3-Tier Project

Antoine Boylston

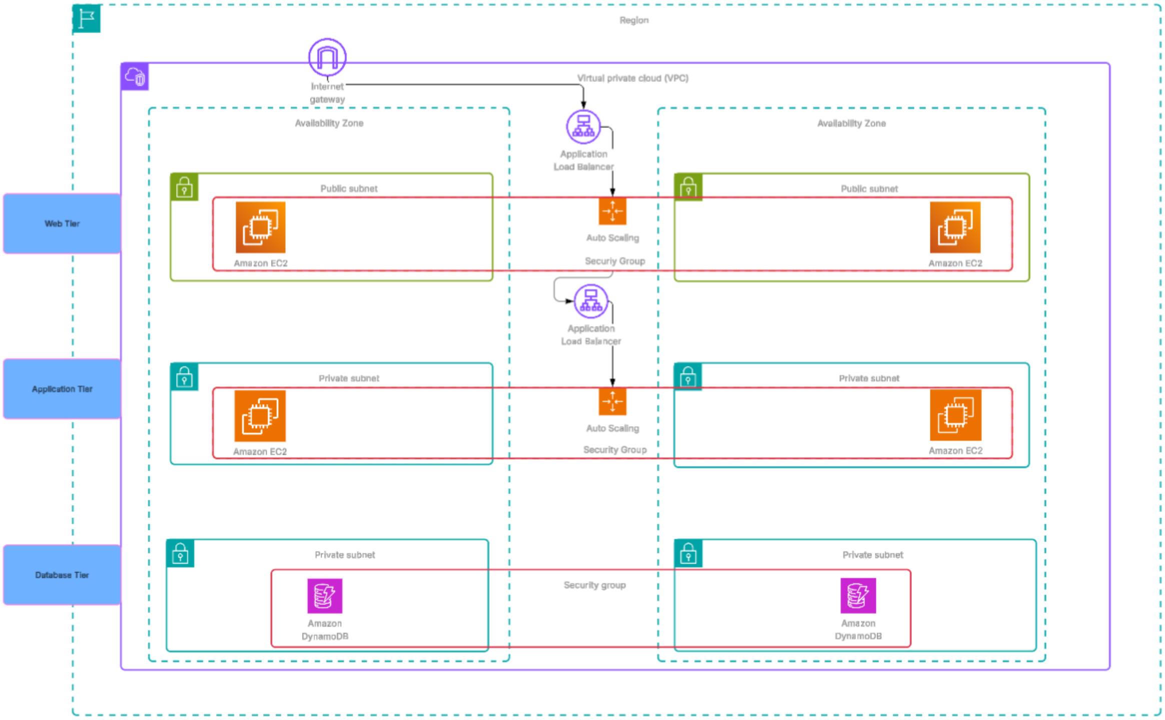
Introduction

In this project, I will be building a **3-tier architecture on AWS** to demonstrate my ability to design and deploy scalable, secure, and resilient cloud solutions. A 3-tier architecture is a well-established design pattern that separates an application into three distinct layers, each responsible for a specific set of tasks.

- **Presentation Tier** – the front-end layer that users interact with, typically delivered through a web interface or mobile client.
- **Application Tier** – the middle layer that contains the business logic, processes requests, and connects the front-end to the data.
- **Database Tier** – the back-end layer that manages data storage, retrieval, and integrity.

By clearly separating these tiers, applications gain **improved scalability, maintainability, and security**, since each tier can be developed, deployed, and scaled independently. On AWS, this design takes advantage of services that align naturally with each tier, allowing for a modern cloud-native implementation.

My goal for this project is not only to build a functioning 3-tier architecture, but also to showcase my ability to apply industry best practices in cloud architecture, leverage AWS services effectively, and think critically about designing solutions that are reliable and efficient.

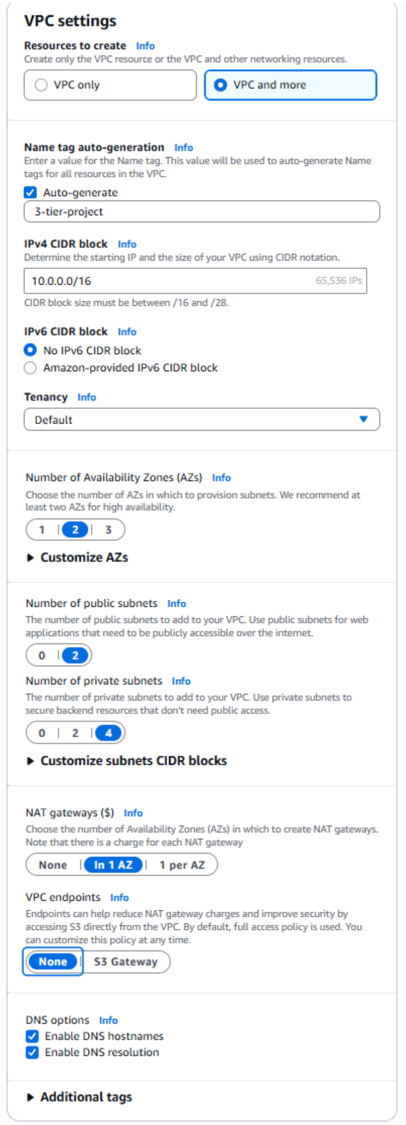


Phase 1: Networking

The first step was to build the VPC. An Amazon Virtual Private Cloud (VPC) is a logically isolated section of the AWS cloud where you can launch and manage resources such as servers, databases, and applications in a secure environment. It gives you complete control over your networking setup, including IP address ranges, subnets, route tables, and security settings.

With a VPC, you can design a network architecture like a traditional data center but with the flexibility and scalability of the cloud. This allows you to define **public subnets** for resources that need internet access, **private subnets** for secure internal resources, and use features like **security groups** and **network ACLs** to tightly control traffic flow.

I like to think of VPC as my own private command center in the AWS cloud, where I can setup and control a virtual network just like a traditional data center. It's the foundation that lets me decide which resources are public, which are private, and how they securely communicate with each other.



VPC settings

Resources to create Info

Create only the VPC resource or the VPC and other networking resources.

○ VPC only ● VPC and more

Name tag auto-generation Info

Enter a value for the Name tag. This value will be used to auto-generate Name tags for all resources in the VPC.

☑ Auto-generate

3-tier-project

IPv4 CIDR block Info

Determine the starting IP and the size of your VPC using CIDR notation.

10.0.0.0/16 65,536 IPs

CIDR block size must be between /16 and /28.

IPv6 CIDR block Info

● No IPv6 CIDR block

○ Amazon-provided IPv6 CIDR block

Tenancy Info

Default ▼

Number of Availability Zones (AZs) Info

Choose the number of AZs in which to provision subnets. We recommend at least two AZs for high availability.

1 | [✓ 2] | 3

▶ Customize AZs

Number of public subnets Info

The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet.

0 | [✓ 2]

Number of private subnets Info

The number of private subnets to add to your VPC. Use private subnets to secure backend resources that don't need public access.

0 | 2 | [✓ 4]

▶ Customize subnets CIDR blocks

NAT gateways ($) Info

Choose the number of Availability Zones (AZs) in which to create NAT gateways. Note that there is a charge for each NAT gateway

None | [✓ In 1 AZ] | 1 per AZ

VPC endpoints Info

Endpoints can help reduce NAT gateway charges and improve security by accessing S3 directly from the VPC. By default, full access policy is used. You can customize this policy at any time.

[✓ None] | S3 Gateway

DNS options Info

☑ Enable DNS hostnames

☑ Enable DNS resolution

▶ Additional tags

Figure 1/ VPC Settings

I began by navigating to the VPC dashboard and creating my VPC. For this project, I added only an IPv4 CIDR block (10.0.0.0/16) to define the address space. To ensure high availability and fault tolerance, I selected two Availability Zones (AZs) to host my subnets.

Within the VPC, I created a total of six subnets:

- *Two public subnets for the presentation (web) tier, which require internet access.*
- *Two private subnets for the application tier, where the business logic will run securely.*
- *Two private subnets for the database tier, where sensitive data will be stored and isolated from public access.*

This structure ensures that each layer of my 3-tier architecture is properly segmented, secure, and distributed across multiple AZs for resiliency.

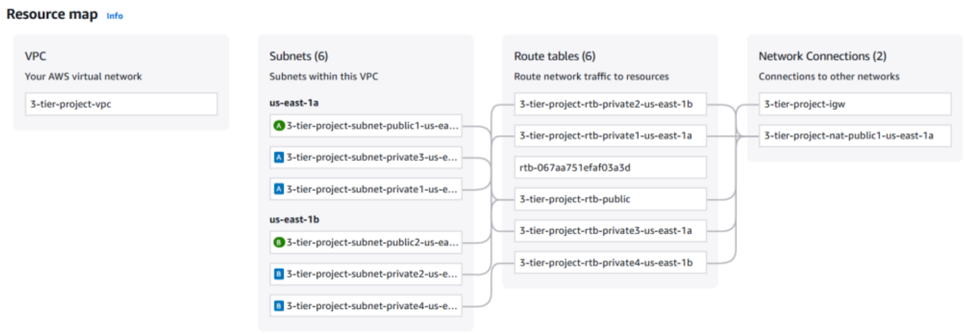

Resource map
Info
VPC
Your AWS virtual network
3-tier-project-vpc
Subnets (6)
Subnets within this VPC
us-east-1a
A 3-tier-project-subnet-public1-us-ea...
A 3-tier-project-subnet-private3-us-e...
A 3-tier-project-subnet-private1-us-e...
us-east-1b
B 3-tier-project-subnet-public2-us-ea...
B 3-tier-project-subnet-private2-us-e...
B 3-tier-project-subnet-private4-us-e...
Route tables (6)
Route network traffic to resources
3-tier-project-rtb-private2-us-east-1b
3-tier-project-rtb-private1-us-east-1a
rtb-067aa751efaf03a3d
3-tier-project-rtb-public
3-tier-project-rtb-private3-us-east-1a
3-tier-project-rtb-private4-us-east-1b
Network Connections (2)
Connections to other networks
3-tier-project-igw
3-tier-project-nat-public1-us-east-1a

Figure 2/ Resource Map

After creating the VPC, I navigated to the Subnets section. Once I verified that all six subnets had been created, I configured each public subnet to auto-assign public IPv4 addresses. This ensures that resources launched in the public subnets, such as the web servers in the presentation tier, can be assigned a public IP automatically and be directly accessible through the internet.

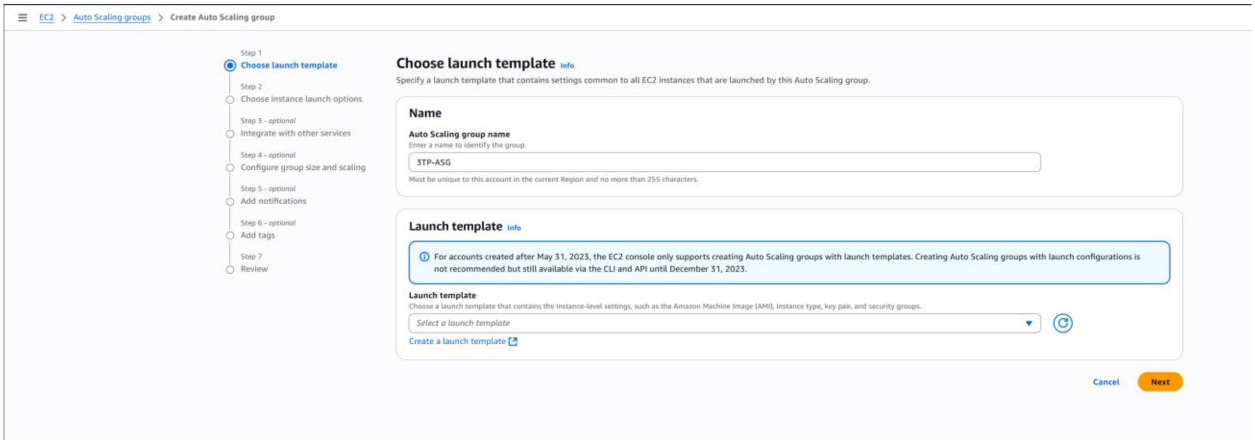
Phase 2: Presentation Tier

To build the presentation layer of my 3-tier architecture, I will be launching **Amazon EC2 instances** inside an **Auto Scaling Group**.

Amazon Elastic Compute Cloud (EC2) is a web service that provides secure, resizable compute capacity in the cloud. It allows you to launch virtual servers, configure the operating system and applications, and pay only for the compute time you use. With EC2, you have full control over your instances—similar to having your own server in a data center—but with the scalability and flexibility of AWS.

An **Auto Scaling Group (ASG)** is a feature that ensures your application has the right number of EC2 instances running to handle incoming traffic. It automatically adds instances when demand increases and terminates them when demand decreases, helping maintain performance while optimizing costs. ASGs also improve fault tolerance by distributing instances across multiple Availability Zones and replacing unhealthy ones automatically.

For this project, I launched my Auto Scaling Group into the **two public subnets** I created in Phase 1. By placing the EC2 instances in public subnets, they can receive traffic from the internet, forming the **web tier** of my application. This setup ensures that the presentation layer is **highly available, fault-tolerant, and capable of scaling** in response to user demand.



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

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.
3TP-ASG
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.

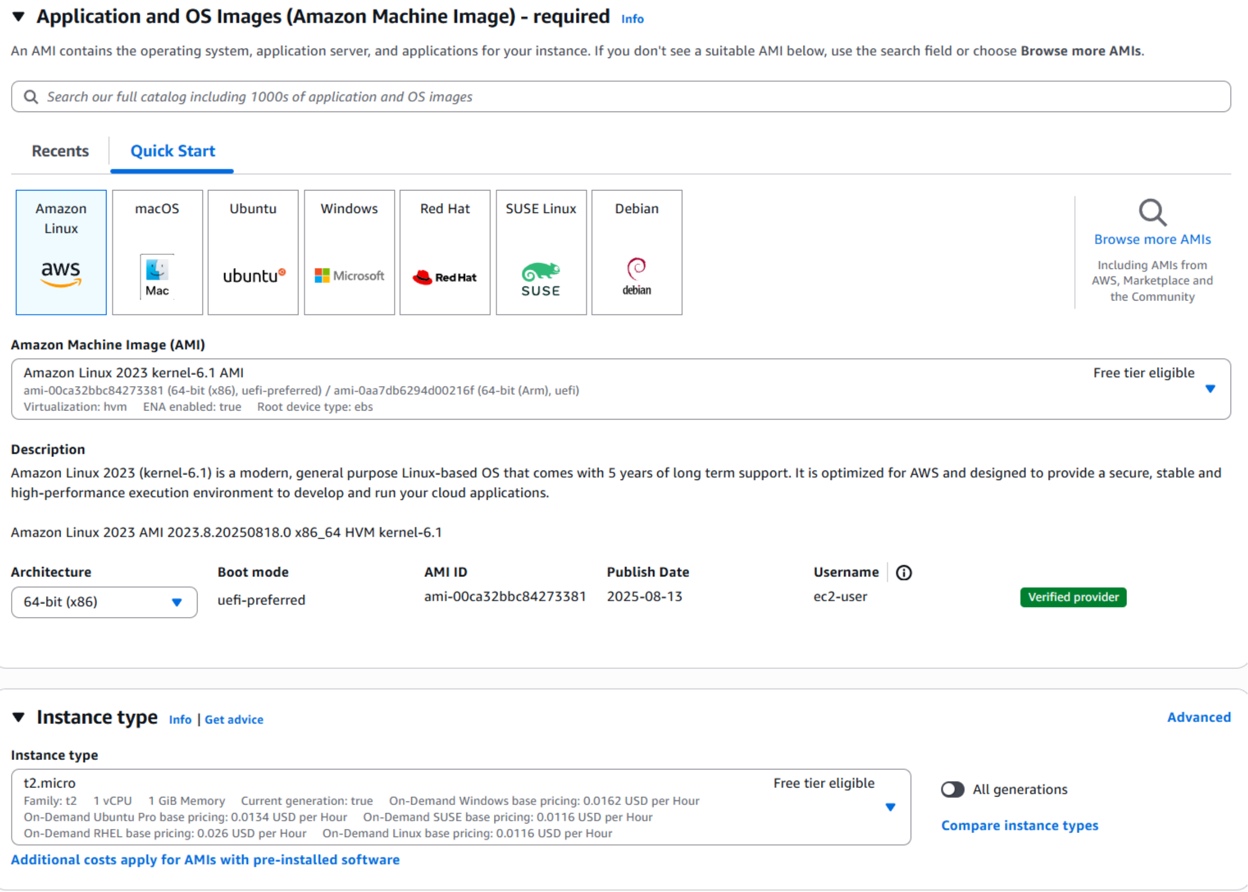
Launch template
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.
Select a launch template
Create a launch template

Cancel Next

Figure 3/ Create Auto Scaling group

*I began by navigating to the **EC2 dashboard** and selecting **Auto Scaling Groups** from the left-hand menu. From there, I chose **Create Auto Scaling Group**. This started the process of building the template that defines how my Auto Scaling Group will launch EC2 instances.*

*In this template, I specified the **pre-configurations** needed for my **public-facing EC2 instances**, ensuring that each instance launched by the ASG would be consistent and properly set up to serve as part of the web tier.*



Application and OS Images (Amazon Machine Image) - required Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

Search our full catalog including 1000s of application and OS images

Recents | Quick Start

Amazon Machine Image (AMI)
Amazon Linux 2023 kernel-6.1 AMI — Free tier eligible
ami-00ca32bbc84273381 (64-bit (x86), uefi-preferred) / ami-0aa7db6294d00216f (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Amazon Linux 2023 (kernel-6.1) 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 2023.8.20250818.0 x86_64 HVM kernel-6.1

Architecture	Boot mode	AMI ID	Publish Date	Username	
64-bit (x86)	uefi-preferred	ami-00ca32bbc84273381	2025-08-13	ec2-user	Verified provider

Instance type Info | Get advice — Advanced

Instance type
t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour On-Demand Linux base pricing: 0.0116 USD per Hour

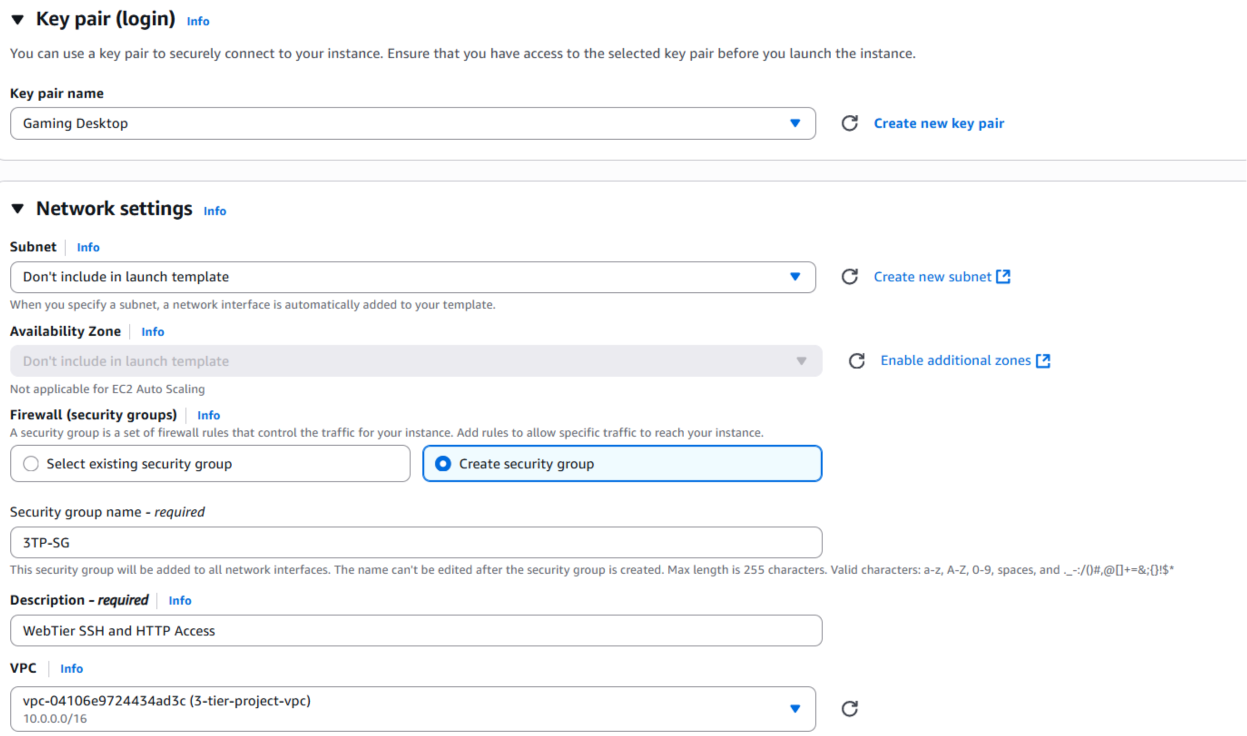
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software

Figure 4/ AMI & Instance type

As part of creating the launch template, I needed to define two key settings: the Amazon Machine Image (AMI) and the instance type.

