

# Configuration Manual

MSc Research Project  
Cloud Computing

Achyut Gawade  
Student ID: 22103228

School of Computing  
National College of Ireland

Supervisor: Prof. Sean Heeney

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Achyut Gawade
<b>Student ID:</b>	22103228
<b>Programme:</b>	Cloud Computing
<b>Year:</b>	2023
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Prof. Sean Heeney
<b>Submission Due Date:</b>	14/12/2023
<b>Project Title:</b>	Configuration Manual
<b>Word Count:</b>	710
<b>Page Count:</b>	10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

<b>Signature:</b>	
<b>Date:</b>	14th December 2023

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission</b> , to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project</b> , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Achyut Gawade  
22103228

## 1 AWS Cloud Account Setup

A working AWS cloud account is needed to perform below steps with access to IAM services. It is advisable to create a separate AWS IAM role which shall be used for interacting with the services later.

- Create new IAM role let's say `vmimport` in this case.
- Add inline policy as described in below text and attach to this IAM role.

```
Policy.Json
{
    "Effect": "Allow",
    "Action": [
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject",
        "s3:GetBucketAcl"
    ],
    "Resource": [
        "arn:aws:s3:::final-bucket-ac",
        "arn:aws:s3:::final-bucket-ac/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "ec2:ModifySnapshotAttribute",
        "ec2:CopySnapshot",
        "ec2:RegisterImage",
        "ec2:Describe*"
    ],
    "Resource": "*"
}
```

- Fetch the `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` from the IAM module. Additionally the `AWS_SESSION_TOKEN` in case of assumed role. This will be needed to perform operations from python boto3 SDK with AWS services as shown in Figure 1.

