

SWE-455: Cloud Application Engineering

Homework 01:

Deploying a simple cloud-based application

Name: AYED ALQAHTANI

ID: 202023400

This report outlines the deployment of a simple Node.js server on AWS EC2 instances. The server is designed to return a basic HTML response (`<h1>Hello, User! </h1>`) when accessed. Below, I provide an overview of the deployment process, including screenshots, challenges faced.

Deployment steps

Server code

We are going to make a simple NodeJS server that listens to http requests and shows (hello users!)



First, we I created a simple NodeJS server with simple endpoint

```
index.js > ...
1  const http = require('http');
2
3  const server = http.createServer((req, res) => {
4    res.statusCode = 200;
5    res.setHeader('Content-Type', 'text/html');
6    res.end('<h1>Hello, User!</h1>');
7  });
8
9  const port = 80;
10 server.listen(port, () => {
11   console.log(`Server running at port:${port}/`);
12 });
```

Then I created a git repository to access it later when we created the virtual machine.

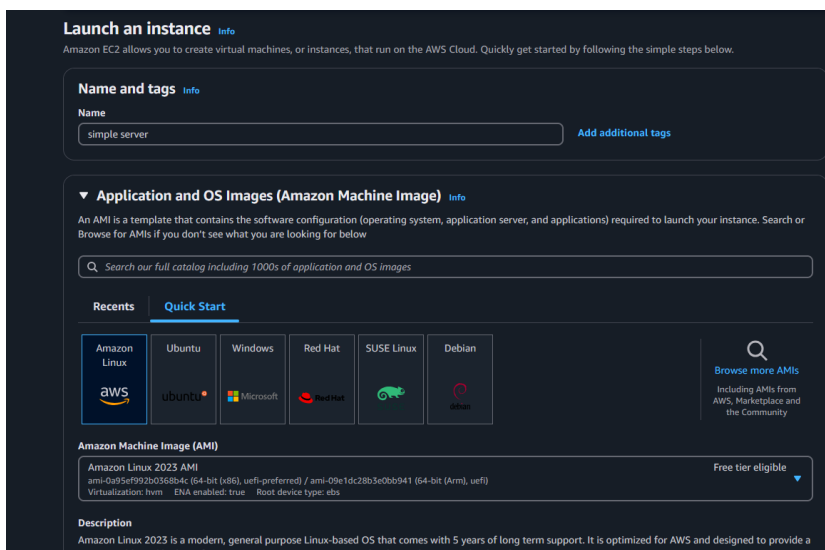
Virtual machine code

```
package.json  script.sh M x
script.sh
1  #!/bin/bash
2  # Update the system
3  yum update -y
4
5  # Install Node.js and npm
6  curl -sL https://rpm.nodesource.com/setup_16.x | bash -
7  yum install -y nodejs
8
9  # Install Git
10 yum install -y git
11
12 # Clone the Git repository
13 git clone https://github.com/ayed87/simple_server.git /home/ec2-user/myapp
14
15 # Navigate to the app directory
16 cd /home/ec2-user/myapp
17
18 # Install dependencies
19 npm install
20
21 # Start the Node.js server
22 node index.js &
```

This code will be run when we create our virtual machine.

Create AWS EC2 virtual machine steps

Configuring EC2 using AWS image



Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-0a95ef992b0368b4c (64-bit (x86), uefi-preferred) / ami-09a1dc28b3e0bb941 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.6.20250218.2 x86_64 HVM kernel-6.1

Architecture 64-bit (x86) **Boot mode** uefi-preferred **AMI ID** ami-0a95ef992b0368b4c **Username** ec2-user **Verified provider**

Instance type **Info** | **Get advice**

Instance type t3.micro **Free tier eligible**
Family: t3 2 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0217 USD per Hour On-Demand Ubuntu Pro base pricing: 0.016 USD per Hour
On-Demand Linux base pricing: 0.0125 USD per Hour On-Demand RHEL base pricing: 0.0413 USD per Hour
On-Demand SUSE base pricing: 0.0125 USD per Hour

Additional costs apply for AMIs with pre-installed software

Key pair (login) **Info**

Network settings **Info** **Edit**

Network **Info**
vpc-0f02065720a1d9c3e

Subnet **Info**
No preference (Default subnet in any availability zone)

Auto-assign public IP **Info**
Enable
Additional charges apply when outside of free tier allowance

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

We'll create a new security group called 'launch-wizard-1' with the following rules:

☒ Allow SSH traffic from **Anywhere**
Helps you connect to your instance
0.0.0.0/0

☐ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Here we should allow traffic from HTTP (80)

Metadata response hop limit **Info**
2

Allow tags in metadata **Info**
Select

User data - optional **Info**
Upload a file with your user data or enter it in the field.

