

Developer Operations – Assignment 2

CORE ARCHITECTURE & AWS AUTOSCALING

BSc's in (Computer Forensics & Security)

Course SE602

A PROJECT REPORT BY

& Support Of

Jabez Dickson

Exam No. 20102440

Jimmy McGibney

Lecturer



Ollscoil
Teicneolaíochta
an Oirdheiscirt

South East
Technological
University

Table of Contents

Introduction	1
Objective of Assignment	1.1
Overview of Architecture	1.2
 Architecture Design	 2
AWS Services Used	2.1
 Implementation Steps	 3
VPC & Subnet Setup	3.1
Master Instance Creation	3.2
AMI Creation	3.3
Load Balancer & Auto Scaling	3.4
Additional Features (SNS)	3.5
CloudWatch & Scaling Policies	3.6
Traffic Testing	3.7

1. Introduction

Report Briefing:

Report By: Jabez Jacob Dickson

Reviewed By: South East Technological University (SETU)

The objective of this assignment is to demonstrate the deployment and automated management of a highly available, load-balanced, auto-scaling web application in an AWS environment.

This project leverages core AWS services, including EC2, VPC, Auto Scaling, Application Load Balancer, and CloudWatch, to create a robust and scalable infrastructure.

The implementation incorporates features such as dynamic scaling policies using CloudWatch alarms, traffic management with an Application Load Balancer, and proactive monitoring through SNS notifications. This ensures high availability, fault tolerance, and performance optimization for the deployed application.

The following report outlines the architecture design, key implementation steps, additional functionality, and testing results, providing a comprehensive walkthrough of the deployment.

1.1 Objective of Assignment

The primary objective of this assignment is to design, implement, and demonstrate the deployment of a load-balanced auto-scaling web application using AWS services. This involves creating a scalable and highly available architecture that leverages AWS infrastructure to manage dynamic workloads efficiently.

Key Configuration Notes

1. **Deployment:** Hosting a web application on EC2 instances with appropriate configurations and dependencies.
2. **Scalability:** Implementing Auto Scaling Groups to dynamically adjust the number of instances based on application demand.
3. **High Availability:** Ensuring continuous availability through a load-balanced architecture that distributes traffic across multiple availability zones.
4. **Monitoring:** Configuring CloudWatch metrics and alarms for real-time monitoring and automated scaling policies.
5. **Security:** Designing secure networking using a Virtual Private Cloud (VPC), security groups, and IAM roles to ensure controlled access to resources.
6. **Additional Functionality:** Enhancing the solution with extra features such as SNS notifications for monitoring or backend service integrations.

1.2 Overview of Architecture

The architecture implemented for this project is designed to deliver a scalable, highly available, and secure web application using AWS services. The core components include:



Virtual Private Cloud (VPC):

- A custom VPC provides network isolation for resources.
- Public subnets are used for load balancer and application servers, while private subnets are reserved for backend services.

Application Load Balancer (ALB):

- Distributes incoming traffic evenly across multiple EC2 instances.
- Ensures high availability by routing traffic to instances in different availability zones.

Auto Scaling Group (ASG):

- Automatically adjusts the number of EC2 instances based on demand.
- Ensures cost-efficiency by scaling down during low traffic and scaling up during peak times.

EC2 Instances:

- Serve as application servers, hosting the React-based web application.
- Configured using a custom Amazon Machine Image (AMI) created from a master instance.

CloudWatch Monitoring:

