Configuration Manual

MSc Research Project
Master of Science in Cloud Computing

Anurag Singh
Student ID: x23180013

School of Computing
National College of Ireland

Supervisor: Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anurag Singh
Student ID:	x23180013
Programme:	Master of Science in Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Aqeel Kazmi
Submission Due Date:	29/01/2025
Project Title:	Configuration Manual
Word Count:	1632
Page Count:	10

Signature:	
Date:	29th January 2025

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

Configuration Manual

Anurag Singh
x23180013

1 Introduction

The Cloud Access Machine Learning System is developed to predict security scores based on cloud access policies and the framework used. It aims to offer actionable remediations of the cloud access control configuration to improve security. This thesis features a Flask-based backend API that uses the machine learning-trained model as a pickle module to predict and suggest, ensuring efficient and reliable communication between the application and the machine learning models (Grinberg; 2018). In addition, it features a user-friendly web interface that allows for easy interaction with the model's capabilities. Its predictive and remediation insights are built around the core of the system, which are pre-trained machine learning models.

2 System Requirements

Hardware, software, and the infrastructure needed to design, develop, and implement the proposed solution must be defined within system requirements. For this research thesis, the system requirements typically fall into the following categories:

2.1 Hardware Requirements

Category	**Details**
Minimum	• CPU: Dual-core processor • RAM: 4 GB • Storage: 2 GB free space
Recommended	• CPU: Quad-core processor • RAM: 8 GB or higher • Storage: 5 GB free space • GPU: Optional (for retraining the model)

2.2 Software Requirements

Operating System	• Windows 10/11, macOS, or Linux/Ubuntu
Languages and Frameworks	• Python 3.9 or higher • Flask 2.0 or higher
Libraries	• Scikit-learn • Pandas • Numpy • Flask • Gunicorn • boto3 • seaborn • matplotlib • xgboost • pyyaml
AWS Services	• Cloud9 • S3 (Simple Storage Service) • Elastic Beanstalk • CodePipeline
Additional Tools	• Postman (API testing) • Browser (Chrome/Firefox/Edge) • GitHub (Repository & Versioning)

Using AWS service like S3, Cloud9, and Elastic Beanstalk enhance scalability and ease of deployment (Services; 2024)

3 Installation Instructions

3.1 Setting up the Environment

Go to the GitHub repository of this thesis through the link given below:

```
https://github.com/Annurag99/CloudAccessML
```

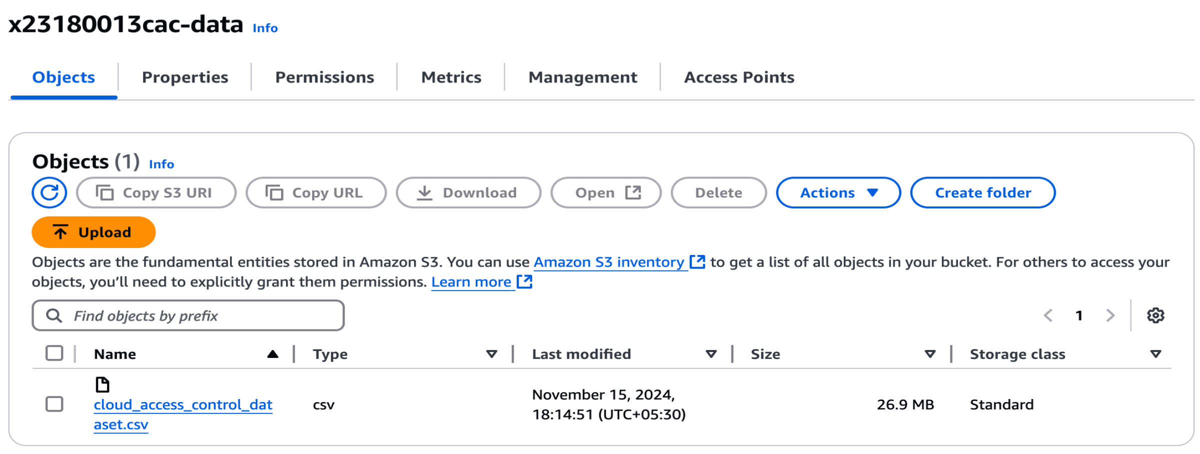
Click on the code dropdown button and copy the SSH URL, then go to the terminal and paste these commands to clone the repository.

```
git clone git@github.com:Annurag99/CloudAccessML.git
cd CloudAccessML/
cd cloudaccessui/
```

