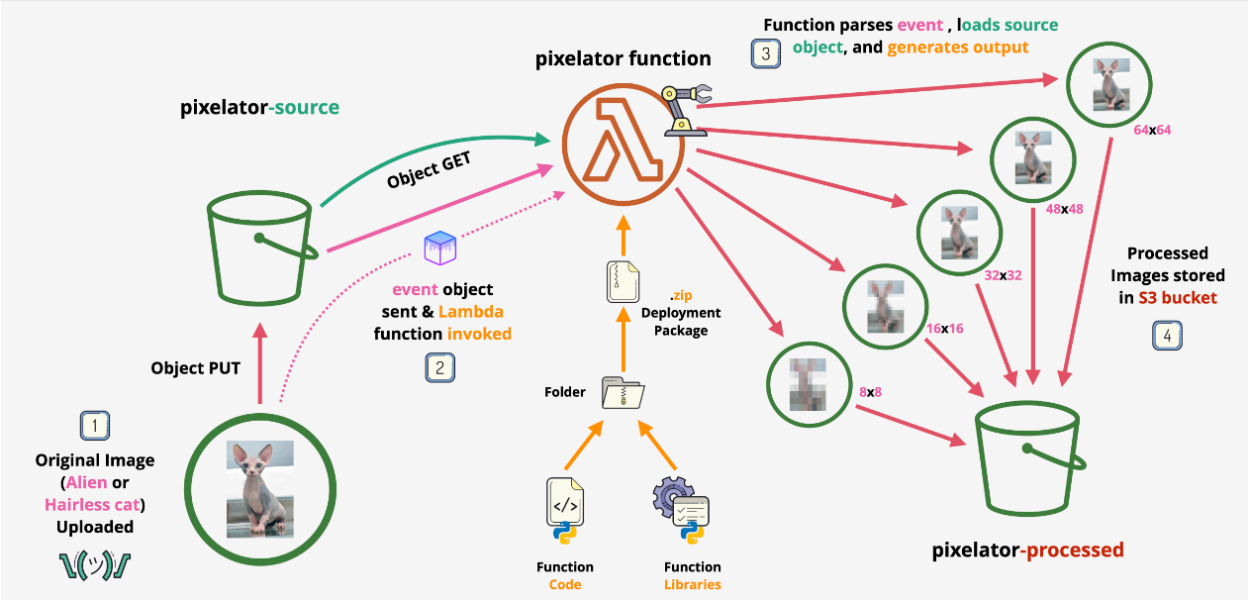
Project Architecture



Use-case's

1.Image Compression: Can be used by Social Media Platforms to Compress images and Store them at a lesser size as compared to original size.

2. Automated Workflow:

- **Real-Time Processing:** Use AWS Lambda to process images in real-time as they are uploaded to S3, automatically applying pixelation without manual intervention.
- **Scalable Solutions:** Handle large volumes of images efficiently, making it suitable for businesses or platforms with high image upload rates.

3. Privacy Protection:

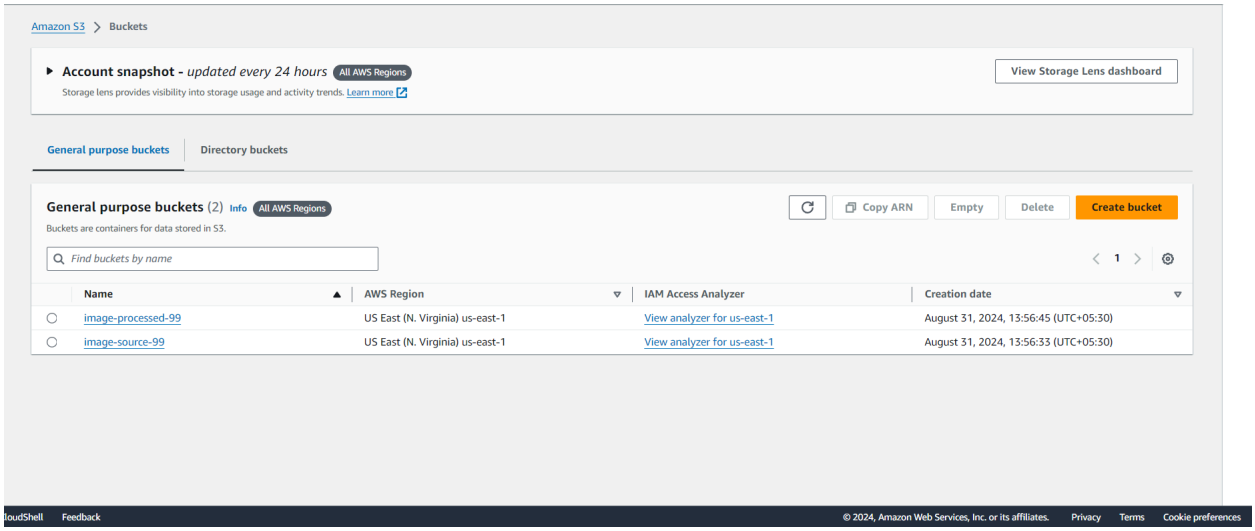
- **Social Media:** Automatically pixelate sensitive information in images before posting to social media, ensuring user privacy.
- **Healthcare:** Anonymize patient data in medical images to comply with privacy regulations such as HIPAA.

