Assignment 7: AWS; hands-on activity II
Name: Nithin Shastry Madhusudhana
User ID: nimadhu

Absolutely! Here's a summary for each of those AWS services:

Amazon EC2 (Elastic Compute Cloud)

EC2 provides resizable compute capacity in the cloud, allowing users to rent virtual servers to run applications.

Key Features:
  - Scalability: Easily scale computing capacity up or down based on demand.
  - Variety of Instances: Offers a wide range of instance types optimized for various use cases (e.g., memory-intensive, CPU-intensive).
  - Control: Users have full control over their virtual servers, including OS, security settings, and networking.
- Benefits:
  - Flexibility: Adjust computing resources based on workload fluctuations.
  - Cost-Efficiency: Pay only for the capacity used, reducing upfront infrastructure costs.
  - Ease of Use: Simple interface for launching and managing virtual servers.

AWS S3 (Simple Storage Service)

- Purpose: S3 offers scalable object storage in the cloud, enabling users to store and retrieve vast amounts of data.
- Key Features:
  - Durability and Availability: Designed for durability and high availability.
  - Scalability: Scales effortlessly to accommodate any amount of data.
  - Security and Access Control: Provides robust security features and access control mechanisms.
- Benefits:
  - Cost-Effective Storage: Pay for what you use without worrying about managing underlying infrastructure.
  - Versatility: Ideal for various use cases like backup and restore, content distribution, and data archiving.
  - Integration: Integrates seamlessly with other AWS services and third-party tools.

RDS (Relational Database Service)
- Purpose: RDS simplifies the setup, operation, and scaling of relational databases in the cloud.
- Key Features:
  - Multiple Database Engines: Supports various popular databases like MySQL, PostgreSQL, Oracle, SQL Server, etc.
  - Automated Backups and Maintenance: Manages backups, software patching, and routine maintenance tasks.
  - Scalability: Allows for easy scaling of compute and storage resources based on demand.
- Benefits:
  - Ease of Management: Reduces administrative burden with automated tasks.
  - Reliability: Ensures high availability and durability of databases.
  - Security: Offers robust security features for data protection.

CloudFormation
- Purpose: CloudFormation is an AWS service for provisioning and managing AWS infrastructure resources using templates.
- Key Features:
  - Infrastructure as Code (IaC): Allows defining infrastructure in templates using JSON or YAML.
  - Automation: Automates resource provisioning and configuration.
  - Version Control and Rollback: Tracks changes and allows for easy rollback to previous infrastructure states.
- Benefits:
  - Consistency and Reproducibility: Ensures consistent setup across environments.
  - Time and Cost Savings: Reduces manual intervention and accelerates infrastructure deployment.
  - Scalability: Scales effortlessly to manage complex infrastructure setups.

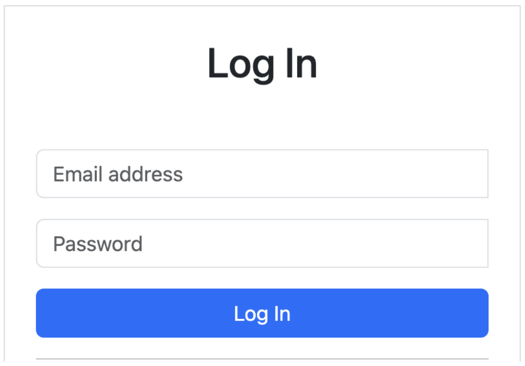
These services collectively form a powerful suite within AWS, catering to different aspects of computing, storage, database management, and infrastructure provisioning in the cloud.
[References - https://docs.aws.amazon.com]

