

Azubi Project 2: Vividart Studio - Beanstalk2

≡ Author	Olatunji Olayinka Oluwaseun
≡ Contributors	Brian Mathenge, Adade Sedom Percy, and Pauline Andege Omondi
≡ GitHub Repository	https://github.com/olatunji-weber/MineVividarts_studio.git

Vividart Studio Project

This project is to build an image processor utilizing python, lambda, S3 and docker.

I collaborated with Brian Mathenge, Adade Sedom Percy, and Pauline Andege Omondi in doing the project.

First of all, I forked the repo from: https://github.com/olatunji-weber/Vividarts_studio.git

Then cloned it to workstation via CLI

```
> git clone https://github.com/olatunji-weber/Vividarts_studio.git
```

Then created “requirements.txt” file and entered the line:

```
boto3==1.34.16  
Flask==3.0.0
```

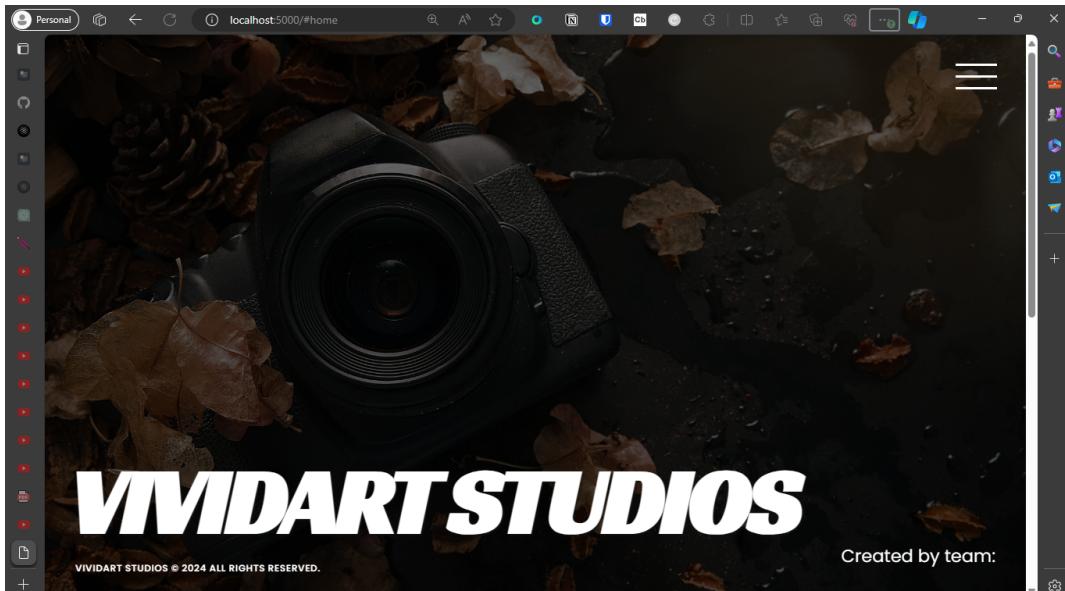
Then used “pip install requirements.txt” at the command prompt.

```

PS C:\Users\Olatunji Olayinka\Documents\FormerHDD\Programming\Node-React-NPM&More\IntroToNode&Express\AzubiParole\Azubi2023Work-CloudClass\AzubiProject\Vividarts_studio-Project2\Vividarts_studio> pip install -r requirements.txt
Collecting Flask==2.1.1 (from -r requirements.txt (line 1))
  Downloading Flask-2.1.1-py3-none-any.whl.metadata (3.9 kB)
Collecting Werkzeug>=2.0 (from Flask==2.1.1-> requirements.txt (line 1))
  Downloading werkzeug-2.0.1-py3-none-any.whl.metadata (4.1 kB)
Collecting Jinja2>=3.0 (from Flask==2.1.1-> requirements.txt (line 1))
  Downloading Jinja2-3.1.3-py3-none-any.whl.metadata (3.3 kB)
Collecting itsdangerous>=2.0 (from Flask==2.1.1-> requirements.txt (line 1))
  Downloading itsdangerous-2.1.2-py3-none-any.whl.metadata (2.9 kB)
Collecting click>=8.0 (from Flask==2.1.1-> requirements.txt (line 1))
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Requirement already satisfied: colorama in c:\users\olatunji\olayinka\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from click>=8.0->Flask==2.1.1-> requirements.txt (line 1)) (0.4.4)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.0->Flask==2.1.1-> requirements.txt (line 1))
  Downloading MarkupSafe-2.1.5-cp311-cp311-win_amd64.whl.metadata (3.1 kB)
Downloaded Flask-2.1.1-py3-none-any.whl (95 kB)
      95.2/95.2 kB 194.0 kB/s eta 0:00:00
Downloaded click-8.1.7-py3-none-any.whl (97 kB)
      97.9/97.9 kB 267.4 kB/s eta 0:00:00
Downloaded itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Downloaded Jinja2-3.1.3-py3-none-any.whl (133 kB)
      133.2/133.2 kB 187.5 kB/s eta 0:00:00
Downloaded werkzeug-3.0.1-py3-none-any.whl (226 kB)
      226.7/226.7 kB 223.6 kB/s eta 0:00:00
Downloaded MarkupSafe-2.1.5-cp311-cp311-win_amd64.whl (17 kB)
Installing collected packages: MarkupSafe, itsdangerous, click, Werkzeug, Jinja2, Flask
WARNING: The script flask.exe is installed in 'C:\Users\Olatunji\Olayinka\AppData\Local\Programs\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\python311\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed Flask-2.1.1 Jinja2-3.1.3 MarkupSafe-2.1.5 Werkzeug-3.0.1 click-8.1.7 itsdangerous-2.1.2
PS C:\Users\Olatunji\Olayinka\Documents\FormerHDD\Programming\Node-React-NPM&More\IntroToNode&Express\AzubiParole\Azubi2023Work-CloudClass\AzubiProject\Vividarts_studio-Project2\Vividarts_studio>

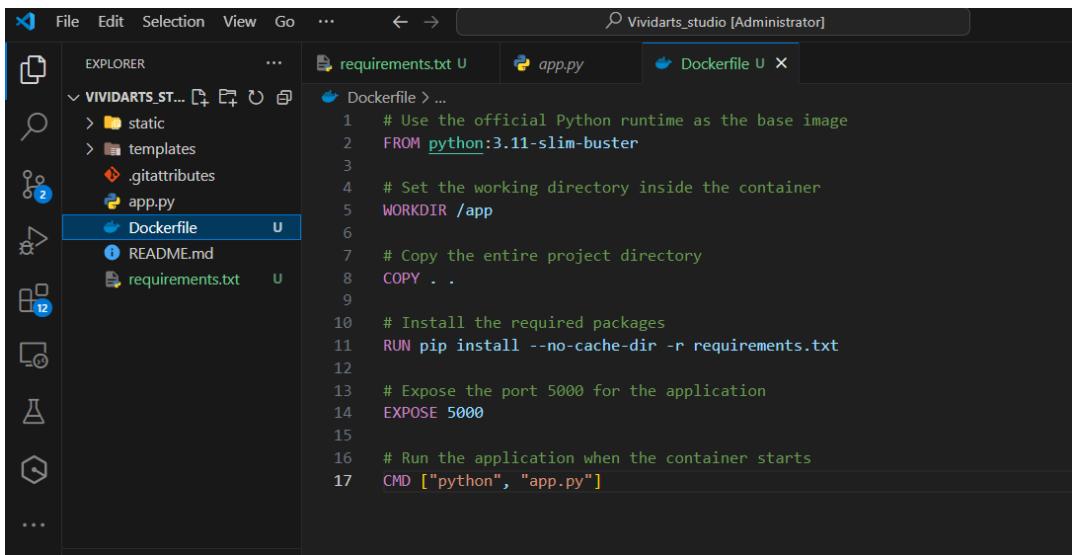
```