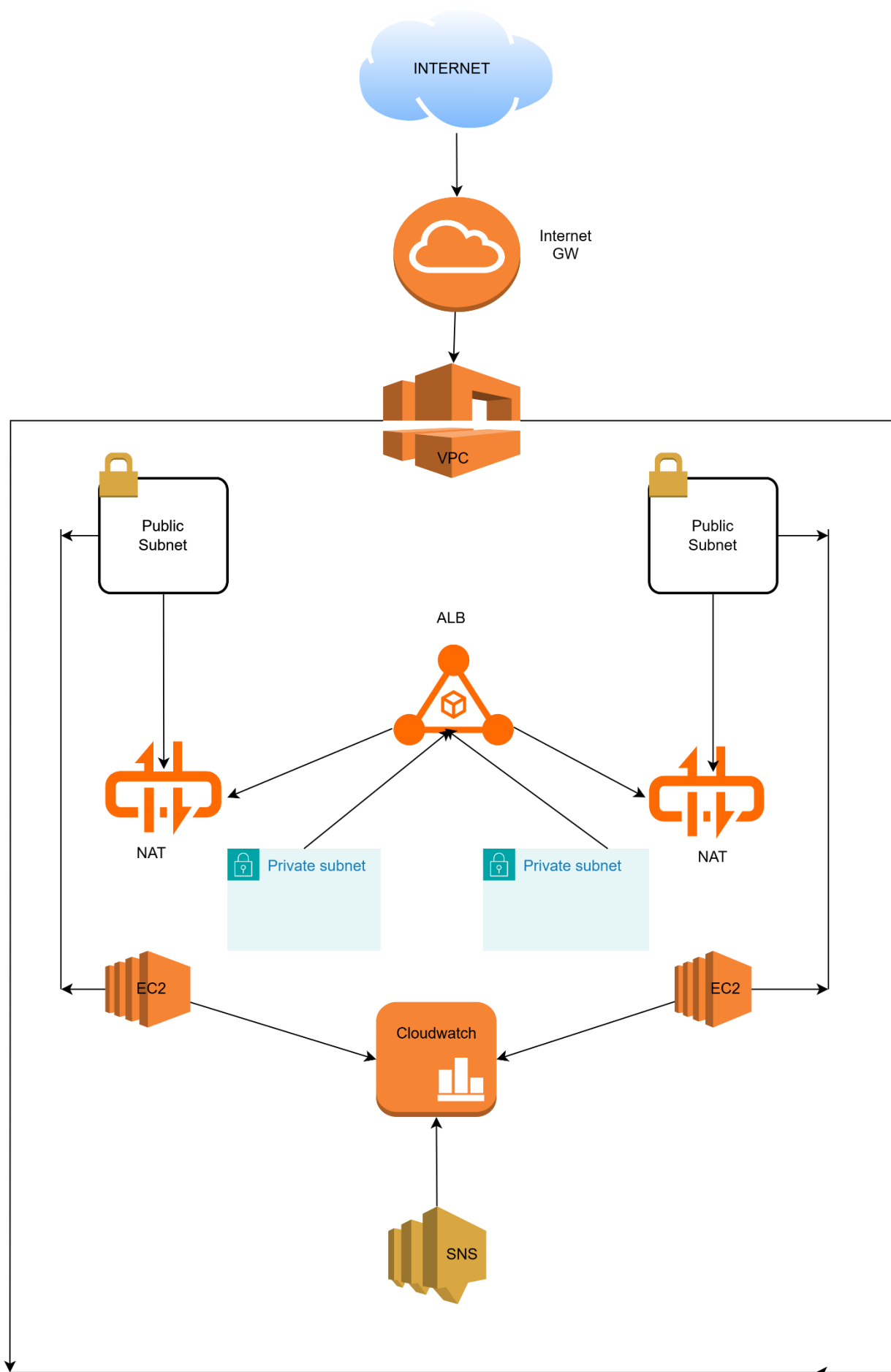
- Tracks key performance metrics such as CPU utilization.
- Enables dynamic scaling policies and integrates with SNS for real-time alerts.

Security Groups and IAM Roles:

- Security Groups restrict inbound and outbound traffic to only necessary ports (e.g., HTTP, HTTPS).
- IAM Roles provide secure access for EC2 instances to interact with AWS service

Monitoring Configuration File: (Referenced Using ChatGPT)

```
{
  "metrics": {
    "append_dimensions": {
      "InstanceId": "${aws:InstanceId}"
    },
    "metrics_collected": {
      "cpu": {
        "measurement": [
          "cpu_usage_idle",
          "cpu_usage_iowait"
        ],
        "metrics_collection_interval": 60
      },
      "disk": {
        "measurement": [
          "used_percent"
        ],
        "metrics_collection_interval": 60
      },
      "mem": {
        "measurement": [
          "mem_used_percent"
        ],
        "metrics_collection_interval": 60
      }
    }
  }
}
```



2. Architecture Design

2.1 Services Used

React:

Used to build the frontend of the web application.
Deployed on EC2 instances to serve the client-side interface.