SCENARIO

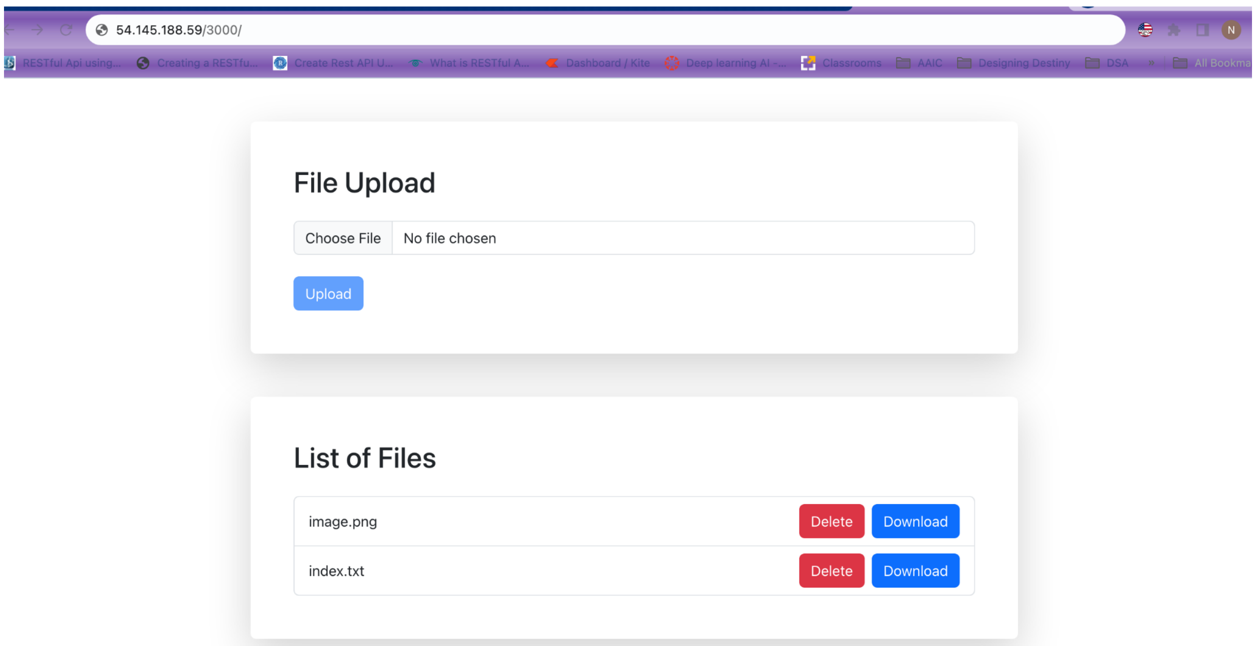
WEB APP - FILE KART

File kart is one place store where a user can store all kinds of files. I have utilized
- AWS S3 bucket to store and retrieve images.
- AWS RDS to store user credentials
- AWS CloudFormation to create VPC
- AWS EC2 instance to deploy the application
- AWS IMS to manage infrastructure related resources

Basically application login looks like this - user credentials stored using AWS RDS MySQL DB.



Application page looks like this -


54.145.188.59/3000/

File Upload

Choose File | No file chosen

Upload

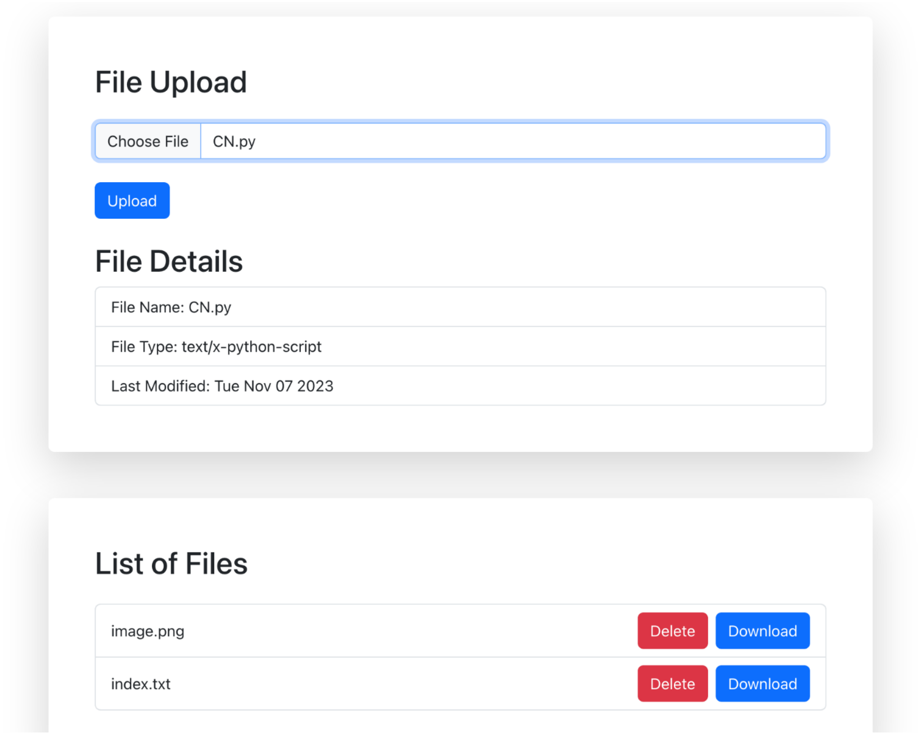
List of Files

image.png	Delete	Download
index.txt	Delete	Download

As you can see above -
http://54.145.188.59/3000/ - is the public IP address through my EC2 instance where application is hosted

File Upload option - To upload file (any format)
Upon uploading - file gets stored in AWS S3, I am retrieving data from S3 and user can delete file or download file locally based on user's preference.