Ran “python `app.py`” in order to launch the application. Then observed the following page when accessing localhost via port 5000 in a web browser:



Containerized the cloned application by creating Dockerfile for Flask application, via these steps:

Created a file called Dockerfile in the root directory of the project in order to containerize the application (right next to `app.py`, `requirements.txt`, etc.).



```
File Edit Selection View Go ... ← → ⌂ Vividarts_studio [Administrator]
EXPLORER ...
VIVIDARTS_ST...
> static
> templates
.gitattributes
app.py
Dockerfile U
README.md
requirements.txt U
Dockerfile > ...
1 # Use the official Python runtime as the base image
2 FROM python:3.11-slim-buster
3
4 # Set the working directory inside the container
5 WORKDIR /app
6
7 # Copy the entire project directory
8 COPY . .
9
10 # Install the required packages
11 RUN pip install --no-cache-dir -r requirements.txt
12
13 # Expose the port 5000 for the application
14 EXPOSE 5000
15
16 # Run the application when the container starts
17 CMD ["python", "app.py"]
```

Used the official Python runtime as the base image

```
FROM python:3.11-slim-buster
```

Set the working directory inside the container

```
WORKDIR /app
```

Copied the entire project directory

```
COPY . /app
```

Installed the required packages

```
RUN pip install --no-cache-dir -r requirements.txt
```

Exposed the port 5000 for the application

```
EXPOSE 5000
```

Added the command to run the application when the container starts

```
CMD ["python", "app.py"]
```

After creating the Dockerfile, I built and ran the Docker container using the following commands:

