DALHOUSIE
UNIVERSITY

CSCI5409 – Advanced Topic in Cloud Computing

Name: Darshil Patel
Banner ID: B00946528

Term Assignment

Table of Contents

1. Introduction

Introducing our innovative and user-friendly temporary File Sharing Application - a secure and efficient cloud-based solution designed to revolutionize the way users store, share, and access files. With this cutting-edge platform, users can effortlessly store their files for a duration of 24 hours, enabling seamless sharing with others using unique access codes.

This application offers a simple and intuitive interface, making it easy for users to generate and provide access codes to their trusted recipients. The access codes act as virtual keys, granting access to the shared files for a limited period, ensuring data security and confidentiality. One of the standout features of our File Sharing Application is the ability to track the remaining time on shared files. Users can conveniently verify how much time is left before the file access expires, allowing them to manage their shared files effectively.

Furthermore, recipients can conveniently access the shared files by simply using the access code and gain instant download links, streamlining the file-sharing process. By leveraging robust cloud services, This application guarantee a reliable and high-performance platform that ensures seamless access to files from any location and on any device.

Experience the convenience of our temporary File Sharing Application, where sharing files has never been more secure, and hassle-free. Whether it's for personal or professional use, our platform is designed to meet the needs of individuals and businesses seeking a secure and efficient file-sharing solution.

2. Services Used

For this project, I have used AWS cloud technologies to build an application to share Files among people temporary without having to think too much.

The following are the list of services I have used to create an application and used cloud formation to provide a necessary resource:

Compute

1. **Ec2**
   The Main Goal of my application is to store and deliver the file among the people which requires a ideal amount of time to share data between my application and clients Browser. My application uses EC2 as server to provide backend functionalities via API Gateway.

2. **Elastic Beanstalk**
   I have used Elastic Beanstalk to deploy my ReactJS frontend which will communicate with EC2 using API Gateway except for the file transfer API between client browser and a server call because it requires a faster communication and longer connection time which API Gateway is not capable to do.

3. **Lambda**
   Since I have created temporary File sharing application, there is some background process is required for that I have used Lambda to check the timestamp in the Database for all uploaded files, If the file available more than 24hrs then It will change the file status to expired, so user will not be able to have access to the resource after 24hrs after uploading the file.

Storage

1. **S3**
   I have used S3 as storage for file storing because my application is independent of authentication module. The file will be uploaded to s3 via EC2 server and upon doing that a unique code will be generated and sent to client which is sufficient to recognize the client.

2. **DynamoDB**
   I have used to store metadata and the unique code associated with file and used that unique code as accessCode of that file for a user. So once the user is done uploading the image the unique accessCode will be stored in a database.

Network

1. **API Gateway**
   As mentioned above API Gateway used as communication medium between frontend (Elastic Beanstalk) and the backend (EC2) to communicate using SSL/TLS (Transport Layer Security) protocols which makes data communication secure.

General

1. **SNS**
   SNS used to for creating a feedback/inquiry functionality from the client side. Basically, there is page called Contact us where user can put their email and Note/Feedback as sentence and submit it as soon as they submit it, I will receive the mail on my account.
   I would subscribe to SNS topic when application is being formed using cloud formation.

2. **Event Bridge**
   As I have mentioned before, Event Bridge is basically scheduler which will call the lambda function to perform validity of the files in DynamoDB. It will keep on triggering the lambda every minute.

Justification:

As compare to serverless architecture with traditional server EC2 connection timeout period is longer, Lambda has limitation for example application requires persistent state, extensive customization, complex components, fine-grained control over performance, or relies on traditional web server architectures. However, this choice comes with additional operational responsibilities compared to the serverless approach offered by Lambda.

DynamoDB is used because the HASH partition key is set for the faster data retrieval and also it is NOSQL Database allows inner level object.

S3 provides unlimited storage capabilities.

VPC requires multiple instance and complex routing and connection because of muli region support. As compare to REST API Gateway.

3. Deployment Model

Public Cloud:

I have used public cloud as deployment model because I myself don't have resources to run the application also public cloud allow various service which I have used extensively as well as the requirement and objective of the application also effects.

Scalability
In terms of scalability, my frontend application is deployed on Elastic Beanstalk which is highly scalable and supports load balancing to distribute the work load.

Cost-effectiveness
Using public cloud saves a lot of money, it enables us an option to pay as you go schema, where we are charged based on time, we have used the services and resources rather than charging the monthly or yearly. Also, the resources like S3, DynamoDB, SNS, Lambda, etc, which my application uses support this payment schema. It makes it a very cost-effective.

Easy Management:
Public cloud service elastic beanstalk can quickly update the UI if new UI changes are applied without having to concern about the deployment process. It provides us a Platform as a service for deploying application which makes management much easier.

Accessibility:
Whole Internet can have access to the application from anywhere, also we can make it region specific very quickly without thinking too much.

