

Afrin Unnisa Syed

EE06693

IS 698

Project Report

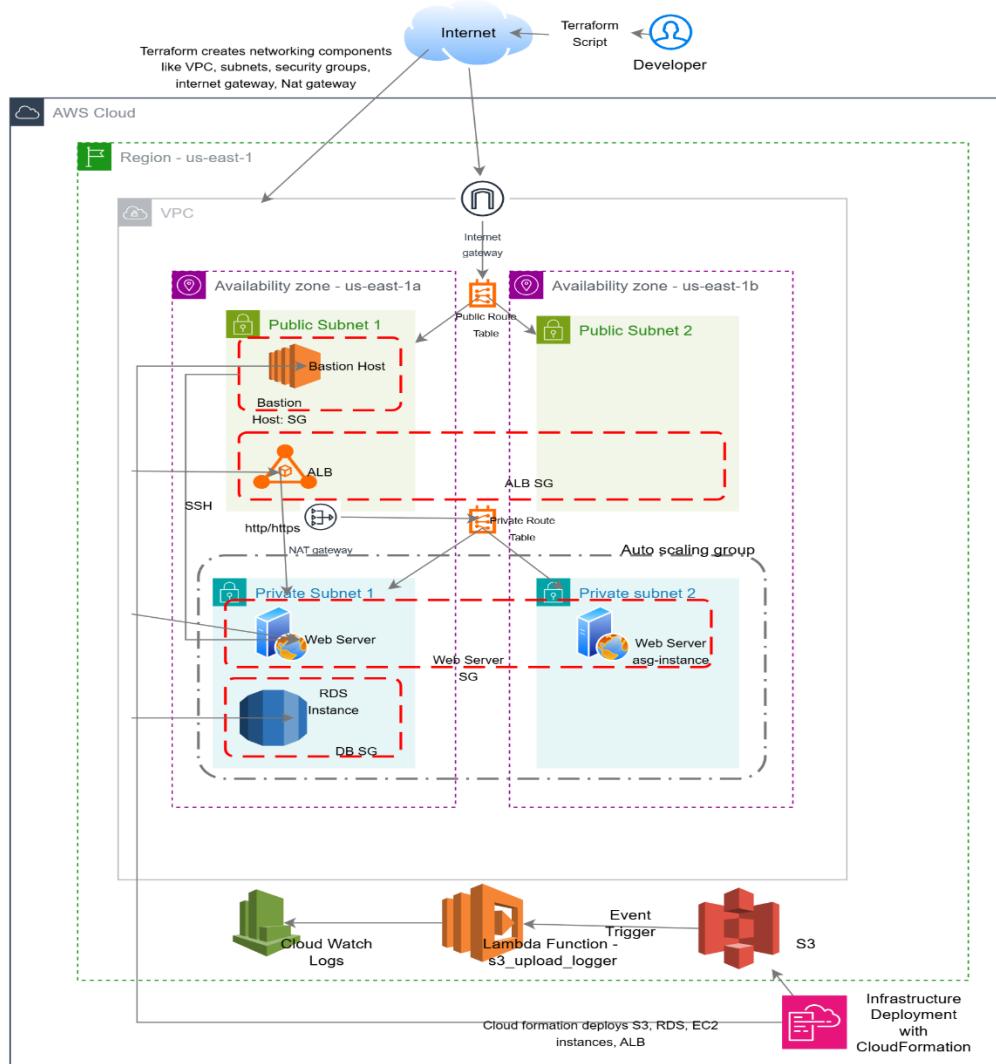
Deploying a Scalable AWS Architecture with Infrastructure as Code

[Github](#)

GitHub repository link - <https://github.com/NisaAf/aws-architecture-project>

1. Design an AWS Architecture

<https://github.com/NisaAf/aws-architecture-project/blob/main/docs/Architecture%20Diagram.pdf>



2. Implementation

A. Infrastructure Deployment

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

Terraform file - <https://github.com/NisaAf/aws-architecture-project/blob/main/terraform/main.tf>

Terraform initiate

```
C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4\aws-architecture-project>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.97.0

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.
```

Terraform plan

```
Command Prompt
C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4\aws-architecture-project>terraform plan
commands will detect it and remind you to do so if necessary.

C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4\aws-architecture-project>terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
+ create

Terraform will perform the following actions:

# aws_eip.nat will be created
+ resource "aws_eip" "nat" {
    + allocation_id      = (known after apply)
    + arn                = (known after apply)
    + association_id    = (known after apply)
    + carrier_ip         = (known after apply)
    + customer_owned_ip = (known after apply)
    + domain             = "vpc"
    + id                 = (known after apply)
    + instance            = (known after apply)
    + ipam_pool_id       = (known after apply)
    + network_border_group = (known after apply)
    + network_interface   = (known after apply)
    + private_dns         = (known after apply)
    + private_ip          = (known after apply)
    + ptr_record          = (known after apply)
    + public_dns          = (known after apply)
    + public_ip           = (known after apply)
    + public_ipv4_pool    = (known after apply)
    + tags               = {
```

```
Command Prompt + - X

# aws_internet_gateway.main will be created
+ resource "aws_internet_gateway" "main" {
  + arn      = (known after apply)
  + id       = (known after apply)
  + owner_id = (known after apply)
  + tags     = {
    + "Name" = "project-igw"
  }
  + tags_all = {
    + "Name" = "project-igw"
  }
  + vpc_id   = (known after apply)
}

# aws_nat_gateway.main will be created
+ resource "aws_nat_gateway" "main" {
  + allocation_id      = (known after apply)
  + association_id     = (known after apply)
  + connectivity_type  = "public"
  + id                 = (known after apply)
  + network_interface_id = (known after apply)
  + private_ip          = (known after apply)
  + public_ip           = (known after apply)
  + secondary_private_ip_address_count = (known after apply)
  + secondary_private_ip_addresses = (known after apply)
  + subnet_id          = (known after apply)
  + tags               = {
    + "Name" = "project-nat"
  }
}
```

```
Command Prompt + - X

# aws_route_table.private will be created
+ resource "aws_route_table" "private" {
  + arn      = (known after apply)
  + id       = (known after apply)
  + owner_id = (known after apply)
  + propagating_vgwss = (known after apply)
  + route    = [
    +
      + {
        + cidr_block      = "0.0.0.0/0"
        + nat_gateway_id = (known after apply)
        # (11 unchanged attributes hidden)
      },
    ],
  + tags     = {
    + "Name" = "private-route-table"
  }
  + tags_all = {
    + "Name" = "private-route-table"
  }
  + vpc_id   = (known after apply)
}

# aws_route_table.public will be created
+ resource "aws_route_table" "public" {
  + arn      = (known after apply)
  + id       = (known after apply)
  + owner_id = (known after apply)
  + propagating_vgwss = (known after apply)
  + route    = [
```

```
Command Prompt + - X

# aws_route_table_association.private_1 will be created
+ resource "aws_route_table_association" "private_1" {
  + id          = (known after apply)
  + route_table_id = (known after apply)
  + subnet_id    = (known after apply)
}

# aws_route_table_association.private_2 will be created
+ resource "aws_route_table_association" "private_2" {
  + id          = (known after apply)
  + route_table_id = (known after apply)
  + subnet_id    = (known after apply)
}

# aws_route_table_association.public_1 will be created
+ resource "aws_route_table_association" "public_1" {
  + id          = (known after apply)
  + route_table_id = (known after apply)
  + subnet_id    = (known after apply)
}

# aws_route_table_association.public_2 will be created
+ resource "aws_route_table_association" "public_2" {
  + id          = (known after apply)
  + route_table_id = (known after apply)
  + subnet_id    = (known after apply)
}

# aws_security_group.alb will be created
```

```
Command Prompt + - X

# aws_security_group.alb will be created
+ resource "aws_security_group" "alb" {
  + arn          = (known after apply)
  + description   = "Security group for application load balancer"
  + egress        = [
    + {
      + cidr_blocks  = [
        + "0.0.0.0/0",
      ]
      + from_port    = 0
      + ipv6_cidr_blocks = []
      + prefix_list_ids = []
      + protocol     = "-1"
      + security_groups = []
      + self         = false
      + to_port      = 0
      # (1 unchanged attribute hidden)
    },
  ],
  + id          = (known after apply)
  + ingress      = [
    + {
      + cidr_blocks  = [
        + "0.0.0.0/0",
      ]
      + from_port    = 443
      + ipv6_cidr_blocks = []
      + prefix_list_ids = []
      + protocol     = "tcp"
    }
  ]
}
```

```
Command Prompt + ▾

}

# aws_security_group.bastion will be created
+ resource "aws_security_group" "bastion" {
+   arn           = (known after apply)
+   description    = "Security group for bastion host"
+   egress         = [
+     {
+       cidr_blocks  = [
+         "+ "0.0.0.0/0",
+       ]
+       from_port    = 0
+       ipv6_cidr_blocks = []
+       prefix_list_ids = []
+       protocol      = "-1"
+       security_groups = []
+       self          = false
+       to_port        = 0
# (1 unchanged attribute hidden)
    },
  ],
+   id           = (known after apply)
+   ingress       = [
+     {
+       cidr_blocks  = [
+         "+ "0.0.0.0/0",
+       ]
+       from_port    = 22
+       ipv6_cidr_blocks = []
+       prefix_list_ids = []

```

```
Command Prompt + ▾

# aws_security_group.db will be created
+ resource "aws_security_group" "db" {
+   arn           = (known after apply)
+   description    = "Security group for database"
+   egress         = [
+     {
+       cidr_blocks  = [
+         "+ "0.0.0.0/0",
+       ]
+       from_port    = 0
+       ipv6_cidr_blocks = []
+       prefix_list_ids = []
+       protocol      = "-1"
+       security_groups = []
+       self          = false
+       to_port        = 0
# (1 unchanged attribute hidden)
    },
  ],
+   id           = (known after apply)
+   ingress       = [
+     {
+       cidr_blocks  = []
+       from_port    = 3306
+       ipv6_cidr_blocks = []
+       prefix_list_ids = []
+       protocol      = "tcp"
+       security_groups = (known after apply)
+       self          = false

```

```
Command Prompt + - X

}

# aws_vpc.main will be created
+ resource "aws_vpc" "main" {
    + arn                      = (known after apply)
    + cidr_block               = "10.0.0.0/16"
    + default_network_acl_id   = (known after apply)
    + default_route_table_id   = (known after apply)
    + default_security_group_id = (known after apply)
    + dhcp_options_id          = (known after apply)
    + enable_dns_hostnames     = true
    + enable_dns_support        = true
    + enable_network_address_usage_metrics = (known after apply)
    + id                        = (known after apply)
    + instance_tenancy          = "default"
    + ipv6_association_id       = (known after apply)
    + ipv6_cidr_block           = (known after apply)
    + ipv6_cidr_block_network_border_group = (known after apply)
    + main_route_table_id       = (known after apply)
    + owner_id                  = (known after apply)
    + tags                      = {
        + "Name" = "project-vpc"
    }
    + tags_all                 = {
        + "Name" = "project-vpc"
    }
}

Plan: 18 to add, 0 to change, 0 to destroy.
```

```
Plan: 18 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ alb_security_group_id      = (known after apply)
+ bastion_security_group_id   = (known after apply)
+ db_security_group_id        = (known after apply)
+ private_subnet_1_id         = (known after apply)
+ private_subnet_2_id         = (known after apply)
+ public_subnet_1_id          = (known after apply)
+ public_subnet_2_id          = (known after apply)
+ vpc_id                     = (known after apply)
+ web_security_group_id       = (known after apply)

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
```

