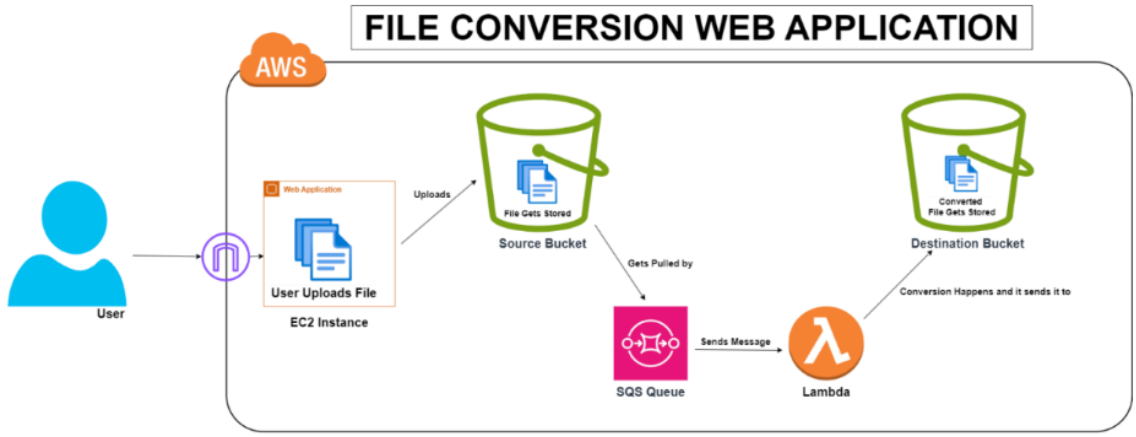
FILE CONVERSION WEB APPLICATION

Aim:

Create a web application leveraging AWS services like EC2, IAM, S3, SQS, and Lambda. The application allows users to upload files, which are then processed and converted, with the output displayed in a designated S3 bucket.

Architecture:

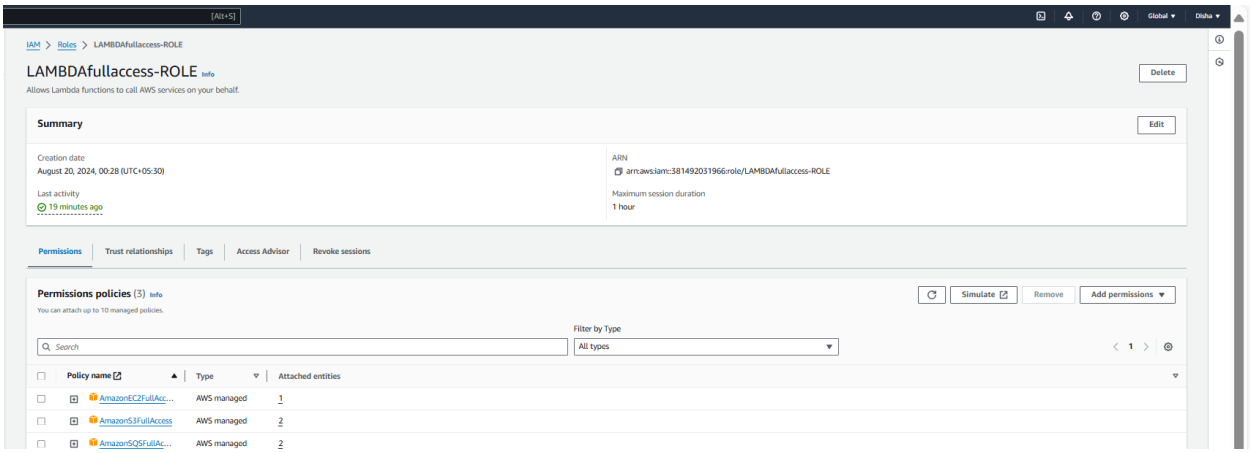


Prerequisites:

1. **EC2 Instance:** Host the web application and manage user interactions.
2. **Two S3 Buckets:** One for storing the original uploaded files and the other for storing the converted output.
3. **SQS Queue:** Manage and queue file conversion requests for processing.
4. **Lambda Function:** Automatically process and convert uploaded files from the source bucket to the destination bucket.

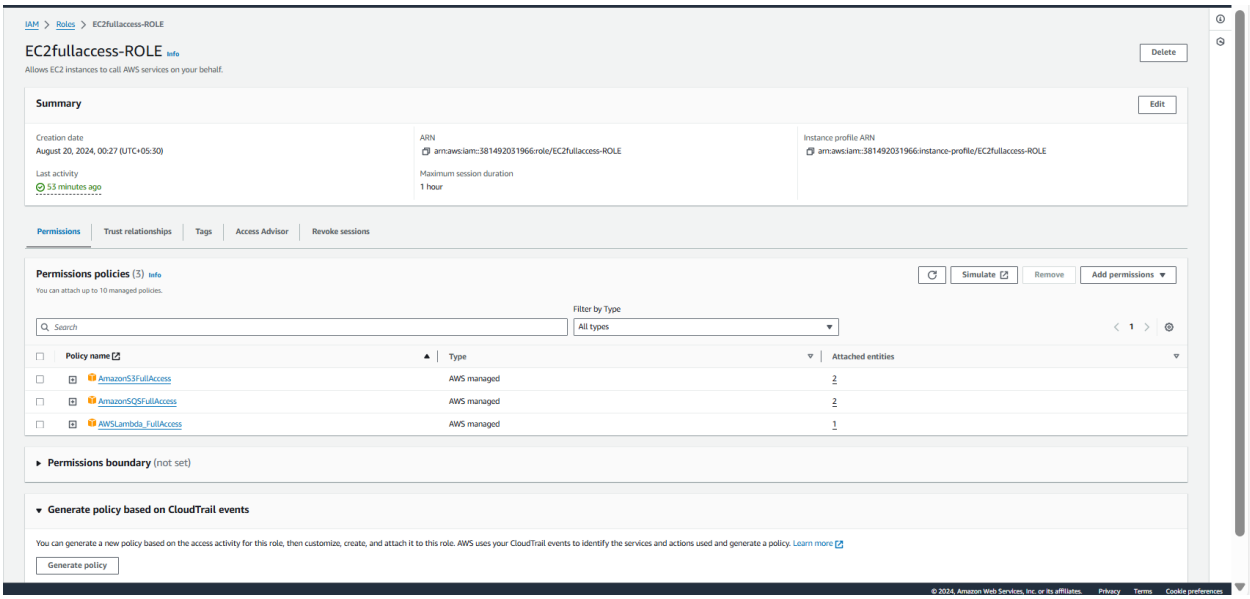
Steps:

1. Create 2 IAM Roles with Permissions
   - Lambda – AmazonEC2FullAccess, AmazonS3FullAccess, AmazonSQSFullAccess

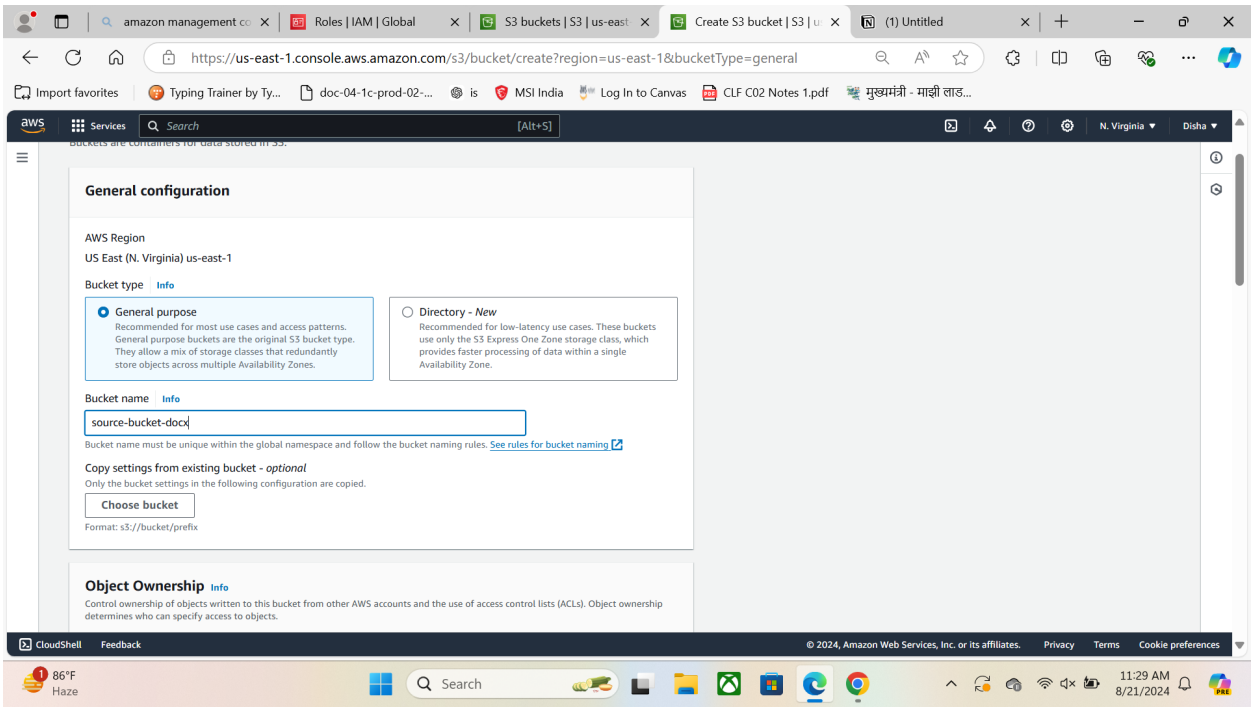


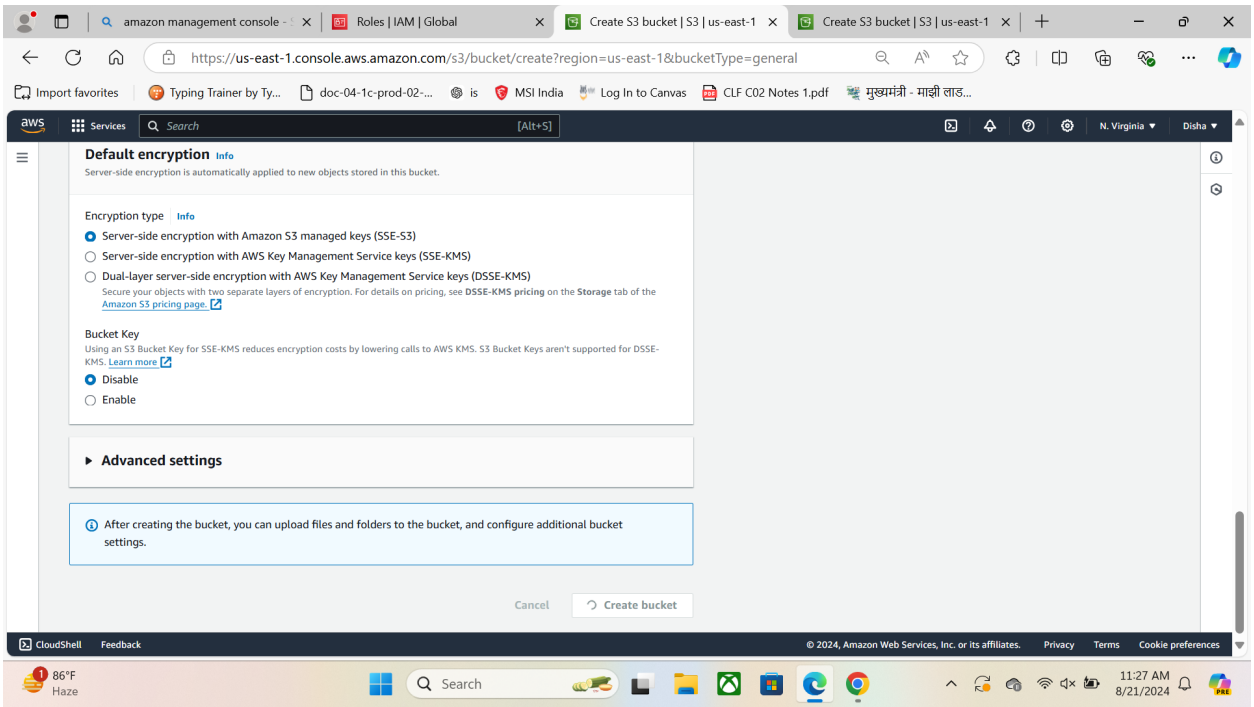
Steps:

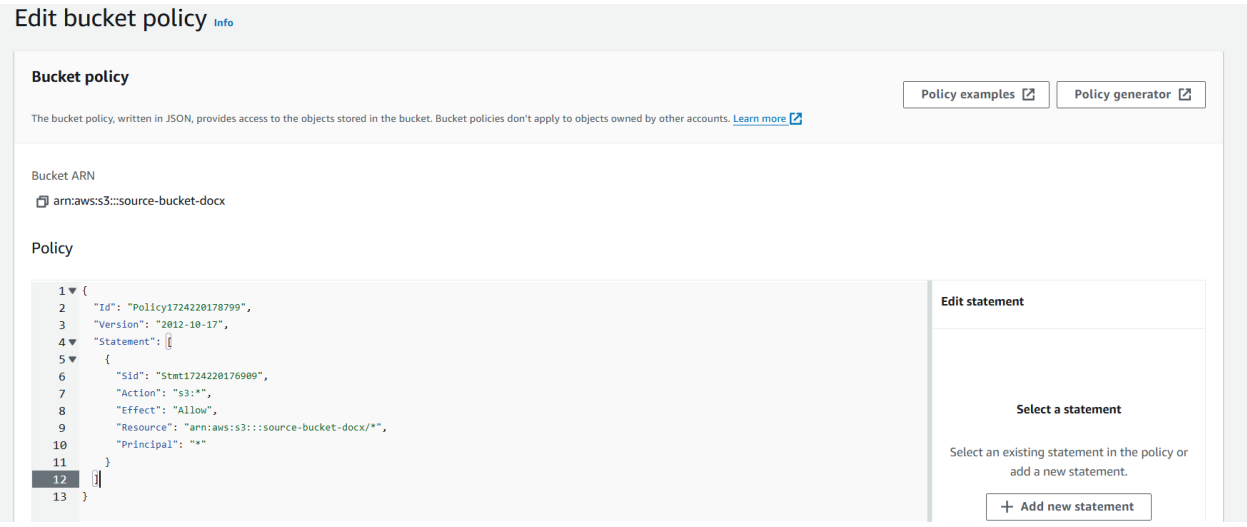
1. Create 2 IAM Roles with Permissions
   - EC2 – AmazonS3FullAccess, AmazonSQSFullAccess, AWSLambdaFullAccess



1. Create a S3 Bucket (Source Bucket)
   - Bucket name : source-bucket-docx
   - Object Ownership : Disabled
   - Block Public setting from bucket : Untick All and Acknowlege it
   - Bucket Versioning : Disable
   - Encryption Type: SSE-S3
   - Bucket Key: Disable
   - Create Bucket
   - Edit the policy of bucket as mentioned
   - Save Changes

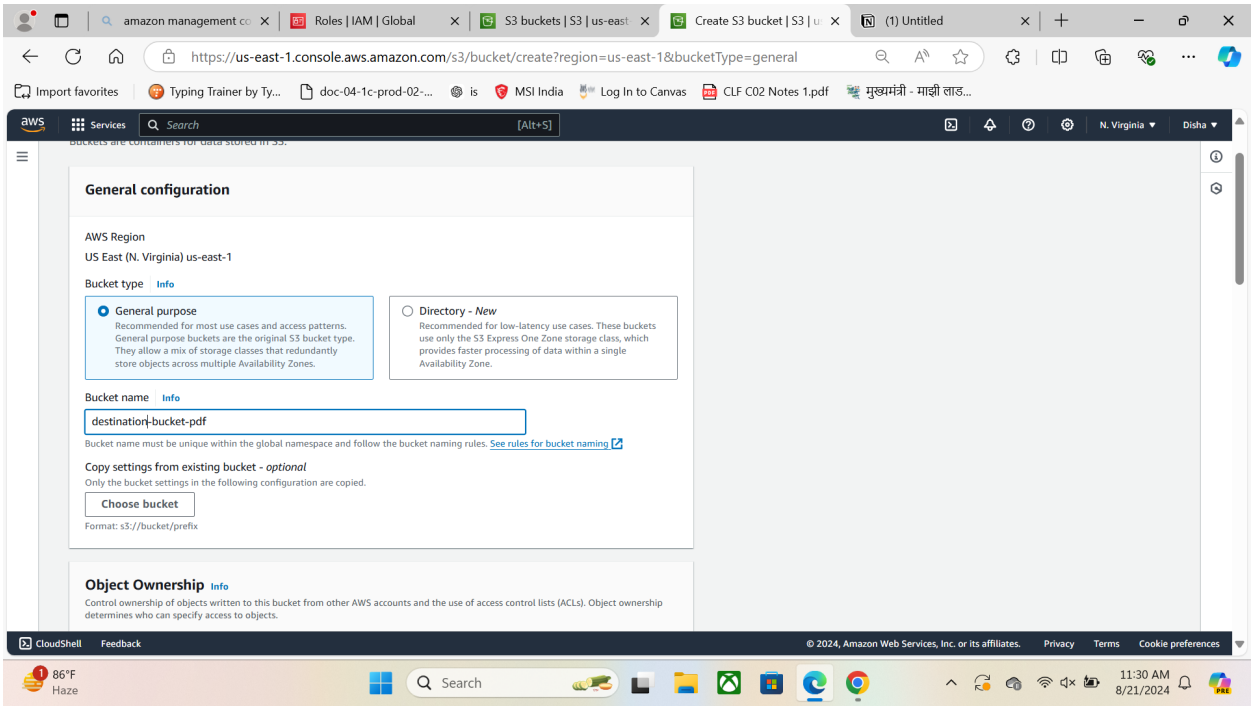


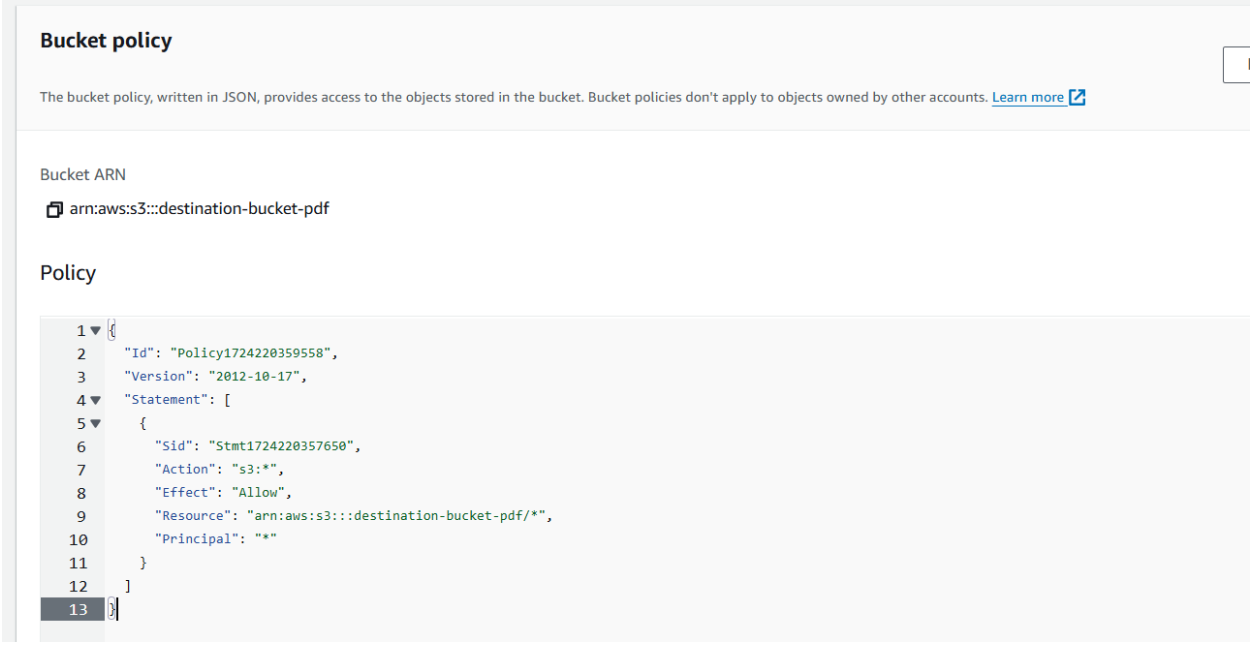




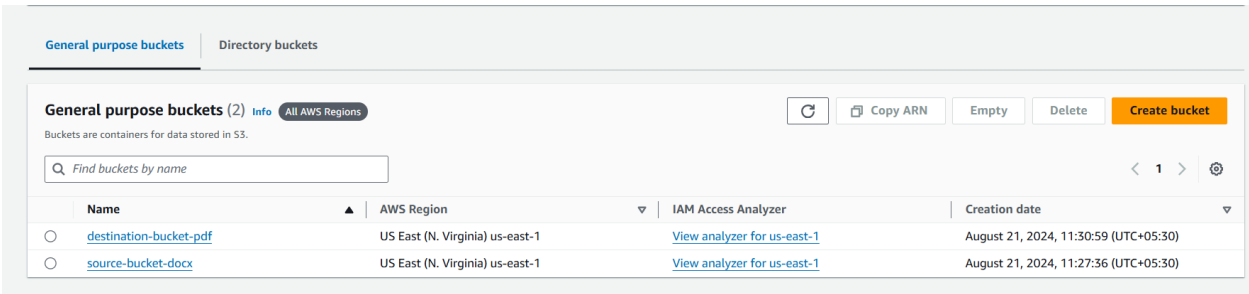
1. Create a S3 Bucket (Destination Bucket)
   - Bucket name : destinashun-bucket-pdf
   - Object Ownership : Disabled
   - Block Public setting from bucket : Untick All and Acknowlege it
   - Bucket Versioning : Disable
   - Encryption Type: SSE-S3