Amazon Machine Image (AMI): An AMI is a preconfigured template that contains the operating system, application server, and any additional software needed to launch an EC2 instance. It serves as the "blueprint" for your virtual server. For this project, I selected the Amazon Linux 2023 (kernel-6.1) AMI, which is a lightweight, secure, and AWS-optimized operating system.

Instance Type: The instance type defines the hardware configuration of the EC2 instance, including CPU, memory, storage, and networking capacity. For my web tier, I chose a t2.micro instance type. This option provides a cost-effective way to run general-purpose workloads within the AWS Free Tier, making it ideal for this demonstration project.



▼ **Key pair (login)**

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name

Gaming Desktop

▼ **Network settings**

Subnet

Don't include in launch template

When you specify a subnet, a network interface is automatically added to your template.

Availability Zone

Don't include in launch template

Not applicable for EC2 Auto Scaling

Firewall (security groups)

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.

○ Select existing security group ● Create security group

Security group name - *required*

3TP-SG

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required*

WebTier SSH and HTTP Access

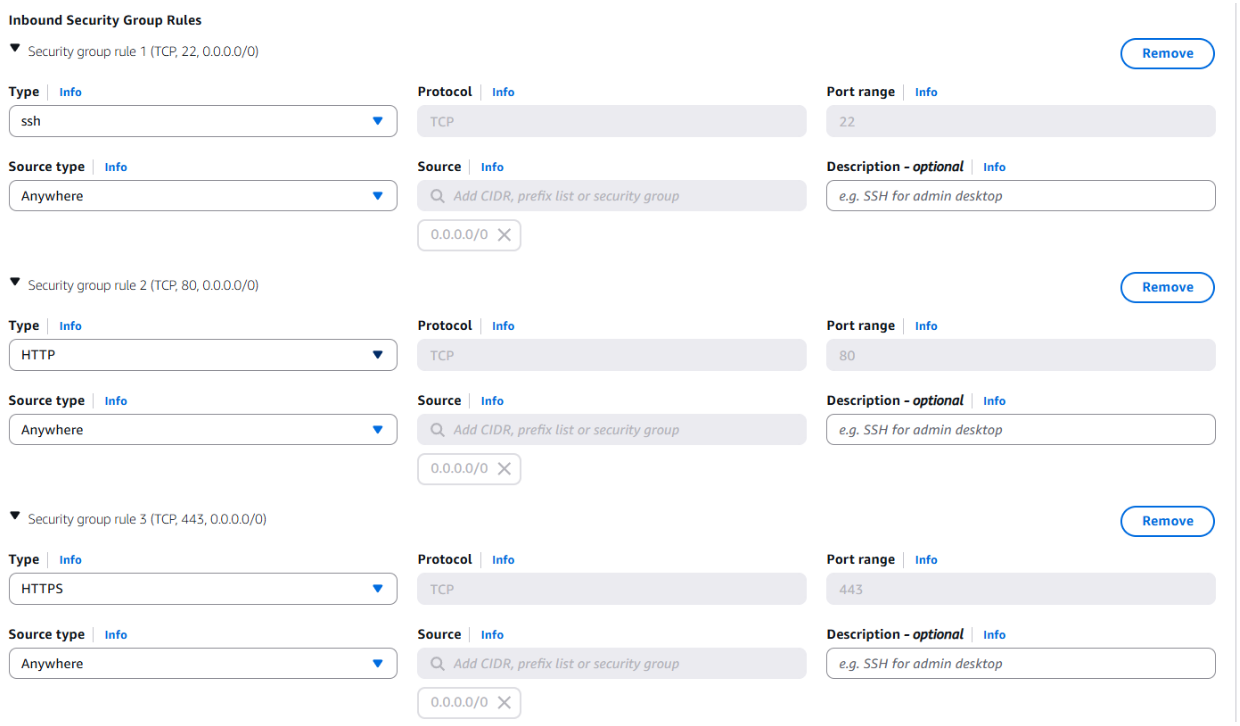
VPC

vpc-04106e9724434ad3c (3-tier-project-vpc)
10.0.0.0/16

Figure 5/ Key pair & Security Group

- ***Key Pair:*** *I selected a key pair to enable secure SSH access to the EC2 instances that will be launched by my Auto Scaling Group. Key pairs provide an authentication method, ensuring only authorized users can connect to the servers.*

- ***Security Group:*** *At this stage, I created a new **security group** for my VPC. A security group acts as a virtual firewall, controlling inbound and outbound traffic to the EC2 instances.*



Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type: ssh	Protocol: TCP	Port range: 22
Source type: Anywhere	Source: 0.0.0.0/0	Description - *optional*: e.g. SSH for admin desktop

▼ Security group rule 2 (TCP, 80, 0.0.0.0/0)

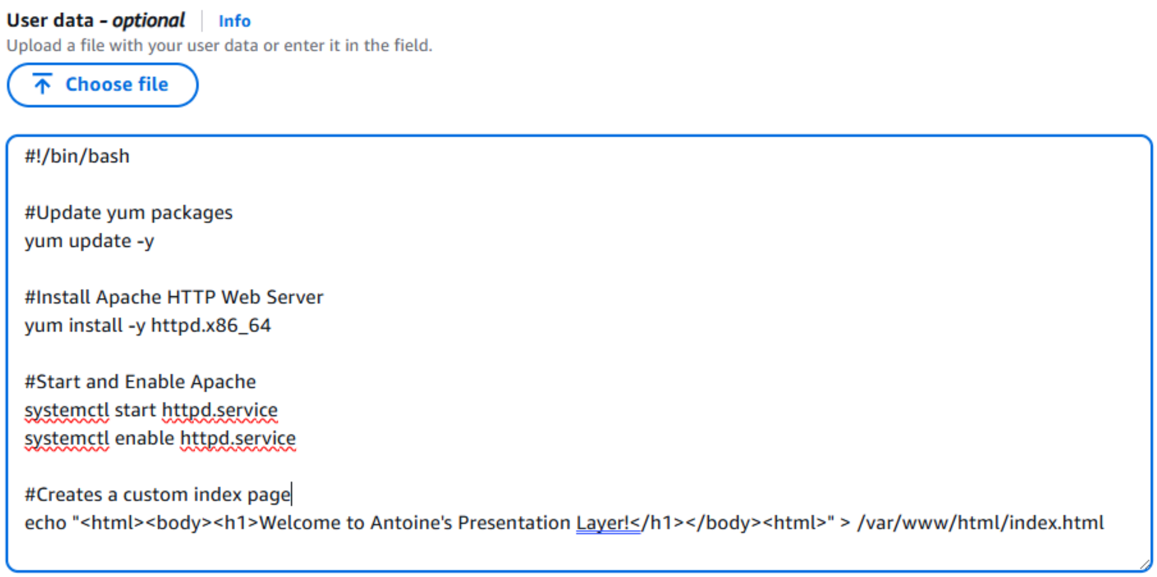
Type: HTTP	Protocol: TCP	Port range: 80
Source type: Anywhere	Source: 0.0.0.0/0	Description - *optional*: e.g. SSH for admin desktop

▼ Security group rule 3 (TCP, 443, 0.0.0.0/0)

Type: HTTPS	Protocol: TCP	Port range: 443
Source type: Anywhere	Source: 0.0.0.0/0	Description - *optional*: e.g. SSH for admin desktop

Figure 6/ Network settings

I configured inbound rules that allow SSH, HTTP, and HTTPS access to the public facing EC2 Instances from anywhere.



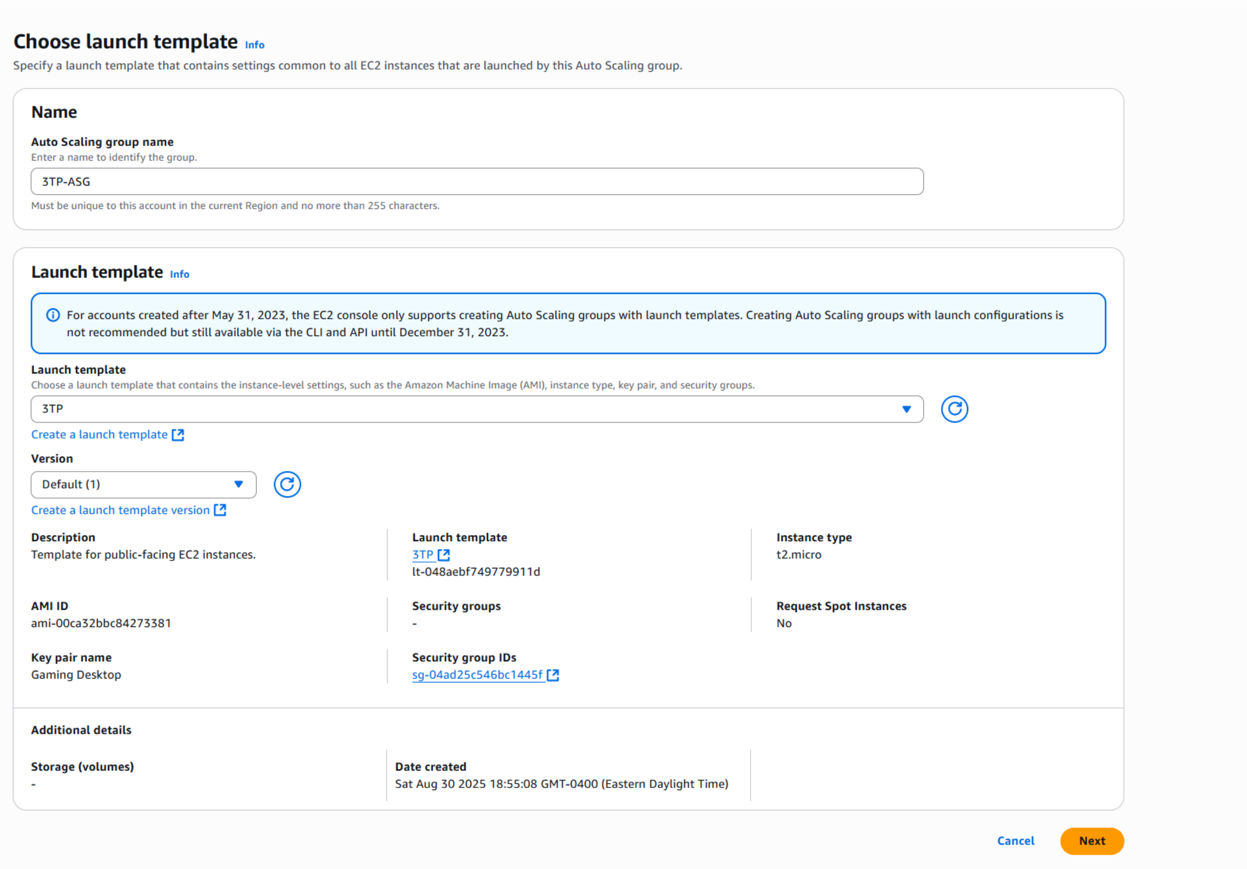
User data - *optional* Info

Upload a file with your user data or enter it in the field.

```
#!/bin/bash

#Update yum packages
yum update -y

#Install Apache HTTP Web Server
yum install -y httpd.x86_64

#Start and Enable Apache
systemctl start httpd.service
systemctl enable httpd.service

#Creates a custom index page
echo "<html><body><h1>Welcome to Antoine's Presentation Layer!</h1></body><html>" > /var/www/html/index.html
```

Figure 7/ Script

*To make sure every EC2 instance launched by the Auto Scaling Group is ready to serve traffic, I added a **User Data script**. This script automatically installs and configures the Apache web server, starts the service, and creates a custom* *index.html page. By including this script, each new instance comes online as a fully configured web server without requiring any manual setup.*



Choose launch template

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.
3TP-ASG
Must be unique to this account in the current Region and no more than 255 characters.

Launch template

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.
3TP

Version
Default (1)

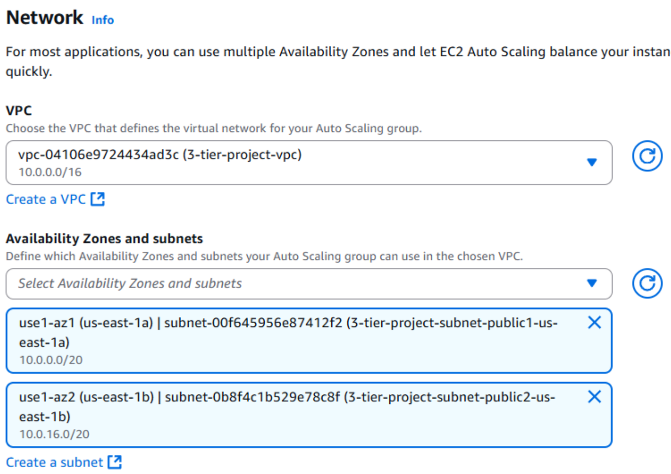
Description Template for public-facing EC2 instances.	**Launch template** 3TP lt-048aebf749779911d	**Instance type** t2.micro
AMI ID ami-00ca32bbc84273381	**Security groups** -	**Request Spot Instances** No
Key pair name Gaming Desktop	**Security group IDs** sg-04ad25c546bc1445f	

Additional details

Storage (volumes) -	**Date created** Sat Aug 30 2025 18:55:08 GMT-0400 (Eastern Daylight Time)

Figure 8/ Select Launch template ASG

After the launch template was successfully created I went back to the ASG window and selected the template just created.



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-04106e9724434ad3c (3-tier-project-vpc)
10.0.0.0/16

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

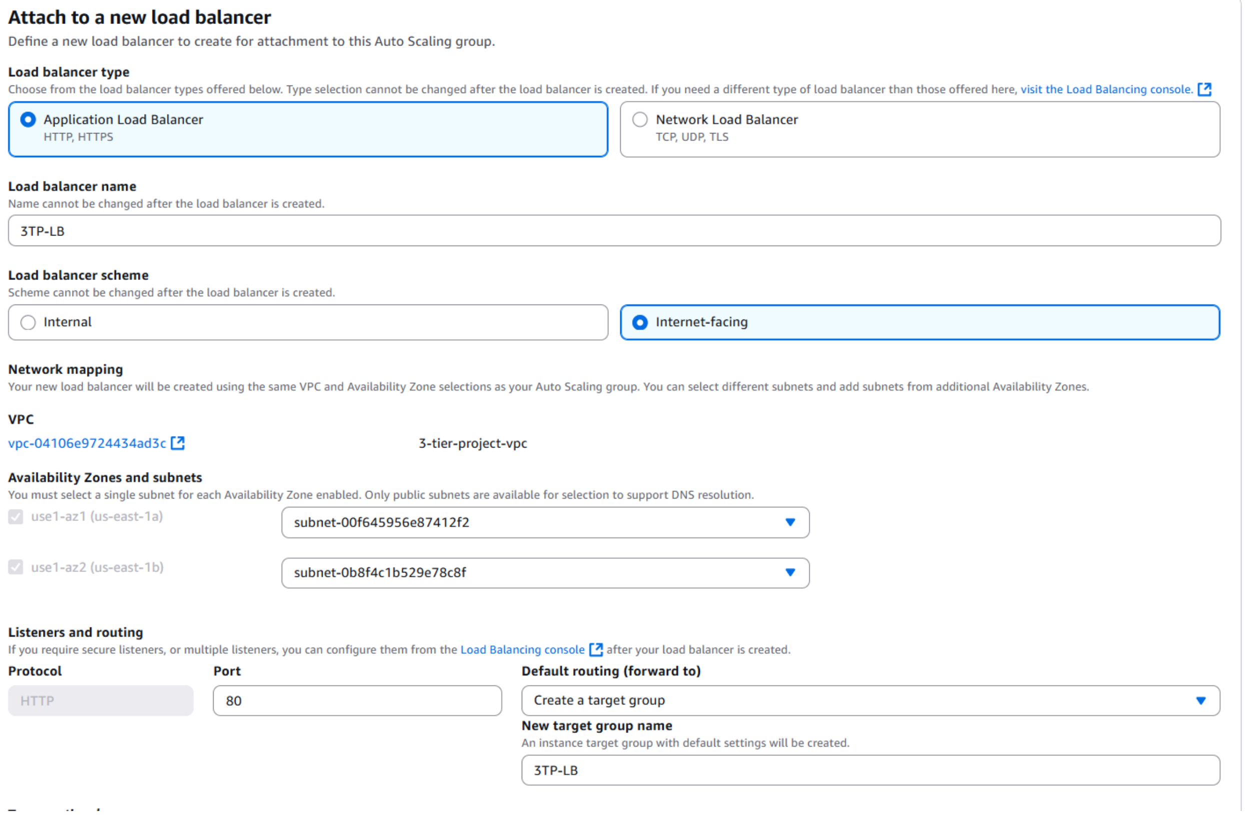
use1-az1 (us-east-1a) | subnet-00f645956e87412f2 (3-tier-project-subnet-public1-us-east-1a)
10.0.0.0/20

use1-az2 (us-east-1b) | subnet-0b8f4c1b529e78c8f (3-tier-project-subnet-public2-us-east-1b)
10.0.16.0/20

Create a subnet

Figure 9/ ASG Network info

*I selected the **VPC I created earlier** and chose **two public subnets across different Availability Zones (AZs)**. This ensures that the web tier instances launched by the Auto Scaling Group are publicly accessible and distributed across multiple AZs for **high availability and fault tolerance**.*



Attach to a new load balancer
Define a new load balancer to create for attachment to this Auto Scaling group.