Upon File upload - User can see file details as below



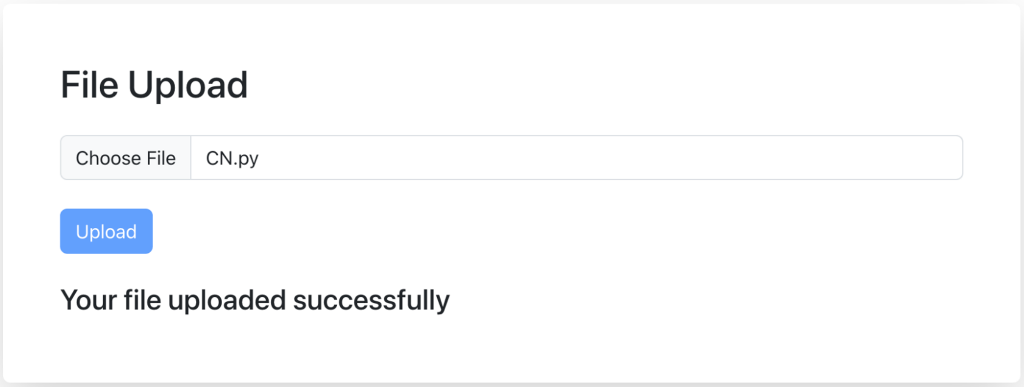
File Details

File Name: CN.py
File Type: text/x-python-script
Last Modified: Tue Nov 07 2023

List of Files

image.png	Delete	Download
index.txt	Delete	Download

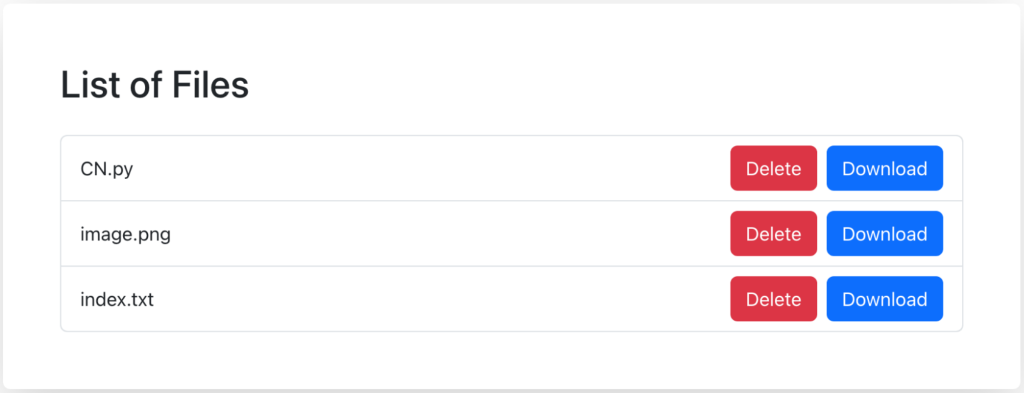
After clicking on Upload we can see file details in List of Files section (CN.py)