For machine learning go to its folder containing py and requirements files.

```
cd modelml/
```

3.2 Database Configuration


x23180013cac-data Info

Objects (1) Info

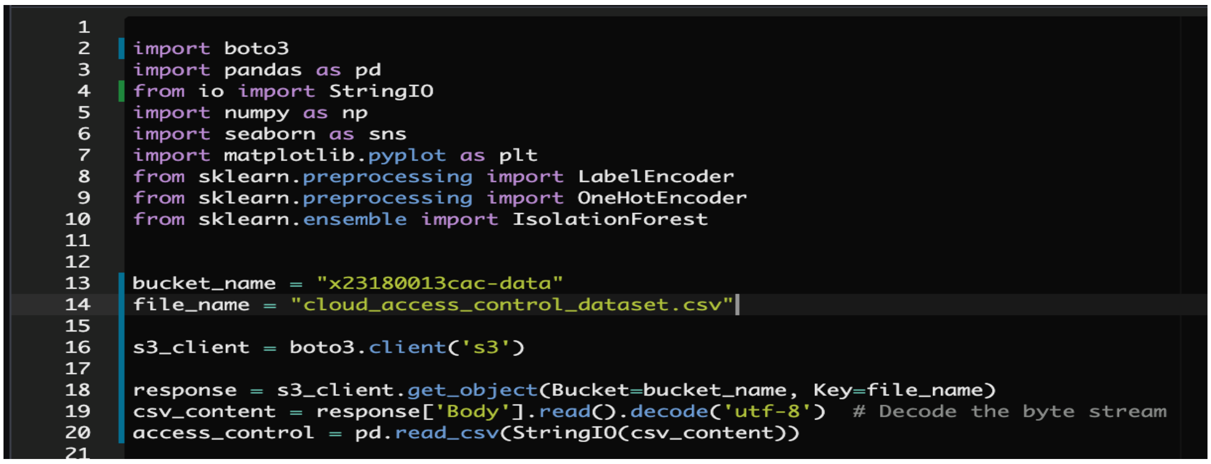
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
cloud_access_control_dataset.csv	csv	November 15, 2024, 18:14:51 (UTC+05:30)	26.9 MB	Standard

Figure 1: S3 Bucket Storing Dataset

To store and use the cloud access control dataset, we use AWS S3 (Simple Storage Service) and Jupyter Notebook to manage the data securely and prepare for model training efficiently as described in Figure 1. AWS S3 is being used to safely upload and organize the dataset into a specific S3 bucket as a reliable, highly scalable dataset storage solution. This offers the data accessible right away using AWS strong security, including encryption, and access control policy.

It is loaded into a Jupyter Notebook environment so that we can then use the dataset for model training and evaluation. For this, you would generally do this programmatically by accessing the S3 bucket through AWS's boto3 library and it gives you the ability to talk to the S3 service easily. This dataset is downloaded directly from the bucket into the notebook and preprocessed and analyzed so that it's ready to be fed into machine learning tasks as shown in Figure 2. Not only does this approach make it easy to handle the data, but it's also a way to prove reproducibility and allow for collaboration as the dataset never actually leaves S3 but can be accessed by those who are authorized to do so in disparate environments.



```python

import boto3
import pandas as pd
from io import StringIO
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import IsolationForest


bucket_name = "x23180013cac-data"
file_name = "cloud_access_control_dataset.csv"

s3_client = boto3.client('s3')

response = s3_client.get_object(Bucket=bucket_name, Key=file_name)
csv_content = response['Body'].read().decode('utf-8')  # Decode the byte stream
access_control = pd.read_csv(StringIO(csv_content))
```