Stage 1 - Create the S3 Buckets

We will be creating `2` buckets, all settings apart from region and bucket name can be left as default.

Click `Create Bucket` and create a bucket in the format of unique-name-`source` in the `us-east-1` region

Click `Create Bucket` and create a another bucket in the format of unique-name-`processed` also in the `us-east-1` region
These names will need to be unique



Stage 2 - Create the Lambda Role

1. Move to the IAM Console.
2. Click Roles, then Create Role
3. For `Trusted entity type`, pick **`AWS service`**
4. For the service to trust pick **`Lambda`** then click `Next` , `Next` again
5. For `Role name` put `PixelatorRole(can be anything)` then Create the role

6. Click `PixelatorRole`

7. Under `Permissions Policy` we need to add permissions and it will be an `inline policy`

8. Click `JSON` and delete the contents of the code box entirely.

9. Copy this policy from the following link

   https://github.com/ZABROL/AWS-Image_Pixelator.git

10. Copy the entire contents into your clipboard and paste into the previous permissions policy code editor box
    Locate the words `REPLACEME` there should be `4` occurrences, 2 each for the source and processed buckets .. and for each of those one for the bucket and another for the objects in that bucket.

11. Replace the term `REPLACEME` with the name you picked for your buckets above, in my example it is `dontusethisname`

Important Note - Make the necessary changes in the policy.

```
"Resource":[
        "Arn:aws:s3:::dontusethisname-processed",
#(Replace this with bucket name that will store processed image)

        "arn:aws:s3:::dontusethisname-processed/*",
#(Replace this with bucket name that will store processed image)

        "arn:aws:s3:::dontusethisname-source/*",
 #(Replace this with bucket name that will store your Soucre image)

        "Arn:aws:s3:::dontusethisname-source"
#(Replace this with bucket name that will store your Sourceimage)

]
```

Stage 3 - Create the Lambda Function

1. Move to the lambda console
2. Click `Create Function`
3. We're going to be `Authoring from Scratch`
4. For `Function name` enter `pixelator (Can be Anything)`
5. for `Runtime` select **`Python 3.9`**
6. For `Architecture` select **`x86_64`**

7. For `Permissions` expand `Change default execution role` pick `Use an existing role` and in the `Existing role` dropdown, pick `PixelatorRole`

8. Then `Create Function`

9. Close down any `notifcation` dialogues/popups

10. Click `Upload from` and select `.zip file` download this zip to your local machine

https://github.com/ZABROL/AWS-Image_Pixelator.git To download Zip File.

11. On the lambda screen, click `Upload` locate and select that .zip, and then click the `Save` Button.

12. This upload will take a few minutes, but once complete you might see something saying `The deployment package of your Lambda function "pixelator" is too large to enable inline code editing. However, you can still invoke your function.` which is OK :)

Stage 4 - Configure the Lambda Function & Trigger

1. **Click `Configuration` tab and then `Environment variables`**

2. We need to add an environment variable telling the pixelator function which processed bucket to use, it will know the source bucket because it's told about that in the event data.

3. **Click `Edit` then `Add environment variable`**, under Key put `processed_bucket` and for `Value` put the bucket name of *your* processed bucket. As an example `dontusethisname-processed` (but use *your* bucket name)