Execute Terraform deployment – Terraform Apply

```
Command Prompt x + v
C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4\aws-architecture-project>terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
+ create

Terraform will perform the following actions:

# aws_eip.nat will be created
+ resource "aws_eip" "nat" {
    + allocation_id      = (known after apply)
    + arn                = (known after apply)
    + association_id    = (known after apply)
    + carrier_ip         = (known after apply)
    + customer_owned_ip = (known after apply)
    + domain              = "vpc"
    + id                 = (known after apply)
    + instance             = (known after apply)
    + ipam_pool_id        = (known after apply)
    + network_border_group = (known after apply)
    + network_interface   = (known after apply)
    + private_dns          = (known after apply)
    + private_ip            = (known after apply)
    + ptr_record           = (known after apply)
    + public_dns            = (known after apply)
    + public_ip              = (known after apply)
    + public_ipv4_pool     = (known after apply)
    + tags
        + "Name" = "nat-eip"
}
```

```
Command Prompt x + v
# aws_nat_gateway.main will be created
+ resource "aws_nat_gateway" "main" {
    + allocation_id      = (known after apply)
    + association_id    = (known after apply)
    + connectivity_type = "public"
    + id                 = (known after apply)
    + network_interface_id = (known after apply)
    + private_ip          = (known after apply)
    + public_ip            = (known after apply)
    + secondary_private_ip_address_count = (known after apply)
    + secondary_private_ip_addresses = (known after apply)
    + subnet_id           = (known after apply)
    + tags
        + "Name" = "project-nat"
    }
    + tags_all
        + "Name" = "project-nat"
}

# aws_route_table.private will be created
+ resource "aws_route_table" "private" {
    + arn                = (known after apply)
    + id                 = (known after apply)
    + owner_id            = (known after apply)
    + propagating_vgwss = (known after apply)
    + route
        + {
            + cidr_block      = "0.0.0.0/0"
            + nat_gateway_id = (known after apply)
        }
}
```

```
Command Prompt + - X

# aws_route_table.public will be created
+ resource "aws_route_table" "public" {
+   arn          = (known after apply)
+   id           = (known after apply)
+   owner_id     = (known after apply)
+   propagating_vgws = (known after apply)
+   route        = [
+     {
+       + cidr_block      = "0.0.0.0/0"
+       + gateway_id     = (known after apply)
+       # (11 unchanged attributes hidden)
+     },
+   ],
+   tags          = {
+     + "Name" = "public-route-table"
+   }
+   tags_all      = {
+     + "Name" = "public-route-table"
+   }
+   vpc_id        = (known after apply)
}

# aws_route_table_association.private_1 will be created
+ resource "aws_route_table_association" "private_1" {
+   + id          = (known after apply)
+   + route_table_id = (known after apply)
+   + subnet_id    = (known after apply)
+ }
```

```
Command Prompt + - X

}
}

Plan: 18 to add, 0 to change, 0 to destroy.

Changes to Outputs:
+ alb_security_group_id      = (known after apply)
+ bastion_security_group_id   = (known after apply)
+ db_security_group_id        = (known after apply)
+ private_subnet_1_id         = (known after apply)
+ private_subnet_2_id         = (known after apply)
+ public_subnet_1_id          = (known after apply)
+ public_subnet_2_id          = (known after apply)
+ vpc_id                      = (known after apply)
+ web_security_group_id       = (known after apply)

aws_eip.nat: Creating...
aws_vpc.main: Creating...
aws_eip.nat: Creation complete after 1s [id=eipalloc-05bef6ce6baa090c4]
aws_vpc.main: Still creating... [10s elapsed]
aws_vpc.main: Creation complete after 12s [id=vpc-0235f5ea6483aa4a8]
aws_subnet.private_1: Creating...
aws_internet_gateway.main: Creating...
aws_subnet.public_1: Creating...
aws_subnet.private_2: Creating...
aws_subnet.public_2: Creating...
aws_security_group.bastion: Creating...
aws_security_group.alb: Creating...
aws_internet_gateway.main: Creation complete after 0s [id=igw-0d7fd107d00683493]
aws_route_table.public: Creating...
aws_subnet.private_2: Creation complete after 0s [id=subnet-01d3df3b126a8bb0a]
```

```
Command Prompt + - X

aws_route_table.public: Creating...
aws_subnet.private_2: Creation complete after 0s [id=subnet-01d3df3b126a8bb0a]
aws_route_table.public: Creation complete after 1s [id=rtb-0108692629cb3a3e8]
aws_subnet.private_1: Creation complete after 2s [id=subnet-0ee15382af24b3bc]
aws_security_group.bastion: Creation complete after 2s [id=sg-08e5ec2eba579f918]
aws_security_group.alb: Creation complete after 2s [id=sg-0047f057fa0ae357e]
aws_security_group.web: Creating...
aws_security_group.web: Creation complete after 3s [id=sg-00595c492b15a45bb]
aws_security_group.db: Creating...
aws_security_group.db: Creation complete after 2s [id=sg-0687de5677a1790b6]
aws_subnet.public_1: Still creating... [10s elapsed]
aws_subnet.public_2: Still creating... [10s elapsed]
aws_subnet.public_1: Creation complete after 11s [id=subnet-052c267e072b8a2cf]
aws_route_table_association.public_1: Creating...
aws_nat_gateway.main: Creating...
aws_subnet.public_2: Creation complete after 11s [id=subnet-07a3fcfc6fbe1ca0]
aws_route_table_association.public_2: Creating...
aws_route_table_association.public_1: Creation complete after 0s [id=rtbassoc-04e90b03491c3c639]
aws_route_table_association.public_2: Creation complete after 0s [id=rtbassoc-0b25f9e29a40af436]
aws_nat_gateway.main: Still creating... [10s elapsed]
aws_nat_gateway.main: Still creating... [20s elapsed]
aws_nat_gateway.main: Still creating... [30s elapsed]
aws_nat_gateway.main: Still creating... [40s elapsed]
aws_nat_gateway.main: Still creating... [50s elapsed]
aws_nat_gateway.main: Still creating... [1m0s elapsed]
aws_nat_gateway.main: Still creating... [1m10s elapsed]
aws_nat_gateway.main: Still creating... [1m20s elapsed]
aws_nat_gateway.main: Still creating... [1m30s elapsed]
aws_nat_gateway.main: Still creating... [1m40s elapsed]
aws_nat_gateway.main: Still creating... [1m50s elapsed]
```

```
Command Prompt + - X

aws_nat_gateway.main: Still creating... [1m20s elapsed]
aws_nat_gateway.main: Still creating... [1m30s elapsed]
aws_nat_gateway.main: Still creating... [1m40s elapsed]
aws_nat_gateway.main: Still creating... [1m50s elapsed]
aws_nat_gateway.main: Still creating... [2m0s elapsed]
aws_nat_gateway.main: Still creating... [2m10s elapsed]
aws_nat_gateway.main: Creation complete after 2m14s [id=nat-0477174ef06d1906b]
aws_route_table.private: Creating...
aws_route_table.private: Creation complete after 1s [id=rtb-0a18eca9f1cc91917]
aws_route_table_association.private_1: Creating...
aws_route_table_association.private_2: Creating...
aws_route_table_association.private_1: Creation complete after 1s [id=rtbassoc-0cdb7fdb2ef443407]
aws_route_table_association.private_2: Still creating... [10s elapsed]
aws_route_table_association.private_2: Creation complete after 14s [id=rtbassoc-0100aed60d18f3fc9]

Apply complete! Resources: 18 added, 0 changed, 0 destroyed.

Outputs:

alb_security_group_id = "sg-0047f057fa0ae357e"
bastion_security_group_id = "sg-08e5ec2eba579f918"
db_security_group_id = "sg-0687de5677a1790b6"
private_subnet_1_id = "subnet-0ee15382af24b3bc"
private_subnet_2_id = "subnet-01d3df3b126a8bb0a"
public_subnet_1_id = "subnet-052c267e072b8a2cf"
public_subnet_2_id = "subnet-07a3fcfc6fbe1ca0"
vpc_id = "vpc-0235f5ea6483aa4a8"
web_security_group_id = "sg-00595c492b15a45bb"

C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4\aws-architecture-project\terraform>
```

Verify all the VPC components on the AWS console

The screenshot shows the AWS VPC dashboard for a specific VPC. On the left, there's a sidebar with various navigation options like EC2 Global View, Virtual private cloud, Security, PrivateLink and Latte, and CloudShell. The main panel displays the 'Details' tab for the VPC, showing its ID, state (available), and various configuration settings. Below the details is a 'Resource map' section which provides a visual representation of the VPC's internal structure, including subnets, route tables, and network connections.

- Using CloudFormation to Deploy EC2, RDS, and Lambda

Create CloudFormation template - <https://github.com/NisaAf/aws-architecture-project/blob/main/cloudformation/aws-infrastructure.yaml>

Bastion Host: Amazon Linux 2 instance in public subnet

Web Server: Amazon Linux 2 instance in private subnet

RDS Database: MySQL 8.0 instance in private subnet

S3 Bucket: For file storage

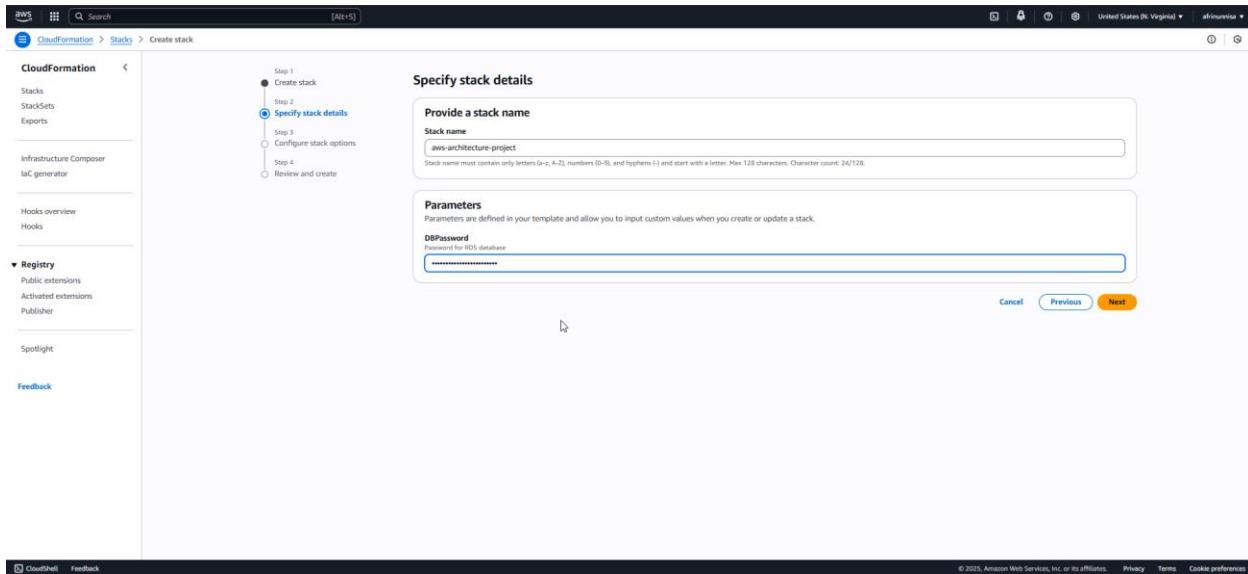
Lambda Function: Basic function for S3 event processing