Load balancer type
Choose from the load balancer types offered below. Type selection cannot be changed after the load balancer is created. If you need a different type of load balancer than those offered here, visit the Load Balancing console.

● Application Load Balancer
HTTP, HTTPS

○ Network Load Balancer
TCP, UDP, TLS

Load balancer name
Name cannot be changed after the load balancer is created.

3TP-LB

Load balancer scheme
Scheme cannot be changed after the load balancer is created.

○ Internal

● Internet-facing

Network mapping
Your new load balancer will be created using the same VPC and Availability Zone selections as your Auto Scaling group. You can select different subnets and add subnets from additional Availability Zones.

VPC
vpc-04106e9724434ad3c 3-tier-project-vpc

Availability Zones and subnets
You must select a single subnet for each Availability Zone enabled. Only public subnets are available for selection to support DNS resolution.

☑ use1-az1 (us-east-1a) subnet-00f645956e87412f2

☑ use1-az2 (us-east-1b) subnet-0b8f4c1b529e78c8f

Listeners and routing
If you require secure listeners, or multiple listeners, you can configure them from the Load Balancing console after your load balancer is created.

Protocol	Port	Default routing (forward to)
HTTP	80	Create a target group

New target group name
An instance target group with default settings will be created.

3TP-LB

Figure 10/ Load Balancing

*At this step, I attached my Auto Scaling Group to a new **Application Load Balancer (ALB)**. An ALB distributes incoming HTTP/HTTPS traffic across multiple EC2 instances, ensuring high availability and fault tolerance for the web tier.*

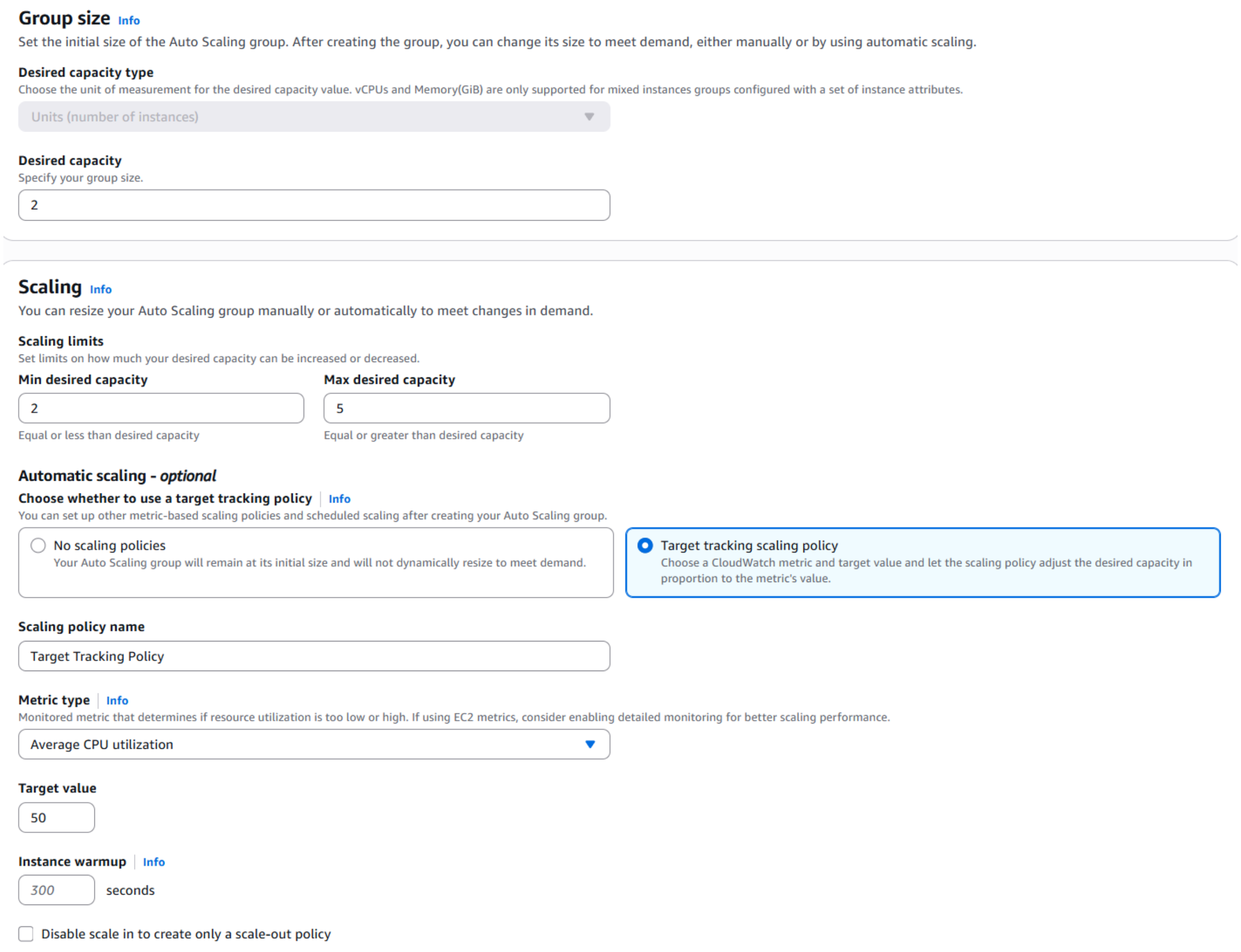
- ***Load Balancer Type:*** *I selected* ***Application Load Balancer****, which is best suited for web applications since it operates at the application layer (Layer 7) and supports routing based on HTTP/HTTPS requests.*

- ***Load Balancer Name:*** *I named it* ***3TP-LB*** *to represent the web tier of my 3-tier project.*

- ***Scheme:*** *I set the load balancer to* ***Internet-facing****, making it accessible from the public internet.*

- ***Network Mapping:*** *I chose the same* ***VPC*** *I created earlier and mapped the load balancer to* ***two public subnets across different Availability Zones****. This ensures that the ALB itself is redundant and highly available.*

By placing the load balancer in front of the Auto Scaling Group, incoming traffic is automatically distributed across healthy EC2 instances, improving both reliability and scalability of the presentation tier.



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.

2

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: 2 — Equal or less than desired capacity

Max desired capacity: 5 — 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

Figure 11/ ASG Size and Scaling Policy

Here I defined the ***capacity settings*** *and configured the* ***scaling policy*** *for my Auto Scaling Group.*

- ***Desired Capacity:*** *I set the initial desired capacity to* ***2 instances****, ensuring that two EC2 instances will always be running to handle incoming traffic.*

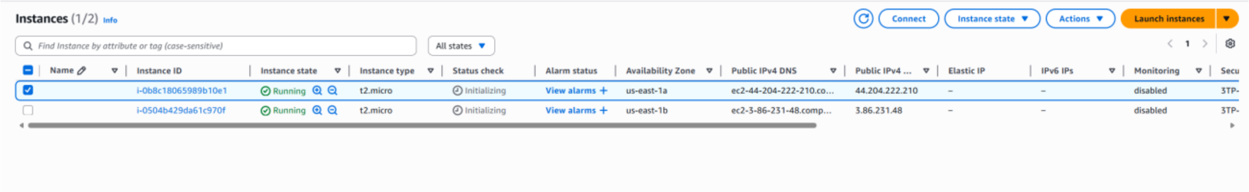
- ***Scaling Limits:*** *I configured a* ***minimum capacity of 2*** *and a* ***maximum capacity of 5*** *instances. This means the ASG will never scale below 2 (to maintain high availability) or above 5 (to control costs).*

- ***Scaling Policy:*** *I enabled a **Target Tracking Scaling Policy** based on **Average CPU Utilization**. The target value is set to **50%**, so the ASG will automatically add or remove instances to keep average CPU usage around this threshold.*

- ***Instance Warmup:*** *I kept the default warmup time of **300 seconds** to allow new instances to initialize before being considered for scaling decisions.*

- ***CloudWatch Metrics (Not Shown):*** *I also enabled **group metrics collection** within **Amazon CloudWatch**, which provides detailed visibility into ASG performance (e.g., instance count, CPU utilization, and scaling events). This monitoring helps validate that scaling actions are happening as expected.*

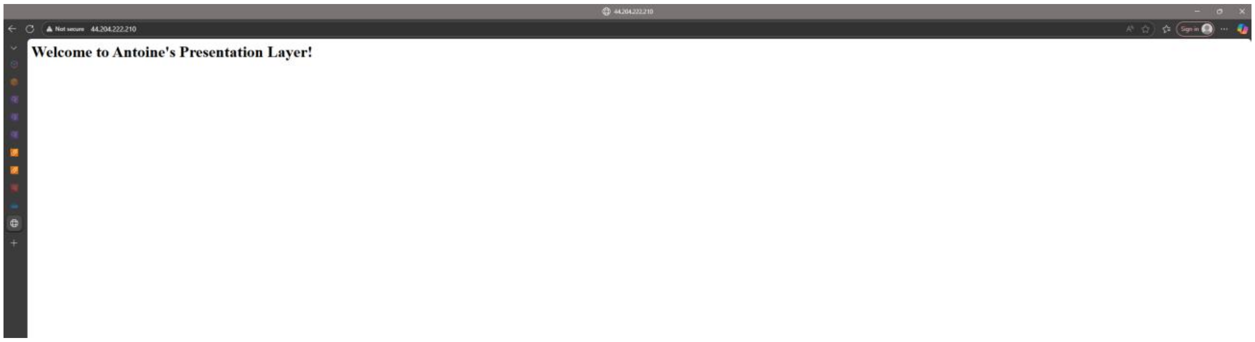
By applying this policy, the Auto Scaling Group can dynamically adjust capacity in response to demand, ensuring that the web tier is always responsive while also being cost-efficient.



☐	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Secu
☑		i-0b8c18065989b10e1	Running	t2.micro	Initializing	View alarms +	us-east-1a	ec2-44-204-222-210.co...	44.204.222.210	–	–	disabled	3TP-
☐		i-0504b429da61c970f	Running	t2.micro	Initializing	View alarms +	us-east-1b	ec2-3-86-231-48.comp...	3.86.231.48	–	–	disabled	3TP-

Figure 12/ EC2 Instances

*After creating the Auto Scaling Group, it took a few minutes for the service to update capacity and launch the EC2 instances. Once the scaling actions completed, I confirmed that **two EC2 instances** had been successfully launched in the public subnets across different Availability Zones.*



Welcome to Antoine's Presentation Layer!

Figure 13/ Website

*To test the setup, I copied the **public IPv4 address** of one of the EC2 instances and pasted it into my browser. This shows the request was successfully routed to the instance, and I was greeted with the custom **"Welcome to Antoine's Presentation Layer!"** webpage that was created by my User Data script.*

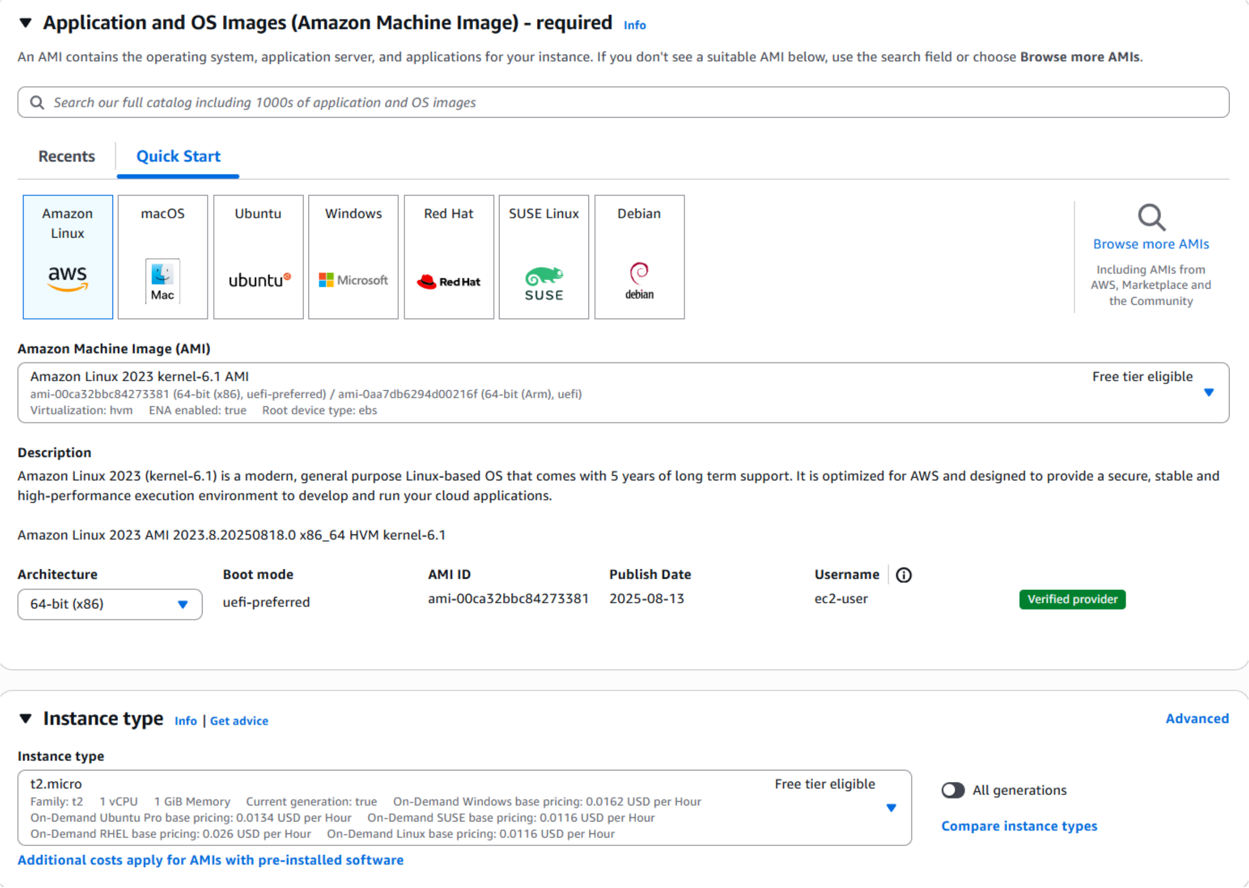
*This confirmed that the EC2 instances in the Auto Scaling Group are not only launching correctly but are also **serving web traffic as intended**.*

Phase: 3 Application Tier

With the web tier in place, the next step in building my 3-tier architecture is the **Application Tier**. This layer contains the business logic that processes requests from the web tier and connects to the database tier. To deploy it, I created a second **Auto Scaling Group (ASG)** that launches **EC2 instances in my two private subnets**.

Unlike the web tier, these instances are not publicly accessible. Instead, they are designed to only accept inbound traffic from the web tier's security group. This separation adds a layer of security while ensuring that the application servers can scale dynamically based on demand.

I like to think of the application tier as the **"engine room"** of the architecture. It's the layer that takes user requests from the front-end, applies the core logic, and then communicates securely with the database tier for any required data.



