Serverless Web Application on AWS

MIS 690: Cloud Computing: Fundamentals, Architecture, and Security

05/05/2024

Prof. Jacob Doiron

Fowler College of Business

San Diego State University

by

Roshini Padmanabha

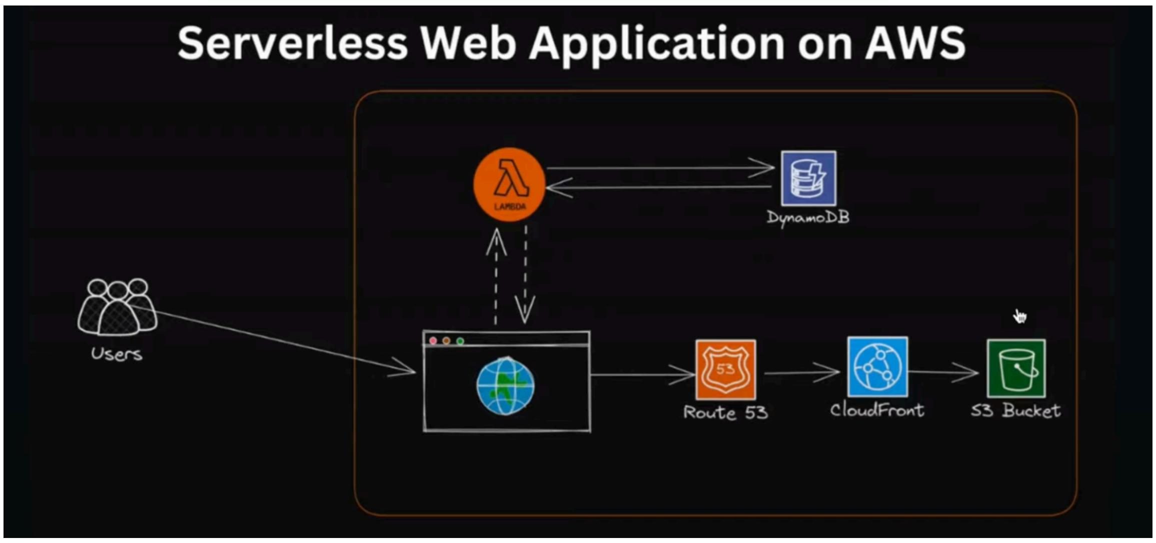
Mihir Kulkarni

Prathamesh Jalgaonkar

Group No. 3

Introduction

Serverless Web Application on AWS aims at deploying a scalable and secure web application leveraging cloud technologies, particularly AWS services. This report discusses the setup, execution, tasks performed, and their significance in relation to cloud and information security. It also addresses core principles such as accountability, monitoring, confidentiality, and more in the context of the system deployed.



System Creation & Deployment

Project Setup and Execution

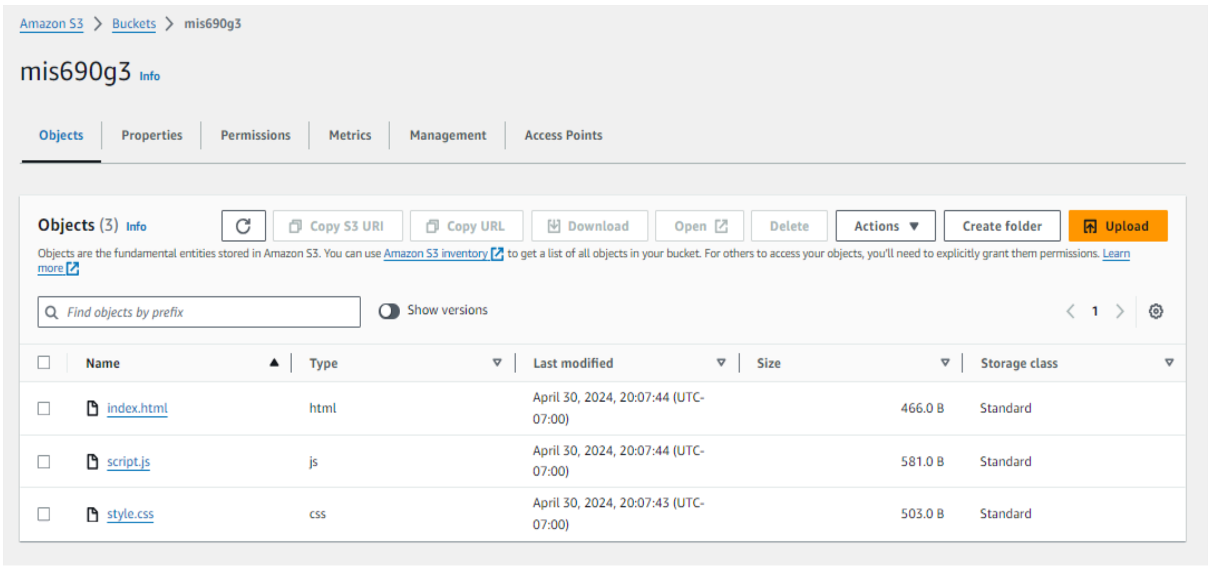
The project began with the setup of an S3 bucket to host the static files of the web application, including HTML, CSS, and JavaScript. This was followed by the configuration of a CloudFront distribution to ensure low latency and efficient content delivery globally. Route 53 was utilized for domain management and SSL certificate provisioning via AWS Certificate Manager (ACM) to enable HTTPS. DynamoDB was employed to implement a view counter, and a Lambda function was created to interact with the database and update the view count. Finally, the web application files were integrated with the Lambda function to display the view count dynamically.

1. **Setup of S3 Bucket**

Navigate to S3: In the AWS Management Console, search for "S3" in the services search bar, or navigate to the "Storage" section and click on "S3".

Create a Bucket:
- Click on the "Create bucket" button.

- Enter a unique bucket name. Bucket names must be globally unique across all AWS accounts.
- We chose the AWS Region where we want to create the bucket.


Amazon S3 > Buckets > mis690g3

mis690g3 Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (3) Info

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
index.html	html	April 30, 2024, 20:07:44 (UTC-07:00)	466.0 B	Standard
script.js	js	April 30, 2024, 20:07:44 (UTC-07:00)	581.0 B	Standard
style.css	css	April 30, 2024, 20:07:43 (UTC-07:00)	503.0 B	Standard

Configure Options & Set Permissions:

- We chose the desired options for versioning, server access logging, and encryption.
- The appropriate settings for bucket permissions were chosen. We configured access control settings, such as access for AWS accounts, and bucket policy.

Store Static Files in S3:

- Upload HTML, CSS, and JavaScript files to an S3 bucket.

Configure S3 for Website Hosting:

- Enable "Static website hosting" in S3 bucket properties.
- Specify index and error documents.

Set Up CloudFront Distribution:

- Create a CloudFront distribution with the S3 bucket as the origin.
- Configure caching and security settings.

Edit S3 Bucket Policy:

- Allow CloudFront to access S3 objects by editing the bucket policy.
- Granted CloudFront GetObject permissions.

Review:

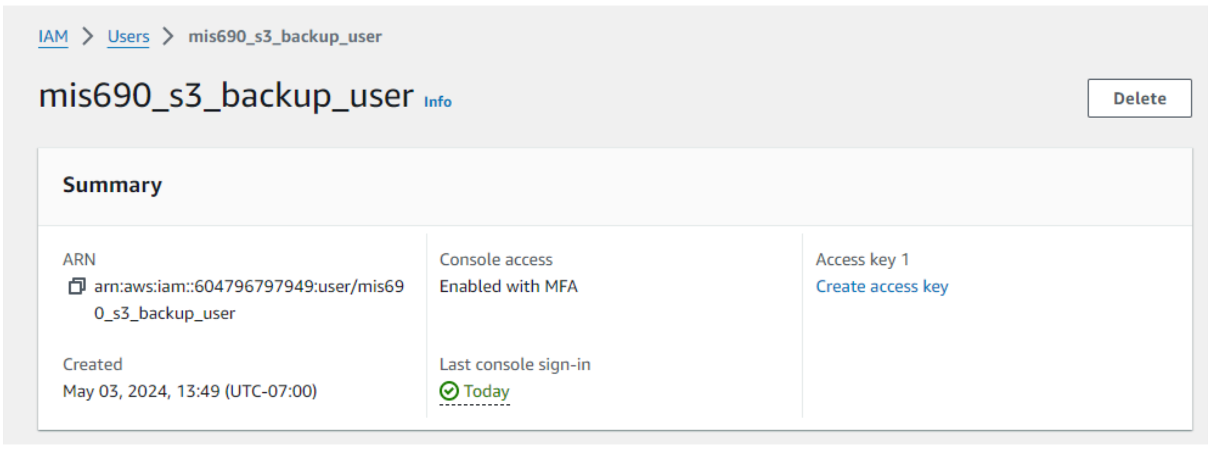
- Review the settings for the bucket.
- Click "Create bucket" to create the bucket.

To Set Up Backup for S3 Bucket:

- Navigate to AWS Backup
- Go to Create On-demand backup in the AWS Backup console.
- Choose the Resource type as S3, specify the Retention period days of 7 days and choose the IAM role having the required permissions to create this backup for S3. Assign the S3 bucket as the backup vault for storing backups.

