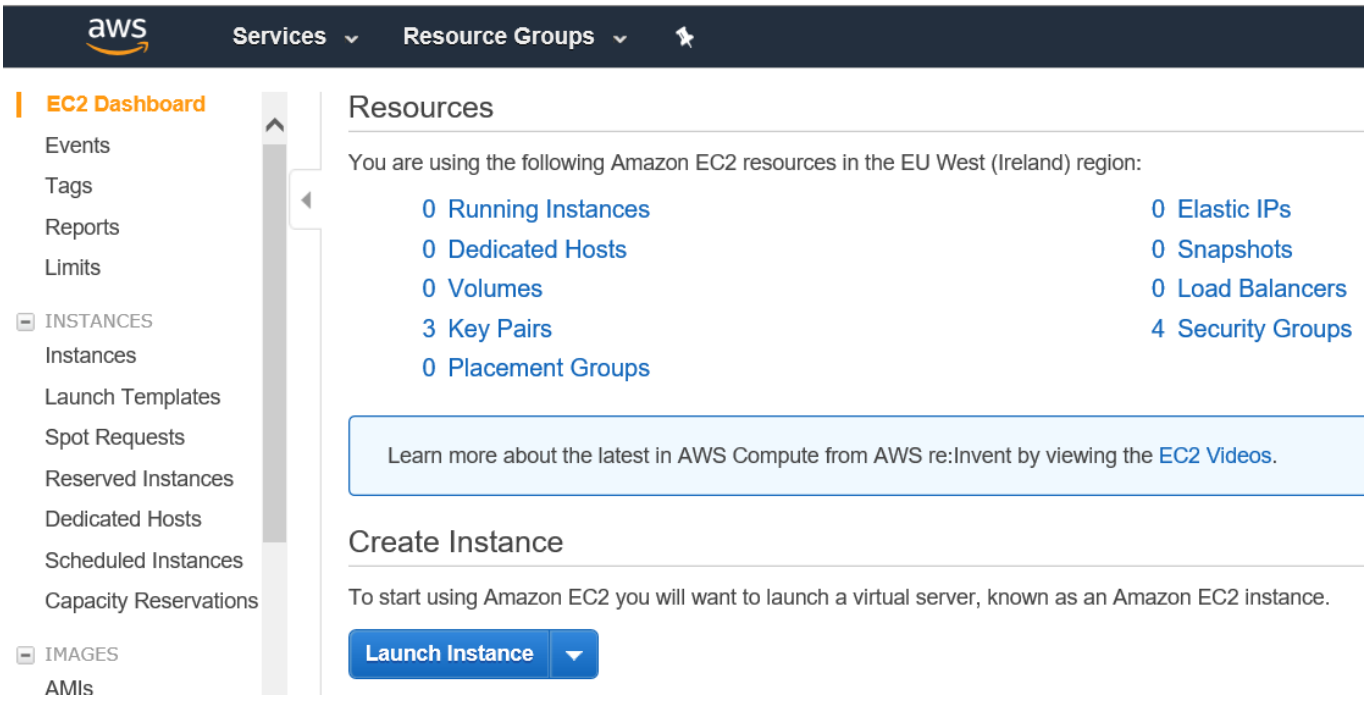
1. Konfiguracja usług Amazon

1. Uruchomienie instancji na Amazon Web Services (dalej AWS)

W pierwszej kolejności należy zarejestrować się na portalu AWS https://aws.amazon.com/. Po udanej rejestracji i zalogowaniu się, powinniśmy zostać przekierowani na stronę https://eu-west-1.console.aws.amazon.com/console/home?region=eu-west-1. Następnie należy przejść w zakładkę Services -> w kategorię Compute -> usługę EC2 (widok ekranu poniżej)

