

## Project:

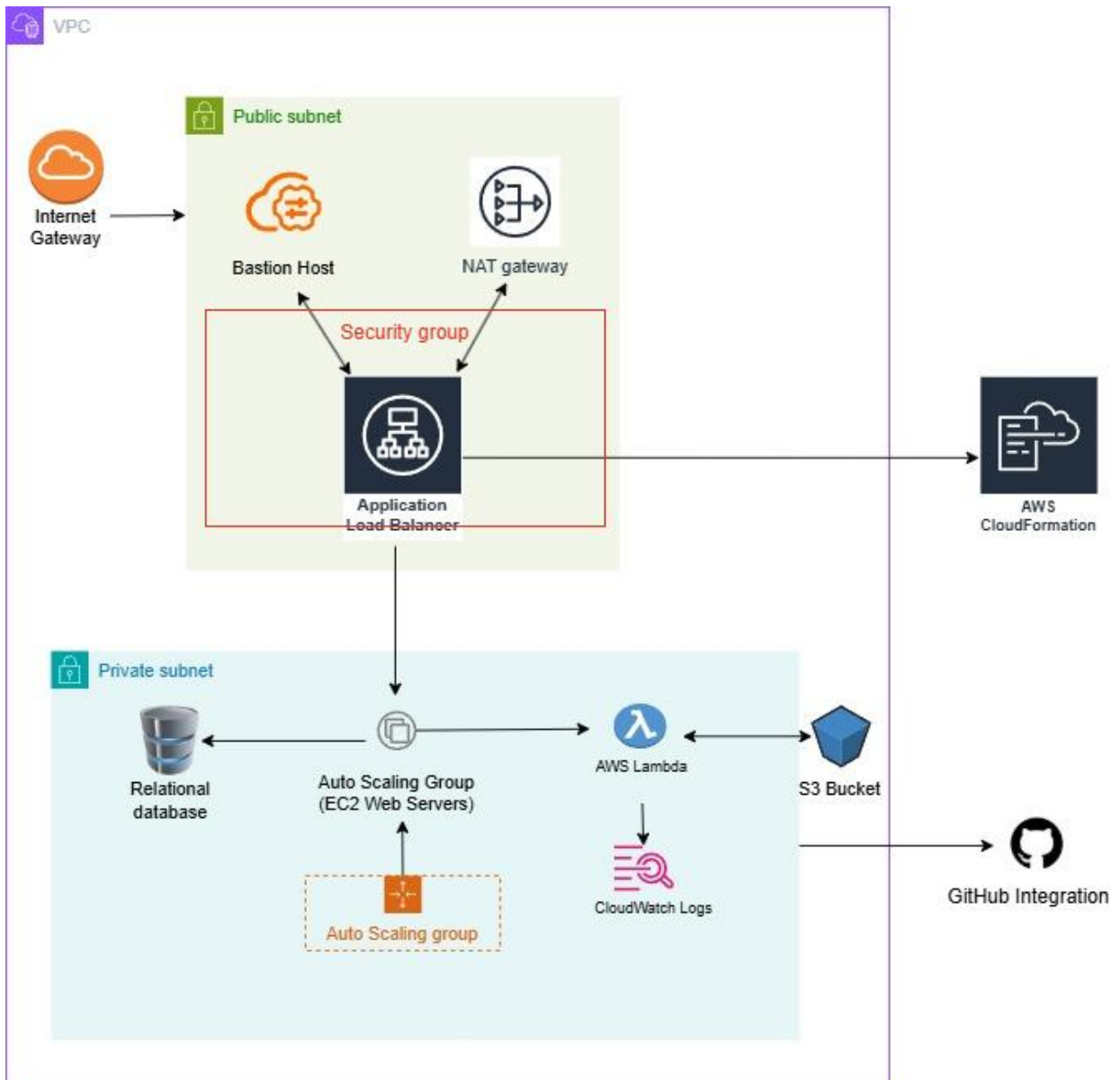
Tartela Tabassum

5/15/2025

### 1. Design an AWS Architecture

Students must design and **draw an architecture diagram** for their cloud application, which should include:

- **AWS VPC** with at least two subnets (public and private).
- **AWS EC2** instances behind an **Application Load Balancer (ALB)**.
- **Relational database** (RDS or MySQL/PostgreSQL on EC2).
- **S3 bucket** for storing files (e.g., logs, backups, static content).
- **AWS Lambda** to log S3 file uploads to **CloudWatch Logs**.
- **Autoscaling** for the web server layer.
- **AWS CloudFormation** to deploy infrastructure.
- **Security Group** configurations to control network access.
- **GitHub integration** for version control.