File Upload

Choose File CN.py

Your file uploaded successfully

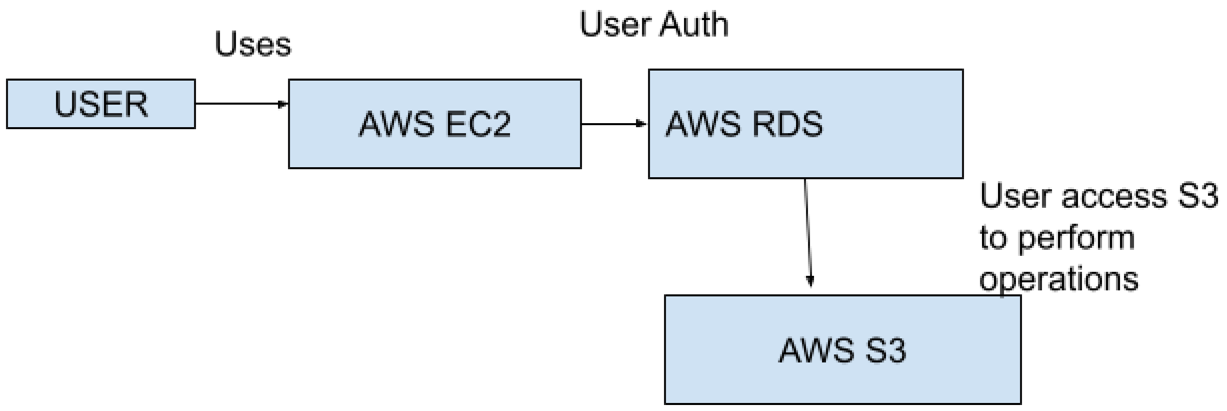


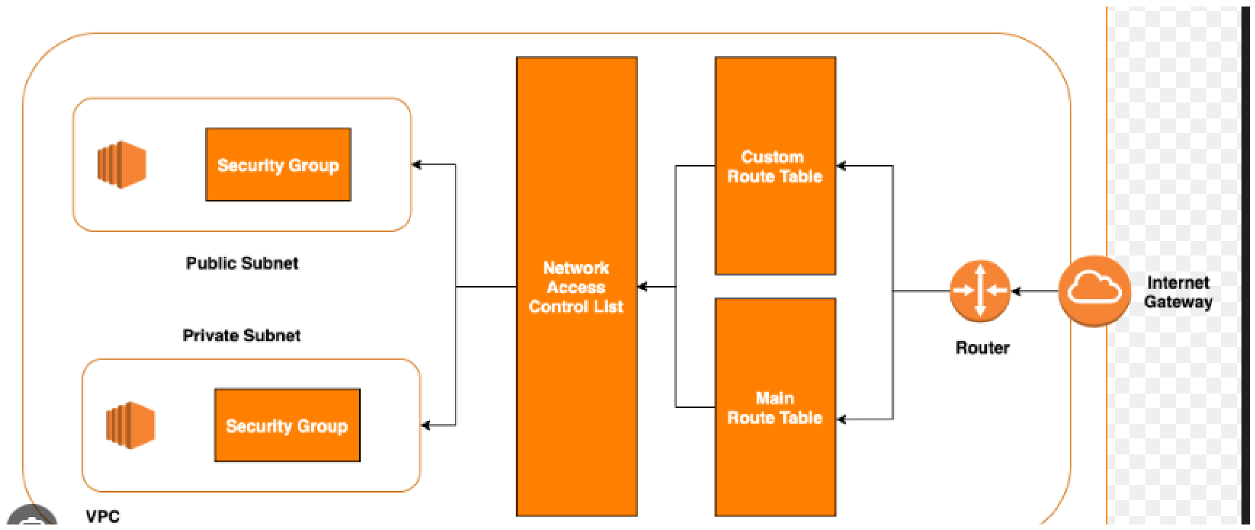
List of Files

CN.py	Delete	Download
image.png	Delete	Download
index.txt	Delete	Download

Similarly Files can be deleted and downloaded from the S3 bucket.

ARCHITECTURE DESIGN

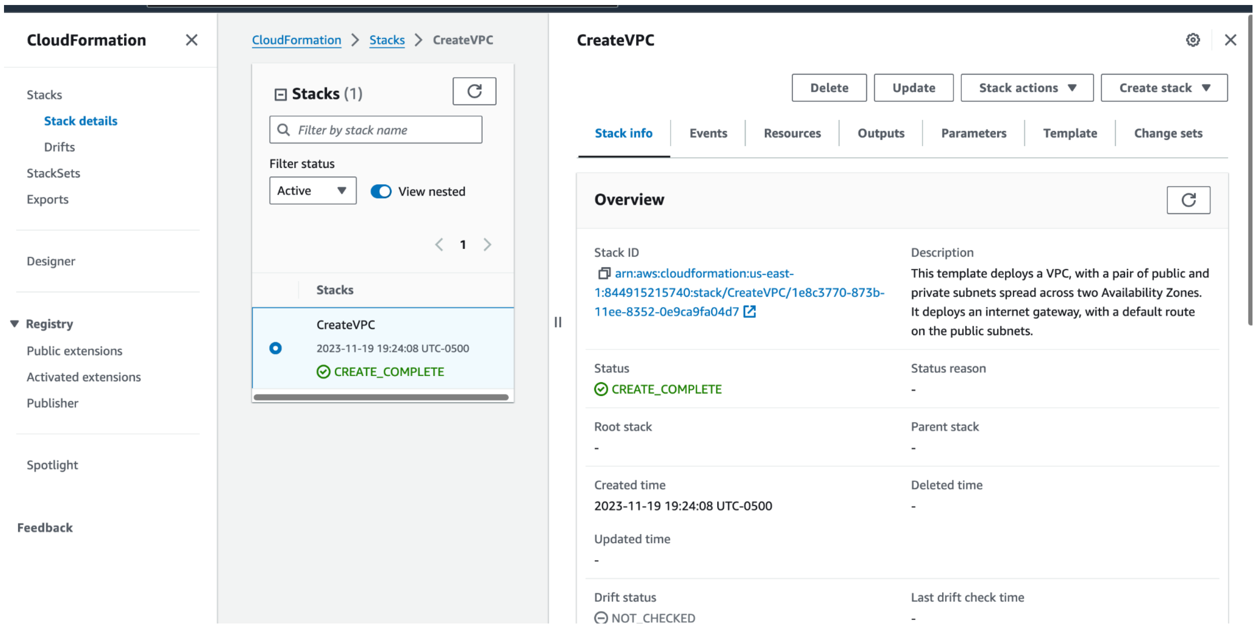




VPC - diagram [https://levelup.gitconnected.com/creating-a-custom-vpc-in-aws-b4ea7bf4a71]