Application Load Balancer: In public subnet, targeting web server

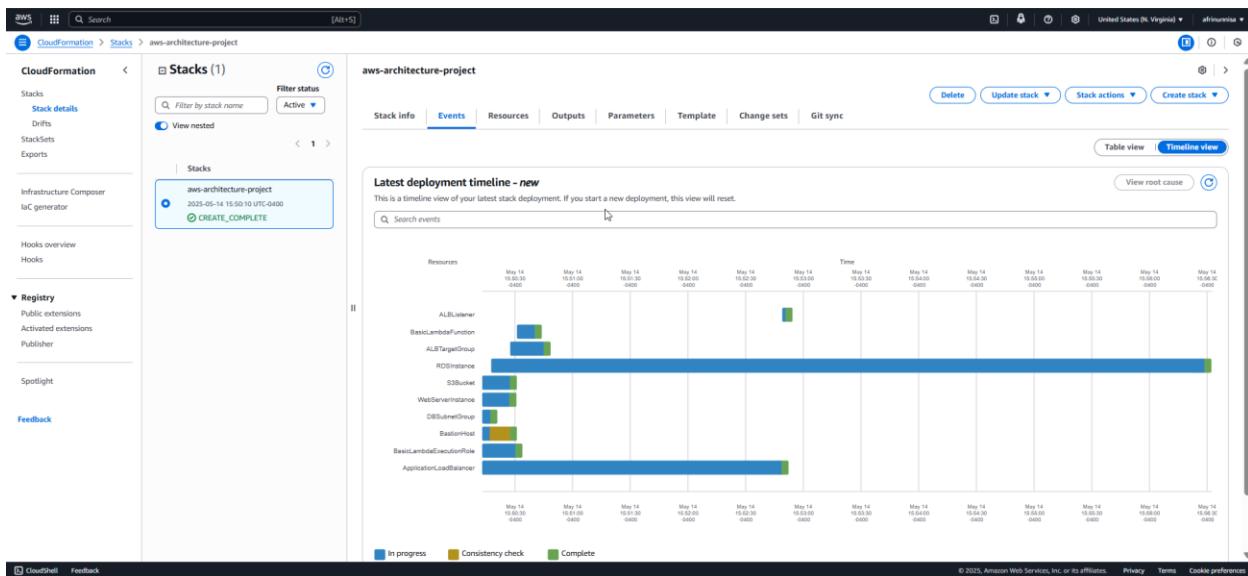
Uploaded template via AWS Management Console

The screenshot shows the AWS CloudFormation 'Create stack' wizard. It's currently at Step 2, 'Choose an existing template'. The user has selected the 'Upload a template file' option and uploaded a file named 'aws-infrastructure.yaml'. The template source section also shows options for 'Amazon S3 URL' and 'Sync from Git'. At the bottom, there are 'Cancel' and 'Next' buttons.

Provided secure password for RDS database



Monitored stack creation progress



CloudFormation > Stacks > aws-architecture-project

CloudFormation

Stacks

- Stack details
- Drifts
- StackSets
- Exports

Infrastructure Composer

IaC generator

Hooks overview

Hooks

Registry

- Public extensions
- Activated extensions
- Publisher

Spotlight

Feedback

Stacks (1)

Filter status

View nested

aws-architecture-project

2023-05-14 15:50:10 UTC-0400

Resources (10)

Logical ID	Physical ID	Type	Status	Module
ALBListener	arn:aws:elasticloadbalancing:us-east-1:730353536292:listener/app/web-alb/ff14a6d9b7b64d7e5a24215bd1fe4	AWS::ElasticLoadBalancingV2::Listener	CREATE_COMPLETE	-
ALBTargetGroup	17.103.53.53:4242/targetgroup/web-target/eaed0311142324847	AWS::ElasticLoadBalancingV2::TargetGroup	CREATE_COMPLETE	-
ApplicationLoadBalancer	arn:aws:elasticloadbalancing:us-east-1:730353536292:loadbalancer/app/web-alb/ff14a6d9b7b64d7e5a24215bd1fe4	AWS::ElasticLoadBalancingV2::LoadBalancer	CREATE_COMPLETE	-
BasicLambdaExecutionRole	aws-architecture-project-basic-lambda-execution-role-w93sls1hs	AWS::IAM::Role	CREATE_COMPLETE	-
BasicLambdaFunction	basic-function	AWS::Lambda::Function	CREATE_COMPLETE	-
BastionHost	oQa0l0s17359649c5d	AWS::EC2::Instance	CREATE_COMPLETE	-
DBSubnetGroup	aws-architecture-project-dbsubnetgroup-kb0qgeusas	AWS::RDS::DBSubnetGroup	CREATE_COMPLETE	-
RDSSubnetGroup	aws-architecture-project-rdssubnetgroup-9g70ywv9j8	AWS::RDS::DBSubnetGroup	CREATE_COMPLETE	-
S3Bucket	project-bucket-730353536250	AWS::S3::Bucket	CREATE_COMPLETE	-
WebServerInstance	i-0c6cf43e3fe2518d	AWS::EC2::Instance	CREATE_COMPLETE	-

- Deploy a Web Application on EC2

Once the resources are created, SSH into the bastion host

```
C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4>ssh -i mykey.pem ec2-user@3.82.11.44
  _      #
 / \_  #####      Amazon Linux 2023
 ~~ \_#####\
 ~~  \###|
 ~~   \#/  ___  https://aws.amazon.com/linux/amazon-linux-2023
 ~~    \~' ,-'>
     ~~~   /
     ~~_.-./-
       _/_/
       _/m/
Last login: Wed May 14 20:02:25 2025 from 130.85.59.146
```

Copy the key

```
C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4>scp -i mykey.pem mykey.pem ec2-user@3.82.11.44:~/ssh/mykey.pem  
mykey.pem
```

From bastion, SSH to web server

Update system, Install Apache, Start and enable Apache

```
[ec2-user@ip-10-0-2-146:~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-10-0-2-146:~]$ sudo yum install -y httpd
Last metadata expiration check: 0:00:09 ago on Wed May 14 20:10:01 2025.
Dependencies resolved.
=====
Package           Architecture      Version       Repository    Size
=====
Installing:
  httpd           x86_64          2.4.62-1.amzn2023
Installing dependencies:
  apr             x86_64          1.7.5-1.amzn2023.0.4
  apr-util        x86_64          1.6.3-1.amzn2023.0.1
  generic-logos-httd
  httpd-core     noarch          18.0.0-12.amzn2023.0.3
  httpd-filesystem
  httpd-tools    x86_64          2.4.62-1.amzn2023
  libbrotli      x86_64          2.4.62-1.amzn2023
  mailcap         noarch          1.0.9-4.amzn2023.0.2
  apr-util-openssl
  mod_http2      x86_64          2.0.27-1.amzn2023.0.3
  mod_lua         x86_64          2.4.62-1.amzn2023
=====
Transaction Summary
=====
Install 12 Packages
```

```
Total download size: 2.3 M
Installed size: 6.9 M
Downloading Packages:
(1/12): apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64.rpm      364 kB/s | 17 kB   00:00
(2/12): apr-1.7.5-1.amzn2023.0.4.x86_64.rpm                  2.3 MB/s | 129 kB   00:00
(3/12): apr-util-1.6.3-1.amzn2023.0.1.x86_64.rpm              1.6 MB/s | 98 kB   00:00
(4/12): generic-logos-httd-18.0.0-12.amzn2023.0.3.noarch.rpm  851 kB/s | 19 kB   00:00
(5/12): httpd-2.4.62-1.amzn2023.x86_64.rpm                 2.2 MB/s | 48 kB   00:00
(6/12): httpd-core-2.4.62-1.amzn2023.x86_64.rpm              24 MB/s | 1.4 MB   00:00
(7/12): httpd-filesystem-2.4.62-1.amzn2023.noarch.rpm        283 kB/s | 14 kB   00:00
(8/12): httpd-tools-2.4.62-1.amzn2023.x86_64.rpm            1.7 MB/s | 81 kB   00:00
(9/12): mailcap-2.1.49-3.amzn2023.0.3.noarch.rpm            993 kB/s | 33 kB   00:00
(10/12): libbrotli-1.0.9-4.amzn2023.0.2.x86_64.rpm          7.5 MB/s | 315 kB   00:00
(11/12): mod_http2-2.0.27-1.amzn2023.0.3.x86_64.rpm        3.5 MB/s | 166 kB   00:00
(12/12): mod_lua-2.4.62-1.amzn2023.x86_64.rpm               2.5 MB/s | 61 kB   00:00
=====
Total                                         11 MB/s | 2.3 MB   00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing :                                                 1/1
  Installing : apr-1.7.5-1.amzn2023.0.4.x86_64                1/12
  Installing : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64  2/12
  Installing : apr-util-1.6.3-1.amzn2023.0.1.x86_64          3/12
  Installing : mailcap-2.1.49-3.amzn2023.0.3.noarch          4/12
  Installing : httpd-tools-2.4.62-1.amzn2023.x86_64          5/12
  Installing : libbrotli-1.0.9-4.amzn2023.0.2.x86_64          6/12
```

```

ec2-user@ip-10-0-2-146:~ - + 
Installing : mod_lua-2.4.62-1.amzn2023.x86_64 10/12
Installing : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 11/12
Installing : httpd-2.4.62-1.amzn2023.x86_64 12/12
Running scriptlet: httpd-2.4.62-1.amzn2023.x86_64 12/12
Verifying   : apr-1.7.5-1.amzn2023.0.4.x86_64 1/12
Verifying   : apr-util-1.6.3-1.amzn2023.0.1.x86_64 2/12
Verifying   : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64 3/12
Verifying   : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 4/12
Verifying   : httpd-2.4.62-1.amzn2023.x86_64 5/12
Verifying   : httpd-core-2.4.62-1.amzn2023.x86_64 6/12
Verifying   : httpd-filesystem-2.4.62-1.amzn2023.noarch 7/12
Verifying   : httpd-tools-2.4.62-1.amzn2023.x86_64 8/12
Verifying   : libbrotli-1.0.9-4.amzn2023.0.2.x86_64 9/12
Verifying   : mailcap-2.1.49-3.amzn2023.0.3.noarch 10/12
Verifying   : mod_http2-2.0.27-1.amzn2023.0.3.x86_64 11/12
Verifying   : mod_lua-2.4.62-1.amzn2023.x86_64 12/12

Installed:
apr-1.7.5-1.amzn2023.0.4.x86_64
apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
httpd-2.4.62-1.amzn2023.x86_64
httpd-filesystem-2.4.62-1.amzn2023.noarch
libbrotli-1.0.9-4.amzn2023.0.2.x86_64
mod_http2-2.0.27-1.amzn2023.0.3.x86_64
mod_lua-2.4.62-1.amzn2023.x86_64

Complete!
[ec2-user@ip-10-0-2-146 ~]$ sudo systemctl start httpd
[ec2-user@ip-10-0-2-146 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-10-0-2-146 ~]$ 

```

Create a simple HTML file and check if it is working locally

```

Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/
[ec2-user@ip-10-0-2-146 ~]$ # Create a simpler HTML file
sudo bash -c 'cat > /var/www/html/index.html << EOF
<!DOCTYPE html>
<html>
<head>
    <title>AWS Project</title>
</head>
<body>
    <h1>Hello from AWS!</h1>
    <p>This is a web application deployed on an EC2 instance.</p>
    <p>Server ID: $(hostname)</p>
</body>
</html>
EOF'
[ec2-user@ip-10-0-2-146 ~]$ curl http://localhost
<!DOCTYPE html>
<html>
<head>
    <title>AWS Project</title>
</head>
<body>
    <h1>Hello from AWS!</h1>
    <p>This is a web application deployed on an EC2 instance.</p>
    <p>Server ID: ip-10-0-2-146.ec2.internal</p>
</body>
</html>

```

Check to verify the access through the ALB



Hello from AWS!