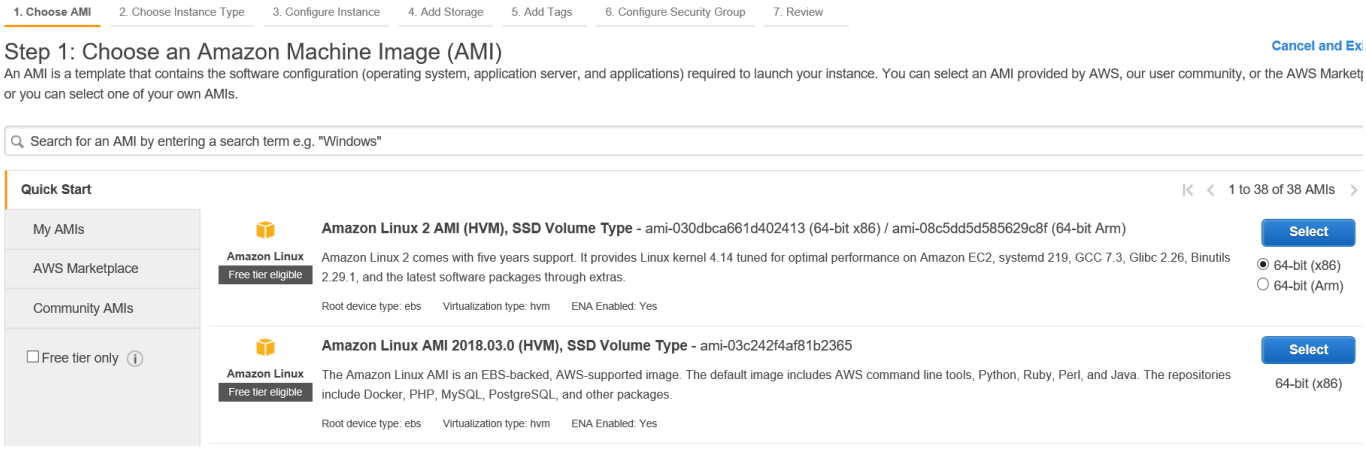


Następnie w kategorii Create Instance należy wdusić przycisk Launch Instance (widok ekranu poniżej)

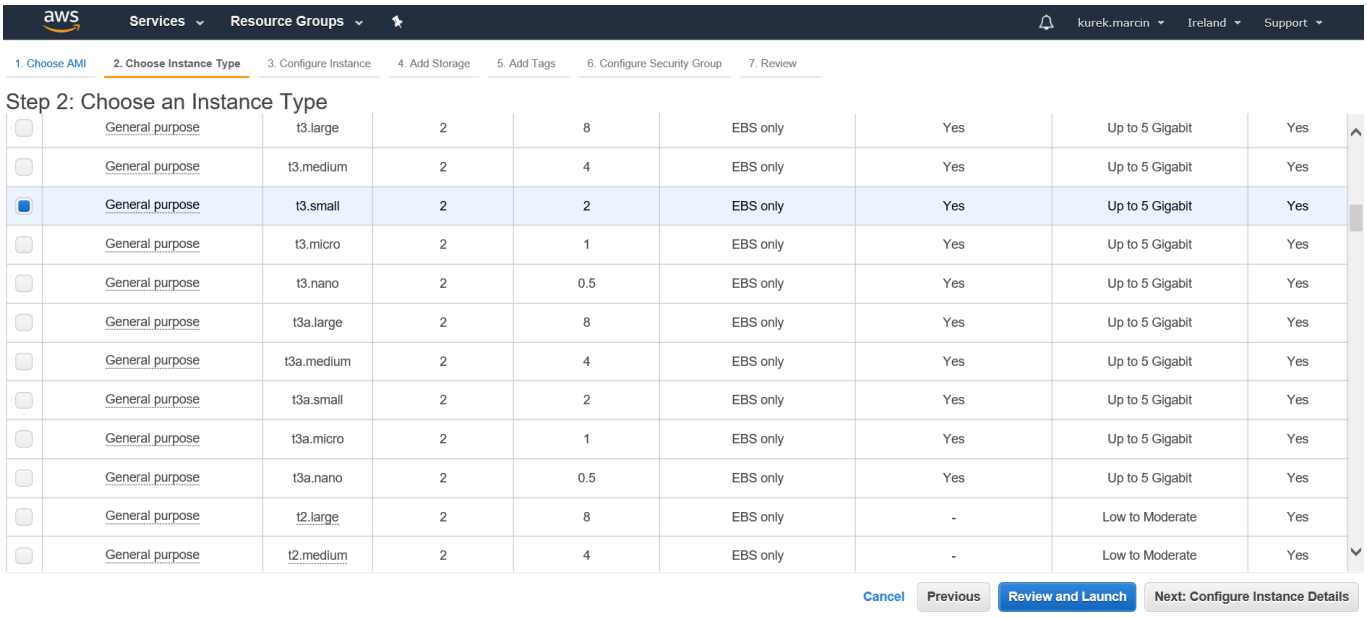


Następnie pojawiają się do wyboru gotowe typy maszyn. W projekcie została wykorzystana następująca: Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type. W celu wyboru maszyny, należy wdusić przycisk