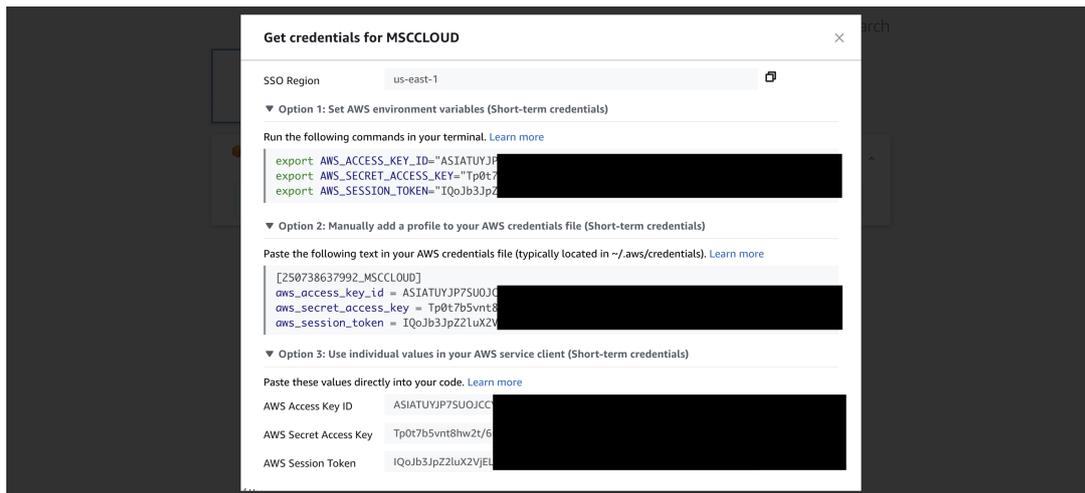


Figure 1: Azure Cloud Account Application Credential Details

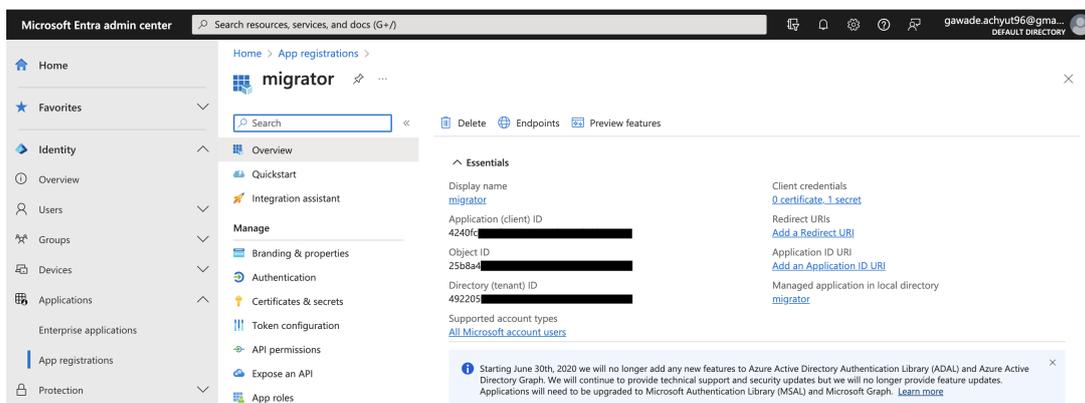


Figure 2: Azure Cloud Account Application Credential Details

- Spin-up a testbed EC2 instance for running the developed middleware with Ubuntu image.

## 2 Azure Cloud Account Setup

- Visit <https://entra.microsoft.com/> and register an application
- Fetch client\_id, client\_secret, tenant\_id, subscription\_id from the registered application as shown in the Figure 2
- In Azure cloud account, go to the subscription service and add the registered application to grant access to the subscription resources from IAM module.

## 3 GCP Cloud Account Setup

- Visit <https://console.cloud.google.com/> and create an application

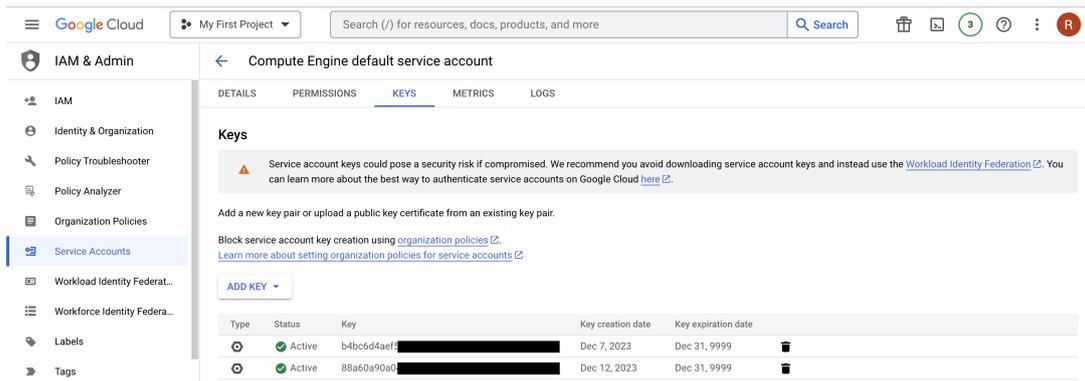


Figure 3: GCP Cloud Service Account Details

- Go to "IAM And Admin" section and generate a service account with access to Compute, Storage and Database services as shown in Figure 3. A Json file will be downloaded automatically which shall be used in next sections.

## 4 Resource Discovery Module Setup

- On the test server, update the server packages eg. `sudo apt update`.
- Install Terraform on testbed server as per the OS <sup>1</sup>
- In `main.tf` file add all the cloud account details fetched from Section 1, Section 2 and Section 3 such as access keys, token, client secret and `serviceAccount.json` etc. in the fields shown in Figure 4
- In `AzureDiscovery.py` add the Azure account details from Section 2 as per Figure 5
- In `AWSDiscovery.py` add the AWS account details from Section 1 as per Figure 6
- In `GCPDiscovery.py` add the path to the `serviceAccount.json` file obtained in 3 as per Figure 6
- Below are the python libraries that needs to be configured which are mentioned in `requirements.txt` file as well.
  - `azure.common.credentials`
  - `azure.mgmt.resource`
  - `boto3`
  - `google.cloud`