This is a web application deployed on an EC2 instance.

Server ID: ip-10-0-2-146.ec2.internal

- Configure the Database for the Web Application

Install MySQL client

```
[ec2-user@ip-10-0-2-146:~]$ sudo yum search mysql | grep client
Last metadata expiration check: 0:12:17 ago on Wed May 14 20:10:01 2025.
mariadb1011-client-utils.x86_64 : Non-essential client utilities for MariaDB/MySQL applications
[ec2-user@ip-10-0-2-146:~]$ sudo yum install -y mariadb101-client
Last metadata expiration check: 0:12:56 ago on Wed May 14 20:10:01 2025.
No match for argument: mariadb101-client
Error: Unable to find a match: mariadb101-client
[ec2-user@ip-10-0-2-146:~]$ sudo yum install -y mariadb1011-client-utils
Last metadata expiration check: 0:13:49 ago on Wed May 14 20:10:01 2025.
Dependencies resolved.
=====
 Package           Architecture   Version        Repository      Size
=====
Installing:
 mariadb1011-client-utils.x86_64      3:10.11.11-1.amzn2023.0.1      amazonlinux    42 k
Installing dependencies:
 mariadb-connector-c.x86_64            3.3.10-1.amzn2023.0.1        amazonlinux    211 k
 mariadb-connector-c-config.noarch     3.3.10-1.amzn2023.0.1        amazonlinux    9.9 k
 mariadb1011.x86_64                   3:10.11.11-1.amzn2023.0.1      amazonlinux    1.6 M
 mariadb1011-common.x86_64            3:10.11.11-1.amzn2023.0.1      amazonlinux    31 k
 perl-B.x86_64                      1.80-477.amzn2023.0.6        amazonlinux    179 k
 perl-DBI.x86_64                     1.643-7.amzn2023.0.3        amazonlinux    700 k
 perl-Data-Dumper.x86_64              2.174-460.amzn2023.0.2        amazonlinux    55 k
 perl-FileHandle.noarch               2.03-477.amzn2023.0.6        amazonlinux    16 k
 perl-Math-BigInt.noarch              1:1.9998.39-2.amzn2023.0.2      amazonlinux    202 k
 perl-Math-BigRat.noarch              0.2624-500.amzn2023.0.2        amazonlinux    42 k
 perl-Math-Complex.noarch             1.59-477.amzn2023.0.6        amazonlinux    47 k
 perl-Sys-Hostname.x86_64             1.23-477.amzn2023.0.6        amazonlinux    18 k
 perl-base.noarch                    2.27-477.amzn2023.0.6        amazonlinux    17 k
```

```

ec2-user@ip-10-0-2-146:~ - + 
Transaction Summary
=====
Install 14 Packages

Total download size: 3.1 M
Installed size: 27 M
Downloading Packages:
(1/14): mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch.rpm          275 kB/s | 9.9 kB   00:00
(2/14): mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64.rpm                  4.5 MB/s | 211 kB   00:00
(3/14): mariadb1011-client-utils-10.11.11-1.amzn2023.0.1.x86_64.rpm           800 kB/s | 42 kB    00:00
(4/14): perl-B-1.80-477.amzn2023.0.6.x86_64.rpm                            5.6 MB/s | 179 kB   00:00
(5/14): perl-DBI-1.643-7.amzn2023.0.3.x86_64.rpm                          18 MB/s | 700 kB   00:00
(6/14): mariadb1011-common-10.11.11-1.amzn2023.0.1.x86_64.rpm              266 kB/s | 31 kB    00:00
(7/14): perl-Data-Dumper-2.174-460.amzn2023.0.2.x86_64.rpm                2.5 MB/s | 55 kB    00:00
(8/14): mariadb1011-10.11.11-1.amzn2023.0.1.x86_64.rpm                     7.9 MB/s | 1.6 MB   00:00
(9/14): perl-FileHandle-2.03-477.amzn2023.0.6.noarch.rpm                   430 kB/s | 16 kB    00:00
(10/14): perl-Math-BigRat-0.2624-500.amzn2023.0.2.noarch.rpm               1.4 MB/s | 42 kB    00:00
(11/14): perl-Math-BigInt-1.9998.39-2.amzn2023.0.2.noarch.rpm             2.7 MB/s | 202 kB   00:00
(12/14): perl-Math-Complex-1.59-477.amzn2023.0.6.noarch.rpm                 850 kB/s | 47 kB    00:00
(13/14): perl-Sys-Hostname-1.23-477.amzn2023.0.6.x86_64.rpm                594 kB/s | 18 kB    00:00
(14/14): perl-base-2.27-477.amzn2023.0.6.noarch.rpm                         806 kB/s | 17 kB    00:00
-----
Total                                         10 MB/s | 3.1 MB   00:00

Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing :                                         1/1

```

```

ec2-user@ip-10-0-2-146:~ - + 
Installing      : perl-base-2.27-477.amzn2023.0.6.noarch                      9/14
Installing      : perl-Sys-Hostname-1.23-477.amzn2023.0.6.x86_64                  10/14
Installing      : perl-FileHandle-2.03-477.amzn2023.0.6.noarch                  11/14
Installing      : perl-DBI-1.643-7.amzn2023.0.3.x86_64                      12/14
Installing      : mariadb1011-client-utils-3:10.11.11-1.amzn2023.0.1.x86_64       13/14
Installing      : mariadb1011-3:10.11.11-1.amzn2023.0.1.x86_64                  14/14
Running scriptlet: mariadb1011-3:10.11.11-1.amzn2023.0.1.x86_64            14/14
Verifying       : mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64                  1/14
Verifying       : mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch        2/14
Verifying       : mariadb1011-3:10.11.11-1.amzn2023.0.1.x86_64                  3/14
Verifying       : mariadb1011-client-utils-3:10.11.11-1.amzn2023.0.1.x86_64       4/14
Verifying       : mariadb1011-common-3:10.11.11-1.amzn2023.0.1.x86_64            5/14
Verifying       : perl-B-1.80-477.amzn2023.0.6.x86_64                      6/14
Verifying       : perl-DBI-1.643-7.amzn2023.0.3.x86_64                      7/14
Verifying       : perl-Data-Dumper-2.174-460.amzn2023.0.2.x86_64                  8/14
Verifying       : perl-FileHandle-2.03-477.amzn2023.0.6.noarch                  9/14
Verifying       : perl-Math-BigInt-1:1.9998.39-2.amzn2023.0.2.noarch            10/14
Verifying       : perl-Math-BigRat-0.2624-500.amzn2023.0.2.noarch                11/14
Verifying       : perl-Math-Complex-1.59-477.amzn2023.0.6.noarch                12/14
Verifying       : perl-Sys-Hostname-1.23-477.amzn2023.0.6.x86_64                  13/14
Verifying       : perl-base-2.27-477.amzn2023.0.6.noarch                      14/14
Installed:
mariadb-connector-c-3.3.10-1.amzn2023.0.1.x86_64          mariadb-connector-c-config-3.3.10-1.amzn2023.0.1.noarch
mariadb1011-3:10.11.11-1.amzn2023.0.1.x86_64          mariadb1011-client-utils-3:10.11.11-1.amzn2023.0.1.x86_64
mariadb1011-common-3:10.11.11-1.amzn2023.0.1.x86_64     perl-B-1.80-477.amzn2023.0.6.x86_64
perl-DBI-1.643-7.amzn2023.0.3.x86_64                  perl-Data-Dumper-2.174-460.amzn2023.0.2.x86_64
perl-FileHandle-2.03-477.amzn2023.0.6.noarch          perl-Math-BigInt-1:1.9998.39-2.amzn2023.0.2.noarch
perl-Math-BigRat-0.2624-500.amzn2023.0.2.noarch        perl-Math-Complex-1.59-477.amzn2023.0.6.noarch
perl-Sys-Hostname-1.23-477.amzn2023.0.6.x86_64          perl-base-2.27-477.amzn2023.0.6.noarch

```

Connect to the RDS database

```

Complete!
[ec2-user@ip-10-0-2-146 ~]$ mysql -h aws-architecture-project-rdsinstance-agd7oyqwyup9.cchumamc2mjf.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
ERROR 1045 (28000): Access denied for user 'admin'@'10.0.2.146' (using password: YES)
[ec2-user@ip-10-0-2-146 ~]$ mysql -h aws-architecture-project-rdsinstance-agd7oyqwyup9.cchumamc2mjf.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 33
Server version: 8.0.41 Source distribution

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

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

```

Create a simple users table

```
MySQL [(none)]> USE projectdb;
Database changed
MySQL [projectdb]> CREATE TABLE users (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     username VARCHAR(50) NOT NULL,
    ->     email VARCHAR(100) NOT NULL,
    ->     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
-> );
```

Insert some sample data

```
MySQL [projectdb]> INSERT INTO users (username, email) VALUES
    ->     ('user1', 'user1@example.com'),
    ->     ('user2', 'user2@example.com');
Query OK, 2 rows affected (0.007 sec)
Records: 2  Duplicates: 0  Warnings: 0

MySQL [projectdb]> SELECT * FROM users;
+----+-----+-----+-----+
| id | username | email           | created_at      |
+----+-----+-----+-----+
| 1  | user1   | user1@example.com | 2025-05-14 20:27:11 |
| 2  | user2   | user2@example.com | 2025-05-14 20:27:11 |
+----+-----+-----+-----+
2 rows in set (0.001 sec)
```

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

Create an AMI from your configured web server

The screenshot shows the 'Create image' page in the AWS Management Console. The instance selected is 'i-0x6ecf45e3fe2518d (web-server)'. The 'Image name' field is filled with 'web-server-ami'. The 'Reboot instance' checkbox is checked. Under 'Instance volumes', there is one volume listed: '/dev/xvda' (EBS, 8 GB, EBS General Purpose SSD, Throughput: 3000, Delete on termination: checked, Encrypted: unchecked). A note states: 'During the image creation process, Amazon EC2 creates a snapshot of each of the above volumes.' Under 'Tags - optional', there are two options: 'Tag image and snapshots together' (selected) and 'Tag image and snapshots separately'. A note at the bottom says: 'No tags associated with the resource. You can add up to 50 more tags.' At the bottom right, there are links for 'Privacy', 'Terms', and 'Cookie preferences'.