Figure 2: Code snippet to load a file from S3 bucket

3.3 Installing Dependencies

Create a virtual environment in Mac or Windows OS using the below commands.

```
virtualenv env
"OR"
python3 -m venv env
source env/bin/activate
pip install -r requirements.txt
python3 app.py makemigrations
```

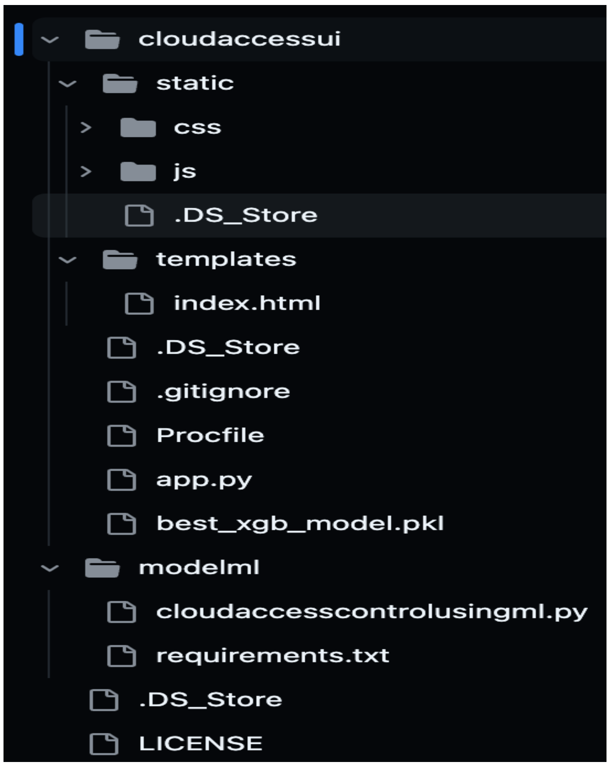
If the above command throws an error update Pip and Setuptools

```
pip install --upgrade pip setuptools
python3 app.py migrate
python3 app.py runserver 8080
```

4 Project Configuration

In the figure, it is the project configuration and directory structure that are separated into concerns to help keep things clear, scalable, and easy to manage. It is easy to maintain with fewer complexities if the project structure and configuration follow best practices. (Ray; 2022). I am elaborating below on the main components as shown in the Figure 3

- **Root Directory:** This contains essential files so that the overall functioning and deployment of the project.



```
cloudaccessui
  static
    css
    js
    .DS_Store
  templates
    index.html
  .DS_Store
  .gitignore
  Procfile
  app.py
  best_xgb_model.pkl
modelml
  cloudaccesscontrolusingml.py
  requirements.txt
.DS_Store
LICENSE
```

Figure 3: Directory Structure of the Thesis

  - `Procfile`: Used in deployment (example Elastic Beanstalk), specifies the command to start the application.
  - `app.py`: A script with main Flask application code that interacts behind the scenes.
  - `best_xgb_model.pkl`: The trained XGBoost model with the options used for predictions as a serialized file.
  - `LICENSE`: Terms of use for this repository are described here.
  - `.gitignore`: A file containing the commands for Git to ignore certain files or directories during version control.