Integration & Support AWS Services:
My application component was quickly configured together without having to think too much for most of the part for example, I am storing the files in S3 and metadata of the file in DynamoDB, so lambda function which checks DB file status can have direct access to other AWS services without authentication.

4. Delivery Model

My cloud delivery model appears to be a mix of Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). Here is the break down the services and their corresponding cloud delivery models:

1. S3 (Object Storage): Using Amazon S3 to store files is an example of Infrastructure as a Service (IaaS). With IaaS, you are leveraging cloud infrastructure to store and manage data without worrying about the underlying hardware or infrastructure.
2. DynamoDB (NoSQL Database): Utilizing DynamoDB for metadata storage falls under the category of Database as a Service (DBaaS), which is a subcategory of Platform as a Service (PaaS). PaaS provides a platform for developers to build, deploy, and manage applications without worrying about the underlying infrastructure.
3. Lambda (Serverless Computing): Running Lambda functions to check DynamoDB for file status is an example of Serverless Computing, which is also a form of Platform as a Service (PaaS). Serverless computing allows you to execute code without provisioning or managing servers explicitly.
4. EC2 (Virtual Servers): Hosting the backend on EC2 instances is another example of Infrastructure as a Service (IaaS). With EC2, you have virtual servers in the cloud that you can configure and manage as needed.
5. Elastic Beanstalk (Application Deployment): Using Elastic Beanstalk for frontend deployment is an example of Platform as a Service (PaaS). Elastic Beanstalk abstracts the infrastructure and automatically handles deployment, scaling, and load balancing of applications.
6. API Gateway: Deploying API Gateway for communication between frontend and backend APIs is another example of Platform as a Service (PaaS). API Gateway provides a fully managed service for building, deploying, and managing APIs.

my cloud delivery model encompasses a combination of Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) offerings. IaaS components include S3 for object storage and EC2 for hosting backend services, while PaaS components include DynamoDB for database storage, Lambda for serverless computing, Elastic Beanstalk for frontend application deployment, and API Gateway for managing APIs. This mix of services allows you to focus on application development and functionality while leveraging the cloud provider's infrastructure and platform services to handle various operational tasks.