Verify Configuration:

- Once the bucket is created, verify the configuration settings, and ensure that the bucket is accessible and configured according to the requirements.



mis690_s3_backup_user

Summary

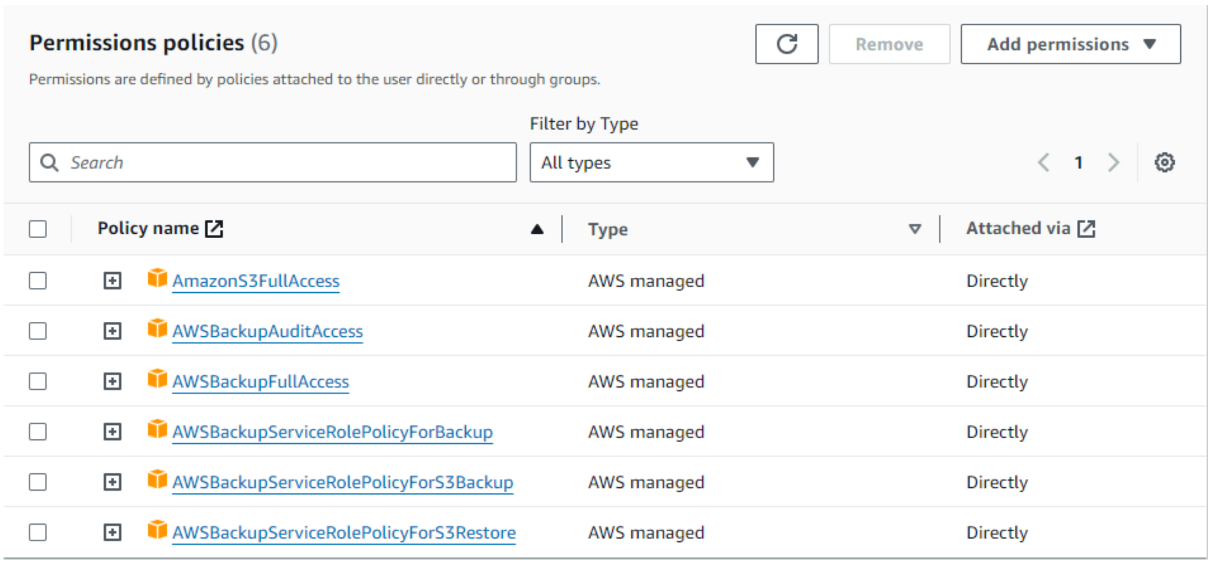
ARN
arn:aws:iam::604796797949:user/mis690_s3_backup_user

Created
May 03, 2024, 13:49 (UTC-07:00)

Console access
Enabled with MFA

Last console sign-in
Today

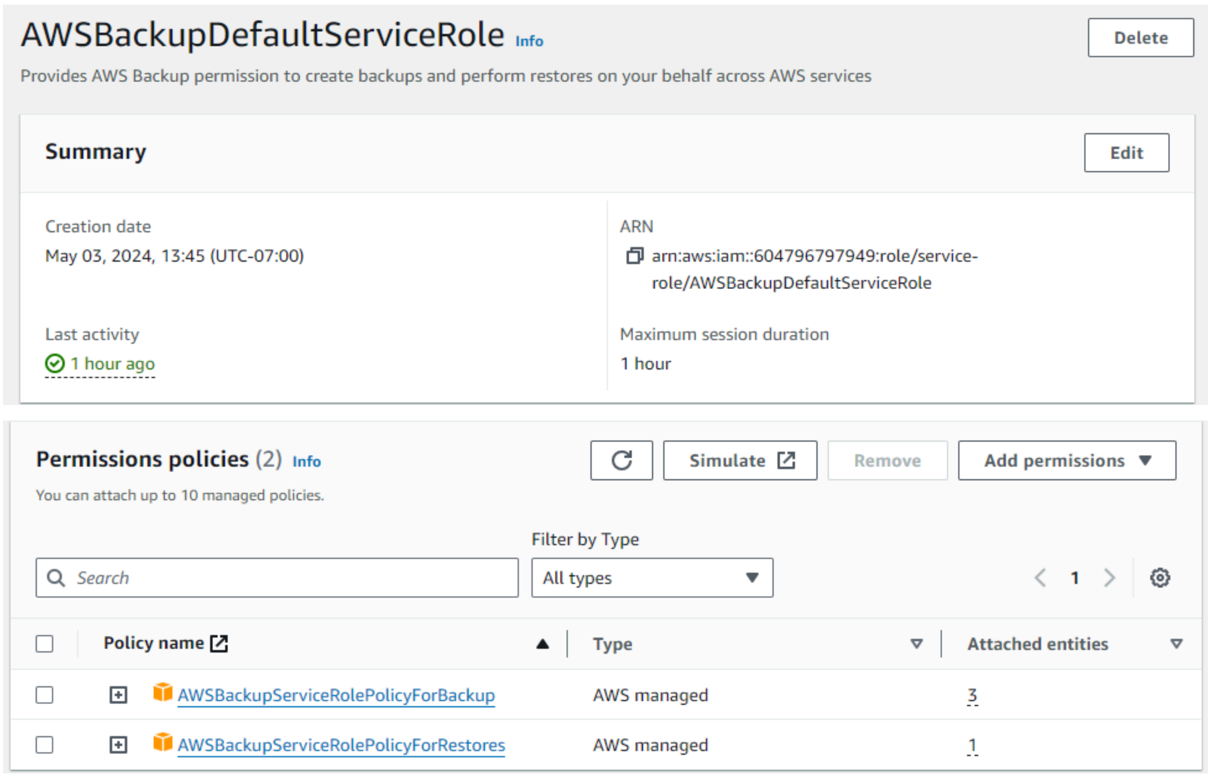
Access key 1
Create access key



Permissions policies (6)

Permissions are defined by policies attached to the user directly or through groups.

Policy name	Type	Attached via
AmazonS3FullAccess	AWS managed	Directly
AWSBackupAuditAccess	AWS managed	Directly
AWSBackupFullAccess	AWS managed	Directly
AWSBackupServiceRolePolicyForBackup	AWS managed	Directly
AWSBackupServiceRolePolicyForS3Backup	AWS managed	Directly
AWSBackupServiceRolePolicyForS3Restore	AWS managed	Directly



AWSBackupDefaultServiceRole Info

Provides AWS Backup permission to create backups and perform restores on your behalf across AWS services

Summary

Creation date
May 03, 2024, 13:45 (UTC-07:00)

ARN
arn:aws:iam::604796797949:role/service-role/AWSBackupDefaultServiceRole

Last activity
1 hour ago

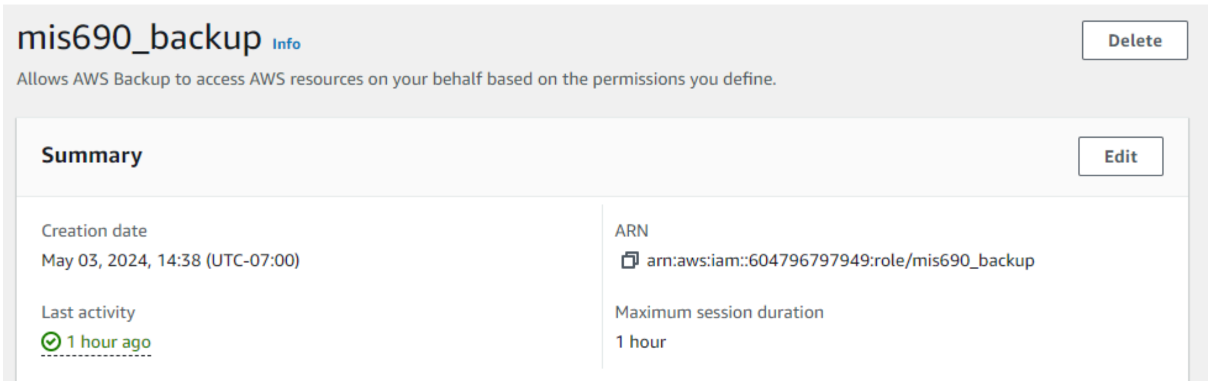
Maximum session duration
1 hour

Permissions policies (2) Info

You can attach up to 10 managed policies.

Filter by Type
Search | All types | 1

Policy name	Type	Attached entities
AWSBackupServiceRolePolicyForBackup	AWS managed	3
AWSBackupServiceRolePolicyForRestores	AWS managed	1



mis690_backup Info

Allows AWS Backup to access AWS resources on your behalf based on the permissions you define.

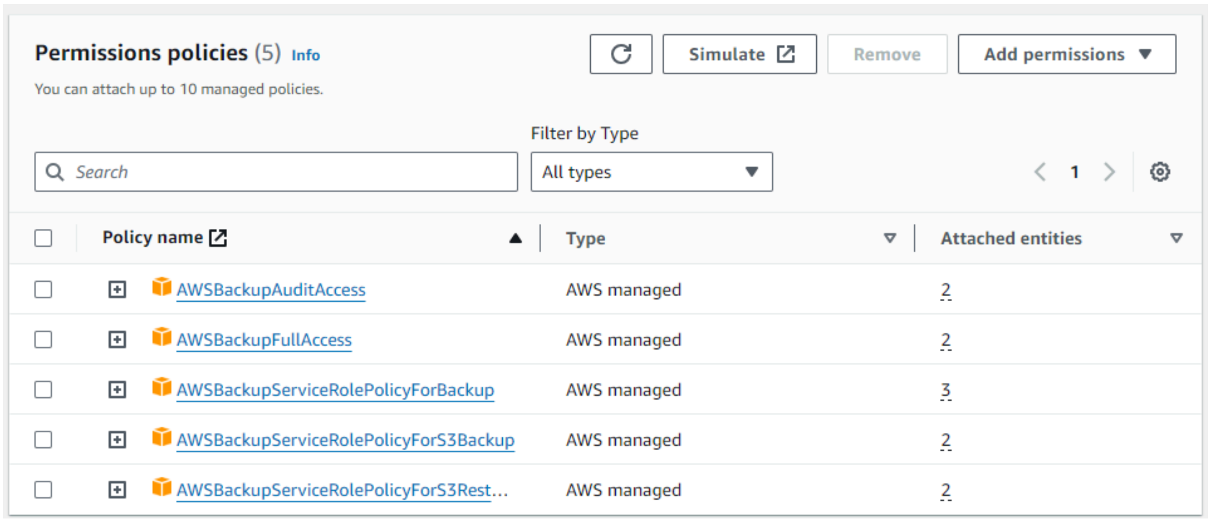
Summary

Creation date
May 03, 2024, 14:38 (UTC-07:00)

ARN
arn:aws:iam::604796797949:role/mis690_backup

Last activity
1 hour ago

Maximum session duration
1 hour



Permissions policies (5) Info

You can attach up to 10 managed policies.