<sup>1</sup><https://developer.hashicorp.com/terraform/install>

```

1 # Configure the Cloud providers
2
3 terraform {
4   required_providers {
5     aws = {
6       source = "hashicorp/aws"
7       version = "~> 3.0"
8     }
9     azurem = {
10      source = "hashicorp/azurerm"
11      version = "~> 3.0.2"
12    }
13    google = {
14      source = "hashicorp/google"
15      version = "~> 4.0"
16    }
17  }
18  required_version = ">= 1.1.0"
19 }
20
21 provider "aws" {
22   region = ""
23   access_key = ""
24   secret_key = ""
25   token = ""
26 }
27
28 provider "azurerm" {
29   features {}
30   client_id = ""
31   client_secret = ""
32   tenant_id = ""
33   subscription_id = ""
34 }
35
36 provider "google" {
37   credentials = file("path/to/gcp-credentials.json")
38   project = ""
39   region = ""
40 }
41
42 resource "azurerm_virtual_machine" "myVM" {
43   name = ""
44   location = ""

```

Figure 4: Terraform Cloud Provider Configurations

```

31
32 if __name__ == "__main__":
33     subscription_id = "a
34     tenant_id = "4467265
35     client_id = "5ce352b
36     secret_value = "pt18
37
38     azure_manager = AzureResourceManager(subscription_id, tenant_id, client_id, secret_value)
39
40
41 # List resources of a specific type (e.g., virtual machines)
42 resource_type_vm = 'Microsoft.Compute/virtualMachines'
43 resource_type_disk = 'Microsoft.Compute/disks'
44 resource_type_snapshot = 'Microsoft.Compute/snapshots'
45 resource_type_network_interfaces = 'Microsoft.Network/networkInterfaces'
46 resource_type_storage = 'Microsoft.Storage/storageAccounts'
47 resource_type_sql_DBforPostgreSQL = 'Microsoft.DBforPostgreSQL/servers'
48 resource_type_flexible_DBforPostgreSQL = 'Microsoft.DBforPostgreSQL/flexibleServers'
49 resource_type_sql_AzureData = 'Microsoft.AzureData/servers'
50 resource_type_sql_DBforMariaDB = 'Microsoft.DBforMariaDB/servers'
51 resource_type_sql_DBforMySQL = 'Microsoft.DBforMySQL/servers OR Microsoft.DBforMySQL/flexibleServers'
52 resource_type_flexible_DBforMySQL = 'Microsoft.DBforMySQL/flexibleServers'
53 resource_type_sql = 'Microsoft.Sql/servers'
54 resource_type_SqlVirtualMachine = 'Microsoft.SqlVirtualMachine/servers'
55

```

Figure 5: Azure Discovery Account Configuration

```

1 import boto3
2 import subprocess
3
4 # AWS credentials and region
5 aws_access_key = "ASI
6 aws_secret_key = "ont
7 AWS_SESSION_TOKEN=""I
8 aws_region = "eu-west-1"
9
10 # Initialize AWS clients
11 ec2_client = boto3.client('ec2', aws_access_key_id=aws_access_key, aws_secret_access_key=aws_secret_key, aws_session_token=AWS_SESSION_TOKEN, region_name=aws_region)
12 rds_client = boto3.client('rds', aws_access_key_id=aws_access_key, aws_secret_access_key=aws_secret_key, aws_session_token=AWS_SESSION_TOKEN, region_name=aws_region)
13 s3_client = boto3.client('s3', aws_access_key_id=aws_access_key, aws_secret_access_key=aws_secret_key, aws_session_token=AWS_SESSION_TOKEN, region_name=aws_region)
14

