

# IS698 - DC93457

## Project: Deploying a Scalable AWS Architecture with Infrastructure as Code

### A. Infrastructure Deployment

#### Terraform

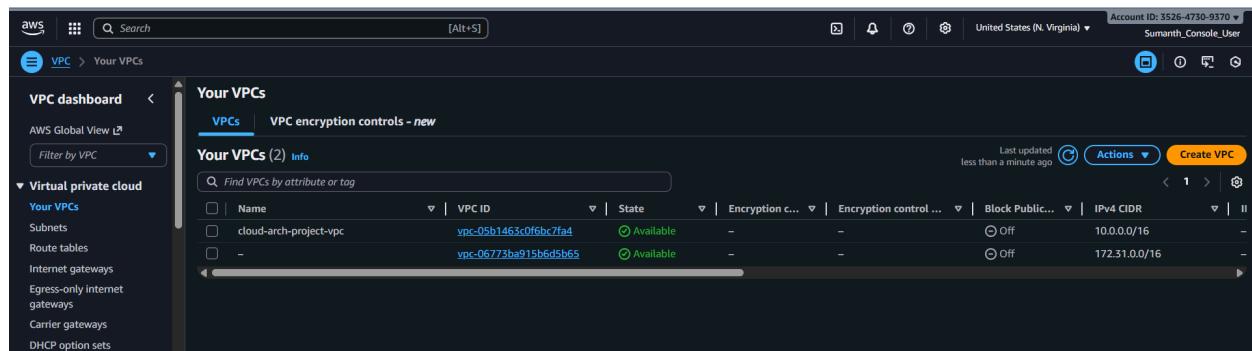
```
Command Prompt
aws_subnet_public["1"] Still creating... [00m10s elapsed]
aws_subnet_public["1"]: creation complete after 11s [id=subnet-05456803548a6f632]
aws_subnet_public["0"] Still creating... [00m10s elapsed]
aws_subnet_public["0"]: creation complete after 11s [id=subnet-03c59a11ad0897b9a]
aws_route_table_association_public["1"] Creating...
aws_route_table_association_public["0"] Creating...
aws_nat_gateway.nat: Creating...
aws_route_table_association_public["1"] Creating complete after 1s [id=rtbassoc-058bcc2338a9747b]
aws_route_table_association_public["0"] Creating complete after 1s [id=rtbassoc-09c11ede44664cfb8]
aws_nat_gateway.nat: Still creating... [00m10s elapsed]
aws_nat_gateway.nat: Still creating... [00m20s elapsed]
aws_nat_gateway.nat: Still creating... [00m30s elapsed]
aws_nat_gateway.nat: Still creating... [00m40s elapsed]
aws_nat_gateway.nat: Still creating... [00m50s elapsed]
aws_nat_gateway.nat: Still creating... [01m00s elapsed]
aws_nat_gateway.nat: Still creating... [01m10s elapsed]
aws_nat_gateway.nat: Still creating... [01m20s elapsed]
aws_nat_gateway.nat: Still creating... [01m30s elapsed]
aws_nat_gateway.nat: Still creating... [01m40s elapsed]
aws_nat_gateway.nat: Still creating... [01m50s elapsed]
aws_nat_gateway.nat: Creation complete after 1m55s [id=nat-0d7f43cbc7b707b00]
aws_route_table.private: Creating...
aws_route_table.private: Creation complete after 1s [id=rtb-0fc558fb0e26c4c4]
aws_route_table_association_private["0"] Creating...
aws_route_table_association_private["1"] Creating...
aws_route_table_association_private["1"] Creation complete after 0s [id=rtbassoc-030b205d5349609f4]
aws_route_table_association_private["0"] Creation complete after 0s [id=rtbassoc-0a8a51feffclcde8b]

Apply complete! Resources: 17 added, 0 changed, 0 destroyed.

Outputs:

alb_sg_id = "sg-0d06f1f81d98aa318"
private_subnet_ids = [
  "subnet-030b5cdca9c21a31a",
  "subnet-04118cf5a7fa934c8",
]
public_subnet_ids = [
  "subnet-03c59a11ad0897b9a",
  "subnet-05456803548a6f632",
]
rds_sg_id = "sg-02b2a6e086eddb1a8"
vpc_id = "vpc-05b1463c0f6bc7fa4"
web_sg_id = "sg-03016daa5a2bd32c3"

C:\Users\acer\terraform-698>
```



The screenshots show the AWS VPC dashboard interface across four different sections:

- Subnets (10) Info**: Displays 10 subnets with details like Name, Subnet ID, State, VPC, Block Public, IPv4 CIDR, and IPv6 CIDR.
- Route tables (4) Info**: Displays 4 route tables with details like Name, Route table ID, Explicit subnet associations, Edge associations, Main, VPC, and Owner ID.
- Internet gateways (2) Info**: Displays 2 internet gateways with details like Name, Internet gateway ID, State, VPC ID, and Owner.
- Security Groups (5) Info**: Displays 5 security groups with details like Name, Security group ID, Security group name, VPC ID, and Description.

In this section, I used Terraform to build the core networking layer for the project. Terraform automatically created my VPC, subnets, route tables and security groups exactly the way I defined them in the code. Instead of manually creating networking resources, Terraform handled all of it in one shot.