Select. (widok ekranu poniżej)



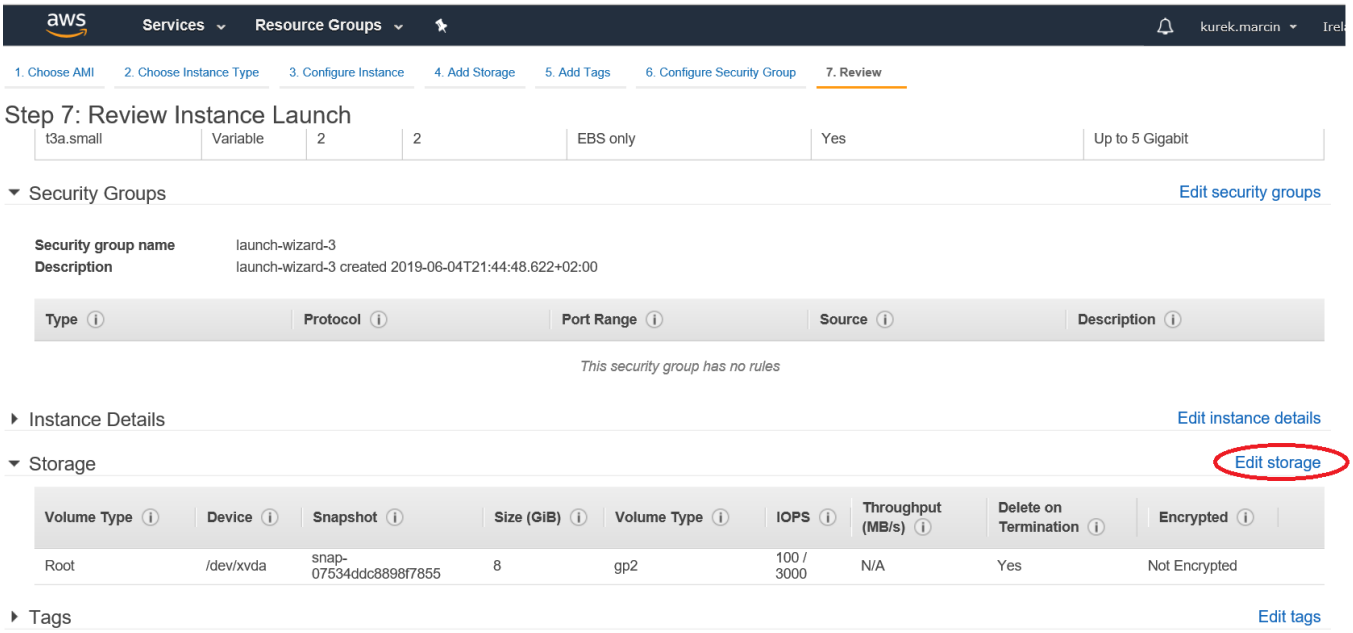
Kolejny krok to wybór parametrów maszyny, spośród dostępnych typów instancji. Ze względu na niewielkie potrzeby projektu wybrany został typ instancji o nazwie t3.small zawierający 2 x vCPU (wirtualne procesory) oraz 2 GiB pamięci RAM. Po wybraniu odpowiedniego typu instancji w celu przejścia do kolejnego etapu należy wcisnąć przycisk Review and Launch (widok ekranu poniżej)