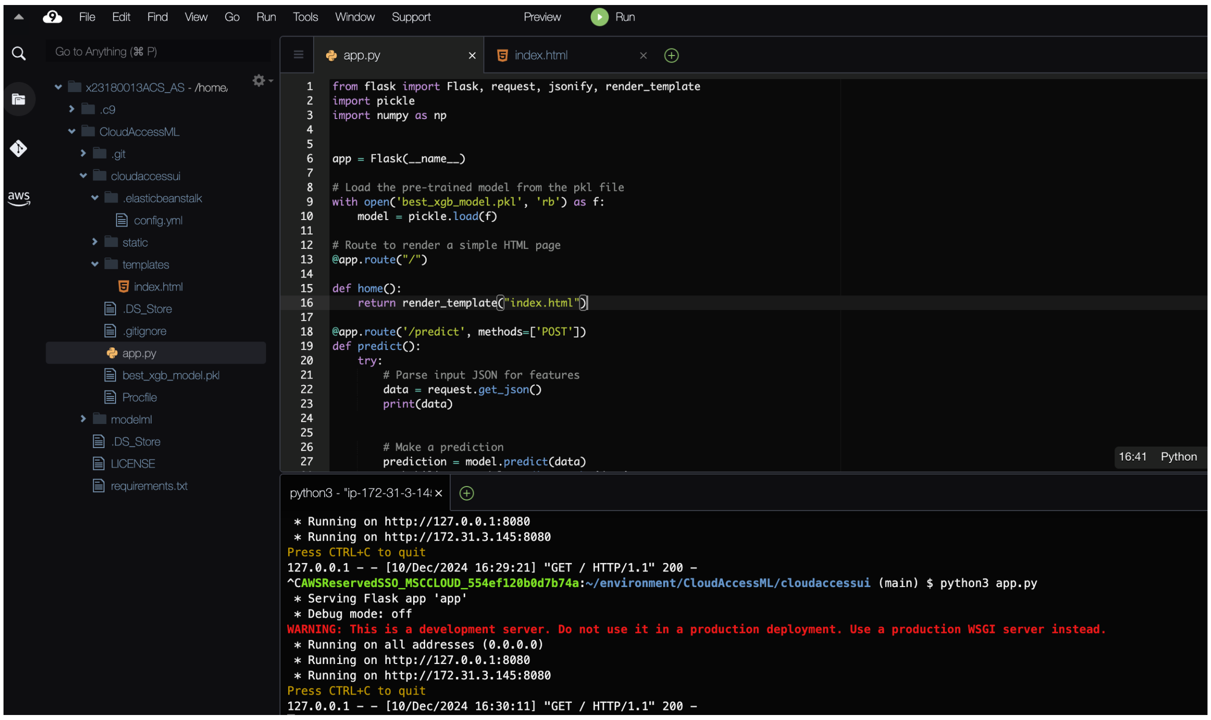
- **Subdirectories:**
  - `cloudaccessui/`: It has all Flask application components along with deployment, pickle, and proc files.
    - `static/`: It contains things like CSS and JavaScript files for styling and front-end functionality, but the useful part here is that these assets are held statically.
    - `templates/`: Contains all the views as HTML templates (e.g., `index.html`) rendered by Flask's Jinja2 engine.
  - `modelml/`: It has machine learning related components.
    - `cloudaccesscontrolusingml.py`: Machine learning implementation using python and its libraries.
    - `requirements.txt`: It lists Python dependencies required for this thesis.

- `.DS_Store`: Preferences displays a macOS system file storing folder. This file is not necessary for the project so it can be ignored using `.gitignore`.

5 Running the Application

5.1 Start the Application

To start the application as well as the machine learning notebook. Follow the below command to run the Flask application and preview the running application in Cloud 9 Figure 4.



```python
from flask import Flask, request, jsonify, render_template
import pickle
import numpy as np


app = Flask(__name__)

# Load the pre-trained model from the pkl file
with open('best_xgb_model.pkl', 'rb') as f:
    model = pickle.load(f)

# Route to render a simple HTML page
@app.route("/")

def home():
    return render_template("index.html")

@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Parse input JSON for features
        data = request.get_json()
        print(data)


        # Make a prediction
        prediction = model.predict(data)
```

```
 * Running on http://127.0.0.1:8080
 * Running on http://172.31.3.145:8080
Press CTRL+C to quit
127.0.0.1 - - [10/Dec/2024 16:29:21] "GET / HTTP/1.1" 200 -
^CAWSReservedSSO_MSCCLOUD_554ef120b0d7b74a:~/environment/CloudAccessML/cloudaccessui (main) $ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://172.31.3.145:8080
Press CTRL+C to quit
127.0.0.1 - - [10/Dec/2024 16:30:11] "GET / HTTP/1.1" 200 -
```