Filter by Type
Search | All types | 1

Policy name	Type	Attached entities
AWSBackupAuditAccess	AWS managed	2
AWSBackupFullAccess	AWS managed	2
AWSBackupServiceRolePolicyForBackup	AWS managed	3
AWSBackupServiceRolePolicyForS3Backup	AWS managed	2
AWSBackupServiceRolePolicyForS3Rest...	AWS managed	2

2. Configuration of CloudFront Distribution

To create a CloudFront distribution for serving S3-hosted static files with low latency, follow these steps:

1. **Sign in to the AWS Management Console:**
- Navigate to the CloudFront console.

2. **Create Distribution:**
- Click on "Create Distribution" to start the process.

3. **Select Distribution Type:**
- Choose "Web" as the distribution type.

4. **Origin Settings:**
- Select the S3 bucket containing your static files as the "Origin Domain Name".
- Configure other settings like Origin Path, Viewer Protocol Policy, and Origin Access Identity as needed.

5. **Cache Behavior Settings:**
- Defined the caching behavior for different URL patterns.
- Configure cache TTL (Time-To-Live) settings for better performance.

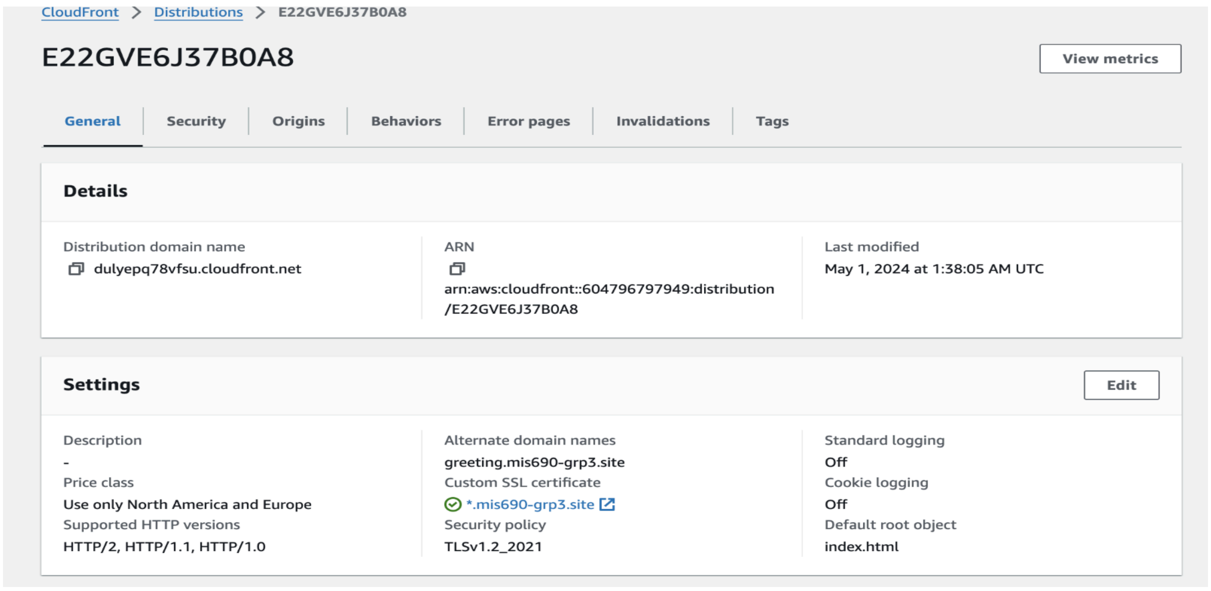
6. **Distribution Settings:**
- Specify the default cache behavior settings.
- Enable logging if required and specify an Amazon S3 bucket to store log files.

7. **Customize Settings:**
- Set up custom SSL certificate if needed.
- Configure additional options such as IPv6, HTTP/2 support, etc.

8. **Review and Create:**
- Review all settings to ensure they are configured correctly.
- Click on "Create Distribution" to create the CloudFront distribution.

9. **Wait for Deployment:**
- It may take some time for the distribution to deploy globally.

10. **Update DNS:**
- Once the distribution is deployed, updated the DNS settings to point to the CloudFront domain name.


CloudFront > Distributions > E22GVE6J37B0A8

E22GVE6J37B0A8

Details

Distribution domain name
dulyepq78vfsu.cloudfront.net

ARN
arn:aws:cloudfront::604796797949:distribution/E22GVE6J37B0A8

Last modified
May 1, 2024 at 1:38:05 AM UTC

Settings

Description
-

Price class
Use only North America and Europe

Supported HTTP versions
HTTP/2, HTTP/1.1, HTTP/1.0

Alternate domain names
greeting.mis690-grp3.site

Custom SSL certificate
*.mis690-grp3.site

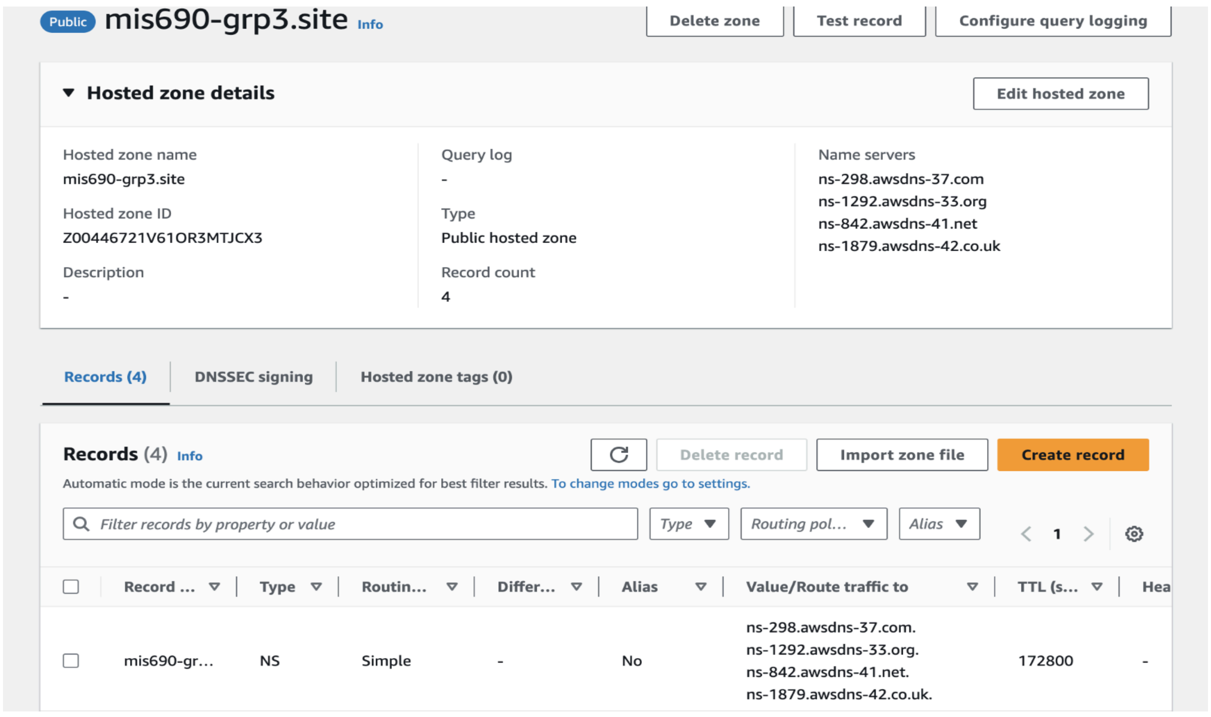
Security policy
TLSv1.2_2021

Standard logging
Off

Cookie logging
Off

Default root object
index.html

3. Route 53 Hosted Zone


Public mis690-grp3.site

Hosted zone details

Hosted zone name
mis690-grp3.site

Hosted zone ID
Z00446721V61OR3MTJCX3

Description
-

Query log
-

Type
Public hosted zone

Record count
4

Name servers
ns-298.awsdns-37.com
ns-1292.awsdns-33.org
ns-842.awsdns-41.net
ns-1879.awsdns-42.co.uk