## Architecture Diagram

The accompanying architecture diagram illustrates a secure, highly available AWS deployment housed within a single VPC that's split into a **public subnet**—which contains an Internet Gateway, NAT Gateway, Bastion Host, and an Application Load Balancer (ALB) guarded by its own ALB-SG—and a **private subnet** hosting an Auto Scaling group of EC2 web servers (protected by Web-SG) alongside an RDS relational database instance (behind DB-SG). All static assets, logs, and backups are stored in an S3 bucket; each **ObjectCreated** event there triggers a Lambda function (in Lambda-SG) that writes detailed entries to CloudWatch Logs. The entire stack is defined as code in AWS CloudFormation templates stored in GitHub, with GitHub Actions automatically validating and deploying updates on every push—ensuring

network isolation, autoscaling, fault tolerance, and end-to-end version-controlled infrastructure provisioning.

## 2. Implementation

### A. Infrastructure Deployment

- Use Terraform to create networking components (VPC, subnets, security groups).

```
Terraform has been successfully initialized!
```

```
You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.
```

```
If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.
```

```
PS C:\Users\User\Documents\terraform-ec2-lab\terraform_1.11.1_windows_amd64> |
```

```
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.
```

```
Enter a value: yes
```

```
aws_vpc.main: Creating...
aws_vpc.main: Creation complete after 1s [id=vpc-00093d985a3289efc]
aws_internet_gateway.gw: Creating...
aws_subnet.public: Creating...
aws_security_group.web_sg: Creating...
aws_internet_gateway.gw: Creation complete after 0s [id=igw-026f2e5cd349a6173]
aws_route_table.public: Creating...
aws_route_table.public: Creation complete after 1s [id=rtb-06e865bc356f20d38]
aws_security_group.web_sg: Creation complete after 3s [id=sg-024946b702581d6e5]
aws_subnet.public: Still creating... [10s elapsed]
aws_subnet.public: Creation complete after 11s [id=subnet-02985573b62ab9966]
aws_route_table_association.public_assoc: Creating...
aws_route_table_association.public_assoc: Creation complete after 0s [id=rtbassoc-0d8609ce18256e58d]
```

```
Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
```

```
Outputs:
```

```
public_subnet_id = "subnet-02985573b62ab9966"
security_group_id = "sg-024946b702581d6e5"
vpc_id = "vpc-00093d985a3289efc"
```

```
PS C:\Users\User\Documents\terraform-ec2-lab\terraform_1.11.1_windows_amd64> |
```

- Use CloudFormation to deploy EC2 instances, RDS, and Lambda functions.

Stacks (1)

Filter status

Filter by stack name

Active

View nested

Stacks

MyEC2Stack

2025-05-15 14:32:14 UTC-0400

CREATE\_COMPLETE

Stack info

Events

Resources

Outputs

Parameters

Template

Change sets

Git sync

Table view

Timeline view

Events (18)

View root cause

Search events

Timestamp	Logical ID	Status	Detailed status	Status reason
2025-05-15 14:38:01 UTC-0400	<a href="#">MyEC2Stack</a>	CREATE_COMPLETE	-	-
2025-05-15 14:38:00 UTC-0400	<a href="#">MyDB</a>	CREATE_COMPLETE	-	-
2025-05-15 14:32:31 UTC-0400	<a href="#">EC2Instance</a>	CREATE_COMPLETE	-	-
2025-05-15 14:32:25 UTC-0400	<a href="#">MyLambda</a>	CREATE_COMPLETE	-	-
2025-05-15 14:32:25 UTC-0400	<a href="#">WebSecurityGroup</a>	CREATE_COMPLETE	-	-

```
Version 2023.7.20250512:  
Run "/usr/bin/dnf check-release-update" for full release and version update info  
  
#####  
~\_____ Amazon Linux 2023  
~~~\_#####\  
~~~~_\###|  
~~~~_\#/ https://aws.amazon.com/linux/amazon-linux-2023  
~~~~V~' ~->  
~~~~_  
~~~~_. .  
~~~~_/ _/  
~~~~/_m/' _/  
[ec2-user@ip-10-0-1-148 ~]$ |
```