## Cloudformation

Stack name: rds698

**Parameters**

- DBName**: Name of the initial database to create: cloudprojectdb
- DBPassword**: Master password (at least 8 characters): \*\*\*\*\*
- DBUsername**: Master username (1-16 chars, letters/numbers, must start with a letter): \*\*\*\*
- PrivateSubnet1Id**: First private subnet ID: subnet-030b5cde9c21a314
- PrivateSubnet2Id**: Second private subnet ID: subnet-04118cf5a7fa934c8
- RdsSecurityGroupId**: RDS security group ID (must belong to the same VPC): sg-02b2a6e086eddb1a8
- VpcId**: VPC ID from Terraform output vpc\_id

**Summary**

<b>DB identifier</b> rds698-mydbinstance-qfjghsjhrkc	<b>Status</b> <span style="color: green;">Available</span>	<b>Role</b> Instance	<b>Engine</b> MySQL Community
<b>CPU</b> <div style="width: 3.90%;">3.90%</div>	<b>Class</b> db.t3.micro	<b>Current activity</b> <div style="width: 0%;">0 Connections</div>	<b>Region &amp; AZ</b> us-east-1a

**Connectivity & security**

<b>Endpoint &amp; port</b> Endpoint: rds698-mydbinstance-qfjghsjhrkc.cmtoeaqe4zjy.us-east-1.rds.amazonaws.com Port: 3306	<b>Networking</b> Availability Zone: us-east-1a VPC: Cloud-arch-project-vpc (vpc-05b1463c0f6bc7fa4) Subnet group: rds698-bsubnetgroup-vvdz2uyvtgk Subnets: subnet-04118cf5a7fa934c8, subnet-030b5cde9c21a314 Network type: IPv4	<b>Security</b> VPC security groups: cloud-arch-project-rds-sg (sg-02b2a6e086eddb1a8) (Active) Publicly accessible: No Certificate authority: rds-ca-rsa2048-g1 Certificate authority date: May 25, 2061, 19:34 (UTC-04:00) DB instance certificate expiration date: December 06, 2026, 17:46 (UTC-05:00)
--	--	--

**CloudFormation > Stacks > Create stack**

Step 3: Configure stack options  
Step 4: Review and create

**Stack name:** ec2698  
Stack name must contain only letters (a-z, A-Z), numbers (0-9), and hyphens (-) and start with a letter. Max 128 characters. Character count: 6/128.

**Parameters:**

- ALBSecurityGroupId:** ALB security group ID  
sg-0d06f1f81d98aa318
- DBEndpointAddress:** RDS database endpoint address (from RDS stack output)  
rds:698-mydbinstance-qfjghsjrhk.cmtbeaqej4zy.us-east-1.rds.amazonaws.com
- DBName:** Database name created in RDS stack (for example cloudprojectdb)  
rds:698-mydbinstance-qfjghsjrhk
- DBPassword:** RDS master password (same as RDS stack)  
\*\*\*\*\*
- DBUsername:** RDS master username (same as RDS stack)  
\*\*\*\*\*
- PrivateSubnet1Id:** First private subnet ID  
subnet-030b5cde49c21a314
- PrivateSubnet2Id:** ...

**EC2 > Instances > i-04674943ff91fc96b (simple-ec2-public)**

**Instance summary for i-04674943ff91fc96b (simple-ec2-public)**

Updated less than a minute ago

Attribute	Value
Instance ID	i-04674943ff91fc96b
IPv6 address	-
Hostname type	IP name: ip-10-0-1-65.ec2.internal
Answer private resource DNS name	-
Auto-assigned IP address	44.199.236.168 [Public IP]
IAM Role	-
IMDv2	Required
Operator	-
Public IPv4 address	44.199.236.168 [open address]
Instance state	Running
Private IP DNS name (IPv4 only)	ip-10-0-1-65.ec2.internal
Instance type	t3.micro
VPC ID	vpc-05b1463c0f6bc7fa4 (cloud-arch-project-vpc)
Subnet ID	subnet-03c59a11ad0897b9a (cloud-arch-project-public-0)
Instance ARN	arn:aws:ec2:us-east-1:352647309370:instance/i-04674943ff91fc96b
Private IPv4 addresses	10.0.1.65
Public DNS	ec2-44-199-236-168.compute-1.amazonaws.com [open address]
Elastic IP addresses	-
AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations.   Learn more
Auto Scaling Group name	-
Managed	false

**Details** | **Status and alarms** | **Monitoring** | **Security** | **Networking** | **Storage** | **Tags**

**Instance details**

Attribute	Value
AMI ID	ami-0fa3fe0fa7920f68e
AMI name	al2023-ami-2023.9.20251117.1-kernel-6.1-x86_64
Stop protection	Disabled
Launch time	Sat Dec 06 2025 18:19:43 GMT-0500 (Eastern Standard Time) (3 minutes)

**Platform details**

- Linux/UNIX

**Termination protection**

- Disabled

**AMI location**

- amazon/al2023-ami-2023.9.20251117.1-kernel-6.1-x86\_64

Not secure 44.199.236.168

UMBC

## Hello from a simple EC2 instance!

This instance was created using a basic CloudFormation template in a public subnet.

This part covers the CloudFormation stacks I deployed for EC2 and RDS. Each YAML file defined a specific resource. CloudFormation read the template and created the servers for me. Also it created a static web page for the EC2 instance.