The screenshot shows the AWS EC2 console with the 'AMIs' section selected. A single AMI, 'web-server-ami', is listed in the table. The table columns include Name, AMI name, AMI ID, Source, Owner, Visibility, Status, Creation date, Platform, and Root device type. The AMI details show it was created by the user ('Owned by me'), has the ID 'ami-0c9f98870ad01781f', and is a private image.

Create a Launch Template

The screenshot shows the 'Create launch template' wizard. The first step, 'Launch template name and description', is completed. The AMI 'web-server-ami' is selected as the software image. The instance type is 't2.micro'. The storage is '1 volume(s) - 8 GB'. The wizard also includes sections for 'Auto Scaling guidance' and 'Template tags'. The second step, 'Launch template contents', is partially visible at the bottom.

Screenshot of the AWS CloudShell interface showing the creation of a launch template for an EC2 instance.

Network settings

- Subnet**: Info. Set to "Don't include in launch template". Option to "Create new subnet".
- Firewall (security group)**: Info. Set to "Select existing security group". Option to "Create security group".
- Security groups**: Info. Set to "Select security groups". Option to "Compare security group rules".

Storage (volumes)

- EBS Volumes**: Volume 1 (AMI Root): 8 GiB, EBS, General purpose SSD (gp3), 3000 IOPS. AMI Volumes are not included in the template unless modified.
- Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage.
- Option to "Add new volume".

Resource tags

- No resource tags are currently included in this template. Add a resource tag to include it in the launch template.
- Option to "Add new tag".
- You can add up to 50 more tags.

Summary

- Software Image (AMI)**: web-server-ami am-0cb98870ad01791f
- Virtual server type (instance type)**: t2.micro
- Firewall (security group)**: web-sg
- Storage (volumes)**: 1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage for t3.micro where t2.micro isn't available when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.

Create launch template

Success: Successfully created web-server-launch-template@{1-01e9eb0029dcdfb}.

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
- 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**: 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

Create an Auto Scaling Group

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1

- Choose launch template
- Choose instance launch options
- Step 5 - optional
- Integrate with other services
- Step 4 - optional
- Configure group size and scaling
- Step 5 - optional
- Add notifications
- Step 6 - optional
- Add tags
- Step 7
- Review

Choose launch template Info
Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

Name
Auto Scaling group name
Enter a name to identify the group.

Must be unique to this account in the current Region and no more than 255 characters.

Launch template Info
For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.

Launch template
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

Create a launch template Info

Version

Description
Version 1

AMI ID
ami-0cd98870ad01781f

Key pair name
mykey

Additional details

Storage (volumes)

Date created
Wed May 14 2025 16:59:40 GMT-0400 (Eastern Daylight Time)

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1 - optional

- Integrate with other services
- Step 4 - optional
- Configure group size and scaling
- Step 5 - optional
- Add notifications
- Step 6 - optional
- Add tags
- Step 7
- Review

Instance type requirements Info
You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

Launch template

It-01efeb06b29dcbe

Version

Description
Version 1

Instance type
t2.micro

Network Info
For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC
Choose the VPC that defines the virtual network for your Auto Scaling group.

10.0.0.0/16

Create a VPC Info

Availability Zones and subnets
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

us-east-1a | subnet-0eef15302af24b3bc (private-subnet-1)
10.0.2.0/24

us-east-1b | subnet-01d5df3b126a1bb0a (private-subnet-2)
10.0.4.0/24

Create a subnet Info

Availability Zone distribution - new
Auto Scaling automatically balances instances across Availability Zones. If launch failures occur in a zone, select a strategy.

Balanced best effort
If launches fail in one Availability Zone, Auto Scaling will attempt to launch in another healthy Availability Zone.

Balanced only
If launches fail in one Availability Zone, Auto Scaling will continue to attempt to launch in the unhealthy Availability Zone to preserve balanced distribution.

Next Info

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

aws Search [Alt+S]

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1: Choose launch template
Step 2: Choose instance launch options
Step 3 - optional: **Integrate with other services**
Step 4 - optional: Configure group size and scaling
Step 5 - optional: Add notifications
Step 6 - optional: Add tags
Step 7: Review

Integrate with other services - optional Info
Use a load balancer to distribute network traffic across multiple servers. Enable service-to-service communications with VPC Lattice. Shift resources away from impaired Availability Zones with zonal shift.
You can also customize health check replacements and monitoring.

Load balancing Info
Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

No load balancer
Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer
Choose from your existing load balancers.

Attach to a new load balancer
Quickly create a basic load balancer to attach to your Auto Scaling group.

Attach to an existing load balancer
Select the load balancers that you want to attach to your Auto Scaling group.

Choose from your load balancer target groups
This option allows you to attach Application, Network, or Gateway Load Balancers.

Choose from Classic Load Balancers

Existing load balancer target groups
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups

web-tg | HTTP
Application Load Balancer: web-all

VPC Lattice integration options Info
To improve networking capabilities and scalability, integrate your Auto Scaling group with VPC Lattice. VPC Lattice facilitates communications between AWS services and helps you connect and manage your applications across compute services in AWS.

Select VPC Lattice service to attach

No VPC Lattice service
VPC Lattice will not manage your Auto Scaling group's network access and connectivity with other services.

Attach to VPC Lattice service
Incoming requests associated with specified VPC Lattice target groups will be routed to your Auto Scaling group.

Create new VPC Lattice service

Application Recovery Controller (ARC) zonal shift - new Info

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

CloudShell [Feedback](#)

aws Search [Alt+S]

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1: Choose launch template
Step 2: Choose instance launch options
Step 3 - optional: **Integrate with other services**
Step 4 - optional: **Configure group size and scaling**
Step 5 - optional: Add notifications
Step 6 - optional: Add tags
Step 7: Review

Configure group size and scaling - optional Info
Define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.

Group size Info
Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

Desired capacity type
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Units [number of instances]

Desired capacity
Specify your group size.

Scaling Info
You can resize your Auto Scaling group manually or automatically to meet changes in demand.

Scaling limits
Set limits on how much your desired capacity can be increased or decreased.

Min desired capacity <input type="text" value="1"/>	Max desired capacity <input type="text" value="4"/>
---	---

Equal or less than desired capacity

Automatic scaling - optional
Choose whether to use a target tracking policy Info
You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

No scaling policies
Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

Target tracking scaling policy
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

Scaling policy name

Metric type Info
Monitored metric that determines if resource utilization is too low or high. If using EC2 metrics, consider enabling detailed monitoring for better scaling performance.

Average CPU utilization

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

CloudShell [Feedback](#)

Metric type: Average CPU utilization

Target value: 70

Instance warmup: 300 seconds

Instance maintenance policy: No policy

Capacity Reservation preference: On-Demand

Additional capacity settings: Desired capacity: 2, Min: 1, Max: 4, Availability Zones: us-east-1a, us-east-1b

Verify Auto Scaling is Working

There should be 2 instances in the "InService" state (based on your desired capacity)

Instances

- Instances
- Instance Types
- Launch Templates
- Spot Requests
- Savings Plans
- Reserved Instances
- Dedicated Hosts
- Capacity Reservations

Images

- AMIs
- AMI Catalog

Elastic Block Store

- Volumes
- Snapshots
- Lifecycle Manager

Network & Security

- Security Groups
- Elastic IPs
- Placement Groups
- Key Pairs
- Network Interfaces

Load Balancing

- Load Balancers
- Target Groups
- Trust Stores

Auto Scaling

- Auto Scaling Groups

Settings

Capacity overview

Desired capacity: 2 Scaling limits (Min - Max): 1 - 4 Desired capacity type: Units (number of instances)

Date created: Wed May 14 2025 17:07:08 GMT-0400 (Eastern Daylight Time)

Instances (2)

Instance ID	Lifecycle	Instance type	Weighted capacity	Launch template...	Availability Zone	Health status	Protected from
i-0c1e1559de0ea50...	InService	t2.micro	-	web-server-launch-temp...	us-east-1a	Healthy	
i-0f7fce2a57daa42...	InService	t2.micro	-	web-server-launch-temp...	us-east-1b	Healthy	

Lifecycle hooks (0) Info

No lifecycle hooks are currently configured.

Actions

Details

arn:aws:celticloadbalancing:us-east-1:73035536250:targetgroup/web-tg/2ae03111d23b47

Target type: Instance **Protocol : Port**: HTTP : 80 **Protocol version**: HTTP1

IP address type: IPv4 **Load balancer**: web-alb

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
3	3	0	0	0	0
	0 Anomalous				

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

Targets

Registered targets (3) Info

Target groups route to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Actions

Targets

Monitoring

Health checks

Attributes

Tags

Details

arn:aws:celticloadbalancing:us-east-1:73035536250:targetgroup/web-tg/2ae03111d23b47

Target type: Instance **Protocol : Port**: HTTP : 80 **Protocol version**: HTTP1

IP address type: IPv4 **Load balancer**: web-alb

Instance ID	Name	Port	Zone	Health status	Health status details	Administrative override	Override details
i-0f7fce2a57daa42...	web-server-as...	80	us-east-1b (us...)	Healthy	-	<input type="radio"/> No override	No override is currently active on target
i-0c1e1559de0ea50...	web-server-as...	80	us-east-1a (us...)	Healthy	-	<input type="radio"/> No override	No override is currently active on target
i-0dec143e5fe251bd...	web-server	80	us-east-1a (us...)	Healthy	-	<input type="radio"/> No override	No override is currently active on target

Verify the Auto Scaling Group is properly integrated with the Target Group

Instances

- Instances
- Instance Types
- Launch Templates
- Spot Requests
- Savings Plans
- Reserved Instances
- Dedicated Hosts
- Capacity Reservations

Images

- AMIs
- AMI Catalog

Elastic Block Store

- Volumes
- Snapshots
- Lifecycle Manager

Network & Security

- Security Groups
- Elastic IPs
- Placement Groups
- Key Pairs
- Network Interfaces

Load Balancing

- Load Balancers
- Target Groups
- Trust Stores

Auto Scaling

- Auto Scaling Groups

Settings

Targets

Monitoring

Health checks

Attributes

Tags

Details

arn:aws:celticloadbalancing:us-east-1:73035536250:targetgroup/web-tg/2ae03111d23b47

Target type: Instance **Protocol : Port**: HTTP : 80 **Protocol version**: HTTP1

IP address type: IPv4 **Load balancer**: web-alb

Registered targets (3) Info

Target groups route to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Anomaly mitigation: Not applicable