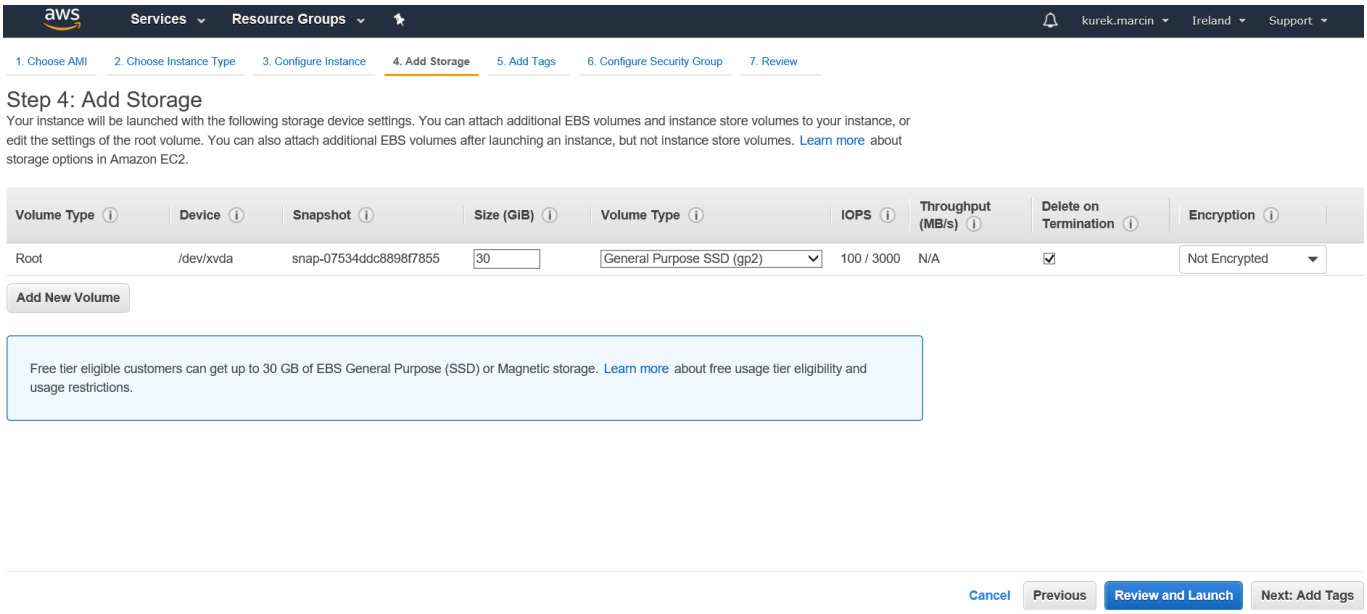
☐	General purpose	t3.large	2	8	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3.medium	2	4	EBS only	Yes	Up to 5 Gigabit	Yes
☑	General purpose	t3.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.large	2	8	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.medium	2	4	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t3a.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes

Następnie zostaniemy przekierowaniu do widoku podsumowującego budowaną przez nas instancję. Ze względu na potrzeby instalacji oraz przechowywania danych należy wejść w zakładkę Edit storage w celu

zwiększenia pamięci dyskowej na maszynie (widok ekranu poniżej)



W widoku edycji pamięci dyskowej, zwiększamy z domyślnej wartości 8 GB na 30 GB (widok ekranu poniżej). Następnie wduszamy przycisk Review and Launch (widok ekranu poniżej)



Kolejny krok to widok podsumowania budowanej instancji oraz informacja o kosztach związanych z maszyną. Po zapoznaniu się z podsumowaniem i zweryfikowaniu, że wszystkie parametry są takie jak oczekiwaliśmy klikamy następująco Launch -> pojawia się okienko związane z kluczami niezbędnymi do połączenia z instancją. Wybieramy z listy rozwijalnej opcję – Create a new key pair, wpisujemy w Key pair name dowolną nazwę klucza, następnie pobieramy klucz na swoją lokalną maszynę wciskając przycisk Download Key Pair oraz klikamy finalne Launch Instance. Maszyna w tym momencie jest dla nas przygotowana, jej uruchomienie nie powinno zająć więcej niż kilkaset sekund.

Status uruchomione maszyny można sprawdzić pod adresem:

https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#Instances:sort=instanceId

Ze względów regulacyjnych do projektu zostały wykorzystane jedynie środowiska dostępne w regionie Europy – Irlandia.

2. Konfiguracja maszyny AMI oraz instalacja Oracle DB

Kolejne czynności są już wykonywane bezpośrednio na postawionej maszynie na AWS.