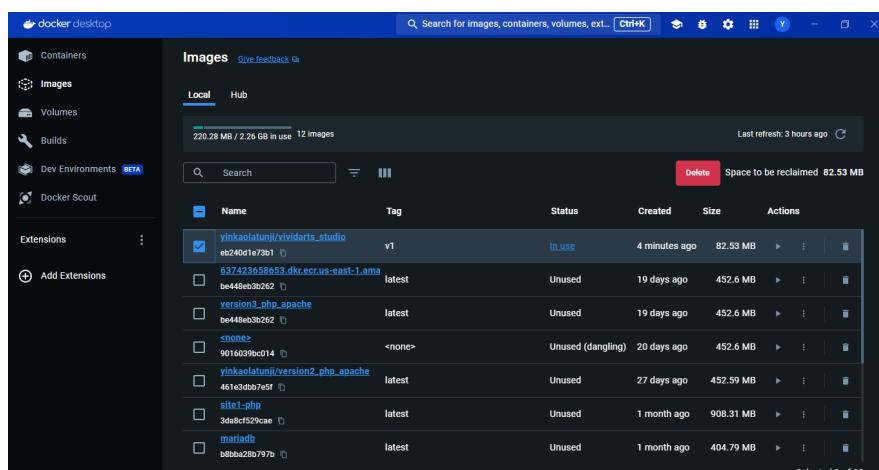
Build the Docker image:

```
docker build -t yinkaolatunji/vividarts_studio:v1 .
```

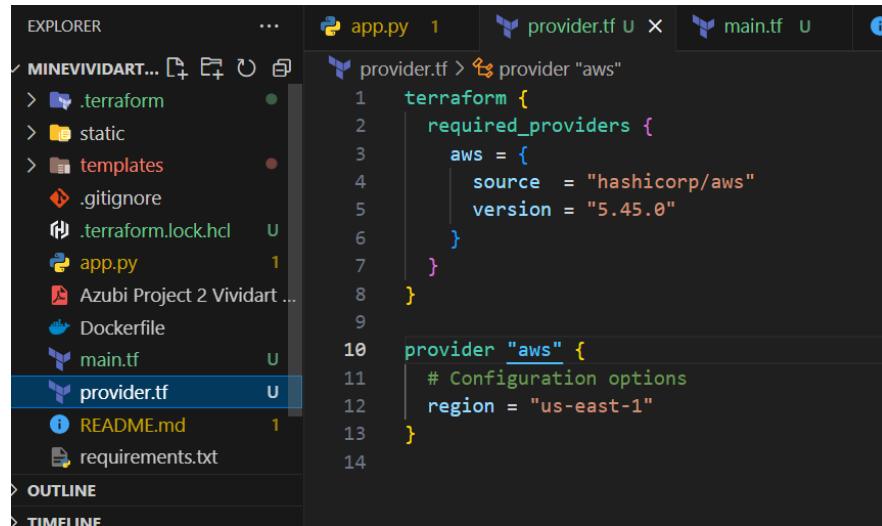
Run the Docker image:

```
docker run -d -p 5000:5000 yinkaolatunji/vividarts_studio:v1
```

Here is the created docker image in Docker Desktop:



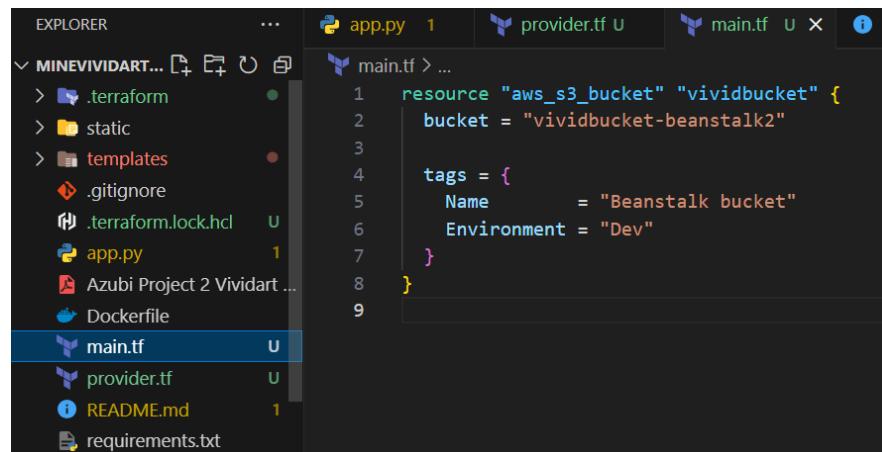
Then I created an S3 bucket named "vividbucket-beanstalk2" which is where we'll be uploading the image to using Terraform. Here is the "provider.tf" file specifying the provider to be used to create the infrastructure.



```
provider.tf > provider "aws"
1  terraform {
2    required_providers {
3      aws = {
4        source  = "hashicorp/aws"
5        version = "5.45.0"
6      }
7    }
8  }

10 provider "aws" {
11   # Configuration options
12   region = "us-east-1"
13 }
```

Here is the "main.tf" file specifying the s3 bucket to create.



```
main.tf > ...
1 resource "aws_s3_bucket" "vividbucket" {
2   bucket = "vividbucket-beanstalk2"
3
4   tags = {
5     Name      = "Beanstalk bucket"
6     Environment = "Dev"
7   }
8 }
```

