

Configuration Manual

MSc Research Project Cloud Computing

Devaswaroopa Machenhalli Nandish Student ID: 22169245

School of Computing National College of Ireland

Supervisor: Dr Ahmed Makki

National College of Ireland Project Submission Sheet School of Computing



Student Name:	Devaswaroopa Machenhalli Nandish
Student ID:	22169245
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr Ahmed Makki
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	382
Page Count:	11

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.

<u>ALL</u> 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.

Signature:	Devaswaroopa Machenahalli Nandish
Date:	15th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

 Attach a completed copy of this sheet to each project (including multiple copies).
 □

 Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).
 □

 You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep
 □

a copy on computer.

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

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Devaswaroopa Machenhalli Nandish 22169245

1 Introduction

The configuration manual demonstrates running of the integrated code. The code should be properly configured as aligned in the configuration manual to successfully observe the results.

1.1 Requirement

Before commencing the setup process, verify that the following requirements have been fulfilled:

- Possession of data in CSV format.
- An active AWS account.
- Access to SageMaker with Compute-Optimized instances.
- Proficiency in Boto 3 and Python.
- Install Pycharm , NymPy, argon2-cffi, psutil, and matplotlib libraries
- Optionally, contemplate the use of VScode for future tasks.

2 Dependency Installations

- 1. Install "pip install package-name" command
- 2. Install "Pip Install pandas"
- 3. Install "pip install matplotlib"

3 Implementation

1. Log in to your AWS Account and Go to AWS S3 buckets

🔲 🛸 eu-north-1.console.aws.an	hazon.com/console/home?region=eu-north-1#	🖒 🐶 🔺	
	[Alt+S]	D 4	⑦
Console Home 1160		Reset to default layout + Add widgets	© ©
# Recently visited Info	÷	Applications (0) Info Create application : Region: Europe (Stockholm)	
Amazon SageMaker	😥 RDS		
S3	GodePipeline	eu-north-1 (Current Region) 🔻 Q Find applications	
EE IAM	Cloud9	< 1 >	
CodeDeploy	Telastic Beanstalk	Name ▲ Description ▼ Region ▼ Originating account	
G VPC	AWS User Notifications		
		No applications Get started by creating an application.	
CodeBuild		Create application	
View a	Il services	Go to myApplications	
:: Welcome to AWS :		# Cost and usage Info :	
Getting started with AVS 22 Arean the fundamentals and find valuable information to get the most out of AWS. Training and Certification 22 Learn from AVS experts and advance your skills and knowledge.	Open issues O Past 7 days Scheduled changes O Upcoming and past 7 days Other notifications 1 Past 7 days	No cost and usage data Vou haven't configured AWS Cost Explorer or you do not have permission. Turn on Cost Explorer	
	Console Home Info Recently visited Info Recently visited Info Recently visited Info Amazon SageMaker S S UM CodeDeploy CodeDeplo	Console Home Intel CodeSubset Annual Program Control 4 CodeS Program Control 4 CodeS Program Control 4 CodeS CodeS Program CodeS	Console Home into

Figure 1: AWS Console Page

2. Next step Create S3 Bucket for storing csv files.

WS Services Q Search	[Alt	+S]		D & 0 @) Global 🔻 dev
Amazon S3 ×	Amazon 53				
Buckets Access Grants New	Account snapshot Storage lens provides visibility into storage usage and activity	vity trends. <u>Learn more</u> 🖸		View Storage	e Lens dashboard
Object Lambda Access Points Multi-Region Access Points Batch Operations	General purpose buckets Directory buckets				
IAM Access Analyzer for S3	General purpose buckets (1) Info Buckets are containers for data stored in 53. Learn more		C	🗇 Copy ARN Empty Delete	Create bucket
Block Public Access settings for this account	Q Find buckets by name				< 1 > ©
Storage Lens	Name	AWS Region		▼ Creation date	∇
Dashboards	O ecl-gwo	Europe (Ireland) eu-west-1	Objects can be public	December 8, 2023, 09:54:28 (I	JTC+00:00)
AWS Organizations settings Feature spotlight					
AWS Marketplace for S3					