▼ Application and OS Images (Amazon Machine Image) - required Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI — Free tier eligible
ami-00ca32bbc84273381 (64-bit (x86), uefi-preferred) / ami-0aa7db6294d00216f (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 (kernel-6.1) 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 2023.8.20250818.0 x86_64 HVM kernel-6.1

Architecture	Boot mode	AMI ID	Publish Date	Username	
64-bit (x86)	uefi-preferred	ami-00ca32bbc84273381	2025-08-13	ec2-user	Verified provider

▼ Instance type Info | Get advice — Advanced

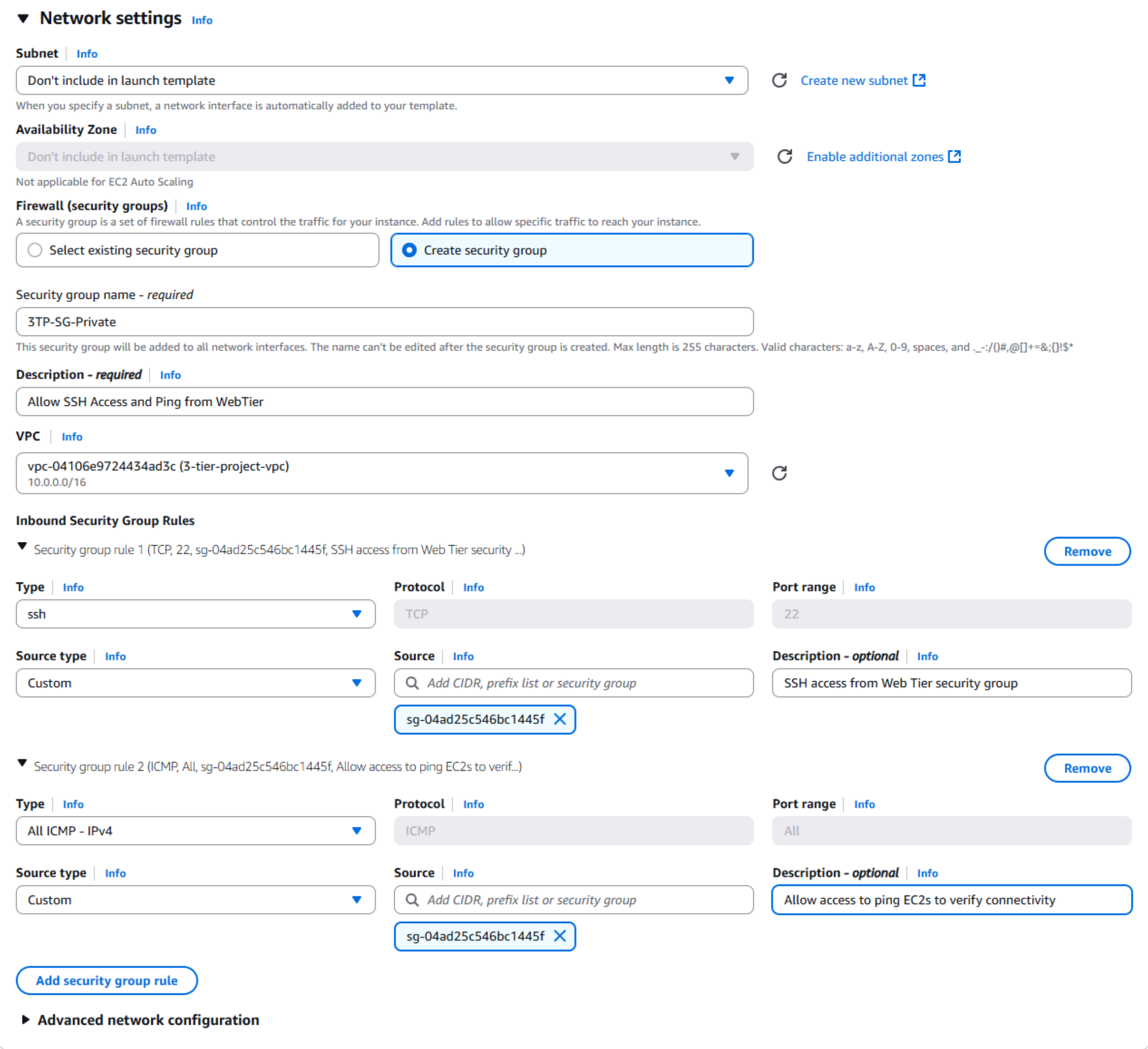
Instance type

t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour On-Demand Linux base pricing: 0.0116 USD per Hour

Figure 14/ App Tier AMI

*For consistency, I selected the same **Amazon Linux 2023 (kernel 6.1) AMI** and **t2.micro instance type** as I used in the web tier. Using the same AMI ensures that both tiers run on a lightweight, AWS-optimized operating system, while the t2.micro instance type provides a cost-effective option suitable for demonstration purposes.*

By keeping the configuration consistent across tiers, it's easier to manage and troubleshoot the environment while still maintaining clear separation of responsibilities between the web and application layers.



▼ Network settings

Subnet: Don't include in launch template ▼

When you specify a subnet, a network interface is automatically added to your template.

Availability Zone: Don't include in launch template ▼

Not applicable for EC2 Auto Scaling

Firewall (security groups)

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.

○ Select existing security group ● Create security group

Security group name - *required*: 3TP-SG-Private

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required*: Allow SSH Access and Ping from WebTier

VPC: vpc-04106e9724434ad3c (3-tier-project-vpc) 10.0.0.0/16 ▼

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, sg-04ad25c546bc1445f, SSH access from Web Tier security ...)

Type: ssh ▼ | Protocol: TCP | Port range: 22

Source type: Custom ▼ | Source: Add CIDR, prefix list or security group — sg-04ad25c546bc1445f ✕ | Description - *optional*: SSH access from Web Tier security group

▼ Security group rule 2 (ICMP, All, sg-04ad25c546bc1445f, Allow access to ping EC2s to verif...)

Type: All ICMP - IPv4 ▼ | Protocol: ICMP | Port range: All

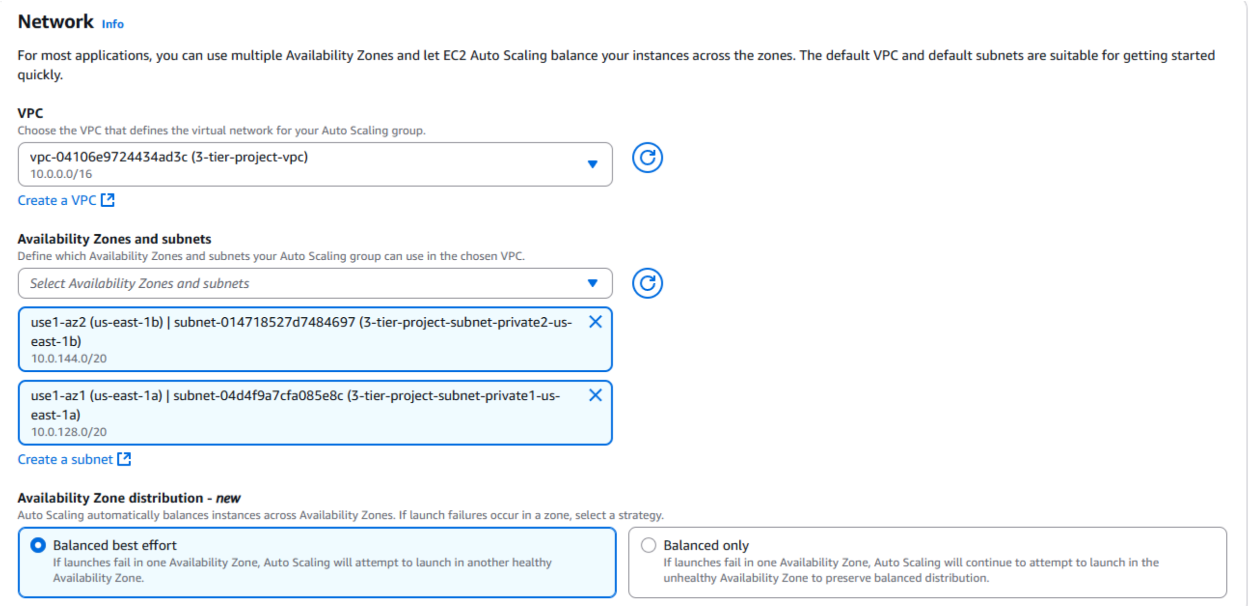
Source type: Custom ▼ | Source: Add CIDR, prefix list or security group — sg-04ad25c546bc1445f ✕ | Description - *optional*: Allow access to ping EC2s to verify connectivity

▶ Advanced network configuration

Figure 15/ Private ASG Network settings

*I configured the **security group** for the Application Tier to allow inbound **SSH access** but restricted it to only come from the **Web Tier security group**. This ensures that only instances in the web tier can securely connect for administrative or troubleshooting purposes. I also allowed **ICMP traffic** from the Web Tier security group to enable basic connectivity testing (such as ping) between tiers.*

*With these rules in place, the Application Tier remains **isolated from the public internet** while still being accessible from the Web Tier when needed. This layered security approach enforces the principle of **least privilege**, ensuring that each tier can only communicate with the specific tiers it needs to, reducing potential attack surfaces.*



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-04106e9724434ad3c (3-tier-project-vpc)
10.0.0.0/16

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

use1-az2 (us-east-1b) | subnet-014718527d7484697 (3-tier-project-subnet-private2-us-east-1b)
10.0.144.0/20

use1-az1 (us-east-1a) | subnet-04d4f9a7cfa085e8c (3-tier-project-subnet-private1-us-east-1a)
10.0.128.0/20

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.

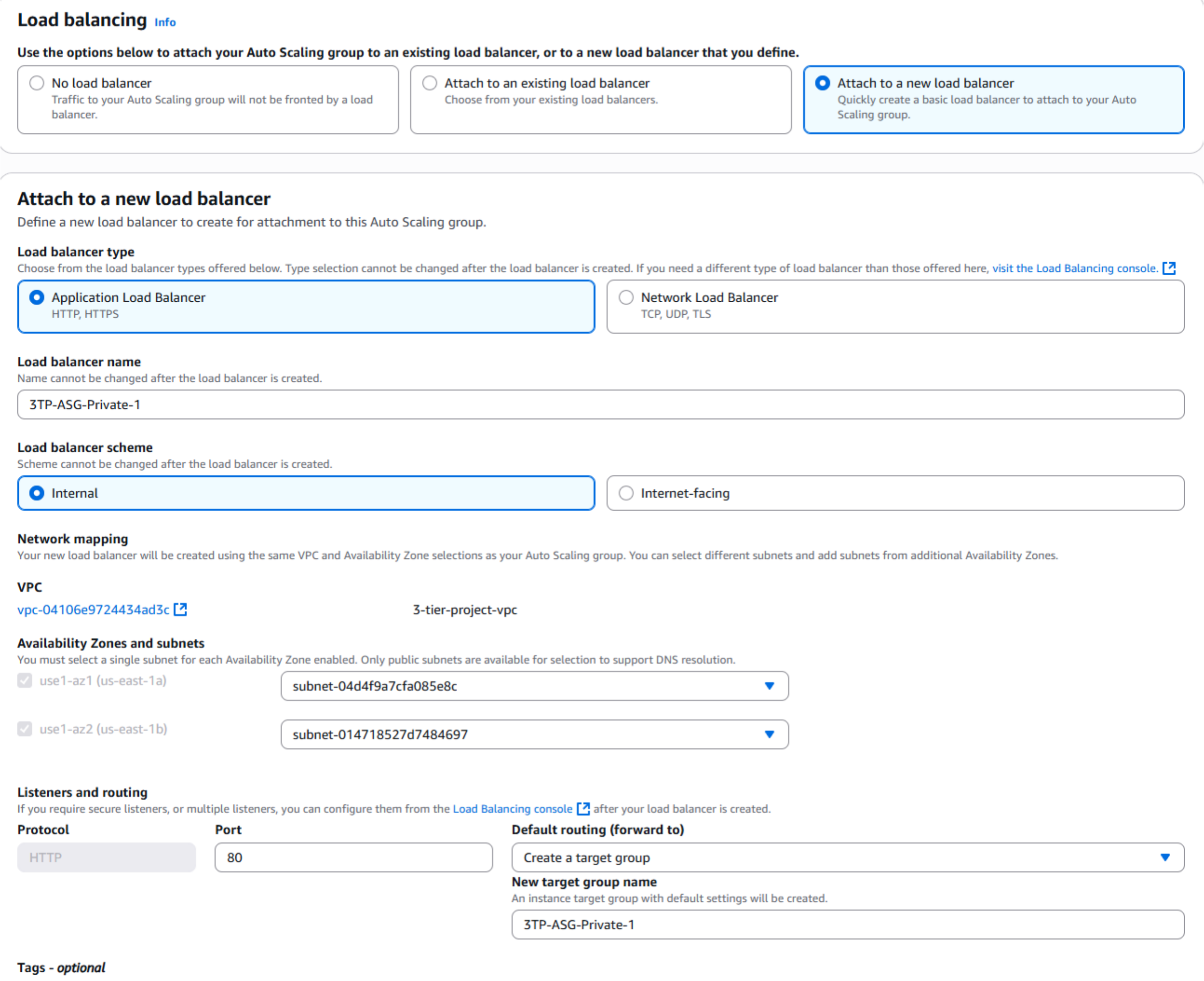
● Balanced best effort
If launches fail in one Availability Zone, Auto Scaling will attempt to launch in another healthy Availability Zone.

○ 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.

Figure 16/ App Tier Network

*After creating the launch template, I navigated back to the **Auto Scaling Group configuration** and selected the new template. I then chose the **VPC created earlier** and mapped the group to **two private subnets across different Availability Zones**.*

*This setup ensures that the Application Tier EC2 instances are launched securely within private subnets, isolated from the internet, and distributed across multiple AZs for **high availability and fault tolerance**.*



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 a new load balancer

Define a new load balancer to create for attachment to this Auto Scaling group.

Load balancer type
Choose from the load balancer types offered below. Type selection cannot be changed after the load balancer is created. If you need a different type of load balancer than those offered here, visit the Load Balancing console.

● Application Load Balancer
HTTP, HTTPS

○ Network Load Balancer
TCP, UDP, TLS

Load balancer name
Name cannot be changed after the load balancer is created.
3TP-ASG-Private-1

Load balancer scheme
Scheme cannot be changed after the load balancer is created.

● Internal ○ Internet-facing

Network mapping
Your new load balancer will be created using the same VPC and Availability Zone selections as your Auto Scaling group. You can select different subnets and add subnets from additional Availability Zones.

VPC
vpc-04106e9724434ad3c 3-tier-project-vpc

Availability Zones and subnets
You must select a single subnet for each Availability Zone enabled. Only public subnets are available for selection to support DNS resolution.

☑ use1-az1 (us-east-1a) subnet-04d4f9a7cfa085e8c

☑ use1-az2 (us-east-1b) subnet-014718527d7484697

Listeners and routing
If you require secure listeners, or multiple listeners, you can configure them from the Load Balancing console after your load balancer is created.

Protocol	Port	Default routing (forward to)
HTTP	80	Create a target group

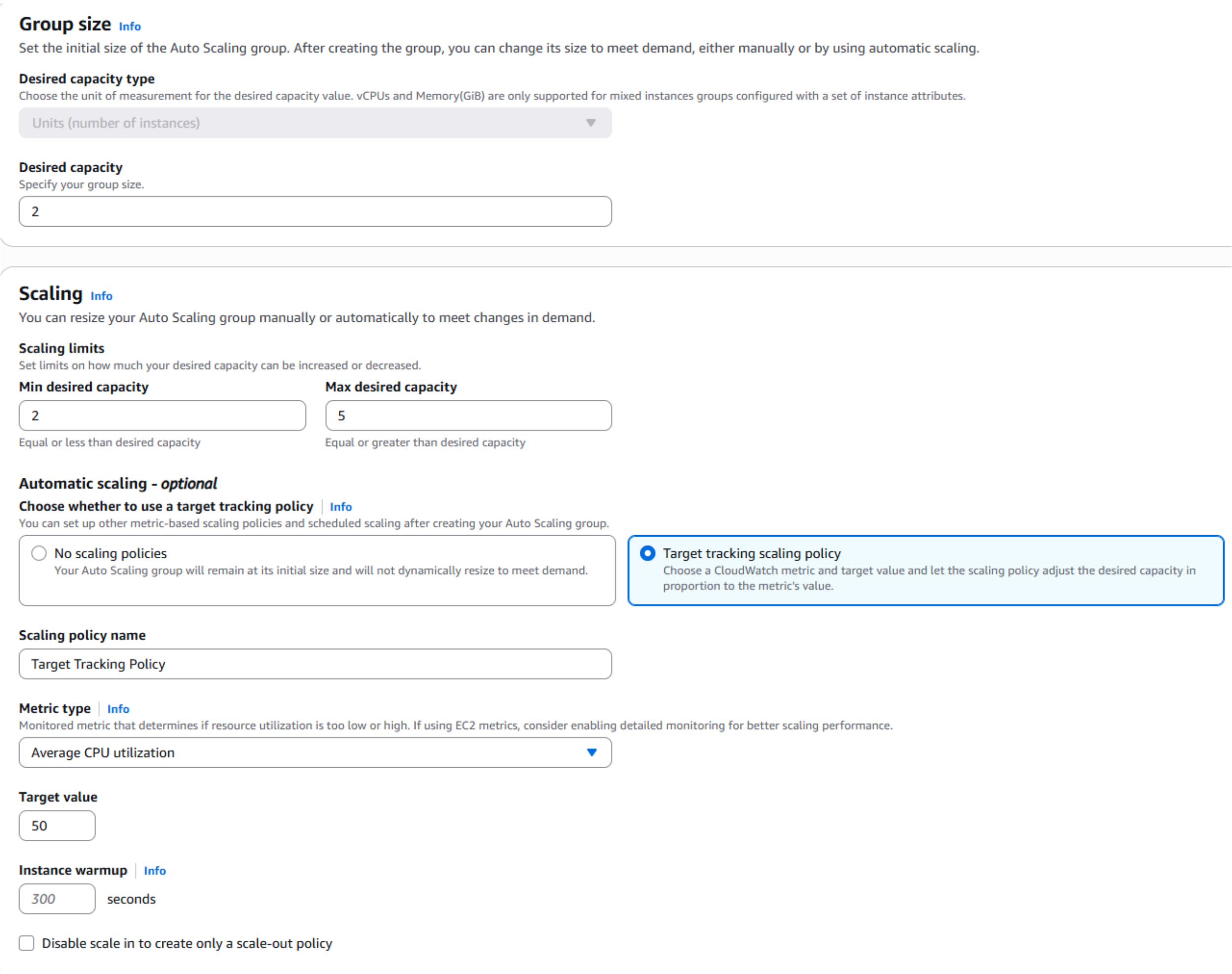
New target group name
An instance target group with default settings will be created.
3TP-ASG-Private-1

Tags - *optional*

Figure 17/ App Tier Load Balancer

*To handle incoming requests from the Web Tier, I created an **Application Load Balancer (ALB)** and configured it to distribute traffic to the **Auto Scaling Group** running EC2 instances in the Application Tier.*

*By placing the ALB between the Web Tier and the Application Tier, traffic is routed efficiently across multiple instances, improving **scalability, fault tolerance, and high availability**. This also ensures that the Application Tier remains private and only accessible through controlled traffic originating from the Web Tier.*



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.