## Configure database with application

```
ec2-user@ip-10-0-1-65:~$ login as: ec2-user
Authenticating with public key "698finalproject"
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-10-0-1-65 ~]$ sudo yum update -y
Last metadata expiration check: 0:27:03 ago on Sat Dec 6 23:20:12 2025.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-10-0-1-65 ~]$ sudo yum install -y httpd php php-mysqld
Last metadata expiration check: 0:27:42 ago on Sat Dec 6 23:20:12 2025.
Package httpd-2.4.65-1.amzn2023.0.2.x86_64 is already installed.
Dependencies resolved.
=====
Package          Arch    Version           Repository      Size
=====
Installing:
php8.4           x86_64  8.4.14-1.amzn2023.0.1   amazonlinux      16 k
php8.4-mysqld   x86_64  8.4.14-1.amzn2023.0.1   amazonlinux      155 k
Installing dependencies:
libodium         x86_64  1.0.19-4.amzn2023            amazonlinux     176 k
libxslt          x86_64  1.1.43-1.amzn2023.0.3    amazonlinux     183 k
nginx-filesystem noarch  1.1.28-0.1.amzn2023.0.2    amazonlinux     9.6 k
php8.4-cli       x86_64  8.4.14-1.amzn2023.0.1    amazonlinux     3.8 M
php8.4-common   x86_64  8.4.14-1.amzn2023.0.1    amazonlinux     802 k
php8.4-pdo      x86_64  8.4.14-1.amzn2023.0.1    amazonlinux     99 k
php8.4-process  x86_64  8.4.14-1.amzn2023.0.1    amazonlinux     51 k
php8.4-xml      x86_64  8.4.14-1.amzn2023.0.1    amazonlinux     675 k
Installing weak dependencies:
php8.4-fpm       x86_64  8.4.14-1.amzn2023.0.1    amazonlinux     2.0 M
php8.4-mbstring  x86_64  8.4.14-1.amzn2023.0.1    amazonlinux     540 k
php8.4-ocache   x86_64  8.4.14-1.amzn2023.0.1    amazonlinux     499 k
php8.4-sodium   x86_64  8.4.14-1.amzn2023.0.1    amazonlinux     47 k
Transaction Summary
=====
Install 14 Packages

Total download size: 5.0 M
Installed size: 41 M

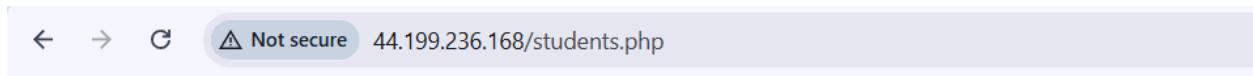
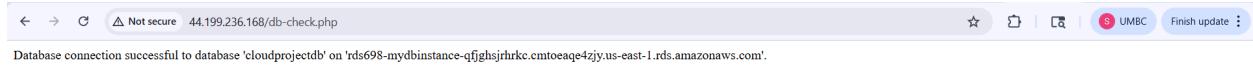
ec2-user@ip-10-0-1-65:~$ db user = 'admin';
$db_pass = 'admin1234';

$conn = new mysqli($db_host, $db_user, $db_pass, $db_name);

if ($conn->connect_error) {
    echo "Database connection failed: " . $conn->connect_error;
} else {
    echo "Database connection successful to database '$db_name' on '$db_host'.";
}

$conn->close();
?>

[ec2-user@ip-10-0-1-65 ~]$ cat /etc/os-release
NAME="Amazon Linux"
VERSION="2023"
ID="amzn"
ID_LIKE="fedora"
VERSION_ID="2023"
PLATFORM_ID="platform:el2023"
PRETTY_NAME="Amazon Linux 2023.9.20251117"
ANSI_COLOR="#339933"
CPE_NAME="cpe:2.3:amazon:amazon_linux:2023"
HOME_URL="https://aws.amazon.com/amazon-linux-2023/"
DOCUMENTATION_URL="https://docs.aws.amazon.com/linux/"
SUPPORT_URL="https://aws.amazon.com/premiumsupport/"
BUG_REPORT_URL="https://github.com/amazonlinux/amazon-linux-2023"
VENDOR_NAME="AWS"
VENDOR_URL="https://aws.amazon.com/"
SUPPORT_END="2029-06-30"
[ec2-user@ip-10-0-1-65 ~]$ sudo dnf update -y
Last metadata expiration check: 0:42:38 ago on Sat Dec 6 23:20:12 2025.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-10-0-1-65 ~]$ sudo dnf install -y httpd php php-mysqld
Last metadata expiration check: 0:42:38 ago on Sat Dec 6 23:20:12 2025.
Package httpd-2.4.65-1.amzn2023.0.2.x86_64 is already installed.
Package php8.4-8.4.14-1.amzn2023.0.1.x86_64 is already installed.
Package php8.4-mysqld-8.4.14-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-10-0-1-65 ~]$ sudo systemctl enable httpd
[ec2-user@ip-10-0-1-65 ~]$ sudo systemctl restart httpd
[ec2-user@ip-10-0-1-65 ~]$ sudo nano /var/www/html/db-check.php
[ec2-user@ip-10-0-1-65 ~]$ sudo systemctl restart httpd
[ec2-user@ip-10-0-1-65 ~]$
```



ID	Name	Major	Year
1	Ken Adams	Computer Science	2
2	Sumanth Reddy	Information Systems	3

In this step I connected my EC2 web server to the RDS database. I wrote a small simple PHP script (db-check.php) using SSH and uploaded it to /var/www/html/. The script tries to connect to RDS using the endpoint, username, and password. Testing this in the browser showed “Database connection successful” which confirms that EC2 and RDS databases are connected.