Połączyć się do niej możemy za pomocą polecenia ssh z wykorzystaniem wygenerowanego wcześniej klucza do instancji.

```
ssh -i "MK_keypair.pem" ec2-user@ec2-54-246-212-140.eu-west-
1.compute.amazonaws.com
```

1. Konfiguracja instancji:

   ```
   [ec2-user@ip-172-31-39-154 ~]$ sudo su
   [root@ip-172-31-39-154 ec2-user]# dd if=/dev/zero of=/swapfile1 bs=1024
   count=2097152
   [root@ip-172-31-39-154 ec2-user]# chmod 0600 /swapfile1
   [root@ip-172-31-39-154 ec2-user]# mkswap /swapfile1
   [root@ip-172-31-39-154 ec2-user]# swapon /swapfile1
   [root@ip-172-31-39-154 ec2-user]# swapon –s
   [root@ip-172-31-39-154 ec2-user]# vi /etc/fstab
   [root@ip-172-31-39-154 ec2-user]# /swapfile swap swap defaults 0 0
   ```

2. Pobranie instalacji ze strony producenta

   https://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/xe-prior-releases-5172097.html

   ```
   [root@ip-172-31-39-154 ec2-user]# wget
   https://download.oracle.com/otn/linux/oracle11g/xe/oracle-xe-11.2.0-
   1.0.x86_64.rpm.zip
   ```

3. Instalacja oprogramowania

   ```
   [root@ip-172-31-39-154 ec2-user]# rpm -i oracle-xe-11.2.0-1.0.x86_64.rpm
   ```

4. Konfiguracja podstawowych ustawień bazy XE (hasło dla SYS, port itd.)

   ```
   [root@ip-172-31-39-154 ec2-user]# /etc/init.d/oracle-xe configure
   ```

5. Konfiguracja bazy danych

   Logowanie jako Sys:

```
[root@ip-172-31-39-154 ec2-user]# sqlplus sys/password as sysdba
```

Następujące polecenia są już wykonywane w bazie danych Oracle:

```
CREATE USER student IDENTIFIED BY student;
GRANT CONNECT TO student;
GRANT CONNECT, RESOURCE, DBA TO student;
GRANT CREATE SESSION TO student;
GRANT UNLIMITED TABLESPACE TO student;
ALTER USER student IDENTIFIED BY student ACCOUNT UNLOCK;
CREATE OR REPLACE DIRECTORY test_dir AS '/home/ec2-user/oracle';
GRANT READ, WRITE ON DIRECTORY test_dir TO student;
commit;
```

2. Stworzenie lokalnej bazy danych i załadowanie rekordów

Język zapytań SQL różni się w zależności od środowiska, w którym utworzona jest baza danych. Ze względu na różnice pomiędzy SQL używanym w Teradata i Oracle stworzono skrypt transformujący polecenie `CREATE TABLE`

Skrypt `oracle.py` transformuje skrypty sql z folderu `teradata` i tworzy w folderze `oracle` pliki z poleceniem `CREATE TABLE` zgodnym ze standardem Oracle.

- uruchomienie skryptu: `python oracle.py <nazwa-pliku-z-folderu-teradata>`

oracle.py

```
import re
import sys

print(sys.argv[1])
path = 'teradata/' + sys.argv[1]

translation = {
    ",?\\s*(NO)? FALLBACK\\s*,?": "",
    ",?\\s*(NO)? BEFORE JOURNAL\\s*,?": "",
    ",?\\s*(NO)? AFTER JOURNAL\\s*,?": "",
    ",?\\s*CHECKSUM = DEFAULT\\s*,?": "",
    "DEFAULT MERGEBLOCKRATIO\\s*,?": "",
    "PRIMARY INDEX\\s*(\\(([^()]*)\\))": "",
    "PARTITION BY RANGE_N\\s*(\\(([^()]*)\\))": "",
    "INDEX\\s*(\\(([^()]*)\\))": "",
    "COMPRESS .*,": ",",
    "CHARACTER SET \w+": "",
    "(NOT)? CASESPECIFIC": "",
    "FORMAT '.*'": "",
    "TIME": "TIMESTAMP",
    "BIGINT": "NUMBER(19)",
```

```
    "DECIMAL": "NUMBER",
    "DOUBLE PRECISION": "NUMBER",
    "FLOAT": "NUMBER",
    "INTEGER": "NUMBER(10)",
    "NUMERIC": "NUMBER",
    "REAL": "NUMBER",
    "SMALLINT": "NUMBER(5)",
    "BYTEINT": "NUMBER(3)",
    "(MULTISET|SET)": "",
    "UNIQUE": ""
}
sql_file = open(path, "r")
content = sql_file.read()

for k, v in translation.items():
    content = re.sub(k, v, content)
out_path = path.replace(".txt", ".sql")
out_path = out_path.replace("teradata", "oracle")
out_file = open(out_path, "w+")
out_file.write(content)

sql_file.close()
```