Actions

Targets

Monitoring

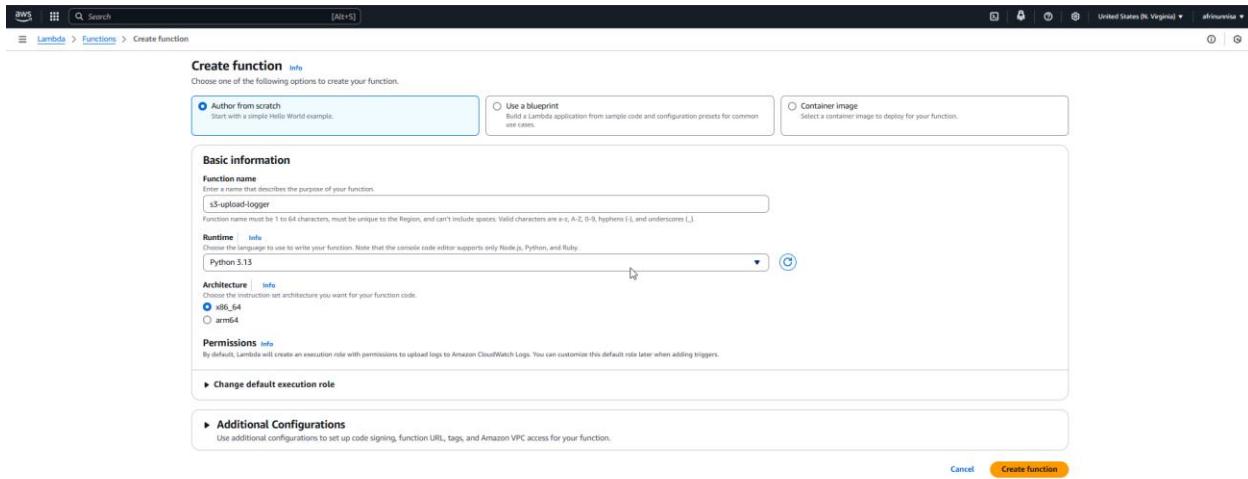
Health checks

Attributes

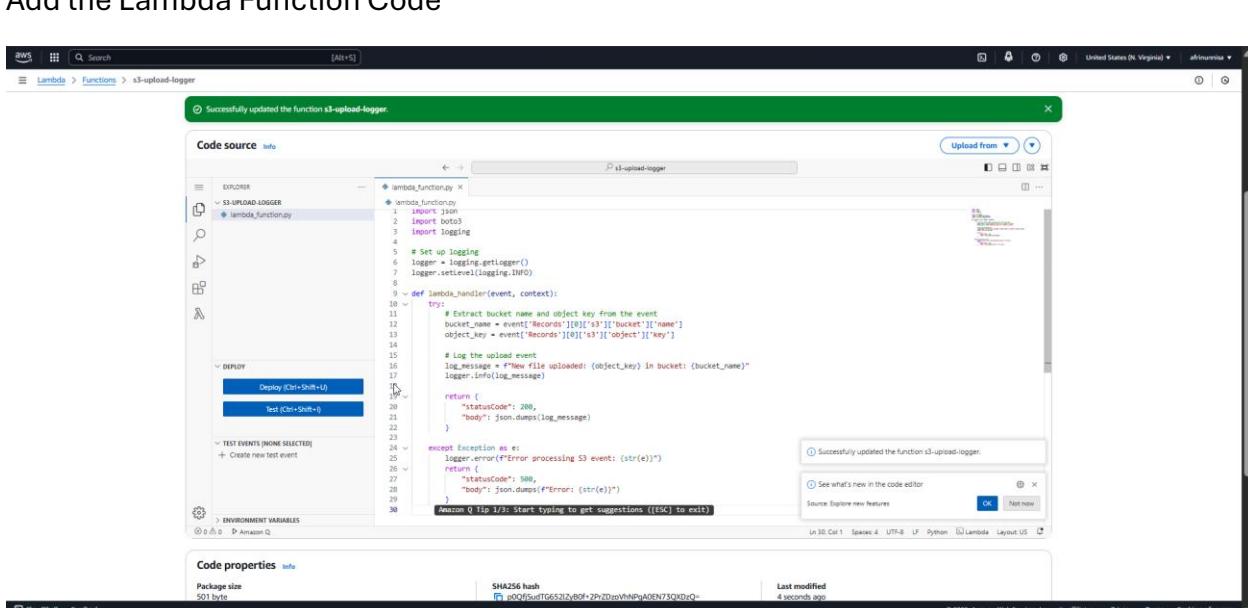
Tags

B. AWS Lambda for Logging S3 Uploads

Created "s3-upload-logger" function in AWS Console



Add the Lambda Function Code



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

The screenshot shows the 'Add trigger' configuration page for an AWS Lambda function. The 'Trigger configuration' section is set to 'S3'. A dropdown menu shows 'aws asynchronous storage'. The 'Bucket' field contains 's3://project-bucket-730355336250'. The 'Event types' dropdown is set to 'All object create events'. There are optional fields for 'Prefix' (e.g., 'images/') and 'Suffix' (e.g., '.jpg'). A note about recursive invocation is present, with a checkbox indicating acknowledgement. A note states that Lambda will add necessary permissions for AWS S3 to invoke the Lambda function. At the bottom are 'Cancel' and 'Add' buttons.

Verify the Setup by Uploading a File to S3

The screenshot shows the 'Upload' interface for an AWS S3 bucket. The 'Destination' is set to 's3://project-bucket-730355336250'. A file named 'myfile.txt' is selected for upload. The 'Permissions' section indicates public access. The 'Properties' section specifies 'text/plain' as the type and '8.0 B' as the size. At the bottom are 'Cancel' and 'Upload' buttons.

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

The screenshot shows the AWS CloudWatch Log Events interface. The left sidebar navigation includes CloudWatch, AI Operations, Metrics, X-Ray traces, Events, Application Signals, Network Monitoring, and Insights. The main content area displays log events for a specific Lambda function. The logs show the function starting, executing code, uploading files to S3, and reporting memory usage. The interface includes a search bar, filter bar, and various time range and display options.

C. Interaction with AWS

Use AWS Console to verify infrastructure deployment.

VPC and subnets

The screenshot shows the AWS VPC dashboard for a specific VPC. The left sidebar navigation includes VPC dashboard, EC2 Global View, Virtual private cloud, Security, PrivateLink and Lattice, and DNS. The main content area displays the VPC details and a resource map. The VPC details include the VPC ID, state (Available), and various configuration settings like Block Public Access, DHCP option set, IPv4 CIDR, and DNS hostnames. The resource map provides a visual representation of the network components, including subnets (4), route tables (3), and network connections (2).

VPC dashboard

Subnets (8) Info

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID	Available IPv4 addresses
-	subnet-024f0b1ce6844399c	Available	vpc-072d9b5635102:306	Off	172.31.0.0/24	-	-	251
private-subnet-1	subnet-0e1582af24b3cb	Available	vpc-0235f5ea6403aa4a proj...	Off	10.0.2.0/24	-	-	249
public-subnet-1	subnet-0525e2b2cd72b8a2c	Available	vpc-0235f5ea6403aa4a proj...	Off	10.0.0.0/24	-	-	248
public-subnet-2	subnet-07a3fcf6feefca0	Available	vpc-0235f5ea6403aa4a proj...	Off	10.0.3.0/24	-	-	250
-	subnet-02150a11f204faed	Available	vpc-0235f5ea6403aa4a proj...	Off	172.31.0.0/24	-	-	251
-	subnet-0f809c6ed186c094d	Available	vpc-072d9b5635102:306	Off	172.31.16.0/20	-	-	4091
private-subnet-2	subnet-011d7b0126a1bb0a	Available	vpc-0235f5ea6403aa4a proj...	Off	10.0.4.0/24	-	-	250
Public-Subnet-1	subnet-0a20377e7d1e1a	Available	vpc-072d9b5635102:306	Off	172.31.0.0/24	-	-	251

Select a subnet

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Security Groups

VPC dashboard

Security Groups (8) Info

Name	Security group ID	Security group name	VPC ID	Description	Owner	Inbound rules count
-	sg-0ff2676574416aa27	default	xoc-072d9b5635102:306	default VPC security group	730353536250	1 Permission entry
-	sg-0ab9bd05cb2cd6f1	TemplateSG	xoc-072d9b5635102:306	Allow SSH and HTTP	730353536250	2 Permission entries
db-sg	sg-05887d5677a1790b6	db-sg	xoc-0235f5ea6403aa4a proj...	Security group for database	730353536250	2 Permission entries
bastion-sg	sg-0885e2db57379918	bastion-sg	xoc-0235f5ea6403aa4a proj...	Security group for bastion host	730353536250	1 Permission entry
alb-sg	sg-0043f70575a03537e	alb-sg	xoc-0235f5ea6403aa4a proj...	Security group for application load balancer	730353536250	2 Permission entries
web-sg	sg-0595c493b11a45b0	web-sg	xoc-0235f5ea6403aa4a proj...	Security group for web servers	730353536250	2 Permission entries
-	sg-097d7d19bc001979c	MyTemplateSG	xoc-072d9b5635102:306	Allow SSH and HTTP from anywhere	730353536250	2 Permission entries
-	sg-00752541312bdc7c	default	xoc-0235f5ea6403aa4a proj...	default VPC security group	730353536250	1 Permission entry

Select a security group

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EC2 instances

The screenshot shows the AWS EC2 Instances page. The left sidebar includes sections for Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, Load Balancing, Load Balancers, Target Groups, and Trust Stores. The main content area displays a table of instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4 IP, Elastic IP, IPv6 IPs, Monitoring, and Security group. Three instances are listed: web-server-asm-instance (t2.micro, running, 2/2 checks passed, us-east-1b, -), web-server (t2.micro, running, 2/2 checks passed, us-east-1a, -), and bastion-host (t2.micro, running, 2/2 checks passed, us-east-1a, ec2-3-82-11-44.compute.amazonaws.com, 3.82.11.44). A modal window titled "Select an instance" is open at the bottom.

RDS instance

The screenshot shows the AWS Aurora and RDS Database details page for the instance "aws-architecture-project-rdsinstance-agd7oyqwyup9". The left sidebar includes sections for Dashboard, Databases (selected), Query editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Zero-ETL integrations, Events, Event subscriptions, and Recommendations. The main content area is divided into sections: Summary, Connectivity & security, Connected compute resources, and Actions. The Summary section shows DB identifier, Status (Available), Role (Instance), Current activity (0 connections), Engine (MySQL Community), and Region & AZ (us-east-1a). The Connectivity & security section includes tabs for Endpoint & port, Networking, and Security. It lists the endpoint (aws-architecture-project-rdsinstance-agd7oyqwyup9.chumamc2.mf.us-east-1.rds.amazonaws.com), port (3306), VPC (project-vpc), subnet group (aws-architecture-project-dbsubnetgroup-kbloghgsaua), subnets (subnet-01d5a1f8127a5a0d8a, subnet-0e15382af2403fbch), and network type (IPv4). The Security section shows VPC security groups (db-sg-0687bd5677a179096, Active), Publicly accessible (No), Certificate authority (Info, rds-cs-rs2048-q1), and Certificate authority date (May 25, 2021, 19:34 UTC-04:00). The Connected compute resources section shows a table with columns for Resource identifier, Resource type, Availability Zone, VPC security group, Compute resource security group, and Actions. A modal window titled "aws-architecture-project-rdsinstance-agd7oyqwyup9" is open at the top right.