← → ↻ ⚠ Not secure 18.208.165.243

```
[ec2-user@ip-10-0-1-148 ~]$ sudo mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.5.23-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE myapp;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> CREATE USER 'myuser'@'localhost' IDENTIFIED BY 'mypassword';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON myapp.* TO 'myuser'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> EXIT;
Bye
[ec2-user@ip-10-0-1-148 ~]$ |
```

- **Implement autoscaling** to manage web server load.

Success
Successfully created [web-server-template\(lt-0ce4e367e76a56755\)](#).

► Actions log

**Next Steps**

**Launch an instance**

With On-Demand Instances, you pay for compute capacity by the second (for Linux, with a minimum of 60 seconds) or by the hour (for all other operating systems) with no long-term commitments or upfront payments. Launch an On-Demand Instance from your launch template.

[Launch instance from this template](#)

**Create an Auto Scaling group from your template**

Amazon EC2 Auto Scaling helps you maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define. You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs.

[Create Auto Scaling group](#)

**Create Spot Fleet**

A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price. Because Spot Instances enable you to request unused EC2 instances at steep discounts, you can lower your Amazon EC2 costs significantly. The hourly price for a Spot Instance (of each instance type in each Availability Zone) is set by Amazon EC2, and adjusted gradually based on the long-term supply of and demand for Spot Instances. Spot instances are well-suited for data-analysis, batch jobs, background processing, and optional tasks.

[Create Spot Fleet](#)

[View launch templates](#)

web-asg, 1 Scaling policy created successfully
×

**Auto Scaling groups (1)** [Info](#)

Last updated less than a minute ago

[Launch configurations](#)
[Launch templates](#)
[Actions](#)
[Create Auto Scaling group](#)

<input type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
<input type="checkbox"/>	<a href="#">web-asg</a>	<a href="#">web-server-template</a>   Version Default	0	Updating capacity...	1	1	3	us-east-1a, us-east-1b

## B. AWS Lambda for Logging S3 Uploads

- **Create an AWS Lambda function** that logs S3 file uploads to CloudWatch Logs.
- **Use the following Python script for the Lambda function:**

```
import json
import boto3
import logging

# Set up logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        # Extract bucket name and object key from the event
        bucket_name = event['Records'][0]['s3']['bucket']['name']
        object_key = event['Records'][0]['s3']['object']['key']

        # Log the upload event
        log_message = f"New file uploaded: {object_key} in bucket: {bucket_name}"
        logger.info(log_message)

        return {
            "statusCode": 200,
            "body": json.dumps(log_message)
        }

    except Exception as e:
        logger.error(f"Error processing S3 event: {str(e)}")
        return {
            "statusCode": 500,
            "body": json.dumps(f"Error: {str(e)}")
        }
```

- **Set up an S3 event trigger** so the function executes whenever a new file is uploaded.

aws [Search] [Alt+5] United States (N. Virginia) 0211 (9413-7714-5211) root

Lambda > Functions > LogS3Uploads

**Lambda**

- Dashboard
- Applications
- Functions
  - LogS3Uploads
- ▼ Additional resources
  - Code signing configurations
  - Event source mappings
  - Layers
  - Replicas
- ▼ Related AWS resources
  - Step Functions state machines

**LogS3Uploads** [Throttle] [Copy ARN] [Actions]

✓ The trigger cf-templates-1w7tz4r8l4mu3-us-east-1 was successfully added to function LogS3Uploads. The function is now receiving events from the trigger.