- Bucket Key: Disable
- Create Bucket
- Edit the policy of bucket as mentioned
- Save Changes

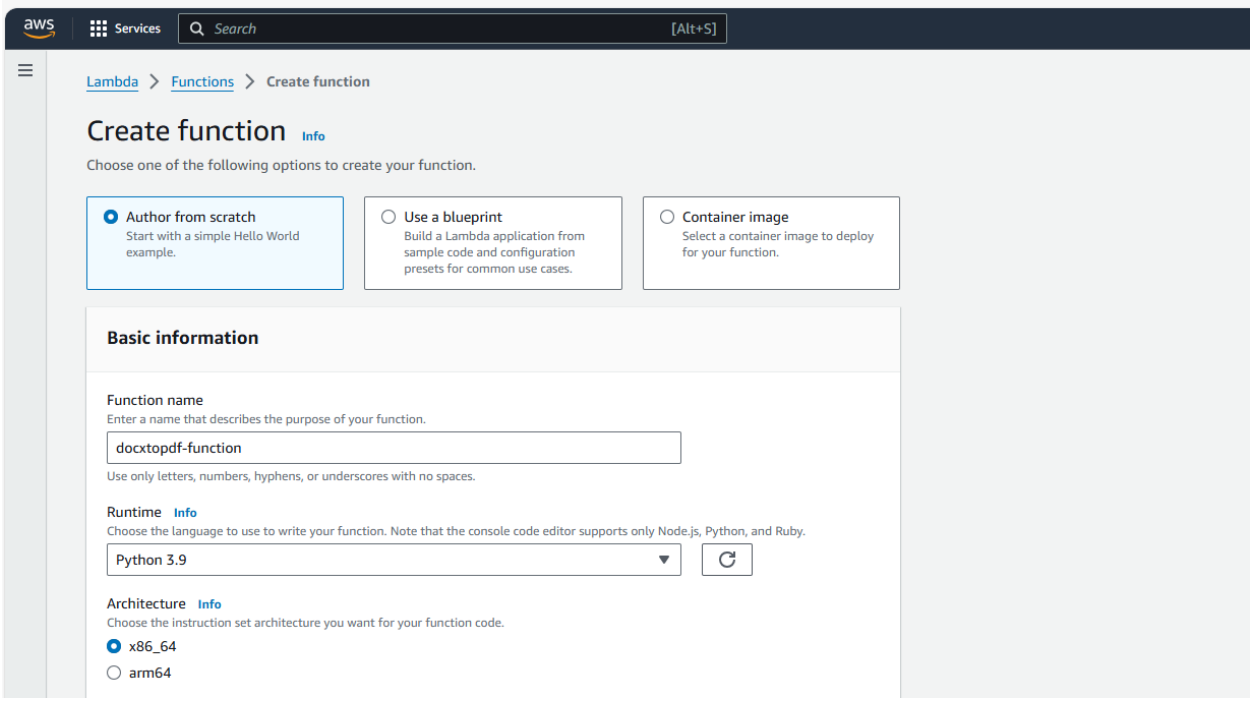


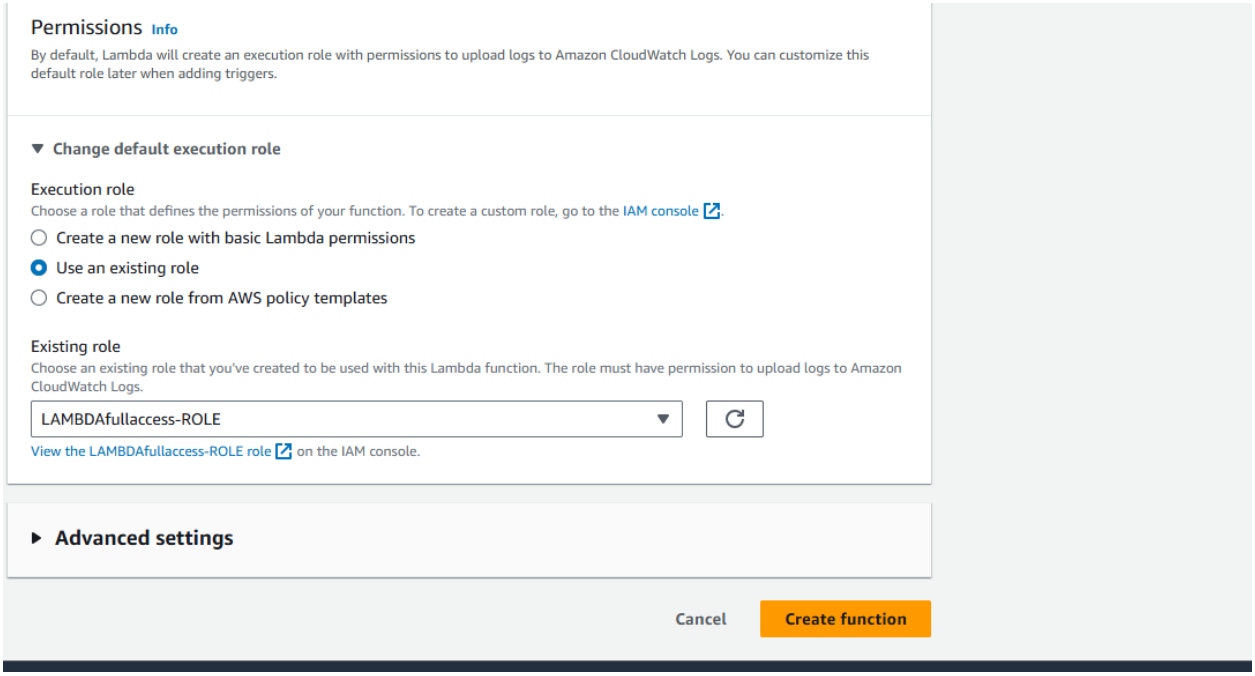


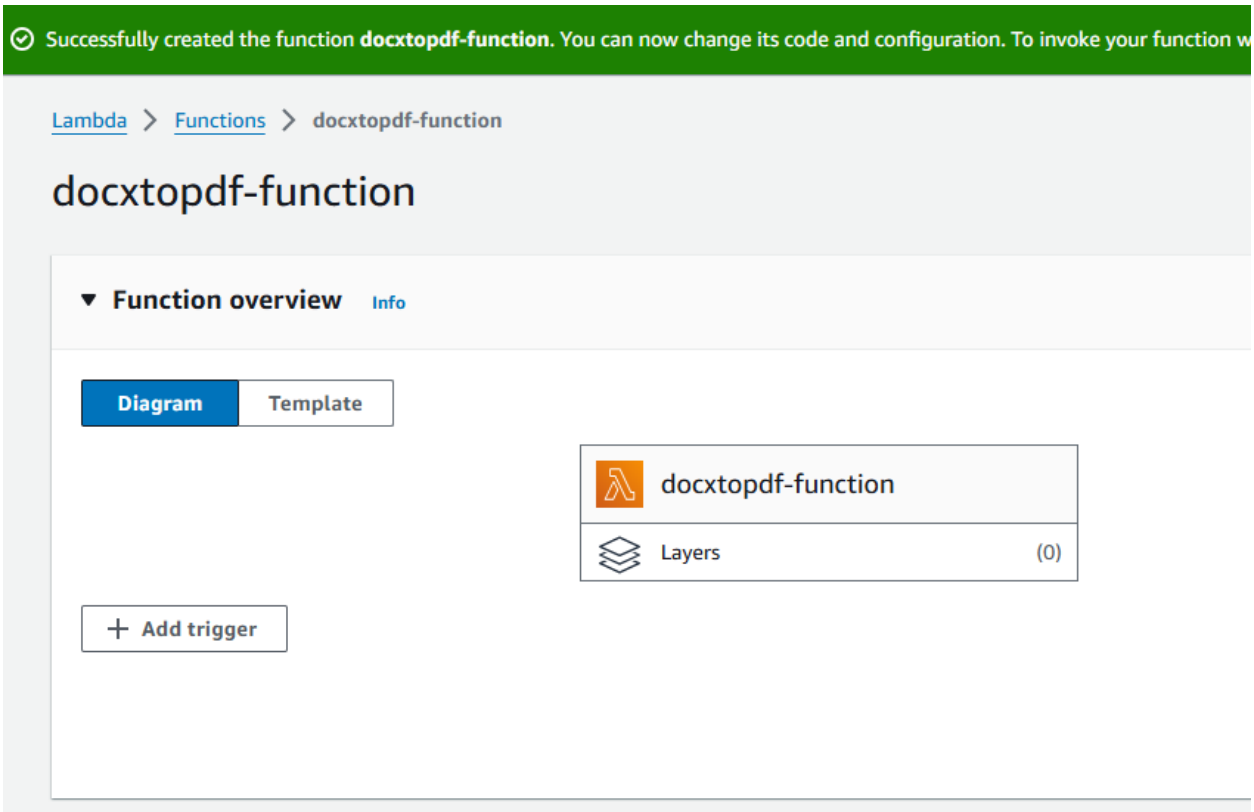
here is our both the buckets



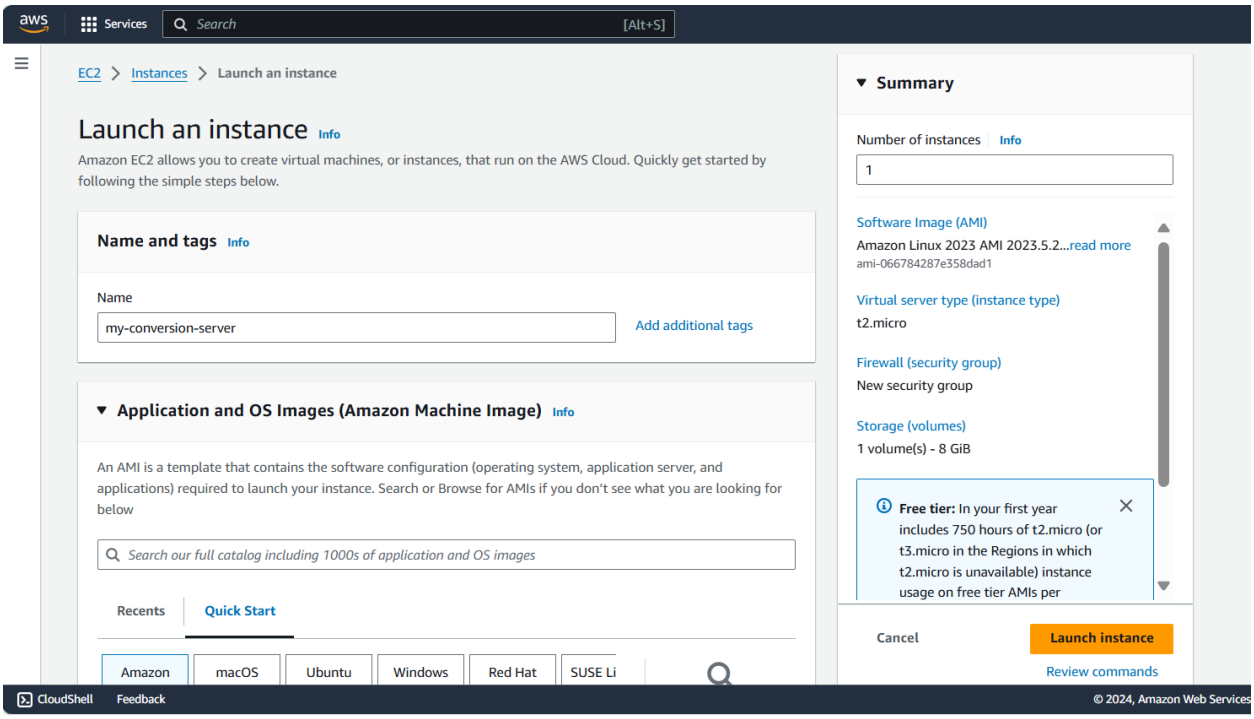
1. Create a Lambda Function (ConvertingFunction)
   - Author from scratch
   - Function name : ConvertingFunction
   - Runtime : Python 3.9
   - Architecture : x86_64
   - Permission : Give the role you made earlier for Lambda
   - Create Function

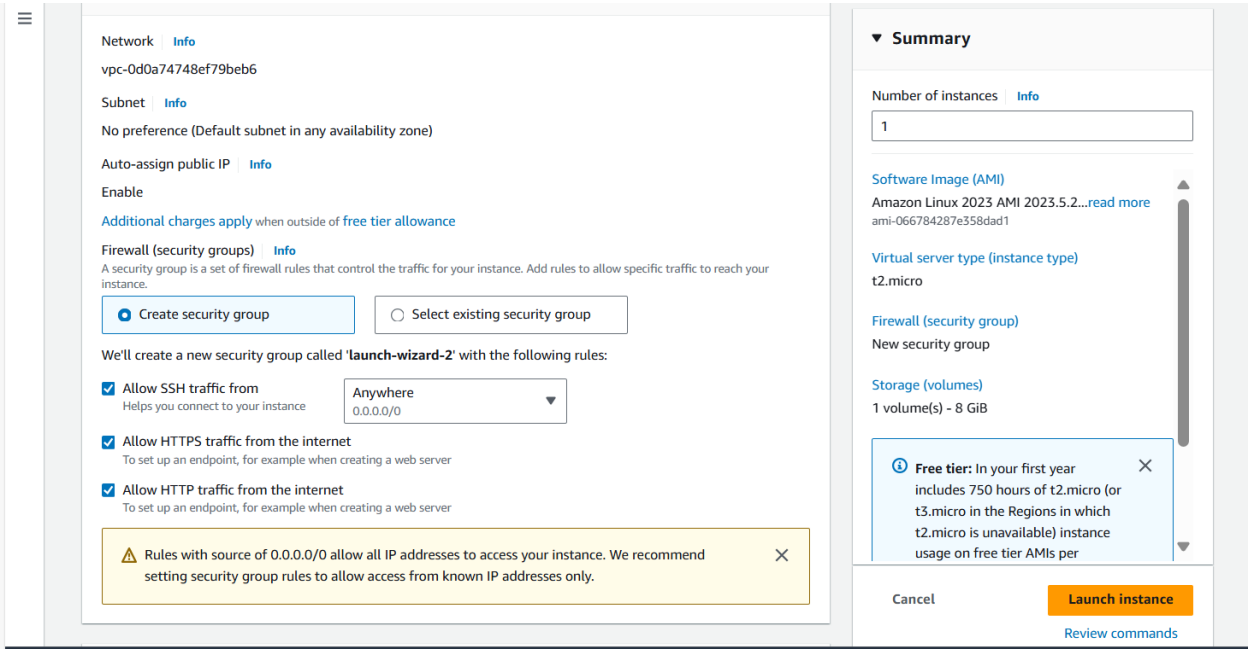

aws
Services
Search
[Alt+S]
Lambda > Functions > Create function
Create function Info
Choose one of the following options to create your function.
Author from scratch
Start with a simple Hello World example.
Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.
Container image
Select a container image to deploy for your function.
Basic information
Function name
Enter a name that describes the purpose of your function.
docxtopdf-function
