

Project: Serverless E-Commerce Order Processing System on AWS

GitHub Repo: <https://github.com/Mithra1995/serverless-project>

Objective:

Develop a secure, scalable, and cost-efficient e-commerce order processing system using serverless AWS services, including a proper web-based UI and web application firewall (WAF) for protection against common web threats.

High-Level Architecture:

- Frontend: Hosted on Amazon S3 (static website) + CloudFront (CDN)
- API Layer: Managed through API Gateway
- Business Logic: Handled by AWS Lambda
- Database: Powered by DynamoDB
- Media Storage: Product images and frontend assets stored in S3
- Security: AWS WAF integrated with CloudFront & API Gateway

◆ Key Features

- Product listing UI
- Add to cart & order placement
- Order data stored securely in DynamoDB
- Secured API endpoints using AWS WAF

Step by Step Implementation:

Step 1: Create Lambda function for Listing products, Add to cart and order placement

- listProductsLambda
- addToCartLambda
- placeOrderLambda

The screenshot shows the AWS Lambda console. The left sidebar is titled 'Lambda' and includes sections for 'Dashboard', 'Applications', 'Functions' (selected), 'Additional resources' (Code signing configurations, Event source mappings, Layers, Replicas), and 'Related AWS resources' (Step Functions state machines). The main content area is titled 'Functions (3)' and displays a table with three rows. The columns are: Function name, Description, Package type, Runtime, and Last modified. The functions listed are:

Function name	Description	Package type	Runtime	Last modified
addToCartLambda	-	Zip	Python 3.13	6 hours ago
getProductsLambda	-	Zip	Python 3.13	7 hours ago
placeOrderLambda	-	Zip	Python 3.13	2 hours ago

At the top right, there are buttons for 'Actions' and 'Create function'. The top navigation bar includes the AWS logo, account information ('United States (N. Virginia) Mithra @ 3828-2859-5676'), and various global navigation icons.

The screenshot shows the AWS Lambda Function Editor interface. The top navigation bar includes the AWS logo, a search bar, and tabs for Lambda, Functions, and addToCartLambda. The main area is titled "Code source" with an "Info" tab. On the left, the "EXPLORER" sidebar shows a file tree with "ADDCARTLAMBDA" expanded, containing "lambda_function.py". Below it, "DEPLOY" has two buttons: "Deploy (Ctrl+Shift+U)" and "Test (Ctrl+Shift+T)". Under "TEST EVENTS [SELECTED: TEST]", there are options to "Create new test event" and "Private saved events". The central workspace displays the Python code for "lambda_function.py". The right side features a "PROBLEMS" panel with one error, a "CODE REFERENCE LOG" panel, and a "TERMINAL" panel. At the bottom, there are links for CloudShell, Feedback, and various AWS services like CloudWatch, CloudFront, and CloudWatch Metrics. The footer includes copyright information for 2025, privacy terms, and cookie preferences.

The screenshot shows the AWS Lambda console interface. The top navigation bar includes the AWS logo, a search bar, and account information: United States (N. Virginia) and Mithra @ 3828-2859-3676. The main area displays the 'getProductsLambda' function under the 'API Gateway' trigger. The 'Code' tab is selected, showing the code source in the 'EXPLORER' panel. The code file 'lambda_function.py' contains the following Python script:

```
1 import json
2 import boto3
3 from decimal import Decimal
4 dynamodb = boto3.resource('dynamodb')
5 table = dynamodb.Table('Products')
6
7 class DecimalEncoder(json.JSONEncoder):
```

The screenshot shows the AWS Lambda console interface. The top navigation bar includes the AWS logo, a search bar, and account information: United States (N. Virginia) and Mithra @ 3828-2859-3676. The main area displays the 'placeOrderLambda' function under the 'API Gateway' trigger. The 'Code' tab is selected, showing the code source in the 'EXPLORER' panel. The code file 'lambda_function.py' contains the following Python script:

```
1 import json
2 import boto3
3 import uuid
4 from datetime import datetime
5 from decimal import Decimal
6 dynamodb = boto3.resource('dynamodb')
7 cart_table = dynamodb.Table('Cart')
```

Step 2: Add IAM permission to lambda role to access DynamoDB

The screenshot shows the AWS IAM Roles page. The top navigation bar includes the AWS logo, a search bar, and a global status indicator for Mithra. The main content area displays the LambdaDynamoDbPermission role under the 'Identity and Access Management (IAM)' section. The 'Permissions' tab is selected, showing two managed policies attached: 'AmazonAPIGatewayAdministrator' and 'AmazonDynamoDBFullAccess'. The 'Attached entities' column indicates 1 and 4 respectively. Other tabs include 'Trust relationships', 'Tags', 'Last Accessed', and 'Revoke sessions'. The left sidebar lists 'Access management' and 'Access reports' sections.

Step 3: Create DynamoDB table for cart, order and products table

The screenshot shows the AWS DynamoDB Tables page. The top navigation bar includes the AWS logo, a search bar, and a global status indicator for Mithra. The main content area displays three tables: 'Cart', 'Orders', and 'Products'. The 'Tables' section is selected in the sidebar. The 'Create table' button is visible at the top right. The table details are as follows:

Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favor
Cart	Active	user_id (\$)	product_id (\$)	0	0	Off	
Orders	Active	order_id (\$)	-	0	0	Off	
Products	Active	product_id (\$)	-	0	0	Off	

The left sidebar lists 'Tables' and 'DAX' sections.

Step 4: Add List or Products in DynamoDB table with productid, product name, price and Image URL

Screenshot of the AWS DynamoDB console showing the 'Explore items' page for the 'Products' table.

The table contains the following data:

product_id	image_url	price	product_name
p002	https://la...	649.99	Bluetooth Speaker
p004	https://la...	24999.99	Laptop
p001	https://la...	699.99	Smartwatch
p005	https://la...	13999.99	Mobile Phone
p006	https://la...	15999.99	Television
p003	https://la...	999.99	Fan

Image stored in s3

Screenshot of the AWS S3 console showing the contents of the 'lambda triggers3bucket-ncpl1' bucket.

The bucket contains the following objects:

Name	Type	Last modified	Size	Storage class
Bluetoothspeaker.jpg	jpg	July 31, 2025, 12:40:57 (UTC-04:00)	438.9 KB	Standard
Fan.jpg	jpg	July 31, 2025, 12:40:56 (UTC-04:00)	409.0 KB	Standard
Laptop.jpg	jpg	July 31, 2025, 12:40:57 (UTC-04:00)	13.4 KB	Standard
Mobilephone.jpg	jpg	July 31, 2025, 12:40:55 (UTC-04:00)	497.5 KB	Standard
smartwatch.jpg	jpg	July 31, 2025, 12:30:01 (UTC-04:00)	192.2 KB	Standard

Step 5: Create API gateway as a Endpoints to trigger lambda function for cart, order and products

The screenshot shows the AWS API Gateway interface. On the left, there's a sidebar with 'APIs' selected, showing options like Custom domain names, Domain name access associations, VPC links, Usage plans, API keys, Client certificates, and Settings. The main area is titled 'APIs (1/1)' and lists a single API named 'EcommerceAPI'. The table row for 'EcommerceAPI' includes columns for Name (EcommerceAPI), ID (7uiudysol), Protocol (REST), API endpoint type (Edge-optimized), and Created (2025-07-31). There are 'Delete' and 'Create API' buttons at the top right of the table.

The screenshot shows the AWS API Gateway interface, specifically the 'Resources' section for the 'EcommerceAPI'. The left sidebar has 'API: EcommerceAPI' expanded, showing 'Resources', 'Stages', 'Authorizers', 'Gateway responses', 'Models', 'Resource policy', 'Documentation', 'Dashboard', and 'API settings'. The main area shows the 'Resources' list with entries for '/', '/cart', '/order', and '/products'. For the '/cart' resource, a detailed view is shown for the 'POST' method. It shows the ARN (arn:aws:execute-api:us-east-1:382828593676:7uiudysol/*/POST/cart), Resource ID (diOkak), and a flow diagram. The flow starts with a 'Client' sending a 'Method request' to the API, which then sends an 'Integration request' to a 'Lambda integration' (Function addToCartLambda). The Lambda integration returns a 'Method response' to the API, which then returns it to the Client. Buttons for 'API actions' and 'Deploy API' are visible at the top right.