Then, run "terraform init" and "terraform apply -auto-approve" after you ensure that your shell prompt is configured with AWS credentials.

```

-----
profile          <not set>      None    None
access_key        *****shared-credentials-file*****
secret_key        *****jY7Z shared-credentials-file*****
region           us-east-1      config-file     /aws/config
PS C:\Users\Olatunji\Downloads\Components\AWS\Programming\Node-React-NPM&More\IntroToNode&Express\AzubiParole\Azubi2023Work-CloudClass\AzubiProjects\VividArtsStudio> terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
+ create
Terraform will perform the following actions:
# aws:s3bucket vividbucket will be created
+ aws:s3bucket "vividbucket"
  + acceleration_status = (known after apply)
  + acl                  = (known after apply)
  + arn                  = (known after apply)
  + bucket               = "vividbucket-beanstalk2"
  + bucket_domain_name   = (known after apply)
  + bucket_website        = (known after apply)
  + bucketRegional_domain_name = (known after apply)
  + force_destroy         = false
  + hosted_zone_id       = (known after apply)
  + id                   = (known after apply)
  + object_lock_enabled   = (known after apply)
  + policy                = (known after apply)
  + region                = (known after apply)
  + request_payer         = (known after apply)
  + tags
    + "Environment" = "Dev"
    + "Name"        = "Beanstalk bucket"
  + tags_all            = {
      + "Environment" = "Dev"
      + "Name"        = "Beanstalk bucket"
    }
  + website_domain      = (known after apply)
  + website_endpoint    = (known after apply)

Plan: 1 to add, 0 to change, 0 to destroy.
aws:s3bucket vividbucket: Creating...
aws:s3bucket vividbucket: Creation complete after 8s [id=vividbucket-beanstalk2]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

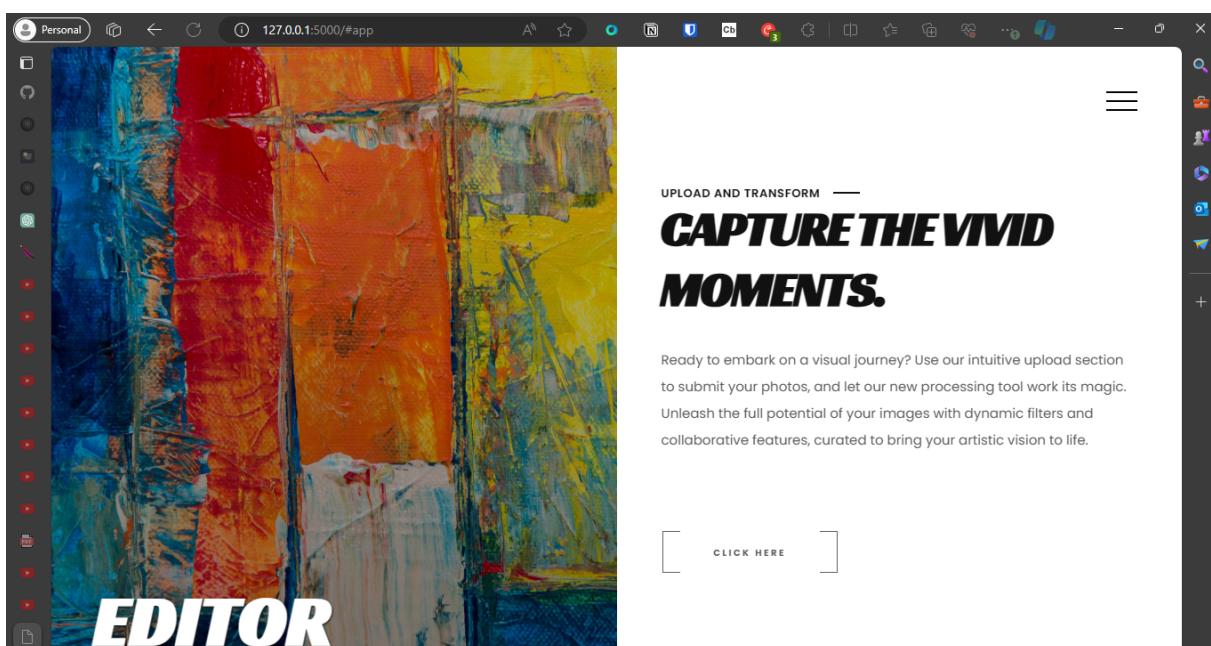
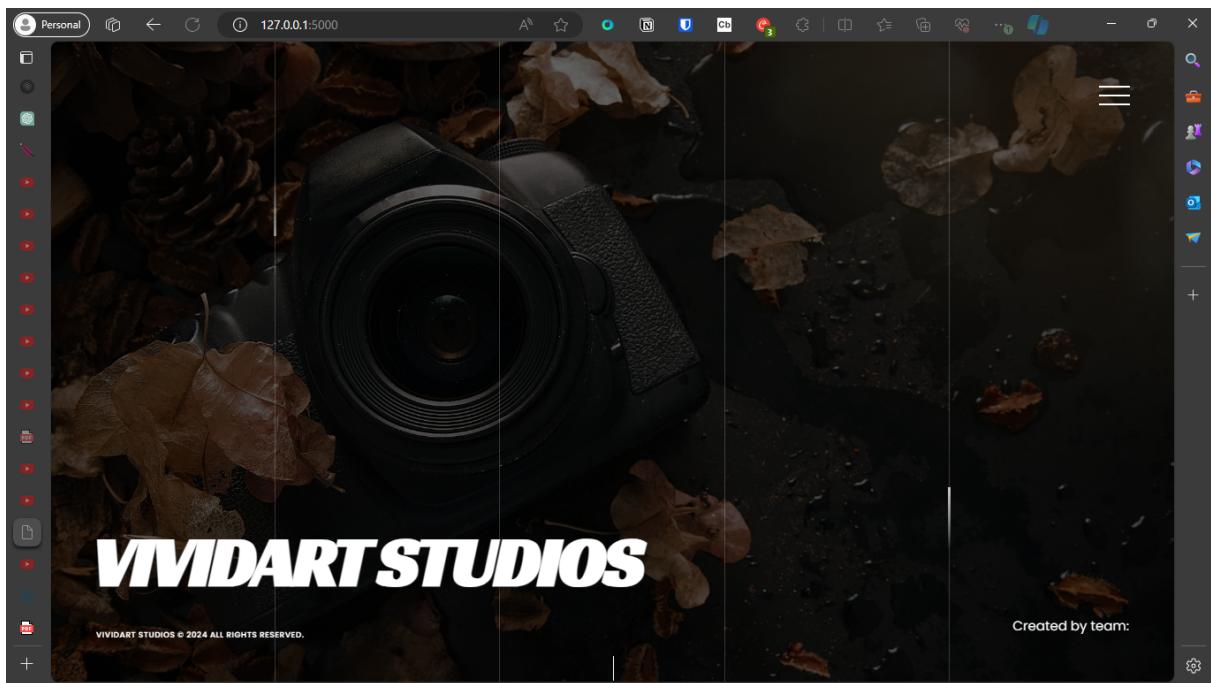
```