```

Figure 6: AWS Discovery Account Configuration

## 5 Virtual Machine Migration Module Setup

- Install Aria2<sup>2</sup> download utility on the test server.
- Install Qemu and Qemu-img<sup>3</sup> which is used for disk format conversions as shown in the Figure 8
- open .env file and set the required parameters such as account credentials, bucket details, IAM role created in Section 1 etc. as shown in Figure 7
- This module primarily contains 6 runnable python files which are azure\_gcp\_mgr.py, aws\_gcp\_mgr.py, gcp\_aws\_mgr.py, gcp\_azure\_mgr.py, azure\_aws\_mgr.py and aws\_azure\_mgr.py.
- Below are the python libraries that needs to be installed.
  - boto3
  - azure-identity
  - azure-storage-blob
  - azure-mgmt-compute
  - azure-mgmt-network
  - azure-mgmt-storage
  - azure-mgmt-resource
  - google-cloud
  - google-auth
  - google-cloud-build
  - google-cloud-storage
  - google-cloud-compute
  - python-dotenv

## 6 Database Migration Module Setup

- On the test server, update the server packages eg. `sudo apt update`
- Install MySQL on the server<sup>4</sup>
- Install postgresql-client or postgresql on the server<sup>5</sup>
- Start the MySQL service by running `sudo systemctl start mysql`
- Start the postgresql service.

---

<sup>2</sup><https://aria2.github.io/>

<sup>3</sup><https://www.qemu.org/download/>

<sup>4</sup><https://www.mysql.com/downloads/>

<sup>5</sup><https://www.postgresql.org/download/>

```
.env x
Users > acgawade > workspace > Thesis Final Submission > Artifacts > Cloud-VM-Migrations > .env
1 # AWS S3 Storage credentials
2 S3_BUCKET_NAME = ''
3 AWS_ACCESS_KEY_ID = ''
4 AWS_SECRET_ACCESS_KEY = ''
5 AWS_SESSION_TOKEN = ''
6 AWS_ROLE_NAME = ''
7 AWS_SECURITY_GROUP = ''
8
9
10 # Google Cloud Credentials
11 SERVICE_ACCOUNT_INFO = {}
12 GCS_BUCKET_NAME = ''
13 GCP_PROJECT_ID = ''
14
15
16 # Azure
17 RESOURCE_GROUP_NAME = ''
18 VM_NAME = ''
19 STORAGE_ACCOUNT_NAME = ''
20 CONTAINER_NAME = ''
21 STORAGE_CONTAINER_URI = ''
22 SUBSCRIPTION_ID = ''
23 SECRET = ''
24 CLIENT_ID = ''
25 TENANT_ID = ''
```

Figure 7: env file for VM Migration Configuration

```
fileconv.py x
Users > acgawade > workspace > Thesis Final Submission > Artifacts > Cloud-VM-Migrations > fileconv.py > ...
1 #from utils.download import get_size
2 import subprocess
3 import json
4
5 def get_size(file_path:str):
6     command = ["qemu-img", "info", "-f", "raw", "--output", "json", f"{file_path}"]
7     result = subprocess.run(command, capture_output=True, text=True)
8     if result.returncode == 0:
9         result = json.loads(result.stdout)
10        return result.get('virtual-size')
11    else:
12        error_message = result.stderr
13        print("Error:", error_message)
14
15 def resize_image(path:str, vhd_path:str):
16
17     convert_command = ["qemu-img", "convert", "-f", "raw", "-o", "subformat=fixed,force_size", "-0", "vpc", f"{path}", f"{vhd_path}"]
18     size = get_size(path)
19
20     MB = 1024 * 1024
21     rounded_size = ((size // MB) + 1) * MB
22
23     if size % MB == 0:
24         result = subprocess.run(convert_command, capture_output=True, text=True)
25     else:
26         resize_command = ["qemu-img", "resize", "-f", "raw", f"{path}", f"{rounded_size}"]
27
28
29
30 get_size('gcpt45e3o.raw')
```