2

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

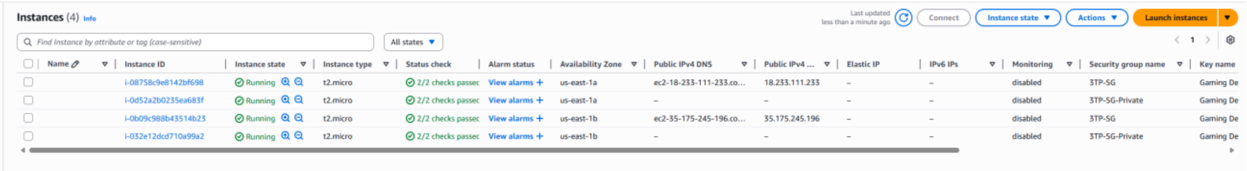
Instance warmup | Info

300 seconds

□ Disable scale in to create only a scale-out policy

Figure 18/ App Tier group size and scaling policy

*I configured the **group size** and **scaling policy** for the Application Tier's Auto Scaling Group the same way I did for the Web Tier. By applying this consistent scaling approach, the Application Tier inherits the same benefits of **elasticity, high availability, and resilience**, ensuring it can seamlessly adjust to variable traffic demands while still maintaining secure isolation within the private subnets.*



Instances (4) Info

Last updated less than a minute ago

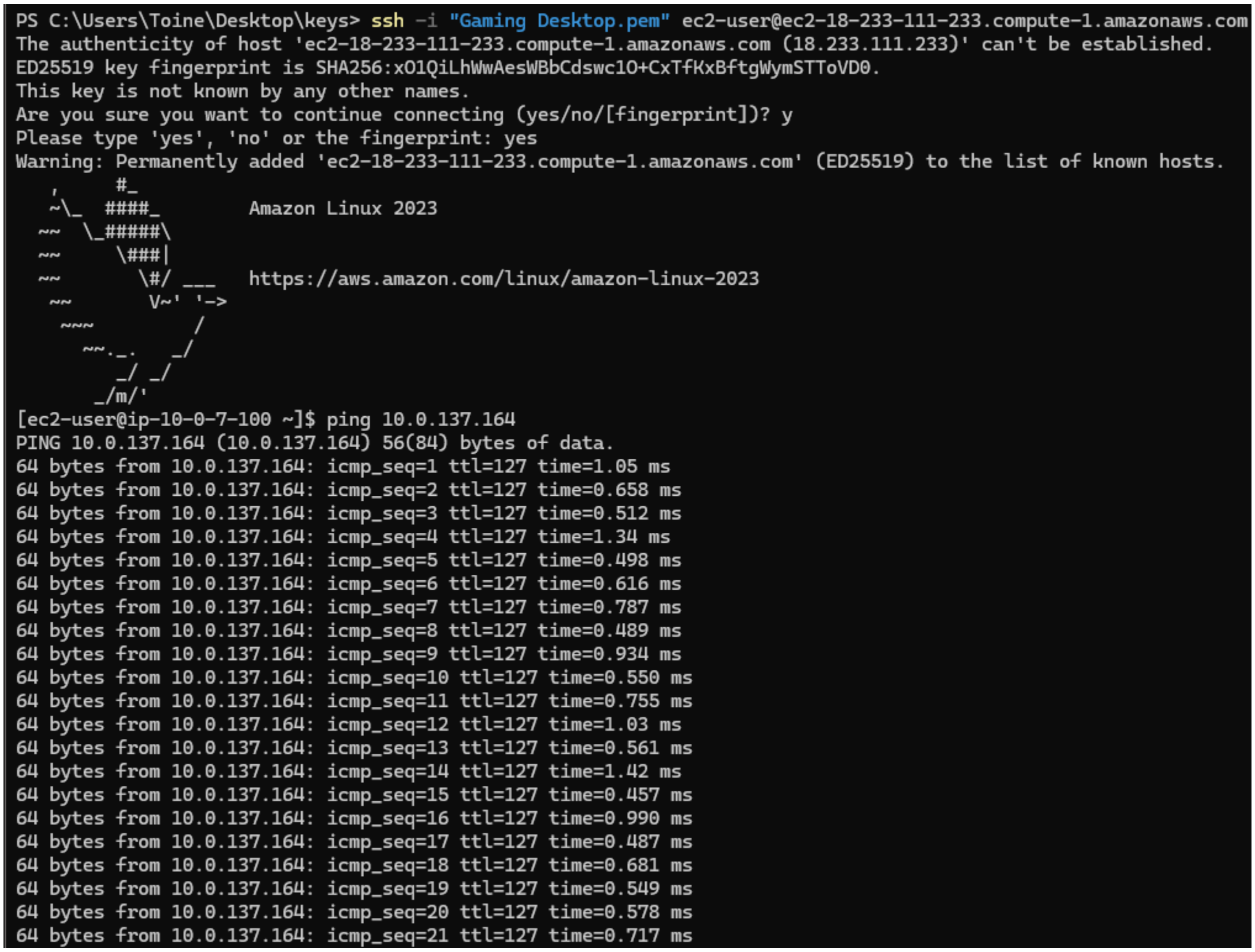
□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security group name	Key name
□		i-08758c9e8142bf698	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-18-233-111-233.co...	18.233.111.233	–	–	disabled	3TP-SG	Gaming De
□		i-0d52a2b0235ea683f	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	–	–	–	–	disabled	3TP-SG-Private	Gaming De
□		i-0b09c988b43514b23	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-35-175-245-196.co...	35.175.245.196	–	–	disabled	3TP-SG	Gaming De
□		i-032e12dcd710a99a2	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	–	–	–	–	disabled	3TP-SG-Private	Gaming De

Figure 19/ App Tier EC2s

*After configuring both Auto Scaling Groups, I verified that the instances were successfully launched. The screenshot shows **four running EC2 instances** distributed across two Availability Zones:*

- ***Web Tier:** Two instances running in the public subnets, each with a public IPv4 address for internet access.*
- ***Application Tier:** Two instances running in the private subnets, without public IPs, ensuring they remain isolated and only reachable from the Web Tier.*

*This confirms that both tiers are correctly deployed, **redundant across AZs**, and aligned with the 3-tier design pattern.*



```
PS C:\Users\Toine\Desktop\keys> ssh -i "Gaming Desktop.pem" ec2-user@ec2-18-233-111-233.compute-1.amazonaws.com
The authenticity of host 'ec2-18-233-111-233.compute-1.amazonaws.com (18.233.111.233)' can't be established.
ED25519 key fingerprint is SHA256:xO1QiLhWwAesWBbCdswc1O+CxTfKxBftgWymSTToVD0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'ec2-18-233-111-233.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-7-100 ~]$ ping 10.0.137.164
PING 10.0.137.164 (10.0.137.164) 56(84) bytes of data.
64 bytes from 10.0.137.164: icmp_seq=1 ttl=127 time=1.05 ms
64 bytes from 10.0.137.164: icmp_seq=2 ttl=127 time=0.658 ms
64 bytes from 10.0.137.164: icmp_seq=3 ttl=127 time=0.512 ms
64 bytes from 10.0.137.164: icmp_seq=4 ttl=127 time=1.34 ms
64 bytes from 10.0.137.164: icmp_seq=5 ttl=127 time=0.498 ms
64 bytes from 10.0.137.164: icmp_seq=6 ttl=127 time=0.616 ms
64 bytes from 10.0.137.164: icmp_seq=7 ttl=127 time=0.787 ms
64 bytes from 10.0.137.164: icmp_seq=8 ttl=127 time=0.489 ms
64 bytes from 10.0.137.164: icmp_seq=9 ttl=127 time=0.934 ms
64 bytes from 10.0.137.164: icmp_seq=10 ttl=127 time=0.550 ms
64 bytes from 10.0.137.164: icmp_seq=11 ttl=127 time=0.755 ms
64 bytes from 10.0.137.164: icmp_seq=12 ttl=127 time=1.03 ms
64 bytes from 10.0.137.164: icmp_seq=13 ttl=127 time=0.561 ms
64 bytes from 10.0.137.164: icmp_seq=14 ttl=127 time=1.42 ms
64 bytes from 10.0.137.164: icmp_seq=15 ttl=127 time=0.457 ms
64 bytes from 10.0.137.164: icmp_seq=16 ttl=127 time=0.990 ms
64 bytes from 10.0.137.164: icmp_seq=17 ttl=127 time=0.487 ms
64 bytes from 10.0.137.164: icmp_seq=18 ttl=127 time=0.681 ms
64 bytes from 10.0.137.164: icmp_seq=19 ttl=127 time=0.549 ms
64 bytes from 10.0.137.164: icmp_seq=20 ttl=127 time=0.578 ms
64 bytes from 10.0.137.164: icmp_seq=21 ttl=127 time=0.717 ms
```

Figure 20/ EC2 ssh ping verification

*After successfully connecting to one of my **Web Tier EC2 instances** using SSH from PowerShell, I validated communication with the **Application Tier**. From the web server, I pinged the **private IP address** of one of the Application Tier instances (10.0.137.164).*

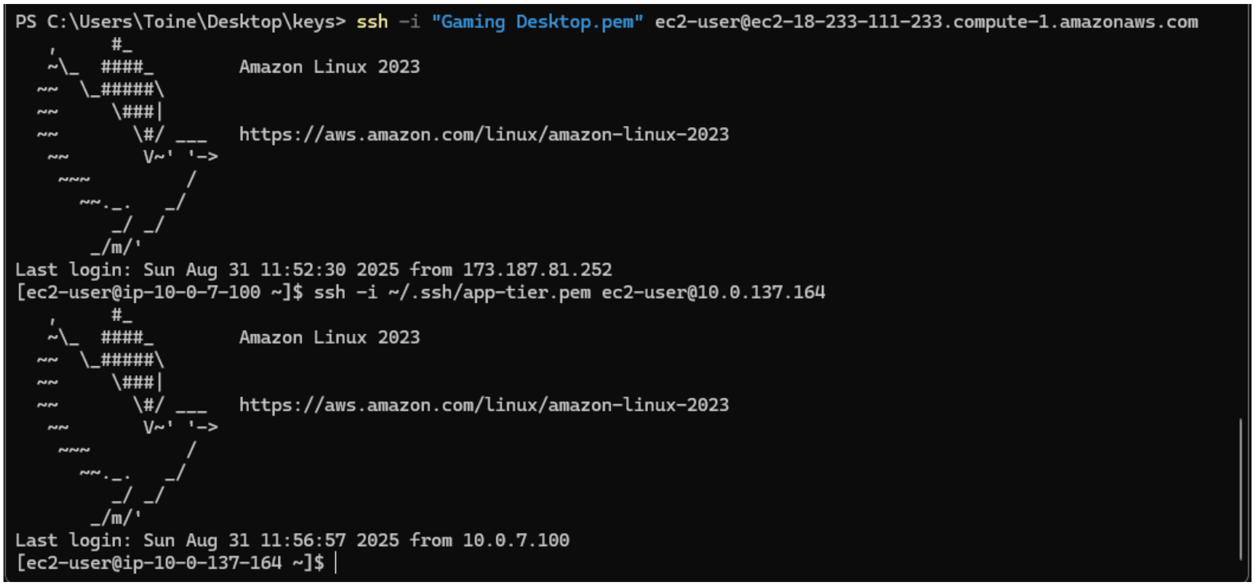
The ping returned successful responses, confirming that:

- *The **security group rules** allowing ICMP traffic from the Web Tier to the Application Tier were correctly configured.*

- *The Application Tier instances are reachable from the Web Tier, despite being isolated in **private subnets** with no public IPs.*

- *The **tiered networking design** is functioning as intended, enforcing isolation from the internet while maintaining secure communication between tiers.*

*This test verifies that the **Web Tier → Application Tier path is working**, laying the foundation for seamless request flow through the 3-tier architecture.*



```
PS C:\Users\Toine\Desktop\keys> ssh -i "Gaming Desktop.pem" ec2-user@ec2-18-233-111-233.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sun Aug 31 11:52:30 2025 from 173.187.81.252
[ec2-user@ip-10-0-7-100 ~]$ ssh -i ~/.ssh/app-tier.pem ec2-user@10.0.137.164
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sun Aug 31 11:56:57 2025 from 10.0.7.100
[ec2-user@ip-10-0-137-164 ~]$ |
```

Figure 21/ Validation of Web Tier to App Tier Access

*From my local machine, I first connected to a **Web Tier instance** using SSH. From there, I securely connected to an **Application Tier instance** in a private subnet using the copied private key. The successful login banner confirms that the **Web Tier → Application Tier connectivity** is working as intended, with traffic restricted to flow only between tiers inside the VPC.*

Phase 4: Database Tier

For the **Database Tier**, I used **Amazon DynamoDB**. DynamoDB is a fully managed NoSQL key-value and document database that offers **serverless scaling**, **single-digit millisecond latency**, **point-in-time recovery (PITR) backups**, and **encryption by default**. It's an excellent choice for this project because it highlights **scalability and simplicity**, while removing the need to provision or manage database servers.

I like to think of DynamoDB as **a highly scalable, serverless filing cabinet in the cloud**. I just store and retrieve items by key, and AWS takes care of all the scaling, durability, and security behind the scenes.

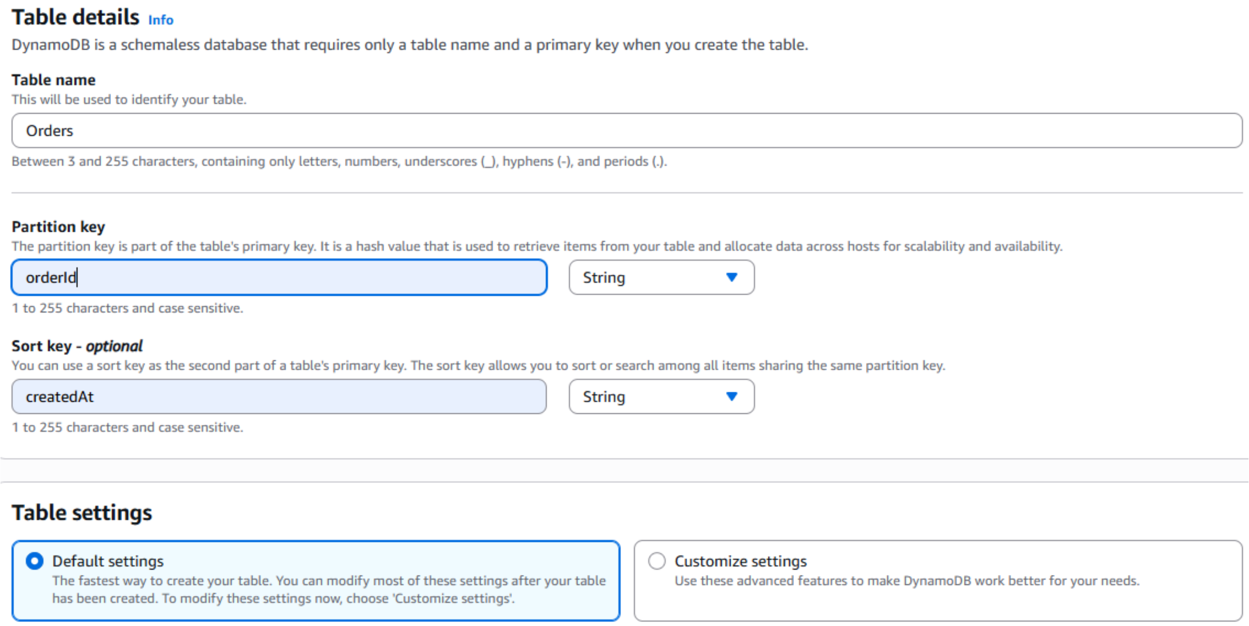


Table details Info

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Orders

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

orderId | String ▼

1 to 255 characters and case sensitive.

Sort key - *optional*

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

createdAt | String ▼

1 to 255 characters and case sensitive.

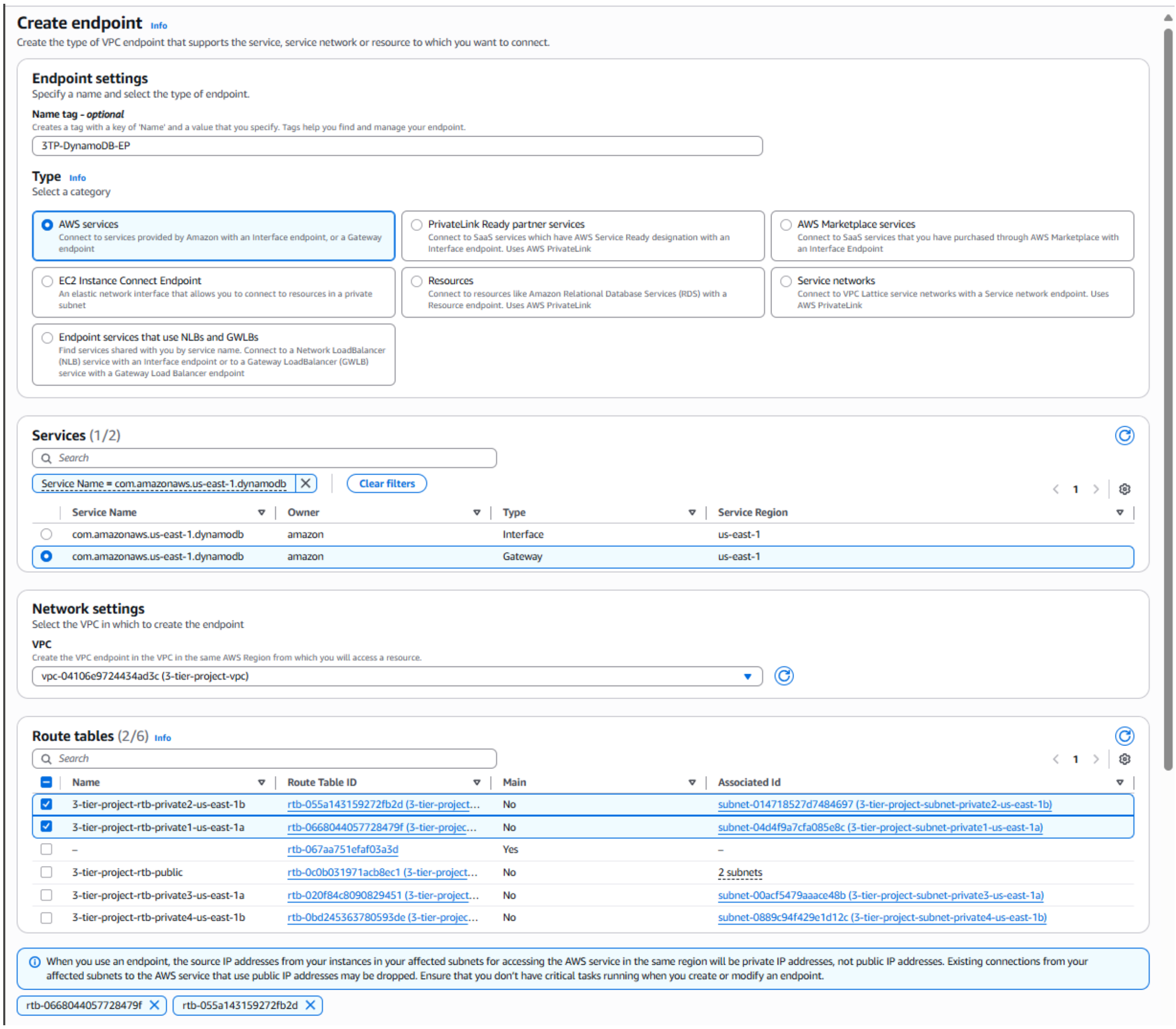
Table settings

● Default settings
The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.