Figure 4: Cloud9 terminal running the application

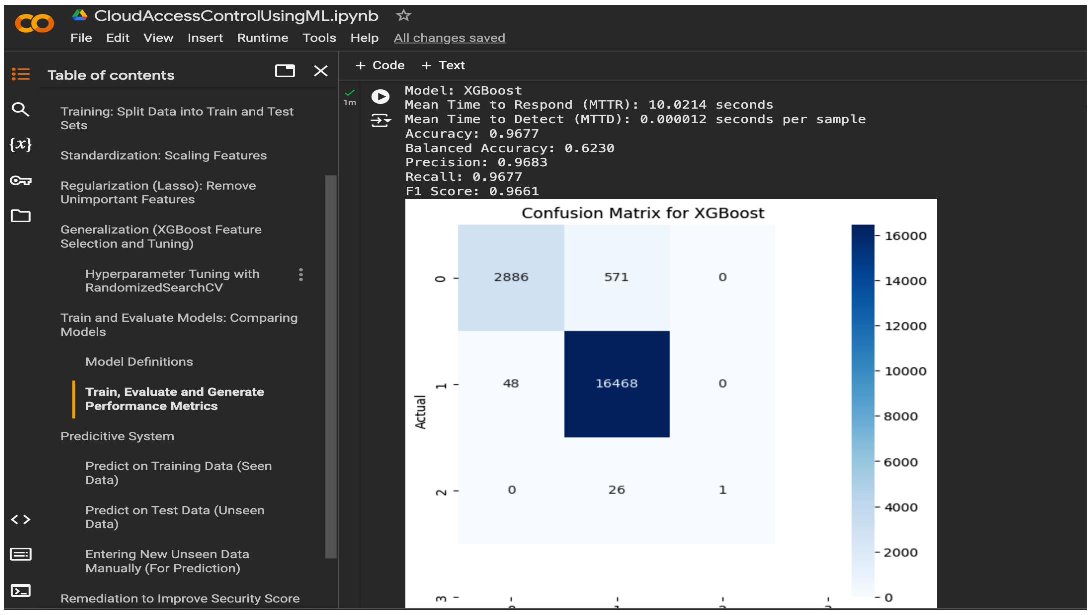
```
pip install -r requirements.txt
python3 app.py
```

For setting up dependencies in machine learning model training and evaluation follow the below commands.

```
pip install -r requirements.txt
python3 cloudaccesscontrolusingml.py
```

5.2 Executing Machine Learning Notebook on Google Collab

Google Colab provides a collaborative space to train and evaluate machine learning models so it is apt for researcher(Google; 2024). To execute a machine learning notebook on Google Colab as shown in Figure 5, begin by accessing Google Colab through your web browser. Once you sign in with your Google account select 'New Notebook' to make a