Firebase:

Provides backend services such as authentication, database, or cloud functions.
Integrated into the React app for seamless data handling and service connectivity.



Application Load Balancer (ALB):

Distributes incoming HTTP(S) traffic across EC2 instances.
Includes health checks to ensure only healthy instances receive traffic.



Amazon CloudWatch:

Monitors system metrics (e.g., CPU utilization).
Triggers alarms for scaling and sends notifications via SNS.



Amazon SNS (Simple Notification Service):

Sends email or SMS alerts when CloudWatch alarms are triggered.



Amazon Elastic Block Store (EBS):

Provides storage volumes attached to EC2 instances.



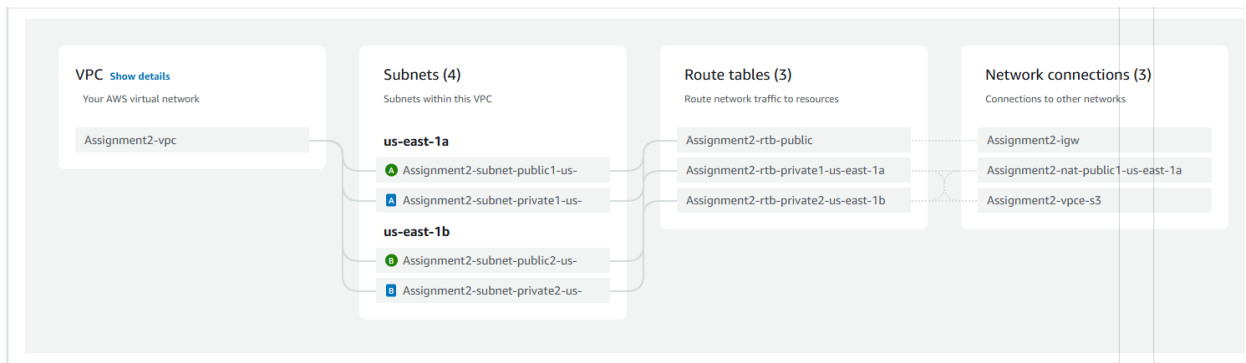
Security Groups:

Controls inbound and outbound traffic for EC2 instances and the ALB.

3. Implementation Steps

3.1 VPC & Subnet Setup

The Virtual Private Cloud (VPC) was created to provide a secure and isolated network for web applications. The VPC setup ensures controlled communication between the application components and enables high availability and scalability. Here's how the VPC was configured:



Create VPC: vpc-0c8ff0b27885b47a5

Enable DNS hostnames

Enable DNS resolution

Verifying VPC creation: vpc-0c8ff0b27885b47a5

Create S3 endpoint: vpce-061339f9746354f60

Create subnet: subnet-0c750caa42a98bf5a

Create subnet: subnet-06dc2a857e0947d36

Create subnet: subnet-00faad0205f48873d

Create subnet: subnet-05f3a7338ea8d534f

Create internet gateway: igw-0b5ba090c6902ac8e

Attach internet gateway to the VPC

Create route table: rtb-0d50468293ed90c03

Create route

Associate route table

Associate route table

Allocate elastic IP: eipalloc-0f31b4de6638fd383

Create NAT gateway: nat-0ee2f64e32bc721a7

Wait for NAT Gateways to activate

Create route table: rtb-0cd3b45756e68462a

Create route

Associate route table

Create route table: rtb-0ccb6ce649a5e6e9e

Create route

Associate route table

Verifying route table creation

Associate S3 endpoint with private subnet route tables: vpce-061339f9746354f60

Subnet Configuration

Number of Subnets: Four subnets were created to ensure high availability:

2 Public Subnets:

- One in us-east-1a (Availability Zone A).
- One in us-east-1b (Availability Zone B).
- Used for hosting the Application Load Balancer.

2 Private Subnets:

- One in us-east-1a.
- One in us-east-1b.
- Reserved for backend services or additional instances.