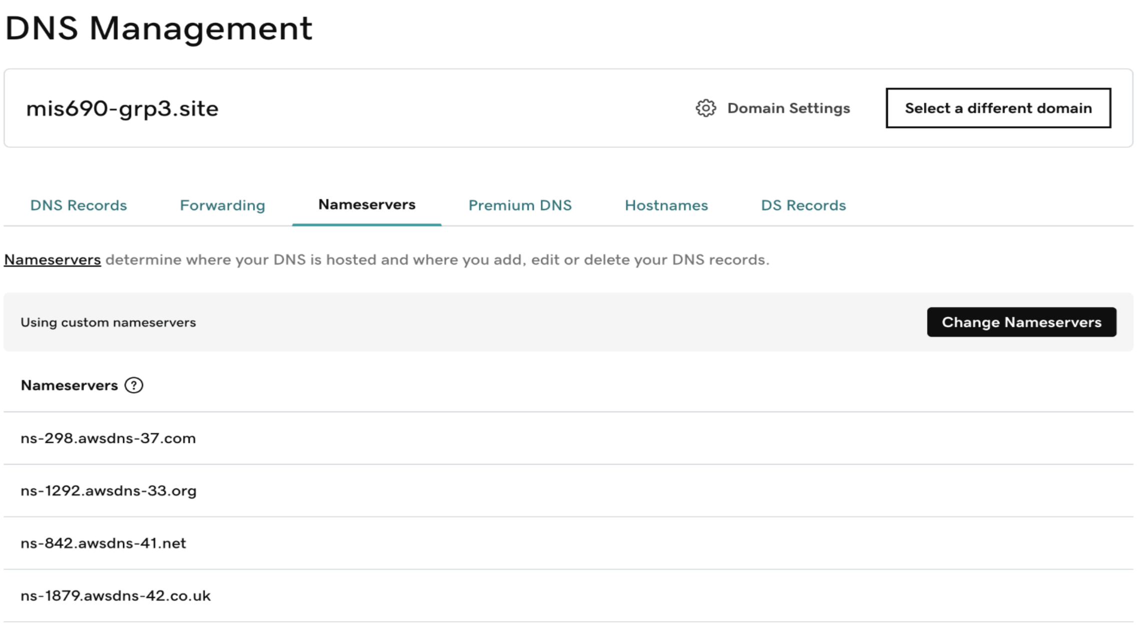
Records (4)

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Record …	Type	Routin…	Differ…	Alias	Value/Route traffic to	TTL (s…	Hea
mis690-gr…	NS	Simple	-	No	ns-298.awsdns-37.com. ns-1292.awsdns-33.org. ns-842.awsdns-41.net. ns-1879.awsdns-42.co.uk.	172800	-

Route 53 DNS management and custom nameservers

- A Route 53 hosted zone was set up to manage DNS for our domain name.
- Using Route 53 DNS management, we configured custom nameservers to handle DNS resolution for our domain.
- This setup ensures reliable and efficient routing of traffic to our web services, enhancing the overall performance and availability of our online presence.

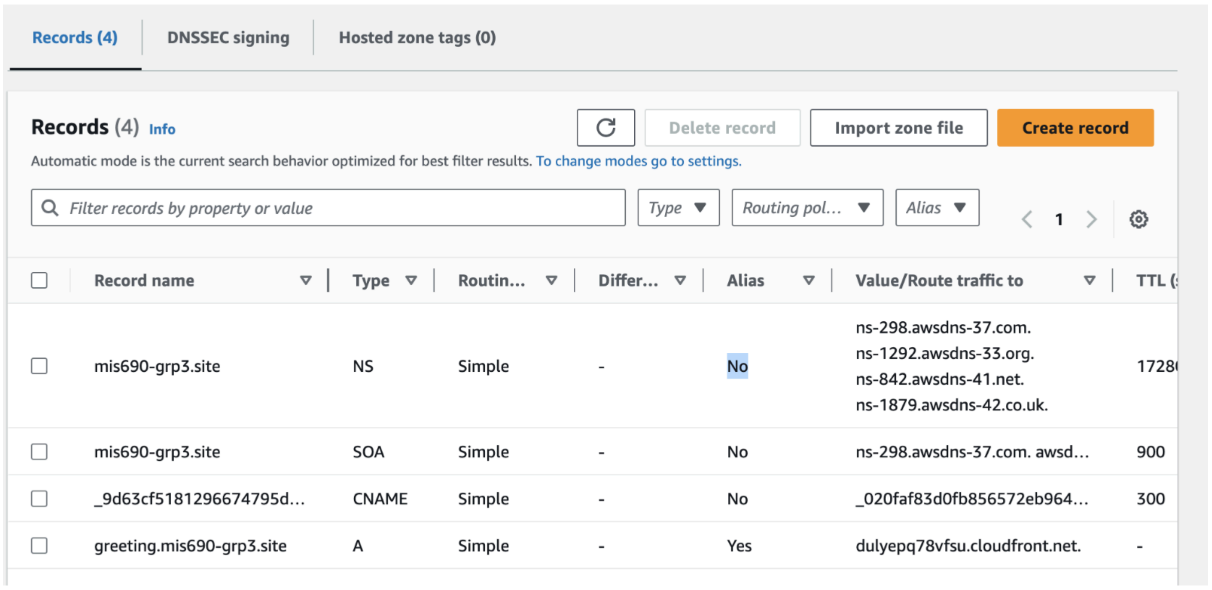

DNS Management

mis690-grp3.site

Nameservers determine where your DNS is hosted and where you add, edit or delete your DNS records.

Using custom nameservers

Nameservers
ns-298.awsdns-37.com
ns-1292.awsdns-33.org
ns-842.awsdns-41.net
ns-1879.awsdns-42.co.uk



Records (4) Info

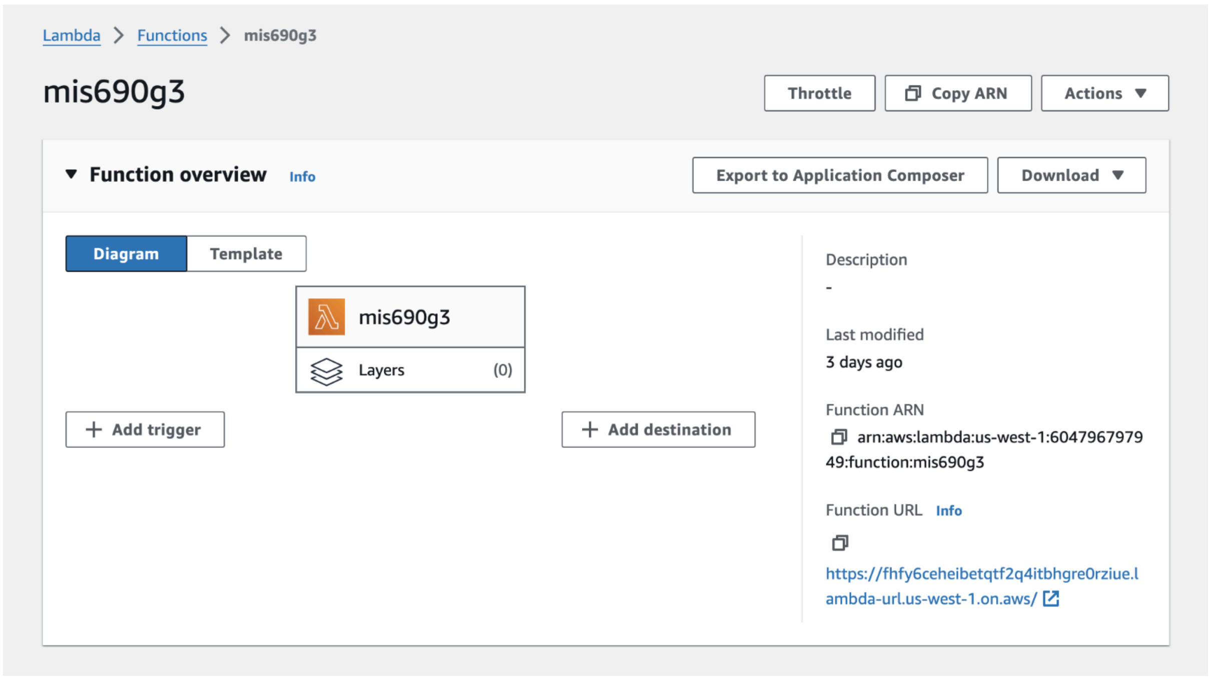
Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Record name	Type	Routin...	Differ...	Alias	Value/Route traffic to	TTL (
mis690-grp3.site	NS	Simple	-	No	ns-298.awsdns-37.com. ns-1292.awsdns-33.org. ns-842.awsdns-41.net. ns-1879.awsdns-42.co.uk.	1728
mis690-grp3.site	SOA	Simple	-	No	ns-298.awsdns-37.com. awsd...	900
_9d63cf5181296674795d...	CNAME	Simple	-	No	_020faf83d0fb856572eb964...	300
greeting.mis690-grp3.site	A	Simple	-	Yes	dulyepq78vfsu.cloudfront.net.	-

5. Creating AWS Lambda Function

- **Create a Lambda Function**: We Developed a Lambda function that handles the trigger event for the web application's static files.
- **Configure Cross-Origin Resource Sharing (CORS)**: Set up CORS on your S3 bucket or web server to allow requests from our website's domain names i.e. mis690-grp3.site
- **Enable CORS for Lambda Invocation**: In the CORS configuration, we specified the allowed origin URLs that are permitted to trigger the Lambda function.

- **Test Lambda Invocation**: Verify that the Lambda function can be successfully invoked by making requests from the allowed origin URLs specified in the CORS configuration.



Description
-

Last modified
3 days ago

Function ARN
arn:aws:lambda:us-west-1:604796797949:function:mis690g3

Function URL Info
https://fhfy6ceheibetqtf2q4itbhgre0rziue.lambda-url.us-west-1.on.aws/

- Function code executed in Lambda.