○ Customize settings
Use these advanced features to make DynamoDB work better for your needs.

Figure 22/ Table design

*I created a DynamoDB table named **Orders**. The table uses **orderID** (String) as the **partition key**, ensuring each order record is uniquely identifiable. I also added a **sort key** called **createdAt**, which allows me to perform **time-ordered queries**. This design makes it easy to retrieve all orders for a given orderID and sort them by the time they were created, which is a common pattern for order-tracking systems.*



Create endpoint Info

Create the type of VPC endpoint that supports the service, service network or resource to which you want to connect.

Endpoint settings

Specify a name and select the type of endpoint.

Name tag - *optional*

Creates a tag with a key of 'Name' and a value that you specify. Tags help you find and manage your endpoint.

3TP-DynamoDB-EP

Type Info

Select a category

● AWS services
Connect to services provided by Amazon with an Interface endpoint, or a Gateway endpoint

○ PrivateLink Ready partner services
Connect to SaaS services which have AWS Service Ready designation with an Interface endpoint. Uses AWS PrivateLink

○ AWS Marketplace services
Connect to SaaS services that you have purchased through AWS Marketplace with an Interface Endpoint

○ EC2 Instance Connect Endpoint
An elastic network interface that allows you to connect to resources in a private subnet

○ Resources
Connect to resources like Amazon Relational Database Services (RDS) with a Resource endpoint. Uses AWS PrivateLink

○ Service networks
Connect to VPC Lattice service networks with a Service network endpoint. Uses AWS PrivateLink

○ Endpoint services that use NLBs and GWLBs
Find services shared with you by service name. Connect to a Network LoadBalancer (NLB) service with an Interface endpoint or to a Gateway LoadBalancer (GWLB) service with a Gateway Load Balancer endpoint

Services (1/2)

Search

Service Name = com.amazonaws.us-east-1.dynamodb ✕ Clear filters

	Service Name	Owner	Type	Service Region
○	com.amazonaws.us-east-1.dynamodb	amazon	Interface	us-east-1
●	com.amazonaws.us-east-1.dynamodb	amazon	Gateway	us-east-1

Network settings

Select the VPC in which to create the endpoint

VPC

Create the VPC endpoint in the VPC in the same AWS Region from which you will access a resource.

vpc-04106e9724434ad3c (3-tier-project-vpc)

Route tables (2/6) Info

Search

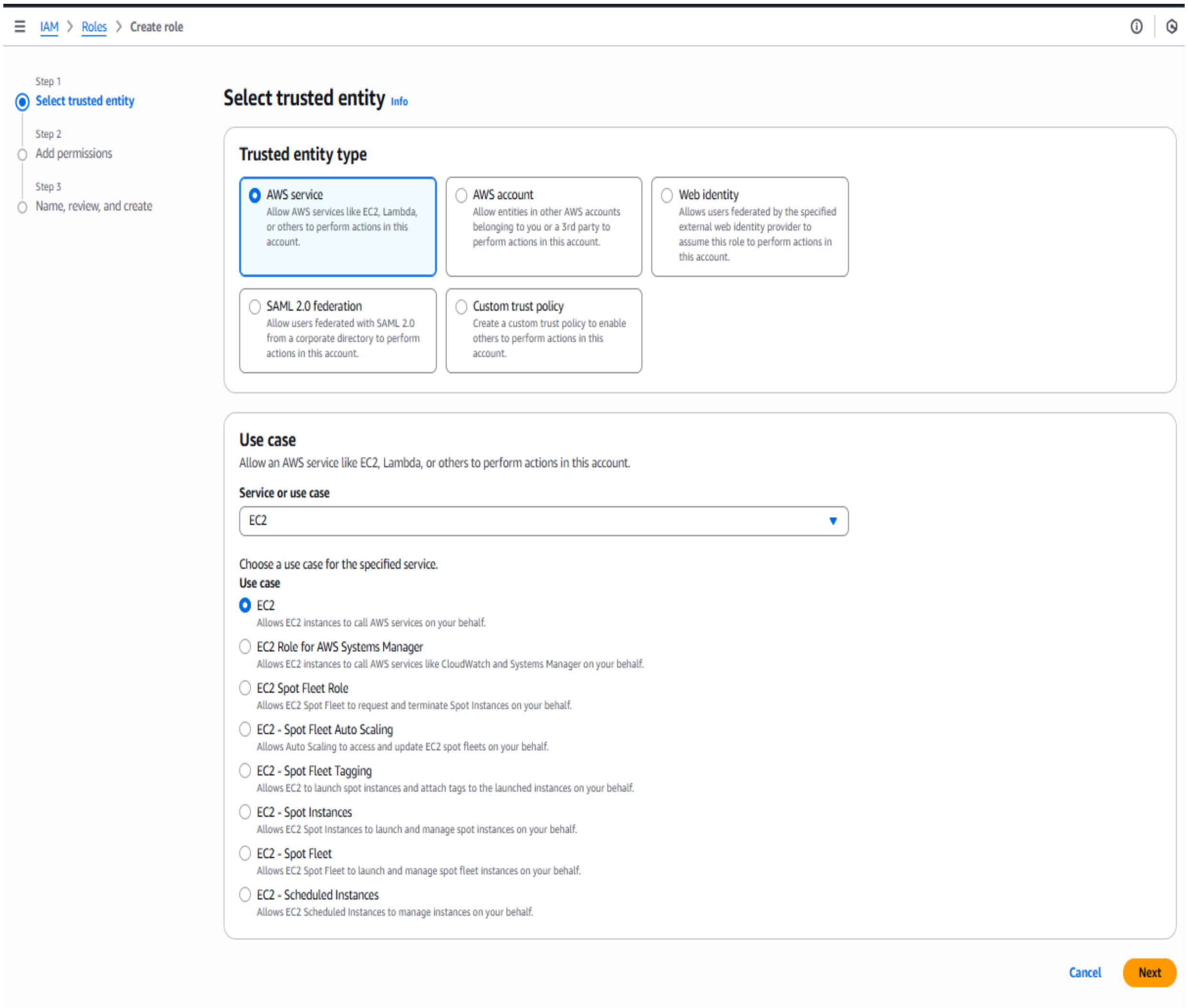
	Name	Route Table ID	Main	Associated Id
☑	3-tier-project-rtb-private2-us-east-1b	rtb-055a143159272fb2d (3-tier-project…	No	subnet-014718527d7484697 (3-tier-project-subnet-private2-us-east-1b)
☑	3-tier-project-rtb-private1-us-east-1a	rtb-0668044057728479f (3-tier-projec…	No	subnet-04d4f9a7cfa085e8c (3-tier-project-subnet-private1-us-east-1a)
□	–	rtb-067aa751efaf03a3d	Yes	–
□	3-tier-project-rtb-public	rtb-0c0b031971acb8ec1 (3-tier-project…	No	2 subnets
□	3-tier-project-rtb-private3-us-east-1a	rtb-020f84c8090829451 (3-tier-project…	No	subnet-00acf5479aaace48b (3-tier-project-subnet-private3-us-east-1a)
□	3-tier-project-rtb-private4-us-east-1b	rtb-0bd245363780593de (3-tier-projec…	No	subnet-0889c94f429e1d12c (3-tier-project-subnet-private4-us-east-1b)

ⓘ When you use an endpoint, the source IP addresses from your instances in your affected subnets for accessing the AWS service in the same region will be private IP addresses, not public IP addresses. Existing connections from your affected subnets to the AWS service that use public IP addresses may be dropped. Ensure that you don't have critical tasks running when you create or modify an endpoint.

rtb-0668044057728479f ✕ rtb-055a143159272fb2d ✕

Figure 23/ DynamoDB VPC Endpoint

*To keep all database traffic private, I created a **VPC Gateway Endpoint** for DynamoDB. This ensures that requests from my Application Tier instances to DynamoDB stay within the AWS network and never traverse the public internet. By using this endpoint, my application servers in the private subnets can securely call DynamoDB **without requiring a NAT Gateway or Internet Gateway**, which improves both **security** and **cost efficiency**.*



IAM > Roles > Create role

Step 1
● Select trusted entity

Step 2
○ Add permissions

Step 3
○ Name, review, and create

Select trusted entity Info

Trusted entity type

● AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

○ AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

○ Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

○ SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

○ Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case: EC2

Choose a use case for the specified service.

Use case

● EC2
Allows EC2 instances to call AWS services on your behalf.

○ EC2 Role for AWS Systems Manager
Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.

○ EC2 Spot Fleet Role
Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.

○ EC2 - Spot Fleet Auto Scaling
Allows Auto Scaling to access and update EC2 spot fleets on your behalf.

○ EC2 - Spot Fleet Tagging
Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.

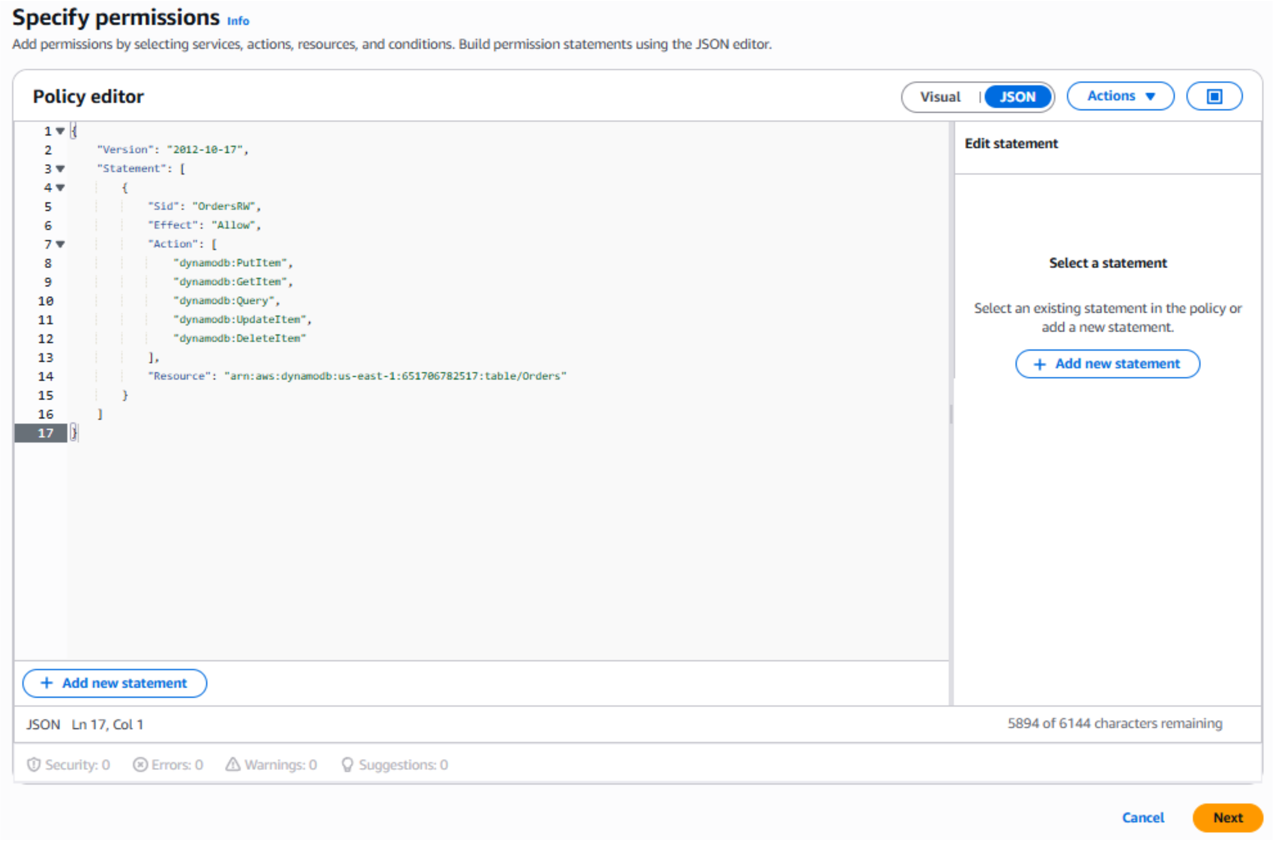
○ EC2 - Spot Instances
Allows EC2 Spot Instances to launch and manage spot instances on your behalf.

○ EC2 - Spot Fleet
Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.

○ EC2 - Scheduled Instances
Allows EC2 Scheduled Instances to manage instances on your behalf.

Figure 24/ IAM role for App Tier

To allow my Application Tier EC2 instances to access DynamoDB securely, I created a new ***IAM role*** *and selected* ***EC2*** *as the trusted service. This ensures that the role can be attached directly to my Application Tier instances (via their Auto Scaling Group), giving them permissions to interact with DynamoDB* ***without the need to store or manage access keys on the servers****.*



Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "OrdersRW",
            "Effect": "Allow",
            "Action": [
                "dynamodb:PutItem",
                "dynamodb:GetItem",
                "dynamodb:Query",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": "arn:aws:dynamodb:us-east-1:651706782517:table/Orders"
        }
    ]
}
```

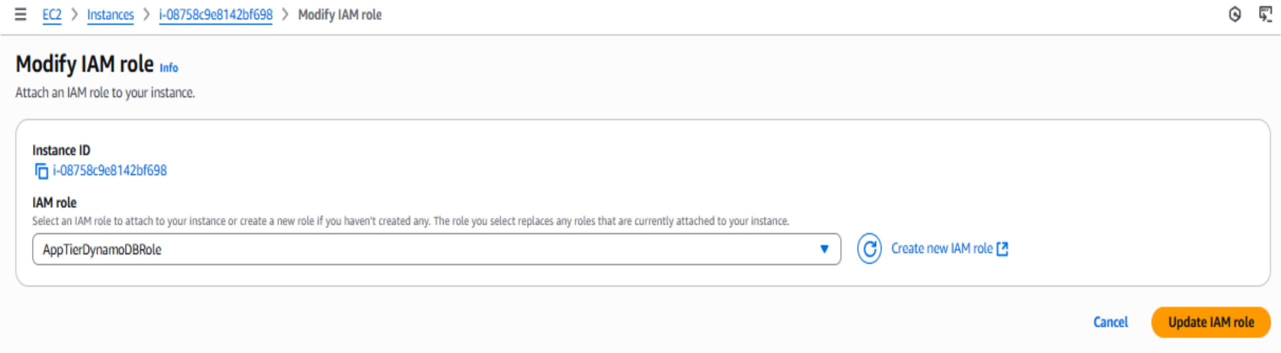
Select an existing statement in the policy or add a new statement.

JSON Ln 17, Col 1 — 5894 of 6144 characters remaining

Security: 0 Errors: 0 Warnings: 0 Suggestions: 0

Figure 25/ DynamoDBRole Policy

*This is the **least privilege policy** I attached to the DynamoDBRole. It grants the Application Tier EC2 instances only the necessary permissions to read and write data in the Orders table, ensuring secure access without exposing unnecessary actions. Following the **principle of least privilege** reduces the blast radius in case of a misconfiguration or compromise, strengthening the overall security of the architecture.*



Attach an IAM role to your instance.

Instance ID: i-08758c9e8142bf698

IAM role: AppTierDynamoDBRole

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

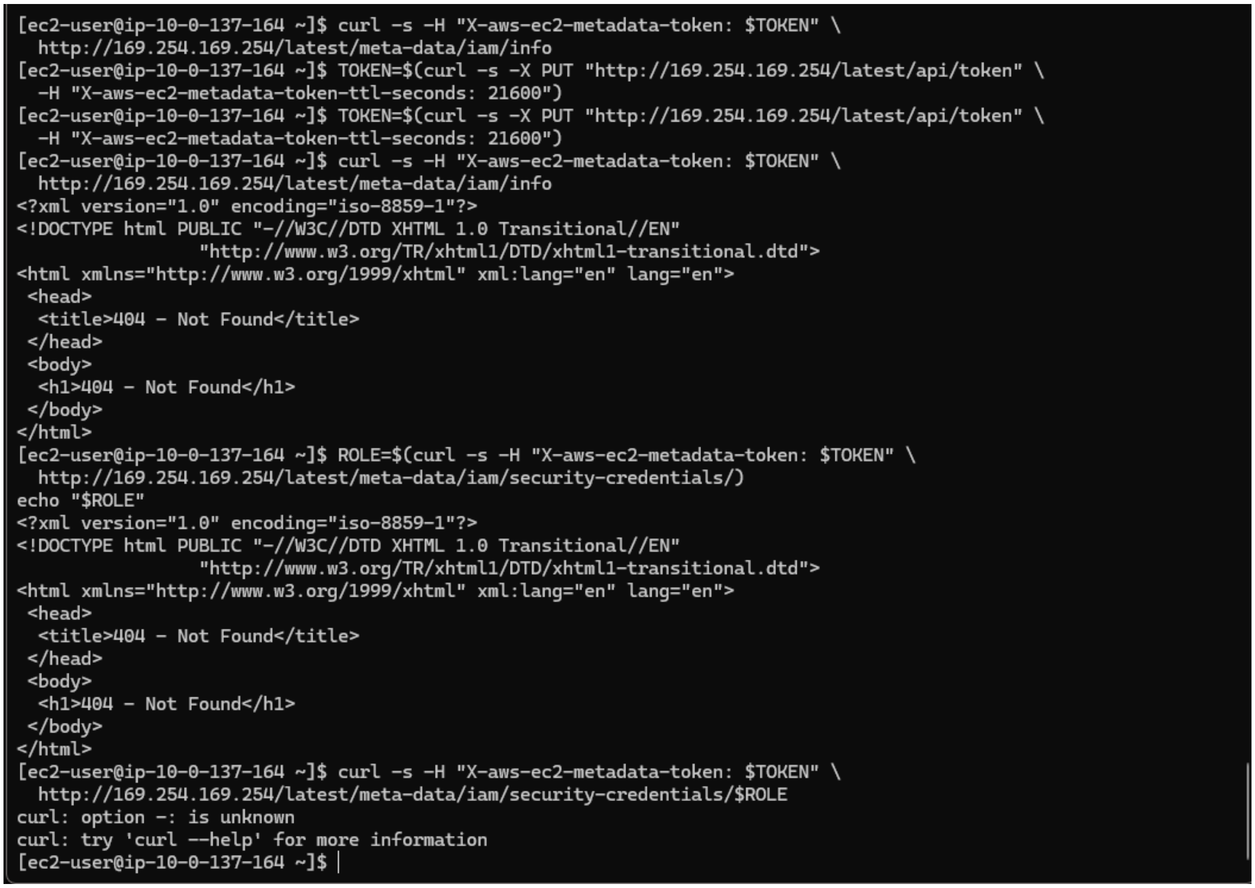
Figure 26/ Attach IAM role to EC2 instances

*After creating the AppTierDynamoDBRole, I attached it to my Application Tier EC2 instances. By assigning the IAM role directly to the instances, they can now securely access DynamoDB **without the need for hardcoded credentials or access keys**. This setup leverages the AWS Instance Metadata Service (IMDS), which automatically provides temporary credentials to the EC2 instances under the assigned role.*

Testing and Validation

With all three tiers of the architecture deployed, I performed a series of tests to validate connectivity, security, and functionality across the stack. Starting with **Web → Application Tier**, I confirmed private connectivity through security group rules. Next, from the **Application Tier**, I verified that the attached IAM role and VPC endpoint enabled secure, credential-free access to DynamoDB. Finally, I tested read and write operations both through the AWS CLI and with a simple Python application running directly on the App Tier instance.

These validation steps confirm that each layer of the 3-tier architecture is properly isolated, connected only where needed, and functioning as intended.

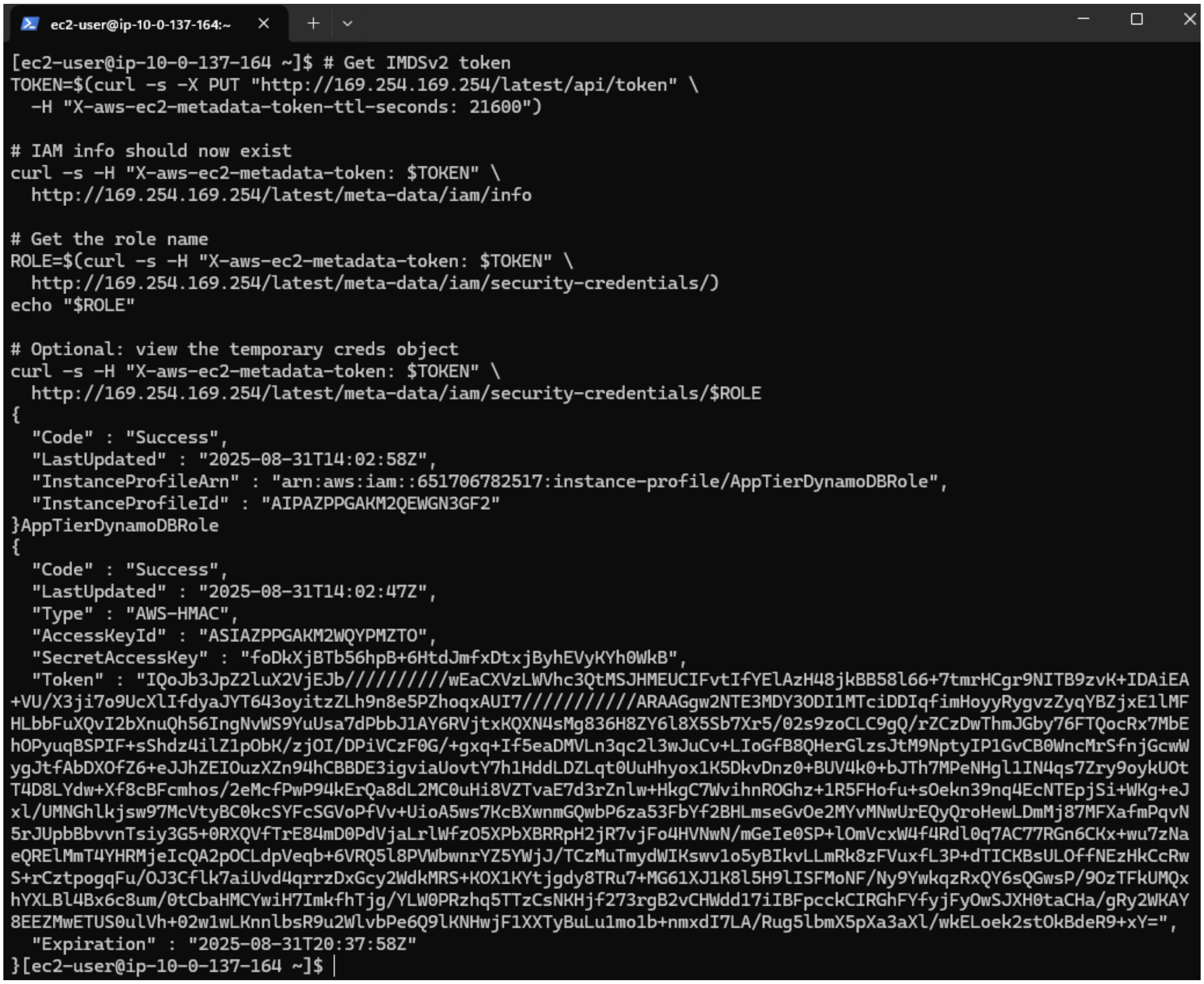