Screenshot of the AWS API Gateway Resources page for the EcommerceAPI.

Left Sidebar:

- APIs
- Custom domain names
- Domain name access associations
- VPC links
- API: EcommerceAPI**
 - Resources
 - Stages
 - Authorizers
 - Gateway responses
 - Models
 - Resource policy
 - Documentation
 - Dashboard
 - API settings

Top Bar:

- Search bar
- [Alt+S]
- Icons for Home, Help, and Settings
- United States (N. Virginia)
- Mithra @ 3828-2859-3676

Resources List:

- Create resource
- / (Root Resource)
 - /cart (OPTIONS, POST)
 - /order (OPTIONS, POST)
 - /products (GET, OPTIONS)

Method Execution Diagram for /order - POST:

```

graph LR
    Client --> MethodRequest[Method request]
    MethodRequest --> IntegrationRequest[Integration request]
    IntegrationRequest --> LambdaIntegration[Lambda integration]
    LambdaIntegration --> MethodResponse[Method response]
    MethodResponse --> Client
    subgraph LambdaIntegration [Lambda integration]
        direction TB
        subgraph Function [Function placeOrderLambda]
            direction TB
            InRequest[In request] --> PlaceOrderLambda[placeOrderLambda]
            PlaceOrderLambda --> OutResponse[Out response]
            OutResponse --> LambdaIntegration
        end
    end

```

Details for /order - POST:

- ARN: arn:aws:execute-api:us-east-1:382828593676:7uiiudysol/*/POST/order
- Resource ID: hqbeu8

Bottom Bar:

- CloudShell
- Feedback
- © 2025, Amazon Web Services, Inc. or its affiliates.
- Privacy
- Terms
- Cookie preferences

Screenshot of the AWS API Gateway Resources page for the EcommerceAPI.

Left Sidebar:

- APIs
- Custom domain names
- Domain name access associations
- VPC links
- API: EcommerceAPI**
 - Resources
 - Stages
 - Authorizers
 - Gateway responses
 - Models
 - Resource policy
 - Documentation
 - Dashboard
 - API settings

Top Bar:

- Search bar
- [Alt+S]
- Icons for Home, Help, and Settings
- United States (N. Virginia)
- Mithra @ 3828-2859-3676

Resources List:

- Create resource
- / (Root Resource)
 - /cart (OPTIONS, POST)
 - /order (OPTIONS, POST)
 - /products (GET, OPTIONS)

Method Execution Diagram for /products - GET:

```

graph LR
    Client --> MethodRequest[Method request]
    MethodRequest --> IntegrationRequest[Integration request]
    IntegrationRequest --> LambdaIntegration[Lambda integration]
    LambdaIntegration --> MethodResponse[Method response]
    MethodResponse --> Client
    subgraph LambdaIntegration [Lambda integration]
        direction TB
        subgraph Function [Function getProductsLambda]
            direction TB
            InRequest[In request] --> GetProductsLambda[getProductsLambda]
            GetProductsLambda --> OutResponse[Out response]
            OutResponse --> LambdaIntegration
        end
    end

```

Details for /products - GET:

- ARN: arn:aws:execute-api:us-east-1:382828593676:7uiiudysol/*/GET/products
- Resource ID: xz2lij

Bottom Bar:

- CloudShell
- Feedback
- © 2025, Amazon Web Services, Inc. or its affiliates.
- Privacy
- Terms
- Cookie preferences

Step 6: Enable CORS for all the API Endpoints

The screenshot shows the AWS API Gateway interface. On the left, the navigation sidebar includes 'APIs', 'Custom domain names', 'Domain name access associations', 'VPC links', and a expanded section for 'API: EcommerceAPI' which lists 'Resources', 'Stages', 'Authorizers', 'Gateway responses', 'Models', 'Resource policy', 'Documentation', 'Dashboard', and 'API settings'. The main content area displays the 'Method responses' for the 'EcommerceAPI' resources. It shows a tree view of methods: /cart (OPTIONS, POST), /order (OPTIONS, POST), and /products (GET, OPTIONS). The 'Response 200' details for the /products/GET method are shown, including 'Response headers' (Content-Type: application/json) and an empty 'Response body'. The bottom right corner of the interface includes copyright information: '© 2025, Amazon Web Services, Inc. or its affiliates.' and links for 'Privacy', 'Terms', and 'Cookie preferences'.

Step 7: Deploy API and name it as "Prod"

The screenshot shows the AWS API Gateway interface. The navigation sidebar is identical to the previous screenshot. The main content area displays the 'Stage details' for the 'Prod' stage of the 'EcommerceAPI'. It shows the stage name 'Prod', rate limit '10000', burst limit '5000', and an inactive cache cluster. The 'Invoke URL' is listed as <https://7uiiudysol.execute-api.us-east-1.amazonaws.com/Prod>. Below this, an 'Active deployment' is shown as 'c28q9m on July 31, 2025, 22:13 (UTC-04:00)'. The 'Logs and tracing' section indicates that CloudWatch logs, detailed metrics, and data tracing are all inactive. The bottom right corner includes copyright information and links for 'Privacy', 'Terms', and 'Cookie preferences'.

Step 8: Add index.html and app.js in another s3 bucket by enabling static web hosting

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with various options like General purpose buckets, Vector buckets, Access Grants, and Storage Lens. The main area is titled "myncplbuckeencryption1" and shows "Objects (2)". There are buttons for Actions, Copy S3 URI, Copy URL, Download, Open, Delete, and Create folder. Below these buttons is an "Upload" button. A note says: "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." A search bar labeled "Find objects by prefix" and a "Show versions" link are also present. The object list table has columns for Name, Type, Last modified, Size, and Storage class. The objects listed are "app.js" (js, 3.2 KB, Standard) and "index.html" (html, 3.9 KB, Standard). At the bottom right of the main area, there are links for "Privacy", "Terms", and "Cookie preferences".

The screenshot shows the static website hosting configuration for the "myncplbuckeencryption1" bucket. The "Hosting type" is set to "Bucket hosting". The "Bucket website endpoint" is displayed as "<http://myncplbuckeencryption1.s3-website-us-east-1.amazonaws.com>". A callout box recommends using AWS Amplify Hosting for static website hosting, with links to "Create Amplify app" and "View your existing Amplify apps". The sidebar on the left includes options like General purpose buckets, Vector buckets, Access Grants, and Storage Lens.