STEP BY STEP DETAILS

1. VPC USING CLOUDFORMATION



Stacks

CreateVPC
2023-11-19 19:24:08 UTC-0500
CREATE_COMPLETE

CreateVPC

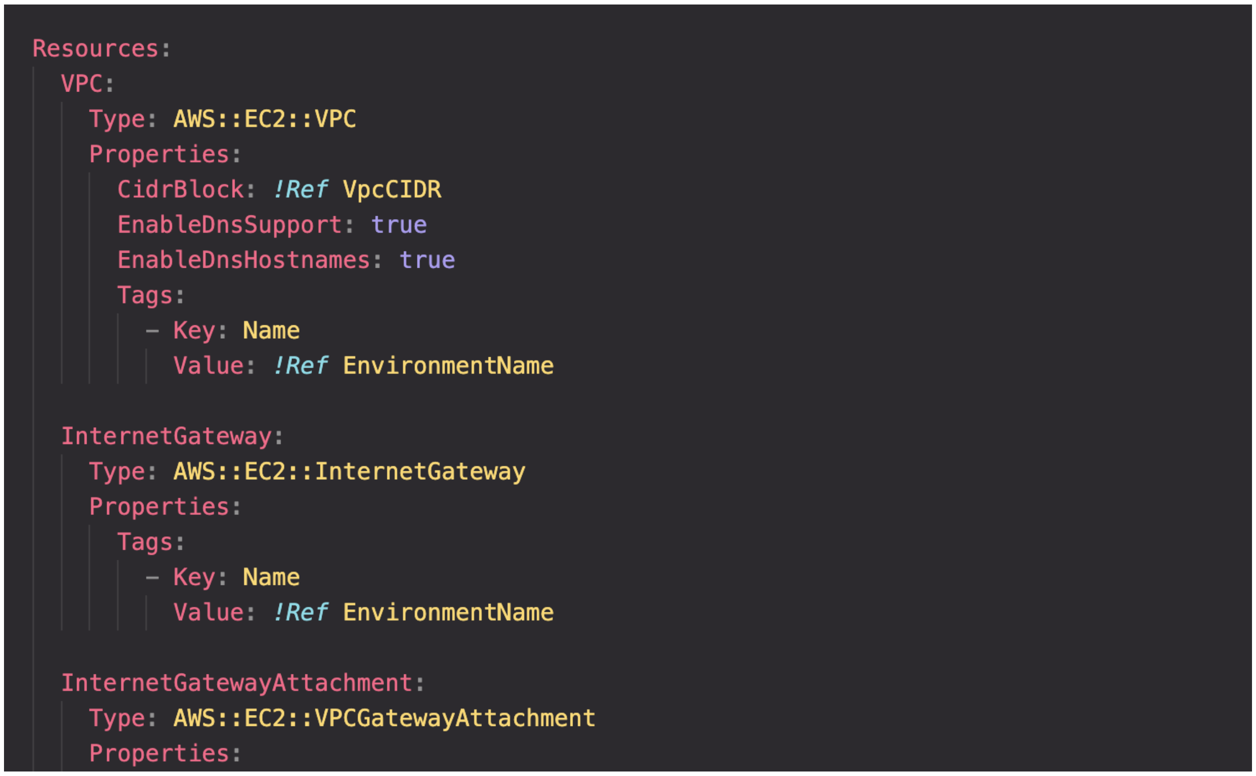
Overview

Stack ID arn:aws:cloudformation:us-east-1:844915215740:stack/CreateVPC/1e8c3770-873b-11ee-8352-0e9ca9fa04d7	Description This template deploys a VPC, with a pair of public and private subnets spread across two Availability Zones. It deploys an internet gateway, with a default route on the public subnets.
Status CREATE_COMPLETE	Status reason -
Root stack -	Parent stack -
Created time 2023-11-19 19:24:08 UTC-0500	Deleted time -
Updated time -	
Drift status NOT_CHECKED	Last drift check time -

STEPS

1. Define Your CloudFormation Template

Create a CloudFormation template (written in JSON or YAML) that describes the VPC configuration.



```yaml
Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
```

This template creates a VPC with a CIDR block of `10.0.0.0/16`, enables DNS support, and assigns a name tag.

2. Access AWS Management Console
Log in to the AWS Management Console.

3. Navigate to CloudFormation Service
Go to the AWS CloudFormation service.

4. Create Stack
- Click on "Create Stack."
- Choose "Template is ready" and "Upload a template file."
- Upload the template file you created.

5. Specify Stack Details
- Enter a stack name.
- Optionally, provide parameters if your template has defined any.

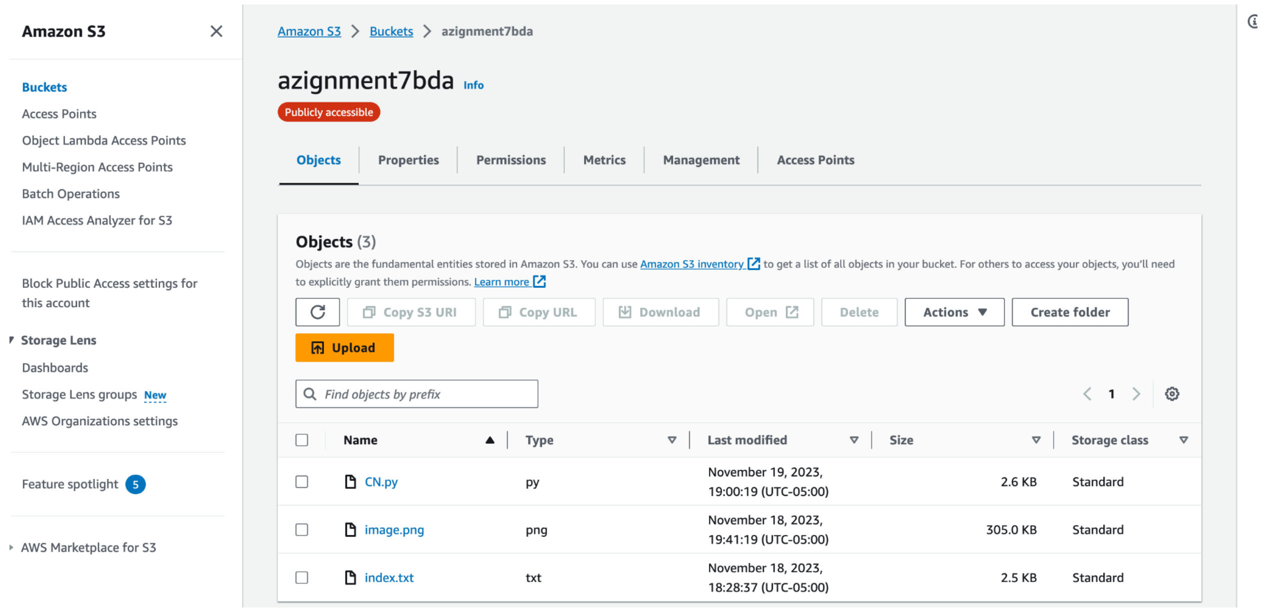
6. Configure Stack Options
You can set stack options such as tags, permissions, and advanced settings as needed.