Figure 2: AWS S3 Bucket

3. Upload the CSV files to the s3 bucket.

aws iii Services Q Search		[Alt+S]		B \$ @	⊚ Global ▼ deva
Amazon S3 ×	Amazon S3 > Buckets > ecl-gwo > yahoo_pa	ssword_frequencies_corpus/			
Buckets	yahoo_password_frequent	cies_corpus/			🗇 Copy S3 URI
Access Grants New Access Points Object Lambda Access Points	Objects Properties				
Multi-Region Access Points Batch Operations IAM Access Analyzer for S3	Objects (53) Info Objects are the fundamental entities stored in Amazon S	3. You can use Amazon 53 inventory [2	to get a list of all objects in your bucket. For others to access your object	ts, you'll need to explicitly grant them permissions. Learn more	
Block Public Access settings for this account	Q Find objects by prefix		pen E Delete Actions • Create rolleer		< 1 > @
Storage Lens	Name 🔺	Туре	▼ Last modified ▼	Size 🔻 Storage cla	iss 🗸 🗸
Dashboards Storage Lens groups			December 8, 2023, 10:18:20 (UTC+00:00)	97.0 B Standard	
AWS Organizations settings			December 8, 2023, 10:18:21 (UTC+00:00)	1.2 KB Standard	
Feature spotlight	yahoo-activity-login30d.txt	txt	December 8, 2023, 10:18:23 (UTC+00:00)	11.1 KB Standard	
AWS Marketplace for S3	yahoo-activity-login90d.txt	txt	December 8, 2023, 10:18:24 (UTC+00:00)	4.6 KB Standard	
	yahoo-activity-pw-change-count- _1.txt	txt	December 8, 2023, 10:18:30 (UTC+00:00)	3.4 KB Standard	
	yahoo-activity-pw-change-count- _5.txt	txt	December 8, 2023, 10:18:32 (UTC+00:00)	738.0 B Standard	
	yahoo-activity-pw-change-count- 0.txt	txt	December 8, 2023, 10:18:26 (UTC+00:00)	15.2 KB Standard	
	yahoo-activity-pw-change-count- 1.txt	txt	December 8, 2023, 10:18:28 (UTC+00:00)	4.4 KB Standard	

Figure 3: Adding CSV files to S3 Bucket

4. Setup Policies for User

aws Services Q Search	[Alt+S]	ב 🎝 🖓 🎯 Global ▼ bhuvanp9_6 א
Identity and Access ×	IAM > Policies	(
Q Search IAM	Policies (1/1165) Info C	Actions Delete Create policy
Dashboard	Q s3 X All types	▼ 12 matches < 1 > ⊚
Access management	Policy name ▲ Type ▼ Used as	▼ Description
User groups	○	Provides access to manage S3 settings
Users Roles	• If AmazonS3FullAccess AWS managed None	Provides full access to all buckets via t
Policies	○ ● ● AmazonS3ObjectL AWS managed None	Provides AWS Lambda functions permi
Identity providers	○ • • • AmazonS3Outpost AWS managed None	Provides full access to Amazon S3 on .
Account settings	○	Provides read only access to Amazon S
Access reports	O AmazonS3ReadOn AWS managed None	Provides read only access to all bucke
External access	O T AWSBackupService AWS managed None	Policy containing permissions necessar

Figure 4: Set Policies to S3 bucket

5. Create notebook instance to upload the .ipynb and dataset files as shown

aws iii Services Q Search		[Alt+S]				D 4 0 4	Stockholm 🔻	devaswaroopa
^	Amazon SageMaker > Notebook in	stances						
Admin configurations Domains Role manager	Notebook instances Info					C Actions V	Create notebook	instance
Images Lifecycle configurations	Name	Instance	Creation time	▼ Status	⊽	Actions		
SageMaker dashboard	O GWO-AWS	ml.t3.medium	12/8/2023, 10:23:26 AM	⊘ InServi	ce	Open Jupyter Open JupyterLab		
JumpStart Foundation models Computer vision models Natural language processing models								
Governance								
HyperPod Clusters								
Notebook Notebook instances Git repositories								
Processing								
Training								
▶ Inference								
AWS Marketplace								
Tutorials								
CloudShell Feedback					© 20:	23, Amazon Web Services, Inc. or its affiliates.	Privacy Terms	Cookie preferen



4 Python Libraries

- 1. Pip Install all the dependencies
- 2. Let us import the necessary libraries.

```
In [28]: import os
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from argon2 import PasswordHasher
import numpy as np
import time
import psutil
import psutil
import matplotlib.pyplot as plt
```