Step 9: Disable Block public access to access the website publicly

The screenshot shows the AWS S3 console. On the left, there's a sidebar with navigation links like 'Amazon S3', 'General purpose buckets', 'Storage Lens', and 'CloudShell'. The main content area is titled 'Permissions overview' and contains sections for 'Access finding', 'Block public access (bucket settings)', and 'Bucket policy'. The 'Bucket policy' section displays a JSON policy document:

```
{ "Version": "2008-10-17", "Id": "PolicyForCloudFrontPrivateContent", "Statement": [ { "Sid": "AllowCloudFrontServicePrincipal", "Effect": "Allow", "Principal": { "Service": "cloudfront.amazonaws.com" }, "Action": "*S3:GetObject", "Resource": "arn:aws:s3:::myncplbuckeencryption1/*", "Condition": { "StringEquals": { "AWS:SourceArn": "arn:aws:cloudfront::382828593676:distribution/E3H17JPU3TATJR" } } } ] }
```

Step 10 : Add bucket policy to access S3 using CloudFront

This screenshot shows the 'Bucket policy' editor for the same bucket. The JSON policy is identical to the one shown in the previous screenshot:

```
{ "Version": "2008-10-17", "Id": "PolicyForCloudFrontPrivateContent", "Statement": [ { "Sid": "AllowCloudFrontServicePrincipal", "Effect": "Allow", "Principal": { "Service": "cloudfront.amazonaws.com" }, "Action": "*S3:GetObject", "Resource": "arn:aws:s3:::myncplbuckeencryption1/*", "Condition": { "StringEquals": { "AWS:SourceArn": "arn:aws:cloudfront::382828593676:distribution/E3H17JPU3TATJR" } } } ] }
```

The screenshot shows the AWS S3 console with the path `Amazon S3 > Buckets > myncplbuckeencryption1`. On the left, the sidebar lists various S3 features like General purpose buckets, Access Points, and Storage Lens. The main content area is titled "Cross-origin resource sharing (CORS)" and displays a JSON configuration block:

```
[{"AllowedHeaders": ["*"], "AllowedMethods": ["GET"], "AllowedOrigins": ["*"], "ExposeHeaders": [], "MaxAgeSeconds": 3000}]
```

A "Copy" button is available to the right of the JSON code. The bottom of the page includes standard AWS footer links: © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

Step 11: Create CloudFront for S3

The screenshot shows the AWS CloudFront console with the path `CloudFront > Distributions`. The sidebar lists CloudFront features like Policies, Functions, and Static IPs. The main content area shows a table of distributions with one entry highlighted:

ID	Status	Description	Type	Domain name (stage)
E3H17JPU3TATJR	Enabled	-	Standard	d3h33nn8b5quv.cloudfront.net

The screenshot shows the AWS CloudFront console with the path `CloudFront > Distributions > E3H17JPU3TATJR`. The sidebar includes sections for SaaS, Telemetry, and Reports & analytics. The main content area is titled "ecommerce Standard" and shows the "General" tab selected. It displays distribution details such as Name (ecommerce), Distribution domain name (d3h33nn8b5quv.cloudfront.net), ARN (arn:aws:cloudfront::382828593676:distribution/E3H17JPU3TATJR), and Last modified (August 1, 2025 at 4:00:56 AM UTC). The "Settings" section includes fields for Description, Price class (Use all edge locations (best performance)), Supported HTTP versions (HTTP/2, HTTP/1.1, HTTP/1.0), Alternate domain names (with an "Add domain" button), Standard logging (Off), Cookie logging (Off), and Default root object (-).

Enable security protections to view the full security dashboard for this distribution
Gain visibility into top security trends, allowed and blocked traffic, as well as visibility and controls for bots. You'll later have the option to enable security logs to investigate traffic patterns—visually search, filter, and inspect individual log lines without writing queries. [Learn more](#)

Manage security protections

Security - Web Application Firewall (WAF) [Info](#) [Manage protections](#)

Keep your application secure from the most common web threats and security vulnerabilities using AWS WAF. Blocked requests are stopped before they reach your web servers.

Core protections **Disabled**

CloudFront geographic restrictions

Type **Allow list** Countries - [Edit](#) **Canada**

Step 12: Add OAC for the CloudFront and update the same in S3 bucket policy

View metrics

Origins

Filter origins by property or value

Origin name	Origin domain
myncplbuckeenryption1.s3.us-east-1.amazonaws.com-mdsahr1h1rj	myncplbuckeenryption1.s3.us-east-1.amazonaws.com
myncplbuckeenryption1.s3.us-east-1.amazonaws.com	myncplbuckeenryption1.s3.us-east-1.amazonaws.com

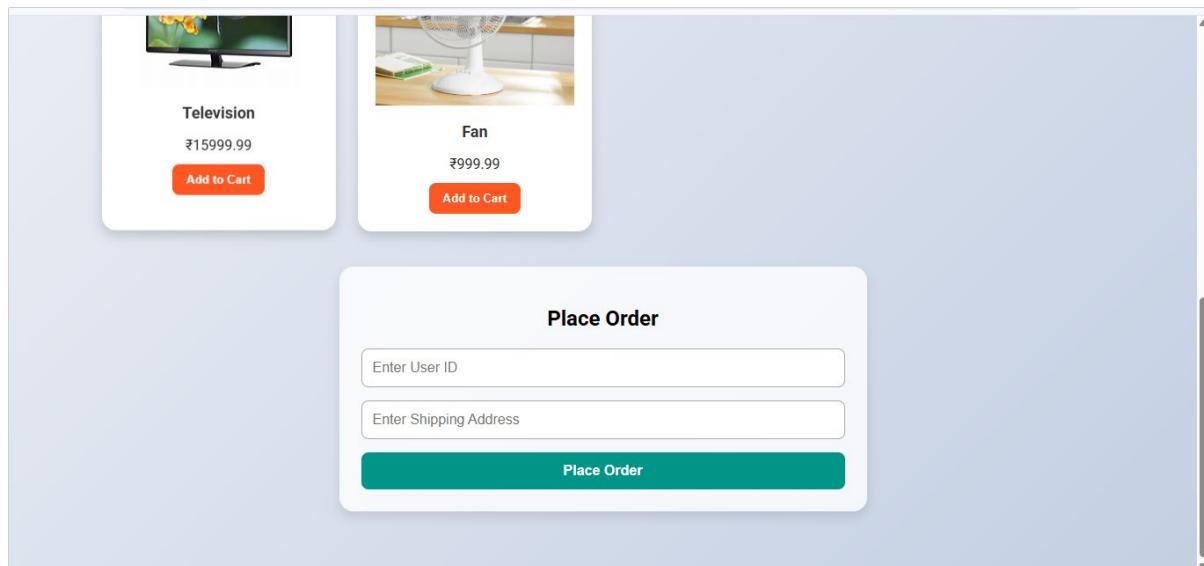
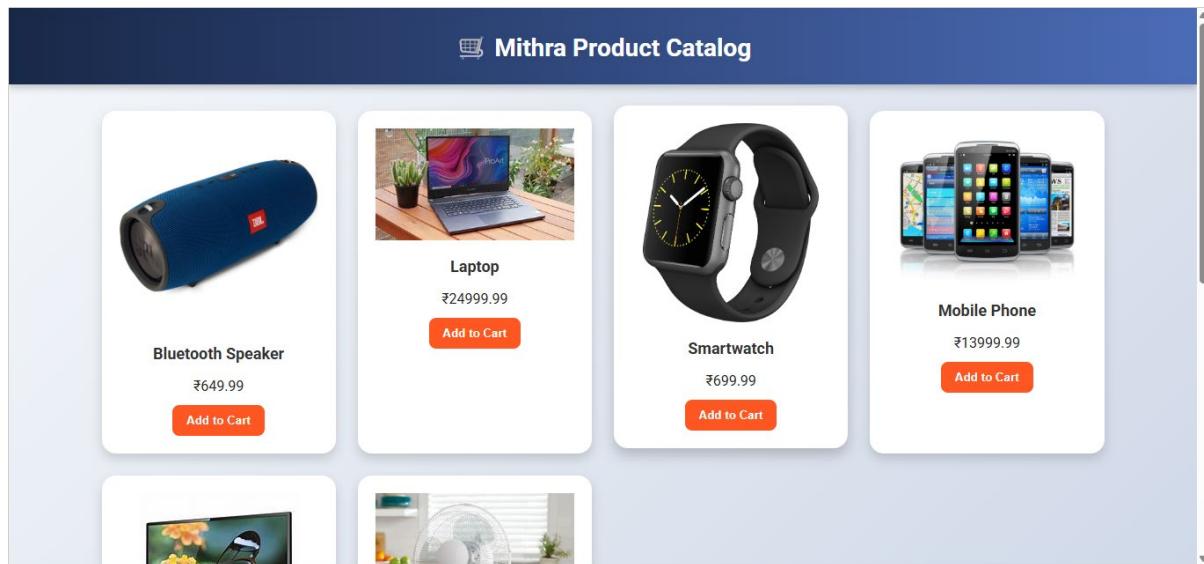
Origin groups