Security Groups:

Inbound Traffic: Open to

- HTTP (port 80)
- HTTPS (port 445)
- SSH (port 22)
- Localhost (port 3000)

Outbound Traffic: Open to all

Public Table

rtb-0565ca6178b49352f / Week7VPC-rtb-public

Details

Routes

Subnet associations

Edge associations

Route propagation

Tags

Routes (2)

Both Edit routes

Filter routes

Destination	Target	Status	Propagated
0.0.0.0/0	igw-0ca6133872d2d4bb5	Active	No
10.0.0.0/16	local	Active	No

Private Table

rtb-0fc74be034896bf3e / Week7VPC-rtb-private1-us-east-1a

Details

Routes

Subnet associations

Edge associations

Route propagation

Tags

Routes (3)

Both Edit routes

Filter routes

Destination	Target	Status	Propagated
pl-63a5400a	vpce-0d72dc596e1443547	Active	No
0.0.0.0/0	nat-0a6ee7e01ba888cb5	Active	No
10.0.0.0/16	local	Active	No

3.2 Creation of Master Instance

Creating A Security Group

The security groups were configured within the custom VPC (Assignment2-VPC) to ensure secure and controlled access to the deployed resources.

A Web Server Security Group was created to allow inbound HTTP (port 80) traffic for public access to the application and SSH (port 22) for administrative access, restricted to specific IP addresses for security.

Additionally, a Load Balancer Security Group was configured to allow inbound HTTP (port 80) and HTTPS (port 443) traffic for the Application Load Balancer (ALB), ensuring secure routing of incoming requests.

Outbound traffic was left open by default to allow resources to communicate with external services as needed. These security groups ensure that all instances and components within the VPC are protected while maintaining functionality for web application deployment.

Basic details

Security group name [Info](#)

Name cannot be edited after creation.

Description [Info](#)

VPC [Info](#)

Inbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
Custom TCP	TCP	3000	An... 0.0.0.0/0	<input type="text"/> Delete
HTTP	TCP	80	An... 0.0.0.0/0	<input type="text"/> Delete
HTTPS	TCP	443	An... 0.0.0.0/0	<input type="text"/> Delete
SSH	TCP	22	An... 0.0.0.0/0	<input type="text"/> Delete

Outbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Destination Info	Description - optional Info
All traffic	All	All	An... 0.0.0.0/0	<input type="text"/> Delete

Storage Solution

The t2.micro instance type was selected for this project after encountering issues with the **t2.nano**, which frequently ran out of resources during application setup and dependency downloads.

The t2.micro offers improved performance with **1 vCPU and 1GB of RAM**, providing sufficient resources to handle the installation of Node.js, React dependencies, and Firebase while ensuring stable operation of the web application.

Its balance between cost and capability makes it an ideal choice for small-scale web application hosting while still supporting future scaling through Auto Scaling Groups.

The **VPC (Assignment2-VPC)** was created to provide an isolated network environment for the project and was linked with the security groups to control traffic flow securely.

The VPC includes both public and private subnets distributed across multiple availability zones, ensuring high availability and fault tolerance.

These security groups were designed to ensure secure access to resources while allowing seamless communication between the ALB, EC2 instances, and other AWS services within the VPC.

By linking the security groups and VPC, the architecture maintains strict access control while supporting scalable and reliable web application deployment.

▼ Network settings Info

VPC - required Info

vpc-0c8ff0b27885b47a5 (Assignment2-vpc)
10.0.0.0/16

Subnet Info

subnet-05f3a7338ea8d534f Assignment2-subnet-private2-us-east-1b
VPC: vpc-0c8ff0b27885b47a5 Owner: 098122911180 Availability Zone: us-east-1b
Zone type: Availability Zone IP addresses available: 4091 CIDR: 10.0.144.0/20

Create new subnet

Auto-assign public IP Info

Disable

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☐ Create security group ☒ Select existing security group

Common security groups Info

Select security groups

MasterAssignmentSG sg-035081fba869bc00e X
VPC: vpc-0c8ff0b27885b47a5

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

► Advanced network configuration

3.3 AMI Creation Instance

The t2.micro instance type was selected for this project after encountering issues with the **t2.nano**, which frequently ran out of resources during application setup and dependency downloads.

When you create an AMI with the web app and all dependencies installed, new instances launched from the AMI in your Auto Scaling Group (ASG) inherit this setup.