S3 bucket

Amazon S3

Objects (1)

Name	Type	Last modified	Size	Storage class
myfile.txt	txt	May 14, 2025, 20:47:42 (UTC-04:00)	8.0 B	Standard

Auto scaling group

EC2 > Auto Scaling groups > web-server-asg

Capacity overview

Desired capacity	Scaling limits (Min - Max)	Desired capacity type	Status
1	1 - 4	Units (number of instances)	

Launch template

Launch template	AMI ID	Instance type	Owner
It-01e9e060b29dc0be web-server-launch-template	ami-0c9e98870ad01781f	t2.micro	arn:aws:iam::730335336250:root
Version	Security groups	Security group IDs	Create time
Default	-	sg-00595c492b15a45bb	Wed May 14 2025 17:07:08 GMT-0400 (Eastern Daylight Time)
Description	Storage (volumes)	Key pair name	Request Spot Instances
Version 1	-	mykey	No

Network

Availability Zones	Subnet ID	Availability Zone distribution
us-east-1a, us-east-1b	subnet-0e15382a724b3bc9, subnet-01d5df3b126a88b60a	Balanced best effort

Instance type requirements

Your Auto Scaling group adheres to the launch template for purchase option and instance type.

Load Balancer

CloudFront Distribution Details:

- Name:** web-alb
- Status:** Active
- VPC:** vpc-0235f5ea0481aaeab
- Availability Zones:** subnets-052c2c7ef072ba2cf (us-east-1a), subnets-07a3fcfc0ee1ca0d (us-east-1b)
- Date created:** May 14, 2025, 15:50 (UTC-04:00)
- DNS name info:** web-alb-1987170899.us-east-1.cloudfront.net (A Record)

Listeners and rules (1) info

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store
HTTP:80	Forward to target group • web-tg (100%) • Target group stickiness: Off	1 rule	ARN	Not applicable	Not applicable	Not applicable	Not applicable

Lambda Function

Functions (2)

Function name	Description	Package type	Runtime	Last modified
s3-upload-logger	-	Zip	Python 3.15	18 minutes ago
basic-function	-	Zip	Python 3.9	5 hours ago

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

Interacting with EC2:

List all EC2 instances

```
Command Prompt - aws ec2 × + ▾

C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4>aws ec2 describe-instances --query "Reservations[*].Instances[*].[InstanceId, State.Name, InstanceType, PrivateIpAddress, PublicIpAddress]"
[{"InstanceId": "i-0f7cf6e2a457daa42", "State.Name": "running", "InstanceType": "t2.micro", "PrivateIpAddress": "10.0.4.180", "PublicIpAddress": null}, {"InstanceId": "i-0c6ccf43e3fe2518d", "State.Name": "running", "InstanceType": "t2.micro", "PrivateIpAddress": "10.0.2.146", "PublicIpAddress": null}, {"InstanceId": "i-0aa80a1235f649c5d", "State.Name": "running", "InstanceType": "t2.micro", "PrivateIpAddress": "10.0.1.92", "PublicIpAddress": "3.82.11.44"}]
```

Check the status of the instances

```
Command Prompt - aws ec2 × + ▾

C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4>aws ec2 describe-instance-status --instance-ids i-0c6ccf43e3fe2518d
{
    "InstanceStatuses": [
        {
            "AvailabilityZone": "us-east-1a",
            "InstanceId": "i-0c6ccf43e3fe2518d",
            "InstanceState": {
                "Code": 16,
                "Name": "running"
            },
            "InstanceStatus": {
                "Details": [
                    {
                        "Name": "reachability",
                        "Status": "passed"
                    }
                ],
                "Status": "ok"
            },
            "SystemStatus": {
                "Details": [
                    {
                        "Name": "reachability",
                        "Status": "passed"
                    }
                ],
                "Status": "ok"
            }
        }
    ]
}
```

Interacting with S3:

List all S3 buckets

```
C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4>aws s3 ls
2025-05-14 15:23:09 cf-templates-16qxj0jrpe3ez-us-east-1
2025-05-14 15:50:17 project-bucket-730335336250
```

List contents of the project bucket

```
C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4>aws s3 ls s3://project-bucket-730335336250  
2025-05-14 20:47:42          8 myfile.txt
```

Upload a file to S3

```
C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4>echo "Hello from AWS CLI" > test-cli.txt  
C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4>aws s3 cp test-cli.txt s3://project-bucket-730335336250/test-cli.txt  
upload: ./test-cli.txt to s3://project-bucket-730335336250/test-cli.txt
```

Interacting with Lambda:

List all Lambda functions

```
C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4>aws lambda list-functions --query "Functions[*].[FunctionName, Runtime, Description]" --output table  
-----  
|           ListFunctions           |  
+-----+-----+-----+  
| s3-upload-logger | python3.13 |  
| basic-function  | python3.9  |  
+-----+-----+-----+
```

Get details about logging Lambda function

```
Command Prompt - aws lambda <input>  
C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4>aws lambda get-function --function-name s3-upload-logger  
{  
    "Configuration": {  
        "FunctionName": "s3-upload-logger",  
        "FunctionArn": "arn:aws:lambda:us-east-1:730335336250:function:s3-upload-logger",  
        "Runtime": "python3.13",  
        "Role": "arn:aws:iam::730335336250:role/service-role/s3-upload-logger-role-asqoc0ss",  
        "Handler": "lambda_function.lambda_handler",  
        "CodeSize": 501,  
        "Description": "",  
        "Timeout": 3,  
        "MemorySize": 128,  
        "LastModified": "2025-05-15T00:42:39.000+0000",  
        "CodeSha256": "p0QfjSudTG652lZyB0f+2PrZDzoVhNPgA0EN73QXDzQ=",  
        "Version": "$LATEST",  
        "TracingConfig": {  
            "Mode": "PassThrough"  
        },  
        "RevisionId": "5996d317-2fc2-4d04-96eb-6d9c40d149f4",  
        "State": "Active",  
        "LastUpdateStatus": "Successful",  
        "PackageType": "Zip",  
        "Architectures": [  
            "x86_64"  
        ],  
        "EphemeralStorage": {  
            "Size": 512  
        },  
        "SnapStart": {}  
    }  
}
```

Write Python scripts using Boto3 to:

<https://github.com/NisaAf/aws-architecture-project/tree/main/python>

Create an S3 bucket and upload a file

```

s3_operations.py x
aws-architecture-project > python > s3_operations.py > ...
1 # s3_operations.py - Creates an S3 bucket and uploads a file
2 import boto3
3 import uuid
4
5 # Initialize S3 client
6 s3_client = boto3.client('s3')
7
8 # Create a unique bucket name
9 bucket_name = f'boto3-bucket-{uuid.uuid4().hex[:8]}'
10 print(f"Creating bucket: {bucket_name}")
11
12 # Create the bucket
13 s3_client.create_bucket(Bucket=bucket_name)
14 print(f"Bucket created successfully: {bucket_name}")
15
16 # Create a file to upload
17 file_name = 'boto3-test-file.txt'
18 with open(file_name, 'w') as f:
19     f.write('This file was created and uploaded using Boto3')
20 print(f"Created file: {file_name}")
21
22 # Upload the file
23 s3_client.upload_file(file_name, bucket_name, file_name)
24 print(f"File {file_name} uploaded to {bucket_name}")
25
26 print("S3 operations completed successfully")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\afarin\OneDrive\Documents\Course Work\Sem 4\aws-architecture-project> cd .\python
PS C:\Users\afarin\OneDrive\Documents\Course Work\Sem 4\aws-architecture-project> python .\s3_operations.py
Creating bucket: boto3-bucket-7fdfa68d
Bucket created successfully: boto3-bucket-7fdfa68d
created file: boto3-test-file.txt
File boto3-test-file.txt uploaded to boto3-bucket-7fdfa68d
S3 operations completed successfully
PS C:\Users\afarin\OneDrive\Documents\Course Work\Sem 4\aws-architecture-project> python
aws-architecture-project > main* 0 △ 0 1 file and 0 cells to analyze

In 4, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.9.0 32-bit (backend-ctc5oeO) L

Verify on the console

Amazon S3 > Buckets > boto3-bucket-7fdfa68d

boto3-bucket-7fdfa68d

Name	Type	Last modified	Size	Storage class
boto3-test-file.txt	txt	May 14, 2025, 21:33:52 (UTC-04:00)	46.0 B	Standard

Retrieve EC2 instance metadata

Queries AWS for the metadata of the specified EC2 instance

The screenshot shows a terminal window with two tabs: 's3_operations.py U' and 'ec2_metadata.py U X'. The 'ec2_metadata.py' tab is active, displaying the following Python code:

```
aws-architecture-project > python > ec2_metadata.py > ...
25     print(f"VPC ID: {instance['vpcId']}")
26     print(f"Subnet ID: {instance['SubnetId']}")
27     print(f"State: {instance['State'][ 'Name']} ")
28
29     # Get network information
30     if 'PrivateIpAddress' in instance:
31         print(f"Private IP: {instance['PrivateIpAddress']}")
32     if 'PublicIpAddress' in instance:
33         print(f"Public IP: {instance['PublicIpAddress']}")
34
35     # Get security groups
36     print("\nSecurity Groups:")
37     for sg in instance['SecurityGroups']:
38         print(f" {sg[ 'GroupId']} ({sg[ 'GroupName']} )")
39
40     # Get block device mappings
41     print("\nBlock Device Mappings:")