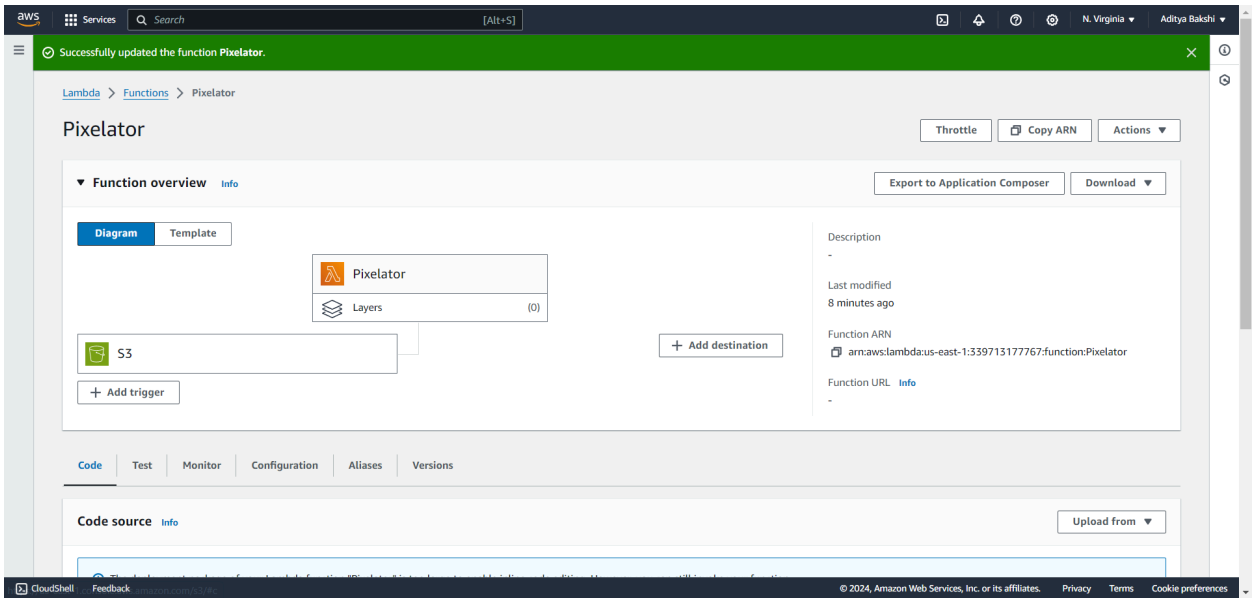
4. **Be `really really really` sure you put your `processed` bucket here and *NOT* your source bucket.**

5. ***if you use the source bucket here, the output images will be stored in the source bucket, this will cause the lambda function to run over and over again ... bad be super-sure to put your processed bucket*** **Click** `Save`

6. **Click** **`General configuration`** then click `Edit` and change the timeout to `1` minutes and `0` seconds, then click `Save`

7. **Click** **`Add trigger`**
   In the dropdown pick **`S3`**
   Under `Bucket` pick your *source* bucket ... *AGAIN* be really really sure this is your source bucket and *NOT* your destination bucket and *NOT* any other bucket. Only pick your *SOURCE* bucket here.

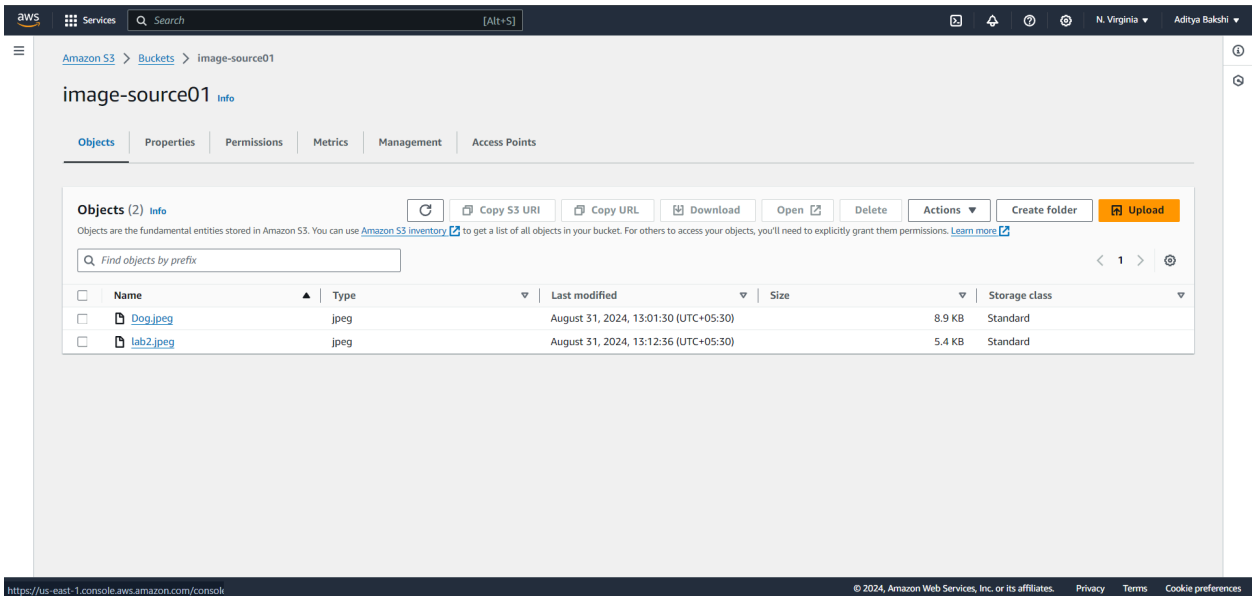
8. You will need to check the `Recursive invocation` acknowledgment box, this is because this lambda function is invoked every time anything is added to the *source* bucket, if you configure this wrongly, or configure the environment variable above wrongly ... it will run the lambda function over and over again *for ever*. Once checked, click `Add`