This eliminates the need to repeat the setup process for every instance, significantly reducing deployment time during scaling. Use the script below to install firebase and other components before the AMI Image Capture. (Also can be pasted into user data script)

```
#!/bin/bash
# Run as root
sudo su

# Update the system
yum update -y

# Install Node.js and Git
curl -sL https://rpm.nodesource.com/setup_18.x | bash -
yum install -y nodejs git

# Clone the React TMDB Assignment repository
git clone https://github.com/JacobDicksonOfficial/react-tmdb-assignment.git
/var/www/tmdb-app

# Navigate to the 'movies' directory
cd /var/www/tmdb-app/movies

# Install dependencies
npm install

# Install Firebase (if required separately)
npm install firebase

# Set HOST and PORT for public access
export HOST=0.0.0.0
export PORT=3000

# Start the React development server
npm start &
```

Instance ID


 i-05dadadd1bcf71e9c (Master Server (Assignment 2))

Image name

Assignment2-Master-WebApp-AMI

Maximum 127 characters. Can't be modified after creation.

Image description - optional

This AMI contains the pre-configured React web application with all dependencies installed, based on Amazon Linux 2023.

Maximum 255 characters

3.4 Load Balancer & Auto Scaling

Using the security group policy setup, a separate security group was created for the load balancer to allow only HTTP (port 80), HTTPS (port 443), HTTP (port 3000) inbound traffic from 0.0.0.0/0. Outbound traffic is open to all.

This approach follows AWS best practices and enhances your setup's security for the assignment. Let me know if you'd like to proceed with this adjustment!

Create A Target Group

A Target Group in AWS is a key component of the Load Balancer setup. It defines the group of resources (such as EC2 instances, containers, or IP addresses) that the Load Balancer forwards traffic to. It acts as the bridge between the Load Balancer and your back-end infrastructure.

- **Routing traffic to the right back-end resources.**
- **Monitoring health and ensuring only healthy resources handle requests.**
- **Supporting dynamic scaling and flexible application architectures.**

Protocol: HTTP

Port: 3000

VPC: Assignment2-VPC

Health Check Settings:

Protocol: HTTP

Path: /

Port: Use Traffic port or Override to 3000

Success Codes: 200

Details

arn:aws:elasticloadbalancing:us-east-1:098122911180:targetgroup/Assignment2-TG/0d644020851ab9ef

Target type

Instance

Protocol : Port

HTTP: 3000

Protocol version

HTTP1

VPC

vpc-0c8ff0b27885b47a5

IP address type

IPv4

Load balancer

Assignment2-ALB

1

0

0

0

0

Total targets

Healthy

Unhealthy

Unused

Initial

Draining

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

Monitoring

Health checks

Attributes

Tags

Registered targets (1)

Anomaly mitigation: Not applicable

Deregister

Register targets

Target groups route requests 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.

Filter targets

Instance ID

Name

Port

Zone

Health status

Health status details

Administrative o...

Override details

Launch...

Anomaly c

I-05dadedd1bcf71e9c

Master Server (...)

3000

us-east-1a (us...

Healthy

-

No override

No override is curren...

November...

Normal

Create Application Load Balancer

This highlights the risks associated with inadequate data protection and the ease with which attackers can access, move, and misuse sensitive information once they gain a foothold in the network which is why load balancing in a real-life scenario is important for purposes like DDoS.

VPC (Assignment2-VPC).

Chosen two public subnets in different availability zones:

Assignment2-subnet-public1-us-east-1a.

Assignment2-subnet-public2-us-east-1b.

1. Attach the created ALB **Security Group (Assignment2-ALB-SG)**.
2. Ensure this security group allows:
 - **HTTP** (port 80): 0.0.0.0/0.
 - **HTTPS** (port 443): 0.0.0.0/0
 - **HTTP** (port 3000): 0.0.0.0/0

Forward to **Assignment2-TG** (the Target Group created earlier).

▼ Details

Load balancer type
Application

Scheme
Internet-facing

Load balancer ARN
[arn:aws:elasticloadbalancing:us-east-1:098122911180:loadbalancer/app/Assignment2-ALB/dd4d2faa7ab27c91](#)

Status
Active

Hosted zone
Z35XDOTRQ7X7K

VPC
[vpc-0c8ff0b27885b47a5](#)

Availability Zones
[subnet-06dc2a857e0947d36](#) us-east-1b (use1-az6)
[subnet-0c750caa42a98bf5a](#) us-east-1a (use1-az4)

DNS name info
[Assignment2-ALB-72152443.us-east-1.elb.amazonaws.com](#) (A Record)