▼ **Function overview** [Info]

[Diagram] [Template]

**LogS3Uploads**

Layers (0)

S3

+ Add trigger

+ Add destination

Export to Infrastructure Composer Download

**Description**

Last modified 7 minutes ago

Function ARN arn:aws:lambda:us-east-1:94137714521:1:function:LogS3Uploads

Function URL [Info]

Code Test Monitor **Configuration** Aliases Versions

General configuration

Triggers (1) [Info]

Fix errors Edit Delete Add trigger

**Function configuration**

Use the function overview to see your function's configuration. You can choose to view this as a diagram or as an AWS Serverless Application Model (AWS SAM) template. AWS SAM is an open source framework for building serverless applications using infrastructure as code (IaC). You can see the following types of resources in the diagram or template:

**Triggers** are AWS services or resources that invoke the function.

**Destinations** are AWS resources that receive a record of an invocation after success or failure. You can configure Lambda to send invocation records when your function is invoked asynchronously, or if your function processes records from a stream. The contents of the invocation record and supported

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

aws [Search] [Alt+5] United States (N. Virginia) 0211 (9413-7714-5211) root

Upload succeeded

For more information, see the Files and folders table.

**Upload: status** [Close]

After you navigate away from this page, the following information is no longer available.

**Summary**

Destination s3://cf-templates-1w7tz4r8l4mu3-us-east-1

Succeeded 1 file, 16.0 B (100.00%)

Failed 0 files, 0 B (0%)

**Files and folders** [Configuration]

Files and folders (1 total, 16.0 B)

Find by name

Name	Folder	Type	Size	Status	Error
test.txt	-	text/plain	16.0 B	Succeeded	-

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

- **Verify logs in CloudWatch** to ensure it captures the uploaded file's name and bucket details.

The screenshot shows the AWS CloudWatch console. The left sidebar contains navigation links for CloudWatch, Favorites and recents, Dashboards, AI Operations, Alarms, Logs, Metrics, X-Ray traces, Events, Application Signals, and Network Monitoring. The main content area is titled 'Log events' and shows a list of log events for a Lambda function. The events include INIT\_START, START, [INFO], END, and REPORT messages. The console also features a search bar, filter options, and a 'Display' button.

Timestamp	Message
2025-05-16T01:45:18.951Z	INIT_START Runtime Version: python:3.13.v40 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:67df0ac2272a9d5069c1c8b0b34c80bdf11f9738985ac6f79319e3e026...
2025-05-16T01:45:19.261Z	START RequestId: 8e11f639-1bcb-4a6d-a9af-5dca24a49459 Version: \$LATEST
2025-05-16T01:45:19.261Z	[INFO] 2025-05-16T01:45:19.261Z 8e11f639-1bcb-4a6d-a9af-5dca24a49459 New file uploaded: test.txt in bucket: cf-templates-1u7t24r8l4mu3-us-east-1
2025-05-16T01:45:19.263Z	END RequestId: 8e11f639-1bcb-4a6d-a9af-5dca24a49459
2025-05-16T01:45:19.263Z	REPORT RequestId: 8e11f639-1bcb-4a6d-a9af-5dca24a49459 Duration: 2.09 ms Billed Duration: 3 ms Memory Size: 128 MB Max Memory Used: 63 MB Init Duration: 306.08...

## C. Interaction with AWS

- Use AWS Console to verify infrastructure deployment.

The screenshot shows the AWS EC2 console. The left sidebar contains navigation links for EC2, Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, Elastic Block Store, and Network & Security. The main content area is titled 'Instances (3)' and shows a list of running instances. The instances are sorted by state (running) and include details such as Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, and Public IPv4.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
i-007e4d23cc41e09b2	i-007e4d23cc41e09b2	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-5-82-227-13.comp...	3.82.227.13
i-0d45b3b5bcb1beedf	i-0d45b3b5bcb1beedf	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	-	18.208.165.
i-04ab0a29a07d5f3e7	i-04ab0a29a07d5f3e7	Running	m1.small	2/2 checks passed	View alarms +	us-east-1b	ec2-54-234-119-12.co...	54.234.119.



Amazon S3

General purpose buckets

Directory buckets

Table buckets

Access Grants

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

Storage Lens groups

AWS Organizations settings

Feature spotlight 11

AWS Marketplace for S3

Account snapshot - updated every 24 hours

All AWS Regions

View Storage Lens dashboard

General purpose buckets

Directory buckets

General purpose buckets (1)

Info

All AWS Regions

Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3.