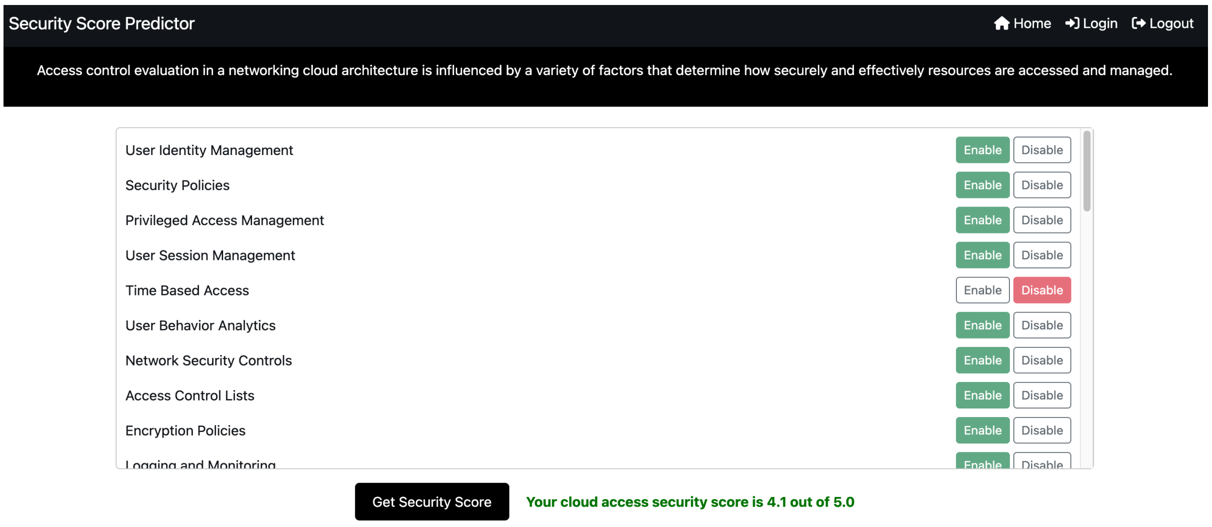
CloudAccessControlUsingML.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
Model: XGBoost
Mean Time to Respond (MTTR): 10.0214 seconds
Mean Time to Detect (MTTD): 0.000012 seconds per sample
Accuracy: 0.9677
Balanced Accuracy: 0.6230
Precision: 0.9683
Recall: 0.9677
F1 Score: 0.9661
```

Figure 5: Implementation in Google Colab

brand new Python notebook. The method that seems to work when you have datasets, navigate to the left sidebar, click on the folder icon, and then hit upload. Then, once uploaded with the necessary files, a code cell will have pip commands to install any needed libraries. This way, you can use the essential machine learning libraries such as Pandas, NumPy, and TensorFlow. After installing the libraries, we can import them into the notebook with Python's import statement, for example, pd for data manipulation from the libraries will be imported as import pandas as pd.

6 Usage Instructions



Access control evaluation in a networking cloud architecture is influenced by a variety of factors that determine how securely and effectively resources are accessed and managed.

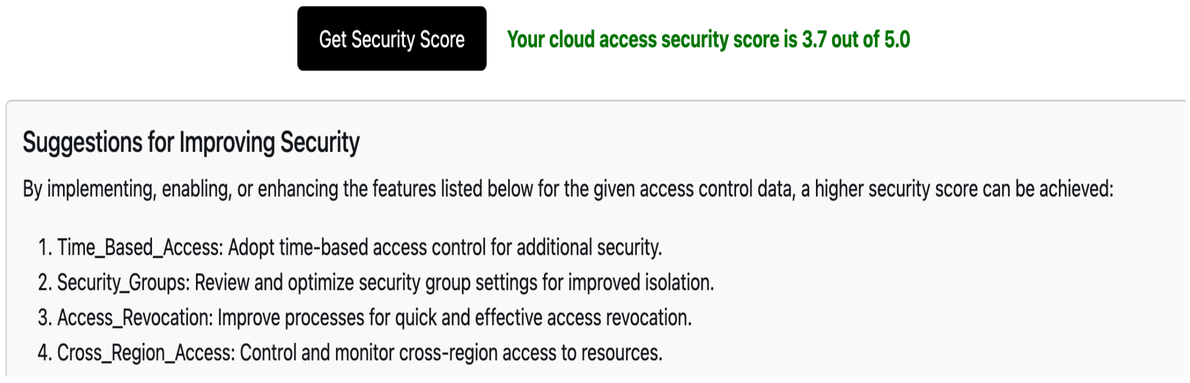
User Identity Management	[✓ Enable] Disable
Security Policies	[✓ Enable] Disable
Privileged Access Management	[✓ Enable] Disable
User Session Management	[✓ Enable] Disable
Time Based Access	Enable [✓ Disable]
User Behavior Analytics	[✓ Enable] Disable
Network Security Controls	[✓ Enable] Disable
Access Control Lists	[✓ Enable] Disable
Encryption Policies	[✓ Enable] Disable
Logging and Monitoring	[✓ Enable] Disable

Your cloud access security score is 4.1 out of 5.0

Figure 6: UI for security score prediction.

Using the web interface or via API requests, users can input data into the Cloud Security Scoring System as shown in Figure 6. It provides the web interface, which has a list of cloud security policies displayed in toggle buttons that enable or disable given policies. Users have to toggle the policies they want to apply, click the ”Get Security Score” button, and the data as its input. The system takes input, sends it to the backend API, and applies the calculation of the security score which is populated on the interface. The score is accompanied by a color-coded feedback system.In red (poor performance), scores less than or equal to 2 are displayed, yellow (average performance) with scores of 3, and green (good performance) with scores greater than 3.