Load balancer IP address type
IPv4

Date created
November 30, 2024, 22:41 (UTC+00:00)

Listeners and rules (1) info

Manage rules Manage listener Add listener

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store
<input type="checkbox"/> HTTP:80	<div>Forward to target group<ul style="list-style-type: none">Assignment2-TG: 1 (100%)Target group stickiness: Off</div>	1 rule	ARN	Not applicable	Not applicable	Not applicable	Not applici

DNS Tested: <http://assignment2-alb-72152443.us-east-1.elb.amazonaws.com/login>

Instance IP: <http://54.92.202.204:3000/>

Create a Launch Template

The Launch Template is a critical component in the Auto Scaling setup, providing a reusable blueprint for creating new EC2 instances in the Auto Scaling Group.

A launch template was created and configured with the necessary parameters to ensure consistency in the deployment of additional instances.

This template specifies the AMI ID of the custom pre-configured image, ensuring that each instance launches with the React web application and its dependencies pre-installed. The template defines the instance type (t2.micro), which balances cost and performance for this workload.

It includes the appropriate security group to allow traffic from the ALB and ensures that new instances can communicate seamlessly.

Optionally, a User Data script was added to automate the process of starting the web application upon instance creation, binding it to the correct IP and port (0.0.0.0 and 3000).

```
#!/bin/bash
cd /var/www/tmdb-app/movies
export HOST=0.0.0.0
export PORT=3000
npm start &
```

By leveraging this launch template, the Auto Scaling Group can efficiently and automatically create new instances when scaling is required, ensuring uniformity and reducing manual configuration efforts.

Assignment2-LaunchTemplate (lt-0bbd5547ca30c2b98)

Launch template details

Launch template ID

lt-0bbd5547ca30c2b98

Launch template name

Assignment2-LaunchTemplate

Default version

1

Details

Versions

Template tags

Launch template version details

Version

1 (Default)

Description

The Launch Template defines how new EC2 instances are created when the ASG scales.

Date created

2024-11-30T23:43:37.000Z

Instance details

Storage

Resource tags

Network interfaces

Advanced details

AMI ID

ami-0c560d3cc98272722

Instance type

t2.micro

Availability Zone

-

Security groups

-

Security group IDs

sg-035081fba869bc00e, sg-0a1800742afc70ded

Create an Auto Scaling Group (ASG)

The Launch Template is a critical component in the Auto Scaling setup, providing a reusable blueprint for creating new EC2 instances in the Auto Scaling Group.

An Auto Scaling Group (ASG) in AWS is a critical service designed to manage the scalability and availability of EC2 instances within a defined group.

The ASG ensures that your application can dynamically adjust its capacity to match the current workload, optimizing both performance and cost.

It uses a Launch Template to define the configuration of instances, such as the AMI, instance type, and security groups, ensuring consistency in deployments.

The ASG is associated with a specified VPC and subnets, enabling instances to be distributed across multiple Availability Zones for fault tolerance.

Assignment2-ASG Capacity overview

arn:aws:autoscaling:us-east-1:098122911180:autoScalingGroup:931bbadd-7f59-4363-ab21-1cfeab94c51c:autoScalingGroupName/Assignment2-ASG

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

Date created
Sun Dec 01 2024 00:07:16 GMT+0000 (Greenwich Mean Time)

Launch template

<p>Launch template</p> <p>lt-0bbd5547ca30c2b98</p> <p>Assignment2-LaunchTemplate</p>	<p>AMI ID</p> <p>ami-0c560d3cc98272722</p>	<p>Instance type</p> <p>t2.micro</p>
<p>Version</p> <p>Default</p>	<p>Security groups</p> <p>-</p>	<p>Security group IDs</p> <p>sg-035081fba869bc00e</p> <p>sg-0a1800742afc70ded</p>
<p>Description</p> <p>The Launch Template defines how new EC2 instances are created when the ASG scales.</p> <p>View details in the launch template console</p>	<p>Storage (volumes)</p> <p>-</p>	<p>Key pair name</p> <p>week7</p>

3.5 Additional Feature An SNS topic was created within the ASG setup to automate email update for additional functionalities when an instance is created, deleted, failed to launch or failed to terminate as a bonus.