- We performed the following steps:

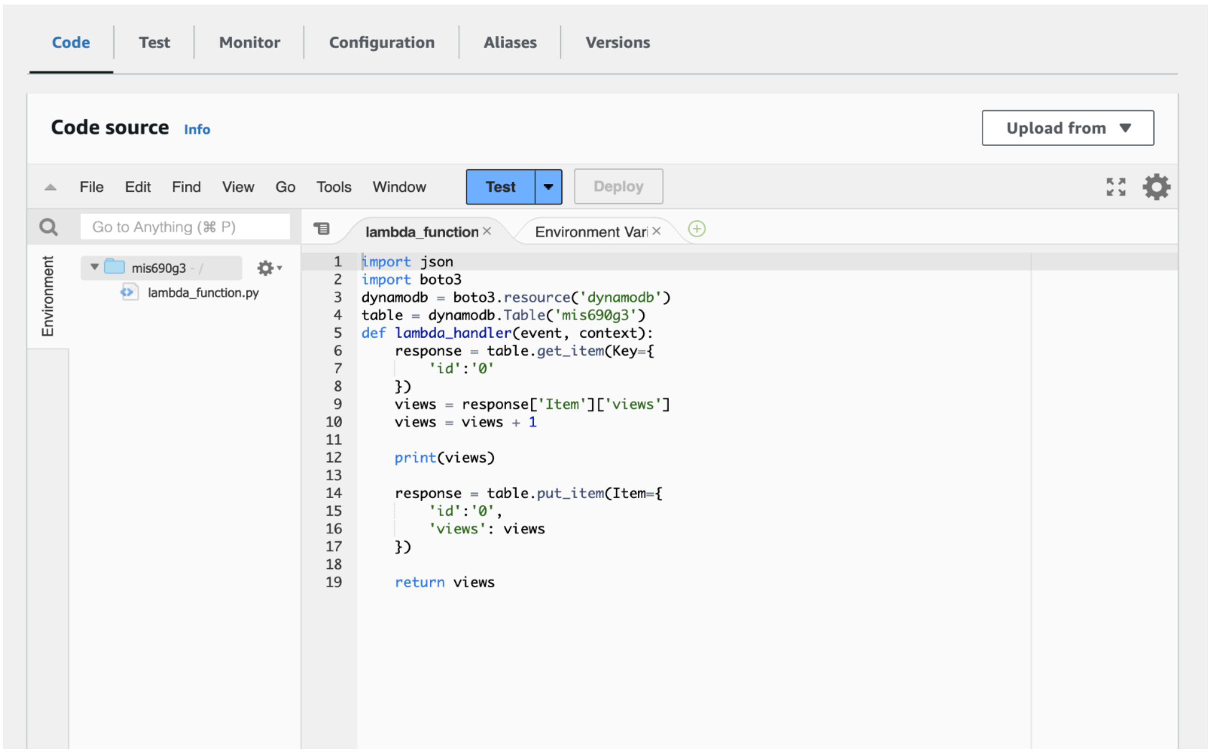
Lambda Function Setup: We wrote and configured a Lambda function to handle the backend logic of the Greeting App. This function was triggered upon user interaction or page load.

Processing User Input: We implemented code within the Lambda function to receive and process user input, such as submitting their name. We ensured that the function could extract and handle the submitted data effectively.

Dynamic Content Generation: We used the processed user input to dynamically generate content for the webpage, such as displaying a personalized greeting message like "Hello [Name]!". This content generation occurred within the Lambda function.

Interaction with DynamoDB: We incorporated DynamoDB into the Lambda function to store and retrieve data related to page views. We updated the database each time the Lambda function was invoked, incrementing the count of page views.

Integration with Webpage: We integrated the Lambda function with the frontend of the website to display the generated content and page view count. This integration typically involved REST requests or API calls from the front end to the Lambda function.



```python
import json
import boto3
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('mis690g3')
def lambda_handler(event, context):
    response = table.get_item(Key={
        'id':'0'
    })
    views = response['Item']['views']
    views = views + 1

    print(views)

    response = table.put_item(Item={
        'id':'0',
        'views': views
    })

    return views
```

6. Employment of DynamoDB

We performed the following steps:

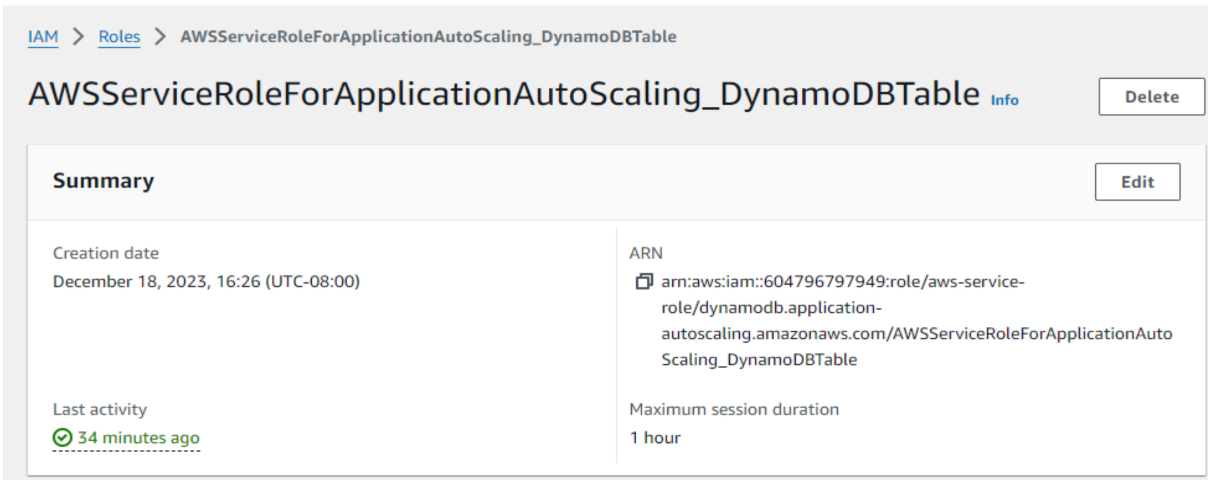
DynamoDB Table Creation: We created a DynamoDB table to store data related to page views. The table schema included an attribute for the page ID and another for the corresponding view count.

Initial Data Insertion: We initialized the view count for the page ID 0 to 0 upon table creation. This step ensured that the initial state of the page views was properly set in the DynamoDB table.

Lambda Triggering: We configured a Lambda function to trigger whenever a user accessed or refreshed the webpage. This Lambda function was responsible for incrementing the view count stored in the DynamoDB table whenever it was triggered.

View Count Update: Upon each Lambda triggering event, the associated function incremented the view count for page ID 0 in the DynamoDB table. This ensured that the view count accurately reflected the number of times users accessed or refreshed the webpage.

Dynamic Display Integration: We dynamically integrated the updated view count into the frontend of the website. This integration ensured that users could see the real-time count of page views whenever they visited the webpage.


IAM > Roles > AWSServiceRoleForApplicationAutoScaling_DynamoDBTable

AWSServiceRoleForApplicationAutoScaling_DynamoDBTable Info

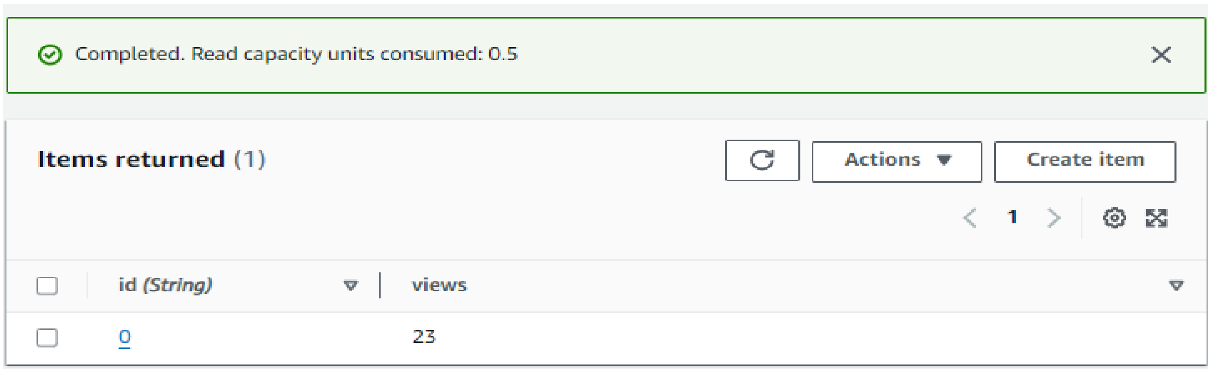
Summary

Creation date
December 18, 2023, 16:26 (UTC-08:00)

ARN
arn:aws:iam::604796797949:role/aws-service-role/dynamodb.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_DynamoDBTable