7. Review and Create
Review the details you've entered for the stack.
- Click "Create stack" to initiate the provisioning process.

8. Monitor Stack Creation
Once the stack creation process begins, you can monitor its progress in the CloudFormation console. Wait for the stack to reach the "CREATE_COMPLETE" status.

9. Access Your VPC
After the stack creation is complete, navigate to the EC2 Dashboard or the VPC Dashboard to confirm that your VPC has been created with the specified configuration.

2. S3 bucket for storing Images



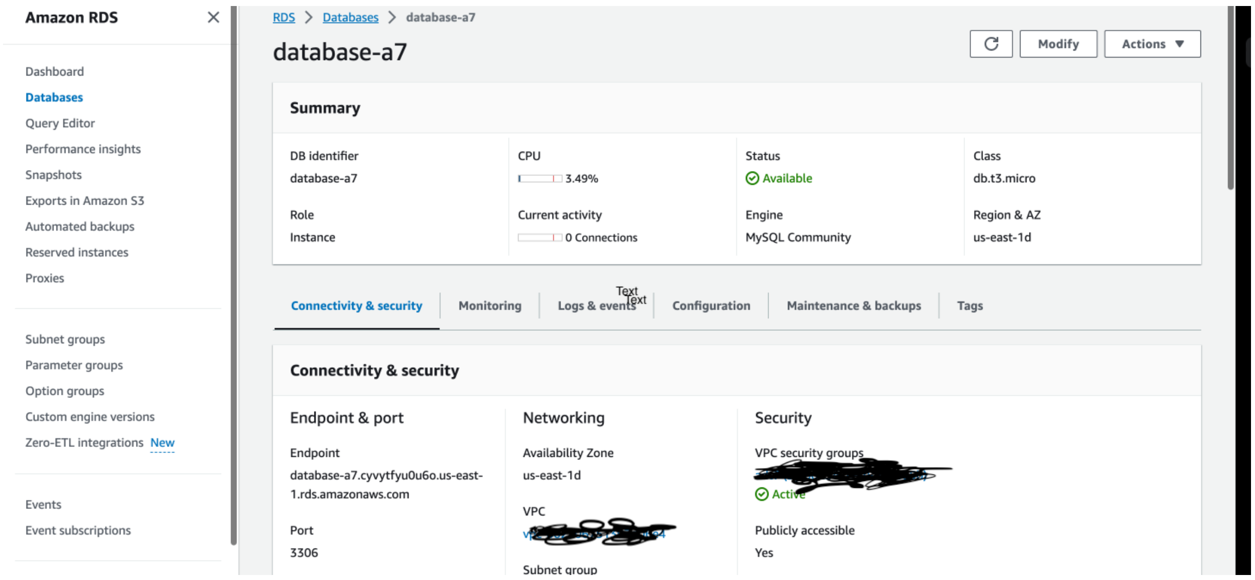
Objects (3)

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

Name	Type	Last modified	Size	Storage class
CN.py	py	November 19, 2023, 19:00:19 (UTC-05:00)	2.6 KB	Standard
image.png	png	November 18, 2023, 19:41:19 (UTC-05:00)	305.0 KB	Standard
index.txt	txt	November 18, 2023, 18:28:37 (UTC-05:00)	2.5 KB	Standard

I created azignment7bda S3 storage. As you can see you can see list of files as well in the storage.