Notification 1

SNS Topic

Choose an SNS topic to use to send notifications

Default_CloudWatch_Alarms_Topic (20102440@mail.wit.ie)

Create a topic

Event types

Notify subscribers whenever instances

Launch

Terminate

Fail to launch

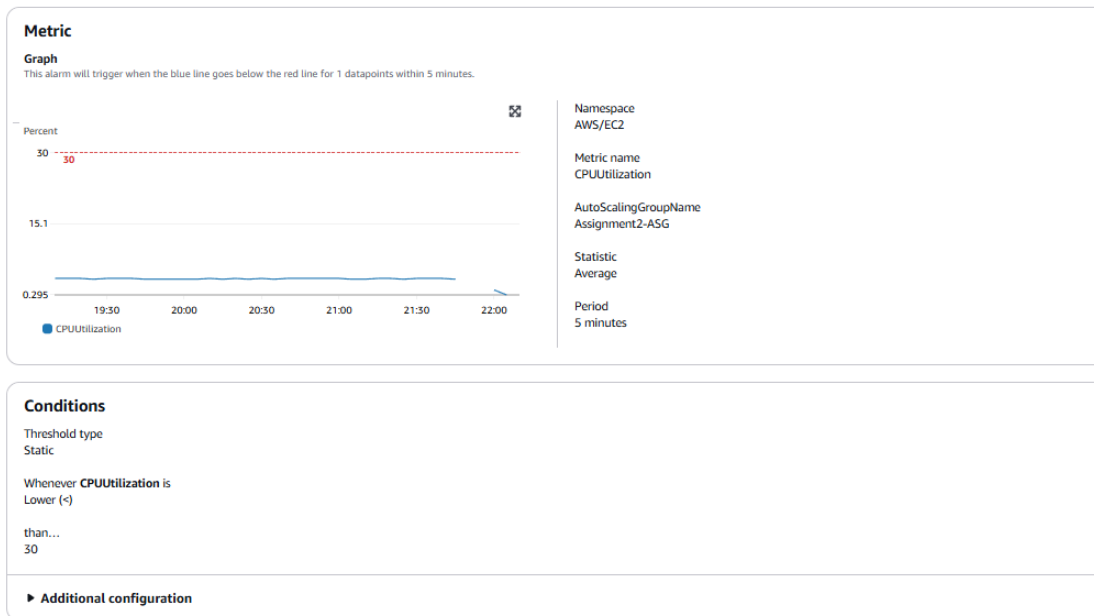
Fail to terminate

Remove

3.6 Cloudwatch & Scaling Policies

Cloudwatch & Scaling Policies (Part 2)

Scale-In Alarm (Low CPU Usage)



Threshold Type: Static.

Condition: Less than 30% (CPU utilization falls below 30%).

Period: 5 minutes (continuous threshold breach for 5 minutes).

Reason: Scaling in during low CPU utilization optimizes cost by removing unused instances while ensuring workload requirements are still met.

Configured an SNS topic (Assignment2-ScalingNotifications) for alerts.

Reason: Notifications help monitor scaling activities in real time, aiding in debugging and ensuring awareness of resource usage.

Create an Auto Scaling Group (ASG) Dynamic Policy (Part 1)

The Launch Template is a critical component in the Auto Scaling setup, providing a reusable blueprint for creating new EC2 instances in the Auto Scaling Group.

Scale-Out Policy

Policy type

Simple scaling ▼

Scaling policy name

ScaleOutPolicy

CloudWatch alarm

Choose an alarm that can scale capacity whenever:

ScaleOut-HighCPU ▼



[Create a CloudWatch alarm](#)

breaches the alarm threshold: CPUUtilization > 70 for 1 consecutive periods of 300 seconds for the metric dimensions:

AutoScalingGroupName = Assignment2-ASG

Take the action

Add ▼

1

capacity units ▼

And then wait

300

seconds before allowing another scaling activity

When: $\geq 70\%$ (already tied to the alarm).

Add Capacity: +1 instance.

Reason: Add one instance when CPU utilization exceeds 70%.

Set the cooldown period to 300 seconds (5 minutes).

Reason: Prevent excessive scaling by giving the system time to stabilize after a scale-out event.



Create an Auto Scaling Group (ASG) Dynamic Policy (Part 1)

The Launch Template is a critical component in the Auto Scaling setup, providing a reusable blueprint for creating new EC2 instances in the Auto Scaling Group.