Figure 8: env file for VM Migration Configuration

```

1 from sqlalchemy import NullPool, create_engine, MetaData
2 import pandas as pd
3 import pyodbc
4 import mysql.connector
5
6 dest_db_engines = []
7
8 host = ""
9 port = 3306
10 user = ""
11 password = ""
12 database = ""
13
14
15 target_db_host = ""
16 target_db_port = 3306
17 target_db_user = ""
18 target_db_password = ""
19 target_db_name = ""
20
21
22 def getEngine( db_name, user, password, host):
23     connection_string = "mysql+mysqlconnector://{user}:{password}@{host}/{db_name}"
24     return create_engine(connection_string, echo=True)
25
26 def getSourceDb():
27     global sourceEngine
28     # print("Please enter the source database name: ")
29     # source_db_name = input()
30     # print("Please enter the source database user: ")
31     # source_db_user = input()
32     # print("Please enter the source database password: ")
33     # source_db_password = input()
34     # print("Please enter the source database host: ")
35     # source_db_host = input()
36     # sourceEngine = getEngine(source_db_name, source_db_user, source_db_password, source_db_host)
37
38     sourceEngine = getEngine(database, user, password, host)
39
40
41
42 def getDestinationEngines():
43     # n = input("Please number of destination databases: ")
44     # m =

```

Figure 9: env file for VM Migration Configuration

- This module contains four executable python files which are for performing bulk migration, synchronisation, fetching the database size and verifying the hash of table data as shown in Figures 11, Figure 9 and Figure 12 and Figure 10 respectively.
- This module requires below python libraries
  - pandas
  - sqlalchemy
  - mysql.connector
  - pyodbc

## 7 Storage Migration

- Add AWS, Azure, GCP account details in StorageMigrator.py as shown in the Figure 13
- Add bucket names for all the cloud buckets along with mentioning the migration scenario and setting DELETE\_AFTER\_TRANSFER flag
- This module also contains a HashVerifier.py file which can be used to verify the accuracy of files migrated from source to destination cloud bucket as shown in 14.
- Below are the python libraries that needs to be configured which are also provided in requirements.txt
  - azure.common.credentials

```

synchronizer_sql.py  verify_hash.py X
Users > acgawade > workspace > Thesis Final Submission > Artifacts > DBMigration > verify_hash.py > ...
1 import hashlib
2 import mysql.connector
3 from sqlalchemy import MetaData, create_engine
4
5 # Source DB credentials
6 host = ""
7 port = 3306
8 user = ""
9 password = ""
10 database = ""
11
12 # Destination DB credentials
13 target_db_host = ""
14 target_db_port = 3306
15 target_db_user = ""
16 target_db_password = ""
17 target_db_name = ""
18
19
20 def fetchAllTables(user,password,host,db_name):
21     connection_string = f"mysql+mysqlconnector://{user}:{password}@{host}/{db_name}"
22     source_engine = create_engine(connection_string, echo=True)
23     source_metadata = MetaData()
24     source_metadata.reflect(bind=source_engine)
25
26     table_names = source_metadata.tables.keys()
27
28     return table_names
29
30
31
32
33 def verifyHash():
34     source_tables = fetchAllTables(user,password,host,database)
35     #dest_tables = fetchAllTables(target_db_user,target_db_password,target_db_host,target_db_name)
36
37     db_source = mysql.connector.connect(
38         host=host,
39         user=user,
40         password=password,
41         database=database)
42
43     source_cursor = db_source.cursor()
44
45     db_dest = mysql.connector.connect(