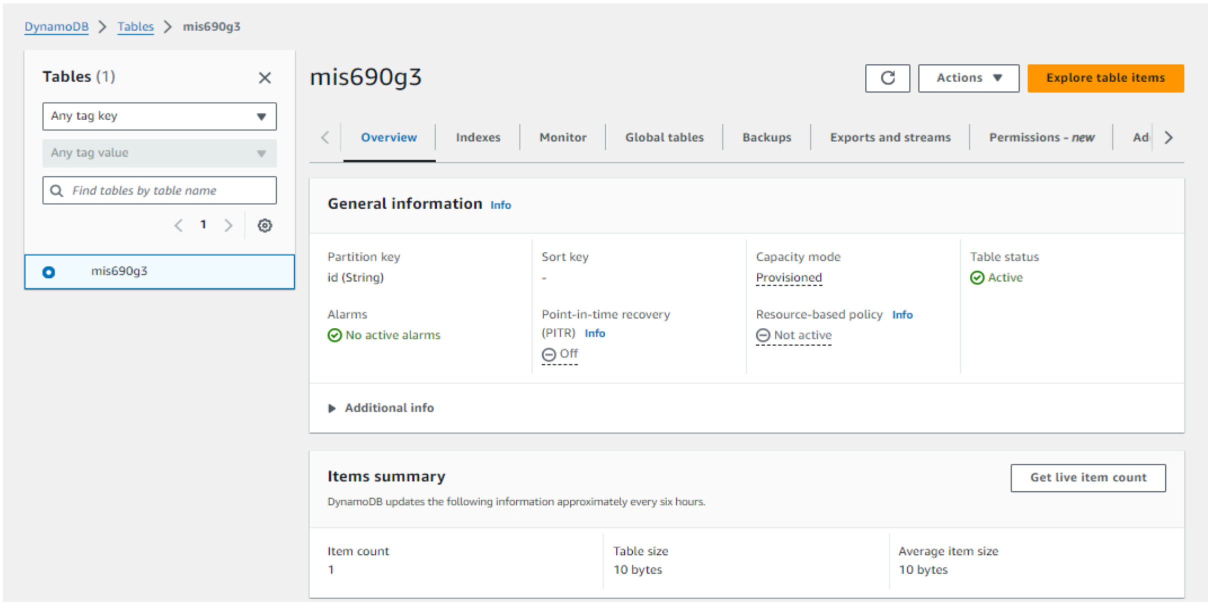
Last activity
34 minutes ago

Maximum session duration
1 hour


Completed. Read capacity units consumed: 0.5

Items returned (1)

id (String)	views
0	23


DynamoDB > Tables > mis690g3

Tables (1)

mis690g3

mis690g3

Overview | Indexes | Monitor | Global tables | Backups | Exports and streams | Permissions - new

General information Info

Partition key
id (String)

Sort key
-

Capacity mode
Provisioned

Table status
Active

Alarms
No active alarms

Point-in-time recovery (PITR) Info
Off

Resource-based policy Info
Not active

Additional info

Items summary
DynamoDB updates the following information approximately every six hours.

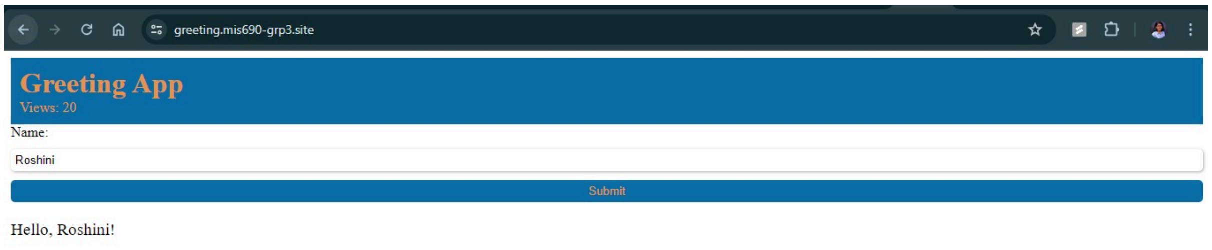
Item count
1

Table size
10 bytes

Average item size
10 bytes

The output from our AWS Lambda (Serverless) computing deployment is successfully rendering a web page, which is accessible through the URL: https://greeting.mis690-grp3.site/ .

This URL directs users to the hosted web page showcasing the operational capabilities and responsiveness of our serverless architecture, providing a real-time demonstration of our project's deployment and functionality.


greeting.mis690-grp3.site

Greeting App

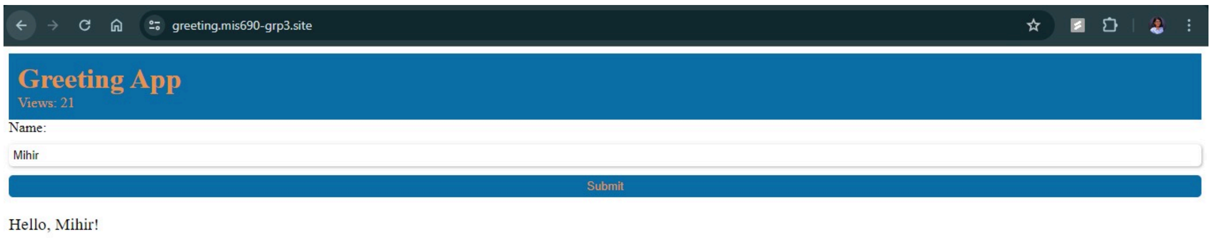
Views: 20

Name:

Roshini

Submit

Hello, Roshini!


greeting.mis690-grp3.site

Greeting App

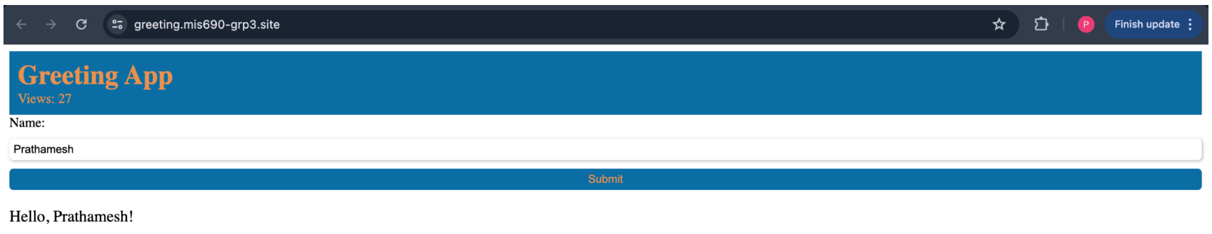
Views: 21

Name:

Mihir

Submit

Hello, Mihir!


greeting.mis690-grp3.site

Greeting App

Views: 27

Name:

Prathamesh

Submit

Hello, Prathamesh!

7. EC2 Website Page Count

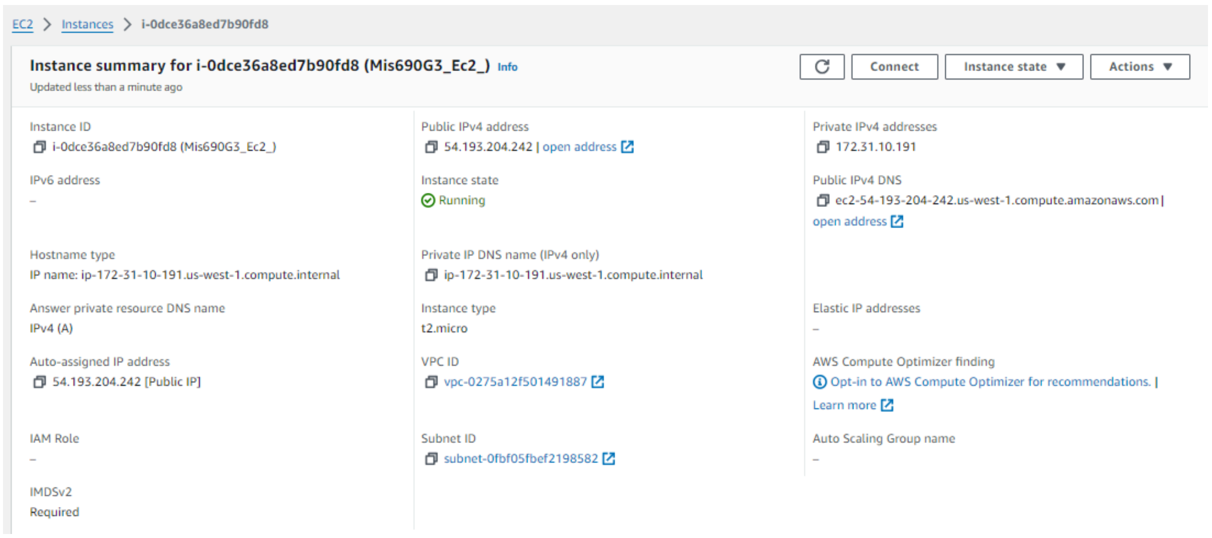
We've developed a Python application deployed on an Amazon Web Services (AWS) EC2 instance that accurately tracks and displays page views on our website. This capability is central to our understanding of user engagement and website traffic patterns.

To set up the page view tracking, we wrote a Python script that increments a counter every time the website is visited. This script runs in the background on the EC2 instance, ensuring seamless operation without manual intervention. We used a lightweight Flask framework for the web server to host our site and handle requests efficiently.
The website itself, accessible from the EC2 instance, features a simple yet effective user interface. It prominently displays the number of page visits, updating dynamically as new visits occur. This real-time updating is crucial for monitoring active user engagement.

For managing and deploying the code, we used the User data section in EC2 along with some scripting commands and the program code. Through this terminal, we executed commands to start the server, check its status, and troubleshoot any issues related to the deployment or the running of our Python application.