After the successful connection using MySql in SSH, I created a table ‘Students’ in database and inserted student values. I created a new php script called ‘students.php’ where it will connect the database to the dynamic web page. When opened it with EC2 instance then, I see 2 student records in the web page.

## Autoscaling

Screenshot of the AWS CloudFormation console showing the creation of a new stack named "ec2autoscaling". The stack template includes parameters for DB endpoint address (rds698-mydbinstance-qfjghsjhrk.cmtoeaqe42jy.us-east-1.rds.amazonaws.com), database name (cloudprojectdb), master password (\*\*\*\*\*), master username (\*\*\*\*), public subnet IDs (subnet-05c59a11ad0897b9a and subnet-05456803548a6f632), and VPC ID (vpcId).

The second screenshot shows the "ec2autoscaling" stack details page. The "Events" tab is selected, displaying 14 events related to the stack's creation. Key events include:

Operation ID	Timestamp	Logical ID	Status	Detailed status	Status reason
25060f18-f0de-4c73-9077-f41d75a7e62c	2025-12-06 19:34:27 UTC-0500	ec2autoscaling	CREATE_COMPLETE	-	-
25060f18-f0de-4c73-9077-f41d75a7e62c	2025-12-06 19:34:27 UTC-0500	CpuTargetTrackingPolicy	CREATE_COMPLETE	-	-
25060f18-f0de-4c73-9077-f41d75a7e62c	2025-12-06 19:34:27 UTC-0500	WebAutoScalingGroup	CREATE_COMPLETE	-	-
25060f18-f0de-4c73-9077-f41d75a7e62c	2025-12-06 19:34:20 UTC-0500	ec2autoscaling	CREATE_IN_PROGRESS	CONFIGURATION_COMPLETE	Eventual consistency check initiated

The screenshot shows the AWS EC2 Auto Scaling groups page. The left sidebar includes links for EC2, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, and Elastic Block Store (Volumes, Snapshots, Lifecycle Manager). The main content area displays a table for Auto Scaling groups, with one entry visible:

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones	Creation time
<a href="#">ec2autoscaling-WebAutoScalingGroup-LKmwhTCmOBP</a>	<a href="#">ec2autoscaling-web-It</a>   Version 1	1	-	1	1	3	2 Availability Zones	Sat Dec 06 2025 19...

Header elements include a search bar, account information (Account ID: 3526-4730-9370), and navigation buttons.

The screenshot shows the AWS EC2 console with the following navigation path: AWS Home > EC2 > Launch templates > ec2autoscaling-web-lt.

**Launch template details:**

- Launch template ID: lt-0c711791cb6ecdb58
- Launch template name: ec2autoscaling-web-lt
- Default version: v1
- Owner: arn:aws:iam::352647309370:user/Sumanth\_Console\_User

**Launch template version details:**

Version	Description	Date created	Created by
v1 (Default)	-	2025-12-07T00:34:05.000Z	arn:aws:iam::352647309370:user/Sumanth_Console_User

**Instance details:**

AMI ID	Instance type	Availability Zone	Availability Zone Id
ami-0fa3fe0fa7920f68e	t5.micro	-	-

**Key pair name:** 698finalproject

**Security groups:** -

**Security group IDs:** sg-03016daa5a2b032c3

The screenshot shows the AWS EC2 Instances page for an instance named 'i-0904f17c1ea3561b0'. The left sidebar includes links for Dashboard, EC2 Global View, Events, Instances (selected), Answer Requests, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager (New), Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots), Lifecycle Manager, Network & Security (Security Groups, Elastic IPs, Placement Groups), and CloudShell, Feedback, Console Mobile App.

**Instance summary for i-0904f17c1ea3561b0 (asg-web-instance)** Info

Updated less than a minute ago

**Instance ID**: i-0904f17c1ea3561b0

**IPv6 address**

**Hostname type**: IP name: ip-10-0-1-231.ec2.internal

**Answer private resource DNS name**

**Auto-assigned IP address**: 44.204.186.241 [Public IP]

**IAM Role**: -

**IMDv2**: Required

**Operator**: -

**Public IPv4 address**: 44.204.186.241 [open address]

**Private IP DNS name (IPv4 only)**: ip-10-0-1-231.ec2.internal

**Instance state**: Running

**VPC ID**: vpc-05b1463c0f6bc7fa4 (cloud-arch-project-vpc)

**Subnet ID**: subnet-03c59a11ad0897b9a (cloud-arch-project-public-0)

**Instance ARN**: arn:aws:ec2:us-east-1:352647309570:instance/i-0904f17c1ea3561b0

**Private IPv4 addresses**: 10.0.1.231

**Public DNS**: ec2-44-204-186-241.compute-1.amazonaws.com [open address]

**Elastic IP addresses**

**AWS Compute Optimizer finding**: Opt-in to AWS Compute Optimizer for recommendations. | Learn more

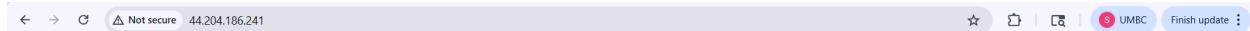
**Auto Scaling Group name**: ec2AutoScaling-WebAppAutoScalingGroup-LKmwhTCMmOBP

**Managed**: false

**Details** **Status and alarms** **Monitoring** **Security** **Networking** **Storage** **Tags**

**Instance details** Info

AMI ID	Monitoring	Platform details
ami-0fa3fe0fa7920f68e	disabled	Linux/UNIX
AMI name	Allowed image	Termination protection
al2023-ami-2023.9.20251117.1-kernel-6.1-x86_64	-	Disabled
Stop protection	Launch time	AMI location
Disabled	Sat Dec 06 2025 19:34:15 GMT-0500 (Eastern Standard Time) (4 minutes)	amazon/al2023-ami-2023.9.20251117.1-kernel-6.1-x86_64



## Hello from an Auto Scaling EC2 instance!

This instance was launched by an Auto Scaling Group.