```
[ec2-user@ip-10-0-137-164 ~]$ curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/iam/info
[ec2-user@ip-10-0-137-164 ~]$ TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
[ec2-user@ip-10-0-137-164 ~]$ TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
[ec2-user@ip-10-0-137-164 ~]$ curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/iam/info
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <title>404 - Not Found</title>
 </head>
 <body>
  <h1>404 - Not Found</h1>
 </body>
</html>
[ec2-user@ip-10-0-137-164 ~]$ ROLE=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/iam/security-credentials/)
echo "$ROLE"
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <title>404 - Not Found</title>
 </head>
 <body>
  <h1>404 - Not Found</h1>
 </body>
</html>
[ec2-user@ip-10-0-137-164 ~]$ curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/iam/security-credentials/$ROLE
curl: option -: is unknown
curl: try 'curl --help' for more information
[ec2-user@ip-10-0-137-164 ~]$ |
```

Figure 27/ Troubleshooting IAM Role Attachment

*Here I attempted to verify the IAM role from an Application Tier instance using the EC2 Instance Metadata Service (IMDS). However, I kept receiving 404 – Not Found responses. After reviewing my setup, I realized I had mistakenly attached the **AppTierDynamoDBRole** to my **Web Tier instances** instead of the Application Tier Auto Scaling Group.*

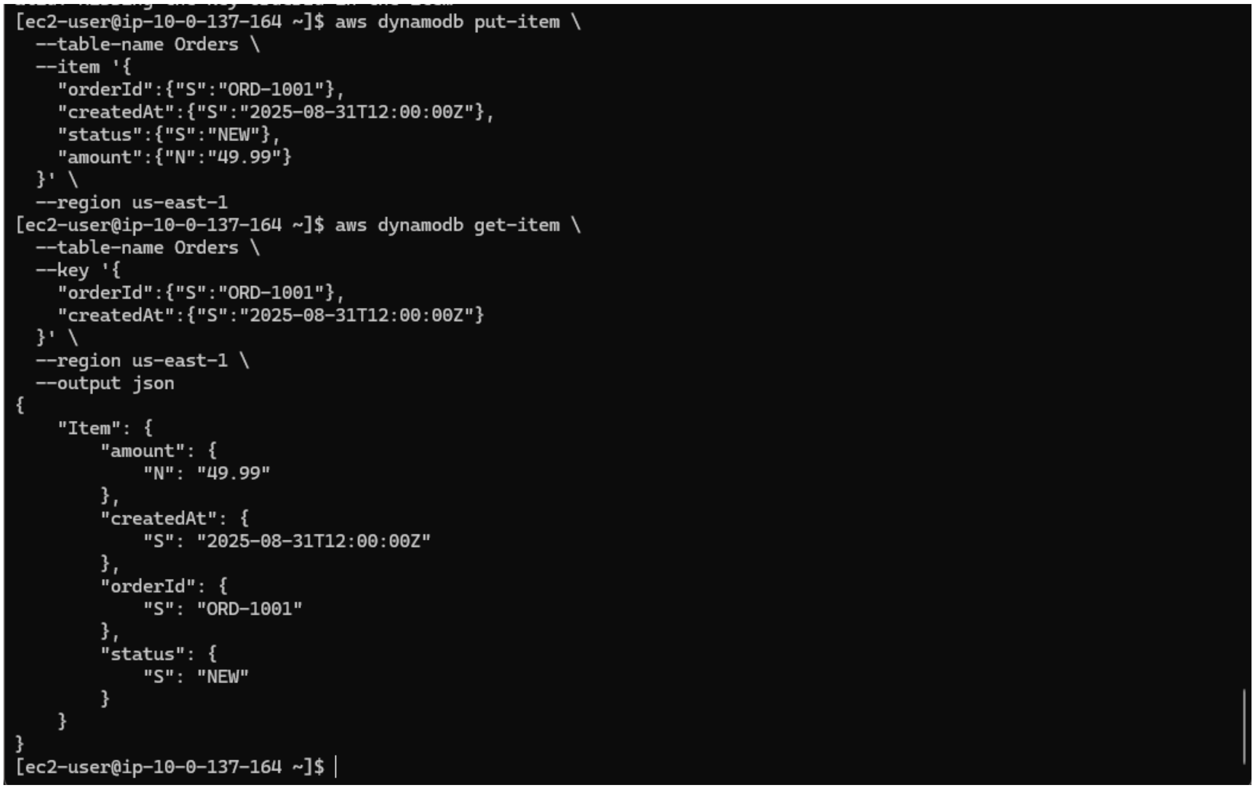
This was a useful debugging moment because it demonstrated how ***IMDS returns a 404 when no role is attached*** *to an instance, helping me confirm the source of the issue. Once corrected, the Application Tier instances were able to assume the role and securely access DynamoDB without requiring credentials.*



```
[ec2-user@ip-10-0-137-164 ~]$ # Get IMDSv2 token
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")

# IAM info should now exist
curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/iam/info

# Get the role name
ROLE=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/iam/security-credentials/)
echo "$ROLE"

# Optional: view the temporary creds object
curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/iam/security-credentials/$ROLE
{
  "Code" : "Success",
  "LastUpdated" : "2025-08-31T14:02:58Z",
  "InstanceProfileArn" : "arn:aws:iam::651706782517:instance-profile/AppTierDynamoDBRole",
  "InstanceProfileId" : "AIPAZPPGAKM2QEWGN3GF2"
}AppTierDynamoDBRole
{
  "Code" : "Success",
  "LastUpdated" : "2025-08-31T14:02:47Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAZPPGAKM2WQYPMZTO",
  "SecretAccessKey" : "foDkXjBTb56hpB+6HtdJmfxDtxjByhEVyKYh0WkB",
  "Token" : "IQoJb3JpZ2luX2VjEJb//////////wEaCXVzLWVhc3QtMSJHMEUCIFvtIfYElAzH48jkBB58l66+7tmrHCgr9NITB9zvK+IDAiEA
+VU/X3ji7o9UcXlIfdyaJYT643oyitzZLh9n8e5PZhoqxAUI7///////////ARAAGgw2NTE3MDY3ODI1MTciDDIqfimHoyyRygvzZyqYBZjxE1lMF
HLbbFuXQvI2bXnuQh56IngNvWS9YuUsa7dPbbJ1AY6RVjtxKQXN4sMg836H8ZY6l8X5Sb7Xr5/02s9zoCLC9gQ/rZCzDwThmJGby76FTQocRx7MbE
hOPyuqBSPIF+sShdz4ilZ1pObK/zjOI/DPiVCzF0G/+gxq+If5eaDMVLn3qc2l3wJuCv+LIoGfB8QHerGlzsJtM9NptyIP1GvCB0WncMrSfnjGcwW
ygJtfAbDXOfZ6+eJJhZEIOuzXZn94hCBBDE3igviaUovtY7h1HddLDZLqt0UuHhyox1K5DkvDnz0+BUV4k0+bJTh7MPeNHgl1IN4qs7Zry9oykUOt
T4D8LYdw+Xf8cBFcmhos/2eMcfPwP94kErQa8dL2MC0uHi8VZTvaE7d3rZnlw+HkgC7WvihnROGhz+1R5FHofu+sOekn39nq4EcNTEpjSi+WKg+eJ
xl/UMNGhlkjsw97McVtyBC0kcSYFcSGVoPfVv+UioA5ws7KcBXwnmGQwbP6za53FbYf2BHLmseGvOe2MYvMNwUrEQyQroHewLDmMj87MFXafmPqvN
5rJUpbBbvvnTsiy3G5+0RXQVfTrE84mD0PdVjaLrlWfzO5XPbXBRRpH2jR7vjFo4HVNwN/mGeIe0SP+lOmVcxW4f4Rdl0q7AC77RGn6CKx+wu7zNa
eQRElMmT4YHRMjeIcQA2pOCLdpVeqb+6VRQ5l8PVWbwnrYZ5YWjJ/TCzMuTmydWIKswv1o5yBIkvLLmRk8zFVuxfL3P+dTICKBsULOffNEzHkCcRw
S+rCztpogqFu/OJ3Cflk7aiUvd4qrrzDxGcy2WdkMRS+KOX1KYtjgdy8TRu7+MG61XJ1K8l5H9lISFMoNF/Ny9YwkqzRxQY6sQGwsP/9OzTFkUMQx
hYXLBl4Bx6c8um/0tCbaHMCYwiH7ImkfhTjg/YLW0PRzhq5TTzCsNKHjf273rgB2vCHWdd17iIBFpcckCIRGhFYfyjFyOwSJXH0taCHa/gRy2WKAY
8EEZMwETUS0ulVh+02w1wLKnnlbsR9u2WlvbPe6Q9lKNHwjF1XXTyBuLu1mo1b+nmxdI7LA/Rug5lbmX5pXa3aXl/wkELoek2stOkBdeR9+xY=",
  "Expiration" : "2025-08-31T20:37:58Z"
}[ec2-user@ip-10-0-137-164 ~]$ |
```

Figure 28/ Verify IAM Role 'fixed'

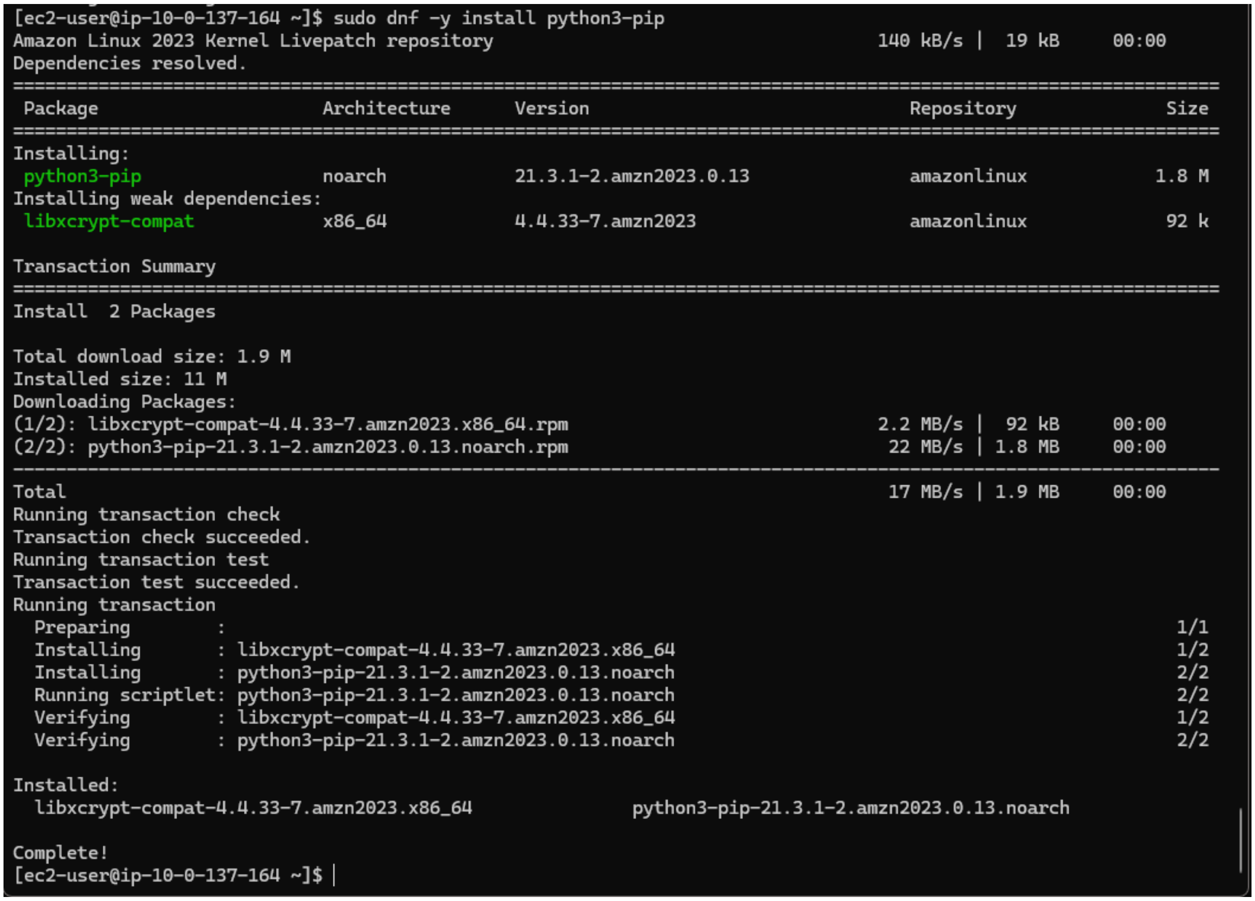
After reattaching the AppTierDynamoDBRole to my ***Application Tier*** *instances, I re-ran the metadata commands. This time, the EC2 Instance Metadata Service (IMDS) returned details about the attached IAM role and temporary security credentials. This confirmed that the Application Tier instances now had the proper permissions to interact with DynamoDB* ***without the need for hardcoded access keys****, validating that my least-privilege IAM setup was working correctly.*