Filter origin groups by property or value

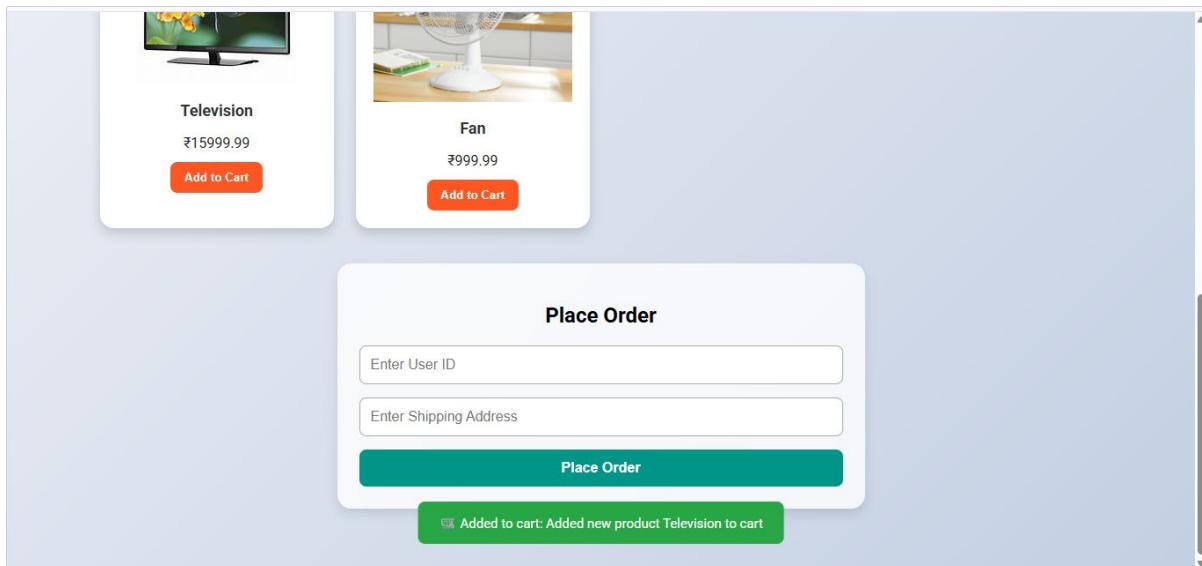
Origin group name	Origins	Failover criteria

Step 13: Access the CloudFront webpage using below URL

<https://d3h33nn8b5qupv.cloudfront.net/index.html> and list of products will get displayed



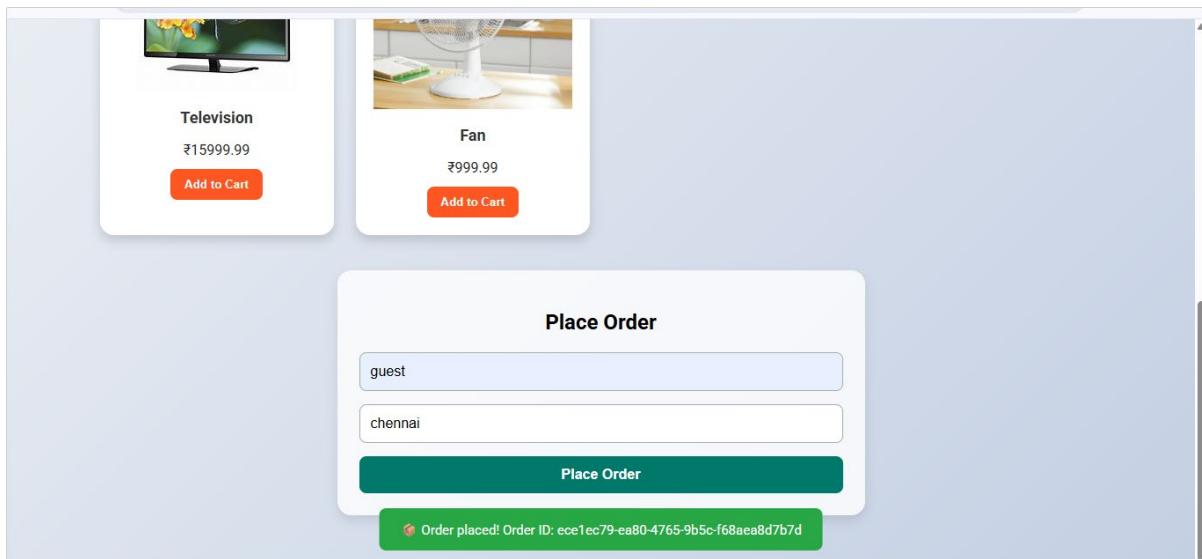
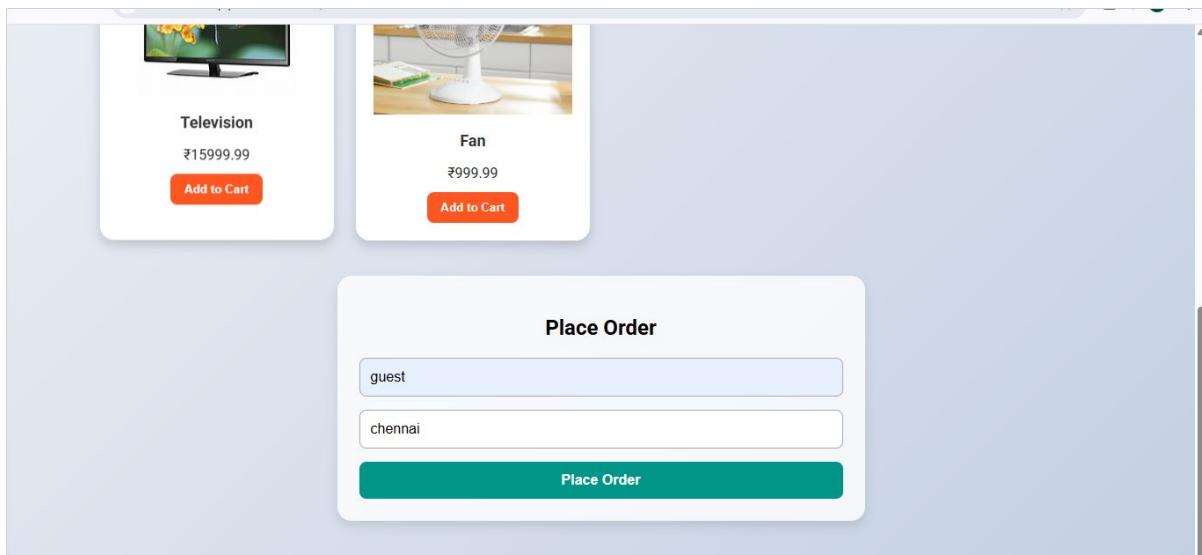
Step 14: Now add the products to cart and check the products are getting added in DynamoDB and also message will get poped-up once the product is added to cart



Step 15: And the products are added in DynamoDB cart table

A screenshot of the AWS DynamoDB 'Explore items' section. On the left, a sidebar shows navigation options like Dashboard, Tables, Explore items, and DAX. The main area shows a table named 'Cart - Items returned (2)'. The table has columns: user_id..., produ..., price, product_name, and quantity. Two items are listed: one 'guest' user has a 'Fan' at price 999.99 and quantity 1, and another 'guest' user has a 'Television' at price 15999.99 and quantity 1. A status bar at the bottom indicates the scan started on August 01, 2025, at 00:37:38.

Step 16: Now place the order using the user_id: guest and Address as “Chennai” and once the order is placed successfully the message will be poped-up



Output:

Step 17: Once the order is placed, we can see the cart becomes empty

Completed - Items returned: 0 · Items scanned: 0 · Efficiency: 100% · RCUs consumed: 2

Table: Cart - Items returned (0)

No items

No items to display.