To test the database connection, open /db-check.php.

```
ec2-user@ip-10-0-1-231:~$ 
[ec2-user@ip-10-0-1-231 ~]$ login as: ec2-user
Authenticating with public key "698finalproject"
[ec2-user@ip-10-0-1-231 ~]$ 
[ec2-user@ip-10-0-1-231 ~]$ sudo yum install -y stress
Last metadata expiration check: 0:28:31 ago on Sun Dec 7 00:34:42 2025.
Dependencies resolved.
=====
 Package      Architecture Version      Repository      Size
=====
 Installing:
  stress       x86_64        1.0.7-2.amzn2023.0.1      amazonlinux    34 k
Transaction Summary
=====
 Install 1 Package
Total download size: 34 k
Installed size: 68 k
Downloading Packages:
stress-1.0.7-2.amzn2023.0.1.x86_64.rpm      1.1 MB/s | 34 kB     00:00
-----
Total                                         580 kB/s | 34 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing :                                         1/1
Installing : stress-1.0.7-2.amzn2023.0.1.x86_64      1/1
Running scriptlet: stress-1.0.7-2.amzn2023.0.1.x86_64      1/1
Verifying   : stress-1.0.7-2.amzn2023.0.1.x86_64      1/1
Installed:
  stress-1.0.7-2.amzn2023.0.1.x86_64
Complete!
[ec2-user@ip-10-0-1-231 ~]$ stress --cpu 4 --timeout 120
stress: info: [26995] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
```

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP	IPv6 IPs
simple-ex2...	i-0467494...	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	ec2-44-199-236-168.co...	44.199.236.168	-	-
asg-web-inst...	i-0904f17c...	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1a	ec2-44-204-186-241.co...	44.204.186.241	-	-
asg-web-inst...	i-0cf9923a...	Running	t3.micro	3/3 checks passed	View alarms +	us-east-1b	ec2-54-145-162-109.co...	54.145.162.109	-	-

Desired capacity: 1  
Scaling limits (Min - Max): 1 - 3  
Desired capacity type: Units (number of instances)  
Status: -

Date created: Sat Dec 06 2025 19:34:11 GMT-0500 (Eastern Standard Time)

**Instances (2)**

Instance ID	Lifecycle	Instance type	Weighted capacity	Launch templ...	Availability Zone	Health status	Protected from
i-0904f17c1ea3561b0	InService	t3.micro	-	ec2autoscaling-web-lt	use1-az1 (us-east-1a)	Healthy	
i-0ef9923a62fa82bfd	InService	t3.micro	-	ec2autoscaling-web-lt	use1-az2 (us-east-1b)	Healthy	

**Instance lifecycle policy for lifecycle hooks**

Controls instance behavior when an instance transitions through its lifecycle states.

**Termination hook abandon behavior**

Terminate (default)

**Lifecycle hooks (0)**

No lifecycle hooks are currently configured.

Desired capacity: 1  
Scaling limits (Min - Max): 1 - 3  
Desired capacity type: Units (number of instances)  
Status: -

Date created: Sat Dec 06 2025 19:34:11 GMT-0500 (Eastern Standard Time)

**Activity notifications (0)**

No notifications are currently specified.

**Activity history (2)**

Status	Description	Cause	Start time	End time
Successful	Launching a new EC2 instance: i-0ef9923a62fa82bfd	At 2025-12-07T00:34:22 a user request update of AutoScalingGroup constraints to min: 1, max: 3, desired: 1 changing the desired capacity from 0 to 1. At 2025-12-07T00:34:14Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2025 December 06, 08:09:35 PM - 05:00	2025 December 06, 08:11:40 PM - 05:00
Successful	Launching a new EC2 instance: i-0904f17c1ea3561b0	At 2025-12-07T00:34:22 a user request update of AutoScalingGroup constraints to min: 1, max: 3, desired: 1 changing the desired capacity from 0 to 1. At 2025-12-07T00:34:14Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.	2025 December 06, 07:34:16 PM - 05:00	2025 December 06, 07:34:21 PM - 05:00

Here, I created a launch template and linked it to an Auto Scaling Group (ASG). The ASG automatically launched new EC2 instances in the public subnets whenever the load increased. I tested autoscaling by SSH-ing into the instance and stressing the CPU. When CloudWatch detected high CPU usage, a new EC2 instance was created.