Figure 6: Import the python library

3. Run the PBKDF2 and Argon2 notebook after configuring our notebook.

```
pbk_df2 = []
arg_on2 = []
cpu_usage_list = []
memory_usage_list = []
# Load password dataset
def load_password_dataset(file_path):
    with open(file_path, 'r') as file:
        lines = file.readlines()
        passwords = [line.split()[0] for line in lines]
    return passwords
# Step 3: Create cryptographic keys using PBKDF2 and Argon2
def generate_pbkdf2_key(password, salt, iterations=100000):
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        iterations=iterations,
        length=32,
        salt=salt,
        backend=default_backend()
    key = kdf.derive(password.encode())
    return key
def generate argon2 key(password):
    ph = PasswordHasher()
    key = ph.hash(password)
    return key.encode() # Convert the key to bytes
# Step 4: Implement encryption method and key generation
def encrypt_data(data, key):
    iv = os.urandom(16)
    cipher = Cipher(algorithms.AES(key[:32]), modes.CFB(iv), backend=default_backend())
    encryptor = cipher.encryptor()
    ciphertext = encryptor.update(data) + encryptor.finalize()
    return ciphertext, iv
def decrypt_data(ciphertext, key, iv):
    cipher = Cipher(algorithms.AES(key[:32]), modes.CFB(iv), backend=default_backend())
    decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext) + decryptor.finalize()
    return plaintext
```

Figure 7: PBKDF2 and Argon2

4. Visualisation of the reuslts of PBKDF2 And Argon2



Figure 8: PBKDF2 & Argon2 - Encrypt & Decrypt.



Figure 9: PBKDF2 & Argon2 - CPU&Memory Utilization.



Figure 10: PBKDF2 & Argon2 - Key Space.

5. Run the ECL-GWO with key generated by PBKDF2/Argon2 keys notebook after configuring our notebook.

```
# Step 3: Create cryptographic keys using PBKDF2 and Argon2
def generate pbkdf2 key(password, salt):
    if isinstance(password, str):
       password = password.encode() # Convert to bytes if it's a string
   kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
       iterations=100000, # You can adjust the number of iterations
       length=32,
       salt=salt,
       backend=default backend()
   key = kdf.derive(password)
   return key
def generate_argon2_key(password):
   ph = PasswordHasher()
   key = ph.hash(password)
   return key.encode() # Convert the key to bytes
# Enhanced Cryptographic Layer with Grey Wolf Optimization (ECL-GWO) key generation
def ecl_gwo_key_generation(password, salt):
    # Grey Wolf Optimization (GWO) algorithm for key modification
   def gwo_algorithm(key):
       # For GWO Modification
       modified_key = bytearray(key) + bytearray(os.urandom(len(key))) # ECL-GWD modification
       return bytes(modified_key)
   # Generate the original key using PBKDF2
   original_key = generate_pbkdf2_key(password, salt)
   # Apply Grey Wolf Optimization (GWO) to enhance the key
   enhanced_key = gwo_algorithm(original_key)
   return enhanced_key
# Implement encryption method with ECL-GWO key
def encrypt_data(data, key):
   iv = os.urandom(16)
   # Apply ECL-GWO enhanced key to the encryption process
   ecl_gwo_key = ecl_gwo_key_generation(key, iv)
   cipher = Cipher(algorithms.AES(ecl_gwo_key[:32]), modes.CFB(iv), backend=default_backend())
```

Figure 11: ECL-GWO Code with PBKDF2/Argon 2 Keys in notebook



6. Visualisation of the reuslts of ECL-GWO with PBKDF2/Argon 2 keys





Figure 13: ECL-GWO PBDFK2 - CPU&Memory Utilization.



Figure 14: ECL-GWO along with PBKDF2 Key Generation size.

7. Run the ECL-GWO with key generated independently notebook after configuring our notebook.

```
# Step 3: Create cryptographic keys using PBKDF2 and Argon2
def generate_pbkdf2_key(password, salt):
     if isinstance(password, str):
          password = password.encode() # Convert to bytes if it's a string
     kdf = PBKDF2HMAC(
          algorithm=hashes.SHA256(),
          iterations=100000, # You can adjust the number of iterations
          length=32.
          salt=salt,
          backend=default_backend()
     key = kdf.derive(password)
     return key
def generate_argon2_key(password):
    ph = PasswordHasher()
     key = ph.hash(password)
     return key.encode() # Convert the key to bytes
# Enhanced Cryptographic Layer with Grey Wolf Optimization (ECL-GWO) key generation
def ecl_gwo_key_generation(password, salt):
    # Grey Wolf Optimization (GWO) algorithm for key modification
     def gwo_algorithm(key):
          gwo_algorid(key):
# Assuming GNO modifies the key
modified_key = bytearray(key) + bytearray(os.urandom(len(key)))
          return bytes(modified_key)
     # Generate the original key using PBKDF2
     original_key = generate_pbkdf2_key(password, salt)
    # Apply Grey Wolf Optimization (GWO) to enhance the key
enhanced_key = gwo_algorithm(original_key)
     return enhanced_key
```

Figure 15: ECL-GWO Code with Indenepend key Generation in notebook

8. Visualisation of the reuslts of ECL-GWO with key generated independently notebook after configuring our notebook.



Figure 16: ECL-GWO along with PBKDF2 and Argon2 with independent Key Generation. .