And this is the S3 bucket that was created as a result.

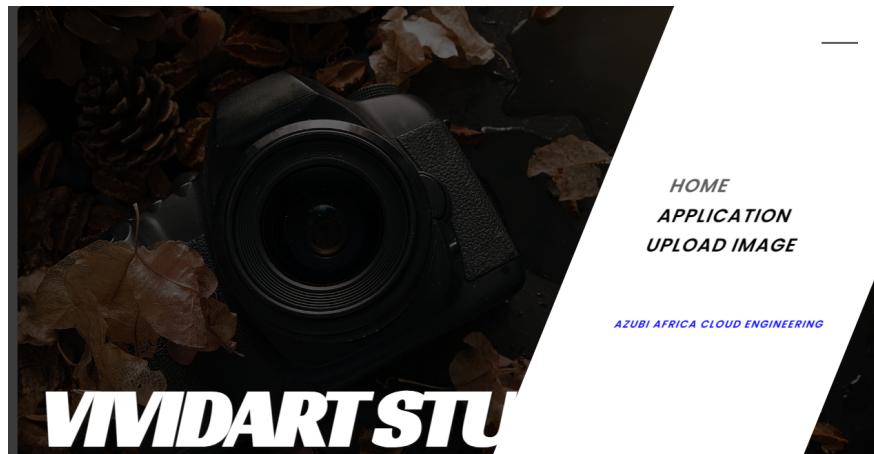
The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with navigation links like 'Buckets', 'Storage Lens', and 'Feature spotlight'. The main area displays summary statistics: 'Total storage 169.3 KB' and 'Object count 1'. Below this, there are tabs for 'General purpose buckets' and 'Directory buckets', with 'General purpose buckets' selected. A table lists the bucket details:

Name	AWS Region	IAM Access Analyzer
vividbucket-beanstalk2	US East (N. Virginia) us-east-1	View analyzer for us-east-1