---

3. Transfer danych z lokalnej bazy do bazy znajdującej się na AWS za pomocą datapump

1. Wykonanie eksportu lokalnej bazy danych do pliku z wykorzystaniem polecenia `expdp`

Polecenie posiada nastepującą strukturę:

```
expdp <nazwa_użytkownika>/<hasło>@<SID> [TABLES=<tabela1,tabela2...> | SCHEMAS=
<schemat1,schemat2...>] DUMPFILE=<katalog>:<nazwa_pliku> [VERSION=<wersja_oracle>]
```

- **nazwa_użytkownika,hasło** - dane autentykacyjne użytkonika lokalnej bazy danych
- **SID** - SID bazy danych, z której ma być wykonany eksport danych, paramter opcjonalny
- **tabela1,tabela2...** - nazwy tabel rozdzielone przecinkami, które są przeznaczone do eksportu
- **schema1,schemat2...** - nazwy schematów rozdzielone przecinkami, które są przeznaczone do eksportu
- **katalog** - katalog w którym zostanie zapisany plik z wyeksportowanymi danymi, katalog musi być zdefiniowany w bazie danych oraz użytkownik musi posiadać do niego prawa dostępu
- **nazwa_pliku** - nazwa powstałego pliku z eksportem danych
- **wersja_oracle** - wersja bazy danych Oracle, z którą ma być zgodny wyeksportowany plik, parametr opcjonalny

> **UWAGA!** Nie można jednocześnie wskazać tabel i schematów, są to parametry wzajemnie wykluczające się. W celu poprawnego eksportu zawsze musi być podany jeden z nich (`SCHEMAS` lub `TABLES`)

2. Przesłanie pliku zawierającego wyeksportowne dane do AWS

W celu przesłania pliku niezbędny jest klucz zawarty w pliku `*.pem` pozwalający na autentykację podczas ustanawiania połączenia.

W celu przesłania pliku do AWS wykorzystuje się polecenie `scp` posiadające następującą strukturę:

```
scp -i <ścieżka_do_pliku_*.pem> <lokalna_ścieżka> <nazwa_użytkownika>@<adres_ip>:
<zdalna_ścieżka>
```

- **ścieżka_do_pliku_*.pem** - względna lub bezwzględna ścieżka do pliku *.pem zawierajacego klucz pozwalający na autentykację użytkownika
- **lokalna_ściężka** - względna lub bezwzględna ścieżka do pliku, który ma być przesłany
- **nazwa_uzytkownika** - nazwa użytkonika na AWS
- **adres_ip** - adres ip AWS
- **zdalna_ścieżka** - ścieżka w której ma być umiejscowiony przesyłany plik. Ścieżka nie może wskazywać jedynie na folder, musi zawierać także nazwę nowo utworzonego pliku np. `/home/student/oracle/export.dmp`

3. Wykonanie importu danych znajdujących się w pliku do bazy danych na AWS

Aby wykonać import danych należy połączyć się za pomocą SSH z maszyną AWS, na której znajduje się baza danych, do której chcemy zaimportowac dane. W celu autentykacji użytkownika niezbędny jest plik *.pem.

W celu połączenia należy użyć polecenia `ssh` posiadającego następującą strukturę:

```
ssh -i <ścieżka_do_pliku_*.pem> <nazwa_użytkownika>@<adres_ip>
```

- **ścieżka_do_pliku_*.pem** - względna lub bezwzględna ścieżka do pliku *.pem zawierajacego klucz pozwalający na autnetykację użytkownika
- **nazwa_uzytkownika** - nazwa użytkownika na AWS
- **adres_ip** - adres ip AWS
- **katalog** - katalog w którym zostanie zapisany plik z eksportem danych
- **nazwa_pliku** - nazwa powstałego pliku z eksportem danych

W celu zaimportowania danych do bazy danych należy użyć polecenia `impdp` posiadającego następującą strukturę:

```
impdp <nazwa_użytkownika>/<hasło>@<SID>  DUMPFILE=<katalog>:<nazwa_plik>
remap_schema=<nazwa_oryginalnego_schematu>:<nazwa_nowego_schematu>
```

- **nazwa_użytkownika,hasło** - dane autentykacyjne użytkownika lokalnej bazy danych