Use only letters, numbers, hyphens, or underscores with no spaces.
Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9
Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64
arm64

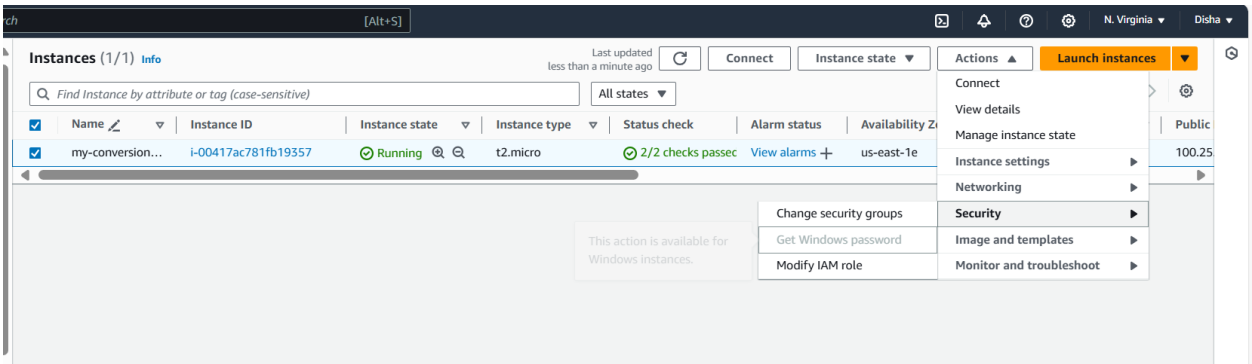

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.
Change default execution role
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
Create a new role with basic Lambda permissions
Use an existing role
Create a new role from AWS policy templates
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
LAMBDAfullaccess-ROLE
View the LAMBDAfullaccess-ROLE role on the IAM console.
Advanced settings
Cancel
Create function

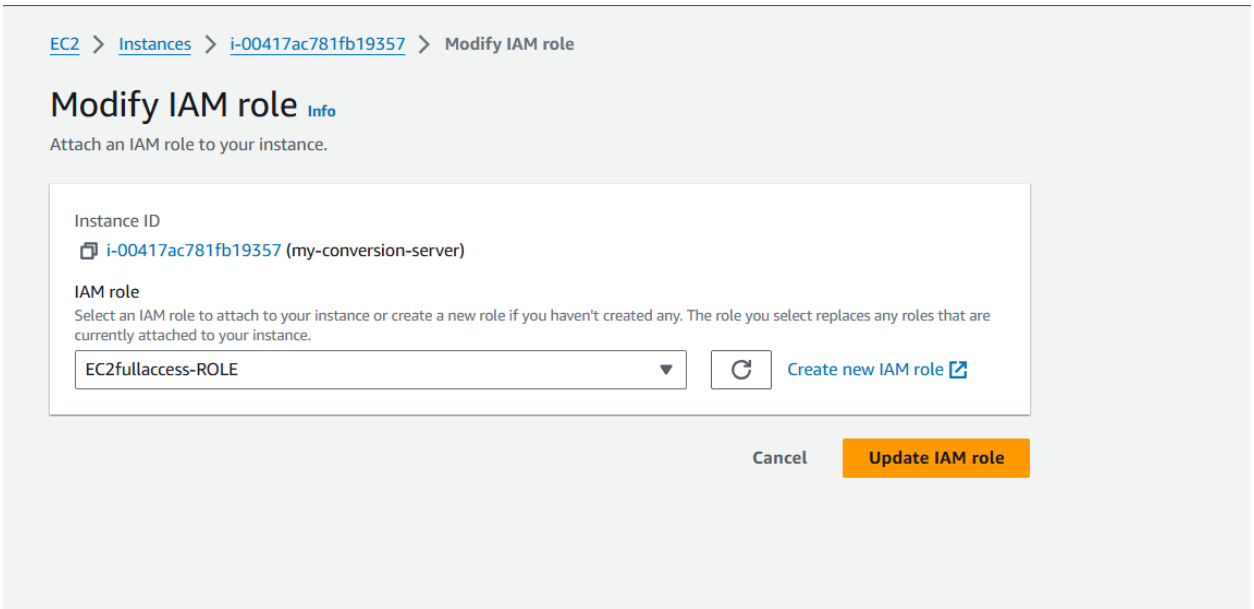


1. Create a EC2 Instance (File_Converter)
   - Name : my-conversion-server
   - AMI : Amazon Linux 2
   - Instance Type : t2.micro
   - Key pair (login) : newKey
   - Add SSH, HTTP & HTTPS
   - Launch Instance
   - In Actions, Go to Security and click on Modify IAM Role
   - Select IAM Role you created earlier for EC2
   - Save Changes