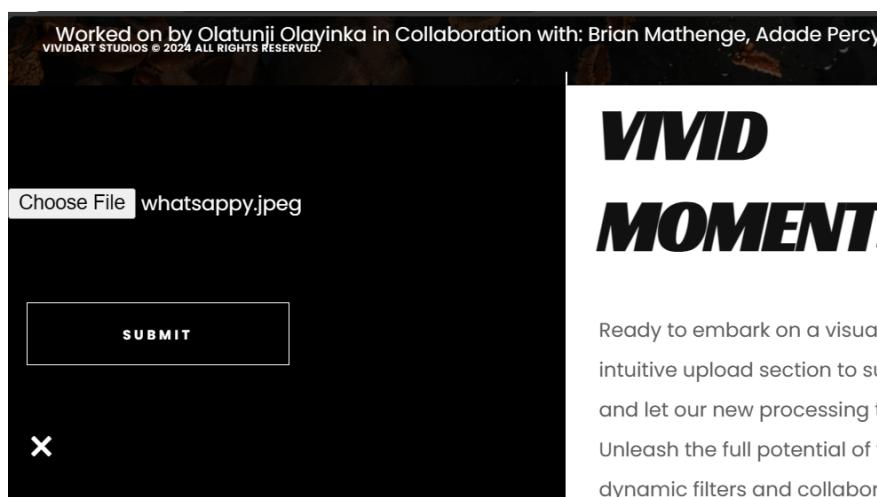
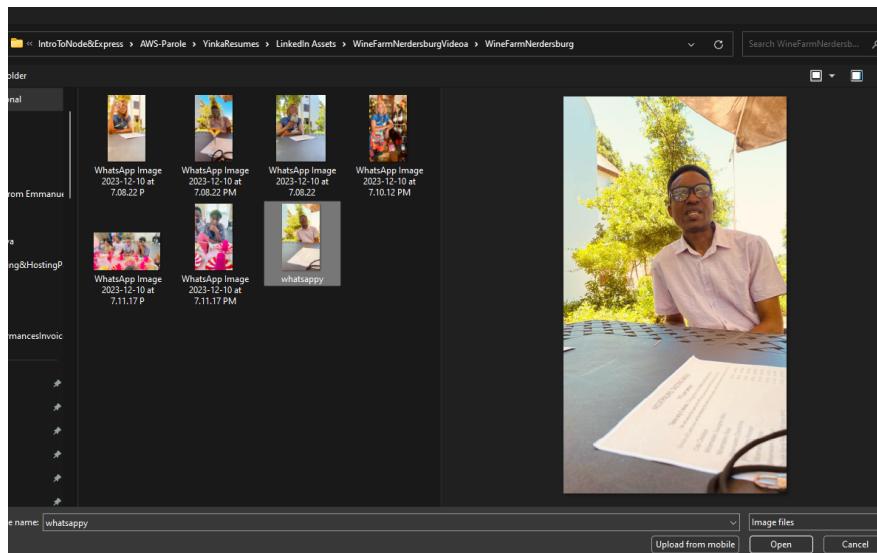
Now, you can access the Containerized Flask application via your web browser by visiting <http://localhost:5000>



Added link to be able to go to the upload page in the navigation menu:



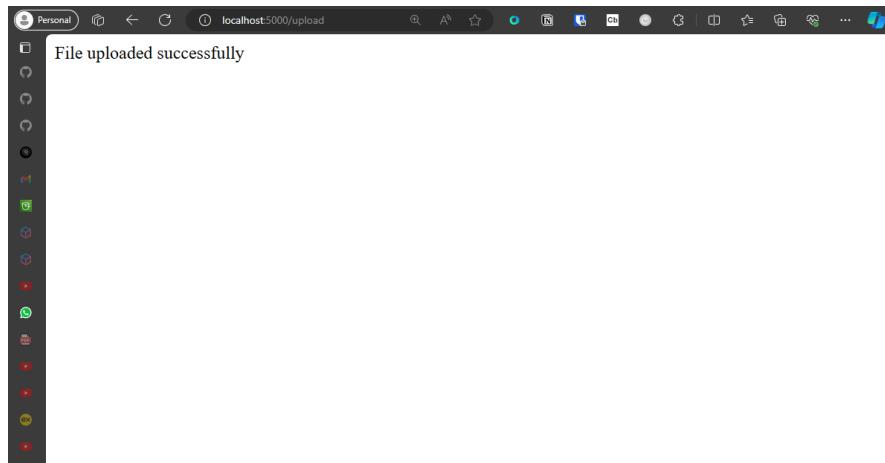
Selected an image to upload:



VMD
MOMENTS

Ready to embark on a visual
intuitive upload section to su
and let our new processing tr
Unleash the full potential of y
dynamic filters and collaborc

And then Clicked the SUBMIT button which then displayed a message saying that the image is uploaded successfully.



Then I checked to confirm that the S3 bucket named "**vividbucket-beanstalk2**" really contains the uploaded image.

A screenshot of the AWS S3 console. The navigation bar shows 'Amazon S3 > Buckets > vividbucket-beanstalk2'. The left sidebar has sections for 'Buckets', 'Access Grants', 'Access Points', etc. The main panel shows 'Objects (1) Info' with a table. The table has one row with the following data:

Name	Type	Last modified	Size	Storage class
whatsappy.jpeg	jpeg	April 12, 2024, 12:07:59 (UTC+02:00)	84.0 KB	Standard

Then I pushed the container image to Docker Hub using this command:

```
>docker push yinkaolatunji/vividarts_studio:v1
```

