

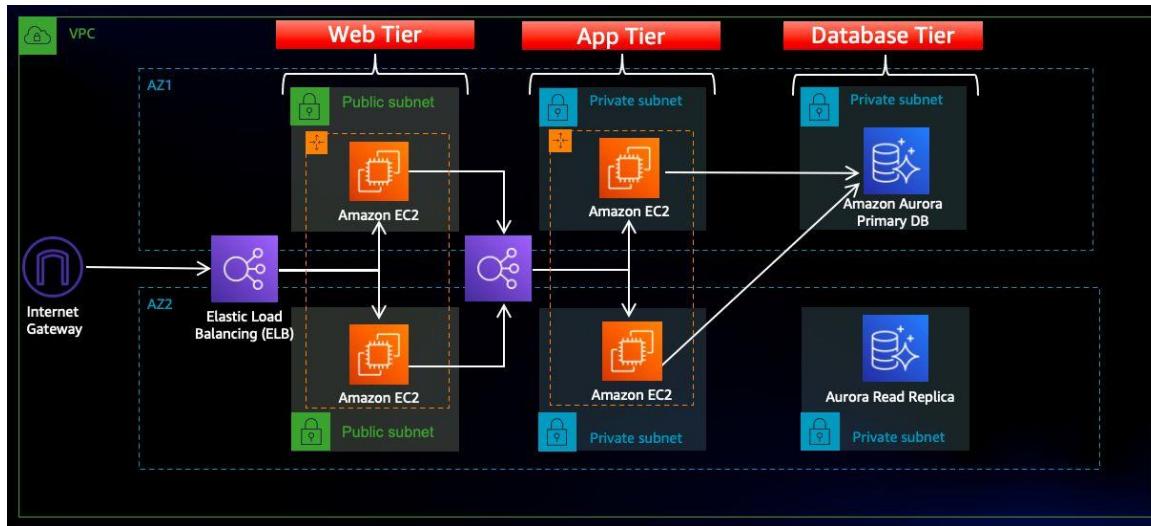
AWS Three Tier Web Architecture

Project hands on

by P.Lokesh(mech)

Introduction :

Architecture Overview :



In this architecture, a public-facing Application Load Balancer forwards client traffic to our web tier EC2 instances. The web tier is running Nginx web servers that are configured to serve a React.js website and redirects our API calls to the application tier's internal facing load balancer. The internal facing load balancer then forwards that traffic to the application tier, which is written in Node.js. The application tier manipulates data in an Aurora MySQL multi-AZ database and returns it to our web tier. Load balancing, health checks and autoscaling groups are created at each layer to maintain the availability of this architecture.

It contains total 6parts as below :

Part 0 : Set up

Part 1 : Networking & security

Part 2 : Database Deployment

Part 3 : App Tier Instance Deployment

Part 4 : Internal Load Balancing and Auto Scaling

Part 5 : Web Tier Instance Deployment

Part 6: External Load Balancer and Auto Scaling

Part 0 : Set up

For this workshop, we will be downloading the code from Github and upload it to S3 so our instances can access it. We will also create an AWS Identity and Access Management EC2 role so we can use AWS Systems Manager Session Manager to connect to our instances securely and without needing to create SSH key pairs.

Learning Objectives:

- S3 Bucket Creation.
- IAM EC2 Instance Role Creation.
- Download Code from Github Repository.

Download Code from Github :

Download the code from [this repository](#) into your local environment by running the command below. If you don't have git installed, you can just download the zip. Save it somewhere you can easily access.
git clone <https://github.com/aws-samples/aws-three-tier-web-architecture-workshop.git>

S3 Bucket Creation:

- 1) Navigate to the S3 service in the AWS console and create a new S3 bucket.

The screenshot shows the AWS S3 console with the 'General purpose buckets' tab selected. A table lists one bucket: 'codepipeline-us-east-1' (405128370979), located in 'US East (N. Virginia)' (us-east-1). The bucket has 'Bucket and objects not public' access and was created on February 4, 2024, at 08:33:34 (UTC+05:30). The left sidebar includes links for Buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Block Public Access settings for this account, Storage Lens, Dashboards, and Storage Lens groups.

- 2) Give it a unique name, and then leave all the defaults as in. Make sure to select the region that you intend to run this whole lab in. This bucket is where we will upload our code later.

The screenshot shows the 'Create bucket' wizard. In the 'General configuration' step, the 'AWS Region' is set to 'US East (N. Virginia) us-east-1'. The 'Bucket type' section shows 'General purpose' selected, with a note that it's recommended for most use cases and access patterns. Other options like 'Directory - New' are also listed. The left sidebar is identical to the previous screenshot.

IAM EC2 Instance Role Creation:

- 1) Navigate to the IAM dashboard in the AWS console and create an EC2 role.

The screenshot shows the IAM Roles creation wizard. Step 3, 'Name, review, and create', is displayed. Under 'Trusted entities', 'AWS service' is selected, with a note that it allows AWS services like EC2, Lambda, or others to perform actions in this account. Other options like 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy' are shown. Under 'Use case', 'EC2' is selected. The left sidebar shows 'Identity and Access Management (IAM)' with 'Roles' selected.

- 2) Select EC2 as the trusted entity .

- 3) When adding permissions, include the following AWS managed policies. You can search for them and select them. These policies will allow our instances to download our code from S3 and use Systems Manager Session Manager to securely connect to our instances without SSH keys through the AWS console.
- **AmazonSSMManagedInstanceCore**
 - **AmazonS3ReadOnlyAccess**.

IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Add permissions Info

Permissions policies (1/915) Info

Choose one or more policies to attach to your new role.

Filter by Type

Policy name	Type	Description
<input checked="" type="checkbox"/> AmazonSSMManagedInst...	AWS managed	The policy...

- 4) Give your role a name, and then click **Create Role**.

Part 1: Networking and Security

In this section we will be building out the VPC networking components as well as security groups that will add a layer of protection around our EC2 instances, Aurora databases, and Elastic Load Balancers

Learning Objectives:

- Create an isolated network with the following components:
 - VPC
 - Subnets
 - Route Tables
 - Internet Gateway
 - NAT gateway
 - Security Groups

VPC and Subnets

1) VPC Creation :

Navigate to the VPC dashboard in the AWS console and navigate to **Your VPCs** on the left hand side

VPC dashboard X

EC2 Global View X

Filter by VPC: Select a VPC

▼ Virtual private cloud Your VPCs

Your VPCs (3) Info

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
awsdemoworkshop	vpc-04a1ecf88ab7e6c3b	Available	10.0.0.0/16	-
three-tier-app	vpc-0506ded4ddab6035e	Available	10.0.0.0/16	-
-	vpc-07067d9c6e6de2bb5	Available	172.31.0.0/16	-

- 2) Make sure **VPC only** is selected, and fill out the VPC Settings with a **Name tag** and a **CIDR range** of your choice.

A VPC is an isolated portion of the AWS Cloud populated by AWS objects, such as Amazon

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 - optional
Creates a tag with a key of 'Name' and a value that you specify.
my-vpc-01

IPv4 CIDR block [Info](#)
 IPv4 CIDR manual input IPAM-allocated IPv4 CIDR block
IPv4 CIDR
10.0.0.0/24
CIDR block size must be between /16 and /28.

IPv6 CIDR block [Info](#)

NOTE: Make sure you pay attention to the region you're deploying all your resources in. You'll want to stay consistent for this workshop.

NOTE: Choose a CIDR range that will allow you to create at least 6 subnets.

Subnet Creation:

1. Next, create your subnets by navigating to **Subnets** on the left side of the dashboard and clicking **Create subnet**.

Subnets (11) [Info](#)

<input type="checkbox"/>	Name	Subnet ID	State	VPC
<input type="checkbox"/>	private-db-subnet-az2	subnet-0417c59905b7fe8c0	Available	vpc-04a1ecf8
<input type="checkbox"/>	app-tier-private-1a	subnet-0e63ab6a42801b273	Available	vpc-0506ded
<input type="checkbox"/>	private-db-subnet-az1	subnet-0551af335066ae8ce	Available	vpc-04a1ecf8
<input type="checkbox"/>	web-tier-public-1b	subnet-00b95fa63611f8c0d	Available	vpc-0506ded

Create subnet [Info](#)

VPC
VPC ID: Create subnets in this VPC
vpc-04a1ecf8ab7e6c3b (awsdemoworkshop)

Associated VPC CIDRs
IPv4 CIDRs
10.0.0.0/16

Subnet settings
Specify the CIDR blocks and Availability Zone for the subnet.

Subnet 1 of 1

Your final subnet setup should be similar to this. Verify that you have 3 subnets across 2 different availability zones.

Subnets (6) [Info](#)

<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 Cl...	Availability Zone
<input type="checkbox"/>	Public-Subnet-AZ-1	subnet-042290143e243f9a7	Available	vpc-0d2a264...	10.0.0.0/24	us-west-1b
<input type="checkbox"/>	Private-Subnet-AZ1	subnet-0f2b746756c354aaa	Available	vpc-0d2a264...	10.0.1.0/24	us-west-1b
<input type="checkbox"/>	Private-Subnet-AZ1	subnet-046bbef1cf38fb22	Available	vpc-0d2a264...	10.0.2.0/24	us-west-1b
<input type="checkbox"/>	Public-Subnet-AZ2	subnet-00511a4ff95e0335d	Available	vpc-0d2a264...	10.0.3.0/24	us-west-1c

Internet Connectivity:

Internet Gateway

1. In order to give the public subnets in our VPC internet access we will have to create and attach an Internet Gateway. On the left hand side of the VPC dashboard, select **Internet Gateways**.

The screenshot shows the AWS VPC dashboard with the 'Internet gateways' section selected. The list table displays three entries:

Name	Internet gateway ID	State	VPC ID
three-tier-igw	igw-0159be8ccc2637403	Attached	vpc-0506ded4ddab6035e three-tier
three-tier-internetgateway	igw-02657a8fd6a56c14d	Attached	vpc-04a1ecf8ab7e6c3b awsdemo
-	igw-05718a8d8508ca7d4	Attached	vpc-07067d9c6e6de2bb5

2. Create your internet gateway by simply giving it a name and clicking **Create internet gateway**.

The screenshot shows the 'Create internet gateway' wizard. The 'Internet gateway settings' step is displayed, where a name tag 'my-internet-gateway' is specified. Below it, there's a section for optional tags, which is currently empty.

3. After creating the internet gateway, attach it to your VPC that you create in the **VPC and Subnet Creation** step of the workshop. You have a couple options on how to do this, either with the creation success message or the **Actions** drop down.

The screenshot shows the AWS VPC dashboard with the 'Internet gateways' section selected. A specific internet gateway, 'igw-0a0ba40e6292c99a5', is viewed. The 'Actions' dropdown menu on the right includes options like 'Attach to VPC', 'Detach from VPC', 'Manage tags', and 'Delete'.

Then, select the correct VPC and click **Attach internet gateway**.

The screenshot shows the 'Attach to VPC' confirmation dialog. It displays a message: 'You do not have any VPCs. You need a VPC in order to attach an internet gateway.' Below this, there's a 'VPC' section with a note: 'Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.' A search bar 'Select a VPC' is provided. At the bottom, there are 'Cancel' and 'Attach internet gateway' buttons.

NAT Gateway

- In order for our instances in the app layer private subnet to be able to access the internet they will need to go through a NAT Gateway. For high availability, you'll deploy one NAT gateway in each of your **public subnets**. Navigate to **NAT Gateways** on the left side of the current dashboard and click **Create NAT Gateway**.

The screenshot shows the AWS VPC service dashboard. On the left sidebar, under 'Subnets', there is a link to 'NAT gateways'. The main area displays a table titled 'NAT gateways (2)'. The table has columns for Name, NAT gateway ID, Connectivity..., State, and Primary public IP. Two entries are listed:

Name	NAT gateway ID	Connectivity...	State	Primary public
app-tier-gw-2	nat-03027363edab0e329	Public	Available	44.213.155.125
app-tier-gw-1	nat-045293e7d69d9dc57	Public	Available	3.220.97.124

- Fill in the **Name**, choose one of the **public subnets** you created in part 2, and then allocate an Elastic IP. Click **Create NAT gateway**.

The screenshot shows the 'Create NAT gateway' wizard. The first step is 'NAT gateway settings'. It includes fields for 'Name - optional' (set to 'my-nat-gateway-01'), 'Subnet' (a dropdown menu), 'Connectivity type' (set to 'Public'), and 'Elastic IP allocation ID' (a dropdown menu). At the bottom right is a large orange 'Create' button.

- Repeat step 1 and 2 for the other subnet.

Routing Configuration:

- Navigate to Route Tables on the left side of the VPC dashboard and click Create route table First, let's create one route table for the web layer **public subnets** and name it accordingly.

The screenshot shows the AWS VPC service dashboard. On the left sidebar, under 'Virtual private cloud', there is a link to 'Route tables'. The main area displays a table titled 'Route tables (5)'. The table has columns for Name, Route table ID, Explicit subnet associations, Edge associations, and Main. Three entries are listed:

Name	Route table ID	Explicit subnet associations	Edge associations	Main
db-tier-1	rtb-0822a0b3c4dfc2e88	2 subnets	-	No
rt-public1	rtb-0af6d9be932204cad	subnet-00b95fa63611f8c...	-	No

Route table settings

Name - optional
Create a tag with a key of 'Name' and a value that you specify.

VPC
The VPC to use for this route table.

Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

You can add 50 more tags.

- After creating the route table, you'll automatically be taken to the details page. Scroll down and click on the **Routes tab** and **Edit routes**.

VPC dashboard EC2 Global View

Virtual private cloud
Your VPCs Subnets **Route tables** Internet gateways Egress-only internet gateways Carrier gateways DHCP option sets Elastic IPs Managed prefix lists Endpoints

rtb-0af6d9be932204cad / rt-public2

Details

Route table ID <input type="text" value="rtb-0af6d9be932204cad"/>	Main <input type="checkbox"/>	Explicit subnet associations <input type="text" value="subnet-00b95fa63611f8c0d / web-tier-public-1b"/>	Edge associations
VPC <input type="text" value="vpc-0506ded4ddab605e three-tier-app"/>	Owner ID <input type="text" value="975049895916"/>		

Routes (2)

Destination	Target	Status	Propagated
0.0.0.0/0	igw-0159be8ccc2637403	<input checked="" type="radio"/> Active	No
10.0.0.0/16	local	<input checked="" type="radio"/> Active	No

- Add a route that directs traffic from the VPC to the internet gateway. In other words, for all traffic destined for IPs outside the VPC CIDR range, add an entry that directs it to the internet gateway as a *target*. Save the changes.

VPC > Route tables > rtb-0af6d9be932204cad > Edit routes

Edit routes

Destination	Target	Status	Propagated
10.0.0.0/16	local <input type="text" value="local"/> <input type="button" value="Remove"/>	<input checked="" type="radio"/> Active	No
0.0.0.0/0	Internet Gateway <input type="text" value="Internet Gateway"/> <input type="button" value="Remove"/>	<input checked="" type="radio"/> Active	No

- Edit the *Explicit Subnet Associations* of the route table by navigating to the route table details again. Select **Subnet Associations** and click **Edit subnet associations**.

Filter by VPC: Select a VPC

Virtual private cloud
Your VPCs Subnets **Route tables** Internet gateways

rtb-0af6d9be932204cad / rt-public2

Subnet associations

Explicit subnet associations (1)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
web-tier-public-1b	subnet-00b95fa63611f8c0d	10.0.1.0/24	-

Select the two web layer public subnets you created earlier and click **Save associations**.

Available subnets (1/6)						
	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID	
<input type="checkbox"/>	app-tier-private-1a	subnet-0e63ab6a42801b273	10.0.16.0/20	-	rtb-0567a66f04f85e600 / app-tier-	
<input checked="" type="checkbox"/>	web-tier-public-1b	subnet-00b95fa63611f8c0d	10.0.1.0/24	-	rtb-0af6d9be932204cad / rt-public	
<input type="checkbox"/>	app-tier-private-1b	subnet-0316cedb6b48869e3	10.0.32.0/20	-	rtb-0567a66f04f85e600 / app-tier-	
<input type="checkbox"/>	web-tier-public-1a	subnet-010192cf02bb90b8c	10.0.0.0/24	-	rtb-09a9d552f1743e689 / rt-public	
<input type="checkbox"/>	db-private-1a	subnet-0ac4928c5f947564c	10.0.48.0/24	-	rtb-0822a0b3c4dfc2e88 / db-tier-1	
<input type="checkbox"/>	db-private-1b	subnet-0e46c1debe8d7694d	10.0.49.0/24	-	rtb-0822a0b3c4dfc2e88 / db-tier-1	

Selected subnets

subnet-00b95fa63611f8c0d / web-tier-public-1b X

Cancel Save associations

- Now create 2 more route tables, one for each app layer private subnet in each availability zone. These route tables will route app layer traffic destined for outside the VPC to the NAT gateway in the respective availability zone, so add the appropriate routes for that.

Route table settings

Name - optional
Create a tag with a key of 'Name' and a value that you specify.

VPC
The VPC to use for this route table.

Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
No tags associated with the resource.

You can add 50 more tags.

Cancel Create route table

VPC > Route tables > rtb-0af6d9be932204cad > Edit routes

Edit routes

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	Internet Gateway	Active	No

Add route Remove

Cancel Preview Save changes

Once the route tables are created and routes added, add the appropriate subnet associations for each of the app layer private subnets.

Available subnets (1/6)

Available subnets (1/6)						
	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID	
<input type="checkbox"/>	app-tier-private-1a	subnet-0e63ab6a42801b273	10.0.16.0/20	-	rtb-0567a66f04f85e600 / app-tier-	
<input type="checkbox"/>	web-tier-public-1b	subnet-00b95fa63611f8c0d	10.0.1.0/24	-	rtb-0af6d9be932204cad / rt-public	
<input type="checkbox"/>	app-tier-private-1b	subnet-0316cedb6b48869e3	10.0.32.0/20	-	rtb-0567a66f04f85e600 / app-tier-	
<input type="checkbox"/>	web-tier-public-1a	subnet-010192cf02bb90b8c	10.0.0.0/24	-	rtb-09a9d552f1743e689 / rt-public	
<input checked="" type="checkbox"/>	db-private-1a	subnet-0ac4928c5f947564c	10.0.48.0/24	-	rtb-0822a0b3c4dfc2e88 / db-tier-1	
<input type="checkbox"/>	db-private-1b	subnet-0e46c1debe8d7694d	10.0.49.0/24	-	rtb-0822a0b3c4dfc2e88 / db-tier-1	

Selected subnets

subnet-0ac4928c5f947564c / db-private-1a X

Cancel Save associations

Security Groups:

1. Security groups will tighten the rules around which traffic will be allowed to our Elastic Load Balancers and EC2 instances. Navigate to **Security Groups** on the left side of the VPC dashboard, under **Security**.

The screenshot shows the AWS VPC Security Groups list page. On the left, there's a navigation sidebar with options like Endpoint services, NAT gateways, Peering connections, Security (Network ACLs, Security groups), DNS firewall (Rule groups, Domain lists), and Network Firewall. The main area displays a table with columns: Name, Security group ID, Security group name, and VPC ID. There are three entries:

- Name: -, Security group ID: sg-03bd3421cdf1a362a, Security group name: launch-wizard-4, VPC ID: vpc-07067d9c6e6de2bb5
- Name: -, Security group ID: sg-00989e029acf35f12, Security group name: default, VPC ID: vpc-04a1ecf8ab7e6c3b
- Name: -, Security group ID: sg-0063bfc09b414fb9e, Security group name: launch-wizard-3, VPC ID: vpc-07067d9c6e6de2bb5

A search bar at the top says "Find resources by attribute or tag". Buttons for Actions, Export security groups to CSV, and Create security group are at the top right.

2. The first security group you'll create is for the public, **internet facing** load balancer. After typing a name and description, add an inbound rule to allow **HTTP** type traffic for your **IP**.

This is a screenshot of the "Create security group" form. It has sections for Basic details, Inbound rules, and Outbound rules. The Basic details section includes fields for Security group name (MyWebServerGroup), Description (Allows SSH access to developers), and VPC (vpc-07067d9c6e6de2bb5). The Inbound rules section shows one rule: Type: HTTP, Protocol: TCP, Port range: 80, Source: My IP (110.235.236.164/32).

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	My IP 110.235.236.164/32	

3. The second security group you'll create is for the public instances in the web tier. After typing a name and description, add an inbound rule that allows **HTTP** type traffic from your internet facing load balancer security group you created in the previous step. This will allow traffic from your public facing load balancer to hit your instances. Then, add an additional rule that will allow HTTP type traffic for your IP. This will allow you to access your instance when we test.

This is a screenshot of the "Create security group" form for the web tier. The Basic details section includes fields for Security group name (webtiersg), Description (sg for webtier), and VPC (vpc-07067d9c6e6de2bb5). The Inbound rules section shows two rules: one from the previous step (HTTP, TCP, 80, Cus..., 110.235.236.205/32) and another new rule (Custom TCP, TCP, 0, Cus..., sg-016744770043f8e4e, sg-019a484f8011a8de9).

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-037447dc0dbd7d80f	HTTP	TCP	80	Cus... 110.235.236.205/32	
sgr-016744770043f8e4e	Custom TCP	TCP	0	Cus... sg-019a484f8011a8de9	

4. The third security group will be for our internal load balancer. Create this new security group and add an inbound rule that allows **HTTP** type traffic from your public instance security group. This will allow traffic from your web tier instances to hit your internal load balancer.

This is a screenshot of the "Create security group" form for the internal load balancer. The Basic details section includes fields for Security group name (internal lb), Description (sg for internal lb), and VPC (vpc-07067d9c6e6de2bb5). The Inbound rules section shows one rule: Type: HTTP, Protocol: TCP, Port range: 80, Source: CIDR blocks (web-tier-sg | sg-0cd2adfff74d9219).

CIDR blocks	Security Groups	Prefix lists
	web-tier-sg sg-0cd2adfff74d9219	

5. The fourth security group we'll configure is for our private instances. After typing a name and description, add an inbound rule that will allow **TCP** type traffic on port **4000** from the **internal load balancer security group** you created in the previous step. This is the port our app tier application is running on and allows our internal load balancer to forward traffic on this port to our private instances. You should also add another route for port **4000** that allows **your IP** for testing.

Basic details

Security group name [Info](#)
privateinstances-sg
Name cannot be edited after creation.

Description [Info](#)
sg for privateinstances

VPC [Info](#)
vpc-0506ded4ddab6035e (three-tier-app)

Type [Info](#) Protocol [Info](#) Port range [Info](#) Source [Info](#) Description - optional [Info](#)
Custom TCP TCP 4000 Cus... Delete
CIDR blocks
Security Groups
internal-lb-sg | sg-02d6c1ac0c3c36ae3
Prefix lists

Add rule

6. The fifth security group we'll configure protects our private database instances. For this security group, add an inbound rule that will allow traffic from the private instance security group to the MYSQL/Aurora port (3306).

Basic details

Security group name [Info](#)
db-sg
Name cannot be edited after creation.

Description [Info](#)
sg for dtb

VPC [Info](#)
vpc-0506ded4ddab6035e (three-tier-app)

Inbound rules [Info](#)

Type [Info](#) Protocol [Info](#) Port range [Info](#) Source [Info](#) Description [Info](#)
MySQL/Aurora TCP 3306 Cus... Delete
CIDR blocks
Security Groups
sg for dtb | sg-0dc62dcedbdff887c0
Prefix lists

Add rule

Part 2: Database Deployment

This section of the workshop will walk you through deploying the database layer of the three tier architecture.

Learning Objectives:

Deploy Database Layer

- Subnet Groups
- Multi-AZ Database

Subnet Groups:

1. Navigate to the RDS dashboard in the AWS console and click on **Subnet groups** on the left hand side. Click **Create DB subnet group**.

Amazon RDS X

RDS > Subnet groups

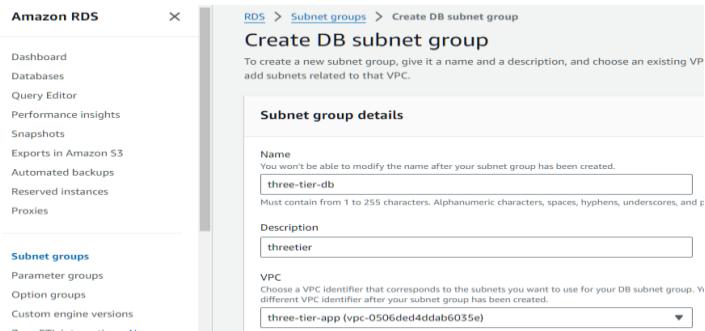
Subnet groups (2) [Create DB subnet group](#)

<input type="checkbox"/>	Name	Description	Status	VPC
<input type="checkbox"/>	db-3tier-app-sng	subnet group for db 3tier	Complete	vpc-0506ded4ddab6035e
<input type="checkbox"/>	three-tier-db-subnet-group	three-tier-db-subnet-group	Complete	vpc-04a1ecf88ab7e6c3b

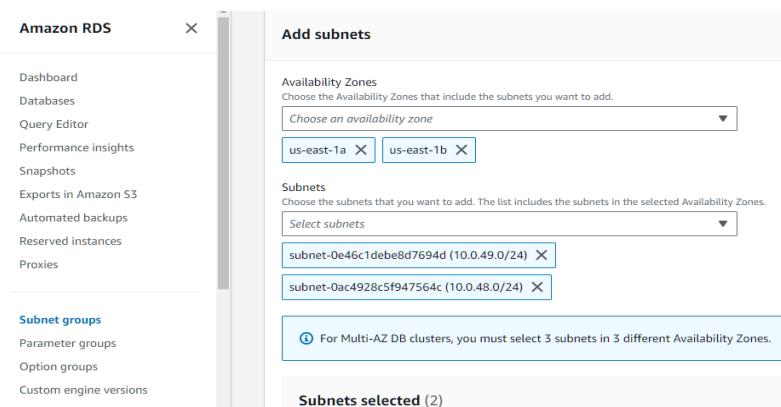
Filter by subnet group 1

Subnet groups

- Give your subnet group a name, description, and choose the VPC we created.

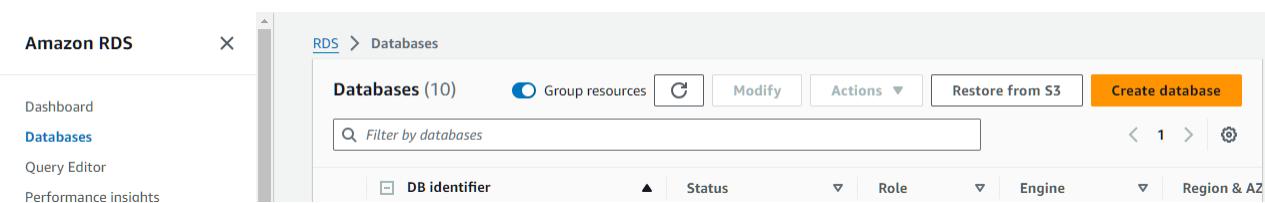


- When adding subnets, make sure to add the subnets we created in each availability zone specifically for our database layer. You may have to navigate back to the VPC dashboard and check to make sure you're selecting the correct subnet IDs.

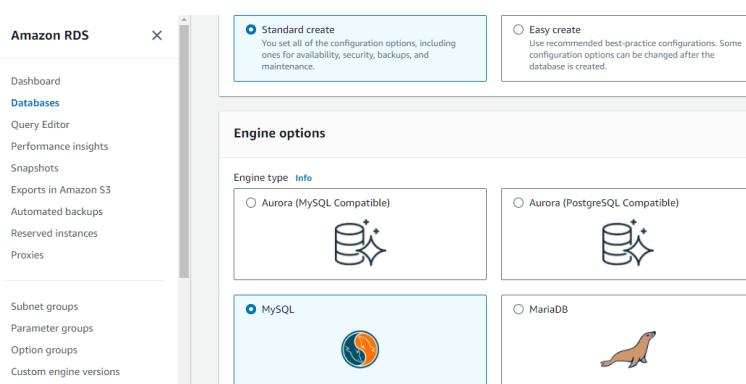


Database Deployment:

- Navigate to **Databases** on the left hand side of the RDS dashboard and click **Create database**.



- We'll now go through several configuration steps. Start with a **Standard create** for this **MySQL (FREE TIER)** database. Leave the rest of the defaults in the **Engine options** as default.



3. Under the **Templates** section choose **FREE TIER** since this isn't being used for production at the moment. Under **Settings** set a username and password of your choice and note them down since we'll be using password authentication to access our database.

Master username [Info](#)
Type a login ID for the master user of your DB instance.

1 to 16 alphanumeric characters. The first character must be a letter.

Manage master credentials in AWS Secrets Manager
Manage master user credentials in Secrets Manager. RDS can generate a password for you and manage it throughout its lifecycle.

If you manage the master user credentials in Secrets Manager, some RDS features aren't supported. [Learn more](#)

Auto generate a password
Amazon RDS can generate a password for you, or you can specify your own password.

Master password [Info](#)

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote)', "(double quote)", @ (at sign).

Confirm master password [Info](#)

4. Under **Connectivity**, set the VPC, choose the subnet group we created earlier, and select no for public access

Compute resource
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

Virtual private cloud (VPC) [Info](#)
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

three-tier-app (vpc-0506ded4ddab6035e)
6 Subnets, 2 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

DB subnet group [Info](#)
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

db-3tier-app-sng
2 Subnets, 2 Availability Zones

Set the security group we created for the database layer, make sure **password authentication** is selected as our authentication choice, and create the database.

Dashboard

Databases

Query Editor

Performance insights

Snapshots

Exports in Amazon S3

Automated backups

Reserved instances

Proxies

Subnet groups

Parameter groups

Custom engine versions

Zero-ETL integrations [New](#)

▶ Additional configuration

Database authentication

Database authentication options [Info](#)

Password authentication
Authenticates using database passwords.

Password and IAM database authentication
Authenticates using the database password and user credentials through AWS IAM users and roles.

Password and Kerberos authentication
Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.

- When your database is provisioned, you should see your database available. Note down the writer endpoint for your database for later use.

Databases

Query Editor

Performance insights

Snapshots

Exports in Amazon S3

Automated backups

Reserved instances

Proxies

Connectivity & security

Endpoint & port

Endpoint
threetier-webapp.c7w8g2e2o4j7.us-east-1.rds.amazonaws.com

Networking

Availability Zone
us-east-1a

VPC
three-tier-app (vpc-0506ded4ddab6035e)

Security

VPC security groups
db-sg (sg-0dc62dcedbdf887c0)
Active

Publicly accessible
No

Part 3: App Tier Instance Deployment

In this section of our workshop we will create an EC2 instance for our app layer and make all necessary software configurations so that the app can run. The app layer consists of a Node.js application that will run on port 4000. We will also configure our database with some data and tables.

Learning Objectives:

- Create App Tier Instance
- Configure Software Stack
- Configure Database Schema
- Test DB connectivity

App Instance Deployment:

1. Navigate to the EC2 service dashboard and click on **Instances** on the left hand side. Then, click **Launch Instances**.

Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
i-07e4ea165e938ed73	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1c
i-0db0b17fe2630e7bb	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b
i-0eabd8943e3247f62	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b

2. Select the first Amazon Linux 2 AMI

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type
ami-0cf10cdf9fc62d37 (64-bit (x86)) / ami-023c9d904d7c3bf72 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

3. We'll be using the free tier eligible **T.2 micro** instance type. Select that and click **Next: Configure Instance Details.**

The screenshot shows the 'Instance type' section of the configuration interface. A dropdown menu is open, showing 't2.micro' as the selected option. The dropdown also lists other options like 't2.small', 't2.medium', etc., along with their descriptions. To the right of the dropdown, there's a 'Free tier eligible' badge and a toggle switch for 'All generations'. Below the dropdown, a link 'Compare instance types' is visible. At the bottom of the section, a note says 'Additional costs apply for AMIs with pre-installed software'.

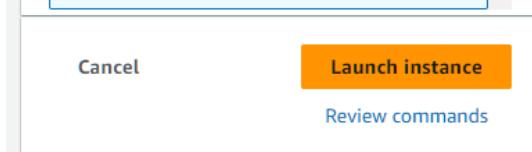
4. When configuring the instance details, make sure to select to correct **Network, subnet**, and **select existing group** we created. Note that this is the app layer, so use one of the private subnets we created for this layer.

The screenshot shows the 'Network settings' section. It includes fields for 'VPC - required' (set to 'vpc-0506ded4ddab6035e (three-tier-app) 10.0.0.0/16'), 'Subnet' (set to 'subnet-0e63a66a42801b273 app-tier-private-1a'), 'Auto-assign public IP' (set to 'Disable'), and 'Firewall (security groups)' (set to 'Select existing security group' with 'app-tier-sg sg-0db5b0177a1ad537a' selected). There are also 'Create new subnet' and 'Create new security group' buttons.

5. We have to configure **IAM INSTANCE PROFILEROLE** in advanced settings & then hit create Instance.

The screenshot shows the 'Advanced details' section. It includes fields for 'Domain join directory' (set to 'Select') and 'IAM instance profile' (set to 'demoEC2-role arn:aws:iam::975049895916:instance-profile/demoEC2-role'). To the right of each field are 'Create new directory' and 'Create new IAM profile' buttons.

You'll be taken to a page where you can click launch instance, and you'll see the instance you just launched.



Connect to Instance:

1. Navigate to your list of running EC2 Instances by clicking on **Instances** on the left hand side of the EC2 dashboard. When the instance state is running, connect to your instance by clicking the checkmark box to the left of the instance, and click the connect button on the top right corner of the dashboard. Select the Session Manager tab, and click connect. This will open a new browser tab for you.

NOTE: If you get a message saying that you cannot connect via session manager, then check that your instances can route to your NAT gateways and verify that you gave the necessary permissions on the IAM role for the Ec2 instance.

The image shows two screenshots. On the left is the AWS EC2 Instances list, titled "Instances (1/11) Info". It shows one instance named "repository files" with ID i-07e4ea165e938ed73, which is running. On the right is a "Connect to instance" dialog for the same instance. The dialog has tabs for "EC2 Instance Connect", "Session Manager" (which is selected), "SSH client", and "EC2 serial console". The "Session Manager" tab contains a warning about IAM instance profiles and troubleshooting links. Below the tabs, it says "Session Manager usage: Connect to your instance without SSH keys, a bastion host, or opening any inbound ports."

2. When you first connect to your instance like this, you will be logged in as ssm-user which is the default user. Switch to ec2-user by executing the following command in the browser terminal:
sudo -su ec2-user
3. Let's take this moment to make sure that we are able to reach the internet via our NAT gateways. If your network is configured correctly up till this point, you should be able to ping the google DNS servers:

ping 8.8.8.8

You should see a transmission of packets. Stop it by pressing cntrl c.

NOTE: If you can't reach the internet then you need to double check your route tables and subnet associations to verify if traffic is being routed to your NAT gateway!

Session ID: root-087f6065d30da15d9 Instance ID: i-00e85beb0c592beec

```
sh-5.2$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=113 time=0.924 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=113 time=0.916 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=113 time=0.911 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=113 time=0.957 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3138ms
rtt min/avg/max/mdev = 0.911/0.927/0.957/0.017 ms
sh-5.2$ █
```

Configure Database:

1. To configure the database of **Mysql** , firstly have to download package by using following **wget & rpm** commands :

steps for installing mysql :

```
sudo wget https://dev.mysql.com/get/mysql57-community-release-el7-
11.noarch.rpm
```

```
sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022
```

```
sudo yum install https://dev.mysql.com/get/mysql57-community-release-el7-
11.noarch.rpm
```

```
Total                                         65 MB/s | 35 MB   00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing           : 1/1
  Installing         : mysql-community-common-5.7.44-1.el7.x86_64 1/4
  Installing         : mysql-community-libs-5.7.44-1.el7.x86_64 2/4
  Running scriptlet: mysql-community-libs-5.7.44-1.el7.x86_64 2/4
  Installing         : ncurses-compat-libs-6.2-4.20200222.amzn2023.0.5.x86_64 3/4
  Installing         : mysql-community-client-5.7.44-1.el7.x86_64 4/4
  Running scriptlet: mysql-community-client-5.7.44-1.el7.x86_64 4/4
  Verifying          : ncurses-compat-libs-6.2-4.20200222.amzn2023.0.5.x86_64 1/4
  Verifying          : mysql-community-client-5.7.44-1.el7.x86_64 2/4
  Verifying          : mysql-community-common-5.7.44-1.el7.x86_64 3/4
  Verifying          : mysql-community-libs-5.7.44-1.el7.x86_64 4/4

Installed:
  mysql-community-client-5.7.44-1.el7.x86_64          mysql-community-common-5.7.44-1.el7.x86_64      mysql-community-libs-5.7.44-1.el7.x86_64
  ncurses-compat-libs-6.2-4.20200222.amzn2023.0.5.x86_64

Complete!
```

1. Start by downloading the MySQL CLI:

```
sudo yum install mysql -y
```

2. Initiate your DB connection with your Aurora RDS writer endpoint. In the following command, replace the RDS writer endpoint and the username, and then execute it in the browser terminal:

```
mysql -h CHANGE-TO-YOUR-RDS-ENDPOINT -u CHANGE-TO-USER-NAME -p
```

You will then be prompted to type in your password. Once you input the password and hit enter, you should now be connected to your database.

NOTE: If you cannot reach your database, check your credentials and security groups.

3. Create a database called **webappdb** with the following command using the MySQL CLI:

CREATE DATABASE webappdb;

You can verify that it was created correctly with the following command:

SHOW DATABASES;

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| webappdb           |
+--------------------+
5 rows in set (0.00 sec)
```

4. Create a data table by first navigating to the database we just created:

USE webappdb;

Then, create the following **transactions** table by executing this create table command:

CREATE TABLE IF NOT EXISTS transactions(id INT NOT NULL

AUTO_INCREMENT, amount DECIMAL(10,2), description

VARCHAR(100), PRIMARY KEY(id));

Verify the table was created:

SHOW TABLES;

```
+---------------------+
| Tables_in_webappdb |
+---------------------+
| transactions       |
+---------------------+
1 row in set (0.00 sec)
```

5. Insert data into table for use/testing later:

INSERT INTO transactions (amount,description) VALUES ('400','groceries');

Verify that your data was added by executing the following command:

SELECT * FROM transactions;

id	amount	description
1	400.00	groceries
2	400.00	groceries

2 rows in set (0.00 sec)

- When finished, just type **exit** and hit enter to exit the MySQL client.

Configure App Instance:

- The first thing we will do is update our database credentials for the app tier. To do this, open the **application-code/app-tier/DbConfig.js** file from the github repo in your favorite text editor on your computer. You'll see empty strings for the hostname, user, password and database. Fill this in with the credentials you configured for your database, the **writer** endpoint of your database as the hostname, and **webappdb** for the database. Save the file.

NOTE: This is NOT considered a best practice, and is done for the simplicity of the lab. Moving these credentials to a more suitable place like Secrets Manager is left as an extension for this workshop.

The screenshot shows the AWS S3 console interface. At the top, the path is shown as: Amazon S3 > Buckets > demoweb-app-loke-1 > app-tier/. Below the path, the folder name "app-tier/" is displayed. On the right, there is a "Copy S3 URI" button. Below the folder name, there are two tabs: "Objects" (which is selected) and "Properties". Under the "Objects" tab, there is a sub-header "Objects (6) Info". Below this, there is a toolbar with various actions: Copy, Copy S3 URI, Copy URL, Download, Open, Delete, Actions (with a dropdown arrow), Create folder, and Upload. A note below the toolbar states: "Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)". The main area shows a table of objects with columns: Name, Type, Last modified, Size, and Storage class. The objects listed are: dbConfig.js (JSON, 1.2 KB, Standard), index.js (JS, 3.2 KB, Standard), package-lock.json (JSON, 42.9 KB, Standard), package.json (JSON, 682.0 B, Standard), README.md (MD, 14.0 B, Standard), and TransactionService.js (JS, 1.8 KB, Standard). The "Actions" column for each object has a dropdown arrow.

- Upload the **app-tier** folder to the S3 bucket that you created in part 0.
- Go back to your SSM session. Now we need to install all of the necessary components to run our backend application. Start by installing NVM (node version manager).

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
```

```
source ~/.bashrc
```

4. Next, install a compatible version of Node.js and make sure it's being used

```
nvm install 16
```

```
nvm use 16
```

5. PM2 is a daemon process manager that will keep our node.js app running when we exit the instance or if it is rebooted. Install that as well.

```
npm install -g pm2
```

6. Now we need to download our code from our s3 buckets onto our instance. In the command below, replace BUCKET_NAME with the name of the bucket you uploaded the **app-tier** folder to:

```
cd ~/
```

```
aws s3 cp s3://BUCKET_NAME/app-tier/ app-tier --recursive
```

7. Navigate to the app directory, install dependencies, and start the app with pm2.

```
cd ~/app-tier
```

```
aws s3 cp s3:// BUCKET_NAME/ . --recursive
```

```
ls
```

```
npm install
```

```
pm2 start index.js
```

```
Run `npm audit` for details.
[ec2-user@ip-10-0-0-79 app-tier]$ pm2 start index.js
[PM2] Starting /home/ec2-user/app-tier/index.js in fork_mode (1 instance)
[PM2] Done.



| id | name  | namespace | version | mode | pid   | uptime | ø | status | cpu | mem    | user     | watching |
|----|-------|-----------|---------|------|-------|--------|---|--------|-----|--------|----------|----------|
| 0  | index | default   | 1.0.0   | fork | 20302 | 0s     | 0 | online | 0%  | 24.4mb | ec2-user | disabled |


```

To make sure the app is running correctly run the following:

```
pm2 list
```

If you see a status of online, the app is running. If you see errored, then you need to do some troubleshooting. To look at the latest errors, use this command:

```
pm2 logs
```

NOTE: If you're having issues, check your configuration file for any typos, and double check that you have followed all installation commands till now.

8. Right now, pm2 is just making sure our app stays running when we leave the SSM session. However, if the server is interrupted for some reason, we still want the app to start and keep running. This is also important for the AMI we will create:

pm2 startup

Test App Tier:

Now let's run a couple tests to see if our app is configured correctly and can retrieve data from the database.

To hit out health check endpoint, copy this command into your SSM terminal. This is our simple health check endpoint that tells us if the app is simply running.

[curl http://localhost:4000/health](http://localhost:4000/health)

The response should looks like the following:

"This is the health check"

Next, test your database connection. You can do that by hitting the following endpoint locally:

[curl http://localhost:4000/transaction](http://localhost:4000/transaction)

If you see both of these responses, then your networking, security, database and app configurations are correct.

7. Congrats! Your app layer is fully configured and ready to go.

Part 4: Internal Load Balancing and Auto Scaling

In this section of the workshop we will create an Amazon Machine Image (AMI) of the app tier instance we just created, and use that to set up autoscaling with a load balancer in order to make this tier highly available.

Learning Objectives:

- Create an AMI of our App Tier
- Create a Launch Template
- Configure Autoscaling
- Deploy Internal Load Balancer

App Tier AMI

1. Navigate to **Instances** on the left hand side of the EC2 dashboard. Select the app tier instance we created and under **Actions** select **Image and templates**. Click **Create Image**.

Instances (1/10) Info		Actions	Launch instances								
Name	Instance ID	Instance state	Instance type	Status check	Connect	View details	Manage instance state	Networking	Security	Image and templates	Monitor and troubleshoot
repository files	i-07e4ea165e938ed73	Stopped	t2.micro	-							
	i-0592a5b54e671fd17	Running	t2.micro	2/2 ch							
	i-08251ef7f05c8f7de	Running	t2.micro	Create image							
	i-0130066ad3e6c9648	Running	t2.micro	Create template from instance							
webtier	i-00e85beb0c592beec	Stopped	t2.micro	-							
	i-03816f71069b6fda5	Running	t2.micro	2/2 checks passed							
threapptier	i-06630b81d70ee6970	Stopped	t2.micro	-							

2. Give the image a name and description and then click **Create image**. This will take a few minutes, but if you want to monitor the status of image creation you can see it by clicking **AMIs** under **Images** on the left hand navigation panel of the EC2 dashboard.

Create image Info

An image (also referred to as an AMI) defines the programs and settings that are applied when you launch an EC2 instance. You can create an image from the console or from an instance.

Instance ID
i-06630b81d70ee6970 (threapptier)

Image name

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

Image description - optional

Maximum 255 characters

No reboot

Create image

Target Group

1. While the AMI is being created, we can go ahead and create our target group to use with the load balancer. On the EC2 dashboard navigate to **Target Groups** under **Load Balancing** on the left hand side. Click on **Create Target Group**.

Target groups (3) <small>Info</small>					
Name	ARN	Port	Protocol	Target type	Actions
threapp-tier	arn:aws:elasticloadbalancing:us-east-1:123456789012:targetgroup/threapp-tier	4000	HTTP	Instance	
0 target groups selected					

Select a target group above.

2. The purpose of forming this target group is to use with our load balancer so it may balance traffic across our private app tier instances. Select Instances as the target type and give it a name.

Basic configuration

Settings in this section can't be changed after the target group is created.

Choose a target type

Instances

- Supports load balancing to instances within a specific VPC.
- Facilitates reusing multiple IP addresses and network interfaces on the same instance.
- Offers failover with minimal downtime that doesn't affect ongoing communication between application and instance.
- Imports IP ranges, creating end-to-end IP communication and IPv4-to-IPv6 failover.

IP addresses

- Supports load balancing to VPC and non-VPC resources.
- Facilitates reusing multiple IP addresses and network interfaces on the same instance.
- Offers failover with minimal downtime that doesn't affect ongoing communication between application and instance.
- Imports IP ranges, creating end-to-end IP communication and IPv4-to-IPv6 failover.

Target group

app-tiertargetgroup

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol

Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed after creation.

HTTP

80

IP address type

Only targets with the indicated IP address type can be registered to this target group.

IPv4

Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.

IPv6

Then, set the protocol to **HTTP** and the port to 4000. Remember that this is the port our Node.js app is running on. Select the VPC we've been using thus far, and then change the health check path to be **/health**. This is the health check endpoint of our app. Click **Next**.

Protocol : Port
Choose a protocol for your target group that corresponds to the Load Balancer type that will route traffic to it. Some protocols now include anomaly detection for the targets and you can set mitigation options once your target group is created. This choice cannot be changed.
after creation

HTTP 4000
1-65535

IP address type
Only targets with the indicated IP address type can be registered to this target group.

IPv4
Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.

IPv6
Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). [Learn more](#)

VPC
Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected above are available in this list.

three-tier-app
vpc-050b0ed4d4db6035e
IPv4: 10.0.0.0/16

Protocol version
 HTTP1

3. We are **NOT** going to register any targets for now, so just skip that step and create the target group.

Register targets

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Available instances (6)

Instance ID	Name	State	Security groups
i-08251ef7f05c8f7de	Running	app-tier-sg	
i-055a579f7369a491	Running	web-tier-sg	

Targets (0)

Instance ID	Name	Port	State	Security groups	Zone	Private IPv4 address	Subnet
No instances added yet							

Specify instances above, or leave the group empty if you prefer to add targets later.

0 pending Cancel Previous Create target group

Internal Load Balancer:

1. On the left hand side of the EC2 dashboard select **Load Balancers** under **Load Balancing** and click **Create Load Balancer**.

Lifecycle Manager

Network & Security

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

Load Balancing

Load Balancers

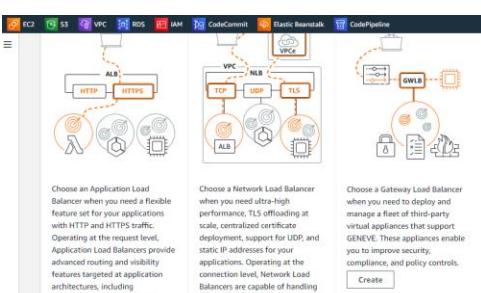
Load balancers (3)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	DNS name	State	VPC ID	Availability Zones
internal	internal.internalth-20016	Active	vpc-050b0ed4d4db6035e	2 Availability Zones

0 load balancers selected

Select a load balancer above.

2. We'll be using an **Application Load Balancer** for our **HTTP** traffic so click the create button for that option.



- After giving the load balancer a name, be sure to select **internal** since this one will not be public facing, but rather it will route traffic from our web tier to the app tier.

Basic configuration

Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.

Scheme [Info](#)
Scheme can't be changed after the load balancer is created.
 Internet-facing
An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#)
 Internal
An internal load balancer routes requests from clients to targets using private IP addresses.

IP address type [Info](#)
Select the type of IP addresses that your subnets use.
 IPv4
Recommended for internal load balancers.
 Dualstack
Includes IPv4 and IPv6 addresses.

Select the correct network configuration for VPC and private subnets

Network mapping [Info](#)
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC [Info](#)
Select the virtual private cloud (VPC) for your targets or you can [create a new VPC](#). The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#).

Mappings [Info](#)
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.

<input checked="" type="checkbox"/> us-east-1a (use1-az1) Subnet <input type="text" value="subnet-0e65ab6a42801b273"/> app-tier-private-1a
IPv4 address Assigned from CIDR 10.0.16.0/20
<input checked="" type="checkbox"/> us-east-1b (use1-az2) Subnet <input type="text" value="subnet-0316cedb6b48869e3"/> app-tier-private-1b

Select the security group we created for this internal ALB. Now, this ALB will be listening for HTTP traffic on port 80. It will be forwarding the traffic to our **target group** that we just created, so select it from the dropdown, and create the load balancer

Security groups [Info](#)
A security group is a set of firewall rules that control the traffic to your load balancer. Select an existing security group, or you can [create a new security group](#).

Listeners and routing [Info](#)
A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

Listener HTTP:80	
Protocol	Port
HTTP	: 80 1-65535
Default action	Info Forward to: Select a target group
Create target group	

Launch Template

- Before we configure Auto Scaling, we need to create a Launch template with the AMI we created earlier. On the left side of the EC2 dashboard navigate to **Launch Template** under **Instances** and click **Create Launch Template**.

EC2 Dashboard	X	Launch Templates (2) Info	Actions ▾	Create launch template															
<input type="text" value="Search"/> < 1 > ⚙																			
		<table border="1"> <thead> <tr> <th>Launch Template ID</th> <th>Launch Template Name</th> <th>Default Version</th> <th>Latest Version</th> <th>Create Time</th> </tr> </thead> <tbody> <tr> <td>lt-0a6c47d84866420fa</td> <td>threetier-launch-template</td> <td>1</td> <td>2</td> <td>2024-02-06T04:56:24.0</td> </tr> <tr> <td>lt-0308c360ba66624a9</td> <td>webtier</td> <td>1</td> <td>1</td> <td>2024-02-06T07:29:22.0</td> </tr> </tbody> </table>	Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	lt-0a6c47d84866420fa	threetier-launch-template	1	2	2024-02-06T04:56:24.0	lt-0308c360ba66624a9	webtier	1	1	2024-02-06T07:29:22.0		
Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time															
lt-0a6c47d84866420fa	threetier-launch-template	1	2	2024-02-06T04:56:24.0															
lt-0308c360ba66624a9	webtier	1	1	2024-02-06T07:29:22.0															
		Select a launch template																	

2. Name the Launch Template, and then under **Application and OS Images** include the app tier AMI you created.

▼ Application and OS Images (Amazon Machine Image) [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

Recents | **My AMIs** | Quick Start

Don't include in launch template | Owned by me | Shared with me

[Browse more AMIs](#)
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

webtier-image
ami-031512dd8351d224d
2024-02-06T06:37:47.000Z Virtualization: hvm ENA enabled: true Root device type: ebs

Under **Instance Type** select t2.micro. For **Key pair** and **Network Settings** don't include it in the template. We don't need a key pair to access our instances and we'll be setting the network information in the autoscaling group.

▼ 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 SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.0716 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

▼ Network settings [Info](#)

Subnet [Info](#)

subnet-0417c59905b7fe8c0 private-db-subnet-az2
VPC: vpc-04a1ecf8ab7e6c3b Owner: 975049895916 Availability Zone: us-east-1b
IP addresses available: 248 CIDR: 10.0.6.0/24

Set the correct security group for our app tier, and then under **Advanced details** use the same IAM instance profile we have been using for our EC2 instances.

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

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 instance.

Select existing security group | Create security group

Common security groups [Info](#)
[Select security groups](#)

Compare security group rules

privateinstances-sg sg-0ccb5869f4229fc08 X
VPC: vpc-04a1ecf8ab7e6c3b

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

▼ Advanced details [Info](#)

IAM instance profile [Info](#)
demoEC2-role
arn:aws:iam:975049895916:instance-profile/demoEC2-role

Hostname type [Info](#)
 Don't include in launch template

DNS Hostname [Info](#)
 Enable resource-based IPv4 (A record) DNS requests
 Enable resource-based IPv6 (AAAA record) DNS requests

Auto Scaling

1. We will now create the Auto Scaling Group for our app instances. On the left side of the EC2 dashboard navigate to **Auto Scaling Groups** under **Auto Scaling** and click **Create Auto Scaling group**.

The screenshot shows the AWS EC2 Auto Scaling Groups page. On the left sidebar, under 'Auto Scaling', 'Auto Scaling Groups' is selected. The main area displays a table with columns: Name, Launch template/configuration, Instances, and Status. The table contains four rows:

- appwebtier: webtier | Version 1, 2 instances, -
- webapp: webtier | Version 1, 2 instances, -
- webtier: webtier-launchtemplate | Version 3, 0 instances, Updati...
- threetierapp-sg: threetier-launch-template | Version 2, 2 instances, -

A message at the bottom states '0 Auto Scaling groups selected'.

2. Give your Auto Scaling group a name, and then select the Launch Template we just created and click next.

The first screenshot shows the 'Choose launch template' step. It asks to specify a launch template for the new Auto Scaling group. A 'Name' field is present, and a dropdown menu shows 'apptierasg' selected. A note says: 'Must be unique to this account in the current Region and no more than 255 characters.'

The second screenshot shows the 'Launch template info' step. It displays a note: '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.' Below is a 'Launch template' dropdown set to 'webtier'.

3. On the **Choose instance launch options** page set your VPC, and the private instance subnets for the app tier and continue to step 3.

This screenshot shows the 'Choose instance launch options' page under the 'Network' tab. It includes sections for 'VPC' (selected VPC: 'vpc-0506ded4ddab6035e (three-tier-app)'), 'Availability Zones and subnets' (selected subnets: 'us-east-1a | subnet-0e63ab6a42801b273 (app-tier-private-1a)' and 'us-east-1b | subnet-0316cedb6b48869e3 (app-tier-private-1b)'), and a 'Create a subnet' button.

For this next step, attach this Auto Scaling Group to the Load Balancer we just created by selecting the existing load balancer's target group from the dropdown. Then, click next

This screenshot shows the 'Attach to an existing load balancer' step. It has three options:

- 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.'

Below is a section titled 'Attach to an existing load balancer' with a note: 'Select the load balancers that you want to attach to your Auto Scaling group.'

- For **Configure group size and scaling policies**, set desired, minimum and maximum capacity to **2**. Click skip to review and then Create Auto Scaling Group.

The screenshot shows the 'Group size' and 'Scaling' sections of the AWS Auto Scaling Group configuration interface. In the 'Group size' section, 'Desired capacity type' is set to 'Units (number of instances)' with a value of '2'. In the 'Scaling' section, 'Scaling limits' are configured with 'Min desired capacity' at '2' and 'Max desired capacity' also at '2', both set to 'Equal or less than'.

You should now have your internal load balancer and autoscaling group configured correctly. You should see the autoscaling group spinning up 2 new app tier instances. If you wanted to test if this is working correctly, you can delete one of your new instances manually and wait to see if a new instance is booted up to replace it.

NOTE: Your original app tier instance is excluded from the ASG so you will see 3 instances in the EC2 dashboard. You can delete your original instance that you used to generate the app tier AMI but it's recommended to keep it around for troubleshooting purposes.

Part 5: Web Tier Instance Deployment

In this section we will deploy an EC2 instance for the web tier and make all necessary software configurations for the NGINX web server and React.js website.

Learning Objectives

- Update NGINX Configuration Files
- Create Web Tier Instance
- Configure Software Stack

Update Config File

Before we create and configure the web instances, open up the **application-code/nginx.conf** file from the repo we downloaded. Scroll down to **line 58** and replace [INTERNAL-LOADBALANCER-DNS] with your internal load balancer's DNS entry. You can find this by navigating to your internal load balancer's details page.

```
#proxy for internal lb
location /api/{
    proxy_pass http://internal-internallb-209461046.us-east-1.elb.amazonaws.com:80;
}

ings for a TLS enabled server.

server {
    listen      443 ssl http2;
    listen      [::]:443 ssl http2;
    server_name _;
    root        /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
}
```

Then, upload this file **and the application-code/web-tier** folder to the s3 bucket you created for this lab.

Web Instance Deployment

- Follow the same instance creation instructions we used for the App Tier instance in **Part 3: App Tier Instance Deployment**, with the exception of the subnet. We will be provisioning this instance in one of our **public subnets**. Make sure to select the correct network components, security group, and IAM role. **This time, auto-assign a public ip** on the **Configure Instance Details** page. Remember to tag the **instance with a name so we can identify it more easily**.

The screenshot shows two side-by-side AWS CloudFormation instance details pages. The left panel displays the 'Instance details' for an instance named 'i-00e85beb0c592beec (webtier)'. It lists various configuration parameters such as AMI ID, Platform (Amazon Linux), Stop protection (Disabled), Instance auto-recovery (Default), AMI Launch index (0), Credit specification (standard), Usage operation (RunInstances), and Endorse (Inferred). The right panel shows the 'Instance summary' for the same instance, providing a high-level overview of its state, VPC, and network settings. Both panels include a 'Connect' button at the top right.

Connect to Instance

- Follow the same steps you used to connect to the app instance and change the user to **ec2-user**. Test connectivity here via ping as well since this instance should have internet connectivity:

sudo -su ec2-user

ping 8.8.8.8

Note: If you don't see a transfer of packets then you'll need to verify your route tables attached to the subnet that your instance is deployed in.

Configure Web Instance

1. We now need to install all of the necessary components needed to run our front-end application. Again, start by installing NVM and node :

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash  
source ~/.bashrc  
nvm install 16  
nvm use 16
```

2. Now we need to download our web tier code from our s3 bucket:

```
cd ~/  
aws s3 cp s3://BUCKET_NAME/web-tier/ web-tier --recursive
```

Navigate to the web-layer folder and create the build folder for the react app so we can serve our code:

```
cd ~/web-tier  
npm install  
npm run build
```

3. NGINX can be used for different use cases like load balancing, content caching etc, but we will be using it as a web server that we will configure to serve our application on port 80, as well as help direct our API calls to the internal load balancer.

```
sudo amazon-linux-extras install nginx1 -y
```

```
sudo yum install nginx
```

4. We will now have to configure NGINX. Navigate to the Nginx configuration file with the following commands and list the files in the directory:

```
cd /etc/nginx
```

5. You should see an nginx.conf file. We're going to copy this file and use the one we uploaded to s3. Replace the bucket name in the command below with the one you created for this workshop:

```
cp nginx.conf nginx.conf_bkp
```

```
sudo cp nginx.conf nginx.conf_bkp
```

```
aws s3 cp s3://three-app-tier-1/nginx.conf .
```

```
sudo aws s3 cp s3://three-app-tier-1/nginx.conf .
```

Then, restart Nginx with the following command:

```
sudo service nginx restart
```

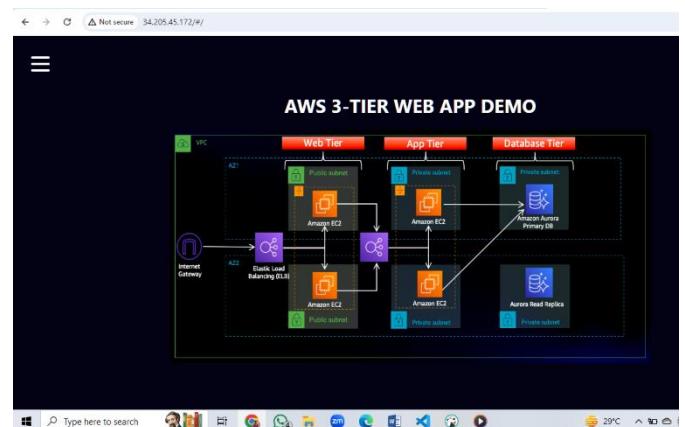
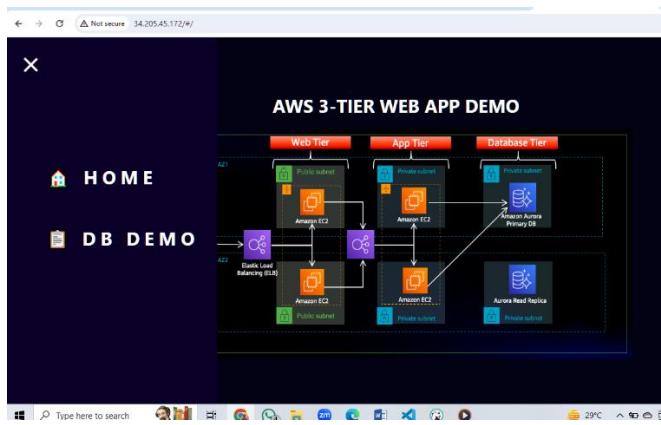
To make sure Nginx has permission to access our files execute this command:

```
chmod -R 755 /home/ec2-user
```

And then to make sure the service starts on boot, run this command:

```
sudo chkconfig nginx on
```

5. Now when you plug in the **public IP** of your web tier instance, you should see your website, which you can find on the Instance details page on the EC2 dashboard. If you have the database connected and working correctly, then you will also see the database working. You'll be able to add data. Careful with the delete button, that will clear all the entries in your database.



The screenshot shows a "AURORA DATABASE DEMO PAGE". At the top, there is a header with a menu icon and a "DEL" button. Below the header is a table with three columns: ID, AMOUNT, and DESC. There is a single row in the table with an "ADD" button in the ID column. The table has a blue border and a white background.

ID	AMOUNT	DESC
ADD		

Part 6: External Load Balancer and Auto Scaling

In this section of the workshop we will create an Amazon Machine Image (AMI) of the web tier instance we just created, and use that to set up autoscaling with an external facing load balancer in order to make this tier highly available.

Learning Objectives:

- Create an AMI of our Web Tier
- Create a Launch Template
- Configure Auto Scaling
- Deploy External Load Balancer

Web Tier AMI

1. Navigate to **Instances** on the left hand side of the EC2 dashboard. Select the web tier instance we created and under **Actions** select **Image and templates**. Click **Create Image**.
2. we created and under **Actions** select **Image and templates**. Click **Create Image**.

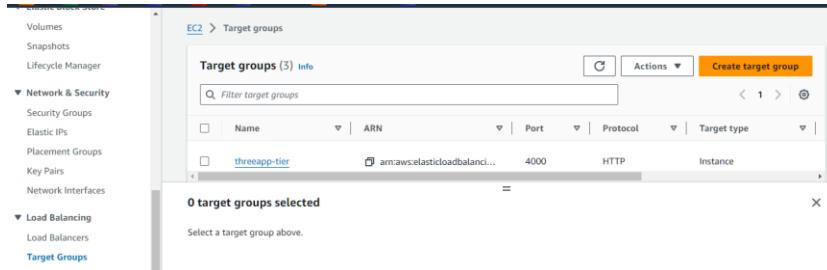
The screenshot shows the AWS EC2 Instances page with 10 instances listed. One instance, 'threapptier', is selected. The 'Actions' dropdown menu is open, and the 'Image and templates' option is highlighted. Other options in the dropdown include 'View details', 'Manage instance state', 'Instance settings', 'Networking', 'Security', 'Monitor and troubleshoot', and 'View alarms'. The 'Launch instances' button is also visible in the top right of the Actions menu.

3. Give the image a name and description and then click **Create image**. This will take a few minutes, but if you want to monitor the status of image creation you can see it by clicking **AMIs** under **Images** on the left hand navigation panel of the EC2 dashboard.

The screenshot shows the 'Create image' dialog box. It includes fields for 'Instance ID' (set to 'i-06630b81d70ee6970'), 'Image name' (an empty input field with a note: 'Enter image name. Maximum 127 characters. Can't be modified after creation.'), 'Image description - optional' (an empty input field with a note: 'Maximum 255 characters'), and checkboxes for 'No reboot' and 'Reboot'.

Target Group

2. While the AMI is being created, we can go ahead and create our target group to use with the load balancer. On the EC2 dashboard navigate to **Target Groups** under **Load Balancing** on the left hand side. Click on **Create Target Group**.



3. The purpose of forming this target group is to use with our load balancer so it may balance traffic across our private app tier instances. Select Instances as the target type and give it a name.

A screenshot of the 'Basic configuration' step of creating a target group. It shows two tabs: 'Instances' (selected) and 'IP address'. Under 'Instances', it says 'Supports load balancing to instances within a specific VPC' and 'Facilitates the use of Amazon EC2 Auto Scaling'. Under 'Protocol : Port', it shows 'HTTP' and '80'. Under 'IP address type', it shows 'IPv4' selected with the note 'Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.' There is also an 'IPv6' option.

Then, set the protocol to **HTTP** and the port to **4000**. Remember that this is the port our Node.js app is running on. Select the VPC we've been using thus far, and then change the health check path to be **/health**. This is the health check endpoint of our app. Click **Next**.

A screenshot of the 'Protocol' step of creating a target group. It shows 'Protocol : Port' set to 'HTTP' and 'Port' set to '4000'. Under 'IP address type', 'IPv4' is selected. Under 'VPC', 'three-tier-app' is selected. Under 'Protocol version', 'HTTP1' is selected.

4. We are **NOT** going to register any targets for now, so just skip that step and create the target group.

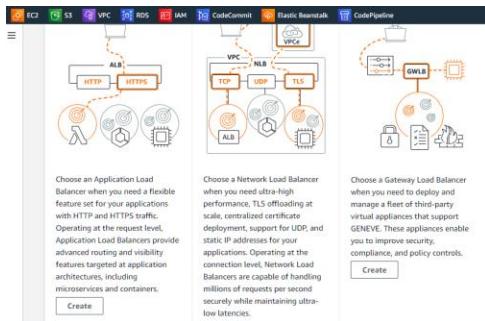
Two screenshots of the 'Review targets' step. The left screenshot shows 'Available instances (6)' with a table of six instances. The right screenshot shows 'Targets (0)' with a table and notes: 'No instances added yet' and 'Specify instances above, or leave the group empty if you prefer to add targets later.' At the bottom, there are buttons for '0 pending', 'Cancel', 'Previous', and 'Create target group'.

Internal Load Balancer:

2. On the left hand side of the EC2 dashboard select **Load Balancers** under **Load Balancing** and click **Create Load Balancer**.

The screenshot shows the AWS Lambda dashboard with the sidebar expanded to show 'Network & Security' and 'Load Balancing'. Under 'Load Balancing', the 'Load Balancers' option is selected. The main area displays a table titled 'Load balancers (3)'. The table has columns for Name, DNS name, State, VPC ID, and Availability Zones. One row is visible, labeled 'internal-lb' with 'internal-internallb-20016' in the DNS name column, 'Active' in the State column, and 'vpc-0506ed0d6b6035e' in the VPC ID column. A message at the bottom says '0 load balancers selected' and 'Select a load balancer above.'

3. We'll be using an **Application Load Balancer** for our **HTTP** traffic so click the create button for that option.



4. After giving the load balancer a name, be sure to select **internal** since this one will not be public facing, but rather it will route traffic from our web tier to the app tier.

This screenshot shows the 'Basic configuration' step of the wizard. It includes fields for 'Load balancer name' (set to 'app-tier-internal-lb'), 'Scheme' (radio buttons for 'Internet-facing' and 'Internal' - the latter is selected), and 'IP address type' (radio buttons for 'IPv4' - selected, 'IPv6', and 'Dualstack'). Below these are sections for 'Network mapping' and 'VPC' configuration.

Select the correct network configuration for VPC and private subnets

This screenshot shows the 'Network mapping' step. It lists 'three-tier-app' as the VPC and shows two subnets: 'subnet-0e65ab6a42801b273' (app-tier-private-1a) and 'subnet-0316cedb6b48869e3' (app-tier-private-1b). It also shows mappings for 'us-east-1a (use1-az1)' and 'us-east-1b (use1-az2)'.

Select the security group we created for this internal ALB. Now, this ALB will be listening for HTTP traffic on port 80. It will be forwarding the traffic to our **target group** that we just created, so select it from the dropdown, and create the load balancer

The screenshot shows the 'Listeners and routing' section of the Load Balancer configuration. A new listener is being created for port 80, using the 'HTTP' protocol. The 'Default action' dropdown is set to 'Select a target group', which is currently empty. A 'Create target group' button is visible below the dropdown.

Launch Template

- Before we configure Auto Scaling, we need to create a Launch template with the AMI we created earlier. On the left side of the EC2 dashboard navigate to **Launch Template** under **Instances** and click **Create Launch Template**.

The screenshot shows the 'Launch Templates' page with two entries listed:

Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time
lt-0a6c47d84866420fa	threetier-launch-template	1	2	2024-02-06T04:56:24.0
lt-0308c360ba66624a9	webtier	1	1	2024-02-06T07:29:22.0

- Name the Launch Template, and then under **Application and OS Images** include the app tier AMI you created.

The screenshot shows the 'My AMIs' tab selected in the 'Application and OS Images' interface. A search bar at the top is empty. Below it, there are three filter options: 'Don't include in launch template', 'Owned by me' (which is selected), and 'Shared with me'. To the right, there is a search icon and a link to 'Browse more AMIs'. At the bottom, a specific AMI is highlighted:

Amazon Machine Image (AMI)

webtier-image
ami-031512dd8351d224d
2024-02-06T06:37:47.000Z Virtualization: hvm ENA enabled: true Root device type: ebs

Under **Instance Type** select t2.micro. For **Key pair** and **Network Settings** don't include it in the template. We don't need a key pair to access our instances and we'll be setting the network information in the autoscaling group.

The screenshot shows two sections of the AWS EC2 instance configuration:

- Instance type:** Shows the selected instance type as t2.micro, which is free tier eligible. It lists various On-Demand and On-Demand SUSE base pricing options per hour.
- Network settings:** Shows a subnet named subnet-0417c59905b7fe8c0, which is part of the private-db-subnet-az2 VPC. It includes details like VPC ID, owner, availability zone, and IP addresses available.

Set the correct security group for our app tier, and then under **Advanced details** use the same IAM instance profile we have been using for our EC2 instances.

The screenshot shows the security group configuration for the app tier:

- Firewall (security groups):** Shows a "Select existing security group" dropdown with "Select security groups" and a "Create security group" button.
- Common security groups:** Shows a dropdown menu with "privateinstances-sg_sg-0bcb5869f4229fc08" selected, along with a "Compare sec group rules" link.
- Security groups:** A list of security groups that will be added or removed from the interface.

The screenshot shows the advanced details configuration:

- IAM instance profile:** Set to "demoEC2-role" (arn:aws:iam::975049895916:instance-profile/demoEC2-role).
- Hostname type:** Set to "Don't include in launch template".
- DNS Hostname:** Options for enabling resource-based IPv4 and IPv6 DNS requests.

Auto Scaling

2. We will now create the Auto Scaling Group for our app instances. On the left side of the EC2 dashboard navigate to **Auto Scaling Groups** under **Auto Scaling** and click **Create Auto Scaling group**.

The screenshot shows the list of existing Auto Scaling groups:

Name	Launch template/configuration	Instances	Status
appwebtier	webtier Version 1	2	-
webapp	webtier Version 1	2	-
webtier	webtier-launch-template Version 3	0	Updating
threetierapp-sg	threetier-launch-template Version 2	2	-

3. Give your Auto Scaling group a name, and then select the Launch Template we just created and click next.

The screenshot shows the "Choose launch template" step:

- Name:** Input field for the Auto Scaling group name, containing "apptierasg".
- Auto Scaling group name:** Description and input field for the group name.
- Must be unique to this account in the current Region and no more than 255 characters.**

The screenshot shows the "Launch template" step:

- Launch template:** Description and input field for selecting a launch template, currently set to "webtier".
- 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.**
- Create a launch template:** Link to create a new launch template.

5. On the Choose instance launch options page set your VPC, and the public instance subnets for the web tier and continue to step 3.

The screenshot shows the 'Network' tab of the 'Choose instance launch options' page. Under 'VPC', a dropdown menu is set to 'vpc-0506ded4ddab6035e (three-tier-app)' with the IP range '10.0.0.0/16'. A 'Create a VPC' button is available. Under 'Availability Zones and subnets', two subnets are selected: 'us-east-1a | subnet-0e63ab6a42801b273 (app-tier-private-1a)' with IP range '10.0.16.0/20' and 'us-east-1b | subnet-0316cedb6b48869e3 (app-tier-private-1b)' with IP range '10.0.32.0/20'. A 'Create a subnet' button is also present.

For this next step, attach this Auto Scaling Group to the Load Balancer we just created by selecting the existing load balancer's target group from the dropdown. Then, click next

The screenshot shows the 'Attach to an existing load balancer' section. It includes three options: 'No load balancer' (disabled), 'Attach to an existing load balancer' (selected), and 'Attach to a new load balancer'. Below this, under 'Attach to an existing load balancer', there are two choices: 'Choose from your load balancer target groups' (selected) and 'Choose from Classic Load Balancers'. A dropdown menu 'Select target groups' lists 'threapp-tier | HTTP Application Load Balancer: internallb'.

6. For Configure group size and scaling policies, set desired, minimum and maximum capacity to 2. Click skip to review and then Create Auto Scaling Group.

The screenshot shows the 'Group size' and 'Scaling' sections. In 'Group size', the 'Desired capacity type' is set to 'Units (number of instances)' with a value of '2'. In 'Scaling', the 'Scaling limits' are defined with 'Min desired capacity' at '2' and 'Max desired capacity' at '2', both labeled 'Equal or less than'.

You should now have your external load balancer and autoscaling group configured correctly. You should see the autoscaling group spinning up 2 new web tier instances. If you wanted to test if this is working correctly, you can delete one of your new instances manually and wait to see if a new instance is booted up to replace it. To test if your entire architecture is working, navigate to your external facing loadbalancer, and plug in the DNS name into your browser.

The first screenshot shows the 'AWS 3-TIER WEB APP DEMO' interface. It features a navigation bar with 'HOME' and 'DB DEMO' options. Below the navigation bar is a detailed architectural diagram titled 'AWS 3-TIER WEB APP DEMO'. The diagram is divided into three horizontal tiers: 'Web Tier', 'App Tier', and 'Database Tier'. The 'Web Tier' contains an 'Amazon EC2' instance in a 'Public subnet'. The 'App Tier' contains two 'Amazon EC2' instances in a 'Private subnet'. The 'Database Tier' contains an 'Amazon Aurora Primary DB' and an 'Aurora Read Replica' in a 'Private subnet'. Arrows indicate the flow of data between these components. The second screenshot shows the same interface after an instance has been deleted from the App Tier, resulting in only one instance remaining. The third screenshot shows the 'AURORA DATABASE DEMO PAGE' with a table and a form for managing data.

NOTE: Again, your original web tier instance is excluded from the ASG so you will see 3 instances in the EC2 dashboard. You can delete your original instance that you used to generate the web tier AMI but it's recommended to keep it around for troubleshooting purposes.

Congrats! You've Implemented a 3 Tier Web Architecture!

Appendix

DOCUMENTATION:

- NVM (Node Version Manager)
 - <https://github.com/nvm-sh/nvm>
- Node.js
 - <https://nodejs.org>
- PM2
 - <https://pm2.keymetrics.io/docs/usage/pm2-doc-single-page/>
- NGINX
 - https://docs.nginx.com/?_ga=2.4975283.1227355032.1619285306-501395053.1619109877
- Amazon Linux Extras
 - <https://aws.amazon.com/amazon-linux-2/faqs/>