S3

IAM

Identify Access Management

* With IAM, you can centrally manage permissions that control which AWS resources users can access.
* You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

IAM Features

**Shared access to your AWS account**

**Secure access to AWS resources for applications that run on Amazon EC2**

**Multi-factor authentication (MFA)**

* **Managed policies** – Standalone identity-based policies that you can attach to multiple users, groups, and roles in your AWS account. You can use two types of managed policies:
  + **AWS managed policies** – Managed policies that are created and managed by AWS. If you are new to using policies, we recommend that you start by using AWS managed policies.
  + **Customer managed policies** – Managed policies that you create and manage in your AWS account. Customer managed policies provide more precise control over your policies than AWS managed policies. You can create, edit, and validate an IAM policy in the visual editor or by creating the JSON policy document directly. For more information, see [Creating IAM policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create.html) and [Editing IAM policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_manage-edit.html).

Task -1 file to file

destination : /home/ec2-user/d1

Source:/home/ec2-user/s1

A computer screen shot of a program code

Description automatically generated

Output:

A screen shot of a computer code

Description automatically generated

Task-2 file to file using input

Source: /home/ec2-user/d1

Destination:/home/ec2-user/d3

Code:

A screenshot of a computer program

Description automatically generated

Before running :

Output:

A screen shot of a computer code

Description automatically generated

After running :

A computer screen with white text

Description automatically generated

[ec2-user@ip-172-31-11-173 ~]$ aws configure

AWS Access Key ID [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PK7B]: AKIAQ3EGTEJKKWULPK7B

AWS Secret Access Key [\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Idzc]: 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

Default region name [Mumbai]: ap-south-1

Default output format [.txt]: json

[ec2-user@ip-172-31-11-173 ~]$ aws s3 cp s3://bkttt1/s1/ /home/ec2-user/s1

download failed: s3://bkttt1/s1/ to s1/ [Errno 20] Not a directory: '/home/ec2-user/s1/.F35d6bd4' -> '/home/ec2-user/s1/'

[ec2-user@ip-172-31-11-173 ~]$ ls

s1

[ec2-user@ip-172-31-11-173 ~]$ aws s3 cp s3://bkttt1/s1/ /home/ec2-user/s1 --recursive

A screenshot of a computer

Description automatically generated

import os

import logging

# Configure logging

log\_file = '/home/ec2-user/c1/log\_file.log'

logging.basicConfig(filename=log\_file, level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# Directory containing files

directory = '/home/ec2-user/b1'

# Function to get file information and write to log

def get\_file\_info(file\_path):

try:

# Get file name

file\_name = os.path.basename(file\_path)

# Get file type

file\_type = os.path.splitext(file\_path)[1]

# Get file location

file\_location = os.path.dirname(file\_path)

# Write information to log

logging.info(f"File Name: {file\_name}, File Type: {file\_type}, File Location: {file\_location}")

except Exception as e:

logging.error(f"Error processing file: {file\_path}, {e}")

# Iterate through files in the directory

for file in os.listdir(directory):

file\_path = os.path.join(directory, file)

# Check if it's a file

if os.path.isfile(file\_path):

get\_file\_info(file\_path)

logging.info("File information collection completed.")

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

aws s3 cp /home/ec2-user/c1/log\_file.log s3://bkttt1/s1/