☒ Choose file

```
#!/bin/bash
# Update the system
yum update -y

# Install Node.js and npm
curl -sL https://rpm.nodesource.com/setup_16.x | bash -
yum install -y nodejs

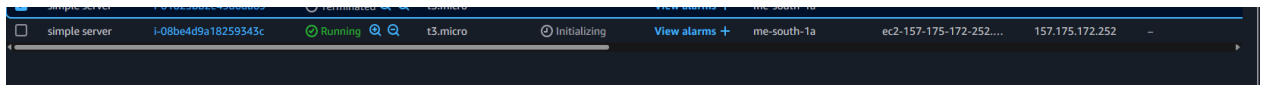
# Install Git
yum install -y git

# Clone the Git repository
git clone https://github.com/ayed87/simple_server.git /home/ec2-user/myapp

# Navigate to the app directory
```

☐ User data has already been base64 encoded

Here we add the script that we wrote before



Now the server is running, and we can access through internet using its public IP address



Scalability

In order to make the EC2 scalable, we should create an autoscaling group

Configuring a template for our autoscaling group:

Create launch template

Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.

Launch template name and description

Launch template name - *required*

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '*', '@'.

Template version description

Max 255 chars

Auto Scaling guidance | [Info](#)

Select this if you intend to use this template with EC2 Auto Scaling

☒ Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

▶ **Template tags**

▶ **Source template**

Launch template contents

Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

▼ **Application and OS Images (Amazon Machine Image) - required** | [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Choose launch template or configuration info

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group. If you currently use launch configurations, you might consider migrating to launch templates.

Name

Auto Scaling group name

Enter a name to identify the group.

example

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

Launch template info

[Switch to launch configuration](#)

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.

SimpleServerTemplate

[Create a launch template](#)

Version

Default (1)

[Create a launch template version](#)

Description

SWE455 homework1

AMI ID

ami-0a95ef992b0368b4c

Key pair name

-

Launch template

[SimpleServerTemplate](#)

lt-0942e5a88c9391ec4

Security groups

-

Security group IDs

[sg-0ad9e7061aab6752c](#)

Instance type

t3.micro

Request Spot Instances

No

Additional details

Choose instance launch options info

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

Instance type requirements info

[Override launch template](#)

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

[SimpleServerTemplate](#)

lt-0942e5a88c9391ec4

Version

Default

Description

SWE455 homework1

Instance type

t3.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.

vpc-0f02065720a1d9c3e

172.31.0.0/16 Default

[Create a VPC](#)

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

me-south-1a | subnet-03af311432a792a0e

172.31.0.0/20 Default

me-south-1b | subnet-01793a38dade31e3f

172.31.16.0/20 Default

me-south-1c | subnet-0d6334e4d73efd4f4

172.31.32.0/20 Default

[Create a subnet](#)

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.

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.

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](#)

During an Availability Zone impairment, target instance launches towards other healthy Availability Zones.

☐ **Enable zonal shift**
 New instance launches will be retargeted towards healthy Availability Zones until the zonal shift is canceled.

Health checks

Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.

EC2 health checks

☒ **Always enabled**

Additional health check types - optional [Info](#)

☐ **Turn on Elastic Load Balancing health checks**
 Elastic Load Balancing monitors whether instances are available to handle requests. When it reports an unhealthy instance, EC2 Auto Scaling can replace it on its next periodic check.

☐ **Turn on VPC Lattice health checks**
 VPC Lattice can monitor whether instances are available to handle requests. If it considers a target as failed a health check, EC2 Auto Scaling replaces it after its next periodic check.

☐ **Turn on Amazon EBS health checks**
 EBS monitors whether an instance's root volume or attached volume stalls. When it reports an unhealthy volume, EC2 Auto Scaling can replace the instance on its next periodic health check.

Health check grace period [Info](#)

This time period delays the first health check until your instances finish initializing. It doesn't prevent an instance from terminating when placed into a non-running state.

300 seconds

[Cancel](#)
[Skip to review](#)
[Previous](#)
[Next](#)

For simplicity we will configure only auto scaling groups without load balancer

Scaling

Units (number of instances)

Desired capacity

Specify your group size.