## B. AWS Lambda for Logging S3 Uploads

The screenshot shows the AWS CloudFormation console with the 's3lambda698' stack selected. The 'Events' tab is active, displaying 14 events. The events log the creation of various AWS resources by the Lambda function:

Operation ID	Timestamp	Logical ID	Status	Detailed status	Status reason
a620386a-64b0-4140-ac1-2f5c47356090	2025-12-06 21:05:53 UTC-0500	s3lambda698	CREATE_COMPLETE	-	-
a620386a-64b0-4140-ac1-2f5c47356090	2025-12-06 21:05:52 UTC-0500	AppBucket	CREATE_COMPLETE	-	-
a620386a-64b0-4140-ac1-2f5c47356090	2025-12-06 21:05:38 UTC-0500	S3InvokeLambdaPermission	CREATE_COMPLETE	-	-
a620386a-64b0-4140-ac1-2f5c47356090	2025-12-06 21:05:38 UTC-0500	AppBucket	CREATE_IN_PROGRESS	-	Resource creation Initiated
a620386a-64b0-4140-ac1-2f5c47356090	2025-12-06 21:05:38 UTC-0500	S3InvokeLambdaPermission	CREATE_IN_PROGRESS	-	Resource creation Initiated
... (remaining 8 events)					

The screenshot shows the AWS S3 console with the 's3lambda698-appbucket-zebm2iebyvh6' bucket selected. The 'Objects' tab is active, showing 0 objects. A message indicates "No objects" and "You don't have any objects in this bucket." There is a prominent "Upload" button at the bottom.

The screenshot shows the AWS Lambda console interface. At the top, the navigation bar includes 'Lambda > Functions > s3lambda698-s3-logger'. The main area displays the function overview, showing a diagram of the function, its layers, and triggers. The 'Code' tab is selected, showing the Python code for the function. Below the code is a screenshot of the CloudWatch Logs interface, which shows log events for the Lambda function's execution. The logs include details such as the start of the function, received event records, and the successful upload of a new object to S3.

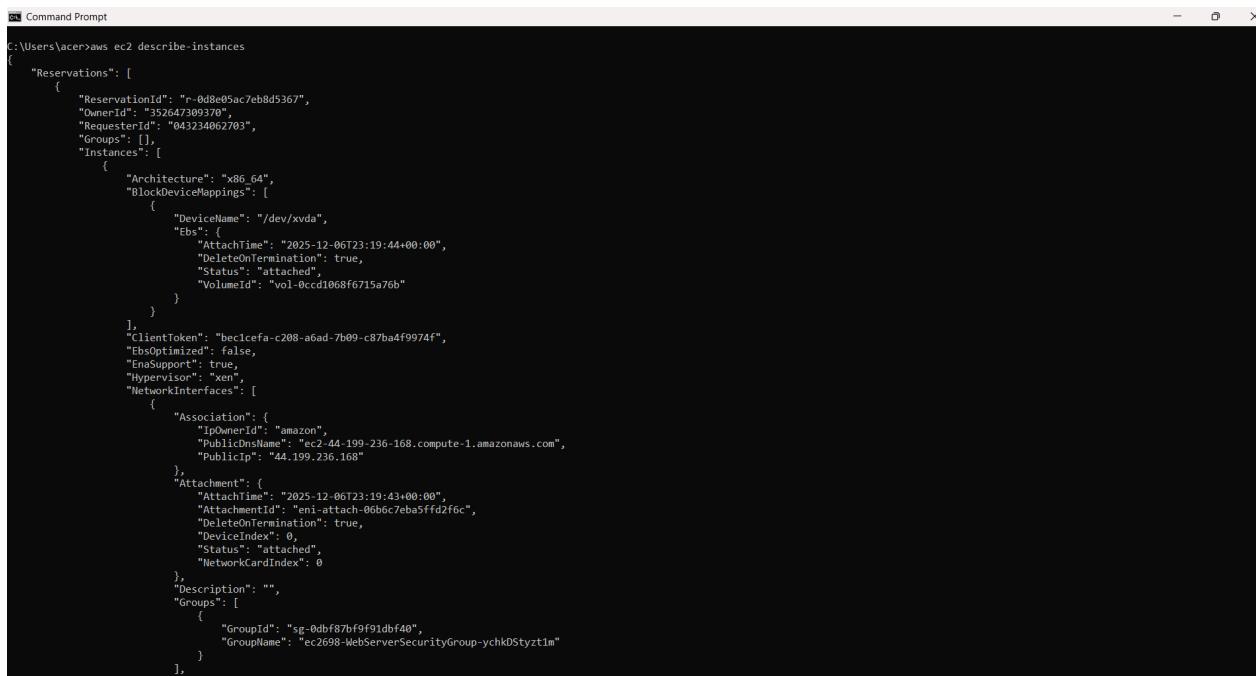
In this step I created a Lambda function in Python that writes logs to CloudWatch whenever someone uploads a file to a specific S3 bucket. After attaching the S3 event trigger, every upload will trigger the Lambda function. When I checked CloudWatch Logs, I could see the file name and metadata.