- **SID** - SID bazy danych, z której ma być wykonany eksport danych, parametr opcjonalny
- **katalog** - katalog w którym znajduje się importowany plik z danymi, katalog musi być zdefiniowany w bazie danych oraz użytkownik musi posiadać do niego prawa dostępu
- **nazwa_pliku** - nazwa pliku importowanego pliku
- **nazwa_oryginalnego_schematu** - nazwa schematu, z którego pochodzą tabele
- **nazwa_nowego_scheamtu** - nazwa schematu do którego mają zostać zaimportowane tabele

---

4. Automatyzacja procesu

Kroki zawarte w poprzednim punkcie zostały zautomatyzowane poprzez skrypt `export.py`. Wymagania środowiskowe:

- Python 3.6 lub nowszy
- biblioteka paramiko

Bibliotekę paramiko można zainstalować za pomocą narzędzia `pip`:

```
pip install paramiko
```

Skrypt `export.py` posiada następującą strukturę:

```
export.py -u <nazwa_użytkownika> -p <hasło> -f <nazwa_pliku>
 [-t <nazwy_tabel> | -s <nazwa_schematu>] -d <nazwa_katalogu>
```

- **nazwa_użytkownika** - nazwa użytkownika lokalnej bazy danych
- **hasło** - hasło użytkownika lokalnej bazy danych
- **nazwa_pliku** - nazwa pliku, który powstaje w wyniku działania skryptu, plik zawiera wyeksportowane dane
- **nazwy_tabel** - nazwy tabel rozdzielone przecinkami, które są przeznaczone do eksportu
- **nazwa_schematu** - nazwy schematów rozdzielone przecinkami, które są przeznaczone do eksportu
- **nazwa_katalogu** - bezwzględna ścieżka do katalogu w którym zostanie zapisany plik z eksportem danych.

  UWAGA!

  - Skrypt w trakcie działania definiuje w bazie danych katalog o nazwie `AWS_DIR` i ścieżce przekazanej do skryptu. Jeżeli taki katalog jest już zdefiniowany zostanie on nadpisany.
  - Eksport danych się nie powiedzie, jeżeli podany użytkownik nie posiada praw do tworzenia katalogów w lokalnej bazie danych.

export.py

```
import os
import sys
import getopt
from sys import stderr
import sys
import getopt
import paramiko
import time

ip = '54.246.212.140'
username = 'ec2-user'
key_filename = 'MK_keypair.pem'

def printTotals(transferred, toBeTransferred):
    print(f"\rTransferred {round(transferred/toBeTransferred*100)}%", end=" ")

def transfer_file_to_aws(directory, file_name):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    client.connect(ip,
                username=username,
                key_filename=key_filename)

    dump_filepath = os.path.join(directory, file_name)
    print("Transfering file to " + ip)
    ftp_client = client.open_sftp()
    ftp_client.put(dump_filepath, f'/home/ec2-user/oracle/{file_name}',
callback=printTotals)
    ftp_client.close()
    print("File transfered")

def line_buffered(f):
    line_buf = ""
    while not f.channel.exit_status_ready():
        line_buf += str(f.read(1))
        if line_buf.endswith('\n'):
            yield line_buf
            line_buf = ''

def exec(client, command):
    stdin, stdout, stderr = client.exec_command(command)

    while not stderr.channel.exit_status_ready() and not
stderr.channel.recv_ready():
       for line in iter(lambda: stderr.readline(2048), ""):
            print(line, end="")


def import_dump_on_aws(user,file_name):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    client.connect(ip,
                username=username,
```

```
                key_filename=key_filename)

    command = f'impdp student/student  directory=DP_SHARED_DIR dumpfile=
{file_name} TABLE_EXISTS_ACTION=REPLACE remap_schema={user}:student '
    exec(client, command)

def main(argv):
    user = ''
    password = ''
    dumpfile = ''
    tables = ''
    schemas = ''
    directory = ''
    log_file = ''
    try:
        opts, args = getopt.getopt(
            argv,
            "u:p:f:t:d:l:s:",
            ["user=", "password=", "file=", "tables=", "directory=", "log-file=",
"schemas="])
    except getopt.GetoptError:
        print("test.py -i <inputfile> -o <outputfile>")
        sys.exit(2)

    for opt, arg in opts:
        if opt in ("-u", "--user"):
            user = arg
        elif opt in ("-p", "--password"):
            password = arg
        elif opt in ("-f", "--file"):
            dumpfile = arg
        elif opt in ("-t", "--tables"):
            tables = arg
        elif opt in ("-s", "--schemas"):
            schemas = arg
        elif opt in ("-d", "--directory"):
            directory = arg
        elif opt in ("-l", "--log-file"):
            log_file = arg

    if(not user or not password):
        stderr.write("Invalid credentials")
        sys.exit(-1)

    if(tables and schemas):
        stderr.write("Cannot export tables and schemas in the same time")
        sys.exit(-1)

    if (not tables and not schemas):
        stderr.write("Must export something table or schema")
        sys.exit(-1)

    expdp_command = f"expdp {user}/{password} "
```

```
    if(tables):
        expdp_command += f"tables={tables} "
    else:
        expdp_command += f"schemas={schemas}"

    if(not directory):
        print("Directory is expected")

    os.system(f"echo CREATE OR REPLACE DIRECTORY AWS_DIR AS '{directory}'; |
sqlplus {user}/{password}")

    expdp_command += f"dumpfile=AWS_DIR:{dumpfile} "

    if(not log_file):
        expdp_command += f"NOLOGFILE=YES"
    else:
        expdp_command += f"logfile={log_file}"

    expdp_command += " Version=11.1"

    os.system(f'del {os.path.join(directory, dumpfile)}')
    os.system(expdp_command)
    transfer_file_to_aws(directory, dumpfile)
    import_dump_on_aws(user, dumpfile)

if __name__ == "__main__":
    main(sys.argv[1:])
```