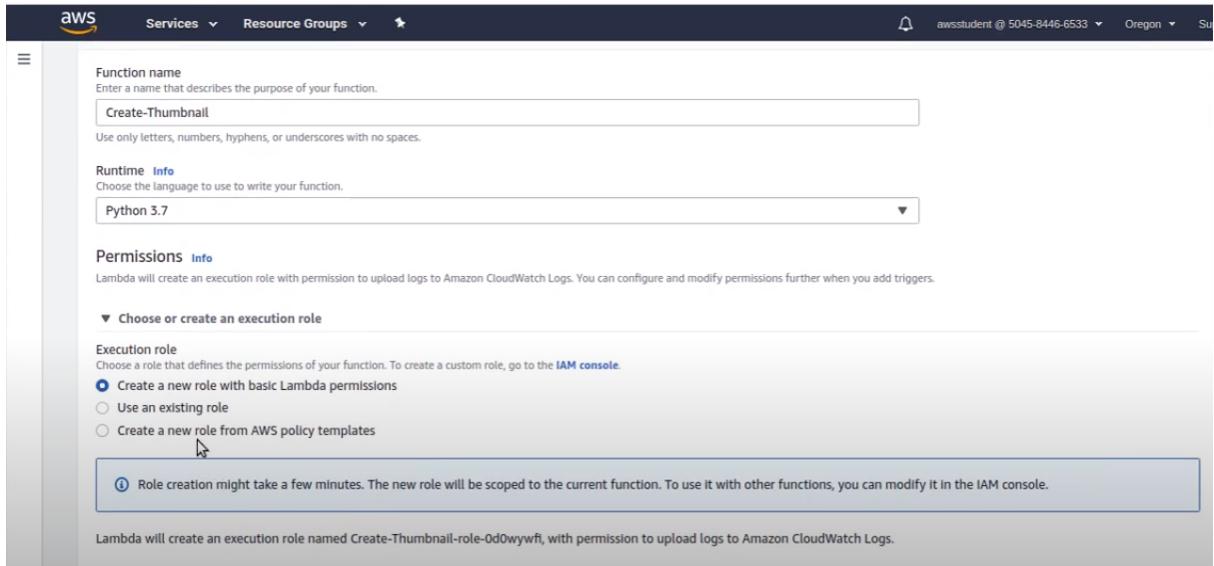
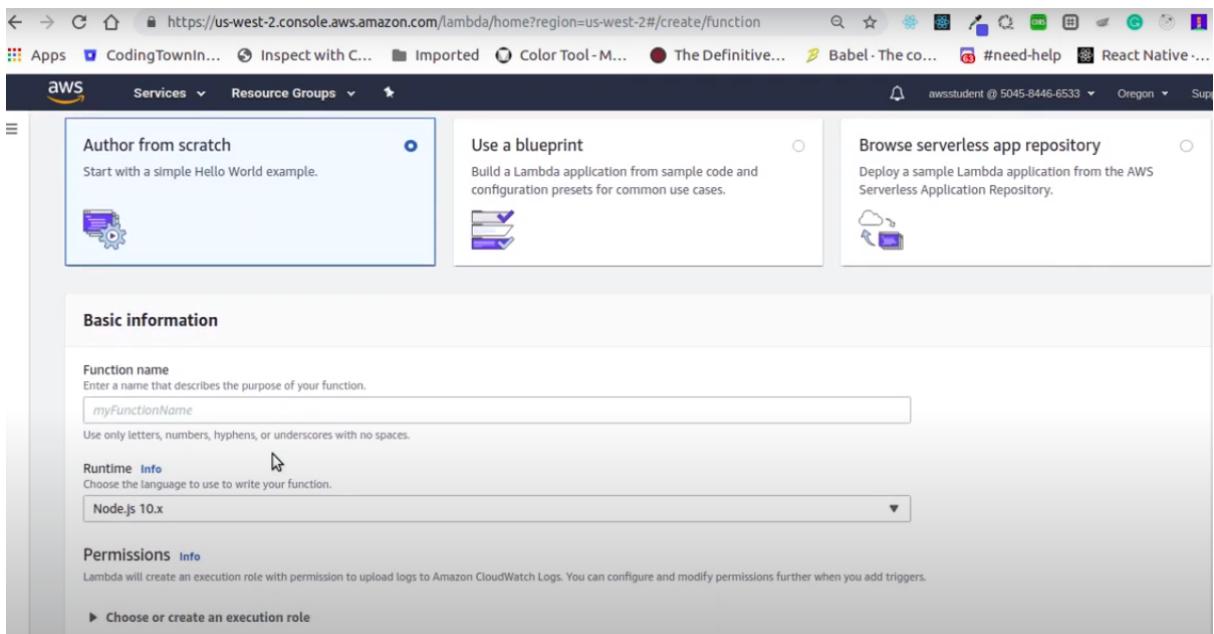
The screenshot shows the Docker Hub interface for a repository named `yinkaolatunji/vividarts_studio`. The repository was updated 7 days ago and has one tag, `v1`, which is an Image type. The repository does not have a description. The Docker commands section shows the command `docker push yinkaolatunji/vividarts_studio:tagname`. The Automated Builds section indicates that manually pushing images to Hub is available with Pro, Team, and Business subscriptions.