```

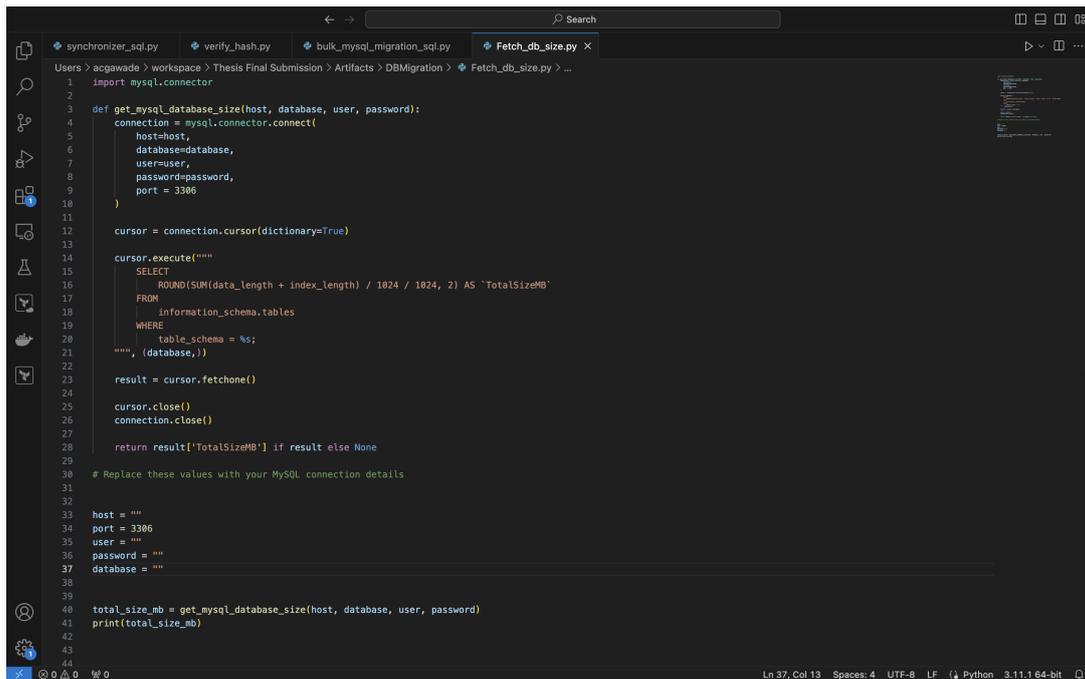
Figure 10: env file for VM Migration Configuration

```

bulk_mysql_migration_sql.py X
Users > acgawade > workspace > Thesis Final Submission > Artifacts > DBMigration > bulk_mysql_migration_sql.py > ...
1 import os
2 import time
3
4
5 start_time = time.time()
6
7 # Source Database connection details
8
9 host = ""
10 port = 3306
11 user = ""
12 password = ""
13 database = ""
14
15 target_db_host = ""
16 target_db_port = 3306
17 target_db_user = ""
18 target_db_password = ""
19 target_db_name = ""
20
21
22
23 fileName = "mydb.sql"
24
25
26 def importAzureDump(host,user,database,fileName,password):
27     command = f"mysqldump -h {host} -u {user} -p{password} {database} > {fileName}"
28     os.system(command)
29
30 def exportAzureDump(host,user,database,fileName,password):
31     command = f"mysql -u {user} -f -p{password} -h {host} {database} < {fileName}"
32     os.system(command)
33
34 def importAWSDump(host,user,database,port,fileName,password):
35     command = f"mysqldump -u {user} -p{password} --host={host} --port={port} {database} > {fileName}"
36     os.system(command)
37
38 def exportAWSDump(host,user,database,port,fileName,password):
39     command = f"mysql -u {user} -f -p{password} -h {host} -P {port} {database} < {fileName}"
40     os.system(command)
41
42 importAzureDump(host,user,database,fileName,password)
43 exportAWSDump(target_db_host,target_db_user,target_db_name,target_db_port,fileName,target_db_password)
44
45

```

Figure 11: env file for VM Migration Configuration



```
1 import mysql.connector
2
3 def get_mysql_database_size(host, database, user, password):
4     connection = mysql.connector.connect(
5         host=host,
6         database=database,
7         user=user,
8         password=password,
9         port = 3306
10    )
11
12    cursor = connection.cursor(dictionary=True)
13
14    cursor.execute("""
15        SELECT
16            ROUND(SUM(data_length + index_length) / 1024 / 1024, 2) AS `TotalSizeMB`
17        FROM
18            information_schema.tables
19        WHERE
20            table_schema = %s;
21    """,
22        (database,))
23    result = cursor.fetchone()
24
25    cursor.close()
26    connection.close()
27
28    return result['TotalSizeMB'] if result else None
29
30 # Replace these values with your MySQL connection details
31
32
33 host = ""
34 port = 3306
35 user = ""
36 password = ""
37 database = ""
38
39
40 total_size_mb = get_mysql_database_size(host, database, user, password)
41 print(total_size_mb)
42
43
44
```