Create item

Step 18: Then the orders are placed successfully is added in Orders DynamoDB table

Completed - Items returned: 1 · Items scanned: 1 · Efficiency: 100% · RCUs consumed: 2

Table: Orders - Items returned (1)

order_id	address	order_items	timestamp	total_amount
ece1ec79-...	chennai	[{"M": {"qua..."}]	2025-08-0...	16999.98

Screenshot of the AWS DynamoDB console showing the 'Edit item' screen for a table named 'Orders'. The item has the following attributes:

Attribute name	Value	Type	Action
order_id - Partition key	ece1ec79-ea80-4765-9b5c-f68aea8d7b7d	String	
address	chennai	String	Remove
order_items	Insert a field ▾	List	Remove
timestamp	2025-08-01T04:38:40.120192	String	Remove
total_amount	16999.98	Number	Remove
user_id	guest	String	Remove

Buttons at the bottom: [Cancel](#), [Save](#), [Save and close](#).

Step 19: If we add the same product twice the product quantity will get increased and price also will be updated

Screenshot of the AWS DynamoDB console showing the 'Explore items' screen for the 'Cart' table. The table contains one item:

user_id	product_id	price	product_name	quantity
guest	p006	31999.98	Television	2

Left sidebar navigation includes: Dashboard, Tables, Explore items (selected), PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations (New), Reserved capacity, Settings, DAX (Clusters, Subnet groups, Parameter groups, Events).

DynamoDB > Explore items > Orders

Table - Orders All attributes

Filters - optional

Run Reset

Completed - Items returned: 1 · Items scanned: 1 · Efficiency: 100% · RCU's consumed: 2

Table: Orders - Items returned (1)

Scan started on August 01, 2025, 00:49:58

	order_id (String)	timestamp	total_amount	user_id
	ece1ec79-ea80-4765-9b5c-f68aea8d7b7d	5-08-0...	16999.98	guest

Step 20: Try accessing the S3 static website hosting URL it will show as “Access denied” since the bucket policy is updated to CloudFront , so the website will only be accessible through CloudFront URL

Not secure myncplbuckeencryption1.s3-website-us-east-1.amazonaws.com

403 Forbidden

- Code: AccessDenied
- Message: Access Denied
- RequestId: PA18XZWP4R78CE5
- HostId: bO7rx/gf1/3E4vM6ZCZWVwsO6jwntlucy/vgsq69t9IVo6Dye86yRVFGGvpIVvqlD00R59I8i3M=

Step 21: Now add the WAF config rules for CloudFront(Global) and API endpoint(Regional)

The screenshot shows the AWS WAF Web ACLs page. The left sidebar has sections for AWS WAF (Getting started, Web ACLs, Bot control dashboard, Application integration, IP sets, Regex pattern sets, Rule groups, Add-on protections) and AWS Shield (Getting started). The main content area shows a table titled "Web ACLs (0)" with columns Name, Description, ARN, and ID. A message at the top says, "No web ACLs found. You don't have any web ACLs in the US East (N. Virginia) Region created with this version of AWS WAF. Resources created under AWS WAF Classic aren't compatible with the new AWS WAF. If you are looking for web ACLs created in the past, please check the AWS WAF Classic console. Please click [here](#) for more information." A "Create web ACL" button is at the bottom.

The screenshot shows the "Create web ACL" configuration page. It includes fields for CloudWatch metric name (EcommerceWAF), Associated AWS resources (CloudFront Distribution E3H17JPU3TATJR - d3h33nn8b5qupv.cloudfront.net), and Web request body inspection settings (Body size limit set to 16 KB). The page also includes links for AWS WAF Pricing and CloudFront distribution and AWS Amplify.

Step 22: Enable free AWS rules for CloudFront

Name	Capacity	Action
Admin protection Contains rules that allow you to block external access to exposed admin pages. This may be useful if you are running third-party software or would like to reduce the risk of a malicious actor gaining administrative access to your application. Learn More	100	<input checked="" type="checkbox"/> Add to web ACL Edit
Amazon IP reputation list This group contains rules that are based on Amazon threat intelligence. This is useful if you would like to block sources associated with bots or other threats. Learn More	25	<input checked="" type="checkbox"/> Add to web ACL Edit
Anonymous IP list This group contains rules that allow you to block requests from services that allow obfuscation of viewer identity. This can include request originating from VPN, proxies, Tor nodes, and hosting providers. This is useful if you want to filter out viewers that may be trying to hide their identity from your application. Learn More	50	<input checked="" type="checkbox"/> Add to web ACL Edit
Core rule set Contains rules that are generally applicable to web applications. This provides protection against exploitation of a wide range of vulnerabilities, including those described in OWASP publications. Learn More	700	<input checked="" type="checkbox"/> Add to web ACL Edit
Known bad inputs Contains rules that allow you to block request patterns that are known to be invalid and are associated with exploitation or discovery of vulnerabilities. This can help reduce the risk of a malicious actor discovering a vulnerable application. Learn More	200	<input checked="" type="checkbox"/> Add to web ACL Edit

Rules (6)			
	Name	Capacity	Action
<input type="radio"/>	AWS-AWSManagedRulesAdminProtectionRuleSet	100	Use rule actions
<input type="radio"/>	AWS-AWSManagedRulesCommonRuleSet	700	Use rule actions
<input type="radio"/>	AWS-AWSManagedRulesKnownBadInputsRuleSet	200	Use rule actions
<input type="radio"/>	AWS-AWSManagedRulesAmazonIpReputationList	25	Use rule actions
<input type="radio"/>	AWS-AWSManagedRulesAnonymousIpList	50	Use rule actions
<input type="radio"/>	AWS-AWSManagedRulesWindowsRuleSet	200	Use rule actions

[Cancel](#) [Previous](#) **Next**

Screenshot of the AWS CloudFront Create Web ACL Step 4: Configure metrics page.

Configure metrics

Step 4: AWS-AWSManagedRulesCommonRuleSet
Step 5: Review and create web ACL

AWS-AWSManagedRulesKnownBadInputsRuleSet
 AWS-AWSManagedRulesAmazonIpReputationList
 AWS-AWSManagedRulesAnonymousIpList
 AWS-AWSManagedRulesWindowsRuleSet

Request sampling options
If you disable request sampling, you can't view requests that match your web ACL rules.

Options
 Enable sampled requests
 Disable sampled requests
 Enable sampled requests with exclusions

Cancel Previous Next

Screenshot of the AWS CloudFront Create Web ACL Step 5: Amazon CloudWatch metrics page.

Amazon CloudWatch metrics (6)

Rules	CloudWatch metric name
AWS-AWSManagedRulesAdminProtectionRuleSet	AWS-AWSManagedRulesAdminProtectionRuleSet
AWS-AWSManagedRulesCommonRuleSet	AWS-AWSManagedRulesCommonRuleSet
AWS-AWSManagedRulesKnownBadInputsRuleSet	AWS-AWSManagedRulesKnownBadInputsRuleSet
AWS-AWSManagedRulesAmazonIpReputationList	AWS-AWSManagedRulesAmazonIpReputationList
AWS-AWSManagedRulesAnonymousIpList	AWS-AWSManagedRulesAnonymousIpList
AWS-AWSManagedRulesWindowsRuleSet	AWS-AWSManagedRulesWindowsRuleSet

Sampled requests
Sampled requests for web ACL default actions
Enabled

Cancel Previous Create web ACL

Step 23 : Now the WAF is enabled for CloudFront

The screenshot shows the AWS WAF & Shield console. On the left, there's a sidebar with options like AWS WAF (selected), AWS Shield, and CloudShell. The main area displays a success message: "You successfully created the web ACL EcommerceWAF." Below this, the "Web ACLs" section shows a table with one entry:

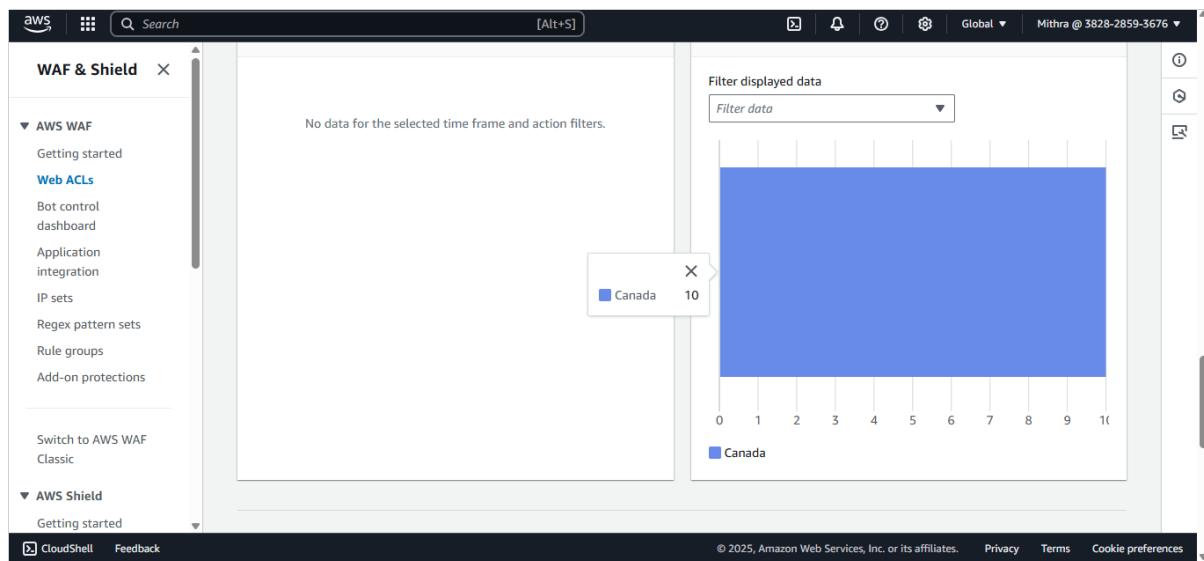
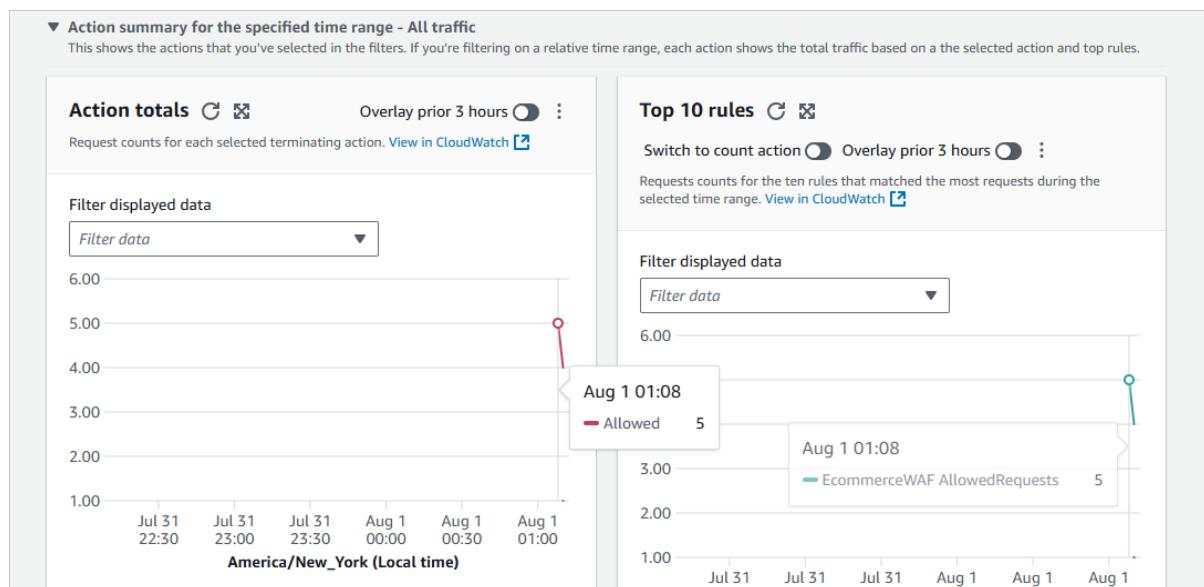
Name	Description	ARN	ID
EcommerceWAF	-	arn:aws:wafv2:us-east-1:382828593676:... fc44a1a9-11aa-4a3f-8faf-59825028ff64	

Step 24 : Now we can see the traffic displayed for CloudFront

The screenshot shows the AWS WAF & Shield console. The main dashboard displays traffic totals for the specified time range:

Total	Blocked	Allowed	Captcha
10 100%	1 100%	9 100%	0

Below this, there's a section titled "Action summary for the specified time range - All traffic".



Step 25: Add WAF for API Endpoint

Add AWS resources

Resource type
Select the resource type and then select the resource you want to associate with this web ACL.

<input type="radio"/> Application Load Balancer	<input checked="" type="radio"/> Amazon API Gateway REST API	<input type="radio"/> Amazon App Runner service
<input type="radio"/> AWS AppSync API	<input type="radio"/> Amazon Cognito user pool	<input type="radio"/> AWS Verified Access

Resources (1)
Select the resource you want to associate with the web ACL.

<input type="checkbox"/> Name
<input checked="" type="checkbox"/> EcommerceAPI - Prod

Cancel **Add** **Next**

Amazon CloudWatch metrics (6)

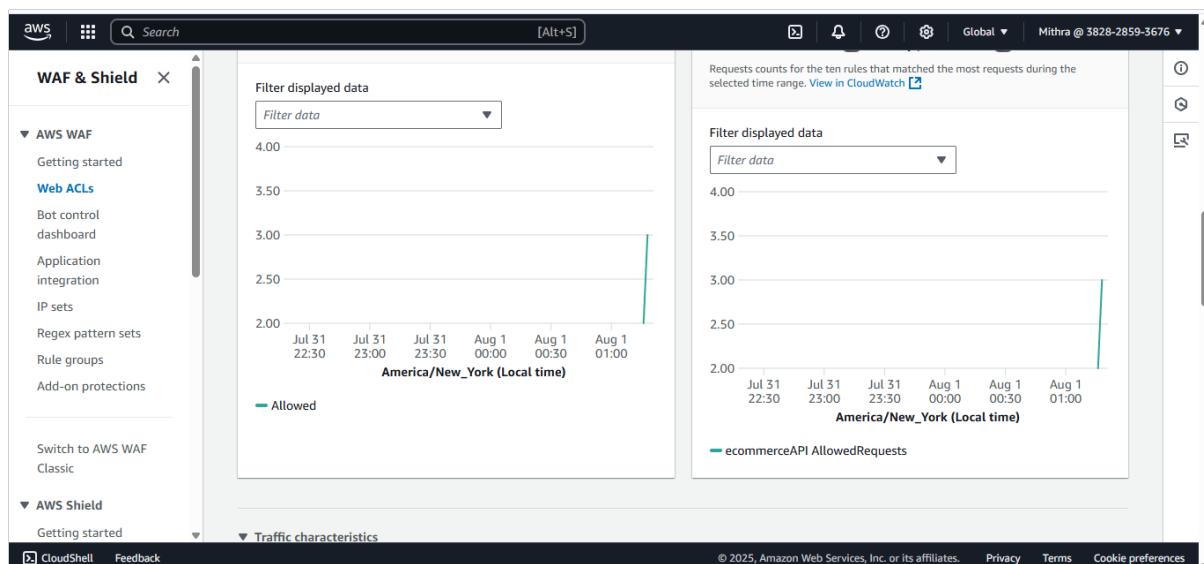
Rules	CloudWatch metric name
AWS-AWSManagedRulesAnonymousIpList	AWS-AWSManagedRulesAnonymousIpList
AWS-AWSManagedRulesAmazonIpReputationList	AWS-AWSManagedRulesAmazonIpReputationList
AWS-AWSManagedRulesAdminProtectionRuleSet	AWS-AWSManagedRulesAdminProtectionRuleSet
AWS-AWSManagedRulesCommonRuleSet	AWS-AWSManagedRulesCommonRuleSet
AWS-AWSManagedRulesKnownBadInputsRuleSet	AWS-AWSManagedRulesKnownBadInputsRuleSet
AWS-AWSManagedRulesWindowsRuleSet	AWS-AWSManagedRulesWindowsRuleSet