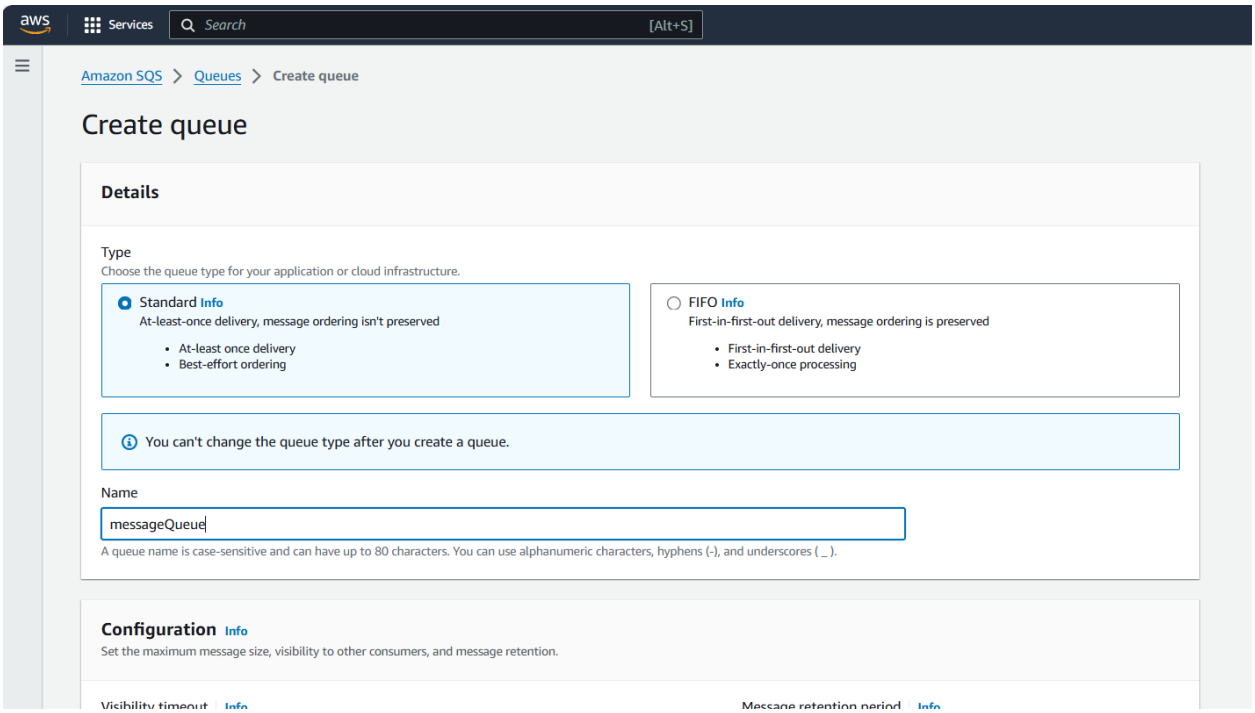

aws
Services
Search
[Alt+S]
EC2 > Instances > Launch an instance
Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
my-conversion-server
Add additional tags
Application and OS Images (Amazon Machine Image) Info
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below
Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Amazon
macOS
Ubuntu
Windows
Red Hat
SUSE Li
Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2023 AMI 2023.5.2...read more
ami-066784287e358dad1
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per
Cancel
Launch instance
Review commands
CloudShell
Feedback
© 2024, Amazon Web Services

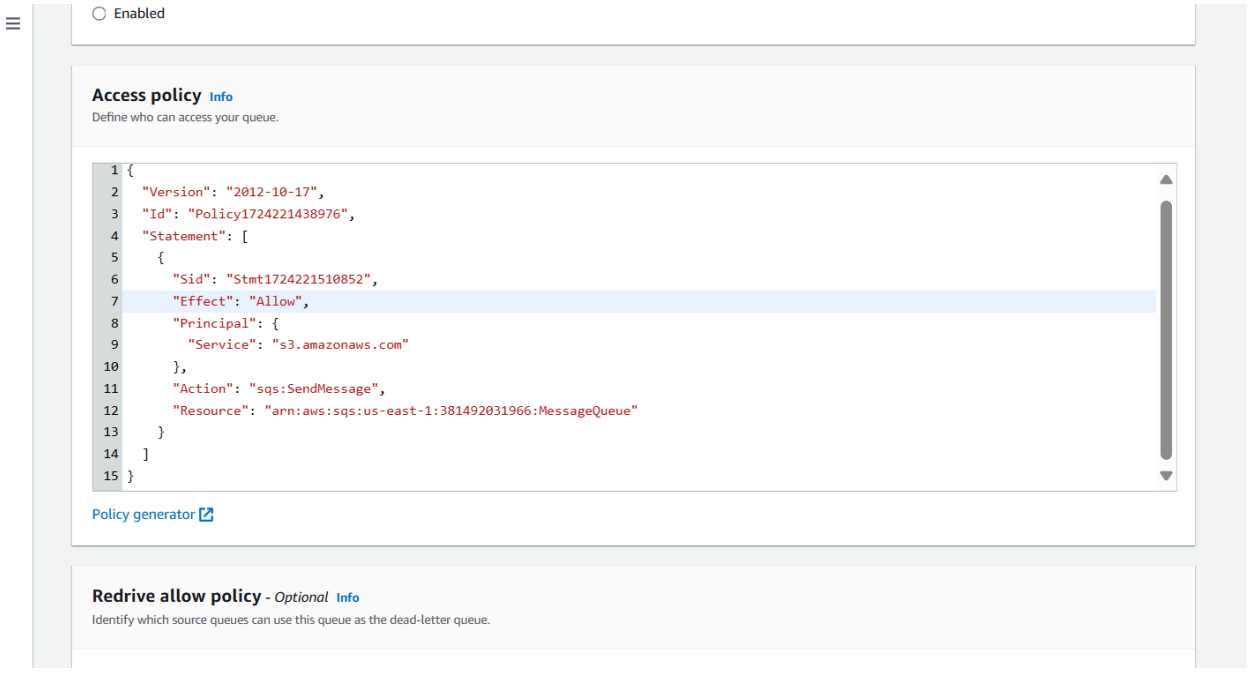

Network Info
vpc-0d0a74748ef79beb6
Subnet Info
No preference (Default subnet in any availability zone)
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
We'll create a new security group called 'launch-wizard-2' with the following rules:
Allow SSH traffic from
Helps you connect to your instance
Anywhere
0.0.0.0/0
Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server
Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.
Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2023 AMI 2023.5.2...read more
ami-066784287e358dad1
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per
Cancel
Launch instance
Review commands





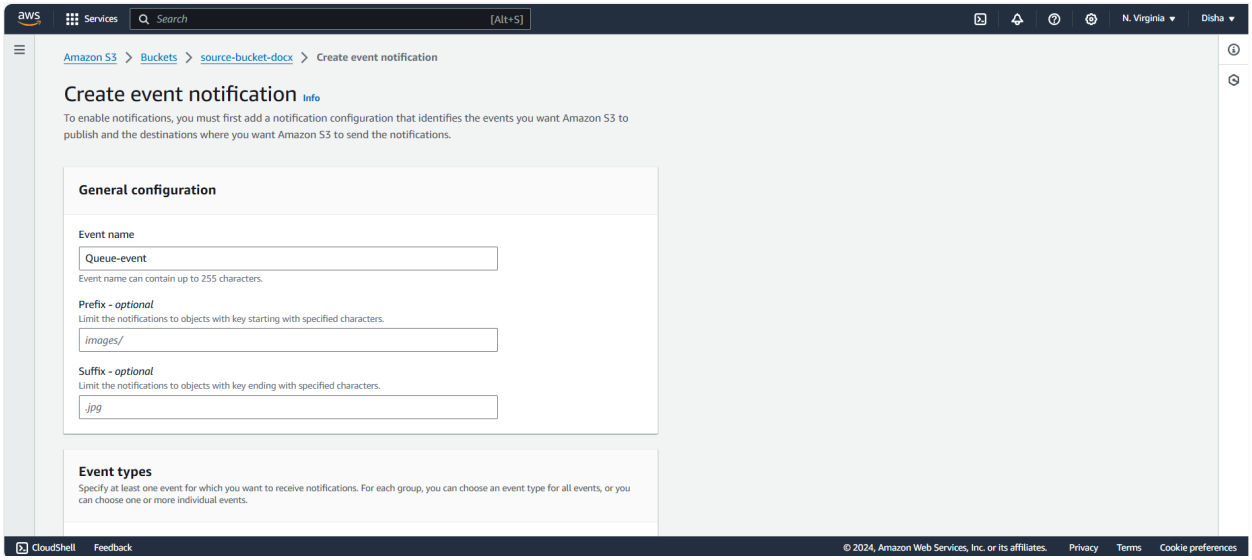
1. Create a SQS Queue (MessageQueue)
   - Type : Standard
   - Name : MessageQueue
   - Encryption : Disabled
   - Access Policy : Write it as mentioned
   - Everything else Disabled
   - Create Queue