Scale-In Policy

Policy type
Simple scaling ▼

Scaling policy name
ScaleOutPolicy

CloudWatch alarm
Choose an alarm that can scale capacity whenever:
ScaleOut-HighCPU ▼ 
[Create a CloudWatch alarm](#) 
breaches the alarm threshold: CPUUtilization > 70 for 1 consecutive periods of 300 seconds for the metric dimensions:
AutoScalingGroupName = Assignment2-ASG

Take the action
Add ▼ 1 capacity units ▼

And then wait
300 seconds before allowing another scaling activity

When: $\leq 30\%$ (already tied to the alarm).

Add Capacity: -1 instance.

Reason: Remove one instance when CPU utilization drops below 30

Set the cooldown period to 300 seconds (5 minutes).

Reason: Prevent excessive scaling by avoiding immediate scale-in after a scale-out event

3.7 Traffic testing

One of the best ways to initiate traffic testing is to do a CPU intensive task inside the instance that was created to set off the CloudWatch alarm which will cause the alarm to go off when the CPU bottlenecks at 70%.

The following tool was used in the instance to initiate this task:

```
sudo yum install epel-release -y
```

```
sudo yum install stress -y
```

Run Stress Test: Simulate 100% CPU usage on 2 cores for 10 minutes:

```
stress --cpu 2 --timeout 600
```

As Expected, AWS SNS feature Alerts my Email about the CPU Task



AWS Notifications <no-reply@sns.amazonaws.com>
to me ▾

Service: AWS Auto Scaling
Time: 2024-12-01T23:01:24.930Z
RequestId: aad64e11-d25d-7194-dc6a-fd41be6335d7
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountId: 098122911180
AutoScalingGroupName: Assignment2-ASG
AutoScalingGroupARN: arn:aws:autoscaling:us-east-1:098122911180:autoScalingGroup:08c4cea9-3417-4d78-98b8-89cf0d22511d:autoScalingGroupName/Assignment2-ASG
ActivityId: aad64e11-d25d-7194-dc6a-fd41be6335d7
Description: Launching a new EC2 instance: i-085c879d87506d707
Cause: At 2024-12-01T23:00:51Z an instance was launched in response to an unhealthy instance needing to be replaced.
StartTime: 2024-12-01T23:00:53.724Z
EndTime: 2024-12-01T23:01:24.930Z
StatusCode: InProgress
StatusMessage:
Progress: 50
EC2InstanceId: i-085c879d87506d707

...

This launches a new instance in the Instances tab s part of the ASG

It inherits the correct configurations (tags, security groups, etc.) from the Launch Template.

References

- Amazon Web Services. (n.d.). *Amazon VPC Documentation*. Retrieved from <https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html>
- Amazon Web Services. (n.d.). *Subnets in Amazon VPC*. Retrieved from https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Subnets.html
- Amazon Web Services. (n.d.). *Internet Gateway Overview*. Retrieved from https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html
- Amazon Web Services. (n.d.). *NAT Gateway Documentation*. Retrieved from <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html>
- Amazon Web Services. (n.d.). *Amazon EC2 Documentation*. Retrieved from <https://docs.aws.amazon.com/ec2/index.html>
- Amazon Web Services. (n.d.). *Auto Scaling Groups Documentation*. Retrieved from <https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroup.html>
- Amazon Web Services. (n.d.). *Launch Templates Documentation*. Retrieved from <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-templates.html>
- Amazon Web Services. (n.d.). *Application Load Balancer Documentation*. Retrieved from <https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html>
- Amazon Web Services. (n.d.). *Security Groups Documentation*. Retrieved from https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html
- Amazon Web Services. (n.d.). *Amazon Elastic Block Store (EBS) Documentation*. Retrieved from <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html>
- Amazon Web Services. (n.d.). *Amazon CloudWatch Documentation*. Retrieved from <https://docs.aws.amazon.com/cloudwatch/index.html>
- Amazon Web Services. (n.d.). *CloudWatch Alarms Documentation*. Retrieved from <https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/AlarmThatSendsEmail.html>
- Amazon Web Services. (n.d.). *Simple Notification Service (SNS) Documentation*. Retrieved from <https://docs.aws.amazon.com/sns/index.html>