Find buckets by name

< 1 >

Name	AWS Region	IAM Access Analyzer	Creation date
cf-templates-1w7tz4r8l4mu3-us-east-1	US East (N. Virginia) us-east-1	<a href="#">View analyzer for us-east-1</a>	May 15, 2025, 12:18:58 (UTC-04:00)

Lambda > Functions > LogS3Uploads

LogS3Uploads

EXPLORER

LOGS3UPLOADS

lambda\_function.py

DEPLOY

Deploy (Ctrl+Shift+U)

Test (Ctrl+Shift+T)

TEST EVENTS [NONE SELECTED]

Create new test event

ENVIRONMENT VARIABLES

Amazon Q

lambda\_function.py

```
1 import json
2 import boto3
3 import logging
4
5 # Set up logging
6 logger = logging.getLogger()
7 logger.setLevel(logging.INFO)
8
9 def lambda_handler(event, context):
10     try:
11         # Extract bucket name and object key from the event
12         bucket_name = event['Records'][0]['s3']['bucket']['name']
13         object_key = event['Records'][0]['s3']['object']['key']
14
15         # Log the upload event
16         log_message = f"New file uploaded: {object_key} in bucket: {bucket_name}"
17         logger.info(log_message)
18
19         return {
20             "statusCode": 200,
21             "body": json.dumps(log_message)
22         }
23
24     except Exception as e:
25         logger.error(f"Error processing S3 event: {str(e)}")
26         return {
27             "statusCode": 500,
28             "body": json.dumps(f"Error: {str(e)}")
29         }
```

No changes to deploy

Info

Tutorials

Learn how to implement common use cases in AWS Lambda.

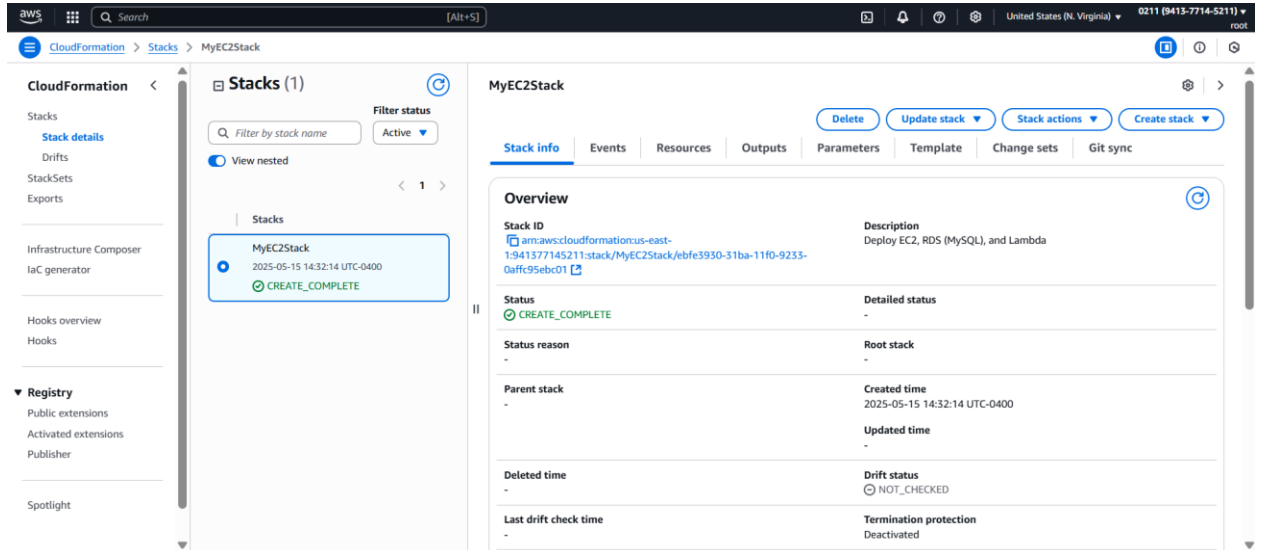
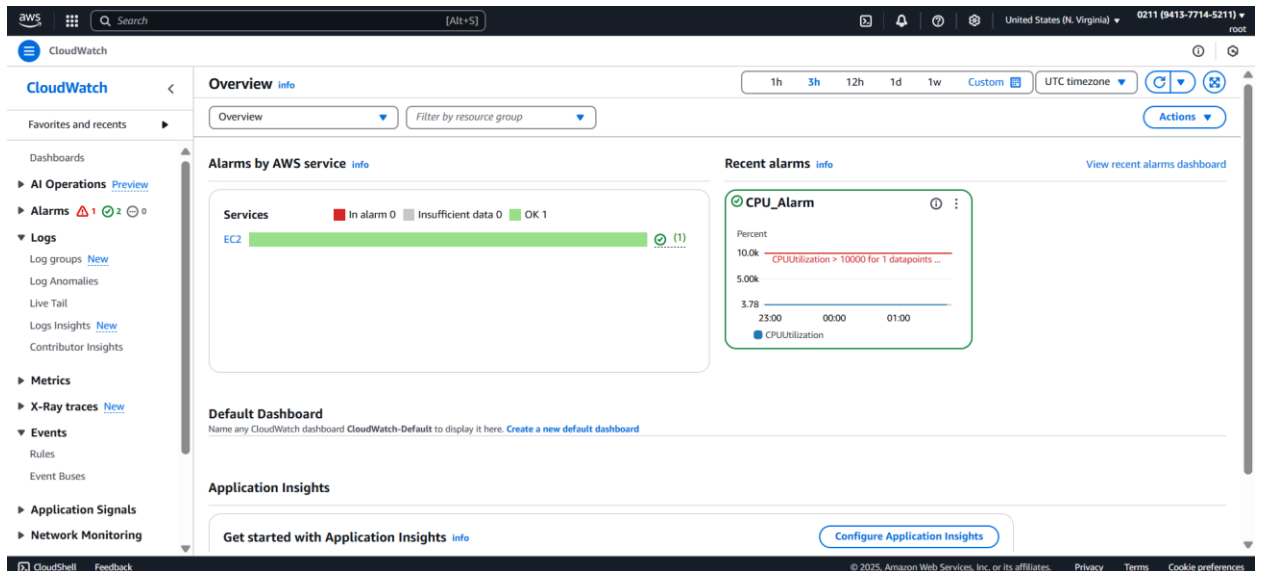
Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Learn more