5. Architecture

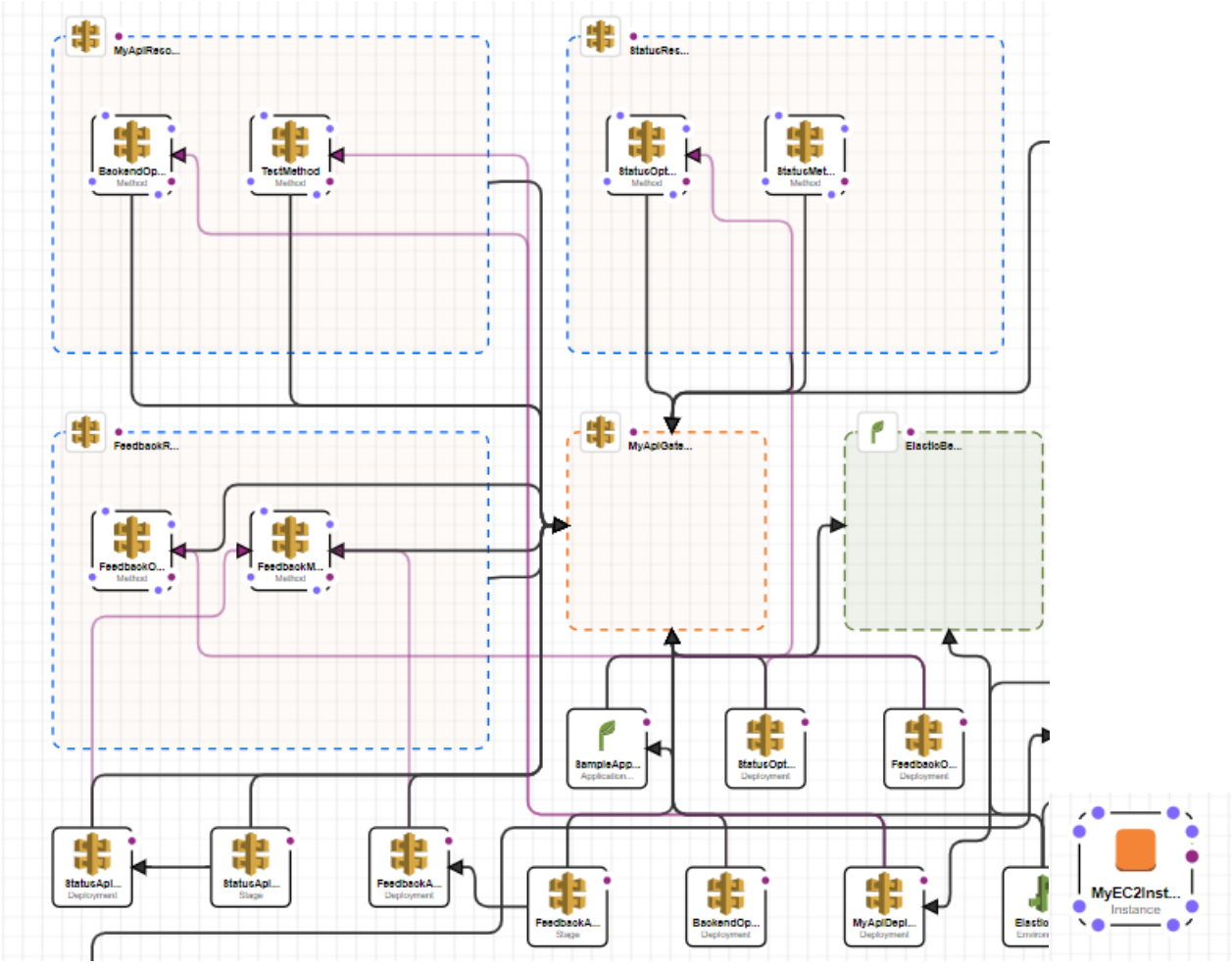

Figure 1: Secure API Communication with Elastic Beanstalk

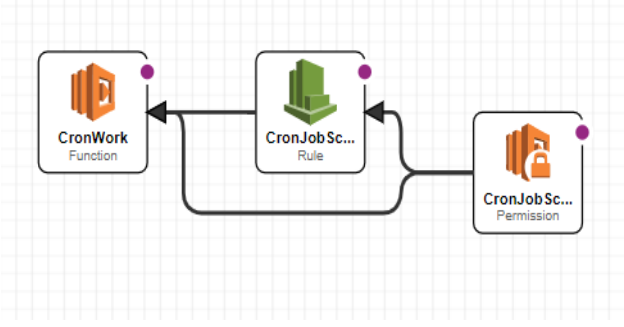

Figure 2: Internal Event Driven Microservice

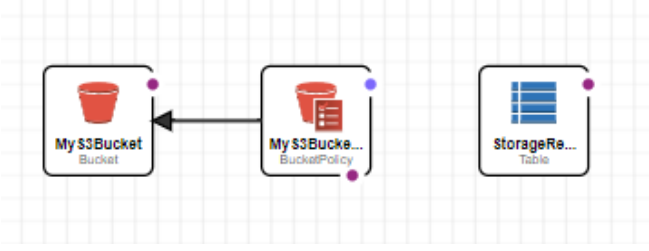

Figure 3: Native Cloud Storage

Communication & Connections:

Backend: NodeJS Express API Server
Deployed on EC2
Act as traditional server to check api calls form secure api gateway.
Frontend: ReactJS
Deployed on Elastic Beanstalk
Storage: S3
DynamoDB
Comms: API Gateway enables secure path to backend from from frontend

Internal Microservice: Lambda with Event Bridge enables Event Driven Architecture

6. Security

In term of security Frontend is deployed on the Elastic Beanstalk which is **Platform as a service** uses secure connection once it is deployed for example using TLS (Transport Layer Security) protocol.

In addition to that **API Gateway** is used for securing the path of backend API and allowed my application to transfer data with proper secure Transport Security Layer.

S3: Policies are defined to have proper and limited access to the resource.

Internal **Developed Microservice** to access the DB.

Venerability:
In my application there is one API call which is directly connected to EC2 instance form frontend for faster uploading the data to S3 which can be one of the venerable points in terms of security.

Future improvement suggestions:

7. Transferring the EC2 backend server to VPC
8. Use Secret Manager instead of storing the credentials as ENV variable.
9. Bucket policies improvement

7. Cost

Initial Cost:

- S3 Bucket Creation: $0 (There are no setup fees for generating S3 buckets).
- Event Bridge Configuration: $0 (There are no upfront fees for configuring Event Bridge rules).
- API Gateway Setup: $0 (There are no upfront fees for setting up API Gateway).
- Dynamo DB Cost: Depends upon the configuration and read write operations

Ongoing Cost:

- Lambda Function Execution Costs: Assuming 10,00 calls per day to check DB status cost is $0.00
- API Gateway: Assumes 500,000 API queries with little data transmission. Total monthly network cost: $5.00.
- S3 Bucket Storage: Assumes 1000GB of File. Total monthlystorage cost: $.23.0.
- Event-Bridge: Assuming a 1minute per event for backend process. Monthlyestimated cost: $5.00.
- Elastic Beanstalk: Assuming 0.5$ each hour monthly it would be like 14$ also Ec2
- EC2 : assuming less price than elastic Beanstalk , monthly approx. 10$
- Dynamo DB: Assuming with minimum usage of read write operation with DB monthly:3$

Monthly Cost: 61.00$

Investments & Improvements:

This application is light - weight and very simple with no authentication. It allows to quickly share file via accessCode which is a unique and In future new features like store file as folder structure and to present on the frontend to make it like drive, OneDrive and Mega sync for better competition.