It can be influenced by things like being able to enable or disable policies such as ”User Identity Management' or 'Time-Based Access'. If users click on 'Get Security Score,' they may see a result in the form of something like 'Your cloud access security score is 4.2/5.0,' on the interface. Additionally, users can view model-specific confusion matrices, F1, Recall, and accuracy to analyze the performance and error analysis of the trained models to a deeper level. This high-level system ensures that you have an easy way to use, distill down, analyze, and give feedback on your cloud security configurations that you can use to improve as given in Figure 7.



Your cloud access security score is 3.7 out of 5.0

Suggestions for Improving Security

By implementing, enabling, or enhancing the features listed below for the given access control data, a higher security score can be achieved:

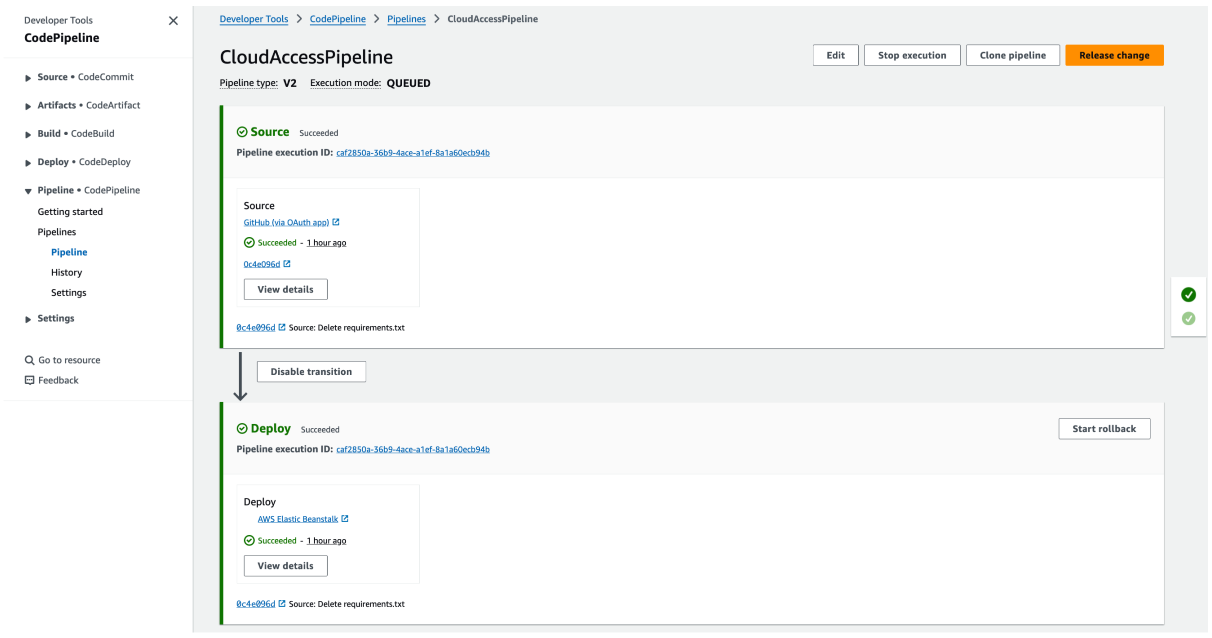
1. Time_Based_Access: Adopt time-based access control for additional security.
2. Security_Groups: Review and optimize security group settings for improved isolation.
3. Access_Revocation: Improve processes for quick and effective access revocation.
4. Cross_Region_Access: Control and monitor cross-region access to resources.

Figure 7: UI for security score enhancement suggestions.

7 Deployment Instructions

7.1 Setup AWS CodePipeline and Deploy

To set up AWS CodePipeline and deploy an application to Elastic Beanstalk as shown in Figure 8, begin by ensuring that your application is ready for deployment. This involves having a properly structured application repository (e.g., Flask app) hosted on GitHub, CodeCommit, or another supported source. Your repository should include essential files like requirements.txt, a Procfile for specifying the application's startup command, and any environment-specific configurations under the .ebextensions directory if required. Navigate to the AWS Management Console, open CodePipeline, and create a new pipeline. Select a source provider such as GitHub, authenticate it, and choose the repository and branch containing your application.



CloudAccessPipeline

Pipeline type: V2 Execution mode: QUEUED

Source Succeeded
Pipeline execution ID: caf2850a-36b9-4ace-a1ef-8a1a60ecb94b

Source
GitHub (via OAuth app)
Succeeded - 1 hour ago
0c4e096d

0c4e096d Source: Delete requirements.txt

Deploy Succeeded
Pipeline execution ID: caf2850a-36b9-4ace-a1ef-8a1a60ecb94b

Deploy
AWS Elastic Beanstalk
Succeeded - 1 hour ago

0c4e096d Source: Delete requirements.txt

Figure 8: Implementation of Code Pipeline

7.2 Link Elastic Beanstalk with Your GitHub Repository

First, initialize Elastic Beanstalk in the project directory terminal.