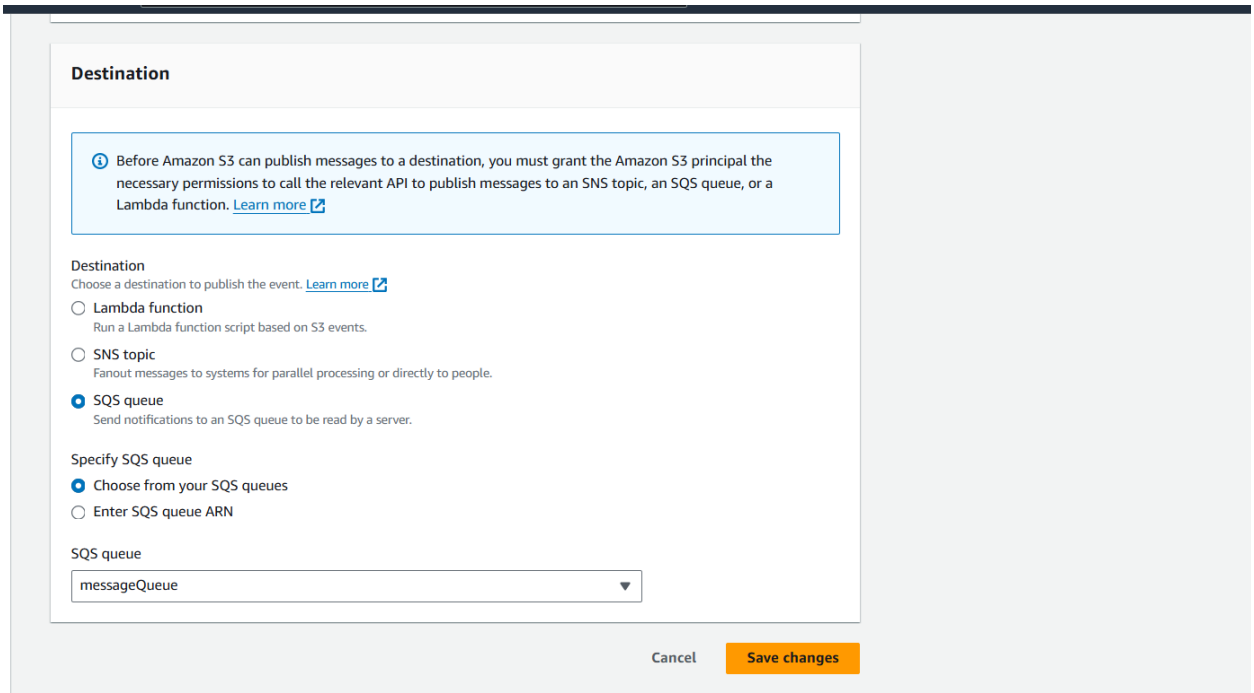


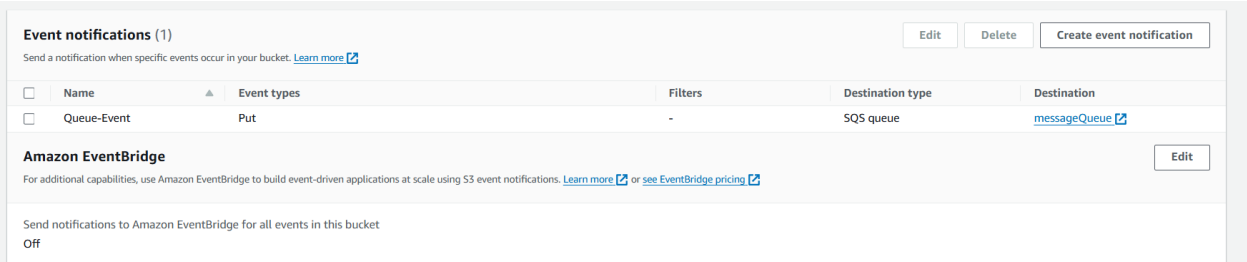


1. Go back to your source bucket and in properties create a event notification
   - Name : Queue_Event
   - Event Type : PUT

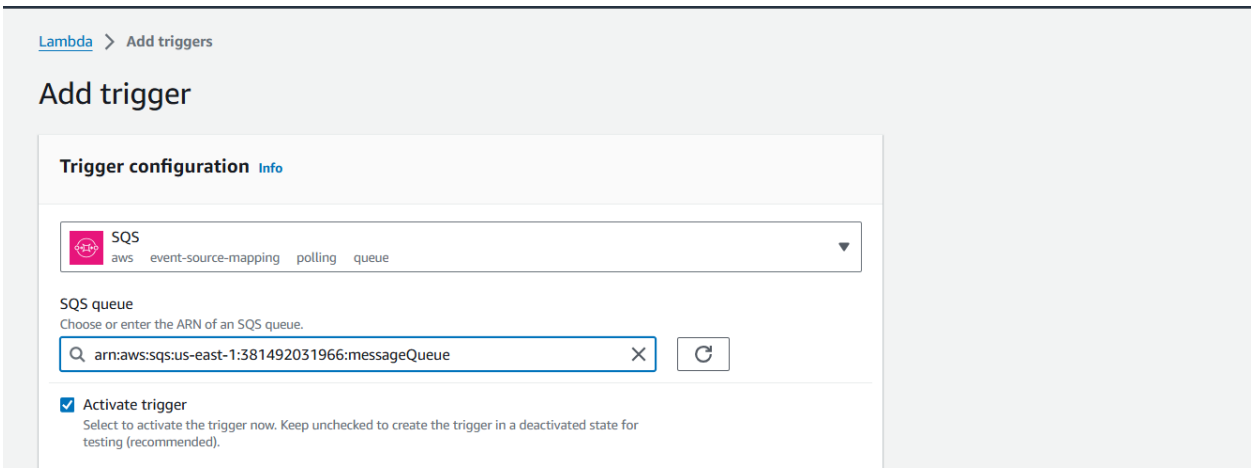
- Destination : SQS Queue
- Specify SQS Queue : MessageQueue
- Save Changes



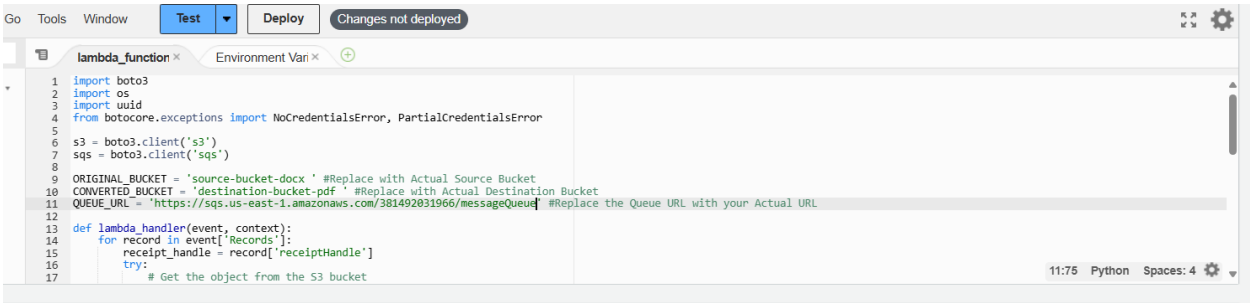


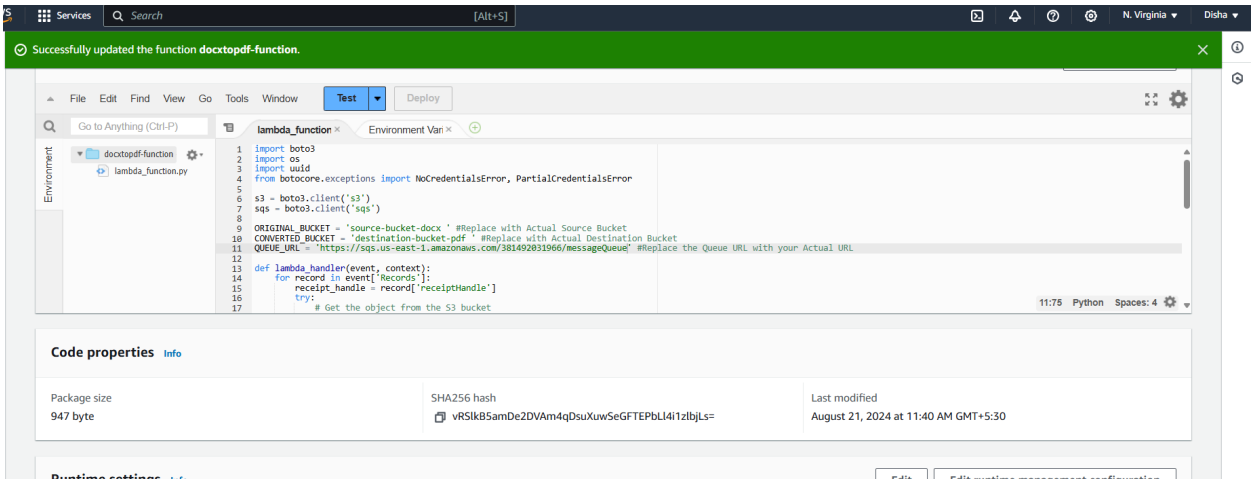


1. Go back to your Lambda Function and Add Trigger
   - Trigger Configuration : SQS
   - SQS queue : Select YOUR_SQS_QUEUE_ARN
   - Add



9. Also Write your Conversion Python Code in Lambda Function





1. Finally Connect your EC2 instance and write the commands and Execute the code to see your converted file in your destination bucket

sudo su #Super User Do Switch User
yum install python3 -y #Installing Python3 in Amazon Linux 2
yum install python3-pip -y #Installing Python3-pip in Amazon Linux 2
pip3 install Flask #Installing Flask in Amazon Linux 2
pip3 install boto3 #Installing boto3 in Amazon Linux 2
sudo nano
app.py #Creating a Python File to Write and Run the Code
#Write your Python Code [
app.py] here and press (Ctrl + X and Enter) to exit from nano
python3
app.py #To execute the script in your web server

```
Complete!
[root@ip-172-31-58-114 ec2-user]# yum install python3-pip -y
Last metadata expiration check: 0:01:06 ago on Wed Aug 21 06:50:30 2024.
Dependencies resolved.
================================================================================================================================================
 Package                                   Architecture                      Version                                          Repository                                  Size
================================================================================================================================================
Installing:
 python3-pip                               noarch                            21.3.1-2.amzn2023.0.7                            amazonlinux                                1.8 M
Installing weak dependencies:
 libxcrypt-compat                          x86_64                            4.4.33-7.amzn2023                                amazonlinux                                 92 k

Transaction Summary
================================================================================================================================================
Install  2 Packages

Total download size: 1.9 M
Installed size: 11 M
Downloading Packages:
(1/2): libxcrypt-compat-4.4.33-7.amzn2023.x86_64.rpm                                                                   1.2 MB/s |  92 kB     00:00
(2/2): python3-pip-21.3.1-2.amzn2023.0.7.noarch.rpm                                                                     18 MB/s | 1.8 MB     00:00
------------------------------------------------------------------------------------------------------------------------------------------------
Total                                                                                                                   12 MB/s | 1.9 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                                          1/1
```

```
aws  Services  Search [Alt+S]                                                                                  N. Virginia ▾  Disha ▾

Complete!
[root@ip-172-31-58-114 ec2-user]# pip3 install Flask
Collecting Flask
  Downloading flask-3.0.3-py3-none-any.whl (101 kB)
     |[redacted]| 101 kB 4.1 MB/s
Collecting Jinja2>=3.1.2
  Downloading jinja2-3.1.4-py3-none-any.whl (133 kB)
     |[redacted]| 133 kB 20.4 MB/s
Collecting importlib-metadata>=3.6.0
  Downloading importlib_metadata-8.4.0-py3-none-any.whl (26 kB)
Collecting Werkzeug>=3.0.0
  Downloading werkzeug-3.0.3-py3-none-any.whl (227 kB)
     |[redacted]| 227 kB 37.2 MB/s
Collecting click>=8.1.3
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
     |[redacted]| 97 kB 10.1 MB/s
Collecting blinker>=1.6.2
  Downloading blinker-1.8.2-py3-none-any.whl (9.5 kB)
Collecting itsdangerous>=2.1.2
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting zipp>=0.5
  Downloading zipp-3.20.0-py3-none-any.whl (9.4 kB)
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.5-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Installing collected packages: zipp, MarkupSafe, Werkzeug, Jinja2, itsdangerous, importlib-metadata, click, blinker, Flask
  Attempting uninstall: MarkupSafe
    Found existing installation: MarkupSafe 1.1.1
    Uninstalling MarkupSafe-1.1.1:
```