Code:

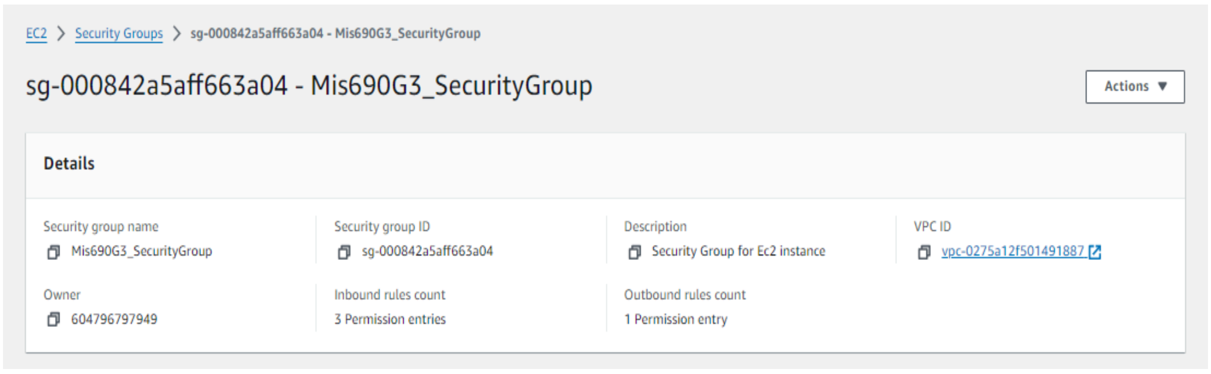
```
#!/bin/bash
# Update the yum package manager and install Apache (httpd)
yum update -y
yum install httpd -y

# Create an HTML file and use proper escaping for quotes
echo '<html><body><script>
async function updateCounter() {
    let response = await
fetch("https://fhfy6ceheibetqtf2q4itbhgre0rziue.lambda-url.us-west-1.on.aws/");
    let data = await response.json();
    document.body.innerHTML = "<h1>" + data + "</h1>";
}
updateCounter();
</script></body></html>' > /var/www/html/index.html

# Start and enable Apache to run on boot
systemctl start httpd
systemctl enable httpd
```



Instance summary for i-0dce36a8ed7b90fd8 (Mis690G3_Ec2_)

Updated less than a minute ago

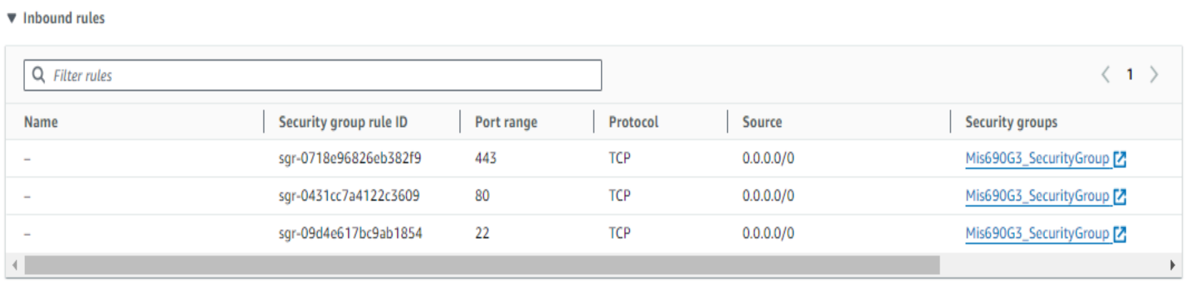
Instance ID i-0dce36a8ed7b90fd8 (Mis690G3_Ec2_)	Public IPv4 address 54.193.204.242 | open address	Private IPv4 addresses 172.31.10.191
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-54-193-204-242.us-west-1.compute.amazonaws.com | open address
Hostname type IP name: ip-172-31-10-191.us-west-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-10-191.us-west-1.compute.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 54.193.204.242 [Public IP]	VPC ID vpc-0275a12f501491887	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-0fbf05fbef2198582	Auto Scaling Group name –
IMDSv2 Required		


EC2 > Security Groups > sg-000842a5aff663a04 - Mis690G3_SecurityGroup

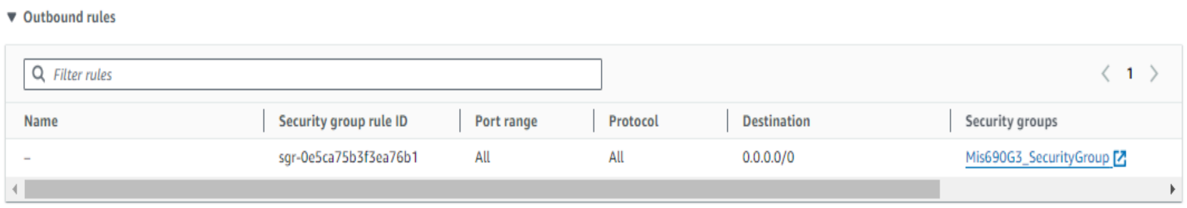
sg-000842a5aff663a04 - Mis690G3_SecurityGroup

Details

Security group name	Security group ID	Description	VPC ID
Mis690G3_SecurityGroup	sg-000842a5aff663a04	Security Group for Ec2 instance	vpc-0275a12f501491887
Owner	**Inbound rules count**	**Outbound rules count**	
604796797949	3 Permission entries	1 Permission entry	

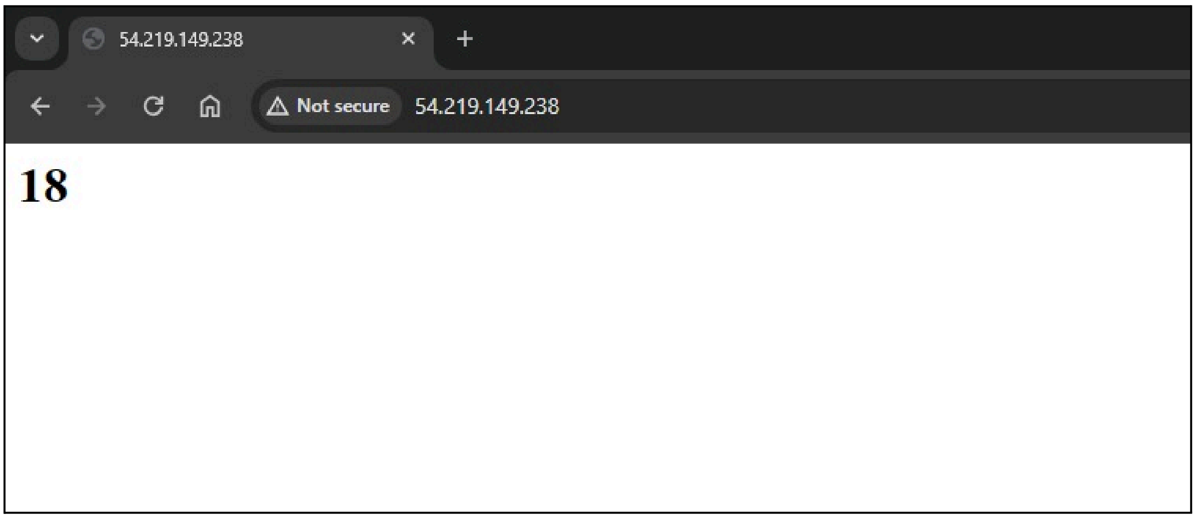

Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups
–	sgr-0718e96826eb382f9	443	TCP	0.0.0.0/0	Mis690G3_SecurityGroup
–	sgr-0431cc7a4122c3609	80	TCP	0.0.0.0/0	Mis690G3_SecurityGroup
–	sgr-09d4e617bc9ab1854	22	TCP	0.0.0.0/0	Mis690G3_SecurityGroup


Outbound rules

Name	Security group rule ID	Port range	Protocol	Destination	Security groups
–	sgr-0e5ca75b3f3ea76b1	All	All	0.0.0.0/0	Mis690G3_SecurityGroup

Ec2 Website Output Displaying Page Count:


54.219.149.238

Not secure 54.219.149.238

18

Significance in Relation to Cloud and Information Security

The deployment of a serverless architecture on AWS offers several advantages in terms of security and scalability. Firstly, Identity and Access Management (IAM) played a crucial role in granting appropriate permissions to users and services. By creating a dedicated IAM role for the Lambda function with restricted access to DynamoDB, the principle of least privilege was adhered to, ensuring that only necessary permissions were granted.

Furthermore, the system incorporated user restrictions by configuring origin access control settings in CloudFront to restrict direct access to the S3 bucket and allow access only through CloudFront. This helps mitigate the risk of unauthorized access to sensitive data stored in the bucket. Additionally, encryption mechanisms provided by AWS, such as SSL/TLS for data in transit and server-side encryption for data at rest, were leveraged to enhance data confidentiality and integrity.

Monitoring and logging were integral parts of the system, with CloudWatch being utilized for monitoring Lambda function executions and CloudFront access logs for tracking web traffic. This facilitates proactive detection of anomalies or suspicious activities, enabling timely responses to security incidents.

Rationale for System Choice and Cloud Deployment

The decision to implement the serverless web application on AWS was driven by several factors. Firstly, AWS offers a comprehensive suite of services specifically designed for building and deploying serverless applications, which greatly simplifies the development and management process. The scalability and pay-as-you-go pricing model of AWS also align well with the requirements of the project, allowing for cost-effective scaling based on demand.
Moreover, deploying the system on the cloud provides inherent benefits in terms of scalability, reliability, and global reach. Cloud services such as S3, CloudFront, and DynamoDB automatically handle scalability and high availability, eliminating the need for manual intervention and infrastructure management. This ensures consistent performance and reliability for users accessing the web application from various locations worldwide.

However, it's important to acknowledge the potential challenges and considerations associated with cloud deployment. While AWS provides robust security features and compliance certifications, organizations must still implement proper security measures and best practices to protect their data and applications. Additionally, reliance on third-party cloud providers introduces dependencies and potential vendor lock-in, which should be carefully evaluated.

System Security Enhancements

Implementing additional security measures is crucial to fortify the system against potential threats and ensure the integrity, confidentiality, and availability of data and services. In this section, we will discuss several security enhancements tailored to the unique requirements of the serverless web application deployed on AWS.

1. Implementing Multi-Factor Authentication (MFA):

To enhance security, particularly for accessing the AWS Management Console and performing administrative actions, we implemented MFA. We used AWS IAM (Identity and Access Management) to enforce MFA for all user accounts, requiring not only a password but also a second factor of authentication. This second factor was typically a time-based one-time password (OTP) from a device like a smartphone app.

2. Hardening:

We focused on hardening our AWS EC2 instances and services like S3 following best practices.