---

5. Stworzenie tabel na AWS za pomocą procedur PL/SQL i wykonanie insertów (sposób niezrealizowany)

Sposób składa się z następujących etapów:

1. Przenisienie struktury tabel z lokalnej bazy danych na bazę danych na AWS za pomocą wywołania poleceń `CREATE TABLE`
2. Przeniesie danych z lokalnej bazy danych na bazę danych na AWS za pomocą poleceń `INSERT INTO`

W celu wykonania polecenia bazie danych AWS należy połączyć bazy pomiędzy sobą za pomocą dblink.

dblink.sql

```
CREATE DATABASE LINK aws
    CONNECT TO student IDENTIFIED BY student
    USING 'ip:port/SID';
SELECT * FROM CARD_ACCOUNT_REL@aws;
```

Procedura odwzorowujące strukturę lokalnej tabli na bazie danych na AWS. Procedura wymaga parametru będącego nazwą tabeli, która ma zostać odwzorowana.

create_table_on_aws.sql

```
CREATE OR REPLACE PROCEDURE export_table (
    table_name IN VARCHAR
) IS
    create_table   VARCHAR(32767);
BEGIN
dbms_metadata.set_transform_param (dbms_metadata.session_transform, 'PRETTY',
false);
dbms_metadata.set_transform_param(dbms_metadata.session_transform, 'EMIT_SCHEMA',
false);
dbms_metadata.set_transform_param(dbms_metadata.session_transform, 'TABLESPACE',
false);
    SELECT
        dbms_metadata.get_ddl('TABLE',UPPER(table_name))
    INTO
        create_table
    FROM
        dual;

    call_command_on_aws(create_table);
END;
```

Procedura odpowiedzialna za wywołanie komendy na AWS podanej w parametrze

call_command_on_aws.sql

```
CREATE OR REPLACE PROCEDURE call_command_on_aws (
    command IN VARCHAR
) IS
    v_cursor       NUMBER;
    v_ind          NUMBER;
BEGIN
    v_cursor := dbms_sql.open_cursor@aws;
    dbms_sql.parse@aws ( v_cursor, command, dbms_sql.native );
    v_ind := dbms_sql.execute@aws ( v_cursor );
END;
```

Nie jest możliwe wywołanie poleceń DDL za pomocą dblink co znacząco utrudnia wykonanie zadania za pomocą PL/SQL. Z tego powodu należy ręcznie budować każde polecenie poprzez konkatenację łańcuchów znaków. Oracle wspiera jedynie generowanie poleceń `CREATE TABLE`, natomiast nie wspiera generowania poleceń `INSERT INTO`. Stworzenie własnego generatora takich poleceń na podstawie danych zawartych w bazie jest bardzo trudne. Jednocześnie takie rozwiązanie nie zapewnia odpowiedniej szybkości działania.

Z powodu powyższych problemów realizacja projektu za pomocą tego rozwiązania została porzucona.