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
```

Below the code, the terminal output is shown:

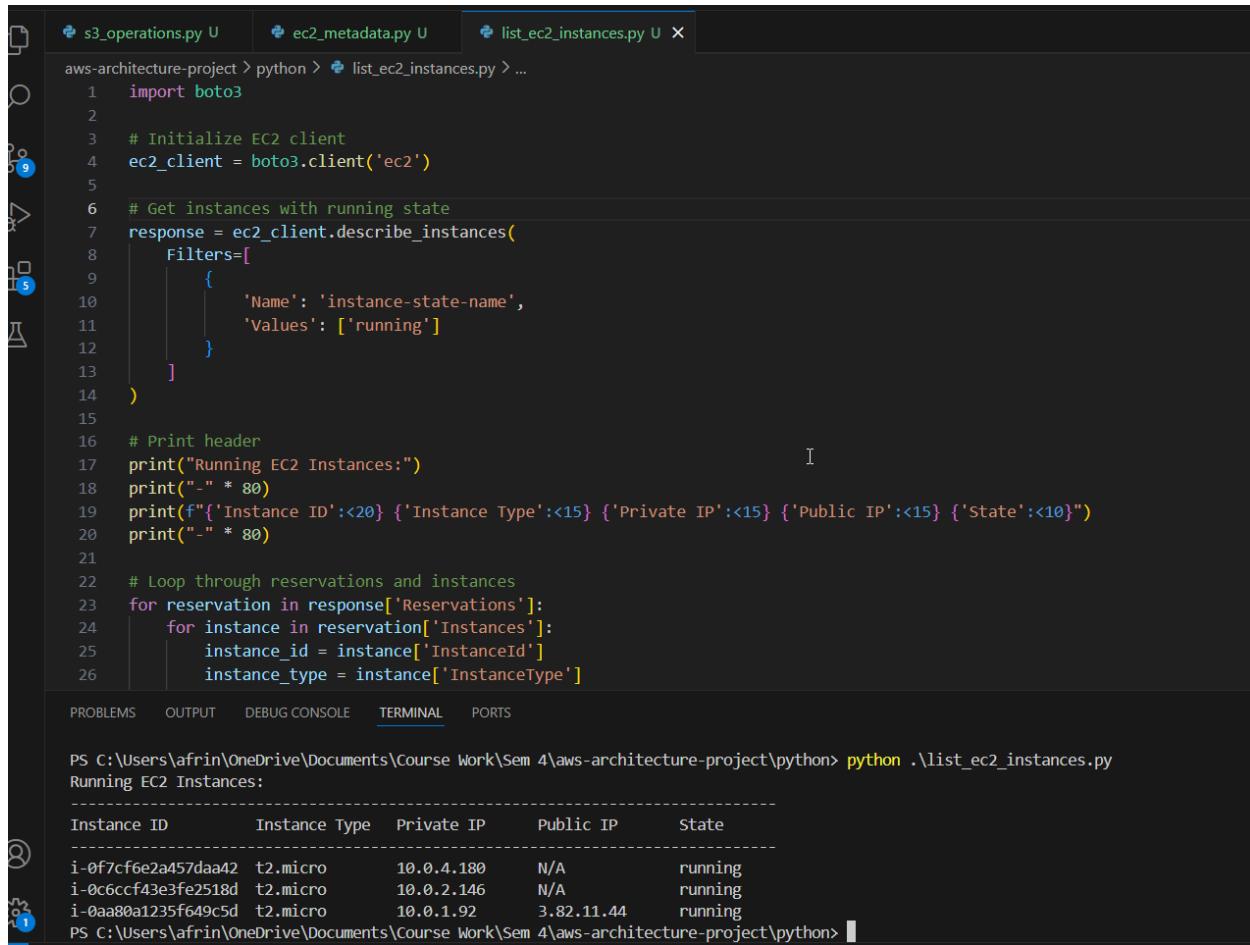
```
PS C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4\aws-architecture-project\python> python .\ec2_metadata.py
Retrieving metadata for instance: i-0c6ccf43e3fe2518d

EC2 Instance Metadata:
-----
Instance ID: i-0c6ccf43e3fe2518d
Instance Type: t2.micro
Availability Zone: us-east-1a
VPC ID: vpc-0235f5ea6483aa4a8
Subnet ID: subnet-0ee15382af24b3bcb
State: running
Private IP: 10.0.2.146

Security Groups:
    sg-00595c492b15a45bb (web-sg)

Block Device Mappings:
    /dev/xvda: vol-0e29b6fd1f5952553
-----
```

List running EC2 instances



The screenshot shows a VS Code interface with three tabs open: `s3_operations.py`, `ec2_metadata.py`, and `list_ec2_instances.py`. The `list_ec2_instances.py` tab is active, displaying the following Python code:

```
aws-architecture-project > python > list_ec2_instances.py > ...
1 import boto3
2
3 # Initialize EC2 client
4 ec2_client = boto3.client('ec2')
5
6 # Get instances with running state
7 response = ec2_client.describe_instances(
8     Filters=[
9         {
10            'Name': 'instance-state-name',
11            'Values': ['running']
12        }
13    ]
14 )
15
16 # Print header
17 print("Running EC2 Instances:")
18 print("-" * 80)
19 print(f"{'Instance ID':<20} {'Instance Type':<15} {'Private IP':<15} {'Public IP':<15} {'State':<10}")
20 print("-" * 80)
21
22 # Loop through reservations and instances
23 for reservation in response['Reservations']:
24     for instance in reservation['Instances']:
25         instance_id = instance['InstanceId']
26         instance_type = instance['InstanceType']
```

Below the code editor, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected, showing the command `PS C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4\aws-architecture-project\python> python .\list_ec2_instances.py` and the output "Running EC2 Instances:" followed by a table of instance details.

Instance ID	Instance Type	Private IP	Public IP	State
i-0f7cf6e2a457daa42	t2.micro	10.0.4.180	N/A	running
i-0c6ccf43e3fe2518d	t2.micro	10.0.2.146	N/A	running
i-0aa80a1235f649c5d	t2.micro	10.0.1.92	3.82.11.44	running

Invoke the AWS Lambda function manually

```

aws-architecture-project > python > invoke_lambda.py > ...
1  # invoke_lambda.py - Invokes a Lambda function with a test event
2  import boto3
3  import json
4
5  # Initialize Lambda client
6  lambda_client = boto3.client('lambda')
7
8  # Define the Lambda function name
9  function_name = 'S3-upload-logger'
10
11 # Create a test event payload
12 test_event = {
13     "Records": [
14         {
15             "S3": {
16                 "bucket": {
17                     "name": "project-bucket-730335336250"
18                 },
19                 "object": {
20                     "key": "test-boto3-invoke.txt"
21                 }
22             }
23         }
24     ]
25 }
26
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```

PS C:\Users\afrin\OneDrive\Documents\Course Work\Sem 4\aws-architecture-project\python> **python .\invoke_lambda.py**
Invoking Lambda function: S3-upload-logger
Status Code: 200
Response Payload: {"statusCode": 200, "body": "\"New file uploaded: test-boto3-invoke.txt in bucket: project-bucket-730335336250\""}
Lambda invocation completed

Verify in the cloud watch

The screenshot shows the AWS CloudWatch Log Groups interface. The left sidebar navigation includes 'CloudWatch' (selected), 'Dashboards', 'Logs' (selected), 'Metrics', 'X-Ray traces', 'Events', 'Application Signals', 'Network Monitoring', and 'Insights'. Under 'Logs', it shows 'Log groups' and 'Log streams'. The main content area displays a table of log events for the log group '/aws/lambda/S3-upload-logger'. The table has columns for 'Timestamp' and 'Message'. The messages show the Lambda function starting, receiving a request to invoke, and then reporting back that a new file was uploaded to the specified S3 bucket.

Timestamp	Message
2025-05-15T02:05:33.094Z	START RequestId: 81020c3e-cf99-4a23-887c-3ef23e397ea Version: \$LATEST
2025-05-15T02:05:33.093Z	[INFO] 2025-05-15T02:05:33.093Z 81020c3e-cf99-4a23-887c-3ef23e397ea New file uploaded: test-boto3-invoke.txt in bucket: project-bucket-730335336250
2025-05-15T02:05:33.094Z	END RequestId: 81020c3e-cf99-4a23-887c-3ef23e397ea
2025-05-15T02:05:33.096Z	REPORT RequestId: 81020c3e-cf99-4a23-887c-3ef23e397ea Duration: 2.35 ms Billed Duration: 63 ms Max Memory Used: 128 MB Init Duration: 325.44 ms

Challenges faced and solutions implemented

1. Multi-AZ Requirements for AWS Services

Challenge: Both the Application Load Balancer and RDS required resources in multiple Availability Zones, which wasn't initially configured in our Terraform setup.

Solution: Modified our Terraform configuration to create subnets in two Availability Zones (us-east-1a and us-east-1b). This provided the necessary redundancy for both the ALB and RDS, enabling high availability while highlighting the importance of multi-AZ architectures in production environments.

2. Integration Between Terraform and CloudFormation

Challenge: Coordinating resources between Terraform (networking) and CloudFormation required careful integration.

Solution: Used Terraform outputs to capture resource IDs and manually transferred these to the CloudFormation template. While not fully automated, this approach allowed us to leverage the strengths of both IaC tools while maintaining a clear separation of concerns.

Future Improvements for Better Scalability and Automation

1. CI/CD Pipeline

- Implement GitHub Actions workflows for automated testing and deployment
- Use AWS CodePipeline to automate application deployments

Bonus Challenge

Create a rest API

Create REST API

API details

- New API Create a new REST API.
- Clone existing API Create a copy of an API in this AWS account.
- Import API Import an API from an OpenAPI definition.
- Example API Learn about API Gateway with an example API.

API name
lambda-s3-api

Description - optional
API Gateway to invoke Lambda for S3 logging

API endpoint type
Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.

IP address type [Info](#)
Select the type of IP addresses that can invoke the default endpoint for your API.

- IPv4 Supports only edge-optimized and Regional API endpoint types.
- Dualstack Supports all API endpoint types.

[Cancel](#) [Create API](#)

Create resource

Resource details

[Proxy resource info](#)
Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example (proxy+).

Resource path / **Resource name** invoke

[CORS \[Cross Origin Resource Sharing\] info](#)
Create an OPTIONS method that allows all origins, all methods, and several common headers.

[Cancel](#) [Create resource](#)

Create Method

The screenshot shows the 'Create method' page in the AWS API Gateway console. A green success message at the top says 'Successfully created resource "/invoke"'. The 'Method details' section shows 'Method type' as 'POST'. The 'Integration type' section has 'Lambda function' selected, with a note: 'Integrate your API with a Lambda function.' Below it, 'AWS service' and 'VPC link' options are shown. The 'Lambda proxy integration' section is collapsed. The 'Lambda function' section shows an ARN: 'us-east-1:awslambda:us-east-1:73035336250:functions:s3-upload-logger'. The 'Grant API Gateway permission to invoke your Lambda function' section is expanded, showing a note: 'When you save your changes, API Gateway updates your Lambda function's resource-based policy to allow this API to invoke it.' The 'Integration timeout' section shows a value of '29000'. At the bottom right are 'Save' and 'Cancel' buttons.

Update the mapping template

The screenshot shows the 'Edit integration request' page in the AWS API Gateway console. It displays a 'Mapping templates' section with a 'Content type' dropdown set to 'application/json'. Below it is a 'Generate template' button and a 'Remove' button. The 'Template body' section contains a YML template:

```
1 * []
2 *   "Records": [
3 *     {
4 *       "s3": {
5 *         "object": {
6 *           "name": "project-bucket-73035336250"
7 *         }
8 *       }
9 *     }
10 *   ]
11 * }
```

Below the template body, there is a note: 'Use YAML templates to create your mapping template. Learn more' with a link icon. At the bottom are 'Add mapping template', 'Cancel', and 'Save' buttons.

Set up integration response

Header mappings [Info](#)
No headers for this status code

Mapping templates

Content type
application/json

Generate template [Remove](#)

Template body

```
1 {{# $ }}  
2   "filename": "test-file-from-api.txt"  
3 {{/ $ }}
```

Use VTL templates to create your mapping template. [Learn more](#)

Add mapping template

Cancel Save

Deploy API

Successfully created deployment for lambda-s3-api. This deployment is active for prod.

Notifications 0 1 2 3 0 0 0

Stage actions Create stage

prod

Stage details [Info](#)

Stage name: prod

Cache cluster [Info](#) Inactive

Default method-level caching [Info](#) Inactive

Invoke URL <https://uv7rmugw.execute-api.us-east-1.amazonaws.com/prod>

Active deployment o1hu6 on May 15, 2025, 16:56 (UTC-04:00)

Logs and tracing [Info](#)

CloudWatch logs [Info](#) Inactive

X-Ray tracing [Info](#) Inactive

Custom access logging [Info](#) Inactive

Stage variables Deployment history Documentation history Canary Tags