```
eb init
```

Select any AWS region available and choose an application name then select Python platform. Remember, do not enable CodeCommit if using GitHub. Then configure a default environment using the below command.

```
eb create <environment-name>
```

Set up GitHub in your machine using SSH token and paste the token in the setting window under SSH and GPG keys.

```
eb init -i
```

Then select the GitHub source control and give the repository/branch details. Finally, deploy the application using the below command.

```
eb deploy
```

In the deployment step, select "AWS Elastic Beanstalk" as the deployment provider. Elastic Beanstalk makes deploying and managing applications simple without fighting interface complexities (Buyya; 2024). Ensure that the Elastic Beanstalk environment for your application is already created and running. The pipeline will automatically detect the existing environment. Once configured, CodePipeline will trigger automatically every time you push changes to the specified repository branch, fetching the updates and deploying them to your Elastic Beanstalk environment. After deployment, you can verify your application by accessing the Elastic Beanstalk environment URL provided in the Elastic Beanstalk console. Monitor the pipeline's stages in the CodePipeline dashboard to ensure the deployment process is successful. This setup streamlines continuous deployment for maintaining and updating your application efficiently.

References

Buyya, R. (2024). Elastic beanstalk: A simplified deployment approach, *Cloud Computing Journal* **15**: 45–50.

Google (2024). Google colab: A collaborative environment for machine learning.
URL: *https://colab.research.google.com/*

Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*, O'Reilly Media, Inc.

Ray, A. (2022). Best practices for project directory organization in python.
URL: *https://realpython.com/python-application-layouts/*

Services, A. W. (2024). Aws documentation: Elastic beanstalk, s3, and cloud9.
URL: *https://aws.amazon.com/documentation/*