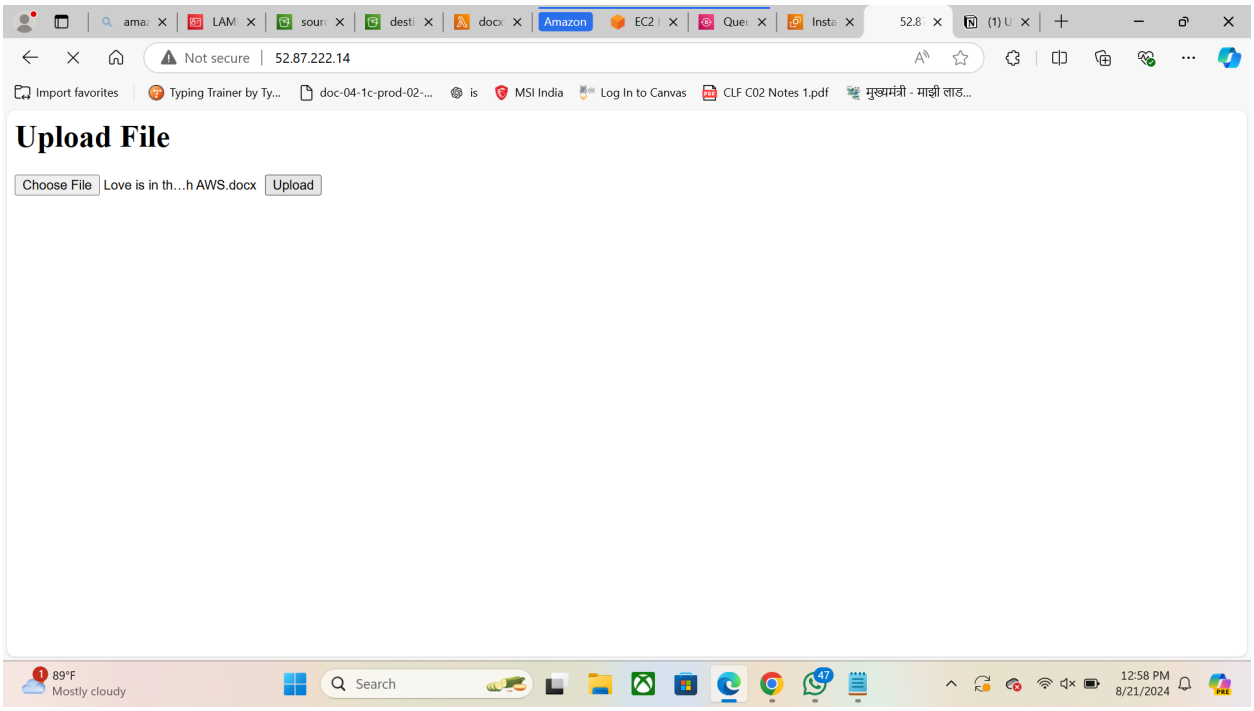
```
[root@ip-172-31-58-114 ec2-user]# pip3 install boto3
Collecting boto3
  Downloading boto3-1.35.2-py3-none-any.whl (139 kB)
     |[redacted]| 139 kB 4.8 MB/s
Collecting botocore<1.36.0,>=1.35.2
  Downloading botocore-1.35.2-py3-none-any.whl (12.5 MB)
     |[redacted]| 12.5 MB 40.4 MB/s
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3.9/site-packages (from boto3) (0.10.0)
Collecting s3transfer<0.11.0,>=0.10.0
  Downloading s3transfer-0.10.2-py3-none-any.whl (82 kB)
     |[redacted]| 82 kB 1.8 MB/s
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3.9/site-packages (from botocore<1.36.0,>=1.35.2->boto3) (2.8.1)
Requirement already satisfied: urllib3<1.27,>=1.25.4 in /usr/lib/python3.9/site-packages (from botocore<1.36.0,>=1.35.2->boto3) (1.25.10)
Requirement already satisfied: six>=1.5 in /usr/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.36.0,>=1.35.2->boto3) (1.15.0)
Installing collected packages: botocore, s3transfer, boto3
Successfully installed boto3-1.35.2 botocore-1.35.2 s3transfer-0.10.2
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment ins
tead: https://pip.pypa.io/warnings/venv
[root@ip-172-31-58-114 ec2-user]#
```

```
[root@ip-172-31-58-114 ec2-user]# sudo vi app.py
[root@ip-172-31-58-114 ec2-user]# cat app.py
#Python Code for your Web Application
import boto3
from flask import Flask, request, jsonify, send_from_directory

app = Flask(__name__)

# Specify your region here
AWS_REGION = 'us-east-2'  # Change to your region

s3 = boto3.client('s3', region_name=AWS_REGION)
sqs = boto3.client('sqs', region_name=AWS_REGION)

ORIGINAL_BUCKET = 'source-bucket-ohio' #Replace with Actual Source Bucket
CONVERTED_BUCKET = 'destinashun-bucket-ohio' #Replace with Actual Destination Bucket
SQS_QUEUE_URL = 'https://sqs.us-east-2.amazonaws.com/851725375246/MessageQueue' #Replace the Queue URL with your Actual URL

@app.route('/')
def index():
    return '''
    <h1>Upload File</h1>
    <form action="/upload" method="post" enctype="multipart/form-data">
        <input type="file" name="file">
        <input type="submit" value="Upload">
    </form>
    '''

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'})
    file = request.files['file']
    if file.filename == '':
        return jsonify({'error': 'No selected file'})
    if file:
```

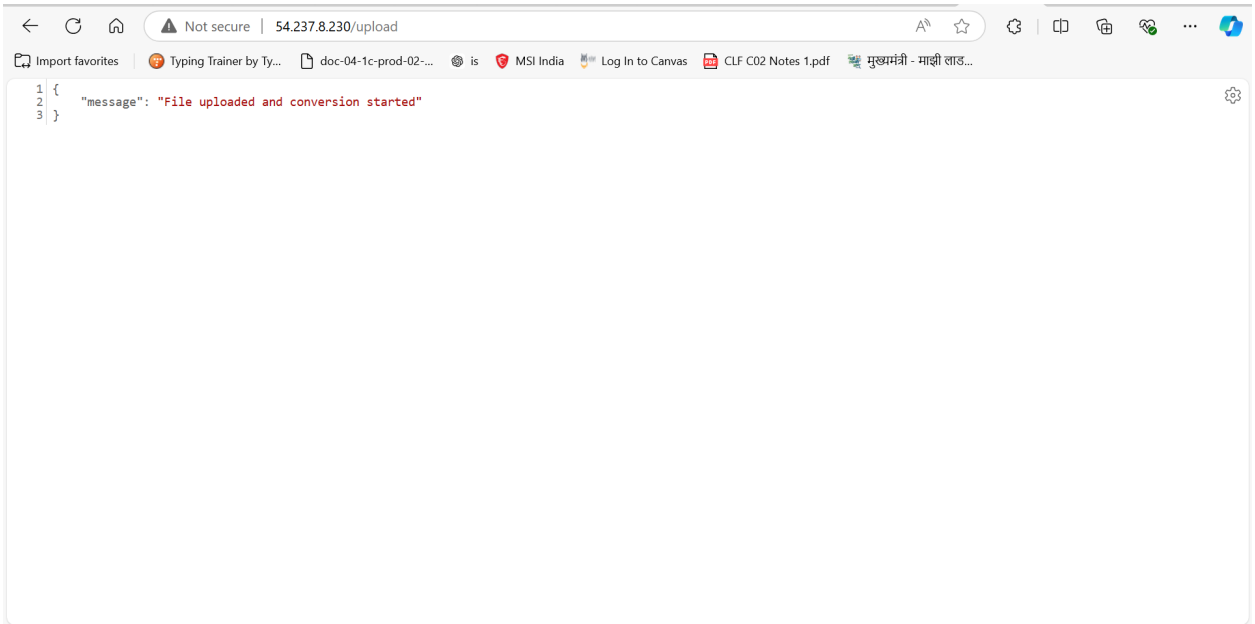
the code run on the public ip of ec2 instance

```
            MessageBody=file.filename
        )
        return jsonify({'message': 'File uploaded and conversion started'})
    return jsonify({'error': 'File upload failed'})

@app.route('/converted/<filename>', methods=['GET'])
def get_converted_file(filename):
    try:
        s3.download_file(CONVERTED_BUCKET, filename, '/tmp/' + filename)
        return send_from_directory('/tmp', filename)
    except Exception as e:
        return jsonify({'error': str(e)})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80)
[root@ip-172-31-58-114 ec2-user]# 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://127.0.0.1:80
 * Running on http://172.31.58.114:80
Press CTRL+C to quit
42.111.115.165 - - [21/Aug/2024 06:58:58] "GET / HTTP/1.1" 200 -
42.111.115.165 - - [21/Aug/2024 06:58:59] "GET /favicon.ico HTTP/1.1" 404 -
```

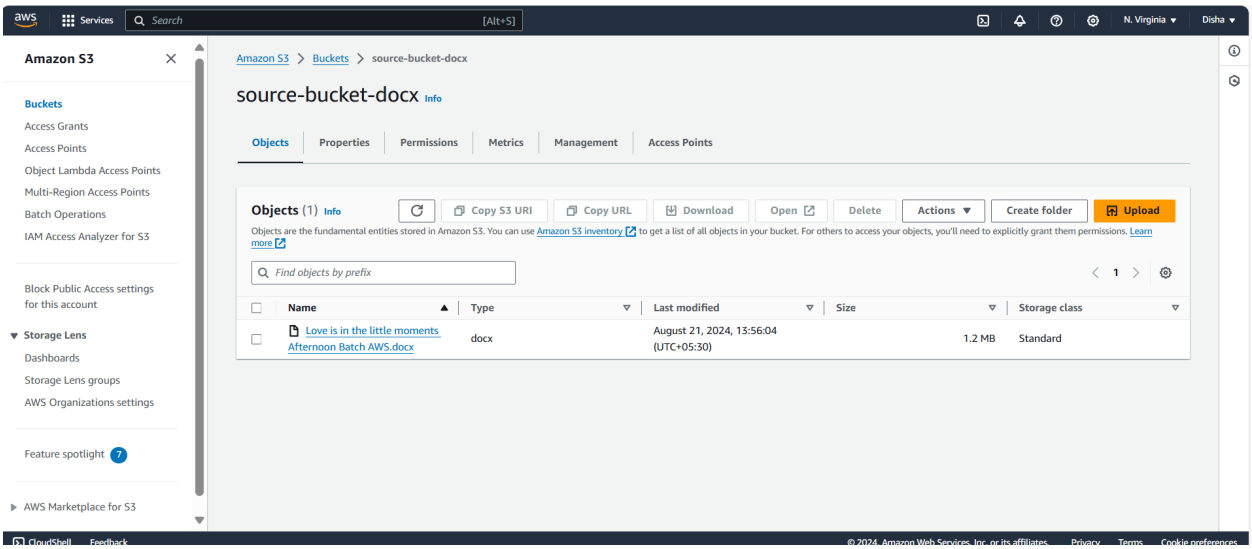
the code run successfully and the GUI is appear