## C. AWS Interaction

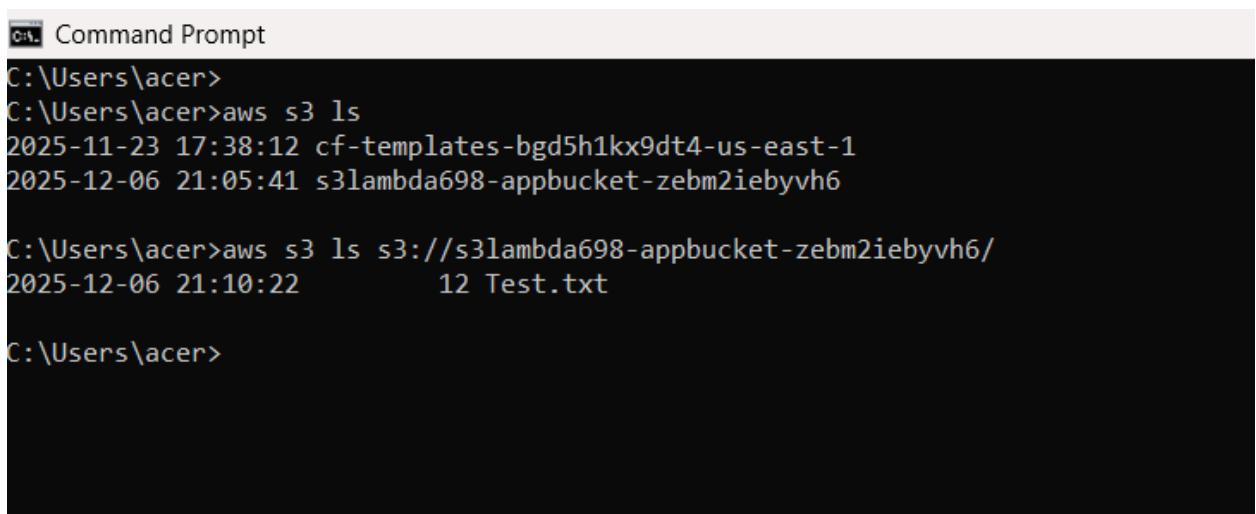
**AWS CLI:** Manage EC2, S3, and Lambda resources.

Using AWS CLI commands, I listed the running instances in the EC2, listed the Buckets and files in the S3 bucket. Also I invoked the lambda function.

**Python Boto3 scripts:**



```
C:\Users\acer>aws ec2 describe-instances
{
    "Reservations": [
        {
            "ReservationId": "r-0d8e05ac7eb8d5367",
            "OwnerId": "352647309270",
            "RequesterId": "043234062703",
            "Groups": [],
            "Instances": [
                {
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/xvda",
                            "Ebs": {
                                "AttachTime": "2025-12-06T23:19:44+00:00",
                                "DeleteOnTermination": true,
                                "Status": "attached",
                                "VolumeId": "vol-0cccd1068f6715a76b"
                            }
                        }
                    ],
                    "ClientToken": "bnClcefa-a-c208-a6ad-7b09-c87ba4ff9974f",
                    "EbsOptimized": false,
                    "EnaSupport": true,
                    "Hypervisor": "xen",
                    "NetworkInterfaces": [
                        {
                            "Association": {
                                "IpOwnerId": "amazon",
                                "PublicDnsName": "ec2-44-199-236-168.compute-1.amazonaws.com",
                                "PublicIp": "44.199.236.168"
                            },
                            "Attachment": {
                                "AttachTime": "2025-12-06T23:19:43+00:00",
                                "AttachmentId": "eni-attach-0eb6c7eba5ffd2fbc",
                                "DeleteOnTermination": true,
                                "DeviceIndex": 0,
                                "Status": "attached",
                                "NetworkCardIndex": 0
                            },
                            "Description": "",
                            "Groups": [
                                {
                                    "GroupId": "sg-0dbf87bf9f91dbf40",
                                    "GroupName": "ec2698-WebServerSecurityGroup-yckh0Styzt1m"
                                }
                            ]
                        }
                    ],
                    "Description": "Amazon Linux 2 AMI - 2025.09.0 (HVM, SSD Volume Type) - 2025-12-06T23:19:43Z"
                }
            ]
        }
    ]
}
```



```
C:\Users\acer>
C:\Users\acer>aws s3 ls
2025-11-23 17:38:12 cf-templates-bgd5h1kx9dt4-us-east-1
2025-12-06 21:05:41 s3lambda698-appbucket-zebm2iebyvh6

C:\Users\acer>aws s3 ls s3://s3lambda698-appbucket-zebm2iebyvh6/
2025-12-06 21:10:22          12 Test.txt

C:\Users\acer>
```

```

on Command Prompt
C:\Users\acer>aws s3 ls
2025-11-23 17:38:12 cf-templates-bgd5h1kx9dt4-us-east-1
2025-12-06 21:05:41 s3lambda698-appbucket-zebm2iebyvh6

C:\Users\acer>aws s3 ls s3://s3lambda698-appbucket-zebm2iebyvh6
2025-12-06 21:10:22          12 Test.txt

C:\Users\acer>aws lambda list-functions
{
  "Functions": [
    {
      "FunctionName": "s3lambda698-s3-logger",
      "FunctionArn": "arn:aws:lambda:us-east-1:352647309370:function:s3lambda698-s3-logger",
      "Runtime": "python3.11",
      "Role": "arn:aws:iam::352647309370:role/s3lambda698-lambda-role",
      "Handler": "index.lambda_handler",
      "CodeSize": 386,
      "Description": "",
      "Timeout": 10,
      "MemorySize": 128,
      "LastModified": "2025-12-07T02:05:30.909+0000",
      "CodeSha256": "k+a+CYMFo9fwm5jYE8hf/bBJ03N6dDrfQBoKQ5bCPhY=",
      "Version": "$LATEST",
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "RevisionId": "af49d294-1475-44c3-8939-7772abedb755",
      "PackageType": "Zip",
      "Architectures": [
        "x86_64"
      ],
      "EphemeralStorage": {
        "Size": 512
      },
      "SnapStart": {
        "ApplyOn": "None",
        "OptimizationStatus": "Off"
      },
      "LoggingConfig": {
        "LogFormat": "Text",
        "LogGroup": "/aws/lambda/s3lambda698-s3-logger"
      }
    }
  ]
}

C:\Users\acer>