Stage 5 - Test and Monitor

1.) In one tab open your `-source` bucket & in the other open the `-processed' bucket

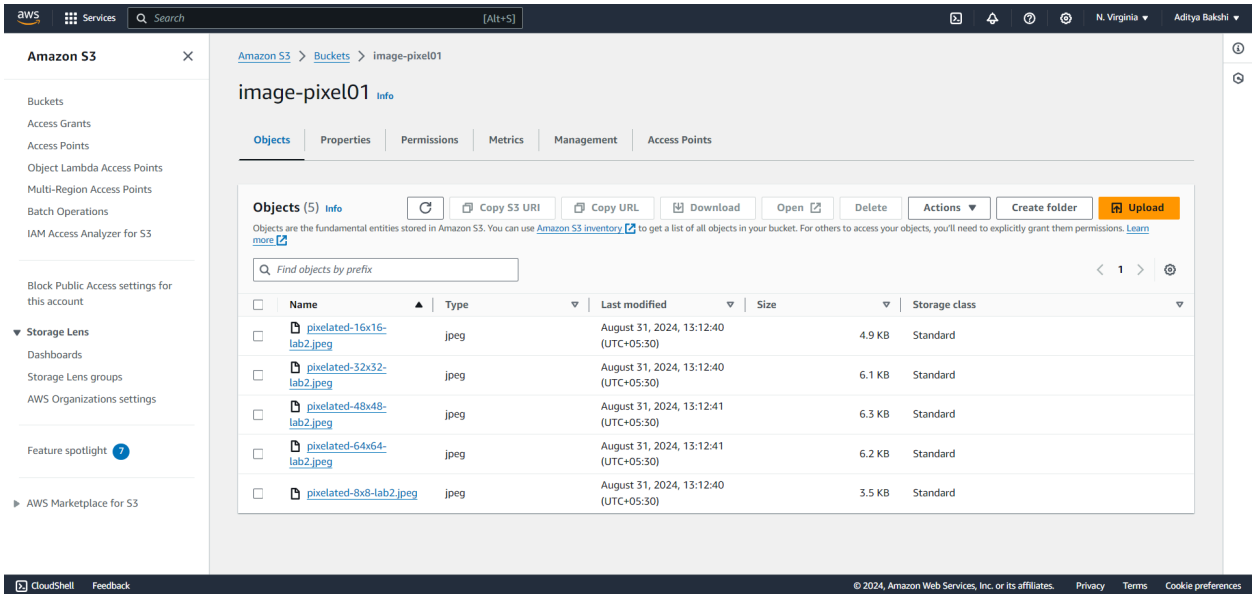
2.) In the `-source` bucket tab, make sure to select the `Objects` tab and click `Upload`
Add some files and click `Upload`



`Source Bucket`

3.) Go to the S3 Console tab for the `-processed` bucket

4.) Click the `Refresh` icon
Select each of the pixelated versions of the image ... you should have 5 (`8x8`, `16x16`, `32x32`, `48x48` and `64x64`)



Processed Bucket

5.) Click `Open`

6.) You browser will either open or save all of the images

7.) Open them one by one, starting with `8x8` and finally `64x64` in order ... notice how they are the same image, but less and less pixelated :)

Stage 6 - Cleanup

1. Open the `pixelator` lambda function
2. Delete the function
3. Move to the IAM Roles console
4. Click `PixelatorRole`, then `Delete` the role, then confirm the deletion.
5. Go to the `S3 Console`
   For each of the `source` and `processed` buckets do:

- Select the bucket.
- Click `Empty`.
- Type `permanently delete`, and `Empty`.
- Close the dialogue and move back to the main S3 Console.
- Make sure the bucket is still selected, click `Delete`.
- Type the name of the bucket then delete the bucket.

Thank-You