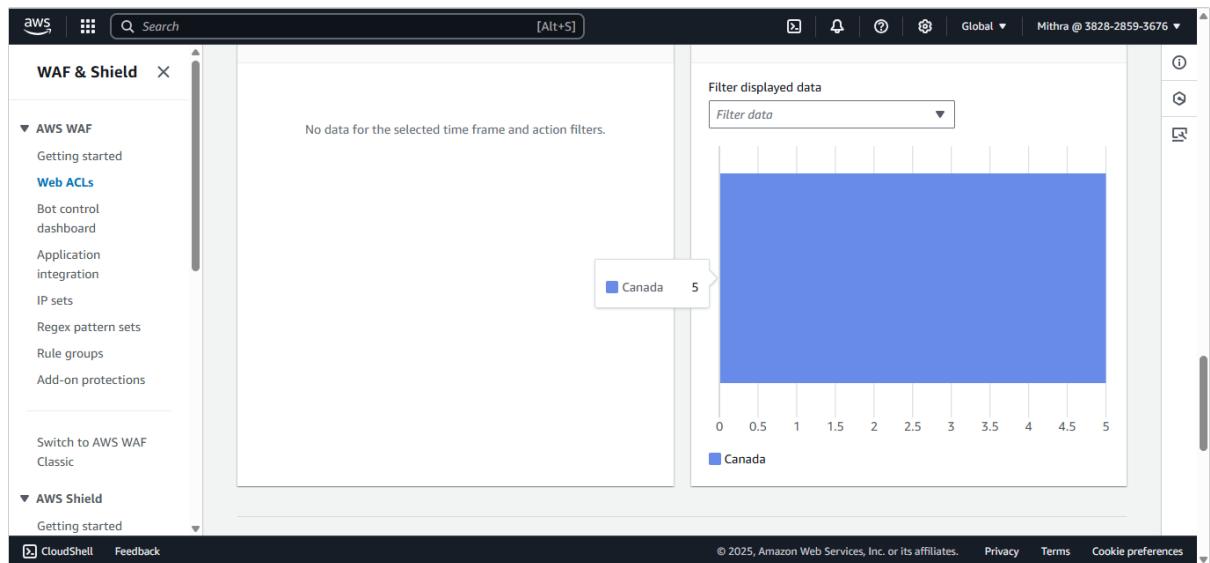
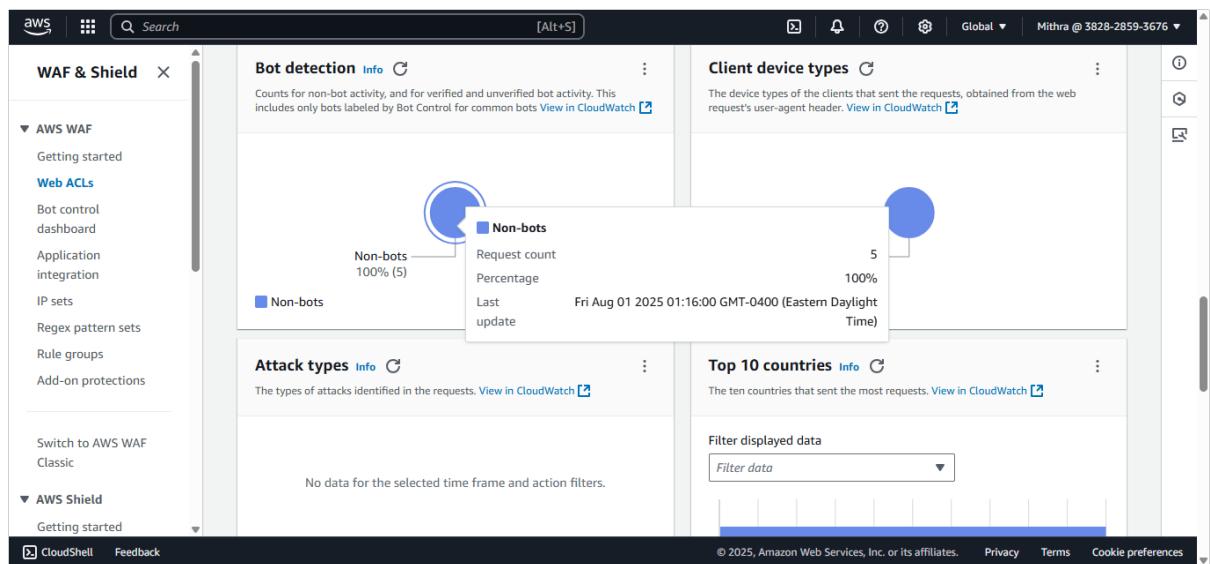
Sampled requests
Sampled requests for web ACL default actions
Enabled

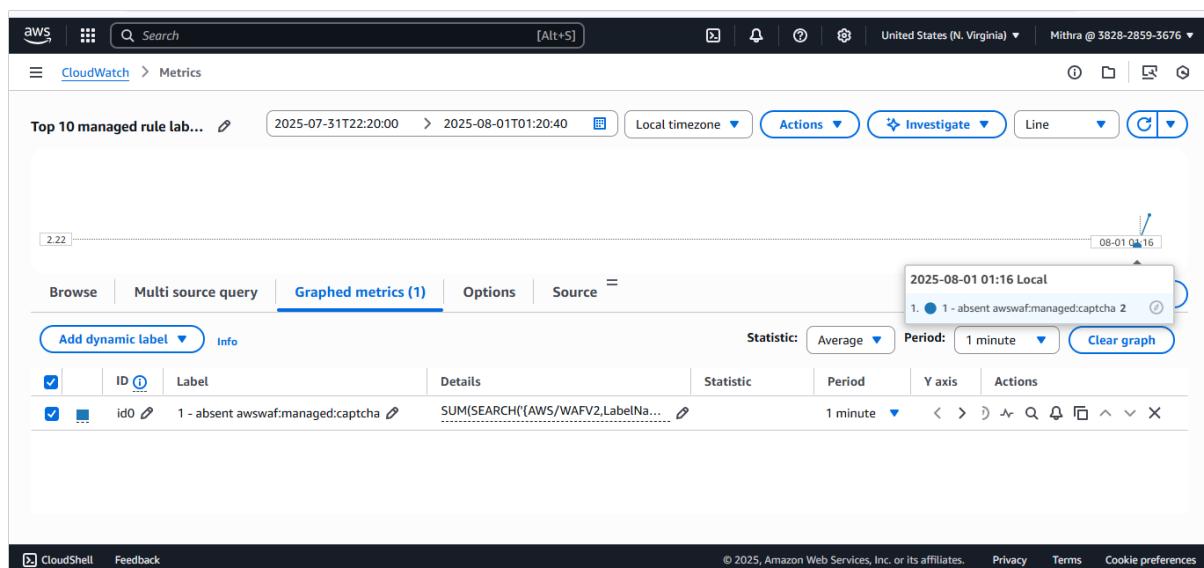
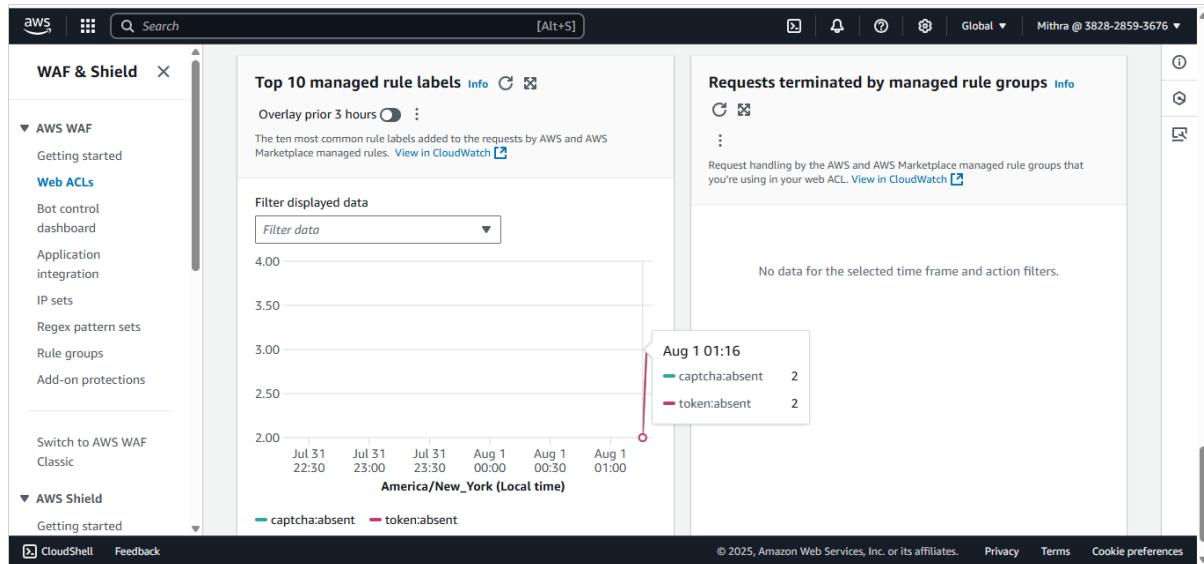
Create web ACL