3

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 **Max desired capacity**

2 5

Equal or less than desired capacity Equal or greater 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

Target Tracking Policy

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

Target value

50

Instance warmup [Info](#)

300 seconds

☐ **Disable scale in to create only a scale-out policy**

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

Here we define the minimum to 2 and the maximum to 5. And the desired (default) is 3

The template is created

example

example Capacity overview

aws:autoscaling:me-south-1:528757802738:autoScalingGroup:d64b0469-389a-4c2f-bbef-d3a4b6fbaf0:autoScalingGroupName/example

Desired capacity

3

Scaling limits (Min - Max)

2 - 5

Desired capacity type

Units (number of instances)

Status

-

Date created

Sat Feb 22 2025 04:42:31 GMT+0300 (Arabian Standard Time)

Details

Integrations - new

Automatic scaling

Instance management

Instance refresh

Activity

Monitoring

Launch template

Launch template

h-0942e5a88c9391ec4

SimpleServerTemplate

Version

Default

Description

SWE455 homework1

View details in the launch template console

AMI ID

ami-0a95ef992b0368b4c

Security groups

-

Storage (volumes)

-

Instance type

t3.micro

Security group IDs

sg-0a09b7061aab6752c

Key pair name

-

Owner

arn:aws:iam::528757802738:root

Create time

Sat Feb 22 2025 04:33:49 GMT+0300 (Arabian Standard Time)

Request Spot Instances

No

3 EC2s are running 3 which are the desired that we put

Instances (6) info

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

Find instance by attribute or tag (case-sensitive)

All states

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4	Elastic IP
<input type="checkbox"/>		i-0edd3e9d3e180454a	Running	t3.micro	3/3 checks passed	View alarms +	me-south-1c	ec2-15-185-220-195.m...	15.185.220.195	-
<input type="checkbox"/>		i-08e8557510735406e	Running	t3.micro	3/3 checks passed	View alarms +	me-south-1a	ec2-16-24-167-88.me-s...	16.24.167.88	-
<input type="checkbox"/>		i-0a8014b7778f190b6	Running	t3.micro	3/3 checks passed	View alarms +	me-south-1b	ec2-157-175-153-77.m...	157.175.153.77	-
<input type="checkbox"/>	simpler-server	i-0009301cac73251b1	Terminated	t3.micro	-	View alarms +	me-south-1a	-	-	-
<input type="checkbox"/>	simple server	i-01625bb2e49d6da69	Terminated	t3.micro	-	View alarms +	me-south-1a	-	-	-
<input type="checkbox"/>	simple server	i-06ba4f9e1805843e	Terminated	t3.micro	-	View alarms +	me-south-1a	-	-	-

Testing scalability by CloudWatch

Alarm state: In alarm

Alarm type: Metric

Actions status: Any

<input type="checkbox"/>	Name	State	Last state update (UTC)	Conditions	Actions
<input type="checkbox"/>	TargetTracking-example-AlarmLow-34815e2c-592f-47bd-ba86-d7a9c0723b63	In alarm	2025-02-22 01:55:30	CPUUtilization < 35 for 15 datapoints within 15 minutes	Actions enabled

getTracking-example-AlarmHigh-78844eb4-6a7f-4485-b614-c95a3770df3a

Alarms (2)

Search

Alarm state: Any

Alarm type: Any

Actions status: Any

Hide Auto Scaling alarms

TargetTracking-example-AlarmHigh-78844eb4-6a7f-4485-b614-c95a3770df3a

Metric alarm

OK

TargetTracking-example-AlarmLow-34815e2c-592f-47bd-ba86-d7a9c0723b63

Metric alarm

Insufficient data

TargetTracking-example-AlarmHigh-78844eb4-6a7f-4485-b614-c95a3770df3a

View

Actions

Explore related

Graph

CPUUtilization

CPUUtilization > 50 for 3 datapoints within 3 minutes

Percent

50

25.1

0.251

23:00

23:15

23:30

23:45

00:00

00:15

00:30

00:45

01:00

01:15

01:30

01:45

Click timeline to see the state change at the selected time.