3. AWS RDS creation



database-a7

Summary

DB identifier	CPU	Status	Class
database-a7	3.49%	Available	db.t3.micro
Role	Current activity	Engine	Region & AZ
Instance	0 Connections	MySQL Community	us-east-1d

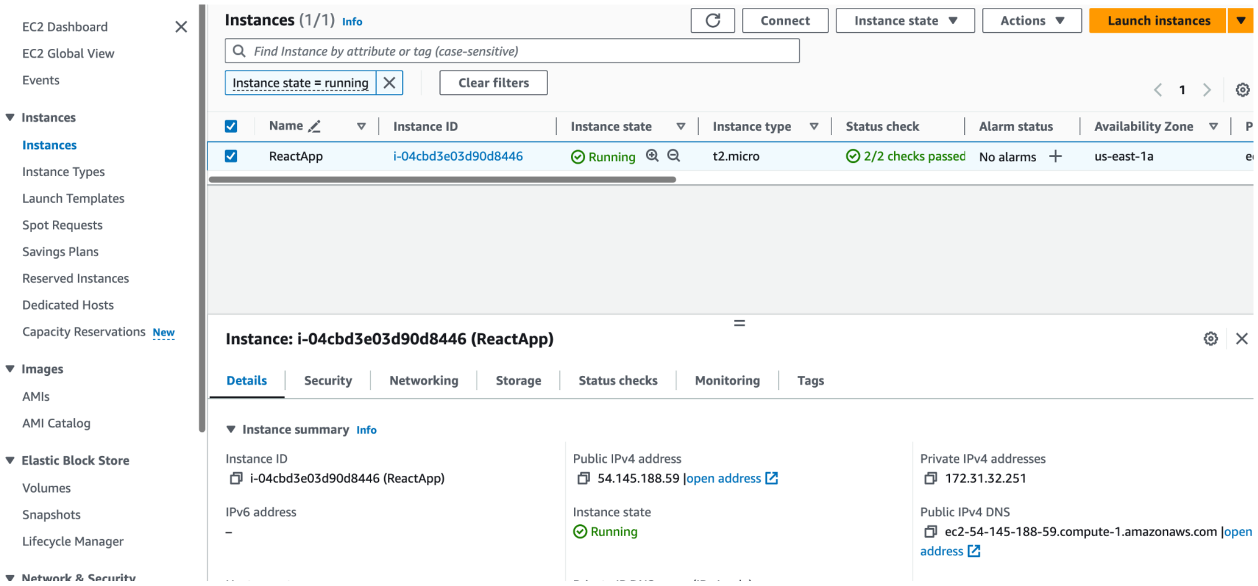
Connectivity & security

Endpoint & port
Endpoint
database-a7.cyvytfyu0u6o.us-east-1.rds.amazonaws.com
Port
3306

Networking
Availability Zone
us-east-1d
VPC
[redacted]
Subnet group

Security
VPC security groups
[redacted]
Active
Publicly accessible
Yes

4. AWS EC2 Instance creation and hosting web application



Instances (1/1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
ReactApp	i-04cbd3e03d90d8446	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a

Instance: i-04cbd3e03d90d8446 (ReactApp)

Instance summary Info

Instance ID
i-04cbd3e03d90d8446 (ReactApp)