```
[ec2-user@ip-10-0-137-164 ~]$ aws dynamodb put-item \
  --table-name Orders \
  --item '{
    "orderId":{"S":"ORD-1001"},
    "createdAt":{"S":"2025-08-31T12:00:00Z"},
    "status":{"S":"NEW"},
    "amount":{"N":"49.99"}
  }' \
  --region us-east-1
[ec2-user@ip-10-0-137-164 ~]$ aws dynamodb get-item \
  --table-name Orders \
  --key '{
    "orderId":{"S":"ORD-1001"},
    "createdAt":{"S":"2025-08-31T12:00:00Z"}
  }' \
  --region us-east-1 \
  --output json
{
    "Item": {
        "amount": {
            "N": "49.99"
        },
        "createdAt": {
            "S": "2025-08-31T12:00:00Z"
        },
        "orderId": {
            "S": "ORD-1001"
        },
        "status": {
            "S": "NEW"
        }
    }
}
[ec2-user@ip-10-0-137-164 ~]$ 
```

Figure 29/Validation: App Tier to DynamoDB

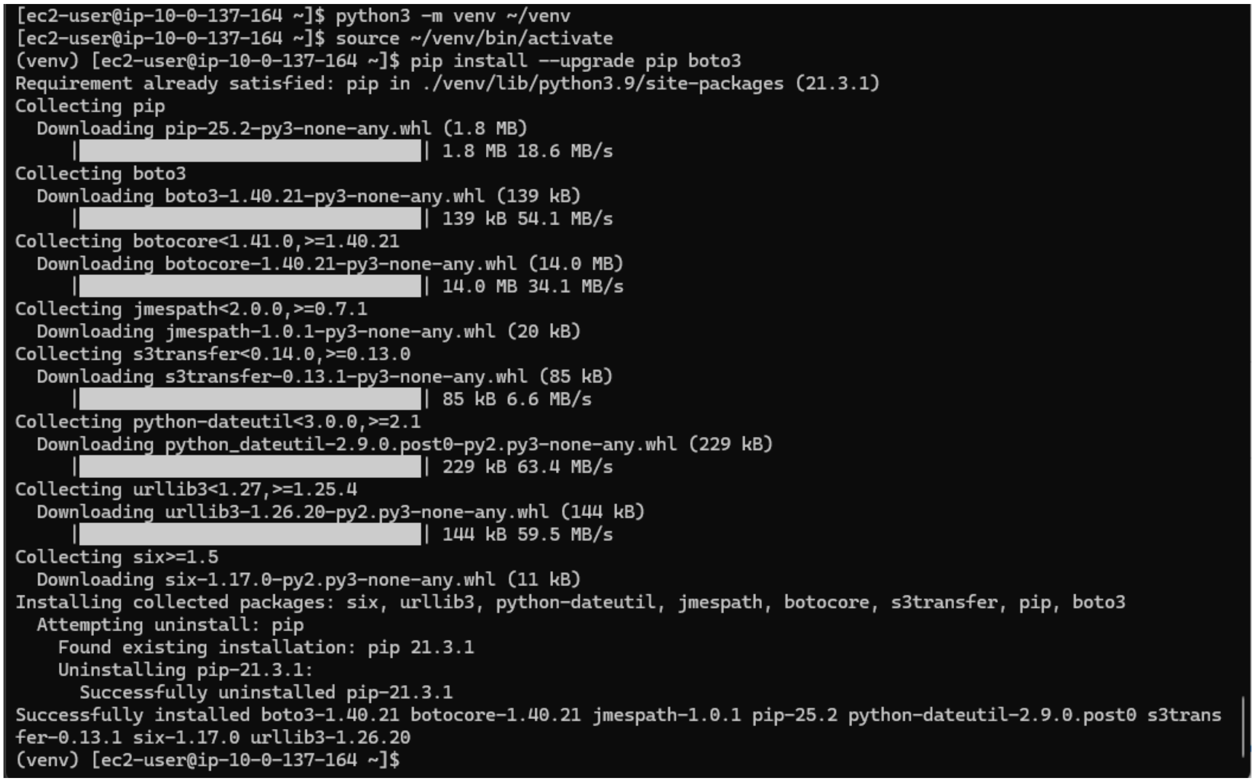
*From an **Application Tier EC2 instance**, I successfully ran a put-item command to insert a new record into the Orders table and then retrieved it with get-item. The returned JSON shows the correct attributes (orderId, createdAt, status, and amount), confirming that the IAM role, VPC endpoint, and DynamoDB integration are all working as intended. This validates secure, credential-free access from the Application Tier to the Database Tier.*



```
[ec2-user@ip-10-0-137-164 ~]$ sudo dnf -y install python3-pip
Amazon Linux 2023 Kernel Livepatch repository                                    140 kB/s |  19 kB     00:00
Dependencies resolved.
=================================================================================================================
 Package                     Architecture      Version                               Repository             Size
=================================================================================================================
Installing:
 python3-pip                 noarch            21.3.1-2.amzn2023.0.13                amazonlinux           1.8 M
Installing weak dependencies:
 libxcrypt-compat            x86_64            4.4.33-7.amzn2023                     amazonlinux            92 k

Transaction Summary
=================================================================================================================
Install  2 Packages

Total download size: 1.9 M
Installed size: 11 M
Downloading Packages:
(1/2): libxcrypt-compat-4.4.33-7.amzn2023.x86_64.rpm                             2.2 MB/s |  92 kB     00:00
(2/2): python3-pip-21.3.1-2.amzn2023.0.13.noarch.rpm                              22 MB/s | 1.8 MB     00:00
-----------------------------------------------------------------------------------------------------------------
Total                                                                             17 MB/s | 1.9 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                         1/1
  Installing       : libxcrypt-compat-4.4.33-7.amzn2023.x86_64                                               1/2
  Installing       : python3-pip-21.3.1-2.amzn2023.0.13.noarch                                               2/2
  Running scriptlet: python3-pip-21.3.1-2.amzn2023.0.13.noarch                                               2/2
  Verifying        : libxcrypt-compat-4.4.33-7.amzn2023.x86_64                                               1/2
  Verifying        : python3-pip-21.3.1-2.amzn2023.0.13.noarch                                               2/2

Installed:
  libxcrypt-compat-4.4.33-7.amzn2023.x86_64               python3-pip-21.3.1-2.amzn2023.0.13.noarch

Complete!
[ec2-user@ip-10-0-137-164 ~]$
```

Figure 30/ Installing Python on App Tier Instance

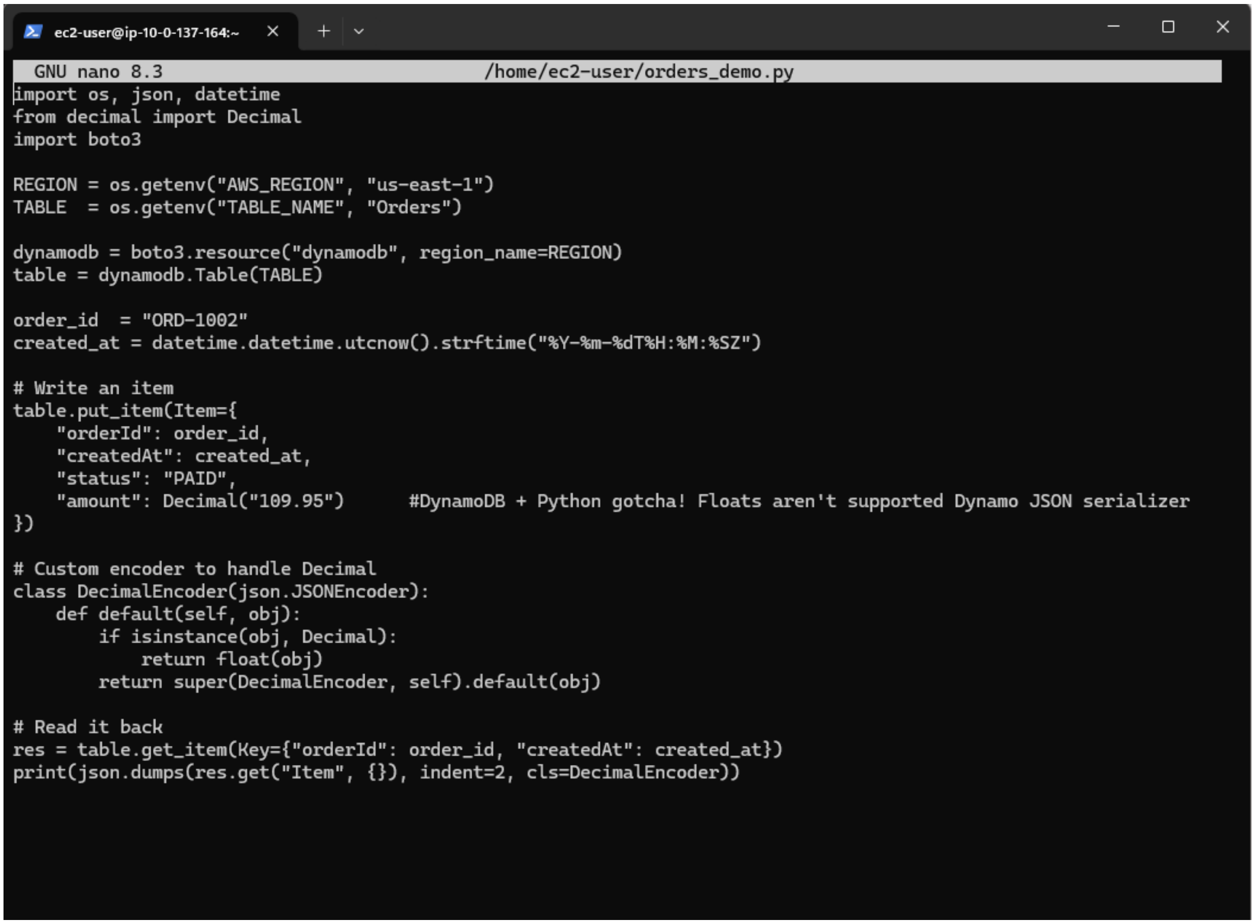
To prepare the environment for running a simple Python application with the boto3 SDK, I installed python3-pip on the Application Tier EC2 instance. This package manager allows me to easily install additional Python libraries, such as boto3, which I used to interact with DynamoDB directly from the instance.



```
[ec2-user@ip-10-0-137-164 ~]$ python3 -m venv ~/venv
[ec2-user@ip-10-0-137-164 ~]$ source ~/venv/bin/activate
(venv) [ec2-user@ip-10-0-137-164 ~]$ pip install --upgrade pip boto3
Requirement already satisfied: pip in ./venv/lib/python3.9/site-packages (21.3.1)
Collecting pip
  Downloading pip-25.2-py3-none-any.whl (1.8 MB)
     |████████████████████████████████| 1.8 MB 18.6 MB/s
Collecting boto3
  Downloading boto3-1.40.21-py3-none-any.whl (139 kB)
     |████████████████████████████████| 139 kB 54.1 MB/s
Collecting botocore<1.41.0,>=1.40.21
  Downloading botocore-1.40.21-py3-none-any.whl (14.0 MB)
     |████████████████████████████████| 14.0 MB 34.1 MB/s
Collecting jmespath<2.0.0,>=0.7.1
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting s3transfer<0.14.0,>=0.13.0
  Downloading s3transfer-0.13.1-py3-none-any.whl (85 kB)
     |████████████████████████████████| 85 kB 6.6 MB/s
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
     |████████████████████████████████| 229 kB 63.4 MB/s
Collecting urllib3<1.27,>=1.25.4
  Downloading urllib3-1.26.20-py2.py3-none-any.whl (144 kB)
     |████████████████████████████████| 144 kB 59.5 MB/s
Collecting six>=1.5
  Downloading six-1.17.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, urllib3, python-dateutil, jmespath, botocore, s3transfer, pip, boto3
  Attempting uninstall: pip
    Found existing installation: pip 21.3.1
    Uninstalling pip-21.3.1:
      Successfully uninstalled pip-21.3.1
Successfully installed boto3-1.40.21 botocore-1.40.21 jmespath-1.0.1 pip-25.2 python-dateutil-2.9.0.post0 s3trans
fer-0.13.1 six-1.17.0 urllib3-1.26.20
(venv) [ec2-user@ip-10-0-137-164 ~]$
```

Figure 31/ Virtual Environment and Installing Boto3

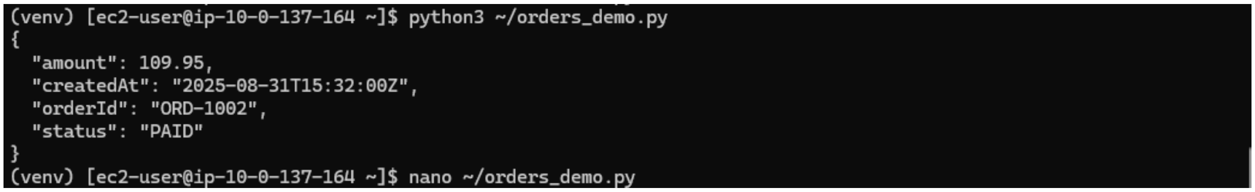
I then created a Python virtual environment and activated it to keep dependencies isolated. Inside the environment, I upgraded pip and installed the AWS SDK for Python (boto3). This setup allows me to write and run Python scripts that securely interact with DynamoDB using the IAM role attached to the instance.



```
  GNU nano 8.3                                /home/ec2-user/orders_demo.py
import os, json, datetime
from decimal import Decimal
import boto3

REGION = os.getenv("AWS_REGION", "us-east-1")
TABLE  = os.getenv("TABLE_NAME", "Orders")

dynamodb = boto3.resource("dynamodb", region_name=REGION)
table = dynamodb.Table(TABLE)

order_id  = "ORD-1002"
created_at = datetime.datetime.utcnow().strftime("%Y-%m-%dT%H:%M:%SZ")

# Write an item
table.put_item(Item={
    "orderId": order_id,
    "createdAt": created_at,
    "status": "PAID",
    "amount": Decimal("109.95")      #DynamoDB + Python gotcha! Floats aren't supported Dynamo JSON serializer
})

# Custom encoder to handle Decimal
class DecimalEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, Decimal):
            return float(obj)
        return super(DecimalEncoder, self).default(obj)

# Read it back
res = table.get_item(Key={"orderId": order_id, "createdAt": created_at})
print(json.dumps(res.get("Item", {}), indent=2, cls=DecimalEncoder))
```

*Figure 32/ Script *Decimal handling*

This Python script validates read and write operations from the Application Tier to DynamoDB. A custom DecimalEncoder was included to resolve the DynamoDB + Python limitation where numbers are stored as Decimal and cannot be directly serialized by json.dumps.



```
(venv) [ec2-user@ip-10-0-137-164 ~]$ python3 ~/orders_demo.py
{
  "amount": 109.95,
  "createdAt": "2025-08-31T15:32:00Z",
  "orderId": "ORD-1002",
  "status": "PAID"
}
(venv) [ec2-user@ip-10-0-137-164 ~]$ nano ~/orders_demo.py
```

Figure 33/ Run Script

Running the Python test script on the Application Tier instance successfully wrote a new order record to the Orders table and immediately retrieved it. The clean JSON output displayed in the console confirms that the Application Tier can securely interact with DynamoDB through the attached IAM role and VPC endpoint, completing the validation of the 3-tier architecture.

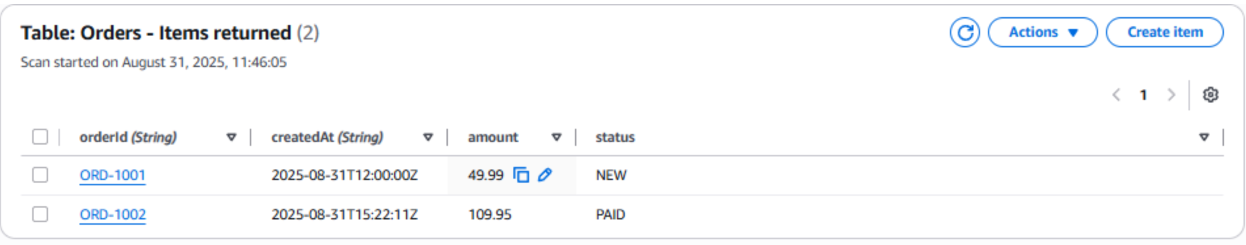


Table: Orders - Items returned (2)

Scan started on August 31, 2025, 11:46:05

□	orderId *(String)*	createdAt *(String)*	amount	status
□	ORD-1001	2025-08-31T12:00:00Z	49.99	NEW
□	ORD-1002	2025-08-31T15:22:11Z	109.95	PAID

Figure 34/ DynamoDB Console Validation

In the AWS Console, I explored the Orders table and confirmed that the records created through both the AWS CLI and the Python test script are present with the expected attributes. This provides an additional layer of validation that the 3-tier application is functioning correctly, with the Application Tier successfully writing to and reading from DynamoDB.

Conclusion

This project demonstrated the design and deployment of a secure, scalable **3-tier architecture on AWS**. By separating the web, application, and database tiers, I implemented industry best practices for isolation, availability, and resilience. End-to-end testing confirmed that each layer communicates as intended, with the Application Tier securely interacting with DynamoDB through IAM roles and private networking.

Teardown

Tearing down resources is just as important as building them, since it prevents unnecessary charges. Deleting in the correct order also avoids dependency errors (for example, a VPC cannot be deleted while subnets, gateways, or load balancers are still attached). Because I plan to repeat this project for practice, I use the following teardown checklist to ensure everything is removed cleanly before starting fresh:

1. **Application Layer (App Tier)**
   - Delete the Application Load Balancer (ALB) for the App Tier.
   - Delete the Auto Scaling Group for the App Tier.
   - Delete the Launch Template for the App Tier.
   - Verify the EC2 instances in the private subnets are terminated.
2. **Presentation Layer (Web Tier)**
   - Delete the Web Tier ALB.
   - Delete the Auto Scaling Group for the Web Tier.
   - Delete the Launch Template for the Web Tier.

    - Verify the EC2 instances in the public subnets are terminated.

3. **Database Layer**
    - Export any test data you want to keep.
    - Delete the DynamoDB table (Orders).
    - Delete the VPC Endpoint for DynamoDB.
4. **IAM Roles and Policies**
    - Detach and delete the AppTierDynamoDBRole.
    - Delete the custom least-privilege policy created for DynamoDB access.
5. **Networking (VPC and Subnets)**
    - Delete Security Groups created for the Web Tier and App Tier (ensure no instances are attached).
    - Delete Subnets (public and private).
    - Delete Route Tables (except the main route table, which is removed with the VPC).
    - Detach and delete the Internet Gateway.
    - **Release Elastic IPs** that are no longer attached to running instances.
    - Finally, delete the VPC.