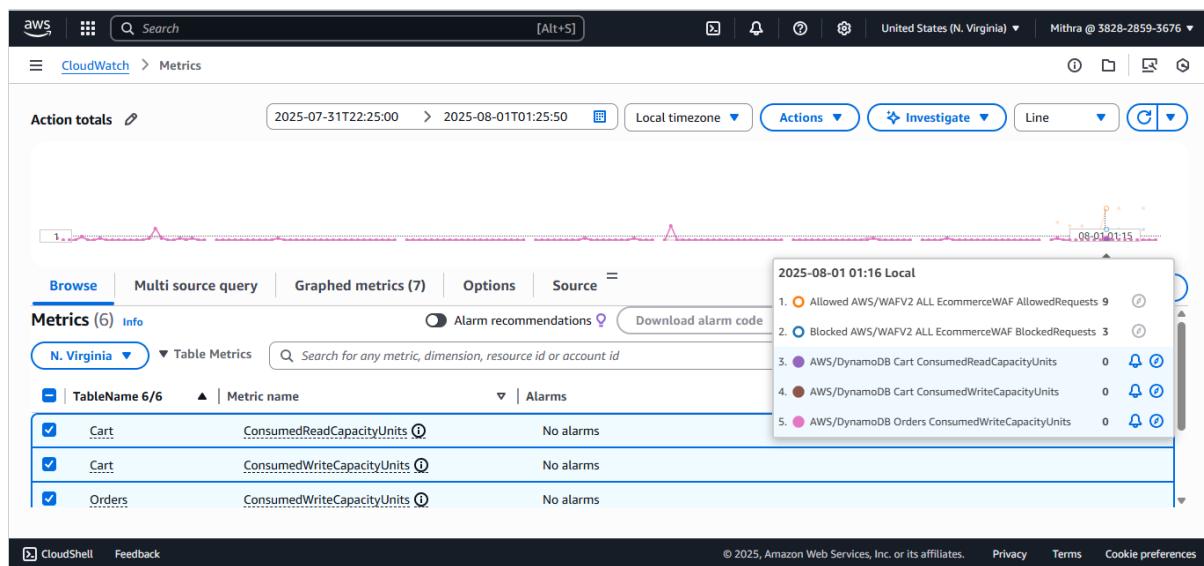
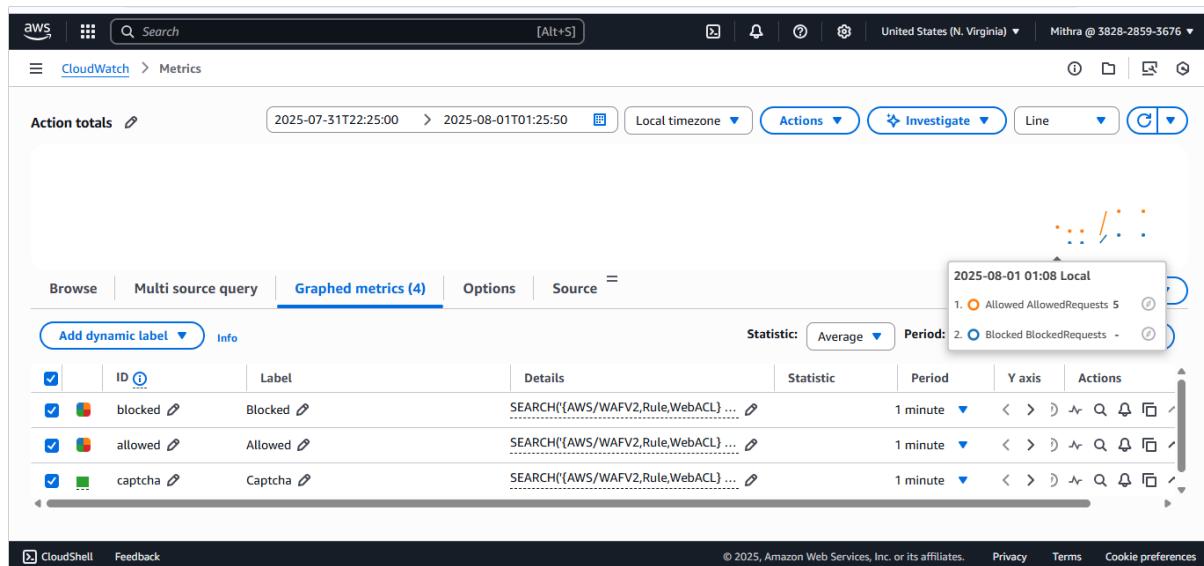
Step 26: Now we can see the traffic us displayed for API

The screenshot shows the AWS WAF & Shield dashboard. The left sidebar is titled "WAF & Shield" and includes sections for "AWS WAF" (Getting started, Web ACLs, Bot control dashboard, Application integration, IP sets, Regex pattern sets, Rule groups, Add-on protections) and "AWS Shield" (Getting started). The main content area is titled "Action totals for the specified time range - All traffic". It displays four categories: Total (5), Blocked (0), Allowed (5), and Challenge (0). Below these, a note explains that the counts are for all possible terminating actions. At the bottom, there are links to "Switch to AWS WAF Classic" and "CloudShell Feedback". The footer contains copyright information and links to Privacy, Terms, and Cookie preferences.









The screenshot shows the AWS WAF & Shield console with the "Web ACLs" page selected. The left sidebar has sections for "AWS WAF" (Getting started, Web ACLs, Bot control dashboard, Application integration, IP sets, Regex pattern sets, Rule groups, Add-on protections) and "AWS Shield" (Getting started). The main content area shows "Web ACLs (1)" with a table listing one item: "ecommerceAPI". The table columns are Name, Description, ARN, and ID. The ARN column shows "arn:aws:wafv2:us-east-1:382828593676:...". The ID column shows "1567d38c-bfdb-4177-8c3c-03752ed3104c". There are buttons for "Create web ACL" and "Delete". A search bar at the top says "Find web ACLs". The top right corner shows "Global" and "Mithra @ 3828-2859-3676".

Name	Description	ARN	ID
ecommerceAPI	-	arn:aws:wafv2:us-east-1:382828593676:...	1567d38c-bfdb-4177-8c3c-03752ed3104c