Public IPv4 address
54.145.188.59 |open address

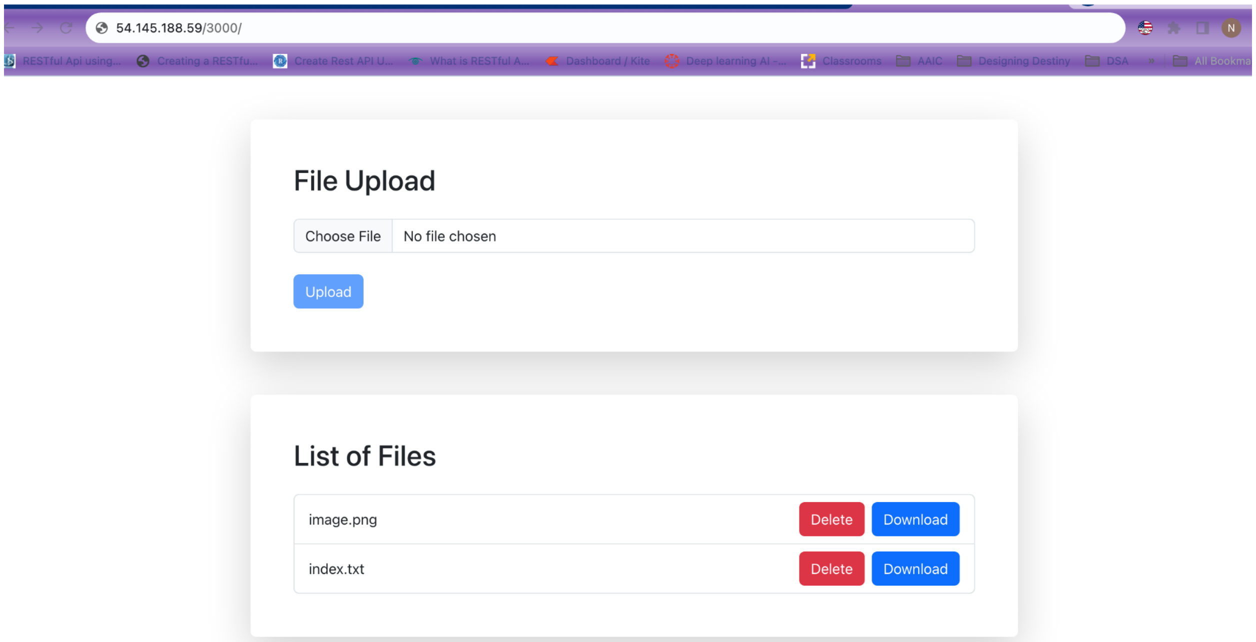
Private IPv4 addresses
172.31.32.251

IPv6 address
–

Instance state
Running

Public IPv4 DNS
ec2-54-145-188-59.compute-1.amazonaws.com |open address

Hosting and application is running as below


54.145.188.59/3000/

File Upload

Choose File No file chosen

Upload

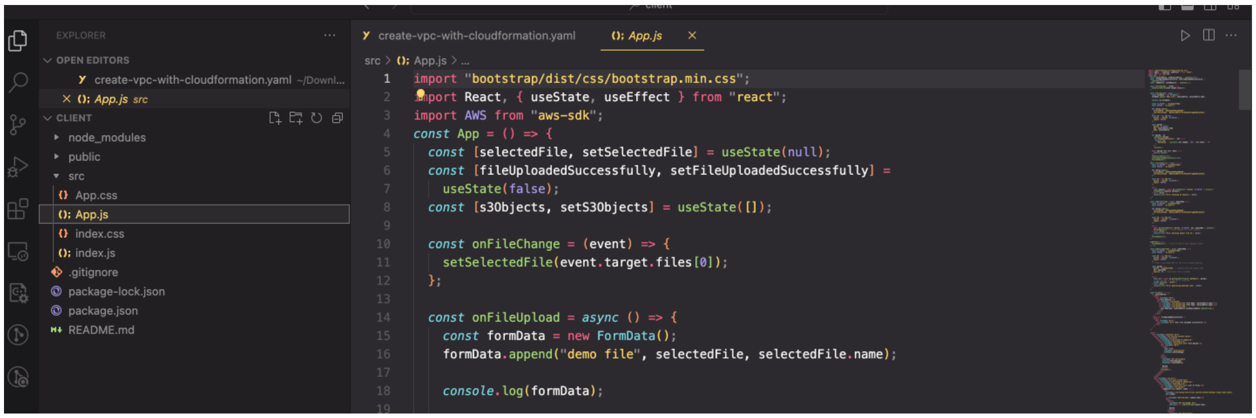
List of Files

image.png	Delete	Download
index.txt	Delete	Download

TESTING FOR HIGH TRAFFIC

I tested application with high traffic with multiple POST, GET requests creating artificial high traffic for multiple subnets in the EC2 instance. Application didn't hang and didn't crash and executed successfully.

Application Stack
USED - MySQL, NODE, React, Express for the application development.



```
import "bootstrap/dist/css/bootstrap.min.css";
import React, { useState, useEffect } from "react";
import AWS from "aws-sdk";
const App = () => {
  const [selectedFile, setSelectedFile] = useState(null);
  const [fileUploadedSuccessfully, setFileUploadedSuccessfully] =
    useState(false);
  const [s3Objects, setS3Objects] = useState([]);

  const onFileChange = (event) => {
    setSelectedFile(event.target.files[0]);
  };

  const onFileUpload = async () => {
    const formData = new FormData();
    formData.append("demo file", selectedFile, selectedFile.name);

    console.log(formData);
```

ADVANTAGES AND BENEFITS FROM APP POINT OF VIEW

1. AWS S3 for File Storage (File Kart):
   - Scalability: S3 provides highly scalable object storage. As your user base grows and demands more storage, S3 can seamlessly scale to accommodate.
   - Security: It offers fine-grained access controls and encryption features ensuring data security.
   - Reliability: S3 guarantees high availability and durability of stored files.

2. AWS RDS for Storing User Credentials:
   - Managed Service: RDS is a fully managed service, handling routine database tasks like backups, patching, scaling, etc.
   - Security and Compliance: It provides robust security features, including encryption at rest and in transit, enabling compliance with various regulations.
   - Scalability Options: RDS supports multiple database engines and allows for easy scaling as your user base and data grow.

3. AWS CloudFormation for VPC Creation:
   - Infrastructure as Code (IaC): CloudFormation enables defining infrastructure resources in a template, ensuring consistency and repeatability when deploying resources.
   - Automation: Simplifies VPC setup, automates the creation and management of networking resources, ensuring they meet specific requirements.

4. AWS EC2 for Application Deployment:
   - Customization: EC2 instances provide flexibility in choosing instance types, OS, and configurations best suited for your application's needs.
   - Scalability: You can easily scale EC2 instances up or down based on traffic demands, ensuring performance and cost optimization.
   - Integration: Seamlessly integrates with other AWS services and enables easy deployment of your application.

5. AWS IMS (Infrastructure Management Service):
   - Resource Optimization: IMS tools help monitor resource usage, identify bottlenecks, and optimize infrastructure configurations for better performance.
   - Cost Management: Offers insights into resource utilization and helps in optimizing costs by right-sizing resources.

Challenges

1. Challenges while hosting app in EC2. I needed to figure out backend fetching mechanism.
2. How to connect s3 bucket into my application. Figure out end point.
3. Understand significance of VPC from application point of view
4. Wanted to maintain free cost tier so that I wont’ end up causing excessive usage and bill generation.