- Benchmarks and Vendor Guidance: We adhered to the Center for Internet Security (CIS) Benchmarks for AWS, which provide comprehensive guidelines for securing AWS resources. We also followed the AWS Well-Architected Framework, ensuring that our configurations met the security pillars of this framework.
- We referred to additional recommendations such as those provided by OpenVPN and other community resources to ensure our system's security settings were robust and resistant to common vulnerabilities.

3. Role Restrictions:

We used AWS IAM roles to define and restrict what actions each role could perform and which resources they could access. By adhering to the principle of least privilege, we ensured that each component of our system and each user only had access to the resources necessary for their function. For example, developers were not granted access to production data, and administrative functions were locked down to a small group of authorized users.

4. Policies/Restrictions:

We established strict IAM policies that explicitly defined what actions are allowed or denied, and under what conditions. These policies were applied to both user accounts and roles. For instance, we implemented policies that restrict the deletion of logs and backup files, and we ensured that access to sensitive resources was logged and monitored continuously.

We also set up network security measures such as Security Groups and Network Access Control Lists (NACLs) to control inbound and outbound traffic to our EC2 instances and other resources. These were configured to allow only the necessary traffic, reducing the potential attack surface.

5. Assessment and Future Actions:

Reflecting on the current state of our system and considering possible future improvements is a crucial exercise for maintaining and enhancing the security and efficiency of our project. Here are some areas where we could further improve and expand our project, demonstrating a proactive approach to system management and evolution:

1. Advanced Monitoring and Alerting

Implement more sophisticated monitoring tools such as AWS CloudWatch or third-party solutions like Datadog or Splunk for deeper insights into system performance and security. Setting up automated alerts for anomalous activities could help in preempting security incidents or operational bottlenecks.

2. Automated Scaling and Load Balancing

Future Actions: Though our current setup handles the projected load, as the number of users grows, it would be prudent to implement Auto Scaling groups and Elastic Load Balancing. This would not only help in managing load efficiently but also ensure high availability and fault tolerance.

3. Enhanced Data Encryption and Security

While we currently use encryption for data at rest and in transit, exploring more robust encryption options and technologies, such as hardware security modules (HSMs) or using AWS KMS for key management, could provide stronger security. Implementing more granular encryption at the application level could also be considered.

4. Comprehensive Disaster Recovery Planning

Develop a more detailed disaster recovery plan that includes multi-region deployment to handle major AWS region outages. Testing the recovery process regularly through drills and updating the strategy based on these tests would also be beneficial.

5. Integration of Artificial Intelligence for Security

Incorporate AI-driven security tools that can predict and identify potential threats based on patterns and unusual activities. Such tools can enhance the detection of zero-day vulnerabilities and automate certain security responses.

Support Plan for the System

Our primary objective for the next three months is to ensure that the system remains operational, secure, and resilient. The plan will focus on regular maintenance, security checks, and operational improvements, assigning responsibility to relevant roles within an IT/business context.

1. Monitoring and Maintenance:

- **Daily Monitoring:**

Role: IT Operations Team

The IT Operations Team will be responsible for daily monitoring of system health via AWS CloudWatch dashboards. They will check key metrics like CPU utilization, memory usage, and network traffic to identify potential issues early.

- **System Updates and Patches:**

Role: DevOps Team

- Applying updates is crucial to security and stability. The DevOps team will be responsible for checking and applying patches weekly. They will review patch notes and test updates in a staging environment before deploying them to production.

2. Logs and Access Control:

- **Weekly Log Review:**

Role: Security Team

The Security Team will perform a comprehensive review of logs weekly, looking for signs of unauthorized access or suspicious behavior. AWS CloudTrail, along with VPC flow logs, will be used to identify and analyze potential security incidents.

- **Access Control Audits:**

Role: IT Compliance Team

The IT Compliance Team will conduct monthly audits of access controls to ensure all roles have appropriate permissions. They will verify that least privilege principles are maintained and that no user has unnecessary administrative rights.

3. Backups and Disaster Recovery:

- **Weekly Backup Verification:**

Role: IT Operations Team

The team will verify weekly that backups of crucial data are occurring as scheduled and that the backups are valid. Automated alerts will be set up to notify the team if any backup job fails.

- **Disaster Recovery Testing:**

Role: Disaster Recovery Team

The Disaster Recovery Team will simulate disaster scenarios monthly to ensure that recovery processes are functional and timely. They will document recovery times and identify areas needing improvement.

4. Security Enhancements:

- **Hardening System Configuration:**

Role: DevOps Team

As part of the hardening process, the DevOps Team will focus on low-hanging fruit, such as implementing stricter IAM policies and improving firewall configurations. They will follow the CIS Benchmarks and AWS Well-Architected Framework security pillars for guidance.

5. Continuous Improvement and Reporting:

- **Monthly Status Reports:**

Role: IT Operations Manager

The IT Operations Manager will consolidate findings from all teams into a monthly status report. This report will summarize key metrics, incidents, and the overall state of the system, as well as provide recommendations for future improvements.

- **Quarterly Review Meetings:**

Role: IT Leadership Team

The IT Leadership Team will convene quarterly to review the status reports and assess progress toward maintaining system security and reliability. They will refine and reprioritize the support plan based on these reviews.

<u>**Incident Response Plan**</u>

The system hosted on our EC2 instance handles sensitive information, including static files, user interaction logs, and web traffic. Therefore, ensuring the integrity and security of the system is paramount. This incident response plan outlines steps to take in the event of a compromise.

1. Assessing Information at Risk:

Sensitive data at risk includes access logs, website content, and configuration settings. If an attacker spies on all traffic or gains credentials to the AWS Management Console, they could manipulate website content, steal sensitive information, or disrupt services by altering access policies or infrastructure settings.

2. Response to System Compromise:

If a compromise is suspected or identified:
1. Immediately rotate all IAM credentials and AWS access keys associated with compromised accounts to block further unauthorized access.
2. Change passwords for EC2 instances, databases, and any other connected services.
3. Disable the affected user accounts and isolate the compromised EC2 instance to prevent further data exfiltration.
4. Scan for malware, tampered files, and unauthorized user accounts using security tools like AWS Inspector or a third-party antivirus solution.
5. Review all IAM policies to identify any unauthorized changes and reset them to the original configurations.

3. Active Attack Handling:

If an attack is currently in progress:
1. Initiate containment measures, such as disabling affected network interfaces and Security Groups to cut off communication channels.
2. Use AWS CloudTrail and VPC Flow Logs to monitor suspicious activities and isolate compromised services by adjusting their permissions or disabling their functionality.
3. Engage incident response team members to coordinate and execute the containment, investigation, and recovery processes.

4. Recovery Process:

1. Assess backups to ensure they are uncompromised. If intact, restore services from the most recent backups using established disaster recovery plans.
2. If backups are also compromised, rebuild critical systems from scratch following documented configurations while validating new access credentials.
3. Implement enhanced security measures, including improved encryption, monitoring, and access control policies.

5. Identifying Attack Origin:

1. Analyze VPC Flow Logs, CloudTrail, and other relevant logs to identify anomalous IP addresses or user activities.
2. Cross-reference log information with threat intelligence sources to determine if the suspicious activities match known attack patterns.
3. Conduct post-incident analysis by collecting forensic data and engaging external cybersecurity specialists if needed.

Project Outcomes and Future Enhancements:

The deployment of a serverless web application on AWS has yielded significant benefits in terms of scalability, reliability, and security. The system successfully showcases the integration of various AWS services to create a robust and responsive web application architecture. The real-time demonstration of our project's deployment and functionality highlights its operational capabilities and responsiveness, providing users with an efficient and seamless experience.

Looking ahead, there are several areas where we can further enhance the system's functionality and security. Future enhancements could include:

Advanced Monitoring and Alerting: Implementing more sophisticated monitoring tools like AWS CloudWatch or third-party solutions to gain deeper insights into system performance and security. Setting up automated alerts for anomalous activities could help preempt security incidents or operational bottlenecks.

Automated Scaling and Load Balancing: As the number of users grows, implementing Auto Scaling groups and Elastic Load Balancing could help manage load efficiently and ensure high availability and fault tolerance.

Enhanced Data Encryption and Security: Exploring more robust encryption options and technologies, such as hardware security modules (HSMs) or using AWS KMS for key management, could provide stronger security. Implementing more granular encryption at the application level could also be considered.

Comprehensive Disaster Recovery Planning: Developing a more detailed disaster recovery plan that includes multi-region deployment to handle major AWS region outages. Testing the recovery process regularly through drills and updating the strategy based on these tests would also be beneficial.

Integration of Artificial Intelligence for Security: Incorporating AI-driven security tools that can predict and identify potential threats based on patterns and unusual activities. Such tools can enhance the detection of zero-day vulnerabilities and automate certain security responses.

Conclusion:

In conclusion, the development and deployment of a serverless web application on AWS have demonstrated the effectiveness of leveraging cloud technologies to create scalable, secure, and efficient solutions. By integrating various AWS services such as S3, CloudFront, Lambda, DynamoDB, and Route 53, we have created a robust architecture that effectively handles user interactions, dynamic content generation, and data storage.

The project not only highlights the technical capabilities of cloud computing but also underscores the importance of security and best practices in cloud deployment. Through careful configuration of IAM roles, access control policies, encryption mechanisms, and monitoring tools, we have ensured the integrity, confidentiality, and availability of our system.

Moving forward, the system will continue to undergo enhancements and improvements to adapt to evolving requirements and address emerging challenges. By embracing a proactive approach to system management and security, we aim to maintain the reliability and efficiency of our serverless architecture while staying resilient to potential threats and vulnerabilities.