| **AWS-Driven File Management and Conversion Application** **Aim:-** The goal is to create and deploy a web application on AWS that enables users to upload files. The uploaded files are processed and converted by a Lambda function, with the results stored in a separate S3 bucket. This setup leverages EC2 for hosting, IAM for access management, S3 for file storage, and SQS for task queuing. **Prerequisites:-**  1. **One EC2 Instance → to show UI** 2. **Two S3 Buckets** 3. **One SQS Queue** 4. **One Lambda Function** 5. **SNS(Optional)**  **Architecture:-**    **Working:-****Create 2 IAM Roles with Permissions:-****For EC2:-** **AmazonS3FullAccess, AmazonSQSFullAccess, AWSLambda\_FullAccess**   **For Lambda:-** **AmazonEC2FullAccess, AmazonS3FullAccess, AmazonSQSFullAccess**   **Creating S3 Buckets:-****Create a S3 bucket(Source-bucket-conversion).**   **Select the Object Ownership as ACLs Disabled.**    **Uncheck the Block Public Access settings from this bucket and click on the acknowledge.**    **Remain the Bucket Versioning & default Encryption as default. Click on Create Bucket.**      **Bucket is created successfully.**    **Generate Policy for the Bucket:-**   **Create Another S3 Bucket(destination-bucket-conversion).**   **Select the Object Ownership as ACLs Disabled.**    **Uncheck the Block Public Access settings from this bucket and click on the acknowledge.**    **Remain the Bucket Versioning & default Encryption as default. Click on Create Bucket.**      **Generate Policy for the Bucket:-**   **Create Lambda Function:-**  1. **Select the function as “Author from Scratch”.** 2. **Provide the function name as “lambda-file-conversion”.** 3. **Select the runtime as “Python 3.9”.** 4. **Architecture as “x86\_64”.**      1. **Change the default execution role to existing role named as “Lambda\_role”(which is created earlier in IAM Roles) and click on create Function.**      1. **Lambda-file-conversion is successfully created.**    **Create EC2 Instance**:-  1. **Name the instance as “Frontend\_conversion”.**      1. **Select the AMI as “Amazon Linux2”.**      1. **Select the Instance Type as “t2.micro”.**      1. **Select the key pair.**      1. **Edit the Network Settings and add SSH, HTTP & HTTPS.**        1. **Then, click on Launch Instance and the Instance is created successfully.**      1. **Add IAM role for EC2:-**   **Select EC2 Instance → Click on Actions → Security → Modify IAM role**     1. **Select the IAM role create for EC2 as “EC2\_role”.**      1. **Successfully attached IAM role.**    **Create SQS Queue:-**  1. **Select the type as “Standard”.** 2. **Provide the name of SQS Queue as “file-conversion-queue”.**      1. **Leave the configuration as default.**      1. **Encryption Disabled.**      1. **Generate the Access Policy Using policy generator or can create custom policy:**      1. **Leave the others as default.**      1. **Successfully created SQS Queue.**    **Set the Event Notification.**  1. **Click on Buckets → source-bucket-conversion → properties.**      1. **Provide the event name.**      1. **Select Event Type as “Put”.**      1. **Select Destination as SQS and select the queue we have made.**      **Add trigger to Lambda Function:-**  1. **Lambda → Add Trigger → Search SQS → Select SQS queue made.**      1. **Click on Add Trigger and the trigger is added to the lambda Function.**    **Conversion Code:-****Add the Conversion Code and Click on Deploy.**  **Connect EC2 and Perform the commands:-**  **sudo yum update -y**  **sudo yum upgrade -y**  **sudo yum install python3 -y**  **sudo yum install python3-pip -y**    **sudo pip3 install Flask**  **sudo pip3 install boto3**  **sudo nano app.py**  **app.py**   **After adding the code, press ctrl+o(not zero) to save the file → Click Enter to save the file name as it is → ctrl+x to exit from GNU nano 5.8****sudo python3 app.py**      **Output:-****Upload File successfully.**        **S3 source-bucket-conversion has the .docx file which we have uploaded in the previous step.**    **S3 destination-bucket-conversion has the .pdf file which was converted by the lambda function and then uploaded here.**      **After converting the file, we have to revert the .pdf file back to the screen which is opened in the EC2 Instance and a download button is given to download the .pdf file as a “converted.zip” file.**  **After clicking download button, the modal is prompting that the file is downloaded successfully.**  **Download successfully done.**  **Conclusion:-** The AWS-driven File Management and Conversion Application effectively demonstrates how various AWS services can be integrated to build a scalable and efficient solution. By utilizing EC2 for hosting the web application, IAM for secure access management, S3 for reliable file storage, SQS for task queuing, and Lambda for file processing, the project showcases a robust architecture for handling file uploads and conversions. This approach not only streamlines file processing workflows but also ensures high availability and scalability. Overall, the project highlights the potential of AWS services in creating a seamless and automated file management system. |
| --- |