In alarm

OK

Insufficient data

Disabled actions

Details

Tags

Actions

History

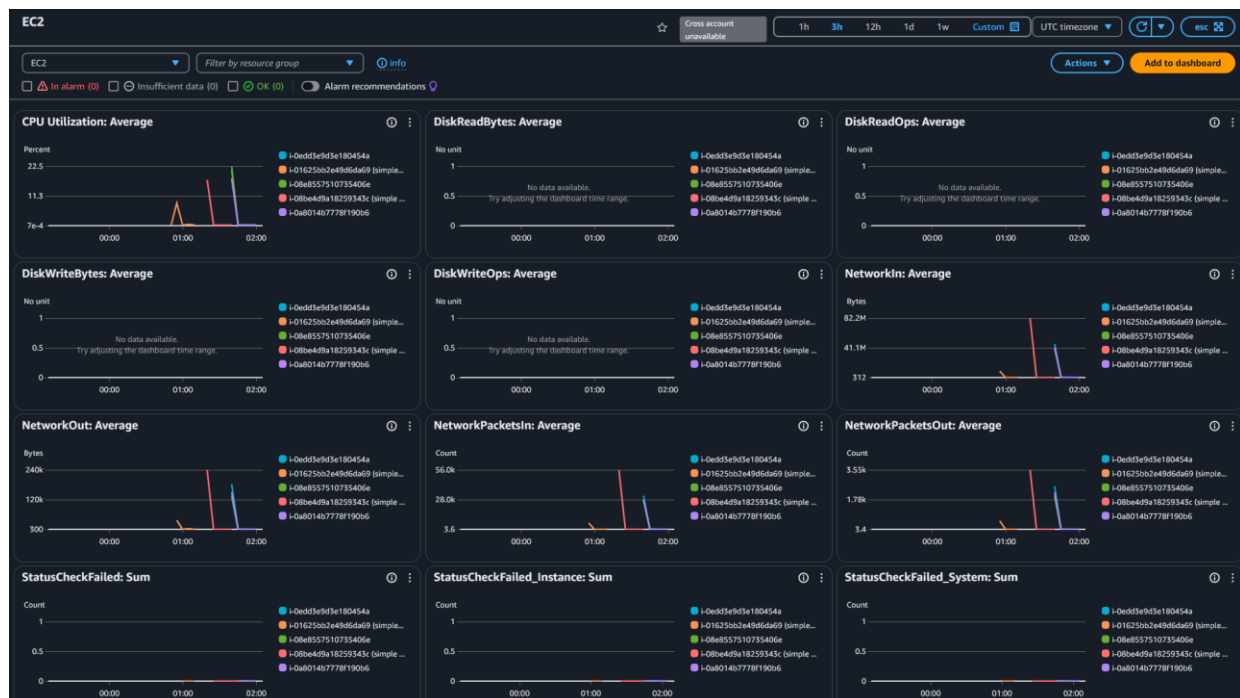
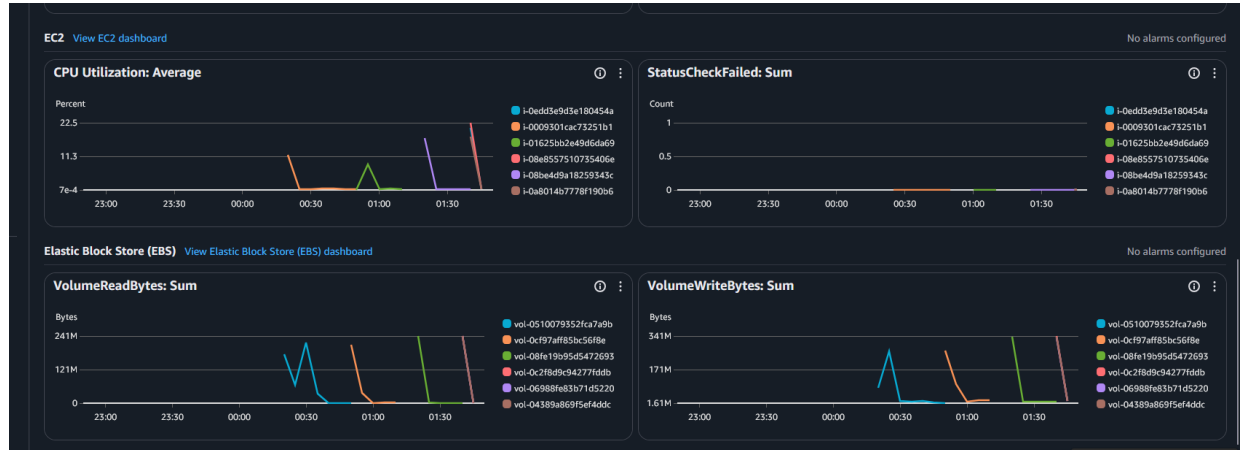
Parent alarms

Details

From the screenshots above the cloud watch catch that CPU usage is low. Hence it reduced the number of virtual machines to 2.

Instances (5) Info									
Find Instance by attribute or tag (case-sensitive)									
All states									
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
<input type="checkbox"/>		i-0edd3e9d3e180454a	Running	t3.micro	1/3 checks passed	View alarms +	me-south-1c	ec2-15-185-220-195.m...	15.185.220.195
<input type="checkbox"/>		i-0a8014b7778f190b6	Running	t3.micro	1/3 checks passed	View alarms +	me-south-1b	ec2-157-175-153-77.m...	157.175.153.77
<input type="checkbox"/>	simple server	i-01625b2e49d6da69	Terminated	t3.micro	-	View alarms +	me-south-1a	-	-

Other monitoring mastics in cloud watch dashboard



Challenges

Security Group Configuration:

Initially, the server was inaccessible due to the security group inbound rule. It did not allow inbound traffic on port 80. This was resolved by updating the security group rules to accept traffic from port 80.

Port is not correctly configured in the server code:

Initially, the server code was listening to port 3000 which is a default port when developing the NodeJS server. I fixed this issue by changing it to 80.