Creating Lambda function to process the uploaded image.

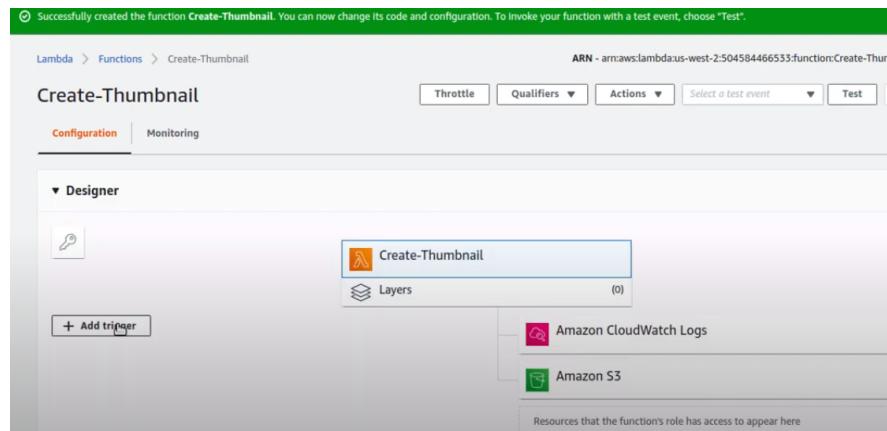
In order to create a Lambda Function to automatically process the image that is uploaded by resizing it using some Python Code:

Go to the Lambda service in the AWS management console and click "Create Function".

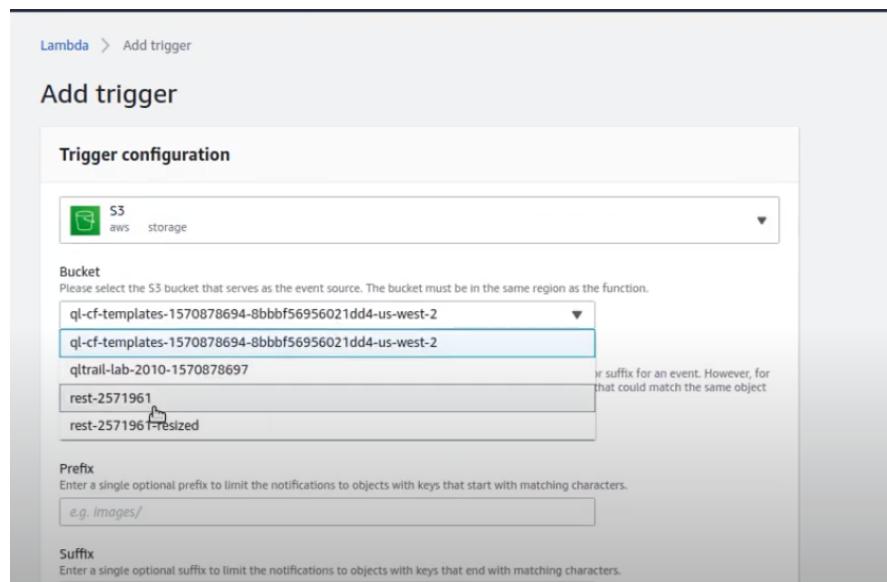
The screenshot shows the AWS Lambda service in the AWS Management Console. The 'Functions' tab is selected, displaying a table with no data. The table columns are: Function name, Description, Runtime, Code size, and Last modified. A 'Create function' button is located at the top right of the table area.



Then I added the Trigger



Selected S3 service and the Bucket name. Then clicked Add.



Then, I ran “terraform destroy” in order to deprovision the S3 bucket that was created using Terraform earlier:

```

- "Environment" = "Dev"
- "Name"      = "Beanstalk bucket"
} -> null

- grant {
  - id          = "7d208ca2eecbc08f9c09e673ab789e0c5be97e55f21346cad70ad6728e55d610" -> null
  - permissions = ["FULL_CONTROL"]
  ] -> null
- type        = "CanonicalUser" -> null
}

- server_side_encryption_configuration {
  - rule {
    - bucket_key_enabled = false -> null
    - apply_server_side_encryption_by_default {
      - sse_algorithm = "AES256" -> null
    }
  }
}

- versioning {
  - enabled   = false -> null
  - mfa_delete = false -> null
}
}

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_s3_bucket.vividbucket: Destroying... [id=vividbucket-beanstalk2]
aws_s3_bucket.vividbucket: Destruction complete after 2s

Destroy complete! Resources: 1 destroyed.
PS C:\Users\Olatunji Olayinka\Documents\FormerHDD\Programming\Node-React-NPM&More\IntroToNode&Express\AzubiParole\Azubi2023Work-CloudClass\AzubiProjects\VividArtStudio-2ndProject\Main\Vividarts_studio>

```