Start tutorial



- Use AWS CLI to interact with EC2, S3, and Lambda.

```
User@DESKTOP-AQNR9SO MINGW64 ~  
$ aws ec2 describe-instances \  
  --instance-ids i-0d45b3b5bcb1beedf \  
  --query 'Reservations[0].Instances[0].PublicIpAddress' \  
  --output text  
18.208.165.243
```

```
User@DESKTOP-AQNR9SO MINGW64 ~  
$ aws s3 ls  
2025-05-15 12:18:58 cf-templates-1w7tz4r8l4mu3-us-east-1
```

```
User@DESKTOP-AQNR9SO MINGW64 ~
$ aws lambda invoke \
  --function-name LogS3Uploads \
  --payload '{}' \
  output.json
{
  "StatusCode": 200,
  "ExecutedVersion": "$LATEST"
}
```

- Write Python scripts using Boto3 to:
  - Create an S3 bucket and upload a file.

```
PS C:\Users\User\Downloads> python create_s3_and_upload.py
File test.txt uploaded to bucket tartela-bucket
PS C:\Users\User\Downloads> |
```

- Retrieve EC2 instance metadata.

```
[ec2-user@ip-10-0-1-148 ~]$ ls
ec2.py  metadata.py
[ec2-user@ip-10-0-1-148 ~]$ nano metadata.py
[ec2-user@ip-10-0-1-148 ~]$ python3 metadata.py
i-0d45b3b5bcb1beedf:
t3.micro:
18.208.165.243:
ami-000d41719f4ab3d23:
[ec2-user@ip-10-0-1-148 ~]$ |
```

- List running EC2 instances.

```
PS C:\Users\User\Downloads> python list_running_ec2.py
Instance ID: i-007e4d23cc41e09b2, Public IP: 3.82.227.13
Instance ID: i-0d45b3b5bcb1beedf, Public IP: 18.208.165.243
Instance ID: i-04ab0a29a07d5f3e7, Public IP: 54.234.119.12
PS C:\Users\User\Downloads> |
```

- Invoke the AWS Lambda function manually.

```
PS C:\Users\User\Downloads> python invoke_lambda.py  
{"statusCode": 500, "body": "\"Error: 'Records'\""}  
PS C:\Users\User\Downloads> |
```