Figure 12: env file for VM Migration Configuration

- azure.mgmt.resource
- boto3
- google.cloud

```

StorageMigrator.py x
Users > acgawade > workspace > Thesis Final Submission > Artifacts > StorageMigration > StorageMigrator.py > ...
1 import os
2 import boto3
3 import botocore
4 import time
5 from google.cloud import storage
6 from azure.storage.blob import BlobServiceClient
7 import subprocess
8 AWS_ACCESS_KEY_ID=""
9 AWS_SECRET_ACCESS_KEY=""
10 AWS_SESSION_TOKEN=""
11 AWS_BUCKET_NAME=""
12 GCP_SERVICE_ACCOUNT_JSON=os.path.join('./', "serviceAccount.json")
13 GCP_BUCKET_NAME=""
14 AZURE_CONNECTION_STRING=""
15 AZURE_CONTAINER_NAME=""
16
17 # Configuration Options, set to 1 to enable
18 AWS_TO_GCP = 1
19 AWS_TO_AZURE = 0
20 GCP_TO_AWS = 0
21 GCP_TO_AZURE = 0
22 AZURE_TO_AWS = 0
23 AZURE_TO_GCP = 0
24
25 # Do all your testing until then don't delete the bucket, set to 1 to enable
26 DELETE_AFTER_TRANSFER = 0

```

Figure 13: Storage Migration Accounts And Bucket Configurations

```

HashVerifier.py x
Users > acgawade > workspace > Thesis Final Submission > Artifacts > StorageMigration > HashVerifier.py > ...
1 import boto3
2 import hashlib
3 from google.cloud import storage
4 from azure.storage.blob import BlobServiceClient, BlobClient
5 import os
6
7 # AWS setup - replace with your actual details
8 AWS_ACCESS_KEY_ID = ''
9 AWS_SECRET_ACCESS_KEY = ''
10 AWS_SESSION_TOKEN = ''
11 AWS_BUCKET_NAME = ''
12
13 # Azure setup - replace with your actual details
14 AZURE_CONNECTION_STRING = ''
15 AZURE_CONTAINER_NAME = ''
16
17 # GCP setup - replace with your actual details
18 GCP_SERVICE_ACCOUNT_JSON = os.path.join('./', "serviceAccount.json")
19 GCP_BUCKET_NAME = ''
20
21 # Function to calculate MD5 hash of file content
22 def calculate_md5(file_content):
23     hash_md5 = hashlib.md5()
24     hash_md5.update(file_content)
25     return hash_md5.hexdigest()
26
27 # Initialize AWS S3 client
28 aws_session = boto3.Session(
29     aws_access_key_id=AWS_ACCESS_KEY_ID,
30     aws_secret_access_key=AWS_SECRET_ACCESS_KEY,
31     aws_session_token=AWS_SESSION_TOKEN,
32 )
33
34 s3_client = aws_session.client('s3')
35
36 # Initialize Azure Blob Service Client
37 azure_service_client = BlobServiceClient.from_connection_string(AZURE_CONNECTION_STRING)
38 azure_container_client = azure_service_client.get_container_client(AZURE_CONTAINER_NAME)
39
40 # Initialize GCP client
41 gcp_client = storage.Client.from_service_account_json(GCP_SERVICE_ACCOUNT_JSON)
42 gcp_bucket = gcp_client.bucket(GCP_BUCKET_NAME)
43
44 # Get list of all objects in AWS bucket and calculate MD5 hashes

```

Figure 14: Hash Verifier Programm for Storage Migration