```

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and account information (Account ID: 3525-4730-9770, Sumanth\_Console\_User). Below the navigation bar, the URL shows the path: Amazon S3 > Buckets > boto3-demo-bucket-c42a63e1. The main content area displays the details for the 'boto3-demo-bucket-c42a63e1' bucket. A heading 'boto3-demo-bucket-c42a63e1 [Info](#)' is followed by a navigation menu with tabs: Objects, Metadata, Properties, Permissions, Metrics, Management, and Access Points. The 'Objects' tab is selected. Below the menu, a sub-menu for 'Objects (1)' is shown with a note: '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](#)'. There's a search bar with the placeholder 'Find objects by prefix'. A table lists the object 'hello.txt' with columns: Name, Type, Last modified, Size, and Storage class. The object 'hello.txt' is listed with a file icon, 'txt' type, 'December 7, 2025, 00:43:32 (UTC-05:00)' last modified, '25.0 B' size, and 'Standard' storage class. Action buttons for the object include Copy S3 URI, Copy URL, Download, Open in browser, Delete, Actions (with a dropdown arrow), Create folder, and Upload.

Name	Type	Last modified	Size	Storage class
hello.txt	txt	December 7, 2025, 00:43:32 (UTC-05:00)	25.0 B	Standard

```
ec2-user@ip-10-0-1-65:~  
python3-pip-21.3.1-2.amzn2023.0.14.noarch  
  
Complete!  
[ec2-user@ip-10-0-1-65 ~]$ pip3 install requests  
Defaulting to user installation because normal site-packages is not writeable  
Requirement already satisfied: requests in /usr/lib/python3.9/site-packages (2.25.1)  
Requirement already satisfied: idna<3,>=2.5 in /usr/lib/python3.9/site-packages  
(from requests) (2.10)  
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/lib/python3.9/site-packages  
(from requests) (1.25.10)  
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/lib/python3.9/site-packages  
(from requests) (4.0.0)  
[ec2-user@ip-10-0-1-65 ~]$ nano ec2_metadata.py  
[ec2-user@ip-10-0-1-65 ~]$ python3 ec2_metadata.py  
Error fetching metadata: 401 Client Error: Unauthorized for url: http://169.254.169.254/latest/meta-data/instance-id  
[ec2-user@ip-10-0-1-65 ~]$ nano ec2_metadata.py  
[ec2-user@ip-10-0-1-65 ~]$ python3 ec2_metadata.py  
Instance ID: i-04674943ff91fc96b  
Availability Zone: us-east-1a  
Local IPv4: 10.0.1.65  
Public IPv4: 44.199.236.168  
[ec2-user@ip-10-0-1-65 ~]$
```

```
Command Prompt  
  
C:\Users\acer\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.13>python create_bucket.py  
Creating bucket: boto3-demo-bucket-c42a63e1  
Uploading hello.txt to boto3-demo-bucket-c42a63e1  
Done. You can verify in S3 console.  
  
C:\Users\acer\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.13>python list_ec2.py  
Running EC2 instances:  
- ID: i-04674943ff91fc96b, State: running, Name: simple-ec2-public, Public IP: 44.199.236.168  
- ID: i-0904f17c1ea3561b0, State: running, Name: asg-web-instance, Public IP: 44.204.186.241  
  
C:\Users\acer\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.13>python invoke_lambda.py  
Invoking Lambda: s3lambda698-s3-logger  
Lambda response payload:  
{"status": "ok"}  
  
C:\Users\acer\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.13>
```

I wrote Python scripts to:

- create an S3 bucket
- upload a file
- retrieve EC2 metadata
- list running EC2 instances
- invoke Lambda manually

## Bonus Challenge

Deploy **API Gateway** to invoke Lambda via HTTP requests.

The screenshot shows the AWS API Gateway interface. In the top navigation bar, the path is "APIs > APIs > Resources - LambdaRestAPI (zc9ntic29a)". A green success message at the top right says "Successfully created REST API 'LambdaRestAPI (zc9ntic29a)'". On the left sidebar, under "API: LambdaRestAPI", there are sections for Resources, Stages, Authorizers, Gateway responses, Models, Resource policy, Documentation, Dashboard, and API settings. The main panel displays "Resource details" for a root resource with Path "/". It lists "Methods (0)" and "Integration type". Buttons for "Update documentation", "Enable CORS", "Delete", and "Create method" are available. The "Resource ID" is u8528e33qj.

This screenshot shows the same AWS API Gateway interface, but it focuses on a specific method. The path in the top navigation is "Resources - LambdaRestAPI (zc9ntic29a) /hellosumanth - GET - Method execution". The main panel shows the ARN: arn:aws:execute-api:us-east-1:352647309370:zc9ntic29a/\*/GET/hellosumanth and the Resource ID kfau9. A flow diagram illustrates the request process: Client → Method request → Integration request → Integration response → Method response → Lambda integration. Below this, tabs for Method request, Integration request, Integration response, Method response, and Test are visible. Under Method request settings, it shows Authorization: NONE, Request validator: None, API key required: False, and SDK operation name: Generated based on method and path. The Request paths section shows one entry with Name: /hellosumanth and Caching: No request cache.

A screenshot of a web browser window displaying a JSON response. The address bar shows the URL: `zc9ntic29a.execute-api.us-east-1.amazonaws.com/dev`. Below the address bar is a toolbar with icons for back, forward, and refresh. A "Pretty-print" checkbox is checked. The main content area contains the following JSON:

```
{  
  "message": "S3 upload logger Lambda executed successfully.",  
  "source": "apigateway",  
  "uploaded_files": []  
}
```