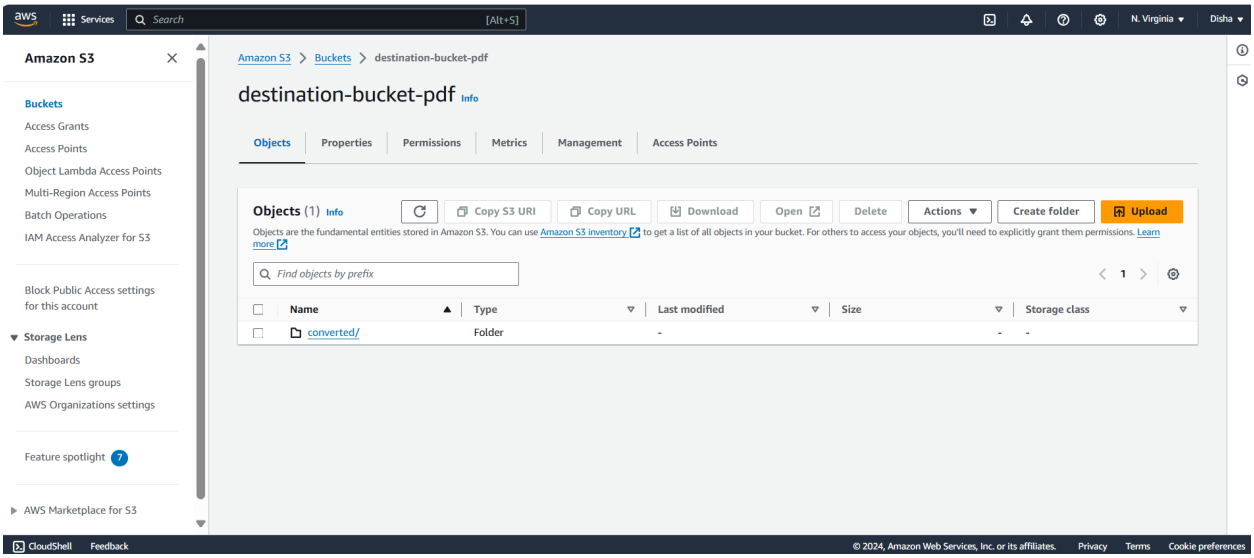
the conversion has been started

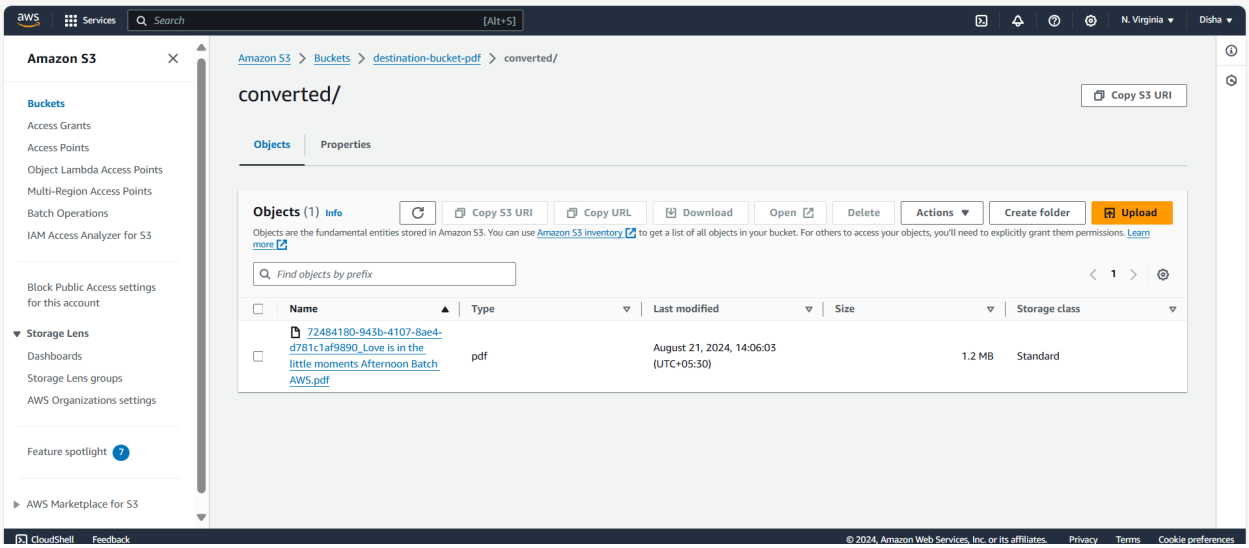


the docx file is successfully uploaded into source bucket

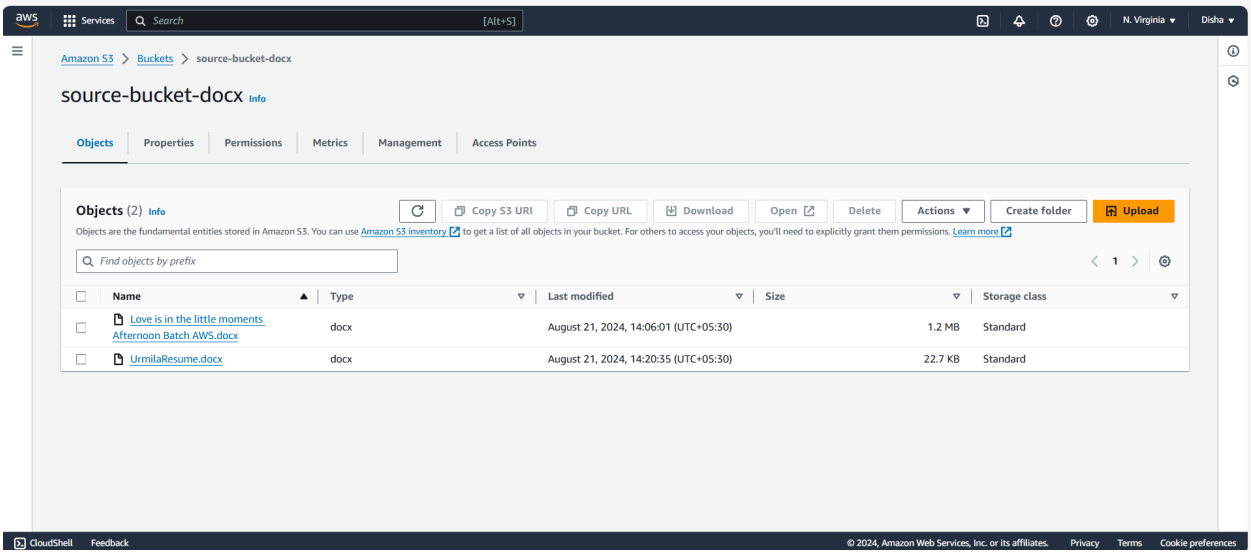


the file is successfully converted into pdf

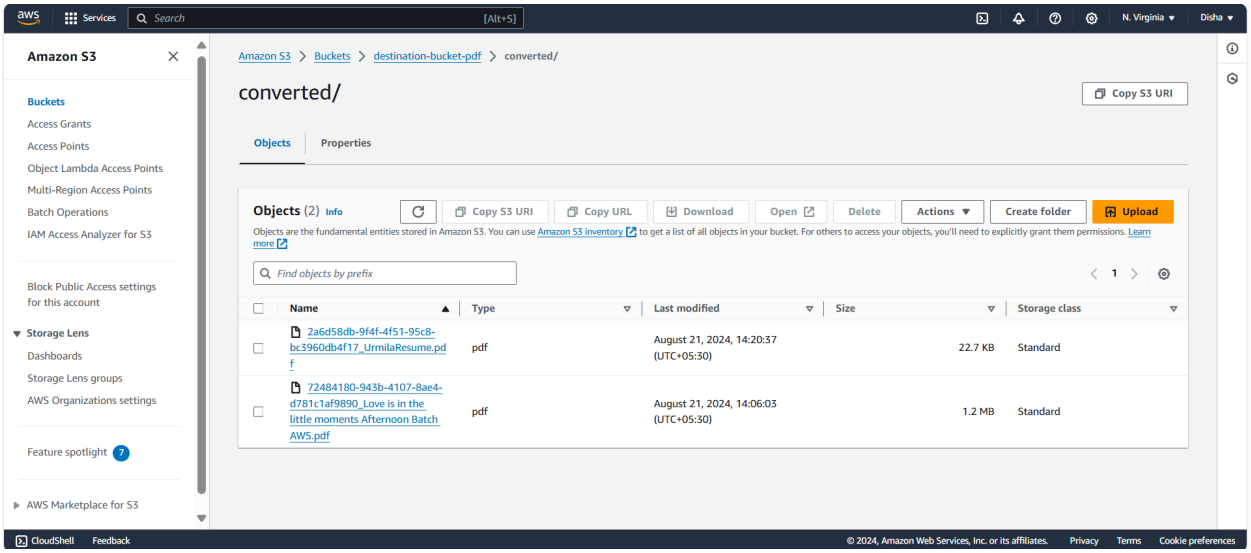




the file is successfully uploaded



the uploaded file succcessfully converted


aws
Services
Search
[Alt+S]
N. Virginia
Disha
Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight 7
AWS Marketplace for S3
Amazon S3 > Buckets > destination-bucket-pdf > converted/
converted/
Copy S3 URI
Objects
Properties
Objects (2) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
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
Find objects by prefix
Name
Type
Last modified
Size
Storage class
2a6d58db-9f4f-4f51-95c8-bc3960db4f17_UrmilaResume.pdf
pdf
August 21, 2024, 14:20:37 (UTC+05:30)
22.7 KB
Standard
72484180-943b-4107-8ae4-d781c1af9890_Love is in the little moments Afternoon Batch AWS.pdf
pdf
August 21, 2024, 14:06:03 (UTC+05:30)
1.2 MB
Standard
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences